aws

開發人員指南

Amazon Cognito

Amazon Cognito: 開發人員指南

Table of Contents

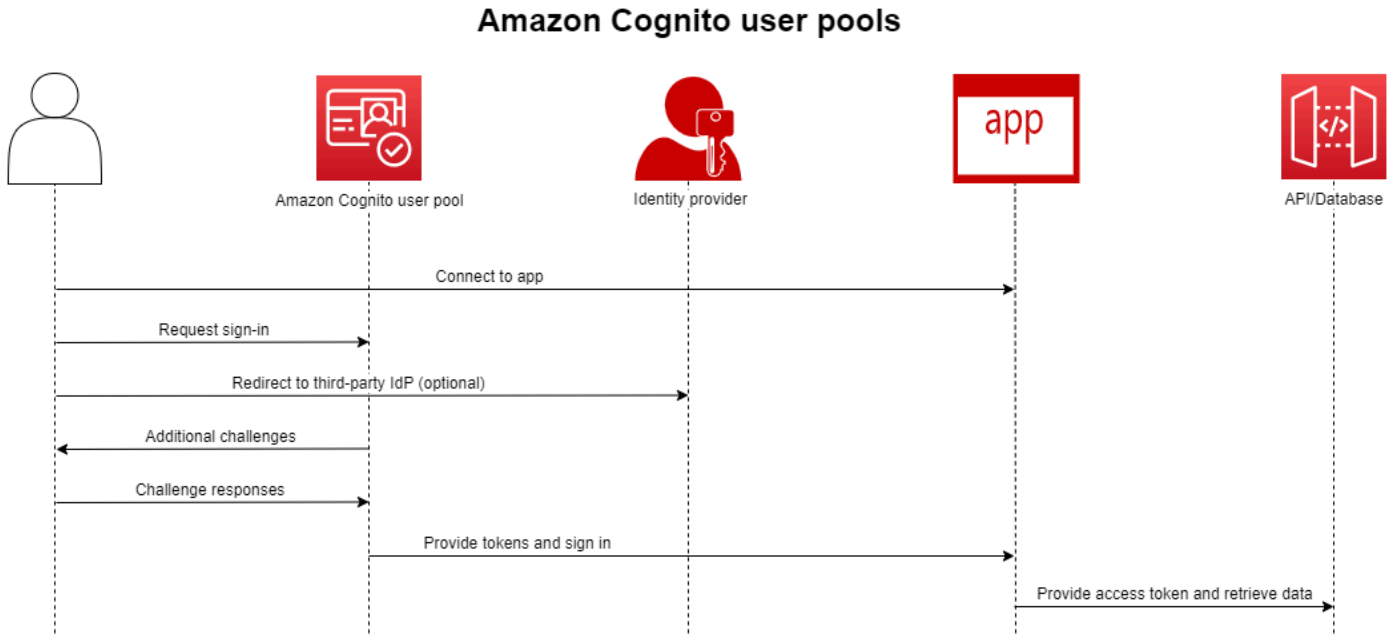
什麼是 Amazon Cognito？

Amazon Cognito 是適用於 Web 和行動應用程式的身分平台。是一種使用者目錄、身分驗證伺服器，以及 OAuth 2.0 存取權杖和 AWS 憑證的授權服務。使用 Amazon Cognito，您可以從內建的使用者目錄、企業目錄以及 Google 和 Facebook 等消費者身分提供者進行驗證和授權使用者。

主題

接下來兩個元件組成 Amazon Cognito。兩個元件會根據您對使用者的存取需求獨立或一併運作。

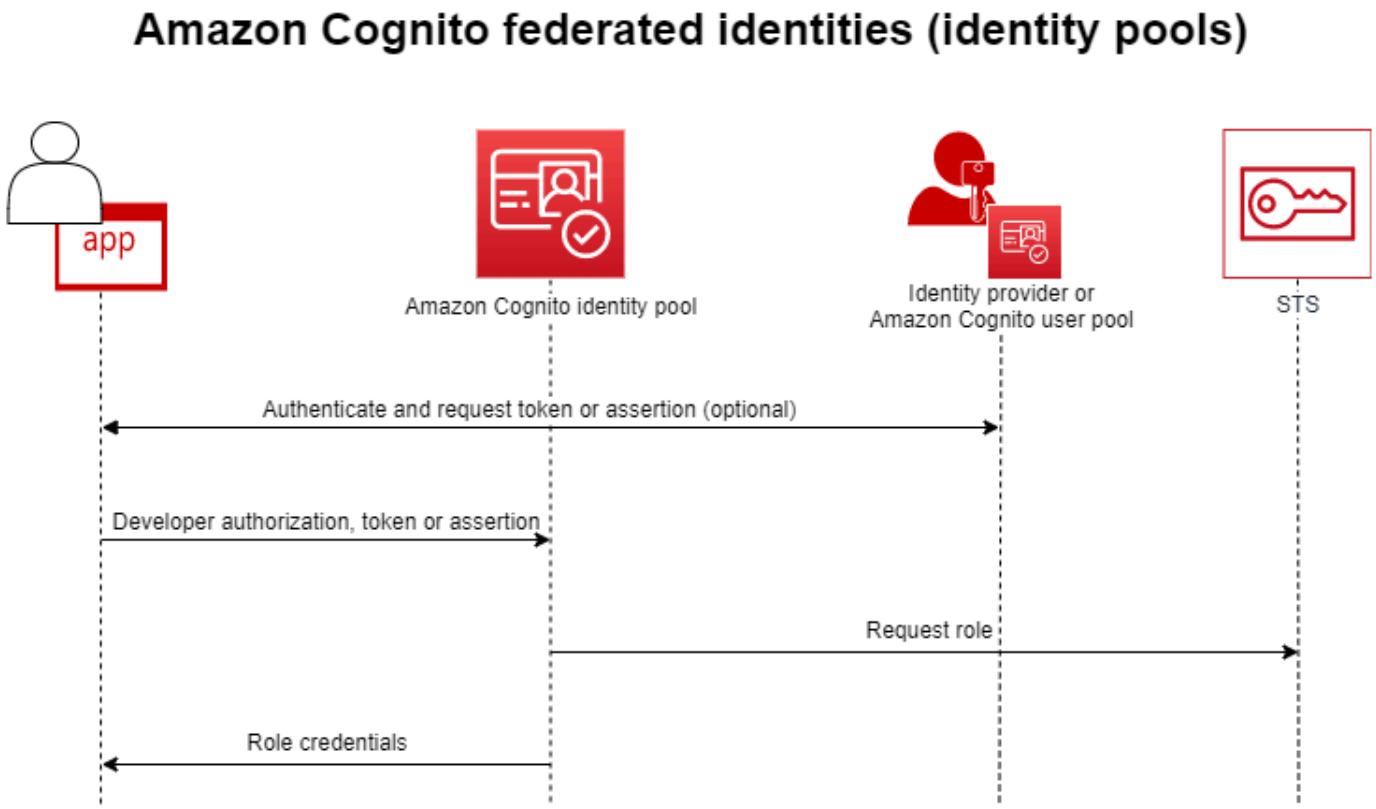
使用者集區



當您要對應用程式或 API 進行驗證和使用者授權時，請建立使用者集區。使用者集區是具有自助服務和管理員驅動的使用者建立、管理和驗證的使用者目錄。您的使用者集區可以是獨立的目錄和 OIDC 身分提供者 (IdP)，也可以是第三方員工和客戶身分提供者的中繼服務提供者 (SP)。您可以在應用程式中為組織的 SAML 2.0 和 OIDC 中的員工身分識別提供單一登入 (SSO) 以及 IdPs 具有使用者集區的 OIDC。針對您組織在公開 OAuth 2.0 身分儲存 Amazon、Google、Apple 和 Facebook 中的客戶身分，您也可以在應用程式提供 SSO。如需客戶身份與存取管理 (CIAM) 的更多資訊，請參閱 CIAM 是什麼？。

使用者集區不需要與身分池整合。在使用者集區中，您可以將經過驗證的 JSON Web 權杖 (JWT) 直接發布至應用程式、Web 伺服器或 API。

身分集區



當您想要授權已驗證或匿名使用者存取您的 AWS 資源時，請設定 Amazon Cognito 身分集區。身分集區會為您的應用程式發出 AWS 認證，以便為使用者提供資源。您可以使用受信任的身分提供者 (例如，使用者集區或 SAML 2.0 服務) 來驗證使用者。也可以選擇性地為訪客使用者發布憑證。身分識別集區使用角色型和屬性型存取控制，來管理使用者存取資源的授權。 AWS

身分池不需要與使用者集區整合。身分池可以直接從員工和消費者身分提供者接受經過驗證的宣告。

一併使用的 Amazon Cognito 使用者集區與身分池

在本主題開始的圖表中，您可以使用 Amazon Cognito 驗證您的使用者，然後授與他們存取 AWS 服務。

1. 您的應用程式使用者會透過使用者集區登入，並收到 OAuth 2.0 權杖。
2. 您的應用程序將用戶池令牌與身份池交換為可與 AWS API 和 AWS Command Line Interface （AWS CLI）一起使用的臨時 AWS 憑據。
3. 您的應用程式會將登入資料工作階段指派給您的使用者，並提供授權存取， AWS 服務 例如 Amazon S3 和 Amazon DynamoDB。

如需使用身分池和使用者集區的更多範例，請參閱 Amazon Cognito 常見案例。

Amazon Cognito，共同責任模型的雲端安全性義務符合 SOC 1-3、PCI DSS、ISO 27001，並具有 HIPAA-BAA 資格。您可以在 Amazon Cognito 設計雲端安全性，以符合 SOC1-3、27001、HIPAA-BAA 規範，但不具有 PCI DSS。如需詳細資訊，請參閱 AWS 服務範圍。請同時參閱區域資料考量。

Amazon Cognito 功能

使用者集區

Amazon Cognito 使用者集區是一種使用者目錄。透過使用者集區，您的使用者就可以透過 Amazon Cognito 登入 Web 或行動應用程式，或者透過第三方 IdP 以聯合身分登入。聯合和本機使用者在使用者集區中有一個使用者設定檔。

本機使用者是直接在使用者集區中註冊或建立的使用者。您可以在 AWS SDK 或 AWS Command Line Interface (AWS CLI) 中管理和自訂這些使用者設定檔。 AWS Management Console

Amazon Cognito 使用者集區接受來自第三方的權杖和宣告 IdPs，並將使用者屬性收集到其發出給您的應用程式的 JWT 中。您可以在一組 JWT 上標準化應用程式，同時 Amazon Cognito 處理與之互動 IdPs，將其宣告對應至中央權杖格式。

Amazon Cognito 使用者集區可以是獨立的 IdP。Amazon Cognito 從 OpenID Connect (OIDC) 標準中提取，產生用於身分驗證和授權的 JWT。當您登入本機使用者時，您的使用者集區對這些使用者具有權威性。當您驗證本機使用者時，可以存取下列功能。

- 實作自己的網頁前端，該前端會呼叫 Amazon Cognito 使用者集區 API 來驗證、授權和管理您的使用者。
- 為使用者設定多重要素驗證 (MFA) Amazon Cognito 支援以時間為基礎的一次性密碼 (TOTP) 和 SMS 訊息 MFA。
- 防止受到惡意控制的使用者帳戶存取。
- 建立您自己的自訂多步驟身份驗證流程。
- 查詢其他目錄中的使用者，然後將其遷移到 Amazon Cognito。

Amazon Cognito 使用者集區也可以對您的服務提供者 (SP) 以及您 IdPs的應用程式執行 IdP 的雙重角色。Amazon Cognito 用戶池可以連接到消費者， IdPs 如 Facebook 和谷歌，或者 IdPs 像 Okta 和活動目錄聯合服務（ ADFS ）這樣的勞動力。

使用 Amazon Cognito 使用者集區發布的 OAuth 2.0 和 OpenID Connect (OIDC) 權杖，您可以

- 在您的應用程式中接受對使用者進行身分驗證的 ID 權杖，並提供設定使用者設定檔所需的資訊
- 在您的 API 中接受具有授權使用者 API 呼叫 OIDC 範圍的存取權杖。
- 從 Amazon Cognito 身分集區擷取 AWS 登入資料。

Amazon Cognito 使用者集區功能

功能	描述
OIDC IdP	發行 ID 令牌以驗證用戶
授權伺服器	發出存取權杖以授權使用者存取 API
薩姆爾 2.0 SP	將 SAML 判斷提示轉換為 ID 和存取權杖
發動機	將 OIDC 令牌轉換為 ID 和訪問令牌
驗證 2.0 SP	將蘋果，Facebook，Amazon 或谷歌的 ID 令牌轉換為您自己的 ID 和訪問令牌
驗證前端服務	使用託管 UI 註冊、管理和驗證使用者
為您自己的 UI 提供 API 支援	透過支援的 AWS SDK 中的 API 要求建立、管理及驗證使用者 [1]
MFA	使用 SMS 訊息、TOTP 或使用者的裝置作為額外的驗證因素 [1]
安全監控與回應	抵禦惡意活動和不安全的密碼 [1]
自訂驗證流程	建置您自己的驗證機制，或新增自訂步驟至現有流程 [1]
群組	建立使用者的邏輯群組，以及將權杖傳遞至身分集區時的 IAM 角色宣告階層
自定義 ID 令牌	使用新的，修改和隱藏的聲明自定義 ID 令牌
自訂使用者屬性	將值指派給使用者屬性，並新增您自己的自訂屬性

[1] 功能僅適用於本機使用者。

如需使用者集區的詳細資訊，請參閱 使用者集區入門 和 Amazon Cognito 使用者集區 API 參考。

身分集區

身分集區是指派給使用者或來賓並授權接收臨時 AWS 登入資料的唯一識別碼或身分識別的集合。當您以來自 SAML 2.0、OpenID Connect (OIDC) 或 OAuth 2.0 社交身分提供者 (IdP) 的受信任宣告形式向身分池提供驗證時，表示您將您的使用者與身分池中的身分建立關聯。您的身份池為身份創建的令牌可以從 AWS Security Token Service （ AWS STS ）檢索臨時會話憑據。

若要補充已驗證的身分識別，您也可以設定身分集區以在沒有 IdP 驗證的情況下授權 AWS 存取。您可以提供自己自訂的身分驗證，也可以不提供身分驗證。您可以將臨時 AWS 憑據授予任何請求它們的應用程序用戶，具有未經身份驗證的身份。身分池也會接受宣告，並使用開發人員驗證身分，根據您自己的自訂結構描述發布憑證。

使用 Amazon Cognito 身分池，您可以透過兩種方式與您 AWS 帳戶的 IAM 政策整合。您可以一起使用或單獨使用這兩項功能。

角色類型存取控制

當您的使用者將宣告傳遞至您的身分池時，Amazon Cognito 會選擇其請求的 IAM 角色。若要根據您的需求自訂角色許可，請將 IAM 政策套用至每個角色。例如，如果您的使用者證明他們位於行銷部門，則他們會收到角色憑證，其中包含針對行銷部門存取需求而量身打造的原則。Amazon Cognito 可以請求預設角色、根據查詢使用者宣告的規則來請求角色，或根據使用者集區中使用者群組成員資格請求角色。您也可以設定角色信任政策，讓 IAM 僅信任您的身分池來產生臨時工作階段。

存取控制的屬性

您的身分池會從使用者的宣告中讀取屬性，並將其映射至使用者臨時工作階段中的主要標籤。然後，您可以設定以 IAM 資源為基礎的政策，以從身分池中具有工作階段標籤的 IAM 主體允許或拒絕存取資源。例如，如果您的使用者證明他們位於行銷部門，請 AWS STS 標記他們的工作階段`Department: marketing`。您的 Amazon S3 儲存貯體允許根據 aws：PrincipalTag條件需要`Department`標籤值`marketing`為的讀取操作。

Amazon Cognito 身分池功能

功能	描述
Amazon Cognito 用戶池 SP	從您的用戶池交換 ID 令牌以獲取 Web 身份憑據 AWS STS

薩姆爾 2.0 SP	從以下位置交換 SAML 判斷提示以取得 Web 身分認證 AWS STS
發動機	從以下位置交換 OIDC 令牌以取得網路身分認證 AWS STS
驗證 2.0 SP	從 Amazon，臉譜，谷歌，蘋果和推特交換 OAuth 令牌以獲取網絡身份憑據 AWS STS
自定義 SP	使用 AWS 憑據，將任何格式的聲明交換為 Web 身份憑據 AWS STS
未驗證的存取	在 AWS STS 沒有驗證的情況下發出有限訪問的 Web 身份
角色類型存取控制	根據已驗證使用者的宣告選擇 IAM 角色，並將您的角色設定為僅在身分集區的內容中擔任
屬性型存取控制	將宣告轉換為 AWS STS 臨時工作階段的主要標籤，並使用 IAM 政策根據主要標籤篩選資源存取

如需身分集區的詳細資訊，請參閱 開始使用 Amazon Cognito 身分集區 和 Amazon Cognito 身分集區 API 參考。

Amazon Cognito 使用者集區和身分池比較

功能	描述	使用者集區	身分集區
OIDC IdP	發出 OIDC ID 令牌以驗證應用程序用戶	✓	
API 授權伺服器	發出訪問令牌以授權用戶訪問接受 OAuth 2.0 授權範圍的 API，數據庫和其他資源	✓	

IAM 網絡身份授權服務器	生成可用於交換臨時 AWS 憑據 AWS STS 的令牌		✓
薩姆爾 2.0 SP & OIDC IdP	根據來自 SAML 2.0 IdP 的聲明發行自定義的 OIDC 令牌	✓	
OIDC SP & OIDC IdP	根據來自 OIDC IdP 的聲明發行自定義的 OIDC 令牌	✓	
OAuth 2.0 SP 和 OIDC IdP	根據來自 OAuth 2.0 社交提供商（如蘋果和谷歌）的範圍發行自定義的 OIDC 令牌	✓	
SAML 2.0 SP 和憑證經紀人	根據來自 SAML 2.0 IdP 的宣告發出臨時 AWS 登入資料		✓
OIDC SP 和憑證代理商	根據來自 OIDC IdP 的宣告發出臨時 AWS 登入資料		✓
OAuth 2.0 SP 和憑據代理人	根據來自 OAuth 2.0 社交提供商（如蘋果和谷歌）的範圍發出臨時 AWS 憑據		✓
Amazon Cognito 用戶池 SP 和憑據代理人	根據來自 Amazon Cognito 使用 AWS 者集區的 OIDC 宣告發出臨時登入資料		✓
自定義 SP 和憑據代理	根據開發人員 IAM 授權發出臨時 AWS 登入資料		✓

驗證前端服務	使用託管 UI 註冊、管理和驗證使用者	✓	
API 支援您自己的驗證 UI	透過支援的 AWS SDK 中的 API 要求建立、管理及驗證使用者 [1]	✓	
MFA	使用 SMS 訊息、TOTP 或使用者的裝置作為額外的驗證因素 [1]	✓	
安全監控與回應	防範惡意活動和不安全的密碼 [1]	✓	
自訂驗證流程	建置您自己的驗證機制，或新增自訂步驟至現有流程 [1]	✓	
群組	建立使用者的邏輯群組，以及將權杖傳遞至身分集區時的 IAM 角色宣告階層	✓	
自定義 ID 令牌	使用新的，修改和隱藏的聲明自定義 ID 令牌	✓	
AWS WAF 網路 ACL	使用以下方式監視和控制驗證環境的要求 AWS WAF	✓	
自訂使用者屬性	將值指派給使用者屬性，並新增您自己的自訂屬性	✓	
未驗證的存取	在 AWS STS 沒有驗證的情況下發出有限訪問的 Web 身份		✓

角色類型存取控制	根據已驗證使用者的宣告選擇 IAM 角色，並將您的角色設定為僅在身分集區的內容中擔任	✓
屬性型存取控制	將使用者宣告轉換為 AWS STS 臨時工作階段的主要標籤，並使用 IAM 政策根據主要標籤篩選資源存取	✓

[1] 功能僅適用於本機使用者。

Amazon Cognito 入門

例如，使用者集區應用程式，請參閱使用者集區入門。

如需身分集區的簡介，請參閱開始使用 Amazon Cognito 身分集區。

如需使用者集區和身分識別集區的引導式設定體驗的連結，請參閱Amazon Cognito 的引導式設定選項。

如需影片、文章、文件和更多範例應用程式，請參閱 Amazon Cognito 開發人員資源。

若要使用 Amazon Cognito，您需要一個 AWS 帳戶。如需詳細資訊，請參閱 開始使用 AWS。

區域可用性

Amazon Cognito 可在全球多個 AWS 區域使用。在每個區域中，Amazon Cognito 都會分佈在多個可用區域。這些可用區域各自實體隔離，但以私有、低延遲、高輸送量、高度冗餘的網路連線加以整合。這些可用區域能 AWS 夠提供包括 Amazon Cognito 在內的服務，並具有非常高的可用性和備援，同時將延遲降至最低。

如需可使用 Amazon Cognito 的所有區域清單，請參閱 Amazon Web Services 一般參考 中的 AWS 區域與端點。如需進一步了解各區域之可用區域數量的資訊，請參閱 AWS 全球基礎設施。

Amazon Cognito 定價

如需 Amazon Cognito 定價的相關資訊，請參閱 Amazon Cognito 定價。

身份驗證如何與 Amazon Cognito 使用者集區和身分集區搭配使用

當您的客戶登入 Amazon Cognito 使用者集區時，您的應用程式會收到 JSON 網頁權杖 (JWT)。

當您的客戶使用使用者集區權杖或其他提供者登入身分集區時，您的應用程式會收到臨時 AWS 認證。

通過用戶池登錄，您可以完全使用 AWS SDK 實現身份驗證和授權。如果您不想建立自己的使用者介面 (UI) 元件，可以叫用預先建置的 Web UI (託管 UI) 或第三方身分識別提供者 (IdP) 的登入頁面。

本主題概述您的應用程式可與 Amazon Cognito 互動以使用 ID 權杖進行驗證、使用存取權杖授權，以及使用身分集區登入資料進行存取 AWS 服務 的一些方式。

主題

- 使用 AWS SDK 進行使用者集區 API 驗證和授權
- 使用託管 UI 進行使用者集區驗證
- 使用第三方身分識別提供者進行使用者
- 身分集區驗證

使用 AWS SDK 進行使用者集區 API 驗證和授權

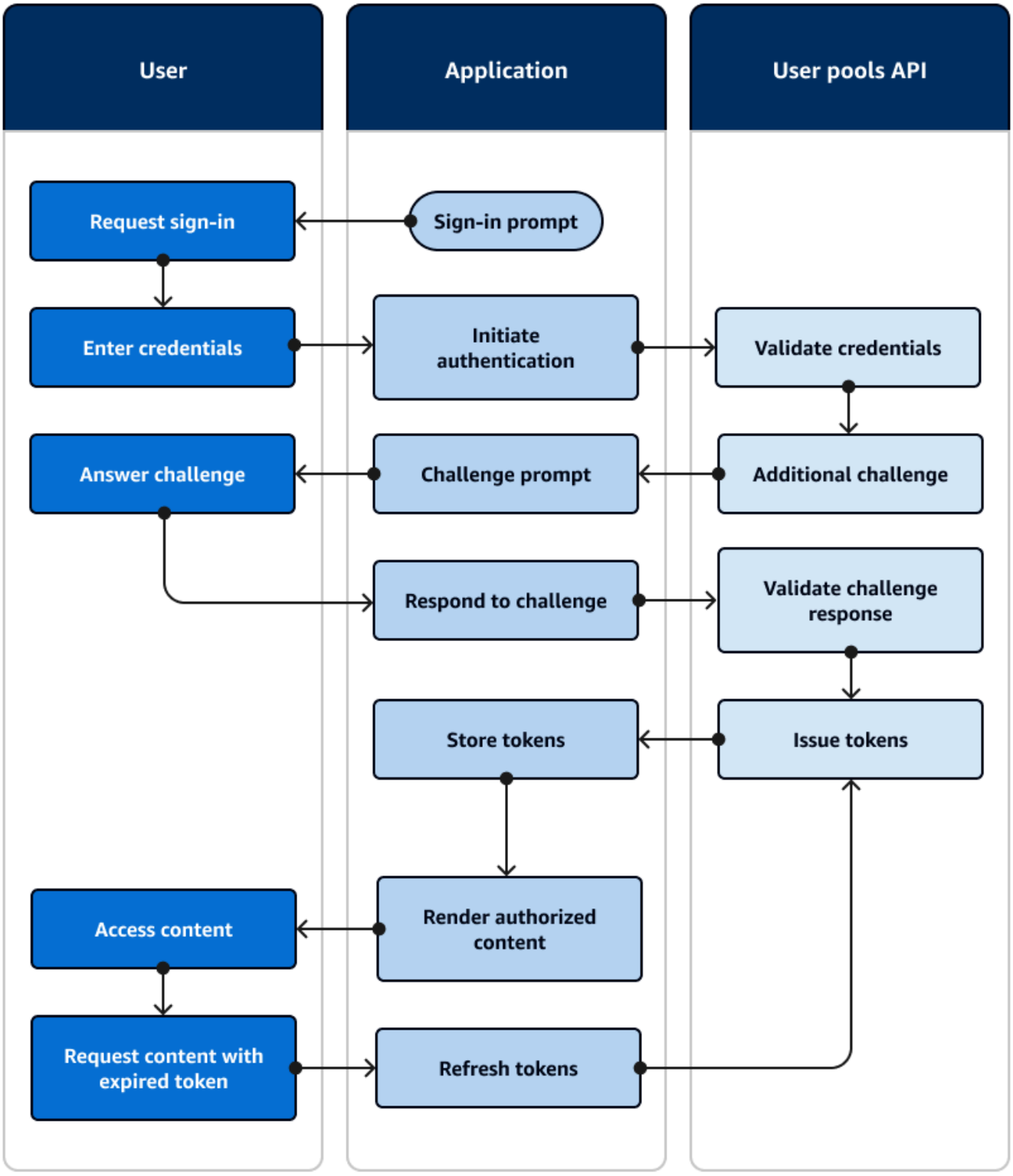
AWS 已經在各種開發人員框架中為 Amazon Cognito 使用者集區或 Amazon Cognito 身分供應商開發了元件。這些開發套件內建的方法會呼叫 Amazon Cognito 使用者集區 API。相同的用戶池 API 命名空間具有用於配置用戶池和用戶身份驗證的操作。如需更全面的概觀，請參閱使用 Amazon Cognito 使用者集區 API 和使用者集區端點。

API 身份驗證適合您的應用程序具有現有 UI 組件，並且主要依賴用戶池作為用戶目錄的模型。此設計將 Amazon Cognito 新增為較大應用程式中的元件。它需要程序化邏輯來處理複雜的挑戰和響應鏈。

此應用程序不需要實現完整的 OpenID Connect（OIDC）依賴方實現。相反，它具有解碼和使用 JWT 的能力。當您想要存取本機使用者的完整使用者集區功能時，請在開發環境中使用 Amazon Cognito SDK 建立身份驗證。

具有自定義 OAuth 範圍的 API 身份驗證不太面向外部 API 授權。要從 API 身份驗證將自定義範圍添加到訪問令牌，請在運行時使用產生權杖前 Lambda 觸發程序.

下圖說明 API 驗證的典型登入工作階段。


User
Application
User pools API
Request sign-in
Sign-in prompt
Enter credentials
Initiate authentication
Validate credentials
Answer challenge
Challenge prompt
Additional challenge
Respond to challenge
Validate challenge response
Store tokens
Issue tokens
Access content
Render authorized content
Request content with expired token
Refresh tokens

API 身份驗證流程

1. 使用者存取您的應用程式。
2. 他們選擇一個「登錄」鏈接。
3. 他們輸入他們的用戶名和密碼。
4. 應用程式會叫用發出 InitiateAuthAPI 要求的方法。要求會將使用者的認證傳送至使用者集區。
5. 使用者集區會驗證使用者的認證，並判斷使用者是否已啟動多重要素驗證 (MFA)。
6. 使用者集區會回應要求 MFA 程式碼的挑戰。
7. 應用程式會產生從使用者收集 MFA 程式碼的提示。
8. 應用程式會叫用發出 RespondToAuthChallengeAPI 要求的方法。要求會傳遞使用者的 MFA 程式碼。
9. 使用者集區會驗證使用者的 MFA 程式碼。
10. 使用者集區會回應使用者的 JWT。
11. 應用程序解碼，驗證，並存儲或緩存用戶的 JWT。
12. 應用程式會顯示要求的存取控制元件。
13. 使用者檢視其內容。
14. 稍後，用戶的訪問令牌已過期，並請求查看訪問控制的組件。
15. 應用程式會判斷使用者的工作階段應該持續存在。它使用刷新令牌再次調用該InitiateAuth方法並檢索新令牌。

變體和自訂

您可以通過其他挑戰來擴大此流程-例如，您自己的自定義身份驗證挑戰。您可以針對密碼遭到入侵的使用者自動限制存取，或其未預期登入特性可能表示惡意登入嘗試的使用者。對於註冊，更新用戶屬性和重置密碼的操作，此流程看起來大致相同。大多數這些流程都具有重複的公共（客戶端）和機密（服務器端）API 操作。

相關資源

- Amazon Cognito 用戶池 API
- 使用者集區入門
- 將 Amazon Cognito 身分驗證和授權與 Web 和行動應用程式整合
- 使用 Amazon Cognito 使用者集區 API 和使用者集區端點

使用託管 UI 進行使用者集區驗證

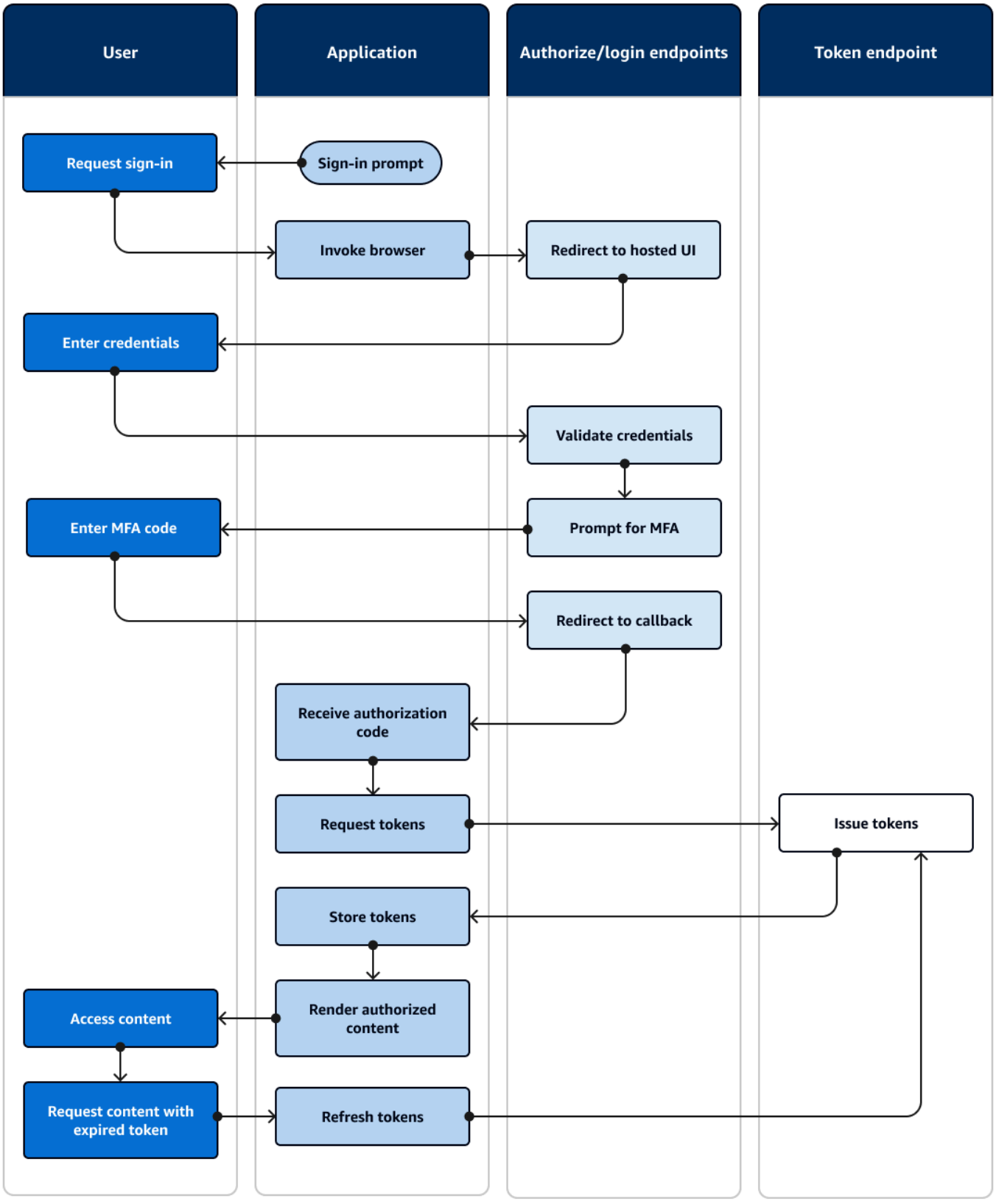
託管用戶界面是鏈接到您的用戶池和應用程序客戶端的網站。它可以為您的用戶執行登錄，註冊和密碼重置操作。具有用於驗證的託管 UI 組件的應用程序可能需要較少的開發人員的工作來實現。應用程序可以跳過 UI 組件進行身份驗證，並在用戶的瀏覽器中調用託管的 UI。

應用程序通過 Web 或應用程序重定向位置收集用戶的 JWT。實作託管 UI 的應用程式可以 Connect 到使用者集區進行驗證，就像它們是 OpenID 連線 (OIDC) IdP 一樣。

託管 UI 身份驗證適合應用程序需要授權服務器但不需要自定義身份驗證，身份池集成或用戶屬性自助服務等功能的模型。當您想要使用其中一些進階選項時，可以使用 SDK 的使用者集區元件來實作這些選項。

主要依賴 OIDC 實施的託管 UI 和第三方 IdP 身份驗證模型最適合具有 OAuth 2.0 範圍的高級授權模型。

下圖說明託管 UI 驗證的典型登入工作階段。


User
Application
Authorize/login endpoints
Token endpoint
Request sign-in
Sign-in prompt
Invoke browser
Redirect to hosted UI
Enter credentials
Validate credentials
Enter MFA code
Prompt for MFA
Redirect to callback
Receive authorization code
Request tokens
Issue tokens
Store tokens
Access content
Render authorized content
Request content with expired token
Refresh tokens

託管 UI 身份驗證流程

1. 使用者存取您的應用程式。
2. 他們選擇一個「登錄」鏈接。
3. 應用程式會將使用者導向至託管的 UI 登入提示。
4. 他們輸入他們的用戶名和密碼。
5. 使用者集區會驗證使用者的認證，並判斷使用者是否已啟動多重要素驗證 (MFA)。
6. 託管的 UI 會提示使用者輸入 MFA 代碼。
7. 使用者輸入他們的 MFA 代碼。
8. 託管 UI 將用戶重定向到應用程序。
9. 應用程式會從 URL 要求參數收集授權碼，該參數會將裝載的 UI 附加至回呼 URL。
10. 應用程序請求帶有授權代碼的令牌。
11. 令牌端點返回 JWT 到應用程序。
12. 應用程序解碼，驗證，並存儲或緩存用戶的 JWT。
13. 應用程式會顯示要求的存取控制元件。
14. 使用者檢視其內容。
15. 稍後，用戶的訪問令牌已過期，並請求查看訪問控制的組件。
16. 應用程式會判斷使用者的工作階段應該持續存在。它使用刷新令牌從令牌端點請求新令牌。

變體和自訂

您可以在任何應用程序客戶端中使用 CSS 自定義託管 UI 的外觀和感覺。您也可以使用自己的身分識別提供者、範圍、使用者屬性的存取權和進階安全性設定來設定應用程式用戶端。

相關資源

- 設定和使用 Amazon Cognito 託管 UI 和聯合端點
- 使用託管 UI 註冊和登入
- 使用資源伺服器進行範圍、M2M 和 API 授權
- 使用者集區聯合端點和託管 UI 參考

使用第三方身分識別提供者進行使用者

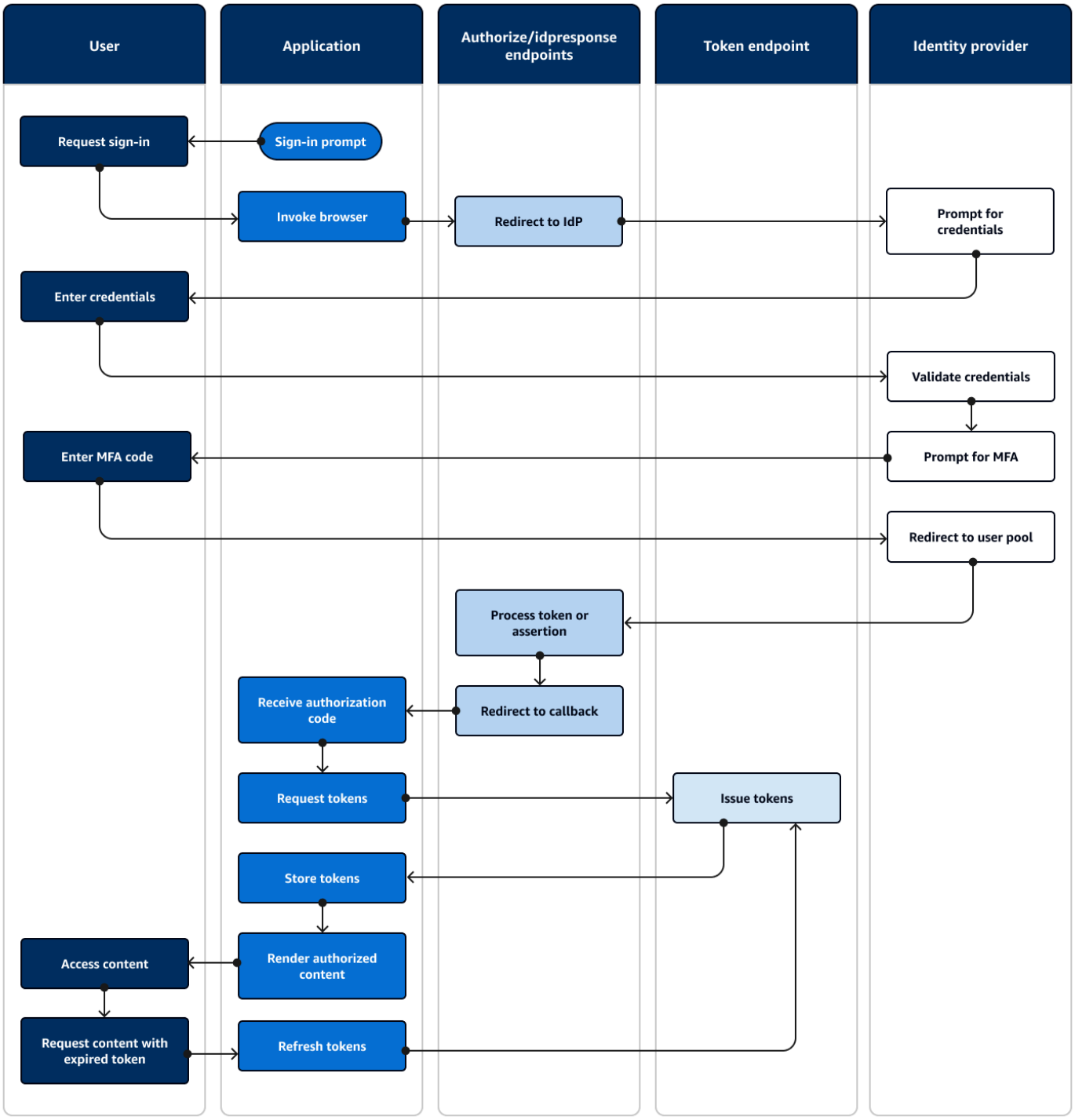
使用外部身分識別提供者 (IdP) 或聯合驗證登入與託管 UI 類似的模型。您的應用程式是使用者集區的 OIDC 信賴方，而您的使用者集區則是 IdP 的傳遞。IdP 可以是消費者使用者目錄，如臉書或谷歌，也可以是 SAML 2.0 或 OIDC 企業目錄，如 Azure。

您的應用程式會叫用使用者集區授權伺服器上的重新導向端點，而不是使用者瀏覽器中的託管 UI。從用戶的視圖中，他們選擇應用程序中的登錄按鈕。然後他們的 IdP 會提示他們登入。與託管的 UI 身份驗證一樣，應用程序會在應用程序中的重定向位置收集 JWT。

使用第三方 IdP 進行驗證適合使用者在註冊應用程式時可能不想要提供新密碼的模型。協力廠商驗證可以輕鬆新增至實作託管 UI 驗證的應用程式。實際上，託管的 UI 和協力廠商會根據您在使用者瀏覽器中呼叫的細微差異， IdPs 產生一致的驗證結果。

與託管的 UI 身份驗證一樣，聯合身份驗證最適合具有 OAuth 2.0 範圍的高級授權模型。

下圖說明聯合驗證的典型登入工作階段。



聯合身份驗證流程

1. 使用者存取您的應用程式。
2. 他們選擇一個「登錄」鏈接。

3. 應用程式會將使用者引導至具有 IdP 的登入提示。
4. 他們輸入他們的用戶名和密碼。
5. IdP 會驗證使用者的認證，並判斷使用者是否已啟動多重要素驗證 (MFA)。
6. IdP 會提示使用者輸入 MFA 代碼。
7. 使用者輸入他們的 MFA 代碼。
8. IdP 會使用 SAML 回應或授權碼將使用者重新導向至使用者集區。
9. 如果使用者傳遞授權碼，則使用者集區會以無訊息方式交換 IdP 權杖的程式碼。使用者集區驗證 IdP 權杖，並使用新的授權碼將使用者重新導向至應用程式。
10. 應用程式會從 URL 要求參數 (使用者集區附加至回呼 URL) 收集授權碼。
11. 應用程序請求帶有授權代碼的令牌。
12. 令牌端點返回 JWT 到應用程序。
13. 應用程序解碼，驗證，並存儲或緩存用戶的 JWT。
14. 應用程式會顯示要求的存取控制元件。
15. 使用者檢視其內容。
16. 稍後，用戶的訪問令牌已過期，並請求查看訪問控制的組件。
17. 應用程式會判斷使用者的工作階段應該持續存在。它使用刷新令牌從令牌端點請求新令牌。

變體和自訂

您可以在託管 UI 中啟動聯合身份驗證，用戶可以從分配給應用程序客戶端 IdPs 的列表中進行選擇。託管的 UI 也可以提示輸入電子郵件地址，並自動將使用者的要求路由到對應的 SAML IdP。使用第三方身分識別提供者進行驗證不需要使用者與託管 UI 互動。您的應用程式可以將要求參數新增至使用者的授權伺服器要求，並導致使用者以無訊息方式重新導向至其 IdP 登入頁面。

相關資源

- 透過第三方新增使用者集區登入
- 範例案例：在企業儀表板中將 Amazon Cognito 應用程式加入書籤
- 使用資源伺服器進行範圍、M2M 和 API 授權
- 使用者集區聯合端點和託管 UI 參考

身分集區驗證

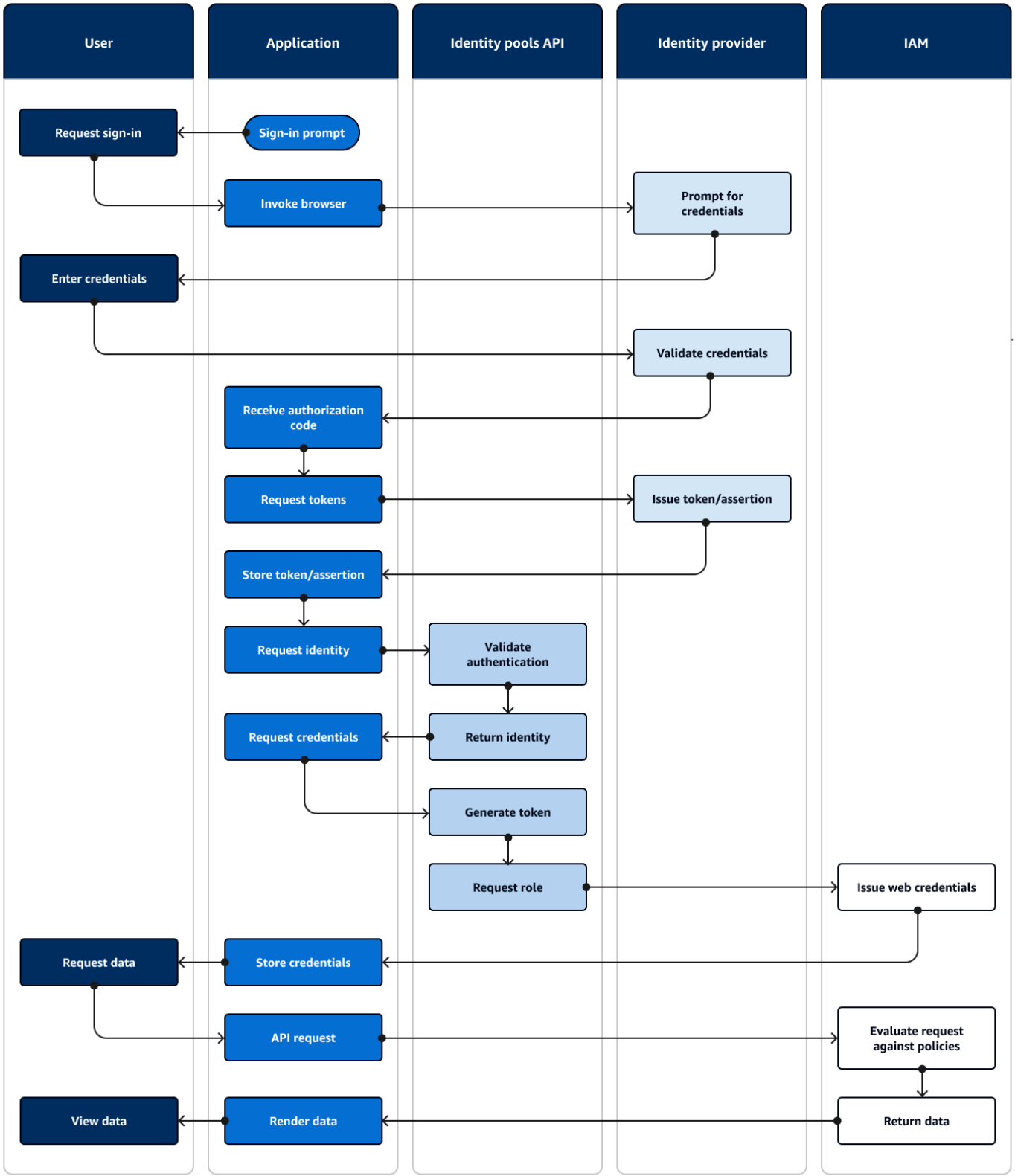
身分集區是應用程式的元件，與函數、API 命名空間和 SDK 模型中的使用者集區不同。在使用者集區提供基於令牌的身份驗證和授權的情況下，身份集區會為 AWS Identity and Access Management（IAM）提供授權。

您可以將一組指派 IdPs 給身分集區，並使用它們登入使用者。使用者集區與身分集區緊密整合， IdPs 並為身分集區提供存取控制的最多選項。同時，身分集區有多種驗證選項可供選擇。使用者集區會將 SAML、OIDC、社交、開發人員和來賓身分識別來源結合為從身分集區到臨時 AWS 登入資料的路由。

身分識別集區的驗證是外部的 — 它遵循先前說明的其中一個使用者集區流程，或是您使用另一個 IdP 獨立開發的流程。在您的應用程式執行初始驗證之後，它會將證明傳遞至身分集區，並接收暫時工作階段作為回報。

使用身分集區進行身份驗證適合您使用 IAM 授權對應用程式資產和資料強制執行存取控制 AWS 服務的模型。就像使用者集區中的 API 驗證一樣，成功的應用程式會針對您要存取的每個服務包含 AWS SDK，以供使用者受益。 AWS SDK 會將身分識別集區驗證的認證套用為 API 要求的簽章。

下圖說明使用 IdP 進行身分集區驗證的典型登入工作階段。


User
Application
Identity pools API
Identity provider
IAM
Request sign-in
Sign-in prompt
Invoke browser
Prompt for credentials
Enter credentials
Validate credentials
Receive authorization code
Request tokens
Issue token/assertion
Store token/assertion
Request identity
Validate authentication
Request credentials
Return identity
Generate token
Request role
Issue web credentials
Request data
Store credentials
API request
Evaluate request against policies
View data
Render data
Return data

聯合身份驗證流程

1. 使用者存取您的應用程式。
2. 他們選擇一個「登錄」鏈接。
3. 應用程式會將使用者引導至具有 IdP 的登入提示。
4. 他們輸入他們的用戶名和密碼。
5. IdP 會驗證使用者的認證。
6. IdP 會使用 SAML 回應或授權碼將使用者重新導向至應用程式。
7. 如果用戶傳遞了授權碼，則應用程序將代碼交換為 IdP 令牌。
8. 應用程序解碼，驗證，並存儲或緩存用戶的 JWT 或斷言。
9. 應用程式會叫用發出 GetIdAPI 要求的方法。它傳遞用戶的令牌或斷言，並請求一個身份 ID。
10. 身分集區會針對已設定的身分識別提供者驗證 Token 或宣告。
11. 身分集區會傳回身分識別碼。
12. 應用程式會叫用發出 GetCredentialsForIdentityAPI 要求的方法。它會傳遞使用者的權杖或宣告，並請求 IAM 角色。
13. 身分集區會產生新的 JWT。新的 JWT 包含請求 IAM 角色的聲明。身分識別集區會根據使用者的要求和 IdP 身分集區組態中的角色選取條件來決定角色。
14. AWS Security Token Service (AWS STS) 回應來自身分識別集區的AssumeRoleWithWebIdentity要求。回應包含具有 IAM 角色之臨時工作階段的 API 登入資料。
15. 應用程式會儲存工作階段認證。
16. 使用者在應用程式中執行需要存取保護資源的動作。 AWS
17. 該應用程序將臨時憑據作為簽名應用於所需的 API 請求 AWS 服務。
18. IAM 會評估登入資料中附加至角色的政策。它將它們與請求進行比較。
19. 會 AWS 服務 傳回要求的資料。
20. 應用程式會在使用者介面中呈現資料。
21. 使用者檢視資料。

變體和自訂

若要使用使用者集區視覺化驗證，請在 [問題權杖/宣告] 步驟之後插入先前的使用者集區概觀之一。開發人員身份驗證用開發人員憑據簽署的請求替換請求身份之前的所有步驟 訪客身份驗證還會直接跳到「請求身分」，不驗證身份驗證，並返回限制存取 IAM 角色的登入資料。

相關資源

- Amazon Cognito 身分集區
- 使用者 IAM 角色
- 身分集區概念
- 身分集區 (聯合身分) 驗證流程

Amazon Cognito 條款

Amazon Cognito 提供網頁和行動應用程式的登入資料。它從身分識別和存取管理中常見的條款汲取並建立起來。提供許多通用身分識別和存取條款的指南。部分範例如下：

- IDPro 知識體系中的術語
- AWS 身份服務
- NIST 中國證監會詞彙表

下列清單描述 Amazon Cognito 專屬的術語，或在 Amazon Cognito 中具有特定內容的術語。

主題

- 一般
- 使用者集區
- 身分集區

一般

此清單中的術語並不特定於 Amazon Cognito，而且在身分和存取管理從業人員中得到廣泛認可。以下不是詳盡的術語清單，而是本指南中特定 Amazon Cognito 上下文的指南。

应用

通常情況下，移動應用程序。在本指南中，應用程式通常是連線至 Amazon Cognito 的網路應用程式或行動應用程式的簡寫。

屬性型存取控制 (ABAC)

一種模型，其中應用程序根據用戶的屬性（例如職稱或部門）確定對資源的訪問權限。用於強制執行 ABAC 的 Amazon Cognito 工具會在使用者集區中包含 ID 權杖，以及身分集區中的主要標籤。

授權伺服器

生成 JSON 網絡令牌的基於 Web 的系統。Amazon Cognito 使用者集區聯合端點是使用者集區中兩種身份驗證和授權方法的授權伺服器元件。另一種方法是用戶池 API。

機密應用，服務器端應用

使用者透過應用程式伺服器上的程式碼和密碼存取遠端連線的應用程式。這通常是一個 Web 應用程序。

Identity provider (IdP) (身分提供者 (IdP))

儲存和驗證使用者身分的服務。Amazon Cognito 可以向外部供應商請求身份驗證，並成為應用程式的 IdP。

網絡令牌（JWT）

JSON 格式的文件，其中包含有關已驗證使用者的宣告。ID 令牌對用戶進行身份驗證，訪問令牌授權用戶並刷新令牌更新憑據。Amazon Cognito 接收來自外部供應商的權杖，並向應用程式或 AWS STS發行權杖。

多重要素驗證 (MFA)

使用者在提供使用者名稱和密碼之後提供額外驗證的要求。Amazon Cognito 使用者集區具有適用於本機使用者的 MFA 功能。

OAuth 2.0（社交）提供商

提供 JWT 存取和重新整理權杖的使用者集區或身分集區的 IdP。Amazon Cognito 使用者集區會在使用者驗證後自動與社交供應商的互動。

OpenID Connect（OIDC）提供商

IdP 至使用者集區或身分集區，可擴充 OAuth 規格以提供 ID 權杖。Amazon Cognito 使用者集區會在使用者驗證後自動化與 OIDC 提供者的互動。

公共應用

設備上獨立的應用程序，其代碼存儲在本地並且無法訪問密碼。這通常是行動應用程式。

資源伺服器

具有存取控制的 API。Amazon Cognito 使用者集區也會使用資源伺服器來描述定義與 API 互動之組態的元件。

角色型存取控制 (RBAC)

根據使用者的功能指定來授與存取權的模型。Amazon Cognito 身分集區會實作 RBAC，並區分 IAM 角色。

服務提供者 (SP)、信賴方 (RP)

依賴 IdP 來判斷使用者值得信賴的應用程式。Amazon Cognito 充當外部的 SP IdPs，並作為基於應用程序的 SP 的 IdP。

SAML 提供者

使用者集區或身分集區的 IdP，可產生數位簽署的宣告文件，讓您的使用者傳遞至 Amazon Cognito。

通用唯一識別碼 (UUID)

套用至物件的 128 位元標籤。Amazon Cognito UUID 是每個使用者集區或身分集區唯一的。

使用者目錄

將該資訊提供給其他系統的使用者及其屬性的集合。Amazon Cognito 使用者集區是使用者目錄，也是整合外部使用者目錄中使用者的工具。

使用者集區

當您在本指南中看到以下清單中的術語時，這些詞彙是指使用者集區的特定功能或設定。

Amazon Cognito 用戶池 API

一組身份驗證和授權 API 操作，您可以使用 AWS SDK 將其添加到應用程序中。該 API 可以登錄本地用戶和鏈接的用戶。

自適應身分驗證

進階安全性功能，可偵測潛在的惡意活動，並將額外的安全性套用至使用者設定檔。

進階安全功能

新增使用者安全性工具的選用元件。

應用客戶端

將使用者集區設定定義為一個應用程式的 IdP 的元件。

回調網址，重定向 URI

應用程式用戶端中的設定，以及對使用者集區聯合端點的要求中的參數。回呼 URL 是應用程式中已驗證使用者的初始目的地。

憑證洩漏

進階安全性功能，可偵測攻擊者可能知道的使用者密碼，並將額外的安全性套用至使用者設定檔。

確認

決定已符合先決條件以允許新使用者登入的程序。確認通常通過電子郵件地址或電話號碼驗證完成。

自訂身分驗證

Lambda 觸發程序的驗證程序延伸，可定義其他使用者挑戰和回應。

裝置驗證

使用受信任裝置 ID 的登入取代 MFA 的驗證程序。

外部供應商，第三方供應

與使用者集區具有信任關係的 IdP。

聯合身分使用者

已由外部提供者驗證的使用者集區中的使用者。

同盟端點

您的使用者集區網域上的一組網頁，用於託管與 IdPs 和應用程式互動的服務。

託管 UI

您的使用者集區網域上的一組互動式網頁，主控服務以供使用者驗證。

Lambda 觸發程序

使用者集區 AWS Lambda 可在使用者驗證程序中的關鍵點自動叫用的函數。您可以使用 Lambda 觸發程序來自訂驗證結果。

本機使用者

使用者集區使用者目錄中的使用者設定檔，並非透過外部提供者驗證所建立的使用者設定檔。

連結使用者

來自外部提供者的使用者，其身分已與本機使用者合併。

令牌定制

預先產生權杖 Lambda 觸發程序的結果，該觸發程序會在執行階段修改使用者的 ID 或存取權杖。

使用者集區、亞馬遜認可身分供應商**cognito-idp**、Amazon Cognito 使用者集區

具有驗證和授權服務的 AWS 資源，適用於與 OIDC IdPs 一起使用的應用程式。

使用者集區網域

您新增至使用者集區的網站名稱。網域是託管 UI 和同盟端點的基礎 URL。

驗證

確認使用者擁有電子郵件地址或電話號碼的程序。使用者集區會傳送代碼給已輸入新電子郵件地址或電話號碼的使用者。當他們將代碼提交給 Amazon Cognito 時，他們會驗證其對訊息目標的所有權，並可從使用者集區接收其他訊息。另外，請參閱確認。

用戶配置文件，用戶帳戶

使用者目錄中使用者的項目。所有用戶在其用戶池中都有一個配置文件。

身分集區

當您在本指南中看到下列清單中的術語時，這些詞彙是指識別集區的特定功能或設定。

存取控制的屬性

在識別集區中實作以屬性為基礎的存取控制。識別集區會將使用者屬性做為標記套用至使用者認證

基本 (傳統) 驗證

一種驗證程序，您可以在其中自訂使用者認證的要求。

開發人員驗證的身分

使用開發人員認證授權身分集區使用者憑證的驗證程序。

開發者憑證

身分集區管理員的 IAM API 金鑰。

增強型驗證

一種身份驗證流程，可根據您在身分集區中定義的邏輯選擇 IAM 角色並套用主要標籤。

Identity

一種 UUID，可將應用程式使用者及其使用者憑證連結至其個人資料的外部使用者目錄中與身分識別集區具有信任關係。

身份池，Amazon Cognito 聯合身份，Amazon Cognito 身份，**cognito-identity**

具有驗證和授權服務的 AWS 資源，適用於使用臨時 AWS 認證的應用程式。

未驗證的 身分

未使用識別集區 IdP 登入的使用者。您可以允許使用者在驗證之前為單一 IAM 角色產生有限的使用者登入資料。

使用者認證

使用者在身分集區驗證後接收的暫時 AWS API 金鑰。

搭配 AWS SDK 使用此服務

AWS 軟件開發套件（SDK）可用於許多流行的編程語言。每個 SDK 都提供 API、程式碼範例和說明文件，讓開發人員能夠更輕鬆地以偏好的語言建置應用程式。

SDK 文件	代碼範例
AWS SDK for C++	AWS SDK for C++ 程式碼範例
AWS CLI	AWS CLI 程式碼範例
AWS SDK for Go	AWS SDK for Go 程式碼範例
AWS SDK for Java	AWS SDK for Java 程式碼範例
AWS SDK for JavaScript	AWS SDK for JavaScript 程式碼範例
適用於 Kotlin 的 AWS SDK	適用於 Kotlin 的 AWS SDK 程式碼範例
AWS SDK for .NET	AWS SDK for .NET 程式碼範例
AWS SDK for PHP	AWS SDK for PHP 程式碼範例
AWS Tools for PowerShell	PowerShell 程式碼範例工具

SDK 文件	代碼範例
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

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 AWS

在您開始使用 Amazon Cognito 之前，請先為自己設定一些必要的 AWS 資源。

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 AWS 帳戶 AWS 帳戶根使用者、啟用和建立系統管理使用者 AWS IAM Identity Center，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，請參閱使用指南中的登入 AWS 存取入口網站。AWS 登入

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」。

使用者集區入門

您可以使用本節中的指南來建立初始使用者集區資源。如需 step-by-step 逐步解說，請從 React 開 JavaScript 發人員環境中的基本 Web 應用程式開始。從那裡，您可以繼續添加託管用戶界面（託管 UI）等功能，以及與外部社交或 SAML 2.0 身份提供者的聯合登錄（IdPs）。

當您努力擴展功能集並整合 Amazon Cognito 的更多元件時，請閱讀 Amazon Cognito 使用者集區章節，瞭解您可以使用使用者集區執行的所有操作的完整說明。

本節中的使用者集區和應用程式範例示範應用程式資源與 Amazon Cognito 使用者集區的基本整合。稍後，您可以調整使用者集區，以使用更多可用的選項。然後，您可以更新應用程式以採用新的 API，並與託管的 UI 和 IdPs.

本節中的教學課程會建立具有自訂 UI 和以 SDK 為基礎的 API 驗證的應用程式 AWS 。您以這種方式建置的應用程式非常適合驗證本機使用者。若要從具有預先建置的 UI、自動處理某些使用者集區功能以及聯合使用者驗證的應用程式開始，請跳至。使用託管 UI 添加應用程序客戶端

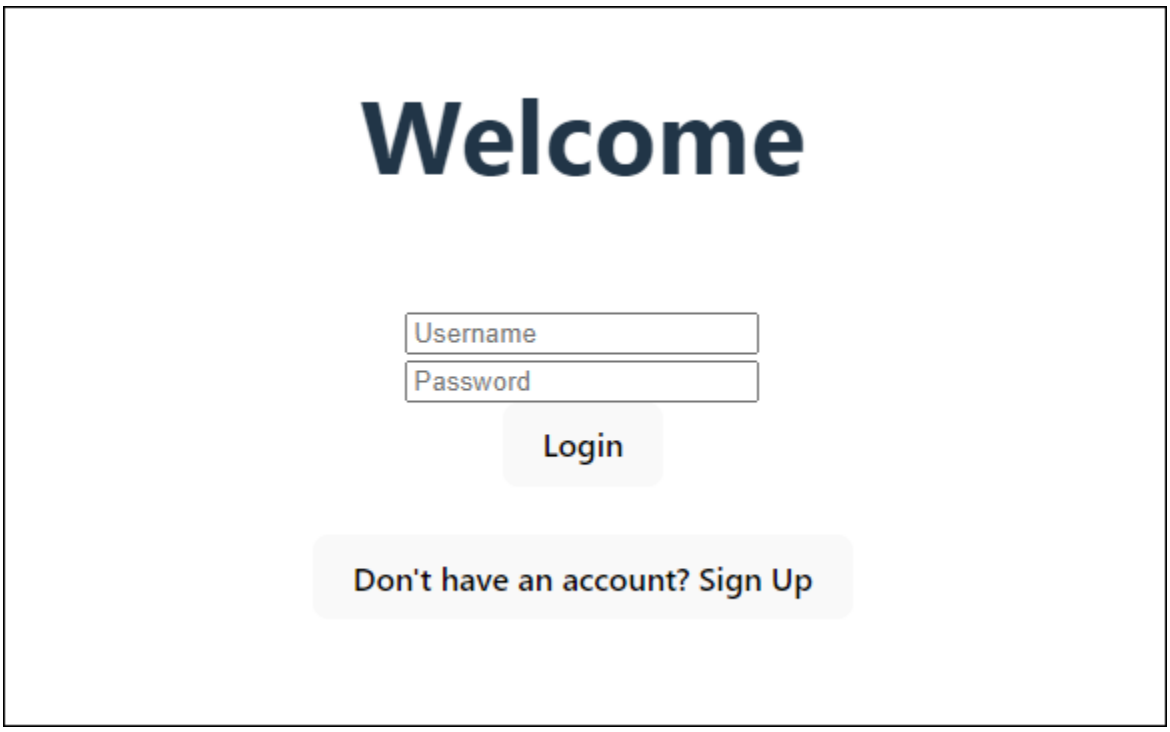
主題

- 設置一個示例 React 單頁應用程序
- 設置一個示例 Android 應用程序與撲
- 後續步驟

設置一個示例 React 單頁應用程序

在本教學中，您將建立 React 單頁應用程式，以測試使用者註冊、確認和登入。React 是一個 JavaScript基於 Web 和移動應用程序的庫，重點放在用戶界面（UI）。此範例應用程式示範 Amazon Cognito 使用者集區的一些基本功能。如果您已經在使用 React 進行 Web 應用程式開發方面有經驗，請從下載範例應用程式 GitHub。

下列螢幕擷取畫面是您要建立的應用程式中的初始驗證頁面。



[建立使用者集區] 程序會為您設定可搭配範例應用程式使用的使用者集區。如果您的使用者集區符合下列需求，則可略過此步驟：

- 使用者可以使用其電子郵件地址登入。Cognito 使用者集區登入選項：電子郵件。
- 使用者名稱不區分大小寫。使用者名稱需求：未選取使用者名稱區分大小寫。
- 不需要多因素身份驗證（MFA）。MFA 執行：可選 M FA。
- 您的使用者集區會透過電子郵件訊息驗證使用者設定檔確認的屬性。要驗證的屬性：傳送電子郵件訊息、驗證電子郵件地址。
- 電子郵件是唯一必需的屬性。必要屬性：電子郵件。
- 使用者可以在您的使用者集區中自行註冊。自助註冊：已選取「啟用自助註冊」。
- 您的初始應用程序客戶端是一個公共客戶端，允許使用用戶名和密碼登錄。應用程序類型：公共客戶端，身份驗證流程：ALLOW_USER_PASSWORD_AUTH。

建立使用者集區

建立新的使用者集區

1. 前往 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 選擇 [建立使用者集區] 按鈕。您可能需要從左側導覽窗格中選取 [使用者集區]，以顯示此選項。

3. 在頁面右上角，選擇 Create a user pool (建立使用者集區)，以開始建立使用者集區精靈。
4. 在 [設定登入體驗] 中，您可以選擇要用於此使用者集區的身分識別提供者 (IdPs)。如需詳細資訊，請參閱 透過第三方新增使用者集區登入。

   a. 在驗證提供者下，對於提供者類型，請確保僅選取 Cognito 使用者集區。

   b. 在 Cognito 使用者集區登入選項中，選擇 [使用者名稱]。請勿選取任何其他使用者名稱要求。

   c. 保留所有其他選項為預設值，然後選擇「下一步」。
5. 在 [設定安全性需求] 中，您可以選擇密碼原則、多重要素驗證 (MFA) 需求，以及使用者帳戶復原選項。如需詳細資訊，請參閱 使用 Amazon Cognito 使用者集區安全性功能。

   a. 對於密碼原則，請確認密碼原則模式設定為 Cognito 預設值。

   b. 在多因素驗證下，對於 MFA 強制，請選擇選擇性 MFA。

   c. 對於 MFA 方法，請選擇驗證器應用程式和 SMS 訊息。

   d. 對於使用者帳戶復原，請確認已選取 [啟用自助帳戶復原]，且使用者帳戶復原郵件傳遞方法設定為 [僅限電子郵件]。

   e. 保留所有其他選項為預設值，然後選擇「下一步」。
6. 在 [設定註冊體驗] 中，您可以決定新使用者在以新使用者身分註冊時如何驗證其身分，以及在使用者註冊流程期間，哪些屬性應為必要或選用屬性。如需詳細資訊，請參閱 管理使用者集區中的使用者。

   a. 確認已選取 「啟用自助註冊」。此設定會開啟您的使用者集區，以便從網際網路上的任何人進行註冊。這是用於範例應用程式的目的，但在生產環境中請謹慎套用此設定。

   b. 在「Cognito 協助驗證與確認」下，確認已選取「允許 Cognito 自動傳送訊息以進行驗證並確認」核取方塊。

   c. 確認要驗證的屬性設定為傳送電子郵件訊息，驗證電子郵件地址。

   d. 在 [驗證屬性變更] 下，確認已選取預設選項：選取更新處於擱置狀態時保留原始屬性值，以及將更新擱置時的 Active 屬性值設定為 [電子郵件地址]。

   e. 在必要屬性下，確認根據先前選取的必要屬性顯示電子郵件。

   > **Important**
   >
   > 對於此示例應用程序，您的用戶池不得將 phone_number 設置為必需屬性。如果 phone_number 顯示為必要屬性，請檢閱並更新您先前的選擇：
   >
   > - 可選 MFA，僅用於用戶帳戶恢復消息的傳遞方法的電子郵件

- 發送電子郵件，驗證要驗證屬性的電子郵件地址

f. 保留所有其他選項為預設值，然後選擇「下一步」。

7. 在設定訊息傳遞中，您可以設定與 Amazon 簡單電子郵件服務和 Amazon 簡單通知服務的整合，以傳送電子郵件和簡訊給使用者以進行註冊、帳戶確認、MFA 和帳戶復原。如需詳細資訊，請參閱 Amazon Cognito 使用者集區的電子郵件設定 及 Amazon Cognito 使用者集區的簡訊設定。

   a. 對於電子郵件提供者，請選擇「使用 Cognito 傳送電子郵件」，並使用 Amazon Cognito 提供的預設電子郵件寄件者。這個低電子郵件量的設定已足以進行應用程式測試。您可以在使用 Amazon 簡易電子郵件服務 (Amazon SES) 驗證電子郵件地址並選擇使用 Amazon SES 傳送電子郵件後退貨。

   b. 對於 SMS，請選取建立新的 IAM 角色，然後輸入 IAM 角色名稱。這會建立一個角色，授予 Amazon Cognito 許可以傳送簡訊的權限。

   c. 保留所有其他選項為預設值，然後選擇「下一步」。

8. 在「整合您的應用程式」中，您可以為使用者集區命名、設定託管 UI，以及建立應用程式用戶端。如需詳細資訊，請參閱 使用託管 UI 添加應用程序客戶端。範例應用程式不使用託管 UI。

   a. 在 [使用者集區名稱] 下，輸入使用者集區名稱。

   b. 請勿選取 [使用 Cognito 託管的使用者介面]。

   c. 在初始應用程序客戶端下，確認應用程序類型設置為公共客戶端。

   d. 在 [用戶端密碼] 下，確認已選取 [不產生用戶端密碼]。

   e. 輸入 App client name (應用程式用戶端名稱)。

   f. 展開進階應用程式用戶端設定 新增`ALLOW_USER_PASSWORD_AUTH`至驗證流程清單。

   g. 保留所有其他選項為預設值，然後選擇「下一步」。

9. 在「檢閱並建立」畫面中檢閱您的選擇，並視需要修改任何選取項目。如果您對使用者集區設定感到滿意，請選擇 [建立使用者集區] 以繼續。

10. 在 [使用者集區] 頁面中，選擇新的使用者集區。

11. 在使用者集區概觀下，記下您的使用者集區 ID。您將在建立範例應用程式時提供此字串。

12. 選擇「應用程式整合」標籤，然後找到「應用程式用戶端和分析」 選取新的應用程式用戶端。記下您的客戶 ID。

相關資源

- Amazon Cognito 使用者集區

- 使用者集區身分驗證流程
- 將權杖用於使用者集區

建立應用程式

若要建置此應用程式，您必須設定開發人員環境。開發人員環境要求為：

1. Node.js 已安裝並更新。
2. 節點包管理器（npm）已安裝並更新至至少版本 10.2.3。
3. 您可以在網頁瀏覽器中的 TCP 連接埠 5173 存取該環境。

要創建一個示例反應 Web 應用程序

1. 登入您的開發人員環境，並導覽至應用程式的父目錄。

```
cd ~/path/to/project/folder/
```

2. 創建一個新的反應服務。

```
npm create vite@latest frontend-client -- --template react-ts
```

3. 從上的cognito-developer-guide-react-example程 AWS 式碼範例儲存庫複製專案資料夾 GitHub。

```
cd ~/some/other/path
```

```
git clone https://github.com/awsdocs/aws-doc-sdk-examples.git
```

```
cp -r ./aws-doc-sdk-examples/javascriptv3/example_code/cognito-identity-provider/
scenarios/cognito-developer-guide-react-example/frontend-client ~/path/to/project/
folder/frontend-client
```

4. 導航到項src目中的目錄。

```
cd ~/path/to/project/folder/frontend-client/src
```

5. 編輯config.ts並取代下列值：

a. YOUR_AWS_REGION以 AWS 區域 程式碼取代。例如：us-east-1。

b. 取代YOUR_COGNITO_USER_POOL_ID為您指定用於測試的使用者集區 ID。例如：us-east-1_EXAMPLE。使用者集區必須 AWS 區域 是您在上一個步驟中輸入的。

c. YOUR_COGNITO_APP_CLIENT_ID替換為您指定要測試的應用程序客戶端的 ID。例如：1example23456789。應用程式用戶端必須位於上一個步驟的使用者集區中。

6. 如果您要從 IP 以外的 IP 存取範例應用程式localhost，請將該行編輯package.json並變更"dev": "vite",為"dev": "vite --host 0.0.0.0",。

7. 安裝您的應用程序。

```
npm install
```

8. 啟動應用程序。

```
npm run dev
```

9. 在網頁瀏覽器中存取應用程式，位於http://localhost:5173或http://[IP address]:5173。

10. 使用有效的電子郵件地址註冊新用戶。

11. 從您的電子郵件中檢索確認代碼。在應用程序中輸入確認代碼。

12. 使用您的用戶名和密碼登錄。

使用 Amazon Lightsail 創建反應開發人員環境

開始使用此應用程式的快速方法是使用 Amazon Lightsail 建立虛擬雲端伺服器。

使用 Lightsail，您可以快速建立小型伺服器執行個體，並預先設定了此範例應用程式的先決條件。您可以透過以瀏覽器為基礎的用戶端透過 SSH 連線至執行個體，並透過公用或私有 IP 位址連線至 Web 伺服器。

若要為此範例應用程式建立 Lightsail 實體

1. 前往 Li ghtsail 台。如果出現提示，請輸入您的 AWS 認證。

2. 選擇 建立執行個體。

3. 針對 [選取平台]，選擇 [Linu x/Unix]。

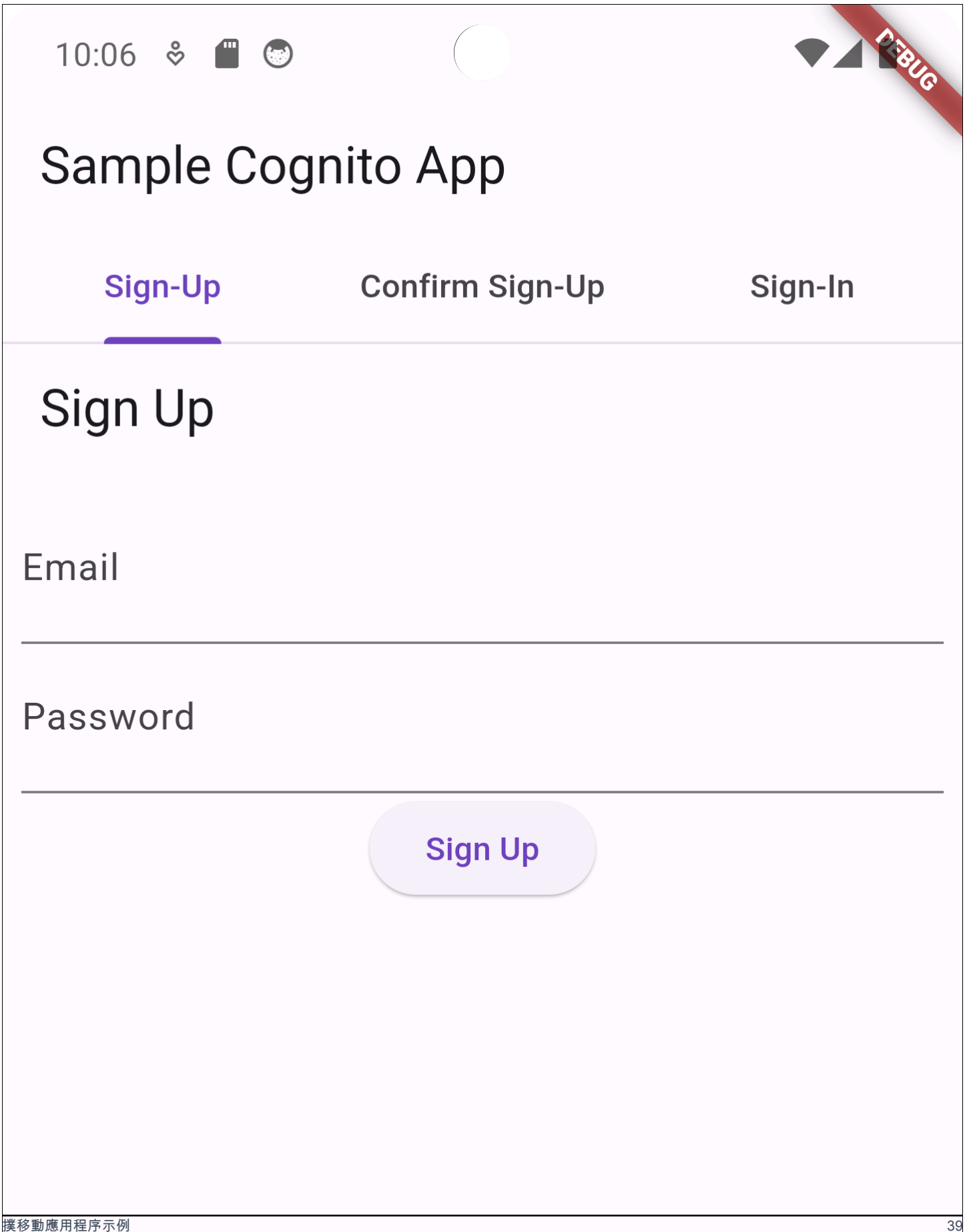
4. 針對 [選取藍圖]，選擇 Node.js。

5. 在 [識別您的執行個體] 底下，為您的開發環境提供易記的名
6. 選擇 建立執行個體。
7. Lightsail 建立您的執行個體之後，選取它，然後從 [Connect] 索引標籤中選擇 [使用 SSH Connect]。
8. SSH 工作階段會在瀏覽器視窗中開啟。執行`node -v`並`npm -v`確認您的執行個體是使用 Node.js 佈建的，以及 10.2.3 的最低 npm 版本。
9. 繼續配置您的 React 應用程序。

設置一個示例 Android 應用程序與撲

在本教程中，您將在 Android Studio 中創建一個移動應用程序，您可以在其中模擬設備並測試用戶註冊，確認和登錄。這個示例應用程序創建一個基本的 Amazon Cognito 用戶池移動客戶端為 Android 在 Flutter。如果您已經在使用 Flutter 進行移動應用程序開發方面有經驗，請從 GitHub下載示例應用程序。

下列螢幕擷取畫面顯示在虛擬 Android 裝置上執行的應用程式。


10:06
DEBUG
Sample Cognito App
Sign-Up
Confirm Sign-Up
Sign-In
Sign Up
Email
Password
Sign Up

[建立使用者集區] 程序會為您設定可搭配範例應用程式使用的使用者集區。如果您的使用者集區符合下列需求，則可略過此步驟：

- 使用者可以使用其電子郵件地址登入。Cognito 使用者集區登入選項：電子郵件。
- 使用者名稱不區分大小寫。使用者名稱需求：未選取使用者名稱區分大小寫。
- 不需要多因素身份驗證（MFA）。MFA 執行：可選 M FA。
- 您的使用者集區會透過電子郵件訊息驗證使用者設定檔確認的屬性。要驗證的屬性：傳送電子郵件訊息、驗證電子郵件地址。
- 電子郵件是唯一必需的屬性。必要屬性：電子郵件。
- 使用者可以在您的使用者集區中自行註冊。自助註冊：已選取「啟用自助註冊」。
- 您的初始應用程序客戶端是一個公共客戶端，允許使用用戶名和密碼登錄。應用程序類型：公共客戶端，身份驗證流程：ALLOW_USER_PASSWORD_AUTH。

建立使用者集區

建立新的使用者集區

1. 前往 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 選擇 [建立使用者集區] 按鈕。您可能需要從左側導覽窗格中選取 [使用者集區]，以顯示此選項。
3. 在頁面右上角，選擇 Create a user pool (建立使用者集區)，以開始建立使用者集區精靈。
4. 在 [設定登入體驗] 中，您可以選擇要用於此使用者集區的身分識別提供者 (IdPs)。如需詳細資訊，請參閱 透過第三方新增使用者集區登入。
   a. 在驗證提供者下，對於提供者類型，請確保僅選取 Cognito 使用者集區。
   b. 在 Cognito 使用者集區登入選項中，選擇 [使用者名稱]。請勿選取任何其他使用者名稱要求。
   c. 保留所有其他選項為預設值，然後選擇「下一步」。
5. 在 [設定安全性需求] 中，您可以選擇密碼原則、多重要素驗證 (MFA) 需求，以及使用者帳戶復原選項。如需詳細資訊，請參閱 使用 Amazon Cognito 使用者集區安全性功能。
   a. 對於密碼原則，請確認密碼原則模式設定為 Cognito 預設值。
   b. 在多因素驗證下，對於 MFA 強制，請選擇選擇性 MFA。
   c. 對於 MFA 方法，請選擇驗證器應用程式和 SMS 訊息。
   d. 對於使用者帳戶復原，請確認已選取 [啟用自助帳戶復原]，且使用者帳戶復原郵件傳遞方法設定為 [僅限電子郵件]。

e. 保留所有其他選項為預設值，然後選擇「下一步」。

6. 在 [設定註冊體驗] 中，您可以決定新使用者在以新使用者身分註冊時如何驗證其身分，以及在使用者註冊流程期間，哪些屬性應為必要或選用屬性。如需詳細資訊，請參閱 管理使用者集區中的使用者。

   a. 確認已選取 「啟用自助註冊」。此設定會開啟您的使用者集區，以便從網際網路上的任何人進行註冊。這是用於範例應用程式的目的，但在生產環境中請謹慎套用此設定。

   b. 在「Cognito 協助驗證與確認」下，確認已選取「允許 Cognito 自動傳送訊息以進行驗證並確認」核取方塊。

   c. 確認要驗證的屬性設定為傳送電子郵件訊息，驗證電子郵件地址。

   d. 在 [驗證屬性變更] 下，確認已選取預設選項：選取更新處於擱置狀態時保留原始屬性值，以及將更新擱置時的 Active 屬性值設定為 [電子郵件地址]。

   e. 在必要屬性下，確認根據先前選取的必要屬性顯示電子郵件。

   > **Important**
   >
   > 對於此示例應用程序，您的用戶池不得將 phone_number 設置為必需屬性。如果 phone_number 顯示為必要屬性，請檢閱並更新您先前的選擇：
   >
   > - 可選 MFA，僅用於用戶帳戶恢復消息的傳遞方法的電子郵件
   > - 發送電子郵件，驗證要驗證屬性的電子郵件地址

   f. 保留所有其他選項為預設值，然後選擇「下一步」。

7. 在設定訊息傳遞中，您可以設定與 Amazon 簡單電子郵件服務和 Amazon 簡單通知服務的整合，以傳送電子郵件和簡訊給使用者以進行註冊、帳戶確認、MFA 和帳戶復原。如需詳細資訊，請參閱 Amazon Cognito 使用者集區的電子郵件設定 及 Amazon Cognito 使用者集區的簡訊設定。

   a. 對於電子郵件提供者，請選擇「使用 Cognito 傳送電子郵件」，並使用 Amazon Cognito 提供的預設電子郵件寄件者。這個低電子郵件量的設定已足以進行應用程式測試。您可以在使用 Amazon 簡易電子郵件服務 (Amazon SES) 驗證電子郵件地址並選擇使用 Amazon SES 傳送電子郵件後退貨。

   b. 對於 SMS，請選取建立新的 IAM 角色，然後輸入 IAM 角色名稱。這會建立一個角色，授予 Amazon Cognito 許可以傳送簡訊的權限。

   c. 保留所有其他選項為預設值，然後選擇「下一步」。

8. 在「整合您的應用程式」中，您可以為使用者集區命名、設定託管 UI，以及建立應用程式用戶端。如需詳細資訊，請參閱 使用託管 UI 添加應用程序客戶端。範例應用程式不使用託管 UI。

a. 在 [使用者集區名稱] 下，輸入使用者集區名稱。

b. 請勿選取 [使用 Cognito 託管的使用者介面]。

c. 在初始應用程序客戶端下，確認應用程序類型設置為公共客戶端。

d. 在 [用戶端密碼] 下，確認已選取 [不產生用戶端密碼]。

e. 輸入 App client name (應用程式用戶端名稱)。

f. 展開進階應用程式用戶端設定 新增`ALLOW_USER_PASSWORD_AUTH`至驗證流程清單。

g. 保留所有其他選項為預設值，然後選擇「下一步」。

9. 在「檢閱並建立」畫面中檢閱您的選擇，並視需要修改任何選取項目。如果您對使用者集區設定感到滿意，請選擇 [建立使用者集區] 以繼續。

10. 在 [使用者集區] 頁面中，選擇新的使用者集區。

11. 在使用者集區概觀下，記下您的使用者集區 ID。您將在建立範例應用程式時提供此字串。

12. 選擇「應用程式整合」標籤，然後找到「應用程式用戶端和分析」 選取新的應用程式用戶端。記下您的客戶 ID。

相關資源

- Amazon Cognito 使用者集區
- 使用者集區身分驗證流程
- 將權杖用於使用者集區

建立應用程式

若要建立範例安卓應用程式

1. 安裝 Android 工作室和命令列工具。

2. 在安卓工作室中，安裝撲插件。

3. 從此示例應用程序中的`cognito_flutter_mobile_app`目錄內容創建一個新的 Android 工作室項目。

   - 編輯`assets/config.json`<<YOUR USER POOL ID>>和<< YOUR CLIENT ID>>取代您先前建立的使用者集區和應用程式用戶端的 ID。

4. 安裝撲。

a. 添加撲到您的 PATH 變量。

b. 使用下列命令接受授權。

```
flutter doctor --android-licenses
```

c. 驗證您的 Flutter 環境並安裝任何缺少的組件。

```
flutter doctor
```

- 如果缺少任何組件，請運行`flutter doctor -v`以了解如何解決此問題。

d. 切換到新的 Flutter 項目的目錄並安裝依賴項。

- 執行 `flutter pub add amazon_cognito_identity_dart_2`。

e. 執行 `flutter pub add flutter_secure_storage`。

5. 創建一個虛擬的安卓設備。

1. 在 Android 工作室圖形用戶界面中，使用設備管理器創建一個新設備。
2. 在 CLI 中，執行`flutter emulators --create --name android-device`。

6. 啟動您的虛擬安卓設備。

1. 在 Android 工作室圖形用戶界面中，選擇虛擬設備 旁邊的開始圖標。
2. 在 CLI 中，執行`flutter emulators --launch android-device`。

7. 在虛擬設備上啟動您的應用程序。

1. 在安卓工作室圖形用戶界面中，選擇部署 圖標。
2. 在 CLI 中，執行`flutter run`。

8. 在 Android 工作室中導航到正在運行的虛擬設備。
9. 使用有效的電子郵件地址註冊新用戶。
10. 從您的電子郵件中檢索確認代碼。在應用程序中輸入確認代碼。
11. 使用您的用戶名和密碼登錄。

後續步驟

依照教學課程完成範例應用程式之後，您可以擴大使用者集區實作的範圍。您可以建立其他使用者集區、為其他應用程式自訂使用者集區功能，或新增外部身分識別提供者。當您計劃在生產應用程式中放置 Amazon Cognito 使用者集區時，您可以評估其他範例和教學課程。

以下是一些額外的 Amazon Cognito 使用者集區功能：

- 自訂內建的註冊與登入網頁
- 將 MFA 新增到使用者集區
- 將進階安全性新增到使用者集區
- 使用 Lambda 觸發程序來自訂使用者集區工作流程
- 使用 Amazon Pinpoint 分析搭配 Amazon Cognito 使用者集區

如需 Amazon Cognito 身份驗證和授權模式的概觀，請參閱身份驗證如何與 Amazon Cognito 使用者集區和身分集區搭配使用。

若要在成功驗證使用者集區 AWS 服務 後存取其他，請參閱登入後 AWS 服務 使用身分集區存取。

除了使用 AWS Management Console 和使用者集區 SDK 之外，您還可以使用 AWS Command Line Interface.

主題

- 建立新的使用者集區
- 使用託管 UI 添加應用程序客戶端
- 新增社交登入至使用者集區 (選用)
- 以 SAML 身分提供者新增登入至使用者集區 (選用)

建立新的使用者集區

利用使用者集區，您的使用者可以透過 Amazon Cognito 登入您的 Web 或行動應用程式。

建立新的使用者集區

1. 前往 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 選擇 [建立使用者集區] 按鈕。您可能需要從左側導覽窗格中選取 [使用者集區]，以顯示此選項。
3. 在頁面右上角，選擇 Create a user pool (建立使用者集區)，以開始建立使用者集區精靈。

4. 在 [設定登入體驗] 中，您可以選擇要用於此使用者集區的身分識別提供者 (IdPs)。如需詳細資訊，請參閱 透過第三方新增使用者集區登入。

   a. 在驗證提供者下，對於提供者類型，請確保僅選取 Cognito 使用者集區。

   b. 在 Cognito 使用者集區登入選項中，選擇 [使用者名稱]。請勿選取任何其他使用者名稱要求。

   c. 保留所有其他選項為預設值，然後選擇「下一步」。

5. 在 [設定安全性需求] 中，您可以選擇密碼原則、多重要素驗證 (MFA) 需求，以及使用者帳戶復原選項。如需詳細資訊，請參閱 使用 Amazon Cognito 使用者集區安全性功能。

   a. 對於密碼原則，請確認密碼原則模式設定為 Cognito 預設值。

   b. 在多因素驗證下，對於 MFA 強制，請選擇選擇性 MFA。

   c. 對於 MFA 方法，請選擇驗證器應用程式和 SMS 訊息。

   d. 對於使用者帳戶復原，請確認已選取 [啟用自助帳戶復原]，且使用者帳戶復原郵件傳遞方法設定為 [僅限電子郵件]。

   e. 保留所有其他選項為預設值，然後選擇「下一步」。

6. 在 [設定註冊體驗] 中，您可以決定新使用者在以新使用者身分註冊時如何驗證其身分，以及在使用者註冊流程期間，哪些屬性應為必要或選用屬性。如需詳細資訊，請參閱 管理使用者集區中的使用者。

   a. 確認已選取 「啟用自助註冊」。此設定會開啟您的使用者集區，以便從網際網路上的任何人進行註冊。這是用於範例應用程式的目的，但在生產環境中請謹慎套用此設定。

   b. 在「Cognito 協助驗證與確認」下，確認已選取「允許 Cognito 自動傳送訊息以進行驗證並確認」核取方塊。

   c. 確認要驗證的屬性設定為傳送電子郵件訊息，驗證電子郵件地址。

   d. 在 [驗證屬性變更] 下，確認已選取預設選項：選取更新處於擱置狀態時保留原始屬性值，以及將更新擱置時的 Active 屬性值設定為 [電子郵件地址]。

   e. 在必要屬性下，確認根據先前選取的必要屬性顯示電子郵件。

   > **Important**
   >
   > 對於此示例應用程序，您的用戶池不得將 phone_number 設置為必需屬性。如果 phone_number 顯示為必要屬性，請檢閱並更新您先前的選擇：
   >
   > - 可選 MFA，僅用於用戶帳戶恢復消息的傳遞方法的電子郵件
   > - 發送電子郵件，驗證要驗證屬性的電子郵件地址

f. 保留所有其他選項為預設值，然後選擇「下一步」。

7. 在設定訊息傳遞中，您可以設定與 Amazon 簡單電子郵件服務和 Amazon 簡單通知服務的整合，以傳送電子郵件和簡訊給使用者以進行註冊、帳戶確認、MFA 和帳戶復原。如需詳細資訊，請參閱 Amazon Cognito 使用者集區的電子郵件設定 及 Amazon Cognito 使用者集區的簡訊設定。

   a. 對於電子郵件提供者，請選擇「使用 Cognito 傳送電子郵件」，並使用 Amazon Cognito 提供的預設電子郵件寄件者。這個低電子郵件量的設定已足以進行應用程式測試。您可以在使用 Amazon 簡易電子郵件服務 (Amazon SES) 驗證電子郵件地址並選擇使用 Amazon SES 傳送電子郵件後退貨。

   b. 對於 SMS，請選取建立新的 IAM 角色，然後輸入 IAM 角色名稱。這會建立一個角色，授予 Amazon Cognito 許可以傳送簡訊的權限。

   c. 保留所有其他選項為預設值，然後選擇「下一步」。

8. 在「整合您的應用程式」中，您可以為使用者集區命名、設定託管 UI，以及建立應用程式用戶端。如需詳細資訊，請參閱 使用託管 UI 添加應用程序客戶端。範例應用程式不使用託管 UI。

   a. 在 [使用者集區名稱] 下，輸入使用者集區名稱。

   b. 請勿選取 [使用 Cognito 託管的使用者介面]。

   c. 在初始應用程序客戶端下，確認應用程序類型設置為公共客戶端。

   d. 在 [用戶端密碼] 下，確認已選取 [不產生用戶端密碼]。

   e. 輸入 App client name (應用程式用戶端名稱)。

   f. 展開進階應用程式用戶端設定 新增`ALLOW_USER_PASSWORD_AUTH`至驗證流程清單。

   g. 保留所有其他選項為預設值，然後選擇「下一步」。

9. 在「檢閱並建立」畫面中檢閱您的選擇，並視需要修改任何選取項目。如果您對使用者集區設定感到滿意，請選擇 [建立使用者集區] 以繼續。

10. 在 [使用者集區] 頁面中，選擇新的使用者集區。

11. 在使用者集區概觀下，記下您的使用者集區 ID。您將在建立範例應用程式時提供此字串。

12. 選擇「應用程式整合」標籤，然後找到「應用程式用戶端和分析」 選取新的應用程式用戶端。記下您的客戶 ID。

建立一項使用者集區物件

1. 前往 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。

2. 選擇 User Pools (使用者集區)。

3. 在頁面右上角，選擇 Create a user pool (建立使用者集區)，以開始建立使用者集區精靈。

4. 在 Configure sign-in experience (設定登入體驗) 下，選擇您將與此使用者集區搭配使用的聯合提供者。如需詳細資訊，請參閱 透過第三方新增使用者集區登入。
5. 在 Configure security requirements (設定安全需求) 中，選擇您的密碼政策、多重要素驗證 (MFA) 需求，以及使用者帳戶復原選項。如需詳細資訊，請參閱 使用 Amazon Cognito 使用者集區安全性功能。
6. 在 Configure sign-up experience (設定註冊體驗) 中，決定新使用者在註冊時如何驗證其身分，以及在使用者註冊流程期間應該是必要或選用的屬性。如需詳細資訊，請參閱 管理使用者集區中的使用者。

   > **Important**
   >
   > 如果您在使用者集區中啟用使用者註冊，則網際網路上的任何人都可以註冊帳戶並登入您的應用程式。除非您想要開放您的應用程式供公開註冊，否則請勿在使用者集區中啟用自助註冊。若要變更此設定，請在使用者集區主控台的 [註冊] 體驗索引標籤中更新自助註冊，或更新 CreateUserPool或 UpdateUserPoolAPI 要求 AllowAdminCreateUserOnly中的值。
   >
   > 如需可在使用者集區中設定之安全功能的詳細資訊，請參閱 使用 Amazon Cognito 使用者集區安全性功能。

7. 在 Configure message delivery (設定訊息交付) 中，設定與 Amazon Simple Email Service 和 Amazon Simple Notification Service 的整合，以傳送電子郵件和 SMS 訊息給您的使用者，以便進行註冊、帳戶確認、MFA 和帳戶復原。如需詳細資訊，請參閱 Amazon Cognito 使用者集區的電子郵件設定 及 Amazon Cognito 使用者集區的簡訊設定。
8. 在 Integrate your app (整合您的應用程式) 中，命名您的使用者集區、設定託管 UI，以及建立應用程式用戶端。如需詳細資訊，請參閱 使用託管 UI 添加應用程序客戶端。
9. 在「檢閱並建立」畫面中檢閱您的選擇，並視需要修改任何選取項目。如果您滿意使用者集區設定，請選取 [建立使用者集區] 以繼續。

相關資源

如需使用者集區的詳細資訊，請參閱Amazon Cognito 使用者集區。

另請參閱：使用者集區身分驗證流程和將權杖用於使用者集區。

使用託管 UI 添加應用程序客戶端

建立使用者集區之後，您可以為應用程式建立應用程式用戶端，以顯示託管 UI 的內建網頁。在託管 UI 中，用戶可以：

- 註冊使用者設定檔。
- 使用第三方身分識別提供者登入。
- 使用或不使用多重要素驗證登入。
- 重設密碼。

若要建立用於託管 UI 登入的應用程式用戶端

1. 前往 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。如果您建立新的使用者集區，系統會在精靈期間提示您設定應用程式用戶端，並設定託管 UI。
4. 選擇使用者集區的 App integration (應用程式整合) 索引標籤。
5. 在網域旁，選擇動作，然後選取建立自訂網域或建立 Amazon Cognito 網域。如果您已設定使用者集區網域，請選擇刪除 Amazon Cognito 網域或刪除自訂網域，然後再建立新的自訂網域。
6. 輸入可用的網域字首來搭配 Amazon Cognito 網域使用。如需設定 Custom domain (自訂網域) 的詳細資訊，請參閱將自有網域用於託管 UI。
7. 選擇建立。
8. 導覽回相同使用者集區的 App integration (應用程式整合) 索引標籤，並找出 App clients (應用程式用戶端)。選擇 Create an app client (建立應用程式用戶端)。
9. 選擇 Application type (應用程式類型)。系統會根據您的選擇提供一些建議的設定。使用託管 UI 的應用程式是 Public client (公有用戶端)。
10. 輸入 App client name (應用程式用戶端名稱)。
11. 在本練習中，請選擇 Don't generate client secret (不要產生用戶端密碼)。用戶端密碼是由機密應用程式使用，從集中式應用程式對使用者進行身分驗證。在本練習中，您將向您的使用者顯示託管 UI 登入頁面，而且不需要用戶端密碼。
12. 選擇您將允許與應用程序一起使用的身份驗證流程。確保已選取 USER_SRP_AUTH。
13. 根據需要自訂 token expiration (權杖過期)、Advanced security configuration (進階安全組態)，以及 Attribute read and write permissions (屬性讀取和寫入許可)。如需詳細資訊，請參閱設定應用程式用戶端設定。

14. 為您的應用程式用戶端 Add a callback URL (新增回呼 URL)。這是託管 UI 身分驗證之後將您導向至的地方。在您能夠在應用程序中實現註銷之前，您無需添加允許的註銷 URL。

    若為 iOS 或 Android 應用程式，您可以使用 `myapp://`之類的回呼 URL。

15. 選取應用程式用戶端的 Identity providers (身分提供者)。至少啟用 Amazon Cognito 使用者集區作為提供者。

    > **Note**
    >
    > 要使用外部身份提供商（IdPs）（例如 Facebook，Amazon，谷歌和蘋果）以及通過 OpenID Connect（OIDC）或 SAML 登錄 IdPs，請先按照通過第三方添加用戶池登錄中所示進行配置。然後返回應用程序客戶端設置頁面以啟用它們。

16. 選擇 OAuth 2.0 Grant Types (OAuth 2.0 授予類型)。選取 Authorization code grant (授權程式碼授予)，傳回使用者集區權杖交換的授權程式碼。由於權杖不會直接向最終使用者公開，因此他們不太可能遭到入侵。但自訂應用程式需由後端交換使用者集區權杖所需的授權程式碼。基於安全考量，強烈建議您的行動應用程式僅使用授權碼授予流程，搭配 代碼交換證明金鑰 (PKCE)。

    選取 Implicit grant (隱含授予)，讓使用者集區 JSON Web 權杖 (JWT) 從 Amazon Cognito 傳回給您。您可以使用此流程，但不提供後端為權杖交換授權程式碼。還可以協助您除錯權杖。

    > **Note**
    >
    > 您可以啟用 Authorization code grant (授權碼授予) 以及 Implicit code grant (隱含程式碼授予)，並視需要使用授予。
    > 唯有當您的應用程式需要代表自己，而不是代表使用者請求存取權杖，才選取 Client credentials (用戶端憑證)。

17. 除非您特別想要排除某個選項，否則請選取所有 OpenID Connect scopes (OpenID Connect 範圍)。
18. 選擇您已配置的任何自定義範圍。自訂範圍通常與機密用戶端搭配使用。
19. 選擇建立。

查看您的登入頁面

在您的應用程序客戶端頁面中，選擇查看託管 UI 以打開新的瀏覽器選項卡，該頁面已預先填充應用程序客戶端 ID，範圍，授予和回調 URL 參數的登錄頁面。

您可以使用以下 URL 手動檢視託管 UI 的登入頁面。請記下 `response_type`。在此情況下，以 response_type=code 授予授權碼。

```
https://your_domain/login?
response_type=code&client_id=your_app_client_id&redirect_uri=your_callback_url
```

您可以使用以下 URL 檢視託管 UI 登入網頁，利用 response_type=token 授予隱含程式碼。成功登入後，Amazon Cognito 會將使用者集區權杖傳回至 Web 瀏覽器的網址列。

```
https://your_domain/login?
response_type=token&client_id=your_app_client_id&redirect_uri=your_callback_url
```

您可以在回應中的 `#idtoken=` 參數之後找到 JSON Web Token (JWT) 身分權杖。

以下 URL 是來自隱含授予請求的回應範例。您的身分權杖字串將更長。

```
https://www.example.com/
#id_token=123456789tokens123456789&expires_in=3600&token_type=Bearer
```

Amazon Cognito 使用者集區權杖以 RS256 演算法登入。您可以使用解碼和驗證用戶池令牌 AWS Lambda。如需詳細資訊，請參閱在網站上解碼和驗證 Amazon Cognito JWT 權杖。 AWS GitHub

您的網域會顯示在 Domain name (網域名稱) 頁面中。您的應用程式用戶端 ID 和回呼 URL 會顯示在 General settings (一般設定) 頁面中。如果您在主控台中所做的變更沒有立即顯示，請稍候幾分鐘，然後重新整理瀏覽器。

新增社交登入至使用者集區 (選用)

您可以讓您的應用程式使用者透過社交身分提供者 (如 Facebook、Google、Amazon 和 Apple) 登入 (IdP)。無論您的使用者直接或透過第三方間接登入，所有使用者在使用者集區中都有設定檔。如果您不想新增透過社交登入的身分提供者，請略過此步驟。

使用社交 IdP 註冊

與 Amazon Cognito 建立社交 IdP 之前，您必須將您的應用程式註冊至社交 IdP 以接收用戶端 ID 和用戶端密碼。

向 Facebook 註冊應用程式

1. 向 Facebook 建立開發人員帳戶。
2. 使用您的 Facebook 憑證登入。
3. 從 My Apps (我的應用程式) 選單中，選擇 Create New App (建立新的應用程式)。

   如果您沒有現有的 Facebook 應用程序，則會看到不同的選項。選擇 Create App (建立應用程式)。
4. 在建立應用程式頁面上，選擇應用程式的使用案例，然後選擇下一步。
5. 輸入您 Facebook 應用程式的名稱，然後選擇建立應用程式 ID。
6. 在左側導覽列中，選擇應用程式設定，然後選擇基本。
7. 記下 App ID (應用程式 ID) 和 App Secret (應用程式秘密)。您會在下一節中用到它們。
8. 從頁面底部選擇 + 新增平台。
9. 在 [選取平台] 畫面上，選取您的平台，然後選擇 [下一步]。
10. 選擇儲存變更。
11. 對於 App Domains (應用程式網域)，輸入您的使用者集區網域。

    ```
    https://your_user_pool_domain
    ```
12. 選擇儲存變更。
13. 從導覽列選擇「產品」，然後選擇「從 Facebook 登入設定」。
14. 從 Facebook 登入、設定選單中，選擇設定。

    在 Valid OAuth Redirect URIs (有效的 OAuth 重新引導 URI) 中輸入您的重新導向 URL。重新導向 URL 是由您的使用者集區網域和`/oauth2/idpresponse`端點所組成。

    ```
    https://your_user_pool_domain/oauth2/idpresponse
    ```
15. 選擇 Save changes (儲存變更)。

向 Amazon 註冊應用程式

1. 向 Amazon 建立開發人員帳戶。
2. 使用您的 Amazon 憑證登入。
3. 您需要建立 Amazon 安全性設定檔以便接收 Amazon 用戶端 ID 和用戶端秘密。

從頁面頂端的導覽列選擇「應用程式和服務」，然後選擇「Login with Amazon」。

4. 選擇 Create a Security Profile (建立安全性設定檔)。
5. 輸入 Security Profile Name (安全設定檔名稱)、Security Profile Description (安全設定檔描述) 和 Consent Privacy Notice URL (同意隱私權聲明 URL)。
6. 選擇 Save (儲存)。
7. 選擇 Client ID (用戶端 ID) 和 Client Secret (用戶端秘密)，以顯示用戶端 ID 和秘密。您會在下一節中用到它們。
8. 將滑鼠移到齒輪圖示上方並選擇 Web Settings (Web 設定)，然後選擇 Edit (編輯)。
9. 在 Allowed Origins (允許的來源) 中輸入您的使用者集區網域。

```
https://<your-user-pool-domain>
```

10. 在 Allowed Return URLs (允許的傳回 URL) 中輸入具有 `/oauth2/idpresponse` 端點的使用者集區網域。

```
https://<your-user-pool-domain>/oauth2/idpresponse
```

11. 選擇 Save (儲存)。

向 Google 註冊應用程式

如需 Google Cloud Platform 中 OAuth 2.0 的詳細資訊，請參閱 Google Workspace for Developers 文件中的瞭解身分驗證和授權。

1. 向 Google 建立開發人員帳戶。
2. 登入 Google Cloud Platform 主控台。
3. 在頂端導覽列中，選擇 Select a project (選取專案)。如果您在 Google 平台中已經有專案，則此功能表將改為顯示您的預設專案。
4. 選擇 NEW PROJECT (新專案)。
5. 輸入專案的名稱，然後選擇 CREATE (建立)。
6. 在左側導航欄上，選擇 API 和服務，然後選擇 Oauth 同意屏幕。
7. 輸入應用程式資訊、應用程式網域、授權網域和開發人員聯絡資訊。您的授權網域必須包含`amazoncognito.com`自訂網域的根目錄。例如：`example.com`。選擇 SAVE AND CONTINUE (儲存並繼續)。
8. 1. 在 [範圍] 下，選擇 [新增或移除範圍]，然後至少選擇下列 OAuth 範圍。

1. `.../auth/userinfo.email`
2. `.../auth/userinfo.profile`
3. openid

9. 在 Test users (測試使用者) 下方，選擇 Add users (新增使用者)。輸入您的電子郵件地址和任何其他授權測試使用者，然後選擇 [儲存並繼續]。
10. 再次展開左側導覽列，選擇 [API 和服務]，然後選擇 [認證]。
11. 選擇 [建立認證]，然後選擇 [OAuth 用戶端識別碼]。
12. 選擇 Application type (應用程式類型)，並為您的用戶端取一個 Name (名稱)。
13. 在 [授權 JavaScript 來源] 下，選擇 [新增 URI]。輸入您的使用者集區網域。

```
https://<your-user-pool-domain>
```

14. 在 Authorized redirect URIs (授權的重新導向 URI) 下方，選擇 ADD URI (新增 URI)。輸入您使用者集區網域的 `/oauth2/idpresponse` 端點路徑。

```
https://<your-user-pool-domain>/oauth2/idpresponse
```

15. 選擇 CREATE (建立)。
16. 安全地儲存 Google 在您的用戶端 ID和您的用戶端密碼下顯示的值。當您新增 Google IdP 時，請將這些值提供給 Amazon Cognito。

向 Apple 註冊應用程式

如需設定 Sign of Apple 的詳細資訊，請參閱 Apple Developer 文件中的設定 Sign in with Apple 的環境。

1. 向 Apple 建立開發人員帳戶。
2. 使用您的 Apple 憑證登入。
3. 在左側導覽列上，選擇 Certificates, Identifiers & Profiles (憑證、識別碼與設定檔)。
4. 在左側導覽列上，選擇 Identifiers (識別碼)。
5. 在 Identifiers (識別碼) 頁面上，選擇 + 圖示。
6. 在 Register a New Identifier (註冊新的識別碼) 頁面上，選擇 App IDs (應用程式 ID)，然後選擇 Continue (繼續)。
7. 在 [選取類型] 頁面上，選擇 [應用程式]，然後選擇 [繼續]。

8. 在 Register an App ID (註冊應用程式 ID) 頁面上，執行下列操作：

    1. 在 Description (描述) 下方輸入描述。
    2. 在 App ID Prefix (應用程式 ID 字首) 下方，輸入 Bundle ID (套件 ID)。記下 App ID Prefix (應用程式 ID 字首) 下的值。在 步驟 2：新增社交 IdP 到您的使用者集區 中選擇 Apple 作為您的身分提供者之後，您將會使用此值。
    3. 在 Capabilities (功能) 下方，選擇 Sign In with Apple，然後選擇 Edit (編輯)。
    4. 在 Sign in with Apple: App ID Configuration (Sign in with Apple：應用程式 ID 組態) 頁面上，選擇將應用程式設定為主要應用程式或與其他應用程式 ID 群組，然後選擇 Save (儲存)。
    5. 選擇繼續。
9. 在 Confirm your App ID (確認您的應用程式 ID) 頁面上，選擇 Register (註冊)。
10. 在 Identifiers (識別碼) 頁面上，選擇 + 圖示。
11. 在 Register a New Identifier (註冊新的識別碼) 頁面上，選擇 Services IDs (服務 ID)，然後選擇 Continue (繼續)。
12. 在 Register a Services ID (註冊服務 ID) 頁面上，執行下列操作：

    1. 在 Description (描述) 下方輸入描述。
    2. 在 Identifier (識別符) 中，輸入識別符。請記下此服務 ID，因為在中選擇 Apple 作為您的身分供應商後，您將需要此值步驟 2：新增社交 IdP 到您的使用者集區。
    3. 選擇 Continue (繼續)，然後選擇 Register (註冊)。
13. 選擇您剛剛從「識別碼」頁面建立的服務 ID。

    1. 選取 Sign In with Apple，然後選擇 Configure (設定)。
    2. 在 Web Authentication Configuration (Web 身分驗證組態) 頁面上，選取您稍早建立的應用程式 ID 作為 Primary App ID (主要應用程式 ID)。
    3. 在 Website URLs (網站 URL) 旁邊選擇 + 圖示。
    4. 在 Domains and subdomains (網域與子網域) 下方，輸入不含 `https://` 字首的使用者集區網域。

        ```
        <your-user-pool-domain>
        ```

    5. 在 Return URLs (傳回 URL) 下，輸入您使用者集區網域的 `/oauth2/idpresponse` 端點路徑。

        ```
        https://<your-user-pool-domain>/oauth2/idpresponse
        ```

6. 選擇 [下一步]，然後選擇 [完成]。您不須驗證網域。
7. 選擇 Continue (繼續)，然後選擇 Save (儲存)。

14. 在左側導覽列上，選擇 Keys (金鑰)。
15. 在 Keys (金鑰) 頁面上，選擇 + 圖示。
16. 在 Register a New Key (註冊新的金鑰) 頁面上，執行下列操作：

    1. 在 Key Name (金鑰名稱) 下，輸入金鑰名稱。
    2. 選擇 Sign In with Apple，然後選擇 Configure (設定)。
    3. 在 [設定金鑰] 頁面上，選取您先前建立的應用程式 ID 做為主要應用程式 ID。選擇儲存。
    4. 選擇 Continue (繼續)，然後選擇 Register (註冊)。

17. 在 [下載您的金鑰] 頁面上，選擇 [下載] 以下載私密金鑰，記下顯示的金鑰 ID，然後選擇 [完成]。您在 步驟 2：新增社交 IdP 到您的使用者集區 中選擇 Apple 當作身分提供者後，需有此頁面上顯示的私有金鑰和 Key ID (金鑰 ID) 值。

新增社交 IdP 到您的使用者集區

在本節中，您會使用上一節的用戶端 ID 和用戶端密碼，在使用者集區設定社交 IdP。

若要設定使用者集區社交身分提供者 AWS Management Console

1. 前往 Amazon Cognito 主控台。系統可能會提示您輸 AWS 入認證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。
4. 選擇 Sign-in experience (登入體驗) 索引標籤。找到 Federated sign-in (聯合登入)，然後選取 Add an identity provider (新增身分提供者)。
5. 選擇社交身分提供者：Facebook、Google、Login with Amazon 或 Sign in with Apple。
6. 根據您選擇的社交身分提供者，從以下步驟中選擇：

   - Google 和 Login with Amazon — 輸入上一節中生成的應用程序客戶端 ID 和應用程序客戶端密鑰。
   - Facebook — 輸入上一節產生的應用程式用戶端 ID 和應用程式用戶端密碼，然後選擇 API 版本 (例如 2.12 版)。我們建議您選擇最新的可能版本 — 每個 Facebook API 都有生命週期和淘汰日期。Facebook 範圍和屬性可以根據 API 版本而有所不同。建議您使用 Facebook 測試您的社交身分登入，以確保聯合使用正常運作。
   - 使用 Apple 登入 — 輸入上一節產生的服務 ID、團隊 ID、金鑰 ID 和私密金鑰。

7. 輸入您要使用的「已授權範圍」的名稱。範圍可定義您要透過應用程式來存取的使用者屬性 (例如 `name` 和 `email`)。若為 Facebook，這些屬性應以逗號分隔。若為 Google 和登入 Amazon，則應以空格分隔。若為 Sign in with Apple，請針對您想要存取的範圍勾選核取方塊。

社交身分提供者	範例範圍
Facebook	`public_profile, email`
Google	`profile email openid`
登入 Amazon	`profile postal_code`
使用 Apple 登入	`email name`

   會向您的應用程式使用者提示同意提供這些屬性給應用程式。如需社交供應商範圍的詳細資訊，請參閱 Google、Facebook、Login with Amazon 或 Sign in with Apple 提供的說明文件。

   透過 Sign in with Apple，以下是可能未傳回範圍的使用者案例：

   - 最終用戶在離開 Apple 的登錄頁面後遇到故障（這可能是來自 Amazon Cognito 內部的內部故障或開發人員編寫的任何內容）。
   - 服務 ID 識別碼會跨使用者集區和/或其他認證服務使用。
   - 開發人員在使用者登入後新增額外的範圍。使用者只有在其進行身分驗證與重新整理其權杖時才會擷取新資訊。
   - 開發人員刪除用戶，然後用戶再次登錄，而不從其 Apple ID 配置文件中刪除該應用程序。
8. 將身分提供者的屬性映射至您的使用者集區。如需詳細資訊，請參閱 對應的須知事項。
9. 選擇 Create (建立)。
10. 從 App client integration (應用程式用戶端整合) 索引標籤選擇清單中的 App clients (應用程式用戶端) 之一，和 Edit hosted UI settings (編輯託管 UI 設定)。將新的社交身分提供者新增至 Identity providers (身分提供者) 下的應用程式用戶端。
11. 選擇儲存變更。

測試社交 IdP 組態

您可以使用前兩節的元素建立登入 URL。用來測試社交 IdP 組態。

```
https://mydomain.us-east-1.amazoncognito.com/login?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

您可以在使用者集區 Domain name (網域名稱) 主控台頁面上尋找您的網域。client_id 位於 App client settings (應用程式用戶端設定) 頁面上。將您的回呼 URL 使用於 redirect_uri 參數。這是您的使用者成功身分驗證後會被重新導向的頁面 URL。

> **Note**
>
> 身分驗證請求若未在 5 分鐘內完成，Amazon Cognito 會將該請求取消，並將使用者重新導向至託管的 UI。此頁面人會顯示「Something went wrong」錯誤訊息。

以 SAML 身分提供者新增登入至使用者集區 (選用)

您可以讓您的應用程式使用者透過 SAML 身分提供者 (IdP) 登入。無論您的使用者直接或透過第三方間接登入，所有使用者在使用者集區中都有設定檔。如果您不想新增透過 SAML 身分提供者，請略過此步驟。

如需詳細資訊，請參閱 搭配使用者集區使用 SAML 身分識別提供者。

您必須更新您的 SAML 身分識別提供者並設定您的使用者集區。如需如何將使用者集區新增為 SAML 2.0 身分識別提供者的信賴憑證者或應用程式的相關資訊，請參閱 SAML 身分識別提供者的文件。

您也必須為您的 SAML 身分識別提供者提供宣告消費者服務 (ACS) 端點。在 SAML 身分提供者中為 SAML 2.0 POST 繫結設定使用者集區網域中的下列端點。如需使用者集區網域的詳細資訊，請參閱設定使用者集區網域。

```
https://Your user pool domain/saml2/idpresponse
With an Amazon Cognito domain:
https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/idpresponse
With a custom domain:
https://Your custom domain/saml2/idpresponse
```

您可以在 Amazon Cognito 主控台的 [網域名稱] 索引標籤上找到您的網域前置詞和使用者集區的區域值。

對於某些 SAML 身分識別提供者，您還需要以下格式提供服務提供者 (SP)urn，也稱為對象 URI 或 SP 實體 ID：

```
urn:amazon:cognito:sp:<yourUserPoolID>
```

您可以在 Amazon Cognito 主控台中的 General settings (一般設定) 標籤上，找到您的使用者集區 ID。

您也應配置 SAML 身分提供者，以為您的使用者集區中的任何必要屬性提供屬性值。通常 `email` 為使用者集區必要的屬性。此情況下，SAML 身分提供者應在 SAML 聲明中提供 `email` 值 (宣告)。

Amazon Cognito 使用者集區支援與 POST 繫結端點進行 SAML 2.0 聯合。這樣就不需要您的應用程式擷取或剖析 SAML 判斷提示回應，因為使用者集區會透過使用者代理直接接收來自身分識別提供者的 SAML 回應。

在您的使用者集區中配置 SAML 2.0 身分提供者

1. 前往 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。
4. 選擇 Sign-in experience (登入體驗) 索引標籤。找到 Federated sign-in (聯合登入)，然後選取 Add an identity provider (新增身分提供者)。
5. 選擇 SAML 社交身分提供者。
6. 輸入以逗號分隔的 Identifiers (識別符)。識別碼告訴 Amazon Cognito，它應該檢查使用者在登入時輸入的電子郵件地址。然後它將它們定向到與其域對應的提供商。
7. 如果您希望 Amazon Cognito 在使用者登出時傳送已簽署的登出請求給您的供應商，則請選擇 Add sign-out flow (新增登出流程)。您必須設定您的 SAML 2.0 身分提供者，以便將登出回應傳送至您設定託管 UI 時建立的 `https://<your Amazon Cognito domain>/saml2/logout` 端點。`saml2/logout`端點使用 POST 繫結。

   > **Note**
   >
   > 如果選取此選項，且您的 SAML 身分提供者預期簽署的登出請求，您還需要設定 Amazon Cognito 與您的 SAML IdP 一起提供的簽署憑證。
   > SAML IdP 將處理已簽署的登出要求，並將您的使用者從 Amazon Cognito 工作階段登出。

8. 選擇 Metadata document source (中繼資料文件來源)。如果您的身分提供者以公有 URL 提供 SAML 中繼資料，則可以選擇 Metadata document URL (中繼資料文件 URL)，然後輸入該公有 URL。否則，請選擇 Upload metadata document (上傳中繼資料文件)，然後選取您先前從供應商處下載的中繼資料檔案。

> **Note**
>
> 如果您的提供者具有公開端點，建議您輸入中繼資料文件 URL，而不是上傳檔案。這可讓 Amazon Cognito 自動重新整理中繼資料。通常每 6 小時或是在中繼資料過期之前 (取較早的時間) 重新整理中繼資料。

9. 選擇 Map attributes between your SAML provider and your app (在 SAML 供應商與應用程式之間映射屬性)，將 SAML 供應商屬性映射至使用者集區中的使用者描述檔。在屬性映射中包含您的使用者集區必要屬性。

   例如，當您選擇 User pool attribute (使用者集區屬性) `email` 時，隨著身分提供者 SAML 聲明中顯示的 SAML 屬性名稱輸入 SAML 屬性名稱。您的身分提供者可能會提供範例 SAML 聲明以供參考。有些身分供應商會使用簡單的名稱，例如 `email`，而其他供應商則會使用 URL 格式的屬性名稱，例如下列範例：

   ```
   http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
   ```

10. 選擇建立。

開始使用 Amazon Cognito 身分集區

透過 Amazon Cognito 身分集區，您可以為使用者建立唯一身分及指派許可。您的身分集區可以包含：

- Amazon Cognito 使用者集區中的使用者
- 使用外部身分提供者 (例如 Facebook、Google、Apple 或 OIDC 或 SAML 身分提供者) 來驗證身分的使用者。
- 透過您自己現有的驗證程序來驗證身分的使用者

透過身分集區，您可以取得具有定義權限的臨時 AWS 登入資料，以直接存取其他資源 AWS 服務 或透過 Amazon API Gateway 存取資源。

主題

在 Amazon Cognito 中建立身分集區

您可以透過 Amazon Cognito 主控台建立身分集區，或者您可以使用 AWS Command Line Interface (CLI) 或 Amazon Cognito API。

在主控台中建立新的身分集區

1. 登入 Amazon Cognito 主控台，然後選取 身分池。
2. 選擇 建立身分池。
3. 在 設定身分池信任 中，為您的身分池選擇 已驗證存取、訪客存取 設定，或同時選擇。
    - 如果您選擇 已驗證存取，請選取一或多個要設定為身分池中已驗證身分來源的 身分類型。如果您設定 自訂開發人員提供者，則建立身分池後無法修改或刪除。
4. 在 設定許可 中，為身分池中的身分驗證使用者或訪客使用者選擇預設 IAM 角色。
    a. 如果您希望 Amazon Cognito 為您建立具有基本許可和與身分池具有信任關係的新角色，請選擇 建立新 IAM 角色。輸入 IAM 角色名稱 以識別您的新角色，例如

myidentitypool_authenticatedrole。選取 檢視政策文件 以檢閱 Amazon Cognito 指派給您新 IAM 角色的許可。

b. 如果您想要使用的角色中已有角色，則可以選擇使用現有的 AWS 帳戶 IAM 角色。您必須設定 IAM 角色信任政策以包含 cognito-identity.amazonaws.com。將您的角色信任政策設定為僅在 Amazon Cognito 提供證據，表明請求來源為特定身分池中已驗證的使用者時，才允許 Amazon Cognito 擔任該角色。如需詳細資訊，請參閱 角色信任和許可。

5. 在 [Connect 身分識別提供者] 中，輸入您在設定身分識別集區信任中選擇的身分識別提供者 (IdPs) 的詳細資料。系統可能會要求您提供 OAuth 應用程式用戶端資訊、選擇 Amazon Cognito 使用者集區、選擇 IAM IdP，或輸入開發人員供應商的自訂識別符。

   a. 為每個 IdP 選擇 角色設定。您可以為該 IdP 使用者指派設定 已驗證角色 時的 預設角色，或您可以 選擇具有規則的角色。使用 Amazon Cognito 使用者集區 IdP，您還可以 選擇權杖中具有 preferred_role 的角色。如需 cognito:preferred_role 宣告的詳細資訊，請參閱 指定優先順序值給群組。

      i. 如果您選擇 使用規則選擇角色，請輸入使用者身分驗證的 宣告 來源、比較宣告的 操作員、導致符合角色選擇的 值，以及當符合 角色指派 時您要指派的 角色。選取 新增另一項 以根據不同的條件建立其他規則。

      ii. 選擇 角色解析。當您的使用者宣告與您的規則不符時，您可以拒絕憑證或向 已驗證角色發出憑證。

   b. 為每個 IdP 分別設定 存取控制屬性。存取控制屬性會將使用者宣告映射至 Amazon Cognito 套用至其臨時工作階段的委託人標籤。您可以建立 IAM 政策，根據您套用至其工作階段的索引標籤篩選使用者存取權限。

      i. 若不套用主要索引標籤，請選擇 非作用中。

      ii. 若要根據 sub 和 aud 宣告套用主要索引標籤，請選擇 使用預設對應。

      iii. 若要建立您自己的自訂屬性結構描述至主要索引標籤，請選擇 使用自訂對應。然後，輸入您要從每個 宣告 中獲取的 標籤金鑰，顯示於索引標籤當中。

6. 在 設定屬性 中，在 身分池名稱 下輸入一個 名稱。

7. 在 基本 (傳統) 身分驗證 下，選擇是否要 啟用基本流程。啟用基本流程後，您可以略過為您所做的角色選擇 IdPs 並AssumeRoleWithWebIdentity直接呼叫。如需詳細資訊，請參閱 身分集區 (聯合身分) 驗證流程。

8. 如果您要將 標籤 套用至身分池，請在索引 標籤 底下選擇 新增標籤。

9. 在 檢閱和建立 中，確認您為新身分池所做的選擇。選取 編輯 以返回精靈並變更任何設定。當您完成時，請選取 建立身分池。

設定 SDK

若要使用 Amazon Cognito 身分識別集區 AWS Amplify，請設定 AWS SDK for Java、或. AWS SDK for .NET如需詳細資訊，請參閱下列主題。

- 在AWS SDK for Java 開發人員指南 JavaScript中設定的 SDK
- 《Amplify 開發人員中心》中的 Amplify 文件
- 《AWS SDK for .NET 開發人員指南》中的 Amazon Cognito 憑證提供者

整合身分提供者

Amazon Cognito 身分池 (聯合身分) 支援透過 Amazon Cognito 使用者集區、聯合身分提供者 (包括 Amazon、Facebook、Google、Apple 和 SAML 身分提供者) 以及未驗證的身分，進行使用者身分驗證。此功能也支援開發人員驗證的身分，可讓您透過自己的後端身分驗證程序來註冊及驗證使用者。

若要進一步了解如何使用 Amazon Cognito 使用者集區來建立自己的使用者目錄，請參閱 Amazon Cognito 使用者集區，以及 登入後 AWS 服務 使用身分集區存取。

若要進一步了解如何使用外部身分供應商，請參閱外部身分提供者身分集區。

若要進一步了解如何整合自己的後端身分驗證程序，請參閱開發人員驗證的身分。

取得憑證

Amazon Cognito 身分集區為訪客 (未驗證) 的使用者以及已驗證和接收權杖的使用者提供臨時 AWS 登入資料。使用這些 AWS 登入資料，您的應用程式可以 AWS 透過 Amazon API Gateway 安全地在內 AWS 外存取後端。請參閱取得憑證。

Amazon Cognito 的引導式設定選項

您可能想要在結構化的引導體驗中評估 Amazon Cognito 的功能。以下是一些外部資源，可提供量身打造的使用者集區和身分識別集區體驗。

完成一個工作坊

AWS 工作坊工作室會舉辦一個工作坊，引導您完成大多數 Amazon Cognito 功能的設定。這些功能包括使用者集區 API、使用者集區託管的 UI、身分集區和安全性設定。

從示例添加應用程序代碼

本指南中的程式碼範例章節包含可與使用者集區和身分集區搭配使用的應用程式程式碼。程式碼範例章節的使用者集區區段包含涵蓋個別作業的簡短片段，以及較長的範例， end-to-end 例如各種程式設計語言的範例應用程式。

使用建立完整堆疊應用程式 AWS Amplify

AWS Amplify是一個 AWS 服務 適用於想要開發和託管應用程序和用戶界面的開發人員。Amazon Cognito 是 Amplify 的身份驗證組件。當您將身份驗證新增至應用程式時，Amplify 可以自動化 Amazon Cognito 使用者集區和身分集區資源的部署。另請參閱 將 Amazon Cognito 身分驗證和授權與 Web 和行動應用程式整合。

更多 Amazon Cognito 應用程序資源 GitHub

- 使用 .NET 的 Amazon Cognito 份驗證流程示例
- Amazon Cognito 無密碼身份驗證
- PetStoreAmazon 驗證許可的示例
- 使用 ABAC + 身份池訪 AWS 問資源的示例反應應用程序
- 使用 CDK Amazon Cognito 和 API Gateway 為基礎的機器到機器 AWS 授權
- 使用 Amazon Cognito、API Gateway 和 IAM 建立細粒度授權
- CloudFront授權條款 @edge

更多工作坊

- 使用 Amazon Cognito 實施無密碼身份驗證 WebAuthn
- 具有 Amazon Cognito 使用者集區的多租戶 SaaS 身分

- Amazon Cognito JWT 深入探討

將 Amazon Cognito 身分驗證和授權與 Web 和行動應用程式整合

當您將應用程式與 Amazon Cognito 應用程式用戶端整合時，您可以調用 API 操作以對使用者進行身分驗證和授權。我們建議您使用 AWS AmplifyAmazon Cognito 與您的網路和行動應用程式整合。 AWS Amplify 這是一個完整的解決方案，可讓前端 Web 和行動開發人員輕鬆建置、連接和託管 fullstack 應用程式 AWS，並可隨著使用案 AWS 服務 例的發展而靈活運用的廣度。Amplify Auth 主要使用 Amazon Cognito 來建置驗證功能。

主題

- 使用驗證 AWS Amplify
- 使用 AWS SDK 進行身分驗證
- 使用 Amazon Verified Permissions 進行授權

Amazon Cognito 的典型實作會混合使用視覺化工具和 API。Amazon Cognito 主控台是用於設定和管理 Amazon Cognito 使用者集區和身分集區的視覺化介面。託管使用者介面是 ready-to-use 網路型登入應用程式，可快速測試和部署 Amazon Cognito 使用者集區。此外，在大多數 Amazon Cognito 部署中，您必須在應用程式中新增程式碼，才能與使用者集區和身分集區互動。例如，您的應用程式可能會調用託管 UI 來進行使用者登入，然後從應用程式程式碼呼叫權杖端點，以便用您的使用者授權碼交換權杖。然後，您的應用程式必須解譯和儲存使用者權杖，並在適當情況下出示權杖以進行驗證和授權。Amplify 針對這些程序新增了具有內建功能的引導式整合工具。

您也可以完全使用程式碼建置 Amazon Cognito 資源。若要開始使用您的自訂應用程式程式碼，請造訪適用 AWS SDK 的 Amazon Cognito 程式碼範例。若要作為 OpenID Connect 身分提供者與 Amazon Cognito 整合，請使用 OpenID Connect 開發人員工具。

在使用 Amazon Cognito 身分驗證和授權之前，請選擇一個應用程式平台並準備好要與服務進行整合的程式碼。如需了解可用的平台，請參閱 使用 AWS SDK 進行身分驗證。 AWS CLI 這是 Amazon Cognito 和其他的命令列開發套件 AWS 服務，是開始熟悉 Amazon Cognito API 的有價值的地方。

> **Note**
>
> 有些 Amazon Cognito 元件只能使用 API 進行設定。例如，您只能使用更新或 UpdateUserPoolAPI 請求中UserPool類別`LambdaConfig`屬性的請求來設定使用者集區自訂 SMS CreateUserPool或電子郵件寄件者 Lambda 觸發程序。

Amazon Cognito 使用者集區 API 會與多種類別的 API 操作共用其命名空間。有一個類別用於設定使用者集區及其程序、身分提供者和使用者。另一個類別包括未經驗證的操作，可讓公有用戶端中的使用者登入、登出和管理其設定檔。API 作業的最後一個類別會在機密的伺服器端用戶端中執行您使用自己的 AWS 認證授權的使用者作業。在您開始實作應用程式程式碼之前，必須先知道預定的應用程式架構。如需詳細資訊，請參閱 使用 Amazon Cognito 使用者集區 API 和使用者集區端點。

使用驗證 AWS Amplify

AWS Amplify 是構建 Web 和移動應用程序的完整解決方案。有了 Amplify，您就可以使用 Amplify 程式庫連線到現有資源，也可以使用 Amplify 命令列介面 (CLI) 建立和設定新資源。Amplify 也有連線 UI 元件，如 Authenticator，可用於設定和自訂您應用程式中的登入和註冊體驗。

若要在前端應用程式中使用 Amplify 驗證功能，請參閱下列各平台的文件。

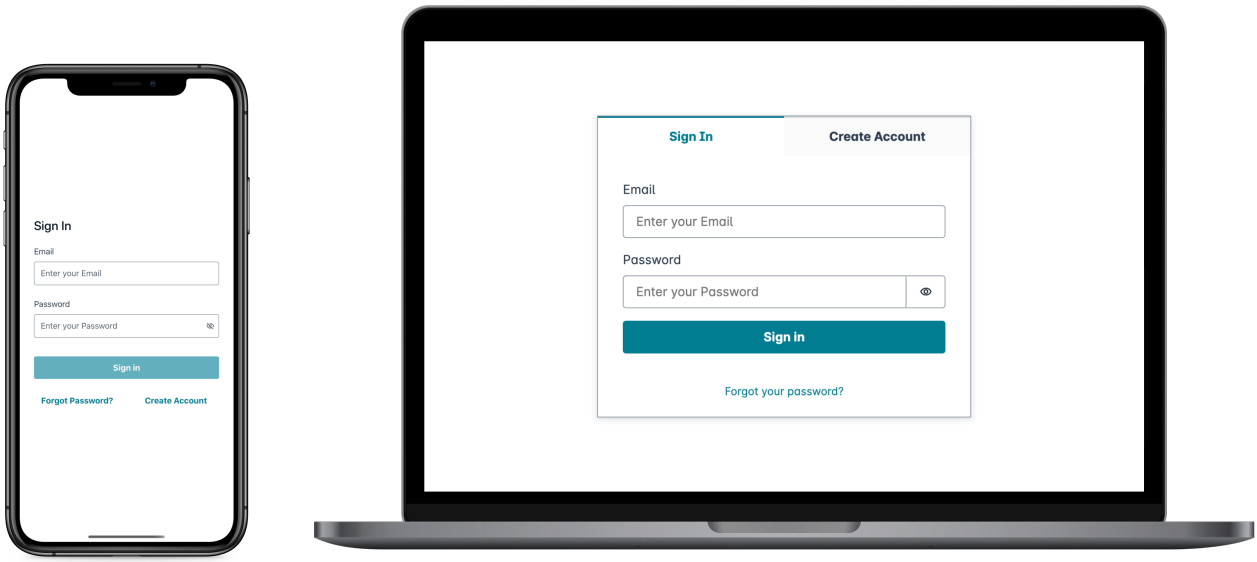
- Amplify 驗證 JavaScript
- iOS 的 Amplify 身分驗證
- Android 的 Amplify 身分驗證
- Flutter 的 Amplify 身分驗證

Amplify 程式庫是開放原始碼的，可在上GitHub使用。若要深入了解 Amplify Auth 如何實作 Amazon Cognito 驗證，請造訪下列程式庫。

- amplify-js
- amplify-swift
- amplify-flutter
- amplify-android

使用 Amplify 建立使用者介面 (UI)

Amazon Cognito 使用者集區託管 UI 能夠滿足 Web 或行動應用程式驗證前端的基本需要。若要自訂使用者介面 (UI) 使其超出託管 UI 所容納的參數，請建立自訂應用程式。Amplify UI 是各種不同語言的可自訂前端元件的集合。



若要開始使用您的自訂驗證元件，請造訪下列 Authenticator 元件的文件。

- Authenticator for Android
- Authenticator for Angular
- Authenticator for Flutter
- Authenticator for React
- Authenticator for React Native
- Authenticator for Swift
- Authenticator for Vue

使用 AWS SDK 進行身分驗證

若要使用安全的後端建立與 Amazon Cognito 互動的身分微服務，請使用您選擇的語言使用 AWS 開發套件連線到 Amazon Cognito 使用者集區和 Amazon Cognito 身分集區 API。

如需各個 API 操作的詳細內容，請參閱 Amazon Cognito 使用者集區 API 參考與 Amazon Cognito API 參考。這些文件包含另請參閱部分，其中包含可在支援平台中使用各種 SDK 的資源。

- AWS 命令列介面

- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2 的开发
- AWS 適用於的 SDK JavaScript
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

使用 Amazon Verified Permissions 進行授權

Amazon Verified Permissions 是一項授權服務，適用於您建置的應用程式。當您新增 Amazon Cognito 使用者集區做為身分來源時，您的應用程式就可以將使用者集區存取權或身分 (ID) 權杖傳遞給 Verified Permissions 以決定允許或拒絕。Verified Permissions 會根據您使用 Cedar 政策語言撰寫的政策來考量您使用者的屬性和請求內容。請求內容可包括所請求文件、影像或其他資源的識別符，以及您的使用者想要對資源執行的動作。

您的應用程序可以將用戶的身份或訪問令牌提供給「已驗證的權限」IsAuthorizedWithToken或BatchIsAuthorizedWithTokenAPI 請求。這些 API 作業會接受您的使用者，`Principal`並針對他們想要存取`Action`的`Resource`項目做出授權決策。其他自訂`Context`可以有助於詳細的存取決策。

當您的應用程式在 `IsAuthorizedWithToken` API 請求中顯示權杖時，Verified Permissions 就會執行以下驗證。

1. 您的使用者集區是針對所請求的政策存放區設定的 Verified Permissions 身分來源。
2. `client_id` 或 `aud` 宣告分別位在您的存取或身分權杖中，兩者之一會符合您提供給 Verified Permissions 的使用者集區應用程式用戶端 ID。若要驗證此宣告，您必須在您的 Verified Permissions 身分來源中設定用戶端 ID 驗證。
3. 您的權杖未過期。
4. 令牌中`token_use`聲明的值與您傳遞給的參數匹配`IsAuthorizedWithToken`。`access`如果您將其傳遞給`accessToken`參數，並且將其傳遞給參數，`id`則該`token_use`為`identityToken`聲明必須是。
5. 權杖中的簽名來自您的使用者集區已發布的 JSON Web 金鑰 (JWK)。您可以在 `https://cognito-idp.`*`Region`*`.amazonaws.com/`*`your user pool ID`*`/.well-known/jwks.json` 檢視您的 JWK。

撤銷的權杖和刪除的使用者

Verified Permissions 只會驗證從您的身分來源得知的資訊，以及您使用者的權杖到期時間資訊。Verified Permissions 不會檢查權杖是否撤銷或使用者是否存在。即使您撤銷了使用者的權杖，或是從使用者集區刪除了使用者的設定檔，在權杖過期之前，Verified Permissions 仍會將該權杖視為有效。

政策評估

將您的使用者集區設定為政策存放區的身分來源。設定讓您的應用程式在請求中提交使用者的權杖給 Verified Permissions。Verified Permissions 會針對每個請求，將權杖中的宣告與政策進行比較。Verified Permissions 政策就像 AWS 中的 IAM 政策。它會宣告主體、資源和動作。`Allow`如果「已驗證權限」符合允許的動作且不符合明確`Deny`動作，則會以「驗證權限」回應您的要求；否則，它會以回應`Deny`。如需詳細資訊，請參閱《Amazon Verified Permissions User Guide》中的 Amazon Verified Permissions policies。

自訂權杖

若要變更、新增和移除您要呈現給已驗證權限的使用者宣告，請使用產生權杖前 Lambda 觸發程序. 使用權杖產生前觸發程序，您就可以新增和修改權杖中的宣告。例如，您可以查詢資料庫中的其他使用者屬性，並將這些屬性編碼到您的 ID 權杖中。

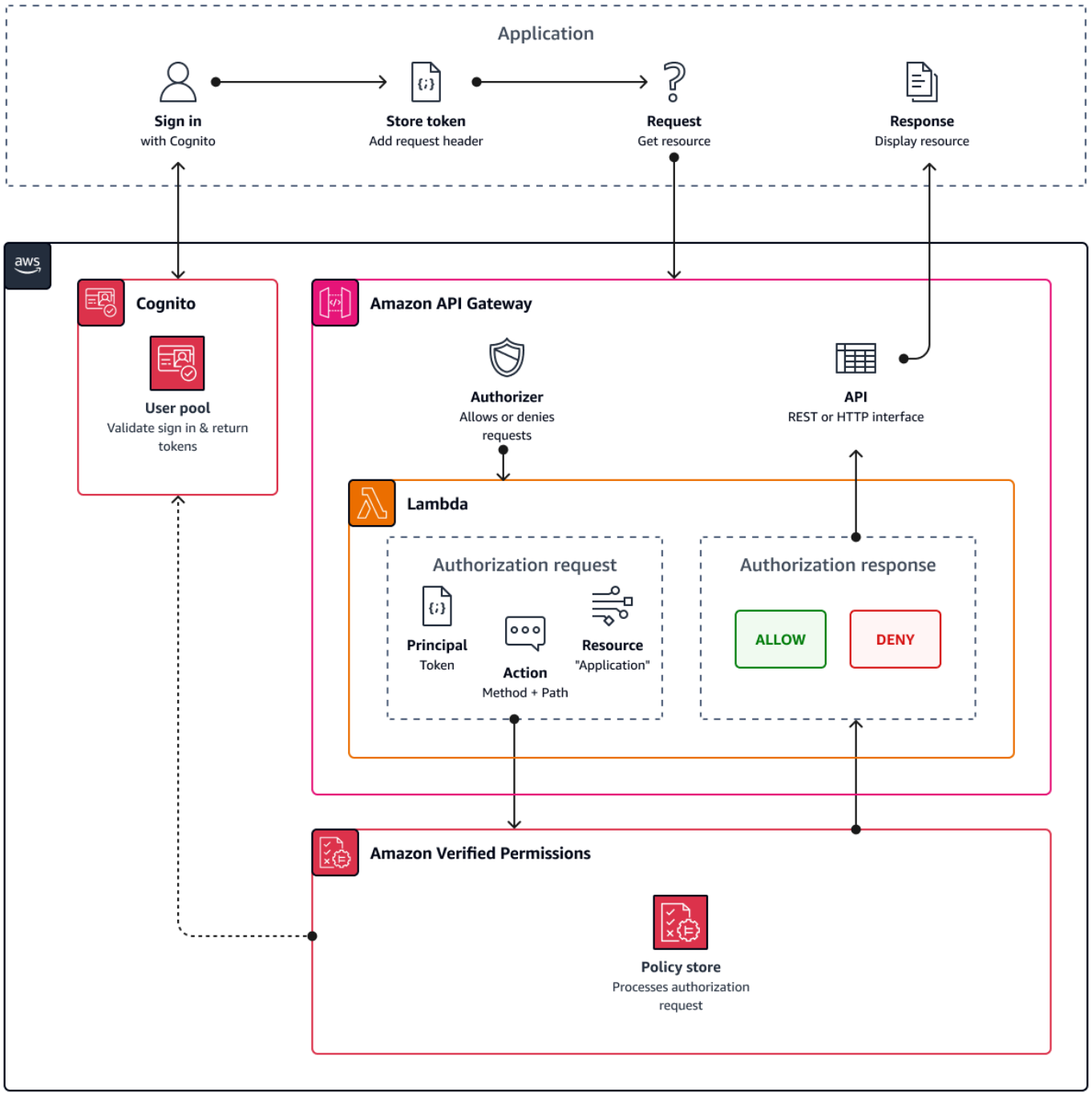
> **Note**
>
> 由於 Verified Permissions 處理宣告的方式，請勿在您的權杖產生前函數中新增名為 `cognito`、`dev` 或 `custom` 的宣告。若您不是採用冒號分隔格式 (如 `cognito:username`) 顯示這些保留的宣告字首，而是採用完整宣告名稱，那麼您的授權請求便會失敗。

如需有關 Verified Permissions 如何將 Amazon Cognito 權杖中的宣告對應至授權政策的詳細資訊，請參閱將 Amazon Cognito 權杖對應至 Verified Permissions 結構描述。

具有已驗證權限的 API 授權

您的 ID 或存取權杖可以授權對具有已驗證許可的後端 Amazon API Gateway REST API 的請求。您可以建立原則存放區，其中包含使用者集區和 API 的立即連結。使用「使用 Cognito 和 API Gateway 設定」啟動選項，「已驗證的權限」會將使用者集區身分識別來源新增至政策存放區，並將 Lambda 授權者新增至 API。當您的應用程式將使用者集區承載權杖傳遞至 API 時，Lambda 授權者會叫用已驗證的權限。授權者將 Token 作為主體傳遞，並將請求路徑和方法作為動作傳遞。

下圖說明具有已驗證權限的 API Gateway API 的授權流程。如需詳細細資料，請參閱 Amazon 驗證許可使用者指南中的 API 連結政策存放區。



已驗證的權限結構圍繞用戶集區組的 API 授權。由於 ID 和存取權杖都包含`cognito:groups`宣告，因此您的原則存放區可以在各種應用程式內容中針對 API 管理以角色為基礎的存取控制 (RBAC)。

選擇策略存放區設定

在原則存放區上設定身分識別來源時，您必須選擇是要處理存取權或 ID Token。這項決定對您的政策引擎的運作方式非常重要。ID 令牌包含用戶屬性。訪問令牌包含用戶訪問控制信息：O Auth 範圍。雖然這兩種權杖類型都有群組成員資格資訊，但我們通常建議使用具有已驗證權限原則存放區的 RBAC 存取權杖。訪問令牌添加到具有範圍的組成員資格中，這些範圍可以促進授權決策。訪問令牌中的聲明成為授權請求中的上下文。

當您將使用者集區設定為身分識別來源時，也必須設定使用者和群組實體類型。實體類型是主參與者、動作和資源識別元，您可以在已驗證的權限原則中參照。原則存放區中的實體可以具有成員資格關係，其中一個實體可以是父實體的成員。有了成員資格，您就可以參照主參與者群組、動作群組和資源群組。在使用者集區群組的情況下，您指定的使用者實體類型必須是群組實體類型的成員。當您在 [已驗證的權限] 主控台中設定 API 連結的原則存放區或遵循 [引導式設定] 時，您的原則存放區會自動具有此父成員關係。

ID 令牌可以將 RBAC 與基於屬性的訪問控制（ABAC）結合使用。建立 API 連結的原則存放區之後，您可以使用使用者屬性和群組成員資格來增強您的策略。ID 令牌中的屬性聲明成為授權請求中的主要屬性。您的原則可以根據主參與者屬性做出授權決策。

您也可以設定政策存放區，以接受符合您提供之可接受應用程式用戶端清單的`aud`或`client_id`宣告的權杖。

角色型 API 授權的範例政策

下列範例原則是透過針對範PetStore例 REST API 設定已驗證權限原則存放區所建立。

```
permit(
  principal in PetStore::UserGroup::"us-east-1_EXAMPLE|MyGroup",
  action in [ PetStore::Action::"get /pets", PetStore::Action::"get /pets/{petId}" ],
  resource
  );
```

在以下情況下，驗證權限會從您的應用程式傳回授權要求的`Allow`決定：

1. 您的應用程序在`Authorization`標題中傳遞 ID 或訪問令牌作為承載令牌。
2. 您的應用程序傳遞了一個帶有包含該字符串的`cognito:groups`聲明的令牌`MyGroup`。
3. 您的應用程式`HTTP GET`向 (例如，`https://myapi.example.com/pets`或) 提出要求`https://myapi.example.com/pets/scrappy`。

Amazon Cognito 使用者的政策範例

您的使用者集區也可以在 API 要求以外的情況下，產生對已驗證權限的授權要求。您可以將應用程式中的任何存取控制決策提交至原則存放區。例如，您可以在任何請求傳輸網路之前，使用以屬性為基礎的存取控制來補充 Amazon DynamoDB 或 Amazon S3 安全性，從而減少配額使用。

以下範例使用 Cedar 政策語言，允許透過某一個使用者集區應用程式用戶端進行驗證的財務使用者讀取和寫入 `example_image.png`。John 是您的應用程式中的使用者，他從您的應用程式用戶端收到 ID 權杖，並在 GET 請求中將該權杖傳遞至需要授權的 URL `https://example.com/images/example_image.png`。John 的 ID 權杖內有您的使用者集區應用程式用戶端 ID `1234567890example` 的 `aud` 宣告。您的權杖產生前 Lambda 函數還針對 John 插入了一個值為 `Finance1234` 的新宣告 `costCenter`。

```
permit (
   principal,
   actions in [ExampleCorp::Action::"readFile", "writeFile"],
   resource == ExampleCorp::Photo::"example_image.png"
)
when {
   principal.aud == "1234567890example" &&
   principal.custom.costCenter like "Finance*"
};
```

以下請求內文會產生 `Allow` 回應。

```
{
   "accesstoken": "[John's ID token]",
   "action": {
      "actionId": "readFile",
      "actionType": "Action"
   },
   "resource": {
      "entityId": "example_image.png",
      "entityType": "Photo"
   }
}
```

若您想要在 Verified Permissions 政策中指定主體，請使用下列格式：

```
permit (
   principal == [Namespace]::[Entity]::"[user pool ID]"|"[user sub]",
```

```
   action,
   resource
);
```

以下是識別碼為 sub 或使用者 ID us-east-1_Example 之使用者集區中之使用者的範例主體973db890-092c-49e4-a9d0-912a4c0a20c7。

```
principal == ExampleCorp::User::"us-east-1_Example|973db890-092c-49e4-
a9d0-912a4c0a20c7",
```

當您要在 [已驗證權限] 原則中指定使用者群組時，請使用下列格式：

```
permit (
   principal in [Namespace]::[Group Entity]::"[Group name]",
   action,
   resource
);
```

下面是一個例子

屬性型存取控制

針對您的應用程式具有已驗證許可的授權，以及適用於 AWS 登入資料的 Amazon Cognito 身分集區的存取控制功能的屬性，都是以屬性為基礎的存取控制 (ABAC)。以下是 Verified Permissions 與 Amazon Cognito ABAC 的比較。在 ABAC 中，系統會檢查實體的屬性，並從您定義的條件做出授權決策。

服務	處理	結果
Amazon Verified Permissions	從使用者集區 JWT 的分析傳回Allow或Deny決策。	根據 Cedar 原則評估，存取應用程式資源會成功或失敗。
Amazon Cognito 可身份集區 (用於存取控制的屬性)	根據用戶的屬性將會話標籤分配給用戶。IAM 政策條件可以檢查標籤Allow或Deny使用者的存取權 AWS 服務。	具有 IAM 角色臨時 AWS 登入資料的標記工作階段。

適用於使用 AWS SDK 的 Amazon Cognito 程式碼範例

下列程式碼範例示範如何使用 Amazon Cognito 搭配 AWS 軟體開發套件 (SDK)。

如需完整的 AWS SDK 開發人員指南和程式碼範例清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含入門相關資訊和舊版 SDK 的詳細資訊。

程式碼範例

- 使 AWS 用 SDK 的 Amazon Cognito 身份的代碼示例
  - 使 AWS 用 SDK 對亞馬遜認知身份進行操作
    - 搭CreateIdentityPool配 AWS 開發套件或 CLI 使用
    - 搭DeleteIdentityPool配 AWS 開發套件或 CLI 使用
    - 搭DescribeIdentityPool配 AWS 開發套件或 CLI 使用
    - 搭GetCredentialsForIdentity配 AWS 開發套件或 CLI 使用
    - 搭GetIdentityPoolRoles配 AWS 開發套件或 CLI 使用
    - 搭ListIdentityPools配 AWS 開發套件或 CLI 使用
    - 搭SetIdentityPoolRoles配 AWS 開發套件或 CLI 使用
    - 搭UpdateIdentityPool配 AWS 開發套件或 CLI 使用
  - 使用軟體開發套件的 Amazon Cognito 身分識別跨服務範例 AWS
    - 建置 Amazon Transcribe 應用程式
    - 建立 Amazon Textract Explorer 應用程式
- 使 AWS 用軟體開發套件的 Amazon Cognito 身分供應商程式碼範例
  - 使 AWS 用 SDK 的 Amazon Cognito 身份提供商的操作
    - 搭AdminCreateUser配 AWS 開發套件或 CLI 使用
    - 搭AdminGetUser配 AWS 開發套件或 CLI 使用
    - 搭AdminInitiateAuth配 AWS 開發套件或 CLI 使用
    - 搭AdminRespondToAuthChallenge配 AWS 開發套件或 CLI 使用
    - 搭AdminSetUserPassword配 AWS 開發套件或 CLI 使用
    - 搭AssociateSoftwareToken配 AWS 開發套件或 CLI 使用
    - 搭ConfirmDevice配 AWS 開發套件或 CLI 使用
    - 搭ConfirmForgotPassword配 AWS 開發套件或 CLI 使用
    - 搭ConfirmSignUp配 AWS 開發套件或 CLI 使用

使 AWS 用 SDK 的 Amazon Cognito 身份的代碼示例

下列程式碼範例說明如何搭配 AWS 軟體開發套件 (SDK) 使用 Amazon Cognito 身分識別。

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。

Cross-service examples (跨服務範例) 是跨多個 AWS 服務執行的應用程式範例。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。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

- 使 AWS 用 SDK 對亞馬遜認知身份進行操作
  - 搭CreateIdentityPool配 AWS 開發套件或 CLI 使用
  - 搭DeleteIdentityPool配 AWS 開發套件或 CLI 使用
  - 搭DescribeIdentityPool配 AWS 開發套件或 CLI 使用
  - 搭GetCredentialsForIdentity配 AWS 開發套件或 CLI 使用
  - 搭GetIdentityPoolRoles配 AWS 開發套件或 CLI 使用
  - 搭ListIdentityPools配 AWS 開發套件或 CLI 使用
  - 搭SetIdentityPoolRoles配 AWS 開發套件或 CLI 使用
  - 搭UpdateIdentityPool配 AWS 開發套件或 CLI 使用
- 使用軟體開發套件的 Amazon Cognito 身分識別跨服務範例 AWS
  - 建置 Amazon Transcribe 應用程式
  - 建立 Amazon Textract Explorer 應用程式

使 AWS 用 SDK 對亞馬遜認知身份進行操作

下列程式碼範例示範如何使用 AWS SDK 執行個別 Amazon Cognito 身分識別動作。這些摘錄會呼叫 Amazon Cognito Identity 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 Amazon Cognito 身分 API 參考。

範例

- 搭CreateIdentityPool配 AWS 開發套件或 CLI 使用
- 搭DeleteIdentityPool配 AWS 開發套件或 CLI 使用
- 搭DescribeIdentityPool配 AWS 開發套件或 CLI 使用
- 搭GetCredentialsForIdentity配 AWS 開發套件或 CLI 使用
- 搭GetIdentityPoolRoles配 AWS 開發套件或 CLI 使用
- 搭ListIdentityPools配 AWS 開發套件或 CLI 使用
- 搭SetIdentityPoolRoles配 AWS 開發套件或 CLI 使用
- 搭UpdateIdentityPool配 AWS 開發套件或 CLI 使用

搭**CreateIdentityPool**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`CreateIdentityPool`。

CLI

AWS CLI

若要使用 Cognito 身分集區提供者建立身分集區

此範例會建立名為的身分集區 MyIdentityPool。它具有 Cognito 身分集區提供者。不允許使用未驗證的身分。

命令：

```
aws cognito-identity create-identity-pool --identity-pool-name
 MyIdentityPool --no-allow-unauthenticated-identities --cognito-
identity-providers ProviderName="cognito-idp.us-west-2.amazonaws.com/us-
west-2_aaaaaaaaa",ClientId="3n4b5urk1ft4fl3mg5e62d9ado",ServerSideTokenCheck=false
```

輸出：

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[CreateIdentityPool](#)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.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * 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 CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <identityPoolName>\s

                Where:
                    identityPoolName - The name to give your identity pool.
                """;

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

```
        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    public static String createIdPool(CognitoIdentityClient cognitoClient, String
 identityPoolName) {
        try {
            CreateIdentityPoolRequest poolRequest =
 CreateIdentityPoolRequest.builder()
                    .allowUnauthenticatedIdentities(false)
                    .identityPoolName(identityPoolName)
                    .build();

            CreateIdentityPoolResponse response =
 cognitoClient.createIdentityPool(poolRequest);
            return response.identityPoolId();

        } catch (CognitoIdentityProviderException 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CreateIdentityPool中的。

PowerShell

適用的工具 PowerShell

範例 1：建立允許未驗證身分的新身分識別集區。

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

輸出：

```
LoggedAt                        : 8/12/2015 4:56:07 PM
AllowUnauthenticatedIdentities  : True
DeveloperProviderName           :
IdentityPoolId                  : us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3
IdentityPoolName                : CommonTests13
OpenIdConnectProviderARNs       : {}
SupportedLoginProviders         : {}
ResponseMetadata                : Amazon.Runtime.ResponseMetadata
ContentLength                   : 136
HttpStatusCode                  : OK
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程CreateIdentityPool式參考中的。

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。

建立新的身分集區。

```
    /// Create a new identity pool and return its ID.
    ///
    /// - Parameters:
```

```
    ///     - name: The name to give the new identity pool.
    ///
    /// - Returns: A string containing the newly created pool's ID, or `nil`
    ///   if an error occurred.
    ///
    func createIdentityPool(name: String) async throws -> String? {
        let cognitoInputCall = CreateIdentityPoolInput(developerProviderName:
"com.exampleco.CognitoIdentityDemo",
                                                       identityPoolName: name)

        let result = try await cognitoIdentityClient.createIdentityPool(input:
cognitoInputCall)
        guard let poolId = result.identityPoolId else {
            return nil
        }

        return poolId
    }
```

- 如需詳細資訊，請參閱適用於 Swift 的AWS SDK 開發人員指南。
- 有關 API 的詳細信息，請參閱 AWS SDK CreateIdentityPool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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**DeleteIdentityPool**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteIdentityPool。

CLI

AWS CLI

若要刪除身分集區

以下範例 delete-identity-pool 會刪除指定的身分集區。

命令：

```
aws cognito-identity delete-identity-pool \
    --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111"
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteIdentityPool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.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;

/**
 * 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 DeleteIdentityPool {

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

                Usage:
                    <identityPoolId>\s

                Where:
                    identityPoolId - The Id value of your identity pool.
                """;
```

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

        String identityPoold = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        deleteIdPool(cognitoIdClient, identityPoold);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
 identityPoold) {
        try {

            DeleteIdentityPoolRequest identityPoolRequest =
 DeleteIdentityPoolRequest.builder()
                    .identityPoolId(identityPoold)
                    .build();

            cognitoIdClient.deleteIdentityPool(identityPoolRequest);
            System.out.println("Done");

        } catch (AwsServiceException 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DeleteIdentityPool中的。

PowerShell

適用的工具 PowerShell

範例 1：刪除特定身分集區。

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteIdentityPool式參考中的。

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。

刪除指定的身分集區。

```
    /// Delete the specified identity pool.
    ///
    /// - Parameters:
    ///   - id: The ID of the identity pool to delete.
    ///
    func deleteIdentityPool(id: String) async throws {
        let input = DeleteIdentityPoolInput(
            identityPoolId: id
        )

        _ = try await cognitoIdentityClient.deleteIdentityPool(input: input)
    }
```

- 如需詳細資訊，請參閱適用於 Swift 的AWS SDK 開發人員指南。
- 有關 API 的詳細信息，請參閱 AWS SDK DeleteIdentityPool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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DescribeIdentityPool配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DescribeIdentityPool。

CLI

AWS CLI

描述身分集區

此範例說明身分集區。

命令：

```
aws cognito-identity describe-identity-pool --identity-pool-id "us-
west-2:11111111-1111-1111-1111-111111111111"
```

輸出：

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DescribeIdentityPool中的。

PowerShell

適用的工具 PowerShell

範例 1：依其 id 擷取特定識別集區的相關資訊。

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

輸出：

```
LoggedAt                        : 8/12/2015 4:29:40 PM
AllowUnauthenticatedIdentities  : True
DeveloperProviderName           :
IdentityPoolId                  : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName                : CommonTests1
OpenIdConnectProviderARNs       : {}
SupportedLoginProviders         : {}
ResponseMetadata                : Amazon.Runtime.ResponseMetadata
ContentLength                   : 142
HttpStatusCode                  : OK
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DescribeIdentityPool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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetCredentialsForIdentity配 AWS 開發套件或 CLI 使用

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

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.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
```

```
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * 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 GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <identityId>\s

            Where:
                identityId - The Id of an existing identity in the format
 REGION:GUID.
            """;

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

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
    }

    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
 String identityId) {
        try {
```

```
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
 GetCredentialsForIdentityRequest
                    .builder()
                    .identityId(identityId)
                    .build();

            GetCredentialsForIdentityResponse response = cognitoClient
                    .getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println(
                    "Identity ID " + response.identityId() + ", Access key ID " +
 response.credentials().accessKeyId());

        } catch (CognitoIdentityProviderException 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GetCredentialsForIdentity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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetIdentityPoolRoles配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetIdentityPoolRoles。

CLI

AWS CLI

若要取得識別集區角色

此範例取得識別集區角色。

命令：

```
aws cognito-identity get-identity-pool-roles --identity-pool-id "us-
west-2:11111111-1111-1111-1111-111111111111"
```

輸出：

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "Roles": {
      "authenticated": "arn:aws:iam::111111111111:role/
Cognito_MyIdentityPoolAuth_Role",
      "unauthenticated": "arn:aws:iam::111111111111:role/
Cognito_MyIdentityPoolUnauth_Role"
  }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetIdentityPoolRoles中的。

PowerShell

適用的工具 PowerShell

範例 1：取得特定識別集區之角色的相關資訊。

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

輸出：

```
LoggedAt         : 8/12/2015 4:33:51 PM
IdentityPoolId   : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
Roles            : {[unauthenticated, arn:aws:iam::123456789012:role/
CommonTests1Role]}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 165
HttpStatusCode   : OK
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetIdentityPoolRoles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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListIdentityPools**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListIdentityPools。

CLI

AWS CLI

若要列出身分集區

此範例會列出身分集區。最多列出 20 個身分識別。

命令：

```
aws cognito-identity list-identity-pools --max-results 20
```

輸出：

```
{
  "IdentityPools": [
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "MyIdentityPool"
      },
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "AnotherIdentityPool"
      },
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "IdentityPoolRegionA"
      }
  ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListIdentityPools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.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * 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 ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
 ListIdentityPoolsRequest.builder()
```

```
                    .maxResults(15)
                    .build();

            ListIdentityPoolsResponse response =
 cognitoClient.listIdentityPools(poolsRequest);
            response.identityPools().forEach(pool -> {
                System.out.println("Pool ID: " + pool.identityPoolId());
                System.out.println("Pool name: " + pool.identityPoolName());
            });

        } catch (CognitoIdentityProviderException 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ListIdentityPools中的。

PowerShell

適用的工具 PowerShell

範例 1：擷取現有識別集區的清單。

```
Get-CGIIdentityPoolList
```

輸出：

```
IdentityPoolId
 IdentityPoolName
--------------
 ----------------
us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1                    CommonTests1
us-east-1:118d242d-204e-4b88-b803-EXAMPLEGUID2                    Tests2
us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3                    CommonTests13
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListIdentityPools式參考中的。

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。

尋找特定其名稱的身分集區 ID。

```
    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned.
    ///
    /// - Returns: A string containing the ID of the specified identity pool
    ///   or `nil` on error or if not found.
    ///
    func getIdentityPoolID(name: String) async throws -> String? {
        var token: String? = nil

        // Iterate over the identity pools until a match is found.

        repeat {
            /// `token` is a value returned by `ListIdentityPools()` if the
            /// returned list of identity pools is only a partial list. You
            /// use the `token` to tell Amazon Cognito that you want to
            /// continue where you left off previously. If you specify `nil`
            /// or you don't provide the token, Amazon Cognito will start at
            /// the beginning.

            let listPoolsInput = ListIdentityPoolsInput(maxResults: 25,
nextToken: token)

            /// Read pages of identity pools from Cognito until one is found
```

```
            /// whose name matches the one specified in the `name` parameter.
            /// Return the matching pool's ID. Each time we ask for the next
            /// page of identity pools, we pass in the token given by the
            /// previous page.

            let output = try await cognitoIdentityClient.listIdentityPools(input:
listPoolsInput)

            if let identityPools = output.identityPools {
                for pool in identityPools {
                    if pool.identityPoolName == name {
                        return pool.identityPoolId!
                    }
                }
            }

            token = output.nextToken
        } while token != nil

        return nil
    }
```

取得現有身分集區的 ID 或建立它 (如果尚不存在)。

```
    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned
    ///
    /// - Returns: A string containing the ID of the specified identity pool.
    ///   Returns `nil` if there's an error or if the pool isn't found.
    ///
    public func getOrCreateIdentityPoolID(name: String) async throws -> String? {
        // See if the pool already exists. If it doesn't, create it.

        guard let poolId = try await self.getIdentityPoolID(name: name) else {
            return try await self.createIdentityPool(name: name)
        }

        return poolId
    }
```

- 如需詳細資訊，請參閱適用於 Swift 的AWS SDK 開發人員指南。
- 有關 API 的詳細信息，請參閱 AWS SDK ListIdentityPools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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`SetIdentityPoolRoles`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`SetIdentityPoolRoles`。

CLI

AWS CLI

若要設定識別集區角色

下列`set-identity-pool-roles`範例會設定識別集區角色。

```
aws cognito-identity set-identity-pool-roles \
    --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111" \
    --roles authenticated="arn:aws:iam::111111111111:role/
Cognito_MyIdentityPoolAuth_Role"
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考SetIdentityPoolRoles中的。

PowerShell

適用的工具 PowerShell

範例 1：將特定身分集區設定為具有未驗證的 IAM 角色。

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程SetIdentityPoolRoles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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭UpdateIdentityPool配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用UpdateIdentityPool。

CLI

AWS CLI

更新身分識別集區

此範例會更新身分集區。它將名稱設置為 MyIdentityPool。它將 Cognito 添加為身份提供者。它不允許未經身份驗證的身份。

命令：

```
aws cognito-identity update-identity-pool --identity-pool-id "us-
west-2:11111111-1111-1111-1111-111111111111" --identity-pool-name
 "MyIdentityPool" --no-allow-unauthenticated-identities --cognito-
identity-providers ProviderName="cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",ClientId="3n4b5urk1ft4fl3mg5e62d9ado",ServerSideTokenCheck=false
```

輸出：

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考UpdateIdentityPool中的。

PowerShell

適用的工具 PowerShell

範例 1：更新部分識別集區內容，在此情況下為識別集區的名稱。

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

輸出：

```
LoggedAt                        : 8/12/2015 4:53:33 PM
AllowUnauthenticatedIdentities  : False
DeveloperProviderName           :
IdentityPoolId                  : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName                : NewPoolName
OpenIdConnectProviderARNs       : {}
SupportedLoginProviders         : {}
ResponseMetadata                : Amazon.Runtime.ResponseMetadata
ContentLength                   : 135
HttpStatusCode                  : OK
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程UpdateIdentityPool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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用軟體開發套件的 Amazon Cognito 身分識別跨服務範例 AWS

下列範例應用程式使用 AWS SDK 將 Amazon Cognito 身分與其他應用程式結合在一起。 AWS 服務每個範例都包含一個連結 GitHub，您可以在其中找到如何設定和執行應用程式的指示。

範例

- 建置 Amazon Transcribe 應用程式
- 建立 Amazon Textract Explorer 應用程式

建置 Amazon Transcribe 應用程式

下列程式碼範例示範如何使用 Amazon Transcribe 在瀏覽器中轉錄和顯示語音錄音。

JavaScript

適用於 JavaScript (v3) 的開發套件

建立使用 Amazon Transcribe 的應用程式，在瀏覽器中轉錄和顯示語音錄音。應用程式使用兩個 Amazon Simple Storage Service (Amazon S3) 儲存貯體，一個負責支援應用程式程式碼，另一個負責存放文字記錄。應用程式會使用 Amazon Cognito 使用者集區來對您的使用者進行身分驗證。經過驗證的使用者具有 AWS Identity and Access Management (IAM) 許可以存取所需 AWS 服務。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例也可在 AWS SDK for JavaScript v3 開發人員指南中取得。

此範例中使用的服務

- Amazon Cognito Identity
- Amazon S3
- Amazon Transcribe

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

建立 Amazon Textract Explorer 應用程式

下列程式碼範例示範如何透過互動式應用程式探索 Amazon Textract 輸出。

JavaScript

適用於 JavaScript (v3) 的開發套件

示範如何使用建置 React 應用程式，該應用程式使用 Amazon Textract 擷取文件影像中的資料，並將其顯示在互動式網頁中。 AWS SDK for JavaScript 此範例會在 Web 瀏覽器中執行，且登入資料需要經過驗證的 Amazon Cognito 身分。它使用 Amazon Simple Storage Service (Amazon S3 進行儲存，對於通知，它會輪詢訂閱 Amazon Simple Notification Service (Amazon SNS)) 主題的 Amazon Simple Queue Service (Amazon SQS) 佇列。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Cognito Identity

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使 AWS 用軟體開發套件的 Amazon Cognito 身分供應商程式碼範例

下列程式碼範例顯示如何搭配 AWS 軟體開發套件 (SDK) 使用 Amazon Cognito 身分供應商。

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 SDK 使用此服務。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 Amazon Cognito

下列程式碼範例顯示如何開始使用 Amazon Cognito。

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 cognito-idp)

# Set this project's name.
project("hello_cognito")

# 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_cognito.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_cognito.cpp 來源檔案的程式碼。

```
#include <aws/core/Aws.h>
#include <aws/cognito-idp/CognitoIdentityProviderClient.h>
#include <aws/cognito-idp/model/ListUserPoolsRequest.h>
#include <iostream>

/*
 *  A "Hello Cognito" starter application which initializes an Amazon Cognito
 client and lists the Amazon Cognito
 *  user pools.
 *
 *  main function
 *
 *  Usage: 'hello_cognito'
 *
 */

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

        Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
 cognitoClient(clientConfig);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::String> userPools;

        do {
            Aws::CognitoIdentityProvider::Model::ListUserPoolsRequest
 listUserPoolsRequest;
            if (!nextToken.empty()) {
                listUserPoolsRequest.SetNextToken(nextToken);
            }

            Aws::CognitoIdentityProvider::Model::ListUserPoolsOutcome
 listUserPoolsOutcome =
```

```
                    cognitoClient.ListUserPools(listUserPoolsRequest);

            if (listUserPoolsOutcome.IsSuccess()) {
                for (auto &userPool:
 listUserPoolsOutcome.GetResult().GetUserPools()) {

                    userPools.push_back(userPool.GetName());
                }

                nextToken = listUserPoolsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "ListUserPools error: " <<
 listUserPoolsOutcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }


        } while (!nextToken.empty());
        std::cout << userPools.size() << " user pools found." << std::endl;
        for (auto &userPool: userPools) {
            std::cout << "   user pool: " << userPool << std::endl;
        }
    }

    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ListUserPools中的。

Go

SDK for Go V2

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

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
```

```
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考ListUserPools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.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * 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 ListUserPools {
    public static void main(String[] args) {
```

```
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient
cognitoClient) {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response =
cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID "
+ userpool.id());
            });

        } catch (CognitoIdentityProviderException 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ListUserPools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 {
  paginateListUserPools,
  CognitoIdentityProviderClient,
} from "@aws-sdk/client-cognito-identity-provider";

const client = new CognitoIdentityProviderClient({});

export const helloCognito = async () => {
  const paginator = paginateListUserPools({ client }, {});

  const userPoolNames = [];

  for await (const page of paginator) {
    const names = page.UserPools.map((pool) => pool.Name);
    userPoolNames.push(...names);
  }

  console.log("User pool names: ");
  console.log(userPoolNames.join("\n"));
  return userPoolNames;
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ListUserPools中的。

程式碼範例

- 使 AWS 用 SDK 的 Amazon Cognito 身份提供商的操作
  - 搭AdminCreateUser配 AWS 開發套件或 CLI 使用
  - 搭AdminGetUser配 AWS 開發套件或 CLI 使用
  - 搭AdminInitiateAuth配 AWS 開發套件或 CLI 使用
  - 搭AdminRespondToAuthChallenge配 AWS 開發套件或 CLI 使用
  - 搭AdminSetUserPassword配 AWS 開發套件或 CLI 使用
  - 搭AssociateSoftwareToken配 AWS 開發套件或 CLI 使用
  - 搭ConfirmDevice配 AWS 開發套件或 CLI 使用
  - 搭ConfirmForgotPassword配 AWS 開發套件或 CLI 使用
  - 搭ConfirmSignUp配 AWS 開發套件或 CLI 使用
  - 搭CreateUserPool配 AWS 開發套件或 CLI 使用

- 搭CreateUserPoolClient配 AWS 開發套件或 CLI 使用
- 搭DeleteUser配 AWS 開發套件或 CLI 使用
- 搭ForgotPassword配 AWS 開發套件或 CLI 使用
- 搭InitiateAuth配 AWS 開發套件或 CLI 使用
- 搭ListUserPools配 AWS 開發套件或 CLI 使用
- 搭ListUsers配 AWS 開發套件或 CLI 使用
- 搭ResendConfirmationCode配 AWS 開發套件或 CLI 使用
- 搭RespondToAuthChallenge配 AWS 開發套件或 CLI 使用
- 搭SignUp配 AWS 開發套件或 CLI 使用
- 搭UpdateUserPool配 AWS 開發套件或 CLI 使用
- 搭VerifySoftwareToken配 AWS 開發套件或 CLI 使用

- 使 AWS 用開發套件的 Amazon Cognito 身分供應商的案例
  - 使用開發套件，透過 Lambda 函數自動確認已知的 Amazon Cognito 認知使用者 AWS
  - 使用開發套件，透過 Lambda 函數自動遷移已知的 Amazon Cognito 知使用者 AWS
  - 使用需要 MFA 使用者使用開發套件的 Amazon Cognito 使用者集區註冊使用者 AWS
  - 使用開發套件 AWS 進行 Amazon Cognito 使用者身份驗證之後，使用 Lambda 函數寫入自訂活動

使 AWS 用 SDK 的 Amazon Cognito 身份提供商的操作

下列程式碼範例示範如何使用 AWS SDK 執行個別 Amazon Cognito 身分識別提供者動作。這些摘錄會呼叫 Amazon Cognito 身分提供者 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 Amazon Cognito 身分提供者 API 參考。

範例

- 搭AdminCreateUser配 AWS 開發套件或 CLI 使用
- 搭AdminGetUser配 AWS 開發套件或 CLI 使用
- 搭AdminInitiateAuth配 AWS 開發套件或 CLI 使用
- 搭AdminRespondToAuthChallenge配 AWS 開發套件或 CLI 使用
- 搭AdminSetUserPassword配 AWS 開發套件或 CLI 使用
- 搭AssociateSoftwareToken配 AWS 開發套件或 CLI 使用

- 搭ConfirmDevice配 AWS 開發套件或 CLI 使用
- 搭ConfirmForgotPassword配 AWS 開發套件或 CLI 使用
- 搭ConfirmSignUp配 AWS 開發套件或 CLI 使用
- 搭CreateUserPool配 AWS 開發套件或 CLI 使用
- 搭CreateUserPoolClient配 AWS 開發套件或 CLI 使用
- 搭DeleteUser配 AWS 開發套件或 CLI 使用
- 搭ForgotPassword配 AWS 開發套件或 CLI 使用
- 搭InitiateAuth配 AWS 開發套件或 CLI 使用
- 搭ListUserPools配 AWS 開發套件或 CLI 使用
- 搭ListUsers配 AWS 開發套件或 CLI 使用
- 搭ResendConfirmationCode配 AWS 開發套件或 CLI 使用
- 搭RespondToAuthChallenge配 AWS 開發套件或 CLI 使用
- 搭SignUp配 AWS 開發套件或 CLI 使用
- 搭UpdateUserPool配 AWS 開發套件或 CLI 使用
- 搭VerifySoftwareToken配 AWS 開發套件或 CLI 使用

搭**AdminCreateUser**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`AdminCreateUser`。

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：

- 在 Amazon Cognito 使用者身份驗證後，使用 Lambda 函數寫入自訂活動資料

CLI

AWS CLI

若要建立使用者

下列`admin-create-user`範例會建立具有指定設定的電子郵件地址和電話號碼的使用者。

```
aws cognito-idp admin-create-user \
    --user-pool-id us-west-2_aaaaaaaaa \
```

```
    --username diego \
    --user-attributes Name=email,Value=diego@example.com
 Name=phone_number,Value="+15555551212" \
    --message-action SUPPRESS
```

輸出：

```
{
    "User": {
        "Username": "diego",
        "Attributes": [
            {
                "Name": "sub",
                "Value": "7325c1de-b05b-4f84-b321-9adc6e61f4a2"
            },
            {
                "Name": "phone_number",
                "Value": "+15555551212"
            },
            {
                "Name": "email",
                "Value": "diego@example.com"
            }
        ],
        "UserCreateDate": 1548099495.428,
        "UserLastModifiedDate": 1548099495.428,
        "Enabled": true,
        "UserStatus": "FORCE_CHANGE_PASSWORD"
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考AdminCreateUser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。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create 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AdminCreateUser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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**AdminGetUser**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用AdminGetUser。

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：

- 使用需要 MFA 的使用者集區註冊使用者

.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>
    /// Get the specified user from an Amazon Cognito user pool with
administrator access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
    /// <returns>Async task.</returns>
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考AdminGetUser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::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                          outcome.GetResult().GetUserStatus()) << std::endl;
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() <<
std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考AdminGetUser中的。

CLI

AWS CLI

若要取得使用者

此範例取得使用者名稱 jane@example.com 的相關資訊。

命令：

```
aws cognito-idp admin-get-user --user-pool-id us-west-2_aaaaaaaaa --username
 jane@example.com
```

輸出：

```
{
  "Username": "4320de44-2322-4620-999b-5e2e1c8df013",
  "Enabled": true,
  "UserStatus": "FORCE_CHANGE_PASSWORD",
  "UserCreateDate": 1548108509.537,
  "UserAttributes": [
      {
          "Name": "sub",
          "Value": "4320de44-2322-4620-999b-5e2e1c8df013"
      },
      {
          "Name": "email_verified",
          "Value": "true"
      },
      {
          "Name": "phone_number_verified",
          "Value": "true"
      },
      {
          "Name": "phone_number",
          "Value": "+01115551212"
      },
      {
          "Name": "email",
          "Value": "jane@example.com"
      }
  ],
  "UserLastModifiedDate": 1548108509.537
```

```
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考AdminGetUser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。

```
   public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
            String poolId) {
       try {
           AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                   .username(userName)
                   .userPoolId(poolId)
                   .build();

           AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
           System.out.println("User status " + response.userStatusAsString());

       } catch (CognitoIdentityProviderException 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AdminGetUser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。

```
const adminGetUser = ({ userPoolId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminGetUserCommand({
    UserPoolId: userPoolId,
    Username: username,
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考AdminGetUser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 getAdminUser(userNameVal: String?, poolIdVal: String?) {
    val userRequest = AdminGetUserRequest {
        username = userNameVal
        userPoolId = poolIdVal
    }
```

```
    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

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

Python

適用於 Python (Boto3) 的 SDK

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

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_up_user(self, user_name, password, user_email):
        """
```

```
        Signs up a new user with Amazon Cognito. This action prompts Amazon
Cognito
        to send an email to the specified email address. The email contains a
code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
```

```
                raise
        return confirmed
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件AdminGetUser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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭AdminInitiateAuth配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用AdminInitiateAuth。

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：

- 使用需要 MFA 的使用者集區註冊使用者

.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>
    /// Initiate an admin auth request.
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
userPoolId, string userName, string password)
    {
```

```
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考AdminInitiateAuth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::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(
```

```
Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


   Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
           client.AdminInitiateAuth(request);

   if (outcome.IsSuccess()) {
       std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
       sessionResult = outcome.GetResult().GetSession();
   }
   else {
       std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                 << outcome.GetError().GetMessage()
                 << std::endl;
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考AdminInitiateAuth中的。

CLI

AWS CLI

若要啟動授權

此範例使用 ADMIN_NO_SRP_AUTH 流程對使用者名稱 jane@example.com 啟動授權

用戶端必須啟用以伺服器為基礎的驗證 (ADMIN_NO_SRP_AUTH) 的登入 API。

使用返回值中的會話信息來調用 admin-respond-to-auth-challenge。

命令：

```
aws cognito-idp admin-initiate-auth --user-pool-id us-west-2_aaaaaaaaa --client-
id 3n4b5urk1ft4fl3mg5e62d9ado --auth-flow ADMIN_NO_SRP_AUTH --auth-parameters
 USERNAME=jane@example.com,PASSWORD=password
```

輸出：

```
{
  "ChallengeName": "NEW_PASSWORD_REQUIRED",
```

```
  "Session": "SESSION",
  "ChallengeParameters": {
      "USER_ID_FOR_SRP": "84514837-dcbc-4af1-abff-f3c109334894",
      "requiredAttributes": "[]",
      "userAttributes": "{\"email_verified\":\"true\",\"phone_number_verified\":
\"true\",\"phone_number\":\"+01xxx5550100\",\"email\":\"jane@example.com\"}"
  }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考AdminInitiateAuth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。

```
   public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
           String clientId, String userName, String password, String userPoolId)
{
       try {
           Map<String, String> authParameters = new HashMap<>();
           authParameters.put("USERNAME", userName);
           authParameters.put("PASSWORD", password);

           AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                   .clientId(clientId)
                   .userPoolId(userPoolId)
                   .authParameters(authParameters)
                   .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                   .build();

           AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
           System.out.println("Result Challenge is : " +
response.challengeName());
```

```
            return response;

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

        return null;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考AdminInitiateAuth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。

```
const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考AdminInitiateAuth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 checkAuthMethod(clientIdVal: String, userNameVal: String,
 passwordVal: String, userPoolIdVal: String): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest = AdminInitiateAuthRequest {
        clientId = clientIdVal
        userPoolId = userPoolIdVal
        authParameters = authParas
        authFlow = AuthFlowType.AdminUserPasswordAuth
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

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

Python

適用於 Python (Boto3) 的 SDK

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

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
 server.

        If the user pool is configured to require MFA and this is the first sign-
in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
```

```
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
this
                 returns an access token that can be used to get AWS credentials.
Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
not "
                        "configured for TOTP MFA. This example requires TOTP
MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
```

```
            return response
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件AdminInitiateAuth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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭AdminRespondToAuthChallenge配 AWS 開發套件或 CLI 使用

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

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：

- 使用需要 MFA 的使用者集區註冊使用者

.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>
/// Respond to an admin authentication challenge.
/// </summary>
/// <param name="userName">The name of the user.</param>
/// <param name="clientId">The client ID.</param>
/// <param name="mfaCode">The multi-factor authentication code.</param>
/// <param name="session">The current application session.</param>
/// <param name="clientId">The user pool ID.</param>
/// <returns>The result of the authentication response.</returns>
public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
    string userName,
```

```
        string clientId,
        string mfaCode,
        string session,
        string userPoolId)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>();
        challengeResponses.Add("USERNAME", userName);
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        var respondToAuthChallengeRequest = new
AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考AdminRespondToAuthChallenge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::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
        request.AddChallengeResponses("USERNAME", userName);
        request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
        request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
        request.SetClientId(clientID);
        request.SetUserPoolId(userPoolID);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
                client.AdminRespondToAuthChallenge(request);

        if (outcome.IsSuccess()) {
            std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                      << std::endl;

            accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考AdminRespondToAuthChallenge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。

```
   // Respond to an authentication challenge.
   public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
           String userName, String clientId, String mfaCode, String session) {
       System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
       Map<String, String> challengeResponses = new HashMap<>();

       challengeResponses.put("USERNAME", userName);
       challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

       AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
               .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
               .clientId(clientId)
               .challengeResponses(challengeResponses)
               .session(session)
               .build();

       AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
               .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
               + respondToAuthChallengeResult.authenticationResult());
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考AdminRespondToAuthChallenge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。

```
const adminRespondToAuthChallenge = ({
  userPoolId,
  clientId,
  username,
  totp,
  session,
}) => {
  const client = new CognitoIdentityProviderClient({});
  const command = new AdminRespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: totp,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考AdminRespondToAuthChallenge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。

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(userName: String, clientIdVal: String?,
 mfaCode: String, sessionVal: String?) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest = AdminRespondToAuthChallengeRequest {
        challengeName = ChallengeNameType.SoftwareTokenMfa
        clientId = clientIdVal
        challengeResponses = challengeResponsesOb
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

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

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。

透過提供關聯的 MFA 應用程式所產生的程式碼，以回應 MFA 挑戰。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step
 of
        a two-factor sign-in. When sign-in is successful, it returns an access
 token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
 initiate
                        authentication.
```

```
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You
might have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s:
%s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件AdminRespondToAuthChallenge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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭AdminSetUserPassword配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用AdminSetUserPassword。

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：

- 在 Amazon Cognito 使用者身份驗證後，使用 Lambda 函數寫入自訂活動資料

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。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
```

```
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password 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AdminSetUserPassword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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**AssociateSoftwareToken**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用AssociateSoftwareToken。

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：

- 使用需要 MFA 的使用者集區註冊使用者

.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>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
_cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;

        Console.WriteLine($"Use the following secret code to set up the
authenticator: {secretCode}");

        return tokenResponse.Session;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考AssociateSoftwareToken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::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
 client(clientConfig);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest
 request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome
 outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for
 example Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考AssociateSoftwareToken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。

```
   public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
       AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
               .session(session)
               .build();

       AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
               .associateSoftwareToken(softwareTokenRequest);
       String secretCode = tokenResponse.secretCode();
       System.out.println("Enter this token into Google Authenticator");
       System.out.println(secretCode);
       return tokenResponse.session();
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考AssociateSoftwareToken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。

```
const associateSoftwareToken = (session) => {
  const client = new CognitoIdentityProviderClient({});
```

```
  const command = new AssociateSoftwareTokenCommand({
    Session: session,
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考AssociateSoftwareToken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 getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest = AssociateSoftwareTokenRequest {
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}
```

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

Python

適用於 Python (Boto3) 的 SDK

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

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the
 user.

        :param session: Session information returned from a previous call to
 initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
 self.cognito_idp_client.associate_software_token(Session=session)
```

```
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件AssociateSoftwareToken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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ConfirmDevice配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ConfirmDevice。

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：

- 使用需要 MFA 的使用者集區註冊使用者

.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>
    /// Initiates and confirms tracking of the device.
```

```
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</
param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string
 deviceKey, string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ConfirmDevice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。

```
const confirmDevice = ({ deviceKey, accessToken, passwordVerifier, salt }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmDeviceCommand({
    DeviceKey: deviceKey,
    AccessToken: accessToken,
    DeviceSecretVerifierConfig: {
      PasswordVerifier: passwordVerifier,
```

```
      Salt: salt,
    },
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ConfirmDevice中的。

Python

適用於 Python (Boto3) 的 SDK

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

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def confirm_mfa_device(
        self,
        user_name,
```

```
        device_key,
        device_group_key,
        device_password,
        access_token,
        aws_srp,
    ):
        """
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a
new
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example
uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False.
When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt +
device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value,
srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
```

```
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ConfirmDevice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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ConfirmForgotPassword**配 AWS 開發套件或 CLI 使用

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

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 Lambda 函數自動遷移已知的使用者

CLI

AWS CLI

確認忘記密碼

此範例會確認忘記使用者名稱 diego@example.com 的密碼。

命令：

```
aws cognito-idp confirm-forgot-password --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username=diego@example.com --password PASSWORD --confirmation-code CONF_CODE
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ConfirmForgotPassword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。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(clientId string, code string,
 userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(context.TODO(),
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
```

```
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考ConfirmForgotPassword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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ConfirmSignUp配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ConfirmSignUp。

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：

- 使用需要 MFA 的使用者集區註冊使用者

.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>
    /// Confirm that the user has signed up.
    /// </summary>
```

```
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
string userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ConfirmSignUp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::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

       Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
       request.SetClientId(clientID);
       request.SetConfirmationCode(confirmationCode);
       request.SetUsername(userName);

       Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
               client.ConfirmSignUp(request);

       if (outcome.IsSuccess()) {
           std::cout << "ConfirmSignup was Successful."
                     << std::endl;
       }
       else {
           std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考ConfirmSignUp中的。

CLI

AWS CLI

若要確認註冊

此範例會確認註冊使用者名稱 diego@example.com。

命令：

```
aws cognito-idp confirm-sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username=diego@example.com --confirmation-code CONF_CODE
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ConfirmSignUp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。

```
   public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
           String userName) {
       try {
           ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                   .clientId(clientId)
                   .confirmationCode(code)
                   .username(userName)
                   .build();

           identityProviderClient.confirmSignUp(signUpRequest);
           System.out.println(userName + " was confirmed");

       } catch (CognitoIdentityProviderException 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ConfirmSignUp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。

```
const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ConfirmSignUp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 confirmSignUp(clientIdVal: String?, codeVal: String?, userNameVal:
 String?) {
    val signUpRequest = ConfirmSignUpRequest {
        clientId = clientIdVal
        confirmationCode = codeVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

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

Python

適用於 Python (Boto3) 的 SDK

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

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
 registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
```

```
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ConfirmSignUp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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**CreateUserPool**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`CreateUserPool`。

CLI

AWS CLI

建立最低限度設定的使用者集區

此範例會建立 MyUserPool 使用預設值命名的使用者集區。沒有必要的屬性，也沒有應用程式用戶端。MFA 和進階安全性已停用。

命令：

```
aws cognito-idp create-user-pool --pool-name MyUserPool
```

輸出：

```
{
  "UserPool": {
      "SchemaAttributes": [
          {
              "Name": "sub",
              "StringAttributeConstraints": {
                  "MinLength": "1",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": true,
              "AttributeDataType": "String",
              "Mutable": false
          },
          {
              "Name": "name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": false,
              "AttributeDataType": "String",
              "Mutable": true
          },
          {
              "Name": "given_name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": false,
              "AttributeDataType": "String",
              "Mutable": true
          },
          {
              "Name": "family_name",
              "StringAttributeConstraints": {
```

```
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "middle_name",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "nickname",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "preferred_username",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "profile",
            "StringAttributeConstraints": {
```

```
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "picture",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "website",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "email",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "AttributeDataType": "Boolean",
            "DeveloperOnlyAttribute": false,
```

```
                "Required": false,
                "Name": "email_verified",
                "Mutable": true
            },
            {
                "Name": "gender",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "birthdate",
                "StringAttributeConstraints": {
                    "MinLength": "10",
                    "MaxLength": "10"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "zoneinfo",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "locale",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
```

```
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "phone_number",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "AttributeDataType": "Boolean",
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "Name": "phone_number_verified",
            "Mutable": true
        },
        {
            "Name": "address",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "updated_at",
            "NumberAttributeConstraints": {
                "MinValue": "0"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "Number",
            "Mutable": true
        }
    ],
```

```
      "MfaConfiguration": "OFF",
      "Name": "MyUserPool",
      "LastModifiedDate": 1547833345.777,
      "AdminCreateUserConfig": {
          "UnusedAccountValidityDays": 7,
          "AllowAdminCreateUserOnly": false
      },
      "EmailConfiguration": {},
      "Policies": {
          "PasswordPolicy": {
              "RequireLowercase": true,
              "RequireSymbols": true,
              "RequireNumbers": true,
              "MinimumLength": 8,
              "RequireUppercase": true
          }
      },
      "CreationDate": 1547833345.777,
      "EstimatedNumberOfUsers": 0,
      "Id": "us-west-2_aaaaaaaaa",
      "LambdaConfig": {}
  }
}
```

用兩個必要屬性建立新的使用者集區

此範例會建立使用者集區 MyUserPool。集區設定為接受電子郵件作為使用者名稱屬性。它也會使用 Amazon Simple Email Service，將電子郵件來源地址設定為經過驗證的地址。

命令：

```
aws cognito-idp create-user-pool --pool-name MyUserPool --username-
attributes "email" --email-configuration=SourceArn="arn:aws:ses:us-
east-1:111111111111:identity/
jane@example.com",ReplyToEmailAddress="jane@example.com"
```

輸出：

```
{
  "UserPool": {
      "SchemaAttributes": [
          {
              "Name": "sub",
```

```
                "StringAttributeConstraints": {
                    "MinLength": "1",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": true,
                "AttributeDataType": "String",
                "Mutable": false
            },
            {
                "Name": "name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "given_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "family_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "middle_name",
```

```
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "nickname",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "preferred_username",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "profile",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "picture",
```

```
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "website",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "email",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "AttributeDataType": "Boolean",
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "Name": "email_verified",
                "Mutable": true
            },
            {
                "Name": "gender",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
```

```
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "birthdate",
            "StringAttributeConstraints": {
                "MinLength": "10",
                "MaxLength": "10"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "zoneinfo",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "locale",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "phone_number",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
```

```
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "AttributeDataType": "Boolean",
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "Name": "phone_number_verified",
            "Mutable": true
        },
        {
            "Name": "address",
            "StringAttributeConstraints": {
                "MinLength": "0",
                "MaxLength": "2048"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "String",
            "Mutable": true
        },
        {
            "Name": "updated_at",
            "NumberAttributeConstraints": {
                "MinValue": "0"
            },
            "DeveloperOnlyAttribute": false,
            "Required": false,
            "AttributeDataType": "Number",
            "Mutable": true
        }
    ],
    "MfaConfiguration": "OFF",
    "Name": "MyUserPool",
    "LastModifiedDate": 1547837788.189,
    "AdminCreateUserConfig": {
        "UnusedAccountValidityDays": 7,
        "AllowAdminCreateUserOnly": false
    },
    "EmailConfiguration": {
        "ReplyToEmailAddress": "jane@example.com",
```

```
            "SourceArn": "arn:aws:ses:us-east-1:111111111111:identity/
jane@example.com"
        },
        "Policies": {
            "PasswordPolicy": {
                "RequireLowercase": true,
                "RequireSymbols": true,
                "RequireNumbers": true,
                "MinimumLength": 8,
                "RequireUppercase": true
            }
        },
        "UsernameAttributes": [
            "email"
        ],
        "CreationDate": 1547837788.189,
        "EstimatedNumberOfUsers": 0,
        "Id": "us-west-2_aaaaaaaaa",
        "LambdaConfig": {}
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreateUserPool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.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * 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 CreateUserPool {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolName>\s

                Where:
                    userPoolName - The name to give your user pool when it's
 created.
                """;

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

        String userPoolName = args[0];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String id = createPool(cognitoClient, userPoolName);
        System.out.println("User pool ID: " + id);
        cognitoClient.close();
    }

    public static String createPool(CognitoIdentityProviderClient cognitoClient,
 String userPoolName) {
        try {
```

```
            CreateUserPoolRequest request = CreateUserPoolRequest.builder()
                    .poolName(userPoolName)
                    .build();

            CreateUserPoolResponse response =
 cognitoClient.createUserPool(request);
            return response.userPool().id();

        } catch (CognitoIdentityProviderException 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CreateUserPool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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭CreateUserPoolClient配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CreateUserPoolClient。

CLI

AWS CLI

若要建立使用者集區用戶端

此範例會建立具有兩個明確授權流程的新使用者集區用戶端：使用者 _ 密碼驗證和 ADMIN_NO_SRP_AUTH。

命令：

```
aws cognito-idp create-user-pool-client --user-pool-id us-west-2_aaaaaaaaa
  --client-name MyNewClient --no-generate-secret --explicit-auth-flows
 "USER_PASSWORD_AUTH" "ADMIN_NO_SRP_AUTH"
```

輸出：

```
{
  "UserPoolClient": {
      "UserPoolId": "us-west-2_aaaaaaaaa",
      "ClientName": "MyNewClient",
      "ClientId": "6p3bs000no6a4ue1idruvd05ad",
      "LastModifiedDate": 1548697449.497,
      "CreationDate": 1548697449.497,
      "RefreshTokenValidity": 30,
      "ExplicitAuthFlows": [
          "USER_PASSWORD_AUTH",
          "ADMIN_NO_SRP_AUTH"
      ],
      "AllowedOAuthFlowsUserPoolClient": false
  }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreateUserPoolClient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.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientReques
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRespon

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
```

```
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * 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 CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <clientName> <userPoolId>\s

                Where:
                    clientName - The name for the user pool client to create.
                    userPoolId - The ID for the user pool.
                """;

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

        String clientName = args[0];
        String userPoolId = args[1];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPoolClient(cognitoClient, clientName, userPoolId);
        cognitoClient.close();
    }

    public static void createPoolClient(CognitoIdentityProviderClient
 cognitoClient, String clientName,
            String userPoolId) {
```

```
        try {
            CreateUserPoolClientRequest request =
 CreateUserPoolClientRequest.builder()
                    .clientName(clientName)
                    .userPoolId(userPoolId)
                    .build();

            CreateUserPoolClientResponse response =
 cognitoClient.createUserPoolClient(request);
            System.out.println("User pool " +
 response.userPoolClient().clientName() + " created. ID: "
                    + response.userPoolClient().clientId());

        } catch (CognitoIdentityProviderException 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CreateUserPoolClient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 SDK 使用此服務。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：

- 使用 Lambda 函數自動確認已知使用者
- 使用 Lambda 函數自動遷移已知的使用者
- 在 Amazon Cognito 使用者身份驗證後，使用 Lambda 函數寫入自訂活動資料

C++

適用於 C++ 的 SDK

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

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

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考DeleteUser中的。

CLI

AWS CLI

若要刪除使用者

此範例會刪除使用者。

命令：

```
aws cognito-idp delete-user --access-token ACCESS_TOKEN
```

- 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。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(userAccessToken string) error {
 _, err := actor.CognitoClient.DeleteUser(context.TODO(),
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考DeleteUser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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ForgotPassword配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ForgotPassword。

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 Lambda 函數自動遷移已知的使用者

CLI

AWS CLI

強制變更密碼

下列forgot-password範例會傳送訊息至 jane@example.com 以變更其密碼。

```
aws cognito-idp forgot-password --client-id 38fjsnc484p94kpqsnet7mpld0 --username
 jane@example.com
```

輸出：

```
{
    "CodeDeliveryDetails": {
        "Destination": "j***@e***.com",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ForgotPassword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。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(clientId string, userName string)
 (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(context.TODO(),
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考ForgotPassword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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`InitiateAuth`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`InitiateAuth`。

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 Lambda 函數自動確認已知使用者
- 使用 Lambda 函數自動遷移已知的使用者
- 使用需要 MFA 的使用者集區註冊使用者
- 在 Amazon Cognito 使用者身份驗證後，使用 Lambda 函數寫入自訂活動資料

.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>
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest
```

```
        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考InitiateAuth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。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(clientId string, userName string, password
 string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(context.TODO(),
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
```

```
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

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

```
const initiateAuth = ({ username, password, clientId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new InitiateAuthCommand({
    AuthFlow: AuthFlowType.USER_PASSWORD_AUTH,
    AuthParameters: {
      USERNAME: username,
      PASSWORD: password,
    },
    ClientId: clientId,
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考InitiateAuth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。

此範例示範如何使用追蹤的裝置開始進行身分驗證。若要完成登入，用戶端必須正確回應安全遠端密碼 (SRP) 挑戰。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
```

```
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
```

```
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
```

```
            return auth_tokens
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件InitiateAuth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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListUserPools配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListUserPools。

.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 Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListUserPools中的。

CLI

AWS CLI

若要列出使用者集區

此範例會列出多達 20 個使用者集區。

命令：

```
aws cognito-idp list-user-pools --max-results 20
```

輸出：

```
{
  "UserPools": [
      {
         "CreationDate": 1547763720.822,
         "LastModifiedDate": 1547763720.822,
         "LambdaConfig": {},
         "Id": "us-west-2_aaaaaaaaa",
         "Name": "MyUserPool"
      }
  ]
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListUserPools中的。

Go

SDK for Go V2

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

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
```

```
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考ListUserPools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.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;
```

```
/**
 * 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 ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient
 cognitoClient) {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response =
 cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID "
 + userpool.id());
            });

        } catch (CognitoIdentityProviderException 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ListUserPools中的。

Rust

適用於 Rust 的 SDK

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

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client.list_user_pools().max_results(10).send().await?;
    let pools = response.user_pools();
    println!("User pools:");
    for pool in pools {
        println!("  ID:              {}", pool.id().unwrap_or_default());
        println!("  Name:            {}", pool.name().unwrap_or_default());
        println!("  Lambda Config:   {:?}", pool.lambda_config().unwrap());
        println!(
            "  Last modified:   {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!(
            "  Creation date:   {:?}",
            pool.creation_date().unwrap().to_chrono_utc()
        );
        println!();
    }
    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK ListUserPools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 SDK 使用此服務。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：

- 使用需要 MFA 的使用者集區註冊使用者

.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>
/// Get a list of users for the Amazon Cognito user pool.
/// </summary>
/// <param name="userPoolId">The user pool ID.</param>
/// <returns>A list of users.</returns>
public async Task<List<UserType>> ListUsersAsync(string userPoolId)
{
    var request = new ListUsersRequest
    {
        UserPoolId = userPoolId
    };

    var users = new List<UserType>();

    var usersPaginator = _cognitoService.Paginators.ListUsers(request);
    await foreach (var response in usersPaginator.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。

CLI

AWS CLI

若要列出使用者

此範例最多可列出 20 位使用者。

命令：

```
aws cognito-idp list-users --user-pool-id us-west-2_aaaaaaaaa --limit 20
```

輸出：

```
{
  "Users": [
      {
          "Username": "22704aa3-fc10-479a-97eb-2af5806bd327",
          "Enabled": true,
          "UserStatus": "FORCE_CHANGE_PASSWORD",
          "UserCreateDate": 1548089817.683,
          "UserLastModifiedDate": 1548089817.683,
          "Attributes": [
              {
                  "Name": "sub",
                  "Value": "22704aa3-fc10-479a-97eb-2af5806bd327"
              },
              {
                  "Name": "email_verified",
                  "Value": "true"
              },
              {
                  "Name": "email",
                  "Value": "mary@example.com"
              }
          ]
      }
  ]
```

```
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考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.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;

/**
 * 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) {

        final String usage = """

                Usage:
                    <userPoolId>\s

                Where:
```

```
                    userPoolId - The ID given to your user pool when it's
created.
                """";

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

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }

    public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {
        try {
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created "
                        + user.userCreateDate());
            });

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

    // Shows how to list users by using a filter.
    public static void listUsersFilter(CognitoIdentityProviderClient
cognitoClient, String userPoolId) {
```

```
        try {
            String filter = "email = \"tblue@noserver.com\"";
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .filter(filter)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User with filter applied " + user.username()
 + " Status " + user.userStatus()
                        + " Created " + user.userCreateDate());
            });

        } catch (CognitoIdentityProviderException 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。

```
const listUsers = ({ userPoolId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ListUsersCommand({
    UserPoolId: userPoolId,
  });

  return client.send(command);
```

```
};
```

- 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(userPoolId: String) {

    val request = ListUsersRequest {
        this.userPoolId = userPoolId
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

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

Python

適用於 Python (Boto3) 的 SDK

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

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
 self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
```

```
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            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。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ResendConfirmationCode配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ResendConfirmationCode。

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：

- 使用需要 MFA 的使用者集區註冊使用者

.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>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
```

```
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
_cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
{response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ResendConfirmationCode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::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest
request;
        request.SetUsername(userName);
```

```
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome
outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                    << std::endl;
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考ResendConfirmationCode中的。

CLI

AWS CLI

若要重新傳送確認碼

下列 `resend-confirmation-code` 範例會傳送確認碼給使用者 `jane`。

```
aws cognito-idp resend-confirmation-code \
    --client-id 12a3b456c7de890f11g123hijk \
    --username jane
```

輸出：

```
{
    "CodeDeliveryDetails": {
        "Destination": "j***@e***.com",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
```

```
    }
}
```

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的註冊及確認使用者帳戶。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ResendConfirmationCode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。

```
   public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
           String userName) {
       try {
           ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                   .clientId(clientId)
                   .username(userName)
                   .build();

           ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
           System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

       } catch (CognitoIdentityProviderException 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ResendConfirmationCode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。

```
const resendConfirmationCode = ({ clientId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ResendConfirmationCodeCommand({
    ClientId: clientId,
    Username: username,
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ResendConfirmationCode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 resendConfirmationCode(clientIdVal: String?, userNameVal: String?) {
    val codeRequest = ResendConfirmationCodeRequest {
        clientId = clientIdVal
        username = userNameVal
```

```
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

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

Python

適用於 Python (Boto3) 的 SDK

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

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret
```

```
    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ResendConfirmationCode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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭RespondToAuthChallenge配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用RespondToAuthChallenge。

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：

- 使用需要 MFA 的使用者集區註冊使用者

CLI

AWS CLI

若要回應身分驗證挑戰

此範例回應使用初始化身份驗證啟動的授權挑戰。這是對 NEW_PASSWORD_REQUIRED 挑戰的回應。它為使用者 jane@example.com 設定密碼。

命令：

```
aws cognito-idp respond-to-auth-challenge --client-id 3n4b5urk1ft4fl3mg5e62d9ado
 --challenge-name NEW_PASSWORD_REQUIRED --challenge-responses
 USERNAME=jane@example.com,NEW_PASSWORD="password" --session "SESSION_TOKEN"
```

輸出：

```
{
  "ChallengeParameters": {},
  "AuthenticationResult": {
      "AccessToken": "ACCESS_TOKEN",
      "ExpiresIn": 3600,
      "TokenType": "Bearer",
      "RefreshToken": "REFRESH_TOKEN",
      "IdToken": "ID_TOKEN",
      "NewDeviceMetadata": {
          "DeviceKey": "us-west-2_fec070d2-fa88-424a-8ec8-b26d7198eb23",
          "DeviceGroupKey": "-wt2ha1Zd"
      }
  }
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考RespondToAuthChallenge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。

```
const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考RespondToAuthChallenge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。

使用追蹤的裝置登入。若要完成登入，用戶端必須正確回應安全遠端密碼 (SRP) 挑戰。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
```

```
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
```

```
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件RespondToAuthChallenge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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭SignUp配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用SignUp。

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 Lambda 函數自動確認已知使用者
- 使用 Lambda 函數自動遷移已知的使用者
- 使用需要 MFA 的使用者集區註冊使用者

.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>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
```

```
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考SignUp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::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
```

```
        Aws::CognitoIdentityProvider::Model::SignUpRequest request;
        request.AddUserAttributes(
                Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                        "email").WithValue(email));
        request.SetUsername(userName);
        request.SetPassword(password);
        request.SetClientId(clientID);
        Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
                client.SignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "The signup request for " << userName << " was
successful."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
            std::cout
                    << "The username already exists. Please enter a different
username."
                    << std::endl;
            userExists = true;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考SignUp中的。

CLI

AWS CLI

若要將使用者登出

此範例註冊了 jane@example.com。

命令：

```
aws cognito-idp sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username jane@example.com --password PASSWORD --user-attributes
 Name="email",Value="jane@example.com" Name="name",Value="Jane"
```

輸出：

```
{
  "UserConfirmed": false,
  "UserSub": "e04d60a6-45dc-441c-a40b-e25a787d4862"
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考SignUp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。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
```

```
    {Name: aws.String("email"), Value: aws.String(userEmail)},
   },
  })
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    log.Println(*invalidPassword.Message)
   } else {
    log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
   }
  } else {
   confirmed = output.UserConfirmed
  }
  return confirmed, err
}
```

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

```
    public static void signUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
```

```
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

        } catch (CognitoIdentityProviderException 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SignUp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。

```
const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考SignUp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 signUp(clientIdVal: String?, userNameVal: String?, passwordVal:
 String?, emailVal: String?) {
    val userAttrs = AttributeType {
        name = "email"
        value = emailVal
    }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest = SignUpRequest {
        userAttributes = userAttrsList
        username = userNameVal
        clientId = clientIdVal
        password = passwordVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

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

Python

適用於 Python (Boto3) 的 SDK

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

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
 Cognito
        to send an email to the specified email address. The email contains a
 code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
 whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
```

```
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件SignUp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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭UpdateUserPool配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用UpdateUserPool。

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 Lambda 函數自動確認已知使用者
- 使用 Lambda 函數自動遷移已知的使用者
- 在 Amazon Cognito 使用者身份驗證後，使用 Lambda 函數寫入自訂活動資料

CLI

AWS CLI

若要更新使用者集區

此範例會將標籤新增至使用者集區。

命令：

```
aws cognito-idp update-user-pool --user-pool-id us-west-2_aaaaaaaaa --user-pool-
tags Team=Blue,Area=West
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考UpdateUserPool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。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
```

```
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考UpdateUserPool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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`VerifySoftwareToken`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`VerifySoftwareToken`。

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：

- 使用需要 MFA 的使用者集區註冊使用者

.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>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
```

```
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考VerifySoftwareToken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::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);
```

```
        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考VerifySoftwareToken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。

```
   // Verify the TOTP and register for MFA.
   public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
        try {
            VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                    .userCode(code)
                    .session(session)
                    .build();

            VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
```

```
            System.out.println("The status of the token is " +
 verifyResponse.statusAsString());

        } catch (CognitoIdentityProviderException 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VerifySoftwareToken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。

```
const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考VerifySoftwareToken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。

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(sessionVal: String?, codeVal: String?) {
    val tokenRequest = VerifySoftwareTokenRequest {
        userCode = codeVal
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

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

Python

適用於 Python (Boto3) 的 SDK

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

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
 initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
```

```
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件VerifySoftwareToken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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使 AWS 用開發套件的 Amazon Cognito 身分供應商的案例

下列程式碼範例說明如何在具有 AWS SDK 的 Amazon Cognito 身分識別提供者中實作常見案例。這些案例會向您展示如何呼叫 Amazon Cognito 身分提供者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

- 使用開發套件，透過 Lambda 函數自動確認已知的 Amazon Cognito 認知使用者 AWS
- 使用開發套件，透過 Lambda 函數自動遷移已知的 Amazon Cognito 知使用者 AWS
- 使用需要 MFA 使用者使用開發套件的 Amazon Cognito 使用者集區註冊使用者 AWS
- 使用開發套件 AWS 進行 Amazon Cognito 使用者身份驗證之後，使用 Lambda 函數寫入自訂活動

使用開發套件，透過 Lambda 函數自動確認已知的 Amazon Cognito 認知使用者 AWS

下列程式碼範例顯示如何使用 Lambda 函數自動確認已知的 Amazon Cognito 使用者。

- 設定使用者集區以呼叫`PreSignUp`觸發器的 Lambda 函數。
- 使用 Amazon Cognito 可註冊一個用戶。

- Lambda 函數會掃描 DynamoDB 表格，並自動確認已知使用者。
- 以新使用者身分登入，然後清理資源。

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。

```
// AutoConfirm separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type AutoConfirm struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) AutoConfirm {
 scenario := AutoConfirm{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}
```

```
// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
 PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(userPoolId string, functionArn
 string) {
 log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
 Cognito.\n" +
  "This trigger happens when a user signs up, and lets your function take action
 before the main Cognito\n" +
  "sign up processing occurs.\n")
 err := runner.cognitoActor.UpdateTriggers(
  userPoolId,
  actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
 trigger.\n",
  functionArn, userPoolId)
}

// SignUpUser signs up a user from the known user table with a password you
 specify.
func (runner *AutoConfirm) SignUpUser(clientId string, usersTable string)
 (string, string) {
 log.Println("Let's sign up a user to your Cognito user pool. When the user's
 email matches an email in the\n" +
  "DynamoDB known users table, it is automatically verified and the user is
 confirmed.")

 knownUsers, err := runner.helper.GetKnownUsers(usersTable)
 if err != nil {
  panic(err)
 }
 userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
 knownUsers.UserNameList())
 user := knownUsers.Users[userChoice]

 var signedUp bool
 var userConfirmed bool
 password := runner.questioner.AskPassword("Enter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !signedUp {
```

```
  log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
 user.UserEmail)
  userConfirmed, err = runner.cognitoActor.SignUp(clientId, user.UserName,
 password, user.UserEmail)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("Enter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   signedUp = true
  }
 }
 log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(clientId string, userName string, password
 string) string {
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 log.Printf("Let's sign in as %v...\n", userName)
 authResult, err := runner.cognitoActor.SignIn(clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 log.Println(strings.Repeat("-", 88))
 return *authResult.AccessToken
}

// Run runs the scenario.
func (runner *AutoConfirm) Run(stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup()
  }
```

```
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(stackOutputs["TableName"])

 runner.AddPreSignUpTrigger(stackOutputs["UserPoolId"],
 stackOutputs["AutoConfirmFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
 userName, password := runner.SignUpUser(stackOutputs["UserPoolClientId"],
 stackOutputs["TableName"])
 runner.helper.ListRecentLogEvents(stackOutputs["AutoConfirmFunction"])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
  runner.SignInUser(stackOutputs["UserPoolClientId"], userName, password))

 runner.resources.Cleanup()

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

使用 Lambda 函數處理PreSignUp觸發器。

```
const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}
```

```
// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
 error) {
 log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "PreSignUp_SignUp" {
  // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the
 response from this handler.
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserEmail: event.Request.UserAttributes["email"],
 }
 log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
 output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key:       user.GetKey(),
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Error looking up email %v.\n", user.UserEmail)
  return event, err
 }
 if output.Item == nil {
  log.Printf("Email %v not found. Email verification is required.\n",
 user.UserEmail)
  return event, err
```

```
 }

 err = attributevalue.UnmarshalMap(output.Item, &user)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
  return event, err
 }

 if user.UserName != event.UserName {
  log.Printf("UserEmail %v found, but stored UserName '%v' does not match
 supplied UserName '%v'. Verification is required.\n",
   user.UserEmail, user.UserName, event.UserName)
 } else {
  log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.
\n", user.UserEmail, user.UserName)
  event.Response.AutoConfirmUser = true
  event.Response.AutoVerifyEmail = true
 }

 return event, err
}

func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

建立執行一般工作的結構。

```
// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(stackName string) (actions.StackOutputs, error)
```

```
 PopulateUserTable(tableName string)
 GetKnownUsers(tableName string) (actions.UserList, error)
 AddKnownUser(tableName string, user actions.User)
 ListRecentLogEvents(functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(stackName), nil
}
```

```
// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(tableName string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(tableName string) (actions.UserList,
 error) {
 knownUsers, err := helper.dynamoActor.Scan(tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(tableName string, user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(functionName)
 if err != nil {
  panic(err)
```

```
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

創建一個包裝 Amazon Cognito 操作的結構。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
```

```
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
```

```
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(clientId string, userName string, password
 string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(context.TODO(),
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

```
// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(clientId string, userName string)
 (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(context.TODO(),
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(clientId string, code string,
 userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(context.TODO(),
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}
```

```
// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(userAccessToken string) error {
 _, err := actor.CognitoClient.DeleteUser(context.TODO(),
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
```

```
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

建立包裝 DynamoDB 動作的結構。

```
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
```

```
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(tableName string) error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(context.TODO(),
 &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
```

```
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(tableName string) (UserList, error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(context.TODO(), &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(tableName string, user User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

建立包裝 CloudWatch 記錄動作的結構。

```
type CloudWatchLogsActions struct {
```

```
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(functionName string)
 (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(context.TODO(),
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(functionName string,
 logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(context.TODO(),
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
```

```
}
```

創建一個包裝 AWS CloudFormation 動作的結構。

```
// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(stackName string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(context.TODO(),
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

清理資源。

```
// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger
```

```
 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(resources.userPoolId,
 triggerList...)
  if err != nil {
```

```
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Go API 參考》中的下列主題。
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件，透過 Lambda 函數自動遷移已知的 Amazon Cognito 知使用者 AWS

下列程式碼範例說明如何使用 Lambda 函數自動遷移已知的 Amazon Cognito 使用者。

- 設定使用者集區以呼叫`MigrateUser`觸發器的 Lambda 函數。
- 使用不在使用者集區中的使用者名稱和電子郵件登入 Amazon Cognito。
- Lambda 函數會掃描 DynamoDB 表格，並自動將已知使用者移轉至使用者集區。
- 執行忘記密碼流程以重設已遷移使用者的密碼。
- 以新使用者身分登入，然後清理資源。

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。

```
import (
 "errors"
 "fmt"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// MigrateUser separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type MigrateUser struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) MigrateUser {
 scenario := MigrateUser{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
```

```
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
 MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(userPoolId string, functionArn
 string) {
 log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
 Cognito.\n" +
  "This trigger happens when an unknown user signs in, and lets your function
 take action before Cognito\n" +
  "rejects the user.\n\n")
 err := runner.cognitoActor.UpdateTriggers(
  userPoolId,
  actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
 trigger.\n",
  functionArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser adds a new user to the known users table and signs that user in to
 Amazon Cognito.
func (runner *MigrateUser) SignInUser(usersTable string, clientId string) (bool,
 actions.User) {
 log.Println("Let's sign in a user to your Cognito user pool. When the username
 and email matches an entry in the\n" +
  "DynamoDB known users table, the email is automatically verified and the user
 is migrated to the Cognito user pool.")

 user := actions.User{}
 user.UserName = runner.questioner.Ask("\nEnter a username:")
 user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This
 email will be used to confirm user migration\n" +
  "during this example:")
```

```
 runner.helper.AddKnownUser(usersTable, user)

 var err error
 var resetRequired *types.PasswordResetRequiredException
 var authResult *types.AuthenticationResultType
 signedIn := false
 for !signedIn && resetRequired == nil {
  log.Printf("Signing in to Cognito as user '%v'. The expected result is a
 PasswordResetRequiredException.\n\n", user.UserName)
  authResult, err = runner.cognitoActor.SignIn(clientId, user.UserName, "_")
  if err != nil {
   if errors.As(err, &resetRequired) {
    log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
 DynamoDB known users table.\n"+
     "User migration is started and a password reset is required.",
 user.UserName)
   } else {
    panic(err)
   }
  } else {
   log.Printf("User '%v' successfully signed in. This is unexpected and probably
 means you have not\n"+
    "cleaned up a previous run of this scenario, so the user exist in the Cognito
 user pool.\n"+
    "You can continue this example and select to clean up resources, or manually
 remove\n"+
    "the user from your user pool and try again.", user.UserName)
   runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
   signedIn = true
  }
 }

 log.Println(strings.Repeat("-", 88))
 return resetRequired != nil, user
}

// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(clientId string, user actions.User) {
 wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user
 to Cognito, you must be able to receive a confirmation\n"+
  "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
 if !wantCode {
```

```
 log.Println("To complete this example and successfully migrate a user to
Cognito, you must enter an email\n" +
  "you own that can receive a confirmation code.")
 return
}
codeDelivery, err := runner.cognitoActor.ForgotPassword(clientId, user.UserName)
if err != nil {
 panic(err)
}
log.Printf("\nA confirmation code has been sent to %v.",
*codeDelivery.Destination)
code := runner.questioner.Ask("Check your email and enter it here:")

confirmed := false
password := runner.questioner.AskPassword("\nEnter a password that has at least
eight characters, uppercase, lowercase, numbers and symbols.\n"+
 "(the password will not display as you type):", 8)
for !confirmed {
 log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
 err = runner.cognitoActor.ConfirmForgotPassword(clientId, code, user.UserName,
password)
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   password = runner.questioner.AskPassword("\nEnter another password:", 8)
  } else {
   panic(err)
  }
 } else {
  confirmed = true
 }
}
log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
log.Println("Signing in with your username and password...")
authResult, err := runner.cognitoActor.SignIn(clientId, user.UserName, password)
if err != nil {
 panic(err)
}
log.Printf("Successfully signed in. Your access token starts with: %v...\n",
(*authResult.AccessToken)[:10])
runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
*authResult.AccessToken)

log.Println(strings.Repeat("-", 88))
```

```
}

// Run runs the scenario.
func (runner *MigrateUser) Run(stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup()
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]

 runner.AddMigrateUserTrigger(stackOutputs["UserPoolId"],
 stackOutputs["MigrateUserFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers,
 actions.UserMigration)
 resetNeeded, user := runner.SignInUser(stackOutputs["TableName"],
 stackOutputs["UserPoolClientId"])
 if resetNeeded {
  runner.helper.ListRecentLogEvents(stackOutputs["MigrateUserFunction"])
  runner.ResetPassword(stackOutputs["UserPoolClientId"], user)
 }

 runner.resources.Cleanup()

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

使用 Lambda 函數處理`MigrateUser`觸發器。

```
const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsMigrateUser)
 (events.CognitoEventUserPoolsMigrateUser, error) {
 log.Printf("Received migrate trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 if event.TriggerSource != "UserMigration_Authentication" {
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName: event.UserName,
 }
 log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
 filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
 expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
 if err != nil {
  log.Printf("Error building expression to query for user '%v'.\n",
 user.UserName)
  return event, err
 }
 output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName:                 aws.String(tableName),
  FilterExpression:          expr.Filter(),
  ExpressionAttributeNames:  expr.Names(),
  ExpressionAttributeValues: expr.Values(),
 })
```

```
 if err != nil {
  log.Printf("Error looking up user '%v'.\n", user.UserName)
  return event, err
 }
 if output.Items == nil || len(output.Items) == 0 {
  log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
  return event, err
 }

 var users []UserInfo
 err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
  return event, err
 }

 user = users[0]
 log.Printf("UserName '%v' found with email %v. User is migrated and must reset
 password.\n", user.UserName, user.UserEmail)
 event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes =
 map[string]string{
  "email":          user.UserEmail,
  "email_verified": "true", // email_verified is required for the forgot password
 flow.
 }
 event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus =
 "RESET_REQUIRED"
 event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

 return event, err
}

func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

建立執行一般工作的結構。

```
// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(stackName string) (actions.StackOutputs, error)
 PopulateUserTable(tableName string)
 GetKnownUsers(tableName string) (actions.UserList, error)
 AddKnownUser(tableName string, user actions.User)
 ListRecentLogEvents(functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
```

```
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(tableName string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(tableName string) (actions.UserList,
 error) {
 knownUsers, err := helper.dynamoActor.Scan(tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(tableName string, user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(tableName, user)
 if err != nil {
  panic(err)
 }
}
```

```
// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

創建一個包裝 Amazon Cognito 操作的結構。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
```

```
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

```
// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(clientId string, userName string, password
 string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(context.TODO(),
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
```

```
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(clientId string, userName string)
 (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(context.TODO(),
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(clientId string, code string,
 userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(context.TODO(),
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
```

```
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(userAccessToken string) error {
 _, err := actor.CognitoClient.DeleteUser(context.TODO(),
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
```

```
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

建立包裝 DynamoDB 動作的結構。

```
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
```

```
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(tableName string) error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
```

```
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(context.TODO(),
 &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(tableName string) (UserList, error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(context.TODO(), &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(tableName string, user User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
```

```
 }
 return err
}
```

建立包裝 CloudWatch 記錄動作的結構。

```
type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(functionName string)
 (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(context.TODO(),
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(functionName string,
 logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(context.TODO(),
 &cloudwatchlogs.GetLogEventsInput{
```

```
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

創建一個包裝 AWS CloudFormation 動作的結構。

```
// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(stackName string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(context.TODO(),
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

清理資源。

```
// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
```

```
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Go API 參考》中的下列主題。
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用需要 MFA 使用者使用開發套件的 Amazon Cognito 使用者集區註冊使用者 AWS

下列程式碼範例示範如何：

- 使用使用者名稱、密碼和電子郵件地址註冊並確認使用者。

- 透過將 MFA 應用程式與使用者建立關聯，以設定多重要素身分驗證。
- 使用密碼和 MFA 代碼登入。

.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 CognitoBasics;

public class CognitoBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Cognito.
        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<IAmazonCognitoIdentityProvider>()
            .AddTransient<CognitoWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<CognitoBasics>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
```

```
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var cognitoWrapper = host.Services.GetRequiredService<CognitoWrapper>();

        Console.WriteLine(new string('-', 80));
        UiMethods.DisplayOverview();
        Console.WriteLine(new string('-', 80));

        // clientId - The app client Id value that you get from the AWS CDK
script.
        var clientId = configuration["ClientId"]; // "*** REPLACE WITH CLIENT ID
VALUE FROM CDK SCRIPT";

        // poolId - The pool Id that you get from the AWS CDK script.
        var poolId = configuration["PoolId"]!; // "*** REPLACE WITH POOL ID VALUE
FROM CDK SCRIPT";
        var userName = configuration["UserName"];
        var password = configuration["Password"];
        var email = configuration["Email"];

        // If the username wasn't set in the configuration file,
        // get it from the user now.
        if (userName is null)
        {
            do
            {
                Console.Write("Username: ");
                userName = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(userName));
        }
        Console.WriteLine($"\nUsername: {userName}");

        // If the password wasn't set in the configuration file,
        // get it from the user now.
        if (password is null)
        {
            do
            {
                Console.Write("Password: ");
                password = Console.ReadLine();
            }
```

```
            while (string.IsNullOrEmpty(password));
        }

        // If the email address wasn't set in the configuration file,
        // get it from the user now.
        if (email is null)
        {
            do
            {
                Console.Write("Email: ");
                email = Console.ReadLine();
            } while (string.IsNullOrEmpty(email));
        }

        // Now sign up the user.
        Console.WriteLine($"\nSigning up {userName} with email address:
{email}");
        await cognitoWrapper.SignUpAsync(clientId, userName, password, email);

        // Add the user to the user pool.
        Console.WriteLine($"Adding {userName} to the user pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        UiMethods.DisplayTitle("Get confirmation code");
        Console.WriteLine($"Conformation code sent to {userName}.");
        Console.Write("Would you like to send a new code? (Y/N) ");
        var answer = Console.ReadLine();

        if (answer!.ToLower() == "y")
        {
            await cognitoWrapper.ResendConfirmationCodeAsync(clientId, userName);
            Console.WriteLine("Sending a new confirmation code");
        }

        Console.Write("Enter confirmation code (from Email): ");
        var code = Console.ReadLine();

        await cognitoWrapper.ConfirmSignupAsync(clientId, code, userName);

        UiMethods.DisplayTitle("Checking status");
        Console.WriteLine($"Rechecking the status of {userName} in the user
pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);
```

```
        Console.WriteLine($"Setting up authenticator for {userName} in the user
 pool");
        var setupResponse = await cognitoWrapper.InitiateAuthAsync(clientId,
 userName, password);

        var setupSession = await
 cognitoWrapper.AssociateSoftwareTokenAsync(setupResponse.Session);
        Console.Write("Enter the 6-digit code displayed in Google Authenticator:
 ");
        var setupCode = Console.ReadLine();

        var setupResult = await
 cognitoWrapper.VerifySoftwareTokenAsync(setupSession, setupCode);
        Console.WriteLine($"Setup status: {setupResult}");

        Console.WriteLine($"Now logging in {userName} in the user pool");
        var authSession = await cognitoWrapper.AdminInitiateAuthAsync(clientId,
 poolId, userName, password);

        Console.Write("Enter a new 6-digit code displayed in Google
 Authenticator: ");
        var authCode = Console.ReadLine();

        var authResult = await
 cognitoWrapper.AdminRespondToAuthChallengeAsync(userName, clientId, authCode,
 authSession, poolId);
        Console.WriteLine($"Authenticated and received access token:
 {authResult.AccessToken}");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cognito scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }
}


using System.Net;

namespace CognitoActions;

/// <summary>
/// Methods to perform Amazon Cognito Identity Provider actions.
/// </summary>
public class CognitoWrapper
```

```
{
    private readonly IAmazonCognitoIdentityProvider _cognitoService;

    /// <summary>
    /// Constructor for the wrapper class containing Amazon Cognito actions.
    /// </summary>
    /// <param name="cognitoService">The Amazon Cognito client object.</param>
    public CognitoWrapper(IAmazonCognitoIdentityProvider cognitoService)
    {
        _cognitoService = cognitoService;
    }

    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
 ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }


    /// <summary>
    /// Get a list of users for the Amazon Cognito user pool.
    /// </summary>
    /// <param name="userPoolId">The user pool ID.</param>
    /// <returns>A list of users.</returns>
    public async Task<List<UserType>> ListUsersAsync(string userPoolId)
    {
        var request = new ListUsersRequest
        {
            UserPoolId = userPoolId
        };
```

```
        var users = new List<UserType>();

        var usersPaginator = _cognitoService.Paginators.ListUsers(request);
        await foreach (var response in usersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Respond to an admin authentication challenge.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="clientId">The client ID.</param>
    /// <param name="mfaCode">The multi-factor authentication code.</param>
    /// <param name="session">The current application session.</param>
    /// <param name="clientId">The user pool ID.</param>
    /// <returns>The result of the authentication response.</returns>
    public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
        string userName,
        string clientId,
        string mfaCode,
        string session,
        string userPoolId)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>();
        challengeResponses.Add("USERNAME", userName);
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        var respondToAuthChallengeRequest = new
AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };
```

```
        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }


    /// <summary>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }


    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
_cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
```

```
        var secretCode = tokenResponse.SecretCode;

        Console.WriteLine($"Use the following secret code to set up the
 authenticator: {secretCode}");

        return tokenResponse.Session;
    }


    /// <summary>
    /// Initiate an admin auth request.
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
 userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
    }

    /// <summary>
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
    /// <param name="password">The password for the user who is authenticating.</
param>
```

```
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }

    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
string userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
```

```
        }
        return false;
    }


    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</
param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string
 deviceKey, string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }


    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };
```

```
        var response = await
 _cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
 {response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }


    /// <summary>
    /// Get the specified user from an Amazon Cognito user pool with
 administrator access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
    /// <returns>Async task.</returns>
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
 poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
    }


    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
```

```
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的下列主題。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp

- VerifySoftwareToken

C++

適用於 C++ 的 SDK

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

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

//! Scenario that adds a user to an Amazon Cognito user pool.
/*!
  \sa gettingStartedWithUserPools()
  \param clientID: Client ID associated with an Amazon Cognito user pool.
  \param userPoolID: An Amazon Cognito user pool ID.
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::Cognito::gettingStartedWithUserPools(const Aws::String &clientID,
                                                  const Aws::String &userPoolID,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    printAsterisksLine();
    std::cout
            << "Welcome to the Amazon Cognito example scenario."
            << std::endl;
    printAsterisksLine();

    std::cout
            << "This scenario will add a user to an Amazon Cognito user pool."
            << std::endl;
    const Aws::String userName = askQuestion("Enter a new username: ");
    const Aws::String password = askQuestion("Enter a new password: ");
    const Aws::String email = askQuestion("Enter a valid email for the user: ");
```

```
   std::cout << "Signing up " << userName << std::endl;

   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
   bool userExists = false;
   do {
       // 1. Add a user with a username, password, and email address.
       Aws::CognitoIdentityProvider::Model::SignUpRequest request;
       request.AddUserAttributes(
               Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                       "email").WithValue(email));
       request.SetUsername(userName);
       request.SetPassword(password);
       request.SetClientId(clientID);
       Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
               client.SignUp(request);

       if (outcome.IsSuccess()) {
           std::cout << "The signup request for " << userName << " was
successful."
                     << std::endl;
       }
       else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
           std::cout
                   << "The username already exists. Please enter a different
username."
                   << std::endl;
           userExists = true;
       }
       else {
           std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
   } while (userExists);

   printAsterisksLine();
   std::cout << "Retrieving status of " << userName << " in the user pool."
             << std::endl;
   // 2. Confirm that the user was added to the user pool.
   if (!checkAdminUserStatus(userName, userPoolID, client)) {
```

```
        return false;
    }

    std::cout << "A confirmation code was sent to " << email << "." << std::endl;

    bool resend = askYesNoQuestion("Would you like to send a new code? (y/n) ");
    if (resend) {
        // Request a resend of the confirmation code to the email address.
(ResendConfirmationCode)
        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest
request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome
outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                    << std::endl;
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();

    {
        // 4. Send the confirmation code that's received in the email.
(ConfirmSignUp)
        const Aws::String confirmationCode = askQuestion(
                "Enter the confirmation code that was emailed: ");
        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);
```

```
        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "Rechecking the status of " << userName << " in the user pool."
              << std::endl;
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    printAsterisksLine();

    std::cout << "Initiating authorization using the username and password."
              << std::endl;

    Aws::String session;
    // 5. Initiate authorization with username and password. (AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID,  userName, password,
 session, client)) {
        return false;
    }

    printAsterisksLine();

    std::cout
            << "Starting setup of time-based one-time password (TOTP) multi-
factor authentication (MFA)."
            << std::endl;

    {
        // 6. Request a setup key for one-time password (TOTP)
        //    multi-factor authentication (MFA). (AssociateSoftwareToken)
```

```
        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest
 request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome
 outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for
 example Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
    askQuestion("Type enter to continue...", alwaysTrueTest);

    printAsterisksLine();

    {
        Aws::String userCode = askQuestion(
                "Enter the 6 digit code displayed in the authenticator app: ");

        //  7. Send the MFA code copied from an authenticator app.
 (VerifySoftwareToken)
```

```
        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "You have completed the MFA authentication setup." << std::endl;
    std::cout << "Now, sign in." << std::endl;

    // 8. Initiate authorization again with username and password.
(AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID, userName, password,
session, client)) {
        return false;
    }

    Aws::String accessToken;
    {
        Aws::String mfaCode = askQuestion(
                "Re-enter the 6 digit code displayed in the authenticator app:
");

        // 9. Send a new MFA code copied from an authenticator app.
(AdminRespondToAuthChallenge)
        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
        request.AddChallengeResponses("USERNAME", userName);
        request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
```

```
        request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
        request.SetClientId(clientID);
        request.SetUserPoolId(userPoolID);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
                client.AdminRespondToAuthChallenge(request);

        if (outcome.IsSuccess()) {
            std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                      << std::endl;

            accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

        std::cout << "You have successfully added a user to Amazon Cognito."
                  << std::endl;
    }

    if (askYesNoQuestion("Would you like to delete the user that you just added?
(y/n) ")) {
        // 10. Delete the user that you just added. (DeleteUser)
        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
```

```
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return true;
}

//! Routine which checks the user status in an Amazon Cognito user pool.
/*!
 \sa checkAdminUserStatus()
 \param userName: A username.
 \param userPoolID: An Amazon Cognito user pool ID.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::checkAdminUserStatus(const Aws::String &userName,
                                           const Aws::String &userPoolID,
                                           const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

 Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                          outcome.GetResult().GetUserStatus()) << std::endl;
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() <<
 std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}

//! Routine which starts authorization of an Amazon Cognito user.
//! This routine requires administrator credentials.
/*!
 \sa adminInitiateAuthorization()
 \param clientID: Client ID of tracked device.
 \param userPoolID: An Amazon Cognito user pool ID.
 \param userName: A username.
 \param password: A password.
 \param sessionResult: String to receive a session token.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::adminInitiateAuthorization(const Aws::String &clientID,
                                                 const Aws::String &userPoolID,
                                                 const Aws::String &userName,
                                                 const Aws::String &password,
                                                 Aws::String &sessionResult,
                                                 const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

 Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
            client.AdminInitiateAuth(request);

    if (outcome.IsSuccess()) {
        std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
        sessionResult = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                  << 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》中的下列主題。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

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.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRespons
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito_scenario_user_pool_with_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * “ChallengeName”: “MFA_SETUP”).
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
 * key. This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 * MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 * prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
 InvalidKeyException {
        final String usage = """

                Usage:
                    <clientId> <poolId>

                Where:
                    clientId - The app client Id value that you can get from the
 AWS CDK script.
                    poolId - The pool Id that you can get from the AWS CDK
 script.\s
                """;
```

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

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("*** Enter your password");
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();

        System.out.println("1. Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Getting " + userName + " in the user pool");
        getAdminUser(identityProviderClient, userName, poolId);

        System.out
                .println("*** Conformation code sent to " + userName + ". Would
you like to send a new code? (Yes/No)");
        System.out.println(DASHES);

        System.out.println(DASHES);
        String ans = in.nextLine();

        if (ans.compareTo("Yes") == 0) {
```

```
            resendConfirmationCode(identityProviderClient, clientId, userName);
            System.out.println("3. Sending a new confirmation code");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enter confirmation code that was emailed");
        String code = in.nextLine();
        confirmSignUp(identityProviderClient, clientId, code, userName);
        System.out.println("Rechecking the status of " + userName + " in the user
pool");
        getAdminUser(identityProviderClient, userName, poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Invokes the initiateAuth to sign in");
        AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        String mySession = authResponse.session();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to
generate a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient,
mySession);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
```

```
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9.  Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
            String userName, String clientId, String mfaCode, String session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
                .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
                .clientId(clientId)
                .challengeResponses(challengeResponses)
                .session(session)
                .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
                + respondToAuthChallengeResult.authenticationResult());
    }
```

```
   // Verify the TOTP and register for MFA.
   public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
       try {
           VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                   .userCode(code)
                   .session(session)
                   .build();

           VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
           System.out.println("The status of the token is " +
verifyResponse.statusAsString());

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

   public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
           String clientId, String userName, String password, String userPoolId)
{
       try {
           Map<String, String> authParameters = new HashMap<>();
           authParameters.put("USERNAME", userName);
           authParameters.put("PASSWORD", password);

           AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                   .clientId(clientId)
                   .userPoolId(userPoolId)
                   .authParameters(authParameters)
                   .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                   .build();

           AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
           System.out.println("Result Challenge is : " +
response.challengeName());
           return response;
```

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

        return null;
    }

    public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
        AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                .session(session)
                .build();

        AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
                .associateSoftwareToken(softwareTokenRequest);
        String secretCode = tokenResponse.secretCode();
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }

    public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
            System.out.println(userName + " was confirmed");

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

    public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
```

```
            String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                    .clientId(clientId)
                    .username(userName)
                    .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

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

    public static void signUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

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

```
    }

    public static void getAdminUser(CognitoIdentityProviderClient
 identityProviderClient, String userName,
            String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

            AdminGetUserResponse response =
 identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

        } catch (CognitoIdentityProviderException 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。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

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。

為獲得最佳體驗，請複製 GitHub 儲存庫並執行此範例。下列程式碼代表完整範例應用程式的範例。

```
import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { signUp } from "../../../actions/sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username, password, email) => {
  if (!(username && password && email)) {
    throw new Error(
      `Username, password, and email must be provided as arguments to the 'sign-
up' command.`,
    );
  }
};

const signUpHandler = async (commands) => {
  const [_, username, password, email] = commands;

  try {
    validateUser(username, password, email);
    /**
```

```
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    log(`Signing up.`);
    await signUp({ clientId, username, password, email });
    log(`Signed up. A confirmation email has been sent to: ${email}.`);
    log(`Run 'confirm-sign-up ${username} <code>' to confirm your account.`);
  } catch (err) {
    log(err);
  }
};

export { signUpHandler };

const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};

import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { confirmSignUp } from "../../../actions/confirm-sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username) => {
```

```
  if (!username) {
    throw new Error(
      `Username name is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const validateCode = (code) => {
  if (!code) {
    throw new Error(
      `Verification code is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const confirmSignUpHandler = async (commands) => {
  const [_, username, code] = commands;

  try {
    validateUser(username);
    validateCode(code);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    log(`Confirming user.`);
    await confirmSignUp({ clientId, username, code });
    log(
      `User confirmed. Run 'admin-initiate-auth ${username} <password>' to sign
 in.`,
    );
  } catch (err) {
    log(err);
  }
};

export { confirmSignUpHandler };

const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});
```

```
  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};

import qrcode from "qrcode-terminal";
import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminInitiateAuth } from "../../../actions/admin-initiate-auth.js";
import { associateSoftwareToken } from "../../../actions/associate-software-
token.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";

const handleMfaSetup = async (session, username) => {
  const { SecretCode, Session } = await associateSoftwareToken(session);

  // Store the Session for use with 'VerifySoftwareToken'.
  process.env.SESSION = Session;

  console.log(
    "Scan this code in your preferred authenticator app, then run 'verify-
software-token' to finish the setup.",
  );
  qrcode.generate(
    `otpauth://totp/${username}?secret=${SecretCode}`,
    { small: true },
    console.log,
  );
};

const handleSoftwareTokenMfa = (session) => {
  // Store the Session for use with 'AdminRespondToAuthChallenge'.
  process.env.SESSION = session;
};

const validateClient = (id) => {
  if (!id) {
    throw new Error(
      `User pool client id is missing. Did you run 'create-user-pool'?`,
```

```
    );
  }
};

const validateId = (id) => {
  if (!id) {
    throw new Error(`User pool id is missing. Did you run 'create-user-pool'?`);
  }
};

const validateUser = (username, password) => {
  if (!(username && password)) {
    throw new Error(
      `Username and password must be provided as arguments to the 'admin-
initiate-auth' command.`,
    );
  }
};

const adminInitiateAuthHandler = async (commands) => {
  const [_, username, password] = commands;

  try {
    validateUser(username, password);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    validateId(userPoolId);
    validateClient(clientId);

    log("Signing in.");
    const { ChallengeName, Session } = await adminInitiateAuth({
      clientId,
      userPoolId,
      username,
      password,
    });

    if (ChallengeName === "MFA_SETUP") {
      log("MFA setup is required.");
      return handleMfaSetup(Session, username);
    }

    if (ChallengeName === "SOFTWARE_TOKEN_MFA") {
      handleSoftwareTokenMfa(Session);
```

```
      log(`Run 'admin-respond-to-auth-challenge ${username} <totp>'`);
    }
  } catch (err) {
    log(err);
  }
};

export { adminInitiateAuthHandler };

const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};

import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminRespondToAuthChallenge } from "../../../actions/admin-respond-to-
auth-challenge.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";
import { FILE_USER_POOLS } from "./constants.js";

const verifyUsername = (username) => {
  if (!username) {
    throw new Error(
      `Username is missing. It must be provided as an argument to the 'admin-
respond-to-auth-challenge' command.`,
    );
  }
};

const verifyTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) is missing. It must be provided as an
 argument to the 'admin-respond-to-auth-challenge' command.`,
    );
  }
```

```
};

const storeAccessToken = (token) => {
  process.env.AccessToken = token;
};

const adminRespondToAuthChallengeHandler = async (commands) => {
  const [_, username, totp] = commands;

  try {
    verifyUsername(username);
    verifyTotp(totp);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    const session = process.env.SESSION;

    const { AuthenticationResult } = await adminRespondToAuthChallenge({
      clientId,
      userPoolId,
      username,
      totp,
      session,
    });

    storeAccessToken(AuthenticationResult.AccessToken);

    log("Successfully authenticated.");
  } catch (err) {
    log(err);
  }
};

export { adminRespondToAuthChallengeHandler };

const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
```

```
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};

import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { verifySoftwareToken } from "../../../actions/verify-software-token.js";

const validateTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) must be provided to the 'validate-
software-token' command.`,
    );
  }
};
const verifySoftwareTokenHandler = async (commands) => {
  const [_, totp] = commands;

  try {
    validateTotp(totp);

    log("Verifying TOTP.");
    await verifySoftwareToken(totp);
    log("TOTP Verified. Run 'admin-initiate-auth' again to sign-in.");
  } catch (err) {
    console.log(err);
  }
};

export { verifySoftwareTokenHandler };

const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
```

```
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱《AWS SDK for JavaScript API 參考》中的下列主題。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

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。

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 TIP: To set up the required user pool, run the AWS Cloud Development
 Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
cognito_scenario_user_pool_with_mfa.

 This code example performs the following operations:

 1. Invokes the signUp method to sign up a user.
 2. Invokes the adminGetUser method to get the user's confirmation status.
 3. Invokes the ResendConfirmationCode method if the user requested another code.
 4. Invokes the confirmSignUp method.
 5. Invokes the initiateAuth to sign in. This results in being prompted to
 set up TOTP (time-based one-time password). (The response is “ChallengeName”:
 “MFA_SETUP”).
 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.
 This can be used with Google Authenticator.
 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 MFA.
 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
```

```
            <clientId> <poolId>
        Where:
            clientId - The app client Id value that you can get from the AWS CDK
script.
            poolId - The pool Id that you can get from the AWS CDK script.
    """

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    val clientId = args[0]
    val poolId = args[1]

    // Use the console to get data from the user.
    println("*** Enter your use name")
    val inOb = Scanner(System.`in`)
    val userName = inOb.nextLine()
    println(userName)

    println("*** Enter your password")
    val password: String = inOb.nextLine()

    println("*** Enter your email")
    val email = inOb.nextLine()

    println("*** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("*** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("*** Conformation code sent to $userName. Would you like to send a
new code? (Yes/No)")
    val ans = inOb.nextLine()

    if (ans.compareTo("Yes") == 0) {
        println("*** Sending a new confirmation code")
        resendConfirmationCode(clientId, userName)
    }
    println("*** Enter the confirmation code that was emailed")
    val code = inOb.nextLine()
    confirmSignUp(clientId, code, userName)
```

```
    println("*** Rechecking the status of $userName in the user pool")
    getAdminUser(userName, poolId)

    val authResponse = checkAuthMethod(clientId, userName, password, poolId)
    val mySession = authResponse.session
    val newSession = getSecretForAppMFA(mySession)
    println("*** Enter the 6-digit code displayed in Google Authenticator")
    val myCode = inOb.nextLine()

    // Verify the TOTP and register for MFA.
    verifyTOTP(newSession, myCode)
    println("*** Re-enter a 6-digit code displayed in Google Authenticator")
    val mfaCode: String = inOb.nextLine()
    val authResponse1 = checkAuthMethod(clientId, userName, password, poolId)
    val session2 = authResponse1.session
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun checkAuthMethod(clientIdVal: String, userNameVal: String,
 passwordVal: String, userPoolIdVal: String): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest = AdminInitiateAuthRequest {
        clientId = clientIdVal
        userPoolId = userPoolIdVal
        authParameters = authParas
        authFlow = AuthFlowType.AdminUserPasswordAuth
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}

suspend fun resendConfirmationCode(clientIdVal: String?, userNameVal: String?) {
    val codeRequest = ResendConfirmationCodeRequest {
        clientId = clientIdVal
        username = userNameVal
```

```
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(userName: String, clientIdVal: String?,
 mfaCode: String, sessionVal: String?) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest = AdminRespondToAuthChallengeRequest {
        challengeName = ChallengeNameType.SoftwareTokenMfa
        clientId = clientIdVal
        challengeResponses = challengeResponsesOb
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(sessionVal: String?, codeVal: String?) {
    val tokenRequest = VerifySoftwareTokenRequest {
        userCode = codeVal
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
```

```
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest = AssociateSoftwareTokenRequest {
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}

suspend fun confirmSignUp(clientIdVal: String?, codeVal: String?, userNameVal:
 String?) {
    val signUpRequest = ConfirmSignUpRequest {
        clientId = clientIdVal
        confirmationCode = codeVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}

suspend fun getAdminUser(userNameVal: String?, poolIdVal: String?) {
    val userRequest = AdminGetUserRequest {
        username = userNameVal
        userPoolId = poolIdVal
    }
```

```
    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}

suspend fun signUp(clientIdVal: String?, userNameVal: String?, passwordVal:
 String?, emailVal: String?) {
    val userAttrs = AttributeType {
        name = "email"
        value = emailVal
    }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest = SignUpRequest {
        userAttributes = userAttrsList
        username = userNameVal
        clientId = clientIdVal
        password = passwordVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- 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。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers

- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

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。

建立包裝案例中所用 Amazon Cognito 函數的類別。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def _secret_hash(self, user_name):
        """
        Calculates a secret hash from a user name and a client secret.
```

```
        :param user_name: The user name to use when calculating the hash.
        :return: The secret hash.
        """
        key = self.client_secret.encode()
        msg = bytes(user_name + self.client_id, "utf-8")
        secret_hash = base64.b64encode(
            hmac.new(key, msg, digestmod=hashlib.sha256).digest()
        ).decode()
        logger.info("Made secret hash for %s: %s.", user_name, secret_hash)
        return secret_hash

    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
Cognito
        to send an email to the specified email address. The email contains a
code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
```

```
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery


    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
```

```
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
                self.user_pool_id,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users


    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
 server.

        If the user pool is configured to require MFA and this is the first sign-
in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
 this
                 returns an access token that can be used to get AWS credentials.
 Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
 application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
 self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
```

```
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
not "
                        "configured for TOTP MFA. This example requires TOTP
MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the
user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            response.pop("ResponseMetadata", None)
            return response


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step
of
        a two-factor sign-in. When sign-in is successful, it returns an access
token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
```

```
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You
might have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s:
%s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result


    def confirm_mfa_device(
        self,
```

```
        user_name,
        device_key,
        device_group_key,
        device_password,
        access_token,
        aws_srp,
    ):
        """
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a
new
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example
uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False.
When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt +
device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value,
srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
```

```
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
code.

        Signing in with a tracked device requires that the client respond to the
SRP
        protocol. The scenario associated with this example uses the warrant
package
        to help with SRP calculations.
```

```
        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
 {response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
```

```
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
 user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
 self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

建立可執行案例的類別。此範例也會註冊要由 Amazon Cognito 追蹤的 MFA 裝置，並向您展示如何使用來自追蹤裝置的密碼和資訊登入。這樣就不需要輸入新的 MFA 代碼。

```
def run_scenario(cognito_idp_client, user_pool_id, client_id):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
```

```
    print("Welcome to the Amazon Cognito user signup with MFA demo.")
    print("-" * 88)

    cog_wrapper = CognitoIdentityProviderWrapper(
        cognito_idp_client, user_pool_id, client_id
    )

    user_name = q.ask("Let's sign up a new user. Enter a user name: ",
q.non_empty)
    password = q.ask("Enter a password for the user: ", q.non_empty)
    email = q.ask("Enter a valid email address that you own: ", q.non_empty)
    confirmed = cog_wrapper.sign_up_user(user_name, password, email)
    while not confirmed:
        print(
            f"User {user_name} requires confirmation. Check {email} for "
            f"a verification code."
        )
        confirmation_code = q.ask("Enter the confirmation code from the email: ")
        if not confirmation_code:
            if q.ask("Do you need another confirmation code (y/n)? ",
q.is_yesno):
                delivery = cog_wrapper.resend_confirmation(user_name)
                print(
                    f"Confirmation code sent by {delivery['DeliveryMedium']} "
                    f"to {delivery['Destination']}."
                )
        else:
            confirmed = cog_wrapper.confirm_user_sign_up(user_name,
confirmation_code)
    print(f"User {user_name} is confirmed and ready to use.")
    print("-" * 88)

    print("Let's get a list of users in the user pool.")
    q.ask("Press Enter when you're ready.")
    users = cog_wrapper.list_users()
    if users:
        print(f"Found {len(users)} users:")
        pp(users)
    else:
        print("No users found.")
    print("-" * 88)

    print("Let's sign in and get an access token.")
    auth_tokens = None
```

```
    challenge = "ADMIN_USER_PASSWORD_AUTH"
    response = {}
    while challenge is not None:
        if challenge == "ADMIN_USER_PASSWORD_AUTH":
            response = cog_wrapper.start_sign_in(user_name, password)
            challenge = response["ChallengeName"]
        elif response["ChallengeName"] == "MFA_SETUP":
            print("First, we need to set up an MFA application.")
            qr_img = qrcode.make(
                f"otpauth://totp/{user_name}?secret={response['SecretCode']}"
            )
            qr_img.save("qr.png")
            q.ask(
                "Press Enter to see a QR code on your screen. Scan it into an MFA
"
                "application, such as Google Authenticator."
            )
            webbrowser.open("qr.png")
            mfa_code = q.ask(
                "Enter the verification code from your MFA application: ",
q.non_empty
            )
            response = cog_wrapper.verify_mfa(response["Session"], mfa_code)
            print(f"MFA device setup {response['Status']}")
            print("Now that an MFA application is set up, let's sign in again.")
            print(
                "You might have to wait a few seconds for a new MFA code to
appear in "
                "your MFA application."
            )
            challenge = "ADMIN_USER_PASSWORD_AUTH"
        elif response["ChallengeName"] == "SOFTWARE_TOKEN_MFA":
            auth_tokens = None
            while auth_tokens is None:
                mfa_code = q.ask(
                    "Enter a verification code from your MFA application: ",
q.non_empty
                )
                auth_tokens = cog_wrapper.respond_to_mfa_challenge(
                    user_name, response["Session"], mfa_code
                )
            print(f"You're signed in as {user_name}.")
            print("Here's your access token:")
            pp(auth_tokens["AccessToken"])
```

```
            print("And your device information:")
            pp(auth_tokens["NewDeviceMetadata"])
            challenge = None
        else:
            raise Exception(f"Got unexpected challenge
 {response['ChallengeName']}")
    print("-" * 88)

    device_group_key = auth_tokens["NewDeviceMetadata"]["DeviceGroupKey"]
    device_key = auth_tokens["NewDeviceMetadata"]["DeviceKey"]
    device_password = base64.standard_b64encode(os.urandom(40)).decode("utf-8")

    print("Let's confirm your MFA device so you don't have re-enter MFA tokens
 for it.")
    q.ask("Press Enter when you're ready.")
    cog_wrapper.confirm_mfa_device(
        user_name,
        device_key,
        device_group_key,
        device_password,
        auth_tokens["AccessToken"],
        aws_srp,
    )
    print(f"Your device {device_key} is confirmed.")
    print("-" * 88)

    print(
        f"Now let's sign in as {user_name} from your confirmed device
 {device_key}.\n"
        f"Because this device is tracked by Amazon Cognito, you won't have to re-
enter an MFA code."
    )
    q.ask("Press Enter when ready.")
    auth_tokens = cog_wrapper.sign_in_with_tracked_device(
        user_name, password, device_key, device_group_key, device_password,
 aws_srp
    )
    print("You're signed in. Your access token is:")
    pp(auth_tokens["AccessToken"])
    print("-" * 88)

    print("Don't forget to delete your user pool when you're done with this
 example.")
    print("\nThanks for watching!")
```

```
    print("-" * 88)


def main():
    parser = argparse.ArgumentParser(
        description="Shows how to sign up a new user with Amazon Cognito and
 associate "
        "the user with an MFA application for multi-factor authentication."
    )
    parser.add_argument(
        "user_pool_id", help="The ID of the user pool to use for the example."
    )
    parser.add_argument(
        "client_id", help="The ID of the client application to use for the
 example."
    )
    args = parser.parse_args()
    try:
        run_scenario(boto3.client("cognito-idp"), args.user_pool_id,
 args.client_id)
    except Exception:
        logging.exception("Something went wrong with the demo.")


if __name__ == "__main__":
    main()
```

- 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。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode

- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件 AWS 進行 Amazon Cognito 使用者身份驗證之後，使用 Lambda 函數寫入自訂活動

下列程式碼範例示範如何在 Amazon Cognito 使用者身分驗證後，使用 Lambda 函數撰寫自訂活動資料。

- 使用管理員功能將使用者新增至使用者集區。
- 設定使用者集區以呼叫PostAuthentication觸發器的 Lambda 函數。
- 將新用戶登錄到 Amazon Cognito。
- Lambda 函數會將自訂資訊寫入 CloudWatch 日誌和 DynamoDB 資料表。
- 從 DynamoDB 表格取得並顯示自訂資料，然後清理資源。

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。

```
// ActivityLog separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ActivityLog struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
```

```
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) ActivityLog {
 scenario := ActivityLog{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
 credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(userPoolId string, tableName string)
 (string, string) {
 log.Println("To facilitate this example, let's add a user to the user pool using
 administrator privileges.")
 users, err := runner.helper.GetKnownUsers(tableName)
 if err != nil {
  panic(err)
 }
 user := users.Users[0]
 log.Printf("Adding known user %v to the user pool.\n", user.UserName)
 err = runner.cognitoActor.AdminCreateUser(userPoolId, user.UserName,
 user.UserEmail)
 if err != nil {
  panic(err)
 }
 pwSet := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !pwSet {
  log.Printf("\nSetting password for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.AdminSetUserPassword(userPoolId, user.UserName,
 password)
  if err != nil {
```

```
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   pwSet = true
  }
 }

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
 PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(userPoolId string,
 activityLogArn string) {
 log.Println("Let's add a Lambda function to handle the PostAuthentication
 trigger from Cognito.\n" +
  "This trigger happens after a user is authenticated, and lets your function
 take action, such as logging\n" +
  "the outcome.")
 err := runner.cognitoActor.UpdateTriggers(
  userPoolId,
  actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
 aws.String(activityLogArn)})
 if err != nil {
  panic(err)
 }
 runner.resources.triggers = append(runner.resources.triggers,
 actions.PostAuthentication)
 log.Printf("Lambda function %v added to user pool %v to handle
 PostAuthentication Cognito trigger.\n",
  activityLogArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(clientId string, userName string, password
 string) {
```

```
 log.Printf("Now we'll sign in user %v and check the results in the logs and the
 DynamoDB table.", userName)
 runner.questioner.Ask("Press Enter when you're ready.")
 authResult, err := runner.cognitoActor.SignIn(clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Println("Sign in successful.",
  "The PostAuthentication Lambda handler writes custom information to CloudWatch
 Logs.")

 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
 table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(tableName string, userName
 string) {
 log.Println("The PostAuthentication handler also writes login data to the
 DynamoDB table.")
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 users, err := runner.helper.GetKnownUsers(tableName)
 if err != nil {
  panic(err)
 }
 for _, user := range users.Users {
  if user.UserName == userName {
   log.Println("The last login info for the user in the known users table is:")
   log.Printf("\t%+v", *user.LastLogin)
  }
 }
 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *ActivityLog) Run(stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup()
  }
 }()
```

```
 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(stackOutputs["TableName"])
 userName, password := runner.AddUserToPool(stackOutputs["UserPoolId"],
 stackOutputs["TableName"])

 runner.AddActivityLogTrigger(stackOutputs["UserPoolId"],
 stackOutputs["ActivityLogFunctionArn"])
 runner.SignInUser(stackOutputs["UserPoolClientId"], userName, password)
 runner.helper.ListRecentLogEvents(stackOutputs["ActivityLogFunction"])
 runner.GetKnownUserLastLogin(stackOutputs["TableName"], userName)

 runner.resources.Cleanup()

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

使用 Lambda 函數處理PostAuthentication觸發器。

```
const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
 format.
type LoginInfo struct {
 UserPoolId string `dynamodbav:"UserPoolId"`
 ClientId   string `dynamodbav:"ClientId"`
 Time       string `dynamodbav:"Time"`
}
```

```
// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string    `dynamodbav:"UserName"`
 UserEmail string    `dynamodbav:"UserEmail"`
 LastLogin LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to
 the logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
 event events.CognitoEventUserPoolsPostAuthentication)
 (events.CognitoEventUserPoolsPostAuthentication, error) {
 log.Printf("Received post authentication trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName:  event.UserName,
  UserEmail: event.Request.UserAttributes["email"],
  LastLogin: LoginInfo{
   UserPoolId: event.UserPoolID,
   ClientId:   event.CallerContext.ClientID,
   Time:       time.Now().Format(time.UnixDate),
  },
 }
 // Write to CloudWatch Logs.
 fmt.Printf("%#v", user)

 // Also write to an external system. This examples uses DynamoDB to demonstrate.
 userMap, err := attributevalue.MarshalMap(user)
```

```
 if err != nil {
  log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
 } else if len(userMap) == 0 {
  log.Printf("User info marshaled to an empty map.")
 } else {
  _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
   Item:      userMap,
   TableName: aws.String(tableName),
  })
  if err != nil {
   log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
  } else {
   log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
  }
 }

 return event, nil
}

func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

建立執行一般工作的結構。

```
// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(stackName string) (actions.StackOutputs, error)
 PopulateUserTable(tableName string)
 GetKnownUsers(tableName string) (actions.UserList, error)
 AddKnownUser(tableName string, user actions.User)
```

```
 ListRecentLogEvents(functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(tableName string) {
```

```
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(tableName string) (actions.UserList,
 error) {
 knownUsers, err := helper.dynamoActor.Scan(tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(tableName string, user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
```

```
 events, err := helper.cwlActor.GetLogEvents(functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

創建一個包裝 Amazon Cognito 操作的結構。

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
```

```
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
```

```
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(clientId string, userName string, password
 string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(context.TODO(),
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

```
// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(clientId string, userName string)
 (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(context.TODO(),
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(clientId string, code string,
 userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(context.TODO(),
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
```

```
func (actor CognitoActions) DeleteUser(userAccessToken string) error {
 _, err := actor.CognitoClient.DeleteUser(context.TODO(),
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
```

```
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

建立包裝 DynamoDB 動作的結構。

```
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
```

```
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(tableName string) error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(context.TODO(),
 &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
```

```
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(tableName string) (UserList, error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(context.TODO(), &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(tableName string, user User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

建立包裝 CloudWatch 記錄動作的結構。

```
type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
```

```
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(functionName string)
 (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(context.TODO(),
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(functionName string,
 logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(context.TODO(),
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

創建一個包裝 AWS CloudFormation 動作的結構。

```
// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(stackName string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(context.TODO(),
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

清理資源。

```
// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger
```

```
 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
```

```
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Go API 參考》中的下列主題。
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使 AWS 用 SDK 的 Amazon Cognito 同步程式碼範例

下列程式碼範例說明如何將 Amazon Cognito 同步與 AWS 軟體開發套件 (SDK) 搭配使用。

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。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱搭配 AWS SDK 使用此服務。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

- 使 AWS 用 SDK 進行 Amazon Cognito 同步的操作
  - 搭ListIdentityPoolUsage配 AWS 開發套件或 CLI 使用

使 AWS 用 SDK 進行 Amazon Cognito 同步的操作

下列程式碼範例示範如何使用 AWS SDK 執行個別 Amazon Cognito 同步動作。這些摘錄會呼叫 Amazon Cognito Sync 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 Amazon Cognito Sync API 參考。

範例

- 搭ListIdentityPoolUsage配 AWS 開發套件或 CLI 使用

搭`ListIdentityPoolUsage`配 AWS 開發套件或 CLI 使用

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

Rust

適用於 Rust 的 SDK

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

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client
        .list_identity_pool_usage()
        .max_results(10)
        .send()
        .await?;

    let pools = response.identity_pool_usages();
    println!("Identity pools:");

    for pool in pools {
        println!(
            "  Identity pool ID:    {}",
            pool.identity_pool_id().unwrap_or_default()
        );
```

```
        println!(
            "  Data storage:        {}",
            pool.data_storage().unwrap_or_default()
        );
        println!(
            "  Sync sessions count: {}",
            pool.sync_sessions_count().unwrap_or_default()
        );
        println!(
            "  Last modified:       {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!();
    }

    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK ListIdentityPoolUsage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 SDK 使用此服務。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

多重租用戶應用程式最佳實務

Amazon Cognito 使用者集區會與多租戶應用程式搭配運作，這些應用程式會產生大量的請求，這些請求必須保留在 Amazon Cognito 若要在客戶群成長時擴充此容量，您可以購買額外的配額容量。

> **Note**
>
> Amazon Cognito 配額是按 AWS 帳戶 和 AWS 區域套用。應用程式中的所有租用戶會共用這些配額。檢閱 Amazon Cognito 服務配額，並確定配額符合預期的數量和應用程式中預期的租用戶數量。

本節說明您可以在相同區域和 AWS 帳戶區域內的 Amazon Cognito 資源之間分隔租用戶實作的方法。您也可以將租用戶分配給多個 AWS 帳戶 或區域，並為每個租用戶提供自己的配額。多區域多租戶的其他優點包括可能的最高層級的隔離、全球分散式使用者的最短網路傳輸時間，以及遵守組織中現有的發佈模式。

單一區域多租戶也可以為您的客戶和系統管理員帶來優勢。

以下清單涵蓋了多租戶與共享資源的一些優點。

多租戶的優勢

一般使用者目錄

多租戶支援客戶在多個應用程式中擁有帳戶的模型。您可以將來自協力廠商提供者的身分識別連結至單一一致的使用者集區 如果使用者設定檔對其租用戶而言是唯一的，任何具有單一使用者集區的多租戶策略都有一個使用者管理的進入點。

共同安全性

在共用使用者集區中，您可以建立單一安全性標準，並將相同的進階安全性、多重要素驗證 (MFA) 和標AWS WAF準套用至所有承租人。由於 AWS WAF Web ACL 必須與其關聯的資源位於 AWS 區域 相同的位置，因此多租用戶可提供複雜資源的共用存取權。當您想要在多區域 Amazon Cognito 應用程式中維持一致的安全組態時，必須套用在資源之間複寫組態的操作標準。

常見的定制

您可以使用自訂使用者集區和身分集區 AWS Lambda。使用者集區中的 Lambda 觸發程序和身分集區中 Amazon Cognito 事件的設定可能會變得複雜。Lambda 函數必須與您的使用者集區或身分

集區位於相同 AWS 區域 的位置。共用 Lambda 函數可針對區域內的自訂驗證流程、使用者移轉、權杖產生及其他功能強制執行標準。

常見訊息

Amazon Simple Notification Service (Amazon SNS) 需要在區域中進行額外的設定，才能傳送簡訊給使用者。您可以使用經過 Amazon SES 驗證的區域內包含的身分和網域來傳送電子郵件訊息。

透過多租戶，您可以在所有租用戶之間共用此組態和維護開銷。由於並非所有 Amazon SNS 和 Amazon SES 都可用 AWS 區域，因此在不同區域之間分割資源需要額外考量。

當您使用自訂訊息提供者時，您會獲得單一 Lambda 函數的通用自訂，以管理訊息傳遞。

託管 UI 在瀏覽器中設置會話 cookie，以便識別已經驗證的用戶。當您在使用者集區中驗證本機使用者時，其工作階段 Cookie 會針對相同使用者集區中的所有應用程式用戶端進行驗證。本機使用者僅存在於您的使用者集區目錄中，不會透過外部 IdP 進行聯合。工作階段 Cookie 的有效期為一小時。您無法變更工作階段 Cookie 持續時間。

有兩種方法可以防止使用託管的 UI 會話 cookie 在應用程序客戶端之間登錄。

- 將您的使用者分成每個租用戶使用者集區。
- 將託管的 UI 登入取代為 Amazon Cognito 使用者集區 API 登入。

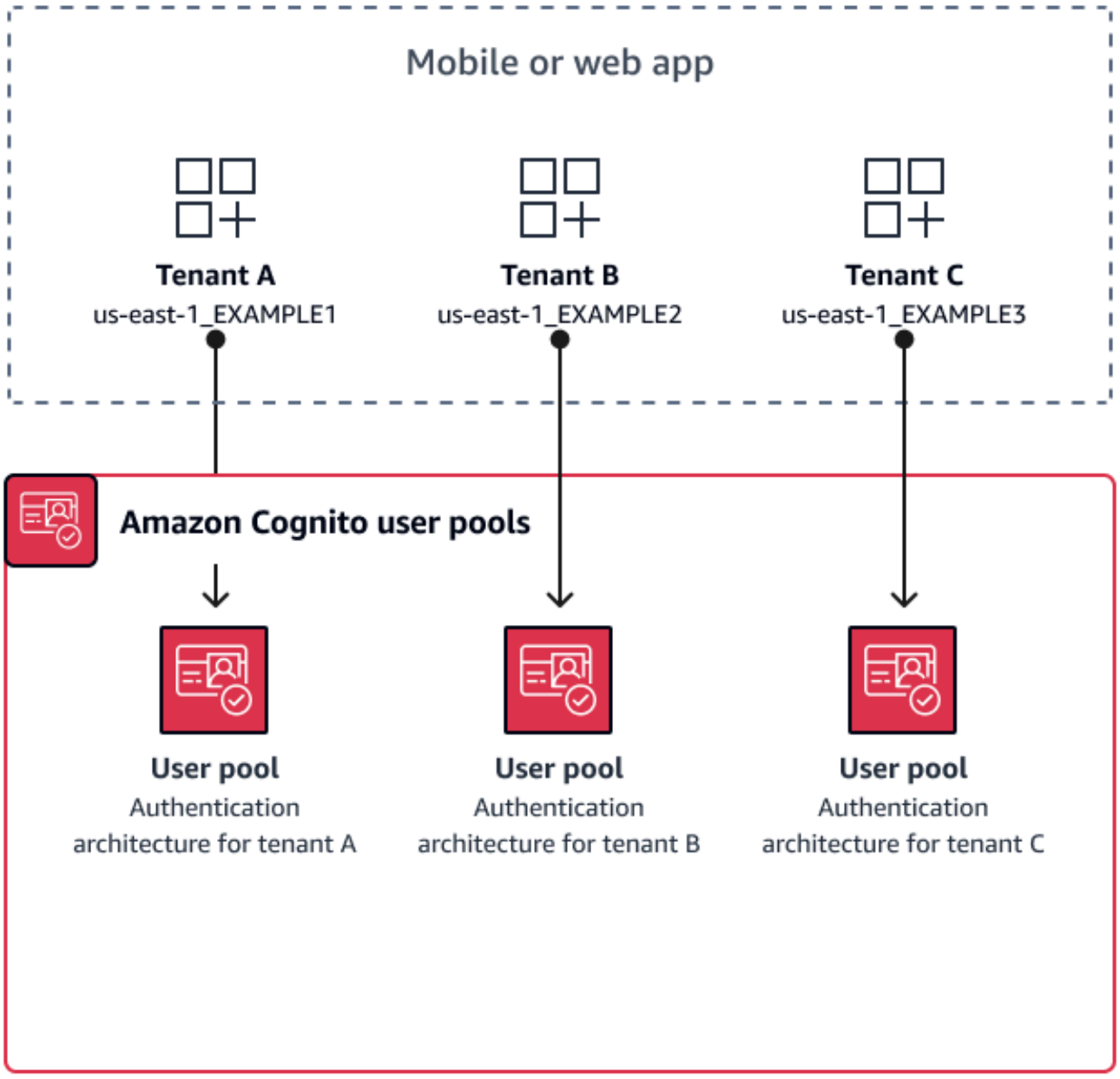
主題

使用者集區多租戶最佳做法

為應用程式中的每個租用戶建立使用者集區。此方法可為各個租用戶提供最大的隔離。您可以針對各個租用戶實作不同組態。使用者集區的租用戶隔離可讓您彈性進行 user-to-tenant 對應。您可以針對相同使用者建立多個設定檔。但每個使用者都必須針對可存取的每個租用戶個別註冊。

使用此方法，您可以為每個承租人獨立設定託管 UI，並將使用者重新導向至其應用程式的承租人特定執行個體。您也可以使用此方法與 Amazon API Gateway 等後端服務整合。

下圖顯示具有專用使用者集區的每個租用戶。



何時實作使用者集區多租戶

當隔離和自訂是您的首要考量時。在具有多個使用者集區的架構中，使用者和租用戶之間的關係可能很複雜。考慮一個例子，你有兩個教育租戶。同一位使用者可能是某個 app 中存取權限限制的學生，而在另一個 app 中則是擁有高等級權限的老師。您可能需要 MFA 在一個應用程序中，但不需要另一個應用程序，或者有不同的密碼策略。由於本機使用者可以使用託管 UI 登入使用者集區中的多個應用程式用戶端，因此當您希望多個租用戶使用託管 UI 登入時，使用者集區多租用戶也是理想的選擇。

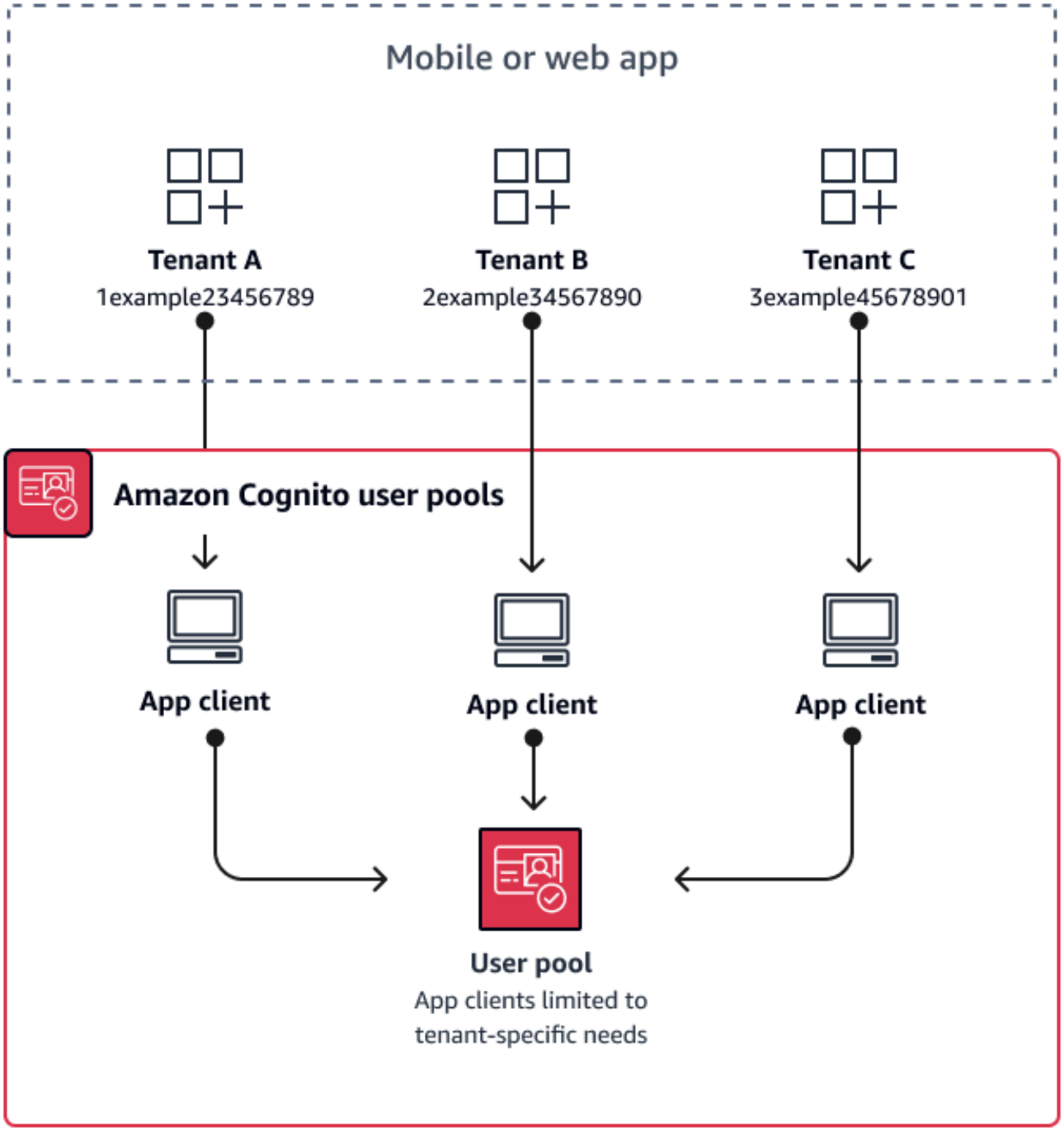
努力程度

使用這種方法，開發和操作的投入量會很高。為了確保您的應用程式系列產生一致且可預測的結果，您必須將 Amazon Cognito 資源與自動化工具整合，並在身份驗證架構日益複雜時維護基準。當您想要為應用程式建立單一起始位置時，您必須建置使用者介面 (UI) 項目，以擷取將使用者路由至正確資源的初始決定。

應用程式用戶端多租戶最佳做法

為應用程式中的每個租用戶建立應用程式用戶端。透過應用程式用戶端多租戶，您可以將任何使用者指派給承租人連結的應用程式用戶端，並保留單一使用者設定檔。由於您可以將使用者集區中的任何或所有身分識別提供者 (IdPs) 指派給應用程式用戶端，因此租用戶應用程式用戶端可以允許使用承租人特定 IdP 進行登入。當使用者存在 IdPs 於多個承租人中時，您可以將其設定檔與多個設定檔連結，以獲得一致的使用者

下圖顯示每個租用戶在共用使用者集區中具有專用應用程式用戶端。



何時實作應用程式用戶端多租戶

當您可以為使用者集區層級的設定選擇通用組態時，例如 Lambda 觸發程序、密碼原則，以及電子郵件和 SMS 訊息的內容和傳遞方式。由於共用使用者集區中的使用者可以登入任何應用程式用戶端，因

此應用程式用戶端多租戶非常適合使用 app-client-specific IdPs 或使用 Amazon Cognito 使用者集區 API 進行登入。應用程式用戶端多租戶也非常適合您要允許使用者在多個應用程式之間轉換的 one-to-many環境。

努力程度

應用程式用戶端多租用需要中等的努力。應用程式用戶端多租用的主要挑戰是租用戶能夠呈現託管 UI Cookie 並在應用程式之間切換的能力。在應用程式用戶端多租戶架構中，請避免在需要隔離的情況下進行託管 UI 登入。您可以使用內建的應用程式用戶端邏輯來散發行動應用程式或 Web 應用程式的連結，也可以建立決定使用者租用的初始 UI 元素。因為您不需要在多個使用者集區和身分集區之間進行標準化和維護設定，因此工作程度較低。

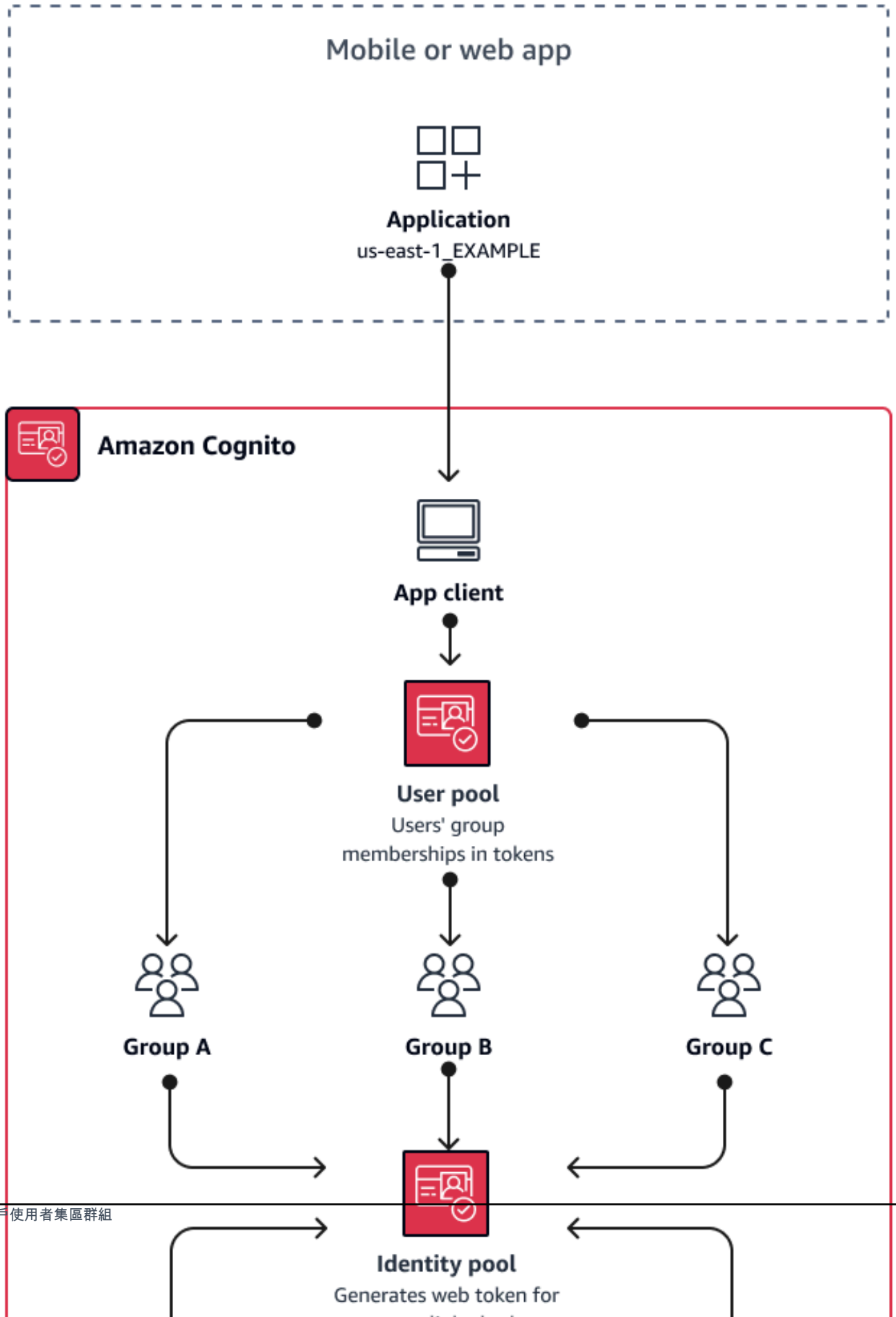
使用者集區群組多租戶最佳做法

當您的架構需要具有身分集區的 Amazon Cognito 使用者集區時，以群組為基礎的多租戶最有效。

用戶池 ID 和訪問令牌包含`cognito:groups`聲明。此外，ID 令牌包含`cognito:roles`和`cognito:preferred_role`聲明。當您的應用程式中進行身份驗證的主要結果是來自身分集區的臨時 AWS 登入資料時，您的使用者群組成員資格可以決定他們收到的 IAM 角色和許可。

舉例來說，假設三個租用戶，每個租用戶將應用程式資產存放在自己的 Amazon S3 儲存貯體中。將每個承租人的使用者指派給相關聯的群組、設定群組的慣用角色，以及將該角色讀取存取權授與其值區。

下圖顯示與使用者集區共用應用程式用戶端和使用者集區的租用戶，以及決定其具有 IAM 角色資格的使用者集區中的專用群組。


Mobile or web app
Application
us-east-1_EXAMPLE
Amazon Cognito
App client
User pool
Users' group
memberships in tokens
Group A
Group B
Group C
Identity pool
Generates web token for

何時實施群組多租戶

訪問 AWS 資源是您的首要關注點時。Amazon Cognito 使用者集區使用者集區中的群組是以角色為基礎的存取控制 (RBAC) 的一種機制。您可以在使用者集區中設定多個群組，並以群組優先順序做出複雜的 RBAC 決策。身分集區可以為優先順序最高的角色、群組中宣告的任何角色或使用者權杖中的其他宣告指派認證。

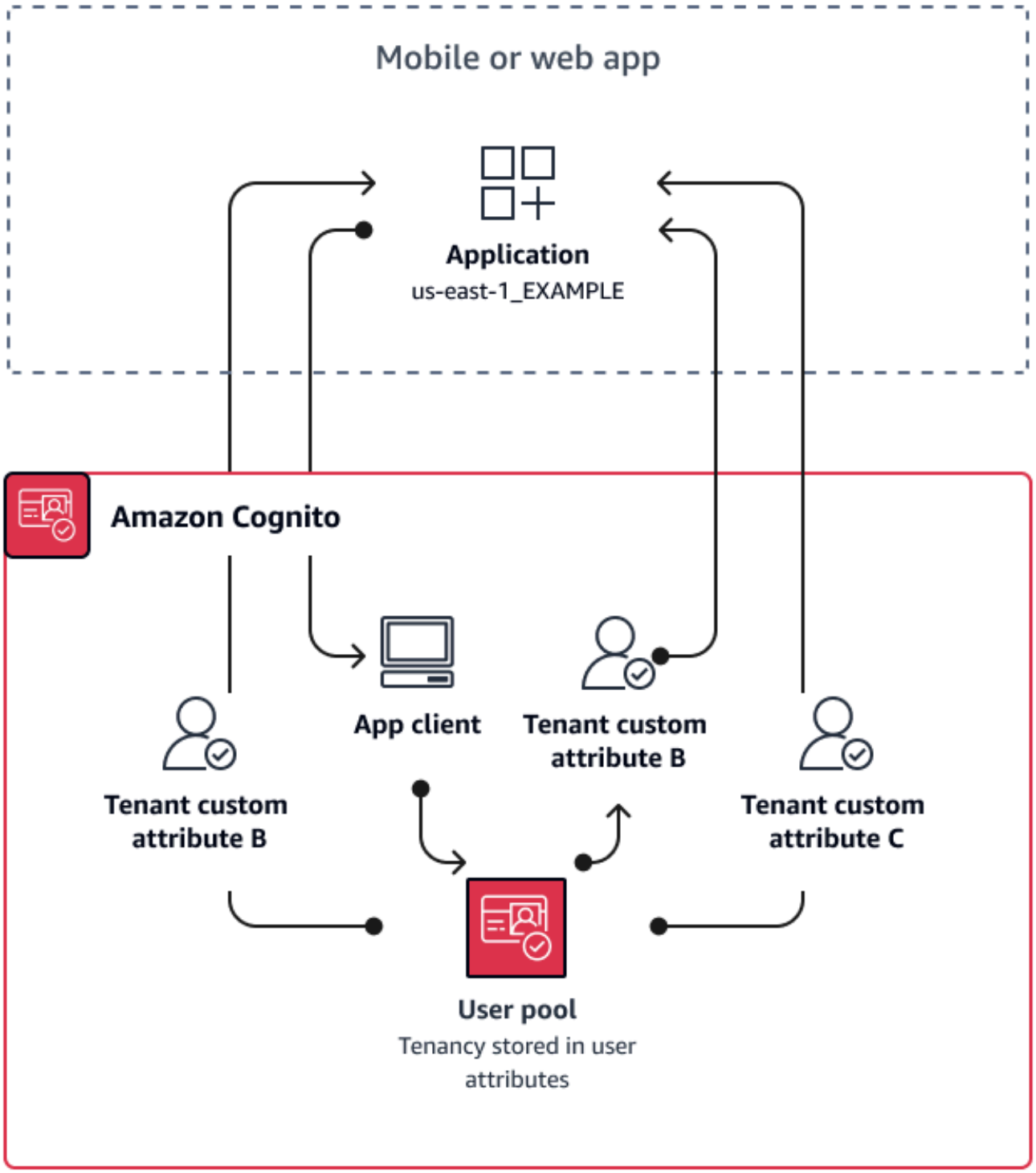
努力程度

僅使用群組成員資格維護多租戶的努力程度很低。不過，若要將使用者集區群組的角色擴充到 IAM 角色選取的內建容量之外，您必須建置應用程式邏輯，以處理使用者權杖中的群組成員資格，並決定要在用戶端中執行的動作。您可以將 Amazon 驗證許可與應用程式整合，以做出用戶端授權決策。群組識別碼目前不會在已驗證的權限 IsAuthorizedWithTokenAPI 作業中處理，但您可以開發自訂程式碼來剖析權杖的內容，包括群組成員資格宣告。

自訂屬性多租戶最佳做法

Amazon Cognito 支援具有您選擇的名稱的自訂屬性。自訂屬性很有用的一個案例是當它們區分共用使用者集區中的使用者租用時。當您為用戶分配屬性值（例如）時`custom:tenantID`，您的應用程序可以相應地為承租人特定資源分配訪問權限。定義租用戶 ID 的自訂屬性對應用程式用戶端而言應該是不可變或唯讀的。

下圖顯示共用應用程式用戶端和使用者集區的租用戶，其中包含使用者集區中指出其所屬租用戶的自訂屬性。



當自訂屬性決定租用時，您可以散發單一應用程式或登入 URL。用戶登錄後，您的應用程序可以處理`custom:tenantID`聲明確定要加載的資產，應用的品牌以及要顯示的功能。如需根據使用者屬性進

行進階存取控制決策，請在 Amazon 驗證許可中將您的使用者集區設定為身分提供者，並根據 ID 或存取權杖的內容產生存取決策。

何時實作自訂屬性多租戶

當租賃是表面水平的。承租人屬性可以有助於品牌和版面配置的結果。當您想要在租用戶之間實現顯著隔離時，自訂屬性並不是最佳選擇。必須在使用者集區或應用程式用戶端層級 (例如 MFA 或託管 UI 品牌) 設定的租用戶之間的任何差異，都需要您以不提供自訂屬性的方式在租用戶之間建立區別。透過身分集區，您甚至可以從使用者的 ID 權杖中的自訂屬性宣告中選擇 IAM 角色。

努力程度

由於自定義屬性多租戶轉移了應用程序上基於租戶的授權決策的責任，因此工作水平往往很高。如果您已經熟悉剖析 OIDC 宣告的用戶端組態，或在 Amazon 驗證許可中，此方法可能需要最低的工作量。

多重租用安全建議

為協助讓您的應用程式更加安全，我們建議：

- 使用 Amazon 驗證許可在您的應用程式中驗證租用。建立原則，在您允許應用程式中的使用者要求之前，先檢查使用者集區、應用程式用戶端、群組或自訂屬性權利。 AWS 使用 Amazon Cognito 使用者集區建立已驗證的許可身分來源。已驗證的權限具有多租戶管理的其他指導方針。
- 請務必使用已驗證的電子郵件地址，根據網域比對來授權使用者存取租用戶。不要信任電子郵件地址和電話號碼，除非您的應用程式驗證它們，或者外部 IdP 提供驗證證明。如需設定上述許可的詳細資訊，請參閱屬性許可和範圍。
- 針對識別承租人的使用者設定檔屬性使用不可變或唯讀的自訂屬性。只有當您在使用者集區中建立使用者或使用者註冊時，才能設定不可變屬性的值。此外，提供應用程式用戶端對屬性的唯讀存取權。
- 在租用戶的外部 IdP 和應用程式用戶端之間使用 1:1 對應，以防止未經授權的跨租用戶存取。已經由外部 IdP 進行身分驗證且具有有效 Amazon Cognito 工作階段 Cookie 的使用者，可以存取信任相同 IdP 的其他租用戶應用程式。
- 在應用程式中實作符合租用戶和授權邏輯時，請限制使用者使其無法修改授權使用者存取租用戶的條件。此外，如果外部 IdP 正用於聯合身分，請限制租用者身分提供者管理員，使其無法變更使用者存取權限。

常見 Amazon Cognito 案例

此主題說明使用 Amazon Cognito 的六個常見案例。

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 Web 和行動應用程式使用者提供註冊和登入的選項。身分集區提供臨時 AWS 登入資料，以授與您的使用者存取其他身 AWS 服務分

使用者集區是在 Amazon Cognito 中的使用者目錄。您的應用程式使用者可以直接透過使用者集區登入，也可以透過第三方身分識別提供者 (IdP) 進行聯合。用戶池管理處理通過 Facebook，谷歌，Amazon 和蘋果以及 OpenID Connect（OIDC）和 SAML 從社交登錄返回的令牌的開銷。 IdPs無論您的使用者直接登入或透過第三方登入，使用者集區的所有成員都有目錄的設定檔，您可以透過軟體開發套件的存取這些設定檔。

透過身分集區，您的使用者可以取得臨時 AWS 登入資料以存取 AWS 服務，例如 Amazon S3 和 DynamoDB。身分集區支援匿名來賓使用者，以及透過第三方進行聯合 IdPs。

主題

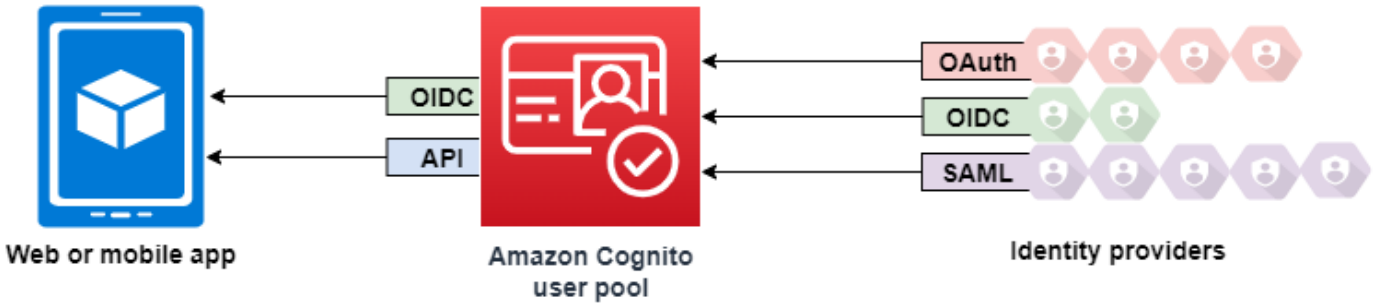
- 以使用者集區進行身分驗證
- 透過使用者集區存取您的伺服器端資源
- 以使用者集區透過 API Gateway 和 Lambda 存取資源
- 使用使用者集區和身分集區存取 AWS 服務
- 以身分集區進行第三方身分驗證以及存取 AWS 服務
- 使用 Amazon Cognito 存取 AWS AppSync 資源

以使用者集區進行身分驗證

您可以讓您的使用者透過使用者集區進行身分驗證。您的應用程式使用者可以直接透過使用者集區登入，也可以透過第三方身分識別提供者 (IdP) 進行聯合。用戶池管理處理通過 Facebook，谷歌，Amazon 和蘋果以及 OpenID Connect（OIDC）和 SAML 從社交登錄返回的令牌的開銷。 IdPs

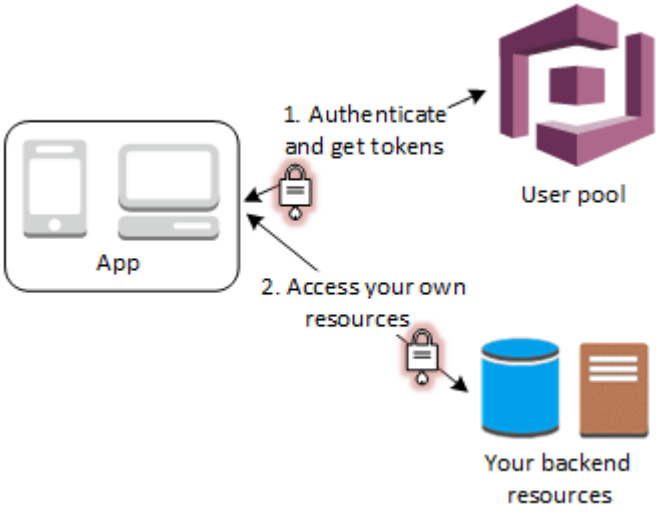
身分驗證成功後，您的 Web 或行動應用程式就會收到來自 Amazon Cognito 的使用者集區權杖。您可以使用這些權杖擷取可讓應用程式存取其他 AWS 服務的 AWS 登入資料，也可以選擇使用這些權杖來控制對伺服器端資源或 Amazon API Gateway 的存取。

如需詳細資訊，請參閱使用者集區身分驗證流程及將權杖用於使用者集區。



透過使用者集區存取您的伺服器端資源

使用者集區登入成功後，您的 Web 或行動應用程式就會收到來自 Amazon Cognito 的使用者集區權杖。您可以使用這些權杖來控制存取您的伺服器端資源。您也可以建立使用者集區群組來管理許可，以及呈現不同類型的使用者。如需有關使用群組來控制存取資源的詳細資訊，請參閱新增群組至使用者集區。



在使用者集區的網域設定完成後，Amazon Cognito 會佈建一個託管 Web UI，供您用來將註冊和登入頁面新增至您的應用程式。使用此 OAuth 2.0 基礎，您可以建立自己的資源伺服器，讓您的使用者能夠存取受保護的資源。如需詳細資訊，請參閱 使用資源伺服器進行範圍、M2M 和 API 授權。

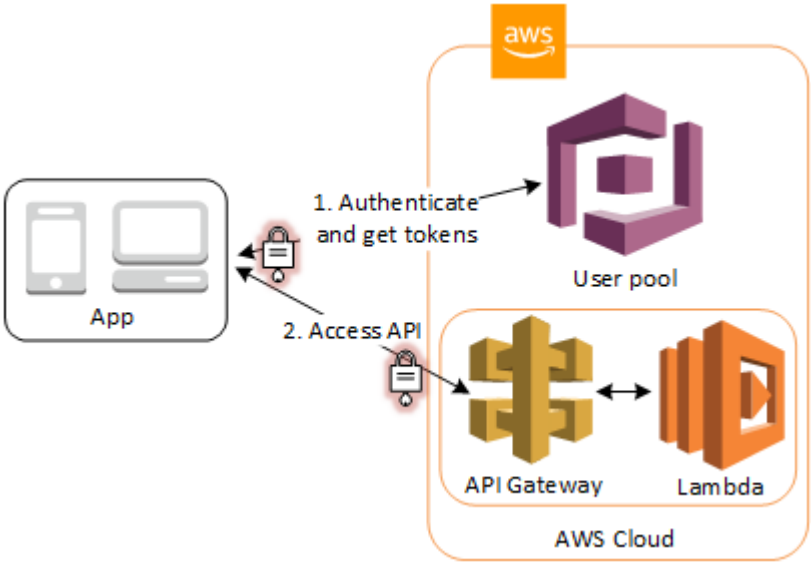
如需有關使用者集區身分驗證的詳細資訊，請參閱 使用者集區身分驗證流程 和 將權杖用於使用者集區。

以使用者集區透過 API Gateway 和 Lambda 存取資源

您可以讓使用者透過 API Gateway 存取您的 API。API Gateway 從成功的使用者集區身分驗證來驗證權杖，並使用它們來授與使用者存取資源，包括 Lambda 函數或您自己的 API。

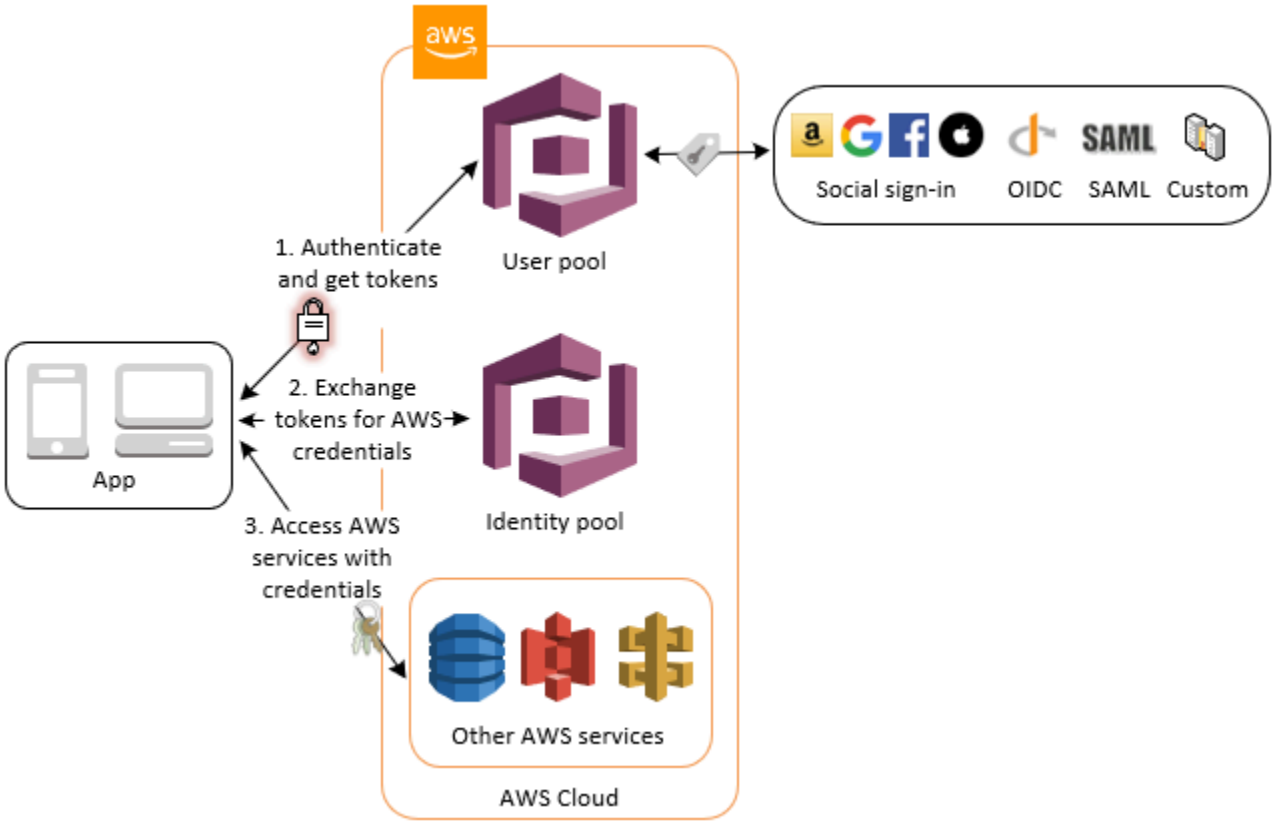
您可以使用使用者集區中的群組，藉由將群組成員資格對應至 IAM 角色來使用 API Gateway 控制許可。使用者隸屬的群組，將會包含在使用者集區於應用程式使用者登入時所提供的 ID 權杖中。如欲了解使用者集區群組的詳細資訊，請參閱新增群組至使用者集區。

您可以提交您的使用者集區權杖，包含 Amazon Cognito 授權程式 Lambda 函數對 API Gateway 提出驗證的請求。如需 API Gateway 的詳細資訊，請參閱搭配 Amazon Cognito 使用者集區使用 API Gateway。



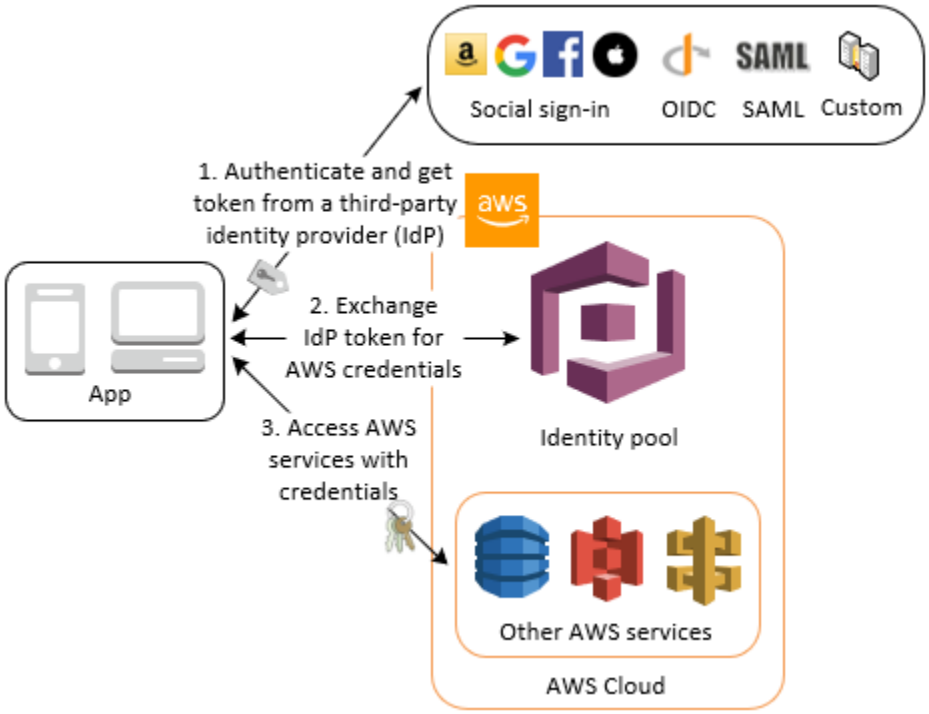
使用使用者集區和身分集區存取 AWS 服務

使用者集區身分驗證成功後，您的應用程式將會收到來自 Amazon Cognito 的使用者集區權杖。您可以將它們交換為具有身分集區的其他 AWS 服務的臨時訪問權限。如需詳細資訊，請參閱 登入後 AWS 服務 使用身分集區存取 及 開始使用 Amazon Cognito 身分集區。



以身分集區進行第三方身分驗證以及存取 AWS 服務

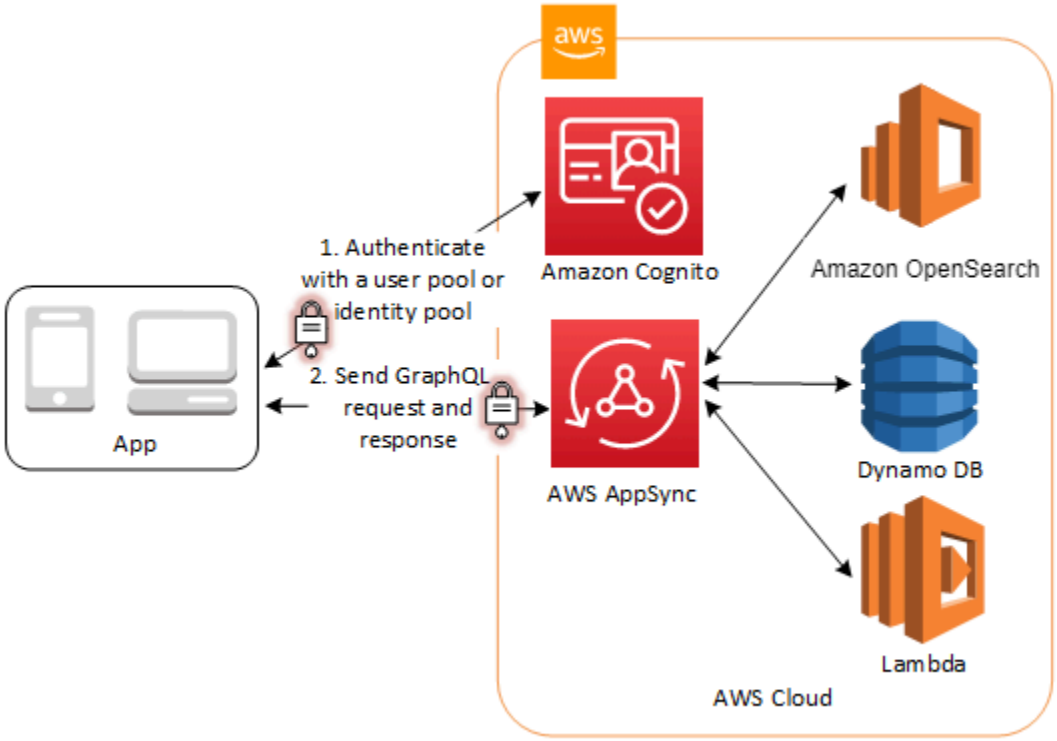
您可以透過身分識別集區讓使用者存取 AWS 服務。身分集區需使用者的 IdP 字符，且須由第三方身分提供者經身分驗證 (若為匿名訪客則無)。作為交換，身分集區會授與您可用來存取其他 AWS 服務的臨時 AWS 登入資料。如需詳細資訊，請參閱 開始使用 Amazon Cognito 身分集區。



使用 Amazon Cognito 存取 AWS AppSync 資源

您可以透過成功的 Amazon Cognito 使用者集區身分驗證，授與使用者使用權杖存取 AWS AppSync 資源。如需詳細資訊，請參閱《AWS AppSync 開發人員指南》中的 AMAZON_COGNITO_USER_POOLS 授權。

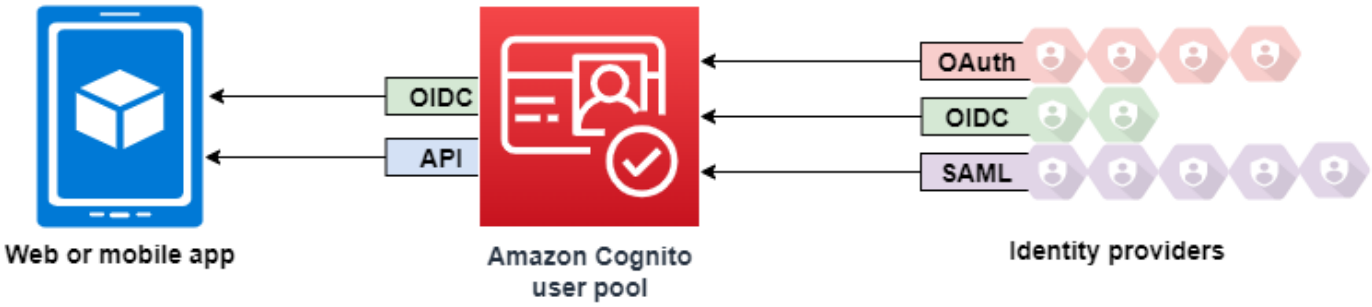
您也可以使用從身分集區收到的 IAM 登入資料，簽署要求至 AWS AppSync GraphQL API。請見 AWS_IAM 授權。


aws
Amazon Cognito
1. Authenticate with a user pool or identity pool
2. Send GraphQL request and response
App
AWS AppSync
Amazon OpenSearch
Dynamo DB
Lambda
AWS Cloud

Amazon Cognito 使用者集區

Amazon Cognito 使用者集區是用於網路和行動應用程式身份驗證和授權的使用者目錄。從您的應用程式角度來看，Amazon Cognito 使用者集區是 OpenID Connect (OIDC) 身分提供者 (IdP)。使用者集區為安全性、聯合身分、應用程式整合和自訂使用者體驗新增額外的功能層。

例如，您可以驗證使用者的工作階段來自信任的來源。您可以將 Amazon Cognito 目錄與外部身分提供者結合使用。使用您偏好的 AWS SDK，您可以選擇最適合您應用程式的 API 授權模型。此外，您還可以新增 AWS Lambda 函數，修改或徹底改造 Amazon Cognito 的預設行為。



主題

- 使用 Amazon Cognito 使用者集區安全性功能

功能

Amazon Cognito 使用者集區具有下列功能。

註冊

Amazon Cognito 使用者集區具有使用者驅動、管理員驅動和程式設計方法，將使用者設定檔新增至您的使用者集區。Amazon Cognito 使用者集區支援下列註冊模型。您可以在應用程式中任意組合使用模型。

> **Important**
>
> 如果您在使用者集區中啟用使用者註冊，則網際網路上的任何人都可以註冊帳戶並登入您的應用程式。除非您想要開放您的應用程式供公開註冊，否則請勿在使用者集區中啟用自助註冊。若要變更此設定，請在使用者集區主控台的 [註冊] 體驗索引標籤中更新自助註冊，或更新 CreateUserPool或 UpdateUserPoolAPI 要求 AllowAdminCreateUserOnly中的值。
> 如需可在使用者集區中設定之安全功能的詳細資訊，請參閱 使用 Amazon Cognito 使用者集區安全性功能。

1. 您的使用者可以在您的應用程式中輸入他們的資訊，並建立您使用者集區原生的使用者設定檔。您可以呼叫 API 註冊作業，在您的使用者集區中註冊使用者。您可以向任何人打開這些註冊操作，也可以使用客戶端密鑰或 AWS 憑據對其進行授權。
2. 您可以將使用者重新導向至第三方 IdP，他們可以授權將其資訊傳遞給 Amazon Cognito。Amazon Cognito 會將 OIDC ID 權杖、OAuth 2.0 `userInfo` 資料和 SAML 2.0 宣告處理到使用者集區中的使用者設定檔中。您可以根據屬性映射規則控制想要 Amazon Cognito 接收的屬性。
3. 您可以跳過公開或聯合註冊，並根據自己的資料來源和結構描述建立使用者。直接在 Amazon Cognito 主控台或 API 中新增使用者。從 CSV 檔案匯入使用者。運行一個在現有目錄中查找新用戶的 just-in-time AWS Lambda 函數，並從現有數據填充他們的用戶配置文件。

使用者註冊後，您可以將他們新增至 Amazon Cognito 列於存取和 ID 權杖清單中的群組。當您將 ID 權杖傳遞至身分池時，您也可以將使用者集區群組連結至 IAM 角色。

相關主題

- 管理使用者集區中的使用者
- 使用 Amazon Cognito 使用者集區 API 和使用者集區端點
- 使 AWS 用軟體開發套件的 Amazon Cognito 身分供應商程式碼範例

登入

Amazon Cognito 可以是您應用程式的獨立使用者目錄和身分提供者 (IdP)。您的使用者可以使用由 Amazon Cognito 託管的使用者介面登入，或透過 Amazon Cognito 使用者集區 API 使用您自己的使用者介面登入。前端終端自訂使用者介面背後的應用程式層，可以使用多種方法中的任一種來請求後端，以確認合法請求。

若要使用外部目錄登入使用者 (選擇性地與 Amazon Cognito 內建的使用者目錄結合)，您可以新增下列整合功能。

1. 使用 OAuth 2.0 社交登入，以登入和匯入消費者使用資料。Amazon Cognito 支援透過 OAuth 2.0 登入 Google、Facebook、Amazon 和Apple。
2. 使用 SAML 和 OIDC 登入，以登入和匯入企業使用者資料。您也可以設定 Amazon Cognito 接受任何 SAML 或 OpenID Connect (OIDC) 身分提供者 (IdP) 的宣告。
3. 將外部使用者設定檔連結至原生使用者設定檔。連結的使用者可以使用第三方使用者身分登入，並接收您指派給內建目錄中使用者的存取權。

相關主題

- 透過第三方新增使用者集區登入
- 將聯合身分使用者連結至現有的使用者描述檔

M achine-to-machine 授權

有些工作階段不是 human-to-machine 互動。您可能需要一個服務帳戶，該帳戶可以通過自動化流程對 API 請求授權。要使用 OAuth 2.0 範圍生成 machine-to-machine 授權的訪問令牌，您可以添加生成客戶端憑據授權的應用程序客戶端。

相關主題

- 使用資源伺服器進行範圍、M2M 和 API 授權

託管 UI

當您不想建立使用者介面時，可以向使用者顯示自訂的 Amazon Cognito 託管 UI。託管 UI 是一套 Web 頁面，用於註冊、登入、多重要素驗證 (MFA) 和密碼重設。您可以將託管 UI 新增至您現有的網域，或在 AWS 子網域中使用前置字元識別碼。

相關主題

- 設定和使用 Amazon Cognito 託管 UI 和聯合端點
- 設定使用者集區網域

安全

您的本機使用者可以透過 SMS 訊息，或產生多重要素驗證 (MFA) 代碼的應用程式，提供額外的驗證因素。您可以建立機制，以在應用程式中設定和處理 MFA，也可以讓託管 UI 進行管理。當您的使用者從受信任的裝置登入時，Amazon Cognito 使用者集區可以略過 MFA。

如果您不想一開始就要求用戶進行 MFA，您可以有條件地要求。透過進階安全性功能，Amazon Cognito 可偵測潛在的惡意活動，並要求您的使用者設定 MFA 或封鎖登入。

如果使用者集區的網路流量可能是惡意的，您可以對其進行監控，並對 AWS WAF Web ACL 採取處理行動。

相關主題

- 將 MFA 新增到使用者集區
- 將進階安全性新增到使用者集區
- 建立 AWS WAF Web ACL 與使用者集區的關聯

自訂使用者體驗

在使用者註冊、登入或設定檔更新的大部分階段，您都可以自訂 Amazon Cognito 處理請求的方式。使用 Lambda 觸發程序，您可以根據自訂條件修改 ID 權杖或拒絕註冊請求。您可以建立自己的自訂身分驗證流程。

您可以上傳自定的 CSS 和標誌，讓託管 UI 為您的使用者提供熟悉的外觀和感覺。

相關主題

- 使用 Lambda 觸發程序來自訂使用者集區工作流程
- 自訂身分驗證挑戰 Lambda 觸發程序
- 自訂內建的註冊與登入網頁

監控與分析

Amazon Cognito 使用者集區會將 API 請求 (包括對託管 UI 的請求) 記錄到 AWS CloudTrail。您可以在 Service Quotas 主控台中檢閱 Amazon CloudWatch 日誌中的效能指標、 CloudWatch 使用 Lambda 觸發器將自訂日誌推送至，以及監控 API 請求量。

您也可以將 API 請求中的裝置和工作階段資料記錄到 Amazon Pinpoint 廣告活動。使用 Amazon Pinpoint，您可以根據對使用者活動的分析，從應用程式傳送推播通知。

相關主題

- 使用記錄 Amazon Cognito API 呼叫 AWS CloudTrail
- 追蹤以及 Service Quotas 中的配額 CloudWatch 和使用情況
- 使用 Amazon Pinpoint 分析搭配 Amazon Cognito 使用者集區

Amazon Cognito 身分池整合

Amazon Cognito 的另一半是身分池。身分集區提供的登入資料可授權和監控來自使用者的 API 請求 (例如 Amazon DynamoDB 或 Amazon S3)。 AWS 服務您可以建立以身分為基礎的存取原則，根據您在使用者集區中對使用者進行分類的方式來保護您的資料。身分池也可以接受來自各種身分提供者的權杖和 SAML 2.0 宣告，而不受使用者集區驗證影響。

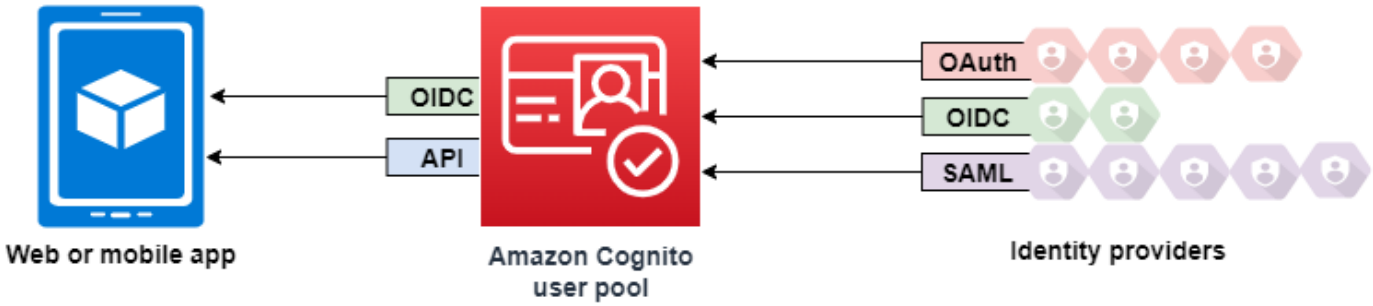
相關主題

- 登入後 AWS 服務 使用身分集區存取
- Amazon Cognito 身分集區

以使用者集區進行身分驗證

您的應用程式使用者可以直接透過使用者集區登入，也可以透過第三方身分識別提供者 (IdP) 進行聯合。用戶池管理處理通過 Facebook，谷歌，Amazon 和蘋果以及 OpenID Connect（OIDC）和 SAML 從社交登錄返回的令牌的開銷。 IdPs

身分驗證成功之後，Amazon Cognito 會將使用者集區權杖傳回至您的應用程式。您可以使用這些權杖，授予使用者存取您自己的伺服器端資源或 Amazon API Gateway 的權限。或者，您可以將它們交換為 AWS 憑據以訪問其他 AWS 服務。



Web 和行動應用程式的使用者集區權杖處理和管理任務，就是在用戶端上透過 Amazon Cognito 軟體開發套件所提供。同樣地，如果這時存在有效 (未過期) 重新整理權杖，而且 ID 權杖和存取權杖的有效性至少還剩餘 5 分鐘，Mobile SDK for iOS 和 Mobile SDK for Android 就會自動重新整理您的 ID 權杖和存取權杖。如需有關開發套件的資訊，以及 Android 和 iOS 的範例程式碼 JavaScript，請參閱 Amazon Cognito 使用者集區開發套件。

當您的應用程式使用者登入成功之後，Amazon Cognito 就會為該驗證成功的使用者建立工作階段，並且傳回 ID、存取權限和重新整理權杖。

JavaScript

```
// Amazon Cognito creates a session which includes the id, access, and refresh
 tokens of an authenticated user.

var authenticationData = {
        Username : 'username',
        Password : 'password',
    };
    var authenticationDetails = new
 AmazonCognitoIdentity.AuthenticationDetails(authenticationData);
    var poolData = { UserPoolId : 'us-east-1_ExaMPle',
        ClientId : '1example23456789'
```

```
    };
    var userPool = new AmazonCognitoIdentity.CognitoUserPool(poolData);
    var userData = {
        Username : 'username',
        Pool : userPool
    };
    var cognitoUser = new AmazonCognitoIdentity.CognitoUser(userData);
    cognitoUser.authenticateUser(authenticationDetails, {
        onSuccess: function (result) {
            var accessToken = result.getAccessToken().getJwtToken();

            /* Use the idToken for Logins Map when Federating User Pools with
 identity pools or when passing through an Authorization Header to an API Gateway
 Authorizer */
            var idToken = result.idToken.jwtToken;
        },

        onFailure: function(err) {
            alert(err);
        },

});
```

Android

```
// Session is an object of the type CognitoUserSession, and includes the id, access,
 and refresh tokens for a user.

String idToken = session.getIdToken().getJWTToken();
String accessToken = session.getAccessToken().getJWT();
```

iOS - swift

```
// AWSCognitoIdentityUserSession includes id, access, and refresh tokens for a user.

- (AWSTask<AWSCognitoIdentityUserSession *> *)getSession;
```

iOS - objective-C

```
// AWSCognitoIdentityUserSession includes the id, access, and refresh tokens for a
 user.
```

```
[[user getSession:@"username" password:@"password" validationData:nil scopes:nil]
 continueWithSuccessBlock:^id _Nullable(AWSTask<AWSCognitoIdentityUserSession *> *
 _Nonnull task) {
    // success, task.result has user session
    return nil;
}];
```

主題

- 使用者集區身分驗證流程
- 使用者集區應用程式用戶端
- 在使用者集區中使用使用者裝置

使用者集區身分驗證流程

Amazon Cognito 包含幾種對您的使用者進行身分驗證的方法。所有使用者集區 (無論您是否擁有網域) 都可以在使用者集區 API 中對使用者進行身分驗證。如果您將網域新增至使用者集區，則可以使用使用者集區端點。使用者集區 API 支援各種授權模型和 API 請求的請求流程。

為驗證使用者的身分，Amazon Cognito 支援身分驗證流程納入密碼以外的新挑戰類型。Amazon Cognito 身分驗證通常會要求您按以下順序實作兩個 API 操作：

Public authentication

1. InitiateAuth
2. RespondToAuthChallenge

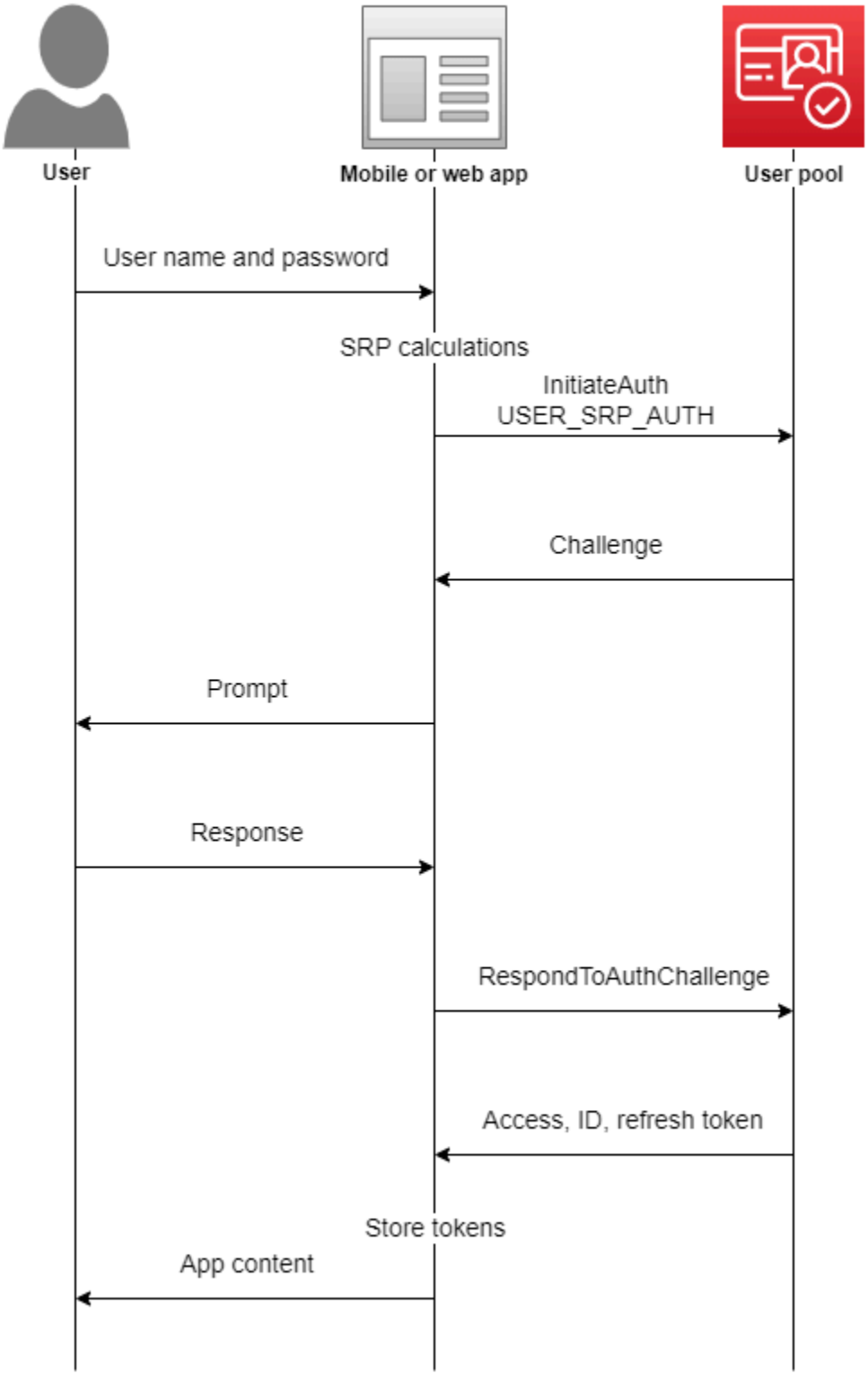
`InitiateAuth` 和 `RespondToAuthChallenge` 是未驗證的 API，可用於用戶端公有應用程式用戶端。

Server-side authentication

1. AdminInitiateAuth
2. AdminRespondToAuthChallenge

`AdminInitiateAuth` 和 `AdminRespondToAuthChallenge` 需要 IAM 憑證，且適用於伺服器端機密應用程式用戶端。

使用者身分驗證會透過回答連續挑戰，直到驗證失敗或者 Amazon Cognito 發出權杖給使用者。您可以使用 Amazon Cognito 在包含不同挑戰的程序中重複這些步驟，以支援任何自訂的身分驗證流程。



通常，您的應用程式會產生從使用者收集資訊的提示，並將該資訊以 API 請求提交給 Amazon Cognito。考慮使用者集區中的 `InitiateAuth` 流程，您已在其中為使用者設定多重要素驗證 (MFA)。

1. 您的應用程式會提示您的使用者輸入其使用者名稱和密碼。
2. 您可將使用者名稱和密碼作為參數包含於 `InitiateAuth` 中。
3. Amazon Cognito 會傳回一個 `SMS_MFA` 挑戰和工作階段識別符。
4. 您的應用程式會提示使用者從手機輸入 MFA 代碼。
5. 您將該代碼和工作階段識別符包含於 `RespondToAuthChallenge` 請求中。

根據您使用者集區的功能，您最終可能會在應用程式從 Amazon Cognito 擷取字符之前回應對 `InitiateAuth` 的多個挑戰。Amazon Cognito 會在每個請求的回應中包含一個工作階段字串。如要將您的 API 請求合併至身分驗證流程中，請在每個後續請求中包含來自上一個請求回應的工作階段字串。依預設，您的使用者在工作階段字串到期前，有三分鐘的時間完成每項挑戰。如要調整此期間，請變更您的應用程式用戶端身分驗證流程工作階段持續時間。下列程序說明如何在應用程式用戶端設定中變更此設定。

> **Note**
>
> 身分驗證流程工作階段持續時間 設定適用於 Amazon Cognito 使用者集區 API 的身份驗證。Amazon Cognito 託管的使用者介面將多因素身份驗證的工作階段持續時間設定為 3 分鐘，密碼重設代碼則設定為 8 分鐘。

Amazon Cognito console

設定應用程式用戶端驗證流程工作階段持續時間 (AWS Management Console)

1. 在使用者集區的 App integration (應用程式整合) 索引標籤，選取 App clients and analytics (應用程式用戶端和分析) 容器的應用程式用戶端名稱。
2. 在 應用程式用戶端資訊 容器中選擇 編輯。
3. 將 Authentication flow session duration (驗證流程工作階段持續時間) 的值變更為您要的 SMS MFA 代碼的有效期限 (以分鐘為單位)。這也會改變任何使用者在應用程式用戶端中完成任何驗證挑戰的時間長度。
4. 選擇儲存變更。

Amazon Cognito API

設定應用程式用戶端驗證流程工作階段持續時間 (Amazon Cognito API)

1. 使用 `DescribeUserPoolClient` 請求中現有的使用者集區設定來準備 `UpdateUserPoolClient` 請求。您的 `UpdateUserPoolClient` 請求必須包含所有現有的應用程式用戶端屬性。
2. 將 `AuthSessionValidity` 的值變更為您要的 SMS MFA 代碼的有效期限 (以分鐘為單位)。這也會改變任何使用者在應用程式用戶端中完成任何驗證挑戰的時間長度。

如需應用程式用戶端的詳細資訊，請參閱 使用者集區應用程式用戶端。

您可以使用 AWS Lambda 觸發器來自訂使用者驗證的方式。這些觸發會發出並驗證自身的挑戰做為驗證流程的一部分。

您也可以針對安全的後端伺服器使用系統管理員驗證流程。您可以利用使用者遷移身分驗證流程讓使用者遷移成為可能，而無需使用者重設密碼。

針對失敗登入嘗試的 Amazon Cognito 鎖定行為

使用密碼嘗試進行未驗證或經 IAM 驗證的登入失敗五次後，Amazon Cognito 會將您的使用者鎖定一秒鐘。然後，鎖定持續時間會在每增加一次失敗嘗試後加倍，最多可達約 15 分鐘。在鎖定期間進行的嘗試會產生 `Password attempts exceeded` 例外狀況，並且不會影響後續鎖定期間的持續時間。對於嘗試登入失敗的累計次數 n (不包括 `Password attempts exceeded` 例外狀況)，Amazon Cognito 會將您的使用者鎖定 2^(n-5) 秒。若要將鎖定重設為 n=0 初始狀態，您的使用者必須在鎖定期間過後成功登入，或是在鎖定後連續 15 分鐘內不得進行任何登入嘗試。此行為可能會有所變更。此行為不適用於自訂挑戰，除非該挑戰也執行以密碼為基礎的身份驗證。

主題

- 用戶端身分驗證流程
- 伺服器端身分驗證流程
- 自訂身分驗證流程
- 內建身分驗證流程與挑戰
- 自訂身分驗證流程與挑戰
- 在自訂身分驗證流程中使用 SRP 密碼驗證
- 管理員身分驗證流程
- 使用者遷移身分驗證流程

用戶端身分驗證流程

以下程序適用於您使用 AWS Amplify 或 AWS SDK 所建立的使用者用戶端應用程式。

1. 使用者在應用程式中輸入使用者名稱和密碼。
2. 應用程式會以使用者的使用者名稱和安全遠端密碼 (SRP) 詳細資訊呼叫 `InitiateAuth` 操作。

   此 API 操作會傳回驗證參數。

   > **Note**
   >
   > 此應用程式使用 AWS SDK 內建的 Amazon Cognito SRP 功能產生 SRP 詳細資訊。

3. 應用程式會呼叫 `RespondToAuthChallenge` 操作。如果呼叫成功，Amazon Cognito 會傳回使用者的權杖，身分驗證流程便已完成。

   如果 Amazon Cognito 需要另一個挑戰，對 `RespondToAuthChallenge` 的呼叫就不會傳回任何權杖。相反地，呼叫會傳回工作階段。
4. 如果 `RespondToAuthChallenge` 傳回工作階段，應用程式會再次呼叫 `RespondToAuthChallenge`，這次會使用工作階段和挑戰回應 (例如，MFA 程式碼)。

伺服器端身分驗證流程

如果您沒有使用者應用程式，而是使用 Java、Ruby 或 Node.js 安全後端或伺服器端應用程式，您可以使用適用於 Amazon Cognito 使用者集區的已驗證伺服器端 API。

如果是伺服器端應用程式，使用者集區身分驗證類似於用戶端應用程式的身分驗證，但以下除外：

- 伺服器端應用程式會呼叫 `AdminInitiateAuth` API 操作 (而非 `InitiateAuth`)。此作業需要具有包含`cognito-idp:AdminInitiateAuth`和權限的 AWS 認證`cognito-idp:AdminRespondToAuthChallenge`。此操作會傳回必要的身分驗證參數。
- 伺服器端應用程式取得身分驗證參數後，就會呼叫 `AdminRespondToAuthChallenge` API 操作 (而不是 `RespondToAuthChallenge`)。只有當您提供 AWS 認證時，`AdminRespondToAuthChallenge`API 作業才會成功。

如需有關使用登 AWS 入資料簽署 Amazon Cognito API 請求的詳細資訊，請參閱AWS 一般參考中的簽名版本 4 簽署程序。

AdminInitiateAuth和 AdminRespondToAuthChallenge API 操作無法接受 username-and-password 用戶憑據進行管理員登錄，除非您通過以下方式之一明確允許他們這樣做：

- 當您呼叫 CreateUserPoolClient 或 UpdateUserPoolClient 時，在 ExplicitAuthFlow 參數中包含 ALLOW_ADMIN_USER_PASSWORD_AUTH (先前稱為 ADMIN_NO_SRP_AUTH)。
- 將 ALLOW_ADMIN_USER_PASSWORD_AUTH 新增至您的應用程式用戶端 身份驗證流程 清單。在 App clients and analytics (應用程式用戶端與分析) 下，您使用者集區中的 App integration (應用程式整合) 索引標籤上設定應用程式用戶端。如需詳細資訊，請參閱 使用者集區應用程式用戶端。

自訂身分驗證流程

Amazon Cognito 使用者集區也讓您能夠使用自訂身份驗證流程，協助您使用觸發器建立挑戰/回應型身份驗證模型。 AWS Lambda

> **Note**
>
> 您無法將進階安全性功能用於遭到入侵的憑證，以及具有自訂驗證流程的調適性驗證。如需詳細資訊，請參閱 將進階安全性新增到使用者集區。

自訂身分驗證流程可建立自訂挑戰和回應週期，以滿足不同需求。流程一開始會呼叫 InitiateAuth API 操作，指出所要使用的身分驗證類型，並提供任何初始身分驗證參數。Amazon Cognito 會以下列其中一種資訊類型回應 InitiateAuth 呼叫：

- 對使用者的挑戰，伴隨工作階段和參數。
- 錯誤 (如果使用者驗證失敗)。
- ID、存取和重新整理權杖 (如果 InitiateAuth 呼叫中提供的參數足以讓使用者登入)。(通常，使用者或應用程式必須先回答挑戰，但您的自訂程式碼必須確定這一點)。

如果 Amazon Cognito 以挑戰回應 InitiateAuth 呼叫，應用程式將會收集更多輸入，並呼叫 RespondToAuthChallenge 操作。此呼叫提供挑戰回應，並將其傳回工作階段。Amazon Cognito 回應 RespondToAuthChallenge 呼叫的方式類似於回應 InitiateAuth 呼叫。如果使用者已登入，Amazon Cognito 會提供權杖，或者如果使用者未登入，Amazon Cognito 會提供另一個挑戰，或者錯誤。如果 Amazon Cognito 傳回另一項挑戰，順序將會重複，而應用程式將會呼叫 RespondToAuthChallenge，直到使用者成功登入或傳回錯誤。如需 InitiateAuth 和 RespondToAuthChallenge API 操作的詳細資訊，請參閱 API 文件。

內建身分驗證流程與挑戰

Amazon Cognito 含有一些內建的 AuthFlow 和 ChallengeName 值，可讓標準身分驗證流程透過安全遠端密碼 (SRP) 通訊協定來驗證使用者名稱和密碼。 AWS 開發套件已透過 Amazon Cognito 內建對這些流程的支援。

流程一開始會傳送 USER_SRP_AUTH 作為 AuthFlow 至 InitiateAuth。您也可以傳送 AuthParameters 中的 USERNAME 和 SRP_A 值。如果 InitiateAuth 呼叫成功，則回應會包含 PASSWORD_VERIFIER 並將其作為在挑戰參數中的 ChallengeName 和 SRP_B。然後，應用程式會呼叫 RespondToAuthChallenge 搭配 PASSWORD_VERIFIER ChallengeName 和 ChallengeResponses 中的必要參數。如果對 RespondToAuthChallenge 的呼叫成功且使用者登入，Amazon Cognito 就會發出權杖。如果您為使用者啟用了多重要素驗證 (MFA)，Amazon Cognito 就會傳回 SMS_MFA 的 ChallengeName。此應用程式可透過另一個對 RespondToAuthChallenge 的呼叫來提供必要的程式碼。

自訂身分驗證流程與挑戰

應用程式可以呼叫 InitiateAuth，並將CUSTOM_AUTH 做為 Authflow，以啟動自訂身分驗證流程。使用自訂身分驗證流程時，三個 Lambda 觸發程序會控制挑戰和回應的驗證。

- DefineAuthChallenge Lambda 觸發程序使用以前的挑戰和回應的工作階段陣列作為輸入。然後，它產生下一個挑戰名稱和布林值，指出使用者是否通過身分驗證 (並可被授予權杖)。這個 Lambda 觸發器是一種狀態機器，可控制使用者通過挑戰的路徑。
- CreateAuthChallenge Lambda 觸發程序會以挑戰名稱作為輸入，並產生挑戰和參數來評估回應。當 DefineAuthChallenge 傳回 CUSTOM_CHALLENGE 作為下一個挑戰，身分驗證流程將會呼叫 CreateAuthChallenge。CreateAuthChallenge Lambda 觸發程序會在挑戰中繼資料參數中傳遞下一個挑戰類型。
- VerifyAuthChallengeResponse Lambda 函數會評估回應，並傳回布林值，指出回應是否有效。

自訂驗證流程也可以結合內建挑戰，例如 SRP 密碼驗證和透過 SMS 的 MFA。它可以使用自訂挑戰，如 CAPTCHA 或秘密提示問題。

在自訂身分驗證流程中使用 SRP 密碼驗證

如果您想要將 SRP 包含在自訂身分驗證流程中，就需要從 SRP 開始。

- 若要在自訂流程中起始 SRP 密碼驗證，應用程式會呼叫 InitiateAuth 並以 CUSTOM_AUTH 為 Authflow。在 AuthParameters 地圖中，來自您的應用程式的請求包含 SRP_A: (SRP A 值) 和 CHALLENGE_NAME: SRP_A。
- CUSTOM_AUTH 流程會以 challengeName: SRP_A 和 challengeResult: true 的初始工作階段叫用 DefineAuthChallenge Lambda 觸發程序。您的 Lambda 函數以 challengeName: PASSWORD_VERIFIER、issueTokens: false 和 failAuthentication: false 回應。
- 應用程式接下來需要用 challengeName: PASSWORD_VERIFIER 呼叫 RespondToAuthChallenge 和 challengeResponses 映射中 SRP 所需的其他參數。
- 如果 Amazon Cognito 驗證密碼，RespondToAuthChallenge 會以 challengeName: PASSWORD_VERIFIER 和 challengeResult: true 的第二個工作階段叫用 DefineAuthChallenge Lambda 觸發程序。到時 DefineAuthChallenge Lambda 觸發程序可以回應 challengeName: CUSTOM_CHALLENGE 以開始自訂挑戰。
- 如果為使用者啟用 MFA，則 Amazon Cognito 驗證密碼後，便會要求使用者設定或使用 MFA 登入。

> **Note**
>
> Amazon Cognito 託管的登入網頁無法啟用 自訂身分驗證挑戰 Lambda 觸發程序。

如需 Lambda 觸發器的詳細資訊，包括範本程式碼，請參閱 使用 Lambda 觸發程序來自訂使用者集區工作流程。

管理員身分驗證流程

身分驗證的最佳實務是使用自訂身分驗證流程所述的 API 操作搭配 SRP 進行密碼驗證。AWS SDK 使用這種方法，這種方法可以幫助他們使用 SRP。不過，如果您想要避開 SRP 計算，還有另一組可用於安全後端伺服器的管理 API 操作。對於這些後端管理員實作，會使用 AdminInitiateAuth 來代替 InitiateAuth，並使用 AdminRespondToAuthChallenge 來代替 RespondToAuthChallenge。由於您可以用純文字提交密碼，因此在使用這些操作時，無需執行 SRP 計算。請見此處範例：

```
AdminInitiateAuth Request {
    "AuthFlow":"ADMIN_USER_PASSWORD_AUTH",
    "AuthParameters":{
        "USERNAME":"<username>",
        "PASSWORD":"<password>"
        },
```

```
    "ClientId":"<clientId>",
    "UserPoolId":"<userPoolId>"
}
```

這些管理員身分驗證操作需要開發人員登入資料，並使用 AWS Signature Version 4 (SigV4) 簽署程序。這些操作可在標準 AWS 開發套件中使用，其中包含 Node.js，可方便用於 Lambda 函數。若要使用這些操作並讓它們接受純文字密碼，您必須在主控台為應用程式啟用它們。或者，您可以在呼叫 CreateUserPoolClient 或 UpdateUserPoolClient 時對 ExplicitAuthFlow 參數傳遞 ADMIN_USER_PASSWORD_AUTH。InitiateAuth 和 RespondToAuthChallenge 操作不接受 ADMIN_USER_PASSWORD_AUTH AuthFlow。

在 AdminInitiateAuth 回應 ChallengeParameters 中，USER_ID_FOR_SRP 屬性 (如果存在) 將會包含使用者的實際使用者名稱，而不是別名 (例如電子郵件地址或電話號碼)。在呼叫 AdminRespondToAuthChallenge 期間，在 ChallengeResponses 中，您必須在 USERNAME 參數中傳遞此使用者名稱。

> **Note**
>
> 由於後端管理實作使用管理員身分驗證流程，所以此流程不支援裝置追蹤。當您開啟裝置追蹤功能時，管理員身分驗證會成功，但是用來重新整理存取權杖的任何呼叫都會失敗。

使用者遷移身分驗證流程

使用者遷移 Lambda 觸發程序有助於將使用者從舊式使用者管理系統遷移到使用者集區。如果您選擇 USER_PASSWORD_AUTH 身分驗證流程，使用者就不必使用者遷移期間重設密碼。此流程會在驗證期間透過加密的 SSL 連線，將您的使用者密碼傳送至服務。

當您已遷移所有使用者時，請將流程切換至更安全的 SRP 流程。SRP 流程不會透過網路傳送任何密碼。

若要進一步了解 Lambda 觸發程序，請參閱使用 Lambda 觸發程序來自訂使用者集區工作流程。

如需使用 Lambda 觸發程序來遷移使用者的詳細資訊，請參閱透過使用者遷移 Lambda 觸發程序將使用者匯入到使用者集區。

使用者集區應用程式用戶端

使用者集區應用程式用戶端是使用者集區內的組態，它會與透過 Amazon Cognito 進行驗證的一個行動或 Web 應用程式互動。應用程式用戶端可以呼叫已驗證和未驗證的 API 操作，以及讀取或修改您的使

用者屬性的一部分或全部。您的應用程式必須在註冊、登入及處理忘記密碼的操作過程中，對應用程式用戶端表明自己的身分。這些 API 請求必須包含具有應用程式用戶端 ID 的自我識別，以及具有選用用戶端密碼的授權。您必須保護任何應用程式用戶端 ID 或密碼的安全，如此只有經授權的用戶端應用程式可以呼叫這些未驗證的操作。此外，如果您將應用程序配置為使用 AWS 憑據簽署已驗證的 API 請求，則必須保護憑據以防止用戶檢查。

您可以為使用者集區建立多個應用程式。應用程式用戶端可能會連結至應用程式的程式碼平台，或是使用者集區中的另一個租用戶。例如，您可能為伺服器端應用程式建立一個應用程式，以及建立不同的 Android 應用程式。每個應用程式都有自己的應用程式用戶端 ID。

應用程式用戶端類型

在 Amazon Cognito 中建立應用程式用户端時，您可以根據標準 OAuth 用户端類型 public client (公有用戶端) 和 confidential client (機密用戶端) 預先填入選項。以 client secret (用户端密碼) 設定 confidential client (機密用户端)。如需有關用戶端類型的詳細資訊，請參閱 IETF RFC 6749 #2.1。

Public client (公有用户端)

公有用户端在瀏覽器或行動裝置上執行。因為它沒有受信任的伺服器端資源，因此沒有用户端密碼。

Confidential client (機密用户端)

機密用户端具有伺服器端資源，這些資源可透過 client secret (用户端密碼) 受到信任以用於未經身分驗證的 API 操作。應用程式可在後端伺服器上作為常駐程式或 Shell 指令碼執行。

Client secret (用戶端密碼)

用戶端秘密 (或用戶端密碼) 是固定的字串，您的應用程式必須在對應用程式用戶端的所有 API 請求中使用它。您的應用程式用户端必須具有用户端密碼才能執行 `client_credentials` 授予。如需詳細資訊，請參閱 IETF RFC 6749 #2.3.1。

您無法在建立應用程式後變更密碼。如果您要輪換密碼，可以建立具有新密碼的新應用程式。您也可以刪除應用程式，以封鎖從使用該應用程式用戶端 ID 的應用程式進行存取。

您可以在公有應用程式中使用機密用户端和用户端密碼。使用 Amazon CloudFront 代理添加傳輸`SECRET_HASH`中。如需詳細資訊，請參閱使用 AWS 部落格上的 Amazon CloudFront 代理來保護 Amazon Cognito 的公用用戶端。

JSON Web 權杖

Amazon Cognito 應用程式用戶端可以發出以下類型的 JSON Web 權杖 (JWT)。

身分 (ID) 權杖

您的使用者已透過您的使用者集區進行身分驗證的可驗證陳述式。OpenID Connect (OIDC) 將 ID 權杖規範新增到 OAuth 2.0 定義的存取和重新整理權杖標準中。ID 權杖包含身分資訊，例如您的應用程式可用於建立使用者設定檔和佈建資源的使用者屬性。如需詳細資訊，請參閱使用 ID 權杖。

存取權杖

使用者存取許可的可驗證陳述式。存取權杖包含範圍，這是 OIDC 和 OAuth 2.0 的功能。您的應用程式可以向後端資源呈現範圍，並證明您的使用者集區授權使用者或機器從 API 或其自己的使用者資料中存取資料。具有自訂範圍的存取權杖 (通常來自 M2M 用戶端憑證授予) 會授權對資源伺服器的存取。如需詳細資訊，請參閱使用存取權杖。

重新整理權杖

當使用者的權杖過期時，您的應用程式可以向您的使用者集區提供初始驗證的加密陳述式。重新整理權杖請求會傳回新的未過期存取和 ID 權杖。如需詳細資訊，請參閱使用重新整理權杖。

您可以從 Amazon Cognito 主控台中使用者集區的應用程式整合標籤，為每個應用程式用戶端設定這些權杖的到期日。

應用程式用戶端條款

下列條款是 Amazon Cognito 主控台中應用程式用戶端的可用屬性。

允許的回呼 URL

回呼 URL 表示使用者登入成功後，將重新導向至哪個位置。選擇至少一個回呼 URL。回呼 URL 必須：

- 是絕對 URI。
- 已向用戶端預先註冊。
- 不包含片段元件。

請參閱 OAuth 2.0 - 重新導向端點。

Amazon Cognito 必須使用 HTTPS，而不使用 HTTP，除非是 `http://localhost` 要進行測試。

同時可支援像是 `myapp://example` 等應用程式回呼 URL。

允許的登出 URL

登出 URL 表示您的使用者在登出之後，將重新導向至哪個位置。

屬性讀取和寫入許可

您的用戶池可能有許多客戶，每個客戶都有自己的應用程序客戶端和 IdPs. 您可以將應用程式用戶端設定為僅對與應用程式相關的使用者屬性具有讀取和寫入存取許可。在類似 machine-to-machine (M2M) 授權的情況下，您可以授予任何使用者屬性的存取權。

屬性讀取和寫入權限組態的考量事項

- 當您建立應用程式用戶端且未自訂屬性讀取和寫入權限時，Amazon Cognito 會授予所有使用者集區屬性的讀取和寫入權限。
- 您可以授與不可變自訂屬性的寫入存取權。當您建立或註冊使用者時，您的應用程式用戶只能將值寫入不可變屬性中。在此之後，您無法將值寫入用戶的任何不可變自訂屬性。
- 應用程式用戶必須具有使用者集區中必要屬性的寫入存取權。Amazon Cognito 主控台會自動將必要屬性設定為可寫入。
- 您無法授與 `email_verified` 或 `phone_number_verified` 的寫入權限給應用程式用戶。使用者集區管理員可以修改這些值。使用者只能透過屬性驗證來變更這些屬性的值。

身分驗證流程

您的應用程式用戶端允許登入的方法。您的應用程式可以支援使用者名稱和密碼的驗證、安全遠端密碼 (SRP)、含 Lambda 觸發程式的自訂驗證，以及權杖重新整理。最佳安全性做法是使用 SRP 驗證作為主要登入方法。託管 UI 會自動使用 SRP 登入使用者。

自訂範圍

自訂範圍是您在 Resource Servers (資源伺服器) 索引標籤中，為自己的資源伺服器定義的範圍。格式為 *`resource-server-identifier`*`/##`。請參閱使用資源伺服器進行範圍、M2M 和 API 授權。

預設重新導向 URI

以協力廠商取代使用者驗證要求中的`redirect_uri`參數 IdPs。使用CreateUserPoolClient或UpdateUserPoolClientAPI 請求的`DefaultRedirectURI`參數配置此應用程序客戶端設置。此 URL 也必須是您應用程式用戶端的成員。`CallbackURLs`在以下情況下，Amazon Cognito 會將經過驗證的工作階段重

1. 您的應用程序客戶端分配了一個身份提供商，並定義了多個回調 URL 當您的使用者集區未包含參數時，會將驗證要求重新導向至授權伺服器重新導向 `redirect_uri` URI。
2. 您的應用程序客戶端分配了一個身份提供程序和一個回調 URL。在這個案例中，不需要定義預設回呼 URL。不包含`redirect_uri`參數的要求會重新導向至一個可用的回呼 URL。

身分提供者

您可以選擇部分或全部使用者集區外部身分識別提供者 (IdPs) 來驗證您的使用者。您的應用程式用戶端也可以只驗證使用者集區中的本機使用者。當您將 IdP 新增至應用程式用戶端時，您可以產生指向 IdP 的授權連結，並將其顯示在託管的 UI 登入頁面上。您可以指派多個 IdPs，但必須至少指派一個。如需使用外部的詳細資訊 IdPs，請參閱透過第三方新增使用者集區登入。

OpenID Connect 範圍

選擇下列一或多個 0Auth範圍，以指定可以為存取權杖請求的存取權限。

- openid 範圍宣告您要擷取 ID 權杖和使用者唯一 ID。它還請求全部或部分使用者屬性，具體取決於請求中的其他範圍。Amazon Cognito 不會傳回 ID 權杖，除非您請求 openid 範圍。該 openid 範圍授權結構化 ID 權杖宣告 (例如過期和金鑰 ID)，並確定您在 UserInfo 端點的回應中收到的使用者屬性。
  - 當 openid 是您請求的唯一範圍時，Amazon Cognito 會使用目前應用程式用戶端可讀取的所有使用者屬性填入 ID 權杖。僅具有此範圍的存取權杖的 userInfo 回應會傳回所有使用者屬性。
  - 當您使用其他範圍 (例如 phone、email 或 profile) 請求 openid 時，ID 權杖和 userInfo 將傳回使用者的唯一 ID 以及其他範圍定義的屬性。
- phone範圍可授予對 phone_number 和 phone_number_verified 宣告的存取權。此範圍只能隨 openid範圍一起請求。
- email範圍可授予對 email 和 email_verified 宣告的存取權。此範圍只能隨 openid範圍一起請求。
- 該aws.cognito.signin.user.admin範圍授予對需要存取權杖的 Amazon Cognito 使用者集區 API 操作的存取權，例如UpdateUserAttributes和VerifyUserAttribute。
- profile範圍可供存取用戶端可讀取的所有使用者屬性。此範圍只能隨 openid範圍一起請求。

如需有關範圍的詳細資訊，請參閱標準 OIDC 範圍清單。

OAuth 授予類型

OAuth 授予是一種擷取使用者集區權杖的身分驗證方法。Amazon Cognito 支援下列三種類型的授予。若要將這些 OAuth 授予整合到應用程式中，您必須將網域新增至使用者集區。

授予授權碼

授權碼授予會產生一個程式碼，您的應用程式可以使用該程式碼與 權杖端點 交換使用者集區權杖。在您交換授權碼時，您的應用程式會接收 ID、存取和重新整理權杖。此 OAuth 流程與隱含授

予一樣，發生在使用者的瀏覽器中。授權碼授予是 Amazon Cognito 提供的最安全的授權，因為在您使用者的工作階段中是看不到權杖的。相反，您的應用程式會產生傳回權杖的請求，並可以將其快取在受保護的儲存中。如需詳細資訊，請參閱 IETF RFC 6749 #1.3.1 中的授權碼

> **Note**
>
> 作為公有用戶端應用程式的最佳安全實務，應僅啟用授予 OAuth 流程的授權程式碼，並實現代碼交換證明金鑰 (PKCE) 以限制權杖交換。透過 PKCE，只有在用戶端向權杖端點提供了與原始身分驗證請求中提供的相同秘密時，用戶端才能交換授權碼。如需 PKCE 詳細資訊，請參閱 IETF RFC 7636。

隱含授與

隱含授予會將存取權和 ID 權杖 (但不重新整理權杖) 直接從 授權端點 交付給您的使用者瀏覽器工作階段。隱含授予移除了對權杖端點單獨請求的需求，但與 PKCE 不相容，並且不會傳回重新整理權杖。此授予適用於無法完成授權程式碼授予的測試案例和應用程式體系架構。如需詳細資訊，請參閱 IETF RFC 6749 #1.3.2 中的隱含授予。您可以在應用程式用戶端中啟用授權碼授予以及隱含授予，並視需要使用個別授予。

用戶端憑證授予

用戶端認證授與用於 machine-to-machine (M2M) 通訊。授權程式碼和隱含授予會向經過身分驗證的人類使用者發出權杖。用戶端憑證會將以範圍為基礎的授權從非互動式系統授予 API。您的應用程式可以直接從權杖端點請求用戶端憑證並接收存取權杖。如需詳細資訊，請參閱 IETF RFC 6749 #1.3.4 中的用戶端憑證。您只能在具有用戶端秘密且不支援授權碼或隱含授予的應用程式用戶端中啟用用戶端憑證授予。

> **Note**
>
> 由於您沒有以使用者身分調用用戶端憑證流程，此授予只能新增自訂範圍來存取權杖。自訂範圍是您為自己的資源伺服器定義的範圍。預設範圍如 `openid` 和 `profile` 不適用於非人類使用者。
>
> 由於 ID 權杖是使用者屬性的驗證，因此它們與 M2M 通訊無關，並且用戶端憑證授予不會發給它們。請參閱使用資源伺服器進行範圍、M2M 和 API 授權。

用戶端憑證授予您的帳 AWS 單增加成本。如需詳細資訊，請參閱 Amazon Cognito 定價。

建立應用程式用戶端

AWS Management Console

建立應用程式用戶端 (主控台)

1. 前往 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。
4. 選取 App integration (應用程式整合) 索引標籤。
5. 在 App clients (應用程式用戶端) 下，選取 Create an app client (建立應用程式用戶端)。
6. 選取 App type (應用程式類型)：Public client (公有用戶端)、Confidential client (機密用戶端)，或 Other (其他)。
7. 輸入 App client name (應用程式用戶端名稱)。
8. 選擇 產生用戶端機密，讓 Amazon Cognito 為您產生用戶端密碼。用戶端密碼通常與機密用戶端相關聯。
9. 選取您想要在應用程式用戶端中允許的 Authentication flows (身分驗證流程)。
10. 設定 Authentication flow session duration (身分驗證流程工作階段持續時間)。這是您的使用者必須在工作階段字符到期之前完成每個身分驗證挑戰的時間量。
11. (選用) 如果您想要設定權杖過期，請完成以下步驟：

    a. 指定應用程式用戶端的 Refresh token expiration (重新整理權杖過期)。預設值為 30 天。您可以將其變更為 1 小時到 10 年的任何值。

    b. 指定應用程式用戶端的 Access token expiration (存取權杖過期)。預設值為 1 小時。您可以將其變更為 5 分鐘到 24 小時的任何值。

    c. 指定應用程式用戶端的 ID token expiration (ID 權杖過期)。預設值為 1 小時。您可以將其變更為 5 分鐘到 24 小時的任何值。

    > **Important**
    >
    > 如果您使用託管 UI 並設定小於一小時的權杖生命週期，則使用者可根據目前固定在一小時的其工作階段 Cookie 持續時間，使用權杖。

12. 選擇是否要為此應用程式用戶端 Enable token revocation (啟用權杖撤銷)。這將增加 Amazon Cognito 所簽發權杖的大小。

13. 選擇是否要為此應用程式用戶端 防止使用者存在錯誤。Amazon Cognito 將回應不存在之使用者的登入請求，並顯示一則通用訊息，指出使用者名稱或密碼不正確。
14. 如果要將託管 UI 與此應用程式用戶端搭配使用，請設定 託管 UI 設定。
    a. 輸入一或多個 允許的回呼 URL。這些是您希望 Amazon Cognito 在使用者完成身份驗證後重新導向的網路或應用程式 URL。
    b. 輸入一或多個 允許的登出 URL。這些是您希望應用程式在對 登出端點 的要求中接受的 URL。
    c. 選擇讓您可登入應用程式使用者的一或多個 身分提供者。您可以選擇現有的任意組合 IdPs。您可以單獨使用您的使用者集區，或透過您在使用者集區中設定 IdPs 的一或多個第三方驗證使用者。
    d. 選擇您希望應用程式用戶端接受的 OAuth 2.0 授予類型。
        - 選擇 授權碼授與 以將代碼傳遞給您的應用程式，該代碼可以使用 權杖端點 以兌換權杖。
        - 選取 隱含授與 以將 ID 和存取權杖直接傳遞給您的應用程式。隱含授予流程會直接向您的使用者公開權杖。
        - 選擇 用戶端登入資料，以根據其對用戶端秘密 (非使用者憑證) 之了解，將存取權杖傳遞給您的應用程式。用戶端憑證授予流程和授權代碼與隱含授予流程是互斥的。
    e. 選擇您要授權與您應用程式用戶端一起使用的 OpenID Connect 範圍。您可以透過使用者集區 API 產生僅具有 `aws.cognito.signin.user.admin` 範圍的存取權杖。對於其他範圍，您必須向 權杖端點 要求存取權杖。
    f. 選擇您要使用應用程式用戶端授權的 自訂範圍。自訂範圍最常用於授權第三方 API 的存取權。
15. 為此應用程式用戶端設定 屬性讀取和寫入許可。您的應用程式用戶端可以擁有讀取和寫入使用者集區屬性結構描述子集之所有，或有限的許可。
16. 選擇 Create app client (建立應用程式用戶端)。
17. 請記下 Client id (用戶端 ID)。這會識別註冊和登入請求中的應用程式用戶端。

AWS CLI

```
aws cognito-idp create-user-pool-client --user-pool-id MyUserPoolID --client-
name myApp
```

> **Note**
>
> 請使用 JSON 格式的回呼和登出 URL，以防止 CLI 將其視為遠端參數檔案：
>
> ```
> --callback-urls "["https://example.com"]"
> --logout-urls "["https://example.com"]"
> ```

如需詳細資訊，請參閱 AWS CLI 命令參考：create-user-pool-client

Amazon Cognito user pools API

產生 CreateUserPoolClientAPI 要求。您必須為不想設定為預設值的所有參數指定一個值。

更新使用者集區應用程式用戶端 (AWS CLI 和 AWS API)

在中 AWS CLI，輸入下列命令：

```
aws cognito-idp update-user-pool-client --user-pool-id  "MyUserPoolID" --client-id
 "MyAppClientID" --allowed-o-auth-flows-user-pool-client --allowed-o-auth-flows "code"
 "implicit" --allowed-o-auth-scopes "openid" --callback-urls "["https://example.com"]"
 --supported-identity-providers "["MySAMLIdP", "LoginWithAmazon"]"
```

如果命令成功，則 AWS CLI 返回一個確認：

```
{
    "UserPoolClient": {
        "ClientId": "MyClientID",
        "SupportedIdentityProviders": [
            "LoginWithAmazon",
            "MySAMLIdP"
        ],
        "CallbackURLs": [
            "https://example.com"
        ],
        "AllowedOAuthScopes": [
            "openid"
        ],
        "ClientName": "Example",
        "AllowedOAuthFlows": [
```

```
            "implicit",
            "code"
        ],
        "RefreshTokenValidity": 30,
        "AuthSessionValidity": 3,
        "CreationDate": 1524628110.29,
        "AllowedOAuthFlowsUserPoolClient": true,
        "UserPoolId": "MyUserPoolID",
        "LastModifiedDate": 1530055177.553
    }
}
```

如需詳細資訊，請參閱 AWS CLI 命令參考：update-user-pool-client。

AWS API：UpdateUserPoolClient

取得使用者集區應用程式用戶端 (AWS CLI 和 AWS API) 的相關資訊

```
aws cognito-idp describe-user-pool-client --user-pool-id MyUserPoolID --client-
id MyClientID
```

如需詳細資訊，請參閱 AWS CLI 命令參考：describe-user-pool-client。

AWS API：DescribeUserPoolClient

列出用戶池中的所有應用程序客戶端信息（AWS CLI 和 AWS API）

```
aws cognito-idp list-user-pool-clients --user-pool-id "MyUserPoolID" --max-results 3
```

如需詳細資訊，請參閱 AWS CLI 命令參考：list-user-pool-clients。

AWS API：ListUserPoolClients

刪除使用者集區應用程式用戶端 (AWS CLI 和 AWS API)

```
aws cognito-idp delete-user-pool-client --user-pool-id "MyUserPoolID" --client-id
 "MyAppClientID"
```

如需詳細資訊，請參閱 AWS CLI 命令參考：delete-user-pool-client

AWS API：DeleteUserPoolClient

在使用者集區中使用使用者裝置

當您使用 Amazon Cognito 使用者集區 API 登入本機使用者集區的使用者時，可以將來自進階安全功能的使用者活動日誌與其每部裝置建立關聯，並選擇性地允許使用者在受信任的裝置上略過多重要素驗證 (MFA)。Amazon Cognito 會針對任何尚未包含裝置資訊的登入，在回應中包含裝置金鑰。裝置金鑰的格式為 *Region_UUID*。使用裝置金鑰、Secure Remote Password (SRP) 程式庫，以及允許裝置身分驗證的使用者集區，您就可以在應用程式中提示使用者信任目前的裝置，並且不再於登入時提示輸入 MFA 代碼。

主題

- 設定記住的裝置
- 取得裝置金鑰
- 使用裝置登入
- 檢視、更新和忘記裝置

設定記住的裝置

使用 Amazon Cognito 使用者集區，您就可以將使用者的每一部裝置與唯一裝置識別碼 (也就是裝置金鑰) 建立關聯。當您在登入時出示裝置金鑰並執行裝置身分驗證時，可以利用兩項功能。

1. 基於安全和分析目的，您可以利用進階安全功能監控特定裝置上的使用者活動。當使用者登入時，您的應用程式可以選擇對每一位使用者及其裝置進行身分驗證，並將裝置資訊新增至使用者的活動日誌。
2. 記住裝置還可支援受信任的裝置身分驗證流程，在此流程中，您的使用者可以選擇在適合應用程式安全需求的時段內，不使用 MFA 進行登入。當您想要重新提示使用者提交 MFA 代碼時，您可以變更其裝置的記住狀態。

   記住的裝置只能在 MFA 處於作用中狀態的使用者集區中覆寫 MFA。

當您的使用者使用記住的裝置登入時，您必須在其身分驗證流程中執行額外的裝置身分驗證。如需更多詳細資訊，請參閱 使用裝置登入。

在使用者集區的登入體驗索引標籤中，於裝置追蹤下設定您的使用者集區以記住裝置。透過 Amazon Cognito 主控台來設定記住的裝置功能時，您有三個選項：Always (一律)、User Opt-In (使用者選擇記住) 和 No (否)。

不要記住

您的使用者集區不會在登入時提示使用者記住裝置。

永遠記住

當您的應用程式確認使用者的裝置時，您的使用者集區會一律記住該裝置，而且後續裝置成功登入時，不會傳回 MFA 挑戰。

使用者選擇加入

當您的應用程式確認使用者的裝置時，您的使用者集區不會自動抑制 MFA 挑戰。您必須提示使用者選擇是否要記住裝置。

若您選擇永遠記住或使用者選擇加入，每次使用者從無法辨識的裝置登入時，Amazon Cognito 都會產生裝置識別碼金鑰和秘密。裝置金鑰是使用者執行裝置身分驗證時，您的應用程式傳送至使用者集區的初始識別碼。

無論使用者裝置是自動記住或選擇加入，一經確認後，您就可以使用裝置識別碼金鑰和秘密在每次使用者登入時進行裝置身分驗證。

您也可以在CreateUserPool 或 UpdateUserPool API 請求中，為使用者集區設定記住的裝置設定。如需詳細資訊，請參閱 DeviceConfiguration 屬性。

Amazon Cognito 使用者集區 API 有其他可針對記住的裝置進行的操作。

1. ListDevices 和 AdminListDevices 會傳回使用者的裝置金鑰及其中繼資料的清單。
2. GetDevice 和 AdminGetDevice 會傳回單一裝置的裝置金鑰和中繼資料。
3. UpdateDeviceStatus 和 AdminUpdateDeviceStatus 會將使用者裝置設定為記住或不記住。
4. ForgetDevice 和 AdminForgetDevice 會從設定檔中移除使用者已確認的裝置。

名稱開頭為 `Admin` 的 API 操作可用於伺服器端應用程式，且必須透過 IAM 憑證授權。如需更多詳細資訊，請參閱 使用 Amazon Cognito 使用者集區 API 和使用者集區端點。

取得裝置金鑰

只要您的使用者透過使用者集區 API 登入，且未在身分驗證參數中包含裝置金鑰作為 `DEVICE_KEY`，Amazon Cognito 就會在回應中傳回新的裝置金鑰。在您的公有用戶端應用程式中，將裝置金鑰放入應用程式儲存中，您就可以在後續請求中包含該金鑰。在機密伺服器端應用程式中，使用您使用者的裝置金鑰設定瀏覽器 Cookie 或其他用戶端權杖。

您的應用程式必須先確認裝置金鑰並提供其他資訊，您的使用者才能使用其受信任的裝置登入。對 Amazon Cognito 產生 ConfirmDevice 請求，以透過裝置金鑰、易記名稱、密碼驗證程式和 salt 來確認使用者裝置。如果您設定使用者集區進行選擇加入裝置身分驗證，Amazon Cognito 會回應您的 `ConfirmDevice` 請求，並提示您的使用者必須選擇是否要記住目前的裝置。在 UpdateDeviceStatus 請求中以使用者的選擇回應。

若您確認使用者的裝置，但未將其設定為記住，Amazon Cognito 會儲存關聯，但是在您提供裝置金鑰時，會進行非裝置登入。裝置可以產生日誌，對於使用者安全和疑難排解來說很實用。已確認但未記住的裝置不會利用登入功能，而是利用安全監控日誌功能。當您為應用程式用戶端啟用進階安全功能，並將裝置用量編碼到請求中時，Amazon Cognito 會將使用者事件與已確認的裝置建立關聯。

取得新的裝置金鑰

1. 透過 InitiateAuth API 請求啟動使用者登入工作階段。
2. 使用 RespondToAuthChallenge 回應所有身分驗證挑戰，直到收到將您的使用者登入工作階段標記為完成的 JSON Web 權杖 (UWT) 為止。
3. 在您的應用程式中，記錄 Amazon Cognito 在其 `RespondToAuthChallenge` 或 `InitiateAuth` 回應中的 `NewDeviceMetadata` 傳回的值：`DeviceGroupKey` 和 `DeviceKey`。
4. 為您的使用者產生新的 SRP 秘密：salt 和密碼驗證程式。此功能可在提供 SRP 程式庫的 SDK 中使用。
5. 提示您的使用者輸入裝置名稱，或根據使用者裝置的特性產生名稱。
6. 在 ConfirmDevice API 請求中，提供使用者的存取權仗、裝置金鑰、裝置名稱及 SRP 秘密。如果您的使用者集區設定為永遠記住裝置，表示您的使用者註冊已完成。
7. 如果 Amazon Cognito 以 `"UserConfirmationNecessary": true` 回應 `ConfirmDevice`，請提示您的使用者選擇是否要記住裝置。如果使用者確認要記住裝置，請利用使用者的存取權杖、裝置金鑰及 `"DeviceRememberedStatus": "remembered"` 產生 UpdateDeviceStatus API 請求。
8. 如果您已指示 Amazon Cognito 記住裝置，則會在使用者下一次登入時，對其顯示 `DEVICE_SRP_AUTH` 挑戰，而不是 MFA 挑戰。

使用裝置登入

設定要記住的使用者裝置後，當使用者使用相同的裝置金鑰登入時，Amazon Cognito 就不會再要求他們提交 MFA 代碼。裝置身分驗證只會將 MFA 身分驗證挑戰取代為裝置身分驗證挑戰。您無法只使用裝置身分驗證讓使用者登入。您的使用者必須先使用其密碼或自訂挑戰完成身分驗證。以下是使用者在記住的裝置上進行身分驗證的程序。

若要在使用自訂身分驗證挑戰 Lambda 觸發條件的流程中執行裝置身分驗證，請在 InitiateAuth API 請求中傳遞 DEVICE_KEY 參數。在使用者成功通過所有挑戰，且 CUSTOM_CHALLENGE 挑戰傳回的 issueTokens 值 true 之後，Amazon Cognito 就會傳回最後一項 DEVICE_SRP_AUTH 挑戰。

使用裝置登入

1. 從用戶端儲存擷取使用者的裝置金鑰。
2. 透過 InitiateAuth API 請求啟動使用者登入工作階段。選擇 USER_SRP_AUTH、REFRESH_TOKEN_AUTH、USER_PASSWORD_AUTH 或 CUSTOM_AUTH 的 AuthFlow。在 AuthParameters 中，將使用者的裝置金鑰新增至 DEVICE_KEY 參數，並包含所選登入流程的其他必要參數。

   a. 您也可以在 PASSWORD_VERIFIER 回應的參數中，將 DEVICE_KEY 傳遞至身分驗證挑戰。
3. 完成挑戰回應，直到您在回應中收到 DEVICE_SRP_AUTH 挑戰為止。
4. 在 RespondToAuthChallenge API 請求中，傳送 DEVICE_SRP_AUTH 的 ChallengeName，以及 USERNAME、DEVICE_KEY 和 SRP_A 的參數。
5. Amazon Cognito 會以 DEVICE_PASSWORD_VERIFIER 挑戰回應。此挑戰回應包括 SECRET_BLOCK 和 SRP_B 的值。
6. 使用 SRP 程式庫產生並提交 PASSWORD_CLAIM_SIGNATURE、PASSWORD_CLAIM_SECRET_BLOCK、TIMESTAMP、USERNAME 和 DEVICE_KEY 參數。在另一個 RespondToAuthChallenge 請求中提交這些參數。
7. 完成其他挑戰，直到您收到使用者的 JWT 為止。

下列虛擬程式碼示範如何計算 DEVICE_PASSWORD_VERIFIER 挑戰回應的值。

```
PASSWORD_CLAIM_SECRET_BLOCK = SECRET_BLOCK
TIMESTAMP = Tue Sep 25 00:09:40 UTC 2018
PASSWORD_CLAIM_SIGNATURE = Base64(SHA256_HMAC(K_USER, DeviceGroupKey + DeviceKey +
 PASSWORD_CLAIM_SECRET_BLOCK + TIMESTAMP))
K_USER = SHA256_HASH(S_USER)
S_USER = (SRP_B - k * gx)(a + ux)
x = SHA256_HASH(salt + FULL_PASSWORD)
u = SHA256_HASH(SRP_A + SRP_B)
k = SHA256_HASH(N + g)
```

檢視、更新和忘記裝置

您可以使用 Amazon Cognito API 在應用程式中實作下列功能。

1. 顯示有關使用者目前裝置的資訊。
2. 顯示使用者所有裝置的清單。
3. 忘記裝置。
4. 更新裝置的記住狀態。

下列說明中授權給 API 請求的存取權杖必須包含 `aws.cognito.signin.user.admin` 範圍。Amazon Cognito 會將此範圍的宣告新增至您使用 Amazon Cognito 使用者集區 API 產生的所有存取權杖。第三方 IdP 必須針對向 Amazon Cognito 進行身分驗證的使用者，分別管理裝置和 MFA。在託管 UI 中，您可以請求 `aws.cognito.signin.user.admin` 範圍，但託管 UI 會自動將裝置資訊新增至進階安全使用者日誌，且不會提議記住裝置。

顯示有關裝置的資訊

您可以查詢有關使用者裝置的資訊，以判斷該裝置目前是否仍在使用中。例如，您可能希望在記住的裝置未登入長達 90 天後，將裝置停用。

- 若要在公有用戶端應用程式中顯示使用者的裝置資訊，請在 GetDevice API 請求中提交使用者的存取金鑰和裝置金鑰。
- 若要在機密用戶端應用程式中顯示使用者的裝置資訊，請使用 AWS 憑證簽署 AdminGetDevice API 請求，然後提交使用者的使用者名稱、裝置金鑰和使用者集區。

顯示使用者所有裝置的清單

您可以顯示使用者所有裝置及其屬性的清單。例如，您可能希望驗證目前的裝置是否與記住的裝置相符。

- 在公有用戶端應用程式中，於 ListDevices API 請求中提交使用者的存取權仗。
- 在機密用戶端應用程式中，使用 AWS 憑證簽署 AdminListDevices API 請求，然後提交使用者的使用者名稱和使用者集區。

忘記裝置

您可以刪除使用者的裝置金鑰。當您判斷使用者不再使用某部裝置，或是偵測到異常活動並且想要提示使用者再次完成 MFA 時，就會希望這麼做。若稍後要再次註冊裝置，您必須產生並儲存新的裝置金鑰。

- 在公有用戶端應用程式中，於 ForgetDevice API 請求中提交使用者的裝置金鑰和存取權仗。

- 在機密用戶端應用程式中，於 AdminForgetDevice API 請求中提交使用者的裝置金鑰和存取權仗。

使用 Amazon Cognito 使用者集區 API 和使用者集區端點

當您想要註冊、登入和管理使用者集區中的使用者，您有兩種選擇。

1. 您的使用者集區端點包括託管 UI 和聯合端點。在您為使用者集區選擇網域時，會共同組成 Amazon Cognito 啟用的公開網頁套件。若要快速入門 Amazon Cognito 使用者集區的身分驗證和授權功能，包括用於註冊、登入、密碼管理和多重要素驗證 (MFA) 的頁面，請使用託管 UI 的內建使用者介面。其他使用者集區端點有助於通過第三方身分提供者 (IdP) 的身分驗證。執行的服務包括下列項目。
   a. 服務提供者回呼來自 IdP 的已驗證宣告的端點，例如 `saml2/idpresponse` 和 `oauth2/idpresponse`。當 Amazon Cognito 是應用程式和 IdP 之間的中繼服務提供者 (SP) 時，回呼端點代表該服務。
   b. 提供環境相關資訊的端點，例如 `oauth2/userInfo` 和 `jwks.json`。當您的應用程式使用 AWS SDK 和 OAuth 2.0 資料庫驗證權杖或取得使用者設定檔資料時會使用這些端點。
2. Amazon Cognito 使用者集區 API 是可讓您的網頁或行動應用程式在您自己的自訂前端中收集登入資訊，以驗證使用者的一套工具。使用者集區 API 驗證會產生下列 JSON Web 權杖。
   a. 使用者具有驗證屬性宣告的身分權杖。
   b. 存取權杖授權使用者向 AWS服務端點建立權杖授權的 API 請求。

> **Note**
>
> 預設中，來自使用者集區 API 身分驗證的存取權杖僅包含 `aws.cognito.signin.user.admin` 範圍。如果您想要產生具有其他範圍的存取權杖，例如授權對第三方 API 的請求，請透過使用者集區端點在驗證時要求範圍，或在 產生權杖前 Lambda 觸發程序 中新增自訂範圍。存取權杖自訂會增加您 AWS 帳單的成本。

您可以將通常透過使用者集區端點登入的聯合身分使用者連結至您使用者集區的本機身分使用者。本機使用者僅存在於您的使用者集區目錄中，不會透過外部 IdP 進行聯合。如果您在 AdminLinkProviderForUser API 請求中將其聯合身分連結至本機使用者，則他們可以使用使用者集區 API 登入。如需更多詳細資訊，請參閱 將聯合身分使用者連結至現有的使用者描述檔。

Amazon Cognito 使用者集區 API 有雙重用途。它會建立和設定您的 Amazon Cognito 使用者集區資源。例如，您可以建立使用者集區、新增 AWS Lambda 觸發程序，以及設定託管 UI 網域。使用者集區 API 也為本機和連結使用者執行註冊、登入和其他使用者操作。

Amazon Cognito 使用者集區 API 的範例案例

1. 您的使用者選擇您在應用中建立的「建立帳戶」按鈕。他們輸入電子郵件地址和密碼。
2. 您的應用程式傳送 SignUp API 請求，並在使用者集區中建立新使用者。
3. 您的應用程式提示使用者輸入電子郵件確認碼。您的使用者輸入他們在電子郵件訊息中收到的代碼。
4. 您的應用程式傳送帶有使用者確認代碼的 ConfirmSignUp API 請求。
5. 您的應用程式會提示您的使用者輸入其使用者名稱和密碼，然後輸入其資訊。
6. 您的應用程式傳送 InitiateAuth API 請求並儲存 ID 權杖、存取權杖和重新整理權杖。您的應用程式會呼叫 OIDC 程式庫以管理使用者權杖並維持該使用者的持久性工作階段。

在 Amazon Cognito 使用者集區 API 中，您無法登入透過 IdP 聯合的使用者。您必須透過使用者集區端點驗證這些使用者。如需包含託管 UI 之使用者集區端點的詳細資訊，請參閱 使用者集區聯合端點和託管 UI 參考。您的聯合身分使用者可以從託管 UI 中開始並選取其 IdP，或者您可以略過託管 UI ，並將使用者直接傳送至您的 IdP 來登入。當您對 授權端點 的 API 請求包含 IdP 參數時，Amazon Cognito 會以無訊息方式將您的使用者重新導向至 IdP 登入頁面。

使用者集區端點的範例案例

1. 您的使用者選擇您在應用程式中建立的「建立帳戶」按鈕。
2. 您向使用者顯示已註冊開發人員憑證的社交身分提供者清單。您的使用者選擇 Apple。
3. 您的應用程式會以提供者名稱 `SignInWithApple` 向 授權端點 展開請求。
4. 您使用者的瀏覽器會開啟 Apple OAuth 授權頁面。您的使用者選擇允許 Amazon Cognito 讀取其設定檔資訊。
5. Amazon Cognito 確認 Apple 存取權杖並查詢使用者的 Apple 設定檔。
6. 您的使用者會向您的應用程式提供 Amazon Cognito 授權碼。
7. 您的應用程式與 權杖端點 交換授權碼，並儲存 ID 權杖、存取權杖和重新整理權杖。您的應用程式會呼叫 OIDC 程式庫以管理使用者權杖並維持該使用者的持久性工作階段。

本指南中所述的使用者集區 API 和使用者集區端點支援各種案例。以下各節將探討使用者集區 API 如何進一步劃分為支援您註冊、登入和資源管理需求的類別。

Amazon Cognito 使用者集區經身分驗證和未進行身分驗證的 API 操作

Amazon Cognito 使用者集區 API (同時是資源管理介面和面向使用者的身分驗證和授權介面) 根據其操作結合授權模型。根據 API 操作，您可能必須提供 IAM 憑證、存取權杖、工作階段權杖、用戶端秘密，或是以上的組合來提供授權。對於許多使用者身分驗證和授權操作，您可以選擇經身分驗證和未進行身分驗證的請求版本。對於分發給使用者的應用程式 (例如行動應用程式) 而言，未進行身分驗證的操作是最佳安全實務；您不需要在程式碼中加入任何秘密。

您只能在 以 IAM 身分驗證的管理操作 和 以 IAM 身分驗證的使用者操作 的 IAM 政策中指派許可。

以 IAM 身分驗證的管理操作

以 IAM 身分驗證的管理操作會修改和檢視您的使用者集區和應用程式用戶端設定，就像在 AWS Management Console。

例如，若要修改 UpdateUserPool API 請求中的使用者集區，您必須出示 AWS 憑證和 IAM 許可才能更新資源。

若要在 AWS Command Line Interface (AWS CLI) 或 AWS SDK 中授權這些請求，請使用環境變數或用戶端設定來設定環境，將 IAM 憑證新增至請求。如需詳細資訊，請參閱 AWS 一般參考 中的使用 AWS 憑證存取 AWS。您也可以將請求直接傳送到 Amazon Cognito 使用者集區 API 的服務端點。您必須使用內嵌在請求標頭中的 AWS 憑證來授權或簽署這些請求。如需相關資訊，請參閱簽署 AWS API 請求。

以 IAM 身分驗證的管理操作

AddCustomAttributes

CreateGroup

CreateIdentityProvider

CreateResourceServer

CreateUserImportJob

CreateUserPool

CreateUserPoolClient

CreateUserPoolDomain

以 IAM 身分驗證的管理操作

DeleteGroup

DeleteIdentityProvider

DeleteResourceServer

DeleteUserPool

DeleteUserPoolClient

DeleteUserPoolDomain

DescribeIdentityProvider

DescribeResourceServer

DescribeRiskConfiguration

DescribeUserImportJob

DescribeUserPool

DescribeUserPoolClient

DescribeUserPoolDomain

GetCSVHeader

GetGroup

GetIdentityProviderByIdentifier

GetSigningCertificate

GetUICustomization

GetUserPoolMfaConfig

ListGroups

ListIdentityProviders

以 IAM 身分驗證的管理操作

ListResourceServers

ListTagsForResource

ListUserImportJobs

ListUserPoolClients

ListUserPools

ListUsers

ListUsersInGroup

SetRiskConfiguration

SetUICustomization

SetUserPoolMfaConfig

StartUserImportJob

StopUserImportJob

TagResource

UntagResource

UpdateGroup

UpdateIdentityProvider

UpdateResourceServer

UpdateUserPool

UpdateUserPoolClient

UpdateUserPoolDomain

以 IAM 身分驗證的使用者操作

以 IAM 身分驗證的使用者操作註冊、登入、管理憑證、修改及檢視您的使用者。

例如，您可以擁有一個支援 Web 前端的伺服器端應用程式層。您的伺服器端應用程式是您信任的 OAuth 機密用戶端，可獲得 Amazon Cognito 資源的特殊權限存取權。若要在應用程式中註冊使用者，您的伺服器可以在 AdminCreateUser API 請求中包含 AWS 憑證。有關 OAuth 用戶端類型的詳細資訊，請參閱 OAuth 2.0 授權架構中的用戶端類型。

若要在 AWS CLI 或 AWS SDK 中授權這些請求，請使用環境變數或用戶端設定來設定伺服器端應用程式環境，將 IAM 憑證新增至請求。如需詳細資訊，請參閱 AWS 一般參考 中的使用 AWS 憑證存取 AWS。您也可以將請求直接傳送到 Amazon Cognito 使用者集區 API 的服務端點。您必須使用內嵌在請求標頭中的 AWS 憑證來授權或簽署這些請求。如需相關資訊，請參閱簽署 AWS API 請求。

如果您的應用程式用戶端具有用戶端秘密，您必須同時提供 IAM 憑證，並根據操作提供 `SecretHash` 參數或 `AuthParameters` 中的 `SECRET_HASH` 值。如需更多詳細資訊，請參閱 運算私密雜湊值。

以 IAM 身分驗證的使用者操作

`AdminAddUserToGroup`

`AdminConfirmSignUp`

`AdminCreateUser`

`AdminDeleteUser`

`AdminDeleteUserAttributes`

`AdminDisableProviderForUser`

`AdminDisableUser`

`AdminEnableUser`

`AdminForgetDevice`

`AdminGetDevice`

`AdminGetUser`

`AdminInitiateAuth`

以 IAM 身分驗證的使用者操作

AdminLinkProviderForUser

AdminListDevices

AdminListGroupsForUser

AdminListUserAuthEvents

AdminRemoveUserFromGroup

AdminResetUserPassword

AdminRespondToAuthChallenge

AdminSetUserMFAPreference

AdminSetUserPassword

AdminSetUserSettings

AdminUpdateAuthEventFeedback

AdminUpdateDeviceStatus

AdminUpdateUserAttributes

AdminUserGlobalSignOut

未經身分驗證的使用者操作

未經身分驗證的使用者操作為您的使用者註冊、登入及啟動密碼重設。當您希望網際網路上的任何人註冊並登入您的應用程式，請使用未經身分驗證或公有 API 操作。

例如，要在您的應用程式中註冊使用者，您可以分發 OAuth 公有用戶端，該用戶端不提供對秘密的特殊存取權限。您可以未經身分驗證的 API 操作 SignUp 註冊此使用者。

若要在您使用 AWS SDK 開發的公有用戶端中傳送這些請求，您不需要設定任何憑證。您也可以將請求直接傳送到 Amazon Cognito 使用者集區 API 的服務端點，不需要額外授權。

如果您的應用程式用戶端具有用戶端秘密，則必須根據操作提供 SecretHash 參數或 AuthParameters 中的 SECRET_HASH 值。如需更多詳細資訊，請參閱 運算私密雜湊值。

未經身分驗證的使用者操作

SignUp

ConfirmSignUp

ResendConfirmationCode

ForgotPassword

ConfirmForgotPassword

InitiateAuth

權杖授權的使用者操作

權杖授權的使用者操作會在使用者登入或開始登入程序後，登出、管理憑證、修改和檢視您的使用者。如果您不想在應用程式中分發秘密，並且想要以使用者自己的憑證授權請求時，請使用權杖授權 API 操作。如果您的使用者已完成登入，則必須使用存取權杖授權其權杖授權的 API 請求。如果您的使用者正在進行登入程序，您必須使用 Amazon Cognito 在回應先前請求時傳回的工作階段權杖來授權其權杖授權的 API 請求。

例如，在公有用戶端中，您可能會想要將更新使用者設定檔的寫入存取權限，限制為僅針對使用者自己的設定檔。若要進行此更新，您的用戶端可以在 UpdateUserAttributes API 請求中包含使用者的存取權杖。

若要在您使用 AWS SDK 開發的公有用戶端中傳送這些請求，您不需要設定任何憑證。在請求中包含 AccessToken 或 Session 參數。您也可以將請求直接傳送到 Amazon Cognito 使用者集區 API 的服務端點。要授權對服務端點的請求，請在請求的 POST 內文中包含存取或工作階段權杖。

若要簽署權杖授權操作的 API 請求，請在請求中以 Bearer *<Base64-encoded access token>* 格式包括存取權杖作為 Authorization 標頭。

權杖授權的使用者操作	AccessToken	Session (工作階段)
RespondToAuthChallenge		✓
ChangePassword	✓	
GetUser	✓	
UpdateUserAttributes	✓	
DeleteUserAttributes	✓	
DeleteUser	✓	
ConfirmDevice	✓	
ForgetDevice	✓	
GetDevice	✓	
ListDevices	✓	
UpdateDeviceStatus	✓	
GetUserAttributeVerificationCode	✓	
VerifyUserAttribute	✓	
SetUserSettings	✓	
SetUserMFAPreference	✓	

權杖授權的使用者操作	AccessToken	Session (工作階段)
GlobalSignOut	✓	
AssociateSoftwareToken	✓	✓
UpdateAuthEventFeedback		✓
VerifySoftwareToken	✓	✓
RevokeToken [1]		

[1] RevokeToken 需要重新整理權杖作為參數。重新整理權杖用作授權權杖，並作為目標資源。

更新使用者集區組態

若要變更中 Amazon Cognito 使用者集區的設定 AWS Management Console，請瀏覽使用者集區設定中的功能型索引標籤，並按照本指南其他區域的說明更新欄位。建立使用者集區之後，您就無法變更某些設定。如果您想要變更以下設定，則必須建立新的使用者集區或應用程式用戶端。

使用者集區名稱

API 參數名稱：PoolName

您指派給您使用者集區的易記名稱。若要變更使用者集區的名稱，請建立新的使用者集區。

Amazon Cognito 使用者集區登入選項

API 參數名稱：AliasAttributes和 UsernameAttributes

使用者登入時可以作為使用者名稱傳遞的屬性。當您建立使用者集區時，您可以選擇允許以使用者名稱、電子郵件地址、電話號碼或偏好的使用者名稱進行登入。若要變更使用者集區登入選項，請建立新的使用者集區。

讓使用者名稱區分大小寫

API 參數名稱：UsernameConfiguration

當您建立的使用者名稱符合其他使用者名稱 (字母大小寫除外) 時，Amazon Cognito 可以將他們視為同一個使用者或獨一無二的多個使用者。如需詳細資訊，請參閱 使用者集區大小寫區分。若要變更區分大小寫，請建立新的使用者集區。

Client secret (用戶端密碼)

API 參數名稱：GenerateSecret

若建立應用程式用戶端，您可以產生用戶端密碼，以便只有信任的來源才能向您的使用者集區提出請求。如需詳細資訊，請參閱 使用者集區應用程式用戶端。若要變更用戶端密碼，請在同一個使用者集區中建立新的應用程式用戶端。

必要屬性

API 參數名稱：綱要

您的使用者註冊時，或您建立使用者時，您的使用者必須提供其值的屬性。如需詳細資訊，請參閱使用者集區屬性。若要變更所需的屬性，請建立新的使用者集區。

自訂屬性

API 參數名稱：綱要

具有自訂名稱的屬性。您可以變更使用者的自訂屬性值，但無法將自訂屬性從使用者集區中刪除。如需詳細資訊，請參閱 使用者集區屬性。如果達到自訂屬性數目上限，且想要修改清單，請建立一個新的使用者集區。

簡訊設定

在使用者集區中啟用 SMS 訊息之後，就無法停用它們。

- 如果您在建立使用者集區時選擇設定 SMS 訊息，則在完成設定後無法停用 SMS。
- 您可以在建立的使用者集區中啟用 SMS 訊息，但之後就無法停用 SMS。
- Amazon Cognito 可以使用簡訊進行使用者帳戶邀請和復原、屬性驗證和多重要素身份驗證 (MFA)。啟用簡訊後，您可以隨時開啟或關閉這些功能的 SMS 訊息。
- SMS 訊息組態包括一個 IAM 角色，您可以將其委派給 Amazon Cognito，以便透過 Amazon SNS 傳送訊息。您可以隨時變更指定的角色。

使用 AWS SDK 或 REST API 更新使用者集區 AWS CDK

在 Amazon Cognito 主控台中，您可以一次變更一個參數的使用者集區設定。例如，若要新增 Lambda 觸發器，您可以選擇新增 Lambda 觸發器，然後選擇函數和觸發器類型。Amazon Cognito 使用者集區 API 的結構方式為使用者集區和應用程式用戶端更新操作需要使用者集區的完整參數集。不過，主控台會透明地將此更新作業與您的其他使用者集區設定自動化。

有時您可能會發現，當更新與 AWS 帳戶 您要更改的設置無關時，可能會導致更新生成錯誤。例如 AWS WAF，已刪除的 Amazon SES 身分識別或 IAM 許可中的變更。如果其中一個目前的參數不再有效，則在修復之前無法更新設定。遇到此類錯誤時，請檢查錯誤回應並驗證其所提及的設定。

Amazon Cognito 使用者集區 REST API 和AWS 開發套件是用於 Amazon Cognito 資源的自動化和程式設計配置的工具。AWS Cloud Development Kit (AWS CDK)使用這些工具的請求也必須像 Amazon Cognito 主控台一樣，在請求主體中使用完整的資源組態更新設定。在較高層次，您必須執行以下過程。

1. 從描述現有資源組態的作業擷取輸出。
2. 通過設置更改修改輸出。
3. 在更新資源的作業中傳送修改過的組態。

下列程序會透過 UpdateUserPoolAPI 作業更新您的組態。具有不同輸入字段的相同方法適用於 UpdateUserPoolClient。

> **Important**
>
> 如果您未提供現有參數的值，則 Amazon Cognito 會將這些參數設定為預設值。例如，當您有現有 `LambdaConfig` 並提交了 `UpdateUserPool` 搭配空白 `LambdaConfig`，您會刪除所有 Lambda 函數給使用者集區觸發條件的指派。當您希望自動變更您的使用者集區組態時，請相應進行規劃。

1. 使用擷取使用者集區的現有狀態 DescribeUserPool。
2. 設定 `DescribeUserPool` 的輸出格式，以符合 `UpdateUserPool` 的請求參數。將下列頂層欄位及其子物件自輸出 JSON 中移除。
   - `Arn`
   - `CreationDate`

- CustomDomain
  - 使用 UpdateUserPoolDomainAPI 作業更新此欄位。
- Domain
  - 使用 UpdateUserPoolDomainAPI 作業更新此欄位。
- EmailConfigurationFailure
- EstimatedNumberOfUsers
- Id
- LastModifiedDate
- Name
- SchemaAttributes
- SmsConfigurationFailure
- Status

3. 確認產生的 JSON 符合 UpdateUserPool 的請求參數。
4. 修改您想要在產生的 JSON 中變更的任何參數。
5. 將您修改的 JSON 作為請求輸入來提交 UpdateUserPool API 請求。

您也可以在 AWS CLI以 update-user-pool 的 --cli-input-json 參數來使用此修改後的 DescribeUserPool 輸出。

或者，執行下列 AWS CLI 命令，為接受的輸入欄位產生包含空白值的 update-user-pool JSON。然後，您可以使用使用者集區中的現有值填入這些欄位。

```
aws cognito-idp update-user-pool --generate-cli-skeleton --output json
```

執行以下命令，為應用程式用戶端產生相同的 JSON 物件。

```
aws cognito-idp update-user-pool-client --generate-cli-skeleton --output json
```

設定和使用 Amazon Cognito 託管 UI 和聯合端點

具有網域的 Amazon Cognito 使用者集區是符合 OAuth-2.0 規範的授權伺服器，以及用於身份驗證的 ready-to-use 託管使用者介面 (UI)。授權伺服器會路由傳送驗證請求、發出和管理 JSON Web 權杖 (JWT)，並傳遞使用者屬性資訊。託管 UI 是使用者集區中，用於基本註冊、登入、多重要素驗證和密

碼重設活動的 Web 介面集合。它也是您與應用程式關聯的第三方身分識別提供者 (IdPs) 進行驗證的中心樞紐。當您要進行身分驗證和授權使用者時，您的應用程式可以調用託管 UI 和授權端點。您可以使用自己的標誌和 CSS 自訂讓託管 UI 使用者體驗合乎您的品牌需求。如需託管 UI 和授權伺服器之元件的詳細資訊，請參閱 使用者集區聯合端點和託管 UI 參考。

> **Note**
>
> Amazon Cognito 託管 UI 不支援使用自訂身分驗證挑戰 Lambda 觸發程序進行自訂身分驗證。

主題

使用設置託管 UI AWS Amplify

如果您使用將身份驗證添加 AWS Amplify 到 Web 或移動應用程序，則可以使用 AWS Amplify 框架中的命令行界面（CLI）和庫來設置託管 UI。若要在您的應用程式中新增身分驗證，您可以使用 AWS Amplify CLI，將 `Auth` 類別新增至您的專案。然後，在用戶端程式碼中，您可以使用程式 AWS Amplify 庫透過 Amazon Cognito 使用者集區驗證使用者。

您可以顯示預先建置的託管 UI，或者透過 OAuth 2.0 端點聯合使用者，並由其重新導向至社交登入供應商，例如 Facebook、Google、Amazon 或 Apple。當使用者成功與社交供應商進行身分驗證後，AWS Amplify 會視需要在使用者集區中建立新使用者，並為您的應用程式提供使用者的 OIDC 權杖。

以下示例顯示如 AWS Amplify 何使用在應用程序中使用社交提供商設置託管 UI。

- AWS Amplify 的驗證 JavaScript。
- AWS Amplify 身份驗證斯威夫特。
- AWS Amplify 對於撲動的身份驗證。

- AWS Amplify 安卓的身份驗證。

使用 Amazon Cognito 主控台設定託管 UI

建立應用程式用戶端

1. 前往 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。
4. 選取 App integration (應用程式整合) 索引標籤。
5. 在 App clients (應用程式用戶端) 下，選取 Create an app client (建立應用程式用戶端)。
6. 選取 App type (應用程式類型)：Public client (公有用戶端)、Confidential client (機密用戶端)，或 Other (其他)。Public client (公有用戶端) 通常從您的使用者裝置運作，並使用未進行身分驗證和已進行權杖驗證的 API。機密用戶端通常是透過您信任的中央伺服器上的應用程式運作，該應用程式會使用用戶端密碼和 API 認證，並使用授權標頭和 AWS Identity and Access Management 認證來簽署要求。對於 Public client (公有用戶端) 或 Confidential client (機密客戶端)，如果您的使用案例與預先設定的應用程式用戶端設定不同，則請選取 Other (其他)。
7. 輸入 App client name (應用程式用戶端名稱)。
8. 選取您想要在應用程式用戶端中允許的 Authentication flows (身分驗證流程)。
9. 設定 Authentication flow session duration (身分驗證流程工作階段持續時間)。這是您的使用者必須在工作階段字符到期之前完成每個身分驗證挑戰的時間量。
10. (選用) 設定權杖過期。
    a. 指定應用程式用戶端的 Refresh token expiration (重新整理權杖過期)。預設值為 30 天。您可以將其變更為 1 小時到 10 年的任何值。
    b. 指定應用程式用戶端的 Access token expiration (存取權杖過期)。預設值為 1 小時。您可以將其變更為 5 分鐘到 24 小時的任何值。
    c. 指定應用程式用戶端的 ID token expiration (ID 權杖過期)。預設值為 1 小時。您可以將其變更為 5 分鐘到 24 小時的任何值。

> **Important**
>
> 如果您使用託管 UI 並設定小於一小時的權杖生命週期，則使用者可根據目前固定在一小時的其工作階段 Cookie 持續時間，使用權杖。

11. 選擇 Generate client secret (產生用戶端密碼)，讓 Amazon Cognito 為您產生用戶端密碼。用戶端密碼通常與機密用戶端相關聯。
12. 選擇是否要為此應用程式用戶端 Enable token revocation (啟用權杖撤銷)。這將增加權杖的大小。如需詳細資訊，請參閱撤銷權杖。
13. 選擇是否要為此應用程式用戶端 Prevent error messages that reveal user existence (阻止顯示使用者存在的錯誤訊息)。Amazon Cognito 將回應不存在之使用者的登入請求，並顯示一則通用訊息，指出使用者名稱或密碼不正確。
14. (選用) 為此應用程式用戶端設定 Attribute read and write permissions (屬性讀取和寫入許可)。您的應用程式用戶端可以擁有讀取和寫入使用者集區屬性結構描述之有限子集的許可。
15. 選擇建立。
16. 請記下 Client id (用戶端 ID)。這會識別註冊和登入請求中的應用程式用戶端。

設定應用程式。

1. 在 App integration (應用程式整合) 標籤上，在 App clients (應用程式用戶端) 下選取您的應用程式用戶端。檢閱目前的 Hosted UI (託管 UI) 資訊。
2. 在 Allowed callback URL(s) (允許的回呼 URL) 下 Add a callback URL (新增回呼 URL)。回呼 URL 是使用者登入成功後，將重新引導至的位置。
3. 在 Allowed sign-out URL(s) (允許的登出 URL) 下 Add a sign-out URL (新增登出 URL)。登出 URL 是使用者在登出之後，將重新引導至的位置。
4. 從 Identity providers (身分提供者) 清單中新增至少一個列出的選項。
5. 在 OAuth 2.0 grant types (OAuth 2.0 授予類型) 下，選取 Authorization code grant (授權程式碼授予)，傳回使用者集區權杖交換的授權程式碼。由於權杖不會直接向最終使用者公開，因此他們不太可能遭到入侵。但自訂應用程式需由後端交換使用者集區權杖所需的授權程式碼。基於安全考量，強烈建議您的行動應用程式使用授權碼授予流程，搭配 代碼交換證明金鑰 (PKCE)。
6. 在 OAuth 2.0 grant types (OAuth 2.0 授予類型) 下，選取 Implicit grant (隱含授予)，讓使用者集區 JSON Web 權杖 (JWT) 從 Amazon Cognito 傳回給您。您可以使用此流程，但不提供後端為權杖交換授權程式碼。還可以協助您除錯權杖。
7. 您可以啟用 Authorization code (授權碼) 和 Implicit code (隱含程式碼)，並視需要使用每個授予。如果未選取 Authorization code (授權碼) 或 Implicit code (隱含程式碼)，並且您的應用程式用戶端擁有用戶端密碼，則您可以啟用 Client credentials (用戶端憑證) 授予。唯有當您的應用程式需要代表自己，而不是代表使用者請求存取權杖，才選取 Client credentials (用戶端憑證)。
8. 選取您想要為此應用程式用戶端授權的 OpenID Connect scopes (OpenID Connect 範圍)。

9. 選擇 Save changes (儲存變更)。

設定網域

1. 導覽至 App integration (應用程式整合) 索引標籤尋找使用者集區。
2. 在 Domain (網域) 旁，選擇 Actions (動作)，並選取 Create custom domain (建立自訂網域) 或者 Create Cognito domain (建立 Cognito 網域)。如果您已設定使用者集區網域，請選擇 Delete Cognito domain (刪除 Cognito 網域) 或者 Delete custom domain (刪除自訂網域)，然後再建立新的自訂網域。
3. 輸入可用的網域字首，以與 Cognito domain (Cognito 網域) 搭配使用。如需設定 Custom domain (自訂網域) 的詳細資訊，請參閱將自有網域用於託管 UI
4. 選擇建立。

檢視您的登入頁面

在 Amazon Cognito 主控台中，於 App integration (應用程式整合) 標籤的 App clients and analytics (應用程式用戶端和分析) 下，選取應用程式用戶端組態中的 View Hosted UI (檢視託管 UI) 按鈕。此按鈕會將您導向託管 UI 中的登入頁面，其中包含下列基本參數。

- 應用程式用戶端 ID
- 授權碼授與請求
- 對您為目前應用程式用戶端啟用的所有範圍的請求
- 目前應用程式用戶端清單中的第一個回呼 URL

當您要測試託管 UI 的基本功能時，View hosted UI (檢視託管 UI) 按鈕會很有用。您可以使用額外和已修改的參數自訂登入 URL。在大多數情況下，View hosted UI (檢視託管 UI) 連結的自動產生參數不會完全符合您的應用程式需求。在這些情況下，您必須自訂應用程式在登入使用者時叫用的 URL。如需登入參數機碼和值的詳細資訊，請參閱 使用者集區聯合端點和託管 UI 參考。

託管 UI 登入網頁使用下列 URL 格式。此範例透過 `response_type=code` 參數要求授權碼授予。

```
https://<your domain>/oauth2/authorize?response_type=code&client_id=<your app client
 id>&redirect_uri=<your callback url>
```

您可以從應用程式整合索引標籤擷取您的使用者集區網域字串。在同一個索引標籤中的應用程式用戶端和分析底下，可以找到應用程式用戶端 ID、其回呼 URL回呼 URL、允許的範圍及其他組態。

當您使用自訂參數導覽至 /oauth2/authorize 端點，Amazon Cognito 會將您重新導向到 /oauth2/login 端點，或者，如果您有 identity_provider 或 idp_identifier 參數，則會以無提示的方式將您重新導向至 IdP 登入頁面。如需略過託管 UI 的範例 URL，請參閱 Amazon Cognito 使用者集區中的 SAML 工作階段啟動。

隱含授予的託管 UI 請求範例

您可以使用以下 URL 檢視託管 UI 登入網頁，利用 response_type=token 授予隱含程式碼。成功登入後，Amazon Cognito 會將使用者集區權杖傳回至 Web 瀏覽器的網址列。

```
            https://mydomain.us-east-1.amazoncognito.com/authorize?
response_type=token&client_id=1example23456789&redirect_uri=https://
mydomain.example.com
```

身分和存取權杖會以參數形式顯示並附加至您的重新導向 URL。

以下是來自隱含授予請求的回應範例。

```
            https://mydomain.example.com/
#id_token=eyJraaBcDeF1234567890&access_token=eyJraGhIjKlM1112131415&expires_in=3600&token_type=
```

Amazon Cognito 使用者集區託管 UI 的相關須知

託管 UI 和確認使用者為管理員

對於使用者集區的本機使用者，當您將使用者集區設定為允許 Cognito 自動傳送訊息以進行驗證和確認時，託管 UI 效果最佳。啟用此設定時，Amazon Cognito 會將包含確認碼的訊息傳送給註冊的使用者。當您確認使用者為使用者集區管理員時，託管 UI 會在註冊後顯示錯誤訊息。在此狀態下，Amazon Cognito 已建立新使用者，但還無法傳送驗證訊息。您仍然可以確認使用者為管理員，但他們可能在遇到錯誤後聯絡您的支援人員。如需管理確認的詳細資訊，請參閱 允許使用者註冊您的應用程式，但以使用者集區管理員身分確認使用者。

檢視您對託管 UI 組態的變更

如果未立即顯示託管 UI 頁面的變更，請稍候幾分鐘，然後重新整理頁面。

解碼使用者集區權杖

Amazon Cognito 使用者集區權杖是使用 RS256 演算法簽署。您可以使用解碼和驗證使用者集區權杖 AWS Lambda，請參閱開啟解碼和驗證 Amazon Cognito JWT 權杖。 GitHub

託管的用戶界面和 TLS 版本

託管的 UI 需要在傳輸過程中加密。由 Amazon Cognito 提供的使用者集區網域需要最低 TLS 版本 1.2。自訂網域支援，但不需要 TLS 1.2 版。由於 Amazon Cognito 會管理託管 UI 和授權伺服器端點的組態，因此您無法修改使用者集區網域的 TLS 需求。

託管 UI 和 CORS 政策

Amazon Cognito 託管 UI 不支援自訂跨來源資源共享 (CORS) 原始政策。託管 UI 中的 CORS 政策會阻止使用者在其請求中傳遞驗證參數。相反，請在應用程式的 Web 前端實作 CORS 政策。Amazon Cognito 會將 `Access-Control-Allow-Origin: *` 回應標頭傳回至下列 OAuth 端點的請求。

1. 權杖端點
2. 撤銷端點
3. UserInfo 端點

託管用戶界面和授權服務器 cookie

Amazon Cognito 使用者集區端點會在使用者的瀏覽器中設定 Cookie。Cookie 符合某些網站未設置第三方 Cookie 的瀏覽器的要求。它們的範圍僅限於您的使用者集區端點，包括下列項目：

- 每個請求的 `XSRF-TOKEN` Cookie。
- 重新導向使用者時，工作階段一致性的 `csrf-state` Cookie。
- 會`cognito`話 cookie，可保留一個小時的成功登錄嘗試。

設定使用者集區網域

完成應用程式用戶端設定之後，您就可以為您的註冊和登入網頁設定地址。您可以使用 Amazon Cognito 託管網域，並選擇可用的網域字首，或者使用您自己的網址做為自訂網域。

若要使用 AWS Management Console 新增應用程式用戶端和 Amazon Cognito 託管網域，請參閱新增應用程式以啟用託管 Web UI。

> **Note**
>
> 您無法在網域字首使用文字 aws、amazon 或 cognito。

主題

將 Amazon Cognito 網域用於託管 UI

完成應用程式用戶端設定之後，您就可以為您的註冊和登入網頁設定地址。您可以搭配自己的網域字首來使用託管的 Amazon Cognito 網域。

> **Note**
>
> 為了增強 Amazon Cognito 應用程式的安全性，使用者集區端點的父網域會在公用尾碼清單 (PSL) 中註冊。PSL 可協助使用者的網頁瀏覽器對您的使用者集區端點及其設定的 Cookie 建立一致的了解。
> 使用者集區端點父網域採用下列格式。
>
> ```
> auth.Region.amazoncognito.com
> auth-fips.Region.amazoncognito.com
> ```

若要使用新增應用程式用戶端和 Amazon Cognito 託管的網域 AWS Management Console，請參閱建立應用程式用戶端。

主題

必要條件

開始之前，您需要：

- 搭配應用程式用戶端的使用者集區。如需詳細資訊，請參閱 使用者集區入門。

步驟 1：設定託管的使用者集區網域

設定託管的使用者集區網域

您可以使用或 AWS Management Console AWS CLI 或 API 來設定使用者集區網域。

Amazon Cognito console

設定網域

1. 導覽至 App integration (應用程式整合) 索引標籤尋找使用者集區。
2. 在網域旁，選擇動作，並選取建立自訂網域或建立 Amazon Cognito 網域。如果您已設定使用者集區網域，請選擇刪除 Amazon Cognito 網域或刪除自訂網域，然後再建立新的自訂網域。
3. 輸入可用的網域字首來搭配 Amazon Cognito 網域使用。如需設定 Custom domain (自訂網域) 的詳細資訊，請參閱將自有網域用於託管 UI
4. 選擇建立。

CLI/API

使用下列命令來建立網域字首，並將其指派給您的使用者集區。

設定使用者集區網域

- AWS CLI: `aws cognito-idp create-user-pool-domain`

  範例：`aws cognito-idp create-user-pool-domain --user-pool-id <user_pool_id> --domain <domain_name>`
- AWS API：CreateUserPoolDomain

取得網域的相關資訊

- AWS CLI: `aws cognito-idp describe-user-pool-domain`

  範例：`aws cognito-idp describe-user-pool-domain --domain <domain_name>`
- AWS API：DescribeUserPoolDomain

刪除網域

- AWS CLI: `aws cognito-idp delete-user-pool-domain`

  範例：`aws cognito-idp delete-user-pool-domain --domain <domain_name>`
- AWS API：DeleteUserPoolDomain

步驟 2：驗證您的登入頁面

- 驗證可從 Amazon Cognito 託管網域使用上述登入頁面。

```
https://<your_domain>/login?
response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
```

您的網域會顯示在 Amazon Cognito 主控台的 Domain name (網域名稱) 頁面中。您的應用程式用戶端 ID 和回呼 URL 會顯示在 App client settings (應用程式用戶端設定) 頁面中。

將自有網域用於託管 UI

設定應用程式用戶端之後，您可以設定使用者集區搭配自訂網域，以用於 Amazon Cognito 託管 UI 和身分驗證 API 端點。利用自訂網域，您就能讓使用者使用您的自有網址登入您的應用程式。

主題

- 將自訂網域新增到使用者集區
- 變更自訂網域所用的 SSL 憑證

將自訂網域新增到使用者集區

若要新增自訂網域到您的使用者集區，請由 Amazon Cognito 主控台指定網域名稱，並提供您透過 AWS Certificate Manager (ACM) 所管理的憑證。在您新增網域後，Amazon Cognito 將提供別名目標以讓您加入至您的 DNS 組態。

必要條件

開始之前，您需要：

- 搭配應用程式用戶端的使用者集區。如需詳細資訊，請參閱 使用者集區入門。

- 您擁有的 Web 網域。其父網域必須有一個有效的 DNS A 記錄。您可以為此記錄指派任何值。父網域可能是網域的根，也可以是網域階層中向上一層的子網域。例如，如果您的自訂網域是 auth.xyz.example.com，則 Amazon Cognito 必須能夠將 yz.example.com 解析至一個 IP 地址。為了避免意外影響客戶基礎設施，Amazon Cognito 不支援將頂層網域 (TLD) 用於自訂網域。如需詳細資訊，請參閱網域名稱。
- 能夠為您的自訂網域建立子網域。建議您使用 auth 做為子網域。例如：*auth.example.com*。

> **Note**
>
> 如果您沒有萬用字元憑證，那麼您可能需要為自訂網域的子網域取得新憑證。

- 由 ACM 管理的 Secure Sockets Layer (SSL) 憑證。

> **Note**
>
> 在申請或匯入憑證之前，您必須在 ACM 主控台中將 AWS 區域變更為美國東部 (維吉尼亞北部)。

- 允許您的使用者集區授權伺服器將 Cookie 新增至使用者工作階段的應用程式。Amazon Cognito 為託管的用戶界面設置了幾個必需的 cookie。其中包括 `cognito`、`cognito-fl` 和 `XSRF-TOKEN`。雖然每個個別 Cookie 都符合瀏覽器大小限制，但是對使用者集區設定的變更可能會導致託管 UI Cookie 的大小增加。自訂網域前面的中繼服務 (例如 Application Load Balancer (ALB) 可能會強制執行標頭大小上限或總 Cookie 大小。如果您的應用程式也設定了自己的 Cookie，使用者的工作階段可能會超過這些限制。我們建議您不要在託管 UI 子網域上設定 Cookie，以避免大小限制衝突。
- 更新 Amazon CloudFront 分發的許可。做法是將下列 IAM 政策陳述式連接至您 AWS 帳戶中的使用者：

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Sid": "AllowCloudFrontUpdateDistribution",
            "Effect": "Allow",
            "Action": [
                "cloudfront:updateDistribution"
            ],
            "Resource": [
                "*"
            ]
```

```
        }
    ]
}
```

如需有關授權動作的詳細資訊 CloudFront，請參閱〈使用以身分為基礎的政策 (IAM 政策)〉。CloudFront

Amazon Cognito 一開始會使用您的 IAM 許可來設定 CloudFront分發，但分配由 AWS管理。您無法變更 Amazon Cognito 與您的使用者集區相關聯的 CloudFront 分發組態。例如，您無法更新安全性政策中支援的 TLS 版本。

步驟 1：輸入您的自訂網域名稱

您可以使用 Amazon Cognito 主控台或 API，新增網域到您的使用者集區。

Amazon Cognito console

從 Amazon Cognito 主控台新增網域到您的使用者集區：

1. 登入 Amazon Cognito 主控台。若出現提示，請輸入 AWS 憑證。
2. 選擇 User pools (使用者集區)。
3. 選擇您要為其新增網域的使用者集區。
4. 選擇 App integration (應用程式整合) 索引標籤。
5. 在 Domain (網域) 旁，選擇 Actions (動作)，然後選擇 Create custom domain (建立自訂網域)。

   > **Note**
   >
   > 如果您已設定使用者集區網域，請選擇 Delete Cognito domain (刪除 Cognito 網域) 或者 Delete custom domain (刪除自訂網域)，以在建立新的自訂網域前刪除現有網域。

6. 對於 Custom domain (自訂網域)，輸入您要與 Amazon Cognito 搭配使用之網域的 URL。網域名稱只能包含小寫字母、數字和連字號。第一個字元或最後一個字元切勿使用連字號。使用句號分隔子網域名稱。
7. 對於 ACM certificate (ACM 憑證)，選擇您要用於此網域的 SSL 憑證。只有美國東部 (維吉尼亞北部) 的 ACM 憑證才能與 Amazon Cognito 自訂網域搭配使用，無論您 AWS 區域 的使用者集區為何。

如果您沒有可用的憑證，則可使用 ACM 在美國東部 (維吉尼亞北部) 佈建一個憑證。如需詳細資訊，請參閱 AWS Certificate Manager 使用者指南中的入門。

8. 選擇建立。
9. Amazon Cognito 會讓您返回 App integration (應用程式整合) 索引標籤。系統會顯示標題為 Create an alias record in your domain's DNS (在您網域的 DNS 中建立別名記錄) 的訊息。請記下主控台中顯示的 Domain (網域) 和 Alias target (別名目標)。會在下一個步驟中使用它們，以將流量導向您的自訂網域。

API

使用 Amazon Cognito API 新增網域到您的使用者集區

- 使用 CreateUserPoolDomain 動作。

步驟 2：新增別名目標和子網域

在此步驟中，您會透過網域名稱伺服器 (DNS) 服務供應商，設定可指回在先前步驟所指定別名目標的別名。如果是使用 Amazon Route 53 進行 DNS 位址解析，請選擇使用 Route 53 新增別名目標和子網域一節。

將別名目標和子網域新增到目前的 DNS 組態

- 如果不是使用 Route 53 進行 DNS 地址解析，則您必須使用 DNS 服務供應商的組態工具，將先前步驟中的別名目標新增為網域的 DNS 記錄。您的 DNS 供應商也必須為您的自訂網域設定子網域。

使用 Route 53 新增別名目標和子網域

1. 登入 Route 53 主控台。若出現提示，請輸入 AWS 憑證。
2. 如果您在 Route 53 中沒有託管區域，請創建一個具有根的根目錄，該根是您的自定義域的父代。如需詳細資訊，請參閱

   a. 選擇建立託管區域。

   b. 輸入您自訂網域的上層網域，例如 *auth.example.com*，來自 Domain Name (網域名稱) 清單的網域，例如 *myapp.auth.example.com*。

   c. 輸入託管區域的說明。

d. 選擇 Public hosted zone (公有託管區域) 的託管區域 Type (類型)，以允許公有用戶端解析您的自訂網域。不支援選擇 Private hosted zone (私有託管區域)。

e. 根據需要套用 Tags (標籤)。

f. 選擇建立託管區域。

> **Note**
>
> 您也可以為自訂網域建立新的託管區域，可以在上層託管區域中建立委派集，上層託管區域會將查詢導向子網域託管區域。否則，請建立 A 記錄。此方法為您的託管區域提供更多靈活性與安全性。如需詳細資訊，請參閱為透過 Amazon Route 53 託管的網域建立子網域。

3. 在 Hosted Zones (託管區域) 頁面上，選擇託管區域的名稱。

4. 如果您還沒有自訂網域的上層網域，請新增 DNS 記錄。新增上層網域的 DNS A 記錄，然後選擇 [建立記錄]。以下是網域 *auth.example.com* 的範例記錄。

```
auth.example.com. 60 IN A 198.51.100.1
```

> **Note**
>
> Amazon Cognito 會驗證您自訂網域的上層網域是否有 DNS 記錄，以防止生產網域遭到意外劫持。如果您沒有上層網域的 DNS 記錄，則 Amazon Cognito 會在您嘗試設定自訂網域時傳回錯誤。授權開始 (SOA) 記錄不足以用於父系網域驗證的 DNS 記錄。

5. 為您的自訂網域新增 DNS 記錄。例如，您記錄必須指向自訂網域別名目標`123example.cloudfront.net`。再次選擇 Create record (建立記錄)。

6. 輸入與您的自訂網域相符的 Record name (記錄名稱)，例如 *myapp* 來建立 *myapp.auth.example.com* 的記錄。

7. 啟用 Alias (別名) 選項。

8. 在 Route traffic to (將流量路由至) 下，選擇 Alias to CloudFront distribution (CloudFront 分佈的別名)。輸入您建立自訂網域時 Amazon Cognito 提供的 Alias target (別名目標)。

9. 選擇 Create Records (建立記錄)。

> **Note**
>
> 您的新記錄可能需要大約 60 秒才能傳播到所有 Route 53 DNS 伺服器。您可以使用 Route 53 GetChangeAPI 方法來驗證您的變更是否已傳播。

步驟 3：驗證您的登入頁面

- 驗證可從自訂網域使用上述登入頁面。

  在瀏覽器中輸入這個地址後，登入您的自訂網域和子網域。下面是自訂網域 *example.com*、子網域 *auth* 的範例 URL：

  ```
  https://myapp.auth.example.com/login?
  response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
  ```

變更自訂網域所用的 SSL 憑證

必要時，您可以使用 Amazon Cognito 變更自訂網域所要套用的憑證。

通常，若是由 ACM 例行憑證續約便不需要這麼做。當您在 ACM 續約現有的憑證時，該憑證的 ARN 將保持不變，而且您的自訂網域會自動使用新的憑證。

不過，若您使用新的憑證取代現有的憑證，則 ACM 將為新的憑證指派新的 ARN。若要將新憑證套用到您的自訂網域，您必須向 Amazon Cognito 提供該 ARN。

在您提供新憑證之後，Amazon Cognito 需歷經 1 小時才能將其分發到您的自訂網域。

> **開始之前**
>
> 若要在 Amazon Cognito 中變更您的憑證，您必須先將該憑證新增至 ACM。如需詳細資訊，請參閱 AWS Certificate Manager 使用者指南中的入門。
> 將憑證新增至 ACM 時，您必須選擇美國東部 (維吉尼亞北部) 當作 AWS 區域。

您可以使用 Amazon Cognito 主控台或 API 變更您的憑證。

AWS Management Console

從 Amazon Cognito 主控台續約憑證：

1. 登入 AWS Management Console 並開啟 Amazon Cognito 主控台，位於https://console.aws.amazon.com/cognito/home。
2. 選擇 User Pools (使用者集區)。
3. 選擇您要為其更新憑證的使用者集區。
4. 選擇 App integration (應用程式整合) 索引標籤。
5. 選擇 Actions (動作)、Edit ACM certificate (編輯 ACM 憑證)。
6. 選取您要與自訂網域產生關聯的新憑證。
7. 選擇 Save changes (儲存變更)。

API

續約憑證 (Amazon Cognito API)

- 使用 UpdateUserPoolDomain 動作。

自訂內建的註冊與登入網頁

您可以使用 AWS Management Console (或是 AWS CLI 或 API) 來指定內建應用程式 UI 體驗的自訂設定。您可以上傳要顯示在應用程式中的自訂標誌影像。您也可以使用階層式樣式表 (CSS) 來自訂 UI 的外觀。

您可以為單一用戶端 (使用特定 `clientId`) 或為所有用戶端 (將 `clientId` 設定為 `ALL`) 指定應用程式使用者界面自訂設定。如果您指定 `ALL`，則會將預設組態用於先前未設定使用者界面自訂的每個用戶端。如果您為特定用戶端指定使用者界面自訂設定，則不會再回復到 `ALL`組態。

設定 UI 自訂的請求大小不得超過 135 KB。在極少數情況下，請求標頭、CSS 檔案和標誌的總和可能超過 135KB。Amazon Cognito 會將影像檔案編碼為 Base64。這會將 100 KB 影像的大小增加到 130 KB，使得請求標頭和 CSS 僅保留五 KB 可用。如果請求太大，AWS Management Console 或您的 `SetUICustomization` API 請求會傳回 `request parameters too large` 錯誤。請將您的標誌影像調整為不超過 100KB，並將 CSS 檔案調整為不超過 3 KB。您不能單獨設定 CSS 和標誌自訂。

> **Note**
>
> 如果要自訂 UI，則您必須為使用者集區設定網域。

為應用程式指定自訂標誌

Amazon Cognito 會將您的自訂標誌置中於 登入端點 輸入欄位的上方。

為您的自訂託管 UI 標誌選擇可縮放為 350 x 178 像素的 PNG、JPG 或 JPEG 檔案。您的標誌檔案大小不得超過 100 KB，或在 Amazon Cognito 編碼為 Base64 之後的大小不得超過 130 KB。若要在 API 的 SetUICustomization 中設定 `ImageFile`，請將檔案轉換為 Base64 編碼的文字字串，或者在 AWS CLI 中提供檔案路徑，然後讓 Amazon Cognito 為您編碼。

為應用程式指定 CSS 自訂項目

您可以為託管的應用程式頁面自訂 CSS，但有下列限制：

- 您可以使用以下任一 CSS 類別名稱：
  - `background-customizable`
  - `banner-customizable`
  - `errorMessage-customizable`
  - `idpButton-customizable`
  - `idpButton-customizable:hover`
  - `idpDescription-customizable`
  - `inputField-customizable`
  - `inputField-customizable:focus`
  - `label-customizable`
  - `legalText-customizable`
  - `logo-customizable`
  - `passwordCheck-valid-customizable`
  - `passwordCheck-notValid-customizable`
  - `redirect-customizable`
  - `socialButton-customizable`
  - `submitButton-customizable`

  - `submitButton-customizable:hover`
  - `textDescription-customizable`
- 屬性值可以包含 HTML，但下列值除外：`@import`、`@supports`、`@page` 或 `@media` 陳述式或 Javascript。

您可以自訂下列 CSS 屬性。

標籤

- font-weight 是 100 的倍數，從 100 到 900。

輸入欄位

- width (寬度) 是以包含區塊的百分比表示的寬度。
- height 是輸入欄位的高度，以像素 (px) 為單位。
- color 是文字顏色，可以是任何標準 CSS 顏色值。
- background-color 是輸入欄位的背景顏色，可以是任何標準 CSS 顏色值。
- border 是標準 CSS 框線值，用以指定應用程式視窗框線的寬度、透明度和顏色。寬度可以是從 1px 到 100px 的任何值。透明度可以是純色或無。顏色可以是任何標準顏色值。

文字描述

- padding-top 是文字描述與上邊框的距離。
- padding-bottom 是文字描述與下邊框的距離。
- display 可以是 `block` 或 `inline`。
- font-size 是文字描述的字型大小。

提交按鈕

- font-size 是按鈕文字的字型大小。
- font-weight 是按鈕文字的字型粗細：`bold`、`italic` 或 `normal`。
- margin 是包含四個值的字串，指出按鈕的上、右、下和左邊界大小。
- font-size 是文字描述的字型大小。
- width 是按鈕文字的寬度，以包含區塊的百分比表示。
- height 是按鈕的高度，以像素 (px) 為單位。
- color 是按鈕文字的顏色，可以是任何標準 CSS 顏色值。
- background-color 是按鈕的背景顏色，可以是任何標準顏色值。

橫幅

- padding 是包含四個值的字串，指出橫幅與上、右、下和左邊框的距離大小。
- background-color 是橫幅的背景顏色，可以是任何標準 CSS 顏色值。

提交按鈕滑鼠游標暫留

- color 是您將滑鼠游標移到按鈕上時，按鈕呈現的前景顏色。可以是任何標準 CSS 顏色值。
- background-color 是您將滑鼠游標移到按鈕上時，按鈕呈現的背景顏色。可以是任何標準 CSS 顏色值。

身分提供者按鈕滑鼠游標暫留

- color 是您將滑鼠游標移到按鈕上時，按鈕呈現的前景顏色。可以是任何標準 CSS 顏色值。
- background-color 是您將滑鼠游標移到按鈕上時，按鈕呈現的背景顏色。可以是任何標準 CSS 顏色值。

密碼檢查無效

- color 是 `"Password check not valid"` 訊息的文字顏色，可以是任何標準 CSS 顏色值。

背景介紹

- background-color 是應用程式視窗的背景顏色，可以是任何標準 CSS 顏色值。

錯誤訊息

- margin 是包含四個值的字串，指出上、右、下和左邊界大小。
- padding 是與邊框的距離大小。
- font-size 是字型大小。
- width 是錯誤訊息的寬度，以包含區塊的百分比表示。
- background 是錯誤訊息的背景顏色，可以是任何標準 CSS 顏色值。
- border 是包含三個值的字串，用以指定框線的寬度、透明度和顏色。
- color 是錯誤訊息的文字顏色，可以是任何標準 CSS 顏色值。
- box-sizing 可用來向瀏覽器指出大小屬性 (寬度和高度) 應該包含哪些項目。

身分提供者按鈕

- height 是按鈕的高度，以像素 (px) 為單位。
- width 是按鈕文字的寬度，以包含區塊的百分比表示。
- text-align 是文字對齊方式設定。其可能是 `left`、`right` 或 `center`。
- margin-bottom 是底部邊界設定。
- color 是按鈕文字的顏色，可以是任何標準 CSS 顏色值。

- background-color 是按鈕的背景顏色，可以是任何標準 CSS 顏色值。
- border-color 是按鈕框線的顏色，可以是任何標準 CSS 顏色值。

身分提供者描述

- padding-top 是描述與上邊框的距離。
- padding-bottom 是描述與下邊框的距離。
- display 可以是 `block` 或 `inline`。
- font-size 是描述的字型大小。

法律文字

- color 是文字顏色，可以是任何標準 CSS 顏色值。
- font-size 是字型大小。

> **Note**
>
> 自訂 Legal text (法規文字) 時，您正在自訂會在登入頁面社交身分提供者下顯示的訊息 We won't post to any of your accounts without asking first (我們不會在沒有事先詢問的情況下發佈到您的任何帳戶)。

標誌

- max-width 是寬度上限，以包含區塊的百分比表示。
- max-height 是高度上限，以包含區塊的百分比表示。

輸入欄位焦點

- border-color 是輸入欄位的顏色，可以是任何標準 CSS 顏色值。
- outline 是輸入欄位的框線寬度，以像素為單位。

社交按鈕

- height 是按鈕的高度，以像素 (px) 為單位。
- text-align 是文字對齊方式設定。其可能是 `left`、`right` 或 `center`。
- width 是按鈕文字的寬度，以包含區塊的百分比表示。
- margin-bottom 是底部邊界設定。

密碼檢查有效

- color 是 `"Password check valid"` 訊息的文字顏色，可以是任何標準 CSS 顏色值。

為使用者集區指定應用程式使用者界面自訂設定 (AWS Management Console)

您可以使用 AWS Management Console來為您的應用程式指定使用者界面自訂設定。

> **Note**
>
> 您可以利用使用者集區的詳細規格來建構下列 URL，並將其輸入瀏覽器中，以檢視採用您的自訂項目的託管使用者界面： `https://<your_domain>/login?response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callbac`
> 您可能需要等到一分鐘後再重新整理瀏覽器，在主控台中所做的變更才會出現。
> 您的網域會顯示在 Domain (網域) 下的 App integration (應用程式整合) 索引標籤上。您的應用程式用戶端 ID 和回呼 URL 會顯示在 App clients (應用程式用戶端) 下。

指定應用程式使用者界面自訂設定

1. 登入 Amazon Cognito 主控台。
2. 在導覽窗格中，選擇 User Pools (使用者集區)，然後選擇您要編輯的使用者集區。
3. 選擇 App integration (應用程式整合) 索引標籤。
4. 若要自訂所有應用程式用戶端的 UI 設定，請找到 Hosted UI customization (託管 UI 自訂)，然後選取 Edit (編輯)。
5. 若要自訂一個應用程式用戶端的 UI 設定，請找到 應用程式用戶端 並選取您要修改的應用程式用戶端，然後找到 託管 UI 自訂，然後選取 編輯。若要將應用程式用戶端從使用者集區預設自訂切換為用戶端特定自訂，則請選取 Use client-level settings (使用用戶端層級設定)。
6. 若要上傳您自己的標誌影像檔案，請選擇 Choose file (選擇檔案) 或者 Replace current file (取代目前的檔案)。
7. 若要自訂託管 UI CSS，請下載 CSS template.css 並使用您想要自訂的值修改範本。只有包含在範本中的金鑰可以與託管 UI 搭配使用。新增的 CSS 金鑰不會反映在您的 UI 中。自訂 CSS 檔案之後，請選擇 Choose file (選擇檔案) 或者 Replace current file (取代目前的檔案) 來上傳您的自訂 CSS 檔案。

為使用者集區指定應用程式 UI 自訂設定 (AWS CLI 和 AWS API)

使用下列命令來為您的使用者集區指定應用程式使用者界面自訂設定。

若要為使用者集區的內建應用程式 UI 取得 UI 自訂設定，請使用下列的 API 操作。

- AWS CLI: `aws cognito-idp get-ui-customization`
- AWS API：GetUICustomization

若要為使用者集區的內建應用程式 UI 進行 UI 自訂設定，請使用下列的 API 操作。

- AWS CLI 從影像檔案：`aws cognito-idp set-ui-customization --user-pool-id <your-user-pool-id> --client-id <your-app-client-id> --image-file fileb://"<path-to-logo-image-file>" --css ".label-customizable{ color: <color>;}"`
- AWS CLI，影像編碼為 Base64 二進制文字：`aws cognito-idp set-ui-customization --user-pool-id <your-user-pool-id> --client-id <your-app-client-id> --image-file <base64-encoded-image-file> --css ".label-customizable{ color: <color>;}"`
- AWS API：SetUICustomization

使用託管 UI 註冊和登入

為使用者集區和應用程式用戶端設定和自訂 Amazon Cognito 託管 UI 後，您的應用程式就可以將其顯示給使用者。託管 UI 支援多種 Amazon Cognito 驗證操作，包括下列範例。

- 在應用程式中註冊為新使用者
- 驗證電子郵件地址或電話號碼
- 設定多重要素驗證 (MFA)
- 以本機使用者名稱和密碼登入
- 使用第三方身分提供者 (IdP) 登入
- 重設密碼

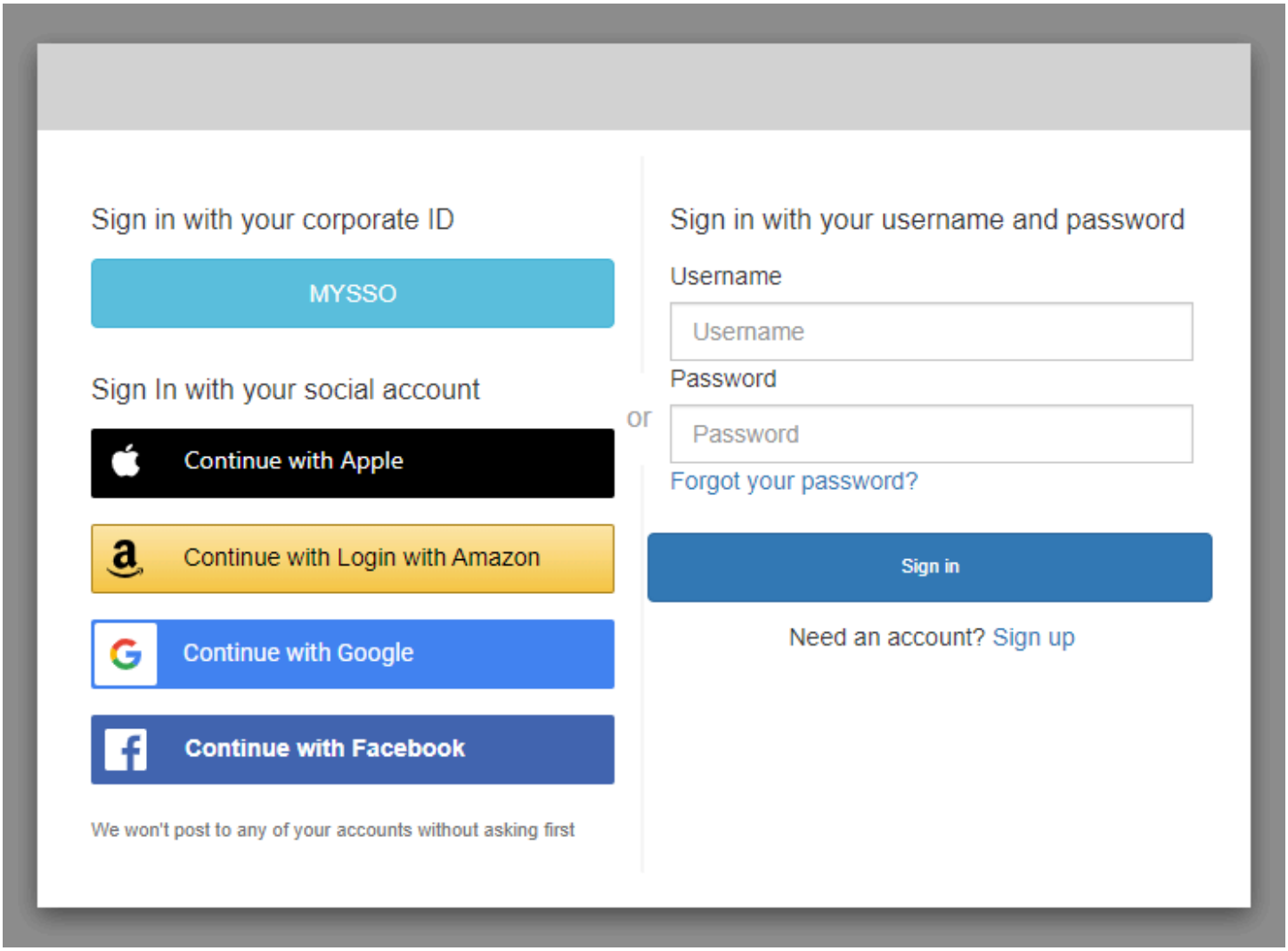
Amazon Cognito 託管 UI 從登入端點開始。登入頁面的 URL 包含了您為使用者集區選擇的網域，以及反映您要發布之 OAuth 2.0 授與的參數、您的應用程式用戶端、應用程式的路徑以及要請求的 OpenID Connect (OIDC) 範圍。

```
https://<your user pool domain>/authorize?client_id=<your app client
 ID>&response_type=<code/token>&scope=<scopes to request>&redirect_uri=<your callback
 URL>
```

下列 URL 會以範例值取代上述預留位置欄位。

```
https://auth.example.com/authorize? /
client_id=1example23456789 /
&response_type=code /
&scope=aws.cognito.signin.user.admin+email+openid+profile /
&redirect_uri=https%3A%2F%2Faws.amazon.com
```

Amazon Cognito 託管 UI 的登入頁面提供選項，可透過使用者集區或您指派給使用者要求之應用程式用戶端的任何身分提供者 (IdP) 登入。它也包含在使用者集區中註冊新使用者帳戶或重設忘記密碼的連結。



主題

如何在 Amazon Cognito 託管 UI 中註冊新帳戶

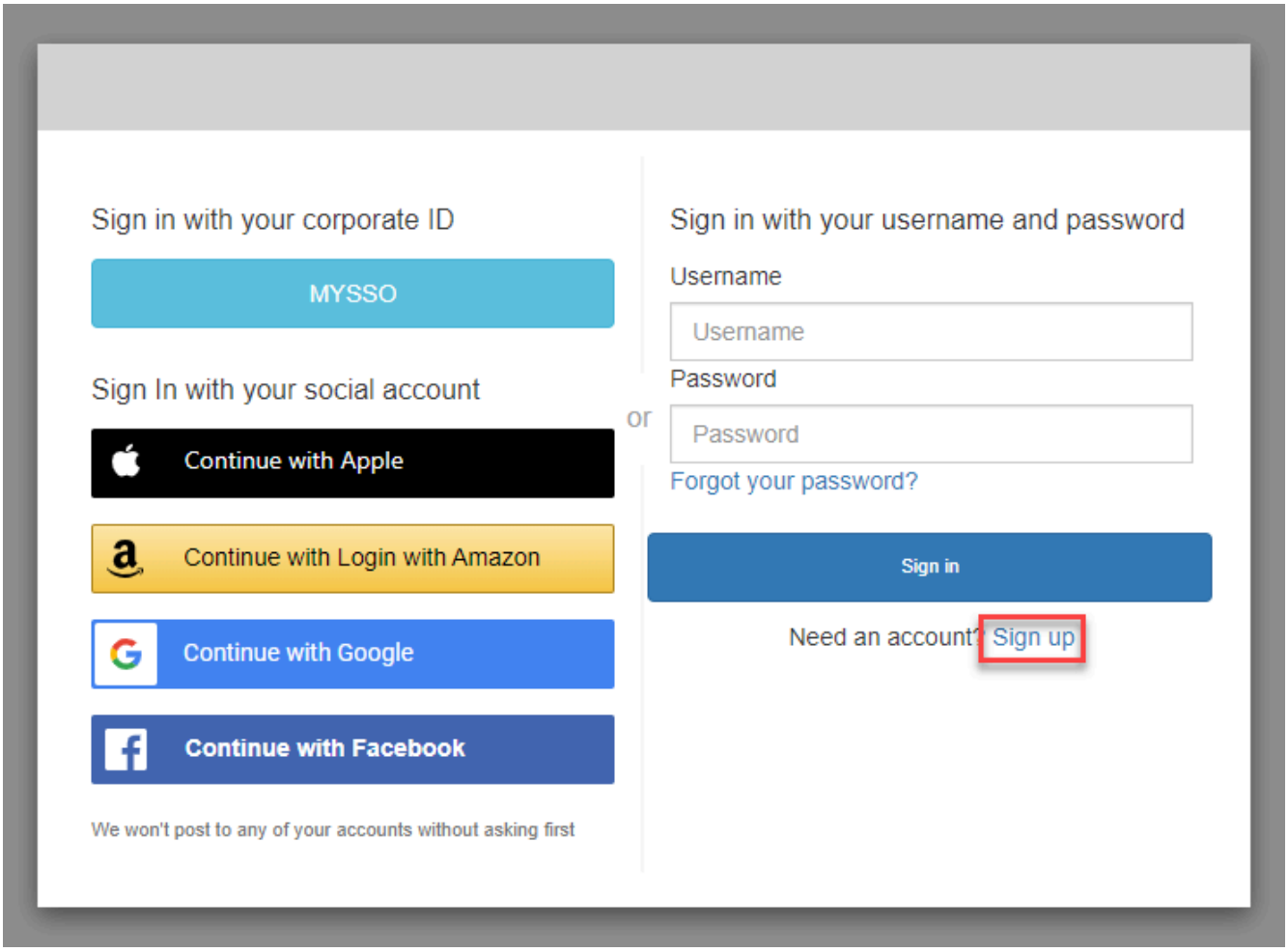
本指南說明如何在使用 Amazon Cognito 的應用程式中註冊使用者帳戶。

> **Note**
>
> 當您登入使用 Amazon Cognito 託管使用者介面 (UI) 的應用程式時，除了本指南所示的基本組態之外，您可能還會看到應用程式擁有者已自訂的頁面。

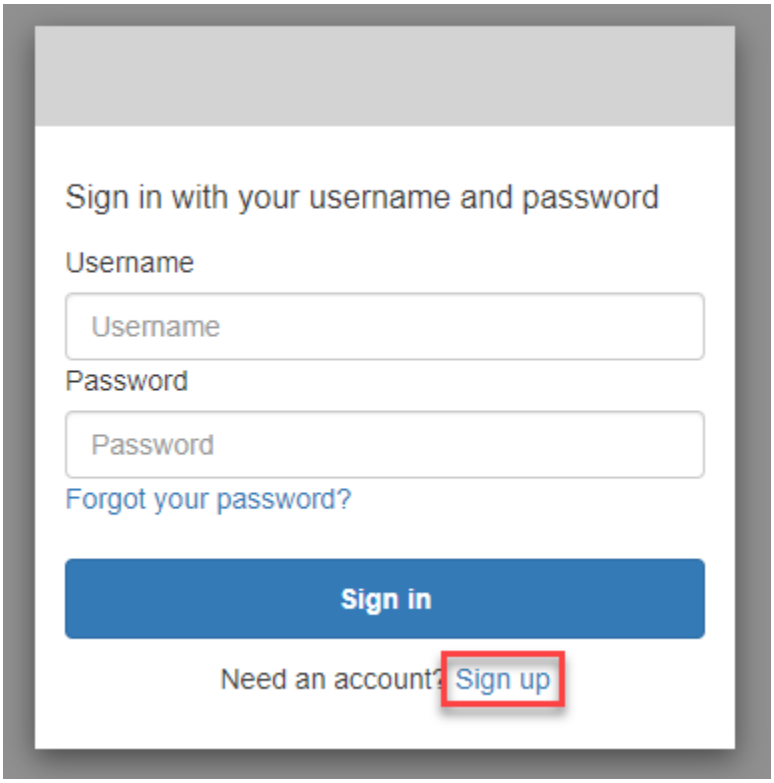
1. 如果您打算使用使用者名稱和密碼透過 Amazon Cognito 登入，而不是使用應用程式擁有者列出的其中一個第三方登入供應商，請從登入頁面選擇 Sign up (註冊)。

   如果您的登入供應商不是 Amazon Cognito，則在您選擇第三方供應商的按鈕之後，就會完成註冊。視應用程式擁有者選擇的選項而定，您可能可以選擇登入時要使用的供應商，或者您可能只會看到輸入使用者名稱和密碼的提示。

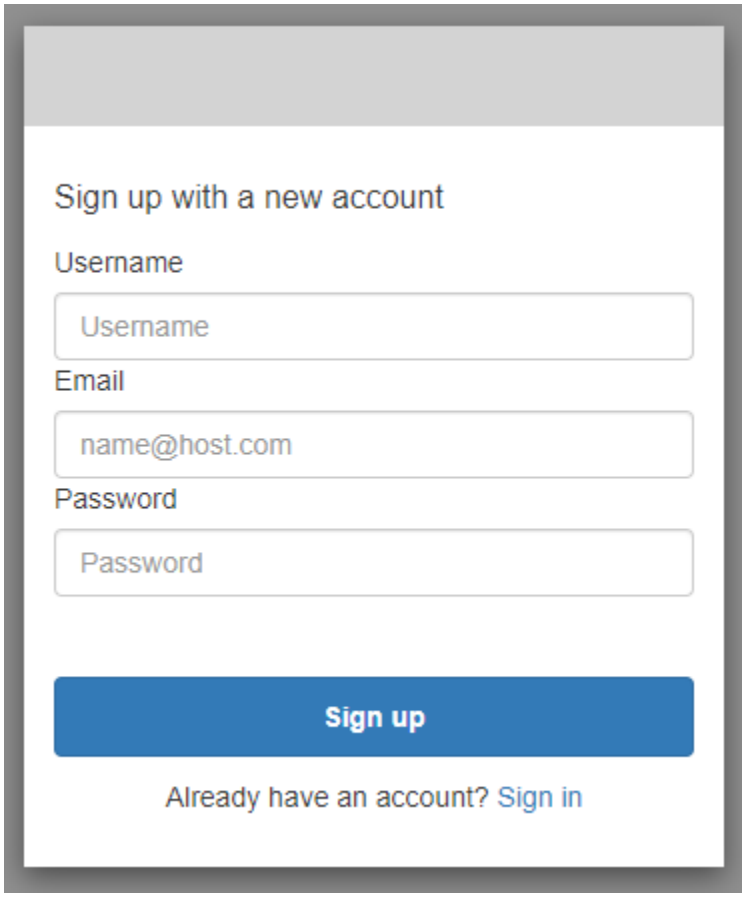
With multiple sign-in providers



With only Amazon Cognito as a sign-in provider

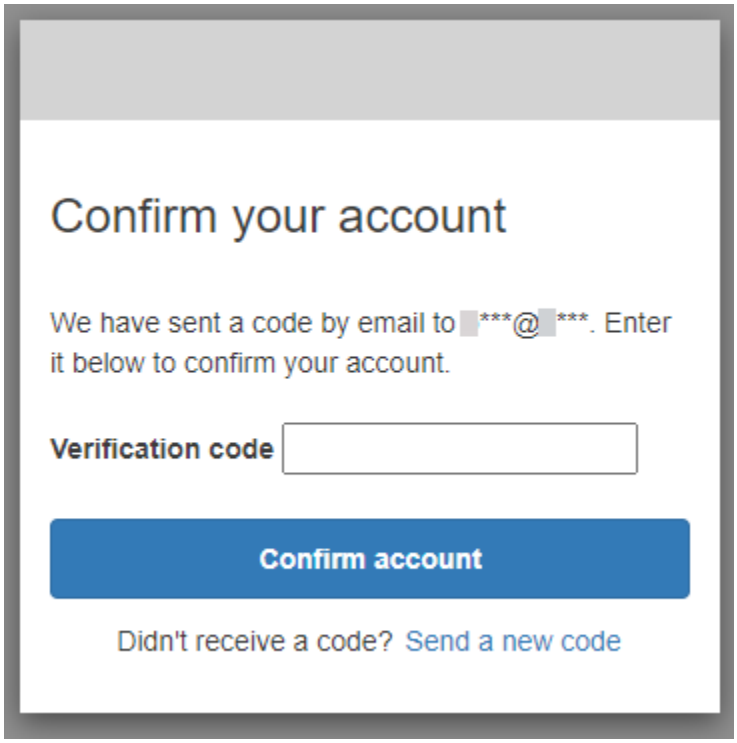


2. 在 Sign up with a new account (使用新帳戶註冊) 頁面中，應用程式擁有者會要求他們提供註冊所需的資訊。他們可能會要求提供使用者名稱、電子郵件地址或電話號碼。輸入必要的資訊然後選擇密碼。



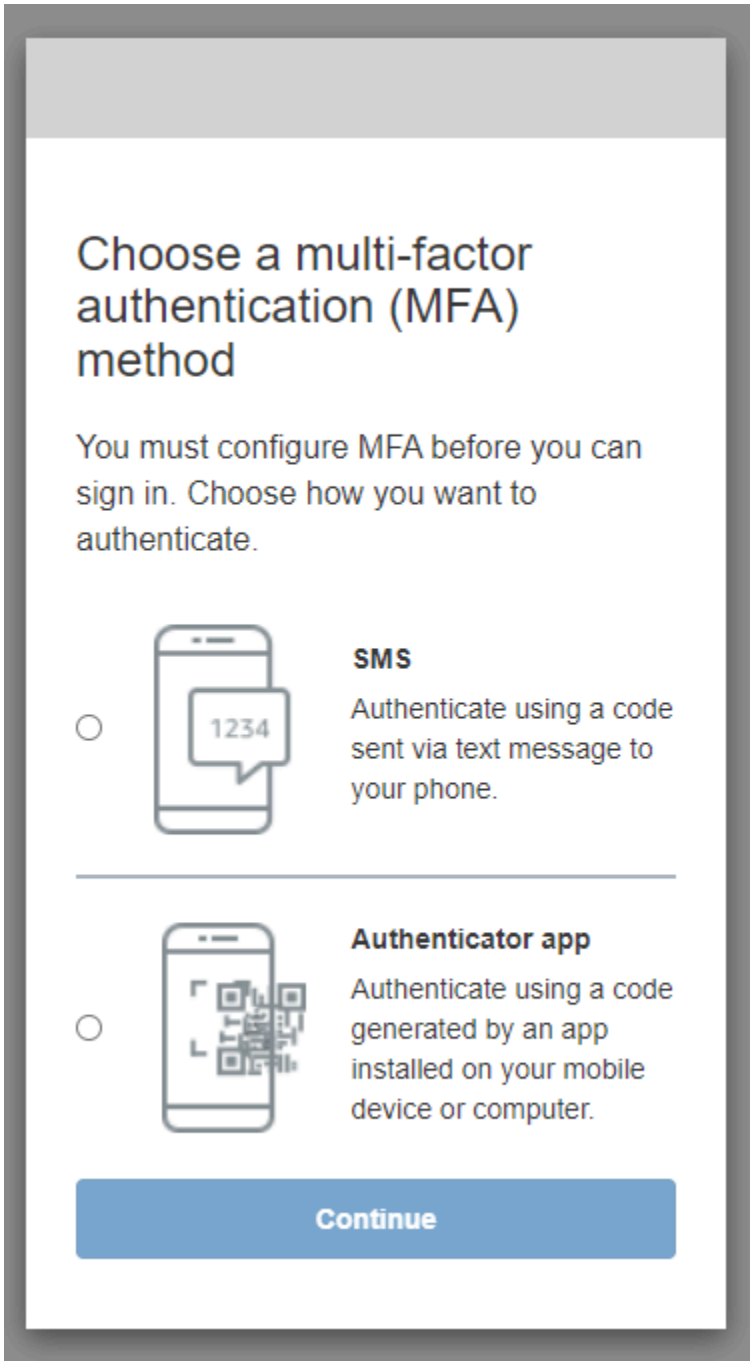
3. 在 Confirm your account (確認您的帳戶) 頁面上，應用程式擁有者可能會要求您確認帳戶，以確認您可以透過您提供的電子郵件地址或電話號碼接收訊息。

   您會在電子郵件或簡訊中收到驗證碼。在表格中輸入驗證碼，以確認您輸入的聯絡資料正確無誤。



4. 應用程式擁有者可能會要求您設定多重要素驗證 (MFA)。您可能會看到要求選擇 MFA 方法的提示，否則您的應用程式會跳到下一個步驟。

   在 Choose a multi-factor authentication (MFA) method (選擇多重要素驗證 (MFA) 方法) 頁面上，選擇 MFA 方法。如果選擇 SMS (簡訊)，您將在簡訊中收到 MFA 密碼。如果選擇 Authenticator app (驗證器應用程式)，您必須在裝置上安裝應用程式，才能產生時間式 MFA 密碼。您必須在 3 分鐘內作出選擇。



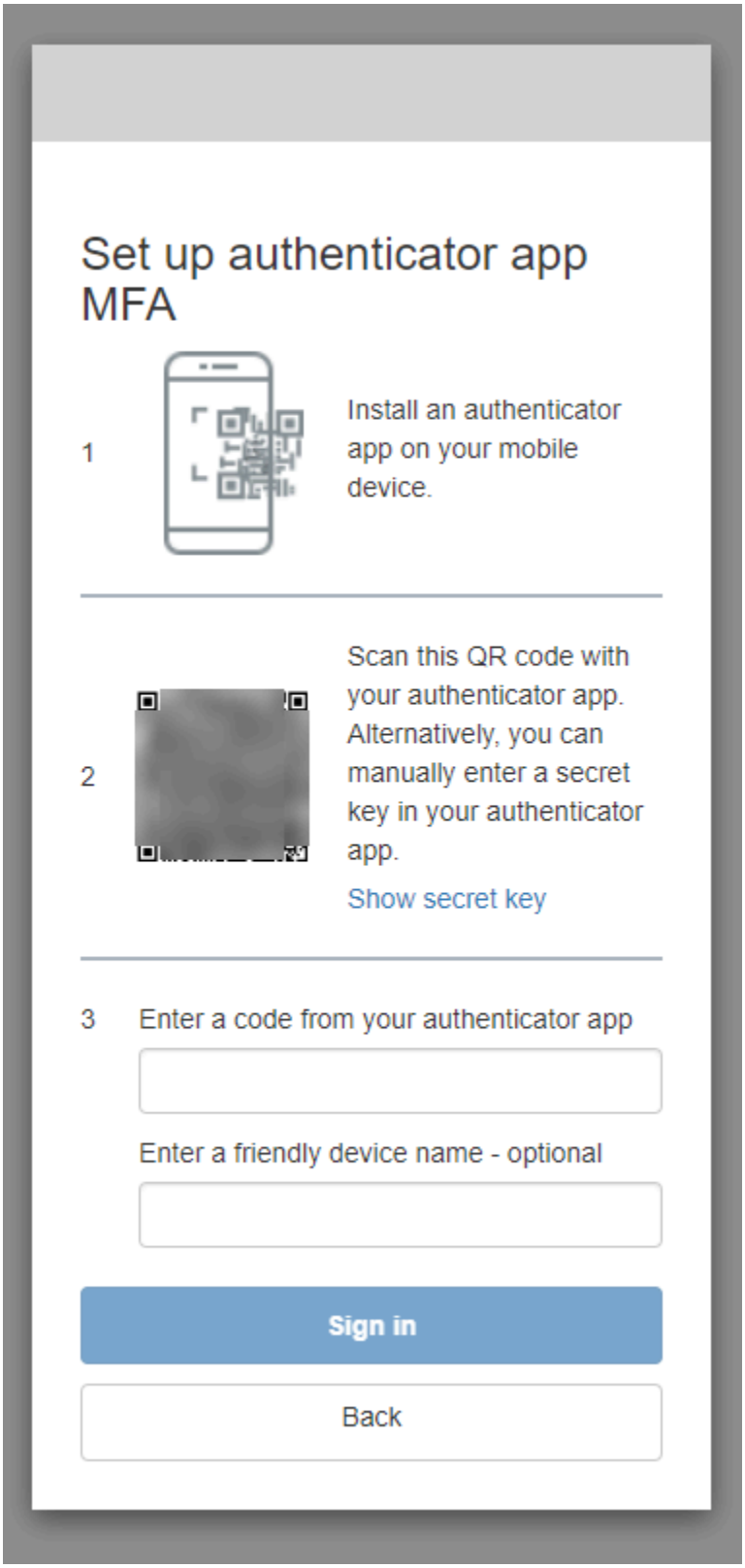
5. Amazon Cognito 會要求您提供來自驗證器應用程式或簡訊的驗證碼。在 3 分鐘內輸入您收到的代碼。

   Authenticator app

   1. 開啟您下載的驗證器應用程式。
   2. 使用相機掃描頁面上的 QR 碼。您可能需要授權應用程式才能使用您的相機。

如果無法掃描 QR 碼，請選擇 Show secret key (顯示私密金鑰)，顯示可讓您手動輸入至驗證器應用程式的驗證碼。

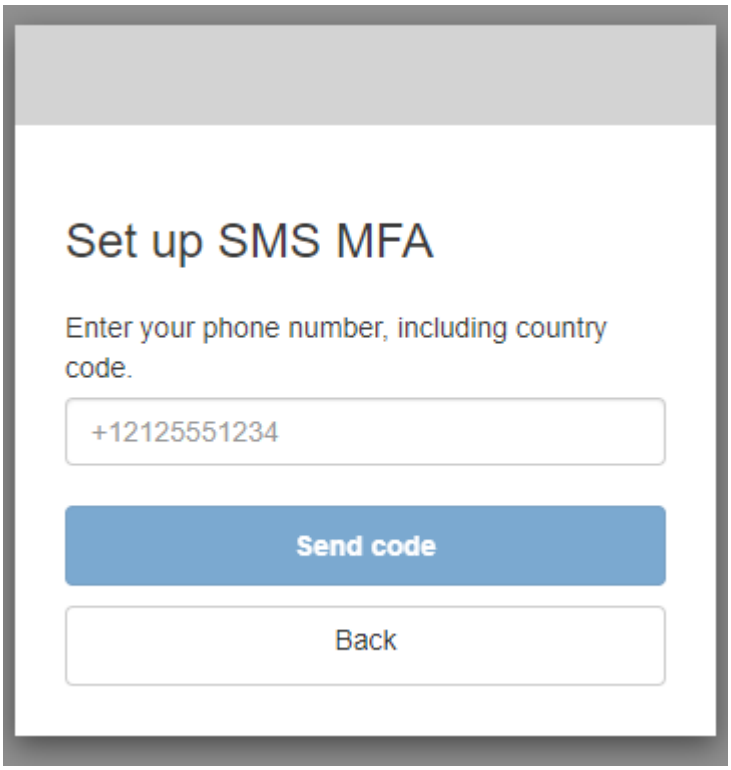
3. 驗證器應用程式會開始顯示驗證碼，每隔幾秒鐘變更一次。輸入應用程式中目前的驗證碼。
4. (選用) 在 Set up authenticator app MFA (設定驗證器應用程式 MFA) 頁面上，選擇裝置的名稱。登入時，Amazon Cognito 會要求您輸入您在此處所提供名稱之裝置上的驗證碼。


Set up authenticator app MFA
1
Install an authenticator app on your mobile device.
2
Scan this QR code with your authenticator app. Alternatively, you can manually enter a secret key in your authenticator app.
Show secret key
3
Enter a code from your authenticator app
Enter a friendly device name - optional
Sign in
Back

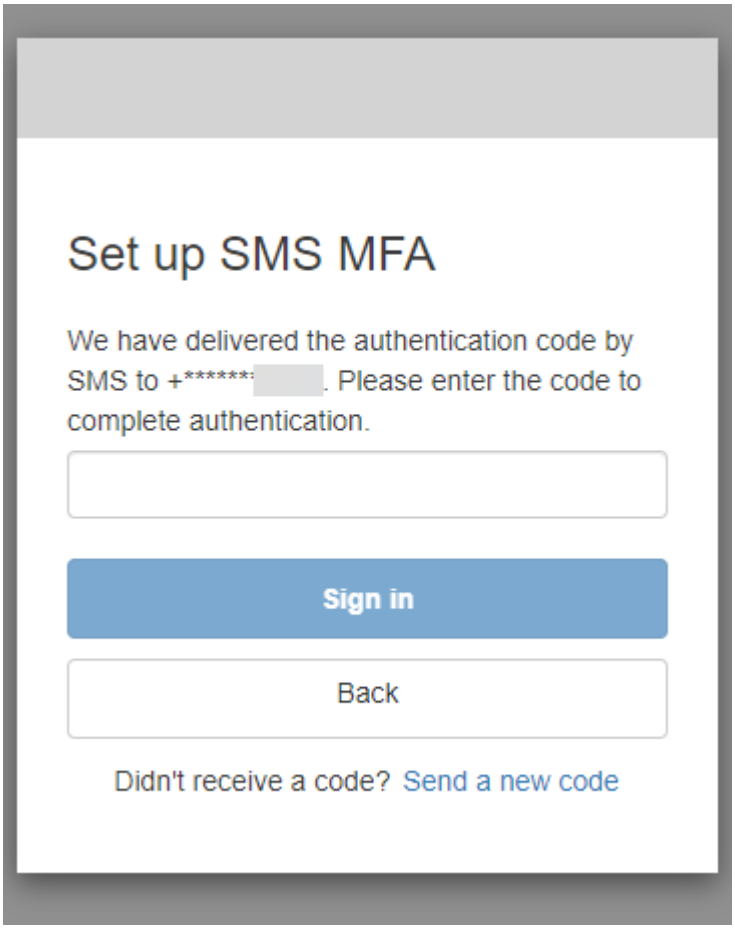
SMS text message

1. 如果應用程式擁有者尚未收集您的電話號碼，Amazon Cognito 會要求您提供電話號碼。

   在 Set up SMS MFA (設定簡訊 MFA) 頁面中，輸入電話號碼，包含 + 符號和國碼，例如 +12125551234。



2. 您會收到包含驗證碼的簡訊。在 Set up SMS MFA (設定簡訊 MFA) 頁面上，輸入驗證碼。如果您沒有收到驗證碼，並想重試，請選擇 Send a new code (傳送新的驗證碼)。選取 Back (返回) 以輸入新的電話號碼。



6. 第一次註冊並確認詳細資料時，Amazon Cognito 會在您完成此程序後授與對應用程式的存取權。

如何使用 Amazon Cognito 託管 UI 登入

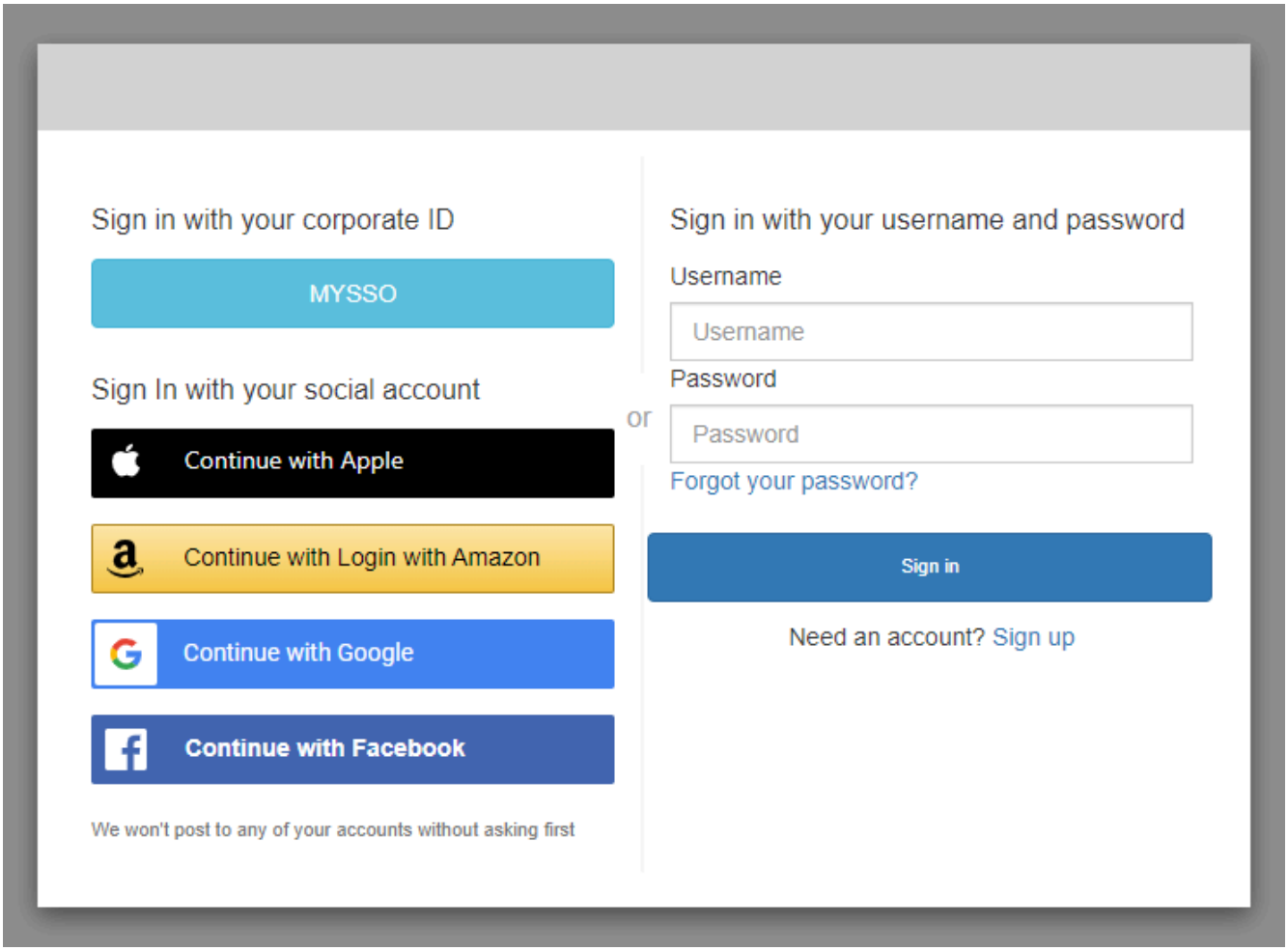
本指南說明如何登入使用 Amazon Cognito 的應用程式。

> **Note**
>
> 當您登入使用 Amazon Cognito 託管使用者介面 (UI) 的應用程式時，除了本指南所示的基本組態之外，您可能還會看到應用程式擁有者已自訂的頁面。

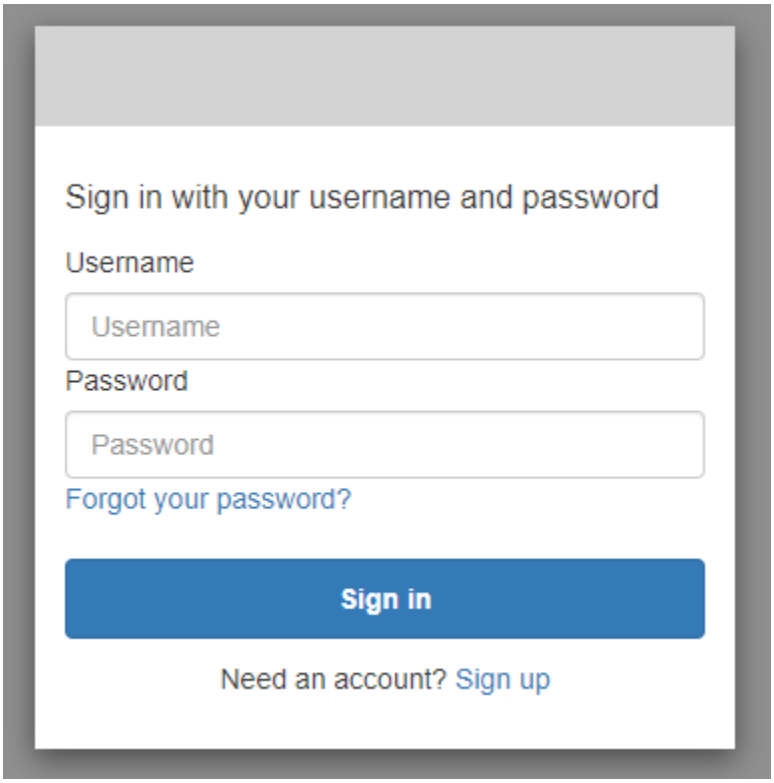
1. 視應用程式擁有者選擇的選項而定，您可能可以選擇登入時要使用的供應商，或者您可能只會看到輸入使用者名稱和密碼的提示。當您從此頁面利用使用者名稱和密碼登入時，Amazon Cognito 就是您的登入供應商。否則，您的登入供應商將以您選擇的按鈕表示。

您可以在這裡選擇供應商，或輸入使用者名稱和密碼，然後立即存取您的應用程式。如果 Amazon Cognito 是您的登入供應商，則應用程式擁有者可能還會要求多重要素驗證。

With multiple sign-in providers

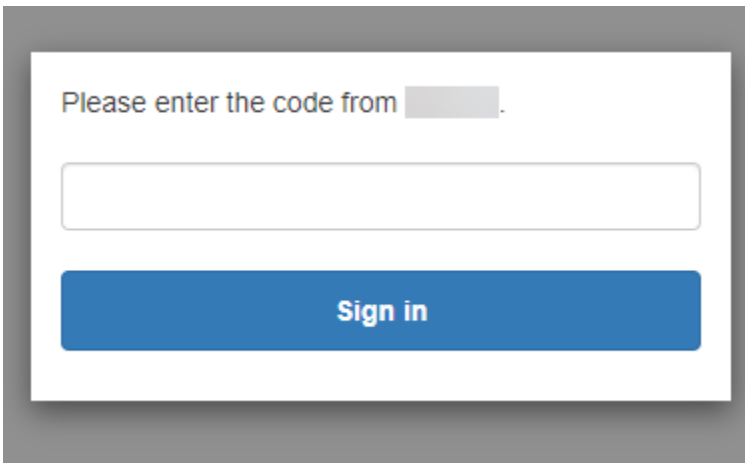


With only Amazon Cognito as a sign-in provider

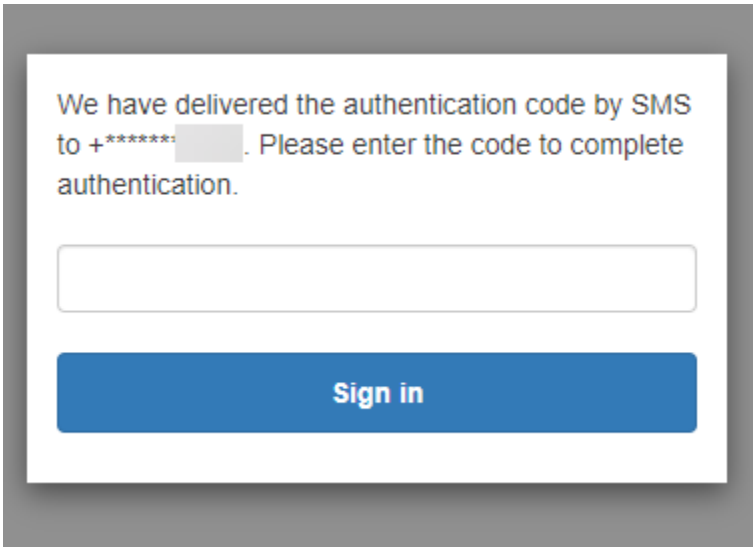


2. 在應用程式中註冊時，您可能已設定 MFA。輸入您在簡訊中所收到或顯示在驗證器應用程式中的 MFA 代碼。您必須在 3 分鐘內輸入此代碼。

With an authenticator app



With an SMS code



3. 在您登入並完成 MFA 之後，Amazon Cognito 會授與您對應用程式的存取權。

如何使用 Amazon Cognito 託管 UI 重設密碼

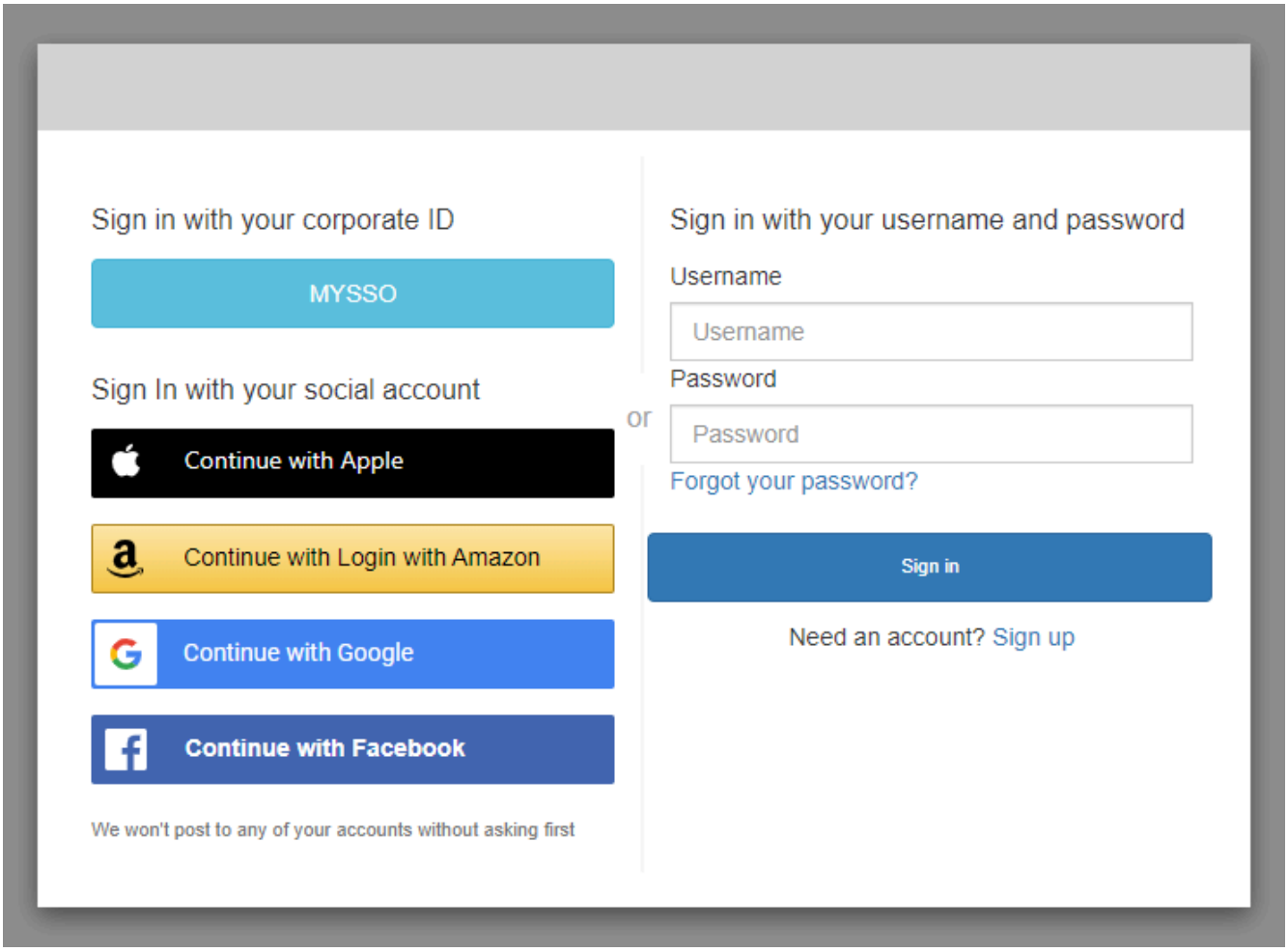
本指南說明如何在使用 Amazon Cognito 的應用程式中重設密碼。

> **Note**
>
> 當您登入使用 Amazon Cognito 託管使用者介面 (UI) 的應用程式時，除了本指南所示的基本組態之外，您可能還會看到應用程式擁有者已自訂的頁面。

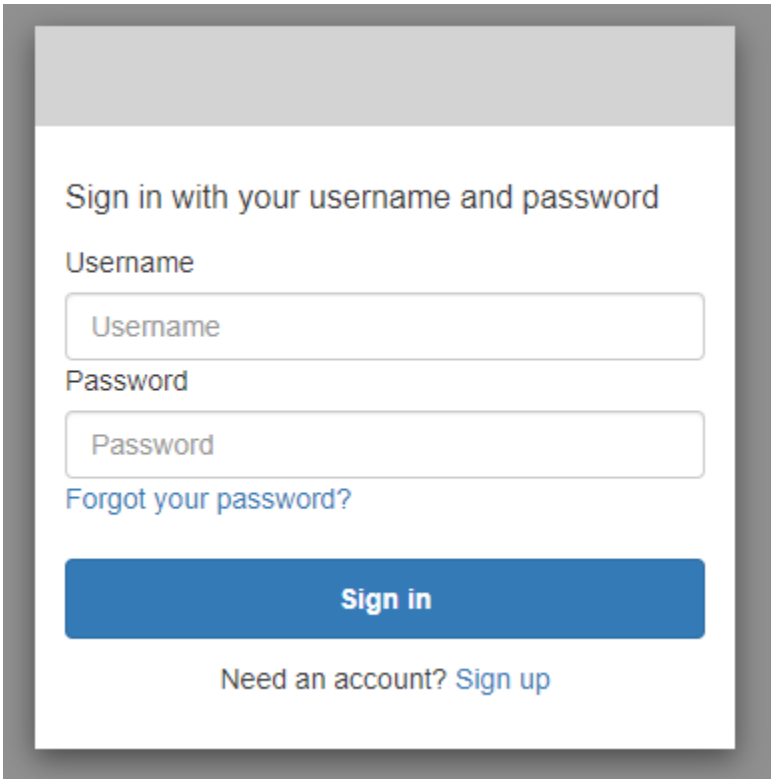
1. 視應用程式擁有者選擇的選項而定，您可能可以選擇登入時要使用的供應商，或者您可能只會看到輸入使用者名稱和密碼的提示。當您從此頁面利用使用者名稱和密碼登入時，Amazon Cognito 就是您的登入供應商。否則，您的登入供應商將以您選擇的按鈕表示。

   如果您通常從登入頁面選擇供應商，但您的密碼無法正常運作，請依照程序向供應商重設密碼。如果 Amazon Cognito 是您的登入供應商，請選擇 Forgot your password? (忘記密碼？)

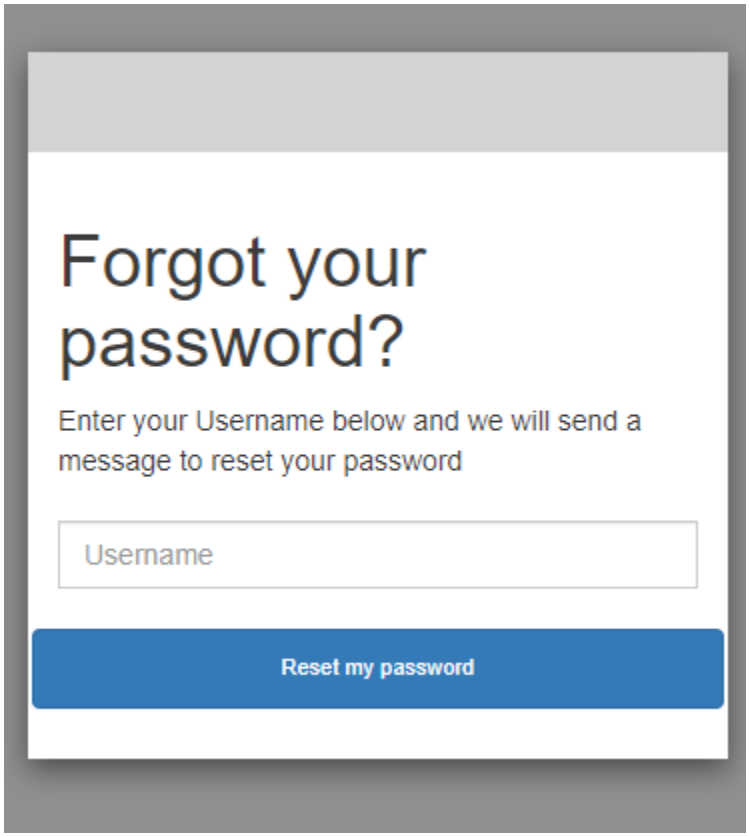
With multiple sign-in providers



With only Amazon Cognito as a sign-in provider

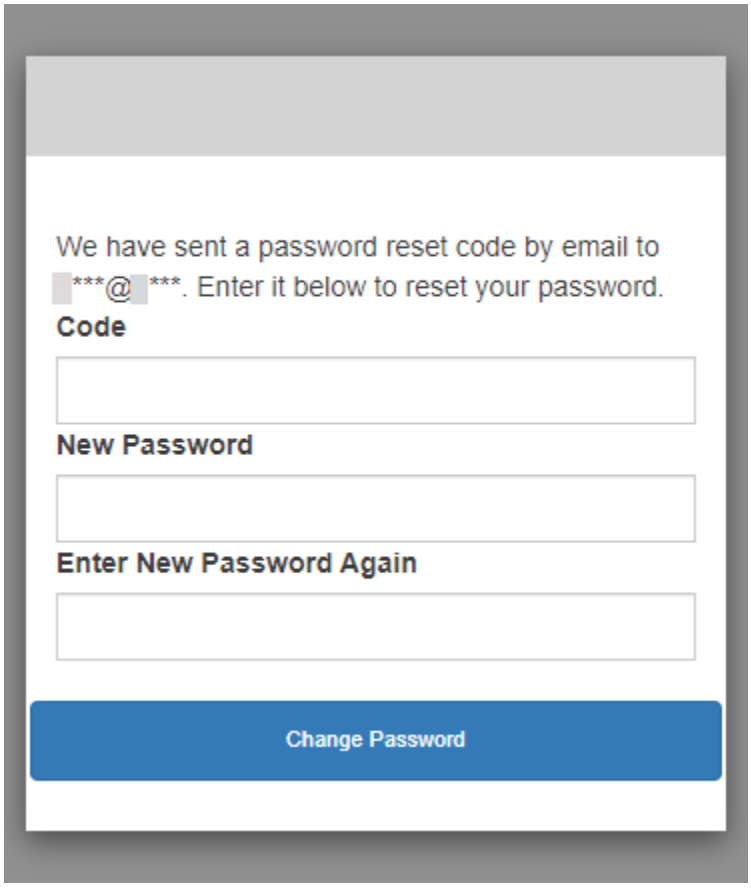


2. 在 Forgot your password? (忘記密碼？) 頁面上，Amazon Cognito 會提示您輸入用於登入的資訊。這可能是您的使用者名稱、電子郵件地址或電話號碼。



3. Amazon Cognito 會以電子郵件訊息或簡訊形式傳送驗證碼給您。

   輸入您收到的驗證碼，然後在提供的欄位中輸入兩次新密碼。您必須在 8 分鐘內輸入重設代碼。



4. 變更密碼後，請返回登入頁面並使用新密碼登入。

使用資源伺服器進行範圍、M2M 和 API 授權

在使用者集區的網域設定完成後，Amazon Cognito 會自動佈建一個 OAuth 2.0 授權伺服器和託管 Web UI，包含您的應用程式可以呈現給使用者的註冊和登入頁面。如需更多資訊，請參閱使用託管 UI 添加應用程序客戶端。您可以選擇希望授權伺服器新增到存取權杖的範圍。範圍授權對資源伺服器和使用者資料的存取權。

資源伺服器是 OAuth 2.0 API 伺服器。為了保護受存取保護的資源，它會驗證來自使用者集區的存取權杖是否包含授權其保護的 API 中請求方法和路徑範圍。它會根據權杖簽章驗證發行者、根據權杖過期驗證有效性，以及根據權杖宣告中的範圍驗證存取層級。用戶池範圍位於訪問令牌`scope`聲明中。如需 Amazon Cognito 存取權杖中宣告的詳細資訊，請參閱 使用存取權杖。

使用 Amazon Cognito，存取權杖中的範圍可以授權對外部 API 或使用者屬性的存取權。您可以向本機使用者、同盟使用者或機器身分發出存取權杖。

M achine-to-machine (M2M) 授權

Amazon Cognito 支援使用機器身分存取 API 資料的應用程式。使用者集區中的機器身分識別是在應用程式伺服器上執行並連線至遠端 API 的機密用戶端。他們的操作在沒有用戶互動的情況下進行：計劃任務，數據流或資產更新。當這些客戶端使用訪問令牌授權其請求時，它們會執行機器對機器或 M2M 授權。在 M2M 授權中，共用密碼會取代存取控制中的使用者認證。

使用 M2M 授權存取 API 的應用程式必須具有用戶端 ID 和用戶端密碼。在您的使用者集區中，您必須建置支援用戶端認證授與的應用程式用戶端。要支持客戶端憑據，您的應用程序客戶端必須具有客戶端密鑰，並且您必須具有用戶池域。在此流程中，您的機器身分會直接從權杖端點. 您可以在用戶端認證授與的存取權杖中，僅授權來自資源伺服器的自訂範圍。如需設定應用程式用戶端的詳細資訊，請參閱使用者集區應用程式用戶端。

來自客戶端憑據授予的訪問令牌是您希望允許您的機器身份從 API 請求的操作的可驗證聲明。要了解有關訪問令牌如何授權 API 請求的更多信息，請繼續閱讀。如需範例應用程式，請參閱使用 AWS CDK 的 Amazon Cognito 和 API Gateway 式機器對機器授權。

M2M 授權的計費模式與每月活躍用戶（MAU）計費方式不同。如果用戶身份驗證承擔每個活躍用戶的費用，M2M 計費反映了活動客戶端憑據應用程序客戶端和令牌請求總量。如需詳細資訊，請參閱 Amazon Cognito 定價。為了控制 M2M 授權的成本，請優化訪問令牌的持續時間和應用程序發出的令牌請求數量。有關使用 API Gateway 緩存來減少 M2M 授權中對新令牌的請求的方法，請參快取權杖閱。

如需最佳化 Amazon Cognito 操作以增加 AWS 帳單成本的相關資訊，請參閱管理成本。

關於範圍

範圍是應用程式可以請求資源的存取層級。在 Amazon Cognito 存取權杖中，範圍是由您使用使用者集區 (具有已知數位簽章的受信任存取權杖發行者) 設定的信任進行備份。使用者集區可以產生具有範圍的存取權杖，這些範圍可以證明您的客戶端可以管理自己部分或全部使用者設定檔，或從後端 API 擷取資料。Amazon Cognito 使用者集區會以使用者集區保留 API 範圍、自訂範圍，和標準範圍發出存取權杖。

使用者集區保留的 API 範圍

`aws.cognito.signin.user.admin` 範圍會授權 Amazon Cognito 使用者集區 API。它授權訪問令牌的持有者使用和 UpdateUserAttributesAPI 操作查詢和更新有關用戶池用戶的GetUser所有信息。在您使用 Amazon Cognito 使用者集區 API 驗證使用者時，這是您在存取權杖中收到的唯一範圍。這也是您讀取和寫入授權應用程式用戶端，用以讀取和寫入的使用者屬性所需的唯一範圍。您也可以在向

授權端點 提出的請求中請求此範圍。僅此範圍不足以向 UserInfo 端點 請求使用者屬性。對於為您的使用者授權使用者集區 API 和 userInfo 請求的存取權杖，您必須在 /oauth2/authorize 請求中同時請求範圍 openid 和 aws.cognito.signin.user.admin。

自訂範圍

自訂範圍授權對資源伺服器保護的外部 API 的請求。您可以使用其他類型的範圍請求自訂範圍。您可以在此頁面中尋找有關自訂範圍的詳細資訊。

標準範圍

在您使用使用者集區 OAuth 2.0 授權伺服器 (包括託管 UI) 對使用者進行身分驗證時，必須請求範圍。您可以在 Amazon Cognito 授權伺服器中驗證使用者集區本機使用者和第三方聯合身分使用者。標準 OAuth 2.0 範圍授權您的應用從使用者集區的 UserInfo 端點 中讀取使用者資訊。您可以從 userInfo 端點查詢使用者屬性的 OAuth 模型，可針對大量使用者屬性要求最佳化您的應用程式。userInfo 端點會傳回由存取權杖中的範圍決定的許可層級的屬性。您可以授權讓您的應用程式用戶端發出具有下列標準 OAuth 2.0 範圍的存取權杖。

openid

OpenID Connect (OIDC) 查詢的最小範圍。授權 ID 權杖、唯一識別符宣告 sub，以及請求其他範圍的能力。

> **Note**
>
> 當您請求 openid 範圍而沒有其他人時，您的使用者集區 ID 權杖和 userInfo 回應包含您的應用程式用戶端可以讀取的所有使用者屬性的宣告。當您請求 openid 和其他標準範圍 (例如 profile、email 和 phone) 時，ID 權杖和 userInfo 回應的內容僅限於其他範圍的限制。
> 例如，使用參數 scope=openid+email 向 授權端點 發出請求會傳回包含 sub、email 和 email_verified 的 ID 權杖。此請求的存取權杖傳回與 UserInfo 端點 相同的屬性。具有參數 scope=openid 的請求傳回返回 ID 權杖和從 userInfo 中的所有用戶端可讀屬性。

profile

授權應用程式用戶端可讀取的所有使用者屬性。

email

授權使用者屬性 email 與 email_verified。如果已明確設定值，Amazon Cognito 會傳回 email_verified。

phone

授權使用者屬性 phone_number 與 phone_number_verified。

關於資源伺服器

資源伺服器 API 可能會授與資料庫中資訊的存取權，或授與控制您的 IT 資源。Amazon Cognito 存取字符可以授權對支援 OAuth 2.0 之 API 的存取。Amazon API Gateway REST API 具有使用 Amazon Cognito 存取字符進行授權的內建支援。您的應用程式會在 API 呼叫中傳遞存取字符至資源伺服器。資源伺服器會檢查存取字符，以判斷是否應授予存取權。

Amazon Cognito 未來可能會對使用者集區存取權杖的架構進行更新。如果您的應用程式在將存取權杖傳遞給 API 之前分析過存取權杖內容，則必須設計程式碼以接受結構描述的更新。

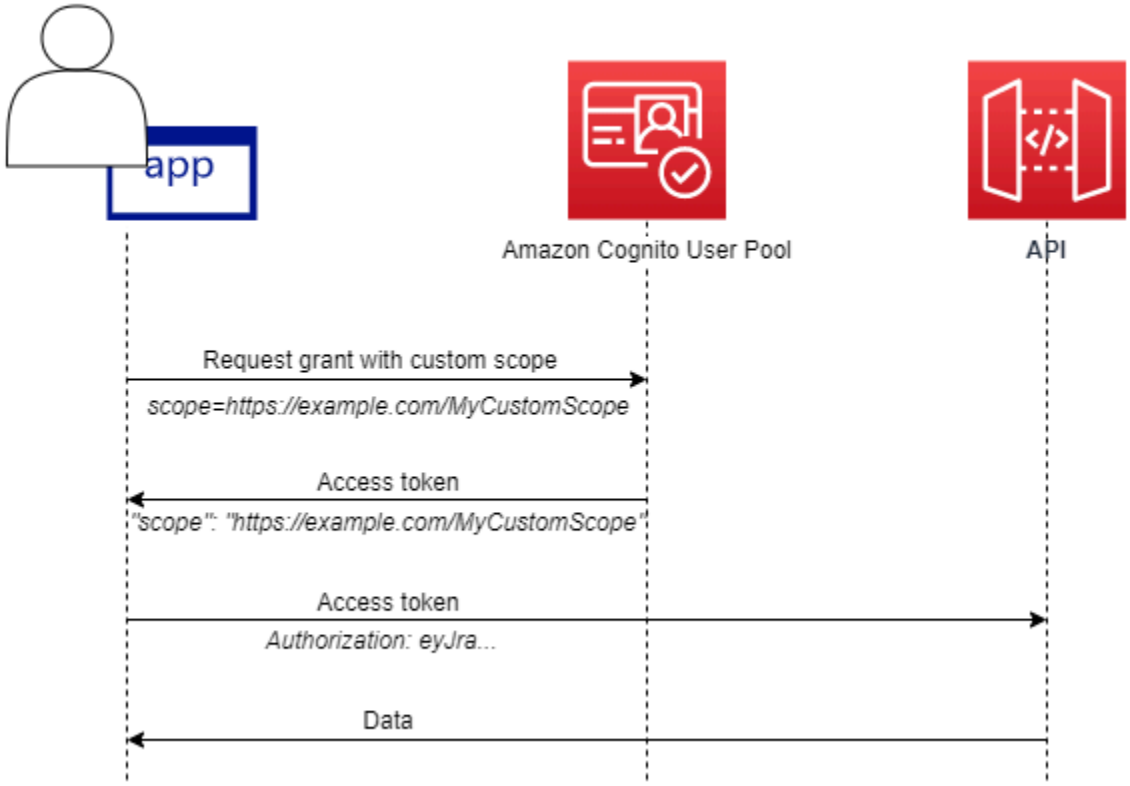
自訂範圍由您定義，並擴充使用者集區的授權功能，以納入與查詢和修改使用者及其屬性無關的用途。例如，如果您有用於照片的資源伺服器，它可能會定義兩個範圍：photos.read 用於照片的讀取存取權，photos.write 用於寫入/刪除存取權。您可以設定 API 接受存取權杖以進行授權，並在 scope 宣告中對存取權杖的 HTTP GET 請求授予 photos.read，以及對權杖的 HTTP POST 請求授予 photos.write。這些都是自訂範圍。

> **Note**
>
> 您的資源伺服器在處理字符內的任何宣告之前，必須先驗證存取字符簽章和過期日期。如需有關驗證權杖的詳細資訊，請參閱 驗證 JSON Web 權杖。如需有關驗證及使用 Amazon API Gateway 中使用者集區權杖的詳細資訊，請參閱將 Amazon Cognito 使用者集區與 API Gateway 整合部落格文章。API Gateway 是檢查存取權杖及保護資源的理想選項。如需 API Gateway Lambda 授權方的詳細資訊，請參閱使用 API Gateway Lambda 授權方。

概觀

您可以使用 Amazon Cognito 建立 OAuth2.0 資源伺服器，並將自訂範圍與這些伺服器建立關聯。存取權杖中的自訂範圍會授權您 API 中的特定動作。您可以授權讓使用者集區中的任何應用程式用戶端從任一部資源伺服器發出自訂範圍。將您的自訂範圍與應用程式用戶端建立關聯，並且在 OAuth2.0 授權碼授予、隱含授予和用戶端憑證授予中向 權杖端點 請求這些範圍。Amazon Cognito 會在存取權杖中

將自訂範圍新增至 scope 宣告中。用戶端可以對其資源伺服器使用存取字符，資源伺服器會根據存在於字符中的範圍來制定授權決策。如需存取字符範圍的詳細資訊，請參閱將字符用於使用者集區。



若要取得具有自訂範圍的存取權杖，您的應用程式必須向 權杖端點 提出請求，兌換授權碼或請求用戶端憑證授予。在託管 UI 中，您還可以透過隱含授予在存取字符中請求自訂範圍。

> **Note**
>
> 因為它們是專為使用者集區做為 IdP 的人工互動式驗證而設計，而InitiateAuth且AdminInitiateAuth要求只會在具有單一值的存取權杖中產生scope宣告。`aws.cognito.signin.user.admin`

管理資源伺服器和自訂範圍

建立資源伺服器時，您必須提供資源伺服器名稱和資源伺服器識別符。您必須為您在資源伺服器中建立的每個範圍，提供範圍名稱和描述。

- 資源伺服器名稱：資源伺服器的易記名稱，例如 `Solar system object tracker` 或 `Photo API`。

- 資源伺服器識別碼：資源伺服器的唯一識別碼。識別符 (例如 `solar-system-data`) 是您想要與 API 建立關聯的任何名稱。您可以設定較長的識別符 (例如 `https://solar-system-data-api.example.com`)，更直接地引用 API URI 路徑，但較長的權杖字串會增加存取權杖的大小。
- 範圍名稱：您想要在 `scope` 宣告中包含的值。例如 `sunproximity.read`。
- 描述：易懂的範圍描述。例如 `Check current proximity to sun`。

Amazon Cognito 可針對任何使用者在存取權杖中包含自訂範圍，無論這些使用者是您的使用者集區本機使用者還是與第三方身分提供者聯合的使用者。您可以使用包含託管 UI 的 OAuth 2.0 授權伺服器，在身分驗證流程期間為使用者的存取權杖選擇範圍。您的使用者身分驗證必須從 授權端點 開始，並以 `scope` 作為請求參數之一。以下是資源伺服器的建議格式。對於識別符，請使用 API 易記名稱。對於自訂範圍，請使用他們授權的動作。

```
resourceServerIdentifier/scopeName
```

例如，您在古柏帶 (Kuiper belt) 中發現了一顆新的小行星，並且想要透過 `solar-system-data` API 進行註冊。授權寫入操作到小行星資料庫的範圍是 `asteroids.add`。當您請求授權您註冊發現的存取權杖時，請將 `scope` HTTPS 請求參數格式化為 `scope=solar-system-data/asteroids.add`。

從資源伺服器刪除範圍並不會刪除其與所有用戶端的關聯。而是會將範圍標記為非作用中。Amazon Cognito 不會新增非作用中範圍至存取權杖，但如果您的應用程式提出請求，則會正常繼續進行。如果您稍後再次將範圍新增至資源伺服器，則 Amazon Cognito 會再次將其寫入存取權杖。如果您請求的範圍尚未與您的應用程式用戶端建立關聯，則無論您是否將該範圍從使用者集區資源伺服器中刪除，驗證都會失敗。

您可以使用 AWS Management Console、API 或 CLI 定義使用者集區的資源伺服器和範圍。

為您的使用者集區定義資源伺服器 (AWS Management Console)

您可以使用 AWS Management Console 定義使用者集區的資源伺服器。

定義資源伺服器

1. 登入 Amazon Cognito 主控台。
2. 在導覽窗格中，選擇 User Pools (使用者集區)，然後選擇您要編輯的使用者集區。
3. 選擇 App integration (應用程式整合) 索引標籤，並找到 Resource servers (資源伺服器)。
4. 選擇 Create a resource server (建立資源伺服器)。

5. 輸入 Resource server name (資源伺服器名稱)。例如 `Photo Server`。
6. 輸入 Resource server identifier (資源伺服器識別符)。例如 `com.example.photos`。
7. 輸入資源的 Custom scopes (自訂範圍)，例如 `read` 和 `write`。
8. 為每個 Scope name (範圍名稱) 輸入 Description (描述)，例如 `view your photos` 和 `update your photos`。
9. 選擇 Create (建立)。

可以在 Custom scopes (自訂範圍) 欄，Resource servers (資源伺服器) 下的 App integration (應用程式整合) 中檢閱您的自訂範圍。可以從 App clients (應用程式用戶端) 下的 App integration (應用程式整合) 索引標籤啟用自訂範圍。選取應用程式用戶端，找到 Hosted UI settings (託管 UI 設定)，然後選擇 Edit (編輯)。新增 Custom scopes (自訂範圍)，然後選擇 Save changes (儲存變更)。

為您的使用者集區 (AWS CLI 和 AWS API) 定義資源伺服器

使用下列命令為您的使用者集區指定資源伺服器設定。

建立資源伺服器

- AWS CLI: `aws cognito-idp create-resource-server`
- AWS API：CreateResourceServer

取得資源伺服器設定的相關資訊

- AWS CLI: `aws cognito-idp describe-resource-server`
- AWS API：DescribeResourceServer

為您的使用者集區列出所有資源伺服器的相關資訊

- AWS CLI: `aws cognito-idp list-resource-servers`
- AWS API：ListResourceServers

刪除資源伺服器

- AWS CLI: `aws cognito-idp delete-resource-server`
- AWS API：DeleteResourceServer

更新資源伺服器的設定

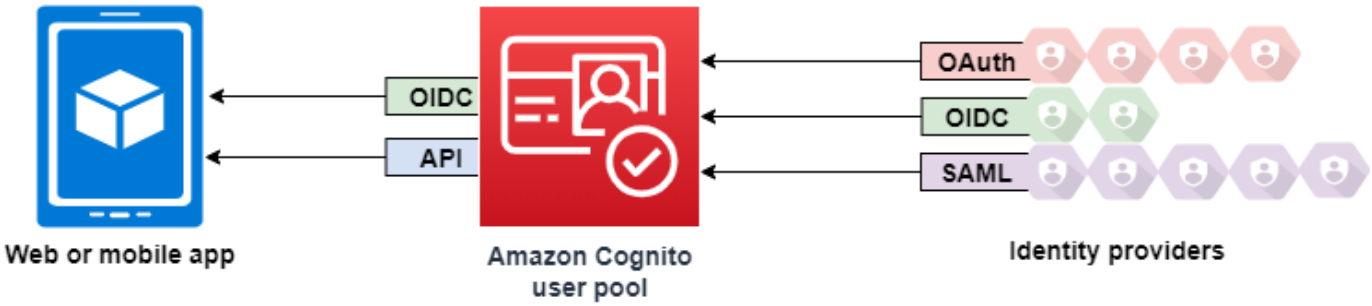
- AWS CLI: `aws cognito-idp update-resource-server`
- AWS API：UpdateResourceServer

透過第三方新增使用者集區登入

您的應用程式使用者可以直接透過使用者集區登入，也可以透過第三方身分識別提供者 (IdP) 進行聯合。用戶池管理處理通過 Facebook，谷歌，Amazon 和蘋果以及 OpenID Connect（OIDC）和 SAML 從社交登錄返回的令牌的開銷。 IdPs透過內建的託管網頁使用者介面，Amazon Cognito 可為所有 IdPs已驗證的使用者提供權杖處理和管理。這樣，您的後端系統可以對一組使用者集區字符進行標準化。

聯合登入在 Amazon Cognito 使用者集區中的運作方式

Amazon Cognito 使用者集區提供透過第三方 (聯合身分) 登入。這項功能與透過 Amazon Cognito 身分集區 (聯合身分) 登入無關。



Amazon Cognito 是使用者目錄與 OAuth 2.0 身分提供者 (IdP)。當您將本機使用者登入至 Amazon Cognito 目錄時，您的使用者集區則為您應用程式的 IdP。本機使用者僅存在於您的使用者集區目錄中，不會透過外部 IdP 進行聯合。

當您將 Amazon Cognito 連線到社交、SAML 或 OpenID Connect (OIDC) IdPs 時，您的使用者集區會充當多個服務供應商與您的應用程式之間的橋樑。對於您的 IdP 而言，Amazon Cognito 是服務供應商 (SP)。您 IdPs 將 OIDC ID 權杖或 SAML 宣告傳遞給 Amazon Cognito。Amazon Cognito 會讀取權杖或聲明中與使用者相關的宣告，並將這些宣告映射至您使用者集區目錄的新使用者描述檔。

然後，Amazon Cognito 會在自己的目錄中為您的聯合身分使用者建立一個使用者描述檔。Amazon Cognito 根據您的 IdP 的宣告 (如果是 OIDC 和社交身分提供者，則根據 IdP 操作的公用 `userinfo`

端點)，將屬性新增至您的使用者。當映射的 IdP 屬性變更時，使用者的屬性會在您的使用者集區中改變。您也可以新增更多與 IdP 屬性無關的屬性。

Amazon Cognito 為您的聯合身分使用者建立設定檔後，它會變更自己的功能，並對您的應用程式 (現在是 SP) 將自己呈現為 IdP。Amazon Cognito 是 OIDC 和 OAuth 2.0 IdP 的組合。它會產生存取權杖、ID 權杖與重新整理權杖。如需字符的詳細資訊，請參閱 將權杖用於使用者集區。

您必須設計一個與 Amazon Cognito 整合的應用程式，以對您的使用者進行身分驗證和授權，無論是聯合身分使用者或本機使用者。

作為使用 Amazon Cognito 服務提供者的應用程式責任

驗證和處理權杖中的資訊

在大多數情況下，Amazon Cognito 會將已身分驗證的使用者重新導向到附加授權碼的應用程式 URL。您的應用程式會針對存取權杖、ID 權杖及重新整理權杖交換代碼。然後，它必須檢查權杖的有效性，並根據權杖中的宣告向您的使用者提供資訊。

使用 Amazon Cognito API 請求回應驗證事件

您的應用程式必須與 Amazon Cognito 使用者集區 API 及驗證 API 端點整合。驗證 API 會您的使用者登入與登出，並管理權杖。使用者集區 API 具有各種操作可管理您的使用者集區、使用者，以及身分驗證環境的安全性。您的應用程式收到來自 Amazon Cognito 的回應時，必須知道接下來該怎麼做。

Amazon Cognito 使用者集區第三方登入相關須知

- 如果希望使用者透過聯合供應商登入，則必須選擇一個網域。這會設定 Amazon Cognito 託管的 UI 和託管的 UI 和 OIDC 端點。如需詳細資訊，請參閱 將自有網域用於託管 UI。
- 您無法使用 InitiateAuth和AdminInitiateAuth之類的 API 操作登錄聯合身份使用者。聯合身分使用者只能使用 登入端點 或 授權端點 登入。
- 授權端點 是重新導向端點。如果您在請求中提供 `idp_identifier` 或 `identity_provider` 參數，它會以無訊息方式重新導向至您的 IdP，而繞過託管 UI。否則，它將重新導向到託管 UI 登入端點。如需範例，請參閱 範例案例：在企業儀表板中將 Amazon Cognito 應用程式加入書籤。
- 當託管的 UI 將工作階段重新導向至聯合 IdP 時，Amazon Cognito 會在請求中包含 `user-agent` 標頭 `Amazon/Cognito`。

- Amazon Cognito 會從固定的識別碼與 IdP 的名稱組合衍生出聯合身分使用者描述檔的 `username` 屬性。若要產生符合您自訂需求的使用者名稱，請建立 `preferred_username` 屬性。如需詳細資訊，請參閱 對應的須知事項。

  範例：`MyIDP_bob@example.com`
- Amazon Cognito 會將有關聯合身分使用者的身分資訊記錄到某個屬性及 ID 權杖中的宣告，稱為 `identities`。此宣告包含您使用者的供應商，及來自該供應商的使用者唯一 ID。您無法直接變更使用者描述檔中的 `identities` 屬性。如需如何連結聯合身分使用者的相關詳細資訊，請參閱 將聯合身分使用者連結至現有的使用者描述檔。
- 當您在 UpdateIdentityProvider API 請求中更新 IdP，您的變更最多可能需要一分鐘的時間才會顯示在託管 UI 中。
- Amazon Cognito 在本身和您的 IdP 之間最多支援 20 個 HTTP 重新導向。
- 當您的使用者使用託管 UI 登入時，其瀏覽器會儲存一個加密的登入工作階段 Cookie，該 Cookie 會記錄他們登入的用戶端和提供者。如果他們嘗試使用相同的參數再次登入，託管 UI 會重複使用任何未過期的現有工作階段，並且使用者無需再次提供憑證即可進行驗證。如果您的使用者使用不同的 IdP 再次登入，包括切換至本機使用者集區登入或從本機使用者集區登入，則必須提供憑證並產生新的登入工作階段。

  您可以將任何使用者集區指派給任 IdPs 何應用程式用戶端，而且使用者只能使用您指派給其應用程式用戶端的 IdP 登入。

主題

為您的使用者集區配置身分提供者

在同盟身分提供者登入下的 [登入體驗] 索引標籤中，您可以將身分識別提供者 (IdPs) 新增至您的使用者集區。如需詳細資訊，請參閱 透過第三方新增使用者集區登入。

主題

- 使用社交 IdP 設定使用者登入
- 使用 OIDC IdP 設定使用者登入
- 使用 SAML IdP 設定使用者登入

使用社交 IdP 設定使用者登入

您可以使用聯合來整合 Amazon Cognito 使用者集區與社交身分提供者 (例如 Facebook、Google、Login with Amazon)。

若要新增社交身分提供者，請先以身分提供者來建立開發人員帳戶。有了開發人員帳戶之後，再向身分提供者註冊您的應用程式。身分提供者會為您的應用程式建立應用程式 ID 和密碼，您再將這些值設定在您的 Amazon Cognito 使用者集區中。

- Google Identity Platform
- Facebook for Developers
- Login with Amazon
- 使用 Apple 登入

整合使用者登入與社交 IdP

1. 登入 Amazon Cognito 主控台。若出現提示，請輸入 AWS 憑證。
2. 在導覽窗格中，選擇 User Pools (使用者集區)，然後選擇您要編輯的使用者集區。
3. 選擇 Sign-in experience (登入體驗) 索引標籤並找到 Federated sign-in (聯合登入)。
4. 選擇 Add an identity provider (新增身分提供者)，或選擇 Facebook、Google、Amazon 或者 Apple 身分提供者，請找到 Identity provider information (身分提供者資訊)，然後選擇 Edit (編輯)。如需有關新增社交身分提供者的詳細資訊，請參閱 透過使用者集區使用社交身分提供者。
5. 根據您選擇的 IdP，完成下列其中一個步驟，輸入社交身分提供者的資訊：

   Facebook、Google 和 Login with Amazon

   輸入您在建立用戶端應用程式時，所收到的應用程式 ID 和應用程式密碼。

   Sign In with Apple

   輸入您提供給 Apple 的服務 ID，以及您在建立應用程式用戶端時收到的團隊 ID、金鑰 ID 和私有金鑰。

6. 針對 Authorize scopes (授權範圍)，輸入您想要映射至使用者集區屬性的社交身分提供者範圍名稱。範圍可定義您要透過應用程式來存取的使用者屬性 (例如名稱和電子郵件)。輸入範圍時，請根據您選擇的 IdP 使用下列指導方針：

   - Facebook – 以逗號分隔範圍。例如：

     ```
     public_profile, email
     ```

   - Google、Login with Amazon 和 Sign In with Apple – 使用空格分隔範圍。例如：
     - Google：`profile email openid`
     - Login with Amazon：`profile postal_code`
     - Sign in with Apple：`name email`

   > **Note**
   >
   > 針對 Sign in with Apple (主控台)，請使用核取方塊選擇範圍。

7. 選擇儲存變更。
8. 從 App client integration (應用程式用戶端整合) 索引標籤選擇清單中的 App clients (應用程式用戶端) 之一，然後選擇 Edit hosted UI settings (編輯託管 UI 設定)。將新的社交身分提供者新增至 Identity providers (身分提供者) 下的應用程式用戶端。
9. 選擇儲存變更。

如需社交的詳細資訊 IdPs，請參閱透過使用者集區使用社交身分提供者。

使用 OIDC IdP 設定使用者登入

您可以使用 OpenID Connect (OIDC) 身分提供者 (IdP) (例如 Salesforce 或 Ping Identity) 整合使用者登入。

新增 OIDC 提供者至使用者集區

1. 前往 Amazon Cognito 主控台。若出現提示，請輸入 AWS 憑證。
2. 從導覽選單中，選擇 Users Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。
4. 選擇 Sign-in experience (登入體驗) 索引標籤。找到 Federated sign-in (聯合登入)，然後選取 Add an identity provider (新增身分提供者)。
5. 選擇 OpenID Connect 身分提供者。

6. 在 Provider name (供應商名稱) 中輸入唯一的名稱。
7. 在 Client ID (用戶端 ID) 中輸入您從供應商處收到的用戶端 ID。
8. 在 Client secret (用戶端密碼) 中輸入您從供應商處收到的用戶端密碼。
9. 輸入此供應商的 Authorized scopes (授權範圍)。範圍會定義您的應用程式從您的供應商請求的使用者屬性群組 (例如 `name` 和 `email`)。範圍必須以空格隔開，遵循 OAuth 2.0 規格。

   您的使用者必須同意提供這些屬性給應用程式。
10. 選擇 Attribute request method (屬性請求方法) 以提供 HTTP 方法 (GET 或 POST) 給 Amazon Cognito，Amazon Cognito 必須使用此方法從您的供應商操作的 userInfo 端點擷取使用者詳細資訊。
11. 透過 Auto fill through issuer URL (透過發行者 URL 自動填入) 或 Manual input (手動輸入)，選擇 Setup method (設定方法) 來擷取 OpenID Connect 端點。您的供應商擁有公有 `.well-known/openid-configuration` 端點 (Amazon Cognito 可在此處擷取 `authorization`、`token`、`userInfo` 和 `jwks_uri` 端點的 URL) 時，請使用 Auto fill through issuer URL (透過發行者 URL 自動填入)。
12. 輸入發行者 URL 或您 IdP 的 `authorization`、`token`、`userInfo` 和 `jwks_uri` 端點 URL。

    > **Note**
    >
    > 您只能將連接埠號 443 和 80 用於探索、自動填入和手動輸入的 URL。如果您的 OIDC 供應商使用任何非標準 TCP 連接埠，使用者登入將會失敗。
    > 發行者 URL 開頭必須為 `https://`，而且結尾不得是 `/` 字元. 例如，Salesforce 使用此 URL：
    > `https://login.salesforce.com`
    > 與您的發行者 URL 關聯的 `openid-configuration` 文件必須為以下值提供 HTTPS URL：`authorization_endpoint`、`token_endpoint`、`userinfo_endpoint`，以及`jwks_uri`。同樣的，當您選擇 Manual input (手動輸入) 時，只能輸入 HTTPS URL。
13. OIDC 宣告 sub 預設會映射至使用者集區屬性 Username (使用者名稱)。您可以將其他 OIDC 宣告對應至使用者集區屬性。輸入 OIDC 宣告，並從下拉式清單中選取對應的使用者集區屬性。例如，宣告 email 通常對應至使用者集區屬性 Email。
14. 將身分提供者的其他屬性映射至您的使用者集區。如需詳細資訊，請參閱為您的使用者集區指定身分提供者屬性映射。
15. 選擇 Create (建立)。

16. 從 App client integration (應用程式用戶端整合) 索引標籤選取清單中的 App clients (應用程式用戶端) 之一，和 Edit hosted UI settings (編輯託管 UI 設定)。將新的 OIDC 身分提供者新增至 Identity providers (身分提供者) 下的應用程式用戶端。
17. 選擇儲存變更。

如需 OIDC 的詳細資訊 IdPs，請參閱。搭配使用者集區使用 OIDC 身分識別提供者

使用 SAML IdP 設定使用者登入

您可以使用 Amazon Cognito 使用者集區的聯合來與 SAML 身分提供者 (IdP) 整合。提供中繼資料文件時，您可以上傳檔案，或是輸入中繼資料文件端點 URL。如需取得協力廠商 SAML 之中繼資料文件的相關資訊 IdPs，請參閱設定您的第三方 SAML 身分提供者。

在您的使用者集區中配置 SAML 2.0 身分提供者

1. 前往 Amazon Cognito 主控台。若出現提示，請輸入 AWS 憑證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。
4. 選擇 Sign-in experience (登入體驗) 索引標籤。找到 Federated sign-in (聯合登入)，然後選取 Add an identity provider (新增身分提供者)。
5. 選擇 SAML 身分提供者。
6. 輸入以逗號分隔的 Identifiers (識別符)。識別符指示 Amazon Cognito 檢查使用者登入電子郵件地址，然後將使用者導向至與其網域對應的供應商。
7. 如果您希望 Amazon Cognito 在使用者登出時傳送已簽署的登出請求給您的供應商，則請選擇 Add sign-out flow (新增登出流程)。設定您的 SAML 2.0 身分提供者，以便將登出回應傳送至您設定託管 UI 時 Amazon Cognito 建立的 `https://`*`mydomain.us-east-1.amazoncognito.com`*`/saml2/logout` 端點。`saml2/logout` 端點會使用 POST 繫結。

   > **Note**
   >
   > 如果選取此選項，您的 SAML 身分提供者預期簽署的登出要求，則您還必須設定由 Amazon Cognito 使用您的 SAML IdP 提供的簽署憑證。
   > SAML IdP 將處理簽署登出要求，以及將您的使用者從 Amazon Cognito 工作階段登出。

8. 選擇 Metadata document source (中繼資料文件來源)。如果您的身分提供者以公有 URL 提供 SAML 中繼資料，則可以選擇 Metadata document URL (中繼資料文件 URL)，然後輸入該公有

URL。否則，請選擇 Upload metadata document (上傳中繼資料文件)，然後選取您先前從供應商處下載的中繼資料檔案。

> **Note**
>
> 如果供應商擁有公有端點，建議您輸入中繼資料文件 URL，而不是上傳檔案。如果您使用 URL，Amazon Cognito 會自動重新整理中繼資料。通常每 6 小時或是在中繼資料過期之前 (取較早的時間) 重新整理中繼資料。

9. Map attributes between your SAML provider and your app (在 SAML 供應商與應用程式之間映射屬性)，將 SAML 供應商屬性映射至使用者集區中的使用者描述檔。在屬性映射中包含您的使用者集區必要屬性。

   例如，當您選擇 User pool attribute (使用者集區屬性) `email` 時，隨著身分提供者 SAML 聲明中顯示的 SAML 屬性名稱輸入 SAML 屬性名稱。您的身分提供者可能會提供範例 SAML 聲明以供參考。有些身分供應商會使用簡單的名稱，例如 `email`，有些則會使用類似以下的 URL 格式屬性名稱：

   ```
   http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
   ```

10. 選擇建立。

> **Note**
>
> 如果您以 HTTPS 中繼資料端點 URL 來建立 SAML IdP 時，看到 `InvalidParameterException`，請確定中繼資料端點已正確設定 SSL，並且有與其相關聯的有效 SSL 憑證。這種例外狀況的例子之一是「Error retrieving metadata from *<metadata endpoint>*」(從 <metadata endpoint> 錯誤擷取中繼資料)。

若要設定 SAML IdP 以新增簽署的憑證

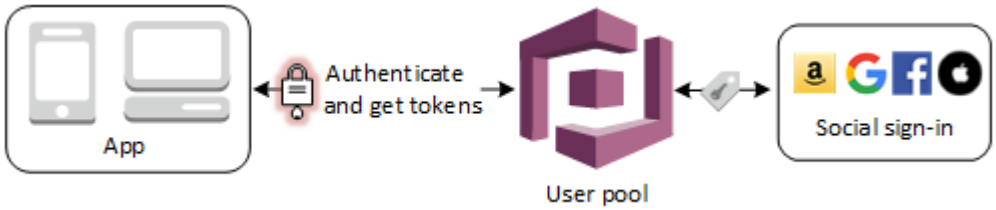
- 若要取得包含 IdP 來驗證簽署登出要求的公有金鑰的憑證， 請選擇 聯合 主控台頁面上 身分提供者 下 SAML 對話方塊上 使用中的 SAML 供應商 下的 顯示簽署憑證。

如需 SAML 的詳細資訊， IdPs 請參閱搭配使用者集區使用 SAML 身分識別提供者。

透過使用者集區使用社交身分提供者

您的 Web 和行動應用程式使用者可透過社交身分提供者 (IdP，如 Facebook、Google、Amazon 或 Apple) 進行登入。Amazon Cognito 使用內建的託管 Web UI，可以為所有已進行身分驗證的使用者提供字符處理和管理。這樣，您的後端系統可以對一組使用者集區字符進行標準化。您必須啟用託管 UI，才能與支援的社交身分提供者整合。當 Amazon Cognito 建立您的託管使用者介面時，它會建立 OAuth 2.0 端點，供 Amazon Cognito 和您的 OIDC 和社群 IdPs 用來交換資訊。如需詳細資訊，請參閱 Amazon Cognito 使用者集區 Auth API 參考。

您可以在中新增社交 IdP AWS Management Console，也可以使用 AWS CLI 或 Amazon Cognito API。



> **Note**
>
> Amazon Cognito 使用者集區提供透過第三方 (聯合身分) 登入。這項功能與透過 Amazon Cognito 身分集區 (聯合身分) 登入無關。

主題

必要條件

開始之前，您必須準備好以下事項：

- 具有應用程式用戶端和使用者集區網域的使用者集區。如需詳細資訊，請參閱建立使用者集區。
- 社交 IdP。

步驟 1：使用社交 IdP 註冊

與 Amazon Cognito 建立社交 IdP 之前，您必須將您的應用程式註冊至社交 IdP 以接收用戶端 ID 和用戶端密碼。

向 Facebook 註冊應用程式

1. 向 Facebook 建立開發人員帳戶。
2. 使用您的 Facebook 憑證登入。
3. 從 My Apps (我的應用程式) 選單中，選擇 Create New App (建立新的應用程式)。
4. 輸入您 Facebook 應用程式的名稱，然後選擇 Create App ID (建立應用程式 ID)。
5. 在左側導覽列中，選擇 Settings (設定) 然後 Basic (基本)。
6. 記下 App ID (應用程式 ID) 和 App Secret (應用程式秘密)。您會在下一節中用到它們。
7. 從頁面底部選擇 + Add Platform (+ 新增平台)。
8. 選擇 Website (網站)。
9. 在 Website (網站) 下，請將您應用程式登入頁面的路徑輸入 Site URL (網站 URL)。

   ```
   https://mydomain.us-east-1.amazoncognito.com/login?
   response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
   ```

10. 選擇儲存變更。
11. 將您的使用者集區網域的根路徑輸入 App Domains (應用程式網域)。

    ```
    https://mydomain.us-east-1.amazoncognito.com
    ```

12. 選擇儲存變更。
13. 從導覽列選擇 Add Product (新增產品)，並為 Facebook Login (Facebook 登入) 產品選擇 Set up (設定)。
14. 從導覽列選擇 Facebook Login (Facebook 登入)，然後選擇 Settings (設定)。

    將您使用者集區網域的 `/oauth2/idpresponse` 端點路徑輸入至 Valid OAuth Redirect URIs (有效的 OAuth 重新導向 URI)。

    ```
    https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
    ```

15. 選擇 Save changes (儲存變更)。

向 Amazon 註冊應用程式

1. 向 Amazon 建立開發人員帳戶。
2. 使用您的 Amazon 憑證登入。
3. 您需要建立 Amazon 安全性設定檔以便接收 Amazon 用戶端 ID 和用戶端秘密。

   從頁面頂端的導覽列中選擇 Apps and Services (應用程式和服務)，然後選擇 Login with Amazon (登入 Amazon)。
4. 選擇 Create a Security Profile (建立安全性設定檔)。
5. 輸入 Security Profile Name (安全設定檔名稱)、Security Profile Description (安全設定檔描述) 和 Consent Privacy Notice URL (同意隱私權聲明 URL)。
6. 選擇 Save (儲存)。
7. 選擇 Client ID (用戶端 ID) 和 Client Secret (用戶端秘密)，以顯示用戶端 ID 和秘密。您會在下一節中用到它們。
8. 將滑鼠移到齒輪圖示上方並選擇 Web Settings (Web 設定)，然後選擇 Edit (編輯)。
9. 在 Allowed Origins (允許的來源) 中輸入您的使用者集區網域。

   ```
   https://mydomain.us-east-1.amazoncognito.com
   ```
10. 在 Allowed Return URLs (允許的傳回 URL) 中輸入具有 `/oauth2/idpresponse` 端點的使用者集區網域。

    ```
    https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
    ```
11. 選擇 Save (儲存)。

向 Google 註冊應用程式

如需 Google Cloud Platform 中 OAuth 2.0 的詳細資訊，請參閱 Google Workspace for Developers 文件中的瞭解身分驗證和授權。

1. 向 Google 建立開發人員帳戶。
2. 登入 Google Cloud Platform 主控台。
3. 在頂端導覽列中，選擇 Select a project (選取專案)。如果您在 Google 平台中已經有專案，則此功能表將改為顯示您的預設專案。
4. 選擇 NEW PROJECT (新專案)。

5. 輸入專案的名稱，然後選擇 CREATE (建立)。
6. 在左側導覽列中，選擇 APIs and Services (API 和服務)，然後選擇 OAuth consent screen (OAuth 同意畫面)。
7. 輸入應用程式資訊，App domain (應用程式網域)、Authorized domains (授權網域)，以及 Developer contact information (開發人員聯絡資訊)。您的 Authorized domains (授權網域) 必須包含您自訂網域的 `amazoncognito.com` 與根目錄，例如 `example.com`。選擇 SAVE AND CONTINUE (儲存並繼續)。
8. 1. 在 Scopes (範圍) 下方，選擇 Add or remove scopes (新增或移除範圍)，然後至少選擇下列 OAuth 範圍。

   1. `.../auth/userinfo.email`
   2. `.../auth/userinfo.profile`
   3. openid
9. 在 Test users (測試使用者) 下方，選擇 Add users (新增使用者)。輸入您的電子郵件地址和任何其他授權的測試使用者，然後選擇 SAVE AND CONTINUE (儲存並繼續)。
10. 再次展開左側導覽列，然後依序選擇 APIs and Services (API 和服務)、Credentials (憑證)。
11. 依序選擇 CREATE CREDENTIALS (建立憑證)、OAuth client ID (OAuth 用戶端 ID)。
12. 選擇 Application type (應用程式類型)，並為您的用戶端取一個 Name (名稱)。
13. 在 [授權 JavaScript 來源] 下，選擇 [新增 URI]。輸入您的使用者集區網域。

    ```
    https://mydomain.us-east-1.amazoncognito.com
    ```
14. 在 Authorized redirect URIs (授權的重新導向 URI) 下方，選擇 ADD URI (新增 URI)。輸入您使用者集區網域的 `/oauth2/idpresponse` 端點路徑。

    ```
    https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
    ```
15. 選擇 CREATE (建立)。
16. 安全地儲存 Google 在 Your client ID (您的用戶端 ID) 和 Your client secret (您的用戶端密碼) 下顯示的值。當您新增 Google IdP 時，請將這些值提供給 Amazon Cognito。

向 Apple 註冊應用程式

如需設定「使用 Apple 登入」的最多 up-to-date 資訊，請參閱 Apple 開發人員說明文件中的「設定使用 Apple 登入的環境」。

1. 向 Apple 建立開發人員帳戶。
2. 使用您的 Apple 憑證登入。
3. 在左側導覽列上，選擇 Certificates, Identifiers & Profiles (憑證、識別碼與設定檔)。
4. 在左側導覽列上，選擇 Identifiers (識別碼)。
5. 在 Identifiers (識別碼) 頁面上，選擇 + 圖示。
6. 在 Register a New Identifier (註冊新的識別碼) 頁面上，選擇 App IDs (應用程式 ID)，然後選擇 Continue (繼續)。
7. 在 Select a type (選取類型) 頁面上，選擇 App (應用程式)，然後選擇 Continue (繼續)。
8. 在 Register an App ID (註冊應用程式 ID) 頁面上，執行下列操作：
    1. 在 Description (描述) 下方輸入描述。
    2. 在 App ID Prefix (應用程式 ID 字首) 下方，輸入 Bundle ID (套件 ID)。記下 App ID Prefix (應用程式 ID 字首) 下的值。在 步驟 2：新增社交 IdP 到您的使用者集區 中選擇 Apple 作為您的身分提供者之後，您將會使用此值。
    3. 在 Capabilities (功能) 下方，選擇 Sign In with Apple，然後選擇 Edit (編輯)。
    4. 在 Sign in with Apple: App ID Configuration (Sign in with Apple：應用程式 ID 組態) 頁面上，選擇將應用程式設定為主要應用程式或與其他應用程式 ID 群組，然後選擇 Save (儲存)。
    5. 選擇繼續。
9. 在 Confirm your App ID (確認您的應用程式 ID) 頁面上，選擇 Register (註冊)。
10. 在 Identifiers (識別碼) 頁面上，選擇 + 圖示。
11. 在 Register a New Identifier (註冊新的識別碼) 頁面上，選擇 Services IDs (服務 ID)，然後選擇 Continue (繼續)。
12. 在 Register a Services ID (註冊服務 ID) 頁面上，執行下列操作：
    1. 在 Description (說明) 下方，輸入說明內容。
    2. 在 Identifier (識別碼) 下方，輸入識別碼。記下服務 ID，因為在 步驟 2：新增社交 IdP 到您的使用者集區 中選擇 Apple 當作您的身分提供者後，您會需要此值。
    3. 選擇 Continue (繼續)，然後選擇 Register (註冊)。
13. 從「Identifiers (識別碼)」頁面中選擇您剛剛建立的服務 ID。
    1. 選取 Sign In with Apple，然後選擇 Configure (設定)。
    2. 在 Web Authentication Configuration (Web 身分驗證組態) 頁面上，選取您稍早建立的應用程式 ID 作為 Primary App ID (主要應用程式 ID)。

3. 在 Website URLs (網站 URL) 旁邊選擇 + 圖示。
4. 在 Domains and subdomains (網域與子網域) 下方，輸入不含 `https://` 字首的使用者集區網域。

```
mydomain.us-east-1.amazoncognito.com
```

5. 在 Return URLs (傳回 URL) 下，輸入您使用者集區網域的 `/oauth2/idpresponse` 端點路徑。

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
```

6. 選擇 Next (下一步)，然後選擇 Done (完成)。您不須驗證網域。
7. 選擇 Continue (繼續)，然後選擇 Save (儲存)。

14. 在左側導覽列上，選擇 Keys (金鑰)。
15. 在 Keys (金鑰) 頁面上，選擇 + 圖示。
16. 在 Register a New Key (註冊新的金鑰) 頁面上，執行下列操作：
    1. 在 Key Name (金鑰名稱) 下，輸入金鑰名稱。
    2. 選擇 Sign In with Apple，然後選擇 Configure (設定)。
    3. 在 Configure Key (設定金鑰) 頁面上，然後選取您稍早建立的應用程式 ID 作為 Primary App ID (主要應用程式 ID)。選擇儲存。
    4. 選擇 Continue (繼續)，然後選擇 Register (註冊)。
17. 在 Download Your Key (下載您的金鑰) 頁面，選擇 Download (下載) 以下載私有金鑰，記下顯示的 Key ID (金鑰 ID)，接著選擇 Done (完成)。您在 步驟 2：新增社交 IdP 到您的使用者集區 中選擇 Apple 當作身分提供者後，需有此頁面上顯示的私有金鑰和 Key ID (金鑰 ID) 值。

步驟 2：新增社交 IdP 到您的使用者集區

設定使用者集區社交 IdP AWS Management Console

1. 前往 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。
4. 選擇 Sign-in experience (登入體驗) 索引標籤。找到 Federated sign-in (聯合登入)，然後選取 Add an identity provider (新增身分提供者)。

5. 選擇社交 IdP：Facebook、Google、Login with Amazon (登入 Amazon) 或 Sign in with Apple (使用 Apple 登入)。
6. 根據您選擇的社交 IdP，從以下步驟中選擇：

   - Google 和 Login with Amazon (登入 Amazon)– 輸入上一節產生的 app client ID (應用程式用戶端 ID) 和 app client secret (應用程式用戶端密碼)。
   - Facebook – 輸入上一節產生的 app client ID (應用程式用戶端 ID) 和 app client secret (應用程式用戶端秘密)，然後選擇 API 版本 (例如版本 2.12)。建議您選擇最新可用的版本，因為每個 Facebook API 都有生命週期和停止支援日期。Facebook 範圍和屬性可能隨 API 版本而有所不同。建議您使用 Facebook 測試您的社交身分登入，以確保聯合使用正常運作。
   - Sign In with Apple – 輸入上一節產生的 Services ID (服務 ID)、Team ID (團隊 ID)、Key ID (金鑰 ID) 和 private key (私有金鑰)。
7. 輸入您要使用的授權範圍名稱。範圍可定義您要透過應用程式來存取的使用者屬性 (例如 `name` 和 `email`)。若為 Facebook，這些屬性應以逗號分隔。若為 Google 和登入 Amazon，則應以空格分隔。若為 Sign in with Apple，請針對您想要存取的範圍勾選核取方塊。

社交身分提供者	範例範圍
Facebook	`public_profile, email`
Google	`profile email openid`
登入 Amazon	`profile postal_code`
使用 Apple 登入	`email name`

會向您的應用程式使用者提示同意提供這些屬性給應用程式。如需社交供應商範圍的詳細資訊，請參閱 Google、Facebook、Login with Amazon 或 Sign in with Apple 提供的說明文件。

透過 Sign in with Apple，以下是可能未傳回範圍的使用者案例：

- 最終使用者在離開 Apple 的登入頁面之後遇到錯誤 (可能是 Amazon Cognito 的內部錯誤或開發人員撰寫的任何項目)
- 服務 ID 識別碼在使用者集區和/或其他身分驗證服務之間使用
- 開發人員在最終使用者先前登入之後新增其他範圍 (未擷取新的資訊)
- 開發人員刪除使用者，然後使用者再次登入而未從他們的 Apple ID 描述檔移除應用程式

8. 將 IdP 的屬性對應至您的使用者集區。如需詳細資訊，請參閱為您的使用者集區指定身分提供者屬性對應。
9. 選擇 Create (建立)。
10. 從 App client integration (應用程式用戶端整合) 標籤，在清單中選擇一個 App clients (應用程式用戶端)，以及 Edit hosted UI settings (編輯託管 UI 設定)。將新的社交 IdP 新增至 Identity providers (身分提供者) 下的應用程式用戶端。
11. 選擇儲存變更。

步驟 3：測試社交 IdP 組態

您可以使用前兩節的元素建立登入 URL。用來測試社交 IdP 組態。

```
https://mydomain.us-east-1.amazoncognito.com/login?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

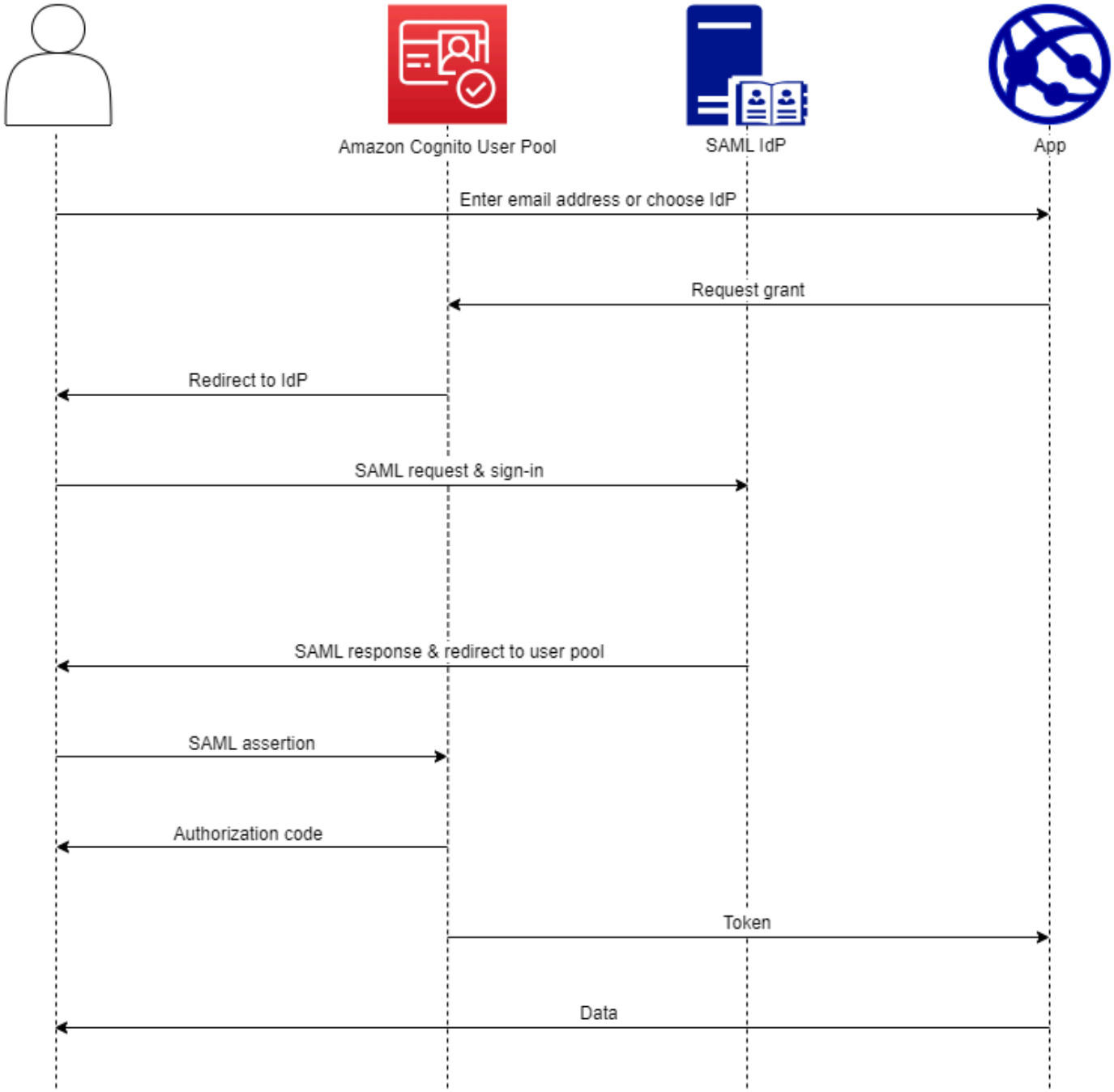
您可以在使用者集區 Domain name (網域名稱) 主控台頁面上尋找您的網域。client_id 位於 App client settings (應用程式用戶端設定) 頁面上。將您的回呼 URL 使用於 redirect_uri 參數。這是您的使用者成功身分驗證後會被重新導向的頁面 URL。

> **Note**
>
> 身分驗證請求若未在 5 分鐘內完成，Amazon Cognito 會將該請求取消，並將使用者重新導向至託管的 UI。此頁面人會顯示「Something went wrong」錯誤訊息。

搭配使用者集區使用 SAML 身分識別提供者

您可以選擇讓您的網路和行動應用程式使用者透過 SAML 身分識別提供者 (IdP) (例如 Microsoft 作用中目錄聯合服務 (ADFS) 或 Shibboleth 等登入。您必須選擇支援 SAML 2.0 標準的 SAML IdP。

透過託管的使用者介面和聯合端點，Amazon Cognito 可驗證本機和第三方 IdP 使用者，並發出 JSON 網路權杖 (JWT)。使用 Amazon Cognito 發行的權杖，您可以將多個身分識別來源合併為所有應用程式的通用 OpenID Connect (OIDC) 標準。Amazon Cognito 可以將來自第三方供應商的 SAML 宣告處理為該 SSO 標準。您可以在 AWS Management Console、透過或使用 Amazon Cognito 使用者集區 API 建立和管理 SAML IdP。 AWS CLI若要在中建立您的第一個 SAML IdP AWS Management Console，請參閱。在使用者集區中新增和管理 SAML 身分識別提供者



Note

透過第三方 IdP 登入的聯合是 Amazon Cognito 使用者集區的一項功能。Amazon Cognito 身分集區 (有時稱為 Amazon Cognito 聯合身分) 是聯合身分的實作，您必須在每個身分集區中個

別設定。使用者集區可以是身分集區的第三方 IdP。如需詳細資訊，請參閱 Amazon Cognito 身分集區。

IdP 組態的快速參考

您必須將 SAML IdP 設定為接受要求並將回應傳送至您的使用者集區。SAML IdP 的說明文件將包含有關如何將使用者集區新增為 SAML 2.0 IdP 的信賴憑證者或應用程式的資訊。下列文件提供您必須為 SP 實體識別碼和宣告取用者服務 (ACS) URL 提供的值。

使用者集區 SAML 值快速參考

SP 實體識別碼

```
urn:amazon:cognito:sp:us-east-1_EXAMPLE
```

ACS 網址

```
https://Your user pool domain/saml2/idpresponse
```

您必須設定使用者集區以支援您的身分識別提供者。新增外部 SAML IdP 的高階步驟如下。

1. 從 IdP 下載 SAML 中繼資料，或擷取中繼資料端點的 URL。請參閱設定您的第三方 SAML 身分提供者。
2. 將新的 IdP 新增至您的使用者集區。上傳 SAML 中繼資料或提供中繼資料 URL。請參閱在使用者集區中新增和管理 SAML 身分識別提供者。
3. 將 IdP 指派給您的應用程式用戶端。請參閱使用者集區應用程式用戶端

主題

- 有關 Amazon Cognito 用戶池 IdPs 中 SAML 的注意事項
- SAML 使用者名稱區分大小寫
- 在使用者集區中新增和管理 SAML 身分識別提供者
- Amazon Cognito 使用者集區中的 SAML 工作階段啟動
- 使用 SP 啟動式 SAML 登入
- 使用 IDP 起始的 SAML 登入
- SAML 登出流程

- SAML 簽章與加密
- SAML 身分識別提供者名稱和識別碼
- 設定您的第三方 SAML 身分提供者

有關 Amazon Cognito 用戶池 IdPs 中 SAML 的注意事項

Amazon Cognito 您處理 SAML 聲明

Amazon Cognito 使用者集區支援與 POST 繫結端點進行 SAML 2.0 聯合。這樣您的應用程式就不需要擷取或剖析 SAML 聲明回應，因為使用者集區會透過使用者代理程式，直接從您的 IdP 收到 SAML 回應。使用者集區會代表您的應用程式做為服務供應商 (SP)。Amazon Cognito 支援 SAML 第 2.0 版技術概觀第 5.1.2 和 5.1.4 節中所述，以 SP 啟動和 IDP 啟動的單一登入 (SSO)。

提供有效的 IdP 簽署憑證

當您在使用者集區中設定 SAML IdP 時，SAML 提供者中繼資料中的簽署憑證不得過期。

使用者集區支援多個簽署憑證

當您的 SAML IdP 在 SAML 中繼資料中包含多個簽署憑證時，登入時，如果 SAML 宣告與 SAML 中繼資料中的任何憑證相符，則您的使用者集區會判斷 SAML 聲明有效。每個簽署憑證的長度不得超過 4,096 個字元。

維護繼電器狀態參數

Amazon Cognito 和您的 SAML IdP 會使用 `relayState` 參數維護工作階段資訊。

1. Amazon Cognito 支援大於 80 個位元組的 `relayState` 值。雖然 SAML 規格指出 `relayState` 值「長度不得超過 80 個位元組」，但現今的業界經常不遵從此規範。而拒絕超過 80 個位元組的 `relayState` 值，將會破壞許多標準 SAML 供應商整合。
2. `relayState`權杖是 Amazon Cognito 維護的狀態資訊的不透明參考。Amazon Cognito 不會保證 `relayState` 參數的內容。請勿解析其內容，如此你的應用程式便會取決於結果。如需詳細資訊，請參閱 SAML 2.0 規格。

點選 ACS 端點

您的 SAML 身分提供者要求您設定宣告消費者端點。您的 IdP 會使用其 SAML 宣告將您的使用者重新導向至此端點。在 SAML 身分提供者中為 SAML 2.0 POST 繫結設定使用者集區網域中的下列端點。

```
https://Your user pool domain/saml2/idpresponse
With an Amazon Cognito domain:
```

```
https://mydomain.us-east-1.amazoncognito.com/saml2/idpresponse
With a custom domain:
https://auth.example.com/saml2/idpresponse
```

如需使用者集區網域的詳細資訊，請參閱 設定使用者集區網域。

沒有重播斷言

您無法對 Amazon Cognito `saml2/idpresponse` 端點重複或重播 SAML 宣告。重播的 SAML 宣告具有重複先前 IdP 回應 ID 的宣告 ID。

使用者集區識別碼為 SP 實體識別碼

您必須在服務提供者 (SP) 中向您的 IdP 提供您的使用者集區 IDurn，也稱為對象 URI 或 SP 實體 ID。您使用者集區的對象 URI 具有以下格式。

```
urn:amazon:cognito:sp:us-east-1_EXAMPLE
```

您可以在 Amazon Cognito 主控台的使用者集區概觀下找到您的使用者集區 ID。

對映所有必要屬性

設定您的 SAML IdP，以為您在使用者集區中視需要設定的任何屬性提供值。例如，`email` 是使用者集區的常用必要屬性。在您的使用者可以登入之前，您的 SAML IdP 宣告必須包含您對應至 使用者集區屬性 `email` 的宣告。如需屬性對應的詳細資訊，請參閱 為您的使用者集區指定身分提供者屬性映射。

斷言格式有特定的要求

您的 SAML IdP 必須在 SAML 宣告中包含下列宣告。

1. 索`NameID`賠。Amazon Cognito 會將 SAML 宣告與目的地使用者建立關聯。`NameID`如果有`NameID`變更，Amazon Cognito 會將此宣告視為適用於新使用者的宣告。您在 IdP 組態`NameID`中設定的屬性必須具有永久值。

   ```
   <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:persistent">
     carlos
   </saml2:NameID>
   ```

2. 具有 `Audience` 值的 `AudienceRestriction` 宣告，會將您的使用者集區 SP 實體 ID 設定為回應的目標。

   ```
   <saml:AudienceRestriction>
     <saml:Audience> urn:amazon:cognito:sp:us-east-1_EXAMPLE
   ```

```
</saml:AudienceRestriction>
```

3. 對於 SP 起始的單一登入，這是一個InResponseTo值為原始 SAML 要求識別碼的Response元素。

```
<saml2p:Response Destination="https://mydomain.us-east-1.amazoncognito.com/
saml2/idpresponse" ID="id123" InResponseTo="_dd0a3436-bc64-4679-
a0c2-cb4454f04184" IssueInstant="Date-time stamp" Version="2.0"
 xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" xmlns:xs="http://
www.w3.org/2001/XMLSchema">
```

> **Note**
>
> IdP 起始的 SAML 判斷提示不得包含值。InResponseTo

4. 具有使用者集區saml2/idpresponse端點Recipient值的SubjectConfirmationData元素，對於 SP 起始的 SAML，則為與原始 SAML 要求識別碼相符的InResponseTo值。

```
<saml2:SubjectConfirmationData InResponseTo="_dd0a3436-bc64-4679-a0c2-
cb4454f04184" NotOnOrAfter="Date-time stamp" Recipient="https://mydomain.us-
east-1.amazoncognito.com/saml2/idpresponse"/>
```

SP 發起的登入要求

當 授權端點 將使用者重新導向至您的 IdP 登入頁面，Amazon Cognito 會在 HTTP GET 請求的 URL 函數中包含 SAML 請求。SAML 要求包含使用者集區的相關資訊，包括 ACS 端點。您可以選擇性地將加密簽章套用至這些要求。

簽署要求並加密回應

每個具有 SAML 提供者的使用者集區都會為 Amazon Cognito 指派給 SAML 請求的數位簽章產生一個非對稱 key pair 和簽署憑證。您設定為支援加密 SAML 回應的每個外部 SAML IdP，都會導致 Amazon Cognito 為該提供者產生新的 key pair 和加密憑證。若要檢視和下載包含公開金鑰的憑證，請在 Amazon Cognito 主控台的 [登入體驗] 索引標籤中選擇您的 IdP。

若要透過來自您的使用者集區的 SAML 要求建立信任，請向您的 IdP 提供您的使用者集區 SAML 2.0 簽署憑證的複本。如果您未將 IdP 設定為接受已簽署的要求，您的 IdP 可能會忽略您的使用者集區已簽署的 SAML 請求。

1. Amazon Cognito 會將數位簽名套用至您的使用者傳遞至 IdP 的 SAML 請求。您的使用者集區會簽署所有單一登出 (SLO) 要求，而且您可以將使用者集區設定為針對任何 SAML 外部 IdP 簽署單一登入 (SSO) 要求。當您提供憑證副本時，您的 IdP 可以驗證使用者 SAML 要求的完整性。

2. 您的 SAML IdP 可以使用加密憑證加密 SAML 回應。當您設定具有 SAML 加密的 IdP 時，您的 IdP 只能傳送加密回應。

編碼非字母數字字元

Amazon Cognito 不接受您的 IdP 作為屬性值傳遞的 # 或等 4 位元組 UTF-8 字元。您可以對字元進行 Base64 編碼，將其作為文字傳遞，然後在應用程式中將其解碼。

在下列範例中，不會接受屬性宣告：

```
<saml2:Attribute Name="Name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:unspecified">
  <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">#</saml2:AttributeValue>
</saml2:Attribute>
```

與之前的範例相反，會接受以下屬性宣告：

```
<saml2:Attribute Name="Name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:unspecified">
  <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">8J+YkA==</saml2:AttributeValue>
</saml2:Attribute>
```

中繼資料端點必須具有有效的傳輸層安全性

如果您以 HTTPS 中繼資料端點 URL 來建立 SAML IdP 時，看到 `InvalidParameterException`，例如 "Error retrieving metadata from *<metadata endpoint>*"，請確定中繼資料端點已正確設定 SSL，並且有與其相關聯的有效 SSL 憑證。如需驗證憑證的詳細資訊，請參閱什麼是 SSL/ TLS 憑證？ 。

具有 IDP 啟動 SAML 的應用程式用戶端只能使用 SAML 登入

當您在應用程式用戶端中啟動支援 IdP 啟動登入的 SAML 2.0 IdP 支援時，您只能將其他 SAML 2.0 IdPs 新增至該應用程式用戶端。您無法將使用者集區中的使用者目錄和所有非 SAML 外部身分識別提供者新增至以這種方式設定的應用程式用戶端。

登出回應必須使用 POST 繫結

`/saml2/logout`端點接受`LogoutResponse`為`HTTP POST`請求。用戶池不接受`HTTP GET`具有綁定的註銷響應。

SAML 使用者名稱區分大小寫

當聯合身分使用者嘗試登入時，SAML 身分識別提供者 (IdP) 會在使用者的 SAML 宣告中NameId將唯一的傳遞給 Amazon Cognito。Amazon Cognito 會依 NameId 宣告識別 SAML 聯合身分使用者。無論使用者集區的區分大小寫設定為何，Amazon Cognito 會在傳遞其唯一且區分大小寫的宣告時，從 SAML IdP 辨識返回的聯合身分使用者。NameId如果您將一個類似 email 的屬性對應至 NameId，且您的使用者變更其電子郵件地址，則他們無法登入到您的應用程式。

從 IdP 屬性對應您 SAML 聲明中的 NameId，其具有不會變更的值。

例如，Carlos 在您不區分大小寫的使用者集區中，有一個來自 Active Directory Federation Services (ADFS) SAML 聲明的使用者描述檔，該聲明將 Carlos@example.com 作為 NameId 值傳遞。下次 Carlos 嘗試登入時，您的 ADFS IdP 將 carlos@example.com 作為 NameId 值傳遞。由於 NameId 必須是大小寫完全符合，因此登入不會成功。

如果您的使用者在其 NameID 變更後無法登入，請將其使用者描述檔從您的使用者集區中刪除。下次該使用者登入時，Amazon Cognito 會建立新的使用者描述檔。

主題

在使用者集區中新增和管理 SAML 身分識別提供者

下列程序示範如何在 Amazon Cognito 使用者集區中建立、修改和刪除 SAML 提供者。

AWS Management Console

您可以使用建立和刪除 SAML 身分識別提供者 (IdPs)。 AWS Management Console

建立 SAML IdP 之前，您必須具有從協力廠商 IdP 取得的 SAML 中繼資料文件。如需有關如何取得或產生所需之 SAML 中繼資料文件的詳細資訊，請參閱 設定您的第三方 SAML 身分提供者。

在您的使用者集區中設定 SAML 2.0 IdP

1. 前往 Amazon Cognito 主控台。若出現提示，請輸入 AWS 憑證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。
4. 選擇 Sign-in experience (登入體驗) 索引標籤。找到 Federated sign-in (聯合登入)，然後選擇 Add an identity provider (新增身分提供者)。
5. 選擇 SAML IdP。
6. 輸入提供者名稱。您可以在`identity_provider`請求參數中將此易記名稱傳遞給授權端點.
7. 輸入以逗號分隔的 Identifiers (識別符)。識別符通知 Amazon Cognito，應該檢查使用者在登入時輸入的電子郵件地址，然後將其導向至與其網域對應的供應商。
8. 如果您希望 Amazon Cognito 在使用者登出時傳送已簽署的登出請求給您的供應商，則請選擇 Add sign-out flow (新增登出流程)。您必須設定您的 SAML 2.0 IdP，以便將登出回應傳送至您設定託管 UI 時建立的 `https://mydomain.us-east-1.amazoncognito.com/saml2/logout` 端點。`saml2/logout` 端點會使用 POST 繫結。

   > **Note**
   >
   > 如果選取此選項，且您的 SAML IdP 需要已簽署的登出要求，您也必須提供來自使用者集區的簽署憑證的 SAML IdP。
   > SAML IdP 將處理簽署登出請求，以及將您的使用者從 Amazon Cognito 工作階段登出。

9. 選擇您的 IDP 起始 SAML 登入組態。作為安全性最佳作法，請選擇「僅接受 SP 起始的 SAML 宣告」。如果您已準備好安全地接受來路不明的 SAML 登入工作階段的環境，請選擇「接受 SP 起始和 IDP 起始的 SAML 判斷提示」。如需詳細資訊，請參閱 Amazon Cognito 使用者集區中的 SAML 工作階段啟動。
10. 選擇 Metadata document source (中繼資料文件來源)。如果您的 IdP 以公有 URL 提供 SAML 中繼資料，則可以選擇 Metadata document URL (中繼資料文件 URL)，然後輸入該公有 URL。否則，請選擇 Upload metadata document (上傳中繼資料文件)，然後選取您先前從供應商處下載的中繼資料檔案。

> **Note**
>
> 如果您的提供者具有公用端點，建議您輸入中繼資料文件 URL，而不是上傳檔案。Amazon Cognito 會自動從中繼資料 URL 重新整理中繼資料。通常每 6 小時或是在中繼資料過期之前 (取較早的時間) 重新整理中繼資料。

11. 對應 SAML 提供者與使用者集區之間的屬性，以將 SAML 提供者屬性對應至使用者集區中的使用者設定檔。在屬性對應中包含您的使用者集區必要屬性。

    例如，當您選擇 User pool attribute (使用者集區屬性) `email` 時，隨著 IdP SAML 聲明中顯示的 SAML 屬性名稱輸入 SAML 屬性名稱。如果 IdP 提供 SAML 聲明範例，則您可以使用這些聲明範例協助您找到名稱。有些 IdPs 使用簡單名稱，例如`email`，而另一些則使用類似下面的名稱。

    ```
    http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
    ```

12. 選擇建立。

API/CLI

您可以使用下列命令來建立及管理 SAML 身分提供者 (IdP)。

建立 IdP 並上傳中繼資料文件

- AWS CLI: `aws cognito-idp create-identity-provider`

  使用中繼資料檔案的範例：`aws cognito-idp create-identity-provider --user-pool-id` *`us-east-1_EXAMPLE`* `--provider-name=`*`SAML_provider_1`* `--provider-type SAML --provider-details file:///details.json --attribute-mapping` *`email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`*

  其中 `details.json` 包含：

  ```
  "ProviderDetails": {
        "MetadataFile": "<SAML metadata XML>",
        "IDPSignout" : "true",
        "RequestSigningAlgorithm" : "rsa-sha256",
        "EncryptedResponses" : "true",
  ```

```
      "IDPInit" : "true"
}
```

> **Note**
>
> 如果<SAML metadata XML>包含字元的任何實體"，您必須加入\為逸出字元:\"。

使用中繼資料 URL 的範例：`aws cognito-idp create-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-type SAML --provider-details MetadataURL=https://myidp.example.com/sso/saml/metadata --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API：CreateIdentityProvider

為 IdP 上傳新的中繼資料文件

- AWS CLI: `aws cognito-idp update-identity-provider`

使用中繼資料檔案的範例：`aws cognito-idp update-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

其中 `details.json` 包含：

```
"ProviderDetails": {
      "MetadataFile": "<SAML metadata XML>",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

> **Note**
>
> 如果<SAML metadata XML>包含字元的任何實體"，您必須加入\為逸出字元:\"。

使用中繼資料 URL 的範例：`aws cognito-idp update-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-details MetadataURL=https://myidp.example.com/sso/saml/metadata --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API：UpdateIdentityProvider

取得特定 IdP 的相關資訊

- AWS CLI: `aws cognito-idp describe-identity-provider`

  `aws cognito-idp describe-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1`

- AWS API：DescribeIdentityProvider

列出所有相關資訊的步驟 IdPs

- AWS CLI: `aws cognito-idp list-identity-providers`

  範例：`aws cognito-idp list-identity-providers --user-pool-id us-east-1_EXAMPLE --max-results 3`

- AWS API：ListIdentityProviders

刪除 IdP

- AWS CLI: `aws cognito-idp delete-identity-provider`

  `aws cognito-idp delete-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1`

- AWS API：DeleteIdentityProvider

設定 SAML IdP 以新增使用者集區做為依賴方

- 使用者集區服務供應商 URN 為：`urn:amazon:cognito:sp:us-east-1_EXAMPLE`。Amazon Cognito 要求受眾限制值與 SAML 回應中的此 URN 相符。將您的 IdP 設定為針對 IDP 到 SP 回應訊息使用下列 POST 繫結端點。

```
https://mydomain.us-east-1.amazoncognito.com/saml2/idpresponse
```

- 您的 SAML IdP 必須在 SAML 宣告中填入`NameID`您的使用者集區的任何必要屬性。 `NameID`用於唯一識別使用者集區中的 SAML 聯合身分使用者。您的 IdP 必須以一致且區分大小寫的格式傳遞每個使用者的 SAML 名稱 ID。使用者名稱 ID 值的任何變化都會建立新的使用者設定檔。

如要對您的 SAML 2.0 IDP 提供簽署憑證

- 若要從 Amazon Cognito 下載 IdP 可用來驗證 SAML 登出請求的公開金鑰複本，請選擇使用者集區的 [登入體驗] 索引標籤，選取您的 IdP，然後在 [檢視簽署憑證] 下選取 [下載為 .crt]。

您可以使用 Amazon Cognito 主控台刪除在使用者集區中設定的任何 SAML 供應商。

刪除 SAML 供應商

1. 登入 Amazon Cognito 主控台。
2. 在導覽窗格中，選擇 User Pools (使用者集區)，然後選擇您要編輯的使用者集區。
3. 選擇 [登入體驗] 索引標籤並找到 [聯合身分提供者登入]。
4. 選取 IdPs 您要刪除的 SAML 旁邊的圓形按鈕。
5. 出現 Delete identity provider (刪除身分提供者) 的提示時，請輸入 SAML 供應商的名稱以確認刪除，然後選擇 Delete (刪除)。

Amazon Cognito 使用者集區中的 SAML 工作階段啟動

Amazon Cognito 支援服務供應商啟動 (SP 啟動) 單一登入 (SSO) 和 IDP 起始的 SSO。最佳安全性做法是在您的使用者集區中實作 SP 起始的 SSO。SAML V2.0 技術概觀的第 5.1.2 節說明 SP 啟動的 SSO。Amazon Cognito 是您應用程式的身分提供者 (IdP)。此應用程式是一種服務供應商 (SP)，會擷取已驗證使用者的權杖。但是，當您使用第三方 IdP 對使用者進行身分驗證時，Amazon Cognito 就是 SP。當您的 SAML 2.0 使用者使用 SP 啟動的流程進行身份驗證時，他們必須始終向 Amazon Cognito 發出請求，然後重新導向至 IdP 進行身份驗證。

對於某些企業使用案例，存取內部應用程式是從在企業 IdP 託管的儀表板上的書籤開始。當使用者選取書籤時，IdP 將產生一個 SAML 回應，並將其傳送到 SP，以透過應用程式驗證使用者身分。

您可以在使用者集區中設定 SAML IdP，以支援 IdP 起始的 SSO。當您支援 IdP 啟動的身份驗證時，Amazon Cognito 無法驗證其是否已請求收到的 SAML 回應，因為 Amazon Cognito 不會使用 SAML 請求啟動身份驗證。在 SP 啟動的 SSO 中，Amazon Cognito 會設定狀態參數，以根據原始請求驗證 SAML 回應。使用 SP 啟動的登錄，您還可以防止跨站點請求偽造（CSRF）。

如需如何在您不希望使用者與使用者集區託管 UI 互動的環境中建置 SP 起始的 SAML 的範例，請參閱。範例案例：在企業儀表板中將 Amazon Cognito 應用程式加入書籤

主題

範例案例：在企業儀表板中將 Amazon Cognito 應用程式加入書籤

您可以在 SAML 或 OIDC Id P 儀表板中建立書籤，以提供 Amazon Cognito 使用者集區對 Web 應用程式的 SSO 存取。您可以透過不需要使用者使用託管 UI 登入的方式連結到 Amazon Cognito。若要這麼做，請將登入書籤新增至入口網站，以下列格式重新導向至 Amazon Cognito 使用者集區。授權端點

```
https://mydomain.us-east-1.amazoncognito.com/authorize?
response_type=code&identity_provider=MySAMLIdP&client_id=1example23456789&redire
www.example.com
```

> **Note**
>
> 您在對授權端點的請求中也可以使用 `idp_identifier` 參數，而不是 `identity_provider` 參數。IdP 識別碼是您在使用者集區中建立身分識別提供者時可以設定的替代名稱或電子郵件網域。請參閱 SAML 身分識別提供者名稱和識別碼。

當您在對 `/authorize` 的請求中使用適當的參數時，Amazon Cognito 會以無提示的方式開始 SP 啟動的登入流程，並重新導向您的使用者以使用您的 IdP 登入。

若要開始使用，請在您的使用者集區中新增 SAML IdP。建立一個應用程式用戶端，該用戶端使用您的 SAML IdP 進行登入，並具有您應用程式的 URL 做為授權回呼 URL。如需應用程式用戶端的詳細資訊，請參閱 使用者集區應用程式用戶端。

在將此經過驗證的存取部署到入口網站之前，請從託管的 UI 測試 SP 啟動的應用程式登入。如需如何在 Amazon Cognito 中設定 SAML IdP 的詳細資訊，請參閱 設定您的第三方 SAML 身分提供者。

下圖顯示的身分驗證流程模擬 IdP 啟動的 SSO。您的使用者可以透過您公司入口網站中的連結使用 Amazon Cognito 進行身分驗證。

在您符合要求之後，建立包[授權端點](#)含`identity_provider`或`idp_identifier`參數的書籤。使用者驗證進行如下。

1. 您的使用者登入 SSO IdP 儀表板。使用者有權存取的企業應用程式會填入此儀表板。
2. 您的使用者會選擇指向使用 Amazon Cognito 進行身分驗證的應用程式連結。在許多 SSO 入口網站中，您可以新增自訂應用程式連結。可用於建立指向您 SSO 入口網站公有 URL 連結的任何功能都將起作用。
3. SSO 入口網站中的自訂應用程式連結會將使用者引導到使用者集區 [授權端點](#)。此連結包含 `response_type`、`client_id`、`redirect_uri` 和 `identity_provider` 的參數。`identity_provider` 參數是您在使用者集區中指派給 IdP 的名稱。您也可以使用 `idp_identifier` 參數，而不是 `identity_provider` 參數。使用者從包含`idp_identifier`或`identity_provider`參數的連結存取您的同盟端點。此使用者會繞過登入頁面，並直接導覽以使用您的 IdP 進行身分驗證。如需命名 SAML 的詳細資訊 IdPs，請參閱[SAML 身分識別提供者名稱和識別碼](#)。

   範例網址

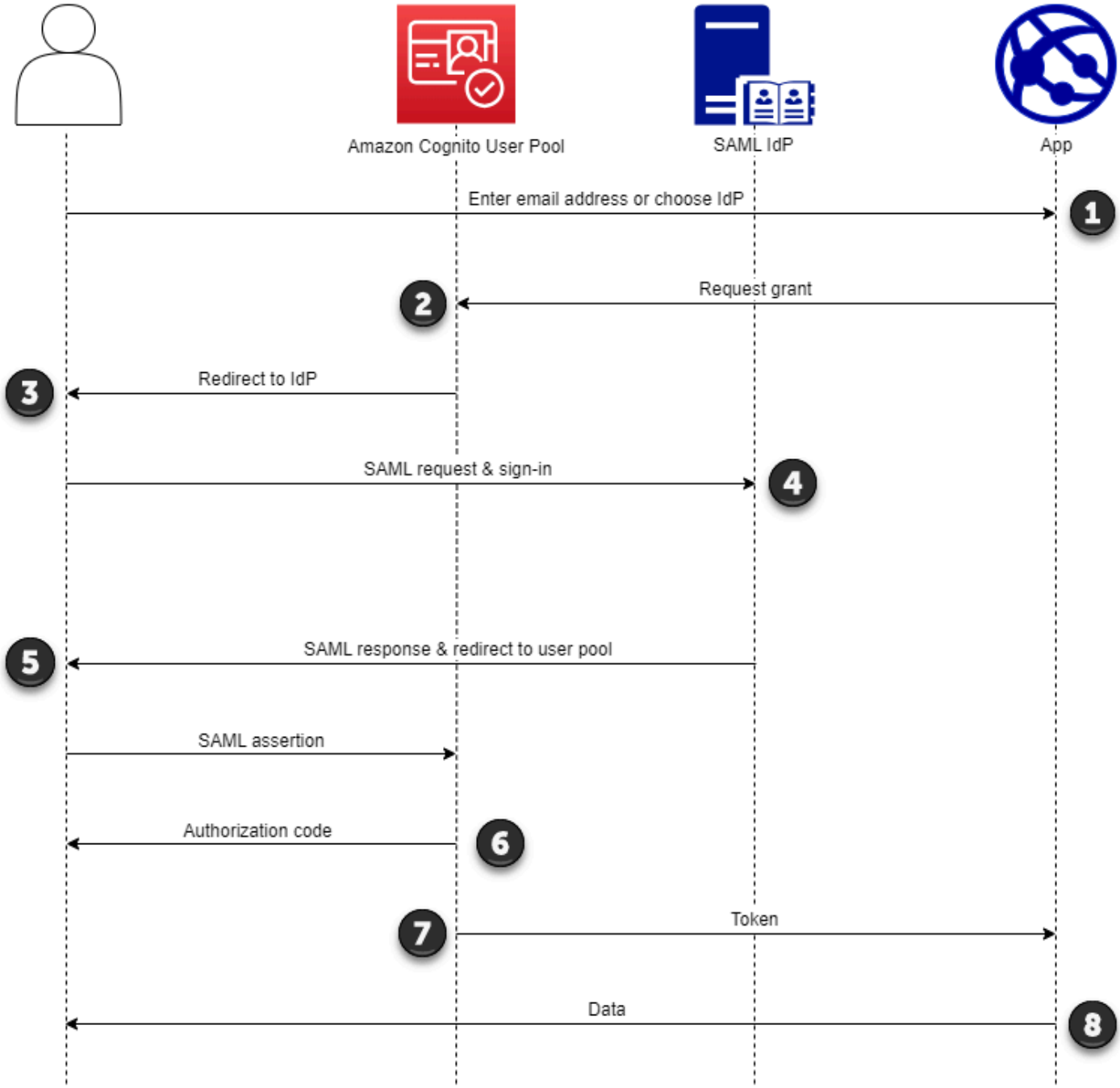
   ```
   https://mydomain.us-east-1.amazoncognito.com/authorize?
   response_type=code&
   identity_provider=MySAMLIdP&
   client_id=1example23456789&
   redirect_uri=https://www.example.com
   ```

4. Amazon Cognito 會將使用者工作階段重新導向至您的 IdP，並附加 SAML 請求。
5. 您的使用者登入到儀表板時，可能會收到來自您 IdP 的工作階段 Cookie。您的 IdP 會使用此 Cookie 以無提示的方式驗證使用者，將使用者重新導向到 Amazon Cognito `idpresponse` 端點並附加 SAML 回應。如果沒有作用中的工作階段，您的 IdP 會在發布 SAML 回應之前對使用者重新進行身分驗證。
6. Amazon Cognito 會驗證 SAML 回應，並根據 SAML 聲明建立或更新使用者描述檔。
7. Amazon Cognito 會使用授權碼將使用者重新導向至您的內部應用程式。您已將內部應用程式 URL 設定為應用程式用戶端的已授權重新導向 URL。
8. 您的應用程式會交換 Amazon Cognito 字符的授權碼。如需詳細資訊，請參閱 [權杖端點](#)。

使用 SP 啟動式 SAML 登入

最佳做法是將 service-provider-initiated (SP 啟動) 登入實作至您的使用者集區。Amazon Cognito 會啟動您的使用者工作階段，並將其重新導向至您的 IdP。使用此方法，您可以最大程度地控制誰提出登錄請求。您也可以在特定情況下允許 IDP 啟動的登入。如需詳細資訊，請參閱 Amazon Cognito 使用者集區中的 SAML 工作階段啟動。

下列程序顯示使用者如何透過 SAML 提供者登入您的使用者集區。



1. 您的使用者在登入頁面中輸入他們的電子郵件地址。若要判斷使用者重新導向至其 IdP，您可以在自訂應用程式中收集其電子郵件地址，或在 Web 檢視中叫用託管 UI。您可以將託管的 UI 設定為顯示 IdPs 或僅提示輸入電子郵件地址的清單。
2. 您的應用程序調用您的用戶池重定向端點，並請求具有與應用程序對應的客戶端 ID 以及與用戶對應的 IdP ID 的會話。

3. Amazon Cognito 會使用元素中的 SAML 請求 (選擇性地簽署) 將您的使用者重新導向至 IdP。`AuthnRequest`
4. IdP 會以互動方式驗證使用者，或使用瀏覽器 Cookie 中記住的工作階段來驗證使用者。
5. IdP 會將您的使用者重新導向至您的使用者集區 SAML 回應端點，並在其 POST 裝載中使用選擇性加密的 SAML 宣告。

> **Note**
>
> Amazon Cognito 會取消 5 分鐘內未收到回應的工作階段，並將使用者重新導向至託管的使用者介面。當您的使用者遇到此結果時，他們會收到`Something went wrong`錯誤訊息。

6. 在驗證 SAML 宣告並對應回應中宣告中的使用者屬性後，Amazon Cognito 會在內部建立或更新使用者集區中的使用者設定檔。通常，您的使用者集區會將授權碼傳回給使用者的瀏覽器工作階段。
7. 您的用戶將其授權碼提供給您的應用程序，該應用程序將代碼交換為 JSON Web 令牌（JWT）。
8. 您的應用程序接受並處理用戶的 ID 令牌作為身份驗證，使用其訪問令牌對資源生成授權請求，並存儲其刷新令牌。

當用戶身份驗證並接收授權碼授予時，用戶池返回 ID，訪問和刷新令牌。ID 權杖是 OIDC 型身分識別管理的驗證物件。訪問令牌是具有 OAuth 2.0 範圍的授權對象。刷新令牌是在用戶當前令牌過期時生成新 ID 和訪問令牌的對象。您可以在用戶池應用程序客戶端中配置用戶令牌的持續時間。

您也可以選擇刷新令牌的持續時間。使用者的重新整理權杖到期後，必須重新登入。如果使用者透過 SAML IdP 進行驗證，則使用者的工作階段持續時間是由其權杖到期而設定，而不是使用其 IdP 的工作階段到期。您的應用程序必須存儲每個用戶的刷新令牌，並在其過期時更新其會話。託管的用戶界面在瀏覽器 cookie 中維護用戶會話，該 cookie 有效期為 1 小時。

使用 IDP 起始的 SAML 登入

當您為 IDP 起始的 SAML 2.0 登入設定身分識別提供者時，您可以向使用者集區網域中的`saml2/idpresponse`端點顯示 SAML 宣告，而不需要在. 授權端點 具有此設定的使用者集區會接受來自要求應用程式用戶端支援之使用者集區外部身分識別提供者的 IDP 起始 SAML 宣告。下列步驟說明使用 IDP 起始的 SAML 2.0 提供者設定和登入的整體程序。

1. 建立或指定使用者集區和應用程式用戶端。
2. 在您的使用者集區中建立 SAML 2.0 IdP。

3. 設定您的 IdP 以支援 IdP 初始化。IDP 起始的 SAML 引入了其他 SSO 提供者不受限制的安全性考量。因此，您無法將非 SAML IdPs（包括使用者集區本身）新增至使用具有 IDP 起始登入之 SAML 提供者的任何應用程式用戶端。
4. 將 IDP 起始的 SAML 提供者與使用者集區中的應用程式用戶端建立關聯。
5. 將您的使用者導向至 SAML IdP 的登入頁面，並擷取 SAML 宣告。
6. 使用其 SAML 判斷提示將您的使用者導向至您的使用者集區`saml2/idpresponse`端點。
7. 接收 JSON 網絡令牌（JWT）。

若要在使用者集區中接受來路不明的 SAML 宣告，您必須考慮其對應用程式安全性的影響。當您接受 IDP 發起的請求時，可能會出現請求欺騙和 CSRF 嘗試。雖然您的使用者集區無法驗證 IDP 起始的登入工作階段，但 Amazon Cognito 會驗證您的請求參數和 SAML 宣告。

此外，您的 SAML 宣告不得包含`InResponseTo`索賠，且必須在前 6 分鐘內發出。

您必須將具有 IDP 起始的 SAML 的請求提交給您的. `/saml2/idpresponse` 對於 SP 起始和託管的 UI 授權請求，您必須提供參數，以識別請求的應用程序客戶端，範圍，重定向 URI 和其他詳細信息作為請求中`HTTP GET`的查詢字符串參數。但是，對於 IdP 起始的 SAML 宣告，請求的詳細資料必須在要求主體中格式化為`RelayState`參數。`HTTP POST`要求主體也必須包含您的 SAML 判斷提示做為`SAMLResponse`參數。

以下是 IdP 起始之 SAML 提供者的範例要求。

```
POST /saml2/idpresponse HTTP/1.1
User-Agent: USER_AGENT
Accept: */*
Host: example.auth.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded

SAMLResponse=[Base64-encoded SAML assertion]&RelayState=identity_provider
%3DMySAMLIdP%26client_id%3D1example23456789%26redirect_uri%3Dhttps%3A%2F
%2Fwww.example.com%26response_type%3Dcode%26scope%3Demail%2Bopenid%2Bphone

HTTP/1.1 302 Found
Date: Wed, 06 Dec 2023 00:15:29 GMT
Content-Length: 0
x-amz-cognito-request-id: 8aba6eb5-fb54-4bc6-9368-c3878434f0fb
Location: https://www.example.com?code=[Authorization code]
```

AWS Management Console

若要為 IdP 起始的 SAML 設定 IdP

1. 建立使用者集區、應用程式用戶端和 SAML 身分識別提供者。
2. 取消所有社交和 OIDC 身分提供者與您的應用程式用戶端的關聯 (如果有關聯)。
3. 導覽至使用者集區的 [登入體驗] 索引標籤。
4. 在「同盟身分提供者登入」下，編輯或新增 SAML 提供者。
5. 在 IDP 起始的 SAML 登入下，選擇接受 SP 起始和 IDP 起始的 SAML 宣告。
6. 選擇儲存變更。

API/CLI

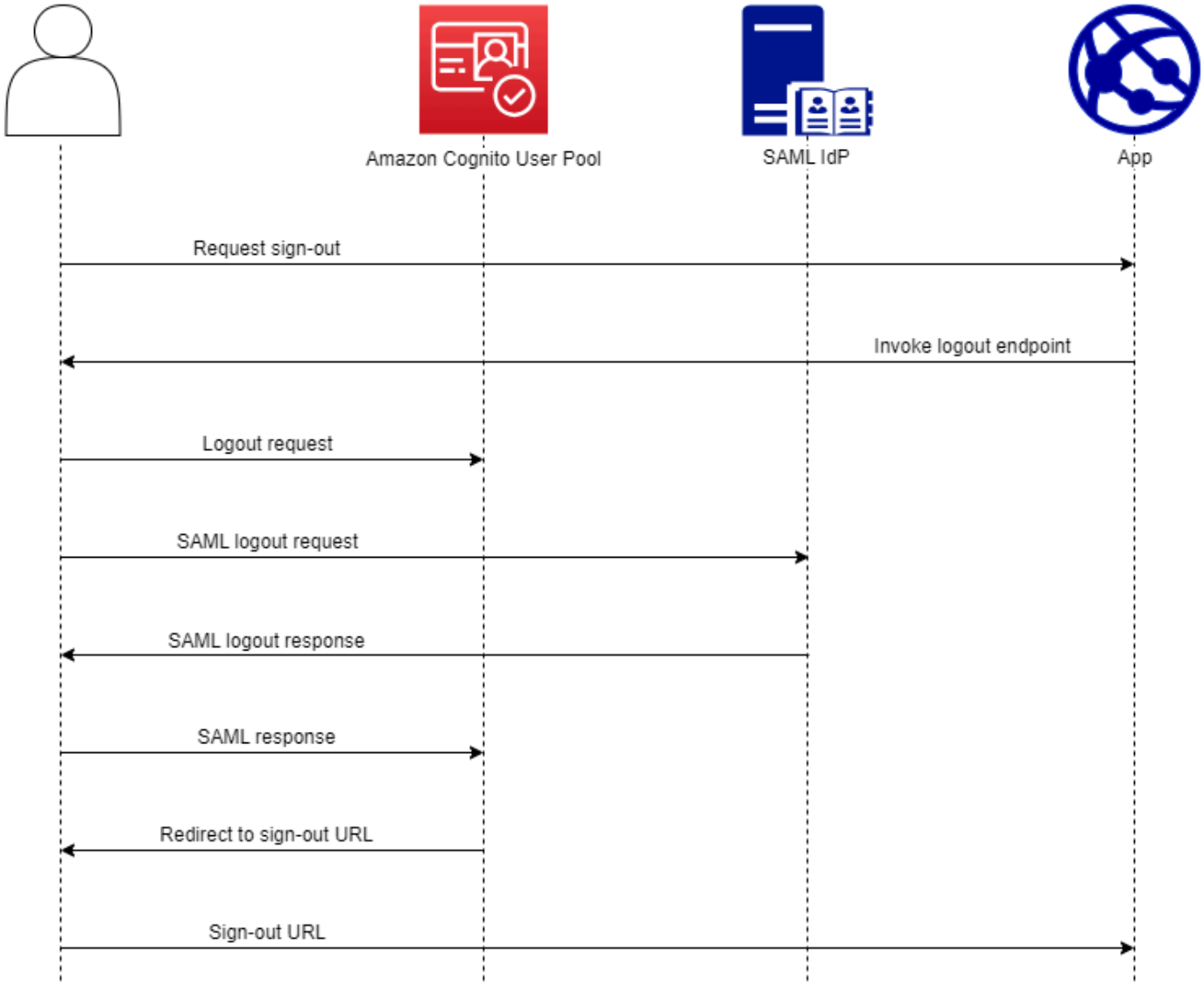
若要為 IdP 起始的 SAML 設定 IdP

使用CreateIdentityProvider或 UpdateIdentityProviderAPI 要求中的`IDPInit`參數來設定 IdP 起始的 SAML。以下是支援 IdP 起始 SAML `ProviderDetails` 的 IdP 範例。

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML 登出流程

Amazon Cognito 支援 SAML 2.0 單一登出。當您將 SAML IdP 設定為支援登出流程時，Amazon Cognito 會透過已簽署的 SAML 登出請求將您的使用者重新導向至您的 IdP。Amazon Cognito 會從 IdP 中繼資料中的 `SingleLogoutService` URL 判斷重新導向位置。Amazon Cognito 使用您的使用者集區簽署憑證來簽署登出請求。



當您將具有 SAML 工作階段的使用者導向至您的使用者集區`/logout`端點時，Amazon Cognito 會透過下列請求將您的 SAML 使用者重新導向至 IdP 中繼資料中指定的 SLO 端點。

```
https://[SingleLogoutService endpoint]?
SAMLRequest=[encoded SAML request]&
RelayState=[RelayState]&
SigAlg=http://www.w3.org/2001/04/xmldsig-more#rsa-sha256&
Signature=[User pool RSA signature]
```

然後，您的用戶使用其 IdP LogoutResponse 中的 a 返回到您的saml2/logout端點。您的 IdP 必須傳LogoutResponse送HTTP POST請求。然後，Amazon Cognito 會從其初始登出請求將其重新導向至重新導向目的地。

您的 SAML 提供者可能會傳送LogoutResponse一個以上AuthnStatement的檔案。此類型回應AuthnStatement中的第一個必須符合最初驗證使用者的 SAML 回應sessionIndex中的。sessionIndex如果在sessionIndex任何其他項目中AuthnStatement，Amazon Cognito 將無法辨識工作階段，而且您的使用者也不會登出。

AWS Management Console

若要設定 SAML 登出

1. 建立使用者集區、應用程式用戶端和 SAML IdP。
2. 建立或編輯 SAML 身分識別提供者時，在身分識別提供者資訊下，勾選標題為 [新增登出流程] 的核取方塊。
3. 在使用者集區的 [登入體驗] 索引標籤的 [同盟身分提供者登入] 下，選擇您的 IdP 並找到 [簽署憑證]。
4. 選擇下載為 .crt。
5. 設定您的 SAML 提供者以支援 SAML 單一登出和要求簽署，並上傳使用者集區簽署憑證。您的 IdP 必須重新導向至您/saml2/logout的使用者集區網域中。

API/CLI

若要設定 SAML 登出

使用CreateIdentityProvider或 UpdateIdentityProviderAPI 請求的IDPSignout參數設定單一登出。以下是支援 SAML 單ProviderDetails一登出的 IdP 範例。

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",,
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML 簽章與加密

Amazon Cognito 支援已簽署的 SAML 請求和加密的 SAML 回應，以供登入和登出使用。使用者集區 SAML 操作期間的所有加密操作都必須使用 Amazon Cognito 產生的 user-pool-provided 金鑰產生簽名和密文。目前，您無法設定使用者集區來簽署要求或使用外部金鑰接受加密的宣告。

> **Note**
>
> 您的使用者集區憑證有效期為 10 年。Amazon Cognito 每年會為您的使用者集區產生一次新的簽署和加密憑證。Amazon Cognito 會在您請求簽署憑證時傳回最新的憑證，並使用最新的簽署憑證簽署請求。您的 IdP 可以使用任何未過期的使用者集區加密憑證來加密 SAML 判斷提示。您之前的證書在整個持續時間內仍然有效。最佳做法是每年更新提供者組態中的憑證。

主題

- 從您的 IdP 接受加密的 SAML 回應
- 簽署 SAML 要求

從您的 IdP 接受加密的 SAML 回應

當使用者登入和登出時，Amazon Cognito 和您的 IdP 可以在 SAML 回應中建立機密性。Amazon Cognito 會為您在使用者集區中設定的每個外部 SAML 提供者指派公開私有 RSA key pair 和憑證。當您為使用者集區 SAML 提供者啟用回應加密時，您必須將憑證上傳至支援加密 SAML 回應的 IdP。在 IdP 開始使用提供的金鑰加密所有 SAML 宣告之前，您與 SAML IdP 的使用者集區連線無法運作。

以下是加密 SAML 登入流程的概觀。

1. 您的使用者開始登入並選擇他們的 SAML IdP。
2. 您的使用者集區會透過 SAML 登入要求將您的使用者授權端點重新導向至其 SAML IdP。您的使用者集區可以選擇性地隨附此要求，並附上簽章，以啟用 IdP 的完整性驗證。當您想要簽署 SAML 要求時，必須將 IdP 設定為接受使用者集區已使用簽署憑證中的公開金鑰簽署的要求。
3. SAML IdP 會在您的使用者中登入並產生 SAML 回應。IdP 會使用公開金鑰加密回應，並將您的使用者重新導向至使用者集區`/saml2/idpresponse`端點。IdP 必須按照 SAML 2.0 規格所定義的回應加密。如需詳細資訊，請參閱 OASIS 安全性判斷提示標記語言 (SAML) V2.0 的宣告和通訊協定`Element <EncryptedAssertion>`中的。
4. 您的使用者集區會使用私密金鑰和登入您的使用者來解密 SAML 回應中的密文。

> **Important**
>
> 當您為使用者集區中的 SAML IdP 啟用回應加密時，您的 IdP 必須使用提供者專屬的公開金鑰來加密所有回應。Amazon Cognito 不接受來自您設定為支援加密的 SAML 外部 IdP 的未加密 SAML 回應。

使用者集區中的任何外部 SAML IdP 都可以支援回應加密，而且每個 IdP 都會收到自己的 key pair。

AWS Management Console

若要設定 SAML 回應加密

1. 建立使用者集區、應用程式用戶端和 SAML IdP。
2. 建立或編輯 SAML 身分識別提供者時，在「簽署要求和加密回應」下，勾選標題為「需要來自此提供者的加密 SAML 宣告」的核取方塊。
3. 在使用者集區的 [登入體驗] 索引標籤的 [同盟身分提供者登入] 下，選取您的 SAML IdP，然後選擇 [檢視加密憑證]。
4. 選擇下載為 .crt，並將下載的檔案提供給您的 SAML IdP。將您的 SAML IdP 設定為使用憑證中的金鑰加密 SAML 回應。

API/CLI

若要設定 SAML 回應加密

使用CreateIdentityProvider或 UpdateIdentityProviderAPI 請求的`EncryptedResponses`參數設定回應加密。以下是支援要求簽署`ProviderDetails`的 IdP 範例。

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

簽署 SAML 要求

能夠向您的 IdP 證明 SAML 2.0 請求的完整性，這是 Amazon Cognito SP 啟動的 SAML 登入的一項安全優勢。每個具有網域的使用者集區都會收到一個使用者集區 X.509 簽署憑證。使用此憑證中的公開金鑰，使用者集區會將加密簽章套用至您的使用者集區在使用者選取 SAML IdP 時產生的登出要求。您可以選擇配置應用程序客戶端以簽署 SAML 登錄請求。當您簽署 SAML 請求時，IdP 可以檢查請求的 XML 中繼資料中的簽章是否與您提供的使用者集區憑證中的公開金鑰相符。

AWS Management Console

若要設定 SAML 要求簽章

1. 建立使用者集區、應用程式用戶端和 SAML IdP。
2. 建立或編輯 SAML 身分識別提供者時，在「簽署要求並加密回應」下，勾選標題為「向此提供者簽署 SAML 要求」的核取方塊。
3. 在使用者集區的 [登入體驗] 索引標籤的 [同盟身分識別提供者登入] 下，選擇 [檢視簽署憑證]。
4. 選擇下載為 .crt，並將下載的檔案提供給您的 SAML IdP。設定您的 SAML IdP 以驗證內送 SAML 要求的簽章。

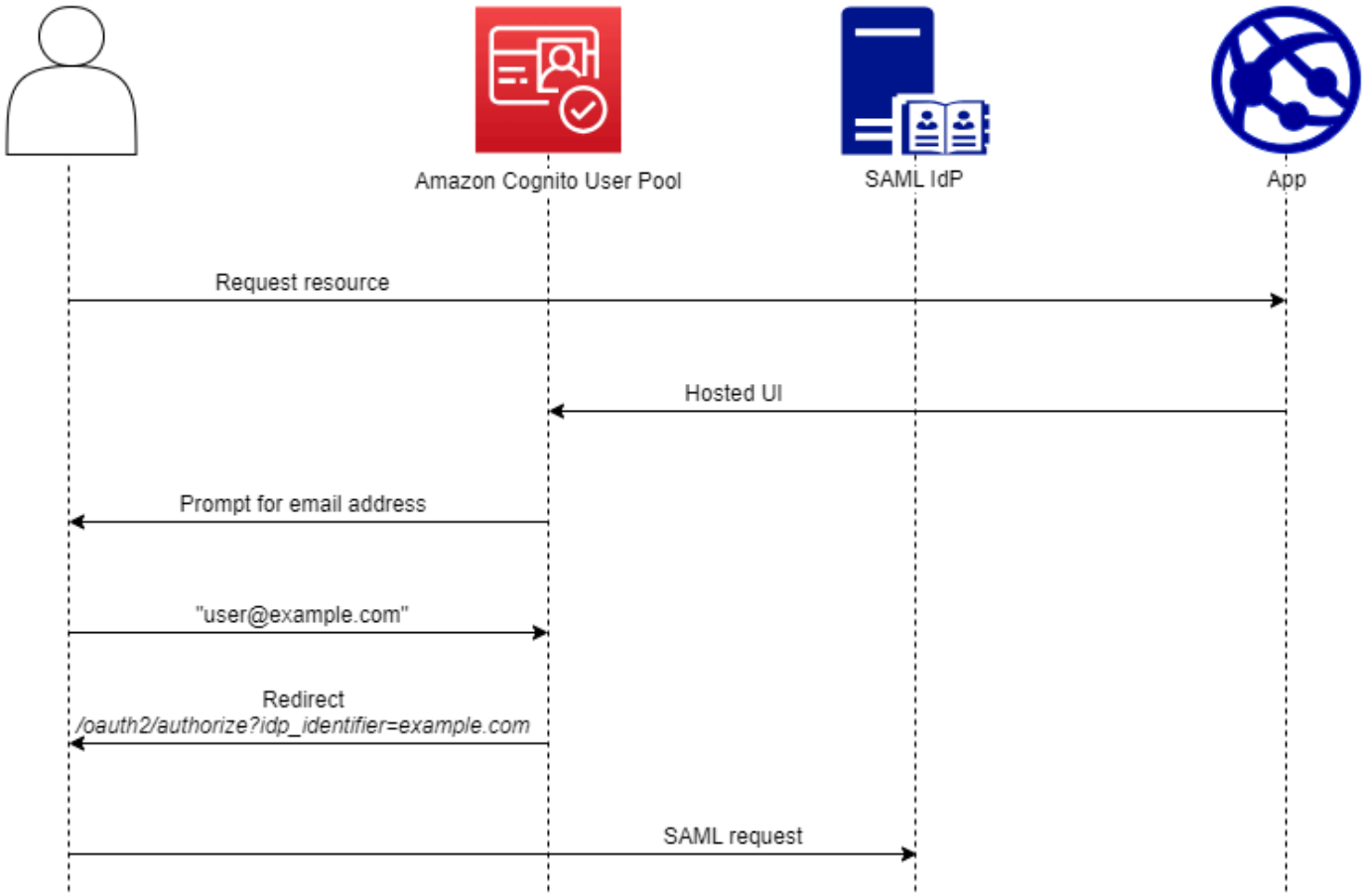
API/CLI

若要設定 SAML 要求簽章

使用CreateIdentityProvider或 UpdateIdentityProviderAPI 請求的`RequestSigningAlgorithm`參數設定要求簽署。以下是支援要求簽署`ProviderDetails`的 IdP 範例。

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML 身分識別提供者名稱和識別碼



當您為 SAML 身分提供者命名 (IdPs) 並指派 IdP 識別碼時，您可以將 SP 起始的登入和登出要求流程自動化給該提供者。如需提供者名稱之字串條件約束的相關資訊，請參閱的`ProviderName`屬性CreateIdentityProvider。

您也可以為 SAML 提供者選擇多達 50 個識別碼。識別碼是您使用者集區中 IdP 的易記名稱，且在使用者集區中必須是唯一的。如果您的 SAML 識別碼與使用者的電子郵件網域相符，Amazon Cognito 託管的 UI 會請求每個使用者的電子郵件地址、評估其電子郵件地址中的網域，並將其重新導向至與其網域對應的 IdP。由於相同的組織可以擁有多個網域，因此單一 IdP 可以有多個識別碼。

無論您是否使用電子郵件網域識別碼，都可以在多租用戶應用程式中使用識別碼，將使用者重新導向至正確的 IdP。當您想要完全略過託管的 UI 時，您可以自訂呈現給使用者的連結，使其透過授權端點直接重新導向至其 IdP。若要使用識別碼登入使用者並重新導向至其 IdP，請在其初始授權要求的要求參數`idp_identifier=`*`myidp.example.com`*中以格式加入識別碼。

將使用者傳遞至 IdP 的另一種方法是以下列 URL 格式填入 IdP 名稱的參數`identity_provider`。

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
identity_provider=MySAMLIdP&
client_id=1example23456789&
redirect_uri=https://www.example.com
```

使用者使用您的 SAML IdP 登入後，您的 IdP 會將內文中的 SAML 回應重新導向至您的HTTP POST端點。/saml2/idpresponseAmazon Cognito 會處理 SAML 宣告，如果回應中的宣告符合預期，則會重新導向至您的應用程式用戶端回呼 URL。用戶以這種方式完成身份驗證後，他們僅針對您的 IdP 和您的應用程序與網頁進行了交互。

使用網域格式的 IdP 識別碼時，Amazon Cognito 託管的 UI 會在登入時請求電子郵件地址，然後在電子郵件網域與 IdP 識別碼相符時，將使用者重新導向至其 IdP 的登入頁面。舉例來說，您建置的應用程式需要兩家不同公司的員工登入。第一家公司， AnyCompany A，擁有exampleA.com和exampleA.co.uk。第二家公司 AnyCompany B 擁有exampleB.com。在此範例中，您為每個公司設定了兩個 IdPs，一個，如下所示：

- 針對 IdP A，您定義識別符 exampleA.com 和 exampleA.co.uk。
- 針對 IdP B，您會定義識別符 exampleB.com。

在您的應用程序中，調用應用程序客戶端的託管 UI 以提示每個用戶輸入其電子郵件地址。Amazon Cognito 會從電子郵件地址衍生網域、使用網域識別碼將網域與 IdP 建立關聯，然後透過請求將您的使用者重新導向至包含請求參數的正確 IdP。[授權端點] idp_identifier例如，如果使用者進入bob@exampleA.co.uk，與其互動的下一個頁面就是 IdP 登入頁面的位https://auth.exampleA.co.uk/sso/saml置。

您也可以獨立實作相同的邏輯。在您的應用程序中，您可以構建一個自定義表單，以收集用戶輸入並根據自己的邏輯將其與正確的 IdP 相關聯。您可以為每個應用程序租戶生成自定義應用程序門戶，其中每個鏈接到授權端點的請求參數中具有租戶的標識符。

若要收集電子郵件地址並剖析託管 UI 中的網域，請至少為指派給應用程式用戶端的每個 SAML IdP 指派一個識別碼。默認情況下，託管的 UI 登錄屏幕會為您分配給應用程序客戶端的每個屏幕顯示一個按鈕。 IdPs 但是，如果您已成功指派識別碼，您的託管 UI 登入頁面會如下圖所示。

在託管 UI 中進行網域剖析需要您使用網域做為 IdP 識別碼。如果您將任何類型的識別碼指派給應用程式用戶端的每個 SAML IdPs ，該應用程式的託管 UI 將不再顯示 IDP 選取按鈕。當您想要使用電子郵件剖析或自訂邏輯產生重新導向時，請新增 SAML 的 IdP 識別碼。當您想要生成靜默重定向並希望託管 UI 顯示列表時 IdPs，請不要分配標識符並在授權請求中使用請求參數。identity_provider

- 如果您只將一個 SAML IdP 指派給應用程式用戶端，則託管的 UI 登入頁面會顯示一個使用該 IdP 登入的按鈕。
- 如果您為應用程式用戶端啟用的每個 SAML IdP 指派識別碼，則託管 UI 登入頁面中會顯示電子郵件地址的使用者輸入提示。
- 如果您有多個識別碼 IdPs ，但未將識別碼指派給所有識別碼，則託管 UI 登入頁面會顯示一個按鈕，以便使用每個指派的 IdP 登入。
- 如果您將標識符分配給您的， IdPs 並且希望託管 UI 顯示選擇的 IdP 按鈕，請向您的應用程序客戶端添加一個沒有標識符的新 IdP，或創建一個新的應用程序客戶端。您也可以刪除現有的 IdP，然後在沒有識別碼的情況下再次新增它。如果您建立新的 IdP，您的 SAML 使用者將會建立新的使用者設定檔。這種複製的作用中使用者可能會在您變更 IdP 組態的當月產生計費影響。

如需 IdP 設定的詳細資訊，請參閱 為您的使用者集區配置身分提供者。

設定您的第三方 SAML 身分提供者

若要設定第三方 SAML 2.0 身分識別提供者 (IdP) 解決方案以與 Amazon Cognito 使用者集區的聯合運作，您必須將 SAML IdP 設定為重新導向至下列宣告消費者服務 (ACS) URL：。`https://mydomain.us-east-1.amazoncognito.com/saml2/idpresponse`如果您的使用者集區具有 Amazon Cognito 網域，您可以在 Amazon Cognito 主控台使用者集區的 App integration (應用程式整合) 標籤中找到使用者集區網域路徑。

某些 SAML IdPs 要求您在表`urn:amazon:cognito:sp:us-east-1_EXAMPLE`單中`urn`提供 (也稱為對象 URI 或 SP 實體識別碼)。您可以在 Amazon Cognito 主控台的使用者集區概觀下找到您的使用者集區 ID。

您也必須設定 SAML IdP，為您在使用者集區中指定為必要屬性的任何屬性提供值。通常，這`email`是使用者集區的必要屬性，在這種情況下，SAML IdP 必須在其 SAML 宣告中提供某`email`種形式的宣告，而且您必須將宣告對應至該提供者的屬性。

下列第三方 SAML 2.0 IdP 解決方案的組態資訊是開始與 Amazon Cognito 使用者集區設定聯盟的好地方。如需最新資訊，請直接參閱供應商的說明文件。

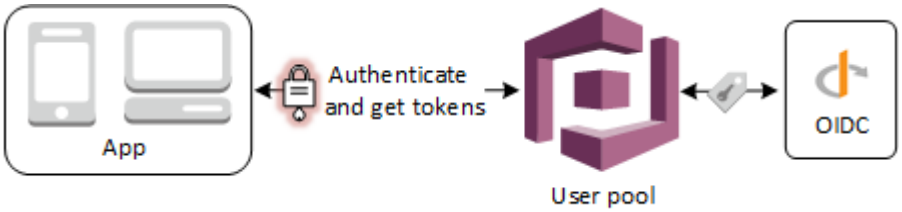
若要簽署 SAML 要求，您必須將 IdP 設定為信任使用者集區簽署憑證所簽署的要求。若要接受加密的 SAML 回應，您必須將 IdP 設定為加密使用者集區的所有 SAML 回應。您的提供商將提供有關配置這些功能的文檔。如需 Microsoft 的範例，請參閱設定 Microsoft Entra SAML 權杖加密。

> **Note**
>
> Amazon Cognito 只需要您的身分識別供應商中繼資料文件。您的供應商可能會提供與 SAML 2.0 AWS 帳戶 聯合的組態資訊；此資訊與 Amazon Cognito 整合無關。

解決方案	其他資訊
Microsoft Active Directory Federation Services (AD FS)	同盟中繼資料總
Okta	如何下載用於 SAML 應用程式整合的 IdP 中繼資料和 SAML 簽署憑證
Auth0	將 Auth0 設定為 SAML 身分識別提供者
平身份（PingFederate）	匯出 SAML 中繼資料 PingFederate
JumpCloud	SAML 組態注意事項
SecureAuth	SAML 應用程式整合

搭配使用者集區使用 OIDC 身分識別提供者

您可以讓已經擁有 OpenID Connect (OIDC) 身分識別提供者 () 帳戶的使用者跳過註冊步驟，並使用現有帳戶登入您的應用程式。IdPsAmazon Cognito 使用內建的託管 Web UI，可以為所有已進行身分驗證的使用者提供字符處理和管理。這樣，您的後端系統可以對一組使用者集區權杖進行標準化。



> **Note**
>
> Amazon Cognito 使用者集區提供透過第三方 (聯合身分) 登入。這項功能與透過 Amazon Cognito 身分集區 (聯合身分) 登入無關。

您可以在、透過或使用使用者集區 API 方法 AWS Management Console，將 OIDC IdP 新增至您的使用者集區。 AWS CLICreateIdentityProvider

主題

必要條件

開始之前，您必須準備好以下事項：

- 具有應用程式用戶端和使用者集區網域的使用者集區。如需詳細資訊，請參閱建立使用者集區。
- 具有以下組態的 OIDC IdP：
  - 支援 `client_secret_post` 用户端身分驗證。Amazon Cognito 不會在 OIDC 探索端點上為您的 IdP 檢查 `token_endpoint_auth_methods_supported` 宣告。Amazon Cognito 不支援 `client_secret_basic` 用户端身分驗證。如需用户端身分驗證的詳細資訊，請參閱 OpenID Connect 文件中的用戶端身分驗證。
  - 僅對 OIDC 端點使用 HTTPS，例如 `openid_configuration`、`userInfo` 和 `jwks_uri`。
  - 僅對 OIDC 端點使用 TCP 連接埠 80 和 443。
  - 僅使用 HMAC-SHA、ECDSA 或 RSA 演算法對 ID 權杖進行簽署。
  - 在其 `jwks_uri` 發佈金鑰 ID `kid` 宣告並在其權杖中包含 `kid` 宣告。

步驟 1：向 OIDC IdP 註冊

在與 Amazon Cognito 建立 OIDC IdP 之前，您必須將您的應用程式註冊至 OIDC IdP 以接收用戶端 ID 和用戶端密碼。

向 OIDC IdP 註冊

1. 向 OIDC IdP 建立開發人員帳戶。

與 OIDC 的連結 IdPs

OIDC IdP	安裝方法	OIDC 探索 URL
Salesforce	安裝 Salesforce 身分提供者	https://login.salesforce.com
Ping 身分	安裝 Ping Identity 身分提供者	https://*Your Ping domain address*:9031/idp/userinfo.openid 例如：`https://pf.company.com:9031/idp/userinfo.openid`
Okta	安裝 Okta 身分提供者	https://*Your Okta subdomain*.oktapreview.com 或 `https://`*Your Okta subdomain*`.okta.com`
Microsoft Azure Active Directory (Azure AD)	安裝 Microsoft Azure AD 身分提供者	`https://login.microsoftonline.com/ {tenant}/v2.0`
Google	安裝 Google 身分提供者	https://accounts.google.com **Note** Amazon Cognito 提供 Google 作為整合的社交登入 IdP。我們建議您使用整合的 IdP。請參閱 透過使用者集區使用社交身分提供者。

2. 透過您的 OIDC IdP 來註冊具有 `/oauth2/idpresponse` 端點的使用者集區網域 URL。這可確保稍後當 OIDC IdP 驗證使用者時，可從 Amazon Cognito 接受它。

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
```

3. 使用您的 Amazon Cognito 使用者集區註冊您的回呼 URL。這是您的使用者成功身分驗證後，Amazon Cognito 會將其重新導向的頁面 URL。

```
https://www.example.com
```

4. 選擇您的範圍。必須填入範圍 openid。需有 email 範圍，以授予 email 和 email_verified 宣告的存取權。
5. OIDC IdP 為您提供用戶端 ID 和用戶端秘密。您在使用者集區中設定 OIDC IdP 會用到它們。

範例：使用 Salesforce 做為使用者集區的 OIDC IdP

當您想要在 OIDC 相容的 IdP (例如 Salesforce) 和您的使用者集區之間建立信任，請使用 OIDC IdP。

1. 在 Salesforce 開發人員網站上建立帳戶。
2. 透過您在之前步驟設定的開發人員帳戶登入。
3. 在 Salesforce 頁面上，執行下列其中一項操作：
   - 如果使用的是 Lightning Experience，請選擇設定齒輪圖示，然後選擇 Setup Home (設定首頁)。
   - 如果是使用 Salesforce Classic，則使用者界面頁首會顯示 Setup (設定)，請選擇此選項。
   - 如果是使用 Salesforce Classic，但頁首卻沒有出現 Setup (設定)，則請從最上面導覽列中選擇您的姓名，然後從下拉式清單中選擇 Setup (設定)。
4. 在左側導覽列中，選擇 Company Settings (公司設定)。
5. 在導覽列，選擇 Domain (網域)、輸入網域，然後選擇 Create (建立)。
6. 在左側導覽列，在 Platform Tools (平台工具) 下選擇 Apps (應用程式)。
7. 選擇 App Manager (應用程式管理員)。
8. a. 選擇 New connected app (新的連線應用程式)。
   b. 完成必要欄位。

      在 Start URL (開始 URL) 下方，使用您的 Salesforce IdP 登入的使用者集區網域其 `/authorize` 端點中。當您的使用者存取連線應用程式時，Salesforce 會將他們引導到此 URL 以完成登入。然後，Salesforce 會將使用者重新導向至已與您應用程式用戶端相關聯的回呼 URL。

```
https://mydomain.us-east-1.amazoncognito.com/authorize?
response_type=code&client_id=<your_client_id>&redirect_uri=https://
www.example.com&identity_provider=CorpSalesforce
```

c. 啟用 OAuth settings (OAuth 設定)，然後在 Callback URL (回呼 URL) 中輸入您使用者集區網域之 `/oauth2/idpresponse` 端點的 URL。這是 Salesforce 核發 Amazon Cognito 用於交換 OAuth 權杖的授權碼所在的 URL。

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/idpresponse
```

9. 選擇您的範圍。您必須包含範圍 OpenID。若要授予 email 與 email_verified 宣告的存取權，請新增 email 範圍。以空格區隔範圍。
10. 選擇 Create (建立)。

    在 Salesforce 中，用戶端 ID 稱為消費者金鑰，而用戶端秘密則稱為消費者秘密。記下您的用戶端 ID 和用戶端秘密。您會在下一節中用到它們。

步驟 2：新增 OIDC IdP 到您的使用者集區

在本節中，您設定使用者集區以處理來自 OIDC IdP 的 OIDC 型身分驗證請求。

新增 OIDC IdP (Amazon Cognito 主控台)

新增 OIDC IdP

1. 前往 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 從導覽選單中，選擇 Users Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。
4. 選擇 Sign-in experience (登入體驗) 索引標籤。找到 Federated sign-in (聯合登入)，然後選取 Add an identity provider (新增身分提供者)。
5. 選擇 OpenID Connect IdP。
6. 在 Provider name (供應商名稱) 中輸入唯一的名稱。
7. 在 Client ID (用戶端 ID) 中輸入您從供應商處收到的用戶端 ID。
8. 在 Client secret (用戶端密碼) 中輸入您從供應商處收到的用戶端密碼。
9. 輸入此供應商的 Authorized scopes (授權範圍)。範圍會定義您的應用程式從您的供應商請求的使用者屬性群組 (例如 `name` 和 `email`)。範圍必須以空格隔開，遵循 OAuth 2.0 規格。

系統會要求您的使用者同意提供這些屬性給應用程式。

10. 選擇 Attribute request method (屬性請求方法) 以提供 HTTP 方法 (GET 或 POST) 給 Amazon Cognito，Amazon Cognito 必須使用此方法從您的供應商操作的 userInfo 端點擷取使用者詳細資訊。
11. 透過 Auto fill through issuer URL (透過發行者 URL 自動填入) 或 Manual input (手動輸入)，選擇 Setup method (設定方法) 來擷取 OpenID Connect 端點。您的供應商擁有公有 `.well-known/openid-configuration` 端點 (Amazon Cognito 可在此處擷取 `authorization`、`token`、`userInfo` 和 `jwks_uri` 端點的 URL) 時，請使用 Auto fill through issuer URL (透過發行者 URL 自動填入)。
12. 輸入發行者 URL 或您 IdP 的 `authorization`、`token`、`userInfo` 和 `jwks_uri` 端點 URL。

> **Note**
>
> URL 的開頭應為 `https://`，結尾不應有正斜線 `/`。只有連接埠號碼 443 和 80 可以搭配此 URL 使用。例如，Salesforce 使用此 URL：`https://login.salesforce.com`
> 如果您選擇自動填入，則探索文件必須針對以下值使用 HTTPS：`authorization_endpoint`、`token_endpoint`、`userinfo_endpoint` 和 `jwks_uri`。否則登入會失敗。

13. OIDC 宣告 sub 預設會映射至使用者集區屬性 Username (使用者名稱)。您可以將其他 OIDC 宣告對應至使用者集區屬性。輸入 OIDC 宣告，並從下拉式清單中選擇對應的使用者集區屬性。例如，宣告 email 通常對應至使用者集區屬性 Email。
14. 將 IdP 的屬性對應至您的使用者集區。如需詳細資訊，請參閱為您的使用者集區指定身分提供者屬性對應。
15. 選擇 Create (建立)。
16. 從 App client integration (應用程式用戶端整合) 標籤，在清單中選擇一個 App clients (應用程式用戶端)，以及 Edit hosted UI settings (編輯託管 UI 設定)。將新的 OIDC IdP 新增至 Identity providers (身分提供者) 下的應用程式用戶端。
17. 選擇儲存變更。

新增 OIDC IdP (AWS CLI)

- 請參閱 CreateIdentityProviderAPI 方法的參數說明。

```
aws cognito-idp create-identity-provider
--user-pool-id string
--provider-name string
--provider-type OIDC
--provider-details map

--attribute-mapping string
--idp-identifiers (list)
--cli-input-json string
--generate-cli-skeleton string
```

使用此供應商詳細資訊地圖：

```
{
  "client_id": "string",
  "client_secret": "string",
  "authorize_scopes": "string",
  "attributes_request_method": "string",
  "oidc_issuer": "string",

  "authorize_url": "string",
  "token_url": "string",
  "attributes_url": "string",
  "jwks_uri": "string"
}
```

步驟 3：測試 OIDC IdP 組態

您可以使用前兩節的元素來建立授權 URL，並使用它們來測試您的 OIDC IdP 組態。

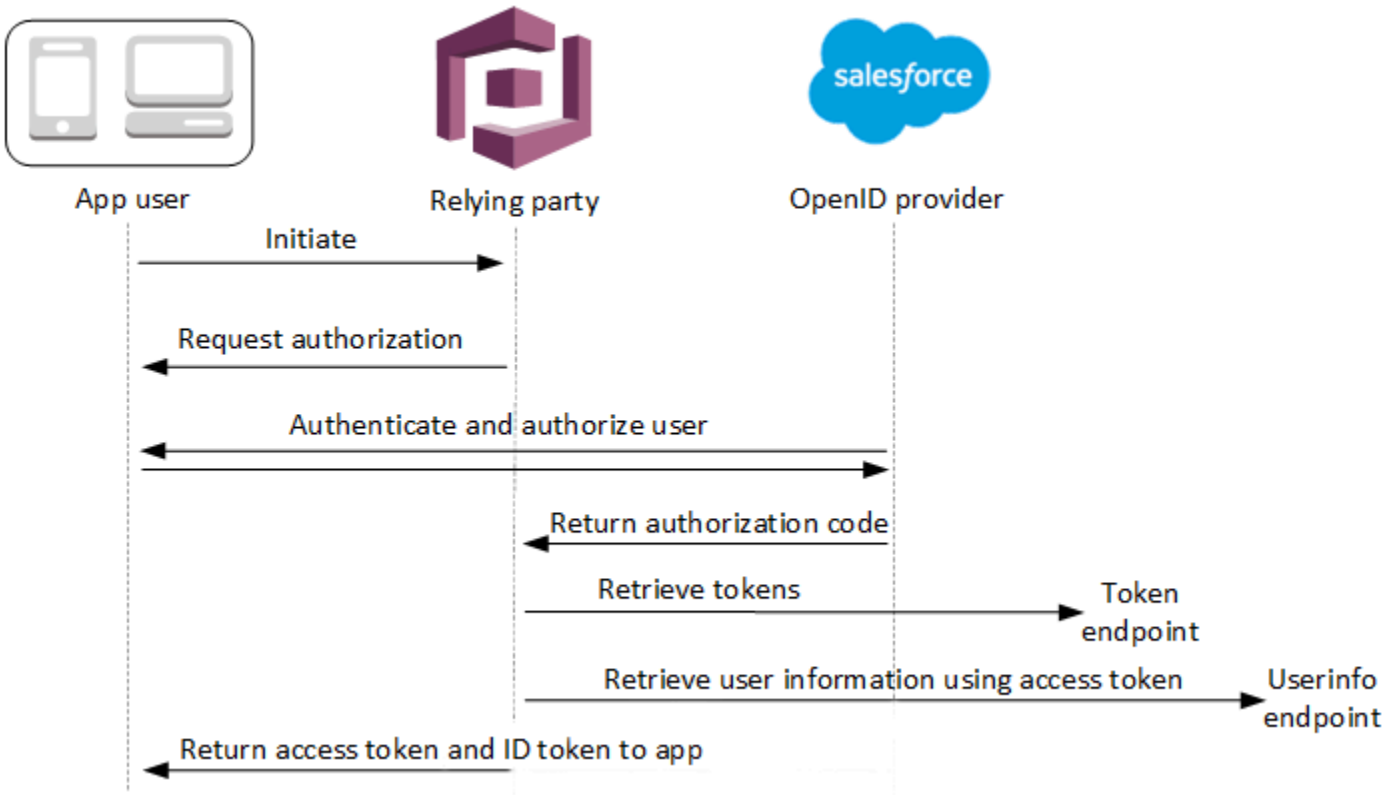
```
https://mydomain.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

您可以在使用者集區 Domain name (網域名稱) 主控台頁面上尋找您的網域。client_id 位於 General settings (一般設定) 頁面上。將您的回呼 URL 使用於 redirect_uri 參數。這是您的使用者成功身分驗證後會被重新引導的頁面 URL。

OIDC 使用者集區 IdP 身分驗證流程

當您的使用者使用 OIDC IdP 登入您的應用程式，其會通過以下身分驗證流程。

1. 您的使用者會登陸 Amazon Cognito 內建登入頁面，其中提供透過 OIDC IdP (例如 Salesforce) 登入的選項。
2. 使用者會被重新導向至 OIDC IdP 的 `authorization` 端點。
3. 您的使用者通過身分驗證後，OIDC IdP 會重新導向到 Amazon Cognito 並包含授權碼。
4. Amazon Cognito 會與 OIDC IdP 交換授權碼以取得存取權杖。
5. Amazon Cognito 建立或更新您使用者集區中的使用者帳戶。
6. Amazon Cognito 發出您的應用程式承載權杖，其中可能包括身分、存取和重新整理權杖。



> **Note**
>
> 身分驗證請求若未在 5 分鐘內完成，Amazon Cognito 會將該請求取消，並將使用者重新導向至託管的 UI。此頁面人會顯示「Something went wrong」錯誤訊息。

OIDC 是 OAuth 2.0 之上的一個身份層，它指定了由 OIDC 客戶端應用程序（依 IdPs 賴方）發行的 JSON 格式（JWT）身份令牌。請參閱 OIDC IdP 的文件以取得新增 Amazon Cognito 做為 OIDC 依賴方的相關資訊。

當使用者使用授權碼授予進行驗證時，使用者集區會傳回 ID、存取和重新整理權杖。ID 權杖是用於身分管理的標準 OIDC 權杖，而存取權杖是標準 OAuth 2.0 權杖。如需使用者集區應用程式用戶端可支援的授予類型詳細資訊，請參閱 授權端點。

使用者集區如何處理來自 OIDC 提供者的宣告

當您的使用者使用第三方 OIDC 提供者完成登入時，Amazon Cognito 託管 UI 會從 IdP 擷取授權碼。您的使用者集區會交換授權碼，以取得您 IdP 的 `token` 端點的存取和 ID 權杖。您的使用者集區不會將這些權杖傳遞至使用者或應用程式，但會用它們來建置使用者設定檔，且包含自己所屬權杖的宣告中顯示的資料。

Amazon Cognito 不會單獨驗證存取權杖。而是向提供者 `userInfo` 端點請求使用者屬性資訊，並預期權杖無效時，請求會遭拒。

Amazon Cognito 會透過下列檢查來驗證提供者 ID 權杖：

1. 檢查提供者是否使用下列集合的演算法簽署權杖：RSA、HMAC、Elliptic Curve。
2. 如果提供者使用非對稱簽署演算法簽署權杖，請檢查權仗 `kid` 宣告中的簽署金鑰 ID 是否於提供者 `jwks_uri` 端點列出。
3. 根據提供者中繼資料，比較 ID 權杖簽名與預期的簽名。
4. 將 `iss` 宣告與針對 IdP 設定的 OIDC 發行者進行比較。
5. 比較 `aud` 宣告是否與 IdP 上設定的用戶端 ID 相符，或者如果 `aud` 宣告中有多個值，則比較宣告是否包含設定的用戶端 ID。
6. 檢查 `exp` 宣告中的時間戳記是否未早於目前時間。

您的使用者集區會驗證 ID 權杖，然後嘗試向提供者 `userInfo` 端點提出提供者存取權杖的請求。它會擷取範圍落在授權其讀取的存取權仗內的任何使用者設定檔資訊。接著，您的使用者集區就可以搜尋您在使用者集區中設定的必要使用者屬性。您必須在提供者組態中為必要的屬性建立屬性映射。您的使用

者集區會檢查提供者 ID 權杖和 `userInfo` 回應。您的使用者集區會將符合映射規則的所有宣告，寫入使用者集區使用者設定檔上的使用者屬性中。您的使用者集區會忽略符合映射規則但不是必要，且在提供者的宣告中找不到的屬性。

為您的使用者集區指定身分提供者屬性映射

您可以使用 AWS Management Console或 AWS CLI 或 API 來指定使用者集區的身分識別提供者 (IdP) 的屬性對應。

對應的須知事項

開始設定使用者屬性對應之前，請先檢閱下列重要詳細資訊。

- 聯合身分使用者登入您的應用程式時，使用者集區所需的每個使用者集區屬性都必須有對應。例如，如果您的使用者集區需要 `email` 屬性進行註冊，請將此屬性對應至 IdP 中的同等屬性。
- 對應的電子郵件地址預設未經驗證。您無法使用單次代碼驗證對應的電子郵件地址。相反地，請對應 IdP 的屬性以取得驗證狀態。例如，Google 和大多數 OIDC 供應商都包含 `email_verified` 屬性。
- 您可以將身分提供者 (IdP) 權杖映射至您的使用者集區中的自訂屬性。社交提供者會顯示存取權仗，OIDC 提供者則會顯示存取和 ID 權仗。若要映射權杖，請新增長度上限為 2,048 個字元的自訂屬性，授予您的應用程式用戶端寫入屬性的權限，以及將 `access_token` 或 `id_token` 從 IdP 映射至自訂屬性。
- 對於每個映射的使用者集區屬性，值的長度上限 2,048 個字元必須足夠容納 Amazon Cognito 從 IdP 取得的值。否則，Amazon Cognito 會在使用者登入至您的應用程式時報告錯誤。權杖長度超過 2,048 個字元時，Amazon Cognito 不支援將 IdP 權杖映射至自訂屬性。
- Amazon Cognito 會從聯合身分 IdP 通過的特定宣告衍生聯合身分使用者設定檔中的`username`屬性，如下表所示。例如，Amazon Cognito 會在此屬性值前面加上您的 IdP 名稱。`MyOIDCIdP_[sub]`當您希望聯合身分使用者擁有與外部使用者目錄中的屬性完全相符的屬性時，請將該屬性對應至 Amazon Cognito 登入屬性，例如。`preferred_username`

身分提供者	**username** 資源屬性
Facebook	`id`
Google	`sub`
登入 Amazon	`user_id`

身分提供者	**username** 資源屬性
使用 Apple 登入	`sub`
SAML 供應商	`NameID`
OpenID Connect (OIDC) 供應商	`sub`

- Amazon Cognito 必須在使用者登入至您的應用程式時更新您已對應的使用者集區屬性。使用者透過 IdP 登入時，Amazon Cognito 會使用 IdP 的最新資訊，更新已對應屬性。Amazon Cognito 會更新每個對應的屬性，即使目前的值已符合最新資訊。為確保 Amazon Cognito 可更新屬性，請確認下列要求：
  - 從 IdP 映射的所有使用者集區自訂屬性都必須是可變的。您隨時都可以更新可變的自訂屬性。相比之下，當您第一次建立使用者描述檔時，您只能為使用者的不可變自訂屬性設定一個值。若要在 Amazon Cognito 主控台中建立可變的自訂屬性，請您在 Sign-up experience (註冊體驗) 索引標籤中選取 Add custom attributes (新增自訂屬性) 時，啟動您所新增屬性的 Mutable (可變) 核取方塊。或者，如果您使用 CreateUserPoolAPI 作業建立使用者集區，則可以將每個屬性的`Mutable`參數設定為`true`。如果您的 IdP 為對應的不可變屬性傳送值，Amazon Cognito 會傳回錯誤，且登入失敗。
  - 在應用程式的應用程式用戶端設定中，對應的屬性必須為可寫入。您可以在 Amazon Cognito 主控台的 App clients (應用程式用戶端) 頁面中，設定哪些屬性為可寫入。或者，若您使用 `CreateUserPoolClient` API 操作來建立應用程式用戶端，您可以將這些屬性新增至 `WriteAttributes` 陣列。如果您的 IdP 為對應的非可寫入屬性傳送值，Amazon Cognito 不會設定屬性值，而會繼續進行身份驗證。
- 當 IdP 屬性包含多個值時，Amazon Cognito 會將所有值平面化為單一逗號分隔的字串，並將 URL 格式編碼包含非英數字元的值 (不包括 "、 . "、'和 -' 字元)。* _您必須先將個別值解碼並剖析，才能在應用程式中使用。

為您的使用者集區指定身分提供者屬性對應 (AWS Management Console)

您可以使用 AWS Management Console 來指定使用者集區 IdP 的屬性對應。

> **Note**
>
> Amazon Cognito 只有在傳入字符中存在宣告時，才會將傳入宣告對應至使用者集區屬性。如果傳入權杖中不再存在先前對應的宣告，則不會刪除或變更該要求。如果您的應用程式需要對

應已刪除的要求，您可以使用預先驗證 Lambda 觸發程序以在身分驗證期間刪除自訂屬性，並允許這些屬性從傳入字符重新填入。

指定社交 IdP 屬性對應

1. 登入 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 在導覽窗格中，選擇 User Pools (使用者集區)，然後選擇您要編輯的使用者集區。
3. 選擇 Sign-in experience (登入體驗) 索引標籤並找到 Federated sign-in (聯合登入)。
4. 選擇 Add an identity provider (新增身分提供者)，或選擇您已設定的 Facebook、Google、Amazon 或者 Apple IdP。找到 Attribute mapping (屬性對應)，然後選擇 Edit (編輯)。

   如需有關新增社交 IdP 的詳細資訊，請參閱透過使用者集區使用社交身分提供者。
5. 針對您需要對應的每個屬性，完成下列步驟：

   a. 從 User pool attribute (使用者集區屬性) 欄選取屬性。這是指派給您使用者集區中使用者描述檔的屬性。自訂屬性會列在標準屬性之後。

   b. 從 ***<provider>*** attribute (<provider> 屬性) 欄選取屬性。這是從供應商目錄傳遞的屬性。來自社交供應商的已知屬性會在下拉式清單中提供。

   c. 若要在 IdP 和 Amazon Cognito 之間映射其他屬性，請選擇 Add another attribute (新增另一個屬性)。
6. 選擇 Save changes (儲存變更)。

指定 SAML 供應商屬性對應

1. 登入 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 在導覽窗格中，選擇 User Pools (使用者集區)，然後選擇您要編輯的使用者集區。
3. 選擇 Sign-in experience (登入體驗) 索引標籤並找到 Federated sign-in (聯合登入)。
4. 選擇 Add an identity provider (新增身分提供者)，或選擇您已設定的 SAML IdP。找到 Attribute mapping (屬性對應)，然後選擇 Edit (編輯)。如需有關新增 SAML IdP 的詳細資訊，請參閱搭配使用者集區使用 SAML 身分識別提供者。
5. 針對您需要對應的每個屬性，完成下列步驟：

   a. 從 User pool attribute (使用者集區屬性) 欄選取屬性。這是指派給您使用者集區中使用者描述檔的屬性。自訂屬性會列在標準屬性之後。

b. 從 SAML attribute (SAML 屬性) 欄選取屬性。這是從供應商目錄傳遞的屬性。

您的 IdP 可能會提供範例 SAML 聲明以供參考。一些 IdPs 使用簡單名稱，例如`email`，而另一些則使用類似於以下內容的 URL 格式屬性名稱：

```
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
```

c. 若要在 IdP 和 Amazon Cognito 之間對應其他屬性，請選擇 Add another attribute (新增另一個屬性)。

6. 選擇儲存變更。

指定使用者集區 (AWS CLI 和 AWS API) 的身分識別提供者屬性對應

使用下列命令來為您的使用者集區指定 IdP 屬性對應。

在建立供應商時指定屬性對應

- AWS CLI: `aws cognito-idp create-identity-provider`

  使用中繼資料檔案的範例：`aws cognito-idp create-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-type SAML --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

  其中 `details.json` 包含：

  ```
  {
      "MetadataFile": "<SAML metadata XML>"
  }
  ```

  > **Note**
  >
  > 如果 *<SAML metadata XML>* 包含任何引號 (")，則必須將其逸出 (\")。

  中繼資料 URL 的範例：

  ```
  aws cognito-idp create-identity-provider \
  --user-pool-id us-east-1_EXAMPLE \
  ```

```
--provider-name=SAML_provider_1 \
--provider-type SAML \
--provider-details MetadataURL=https://myidp.example.com/saml/metadata \
--attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/
emailaddress
```

- AWS API：CreateIdentityProvider

為現有的 IdP 指定屬性對應

- AWS CLI: `aws cognito-idp update-identity-provider`

  範例：`aws cognito-idp update-identity-provider --user-pool-id <user_pool_id> --provider-name <provider_name> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`
- AWS API：UpdateIdentityProvider

取得特定 IdP 的屬性對應相關資訊

- AWS CLI: `aws cognito-idp describe-identity-provider`

  範例：`aws cognito-idp describe-identity-provider --user-pool-id <user_pool_id> --provider-name <provider_name>`
- AWS API：DescribeIdentityProvider

將聯合身分使用者連結至現有的使用者描述檔

通常，同一個用戶具有包含多個身份提供者（IdPs）的配置文件，該配置文件已連接到您的用戶池。Amazon Cognito 可以將使用者每一次的出現與目錄中的同一個使用者描述檔連結。如此一來，一個擁有多個 IdP 使用者的人在您的應用程式可以得到一致的體驗。AdminLinkProviderForUser告知 Amazon Cognito 將您聯合身分目錄中的使用者唯一 ID 識別為使用者集區中的使用者。當您具有零個或多個與使用者描述檔相關聯的聯合身分時，您使用者集區中的使用者會算作一個每月作用中使用者 (MAU) 以用於計費。

當聯合身分使用者首次登入您的使用者集區時，Amazon Cognito 會尋找您已連結至其身分的本機設定檔。如果沒有連結的設定檔存在，您的使用者集區會建立新的設定檔。您可以建立本機設定檔，並在其首次登入之前的任何時間將其連結至聯合身分使用者，也可以在 AdminLinkProviderForUser API 要求中 (在計劃的預備任務中或在. 註冊前 Lambda 觸發程序 在使用者登入且 Amazon Cognito

偵測到連結的本機設定檔後，您的使用者集區會讀取使用者的宣告，並將其與 IdP 的對應規則進行比較。然後，您的使用者集區會使用從其登入對應的陳述式更新連結的本機設定檔。通過這種方式，您可以使用訪問聲明配置本地配置文件，並將其身份聲明保留給您 up-to-date 的提供商。Amazon Cognito 將您的聯合身分使用者與連結的設定檔配對之後，他們一律會登入該設定檔。然後，您可以將更多使用者的提供者身分連結到同一個設定檔，以在您的應用程式中為一個客戶提供一致的體驗。若要連結先前已登入的聯合身分使用者，您必須先刪除其現有的設定檔。您可以透過以下格式來識別現有設定檔：*[Provider name]*_identifier。例如 LoginWithAmazon_amzn1.account.AFAEXAMPLE。您建立並連結至第三方使用者身分識別的使用者具有建立使用者名稱的使用者名稱，以及包含其連結身分詳細資料的identities屬性。

> **Important**
>
> 由於AdminLinkProviderForUser允許具有外部同盟身分的使用者以使用者集區中的現有使用者身分登入，因此只能與應用程式擁有者信任的外部 IdPs 和提供者屬性搭配使用，這一點很重要。

例如，如果您是一個受管服務供應商 (MSP)，與多個客戶共用一個應用程式。每個客户都透過 Active Directory Federation Services (ADFS) 登入到您的應用程式。您的 IT 管理員 Carlos 在您的每個客户網域中都有一個帳户。您希望系統在 Carlos 每次登入時將他辨識為應用程式管理員，無論 IdP 為何。

您的 ADFS 在 IdPs 向 Amazon Cognito 聲明的 Carlos SAML email 聲明msp_carlos@example.com中顯示了 Carlos 的電子郵件地址。您在使用者集區中建立使用者名稱為 Carlos 的使用者。以下 AWS Command Line Interface （ AWS CLI ）命令將卡洛斯的身份從 IdPs ADFS1，ADFS2 和 ADFS3 鏈接起來。

> **Note**
>
> 您可以根據特定的屬性宣告連結使用者。這個能力對於 OIDC 和 SAML 來說是獨一無二的。 IdPs對於其他供應商類型，您必須根據固定來源屬性進行連結。如需詳細資訊，請參閱AdminLinkProviderForUser。當您將社交 IdP 連結到使用者設定檔時，必須將 ProviderAttributeName 設定為 Cognito_Subject。ProviderAttributeValue 必須是使用者在您 IdP 的唯一識別碼。

```
aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
```

```
--source-user
 ProviderName=ADFS1,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com

aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
 ProviderName=ADFS2,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com

aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
 ProviderName=ADFS3,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com
```

您使用者集區中的使用者描述檔 Carlos 現在具有下列 identities 屬性。

```
[{
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS1",
    "providerType": "SAML",
    "issuer": "http://auth.example.com",
    "primary": false,
    "dateCreated": 111111111111111
}, {
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS2",
    "providerType": "SAML",
    "issuer": "http://auth2.example.com",
    "primary": false,
    "dateCreated": 111111111111111
}, {
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS3",
    "providerType": "SAML",
    "issuer": "http://auth3.example.com",
    "primary": false,
    "dateCreated": 111111111111111
}]
```

有關連結聯合身分使用者的須知事項

- 每個使用者描述檔最多可以連結五個聯合身分使用者。

- 您可以將聯合身分使用者連結到現有的聯合身分使用者描述檔，或連結至本機使用者。
- 您無法將提供者連結至中的使用者設定檔 AWS Management Console。
- 您使用者的 ID 權杖在 `identities` 宣告中包含其所有相關聯的供應商。
- 您可以在 API 請求中為自動建立的聯合使用者設定檔設定密碼。AdminSetUserPassword該使用者的狀態會從 `EXTERNAL_PROVIDER` 變更為 `CONFIRMED`。處於此狀態的使用者可以登入為聯合身分使用者，並像連結的本機使用者一樣在 API 中啟動身份驗證流程。他們還可以在令牌身份驗證的 API 請求（如和）中修改其密碼和屬性。ChangePasswordUpdateUserAttributes做為最佳安全實務並為了讓使用者與外部 IdP 保持同步，請勿在聯合身分使用者設定檔上設定密碼。請改為使用 `AdminLinkProviderForUser` 將使用者連結至本機描述檔。
- 當使用者透過其 IdP 登入時，Amazon Cognito 會將使用者屬性填入連結的本機使用者設定檔。Amazon Cognito 會處理 OIDC IdP 的 ID 權杖中的身分宣告，並同時檢查 OAuth 2.0 和 OIDC 提供者的 `userInfo` 端點。Amazon Cognito 會將 ID 權杖中的資訊優先順序置於 `userInfo` 資訊之前。

當您發現使用者不再使用您連結至其設定檔的外部使用者帳戶時，您可以取消該使用者帳戶與您的使用者集區使用者的關聯。當您連結使用者時，您會在請求中提供使用者的屬性名稱、屬性值和提供者名稱。要刪除用戶不再需要的配置文件，請使用對等參數提出 AdminDisableProviderForUserAPI 請求。

AdminLinkProviderForUser如需 AWS SDK 中的其他命令語法和範例，請參閱。

使用 Lambda 觸發程序來自訂使用者集區工作流程

Amazon Cognito 可搭配 AWS Lambda 函數來修改使用者集區的身分驗證行為。您可以將使用者集區設定為在首次註冊之前、完成驗證後以及介於兩者之間的幾個階段，自動調用 Lambda 函數。您的函數可以修改驗證流程的預設行為、發出 API 請求以修改使用者集區或其他 AWS 資源，以及與外部系統通訊。Lambda 函數中的程式碼是您自己的程式碼。Amazon Cognito 會將事件資料傳送到您的函數，等待函數處理資料，而且在大多數情況下會預期回應事件，反映您想要對工作階段進行的任何變更。

在請求和回應事件系統中，您可以引入自己的驗證挑戰、在使用者集區和其他身分存放區之間遷移使用者、自訂訊息，以及修改 JSON Web 權杖 (JWT)。

Lambda 觸發程序可以自訂 Amazon Cognito 在您的使用者集區中啟動動作後，傳送給使用者的回應。例如，您可以阻止本來會成功的使用者登入。他們也可以針對您的 AWS 環境、外部 API、資料庫或身分存放區執行執行期操作。例如，遷移使用者觸發程序可以結合外部動作與 Amazon Cognito 中的變更：您可以查詢外部目錄中的使用者資訊，然後根據該外部資訊為新使用者設定屬性。

當您將 Lambda 觸發程序指派給使用者集區時，Amazon Cognito 會中斷其預設流程，從您的函數請求資訊。Amazon Cognito 會產生 JSON 事件並將其傳遞到您的函數。此事件包含使用者建立使用者帳戶、登入、重設密碼或更新屬性之請求的相關資訊。然後，您的函數有機會採取行動，或將事件原封不動地傳回。

下表摘錄一些使用 Lambda 觸發程序來自訂使用者集區操作的方法：

使用者集區流程	操作	說明
自訂身分驗證流程	定義身分驗證挑戰	決定自訂身分驗證流程中的下一個挑戰
	建立身分驗證挑戰	在自訂身分驗證流程中建立挑戰
	確認身分驗證挑戰回應	判斷自訂身分驗證流程中的回應是否正確
身分驗證事件	the section called “身分驗證前 Lambda 觸發程序”	自訂驗證接受或拒絕登入請求
	the section called “身分驗證後 Lambda 觸發程序”	記錄事件以進行自訂分析
	the section called “產生權杖前 Lambda 觸發程序”	增加或抑制權杖宣告
註冊	the section called “註冊前 Lambda 觸發程序”	執行用於接受或拒絕註冊請求的自訂驗證
	the section called “確認後 Lambda 觸發程序”	新增自訂歡迎訊息或事件記錄以進行自訂分析
	the section called “遷移使用者 Lambda 觸發程序”	將使用者從現有的使用者目錄遷移到使用者集區
訊息	the section called “自訂訊息 Lambda 觸發程序”	執行訊息的進階自訂及當地語系化

使用者集區流程	操作	說明
權杖建立	the section called “產生權杖前 Lambda 觸發程序”	新增或移除 ID 權杖中的屬性
電子郵件和簡訊第三方供應商	the section called “自訂寄件者 Lambda 觸發程序”	透過第三方供應商傳送簡訊和電子郵件

主題

重要考量

準備用於 Lambda 函數的使用者集區時，請考慮下列事項：

- Amazon Cognito 傳送給您的 Lambda 觸發程序的事件可能會隨著新功能而變更。JSON 階層中回應和請求元素的位置可能會變更，或者可能會新增元素名稱。在 Lambda 函數中，您可以期望收到本指南中所述的輸入元素鍵值對，但更嚴格的輸入驗證可能會導致函數失敗。

- 您可以選擇 Amazon Cognito 傳送至某些觸發程序的多個版本事件之一。某些版本可能會要求您接受 Amazon Cognito 定價的變更。如需定價的詳細資訊，請參閱 Amazon Cognito 定價。若要在產生權杖前 Lambda 觸發程序中自訂存取權杖，您必須使用進階安全性功能設定使用者集區，並更新 Lambda 觸發程序組態以使用第 2 版事件。
- 除了自訂寄件者 Lambda 觸發程序之外，Amazon Cognito 會同步叫用 Lambda 函數。當 Amazon Cognito 呼叫您的 Lambda 函數時，該函數必須在 5 秒內回應。若未回應，而可以重試呼叫，則 Amazon Cognito 會重試呼叫。嘗試失敗 3 次後，函數會發生逾時。您無法變更此 5 秒逾時值。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 Lambda 程式設計模型。

  Amazon Cognito 不會重試傳回調用錯誤且 HTTP 狀態碼為 500-599 的函數呼叫。這些代碼表示發生組態問題，導致 Lambda 無法啟動該函數。如需詳細資訊，請參閱 AWS Lambda 中錯誤處理和自動重試。
- 您無法在 Lambda 觸發程式組態中宣告函數版本。Amazon Cognito 使用者集區預設會調用最新版本的函數。但是，您可以將函數版本與別名相關聯，並在 CreateUserPool 或 UpdateUserPool API 請求中將觸發程式 `LambdaArn` 設為別名 ARN。此選項無法用於 AWS Management Console 中。若要進一步了解別名，請參閱 AWS Lambda 開發人員指南中的 Lambda 函數別名。
- 若刪除 Lambda 觸發程序，必須更新使用者集區內對應的觸發程序。例如，若您刪除驗證後觸發，便須在對應的使用者集區內，將 Post authentication (身分驗證後) 觸發設為 none (無)。
- 如果您的 Lambda 函數未將請求和回應參數傳回至 Amazon Cognito，或傳回錯誤，則驗證事件不會成功。您可以在函數中傳回錯誤，以防止使用者註冊、驗證、產生權杖，或進行驗證流程中調用 Lambda 觸發程序的任何其他階段。

  Amazon Cognito 託管 UI 會在登入提示上方，以錯誤文字的形式傳回 Lambda 觸發程序產生的錯誤。Amazon Cognito 使用者集區 API 會使用 *`[trigger]`* `failed with error` *`[error text from response]`* 格式傳回觸發程序錯誤。最佳實務是僅在您希望使用者看到的 Lambda 函數中產生錯誤。使用類似 `print()` 的輸出方法將任何敏感或偵錯資訊記錄到 CloudWatch Logs。如需範例，請參閱 註冊前範例：如果使用者名稱少於五個字元，則拒絕註冊。
- 您可以在另一個 AWS 帳戶 中新增 Lambda 函數，做為使用者集區的觸發程序。您必須使用 CreateUserPool 和 UpdateUserPool API 操作或與 AWS CloudFormation 中和 AWS CLI 中的對等項目來新增跨帳戶觸發程序。您無法在 AWS Management Console 中新增跨帳戶功能。
- 當您在 Amazon Cognito 主控台中新增 Lambda 觸發程序時，Amazon Cognito 會在您的函數中新增以資源為基礎的政策，以允許您的使用者集區叫用該函數。在 Amazon Cognito 主控台外部建立 Lambda 觸發程序時，包括跨帳戶之間的函數，您必須為 Lambda 函數新增基於資源政策的許可。您新增的許可必須允許 Amazon Cognito 以代表您的使用者集區叫用函數。您可以從 Lambda 主控台新增許可或使用 Lambda AddPermission API 操作。

以 Lambda 資源為基礎的政策範例

下列 Lambda 資源型政策範例允許 Amazon Cognito 叫用 Lambda 函數的有限功能。執行此操作時，Amazon Cognito 只能代表 `aws:SourceArn` 條件中的使用者集區和 `aws:SourceAccount` 條件中的帳戶叫用函數。

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "lambda-allow-cognito",
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-idp.amazonaws.com"
            },
            "Action": "lambda:InvokeFunction",
            "Resource": "<your Lambda function ARN>",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "AWS:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
}
```

新增使用者集區 Lambda 觸發程序

使用主控台新增使用者集區 Lambda 觸發程序

1. 使用 Lambda 主控台建立 Lambda 函數。如需 Lambda 函數的詳細資訊，請參閱《AWS Lambda 開發人員指南》。
2. 前往 Amazon Cognito 主控台，然後選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。

4. 選擇 User pool properties (使用者集區屬性) 索引標籤並找到 Lambda triggers (Lambda 觸發程序)。
5. 選擇 Add a Lambda trigger (新增 Lambda 觸發程序)。
6. 根據您要自訂的身分驗證階段，選取 Lambda 觸發程序 Category (類別)。
7. 選擇 Assign Lambda function (指定 Lambda 函數) 並在同一個 AWS 區域 選取函數作為您的使用者集區。

> **Note**
>
> 如果您的AWS Identity and Access Management (IAM) 憑證具有更新 Lambda 函數的許可，則 Amazon Cognito 會新增以 Lambda 資源為基礎的政策。使用此政策，Amazon Cognito 可以叫用您選取的函數。如果登入的憑證沒有足夠的 IAM 許可，則您必須個別更新以資源為基礎的政策。如需更多詳細資訊，請參閱 the section called “重要考量”。

8. 選擇 Save changes (儲存變更)。
9. 您可以使用 Lambda 主控台中的 CloudWatch 記錄您的 Lambda 函數。如需詳細資訊，請參閱存取 Lambda 的 CloudWatch Logs。

使用者集區 Lambda 觸發程序事件

Amazon Cognito 會將事件資訊傳遞至您的 Lambda 函數。Lambda 函數會將相同事件物件傳回 Amazon Cognito，並在回應中附上任何變更。這個事件顯示 Lambda 觸發程序的常用參數：

JSON

```
{
    "version": "string",
    "triggerSource": "string",
    "region": AWSRegion,
    "userPoolId": "string",
    "userName": "string",
    "callerContext":
        {
            "awsSdkVersion": "string",
            "clientId": "string"
        },
    "request":
        {
            "userAttributes": {
```

```
                "string": "string",
                ....
            }
        },
    "response": {}
}
```

使用者集區 Lambda 觸發程序的常用參數

version

Lambda 函數的版本編號。

triggerSource

觸發 Lambda 函數的事件名稱。如需每個 triggerSource 的說明，請參閱 [將 Lambda 觸發程序連接至使用者集區功能操作](#)。

region

AWS 區域 作為 AWSRegion 執行個體。

userPoolId

使用者集區的 ID。

使用者名稱

目前使用者的使用者名稱。

callerContext

有關請求和程式碼環境的中繼資料。它包含欄位 awsSdkVersion 和 clientId。

awsSdkVersion

產生請求的 AWS SDK 版本。

clientId

使用者集區應用程式用戶端的 ID。

請求

使用者 API 請求的詳細資訊。它包括下列欄位，以及觸發程序特有的任何請求參數。例如，Amazon Cognito 傳送至預先身分驗證觸發程序的事件也會包含一個 userNotFound 參數。您可以處理此參數的值，以便在使用者嘗試使用未註冊的使用者名稱登入時採取自訂動作。

userAttributes

一或多組使用者屬性名稱和值的鍵/值對，例如 `"email": "john@example.com"`。

response

此參數不包含原始請求中的任何資訊。您的 Lambda 函數必須將整個事件傳回給 Amazon Cognito，並將任何傳回參數新增至 `response`。若要查看函數可以包含哪些傳回參數，請參閱您要使用之觸發程序的文件。

將 API 操作連線至 Lambda 觸發程序

以下各節說明 Amazon Cognito 從使用者集區中的活動呼叫的 Lambda 觸發程序。

當您的應用程式透過 Amazon Cognito 使用者集區 API、託管 UI 或使用者集區端點登入使用者時，Amazon Cognito 會根據工作階段內容叫用您的 Lambda 函數。如需有關 Amazon Cognito 使用者集區 API 和使用者集區端點的詳細資訊，請參閱 使用 Amazon Cognito 使用者集區 API 和使用者集區端點。接下來各節中的表格描述了導致 Amazon Cognito 叫用函數的事件，以及 Amazon Cognito 在請求中包含的 `triggerSource` 字串。

主題

- Amazon Cognito API 中的 Lambda 觸發程序
- 託管 UI 中 Amazon Cognito 本機使用者的 Lambda 觸發程序
- 適用於聯合身分使用者的 Lambda 觸發程序

Amazon Cognito API 中的 Lambda 觸發程序

下表說明當您的應用程式建立、登入或更新本機使用者時，Amazon Cognito 可呼叫 Lambda 觸發程序的來源字串。

Amazon Cognito API 中的本機使用者觸發程序來源

API 操作	Lambda 觸發程序	觸發程序來源
AdminCreateUser	註冊前	`PreSignUp_AdminCreateUser`
	產生權杖前	`TokenGeneration_NewPasswordChallenge`

API 操作	Lambda 觸發程序	觸發程序來源
	自訂訊息	CustomMessage_AdminCreateUser
	自訂電子郵件寄件者	CustomEmailSender_AdminCreateUser
	自訂簡訊寄件者	CustomSMSSender_AdminCreateUser
SignUp	註冊前	PreSignUp_SignUp
	自訂訊息	CustomMessage_SignUp
	自訂電子郵件寄件者	CustomEmailSender_SignUp
	自訂簡訊寄件者	CustomSMSSender_SignUp
ConfirmSignUp AdminConfirmSignUp	確認後	PostConfirmation_ConfirmSignUp
InitiateAuth AdminInitiateAuth	身分驗證前	PreAuthentication_Authentication
	定義驗證挑戰	DefineAuthChallenge_Authentication
	建立驗證挑戰	CreateAuthChallenge_Authentication

API 操作	Lambda 觸發程序	觸發程序來源
	產生權杖前	TokenGeneration_Authentication TokenGeneration_AuthenticateDevice TokenGeneration_RefreshTokens
	遷移使用者	UserMigration_Authentication
	自訂訊息	CustomMessage_Authentication
	自訂電子郵件寄件者	CustomEmailSender_AccountTakeOverNotification
	自訂簡訊寄件者	CustomSMSSender_Authentication
ForgotPassword	遷移使用者	UserMigration_ForgotPassword
	自訂訊息	CustomMessage_ForgotPassword
	自訂電子郵件寄件者	CustomEmailSender_ForgotPassword
	自訂簡訊寄件者	CustomSMSSender_ForgotPassword
ConfirmForgotPassword	確認後	PostConfirmation_ConfirmForgotPassword

API 操作	Lambda 觸發程序	觸發程序來源
UpdateUserAttributes AdminUpdateUserAttributes	自訂訊息	CustomMessage_UpdateUserAttribute
	自訂電子郵件寄件者	CustomEmailSender_UpdateUserAttribute
	自訂簡訊寄件者	CustomSMSSender_UpdateUserAttribute
VerifyUserAttributes	自訂訊息	CustomMessage_VerifyUserAttribute
	自訂電子郵件寄件者	CustomEmailSender_VerifyUserAttribute
	自訂簡訊寄件者	CustomSMSSender_VerifyUserAttribute

託管 UI 中 Amazon Cognito 本機使用者的 Lambda 觸發程序

下表說明當本機使用者透過託管 UI 登入您的使用者集區時，Amazon Cognito 可呼叫 Lambda 觸發程序的來源字串。

託管 UI 中的苯機使用者觸發程序來源

託管 UI URI	Lambda 觸發程序	觸發程序來源
/signup	註冊前	PreSignUp_SignUp
	自訂訊息	CustomMessage_SignUp
	自訂電子郵件寄件者	CustomEmailSender_SignUp
	自訂簡訊寄件者	CustomSMSSender_SignUp

託管 UI URI	Lambda 觸發程序	觸發程序來源
/confirmuser	確認後	PostConfirmation_ConfirmSignUp
/login	身分驗證前	PreAuthentication_Authentication
	定義驗證挑戰	DefineAuthChallenge_Authentication
	建立驗證挑戰	CreateAuthChallenge_Authentication
	產生權杖前	TokenGeneration_Authentication TokenGeneration_AuthenticateDevice TokenGeneration_RefreshTokens
	遷移使用者	UserMigration_Authentication
	自訂訊息	CustomMessage_Authentication
	自訂電子郵件寄件者	CustomEmailSender_AccountTakeOverNotification
	自訂簡訊寄件者	CustomSMSSender_Authentication
/forgotpassword	遷移使用者	UserMigration_ForgotPassword

託管 UI URI	Lambda 觸發程序	觸發程序來源
	自訂訊息	CustomMessage_ForgotPassword
	自訂電子郵件寄件者	CustomEmailSender_ForgotPassword
	自訂簡訊寄件者	CustomSMSSender_ForgotPassword
/confirmforgotpassword	確認後	PostConfirmation_ConfirmForgotPassword

適用於聯合身分使用者的 Lambda 觸發程序

您可以使用以下 Lambda 觸發程序，為使用聯合身分提供者登入的使用者自訂使用者集區工作流程。

> **Note**
>
> 聯合身分使用者可以使用 Amazon Cognito 託管 UI 登入，或者您也可以向 授權端點 產生請求，以無訊息方式將使用者重新導向至其身分提供者登入頁面。您不能使用 Amazon Cognito 使用者集區 API 登入聯合身分使用者。

聯合身分使用者觸發程序來源

登入事件	Lambda 觸發程序	觸發程序來源
首次登入	註冊前	PreSignUp_ExternalProvider
	確認後	PostConfirmation_ConfirmSignUp
	產生權杖前	TokenGeneration_HostedAuth

登入事件	Lambda 觸發程序	觸發程序來源
後續登入	身分驗證前	`PreAuthentication_Authentication`
	身分驗證後	`PostAuthentication_Authentication`
	產生權杖前	`TokenGeneration_HostedAuth`

聯合身分登入不會叫用使用者集區中的任何 自訂身分驗證挑戰 Lambda 觸發程序、遷移使用者 Lambda 觸發程序、自訂訊息 Lambda 觸發程序 或 自訂寄件者 Lambda 觸發程序。

將 Lambda 觸發程序連接至使用者集區功能操作

每個 Lambda 觸發程序都在您的使用者集區中扮演一個功能角色。例如，觸發程序可以修改您的註冊流程，或新增自訂身分驗證挑戰。Amazon Cognito 傳送至 Lambda 函數的事件可反映組成該功能角色的多個動作之一。例如，Amazon Cognito 會在您的使用者註冊以及您建立使用者時，叫用預先註冊觸發程序。相同功能角色在這些不同情況中都有自己的 `triggerSource` 值。Lambda 函數可以根據呼叫它的操作，以不同的方式處理傳入事件。

當事件對應於觸發程序來源時，Amazon Cognito 也會叫用所有指派的函數。例如，當使用者登入您指派了遷移使用者和預先身分驗證觸發程序的使用者集區時，他們會同時啟動兩者。

註冊、確認和登入 (身分驗證) 觸發程序

觸發條件	triggerSource 值	事件
註冊前	`PreSignUp_SignUp`	註冊前。
註冊前	`PreSignUp_AdminCreateUser`	註冊前 (當管理員建立新使用者時)。
註冊前	`PreSignUp_ExternalProvider`	外部身分提供者註冊前。
確認後	`PostConfirmation_ConfirmSignUp`	註冊後確認。

觸發條件	triggerSource 值	事件
確認後	`PostConfirmation_ConfirmForgotPassword`	忘記密碼後確認。
身分驗證前	`PreAuthentication_Authentication`	身分驗證前。
身分驗證後	`PostAuthentication_Authentication`	身分驗證後。

自訂身分驗證挑戰觸發程序

觸發條件	triggerSource 值	事件
定義驗證挑戰	`DefineAuthChallenge_Authentication`	定義驗證挑戰。
建立驗證挑戰	`CreateAuthChallenge_Authentication`	建立驗證挑戰。
確認驗證挑戰	`VerifyAuthChallengeResponse_Authentication`	確認驗證挑戰回應。

產生權杖前觸發程序

觸發條件	triggerSource 值	事件
產生權杖前	`TokenGeneration_HostedAuth`	Amazon Cognito 從您的託管 UI 登入頁面對使用者進行身分驗證。
產生權杖前	`TokenGeneration_Authentication`	使用者身分驗證流程完成。

觸發條件	triggerSource 值	事件
產生權杖前	TokenGeneration_NewPasswordChallenge	管理員建立使用者。當使用者必須變更臨時密碼時，Amazon Cognito 會叫用此功能。
產生權杖前	TokenGeneration_AuthenticateDevice	使用者裝置的身分驗證結束時。
產生權杖前	TokenGeneration_RefreshTokens	使用者嘗試重新整理身分和存取權杖時。

遷移使用者觸發程序

觸發條件	triggerSource 值	事件
使用者遷移	UserMigration_Authentication	在登入時的使用者遷移。
使用者遷移	UserMigration_ForgotPassword	忘記密碼流程時的使用者遷移。

自訂訊息觸發程序

觸發條件	triggerSource 值	事件
自訂訊息	CustomMessage_SignUp	當使用者在使用者集區註冊時的自訂訊息。
自訂訊息	CustomMessage_AdminCreateUser	當您以管理員身分建立使用者，而且 Amazon Cognito 向這些使用者發送臨時密碼時的自訂訊息。
自訂訊息	CustomMessage_ResendCode	當現有使用者請求新確認代碼時的自訂訊息。

觸發條件	triggerSource 值	事件
自訂訊息	CustomMessage_ForgotPassword	當使用者請求密碼重設時的自訂訊息。
自訂訊息	CustomMessage_UpdateUserAttribute	當使用者變更其電子郵件地址或電話號碼，而且 Amazon Cognito 傳送驗證碼時的自訂訊息。
自訂訊息	CustomMessage_VerifyUserAttribute	當使用者新增電子郵件地址或電話號碼，而且 Amazon Cognito 傳送驗證碼的自訂訊息。
自訂訊息	CustomMessage_Authentication	當已設定簡訊 MFA 的使用者登入時的自訂訊息。

註冊前 Lambda 觸發程序

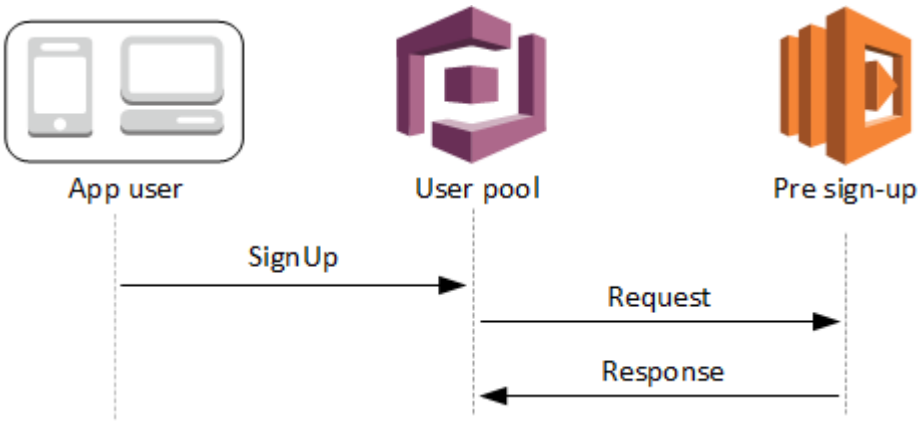
系統會先啟用註冊前 AWS Lambda 函數，之後 Amazon Cognito 隨即註冊新使用者。註冊過程中，您可以使用此函數執行自訂驗證，並根據驗證結果接受或拒絕註冊請求。

主題

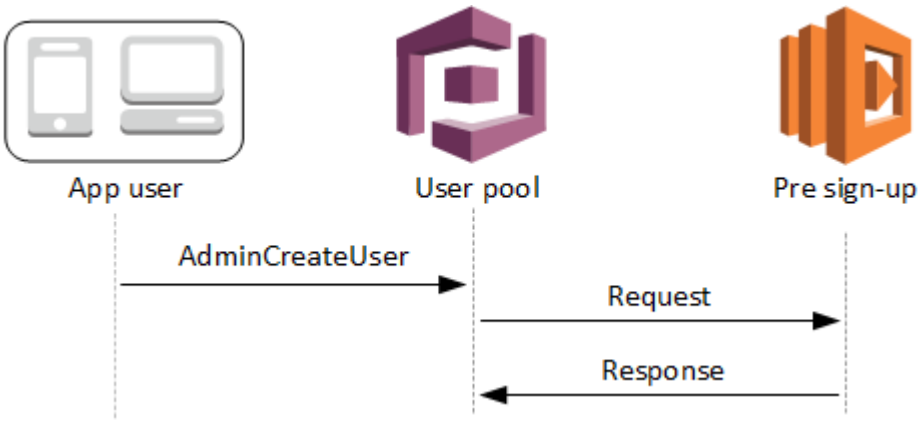
- 註冊前 Lambda 流程
- 註冊前 Lambda 觸發程序參數
- 註冊教學課程
- 註冊前範例：自動確認來自已註冊網域的使用者
- 註冊前範例：自動確認及自動驗證所有使用者
- 註冊前範例：如果使用者名稱少於五個字元，則拒絕註冊

註冊前 Lambda 流程

用戶端註冊流程



伺服器端註冊流程



請求中包含來自用戶端的驗證資料。此數據來自傳遞給用戶池 SignUp 和 AdminCreateUser API 方法的`ValidationData`值。

註冊前 Lambda 觸發程序參數

Amazon Cognito 傳遞至此 Lambda 函數的請求，是以下參數和 Amazon Cognito 新增至所有請求的[常用參數]之組合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
```

```
        },
        "validationData": {
            "string": "string",
            . . .
         },
        "clientMetadata": {
            "string": "string",
            . . .
         }
    },

    "response": {
        "autoConfirmUser": "boolean",
        "autoVerifyPhone": "boolean",
        "autoVerifyEmail": "boolean"
    }
}
```

註冊前請求參數

userAttributes

代表使用者屬性的一或多組名稱/值。屬性名稱是索引鍵。

validationData

您的應用程式在建立新使用者的請求中傳遞給 Amazon Cognito 的一或多個鍵值對與使用者屬性資料。將此資訊傳送至您AdminCreateUser或 SignUpAPI 請求 ValidationData 參數中的 Lambda 函數。

Amazon Cognito 不會將您的 ValidationData 資料設定為您建立之使用者的屬性。 ValidationData 是您為了預先註冊 Lambda 觸發器而提供的臨時使用者資訊。

clientMetadata

您可以做為 Lambda 函數的自訂輸入提供的一個或多個鍵值組，該函數是您用於註冊前觸發程序所指定。您可以使用下列 API 動作中的 ClientMetadata參數，將此資料傳遞至 Lambda 函數：AdminCreateUserAdminRespondToAuthChallengeForgotPassword、、和SignUp。

註冊前回應參數

如果您想要自動確認使用者，可以在回應中將 autoConfirmUser設定為 true。您可以將 autoVerifyEmail設定為 true，以自動驗證使用者的電子郵件。您可以將 autoVerifyPhone設定為 true，以自動驗證使用者的電話號碼。

> **Note**
>
> AdminCreateUser API 觸發尚未註冊的 Lambda 函數時，Amazon Cognito 會忽略回應參數 autoVerifyPhone、autoVerifyEmail 和 autoConfirmUser。

autoConfirmUser

設定為 true以自動確認使用者，否則設定為 false。

autoVerifyEmail

設定為 true 以設定為已驗證註冊使用者的電子郵件地址，否則設定為 false。如果 autoVerifyEmail設定為 true，email 屬性必須包含有效的非 null 值。否則會發生錯誤，而且使用者將無法完成註冊。

如果選取 email 屬性做為別名，則在設定 autoVerifyEmail 時，將會為使用者的電子郵件地址建立別名。如果使用該電子郵件地址的別名已存在，則會將別名移至新使用者，而舊使用者的電子郵件地址會標記為未驗證。如需詳細資訊，請參閱 自訂登入屬性。

autoVerifyPhone

設定為 true以設定為已驗證註冊使用者的電話號碼，否則設定為 false。如果 autoVerifyPhone設定為 true，phone_number 屬性必須包含有效的非 null 值。否則會發生錯誤，而且使用者將無法完成註冊。

如果選取 phone_number屬性做為別名，則在設定 autoVerifyPhone 時，將會為使用者的電話號碼建立別名。如果使用該電話號碼的別名已存在，則會將別名移至新使用者，而舊使用者的電話號碼會標記為未驗證。如需詳細資訊，請參閱 自訂登入屬性。

註冊教學課程

註冊前 Lambda 函數會在使用者註冊之前觸發。請參閱這些適 JavaScript用於安卓系統和 iOS 的亞馬遜認可註冊教程。

平台	教學課程
JavaScript 身分識別 SDK	使用者註冊 JavaScript
Android 身分 SDK	使用 Android 註冊使用者
iOS 身分 SDK	使用 iOS 註冊使用者

註冊前範例：自動確認來自已註冊網域的使用者

您可以使用這個註冊前 Lambda 觸發程序，新增自訂邏輯來驗證註冊您使用者集區的新使用者。這是一個示例 JavaScript 程序，顯示如何註冊新用戶。在身分驗證期間，它會叫用註冊前 Lambda 觸發程序。

JavaScript

```
var attributeList = [];
var dataEmail = {
  Name: "email",
  Value: "...", // your email here
};
var dataPhoneNumber = {
  Name: "phone_number",
  Value: "...", // your phone number here with +country code and no delimiters in
 front
};

var dataEmailDomain = {
  Name: "custom:domain",
  Value: "example.com",
};
var attributeEmail = new AmazonCognitoIdentity.CognitoUserAttribute(dataEmail);
var attributePhoneNumber = new AmazonCognitoIdentity.CognitoUserAttribute(
  dataPhoneNumber
);
var attributeEmailDomain = new AmazonCognitoIdentity.CognitoUserAttribute(
  dataEmailDomain
);

attributeList.push(attributeEmail);
attributeList.push(attributePhoneNumber);
```

```
attributeList.push(attributeEmailDomain);

var cognitoUser;
userPool.signUp(
  "username",
  "password",
  attributeList,
  null,
  function (err, result) {
    if (err) {
      alert(err);
      return;
    }
    cognitoUser = result.user;
    console.log("user name is " + cognitoUser.getUsername());
  }
);
```

這是在註冊之前，使用使用者集區註冊前 Lambda 觸發程序呼叫的範例 Lambda 觸發程序。它會使用自訂屬性 custom:domain，自動確認來自特定電子郵件網域的新使用者。不在自訂網域中的任何新使用者都會新增到這個使用者集區，但不會自動確認。

Node.js

```
exports.handler = (event, context, callback) => {
  // Set the user pool autoConfirmUser flag after validating the email domain
  event.response.autoConfirmUser = false;

  // Split the email address so we can compare domains
  var address = event.request.userAttributes.email.split("@");

  // This example uses a custom attribute "custom:domain"
  if (event.request.userAttributes.hasOwnProperty("custom:domain")) {
    if (event.request.userAttributes["custom:domain"] === address[1]) {
      event.response.autoConfirmUser = true;
    }
  }

  // Return to Amazon Cognito
  callback(null, event);
};
```

Python

```
def lambda_handler(event, context):
    # It sets the user pool autoConfirmUser flag after validating the email domain
    event['response']['autoConfirmUser'] = False

    # Split the email address so we can compare domains
    address = event['request']['userAttributes']['email'].split('@')

    # This example uses a custom attribute 'custom:domain'
    if 'custom:domain' in event['request']['userAttributes']:
        if event['request']['userAttributes']['custom:domain'] == address[1]:
            event['response']['autoConfirmUser'] = True

    # Return to Amazon Cognito
    return event
```

Amazon Cognito 會將事件資訊傳遞至您的 Lambda 函數。此函數會將相同事件物件傳回 Amazon Cognito，並在回應中附上任何變更。在 Lambda 主控台中，您可使用與 Lambda 觸發程序相關聯的資料來設定測試事件。下列是此程式碼範例的測試事件：

JSON

```
{
    "request": {
        "userAttributes": {
            "email": "testuser@example.com",
            "custom:domain": "example.com"
        }
    },
    "response": {}
}
```

註冊前範例：自動確認及自動驗證所有使用者

此範例會確認所有使用者並設定使用者的 email 和 phone_number 屬性以驗證屬性是否存在。此外，如果啟用別名，則當設定自動驗證時，將會為 phone_number和 email 建立別名。

> **Note**
>
> 如果使用相同電話號碼的別名已存在，則會將別名移至新使用者，而舊使用者的phone_number會標記為未驗證。電子郵件地址也是一樣。為了避免這種情況發生，您可以使用用戶集區 ListUsers API 來查看是否有現有用戶已使用新用戶的電話號碼或電子郵件地址作為別名。

Node.js

```
const handler = async (event) => {
  // Confirm the user
  event.response.autoConfirmUser = true;
  // Set the email as verified if it is in the request
  if (event.request.userAttributes.hasOwnProperty("email")) {
    event.response.autoVerifyEmail = true;
  }

  // Set the phone number as verified if it is in the request
  if (event.request.userAttributes.hasOwnProperty("phone_number")) {
    event.response.autoVerifyPhone = true;
  }

  return event;
};

export { handler };
```

Python

```
def lambda_handler(event, context):
    # Confirm the user
    event['response']['autoConfirmUser'] = True

    # Set the email as verified if it is in the request
    if 'email' in event['request']['userAttributes']:
        event['response']['autoVerifyEmail'] = True

    # Set the phone number as verified if it is in the request
    if 'phone_number' in event['request']['userAttributes']:
        event['response']['autoVerifyPhone'] = True
```

```
    # Return to Amazon Cognito
    return event
```

Amazon Cognito 會將事件資訊傳遞至您的 Lambda 函數。此函數會將相同事件物件傳回 Amazon Cognito，並在回應中附上任何變更。在 Lambda 主控台中，您可使用與 Lambda 觸發程序相關聯的資料來設定測試事件。下列是此程式碼範例的測試事件：

JSON

```
{
  "request": {
    "userAttributes": {
      "email": "user@example.com",
      "phone_number": "+12065550100"
    }
  },
  "response": {}
}
```

註冊前範例：如果使用者名稱少於五個字元，則拒絕註冊

此範例會檢查註冊請求中的使用者名稱長度。如果使用者輸入的名稱長度少於五個字元，則範例會傳回錯誤。

Node.js

```
exports.handler = (event, context, callback) => {
    // Impose a condition that the minimum length of the username is 5 is imposed on
 all user pools.
    if (event.userName.length < 5) {
        var error = new Error("Cannot register users with username less than the
 minimum length of 5");
        // Return error to Amazon Cognito
        callback(error, event);
    }
    // Return to Amazon Cognito
    callback(null, event);
};
```

Python

```
def lambda_handler(event, context):
    if len(event['userName']) < 5:
        raise Exception("Cannot register users with username less than the minimum
 length of 5")
    # Return to Amazon Cognito
    return event
```

Amazon Cognito 會將事件資訊傳遞至您的 Lambda 函數。此函數會將相同事件物件傳回 Amazon Cognito，並在回應中附上任何變更。在 Lambda 主控台中，您可使用與 Lambda 觸發程序相關聯的資料來設定測試事件。下列是此程式碼範例的測試事件：

JSON

```
{
  "userName": "rroe",
  "response": {}
}
```

確認後 Lambda 觸發程序

Amazon Cognito 會在已註冊的使用者確認其使用者帳戶後，調用此觸發程序。在您的確認後 Lambda 函數中，您可以傳送自訂訊息或新增自訂 API 請求。例如，您可以查詢外部系統並對使用者填入其他屬性。Amazon Cognito 只會針對在您的使用者集區中註冊的使用者調用此觸發程序，而不會針對您使用管理員憑證建立的使用者帳戶調用此觸發程序。

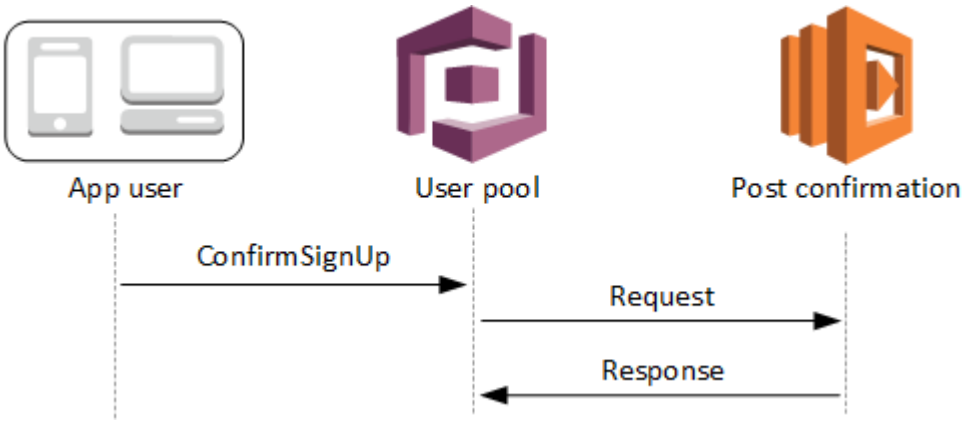
請求中包含已確認使用者的目前屬性。

主題

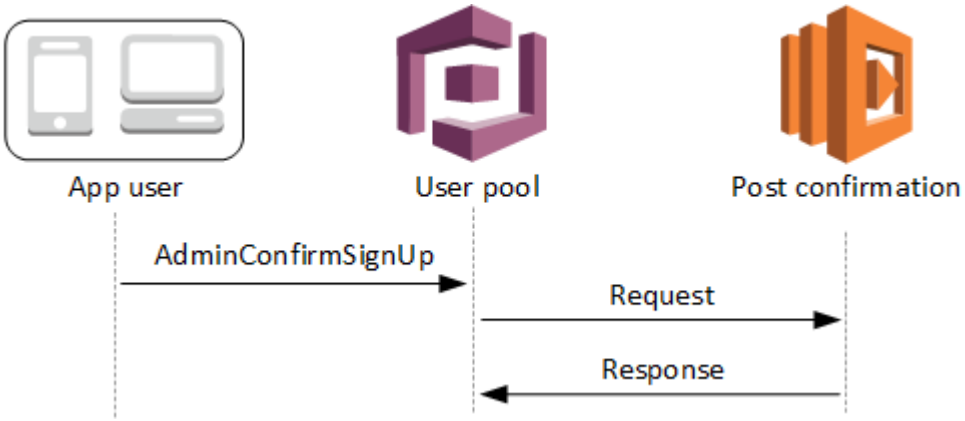
- 確認後 Lambda 流程
- 確認後 Lambda 觸發程序參數
- 使用者確認教學課程
- 確認後範例

確認後 Lambda 流程

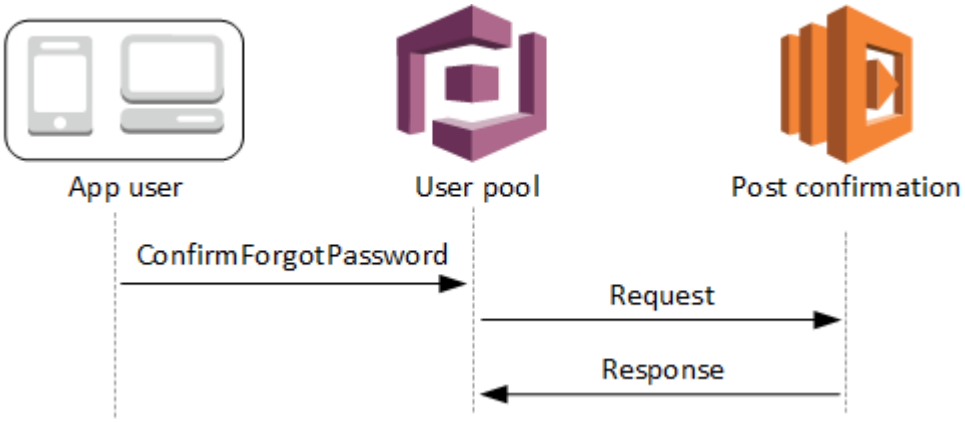
用戶端確認註冊流程



伺服器端確認註冊流程



確認忘記密碼流程



確認後 Lambda 觸發程序參數

Amazon Cognito 傳遞至此 Lambda 函數的請求，是以下參數和 Amazon Cognito 新增至所有請求的常用參數之組合。

JSON

```
{
    "request": {
            "userAttributes": {
                "string": "string",
                . . .
            },
            "clientMetadata": {
             "string": "string",
             . . .
            }
        },
    "response": {}
}
```

確認後請求參數

userAttributes

代表使用者屬性的一或多個鍵值組。

clientMetadata

您可以做為 Lambda 函數的自訂輸入提供的一個或多個鍵值組，該函數是您用於確認後觸發程序所指定。您可以使用下列 API 動作中的 ClientMetadata 參數，將此資料傳遞至您的 Lambda 函數：AdminConfirmSignUp、ConfirmForgotPassword、ConfirmSignUp 和 SignUp。

確認後回應參數

回應中不應有額外的傳回資訊。

使用者確認教學課程

確認後 Lambda 函數會在 Amazon Cognito 確認新使用者之後觸發。請參閱 JavaScript、Android 和 iOS 適用的使用者確認教學課程。

平台	教學課程
JavaScript 身分 SDK	使用 JavaScript 確認使用者
Android 身分 SDK	使用 Android 確認使用者
iOS 身分 SDK	使用 iOS 確認使用者

確認後範例

這個範例 Lambda 函數會使用 Amazon SES，將確認電子郵件訊息傳送給您的使用者。如需詳細資訊，請參閱《Amazon Simple Storage Service 開發人員指南》。

Node.js

```
// Import required AWS SDK clients and commands for Node.js. Note that this requires
// the `@aws-sdk/client-ses` module to be either bundled with this code or included
// as a Lambda layer.
import { SES, SendEmailCommand } from "@aws-sdk/client-ses";
const ses = new SES();

const handler = async (event) => {
  if (event.request.userAttributes.email) {
    await sendTheEmail(
      event.request.userAttributes.email,
      `Congratulations ${event.userName}, you have been confirmed.`
    );
  }
  return event;
};

const sendTheEmail = async (to, body) => {
  const eParams = {
    Destination: {
      ToAddresses: [to],
    },
    Message: {
      Body: {
        Text: {
          Data: body,
        },
```

```
      },
      Subject: {
        Data: "Cognito Identity Provider registration completed",
      },
    },
    // Replace source_email with your SES validated email address
    Source: "<source_email>",
  };
  try {
    await ses.send(new SendEmailCommand(eParams));
  } catch (err) {
    console.log(err);
  }
};

export { handler };
```

Amazon Cognito 會將事件資訊傳遞至您的 Lambda 函數。此函數會將相同事件物件傳回 Amazon Cognito，並在回應中附上任何變更。在 Lambda 主控台中，您可使用與 Lambda 觸發程序相關聯的資料來設定測試事件。下列是此程式碼範例的測試事件：

JSON

```
{
    "request": {
        "userAttributes": {
            "email": "user@example.com",
            "email_verified": true
        }
    },
    "response": {}
}
```

身分驗證前 Lambda 觸發程序

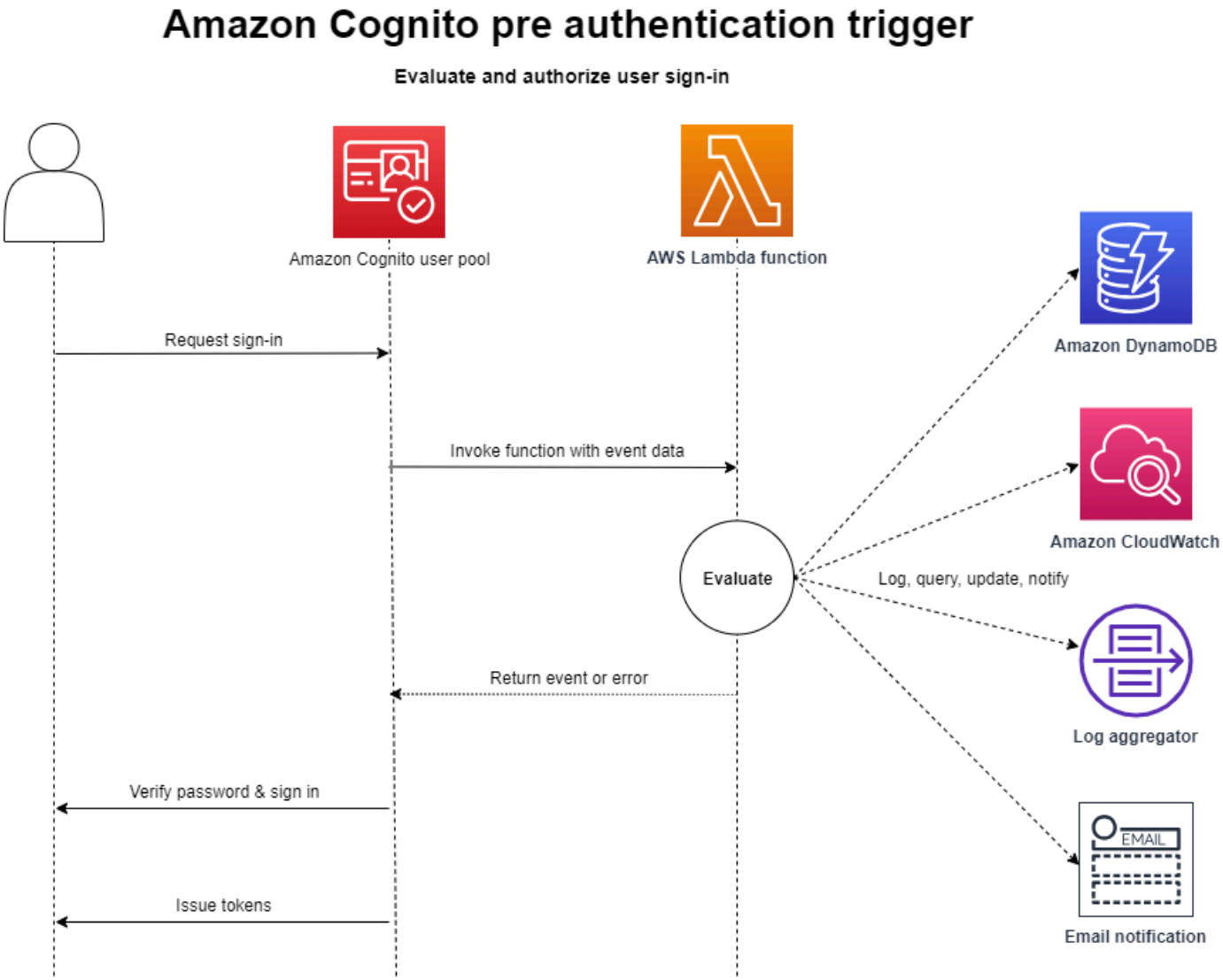
Amazon Cognito 會在使用者嘗試登入時叫用此觸發程序，因此您可以建立自訂驗證來執行預先準備的動作。例如，您可以拒絕身分驗證請求，或將工作階段資料記錄到外部系統。

> **Note**
>
> 當使用者不存在，或者您的使用者集區中已有工作階段時，不會啟用 Lambda 觸發程式。如果使用者集區應用程式用戶端的 `PreventUserExistenceErrors` 設定設為 `ENABLED`，則 Lambda 觸發程式將會啟用。

主題

- 身分驗證流程概觀
- 身分驗證前 Lambda 觸發程序參數
- 身分驗證前範例

身分驗證流程概觀



此請求包括來自應用程式傳遞至使用者集區 `InitiateAuth` 和 `AdminInitiateAuth` API 操作的 `ClientMetadata` 值的用戶端驗證資料。

如需更多詳細資訊，請參閱 使用者集區身分驗證流程。

身分驗證前 Lambda 觸發程序參數

Amazon Cognito 傳遞至此 Lambda 函數的請求，是以下參數和 Amazon Cognito 新增至所有請求的常用參數之組合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "validationData": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {}
}
```

身分驗證前請求參數

userAttributes

代表使用者屬性的一或多組名稱/值。

userNotFound

當您將使用者集區的 `PreventUserExistenceErrors` 設定為 `ENABLED` 時，Amazon Cognito 會填入此布林值。

validationData

一或多個鍵值組，包含使用者登入請求中的驗證資料。若要將此資料傳遞至您的 Lambda 函數，您可以使用 InitiateAuth 和 AdminInitiateAuth API 動作中的 ClientMetadata 參數。

身分驗證前回應參數

Amazon Cognito 不預期會在回應中收到任何其他傳回的資訊。您的函數可傳回錯誤以拒絕登入嘗試，或者使用 API 操作以查詢和修改您的資源。

身分驗證前範例

此範例函數可防止使用者使用特定的應用程式用戶端登入您的使用者集區。由於預先驗證 Lambda 函數在使用者具有現有工作階段時不會叫用，因此該函數只會防止您要封鎖的應用程式用戶端 ID 展開新的工作階段。

Node.js

```
const handler = async (event) => {
  if (
    event.callerContext.clientId === "user-pool-app-client-id-to-be-blocked"
  ) {
    throw new Error("Cannot authenticate users from this user pool app client");
  }

  return event;
};

export { handler };
```

Python

```
def lambda_handler(event, context):
    if event['callerContext']['clientId'] == "<user pool app client id to be
 blocked>":
        raise Exception("Cannot authenticate users from this user pool app client")

    # Return to Amazon Cognito
    return event
```

Amazon Cognito 會將事件資訊傳遞至您的 Lambda 函數。此函數會將相同事件物件傳回 Amazon Cognito，並在回應中附上任何變更。在 Lambda 主控台中，您可使用與 Lambda 觸發程序相關聯的資料來設定測試事件。下列是此程式碼範例的測試事件：

JSON

```
{
    "callerContext": {
        "clientId": "<user pool app client id to be blocked>"
    },
    "response": {}
```

```
}
```

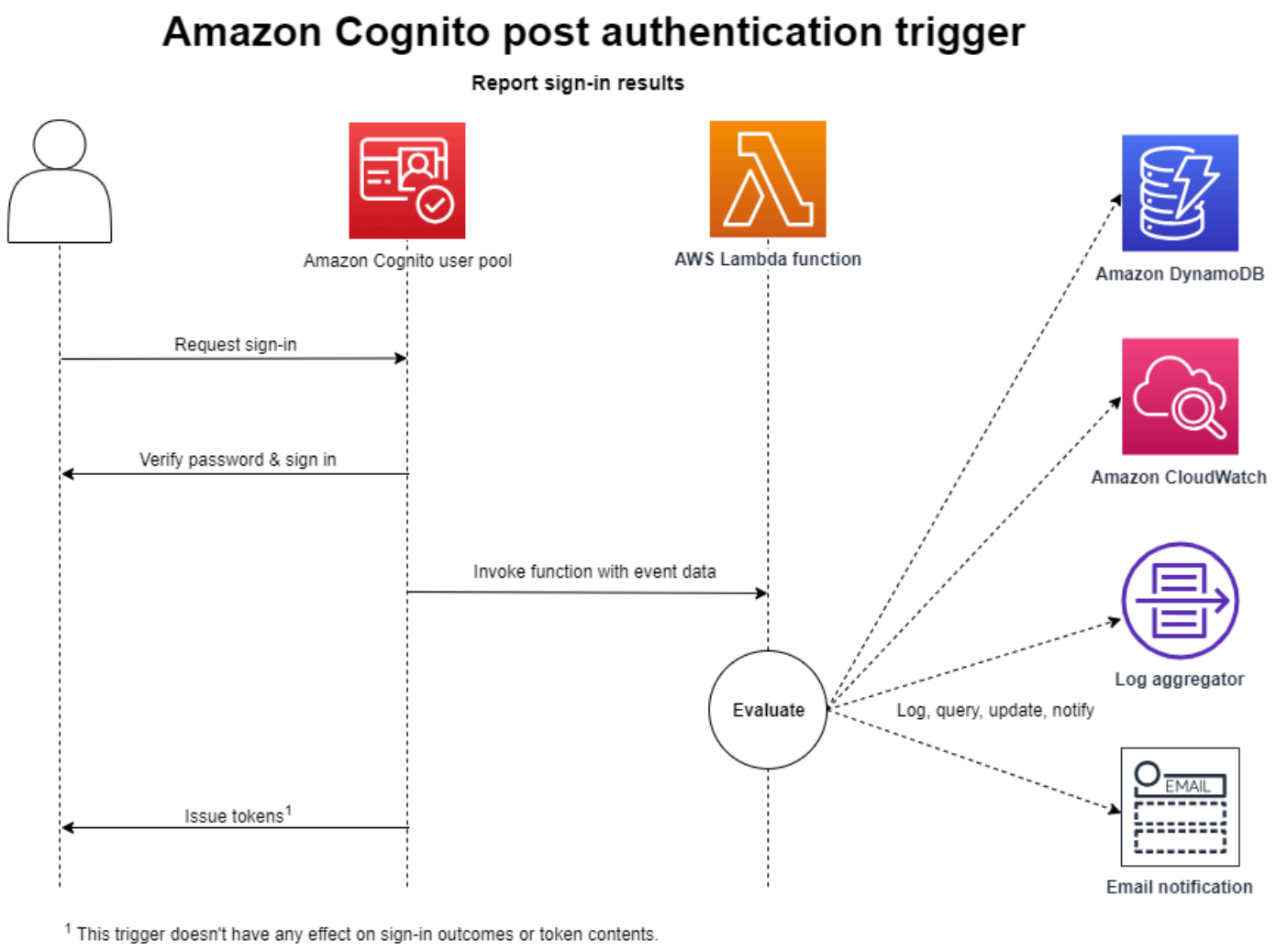
身分驗證後 Lambda 觸發程序

由於 Amazon Cognito 會在登入使用者之後叫用此觸發程序，因此您可以在 Amazon Cognito 驗證使用者後新增自訂邏輯。

主題

- 身分驗證流程概觀
- 身分驗證後 Lambda 觸發程序參數
- 身分驗證教學課程
- 身分驗證後範例

身分驗證流程概觀



如需更多詳細資訊，請參閱 使用者集區身分驗證流程。

身分驗證後 Lambda 觸發程序參數

Amazon Cognito 傳遞至此 Lambda 函數的請求，是以下參數和 Amazon Cognito 新增至所有請求的常用參數之組合。

JSON

```
{
    "request": {
        "userAttributes": {
```

```
            "string": "string",
            . . .
        },
        "newDeviceUsed": boolean,
        "clientMetadata": {
            "string": "string",
            . . .
           }
       },
    "response": {}
}
```

身分驗證後請求參數

newDeviceUsed

此旗標指出使用者是否已在新的裝置登入。唯有當使用者集區的記住裝置值為 `Always` 或 `User Opt-In` 時，Amazon Cognito 才會設定此旗標。

userAttributes

代表使用者屬性的一或多組名稱/值。

clientMetadata

您可以做為 Lambda 函數的自訂輸入提供的一個或多個鍵值組，該函數是您用於身分驗證後觸發程序所指定。若要將此資料傳遞至您的 Lambda 函數，您可以使用 AdminRespondToAuthChallenge 和 RespondToAuthChallenge API 動作中的 ClientMetadata 參數。Amazon Cognito 不包含其傳遞至身分驗證後函數的請求中的 AdminInitiateAuth 和 InitiateAuth API 操作的 ClientMetadata 參數中的資料。

身分驗證後回應參數

Amazon Cognito 不預期會在回應中收到任何其他傳回的資訊。您的函數可使用 API 操作來查詢和修改您的資源，或將事件中繼資料記錄到外部系統。

身分驗證教學課程

Amazon Cognito 登入使用者之後，會立即啟用身分驗證後 Lambda 函數。請參閱 JavaScript、Android 和 iOS 適用的登入教學課程。

平台	教學課程
JavaScript 身分 SDK	使用 JavaScript 登入使用者
Android 身分 SDK	使用 Android 登入使用者
iOS 身分 SDK	使用 iOS 登入使用者

身分驗證後範例

這個身分驗證後範本 Lambda 函數，會將成功登入的資料傳送到 CloudWatch Logs。

Node.js

```
const handler = async (event) => {
  // Send post authentication data to Amazon CloudWatch logs
  console.log("Authentication successful");
  console.log("Trigger function =", event.triggerSource);
  console.log("User pool = ", event.userPoolId);
  console.log("App client ID = ", event.callerContext.clientId);
  console.log("User ID = ", event.userName);

  return event;
};

export { handler }
```

Python

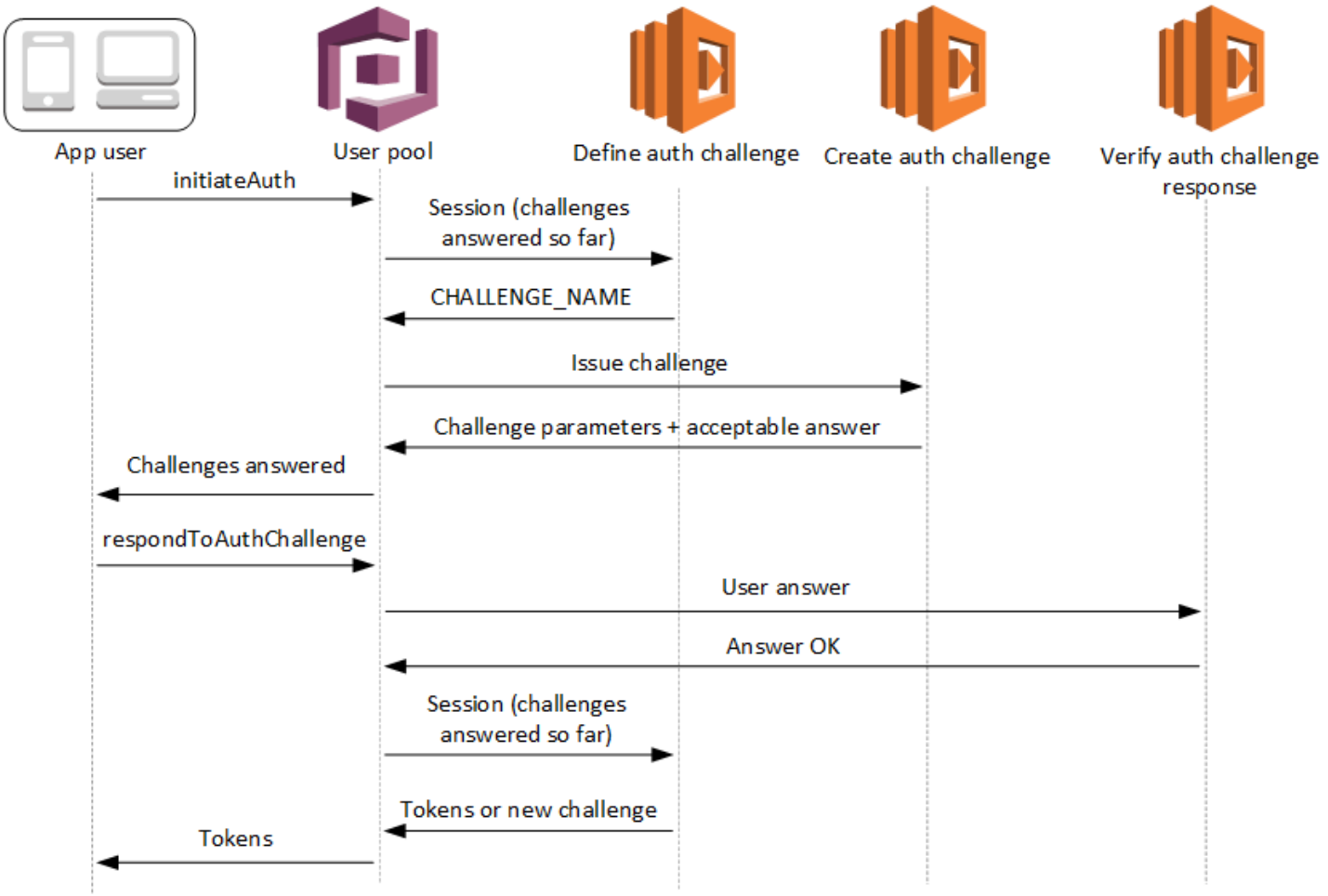
```
import os
def lambda_handler(event, context):

    # Send post authentication data to Cloudwatch logs
    print ("Authentication successful")
    print ("Trigger function =", event['triggerSource'])
    print ("User pool = ", event['userPoolId'])
    print ("App client ID = ", event['callerContext']['clientId'])
    print ("User ID = ", event['userName'])

    # Return to Amazon Cognito
    return event
```

Amazon Cognito 會將事件資訊傳遞至您的 Lambda 函數。此函數會將相同事件物件傳回 Amazon Cognito，並在回應中附上任何變更。在 Lambda 主控台中，您可使用與 Lambda 觸發程序相關聯的資料來設定測試事件。下列是此程式碼範例的測試事件：

JSON

```
{
  "triggerSource": "testTrigger",
  "userPoolId": "testPool",
  "userName": "testName",
  "callerContext": {
      "clientId": "12345"
  },
  "response": {}
}
```

自訂身分驗證挑戰 Lambda 觸發程序



這些 Lambda 觸發程序會發出並驗證自身的挑戰，做為使用者集區自訂身分驗證流程的一部分。

定義驗證挑戰

Amazon Cognito 會叫用此觸發程序來啟動自訂身分驗證流程。

建立驗證挑戰

Amazon Cognito 會在 Define Auth Challenge (定義驗證挑戰) 之後叫用此觸發程序，以建立自訂挑戰。

確認驗證挑戰回應

Amazon Cognito 會叫用此觸發程序來驗證最終使用者對自訂挑戰的回應是否有效。

您可以將新挑戰類型納入這些挑戰 Lambda 觸發程序。例如，這些挑戰類型可能包含 CAPTCHAs 或動態挑戰問題。

您可以使用使用者集區 InitiateAuth 和 RespondToAuthChallenge API 方法，將身分驗證一般化為兩個常見步驟。

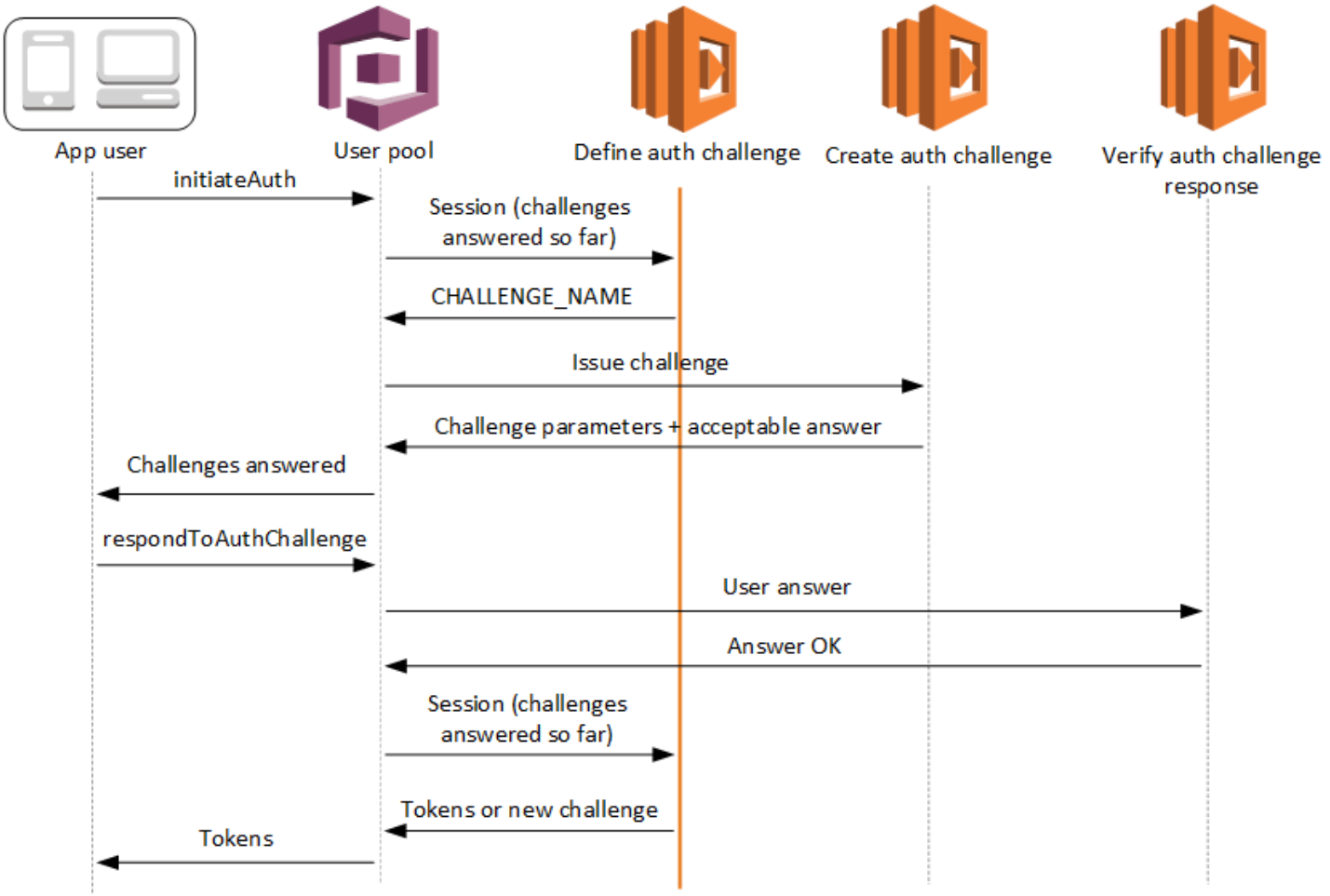
在此流程中，使用者身分驗證會透過回答連續挑戰，直到驗證失敗或者使用者發出權杖為止。這兩個 API 呼叫可重複以包含不同的挑戰。

> **Note**
>
> Amazon Cognito 託管 UI 不支援使用自訂身分驗證挑戰 Lambda 觸發程序進行自訂身分驗證。

主題

- 定義驗證挑戰 Lambda 觸發程序
- 建立驗證挑戰 Lambda 觸發程序
- 確認驗證挑戰回應 Lambda 觸發程序

定義驗證挑戰 Lambda 觸發程序



定義驗證挑戰

Amazon Cognito 會叫用此觸發程序來啟動自訂身分驗證流程。

此 Lambda 觸發程序的請求包含 `session`。`session` 參數是一個陣列，其中包含目前身分驗證程序期間向使用者提出的所有挑戰。該要求也包含對應的結果。`session` 陣列依照時間順序存放挑戰詳細內容 (`ChallengeResult`)。挑戰 `session[0]` 代表使用者收到的第一個挑戰。

您可以讓 Amazon Cognito 在發出自訂挑戰之前驗證使用者密碼。當您在自訂挑戰流程中執行 SRP 驗證時，會執行請求率配額驗證類別中關聯的任何 Lambda 觸發程序。下列為此程序的概觀：

1. 您的應用程式使用 `AuthParameters` 對應透過呼叫 `InitiateAuth` 或 `AdminInitiateAuth` 啟動登入。參數必須包含 `CHALLENGE_NAME: SRP_A`，以及 `SRP_A` 和 `USERNAME` 的值。
2. Amazon Cognito 以包含 `challengeName: SRP_A` 和 `challengeResult: true` 初始工作階段，叫用您的定義驗證挑戰 Lambda 觸發程序。

3. 收到這些輸入後，您的 Lambda 函數會以 `challengeName: PASSWORD_VERIFIER`、`issueTokens: false`、`failAuthentication: false` 回應。
4. 如果密碼驗證成功，Amazon Cognito 會再次以包含 `challengeName: PASSWORD_VERIFIER` 和 `challengeResult: true` 的新工作階段，叫用您的 Lambda 函數。
5. 您的 Lambda 函數會以 `challengeName: CUSTOM_CHALLENGE`、`issueTokens: false` 和 `failAuthentication: false` 回應，啟動您的自訂挑戰。如果您不想透過密碼驗證開始自訂驗證流程，您可以使用 `AuthParameters` 對應 (包括 `CHALLENGE_NAME: CUSTOM_CHALLENGE`) 啟動登入。
6. 除非已回答所有挑戰，否則挑戰迴圈會不斷重複。

主題

- 定義驗證挑戰 Lambda 觸發程序參數
- 定義驗證挑戰範例

定義驗證挑戰 Lambda 觸發程序參數

Amazon Cognito 傳遞至此 Lambda 函數的請求，是以下參數和 Amazon Cognito 新增至所有請求的常用參數之組合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
                . . .
        },
        "session": [
            ChallengeResult,
            . . .
        ],
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "challengeName": "string",
```

```
        "issueTokens": boolean,
        "failAuthentication": boolean
    }
}
```

定義驗證挑戰請求參數

當 Amazon Cognito 叫用您的 Lambda 函數時，Amazon Cognito 會提供以下參數：

userAttributes

代表使用者屬性的一或多組名稱/值。

userNotFound

當您的使用者集區用戶端的 `PreventUserExistenceErrors` 設定為 `ENABLED` 時，Amazon Cognito 將會填入的布林值。值 `true` 表示使用者 ID (使用者名稱、電子郵件地址和其他詳細資訊) 與任何現有使用者不相符。當 `PreventUserExistenceErrors` 設定為 `ENABLED`，該服務不會通知應用程式有不存在的使用者。我們建議您的 Lambda 函數維持相同的使用者體驗並考慮延遲。如此一來，當使用者存在或不存在時，呼叫者無法偵測到不同的行為。

工作階段

`ChallengeResult` 元素的陣列。每個都包含下列元素：

challengeName

下列其中一種挑戰類型：`CUSTOM_CHALLENGE`、`SRP_A`、`PASSWORD_VERIFIER`、`SMS_MFA`, `DEVICE_SRP_AUTH`、`DEVICE_PASSWORD_VERIFIER` 或`ADMIN_NO_SRP_AUTH`。

當您的定義驗證挑戰函數向已設定多重因素身分驗證的使用者發出 `PASSWORD_VERIFIER` 挑戰時，Amazon Cognito 會隨之發出 `SMS_MFA` 挑戰。您的函數中，包括處理來自 `SMS_MFA` 挑戰的輸入事件。您無需從定義驗證挑戰函數叫用 `SMS_MFA` 挑戰。

> **Important**
>
> 當您的函數正在確定使用者是否成功驗證，以及是否該發出權杖時，請務必檢查您定義驗證挑戰函數中的 `challengeName`，並且驗證是否與期望值相符。

challengeResult

如果使用者順利完成挑戰，則設定為 `true`，否則設定為 `false`。

challengeMetadata

您的自訂挑戰名稱。唯有在 `challengeName` 為 `CUSTOM_CHALLENGE` 時使用。

clientMetadata

您可以做為 Lambda 函數的自訂輸入提供的一個或多個鍵值組，該函數是您用於定義驗證挑戰觸發程序所指定。若要將此資料傳遞至您的 Lambda 函數，您可以使用 AdminRespondToAuthChallenge 和 RespondToAuthChallenge API 操作中的 `ClientMetadata` 參數。叫用定義驗證挑戰函數的請求不包含傳遞至 AdminInitiateAuth 和 InitiateAuth API 操作的 ClientMetadata 參數中的資料。

定義驗證挑戰回應參數

在回應中，您可以傳回身分驗證程序的下一個階段。

challengeName

包含下一個挑戰名稱的字串。如果您想要向使用者提出新的挑戰，請在這裡指定挑戰名稱。

issueTokens

如果您判斷使用者已完成身分驗證挑戰，請設定為 `true`。如果使用者未成功完成挑戰，請設定為 `false`。

failAuthentication

如果您要結束目前的身分驗證程序，請設定為 `true`。若要繼續目前的身分驗證程序，請設定為 `false`。

定義驗證挑戰範例

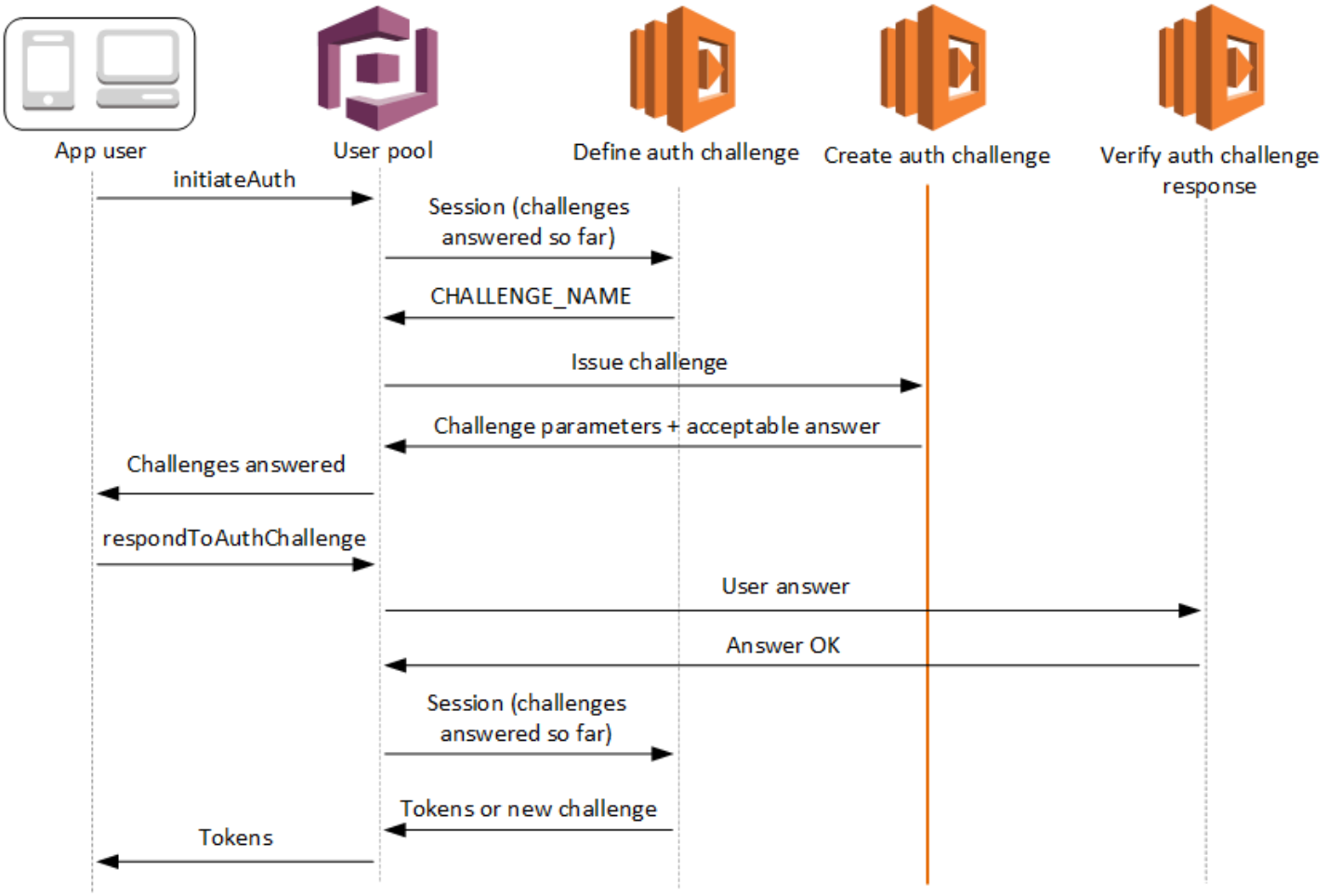
此範例會定義一系列用來進行身分驗證的挑戰，且唯有在使用者完成所有挑戰時，才會發出權杖。

Node.js

```
const handler = async (event) => {
  if (
    event.request.session.length == 1 &&
    event.request.session[0].challengeName == "SRP_A"
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "PASSWORD_VERIFIER";
```

```
  } else if (
    event.request.session.length == 2 &&
    event.request.session[1].challengeName == "PASSWORD_VERIFIER" &&
    event.request.session[1].challengeResult == true
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "CUSTOM_CHALLENGE";
  } else if (
    event.request.session.length == 3 &&
    event.request.session[2].challengeName == "CUSTOM_CHALLENGE" &&
    event.request.session[2].challengeResult == true
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "CUSTOM_CHALLENGE";
  } else if (
    event.request.session.length == 4 &&
    event.request.session[3].challengeName == "CUSTOM_CHALLENGE" &&
    event.request.session[3].challengeResult == true
  ) {
    event.response.issueTokens = true;
    event.response.failAuthentication = false;
  } else {
    event.response.issueTokens = false;
    event.response.failAuthentication = true;
  }

  return event;
};

export { handler }
```

建立驗證挑戰 Lambda 觸發程序



建立驗證挑戰

如果自訂挑戰已指定為 Define Auth Challenge (定義驗證挑戰) 觸發程序的一部分，Amazon Cognito 就會在 Define Auth Challenge (定義驗證挑戰) 之後叫用此觸發程序。它會建立自訂身分驗證流程。

叫用此 Lambda 觸發程序可以建立要向使用者提出的挑戰。此 Lambda 觸發程序的請求包含 `challengeName` 和 `session`。`challengeName`是要向使用者提出之下一個挑戰的名稱字串。此屬性的值是設定在「定義驗證挑戰 Lambda 觸發程序」中。

除非所有挑戰都已回答，否則挑戰迴圈會不斷重複。

主題

- 建立驗證挑戰 Lambda 觸發程序參數
- 建立驗證挑戰範例

建立驗證挑戰 Lambda 觸發程序參數

Amazon Cognito 傳遞至此 Lambda 函數的請求，是以下參數和 Amazon Cognito 新增至所有請求的常用參數之組合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "challengeName": "string",
        "session": [
            ChallengeResult,
            . . .
        ],
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "publicChallengeParameters": {
            "string": "string",
            . . .
        },
        "privateChallengeParameters": {
            "string": "string",
            . . .
        },
        "challengeMetadata": "string"
    }
}
```

建立驗證挑戰請求參數

userAttributes

代表使用者屬性的一或多組名稱/值。

userNotFound

當您的使用者集區用戶端的 PreventUserExistenceErrors 設定為 ENABLED 時，將會填入此布林值。

challengeName

新挑戰的名稱。

工作階段

工作階段元素是 ChallengeResult元素陣列，每個陣列都包含下列元素：

challengeName

挑戰類型。下列其中一項："CUSTOM_CHALLENGE"、"PASSWORD_VERIFIER"、"SMS_MFA"、"DEVICE_SRP_AUTH"、"D或 "ADMIN_NO_SRP_AUTH"。

challengeResult

如果使用者順利完成挑戰，則設定為 true，否則設定為 false。

challengeMetadata

您的自訂挑戰名稱。唯有在 challengeName 為 "CUSTOM_CHALLENGE" 時使用。

clientMetadata

您可以做為 Lambda 函數的自訂輸入提供的一個或多個鍵值組，該函數是您用於建立驗證挑戰觸發程序所指定。您可以使用 AdminRespondToAuthChallenge 和 RespondToAuthChallenge API 動作中的 ClientMetadata 參數，將此資料傳遞至您的 Lambda 函數。叫用建立身分驗證挑戰函數的請求不包含傳遞至 AdminInitiateAuth 和 InitiateAuth API 操作的 ClientMetadata 參數中的資料。

建立驗證挑戰回應參數

publicChallengeParameters

可讓用戶端應用程式用在要向使用者提出的挑戰中的一或多個鍵值組。此參數應包含所有必要資訊，以準確地向使用者提出挑戰。

privateChallengeParameters

此參數僅供「確認驗證挑戰回應 Lambda 觸發程序」使用。此參數應包含驗證使用者對挑戰的回應時，所有必要的資訊。換言之，publicChallengeParameters參數包含向使用者提出的問題，而 privateChallengeParameters 包含問題的有效答案。

challengeMetadata

您的自訂挑戰名稱 (如果這是自訂挑戰)。

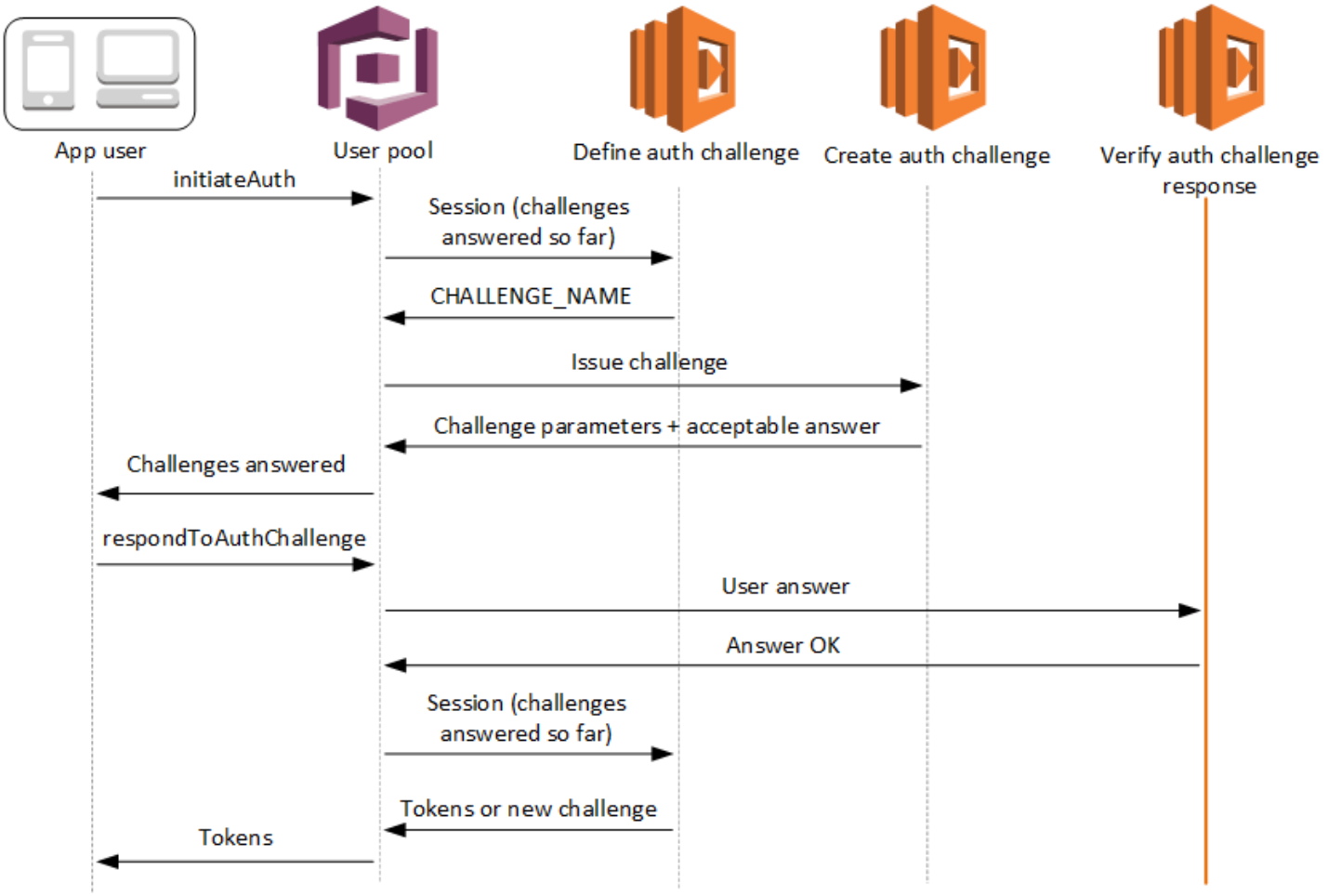
建立驗證挑戰範例

建立 CAPTCHA 以做為對使用者的挑戰。CAPTCHA 影像的 URL 會以 "`captchaUrl`" 新增至公有挑戰參數，而預期的答案會新增至私有挑戰參數。

Node.js

```
const handler = async (event) => {
  if (event.request.challengeName !== "CUSTOM_CHALLENGE") {
    return event;
  }

  if (event.request.session.length === 2) {
    event.response.publicChallengeParameters = {};
    event.response.privateChallengeParameters = {};
    event.response.publicChallengeParameters.captchaUrl = "url/123.jpg";
    event.response.privateChallengeParameters.answer = "5";
  }

  if (event.request.session.length === 3) {
    event.response.publicChallengeParameters = {};
    event.response.privateChallengeParameters = {};
    event.response.publicChallengeParameters.securityQuestion =
      "Who is your favorite team mascot?";
    event.response.privateChallengeParameters.answer = "Peccy";
  }

  return event;
};

export { handler }
```

確認驗證挑戰回應 Lambda 觸發程序



確認驗證挑戰回應

Amazon Cognito 會叫用此觸發程序來驗證使用者對自訂驗證挑戰的回應是否有效。這屬於使用者集區[自訂身分驗證流程]的一部分。

此觸發程序的請求包含 `privateChallengeParameters`和 `challengeAnswer` 參數。「建立驗證挑戰 Lambda 觸發程序」傳回 `privateChallengeParameters` 值，其中包含預期的使用者回應。`challengeAnswer`參數包含使用者對挑戰的回應。

回應包含 `answerCorrect` 屬性。如果使用者成功完成挑戰，Amazon Cognito 會將屬性值設定為 `true`。如果使用者未能成功完成挑戰，Amazon Cognito 會將此值設定為 `false`。

除非使用者已回答所有挑戰，否則挑戰迴圈會不斷重複。

主題

- 確認驗證挑戰 Lambda 觸發程序參數
- 確認驗證挑戰回應範例

確認驗證挑戰 Lambda 觸發程序參數

Amazon Cognito 傳遞至此 Lambda 函數的請求，是以下參數和 Amazon Cognito 新增至所有請求的常用參數之組合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "privateChallengeParameters": {
            "string": "string",
            . . .
        },
        "challengeAnswer": "string",
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "answerCorrect": boolean
    }
}
```

確認驗證挑戰請求參數

userAttributes

此參數包含代表使用者屬性的一或多個名稱/值對。

userNotFound

當 Amazon Cognito 將您的使用者集區用戶端的 `PreventUserExistenceErrors` 設定為 `ENABLED` 時，Amazon Cognito 會填入此布林值。

privateChallengeParameters

此參數來自建立身分驗證挑戰觸發程序。為了確定使用者是否已通過挑戰，Amazon Cognito 會將此參數與使用者的 challengeAnswer 進行比較。

此參數包含驗證使用者對挑戰的回應時，所有必要的資訊。此訊息包含亞 Amazon Cognito 向使用者提出的問題 (`publicChallengeParameters`)，以及問題的有效答案 (`privateChallengeParameters`)。只有「確認身分驗證挑戰回應 Lambda 觸發程序」會使用此參數。

challengeAnswer

此參數值是使用者對挑戰回應的答案。

clientMetadata

此參數包含您可以做為 Lambda 函數的自訂輸入提供的一個或多個鍵值組，該函數是您用於確認身分驗證挑戰觸發程序所指定。若要將此資料傳遞至您的 Lambda 函數，您可以使用 AdminRespondToAuthChallenge 和 RespondToAuthChallenge API 操作中的 ClientMetadata 參數。Amazon Cognito 不包含其傳遞至確認身分驗證挑戰函數之請求中的 AdminInitiateAuth 和 InitiateAuth API 操作中的 ClientMetadata 參數的資料。

確認驗證挑戰回應參數

answerCorrect

如果使用者成功完成挑戰，Amazon Cognito 會將此參數設定為 `true`。如果使用者未能成功完成挑戰，Amazon Cognito 會將此參數設定為 `false`。

確認驗證挑戰回應範例

在這個範例中，Lambda 函數會檢查使用者對挑戰的回應是否符合預期的回應。如果使用者的回應符合預期的回應，則 Amazon Cognito 會將 `answerCorrect` 參數設定為 `true`。

Node.js

```
const handler = async (event) => {
  if (
    event.request.privateChallengeParameters.answer ==
    event.request.challengeAnswer
  ) {
```

```
    event.response.answerCorrect = true;
  } else {
    event.response.answerCorrect = false;
  }

  return event;
};

export { handler };
```

產生權杖前 Lambda 觸發程序

由於 Amazon Cognito 會在產生權杖之前叫用此觸發程序，因此您可以自訂使用者集區的權杖宣告。透過第一版或 `V1_0` 預先產生權杖觸發事件的基本功能，您可以自訂身份 (ID) 權杖。在具有高級安全功能活動的使用者集區中，您可以通過存取權杖自訂生成版本 2 或 `V2_0` 觸發事件。

Amazon Cognito 會將 `V1_0` 事件作為請求傳送至您的函數，其中包含會寫入 ID 權杖的資料。`V2_0` 事件是單一請求，其中包含 Amazon Cognito 將寫入身分和存取權杖的資料。若要自訂兩個權杖，您必須更新您的函數以使用最新的觸發程序版本，並在相同的回應中傳送兩個權杖的資料。

在 Amazon Cognito 將權杖發佈到您的應用程式之前，此 Lambda 觸發程序可以新增、移除和修改身分與存取權杖中的某些宣告。若要使用此功能，請從 Amazon Cognito 使用者集區主控台來連結一個 Lambda 函數，或是透過 AWS Command Line Interface (AWS CLI) 來更新您的使用者集區 `LambdaConfig`。

活動版本

您的使用者集區可以向 Lambda 函數傳遞不同版本的權杖產生觸發器事件。`V1_0`觸發器提供用於修改 ID 令牌的參數。`V2_0`觸發器提供以下參數。

1. `V1_0`觸發器的功能。
2. 自定義訪問令牌的能力。
3. 將複雜數據類型傳遞給 ID 和訪問令牌聲明值的能力：
   - 字串
   - Number
   - Boolean
   - 字符串，數字，布爾值或任何這些組合的數組

- JSON

> **Note**
>
> 在 ID 令牌中，您可以將複雜對象填充到聲明的值 `phone_number_verifiedemail_verified`，除了`updated_at`，和`address`。

依預設，使用者集區會傳送`V1_0`事件。若要設定使用者集區以傳送`V2_0`事件，請在 Amazon Cognito 主控台中設定觸發器時，選擇基本功能 + 存取權杖自訂的觸發事件版本。您也可以在UpdateUserPool 或 CreateUserPool API 請求`LambdaVersion`的LambdaConfig參數中設定的值。使用`V2_0`事件進行存取權杖自訂需支付額外費用。如需詳細資訊，請參閱 Amazon Cognito 定價。

排除宣告和範圍

Amazon Cognito 限制您可以在存取權和身分權杖中新增、修改或隱藏的宣告和範圍。如果您的 Lambda 函數嘗試為這些宣告中的任何一個設定值，Amazon Cognito 會發出具有原始宣告值的權杖 (如果請求中存在該權杖的話)。

共用宣告

- `acr`
- `amr`
- `at_hash`
- `auth_time`
- `azp`
- `exp`
- `iat`
- `iss`
- `jti`
- `nbf`
- `nonce`
- `origin_jti`
- `sub`
- `token_use`

ID 令牌宣告

- identities
- aud
- cognito:username

存取權杖宣告

- username
- client_id
- scope

> **Note**
>
> 您可以使用 scopesToAdd 和 scopesToSuppress 回應值變更存取權杖中的範圍，但不能直接修改 scope 宣告。您無法新增開頭為 aws.cognito 的範圍，包括使用者集區保留的範圍 aws.cognito.signin.user.admin。

- device_key
- event_id
- version

您無法使用以下字首新增或覆寫宣告，但可以隱藏它們，或阻止其出現在權杖中。

- dev:
- cognito:

您可以新增 aud 宣告以存取權杖，但其值必須與目前工作階段的應用程式用戶端 ID 相符。您可以從 event.callerContext.clientId 的請求事件中中衍生出用戶端 ID。

自訂身分權杖

透過產生權杖前 Lambda 觸發程序，您可以自訂使用者集區中身分 (ID) 權杖的內容。ID 權杖提供來自受信任身分來源的使用者屬性，以便登入 Web 或行動應用程式。如需 ID 權杖的詳細資訊，請參閱 使用 ID 權杖。

產生權杖前 Lambda 觸發程序搭配 ID 權杖的使用方式如下。

- 在執行階段變更您的使用者從身分集區請求的 IAM 角色。
- 從外部來源新增使用者屬性。
- 新增或取代現有的使用者屬性值。
- 禁止披露使用者屬性，因為使用者的授權範圍以及您授予應用程式用戶端的屬性的讀取存取權限，這些屬性可能會傳遞給您的應用程式。

自訂存取權杖

透過產生權杖前 Lambda 觸發程序，您可以自訂使用者集區中存取權杖的內容。存取權杖授權使用者從受存取保護的資源 (例如 Amazon Cognito 權杖授權的 API 操作和第三方 API) 擷取資訊。雖然您可以使用用戶端登入資料授與 Amazon Cognito 產生 machine-to-machine (M2M) 授權的存取權杖，但 M2M 請求不會叫用權杖產生前觸發器函數，也無法發出自訂存取權杖。如需存取權杖的詳細資訊，請參閱 使用存取權杖。

產生權杖前 Lambda 觸發程序搭配存取權杖的使用方式包括下列項目。

- 在 `scope` 宣告中新增或隱藏 OAuth 2.0 範圍。例如，您可以將範圍新增至僅指派範圍 `aws.cognito.signin.user.admin` 的 Amazon Cognito 使用者集區 API 身份驗證所產生的存取權杖。
- 變更使用者集區群組中的使用者成員資格。
- 新增尚未存在於 Amazon Cognito 存取權杖中的宣告。
- 禁止披露宣告，否則宣告會傳遞給您的應用程式。

若要支援使用者集區中的存取自訂，您必須設定使用者集區以產生觸發請求的更新版本。更新您的使用者集區，如下列流程所示。

AWS Management Console

若要支援產生權杖前 Lambda 觸發程序中的存取權杖自訂

1. 前往 Amazon Cognito 主控台，然後選擇 User Pools (使用者集區)。
2. 從清單中選擇現有的使用者集區，或建立使用者集區。
3. 如果您尚未啟用，請從應用程式整合標籤啟用進階安全性功能。
4. 選擇 User pool properties (使用者集區屬性) 索引標籤並找到 Lambda triggers (Lambda 觸發程序)。
5. 新增或編輯產生權杖前觸發程序。

6. 在指派 Lambda 函數下選擇一個 Lambda 函數。
7. 選擇基本功能 + 存取權杖自訂的觸發事件版本。此設定會更新 Amazon Cognito 傳送至您函數的請求參數，以包含用於存取權杖自訂的欄位。

User pools API

若要支援產生權杖前 Lambda 觸發程序中的存取權杖自訂

產生CreateUserPool或 UpdateUserPoolAPI 要求。您必須為不想設定為預設值的所有參數指定一個值。如需詳細資訊，請參閱 更新使用者集區組態。

在請求的 `LambdaVersion` 參數中包含以下內容。`V2_0` 的 `LambdaVersion` 值會導致您的使用者集區新增參數以進行存取權杖自訂。若要叫用特定函數版本，請使用 Lambda 函數 ARN，並將函數版本作為 `LambdaArn` 的值。

```
"PreTokenGenerationConfig": {
   "LambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction",
   "LambdaVersion": "V2_0"
},
```

主題

- 產生權杖前 Lambda 觸發程序來源
- 產生權杖前 Lambda 觸發程序參數
- 觸發權杖前事件版本二範例：新增和抑制宣告，範圍和群組
- 前令牌生成事件版本二示例：添加具有複雜對象的聲明
- 產生權杖前事件版本一範例：新增宣告及抑制現有宣告
- 產生權杖前事件版本一範例：修改使用者的群組成員資格

產生權杖前 Lambda 觸發程序來源

triggerSource 值	事件
`TokenGeneration_HostedAuth`	在身分驗證期間，從 Amazon Cognito 託管的 UI 登入頁面呼叫。
`TokenGeneration_Authentication`	在使用者身分驗證流程完成之後呼叫。

triggerSource 值	事件
TokenGeneration_NewPasswordChallenge	在管理員建立使用者之後呼叫。當使用者必須變更臨時密碼時，會叫用此流程。
TokenGeneration_AuthenticateDevice	在使用者裝置的身分驗證結束時呼叫。
TokenGeneration_RefreshTokens	當使用者嘗試重新整理身分和存取權杖時呼叫。

產生權杖前 Lambda 觸發程序參數

Amazon Cognito 傳遞至此 Lambda 函數的請求，是以下參數和 Amazon Cognito 新增至所有請求的常用參數之組合。當您將產生權杖前 Lambda 觸發程序新增至使用者集區時，您可以選擇觸發程序版本。此版本決定 Amazon Cognito 是否將請求傳遞至您的 Lambda 函數 以及用於存取權杖自訂的其他參數。

Version 1

版本 1 令牌可以在 ID 令牌中設置組成員資格，IAM 角色和新聲明。

```
{
    "request": {
        "userAttributes": {"string": "string"},
        "groupConfiguration": {
                "groupsToOverride": [
                    "string",
                    "string"
                ],
                "iamRolesToOverride": [
                    "string",
                    "string"
                ],
                "preferredRole": "string"
        },
        "clientMetadata": {"string": "string"}
    },
    "response": {
        "claimsOverrideDetails": {
            "claimsToAddOrOverride": {"string": "string"},
            "claimsToSuppress": [
```

```
                "string",
                "string"
            ],
            "groupOverrideDetails": {
                "groupsToOverride": [
                    "string",
                    "string"
                ],
                "iamRolesToOverride": [
                    "string",
                    "string"
                ],
                "preferredRole": "string"
            }
        }
    }
}
```

Version 2

版本 2 請求事件添加了自定義訪問令牌的字段。它還增加了對響應對象複雜`claimsToOverride`數據類型的支持。您的 Lambda 函數可以傳回下列類型的資料值`claimsToOverride`：

- 字串
- Number
- Boolean
- 字符串，數字，布爾值或任何這些組合的數組
- JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string"
        },
        "scopes": ["string", "string"],
        "groupConfiguration": {
            "groupsToOverride": ["string", "string"],
            "iamRolesToOverride": ["string", "string"],
            "preferredRole": "string"
```

```
        },
        "clientMetadata": {
            "string": "string"
        }
    },
    "response": {
        "claimsAndScopeOverrideDetails": {
            "idTokenGeneration": {
                "claimsToAddOrOverride": {
                    "string": [accepted datatype]
                },
                "claimsToSuppress": ["string", "string"]
            },
            "accessTokenGeneration": {
                "claimsToAddOrOverride": {
                    "string": [accepted datatype]
                },
                "claimsToSuppress": ["string", "string"],
                "scopesToAdd": ["string", "string"],
                "scopesToSuppress": ["string", "string"]
            },
            "groupOverrideDetails": {
                "groupsToOverride": ["string", "string"],
                "iamRolesToOverride": ["string", "string"],
                "preferredRole": "string"
            }
        }
    }
}
```

產生權杖前請求參數

名稱	描述	最小觸發事件版本
userAttributes	您的使用者在使用者集區中的設定檔屬性。	1
groupConfiguration	包含目前群組組態的輸入物件。物件包含 `groupsToOverride` 、`iamRolesToOverride` 和 `preferredRole` 。	1

名稱	描述	最小觸發事件版本
groupsToOverride	您的用戶所屬的使用者集區。	1
iamRolesTo覆寫	您可以將使用者集區群組與 AWS Identity and Access Management (IAM) 角色建立關聯。此元素是使用者所屬群組中所有 IAM 角色的清單。	1
preferredRole	您可以為使用者集區群組設定優先順序。此元素包含 `groupsToOverride` 元素中優先度最高之群組的 IAM 角色名稱。	1
clientMetadata	針對權杖產生前觸發程序，您可以做為 Lambda 函數 的自訂輸入來指定與 提供的一個或多個鍵值組。 若要將此資料傳遞至 Lambda 函數，請在AdminRespondToAuthChallenge和 RespondToAuthChallengeAPI 作業中使用 ClientMetadata參數。Amazon Cognito 不會在傳遞給預先權杖產生函`ClientMetadata` 數的請求中包含來自參數AdminInitiateAuth和 InitiateAuthAPI 操作的資料。	1
範圍	您的使用者的 OAuth 2.0 範圍。存取權杖中存在的範圍是使用者請求的使用者集區標準範圍和自訂範圍，以及您授權應用程式用戶端發佈的範圍。	2

產生權杖前回應參數

名稱	描述	最小觸發事件版本
claimsOverrideDetails	`V1_0` 觸發事件中所有元素的容器。	1

名稱	描述	最小觸發事件版本
claimsAndScopeOverrideDetails	`V2_0` 觸發事件中所有元素的容器。	2
idTokenGeneration	您要在使用者 ID 權杖中覆寫、新增或抑制的宣告。ID 權杖自訂值的父元素僅出現在版本 2 事件中，但子元素出現在版本 1 事件中。	2
accessTokenGeneration	您想要在使用者的存取權杖中覆寫、新增或抑制的宣告和範圍。此存取權杖自訂值的父元素僅出現在版本 2 事件中。	2
claimsToAddOrOverride	您要新增或修改的一或多個宣告及其值的映射。對於與群組相關的宣告，請改用 `groupOverrideDetails` 。 在版本 2 事件中，此元素顯示於 `accessTokenGeneration` 和 `idTokenGeneration` 之下。	1*
claimsToSuppress	您希望 Amazon Cognito 抑制的宣告清單。如果您的函數抑制並取代了宣告值，則 Amazon Cognito 將抑制宣告。 在版本 2 事件中，此元素顯示於 `accessTokenGeneration` 和 `idTokenGeneration` 之下。	1

名稱	描述	最小觸發事件版本
groupOverrideDetails	包含目前群組組態的輸出物件。物件包含 groupsToOverride 、iamRolesToOverride 和preferredRole 。 您的函數將使用您提供的物件取代 groupOverrideDetails 物件。如果您在回應中提供空的或空物件，則 Amazon Cognito 會抑制這些群組。若要將現有群組組態保留原狀，請將請求的 groupConfiguration 物件值複製到回應中的 groupOverrideDetails 物件。然後將其傳回服務。 Amazon Cognito ID 權杖和存取權杖都包含 cognito:groups 宣告。您的 groupOverrideDetails 物件取代存取權杖以及 ID 權杖中的 cognito:groups 宣告。	1
scopesToAdd	您想要新增至使用者存取權杖中 scope 宣告的 OAuth 2.0 範圍清單。您無法新增包含一或多個空白字元的範圍值。	2
scopesToSuppress	您想要從使用者存取權杖的 scope 宣告中移除的 OAuth 2.0 範圍清單。	2

* 版本 1 事件的回應物件可傳回字串。對版本 2 事件的響應對象可以返回複雜的對象。

觸發權杖前事件版本二範例：新增和抑制宣告，範圍和群組

此範例會對使用者的權杖進行下列修改。

1. 在 ID 權杖中將其 family_name 設置為 Doe。
2. 防止 email 和 phone_number 宣告出現在 ID 權杖中。
3. 將其 ID 權杖 cognito:roles 宣告設置為 "arn:aws:iam::123456789012:role\/sns_callerA","arn:aws:iam::123456789012:role\/sns_callerC","arn:aws:iam::123456789012:role\/sns_callerB"。
4. 將其 ID 權杖 cognito:preferred_role 宣告設置為 arn:aws:iam::123456789012:role/sns_caller。

5. 將範圍 openid、email 和 solar-system-data/asteroids.add 新增到存取權杖。
6. 抑制存取權杖的範圍 phone_number 和 aws.cognito.signin.user.admin。
   移除 phone_number 可防止從 userInfo 中擷取使用者的電話號碼。移除 aws.cognito.signin.user.admin 可防止使用者透過 Amazon Cognito 使用者集區 API 讀取和修改自身設定檔的 API 請求。

   > **Note**
   >
   > 如存取權杖中的剩餘範圍包含 openid 與至少一個標準範圍，從範圍中刪除 phone_number 僅會防止擷取用戶的電話號碼。如需詳細資訊，請參閱 關於範圍。

7. 將其 ID 和存取權杖 cognito:groups 宣告設置為 "new-group-A","new-group-B","new-group-C"。

JavaScript

```
export const handler = function(event, context) {
  event.response = {
    "claimsAndScopeOverrideDetails": {
      "idTokenGeneration": {
        "claimsToAddOrOverride": {
          "family_name": "Doe"
    },
        "claimsToSuppress": [
          "email",
          "phone_number"
    ]
      },
      "accessTokenGeneration": {
        "scopesToAdd": [
          "openid",
          "email",
          "solar-system-data/asteroids.add"
        ],
        "scopesToSuppress": [
          "phone_number",
          "aws.cognito.signin.user.admin"
        ]
      },
      "groupOverrideDetails": {
        "groupsToOverride": [
```

```
    "new-group-A",
    "new-group-B",
    "new-group-C"
  ],
        "iamRolesToOverride": [
          "arn:aws:iam::123456789012:role/new_roleA",
          "arn:aws:iam::123456789012:role/new_roleB",
          "arn:aws:iam::123456789012:role/new_roleC"
        ],
        "preferredRole": "arn:aws:iam::123456789012:role/new_role",
      }
    }
  };
  // Return to Amazon Cognito
  context.done(null, event);
};
```

Amazon Cognito 會將事件資訊傳遞至您的 Lambda 函數。此函數會將相同事件物件傳回 Amazon Cognito，並在回應中附上任何變更。在 Lambda 主控台中，您可使用與 Lambda 觸發程序相關聯的資料來設定測試事件。下列是此程式碼範例的測試事件：

JSON

```
{
    "version": "2",
    "triggerSource": "TokenGeneration_Authentication",
    "region": "us-east-1",
    "userPoolId": "us-east-1_EXAMPLE",
    "userName": "JaneDoe",
    "callerContext": {
        "awsSdkVersion": "aws-sdk-unknown-unknown",
        "clientId": "1example23456789"
    },
    "request": {
        "userAttributes": {
            "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "cognito:user_status": "CONFIRMED",
            "email_verified": "true",
            "phone_number_verified": "true",
            "phone_number": "+12065551212",
            "family_name": "Zoe",
            "email": "Jane.Doe@example.com"
```

```
        },
        "groupConfiguration": {
            "groupsToOverride": ["group-1", "group-2", "group-3"],
            "iamRolesToOverride": ["arn:aws:iam::123456789012:role/sns_caller1",
 "arn:aws:iam::123456789012:role/sns_caller2", "arn:aws:iam::123456789012:role/
sns_caller3"],
            "preferredRole": ["arn:aws:iam::123456789012:role/sns_caller"]
        },
        "scopes": [
            "aws.cognito.signin.user.admin", "openid", "email", "phone"
        ]
    },
    "response": {
        "claimsAndScopeOverrideDetails": []
    }
}
```

前令牌生成事件版本二示例：添加具有複雜對象的聲明

此範例會對使用者的權杖進行下列修改。

1. 將數字，字符串，布爾和 JSON 類型的聲明添加到 ID 令牌。這是第二版觸發器事件對 ID 令牌可用的唯一更改。
2. 將數字，字符串，布爾和 JSON 類型的聲明添加到訪問令牌。
3. 將三個範圍添加到訪問令牌。
4. 隱藏 ID `email` 和存取權杖中的和`sub`宣告。
5. 隱藏存取權杖中的`aws.cognito.signin.user.admin`範圍。

JavaScript

```
export const handler = function(event, context) {

    var scopes = ["MyAPI.read", "MyAPI.write", "MyAPI.admin"]
    var claims = {}
    claims["aud"]= event.callerContext.clientId;
    claims["booleanTest"] = false;
    claims["longTest"] = 9223372036854775807;
    claims["exponentTest"] = 1.7976931348623157E308;
    claims["ArrayTest"] = ["test", 9223372036854775807, 1.7976931348623157E308,
 true];
```

```
    claims["longStringTest"] = "\{\
        \"first_json_block\": \{\
            \"key_A\": \"value_A\",\
            \"key_B\": \"value_B\"\
        \},\
        \"second_json_block\": \{\
            \"key_C\": \{\
                \"subkey_D\": [\
                    \"value_D\",\
                    \"value_E\"\
                ],\
                \"subkey_F\": \"value_F\"\
            \},\
            \"key_G\": \"value_G\"\
        \}\
    \}";
    claims["jsonTest"] = {
     "first_json_block": {
      "key_A": "value_A",
      "key_B": "value_B"
     },
     "second_json_block": {
      "key_C": {
       "subkey_D": [
        "value_D",
        "value_E"
       ],
       "subkey_F": "value_F"
      },
      "key_G": "value_G"
     }
    };
    event.response = {
        "claimsAndScopeOverrideDetails": {
            "idTokenGeneration": {
                "claimsToAddOrOverride": claims,
                "claimsToSuppress": ["email","sub"]
            },
            "accessTokenGeneration": {
                "claimsToAddOrOverride": claims,
                "claimsToSuppress": ["email","sub"],
                "scopesToAdd": scopes,
                "scopesToSuppress": ["aws.cognito.signin.user.admin"]
            }
```

```
        }
    };
    console.info("EVENT response\n" + JSON.stringify(event, (_, v) => typeof v ===
 'bigint' ? v.toString() : v, 2))
    console.info("EVENT response size\n" + JSON.stringify(event, (_, v) => typeof v
 === 'bigint' ? v.toString() : v).length)
    // Return to Amazon Cognito
    context.done(null, event);
};
```

Amazon Cognito 會將事件資訊傳遞至您的 Lambda 函數。此函數會將相同事件物件傳回 Amazon Cognito，並在回應中附上任何變更。在 Lambda 主控台中，您可使用與 Lambda 觸發程序相關聯的資料來設定測試事件。下列是此程式碼範例的測試事件：

JSON

```
{
    "version": "2",
    "triggerSource": "TokenGeneration_HostedAuth",
    "region": "us-west-2",
    "userPoolId": "us-west-2_EXAMPLE",
    "userName": "JaneDoe",
    "callerContext": {
        "awsSdkVersion": "aws-sdk-unknown-unknown",
        "clientId": "1example23456789"
    },
    "request": {
        "userAttributes": {
            "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "cognito:user_status": "CONFIRMED"
            "email_verified": "true",
            "phone_number_verified": "true",
            "phone_number": "+12065551212",
            "email": "Jane.Doe@example.com"
        },
        "groupConfiguration": {
            "groupsToOverride": ["group-1", "group-2", "group-3"],
            "iamRolesToOverride": ["arn:aws:iam::123456789012:role/sns_caller1"],
            "preferredRole": ["arn:aws:iam::123456789012:role/sns_caller1"]
        },
        "scopes": [
            "aws.cognito.signin.user.admin",
```

```
            "phone",
            "openid",
            "profile",
            "email"
        ]
    },
    "response": {
        "claimsAndScopeOverrideDetails": []
    }
}
```

產生權杖前事件版本一範例：新增宣告及抑制現有宣告

此範例使用版本 1 觸發事件和「產生權杖前 Lambda 函數」來新增宣告及抑制現有宣告。

Node.js

```
const handler = async (event) => {
  event.response = {
    claimsOverrideDetails: {
      claimsToAddOrOverride: {
        my_first_attribute: "first_value",
        my_second_attribute: "second_value",
      },
      claimsToSuppress: ["email"],
    },
  };

  return event;
};

export { handler };
```

Amazon Cognito 會將事件資訊傳遞至您的 Lambda 函數。此函數會將相同事件物件傳回 Amazon Cognito，並在回應中附上任何變更。在 Lambda 主控台中，您可使用與 Lambda 觸發程序相關聯的資料來設定測試事件。下列是此程式碼範例的測試事件：由於程式碼 範例 會處理任何請求參數，因此可以使用具有空白請求的測試事件。如需有關常見請求參數的詳細資訊，請參閱 使用者集區 Lambda 觸發程序事件。

JSON

```
{
  "request": {},
  "response": {}
}
```

產生權杖前事件版本一範例：修改使用者的群組成員資格

此範例使用版本 1 觸發事件和「產生權杖前 Lambda 函數」來修改使用者的群組成員資格。

Node.js

```
const handler = async (event) => {
  event.response = {
    claimsOverrideDetails: {
      groupOverrideDetails: {
        groupsToOverride: ["group-A", "group-B", "group-C"],
        iamRolesToOverride: [
          "arn:aws:iam::XXXXXXXXXXX:role/sns_callerA",
          "arn:aws:iam::XXXXXXXXX:role/sns_callerB",
          "arn:aws:iam::XXXXXXXXXX:role/sns_callerC",
        ],
        preferredRole: "arn:aws:iam::XXXXXXXXXXX:role/sns_caller",
      },
    },
  };

  return event;
};

export { handler };
```

Amazon Cognito 會將事件資訊傳遞至您的 Lambda 函數。此函數會將相同事件物件傳回 Amazon Cognito，並在回應中附上任何變更。在 Lambda 主控台中，您可使用與 Lambda 觸發程序相關聯的資料來設定測試事件。下列是此程式碼範例的測試事件：

JSON

```
{
```

```
  "request": {},
  "response": {}
}
```

遷移使用者 Lambda 觸發程序

當使用者使用密碼登入時或在忘記密碼流程時，該使用者不存在於使用者集區中，Amazon Cognito 會叫用此觸發程序。Lambda 函數成功傳回後，Amazon Cognito 會在使用者集區中建立使用者。如需使用使用者遷移 Lambda 觸發程序進行身分驗證流程的詳細資訊，請參閱透過使用者遷移 Lambda 觸發程序將使用者匯入到使用者集區。

若要在登入時將使用者從現有使用者目錄遷移至 Amazon Cognito 使用者集區，或在忘記密碼流程時使用此 Lambda 觸發程序。

主題

- 遷移使用者 Lambda 觸發程序來源
- 遷移使用者 Lambda 觸發程序參數
- 範例：搭配現有密碼來遷移使用者

遷移使用者 Lambda 觸發程序來源

triggerSource 值	事件
`UserMigration_Authentication`	登入時的使用者遷移。
`UserMigration_ForgotPassword`	忘記密碼流程時的使用者遷移。

遷移使用者 Lambda 觸發程序參數

Amazon Cognito 傳遞至此 Lambda 函數的請求，是以下參數和 Amazon Cognito 新增至所有請求的常用參數之組合。

JSON

```
{
    "userName": "string",
```

```
    "request": {
        "password": "string",
        "validationData": {
            "string": "string",
            . . .
        },
        "clientMetadata": {
            "string": "string",
       . . .
        }
    },
    "response": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "finalUserStatus": "string",
        "messageAction": "string",
        "desiredDeliveryMediums": [ "string", . . .],
        "forceAliasCreation": boolean,
        "enableSMSMFA": boolean
    }
}
```

遷移使用者請求參數

使用者名稱

使用者在登入時輸入的使用者名稱。

密碼

使用者在登入時輸入的密碼。Amazon Cognito 不會在由忘記密碼流程啟動的請求中傳送此值。

validationData

一或多個鍵值組，包含使用者登入請求中的驗證資料。若要將此資料傳遞至您的 Lambda 函數，您可以使用 InitiateAuth 和 AdminInitiateAuth API 動作中的 ClientMetadata 參數。

clientMetadata

您可以做為 Lambda 函數的自訂輸入提供的一個或多個鍵值組，該函數用於遷移使用者觸發程序。若要將此資料傳遞至您的 Lambda 函數，您可以使用 AdminRespondToAuthChallenge 和 ForgotPassword API 動作中的 ClientMetadata 參數。

遷移使用者回應參數

userAttributes

此欄位為必填。

此欄位必須包含一或多個 Amazon Cognito 存放在您使用者集區中的使用者描述檔中並作為使用者屬性使用的名稱值組。您可以同時包含標準和自訂使用者屬性。自訂屬性需要 `custom:` 字首以便與標準屬性區分。如需詳細資訊，請參閱自訂屬性。

> **Note**
>
> 若要在忘記密碼流程中重設密碼，使用者必須擁有經驗證的電子郵件地址或經驗證的電話號碼。Amazon Cognito 會將包含重設密碼代碼的訊息，傳送至使用者屬性中的電子郵件地址或電話號碼。

屬性	要求
在您建立使用者集區時標記為必要的任何屬性	如果在遷移期間遺失任何必要屬性，Amazon Cognito 將會使用預設值。
`username`	如果您已經使用登入使用者名稱以外的別名屬性來設定您的使用者集區，且使用者已輸入有效的別名值作為使用者名稱，則此為必填項。此別名值可以是電子郵件地址、偏好的使用者名稱或電話號碼。 如果請求和使用者集區符合別名要求，則函數的回應必須將收到的 `username` 參數指派給別名屬性。此外，回應必須將您自己的值指派給 `username` 屬性。如果您的使用者集區不符合將接收的 `username` 映射至別名所需要的條件，則回應中的 `username` 參數必須與請求完全匹配，或省略。 **Note** `username` 在使用者集區中必須是唯一的。

finalUserStatus

您可以將此參數設定為 CONFIRMED 以自動確認您的使用者，如此一來使用者即可使用先前的密碼登入。當您將使用者設定為 CONFIRMED，使用者無需進行額外的動作就能登入。如果您沒有將此屬性設定為 CONFIRMED，它被設定為 RESET_REQUIRED。

RESET_REQUIRED 的 finalUserStatus 意味著使用者在登入時必須於遷移後立即變更密碼，並且您的用户端應用程式必須在身分驗證流程中處理 PasswordResetRequiredException。

> **Note**
>
> Amazon Cognito 不會強制執行您在使用 Lambda 觸發程序進行遷移期間為使用者集區設定的密碼強度政策。如果密碼不符合您設定的密碼政策，Amazon Cognito 仍會接受密碼，以便繼續遷移使用者。若要強制執行密碼強度政策，並拒絕不符合政策的密碼，請驗證程式碼中的密碼強度。如果密碼不符合政策，就將 finalUserStatus 設定為 RESET_REQUIRED。

messageAction

您可以將此參數設定為 SUPPRESS 以拒絕傳送 Amazon Cognito 通常會向新使用者傳送的歡迎訊息。如果您的函數未傳回此參數，Amazon Cognito 會傳送歡迎訊息。

desiredDeliveryMediums

您可以將此參數設定為 EMAIL 以透過電子郵件傳送歡迎訊息，或是設為 SMS 以透過簡訊傳送歡迎訊息。如果您的函數未傳回此參數，Amazon Cognito 會透過簡訊傳送歡迎訊息。

forceAliasCreation

如果您將此參數設定為 TRUE，而 UserAttributes 參數中的電話號碼或電子郵件地址已存在做為不同使用者的別名，則 API 呼叫會將別名從之前的使用者遷移到新建立的使用者。先前的使用者將無法再使用該別名登入。

如果您將此參數設定為 FALSE，而且別名已存在，Amazon Cognito 不會遷移使用者，並且會將錯誤傳回用户端應用程式。

如果您未傳回此參數，Amazon Cognito 會假定其值為「false」。

enableSMSMFA

將此參數設定為 true，以要求遷移的使用者完成 SMS 文字訊息多重要素驗證 (MFA) 才能登入。您的使用者集區必須啟用 MFA。請求參數中的用戶屬性必須包含電話號碼，否則該用戶的遷移將失敗。

範例：搭配現有密碼來遷移使用者

此範例 Lambda 函數會使用現有密碼遷移使用者，並避免歡迎訊息從 Amazon Cognito 中出現。

Node.js

```
const validUsers = {
  belladonna: { password: "Test123", emailAddress: "bella@example.com" },
};

// Replace this mock with a call to a real authentication service.
const authenticateUser = (username, password) => {
  if (validUsers[username] && validUsers[username].password === password) {
    return validUsers[username];
  } else {
    return null;
  }
};

const lookupUser = (username) => {
  const user = validUsers[username];

  if (user) {
    return { emailAddress: user.emailAddress };
  } else {
    return null;
  }
};

const handler = async (event) => {
  if (event.triggerSource == "UserMigration_Authentication") {
    // Authenticate the user with your existing user directory service
    const user = authenticateUser(event.userName, event.request.password);
    if (user) {
      event.response.userAttributes = {
        email: user.emailAddress,
        email_verified: "true",
      };
      event.response.finalUserStatus = "CONFIRMED";
      event.response.messageAction = "SUPPRESS";
    }
  } else if (event.triggerSource == "UserMigration_ForgotPassword") {
    // Look up the user in your existing user directory service
    const user = lookupUser(event.userName);
```

```
    if (user) {
      event.response.userAttributes = {
        email: user.emailAddress,
        // Required to enable password-reset code to be sent to user
        email_verified: "true",
      };
      event.response.messageAction = "SUPPRESS";
    }
  }

  return event;
};

export { handler };
```

自訂訊息 Lambda 觸發程序

Amazon Cognito 會在傳送電子郵件或電話驗證訊息或多重要素驗證 (MFA) 碼之前，叫用此觸發程序。您可以使用自訂的訊息觸發程序來動態自訂訊息。您可以在原始的 Amazon Cognito 主控台的 Message Customizations (自訂訊息) 索引標籤中編輯靜態自訂訊息。

請求中包含 `codeParameter`。此字串作為 Amazon Cognito 提供給使用者的代碼的預留位置。請將 `codeParameter` 字串插入訊息內文中您要顯示驗證碼的位置。Amazon Cognito 收到此回應時，會以實際的驗證碼取代 `codeParameter` 字串。

> **Note**
>
> 具有 `CustomMessage_AdminCreateUser` 觸發來源的自訂訊息 Lambda 函數會傳回使用者名稱和驗證碼。由於管理員建立的使用者必須同時接收其使用者名稱和代碼，因此函數的回應必須同時包含 `request.usernameParameter` 和 `request.codeParameter`。

主題

- 自訂訊息 Lambda 觸發程序來源
- 自訂訊息 Lambda 觸發程序參數
- 用於註冊的自訂訊息範例
- 管理員建立使用者的自訂訊息範例

自訂訊息 Lambda 觸發程序來源

triggerSource 值	事件
CustomMessage_SignUp	自訂訊息 - 在註冊後傳送確認碼。
CustomMessage_AdminCreateUser	自訂訊息 - 傳送臨時密碼給新使用者。
CustomMessage_ResendCode	自訂訊息 - 重新傳送確認碼給現有的使用者。
CustomMessage_ForgotPassword	自訂訊息 - 傳送「忘記密碼」請求的確認碼。
CustomMessage_UpdateUserAttribute	自訂訊息 - 當使用者的電子郵件或電話號碼變更時，此觸發程序會自動傳送驗證碼給使用者。無法用於其他屬性。
CustomMessage_VerifyUserAttribute	自訂訊息 - 當使用者為新的電子郵件或電話號碼手動請求驗證碼時，此觸發程序會傳送驗證碼給使用者。
CustomMessage_Authentication	自訂訊息 - 在身分驗證期間傳送 MFA 代碼。

自訂訊息 Lambda 觸發程序參數

Amazon Cognito 傳遞至此 Lambda 函數的請求，是以下參數和 Amazon Cognito 新增至所有請求的常用參數之組合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        }
        "codeParameter": "####",
        "usernameParameter": "string",
        "clientMetadata": {
            "string": "string",
            . . .
```

```
        }
    },
    "response": {
        "smsMessage": "string",
        "emailMessage": "string",
        "emailSubject": "string"
    }
}
```

自訂訊息請求參數

userAttributes

代表使用者屬性的一或多組名稱/值。

codeParameter

可讓您在自訂訊息中用來做為驗證碼預留位置的字串。

usernameParameter

使用者名稱。Amazon Cognito 會在管理員建立的使用者所產生的請求中納入此參數。

clientMetadata

您可以做為 Lambda 函數的自訂輸入提供的一個或多個鍵值組，該函數是您用於自訂訊息觸發程序所指定。叫用自訂訊息函數的要求不包含傳入 ClientMetadata 參數AdminInitiateAuth和InitiateAuthAPI 作業中的資料。若要將此資料傳遞至 Lambda 函數，您可以在下列 API 動作中使用 ClientMetadata 參數：

- AdminResetUserPassword
- AdminRespondToAuthChallenge
- AdminUpdateUserAttributes
- ForgotPassword
- GetUserAttributeVerificationCode
- ResendConfirmationCode
- SignUp
- UpdateUserAttributes

自訂訊息回應參數

在回應中，指定自訂文字以用於傳遞給使用者的訊息中。如需 Amazon Cognito 套用至這些參數的字串限制，請參閱MessageTemplateType。

smsMessage

要傳送給使用者的自訂簡訊。必須包含您在請求中收到的 codeParameter 值。

emailMessage

要傳送給使用者的自訂電子郵件訊息。您可以在 emailMessage 參數中使用 HTML 格式化。必須包含在請求中收到的 codeParameter 值作為變數 {####}。只有在使用者集區的 EmailSendingAccount 屬性為 DEVELOPER 時，Amazon Cognito 才可以使用 emailMessage 參數。如果使用者集區的 EmailSendingAccount 屬性不是 DEVELOPER，並且傳回 emailMessage 參數，Amazon Cognito 會產生 400 錯誤代碼 com.amazonaws.cognito.identity.idp.model.InvalidLambdaResponseException。當您選擇使用 Amazon Simple Email Service (Amazon SES) 傳送電子郵件訊息時，使用者集區的 EmailSendingAccount 屬性為 DEVELOPER。否則，該值為 COGNITO_DEFAULT。

emailSubject

自訂訊息的主旨行。只有在使用者集區的 EmailSendingAccount屬性為時，才能使用emailSubject參數DEVELOPER。如果使用者集區的 EmailSendingAccount 屬性不是 DEVELOPER，並且 Amazon Cognito 傳回 emailSubject 參數，Amazon Cognito 會產生 400 錯誤代碼 com.amazonaws.cognito.identity.idp.model.InvalidLambdaResponseException。當您選擇使用 Amazon Simple Email Service (Amazon SES) 傳送電子郵件訊息時，使用者集區的 EmailSendingAccount 屬性為 DEVELOPER。否則，該值為 COGNITO_DEFAULT。

用於註冊的自訂訊息範例

此 Lambda 函數可讓您在服務要求應用程式傳送驗證碼給使用者時，自訂電子郵件或簡訊的訊息。

Amazon Cognito 可在多個事件中叫用 Lambda 觸發程序：註冊後、重新傳送驗證碼、恢復忘記的密碼，或驗證使用者屬性。回應同時包含簡訊和電子郵件的訊息。訊息必須包含代碼參數 "####"。此參數是使用者接收驗證碼的預留位置。

一封電子郵件的長度上限為 20,000 個 UTF-8 字元，此長度包括驗證碼。您可以在這些電子郵件訊息中使用 HTML 標籤。

簡訊的長度上限為 140 個 UTF-8 字元。此長度包括驗證碼。

Node.js

```
const handler = async (event) => {
  if (event.triggerSource === "CustomMessage_SignUp") {
    const message = `Thank you for signing up. Your confirmation code is
 ${event.request.codeParameter}.`;
    event.response.smsMessage = message;
    event.response.emailMessage = message;
    event.response.emailSubject = "Welcome to the service.";
  }
  return event;
};

export { handler };
```

Amazon Cognito 會將事件資訊傳遞至您的 Lambda 函數。此函數會將相同事件物件傳回 Amazon Cognito，並在回應中附上任何變更。在 Lambda 主控台中，您可使用與 Lambda 觸發程序相關聯的資料來設定測試事件。下列是此程式碼範例的測試事件：

JSON

```
{
  "version": 1,
  "triggerSource": "CustomMessage_SignUp/CustomMessage_ResendCode/
CustomMessage_ForgotPassword/CustomMessage_VerifyUserAttribute",
  "region": "<region>",
  "userPoolId": "<userPoolId>",
  "userName": "<userName>",
  "callerContext": {
      "awsSdk": "<calling aws sdk with version>",
      "clientId": "<apps client id>",
      ...
  },
  "request": {
      "userAttributes": {
          "phone_number_verified": false,
          "email_verified": true,
           ...
      },
      "codeParameter": "####"
```

```
  },
  "response": {
      "smsMessage": "<custom message to be sent in the message with code parameter>"
      "emailMessage": "<custom message to be sent in the message with code
 parameter>"
      "emailSubject": "<custom email subject>"
  }
}
```

管理員建立使用者的自訂訊息範例

Amazon Cognito 傳送至此範例自訂訊息 Lambda 函數的請求的`triggerSource`值為，`CustomMessage_AdminCreateUser`且使用者名稱和臨時密碼。該函數`${event.request.codeParameter}`從請求中的臨時密碼以及請求`${event.request.usernameParameter}`中的用戶名填充。

您的自訂訊息必須在回應物件`emailMessage`中插`usernameParameter`入`smsMessage`和的`codeParameter`值。在此範例中，函數會將相同的訊息寫入回應欄位`event.response.smsMessage`和`event.response.emailMessage`。

一封電子郵件的長度上限為 20,000 個 UTF-8 字元。此長度包括驗證碼。您可以在這些電子郵件中使用 HTML 標籤。簡訊的長度上限為 140 個 UTF-8 字元。此長度包括驗證碼。

回應同時包含簡訊和電子郵件的訊息。

Node.js

```
const handler = async (event) => {
  if (event.triggerSource === "CustomMessage_AdminCreateUser") {
    const message = `Welcome to the service. Your user name is
 ${event.request.usernameParameter}. Your temporary password is
 ${event.request.codeParameter}`;
    event.response.smsMessage = message;
    event.response.emailMessage = message;
    event.response.emailSubject = "Welcome to the service";
  }
  return event;
};

export { handler }
```

Amazon Cognito 會將事件資訊傳遞至您的 Lambda 函數。此函數會將相同事件物件傳回 Amazon Cognito，並在回應中附上任何變更。在 Lambda 主控台中，您可使用與 Lambda 觸發程序相關聯的資料來設定測試事件。下列是此程式碼範例的測試事件：

JSON

```
{
  "version": 1,
  "triggerSource": "CustomMessage_AdminCreateUser",
  "region": "<region>",
  "userPoolId": "<userPoolId>",
  "userName": "<userName>",
  "callerContext": {
      "awsSdk": "<calling aws sdk with version>",
      "clientId": "<apps client id>",
      ...
  },
  "request": {
      "userAttributes": {
          "phone_number_verified": false,
          "email_verified": true,
           ...
      },
      "codeParameter": "####",
      "usernameParameter": "username"
  },
  "response": {
      "smsMessage": "<custom message to be sent in the message with code parameter
 and username parameter>"
      "emailMessage": "<custom message to be sent in the message with code parameter
 and username parameter>"
      "emailSubject": "<custom email subject>"
  }
}
```

自訂寄件者 Lambda 觸發程序

Amazon Cognito 使用者集區提供 Lambda 觸發程序 `CustomEmailSender` 和 `CustomSMSSender`，以啟用第三方電子郵件和簡訊通知。您可以選擇簡訊和電子郵件供應商，傳送通知給 Lambda 函數程式碼中的使用者。Amazon Cognito 需要傳送確認碼、驗證碼或臨時密碼等通知給使用者時，這些事件會啟用您設定的 Lambda 函數。Amazon Cognito 會將代碼和臨時密碼 (秘密) 傳送至您啟用的 Lambda

函數。Amazon Cognito 使用 AWS KMS 客户受管金鑰和 AWS Encryption SDK 加密這些祕密。AWS Encryption SDK 是用戶端加密程式庫，可協助您進行一般資料的加密和解密。

> **Note**
>
> 若要使用這些 Lambda 觸發程序設定使用者集區，您可以使用 AWS CLI 或 SDK。無法透過 Amazon Cognito 主控台使用這些組態。

CustomEmailSender

Amazon Cognito 會叫用此觸發程序，傳送電子郵件通知給使用者。

CustomSMSSender

Amazon Cognito 會叫用此觸發，傳送簡訊通知給使用者。

資源

下列資源可協助您使用 `CustomEmailSender` 和 `CustomSMSSender` 觸發程序。

AWS KMS

AWS KMS 是一種受管服務，用於建立和控制 AWS KMS 金鑰。這些金鑰可加密您的資料。如需詳細資訊，請參閱什麼是 AWS Key Management Service？。

KMS 金鑰

KMS 金鑰是密碼編譯金鑰的邏輯表示。KMS 金鑰包含金鑰 ID、建立日期、說明和金鑰狀態等中繼資料。KMS 金鑰也包含可用來加密和解密資料的金鑰材料。如需詳細資訊，請參閱刪除 AWS KMS 金鑰。

對稱 KMS 金鑰

對稱 KMS 金鑰是 256 位元的加密金鑰，不會在未加密的情況下退出 AWS KMS。若要使用對稱 KMS 金鑰，您必須呼叫 AWS KMS。Amazon Cognito 使用對稱金鑰。相同的金鑰可用於加密和解密。如需詳細資訊，請參閱對稱 KMS 金鑰。

自訂電子郵件寄件者 Lambda 觸發程序

若您將自訂電子郵件寄件者觸發程序指派至使用者集區，Amazon Cognito 會在使用者事件要求傳送電子郵件訊息時，調用 Lambda 函數而非其預設行為。您的 AWS Lambda 函數可以使用自訂寄件者觸發

程序，透過您選擇的方法或提供者傳送電子郵件通知給使用者。您函數的自訂程式碼必須從您的使用者集區處理並傳遞所有電子郵件訊息。

> **Note**
>
> 目前，您無法在 Amazon Cognito 主控台中指派自訂寄件者觸發程序。您可以在 `CreateUserPool` 或 `UpdateUserPool` API 請求中透過 `LambdaConfig` 參數指派觸發程序。

若要設定此觸發程序，請執行下列步驟：

1. 在 AWS Key Management Service (AWS KMS) 中建立對稱加密金鑰。Amazon Cognito 會產生密碼 (臨時密碼、驗證碼及確認碼)，然後使用此 KMS 金鑰將密碼加密。接著您可以在 Lambda 函數中使用 Decrypt API 來解密這些密碼，並以純文字傳送給使用者。對於您函數中的 AWS KMS 操作來說，AWS Encryption SDK 是實用的工具。
2. 建立您要指派為自訂寄件者觸發程序的 Lambda 函數。對 Lambda 函數角色授予您 KMS 金鑰的 `kms:Decrypt` 許可。
3. 授予 Amazon Cognito 服務委託人 `cognito-idp.amazonaws.com` 存取權，以叫用 Lambda 函數。
4. 撰寫 Lambda 函數代碼，以將您的訊息引導至自訂傳遞方法或第三方供應商。為了傳遞您的使用者驗證碼或確認碼，Base64 會解碼並解密請求中 `code` 參數的值。此操作會產生純文字代碼或密碼，且您必須將它加入訊息中。
5. 更新使用者集區，使其使用自訂寄件人 Lambda 觸發程序。使用自訂寄件者觸發程序更新或建立使用者集區的 IAM 主體必須具有建立 KMS 金鑰授予的許可。以下 `LambdaConfig` 程式碼片段會指派自訂簡訊和電子郵件寄件者函數。

```
"LambdaConfig": {
   "KMSKeyID": "arn:aws:kms:us-
east-1:123456789012:key/a6c4f8e2-0c45-47db-925f-87854bc9e357",
   "CustomEmailSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   },
   "CustomSMSSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   }
```

自訂電子郵件寄件者 Lambda 觸發程序參數

Amazon Cognito 傳遞至此 Lambda 函數的請求，是以下參數和 Amazon Cognito 新增至所有請求的常用參數之組合。

JSON

```
{
    "request": {
        "type": "customEmailSenderRequestV1",
        "code": "string",
        "clientMetadata": {
            "string": "string",
             . . .
            },
        "userAttributes": {
            "string": "string",
            . . .
         }
}
```

自訂電子郵件寄件者請求參數

類型

請求版本。對於自訂電子郵件寄件者事件，此字串的值一律為 `customEmailSenderRequestV1`。

code

您的函數可以解密並傳送給您使用者的加密代碼。

clientMetadata

您可以做為自訂電子郵件寄件者 Lambda 函數觸發程序的自訂輸入提供的一個或多個鍵值組。若要將此資料傳遞至您的 Lambda 函數，您可以使用 AdminRespondToAuthChallenge 和 RespondToAuthChallenge API 動作中的 ClientMetadata 參數。Amazon Cognito 不包含其傳遞至身分驗證後函數的請求中的 AdminInitiateAuth 和 InitiateAuth API 操作的 ClientMetadata 參數中的資料。

userAttributes

代表使用者屬性的一個或多個鍵值組。

自訂電子郵件寄件者回應參數

Amazon Cognito 不預期會在自訂電子郵件寄件者回應中收到任何其他傳回的資訊。您的函數可使用 API 操作來查詢和修改您的資源，或將事件中繼資料記錄到外部系統。

啟用自訂電子郵件寄件者 Lambda 觸發程序

若要設定自訂電子郵件寄件者觸發程序以使用自訂邏輯為使用者集區傳送電子郵件訊息，請啟用該觸發程序，如下所示。接下來的程序會將自訂電子郵件觸發器、自訂 SMS 觸發器或兩者指派給您的使用者集區。如果原本是使用 Amazon Simple Email Service 傳送電子郵件訊息，在您新增自訂電子郵件寄件者觸發程序之後，Amazon Cognito 一律會將使用者屬性 (包括電子郵件地址和一次性代碼) 傳送至 Lambda 函數。

> **Important**
>
> Amazon Cognito 會在使用者臨時密碼中逸出 HTML 預留字元，例如 `<` (<) 和 `>` (>)。這些字元可能會出現在 Amazon Cognito 傳送至您自訂電子郵件寄件者功能的臨時密碼中，但不會出現在臨時驗證碼中。若要傳送臨時密碼，您的 Lambda 函數必須在解密密碼後以及將訊息傳送給使用者之前取消逸出這些字元。

1. 在 AWS KMS 中建立加密金鑰。此金鑰可加密 Amazon Cognito 產生的臨時密碼和授權碼。您就可在自訂寄件者 Lambda 函數中解密這些秘密，並以純文字傳送給您的使用者。
2. 授予 Amazon Cognito 服務主體 `cognito-idp.amazonaws.com` 存取權限，以使用 KMS 金鑰加密代碼。

   將下列的資源型政策套用至您的 KMS 金鑰。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cognito-idp.amazonaws.com"
        },
        "Action": "kms:CreateGrant",
        "Resource": "arn:aws:kms:us-
west-2:111222333444:key/1example-2222-3333-4444-999example",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "111222333444"
```

```
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:cognito-idp:us-
west-2:111222333444:userpool/us-east-1_EXAMPLE"
            }
        }
    }]
}
```

3. 為自訂寄件者觸發程序建立 Lambda 函數。Amazon Cognito 使用 AWS 加密 SDK 來加密授權使用者 API 請求的秘密、臨時密碼和授權碼。

   - 將 IAM 角色指派到至少具有 KMS 金鑰 `kms:Decrypt` 許可的 Lambda 函數。

4. 授予 Amazon Cognito 服務委託人 `cognito-idp.amazonaws.com` 存取權，以叫用 Lambda 函數。

   下列 AWS CLI 命令可授予 Amazon Cognito 叫用 Lambda 函數的許可：

   ```
   aws lambda add-permission --function-name lambda_arn --statement-id
    "CognitoLambdaInvokeAccess" --action lambda:InvokeFunction --principal cognito-
   idp.amazonaws.com
   ```

5. 撰寫 Lambda 函數程式碼以傳送訊息。Amazon Cognito 使用 AWS Encryption SDK 來加密秘密，接著 Amazon Cognito 會將秘密傳送至自訂寄件者 Lambda 函數。在您的函數中，解密秘密並處理任何相關的中繼資料。然後將代碼、您自己的自訂訊息和目的地電話號碼傳送到遞送訊息的自訂 API。

6. 將 AWS Encryption SDK 新增到您的 Lambda 函數中。如需詳細資訊，請參閱 AWS 加密軟體 SDK 程式設計語言。若要更新 Lambda 套件，請完成下列步驟。

   a. 將 Lambda 函數在 AWS Management Console 匯出為 .zip 檔案。

   b. 打開您的函數並加入 AWS Encryption SDK。如需詳細資訊和下載連結，請參閱 AWS Encryption SDK 開發人員指南中的 AWS Encryption SDK 程式設計語言。

   c. 使用 SDK 相依性壓縮您的函數，然後將函數上傳至 Lambda。如需詳細資訊，請參閱 AWS Lambda 開發人員指南中的以 .zip 封存檔形式部署 Lambda 函數。

7. 更新使用者集區以新增自訂寄件者 Lambda 觸發程序。在 `UpdateUserPool` API 請求中包含 `CustomSMSSender` 或 `CustomEmailSender` 參數。`UpdateUserPool` API 操作需要您使用者集區的所有參數以及您要變更的參數。如果您未提供所有相關的參數，Amazon Cognito 會將任何

遺漏參數的值設定為其預設值。如下列範例所示，包含您要新增至或保留在使用者集區中的所有 Lambda 函數項目。如需更多詳細資訊，請參閱 更新使用者集區組態。

```
   #Send this parameter in an 'aws cognito-idp update-user-pool' CLI command,
including any existing
   #user pool configurations.

     --lambda-config "PreSignUp=lambda-arn, \
                      CustomSMSSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
                      CustomEmailSender={LambdaVersion=V1_0,LambdaArn=lambda-arn},
\
                      KMSKeyID=key-id"
```

若要使用 update-user-pool AWS CLI 移除自訂寄件者 Lambda 觸發條件，請忽略 --lambda-config 中的 CustomSMSSender 或 CustomEmailSender 參數，並包含您想要與使用者集區搭配使用的所有其他觸發條件。

若要使用 UpdateUserPool API 請求移除自訂寄件者 Lambda 觸發條件，請從包含其他使用者集區組態的請求內文省略 CustomSMSSender 或 CustomEmailSender 參數。

程式碼範例

下列 Node.js 範例處理您的自訂電子郵件寄件者 Lambda 函數中的電子郵件訊息事件。此範例假設您的函數定義了兩個環境變數。

KEY_ALIAS

您想要用來加密和解密使用者程式碼的 KMS 金鑰的別名。

KEY_ARN

您想要用來加密和解密使用者程式碼的 KMS 金鑰的 Amazon Resource Name (ARN)。

```
const AWS = require('aws-sdk');
const b64 = require('base64-js');
const encryptionSdk = require('@aws-crypto/client-node');
//Configure the encryption SDK client with the KMS key from the environment variables.
const { encrypt, decrypt } =
 encryptionSdk.buildClient(encryptionSdk.CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT);
```

```
const generatorKeyId = process.env.KEY_ALIAS;
const keyIds = [ process.env.KEY_ARN ];
const keyring = new encryptionSdk.KmsKeyringNode({ generatorKeyId, keyIds })
exports.handler = async (event) => {
 //Decrypt the secret code using encryption SDK.
 let plainTextCode;
 if(event.request.code){
  const { plaintext, messageHeader } = await decrypt(keyring,
 b64.toByteArray(event.request.code));
  plainTextCode = plaintext
 }
 //PlainTextCode now contains the decrypted secret.
 if(event.triggerSource == 'CustomEmailSender_SignUp'){
  //Send an email message to your user via a custom provider.
  //Include the temporary password in the message.
 }
 else if(event.triggerSource == 'CustomEmailSender_ResendCode'){
 }
 else if(event.triggerSource == 'CustomEmailSender_ForgotPassword'){
 }
 else if(event.triggerSource == 'CustomEmailSender_UpdateUserAttribute'){
 }
 else if(event.triggerSource == 'CustomEmailSender_VerifyUserAttribute'){
 }
 else if(event.triggerSource == 'CustomEmailSender_AdminCreateUser'){
 }
 else if(event.triggerSource == 'CustomEmailSender_AccountTakeOverNotification'){
 }
 return;
};
```

自訂電子郵件寄件者 Lambda 觸發程序來源

下表說明 Lambda 程式碼中自訂電子郵件觸發程序來源的觸發事件。

TriggerSource value	事件
CustomEmailSender_SignUp	使用者註冊後，Amazon Cognito 會傳送歡迎訊息。
CustomEmailSender_ForgotPassword	使用者請求代碼以重置其密碼。

TriggerSource value	事件
CustomEmailSender_ResendCode	使用者請求替換代碼以重置其密碼。
CustomEmailSender_UpdateUserAttribute	使用者更新電子郵件地址或電話號碼屬性，Amazon Cognito 會傳送代碼來驗證屬性。
CustomEmailSender_VerifyUserAttribute	使用者建立新的電子郵件地址或電話號碼屬性，Amazon Cognito 會傳送代碼來驗證屬性。
CustomEmailSender_AdminCreateUser	您可以在使用者集區中建立新使用者，Amazon Cognito 會將臨時密碼傳送給使用者。
CustomEmailSender_AccountTakeOverNotification	Amazon Cognito 偵測到有人嘗試接管使用者帳戶，並向使用者傳送通知。

自訂 SMS 寄件者 Lambda 觸發程序

若您將自訂簡訊寄件者觸發程序指派至使用者集區，Amazon Cognito 會在使用者事件要求傳送簡訊時，調用 Lambda 函數而非其預設行為。使用自定義發件人觸發器，您的 AWS Lambda 功能可以通過您選擇的方法和提供商向用戶發送 SMS 通知。您函數的自訂程式碼必須從您的使用者集區處理並傳遞所有簡訊。

> **Note**
>
> 目前，您無法在 Amazon Cognito 主控台中指派自訂寄件者觸發程序。您可以在 `CreateUserPool` 或 `UpdateUserPool` API 請求中透過 `LambdaConfig` 參數指派觸發程序。

若要設定此觸發程序，請執行下列步驟：

1. 在 AWS Key Management Service (AWS KMS) 中建立對稱加密金鑰。Amazon Cognito 會產生密碼 (臨時密碼、驗證碼及確認碼)，然後使用此 KMS 金鑰將密碼加密。接著您可以在 Lambda 函數中使用 Decrypt API 來解密這些密碼，並以純文字傳送給使用者。對於函數中的 AWS KMS 操作而言，AWS Encryption SDK這是一個有用的工具。
2. 建立您要指派為自訂寄件者觸發程序的 Lambda 函數。對 Lambda 函數角色授予您 KMS 金鑰的 `kms:Decrypt` 許可。

3. 授予 Amazon Cognito 服務委託人 `cognito-idp.amazonaws.com` 存取權，以叫用 Lambda 函數。
4. 撰寫 Lambda 函數代碼，以將您的訊息引導至自訂傳遞方法或第三方供應商。為了傳遞您的使用者驗證碼或確認碼，Base64 會解碼並解密請求中 `code` 參數的值。此操作會產生純文字代碼或密碼，且您必須將它加入訊息中。
5. 更新使用者集區，使其使用自訂寄件人 Lambda 觸發程序。使用自訂寄件者觸發程序更新或建立使用者集區的 IAM 主體必須具有建立 KMS 金鑰授予的許可。以下 `LambdaConfig` 程式碼片段會指派自訂簡訊和電子郵件寄件者函數。

```
"LambdaConfig": {
   "KMSKeyID": "arn:aws:kms:us-
east-1:123456789012:key/a6c4f8e2-0c45-47db-925f-87854bc9e357",
   "CustomEmailSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   },
   "CustomSMSSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   }
```

自訂 SMS 寄件者 Lambda 觸發程序參數

Amazon Cognito 傳遞至此 Lambda 函數的請求，是以下參數和 Amazon Cognito 新增至所有請求的常用參數之組合。

JSON

```
{
    "request": {
        "type": "customSMSSenderRequestV1",
        "code": "string",
        "clientMetadata": {
            "string": "string",
             . . .
            },
        "userAttributes": {
            "string": "string",
            . . .
         }
```

```
}
```

自訂 SMS 寄件者請求參數

type

請求版本。對於自訂 SMS 寄件者事件，此字串的值一律為 `customSMSSenderRequestV1`。

code

您的函數可以解密並傳送給您使用者的加密代碼。

clientMetadata

您可以做為自訂 SMS 寄件者 Lambda 函數觸發程序的自訂輸入提供的一個或多個鍵值組。若要將此資料傳遞至 Lambda 函數，您可以在AdminRespondToAuthChallenge和RespondToAuthChallengeAPI 動作中使用 ClientMetadata 參數。Amazon Cognito 不會在傳遞給身份驗證後功能的請求中包含來自 ClientMetadata 參數AdminInitiateAuth和 InitiateAuthAPI 操作的資料。

userAttributes

代表使用者屬性的一個或多個鍵值組。

自訂 SMS 寄件者回應參數

Amazon Cognito 不預期會在回應中收到任何其他傳回的資訊。您的函數可使用 API 操作來查詢和修改您的資源，或將事件中繼資料記錄到外部系統。

啟用自訂 SMS 寄件者 Lambda 觸發程序

您可以設定自訂 SMS 寄件者觸發程序，以使用自訂邏輯為使用者集區傳送 SMS 訊息。接下來的程序會將自訂 SMS 觸發器、自訂電子郵件觸發器或兩者指派給您的使用者集區。新增自訂 SMS 寄件者觸發程序後，Amazon Cognito 一律會傳送使用者屬性 (包括電話號碼和單次代碼) 到 Lambda 函數，而不是使用 Amazon Simple Notification Service 傳送 SMS 訊息的預設行為。

> **Important**
>
> Amazon Cognito 會在使用者臨時密碼中逸出 HTML 預留字元，例如 `<` (<) 和 `>` (>)。這些字元可能會出現在 Amazon Cognito 傳送至您自訂電子郵件寄件者功能的臨時密碼中，但

不會出現在臨時驗證碼中。若要傳送臨時密碼，您的 Lambda 函數必須在解密密碼後以及將訊息傳送給使用者之前取消逸出這些字元。

1. 在 AWS KMS中建立加密金鑰。此金鑰可加密 Amazon Cognito 產生的臨時密碼和授權碼。您就可在自訂寄件者 Lambda 函數中解密這些秘密，並以純文字傳送給您的使用者。
2. 授予 Amazon Cognito 服務主體 `cognito-idp.amazonaws.com` 存取權限，以使用 KMS 金鑰加密代碼。

   將下列的資源型政策套用至您的 KMS 金鑰。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cognito-idp.amazonaws.com"
        },
        "Action": "kms:CreateGrant",
        "Resource": "arn:aws:kms:us-
west-2:111222333444:key/1example-2222-3333-4444-999example",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "111222333444"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:cognito-idp:us-
west-2:111222333444:userpool/us-east-1_EXAMPLE"
            }
        }
    }]
}
```

3. 為自訂寄件者觸發程序建立 Lambda 函數。Amazon Cognito 使用 AWS 加密 SDK 來加密授權使用者 API 請求的秘密、臨時密碼和授權碼。

   - 將 IAM 角色指派到至少具有 KMS 金鑰 `kms:Decrypt` 許可的 Lambda 函數。

4. 授予 Amazon Cognito 服務委託人 `cognito-idp.amazonaws.com` 存取權，以叫用 Lambda 函數。

   以下 AWS CLI 命令授予 Amazon Cognito 可以叫用您的 Lambda 函數：

```
       aws lambda add-permission --function-name lambda_arn --statement-id
 "CognitoLambdaInvokeAccess" --action lambda:InvokeFunction --principal cognito-
idp.amazonaws.com
```

5. 撰寫 Lambda 函數程式碼以傳送訊息。Amazon Cognito 會在 Amazon Cogni AWS Encryption SDK to 將密碼傳送至自訂寄件者 Lambda 函數之前，用來加密機密。在您的函數中，解密秘密並處理任何相關的中繼資料。然後將代碼、您自己的自訂訊息和目的地電話號碼傳送到遞送訊息的自訂 API。
6. 新增 AWS Encryption SDK 至您的 Lambda 函數。如需詳細資訊，請參閱 AWS 加密軟體 SDK 程式設計語言。若要更新 Lambda 套件，請完成下列步驟。

   a. 將 Lambda 函數在 AWS Management Console匯出為 .zip 檔案。

   b. 打開您的功能並添加 AWS Encryption SDK. 如需詳細資訊和下載連結，請參閱 AWS Encryption SDK 開發人員指南中的 AWS Encryption SDK 程式設計語言。

   c. 使用 SDK 相依性壓縮您的函數，然後將函數上傳至 Lambda。如需詳細資訊，請參閱 AWS Lambda 開發人員指南中的以 .zip 封存檔形式部署 Lambda 函數。
7. 更新使用者集區以新增自訂寄件者 Lambda 觸發程序。在 `UpdateUserPool` API 請求中包含 `CustomSMSSender` 或 `CustomEmailSender` 參數。`UpdateUserPool` API 操作需要您使用者集區的所有參數以及您要變更的參數。如果您未提供所有相關的參數，Amazon Cognito 會將任何遺漏參數的值設定為其預設值。如下列範例所示，包含您要新增至或保留在使用者集區中的所有 Lambda 函數項目。如需詳細資訊，請參閱 更新使用者集區組態。

```
   #Send this parameter in an 'aws cognito-idp update-user-pool' CLI command,
including any existing
   #user pool configurations.

     --lambda-config "PreSignUp=lambda-arn, \
                      CustomSMSSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
                      CustomEmailSender={LambdaVersion=V1_0,LambdaArn=lambda-arn},
\
                      KMSKeyID=key-id"
```

若要使用移除自訂寄件者 Lambda 觸發程序 `update-user-pool` AWS CLI，請省略`CustomSMSSender`或`CustomEmailSender`參數`--lambda-config`，然後包含您要與使用者集區搭配使用的所有其他觸發程序。

若要使用 `UpdateUserPool` API 請求移除自訂寄件者 Lambda 觸發條件，請從包含其他使用者集區組態的請求內文省略 `CustomSMSSender` 或 `CustomEmailSender` 參數。

程式碼範例

下列 Node.js 範例處理您的自訂 SMS 寄件者 Lambda 函數中的 SMS 訊息事件。此範例假設您的函數定義了兩個環境變數。

KEY_ALIAS

您想要用來加密和解密使用者程式碼的 KMS 金鑰的別名。

KEY_ARN

您想要用來加密和解密使用者程式碼的 KMS 金鑰的 Amazon Resource Name (ARN)。

```
const AWS = require('aws-sdk');
const b64 = require('base64-js');
const encryptionSdk = require('@aws-crypto/client-node');
//Configure the encryption SDK client with the KMS key from the environment variables.

const { encrypt, decrypt } =
 encryptionSdk.buildClient(encryptionSdk.CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT);
const generatorKeyId = process.env.KEY_ALIAS;
const keyIds = [ process.env.KEY_ARN ];
const keyring = new encryptionSdk.KmsKeyringNode({ generatorKeyId, keyIds })
exports.handler = async (event) => {
 //Decrypt the secret code using encryption SDK.
 let plainTextCode;
 if(event.request.code){
  const { plaintext, messageHeader } = await decrypt(keyring,
 b64.toByteArray(event.request.code));
  plainTextCode = plaintext
 }
 //PlainTextCode now contains the decrypted secret.
 if(event.triggerSource == 'CustomSMSSender_SignUp'){
  //Send an SMS message to your user via a custom provider.
  //Include the temporary password in the message.
```

```
 }
 else if(event.triggerSource == 'CustomSMSSender_ResendCode'){
 }
 else if(event.triggerSource == 'CustomSMSSender_ForgotPassword'){
 }
 else if(event.triggerSource == 'CustomSMSSender_UpdateUserAttribute'){
 }
 else if(event.triggerSource == 'CustomSMSSender_VerifyUserAttribute'){
 }
 else if(event.triggerSource == 'CustomSMSSender_AdminCreateUser'){
 }
 else if(event.triggerSource == 'CustomSMSSender_AccountTakeOverNotification'){
 }
 return;
};
```

主題

- 使用自訂 SMS 寄件者函數評估 SMS 訊息功能
- 自訂簡訊寄件者 Lambda 觸發程序來源

使用自訂 SMS 寄件者函數評估 SMS 訊息功能

自訂 SMS 寄件者 Lambda 函數會接受您的使用者集區將傳送的 SMS 訊息，且該函數會根據您的自訂邏輯傳遞內容。Amazon Cognito 將 自訂 SMS 寄件者 Lambda 觸發程序參數 傳送到您的函數中。您的函數可以進行您想要使用此資訊做的事。例如，您可以將代碼傳送至 Amazon Simple Notification Service (Amazon SNS) 主題。Amazon SNS 主題訂閱者可以是 SMS 訊息、HTTPS 端點或電子郵件地址。

若要使用自訂簡訊傳送者 Lambda 函數建立 Amazon Cognito 簡訊的測試環境，請參閱上的 AWS 範例程式庫sms-redirected-to-email中的 amazon-cognito-user-pooldevelopment-and-testing-with-。GitHub存放庫包含可 AWS CloudFormation 建立新使用者集區或使用您已有的使用者集區的範本。這些範本會建立 Lambda 函數和 Amazon SNS 主題。由範本指派為自訂 SMS 寄件者觸發程序的 Lambda 函數，會將您的 SMS 訊息重新導向至 Amazon SNS 主題的訂閱者。

當您將此解決方案部署到使用者集區時，對於 Amazon Cognito 通常透過 SMS 簡訊傳送的所有訊息，Lambda 函數會改為傳送到一個中央電子郵件地址。使用此解決方案可以自訂和預覽 SMS 訊息，並測試導致 Amazon Cognito 傳送 SMS 訊息的使用者集區事件。完成測試後，回滾 CloudFormation 堆棧，或從用戶池中刪除自定義 SMS 發送者功能分配。

> **Important**
>
> 不要使用 amazon-cognito-user-pool-development-and-testing-with- 中的模板sms-redirected-to-email來構建生產環境。解決方案中的自訂 SMS 寄件者 Lambda 函數會模擬 SMS 訊息，但 Lambda 函數會將它們全部傳送到一個中央電子郵件地址。您必須先完成在 Amazon Cognito 使用者集區的簡訊設定 顯示的需求，才能在實際產品的 Amazon Cognito 使用者集區中傳送 SMS 訊息。

自訂簡訊寄件者 Lambda 觸發程序來源

下表說明 Lambda 程式碼中自訂簡訊觸發程序來源的觸發事件。

`TriggerSource value`	事件
`CustomSMSSender_SignUp`	使用者註冊後，Amazon Cognito 會傳送歡迎訊息。
`CustomSMSSender_ForgotPassword`	使用者請求代碼以重置其密碼。
`CustomSMSSender_ResendCode`	用戶請求一個新的代碼來確認其註冊。
`CustomSMSSender_VerifyUserAttribute`	使用者建立新的電子郵件地址或電話號碼屬性，Amazon Cognito 會傳送代碼來驗證屬性。
`CustomSMSSender_UpdateUserAttribute`	使用者更新電子郵件地址或電話號碼屬性，Amazon Cognito 會傳送代碼來驗證屬性。
`CustomSMSSender_Authentication`	使用 SMS 多重要素驗證 (MFA) 設定的使用者登入。
`CustomSMSSender_AdminCreateUser`	您可以在使用者集區中建立新使用者，Amazon Cognito 會將臨時密碼傳送給使用者。

使用 Amazon Pinpoint 分析搭配 Amazon Cognito 使用者集區

Amazon Cognito 使用者集區與 Amazon Pinpoint 整合，以提供 Amazon Cognito 使用者集區的分析，並且讓 Amazon Pinpoint 行銷活動的使用者資料更豐富。Amazon Pinpoint 提供分析和目標性的活

動，以透過推送通知來鼓勵使用者使用行動應用程式。透過 Amazon Cognito 使用者集區中的 Amazon Pinpoint 分析支援，可讓您在 Amazon Pinpoint 主控台中追蹤使用者集區註冊、登入、失敗的身分驗證、每日作用中使用者 (DAU) 和每月作用中使用者 (MAU)。您可以深入查看不同日期範圍或屬性的資料，例如裝置平台、裝置地區設定和應用程式版本。

您也可以為應用程式設定自訂屬性。然後您可以在 Amazon Pinpoint 中使用這些屬性來區隔使用者，並傳送目標性的推送通知給他們。如果您在 Amazon Cognito 主控台的 Analytics (分析) 索引標籤，選擇 Share user attribute data with Amazon Pinpoint (與 Amazon Pinpoint 共用使用者屬性資料)，將會為使用者電子郵件地址和電話號碼建立額外的端點。

當您在 Amazon Cognito 主控台的使用者集區中啟用 Amazon Pinpoint 分析時，您同時會建立服務連結角色，當 Amazon Cognito 向 Amazon Pinpoint 為您的使用者集區發出 API 請求時假設為該角色。新增分析組態的 IAM 主體必須具有 CreateServiceLinkedRole 權限。該服務連結角色名稱為 AWSServiceRoleForAmazonCognitoIdp。如需更多詳細資訊，請參閱 使用 Amazon Cognito 的服務連結角色。

當您在 Amazon Cognito API 中將 `AnalyticsConfiguration` 套用至應用程式用戶端時，您可以為 Amazon Pinpoint 指派自訂 IAM 角色，並指派外部 ID 來擔任該角色。該角色必須信任 `cognito-idp` 服務主體，而且如果角色信任原則需要外部 ID，則必須符合您的 `AnalyticsConfiguration`。您必須為 Amazon Pinpoint 專案授予角色權限和下列權限。

- `mobiletargeting:UpdateEndpoint`
- `mobiletargeting:PutEvents`

Amazon Cognito 和 Amazon Pinpoint 區域可用性

下表說明 Amazon Cognito 與 Amazon Pinpoint 的 AWS 區域 映射符合下列條件之一。

- 您只能使用位於美國東部 (維吉尼亞北部) (us-east-1) 區域的 Amazon Pinpoint 專案。
- 您可以使用位於同一區域或位於美國東部 (維吉尼亞北部) (us-east-1) 區域的 Amazon Pinpoint 專案

Amazon Cognito 預設只能將分析傳送至相同 AWS 區域 的 Amazon Pinpoint 專案。此規則的例外是下表中的區域，以及無法使用 Amazon Pinpoint 的區域。

下列區域無法使用 Amazon Pinpoint。這些區域中的 Amazon Cognito 使用者集區不支援分析。

- 歐洲 (米蘭)
- 中東 (巴林)

- 亞太區域 (大阪)
- 以色列 (特拉維夫)
- 非洲 (開普敦)
- 亞太區域 (雅加達)

表格顯示您建置 Amazon Cognito 使用者集區的區域與 Amazon Pinpoint 對應區域的關係。您必須在可用區域中設定 Amazon Pinpoint 專案，才能將其與 Amazon Cognito 整合。

Amazon Cognito 使用者集區區域	Amazon Pinpoint 專案區域
ap-northeast-1	us-east-1
ap-northeast-2	us-east-1
ap-south-1	us-east-1、ap-south-1
ap-southeast-1	us-east-1
ap-southeast-2	us-east-1、ap-southeast-2
ca-central-1	us-east-1
eu-central-1	us-est-1、eu-central-1
eu-west-1	us-est-1、eu-west-1
eu-west-2	us-east-1
us-east-1	us-east-1
us-east-2	us-east-1
us-west-2	us-est-1、us-west-2

區域映射範例

- 如果您在 ap-northeast-1 中建立使用者集區，可以在 us-east-1 中建立您的 Amazon Pinpoint 專案。
- 如果您在 ap-south-1 中建立使用者集區，可以在 us-east-1 或 ap-south-1 中建立您的 Amazon Pinpoint 專案。

> **Note**
>
> 對於除上表以外的所有 AWS 區域，Amazon Cognito 只能使用與您的使用者集區所在區域相同的 Amazon Pinpoint 專案。如果您建立使用者集區的所在區域無法使用 Amazon Pinpoint，並且沒有列在表中，則 Amazon Cognito 不會支援該區域中的 Amazon Pinpoint 分析。如需詳細的 AWS 區域 資訊，請參閱 Amazon Pinpoint 端點和配額。

指定 Amazon Pinpoint 分析設定 (AWS Management Console)

您可以設定 Amazon Cognito 使用者集區，將分析資料傳送到 Amazon Pinpoint。Amazon Cognito 只會針對本機使用者將分析資料傳送至 Amazon Pinpoint。將使用者集區設定為與 Amazon Pinpoint 專案關聯之後，您必須在 API 請求中加入 `AnalyticsMetadata`。如需更多詳細資訊，請參閱 將您的應用程式與 Amazon Pinpoint 整合。

指定分析設定

1. 前往 Amazon Cognito 主控台。您可能會收到提示，要求您輸入 AWS 憑證。
2. 選擇使用者集區，並從清單中選擇現有使用者集區。
3. 選擇 App integration (應用程式整合) 標籤。
4. 在 App clients and analytics (應用程式用戶端和分析) 下，從清單中選擇現有的 App client name (應用程式用戶端名稱)。
5. 在 Pinpoint analytics (Pinpoint 分析) 下，選擇 Enable (啟用)。
6. 選擇 Pinpoint Region (Pinpoint 區域)。
7. 選擇 Amazon Pinpoint project (Amazon Pinpoint 專案)，或選取 Create Amazon Pinpoint project (建立 Amazon Pinpoint 專案)。

> **Note**
>
> Amazon Pinpoint 專案 ID 是 Amazon Pinpoint 專案專屬的 32 字元字串。它會列示在 Amazon Pinpoint 主控台中。
> 您可以將多個 Amazon Cognito 應用程式對應至單一 Amazon Pinpoint 專案。不過，每個 Amazon Cognito 應用程式都只能對應至一個 Amazon Pinpoint 專案。
> 在 Amazon Pinpoint 中，每個專案都應該是單一應用程式。例如，遊戲開發人員有兩個遊戲，即使這兩個遊戲都使用相同的 Amazon Cognito 使用者集區，每個遊戲應該都還是個

別的 Amazon Pinpoint 專案。如需 Amazon Pinpoint 專案的詳細資訊，請參閱在 Amazon Pinpoint 中建立專案。

8. 如果您想要讓 Amazon Cognito 傳送電子郵件地址和電話號碼給 Amazon Pinpoint，以為使用者建立額外的端點，請在 User data sharing (使用者資料共用) 下選擇 Share user attribute data with Amazon Pinpoint (與 Amazon Pinpoint 共用使用者資料)。使用者的電子郵件地址和電話號碼經過驗證後，只有在使用者帳戶可以使用這些資訊的情況下，Amazon Cognito 才會與 Amazon Pinpoint 共用。

> **Note**
>
> 端點會唯一識別您可以透過 Amazon Pinpoint 來傳送推送通知的使用者裝置。如需端點的詳細資訊，請參閱 Amazon Pinpoint 開發人員指南中的新增端點。

9. 選擇 Save changes (儲存變更)。

指定 Amazon Pinpoint 分析設定 (AWS CLI 和 AWS API)

使用下列命令來為您的使用者集區指定 Amazon Pinpoint 分析設定。

在建立應用程式時，為使用者集區現有的用戶端應用程式指定分析設定

- AWS CLI: `aws cognito-idp create-user-pool-client`
- AWS API：CreateUserPoolClient

為使用者集區現有的用戶端應用程式更新分析設定

- AWS CLI: `aws cognito-idp update-user-pool-client`
- AWS API：UpdateUserPoolClient

> **Note**
>
> 在您使用 `ApplicationArn` 時，Amazon Cognito 支援區域內整合

將您的應用程式與 Amazon Pinpoint 整合

您可以在使用者集區 API 中為 Amazon Cognito 本機使用者將分析中繼資料發布到 Amazon Pinpoint。

本機使用者

註冊帳戶或在您的使用者集區中建立，而非透過第三方身分提供者 (IdP) 登入的使用者。

使用者集區 API

您可以與 AWS SDK 整合的操作，使用具有自訂使用者介面 (UI) 的應用程式。您無法為透過託管 UI 登入的聯合或本機使用者傳遞分析中繼資料。如需使用者集區 API 操作的清單，請參閱 Amazon Cognito API 參考。

將使用者集區設定為發佈到行銷活動後，Amazon Cognito 會將中繼資料傳遞至 Amazon Pinpoint，以進行下列 API 操作。

- `AdminInitiateAuth`
- `AdminRespondToAuthChallenge`
- `ConfirmForgotPassword`
- `ConfirmSignUp`
- `ForgotPassword`
- `InitiateAuth`
- `ResendConfirmationCode`
- `RespondToAuthChallenge`
- `SignUp`

若要將使用者工作階段的中繼資料傳遞至 Amazon Pinpoint 促銷活動，請在 API 請求的 `AnalyticsMetadata` 參數中包括 `AnalyticsEndpointId` 值。如需 JavaScript 範例，請參閱 AWS 知識中心中的為什麼我的 Amazon Cognito 使用者集區分析沒有出現在我的 Amazon Pinpoint 儀表板上？。

設定使用者集區分析

使用 Amazon Pinpoint 分析可讓您追蹤 Amazon Cognito 使用者集區註冊、登入、失敗的身分驗證、每日作用中使用者 (DAU) 和每月作用中使用者 (MAU)。您也可以使用 AWS Mobile SDK for Android

或 AWS Mobile SDK for iOS，為您的應用程式設定專屬的使用者屬性。然後您可以在 Amazon Pinpoint 中使用這些屬性來區隔使用者，並傳送目標性的推送通知給他們。

在 應用程式用戶端和分析 下的 應用程式整合 索引標籤中，您可以導覽至現有的應用程式用戶端，或建立新用戶端。在應用程式用戶端組態中，您可以指定要與應用程式搭配使用的 Amazon Pinpoint 專案。如需詳細資訊，請參閱使用 Amazon Pinpoint 分析搭配 Amazon Cognito 使用者集區。

> **Note**
>
> Amazon Pinpoint 在北美、歐洲、亞洲和大洋洲的多個 AWS 區域皆可使用。 Amazon Pinpoint 區域包括 Amazon Pinpoint API。如果 Amazon Cognito 支援 Amazon Pinpoint 區域，Amazon Cognito 會將事件傳送到相同 Amazon Pinpoint 區域內的 Amazon Pinpoint 專案。如果 Amazon Pinpoint 不支援該區域，Amazon Cognito 僅支援在 us-east-1 中傳送事件。如需 Amazon Pinpoint 的詳細區域資訊，請參閱 Amazon Pinpoint 端點和配額及使用 Amazon Pinpoint 分析搭配 Amazon Cognito 使用者集區。

新增分析和行銷活動

1. 選擇 Add analytics and campaigns (新增分析和行銷活動)。
2. 從清單中選擇 Cognito app client (Cognito 應用程式用戶端)。
3. 若要將您的 Amazon Cognito 應用程式對應到 Amazon Pinpoint 專案，請從清單中選擇 Amazon Pinpoint 專案。

   > **Note**
   >
   > Amazon Pinpoint 專案 ID 是 Amazon Pinpoint 專案專屬的 32 字元字串。它會列示在 Amazon Pinpoint 主控台中。
   > 您可以將多個 Amazon Cognito 應用程式對應至單一 Amazon Pinpoint 專案。不過，每個 Amazon Cognito 應用程式都只能對應至一個 Amazon Pinpoint 專案。
   > 在 Amazon Pinpoint 中，每個專案都應該是單一應用程式。例如，遊戲開發人員有兩個遊戲，即使這兩個遊戲都使用相同的 Amazon Cognito 使用者集區，每個遊戲應該都還是個別的 Amazon Pinpoint 專案。

4. 如果您想要讓 Amazon Cognito 傳送電子郵件地址和電話號碼給 Amazon Pinpoint，以為使用者建立額外的端點，請選擇 Share user attribute data with Amazon Pinpoint (與 Amazon Pinpoint 共用使用者屬性資料)。

> **Note**
>
> 端點會唯一識別您可以透過 Amazon Pinpoint 來傳送推送通知的使用者裝置。如需端點的詳細資訊，請參閱 Amazon Pinpoint 開發人員指南中的新增端點至 Amazon Pinpoint。

5. 輸入您已經建立的 IAM role (IAM 角色)，或在 IAM 主控台中選擇 Create new role (建立新角色)，以建立新角色。
6. 選擇 Save changes (儲存變更)。
7. 若要指定其他應用程式對應，請選擇 Add app mapping (新增應用程式對應)。
8. 選擇 Save changes (儲存變更)。

管理使用者集區中的使用者

在您建立使用者集區之後，您可以建立、確認和管理使用者帳戶。您可以使用 Amazon Cognito 使用者集區群組將 IAM 角色映射至群組，以管理您的使用者及其資源存取權限。

利用使用者遷移 Lambda 觸發，將您的使用者匯入至使用者集區。這種方式可在使用者首次登入您的使用者集區時，將使用者從您現有的使用者目錄無縫遷移至使用者集區。

主題

- 設定建立使用者的原則
- 註冊及確認使用者帳戶
- 建立使用者帳戶為管理員
- 新增群組至使用者集區
- 管理及搜尋使用者帳戶
- 復原使用者帳戶
- 將使用者匯入使用者集區
- 使用者集區屬性
- 新增使用者集區密碼要求

設定建立使用者的原則

您的使用者集區可以允許使用者註冊，或也可將其建立為管理員。您還可以控制註冊後的驗證和確認過程有多大程度掌握在用戶手中。例如，您可能想要根據外部驗證程序查看並接受註冊。此組態或管理員建立使用者政策，也定下了使用者將無法再確認自身帳戶前的時限。

Amazon Cognito 可以滿足您大眾客戶的需求，做為軟體的客戶身分和存取管理 (CIAM) 平台。針對任何知道您公開應用程式客戶端 ID 並請求註冊的網路用戶，接受註冊並具有應用程式客戶端 (無論是否具有託管 UI) 的使用者集區可為其創建使用者設定檔。註冊的使用者設定檔可以接收存取和身份權杖，並可存取您授權給應用程式的資源。在使用者集區中啟用註冊之前，請先檢閱您的選項，並確保組態符合安全性標準。請謹慎設定啟用自助註冊與 `AllowAdminCreateUserOnly`，如下列流程所述。

AWS Management Console

使用者集區的註冊體驗索引標籤，以及建立使用者集區精靈的設定註冊體驗步驟，包含了一些在使用者集區中註冊和管理建立使用者的設定。

若要設定註冊體驗

1. 在 Cognito 輔助驗證和確認中，選擇是否允許 Cognito 自動傳送訊息以進行驗證和確認。啟用此設定後，Amazon Cognito 會傳送電子郵件或簡訊給新使用者，其中包含他們必須提供給您的使用者集區之代碼。這會確認他們的電子郵件地址或電話號碼所有權、將等效屬性設置為已驗證、並確認用戶帳戶以進行登入。您選擇的待驗證屬性會決定驗證訊息的傳遞方式和目的地。
2. 驗證屬性變更在建立使用者時並不重要，但與屬性驗證相關。如使用者已變更但尚未驗證登入屬性，您可以允許其繼續使用新屬性值或原始屬性值登入。如需更多詳細資訊，請參閱 在使用者變更電子郵件或電話號碼時進行驗證。
3. 必要屬性顯示必須先提供值的屬性，才能讓使用者註冊或讓您建立使用者。您只能在建立使用者集區精靈中設定必要的屬性。
4. 自訂屬性對使用者建立和註冊程序很重要，因為您只能在第一次建立使用者時為不可變的自訂屬性設定值。如需自訂屬性的詳細資訊，請參閱 自訂屬性。
5. 如果您希望使用者能夠使用未驗證的 `SignUp` API 產生新帳戶，請在自助註冊中選擇啟用自助註冊。如果停用自助註冊，就只能在 Amazon Cognito 主控台或使用 AdminCreateUser 要求以管理員身份建立新使用者。在自助註冊未啟用的使用者集區中，SignUp API 要求會傳回 `NotAuthorizedException`，且託管的 UI 不會顯示註冊連結。

針對您打算以管理員身分建立使用者的使用者集區，您可以在登入體驗索引標籤中的管理員設定的臨時密碼過期時間下，設定其暫時密碼的持續時間。

管理員建立用戶的另一個重要元素是邀請訊息。當您建立新使用者時，Amazon Cognito 會傳送訊息給他們，內含可讓他們進行首次登入的應用程式連結。在訊息範本下的訊息索引標籤中自訂此訊息範本。

您可以使用用戶端機密來設定機密應用程式用戶端 (通常是 Web 應用程式)，防止在沒有應用程式用戶端機密的情況下註冊。作為安全性最佳實務，請勿在公有應用程式用戶端 (通常是行動應用程式) 中散佈應用程式用戶端機密。您可以在 Amazon Cognito 主控台的應用程式整合索引標籤中建立包含用戶端機密的應用程式用戶。

Amazon Cognito user pools API

您可以透過程式設計方式設定參數，以便在 CreateUserPool 或 UpdateUserPool API 要求中的使用者集區建立使用者。

AdminCreateUserConfig 元素會為使用者集區的下列屬性設定值。

1. 啟用自助註冊
2. 您傳送給新管理員建立的使用者之邀請訊息

下列範例新增至完整 API 要求主體時，會設定具有未啟用自助註冊和基本邀請電子郵件的使用者集區。

```
"AdminCreateUserConfig": {
      "AllowAdminCreateUserOnly": true,
      "InviteMessageTemplate": {
         "EmailMessage": "Your username is {username} and temporary password is
 {####}.",
         "EmailSubject": "Welcome to ExampleApp",
         "SMSMessage": "Your username is {username} and temporary password is
 {####}."
      }
   }
```

CreateUserPool 或 UpdateUserPool API 要求的下列其他參數會管理新使用者的建立。

AutoVerifiedAttributes

您想要在註冊新使用者時自動傳送訊息的屬性、電子郵件或電話號碼。

政策

使用者集區密碼政策。

結構描述

使用者集區自訂屬性。它們對使用者建立和註冊程序很重要，因為您只能在第一次建立使用者時為不可變的自訂屬性設定值。

此參數也為使用者集區設定必要屬性。下列文字在插入完整 API 要求主體的 Schema 元素時，會視需要設定 email 屬性。

```
{
            "Name": "email",
            "Required": true
}
```

註冊及確認使用者帳戶

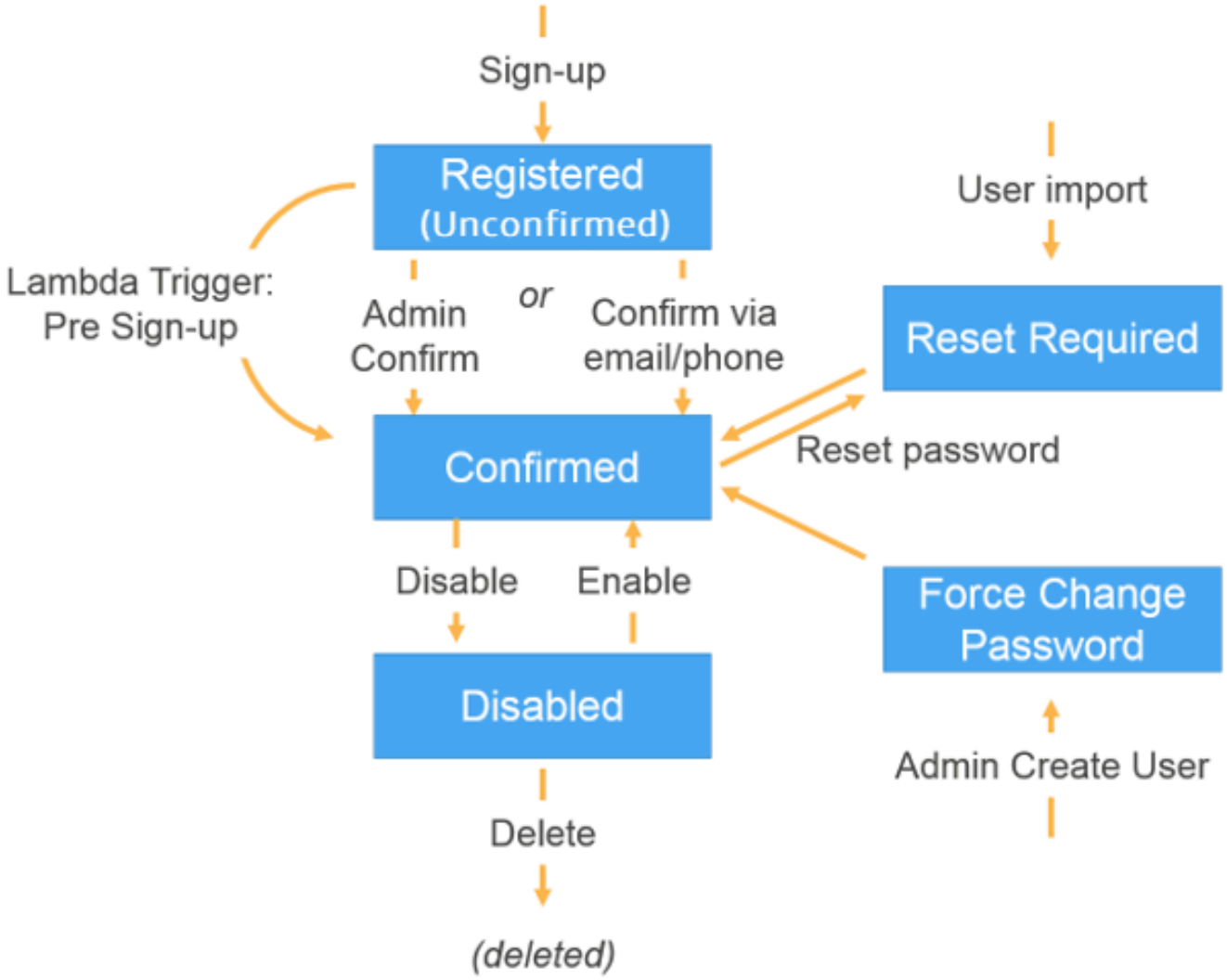
使用者帳戶會以下列其中一種方法新增到您的使用者集區：

- 使用者在使用者集區的用戶端應用程式中註冊。這可能是行動和 Web 應用程式。
- 您可以將使用者的帳戶匯入您的使用者集區中。如需詳細資訊，請參閱 從 CSV 檔案將使用者匯入使用者集區。
- 您可以在您的使用者集區中建立使用者的帳戶，並邀請使用者登入。如需詳細資訊，請參閱 建立使用者帳戶為管理員。

自行註冊的使用者必須經過確認才能登入。匯入和建立的使用者已經過確認，但必須在第一次登入時建立自己的密碼。以下各節將說明確認程序，以及電子郵件和電話驗證。

使用者帳戶確認概觀

下圖說明確認程序：



使用者帳戶可以是下列任何狀態：

已註冊 (未確認)

使用者已成功註冊，但要等到使用者帳戶經過確認後才能登入。使用者已啟用，但未確認在此狀態。

自行註冊的新使用者會從此狀態開始。

已確認

已確認使用者帳戶，且使用者可以登入。當使用者輸入代碼或按照電子郵件連結確認其使用者帳戶時，會自動驗證該電子郵件或電話號碼。確認碼或連結的有效時間為 24 小時。

如果管理員或註冊前 Lambda 觸發程序已確認過使用者帳戶，可能就不會有與帳戶相關聯的已驗證電子郵件或電話號碼。

需要重設密碼

已確認使用者帳戶，但使用者必須請求確認碼，並重設其密碼，才能登入。

由管理員或開發人員匯入的使用者帳戶會從此狀態開始。

強制變更密碼

已確認使用者帳戶，而使用者可以使用臨時密碼登入，但在第一次登入時，使用者必須先將其密碼變更為新的值，才能進行任何其他操作。

由管理員或開發人員建立的使用者帳戶會從此狀態開始。

已停用

您必須先停用該使用者的登入存取權限，才能刪除使用者帳戶。

註冊時驗證聯絡資訊

當新使用者註冊您的應用程式時，您可能希望其提供至少一種聯絡方式。例如，若您有了使用者的聯絡資訊，即可以：

- 在使用者選擇重設其密碼時傳送臨時密碼。
- 在使用者的個人資訊或財務資訊有所更新時通知使用者。
- 傳送促銷訊息，例如特殊優惠或折扣。
- 傳送帳戶摘要或帳單提醒。

對於以上所述的各種使用案例，您的訊息務必傳送至已驗證的目的地。否則，您的訊息將有可能因輸入錯誤而傳送至無效的電子郵件地址或電話號碼。或者更糟的是，您可能會將機密資訊傳送給假冒正當使用者的惡意人士。

為協助確保您的訊息只傳送給合宜的人士，請設定您的 Amazon Cognito 使用者集區，要求使用者註冊時必須提供以下項目：

a. 電子郵件地址或電話號碼。

b. 由 Amazon Cognito 傳送至該電子郵件地址或電話號碼的驗證碼。如果超過 24 小時，且使用者的程式碼或連結不再有效，請呼叫 ResendConfirmationCodeAPI 作業以產生並傳送新的代碼或連結。

透過提供驗證碼，使用者即證明其有權可存取收到該驗證碼的信箱或電話。使用者提供該代碼後，Amazon Cognito 將會更新您的使用者集區中相應使用者的資訊，方式如下：

- 將使用者的狀態設為 `CONFIRMED`。
- 更新使用者的屬性以表明驗證過其電子郵件地址或電話號碼。

若要檢視這項資訊，您可以使用 Amazon Cognito 主控台。或者，您可以在其中一個 AWS SDK 中使用 `AdminGetUser` API 作業 AWS CLI、`admin-get-user`指令搭配使用，或使用對應動作。

如果使用者已驗證過聯絡方式，Amazon Cognito 將在使用者請求重設密碼時自動傳送訊息給該使用者。

設定您的使用者集區以要求進行電子郵件或電話驗證

當您驗證您的使用者電子郵件地址和電話號碼後，便可確保您能夠與您的使用者聯絡。完成中的下列步驟， AWS Management Console 將您的使用者集區設定為要求您的使用者確認其電子郵件地址或電話號碼。

> **Note**
>
> 如果您的帳戶中還沒有使用者集區，請參閱 使用者集區入門。

設定使用者集區

1. 導覽至 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 從導覽窗格中，選擇 Users Pools (使用者集區)。從清單中選擇現有的使用者集區，或建立使用者集區。
3. 選擇 Sign-up experience (註冊體驗) 索引標籤，找到 Attribute verification and user account confirmation (屬性驗證與使用者帳戶確認)。選擇編輯。
4. 在 Cognito 輔助驗證和確認中，選擇是否允許 Cognito 自動傳送訊息以進行驗證和確認。啟用此設定後，Amazon Cognito 會將訊息傳送至使用者聯絡屬性，該屬性是您在使用者註冊或建立設定檔時選擇的。為了驗證屬性並確認使用者設定檔以進行登入，Amazon Cognito 會將代碼或連結以訊息傳送給使用者。然後，使用者必須在您的 UI 中輸入代碼，以便應用程式在 `ConfirmSignUp` 或 `AdminConfirmSignUp` API 請求中確認。

> **Note**
>
> 您也可以停用 Cognito-assisted verification and confirmation (Cognito 協助驗證和確認)，並使用已進行身分驗證的 API 動作或 Lambda 觸發程序來驗證屬性並確認使用者。
> 若您選擇此選項，Amazon Cognito 於使用者註冊時將不會傳送驗證碼。如果您要使用自訂身分驗證流程，驗證至少一種聯絡方式，而不使用由 Amazon Cognito 傳送的驗證碼，請選擇此選項。例如，您可以使用註冊前 Lambda 觸發器，自動驗證屬於特定網域的電子郵件地址。
> 若您並未驗證使用者的聯絡資訊，使用者可能無法使用您的應用程式。請記住，使用者需要已驗證聯絡資訊，才能執行以下操作：
>
> - 重設其密碼 – 當使用者在您的應用程式中選擇某個選項而呼叫了 `ForgotPassword` API 動作時，Amazon Cognito 會傳送臨時密碼至使用者的電子郵件地址或電話號碼。使用者需至少有一種已驗證的聯絡方式，Amazon Cognito 才會傳送此密碼。
> - 使用電子郵件地址或電話號碼作為別名登入 – 如果您將使用者集區設定成允許這類別名，則使用者必須驗證過其別名後才能使用別名登入。如需詳細資訊，請參閱 自訂登入屬性。

5. 選擇 Attributes to verify (要驗證的屬性)：

   傳送 SMS 訊息，驗證電話號碼

   使用者註冊時，Amazon Cognito 將以 SMS 訊息傳送驗證碼。如果您通常透過 SMS 訊息與您的使用者通訊，則請選擇此選項。例如，假使您想傳送交貨通知、約會確認或提醒，就要使用已驗證的電話號碼。確認帳戶時，使用者電話號碼將成為已驗證的屬性；您必須採取額外的動作來驗證使用者電子郵件地址並與其通訊。

   傳送電子郵件訊息，驗證電子郵件地址

   使用者註冊時，Amazon Cognito 將透過電子郵件訊息傳送驗證碼。如果您通常透過電子郵件與您的使用者通訊，請選擇此選項。例如，假使您想傳送帳單、訂單摘要或特殊優惠，就要使用已驗證的電子郵件地址。確認帳戶時，使用者電子郵件地址將成為已驗證的屬性；您必須採取額外的動作來驗證使用者電話號碼並與其通訊。

   如果電話號碼可用就會傳送 SMS 訊息，否則會傳送電子郵件訊息

   若您不需要對所有使用者都用同一種已驗證的聯絡方式，請選擇此選項。在這種情況下，您的應用程式註冊頁面可能會要求使用者僅驗證其偏好的聯絡方式。Amazon Cognito 傳送驗證碼時，會將該代碼傳送至您的應用程式 `SignUp` 請求中提供的聯絡方式。如果使用者同時提供了

電子郵件地址和電話號碼，而且您的應用程式也在 SignUp 請求中提供這兩種聯絡方式，則 Amazon Cognito 只會將驗證碼傳送至電話號碼。

如果您要求使用者一併驗證電子郵件地址和電話號碼，請選擇此選項。Amazon Cognito 會在使用者註冊時驗證一種聯絡方式，而您的應用程式必須在使用者登入後驗證另一種聯絡方式。如需詳細資訊，請參閱 若您要求使用者一併確認電子郵件地址和電話號碼。

6. 選擇 Save changes (儲存變更)。

身分驗證流程搭配電子郵件或電話驗證

如果您的使用者集區要求使用者驗證其聯絡資訊，您的應用程式即必須在使用者註冊時推行以下流程：

1. 使用者輸入使用者名稱、電話號碼和/或電子郵件地址，可能還有其他屬性，以註冊您的應用程式。
2. Amazon Cognito 服務從應用程式收到註冊請求。驗證請求中包含註冊必要的所有屬性之後，服務會完成註冊程序，並傳送確認碼到使用者的電話 (在簡訊中) 或電子郵件。確認碼的有效時間為 24 小時。
3. 服務向應用程式回報註冊完成，且使用者帳戶正在等待確認。回應中包含有關確認碼傳送到哪裡的資訊。此時使用者的帳戶處於未確認狀態，而使用者的電子郵件地址和電話號碼尚未驗證。
4. 應用程式現在可以提示使用者輸入確認碼。使用者不需要立即輸入確認碼。不過，使用者在輸入確認碼之前，將無法登入。
5. 使用者在應用程式中輸入確認碼。
6. 應用程式呼叫 ConfirmSignUp 將確認碼傳送到 Amazon Cognito 服務以供其驗證確認碼，如果確認碼正確，使用者的帳戶即會設為已確認狀態。成功確認使用者帳戶之後，Amazon Cognito 服務會自動將用來確認的屬性 (電子郵件地址或電話號碼) 標記為已驗證。除非此屬性的值變更，否則使用者不需要再驗證。
7. 此時使用者的帳戶處於已確認狀態，使用者可以進行登入。

若您要求使用者一併確認電子郵件地址和電話號碼

Amazon Cognito 在使用者註冊時僅驗證一種聯絡方式。如果 Amazon Cognito 必須擇其一驗證電子郵件地址或是電話號碼，其將選擇驗證電話號碼，即透過簡訊傳送驗證碼。例如，假使您將使用者集區設定成允許使用者驗證電子郵件地址或電話號碼，而且您的應用程式在註冊時提供了這兩個屬性，Amazon Cognito 便只會驗證電話號碼。使用者已驗證其電話號碼後，Amazon Cognito 會將使用者的狀態設為 CONFIRMED，以允許使用者登入您的應用程式。

待使用者登入後，您的應用程式即可提供選項以驗證當初註冊時未驗證的聯絡方式。為了驗證第二種聯絡方式，您的應用程式將呼叫 `VerifyUserAttribute` API 動作。請注意，此動作需要 `AccessToken` 參數，而 Amazon Cognito 只會為已驗證的使用者提供存取權杖。因此，您要等到使用者登入後才能驗證第二種聯絡方式。

若您要求使用者一併驗證電子郵件地址和電話號碼，請執行以下步驟：

1. 設定您的使用者集區以允許使用者驗證電子郵件地址或電話號碼。
2. 在應用程式的註冊流程，要求使用者一併提供電子郵件地址和電話號碼。呼叫 `SignUp` API 動作並由 `UserAttributes` 參數提供電子郵件地址和電話號碼。此時，Amazon Cognito 會將驗證碼傳送到使用者的電話。
3. 由應用程式界面顯示確認頁面，以供使用者輸入驗證碼。呼叫 `ConfirmSignUp` API 動作以確認使用者。此時，使用者的狀態為 `CONFIRMED`，而且使用者的電話號碼已驗證，但仍未驗證其電子郵件地址。
4. 顯示登入頁面，並透過呼叫 `InitiateAuth` API 動作進行使用者身分驗證。使用者通過身分驗證後，Amazon Cognito 會將存取權杖傳回您的應用程式。
5. 呼叫 `GetUserAttributeVerificationCode` API 動作。在請求中指定以下參數：

   - `AccessToken` – Amazon Cognito 在使用者登入時所傳回的存取權杖。
   - `AttributeName` – 指定 `"email"` 做為屬性值。

   Amazon Cognito 會將驗證碼傳送至使用者的電子郵件地址。
6. 顯示確認頁面以供使用者輸入驗證碼。當使用者提交驗證碼時，呼叫 `VerifyUserAttribute` API 動作。在請求中指定以下參數：

   - `AccessToken` – Amazon Cognito 在使用者登入時所傳回的存取權杖。
   - `AttributeName` – 指定 `"email"` 做為屬性值。
   - `Code` – 使用者所提供的驗證碼。

   此時，電子郵件地址即已驗證。

允許使用者註冊您的應用程式，但以使用者集區管理員身分確認使用者

您可能不希望使用者集區在您的使用者集區中自動傳送驗證訊息，但仍想要允許任何人註冊帳戶。例如，該模式為人工審查新的註冊請求，以及批次驗證和處理請求保留空間。您可以在 Amazon Cognito

主控台或使用已經過 IAM 驗證的 API 作業來確認新的使用者帳戶。AdminConfirmSignUp無論您的使用者集區是否傳送驗證訊息，您都可以以管理員身分確認使用者帳戶。

您只能將此技術用於確認使用者自助式註冊。若要確認您以管理員身分建立的使用者，請建立`Permanent`設定為的 AdminSetUserPasswordAPI 要求`True`。

1. 使用者輸入使用者名稱、電話號碼和/或電子郵件地址，可能還有其他屬性，以註冊您的應用程式。
2. Amazon Cognito 服務從應用程式收到註冊請求。驗證請求中包含註冊必要的所有屬性之後，服務會完成註冊程序，並且向應用程式回報註冊完成，正在等待確認。此時使用者的帳戶處於未確認狀態。必須在確認帳戶之後，使用者才能登入。
3. 確認使用者的帳戶。您必須使用 AWS 憑證登入 AWS Management Console 或簽署 API 要求，才能確認帳戶。

   a. 若要在 Amazon Cognito 主控台中確認使用者，請前往 使用者 索引標籤，選擇要確認的使用者，然後從 動作 功能表中選取 確認。

   b. 若要在 AWS API 或 CLI 中確認使用者，請建立 AdminConfirmSignUpAPI 要求，或admin-confirm-sign-up在 AWS CLI.
4. 此時使用者的帳戶處於已確認狀態，使用者可以進行登入。

運算私密雜湊值

最佳實務是將用戶端秘密指派給機密應用程式用戶端。將用戶端秘密指派給應用程式用戶端時，Amazon Cognito 使用者集區 API 請求必須包含在請求內文中包含用戶端秘密的雜湊。要驗證您對以下清單中 API 操作的用戶端秘密的了解，請將用戶端秘密與您的應用程式用戶端 ID 和使用者的使用者名稱串接起來，然後對該字串進行 base64 編碼。

當您的應用程式將使用者登錄到具有私密雜湊的用戶端時，您可用任何使用者集區登錄屬性的值作為私密雜湊的用戶名元素。當您的應用程式用 `REFRESH_TOKEN_AUTH` 在身份驗證操作中請求新權杖時，用戶名元素的值取決於您的登錄屬性。如果您的使用者集區沒有 `username` 作為登錄屬性，請從使用者之存取或 ID 權杖 `sub` 宣告中設定私密雜湊用戶名的值。如果 `username` 是登入屬性，請從 `username` 宣告中設定私密雜湊用戶名的值。

下列 Amazon Cognito 使用者集區 API 接受 `SecretHash` 參數中的用戶端秘密雜湊值。

- ConfirmForgotPassword
- ConfirmSignUp
- ForgotPassword

- ResendConfirmationCode
- SignUp

此外，下列 API 會在身分驗證參數或挑戰回應中接受 SECRET_HASH 參數中的用戶端秘密雜湊值。

API 操作	SECRET_HASH 的原生參數
InitiateAuth	AuthParameters
AdminInitiateAuth	AuthParameters
RespondToAuthChallenge	ChallengeResponses
AdminRespondToAuthChallenge	ChallengeResponses

私密雜湊值是一種 Base 64 編碼的金鑰式雜湊訊息驗證碼 (HMAC)，利用使用者集區用戶端的私密金鑰和使用者名稱，加上訊息中的用戶端 ID 計算而成。下列虛擬程式碼顯示這個值是如何計算出來的。在這個虛擬程式碼中，+代表串聯，HMAC_SHA256 代表使用 HmacSHA256 來產生 HMAC 值的函數，而 Base64 代表產生 Base-64 編碼版雜湊輸出的函數。

```
Base64 ( HMAC_SHA256 ( "Client Secret Key", "Username" + "Client Id" ) )
```

如需如何計算和使用參數的詳細概觀，請SecretHash參閱如何解決 Amazon Cognito 使用者集區 API 中的「無法驗證用戶端的秘密雜湊」錯誤<client-id>？ 在 AWS 知識中心。

您可以在伺服器端應用程式碼中使用下列程式碼範例：

Shell

```
echo -n "[username][app client ID]" | openssl dgst -sha256 -hmac [app client secret]
 -binary | openssl enc -base64
```

Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public static String calculateSecretHash(String userPoolClientId, String
 userPoolClientSecret, String userName) {
```

```
    final String HMAC_SHA256_ALGORITHM = "HmacSHA256";

    SecretKeySpec signingKey = new SecretKeySpec(
            userPoolClientSecret.getBytes(StandardCharsets.UTF_8),
            HMAC_SHA256_ALGORITHM);
    try {
        Mac mac = Mac.getInstance(HMAC_SHA256_ALGORITHM);
        mac.init(signingKey);
        mac.update(userName.getBytes(StandardCharsets.UTF_8));
        byte[] rawHmac =
 mac.doFinal(userPoolClientId.getBytes(StandardCharsets.UTF_8));
        return Base64.getEncoder().encodeToString(rawHmac);
    } catch (Exception e) {
        throw new RuntimeException("Error while calculating ");
    }
}
```

Python

```
import sys
import hmac, hashlib, base64
username = sys.argv[1]
app_client_id = sys.argv[2]
key = sys.argv[3]
message = bytes(sys.argv[1]+sys.argv[2],'utf-8')
key = bytes(sys.argv[3],'utf-8')
secret_hash = base64.b64encode(hmac.new(key, message,
 digestmod=hashlib.sha256).digest()).decode()
print("SECRET HASH:",secret_hash)
```

確認使用者帳戶，而不驗證電子郵件或電話號碼

預先註冊 Lambda 觸發程序可以用來在註冊時自動確認使用者帳戶，而不需要確認碼或驗證電子郵件或電話號碼。以這種方式確認的使用者可以立即登入，而不需要接收確認碼。

您也可以透過這個觸發器，將使用者的電子郵件或電話號碼標記為已驗證。

> **Note**
>
> 雖然這個方法很方便讓使用者入門，但建議您至少要自動驗證電子郵件或電話號碼其中一項。否則，如果使用者忘記密碼，將會無法復原。

如果您不需要使用者在註冊時接收並輸入確認碼，而且您不在預先註冊 Lambda 觸發程序中自動驗證電子郵件和電話號碼，您就沒有該使用者帳戶的已驗證電子郵件地址或電話號碼，這會是個風險。使用者可以之後再驗證電子郵件地址或電話號碼。不過，如果使用者忘記密碼，而且沒有已驗證的電子郵件地址或電話號碼，使用者就無法進入帳戶，因為忘記密碼流程需要已驗證的電子郵件或電話號碼，才能傳送驗證碼給使用者。

在使用者變更電子郵件或電話號碼時進行驗證

當使用者在您應用程式中更新其電子郵件地址或電話號碼時，如果您將使用者集區設定為自動驗證該屬性，Amazon Cognito 會立即將一則內含驗證碼的訊息傳送給使用者。然後，使用者必須將驗證訊息中的驗證碼提供給您的應用程式。然後，您的應用程式會在 VerifyUserAttributeAPI 要求中提交程式碼，以完成新屬性值的驗證。

如果您的使用者集區不要求使用者驗證更新後的電子郵件地址或電話號碼，Amazon Cognito 會立即變更已更新 `email` 或`phone_number` 屬性的值，並將屬性標記為未驗證。您的使用者無法使用未驗證的電子郵件或電話號碼登入。他們必須先完成驗證更新後的值，才能將該屬性作為登入別名。

如果您的使用者集區要求使用者驗證更新後的電子郵件地址或電話號碼，Amazon Cognito 會將該屬性保持為已驗證並將其設定為原始值，直到您的使用者驗證新屬性值為止。如果屬性是用於登入的別名，則在驗證將該屬性變更為新值之前，使用者都可以使用原始屬性值登入。如需如何設定您的使用者集區以要求使用者驗證更新後屬性的詳細資訊，請參閱設定電子郵件或電話驗證。

您可以使用自訂訊息 Lambda 觸發程序來自訂驗證訊息。如需詳細資訊，請參閱 自訂訊息 Lambda 觸發程序。當使用者的電子郵件地址或電話號碼未驗證時，您的應用程式就應該告知使用者必須驗證該屬性，並提供可讓使用者驗證新電子郵件地址或電話號碼的按鈕或連結。

由管理員或開發人員建立之使用者帳戶的確認和驗證程序

由管理員或開發人員建立的使用者帳戶已經是已確認狀態，因此使用者不需要輸入確認碼。Amazon Cognito 服務傳送給這些使用者的邀請訊息包含使用者名稱和臨時密碼。使用者必須先變更密碼，才能登入。如需詳細資訊，請參閱建立使用者帳戶為管理員中的自訂電子郵件和 SMS 訊息，以及使用 Lambda 觸發程序來自訂使用者集區工作流程中的自訂訊息觸發器。

匯入的使用者帳戶的確認和驗證程序

使用 AWS Management Console、CLI 或 API (請參閱從 CSV 檔案將使用者匯入使用者集區) 中的使用者匯入功能所建立的使用者帳戶已處於已確認狀態，因此使用者不需要輸入確認碼。不會傳送任何邀請訊息。不過，匯入的使用者帳戶需要使用者先呼叫 `ForgotPassword`API 來請求代碼，然後呼叫 `ConfirmForgotPassword` API，以使用所傳送的代碼來建立密碼，之後才能登入。如需詳細資訊，請參閱 需要匯入的使用者重設密碼。

當使用者帳戶匯入時，使用者的電子郵件或電話號碼必須標記為已驗證，因此在使用者登入時，不需要進行驗證。

在測試應用程式時傳送電子郵件

當使用者在您使用者集區的用戶端應用程式中建立和管理其帳戶時，Amazon Cognito 會傳送電子郵件訊息給使用者。如果您將使用者集區設定成要求電子郵件驗證，Amazon Cognito 便會在以下情況傳送電子郵件：

- 使用者註冊。
- 使用者更新其電子郵件地址。
- 使用者執行某項動作而呼叫了 `ForgotPassword` API 動作。
- 您以管理員的身分建立使用者帳戶。

視發出電子郵件的動作而定，此電子郵件將包含驗證碼或臨時密碼。您的使用者必須接收這類電子郵件並了解其訊息內容，否則可能無法登入與使用您的應用程式。

為確保順利傳送電子郵件且訊息內容正確無誤，請在您的應用程式中測試從 Amazon Cognito 起始電子郵件傳遞的動作。例如，您可以使用應用程式的註冊頁面或使用 `SignUp` API 動作，透過註冊測試電子郵件地址的方式發出電子郵件。以這種方式測試時，請牢記以下要點：

> **重要**
>
> 當您使用電子郵件地址測試從 Amazon Cognito 發出電子郵件的動作時，切勿使用虛假 (信箱不存在) 的電子郵件地址。請使用能夠收到 Amazon Cognito 發出電子郵件的真實電子郵件地址，以免導致硬退信的狀況。
>
> 硬退信是由於 Amazon Cognito 未能將電子郵件傳遞至收件人的信箱所造成，若信箱不存在就一定會發生。
>
> Amazon Cognito 限制了 AWS 帳戶可以傳送的電子郵件數量，這些電子郵件會持續發生硬退件。

當您測試發出電子郵件的動作時，請使用以下任一種電子郵件地址避免造成硬退信：

- 您所擁有且專供測試用的電子郵件帳戶的地址。若您使用自有的電子郵件地址，便能收到由 Amazon Cognito 傳送的電子郵件。有了這封電子郵件，您即可使用驗證碼測試應用程式的註冊經驗。如果您為使用者集區自訂了電子郵件訊息，就可以檢查自訂內容是否正確無誤。

- 信箱模擬器地址 success@simulator.amazonses.com。若您使用模擬器地址，Amazon Cognito 將能順利傳送電子郵件，但您無法查看其內容。當您無須使用驗證碼且不必檢查電子郵件訊息時，即適用此選項。
- 附加任意標籤的信箱模擬器地址，如 success+user1@simulator.amazonses.com 或 success+user2@simulator.amazonses.com。Amazon Cognito 將能順利傳送電子郵件至這類地址，但您無法查看其傳送的電子郵件。當您希望透過新增多名測試使用者至使用者集區的方式測試註冊程序，而且每一名測試使用者皆有其獨一的電子郵件地址時，即適用此選項。

設定電子郵件或電話驗證

您可以在 傳訊 索引標籤下方，選擇電子郵件或電話驗證的各項設定。有關多重要素驗證 (MFA) 的詳細資訊，請參閱簡訊 MFA。

Amazon Cognito 使用 Amazon SNS 以傳送 簡訊。如果您 AWS 服務 之前沒有從 Amazon Cognito 或其他任何其他方式傳送簡訊，Amazon SNS 可能會將您的帳戶放在簡訊沙箱中。我們建議您先將測試訊息傳送到已驗證的電話號碼，然後再將帳戶從沙盒移至生產環境。此外，如果您打算將簡訊傳送至美國目的地電話號碼，必須向 Amazon Pinpoint 取得來源或寄件者 ID。若要為簡訊設定 Amazon Cognito 使用者集區，請參閱 Amazon Cognito 使用者集區的簡訊設定。

Amazon Cognito 可以自動驗證電子郵件地址或電話號碼。若要進行此驗證，Amazon Cognito 會傳送驗證碼或驗證連結。如果是電子郵件地址，Amazon Cognito 會以電子郵件訊息傳送代碼或連結。當您在 Amazon Cognito 主控台的 傳訊 索引標籤中編輯 驗證訊息 範本時，可以選擇 代碼 或 連結 的 驗證類型。如需詳細資訊，請參閱 自訂電子郵件驗證訊息。

如果是電話號碼，Amazon Cognito 會以簡訊傳送驗證碼。

Amazon Cognito 必須驗證電話號碼或電子郵件地址以確認使用者，並協助使用者恢復忘記的密碼。或者，您可以使用預先註冊 Lambda 觸發程序自動確認使用者，或使用 AdminConfirmSignUpAPI 作業。如需詳細資訊，請參閱 註冊及確認使用者帳戶。

驗證碼或連結的有效時間為 24 小時。

如果您選擇要求驗證電子郵件地址或電話號碼，Amazon Cognito 會在使用者註冊時自動傳送驗證碼或連結。如果使用者集區具有 自訂 SMS 寄件者 Lambda 觸發程序 或 自訂電子郵件寄件者 Lambda 觸發程序，將會改為叫用該函數。

備註

- Amazon SNS 會針對其用於驗證電話號碼的簡訊部分另外收費。傳送電子郵件訊息是免費的。如需 Amazon SNS 定價的相關資訊，請參閱全球 SMS 定價。如需可使用 SMS 簡訊的國家/地區最新清單，請參閱支援的區域和國家。
- 當您在應用程式中測試從 Amazon Cognito 產生電子郵件的動作時，請使用 Amazon Cognito 能夠送達而不致發生硬退信狀況的真實電子郵件地址。如需詳細資訊，請參閱 the section called “在測試應用程式時傳送電子郵件”。
- 忘記密碼流程需要使用者的電子郵件或電話號碼以驗證使用者。

Important

如果使用者同時使用電子郵件地址和電話號碼來進行註冊，而您的使用者集區設定要求這兩個屬性都要驗證，Amazon Cognito 會透過簡訊將驗證碼傳送至該電話號碼。Amazon Cognito 尚未驗證電子郵件地址，因此您的應用程式必須致電GetUser以查看電子郵件地址是否等待驗證。如果確實需要驗證，則應用程序必須致電GetUserAttributeVerificationCode以啟動電子郵件驗證流程。然後，它必須通過調用提交驗證碼VerifyUserAttribute。

您可以針對 AWS 帳戶 和個別訊息調整 SMS 訊息支出配額。這些限制僅適用於傳送簡訊的成本。如需詳細資訊，請參閱 Amazon SNS 常見問答集中的什麼是帳戶層級與訊息層級費用配額及其運作方式？。

Amazon Cognito 使用 Amazon SNS 資源在您建立使用者集區的 AWS 區域 位置，或從下表傳統 Amazon SNS 替代區域傳送簡訊。亞太 (首爾) 區域的 Amazon Cognito 使用者集區除外。這些使用者集區使用您在亞太 (東京) 區域的 Amazon SNS 組態。如需詳細資訊，請參閱 選擇 AWS 區域 Amazon SNS 短信。

Amazon Cognito 區域	舊式 Amazon SNS 備用區域
美國東部 (俄亥俄)	美國東部 (維吉尼亞北部)
亞太區域 (孟買)	亞太區域 (新加坡)
亞太區域 (首爾)	亞太區域 (東京)

Amazon Cognito 區域	舊式 Amazon SNS 備用區域
加拿大 (中部)	美國東部 (維吉尼亞北部)
歐洲 (法蘭克福)	歐洲 (愛爾蘭)
歐洲 (倫敦)	歐洲 (愛爾蘭)

範例：如果您的 Amazon Cognito 使用者集區位於亞太區域 (孟買)，而且已在 ap-southeast-1 區域增加支出限制，那麼您可能不希望個別請求增加 ap-south-1 的支出限制。反之，您可以使用亞太區域 (新加坡) 的 Amazon SNS 資源。

驗證對電子郵件地址和電話號碼的更新

在使用者變更電子郵件地址或電話號碼屬性值後，該屬性會立即變為作用中且未驗證。Amazon Cognito 還會要求您的使用者在 Amazon Cognito 更新屬性之前先驗證新值。當您要求使用者先驗證新值時，他們可以使用原始值登入與接收訊息，直到他們驗證新值為止。

當使用者可以將其電子郵件地址或電話號碼作為您使用者集區中的登入別名時，已更新屬性的登入名稱取決於您是否要求驗證已更新的屬性。當您要求使用者驗證已更新的屬性時，使用者可以使用原始屬性值登入，直到他們驗證新值為止。如果您不要求使用者驗證已更新的屬性，則使用者無法以新的或原始屬性值登入或接收訊息，直到他們驗證新值為止。

例如，您的使用者集區允許以電子郵件地址別名登入，且要求使用者在更新時驗證其電子郵件地址。Sue 使用 `sue@example.com` 登入，她想要將電子郵件地址變更為 `sue2@example.com`，但不小心輸入成 **s**`sue2@example.com`。Sue 沒有收到驗證電子郵件，因此無法驗證 `ssue2@example.com`。Sue 使用 `sue@example.com` 登入，並在您的應用程式中重新提交表單，將其電子郵件地址變更為 `sue2@example.com`。她收到此電子郵件，將驗證碼提供給您的應用程式，並開始使用 `sue2@example.com` 登入。

當使用者更新屬性且您的使用者集區驗證新屬性值時

- 他們可以在確認代碼以驗證新值之前，使用原始屬性值登入。
- 他們只能在確認代碼以驗證新值之後，使用新屬性值登入。
- 如果您在 AdminUpdateUserAttributesAPI 請求`true`中設定`email_verified`或`phone_number_verified`設定，他們可以在確認 Amazon Cognito 傳送給他們的程式碼之前登入。

當使用者更新屬性，但您的使用者集區未驗證新屬性值時

- 他們無法使用原始屬性值登入或接收訊息。
- 在確認代碼以驗證新值之前，他們無法使用新屬性值登入或接收確認碼以外的訊息。
- 如果您在 AdminUpdateUserAttributesAPI 請求`true`中設定`email_verified`或`phone_number_verified`設定，他們可以在確認 Amazon Cognito 傳送給他們的程式碼之前登入。

要求使用者更新其電子郵件地址或電話號碼時需要驗證屬性

1. 登入 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 在導覽窗格中，選擇 User Pools (使用者集區)，然後選擇您要編輯的使用者集區。
3. 在 Sign-up experience (註冊體驗) 索引標籤中，選擇 Attribute verification and user account confirmation (屬性驗證與使用者帳戶確認) 下方的 Edit (編輯)。
4. 選擇 Keep original attribute value active when an update is pending (當更新待處理時，保持原始屬性值處於作用中)。
5. 在 Active attribute values when an update is pending (更新待處理時的作用中屬性值) 下方，選擇您希望在 Amazon Cognito 更新值之前要求使用者驗證的屬性。
6. 選擇儲存變更。

若要要求使用 Amazon Cognito API 進行屬性更新驗證，您可以在UpdateUserPool請求中設定`AttributesRequireVerificationBeforeUpdate`參數。

授權 Amazon Cognito 代表您傳送 SMS 訊息

Amazon Cognito 需要您的許可，才能代表您傳送簡訊給使用者。若要授與該權限，您可以建立 AWS Identity and Access Management (IAM) 角色。在 Amazon Cognito 主控台的 傳訊 索引標籤中，選擇 編輯 以設定角色。

設定 SMS、電子郵件驗證訊息和使用者邀請訊息

Amazon Cognito 可讓您自訂 SMS 和電子郵件驗證訊息以及使用者邀請訊息，以增強應用程式的安全性和使用者體驗。使用 Amazon Cognito，您可以根據應用程式的需求，在程式碼型或單鍵連結驗證之間進行選擇。本主題討論如何在 Amazon Cognito 主控台中個人化多因素身份驗證 (MFA) 和驗證通訊。

在 訊息範本 下的 傳訊 索引標籤中，您可以自訂：

- 您的 SMS 文字訊息多重要素驗證 (MFA) 訊息
- 您的簡訊和電子郵件驗證訊息
- 電子郵件的驗證類型：代碼或連結
- 您的使用者邀請訊息
- 通行於使用者集區之電子郵件的 FROM (寄件者) 和 REPLY-TO (回覆至) 電子郵件地址

> **Note**
>
> 您必須在 Verifications (驗證) 索引標籤中，選擇需要電話號碼和電子郵件驗證，才會顯示簡訊和電子郵件驗證訊息範本。同樣地，MFA 設定必須是 required (必要) 或 optional (選用)，才會顯示 SMS MFA 訊息範本。

主題

- 訊息範本
- 自訂 SMS 訊息
- 自訂電子郵件驗證訊息
- 自訂使用者邀請訊息
- 自訂您的電子郵件地址
- 授權 Amazon Cognito 代表您傳送 Amazon SES 電子郵件 (從自訂 FROM (寄件者) 電子郵件地址)

訊息範本

您可以使用訊息範本將欄位插入使用預留位置的訊息中，對應值將會取代這些預留位置。

範本預留位置

描述	權杖
驗證碼	`{####}`
臨時密碼	`{####}`
使用者名稱	`{username}`

> **Note**
>
> 您無法在驗證電子郵件訊息中使用 {username} 預留位置。您可以在隨AdminCreateUser作業產生的邀請電子郵件訊息中使用{username}預留位置。這些邀請電子郵件訊息使用兩個預留位置：使用者名稱 ({username}) 和臨時密碼 ({####})。

您可以使用進階安全範本預留位置來執行下列作業，如下所示：

- 包含事件的特定詳細資訊，例如 IP 地址、城市、國家、登入時間和裝置名稱。Amazon Cognito 進階安全功能可以分析這些詳細內容。
- 驗證一鍵式連結是否有效。
- 使用事件 ID、意見回饋權杖和使用者名稱，建置您自己的一鍵式連結。

> **Note**
>
> 若要在進階安全性電子郵件範本中產生一鍵式連結接並使用 {one-click-link-valid} 和 {one-click-link-invalid} 預留位置，您必須已經為使用者集區設定網域。

進階安全範本預留位置

描述	權杖
IP 地址	{ip-address}
City	{city}
Country	{country}
登入時間	{login-time}
裝置名稱	{device-name}
一鍵式連結有效	{one-click-link-valid}
一鍵式連結無效	{one-click-link-invalid}
事件 ID	{event-id}

描述	權杖
意見回饋權杖	`{feedback-token}`

自訂 SMS 訊息

> **Note**
>
> 在新的 Amazon Cognito 主控台體驗中，您可以自訂簡訊。

您可以在訊息索引標籤中的訊息範本標題下，自訂簡訊以進行多重要素驗證 (MFA)。

> **Important**
>
> 您的自訂訊息必須包含 `{####}` 預留位置。傳送訊息前，此預留位置會取代為身分驗證代碼。

Amazon Cognito 會強制實施簡訊長度上限，包括驗證碼在內不可超過 140 個 UTF-8 字元。

自訂 SMS 驗證訊息

若要自訂用來進行電話號碼驗證的簡訊，您可以編輯 Do you want to customize your SMS verification messages? (您要自訂簡訊驗證訊息嗎？) 標題底下的範本。

> **Important**
>
> 您的自訂訊息必須包含 `{####}` 預留位置。傳送訊息前，此預留位置會取代為驗證碼。

訊息的長度上限為 140 個 UTF-8 字元，包括驗證碼。

自訂電子郵件驗證訊息

若要使用 Amazon Cognito 驗證使用者集區中使用者的電子郵件地址，可以向使用者發送電子郵件，其中包含一個使用者可選取接的連結，或者可以向使用者發送可供其輸入的代碼。

若要自訂電子郵件地址驗證訊息的電子郵件主旨和郵件內容，請在使用者集區的 傳訊 索引標籤中編輯 驗證訊息 範本。當您編輯 驗證訊息 範本時，可以選擇 代碼 或 連結 的 驗證類型。

在您選擇 代碼 作為驗證類型時，您的自訂訊息就必須包含 {####} 預留位置。當您傳送訊息時，驗證碼會取代此預留位置。

在您選擇 連結 作為驗證類型時，您的自訂訊息就必須包含 {##Verify Your Email##} 格式的預留位置。您可以變更預留位置字元之間的文字字串，例如 {##Click here##}。標題為驗證您的電子郵件的驗證鏈接會取代此預留位置。

電子郵件驗證訊息的連結會將您的使用者引導至 URL，如下列範例所示。

```
https://<your user pool domain>/confirmUser/?
client_id=abcdefg12345678&user_name=emailtest&confirmation_code=123456
```

訊息的長度上限為 20,000 個 UTF-8 字元，包括驗證碼 (如果有)。您可以在此訊息中使用 HTML 標籤來格式化內容。

自訂使用者邀請訊息

您可以透過編輯 傳訊 索引標籤中的 邀請訊息 範本來自訂 Amazon Cognito 透過簡訊或電子郵件發送給新使用者的使用者邀請訊息。

> **Important**
>
> 您的自訂訊息必須包含 {username} 和 {####} 預留位置。Amazon Cognito 傳送邀請訊息時，會以使用者的使用者名稱和密碼取代這些預留位置。

SMS 訊息的長度上限為 140 個 UTF-8 字元，包括驗證碼。電子郵件訊息的長度上限為 20,000 個 UTF-8 字元，包括驗證碼。您可以在此電子郵件訊息中使用 HTML 標籤來格式化內容。

自訂您的電子郵件地址

依預設，Amazon Cognito 會從地址 no-reply@verificationemail.com 傳送電子郵件訊息給使用者集區中的使用者。您可以選擇指定自訂 FROM (寄件者) 電子郵件地址和 REPLY-TO (回覆至) 電子郵件地址，而不要使用 no-reply@verificationemail.com。

自訂 FROM (寄件者) 電子郵件地址和 REPLY-TO (回覆至) 電子郵件地址

1. 導覽到 Amazon Cognito 主控台，選擇 User Pools (使用者集區)。
2. 從清單中選擇現有的使用者集區，或建立使用者集區。
3. 選擇 Messaging (簡訊) 索引標籤。在 Email (電子郵件) 下，選擇 Edit (編輯)。

4. 選擇 SES Region (SES 區域)。
5. 從您已使用所選 SES Region (SES 區域) 中 Amazon SES 進行驗證的電子郵件地址清單中，選擇 FROM email address (寄件者電子郵件地址)。若要使用來自已驗證網域的電子郵件地址，請在 AWS Command Line Interface 或 AWS API 中設定電子郵件集。如需詳細資訊，請參閱《Amazon Simple Email Service 開發人員指南》中的在 Amazon SES 中驗證電子郵件地址和網域。
6. 從您所選擇 SES Region (SES 區域) 中的組態設定清單中選擇 Configuration set (組態集)。
7. 請為您的電子郵件訊息輸入易於使用的 FROM sender name (FROM (寄件者) 傳送人名稱)，格式為 `John Stiles <johnstiles@example.com>`。
8. 若要自訂 REPLY-TO (回覆至) 電子郵件地址，請在 REPLY-TO email address (回覆至電子郵件地址) 欄位中輸入有效的電子郵件地址。

授權 Amazon Cognito 代表您傳送 Amazon SES 電子郵件 (從自訂 FROM (寄件者) 電子郵件地址)

您可以將 Amazon Cognito 設定為從自訂 FROM (寄件者) 電子郵件地址，而非其預設地址，傳送電子郵件。若要使用自訂地址，您必須授予 Amazon Cognito 許可，才能從 Amazon SES 驗證的身分傳送電子郵件訊息。大部分情況，您可以透過建立傳送授權政策授予許可。如需詳細資訊，請參閱《Amazon Simple Email Service 開發人員指南》中的搭配 Amazon SES 使用傳送授權。

當您將使用者集區設定為使用 Amazon SES 來傳送電子郵件訊息時，Amazon Cognito 會在您帳戶中建立 `AWSServiceRoleForAmazonCognitoIdpEmailService` 角色，以授予對 Amazon SES 的存取權。使用 `AWSServiceRoleForAmazonCognitoIdpEmailService` 服務連結角色時，不需要傳送授權政策。當您在使用者集區中使用這兩個預設電子郵件功能時，只需要新增傳送授權政策和經過驗證的 Amazon SES 身分作為 FROM (寄件者) 地址。

如需 Amazon Cognito 所建立服務連結角色的詳細資訊，請參閱使用 Amazon Cognito 的服務連結角色。

下列範例傳送授權政策會向 Amazon Cognito 授予使用 Amazon SES 已驗證身分的有限能力。執行此操作時，Amazon Cognito 只能代表 `aws:SourceArn` 條件中的使用者集區和 `aws:SourceAccount` 條件中的帳戶傳送電子郵件訊息。如需更多範例，請參閱《Amazon Simple Email Service 開發人員指南》中的 Amazon SES 傳送授權政策範例。

> **Note**
>
> 在這個範例中，"Sid" 值是唯一識別陳述式的任意字串。如需政策語法的詳細資訊，請參閱《Amazon Simple Email Service 開發人員指南》中的 Amazon SES 傳送授權政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "stmnt1234567891234",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "email.cognito-idp.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "aws:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
}
```

您從下拉式選單選取 Amazon SES 身分時，Amazon Cognito 主控台就會為您新增類似政策。如果您使用 CLI 或 API 來設定使用者集區，則必須將結構與之前範例相似的政策連接至您的 Amazon SES 身分。

建立使用者帳戶為管理員

建立使用者集區之後，您可以使用 AWS Management Console，也可以使用 AWS Command Line Interface 或 Amazon Cognito API 來建立使用者。您可以在使用者集區中為新使用者建立設定檔，然後透過簡訊或電子郵件，傳送附有註冊指示的歡迎訊息給使用者。

開發人員和管理員可以執行下列工作：

- 使用 AWS Management Console或呼叫 `AdminCreateUser`API 來建立新的使用者描述檔。
- 設定使用者屬性值。

- 建立自訂屬性。
- 在 AdminCreateUser API 要求中設定不可變自訂屬性的值。此功能無法在 Amazon Cognito 主控台中使用。
- 指定臨時密碼，或允許 Amazon Cognito 自動產生密碼。
- 指定是否要為新使用者將所提供的電子郵件地址和電話號碼標記為已驗證。
- 透過 AWS Management Console或自訂訊息 Lambda 觸發器，為新使用者指定自訂簡訊和電子郵件邀請訊息。如需更多詳細資訊，請參閱 使用 Lambda 觸發程序來自訂使用者集區工作流程。
- 指定是否要透過簡訊及/或電子郵件傳送邀請訊息。
- 呼叫 AdminCreateUserAPI，並為 RESEND 參數指定 MessageAction，以重送歡迎訊息給現有的使用者。

> **Note**
>
> 目前無法使用 AWS Management Console來執行此動作。

- 建立使用者後，抑制傳送邀請訊息。
- 為使用者帳戶指定過期時間限制 (最多 90 天)。
- 允許使用者自行登入，或要求僅限由管理員新增新使用者。

由管理員或開發人員建立之使用者的身分驗證流程

這些使用者的身分驗證流程包括提交新密碼，以及為必要屬性提供任何遺漏值等額外步驟。接下來會概述這些步驟，步驟 5、6 和 7 專屬於這些使用者。

1. 使用者第一次開始登入時，需提交他們的使用者名稱和密碼。
2. 軟體開發套件會呼叫 `InitiateAuth(Username, USER_SRP_AUTH)`。
3. Amazon Cognito 會以 Salt & Secret 區塊傳回 PASSWORD_VERIFIER 挑戰。
4. 軟體開發套件會執行 SRP 計算，並呼叫 `RespondToAuthChallenge(Username, <SRP variables>, PASSWORD_VERIFIER)`。
5. Amazon Cognito 傳回 NEW_PASSWORD_REQUIRED 挑戰。此挑戰的正文包括使用者的目前屬性，以及使用者集區中目前在使用者描述檔中沒有值的所有必要屬性。如需詳細資訊，請參閱 RespondToAuthChallenge。
6. 系統會提示使用者輸入新密碼，以及必要屬性的任何遺漏值。

7. 軟體開發套件會呼叫 `RespondToAuthChallenge(Username, <New password>, <User attributes>)`。
8. 如果使用者需要 MFA 的第二要素，Amazon Cognito 會傳回 SMS_MFA 挑戰，並且會提交驗證碼。
9. 使用者成功變更密碼，並選擇性地提供屬性值或完成 MFA 之後，使用者即登入，並且會發出權杖。

當使用者滿足所有挑戰後，Amazon Cognito 服務會將使用者標記為已確認，並且為使用者發出 ID、存取和重新整理權杖。如需更多詳細資訊，請參閱 將權杖用於使用者集區。

在 AWS Management Console中建立新的使用者

您可以使用 Amazon Cognito 主控台設定使用者密碼要求、設定傳送給使用者的邀請和驗證訊息，以及新增使用者。

設定密碼政策並啟用自我註冊

您可以設定最低密碼複雜度的設定，以及使用者是否可以在使用者集區中使用公有 API 進行註冊。

設定密碼政策

1. 導覽到 Amazon Cognito 主控台，選擇 User Pools (使用者集區)。
2. 從清單中選擇現有的使用者集區，或建立使用者集區。
3. 選擇 Sign-in experience (登入體驗) 索引標籤並找到 Password policy (密碼政策)。選擇編輯 。
4. 選擇 Custom (自訂) 的 Password policy mode (密碼政策模式)。
5. 選擇 Password minimum length (密碼長度下限)。如需密碼長度要求的限制，請參閱使用者集區資源配額。
6. 選擇 Password complexity (密碼複雜度) 要求。
7. 選擇管理員設定的密碼有效期限。
8. 選擇 Save changes (儲存變更)。

允許自助註冊

1. 導覽到 Amazon Cognito 主控台，選擇 User Pools (使用者集區)。
2. 從清單中選擇現有的使用者集區，或建立使用者集區。

3. 選擇 Sign-up experience (註冊體驗) 索引標籤並找到 Self-service sign-up (自助註冊)。選擇 Edit (編輯)。
4. 選擇是否要 Enable self-registration (啟用自我註冊)。自我註冊通常與公有應用程式用戶端搭配使用，這些用戶端需要在使用者集區中註冊新使用者，而無需分配用戶端密碼或 AWS Identity and Access Management (IAM) API 憑證。

> **停用自我註冊**
>
> 如果您未啟用自我註冊，則必須使用 IAM API 憑證透過管理 API 動作或透過聯合身分提供者登入來建立新使用者。

5. 選擇 Save changes (儲存變更)。

自訂電子郵件和 SMS 訊息

自訂使用者訊息

您可以自訂 Amazon Cognito 在邀請使用者登入、使用者註冊使用者帳戶或使用者登入並收到提示進行多重要素驗證 (MFA) 時傳送給使用者的訊息。

> **Note**
>
> Invitation message (邀請訊息) 會在您在使用者集區中建立使用者時傳送，並邀請他們登入。Amazon Cognito 會將初始登入資訊傳送至使用者的電子郵件地址或電話號碼。
> 使用者註冊使用者集區中的使用者帳戶時，系統會傳送 Verification message (驗證訊息)。Amazon Cognito 會傳送代碼給使用者。當使用者將代碼提供給 Amazon Cognito 時，他們會驗證其聯絡資訊，並確認其帳戶以進行登入。驗證碼有效時間為 24 小時。
> 您在使用者集區中啟用 SMS MFA 時，以及已設定 SMS MFA 的使用者登入並提示進行 MFA 時，系統會傳送 MFA message (MFA 訊息)。

1. 導覽到 Amazon Cognito 主控台，選擇 User Pools (使用者集區)。
2. 從清單中選擇現有的使用者集區，或建立使用者集區。
3. 選擇 Messaging (簡訊) 索引標籤並找到 Message templates (訊息範本)。選取 Verification messages (驗證訊息)、Invitation messages (邀請訊息) 或 MFA messages (MFA 訊息)，然後選擇 Edit (編輯)。
4. 自訂所選訊息類型的訊息。

> **Note**
>
> 當您自訂訊息時，必須包含訊息範本中的所有變數。如果未包含變數，例如 {####}，則您的使用者將沒有足夠的資訊來完成訊息動作。
> 如需詳細資訊，請參閱訊息範本。

5. a. 驗證訊息

   i. 選擇 Email (電子郵件) 訊息的 Verification type (驗證類型)。Code (代碼) 驗證會傳送使用者必須輸入的數字碼。Link (連結) 驗證會傳送連結，使用者可以按一下以驗證其聯絡資訊。Link (連結) 訊息的變數中的文字會顯示為超連結文字。例如，使用變數 {##Click here##} 的訊息範本會在電子郵件訊息中顯示為請點選此處。

   ii. 輸入 Email (電子郵件) 訊息的 Email subject (電子郵件主旨)。

   iii. 輸入 Email (電子郵件) 訊息的自訂 Email message (電子郵件訊息) 範本。您可以使用 HTML 自訂此範本。

   iv. 輸入 SMS 訊息的 SMS message (SMS 訊息) 範本。

   v. 選擇 Save changes (儲存變更)。

   b. Invitation messages (邀請訊息)

   i. 輸入 Email (電子郵件) 訊息的 Email subject (電子郵件主旨)。

   ii. 輸入 Email (電子郵件) 訊息的自訂 Email message (電子郵件訊息) 範本。您可以使用 HTML 自訂此範本。

   iii. 輸入 SMS 訊息的 SMS message (SMS 訊息) 範本。

   iv. 選擇 Save changes (儲存變更)。

   c. MFA messages (MFA 訊息)

   i. 輸入 SMS 訊息的 SMS message (SMS 訊息) 範本。

   ii. 選擇 Save changes (儲存變更)。

建立使用者

建立使用者

您可以從 Amazon Cognito 主控台為您的使用者集區建立新使用者。一般而言，使用者可以在設定密碼之後登入。若要使用電子郵件地址登入，使用者必須分別驗證 `email` 屬性。若要使用電話號碼登

入，使用者必須驗證 phone_number 屬性。若要以管理員身分確認帳戶，您也可以使用 AWS CLI 或 API，或使用聯合身分提供者建立使用者描述檔。如需詳細資訊，請參閱《Amazon Cognito API 參考》。

1. 導覽到 Amazon Cognito 主控台，選擇 User Pools (使用者集區)。
2. 從清單中選擇現有的使用者集區，或建立使用者集區。
3. 選擇 Users (使用者) 索引標籤，然後選擇 Create Users (建立使用者)。
4. 檢閱 User pool sign-in and security requirements (使用者集區登入和安全要求) 以取得有關密碼要求、可用帳戶復原方法和使用者集區別名屬性的指引。
5. 選擇您要如何傳送 Invitation message (邀請訊息)。選擇 SMS 訊息、電子郵件訊息或兩者皆可。

   > **Note**
   >
   > 請先在使用者集區的 Messaging (簡訊) 索引標準中使用 Amazon Simple Notification Service 和 Amazon Simple Email Service 設定寄件者和 AWS 區域，您才能傳送邀請訊息。適用收件人訊息和數據傳輸費率。Amazon SES 會單獨向您收取電子郵件訊息費用，Amazon SNS 會單獨向您收取 SMS 訊息費用。

6. 為新使用者選擇 Username (使用者名稱)。
7. 選擇您是否要為使用者 Create a password (建立密碼) 或允許 Amazon Cognito Generate a password (產生密碼)。任何臨時密碼必須遵守使用者集區密碼政策。
8. 選擇 Create (建立)。
9. 選擇 Users (使用者) 索引標籤，然後為使用者選擇 User name (使用者名稱) 項目。新增和編輯 User attributes (使用者屬性) 和 Group memberships (群組成員資格)。檢閱 User event history (使用者事件歷史記錄)。

新增群組至使用者集區

Amazon Cognito 使用者集區中的群組支援可讓您建立和管理群組、新增使用者至從群組，以及從群組移除使用者。使用群組來建立使用者集合，以管理其許可，或代表不同類型的使用者。您可以將 AWS Identity and Access Management (IAM) 角色指派給群組，以定義群組成員的許可。

您可以在使用者集區中使用群組來建立使用者集合，通常這麼做以為這些使用者設定許可。例如，您可以針對網站和應用程式的讀者、參與者和編輯者，建立個別的使用者群組。若是使用與群組相關聯的 IAM 角色，您還可以為這些不同的群組設定不同的權限，這樣就只有參與者可以在 Amazon S3 中輸入內容，且只有編輯者可以透過 Amazon API Gateway 中的 API 發佈內容。

您可以從 API 和 CLI 建立和管理使用者集區中的群組。 AWS Management Console身為開發人員 (使用 AWS 認證)，您可以建立、讀取、更新、刪除和列出使用者集區的群組。您也可以新增使用者，以及從群組移除使用者。

在使用者集區中使用群組不會產生額外成本。如需詳細資訊，請參閱 Amazon Cognito 定價。

指派 IAM 角色給群組

您可以使用 IAM 角色透過群組來控制資源的許可。IAM 角色包含信任政策和許可政策。角色信任政策指定誰可以使用該角色。許可政策指定群組成員可以存取的動作和資源。建立 IAM 角色時，請設定角色信任政策讓群組使用者可以擔任該角色。在角色許可政策中，指定您希望群組擁有的許可。

在 Amazon Cognito 中建立群組時，您可以透過提供角色的 ARN 來指定 IAM 角色。當群組成員使用 Amazon Cognito 登入時，可以從身分集區接收暫時性登入資料。其許可由相關聯 IAM 角色決定。

個別使用者可以在多個群組中。身為開發人員，當一個使用者在多個群組中時，您有下列選項可以自動選擇 IAM 角色：

- 您可以為每個群組指派優先順序值。系統會選擇優先順序較高 (值較小) 的群組，並套用其相關聯的 IAM 角色。
- 透過身分集區要求使用者 AWS 認證時，您的應用程式也可以從可用角色中選擇，方法是在GetCredentialsForIdentityCustomRoleARN參數中指定角色 ARN。指定的 IAM 角色必須符合使用者可用的角色。

指定優先順序值給群組

一個使用者可以隸屬於多個群組。在使用者的存取權和 ID 權杖中，cognito:groups 會宣告包含使用者隸屬的所有群組清單。cognito:roles宣告包含對應群組的角色清單。

由於一個使用者可以隸屬於多個群組，因此可以為每個群組各指派一個優先順序。此為非負數，指定在使用者集區中，此群組相對於使用者可隸屬之其他群組的優先順序。零是最高優先順序值。優先順序值較小的群組，優先順序高於較大值或空值的群組。如果使用者隸屬於兩個以上群組，優先順序值最小的群組會將其 IAM 角色套用至使用者 ID 權杖的 cognito:preferred_role 宣告。

兩個群組可以有相同的優先順序值。如果發生這種情況，則兩個群組的優先順序相等。如果優先順序值相同的兩個群組具有相同的角色 ARN，則會為每個群組中的使用者，在 ID 權杖的 cognito:preferred_role宣告中使用該角色。如果這兩個群組有不同的角色 ARN，則不會在使用者的 ID 權杖中設定 cognito:preferred_role宣告。

使用群組來控制 Amazon API Gateway 許可

您可以在使用者集區中使用群組來控制 Amazon API Gateway 許可。使用者隸屬的群組會包含在來自 `cognito:groups` 宣告之使用者集區的 ID 權杖和存取權杖中。您可以將具有請求的 ID 或存取權杖提交到 Amazon API Gateway，並使用 Amazon Cognito 使用者集區授權方來執行 REST API。如需詳細資訊，請參閱《API Gateway 開發人員指南》中的使用 Amazon Cognito 使用者集區作為授權方來控制對 REST API 的存取。

您也可以使用自訂 JWT 授權方授權對 Amazon API Gateway HTTP API 的存取。如需詳細資訊，請參閱《API Gateway 開發人員指南》中的使用 JWT 授權方控制對 HTTP API 的存取。

群組的限制

使用者群組具有下列限制：

- 您可以建立的群組數目受 Amazon Cognito 服務配額的限制。
- 群組不能巢狀組合。
- 您不能在群組中搜尋使用者。
- 您不能依名稱來搜尋群組，但您可以列出群組。

在 AWS Management Console中建立新群組

使用下列程序以建立新的群組。

建立新群組

1. 前往 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區。
4. 選擇 Groups (群組) 索引標籤，然後選擇 Create a group (建立群組)。
5. 在 Create a group (建立群組) 頁面，Group name (群組名稱) 中，輸入新群組的易記名稱。
6. 您可以選擇性地使用下列任一欄位，提供有關此群組的其他資訊：

    - Description (描述) – 輸入有關此新群組之用途的詳細資訊。
    - Precedence (優先順序) – Amazon Cognito 根據使用者所屬的群組具有較低的優先順序值，評估並套用指定使用者的所有群組許可。系統會選擇優先順序較低的群組，並套用其相關聯的 IAM 角色。如需詳細資訊，請參閱 指定優先順序值給群組。

- IAM role (IAM 角色) – 需要控制資源的許可時，您可以將 IAM 角色指派給群組。如果您要將使用者集區與身分集區整合，IAM role (IAM 角色) 設定會決定要在使用者的 ID 權杖中指派哪個角色 (如果身分集區設定為從該權杖中選擇角色)。如需詳細資訊，請參閱 指派 IAM 角色給群組。
- Add users to this group (將使用者新增至此群組) – 建立後，將現有使用者新增為此群組的成員。

7. 選擇 Create (建立) 以確認。

管理及搜尋使用者帳戶

建立使用者集區之後，您可以使用 AWS Management Console，也可以使用 AWS Command Line Interface 或 Amazon Cognito API 來檢視並管理使用者。本主題說明如何使用 AWS Management Console來檢視及搜尋使用者。

檢視使用者屬性

使用下列處理程序，在 Amazon Cognito 主控台中檢視使用者屬性。

檢視使用者屬性

1. 前往 Amazon Cognito 主控台。若出現提示，請輸入 AWS 憑證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區。
4. 選擇 Users (使用者) 標籤，然後在清單中選取使用者。
5. 在使用者詳細資訊頁面，User attributes (使用者屬性) 下，您可以檢視與使用者相關聯的屬性。

重設使用者密碼

使用下列處理程序，在 Amazon Cognito 主控台中重設使用者密碼。

重設使用者密碼

1. 前往 Amazon Cognito 主控台。若出現提示，請輸入 AWS 憑證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區。
4. 選擇 Users (使用者) 標籤，然後在清單中選取使用者。
5. 在使用者詳細資訊頁面上，選擇 Actions (動作)、Reset password (重設密碼)。

6. 在 Reset password (重設密碼) 對話方塊中檢閱資訊，準備好時，選擇 Reset (重設)。

   此動作會導致立即傳送確認碼給使用者，並將使用者狀態變更為 `RESET_REQUIRED`，以停用使用者目前的密碼。Reset password (重設密碼) 代碼有效期為 1 小時。

搜尋使用者屬性

如果您已經建立使用者集區，即可從 中的 UsersAWS Management Console (使用者) 面板進行搜尋。您也可以使用 Amazon Cognito ListUsers API，其接受 Filter 參數。

您可以搜尋下列任何標準屬性。無法搜尋自訂屬性。

- username (區分大小寫)
- email
- phone_number
- name
- given_name
- family_name
- preferred_username
- cognito:user_status (在主控台中稱為 Status (狀態)、區分大小寫)
- status (在主控台中稱為 Enabled (已啟用)、區分大小寫)
- sub

> **Note**
>
> 您也可以使用用戶端篩選條件列出使用者。與伺服器端篩選條件相符的屬性不超過 1 個。若要執行進階搜尋，請將用戶端篩選條件與 AWS Command Line Interface 中的 `list-users` 動作 `--query` 參數搭配使用。當您使用用戶端篩選條件時，ListUsers 會傳回零個或多個使用者的分頁清單。您可以連續接收多個結果為零的頁面。使用傳回的每個分頁權杖重複查詢，直到您收到空分頁權杖值為止，然後檢閱合併的結果。
> 如需伺服器端和用戶端篩選的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的篩選 AWS CLI 輸出。

使用 AWS Management Console 來搜尋使用者

如果您已經建立使用者集區，即可從 中的 UsersAWS Management Console (使用者) 面板進行搜尋。

AWS Management Console搜尋一律為字首 (「開頭為」) 搜尋。

在 Amazon Cognito 主控台中搜尋使用者

1. 前往 Amazon Cognito 主控台。您可能會收到提示，要求您輸入 AWS 憑證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區。
4. 選擇 Users (使用者) 標籤，然後在搜尋欄位中輸入使用者的使用者名稱。請注意，某些屬性值須區分大小寫 (例如 Username (使用者名稱))。

   您也可以透過調整搜尋篩選條件來尋找使用者，將範圍縮小至其他使用者屬性，例如 Email (電子郵件)、Phone number (電話號碼) 或 Last name (姓氏)。

使用 **`ListUsers`** API 來搜尋使用者

若要從您的應用程式搜尋使用者，請使用 Amazon Cognito ListUsers API。此 API 會使用下列參數：

- `AttributesToGet`：任何字串陣列，其中每個字串都是要在搜尋結果中，為每個使用者傳回的使用者屬性名稱。若要擷取所有屬性，請勿包含 `AttributesToGet` 參數，或請求其值為常值字串 `null` 的 `AttributesToGet`。
- `Filter`：格式為「`AttributeName Filter-Type "AttributeValue"`」的篩選字串。篩選字串中的引號必須使用斜線 (\) 字元來逸出。例如 `"family_name = \"Reddy\""`。如果篩選字串是空的，`ListUsers`會傳回使用者集區中的所有使用者。
  - `AttributeName`：所要搜尋的屬性名稱。您一次只能搜尋一個屬性。

    > **Note**
    >
    > 您只能搜尋標準屬性。無法搜尋自訂屬性。這是因為只有已編製索引的屬性可供搜尋，而自訂屬性無法編製索引。

  - `Filter-Type`：若要搜尋完全相符的項目，請使用 `=`，例如，`given_name = "Jon"`。若要搜尋字首 (「開頭為」) 相符的項目，請使用 `^=`，例如，`given_name ^= "Jon"`。
  - `AttributeValue`：必須符合為每個使用者的屬性值。

- `Limit`：所要傳回的使用者數量上限。
- `PaginationToken`：此權杖可用來從先前的搜尋取得更多結果。Amazon Cognito 將在一小時後讓分頁權杖過期。
- `UserPoolId`：要對其執行搜尋之使用者集區的使用者集區 ID。

所有搜尋皆區分大小寫。搜尋結果會依 `AttributeName`字串指定的屬性遞增排序。

使用 **`ListUsers`** API 的範例

下列範例會傳回所有使用者，並包含所有屬性。

```
{
    "AttributesToGet": null,
    "Filter": "",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

下列範例會傳回電話號碼以 "+1312" 開頭的所有使用者，並包含所有屬性。

```
{
    "AttributesToGet": null,
    "Filter": "phone_number ^= \"+1312\"",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

下列範例會傳回姓氏為 "Reddy" 的前 10 個使用者。針對每個使用者，搜尋結果都會包含使用者的名字、電話號碼和電子郵件地址。如果使用者集區中有超過 10 個相符的使用者，則回應會包含分頁權杖。

```
{
    "AttributesToGet": [
        "given_name",
        "phone_number",
        "email"
```

```
    ],
    "Filter": "family_name = \"Reddy\"",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

如果上述範例傳回分頁權杖，則下列範例會傳回下 10 個符合相同篩選字串的使用者。

```
{
    "AttributesToGet": [
        "given_name",
        "phone_number",
        "email"
    ],
    "Filter": "family_name = \"Reddy\"",
    "Limit": 10,
    "PaginationToken": "pagination_token_from_previous_search",
    "UserPoolId": "us-east-1_samplepool"
}
```

復原使用者帳戶

AccountRecoverySetting 參數可讓您自訂當使用者呼叫 ForgotPassword API 來復原其密碼時所用的方法。ForgotPassword 會將復原碼傳送至已驗證過的電子郵件或已驗證過的電話號碼。此復原代碼的有效期限為一小時。當您為使用者集區指定 AccountRecoverySetting 時，Amazon Cognito 會根據您設定的優先順序選擇代碼傳遞目的地。

當您定義 AccountRecoverySetting 且使用者已設定 SMS MFA 時，SMS 將無法用於復原帳戶復原機制。此設定的優先順序是以 1 為最高優先順序，依此類推。Cognito 只會將驗證傳送給其中一個指定方法。

例如，當系統管理員不希望使用者自行復原其帳戶，而要求其聯絡系統管理員為其重設帳戶時，就會使用 admin_only 的值。您無法搭配任何其他帳戶復原機制使用 admin_only。

如果您未指定 AccountRecoverySetting，Amazon Cognito 會使用舊式機制來判斷密碼復原方法。在這種情況下，Cognito 會先使用經過驗證的手機。如果找不到該使用者的已驗證電話，Cognito 會退回，並接著使用已驗證的電子郵件。

如需 AccountRecoverySetting 的詳細資訊，請參閱 Amazon Cognito 身分供應商 API 參考中的 CreateUserPool 和 UpdateUserPool。

忘記密碼行為

在指定時段內，我們允許使用者可嘗試 5 到 20 次來要求或輸入密碼重設代碼，這是忘記密碼和確認忘記密碼時可操作的步驟。確切的值取決於與要求相關的風險參數。請注意，此行為可能會有所變更。

將使用者匯入使用者集區

有兩種方式可以將使用者從現有使用者目錄或使用者資料庫匯入或遷移至 Amazon Cognito 使用者集區。您可以在使用者第一次使用 Amazon Cognito 透過使用者遷移 Lambda 觸發器登入時，來遷移使用者。透過此方法，使用者可以繼續使用他們現有的密碼，且在遷移到您的使用者集區後，不需要重設這些密碼。或者，您可以透過上傳包含所有使用者之使用者設定檔內容的 CSV 檔來大量遷移使用者。下列章節將說明這些方法。

主題

- 透過使用者遷移 Lambda 觸發程序將使用者匯入到使用者集區
- 從 CSV 檔案將使用者匯入使用者集區

透過使用者遷移 Lambda 觸發程序將使用者匯入到使用者集區

有了此方法，當使用者首次使用您的應用程式登入或請求重設密碼時，您可以將使用者從現有使用者目錄無縫遷移至使用者集區。新增 遷移使用者 Lambda 觸發程序 功能到您的使用者集區，便會接收有關嘗試登入的使用者其相關中繼資料，並從外部身分來源傳回使用者描述檔資訊。如需此 Lambda 觸發程序範例程式碼的詳細資訊，包括請求和回應參數，請參閱遷移使用者 Lambda 觸發程序參數。

開始遷移使用者前，請在您的 AWS 帳戶 帳戶中建立使用者遷移 Lambda 函數，並將 Lambda 函數設定為使用者集區中的使用者遷移觸發程序。向您的 Lambda 函數新增授權政策，該政策僅允許 Amazon Cognito 服務帳戶主體 `cognito-idp.amazonaws.com` 叫用 Lambda 函數，且僅在您自己的使用者集區內容中。如需詳細資訊，請參閱針對 AWS Lambda 使用以資源為基礎的政策 (Lambda 函數政策)。

登入程序

1. 使用者開啟您的應用程式，並使用 Amazon Cognito 使用者集區 API 或透過 Amazon Cognito 託管的 UI 登入。如需有關如何使用 Amazon Cognito API 簡化登入的詳細資訊，請參閱 將 Amazon Cognito 身分驗證和授權與 Web 和行動應用程式整合。
2. 您的應用程式會將使用者名稱與密碼傳送到 Amazon Cognito。如果您的應用程式具有您使用 AWS 開發套件建置的自訂登入 UI，您的應用程式必須將 InitiateAuth 或 AdminInitiateAuth 搭配

USER_PASSWORD_AUTH 或 ADMIN_USER_PASSWORD_AUTH 流程使用。當您的應用程式使用這些流程之一時，開發套件會將密碼發送到伺服器。

> **Note**
>
> 在您新增使用者遷移觸發程序之前，請先啟用您應用程式用戶端設定中的 USER_PASSWORD_AUTH 或 ADMIN_USER_PASSWORD_AUTH。您必須使用這些流程，而不是預設的 USER_SRP_AUTH 流程。Amazon Cognito 必須向您的 Lambda 函數傳送密碼，以便它能夠驗證您的使用者在其他目錄中的身分驗證。SRP 會從您的 Lambda 函數中隱藏您使用者的密碼。

3. Amazon Cognito 會檢查提交的使用者名稱是否與使用者集區中的使用者名稱或別名相符。您可以將使用者的電子郵件地址、電話號碼或偏好的使用者名稱設定為使用者集區中的別名。如果使用者不存在，Amazon Cognito 會將包含使用者名稱和密碼的參數傳送到您的 遷移使用者 Lambda 觸發程序 函數。
4. 您的 遷移使用者 Lambda 觸發程序 函數會使用現有的使用者目錄或使用者資料庫，來檢查使用者或驗證其身分。該函數會傳回 Amazon Cognito 存放在使用者集區中使用者描述檔的使用者屬性。只有在已提交的使用者名稱符合別名屬性時，您才能傳回 username 參數。如果您想要使用者繼續使用現有的密碼，您的函數會將 Lambda 回應中的屬性 finalUserStatus 設定為 CONFIRMED。您的應用程式必須傳回在 遷移使用者 Lambda 觸發程序參數 顯示的所有"response" 參數。

> **Important**
>
> 請勿在使用者遷移 Lambda 程式碼中記錄整個請求事件物件。此請求事件物件包含使用者的密碼。如果您未清理日誌，密碼會出現在 CloudWatch Logs 中。

5. Amazon Cognito 在您的使用者集區中建立使用者描述檔，並將字符傳回到您的應用程式用戶端。
6. 您的應用程式會執行權杖擷取、接受使用者身分驗證，並繼續處理請求的內容。

遷移使用者後，請使用 USER_SRP_AUTH 登入。安全遠端密碼 (SRP) 通訊協定不會透過網路傳送密碼，但會提供您在遷移期間所使用 USER_PASSWORD_AUTH 流程的安全優勢。

遷移期間如果發生錯誤，包括用戶端裝置或網路問題，您的應用程式會收到來自 Amazon Cognito 使用者集區 API 的錯誤回應。發生這種情況時，Amazon Cognito 可能會或不會在您的使用者集區中建立使用者帳戶。然後，使用者應嘗試再次登入。如果重複發生登入失敗，請嘗試使用您應用程式的忘記密碼流程重設使用者密碼。

忘記密碼流程還會叫用您的 遷移使用者 Lambda 觸發程序 函數與 `UserMigration_ForgotPassword` 事件來源。由於使用者在請求重設密碼時未提交密碼，因此 Amazon Cognito 在傳送給 Lambda 函數的事件中不會包含密碼。您的函數只能查閱在現有使用者目錄中的使用者，並傳回屬性以新增至使用者集區中的使用者描述檔。在您的函數完成其調用並將回應傳回 Amazon Cognito 之後，您的使用者集區就會透過電子郵件或簡訊傳送密碼重設代碼。在您的應用程式中，提示使用者輸入確認碼和新密碼，然後在 ConfirmForgotPassword API 請求中將該資訊傳送至 Amazon Cognito。您也可以使用 Amazon Cognito 託管 UI 中內建的頁面進行忘記密碼流程。

從 CSV 檔案將使用者匯入使用者集區

您可以將使用者匯入 Amazon Cognito 使用者集區。使用者資訊是從特殊格式的 .csv 檔案匯入。匯入程序會設定所有使用者屬性的值，只有 password (密碼) 除外。不支援匯入密碼，因為安全最佳實務要求不能以純文字提供密碼，而我們不支援匯入雜湊碼。這表示您的使用者必須在第一次登入時變更密碼。因此，使用此方法匯入時，使用者會處於 RESET_REQUIRED 狀態。

您可以使用將 `Permanent` 參數設為 `true` 的 AdminSetUserPassword API 請求設定使用者密碼。

> **Note**
>
> 每個使用者的建立日期，都是該使用者匯入使用者集區中的時間。建立日期不是匯入的屬性之一。

基本步驟為：

1. 在 AWS Identity and Access Management(IAM) 主控台中建立 Amazon CloudWatch Logs 角色。
2. 建立使用者匯入 .csv 檔案。
3. 建立及執行使用者匯入任務。
4. 上傳使用者匯入 .csv 檔案。
5. 啟動及執行使用者匯入任務。
6. 使用 CloudWatch 來檢查事件日誌。
7. 需要匯入的使用者重設密碼。

主題

- 建立 CloudWatch Logs IAM 角色
- 建立使用者匯入 CSV 檔案

- 建立及執行 Amazon Cognito 使用者集區匯入任務
- 在 CloudWatch 主控台中檢視使用者集區匯入結果
- 需要匯入的使用者重設密碼

建立 CloudWatch Logs IAM 角色

如果您是使用 Amazon Cognito CLI 或 API，則您需要建立 CloudWatch IAM 角色。下列程序說明如何建立 IAM 角色，讓 Amazon Cognito 可用來將匯入工作的結果寫入 CloudWatch Logs。

> **Note**
>
> 在 Amazon Cognito 主控台中建立匯入工作時，可以同時建立 IAM 角色。當您選擇 Create a new IAM role (建立新的 IAM 角色) 時，Amazon Cognito 會自動將適當的信任政策和 IAM 政策套用至該角色。

建立 CloudWatch Logs IAM 角色以進行使用者集區匯入 (AWS CLI、API)

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 建立 AWS 服務 的新 IAM 角色。如需進一步說明，請參閱 AWS Identity and Access Management 使用者指南中的為 AWS 服務 建立角色。

   a. 當您為 Trusted entity type (信任的實體類型) 選取使用案例時，請選擇任何服務。Amazon Cognito 目前未列在服務使用案例中。

   b. 在 Add permissions (新增許可) 畫面中，選擇 Create policy (建立政策)，然後插入下列政策陳述式。將 *REGION* (區域) 取代為您使用者集區的 AWS 區域，例如 `us-east-1`。使用您的 AWS 帳戶 ID 取代 *ACCOUNT*，例如 `111122223333`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
```

```
            ],
            "Resource": [
                "arn:aws:logs:REGION:ACCOUNT:log-group:/aws/cognito/*"
            ]
        }
    ]
}
```

3. 由於您在建立角色時並未選擇 Amazon Cognito 做為受信任的實體，您現在必須手動編輯角色的信任關係。在 IAM 主控台的導覽窗格中，選擇 Roles (角色)，然後選擇建立的新角色。
4. 選擇 Trust Relationships (信任關係) 標籤。
5. 選擇 Edit trust policy (編輯信任政策)。
6. 將下列政策陳述式貼到 Edit trust policy (編輯信任政策) 中，取代任何現有的文字：

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

7. 選擇 Update policy (更新政策)。
8. 請記下角色 ARN。當您建立匯入工作時，將需要提供此 ARN。

建立使用者匯入 CSV 檔案

您必須先建立包含欲匯入使用者及其屬性的逗號分隔值 (CSV) 檔案，才能將現有使用者匯入使用者集區。從您的使用者集區中，您可以擷取含有反映使用者集區屬性結構描述之標頭的使用者匯入檔案。然後，您可以插入符合 格式化 CSV 檔案 中格式化需求的使用者資訊。

下載 CSV 檔案標頭 (主控台)

使用下列程序下載 CSV 標頭檔案。

若要下載 CSV 檔案標頭

1. 前往 Amazon Cognito 主控台。您可能會收到提示，要求您輸入 AWS 憑證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區。
4. 選擇 Users (使用者) 索引標籤。
5. 在 Import users (匯入使用者) 區段，選擇 Create an import job (建立匯入任務)。
6. 在 Upload CSV (上傳 CSV) 之下，選取 template.csv 連結並下載 CSV 檔案。

下載 CSV 檔案標頭 (AWS CLI)

若要取得正確的標頭清單，請執行下列 CLI 命令，其中 *USER_POOL_ID* 是要匯入使用者之使用者集區的使用者集區識別符：

```
aws cognito-idp get-csv-header --user-pool-id "USER_POOL_ID"
```

回應範例：

```
{
    "CSVHeader": [
        "name",
        "given_name",
        "family_name",
        "middle_name",
        "nickname",
        "preferred_username",
        "profile",
        "picture",
        "website",
        "email",
        "email_verified",
        "gender",
        "birthdate",
        "zoneinfo",
        "locale",
        "phone_number",
        "phone_number_verified",
        "address",
        "updated_at",
```

```
        "cognito:mfa_enabled",
        "cognito:username"
    ],
    "UserPoolId": "USER_POOL_ID"
}
```

格式化 CSV 檔案

下載的使用者匯入 CSV 標頭檔案，如下字串所示：它也包含您新增到使用者集區中的任何自訂屬性。

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,pi
```

編輯 CSV 檔案，以使其包含此標頭以及使用者的屬性值，並依照下列規則將其格式化：

> **Note**
>
> 如需屬性值的詳細資訊，例如電話號碼的適當格式，請參閱使用者集區屬性。

- 檔案中的第一列是所下載的標頭列，其中包含使用者屬性名稱。
- CSV 檔案中的直欄順序不重要。
- 第一列之後的每一列各包含一個使用者的屬性值。
- 標頭中的所有直欄都必須存在，但不需要在每個直欄中提供值。
- 以下是必要屬性：
  - cognito:username
  - cognito:mfa_enabled
  - email_verified 或 phone_number_verified
    - 每個使用者必須至少有其中一個自動驗證屬性為 `true`。自動驗證屬性是當新使用者加入您的使用者集區時，Amazon Cognito 自動傳送代碼的電子郵件地址或電話號碼。
    - 使用者集區必須至少有一個自動驗證的屬性，可以是 email_verified 或 phone_number_verified。如果使用者集區沒有自動驗證屬性，匯入任務將不會啟動。
    - 如果使用者集區只有一個自動驗證屬性，則必須為每個使用者驗證該屬性。例如，如果使用者集區只有 phone_number 是自動驗證屬性，則每個使用者的 phone_number_verified 值必須為 `true`。

> **Note**
>
> 為了讓使用者能夠重設密碼，他們必須要有已驗證的電子郵件或電話號碼。Amazon Cognito 會將包含重設密碼代碼的訊息傳送至 CSV 檔案中指定的電子郵件或電話號碼。如果訊息是傳送到電話號碼，則是透過 SMS 訊息傳送。如需更多詳細資訊，請參閱 註冊時驗證聯絡資訊。

  - email (如果 email_verified 為 `true`)
  - phone_number (如果 phone_number_verified 為 `true`)
  - 在您建立使用者集區時標記為必要的任何屬性
- 字串形式的屬性值不能用引號括住。
- 如果屬性值包含逗號，您必須在逗號前面放置斜線 (\)。這是因為 CSV 檔案中的欄位是以逗號分隔。
- CSV 檔案內容應該是沒有位元組順序標記的 UTF-8 格式。
- cognito:username 是必要欄位，並且在您的使用者集區中必須是唯一的。它可以是任何 Unicode 字串，但是不能包含空格或 Tab。
- 如果有 birthdate 值，其格式必須為 *`mm/dd/yyyy`*。這表示，例如生日是 1985 年 2 月 1 日，則必須編碼為 **`02/01/1985`**。
- cognito:mfa_enabled 是必要欄位。如果您在使用者集區中將多重要素驗證 (MFA) 設定為必要項目，則所有使用者的這個欄位必須為 `true`。如果您將 MFA 設為關閉，則所有使用者的這個欄位必須為 `false`。如果您將 MFA 設為選用，則這個欄位可以是 `true` 或 `false`，但不能空白。
- 列的長度上限為 16,000 個字元。
- CSV 檔案大小上限為 100 MB。
- 檔案中的列數 (使用者數) 上限為 500,000。此上限不包括標題列。
- updated_at 欄位值應該是 epoch 時間 (以秒為單位)，例如：**1471453471**。
- 屬性值中的任何首尾空格都會截去。

下列清單是沒有自訂屬性之使用者集區的 CSV 匯入檔案範例。您的使用者集區結構描述可能與此範例不同。此時您必須在從使用者集區下載的 CSV 範本中提供測試值。

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,pi
John,,John,Doe,,,,,,,johndoe@example.com,TRUE,,02/01/1985,,,+12345550100,TRUE,123 Any
 Street,,FALSE
```

```
Jane,,Jane,Roe,,,,,,,janeroe@example.com,TRUE,,01/01/1985,,,+12345550199,TRUE,100 Main
 Street,,FALSE
```

建立及執行 Amazon Cognito 使用者集區匯入任務

本節說明如何使用 Amazon Cognito 主控台和 AWS Command Line Interface (AWS CLI) 來建立及執行使用者集區匯入任務。

主題

- 從 CSV 檔案匯入使用者 (主控台)
- 匯入使用者 (AWS CLI)

從 CSV 檔案匯入使用者 (主控台)

下列程序說明如何從 CSV 檔案匯入使用者。

從 CSV 檔案匯入使用者 (主控台)

1. 前往 Amazon Cognito 主控台。您可能會收到提示，要求您輸入 AWS 憑證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區。
4. 選擇 Users (使用者) 索引標籤。
5. 在 Import users (匯入使用者) 區段，選擇 Create an import job (建立匯入任務)。
6. 在 Create import job (建立匯入任務) 頁面上，輸入 Job name (任務名稱)。
7. 選擇Create a new IAM role (建立新的 IAM 角色)，或 Use an existing IAM role (使用現有的 IAM 角色)。

   a. 如果您選擇 Create a new IAM role (建立新的 IAM 角色)，請輸入新角色的名稱。Amazon Cognito 會自動建立具有正確許可和信任關係的角色。建立匯入任務的 IAM 主體，必須具備建立 IAM 角色的許可。

   b. 如果您選擇 Use an existing IAM role (使用現有的 IAM 角色)，請從 IAM role selection (IAM 角色選擇) 下的清單中選擇角色。此角色必須具有 建立 CloudWatch Logs IAM 角色 中所述的許可和信任政策。
8. 選擇 Create job (建立任務) 以提交任務，但稍後再開始。選擇 Create and start job (建立並開始任務)，以提交並立即開始任務。

9. 如果您已建立任務但尚未開始，可以稍後再開始。在 Import users (匯入使用者) 下的 Users (使用者) 標籤中，選擇您的匯入任務，然後選取 Start (開始)。您也可以從 AWS SDK 提交 StartUserImportJob API 請求。
10. 在 Import users (匯入使用者) 下的 Users (使用者) 標籤中監控使用者匯入任務的進度。如果任務未成功，您可以選取 Status (狀態) 值。如需其他詳細資訊，請選取 View the CloudWatch logs for more details (檢視 CloudWatch 日誌以取得更多詳細資訊)，並在 CloudWatch Logs 主控台中檢視任何問題。

匯入使用者 (AWS CLI)

下列 CLI 命令可用來將使用者匯入使用者集區中：

- `create-user-import-job`
- `get-csv-header`
- `describe-user-import-job`
- `list-user-import-jobs`
- `start-user-import-job`
- `stop-user-import-job`

若要取得這些命令的命令列選項清單，請使用 `help` 命令列選項。例如：

```
aws cognito-idp get-csv-header help
```

建立使用者匯入任務

在您建立 CSV 檔案之後，請執行下列 CLI 命令來建立使用者匯入任務，其中 *JOB_NAME* 是您為任務選擇的名稱，*USER_POOL_ID* 是新使用者會新增至其中的使用者集區 ID，而 *ROLE_ARN* 是您在 建立 CloudWatch Logs IAM 角色 中收到的角色 ARN：

```
aws cognito-idp create-user-import-job --job-name "JOB_NAME" --user-pool-id
 "USER_POOL_ID" --cloud-watch-logs-role-arn "ROLE_ARN"
```

在回應中傳回的 *PRE_SIGNED_URL* 有效時間為 15 分鐘。之後就會過期，而您必須建立新的使用者匯入任務來取得新的 URL。

Example 回應範例：

```
{
    "UserImportJob": {
        "Status": "Created",
        "SkippedUsers": 0,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl": "PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

使用者匯入任務的狀態值

在對使用者匯入命令的回應中，您會看到下列其中一個 Status值：

- Created – 任務已建立但未啟動。
- Pending – 轉換狀態。您已啟動任務，但它尚未開始匯入使用者。
- InProgress – 任務已啟動，並且正在匯入使用者。
- Stopping – 您已停止任務，但該任務尚未停止匯入使用者。
- Stopped – 您已停止任務，且該任務已停止匯入使用者。
- Succeeded – 任務已成功完成。
- Failed – 由於發生錯誤，任務已停止。
- Expired – 您已建立任務，但未於 24 到 48 小時內啟動任務。與任務相關聯的所有資料都已刪除，無法啟動任務。

上傳 CSV 檔案

您可以使用下列 curl 命令，將包含使用者資料的 CSV 檔案上傳至您從 create-user-import-job 命令的回應取得的預先簽章的 URL。

```
curl -v -T "PATH_TO_CSV_FILE" -H "x-amz-server-side-encryption:aws:kms"
 "PRE_SIGNED_URL"
```

在此命令的輸出中，尋找這個句子："We are completely uploaded and fine"。這個句子指出已成功上傳檔案。

描述使用者匯入任務

若要取得使用者匯入任務的描述，請使用下列命令，其中 *USER_POOL_ID* 是您的使用者集區 ID，而 *JOB_ID* 是您建立使用者匯入任務時所傳回的任務 ID。

```
aws cognito-idp describe-user-import-job --user-pool-id "USER_POOL_ID" --job-id
 "JOB_ID"
```

Example 回應範例：

```
{
    "UserImportJob": {
        "Status": "Created",
        "SkippedUsers": 0,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl": "PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn":"ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

在上述輸出範例中，*PRE_SIGNED_URL* 是您上傳 CSV 檔案的 URL。*ROLE_ARN* 是您建立角色時，所收到的 CloudWatch Logs 角色 ARN。

列出您的使用者匯入任務

若要列出您的使用者匯入任務，請使用下列命令：

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 2
```

Example 回應範例：

```
{
    "UserImportJobs": [
        {
```

```
            "Status": "Created",
            "SkippedUsers": 0,
            "UserPoolId": "USER_POOL_ID",
            "ImportedUsers": 0,
            "JobName": "JOB_NAME",
            "JobId": "JOB_ID",
            "PreSignedUrl":"PRE_SIGNED_URL",
            "CloudWatchLogsRoleArn":"ROLE_ARN",
            "FailedUsers": 0,
            "CreationDate": 1470957431.965
        },
        {
            "CompletionDate": 1470954227.701,
            "StartDate": 1470954226.086,
            "Status": "Failed",
            "UserPoolId": "USER_POOL_ID",
            "ImportedUsers": 0,
            "SkippedUsers": 0,
            "JobName": "JOB_NAME",
            "CompletionMessage": "Too many users have failed or been skipped during the
 import.",
            "JobId": "JOB_ID",
            "PreSignedUrl":"PRE_SIGNED_URL",
            "CloudWatchLogsRoleArn":"ROLE_ARN",
            "FailedUsers": 5,
            "CreationDate": 1470953929.313
        }
    ],
    "PaginationToken": "PAGINATION_TOKEN"
}
```

任務會依時間順序列出，從最後建立排到最先建立。第二個任務之後的 *PAGINATION_TOKEN* 字串表示此列出命令還有其他結果。若要列出其他結果，請使用 --pagination-token選項，如下所示：

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 10 --
pagination-token "PAGINATION_TOKEN"
```

啟動使用者匯入任務

若要啟動使用者匯入任務，請使用下列命令：

```
aws cognito-idp start-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

每個帳戶一次只能有一個匯入任務處於作用中狀態。

Example 回應範例：

```
{
    "UserImportJob": {
        "Status": "Pending",
        "StartDate": 1470957851.483,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "SkippedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl":"PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

停止使用者匯入任務

若要在使用者匯入任務進行時將其停止，請使用下列命令。停止任務後，就無法再重新啟動。

```
aws cognito-idp stop-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

Example 回應範例：

```
{
    "UserImportJob": {
        "CompletionDate": 1470958050.571,
        "StartDate": 1470958047.797,
        "Status": "Stopped",
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "SkippedUsers": 0,
        "JobName": "JOB_NAME",
        "CompletionMessage": "The Import Job was stopped by the developer.",
        "JobId": "JOB_ID",
        "PreSignedUrl":"PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
```

```
        "CreationDate": 1470957972.387
    }
}
```

在 CloudWatch 主控台中檢視使用者集區匯入結果

您可以在 Amazon CloudWatch 主控台中檢視匯入任務的結果。

主題

- 檢視結果
- 解譯結果

檢視結果

下列步驟說明如何檢視使用者集區匯入結果。

檢視使用者集區匯入的結果

1. 登入 AWS Management Console 並開啟位於 https://console.aws.amazon.com/cloudwatch/ 的 CloudWatch 主控台。
2. 選擇 Logs (日誌)。
3. 選擇使用者集區匯入任務的日誌群組。日誌群組名稱的格式為 `/aws/cognito/userpools/`*`USER_POOL_ID`*`/`*`USER_POOL_NAME`*。
4. 選擇您剛才執行的使用者匯入任務的日誌。日誌名稱的格式為 *`JOB_ID`*`/`*`JOB_NAME`*。日誌中的結果會依行號參照您的使用者。使用者資料不會寫入日誌中。針對每個使用者，會顯示類似下列字行：

   - `[SUCCEEDED] Line Number 5956 - The import succeeded.`
   - `[SKIPPED] Line Number 5956 - The user already exists.`
   - `[FAILED] Line Number 5956 - The User Record does not set any of the auto verified attributes to true. (Example: email_verified to true).`

解譯結果

成功匯入的使用者狀態會設為 "PasswordReset"。

在下列情況下，將不會匯入使用者，但匯入任務會繼續：

- 沒有自動驗證屬性設為 `true`。

- 使用者資料不符合結構描述。
- 因為內部錯誤，無法匯入使用者。

在下列情況下，匯入任務會失敗：

- Amazon CloudWatch Logs 角色無法擔任、沒有正確的存取政策或已刪除。
- 使用者集區已刪除。
- Amazon Cognito 無法剖析 .csv 檔案。

需要匯入的使用者重設密碼

每個匯入的使用者第一次登入並輸入密碼時，都必須輸入新密碼。下列程序說明匯入 CSV 檔案之後，本機使用者在自訂應用程式中的使用者體驗。如果您的使用者使用託管 UI 登入，Amazon Cognito 會在首次登入時提示他們設定新密碼。

需要匯入的使用者重設密碼

1. 在您的應用程式中，使用隨機密碼透過 `InitiateAuth` 默默地登入目前使用者。
2. Amazon Cognito 會在啟用 `PreventUserExistenceErrors` 時傳回 `NotAuthorizedException`。否則會傳回 `PasswordResetRequiredException`。
3. 您的應用程序發出 `ForgotPassword` API 請求並重設使用者的密碼。
    a. 應用程式在 `ForgotPassword` API 請求中提交使用者名稱。
    b. Amazon Cognito 會將代碼傳送至已驗證的電子郵件或電話號碼。目的地取決於您在 CSV 檔案中為 `email_verified` 和 `phone_number_verified` 提供的值。對 `ForgotPassword` 請求的回應指示代碼的目的地。

        > **Note**
        >
        > 您的使用者集區必須經過設定，以驗證電子郵件或電話號碼。如需更多詳細資訊，請參閱 註冊及確認使用者帳戶。

    c. 您的應用程式會向使用者顯示一則訊息，以檢查代碼傳送的位置，並提示您的使用者輸入代碼和新密碼。
    d. 使用者在應用程式中輸入確認碼和新密碼。
    e. 應用程式在 `ConfirmForgotPassword` API 請求中提交代碼和新密碼。

f. 您的應用程式將使用者重新導向到登入步驟。

使用者集區屬性

屬性是協助您識別個別使用者的細項資訊，例如名稱、電子郵件地址和電話號碼。新使用者集區具有一組預設的標準屬性。您也可以將自訂屬性新增至中的使用者集區定義 AWS Management Console。此主題會詳細說明那些屬性，並提供設定使用者集區的秘訣。

請勿將所有的使用者相關資訊都存放在屬性中。例如，將經常變更的使用者資料，例如用量統計資料或遊戲分數，保存在個別的資料存放區中，例如 Amazon Cognito Sync 或 Amazon DynamoDB。

> **Note**
>
> 某些文件和標準將屬性稱為成員。

主題

標準屬性

Amazon Cognito 會根據 OpenID Connect 規格為所有使用者指派一組標準屬性。根據預設，標準和自訂屬性值可以是長度上限 2048 個字元的任何字串，但某些屬性值有格式限制。

標準屬性包括：

- `address`
- `birthdate`
- `email`
- `family_name`
- `gender`

- given_name
- locale
- middle_name
- name
- nickname
- phone_number
- picture
- preferred_username
- profile
- sub
- updated_at
- website
- zoneinfo

除了 sub，所有使用者的標準屬性預設都是選用的。若要設定為必要屬性，請在使用者集區建立程序期間，勾選屬性旁邊的 Required (必要) 核取方塊。Amazon Cognito 會為每個使用者的 sub 屬性指派唯一的使用者識別碼值。只有 email 和 phone_number 屬性能夠驗證。

> **Note**
>
> 將標準屬性標示為 Required (必要) 後，使用者必須提供該屬性的值，才能註冊。若要建立使用者而不提供必要屬性的值，管理員可以使用 AdminCreateUserAPI。建立使用者集區之後，您便無法讓屬性在必要和非必要之間切換。

標準屬性詳細資訊和格式限制

birthdate

值必須是 10 個字元的有效日期，且格式為 YYYY-MM-DD。

email

使用者和管理員可以驗證電子郵件地址值。

具有適當 AWS 帳戶 權限的管理員可以變更使用者的電子郵件地址，並將其標示為已驗證。將電子郵件地址標記為已通過 AdminUpdateUserAttributesAPI 或 admin-update-user-attributes AWS Command Line Interface (AWS CLI) 命令驗證。管理員可以使用此命令將 `email_verified` 屬性變更為 `true`。您也可以在的 [使用者] 索引標籤中編輯使用者， AWS Management Console 將電子郵件地址標示為已驗證。

值必須是遵循標準電子郵件格式的有效電子郵件地址字串，包含 @ 符號和網域，且長度上限為 2048 個字元。

phone_number

如果 SMS 多重要素驗證 (MFA) 處於作用中狀態，則使用者必須提供電話號碼。如需詳細資訊，請參閱 將 MFA 新增到使用者集區。

使用者和管理員可以驗證電話號碼值。

具有適當 AWS 帳戶 權限的管理員可以更改用戶的電話號碼，並將其標記為已驗證。將電話號碼標記為已通過 AdminUpdateUserAttributesAPI 或admin-update-user-attributes AWS CLI 命令驗證。管理員可以使用此命令將 `phone_number_verified` 屬性變更為 `true`。您也可以在的 [使用者] 索引標籤中編輯使用者， AWS Management Console 將電話號碼標示為已驗證。

> **Important**
>
> 電話號碼必須遵循下列格式規則：電話號碼必須以加號 (+) 開頭，後面緊接著國碼。電話號碼只能包含 + 號和數字。請先移除電話號碼中的任何其他字元，例如括號、空格或連字號 (-)，您才能將該值提交至服務。例如，美國地區的電話號碼必須遵循此格式：**+14325551212**。

preferred_username

您可以選取 `preferred_username` 為必填或別名，但不能同時為兩者。如果`preferred_username`是別名，您可以向 UpdateUserAttributesAPI 作業提出要求，並在確認使用者後新增屬性值。

sub

根據 `sub` 屬性建立索引與搜尋您的使用者。`sub` 屬性是每個使用者集區中的唯一使用者識別碼。使用者可以變更屬性，如 `phone_number` 和 `email`。`sub` 屬性具有固定的值。如需尋找使用者的詳細資訊，請參閱 管理及搜尋使用者帳戶。

檢視必要的屬性

使用下列處理程序檢視指定使用者集區的必要屬性。

> **Note**
>
> 建立使用者集區之後，您就無法變更必要的屬性。

檢視必要的屬性

1. 去 Amazon Cognito 在 AWS Management Console. 如果主控台提示您，請輸入您的 AWS 認證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區。
4. 選擇 Sign-up experience (註冊體驗) 標籤。
5. 在 Required attributes (必要的屬性) 區段中，檢視使用者集區的必要屬性。

使用者名稱和偏好的使用者名稱

username 值是另一個屬性，不同於 name 屬性。每個使用者都有 username 屬性。Amazon Cognito 會自動為聯合身分使用者產生使用者名稱。您必須提供 username 屬性，以便在 Amazon Cognito 目錄中建立本機使用者。在您建立使用者之後，您無法變更 username 屬性。

開發人員可以使用 preferred_username 屬性，以提供使用者可以變更的使用者名稱。如需詳細資訊，請參閱 自訂登入屬性。

如果您的應用程式不需要使用者名稱，則不需要求使用者提供。您的應用程式可以在背景為使用者建立唯一的使用者名稱。如果您希望使用者以電子郵件地址和密碼來註冊及登入，這會非常有用。如需詳細資訊，請參閱 自訂登入屬性。

username 在使用者集區中必須是唯一的。username 只有在其已遭刪除且不再使用的情況下方可重新使用。有關username屬性的字符串約束的信息，請參閱 SignUpAPI 請求的用戶名屬性。

自訂登入屬性

建立使用者集區時，如果您希望使用者以電子郵件地址或電話號碼做為使用者名稱進行註冊和登入，可以設定使用者名稱屬性。或者，您可以設定別名屬性，為使用者提供選項：他們可以在註冊時包含多個屬性，然後以使用者名稱、偏好的使用者名稱、電子郵件地址或電話號碼登入。

> **Important**
>
> 建立使用者集區之後，您就無法變更此設定。

如何在別名屬性和使用者名稱屬性之間進行選擇

您的要求	別名屬性	使用者名稱屬性
使用者擁有多個登入屬性	是 [1]	無 [2]
使用者必須先驗證電子郵件地址或電話號碼，才能登入	是	否
使用重複的電子郵件地址或電話號碼註冊用戶，防止出`UsernameExistsException` 錯 [3]	是	否
可以將相同的電子郵件地址或電話號碼屬性值分配給多個用戶	是[4]	否

[1] 可用的登入屬性為使用者名稱、電子郵件地址和偏好的使用者名稱。

[2] 可以使用電子郵件地址或電話號碼登入。

[3] 使用者使用可能重複的電子郵件地址或電話號碼註冊，但沒有使用者名稱時，使用者集區不會產生 `UsernameExistsException` 錯誤。這種行為與防止使用者名稱存在錯誤無關，它適用於登入，但不適用於註冊操作。

[4] 只有最後一位已驗證屬性的使用者才能使用該屬性登入。

選項 1：多個登入屬性 (別名屬性)

如果您想讓使用者在登入時選擇輸入其使用者名稱或其他屬性值，則可以啟動別名。根據預設，使用者會使用其使用者名稱和密碼來登入。使用者名稱是使用者無法變更的固定值。如果您將屬性標記為別名，使用者就可以使用該屬性代替使用者名稱來登入。您可以將電子郵件地址、電話號碼和偏好的使用

者名稱屬性標記為別名。例如，如果您為使用者集區選擇將電子郵件和電話號碼做為別名，該使用者集區中的使用者就可以使用其使用者名稱、電子郵件地址或電話號碼並搭配其密碼來登入。

若要選擇別名屬性，請在建立使用者集區時選取 User name (使用者名稱) 和至少另一個登入選項。

> **Note**
>
> 當您將使用者集區設定為不區分大小寫時，使用者可以使用小寫或大寫字母來註冊或使用別名登入。如需詳細資訊，請參閱 Amazon Cognito 使用者集區 API 參考CreateUserPool中的。

如果您選取電子郵件地址做為別名，Amazon Cognito 不會接受與有效電子郵件地址格式相符的使用者名稱。同樣地，如果您選擇電話號碼做為別名，Amazon Cognito 不會接受與有效電話號碼格式相符的使用者集區的使用者名稱。

> **Note**
>
> 別名值在使用者集區中必須是唯一的。如果您為電子郵件地址或電話號碼設定別名，所提供的值只能在一個帳戶中是已驗證狀態。註冊期間，如果您的使用者提供電子郵件地址或電話號碼做為別名值，而另一個使用者已使用該別名值，則註冊會成功。不過，當使用者嘗試使用此電子郵件 (或電話號碼) 來確認帳戶，並輸入有效的驗證碼時，Amazon Cognito 會傳回 `AliasExistsException` 錯誤。該錯誤會向使用者指出使用此電子郵件地址 (或電話號碼) 的帳戶已存在。此時，使用者可以放棄嘗試建立新帳戶，並改為嘗試為舊帳戶重設密碼。如果使用者繼續建立新帳戶，您的應用程式必須以 `forceAliasCreation` 選項呼叫 `ConfirmSignUp` API。具有 `forceAliasCreation` 的 `ConfirmSignUp` 會將先前的帳戶移到新建立的帳戶，並在先前的帳戶中，將該屬性標記為未驗證。

電話號碼和電子郵件地址只有在經過使用者驗證後，才會成為使用者的作用中別名。如果要使用電子郵件地址和電話號碼做為別名，建議您選擇自動驗證。

選擇別名屬性，以防止使用者註冊時發生電子郵件地址和電話號碼屬性 `UsernameExistsException` 錯誤。

啟用 `preferred_username` 屬性，以便您的使用者可以變更用於登入的使用者名稱，而 `username` 屬性值不會變更。如果您想要設定此使用者體驗，請提交新的 `username`值做為 `preferred_username`，並選擇 `preferred_username` 做為別名。如此一來，使用者就可以使用其輸入的新值來登入。如果您選取 `preferred_username` 做為別名，則使用者只有在確認帳戶後，才能提供該值。使用者在註冊期間無法提供該值。

當使用者以使用者名稱進行註冊時，您可以選擇使用者是否可以使用下列一或多個別名來登入。

- 已驗證的電子郵件地址
- 已驗證的電話號碼
- 偏好的使用者名稱

使用者完成註冊之後，便可以變更這些別名。

> **Important**
>
> 在您的使用者集區支援使用別名登入，且您想授權或查詢使用者時，請勿透過使用者的任何登入屬性來識別您的使用者。固定值使用者識別符 sub 是使用者身份的唯一一致指標。

當您建立使用者集區時，請包括下列步驟，以便使用者可以使用別名登入。

設定使用者集區讓使用者能夠使用偏好的使用者名稱登入

1. 前往 AWS Management Console中的 Amazon Cognito。如果主控台提示您，請輸入您的 AWS 認證。
2. 選擇 User Pools (使用者集區)。
3. 在頁面右上角，選擇 Create a user pool (建立使用者集區)，以開始建立使用者集區精靈。
4. Configure sign-in experience (設定登入體驗) 中，選擇您想要與使用者集區建立關聯的身分 Provider types (供應商類型)。
5. 在 Cognito user pool sign-in options (Cognito 使用者集區登入選項) 下，選擇 User name (使用者名稱)、Email (電子郵件) 和 Phone number (電話號碼) 的任意組合。
6. 在 使用者名稱要求 下，選擇 允許使用以偏好的使用者名稱登入，讓您的使用者可以設定登入時使用的替代使用者名稱。
7. 選擇 Next (下一步)，然後完成精靈中的所有步驟。

選項 2：以電子郵件地址或電話號碼作為登入屬性 (使用者名稱屬性)

當使用者以電子郵件地址或電話號碼做為使用者名稱進行註冊時，您可以選擇使用者是否可以僅使用電子郵件地址、僅電話號碼或兩者之一來註冊。

若要選擇使用者名稱屬性，請勿在建立使用者集區時選取 使用者名稱 做為登入選項。

電子郵件地址或電話號碼必須是唯一的，而且不能是已由其他使用者所使用。這不需要驗證。使用者以電子郵件地址或電話號碼註冊之後，即無法以相同的電子郵件地址或電話號碼來建立新帳戶。使用者只能重複使用現有的帳戶，並依需要重設帳戶密碼。不過，使用者可以將電子郵件地址或電話號碼變更為新的電子郵件地址或電話號碼。如果電子郵件地址或電話號碼尚未使用，就會變成新的使用者名稱。

> **Note**
>
> 如果使用者以電子郵件地址做為使用者名稱來註冊，使用者可以將使用者名稱變更為另一個電子郵件地址，但不能將其變更為電話號碼。如果使用者以電話號碼來註冊，則可以將使用者名稱變更為另一個電話號碼，但不能將其變更為電子郵件地址。

在使用者集區建立程序期間使用下列步驟，以設定使用電子郵件地址或電話號碼來註冊及登入。

將使用者集區設定為以電子郵件地址或電話號碼來註冊及登入

1. 前往 AWS Management Console中的 Amazon Cognito。如果主控台提示您，請輸入您的 AWS 認證。
2. 選擇 User Pools (使用者集區)。
3. 在頁面右上角，選擇 Create a user pool (建立使用者集區)，以開始建立使用者集區精靈。
4. 在 Cognito user pool sign-in options (Cognito 使用者集區登入選項) 下，選擇 Email (電子郵件) 和 Phone number (電話號碼) 的任意組合，組合代表使用者可用於登入的屬性。
5. 選擇 Next (下一步)，然後完成精靈中剩餘的步驟。

> **Note**
>
> 您不需要將電子郵件地址或電話號碼標記為使用者集區的必要屬性。

在您的應用程式中實作選項 2

1. 呼叫 `CreateUserPool` API 以建立您的使用者集區。將 `UserNameAttributes` 參數設定為 `phone_number`、`email` 或 `phone_number | email`。
2. 呼叫 `SignUp` API，並且在 API 的 `username` 參數中傳遞電子郵件地址或電話號碼。此 API 會執行下列作業：

- 如果 username 字串是有效的電子郵件地址格式，使用者集區就會自動在使用者的 email 屬性中填入 username 值。
- 如果 username 字串是有效的電話號碼格式，使用者集區會自動在使用者的 phone_number 屬性中填入 username 值。
- 如果 username 字串格式不是電子郵地址或電話號碼格式，SignUp API 會傳回例外狀況。
- SignUp API 會為您的使用者產生持久性 UUID，並且在內部使用它來做為不可變的使用者名稱屬性。此 UUID 的值與使用者身分字符中的 sub 宣告相同。
- 如果 username 字串中包含已使用中的電子郵件地址或電話號碼，SignUp API 會傳回例外狀況。

您可以使用電子郵件地址或電話號碼做為別名，以取代所有 API (ListUsers API 除外) 中的使用者名稱。當您呼叫 ListUsers 時，您可以依 email 或 phone_number 屬性搜尋。如果您依 username 搜尋，您必須提供真正的使用者名稱，而不是別名。

自訂屬性

您可以新增最多 50 個自訂屬性到您的使用者集區。您可以指定自訂屬性的長度下限及/或上限。不過，任何自訂屬性的長度上限不能超過 2048 個字元。

每個自訂屬性具有下列特性：

- 您可以將它義為字串或數字。Amazon Cognito 僅將自訂屬性值做為字串寫入 ID 權杖。
- 您不能要求使用者為屬性提供值。
- 將它新增至使用者集區之後，便無法移除或變更它。
- 屬性名稱的字元長度在 Amazon Cognito 接受的限制範圍內。如需詳細資訊，請參閱 Amazon Cognito 的配額。
- 可以是可變或不可變。建立使用者時，您只能將值寫入不可變屬性中。如果您的應用程式用戶端對該屬性具有寫入許可，則可以變更可變屬性的值。如需詳細資訊，請參閱屬性許可和範圍。

> **Note**
>
> 在您的程式碼以及 使用以角色為基礎的存取控制 的規則設定中，自訂屬性需要 custom: 字首來將其與標準屬性區分。

您也可以在的屬性中建立使用者集區時新增開發人員*SchemaAttributes*屬性CreateUserPool。開發人員屬性帶有 dev: 字首。您只能使用 AWS 認證修改使用者的開發人員屬性。開發人員屬性是舊版功能，Amazon Cognito 已將它取代為應用程式用戶端讀寫許可。

請使用下列處理程序建立新的自訂屬性。

使用主控台新增自訂屬性

1. 去 Amazon Cognito 在 AWS Management Console. 如果主控台提示您，請輸入您的 AWS 認證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區。
4. 選擇 Sign-up experience (註冊體驗) 索引標籤，然後在 Custom attributes (自訂屬性) 區段中，選擇 Add custom attributes (新增自訂屬性)。
5. 在 Add custom attributes (新增自訂屬性) 頁面中，提供下列有關新屬性的詳細資訊：
   - 輸入 Name (名稱)。
   - 選取 Type (類型) 為 String (字串) 或 Number (數字)。
   - 輸入 Min (最小) 字串長度或數字值。
   - 輸入 Max (最大) 字串長度或數字值。
   - 如果您想要許可使用者在設定初始值之後，可以變更自訂屬性的值，請選擇 Mutable (可變)。
6. 選擇儲存變更。

屬性許可和範圍

針對每個應用程式用戶端，您可以設定每個使用者屬性的讀取和寫入許可。透過這種方式，您可以控制任何應用程式所擁有讀取和修改您為使用者所存放每個屬性的存取權。例如，您有一個自訂屬性指出使用者是否為付費客戶。您的應用程式可能可以看到此屬性，但不能直接變更。反之，您會使用管理工具或背景處理程序來更新此屬性。您可以從 Amazon Cognito 主控台、Amazon Cognito API 或 AWS CLI 設定使用者屬性的許可。預設情況下，除非您設定自訂屬性的讀取和寫入許可，否則任何新的自訂屬性都無法使用。默認情況下，您創建一個新的應用程序客戶端，您授予所有標準和自定義屬性的應用程序讀取和寫入權限。若要限制應用程式只能擁有所需的資訊量，請在應用程式用戶端組態中指派特定許可給屬性。

最佳作法是在建立應用程式用戶端時指定屬性讀取和寫入權限。授予您的應用程序客戶端對應用程序操作所需的最小用戶屬性集的訪問權限。

> **Note**
>
> DescribeUserPoolClient僅在設定預`ReadAttributes`設值以外的應用程式用戶端權限`WriteAttributes`時傳回值。

若要更新屬性許可 (AWS Management Console)

1. 去 Amazon Cognito 在 AWS Management Console. 如果主控台提示您，請輸入您的 AWS 認證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區。
4. 選擇 App integration (應用程式整合) 索引標籤，然後在 App clients (應用程式用戶端) 區段中，從清單中選擇應用程式用戶端。
5. 在 Attribute read and write permissions (屬性讀取和寫入許可) 區段中，選擇 Edit (編輯)。
6. 在 Edit attribute read and write permissions (編輯屬性讀取和寫入許可) 頁面設定您的讀取和寫入許可，然後選擇 Save changes (儲存變更)。

對使用自訂屬性的每個應用程式用戶端重複以上步驟。

您可以針對每個應用程式，將屬性標記為可讀取或可寫入。同樣適用於標準和自訂屬性。您的應用程式可以擷取標記為可讀取之屬性的值，而且可以設定或修改標記為可寫入之屬性的值。如果您的應用程式嘗試為未授權寫入的屬性設定值，則 Amazon Cognito 會傳回 `NotAuthorizedException`。GetUser請求包含具有應用程式用戶端宣告的存取權杖；Amazon Cognito 只會傳回應用程式用戶端可讀取的屬性值。來自應用程式的使用者 ID 權杖僅包含與可讀屬性對應的宣告。所有應用程式用戶端都可以寫入使用者集區所需的屬性。只有在您同時為任何尚未具有值的必要屬性提供值時，您才能在 Amazon Cognito 使用者集區 API 請求中設定屬性的值。

自訂屬性對於讀取和寫入許可擁有不同的功能。您可針對使用者集區建立可變或不可變的屬性，也可以將其設定為任何應用程式用戶端的讀取或寫入屬性。

不可變的自訂屬性可在使用者建立期間更新一次。您可以使用以下方法填入不可變屬性。

- `SignUp`：使用者在具有不可變自訂屬性的寫入存取權的應用程式用戶端註冊。使用者會為該屬性提供值。
- 使用第三方 IdP 登入：使用者在具有不可變自訂屬性的寫入存取權的應用程式用戶端登入。您對於其 IdP 的使用者集區組態設有一項規則，會將提供的宣告映射至不可變屬性。
- `AdminCreateUser`：您為不可變屬性提供值。

如需可指派至應用程式用戶端的範圍相關資訊，請參閱 使用資源伺服器進行範圍、M2M 和 API 授權。

您可以在使用者集區建立之後，變更屬性的許可和範圍。

新增使用者集區密碼要求

強而複雜的密碼是您使用者集區的安全性最佳做法。特別是在對 Internet 開放的應用程序中，弱密碼可能會將用戶的憑據暴露給猜測密碼並嘗試訪問數據的系統。密碼越複雜，猜測就越困難。Amazon Cognito 為注重安全性的管理員提供額外的工具，例如進階安全功能和 AWS WAFWeb ACL，但您的密碼政策是使用者目錄安全性的核心元素。

Amazon Cognito 使用者集區中本機使用者的密碼不會自動過期。最佳作法是記錄外部系統中使用者密碼重設的時間、日期和中繼資料。透過外部密碼保留日誌，您的應用程式或 Lambda 觸發程序可以查詢使用者的密碼有效期限，並在指定期間後要求重設。

您可以將使用者集區設定為要求符合安全性標準的最低密碼複雜性。複雜密碼的長度下限至少為八個字元。它們還包括大寫，數字和特殊字符的混合。

設定使用者集區密碼

1. 建立使用者集區並導覽至設定安全性需求步驟，或存取現有的使用者集區並導覽至登入體驗索引標籤。
2. 瀏覽至密碼政策。
3. 選擇一個密碼政策模式。Cognito 預設會使用建議的最低設定值來設定您的使用者集區。您也可以選擇自訂密碼政策。
4. 設定密碼長度下限。所有使用者都必須使用長度大於或等於此值的密碼來註冊或建立。您可以將此最小值設定為 99，但您的使用者最多可以設定 256 個字元的密碼。
5. 在密碼要求下設定密碼複雜性規則。選擇每個使用者密碼中至少要有一個的字元類型 (數字、特殊字元、大寫字母和小寫字母)。

   您可以在密碼中至少要求下列其中一個字元。Amazon Cognito 驗證密碼是否包含最少必要字元後，您的使用者密碼可以包含任何類型的其他字元，直到密碼長度上限為止。

   - 大小寫的基本拉丁語字母
   - 數字
   - 下列特殊字元。

     ```
     ^ $ * . [ ] { } ( ) ? " ! @ # % & / \ , > < ' : ; | _ ~ ` = + -
     ```

- 非位於開頭及結尾的空格字元。

6. 為管理員設定的臨時密碼過期時間設定一個值。在此期限後，您在 Amazon Cognito 主控台或使用 `AdminCreateUser` 建立的新使用者將無法登入並設定新密碼。當他們用臨時密碼登入後，其使用者帳戶就永遠不會過期。若要更新 Amazon Cognito 使用者集區 API 中的密碼持續時間，請TemporaryPasswordValidityDays 在您的CreateUserPool或 UpdateUserPoolAPI 請求中設定的值。

   - 若要重設過期使用者帳戶的存取權，請執行下列其中一項動作。

     - 刪除使用者描述檔並建立新的描述檔。
     - 在 AdminSetUserPasswordAPI 請求中設置新的永久密碼。
     - 在 AdminResetUserPasswordAPI 請求中生成新的確認代碼。

Amazon Cognito 使用者集區的電子郵件設定

使用者集區用戶端應用程式中的特定事件，會導致 Amazon Cognito 向您的使用者傳送電子郵件。例如，如果您將您的使用者集區設定成需要電子郵件驗證，則當使用者註冊您的應用程式以取得新帳戶，或是重新設定其密碼時，Amazon Cognito 就會傳送電子郵件。視發出電子郵件的動作而定，此電子郵件將包含驗證碼或臨時密碼。

若要處理電子郵件傳送，您可以使用下列任一選項：

- Amazon Cognito 服務內建的預設電子郵件組態。
- 您的 Amazon Simple Email Service (Amazon SES) 組態。

您可以在建立使用者集區之後變更傳遞選項。

Amazon Cognito 會傳送電子郵件訊息給您的使用者，其中包含他們可以輸入的代碼或他們可以選取的 URL 連結。下表顯示可產生電子郵件訊息的事件。

訊息選項

活動	API 操作	傳遞選項	格式選項	可自訂	訊息範本
忘記密碼	ForgotPassword	電子郵件，短信	code	否	N/A

活動	API 操作	傳遞選項	格式選項	可自訂	訊息範本
邀請	AdminCreateUser	電子郵件，短信	code	是	邀請訊息
自助登記	SignUp	電子郵件，短信	代碼，鏈接	是	驗證訊息
電子郵件地址或電話號碼驗證	UpdateUserAttributes	電子郵件，短信	code	是	驗證訊息
多重要素驗證 (MFA)	AdminInitiateAuth, InitiateAuth	簡訊	code	是 [1]	MFA 訊息

[1] 適用簡訊。

Amazon SES 會針對電子郵件訊息收費。如需詳細資訊，請參閱 Amazon SES 定價。

預設的電子郵件

Amazon Cognito 可以使用其預設的電子郵件組態為您處理電子郵件傳送。使用預設選項時，Amazon Cognito 會限制每天為使用者集區寄送的電子郵件數量。如需服務配額的資訊，請參閱 Amazon Cognito 的配額。處理一般生產環境時，預設的電子郵件限制低於需要的傳送數量。若要啟用更多的傳送數量，可以使用 Amazon SES 電子郵件組態。

使用預設組態時，您可以使用受管理的 Amazon SES 資源 AWS 來傳送電子郵件訊息。Amazon SES 會將傳回硬退信的電子郵件地址新增至帳戶層級禁止名單或全域禁止名單。如果無法傳遞的電子郵件地址稍後變成可傳送作業，則當您的使用者集區設定為使用預設組態時，您無法控制該地址從隱藏清單中移除。電子郵件地址可以無限期地保留在 AWS-managed 的隱藏清單中。若要管理無法傳遞的電子郵件地址，請將 Amazon SES 電子郵件組態搭配帳戶層級禁止名單使用，如下一節所述。

使用預設的電子郵件組態時，您可以使用下列任一個電子郵件地址做為 FROM 地址：

- 預設的電子郵件地址是 no-reply@verificationemail.com。
- 自訂的電子郵件地址。在您可以使用自己的電子郵件地址之前，您必須先以 Amazon SES 驗證該地址，並授予 Amazon Cognito 使用該地址的許可。

Amazon SES 電子郵件組態

您的應用程式可能需要超過預設選項可用的更高傳送數量。若要增加可用的傳送數量，請搭配使用 Amazon SES 資源與您的使用者集區，向您的使用者寄送電子郵件。當您使用自己的 Amazon SES 組態傳送電子郵件訊息時，您也可以監控您的電子郵件傳送活動。

您必須向 Amazon SES 驗證一或多個電子郵件地址或網域後，才能使用您的 Amazon SES 組態。使用已驗證的電子郵件地址，或來自已驗證網域的地址，做為您指派至使用者集區的 FROM 電子郵件地址。當 Amazon Cognito 傳送電子郵件給使用者時，它會為您呼叫 Amazon SES，並使用您的電子郵件地址。

當您使用 Amazon SES 組態時，則適用以下條件：

- 您使用者集區的電子郵件傳輸限制就是套用至您 AWS 帳戶中 Amazon SES 驗證電子郵件地址的相同限制。
- 您可以使用 Amazon SES 中的帳戶層級禁止名單，來管理傳送到無法傳遞的電子郵件地址的訊息；該名單會覆寫全域禁止名單。當您使用帳戶層級禁止名單時，電子郵件退信會影響您的帳戶做為寄件者的評價。如需詳細資訊，請參閱《Amazon Simple Email Service 開發人員指南》中的使用 Amazon SES 帳戶層級禁止名單。

Amazon SES 電子郵件組態區域

建立使用者集區的 AWS 區域 位置對於使用 Amazon SES 設定電子郵件訊息的三項需求之一。您可以從 Amazon SES 傳送電子郵件訊息與使用者集區相同的區域、包括相同區域的多個區域，或是一或多個遠端區域傳送電子郵件訊息。若要取得最佳效能，請在您選擇使用者集區的相同區域中傳送具有 Amazon SES 驗證身分的電子郵件訊息。

經過 Amazon SES 驗證身分的區域需求類別

僅限區域內

> 您的使用者集區可以在與使用者集區相同 AWS 區域 的位置傳送具有已驗證身分的電子郵件。在沒有自訂電子郵件地址的預設FROM電子郵件組態中，Amazon Cognito 會在相同區域中使用no-reply@verificationemail.com經過驗證的身分。

向下相容

> 您的使用者集區可以在相同 AWS 區域 或下列其中一個替代區域中傳送具有已驗證身分的電子郵件訊息：

- 美國東部 (維吉尼亞北部)
- 美國西部 (奧勒岡)
- 歐洲 (愛爾蘭)

此功能支援您在服務啟動時可能為符合 Amazon Cognito 要求而建立的使用者集區資源的連續性。該期間的使用者集區只能傳送數量有限的已驗證身分的電子郵件訊息 AWS 區域。在沒有自訂電子郵件地址的預設FROM電子郵件組態中，Amazon Cognito 會在相同區域中使用no-reply@verificationemail.com經過驗證的身分。

替代區域

您的使用者集區可以在使用者集區區域之外的替代 AWS 區域 使用者集區中傳送具有已驗證身分的電子郵件訊息。當 Amazon SES 無法在可使用 Amazon Cognito 的區域中使用時，就會發生此組態。

Amazon SES 在替代區域中針對已驗證身分傳送授權政策，必須信任原始區域的 Amazon Cognito 服務主體。如需詳細資訊，請參閱 授與使用預設電子郵件組態的權限。

在這些區域中，Amazon Cognito 會將電子郵件訊息分割為預設電子郵件組態的兩個替代區域。COGNITO_DEFAULT在這些情況下，若要使用自訂FROM電子郵件地址，在每個替代區域中針對已驗證身分的 Amazon SES 傳送授權政策必須信任原始區域的 Amazon Cognito 服務主體。如需詳細資訊，請參閱 授與使用預設電子郵件組態的權限。使用這些區域中的 Amazon SES 電子郵件組態，您必須DEVELOPER在第一個列出的區域中使用經過驗證的身分，並將其設定為信任使用者集區區域中的 Amazon Cognito 服務主體。例如，在中東 (阿拉伯聯合大公國) 的使用者集區中，在歐洲 (法蘭克福) 設定要信任的已驗證身分cognito-idp.me-central-1.amazonaws.com。在沒有自訂電子郵件地址的預設FROM電子郵件組態中，Amazon Cognito 會在每個區域使用no-reply@verificationemail.com經過驗證的身分。

> **Note**
>
> 在下列條件組合下，您必須在 Region 元素中以萬用字元的EmailConfiguration格式指定SourceArn參數arn:*${Partition}*:ses:*:*${Account}*:identity/*${IdentityName}*。這允許您的使用者集區在兩者中傳送具有相同驗證身分的電 AWS 帳戶 子郵件訊息 AWS 區域。
>
> - 你 EmailSendingAccount 是COGNITO_DEFAULT。
> - 您想要使用自訂FROM位址。
> - 您的使用者集區會在替代區域傳送電子郵件。

- 您的使用者集區具有 Amazon SES 支援區域表格中指定的第二個[1]替代區域，如下所示。

如果您以程式設計方式建立使用者集區 AWS CloudFormation(使用 AWS 開發套件、Amazon Cognito API 或 CLI、或)，您的使用者集區會傳送含有`SourceArn`參數為您的使用者集區指定的 Amazon SES 身分的電子郵件訊息。 AWS CDKEmailConfigurationAmazon SES 身份必須佔用一個支持的 AWS 區域。如果您的 `EmailSendingAccount` 是 `COGNITO_DEFAULT` 並且您未指定 `SourceArn` 參數，Amazon Cognito 會使用您用於建立使用者集區的區域中的資源，以 `no-reply@verificationemail.com` 傳送電子郵件。

下表顯示您可以將 Amazon SES 身分與 Amazon Cognito 搭配使用的 AWS 區域 位置。

使用者集區區域	區域選項	支援 Amazon SES 區域
美國東部 (維吉尼亞北部)	向下相容	美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、歐洲 (愛爾蘭)
美國東部 (俄亥俄)	向下相容	美國東部 (俄亥俄)、美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、歐洲 (愛爾蘭)
美國西部 (加利佛尼亞北部)	僅限區域內	美國西部 (加利佛尼亞北部)
美國西部 (奧勒岡)	向下相容	美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、歐洲 (愛爾蘭)
加拿大 (中部)	向下相容	加拿大 (中部)、美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、歐洲 (愛爾蘭)
亞太區域 (東京)	向下相容	亞太區域 (東京)、美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、歐洲 (愛爾蘭)

使用者集區區域	區域選項	支援 Amazon SES 區域
亞太區域 (首爾)	向下相容	亞太區域 (首爾)、美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、歐洲 (愛爾蘭)
亞太區域 (孟買)	向下相容	亞太區域 (孟買)、美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、歐洲 (愛爾蘭)
亞太區域 (海德拉巴)	替代區域	亞太區域 (孟買)、亞太區域 (新加坡) [1]
亞太區域 (新加坡)	向下相容	亞太區域 (新加坡)、美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、歐洲 (愛爾蘭)
亞太區域 (悉尼)	向下相容	亞太區域 (雪梨)、美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、歐洲 (愛爾蘭)
亞太區域 (大阪)	僅限區域內	亞太區域 (大阪)
亞太區域 (雅加達)	僅限區域內	亞太區域 (雅加達)
亞太區域 (墨爾本)	替代區域	亞太區域 (雪梨)、亞太區域 (新加坡) [1]
歐洲 (愛爾蘭)	向下相容	美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、歐洲 (愛爾蘭)
歐洲 (倫敦)	向下相容	歐洲 (倫敦)、美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、歐洲 (愛爾蘭)
Europe (Paris)	僅限區域內	Europe (Paris)

使用者集區區域	區域選項	支援 Amazon SES 區域
歐洲 (法蘭克福)	向下相容	歐洲 (法蘭克福)、美國東部 (維吉尼亞北部)、美國西部 (奧勒岡)、歐洲 (愛爾蘭)
歐洲 (蘇黎世)	替代區域	歐洲 (法蘭克福)、歐洲 (倫敦) [1]
歐洲 (斯德哥爾摩)	僅限區域內	歐洲 (斯德哥爾摩)
歐洲 (米蘭)	僅限區域內	歐洲 (米蘭)
歐洲 (西班牙)	替代區域	歐洲 (巴黎)、歐洲 (斯德哥爾摩) [1]
Middle East (Bahrain)	僅限區域內	Middle East (Bahrain)
中東 (阿拉伯聯合大公國)	替代區域	歐洲 (法蘭克福)、歐洲 (倫敦) [1]
南美洲 (聖保羅)	僅限區域內	南美洲 (聖保羅)
以色列 (特拉維夫)	僅限區域內	以色列 (特拉維夫)
非洲 (開普敦)	僅限區域內	非洲 (開普敦)

[1] 用於具有預設電子郵件組態的使用者集區。Amazon Cognito 會在每個區域使用相同的電子郵件地址，在經過驗證的身分之間分發電子郵件。若要使用自訂FROM位址，請使EmailConfiguration用格式的SourceArn參數進行設定arn:*${Partition}*:ses:*:*${Account}*:identity/*${IdentityName}*。

為您的使用者集區設定電子郵件

完成下列步驟，為您的使用者集區設定電子郵件設定。根據您使用的設定，您可能需要 Amazon SES、 AWS Identity and Access Management (IAM) 和 Amazon Cognito 中的 IAM 許可。

> **Note**
>
> 您透過這些步驟所建立的資源，無法在這些 AWS 帳戶之間共用。例如，您無法在一個帳戶中設定使用者集區，然後將之與不同帳戶的 Amazon SES 電子郵件地址搭配使用。如果您的多個帳戶都使用 Amazon Cognito，請為這些帳戶個別重複這些步驟。

步驟 1：向 Amazon SES 驗證您的電子郵件地址或網域

在設定您的使用者集區之前，如果您想執行下列其中一項動作，則您必須向 Amazon SES 驗證一或多個網域或電子郵件地址：

- 使用您自己的電子郵件地址做為 FROM 地址
- 使用您的 Amazon SES 組態來處理電子郵件傳輸

驗證過電子郵件地址或網域之後，您就能確認您擁有該地址，協助防止未經授權的使用。

如需使用 Amazon SES 驗證電子郵件地址的資訊，請參閱《Amazon Simple Email Service 開發人員指南》中的驗證電子郵件地址。如需使用 Amazon SES 驗證網域的相關資訊，請參閱驗證網域。

步驟 2：將您的帳戶移出 Amazon SES 沙盒

如果您使用的是預設的 Amazon Cognito 電子郵件組態，請省略此步驟。

當您第一次在任何地方使用 Amazon SES 時 AWS 區域，它會將您放置 AWS 帳戶 在該區域的 Amazon SES 沙箱中。Amazon SES 使用沙盒來防止詐騙和濫用行為。如果您使用 Amazon SES 組態處理電子郵件傳輸，您必須先將您的 AWS 帳戶 移出沙盒，Amazon Cognito 才能向您的使用者寄送電子郵件。

在沙盒中，Amazon SES 會針對您可寄送的電子郵件數量及從何處寄送，強制套用限制。您只能傳送電子郵件到您已向 Amazon SES 驗證的地址及網域，或將電子郵件傳送到 Amazon SES 信箱模擬器地址。當您 AWS 帳戶 保留在沙箱中時，請勿將 Amazon SES 組態用於正在生產中的應用程式。在這種情況下，Amazon Cognito 無法將訊息傳送到使用者的電子郵件地址。

若要 AWS 帳戶 從沙箱中移除，請參閱 Amazon 簡單電子郵件服務開發人員指南中的移出 Amazon SES 沙箱。

步驟 3：授予 Amazon Cognito 電子郵件許可

您可能需要為 Amazon Cognito 授予特定許可，其才能向您的使用者寄送電子郵件。您授與的許可以以及用來授予它們的程序，取決於您使用的是預設電子郵件組態還是 Amazon SES 組態。

授與使用預設電子郵件組態的權限

僅當您將使用者集區設定為使用 Cognito 傳送電子郵件或設**`EmailSendingAccount`**定為時，才完成此步驟。`COGNITO_DEFAULT`

使用預設電子郵件設定時，您的使用者集區可以使用下列其中一個地址傳送電子郵件訊息。

- 預設位址`no-reply@verificationemail.com`。
- 經過驗證的電子郵件地址或 Amazon SES 網域中的自訂寄件者地址。

如果您使用自訂地址，則 Amazon Cognito 會要求額外的許可，以便其使用此地址向您的使用者傳送電子郵件。這些許可由 Amazon SES 中的地址或網域的傳送授權政策授予。如果您要使用 Amazon Cognito 主控台來將自訂地址新增到您的使用者集區，則政策會自動連接到 Amazon SES 驗證過的電子郵件地址。但是，如果您在主控台外部設定使用者集區 (例如使用 AWS CLI 或 Amazon Cognito API)，則必須使用 Amazon SES 主控台或 PutIdentityPolicyAPI 附加政策。

> **Note**
>
> 您只能在已驗證的網域中使用 AWS CLI 或 Amazon Cognito API 設定 FROM (寄件者) 地址。

根據使用 Amazon Cognito 叫用 Amazon SES 的帳戶資源，傳送授權政策允許或拒絕存取。如需以資源為基礎之政策的詳細資訊，請參閱《IAM 使用者指南》。您也可以在《Amazon SES 開發人員指南》中找到以資源為基礎的範例政策。

Example 傳送授權政策

下列範例傳送授權政策會向 Amazon Cognito 授予使用 Amazon SES 已驗證身分的有限能力。執行此操作時，Amazon Cognito 只能代表 `aws:SourceArn` 條件中的使用者集區和 `aws:SourceAccount` 條件中的帳戶傳送電子郵件訊息。

Regions with Amazon SES

您在使用者集區區域或替代區域中的傳送授權政策必須允許 Amazon Cognito 服務主體傳送電子郵件訊息。如需詳細資訊，請參閱「區域」表格。如果您的使用者集區區域與 Amazon SES 區域中的至少一個值相符，請在下列範例中使用全域服務主體設定傳送授權政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "stmnt1234567891234",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "email.cognito-idp.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "aws:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
}
```

Opt-in Regions without Amazon SES

Amazon SES 並非在所有選擇加入 Amazon Cognito 可用的 AWS 區域 地方都可用。中東（阿拉伯聯合大公國）就是一個例子，只能在歐洲（法蘭克福）（ ）發送具有驗證身份的電子郵件（eu-central-1）。在具有預設電子郵件組態的使用者集區中，Amazon Cognito 也會在兩個區域中的每個區域傳送具有已驗證身分的電子郵件訊息。在中東（阿聯酋）的情況下，額外的區域是歐洲（倫敦）。您必須更新這兩個區域中的傳送授權原則。

您在每個替代區域中的傳送授權政策必須允許使用者集區選擇加入區域中的 Amazon Cognito 服務主體傳送電子郵件訊息。如需詳細資訊，請參閱「區域」表格。如果您的區域標示為替代區域，請使用區域服務主體設定傳送授權原則，如下列範例所示。視需要將範例區域識別碼 *me-central-1* 取代為所需的區域 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "cognito-idp.me-central-1.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "aws:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
}
```

如需政策語法的詳細資訊，請參閱《Amazon Simple Email Service 開發人員指南》中的 Amazon SES 傳送授權政策。

如需更多範例，請參閱《Amazon Simple Email Service 開發人員指南》中的 Amazon SES 傳送授權政策範例。

授予許可來使用您的 Amazon SES 組態

如果您將您的使用者集區設定為使用您的 Amazon SES 組態，則 Amazon Cognito 會要求額外許可，以便其在向您的使用者寄送電子郵件時，代表您呼叫 Amazon SES。這份授權是透過 IAM 服務授予。

當您使用此選項設定您的使用者集區時，Amazon Cognito 會建立服務連結的角色，這是您 AWS 帳戶中的 IAM 角色類型。此角色包含的許可，能讓 Amazon Cognito 存取 Amazon SES，以及使用您的地址傳送電子郵件。

Amazon Cognito 會使用設定組態的使用者工作階段 AWS 登入資料來建立您的服務連結角色。此工作階段的 IAM 許可必須包含 `iam:CreateServiceLinkedRole` 動作。如需 IAM 中許可的詳細資訊，請參閱 IAM 使用者指南中的 AWS 資源存取管理。

如需 Amazon Cognito 所建立服務連結角色的詳細資訊，請參閱使用 Amazon Cognito 的服務連結角色。

步驟 4：設定您的使用者集區

如果您需要將您的使用者集區設定為搭配下面任何一種設定，請完成下面步驟：

- 顯示為電子郵件寄件者的自訂 FROM 地址
- 在您的使用者傳送訊息至您的 FROM 地址時負責接收的自訂 REPLY-TO 地址
- 您的 Amazon SES 組態

> **Note**
>
> 如果您的驗證身分是電子郵件地址，Amazon Cognito 會將該電子郵件地址設定為預設的 FROM (寄件者) 電子郵件地址和 REPLY-TO (回覆至) 電子郵件地址。但是，如果您已驗證的身分是網域，則必須提供 FROM (寄件者) 電子郵件地址和 REPLY-TO (回覆至) 電子郵件地址的值。例如，如果您已驗證的網域是 example.com，您可以將 no-reply@example.com 設定為 FROM (寄件者) 電子郵件地址和 REPLY-TO (回覆至) 電子郵件地址。

如果您想要使用預設的 Amazon Cognito 電子郵件組態和地址，請省略此程序。

將使用者集區設定為使用自訂電子郵件地址

1. 前往 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 選擇 User Pools (使用者集區)。

3. 從清單中選擇現有的使用者集區。
4. 選擇 Messaging (簡訊) 索引標籤，尋找 Email configuration (電子郵件組態)，選擇 Edit (編輯)。
5. 在 Edit email configuration (編輯電子郵件組態) 頁面上，選取 Send email from Amazon SES (從 Amazon SES 傳送電子郵件) 或者 Send email with Amazon Cognito (使用 Amazon Cognito 傳送電子郵件)。只有當您選擇 Send email from Amazon SES (從 Amazon SES 傳送電子郵件) 時，才可以自訂 SES Region (SES 區域)、Configuration Set (組態集)，以及 FROM sender name (FROM (寄件者) 姓名)。
6. 若要使用自訂 FROM (寄件者) 地址，請完成下列步驟：
   a. 在 SES Region (SES 區域) 下選擇包含您已驗證電子郵件地址的區域。
   b. 在 FROM email address (FROM (寄件者) 電子郵件地址) 下，請選擇您的電子郵件地址。請使用通過 Amazon SES 驗證的電子郵件地址。
   c. (選用) 在 Configuration set (組態集) 下，選擇 Amazon SES 要使用的組態集。進行並保存此變更會建立服務連結角色。
   d. (選用) 在 FROM sender address (FROM (寄件者) 傳送人地址) 下，輸入電子郵件地址。您可以僅提供電子郵件地址，或是 `Jane Doe <janedoe@example.com>` 格式的電子郵件地址和易記名稱。
   e. (選用) 在 REPLY-TO email address (REPLY-TO (回覆至) 電子郵件地址) 下，輸入電子郵件地址，作為您想要用來接收使用者傳送訊息到 FROM (寄件者) 地址時負責接收的地址。
7. 選擇儲存變更。

相關主題

- 自訂電子郵件驗證訊息
- 自訂使用者邀請訊息

Amazon Cognito 使用者集區的簡訊設定

您使用者集區的部分 Amazon Cognito 事件可能會導致 Amazon Cognito 傳送簡訊給您的使用者。例如，如果您將您的使用者集區設定成需要電話驗證，則當使用者在您的應用程式中註冊新帳戶，或是重新設定其密碼時，Amazon Cognito 就會傳送簡訊。根據啟動簡訊的動作，簡訊會包含驗證碼、臨時密碼或歡迎訊息。

Amazon Cognito 使用 Amazon Simple Notification Service (Amazon SNS) 傳輸簡訊。如果您是第一次透過 Amazon Cognito 或 Amazon SNS 傳送簡訊，Amazon SNS 會將您放入 Amazon SNS 的沙盒環

境中。在沙盒環境中，您可以測試應用程式的簡訊功能。在沙盒中，簡訊只能傳送至已驗證的電話號碼。

Amazon SNS 會針對簡訊收費。如需詳細資訊，請參閱 Amazon SNS 定價。

> **Note**
>
> 由於全世界有大量不受歡迎的 SMS 流量，部分政府對 SMS 訊息的傳送者和接收者之間設有障礙。當您使用 SMS 訊息進行 MFA 和使用者更新時，您必須採取額外步驟以確保訊息傳輸。您還必須監控使用者居住地國家/地區的 SMS 訊息相關法規，並使 SMS 訊息組態保持在最新狀態。如需詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的手機簡訊 (SMS)。
>
> 使用 SMS 訊息進行身分驗證和驗證使用者並不是安全性最佳作法。電話號碼所有者可能會變更，並且對於 MFA 使用者擁有的部分因素來說，可能不是一項可靠的因素。相反，您可以在應用程式中或使用第三方 IdP 中實施 TOTP MFA。您也可以使用 自訂身分驗證挑戰 Lambda 觸發程序 建立其他自訂的身分驗證因素。

Amazon Cognito 會傳送簡訊給您的使用者，當中包含可供他們輸入的代碼。下表顯示可產生簡訊的事件。

訊息選項

活動	API 操作	傳遞選項	格式選項	可自訂	訊息範本
忘記密碼	ForgotPassword	電子郵件，短信	code	否	N/A
邀請	AdminCreateUser	電子郵件，短信	code	是	邀請訊息
自助登記	SignUp	電子郵件，短信	代碼，鏈接	是	驗證訊息
電子郵件地址或電話號碼驗證	UpdateUserAttributes	電子郵件，短信	code	是	驗證訊息

活動	API 操作	傳遞選項	格式選項	可自訂	訊息範本
多重要素驗證 (MFA)	AdminInitiateAuth, InitiateAuth	簡訊、驗證器應用程式	code	是 [1]	MFA 訊息

[1] 適用簡訊。

第一次在 Amazon Cognito 使用者集區中設定簡訊

Amazon Cognito 會使用 Amazon SNS 傳送簡訊到您的使用者集區。您也可以使用 自訂 SMS 寄件者 Lambda 觸發程序 以使用自己的資源來傳送簡訊。當您第一次設定 Amazon SNS 以特定傳送簡訊文字訊息時 AWS 區域，Amazon SNS 會將您置於該區域的 SMS 沙箱 AWS 帳戶 中。Amazon SNS 使用沙箱來防止詐欺和濫用，並符合合規要求。當您 AWS 帳戶 在沙箱中時，Amazon SNS 會施加一些限制。例如，您最多可以將簡訊傳送給 10 個已通過 Amazon SNS 驗證的電話號碼。當您 AWS 帳戶 保留在沙箱中時，請勿將 Amazon SNS 組態用於正在生產中的應用程式。您在沙盒中時，Amazon Cognito 無法將簡訊傳送到使用者的電話號碼。

向使用者集區使用者傳送簡訊

1. 準備 IAM 角色，讓 Amazon Cognito 可用來透過 Amazon SNS 傳送簡訊
2. 選擇 AWS 區域 Amazon SNS 短信
3. 取得來源身分以傳送簡訊至美國電話號碼
4. 確認您在簡訊沙盒中
5. 將您的帳戶移出 Amazon SNS 沙盒
6. 在 Amazon SNS 中為 Amazon Cognito 驗證電話號碼
7. 在 Amazon Cognito 中完成使用者集區設定

準備 IAM 角色，讓 Amazon Cognito 可用來透過 Amazon SNS 傳送簡訊

當您從使用者集區傳送簡訊時，Amazon Cognito 會在您的帳戶中擔任 IAM 角色。Amazon Cognito 會使用指派給該角色的 `sns:Publish` 許可，來傳送簡訊給使用者。在 Amazon Cognito 主控台中，您可以在 SMS 下使用者集區的 Messaging (簡訊) 索引標籤中設定 IAM role selection (IAM 角色選擇)，或在使用者集區建立精靈期間進行此選擇。

下列範例 IAM 角色信任政策授與 Amazon Cognito 使用者集區擔任角色的有限功能。Amazon Cognito 只有在代表 `aws:SourceArn` 條件中的使用者集區和 `aws:SourceAccount` 條件中的 AWS 帳戶時，才能擔任此角色。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cognito-idp.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "<your account number>"
            },
            "ArnLike": {
                "aws:SourceArn": "<your user pool ARN>"
            }
        }
    }]
}
```

您可指定精確的使用者集區 ARN，或在 `aws:SourceArn` 條件的值中使用萬用字元 ARN。在 AWS Management Console 或使用 DescribeUserPoolAPI 請求中查找用戶集區的 ARN。

如需有關 IAM 角色和信任政策的詳細資訊，請參閱AWS Identity and Access Management 使用者指南中的角色術語和概念。

選擇 AWS 區域 Amazon SNS 短信

在某些情況下 AWS 區域，您可以選擇包含要用於 Amazon Cognito SMS 訊息之 Amazon SNS 資源的區域。 AWS 區域 在任何可使用 Amazon Cognito 的地方 (亞太區域 (首爾) 除外，您都可以在建立使用者集區的 AWS 區域 位置使用 Amazon SNS 資源。若要在可以選擇區域時，讓您的簡訊更快、更可靠，請使用與使用者集區位於相同區域的 Amazon SNS 資源。

> **Note**
>
> 在中 AWS Management Console，您只能在切換到新的 Amazon Cognito 主控台體驗之後變更 SMS 資源的區域。

在新增使用者集區精靈的 Configure message delivery (設定訊息傳遞) 步驟中，選擇簡訊資源的區域。您也可以在現有使用者集區的 Messaging (簡訊) 索引標籤中的 SMS (簡訊) 下方，選擇 Edit (編輯)。

在啟動時，對於某些人來說 AWS 區域，Amazon Cognito 會在替代區域中使用 Amazon SNS 資源傳送簡訊。若要設定您偏好的區域，請使用使用者集區的SmsConfigurationType物件SnsRegion參數。當您以程式設計方式在下表的 Amazon Cognito Region (Amazon Cognito 區域) 中建立 Amazon Cognito 使用者集區資源，而且不提供 SnsRegion 參數時，您的使用者集區將使用舊式 Amazon SNS Region (Amazon SNS 區域) 中的 Amazon SNS 資源傳送 SMS 訊息。

亞太區域 (首爾) 的 Amazon Cognito 使用者集區 AWS 區域 必須在亞太區域 (東京) 區域使用您的 Amazon SNS 組態。

Amazon SNS 將所有新帳戶的花費配額設定為每月 1.00 USD (美元)。您可能已經增加了與 Amazon Cognito AWS 區域 一起使用的支出限制。在變更 Amazon SNS SMS 訊息之前，請先在 Sup AWS port 中心開啟配額增加案例，以增加新區域中的限制。 AWS 區域 如需詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的為 Amazon SNS 請求提升您的每月簡訊花費配額。

您可以使用對應的 Amazon SNS Region (Amazon SNS 區域) 中的 Amazon SNS 資源，將 SMS 訊息傳送至下表中的任何 Amazon Cognito Region (Amazon Cognito 區域)。

Amazon Cognito 區域	Amazon SNS 區域
美國東部 (俄亥俄)	美國東部 (俄亥俄)、美國東部 (維吉尼亞北部)
加拿大 (中部)	加拿大 (中部)、美國東部 (維吉尼亞北部)
歐洲 (法蘭克福)	歐洲 (法蘭克福)、歐洲 (愛爾蘭)
歐洲 (倫敦)	歐洲 (倫敦)、歐洲 (愛爾蘭)
亞太區域 (首爾)	亞太區域 (東京)
美國東部 (維吉尼亞北部)	美國東部 (維吉尼亞北部)
美國西部 (加利佛尼亞北部)	美國西部 (加利佛尼亞北部)
美國西部 (奧勒岡)	美國西部 (奧勒岡)
亞太區域 (孟買)	亞太區域 (孟買)、亞太區域 (新加坡)
亞太區域 (海德拉巴)	亞太區域 (海德拉巴)

Amazon Cognito 區域	Amazon SNS 區域
亞太區域 (新加坡)	亞太區域 (新加坡)
亞太區域 (悉尼)	亞太區域 (悉尼)
亞太區域 (東京)	亞太區域 (東京)
亞太區域 (雅加達)	亞太區域 (雅加達)
亞太區域 (大阪)	亞太區域 (大阪)
亞太區域 (墨爾本)	亞太區域 (墨爾本)
歐洲 (愛爾蘭)	歐洲 (愛爾蘭)
Europe (Paris)	Europe (Paris)
歐洲 (斯德哥爾摩)	歐洲 (斯德哥爾摩)
歐洲 (米蘭)	歐洲 (米蘭)
歐洲 (西班牙)	歐洲 (西班牙)
Middle East (Bahrain)	Middle East (Bahrain)
南美洲 (聖保羅)	南美洲 (聖保羅)
以色列 (特拉維夫)	以色列 (特拉維夫)
非洲 (開普敦)	非洲 (開普敦)
中東 (阿拉伯聯合大公國)	中東 (阿拉伯聯合大公國)
歐洲 (蘇黎世)	歐洲 (蘇黎世)

取得來源身分以傳送簡訊至美國電話號碼

如果您打算將簡訊傳送至美國電話號碼，必須取得來源身分，無論您是建置簡訊沙盒測試環境，或是生產環境。

2021 年 6 月 1 日起，美國電信業者要求來源身分，才能將簡訊傳送至美國電話號碼。如果還沒有來源身分，您必須取得一個。若要了解如何取得來源身分，請參閱《Amazon Pinpoint 使用者指南》中的要求號碼。

如果您在以下方面進行操作 AWS 區域，則必須打開 AWS Support 票證才能獲得發起身份。如需說明，請參閱《Amazon Simple Notification Service 開發人員指南》中的要求簡訊支援。

- 美國東部 (俄亥俄)
- 歐洲 (斯德哥爾摩)
- Europe (Paris)
- 歐洲 (米蘭)
- 中東 (巴林)
- 南美洲 (聖保羅)
- 美國西部 (加利佛尼亞北部)

如果您有多個相同的創始身分 AWS 區域，Amazon SNS 會按照下列優先順序選擇一個起始身分類型：短碼、10DLC、免費電話號碼。您無法變更此優先權。如需詳細資訊，請參閱 Amazon SNS 常見問常見問答集。

確認您在簡訊沙盒中

使用下列操作程序來確認您是否位於簡訊沙盒中。針對您擁有生產 Amazon Cognito 使用者集區的每個 AWS 區域 位置重複此動作。

在 Amazon Cognito 主控台中檢視簡訊沙盒狀態

確認您是否在 SMS 沙盒中

1. 前往 Amazon Cognito 主控台。若出現提示，請輸入 AWS 憑證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區。
4. 選擇 Messaging (簡訊) 索引標籤。
5. 在 SMS configuration (SMS 組態) 區段中，展開 Move to Amazon SNS production environment (移至 Amazon SNS 生產環境)。如果您的帳戶在 SMS 沙盒中，您會看到以下訊息：

   ```
   You are currently in the SMS Sandbox and cannot send SMS messages to
   unverified numbers.
   ```

如果您沒有看到此訊息，表示有人已在您的帳戶中設定簡訊。跳至 在 Amazon Cognito 中完成使用者集區設定。

6. 選擇訊息中的 Amazon SNS 連結。如此會在新索引標籤中開啟 Amazon SNS 主控台。
7. 確認您在沙盒環境中。主控台訊息會指出您的沙箱狀態 AWS 區域，如下所示：

```
This account is in the SMS sandbox in US East (N. Virginia).
```

將您的帳戶移出 Amazon SNS 沙盒

如果您正在測試應用程式，而且只需將簡訊傳送到管理員可驗證的電話號碼，請跳過此步驟。

若要在生產環境中使用您的應用程式，請將帳戶移出簡訊沙盒，然後移至生產環境中。在包含您希望 Amazon Cognito 使用的 AWS 區域 Amazon SNS 資源的中設定起始身分後，您可以在 AWS 帳戶 保留在 SMS 沙箱中的同時驗證美國電話號碼。當您的 Amazon SNS 環境處於生產狀態時，您無需驗證 Amazon SNS 中的使用者電話號碼，就能向使用者傳送簡訊。

如需詳細說明，請參閱 Amazon Simple Notification Service 開發人員指南中的移出簡訊沙盒。

在 Amazon SNS 中為 Amazon Cognito 驗證電話號碼

如果您已將帳戶移出簡訊沙盒，請略過此步驟。

當您在簡訊沙盒中時，可以將訊息傳送至已通過 Amazon SNS 驗證的任何電話號碼。

若要驗證電話號碼，請執行以下操作：

1. 在 Amazon SNS 主控台的 Text messaging (SMS) (文字簡訊 (SMS)) 區段中新增 Sandbox destination phone number ((沙盒目的地電話號碼))。
2. 透過您提供的電話號碼接收附帶代碼的簡訊。
3. 在 Amazon SNS 主控台中輸入來自簡訊的 Verification code (驗證碼)。

如需詳細說明，請參閱《Amazon Simple Notification Service 開發人員指南》中的在簡訊沙盒中新增和驗證電話號碼。

> **Note**
>
> Amazon SNS 會限制您在簡訊沙盒時可驗證的目的地電話號碼的數量。請參閱 Amazon Simple Notification Service 開發人員指南中的簡訊沙盒。

在 Amazon Cognito 中完成使用者集區設定

回到您建立或編輯使用者集區的瀏覽器索引標籤。完成程序。成功將 SMS 組態新增至使用者集區後，Amazon Cognito 會將測試訊息傳送至內部電話號碼，以確認您的組態是否有效。Amazon SNS 會針對每個測試簡訊收取費用。

將權杖用於使用者集區

使用權杖 (tokens) 驗證使用者並授與資源的存取權限。權杖中的宣告是有關您的使用者的資訊。ID 權杖包含有關其身分的宣告，例如其使用者名稱、姓氏和電子郵件地址。存取權杖包含 `scope` 之類的宣告，經過驗證的使用者可用來存取第三方 API、Amazon Cognito 使用者自助服務 API 操作，以及 UserInfo 端點。存取和 ID 權杖都包含一個 `cognito:groups` 宣告，其中包含使用者集區中的使用者群組成員資格。如欲了解使用者集區群組的詳細資訊，請參閱新增群組至使用者集區。

Amazon Cognito 也重新整理權杖，您可以用來取得新權杖或撤銷現有權杖。重新整理權杖來擷取新的 ID 和存取權杖。撤銷權杖以撤銷重新整理權杖允許的使用者存取權限。

Amazon Cognito 核發的權杖採用 Base64 編碼字串。您可以將任何 Amazon Cognito ID 或存取權杖從 base64 解碼為純文字 JSON。Amazon Cognito 重新整理權杖已加密，對使用者集區使用者和管理員不透明，並只能由使用者集區讀取。

權杖驗證

使用者登入您的應用程式時，Amazon Cognito 會驗證登入資訊。如果登入成功，Amazon Cognito 就會為已驗證使用者建立工作階段，並傳回 ID 權杖、存取權杖和重新整理權杖。您可以使用這些權杖，授予使用者對下游資源和 Amazon API Gateway 等 API 的存取權。或者，您可以將其用於交換臨時 AWS 憑證，以便存取其他 AWS 服務。

存放權杖

您的應用程式必須能夠存放不同大小的權杖。權杖大小可能會因各種原因而變更，包括但不限於額外宣告、編碼演算法變更和加密演算法變更。當您在使用者集區中啟用權杖撤銷時，Amazon Cognito 會向 JSON Web 權杖新增額外宣告，從而增加其大小。存取權杖和 ID 權杖中會新增 `origin_jti` 和 `jti` 宣告。如需有關權杖撤銷的詳細資訊，請參閱撤銷權杖。

> **Important**
>
> 最佳實務是在應用程式環境中傳輸及儲存所有權杖時，都要確保安全無虞。權杖可以包含有關使用者的個人識別資訊，以及用於使用者集區的安全模型相關資訊。

自訂權杖

您可以自訂 Amazon Cognito 傳遞給您應用程式的存取權和 ID 權杖。在產生權杖前 Lambda 觸發程序中，您可以新增、修改和抑制權杖宣告。產生權杖前觸發程序是 Amazon Cognito 向其傳送一組預設宣告的 Lambda 函數。這些宣告包括 OAuth 2.0 範圍，使用者集區群組成員資格、使用者屬性和其他。然後，函數可以藉此機會在執行階段進行變更，並將更新的權杖宣告傳回 Amazon Cognito。

使用第 2 版事件進行存取權杖自訂需要支付額外費用。如需詳細資訊，請參閱 Amazon Cognito 定價。

主題

- 使用 ID 權杖
- 使用存取權杖
- 使用重新整理權杖
- 撤銷權杖
- 驗證 JSON Web 權杖
- 快取權杖

使用 ID 權杖

ID 權杖是 JSON Web 權杖 (JWT)，它包含已驗證使用者身分的相關宣告，例如 `name`、`email` 和 `phone_number`。您可以在應用程式內部使用此身分資訊。ID 權杖也可以用來向資源伺服器或伺服器應用程式驗證使用者的身分。您還可以透過 Web API 操作，在應用程式外使用 ID 權杖。在那些情況下，您必須先驗證 ID 權杖的簽章，才能信任 ID 權杖中的任何宣告。請參閱驗證 JSON Web 權杖。

您可以將 ID 權杖過期設為 5 分鐘到 1 天的值。此值可根據應用程式用戶端設定。

> **Important**
>
> 當您的使用者以託管 UI 或聯合身分提供者 (IdP) 登入時，Amazon Cognito 會設定有效期為 1 小時的工作階段 Cookie。如果您使用託管 UI 或聯合，並且為存取和 ID 權杖指定少於 1 小時

的最短持續時間，則您的使用者仍將有一個有效的工作階段，直到 Cookie 到期為止。如果使用者的權杖會在一小時的工作階段內過期，則使用者可以重新整理其權杖，無需重新驗證身分。

ID 權杖標頭

標頭包含兩項資訊：金鑰 ID (`kid`) 和演算法 (`alg`)。

```
{
"kid" : "1234example=",
"alg" : "RS256"
}
```

`kid`

金鑰 ID。它的值指出用於保護權杖之 JSON Web Signature (JWS) 的金鑰。您可以在 `jwks_uri` 端點檢視您的使用者集區簽署金鑰 ID。

如需 `kid` 參數的詳細資訊，請參閱金鑰識別符 (kid) 標頭參數。

`alg`

Amazon Cognito 用來保護存取權杖安全的加密演算法。使用者集區是使用 RS256 演算法，即採用 SHA-256 的 RSA 簽章。

如需 `alg` 參數的詳細資訊，請參閱演算法 (alg) 標頭參數。

ID 令牌默認有效載荷

這是來自 ID 令牌的示例有效載荷。它包含有關已驗證使用者的宣告。有關 OpenID Connect（OIDC）標準聲明的更多信息，請參閱 OID C 標準聲明列表。您可以添加您自己設計的聲明與產生權杖前 Lambda 觸發程序.

```
<header>.{
    "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "cognito:groups": [
        "test-group-a",
        "test-group-b",
        "test-group-c"
```

```
    ],
    "email_verified": true,
    "cognito:preferred_role": "arn:aws:iam::111122223333:role/my-test-role",
    "iss": "https://cognito-idp.us-west-2.amazonaws.com/us-west-2_example",
    "cognito:username": "my-test-user",
    "middle_name": "Jane",
    "nonce": "abcdefg",
    "origin_jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "cognito:roles": [
        "arn:aws:iam::111122223333:role/my-test-role"
    ],
    "aud": "xxxxxxxxxxxxexample",
    "identities": [
        {
            "userId": "amzn1.account.EXAMPLE",
            "providerName": "LoginWithAmazon",
            "providerType": "LoginWithAmazon",
            "issuer": null,
            "primary": "true",
            "dateCreated": "1642699117273"
        }
    ],
    "event_id": "64f513be-32db-42b0-b78e-b02127b4f463",
    "token_use": "id",
    "auth_time": 1676312777,
    "exp": 1676316377,
    "iat": 1676312777,
    "jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "email": "my-test-user@example.com"
}
.<token signature>
```

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 宣告是識別特定使用者的最佳方法。

cognito:groups

以您的使用者為成員的使用者集區群組名稱陣列。群組可以是您呈現給應用程式的識別碼，也可以從身分池產生偏好 IAM 角色的請求。

cognito:preferred_role

您與使用者最高優先順序之使用者集區群組相關聯的 IAM 角色 ARN。如需使用者集區如何選取此角色宣告的詳細資訊，請參閱指定優先順序值給群組。

iss

發行權杖的身分提供者。宣告的格式如下。

`https://cognito-idp.<Region>.amazonaws.com/<your user pool ID>`

cognito:username

您的使用者在您的使用者集區中的使用者名稱。

nonce

nonce 宣告來自同名的參數，您可以在對 OAuth 2.0 `authorize` 端點的請求中新增此參數。新增參數時，nonce 宣告會包含在 Amazon Cognito 發出的 ID 權杖中，您可以使用它來防止重新執行攻擊。如果您未提供 nonce 值，Amazon Cognito 會在您透過第三方身分提供者進行身分驗證時自動產生並驗證 Nonce，然後將其新增為 ID 權杖中的 nonce 宣告。Amazon Cognito 中的 nonce 宣告實作符合 OIDC 標準。

origin_jti

與使用者的重新整理權杖相關的權杖撤銷識別符。Amazon Cognito 在檢查您是否使用撤銷端點或 RevokeTokenAPI 作業撤銷使用者權杖時，會參考`origin_jti`宣告。在撤銷權杖時，Amazon Cognito 會使所有具相同 `origin_jti` 值的存取權和 ID 權杖無效。

cognito:roles

與使用者群組相關聯之 IAM 角色的名稱陣列。每個使用者集區群組都可以有一個與其關聯的 IAM 角色。此陣列代表使用者群組的所有 IAM 角色，無論優先順序為何。如需詳細資訊，請參閱 新增群組至使用者集區。

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。

identities

使用者的 `identities` 屬性內容。此屬性包含您藉由聯合登入或將聯合身分使用者連結至本機設定檔，連結至使用者的每一個第三方身分提供者設定檔的相關資訊。此資訊包含其提供者名稱、提供者唯一 ID 和其他中繼資料。

token_use

權杖的用途。在 ID 權杖中，其值為 id。

auth_time

您的使用者完成身分驗證的身分驗證時間 (以 Unix 時間格式表示)。

exp

使用者權杖到期的到期時間 (以 Unix 時間格式表示)。

iat

Amazon Cognito 發行您使用者權杖的時間 (採用 Unix 時間格式)。

jti

JWT 的唯一識別碼。

ID 權杖可包含 OIDC 標準宣告所定義的 OIDC 標準宣告。這個 ID 權杖也可以包含您在使用者集區中所定義的自訂屬性。Amazon Cognito 將自訂屬性值做為字串寫入 ID 權杖，無論屬性類型為何。

> **Note**
>
> 使用者集區自訂屬性一律為前置詞。custom:

ID 權杖簽章

ID 權杖的簽章是根據 JWT 權杖的標頭和酬載計算出來的。在您接受應用程式收到的任何 ID 權杖中的宣告之前，請先驗證權杖的簽章。如需詳細資訊，請參閱「驗證 JSON Web 權杖」。驗證 JSON Web 權杖。

使用存取權杖

使用者集區存取權杖包含已驗證使用者的相關宣告、使用者的群組清單，以及範圍的清單。存取權杖的目的是授權 API 操作。您的使用者集區接受存取權杖以授權使用者自助服務操作。例如，您可使用存取權杖，授予能新增、變更或刪除使用者屬性的使用者存取權限。

使用存取權杖中的 OAuth 2.0 範圍 (衍生自您新增到使用者集區的自訂範圍)，您可以授權使用者從 API 擷取資訊。例如，Amazon API Gateway 支援使用 Amazon Cognito 存取權杖進行授權。您可以使用取

自使用者集區的資訊填入 REST API 授權方，或使用 Amazon Cognito 做為 HTTP API 的 JSON Web 權杖 (JWT) 授權方。若要產生具有自訂範圍的存取權杖，您必須透過使用者集區公有端點來請求。

使用者的存取權杖是可向 UserInfo 端點 要求更多使用者屬性相關資訊的許可。使用者的存取權杖也是讀取和寫入使用者屬性的許可。存取權杖授予屬性的存取層級，取決於您指派給應用程式用戶端的許可，以及您在權杖中授予的範圍。

存取權杖是 JSON Web 權杖 (JWT)。存取權杖標頭的結構與 ID 權杖相同。Amazon Cognito 使用與簽署 ID 權杖不同的金鑰簽署存取權杖。存取金鑰 ID (`kid`) 宣告的值不會符合來自同一使用者工作階段之 ID 權杖中 `kid` 宣告的值。在您的應用程式程式碼中，獨立驗證 ID 權杖和存取權杖。在驗證簽章之前，請勿信任存取權杖中的宣告。如需詳細資訊，請參閱 驗證 JSON Web 權杖。您可以將存取權限有效期限設為 5 分鐘到 1 天之間的任何值。此值可根據應用程式用戶端設定。

> **Important**
>
> 對於存取和 ID 權杖，如果您使用託管 UI，最低不得指定少於一小時。Amazon Cognito 託管 UI 會使用有效期為一小時的 Cookie。如果您輸入不到一小時的最小值，將無法取得更少過期時間。

存取權杖標頭

標頭包含兩項資訊：金鑰 ID (`kid`) 和演算法 (`alg`)。

```
{
"kid" : "1234example="
"alg" : "RS256",
}
```

`kid`

金鑰 ID。它的值指出用於保護權杖之 JSON Web Signature (JWS) 的金鑰。您可以在 `jwks_uri` 端點檢視您的使用者集區簽署金鑰 ID。

如需 `kid` 參數的詳細資訊，請參閱金鑰識別符 (kid) 標頭參數。

`alg`

Amazon Cognito 用來保護存取權杖安全的加密演算法。使用者集區是使用 RS256 演算法，即採用 SHA-256 的 RSA 簽章。

如需 alg 參數的詳細資訊，請參閱演算法 (alg) 標頭參數。

存取權杖預設有效載

這是一個存取權杖酬載範例。如需詳細資訊，請參閱 JWT 宣告。您可以添加您自己設計的聲明與產生權杖前 Lambda 觸發程序.

```
<header>.
{
   "sub":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "device_key": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "cognito:groups":[
      "testgroup"
   ],
   "iss":"https://cognito-idp.us-west-2.amazonaws.com/us-west-2_example",
   "version":2,
   "client_id":"xxxxxxxxxxxxexample",
   "origin_jti":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "event_id":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "token_use":"access",
   "scope":"phone openid profile resourceserver.1/appclient2 email",
   "auth_time":1676313851,
   "exp":1676317451,
   "iat":1676313851,
   "jti":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "username":"my-test-user"
}
.<token signature>
```

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 宣告是識別特定使用者的最佳方法。

cognito:groups

以您的使用者為成員的使用者集區群組名稱陣列。

iss

發行權杖的身分提供者。宣告的格式如下。

`https://cognito-idp.<Region>.amazonaws.com/<your user pool ID>`

client_id

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 ID 權杖 aud 宣告中呈現相同的值。

origin_jti

與使用者的重新整理權杖相關的權杖撤銷識別符。Amazon Cognito 在檢查您是否使用撤銷端點或 RevokeTokenAPI 作業撤銷使用者權杖時，會參考origin_jti宣告。在撤銷權杖時，Amazon Cognito 會使所有具相同 origin_jti 值的存取權和 ID 權杖無效。

token_use

權杖的用途。在存取權杖中，其值為 access。

scope

OAuth 2.0 範圍的清單，用於定義字符提供的存取權。來自 權杖端點 的權杖可以包含您的應用程式用戶端支援的任何範圍。來自 Amazon Cognito API 登入的權杖僅包含 aws.cognito.signin.user.admin 範圍。

auth_time

您的使用者完成身分驗證的身分驗證時間 (以 Unix 時間格式表示)。

exp

使用者權杖到期的到期時間 (以 Unix 時間格式表示)。

iat

Amazon Cognito 發行您使用者權杖的時間 (採用 Unix 時間格式)。

jti

JWT 的唯一識別碼。

username

您的使用者在您的使用者集區中的使用者名稱。

存取權杖簽章

存取權杖的簽章是根據 JWT 權杖的標頭和酬載計算出來的。如果是用在 Web API 的應用程式外部，則您一定要先驗證這個簽章，再處理 ID 權杖。如需詳細資訊，請參閱 驗證 JSON Web 權杖。

使用重新整理權杖

您可以使用重新整理權杖來擷取新的 ID 權杖和存取權杖。在預設情況下，重新整理權杖會在您的應用程式使用者登入您的使用者集區起 30 天後過期。當建立使用者集區的應用程式時，您可以將應用程式的重新整理權杖過期設為 60 分鐘到 10 年之間的任何值。

如果具有效 (未過期) 的重新整理權杖，適用於 iOS 的 Mobile SDK、適用於 Andriod 的 Mobile SDK、適用於 iOS、Android 和 Flutter 的 Amplify，會自動重新整理 ID 和存取權杖。ID 權杖和存取權杖的剩餘有效期至少為 2 分鐘。如果重新整理權杖已過期，您的應用程式使用者就必須再次登入到您的使用者集區，重新驗證。如果存取權杖和 ID 權杖的最低有效期設定為 5 分鐘，且您正在使用 SDK，重新整理權杖會持續用於擷取新的存取和 ID 權杖。若要查看預期的行為，請設定最小值為 7 分鐘，而不是 5 分鐘。

只要使用者在新帳戶的 `UnusedAccountValidityDays` 時間限制之前，至少登入一次，使用者帳戶本身就不會過期。

使用重新整理權杖取得新的存取和身分權杖

使用 API 或託管 UI 以啟動重新整理權杖的身分驗證。

若要使用重新整理權杖透過使用者集區 API 取得新的 ID 和存取權杖，請使用AdminInitiateAuth或 InitiateAuthAPI 作業。為 `AuthFlow` 參數傳遞 `REFRESH_TOKEN_AUTH`。在 `AuthFlow` 的 `AuthParameters` 屬性中，將使用者的重新整理權杖作為 `"REFRESH_TOKEN"` 的值傳遞。在您的 API 請求通過所有挑戰後，Amazon Cognito 會傳回新的 ID 權杖和存取權杖。

> **Note**
>
> 若要使用 Amazon Cognito 使用者集區 API 重新整理託管 UI 使用者的權杖，請產生 `InitiateAuth` 請求。

您也可以將重新整理權杖提交至已設定網域的使用者集區中的 權杖端點。在請求內文中，包含 `refresh_token` 作為 `grant_type` 的值以及您的使用者重新整理權杖作為 `refresh_token` 的值。

撤銷重新整理權杖

您可以撤銷屬於使用者的重新整理權杖。如需撤銷權杖的詳細資訊，請參閱撤銷權杖。

> **Note**
>
> 若撤銷重新整理權杖，將會一併撤銷 Amazon Cognito 從具有該權杖的重新整理請求中發出的所有 ID 和存取權杖。

當您使用 `GlobalSignOut` 和 `AdminUserGlobalSignOut` API 操作來撤銷使用者的所有權杖時，使用者可以從其目前登入的所有裝置登出。使用者登出後，會產生下列影響。

- 使用者的重新整理權杖無法為使用者取得新權杖。
- 使用者的存取權杖無法提出權杖授權 API 請求。
- 使用者必須重新驗證才能取得新權杖。由於託管 UI 工作階段 Cookie 不會自動過期，因此您的使用者可以使用工作階段 Cookie 重新驗證，而不會再提示輸入憑證。您將託管 UI 使用者登出之後，將其重新導向 登出端點，Amazon Cognito 將在該處清除其工作階段 Cookie。

使用重新整理權杖就可以在應用程式中長時間保留使用者的工作階段。經過一段時間後，您的使用者可能想要取消某些已登入裝置的授權，並持續重新整理其工作階段。若要將使用者從單一裝置登出，請撤銷其重新整理權杖。當您的用戶想要從所有經過身份驗證的會話中簽出自己時，請生成 GlobalSignOutAPI 請求。您的應用程式可向您的使用者顯示從所有裝置登出這類選項。`GlobalSignOut` 可接受使用者的 valid–unaltered、unexpired、not-revoked–access 權杖。由於此 API 是權杖授權，因此使用者無法用它來啟動另一位使用者的登出。

但是，您可以生成 AdminUserGlobalSignOutAPI 請求，您可以使用您的 AWS 憑據授權該 API 請求，以便從其所有設備中註銷任何用戶。管理員應用程序必須使用 AWS 開發人員憑據調用此 API 操作，並將用戶池 ID 和用戶的用戶名作為參數傳遞。`AdminUserGlobalSignOut`API 可以將使用者集區中的任何使用者登出。

如需有關可使用 AWS 認證或使用者存取權杖授權之要求的詳細資訊，請參閱Amazon Cognito 使用者集區經身分驗證和未進行身分驗證的 API 操作。

撤銷權杖

您可以使用 AWS API 撤銷使用者的重新整理權杖。撤銷重新整理權杖時，先前該重新整理權杖發行的所有存取權杖都會無效。發行給使用者的其他重新整理權杖不受影響。

> **Note**
>
> JWT 權杖為獨立式，具建立權杖時指派的簽章和過期時間。撤銷的權杖不能與任何需要權杖的 Amazon Cognito API 呼叫搭配使用。但如果使用驗證權杖簽章和過期時間的任何 JWT 程式庫進行驗證，則已撤銷的權杖仍有效。

您可以在啟用撤銷權杖功能的情況下，撤銷使用者集區用戶端的重新整理權杖。建立新使用者集區用戶端時，撤銷權杖功能會預設為啟用。

啟用撤銷權杖功能

您必須先啟用撤銷權杖功能，才能為現有使用者集區用戶端撤銷權杖。您可以使用或 AWS API 為現有的使用者集區用戶端啟 AWS CLI 用權杖撤銷。若要執行此操作，請呼叫 `aws cognito-idp describe-user-pool-client` CLI 命令或 `DescribeUserPoolClient` API 操作，以從您的應用程式用戶端檢索當前設置。然後呼叫 `aws cognito-idp update-user-pool-client` CLI 命令或 `UpdateUserPoolClient` API 操作。包含來自您應用程式用戶端中的當前設定，並將 `EnableTokenRevocation` 參數設為 `true`。

當您使用 AWS Management Console、或 AWS API 建立新的使用者集區用戶端時，根據預設會啟用權杖撤銷。 AWS CLI

啟用撤銷權杖功能後，新宣告會新增到 Amazon Cognito JSON Web 權杖。存取權杖和 ID 權杖中會新增 `origin_jti` 和 `jti` 宣告。這些宣告會增加應用程式用戶端存取和 ID 權杖的大小。

要創建或修改啟用令牌撤銷的應用程序客戶端，請在CreateUserPoolClient或UpdateUserPoolClientAPI 請求中包含以下參數。

```
"EnableTokenRevocation": true
```

撤銷權杖

您可以使用 RevokeTokenAPI 要求 (例如使用 `aws cognito-idp revoke-token` CLI 命令) 撤銷重新整理權杖。您也可以使用 撤銷端點 撤銷權杖。此端點在您將網域新增至使用者集區之後才能使用。您可以在 Amazon Cognito 託管網域或您自己的自訂網域上，使用撤銷端點功能。

> **Note**
>
> 撤銷重新整理權杖的請求中必須包含用來取得該權杖的用戶端 ID。

以下是 RevokeToken API 請求範例的內文：

```
{
   "ClientId": "1example23456789",
   "ClientSecret": "abcdef123456789ghijklexample",
   "Token": "eyJjdHkiOiJKV1QiEXAMPLE"
}
```

以下是對具有自訂網域的使用者集區的 /oauth2/revoke 端點提出的範例 cURL 請求。

```
curl --location 'auth.mydomain.com/oauth2/revoke' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Authorization: Basic Base64Encode(client_id:client_secret)' \
--data-urlencode 'token=abcdef123456789ghijklexample' \
--data-urlencode 'client_id=1example23456789'
```

RevokeToken 操作和 /oauth2/revoke 端點都不需要額外授權，除非您的應用程式用戶端有用戶端密碼。

驗證 JSON Web 權杖

這些步驟描述有關驗證使用者集區 JSON web 權杖 (JWT)。

主題

- 必要條件
- 使用驗證令牌 aws-jwt-verify
- 了解和檢查權杖

必要條件

您的程式庫、軟體開發套件或軟體架構可能已處理本節的任務。 AWS 開發套件提供工具，可讓您在應用程式中處理和管理 Amazon Cognito 使用者集區權杖。 AWS Amplify 包括擷取和重新整理 Amazon Cognito 權杖的功能。

如需詳細資訊，請參閱下列頁面。

- 將 Amazon Cognito 身分驗證和授權與 Web 和行動應用程式整合
- 使 AWS 用軟體開發套件的 Amazon Cognito 身分供應商程式碼範例
- Amplify 開發人員中心中的進階工作流程

許多程式庫可用於解碼和驗證 JSON Web 權杖 (JWT)。如果您想手動處理伺服器端 API 處理的權杖，又或者您使用的是其他程式設計語言，那麼這些程式庫就很實用。請參閱可搭配 JWT 權杖使用的 OpenID Foundation 程式庫清單。

使用驗證令牌 aws-jwt-verify

在 Node.js 應用程序中， AWS 建議aws-jwt-verify庫驗證用戶傳遞給您的應用程序的令牌中的參數。使用 `aws-jwt-verify` 時，您可以使用您要驗證一或多個使用者集區的宣告值填入 `CognitoJwtVerifier`。它可以檢查的一些值包括以下內容。

- 該存取或 ID 權杖的格式正確或未過期，並具有有效的簽章。
- 該存取權杖來自正確的使用者集區和應用程式用戶端。
- 該存取權杖宣告包含正確的 OAuth 2.0 範圍。
- 簽署您的存取和 ID 權杖的金鑰符合來自使用者集區 JWKS URI 中的簽署金鑰 `kid`。

  JWKS URI 包含有關簽署使用者權杖之私有金鑰的公開資訊。您可以在以下位置找到使用者集區的 JWKS URI：`https://cognito-idp.<Region>.amazonaws.com/<userPoolId>/.well-known/jwks.json`。

有關您可以在 Node.js 應用程序或 AWS Lambda 授權者中使用的更多信息和示例代碼，敬請參閱（詳見 aws-jwt-verify）。 GitHub

了解和檢查權杖

在將權杖檢查與應用程式整合之前，請考慮 Amazon Cognito 如何組合 JWT。從使用者集區擷取範例權杖。對它們進行詳細解碼和檢查以了解它們的特徵，並確定您要驗證的內容和時間。例如，您可能想在一種情況下檢查群組成員資格，而在另一種情況下檢查範圍。

以下區段說明在準備應用程式時，手動檢查 Amazon Cognito JWT 的程序。

確認 JWT 的結構。

JSON Web 權杖 (JWT) 包含三個區段，它們之間有一個 . (點號) 分隔符號。

標頭

Amazon Cognito 用來簽署權杖的金鑰 ID (`kid`) 和 RSA 演算法 (`alg`)。Amazon Cognito 使用 `RS256` 的 `alg` 簽署權杖。

承載

權杖宣告。在 ID 權杖中，宣告包括使用者屬性和有關使用者集區 (iss) 和應用程式用戶端 (aud) 的資訊。在存取權杖中，承載包括範圍、群組成員資格、您的使用者集區為 `iss`，以及您的應用程式用戶端為 `client_id`。

簽章

簽章不像標頭和承載那樣是可解碼的 base64。它是從您可在 JWKS URI 中觀察到的簽署金鑰和參數衍生出來的 RSA256 識別符。

標頭和承載是以 base64 編碼的 JSON。您可以透過解碼為起始字元 `{` 的開頭字元 `eyJ` 以識別它們。如果您的使用者向應用程式提供 base64 編碼的 JWT，但其格式不是 `[JSON Header].[JSON Payload].[Signature]`，則它不是有效的 Amazon Cognito 權杖，您可以捨棄它。

驗證 JWT

JWT 簽章是標頭和酬載的雜湊組合。Amazon Cognito 會為每個使用者集區產生兩組 RSA 加密金鑰對。一個私有金鑰簽署存取權杖，另一個簽署 ID 權杖。

驗證 JWT 權杖的簽章

1. 解碼 ID 權杖。

   OpenID Foundation 也會提供可搭配 JWT 權杖使用的程式庫清單。

   您也可以使用 AWS Lambda 來解碼使用者集區 JWT。如需詳細資訊，請參閱使用解碼和驗證 Amazon Cognito JWT 權杖。 AWS Lambda
2. 比較本機金鑰 ID (`kid`) 與公有 `kid`。

   a. 下載並存放對應至您的使用者集區的公開 JSON Web Key (JWK)。JSON Web Key Set (JWKS) 中會提供這個金鑰。針對您的環境建構以下 `jwks_uri` URI 即可找到它：

      ```
      https://cognito-idp.<Region>.amazonaws.com/<userPoolId>/.well-known/jwks.json
      ```

      如需 JWK 和 JWK Set 的詳細資訊，請參閱 JSON Web Key (JWK)。

> **Note**
>
> Amazon Cognito 可能會輪換使用者集區中的簽署金鑰。最佳做法是在應用程式中快取公有金鑰，使用 `kid` 做為快取金鑰，並定期重新整理快取。將應用程式所收到權杖中的 `kid` 與您的快取進行比較。
> 如果您收到的權杖具有正確的發行者但 `kid` 不同，則 Amazon Cognito 可能已輪換簽署金鑰。從您的使用者集區 `jwks_uri` 端點重新整理快取。

這是範本 `jwks.json` 檔案：

```
{
 "keys": [{
  "kid": "1234example=",
  "alg": "RS256",
  "kty": "RSA",
  "e": "AQAB",
  "n": "1234567890",
  "use": "sig"
 }, {
  "kid": "5678example=",
  "alg": "RS256",
  "kty": "RSA",
  "e": "AQAB",
  "n": "987654321",
  "use": "sig"
 }]
}
```

金鑰 ID (**`kid`**)

`kid` 是指出在保護權杖之 JSON Web Signature (JWS) 安全時所使用金鑰的提示。

演算法 (**`alg`**)

`alg` 標頭參數代表用來保護 ID 權杖安全的加密演算法。使用者集區是使用 RS256 演算法，即採用 SHA-256 的 RSA 簽章。如需 RSA 的詳細資訊，請參閱 RSA 加密法。

金鑰類型 (**`kty`**)

`kty` 參數會識別搭配金鑰 (本範例中的金鑰為 "RSA") 的加密演算法系列。

RSA 指數 (**e**)

這個 e 參數包含 RSA 公開金鑰的指數值。它以 Base64urlUInt 編碼值表示。

RSA 模數 (**n**)

這個 n 參數包含 RSA 公開金鑰的模數值。它以 Base64urlUInt 編碼值表示。

用途 (**use**)

這個 use 參數描述公開金鑰的用途。在這個範例中，use 值 sig 代表簽章。

b. 搜尋符合您的 JWT 中 kid 之 kid 的公有 JSON Web 金鑰。

3. 使用 JWT 程式庫，來比較發行者的簽章與權杖中的簽章。發行者簽章衍生自 kid 的公有金鑰 (RSA 模數 "n")，格式時與權杖 kid 相符的 jwks.json。您可能需要先將 JWK 轉換成 PEM 格式。以下範例採用 JWT 和 JWK，並且使用 Node.js 程式庫 jsonwebtoken 來驗證 JWT 簽章：

Node.js

```
var jwt = require('jsonwebtoken');
var jwkToPem = require('jwk-to-pem');
var pem = jwkToPem(jwk);
jwt.verify(token, pem, { algorithms: ['RS256'] }, function(err, decodedToken) {
});
```

驗證宣告

驗證 JWT 宣告

1. 利用下列其中一種方法確認權杖尚未過期。

   a. 解碼權杖，並將 exp 宣告與目前時間進行比較。

   b. 如果您的訪問令牌包含aws.cognito.signin.user.admin聲明，請向 API 發送請求，例如GetUser。如果您的權杖已過期，那麼您使用存取權杖進行授權的 API 請求將會傳回錯誤。

   c. 在對 UserInfo 端點 提出的請求中顯示存取權杖。如果您的權杖已過期，您的請求將會傳回錯誤。

2. 在 ID 權杖中的 aud 宣告和在存取權杖中的 client_id 宣告應符合在 Amazon Cognito 使用者集區建立的應用程式用戶端 ID。

3. 發行者 (iss) 宣告應該符合您的使用者集區。例如，在 us-east-1 區域中建立的使用者集區會有以下 iss 值：

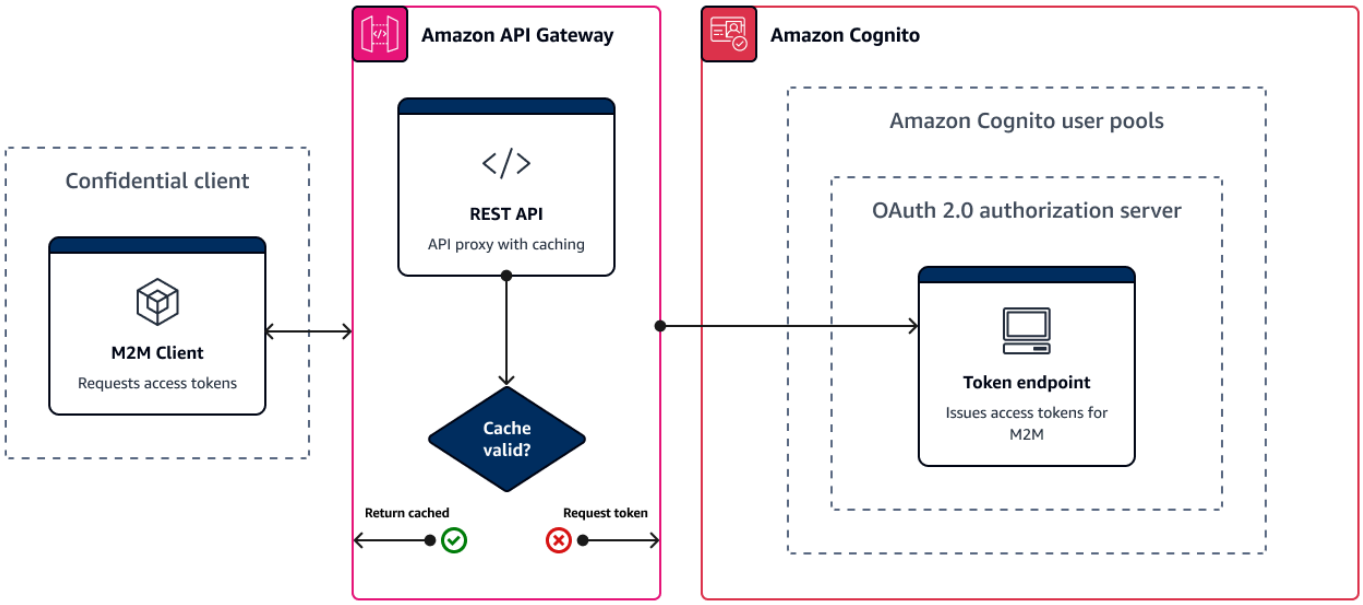
   `https://cognito-idp.us-east-1.amazonaws.com/<userpoolID>`.

4. 檢查 token_use宣告。

   - 如果您在 Web API 操作中只接受存取權杖，其值必須是 access。
   - 如果您只使用 ID 權杖，其值必須是 id。
   - 如果是同時使用 ID 權杖和存取權杖，token_use 宣告必須是 id 或 access。

您現在可以信任權杖內部的宣告。

快取權杖



每次您想要獲取新的 JSON Web 權杖 (JWT) 時，您的應用程式都必須成功完成以下請求之一。

- 從權杖端點請求用戶端憑證或授權碼授與。
- 從您的託管 UI 請求隱含授與。
- 在亞馬遜認可 API 請求中對本地用戶進行身份驗證，例如InitiateAuth。

您可以設定使用者集區，將權杖設定為在幾分鐘、幾小時或幾天後過期。為確保應用程式的效能和可用性，請使用 Amazon Cognito 權杖直到它們過期為止，然後才擷取新的權杖。您為應用程式建立的快取

解決方案可讓權杖保持可用，並在請求率過高時防止 Amazon Cognito 拒絕請求。用戶端應用程式必須將權杖儲存在記憶體快取中。伺服器端應用程式可以新增加密的快取機制來儲存權杖。

當您的使用者集區產生大量的使用者或 machine-to-machine 活動時，您可能會遇到 Amazon Cognito 對您可以發出的權杖請求數量設定的限制。為了減少對 Amazon Cognito 端點發出的請求數量，您可以安全地存放和重複使用驗證資料，或實施指數退避和重試。

驗證資料來自兩種端點類別。Amazon Cognito OAuth 2.0 端點包括權杖端點，用於為用戶端憑證和託管 UI 授權碼請求提供服務。服務端點會回答使用者集區 API 請求，如 `InitiateAuth` 和 `RespondToAuthChallenge`。每種類型的請求都有自己的限制。如需限制的詳細資訊，請參閱 Amazon Cognito 的配額。

使用 Amazon API Gateway 快 machine-to-machine 取存取權杖

透過 API Gateway 權杖快取，您的應用程式可以擴展以回應大於 Amazon Cognito OAuth 端點預設請求率配額的事件。

您可以快取存取權杖，以便您的應用僅在快取權杖過期時請求新的存取權杖。否則，您的快取端點會從快取傳回權杖。如此可防止額外呼叫 Amazon Cognito API 端點。當您使用 Amazon API Gateway 作為 權杖端點 的代理時，您的 API 會回應大部分請求，否則請求會佔用配額，以避免因為速率限制導致請求失敗。

以下 API Gateway 解決方案提供低延遲、低程式碼/無程式碼的權杖快取實作。API Gateway API 在傳輸過程中進行加密，並可選擇是否在靜態時加密。API Gateway 緩存非常適合 OAuth 2.0 客戶端憑據授予，這是一種經常高容量授予類型，可生成訪問令牌以授權 machine-to-machine 和微服務會話。在流量激增導致微服務水平擴展的情況下，您最終可能會在超過用戶池或應用程序客戶端的 AWS 請求率限制的卷中使用相同的客戶端憑據的許多系統。為了保持應用程式可用性和低延遲，快取解決方案是在這種情況下的最佳做法。

在此解決方案中，您可以在 API 中定義快取，以為要在應用程式中請求的每個 OAuth 範圍和應用程式用戶端組合儲存單獨的存取權杖。當您的應用程式發出符合快取金鑰的請求時，您的 API 會以 Amazon Cognito 對符合快取金鑰的第一個請求發出的存取權杖進行回應。當您的快取金鑰持續時間過期時，您的 API 將請求轉發到權杖端點並快取新的存取權杖。

> **Note**
>
> 您的快取金鑰持續時間必須短於應用程式用戶端的存取權杖持續時間。

快取金鑰是您在 `scope` URL 參數中所請求 OAuth 範圍和要求 `Authorization` 標頭的組合。`Authorization` 標題包含您的應用程式用戶端 ID 和用戶端密碼。您無需在應用程式中實作其他邏輯即可實現此解決方案。您只能更新組態以變更使用者集區權杖端點的路徑。

您還可以使ElastiCache 用 Redis 實現令牌緩存。如需使用 AWS Identity and Access Management (IAM) 政策進行精細控制，請考慮 Amazon DynamoDB 快取。

> **Note**
>
> API Gateway 中的快取需支付額外費用。如需詳細資訊，請參閱定價。

使用 API Gateway 設定快取代理

1. 開啟 API Gateway 主控台並建立 REST API。
2. 在 Resources (資源) 中，建立 POST 方法。
   a. 選擇 HTTP Integration type (整合類型)。
   b. 選擇 Use HTTP proxy integration (使用 HTTP 代理整合)。
   c. 輸入 `https://<your user pool domain>/oauth2/token` 的 Endpoint URL (端點 URL)。
3. 在 Resources (資源) 中，設定快取金鑰。
   a. 編輯您 POST 方法的 Method request (方法請求)。
   b. 設定您的 `scope` 參數和 `Authorization` 標題作為您的快取金鑰。
      i. 將查詢字串新增至 URL query string parameters (URL 查詢字串參數) 並為 `scope` 字串選擇 Caching (快取)。
      ii. 將標頭新增至 HTTP request headers (HTTP 請求標頭) 並為 `Authorization` 標頭選擇 Caching (快取)。
4. 在 Stages (階段) 中，設定快取。
   a. 選擇您想要修改的階段。
   b. 在 Settings (設定) 下，選取 Enable API cache (啟用 API 快取)。
   c. 選擇 Cache capacity (快取容量)。
   d. 選擇至少 3600 秒的快取 time-to-live (TTL)。
   e. 清除 需要授權 核取方塊。

5. 在 Stages (階段) 中，記下 Invoke URL (叫用 URL)。
6. 更新您的應用程式以將權杖請求 POST 到 API 的 Invoke URL (叫用網址)，而不是使用者集區的 `/oauth2/token` 端點。

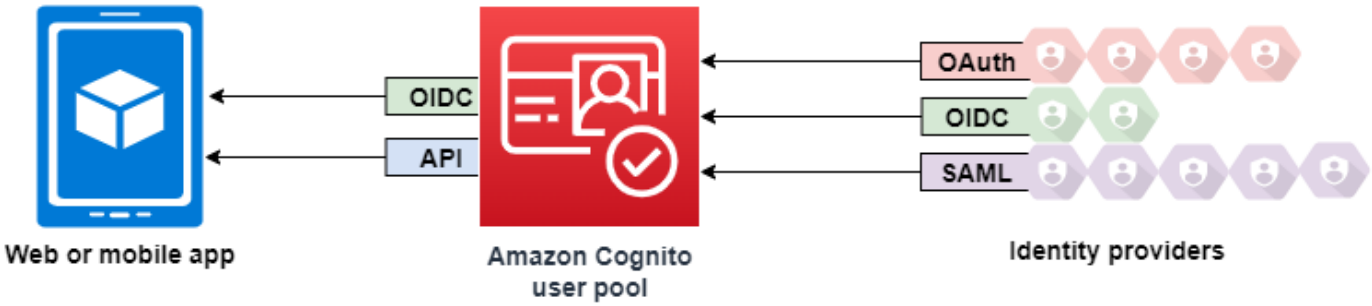
在使用者集區身分驗證成功後存取資源

您的應用程式使用者可以直接透過使用者集區登入，也可以透過第三方身分識別提供者 (IdP) 進行聯合。用戶池管理處理通過 Facebook，谷歌，Amazon 和蘋果以及 OpenID Connect（OIDC）和 SAML 從社交登錄返回的令牌的開銷。 IdPs如需詳細資訊，請參閱 將權杖用於使用者集區。

身分驗證成功後，您的應用程式將會從 Amazon Cognito 接收使用者集區權杖。您可以使用用戶池令牌來：

- 擷取授權 Amazon DynamoDB 和 Amazon S3 AWS 服務 等應用程式資源請求的 AWS 登入資料。
- 提供可撤銷的臨時認證證明。
- 將身份數據填充到應用程序中的用戶配置文件。
- 授權變更使用者集區目錄中已登入的使用者設定檔。
- 使用存取權杖授權使用者資訊的要求。
- 使用訪問令牌授權對受訪問保護的外部 API 後面的數據的請求。
- 使用 Amazon 驗證許可授權存取存放在用戶端或伺服器上的應用程式資產。

如需詳細資訊，請參閱 使用者集區身分驗證流程 及 將權杖用於使用者集區。



主題

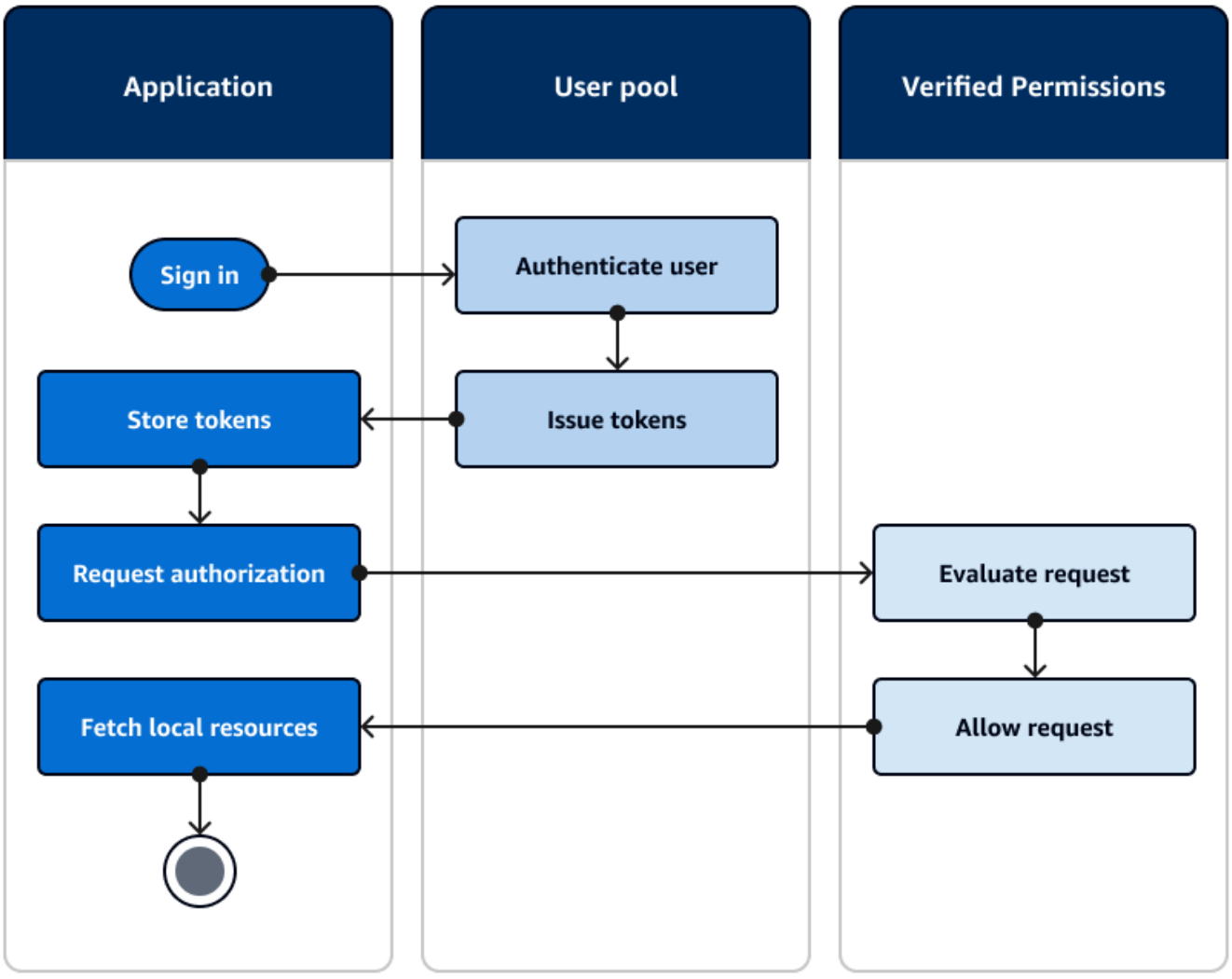
- 使用 Amazon 驗證許可授權存取用戶端或伺服器資源
- 登入後使用 API Gateway 存取資源

- 登入後 AWS 服務 使用身分集區存取

使用 Amazon 驗證許可授權存取用戶端或伺服器資源

您的應用程式可以將權杖從登入的使用者傳遞至 Amazon 驗證許可。已驗證的權限是可擴充、精細的權限管理和授權服務，適用於您已建置的自訂應用程式。Amazon Cognito 使用者集區可以是已驗證許可政策存放區的身分來源。已驗證的權限會從使用者集區權杖中的主體及其屬性`premium_badge.png`，針`GetPhoto`對要求的動作和資源 (例如 to) 做出授權決定。

下圖顯示了應用程序如何在授權請求中將用戶的令牌傳遞給已驗證的權限。



開始使用 Amazon 驗證許可

將使用者集區與已驗證許可整合後，您將獲得所有 Amazon Cognito 應用程式的精細授權中央來源。這樣就消除了對細粒度安全邏輯的需求，否則您必須在所有應用程序之間進行編碼和複製。如需有關使用已驗證權限授權的詳細資訊，請參閱使用 Amazon Verified Permissions 進行授權。

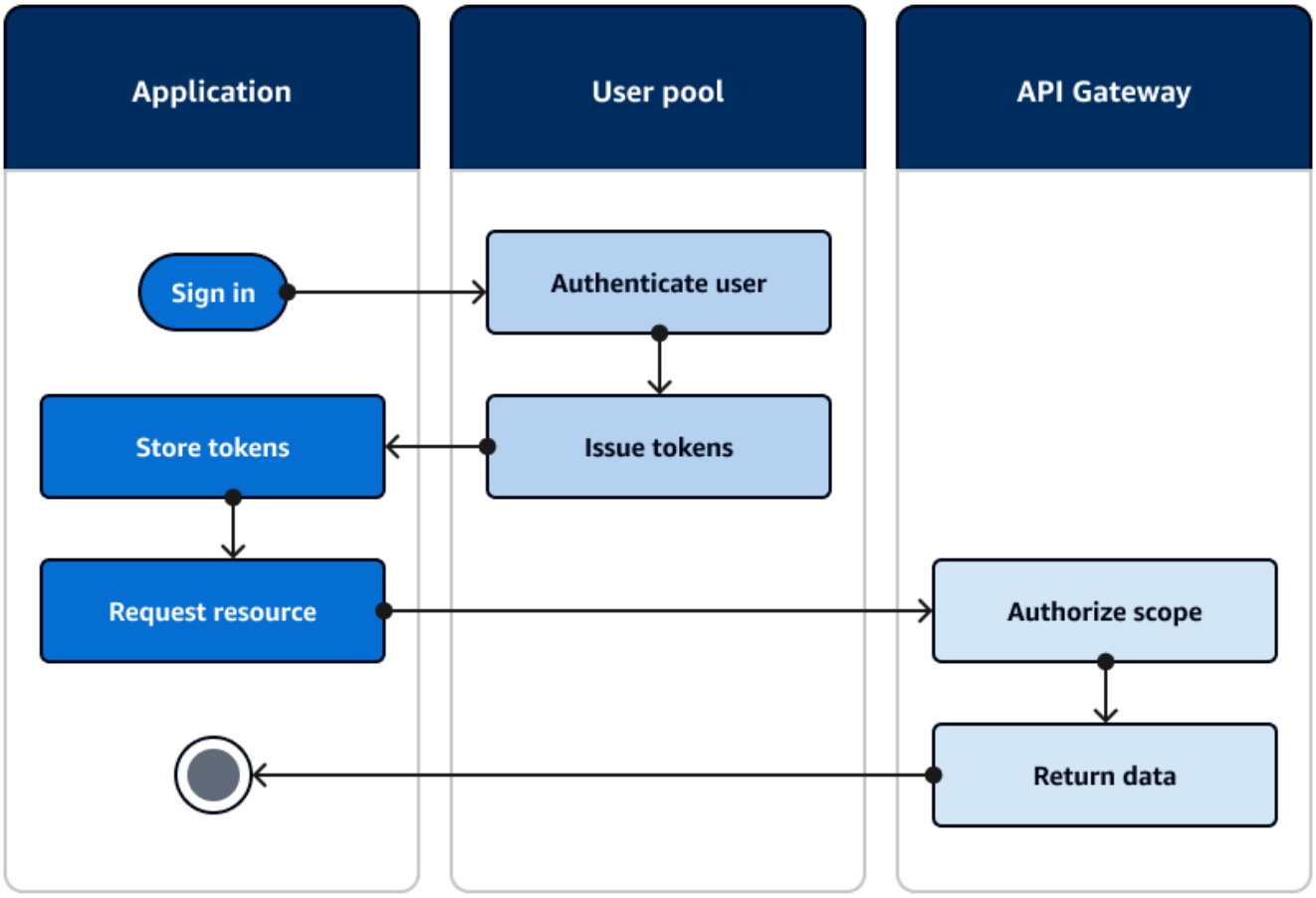
已驗證的權限授權請求需要 AWS 憑證。您可以實作下列一些技術，以安全地將認證套用至授權要求。

- 操作可以在服務器後端存儲密碼的 Web 應用程序。
- 取得已驗證的身分集區認證。
- 通過 access-token-authorized API 代理用戶請求，並將 AWS 憑據附加到請求。

登入後使用 API Gateway 存取資源

Amazon Cognito 使用者集區權杖的常見用途是授權對 API Gateway REST API 的請求。訪問令牌中的 OAuth 2.0 範圍可以授權方法和路徑，HTTP GET例/app_assets如. ID 令牌可以作為 API 的通用身份驗證，並且可以將用戶屬性傳遞給後端服務。API Gateway 具有額外的自訂授權選項，例如 HTTP API 的 JWT 授權程式，以及可套用更精細邏輯的 Lambda 授權者。

下圖說明了一個應用程序，該應用程序正在訪問令牌中使用 OAuth 2.0 範圍獲得對 REST API 的訪問權限。



您的應用程序必須從經過身份驗證的會話中收集令牌，並將其作為承載令牌添加到請求中的`Authorization`標題中。配置您為 API，路徑和方法配置的授權者以評估令牌內容。只有當請求符合您為授權者設定的條件時，API Gateway 才會傳回資料。

API Gateway API 可以核准來自應用程式存取的一些潛在方式為：

- 訪問令牌包含正確的 OAuth 2.0 範圍。適用於 REST API 的 Amazon Cognito 使用者集區授權者是一種常見的實作，具有較低的進入障礙。您也可以向此類授權者評估要求的主體、查詢字串參數和標頭。
- ID 令牌有效且未過期。將 ID 權杖傳遞給 Amazon Cognito 授權者時，您可以對應用程式伺服器上的 ID 權杖內容執行其他驗證。
- 存取或 ID 權杖中的群組、宣告、屬性或角色符合您在 Lambda 函數中定義的需求。Lambda 授權者會剖析要求標頭中的權杖，並對其進行授權決策進行評估。您可以在函數中建構自訂邏輯，或向 Amazon 驗證許可發出 API 請求。

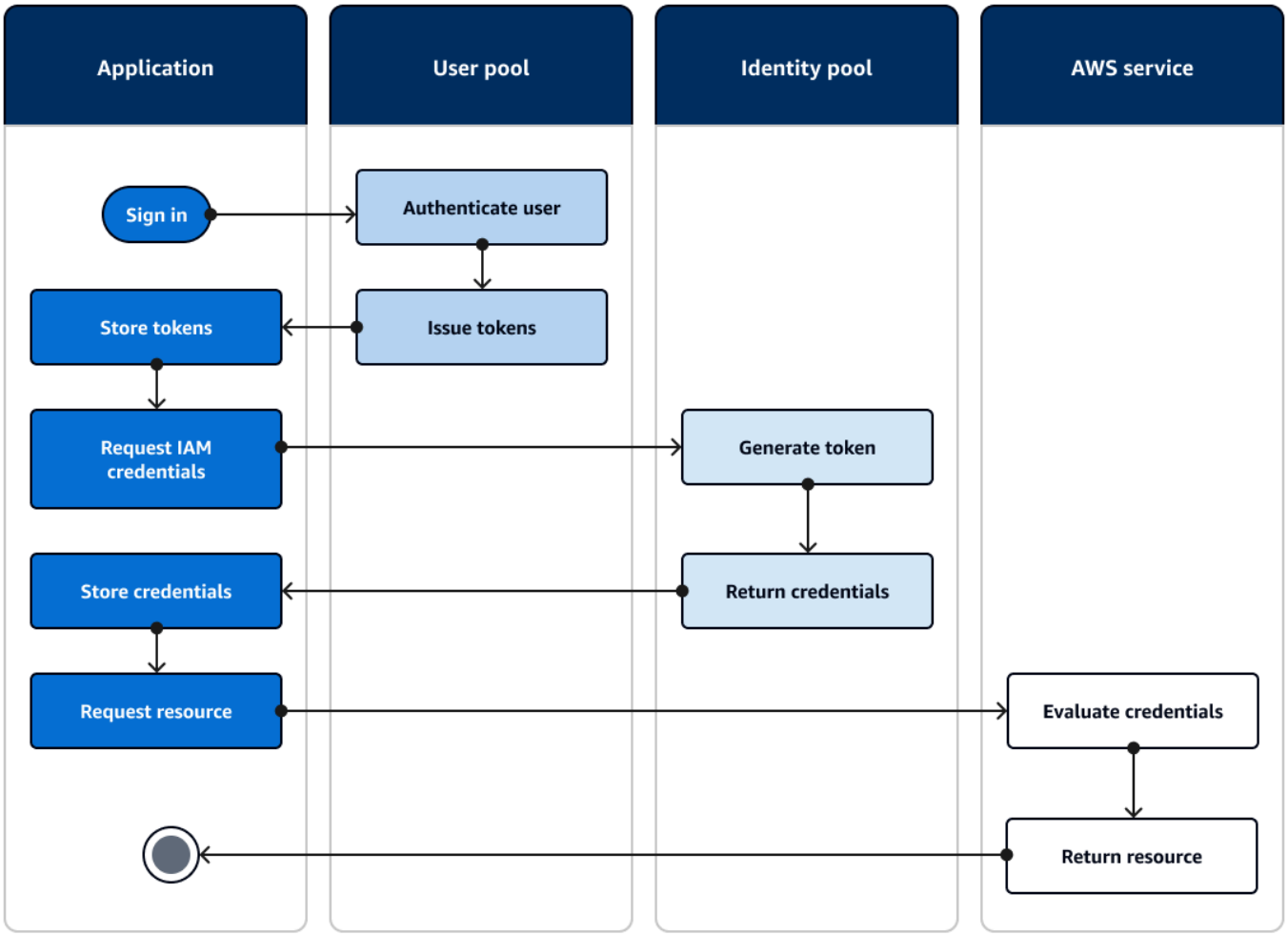
您也可以使用來自使用者集區的權杖來授權對 AWS AppSync GraphQL API 的要求。

登入後 AWS 服務 使用身分集區存取

使用者使用使用者集區登入後，他們可以使 AWS 服務 用從身分集區核發的臨時 API 認證進行存取。

您的 Web 或行動應用程式會接收來自使用者集區的權杖。當您將使用者集區設定為身分識別集區的身分提供者時，身分集區會將權杖交換為暫時 AWS 登入資料。這些登入資料的範圍可以設定為 IAM 角色及其政策，讓使用者能夠存取一組有限的 AWS 資源。如需詳細資訊，請參閱 身分集區 (聯合身分) 驗證流程。

下圖顯示應用程式如何使用使用者集區登入、擷取身分集區認證，以及從 AWS 服務.



您可以使用身分集區認證來：

- 使用您的使用者自己的登入資料向 Amazon 驗證許可提出細微的授權請求。

- Connect 到 Amazon API Gateway REST API 或用於授權與 IAM 連線的 AWS AppSync GraphQL API。
- Connect 到授權與 IAM 連線的資料庫後端，例如 Amazon DynamoDB 或 Amazon RDS。
- 從 Amazon S3 儲存貯體擷取應用程式資產。
- 使用 Amazon WorkSpaces 虛擬桌面啟動工作階段。

身分集區不僅在具有使用者集區的已驗證工作階段內運作。他們也直接接受來自第三方身分識別提供者的驗證，並可以為未經驗證的來賓使用者產生

如需有關將身分集區與使用者集區群組一起使用來控制 AWS 資源存取權的詳細資訊，請參閱新增群組至使用者集區和使用以角色為基礎的存取控制。另外，如需身分集區的詳細資訊 AWS Identity and Access Management，請參閱身分集區概念。

設定使用者集區 AWS Management Console

建立 Amazon Cognito 使用者集區，並記下每個用戶端應用程式的 User Pool ID (使用者集區 ID) 和App Client ID (應用程式用戶端 ID)。如需有關建立使用者集區的詳細資訊，請參閱 使用者集區入門。

設定身分集區 AWS Management Console

下列程序說明如何使用將身分識別集區與一或多個使用者集區和用戶端應用程式整合。 AWS Management Console

若要新增 Amazon Cognito 使用者集區身分提供者 (IdP)

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。
3. 選取 新增身分供應商。
4. 選擇 Amazon Cognito 使用者集區。
5. 輸入使用者集區 ID 和應用程式用戶端 ID。
6. 若要設定 Amazon Cognito 向已通過此提供者進行身分驗證的使用者發布憑證時的角色，請設定角色設定。

    a. 您可以為該 IdP 的使用者提供您在設定已驗證角色時所設定的預設角色，或者您可以選擇具有規則的角色。使用 Amazon Cognito 使用者集區 IdP，您還可以選擇權杖中具有 preferred_role 的角色。如需 `cognito:preferred_role` 宣告的詳細資訊，請參閱 指定優先順序值給群組。

i. 如果您選擇選擇含規則的角色，請輸入來自使用者驗證的來源宣告、要用來比較宣告與規則的操作員、將造成符合此角色選擇的值，以及當「角色」指派相符時要指派的「角色」。選取 新增另一項 以根據不同的條件建立其他規則。

ii. 如果您選擇在權杖中選擇具有偏好角色宣告的角色，Amazon Cognito 會針對使用者宣告中的角色發出登入資料。`cognito:preferred_role`如果沒有偏好的角色宣告存在，Amazon Cognito 會根據您的角色解析發出憑證。

b. 選擇 角色解析。當您的使用者宣告與您的規則不符時，您可以拒絕憑證或向 已驗證角色 發出憑證。

7. 若要變更透過此提供者驗證使用者，Amazon Cognito 發布憑證時指派的主要索引標籤，請設定 存取控制的屬性。

- 若不套用主要索引標籤，請選擇 非作用中。
- 若要根據 sub 和 aud 宣告套用主要索引標籤，請選擇 使用預設對應。
- 若要建立您自己的自訂屬性結構描述至主要索引標籤，請選擇 使用自訂對應。然後，輸入您要從每個 宣告 中獲取的 標籤金鑰，顯示於索引標籤當中。

8. 選取儲存變更。

使用身分集區來整合使用者集區

驗證您的應用程式使用者之後，將該使用者的身分權杖新增至登入資料供應商中的登入對應。供應商名稱取決於您的 Amazon Cognito 使用者集區 ID。其結構如下：

```
cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>
```

您可以<region>從「使用者集區 ID」衍生值。例如，如果使用者集區識別碼為`us-east-1_EXAMPLE1`，則<region>為`us-east-1`. 如果使用者集區識別碼為`us-west-2_EXAMPLE2`，則<region>為`us-west-2`.

JavaScript

```
var cognitoUser = userPool.getCurrentUser();

if (cognitoUser != null) {
 cognitoUser.getSession(function(err, result) {
  if (result) {
   console.log('You are now logged in.');
```

```
   // Add the User's Id Token to the Cognito credentials login map.
   AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'YOUR_IDENTITY_POOL_ID',
    Logins: {
     'cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>':
 result.getIdToken().getJwtToken()
    }
   });
  }
 });
}
```

Android

```
cognitoUser.getSessionInBackground(new AuthenticationHandler() {
 @Override
 public void onSuccess(CognitoUserSession session) {
  String idToken = session.getIdToken().getJWTToken();

  Map<String, String> logins = new HashMap<String, String>();
  logins.put("cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>",
 session.getIdToken().getJWTToken());
  credentialsProvider.setLogins(logins);
 }

});
```

iOS - objective-C

```
AWSServiceConfiguration *serviceConfiguration = [[AWSServiceConfiguration alloc]
 initWithRegion:AWSRegionUSEast1 credentialsProvider:nil];
AWSCognitoIdentityUserPoolConfiguration *userPoolConfiguration =
 [[AWSCognitoIdentityUserPoolConfiguration alloc] initWithClientId:@"YOUR_CLIENT_ID"
  clientSecret:@"YOUR_CLIENT_SECRET" poolId:@"YOUR_USER_POOL_ID"];
[AWSCognitoIdentityUserPool
 registerCognitoIdentityUserPoolWithConfiguration:serviceConfiguration
 userPoolConfiguration:userPoolConfiguration forKey:@"UserPool"];
AWSCognitoIdentityUserPool *pool = [AWSCognitoIdentityUserPool
 CognitoIdentityUserPoolForKey:@"UserPool"];
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider
 alloc] initWithRegionType:AWSRegionUSEast1 identityPoolId:@"YOUR_IDENTITY_POOL_ID"
 identityProviderManager:pool];
```

iOS - swift

```
let serviceConfiguration = AWSServiceConfiguration(region: .USEast1,
 credentialsProvider: nil)
let userPoolConfiguration = AWSCognitoIdentityUserPoolConfiguration(clientId:
 "YOUR_CLIENT_ID", clientSecret: "YOUR_CLIENT_SECRET", poolId: "YOUR_USER_POOL_ID")
AWSCognitoIdentityUserPool.registerCognitoIdentityUserPoolWithConfiguration(serviceConfigura
 userPoolConfiguration: userPoolConfiguration, forKey: "UserPool")
let pool = AWSCognitoIdentityUserPool(forKey: "UserPool")
let credentialsProvider = AWSCognitoCredentialsProvider(regionType: .USEast1,
 identityPoolId: "YOUR_IDENTITY_POOL_ID", identityProviderManager:pool)
```

使用 Amazon Cognito 使用者集區安全性功能

您可以新增多重要素驗證 (MFA) 至使用者集區，以保護您的使用者的身分。MFA 會新增第二個身分驗證要素，因此您的使用者集區不會僅依賴使用者名稱和密碼進行驗證。您可以使用文字簡訊或以時間為基礎的一次性密碼 (TOTP)，作為登入使用者的第二個要素。您也可以使用適應性身分驗證及其風險型模型，預測您需要另外一個身分驗證要素的可能時機。使用者集區進階安全性功能包括適應性身分驗證和針對憑證洩漏的保護。

主題

- 將 MFA 新增到使用者集區
- 將進階安全性新增到使用者集區
- 建立 AWS WAF Web ACL 與使用者集區的關聯
- 使用者集區大小寫區分
- 使用者集區刪除保護
- 管理使用者存在錯誤回應

將 MFA 新增到使用者集區

多重要素驗證 (MFA) 可提高應用程式的安全性。它將一個您擁有的身分驗證因素新增到您知道的使用者名稱和密碼因素。您可以選擇文字簡訊或以時間為基礎的一次性密碼 (TOTP)，作為登入使用者的第二要素。

> **Note**
>
> 新使用者首次登入您的應用程式時，Amazon Cognito 會發出 OAuth 2.0 權杖，即使您的使用者集區需要 MFA 也是如此。您的使用者首次登入時的第二個身分驗證要素是確認 Amazon Cognito 傳送給他們的驗證訊息。如果您的使用者集區需要 MFA，Amazon Cognito 會提示您的使用者註冊一個額外的登入要素，以便在第一次之後的每次登入嘗試期間使用。

透過適應性身分驗證，您可以將使用者集區設定成要求第二要素身分驗證，以便回應更高的風險層級。若要將適應性身分驗證新增到使用者集區，請參閱 將進階安全性新增到使用者集區。

當您將使用者集區的多重要素驗證 (MFA) 設定為 `required` 時，所有使用者都必須完成 MFA 才能登入。每個使用者都至少必須設定一個 MFA 要素 (例如簡訊或 TOTP) 才能登入。當您將 MFA 設定為 `required`，您必須在新使用者加入時包含 MFA 設定，以便您的使用者集區允許這些使用者登入。

如果您啟用簡訊作為 MFA 要素，您可以在註冊期間要求使用者提供電話號碼並驗證這些電話號碼。如果您將 MFA 設定為 `required`，且只支援將簡訊作為要素，則使用者必須提供電話號碼。沒有電話號碼的使用者需要您支援將電話號碼新增至其設定檔，然後使用者才能登入。您可以將未經驗證的電話號碼使用於文字簡訊 MFA。這些號碼將會在 MFA 成功後收到驗證狀態。

如果您已將 MFA 設定為必要，並啟用簡訊和 TOTP 做為支援的驗證方法，Amazon Cognito 會提示沒有電話號碼的新使用者設定 TOTP MFA。如果您已將 MFA 設定為必要，而且您啟用的唯一 MFA 方法是 TOTP，Amazon Cognito 會在所有新使用者第二次登入時提示他們設定 TOTP MFA。Amazon Cognito 在設定 TOTP MFA 以回應InitiateAuth和 AdminInitiateAuthAPI 操作時產生了一項挑戰。

若您將 MFA 設定為必要，則託管 UI 會提示使用者設定 MFA。若您在使用者集區中將 MFA 設定為選用，則託管 UI 不會提示使用者。若要使用選用的 MFA，您必須在應用程式中建置一個介面來提示您的使用者選取要設定 MFA，然後引導他們進行 API 輸入，以驗證他們的另一項登入要素。

嘗試提供 MFA 代碼失敗五次之後，Amazon Cognito 會啟動指數遞增封鎖持續時間程序，如 使用者集區身分驗證流程 中所述。

主題

- 必要條件
- 設定多重要素驗證
- 文字簡訊 MFA。
- TOTP 軟體權杖 MFA

必要條件

設定 MFA 之前，請考慮下列事項：

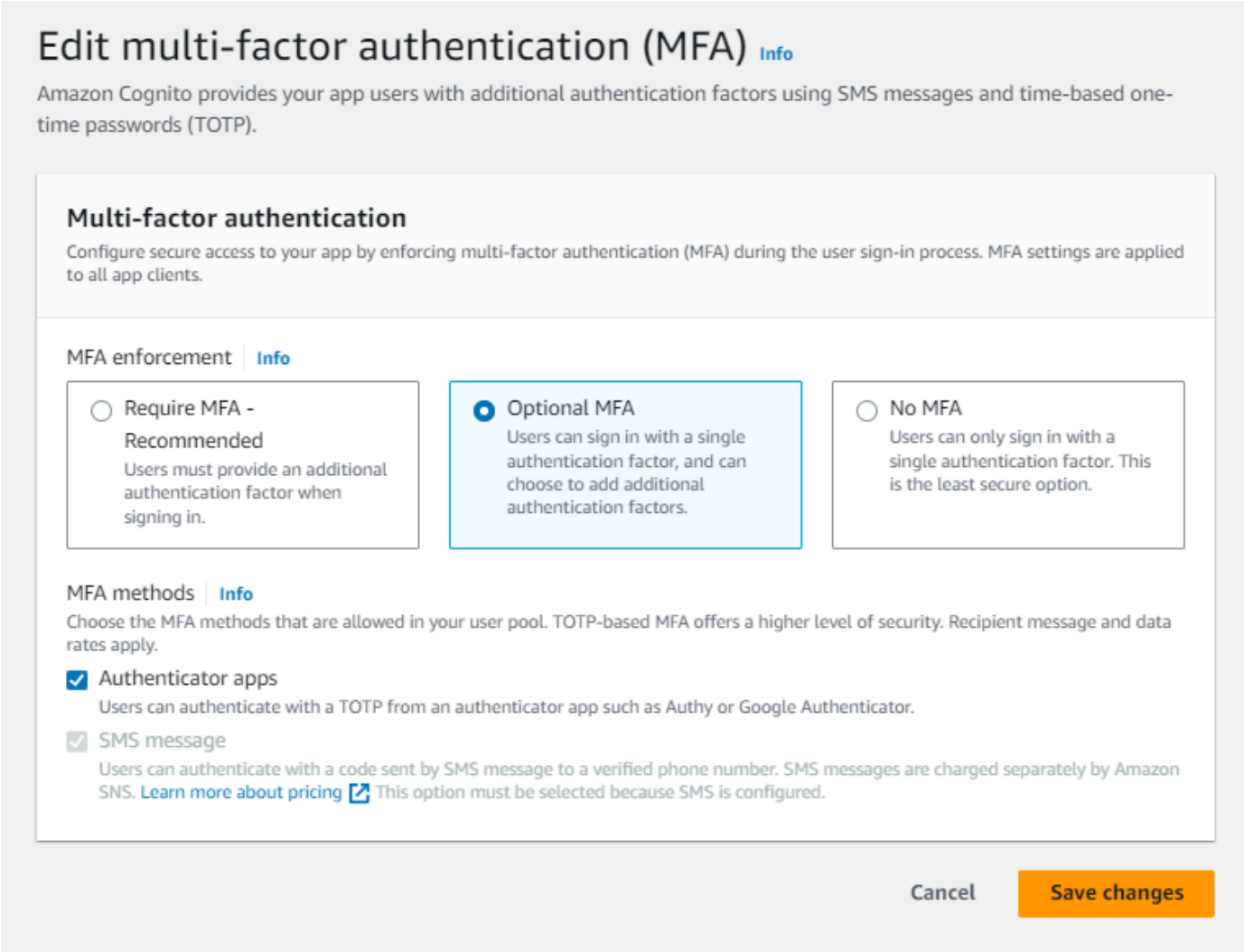
- 當您在使用者集區中啟用 MFA 並選擇 SMS text message (文字簡訊) 作為第二個要素時，可以將簡訊傳送到您尚未在 Amazon Cognito 中驗證的電話號碼屬性。在您的使用者完成簡訊 MFA 之後，Amazon Cognito 會將其 `phone_number_verified` 屬性設定為 `true`。
- 如果您的帳戶位於包含使用者集區之 Amazon Simple Notification Service (Amazon SNS) 資源的 SMS 沙箱中，您必須先驗證 Amazon SNS 中的電話號碼，才能傳送簡訊。 AWS 區域 如需詳細資訊，請參閱 Amazon Cognito 使用者集區的簡訊設定。
- 使用進階安全功能時必須啟用 MFA，並且在 Amazon Cognito 使用者集區主控台中將其設定為選用。如需詳細資訊，請參閱 將進階安全性新增到使用者集區。

設定多重要素驗證

您可以在 Amazon Cognito 主控台中設定 MFA。

在 Amazon Cognito 主控台中設定 MFA

1. 登入 Amazon Cognito 主控台。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。
4. 選擇 Sign-in experience (登入體驗) 索引標籤。找到 Multi-factor authentication (多重要素驗證)，然後選擇 Edit (編輯)
5. 選擇您想要與使用者集區搭配使用的 MFA 強制執行方法。



a. Require MFA (需要 MFA)。使用者集區中的所有使用者都必須使用額外的簡訊代碼或以時間為基礎的一次性密碼 (TOTP) 要素來登入。

b. Optional MFA (選用 MFA)：您可以讓您的使用者選擇註冊額外的登入要素，但仍然允許未設定 MFA 的使用者登入。如果您正在使用適應性身分驗證，請選擇此選項。如需有關適應性身分驗證的詳細資訊，請參閱 將進階安全性新增到使用者集區。

c. No MFA (沒有 MFA)。您的使用者無法註冊額外的登入要素。

6. 選擇您應用程式中支援的 MFA methods (MFA 方法)。您可以設定 SMS message (SMS 訊息) 或產生 TOTP 的 Authenticator apps (驗證器應用程式) 做為第二個要素。我們建議您實作以 TOTP 為基礎的 MFA，以便能夠使用 SMS 訊息復原帳戶。

7. 如果使用文字簡訊作為第二個要素，且您尚未設定與 Amazon Simple Notification Service (Amazon SNS) 搭配用於簡訊的 IAM 角色，則您可以在主控台中建立一個角色。在您的使用者集區 Messaging (簡訊) 索引標籤中，找到 SMS 並選擇 Edit (編輯)。您也可以使用允許 Amazon Cognito 為您將簡訊傳送給使用者的現有角色。如需詳細資訊，請參閱 IAM 角色。

8. 選擇 Save changes (儲存變更)。

文字簡訊 MFA。

如果是在啟用 MFA 的情況下登入，這時使用者要先輸入使用者名稱和密碼並提交。而用戶端應用程式會收到 `getMFA` 回應，指出授權碼已傳送到哪裡。用戶端應用程式應該向使用者指出找到授權碼之處 (例如授權碼傳送到哪個電話號碼)。接著會提供輸入授權碼的表單。最後，用戶端應用程式會提交授權碼以完成登入程序。系統會遮蔽目的地，只留下電話號碼末四碼。如果應用程式使用 Amazon Cognito 託管 UI，則其會顯示一個頁面讓使用者輸入 MFA 代碼。

文字簡訊授權碼的有效期間，是您為應用程式用戶端設定的 Authentication flow session duration (驗證流程工作階段持續時間)。

當您在 App clients and analytics (應用程式用戶端和分析) 下修改應用程式用戶端時，可在 Amazon Cognito 主控台的 App integration (應用程式整合) 索引標籤中設定驗證流程工作階段的持續時間。您也可以在 `CreateUserPoolClient` 或 `UpdateUserPoolClient` API 請求中設定驗證流程工作階段持續時間。如需詳細資訊，請參閱 使用者集區身分驗證流程。

如果使用者已無法存取接收文字簡訊 MFA 代碼的裝置，這時其必須請求客服中心協助。具有必要 AWS 帳戶 權限的管理員可以變更使用者的電話號碼，但只能透過 AWS CLI 或 API 來變更使用者的電話號碼。

使用者成功完成文字簡訊 MFA 流程後，其電話號碼也會標示為已驗證。

> **Note**
>
> MFA 的簡訊需另外收費。(傳送驗證碼到電子郵件地址免費)。如需 Amazon SNS 定價的相關資訊，請參閱全球 SMS 定價。如需可使用 SMS 簡訊的國家/地區最新清單，請參閱支援的區域和國家。

> **Important**
>
> 為了確保已傳送文字簡訊來驗證電話號碼和簡訊 MFA，您必須向 Amazon SNS 請求提高費用限制。
>
> Amazon Cognito 使用 Amazon SNS 來傳送簡訊給使用者。Amazon SNS 傳送的簡訊數量受限於費用限制。您可以針對 AWS 帳戶和個別訊息指定支出限制，而這些限制僅適用於傳送簡訊的費用。

每個帳戶的預設費用限制 (若未指定) 為每個月 1.00 美金。如果您想提高限制，請在 AWS Support 中心提交 SNS 限制提高案例。針對 New limit value (新的限制值) 部分，輸入您想要的每月費用限制。在 Use Case Description (使用案例說明) 欄位中，說明您要求提高簡訊的每月費用限制。

若要在使用者集區中新增 MFA，請參閱 將 MFA 新增到使用者集區。如需使用者集區中使用 Amazon SNS 傳送簡訊的詳細資訊，請參閱Amazon Cognito 使用者集區的簡訊設定。

TOTP 軟體權杖 MFA

當您在使用者集區中設定 TOTP 軟體字符 MFA 時，使用者以使用者名稱和密碼登入，然後使用 TOTP 完成身分驗證。在您的使用者設定並驗證使用者名稱和密碼之後，他們就可以為 MFA 啟用 TOTP 軟體權杖。如果您的應用程式使用 Amazon Cognito 託管的使用者界面來登入使用者，您的使用者將會提交使用者名稱和密碼，然後在額外的登入頁面上提交 TOTP 密碼。

您可以在 Amazon Cognito 主控台中或使用 Amazon Cognito API 操作，為您的使用者集區啟用 TOTP MFA。在使用者集區層級，您可以呼叫SetUserPoolMfaConfig以設定 MFA 並啟用 TOTP MFA。

> **Note**
>
> 如果您尚未啟用使用者集區的 TOTP 軟體權杖 MFA，Amazon Cognito 將無法使用該權杖建立關聯或驗證使用者。在此情況下，使用者會收到 `SoftwareTokenMFANotFoundException` 例外狀況和 `Software Token MFA has not been enabled by the userPool` 描述。如果停用使用者集區的軟體權杖 MFA，則先前已關聯並驗證 TOTP 權杖的使用者可以繼續將其用於 MFA。

為使用者設定 TOTP 是包含多步驟的程序，在此期間，您的使用者會收到一個秘密代碼，其需要輸入一次性密碼來進行驗證。接著，您可以為使用者啟用 TOTP MFA，或將 TOTP 設定為使用者偏好的 MFA 方法。

當您將使用者集區設定為要求 TOTP MFA，而您的使用者在託管 UI 中註冊您的應用程式時，Amazon Cognito 會自動執行使用者程序。Amazon Cognito 會提示您的使用者選擇 MFA 方法、顯示 QR 碼以設定其驗證器應用程式，並驗證其 MFA 註冊。在您允許使用者在 SMS 和 TOTP MFA 之間進行選擇的使用者集區中，Amazon Cognito 也會為您的使用者提供方法選擇。如需託管 UI 註冊體驗的詳細資訊，請參閱 如何在 Amazon Cognito 託管 UI 中註冊新帳戶。

> **Important**
>
> 當您擁有與使用者集區相關聯的 AWS WAF Web ACL，且 Web ACL 中的規則顯示驗證碼時，這可能會導致託管 UI TOTP 註冊中無法復原的錯誤。若要建立具有 CAPTCHA 動作且不會影響託管 UI TOTP 的規則，請參閱 為託管的使用者介面 TOTP MFA 設定您的 AWS WAF 網頁 ACL。如需有關 AWS WAF 網路 ACL 和 Amazon Cognito 的詳細資訊，請參閱。建立 AWS WAF Web ACL 與使用者集區的關聯

若要在您使用 Amazon Cognito API 的自訂 UI 中實作 TOTP MFA，請參閱在 Amazon Cognito 使用者集區 API 中為使用者設定 MFA。

若要在使用者集區中新增 MFA，請參閱 將 MFA 新增到使用者集區。

TOTP MFA 考量與限制

1. Amazon Cognito 透過會產生 TOTP 代碼的驗證器應用程式來支援軟體字符 MFA。Amazon Cognito 不支援以硬體為基礎的 MFA。
2. 當您的使用者集區向尚未設定 TOTP 的使用者要求 TOTP 時，使用者會收到一次性存取權杖，您的應用程式可用它來為該使用者啟用 TOTP MFA。後續的登入嘗試將會失敗，直到使用者註冊其他 TOTP 登入要素。
   - 在您的使用者集區中使用 `SignUp` API 操作或透過託管的使用者界面進行註冊的使用者，在完成註冊時會收到一次性權杖。
   - 在建立使用者且該使用者設定其初始密碼之後，Amazon Cognito 會從託管的使用者界面將一次性權杖發放給使用者。如果您為使用者設定了永久性密碼，Amazon Cognito 會在使用者首次登入時發放一次性權杖。
   - Amazon Cognito 不會向使用或 API 操作登入的管理員建立的使用者發出一次性權杖。InitiateAuthAdminInitiateAuth您的使用者在成功完成設定其初始密碼的挑戰之後，或者如果您為使用者設定了永久性密碼，Amazon Cognito 會立即要求使用者設定 MFA。
3. 如果使用者集區中需要 MFA 的使用者已收到一次性存取權杖，但尚未設定 TOTP MFA，則該使用者在設定 MFA 之前無法使用託管的使用者界面登入。您可以AdminInitiateAuth在AssociateSoftwareToken請求中使用來自`MFA_SETUP`挑戰的`session`響應值，而不InitiateAuth是訪問令牌。
4. 如果您的使用者已設定 TOTP，即使您稍後停用使用者集區的 TOTP，使用者仍可將其用於 MFA。
5. Amazon Cognito 僅接受使用 SHA-1 雜湊函數產生代碼的驗證器應用程式的 TOTP。使用 SHA-256 雜湊產生的代碼會傳回 `Code mismatch` 錯誤。

在 Amazon Cognito 使用者集區 API 中為使用者設定 MFA

當使用者首次登入時，您的應用程式將使用其一次性存取字符以產生 TOTP 私有金鑰，並以文字或 QR 碼格式呈現給使用者。您的使用者設定其驗證器應用程式，並為後續的登入嘗試提供 TOTP。您的應用程式或託管的使用者界面在 MFA 挑戰回應中向 Amazon Cognito 顯示 TOTP。

主題

- 與 TOTP 軟體字符關聯
- 驗證 TOTP 字符
- 使用 TOTP MFA 來登入
- 移除 TOTP 權杖

與 TOTP 軟體字符關聯

若要關聯 TOTP 權杖，請傳送秘密代碼給您的使用者，秘密代碼必須使用一次性密碼進行驗證。關聯權杖需要三個步驟。

1. 當您的使用者選擇 TOTP 軟體 Token MFA 時，請呼叫AssociateSoftwareToken以傳回使用者帳戶產生的唯一共用密鑰代碼。您可以使 AssociateSoftwareToken 用訪問令牌或會話字符串進行授權。
2. 您的應用程式向使用者顯示私有金鑰或是從私有金鑰產生的 QR 代碼。您的使用者必須將金鑰輸入至會產生 TOTP 的應用程式中，例如 Google Authenticator。您可以使用 libqrencode 產生 QR 碼。
3. 您的使用者輸入金鑰，或將 QR 代碼掃描至驗證器應用程式 (如 Google Authenticator) 中，然後該應用程式開始產生代碼。

驗證 TOTP 字符

接下來，驗證 TOTP 權杖。向使用者請求範本代碼並將其提供給 Amazon Cognito 服務，以確認使用者是否已成功產生 TOTP 代碼，如下所示。

1. 您的應用程式會提示使用者輸入代碼，以證明其已正確設定驗證器應用程式。
2. 使用者的驗證器應用程式會顯示一個臨時密碼。驗證器應用程式的密碼以您給予使用者的私密金鑰為基礎。
3. 您的使用者輸入其臨時密碼。您的應用程式在 `VerifySoftwareToken` API 請求中，將臨時密碼傳遞給 Amazon Cognito。
4. Amazon Cognito 保留了與使用者關聯的私密金鑰，產生 TOTP 並將其與您的使用者提供的私密金鑰進行比較。如果兩者相符，`VerifySoftwareToken` 會傳回 `SUCCESS` 回應。

5. Amazon Cognito 將 TOTP 要素與使用者建立關聯。
6. 如果 `VerifySoftwareToken` 操作傳回 `ERROR` 回應，請確定使用者的時鐘是正確的，而且沒有超過最大重試次數。Amazon Cognito 接受嘗試之前或之後 30 秒內的 TOTP 權杖，以解決微小的時鐘偏差問題。解決問題後，請再次嘗試此 VerifySoftwareToken 作業。

使用 TOTP MFA 來登入

此時，您的使用者會使用以時間為基礎的一次性密碼登入。程序如下。

1. 您的使用者要輸入其使用者名稱和密碼來登入您的用戶端應用程式。
2. 您的應用程式會叫用 TOTP MFA 挑戰，並提示使用者輸入臨時密碼。
3. 使用者從關聯的 TOTP 產生應用程式取得臨時密碼。
4. 使用者在您的用戶端應用程式中輸入 TOTP 代碼。您的應用程式通知 Amazon Cognito 服務加以驗證。對於每個登錄，RespondToAuthChallenge應該調用以獲得對新的 TOTP 身份驗證挑戰的響應。
5. 如果權杖經過 Amazon Cognito 驗證，則登入成功，使用者可以繼續進行身分驗證流程。

移除 TOTP 權杖

最後，您的應用程式應允許使用者停用其 TOTP 組態。您目前無法刪除使用者的 TOTP 軟體權杖。若要取代使用者的軟體權杖，請關聯並驗證新的軟體權杖。若要停用使用者的 TOTP MFA，請撥打 M SetUserFP 參考來修改您的使用者不使用 MFA，或僅使用 SMS MFA。

1. 在您的應用程式中，為想要重設 MFA 的使用者建立一個介面。在此介面中提示使用者輸入其密碼。
2. 如果 Amazon Cognito 傳回一項 TOTP MFA 挑戰，請使用 MFP 參考來更新使用者的 MFA 偏好設定。SetUser
3. 在您的應用程式中，告知使用者他們已停用 MFA，並提示他們再次登入。

為託管的使用者介面 TOTP MFA 設定您的 AWS WAF 網頁 ACL

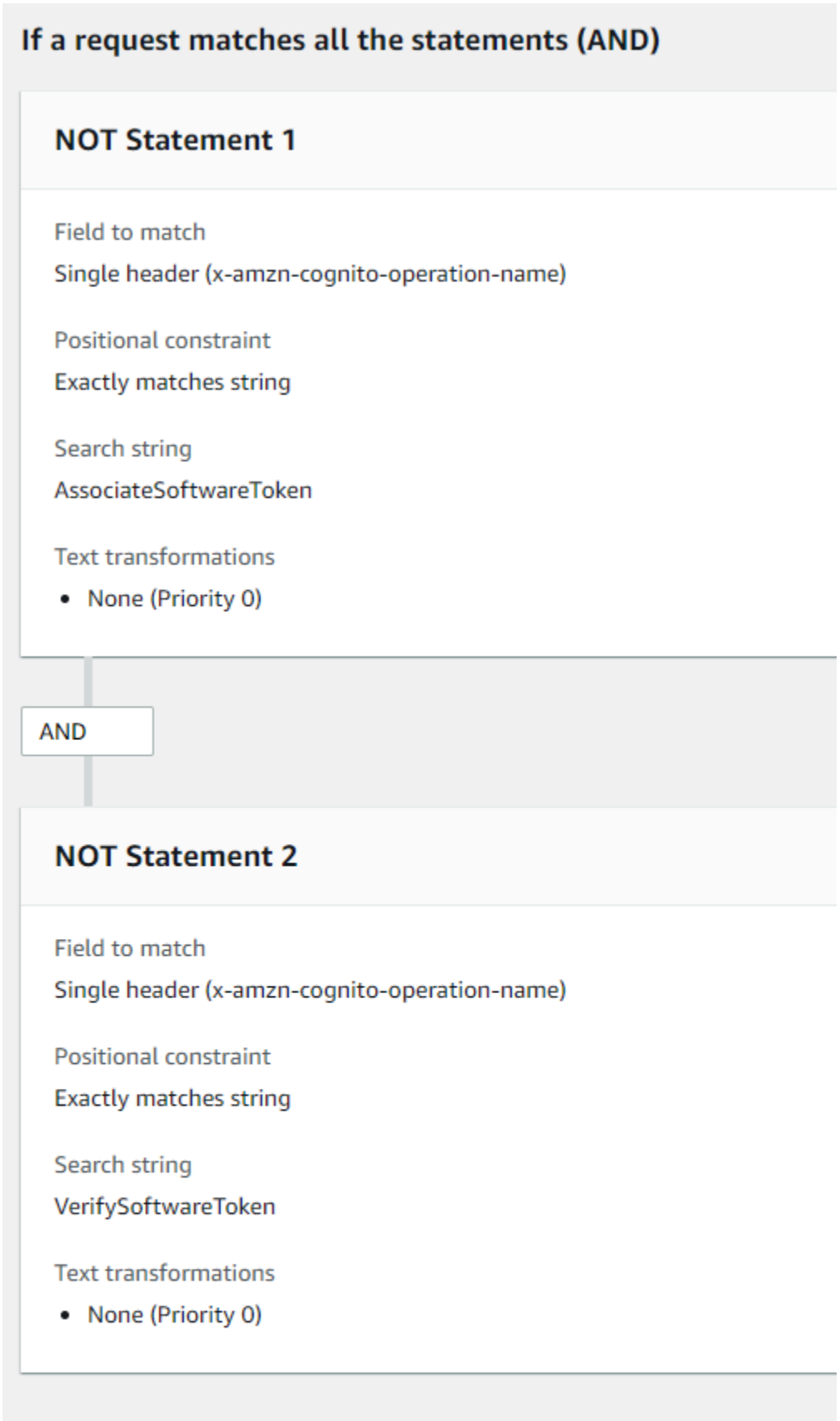
當您擁有與使用者集區相關聯的 AWS WAF Web ACL，且 Web ACL 中的規則顯示驗證碼時，這可能會導致託管 UI TOTP 註冊中無法復原的錯誤。 AWS WAF 驗證碼規則只會以這種方式影響託管 UI 中的 TOTP MFA。SMS MFA 不受影響。

當您的 CAPTCHA 規則無法讓使用者完成 TOTP MFA 設定時，Amazon Cognito 會顯示下列錯誤。

Request not allowed due to WAF captcha. (由於 WAF captcha，不允許請求。)

當 AWS WAF 提示輸入 CAPTCHA 以AssociateSoftwareToken回應您的使用者集區在背景發出的 VerifySoftwareTokenAPI 要求時，就會產生此錯誤。若要建立具有 CAPTCHA 動作且不會影響託管 UI TOTP 的規則，請在您的規則中排除 CAPTCHA 動作中 `AssociateSoftwareToken` 和 `VerifySoftwareToken` 的 `x-amzn-cognito-operation-name` 標頭值。

下列螢幕擷取畫面顯示的範例 AWS WAF 規則會將 CAPTCHA 動作套用至所有沒有`AssociateSoftwareToken`或`x-amzn-cognito-operation-name`標頭值的要求`VerifySoftwareToken`。



Action

The action to take when a web request matches the rule statement.

如需有關 AWS WAF 網路 ACL 和 Amazon Cognito 的詳細資訊，請參閱。建立 AWS WAF Web ACL 與使用者集區的關聯

將進階安全性新增到使用者集區

在您建立您的使用者集區之後，您可以在 Amazon Cognito 主控台導覽列上，存取 Advanced security (進階安全性)。您可以啟用這些使用者集區進階安全性功能，並自訂因應不同風險而執行的動作。或者，您可以使用稽核模式來收集偵測到風險的指標，而不需要使用任何安全性緩解措施。在稽核模式中，進階安全功能會將指標發佈到 Amazon CloudWatch。Amazon Cognito 產生第一個進階安全性事件後，您可以看到進階安全性指標。請參閱檢視進階安全指標。

進階安全性功能包括憑證洩漏偵測和自適應身分驗證。

憑證洩漏

使用者重複使用多個使用者帳戶的密碼。Amazon Cognito 的憑證洩漏功能，會編譯公開洩漏的使用者名稱和密碼，然後將您的使用者憑證與洩漏憑證清單進行比較。憑證洩漏偵測也會檢查容易猜測的密碼。

您可以選擇提示檢查洩漏憑證的使用者動作，以及您希望 Amazon Cognito 採取的動作。對於登入、註冊和密碼變更事件，Amazon Cognito 可以 封鎖登入 或 允許登入。這兩種情況下，Amazon Cognito 都會產生使用者活動日誌，您可以在其中找到有關事件的詳細資訊。

自適應身分驗證

Amazon Cognito 可以檢閱使用者登入請求中的位置和裝置資訊，並套用自動回應來保護使用者集區中的使用者帳戶不受可疑活動影響。

當您啟用進階安全性時，Amazon Cognito 會為使用者活動指派風險評分。您可以為可疑活動指派自動回應：需要 MFA、封鎖登入，或僅記錄活動詳細資料和風險評分。您也可以自動傳送電子郵件訊息，通知您的使用者發現可疑活動，讓他們可以重設密碼或採取其他自助式動作。

存取權杖自訂

啟用進階安全功能時，您可以將使用者集區設定為接受第 2 版 Lambda 觸發程序事件的回應。使用第 2 版，您可以在存取權杖中自訂範圍和其他宣告。這可提高您在使用者驗證時建立彈性授權結果的能力。如需詳細資訊，請參閱 自訂存取權杖。

主題

- 考量與限制

- 必要條件
- 設定進階安全性功能
- 檢查憑證是否洩漏
- 使用適應性身分驗證
- 檢視進階安全指標
- 啟用應用程式中的使用者集區進階安全性。

考量與限制

- Amazon Cognito 進階安全性功能有另外的定價。請參閱 Amazon Cognito 定價頁面。
- Amazon Cognito 透過下列標準身份驗證流程支援調適性身份驗證和惡意登入資料偵測：、和。USER_PASSWORD_AUTH ADMIN_USER_PASSWORD_AUTH USER_SRP_AUTH您不能將進階安全性與 CUSTOM_AUTH 流程和 自訂身分驗證挑戰 Lambda 觸發程序 搭配使用，或與聯合身分登入搭配使用。
- 藉由 完整功能 模式中的 Amazon Cognito 進階安全性功能，您可以建立 IP 地址 一律封鎖 和 一律允許 例外狀況。來自 Always block (一律封鎖) 例外狀況列表上的 IP 地址的工作階段，未透過自適應身分驗證指派風險層級，而且無法登入到您的使用者集區。
- 阻擋來自您使用者集區中的 Always block (一律封鎖) 例外狀況列表上的 IP 地址的請求，有助於您使用者集區的請求速率配額。Amazon Cognito 進階安全功能無法防止分散式阻斷服務 (DDoS) 攻擊。若要在使用者集區中實作容量攻擊防禦，請新增 AWS WAF Web ACL。如需詳細資訊，請參閱 建立 AWS WAF Web ACL 與使用者集區的關聯。
- 用戶端認證授與適用於 machine-to-machine (M2M) 授權，而不會連線至使用者帳戶。進階安全性功能只會監控使用者集區中的帳戶和密碼。若要在 M2M 活動中實作安全性功能，請考慮監控要求率和內容的 AWS WAF 功能。如需詳細資訊，請參閱 建立 AWS WAF Web ACL 與使用者集區的關聯。

必要條件

開始之前，您必須準備好以下事項：

- 搭配應用程式用戶端的使用者集區。如需詳細資訊，請參閱 使用者集區入門。
- 在 Amazon Cognito 主控台中，將多重要素驗證 (MFA) 設定為 Optional (選用)，以使用風險型的適應性身分驗證功能。如需詳細資訊，請參閱 將 MFA 新增到使用者集區。
- 如果是使用電子郵件通知，請移至 Amazon SES 主控台，設定和驗證可搭配電子郵件通知使用的電子郵件地址或網域。如需 Amazon SES 的詳細資訊，請參閱在 Amazon SES 中驗證身分。

設定進階安全性功能

您可以在 AWS Management Console中設定 Amazon Cognito 進階安全性功能。

設定使用者集區的進階安全性

1. 前往 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。
4. 選擇 App integration (應用程式整合) 索引標籤。找到 Advanced security (進階安全性)，然後選擇 Enable (啟用)。如果您先前啟用了進階安全性，則請選擇 Edit (編輯)。
5. 選取 Full function (完整功能)，以設定對憑證洩漏和適應性身分驗證的進階安全性回應。選取「僅稽核」以收集資訊並將使用者集區資料傳送至 CloudWatch。進階安全性定價適用於 Audit only (僅稽核) 和 Full function (完整功能) 模式。如需詳細資訊，請參閱 Amazon Cognito 定價。

   建議您在啟用動作之前，讓進階安全性功能保持處於稽核模式兩週。在此期間，Amazon Cognito 可學習您應用程式使用者的使用模式。
6. 如果您已選取 Audit only (僅稽核)，則請選擇 Save changes (儲存變更)。如果您已選取 Full function (完整功能)：

   a. 選取您是否要採取 Custom (自訂) 動作或使用 Cognito defaults (Cognito 預設值) 回應疑似 Compromised credentials (憑證洩漏)。Cognito defaults (Cognito 預設值) 是：

      i. 對 Sign-in (登入)、Sign-up (註冊) 和 Password change (密碼變更) 偵測憑證洩漏。

      ii. 使用動作 Block sign-in (封鎖登入) 回應憑證洩漏。

   b. 如果您已為 Compromised credentials (憑證洩漏) 選取 Custom (自訂) 動作，請選擇 Amazon Cognito 會用於 Event detection (事件偵測) 的使用者集區動作，以及您希望 Amazon Cognito 採取的 Compromised credentials responses (憑證洩漏回應)。對於疑似洩漏憑證，您可以 Block sign-in (封鎖登入) 或 Allow sign-in (允許登入)。

   c. 在 Adaptive authentication (適應性身分驗證) 下方選擇如何回應惡意登入嘗試。選取您是否要採取 Custom (自訂) 動作或使用 Cognito defaults (Cognito 預設值) 回應疑似惡意活動。當您選取 Cognito defaults (Cognito 預設值) 時，Amazon Cognito 會封鎖所有風險等級的登入，且不會通知使用者。

   d. 如果您已選取適用於 Adaptive authentication (適應性身分驗證) 的 Custom (自訂) 動作，則請選擇 Amazon Cognito 將根據嚴重性層級回應已偵測到之風險採取的 Automatic risk response (自動風險回應) 動作。針對風險層級指派回應時，您無法將較低限制的回應指派給較高層級的風險。您可以將下列回應指派給風險層級：

i. Allow sign-in (允許登入) – 不採取任何預防措施。

ii. Optional MFA (選用 MFA) – 如果使用者已設定 MFA，則 Amazon Cognito 會一律要求使用者在登入時提供額外的 SMS 或以時間為基礎的一次性密碼 (TOTP) 因素。如果使用者沒有設定 MFA，則他們可以繼續正常登入。

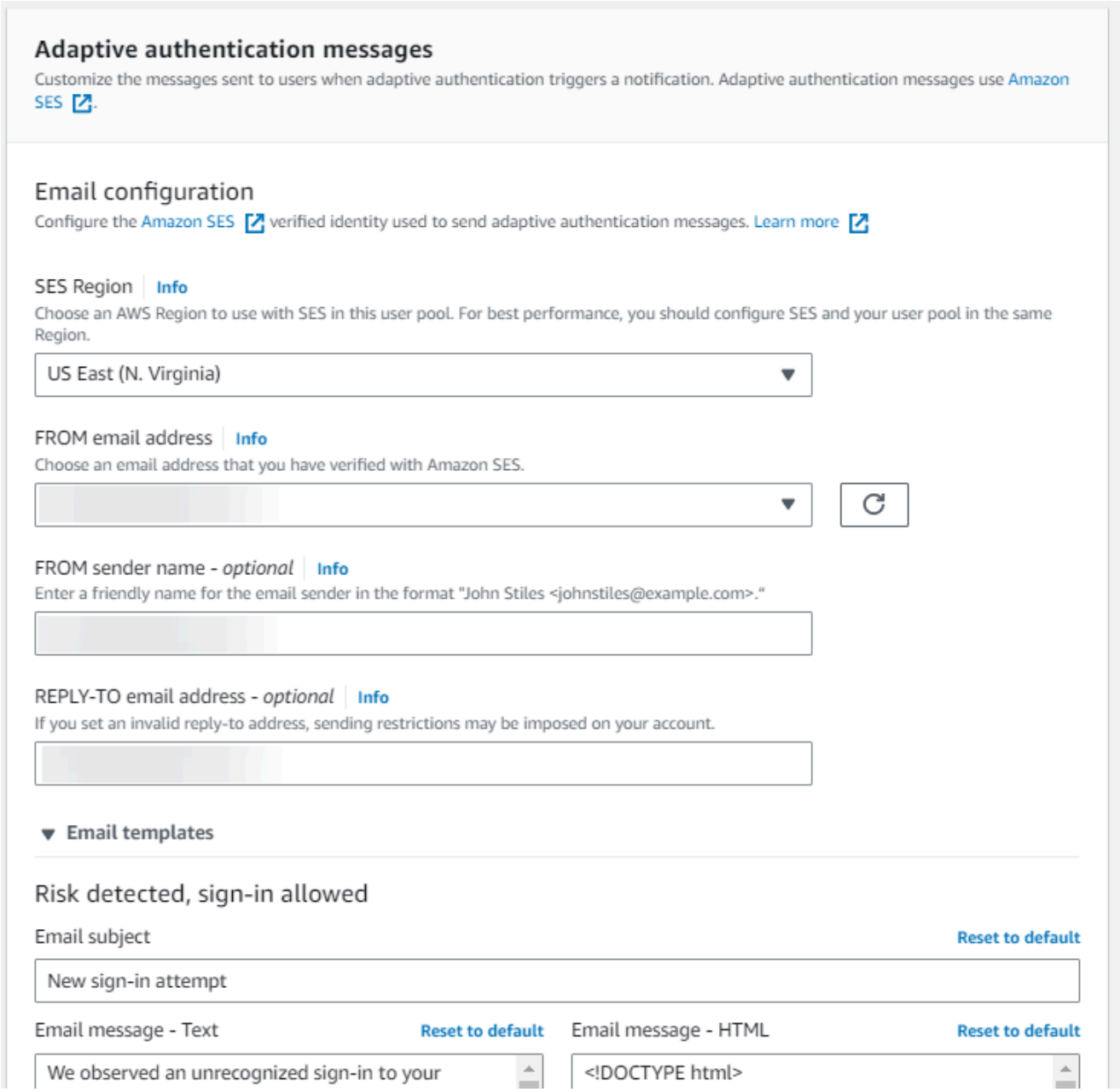
iii. Require MFA (需要 MFA) – 如果使用者已設定 MFA，則 Amazon Cognito 會一律要求使用者在登入時提供額外的 SMS 或 TOTP 因素。如果使用者沒有設定 MFA，則 Amazon Cognito 會提示他們設定 MFA。在自動為使用者要求 MFA 之前，請在應用程式中設定機制來擷取用於 SMS MFA 的電話號碼，或註冊用於 TOTP MFA 的驗證器應用程式。

iv. 封鎖登入 – 阻止使用者登入。

v. 通知使用者 – 傳送電子郵件訊息給使用者，其中包含 Amazon Cognito 已偵測到之風險以及您採取之回應的資訊。您可以為傳送的訊息自訂電子郵件訊息範本。

7. 如果您已在上一步中選擇 Notify user (通知使用者)，則可以自訂電子郵件傳遞設定和電子郵件訊息範本，以進行適應性身分驗證。

   a. 在 Email configuration (電子郵件組態) 下，選擇您想要與適應性身分驗證搭配使用的 SES Region (SES 區域)、FROM email address (寄件者電子郵件地址)、FROM sender name (寄件者傳送人名稱) 以及 REPLY-TO email address (回覆至電子郵件地址)。如需將使用者集區電子郵件訊息與 Amazon Simple Email Service 整合的相關詳細資訊，請參閱 Amazon Cognito 使用者集區的電子郵件設定。



b. 展開 Email templates (電子郵件範本)，自訂同時搭配 HTML 和純文字電子郵件訊息版本的適應性身分驗證通知。若要進一步了解電子郵件訊息範本，請參閱 訊息範本。

8. 展開 IP address exceptions (IP 地址例外狀況)，以建立無論進階安全風險評定為何，一律允許或封鎖的 Always-allow (一律允許) 或 Always-block (一律封鎖) IPv4 或 IPv6 地址範圍清單。以 CIDR notation (CIDR 表示法) 指定 IP 地址範圍 (如 192.168.100.0/24)。

9. 選擇 Save changes (儲存變更)。

檢查憑證是否洩漏

Amazon Cognito 可以偵測出使用者的使用者名稱和密碼是否已於其他位置洩漏。這種問題可能發生在使用者在多個網站上重複使用憑證，或使用者使用不安全的密碼。Amazon Cognito 會檢查在託管的使用者介面中並透過 Amazon Cognito API 以使用者名稱與密碼登入的本機使用者。本機使用者僅存在於您的使用者集區目錄中，不會透過外部 IdP 進行聯合。

從 Amazon Cognito 主控台其 App integration (應用程式整合) 標籤的 Advanced security (進階安全性) 中，您可以設定 Compromised credentials (憑證洩漏)。設定 Event detection (事件偵測) 以選擇您想要監控憑證洩漏的使用者事件。設定 Compromised credentials responses (憑證洩漏回應) 以選擇如果偵測到憑證洩漏時，要允許或封鎖該名使用者。Amazon Cognito 可以檢查登入、註冊和密碼變更時期間憑證是否洩漏。

選擇「允許登入」時，您可以查看 Amazon CloudWatch 日誌以監控 Amazon Cognito 對使用者事件進行的評估。如需詳細資訊，請參閱 檢視進階安全指標。當您選擇 Block sign-in (封鎖登入) 時，Amazon Cognito 可防止使用已洩漏的憑證登入的使用者。Amazon Cognito 封鎖使用者登入時，會將使用者的 `UserStatus` 設定為 `RESET_REQUIRED`。狀態為 `RESET_REQUIRED` 的使用者必須先變更其密碼，才能再次登入。

> **Note**
>
> 目前 Amazon Cognito 不會針對使用安全遠端密碼 (SRP) 流程的登入操作，檢查憑證是否洩漏。SRP 會在登入期間傳送密碼雜湊函數。Amazon Cognito 無法從內部存取密碼，因此只能評估用戶端以純文字形式傳遞的密碼。
> Amazon Cognito 會檢查使用 AdminInitiateAuthAPI 與流程搭配使用的登入資料，以及包含`ADMIN_USER_PASSWORD_AUTH`流程的 InitiateAuthAPI 是否存在遭到入侵的`USER_PASSWORD_AUTH`登入資料。

若要將憑證洩漏保護新增到使用者集區，請參閱 將進階安全性新增到使用者集區。

使用適應性身分驗證

透過調適式身分驗證，您可以將使用者集區設定成封鎖可疑的登入，或是新增第二個要素身分驗證，以便回應更高的風險等級。每次嘗試登入時，Amazon Cognito 都會針對有關登入請求是否可能來自洩漏來源的風險情況進行評分。此風險評分是根據包括裝置和使用者資訊在內的因素而定。當 Amazon Cognito 偵測到使用者工作階段中的風險，且使用者尚未選擇 MFA 方法時，調整型身分驗證可以為使用者集區中的使用者開啟或要求多重要素驗證 (MFA)。當您為使用者啟用 MFA 時，無論您如何設定調

整型身分驗證，他們都會在驗證期間接到提供或設定第二個要素的挑戰。從使用者的角度來看，您的應用程式提供協助他們設定 MFA 的功能，而 Amazon Cognito 則可選擇防止他們再次登入，直到設定其他要素為止。

Amazon Cognito 將登入嘗試、風險等級和失敗的挑戰發佈到 Amazon CloudWatch。如需詳細資訊，請參閱 檢視進階安全指標。

若要將適應性身分驗證新增到使用者集區，請參閱 將進階安全性新增到使用者集區。

主題

- 適應性身分驗證概觀
- 將使用者裝置與工作階段資料新增至 API 請求
- 檢視使用者事件歷史記錄
- 提供事件意見回饋
- 傳送通知訊息

適應性身分驗證概觀

在 Amazon Cognito 主控台 進階安全 頁面的 應用程式整合 索引標籤中，您可以選擇調整型身分驗證的設定，包括在不同風險層級要採取哪些動作，以及自訂要傳送給使用者的通知訊息。您可以為所有應用程式用戶端指派全域進階安全組態，但將用戶端層級組態套用至個別應用程式用戶端。

Amazon Cognito 適應性身分驗證會為每個使用者工作階段指派下列其中一個風險層級：高、中、低或無風險。

當您將 Enforcement method (強制執行方法) 從 Audit-only (僅供稽核使用) 變更為 Full-function (全功能) 時，請仔細考量您的選擇。您套用至風險等級的自動回應會影響 Amazon Cognito 指派給具有相同特性的後續使用者工作階段的風險等級。例如，在您選擇不採取任何動作或 Allow (允許) Amazon Cognito 最初評估為高風險的使用者工作階段後，Amazon Cognito 會將類似工作階段視為較低的風險。

針對每個風險層級，有下列選項可供您選擇：

選項	動作
允許	使用者無需額外的要素即可登入。
選用 MFA	已經設定第二個要素的使用者必須通過第二個挑戰，才能成功登入。簡訊的電話號碼和 TOTP

選項	動作
	軟體權杖是可用的第二個要素。沒有設定第二個要素的使用者只能使用一組憑證登入。
需要 MFA	已經設定第二個要素的使用者必須通過第二個挑戰，才能成功登入。Amazon Cognito 會阻擋未設定第二要素的使用者登入。
封鎖	Amazon Cognito 會阻擋指定風險層級的所有登入嘗試。

> **Note**
>
> 您無需驗證電話號碼，即可將其使用於文字簡訊，作為第二個身分驗證要素。

將使用者裝置與工作階段資料新增至 API 請求

當您使用 API 註冊使用者、登入使用者和重設其密碼時，您可以收集使用者工作階段的相關資訊並傳遞至 Amazon Cognito 進階安全性。此資訊包括使用者的 IP 地址和唯一裝置識別碼。

您的使用者與 Amazon Cognito 之間可能有一個中介網路裝置，如代理服務或應用程式伺服器。您可以收集使用者的內容資料，並將資料傳遞給 Amazon Cognito，讓調整性身分驗證根據使用者端點的特性來計算風險，而非根據您的伺服器或代理。如果您的用戶端應用程式直接呼叫 Amazon Cognito API 操作，則調整性身分驗證會自動記錄來源 IP 地址。但是，它不會記錄如 `user-agent` 等其他裝置資訊，除非您也收集裝置指紋。

使用 Amazon Cognito 內容資料收集程式庫產生此資料，然後使用和參數將其提交給 Amazon Cognito 進階安全性。ContextDataUserContextData內容資料收集程式庫包含在 AWS SDK 中。如需詳細資訊，請參閱將 Amazon Cognito 與 Web 和行動應用程式整合。如果您已在使用者集區中啟用進階安全性功能，便可提交 `ContextData`。如需詳細資訊，請參閱設定進階安全性功能。

當您從應用程式伺服器呼叫以下經過 Amazon Cognito 驗證的 API 操作時，請以 `ContextData` 參數傳遞使用者裝置的 IP。此外，傳遞您的伺服器名稱、伺服器路徑和編碼的裝置指紋資料。

- AdminInitiateAuth
- AdminRespondToAuthChallenge

當您呼叫 Amazon Cognito 未驗證的 API 操作時，您可以將 `UserContextData` 提交到 Amazon Cognito 進階安全性功能。此資料在 `EncodedData` 參數中包含裝置指紋。如果您滿足下列條件，則也可以在您的 `UserContextData` 中提交 `IpAddress` 參數：

- 您已在使用者集區中啟動進階安全性功能。如需詳細資訊，請參閱設定進階安全性功能。
- 您的應用程式用户端具有用戶端密碼。如需詳細資訊，請參閱設定使用者集區應用程式用戶端。
- 您已在應用程式用户端中啟動 Accept additional user context data (接受其他使用者內容資料)。如需詳細資訊，請參閱 接受其他使用者內容資料 (AWS Management Console)。

您的應用程式可在以下 Amazon Cognito 未驗證的 API 操作中，將已編碼裝置的指紋資料與使用者裝置的 IP 地址填入 `UserContextData` 參數。

- InitiateAuth
- RespondToAuthChallenge
- SignUp
- ConfirmSignUp
- ForgotPassword
- ConfirmForgotPassword
- ResendConfirmationCode

接受其他使用者內容資料 (AWS Management Console)

在您啟動 Accept additional user context data (接受其他使用者內容資料) 功能後，您的使用者集區會接受 `UserContextData` 參數中的 IP 地址。在下列情況下，您不需要啟動此功能：

- 您的用戶只能使用經過身份驗證的 API 操作（例如）登錄 AdminInitiateAuth ，並且您使用該`ContextData`參數。
- 您只希望未驗證的 API 操作將裝置指紋 (而不是 IP 地址) 傳送到 Amazon Cognito 進階安全性功能。

在 Amazon Cognito 主控台中如下所示更新您的應用程式用戶端，以新增對其他使用者內容資料的支援。

1. 登入 Amazon Cognito 主控台。
2. 在導覽窗格中選擇 Manage your User Pools (管理您的使用者集區)，然後選擇您要編輯的使用者集區。

3. 選擇 App integration (應用程式整合) 標籤。
4. 在 App clients and analytics (應用程式用户端與分析) 下方，選擇或建立應用程式用户端。如需詳細資訊，請參閱設定使用者集區應用程式用戶端。
5. 從 App client information (應用程式用戶端資訊) 容器中選擇 Edit (編輯)。
6. 在您應用程式用戶端的 Advanced authentication settings (進階驗證設定) 中，選擇 Accept additional user context data (接受其他使用者內容資料)。
7. 選擇儲存變更。

若要將您的應用程式用戶端設定為接受 Amazon Cognito API 中的使用者內容資料，請在CreateUserPoolClient或UpdateUserPoolClient請求`true`中設定`EnablePropagateAdditionalUserContextData`為。如需如何從 Web 或行動應用程式啟用進階安全性的詳細資訊，請參閱啟用應用程式中使用者集區進階安全性。當您的應用程式從伺服器呼叫 Amazon Cognito 時，將從用戶端收集使用者內容資料。以下是使用 JavaScript SDK 方法的範例`getData`。

```
var encodedData =
 AmazonCognitoAdvancedSecurityData.getData(username, userPoolId, clientId);
```

當您將應用程式設計為使用適應性身分驗證時，我們建議您將最新的 Amazon Cognito SDK 整合到您的應用程式中。SDK 的最新版本將收集裝置指紋資訊，例如裝置 ID、型號和時區。如需有關 Amazon Cognito SDK 的詳細資訊，請參閱安裝使用者集區 SDK。Amazon Cognito 進階安全性僅儲存風險分數，並指派給您應用程式以正確格式提交的事件。如果 Amazon Cognito 傳回錯誤回應，請檢查確認您的請求包含有效的私密雜湊，且 `IPaddress` 參數是有效的 IPv4 或 IPv6 地址。

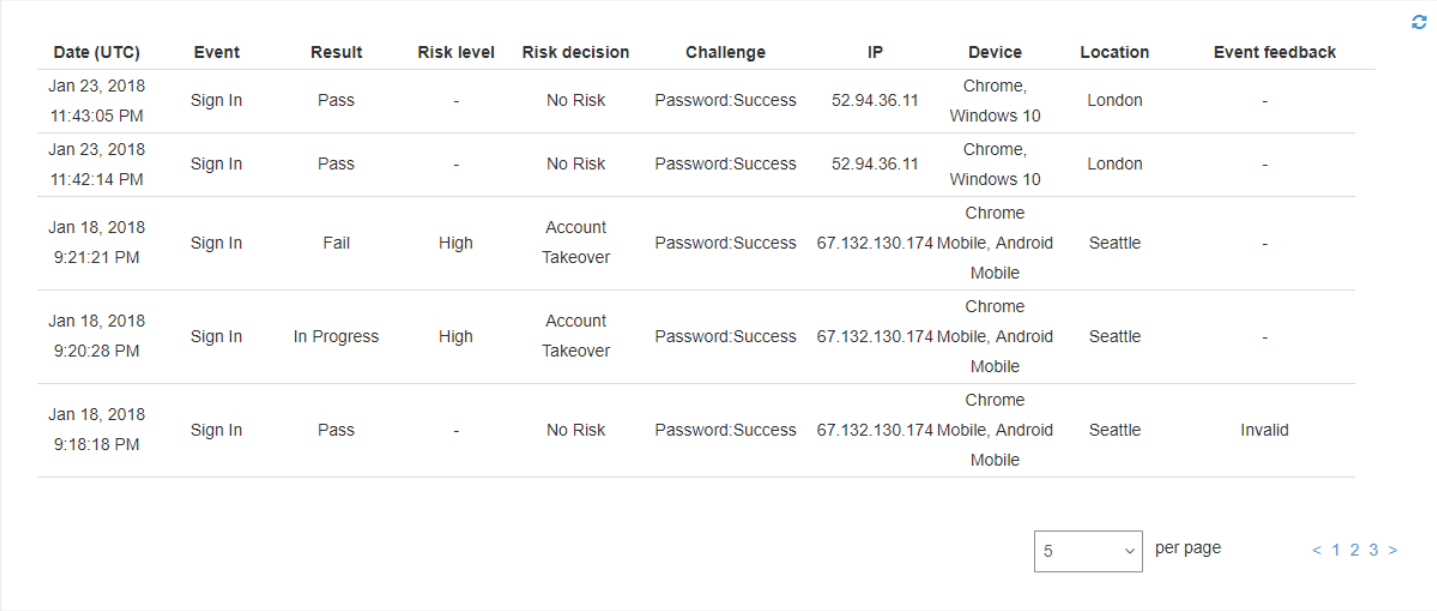
`ContextData` 和 `UserContextData` 資源

- AWS Amplify SDK for Android：GetUserContextData
- AWS Amplify SDK for iOS：userContextData
- JavaScript：amazon-cognito-advanced-security- 數據分鐘

檢視使用者事件歷史記錄

> **Note**
>
> 在新的 Amazon Cognito 主控台中，您可以在 Users (使用者) 標籤中檢視使用者事件歷史記錄。

若要查看該使用者的登入歷史記錄，您可以在 Amazon Cognito 主控台中的 Users (使用者) 索引標籤中選擇使用者。使用者事件歷史記錄將由 Amazon Cognito 保留 2 年。



Date (UTC)	Event	Result	Risk level	Risk decision	Challenge	IP	Device	Location	Event feedback
Jan 23, 2018 11:43:05 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10	London	-
Jan 23, 2018 11:42:14 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10	London	-
Jan 18, 2018 9:21:21 PM	Sign In	Fail	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	-
Jan 18, 2018 9:20:28 PM	Sign In	In Progress	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	-
Jan 18, 2018 9:18:18 PM	Sign In	Pass	-	No Risk	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	Invalid

5 per page < 1 2 3 >

每個登入事件都有一個事件 ID。事件還具有對應的內容資料，例如位置、裝置詳細資訊和風險偵測結果。您可以使用 Amazon Cognito API 作業AdminListUserAuthEvents或使用 AWS Command Line Interface (AWS CLI) 與事件查詢使用者事件歷史記錄。admin-list-user-auth

您還可以在事件 ID 與 Amazon Cognito 記錄事件時發放的字符之間建立關聯。ID 和存取權杖在其承載中包含此事件 ID。Amazon Cognito 還會在重新整理權杖使用與原始事件 ID 之間建立關聯。您可以將原始事件 ID 回溯至導致發放 Amazon Cognito 權杖之登入事件的事件 ID。您可以追溯系統中的字符使用記錄，以找出特定的身分驗證事件。如需詳細資訊，請參閱 將權杖用於使用者集區。

提供事件意見回饋

事件意見回饋會影響即時的風險評估，並可隨著時間改善風險評估演算法。您可以透過 Amazon Cognito 主控台和 API，在登入嘗試的有效期間內提供意見回饋。

> **Note**
>
> 您的事件意見回饋會影響 Amazon Cognito 指派給具有相同特性之後續使用者工作階段的風險等級。

在 Amazon Cognito 主控台中，從 Users (使用者) 索引標籤選擇使用者，然後選取 Provide event feedback (提供事件意見回饋)。您可以查看事件詳細資訊，並 Set as valid (設定為有效) 或 Set as invalid (設定為無效)。

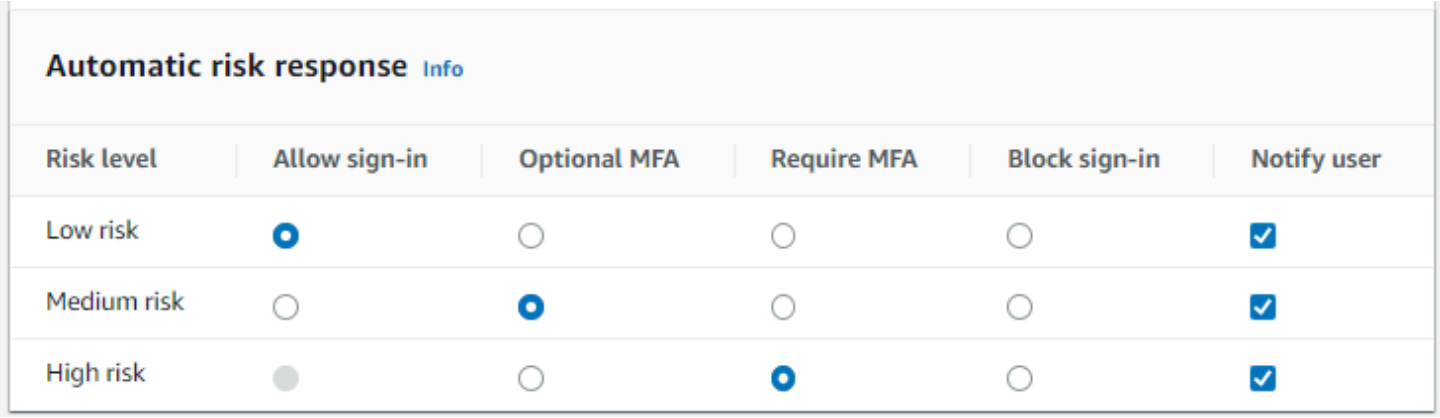
主控台會在 Users and groups (使用者和群組) 索引標籤列出登入歷史記錄。如果您選取一個項目，可以將事件標記為有效或無效。您還可以通過用戶池 API 操作提供反饋 AdminUpdateAuthEventFeedback，並通過命 AWS CLI 令 admin-update-auth-event-feedback。

當您在 Amazon Cognito 主控台中選取 Set as valid (設定為有效) 或在 API 中為 `FeedbackValue` 提供 `valid` 值時，這會告訴 Amazon Cognito 您信任 Amazon Cognito 已評估具一定風險程度的使用者工作階段。當您在 Amazon Cognito 主控台中選取 Set as invalid (設定為無效) 或在 API 中提供 `invalid` 的 `FeedbackValue` 值時，您會告訴 Amazon Cognito 您不信任用者工作階段，或者您認為 Amazon Cognito 評估的風險程度不夠高。

傳送通知訊息

藉由進階安全保護功能，Amazon Cognito 可以通知您的使用者有風險的登入嘗試。Amazon Cognito 還可以提示使用者選取連結以指示登入是有效或無效。Amazon Cognito 使用此意見回饋來提高使用者集區的風險偵測準確性。

於 Automatic risk response (自動風險回應) 區段中，選擇低度、中度和高度風險案例適用的 Notify Users (通知使用者)。



Amazon Cognito 會傳送電子郵件通知給您的使用者，無論他們是否已驗證其電子郵件地址。

您可以自訂通知電子郵件訊息，並提供純文字和 HTML 兩種版本的訊息。若要自訂電子郵件通知，請在進階安全組態中，從 Adaptive authentication messages (調整型身分驗證訊息) 開啟 Email templates (電子郵件範本)。若要進一步了解電子郵件範本，請參閱 訊息範本。

檢視進階安全指標

Amazon Cognito 會將進階安全功能的指標發佈到您在 Amazon CloudWatch 的帳戶中。Amazon Cognito 會依據風險層級，也會依據請求層級，將進階安全指標進行分組。

若要在 CloudWatch 主控台中檢視指標

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 在導覽窗格中，選擇 Metrics (指標)。
3. 選擇 Amazon Cognito。
4. 選擇一組彙總指標，例如 By Risk Classification (依風險分類)。
5. All metrics (所有指標) 標籤會顯示該選項的所有指標。您可以執行下列作業：
   - 若要將資料表排序，請使用直欄標題。
   - 若要將指標圖形化，請勾選指標旁的核取方塊。若要選擇所有指標，請勾選表格標題列中的核取方塊。
   - 若要依資源篩選，請選擇資源 ID，然後選擇 Add to search (新增至搜尋)。
   - 若要依指標篩選，請選擇指標名稱，然後選擇 Add to search (新增至搜尋)。

指標	描述	指標維度
CompromisedCredentialRisk	Amazon Cognito 偵測到憑證洩露的請求。	Operation：操作類型。PasswordChange 、SignIn 或 SignUp 是唯一的維度。 UserPoolId：使用者集區的識別碼。 RiskLevel：高 (預設)、中或低。
AccountTakeoverRisk	Amazon Cognito 偵測到帳戶接管風險的請求。	Operation：操作類型。PasswordChange 、SignIn 或 SignUp 是唯一的維度。 UserPoolId：使用者集區的識別碼。 RiskLevel：高、中或低。

指標	描述	指標維度
OverrideBlock	因為開發人員提供的組態而遭到 Amazon Cognito 封鎖的請求。	Operation：操作類型。PasswordChange 、SignIn 或 SignUp 是唯一的維度。 UserPoolId：使用者集區的識別碼。 RiskLevel：高、中或低。
Risk	Amazon Cognito 標記為有風險的請求。	Operation：操作的類型，例如 PasswordChange 、SignIn 或 SignUp。 UserPoolId：使用者集區的識別碼。
NoRisk	Amazon Cognito 未識別出任何風險的請求。	Operation：操作的類型，例如 PasswordChange 、SignIn 或 SignUp。 UserPoolId：使用者集區的識別碼。

Amazon Cognito 為您提供兩個預先定義的指標群組，供您在中 CloudWatch進行就緒分析。By Risk Classification (依風險分類) 針對 Amazon Cognito 識別為有風險的請求，識別風險層級的精細度。By Request Classification (依請求分類) 反映依請求層級彙整的指標。

彙總指標群組	描述
依風險分類	Amazon Cognito 識別為有風險的請求。
依請求分類	依請求彙總的指標。

啟用應用程式中的使用者集區進階安全性。

在您設定好使用者集區的進階安全性功能之後，您必須在 Web 或行動應用程式中啟用這些功能。

使用進階安全性 JavaScript

1. 將適用於的 Amazon Cognito 身分開發套件新增 JavaScript至您的應用程式。
2. 在 CognitoUserPool.js 中，設定`AdvancedSecurityDataCollectionFlag`為`true`。將 `UserPoolId` 設定為您的使用者集區 ID。
3. 將此源引用添加到您的應用程序的 JavaScript 文件中。取`<region>`代為下列清單 AWS 區域 中的：`us-east-1`、`us-east-2`、`us-west-2`、`eu-west-1`、`eu-west-2`、或`eu-central-1`。

```
<script src="https://amazon-cognito-assets.<region>.amazoncognito.com/amazon-
cognito-advanced-security-data.min.js"></script>
```

搭配 Android 使用進階安全性

1. 使用安卓系統建立您 AWS Amplify 的應用程式。如需詳細資訊，請參閱《AWS Amplify 開發人員中心》中的專案設定。
2. 使用 `userContextDataProvider`，以在驗證請求中包含使用者和裝置資訊。

如需舊版 Android SDK 中新增使用者背景資料的詳細資訊，請參閱 aws-android-sdk-cognitoidentityprovider-asf。

搭配 iOS 使用進階安全性

1. 使 AWS Amplify 用斯威夫特或撲創建您的應用程序。如需詳細資訊，請參閱《AWS Amplify 開發人員中心》的 Swift 專案設定和 Flutter 專案設定。
2. 在驗證請求中包含使用者和裝置資訊。有關InitiateAuthInput與 InitiateAuthAPI 操作一起使用的示例，請`userContextData`參閱上的。 GitHub

如需舊版 iOS SDK 中新增使用者背景資料的詳細資訊，請參閱AWSCognitoIdentityProviderASF。

建立 AWS WAF Web ACL 與使用者集區的關聯

AWS WAF 是一個 Web 應用程序防火牆。使用 AWS WAF Web 存取控制清單 (Web ACL)，您可以保護使用者集區不受對託管 UI 和 Amazon Cognito API 服務端點發出不必要的請求。Web ACL 可讓您

對使用者集區回應的所有 HTTPS Web 要求進行精細控制。如需有關 AWS WAF Web ACL 的詳細資訊，請參閱AWS WAF 開發人員指南中的管理和使用 Web 存取控制清單 (Web ACL)。

當您的 AWS WAF Web ACL 與使用者集區相關聯時，Amazon Cognito 會將選取的非機密標頭和請求內容從您的使用者轉寄到。 AWS WAF AWS WAF 檢查請求的內容，將其與您在 Web ACL 中指定的規則進行比較，然後將回應傳回給 Amazon Cognito。

關於 AWS WAF 網絡 ACL 和 Amazon Cognito 的知識

- 被封鎖的要求 AWS WAF 不會計入任何要求類型的要求率配額中。會在 API 層級節流 AWS WAF 處理常式之前呼叫處理常式。
- 當您建立 Web ACL 時，在 Web ACL 完全傳播並可供 Amazon Cognito 使用之前會經歷一小段時間。傳輸時間可以是幾秒鐘到分鐘數。 AWS WAF `WAFUnavailableEntityException`當您嘗試在 Web ACL 完全傳播之前建立關聯時，會傳回 a。
- 您可建立 Web ACL 與使用者集區的關聯。
- 您的請求可能會導致負載大於 AWS WAF 可以檢查的限制。請參閱AWS WAF 開發人員指南中的超大請求元件處理，了解如何設定如何 AWS WAF 處理 Amazon Cognito 的過大請求。
- 您無法將使用 AWS WAF 詐騙控制帳戶接管預防 (ATP) 的網路 ACL 與 Amazon Cognito 使用者集區建立關聯。您可於新增 `AWS-AWSManagedRulesATPRuleSet` 受管規則群組時實作 ATP 功能。將其與使用者集區關聯之前，請確定您的 Web ACL 並未使用此受管規則群組。
- 當您擁有與使用者集區相關聯的 AWS WAF Web ACL，且 Web ACL 中的規則顯示驗證碼時，這可能會導致託管 UI TOTP 註冊中無法復原的錯誤。若要建立具有 CAPTCHA 動作且不會影響託管 UI TOTP 的規則，請參閱 為託管的使用者介面 TOTP MFA 設定您的 AWS WAF 網頁 ACL。

AWS WAF 檢查對下列端點的要求。

託管 UI

對 使用者集區聯合端點和託管 UI 參考 中所有端點的請求。

公有 API 操作

從您的應用程式向不使用 AWS 登入資料進行授權的 Amazon Cognito API 的請求。這包括 API 操作 InitiateAuth，例如 RespondToAuthChallenge，和GetUser。範圍內的 API 操作 AWS WAF 不需要使用憑據進行身份驗 AWS 證。其未經身分驗證，或使用工作階段字串或存取字符進行授權。如需詳細資訊，請參閱 Amazon Cognito 使用者集區經身分驗證和未進行身分驗證的 API 操作。

您可使用 Count (計數)、Allow (允許)、Block (封鎖) 或顯示 CAPTCHA (驗證碼) 以回應與規則相符之請求的規則作業，來設定 Web ACL 中的規則。如需詳細資訊，請參閱《AWS WAF 開發人員指南》中的 AWS WAF 規則。依據規則動作，您可自訂 Amazon Cognito 傳回給使用者的回應。

> **Important**
>
> 自訂錯誤回應的選項取決於您提出 API 請求的方式。
>
> - 您可自訂託管 UI 請求的錯誤碼和回應主體。您僅可於託管 UI 中為您的使用者提供驗證碼以解決問題。
> - 對於您使用 Amazon Cognito 使用者集區 API 提出的請求，您可自訂接收 封鎖 回應請求的回應主體。您還可於 400–499 範圍內指定自訂錯誤代碼。
> - AWS Command Line Interface (AWS CLI) 和 AWS SDK 會針對產生封鎖或驗證碼回應的要求傳回`ForbiddenException`錯誤。

將 Web ACL 與您的使用者集區建立關聯

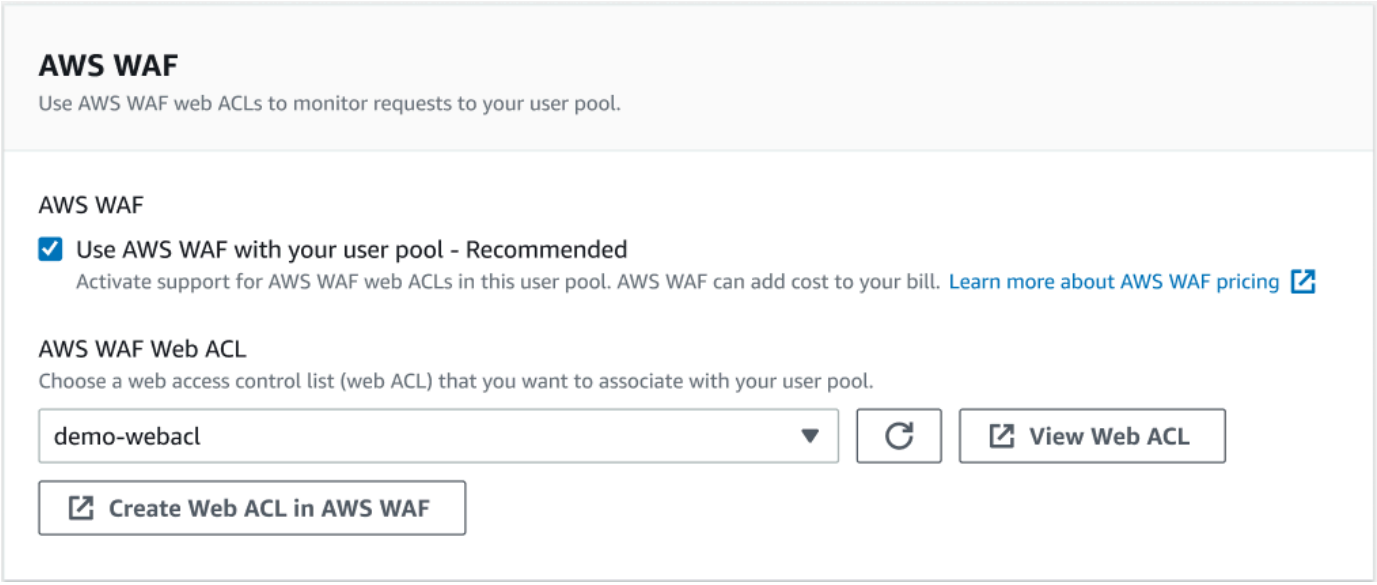
若要在使用者集區中使用 Web ACL，您的 AWS Identity and Access Management (IAM) 主體必須具有下列 Amazon Cognito 許可。如需有關 AWS WAF 權限的詳細資訊，請參閱AWS WAF 開發人員指南中的 AWS WAF API 權限。

- `cognito-idp:AssociateWebACL`
- `cognito-idp:DisassociateWebACL`
- `cognito-idp:GetWebACLForResource`
- `cognito-idp:ListResourcesForWebACL`

雖然您必須授與 IAM 許可，但列出的操作僅限許可，並且不對應於 API 操作。

AWS WAF 為您的使用者集區啟用並關聯 Web ACL 的步驟

1. 登入 Amazon Cognito 主控台。
2. 在導覽窗格中，選擇 User Pools (使用者集區)，然後選擇您要編輯的使用者集區。
3. 選擇 User pool properties (使用者集區屬性) 標籤。
4. 選擇 AWS WAF 旁的 Edit (編輯)。
5. 在下 AWS WAF，選取 AWS WAF 與您的使用者集區搭配使用。



6. 選擇您已經建立的 AWS WAF Web ACL，或選擇在中建立 Web ACL，在中 AWS WAF的新 AWS WAF 工作階段中建立一個 ACL AWS Management Console。
7. 選擇儲存變更。

若要以程式設計方式將 Web ACL 與 AWS Command Line Interface 或 SDK 中的使用者集區建立關聯，請使用 AWS WAF API 中的 AssociateWebACL。Amazon Cognito 沒有關聯 Web ACL 的單獨 API 操作。

測試和記錄 AWS WAF 網頁 ACL

當您在 Web ACL 中將規則動作設定為「計數」時，會將要求 AWS WAF 新增至符合規則的要求計數。如要使用您的使用者集區測試 Web ACL，請將規則動作設定為 Count (計數)，並考慮與每個規則相符的請求量。例如，若您要設定為 Block (封鎖) 動作的規則與您確定為正常使用者流量的大量請求相符合，您可能需要重新設定您的規則。如需詳細資訊，請參閱 AWS WAF 開發人員指南中的測試和調整您的 AWS WAF 保護。

您也可以設定 AWS WAF 將請求標頭記錄到 Amazon CloudWatch 日誌日誌群組、亞馬遜簡單儲存服務 (Amazon S3) 儲存貯體或 Amazon 資料 Firehose。您可透過 `x-amzn-cognito-client-id` 和 `x-amzn-cognito-operation-name` 識別您使用使用者集區 API 提出的 Amazon Cognito 請求。託管的 UI 請求僅包括 `x-amzn-cognito-client-id` 標題。如需詳細資訊，請參閱《AWS WAF 開發人員指南》中的記錄 Web ACL 流量。

AWS WAF 網路 ACL 不受 Amazon Cognito 進階安全功能的定價約束。 AWS WAF 補充 Amazon Cognito 進階安全功能的安全功能的安全功能。您可以在使用者集區中啟用這兩個功能。 AWS WAF 單獨用於檢查用戶池請求的帳單。如需詳細資訊，請參閱 AWS WAF 定價。

記錄 AWS WAF 要求資料需由您指定記錄的服務收取額外費用。如需詳細資訊，請參閱《AWS WAF 開發人員指南》中的記錄 Web ACL 流量資訊的定價。

使用者集區大小寫區分

依預設，您在中建立的 Amazon Cognito 使用者集區不區分大小寫。 AWS Management Console 當使用者集區不區分大小寫時，user@example.com 和 User@example.com 是指同一名使用者。當使用者集區的使用者名稱不區分大小寫時，`preferred_username` 和 `email` 屬性也不會區分大小寫。

若要將使用者集區的大小寫區分設定納入考慮，應根據替代使用者屬性，在應用程式的程式碼中識別使用者。由於是使用者名稱、偏好的使用者名稱或電子郵件地址屬性在不同的使用者描述檔中可能會有所不同，因此請改為參閱 `sub` 屬性。您也可以在使用者集區中建立不可變的自訂屬性，並在每個新的使用者描述檔中，將自己的唯一識別符值指派給屬性。首次建立使用者時，您可以將值寫入您建立的不可變自訂屬性中。

> **Note**
>
> 無論使用者集區的區分大小寫設定如何，Amazon Cognito 都會要求來自 SAML 或 OIDC 身分提供者 (IdP) 的聯合身分使用者傳遞唯一且區分大小寫的 `NameId` 或 `sub` 宣告。如需唯一識別碼區分大小寫和 SAML 的詳細資訊 IdPs，請參閱使用 SP 啟動式 SAML 登入。

建立區分大小寫的使用者集區

如果您使用 AWS Command Line Interface (AWS CLI) 和 API 作業 (例如) 建立資源 CreateUserPool，則必須將布林`CaseSensitive`參數設定為`false`。此設定會建立不區分大小寫的使用者集區。如果您不指定值，`CaseSensitive` 會預設為 `true`。此預設與您在 AWS Management Console中建立之使用者集區的預設行為相反。在 2020 年 2 月 12 日之前，無論平台為何，使用者集區皆預設為區分大小寫。

您可以使用 AWS Management Console 或 DescribeUserPoolAPI 作業的 [登入體驗] 索引標籤，檢閱帳戶中每個使用者集區的區分大小寫設定。

遷移至新的使用者集區

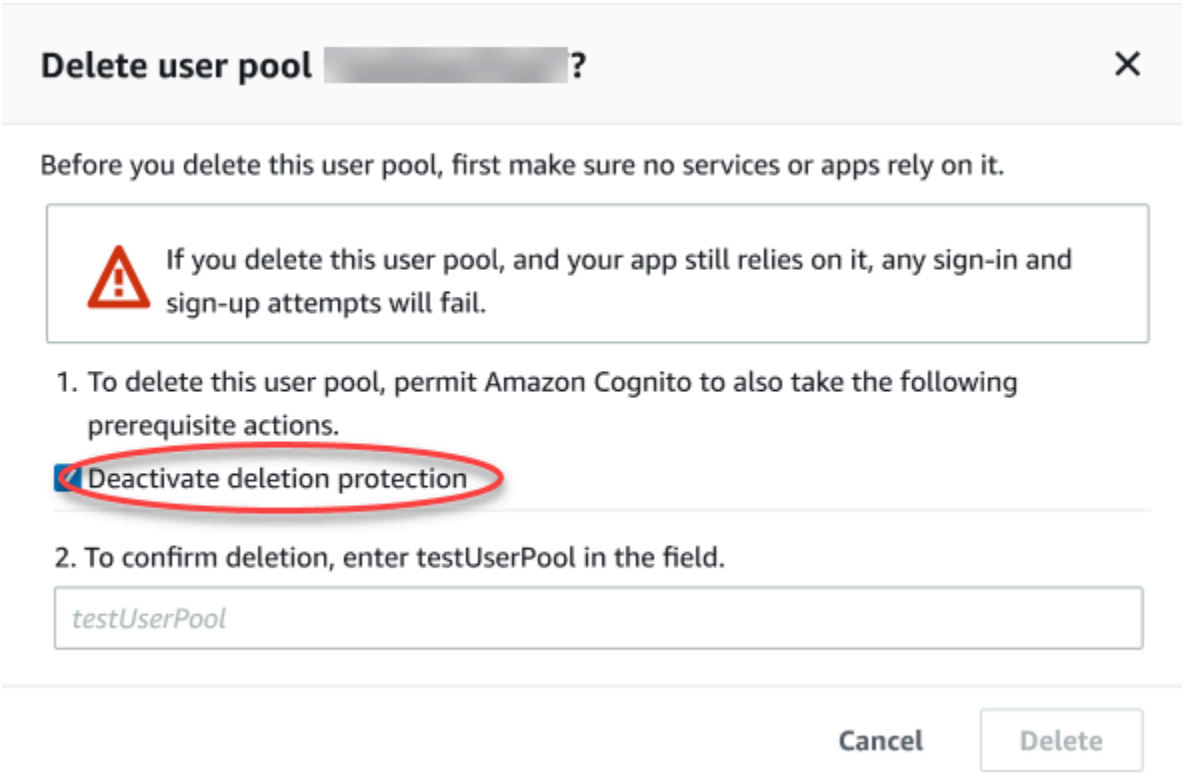
由於使用者描述檔之間的潛在衝突，您無法將 Amazon Cognito 使用者集區從區分大小寫變成不區分大小寫。替代方法為將您的使用者遷移到新的使用者集區。您必須建置遷移代碼以解決與大小寫相關的衝突。此代碼必須傳回唯一的新使用者，或者在偵測到衝突時拒絕登入嘗試。在新的不區分大小寫的使用者集區中，指派一個 遷移使用者 Lambda 觸發程序。該 AWS Lambda 功能可

以在新的不區分大小寫的用戶池中創建用戶。當使用者無法成功登入不區分大小寫的使用者集區時，Lambda 函數會從區分大小寫的使用者集區中查找並複製使用者。您也可以在事件上啟動遷移使用者 Lambda ForgotPassword觸發器。Amazon Cognito 會將登入或密碼恢復動作中的使用者資訊和事件中繼資料傳遞至您的 Lambda 函數。當函數在不區分大小寫的使用者集區中建立新使用者時，您可以使用事件資料來管理使用者名稱與電子郵件地址之間的衝突。這些衝突發生於使用者名稱和電子郵件地址在不區分大小寫的使用者集區中是唯一的，但在區分大小寫的使用者集區中是相同的。

如需有關如何在 Amazon Cognito 使用者集區之間使用遷移使用者 Lambda 觸發器的詳細資訊，請參閱部落格中的將使用者遷移至 Amazon Cognito 使用者集區。 AWS

使用者集區刪除保護

若要讓系統管理員不會意外刪除您的使用者集區，請啟用刪除保護。在啟用刪除保護的情況下，您必須先確認是否要刪除使用者集區，然後才能刪除使用者集區。當您刪除 AWS Management Console 中的使用者集區時，您可以同時停用刪除保護。當您接受停用刪除保護的提示並確認想要刪除 (如下圖所示) 時，Amazon Cognito 會刪除您的使用者集區。



當您想要刪除具有 Amazon Cognito API 請求的使用者集區時，必須先在 UpdateUserPool 請求中將 `DeletionProtection` 變更為 `Inactive`。如果您沒有停用刪除保護，Amazon Cognito 會傳回

InvalidParameterException 錯誤訊息。停用刪除保護後，您可以在 DeleteUserPool 請求中刪除使用者集區。

在 AWS Management Console 建立新使用者集區時，Amazon Cognito 預設會啟用 Deletion protection (刪除保護)。當您使用CreateUserPool API 建立使用者集區，刪除保護預設處於停用狀態。若要在您使用 AWS CLI 或 AWS 開發套件建立的使用者集區中使用此功能，請將 DeletionProtection 參數設為 True。

您可以在 Amazon Cognito 主控台 User pool setting (使用者集區設定) 標籤的 Deletion protection (刪除保護) 容器中啟用或停用刪除保護狀態。

設定刪除保護

1. 前往 Amazon Cognito 主控台。您可能會收到提示，要求您輸入 AWS 憑證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。
4. 選擇 User pool settings (使用者集區設定) 標籤。找到 Deletion Protection (刪除保護)，然後選擇 Activate (啟動) 或 Deactivate (停用)。
5. 在下一個對話框中確認您的選擇。

管理使用者存在錯誤回應

Amazon Cognito 支援自訂使用者集區傳回的錯誤回應。自訂錯誤回應可用於使用者建立和身分驗證、密碼復原和確認操作。

使用使用者集區應用程式用戶端的 PreventUserExistenceErrors 設定，啟用或停用已有使用者相關錯誤。當您使用 Amazon Cognito 使用者集區 API 建立新的應用程式用戶端時LEGACY，預設為或停用。PreventUserExistenceErrors在 Amazon Cognito 主控台中，預設會選取「防止使用者存在錯誤」選項 PreventUserExistenceErrors — 設定ENABLED為 —。若要更新您的PreventUserExistenceErrors組態，請執行下列其中一個動作：

- 變更 UpdateUserPoolClientAPI 要求PreventUserExistenceErrors之間ENABLED和LEGACY之間的值。
- 在 Amazon Cognito 主控台中編輯您的應用程式用戶端，並變更選取 () 和取消選取 (ENABLED) 之間防止使用者存在錯誤的狀態。LEGACY

當此屬性的值為時LEGACY，當用戶嘗試使用用戶池中不存在的用戶名登錄時，您的應用程序客戶端將返回UserNotFoundException錯誤響應。

當此屬性的值為時ENABLED，您的應用程序客戶端不會洩露用戶池中用戶帳戶的不存在，並顯示UserNotFoundException錯誤。的PreventUserExistenceErrors組態ENABLED具有下列影響：

- Amazon Cognito 會回應 API 請求的非特定資訊，否則其回應可能會揭露有效使用者存在。
- Amazon Cognito 登入和忘記密碼 API 會傳回一般身份驗證失敗回應。錯誤回應會告訴您使用者名稱或密碼不正確。
- Amazon Cognito 帳戶確認和密碼復原 API 會傳回回應，指出代碼已傳送至模擬交付媒體，而不是部分表示使用者聯絡資訊。

下列資訊詳細說明使用者集區作業設定PreventUserExistenceErrors為時的行為ENABLED。

身份驗證和用戶創建操作

您可以在使用者名稱密碼和安全遠端密碼 (SR P) 驗證中設定錯誤回應。您也可以使用自訂身分驗證自訂傳回的錯誤。下列 API 會執行這些驗證作業：

- AdminInitiateAuth
- AdminRespondToAuthChallenge
- InitiateAuth
- RespondToAuthChallenge

下列清單示範如何自訂使用者身分驗證操作中的錯誤回應。

使用者名稱和密碼身分驗證

若要使用 ADMIN_USER_PASSWORD_AUTH 和 USER_PASSWORD_AUTH 登入使用者，請在 AdminInitiateAuth 或 InitiateAuth API 請求中包含使用者名稱和密碼。使用者名稱或密碼不正確時，Amazon Cognito 傳回一般 NotAuthorizedException 錯誤。

安全遠端密碼 (SRP) 型身分驗證

若要使用 USER_SRP_AUTH 登入使用者，請在 AdminInitiateAuth 或 InitiateAuth API 請求中包含使用者名稱和 SRP_A 參數。作為回應，Amazon Cognito 返回SRP_B和鹽為用戶。找不到使用者時，Amazon Cognito 會傳回第一個步驟中的模擬回應，如 RFC 5054 所述。Amazon

Cognito 會傳回相同的 'salt'，並以全域唯一識別符 (UUID) 格式傳回相同使用者名稱和使用者集區組合的內部使用者 ID。傳送附密碼證明的 RespondToAuthChallenge API 請求時，若使用者名稱或密碼不正確，Amazon Cognito 會傳回一般 NotAuthorizedException 錯誤。

> **Note**
>
> 如果您使用驗證型別名屬性，且不可變使用者名稱格式不是 UUID，可以用使用者名稱和密碼身分驗證模擬一般回應。

自訂身分驗證挑戰 Lambda 觸發器

如果您使用自訂身分驗證挑戰 Lambda 觸發器並啟用錯誤回應，LambdaChallenge 就會傳回 UserNotFound 布林參數。接著會以 DefineAuthChallenge、VerifyAuthChallenge 及 CreateAuthChallenge Lambda 觸發器請求傳遞。您可以使用此觸發器來模擬不存在使用者的自訂授權挑戰。如果您為不存在的使用者呼叫預先進行身分驗證的 Lambda 觸發器，Amazon Cognito 就會傳回 UserNotFound。

下列清單示範如何自訂使用者建立操作中的錯誤回應。

SignUp

UsernameExistsException當一個用戶名已經採取SignUp操作總是返回。如果您不希望 Amazon Cognito 在應用程式中註冊使用者時，傳回電子郵件地址和電話號碼的 UsernameExistsException 錯誤訊息，請使用驗證型別名屬性。如需關於別名的詳細資訊，請參閱自訂登入屬性。

如需 Amazon Cognito 如何防止使用 SignUp API 請求來探索使用者集區中使用者的範例，請參閱防止註冊時出現電子郵件地址和電話號碼 UsernameExistsException 錯誤。

匯入的使用者

如果啟用 PreventUserExistenceErrors，在驗證匯入的使用者期間，系統會傳回一般 NotAuthorizedException 錯誤，指出使用者名稱或密碼不正確，而不是傳回 PasswordResetRequiredException。如需詳細資訊，請參閱需要匯入的使用者重設密碼。

遷移使用者 Lambda 觸發器

Amazon Cognito 會在 Lambda 觸發器於原始活動內容中設定空白回應時，傳回不存在使用者的模擬回應。如需詳細資訊，請參閱遷移使用者 Lambda 觸發器。

防止註冊時出現電子郵件地址和電話號碼 **UsernameExistsException** 錯誤

下列範例示範如何在使用者集區中設定別名屬性時，避免重複的電子郵件地址和電話號碼產生 UsernameExistsException 錯誤以回應 SignUp API 請求。您必須建立使用者集區，並將電子郵件地址或電話號碼作為別名屬性。如需詳細資訊，請參閱使用者集區屬性的自訂登入屬性一節。

1. Jie 註冊一個新的使用者名稱，也提供電子郵件地址 jie@example.com。Amazon Cognito 會將代碼傳送至其電子郵件地址。

   範例 AWS CLI 命令

   ```
   aws cognito-idp sign-up --client-id 1234567890abcdef0 --username jie --password
    PASSWORD --user-attributes Name="email",Value="jie@example.com"
   ```

   回應範例

   ```
   {
       "UserConfirmed": false,
       "UserSub": "<subId>",
       "CodeDeliveryDetails": {
           "AttributeName": "email",
           "Destination": "j****@e****",
           "DeliveryMedium": "EMAIL"
       }
   }
   ```

2. Jie 提供傳送給他們的代碼，以確認他們對電子郵件地址的所有權。這樣就完成了他們作為使用者的註冊。

   範例 AWS CLI 命令

   ```
   aws cognito-idp confirm-sign-up --client-id 1234567890abcdef0 --username=jie --
   confirmation-code xxxxxx
   ```

3. Shirley 註冊一個新的使用者帳戶並提供電子郵件地址 jie@example.com。Amazon Cognito 不會傳回 UsernameExistsException 錯誤，並將確認代碼傳送到 Jie 的電子郵件地址。

   範例 AWS CLI 命令

   ```
   aws cognito-idp sign-up --client-id 1234567890abcdef0 --username shirley --password
    PASSWORD --user-attributes Name="email",Value="jie@example.com"
   ```

回應範例

```
{
    "UserConfirmed": false,
    "UserSub": "<new subId>",
    "CodeDeliveryDetails": {
        "AttributeName": "email",
        "Destination": "j****@e****",
        "DeliveryMedium": "EMAIL"
    }
}
```

4. 在不同的情況下，Shirley 擁有 jie@example.com。Shirley 會擷取 Amazon Cognito 傳送至 Jie 電子郵件地址的代碼，並嘗試確認帳戶。

   範例 AWS CLI 命令

   ```
   aws cognito-idp confirm-sign-up --client-id 1234567890abcdef0 --username=shirley --
   confirmation-code xxxxxx
   ```

   回應範例

   ```
   An error occurred (AliasExistsException) when calling the ConfirmSignUp operation: An
    account with the email already exists.
   ```

儘管 jie@example.com 已指派給現有使用者，Amazon Cognito 不會對 Shirley 的 aws cognito-idp sign-up 請求傳回錯誤。Shirley 必須先證明電子郵件地址的擁有權，Amazon Cognito 才會傳回錯誤回應。在具有別名屬性的使用者集區中，此行為會防止使用公有 SignUp API 來檢查指定的電子郵件地址或電話號碼是否有使用者。

此行為與 Amazon Cognito 以現有使用者名稱傳回 SignUp 請求的回應不同，如下列範例所示。雖然 Shirley 從此回應中得知已經有使用者名稱為 jie 的使用者，但他們不會得知任何與使用者相關聯的電子郵件地址或電話號碼。

CLI 命令範例

```
aws cognito-idp sign-up --client-id 1example23456789 --username jie --password PASSWORD
      --user-attributes Name="email",Value="shirley@example.com"
```

回應範例

```
An error occurred (UsernameExistsException) when calling the SignUp operation: User
 already exists
```

密碼重設操作

當您防止使用者存在錯誤時，Amazon Cognito 會對使用者密碼重設作業傳回下列回應。

ForgotPassword

當找不到使用者、使用者遭停用，或沒有經過驗證的傳遞機制可復原使用者的密碼時，Amazon Cognito 會為使用者傳回 `CodeDeliveryDetails` 以及模擬的傳遞媒介。模擬的傳遞媒體取決於使用者集區的輸入使用者名稱格式和驗證設定。

ConfirmForgotPassword

Amazon Cognito 會為不存在或已停用的使用者傳回 `CodeMismatchException` 錯誤。如果使用 `ForgotPassword` 時未要求代碼，Amazon Cognito 會傳回 `ExpiredCodeException` 錯誤。

確認操作

當您防止使用者存在錯誤時，Amazon Cognito 會傳回以下對使用者確認和驗證作業的回應。

ResendConfirmationCode

Amazon Cognito 會為已停用或不存在的使用者傳回 `CodeDeliveryDetails`。Amazon Cognito 會將確認碼傳送至現有使用者的電子郵件或電話號碼。

ConfirmSignUp

如果代碼已過期會傳回 `ExpiredCodeException`。使用者未獲授權時，Amazon Cognito 會傳回 `NotAuthorizedException`。如果代碼不符伺服器預期，Amazon Cognito 會傳回 `CodeMismatchException`。

Amazon Cognito 身分集區

Amazon Cognito 身分池是您可以交換 AWS 憑證的聯合身分目錄。身份池為您的應用程序的用戶生成臨時 AWS 憑據，無論他們已登錄還是您尚未識別它們。透過 AWS Identity and Access Management (IAM) 角色和政策，您可以選擇要授與使用者的權限層級。使用者可從訪客開始，擷取您保存在 AWS 服務中的資產。然後，他們可以使用第三方身分提供者登入，以解鎖您提供給註冊會員的資產存取權。第三方身分提供者可以是消費者 (社交) OAuth 2.0 提供者 (例如 Apple 或 Google)，自定義 SAML 或 OIDC 身分提供者，或者您自己設計的自定義身分驗證方案 (也稱為開發人員提供者)。

Amazon Cognito 身分池功能

簽署要求 AWS 服務

簽署 API 請求以 AWS 服務 喜歡 Amazon Simple Storage Service (Amazon S3) 和 Amazon DynamoDB。使用 Amazon Pinpoint 和 Amazon CloudWatch 等服務分析用戶活動。

使用以資源為基礎的政策篩選

對使用者存取您的資源進行精細控制。將使用者宣告轉換為 IAM 工作階段標籤，並建立 IAM 政策，將資源存取權授與使用者的不同子集。

指派訪客存取權

對於尚未登入的使用者，請設定您的身分池，以產生存取範圍更小的 AWS 憑證。透過單一登入提供者驗證使用者，以提升其存取權。

根據使用者特性指派 IAM 角色

為所有已驗證的使用者指派單一 IAM 角色，或根據每個使用者的宣告選擇角色。

接受各種身分提供者

將 ID 或存取權杖、使用者集區權杖、SAML 判斷提示或社交提供者 OAuth 權杖交換認證。 AWS

驗證您自己的身分

執行您自己的用戶驗證，並使用您的開發人員 AWS 憑據為您的用戶發出憑據。

您可能已經擁有 Amazon Cognito 使用者集區，可為您的應用程式提供身份驗證和授權服務。您可以將使用者集區身分提供者 (IdP) 設定為身分池。當您這樣做時，您的使用者可以透過您的使用者集區進行驗證 IdPs、將其宣告合併為通用的 OIDC 身分權杖，然後將該權杖交換為 AWS 憑證。然後，您的使用者可以在已簽署的請求中向您的 AWS 服務顯示憑證。

您也可以將來自任何身分提供者 的已驗證宣告直接提交到您的身分池。Amazon Cognito 會將 SAML、OAuth 和 OIDC 提供者的使用者宣告自訂為短期登入資料的 API AssumeRoleWithWebIdentity請求。

Amazon Cognito 使用者集區就像是已啟用 SSO 之應用程式的 OIDC 身分提供者。身分池充當任何資源依賴性具有 AWS 身分提供者，最適合與 IAM 授權搭配使用。

Amazon Cognito 身分集區支援下列身分提供者：

- 公開提供者：設置 Login with Amazon 作為身份池 IdP、將臉書設定為身分集區 IdP、將谷歌設置為身份池 IdP、設定以 Apple 身分識別集區 IdP 身分登入、Twitter。
- Amazon Cognito 使用者集區
- 將 OIDC 提供者設定為識別集區 IdP
- 將 SAML 提供者設定為身分識別集區 IdP
- 開發人員驗證的身分

如需 Amazon Cognito 身分集區區域可用性的相關資訊，請參閱 AWS 服務區域可用性。

如需 Amazon Cognito 身分集區的詳細資訊，請參閱下列主題。

主題

- 使用身分集區 (聯合身分)
- 身分集區概念
- Amazon Cognito 身分集區的安全性最佳實務
- 使用屬性進行存取控制
- 使用以角色為基礎的存取控制
- 取得憑證
- 存取 AWS 服務
- 外部身分提供者身分集區
- 開發人員驗證的身分
- 將未進行身分驗證使用者切換到已進行身分驗證使用者 (身分集區)

使用身分集區 (聯合身分)

Amazon Cognito 身分集區為訪客 (未驗證) 的使用者以及已通過驗證和接收權杖的使用者提供臨時 AWS 登入資料。身分集區是您的帳戶專屬的使用者身分資料存放區。

在主控台中建立新的身分集區

1. 登入 Amazon Cognito 主控台，然後選取 身分池。
2. 選擇 建立身分池。
3. 在 設定身分池信任 中，為您的身分池選擇 已驗證存取、訪客存取 設定，或同時選擇。
   - 如果您選擇 已驗證存取，請選取一或多個要設定為身分池中已驗證身分來源的 身分類型。如果您設定 自訂開發人員提供者，則建立身分池後無法修改或刪除。
4. 在 設定許可 中，為身分池中的身分驗證使用者或訪客使用者選擇預設 IAM 角色。
   a. 如果您希望 Amazon Cognito 為您建立具有基本許可和與身分池具有信任關係的新角色，請選擇 建立新 IAM 角色。輸入 IAM 角色名稱 以識別您的新角色，例如 `myidentitypool_authenticatedrole`。選取 檢視政策文件 以檢閱 Amazon Cognito 指派給您新 IAM 角色的許可。
   b. 如果您想要使用的角色中已有角色，則可以選擇使用現有的 AWS 帳戶 IAM 角色。您必須設定 IAM 角色信任政策以包含 `cognito-identity.amazonaws.com`。將您的角色信任政策設定為僅在 Amazon Cognito 提供證據，表明請求來源為特定身分池中已驗證的使用者時，才允許 Amazon Cognito 擔任該角色。如需詳細資訊，請參閱 角色信任和許可。
5. 在 [Connect 身分識別提供者] 中，輸入您在設定身分識別集區信任中選擇的身分識別提供者 (IdPs) 的詳細資料。系統可能會要求您提供 OAuth 應用程式用戶端資訊、選擇 Amazon Cognito 使用者集區、選擇 IAM IdP，或輸入開發人員供應商的自訂識別符。
   a. 為每個 IdP 選擇 角色設定。您可以為該 IdP 使用者指派設定 已驗證角色 時的 預設角色，或您可以 選擇具有規則的角色。使用 Amazon Cognito 使用者集區 IdP，您還可以 選擇權杖中具有 preferred_role 的角色。如需 `cognito:preferred_role` 宣告的詳細資訊，請參閱 指定優先順序值給群組。
      i. 如果您選擇 使用規則選擇角色，請輸入使用者身分驗證的 宣告 來源、比較宣告的 操作員、導致符合角色選擇的 值，以及當符合 角色指派 時您要指派的 角色。選取 新增另一項 以根據不同的條件建立其他規則。
      ii. 選擇 角色解析。當您的使用者宣告與您的規則不符時，您可以拒絕憑證或向 已驗證角色發出憑證。

b. 為每個 IdP 分別設定 存取控制屬性。存取控制屬性會將使用者宣告映射至 Amazon Cognito 套用至其臨時工作階段的委託人標籤。您可以建立 IAM 政策，根據您套用至其工作階段的索引標籤篩選使用者存取權限。

   i. 若不套用主要索引標籤，請選擇 非作用中。

   ii. 若要根據 sub 和 aud 宣告套用主要索引標籤，請選擇 使用預設對應。

   iii. 若要建立您自己的自訂屬性結構描述至主要索引標籤，請選擇 使用自訂對應。然後，輸入您要從每個 宣告 中獲取的 標籤金鑰，顯示於索引標籤當中。

6. 在 設定屬性 中，在 身分池名稱 下輸入一個 名稱。
7. 在 基本 (傳統) 身分驗證 下，選擇是否要 啟用基本流程。啟用基本流程後，您可以略過為您所做的角色選擇 IdPs 並AssumeRoleWithWebIdentity直接呼叫。如需詳細資訊，請參閱 身分集區 (聯合身分) 驗證流程。
8. 如果您要將 標籤 套用至身分池，請在索引 標籤 底下選擇 新增標籤。
9. 在 檢閱和建立 中，確認您為新身分池所做的選擇。選取 編輯 以返回精靈並變更任何設定。當您完成時，請選取 建立身分池。

使用者 IAM 角色

IAM 角色會定義使用者存取 AWS 資源的許可，例如Amazon Cognito Sync。您的應用程式的使用者將會獲得您建立的角色。您可以為已驗證和未驗證的使用者指定不同的角色。若要進一步了解 IAM 角色，請參閱 IAM 角色。

已進行身分驗證和未進行身分驗證的身分

Amazon Cognito 身分集區可支援已驗證和未驗證的身分。已驗證的身分屬於已由任何支援的身分提供者驗證的使用者。未驗證的身分則通常屬於訪客使用者。

- 若要以公有登入供應商來設定已驗證的身分，請參閱外部身分提供者身分集區。
- 若要設定您自己的後端身分驗證程序，請參閱開發人員驗證的身分。

啟用或停用訪客存取

Amazon Cognito 身分集區來賓存取 (未經驗證的身分) 為未向身分提供者進行驗證的使用者提供唯一的識別碼和 AWS 登入資料。如果您的應用程式允許未登入的使用者，就可以針對未驗證的身分啟用存取權。如需進一步了解，請參閱開始使用 Amazon Cognito 身分集區。

若要更新身分池中的訪客存取

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。
3. 找到 訪客存取。在目前不支援訪客存取的身分池中，狀態 為 非作用中。

   a. 如果 訪客存取 已 啟用，而您想要停用，請選取 停用。

   b. 如果 訪客存取 非作用中，而您想要使用，請選取 編輯。

      - 為身分池中的訪客使用者選擇預設 IAM 角色。

        A. 如果您希望 Amazon Cognito 為您建立具有基本許可和與身分池具有信任關係的新角色，請選擇 建立新 IAM 角色。輸入 IAM 角色名稱 以識別您的新角色，例如 `myidentitypool_authenticatedrole`。選取 檢視政策文件 以檢閱 Amazon Cognito 指派給您新 IAM 角色的許可。

        B. 如果您想要使用的角色中已有角色，則可以選擇使用現有的 AWS 帳戶 IAM 角色。您必須設定 IAM 角色信任政策以包含 `cognito-identity.amazonaws.com`。將您的角色信任政策設定為僅在 Amazon Cognito 提供證據，表明請求來源為特定身分池中已驗證的使用者時，才允許 Amazon Cognito 擔任該角色。如需詳細資訊，請參閱 角色信任和許可。

        C. 選取儲存變更。

        D. 若要啟用訪客存取權，請在 使用者存取權 索引標籤中選取 啟動。

變更與身分類型相關聯的角色

身分集區中的每個身分為已驗證或未驗證。已驗證的身分隸屬於由公有登入供應商 (Amazon Cognito 使用者集區、Login with Amazon、Sign in with Apple、Facebook、Google、SAML 或任何 OpenID Connect 供應商) 或開發人員供應商 (您自己的後端身分驗證程序) 驗證的使用者。未驗證的身分則通常屬於訪客使用者。

針對每種身分類型，都有指派的角色。此角色具有附加原則，指定該角色可 AWS 服務 存取的原則。當 Amazon Cognito 收到請求時，該服務會判斷身分類型、判斷指派給該身分類型的角色，並使用附屬於該角色的政策來回應。透過修改原則或將不同角色指派給身分識別類型，您可以控制哪些 AWS 服務 身分識別類型可以存取。若要檢視或修改與身分集區中的角色相關聯的政策，請參閱 AWS IAM 主控台。

若要變更身分池預設的已驗證的角色或未驗證的角色

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。
3. 找到 訪客存取 或 已驗證存取。在目前不支援設定訪客類型的身分池中，狀態 為 非作用中。選擇 Edit (編輯)。
4. 為身分池中的訪客或已驗證的使用者選擇預設 IAM 角色。

   a. 如果您希望 Amazon Cognito 為您建立具有基本許可和與身分池具有信任關係的新角色，請選擇 建立新 IAM 角色。輸入 IAM 角色名稱 以識別您的新角色，例如 `myidentitypool_authenticatedrole`。選取 檢視政策文件 以檢閱 Amazon Cognito 指派給您新 IAM 角色的許可。

   b. 如果您想要使用的角色中已有角色，則可以選擇使用現有的 AWS 帳戶 IAM 角色。您必須設定 IAM 角色信任政策以包含 `cognito-identity.amazonaws.com`。將您的角色信任政策設定為僅在 Amazon Cognito 提供證據，表明請求來源為特定身分池中已驗證的使用者時，才允許 Amazon Cognito 擔任該角色。如需詳細資訊，請參閱 角色信任和許可。
5. 選取儲存變更。

編輯身分提供者

如果您允許使用者以消費者身分提供者 (例如 Amazon Cognito 使用者集區、Login with Amazon、Sign in with Apple、Facebook 或 Google) 來進行身分驗證，您可以在 Amazon Cognito 身分池 (聯合身分) 主控台中指定您的應用程式識別符。這會將應用程式 ID (由公有登入供應商提供) 與您的身分集區建立關聯。

您也可以從這個頁面為每個供應商設定身分驗證規則。每個供應商最多可以有 25 個規則。這些規則會依照您為各個供應商儲存的順序套用。如需詳細資訊，請參閱 使用以角色為基礎的存取控制。

> **Warning**
>
> 變更身分池所連結的 IdP 應用程式 ID，將會使現有使用者無法向身分池進行身分驗證。如需詳細資訊，請參閱 外部身分提供者身分集區。

若要更新身分池身分提供者 (IdP)

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。

2. 選擇 使用者存取權 索引標籤。
3. 找到 身分提供者。選擇您要編輯的身分提供者。如果您要新增一個新的 IdP，選取 新增身分供應商。

   - 如果您選擇 新增身分供應商，請選擇一種您要新增的 身分類型。
4. 若要變更應用程式 ID，請選擇 身分提供者資訊 中的 編輯。
5. 若要變更 Amazon Cognito 向已透過此提供者驗證的使用者發布憑證時要求的角色，請選擇 角色設定中的 編輯。

   - 您可以為該 IdP 使用者指派設定 已驗證角色 時的 預設角色，或您可以 選擇具有規則的角色。使用 Amazon Cognito 使用者集區 IdP，您還可以 選擇權杖中具有 preferred_role 的角色。如需 `cognito:preferred_role` 宣告的詳細資訊，請參閱 指定優先順序值給群組。

     i. 如果您選擇 使用規則選擇角色，請輸入使用者身分驗證的 宣告 來源、比較宣告的 操作員、導致符合角色選擇的 值，以及當符合 角色指派 時您要指派的 角色。選取 新增另一項 以根據不同的條件建立其他規則。

     ii. 選擇 角色解析。當您的使用者宣告與您的規則不符時，您可以拒絕憑證或向 已驗證角色發出憑證。
6. 若要變更 Amazon Cognito 向使用者驗證此提供者，發布憑證時指派的委託人標籤，請在 存取控制屬性 中選擇 編輯。

   a. 若不套用主要索引標籤，請選擇 非作用中。

   b. 若要根據 sub 和 aud 宣告套用主要索引標籤，請選擇 使用預設對應。

   c. 若要建立您自己的自訂屬性結構描述至主要索引標籤，請選擇 使用自訂對應。然後，輸入您要從每個 宣告 中獲取的 標籤金鑰，顯示於索引標籤當中。
7. 選取儲存變更。

刪除身分集區

您無法復原刪除的身分池。刪除身分池後，所有依賴該身分池的應用程式和使用者都會停止運作。

刪除身分集區

1. 從 Amazon Cognito 主控台選擇 身分池。在您要刪除的身分池旁邊，選取選項按鈕。
2. 選取刪除。
3. 輸入或貼上身分池的名稱，然後選取 刪除。

> **Warning**
>
> 當您選取 Delete (刪除) 按鈕，將會永久刪除您的身分集區及其包含的所有使用者資料。刪除身分池會導致應用程式和使用該身分池的其他服務停止運作。

從身分集區刪除身分

從身分池刪除身分時，您會移除 Amazon Cognito 為該聯合身分使用者儲存的身分資訊。當您的使用者再次要求認證時，如果您的身分池仍信任其身分提供者，他們會收到新的身分 ID。您無法復原此操作。

若要刪除身分

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 身分瀏覽器 索引標籤。
3. 在您要刪除的身分旁邊，選取核取方塊並選擇 刪除。確認您要刪除身分，然後選擇 刪除。

使用 Amazon Cognito Sync 與身分集區

Amazon Cognito Sync 是一個 AWS 服務 和用戶端程式庫，可讓您跨裝置同步應用程式相關的使用者資料。Amazon Cognito Sync 可跨行動裝置和 Web 同步使用者描述檔資料，無需使用自己的後端系統。用戶端程式庫會在本機快取資料，因此無論裝置的連線狀態為何，您的應用程式可以讀取和寫入資料。當裝置上線時，您可以同步資料。如果您設定推播同步，可立即通知其他裝置有更新可用。

管理資料集

如果您已經在應用程式中實作 Amazon Cognito Sync 功能，Amazon Cognito 身分集區主控台可以讓您針對個別的身分手動建立及刪除資料集和記錄。如果您在 Amazon Cognito 身分集區中對身分的資料集或記錄進行任何變更，則必須在主控台中選取 Synchronize (同步) 才會儲存變更。在身分呼叫 Synchronize (同步)之後，最終使用者才看得到這些變更。您只要重新整理特定身分的列出資料集頁面之後，就會針對個別身分顯示從其他裝置同步的資料。

為身分建立資料集

Amazon Cognito Sync 將資料集與一個身分建立關聯。您可以在資料集中填入身分所代表之使用者的相關資訊，然後將該資訊同步至您使用者的所有裝置。

若要將資料集和資料集記錄新增至身分

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 身分瀏覽器 索引標籤。
3. 選取您要編輯的身分。
4. 在 資料集 中，選擇 建立資料集。
5. 輸入 資料集名稱，然後選取 建立資料集。
6. 如果要將記錄新增至資料集，請從身分詳細資料中選擇資料集。在 記錄 中，選取 建立記錄。
7. 輸入記錄的 金鑰 和 值。選擇確認。重複此步驟以新增更多記錄。

刪除與身分相關聯的資料集

若要從身分中刪除資料集及其記錄

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 身分瀏覽器 索引標籤。
3. 選取包含您要刪除之資料集的身分。
4. 在 資料集 中，選擇要刪除之資料集旁的選項按鈕。
5. 選取刪除。檢視您的選擇並再次選取 刪除。

大量發佈資料

大量發佈可將已存放在 Amazon Cognito Sync 存放區中的資料匯出至 Amazon Kinesis 串流。如需有關如何大量發佈您的所有串流的詳細資訊，請參閱 Amazon Cognito 串流。

啟用推播同步

Amazon Cognito 會自動追蹤身分與裝置之間的關聯。使用推播同步功能可讓您確保在身分資料變更時，指定身分的每個執行個體都會收到通知。推播同步可確保，每當身分的資料集變更時，與該身分相關聯的所有裝置都會收到靜音推播通知，通知他們有所變更。

您可以在 Amazon Cognito 主控台內啟用推播同步。

若要啟用推播同步

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 身分池屬性 標籤。

3. 在 推播同步 中，選取 編輯
4. 選取 使用身分池啟用推播同步。
5. 選擇您目前在 AWS 區域中建立的其中一個 Amazon Simple Notification Service (Amazon SNS) 平台應用程式。Amazon Cognito 會將推播通知發布到您的平台應用程式。選取 建立平台應用程式以前往 Amazon SNS 主控台並建立新的應用程式。
6. 若要將其發布至您的平台應用程式，Amazon Cognito 會在您的 AWS 帳戶中指派 IAM 角色。如果您希望 Amazon Cognito 為您建立具有基本許可和與身分池具有信任關係的新角色，請選擇 建立新 IAM 角色。輸入 IAM 角色名稱 以識別您的新角色，例如 `myidentitypool_authenticatedrole`。選取 檢視政策文件 以檢閱 Amazon Cognito 指派給您新 IAM 角色的許可。
7. 如果您想要使用的角色中已有角色，則可以選擇使用現有的 AWS 帳戶 IAM 角色。您必須設定 IAM 角色信任政策以包含 `cognito-identity.amazonaws.com`。將您的角色信任政策設定為僅在 Amazon Cognito 提供證據，表明請求來源為特定身分池中已驗證的使用者時，才允許 Amazon Cognito 擔任該角色。如需詳細資訊，請參閱 角色信任和許可。
8. 選取儲存變更。

設定 Amazon Cognito 串流

Amazon Cognito 串流可讓開發人員控制並深入了解其存放在 Amazon Cognito Sync 中的資料。開發人員現在可以設定 Kinesis 串流，以資料的形式來接收事件。Amazon Cognito 可以即時將每項資料集變更推送至您擁有的 Kinesis 串流。如需有關如何在 Amazon Cognito 主控台中設定 Amazon Cognito 串流的指示，請參閱 Amazon Cognito 串流。

設定 Amazon Cognito 事件

Amazon Cognito 事件可讓您執行 AWS Lambda 函數以回應 Amazon Cognito 同步中的重要事件。當有資料集同步時，Amazon Cognito Sync 會引發同步觸發器事件。當使用者更新資料時，您可以使用同步觸發器事件來採取動作。如需有關如何從主控台設定 Amazon Cognito 事件的指示，請參閱 Amazon Cognito 事件。

若要進一步瞭解 AWS Lambda，請參閱AWS Lambda。

身分集區概念

您可以使用 Amazon Cognito 身分集區為使用者建立唯一身分，並透過身分提供者驗證使用者。透過身分識別，您可以取得臨時、有限權限的 AWS 登入資料，以存取其他 AWS 服務身分。Amazon

Cognito 身分集區支援公有身分提供者 (Amazon、Apple、Facebook 和 Google)，以及未驗證的身分。它也支援開發人員驗證的身分，可讓您透過自己的後端身分驗證程序來註冊及驗證使用者。

如需 Amazon Cognito 身分集區區域可用性的相關資訊，請參閱 AWS 服務區域可用性。如需 Amazon Cognito 身分集區概念的詳細資訊，請參閱下列主題。

主題

- 身分集區 (聯合身分) 驗證流程
- IAM 角色
- 角色信任和許可

身分集區 (聯合身分) 驗證流程

Amazon Cognito 可協助您為最終使用者建立唯一識別符，以在各種裝置和平台之間保持一致性。Amazon Cognito 也會向您的應用程式提供有限權限的臨時登入資料，以存取 AWS 資源。本頁面涵蓋身分驗證在 Amazon Cognito 中的運作方式基本概念，並說明身分在身分集區中的生命週期。

外部供應商身分驗證流程

向 Amazon Cognito 驗證的使用者會進行多步驟的程序來自舉其憑證。Amazon Cognito 有兩種向公有供應商進行身分驗證的不同流程：強化和基本。

完成上述其中一個流程後，您就可以存取角色 AWS 服務 存取原則所定義的其他流程。Amazon Cognito 主控台預設會建立可存取 Amazon Cognito Sync 存放區和 Amazon Mobile Analytics 的角色。如需有關如何授予其他存取權的詳細資訊，請參閱 IAM 角色。

身分集區接受來自提供者的下列人工因素：

供應商	驗證加工品
Amazon Cognito 使用者集區	ID 權杖
OpenID Connect (OIDC)	ID 權杖
SAML 2.0	SAML 斷言
社交提供者	存取權杖

強化 (簡化) 身分驗證流程

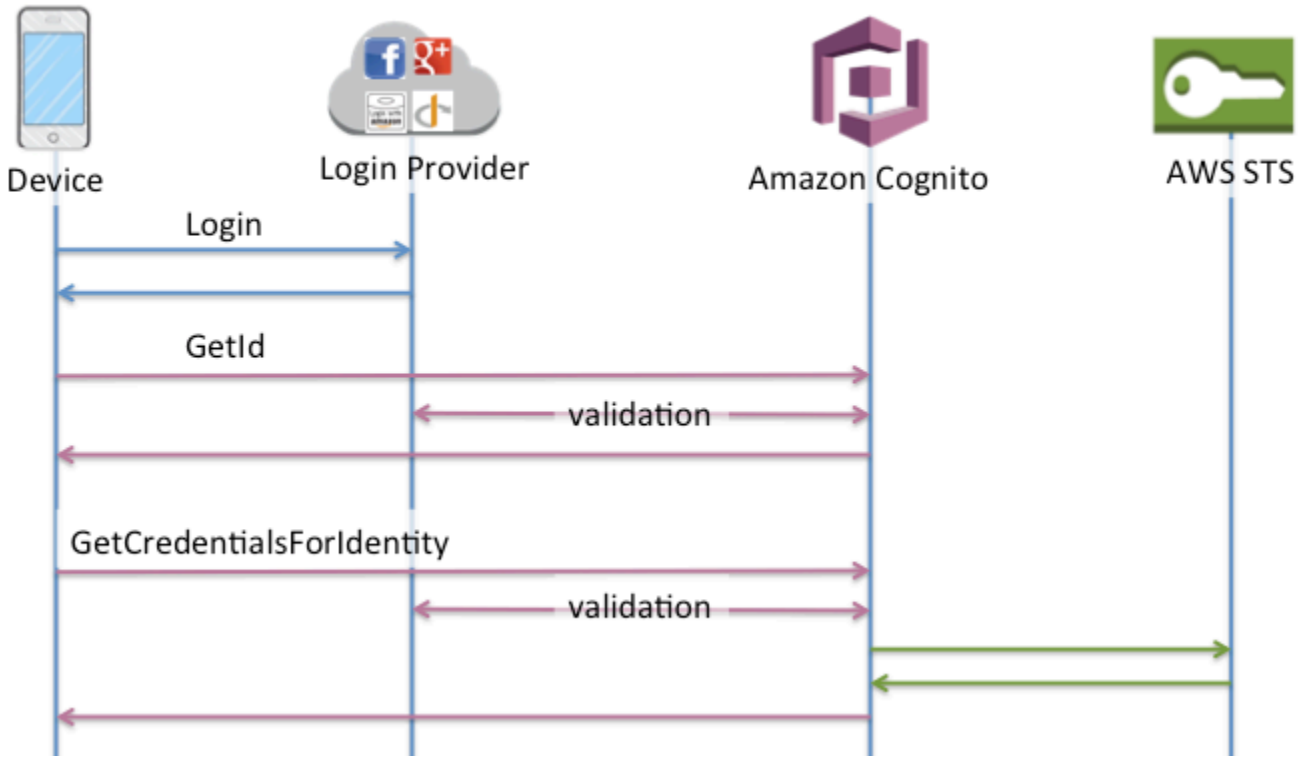
當您使用增強型 authflow 時，您的應用程式會先在請求中提供來自授權 Amazon Cognito 使用者集區或第三方身分提供者的身份驗證證明。GetId

1. 您的應用程式會顯示 GetID 請求中授權的 Amazon Cognito 使用者集區或第三方身分提供者的驗證證明 (JSON 網頁權杖或 SAML 宣告)。
2. 您的身分集區會傳回身分識別碼。
3. 您的應用程式會在GetCredentialsForIdentity要求中結合身分識別 ID 與相同的驗證證明。
4. 您的身分集區會傳回 AWS 認證。
5. 您的應用程式會使用臨時登入資料簽署 AWS API 要求。

增強型身分驗證可管理身分集區設定中 IAM 角色選擇和登入資料擷取的邏輯。您可以將身分識別集區設定為選取預設角色，將屬性型存取控制 (ABAC) 或角色型存取控制 (RBAC) 原則套用至角色選取。增強型驗證的 AWS 認證有效期為一小時。

增強型驗證中的作業順序

1. `GetId`
2. `GetCredentialsForIdentity`



基本 (傳統) 身分驗證流程

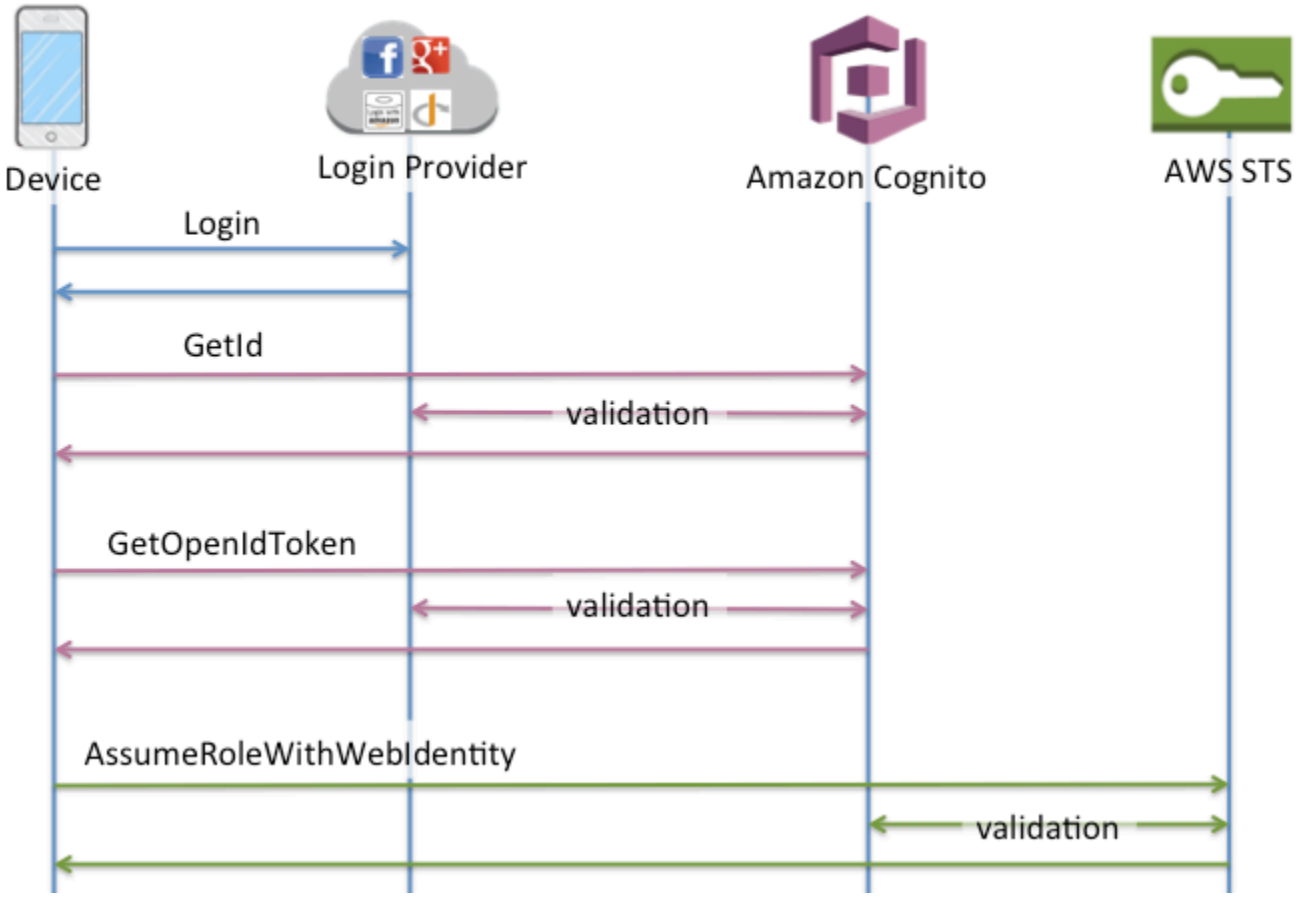
當您使用基本驗證流程時，

1. 您的應用程式會顯示 GetID 請求中授權的 Amazon Cognito 使用者集區或第三方身分提供者的驗證證明 (JSON 網頁權杖或 SAML 宣告)。
2. 您的身分集區會傳回身分識別碼。
3. 您的應用程式會在GetOpenIdToken要求中結合身分識別 ID 與相同的驗證證明。
4. GetOpenIdToken返回由您的身份池發行的新 OAuth 2.0 令牌。
5. 您的應用程序在AssumeRoleWithWebIdentity請求中顯示新令牌。
6. AWS Security Token Service AWS STS) 會傳回 AWS 認證。
7. 您的應用程式會使用臨時登入資料簽署 AWS API 要求。

基本工作流程可讓您更精密控制自己分配給使用者的憑證。強化身分驗證流程的 GetCredentialsForIdentity 請求會根據存取權杖的內容請求角色。傳統工作流程中的AssumeRoleWithWebIdentity要求可讓您的應用程式更有能力為您已設定足夠信任原則的任何 AWS Identity and Access Management 角色要求認證。您也可以請求自訂角色工作階段持續時間。

基本驗證中的操作順序

1. GetId
2. GetOpenIdToken
3. AssumeRoleWithWebIdentity



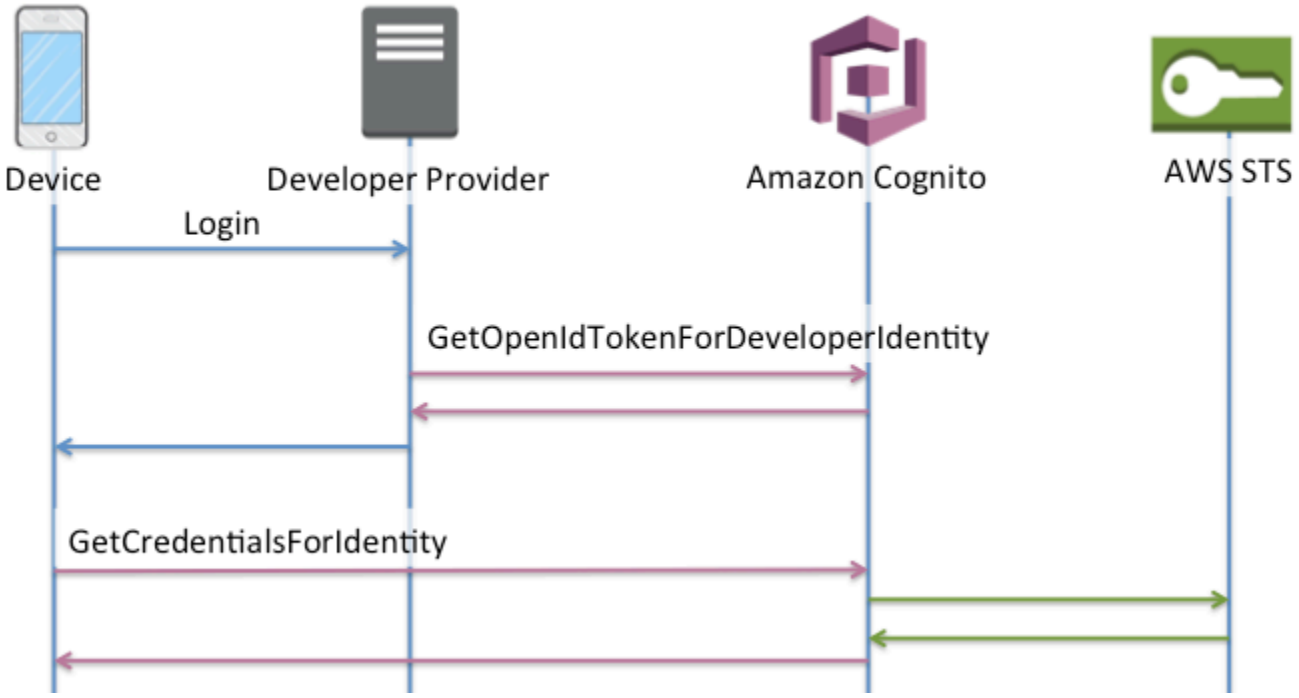
開發人員驗證的身分驗證流程

使用 開發人員驗證的身分 時，用戶端使用包含 Amazon Cognito 外部程式碼的不同身分驗證流程，來驗證您自己身分驗證系統中的使用者。Amazon Cognito 外部程式碼就是指這個意思。

強化身分驗證流程

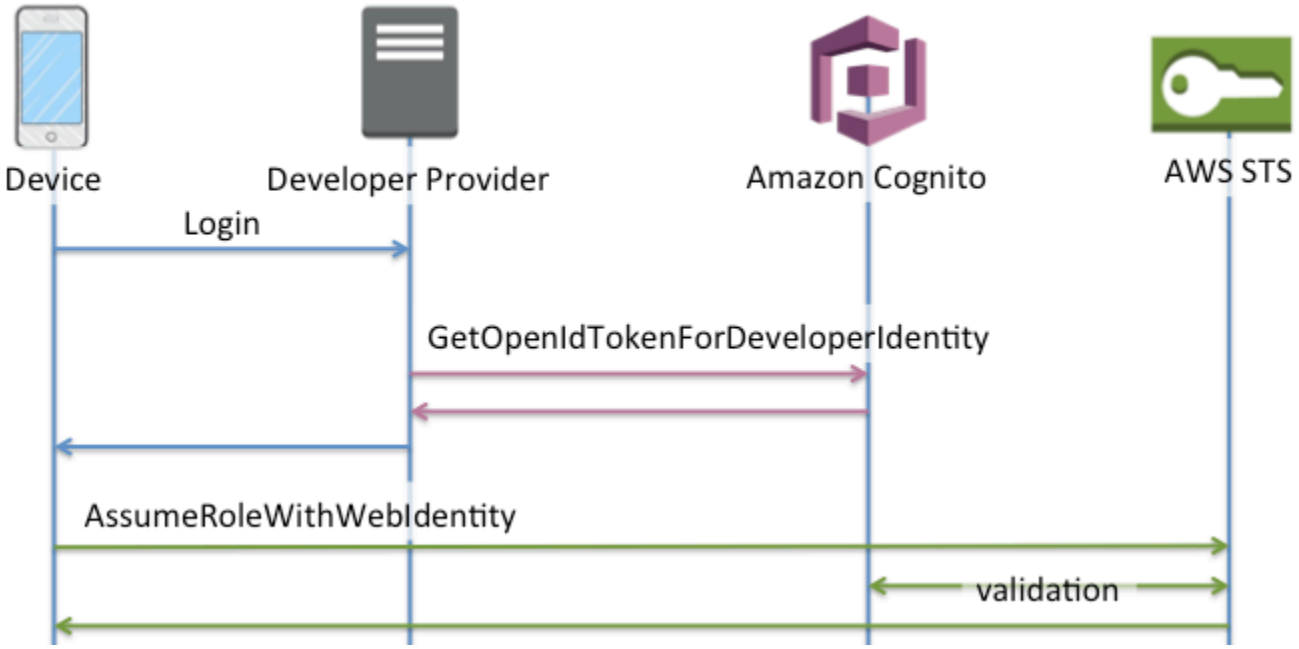
使用開發人員提供者的增強型驗證中的作業順序

1. 透過開發人員供應商 (Amazon Cognito 外部程式碼) 登入
2. 驗證使用者登入 (Amazon Cognito 外部程式碼)
3. GetOpenIdTokenForDeveloperIdentity
4. GetCredentialsForIdentity



開發人員提供程式的基本驗證中的作業順序

1. 在身分集區之外實作邏輯，以登入並產生開發人員提供者識別碼。
2. 擷取儲存伺服器端 AWS 認證。
3. 在使用授權 AWS 憑證簽署的 GetOpenIdTokenForDeveloperIdentityAPI 要求中傳送開發人員提供者識別碼。
4. 使用要求應用程式認證AssumeRoleWithWebIdentity。



我應該使用哪一種身分驗證流程？

增強的流程是最安全的選擇，以最低的開發人員努力：

- 增強的流程可降低 API 請求的複雜性、大小和速率。
- 您的應用程式不需要向 AWS STS.
- 您的身分集區會評估使用者應收到的 IAM 角色登入資料。您不需要在用戶端中嵌入角色選擇的邏輯。

> **Important**
>
> 當您建立新的身分識別集區時，預設不要啟用基本 (傳統) 驗證，做為最佳作法。若要實作基本身分驗證，請先評估 IAM 角色的 Web 身分的信任關係。然後在用戶端中建置角色選擇的邏輯，並保護用戶端不受使用者修改的影響。

基本身份驗證流程會將 IAM 角色選擇的邏輯委派給您的應用程式。在此流程中，Amazon Cognito 會驗證使用者的已驗證或未驗證工作階段，並發出可用來交換登入資料的權杖。 AWS STS用戶可以將基本身份驗證中的令牌交換為任何信任您身份集區和`amr`（或已驗證/未驗證狀態）的 IAM 角色。

同樣地，請瞭解開發人員驗證是驗證身分提供者驗證的捷徑。Amazon Cognito 信任授權請求的 AWS 登入資料，而無需對GetOpenIdTokenForDeveloperIdentity請求內容進行額外驗證。保護使用者存取授權開發人員驗證的密碼。

API 摘要

GetId

GetIdAPI 呼叫是在 Amazon Cognito 中建立新身分所需的第一個呼叫。

未驗證的存取

Amazon Cognito 可在您的應用程式中允許未驗證的訪客存取。如果在您的身分集區中啟用此功能，使用者就可以隨時透過 `GetId` API 來請求新的身分 ID。應用程式應該會快取此身分 ID 來對 Amazon Cognito 進行後續的呼叫。行 AWS 動 SDK 和瀏覽器 JavaScript 中的 AWS SDK 具有可為您處理此快取的認證提供者。

已驗證的存取

如果您為應用程式設定公有登入供應商 (Facebook、Google+、Login with Amazon 或 Sign in with Apple) 的支援，使用者也可以提供在那些供應商中識別其身分的權杖 (OAuth 或 OpenID Connect)。在對 `GetId` 的呼叫中使用時，Amazon Cognito 會建立新的已驗證身分，或傳回已經與該特定登入建立關聯的身分。Amazon Cognito 執行此作業的方式，是向供應商驗證權杖，並確保下列事項：

- 權杖有效，且是來自所設定的供應商。
- 權杖未過期。
- 權杖符合以該供應商 (例如，Facebook 應用程式 ID) 建立的應用程式識別符。
- 權杖符合使用者識別符。

GetCredentialsForIdentity

建立身分識別碼後，即可呼叫 GetCredentialsForIdentityAPI。這個操作在功能上等同於調用 GetOpenIdToken，然後AssumeRoleWithWebIdentity。

若要讓 Amazon Cognito 代表您來呼叫 `AssumeRoleWithWebIdentity`，您的身分集區必須要有與其相關聯的 IAM 角色。您可以透過 Amazon Cognito 主控台執行此操作，也可以透過操作手動執行此SetIdentityPoolRoles操作。

GetOpenIdToken

建立身分識別碼後提出 GetOpenIdTokenAPI 要求。在您的第一個請求之後快取身分 ID，並使用 `GetOpenIdToken` 啟動該身分的後續基本 (傳統) 工作階段。

對 GetOpenIdToken API 請求的回應是 Amazon Cognito 產生的權杖。您可以在AssumeRoleWithWebIdentity請求中提交此令牌作為WebIdentityToken參數。

在提交 OpenID 權杖之前，請先在您的應用程式中進行驗證。您可以在 SDK 或程式庫 (例如 aws-jwt-verify) 中使用 OIDC 程式庫，以確認 Amazon Cognito 已發行權杖。OpenID 權杖的簽署金鑰 ID (或 kid) 是 Amazon Cognito 身分 jwks_uri 文件† 中列出的項目之一。這些金鑰可能會有所變更。您用來驗證 Amazon Cognito 身分權杖的函數應該定期從 jwks_uri 文件更新其金鑰清單。Amazon Cognito 會在 jwks_uri 快取控制回應標頭中設定重新整理持續時間，目前將 max-age 設定為 30 天。

未驗證的存取

若要為未驗證的身分取得權杖，您只需要身分 ID 本身。針對已驗證的身分或您已停用的身分，無法取得未驗證的權杖。

已驗證的存取

如果您有已驗證的身分，您必須針對已經與該身分建立關聯的登入，傳遞至少一個有效的權杖。在 GetOpenIdToken 呼叫期間傳入的所有權杖，都必須通過前面提及的相同驗證；如果有任何權杖失敗，整個呼叫都會失敗。GetOpenIdToken 呼叫的回應也會包含身分 ID。這是因為您傳入的身分 ID 可能不是所傳回的身分 ID。

連結登入

如果針對尚未與任何身分建立關聯的登入提交權杖，該登入會被視為「連結」至相關聯的身分。一個公有供應商只能連結一個登入。如果嘗試將多個登入與一個公有供應商連結，會產生 ResourceConflictException 錯誤回應。如果登入只連結至現有的身分，GetOpenIdToken 傳回的身分 ID 會與傳入的身分 ID 相同。

合併身分

如果您針對目前未連結至指定身分，但已連結至另一個身分的登入傳入權杖，則兩個身分會合併。一旦合併，其中一個身分就會成為所有關聯登入的父項/擁有者，而另一個身分會停用。在這種情況下，會傳回父項/擁有者的身分 ID。如果此值不同，您必須更新本地快取。瀏覽器中的行 AWS 動 SDK 或 AWS SDK JavaScript 中的提供者會為您執行此作業。

GetOpenIdTokenForDeveloperIdentity

使用開發人員驗證身分時，此GetOpenIdTokenForDeveloperIdentity作業會取代裝置的使用GetId和GetOpenIdToken來自裝置的使用。由於您的應用程式使用 AWS 登入資料簽署此 API 作業的請求，因此 Amazon Cognito 信任請求中提供的使用者識別碼有效。開發人員身份驗證會取代 Amazon Cognito 對外部供應商執行的權杖驗證。

此 API 的裝載包括一個`logins`映射。此映射必須包含開發人員提供程序的密鑰和值作為系統中用戶的標識符。如果使用者識別符尚未連結至現有的身分，Amazon Cognito 會建立一個新的身分，並傳回新的身分 ID，以及該身分的 OpenID Connect 權杖。如果使用者識別符已經連結，Amazon Cognito 會傳回既有的身分 ID 和 OpenID Connect 權杖。在您的第一個請求之後快取開發人員身分 ID，並使用 `GetOpenIdTokenForDeveloperIdentity` 啟動該身分的後續基本 (傳統) 工作階段。

對 `GetOpenIdTokenForDeveloperIdentity` API 請求的回應是 Amazon Cognito 產生的權杖。您可以在 `AssumeRoleWithWebIdentity` 請求中提交此權杖作為 `WebIdentityToken` 參數。

在提交 OpenID Connect 權杖之前，請先在您的應用程式中進行驗證。您可以在 SDK 或程式庫 (例如 aws-jwt-verify) 中使用 OIDC 程式庫，來確認 Amazon Cognito 已發行權杖。OpenID Connect 權杖的簽署金鑰 ID (`kid`) 是 Amazon Cognito 身分 jwks_uri 文件中列出的項目之一。這些金鑰可能會有所變更。您用來驗證 Amazon Cognito 身分權杖的函數應該定期從 jwks_uri 文件更新其金鑰清單。Amazon Cognito 會在 jwks_uri `cache-control` 回應標頭中設定重新整理持續時間，目前將 `max-age` 設定為 30 天。

連結登入

如同外部供應商，提供尚未與身分建立關聯的其他登入，將會隱含地將這些登入連結至該身分。如果您將外部供應商登入連結至某身分，使用者就可以透過該供應商來使用外部供應商身分驗證流程。但是在呼叫 `GetId` 或 `GetOpenIdToken` 時，無法在登入對應中使用您的開發人員供應商名稱。

合併身分

透過開發人員驗證身分，Amazon Cognito 支援透過 MergeDeveloperIdentitiesAPI 進行隱含合併和明確合併。透過明確合併，您可將系統中具有使用者識別符的兩個身分標記成單一身分。如果您提供來源和目標使用者識別符，Amazon Cognito 會將其合併。下次當您為其中任一使用者識別符來請求 OpenId Connect 權杖時，會傳回相同的身分 ID。

AssumeRoleWithWebIdentity

擁有 OpenID Connect 令牌後，您可以通過 AssumeRoleWithWebIdentityAPI 請求將其交易為臨時 AWS 憑據 AWS Security Token Service （ AWS STS ）。

由於沒有限制您可建立的身分數量，所以請務必了解您授予使用者的許可。為您的應用程式設定不同的 IAM 角色：一個用於未驗證的使用者，另一個用於已驗證的使用者。當您第一次設定身分集區時，Amazon Cognito 主控台可以建立預設角色。這些角色實際上沒有授予任何權限。修改它們以滿足您的需求。

進一步了解 角色信任和許可。

† 預設的 Amazon Cognito 身分 jwks_uri 文件包含在大部分 AWS 區域中為身分集區簽署權杖的金鑰相關資訊。下列區域有不同的 jwks_uri 文件。

Amazon Cognito Identity JSON web key URIs in other AWS 區域

AWS 區域	jwks_uri 文件的路徑
AWS GovCloud (美國西部)	`https://cognito-identity.us-gov-west-1.amazonaws.com/.well-known/jwks_uri`
中國 (北京)	`https://cognito-identity.cn-north-1.amazonaws.com.cn/.well-known/jwks_uri`
選擇加入歐洲（米蘭）和非洲（開普敦）等地區	`https://cognito-identity.`*`Region`*`.amazonaws.com/.well-known/jwks_uri`

您也可以推斷來自發行者的 jwks_uri 或您在 Amazon Cognito 的 OpenID 權杖收到的 `iss`。OIDC 標準探索端點 `<issuer>/.well-known/openid-configuration` 會列出權杖之 jwks_uri 的路徑。

IAM 角色

建立身分集區時，系統會提示您更新使用者所擔任的 IAM 角色。IAM 角色的運作方式如下：當使用者登入您的應用程式時，Amazon Cognito 會為使用者產生臨時 AWS 登入資料。這些暫時登入資料會與特定 IAM 角色建立關聯。使用 IAM 角色，您可以定義一組許可以存取資 AWS 源。

您可以為已驗證和未驗證的使用者指定預設的 IAM 角色。此外，您也可以定義規則，以依據使用者 ID 權杖中的聲明，為每個使用者選擇角色。如需詳細資訊，請參閱 使用以角色為基礎的存取控制。

Amazon Cognito 主控台預設會建立 IAM 角色，可存取 Amazon Mobile Analytics 和 Amazon Cognito Sync。或者，您也可以選擇使用現有 IAM 角色。

修改 IAM 角色以允許或限制存取其他服務。若要執行此作業，請登入 IAM 主控台。然後選取 Roles (角色)，再選取角色。附屬於所選角色的政策列示在 Permissions (許可) 索引標籤中。您可以選取對應

的 Manage Policy (管理政策) 連結，以自訂存取政策。若要進一步了解如何使用及定義政策，請參閱 IAM 政策概觀。

> **Note**
>
> 根據最佳實務，請定義遵循授予最低權限原則的政策。換言之，政策僅包含使用者執行任務所需要的許可。如需詳細資訊，請參閱《IAM 使用者指南》中的授予最低權限。
> 請記住，未登入您應用程式的使用者會擔任未驗證的身分。一般而言，您指派給未驗證身分的許可，應該比已驗證身分的許可更具限制性。

主題

- 設定信任政策
- 存取政策

設定信任政策

Amazon Cognito 使用 IAM 角色為您的應用程式使用者產生暫時憑證。對許可的存取權是由角色的信任關係來控制。進一步了解 角色信任和許可。

所呈現的權杖 AWS STS 是由身分集區產生，該身分集區會將使用者集區、社交或 OIDC 提供者權杖或 SAML 判斷提供者權杖轉換為其自己的權杖。身分集區權杖包含身分集區 ID 的 aud 宣告。

下列範例角色信任原則允許同盟服務主`cognito-identity.amazonaws.com`體呼叫 AWS STS API `AssumeRoleWithWebIdentity`。只有在 API 請求中的身分集區權杖具有以下宣告時，請求才會成功。

1. 身分集區識別碼 `us-west-2:abcdefg-1234-5678-910a-0e8443553f95` 的 aud 宣告。
2. 當使用者已登入且不是來賓使用者時，新增的`authenticated` amr 宣告。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "cognito-identity.amazonaws.com"
            },
```

```
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "cognito-identity.amazonaws.com:aud": "us-
west-2:abcdefg-1234-5678-910a-0e8443553f95"
                },
                "ForAnyValue:StringLike": {
                    "cognito-identity.amazonaws.com:amr": "authenticated"
                }
            }
        }
    ]
}
```

基本 (傳統) 身份驗證中 IAM 角色的信任政策

您必須至少套用一個條件，以限制與身分識別集區搭配使用之角色的信任原則。當您建立或更新身分識別集區的角色信任政策時，如果您嘗試儲存變更，但沒有至少一個限制來源身分的條件金鑰，IAM 會傳回錯誤訊息。 AWS STS 不允許跨帳戶AssumeRoleWithWebIdentity操作，從身分集區到缺少此類型條件的 IAM 角色。

本主題包含數個限制身分識別集區之來源身分識別的條件。如需完整清單，請參閱 AWS Web 身分同盟的可用金鑰。

在具有身分集區的基本或傳統身份驗證中， AWS STS 如果具有正確的信任策略，則可以承擔任何 IAM 角色。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。此組態不足以保護您的 IAM 角色，以防止非預期的資源存取。此類型的角色必須將其他條件套用至角色信任原則。如果沒有下列條件之一，您就無法建立或修改身分識別集區的角色。

cognito-identity.amazonaws.com:aud

將角色限制為來自一或多個識別集區的作業。Amazon Cognito 會指出身分集區權杖中aud宣告中的來源身分集區。

cognito-identity.amazonaws.com:amr

將角色限制為authenticated或 unauthenticated (訪客) 使用者。Amazon Cognito 會指出身分集區權杖中amr宣告中的身分驗證狀態。

cognito-identity.amazonaws.com:sub

透過 UUID 將角色限制為一個或多個使用者。此 UUID 是身分集區中使用者的身分識別碼。此值不是來自使用者原始身分識別提供者的sub值。Amazon Cognito 在身分集區令牌的sub聲明中表示此 UUID。

增強流程身份驗證要求 IAM 角色與身分集區 AWS 帳戶 相同，但基本身份驗證並非如此。

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，且必須包含特定cognito-identity.amazonaws.com:aud條件。為了防止意外存取您的 AWS 資源，aud條件索引鍵會將角色限制為使用者從條件值中的身分集區。

身分集區為身份發出的令牌包含有關身份集區來源 AWS 帳戶 的信息。當您在AssumeRoleWithWebIdentityAPI 請求中顯示身分集區權杖時，請 AWS STS 檢查原始身分集區是否與 IAM 角色 AWS 帳戶 相同。如果 AWS STS 判斷要求是跨帳戶，它會檢查角色信任原則是否有aud條件。如果角色信任原則中沒有這樣的條件，則 assume-role 呼叫會失敗。如果請求不是跨帳戶，則不 AWS STS 會強制執行此限制。最佳作法是一律將此類型的條件套用至身分識別集區角色的信任原則。

其他信任政策條件

跨身分集區重複使用角色

若要跨多重身分集區重複使用角色 (因為這些集區共用一組通用的許可)，您可以併入多個身分集區，如下所示：

```
"StringEquals": {
    "cognito-identity.amazonaws.com:aud": [
        "us-east-1:12345678-abcd-abcd-abcd-123456790ab",
        "us-east-1:98765432-dcba-dcba-dcba-123456790ab"
    ]
}
```

限制對特定身分的存取權

若要建立政策，並僅限用於一組特定的應用程式使用者，請勾選 cognito-identity.amazonaws.com:sub值：

```
"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"
    ]
}
```

限制對特定供應商的存取權

若要建立政策，並僅限用於透過特定供應商 (可能是您自己的登入供應商) 登入的使用者，請勾選 `cognito-identity.amazonaws.com:amr`值：

```
"ForAnyValue:StringLike": {
    "cognito-identity.amazonaws.com:amr": "login.myprovider.myapp"
}
```

例如，只信任 Facebook 的應用程式會有下列 amr 子句：

```
"ForAnyValue:StringLike": {
    "cognito-identity.amazonaws.com:amr": "graph.facebook.com"
}
```

存取政策

您附加至角色的許可會套用至擔任該角色的所有使用者。若要將使用者的存取權分割，請使用政策條件和變數。如需詳細資訊，請參閱 IAM 政策元素：變數和標籤。您可以使用此 sub 條件，在存取政策中限制對 Amazon Cognito 身分 ID 的動作。請謹慎使用此選項，尤其是針對缺少一致使用者 ID 的未驗證身分。如需有關與 Amazon Cognito 進行網路聯合的 IAM 政策變數的詳細資訊，請參閱AWS Identity and Access Management 使用者指南中的 IAM 和 AWS STS 條件內容金鑰。

為了提升安全性保護，Amazon Cognito 針對您在強化流程中指派未驗證的使用者，使用 `GetCredentialsForIdentity` 對憑證套用縮減規模政策。縮減規模政策會在您套用至未經身分驗證角色的 IAM 政策中新增 內嵌工作階段政策 和 AWS 受管理工作階段。由於您必須同時針對角色和工作階段政策授予 IAM 政策的存取，因此縮減規模政策會限制使用者存取下列清單外的服務。

> **Note**
>
> 在基本 (傳統) 流程中，您可以發出自己的 AssumeRoleWithWebIdentity API 請求，並將這些限制套用至請求。對於最佳安全實務，請勿將此縮減規模政策以上的任何許可指派給未經身分驗證的使用者。

Amazon Cognito 也可防止已驗證和未經驗證的使用者向 Amazon Cognito 身分集區和 Amazon Cognito Sync 發出 API 請求。其他 AWS 服務 可能會限制來自 Web 身分的服務存取。

在具有強化流程的成功請求中，Amazon Cognito 會在背景發出 `AssumeRoleWithWebIdentity` API 請求。在此請求中的參數中，Amazon Cognito 包括以下內容。

1. 使用者的身分 ID。
2. 您的使用者想要擔任的 IAM 角色。
3. 加上內嵌工作階段政策的 `policy` 參數。
4. 一種`PolicyArns.member.N`參數，其值為AWS 受管政策，可在 Amazon 中授予其他許可 CloudWatch。

未經身分驗證的使用者可存取的服務

使用強化流程時，Amazon Cognito 會縮小套用於使用者工作階段的政策範圍，防止使用者使用下列清單以外的任何其他服務。對於子服務，只允許特定的動作。

類別	服務
分析	Amazon 數據 Firehose Amazon Managed Service for Apache Flink
應用程式整合	Amazon Simple Queue Service
擴增實境和虛擬實境	Amazon Sumerian[1]
商業應用程式	Amazon Mobile Analytics Amazon Simple Email Service
運算	AWS Lambda
密碼編譯和 PKI	AWS Key Management Service[1]
資料庫	Amazon DynamoDB Amazon SimpleDB
前端 Web 與行動裝置	AWS AppSync

類別	服務
	Amazon Location Service Amazon Simple Notification Service Amazon Pinpoint
遊戲開發	Amazon GameLift
物聯網 (IoT)	AWS IoT
機器學習	Amazon CodeWhisperer Amazon Comprehend Amazon Lex Amazon Machine Learning Amazon Personalize Amazon Polly Amazon Rekognition Amazon SageMaker [1] Amazon Textract[1] Amazon Transcribe Amazon Translate
管理與治理	Amazon CloudWatch Amazon CloudWatch 日誌
聯網與內容交付	Amazon API Gateway
安全、身分與合規	Amazon Cognito 使用者集區
儲存	Amazon Simple Storage Service

[1] 對於下表 AWS 服務 中的，內嵌政策會授與動作的子集。表格會顯示每個子集中的可用動作。

AWS 服務	未經身分驗證的強化流程使用者的最大許可

AWS Key Management Service	`Encrypt`
	`Decrypt`
	`ReEncrypt`
	`GenerateDataKey`

Amazon SageMaker	`InvokeEndpoint`
Amazon Textract	`DetectDocumentText` `AnalyzeDocument`
Amazon Sumerian	`View*`

若要授與此清單以 AWS 服務 外的存取權，請在您的身分識別集區中啟動基本 (傳統) 驗證流程。如果您的使用者在 AWS 服務 看到 `NotAuthorizedException` 錯誤，而指派給 IAM 角色或未經身分驗證的使用者的政策允許此類服務，請評估您是否可以從使用案例中移除該服務。如果不能，請切換至基本流程。

內嵌工作階段政策

內嵌工作階段原則會限制使用者的有效權限，無法存取下列清單中的任何 AWS 服務 外部權限。您還必須 AWS 服務 在套用至使用者 IAM 角色的政策中授予這些許可。使用者對於擔任角色工作階段的有效許可，是指派給其角色的政策及其工作階段政策的交集。如需詳細資訊，請參閱 AWS Identity and Access Management 使用者指南中的工作階段政策。

Amazon Cognito 會將下列內嵌政策新增至 AWS 區域 中使用者預設啟用的工作階段。

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

```
                "logs:*",
                "dynamodb:*",
                "kinesis:*",
                "mobileanalytics:*",
                "s3:*",
                "ses:*",
                "sns:*",
                "sqs:*",
                "lambda:*",
                "machinelearning:*",
                "execute-api:*",
                "iot:*",
                "gamelift:*",
                "scs:*",
                "cognito-identity:*",
                "cognito-idp:*",
                "lex:*",
                "polly:*",
                "comprehend:*",
                "translate:*",
                "transcribe:*",
                "rekognition:*",
                "mobiletargeting:*",
                "firehose:*",
                "appsync:*",
                "personalize:*",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "sagemaker:InvokeEndpoint",
                "cognito-sync:*",
                "sumerian:View*",
                "codewhisperer:*",
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument",
                "sdb:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
```

```
}
```

對於所有其他區域，內嵌範圍向下兼容規則會包含預設區域中列出的所有項目，但下列 Action 陳述式除外。

```
                "cognito-sync:*",
                "sumerian:View*",
                "codewhisperer:*",
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument",
                "sdb:*"
```

AWS 受管理工作階段原則

Amazon Cognito 也會使用 AWS 受管政策 AmazonCognitoUnAuthedIdentitiesSessionPolicy 將未經驗證使用者的許可範圍，限於增強型流程中未經驗證的使用者。您也必須在連接到未驗證 IAM 角色的政策中授予此許可。

AmazonCognitoUnAuthedIdentitiesSessionPolicy 受管政策具有下列許可。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "rum:PutRumEvents",
            "polly:*",
            "comprehend:*",
            "translate:*",
            "transcribe:*",
            "rekognition:*",
            "mobiletargeting:*",
            "firehose:*",
            "personalize:*",
            "sagemaker:InvokeEndpoint"
        ],
        "Resource": "*"
    }]
}
```

存取政策範例

在本節中，您可以找到 Amazon Cognito 存取政策範例，這些政策會授予您的使用者執行特定操作所需的最低許可。您可以盡可能使用政策變數，進一步限制特定身分 ID 的許可。例如，使用 ${cognito-identity.amazonaws.com:sub}。如需詳細資訊，請參閱 AWS Mobile 部落格的了解 Amazon Cognito 身分驗證第 3 部分：角色和政策。

> **Note**
>
> 基於安全最佳實務，政策應該只包含使用者執行任務所需的許可。這表示您應該盡可能嘗試一律限定個別身分才能存取物件。

授予身分對 Amazon S3 中的單一物件具有讀取存取權

下列存取政策會將讀取許可授予身分，以從指定的 S3 儲存貯體擷取單一物件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/assets/my_picture.jpg"]
    }
  ]
}
```

授予身分同時對 Amazon S3 中的身分特定路徑具有讀取和寫入存取權

下列存取政策將字首映射至 ${cognito-identity.amazonaws.com:sub} 變數，以授予讀取和寫入許可來存取 S3 儲存貯體中的特定字首 “folder”。

使用此政策，透過 ${cognito-identity.amazonaws.com:sub} 插入的身分 (例如 us-east-1:12345678-1234-1234-1234-123456790ab) 可取得、放入和列出 arn:aws:s3:::mybucket/us-east-1:12345678-1234-1234-1234-123456790ab 中的物件。不過，該身分不會獲授予 arn:aws:s3:::mybucket 中其他物件的存取權。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket"],
      "Condition": {"StringLike": {"s3:prefix": ["${cognito-
identity.amazonaws.com:sub}/*"]}}
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/${cognito-identity.amazonaws.com:sub}/*"]
    }
  ]
}
```

將 Amazon DynamoDB 精細定義存取權指派給身分

以下存取政策使用 Amazon Cognito 環境變數，提供對 DynamoDB 資源的精細定義存取控制。這些變數依身分 ID 授予對 DynamoDB 中的項目的存取權。如需詳細資訊，請參閱《Amazon DynamoDB 開發人員指南》中的使用 IAM 政策條件進行精細定義存取控制。

```
{
  "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:us-west-2:123456789012:table/MyTable"
      ],
```

```
      "Condition": {
        "ForAllValues:StringEquals": {
          "dynamodb:LeadingKeys":  ["${cognito-identity.amazonaws.com:sub}"]
        }
      }
    }
  ]
}
```

授予身分許可以叫用 Lambda 函數

下列存取政策授予身分許可以叫用 Lambda 函數。

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
             "Effect": "Allow",
             "Action": "lambda:InvokeFunction",
             "Resource": [
                 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
             ]
         }
     ]
 }
```

授予身分許可以發佈記錄到 Kinesis 資料串流

以下存取政策允許身分對任何 Kinesis Data Streams 使用 `PutRecord` 操作。它可以套用到需要將資料記錄新增至帳戶中所有串流的使用者。如需詳細資訊，請參閱《Amazon Kinesis Data Streams 開發人員指南》中的使用 IAM 控制對 Amazon Kinesis 資料串流資源的存取。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kinesis:PutRecord",
            "Resource": [
                "arn:aws:kinesis:us-east-1:111122223333:stream/stream1"
            ]
        }
    ]
```

```
}
```

授予身分存取 Amazon Cognito Sync 存放區中的資料

以下存取政策授予身分許可只能存取他們自己在 Amazon Cognito Sync 存放區中的資料。

```
{
  "Version": "2012-10-17",
  "Statement":[{
      "Effect":"Allow",
      "Action":"cognito-sync:*",
      "Resource":["arn:aws:cognito-sync:us-east-1:123456789012:identitypool/${cognito-
identity.amazonaws.com:aud}/identity/${cognito-identity.amazonaws.com:sub}/*"]
      }]
  }
```

角色信任和許可

這些角色的不同之處，在於其信任關係。以下為未驗證的角色的範例信任政策：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-corner-
cafe-123456790ab"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "unauthenticated"
        }
      }
    }
  ]
}
```

此政策授予來自 cognito-identity.amazonaws.com (OpenID Connect 權杖的發行者) 的聯合身分使用者許可以擔任這個角色。此外，政策會限制權杖的 aud(在此案例中為身分集區 ID) 來配合身分集區。最後，政策會指定權杖 (由 Amazon Cognito GetOpenIdToken API 操作發出) 之多值 amr 宣告的其中一個陣列成員包含值 unauthenticated。

當 Amazon Cognito 建立權杖時，會將權杖的 amr 設為 unauthenticated 或 authenticated。如果 amr 是 authenticated，該權杖包括身分驗證期間使用的任何供應商。這表示，您只要將 amr 條件變更如下，即可建立只信任透過 Facebook 登入之使用者的角色：

```
"ForAnyValue:StringLike": {
  "cognito-identity.amazonaws.com:amr": "graph.facebook.com"
}
```

在變更角色的信任關係時，或是嘗試跨身分集區使用角色時，請小心謹慎。如果未將角色正確設定為信任您的身分集區，STS 結果會產生如下的例外狀況：

```
AccessDenied -- Not authorized to perform sts:AssumeRoleWithWebIdentity
```

如果您看到此訊息，請檢查您的身分集區和身分驗證類型是否有適當的角色。

Amazon Cognito 身分集區的安全性最佳實務

Amazon Cognito 身分集區為您的應用程式提供臨時 AWS 登入資料。 AWS 帳戶 通常包含應用程式使用者所需的資源，以及私有後端資源。構成 AWS 登入資料的 IAM 角色和政策可以授予這些資源的存取權。

身分集區組態的主要最佳作法是確保您的應用程式可以在沒有過多或意外權限的情況下完成工作。若要防範安全性錯誤設定，請在啟動您要發行至生產環境的每個應用程式之前檢閱這些建議。

主題

- IAM 組態最佳做法
- 身分識別集區組態最佳做

IAM 組態最佳做法

當來賓或經過驗證的使用者在您的應用程式中啟動需要身分集區登入資料的工作階段時，您的應用程式會擷取 IAM 角色的臨時 AWS 登入資料。認證可能是預設角色、身分集區設定中規則所選擇的角色，或應用程式所選擇的自訂角色。透過指派給每個角色的權限，您的使用者可以存取您的 AWS 資源。

如需有關一般 IAM 最佳做法的詳細資訊，請參閱 AWS Identity and Access Management 使用者指南中的 IAM 最佳實務。

在 IAM 角色中使用信任政策條件

IAM 要求身分集區的角色至少具有一個信任政策條件。例如，此條件可以將角色的範圍設定為僅經過驗證的使用者。 AWS STS 還要求跨帳戶基本身份驗證請求具有兩種特定條件：`cognito-identity.amazonaws.com:aud`和`cognito-identity.amazonaws.com:amr`。最佳做法是在信任身分識別集區服務主體的所有 IAM 角色中套用這兩個條件`cognito-identity.amazonaws.com`。

- `cognito-identity.amazonaws.com:aud`：身分集區權杖中的 aud 宣告必須與受信任的身分集區 ID 相符。
- `cognito-identity.amazonaws.com:amr`：身份池令牌中的 amr 聲明必須經過身份驗證或未經身份驗證。在這種情況下，您可以僅保留對未經驗證的來賓角色的存取權，或僅保留對已驗證的使用者的存取權。例如，您可以進一步調整此條件的值，將角色限制為來自特定提供者的使用者`graph.facebook.com`。

下列範例角色信任原則會在下列情況下授與角色的存取權：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-east-1:a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "authenticated"
        }
      }
    }
  ]
```

```
}
```

與身分集區相關的元素

- `"Federated": "cognito-identity.amazonaws.com"`：使用者必須來自身分集區。
- `"cognito-identity.amazonaws.com:aud": "us-east-1:a1b2c3d4-5678-90ab-cdef-example11111"`：使用者必須來自特定的身分集區`us-east-1:a1b2c3d4-5678-90ab-cdef-example11111`。
- `"cognito-identity.amazonaws.com:amr": "authenticated"`：使用者必須經過驗證。來賓使用者無法擔任該角色。

套用最小權限

當您使用 IAM 政策設定許可以進行驗證的存取權或訪客存取時，只授予執行特定工作所需的特定權限或最低權限許可。下列 IAM 政策範例套用至角色時，會授予對 Amazon S3 儲存貯體中單一映像檔案的唯讀存取權。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/assets/my_picture.jpg"]
    }
  ]
}
```

身分識別集區組態最佳做

身分集區具有用於產生 AWS 認證的彈性選項。當您的應用程序可以使用最安全的方法時，請不要使用設計快捷方式。

瞭解訪客存取的影響

未經驗證的訪客存取允許使用者在登入 AWS 帳戶 之前從您擷取資料。任何知道您身分集區 ID 的人都可以要求未驗證的認證。您的身分集區 ID 並非機密資訊。當您啟用來賓存取時，所有人都可以使用您授予未驗證工作階段的 AWS 權限。

最佳做法是停用來賓存取權，並僅在使用者驗證後擷取所需的資源。如果您的應用程式需要在登入前存取資源，請採取下列預防措施。

- 熟悉未驗證角色的自動限制。
- 監控並調整未驗證 IAM 角色的許可，以符合應用程式的特定需求。
- 授予對特定資源的存取權。
- 保護預設未驗證 IAM 角色的信任政策。
- 只有當您確信要將 IAM 角色中的許可授予網際網路上的任何人時，才啟用訪客存取權限。

預設使用增強型驗證

透過基本 (傳統) 身份驗證，Amazon Cognito 會將 IAM 角色的選擇委派給您的應用程式。相反地，增強型流程會使用身分集區中的集中式邏輯來判斷 IAM 角色。它還透過設定 IAM 許可上限的範圍政策，為未驗證身分提供額外的安全性。增強的流程是最安全的選擇，以最低的開發人員努力。若要進一步瞭解這些選項，請參閱身分集區 (聯合身分) 驗證流程。

基本流程可以公開進入角色選擇和組件 AWS STS API 認證請求的用戶端邏輯。增強的流程會隱藏身分集區自動化背後的邏輯和 assume-role 要求。

設定基本身份驗證時，請將 IAM 最佳實務套用至 IAM 角色及其許可。

安全地使用開發人員

開發人員驗證身分是伺服器端應用程式身分集區的一項功能。身分集區進行開發人員身份驗證所需的唯一證據是身份池開發人員的 AWS 憑據。身分集區不會對您在此驗證流程中顯示的開發人員提供者識別碼的有效性強制執行任何限制。

最佳做法是，只有在下列條件下才實作開發人員提供者：

- 若要建立使用開發人員驗證認證的責任，請設計您的開發人員提供者名稱和識別碼，以指出驗證來源。例如："Logins" : {"MyCorp provider" : "*[provider application ID]*"}。
- 避免使用長期使用者憑證。將您的伺服器端用戶端設定為使用服務連結角色 (例如 EC2 執行個體設定檔和 Lambda 執行角色)
- 避免在同一個身分集區中混用內部和外部信任來源。在不同的身分集區中新增您的開發人員提供者和單一登入 (SSO) 提供者。

使用屬性進行存取控制

存取控制的屬性基於屬性型存取控制 (ABAC) 的 Amazon Cognito 身份集區實施。您可以使用 IAM 政策，根據使用者屬性，控制透過 Amazon Cognito 身分集區存取 AWS 資源。這些屬性可以向社交和公司身分提供者取得。您可以將供應商存取權杖和 ID 權杖或 SAML 聲明中的屬性，映射至可在 IAM 許可政策中參照的標籤。

您可以選擇預設映射 (default mappings)，或在 Amazon Cognito 身分集區中建立自己的自訂映射 (custom mappings)。預設對應可讓您根據固定的使用者屬性集，撰寫 IAM 政策。自訂對應可讓您選取 IAM 許可政策中參照的自訂使用者屬性集。Amazon Cognito 主控台的 Attribute names (屬性名稱) 映射至 Tag key for principal (委託人的標籤索引鍵)，這些標籤是 IAM 許可政策中參照的標籤。

舉例來說，假設您有具免費和付費會員資格的媒體串流服務。您可以將媒體檔案存放在 Amazon S3 中，並加上免費或付費標籤。您可以使用存取控制屬性，根據使用者成員資格等級 (屬於使用者描述檔)，允許存取免費和付費內容。您可以將成員資格屬性對應至標籤索引鍵，將委託人傳遞至 IAM 許可政策。如此一來，您就可以建立單一許可政策，並根據成員資格等級值和內容檔案的標籤，有條件地允許存取付費內容。

主題

使用屬性控制存取權限有多個好處：

- 使用存取控制屬性時，可提升管理許可的效率。您可以建立使用使用者屬性的基本許可政策，而不是為不同任務函數建立多個政策。
- 每當為應用程式新增或移除資源或使用者時，您就不必更新政策。許可政策只會將存取權限授與具相符使用者屬性的使用者。例如，您可能需要根據使用者的職稱，控制特定 S3 儲存貯體的存取權限。在這種情況下，您可以建立許可政策，只允許已定義職稱內的使用者存取這些檔案。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。
- 屬性可以當作委託人標籤，傳遞至根據這些屬性值允許或拒絕許可的政策。

透過 Amazon Cognito 身分集區使用屬性進行存取控制

在使用屬性進行存取控制之前，請確認您符合下列先決條件：

- 一個 AWS 帳戶
- 使用者集區
- 身分集區
- 設定 SDK
- 整合的身分提供者
- 憑證

若要使用存取控制屬性，設定為來源資料集的 宣告 值，需設定為您所選的 標籤金鑰。Amazon Cognito 會將標籤金鑰和值套用至使用者的工作階段。您的 IAM 政策可以根據 `${aws:PrincipalTag/tagkey}` 條件評估使用者的存取權限。IAM 會根據政策評估使用者的標籤值。

您必須準備好 IAM 角色，這些角色的憑證將會傳遞給您的使用者。這些角色的信任政策必須允許 Amazon Cognito 為您的使用者擔任該角色。對於存取控制的屬性，您也必須允許 Amazon Cognito 將主體標籤套用至使用者的臨時工作階段。使用 AssumeRoleWithWebIdentity 動作授予擔任角色的許可。使用僅許可動作 `sts:TagSession` 授予標記使用者工作階段的許可。如需詳細資訊，請參閱《AWS Identity and Access Management 使用者指南》中的在 AWS Security Token Service 中傳遞工作階段標記。如需對 Amazon Cognito 服務主體 `cognito-identity.amazonaws.com` 授予 `sts:AssumeRoleWithWebIdentity` 和 `sts:TagSession` 許可的信任策略範例，請參閱 使用屬性進行存取控制的政策範例。

若要在 Amazon Cognito 主控台中設定存取控制屬性

1. 登入 Amazon Cognito 主控台，然後選取 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。
3. 找到 身分提供者。選擇您要編輯的身分提供者。如果您要新增一個新的 IdP，選取 新增身分供應商。
4. 若要變更 Amazon Cognito 向使用者驗證此提供者，發布憑證時指派的委託人標籤，請在 存取控制屬性 中選擇 編輯。
    a. 若不套用主要索引標籤，請選擇 非作用中。
    b. 若要根據 `sub` 和 `aud` 宣告套用主要索引標籤，請選擇 使用預設對應。

c. 若要建立您自己的自訂屬性結構描述至主要索引標籤，請選擇 使用自訂對應。然後，輸入您要從每個 宣告 中獲取的 標籤金鑰，顯示於索引標籤當中。

5. 選取 Save Changes (儲存變更)。

使用屬性進行存取控制的政策範例

考慮這個情況，公司法務部門員工須列出屬於部門的儲存貯體中所有檔案，並以安全等級加以分類。假設此員工向身分提供者取得的權杖包含下列宣告。

宣告

```
{ .
  .
"sub" : "57e7b692-4f66-480d-98b8-45a6729b4c88",
"department" : "legal",
"clearance" : "confidential",
 .
 .
}
```

這些屬性可以對應至標籤，並在 IAM 許可政策中參照為委託人標籤。您現在可以變更身分提供者端的使用者描述檔，管理存取權限。或者，您可以使用名稱或標籤變更資源端的屬性，而不變更政策本身。

下列許可政策會執行兩項作業：

- 允許列出存取所有以符合使用者部門名稱的字首結尾的 S3 儲存貯體。
- 只要檔案的許可標籤符合使用者的許可屬性，就允許讀取存取這些儲存貯體中的檔案。

許可政策

```
{
 "Version": "2012-10-17",
 "Statement": [
     {
         "Effect": "Allow",
         "Action": "s3:List*",
         "Resource": "arn:aws:s3:::*-${aws:PrincipalTag/department}"
```

```
        },
        {
            "Effect": "Allow",
            "Action": "s3:GetObject*",
            "Resource": "arn:aws:s3:::*-${aws:PrincipalTag/department}/*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/clearance": "${aws:PrincipalTag/clearance}"
                }
            }
        }
    ]
}
```

信任政策決定誰可擔任此角色。信任關係政策允許使用 sts:AssumeRoleWithWebIdentity 和 sts:TagSession 允許存取。這會新增條件，將政策限制於您建立的身分池，並確保適用於已驗證的角色。

信任政策

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRoleWithWebIdentity",
        "sts:TagSession"
      ],
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "IDENTITY-POOL-ID"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "authenticated"
        }
      }
```

```
    }
  ]
}
```

關閉以存取控制屬性 (主控台)

按照此程序停用以存取控制屬性。

若要在主控台中停止用存取控制屬性

1. 登入 Amazon Cognito 主控台，然後選取 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。
3. 找到 身分提供者。選擇您要編輯的身分提供者。
4. 在 存取控制屬性 中選擇 編輯。
5. 若不套用主要索引標籤，請選擇 非作用中。
6. 選取 Save Changes (儲存變更)。

預設供應商對應

下表為 Amazon Cognito 支援的身分驗證供應商預設對應資訊。

供應商	權杖類型	委託人標籤值	範例
Amazon Cognito 使用者集區	ID 權杖	aud(client ID) 和 sub(user ID)	"6jk8ltokc7ac9es6jrtg9q572f"、"57e7b692-4f66-480d-98b8-45a6729b4c88"
Facebook	存取權杖	aud(app_id)、sub(user_id)	"492844718097981"、"112177216992379"
Google	ID 權杖	aud(client ID) 和 sub(user ID)	"620493171733-eebk7c0hcp5lj3e1tlqp1gntt3k0rncv.apps.googleuserc

供應商	權杖類型	委託人標籤值	範例
			ontent.com"、"10922 0063452404746097"
SAML	聲明	"http://schemas.xm lsoap.org/ws/2005/ 05/identity/claims/ nameidentifier"、" http://schemas.xml soap.org/ws/2005/05/ identity/claims/name"	"auth0|5e28d196f8f 55a0eaaa9 5de3"、"us er123@gmail.com"
Apple	ID 權杖	aud(client ID) 和 sub (user ID)	"com.amazonaws.ec2 -54-80-172-243.com pute-1.client"、"00 1968.a6ca34e9c1e74 2458a26cf8005854be 9.0733"
Amazon	存取權杖	aud (Client ID on Amzn Dev Ac)、user_ id(user ID)	"amzn1.application -oa2-client.9d70d9 382d34461 08aaee3dd763a0fa6" 、"amzn1.account.AG HNIFJQMFS BG3G6XCPV B35ORQAA"
標準 OIDC 供應商	ID 權杖和存取權杖	aud (as client_id)、sub (as user ID)	"620493171733- eebk7c0hcp5lj 3e1tlqp1gntt3k0rnc v.apps.googleuserc ontent.com"、"10922 0063452404746097"

供應商	權杖類型	委託人標籤值	範例
Twitter	存取權杖	aud (app ID; app Secret)、sub (user ID)	"DfwifTtKEX1FiIBRnOTIR0CFK; Xgj5xb8xIrIVCPjXgLIdkW7fXmwcJJrFvnoK9gwZkLexo1y5z1"、"1269003884292222976"
DevAuth	Map	不適用	"tag1"、"tag2"

> **Note**
>
> Tag Key for Principal (委託人的標籤索引鍵) 和 Attribute (屬性) 名稱會自動填入預設屬性對應選項。您無法變更預設對應。

使用以角色為基礎的存取控制

Amazon Cognito 身分集區會為已驗證的使用者指派一組臨時、有限權限的登入資料，以存取您 AWS 的資源。每個使用者的許可都是透過您建立的 IAM 角色來控制。您也可以定義規則，以依據使用者 ID 權杖中的宣告，為每個使用者選擇角色。您可以為已驗證的使用者定義預設角色。您也可以為未驗證的訪客使用者，另外定義有限許可的 IAM 角色。

建立角色以進行角色映射

為每個角色新增適當的信任政策是很重要的，這樣才能讓 Amazon Cognito 針對身分集區中的已驗證使用者來採用信任政策。以下是這種信任政策的範例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-corner-
cafe-123456790ab"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "authenticated"
        }
      }
    }
  ]
}
```

此政策可讓來自 cognito-identity.amazonaws.com(OpenID Connect 權杖的發行者) 的聯合身分使用者擔任這個角色。此外，政策會限制權杖的 aud(在此案例中為身分集區 ID) 來配合身分集區。最後，政策會指定權杖 (由 Amazon Cognito GetOpenIdToken API 動作發出) 之多值 amr 宣告的其中一個陣列成員包含值 authenticated。

授予傳送角色許可

若要讓使用者設定角色，使擁有的許可超過使用者對身分集區現有的許可，您可以授與該使用者 iam:PassRole 許可，以將角色傳遞至 set-identity-pool-roles API。例如，如果使用者無法寫入 Amazon S3，但是使用者在身分集區上設定的 IAM 角色會授予對 Amazon S3 的寫入許可，則除非將 iam:PassRole 許可授予至此角色，否則該使用者無法設定此角色。下列政策範例顯示如何允許 iam:PassRole 許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myS3WriteAccessRole"
            ]
        }
    ]
```

```
}
```

在此政策範例中，已將 iam:PassRole 許可授與 myS3WriteAccessRole 角色。該角色是使用角色的 Amazon Resource Name (ARN) 來指定。您也必須將此政策附加至使用者。如需詳細資訊，請參閱處理受管政策的相關文章。

> **Note**
>
> Lambda 函數使用的是資源型政策，即是政策直接連接至 Lambda 函數本身。在建立叫用 Lambda 函數的規則時，並不會傳送角色，因此建立該規則的使用者不需擁有 iam:PassRole 許可。如需 Lambda 函數授權的詳細資訊，請參閱管理許可：使用 Lambda 函數政策。

使用權杖來指派角色給使用者

針對透過 Amazon Cognito 使用者集區登入的使用者，可以在由使用者集區指派的 ID 權杖中傳遞角色。這些角色會出現在 ID 權杖的下列宣告中：

- cognito:preferred_role宣告是角色 ARN。
- cognito:roles宣告以逗號分隔的字串，其中包含一組允許的角色 ARN。

宣告設定如下：

- cognito:preferred_role宣告設定為具有最高 (最小) Precedence 值的群組中的角色。如果只有一個允許的角色，則 cognito:preferred_role會設定為該角色。如果有多個角色，而沒有任何角色具有最高優先順序，則不會設定此宣告。
- 如果至少有一個角色，就會設定 cognito:roles宣告。

使用權杖來指派角色時，如果有多個角色可以指派給使用者，Amazon Cognito 身分集區 (聯合身分) 會依下列方式來選擇角色：

- 如果已設定GetCredentialsForIdentityCustomRoleArn參數，且該參數符合cognito:roles宣告中的角色，請使用該參數。如果此參數不符合 cognito:roles中的角色，則拒絕存取。
- 如果已設定 cognito:preferred_role宣告，則加以使用。
- 如果未設定cognito:preferred_role宣告，cognito:roles宣告會設定且CustomRoleArn未在呼叫中指定GetCredentialsForIdentity，則會使用主控台

或AmbiguousRoleResolution欄位 (在 SetIdentityPoolRolesAPI RoleMappings 參數中) 中的 [角色解析] 設定來決定要指派的角色。

使用以規則為基礎的映射來指派角色給使用者

規則可讓您將宣告從身分提供者權杖對應至 IAM 角色。

每個規則都會指定權杖宣告 (如來自 Amazon Cognito 使用者集區之 ID 權杖的使用者屬性)、相符類型、值及 IAM 角色。相符類型可以是 Equals、NotEqual、StartsWith 或 Contains。如果使用者有符合宣告的值，則當使用者取得登入資料時，即可擔任該角色。例如，您可以建立一個規則，將特定 IAM 角色指派給 custom:dept 自訂屬性值為 Sales 的使用者。

> **Note**
>
> 在規則設定中，自訂屬性需要 custom: 字首來將其與標準屬性區分。

規則會依序評估，並且會使用第一個相符規則的 IAM 角色，除非是指定 CustomRoleArn 來覆寫該順序。如需 Amazon Cognito 使用者集區中的使用者屬性詳細資訊，請參閱使用者集區屬性。

您可以在身分集區 (聯合身分) 主控台中，針對身分驗證供應商設定多個規則。規則會依序套用。您可以曳規則來變更順序。第一個相符的規則優先。如果相符類型為 NotEqual，且宣告不存在，則不會評估規則。如果沒有規則相符，則會將 角色解析 設定套用至 使用預設的已驗證角色 或 拒絕。

在 API 和 CLI 中，您可以指定RoleMapping類型欄位中沒有規則符合時要指派的角色，該AmbiguousRoleResolution欄位是在 SetIdentityPoolRolesAPI RoleMappings 參數中指定的。

您可以在 AWS CLI 或 API 中使用類型的欄位，為 OpenID Connect (OIDC) 和 SAML 身分識別提供者設定以規則為基礎的RulesConfiguration對應。RoleMapping您可以在 SetIdentityPoolRolesAPI 的RoleMappings參數中指定此欄位。目 AWS Management Console 前不允許您新增 OIDC 或 SAML 提供者的規則。

例如，下列 AWS CLI 命令會新增一個規則，將角色arn:aws:iam::123456789012:role/Sacramento_team_S3_admin指派給薩克拉門托位置中經過 OIDC IdP 驗證的使用者：arn:aws:iam::123456789012:oidc-provider/myOIDCIdP

```
aws cognito-identity set-identity-pool-roles --region us-east-1 --cli-input-json
 file://role-mapping.json
```

role-mapping.json 的內容：

```
{
    "IdentityPoolId": "us-east-1:12345678-corner-cafe-123456790ab",
    "Roles": {
        "authenticated": "arn:aws:iam::123456789012:role/myS3WriteAccessRole",
        "unauthenticated": "arn:aws:iam::123456789012:role/myS3ReadAccessRole"
    },
    "RoleMappings": {
        "arn:aws:iam::123456789012:oidc-provider/myOIDCIdP": {
            "Type": "Rules",
            "AmbiguousRoleResolution": "AuthenticatedRole",
            "RulesConfiguration": {
                "Rules": [
                    {
                        "Claim": "locale",
                        "MatchType": "Equals",
                        "Value": "Sacramento",
                        "RoleARN": "arn:aws:iam::123456789012:role/
Sacramento_team_S3_admin"
                    }
                ]
            }
        }
    }
}
```

針對每個使用者集區，或是您為身分集區設定的其他身分驗證供應商，最多可以建立 25 個規則。此限制不可調整。如需詳細資訊，請參閱 Amazon Cognito 中的配額。

要用在以規則為基礎之映射中的宣告

Amazon Cognito

Amazon Cognito ID 權杖是以 JSON Web 權杖 (JWT) 表示。權杖中包含已驗證使用者身分的相關宣告，例如 name、family_name 和 phone_number。如需標準宣告的詳細資訊，請參閱 OpenID Connect 規格。除了標準宣告，Amazon Cognito 還有下列專屬的其他宣告：

- cognito:groups
- cognito:roles
- cognito:preferred_role

Amazon

下列宣告及這些宣告的可能值可用於 Login with Amazon：

- `iss`：www.amazon.com
- `aud`：應用程式 ID
- `sub`：來自 Login with Amazon 權杖的 `sub`

Facebook

下列宣告及這些宣告的可能值可用於 Facebook：

- `iss`：graph.facebook.com
- `aud`：應用程式 ID
- `sub`：來自 Facebook 權杖的 `sub`

Google

Google 權杖包含來自 OpenID Connect 規格的標準宣告。OpenID 權杖中的所有宣告都可用於以規則為基礎的對應。請參閱 Google 的 OpenID Connect 網站，以了解來自 Google 權杖的可用宣告。

Apple

Apple 權杖包含來自 OpenID Connect 規格的標準宣告。請參閱 Apple 文件中的 Authenticating Users with Sign in with Apple，以進一步了解來自 Apple 權杖的可用宣告。Apple 的權杖並非一定會包含 `email`。

OpenID

OpenID 權杖中的所有要求都適用於以規則為基礎的對應。如需標準宣告的詳細資訊，請參閱 OpenID Connect 規格。請參閱您的 OpenID 供應商說明文件，以了解任何可用的其他宣告。

SAML

宣告是從收到的 SAML 聲明剖析而來。SAML 聲明中所有可用的宣告，都可以用在以規則為基礎的對應中。

以角色為基礎的存取控制最佳實務

Important

如果最終使用者可以修改您對應至角色的宣告，則任何最終使用者都可以擔任您的角色，並依情況設定政策。請只將無法由最終使用者直接設定的宣告對應至具有更高許可的角色。在 Amazon Cognito 使用者集區中，您可以依每個應用程式來設定每個使用者屬性的讀取和寫入許可。

Important

如果您為 Amazon Cognito 使用者集區中的群組設定角色，將會透過使用者的 ID 權杖來剖析這些角色。若要使用這些角色，身分集區的已驗證角色選項必須設定 Choose role from token (從權杖選擇角色)。
您可以使用控制台中的角色解析設置和 SetIdentityPoolRolesAPI 的`RoleMappings`參數來指定無法從令牌確定正確角色時的默認行為。

取得憑證

您可以使用 Amazon Cognito 向應用程式提供臨時、有限權限的登入資料，以便您的使用者可以存取 AWS 資源。本節說明如何取得憑證，以及如何從身分集區擷取 Amazon Cognito 身分。

Amazon Cognito 可支援已驗證和未驗證的身分。未經授權的使用者的身分未經驗證，因此這個角色適用於您應用程式的訪客使用者，或使用者身分是否已驗證並不重要的情況。已驗證使用者透過第三方身分提供者或驗證其身分的使用者集區來登入應用程式。請務必適當地限制資源許可範圍，以避免從未經授權的使用者授與資源的存取權。

Amazon Cognito 身分不是憑證。它們會使用 AWS Security Token Service (AWS STS) 中的 Web 身分同盟支援來交換憑證。若要您的應用程式使用者取得 AWS 憑證，建議的方法是使用 `AWS.CognitoIdentityCredentials`。然後，會使用將認證物件中的識別交換為認證 AWS STS。

Note

如果您在 2015 年 2 月之前建立身分池，您必須重新建立您的角色與身分池的關聯，以便使用 `AWS.CognitoIdentityCredentials` 建構函式，而無需將角色做為參數。若要執行此作

業，請開啟 Amazon Cognito 主控台，選擇 Manage identity pools (管理身分集區)，選取您的身分集區，然後選擇 Edit identity Pool (編輯身分集區)，指定您的已驗證和未驗證角色，然後儲存變更。

Web 身分憑證提供者是 AWS SDK 中預設憑證提供者鏈結的一部分。若要在 AWS SDK 的本機`config`檔案中設定您的身分識別集區權杖 AWS CLI，或新增設定`web_identity_token_file`檔項目。請參閱 AWS SDK 和工具參考指南中的假設角色憑證提供者。

若要了解有關如何在 SDK 中填入 Web 身分憑證的更多信息，請參閱 SDK 開發人員指南。若要取得最佳結果，請使用內建的身分識別集區整合開始您的專案 AWS Amplify。

AWS 使用身分集區取得及設定認證的 SDK 資源

- 《Amplify 開發人員中心》中的身分池聯合 (Android)
- 《Amplify 開發人員中心》中的身分池聯合 (iOS)
- 在 AWS SDK for JavaScript 開發人員指南中使用 Amazon Cognito 身分驗證使用者
- AWS SDK for .NET 開發人員指南中的 Amazon Cognito 登入資料供應商
- 在 AWS SDK for Go 開發人員指南中以程式設計方式
- 在 AWS SDK for Java 2.x 開發人員指南中的程式碼中提供臨時認證
- assumeRoleWithWebIdentityCredentialProvider AWS SDK for PHP 開發人員指南中的提供者
- 在 AWS SDK for Python (Boto3) 文件中使用 Web 身分提供者指派角色
- 在 適用於 Rust 的 AWS SDK 開發人員指南中指定您的憑證和預設區域

下列各節提供部分舊版 AWS SDK 中的範例程式碼。

Android

您可以使用 Amazon Cognito 向應用程式提供臨時、有限權限的登入資料，以便您的使用者可以存取 AWS 資源。Amazon Cognito 可支援已驗證和未驗證的身分。要為您的應用提供 AWS 憑據，請按照以下步驟操作。

若要在 Android 應用程式中使用亞馬遜認可身分集區，請進行設 AWS Amplify定。如需詳細資訊，請參閱《Amplify 開發人員中心》中的身分驗證。

擷取 Amazon Cognito 身分

如果您要允許未驗證的使用者，可以立即為您的最終使用者擷取唯一 Amazon Cognito 識別符 (身分 ID)。如果您要驗證使用者，可以在登入資料供應商中設定登入權杖之後，擷取身分 ID：

```
String identityId = credentialsProvider.getIdentityId();
Log.d("LogTag", "my ID is " + identityId);
```

> **Note**
>
> 請不要在應用程式的主要執行緒中呼叫 `getIdentityId()`、`refresh()` 或 `getCredentials()`。從 Android 3.0 (API 級別 11) 開始，NetworkOnMainThreadException如果您在主應用程序線程上執行網絡 I/O，您的應用程序將自動失敗並拋出一個。您需要使用 `AsyncTask` 將您的程式碼移到背景執行緒。如需詳細資訊，請參閱 Android documentation。您也可以呼叫 `getCachedIdentityId()`來擷取 ID，但前提是已在本機快取過 ID。否則，此方法會傳回空值。

iOS - Objective-C

您可以使用 Amazon Cognito 向應用程式提供臨時、有限權限的登入資料，以便您的使用者可以存取 AWS 資源。Amazon Cognito 身分集區可支援已驗證和未驗證的身分。若要為您的應用程式提供 AWS 認證，請完成以下步驟。

若要在 iOS 應用程式中使用 Amazon Cognito 身分識別集區，請進行設 AWS Amplify定。如需詳細資訊，請參閱《Amplify 開發人員中心》中的 Swift 身分驗證和 Flutter 身分驗證。

擷取 Amazon Cognito 身分

如果您要允許未驗證的使用者，可以立即為您的最終使用者擷取唯一 Amazon Cognito 識別符 (身分 ID)，或者如果您要驗證使用者，可以在憑證供應商中設定登入權杖之後，擷取身分 ID：

```
// Retrieve your Amazon Cognito ID
[[credentialsProvider getIdentityId] continueWithBlock:^id(AWSTask *task) {
    if (task.error) {
        NSLog(@"Error: %@", task.error);
    }
    else {
        // the task result will contain the identity id
        NSString *cognitoId = task.result;
    }
    return nil;
```

```
}];
```

> **Note**
>
> `getIdentityId` 是非同步呼叫。如果身分 ID 已設定在您的供應商中，您可以呼叫 `credentialsProvider.identityId`來擷取該身分 (已在本機快取過)。然而，如果身分 ID 尚未設定在您的供應商中，則呼叫 `credentialsProvider.identityId`會傳回 `nil`。如需詳細資訊，請參閱 Amplify iOS SDK 參考。

iOS - Swift

您可以使用 Amazon Cognito 向應用程式交付有限權限的臨時登入資料，以便您的使用者可以存取 AWS 資源。Amazon Cognito 可支援已驗證和未驗證的身分。要為您的應用提供 AWS 憑據，請按照以下步驟操作。

若要在 iOS 應用程式中使用 Amazon Cognito 身分識別集區，請進行設 AWS Amplify定。如需詳細資訊，請參閱《Amplify 開發人員中心》中的 Swift 身分驗證。

擷取 Amazon Cognito 身分

如果您要允許未驗證的使用者，可以立即為您的最終使用者擷取唯一 Amazon Cognito 識別符 (身分 ID)，或者如果您要驗證使用者，可以在憑證供應商中設定登入權杖之後，擷取身分 ID：

```
// Retrieve your Amazon Cognito ID
credentialsProvider.getIdentityId().continueWith(block: { (task) -> AnyObject? in
    if (task.error != nil) {
        print("Error: " + task.error!.localizedDescription)
    }
    else {
        // the task result will contain the identity id
        let cognitoId = task.result!
        print("Cognito id: \(cognitoId)")
    }
    return task;
})
```

> **Note**
>
> `getIdentityId` 是非同步呼叫。如果身分 ID 已設定在您的供應商中，您可以呼叫 `credentialsProvider.identityId`來擷取該身分 (已在本機快取過)。然而，如果身分 ID

尚未設定在您的供應商中，則呼叫 `credentialsProvider.identityId`會傳回 `nil`。如需詳細資訊，請參閱 Amplify iOS SDK 參考。

JavaScript

如果您尚未建立身分集區，請在 Amazon Cognito 主控台中建立身分集區，然後再使用 `AWS.CognitoIdentityCredentials`。

在您以您的身分提供者設定身分集區之後，您可以使用 `AWS.CognitoIdentityCredentials` 來驗證使用者。若要設定您的應用程式登入資料使用 `AWS.CognitoIdentityCredentials`，請將 `credentials` 屬性設定為 `AWS.Config` 或每個服務的組態。以下範例使用 `AWS.Config`：

```
// Set the region where your identity pool exists (us-east-1, eu-west-1)
AWS.config.region = 'us-east-1';

// Configure the credentials provider to use your identity pool
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'IDENTITY_POOL_ID',
    Logins: { // optional tokens, used for authenticated login
        'graph.facebook.com': 'FBTOKEN',
        'www.amazon.com': 'AMAZONTOKEN',
        'accounts.google.com': 'GOOGLETOKEN',
        'appleid.apple.com': 'APPLETOKEN'
    }
});

// Make the call to obtain credentials
AWS.config.credentials.get(function(){

    // Credentials will be available when this function is called.
    var accessKeyId = AWS.config.credentials.accessKeyId;
    var secretAccessKey = AWS.config.credentials.secretAccessKey;
    var sessionToken = AWS.config.credentials.sessionToken;

});
```

選用的 `Logins` 屬性是身分提供者名稱與這些身分提供者的身分權杖的對應。您從身分提供者取得權杖的方式，取決於您使用的供應商。例如，如果 Facebook 是您的身分提供者之一，您可以使用 `FB.login`Facebook 開發套件的 函數來取得身分提供者權杖：

```
FB.login(function (response) {
    if (response.authResponse) { // logged in
        AWS.config.credentials = new AWS.CognitoIdentityCredentials({
          IdentityPoolId: 'us-east-1:1699ebc0-7900-4099-b910-2df94f52a030',
          Logins: {
            'graph.facebook.com': response.authResponse.accessToken
          }
        });

        console.log('You are now logged in.');
    } else {
        console.log('There was a problem logging you in.');
    }
});
```

擷取 Amazon Cognito 身分

如果您要允許未驗證的使用者，可以立即為您的最終使用者擷取唯一 Amazon Cognito 識別符 (身分 ID)，或者如果您要驗證使用者，可以在憑證供應商中設定登入權杖之後，擷取身分 ID：

```
var identityId = AWS.config.credentials.identityId;
```

Unity

您可以使用 Amazon Cognito 向應用程式提供臨時、有限權限的登入資料，以便您的使用者可以存取 AWS 資源。Amazon Cognito 可支援已驗證和未驗證的身分。要為您的應用提供 AWS 憑據，請按照以下步驟操作。

AWS SDK for Unity 現在是 AWS SDK for .NET 的一部分。若要在中開始使用 Amazon Cognito AWS SDK for .NET，請參閱開 AWS SDK for .NET 發人員指南中的 Amazon Cognito 登入資料供應商。或者，請參閱 Amplify 開發人員中心以 AWS Amplify取得建置應用程式的選項。

擷取 Amazon Cognito 身分

如果您要允許未驗證的使用者，可以立即為您的最終使用者擷取唯一 Amazon Cognito 識別符 (身分 ID)，或者如果您要驗證使用者，可以在憑證供應商中設定登入權杖之後，擷取身分 ID：

```
credentials.GetIdentityIdAsync(delegate(AmazonCognitoIdentityResult<string> result) {
    if (result.Exception != null) {
        //Exception!
    }
```

```
    string identityId = result.Response;
});
```

Xamarin

您可以使用 Amazon Cognito 向應用程式交付有限權限的臨時登入資料，以便您的使用者可以存取 AWS 資源。Amazon Cognito 可支援已驗證和未驗證的身分。要為您的應用提供 AWS 憑據，請按照以下步驟操作。

AWS SDK for Xamarin 現在是 AWS SDK for .NET 的一部分。若要在中開始使用 Amazon Cognito AWS SDK for .NET，請參閱開 AWS SDK for .NET 發人員指南中的 Amazon Cognito 登入資料供應商。或者，請參閱 Amplify 開發人員中心以 AWS Amplify取得建置應用程式的選項。

> **Note**
>
> 備註：如果您是在 2015 年 2 月之前建立身分集區，則需要將您的角色與身分集區重新建立關聯，以便使用此建構函數，而不需要將角色做為參數。若要執行此作業，請開啟 Amazon Cognito 主控台，選擇 Manage identity pools (管理身分集區)，選取您的身分集區，然後選擇 Edit identity Pool (編輯身分集區)，指定您的已驗證和未驗證角色，然後儲存變更。

擷取 Amazon Cognito 身分

如果您要允許未驗證的使用者，可以立即為您的最終使用者擷取唯一 Amazon Cognito 識別符 (身分 ID)，或者如果您要驗證使用者，可以在憑證供應商中設定登入權杖之後，擷取身分 ID：

```
var identityId = await credentials.GetIdentityIdAsync();
```

存取 AWS 服務

設定 Amazon Cognito 登入資料提供者並擷取 AWS 登入資料後，您可以建立 AWS 服務 用戶端。

AWS 用於建立用戶端的 SDK 資源

- AWS AWS SDK for C++ 開發人員指南中的用戶端設定
- 在 AWS SDK for Go 開發人員指南 AWS 服務中搭配使用 AWS SDK for Go V2
- 在 AWS SDK for Java 2.x 開發人員指南中設定 HTTP 用戶端
- 在 AWS SDK for JavaScript 開發人員指南中建立和呼叫服務物件

- 在 AWS SDK for Python (Boto3) 文件中建立用戶端
- 在 適用於 Rust 的 AWS SDK 開發人員指南中建立服務用戶端
- 在 適用於 Swift 的 AWS SDK 開發人員指南中使用用戶端

下列程式碼片段會初始化 Amazon DynamoDB 用戶端：

Android

若要在 Android 應用程式中使用亞馬遜認可身分集區，請進行設 AWS Amplify定。如需詳細資訊，請參閱《Amplify 開發人員中心》中的身分驗證。

```
// Create a service client with the provider
AmazonDynamoDB client = new AmazonDynamoDBClient(credentialsProvider);
```

登入資料提供者會與 Amazon Cognito 通訊，同時擷取已驗證和未驗證使用者的唯一識別碼，以及行動 SDK 的臨時、有限特權 AWS 登入 AWS 資料。所擷取的登入資料有效期限為 1 小時，當其過期時，供應商會將其重新整理。

iOS - Objective-C

若要在 iOS 應用程式中使用 Amazon Cognito 身分識別集區，請進行設 AWS Amplify定。如需詳細資訊，請參閱《Amplify 開發人員中心》中的 Swift 身分驗證和 Flutter 身分驗證。

```
// create a configuration that uses the provider
AWSServiceConfiguration *configuration = [AWSServiceConfiguration
 configurationWithRegion:AWSRegionUSEast1 provider:credentialsProvider];
// get a client with the default service configuration
AWSDynamoDB *dynamoDB = [AWSDynamoDB defaultDynamoDB];
```

登入資料提供者會與 Amazon Cognito 通訊，同時擷取已驗證和未驗證使用者的唯一識別碼，以及行動 SDK 的臨時、有限特權 AWS 登入 AWS 資料。所擷取的登入資料有效期限為 1 小時，當其過期時，供應商會將其重新整理。

iOS - Swift

若要在 iOS 應用程式中使用 Amazon Cognito 身分識別集區，請進行設 AWS Amplify定。如需詳細資訊，請參閱《Amplify 開發人員中心》中的 Swift 身分驗證。

```
// get a client with the default service configuration
let dynamoDB = AWSDynamoDB.default()
```

```
// get a client with a custom configuration
AWSDynamoDB.register(with: configuration!, forKey: "USWest2DynamoDB");
let dynamoDBCustom = AWSDynamoDB(forKey: "USWest2DynamoDB")
```

登入資料提供者會與 Amazon Cognito 通訊，同時擷取已驗證和未驗證使用者的唯一識別碼，以及行動 SDK 的臨時、有限特權 AWS 登入 AWS 資料。所擷取的登入資料有效期限為 1 小時，當其過期時，供應商會將其重新整理。

JavaScript

```
// Create a service client with the provider
var dynamodb = new AWS.DynamoDB({region: 'us-west-2'});
```

登入資料提供者會與 Amazon Cognito 通訊，同時擷取已驗證和未驗證使用者的唯一識別碼，以及行動 SDK 的臨時、有限權限 AWS 登入資料。 AWS 所擷取的登入資料有效期限為 1 小時，當其過期時，供應商會將其重新整理。

Unity

AWS SDK for Unity 現在是 AWS SDK for .NET 的一部分。若要在中開始使用 Amazon Cognito AWS SDK for .NET，請參閱開 AWS SDK for .NET 發人員指南中的 Amazon Cognito 登入資料供應商。或者，請參閱 Amplify 開發人員中心以 AWS Amplify取得建置應用程式的選項。

```
// create a service client that uses credentials provided by Cognito
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials, REGION);
```

登入資料提供者會與 Amazon Cognito 通訊，同時擷取已驗證和未驗證使用者的唯一識別碼，以及行動 SDK 的臨時、有限權限 AWS 登入資料。 AWS 所擷取的登入資料有效期限為 1 小時，當其過期時，供應商會將其重新整理。

Xamarin

AWS SDK for Xamarin 現在是 AWS SDK for .NET 的一部分。若要在中開始使用 Amazon Cognito AWS SDK for .NET，請參閱開 AWS SDK for .NET 發人員指南中的 Amazon Cognito 登入資料供應商。或者，請參閱 Amplify 開發人員中心以 AWS Amplify取得建置應用程式的選項。

```
// create a service client that uses credentials provided by Cognito
var client = new AmazonDynamoDBClient(credentials, REGION)
```

登入資料提供者會與 Amazon Cognito 通訊，同時擷取已驗證和未驗證使用者的唯一識別碼，以及行動 SDK 的臨時、有限權限 AWS 登入資料。 AWS 所擷取的登入資料有效期限為 1 小時，當其過期時，供應商會將其重新整理。

外部身分提供者身分集區

使用 `logins` 屬性，您便可設定從身分提供者 (IdP).收到的憑證。此外，您可以將身分集區與多個身分集區相關聯 IdPs。例如，您可以在 `logins` 屬性中同時設定 Facebook 和 Google 權杖，以將唯一 Amazon Cognito 身分與這兩個 IdP 登入都建立關聯。使用者可以使用帳戶進行身分驗證，但 Amazon Cognito 會傳回相同的使用者識別符。

下列指示會引導您透過 Amazon Cognito 身分集區支援 IdPs 的身份驗證進行身份驗證。

主題

將臉書設定為身分集區 IdP

Amazon Cognito 身分集區與 Facebook 整合，可為您的行動應用程式使用者提供聯合身分驗證。本節說明如何使用 Facebook 做為 IdP 來註冊及設定您的應用程式。

設定 Facebook

請先向 Facebook 註冊您的應用程式，再驗證 Facebook 使用者，並與 Facebook API 互動。

Facebook 開發人員入口網站可幫助您設定應用程式。您在 Amazon Cognito 身分集區中整合 Facebook 之前，請先執行此操作程序：

設定 Facebook

1. 在 Facebook 開發人員入口網站中，使用您的 Facebook 登入資料來登入。
2. 從 Apps (應用程式) 功能表中，選取 Add a New App (新增應用程式)。

3. 選取平台並完成的快速啟動程序。

Android

如需如何將 Android 應用程式與 Facebook 登入整合的詳細資訊，請參閱 Facebook 入門指南。

iOS - Objective-C

如需如何將 iOS Objective-C 應用程式與 Facebook 登入整合的詳細資訊，請參閱 Facebook 入門指南。

iOS - Swift

如需如何將 iOS Swift 應用程式與 Facebook 登入整合的詳細資訊，請參閱 Facebook 入門指南。

JavaScript

如需有關如何將 JavaScript 網路應用程式與 Facebook 登入整合的詳細資訊，請參閱 Facebook 入門指南。

Unity

如需如何將 Unity 應用程式與 Facebook 登入整合的詳細資訊，請參閱 Facebook 入門指南。

Xamarin

若要新增 Facebook 身分驗證，請先遵循下方適用的流程，將 Facebook 軟體開發套件整合至您的應用程式中。Amazon Cognito 身分集區會使用 Facebook 存取權杖以產生與 Cognito 身分相關的唯一使用者識別符。

- Xamarin 提供的 Facebook iOS 軟體開發套件
- Xamarin 提供的 Facebook Android 軟體開發套件

在 Amazon Cognito 身分池主控台中設定身分提供者

使用下列程序設定您的身分提供者。

若要新增 Facebook 身分提供者 (IdP)

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。
3. 選取 新增身分供應商。

4. 選擇 Facebook。
5. 輸入您在開發人員中繼資料中建立的 OAuth 專案應用程式 ID。如需詳細資訊，請參閱開發人員文件中繼資料中的 Facebook 登入。
6. 若要設定 Amazon Cognito 向已通過此提供者進行身分驗證的使用者發布憑證時的角色，請設定角色設定。

   - 您可以為該 IdP 使用者指派設定 已驗證角色 時的 預設角色，或您可以 選擇具有規則的角色。

     i. 如果您選擇 使用規則選擇角色，請輸入使用者身分驗證的 宣告 來源、比較宣告的 操作員、導致符合角色選擇的 值，以及當符合 角色指派 時您要指派的 角色。選取 新增另一項 以根據不同的條件建立其他規則。

     ii. 選擇 角色解析。當您的使用者宣告與您的規則不符時，您可以拒絕憑證或向 已驗證角色發出憑證。
7. 若要變更透過此提供者驗證使用者，Amazon Cognito 發布憑證時指派的主要索引標籤，請設定 存取控制的屬性。

   a. 若不套用主要索引標籤，請選擇 非作用中。

   b. 若要根據 sub 和 aud 宣告套用主要索引標籤，請選擇 使用預設對應。

   c. 若要建立您自己的自訂屬性結構描述至主要索引標籤，請選擇 使用自訂對應。然後，輸入您要從每個 宣告 中獲取的 標籤金鑰，顯示於索引標籤當中。
8. 選取儲存變更。

使用 Facebook

Android

若要新增 Facebook 身分驗證，請先遵循 Facebook 指南，並將 Facebook 軟體開發套件整合至您的應用程式中。然後，新增 Login with Facebook (使用 Facebook 登入) 按鈕至您的 Android 使用者界面。Facebook 軟體開發套件會使用工作階段物件來追蹤其狀態。Amazon Cognito 使用此工作階段物件中的存取權杖來驗證使用者、產生唯一識別碼，並視需要授與使用者存取其他 AWS 資源。

使用 Facebook 軟體開發套件來驗證您的使用者之後，請將工作階段權杖新增至 Amazon Cognito 憑證供應商。

Facebook 軟體開發套件 4.0 或更新版本：

```
Map<String, String> logins = new HashMap<String, String>();
```

```
logins.put("graph.facebook.com", AccessToken.getCurrentAccessToken().getToken());
credentialsProvider.setLogins(logins);
```

4.0 之前的 Facebook 軟體開發套件：

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("graph.facebook.com", Session.getActiveSession().getAccessToken());
credentialsProvider.setLogins(logins);
```

Facebook 登入程序會在其軟體開發套件中初始化單一工作階段。Facebook 會話對象包含一個 OAuth 令牌，亞馬遜認可用於為您的身份驗證的最終用戶生成 AWS 憑據。Amazon Cognito 也會使用權杖來檢查您的使用者資料庫中是否有符合此特定 Facebook 身分的使用者存在。如果該使用者已存在，API 會傳回現有的識別符，否則，API 會傳回新的識別符。用戶端開發套件會自動將識別符快取在本機裝置上。

> **Note**
>
> 設定登入對應後，請呼叫`refresh`或擷`get`取 AWS 認證。

iOS - Objective-C

若要新增 Facebook 身分驗證，請先遵循 Facebook 指南，並將 Facebook 軟體開發套件整合至您的應用程式中。然後，將 Login with Facebook (使用 Facebook 登入) 按鈕新增至您的使用者界面。Facebook 軟體開發套件會使用工作階段物件來追蹤其狀態。Amazon Cognito 會使用來自此工作階段物件的存取權杖進行使用者身分驗證，並將其繫結至唯一 Amazon Cognito 身分集區 (聯合身分)。

若要提供 Facebook 存取權杖給 Amazon Cognito，請實作 AWSIdentityProviderManager 協定。

在實作 `logins` 方法時，傳回包含 `AWSIdentityProviderFacebook` 的字典。這個字典可做為金鑰，並以來自已驗證之 Facebook 使用者的現行存取權杖做為數值，如下列程式碼範例所示。

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    FBSDKAccessToken* fbToken = [FBSDKAccessToken currentAccessToken];
    if(fbToken){
        NSString *token = fbToken.tokenString;
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook : token }];
    }else{
            return [AWSTask taskWithError:[NSError errorWithDomain:@"Facebook Login"
                                                              code:-1
```

```
                                                        userInfo:@{@"error":@"No current
 Facebook access token"}]];
    }
}
```

當您將 AWSCognitoCredentialsProvider執行個體化時，請傳遞在建構函式中將 AWSIdentityProviderManager 實作為 identityProviderManager 值的類別。如需詳細資訊，請前往AWSCognitoCredentialsProvider參考頁面，然後選擇「initWithRegion類型:identityPoolId:」identityProviderManager。

iOS - Swift

若要新增 Facebook 身分驗證，請先遵循 Facebook 指南，並將 Facebook 軟體開發套件整合至您的應用程式中。然後，將 Login with Facebook (使用 Facebook 登入) 按鈕新增至您的使用者界面。Facebook 軟體開發套件會使用工作階段物件來追蹤其狀態。Amazon Cognito 會使用來自此工作階段物件的存取權杖進行使用者身分驗證，並將其繫結至唯一 Amazon Cognito 身分集區 (聯合身分)。

若要提供 Facebook 存取權杖給 Amazon Cognito，請實作 AWSIdentityProviderManager 協定。

在實作 logins 方法時，傳回包含 AWSIdentityProviderFacebook 的字典。這個字典可做為金鑰，並以來自已驗證之 Facebook 使用者的現行存取權杖做為數值，如下列程式碼範例所示。

```
class FacebookProvider: NSObject, AWSIdentityProviderManager {
    func logins() -> AWSTask<NSDictionary> {
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }
}
```

當您將 AWSCognitoCredentialsProvider執行個體化時，請傳遞在建構函式中將 AWSIdentityProviderManager 實作為 identityProviderManager 值的類別。如需詳細資訊，請前往AWSCognitoCredentialsProvider參考頁面，然後選擇「initWithRegion類型:identityPoolId:」identityProviderManager。

JavaScript

若要新增 Facebook 身分驗證，請遵循網頁版 Facebook 登入，在您的網站上新增 Login with Facebook (使用 Facebook 登入) 按鈕。Facebook 軟體開發套件會使用工作階段物件來追蹤其狀

態。Amazon Cognito 使用此工作階段物件中的存取權杖來驗證使用者、產生唯一識別碼，並視需要授與使用者存取其他 AWS 資源。

使用 Facebook 軟體開發套件來驗證您的使用者之後，請將工作階段權杖新增至 Amazon Cognito 憑證供應商。

```
FB.login(function (response) {

  // Check if the user logged in successfully.
  if (response.authResponse) {

    console.log('You are now logged in.');

    // Add the Facebook access token to the Amazon Cognito credentials login map.
    AWS.config.credentials = new AWS.CognitoIdentityCredentials({
      IdentityPoolId: 'IDENTITY_POOL_ID',
      Logins: {
        'graph.facebook.com': response.authResponse.accessToken
      }
    });

    // Obtain AWS credentials
    AWS.config.credentials.get(function(){
        // Access AWS resources here.
    });

  } else {
    console.log('There was a problem logging you in.');
  }

});
```

Facebook SDK 獲得 OAuth 令牌，亞馬遜認可用於為您的身份驗證的最終用戶生成 AWS 憑據。Amazon Cognito 也會使用權杖來檢查您的使用者資料庫中是否有符合此特定 Facebook 身分的使用者存在。如果該使用者已存在，API 會傳回現有的識別符，否則就會傳回新的識別符。用戶端軟體開發套件會自動將識別符快取在本機裝置上。

> **Note**
>
> 設定登入對應之後，必須發出 `refresh` 或 `get` 呼叫，以取得憑證。如需程式碼範例，請參閱 JavaScript README 檔案中的「使用案例 17，使用 Cognito 身分整合使用者集區」。

Unity

若要新增 Facebook 身分驗證，請先遵循 Facebook 指南，並將 Facebook 軟體開發套件整合至您的應用程式中。Amazon Cognito 會使用來自 FB 物件的 Facebook 存取權杖，產生與 Amazon Cognito 身分相關聯的唯一使用者識別符。

使用 Facebook 軟體開發套件來驗證您的使用者之後，請將工作階段權杖新增至 Amazon Cognito 憑證供應商：

```
void Start()
{
    FB.Init(delegate() {
        if (FB.IsLoggedIn) { //User already logged in from a previous session
            AddFacebookTokenToCognito();
        } else {
            FB.Login ("email", FacebookLoginCallback);
        }
    });
}

void FacebookLoginCallback(FBResult result)
{
    if (FB.IsLoggedIn)
    {
        AddFacebookTokenToCognito();
    }
    else
    {
        Debug.Log("FB Login error");
    }
}

void AddFacebookTokenToCognito()
{
    credentials.AddLogin ("graph.facebook.com",
 AccessToken.CurrentAccessToken.TokenString);
}
```

在使用 FB.AccessToken 之前，請呼叫 FB.Login() 並確認 FB.IsLoggedIn 為 True。

Xamarin

適用於 Android 的 Xamarin：

```
public void InitializeFacebook() {
    FacebookSdk.SdkInitialize(this.ApplicationContext);
    callbackManager = CallbackManagerFactory.Create();
    LoginManager.Instance.RegisterCallback(callbackManager, new FacebookCallback &lt;
 LoginResult &gt; () {
      HandleSuccess = loginResult = &gt; {
        var accessToken = loginResult.AccessToken;
        credentials.AddLogin("graph.facebook.com", accessToken.Token);
        //open new activity
      },
      HandleCancel = () = &gt; {
        //throw error message
      },
      HandleError = loginError = &gt; {
        //throw error message
      }
    });
    LoginManager.Instance.LogInWithReadPermissions(this, new List &lt; string &gt; {
      "public_profile"
    });
  }
```

適用於 iOS 的 Xamarin：

```
public void InitializeFacebook() {
  LoginManager login = new LoginManager();
  login.LogInWithReadPermissions(readPermissions.ToArray(),
 delegate(LoginManagerLoginResult result, NSError error) {
    if (error != null) {
      //throw error message
    } else if (result.IsCancelled) {
      //throw error message
    } else {
      var accessToken = loginResult.AccessToken;
      credentials.AddLogin("graph.facebook.com", accessToken.Token);
      //open new view controller
    }
  });
}
```

設置 Login with Amazon 作為身份池 IdP

Amazon Cognito 與登入 Amazon 整合，可為您的行動和 Web 應用程式使用者提供聯合身分驗證。本節說明如何使用登入 Amazon 做為身分提供者 (IdP) 來註冊及設定您的應用程式。

設定登入 Amazon 以使用開發人員入口網站中的 Amazon Cognito。如需詳細資訊，請參閱登入 Amazon 常見問答集中的設定登入 Amazon。

> **Note**
>
> 若要將登入 Amazon 整合至 Xamarin 應用程式，請遵循 Xamarin 入門指南。

> **Note**
>
> 您原本就無法在 Unity 平台上整合登入 Amazon。請改用 Web 檢視，然後進行瀏覽器登入流程。

設定登入 Amazon

實作 Login with Amazon

在 Amazon 開發人員入口網站中，您可以設定 OAuth 應用程式以與您的身分集區整合、尋找登入 Amazon 文件，並下載開發套件。選擇 Developer console (開發人員主控台)，然後選擇開發人員入口網站中的 Login with Amazon (登入 Amazon)。您可以為您的應用程式建立安全性描述檔，然後在您的應用程式中建置登入 Amazon 身分驗證機制。請參閱 取得憑證 以取得如何將登入 Amazon 身分驗證與應用程式整合的詳細資訊。

Amazon 為您的新安全性描述檔核發 OAuth 2.0 用戶端 ID。您可以在安全性描述檔的 Web Settings (Web 設定) 索引標籤上找到 client ID (用戶端 ID)。在您身分集區中登入 Amazon IdP 的應用程式 ID欄位中輸入安全性設定檔 ID。

> **Note**
>
> 在您身分集區中登入 Amazon IdP 的應用程式 ID欄位中輸入安全性設定檔 ID。這與使用用戶端 ID 的使用者集區不同。

在 Amazon Cognito 主控台中設定外部供應商

若要新增使用 Amazon 登入身分提供者 (IdP)

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。
3. 選取 新增身分供應商。
4. 選擇 Login with Amazon。
5. 輸入您在 Login with Amazon 中建立的 OAuth 專案 應用程式 ID。如需詳細資訊，請參閱 Login with Amazon 說明文件。
6. 若要設定 Amazon Cognito 向已通過此提供者進行身分驗證的使用者發布憑證時的角色，請設定角色設定。

   - 您可以為該 IdP 使用者指派設定 已驗證角色 時的 預設角色，或您可以 選擇具有規則的角色。

     i. 如果您選擇 使用規則選擇角色，請輸入使用者身分驗證的 宣告 來源、比較宣告的 操作員、導致符合角色選擇的 值，以及當符合 角色指派 時您要指派的 角色。選取 新增另一項 以根據不同的條件建立其他規則。

     ii. 選擇 角色解析。當您的使用者宣告與您的規則不符時，您可以拒絕憑證或向 已驗證角色發出憑證。
7. 若要變更透過此提供者驗證使用者，Amazon Cognito 發布憑證時指派的主要索引標籤，請設定 存取控制的屬性。

   a. 若不套用主要索引標籤，請選擇 非作用中。

   b. 若要根據 sub 和 aud 宣告套用主要索引標籤，請選擇 使用預設對應。

   c. 若要建立您自己的自訂屬性結構描述至主要索引標籤，請選擇 使用自訂對應。然後，輸入您要從每個 宣告 中獲取的 標籤金鑰，顯示於索引標籤當中。
8. 選取儲存變更。

使用 Login with Amazon：Android

驗證 Amazon 登錄後，您可以使用界面的 onSuccess 方法將令牌傳遞給 Amazon Cognito 登入資料提供商。 TokenListener 程式碼看起來像這樣：

```
@Override
```

```
public void onSuccess(Bundle response) {
    String token = response.getString(AuthzConstants.BUNDLE_KEY.TOKEN.val);
    Map<String, String> logins = new HashMap<String, String>();
    logins.put("www.amazon.com", token);
    credentialsProvider.setLogins(logins);
}
```

使用 Login with Amazon：iOS - Objective-C

驗證亞馬遜登錄後，您可以使用 AM AccessTokenDelegate ZN 的 requestDidSucceed 方法將令牌傳遞給 Amazon Cognito 登入資料提供商：

```
- (void)requestDidSucceed:(APIResult \*)apiResult {
    if (apiResult.api == kAPIAuthorizeUser) {
        [AIMobileLib getAccessTokenForScopes:[NSArray arrayWithObject:@"profile"]
 withOverrideParams:nil delegate:self];
    }
    else if (apiResult.api == kAPIGetAccessToken) {
        credentialsProvider.logins = @{ @(AWSCognitoLoginProviderKeyLoginWithAmazon):
 apiResult.result };
    }
}}
```

使用 Login with Amazon：iOS - Swift

對 Amazon 登入進行身分驗證之後，您可以在 `AMZNAccessTokenDelegate` 的 `requestDidSucceed` 方法中，將權杖傳遞給 Amazon Cognito 憑證供應商：

```
func requestDidSucceed(apiResult: APIResult!) {
    if apiResult.api == API.AuthorizeUser {
        AIMobileLib.getAccessTokenForScopes(["profile"], withOverrideParams: nil,
 delegate: self)
    } else if apiResult.api == API.GetAccessToken {
        credentialsProvider.logins =
 [AWSCognitoLoginProviderKey.LoginWithAmazon.rawValue: apiResult.result]
    }
}
```

Login with Amazon： JavaScript

使用者以 Login with Amazon 進行驗證，並重新導向回您的網站之後，查詢字串中會提供 Login with Amazon access_token。請將該權杖傳遞至登入資料登入對應中。

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
   IdentityPoolId: 'IDENTITY_POOL_ID',
   Logins: {
       'www.amazon.com': 'Amazon Access Token'
   }
});
```

使用 Login with Amazon：Xamarin

適用於 Android 的 Xamarin

```
AmazonAuthorizationManager manager = new AmazonAuthorizationManager(this,
 Bundle.Empty);

var tokenListener = new APIListener {
  Success = response => {
    // Get the auth token
    var token = response.GetString(AuthzConstants.BUNDLE_KEY.Token.Val);
    credentials.AddLogin("www.amazon.com", token);
  }
};

// Try and get existing login
manager.GetToken(new[] {
  "profile"
}, tokenListener);
```

適用於 iOS 的 Xamarin

在 AppDelegate.cs 中，插入下列內容：

```
public override bool OpenUrl (UIApplication application, NSUrl url, string
 sourceApplication, NSObject annotation)
{
    // Pass on the url to the SDK to parse authorization code from the url
    bool isValidRedirectSignInURL = AIMobileLib.HandleOpenUrl (url, sourceApplication);
    if(!isValidRedirectSignInURL)
        return false;

    // App may also want to handle url
    return true;
}
```

在 ViewController.cs 中，執行下列操作：

```
public override void ViewDidLoad ()
{
    base.LoadView ();

    // Here we create the Amazon Login Button
    btnLogin = UIButton.FromType (UIButtonType.RoundedRect);
    btnLogin.Frame = new RectangleF (55, 206, 209, 48);
    btnLogin.SetTitle ("Login using Amazon", UIControlState.Normal);
    btnLogin.TouchUpInside += (sender, e) => {
        AIMobileLib.AuthorizeUser (new [] { "profile"}, new AMZNAuthorizationDelegate
 ());
    };
    View.AddSubview (btnLogin);
}

// Class that handles Authentication Success/Failure
public class AMZNAuthorizationDelegate : AIAuthenticationDelegate
{
  public override void RequestDidSucceed(ApiResult apiResult)
    {
      // Your code after the user authorizes application for requested scopes
      var token = apiResult["access_token"];
      credentials.AddLogin("www.amazon.com",token);
    }

    public override void RequestDidFail(ApiError errorResponse)
    {
      // Your code when the authorization fails
      InvokeOnMainThread(() => new UIAlertView("User Authorization Failed",
 errorResponse.Error.Message, null, "Ok", null).Show());
    }
}
```

將谷歌設置為身份池 IdP

Amazon Cognito 可整合 Google，為您的行動應用程式使用者提供聯合身分驗證。本節說明如何使用 Google 做為 IdP 來註冊及設定您的應用程式。

Android

> **Note**
>
> 如果您的應用程式使用 Google 並且將可在多個行動平台上使用，則應將其設定為 OpenID Connect 供應商。將所有建立的用戶端 ID 新增為其他對象值，以便達到更佳的整合。若要進一步了解 Google 的跨用戶端身分模型，請參閱跨用戶端身分。

設定 Google

若要啟用適用於 Android 的 Google 登入，請為應用程式建立 Google 開發人員主控台專案。

1. 前往 Google 開發人員主控台，並建立新的專案。
2. 選擇 APIs & Services (API 與服務)，然後選擇 OAuth consent screen (OAuth 同意畫面)。自訂當 Google 要求使用者同意將其設定檔資料與您的應用程式共享時，Google 要向使用者顯示的資訊。
3. 選擇 Credentials (憑證)，然後選擇 Create credentials (建立憑證)。選擇 OAuth client ID (OAuth 用戶端 ID)。選取 Android 做為 Application type (應用程式類型)。為開發應用程式所在的每個平台建立單獨的用戶端 ID。
4. 從 Credentials (憑證)，選擇 Manage service accounts (管理服務帳戶)。選擇 Create service account (建立服務帳戶)。輸入您的服務帳戶詳細資訊，然後選擇 Create and continue (建立並繼續)。
5. 將服務帳戶存取權授與您的專案。根據您應用程式的需求將服務帳戶的存取權授與使用者。
6. 選擇您的新服務帳戶，選擇 Keys (金鑰) 索引標籤，然後選擇 Add key (新增金鑰)。建立並下載新的 JSON 金鑰。

如需使用 Google 開發人員主控台的詳細資訊，請參閱 Google Cloud 文件中的建立與管理專案。

有關如何將 Google 集成到您的 Android 應用程序中的更多信息，請參閱 Google 身份說明文檔中的使用 Google 登錄對用戶進行身份驗證。

若要新增 Google 身分提供者 (IdP)

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。
3. 選取 新增身分供應商。

4. 選擇 Google。
5. 輸入您在 Google Cloud Platform 建立的 OAuth 專案 用戶端 ID。如需詳細資訊，請參閱《Google Cloud Platform 主控台幫助》中的設定 OAuth 2.0。
6. 若要設定 Amazon Cognito 向已通過此提供者進行身分驗證的使用者發布憑證時的角色，請設定角色設定。

   - 您可以為該 IdP 使用者指派設定 已驗證角色 時的 預設角色，或您可以 選擇具有規則的角色。

     i. 如果您選擇 使用規則選擇角色，請輸入使用者身分驗證的 宣告 來源、比較宣告的 操作員、導致符合角色選擇的 值，以及當符合 角色指派 時您要指派的 角色。選取 新增另一項 以根據不同的條件建立其他規則。

     ii. 選擇 角色解析。當您的使用者宣告與您的規則不符時，您可以拒絕憑證或向 已驗證角色發出憑證。

7. 若要變更透過此提供者驗證使用者，Amazon Cognito 發布憑證時指派的主要索引標籤，請設定 存取控制的屬性。

   a. 若不套用主要索引標籤，請選擇 非作用中。

   b. 若要根據 sub 和 aud 宣告套用主要索引標籤，請選擇 使用預設對應。

   c. 若要建立您自己的自訂屬性結構描述至主要索引標籤，請選擇 使用自訂對應。然後，輸入您要從每個 宣告 中獲取的 標籤金鑰，顯示於索引標籤當中。

8. 選取儲存變更。

使用 Google

若要在您的應用程式中啟用以 Google 登入，請遵循適用於 Android 的 Google 說明文件中的指示。當使用者登入時，他們會向 Google 請求 OpenID Connect 身分驗證權杖。然後 Amazon Cognito 會使用該權杖來驗證使用者身分，並產生唯一識別符。

下列範例程式碼說明如何從 Google Play 服務擷取身分驗證權杖：

```
GooglePlayServicesUtil.isGooglePlayServicesAvailable(getApplicationContext());
AccountManager am = AccountManager.get(this);
Account[] accounts = am.getAccountsByType(GoogleAuthUtil.GOOGLE_ACCOUNT_TYPE);
String token = GoogleAuthUtil.getToken(getApplicationContext(), accounts[0].name,
        "audience:server:client_id:YOUR_GOOGLE_CLIENT_ID");
Map<String, String> logins = new HashMap<String, String>();
logins.put("accounts.google.com", token);
```

```
credentialsProvider.setLogins(logins);
```

iOS - Objective-C

> **Note**
>
> 如果您的應用程式使用 Google 並且可在多個行動平台上使用，則應將 Google 設定為 OpenID Connect 供應商。將所有建立的用戶端 ID 新增為其他對象值，以便達到更佳的整合。若要進一步了解 Google 的跨用戶端身分模型，請參閱跨用戶端身分。

設定 Google

若要啟用適用於 iOS 的 Google 登入，請為應用程式建立 Google 開發人員主控台專案。

1. 前往 Google 開發人員主控台，並建立新的專案。
2. 選擇 APIs & Services (API 與服務)，然後選擇 OAuth consent screen (OAuth 同意畫面)。自訂當 Google 要求使用者同意將其設定檔資料與您的應用程式共享時，Google 要向使用者顯示的資訊。
3. 選擇 Credentials (憑證)，然後選擇 Create credentials (建立憑證)。選擇 OAuth client ID (OAuth 用戶端 ID)。選取 iOS 做為 Application type (應用程式類型)。為開發應用程式所在的每個平台建立單獨的用戶端 ID。
4. 從 Credentials (憑證)，選擇 Manage service accounts (管理服務帳戶)。選擇 Create service account (建立服務帳戶)。輸入您的服務帳戶詳細資訊，然後選擇 Create and continue (建立並繼續)。
5. 將您專案的存取權授與服務帳戶。根據您應用程式的需求將服務帳戶的存取權授與使用者。
6. 選擇您的新服務帳戶。選擇 Keys (金鑰) 索引標籤，然後選擇 Add key (新增金鑰)。建立並下載新的 JSON 金鑰。

如需使用 Google 開發人員主控台的詳細資訊，請參閱 Google Cloud 文件中的建立與管理專案。

如需如何將 Google 整合到您 iOS 應用程式的詳細資訊，請參閱 Google Identity 文件中的適用於 iOS 的 Google 登入。

若要新增 Google 身分提供者 (IdP)

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。

3. 選取 新增身分供應商。
4. 選擇 Google。
5. 輸入您在 Google Cloud Platform 建立的 OAuth 專案 用戶端 ID。如需詳細資訊，請參閱《Google Cloud Platform 主控台幫助》中的設定 OAuth 2.0。
6. 若要設定 Amazon Cognito 向已通過此提供者進行身分驗證的使用者發布憑證時的角色，請設定角色設定。

    - 您可以為該 IdP 使用者指派設定 已驗證角色 時的 預設角色，或您可以 選擇具有規則的角色。

        i. 如果您選擇 使用規則選擇角色，請輸入使用者身分驗證的 宣告 來源、比較宣告的 操作員、導致符合角色選擇的 值，以及當符合 角色指派 時您要指派的 角色。選取 新增另一項 以根據不同的條件建立其他規則。

        ii. 選擇 角色解析。當您的使用者宣告與您的規則不符時，您可以拒絕憑證或向 已驗證角色發出憑證。

7. 若要變更透過此提供者驗證使用者，Amazon Cognito 發布憑證時指派的主要索引標籤，請設定 存取控制的屬性。

    a. 若不套用主要索引標籤，請選擇 非作用中。

    b. 若要根據 sub 和 aud 宣告套用主要索引標籤，請選擇 使用預設對應。

    c. 若要建立您自己的自訂屬性結構描述至主要索引標籤，請選擇 使用自訂對應。然後，輸入您要從每個 宣告 中獲取的 標籤金鑰，顯示於索引標籤當中。

8. 選取儲存變更。

使用 Google

若要在您的應用程式中啟用以 Google 登入，請遵循適用於 iOS 的 Google 說明文件。身分驗證成功會產生 OpenID Connect 身分驗證權杖，Amazon Cognito 會用它來驗證使用者，並產生唯一識別符。

身分驗證成功會產生包含 `id_token` 的 `GTMOAuth2Authentication` 物件，Amazon Cognito 會用其來驗證使用者，並產生唯一識別符：

```
- (void)finishedWithAuth: (GTMOAuth2Authentication *)auth error: (NSError *) error {
        NSString *idToken = [auth.parameters objectForKey:@"id_token"];
        credentialsProvider.logins = @{ @(AWSCognitoLoginProviderKeyGoogle): idToken };
    }
```

iOS - Swift

> **Note**
>
> 如果您的應用程式使用 Google 並且可在多個行動平台上使用，則應將 Google 設定為 OpenID Connect 供應商。將所有建立的用戶端 ID 新增為其他對象值，以便達到更佳的整合。若要進一步了解 Google 的跨用戶端身分模型，請參閱跨用戶端身分。

設定 Google

若要啟用適用於 iOS 的 Google 登入，請為應用程式建立 Google 開發人員主控台專案。

1. 前往 Google 開發人員主控台，並建立新的專案。
2. 選擇 APIs & Services (API 與服務)，然後選擇 OAuth consent screen (OAuth 同意畫面)。自訂當 Google 要求使用者同意將其設定檔資料與您的應用程式共享時，Google 要向使用者顯示的資訊。
3. 選擇 Credentials (憑證)，然後選擇 Create credentials (建立憑證)。選擇 OAuth client ID (OAuth 用戶端 ID)。選取 iOS 做為 Application type (應用程式類型)。為開發應用程式所在的每個平台建立單獨的用戶端 ID。
4. 從 Credentials (憑證)，選擇 Manage service accounts (管理服務帳戶)。選擇 Create service account (建立服務帳戶)。輸入您的服務帳戶詳細資訊，然後選擇 Create and continue (建立並繼續)。
5. 將您專案的存取權授與服務帳戶。根據您應用程式的需求將服務帳戶的存取權授與使用者。
6. 選擇您的新服務帳戶，選擇 Keys (金鑰) 索引標籤，然後選擇 Add key (新增金鑰)。建立並下載新的 JSON 金鑰。

如需使用 Google 開發人員主控台的詳細資訊，請參閱 Google Cloud 文件中的建立與管理專案。

如需如何將 Google 整合到您 iOS 應用程式的詳細資訊，請參閱 Google Identity 文件中的適用於 iOS 的 Google 登入。

從 Amazon Cognito 主控台首頁選擇 Manage Identity Pools (管理身分集區)：

在 Amazon Cognito 主控台中設定外部供應商

1. 針對要啟用 Google 做為外部供應商的身分集區，選擇其名稱。該身分集區的 Dashboard (儀表板) 頁面隨即出現。

2. 在 Dashboard (儀表板) 頁面右上角，選擇 Edit identity pool (編輯身分集區)。Edit identity pool (編輯身分集區) 頁面隨即出現。
3. 向下捲動，然後選擇 Authentication providers (身分驗證供應商) 以展開該區段。
4. 選擇 Google 索引標籤。
5. 選擇 Unlock (解除鎖定)。
6. 輸入您從 Google 取得的 Google 用戶端 ID，然後選擇 Save Changes (儲存變更)。

使用 Google

若要在您的應用程式中啟用以 Google 登入，請遵循適用於 iOS 的 Google 說明文件。身分驗證成功會產生 OpenID Connect 身分驗證權杖，Amazon Cognito 會用它來驗證使用者，並產生唯一識別符。

成功的身分驗證會產生 `GTMOAuth2Authentication` 物件，該物件包含 `id_token`。Amazon Cognito 會使用此權杖來驗證使用者身分，並產生唯一識別符。

```
func finishedWithAuth(auth: GTMOAuth2Authentication!, error: NSError!) {
    if error != nil {
      print(error.localizedDescription)
    }
    else {
      let idToken = auth.parameters.objectForKey("id_token")
      credentialsProvider.logins = [AWSCognitoLoginProviderKey.Google.rawValue:
 idToken!]
    }
}
```

JavaScript

> **Note**
>
> 如果您的應用程式使用 Google 並可在多個行動平台上使用，則應將 Google 設定為 OpenID Connect 供應商。將所有建立的用戶端 ID 新增為其他對象值，以便達到更佳的整合。若要進一步了解 Google 的跨用戶端身分模型，請參閱跨用戶端身分。

設定 Google

要為 JavaScript Web 應用程序啟用 Google 登錄，請為您的應用程序創建一個 Google 開發人員控制台項目。

1. 前往 Google 開發人員主控台，並建立新的專案。
2. 選擇 APIs & Services (API 與服務)，然後選擇 OAuth consent screen (OAuth 同意畫面)。自訂當 Google 要求使用者同意將其設定檔資料與您的應用程式共享時，Google 要向使用者顯示的資訊。
3. 選擇 Credentials (憑證)，然後選擇 Create credentials (建立憑證)。選擇 OAuth client ID (OAuth 用戶端 ID)。選取 Web application (Web 應用程式) 做為 Application type (應用程式類型)。為開發應用程式所在的每個平台建立單獨的用戶端 ID。
4. 從 Credentials (憑證)，選擇 Manage service accounts (管理服務帳戶)。選擇 Create service account (建立服務帳戶)。輸入您的服務帳戶詳細資訊，然後選擇 Create and continue (建立並繼續)。
5. 將您專案的存取權授與服務帳戶。根據您應用程式的需求將服務帳戶的存取權授與使用者。
6. 選擇您的新服務帳戶，選擇 Keys (金鑰) 索引標籤，然後選擇 Add key (新增金鑰)。建立並下載新的 JSON 金鑰。

如需使用 Google 開發人員主控台的詳細資訊，請參閱 Google Cloud 文件中的建立與管理專案。

如需如何將 Google 整合到您 Web 應用程式的詳細資訊，請參閱 Google Identity 文件中的使用 Google 登入。

在 Amazon Cognito 主控台中設定外部供應商

若要新增 Google 身分提供者 (IdP)

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。
3. 選取 新增身分供應商。
4. 選擇 Google。
5. 輸入您在 Google Cloud Platform 建立的 OAuth 專案 用戶端 ID。如需詳細資訊，請參閱《Google Cloud Platform 主控台幫助》中的設定 OAuth 2.0。
6. 若要設定 Amazon Cognito 向已通過此提供者進行身分驗證的使用者發布憑證時的角色，請設定角色設定。

    - 您可以為該 IdP 使用者指派設定 已驗證角色 時的 預設角色，或您可以 選擇具有規則的角色。

i. 如果您選擇 使用規則選擇角色，請輸入使用者身分驗證的 宣告 來源、比較宣告的 操作員、導致符合角色選擇的 值，以及當符合 角色指派 時您要指派的 角色。選取 新增另一項 以根據不同的條件建立其他規則。

ii. 選擇 角色解析。當您的使用者宣告與您的規則不符時，您可以拒絕憑證或向 已驗證角色發出憑證。

7. 若要變更透過此提供者驗證使用者，Amazon Cognito 發布憑證時指派的主要索引標籤，請設定 存取控制的屬性。

   a. 若不套用主要索引標籤，請選擇 非作用中。

   b. 若要根據 sub 和 aud 宣告套用主要索引標籤，請選擇 使用預設對應。

   c. 若要建立您自己的自訂屬性結構描述至主要索引標籤，請選擇 使用自訂對應。然後，輸入您要從每個 宣告 中獲取的 標籤金鑰，顯示於索引標籤當中。

8. 選取儲存變更。

使用 Google

若要在您的應用程式中啟用以 Google 登入，請遵循適用於 Web 的 Google 說明文件。

身分驗證成功會產生包含 id_token 的回應物件，Amazon Cognito 會用其來驗證使用者，並產生唯一識別符：

```
function signinCallback(authResult) {
  if (authResult['status']['signed_in']) {

     // Add the Google access token to the Amazon Cognito credentials login map.
     AWS.config.credentials = new AWS.CognitoIdentityCredentials({
        IdentityPoolId: 'IDENTITY_POOL_ID',
        Logins: {
           'accounts.google.com': authResult['id_token']
        }
     });

     // Obtain AWS credentials
     AWS.config.credentials.get(function(){
        // Access AWS resources here.
     });
  }
}
```

Unity

設定 Google

若要啟用適用於 Unity 應用程式的 Google 登入，您必須為應用程式建立 Google 開發人員主控台專案。

1. 前往 Google 開發人員主控台，並建立新的專案。
2. 選擇 APIs & Services (API 與服務)，然後選擇 OAuth consent screen (OAuth 同意畫面)。自訂當 Google 要求使用者同意將其設定檔資料與您的應用程式共享時，Google 要向使用者顯示的資訊。
3. 選擇 Credentials (憑證)，然後選擇 Create credentials (建立憑證)。選擇 OAuth client ID (OAuth 用戶端 ID)。選取 Web application (Web 應用程式) 做為 Application type (應用程式類型)。為開發應用程式所在的每個平台建立單獨的用戶端 ID。
4. 若為 Unity，則針對 Android 建立一個額外的 OAuth client ID (OAuth 用戶端 ID)，並針對 iOS 建立另外一個。
5. 從 Credentials (憑證)，選擇 Manage service accounts (管理服務帳戶)。選擇 Create service account (建立服務帳戶)。輸入您的服務帳戶詳細資訊，然後選擇 Create and continue (建立並繼續)。
6. 將您專案的存取權授與服務帳戶。根據您應用程式的需求將服務帳戶的存取權授與使用者。
7. 選擇您的新服務帳戶，選擇 Keys (金鑰) 索引標籤，然後選擇 Add key (新增金鑰)。建立並下載新的 JSON 金鑰。

如需使用 Google 開發人員主控台的詳細資訊，請參閱 Google Cloud 文件中的建立與管理專案。

在 IAM 主控台中建立 OpenID 供應商

1. 在 IAM 主控台中建立 OpenID 供應商 如需有關如何設定 OpenID 供應商的資訊，請參閱使用 OpenID Connect 身分提供者。
2. 當提示您輸入供應商 URL 時，請輸入 `"https://accounts.google.com"`。
3. 當提示您在 Audience (對象) 欄位中輸入值時，請輸入您在前面步驟中建立的三個用戶端 ID 之一。
4. 選擇供應商名稱，並使用其他兩個用戶端 ID 再新增其他兩個對象。

在 Amazon Cognito 主控台中設定外部供應商

從 Amazon Cognito 主控台首頁選擇 Manage Identity Pools (管理身分集區)：

若要新增 Google 身分提供者 (IdP)

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。
3. 選取 新增身分供應商。
4. 選擇 Google。
5. 輸入您在 Google Cloud Platform 建立的 OAuth 專案 用戶端 ID。如需詳細資訊，請參閱《Google Cloud Platform 主控台幫助》中的設定 OAuth 2.0。
6. 若要設定 Amazon Cognito 向已通過此提供者進行身分驗證的使用者發布憑證時的角色，請設定角色設定。

   - 您可以為該 IdP 使用者指派設定 已驗證角色 時的 預設角色，或您可以 選擇具有規則的角色。

     i. 如果您選擇 使用規則選擇角色，請輸入使用者身分驗證的 宣告 來源、比較宣告的 操作員、導致符合角色選擇的 值，以及當符合 角色指派 時您要指派的 角色。選取 新增另一項 以根據不同的條件建立其他規則。

     ii. 選擇 角色解析。當您的使用者宣告與您的規則不符時，您可以拒絕憑證或向 已驗證角色發出憑證。
7. 若要變更透過此提供者驗證使用者，Amazon Cognito 發布憑證時指派的主要索引標籤，請設定 存取控制的屬性。

   a. 若不套用主要索引標籤，請選擇 非作用中。

   b. 若要根據 sub 和 aud 宣告套用主要索引標籤，請選擇 使用預設對應。

   c. 若要建立您自己的自訂屬性結構描述至主要索引標籤，請選擇 使用自訂對應。然後，輸入您要從每個 宣告 中獲取的 標籤金鑰，顯示於索引標籤當中。
8. 選取儲存變更。

安裝 Unity Google 外掛程式

1. 將 Google Play Games plugin for Unity 新增至您的 Unity 專案。
2. 在 Unity 中，從 Windows 選單使用適用於 Android 和 iOS 平台的三個 ID 來設定外掛程式。

使用 Google

下列範例程式碼說明如何從 Google Play 服務擷取身分驗證權杖：

```
void Start()
{
  PlayGamesClientConfiguration config = new
 PlayGamesClientConfiguration.Builder().Build();
  PlayGamesPlatform.InitializeInstance(config);
  PlayGamesPlatform.DebugLogEnabled = true;
  PlayGamesPlatform.Activate();
  Social.localUser.Authenticate(GoogleLoginCallback);
}

void GoogleLoginCallback(bool success)
{
  if (success)
  {
    string token = PlayGamesPlatform.Instance.GetIdToken();
    credentials.AddLogin("accounts.google.com", token);
  }
  else
  {
    Debug.LogError("Google login failed. If you are not running in an actual Android/
iOS device, this is expected.");
  }
}
```

Xamarin

> **Note**
>
> Amazon Cognito 不是原本就在 Xamarin 平台上支援 Google。整合目前需要使用 Web 檢視畫面來進行瀏覽器登入流程。若要了解 Google 整合如何與其他軟體開發套件搭配使用，請選取其他平台。

若要在您的應用程式中啟用以 Google 登入，請驗證您的使用者，並從其取得 OpenID Connec 權杖。Amazon Cognito 會使用此權杖來產生與 Amazon Cognito 身分相關聯的唯一使用者識別符。很可惜，適用於 Xamarin 的 Google 開發套件不能讓您擷取 OpenID Connect 權杖，因此請使用替代用戶端，或是使用 Web 檢視中的 Web 流程。

取得權杖之後，您可以將其設定在您的 `CognitoAWSCredentials` 中：

```
credentials.AddLogin("accounts.google.com", token);
```

> **Note**
>
> 如果您的應用程式使用 Google 並可在多個行動平台上使用，則應將 Google 設定為 OpenID Connect 供應商。將所有建立的用戶端 ID 新增為其他對象值，以便達到更佳的整合。若要進一步了解 Google 的跨用戶端身分模型，請參閱跨用戶端身分。

設定以 Apple 身分識別集區 IdP 身分登入

Amazon Cognito 與 Sign in with Apple 整合可為您的行動應用程式和 Web 應用程式使用者提供聯合身分驗證。本節說明如何使用「使用 Apple 登入」做為身分提供者 (IdP) 來註冊及設定您的應用程式。

若要以身分驗證供應商身分將「使用 Apple 登入」新增至身分集區，您必須完成兩個程序。首先，在應用程式整合「使用 Apple 登入」，然後在身分集區中設定「使用 Apple 登入」。如需設定「使用 Apple 登入」的最多 up-to-date 資訊，請參閱 Apple 開發人員說明文件中的「設定使用 Apple 登入的環境」。

設定 Sign in with Apple

若要將「使用 Apple 登入」設定為 IdP，請向 Apple 註冊您的應用程式，以接收用戶端 ID。

1. 向 Apple 建立開發人員帳戶。
2. 使用您的 Apple 憑證登入。
3. 在左側導覽窗格中，選擇 Certificates, IDs & Profiles (憑證、ID 和描述檔)。
4. 在左側導覽窗格中，選擇 Identifiers (識別碼)。
5. 在 Identifiers (識別碼) 頁面上，選擇 + 圖示。
6. 在 Register a New Identifier (註冊新的識別碼) 頁面上，選擇 App IDs (應用程式 ID)，然後選擇 Continue (繼續)。
7. 在 Register an App ID (註冊應用程式 ID) 頁面上，執行下列操作：
   a. 在 Description (說明) 下方，輸入說明內容。
   b. 在 Bundle ID (套件 ID) 下，輸入識別碼。記下此 Bundle ID (套件 ID)，因為您需要此值才能將 Apple 設定為身分集區中的供應商。
   c. 在 Capabilities (功能) 下方，選擇 Sign In with Apple，然後選擇 Edit (編輯)。
   d. 在 使用 Apple 登入：應用程式 ID 組態 頁面上，針對您的應用程式選取適當設定。然後選擇 Save (儲存)。
   e. 選擇繼續。

8. 在 Confirm your App ID (確認您的應用程式 ID) 頁面上，選擇 Register (註冊)。
9. 如果您想要將 Sign in with Apple 與原生 iOS 應用程式整合，請繼續進行步驟 10。步驟 11 適用於您想要與 Sign in wih Apple JS 整合的應用程式。
10. 在 Identifiers (識別碼) 頁面中，選擇 App IDs (應用程式 ID) 選單，然後選擇 Services IDs (服務 ID)。選擇 + 圖示。
11. 在 Register a New Identifier (註冊新的識別碼) 頁面上，選擇 Services IDs (服務 ID)，然後選擇 Continue (繼續)。
12. 在 Register a Services ID (註冊服務 ID) 頁面上，執行下列操作：

    a. 在 Description (說明) 下方，輸入說明內容。

    b. 在 Identifier (識別碼) 下方，輸入識別碼。記下服務 ID，因為您需要此數值才能將 Apple 設定為身分集區中的供應商。

    c. 選取 Sign In with Apple，然後選擇 Configure (設定)。

    d. 在 Web Authentication Configuration (Web 驗證組態) 頁面上，選擇 Primary App ID (主要應用程式 ID)。在 Website URL's (網站 URL) 下方，選擇 + 圖示。在 Domains and Subdomains (網域和子網域) 中，輸入應用程式的網域名稱。在 Return URLs (傳回 URL) 中，輸入一個回呼 URL，當使用者在透過「使用 Apple 登入」驗證身分後，該授權會在此 URL 重新導向使用者。

    e. 選擇下一步。

    f. 選擇 Continue (繼續)，然後選擇 Register (註冊)。
13. 在左側導覽窗格中，選擇 Keys (金鑰)。
14. 在 Keys (金鑰) 頁面上，選擇 + 圖示。
15. 在 Register a New Key (註冊新的金鑰) 頁面上，執行下列操作：

    a. 在 Key Name (金鑰名稱) 底下，輸入金鑰名稱。

    b. 選擇 Sign In with Apple，然後選擇 Configure (設定)。

    c. 在 Configure Key (設定金鑰) 頁面上，選擇 Primary App ID (主要應用程式 ID)，然後選擇 Save (儲存)。

    d. 選擇 Continue (繼續)，然後選擇 Register (註冊)。

> **Note**
>
> 若要將 Sign in with Apple 與原生 iOS 應用程式整合，請參閱使用 Sign in with Apple 實作使用者身分驗證。

若要在原生 iOS 以外的平台中整合 Sign in with Apple，請參閱 Sign in with Apple JS。

在 Amazon Cognito 聯合身分主控台中設定外部供應商

使用下列程序設定您的外部供應商。

若要新增「使用 Apple 登入」身分提供者 (IdP)

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。
3. 選取 新增身分供應商。
4. 選擇 使用 Apple 登入。
5. 輸入您使用 Apple Developer 建立的 OAuth 專案 服務 ID。如需詳細資訊，請參閱《使用 Apple 文檔登入》中的使用 Apple 驗證登入使用者。
6. 若要設定 Amazon Cognito 向已通過此提供者進行身分驗證的使用者發布憑證時的角色，請設定角色設定。
   - 您可以為該 IdP 使用者指派設定 已驗證角色 時的 預設角色，或您可以 選擇具有規則的角色。
     i. 如果您選擇 使用規則選擇角色，請輸入使用者身分驗證的 宣告 來源、比較宣告的 操作員、導致符合角色選擇的 值，以及當符合 角色指派 時您要指派的 角色。選取 新增另一項 以根據不同的條件建立其他規則。
     ii. 選擇 角色解析。當您的使用者宣告與您的規則不符時，您可以拒絕憑證或向 已驗證角色發出憑證。
7. 若要變更透過此提供者驗證使用者，Amazon Cognito 發布憑證時指派的主要索引標籤，請設定 存取控制的屬性。
   a. 若不套用主要索引標籤，請選擇 非作用中。
   b. 若要根據 sub 和 aud 宣告套用主要索引標籤，請選擇 使用預設對應。
   c. 若要建立您自己的自訂屬性結構描述至主要索引標籤，請選擇 使用自訂對應。然後，輸入您要從每個 宣告 中獲取的 標籤金鑰，顯示於索引標籤當中。
8. 選取儲存變更。

Amazon Cognito 聯合身分 CLI 範例中以 Sign in with Apple 做為供應商

此範例會建立名為 MyIdentityPool 的身分集區，並以「使用 Apple 登入」做為 IdP。

```
aws cognito-identity create-identity-pool --identity-
pool-name MyIdentityPool --supported-login-providers
appleid.apple.com="sameple.apple.clientid"
```

如需詳細資訊，請參閱建立身分識別集區

產生 Amazon Cognito 身分 ID

此範例會產生 (或擷取) Amazon Cognito ID。這是一個公有 API，因此您不需要任何登入資料來呼叫此 API。

```
aws cognito-identity get-id --identity-pool-id SampleIdentityPoolId --
logins appleid.apple.com="SignInWithAppleIdToken"
```

如需詳細資訊，請參閱這裡。

取得 Amazon Cognito 身分 ID 的憑證

此範例會傳回所提供的身分 ID 和 Sign in with Apple 登入的登入資料。這是一個公有 API，因此您不需要任何登入資料來呼叫此 API。

```
aws cognito-identity get-credentials-for-identity --identity-id
SampleIdentityId --logins appleid.apple.com="SignInWithAppleIdToken"
```

如需詳細資訊，請參閱〈get-credentials-for-identity

使用 Sign in with Apple：Android系統

Apple 不提供支援適用於 Android 之 Sign in with Apple 的開發套件。您可以改在 Web 檢視中使用 Web 流程。

- 若要在您的應用程式中設定 Sign in with Apple，請遵循 Apple 文件中的 Configuring Your Webpage for Sign In with Apple。
- 若要將 Sign in with Apple 按鈕新增至您的 Android 使用者介面，請遵循 Apple 文件中的 Displaying and Configuring Sign In with Apple Buttons。
- 若要使用「使用 Apple 登入」安全地對使用者進行身分驗證，請遵循 Apple 文件中的使用「使用 Apple 登入」對使用者進行身分驗證。

Sign in with Apple 會使用工作階段物件來追蹤其狀態。Amazon Cognito 使用此工作階段物件中的 ID 權杖來驗證使用者、產生唯一識別碼，並視需要授與使用者存取其他 AWS 資源。

```
@Override
public void onSuccess(Bundle response) {
    String token = response.getString("id_token");
    Map<String, String> logins = new HashMap<String, String>();
    logins.put("appleid.apple.com", token);
    credentialsProvider.setLogins(logins);
}
```

使用 Sign in with Apple：iOS - Objective-C

Apple 在原生 iOS 應用程式中提供對 Sign in with Apple 的 SDK 支援。若要在原生 iOS 裝置中使用 Sign in with Apple 來實作使用者身分驗證，請遵循 Apple 文件中的 Implementing User Authentication with Sign in with Apple。

Amazon Cognito 使用 ID 權杖來驗證使用者、產生唯一識別碼，並視需要授與使用者存取其他 AWS 資源。

```
(void)finishedWithAuth: (ASAuthorizationAppleIDCredential *)auth error: (NSError *)
 error {
        NSString *idToken = [ASAuthorizationAppleIDCredential
 objectForKey:@"identityToken"];
        credentialsProvider.logins = @{ "appleid.apple.com": idToken };
    }
```

使用 Sign in with Apple：iOS - Swift

Apple 在原生 iOS 應用程式中提供對 Sign in with Apple 的 SDK 支援。若要在原生 iOS 裝置中使用 Sign in with Apple 來實作使用者身分驗證，請遵循 Apple 文件中的 Implementing User Authentication with Sign in with Apple。

Amazon Cognito 使用 ID 權杖來驗證使用者、產生唯一識別碼，並視需要授與使用者存取其他 AWS 資源。

如需如何在 iOS 中設定「使用 Apple 登入」的詳細資訊，請參閱設定「使用 Apple 登入」

```
func finishedWithAuth(auth: ASAuthorizationAppleIDCredential!, error: NSError!) {
    if error != nil {
      print(error.localizedDescription)
```

```
    }
    else {
      let idToken = auth.identityToken,
      credentialsProvider.logins = ["appleid.apple.com": idToken!]
    }
}
```

使用「使用蘋果登入」：JavaScript

蘋果不提供支持「使用蘋果登錄」的 SDK JavaScript。您可以改在 Web 檢視中使用 Web 流程。

- 若要在您的應用程式中設定 Sign in with Apple，請遵循 Apple 文件中的 Configuring Your Webpage for Sign In with Apple。
- 若要將「使用 Apple 登入」按鈕新增至您的 JavaScript 使用者介面，請依照 Apple 文件中的「顯示和設定使用 Apple 按鈕登入」進行操作。
- 若要透過「使用 Apple 登入」安全地驗證使用者，請遵循 Apple 文件中的 Configuring Your Web page for Sign In with Apple。

Sign in with Apple 會使用工作階段物件來追蹤其狀態。Amazon Cognito 使用此工作階段物件中的 ID 權杖來驗證使用者、產生唯一識別碼，並視需要授與使用者存取其他 AWS 資源。

```
function signinCallback(authResult) {
     // Add the apple's id token to the Amazon Cognito credentials login map.
     AWS.config.credentials = new AWS.CognitoIdentityCredentials({
        IdentityPoolId: 'IDENTITY_POOL_ID',
        Logins: {
           'appleid.apple.com': authResult['id_token']
        }
     });

     // Obtain AWS credentials
     AWS.config.credentials.get(function(){
        // Access AWS resources here.
     });
}
```

使用 Sign in with Apple：Xamarin

我們沒有支援適用於 Xamarin 之 Sign in with Apple 的開發套件。您可以改在 Web 檢視中使用 Web 流程。

- 若要在您的應用程式中設定 Sign in with Apple，請遵循 Apple 文件中的 Configuring Your Webpage for Sign In with Apple。
- 若要將 Sign in with Apple 按鈕新增至您的 Xamarin 使用者介面，請遵循 Apple 文件中的 Displaying and Configuring Sign In with Apple Buttons。
- 若要透過「使用 Apple 登入」安全地驗證使用者，請遵循 Apple 文件中的 Configuring Your Web page for Sign In with Apple。

Sign in with Apple 會使用工作階段物件來追蹤其狀態。Amazon Cognito 使用此工作階段物件中的 ID 權杖來驗證使用者、產生唯一識別碼，並視需要授與使用者存取其他 AWS 資源。

取得權杖之後，您可以將其設定在您的 `CognitoAWSCredentials` 中：

```
credentials.AddLogin("appleid.apple.com", token);
```

將 OIDC 提供者設定為識別集區 IdP

OpenID Connect 是一種用來進行身分驗證的開放標準，許多登入供應商皆支援。Amazon Cognito 支援使用透過 AWS Identity and Access Management 設定的 OpenID Connect 供應商來連結身分。

新增 OpenID Connect 供應商

如需有關如何建立 OpenID Connect 供應商的詳細資訊，請參閱《AWS Identity and Access Management 使用者指南》中的建立 OpenID Connect (OIDC) 身分提供者。

將供應商與 Amazon Cognito 相關聯

若要新增 OIDC 身分提供者 (IdP)

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。
3. 選取 新增身分供應商。
4. 選擇 OpenID Connect (OIDC)。
5. 從您的. IdPs 中的 IAM 選擇 OIDC 身分識別提供者。 AWS 帳戶如果要新增新的 SAML 提供者，請選擇 建立新的提供者 以前往 IAM 主控台。
6. 若要設定 Amazon Cognito 向已通過此提供者進行身分驗證的使用者發布憑證時的角色，請設定角色設定。

- 您可以為該 IdP 使用者指派設定 已驗證角色 時的 預設角色，或您可以 選擇具有規則的角色。

  i. 如果您選擇 使用規則選擇角色，請輸入使用者身分驗證的 宣告 來源、比較宣告的 操作員、導致符合角色選擇的 值，以及當符合 角色指派 時您要指派的 角色。選取 新增另一項 以根據不同的條件建立其他規則。

  ii. 選擇 角色解析。當您的使用者宣告與您的規則不符時，您可以拒絕憑證或向 已驗證角色發出憑證。

7. 若要變更透過此提供者驗證使用者，Amazon Cognito 發布憑證時指派的主要索引標籤，請設定 存取控制的屬性。

   a. 若不套用主要索引標籤，請選擇 非作用中。

   b. 若要根據 `sub` 和 `aud` 宣告套用主要索引標籤，請選擇 使用預設對應。

   c. 若要建立您自己的自訂屬性結構描述至主要索引標籤，請選擇 使用自訂對應。然後，輸入您要從每個 宣告 中獲取的 標籤金鑰，顯示於索引標籤當中。

8. 選取儲存變更。

您可以將多個 OpenID Connect 供應商與單一身分集區相關聯。

使用 OpenID Connect

請參閱供應商的說明文件，以了解如何登入及接收 ID 權杖。

在您擁有權杖後，將該權杖新增至登入對映。使用供應商的 URI 作為金鑰。

驗證 OpenID Connect 權杖

第一次與 Amazon Cognito 整合時，您可能會收到 `InvalidToken` 例外狀況。請務必了解 Amazon Cognito 如何驗證 OpenID Connect (OIDC) 權杖。

> **Note**
>
> 本文指出 (https://tools.ietf.org/html/rfc7523)，Amazon Cognito 提供 5 分鐘的寬限期來處理系統間的所有時脈偏移。

1. `iss` 參數必須符合登入對應所使用的金鑰 (例如 login.provider.com)。

2. 這個簽章必須有效。簽章必須能夠透過 RSA 公開金鑰來驗證。
3. 憑證公有金鑰的指紋符合您在建立 OIDC 供應商時在 IAM 中設定的指紋。
4. 如果 azp 參數存在，請對照 OIDC 供應商中列出的用戶端 ID 來檢查這個值。
5. 如果 azp 參數不存在，請對照 OIDC 供應商中列出的用戶端 ID 來檢查 aud 參數。

jwt.io 網站是很寶貴的資源，您可使用此資源將權杖解碼並驗證這些值。

Android

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("login.provider.com", token);
credentialsProvider.setLogins(logins);
```

iOS - Objective-C

```
credentialsProvider.logins = @{ "login.provider.com": token }
```

iOS - Swift

若要將 OIDC ID 權杖提供給 Amazon Cognito，請實作 `AWSIdentityProviderManager` 協定。

當實作 `logins` 方法時，請傳回包含您所設定 OIDC 供應商名稱的字典。這個字典可做為金鑰，並以來自已驗證使用者的現行 ID 權杖做為數值，如下列程式碼範例所示。

```
class OIDCProvider: NSObject, AWSIdentityProviderManager {
    func logins() -> AWSTask<NSDictionary> {
        let completion = AWSTaskCompletionSource<NSString>()
        getToken(tokenCompletion: completion)
        return completion.task.continueOnSuccessWith { (task) -> AWSTask<NSDictionary>?
 in
            //login.provider.name is the name of the OIDC provider as setup in the
 Amazon Cognito console
            return AWSTask(result:["login.provider.name":task.result!])
        } as! AWSTask<NSDictionary>

    }

    func getToken(tokenCompletion: AWSTaskCompletionSource<NSString>) -> Void {
        //get a valid oidc token from your server, or if you have one that hasn't
 expired cached, return it
```

```
        //TODO code to get token from your server
        //...

        //if error getting token, set error appropriately
        tokenCompletion.set(error:NSError(domain: "OIDC Login", code: -1 , userInfo:
 ["Unable to get OIDC token" : "Details about your error"]))
        //else
        tokenCompletion.set(result:"result from server id token")
    }
}
```

當您實例化時AWSCognitoCredentialsProvider，請在建構函式中傳遞實AWSIdentityProviderManager作為值 identityProviderManager的類別。如需詳細資訊，請前往AWSCognitoCredentialsProvider參考頁面，然後選擇「initWithRegion類型:identityPoolId:」identityProviderManager。

JavaScript

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
 IdentityPoolId: 'IDENTITY_POOL_ID',
 Logins: {
    'login.provider.com': token
 }
});
```

Unity

```
credentials.AddLogin("login.provider.com", token);
```

Xamarin

```
credentials.AddLogin("login.provider.com", token);
```

將 SAML 提供者設定為身分識別集區 IdP

Amazon Cognito 透過安全聲明標記語言 2.0 (SAML 2.0IdPs) 支援身分提供者進行身分驗證 ()。您可以將支援 SAML 的 IdP 用於 Amazon Cognito，以為您的使用者提供簡單的登入流程。支援 SAML 的 IdP 會指定您的使用者可擔任的 IAM 角色。如此一來，不同的用戶可以收到不同的許可集。

為 SAML IdP 設定您的身分集區

下列步驟說明如何設定您的身分集區來使用 SAML IdP。

> **Note**
>
> 在設定身分集區來支援 SAML 供應商之前，您必須先在 IAM 主控台中設定 SAML IdP。如需詳細資訊，請參閱《IAM 使用者指南》中的將第三方 SAML 解決方案供應商與 AWS整合。

若要新增 SAML 身分提供者 (IdP)

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。
3. 選取 新增身分供應商。
4. 選擇 SAML。
5. 從您 AWS 帳戶的. IdPs 中的 IAM 選擇 SAML 身分識別提供者。如果要新增新的 SAML 提供者，請選擇 建立新的提供者 以前往 IAM 主控台。
6. 若要設定 Amazon Cognito 向已通過此提供者進行身分驗證的使用者發布憑證時的角色，請設定角色設定。

   - 您可以為該 IdP 使用者指派設定 已驗證角色 時的 預設角色，或您可以 選擇具有規則的角色。

     i. 如果您選擇 使用規則選擇角色，請輸入使用者身分驗證的 宣告 來源、比較宣告的 操作員、導致符合角色選擇的 值，以及當符合 角色指派 時您要指派的 角色。選取 新增另一項 以根據不同的條件建立其他規則。

     ii. 選擇 角色解析。當您的使用者宣告與您的規則不符時，您可以拒絕憑證或向 已驗證角色發出憑證。
7. 若要變更透過此提供者驗證使用者，Amazon Cognito 發布憑證時指派的主要索引標籤，請設定 存取控制的屬性。

   a. 若不套用主要索引標籤，請選擇 非作用中。

   b. 若要根據 sub 和 aud 宣告套用主要索引標籤，請選擇 使用預設對應。

   c. 若要建立您自己的自訂屬性結構描述至主要索引標籤，請選擇 使用自訂對應。然後，輸入您要從每個 宣告 中獲取的 標籤金鑰，顯示於索引標籤當中。
8. 選取儲存變更。

設定 SAML IdP

建立 SAML 供應商之後，請設定您的 SAML IdP，以在您的 IdP 與 AWS之間新增依賴方信任。對於許多 IdPs，您可以指定 IdP 可用來從 XML 文件讀取信賴憑證者資訊和憑證的 URL。對於 AWS，您可以使用 https://signin.aws.amazon.com/static/saml-metadata.xml。下一個步驟是設定來自 IdP 的 SAML 宣告回應，以填入需要的宣告。 AWS 如需宣告組態的詳細資訊，請參閱設定身分驗證回應的 SAML 聲明。

當您的 SAML IdP 在 SAML 中繼資料中包含多個簽署憑證時，登入時，如果 SAML 宣告與 SAML 中繼資料中的任何憑證相符，則您的使用者集區會判斷 SAML 聲明有效。

使用 SAML 來自訂您的使用者角色

搭配 Amazon Cognito Identity 使用 SAML，您可以為最終使用者自訂角色。Amazon Cognito 僅支援基於 SAML 之 IdP 的增強型流程。您不需要為身分集區指定已驗證或未驗證的角色，就能使用 SAML 型 IdP。`https://aws.amazon.com/SAML/Attributes/Role`宣告屬性會指定一或多組以逗號分隔的角色和供應商 ARN。這些都是使用者可以擔任的角色。您可以設定 SAML IdP 以根據 IdP 提供的使用者屬性資訊來填入角色屬性。如果您在 SAML 聲明中收到多個角色，則在呼叫 `customRoleArn` 時，應填入選擇性的 `getCredentialsForIdentity` 參數。如果該角色符合 SAML 聲明中宣告的角色，則使用者會擔任此 `customRoleArn`。

使用 SAML IdP 驗證使用者身分

若要與 SAML 型 IdP 建立聯合，請決定使用者起始登入的 URL。 AWS 同盟會使用 IDP 起始的登入。在 AD FS 2.0 中，URL 採用的格式為 `https://<fqdn>/adfs/ls/IdpInitiatedSignOn.aspx?loginToRp=urn:amazon:webservices`。

若要在 Amazon Cognito 中新增對 SAML IdP 的支援，必須先從您的 iOS 或 Android 應用程式，使用您的 SAML 身分提供者來驗證使用者身分。用來透過 SAML IdP 進行整合和身分驗證的代碼是 SAML 供應商所特有的。在您對使用者進行身分驗證之後，便可以使用 Amazon Cognito API 將所產生的 SAML 聲明提供給 Amazon Cognito Identity。

您無法在身分識別集區 API 要求的 `Logins` 對應中重複或重新顯示 SAML 宣告。重新顯示的 SAML 宣告具有重複先前 API 要求 ID 的宣告 ID。可以在對應中接受 SAML 宣告的 API 作業包括GetId、GetCredentialsForIdentityGetOpenIdToken、和GetOpen識別碼。`Logins` `TokenForDeveloperIdentity`您可以在身分識別集區驗證流程中的每個 API 要求重新顯示一次 SAML 宣告 ID。例如，您可以在 `GetId` 要求和後續 `GetCredentialsForIdentity` 要求中提供相同的 SAML 宣告，但不能在第二個 `GetId` 要求中提供。

Android

如果您是使用 Android 開發套件，您可以使用 SAML 聲明來填入登入對應，如下所示。

```
Map logins = new HashMap();
logins.put("arn:aws:iam::aws account id:saml-provider/name", "base64 encoded assertion
 response");
// Now this should be set to CognitoCachingCredentialsProvider object.
CognitoCachingCredentialsProvider credentialsProvider = new
 CognitoCachingCredentialsProvider(context, identity pool id, region);
credentialsProvider.setLogins(logins);
// If SAML assertion contains multiple roles, resolve the role by setting the custom
 role
credentialsProvider.setCustomRoleArn("arn:aws:iam::aws account id:role/
customRoleName");
// This should trigger a call to the Amazon Cognito service to get the credentials.
credentialsProvider.getCredentials();
```

iOS

如果您是使用 iOS 開發套件，您可以在 `AWSIdentityProviderManager`中提供 SAML 聲明，如下所示。

```
- (AWSTask<NSDictionary<NSString*,NSString*> *> *) logins {
    //this is hardcoded for simplicity, normally you would asynchronously go to your
 SAML provider
    //get the assertion and return the logins map using a AWSTaskCompletionSource
    return [AWSTask taskWithResult:@{@"arn:aws:iam::aws account id:saml-provider/
name":@"base64 encoded assertion response"}];
}

// If SAML assertion contains multiple roles, resolve the role by setting the custom
 role.
// Implementing this is optional if there is only one role.
- (NSString *)customRoleArn {
    return @"arn:aws:iam::accountId:role/customRoleName";
}
```

開發人員驗證的身分

除了透過 將臉書設定為身分集區 IdP、將谷歌設置為身份池 IdP、設置 Login with Amazon 作為身份池 IdP 和 設定以 Apple 身分識別集區 IdP 身分登入 的 Web 聯合身分，Amazon Cognito 還支援開發人員驗證的身分。透過開發人員驗證的身分，您可以透過自己現有的身分驗證程序註冊和驗證使用者，同時仍使用 Amazon Cognito 同步使用者資料和存取資源。 AWS 使用開發人員驗證的身分時，需要最終使用者裝置、身分驗證後端與 Amazon Cognito 之間的互動。如需詳細資訊，請參閱部 AWS 落格中的了解 Amazon Cognito 身分驗證第 2 部分：開發人員驗證身分。

了解身分驗證流程

GetOpenIdTokenForDeveloperIdentityAPI 操作可以為增強型和基本身份驗證啟動開發人員身份驗證。此 API 會使用系統管理認證來驗證要求。該Logins映射是身份池開發人員提供程序的名稱，如與自定義標識符login.mydevprovider配對。

範例：

```
"Logins": {
        "login.mydevprovider": "my developer identifier"
    }
```

增強型驗證

使用名稱cognito-identity.amazonaws.com和來源令牌值的Logins映射調用GetCredentialsForIdentityAPI 操作GetOpenIdTokenForDeveloperIdentity。

範例：

```
"Logins": {
        "cognito-identity.amazonaws.com": "eyJra12345EXAMPLE"
    }
```

GetCredentialsForIdentity使用開發人員驗證的身分會傳回身分識別集區預設驗證角色的臨時認證。

基本驗證

呼叫 AssumeRoleWithWebIdentityAPI 操作，並請求任何已定義適當信任關係RoleArn的 IAM 角色。WebIdentityToken將的值設定為從中取得的記號GetOpenIdTokenForDeveloperIdentity。

如需開發人員驗證身分 authflow 的相關資訊，以及它們與外部提供者身分識別有何不同，請參閱。身分集區 (聯合身分) 驗證流程

定義開發供應商名稱，並將其與身分集區建立關聯

若要使用開發人員驗證的身分，您需要有與開發供應商相關聯的身分池。若要執行此作業，請遵循下列步驟：

若要新增自訂開發供應商

1. 從 Amazon Cognito 主控台選擇 身分池。選取身分池。
2. 選擇 使用者存取權 索引標籤。
3. 選取 新增身分供應商。
4. 選擇 自訂開發人員提供者。
5. 輸入 開發人員供應商名稱。新增開發供應商之後，您無法變更或刪除。
6. 選取儲存變更。

備註：一旦設定供應商名稱，就無法再變更。

如需使用 Amazon Cognito 主控台的其他指示，請參閱使用 Amazon Cognito 主控台。

實作身分提供者

Android

若要使用開發人員驗證的身分，請實作您自己的身分供應商類別，擴展 `AWSAbstractCognitoIdentityProvider`。您的身分供應商類別應傳回包含字符做為屬性的回應物件。

下列是基礎身分提供者範例。

```
public class DeveloperAuthenticationProvider extends
 AWSAbstractCognitoDeveloperIdentityProvider {

  private static final String developerProvider = "<Developer_provider_name>";

  public DeveloperAuthenticationProvider(String accountId, String identityPoolId,
 Regions region) {
    super(accountId, identityPoolId, region);
    // Initialize any other objects needed here.
```

```
}

// Return the developer provider name which you choose while setting up the
// identity pool in the &COG; Console

@Override
public String getProviderName() {
  return developerProvider;
}

// Use the refresh method to communicate with your backend to get an
// identityId and token.

@Override
public String refresh() {

  // Override the existing token
  setToken(null);

  // Get the identityId and token by making a call to your backend
  // (Call to your backend)

  // Call the update method with updated identityId and token to make sure
  // these are ready to be used from Credentials Provider.

  update(identityId, token);
  return token;

}

// If the app has a valid identityId return it, otherwise get a valid
// identityId from your backend.

@Override
public String getIdentityId() {

  // Load the identityId from the cache
  identityId = cachedIdentityId;

  if (identityId == null) {
     // Call to your backend
  } else {
     return identityId;
  }
```

```
  }
}
```

若要使用此身分提供者，您必須將其傳入 CognitoCachingCredentialsProvider。範例如下：

```
DeveloperAuthenticationProvider developerProvider = new
 DeveloperAuthenticationProvider( null, "IDENTITYPOOLID", context, Regions.USEAST1);
CognitoCachingCredentialsProvider credentialsProvider = new
 CognitoCachingCredentialsProvider( context, developerProvider, Regions.USEAST1);
```

iOS - Objective-C

若要使用開發人員驗證的身分，請實作您自己的身分供應商類別，擴展 AWSCognitoCredentialsProviderHelper。您的身分提供者類別應傳回包含權杖做為屬性的回應物件。

```
@implementation DeveloperAuthenticatedIdentityProvider
/*
 * Use the token method to communicate with your backend to get an
 * identityId and token.
 */

- (AWSTask <NSString*> *) token {
    //Write code to call your backend:
    //Pass username/password to backend or some sort of token to authenticate user
    //If successful, from backend call getOpenIdTokenForDeveloperIdentity with logins
 map
    //containing "your.provider.name":"enduser.username"
    //Return the identity id and token to client
    //You can use AWSTaskCompletionSource to do this asynchronously

    // Set the identity id and return the token
    self.identityId = response.identityId;
    return [AWSTask taskWithResult:response.token];
}

@end
```

若要使用此身分提供者，請將其傳入 AWSCognitoCredentialsProvider，如下列範例所示。

```
DeveloperAuthenticatedIdentityProvider * devAuth =
 [[DeveloperAuthenticatedIdentityProvider alloc]
 initWithRegionType:AWSRegionYOUR_IDENTITY_POOL_REGION
                                          identityPoolId:@"YOUR_IDENTITY_POOL_ID"
                                         useEnhancedFlow:YES
                                identityProviderManager:nil];
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider
 alloc]

 initWithRegionType:AWSRegionYOUR_IDENTITY_POOL_REGION
                                                        identityProvider:devAuth];
```

如果您想要同時支援未驗證的身分和開發人員驗證的身分，請覆寫 `logins` 實作中的 `AWSCognitoCredentialsProviderHelper` 方法。

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else{
        return [super logins];
    }
}
```

如果您想要支援開發人員驗證的身分和社交供應商，您必須在 `logins` 的 `AWSCognitoCredentialsProviderHelper` 實作中，管理誰是目前的供應商。

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else if (/*logic to determine if user is Facebook*/){
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook :
 [FBSDKAccessToken currentAccessToken] }];
    }else {
        return [super logins];
    }
}
```

iOS - Swift

若要使用開發人員驗證的身分，請實作您自己的身分供應商類別，擴展 AWSCognitoCredentialsProviderHelper。您的身分提供者類別應傳回包含權杖做為屬性的回應物件。

```
import AWSCore
/*
 * Use the token method to communicate with your backend to get an
 * identityId and token.
 */
class DeveloperAuthenticatedIdentityProvider : AWSCognitoCredentialsProviderHelper {
    override func token() -> AWSTask<NSString> {
    //Write code to call your backend:
    //pass username/password to backend or some sort of token to authenticate user, if
 successful,
    //from backend call getOpenIdTokenForDeveloperIdentity with logins map containing
 "your.provider.name":"enduser.username"
    //return the identity id and token to client
    //You can use AWSTaskCompletionSource to do this asynchronously

    // Set the identity id and return the token
    self.identityId = resultFromAbove.identityId
    return AWSTask(result: resultFromAbove.token)
}
```

若要使用此身分提供者，請將其傳入 AWSCognitoCredentialsProvider，如下列範例所示。

```
let devAuth =
 DeveloperAuthenticatedIdentityProvider(regionType: .YOUR_IDENTITY_POOL_REGION,
 identityPoolId: "YOUR_IDENTITY_POOL_ID", useEnhancedFlow: true,
 identityProviderManager:nil)
let credentialsProvider =
 AWSCognitoCredentialsProvider(regionType: .YOUR_IDENTITY_POOL_REGION,
 identityProvider:devAuth)
let configuration = AWSServiceConfiguration(region: .YOUR_IDENTITY_POOL_REGION,
 credentialsProvider:credentialsProvider)
AWSServiceManager.default().defaultServiceConfiguration = configuration
```

如果您想要同時支援未驗證的身分和開發人員驗證的身分，請覆寫 logins 實作中的 AWSCognitoCredentialsProviderHelper 方法。

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else {
        return super.logins()
    }
```

```
}
```

如果您想要支援開發人員驗證的身分和社交供應商，您必須在 logins 的 AWSCognitoCredentialsProviderHelper 實作中，管理誰是目前的供應商。

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else if (/*logic to determine if user is Facebook*/){
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }else {
        return super.logins()
    }
}
```

JavaScript

從後端取得身分 ID 和工作階段字符後，您必須將其傳遞至 AWS.CognitoIdentityCredentials 供應商中。範例如下。

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
   IdentityPoolId: 'IDENTITY_POOL_ID',
   IdentityId: 'IDENTITY_ID_RETURNED_FROM_YOUR_PROVIDER',
   Logins: {
      'cognito-identity.amazonaws.com': 'TOKEN_RETURNED_FROM_YOUR_PROVIDER'
   }
});
```

Unity

若要使用開發人員驗證的身分，您必須擴展 CognitoAWSCredentials，並覆寫 RefreshIdentity 方法，以從後端擷取使用者身分 ID 和字符，並將其傳回。下列是身分供應商聯絡假設後端 'example.com' 的簡單範例：

```
using UnityEngine;
using System.Collections;
using Amazon.CognitoIdentity;
using System.Collections.Generic;
```

```
using ThirdParty.Json.LitJson;
using System;
using System.Threading;

public class DeveloperAuthenticatedCredentials : CognitoAWSCredentials
{
    const string PROVIDER_NAME = "example.com";
    const string IDENTITY_POOL = "IDENTITY_POOL_ID";
    static readonly RegionEndpoint REGION = RegionEndpoint.USEast1;

    private string login = null;

    public DeveloperAuthenticatedCredentials(string loginAlias)
        : base(IDENTITY_POOL, REGION)
    {
        login = loginAlias;
    }

    protected override IdentityState RefreshIdentity()
    {
        IdentityState state = null;
        ManualResetEvent waitLock = new ManualResetEvent(false);
        MainThreadDispatcher.ExecuteCoroutineOnMainThread(ContactProvider((s) =>
        {
            state = s;
            waitLock.Set();
        }));
        waitLock.WaitOne();
        return state;
    }

    IEnumerator ContactProvider(Action<IdentityState> callback)
    {
        WWW www = new WWW("http://example.com/?username="+login);
        yield return www;
        string response = www.text;

        JsonData json = JsonMapper.ToObject(response);

        //The backend has to send us back an Identity and a OpenID token
        string identityId = json["IdentityId"].ToString();
        string token = json["Token"].ToString();
```

```
        IdentityState state = new IdentityState(identityId, PROVIDER_NAME, token,
 false);
        callback(state);
    }
}
```

上述程式碼使用執行緒發送器物件來呼叫 coroutine。如果無法在您的專案中這麼做，您可以在您的場景中使用下列指令碼：

```
using System;
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class MainThreadDispatcher : MonoBehaviour
{
    static Queue<IEnumerator> _coroutineQueue = new Queue<IEnumerator>();
    static object _lock = new object();

    public void Update()
    {
        while (_coroutineQueue.Count > 0)
        {
            StartCoroutine(_coroutineQueue.Dequeue());
        }
    }

    public static void ExecuteCoroutineOnMainThread(IEnumerator coroutine)
    {
        lock (_lock) {
            _coroutineQueue.Enqueue(coroutine);
        }
    }
}
```

Xamarin

若要使用開發人員驗證的身分，您必須擴展 `CognitoAWSCredentials`，並覆寫 `RefreshIdentity` 方法，以從後端擷取使用者身分 ID 和字符，並將其傳回。下列是身分供應商聯絡假設後端 'example.com' 的基礎範例：

```
public class DeveloperAuthenticatedCredentials : CognitoAWSCredentials
```

```
{
    const string PROVIDER_NAME = "example.com";
    const string IDENTITY_POOL = "IDENTITY_POOL_ID";
    static readonly RegionEndpoint REGION = RegionEndpoint.USEast1;
    private string login = null;

    public DeveloperAuthenticatedCredentials(string loginAlias)
        : base(IDENTITY_POOL, REGION)
    {
        login = loginAlias;
    }

    protected override async Task<IdentityState> RefreshIdentityAsync()
    {
        IdentityState state = null;
        //get your identity and set the state
        return state;
    }
}
```

更新登入映射 (僅限 Android 和 iOS)

Android

在向您的身分驗證系統成功驗證使用者之後，請更新登入對應中的開發人員供應商名稱和開發使用者識別符。開發使用者識別符是英數字串，可唯一識別的身分驗證系統中的使用者。更新登入後，請務必呼叫 `refresh`方法，因為 `identityId` 對應可能有所變更：

```
HashMap<String, String> loginsMap = new HashMap<String, String>();
loginsMap.put(developerAuthenticationProvider.getProviderName(),
 developerUserIdentifier);

credentialsProvider.setLogins(loginsMap);
credentialsProvider.refresh();
```

iOS - Objective-C

除非沒有登入資料，或登入資料已過期，iOS 軟體開發套件才會呼叫 `logins`方法來取得最新的登入對應。如果您想要強制該軟體開發套件取得新的登入資料 (例如，最終使用者從未驗證變成已驗證，而您想要已驗證使用者的登入資料)，請在 `clearCredentials` 上呼叫 `credentialsProvider`。

```
[credentialsProvider clearCredentials];
```

iOS - Swift

除非沒有登入資料，或登入資料已過期，iOS 軟體開發套件才會呼叫 `logins`方法來取得最新的登入對應。如果您想要強制該軟體開發套件取得新的登入資料 (例如，最終使用者從未驗證變成已驗證，而您想要已驗證使用者的登入資料)，請在 `clearCredentials`上呼叫 `credentialsProvider`。

```
credentialsProvider.clearCredentials()
```

取得權杖 (伺服器端)

您可以透過呼叫取得權杖[GetOpenIdTokenForDeveloperIdentity](#)。必須使用 AWS 開發人員憑據從後端調用此 API。不得從用戶端軟體開發套件叫用。API 會收到 Cognito 身分池 ID、登入對應 (其中包含的索引鍵為您的身分供應商名稱，值為識別符)，並選擇性地收到 Cognito 身分 ID (例如，您要使未驗證的使用者變成已驗證)。識別符可以是使用者的使用者名稱、電子郵件地址或數值。API 回應您的呼叫時，會包含您的使用者的唯一 Cognito ID，以及最終使用者的 OpenID Connect 權杖。

關於 `GetOpenIdTokenForDeveloperIdentity`傳回的權杖，請記得幾件事：

- 您可以指定權杖的自訂過期時間，以便快取權杖。如果您未提供任何自訂過期時間，權杖是有效期限為 15 分鐘。
- 您可以設定的權杖持續時間上限為 24 小時。
- 請留意增加權杖持續時間的安全性影響。如果攻擊者獲得此令牌，他們可以將其交換為令牌持續時間的最終用戶的 AWS 憑據。

下列 Java 程式碼片段說明如何初始化 Amazon Cognito 用戶端，以及擷取開發人員驗證身分的權杖。

```
// authenticate your end user as appropriate
// ....

// if authenticated, initialize a cognito client with your AWS developer credentials
AmazonCognitoIdentity identityClient = new AmazonCognitoIdentityClient(
  new BasicAWSCredentials("access_key_id", "secret_access_key")
);

// create a new request to retrieve the token for your end user
```

```
GetOpenIdTokenForDeveloperIdentityRequest request =
  new GetOpenIdTokenForDeveloperIdentityRequest();
request.setIdentityPoolId("YOUR_COGNITO_IDENTITY_POOL_ID");

request.setIdentityId("YOUR_COGNITO_IDENTITY_ID"); //optional, set this if your client
 has an
                                                   //identity ID that you want to link
 to this
                                                   //developer account

// set up your logins map with the username of your end user
HashMap<String,String> logins = new HashMap<>();
logins.put("YOUR_IDENTITY_PROVIDER_NAME","YOUR_END_USER_IDENTIFIER");
request.setLogins(logins);

// optionally set token duration (in seconds)
request.setTokenDuration(60 * 15l);
GetOpenIdTokenForDeveloperIdentityResult response =
  identityClient.getOpenIdTokenForDeveloperIdentity(request);

// obtain identity id and token to return to your client
String identityId = response.getIdentityId();
String token = response.getToken();

//code to return identity id and token to client
//...
```

按照上述步驟，您應該能夠將開發人員驗證的身分整合在您的應用程式。如果您有任何問題，請隨時張貼在我們的論壇中。

連接至現有的社交身分

當您使用開發人員驗證的身分時，供應商的所有連結都必須從您的後端完成。要將自定義身份連接到用戶的社交身份（Login with Amazon，使用 Apple，Facebook 或 Google 登錄），請在呼叫GetOpenIdTokenForDeveloperIdentity時將身份提供者令牌添加到登錄地圖中。為了做到這一點，當您從用戶端軟體開發套件呼叫後端來驗證最終使用者時，請另外傳遞最終使用者的社交供應商權杖。

例如，如果您嘗試將自訂身分連結至 Facebook，當您呼叫 `GetOpenIdTokenForDeveloperIdentity`時，除了身分提供者識別符，也要將 Facebook 權杖新增至登入對應。

```
logins.put("YOUR_IDENTITY_PROVIDER_NAME","YOUR_END_USER_IDENTIFIER");
```

```
logins.put("graph.facebook.com","END_USERS_FACEBOOK_ACCESSTOKEN");
```

支援供應商之間的轉換

Android

您的應用程式可能需要支援未驗證的身分或使用公有身分供應商 (Login with Amazon、Sign in with Apple、Facebook 或 Google) 驗證的身分，以及開發人員驗證的身分。開發人員驗證的身分與其他身分 (未驗證的身分和使用公有供應商驗證的身分) 之間的基本差異，在於取得 identityId 和字符的方式。針對其他身分，行動應用程式會直接與 Amazon Cognito 互動，而不會聯絡您的身分驗證系統。所以行動應用程式應該能夠依據應用程式使用者的選擇，支援兩種不同的流程。因此，您必須對自訂身分供應商進行一些變更。

`refresh` 方法檢查登錄映射。如果映射不是空的，且擁有開發人員供應商名稱的金鑰，則會呼叫後端。否則，調用該 getIdentityId方法並返回 null。

```
public String refresh() {

   setToken(null);

   // If the logins map is not empty make a call to your backend
   // to get the token and identityId
   if (getProviderName() != null &&
      !this.loginsMap.isEmpty() &&
      this.loginsMap.containsKey(getProviderName())) {

      /**
       * This is where you would call your backend
       **/

      // now set the returned identity id and token in the provider
      update(identityId, token);
      return token;

   } else {
      // Call getIdentityId method and return null
      this.getIdentityId();
      return null;
   }
}
```

同樣地，依據登入對應的內容，`getIdentityId`方法也會有兩個流程：

```
public String getIdentityId() {

   // Load the identityId from the cache
   identityId = cachedIdentityId;

   if (identityId == null) {

      // If the logins map is not empty make a call to your backend
      // to get the token and identityId

      if (getProviderName() != null && !this.loginsMap.isEmpty()
         && this.loginsMap.containsKey(getProviderName())) {

         /**
           * This is where you would call your backend
          **/

         // now set the returned identity id and token in the provider
         update(identityId, token);
         return token;

      } else {
         // Otherwise call &COG; using getIdentityId of super class
         return super.getIdentityId();
      }

   } else {
      return identityId;
   }

}
```

iOS - Objective-C

您的應用程式可能需要支援未驗證的身分或使用公有身分供應商 (Login with Amazon、Sign in with Apple、Facebook 或 Google) 驗證的身分，以及開發人員驗證的身分。若要這麼做，請覆寫方AWSCognitoCredentialsProviderHelper`logins`法，以便能夠根據目前的身分識別提供者傳回正確的登入對應。此範例說明如何在未驗證、Facebook 和開發人員驗證的身分之間轉換。

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
```

```
    }else if (/*logic to determine if user is Facebook*/){
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook :
 [FBSDKAccessToken currentAccessToken] }];
    }else {
        return [super logins];
    }
}
```

當您從未驗證轉換至已驗證時，應呼叫 `[credentialsProvider clearCredentials];`，以強制軟體開發套件取得新的已驗證登入資料。如果您在兩個已驗證的供應商之間切換，而不嘗試將這兩個供應商連結 (例如，您不要在登入字典中提供多個供應商的權杖)，則應呼叫 `[credentialsProvider clearKeychain];`。這樣會同時清除登入資料和身分，並強制軟體開發套件取得新的登入資料和身分。

iOS - Swift

您的應用程式可能需要支援未驗證的身分或使用公有身分供應商 (Login with Amazon、Sign in with Apple、Facebook 或 Google) 驗證的身分，以及開發人員驗證的身分。若要這麼做，請覆寫方AWSCognitoCredentialsProviderHelper`logins`法，以便能夠根據目前的身分識別提供者傳回正確的登入對應。此範例說明如何在未驗證、Facebook 和開發人員驗證的身分之間轉換。

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else if (/*logic to determine if user is Facebook*/){
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }else {
        return super.logins()
    }
}
```

當您從未驗證轉換至已驗證時，應呼叫 `credentialsProvider.clearCredentials()`，以強制軟體開發套件取得新的已驗證登入資料。如果您在兩個已驗證的供應商之間切換，而不嘗試將這兩個供應商連結 (也就是說，您不要在登入字典中提供多個供應商的權杖)，則應呼叫 `credentialsProvider.clearKeychain()`。這樣會同時清除登入資料和身分，並強制軟體開發套件取得新的登入資料和身分。

Unity

您的應用程式可能需要支援未驗證的身分或使用公有身分供應商 (Login with Amazon、Sign in with Apple、Facebook 或 Google) 驗證的身分，以及開發人員驗證的身分。開發人員驗證的身分與其他身分 (未驗證的身分和使用公有供應商驗證的身分) 之間的基本差異，在於取得 identityId 和字符的方式。針對其他身分，行動應用程式會直接與 Amazon Cognito 互動，而不會聯絡您的身分驗證系統。行動應用程式應該能夠依據應用程式使用者的選擇，支援兩種不同的流程。因此，您必須對自訂身分供應商進行一些變更。

在 Unity 中執行此操作的推薦方法是從 AmazonCognitoEnhancedIdentityProvide 而不是擴展身份提供程序 AbstractCognitoIdentityProvider，並調用父 RefreshAsync 方法而不是您自己的方法，以防用戶未使用自己的後端進行身份驗證。如果使用者已驗證，則您可以使用先前說明的相同流程。

Xamarin

您的應用程式可能需要支援未驗證的身分或使用公有身分供應商 (Login with Amazon、Sign in with Apple、Facebook 或 Google) 驗證的身分，以及開發人員驗證的身分。開發人員驗證的身分與其他身分 (未驗證的身分和使用公有供應商驗證的身分) 之間的基本差異，在於取得 identityId 和字符的方式。針對其他身分，行動應用程式會直接與 Amazon Cognito 互動，而不會聯絡您的身分驗證系統。行動應用程式應該能夠依據應用程式使用者的選擇，支援兩種不同的流程。因此，您必須對自訂身分供應商進行一些變更。

將未進行身分驗證使用者切換到已進行身分驗證使用者 (身分集區)

Amazon Cognito 身分集區可支援已驗證和未驗證的使用者。未驗證的使用者即使沒有以任何身分提供者 (IdP) 路徑登入，也能存取您的 AWS 資源。這個程度的存取能在使用者登入前就顯示內容，非常有用。每個未驗證使用者在身分集區中都有專屬身分 (即使使用者尚未個別登入和驗證身分亦同)。

本節說明您的使用者選擇從未驗證身分登入，轉為使用經驗證身分登入的程序。

Android

使用者可使用未驗證的訪客身分來登入您的應用程式，最後他們可能會決定使用其中一個支援的 IdP 來登入。Amazon Cognito 確保舊身分保留與新身分相同的唯一識別符，且設定檔資料會自動合併。

設定檔合併的訊息會透過 `IdentityChangedListener` 界面來通知您的應用程式。請在該界面中實作 `identityChanged` 方法，以接收這些訊息：

```
@override
```

```
public void identityChanged(String oldIdentityId, String newIdentityId) {
    // handle the change
}
```

iOS - Objective-C

使用者可使用未驗證的訪客身分來登入您的應用程式，最後他們可能會決定使用其中一個支援的 IdP 來登入。Amazon Cognito 確保舊身分保留與新身分相同的唯一識別符，且設定檔資料會自動合併。

NSNotificationCenter 會通知您的應用程式設定檔合併的訊息：

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                      selector:@selector(identityIdDidChange:)
                                      name:AWSCognitoIdentityIdChangedNotification
                                      object:nil];

-(void)identityDidChange:(NSNotification*)notification {
    NSDictionary *userInfo = notification.userInfo;
    NSLog(@"identity changed from %@ to %@",
        [userInfo objectForKey:AWSCognitoNotificationPreviousId],
        [userInfo objectForKey:AWSCognitoNotificationNewId]);
}
```

iOS - Swift

使用者可使用未驗證的訪客身分來登入您的應用程式，最後他們可能會決定使用其中一個支援的 IdP 來登入。Amazon Cognito 確保舊身分保留與新身分相同的唯一識別符，且設定檔資料會自動合併。

NSNotificationCenter 會通知您的應用程式設定檔合併的訊息：

```
[NSNotificationCenter.defaultCenter().addObserver(observer: self
   selector:"identityDidChange"
   name:AWSCognitoIdentityIdChangedNotification
   object:nil)

func identityDidChange(notification: NSNotification!) {
  if let userInfo = notification.userInfo as? [String: AnyObject] {
    print("identity changed from: \(userInfo[AWSCognitoNotificationPreviousId])
    to: \(userInfo[AWSCognitoNotificationNewId])")
  }
}
```

JavaScript

最初未進行身分驗證的使用者

使用者通常在一開始會使用未驗證角色。針對此角色，您設定的組態物件之登入資料屬性不需登入屬性。在這種情況下，您的預設組態可能如下所示：

```
// set the default config object
var creds = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'us-east-1:1699ebc0-7900-4099-b910-2df94f52a030'
});
AWS.config.credentials = creds;
```

切換到已進行身分驗證使用者

當未驗證使用者登入 IdP，且您有一個權杖時，您可以透過呼叫自訂函數更新登入資料物件並新增登入權杖，將使用者從未驗證切換為已驗證：

```
// Called when an identity provider has a token for a logged in user
function userLoggedIn(providerName, token) {
    creds.params.Logins = creds.params.Logins || {};
    creds.params.Logins[providerName] = token;

    // Expire credentials to refresh them on the next request
    creds.expired = true;
}
```

您也可建立 `CognitoIdentityCredentials` 物件。如果您建立此物件，您必須重設任何現有的服務物件登入資料屬性，以反映經更新的登入資料組態資訊。請參閱使用全域組態物件。

如需 `CognitoIdentityCredentials` 物件的詳細資訊，請參閱 AWS SDK for JavaScript API 參考中的 AWS.CognitoIdentityCredentials。

Unity

使用者可使用未驗證的訪客身分來登入您的應用程式，最後他們可能會決定使用其中一個支援的 IdP 來登入。Amazon Cognito 確保舊身分保留與新身分相同的唯一識別符，且設定檔資料會自動合併。

您可以訂閱 `IdentityChangedEvent`，以接收設定檔合併的訊息：

```
credentialsProvider.IdentityChangedEvent += delegate(object sender,
 CognitoAWSCredentials.IdentityChangedArgs e)
{
    // handle the change
    Debug.log("Identity changed from " + e.OldIdentityId + " to " + e.NewIdentityId);
};
```

Xamarin

使用者可使用未驗證的訪客身分來登入您的應用程式，最後他們可能會決定使用其中一個支援的 IdP 來登入。Amazon Cognito 確保舊身分保留與新身分相同的唯一識別符，且設定檔資料會自動合併。

```
credentialsProvider.IdentityChangedEvent += delegate(object sender,
 CognitoAWSCredentials.IdentityChangedArgs e){
    // handle the change
    Console.WriteLine("Identity changed from " + e.OldIdentityId + " to " +
 e.NewIdentityId);
};
```

Amazon Cognito Sync

⚠ 如果您第一次使用 Amazon Cognito Sync，請改用 AWS AppSync。像 Amazon Cognito Sync 一樣，AWS AppSync 是讓您在裝置間同步應用程式資料的服務。
可同步使用者資料，如應用程式偏好設定或遊戲狀態。也擴充這些功能，允許多個使用者在共用資料上即時同步及協作。

Amazon Cognito Sync 是 AWS 服務 和用戶端程式庫，可讓您跨裝置同步應用程式相關的使用者資料。Amazon Cognito Sync 可跨行動裝置和 Web 同步使用者描述檔資料，無需使用自己的後端系統。用戶端程式庫會在本機快取資料，因此無論裝置的連線狀態為何，您的應用程式可以讀取和寫入資料。當裝置上線時，您可以同步資料。如果您設定推播同步，可立即通知其他裝置有更新可用。

如需 Amazon Cognito Identity 區域可用性的相關資訊，請參閱 AWS 服務區域可用性。

若要進一步了解 Amazon Cognito Sync，請參閱下列主題。

主題

Amazon Cognito Sync 入門

⚠ 如果您第一次使用 Amazon Cognito Sync，請改用 AWS AppSync。像 Amazon Cognito Sync 一樣，AWS AppSync 是讓您在裝置間同步應用程式資料的服務。
可同步使用者資料，如應用程式偏好設定或遊戲狀態。也擴充這些功能，允許多個使用者在共用資料上即時同步及協作。

Amazon Cognito Sync 是 AWS 服務和用戶端程式庫，可讓您跨裝置同步應用程式相關的使用者資料。您可以用它來同步各行動裝置和 Web 應用程式之間的使用者描述檔資料。用戶端程式庫會在本機快取資料，因此無論裝置的連線狀態為何，您的應用程式可以讀取和寫入資料。當裝置在線上時，您就可以同步資料，而且如果您設定推播同步，還會立即通知其他裝置有更新可用。

在 Amazon Cognito 中設定身分集區

Amazon Cognito Sync 需有 Amazon Cognito 身分集區才能提供使用者身分。您必須先設定身分集區，才能使用 Amazon Cognito Sync。若要建立身分集區和安裝 SDK，請參閱 開始使用 Amazon Cognito 身分集區。

存放及同步資料

設定身分集區並安裝 SDK 之後，您可以開始存放以及在裝置之間同步資料。如需更多詳細資訊，請參閱 同步資料。

同步資料

> ⚠ 如果您第一次使用 Amazon Cognito Sync，請改用 AWS AppSync。像 Amazon Cognito Sync 一樣，AWS AppSync 是讓您在裝置間同步應用程式資料的服務。
> 可同步使用者資料，如應用程式偏好設定或遊戲狀態。也擴充這些功能，允許多個使用者在共用資料上即時同步及協作。

使用 Amazon Cognito 可讓您將包含鍵值組的使用者資料儲存在資料集中。Amazon Cognito 將此資料與身分集區中的身分建立關聯，因此應用程式可以跨登入和裝置進行存取。若要在 Amazon Cognito 服務與最終使用者的裝置之間同步此資料，請呼叫同步方法。每個資料集的大小上限為 1 MB。一個身分最多可以與 20 個資料集相關聯。

Amazon Cognito Sync 用戶端會為身分資料建立本機快取。當您的應用程式讀取和寫入索引鍵時，會與此本機快取進行通訊。此通訊可確保您在裝置上進行的所有變更都可立即用在裝置上，即使在您離線時也一樣。呼叫同步方法時，會將服務的變更提取到裝置，而任何本機變更都會推播到服務。此時，變更即可供其他裝置進行同步。

初始化 Amazon Cognito Sync 用戶端

若要初始化 Amazon Cognito Sync 用戶端，您必須先建立憑證供應商。憑證供應商取得臨時 AWS 憑證，讓您的應用程式能夠存取 AWS 資源。您還必須匯入必要的標頭檔案。請使用下列步驟來初始化 Amazon Cognito Sync 用戶端。

Android

1. 遵循取得憑證中的指示，建立登入資料供應商。
2. 匯入 Amazon Cognito 套件如下：`import com.amazonaws.mobileconnectors.cognito.*;`
3. 初始化 Amazon Cognito Sync。傳入 Android 應用程式內容、身分集區 ID、AWS 區域，以及已初始化的 Amazon Cognito 憑證供應商，如下所示：

```
CognitoSyncManager client = new CognitoSyncManager(
    getApplicationContext(),
    Regions.YOUR_REGION,
    credentialsProvider);
```

iOS - Objective-C

1. 遵循取得憑證中的指示，建立登入資料供應商。
2. 匯入 `AWSCore` 和 `Cognito`，然後初始化 `AWSCognito`，如下所示：

```
#import <AWSiOSSDKv2/AWSCore.h>
#import <AWSCognitoSync/Cognito.h>

AWSCognito *syncClient = [AWSCognito defaultCognito];
```

3. 如果您使用的是 CocoaPods，請以 `AWSCore.h` 取代 `<AWSiOSSDKv2/AWSCore.h>`。在進行 Amazon Cognito 匯入時，請遵循相同的語法。

iOS - Swift

1. 遵循取得憑證中的指示，建立登入資料供應商。
2. 匯入並初始化 `AWSCognito`，如下所示：

```
import AWSCognito
let syncClient = AWSCognito.default()!
```

JavaScript

1. 下載 Amazon Cognito Sync Manager for JavaScript。
2. 將 Sync Manager 程式庫包含在您的專案中。
3. 遵循取得憑證中的指示，建立登入資料供應商。
4. 初始化 Sync Manager，如下所示：

```
var syncManager = new AWS.CognitoSyncManager();
```

Unity

1. 建立 CognitoAWSCredentials 的執行個體，請遵循 取得憑證 中的指示操作。
2. 建立 CognitoSyncManager 的執行個體。傳遞 CognitoAwsCredentials 物件和 AmazonCognitoSyncConfig，並且至少包括區域設定，如下所示：

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =
 REGION };
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

Xamarin

1. 建立 CognitoAWSCredentials 的執行個體，請遵循 取得憑證 中的指示操作。
2. 建立 CognitoSyncManager 的執行個體。傳遞 CognitoAwsCredentials 物件和 AmazonCognitoSyncConfig，並且至少包括區域設定，如下所示：

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =
 REGION };
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

了解資料集

Amazon Cognito 將使用者描述檔資料整理成資料集。每個資料集最多可包含 1 MB 鍵值組形式的資料。資料集是您可以同步的最精細實體。在叫用同步方法之前，對資料集執行的讀取和寫入操作只會影

響本機存放區。Amazon Cognito 使用唯一字串來識別資料集。您可以建立新的資料集或開啟現有的資料集，如下所示。

Android

```
Dataset dataset = client.openOrCreateDataset("datasetname");
```

若要刪除資料集，請先呼叫將其從本機儲存空間中移除的方法，然後呼叫 synchronize 方法，將資料集從 Amazon Cognito 刪除，如下所示：

```
dataset.delete();
dataset.synchronize(syncCallback);
```

iOS - Objective-C

```
AWSCognitoDataset *dataset = [syncClient openOrCreateDataset:@"myDataSet"];
```

若要刪除資料集，請先呼叫將其從本機儲存空間中移除的方法，然後呼叫 synchronize 方法，將資料集從 Amazon Cognito 刪除，如下所示：

```
[dataset clear];
[dataset synchronize];
```

iOS - Swift

```
let dataset = syncClient.openOrCreateDataset("myDataSet")!
```

若要刪除資料集，請先呼叫將其從本機儲存空間中移除的方法，然後呼叫 synchronize 方法，將資料集從 Amazon Cognito 刪除，如下所示：

```
dataset.clear()
dataset.synchronize()
```

JavaScript

```
syncManager.openOrCreateDataset('myDatasetName', function(err, dataset) {
```

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

Unity

```
string myValue = dataset.Get("myKey");
dataset.Put("myKey", "newValue");
```

若要將索引鍵從資料集中刪除，請使用 Remove，如下所示：

```
dataset.Remove("myKey");
```

Xamarin

```
Dataset dataset = syncManager.OpenOrCreateDataset("myDatasetName");
```

若要刪除資料集，請先呼叫將其從本機儲存空間中移除的方法，然後呼叫 synchronize 方法，將資料集從 Amazon Cognito 刪除，如下所示：

```
dataset.Delete();
dataset.SynchronizeAsync();
```

在資料集中讀取和寫入資料

Amazon Cognito 資料集的功能有如字典，可依索引鍵來存取值。您可以讀取、新增或修改資料集的索引鍵和值，就好像資料集是字典一樣，如以下範例所示：

請注意，在呼叫同步方法之前，您寫入資料集的值只會影響資料的本機快取副本。

Android

```
String value = dataset.get("myKey");
dataset.put("myKey", "my value");
```

iOS - Objective-C

```
[dataset setString:@"my value" forKey:@"myKey"];
```

```
NSString *value = [dataset stringForKey:@"myKey"];
```

iOS - Swift

```
dataset.setString("my value", forKey:"myKey")
let value = dataset.stringForKey("myKey")
```

JavaScript

```
dataset.get('myKey', function(err, value) {
  console.log('myRecord: ' + value);
});

dataset.put('newKey', 'newValue', function(err, record) {
  console.log(record);
});

dataset.remove('oldKey', function(err, record) {
  console.log(success);
});
```

Unity

```
string myValue = dataset.Get("myKey");
dataset.Put("myKey", "newValue");
```

Xamarin

```
//obtain a value
string myValue = dataset.Get("myKey");

// Create a record in a dataset and synchronize with the server
dataset.OnSyncSuccess += SyncSuccessCallback;
dataset.Put("myKey", "myValue");
dataset.SynchronizeAsync();

void SyncSuccessCallback(object sender, SyncSuccessEventArgs e) {
  // Your handler code here
}
```

Android

若要將索引鍵從資料集移除，請使用 remove 方法，如下所示：

```
dataset.remove("myKey");
```

iOS - Objective-C

若要將索引鍵從資料集中刪除，請使用 removeObjectForKey，如下所示：

```
[dataset removeObjectForKey:@"myKey"];
```

iOS - Swift

若要將索引鍵從資料集中刪除，請使用 removeObjectForKey，如下所示：

```
dataset.removeObjectForKey("myKey")
```

Unity

若要將索引鍵從資料集中刪除，請使用 Remove，如下所示：

```
dataset.Remove("myKey");
```

Xamarin

您可以使用 Remove 來將索引鍵從資料集中刪除：

```
dataset.Remove("myKey");
```

將本機資料與同步存放區同步化

Android

synchronize 方法會比較本機快取資料與存放在 Amazon Cognito Sync 存放區中的資料。遠端變更會從 Amazon Cognito Sync 存放區提取出來；如果發生任何衝突，將會叫用衝突解決方案；而裝置上的更新值會推播至服務。若要同步資料集，請呼叫其 synchronize方法：

```
dataset.synchronize(syncCallback);
```

synchronize方法會接收 SyncCallback 界面的實作，如下所述。

synchronizeOnConnectivity()方法會在可連線時嘗試進行同步。如果立即有連線可用，synchronizeOnConnectivity()的行為就像 synchronize()。否則就會監控連線的變化，一有連線可用，即執行同步。如果呼叫 synchronizeOnConnectivity()多次，只會保留最後一次同步請求，而且只有最後一次回呼會觸發。如果資料集或回呼已進行記憶體回收，則此方法不會執行同步，且回呼不會觸發。

若要進一步了解資料集同步和不同的回呼，請參閱處理回呼。

iOS - Objective-C

synchronize 方法會比較本機快取資料與存放在 Amazon Cognito Sync 存放區中的資料。遠端變更會從 Amazon Cognito Sync 存放區提取出來；如果發生任何衝突，將會叫用衝突解決方案；而裝置上的更新值會推播至服務。若要同步資料集，請呼叫其 synchronize方法：

synchronize方法不同步，並傳回 AWSTask 物件來處理回應：

```
[[dataset synchronize] continueWithBlock:^id(AWSTask *task) {
    if (task.isCancelled) {
        // Task cancelled.
    } else if (task.error) {
        // Error while executing task.
    } else {
        // Task succeeded. The data was saved in the sync store.
    }
    return nil;
}];
```

synchronizeOnConnectivity方法會在裝置有連線時，嘗試進行同步。首先，synchronizeOnConnectivity會檢查連線，如果裝置在線上，就會立即叫用 synchronize，並傳回與該嘗試相關聯的 AWSTask 物件。

如果裝置離線，synchronizeOnConnectivity會 1) 排定在下次裝置上線時進行同步，並且 2) 傳回 AWSTask 和 nil 結果。排定的同步只在資料集物件的生命週期內有效。如果在恢復連線之前，應用程式已結束，則不會同步資料。如果您希望在排定同步期間發生事件時收到通知，您必須新增 AWSCognito中的通知觀察器。

若要進一步了解資料集同步和不同的回呼，請參閱處理回呼。

iOS - Swift

synchronize 方法會比較本機快取資料與存放在 Amazon Cognito Sync 存放區中的資料。遠端變更會從 Amazon Cognito Sync 存放區提取出來；如果發生任何衝突，將會叫用衝突解決方案；而裝置上的更新值會推播至服務。若要同步資料集，請呼叫其 synchronize方法：

synchronize方法不同步，並傳回 AWSTask 物件來處理回應：

```
dataset.synchronize().continueWith(block: { (task) -> AnyObject? in

            if task.isCancelled {
                // Task cancelled.
            } else if task.error != nil {
                // Error while executing task
            } else {
                // Task succeeded. The data was saved in the sync store.
            }
            return task
})
```

synchronizeOnConnectivity方法會在裝置有連線時，嘗試進行同步。首先，synchronizeOnConnectivity會檢查連線，如果裝置在線上，就會立即叫用 synchronize，並傳回與該嘗試相關聯的 AWSTask 物件。

如果裝置離線，synchronizeOnConnectivity會 1) 排定在下次裝置上線時進行同步，並且 2) 傳回 AWSTask 物件和 nil 結果。排定的同步只在資料集物件的生命週期內有效。如果在恢復連線之前，應用程式已結束，則不會同步資料。如果您希望在排定同步期間發生事件時收到通知，您必須新增 AWSCognito中的通知觀察器。

若要進一步了解資料集同步和不同的回呼，請參閱處理回呼。

JavaScript

synchronize 方法會比較本機快取資料與存放在 Amazon Cognito Sync 存放區中的資料。遠端變更會從 Amazon Cognito Sync 存放區提取出來；如果發生任何衝突，將會叫用衝突解決方案；而裝置上的更新值會推播至服務。若要同步資料集，請呼叫其 synchronize方法：

```
dataset.synchronize();
```

若要進一步了解資料集同步和不同的回呼，請參閱處理回呼。

Unity

Synchronize 方法會比較本機快取資料與存放在 Amazon Cognito Sync 存放區中的資料。遠端變更會從 Amazon Cognito Sync 存放區提取出來；如果發生任何衝突，將會叫用衝突解決方案；而裝置上的更新值會推播至服務。若要同步資料集，請呼叫其 `synchronize`方法：

```
dataset.Synchronize();
```

同步會以非同步方式執行，且最後會呼叫您可以在資料集中指定的其中一個回呼。

若要進一步了解資料集同步和不同的回呼，請參閱處理回呼。

Xamarin

`synchronize` 方法會比較本機快取資料與存放在 Amazon Cognito Sync 存放區中的資料。遠端變更會從 Amazon Cognito Sync 存放區提取出來；如果發生任何衝突，將會叫用衝突解決方案；而裝置上的更新值會推播至服務。若要同步資料集，請呼叫其 `synchronize`方法：

```
dataset.SynchronizeAsync();
```

若要進一步了解資料集同步和不同的回呼，請參閱處理回呼。

處理回呼

> ⚠ 如果您第一次使用 Amazon Cognito Sync，請改用 AWS AppSync。像 Amazon Cognito Sync 一樣，AWS AppSync 是讓您在裝置間同步應用程式資料的服務。
> 可同步使用者資料，如應用程式偏好設定或遊戲狀態。也擴充這些功能，允許多個使用者在共用資料上即時同步及協作。

本節說明如何處理回呼。

Android

SyncCallback 界面

藉由實作 `SyncCallback`界面，您可以在您的應用程式上接收資料集同步的相關通知。然後，您的應用程式可以主動決定是否要刪除本機資料、合併未驗證和已驗證的設定檔，以及解決同步衝突。您應該實作界面所需的下列方法：

- onSuccess()
- onFailure()
- onConflict()
- onDatasetDeleted()
- onDatasetsMerged()

請注意，如果您不想指定所有回呼，您也可以使用 DefaultSyncCallback類別，它會為所有回呼提供預設的空實作。

onSuccess

從同步存放區成功下載資料集後，就會觸發 onSuccess()回呼。

```
@Override
public void onSuccess(Dataset dataset, List<Record> newRecords) {
}
```

onFailure

如果在同步期間發生例外狀況，將會呼叫 onFailure()。

```
@Override
public void onFailure(DataStorageException dse) {
}
```

onConflict

如果在本機存放區和同步存放區中修改相同的索引鍵，可能會發生衝突。onConflict()方法會處理衝突解決方案。如果您不實作此方法，Amazon Cognito Sync 用戶端預設會使用最新的變更。

```
@Override
public boolean onConflict(Dataset dataset, final List<SyncConflict> conflicts) {
    List<Record> resolvedRecords = new ArrayList<Record>();
    for (SyncConflict conflict : conflicts) {
        /* resolved by taking remote records */
        resolvedRecords.add(conflict.resolveWithRemoteRecord());

        /* alternately take the local records */
        // resolvedRecords.add(conflict.resolveWithLocalRecord());
```

```
        /* or customer logic, say concatenate strings */
        // String newValue = conflict.getRemoteRecord().getValue()
        //     + conflict.getLocalRecord().getValue();
        // resolvedRecords.add(conflict.resolveWithValue(newValue);
    }
    dataset.resolve(resolvedRecords);

    // return true so that synchronize() is retried after conflicts are resolved
    return true;
}
```

onDatasetDeleted

刪除資料集後，Amazon Cognito 用戶端會使用 SyncCallback 界面來確認本機快取的資料集副本是否也要刪除。實作 onDatasetDeleted()方法指示用戶端軟體開發套件要如何處理本機資料。

```
@Override
public boolean onDatasetDeleted(Dataset dataset, String datasetName) {
    // return true to delete the local copy of the dataset
    return true;
}
```

onDatasetMerged

當先前未連接的兩個身分連結在一起時，其所有資料集都會合併。此時會透過 onDatasetsMerged()方法來通知應用程式該合併狀況：

```
@Override
public boolean onDatasetsMerged(Dataset dataset, List<String> datasetNames) {
    // return false to handle Dataset merge outside the synchronization callback
    return false;
}
```

iOS - Objective-C

同步通知

Amazon Cognito 用戶端會在同步呼叫期間發出多個 NSNotification 事件。您可以註冊，以透過標準 NSNotificationCenter 來監控這些通知：

```
[NSNotificationCenter defaultCenter]
```

```
addObserver:self
selector:@selector(myNotificationHandler:)
name:NOTIFICATION_TYPE
object:nil];
```

Amazon Cognito 支援五種通知類型，列示如下。

AWSCognitoDidStartSynchronizeNotification

當同步操作開始時呼叫。`userInfo`會包含索引鍵資料集，也就是正在同步之資料集的名稱。

AWSCognitoDidEndSynchronizeNotification

當同步操作完成時呼叫 (成功或其他)。`userInfo`會包含索引鍵資料集，也就是正在同步之資料集的名稱。

AWSCognitoDidFailToSynchronizeNotification

當同步操作失敗時呼叫。`userInfo`會包含索引鍵資料集 (也就是正在同步之資料集的名稱)，以及索引鍵錯誤 (其中包含導致失敗的錯誤)。

AWSCognitoDidChangeRemoteValueNotification

當本機變更成功推播到 Amazon Cognito 時呼叫。`userInfo`會包含索引鍵資料集 (也就是正在同步之資料集的名稱)，以及主要索引鍵 (其中包含所推播之記錄索引鍵的 NSArray)。

AWSCognitoDidChangeLocalValueFromRemoteNotification

因為同步操作而導致本機值變更時呼叫。`userInfo`會包含索引鍵資料集 (也就是正在同步之資料集的名稱)，以及主要索引鍵 (其中包含已變更之記錄索引鍵的 NSArray)。

衝突解決處理常式

在同步操作期間，如果在本機存放區和同步存放區中修改相同的索引鍵，可能會發生衝突。如果您尚未設定衝突解決處理常式，Amazon Cognito 預設會選擇最新的更新。

您可以實作並指派 AWSCognitoRecordConflictHandler，以更改預設的衝突解決方案。針對本機快取資料，以及同步存放區中的衝突記錄，AWSCognitoConflict 輸入參數衝突都有包含 AWSCognitoRecord 物件。您可以使用 AWSCognitoConflict 來解決與本機記錄：[conflict resolveWithLocalRecord]、遠端記錄：[conflict resolveWithRemoteRecord] 或全新值：[conflict resolveWithValue:value] 的衝突。從這個方法傳回 nil 會阻止同步繼續進行，而且下次同步程序開始時，還會再出現衝突。

您可以在用戶端層級設定衝突解決處理常式：

```
client.conflictHandler = ^AWSCognitoResolvedConflict* (NSString *datasetName,
 AWSCognitoConflict *conflict) {
    // always choose local changes
    return [conflict resolveWithLocalRecord];
};
```

或是在資料集層級：

```
dataset.conflictHandler = ^AWSCognitoResolvedConflict* (NSString *datasetName,
 AWSCognitoConflict *conflict) {
    // override and always choose remote changes
    return [conflict resolveWithRemoteRecord];
};
```

刪除資料集處理常式

刪除資料集後，Amazon Cognito 用戶端會使用 AWSCognitoDatasetDeletedHandler 來確認本機快取的資料集副本是否也應刪除。如果未實作 AWSCognitoDatasetDeletedHandler，將會自動清除本機資料。如果您希望先保留一份本機資料副本，再進行抹除，或是要保留本機資料，請實作 AWSCognitoDatasetDeletedHandler。

您可以在用戶端層級設定資料集刪除處理常式：

```
client.datasetDeletedHandler = ^BOOL (NSString *datasetName) {
    // make a backup of the data if you choose
    ...
    // delete the local data (default behavior)
    return YES;
};
```

或是在資料集層級：

```
dataset.datasetDeletedHandler = ^BOOL (NSString *datasetName) {
    // override default and keep the local data
    return NO;
};
```

資料集合併處理常式

當先前未連接的兩個身分連結在一起時，其所有資料集都會合併。此時會透過 DatasetMergeHandler來通知應用程式該合併狀況。處理常式將會收到根資料集的名稱，以及標示為根資料集合併的資料集名稱陣列。

如果未實作 DatasetMergeHandler，將會忽略這些資料集，但是這些資料集會繼續耗用身分最大總數 20 個資料集中的空間。

您可以在用戶端層級設定資料集合併處理常式：

```
client.datasetMergedHandler = ^(NSString *datasetName, NSArray *datasets) {
    // Blindly delete the datasets
    for (NSString *name in datasets) {
        AWSCognitoDataset *merged = [[AWSCognito defaultCognito]
 openOrCreateDataset:name];
       [merged clear];
       [merged synchronize];
    }
};
```

或是在資料集層級：

```
dataset.datasetMergedHandler = ^(NSString *datasetName, NSArray *datasets) {
    // Blindly delete the datasets
    for (NSString *name in datasets) {
        AWSCognitoDataset *merged = [[AWSCognito defaultCognito]
 openOrCreateDataset:name];
        // do something with the data if it differs from existing dataset
        ...
        // now delete it
        [merged clear];
        [merged synchronize];
    }
};
```

iOS - Swift

同步通知

Amazon Cognito 用戶端會在同步呼叫期間發出多個 NSNotification 事件。您可以註冊，以透過標準 NSNotificationCenter 來監控這些通知：

```
NSNotificationCenter.defaultCenter().addObserver(observer: self,
```

```
    selector: "myNotificationHandler",
    name:NOTIFICATION_TYPE,
    object:nil)
```

Amazon Cognito 支援五種通知類型，列示如下。

AWSCognitoDidStartSynchronizeNotification

當同步操作開始時呼叫。`userInfo`會包含索引鍵資料集，也就是正在同步之資料集的名稱。

AWSCognitoDidEndSynchronizeNotification

當同步操作完成時呼叫 (成功或其他)。`userInfo`會包含索引鍵資料集，也就是正在同步之資料集的名稱。

AWSCognitoDidFailToSynchronizeNotification

當同步操作失敗時呼叫。`userInfo`會包含索引鍵資料集 (也就是正在同步之資料集的名稱)，以及索引鍵錯誤 (其中包含導致失敗的錯誤)。

AWSCognitoDidChangeRemoteValueNotification

當本機變更成功推播到 Amazon Cognito 時呼叫。`userInfo`會包含索引鍵資料集 (也就是正在同步之資料集的名稱)，以及主要索引鍵 (其中包含所推播之記錄索引鍵的 NSArray)。

AWSCognitoDidChangeLocalValueFromRemoteNotification

因為同步操作而導致本機值變更時呼叫。`userInfo`會包含索引鍵資料集 (也就是正在同步之資料集的名稱)，以及主要索引鍵 (其中包含已變更之記錄索引鍵的 NSArray)。

衝突解決處理常式

在同步操作期間，如果在本機存放區和同步存放區中修改相同的索引鍵，可能會發生衝突。如果您尚未設定衝突解決處理常式，Amazon Cognito 預設會選擇最新的更新。

您可以實作並指派 `AWSCognitoRecordConflictHandler`，以更改預設的衝突解決方案。針對本機快取資料，以及同步存放區中的衝突記錄，`AWSCognitoConflict`輸入參數衝突都有包含 `AWSCognitoRecord` 物件。您可以使用 `AWSCognitoConflict`來解決與本機記錄：[conflict resolveWithLocalRecord]、遠端記錄：[conflict resolveWithRemoteRecord] 或全新值：[conflict resolveWithValue:value] 的衝突。從這個方法傳回 nil 會阻止同步繼續進行，而且下次同步程序開始時，還會再出現衝突。

您可以在用戶端層級設定衝突解決處理常式：

```
client.conflictHandler = {
    (datasetName: String?, conflict: AWSCognitoConflict?) ->
 AWSCognitoResolvedConflict? in
    return conflict.resolveWithLocalRecord()
}
```

或是在資料集層級：

```
dataset.conflictHandler = {
    (datasetName: String?, conflict: AWSCognitoConflict?) ->
 AWSCognitoResolvedConflict? in
    return conflict.resolveWithLocalRecord()
}
```

刪除資料集處理常式

刪除資料集後，Amazon Cognito 用戶端會使用 AWSCognitoDatasetDeletedHandler 來確認本機快取的資料集副本是否也應刪除。如果未實作 AWSCognitoDatasetDeletedHandler，將會自動清除本機資料。如果您希望先保留一份本機資料副本，再進行抹除，或是要保留本機資料，請實作 AWSCognitoDatasetDeletedHandler。

您可以在用戶端層級設定資料集刪除處理常式：

```
client.datasetDeletedHandler = {
      (datasetName: String!) -> Bool in
      // make a backup of the data if you choose
      ...
      // delete the local data (default behaviour)
      return true
}
```

或是在資料集層級：

```
dataset.datasetDeletedHandler = {
      (datasetName: String!) -> Bool in
      // make a backup of the data if you choose
      ...
      // delete the local data (default behaviour)
      return true
}
```

資料集合併處理常式

當先前未連接的兩個身分連結在一起時，其所有資料集都會合併。此時會透過 DatasetMergeHandler來通知應用程式該合併狀況。處理常式將會收到根資料集的名稱，以及標示為根資料集合併的資料集名稱陣列。

如果未實作 DatasetMergeHandler，將會忽略這些資料集，但是這些資料集會繼續耗用身分最大總數 20 個資料集中的空間。

您可以在用戶端層級設定資料集合併處理常式：

```
client.datasetMergedHandler = {
    (datasetName: String!, datasets: [AnyObject]!) -> Void in
    for nameObject in datasets {
        if let name = nameObject as? String {
            let merged = AWSCognito.defaultCognito().openOrCreateDataset(name)
            merged.clear()
            merged.synchronize()
        }
    }
}
```

或是在資料集層級：

```
dataset.datasetMergedHandler = {
    (datasetName: String!, datasets: [AnyObject]!) -> Void in
    for nameObject in datasets {
        if let name = nameObject as? String {
            let merged = AWSCognito.defaultCognito().openOrCreateDataset(name)
            // do something with the data if it differs from existing dataset
              ...
              // now delete it
            merged.clear()
            merged.synchronize()
        }
    }
}
```

JavaScript

同步回呼

在資料集上執行 synchronize() 時，您可以選擇指定回呼來處理下列每個狀態：

```
dataset.synchronize({

   onSuccess: function(dataset, newRecords) {
      //...
   },

   onFailure: function(err) {
      //...
   },

   onConflict: function(dataset, conflicts, callback) {
      //...
   },

   onDatasetDeleted: function(dataset, datasetName, callback) {
      //...
   },

  onDatasetMerged: function(dataset, datasetNames, callback) {
      //...
  }

});
```

onSuccess()

從同步存放區成功更新資料集後，就會觸發 onSuccess()回呼。如果您不定義回呼，同步成功時將不提示。

```
onSuccess: function(dataset, newRecords) {
   console.log('Successfully synchronized ' + newRecords.length + ' new records.');
}
```

onFailure()

如果在同步期間發生例外狀況，將會呼叫 onFailure()。如果您不定義回呼，同步失敗時將不提示。

```
onFailure: function(err) {
   console.log('Synchronization failed.');
   console.log(err);
```

```
}
```

onConflict()

如果在本機存放區和同步存放區中修改相同的索引鍵，可能會發生衝突。onConflict()方法會處理衝突解決方案。如果您不實作此方法，當有衝突時，同步將會中止。

```
onConflict: function(dataset, conflicts, callback) {

   var resolved = [];

   for (var i=0; i<conflicts.length; i++) {

      // Take remote version.
      resolved.push(conflicts[i].resolveWithRemoteRecord());

      // Or... take local version.
      // resolved.push(conflicts[i].resolveWithLocalRecord());

      // Or... use custom logic.
      // var newValue = conflicts[i].getRemoteRecord().getValue() +
 conflicts[i].getLocalRecord().getValue();
      // resolved.push(conflicts[i].resovleWithValue(newValue);

   }

   dataset.resolve(resolved, function() {
      return callback(true);
   });

   // Or... callback false to stop the synchronization process.
   // return callback(false);

}
```

onDatasetDeleted()

刪除資料集後，Amazon Cognito 用戶端會使用 onDatasetDeleted() 回呼來決定本機快取的資料集副本是否也應刪除。依預設，不會刪除該資料集。

```
onDatasetDeleted: function(dataset, datasetName, callback) {
```

```
   // Return true to delete the local copy of the dataset.
   // Return false to handle deleted datasets outside the synchronization callback.

   return callback(true);

}
```

onDatasetMerged()

當先前未連接的兩個身分連結在一起時，其所有資料集都會合併。此時會透過 onDatasetsMerged()回呼來通知應用程式該合併狀況。

```
onDatasetMerged: function(dataset, datasetNames, callback) {

   // Return true to continue the synchronization process.
   // Return false to handle dataset merges outside the synchronization callback.

   return callback(false);

}
```

Unity

在您開啟或建立的資料集之後，您可以為其設定當您使用同步方法時所要觸發的不同回呼。這是向其註冊回呼的方式：

```
dataset.OnSyncSuccess += this.HandleSyncSuccess;
dataset.OnSyncFailure += this.HandleSyncFailure;
dataset.OnSyncConflict = this.HandleSyncConflict;
dataset.OnDatasetMerged = this.HandleDatasetMerged;
dataset.OnDatasetDeleted = this.HandleDatasetDeleted;
```

請注意，SyncSuccess和 SyncFailure 是使用 +=，而不是 =，因此您可以向其訂閱多個回呼。

OnSyncSuccess

從雲端成功更新資料集後，就會觸發 OnSyncSuccess回呼。如果您不定義回呼，同步成功時將不提示。

```
private void HandleSyncSuccess(object sender, SyncSuccessEvent e)
{
    // Continue with your game flow, display the loaded data, etc.
```

```
}
```

OnSyncFailure

如果在同步期間發生例外狀況，將會呼叫 OnSyncFailure。如果您不定義回呼，同步失敗時將不提示。

```
private void HandleSyncFailure(object sender, SyncFailureEvent e)
{
    Dataset dataset = sender as Dataset;
    if (dataset.Metadata != null) {
        Debug.Log("Sync failed for dataset : " + dataset.Metadata.DatasetName);
    } else {
        Debug.Log("Sync failed");
    }
    // Handle the error
    Debug.LogException(e.Exception);
}
```

OnSyncConflict

如果在本機存放區和同步存放區中修改相同的索引鍵，可能會發生衝突。OnSyncConflict回呼會處理衝突解決方案。如果您不實作此方法，當有衝突時，同步將會中止。

```
private bool HandleSyncConflict(Dataset dataset, List < SyncConflict > conflicts)
{
  if (dataset.Metadata != null) {
    Debug.LogWarning("Sync conflict " + dataset.Metadata.DatasetName);
  } else {
    Debug.LogWarning("Sync conflict");
  }
  List < Amazon.CognitoSync.SyncManager.Record > resolvedRecords = new List <
 Amazon.CognitoSync.SyncManager.Record > ();
  foreach(SyncConflict conflictRecord in conflicts) {
    // SyncManager provides the following default conflict resolution methods:
    //      ResolveWithRemoteRecord - overwrites the local with remote records
    //      ResolveWithLocalRecord - overwrites the remote with local records
    //      ResolveWithValue - to implement your own logic
    resolvedRecords.Add(conflictRecord.ResolveWithRemoteRecord());
  }
  // resolves the conflicts in local storage
  dataset.Resolve(resolvedRecords);
```

```
  // on return true the synchronize operation continues where it left,
  //      returning false cancels the synchronize operation
  return true;
}
```

OnDatasetDeleted

刪除資料集後，Amazon Cognito 用戶端會使用 OnDatasetDeleted 回呼來決定本機快取的資料集副本是否也應刪除。依預設，不會刪除該資料集。

```
private bool HandleDatasetDeleted(Dataset dataset)
  {
      Debug.Log(dataset.Metadata.DatasetName + " Dataset has been deleted");
      // Do clean up if necessary
      // returning true informs the corresponding dataset can be purged in the local
 storage and return false retains the local dataset
      return true;
  }
```

OnDatasetMerged

當先前未連接的兩個身分連結在一起時，其所有資料集都會合併。此時會透過 OnDatasetsMerged回呼來通知應用程式該合併狀況。

```
public bool HandleDatasetMerged(Dataset localDataset, List<string> mergedDatasetNames)
{
    foreach (string name in mergedDatasetNames)
    {
        Dataset mergedDataset = syncManager.OpenOrCreateDataset(name);
        //Lambda function to delete the dataset after fetching it
        EventHandler<SyncSuccessEvent> lambda;
        lambda = (object sender, SyncSuccessEvent e) => {
            ICollection<string> existingValues = localDataset.GetAll().Values;
            ICollection<string> newValues = mergedDataset.GetAll().Values;

            //Implement your merge logic here

            mergedDataset.Delete(); //Delete the dataset locally
            mergedDataset.OnSyncSuccess -= lambda; //We don't want this callback to be
 fired again
            mergedDataset.OnSyncSuccess += (object s2, SyncSuccessEvent e2) => {
                localDataset.Synchronize(); //Continue the sync operation that was
 interrupted by the merge
```

```
            };
            mergedDataset.Synchronize(); //Synchronize it as deleted, failing to do so
 will leave us in an inconsistent state
        };
        mergedDataset.OnSyncSuccess += lambda;
        mergedDataset.Synchronize(); //Asnchronously fetch the dataset
    }

    // returning true allows the Synchronize to continue and false stops it
    return false;
}
```

Xamarin

在您開啟或建立的資料集之後，您可以為其設定當您使用同步方法時所要觸發的不同回呼。這是向其註冊回呼的方式：

```
dataset.OnSyncSuccess += this.HandleSyncSuccess;
dataset.OnSyncFailure += this.HandleSyncFailure;
dataset.OnSyncConflict = this.HandleSyncConflict;
dataset.OnDatasetMerged = this.HandleDatasetMerged;
dataset.OnDatasetDeleted = this.HandleDatasetDeleted;
```

請注意，SyncSuccess和 SyncFailure 是使用 +=，而不是 =，因此您可以向其訂閱多個回呼。

OnSyncSuccess

從雲端成功更新資料集後，就會觸發 OnSyncSuccess回呼。如果您不定義回呼，同步成功時將不提示。

```
private void HandleSyncSuccess(object sender, SyncSuccessEventArgs e)
{
    // Continue with your game flow, display the loaded data, etc.
}
```

OnSyncFailure

如果在同步期間發生例外狀況，將會呼叫 OnSyncFailure。如果您不定義回呼，同步失敗時將不提示。

```
private void HandleSyncFailure(object sender, SyncFailureEventArgs e)
{
```

```
    Dataset dataset = sender as Dataset;
    if (dataset.Metadata != null) {
        Console.WriteLine("Sync failed for dataset : " + dataset.Metadata.DatasetName);
    } else {
        Console.WriteLine("Sync failed");
    }
}
```

OnSyncConflict

如果在本機存放區和同步存放區中修改相同的索引鍵，可能會發生衝突。OnSyncConflict回呼會處理衝突解決方案。如果您不實作此方法，當有衝突時，同步將會中止。

```
private bool HandleSyncConflict(Dataset dataset, List < SyncConflict > conflicts)
{
  if (dataset.Metadata != null) {
    Console.WriteLine("Sync conflict " + dataset.Metadata.DatasetName);
  } else {
    Console.WriteLine("Sync conflict");
  }
  List < Amazon.CognitoSync.SyncManager.Record > resolvedRecords = new List <
 Amazon.CognitoSync.SyncManager.Record > ();
  foreach(SyncConflict conflictRecord in conflicts) {
    // SyncManager provides the following default conflict resolution methods:
    //      ResolveWithRemoteRecord - overwrites the local with remote records
    //      ResolveWithLocalRecord - overwrites the remote with local records
    //      ResolveWithValue - to implement your own logic
    resolvedRecords.Add(conflictRecord.ResolveWithRemoteRecord());
  }
  // resolves the conflicts in local storage
  dataset.Resolve(resolvedRecords);
  // on return true the synchronize operation continues where it left,
  //      returning false cancels the synchronize operation
  return true;
}
```

OnDatasetDeleted

刪除資料集後，Amazon Cognito 用戶端會使用 OnDatasetDeleted 回呼來決定本機快取的資料集副本是否也應刪除。依預設，不會刪除該資料集。

```
private bool HandleDatasetDeleted(Dataset dataset)
{
```

```
    Console.WriteLine(dataset.Metadata.DatasetName + " Dataset has been deleted");
    // Do clean up if necessary
    // returning true informs the corresponding dataset can be purged in the local
 storage and return false retains the local dataset
    return true;
}
```

OnDatasetMerged

當先前未連接的兩個身分連結在一起時，其所有資料集都會合併。此時會透過 OnDatasetsMerged回呼來通知應用程式該合併狀況。

```
public bool HandleDatasetMerged(Dataset localDataset, List<string> mergedDatasetNames)
{
    foreach (string name in mergedDatasetNames)
    {
        Dataset mergedDataset = syncManager.OpenOrCreateDataset(name);

            //Implement your merge logic here

        mergedDataset.OnSyncSuccess += lambda;
        mergedDataset.SynchronizeAsync(); //Asnchronously fetch the dataset
    }

    // returning true allows the Synchronize to continue and false stops it
    return false;
}
```

推播同步

> ⚠ 如果您第一次使用 Amazon Cognito Sync，請改用 AWS AppSync。像 Amazon Cognito Sync 一樣，AWS AppSync 是讓您在裝置間同步應用程式資料的服務。
> 可同步使用者資料，如應用程式偏好設定或遊戲狀態。也擴充這些功能，允許多個使用者在共用資料上即時同步及協作。

Amazon Cognito 會自動追蹤身分與裝置之間的關聯。使用推播同步功能可讓您確保在身分資料變更時，指定身分的每個執行個體都會收到通知。每當特定身分的同步存放區資料變更時，推播同步可確保與該身分相關聯的所有裝置都會收到靜音推送通知，通知他們有所變更。

> **Note**
>
> JavaScript、Unity 或 Xamarin 不支援推播同步。

您必須先設定帳戶來進行推播同步，並且在 Amazon Cognito 主控台中啟用推播同步，才能使用推播同步。

建立 Amazon Simple Notification Service (Amazon SNS) 應用程式

依照 SNS 開發人員指南中的說明，為您的支援平台建立並設定 Amazon SNS 應用程式。

在 Amazon Cognito 主控台中啟用推播同步

您可以透過 Amazon Cognito 主控台來啟用推播同步。從主控台首頁：

1. 針對要啟用推播同步的身分集區，按一下其名稱。該身分集區的 Dashboard (儀表板) 頁面隨即出現。
2. 在 Dashboard (儀表板) 頁面右上角，按一下 Manage Identity Pools (管理身分集區)。此時將出現 Federated Identities (聯合身分) 頁面。
3. 向下捲動，然後按一下 Push synchronization (推播同步) 將其展開。
4. 在 Service role (服務角色) 下拉式選單中，選取授予 Cognito 許可的 IAM 角色來傳送 SNS 通知。按一下 Create role (建立角色)，在 AWS IAM 主控台中建立或修改與身分集區相關聯的角色。
5. 選取平台應用程式，然後按一下 Save Changes (儲存變更)。
6. 授予對應用程式的 SNS 存取權

在 AWS Identity and Access Management 主控台中，將您的 IAM 角色設定為擁有完整的 Amazon SNS 存取權，或建立擁有完整 Amazon SNS 存取權的新角色。下列範例角色信任政策授予 Amazon Cognito Sync 擔任 IAM 角色的有限功能。Amazon Cognito Sync 只能代表 `aws:SourceArn` 條件中的身分集區和 `aws:SourceAccount` 條件中的帳戶擔任執行此操作時的角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-sync.amazonaws.com"
```

```
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "AWS:SourceArn": "arn:aws:cognito-identity:us-
east-1:123456789012:identitypool/us-east-1:177a950c-2c08-43f0-9983-28727EXAMPLE"
                }
            }
        }
    ]
}
```

若要進一步了解 IAM 角色，請參閱角色 (委派和聯合)。

在您的應用程式中使用播送同步：Android

您的應用程式將需要匯入 Google Play 服務。您可以從 Android SDK manager 下載最新版的 Google Play SDK。請遵循 Android 實作的 Android 說明文件來註冊您的應用程式，並從 GCM 接收註冊 ID。取得註冊 ID 之後，您必須向 Amazon Cognito 註冊裝置，如下列程式碼片段所示：

```
String registrationId = "MY_GCM_REGISTRATION_ID";
try {
    client.registerDevice("GCM", registrationId);
} catch (RegistrationFailedException rfe) {
    Log.e(TAG, "Failed to register device for silent sync", rfe);
} catch (AmazonClientException ace) {
    Log.e(TAG, "An unknown error caused registration for silent sync to fail", ace);
}
```

您現在可以訂閱裝置，以從特定資料集接收更新：

```
Dataset trackedDataset = client.openOrCreateDataset("myDataset");
if (client.isDeviceRegistered()) {
    try {
        trackedDataset.subscribe();
    } catch (SubscribeFailedException sfe) {
        Log.e(TAG, "Failed to subscribe to datasets", sfe);
    } catch (AmazonClientException ace) {
        Log.e(TAG, "An unknown error caused the subscription to fail", ace);
```

```
    }
}
```

若要停止從資料集接收推播通知，只要呼叫 unsubscribe 方法即可。若要訂閱 CognitoSyncManager物件中的所有資料集 (或特定子集)，請使用 subscribeAll()：

```
if (client.isDeviceRegistered()) {
    try {
        client.subscribeAll();
    } catch (SubscribeFailedException sfe) {
        Log.e(TAG, "Failed to subscribe to datasets", sfe);
    } catch (AmazonClientException ace) {
        Log.e(TAG, "An unknown error caused the subscription to fail", ace);
    }
}
```

實作 Android BroadcastReceiver 物件時，您可以檢查修改資料集的最新版本，並決定您的應用程式是否需要再次同步：

```
@Override
public void onReceive(Context context, Intent intent) {

    PushSyncUpdate update = client.getPushSyncUpdate(intent);

    // The update has the source (cognito-sync here), identityId of the
    // user, identityPoolId in question, the non-local sync count of the
    // data set and the name of the dataset. All are accessible through
    // relevant getters.

    String source = update.getSource();
    String identityPoolId = update.getIdentityPoolId();
    String identityId = update.getIdentityId();
    String datasetName = update.getDatasetName;
    long syncCount = update.getSyncCount;

    Dataset dataset = client.openOrCreateDataset(datasetName);

    // need to access last sync count. If sync count is less or equal to
    // last sync count of the dataset, no sync is required.

    long lastSyncCount = dataset.getLastSyncCount();
    if (lastSyncCount < syncCount) {
```

```
        dataset.synchronize(new SyncCallback() {
            // ...
        });
    }

}
```

推送通知酬載中有下列索引鍵可供使用：

- `source`：cognito-sync。這可以用來做為通知之間的區分要素。
- `identityPoolId`：身分集區 ID。這可以用於驗證或其他資訊，雖然從接收者的觀點而言，這並不完整。
- `identityId`：集區中的身分 ID。
- `datasetName`：已更新之資料集的名稱。這可以用來進行 openOrCreateDataset 呼叫。
- `syncCount`：遠端資料集的同步計數。您可以利用它來確定本機資料集已過期，而傳入的同步是新的。

在您的應用程式中使用推播同步：iOS - Objective-C

若要取得應用程式的裝置權杖，請依照有關「註冊遠端通知」的 Apple 說明文件來操作。從 APN 收到 NSData 物件形式的裝置權杖之後，您需要使用同步用戶端的 `registerDevice:` 方法，向 Amazon Cognito 註冊裝置，如下所示：

```
AWSCognito *syncClient = [AWSCognito defaultCognito];
    [[syncClient registerDevice: devToken] continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to registerDevice: %@", task.error);
        } else {
            NSLog(@"Successfully registered device with id: %@", task.result);
        }
        return nil;
      }
    ];
```

在偵錯模式下，您的裝置會向 APN 沙盒註冊；在版本模式下，則是向 APN 註冊。若要從特定資料集接收更新，請使用 `subscribe`方法：

```
[[[syncClient openOrCreateDataset:@"MyDataset"] subscribe]
 continueWithBlock:^id(AWSTask *task) {
```

```
        if(task.error){
            NSLog(@"Unable to subscribe to dataset: %@", task.error);
        } else {
            NSLog(@"Successfully subscribed to dataset: %@", task.result);
        }
        return nil;
      }
    ];
```

若要停止從資料集接收推播通知，只要呼叫 unsubscribe方法即可：

```
[[[syncClient openOrCreateDataset:@”MyDataset”] unsubscribe]
 continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to unsubscribe from dataset: %@", task.error);
        } else {
            NSLog(@"Successfully unsubscribed from dataset: %@", task.result);
        }
        return nil;
      }
    ];
```

若要訂閱 AWSCognito物件中的所有資料集，請呼叫 subscribeAll：

```
[[syncClient subscribeAll] continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to subscribe to all datasets: %@", task.error);
        } else {
            NSLog(@"Successfully subscribed to all datasets: %@", task.result);
        }
        return nil;
      }
    ];
```

在呼叫 subscribeAll之前，請務必在每個資料集上至少同步一次，這樣資料集才會存在於伺服器上。

若要對推播通知做出反應，您需要在應用程式委派中實作 didReceiveRemoteNotification方法：

```
- (void)application:(UIApplication *)application didReceiveRemoteNotification:
(NSDictionary *)userInfo
```

```
    {
        [[NSNotificationCenter defaultCenter]
 postNotificationName:@"CognitoPushNotification" object:userInfo];
    }
```

如果您使用通知處理常式來發佈通知，就可以在應用程式中有您的資料集頭銜的其他地方回應通知。如果您訂閱通知，像這樣 ...

```
[[NSNotificationCenter defaultCenter] addObserver:self
 selector:@selector(didReceivePushSync:)
    name: :@"CognitoPushNotification" object:nil];
```

...您可以對通知採取像這樣的動作：

```
- (void)didReceivePushSync:(NSNotification*)notification
    {
        NSDictionary * data = [(NSDictionary *)[notification object]
 objectForKey:@"data"];
        NSString * identityId = [data objectForKey:@"identityId"];
        NSString * datasetName = [data objectForKey:@"datasetName"];
        if([self.dataset.name isEqualToString:datasetName] && [self.identityId
 isEqualToString:identityId]){
            [[self.dataset synchronize] continueWithBlock:^id(AWSTask *task) {
                if(!task.error){
                    NSLog(@"Successfully synced dataset");
                }
                return nil;
            }];
        }
    }
```

推送通知酬載中有下列索引鍵可供使用：

- source：cognito-sync。這可以用來做為通知之間的區分要素。
- identityPoolId：身分集區 ID。這可以用於驗證或其他資訊，雖然從接收者的觀點而言，這並不完整。
- identityId：集區中的身分 ID。
- datasetName：已更新之資料集的名稱。這可以用來進行 openOrCreateDataset呼叫。
- syncCount：遠端資料集的同步計數。您可以利用它來確定本機資料集已過期，而傳入的同步是新的。

在您的應用程式中使用推播同步：iOS - Swift

若要取得應用程式的裝置權杖，請依照有關「註冊遠端通知」的 Apple 說明文件來操作。從 APN 收到 NSData 物件形式的裝置權杖之後，您需要使用同步用戶端的 registerDevice: 方法，向 Amazon Cognito 註冊裝置，如下所示：

```
let syncClient = AWSCognito.default()
syncClient.registerDevice(devToken).continueWith(block: { (task: AWSTask!) ->
 AnyObject! in
    if (task.error != nil) {
        print("Unable to register device: " + task.error.localizedDescription)

    } else {
        print("Successfully registered device with id: \(task.result)")
    }
    return task
})
```

在偵錯模式下，您的裝置會向 APN 沙盒註冊；在版本模式下，則是向 APN 註冊。若要從特定資料集接收更新，請使用 subscribe方法：

```
syncClient.openOrCreateDataset("MyDataset").subscribe().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
  if (task.error != nil) {
      print("Unable to subscribe to dataset: " + task.error.localizedDescription)

  } else {
      print("Successfully subscribed to dataset: \(task.result)")
  }
  return task
})
```

若要停止從資料集接收推播通知，請呼叫 unsubscribe方法：

```
syncClient.openOrCreateDataset("MyDataset").unsubscribe().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
  if (task.error != nil) {
      print("Unable to unsubscribe to dataset: " + task.error.localizedDescription)

  } else {
      print("Successfully unsubscribed to dataset: \(task.result)")
  }
```

```
  return task
})
```

若要訂閱 AWSCognito物件中的所有資料集，請呼叫 subscribeAll：

```
syncClient.openOrCreateDataset("MyDataset").subscribeAll().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
  if (task.error != nil) {
      print("Unable to subscribe to all datasets: " + task.error.localizedDescription)

  } else {
      print("Successfully subscribed to all datasets: \(task.result)")
  }
  return task
})
```

在呼叫 subscribeAll之前，請務必在每個資料集上至少同步一次，這樣資料集才會存在於伺服器上。

若要對推播通知做出反應，您需要在應用程式委派中實作 didReceiveRemoteNotification方法：

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo:
 [NSObject : AnyObject],
  fetchCompletionHandler completionHandler: (UIBackgroundFetchResult) -> Void) {

 NSNotificationCenter.defaultCenter().postNotificationName("CognitoPushNotification",
 object: userInfo)
})
```

如果您使用通知處理常式來發佈通知，就可以在應用程式中有您的資料集頭銜的其他地方回應通知。如果您訂閱通知，像這樣 ...

```
NSNotificationCenter.defaultCenter().addObserver(observer:self,
   selector:"didReceivePushSync:",
   name:"CognitoPushNotification",
   object:nil)
```

...您可以對通知採取像這樣的動作：

```
func didReceivePushSync(notification: NSNotification) {
```

```
    if let data = (notification.object as! [String: AnyObject])["data"] as? [String:
 AnyObject] {
        let identityId = data["identityId"] as! String
        let datasetName = data["datasetName"] as! String

        if self.dataset.name == datasetName && self.identityId == identityId {
          dataset.synchronize().continueWithBlock {(task) -> AnyObject! in
              if task.error == nil {
                print("Successfully synced dataset")
              }
              return nil
          }
        }
    }
}
```

推送通知酬載中有下列索引鍵可供使用：

- `source`：cognito-sync。這可以用來做為通知之間的區分要素。
- `identityPoolId`：身分集區 ID。這可以用於驗證或其他資訊，雖然從接收者的觀點而言，這並不完整。
- `identityId`：集區中的身分 ID。
- `datasetName`：已更新之資料集的名稱。這可以用來進行 `openOrCreateDataset`呼叫。
- `syncCount`：遠端資料集的同步計數。您可以利用它來確定本機資料集已過期，而傳入的同步是新的。

Amazon Cognito 串流

> ⚠ 如果您第一次使用 Amazon Cognito Sync，請改用 AWS AppSync。像 Amazon Cognito Sync 一樣，AWS AppSync 是讓您在裝置間同步應用程式資料的服務。
> 可同步使用者資料，如應用程式偏好設定或遊戲狀態。也擴充這些功能，允許多個使用者在共用資料上即時同步及協作。

Amazon Cognito 串流可讓開發人員控制並深入了解其存放在 Amazon Cognito 中的資料。開發人員現在可以設定 Kinesis 串流，以在資料更新及同步時接收事件。Amazon Cognito 可以即時將每項資料集變更推送至您擁有的 Kinesis 串流。

您可以使用 Amazon Cognito 串流將所有同步資料移到 Kinesis，然後 Kinesis 會串流至諸如 Amazon Redshift 的資料倉儲工具，供進一步分析。若要進一步了解 Kinesis，請參閱 Amazon Kinesis 入門。

設定串流

您可以在 Amazon Cognito 主控台中設定 Amazon Cognito 串流。若要在 Amazon Cognito 主控台中啟用 Amazon Cognito 串流，您需要選取 Kinesis 串流 (發佈的目標)，以及 IAM 角色 (授予將事件放在所選串流中的 Amazon Cognito 許可)。

從主控台首頁：

1. 針對要設定 Amazon Cognito 串流的身分集區，按一下其名稱。該身分集區的 Dashboard (儀表板) 頁面隨即出現。
2. 在 Dashboard (儀表板) 頁面右上角，按一下 Manage Identity Pools (管理身分集區)。Manage Federated Identities (管理聯合身分) 頁面隨即出現。
3. 向下捲動，然後按一下 Cognito Streams (Cognito 串流) 將其展開。
4. 在 Stream name (串流名稱) 下拉式選單中，選取現有 Kinesis 串流的名稱。或者，按一下 Create stream (建立串流)，輸入串流名稱和碎片數量，以建立串流。若要了解碎片並取得預估串流所需碎片數量的協助，請參閱 Kinesis 開發人員指南。
5. 在 Publish role (發佈角色) 下拉式選單中，選取授予 Amazon Cognito 許可的 IAM 角色來發佈串流。按一下 Create role (建立角色)，在 AWS IAM 主控台中建立或修改與身分集區相關聯的角色。
6. 在 Stream status (串流狀態) 下拉式選單中，選取 Enabled (已啟用)，以啟用串流更新。按一下 Save Changes (儲存變更)。

成功設定 Amazon Cognito 串流之後，對此身分集區中的資料集進行的所有後續更新，都會傳送至該串流。

串流內容

傳送至串流的每個記錄都代表單一同步。以下是傳送至串流的記錄範例：

```
{
    "identityPoolId": "Pool Id",
    "identityId": "Identity Id",
    "dataSetName": "Dataset Name",
    "operation": "(replace|remove)",
    "kinesisSyncRecords": [
        {
```

```
            "key": "Key",
            "value": "Value",
            "syncCount": 1,
            "lastModifiedDate": 1424801824343,
            "deviceLastModifiedDate": 1424801824343,
            "op": "(replace|remove)"
        },
        ...
    ],
    "lastModifiedDate": 1424801824343,
    "kinesisSyncRecordsURL": "S3Url",
    "payloadType": "(S3Url|Inline)",
    "syncCount": 1
}
```

如果更新大於 Kinesis 的 1 MB 承載大小上限，Amazon Cognito 將會併入預先簽署的 Amazon Simple Storage Service (Amazon S3) URL，其中包含更新的完整內容。

設定 Amazon Cognito 串流之後，如果您刪除 Kinesis 串流或變更角色信任許可，導致 Amazon Cognito Sync 無法繼續擔任該角色，您就關閉了 Amazon Cognito 串流。您必須重新建立 Kinesis 串流或修復角色，然後必須重新開啟串流。

大量發佈

設定 Amazon Cognito 串流之後，您就可以為身分集區中的現有資料執行大量發佈操作。啟動大量發佈操作 (透過主控台或直接透過 API 皆可) 之後，Amazon Cognito 會開始將此資料發佈至接收您的更新的相同串流。

使用大量發佈操作時，Amazon Cognito 不保證傳送至串流的資料唯一性。您可能會同時從更新和大量發佈的一部分收到相同的更新。在處理來自串流的記錄時，請記得這一點。

若要大量發佈您的所有串流，請按照「設定串流」下的步驟 1-6 操作，然後按一下 Start bulk publish (開始大量發佈)。您一次只能進行一項大量發佈操作，而且每 24 小時內只能有一次成功的大量發佈請求。

Amazon Cognito 事件

⚠ 如果您第一次使用 Amazon Cognito Sync，請改用 AWS AppSync。像 Amazon Cognito Sync 一樣，AWS AppSync 是讓您在裝置間同步應用程式資料的服務。

可同步使用者資料，如應用程式偏好設定或遊戲狀態。也擴充這些功能，允許多個使用者在共用資料上即時同步及協作。

Amazon Cognito 事件可讓您執行 AWS Lambda 函數來回應 Amazon Cognito 中的重要事件。當有資料集同步時，Amazon Cognito 會引發同步觸發器事件。當使用者更新資料時，您可以使用同步觸發器事件來採取動作。此函數可以先評估資料，並選擇性地處理，然後再將資料存放在雲端中，並同步到使用者的其他裝置。這很適合用來驗證來自裝置的資料，然後再同步到使用者的其他裝置，或是根據傳入的資料來更新資料集中的其他值，例如當玩家達到新等級時發出獎項。

下列步驟將引導您設定每次同步 Amazon Cognito 資料集時執行的 Lambda 函數。

> **Note**
>
> 使用 Amazon Cognito 事件時，您只能使用從 Amazon Cognito Identity 取得的憑證。如果您有相關聯的 Lambda 函數，但您使用 AWS 帳戶憑證 (開發人員憑證) 來呼叫 `UpdateRecords`，則不會叫用您的 Lambda 函數。

在 AWS Lambda 中建立函數

若要將 Lambda 與 Amazon Cognito 整合，您需要先在 Lambda 中建立函數。若要這麼做：

在 Amazon Cognito 中選取 Lambda 函數

1. 開啟 Lambda 主控台。
2. 按一下 Create a Lambda function (建立 Lambda 函數)。
3. 在 Select blueprint (選取藍圖) 畫面中，搜尋並選取 "cognito-sync-trigger"。
4. 在 Configure event sources (設定事件來源) 畫面中，將 Event source type (事件來源類型) 保持設為 "Cognito Sync Trigger" (Cognito 同步觸發器)，然後選取您的身分集區。按一下 Next (下一步)。

> **Note**
>
> 在主控台外設定 Amazon Cognito Sync 觸發程序時，您必須新增以 Lambda 資源為基礎的許可，以允許 Amazon Cognito 叫用該函數。您可以從 Lambda 主控台 (請參閱對 AWS Lambda 使用以資源為基礎的政策) 或透過使用 Lambda AddPermission 操作新增此許可。以 Lambda 資源為基礎的政策範例

下列 AWS Lambda 資源型政策允許 Amazon Cognito 叫用 Lambda 函數的有限功能。Amazon Cognito 只能代表 `aws:SourceArn` 條件中的身分集區和 `aws:SourceAccount` 條件中的帳戶叫用函數。

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "lambda-allow-cognito-my-function",
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-sync.amazonaws.com"
            },
            "Action": "lambda:InvokeFunction",
            "Resource": "<your Lambda function ARN>",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "AWS:SourceArn": "<your identity pool ARN>"
                }
            }
        }
    ]
}
```

5. 在 Configure function (設定函數) 畫面中，輸入函數的名稱和描述。將 Runtime (執行時間) 保持設為 "Node.js"。針對我們的範例，程式碼保持不變。預設範例不變更所要同步的資料。它只記錄 Amazon Cognito Sync 觸發器事件發生的事實。將 Handler name (處理常式名稱) 保持設為 "index.handler"。針對角色，選取授予存取 AWS Lambda 程式碼許可的 IAM 角色。若要修改角色，請參閱 IAM 主控台。將 Advanced (進階) 設定保持不變。按一下 Next (下一步)。
6. 在 Review (檢閱) 畫面上，檢閱詳細資訊，然後按一下 Create function (建立函數)。下一頁會顯示您的新 Lambda 函數。

現在您有了以 Lambda 編寫的適當函數，您需要選擇該函數做為 Amazon Cognito Sync 觸發器事件的處理常式。下列步驟將逐步引導您完成此程序。

從主控台首頁：

1. 針對要設定 Amazon Cognito 事件的身分集區，按一下其名稱。該身分集區的 Dashboard (儀表板) 頁面隨即出現。
2. 在 Dashboard (儀表板) 頁面右上角，按一下 Manage Federated Identities (管理聯合身分)。Manage Federated Identities (管理聯合身分) 頁面隨即出現。
3. 向下捲動，然後按一下 Cognito Events (Cognito 事件) 將其展開。
4. 在 Sync Trigger (Sync 觸發器) 下拉式選單中，選取您想要在同步事件發生時觸發的 Lambda 函數。
5. 按一下 Save Changes (儲存變更)。

現在，每次同步資料集時，就會執行您的 Lambda 函數。下節將說明在進行同步時，如何讀取及修改函數中的資料。

編寫 Sync 觸發程序的 Lambda 函數

Sync 觸發程序遵循服務供應商界面使用的程式設計模型。Amazon Cognito 會以下列 JSON 格式提供輸入給您的 Lambda 函數。

```
{
  "version": 2,
  "eventType": "SyncTrigger",
  "region": "us-east-1",
  "identityPoolId": "identityPoolId",
  "identityId": "identityId",
  "datasetName": "datasetName",
  "datasetRecords": {
    "SampleKey1": {
      "oldValue": "oldValue1",
      "newValue": "newValue1",
      "op": "replace"
    },
    "SampleKey2": {
      "oldValue": "oldValue2",
      "newValue": "newValue2",
      "op": "replace"
    },..
  }
}
```

Amazon Cognito 預期函數的傳回值格式與輸入相同。

為同步觸發程序事件編寫函數時，請觀察以下內容：

- Amazon Cognito 在 UpdateRecords 期間叫用您的 Lambda 函數時，該函數必須在 5 秒內回應。如果沒做到，Amazon Cognito Sync 服務會產生 `LambdaSocketTimeoutException` 例外狀況。您無法提高此逾時值。
- 如果你收到 `LambdaThrottledException` 例外狀況，請再次嘗試同步操作以更新記錄。
- Amazon Cognito 提供資料集中的所有記錄做為函數的輸入。
- 應用程式使用者更新的記錄具有設定為 `replace` 的 op 欄位。已刪除的記錄具有設定為 `remove` 的 op 欄位。
- 即使應用程式使用者沒有更新記錄，您也可以修改任何記錄。
- 除了 datasetRecords 之外的所有欄位都是唯讀欄位。請勿變更這些欄位。如果變更這些欄位，將無法更新記錄。
- 若要修改記錄的值，請更新該值，並將 op 設定為 `replace`。
- 若要移除記錄，請將 op 設定為 `remove`，或將該值設定為 null。
- 若要新增記錄，請將新的記錄新增到 datasetRecords 陣列。
- 當 Amazon Cognito 更新記錄時，會忽略回應中任何省略的記錄。

Lambda 函數範例

以下 Lambda 函數範例顯示如何存取、修改及移除資料。

```
console.log('Loading function');

exports.handler = function(event, context) {
    console.log(JSON.stringify(event, null, 2));

    //Check for the event type
    if (event.eventType === 'SyncTrigger') {

        //Modify value for a key
        if('SampleKey1' in event.datasetRecords){
            event.datasetRecords.SampleKey1.newValue = 'ModifyValue1';
            event.datasetRecords.SampleKey1.op = 'replace';
        }

        //Remove a key
        if('SampleKey2' in event.datasetRecords){
            event.datasetRecords.SampleKey2.op = 'remove';
```

```
        }

        //Add a key
        if(!('SampleKey3' in event.datasetRecords)){
            event.datasetRecords.SampleKey3={'newValue':'ModifyValue3', 'op' :
 'replace'};
        }

    }
    context.done(null, event);
};
```

使用 Amazon Cognito 主控台

您可以使用 Amazon Cognito 主控台建立和管理使用者集區與身分集區。

本指南提供 Amazon Cognito 主控台中常見 Amazon Cognito 使用者集區任務的 step-by-step 逐步解說。

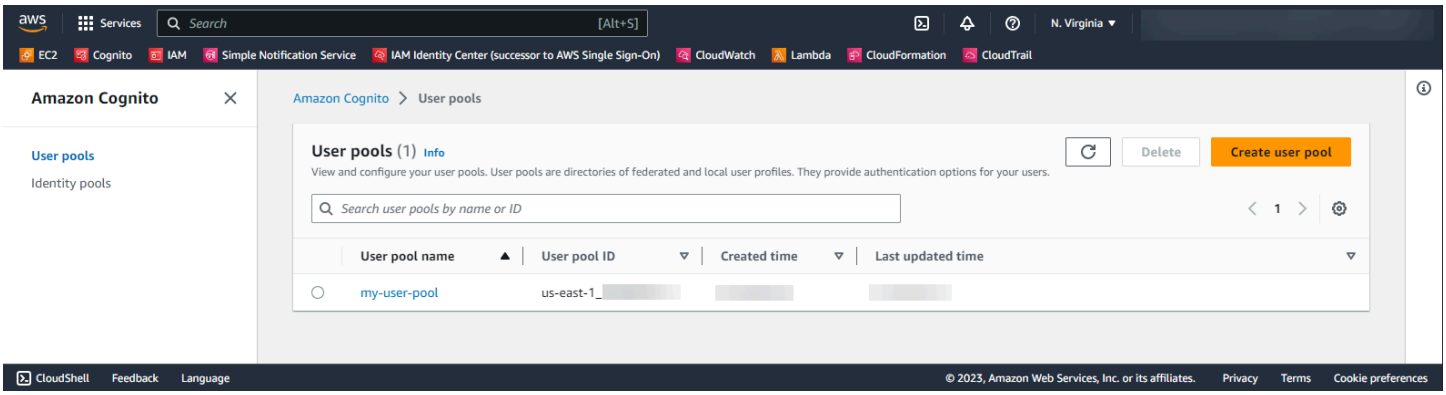
若要使用 Amazon Cognito 主控台

1. 要使用 Amazon Cognito，您需要註冊一個 AWS 帳戶。
2. 前往 Amazon Cognito 主控台。系統可能會提示您輸 AWS 入認證。
3. 若要建立或編輯使用者集區，請從左側導覽窗格選擇 User Pools (使用者集區)。

   如需詳細資訊，請參閱 使用者集區入門。
4. 若要建立或編輯身分池，請選擇 身分池。系統會將您導向至 Amazon Cognito 身分集區的原始主控台。

   如需詳細資訊，請參閱 開始使用 Amazon Cognito 身分集區。

Amazon Cognito 主控台是的一部分 AWS Management Console，其中提供有關您帳戶和帳單的資訊。如需詳細資訊，請參閱使用 AWS Management Console。



主題

- 使用者集區主控台
- 身分集區主控台

使用者集區主控台

在 Amazon Cognito 主控台的使用者集區檢視中，從清單選擇使用者集區以檢視詳細資訊。在詳細檢視中，主控台頂端的使用者集區概觀包含有關您的使用者集區的基本資訊。下列索引標籤會將您的使用者集區組態組織成相關功能。

使用者

使用者索引標籤包含使用者以及從 CSV 檔案匯入的使用者相關資訊。您可以在此索引標籤中新增、移除和編輯使用者。

參考

- 管理使用者集區中的使用者
- 從 CSV 檔案將使用者匯入使用者集區

群組

群組索引標籤包含使用者群組的相關資訊。您可以新增、修改和變更群組中的成員資格，以及變更與群組相關聯的 IAM 角色以進行身分集區整合。

參考

- 新增群組至使用者集區

登入體驗

登入體驗索引標籤包含使用者如何登入您的使用者集區的相關資訊。此索引標籤包括第三方身分識別提供者、使用者名稱選項、密碼政策、多重要素驗證 (MFA) 組態、忘記密碼行為，以及裝置記憶功能。您可以新增和修改身分識別提供者，以及變更使用者集區的整體登入行為。

參考

- 透過第三方新增使用者集區登入
- 自訂登入屬性
- 新增使用者集區密碼要求
- 將 MFA 新增到使用者集區
- 復原使用者帳戶
- 在使用者集區中使用使用者裝置

註冊體驗

註冊體驗索引標籤包含有關自助式註冊、必要屬性、驗證電話號碼和電子郵件地址以及自訂屬性的資訊。

參考

- 註冊及確認使用者帳戶
- 使用者集區屬性
- 註冊時驗證聯絡資訊

簡訊

訊息索引標簽包含有關您想要用來傳送電子郵件和簡訊給使用者的 AWS 服務 ，以及要傳送的訊息格式的資訊。

參考

- Amazon Cognito 使用者集區的電子郵件設定
- Amazon Cognito 使用者集區的簡訊設定
- 設定 SMS、電子郵件驗證訊息和使用者邀請訊息

應用程式整合

應用程式整合索引標籤包含使用者集區應用程式用戶端、指派給使用者集區服務端點的網域、API 資源伺服器、託管 UI 以及進階安全性的相關資訊。您可以深入研究每個應用程式用戶端以設定下列項目。

1. 權杖設定
2. 回呼 URL
3. 身分驗證流程
4. 屬性許可
5. 應用程式專屬的進階安全性和託管 UI 設定
6. Amazon Pinpoint 分析

參考

- 使用者集區應用程式用戶端
- 設定和使用 Amazon Cognito 託管 UI 和聯合端點
- 設定使用者集區網域
- 使用資源伺服器進行範圍、M2M 和 API 授權

- 將進階安全性新增到使用者集區
- 使用 Amazon Pinpoint 分析搭配 Amazon Cognito 使用者集區

使用者集區屬性

[使用者集區內容] 索引標籤包含與使用者無直接相關的使用者集區設定的相關資訊：Lambda 觸發程序、 AWS WAF Web ACL 保護、刪除保護和資源標記。

參考

- 使用 Lambda 觸發程序來自訂使用者集區工作流程
- 建立 AWS WAF Web ACL 與使用者集區的關聯
- 使用者集區刪除保護
- 標記您的 AWS 資源

身分集區主控台

在 Amazon Cognito 主控台的身分集區檢視中，從清單選擇身分集區以檢視詳細資訊。在詳細檢視中，主控台頂端的身分集區概觀包含有關您的使用者集區的基本資訊。下列索引標籤會將您的使用者集區組態組織成相關功能。

使用者統計資料

使用者統計資料索引標籤會顯示有關在您的身分集區中產生身分之使用者的統計資訊。您無法在此索引標籤中設定任何身分集區設定。

身分瀏覽器

身分瀏覽器索引標籤包含有關使用者在您的身分集區中產生之個人身分的資訊。您可以檢視和刪除身分。

參考

- 開始使用 Amazon Cognito 身分集區

使用者存取

使用者存取索引標籤包含有關您連結至身分集區的身分提供者、開發人員提供者、指派給身分的預設 IAM 角色，以及未經驗證訪客存取組態的資訊。您可以深入研究每個身分提供者以設定下列項目。

1. 使用 IAM 角色選取的角色型存取控制

2. 使用存取控制的屬性的屬性型存取控制

參考

- 外部身分提供者身分集區
- IAM 角色
- 已進行身分驗證和未進行身分驗證的身分
- 開發人員驗證的身分
- 使用以角色為基礎的存取控制
- 使用屬性進行存取控制

身分集區屬性

身分集區屬性索引標籤包含有關其他身分集區組態的資訊：基本 (傳統) 驗證和資源標籤。

- 身分集區 (聯合身分) 驗證流程
- 標記您的 AWS 資源

Amazon Cognito 的安全

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，該架構專為滿足對安全性最敏感的組織的需求而打造。

安全是 AWS 與您之間共同承擔的責任。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 Amazon Cognito 的合規計劃，請參閱合規計劃的合規計劃AWS服務範圍服務。
- 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

本文件有助於您了解如何在使用 Amazon Cognito 時套用共同責任模型。說明如何設定 Amazon Cognito 以符合您的安全和合規目標。您也會學到如何使用其他可 AWS 協助您監控和保護 Amazon Cognito 資源的服務。

目錄

Amazon Cognito 的資料保護

AWS 共同責任模型適用於 Amazon Cognito (Amazon Cognito) 中的資料保護。如此模型所述， AWS 負責保護執行所有 AWS 雲端的全域基礎結構。您負責維護在此基礎設施上託管內容的控制權。此內容包括您使用之 AWS 服務的安全性設定和管理工作。如需有關資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。

基於資料保護目的，我們建議您使用 AWS Identity and Access Management (IAM) 保護 AWS 帳戶登入資料，並設定個別使用者帳戶。如此一來，每個使用者都只會獲得授予完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源進行通訊。
- 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 AWS 服務中的所有預設安全性控制。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Simple Storage Service (Amazon Simple Storage Service (Amazon S3)) 的個人資料。

我們強烈建議您絕對不要將客戶帳戶號碼等敏感的識別資訊，放在自由格式的欄位中，例如Name (名稱) 欄位。這包括當您使用主控台、API 或開 AWS 發套件使用 Amazon Cognito 或其他 AWS 服務時。 AWS CLI您在 Amazon Cognito 或其他服務中輸入的任何資料都可能選入診斷日誌中。當您提供外部伺服器的 URL 時，請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

資料加密

資料加密通常分為兩類：靜態加密和傳輸中加密。

靜態加密

Amazon Cognito 中的資料會依照業界標準進行靜態加密。

傳輸中加密

作為一項受管服務，Amazon Cognito 受到 AWS 全球網路安全的保護。有關 AWS 安全服務以及如何 AWS 保護基礎架構的詳細資訊，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 Amazon Cognito。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。

Amazon Cognito 使用者集區和身分集區具有經過 IAM 驗證、未驗證和權杖授權的 API 操作。未經驗證和權杖授權的 API 操作旨在供您的客戶 (應用程式的最終用戶) 使用。未經驗證及權杖授權的 API 操作於靜態及傳輸中加密。如需詳細資訊，請參閱 Amazon Cognito 使用者集區經身分驗證和未進行身分驗證的 API 操作。

> **Note**
>
> Amazon Cognito 會在內部加密您的內容，且不支援客戶提供的金鑰。

Amazon Cognito 的 Identity and Access Management

AWS Identity and Access Management (IAM) 可協助系統管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員負責控制誰可通過驗證 (登入) 和獲得授權 (具有許可) 來使用 Amazon Cognito 資源。您可以使用 IAM AWS 服務 ，無需額外付費。

主題

- 物件
- 使用身分驗證
- 使用政策管理存取權
- Amazon Cognito 如何與 IAM 搭配運作
- Amazon Cognito 的身分型政策範例
- Amazon Cognito 身分和存取疑難排解
- 使用 Amazon Cognito 的服務連結角色

物件

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，具體取決於您在 Amazon Cognito 中所做的工作。

服務使用者 – 如果您使用 Amazon Cognito 執行任務，您的管理員會為您提供所需的憑證和許可。隨著您使用更多 Amazon Cognito 功能來執行工作，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。若您無法存取 Amazon Cognito 中的某項功能，請參閱 Amazon Cognito 身分和存取疑難排解。

服務管理員 – 如果您負責公司內的 Amazon Cognito 資源，您可能具有 Amazon Cognito 的完整存取權。您的任務是判斷您的服務使用者應存取哪些 Amazon Cognito 功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司可搭配 Amazon Cognito 使用 IAM 的方式，請參閱 Amazon Cognito 如何與 IAM 搭配運作。

IAM 管理員 – 如果您是 IAM 管理員，建議您掌握如何撰寫政策以管理 Amazon Cognito 存取權的詳細資訊。若要檢視您可以在 IAM 中使用的範例 Amazon Cognito 身分型政策，請參閱 Amazon Cognito 的身分型政策範例。

使用身分驗證

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)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取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。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時認證 AWS 服務 來存取。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需 IAM Identity Center 的相關資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是您內部的身分，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。

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。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。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 當主參與者 (使用者、root 使用者或角色

工作階段) 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。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的更多相關資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。若要了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些委託人 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon 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。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可範圍的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 實體許可範圍。
- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。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 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。如需組織和 SCP 的更多相關資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需更多資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

Amazon Cognito 如何與 IAM 搭配運作

在您使用 IAM 管理 Amazon Cognito 的存取權之前，請先了解有哪些 IAM 功能可搭配 Amazon Cognito 使用。

您可以搭配 Amazon Cognito 使用的 IAM 功能

IAM 功能	Amazon Cognito 支援
身分型政策	是

IAM 功能	Amazon Cognito 支援
資源型政策	否
政策動作	是
政策資源	是
政策條件索引鍵	是
ACL	否
ABAC(政策中的標籤)	部分
臨時憑證	是
主體許可	否
服務角色	是
服務連結角色	是

若要深入瞭解 Amazon Cognito 和其他 AWS 服務如何與大多數 IAM 功能搭配使用，請參閱 IAM 使用者指南中的可與 IAM 搭配使用的AWS 服務。

Amazon Cognito 身分型政策

支援身分型政策	是

身分型政策是可以連接到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

Amazon Cognito 的身分型政策範例

若要檢視 Amazon Cognito 身分型政策的範例，請參閱 Amazon Cognito 的身分型政策範例。

Amazon Cognito 內的資源型政策

支援以資源基礎的政策	否

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

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。當主體和資源位於不同時 AWS 帳戶，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 角色與資源型政策有何差異。

Amazon Cognito 的政策動作

支援政策動作	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看 Amazon Cognito 動作的清單，請參閱《服務授權參考》中的 Amazon Cognito 定義的動作。

Amazon Cognito 中的政策動作會在動作前使用以下字首：

```
cognito-identity
```

若要在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "cognito-identity:action1",
      "cognito-identity:action2"
         ]
```

簽章與未簽章的 API

當您使用登入 AWS 資料簽署 Amazon Cognito API 請求時，您可以在 AWS Identity and Access Management (IAM) 政策中對其進行限制。您必須使用 AWS 憑證簽署的 API 請求，包括使用 `AdminInitiateAuth` 進行的伺服器端登入，以及建立、檢視或修改您的 Amazon Cognito 資源的動作，例如 `UpdateUserPool`。如需有關已簽署 API 要求的詳細資訊，請參閱簽署 AWS API 要求。

由於 Amazon Cognito 是您要向大眾提供之應用程式的消費者身分產品，因此您可以存取下列未簽署的 API。您的應用程式會為您的使用者和潛在使用者發出這些 API 請求。某些 API 不需要事先授權，例如啟動新的驗證工作階段的 `InitiateAuth`。有些 API 會使用存取字符或工作階段金鑰進行授權，例如，`VerifySoftwareToken` 為具有現有已驗證工作階段的使用者完成 MFA 設定。未簽署、授權的 Amazon Cognito 使用者集區 API 支援請求語法中的 `Session` 或 `AccessToken` 參數，如 Amazon Cognito API 參考中所示。未簽署的 Amazon Cognito 身分 API 支援顯示於 Amazon Cognito 聯合身分 API 參考的 `IdentityId` 參數。

如需 Amazon Cognito 使用者集區 API 操作的授權模型和角色的詳細資訊，請參閱 Amazon Cognito 使用者集區經身分驗證和未進行身分驗證的 API 操作。

Amazon Cognito 身分池 API 操作

- `GetId`
- `GetOpenIdToken`
- `GetCredentialsForIdentity`
- `UnlinkIdentity`

Amazon Cognito 使用者集區 API 操作

- `AssociateSoftwareToken`
- `ChangePassword`
- `ConfirmDevice`
- `ConfirmForgotPassword`

- ConfirmSignUp
- DeleteUser
- DeleteUserAttributes
- ForgetDevice
- ForgotPassword
- GetDevice
- GetUser
- GetUserAttributeVerificationCode
- GlobalSignOut
- InitiateAuth
- ListDevices
- ResendConfirmationCode
- RespondToAuthChallenge
- RevokeToken
- SetUserMFAPreference
- SetUserSettings
- SignUp
- UpdateAuthEventFeedback
- UpdateDeviceStatus
- UpdateUserAttributes
- VerifySoftwareToken
- VerifyUserAttribute

若要檢視 Amazon Cognito 身分型政策的範例，請參閱 Amazon Cognito 的身分型政策範例。

Amazon Cognito 的政策資源

支援政策資源	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

Amazon 資源名稱 (ARN)

適用於 Amazon Cognito 聯合身分的 ARN

在 Amazon Cognito 使用者集區 (聯合身分) 中，您可以利用 Amazon Resource Name (ARN) 格式，限制 IAM 使用者對特定身分集區的存取，如下列範例所示。如需 ARN 的詳細資訊，請參閱 IAM 識別符。

```
arn:aws:cognito-identity:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

Amazon Cognito Sync 的 ARN

在 Amazon Cognito Sync 中，客戶也可以依身分集區 ID、身分 ID 和資料集名稱來限制存取。

若為在身分集區上操作的 API，身分集區 ARN 格式與 Amazon Cognito 聯合身分相同，但服務名稱為 cognito-sync，而不是 cognito-identity：

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

若為在單一身分上操作的 API，例如 RegisterDevice，您可以利用下列 ARN 格式來參照個別身分：

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/
identity/IDENTITY_ID
```

若為在資料集上操作的 API，例如 UpdateRecords和 ListRecords，您可以利用下列 ARN 格式來參照個別資料集：

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/
identity/IDENTITY_ID/dataset/DATASET_NAME
```

適用於 Amazon Cognito 使用者集區的 ARN

針對 Amazon Cognito 您的使用者集區，您可以利用下列 ARN 格式來限制使用者對特定身分集區的存取：

```
arn:aws:cognito-idp:REGION:ACCOUNT_ID:userpool/USER_POOL_ID
```

若要查看 Amazon Cognito 資源類型及其 ARN 的清單，請參閱《服務授權參考》中的 Amazon Cognito 定義的資源。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon Cognito 定義的動作。

若要檢視 Amazon Cognito 身分型政策的範例，請參閱 Amazon Cognito 的身分型政策範例。

Amazon Cognito 政策條件金鑰

支援服務特定政策條件金鑰	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

若要查看 Amazon Cognito 條件金鑰的清單，請參閱《服務授權參考》中的 Amazon Cognito 的條件金鑰。若要了解您可以搭配哪些動作和資源使用條件金鑰，請參閱 Amazon Cognito 定義的動作。

若要檢視 Amazon Cognito 身分型政策的範例，請參閱 Amazon Cognito 的身分型政策範例。

Amazon Cognito 中的存取控制清單 (ACL)

支援 ACL	否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

使用 Amazon Cognito 的屬性型存取控制 (ABAC)

支援 ABAC (政策中的標籤)	部分

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

若要根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件金鑰，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱《IAM 使用者指南》中的使用屬性型存取控制 (ABAC)。

將暫時憑證與 Amazon Cognito 搭配使用

支援臨時憑證	是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配AWS 服務 使用，請參閱 IAM 使用者指南中的 IAM。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立暫時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的切換至角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而非使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

Amazon Cognito 的跨服務主體許可

支援轉寄存取工作階段 (FAS)	否

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》。

Amazon Cognito 的服務角色

支援服務角色	是

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

如需 Amazon Cognito 服務角色的詳細資訊，請參閱 啟用推播同步 和 推播同步。

> **Warning**
>
> 變更服務角色的許可有可能會中斷 Amazon Cognito 功能。只有在 Amazon Cognito 提供指引時，才能編輯服務角色。

Amazon Cognito 的服務連結角色

支援服務連結角色	是

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。

如需建立或管理 Amazon Cognito 服務連結角色的詳細資訊，請參閱 使用 Amazon Cognito 的服務連結角色。

Amazon Cognito 的身分型政策範例

根據預設，使用者和角色不具備建立或修改 Amazon Cognito 資源的許可。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行工作。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

如需 Amazon Cognito 所定義之動作和資源類型的詳細資訊，包括每種資源類型的 ARN 格式，請參閱《服務授權參考》中的適用 Amazon Cognito 的動作、資源和條件金鑰。

主題

- 政策最佳實務
- 使用 Amazon Cognito 主控台
- 允許使用者檢視他們自己的許可
- 限制主控台存取特定身分集區
- 允許集區中的所有身分存取特定資料集

政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 Amazon Cognito 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們可用在您的 AWS 帳戶. 建議您透過定義特定

於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。

- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授與對服務動作的存取權 (如透過特定 AWS 服務) 使用 AWS CloudFormation。如需更多資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。若要在呼叫 API 作業時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

> **Note**
>
> 當您檢視和修改 Amazon Cognito 資源時，原始版本和新版本的 Amazon Cognito 主控台具有不同的基礎行為。如果您僅於條件 `aws:ViaAWSService` 為 true 時授予 `cognito-idp` 服務字首下的動作權限，則可能受影響的 IAM 主體可使用原始主控台中的 Amazon Cognito 資源，但並非於新主控台中。若要於 Amazon Cognito 主控台中使用，請勿於您 IAM 政策中設定 Amazon Cognito 許可上的 `aws:ViaAWSService` 條件。

使用 Amazon Cognito 主控台

若要存取 Amazon Cognito 主控台，您必須擁有最基本的一組許可。這些許可必須允許您列出並檢視您 AWS 帳戶的. 如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

您不需要為僅對 AWS CLI 或 AWS API 進行呼叫的使用者允許最低主控台權限。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

為確保使用者和角色仍可使用 Amazon Cognito 主控台，請同時將 Amazon Cognito `ConsoleAccess` 或`ReadOnly` AWS 受管政策附加到實體。如需詳細資訊，請參閱《IAM 使用者指南》中的新增許可到使用者。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此原則包含在主控台上或以程式設計方式使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

限制主控台存取特定身分集區

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cognito-identity:ListIdentityPools"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cognito-identity:*"
      ],
      "Resource": "arn:aws:cognito-identity:us-east-1:0123456789:identitypool/us-
east-1:1a1a1a1a-ffff-1111-9999-12345678"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cognito-sync:*"
      ],
      "Resource": "arn:aws:cognito-sync:us-east-1:0123456789:identitypool/us-
east-1:1a1a1a1a-ffff-1111-9999-12345678"
    }
  ]
}
```

允許集區中的所有身分存取特定資料集

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cognito-sync:ListRecords",
        "cognito-sync:UpdateRecords"
      ],
```

```
      "Resource": "arn:aws:cognito-sync:us-east-1:0123456789:identitypool/us-
east-1:1a1a1a1a-ffff-1111-9999-12345678/identity/*/dataset/UserProfile"
    }
  ]
}
```

Amazon Cognito 身分和存取疑難排解

請使用以下資訊來協助您診斷和修正使用 Amazon Cognito 和 IAM 時可能發生的常見問題。

主題

- 我未獲授權，不得在 Amazon Cognito 中執行動作
- 我沒有授權執行 iam：PassRole
- 我是管理員，並且想要允許其他人存取 Amazon Cognito
- 我想要允許我 AWS 帳戶以外的人員存取我的 Amazon Cognito 資源

我未獲授權，不得在 Amazon Cognito 中執行動作

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在mateojackson IAM 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 cognito-identity:*GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 cognito-identity:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 mateojackson 使用者的政策，允許使用 cognito-identity:*GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

我沒有授權執行 iam：PassRole

如果您收到錯誤，告知您無權執行 iam:PassRole 動作，則必須更新您的政策，您才能將角色傳遞至 Amazon Cognito。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為 `marymajor` 的 IAM 使用者嘗試使用主控台在 Amazon Cognito 中執行動作時，就會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 `iam:PassRole` 動作。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

我是管理員，並且想要允許其他人存取 Amazon Cognito

若要允許其他人存取 Amazon Cognito，您必須針對需要存取的人員或應用程式建立 IAM 實體 (使用者或角色)。他們將使用該實體的憑證來存取 AWS。然後您必須將政策連接到實體，在 Amazon Cognito 中授予他們正確的許可。

若要立即開始使用，請參閱《IAM 使用者指南》中的建立您的第一個 IAM 委派使用者及群組。

我想要允許我 AWS 帳戶以外的人員存取我的 Amazon Cognito 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 Amazon Cognito 是否支援這些功能，請參閱 Amazon Cognito 如何與 IAM 搭配運作。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱 IAM 使用者指南中您擁有的另一個 AWS 帳戶 的 IAM 使用者提供存取權限。
- 若要了解如何將資源存取權提供給第三方 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。
- 若要了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱 IAM 使用者指南中的 IAM 角色與資源型政策的差異。

使用 Amazon Cognito 的服務連結角色

Amazon Cognito 用 AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是具有信任政策的唯一 IAM 角色類型，允許擔任該角色。 AWS 服務 服務連結角色由 Amazon Cognito 預先定義，並包含服務代表您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓設定 Amazon Cognito 更為簡單，因為您不必手動新增必要的許可。Amazon Cognito 定義其服務連結角色的許可，除非另有定義，否則僅有 Amazon Cognito 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。

您必須先刪除服務連結角色的相關資源，才能將其刪除。如此可保護您的 Amazon Cognito 資源，避免您不小心移除資源的存取許可。

如需有關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) 資料欄顯示為 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。

Amazon Cognito 的服務連結角色許可

Amazon Cognito 使用下列服務連結角色：

- AWSServiceRoleForAmazonCognitoIdpEmailService— 允許 Amazon Cognito 使用者集區服務使用您的 Amazon SES 身分來傳送電子郵件。

- AWSServiceRoleForAmazonCognitoIdp— 允許 Amazon Cognito 使用者集區為您的 Amazon 精準點專案發佈事件和設定端點。

AWSServiceRoleForAmazonCognitoIdpEmailService

`AWSServiceRoleForAmazonCognitoIdpEmailService` 服務連結角色信任下列服務以擔任角色：

- `email.cognito-idp.amazonaws.com`

此角色許可政策允許 Amazon Cognito 對指定資源完成下列動作：

允許的動作 AWSServiceRoleForAmazonCognitoIdpEmailService：

- 動作：`ses:SendEmail` 和 `ses:SendRawEmail`

- 資源：*

此政策拒絕 Amazon Cognito 可在指定資源上完成下列動作：

拒絕的動作

- 動作：`ses:List*`
- 資源：*

使用這些權限，Amazon Cognito 只能使用 Amazon SES 中經驗證的電子郵件地址傳送電子郵件給使用者。您的使用者在使用者集區的用戶端應用程式中執行特定動作，例如註冊或重設密碼時，Amazon Cognito 會傳送電子郵件給您的使用者。

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。

AWSServiceRoleForAmazonCognitoIdp

服 AWSServiceRoleForAmazonCognitoIdp 務連結角色會信任下列服務擔任該角色：

- `email.cognito-idp.amazonaws.com`

此角色許可政策允許 Amazon Cognito 對指定資源完成下列動作：

允許的動作 AWSServiceRoleForAmazonCognitoIdp

- 動作：`cognito-idp:Describe`
- 資源：*

使用此許可，Amazon Cognito 可以為您呼叫 `Describe` Amazon Cognito API 操作。

> **Note**
>
> 使用 `createUserPoolClient` 和 `updateUserPoolClient` 整合 Amazon Cognito 與 Amazon Pinpoint 時，資源許可會新增至 SLR 當作內嵌政策。內嵌政策會提供 `mobiletargeting:UpdateEndpoint` 和 `mobiletargeting:PutEvents` 許可。這些許可讓 Amazon Cognito 能為您與 Cognito 整合的 Pinpoint 專案發佈事件並設定端點。

建立 Amazon Cognito 的服務連結角色

您不需要手動建立一個服務連結角色。當您將使用者集區設定為使用 Amazon SES 組態來處理 AWS Management Console、或 Amazon Cognito API 中的 AWS CLI電子郵件交付時，Amazon Cognito 會為您建立服務連結角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。設定使用者集區以使用 Amazon SES 組態處理電子郵件交付時，Amazon Cognito 會再次為您建立服務連結角色。

您用來設定您的使用者集區的 IAM 許可必須先包含 `iam:CreateServiceLinkedRole` 動作，Amazon Cognito 才能建立此角色。如需更新 IAM 中許可的詳細資訊，請參閱《IAM 使用者指南》中的變更 IAM 使用者的許可。

編輯 Amazon Cognito 的服務連結角色

您無法在 AWS Identity and Access Management中編輯 AmazonCognitoIdp 或 AmazonCognitoIdpEmailService 服務連結角色。因為可能有各種實體會參考服務連結角色，所以您無法在建立角色之後變更其名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除 Amazon Cognito 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。如果您刪除角色，則僅保留 Amazon Cognito 主動監視或維護的實體。在刪除 AmazonCognitoIdp 或 AmazonCognitoIdpEmailService 服務連結角色之前，您必須對使用該角色的每個使用者集區執行下列其中一項動作：

- 刪除使用者集區。
- 在使用者集區中，將電子郵件設定更新成使用預設的電子郵件功能。預設設定不會使用服務連結角色。

請記住，在每個使 AWS 區域 用該角色的用戶池中執行操作。

> **Note**
>
> 若 Amazon Cognito 服務在您試圖刪除資源時正在使用該角色，刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

刪除 Amazon Cognito 使用者集區

1. 登入 AWS Management Console 並開啟 Amazon Cognito 主控台，位於https://console.aws.amazon.com/cognito。
2. 選擇 Manage User Pools (管理使用者集區)。
3. 在 Your User Pools (您的使用者集區) 頁面上，選擇您要刪除的使用者集區。
4. 選擇刪除集區。
5. 在 Delete user pool (刪除使用者集區) 視窗中，鍵入 **delete**，然後選擇 Delete pool (刪除集區)。

更新 Amazon Cognito 使用者集區以使用預設的電子郵件功能

1. 登入 AWS Management Console 並開啟 Amazon Cognito 主控台，位於https://console.aws.amazon.com/cognito。
2. 選擇 Manage User Pools (管理使用者集區)。
3. 在 Your User Pools (您的使用者集區) 頁面上，選擇您要更新的使用者集區。
4. 在左側的導覽選單中，選擇 Message customizations (訊息自訂)。
5. 在 Do you want to send emails through your Amazon SES Configuration? (您是否要透過 Amazon SES 組態傳送電子郵件嗎？)，選擇 No - Use Cognito (Default) (否 - 使用 Cognito (預設))。
6. 當您完成設定電子郵件帳戶選項後，請選擇 Save changes (儲存變更)。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台 AWS CLI、或 AWS API 刪除 AmazonCognitoIdp 或 AmazonCognitoIdpEmailService 服務連結的角色。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。

Amazon Cognito 服務連結角色的支援區域

Amazon Cognito 在所有可用服務的 AWS 區域 地方都支援服務連結角色。如需詳細資訊，請參閱 AWS 區域 和端點。

在 Amazon Cognito 中記錄和監控

監控是維護 Amazon Cognito 和其他 AWS 解決方案的可靠性、可用性和效能的重要組成部分。Amazon Cognito 目前支援下列 AWS 服務 ，可讓您監控組織和其中發生的活動。

- AWS CloudTrail — CloudTrail 您可以透過 Amazon Cognito 主控台擷取 API 呼叫，以及從程式碼呼叫到 Amazon Cognito API 作業擷取 API 呼叫。例如，當使用者進行驗證時， CloudTrail 可以記錄詳細資料，例如要求中的 IP 位址、提出要求的人員以及提出要求的時間。
- Amazon CloudWatch 日誌 — 使用 CloudWatch 日誌，您可以將使用者活動的精細日誌傳送到日誌群組。例如，您可以檢閱詳細的使用者活動日誌，以疑難排解傳送電子郵件和 SMS 訊息給使用者的問題。
- Amazon CloudWatch 指標 — 使用 CloudWatch 指標，您可以在發生事件時以近乎即時的速度監控、報告和採取自動動作。例如，您可以在提供的指標上建立 CloudWatch 儀表板以監控 Amazon Cognito 使用者集區，或者在提供的指標上建立 CloudWatch 警示，以便在違反設定閾值時通知您。
- Amazon CloudWatch 日誌洞見 — 透過 CloudWatch 日誌洞見，您可以設定 CloudTrail 將事件傳送到 CloudWatch 以監控 Amazon Cognito 日 CloudTrail 誌檔。

主題

監控成本

Amazon Cognito 會根據您的使用量的下列維度收取費用。

- 使用者集區每月作用中使用者 (MAU)
- 使用 OIDC 或 SAML 聯盟登入的使用者集區 MAU
- 具有進階安全性功能的使用者集區中的 MAU
- 使用中的使用者集區應用程式用戶端和機器對機器 (M2M) 授權的要求磁碟區與用戶端憑證授權
- 針對某些類別的使用者集區 API，已購買的使用量超過預設

此外，使用者集區的功能 (例如電子郵件訊息、SMS 訊息和 Lambda 觸發器) 可能會在相依服務中產生成本。如需完整概觀，請參閱 Amazon Cognito 定價。

檢視和預測成本

您可以在AWS Billing and Cost Management 主控台中檢視並報告您的 AWS 費用。您可以在「帳單和付款」區段中找到 Amazon Cognito 的最新費用。在「帳單，按服務收費」下，過濾Cognito以查看您的使用情況。如需詳細資訊，請參閱 AWS Billing 使用者指南中的檢視您的帳單。

若要監控 API 要求率，請檢閱「Service Quotas」主控台中的「使用率」指標。例如，用戶端認證要求會顯示為 ClientAuthentication 要求率。在您的帳單中，這些要求會與產生這些要求的應用程式用戶端相關聯。有了這項資訊，您就可以公平地將成本配置給多租用戶架構中的租用戶。

若要取得一段時間內的 M2M 要求計數，您也可以將AWS CloudTrail 事件傳送至 CloudWatch 記錄檔進行分析。使用用戶端認證授與查詢Token_POST事件的 CloudTrail 事件。下列「 CloudWatch深入解析」查詢會傳回此計數。

```
filter eventName = "Token_POST" and @message like '"grant_type":["client_credentials"]'
 | stats count(*)
```

管理成本

Amazon Cognito 根據使用者計數、功能使用情況和請求量計費。以下是管理 Amazon Cognito 成本的一些提示，

不要啟用非作用中使用者

使用戶處於活動狀態的典型操作包括登錄，註冊和密碼重置。如需更完整的清單，請參閱每月作用中使用者。Amazon Cognito 不會將非活動用戶計入您的帳單中。避免任何將使用者設定為作用中的作業。而不是 AdminGetUserAPI 操作，使用操ListUsers作查詢用戶。請勿對非使用中使用者執行使用者集區作業的大量系統管理測試。

連結同盟使用者

使用 SAML 2.0 或 OpenID Connect (OIDC) 身分識別提供者登入的使用者的成本高於本機使用者。您可以將這些使用者連結至本機使用者設定檔。連結的使用者可以使用其同盟使用者隨附的屬性和存取權以本機使用者身分登入。來自 SAML 或 OIDC IdPs 的使用者如果在一個月內只使用連結的本機帳戶登入，則會以本機使用者的身分計費。

管理請求費率

如果您的使用者集區接近配額上限，您可以考慮購買額外容量來處理磁碟區。您也許可以減少應用程式中的要求量。如需詳細資訊，請參閱 針對配額限制最佳化請求率。

請求一個新的令牌，只有當你需要一個

具有用戶端認證授權的機器對機器 (M2M) 授權可達到大量的權杖要求。每個新的 Token 請求都會影響您的請求率配額和帳單大小。為了優化成本，請在應用程序的設計中包含令牌到期設置和令牌處理。

- 緩存訪問令牌，以便當您的應用程序請求新令牌時，它會接收先前發布的令牌的緩存版本。當您實作此方法時，您的快取 Proxy 會充當防護要求存取權杖的應用程式，而不會意識到先前取得的權杖過期。快取權杖非常適合使用 Lambda 函數和 Docker 容器等短期微型服務。
- 在您的應用程式中實作權杖處理機制，以說明權杖到期。在先前的令牌過期之前，不要請求新令牌。評估每個應用程式的機密性和可用性需求，並設定使用者集區應用程式用戶端以發行具有適當有效期的存取權杖。自訂權杖持續時間最適合使用壽命較長的 API 和可持續管理認證要求頻率的伺服器。

刪除未使用的客戶端憑據應用

M2M 授權帳單基於兩個因素：令牌請求的速率和執行客戶端憑據授予的應用程序客戶端數量。當 M2M 授權的應用程序客戶端不在使用中時，請刪除它們或刪除其發出客戶端憑據的授權。如需管理應用程式用戶端設定的詳細資訊，請參閱使用者集區應用程式用戶端。

管理進階安全性

當您在使用者集區中設定進階安全性功能時，進階安全性計費費率會套用至使用者集區中的所有 MAU。如果您的使用者不需要進階安全性功能，請將他們分隔到另一個使用者集區中。

追蹤以及 Service Quotas 中的配額 CloudWatch 和使用情況

您可以使用 Amazon 或使用 Service Quotas 監控 Amazon Cognito 使用 CloudWatch 者集區。您也可以在 Service Quotas 中監視身分識別集區使用情況。 CloudWatch 收集原始數據並將其處理為可讀的近乎實時的指標。在中 CloudWatch，您可以設定警示來監控特定閾值，並在達到這些閾值時傳送通知或採取動作。若要建立服務配額的 CloudWatch 警示，請參閱建立 CloudWatch警示。Amazon Cognito 指標每隔五分鐘提供一次。如需有關中保留期的詳細資訊 CloudWatch，請瀏覽 Amazon CloudWatch 常見問答集頁面。

您可以使用 Service Quotas 來檢視和管理 Amazon Cognito 使用者集區與身份集區配額使用量。Service Quotas 主控台有三個功能，檢視服務配額、要求增加服務配額，以及檢視目前的使用率。您可以使用第一個功能來檢視配額，並查看配額是否可調整。您可以使用第二個功能來要求增加 Service Quotas。您可以使用最後一個功能來檢視配額使用率。只有在您的帳戶啟動一段時間後，才能使用此功能。如需在 Service Quotas 主控台中檢視配額的詳細資訊，請參閱檢視 Service Quotas。

> **Note**
>
> Amazon Cognito 指標每 5 分鐘提供一次。如需有關中保留期的詳細資訊 CloudWatch，請瀏覽 Amazon CloudWatch 常見問答集頁面。

如果您登入的設定為 CloudWatch跨帳戶 AWS 帳戶 可觀察性的監視帳戶，您可以使用該監視帳戶將服務配額視覺化，並針對連結至該監視帳戶的來源帳戶中的指標設定警示。如需詳細資訊，請參閱CloudWatch 跨帳戶可觀察性。

主題

- 記錄來自 Amazon Cognito 使用者集區的其他活動
- Amazon Cognito 使用者集區的指標
- Amazon Cognito 使用者集區的維度
- 使用 Service Quotas 主控台追蹤指標
- 使用主 CloudWatch 控台追蹤指標
- 建立配額的 CloudWatch 警示

記錄來自 Amazon Cognito 使用者集區的其他活動

您可以設定使用者集區，將某些其他活動的詳細記錄傳送至記 CloudWatch 錄群組。這些記錄檔的精細程度比中的記錄更精細 AWS CloudTrail，對於疑難排解使用者集區可能很有用。當您啟用此功能時，可以選擇 Amazon Cognito 要將日誌傳送到其中的日誌群組。當您想要了解使用者集區透過 Amazon SNS 和 Amazon SES 傳遞的電子郵件和簡訊狀態時，使用者活動記錄會非常實用。

目前，您只能從使用者集區傳遞錯誤層級的使用者通知日誌。

詳細記錄不會取代或變更使用者集區的下列記錄功能。

1. CloudTrail 例行用戶活動（例如註冊和登錄）的日誌。
2. 使用 CloudWatch 指標大規模分析使用者活動。

另外，您也可以在日誌中找到來自使用者匯入任務和 Lambda 觸發程序的 CloudWatch 日誌。Amazon Cognito 和 Lambda 會將這些日誌與詳細活動日誌分別儲存在不同的日誌群組中。

您可以在 API 請求中使用 Amazon Cognito 使用者集區 API 設定詳細的SetLogDeliveryConfiguration活動日誌。您可以在 GetLogDeliveryConfigurationAPI 要求中檢視使用者集區的記錄設定。

您必須使用具有下列權限的 AWS 認證來授權這些要求。

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

```
            {
            "Sid": "ManageUserPoolLogs",
            "Action": [
                "cognito-idp:SetLogDeliveryConfiguration",
                "cognito-idp:GetLogDeliveryConfiguration",
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "CognitoLog",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "CognitoLoggingCWL",
            "Action": [
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

下列為來自使用者集區的範例事件。此日誌結構描述可能會變更。部分欄位可能會記錄 null 值。

```
{
```

```
    "eventTimestamp": "1687297330677",
    "eventSource": "USER_NOTIFICATION",
    "logLevel": "ERROR",
    "message": {
        "details": "String"
    },
    "logSourceId": {
        "userPoolId": "String"
    }
}
```

最好從 Amazon Cognito 傳遞日誌。您的使用者集區提供的記錄檔數量，以及 CloudWatch 記錄檔的服務配額，可能會影響記錄檔的傳遞。

CloudWatch 啟用記錄傳遞時，會收取記錄檔費用。如需詳細資訊，請參閱 Amazon CloudWatch 定價中的付費日誌。

若要將日誌傳送至日誌群組，且資源政策規定的大小超過 5120 個字元，請使用開頭為 `/aws/vendedlogs` 的路徑設定日誌群組。如需詳細資訊，請參閱啟用特定 AWS 服務的記錄功能。

Amazon Cognito 使用者集區的指標

下表列出 Amazon Cognito 使用者集區可用的指標。Amazon Cognito 的 Amazon CloudWatch 指標命名空間是 `AWS/Cognito`。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的命名空間。

> **Note**
>
> 過去兩週內沒有任何新資料點的指標不會顯示在主控台中。也不會在您於主控台 All metrics (所有指標) 標籤的搜尋方塊中輸入指標名稱或維度名稱時顯示。此外，它們不會在 list-metrics 命令的結果中傳回。擷取這些指標的最佳方式是使用 AWS CLI 中的`get-metric-data`或`get-metric-statistics`命令。

指標	描述
`SignUpSuccesses`	提供對 Amazon Cognito 使用者集區提出的成功使用者註冊請求總數。成功的使用者註冊請求會產生值 1，而不成功的請求會產生值 0。調節的請求也遭認為是不成功的請求，因此調節的請求也將產生計數 0。

指標	描述
	若要尋找成功的使用者註冊請求百分比，請使用此指標的 Average 統計數字。若要計算使用者註冊請求的總數，請使用此指標的 Sample Count 統計數字。若要計算成功的使用者註冊請求總數，請使用此指標的 Sum 統計數字。若要計算失敗的使用者註冊要求總數，請使用運算 CloudWatch Math式並從Sum統計資料中減去統Sample Count計資料。 此指標會針對每個使用者集區用戶端的每個使用者集區發佈。如果是由管理員執行使用者註冊，則指標會與使用者集區用戶端一起發佈為 Admin。 請注意，不會針對使用者匯入和使用者遷移案例發出此指標。 指標維度：UserPool、UserPoolClient 單位：計數
SignUpThrottles	提供對 Amazon Cognito 使用者集區提出的調節使用者註冊請求總數。每當調節使用者註冊請求時，就會發佈計數 1。 若要計算調節的使用者註冊請求總數，請使用此指標的 Sum 統計數字。 此指標針對每個用戶端的每個使用者集區發佈。如果是由管理員調節請求，則指標會與使用者集區用戶端一起發佈為 Admin。 指標維度：UserPool、UserPoolClient 單位：計數

指標	描述
SignInSuccesses	提供對 Amazon Cognito 使用者集區提出的成功使用者身分驗證請求總數。當身分驗證權杖發行給使用者時，使用者身分驗證會視為成功。成功的身分驗證請求會產生值 1，而不成功的請求會產生值 0。調節的請求也遭認為是不成功的請求，因此調節的請求也將產生計數 0。 若要尋找成功的使用者身分驗證請求百分比，請使用此指標的 Average 統計數字。若要計算使用者身分驗證請求的總數，請使用此指標的 Sample Count 統計數字。若要計算成功的使用者身分驗證請求總數，請使用此指標的 Sum 統計數字。若要計算失敗的使用者驗證要求總數，請使用運算 CloudWatch Math式並從Sum統計資料中減去Sample Count統計值。 此指標針對每個用戶端的每個使用者集區發佈。如果請求中提供了無效的使用者集區用戶端，則指標中對應的使用者集區用戶端值會包含固定值 Invalid，而非請求中傳送的實際無效值。 請注意，此指標中不包含重新整理 Amazon Cognito 權杖的請求。有個別指標可以提供 Refresh 權杖統計資料。 指標維度：UserPool、UserPoolClient 單位：計數

指標	描述
SignInThrottles	提供對 Amazon Cognito 使用者集區提出的調節使用者身分驗證請求總數。每當調節使用者身分驗證請求時，就會發佈計數 1。 若要計算調節的使用者身分驗證請求總數，請使用此指標的 Sum 統計數字。 此指標針對每個用戶端的每個使用者集區發佈。如果請求中提供了無效的使用者集區用戶端，則指標中對應的使用者集區用戶端值會包含固定值 Invalid，而非請求中傳送的實際無效值。 此指標中不包含重新整理 Amazon Cognito 權杖的請求。有個別指標可以提供 Refresh 權杖統計資料。 指標維度：UserPool、UserPoolClient 單位：計數

指標	描述
TokenRefreshSuccesses	提供對 Amazon Cognito 使用者集區提出重新整理 Amazon Cognito 權杖的成功請求總數。成功的重新整理 Amazon Cognito 權杖請求會產生值 1，而不成功的請求會產生值 0。調節的請求也遭認為是不成功的請求，因此調節的請求也將產生計數 0。 若要尋找成功的重新整理 Amazon Cognito 權杖請求百分比，請使用此指標的 Average 統計數字。若要計算重新整理 Amazon Cognito 權杖請求的總數，請使用此指標的 Sample Count 統計數字。若要計算成功的重新整理 Amazon Cognito 權杖請求總數，請使用此指標的 Sum 統計數字。若要計算重新整理 Amazon Cognito 權杖的失敗請求總數，請使用運算 CloudWatchMath式並從Sum統計資料中減去統計資料。Sample Count 此指標會根據每個使用者集區用戶端發佈。如果請求中有無效的使用者集區用戶端，使用者集區用戶端值會包含固定值 Invalid。 指標維度：UserPool、UserPoolClient 單位：計數

指標	描述
TokenRefreshThrottles	提供對 Amazon Cognito 使用者集區提出重新整理 Amazon Cognito 權杖的調節請求總數。每當調節重新整理 Amazon Cognito 權杖請求時，就會發佈計數 1。 若要計算調節的重新整理 Amazon Cognito 權杖請求總數，請使用此指標的 Sum 統計數字。 此指標針對每個用戶端的每個使用者集區發佈。如果請求中提供了無效的使用者集區用戶端，則指標中對應的使用者集區用戶端值會包含固定值 Invalid，而非請求中傳送的實際無效值。 指標維度：UserPool、UserPoolClient 單位：計數
FederationSuccesses	提供對 Amazon Cognito 使用者集區提出的成功聯合身分請求總數。Amazon Cognito 向使用者發出身分驗證權杖時，則聯合身分視為成功。成功的聯合身分請求會產生值 1，而不成功的請求會產生值 0。限流請求和產生身分代碼但是沒有產生權杖的請求會產生值 0。 若要尋找成功的聯合身分請求百分比，請使用此指標的 Average 統計數字。若要計算聯合身分請求的總數，請使用此指標的 Sample Count 統計數字。若要計算成功的聯合身分請求總數，請使用此指標的 Sum 統計數字。若要Sum計算失敗的身分聯合要求總數，請使用運算 CloudWatch Math式並從統計資料中減去Sample Count統計值。 指標維度：UserPool、UserPoolClient 、IdentityProvider 單位：計數

指標	描述
FederationThrottles	提供對 Amazon Cognito 使用者集區提出的調節聯合身分請求總數。每當調節聯合身分請求時，就會發佈計數 1。 若要計算調節的聯合身分請求總數，請使用此指標的 Sum 統計數字。 指標維度：UserPool、UserPoolClient 、IdentityProvider 單位：計數
CallCount	提供與類別相關的客戶呼叫總數。此指標包含所有呼叫，例如調節呼叫、失敗呼叫和成功呼叫。 此指標可在 Usage (用量) nameSpace 取得。 系統會針對帳戶和區域中所有使用者集區中的每個 AWS 帳戶強制執行類別配額。 您可以使用此指標的 Sum 統計數字，計算類別中的呼叫總數。 指標維度：Service、Type、Resource、Class 單位：計數
ThrottleCount	提供與類別相關的調節呼叫總數。 此指標可在 Usage (用量) nameSpace 取得。 此指標在帳戶層級發佈。 您可以使用此指標的 Sum 統計數字，計算類別中的呼叫總數。 指標維度：Service、Type、Resource、Class 單位：計數

Amazon Cognito 使用者集區的維度

以下維度用於調整 Amazon Cognito 發佈的用量指標。維度僅適用於 `CallCount` 和 `ThrottleCount` 指標。

維度	描述
Service (服務)	包含資源的 AWS 服務名稱。對於 Amazon Cognito 用量指標，此維度的值為 `Cognito user pool`。
Type	正在報告的實體類型。Amazon Cognito 用量指標的唯一有效值為 API。
資源	正在執行的資源類型。唯一有效的值為類別名稱。
類別	正在追蹤的資源類別。Amazon Cognito 不會使用類別維度。

使用 Service Quotas 主控台追蹤指標

您可以利用 Service Quotas，從一個集中的位置查看和管理您的 Amazon Cognito 使用者集區和身份集區配額。您可以使用 Service Quotas 主控台來檢視特定配額的詳細資訊、監控配額使用率，以及要求提高配額。對於某些配額類型，您可以建立 CloudWatch警示來追蹤配額使用率。若要進一步了解您可以追蹤哪 Amazon Cognito 指標，請參閱追蹤配額使用量。

若要檢視 Amazon Cognito 使用者集區和身分集區服務配額使用率，請完成下列步驟。

1. 開啟 Service Quotas 主控台。
2. 在導覽窗格中，選擇 AWS services (AWS 服務)。
3. 從AWS 服務清單中搜尋並選擇 Amazon Cognito 使用者集區或 Amazon Cognito 聯合身分。服務配額頁面隨即顯示。
4. 選取支援 CloudWatch 監控的配額。例如，在 Amazon Cognito 使用者集區中選擇 `Rate of UserAuthentication requests`。
5. 向下捲動至 Monitoring (監控)。只有支援 CloudWatch 監控的配額才會顯示此區段。
6. 在 Monitoring (監控) 中，您能以圖表檢視目前的服務配額使用率。

7. 在 Monitoring (監控) 中，選取一小時、三小時、十二小時、一天、三天或一週。
8. 選取圖表內的任何區域，檢視服務配額使用率百分比。從這裡，您可以將圖形新增至儀表板，或使用動作功能表選取「在指標中檢視」，這會將您帶到 CloudWatch 主控台中的相關指標。

使用主 CloudWatch 控台追蹤指標

您可以使用追蹤和收集 Amazon Cognito 使用 CloudWatch者集區指標。 CloudWatch 儀表板將顯示有關您使用的每個 AWS 服務的指標。您可以使用 CloudWatch 建立度量警示。警示可設為傳送通知或變更您正在監控的特定資源。若要檢視中的服務配額度量 CloudWatch，請完成以下步驟。

1. 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇 Metrics (指標)。
3. 在 All metrics (所有指標) 中，選取指標和維度。
4. 選取指標旁的核取方塊。指標會顯示在圖表中。

> **Note**
>
> 過去兩週內沒有任何新資料點的指標不會顯示在主控台中。當您在主控台的 All metrics (所有指標) 索引標籤的搜尋方塊中輸入指標名稱或維度名稱時，它們也不會顯示，而且在 list-metrics 命令的結果中不會傳回這些名稱。擷取這些指標的最佳方法是使用 AWS CLI 中的 `get-metric-data` 或 `get-metric-statistics` 命令。

建立配額的 CloudWatch 警示

Amazon Cognito 提供與 `ThrottleCount` API 的 AWS 服務配額相對應的 CloudWatch 使用`CallCount`量指標。如需中追蹤使用情況的詳細資訊 CloudWatch，請參閱追蹤配額使用量。

在 Service Quotas 主控台中，您可以建立警示，在您的用量接近服務配額時提醒您。要了解如何使用 Service Quotas 控制台設置警 CloudWatch 報，請參閱 Service Quotas 和 CloudWatch 警報。

使用記錄 Amazon Cognito API 呼叫 AWS CloudTrail

Amazon Cognito 與這項服務整合在一起 AWS CloudTrail，可提供 Amazon Cognito 中使用者、角色或服務所採取的動作記錄的 AWS 服務。 CloudTrail 擷取 Amazon Cognito 的 API 呼叫子集作為事件，包括來自 Amazon Cognito 主控台的呼叫，以及從程式碼呼叫到 Amazon Cognito API 作業的呼叫。如果您建立追蹤，您可以選擇將 CloudTrail 事件傳遞到 Amazon S3 儲存貯體，包括 Amazon Cognito 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 Amazon Cognito 發出的請求、提出請求的 IP 地址、提出請求的人員、提出請求的時間以及其他詳細資訊。

若要進一步了解 CloudTrail，包括如何設定和啟用它，請參閱AWS CloudTrail 使用者指南。

您也可以為特定 CloudTrail 事件建立 Amazon CloudWatch 警示。例如，您可以設定 CloudWatch 為在身分識別集區組態變更時觸發警示。如需詳細資訊，請參閱建立 CloudTrail 事件 CloudWatch 警示：範例。

主題

- Amazon Cognito 信息 CloudTrail
- 了解 Amazon Cognito 登入事件
- 利用 Amazon CloudWatch 日誌洞察分析 Amazon Cognito CloudTrail 事件

Amazon Cognito 信息 CloudTrail

CloudTrail 在您建立您的 AWS 帳戶. Amazon Cognito 中發生受支援的事件活動時，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，請參閱使用 CloudTrail 事件歷程記錄檢視事件。

如需 AWS 帳戶中持續記錄事件 (包括 Amazon Cognito 的事件)，請建立追蹤。 CloudTrail 追蹤可將日誌檔案傳送到 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
- 設定亞馬遜 SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 日誌文件並從多個帳戶接收 CloudTrail日誌文件

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否使用根或 IAM 使用者憑證提出。
- 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。

中的機密資料 AWS CloudTrail

由於使用者集區和身分集區會處理使用者資料，因此 Amazon Cognito 會將 CloudTrail 事件中的某些私有欄位隱藏為該值。HIDDEN_FOR_SECURITY_REASONS如需 Amazon Cognito 未填入事件的欄位範例，請參閱 了解 Amazon Cognito 登入事件。Amazon Cognito 只會隱藏某些常包含使用者資訊的欄位，例如密碼和權杖。Amazon Cognito 不會對您填入 API 請求中非私有欄位的個人識別資訊執行任何自動偵測或遮蔽。

Amazon Cognito 使用者集區

Amazon Cognito 支援將 [使用者集區動作] 頁面上列出的所有動作記錄為記 CloudTrail 錄檔中的事件。Amazon Cognito 會將使用者集區事件記錄 CloudTrail 為管理事件。

Amazon Cognito 使用者集區 CloudTrail 項目中的`eventType`欄位會告訴您應用程式是向 Amazon Cognito 使用者集區 API 提出請求，還是向為 OpenID Connect、SAML 2.0 或託管使用者介面提供資源的端點提出請求。API 請求具有 `AwsApiCall` `eventType`，端點請求具有 `AwsServiceEvent` `eventType`。

Amazon Cognito 會將下列託管的 UI 請求記錄到託管的使用者介面，做為中 CloudTrail的事件。

託管的 UI 操作 CloudTrail

作業	描述
`Login_GET` ，`CognitoAuthentication`	使用者檢視或提交憑證至您的 登入端點。
`OAuth2_Authorize_GET` ，`Beta_Authorize_GET`	使用者檢視您的 授權端點。
`OAuth2Response_GET` ，`OAuth2Response_POST`	使用者向您的 `/oauth2/idpresponse` 端點提交 IdP 權杖。
`SAML2Response_POST` ，`Beta_SAML2Response_POST`	使用者向您的 `/saml2/idpresponse` 端點提交 IdP SAML 斷言。
`Login_OIDC_SAML_POST`	使用者在您的 登入端點 輸入使用者名稱以及相匹配的 IdP 識別碼。
`Token_POST` ，`Beta_Token_POST`	使用者向您的 權杖端點 提交授權碼。

作業	描述
Signup_GET , Signup_POST	使用者向您的 /signup 端點提交註冊資訊。
Confirm_GET , Confirm_POST	使用者在託管 UI 中提交確認碼。
ResendCode_POST	使用者在託管 UI 中提交重新發送確認碼的請求。
ForgotPassword_GET , ForgotPassword_POST	使用者向您的 /forgotPassword 端點提交重設密碼的請求。
ConfirmForgotPassword_GET , ConfirmForgotPassword_POST	使用者向確認其 ForgotPassword 請求的您的 /confirmForgotPassword 端點提交代碼。
ResetPassword_GET , ResetPassword_POST	使用者在託管的 UI 中提交新密碼。
Mfa_GET, Mfa_POST	使用者在託管的 UI 中提交多重要素驗證 (MFA) 代碼。
MfaOption_GET , MfaOption_POST	使用者在託管的 UI 中選擇他們偏好的 MFA 方法。
MfaRegister_GET , MfaRegister_POST	使用者在託管的 UI 中提交多重要素驗證 (MFA) 代碼。
Logout	使用者登出您的 /logout 端點。
SAML2Logout_POST	使用者登出您的 /saml2/logout 端點。
Error_GET	使用者在託管的 UI 中檢視錯誤頁面。
UserInfo_GET , UserInfo_POST	使用者或 IdP 與您的 UserInfo 端點 交換資訊。
Confirm_With_Link_GET	使用者根據 Amazon Cognito 在電子郵件中傳送的連結提交確認。

作業	描述
Event_Feedback_GET	使用者向 Amazon Cognito 提交有關進階安全性功能事件的反饋。

> **Note**
>
> Amazon Cognito 會記錄使用者特定請求的記 CloudTrail 錄，UserSub但不會記錄UserName在記錄中。您可以呼叫 ListUsers API並使用子使用者篩選條件，找到特定 UserSub 的使用者。

Amazon Cognito 身分集區

資料事件

Amazon Cognito 會將下列 Amazon Cognito 身分事件記錄 CloudTrail 為資料事件。資料事件是預設 CloudTrail 不會記錄的大容量資料平面 API 作業。資料事件需支付額外的費用。

- GetCredentialsForIdentity
- GetId
- GetOpenIdToken
- GetOpenIdTokenForDeveloperIdentity
- UnlinkIdentity

若要產生這些 API 作業的 CloudTrail 記錄，您必須啟動追蹤中的資料事件，並為 Cognito 身分集區選擇事件選取器。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的記錄資料事件。

您也可以使用下列 CLI 命令，將身分集區事件選取器新增至線索。

```
aws cloudtrail put-event-selectors --trail-name <trail name> --advanced-event-selectors
 \
"{\
   \"Name\": \"Cognito Selector\",\
   \"FieldSelectors\": [\
      {\
         \"Field\": \"eventCategory\",\
         \"Equals\": [\
```

```
            \"Data\"\
         ]\
      },\
      {\
         \"Field\": \"resources.type\",\
         \"Equals\": [\
            \"AWS::Cognito::IdentityPool\"\
         ]\
      }\
   ]\
}"
```

管理事件

Amazon Cognito 會將 Amazon Cognito 身分集區 API 作業的剩餘部分記錄為管理事件。 CloudTrail 依預設，記錄管理事件 API 作業。

如需 Amazon Cognito 記錄的 Amazon Cognito 身分識別集區 API 操作清單 CloudTrail，請參閱 Amazon Cognito 可身分集區 API 參考。

Amazon Cognito Sync

Amazon Cognito 將所有 Amazon Cognito Sync API 操作記錄為管理事件。如需 Amazon Cognito 記錄到的亞馬遜認知同步 API 操作清單 CloudTrail，請參閱 Amazon Cognito 同步 API 參考。

了解 Amazon Cognito 登入事件

追蹤可以將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。

主題

- 託管 UI 註冊的示例 CloudTrail 事件
- SAML 要求的範例 CloudTrail 事件
- 向令牌端點請求的示例 CloudTrail 事件
- 範例 CloudTrail 事件 CreateIdentityPool
- 範例 CloudTrail 事件 GetCredentialsForIdentity
- 範例 CloudTrail 事件 GetId

- 範例 CloudTrail 事件 GetOpenIdToken
- 範例 CloudTrail 事件 GetOpenIdTokenForDeveloperIdentity
- 範例 CloudTrail 事件 UnlinkIdentity

託管 UI 註冊的示例 CloudTrail 事件

下列範例 CloudTrail 事件示範使用者透過託管 UI 註冊時，Amazon Cognito 記錄的資訊。

當新使用者導覽至您應用程式的登入頁面時，Amazon Cognito 會記錄以下事件。


```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-04-06T05:38:12Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Login_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "errorCode": "",
    "errorMessage": "",
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200.0
        },
        "requestParameters":
        {
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "response_type":
            [
                "token"
            ],
            "client_id":
            [
```

```
                "1example23456789"
            ]
        }
    },
    "eventID": "382ae09a-151d-4116-8f2b-6ac0a804a38c",
    "readOnly": true,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

當新使用者在您應用程式的登入頁面選擇 Sign up (註冊) 時，Amazon Cognito 會記錄以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:21:43Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Signup_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "response_type":
            [
                "code"
```

```
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "7a63e7c2-b057-4f3d-a171-9d9113264fff",
    "eventID": "5e7b27a0-6870-4226-adb4-f86cd51ac5d8",
    "readOnly": true,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

當新使用者選擇使用者名稱、輸入電子郵件地址並在您應用程式的登入頁面選擇密碼時，Amazon Cognito 會記錄以下事件。Amazon Cognito 不會將有關使用者身分識別的識別資訊記錄到 CloudTrail。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:22:05Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Signup_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
```

```
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 302
        },
        "requestParameters":
        {
            "password":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "requiredAttributes[email]":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "response_type":
            [
                "code"
            ],
            "_csrf":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ],
            "username":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "9ad58dd8-3517-4aa8-96a5-d17a01df9eb4",
    "eventID": "c75eb7a5-eb8c-43d1-8331-f4412e756e69",
    "readOnly": false,
```

```
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

當新使用者在註冊後存取託管 UI 中的使用者確認頁面時，Amazon Cognito 會記錄以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:22:06Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Confirm_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "response_type":
            [
                "code"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
```

```
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "58a5b170-3127-45bb-88cc-3e652d779e0b",
    "eventID": "7f87291a-6d50-409a-822f-e3a5ec7e60da",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

在託管 UI 的使用者確認頁面中，當使用者輸入 Amazon Cognito 在電子郵件訊息中傳送給他們的代碼時，Amazon Cognito 會記錄以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:23:32Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Confirm_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 302
        },
```

```
        "requestParameters":
        {
            "confirm":
            [
                ""
            ],
            "deliveryMedium":
            [
                "EMAIL"
            ],
            "sub":
            [
                "704b1e47-34fe-40e9-8c41-504997494531"
            ],
            "code":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "destination":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "response_type":
            [
                "code"
            ],
            "_csrf":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "cognitoAsfData":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ],
            "username":
            [
```

```
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "9764300a-ed35-4f87-8a0f-b18b3fe2b11e",
    "eventID": "e24ac6e5-2f70-4c6e-ad4e-2f08a547bb36",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

SAML 要求的範例 CloudTrail 事件

當使用您的 SAML IdP 進行身分驗證的使用者向您的 `/saml2/idpresponse` 端點提交 SAML斷言時，Amazon Cognito 會記錄以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-06T00:50:57Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "SAML2Response_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 302
```

```
        },
        "requestParameters":
        {
            "RelayState":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "SAMLResponse":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "4f6f15d1-c370-4a57-87f0-aac4817803f7",
    "eventID": "9824b50f-d9d1-4fb8-a2c1-6aa78ca5902a",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "625647942648",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

向令牌端點請求的示例 CloudTrail 事件

以下為向 權杖端點 發出請求的範例事件。

當已經過身分驗證且收到授權碼的使用者向您的 /oauth2/token 端點提交授權碼時，Amazon Cognito 會記錄以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T22:12:30Z",
    "eventSource": "cognito-idp.amazonaws.com",
```

```
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "code":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "grant_type":
            [
                "authorization_code"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "f257f752-cc14-4c52-ad5b-152a46915238",
    "eventID": "0bd1586d-cd3e-4d7a-abaf-fd8bfc3912fd",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
```

```
    "eventCategory": "Management"
}
```

當您的後端系統向您的 /oauth2/token 端點提交存取權杖的 client_credentials 請求時，Amazon Cognito 會記錄以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T21:07:05Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "grant_type":
            [
                "client_credentials"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "4f871256-6825-488a-871b-c2d9f55caff2",
    "eventID": "473e5cbc-a5b3-4578-9ad6-3dfdcb8a6d34",
    "readOnly": false,
```

```
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

當您的應用程式與您的 /oauth2/token 端點交換重新整理權杖以取得新 ID 和存取權杖時，Amazon Cognito 會記錄以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T22:16:40Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "refresh_token":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "grant_type":
            [
                "refresh_token"
            ],
```

```
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "2829f0c6-a3a9-4584-b046-11756dfe8a81",
    "eventID": "12bd3464-59c7-44fa-b8ff-67e1cf092018",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

範例 CloudTrail 事件 CreateIdentityPool

下列範例是請求 `CreateIdentityPool` 動作的日誌項目。該請求是由名為 Alice 的 IAM 使用者提出。

```
{
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "['EXAMPLE_KEY_ID']",
        "userName": "Alice"
    },
    "eventTime": "2016-01-07T02:04:30Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "CreateIdentityPool",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "USER_AGENT",
    "requestParameters": {
```

```
        "identityPoolName": "TestPool",
        "allowUnauthenticatedIdentities": true,
        "supportedLoginProviders": {
            "graph.facebook.com": "000000000000000"
        }
    },
    "responseElements": {
        "identityPoolName": "TestPool",
        "identityPoolId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "allowUnauthenticatedIdentities": true,
        "supportedLoginProviders": {
            "graph.facebook.com": "000000000000000"
        }
    },
    "requestID": "15cc73a1-0780-460c-91e8-e12ef034e116",
    "eventID": "f1d47f93-c708-495b-bff1-cb935a6064b2",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

範例 CloudTrail 事件 GetCredentialsForIdentity

下列範例是請求 `GetCredentialsForIdentity` 動作的日誌項目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetCredentialsForIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-credentials-for-identity",
    "requestParameters": {
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "responseElements": {
```

```
        "credentials": {
            "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
            "sessionToken": "aAaAaAaAaAaAab1111111111EXAMPLE",
            "expiration": "Jan 19, 2023 5:55:08 PM"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "659dfc23-7c4e-4e7c-858a-1abce884d645",
    "eventID": "6ad1c766-5a41-4b28-b5ca-e223ccb00f0d",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

範例 CloudTrail 事件 GetId

下列範例是請求 `GetId` 動作的日誌項目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:05Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetId",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-id",
    "requestParameters": {
        "identityPoolId": "us-east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE",
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
```

```
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "dc28def9-07c8-460a-a8f3-3816229e6664",
    "eventID": "c5c459d9-40ec-41fd-8f6b-57865d5a9975",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

範例 CloudTrail 事件 GetOpenIdToken

下列範例是請求 GetOpenIdToken 動作的日誌項目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetOpenIdToken",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-open-id-token",
    "requestParameters": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
```

```
    },
    "requestID": "a506ba18-10d7-4fdb-9548-a8187b2e38bb",
    "eventID": "19ffc1a6-6ed8-4580-a4e1-3062c5ce6457",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

範例 CloudTrail 事件 GetOpenIdTokenForDeveloperIdentity

下列範例是請求 `GetOpenIdTokenForDeveloperIdentity` 動作的日誌項目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA1EXAMPLE:johns-AssumedRoleSession",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/johns-AssumedRoleSession",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA1EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "attributes": {
                "creationDate": "2023-01-19T16:53:14Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
```

```
    "eventName": "GetOpenIdTokenForDeveloperIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "27.0.3.154",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-open-id-token-for-developer-identity",
    "requestParameters": {
        "tokenDuration": 900,
        "identityPoolId": "us-east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE",
        "logins": {
            "JohnsDeveloperProvider": "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "b807df87-57e7-4dd6-b90c-b06f46a61c21",
    "eventID": "f26fed91-3340-4d70-91ae-cdf555547b76",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

範例 CloudTrail 事件 UnlinkIdentity

下列範例是請求 UnlinkIdentity 動作的日誌項目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "UnlinkIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
```

```
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.unlink-identity",
    "requestParameters": {
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "loginsToRemove": ["cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa"]
    },
    "responseElements": null,
    "requestID": "99c2c8e2-9c29-416f-bb17-b650a5cbada9",
    "eventID": "d8e26126-202a-43c2-b458-3f225efaedc7",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

利用 Amazon CloudWatch 日誌洞察分析 Amazon Cognito CloudTrail 事件

您可以使用 Amazon CloudWatch 日誌洞察來搜尋和分析您的 Amazon Cognito CloudTrail 事件。當您將追蹤設定為將事件傳送至 CloudWatch 記錄檔時，只 CloudTrail 會傳送符合追蹤設定的事件。

若要查詢或研究 Amazon Cognito CloudTrail 事件，請務必在 CloudTrail 主控台中選取追蹤設定中的管理事件選項，以便監控在 AWS 資源上執行的管理操作。想找出您帳戶中的錯誤、異常活動或異常使用者行為時，您可以選擇性選取追蹤設定中的 Insights 事件 選項。

Amazon Cognito 查詢範例

您可以在 Amazon CloudWatch 主控台中使用下列查詢。

一般查詢

尋找最近新增的 25 個日誌事件。

```
fields @timestamp, @message | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com"
```

取得最近新增的 25 個日誌事件 (包含例外狀況) 清單。

```
fields @timestamp, @message | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com" and @message like /Exception/
```

例外狀況和錯誤查詢

使用錯誤碼 NotAuthorizedException 及 Amazon Cognito 使用者集區 sub，尋找最近新增的 25 個日誌事件。

```
fields @timestamp, additionalEventData.sub as user | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com" and errorCode=
 "NotAuthorizedException"
```

使用 sourceIPAddress 和相應 eventName，尋找記錄數量。

```
filter eventSource = "cognito-idp.amazonaws.com"
| stats count(*) by sourceIPAddress, eventName
```

尋找觸發 NotAuthorizedException 錯誤的前 25 個 IP 地址。

```
filter eventSource = "cognito-idp.amazonaws.com" and errorCode=
 "NotAuthorizedException"
| stats count(*) as count by sourceIPAddress, eventName
| sort count desc | limit 25
```

尋找呼叫 ForgotPassword API 的前 25 個 IP 地址。

```
filter eventSource = "cognito-idp.amazonaws.com" and eventName = 'ForgotPassword'
| stats count(*) as count by sourceIPAddress
| sort count desc | limit 25
```

Amazon Cognito 的合規驗證

第三方稽核員會評估 Amazon Cognito 的安全性和合規性，做為多個 AWS 合規計劃的一部分。這些計劃包括 SOC、PCI、FedRAMP、HIPAA 等等。

如需特定法規遵循方案範圍內的 AWS 服務清單，請參閱合規計劃AWS 服務範圍)。如需一般資訊，請參閱 AWS 合規計劃。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載報表中的 AWS Artifact。

您使用 Amazon Cognito 時的合規責任取決於資料的敏感度、您的公司的合規目標，以及適用的法律和法規。 AWS 提供以下資源協助您處理合規事宜：

- 安全性與合規快速入門指南：這些部署指南會描述架構考量，並提供在 AWS上部署以安全性及合規為重心之基準環境的步驟。
- 建構 HIPAA 安全性與合規性白皮書 — 本白皮書說明公司如何使用建立符合 HIPAA 標準的應用 AWS 程式。
- 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。
- 使用AWS Config 開發人員指南中的規則評估資源 — AWS Config；評估您的資源配置如何符合內部實踐，業界準則和法規。
- AWS Security Hub— 此 AWS 服務提供安全狀態的全面檢視，協助您檢查您 AWS 是否符合安全性產業標準和最佳做法。

Amazon Cognito 的恢復能力

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。區域提供多個分開且隔離的實際可用區域，並以低延遲、高輸送量和高度備援網路連線相互連結。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需區域和可用區域的詳 AWS 細資訊，請參閱AWS 全域基礎結構。

主題

- 區域資料考量

區域資料考量

Amazon Cognito 使用者集區分別建立在一個 AWS 區域中，它們只會將使用者設定檔資料存放在該區域中。根據選用功能的設定方式，使用者集 AWS 區可以將使用者資料傳送至不同的區域。

- 如果使用預設的 `no-reply@verificationemail.com` 電子郵件地址設定來以 Amazon Cognito 使用者集區傳送電子郵件地址的驗證，則電子郵件會透過與相關聯使用者集區的相同區域傳送。

- 如果使用不同的電子郵件地址來設定具有 Amazon Cognito 使用者集區的 Amazon 簡易電子郵件服務 (Amazon SES)，則該電子郵件地址會透過與 Amazon SES 中電子郵件地址相關聯的 AWS 區域路由傳送。
- 來自 Amazon Cognito 使用者集區的簡訊會透過相同區域的 Amazon SNS 路由傳送，除非設定電子郵件或電話驗證另有說明。
- 如果 Amazon Cognito 使用者集區使用 Amazon Pinpoint 分析，事件資料會路由傳送到美國東部 (維吉尼亞北部) 區域。

> **Note**
>
> 亞 Amazon Pinpoint 可在北美、歐洲、亞洲和大洋洲的多個 AWS 區域使用。Amazon Pinpoint 區域包括 Amazon Pinpoint API。如果 Amazon Cognito 支援 Amazon Pinpoint 區域，Amazon Cognito 會將事件傳送到相同 Amazon Pinpoint 區域內的 Amazon Pinpoint 專案。如果 Amazon Pinpoint 不支援該區域，Amazon Cognito 僅支援在 us-east-1 中傳送事件。如需 Amazon Pinpoint 的詳細區域資訊，請參閱 Amazon Pinpoint 端點和配額及使用 Amazon Pinpoint 分析搭配 Amazon Cognito 使用者集區。

Amazon Cognito 的基礎設施安全

作為一項受管服務，Amazon Cognito 受到 AWS 全球網路安全的保護。有關 AWS 安全服務以及如何 AWS 保護基礎架構的詳細資訊，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 Amazon Cognito。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。

Amazon Cognito 使用者集區中的組態與漏洞分析

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：

- Amazon Cognito 的合規驗證
- 共同的責任模型

AWS Amazon Cognito 的受管政策

若要新增使用者、群組和角色的權限，使用 AWS 受管理的原則比自己撰寫原則更容易。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。若要快速開始使用，您可以使用我們的 AWS 受管政策。這些政策涵蓋常見使用案例，並可在您的 AWS 帳戶中使用。如需 AWS 受管政策的詳細資訊，請參閱 IAM 使用者指南中的AWS 受管政策。

AWS 服務會維護和更新 AWS 受管理的策略。您無法變更 AWS 受管理原則中的權限。服務偶爾會在 AWS 受管政策中新增其他許可以支援新功能。此類型的更新會影響已連接政策的所有身分識別 (使用者、群組和角色)。當新功能啟動或新操作可用時，服務很可能會更新 AWS 受管政策。服務不會從 AWS 受管理的政策移除權限，因此政策更新不會破壞您現有的權限。

此外，還 AWS 支援跨多個服務之工作職能的受管理原則。例如，ReadOnlyAccess AWS 受管理的策略提供對所有 AWS 服務和資源的唯讀存取。當服務啟動新功能時，會為新作業和資源新 AWS 增唯讀權限。如需任務職能政策的清單和說明，請參閱 IAM 使用者指南中有關任務職能的AWS 受管政策。

IAM 主控台中有許多政策可以讓您用來授予對 Amazon Cognito 的存取權：

- `AmazonCognitoPowerUser` - 存取及管理身分集區和使用者集區所有層面的許可。若要檢視此原則的權限，請參閱AmazonCognitoPowerUser。
- `AmazonCognitoReadOnly` – 身分集區和使用者集區唯讀存取的許可。若要檢視此原則的權限，請參閱AmazonCognitoReadOnly。
- `AmazonCognitoDeveloperAuthenticatedIdentities` – 讓您的身分驗證系統與 Amazon Cognito 整合的許可。若要檢視此原則的權限，請參閱AmazonCognitoDeveloperAuthenticatedIdentities。

這些政策是由 Amazon Cognito 團隊維護，所以即使有新的 API 加入，您的使用者還是繼續擁有相同層級的存取權。

> **Note**
>
> 當您建立新的身分集區時，您可以自動建立新角色供通過驗證的使用者和訪客使用者存取時使用。使用新 IAM 角色建立身分集區的管理員也必須具有 IAM 許可才能建立角色。

具有未驗證來賓存取的身分集區會套用額外的 AWS 受管理原則`AmazonCognitoUnAuthedIdentitiesSessionPolicy`，做為未驗證使用者的工作階段原則。此 AWS 受管理策略沒有預定的管理用途。不過，它會限制您可在身分集區增強型驗證流程中套用至訪客使用者的許可範圍。如需詳細資訊，請參閱 IAM 角色。

Amazon Cognito 更新 AWS 受管政策

檢視 Amazon Cognito AWS 受管政策更新的詳細資訊，因為此服務開始追蹤這些變更。如需有關此頁面變更的自動提醒，請訂閱 Amazon Cognito 文件歷程記錄頁面上的 RSS 摘要。

變更	描述	日期
`AmazonCognitoUnAuthedIdentitiesSessionPolicy` - 新政策	新增了針對身分 AWS 集區中來賓使用者權限範圍的受管理原則。	2023 年 7 月 14 日
`AmazonCognitoPowerUser` 和 `AmazonCognitoReadOnly` – 對現有政策的更新	新增許可，讓進階使用者能夠檢視和管理 AWS WAF 網路 ACL 與 Amazon Cognito 使用者集區的關聯。 新增了新許可，允許唯讀使用者檢視 AWS WAF 網路 ACL 與 Amazon Cognito 使用者集區的關聯。	2022 年 7 月 19 日

變更	描述	日期
AmazonCognitoPowerUser – 更新現有政策	新增許可，允許 Amazon Cognito 呼叫 Amazon Simple Email Service PutIdentityPolicy 和 ListConfigurationSets 操作。 這項變更可讓 Amazon Cognito 使用者集區更新 Amazon SES 傳送授權政策，並當您在使用者集區中設定傳送電子郵件時套用 Amazon SES 組態集。	2021 年 11 月 17 日
AmazonCognitoPowerUser – 更新現有政策	新增許可，允許 Amazon Cognito 呼叫 Amazon Simple Notification Service 的 GetSMSSandboxAccountStatus 操作。 這項變更可讓 Amazon Cognito 使用者集區決定是否需淘汰 Amazon Simple Notification Service 沙盒，以便透過使用者集區傳送訊息給所有最終使用者。	2021 年 6 月 1 日
Amazon Cognito 開始追蹤變更	Amazon Cognito 開始追蹤其 AWS 受管政策的變更。	2021 年 3 月 1 日

標記 Amazon Cognito 資源

標籤是您或 AWS 指派給 AWS 資源的中繼資料標籤。每個標籤皆包含鍵與值。對於您指派的標籤，您可以定義鍵與值。例如，您可以將鍵定義為 `stage`，將資源的值定義為 `test`。

標籤可協助您執行以下操作：

- 識別和組織您的 AWS 資源。許多 AWS 服務支援標記，因此您可以對來自不同服務的資源指派相同的標籤。這可協助您指出哪些資源是相關的。例如，您可以將指派給 Amazon DynamoDB 資料表的相同標籤指派給 Amazon Cognito 使用者集區。
- 追蹤您的 AWS 成本。您可以在 AWS Billing and Cost Management 儀表板上啟用這些標籤。AWS 會使用成本配置標籤分類您的成本，並交付每月成本配置報告給您。如需詳細資訊，請參閱《AWS Billing 使用者指南》中的使用成本配置標籤。
- 根據資源獲派的標籤，控制資源的存取權。您可以透過在 AWS Identity and Access Management (IAM) 政策條件中指定標籤索引鍵和值，控制存取。例如，您可以允許使用者更新使用者集區，但該使用者集區的 `owner` 標籤值必須包含該使用者名稱。如需詳細資訊，請參閱《IAM 使用者指南》中的使用標籤控制存取權限。

您可以使用 AWS Command Line Interface 或 Amazon Cognito API 來新增、編輯或刪除使用者集區和身分集區的標籤。您還可以使用 Amazon Cognito 主控台來管理使用者集區的標籤。

如需使用標籤的提示，請參閱 AWS 回答部落格上的 AWS 標記策略文章。

以下章節提供 Amazon Cognito 標籤的詳細資訊。

Amazon Cognito 中支援的資源

Amazon Cognito 中的以下資源支援標記：

- 使用者集區
- 身分集區

標籤限制

以下限制適用於 Amazon Cognito 資源上的標籤：

- 您可以指派給資源的標籤數量上限 – 50

- 金鑰長度上限 - 128 個 Unicode 字元
- 數值長度上限 - 256 個 Unicode 字元
- 索引鍵與值的有效字元 – a-z、A-Z、0-9、空格和下列字元：_ . : / = + - @
- 金鑰和值會區分大小寫
- 請不要使用 `aws:` 做為金鑰的字首；它已保留供 AWS 使用

使用 Amazon Cognito 主控台管理標籤

您可以使用 Amazon Cognito 主控台，管理指派給您的使用者集區的標籤。

新增標籤至使用者集區

1. 導覽至 Amazon Cognito 主控台。若出現提示，請輸入 AWS 憑證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。
4. 選擇 User pool properties (使用者集區屬性) 索引標籤並找到 Tags (標籤)。
5. 選擇 Add tags (新增標籤) 以新增您的第一個標籤。如果您先前在 Manage tags (管理標籤) 中已將標籤指派給此使用者集區，則請選擇 Add another (新增另一個)。
6. 指定 Tag key (標籤金鑰) 和 Tag value (標籤值) 的值。
7. 對於您要新增的每個額外標籤，請選擇 Add another (新增另一個)。
8. 當您完成新增標籤的作業時，請選擇 Save changes (儲存變更)。

在 Manage tags (管理標籤) 頁面上，您也可以編輯任何現有標籤的索引鍵和值。若要移除標籤，請選擇 Remove (移除)。

AWS CLI 範例

AWS CLI 提供的命令，可協助您管理指派給您 Amazon Cognito 使用者集區和身分集區的標籤。

指派標籤

使用下列命令，將標籤指派給現有的使用者集區和身分集區。

Example 使用者集區適用的 **`tag-resource`** 命令

使用 `cognito-idp` 命令集內的 `tag-resource` 指派標籤給使用者集區：

```
$ aws cognito-idp tag-resource \
> --resource-arn user-pool-arn \
> --tags Stage=Test
```

此命令包括下列參數：

- `resource-arn` – 您要將標籤套用其中之使用者集區的 Amazon Resource Name (ARN)。若要查詢 ARN，請在 Amazon Cognito 主控台中選擇使用者集區，然後檢視 General settings (一般設定) 索引標籤上的 Pool ARN (集區 ARN) 值。
- `tags` – 標籤的鍵值組，格式為 *key*=*value*。

若要一次指派多個標籤，請以逗號分隔清單指定這些標籤：

```
$ aws cognito-idp tag-resource \
> --resource-arn user-pool-arn \
> --tags Stage=Test,CostCenter=80432,Owner=SysEng
```

Example 身分集區適用的 **tag-resource** 命令

使用 `cognito-identity` 命令集內的 `tag-resource` 指派標記給身分集區：

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
> --tags Stage=Test
```

此命令包括下列參數：

- `resource-arn` – 您要將標籤套用其中之身分集區的 Amazon Resource Name (ARN)。若要查詢 ARN，請在 Amazon Cognito 主控台中選擇該身分集區，然後選擇 Edit identity pool (編輯身分集區)。接著，在 Identity pool ID (身分集區 ID) 中，選擇 Show ARN (顯示 ARN)。
- `tags` – 標籤的鍵值組，格式為 *key*=*value*。

若要一次指派多個標籤，請以逗號分隔清單指定這些標籤：

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
> --tags Stage=Test,CostCenter=80432,Owner=SysEng
```

檢視標籤

使用下列命令，檢視您已指派給您的使用者集區和身分集區的標籤。

Example 使用者集區適用的 **list-tags-for-resource** 命令

使用 cognito-idp 命令集內的 list-tags-for-resource 檢視指派給使用者集區的標籤：

```
$ aws cognito-idp list-tags-for-resource --resource-arn user-pool-arn
```

Example 身分集區適用的 **list-tags-for-resource** 命令

使用 cognito-identity 命令集內的 list-tags-for-resource 檢視指派給身分集區的標籤：

```
$ aws cognito-identity list-tags-for-resource --resource-arn identity-pool-arn
```

移除標籤

使用下列命令，將標籤自您的使用者集區和身分集區上移除。

Example 使用者集區適用的 **untag-resource** 命令

使用 cognito-idp 命令集內的 untag-resource 從使用者集區移除標籤：

```
$ aws cognito-idp untag-resource \
> --resource-arn user-pool-arn \
> --tag-keys Stage CostCenter Owner
```

對於 --tag-keys 參數，請指定一或多個標籤索引鍵。請勿包含標籤值。請使用空格分隔索引鍵。

Example 身分集區適用的 **untag-resource** 命令

使用 cognito-identity 命令集內的 untag-resource 從身分集區移除標籤：

```
$ aws cognito-identity untag-resource \
> --resource-arn identity-pool-arn \
> --tag-keys Stage CostCenter Owner
```

對於 --tag-keys 參數，請指定一或多個標籤索引鍵。請勿包含標籤值。

> **Important**
>
> 刪除使用者或身分集區後，與所刪除集區相關的標籤在刪除後仍然會顯示於主控台或 API 呼叫中最多 30 天。

在建立資源時套用標籤

使用下列命令，在您建立使用者集區或身分集區時指派標籤。

Example 搭配標籤的 **create-user-pool** 命令

當您使用 create-user-pool 命令建立使用者集區時，您可以搭配 --user-pool-tags 參數指定標籤：

```
$ aws cognito-idp create-user-pool \
> --pool-name user-pool-name \
> --user-pool-tags Stage=Test,CostCenter=80432,Owner=SysEng
```

標籤的鍵值組必須是 *key*=*value* 格式。如果是新增多個標籤，請以逗號分隔清單指定這些標籤。

Example 搭配標籤的 **create-identity-pool** 命令

當您使用 create-identity-pool 命令建立身分集區時，您可以搭配 --identity-pool-tags 參數指定標籤：

```
$ aws cognito-identity create-identity-pool \
> --identity-pool-name identity-pool-name \
> --allow-unauthenticated-identities \
> --identity-pool-tags Stage=Test,CostCenter=80432,Owner=SysEng
```

標籤的鍵值組必須是 *key*=*value* 格式。如果是新增多個標籤，請以逗號分隔清單指定這些標籤。

使用 Amazon Cognito API 管理標籤

您可以在 Amazon Cognito API 中使用下列動作，為您的使用者集區和身分集區管理標籤。

使用者集區標籤適用的 API 動作

使用以下 API 動作，指派、檢視和移除使用者集區的標籤。

- TagResource
- ListTagsForResource
- UntagResource
- CreateUserPool

身分集區標籤適用的 API 動作

使用以下 API 動作，指派、檢視和移除身分集區的標籤。

- TagResource
- ListTagsForResource
- UntagResource
- CreateIdentityPool

Amazon Cognito 的配額

對於您可在帳戶中執行的操作數目上限，Amazon Cognito 有預設配額 (先前稱為限制)。Amazon Cognito 也具有 Amazon Cognito 資源數目上限和大小的配額。

每個 Amazon Cognito 配額代表一合一的請求數量 AWS 區域 上限。 AWS 帳戶例如，您的應用程式可以針對美國東部 (維吉尼亞北部) 中所有使用者集區的 `UserAuthentication` 操作提出最多達預設配額 (RPS) 速率的 API 請求。您在亞太區域 (東京) 的應用程式可以針對自己區域中的所有使用者集區產生相同數量的請求。 AWS 一次只能在一個地區授予配額增加要求。在美國東部 (維吉尼亞北部) 成功增加配額，不會影響您在亞太區域 (東京) 的最大請求率。

主題

了解 API 請求率配額

配額分類

Amazon Cognito 會強制執行 API 作業的最高請求率。如需 Amazon Cognito 提供之 API 操作的詳細資訊，請參閱 Amazon Cognito API 和端點參考 對於使用者集區，這些操作會分為常見使用案例類別，例如 `UserAuthentication` 或 `UserCreation`。如需按類別分類的使用者集區 API 作業清單，請參閱Amazon Cognito 使用者集區 API 操作類別和請求率配額。

在 Ser vice Quotas 主控台中，您可以按類別使用者集區和身分集區來追蹤配額使用情況。如果 Amazon Cognito 使用者集區的請求率或超過配額，則可以購買額外容量。您可以在 Service Quotas 主控台中，依類別追蹤使用者集區配額使用情況，以及購買配額增加情況。

操作配額的定義為同一類別中，所有操作的每秒請求 (RPS) 數目上限。Amazon Cognito 使用者集區服務會將配額套用於每個類別中的所有操作。例如，`UserCreation` 類別包括四個操作：`SignUp`、`ConfirmSignUp`、`AdminCreateUser` 和 `AdminConfirmSignUp`。它會分配到 50 個 RPS 的合併配額。若同時進行多個操作，此類別中的每個操作最多可呼叫 50 個 RPS (單獨或合併計算)。

> **Note**
>
> 類別配額僅適用於使用者集區。Amazon Cognito 會將每個身分集區配額套用至單一操作。對於每個類別和每個操作的請求率配額， AWS 測量來自您 AWS 帳戶 在一個區域中所有用戶集區或身份集區的所有請求的總計速率。

適用特殊請求率處理的 Amazon Cognito 使用者集區 API 操作

操作配額是針對類別層級的合併請求總數計算並強制套用，但 AdminRespondToAuthChallenge 和 RespondToAuthChallenge 操作除外，這兩者適用特殊處理規則。

該UserAuthentication類別包括 Amazon Cognito 使用者集區 API 中的四項操作：AdminInitiateAuthInitiateAuth、AdminRespondToAuthChallenge、和RespondToAuthChallenge。此外，託管 UI 中的使用者驗證也有助於此配額。InitiateAuth 和 AdminInitiateAuth 操作會根據每個類別配額計算和強制執行。相符的 RespondToAuthChallenge 和 AdminRespondToAuthChallenge 操作適用另外的配額，是 UserAuthentication 類別限制的三倍。這種提高的配額可滿足您在應用程式中設定的多個驗證挑戰。此配額足以涵蓋大多數使用案例。在您的應用程序對身份驗證挑戰提出最多三個響應後，其他請求將計入UserAuthentication類別配額中。多重要素驗證 (MFA)、裝置驗證和自訂驗證都是挑戰提示的範例，您可能會設計到使用者集區中。

例如，如果您的UserAuthentication類別配額為 80 RPS，您可以撥打電話RespondToAuthChallenge或AdminRespondToAuthChallenge以高達 240 RPS (3* 80 RPS) 的速率撥打電話。如果您的使用者集區提示每個驗證進行四輪挑戰，而每秒有 70 個使用者登入，則總計RespondToAuthChallenge為 280 RPS (70 x 4)，這比配額高出 40 個 RPS。額外 40 個 RPS 將新增至 70 個 InitiateAuth 呼叫，使 UserAuthentication 類別的總用量成為 110 (40 + 70) 個 RPS。由於這個值超過設定為 80 RPS 的類別配額，因此 Amazon Cognito 會從您的應用程式調節請求。

每月作用中使用者

Amazon Cognito 計算使用者集區帳單時，會向您收取每個月作用中使用者 (MAU) 的費率。在規劃配額增加請求時，考慮您目前和預計的 MAU 計數。如果在一個日曆月內，存在與該使用者相關的身分操作，則該使用者計為 MAU。使用者處於作用中的活動包括以下內容。

- 使用者的註冊或管理之建立
- 登入

- 登出
- 使用者帳戶確認或屬性驗證
- 密碼重設
- 變更使用者屬性、群組成員資格或 MFA 偏好設定
- 查詢使用者的詳細屬性
- 使用者啟用、停用或刪除

> **Note**
>
> 用戶的類別查詢詳細屬性包括 API 操作 AdminGetUser，但不包括ListUsers。大型使用者集區中的詳細 user-by-user 查詢可能會對您的 AWS 帳單產生重大影響。為避免超額費用，請使用外部資料庫收集使用者資料ListUsers或將使用者資訊儲存在外部資料

管理 API 請求速率配額

確認配額需求

> **Important**
>
> 如果您增加、或等類別的 Amazon Cognito 配額 UserAuthentication UserCreationAccountRecovery，您可能需要增加其他 AWS 服務類別的配額。例如，如果這些服務的請求率配額不足，則 Amazon Cognito 使用 Amazon Simple Notification Service (Amazon SNS) 和 Amazon Simple Email Service (Amazon SES) 傳送的訊息可能會失敗。

若要計算配額需求，請決定在特定時段內與您應用程式互動的作用中使用者數量。舉例來說，如果您預期應用程式在八小時內平均有 100 萬個作用中使用者登入，則您必須能每秒平均對 35 個使用者進行身分驗證。

此外，如果您假設使用者工作階段平均為兩個小時，且權杖設定為一小時後過期，則每位使用者必須在此工作階段期間重新整理其權杖一次。於是，支援此負載的 UserAuthentication 類別所需平均配額為 70 個 RPS。

如果您假設 peak-to-average 比率為 3:1，計算八小時期間內使用者登入頻率的差異，則需要 200 RPS 所需的UserAuthentication配額。

> **Note**
>
> 如果您為每個使用者動作呼叫多個操作，必須在類別層級加總個別操作呼叫速率。

針對配額限制最佳化請求率

由於提高 API 費率限制會增加帳 AWS 單成本，因此請先考慮調整使用模式，然後再請求提高配額。以下是優化請求率的應用程序體系結構的一些示例。

在退避等候期間後重試

您可以在每次 API 呼叫時擷取錯誤，然後在退避期間後重試。您可以根據業務需求和負載，調整退避演算法。Amazon 開發套件有內建重試邏輯。如需詳細資訊，請參閱建置在其上的工具 AWS。

為經常更新的屬性使用外部資料庫

如果您的應用程式需要多次呼叫使用者集區來讀取或寫入自訂屬性，請使用外部儲存空間。您可以使用偏好的資料庫來存放自訂屬性，或使用快取層在登入期間載入使用者描述檔。您可以在需要時從快取參考此設定檔，而不必從使用者集區重新載入使用者描述檔。

在客戶端驗證 JSON 網絡令牌（JWT）

應用程式必須先驗證 JWT 權杖，才能信任該權杖。您可以在客戶端驗證令牌的簽名和有效性，而無需向用戶池發送 API 請求。驗證權杖後，您可以信任權杖中的宣告，並使用該宣告，而無需發出更多 `getUser` API 呼叫。如需詳細資訊，請參閱驗證 JSON Web 權杖。

使用等候室調節 Web 應用程式的流量

如果您預期在限時事件期間 (例如應試或參加即時活動) 會有大量使用者登入流量，可以使用自我調節機制來最佳化請求流量。例如，您可以設定等候室，讓使用者在工作階段可供使用之前先行等待，以利您在有可用容量時處理請求。如需等候室的參考架構，請參閱 AWS 虛擬等候室解決方案。

快取 JWT

重複使用訪問令牌直到它們過期。如需在 API Gateway 中具有權杖快取的範例架構，請參閱快取權杖。而不是生成 API 請求來查詢用戶信息，而是緩存 ID 令牌直到它們過期，並從緩存中讀取用戶屬性。

如需中使用 API 要求率的詳細資訊 AWS，請參閱管理和監控工作負載中的 API 節流。如需最佳化 Amazon Cognito 操作以增加 AWS 帳單成本的相關資訊，請參閱管理成本。

追蹤配額使用量

Amazon Cognito 會針對帳戶層級的每 CloudWatch 個 API 操作類別在 Amazon 產生`CallCount`和`ThrottleCount`指標。您可以使用 `CallCount` 追蹤客戶所發出與類別相關的呼叫總數。您可以使用 `ThrottleCount` 追蹤與類別相關的調節呼叫總數。您可以使用具 `Sum` 統計數字的 `CallCount` 和 `ThrottleCount` 指標，計算一個類別的呼叫總數。如需詳細資訊，請參閱CloudWatch 使用量度。

監控服務配額時，使用率是指使用中的服務配額百分比。舉例來說，如果配額值為 200 個資源，其中 150 個資源正在使用中，則使用率為 75%。使用量是指服務配額中使用的資源或操作數量。

透過指標追蹤使用 CloudWatch 量

您可以使用追蹤和收集 Amazon Cognito 使用者集區使用率指標 CloudWatch。 CloudWatch儀表板會顯示您使用的每個項目 AWS 服務 的相關指標。使用 CloudWatch，您可以建立指標警示，以通知您或變更您正在監視的特定資源。如需有關 CloudWatch 指標的詳細資訊，請參閱追蹤您的 CloudWatch 使用量度。

透過 Service Quotas 指標追蹤使用率

Amazon Cognito 使用者集區與 Service Quotas 整合，這是一個主控台界面，用於顯示和管理您的服務配額使用情況。在 Service Quotas 主控台中，您可以查詢特定配額的值、檢視監視資訊、要求增加配額或設定 CloudWatch 警示。在您的帳號有效一段時間後，您可以檢視資源使用率圖表。

Amazon Cognito 使用者集區的 Service Quotas 主控台和 Amazon Cognito 身分集區的已套用帳戶層級配額值欄會顯示您目前的配額。「使用率」欄會顯示您目前的配額使用率。可調整的 Amazon Cognito 使用者集區 requests-per-second (RPS) 配額會顯示其目前的使用量。「Service Quotas」主控台也可以導覽至 CloudWatch 指標，以進一步查看選取的配額指標。如需在 Service Quotas 主控台中檢視配額的詳細資訊，請參閱檢視 Service Quotas。

追蹤每月作用中使用者 (MAU)

使用者集區中的每月作用中使用者 (MAU) 數量會為您的規劃提供重要資料，以提高請求率配額。您可以將 API 請求率與指定時間段內活躍的用戶數進行比較。有了這些知識，您就可以計算應用程式作用中使用者的增加將如何影響使用模式中的配額。例如，假設您在美國西部 (奧勒岡) 的合併應用程式在一個月內產生 200 萬活躍使用者，而且您的`UserAuthentication`類別偶爾會收到限制錯誤，預設配額為每秒 120 個請求 (RPS)。在上個月，在您成功的廣告活動之前，您擁有 100 萬個 MAU，而您的應用程式從未超過 80 RPS。如果您預計會因為新的電視節目而產生類似的峰值，則可以額外購買 40 RPS，以容納下一百萬名使用者，調整後的配額為 160 RPS。

若要檢視您的 MAU

存取主AWS Billing 控台並檢閱最近的帳單。在「依服務收費」下方，您可以篩選 Cognito，以檢視該帳單週期內 MAU 的明細。

請求提高配額

Amazon Cognito 擁有每秒可在每個 AWS 區域使用者集區和身分集區中執行的最大操作數量的配額。您可以購買可調整的 Amazon Cognito 使用者集區 API 請求率配額的增加金額。透過 Service Quotas 主控台或 Service Quotas API 作業和，檢查您目前的配額`ListAWSDefaultServiceQuotas`並增加購買額度`RequestServiceQuotaIncrease`。

- 若要使用 Service Quotas 主控台購買配額增加，請參閱《Service Quotas 使用者指南》中的要求增加 API 配額。
- AWS 目標是在 10 天內完成配額增加要求。不過，有幾項考量可能會導致要求處理時間超過 10 天。例如，某些請求可能需要 Amazon Cognito 佈建額外的硬體容量，而請求數量的季節性增加可能會導致延遲。
- 如果 Service Quotas 中沒有提供配額，請使用增加服務配額表單。

> **Important**
>
> 只能增加可調整的配額。您必須購買增加配額容量。如需加價的定價資訊，請參閱 Amazon Cognito 定價。

Amazon Cognito 使用者集區 API 操作類別和請求率配額

由於 Amazon Cognito 對於不同的授權模型具有重疊的 API 操作類別，因此每個操作都屬於一個類別。每個類別都有自己的集區配額，用於所有成員 API 操作，橫跨您帳戶中一個 AWS 區域 的所有使用者集區。您只能請求增加可調整類別配額。如需詳細資訊，請參閱 請求提高配額。配額調整會套用至您帳戶單一區域中的使用者集區。Amazon Cognito 限制某些類別 [3] 的操作為每個使用者集區每秒 5 個請求 (RPS)。預設配額 (RPS) 還適用於. AWS 帳戶

> **Note**
>
> 每個類別的配額測量單位為每月作用中使用者 (MAU) 數量。MAU 少於 200 萬的 AWS 帳戶 可以在預設配額內操作。如果您有少於一百萬個 MAU，而 Amazon Cognito 正在調節請求，請考慮最佳化您的應用程式。如需詳細資訊，請參閱 針對配額限制最佳化請求率。

類別操作配額會套用於一個 AWS 區域的使用者集區中所有使用者。Amazon Cognito 也會為您的應用程式可針對一個使用者產生的請求數量維護配額。您必須用下表顯示的方式，限制各使用者 API 請求。

Amazon Cognito 使用者集區各使用者請求率配額

作業	每使用者每秒操作數
讀取使用者描述檔 範例: `GetUser`, `GetDevice`	10
寫入使用者描述檔 範例: `UpdateUserAttributes` ， `SetUserSettings`	10

您必須用下表顯示的方式，限制分類 API 請求。

Amazon Cognito 使用者集區 分類請求率配額

類別	描述	預設配額 (RPS)	可調整
`UserAuthentication` • InitiateAuth • 使用 `InitiateAuth` 或 權杖端點 重新整理權杖	驗證 (登入) 使用者的操作。 這些操作受 適用特殊請求率處理的 Amazon Cognito 使用者集區 API 操作 規範。	120	是

類別	描述	預設配額 (RPS)	可調整
• RespondToAuthChallenge[1] • AdminInitiateAuth • AdminRespondToAuthChallenge[1] • 授權碼或隱含授予[2]中的託管 UI 登入和 MFA			
`UserCreation` • SignUp • ConfirmSignUp • AdminCreateUser • AdminConfirmSignUp	建立或確認 Amazon Cognito 本機使用者的操作。原生使用者是您的 Amazon Cognito 使用者集區直接建立和驗證的使用者。	50	是
`UserFederation` 透過第三方身分提供者將使用者聯合 (驗證) 至您 Amazon Cognito 使用者集區的操作。	將 IdP 回應提交至使用者集區聯合端點的操作。產生 IdP 權杖的 OIDC 或社交供應商操作，以及所有 SAML 請求都會計入此配額。	25	是

類別	描述	預設配額 (RPS)	可調整
`UserAccountRecovery` • ChangePassword • ConfirmForgotPassword • ForgotPassword • AdminResetUserPassword • AdminSetUserPassword • RespondToAuthChallenge[1] • AdminRespondToAuthChallenge[1] • 託管 UI 密碼重設	復原使用者帳戶，或者變更或更新使用者密碼的操作。	30	否
`UserRead` • AdminGetUser • GetUser	從使用者集區擷取使用者的操作。	120	是

類別	描述	預設配額 (RPS)	可調整
UserUpdate • AdminAddUserToGroup • AdminDeleteUserAttributes • AdminUpdateUserAttributes • AdminDeleteUser • AdminDisableUser • AdminEnableUser • AdminLinkProviderForUser • AdminDisableProviderForUser • VerifyUserAttribute • DeleteUser • DeleteUserAttributes • UpdateUserAttributes • AdminUserGlobalSignOut • GlobalSignOut • AdminRemoveUserFromGroup	用來管理使用者和使用者屬性的操作。	25	否
UserToken • RevokeToken	權杖管理操作	120	是

類別	描述	預設配額 (RPS)	可調整
`UserResourceRead` • AdminGetDevice • AdminListGroupsForUser • AdminListDevices • GetDevice • ListDevices • GetUserAttributeVerificationCode • ResendConfirmationCode • AdminListUserAuthEvents	從 Amazon Cognito 擷取使用者資源資訊 (例如記住的裝置或群組成員資格) 的操作。	50	是

類別	描述	預設配額 (RPS)	可調整
`UserResourceUpdate` • AdminForgetDevice • AdminUpdateAuthEventFeedback • AdminSetUser多功能事務參考 • AdminSetUserSettings • AdminUpdateDeviceStatus • UpdateDeviceStatus • UpdateAuthEventFeedback • ConfirmDevice • SetUser多功能事務參考 • SetUserSettings • VerifySoftwareToken • AssociateSoftwareToken • ForgetDevice	更新使用者資源資訊 (例如記住的裝置或群組成員資格) 的操作。	25	否
`UserList` • ListUsers • ListUsersInGroup	傳回使用者清單的操作。	30	否

類別	描述	預設配額 (RPS)	可調整
`UserPoolRead` • DescribeUserPool • ListUserPools	讀取使用者集區的操作。	15	否
`UserPoolUpdate` • CreateUserPool • UpdateUserPool • DeleteUserPool	建立、更新或刪除使用者集區的操作。	15	否

類別	描述	預設配額 (RPS)	可調整
`UserPoolResourceRead` • DescribeIdentityProvider • DescribeResourceServer • DescribeUserImportJob • DescribeUserPoolDomain • GetCSVHeader • GetGroup • GetSigningCertificate • GetIdentityProviderByIdentifier • GetUserPoolMfaConfig • ListGroups • ListIdentityProviders • ListResourceServers • ListTagsForResource • ListUserImportJobs • DescribeRiskConfiguration • GetUICustomization	從使用者集區擷取資源 (例如群組或資源伺服器) 相關資訊的操作。[3]	20	否

類別	描述	預設配額 (RPS)	可調整
`UserPoolResourceUpdate` • AddCustomAttributes • CreateGroup • CreateIdentityProvider • CreateResourceServer • CreateUserImportJob • CreateUserPoolDomain • DeleteGroup • DeleteIdentityProvider • DeleteResourceServer • DeleteUserPoolDomain • SetUserPoolMfaConfig • StartUserImportJob • StopUserImportJob • UpdateGroup • UpdateIdentityProvider • UpdateResourceServer • UpdateUserPoolDomain	修改使用者集區中資源 (例如群組或資源伺服器) 的操作。[3]	15	否

類別	描述	預設配額 (RPS)	可調整
• SetRiskConfiguration • SetUICustomization • TagResource • UntagResource			
UserPoolClientRead • DescribeUserPoolClient • ListUserPoolClients	擷取使用者集區用戶端相關資訊的操作。[3]	15	否
UserPoolClientUpdate • CreateUserPoolClient • DeleteUserPoolClient • UpdateUserPoolClient	建立、更新和刪除使用者集區用戶端的操作。[3]	15	否
ClientAuthentication client_credentials 會向權杖端點授予類型請求。	產生認證的作業，以用於授權 machine-to-machine 要求	150	否

[1] A RespondToAuthChallenge 或AdminRespondToAuthChallenge回應，其中ChallengeName一個NEW_PASSWORD_REQUIRED計入UserAccountRecovery類別。所有其他挑戰回應都會計入該UserAuthentication類別。

[2] 登入期間的每個託管 UI 作業都會向配額提供一個要求。例如，登入並提供 MFA 程式碼的使用者會提供 2 個請求。授權代碼授予中的代幣兌換需要以與您在類別中的配額相同的比率進行額外配額分配。UserAuthentication

[3] 此分類中的任何個別作業都有限制，可防止以高於 5 RPS 的速率呼叫單一使用者集區的作業。

Amazon Cognito 身分集區 (聯合身分) API 操作請求率配額

作業	描述	預設配額 (RPS)[1]	可調整	配額增加資格
GetId	從身分集區擷取身分 ID。	25	是	請聯絡您的帳戶團隊。
GetOpenIdToken	在傳統工作流程中從使用者集區擷取 OpenID 權杖。	200	是	請聯絡您的帳戶團隊。
GetCredentialsForIdentity	從增強型工作流程中的身分集區擷取 AWS 認證。	200	是	請聯絡您的帳戶團隊。
GetOpenIdTokenForDeveloperIdentity	在開發人員工作流程中從身分集區擷取 OpenID 權杖。	50	是	請聯絡您的帳戶團隊。
ListIdentities	從身分池擷取身分 ID 清單。	5	是	請聯絡您的帳戶團隊。
DeleteIdentities	從身分池集區刪除一個或多個已註冊的身分。	10	是	請聯絡您的帳戶團隊。
TagResource	將索引標籤套用至身分池。	5	是	請聯絡您的帳戶團隊。

作業	描述	預設配額 (RPS)[1]	可調整	配額增加資格
UntagResource	從身分池移除索引標籤。	5	是	請聯絡您的帳戶團隊。
ListTagsForResource	顯示套用至身分池的索引標籤清單。	10	是	請聯絡您的帳戶團隊。

[1] 預設配額是您的任 AWS 區域 一身分識別集區的最低要求率配額 AWS 帳戶。在某些地區，您的 RPS 配額可能會更高。

資源數量和大小的配額

資源配額是 Amazon Cognito 中資源、輸入欄位、持續時間和其他雜項功能的最大數量或大小。

您可以在 Service Quotas 主控台或從增加服務限制表單請求調整某些資源配額。若要從 Service Quotas 主控台要求配額，請參閱《Service Quotas 使用者指南》中的要求增加配額。如果 Service Quotas 中沒有提供配額，請使用增加服務配額表單。

> **Note**
>
> AWS 帳戶 層級的資源配額 (例如每個 AWS 區域區域的使用者集區) 適用於每個區域中的 Amazon Cognito 資源。例如，您可以在美國東部 (維吉尼亞北部) 有 1,000 個使用者集區，在歐洲 (斯德哥爾摩) 有另外 1,000 個使用者集區。

下表說明預設資源配額，以及它們是否可調整。

Amazon Cognito 使用者集區資源配額

資源	配額	可調整	最大配額
每個使用者集區的應用程式用戶端數量	1,000	是	10,000
每個區域的使用者集區數量	1,000	是	10,000

資源	配額	可調整	最大配額
每個使用者集區的身分提供者數量	300	是	1,000
每個使用者集區的資源伺服器數量	25	是	300
每個使用者集區的使用者數量	40,000,000	是	請聯絡您的帳戶團隊。
產生權杖前 Lambda 觸發程序中的合併變更總數[1]	5,000	是	請聯絡您的帳戶團隊。
每個使用者集區的自訂屬性數量	50	否	N/A
每個屬性的字元數	2048 個位元組	否	N/A
自訂屬性名稱的字元數	20	否	N/A
密碼政策規定的密碼字元數下限	6–99	否	N/A
每日傳送的電子郵件訊息 AWS 帳戶[2]	50	否	N/A
電子郵件主旨中的字元數	140	否	N/A
電子郵件訊息中的字元數	20,000	否	N/A
簡訊驗證訊息中的字元數	140	否	N/A
密碼中的字元數	256	否	N/A

資源	配額	可調整	最大配額
身分提供者名稱中的字元數	32	否	N/A
每個身分提供者的識別符數量	50	否	N/A
連結至使用者的身分數量	5	否	N/A
每個應用程式用戶端的回呼 URL 數量	100	否	N/A
每個應用程式用戶端的登出 URL 數量	100	否	N/A
每個資源伺服器的範圍數量	100	否	N/A
每個應用程式用戶端的範圍數量	50	否	N/A
每個帳戶的自訂網域	4	否	N/A
每個使用者可隸屬的群組數量	100	否	N/A
每個使用者集區的群組數量	10,000	否	N/A

[1] 來自產生權杖前 Lambda 觸發程序的權杖中可能會遇到此配額。現有和新增的宣告數量以及存取和身分權杖中的範圍總合必須小於或等於此配額。抑制的宣告和範圍不計入此配額。

[2] 此配額僅適用於為 Amazon Cognito 使用者集區使用預設的電子郵件功能。若要獲得更高的電子郵件傳送量，請將您的使用者集區設定成使用您的 Amazon SES 電子郵件組態。如需詳細資訊，請參閱 Amazon Cognito 使用者集區的電子郵件設定。

Amazon Cognito 使用者集區工作階段有效性參數

權杖	配額
ID 權杖	5 分鐘 – 1 天
重新整理權杖	1 小時 – 3,650 天
存取權杖	5 分鐘 – 1 天
託管 UI 工作階段 Cookie	1 小時
身分驗證工作階段字符	3 分鐘至 15 分鐘

Amazon Cognito 使用者集區驗證碼安全性資源配額 (不可調整)

資源	配額
註冊確認碼有效期間	24 小時
使用者屬性驗證碼有效期間	24 小時
多重要素驗證 (MFA) 驗證碼有效期間	3 至 15 分鐘
忘記密碼確認碼有效期間	1 小時
每位使用者每小時的最大 ConfirmForgotPassword 與 ForgotPassword 請求數 [1]	5–20
每位使用者每小時的最大 ResendConfirmationCode 請求數	5
每位使用者每小時的最大 ConfirmSignUp 請求數	15
每位使用者每小時的最大 ChangePassword 請求數	5
每位使用者每小時的最大 GetUserAttributeVerificationCode 請求數	5

資源	配額
每位使用者每小時的最大 `VerifyUserAttribute` 請求數	15

[1] Amazon Cognito 會評估請求中更新密碼的風險因素，並指派與評估風險等級相關的配額。如需詳細資訊，請參閱 忘記密碼行為。

Amazon Cognito 使用者集區使用者匯入任務資源配額

資源	配額	可調整	最大配額
每個使用者集區的使用者匯入任務數量	1,000	是	請聯絡您的帳戶團隊。
每個使用者匯入 CSV 列的字元數上限	16,000	否	N/A
CSV 檔案大小上限	100 MB	否	N/A
每個 CSV 檔案的使用者數目上限	500,000	否	N/A

Amazon Cognito 身分集區 (聯合身分) 資源配額

資源	配額	可調整	最大配額
每個帳戶的身分集區數量	1,000	是	N/A
每個身分集區的 Amazon Cognito 使用者集區供應商數量	50	是	1000
身分集區名稱的字元長度	128 位元組	否	N/A

資源	配額	可調整	最大配額
登入供應商名稱的字元長度	2048 個位元組	否	N/A
每個身分集區的身分數量	無限制	否	N/A
可為其指定角色映射的身分提供者數量	10	否	N/A
單一列示或查閱呼叫的結果數量	60	否	N/A
角色型存取控制 (RBAC) 規則數量	25	否	N/A

Amazon Cognito Sync 資源配額

資源	配額	可調整	最大配額
每個身分的資料集數量	20	是	請聯絡您的帳戶團隊。
每個資料集的記錄數量	1,024	是	請聯絡您的帳戶團隊。
單一資料集的大小	1 MB	是	請聯絡您的帳戶團隊。
資料集名稱的字元數	128 位元組	否	N/A
請求成功之後等待大量發佈的時間	24 小時	否	N/A

Amazon Cognito API 和端點參考

下列參考說明 Amazon Cognito 每個功能的服務端點。Amazon Cognito 使用者集區具有下列選項：具有使用者集區網域的使用者集區端點和使用者集區 API。如需 Amazon Cognito 使用者集區的使用者集區 API 的 API 操作類別明細，請參閱 使用 Amazon Cognito 使用者集區 API 和使用者集區端點。

如需 AWS 區域 使用者集區 API 服務端點的完整清單，請參閱《AWS 一般參考》中的服務端點。

主題

- 使用者集區聯合端點和託管 UI 參考
- Amazon Cognito 使用者集區 API 參考
- Amazon Cognito 身分集區 (聯合身分) API 參考
- Amazon Cognito Sync API 參考

使用者集區聯合端點和託管 UI 參考

當您將網域指派給使用者集區，Amazon Cognito 會啟用此處列出的公有網頁。您的網域是所有應用程式用戶端的集中存取點。其中包括託管 UI，您的使用者可以在該處註冊和登入 (登入端點)，然後登出 (登出端點)。如需這些資源的詳細資訊，請參閱 設定和使用 Amazon Cognito 託管 UI 和聯合端點。

這些頁面還包括允許您的用戶池與第三方 SAML，OpenID Connect（OIDC）和 OAuth 2.0 身份提供者（ ）進行通信的公共 Web 資源。IdPs若要透過聯合身分提供者登入使用者，您的使用者必須啟動對互動託管 UI 登入端點 或 OIDC 授權端點 的請求。授權端點會將您的使用者重新導向至您的託管 UI 或 IdP 登入頁面。

您的應用程式也可以透過 Amazon Cognito 使用者集區 API 登入本機使用者。本機使用者僅存在於您的使用者集區目錄中，不會透過外部 IdP 進行聯合。

除了託管的 UI 和聯合端點之外，Amazon Cognito 還與安卓系統 JavaScript、iOS 等開發套件整合。開發套件提供工具可使用使用者集區 API 端點和 Amazon Cognito API 服務端點執行使用者集區 API 操作。如需服務端點的詳細資訊，請參閱 Amazon Cognito Identity endpoints and quotas (Amazon Cognito Identity 端點與配額)。

> **Warning**
>
> 請勿針對 Amazon Cognito 網域釘選最終實體或中繼傳輸層安全性 (TLS) 憑證。 AWS 管理所有使用者集區端點和前置碼網域的所有憑證。支援 Amazon Cognito 憑證的信任鏈中的憑證授

權單位 (CA) 會動態輪換和更新。當您將應用程式釘選到中繼或分葉憑證時，您的應用程式在 AWS 輪換憑證時可能會失敗，恕不另行通知。
請改將應用程式綁定至所有可用的 Amazon 根憑證。如需詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中憑證綁定的最佳實務和建議。

主題

託管 UI 端點參考

當您將網域新增至使用者集區，Amazon Cognito 會啟用本節中的託管 UI 端點。它們是您的使用者可以完成使用者集區核心驗證作業的網頁。包括密碼管理、多重要素驗證 (MFA) 以及屬性驗證的頁面。如需託管 UI 中使用者體驗的詳細資訊，請參閱 使用託管 UI 註冊和登入。

組成託管 UI 的網頁是前端 Web 應用程式，用於與客戶進行互動式使用者工作階段。您的應用程式必須在使用者的瀏覽器中調用託管 UI。Amazon Cognito 不支援以程式設計方式存取本章中的網頁。OAuth 2.0、OpenID Connect 和 SAML 2.0 聯合端點參考 中傳回 JSON 回應的聯合端點可以直接在應用程式程式碼中查詢。授權端點 會重新導向至託管 UI 或 IdP 登入頁面，也必須在使用者的瀏覽器中開啟。

本指南中的主題詳細說明常用的託管 UI 端點。在您將網域指派給使用者集區時，Amazon Cognito 還會提供後續網頁。

託管 UI 端點

端點 URL	描述	如何存取
https://######### /login	登入使用者集區本機和聯合使用者。	從 授權端點、`/logout` 和 `/confirmforgotPassword` 這類端點重新導向。請參閱登入端點。

端點 URL	描述	如何存取
https://######### /logout	登出使用者集區使用者。	直接連結。請參閱登出端點。
https://######### /confirmUser	確認已選取電子郵件連結的使用者，以驗證其使用者帳戶。	使用者在電子郵件訊息中選取連結。
https://######### /signup	註冊一個新使用者。當您的使用者選擇 Sign up (註冊)，/login 頁面會將他們導向到 /signup。	使用與 /oauth2/authorize 相同的參數直接連結。
https://######### /confirm	在您的使用者集區傳送向註冊的使用者傳送確認代碼之後，會提示您的使用者輸入代碼。	僅從 /signup 重新導向。
https://######### /forgotPassword	提示您的使用者輸入其使用者名稱和傳送密碼重設代碼。使用者選取 Forgot your password? (忘記密碼？) 時，/login 頁面會將您的使用者導向 /forgotPassword 。	1. /login 的 忘記密碼 連結。 2. 使用與 /oauth2/authorize 相同的參數直接連結。
https://######### /confirmforgotPassword	提示您的使用者輸入其密碼重設代碼和新密碼。使用者選取 Reset your password (重設您的密碼) 時，/forgotPassword 頁面會將您的使用者導向 /confirmforgotPassword 。	僅從 /forgotPassword 重新導向。
https://######### /resendcode	向已經註冊使用者集區的使用者傳送新的確認碼。	僅從 /confirm 傳送新的代碼連結。

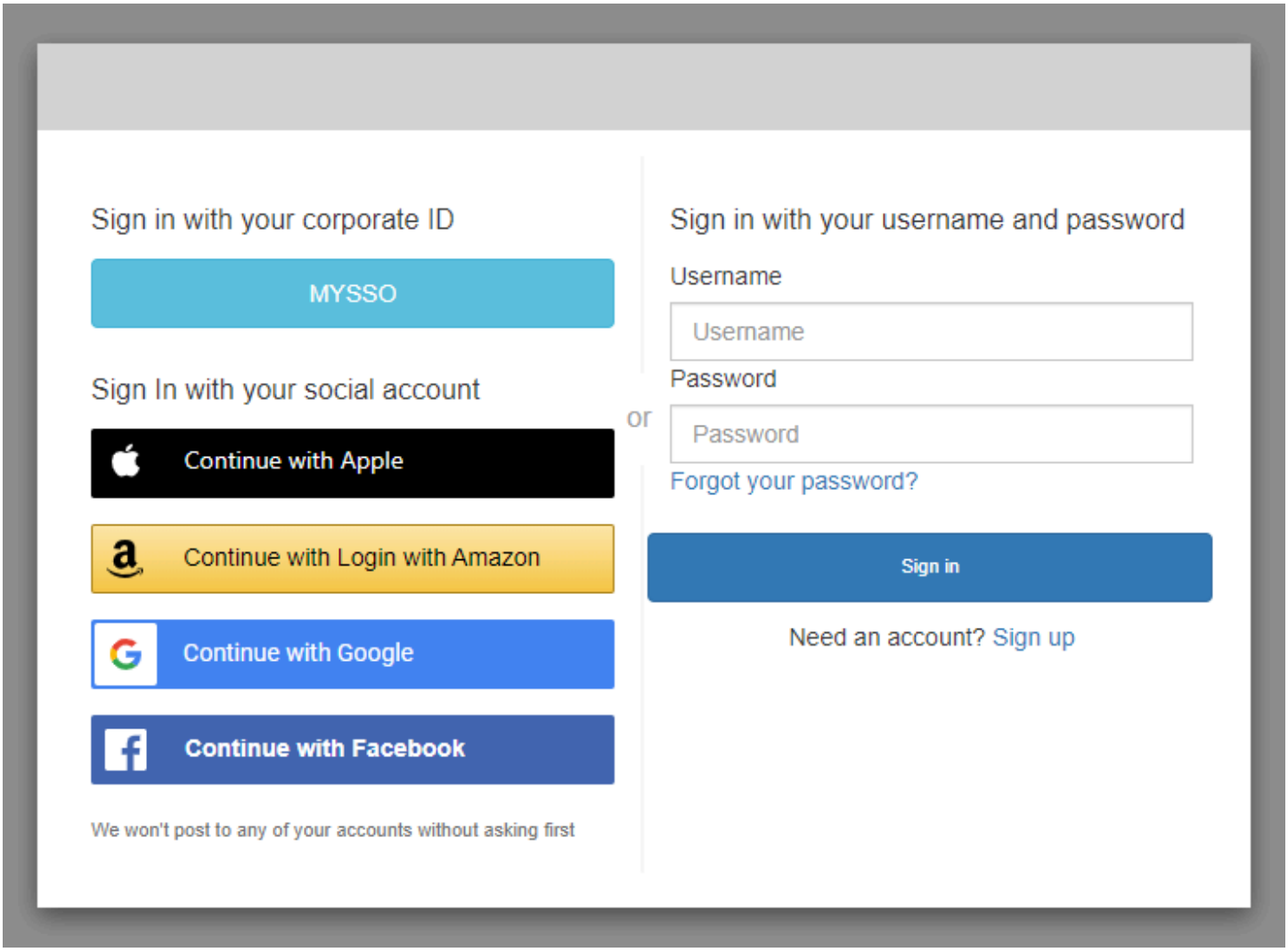
主題

- 登入端點
- 登出端點

登入端點

登入端點是來自 授權端點 的身分驗證伺服器和重新導向目的地。當您未指定身分提供者時，這是託管 UI 的進入點。當您產生登入端點的重新導向時，它會載入登入頁面，並提供使用者為用戶端設定的身分驗證選項。

> **Note**
>
> 登入端點是託管 UI 的元件。在您的應用程式中，調用重新導向至登入端點的聯合和託管 UI 頁面。由使用者直接存取登入端點並不是最佳實務。



GET /login

/login 端點僅支援 HTTPS GET 使用者的初始請求。您的應用程式會在 Chrome 或 Firefox 等瀏覽器中調用頁面。當您/login從重新導向至時授權端點，它會傳遞您在初始請求中提供的所有參數。登入端點支援授權端點的所有請求參數。您也可以直接存取登入端點。最佳實務是以 /oauth2/authorize 做為您所有使用者工作階段的來源。

範例 — 提示使用者登入

此範例顯示登入畫面。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/login?
                response_type=code&
                client_id=ad398u21ijw3s9w3939&
                redirect_uri=https://YOUR_APP/redirect_uri&
                state=STATE&
                scope=openid+profile+aws.cognito.signin.user.admin
```

示例-響應

身分驗證伺服器會重新引導至應用程式，並附上授權碼和狀態。伺服器必須以查詢字串參數傳回授權碼和狀態，而不是以片段傳回。

```
HTTP/1.1 302 Found
                    Location: https://YOUR_APP/redirect_uri?
code=AUTHORIZATION_CODE&state=STATE
```

使用者啟動的登入請求

使用者載入 /login 端點後，就可以輸入使用者名稱和密碼，然後選擇登入。當使用者這樣做時，便會產生 HTTPS POST 請求，此請求的標頭請求參數與 GET 請求相同，且請求內文包含其使用者名稱、密碼及裝置指紋。

登出端點

/logout 端點是重新導向端點。它註銷用戶並重定向到您的應用程序客戶端的授權註銷 URL 或/login端點。GET 請求中對 /logout 端點的可用參數皆是針對 Amazon Cognito 託管 UI 使用案例量身打造。

若要將使用者重新導向至託管 UI 以再次登入，請在請求中新增 `redirect_uri` 參數。使用 `redirect_uri` 參數的 `logout` 請求，必須同時包括後續請求 登入端點 的參數，如 `client_id`、`response_type` 和 `scope`。

登出端點是前端 Web 應用程式，可與客戶進行互動式使用者工作階段。您的應用程式必須在使用者瀏覽器中調用此端點和其他託管的 UI 端點。

若要將使用者重新導向至您選擇的頁面，請將 允許的登出 URL 新增至您的應用程式用戶端。在使用者對 `logout` 端點的請求中，新增 `logout_uri` 和 `client_id` 參數。如果 `logout_uri` 的值是應用程式用戶端的 允許的登出 URL 之一，Amazon Cognito 會將使用者重新導向至該 URL。

使用 SAML 2.0 的單一登出 (SLO) IdPs，Amazon Cognito 會先將您的使用者重新導向至您在 IdP 組態中定義的 SLO 端點。在您的 IdP 將使用者重新導向至之後`saml2/logout`，Amazon Cognito 會再`logout_uri`從您的請求重新導向至`redirect_uri`或進行回應。如需詳細資訊，請參閱 SAML 登出流程。

登出端點不會將使用者登出 OIDC 或社交身分提供者 ()。IdPs若要使用外部 IdP 將使用者從其工作階段登出，請將他們導向至該提供者的登出頁面。

GET /logout

`/logout` 端點僅支援 HTTPS `GET`。使用者集區用戶端通常會透過系統瀏覽器提出這項請求。瀏覽器通常是 Android 中的 Custom Chrome Tab 或 iOS 中的 Safari View Control。

請求參數

client_id

應用程式的應用程式用戶端 ID。若要取得應用程式用戶端 ID，您必須在使用者集區中註冊應用程式。如需詳細資訊，請參閱 使用者集區應用程式用戶端。

必要。

logout_uri

透過 logout_uri 參數將您的使用者重新導向至自訂登出頁面。將其值設定為應用程式用戶端 sign-out URL (登出 URL)，這是您希望使用者登出後將其重新導向至的位置。僅將 logout_uri 搭配 client_id 參數使用。如需詳細資訊，請參閱 使用者集區應用程式用戶端。

您也可以使用 logout_uri 參數將您的使用者重新導向到另一個應用程式用戶端的登入頁面。將其他應用程式用戶端的登入頁面設定為您應用程式用戶端中的 Allowed callback URL(允許的回呼 URL)。在您對 `/logout` 端點的請求中，將 logout_uri 參數值新增至 URL 編碼的登入頁面。

Amazon Cognito 要求在您對 `/logout` 端點的請求中需有 logout_uri 或 redirect_uri 參數。logout_uri 參數會將您的使用者重新導向至另一個網站。如果您向 `/logout` 端點發出的請求中包含 logout_uri 和 redirect_uri 參數，Amazon Cognito 只會利用 logout_uri 參數，並覆蓋 redirect_uri 參數。

redirect_uri

使用 redirect_uri 參數將使用者重新導向至登入頁面，以進行身分驗證。將其值設定為您想要使用者再次登入後重新導向至的應用程式用戶端 Allowed callback URL (允許的回呼 URL)。新增您想要傳遞給 `/login` 端點的 client_id、scope、state 和 response_type 參數。

Amazon Cognito 要求在您對 `/logout` 端點的請求中需有 logout_uri 或 redirect_uri 參數。要將用戶重定向到`/login`端點以重新進行身份驗證並將令牌傳遞給您的應用程序，請添加 redirec t_uri 參數。如果您向*`/logout`*端點發出的請求中包含 logout_uri 和重定向參數，Amazon Cognito 會覆寫重定向參數，並專門處理 logout_uri 參數。

response_type

您希望使用者在登入後從 Amazon Cognito 收到的 OAuth 2.0 回應。`code` 和 `token` 是 response_type 參數的有效值。

如果您使用 redirect_uri 參數，則為必要。

state

當您的應用程式將狀態參數新增至請求時，當`/oauth2/logout`端點重新導向您的使用者時，Amazon Cognito 會將其值傳回給您的應用程式。

將此值新增至您的請求，以防禦 CSRF 攻擊。

您無法將 `state` 參數的值設定為 URL 編碼的 JSON 字串。要在`state`參數中傳遞與此格式匹配的字符串，請將字符串編碼為 base64，然後在應用程序中對其進行解碼。

強烈建議您使用 redirect_uri 參數。

scope

您將使用者登出後，想要使用 redirect_uri 參數向 Amazon Cognito 請求的 OAuth 2.0 範圍。Amazon Cognito 會將您的使用者重新導向至 `/login` 端點，並附加您對 `/logout` 端點請求中的 scope 參數。

如果您使用 redirect_uri 參數，則為可選。如果您未包含 scope 參數，則 Amazon Cognito 會將您的使用者重新導向至 `/login` 端點，並附加 scope 參數。當 Amazon Cognito 重新導向您的使用者並自動填入 `scope` 時，該參數會包含應用程式用戶端的所有已授權範圍。

請求示例

示例-註銷並將用戶重定向到客戶端

除了logout_uri和之外client_id，此端點的所有可能查詢參數都會傳遞至授權端點. 當請求包含 logout_uri 和 client_id 時，Amazon Cognito 會將使用者工作階段重新導向至 logout_uri 值的 URL，並忽略其他所有請求參數。此 URL 必須是應用程式用戶端的授權登出 URL。

以下是登出並重新導向至 https://www.example.com/welcome 的範例請求。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?
  client_id=1example23456789&
  logout_uri=https%3A%2F%2Fwww.example.com%2Fwelcome
```

範例 — 登出並提示使用者以其他使用者身分登入

當請求忽略 logout_uri，但以其他方式向授權端點提供格式正確請求的參數時，Amazon Cognito 會將使用者重新導向至託管的 UI 登入。登出端點會將原始要求中的參數附加至重新導向之目的地。在給登出端點的要求中，redirect_uri 參數並非登出 URL，而是您想要傳遞給授權端點的登入 URL。

以下是將使用者登出、重新導向至登入頁面，以及在登入https://www.example.com後提供授權碼的範例要求。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?
  response_type=code&
  client_id=1example23456789&
  redirect_uri=https%3A%2F%2Fwww.example.com&
  state=example-state-value&
  nonce=example-nonce-value&
  scope=openid+profile+aws.cognito.signin.user.admin
```

OAuth 2.0、OpenID Connect 和 SAML 2.0 聯合端點參考

當您將網域新增至使用者集區，Amazon Cognito 會啟用本節中的端點。聯合端點不是使用者互動式。他們會為您的應用程式執行服務角色，以便與第三方 OAuth 2.0、OIDC 和 SAML 2.0 身分識別提供者 () 進行通訊。IdPs

本指南中的主題說明幾個常用的 OAuth 2.0 和 OIDC 端點。Amazon Cognito 還會在您將網域指派給使用者集區時建立以下端點。

使用者集區聯合端點

端點 URL	描述	如何存取
https://######### /oauth2/authorize	將使用者重新導向至託管 UI 或使用其 IdP 登入。	在客戶瀏覽器中調用以開始用戶身份驗證。請參閱授權端點。
https://######### /oauth2/token	根據授權碼或用戶憑證請求傳回權杖。	應用程序請求檢索令牌。請參閱權杖端點。
https://######### /oauth2/userInfo	根據存取權杖中的 OAuth 2.0 範圍和使用者身分傳回使用者屬性。	應用程序請求檢索用戶配置文件。請參閱UserInfo 端點。
https://######### /oauth2/revoke	撤銷重新整理權杖和關聯的存取權杖。	應用程序要求撤銷令牌。請參閱撤銷端點。
https://cognito-idp.##.amazonaws.com/#######*ID*/.well-known/openid-configuration	使用者集區的 OIDC 架構目錄。	應用程序請求找到用戶池發行者元數據。
https://cognito-idp.##.amazonaws.com/#######*ID*/.well-known/jwks.json	可用於驗證 Amazon Cognito 權杖的公有金鑰。	應用程序請求驗證 JWT。
https://*Your user pool domain*/oauth2/idpresponse	社交身分提供者必須透過授權碼將您的使用者重新導向至此端點。Amazon Cognito 在對您的聯合身分使用者進行身分驗證時，將代碼兌換為權杖。	從 OIDC IdP 登入重新導向為 IdP 用戶端回呼 URL。
https://######### /saml2/idpresponse	用於與 SAML 2.0 身分識別提供者整合的宣告取用者回應 (ACS) URL。	從 SAML 2.0 IdP 重新導向為 ACS 網址，或 IdP 起始登入的起始點重新導向。[1]

端點 URL	描述	如何存取
https://#########/## 2 /登出	用於與 SAML 2.0 身分識別提供者整合的單一登出 (SLO) URL。	從 SAML 2.0 IdP 重新導向為單一登出 (SLO) 網址。僅接受 POST 綁定。

[1] 如需有關 IDP 起始的 SAML 登入的詳細資訊，請參閱。使用 IDP 起始的 SAML 登入

如需 OpenID Connect 和 OAuth 標準的詳細資訊，請參閱 OpenID Connect 1.0 和 OAuth 2.0。

主題

- 授權端點
- 權杖端點
- UserInfo 端點
- 撤銷端點
- 樣本 2/ID自定端點

授權端點

`/oauth2/authorize` 端點是支援兩個重新導向目的地的重新導向端點。如果您在 URL 中包含 `identity_provider` 或 `idp_identifier` 參數，則會以無提示的方式將您的使用者重新導向到該身分提供者 (IdP) 的登入頁面。否則，它會使用您在請求中包含的相同 URL 參數重新導向至登入端點。

授權端點重新導向至託管 UI 或 IdP 登入頁面。此端點上的使用者工作階段目的地是使用者必須直接在瀏覽器中與之互動的網頁。

若要使用授權端點，請在 `/oauth2/authorize` 叫用使用者的瀏覽器，並搭配為使用者集區提供有關下列使用者集區詳細資訊的參數。

- 您要登入的應用程式用戶端。
- 您最後想要使用的回呼 URL。
- 您將要在使用者存取權杖中請求的 OAuth 2.0 範圍。
- 或者，您要用來登入的第三方 IdP。

您也可以提供 `state` 和 `nonce` 參數，讓 Amazon Cognito 用來驗證傳入宣告。

GET **/oauth2/authorize**

/oauth2/authorize 端點僅支援 HTTPS GET。您的應用程式通常會在使用者的瀏覽器中啟動此請求。您只能透過 HTTPS 向 /oauth2/authorize 端點提出請求。

您可以在 Authorization Endpoint (授權端點) 進一步了解 OpenID Connect (OIDC) 標準中的授權端點定義。

請求參數

response_type

(必要) 回應類型。必須是 code 或 token。

response_type 為 code 的成功請求會傳回授權碼授與。授權碼授與是 Amazon Cognito 附加到您的重新導向 URL 的 code 參數。您的應用程式可以針對存取、ID 及重新整理權杖和 權杖端點 交換代碼。作為安全最佳實務，並為您的使用者接收重新整理權杖，請在您的應用程式中使用授權碼授與。

具有 token 之 response_type 的成功請求會傳回隱含授與。隱含授與是 Amazon Cognito 附加到您的重新導向 URL 的 ID 和存取權杖。隱含授與不太安全，因為它會向使用者公開權杖和潛在的識別資訊。您可以在應用程式用戶端的組態中停用對隱含授與的支援。

client_id

(必要) 應用程式用戶端 ID。

client_id 的值必須是您提出請求之使用者集區中應用程式用戶端的 ID。您的應用程式用戶端必須支援 Amazon Cognito 本機使用者或至少一個第三方 IdP 登入。

redirect_uri

(必要) Amazon Cognito 授權使用者後，身份驗證伺服器會重新導向瀏覽器的 URL。

重新導向統一資源識別符 (URI) 必須具有下列屬性：

- 必須是絕對 URI。
- 您必須已預先以用戶端註冊此 URI。
- 不得包含片段元件。

請參閱 OAuth 2.0 - 重新導向端點。

Amazon Cognito 需要您的重新導向 URI 使用 HTTPS，但 `http://localhost` 除外，您可以將其設定為回呼 URL 以供測試之用。

Amazon Cognito 也支援應用程式回呼，例如 `myapp://example`。

`state`

(選用，建議使用) 當您的應用程式將狀態參數新增至請求時，當`/oauth2/authorize`端點重新導向您的使用者時，Amazon Cognito 會將其值傳回給您的應用程式。

將此值新增至您的請求，以防禦 CSRF 攻擊。

您無法將 `state` 參數的值設定為 URL 編碼的 JSON 字串。要在`state`參數中傳遞與此格式匹配的字符串，請將字符串編碼為 base64，然後在您的應用程序中對其進行解碼。

`identity_provider`

（可選）添加此參數以繞過託管 UI 並將您的用戶重定向到提供程序登錄頁面。identity_provider 參數的值是身分提供者 (IdP) 在使用者集區中顯示的名稱。

- 對於社交提供者，您可以使用識別提供者值`Facebook`、`Google`、`LoginWithAmazon`和。`SignInWithApple`
- 對於 Amazon Cognito 使用者集區，請使用值`COGNITO`。
- 對於 SAML 2.0 和 OpenID Connect (OIDC) 身分識別提供者 (IdPs)，請使用您在使用者集區中指派給 IdP 的名稱。

`idp_identifier`

(選擇性) 新增此參數以重新導向至具有 identity _provider 名稱替代名稱的提供者。您可以從 Amazon Cognito 主控台的 [登入體驗] 索引標籤 IdPs 中輸入 SAML 2.0 和 OIDC 的識別碼。

`scope`

(選擇性) 可以是任何系統保留範圍或與用戶端相關聯的自訂範圍的組合。範圍必須以空格隔開。系統預留範圍為 `openid`、`email`、`phone`、`profile` 和 `aws.cognito.signin.user.admin`。所使用的任何範圍都必須與用戶端建立關聯，否則在執行時間會將其忽略。

如果用戶端沒有請求任何範圍，則身分驗證伺服器會使用與用戶端相關聯的所有範圍。

只有在請求 `openid` 範圍時，才會傳回 ID 權杖。只有在請求 `aws.cognito.signin.user.admin` 範圍時，才能對 Amazon Cognito 使用者集區使用存取權杖。只有在也同時請求 `phone` 範圍時，才能請求 `email`、`profile` 和 `openid` 範圍。這些範圍需要有進入 ID 權杖中的宣告。

code_challenge_method

(選擇性) 您用來產生挑戰的雜湊通訊協定。PKCE RFC 定義兩種方法：S256 和一般，但是 Amazon Cognito 身分驗證伺服器只支援 S256。

code_challenge

(選擇性) 您從code_verifier.

僅當您指定 code_challenge_method 參數時才需要。

nonce

(選擇性) 您可以新增至要求的隨機值。您提供的 nonce 值包含在 Amazon Cognito 發出的 ID 權杖中。為了防止重播攻擊，您的應用程式可以檢查 ID 權杖中的 nonce 宣告並將其與您產生的權杖進行比較。如需 nonce 宣告的詳細資訊，請參閱 OpenID Connect standard (OpenID Connect 標準) 中的 ID token validation (ID 權杖驗證)。

具有正面回應的範例要求

下列範例說明/oauth2/authorize端點的 HTTP 要求格式。

授予授權碼

這是授權碼授予的示例請求。

示例-GET 請求

以下請求啟動會話以檢索用戶傳遞給您的應用程序的redirect_uri授權碼。此工作階段請求範圍涵蓋使用者屬性和 Amazon Cognito 自助服務 API 操作的存取權限。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=openid+profile+aws.cognito.signin.user.admin
```

示例-響應

Amazon Cognito 身分驗證伺服器會附上授權碼和狀態，重新引導回應用程式。授權碼有效期為五分鐘。

```
HTTP/1.1 302 Found
Location: https://www.example.com?code=a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111&state=abcdefg
```

使用 PKCE 授予授權碼

這是向 PK CE 授予授權碼的示例請求。

示例-GET 請求

下列要求會將code_challenge參數新增至先前的要求。要完成令牌的代碼交換，您必須在向/oauth2/token端點的請求中包含該code_verifier參數。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin&
code_challenge_method=S256&
code_challenge=a1b2c3d4...
```

示例-響應

身份驗證服務器使用授權代碼和狀態重定向回您的應用程序。代碼和狀態必須在查詢字符串參數中返回，而不是在片段中：

```
HTTP/1.1 302 Found
Location: https://www.example.com?code=a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111&state=abcdefg
```

授予權杖，不含 **openid** 範圍

這是生成一個隱式授予，並直接返回到用戶的會話 JWT 的示例請求。

示例-GET 請求

以下請求是來自您的授權服務器的隱式授權。來自 Amazon Cognito 的存取權杖可授權自助式 API 作業。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
```

```
response_type=token&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin
```

示例-響應

Amazon Cognito 授權伺服器會附上存取權杖，重新引導回應用程式。由於未請求 openid 範圍，Amazon Cognito 不會傳回 ID 權杖。此外，Amazon Cognito 不會在此流程中傳回重新整理權杖。Amazon Cognito 會傳回片段中的存取權杖和狀態，而不是在查詢字串中：

```
HTTP/1.1 302 Found
Location: https://YOUR_APP/
redirect_uri#access_token=ACCESS_TOKEN&token_type=bearer&expires_in=3600&state=STATE
```

授予權杖，含 **openid** 範圍

這是生成一個隱式授予，並直接返回到用戶的會話 JWT 的示例請求。

示例-GET 請求

以下請求是來自您的授權服務器的隱式授權。來自 Amazon Cognito 的存取權杖可授權存取使用者屬性和自助式 API 操作。

```
GET
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=token&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin+openid+profile
```

示例-響應

授權服務器使用訪問令牌和 ID 令牌重定向回您的應用程序（因為openid範圍包含在內）：

```
HTTP/1.1 302 Found
Location: https://
www.example.com#id_token=eyJra67890EXAMPLE&access_token=eyJra12345EXAMPLE&token_type=bearer&exp
```

負向回應的範例

Amazon Cognito 可能會拒絕您的請求。負面要求隨附 HTTP 錯誤碼和說明，您可以用來更正要求參數。以下是負面回應的範例。

- 如果client_id和redirect_uri有效，但請求參數格式不正確，則驗證服務器將錯誤重定向到客戶端，redirect_uri並在 URL 參數中附加錯誤消息。以下是格式錯誤的範例。
  - 請求不包含response_type參數。
  - 授權請求提供了一個code_challenge參數，但不是一個code_challenge_method參數。
  - code_challenge_method參數的值不是S256。

  以下是對具有不正確格式的示例請求的響應。

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_request
  ```

- 如果用戶端請求code或進token入response_type，但沒有這些請求的權限，Amazon Cognito 授權伺服器會返回unauthorized_client到用戶端redirect_uri，如下所示：

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=unauthorized_client
  ```

- 如果用戶端請求的範圍不明、範圍格式不正確或無效，Amazon Cognito 授權伺服器會將 invalid_scope 傳回用戶端的 redirect_uri，如下所示：

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_scope
  ```

- 如果服務器中有任何未預期的錯誤，則驗證服務器返回server_error到客戶端redirect_uri。因為 HTTP 500 錯誤不會傳送至用戶端，因此錯誤不會顯示在使用者的瀏覽器中。授權服務器返回以下錯誤。

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=server_error
  ```

- 當 Amazon Cognito 透過與第三方聯合進行身份驗證時 IdPs，Amazon Cognito 可能會遇到連線問題，例如：
  - 如果在向 IdP 請求權杖時，發生連線逾時，身分驗證伺服器則會如下所示，將錯誤重新導向至用戶端的 redirect_uri：

    ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Timeout+occurred+in+calling+IdP+token
    +endpoint
    ```

  - 如果在調用 ID 令牌驗證的`jwks_uri`端點時發生連接超時，則身份驗證服務器重定向並出現錯誤，`redirect_uri`如下所示：

    ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=error_description=Timeout+in+calling+jwks
    +uri
    ```

- 透過聯合至第三方進行驗證時 IdPs，提供者可能會傳回錯誤回應。這可能是由於配置錯誤或其他原因造成的，例如：
  - 如果從其他供應商收到錯誤回應，身分驗證伺服器則會如下所示，將錯誤重新導向至用戶端的`redirect_uri`：

    ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=[IdP name]+Error+-+[status code]+error
     getting token
    ```

  - 如果從 Google 收到錯誤回應，身分驗證伺服器則會如下所示，將錯誤重新導向至用戶端的`redirect_uri`：

    ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Google+Error+-+[status code]+[Google-
    provided error code]
    ```

- 當 Amazon Cognito 在連線至外部 IdP 時遇到通訊例外狀況時，身分驗證伺服器會以下列其中一個訊息重新導向，並顯示錯誤訊息至用戶端：`redirect_uri`
  - ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Connection+reset
    ```
  - ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Read+timed+out
    ```

權杖端點

`/oauth2/token` 處的 OAuth 2.0 權杖端點發出 JSON Web 權杖 (JWT)。

您的用戶池 OAuth 2.0 授權服務器從令牌端點發布 JSON Web 令牌（JWT）到以下類型的會話：

1. 已完成授權碼授權要求的使用者。成功兌換程式碼會傳回 ID、存取和重新整理權杖。

2. 已完成用戶端認證授與的 M achine-to-machine (M2M) 工作階段。使用用戶端秘密成功授權傳回存取權杖。
3. 先前已登入並接收重新整理權杖的使用者。刷新令牌身份驗證返回新的 ID 和訪問令牌。

> **Note**
>
> 使用授權代碼登錄的用戶在託管 UI 中或通過聯合授予的用戶始終可以從令牌端點刷新其令牌。使用 API 操作登錄`InitiateAuth`並在用戶池中記住的設備未處於活動狀態時，`AdminInitiateAuth`可以使用令牌端點刷新其令牌的用戶。如果記住`AuthFlow`的裝置處於作用中狀態，請使用 `REFRESH_TOKEN_AUTH` in `InitiateAuth` 或 `AdminInitiateAuth` API 要求重新整理權杖。

當您將網域新增至使用者集區時，權杖端點會變為可公開使用。它會接受 HTTP POST 請求。為了確保應用程式安全性，請將 PKCE 與您的授權碼登入事件搭配使用。PKCE 會驗證傳遞授權碼的使用者是經過驗證的相同使用者。如需有關 PKCE 的詳細資訊，請參閱 IETF RFC 7636。

您可以在 使用者集區應用程式用戶端 了解有關使用者集區應用程式用戶端及其授權類型、用戶端密碼、授權範圍和用戶端 ID 的更多資訊。您可以在使用資源伺服器進行範圍、M2M 和 API 授權以下位置了解有關 M2M 授權，客戶端憑據授予以及訪問令牌範圍授權的更多信息。

要從其訪問令牌中檢索有關用戶的信息，請將其傳遞給您的UserInfo 端點或 GetUserAPI 請求。

POST /oauth2/token

`/oauth2/token` 端點僅支援 `HTTPS POST`。您的應用程式會直接對此端點提出請求，而不經由使用者的瀏覽器。

權杖端點支援 `client_secret_basic` 和 `client_secret_post` 身分驗證。如需 OpenID Connect 規格的詳細資訊，請參閱用戶端驗證。如需 OpenID Connect 規格中權杖端點的詳細資訊，請參閱權杖端點。

標頭中的請求參數

`Authorization`

如果有核發秘密給用戶端，則用戶端可在授權標頭中傳遞其 `client_id` 和 `client_secret` 做為 `client_secret_basic` HTTP 授權。您也可以在請求的內文中加入 `client_id` 和 `client_secret` 做為 `client_secret_post` 授權。

授權標頭字串是基本 Base64Encode(client_id:client_secret)。下列範例是djc98u3jiedmi283eu928具有用戶端密碼之應用程式用戶端的授權標頭abcdef01234567890，使用 Base64 編碼的字串版本：djc98u3jiedmi283eu928:abcdef01234567890

```
Authorization: Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw
```

Content-Type

將此參數的值設為 'application/x-www-form-urlencoded'。

內文中的請求參數

grant_type

(必要) 您要求的 OIDC 授權類型。

必須是 authorization_code、refresh_token 或 client_credentials。在以下情況下，您可以從令牌端點請求自定義範圍的訪問令牌：

- 您在應用程序客戶端配置中啟用了請求的範圍。
- 您已將應用程式用戶端設定為用戶端密碼。
- 您在應用程序客戶端中啟用客戶端憑據授予。

client_id

（可選）用戶池中應用程序客戶端的 ID。指定驗證用戶的相同應用程序客戶端。

如果用戶端為公用且沒有密碼，或client_secret在client_secret_post授權中使用，您必須提供此參數。

client_secret

（可選）驗證用戶的應用程序客戶端的客戶端密鑰。如果您的應用程式用戶端具有用戶端秘密，且您沒有傳送 Authorization 標頭，則需要此項。

scope

（可選）可以是與應用程序客戶端關聯的任何自定義範圍的組合。您請求的任何範圍都必須為應用程序客戶端激活。如果沒有，Amazon Cognito 將忽略它。如果客戶端未請求任何範圍，則身份驗證服務器將分配您在應用程序客戶端配置中授權的所有自定義範圍。

只有在 grant_type 為 client_credentials 時使用。

redirect_uri

（可選）必須與用redirect_uri於進authorization_code入的內容相同/oauth2/authorize。

如果是，則必須提供grant_type此參數authorization_code。

refresh_token

（可選）要為用戶的會話生成新的訪問和 ID 令牌，/oauth2/token請將請求中的refresh_token參數值設置為先前從同一應用程序客戶端發布的刷新令牌。

code

（可選）授權碼授予授權碼。如果您的授權請求包含的，則必須提供此參grant_type數authorization_code。

code_verifier

(選擇性) 您用來透過 PKCE 計算授權碼授與要求code_challenge中的任意值。

具有正面回應的範例要求

將授權碼交換成權杖

示例-發布請求

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token&
                            Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                            grant_type=authorization_code&
                            client_id=1example23456789&
                            code=AUTHORIZATION_CODE&
                            redirect_uri=com.myclientapp://myclient/redirect
```

示例-響應

```
HTTP/1.1 200 OK
```

```
                            Content-Type: application/json

                            {
                             "access_token":"eyJra1example",
                             "id_token":"eyJra2example",
                             "refresh_token":"eyJj3example",
                             "token_type":"Bearer",
                             "expires_in":3600
                            }
```

> **Note**
>
> 只有當 refresh_token 是 grant_type 時，權杖端點才會傳回 authorization_code。

將用戶端憑證交換為存取權杖：授權標頭中的用戶端機密

示例-發布請求

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
                            Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                            grant_type=client_credentials&
                            client_id=1example23456789&

 scope=resourceServerIdentifier1/scope1 resourceServerIdentifier2/scope2
```

示例-響應

```
HTTP/1.1 200 OK
                            Content-Type: application/json

                            {
                            "access_token":"eyJra1example",
                            "token_type":"Bearer",
                            "expires_in":3600
                            }
```

將用戶端憑證交換為存取權杖：請求主體中的用戶端機密

示例-發布請求

```
POST /oauth2/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
X-Amz-Target: AWSCognitoIdentityProviderService.Client credentials request
User-Agent: USER_AGENT
Accept: /
Accept-Encoding: gzip, deflate, br
Content-Length: 177
Referer: http://auth.example.com/oauth2/token
Host: auth.example.com
Connection: keep-alive

grant_type=client_credentials&client_id=1example23456789&scope=my_resource_server_identifier%2F
```

示例-響應

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Tue, 05 Dec 2023 16:11:11 GMT
x-amz-cognito-request-id: 829f4fe2-a1ee-476e-b834-5cd85c03373b

{
"access_token": "eyJra12345EXAMPLE",
"expires_in": 3600,
"token_type": "Bearer"
}
```

將使用 PKCE 授予的授權碼交換成權杖

示例-發布請求

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token
                                Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                                grant_type=authorization_code&
                                client_id=1example23456789&
                                code=AUTHORIZATION_CODE&
                                code_verifier=CODE_VERIFIER&
```

```
                                redirect_uri=com.myclientapp://myclient/redirect
```

示例-響應

```
HTTP/1.1 200 OK
                               Content-Type: application/json

                               {
                               "access_token":"eyJra1example",
                               "id_token":"eyJra2example",
                               "refresh_token":"eyJj3example",
                               "token_type":"Bearer",
                               "expires_in":3600
                               }
```

> **Note**
>
> 只有當 refresh_token 是 grant_type 時，權杖端點才會傳回 authorization_code。

將重新整理權杖交換成權杖

示例-發布請求

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
                              Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                              grant_type=refresh_token&
                              client_id=1example23456789&
                              refresh_token=eyJj3example
```

示例-響應

```
HTTP/1.1 200 OK
                              Content-Type: application/json
```

```
                        {
                        "access_token":"eyJra1example",
                        "id_token":"eyJra2example",
                        "token_type":"Bearer",
                        "expires_in":3600
                              }
```

> **Note**
>
> 只有當 refresh_token 是 grant_type 時，權杖端點才會傳回 authorization_code。

負向回應的範例

示例-錯誤響應

```
HTTP/1.1 400 Bad Request
                        Content-Type: application/json;charset=UTF-8

                        {
                        "error":"invalid_request|invalid_client|invalid_grant|
unauthorized_client|unsupported_grant_type"
                        }
```

invalid_request

請求遺漏必要參數、包含不支援的參數值 (非 unsupported_grant_type)，或是格式不正確。例如，grant_type 是 refresh_token，但沒有包含 refresh_token。

invalid_client

用戶端身分驗證失敗。例如，用戶端在授權標頭中包含 client_id 和 client_secret，但並沒有使用該 client_id 和 client_secret 的用戶端存在。

invalid_grant

重新整理權杖已撤銷。

授權碼已被使用或不存在。

應用程式用戶端不具有對所請求範圍中所有屬性的讀取存取權。例如，您的應用程式請求 email 範圍，且您的應用程式用戶端可以讀取 email 屬性，但不能讀取 email_verified。

unauthorized_client

用戶端無權進行授權碼授予流程或重新整理權杖。

unsupported_grant_type

如果 grant_type 不是 authorization_code、refresh_token 或 client_credentials，就會傳回此值。

UserInfo 端點

userInfo 端點是 OpenID Connect (OIDC) userInfo 端點。服務提供者出示您的 權杖端點 核發的存取權杖時，它會以使用者屬性回應。使用者存取權杖中的範圍定義了 userInfo 端點在其回應中傳回的使用者屬性。openid 範圍必須是存取權杖宣告之一。

Amazon Cognito 核發存取權杖以回應使用者集區 API 請求，例如 InitiateAuth。因為它們不包含任何範圍，因此 userInfo 端點不接受這些存取權杖。反之，您必須從權杖端點出示存取權杖。

您的 OAuth 2.0 的第三方身分提供者 (IdP) 也會託管 userInfo 端點。當您的使用者使用該 IdP 進行驗證時，Amazon Cognito 會以無訊息方式與 IdP 端點交換授權碼。token您的使用者集區會傳遞 IdP 存取權杖，以授權從 IdP 端userInfo點擷取使用者資訊。

GET /oauth2/userInfo

您的應用程式會直接對此端點提出請求，而不經由瀏覽器。

如需詳細資訊，請參閱 OpenID Connect (OIDC) 規格中的 UserInfo 端點。

主題

- 標頭中的請求參數
- 示例-請求
- 示例-積極響應
- 負面回應範例

標頭中的請求參數

Authorization: Bearer *<access_token>*

傳遞授權標頭字段中的訪問令牌。

必要。

示例-請求

```
GET /oauth2/userInfo HTTP/1.1
Content-Type: application/x-amz-json-1.1
Authorization: Bearer eyJra12345EXAMPLE
User-Agent: [User agent]
Accept: */*
Host: auth.example.com
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

示例-積極響應

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: [Integer]
Date: [Timestamp]
x-amz-cognito-request-id: [UUID]
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
Server: Server
Connection: keep-alive
{
    "sub": "[UUID]",
    "email_verified": "true",
    "custom:mycustom1": "CustomValue",
    "phone_number_verified": "true",
    "phone_number": "+12065551212",
    "email": "bob@example.com",
    "username": "bob"
}
```

如需 OIDC 宣告的清單，請參閱標準宣告。目前，Amazon Cognito 會針對 `email_verified` 和 `phone_number_verified` 傳回字串值。

負面回應範例

示例-錯誤的請求

```
HTTP/1.1 400 Bad Request
WWW-Authenticate: error="invalid_request",
error_description="Bad OAuth2 request at UserInfo Endpoint"
```

`invalid_request`

要求缺少必要的參數、包含不受支援的參數值，或是格式不正確。

示例-壞令牌

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: error="invalid_token",
error_description="Access token is expired, disabled, or deleted, or the user has
 globally signed out."
```

`invalid_token`

訪問令牌已過期，撤銷，格式錯誤或無效。

撤銷端點

/oauth2/revoke端點會撤銷 Amazon Cognito 最初使用您提供的重新整理權杖發行的使用者存取權杖。此端點還撤銷來自同一刷新令牌的所有後續訪問和身份令牌。端點撤銷權杖後，您無法使用已撤銷的存取權杖存取 Amazon Cognito 權杖進行身分驗證的 API。

POST /oauth2/revoke

/oauth2/revoke 端點僅支援 HTTPS POST。使用者集區用戶端會直接對此端點提出請求，而不是透過系統瀏覽器。

標頭中的請求參數

`Authorization`

如果您的應用程序客戶端具有客戶端密鑰，則應用程序必須通過 Basic HTTP 授權client_secret在授權標頭中傳遞其client_id密鑰。密碼是基本 Base64Encode(client_id:client_secret)。

Content-Type

必須一律為 'application/x-www-form-urlencoded'。

內文中的請求參數

token

(必要) 用戶端要撤銷的重新整理權杖。請求也會撤銷 Amazon Cognito 從重新整理權杖發出的所有存取權杖。

必要。

client_id

（可選）要撤銷的令牌的應用程序客戶端 ID。

如果用戶端是公開的，且沒有密碼，此為必要值。

撤銷請求範例

範例 1：在沒有用戶端機密的情況下撤銷應用程式用戶端的權杖

```
POST /oauth2/revoke HTTP/1.1
    Host: https://mydomain.auth.us-east-1.amazoncognito.com
    Accept: application/json
    Content-Type: application/x-www-form-urlencoded
    token=2YotnFZFEjr1zCsicMWpAA&
    client_id=djc98u3jiedmi283eu928
```

範例 2：在有用戶端機密的情況下撤銷應用程式用戶端的權杖

```
POST /oauth2/revoke HTTP/1.1
    Host: https://mydomain.auth.us-east-1.amazoncognito.com
    Accept: application/json
    Content-Type: application/x-www-form-urlencoded
    Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
    token=2YotnFZFEjr1zCsicMWpAA
```

撤銷錯誤回應

成功回應內文會空白。錯誤回應是具有 `error` 欄位且某些情形下有 `error_description` 欄位的 JSON 物件。

端點錯誤

- 如果請求中沒有權杖，或應用程式用戶端已停用該功能，您會收到 HTTP 400 和錯誤 `invalid_request`。
- 如果 Amazon Cognito 在撤銷請求中傳送的權杖不是重新整理權杖，您會收到 HTTP 400 和錯誤 `unsupported_token_type`。
- 如果用户端憑證無效，您會收到 HTTP 401 和錯誤 `invalid_client`。
- 如果權杖已被撤銷，或者用户端提交無效的權杖，您會收到 HTTP 200 OK。

樣本 2/ID自定端點

`/saml2/idpresponse`接收到 SAML 判斷提示。在 service-provider-initiated (SP 發起的) 登入中，您的 SAML 2.0 身分識別提供者 (IdP) 會使用其 SAML 回應將您的使用者重新導向至此端點。在 SP 啟動的登入中，您的應用程式不會與此端點互動。將您的 IdP 設定`saml2/idpresponse`為宣告用戶服務 (ACS) URL 的路徑。如需工作階段初始化的詳細資訊，請參閱Amazon Cognito 使用者集區中的 SAML 工作階段啟動。

在 IdP 起始的登入中，您的使用者可以透過您自己的程序使用 IdP 登入，並透過 HTTPS 在要求內文中提交 SAML 宣告。`HTTP POST`POST請求的主體必須是`SAMLResponse`參數和`Relaystate`參數。如需詳細資訊，請參閱 使用 IDP 起始的 SAML 登入。

郵政 `/saml2/idpresponse`

若要在 IDP 起始的登入中使用`/saml2/idpresponse`端點，請產生包含參數的 POST 要求，以便為您的使用者集區提供有關使用者工作階段的資訊。

- 他們想要登入的應用程式用戶端。
- 他們想要結束的回調 URL。
- 他們想要在用戶訪問令牌中請求的 OAuth 2.0 範圍。
- 起始登入要求的 IdP。

IDP 起始的要求主體參數

SAML 回應

來自 IdP 的 Base64 編碼 SAML 判斷提示，與使用者集區中的有效應用程式用戶端和 IdP 組態相關聯。

RelayState

`RelayState`參數包含要求參數，否則您將傳遞給`oauth2/authorize`端點。如需這些參數的詳細資訊，請參閱授權端點。

response_type

該 OAuth 2.0 授予類型。

client_id

應用程式用戶端 ID。

redirect_uri

Amazon Cognito 授權使用者之後，身分驗證伺服器會將瀏覽器重新引導至此 URL。

identity_provider

您要重新導向使用者的身分識別提供者名稱。

idp_identifier

您要重新導向使用者的身分識別提供者的識別碼。

scope

您希望用戶從授權服務器請求的 OAuth 2.0 範圍。

具有正面回應的範例要求

示例-發布請求

以下請求用於從應用程序客戶端`MySAMLIdP`中的 IdP 為用戶`1example23456789`授予授權代碼。用戶使`https://www.example.com`用其授權代碼重定向到，可以將其交換為包含具有 OAuth 2.0 範圍`openid`和`phone`的訪問令牌的令牌。`email`

```
POST /saml2/idpresponse HTTP/1.1
User-Agent: USER_AGENT
```

```
Accept: */*
Host: example.auth.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded

SAMLResponse=[Base64-encoded SAML assertion]&RelayState=identity_provider
%3DMySAMLIdP%26client_id%3D1example23456789%26redirect_uri%3Dhttps%3A%2F
%2Fwww.example.com%26response_type%3Dcode%26scope%3Demail%2Bopenid%2Bphone
```

示例-響應

以下是對先前請求的響應。

```
HTTP/1.1 302 Found
Date: Wed, 06 Dec 2023 00:15:29 GMT
Content-Length: 0
x-amz-cognito-request-id: 8aba6eb5-fb54-4bc6-9368-c3878434f0fb
Location: https://www.example.com?code=[Authorization code]
```

OAuth 2.0 授予

Amazon Cognito 使用者集區 OAuth 2.0 授權伺服器會發出權杖，以回應三種類型的 OAuth 2.0 授權授予。您可以為使用者集區中的每個應用程式用戶端設定支援的授予類型。您無法在相同應用程式用戶端中，將用戶端憑證授予啟用為隱含或授權碼授予。隱含和授權碼授予的請求是從您的 授權端點 開始，而用戶端憑證授予的請求則是從您的 權杖端點 開始。

授予授權碼

若您的身分識別請求成功，授權伺服器會在 code 參數中附加授權碼至您的回呼 URL 做為回應。您必須將 ID、存取及重新整理權杖的授權碼與 權杖端點 進行交換。若要請求授權碼授予，請在您的請求中將 response_type 設定為 code。如需請求範例，請參閱 授予授權碼。

授權碼授予是最安全的授權授予形式。它不會直接向您的使用者顯示權杖內容。而是由您的應用程式負責擷取並安全儲存使用者的權杖。在 Amazon Cognito 中，授權碼授予是從授權伺服器取得全部三種權杖類型 (ID、存取和重新整理) 的唯一方式。您也可以透過 Amazon Cognito 使用者集區 API，從身分驗證取得全部三種權杖類型，但 API 不會發出範圍不是 aws.cognito.signin.user.admin 的存取權杖。

隱含授與

若您的身分驗證請求成功，授權伺服器會在 access_token 參數中附加存取權杖以及在 id_token 參數中附加 ID 權杖至您的回呼 URL 做為回應。隱含授予不需與 權杖端點 進一步互

動。若要請求隱含授予，請在您的請求終將 `response_type` 設定為 `token`。隱含授予只會產生 ID 和存取權杖。如需請求範例，請參閱 授予權杖，不含 openid 範圍。

隱含授予是舊有的授權授予。與授權碼授予不同之處在於，使用者可以攔截並檢查您的權杖。為了防止透過隱含授予傳遞權杖，請將您的應用程式用戶端設定為僅支援授權碼授予。

用戶端憑證

用戶端認證是存取權的僅授權授與。 machine-to-machine若要接收用戶端憑證授予，請略過 授權端點 並直接對 權杖端點 產生請求。您的應用程式用戶端必須有用戶端密碼，且只支援用戶端憑證授予。若您的請求成功，授權伺服器會傳回存取權杖做為回應。

來自用戶端憑證授予的存取權杖是包含 OAuth 2.0 範圍的授權機制。通常權杖會包含自訂範圍宣告，以授權對存取受保護的 API 執行 HTTP 操作。如需詳細資訊，請參閱 使用資源伺服器進行範圍、M2M 和 API 授權。

用戶端憑證授予您的帳 AWS 單增加成本。如需詳細資訊，請參閱 Amazon Cognito 定價。

透過 Amazon Cognito 使用者集區在授權碼授與中使用 PKCE

Amazon Cognito 支援授權代碼交換證明金鑰 (PKCE) 身份驗證。PKCE 是為公共客戶端授予 OAuth 2.0 授權碼的擴展。PKCE 防止贖回被截獲的授權碼。

亞馬遜認知如何使用 PKCE

若要使用 PKCE 開始驗證，您的應用程式必須產生唯一的字串值。此字串是程式碼驗證器，Amazon Cognito 用來比較要求初始授權授權給用戶端以交換權杖之授權碼的用戶端的秘密值。

您的應用程序必須將 SHA256 哈希應用於代碼驗證器字符串，並將結果編碼為 base64。將雜湊字串傳遞給要求主體中的`code_challenge`參數。授權端點當您的應用程序交換令牌的授權代碼時，它必須包含明文中的代碼驗證程序字符串作為請求主體中的`code_verifier`參數。權杖端點Amazon Cognito 會在程式碼驗證器上執 hash-and-encode 行相同的作業。Amazon Cognito 僅在判斷程式碼驗證器導致授權請求中收到的相同程式碼挑戰時，才會傳回 ID、存取和重新整理權杖。

使用 PKCE 實作授權授權流程

1. 開啟 Amazon Cognito 主控台。如果出現提示，請輸入您的 AWS 認證。
2. 選擇 User Pools (使用者集區)。
3. 從清單中選擇現有的使用者集區，或建立使用者集區。如果您建立使用者集區，系統會在精靈期間提示您設定應用程式用戶端並設定託管的 UI。

a. 如果您建立新的使用者集區，請在引導式設定期間設定應用程式用戶端並設定託管 UI。

b. 如果您設定現有的使用者集區，請新增網域和公用應用程式用戶端 (如果尚未設定)。

4. 產生隨機的英數字串，通常是通用唯一識別碼 (UUID)，以建立 PKCE 的程式碼挑戰。此字串是您將在要求中提交給的code_verifier參數值權杖端點。

5. 使用 SHA256 算法散列code_verifier字符串。將雜湊作業的結果編碼為 base64。此字串是您將在要求中提交給的code_challenge參數值授權端點。

   下列Python範例會產生code_verifier並計算code_challenge：

```
#!/usr/bin/env python3

import random
from base64 import urlsafe_b64encode
from hashlib import sha256
from string import ascii_letters
from string import digits

# use a cryptographically strong random number generator source
rand = random.SystemRandom()

code_verifier = ''.join(rand.choices(ascii_letters + digits, k=128))
code_verifier_hash = sha256(code_verifier.encode()).digest()
code_challenge = urlsafe_b64encode(code_verifier_hash).decode().rstrip('=')

print(f"code challenge: {code_challenge}")
print(f"code verifier: {code_verifier}")
```

   以下是Python指令碼輸出的範例：

```
code challenge: Eh0mg-OZv7BAyo-tdv_vYamx1boOYDulDklyXoMDtLg
code verifier: 9D-aW_iygXrgQcWJd0y0tNVMPSXSChIc2xceDhvYVdGLCBk-
JWFTmBNjvKSdOrjTTYazOFbUmrFERrjWx6oKtK2b6z_x4_gHBDlr4K1mRFGyE8yA-05-_v7Dxf3EIYJH
```

6. 完成託管 UI 登錄，並使用 PKCE 授予授權代碼請求。以下是範例 URL：

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com&code_challenge=Eh0mg-OZv7BAyo-
tdv_vYamx1boOYDulDklyXoMDtLg&code_challenge_method=S256
```

7. 收集授權code並使用令牌端點將其兌換為令牌。以下是請求示例：

```
POST /oauth2/token HTTP/1.1
Host: mydomain.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 296

redirect_uri=https%3A%2F%2Fwww.example.com&
client_id=1example23456789&
code=7378f445-c87f-400c-855e-0297d072ff03&
grant_type=authorization_code&
code_verifier=9D-aW_iygXrgQcWJd0y0tNVMPSXSChIc2xceDhvYVdGLCBk-
JWFTmBNjvKSdOrjTTYazOFbUmrFERrjWx6oKtK2b6z_x4_gHBDlr4K1mRFGyE8yA-05-_v7Dxf3EIYJH
```

8. 檢閱回應。它將包含 ID，訪問和刷新令牌。如需有關使用 Amazon Cognito 使用者集區權杖的詳細資訊，請參閱將權杖用於使用者集區。

託管 UI 和聯合錯誤回應

託管 UI 或聯合登入中的登入程序可能會傳回錯誤。下列是一些可能導致以錯誤結束驗證的情況。

- 使用者執行使用者集區無法完成的操作。
- Lambda 觸發程序沒有以預期的語法回應。
- 您的身分提供者 (IdP) 傳回錯誤。
- Amazon Cognito 無法驗證您的使用者所提供的屬性資訊。
- 您的 IdP 沒有向映射的屬性傳送所需的宣告。

當 Amazon Cognito 發生錯誤時，會以下列其中一種方式進行通訊。

1. Amazon Cognito 傳送的請求參數中包含錯誤的重新導向 URL。
2. Amazon Cognito 在託管 UI 中顯示錯誤。

Amazon Cognito 附加到請求參數的錯誤具有下列格式。

```
https://<Callback URL>/?error_description=error+description&error=error+name
```

當您協助使用者在無法執行操作而提交錯誤資訊時，請要求使用者擷取 URL 以及頁面的文字或是螢幕擷取畫面。

> **Note**
>
> Amazon Cognito 錯誤說明並非固定字串，因此您不應使用依賴固定模式或格式的邏輯。

OIDC 和社交身分提供者錯誤訊息

您的身分提供者可能會傳回錯誤訊息。當 OIDC 或 OAuth 2.0 IdP 傳回符合標準的錯誤時，Amazon Cognito 會將您的使用者重新導向至回呼 URL，並將提供者錯誤回應新增至錯誤請求參數。Amazon Cognito 會將提供者名稱和 HTTP 錯誤代碼新增至現有的錯誤字串。

下列 URL 是從 IdP 傳回錯誤至 Amazon Cognito 的重新導向範例。

```
https://www.amazon.com/?error_description=LoginWithAmazon+Error+-+400+invalid_request
+The+request+is+missing+a+required+parameter+%3A+client_secret&error=invalid_request
```

由於 Amazon Cognito 只會傳回從提供者收到的資訊，因此您的使用者可能會看到此資訊的子集。

當您的使用者透過 IdP 進行初始登入遇到問題時，IdP 會直接向您的使用者傳送任何錯誤訊息。當 Amazon Cognito 向您的 IdP 產生請求以驗證使用者的工作階段時，會向您的使用者轉送錯誤訊息。Amazon Cognito 從下列端點轉送 OAuth 和 OIDC IdP 錯誤訊息。

`/token`

Amazon Cognito 會與 IdP 交換授權碼以取得存取權杖。

`/.well-known/openid-configuration`

Amazon Cognito 會探索發布者端點的路徑。

`/.well-known/jwks.json`

為了驗證使用者的 JSON 網頁權杖 (JWT)，Amazon Cognito 會探索您的 IdP 用來簽署權杖的 JSON 網頁金鑰 (JWK)。

由於 Amazon Cognito 不會啟用可能會回傳 HTTP 錯誤的 SAML 2.0 提供者傳出工作階段，因此，SAML 2.0 IdP 工作階段期間的使用者錯誤，不會包括在此提供者錯誤訊息格式之內。

Amazon Cognito 使用者集區 API 參考

透過 Amazon Cognito 使用者集區，您可以讓使用者用 Web 和行動應用程式註冊和登入。您可以為已驗證的使用者變更密碼和為未驗證的使用者啟動忘記密碼流程。如需詳細資訊，請參閱使用者集區身分驗證流程及將權杖用於使用者集區。

Amazon Cognito 使用者集區 API 包含用於檢視和修改使用者集區和使用者，以及執行使用者身分驗證和授權的操作。如需合併至 Amazon Cognito 使用者集區 API 之 API 操作類別的說明，請參閱 使用 Amazon Cognito 使用者集區 API 和使用者集區端點。

如需 Amazon Cognito 使用者集區 API 操作和語法的詳細清單，請參閱 Amazon Cognito 使用者集區 API 參考。Amazon Cognito 使用者集區 API 參考的每個頁面都會連結參考資料，其中包含各種 AWS SDK 的語法和範例。

Amazon Cognito 身分集區 (聯合身分) API 參考

有了 Amazon Cognito 身分集區，您的 Web 和行動應用程式使用者可以取得權限有限的臨時 AWS 憑證，讓他們能夠存取其他 AWS 服務。

如需完整的身分集區 (聯合身分) API 參考，請參閱 Amazon Cognito API 參考。

Amazon Cognito Sync API 參考

Amazon Cognito Sync 是 AWS 服務和用戶端程式庫，可讓您跨裝置同步應用程式相關的使用者資料。

如需 Amazon Cognito Sync API 參考的詳細資訊，請參閱 Amazon Cognito Sync API 參考。

Amazon Cognito 的文件歷史記錄

下表說明 Amazon Cognito 文件的重要增補項目。我們還會根據您傳送的意見回饋，對文件進行頻繁的小幅更新。如要提交意見回饋，請找出 Amazon Cognito 文件中任何頁面底部的 Feedback (意見反應) 連結。

變更	描述	日期
增加了對預令牌 Lambda 觸發器中複雜對象的支持	您現在可以將數組和 JSON 對象添加到 ID 和訪問令牌聲明中。	2024年5月30日
已更新有關已驗證許可和 Amazon Cognito 的信息。	Amazon 驗證許可現在可以更直接地與 Amazon Cognito 集成。	2024年5月15日
多區域 Amazon SES 驗證身分識別。	在 AWS 區域 沒有 Amazon SES 的情況下，Amazon Cognito 使用者集區會負載平衡兩個遠端區域之間的電子郵件。	2024年5月10日
新增 M2M 授權和管理成本的相關資訊。	了解如何在 Amazon Cognito 使用者集區中使用用戶端登入資料授與 machine-to-machine (M2M) 使用案例。	2024年5月9日
Amazon Cognito 現已在歐洲 (西班牙) 和亞太區域 (海得拉巴) AWS 區域推出。	您現在可以在歐洲 (西班牙) 和亞太區域 (海德拉巴) 區域建立 Amazon Cognito 資源。	2024年4月15日
Amazon Cognito 現已在亞太區域 (墨爾本) AWS 區域推出。	您現在可以在亞太區域 (墨爾本) 區域建立 Amazon Cognito 資源。	2024年4月4日
在 Flutter 中為 Amazon Cognito 用戶池添加了一個示例 Android 應用程序。	您可以從上的範例 Flutter 應用程式建立適用於 Amazon	2024年4月4日

	Cognito 的入門行動應用程式。GitHub	
新的開始內容	開始使用的擴充內容、常見案例、多租用戶最佳做法，以及登入後存取資源。	2024年4月1日
Amazon Cognito 現已在歐洲 (蘇黎世) AWS 區域推出。	您現在可以在歐洲 (蘇黎世) 區域建立 Amazon Cognito 資源。	2024年3月14日
Amazon Cognito 現已在中東（阿聯酋）AWS 區域上市。	您現在可以在中東 (阿拉伯聯合大公國) 區域建立 Amazon Cognito 資源。	2024年3月8日
新的 SAML 功能和改進的內容。	您現在可以簽署 SAML 要求、加密 SAML 回應，以及設定 IdP 起始的 SAML SSO。	2024年2月1日
可增加配額。	您現在可以為 Amazon Cognito 請求率配額購買額外容量。	2024年1月25日
Amazon Cognito 身分集區支援 Service Quotas 中的請求費率。	您現在可以在 Service Quotas 主控台中監控 Amazon Cognito 身分集區的 requests-per-second (RPS) 配額，並請求增加請求。	2023 年 12 月 19 日
添加了用於自定義訪問令牌內容的新功能。	您現在可以新增、修改和移除使用者集區存取權杖中的宣告和範圍。	2023 年 12 月 12 日
改善應用程式用戶端和 OAuth 範圍的相關內容。	對 使用者集區應用程式用戶端 和 使用資源伺服器進行範圍、M2M 和 API 授權 進行清晰的編輯和更正。舊版主控台指示已移除。	2023 年 11 月 14 日

改進了有關設備和設備驗證的內容。	關於使用裝置金鑰和裝置 SRP 驗證的新內容。	2023 年 10 月 18 日
已更新 AWS Management Console 指引。	移除相關主題內的使用者集區主控台參考並重新分配主題，以及在 Amazon Cognito 主控台中新增標籤式組織的指引。	2023 年 8 月 30 日
不強調直接訪問登錄端點。	新增使用者集區 登入端點 的視覺化概觀並強調開始透過 授權端點 進行身分驗證。	2023 年 8 月 30 日
Amazon Cognito 現已在亞太區域 (大阪) 和以色列 (特拉維夫) AWS 區域推出。	您現在可以在亞太區域 (大阪) 和以色列 (特拉維夫) 區域建立 Amazon Cognito 資源。	2023 年 8 月 30 日
介紹了有關具有 Amazon 驗證許可的 Amazon Cognito 授權的信息。	在您的應用程式中，您可以調用 Verified Permissions API，以從中央授權機構產生存取決策。	2023 年 8 月 1 日
新增將使用者集區詳細使用者活動記錄到 Amazon Lo CloudWatch gs 的新功能。	現在，您可以將電子郵件和 SMS 消息傳遞錯誤 CloudWatch 記錄到日誌組中。	2023 年 8 月 1 日
已更新識別集區來賓使用者 AWS 受管理原則的相關資訊。	身分集區來賓使用者的權限範圍現在包含內嵌工作階段原則和 AWS 受管理的工作階段原則。	2023 年 5 月 16 日
Amazon Cognito 身分識別集區的內容改進和新主控台指示。	新增了主控台逐步解說，以反映新的控制台體驗，改進了身分集區的代碼整合詳細資訊。	2023 年 5 月 16 日
服務主頁和用戶池首頁的增加和改進。	已更新 Amazon Cognito 和使用者集區的概觀頁面。	2023 年 5 月 16 日
用戶池令牌文檔的一般改進。	更新範例權杖，並新增驗證權杖的新資訊。	2023 年 2 月 16 日

您現在可以在中 AWS CloudTrail記錄 Amazon Cognito 身分識別集區資料事件。	CloudTrail 支援在記錄資料事件的追蹤中選擇 Amazon Cognito 身分集區大量 API 操作。	2023 年 2 月 15 日
更新 Lambda 觸發器範例和說明。	Lambda 觸發器範例已更新至第 3 JavaScript 版。您現在可以將 Lambda 觸發程序直接關聯至 API 動作。	2023 年 1 月 31 日
Amazon Cognito 身分集區會將 AWS 受管政策套用至未經驗證的工作階段。	使用增強型流程進行驗證的身分集區使用者現在會將額外的 AWS 受管理原則套用至其工作階段。	2023 年 1 月 31 日
新增程式碼範例。	本指南現在包含各種程式設計語言中適用於 Amazon Cognito 應用程式的範例程式碼。	2023 年 1 月 23 日
已新增 API 模型和使用 Amazon Cognito 使用者集區的身份驗證的相關資訊。	Amazon Cognito 使用者集區具有多種供請求授權使用的 API 介面和格式。	2022 年 12 月 15 日
Amazon Cognito 現已在歐洲（米蘭）AWS 區域上市。	現在您可以在歐洲 (米蘭) 區域建立 Amazon Cognito 使用者集區。	2022 年 12 月 6 日
新增有關使用者集區刪除防護的資訊。	當您使用建立新的使用者集區時 AWS Management Console，預設情況下會保護該使用者集區不會遭到刪除。	2022 年 10 月 20 日
已新增託管使用者介面的使用者指南，以及託管 UI 中 TOTP MFA 的相關資訊。	您的使用者現在可以在 Amazon Cognito 託管 UI 中註冊 TOTP MFA 裝置。您現在可以預覽預設的託管 UI。	2022 年 9 月 8 日

添加了有關 AWS WAF 和 Amazon Cognito 的信息。	您現在可以將 AWS WAF 網頁 ACL 與 Amazon Cognito 使用者集區建立關聯。	2022 年 8 月 3 日
增加了更多示例 AWS CloudTrail 事件。	Amazon Cognito 現在會將聯合和託管的 UI 請求記錄到您的追蹤中。	2022 年 6 月 15 日
已新增有關雙步驟屬性驗證的資訊。	您現在可以選擇使用者是否必須先驗證新的電子郵件地址或電話號碼才能登入。	2022 年 6 月 9 日
更新的聯邦文檔。新的 IP 地址傳播功能。	更新了設置用戶池社交的演練。 IdPs已新增有關聯合身分使用者描述檔與屬性映射的資訊。添加了有關設備指紋的新信息以實現高級安全性	2022 年 5 月 31 日
登入同盟使用者，無需與託管 UI 互動	已新增有關如何將應用程式加入書籤的新頁面，讓 Amazon Cognito 以無訊息方式將使用者導向聯合登入。	2022年5月29日
適用於 Amazon Cognito 使用者集區的區域內簡訊和電子郵件簡訊	您現在可以針對 SMS 訊息使用 Amazon 簡單通知服務，而 Amazon 簡單電子郵件服務可用於與使用者集區 AWS 區域相同的電子郵件訊息。	2022 年 3 月 14 日
配額頁面的更新	新增並澄清資源和要求率配額。	2022 年 1 月 10 日
全新 Amazon Cognito 用者集區主控台體驗	已更新在更新後的 Amazon Cognito 主控台中建立與管理使用者集區的指示。	2021 年 11 月 18 日

RevokeToken API 和撤銷端點	您可以使用此 RevokeToken 作業來撤銷使用者的重新整理權杖。	2021 年 6 月 10 日
多租戶最佳做法	已新增多租用戶應用程式的最佳做法。	2021 年 3 月 4 日
存取控制的屬性	Amazon Cognito 身分識別集區提供存取控制 (AFAC) 的屬性，讓客戶可以授與使用者存取 AWS 資源。可根據用來與 Amazon Cognito 聯合的身分提供者使用者屬性，完成授權。	2021 年 1 月 15 日
自訂簡訊寄件者 Lambda 觸發器和自訂電子郵件寄件者	自訂簡訊寄件者 Lambda 觸發器和自訂電子郵件寄件者 Lambda 觸發器可讓您啟用第三方供應商，從 Lambda 函數程式碼中傳送電子郵件和簡訊通知給您的使用者。	2020 年 11 月 30 日
Amazon Cognito 令牌更新	更新的過期資訊已新增至存取、ID 和重新整理權杖。	2020 年 10 月 29 日
Amazon Cognito Service Quotas	Service Quotas 適用於 Amazon Cognito 類別配額。您可以使用 Service Quotas 主控台來檢視配額使用情況、要求增加配額，以及建立 CloudWatch 警示來監控配額使用情況。在此變更的過程中，Amazon Cognito 使用者集區的可用 CloudWatch指標區段已更新，以反映新資訊。新的區段名稱為：追蹤配額和使用情況以 CloudWatch及 Service Quotas	2020 年 10 月 29 日

Amazon Cognito 配額分類	配額類別可協助您監控配額使用量並要求增加。配額會根據常見的使用案例分類。	2020 年 8 月 17 日
美 AWS 國 GovCLoud 端支援 Amazon Cognito	Amazon Cognito 現在在 AWS GovCloud (美國) 區域受到支援。	2020 年 5 月 13 日
Amazon Cognito Pinpoint 文檔更新	新增服務連結角色。說明已更新於「使用 Amazon Pinpoint 分析搭配 Amazon Cognito 使用者集區」。	2020 年 5 月 13 日
全新 Amazon Cognito 專屬安全章節	安全性章節可協助您的組織取得有關 AWS 服務內建和可設定安全性的深入資訊。我們的新章節提供有關雲端和雲端安全性的資訊。	2020 年 4 月 30 日
Amazon Cognito 身份池現在支持使用蘋果登錄	除了 cn-north-1 區域之外，Amazon Cognito 營運的所有區域均已提供 Sign in with Apple 功能。	2020 年 4 月 7 日
新的臉書 API 版本控制	新增 Facebook API 版本選擇。	2020 年 4 月 3 日
用戶名不區分大小寫更新	新增了在建立使用者集區之前，啟用不區分使用者名稱大小寫的建議。	2020 年 2 月 11 日
關於新資訊 AWS Amplify	已新增使用 AWS Amplify SDK 和程式庫將 Amazon Cognito 與您的網頁或行動應用程式整合的相關資訊。移除使用 Amazon Cognito 開發套件以取代 AWS Amplify的詳細資訊。	2019 年 11 月 22 日

使用者集區觸發器的新屬性	Amazon Cognito 現在在傳遞給大多`clientMetadata` 數使用者集區觸發器 AWS Lambda 函數的事件資訊中包含一個參數。您可以使用此參數，以其他資料增強自訂驗證工作流程。	2019 年 10 月 4 日
更新限制	ListUsers API 動作的節流限制已更新。	2019 年 6 月 25 日
新限制	使用者集區的軟性限制現在包含使用者人數的限制。	2019 年 6 月 17 日
亞馬遜網路使用者集區的 Amazon SES 電子郵件設定	您可以設定使用者集區，以便 Amazon Cognito 使用 Amazon SES 組態將電子郵件寄給您的使用者。此設定允許 Amazon Cognito 使用超出預定的更高傳輸數量傳送電子郵件。	2019 年 4 月 8 日
標記支持	已新增標記 Amazon Cognito 資源的相關資訊。	2019 年 3 月 26 日
變更自訂網域的憑證	如果您使用自訂網域託管 Amazon Cognito 託管 UI，即可視需要變更該網域所用的 SSL 憑證。	2018 年 12 月 19 日
新限制	已增加新的限制，規定了每位使用者可隸屬的群組數目上限。	2018 年 12 月 14 日
更新的限制	已針對使用者集區更新了軟性限制。	2018 年 12 月 11 日

驗證電子郵件地址和電話號碼的文件更新	新增了有關設定使用者集區以要求使用者在您的應用程式中註冊時進行電子郵件或電話驗證的資訊。	2018 年 11 月 20 日
測試電子郵件的文檔更新	新增了有關測試應用程式從 Amazon Cognito 發出電子郵件的指引。	2018 年 11 月 13 日
Amazon Cognito 高級安全	新增新的安全功能，讓開發人員能保護其應用程式和使用者不受惡意機器人侵擾、保護使用者帳戶以避免登入資料洩漏，並根據所計算的登入嘗試風險，自動調整登入所需的挑戰。	2018 年 6 月 14 日
Amazon Cognito 託管用戶界面的自定義域	允許開發人員將自己的完全自訂網域用於 Amazon Cognito 使用者集區中的託管 UI。	2018 年 4 月 6 日
Amazon Cognito 戶集區 OIDC 身份提供商	透過 OpenID Connect (OIDC) 身分提供者新增使用者集區登入，例如 Salesforce 或 Ping Identity。	2018 年 5 月 17 日
Amazon Cognito Lambda 遷移觸發器	新增有關 Lambda Migration 觸發功能的頁面	二○一八年四月八日
Amazon Cognito 人員指南更新	新增頂層「什麼是 Amazon Cognito」和「Amazon Cognito 入門」。另外新增常用案例和重新整理過的使用者集區目錄。新增「Amazon Cognito 使用者集區入門」一節。	2018 年 4 月 6 日

Amazon Cognito 高級安全測試版	新增新的安全功能，讓開發人員能保護其應用程式和使用者不受惡意機器人侵擾、保護使用者帳戶杜絕已網際網路上其他地方洩漏的萬用登入資料，並根據所計算的登入嘗試風險，自動調整登入所需的挑戰。	2017 年 11 月 28 日
Amazon Pinpoint 整合	新增可使用 Amazon Pinpoint 來為 Amazon Cognito 使用者集區應用程式提供分析，以及讓 Amazon Pinpoint 行銷活動的使用者資料更豐富的功能。	2017 年 9 月 26 日
Amazon Cognito 使用者集區的聯合和內建應用程式 UI 功能	新增可讓使用者透過 Facebook、Google、Login with Amazon 或 SAML 身分提供者登入使用者集區的功能。新增可自訂的內建應用程式使用者界面，以及具有自訂宣告的 OAuth 2.0 支援。	2017 年 8 月 10 日
HIPAA 和 PCI 合規性相關功能變更	新增可讓使用者使用電話號碼或電子郵件地址做為其使用者名稱的功能。	2017 年 7 月 6 日
使用者群組和角色型存取控制功能	新增可建立及管理使用者群組的管理功能。管理員可以根據群組成員資格和管理員建立的規則，指派 IAM 角色給使用者。	2016 年 12 月 15 日
文件更新	已更新範例，說明如何在使用者集區中使用 AWS Lambda 觸發程序。	二○一六年十一月二
文件更新	更新的 iOS 代碼的例子。	2016 年 11 月 18 日

文件更新	新增使用者帳戶確認流程的相關資訊。	2016 年 11 月 9 日
建立使用者帳戶功能	新增可透過 Amazon Cognito 主控台和 API 來建立使用者帳戶的管理功能。	2016 年 10 月 6 日
使用者匯入功能	新增 Cognito 使用者集區的大量匯入功能。使用此功能可將使用者從您現有的身分提供者移轉至 Amazon Cognito 使用者集區。	2016 年 9 月 1 日
Cognito 使用者集區的一般可用性	新增 Cognito 使用者集區功能。此功能可讓您利用使用者集區來建立和維護使用者目錄，以及將註冊和登入資料新增至您的行動應用程式和 Web 應用程式。	2016 年 7 月 28 日
SAML 支援	新增透過安全聲明標記語言 (SAML 2.0) 來向身分提供者進行身分驗證的支援。	2016 年 6 月 23 日
CloudTrail 整合	增加了集成 AWS CloudTrail.	2016 年 2 月 18 日
與 Lambda 整合事件	可讓您執行 AWS Lambda 函數以回應 Amazon Cognito 中的重要事件。	2015 年 4 月 9 日
資料串流至 Amazon Kinesis	提供對資料串流的控制和洞察。	2015 年 3 月 4 日
OpenID Connect 支持	可支援 OpenID Connect 供應商。	二○一四年十一月二
推送同步	可支援靜音推播同步。	2014 年 11 月 6 日

已新增開發人員驗證身分支援	可讓擁有自己的身分驗證和身分管理系統的開發人員，在 Amazon Cognito 中視為是身分提供者。	2014 年 9 月 29 日
Amazon Cognito 一般可用性		2014 年 7 月 10 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。