



開發人員指南

Amazon Cognito



Amazon Cognito: 開發人員指南

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

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

Table of Contents

什麼是 Amazon Cognito ?	1
使用者集區	2
身分集區	3
Amazon Cognito 功能	4
使用者集區	4
身分集區	6
Amazon Cognito 使用者集區和身分池比較	8
Amazon Cognito 入門	12
區域可用性	12
Amazon Cognito 定價	12
術語和概念	12
一般	13
使用者集區	15
身分集區	19
入門 AWS	20
註冊 AWS 帳戶	20
建立具有管理存取權的使用者	21
使用者集區入門	23
您的第一個應用程式和使用者集區	23
其他應用程式選項	25
React SPA 範例	26
Flutter 行動應用程式範例	29
後續步驟	32
新增社交供應商	33
新增 SAML IdP	40
身分集區入門	43
在 Amazon Cognito 中建立身分集區	43
設定 SDK	45
整合身分提供者	45
取得憑證	45
其他入門選項	46
整合應用程式	48
使用 進行身分驗證 AWS Amplify	49
使用 Amplify 建立使用者介面 (UI)	50

使用 AWS SDK 進行身分驗證	50
身分驗證的運作方式	51
受管登入身分驗證	52
SDK 身分驗證	55
第三方身分提供者身分驗證	58
身分集區身分驗證	60
使用 AWS SDKs	64
使用 Amazon Verified Permissions 進行授權	65
具有已驗證許可的 API 授權	67
Amazon Cognito 使用者的政策範例	70
程式碼範例	72
Amazon Cognito 身分	73
基本概念	74
案例	99
Amazon Cognito 身分提供者	100
基本概念	111
案例	264
Amazon Cognito Sync	429
基本概念	430
多租戶最佳實務	433
每個租戶使用者集區	434
每個租戶應用程式用戶端	436
每個租戶的使用者集區群組	438
每個租用戶的自訂屬性	440
每個租用戶自訂範圍	442
資源範例	444
多重租用安全建議	446
常見 Amazon Cognito 案例	447
以使用者集區進行身分驗證	447
存取伺服器端資源	448
透過 API Gateway 和 Lambda 存取資源	449
使用使用者集區和身分集區存取 AWS 服務	449
以身分集區進行第三方身分驗證以及存取 AWS 服務	450
使用 Amazon Cognito 存取 AWS AppSync 資源	451
Amazon Cognito 使用者集區	453
功能	454

註冊	454
登入	455
受管登入	456
安全	456
自訂使用者體驗	456
監控與分析	457
Amazon Cognito 身分池整合	457
使用者集區功能計劃	458
選取功能計劃	459
依計劃列出的功能	460
基本計劃功能	462
Plus 計劃功能	466
關閉不符合資格的功能	468
安全最佳實務	469
在網路層級保護您的使用者集區	469
防止簡訊濫用	470
了解公有身分驗證	470
使用用戶端秘密保護機密用戶端	473
保護其他秘密	473
使用者集區管理最低權限	474
保護和驗證字符	476
決定您要信任的身分提供者	477
了解範圍對存取使用者設定檔的影響	477
清理使用者屬性的輸入	477
身分驗證	478
實作身分驗證流程	478
須知事項	481
身分驗證流程範例	483
受管登入身分驗證	486
SDK 身分驗證	488
身分驗證流程	492
SDK 授權模型	514
第三方 IdP 登入	527
聯合登入在 Amazon Cognito 使用者集區中的運作方式	528
作為使用 Amazon Cognito 服務提供者的應用程式責任	529
Amazon Cognito 使用者集區第三方登入相關須知	529

身分提供者	530
社交身分提供者	536
SAML 供應商	543
OIDC 供應商	571
映射 IdP 屬性	581
連結聯合身分使用者	586
受管登入	589
受管登入當地語系化	591
使用 設定受管登入 AWS Amplify	592
使用 Amazon Cognito 主控台設定受管登入	592
檢視您的登入頁面	593
自訂身分驗證頁面	594
受管登入和託管 UI 的須知事項	594
設定網域	597
品牌與自訂	609
使用 Lambda 觸發程序	627
須知事項	629
設定觸發	631
使用者集區 Lambda 觸發程序事件	631
使用者集區 Lambda 觸發程序的常用參數	632
用戶端中繼資料	633
依操作觸發來源	636
依函數觸發來源	642
註冊前	646
確認後	653
身分驗證前	656
身分驗證後	660
自訂挑戰	664
產生權杖前	679
遷移使用者	700
自訂訊息	705
自訂寄件者	711
管理使用者	729
允許使用者註冊	730
註冊及確認使用者帳戶	733
以管理員身分建立使用者	756

新增群組至使用者集區	762
管理及搜尋使用者	765
密碼	772
將使用者匯入使用者集區	776
Attributes	793
使用者集區字符	807
ID 字符	809
存取權杖	813
重新整理權杖	816
撤銷權杖	821
驗證 JSON Web 權杖	823
管理使用者集區字符過期和快取	829
登入後存取資源	832
使用 Verified Permissions 存取資源	448
存取 API Gateway 資源	835
使用身分集區存取 AWS 資源	836
M2M 和範圍	841
API 授權	841
Machine-to-machine(M2M) 授權	842
關於範圍	843
關於資源伺服器	844
額外功能	849
更新使用者集區和應用程式用戶端	849
應用程式用戶端	853
使用裝置	862
使用 Amazon Pinpoint 分析	867
電子郵件設定	872
簡訊設定	885
使用安全性功能	893
新增 MFA	895
威脅防護	912
AWS WAF Web ACLs	937
區分大小寫	941
刪除保護	942
管理使用者披露	944
使用者集區端點參考	950

受管登入端點	951
聯合端點	957
OAuth 2.0 授予	986
使用 PKCE	987
受管登入和聯合錯誤回應	989
Amazon Cognito 身分集區	991
設定身分集區	993
建立 身分集區	993
使用者 IAM 角色	995
已進行身分驗證和未進行身分驗證的身分	995
啟用或停用訪客存取	995
變更與身分類型相關聯的角色	996
編輯身分提供者	997
刪除身分集區	998
從身分集區刪除身分	998
使用 Amazon Cognito Sync 與身分集區	999
身分集區身分驗證流程	1001
IAM 角色	1010
設定信任政策	1011
存取政策	1014
角色信任和許可	1024
安全最佳實務	1025
IAM 組態最佳實務	1026
身分集區組態最佳實務	1028
使用屬性進行存取控制	1029
透過 Amazon Cognito 身分集區使用屬性進行存取控制	1030
使用屬性進行存取控制的政策範例	1031
關閉存取控制的屬性	1033
預設供應商對應	1033
使用以角色為基礎的存取控制	1035
建立角色以進行角色映射	1035
授予傳送角色許可	1036
使用權杖來指派角色給使用者	1037
使用以規則為基礎的映射來指派角色給使用者	1038
要用在以規則為基礎之映射中的宣告	1040
以角色為基礎的存取控制最佳實務	1041

取得憑證	1041
使用 憑證	1048
第三方身分提供者	1050
Facebook	1051
登入 Amazon	1058
Google	1062
使用 Apple 登入	1070
Open ID Connect 供應商	1075
SAML 身分供應商	1078
開發人員驗證的身分	1080
了解身分驗證流程	1080
定義開發供應商名稱，並將其與身分集區建立關聯	1081
實作身分提供者	1082
更新登入映射 (僅限 Android 和 iOS)	1090
取得權杖 (伺服器端)	1090
連接至現有的社交身分	1092
支援供應商之間的轉換	1092
切換身分	1096
Android	1096
iOS - Objective-C	1096
iOS - Swift	1097
JavaScript	1097
Unity	1098
Xamarin	1098
Amazon Cognito Sync	1100
Amazon Cognito Sync 入門	1100
在 Amazon Cognito 中設定身分集區	1101
存放及同步資料	1101
跨用戶端同步資料	1101
初始化 Amazon Cognito Sync 用戶端	1102
了解資料集	1103
在資料集中讀取和寫入資料	1105
將本機資料與同步存放區同步化	1107
處理事件回呼	1110
Android	1111
iOS - Objective-C	1113

iOS - Swift	1116
JavaScript	1119
Unity	1121
Xamarin	1124
實作推送同步	1126
建立 Amazon Simple Notification Service (Amazon SNS) 應用程式	1127
在 Amazon Cognito 主控台中啟用推播同步	1127
在您的應用程式中使用播送同步：Android	1128
在您的應用程式中使用推播同步：iOS - Objective-C	1130
在您的應用程式中使用推播同步：iOS - Swift	1133
實作 Amazon Cognito Sync 串流	1135
使用 Amazon Cognito Events 自訂工作流程	1138
安全	1143
資料保護	1143
資料加密	1144
身分與存取管理	1145
目標對象	1145
使用身分驗證	1146
使用政策管理存取權	1148
Amazon Cognito 如何與 IAM 搭配運作	1150
身分型政策範例	1158
故障診斷	1162
使用服務連結角色	1164
日誌記錄和監控	1168
監控成本	1169
匯出使用者集區日誌	1171
監控配額和用量	1181
CloudTrail 日誌	1195
法規遵循驗證	1222
恢復能力	1223
區域資料考量	1223
基礎架構安全	1224
組態與漏洞分析	1224
AWS 受管政策	1224
政策更新	1226
標記 資源	1228

支援的資源	1228
標籤限制	1228
使用主控台管理標籤	1229
AWS CLI 範例	1229
指派標籤	1229
檢視標籤	1231
移除標籤	1231
在建立資源時套用標籤	1232
API 動作	1232
使用者集區標籤適用的 API 動作	1232
身分集區標籤適用的 API 動作	1233
配額	1234
了解 API 請求率配額	1234
配額分類	1234
適用特殊請求率處理的 Amazon Cognito 使用者集區 API 操作	1235
每月作用中使用者	1235
管理 API 請求速率配額	1237
確認配額需求	1237
最佳化請求率	1238
追蹤配額使用量	1238
追蹤每月作用中使用者 MAUs)	1239
請求提高配額	1240
使用者集區請求率配額	1240
身分集區請求率配額	1251
資源數量和大小的配額	1252
文件歷史紀錄	1259
.....	mcclxxiv

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。

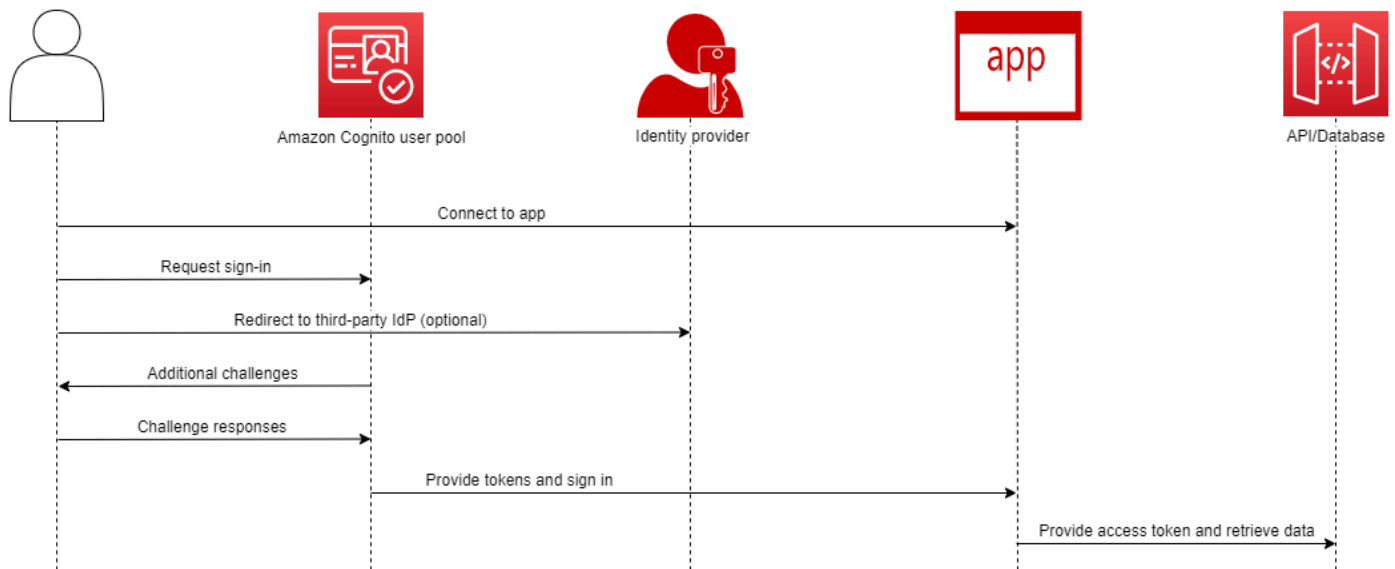
主題

- [使用者集區](#)
- [身分集區](#)
- [Amazon Cognito 功能](#)
- [Amazon Cognito 使用者集區和身分池比較](#)
- [Amazon Cognito 入門](#)
- [區域可用性](#)
- [Amazon Cognito 定價](#)
- [常見 Amazon Cognito 術語和概念](#)
- [入門 AWS](#)

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

Amazon Cognito user pools

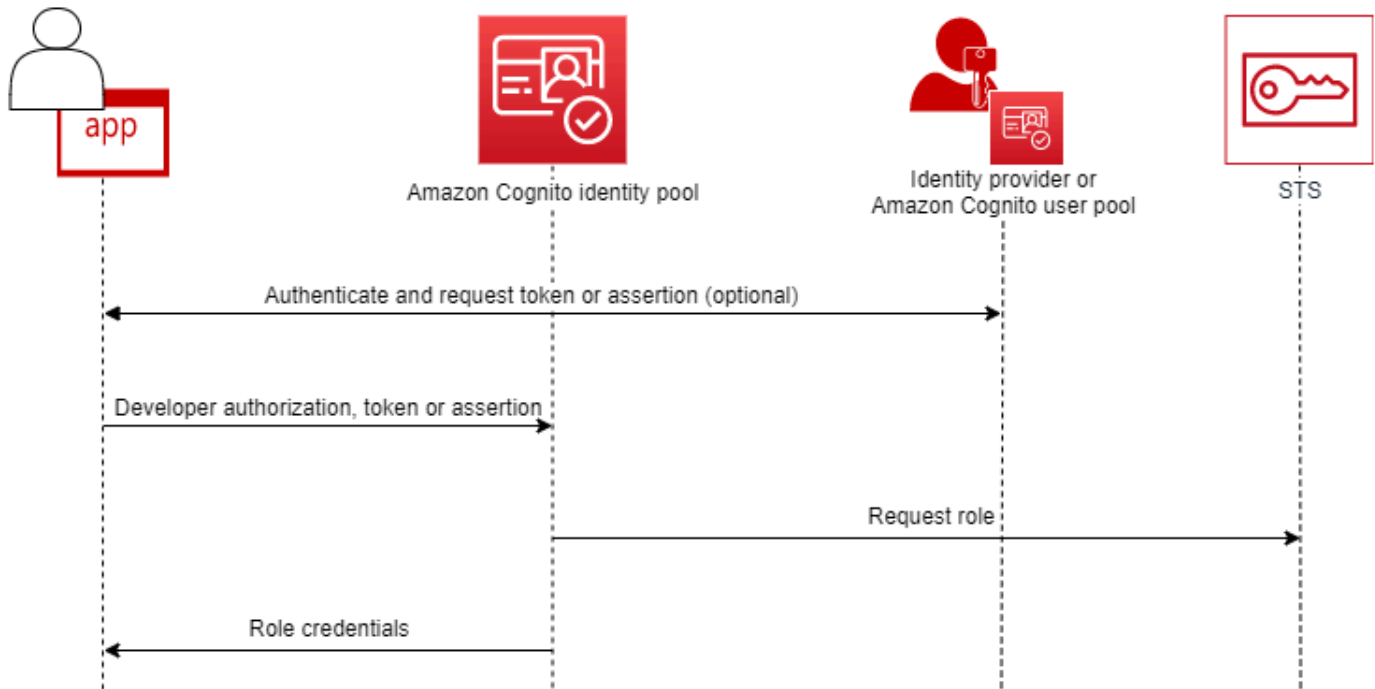


當您要對應用程式或 API 進行驗證和使用者授權時，請建立使用者集區。使用者集區是具有自助服務和管理員驅動的使用者建立、管理和驗證的使用者目錄。您的使用者集區可以是獨立的目錄和 OIDC 身分提供者 (IdP)，也可以是第三方員工和客戶身分提供者的中繼服務提供者 (SP)。針對您組織在具使用者集區之 SAML 2.0 與 OIDC IdPs 中的人力身份，您可以在應用程式提供單一登入 (SSO)。針對您組織在公開 OAuth 2.0 身分儲存 Amazon、Google、Apple 和 Facebook 中的客戶身分，您也可以在此應用程式提供 SSO。如需客戶身份與存取管理 (CIAM) 的更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

Amazon Cognito federated identities (identity pools)



當您想要授權已驗證或匿名的使用者存取您的 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 憑證，以為使用者提供資源。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。身分集區同時使用角色型和屬性型存取控制，來管理使用者存取 AWS 資源的授權。

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. 您的應用程式會與身分集區交換使用者集區字符，以取得您可以搭配 AWS APIs 和 AWS Command Line Interface () 使用的臨時 AWS 登入資料 AWS CLI。
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，並提供授權存取給，AWS 服務 例如 Amazon S3 和 Amazon DynamoDB。

如需使用身分池和使用者的更多範例，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 Management Console、AWS 開發套件或 AWS Command Line Interface () 中管理和自訂這些使用者設定檔AWS CLI。

Amazon Cognito 使用者集區接受來自第三方 IdP 的權杖和宣告，並收集使用者屬性到發布至應用程式的 JWT 中。您可以在一組 JWT 上標準化應用程式，而 Amazon Cognito 會處理與 IdPs 的互動，將其宣告映射至中央權杖格式。

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 使用者集區也可以扮演雙重角色，既是 IdP 服務提供者 (SP) 也是應用程式執行 IdP。Amazon Cognito 使用者集區可以連接至消費者 IdP，如 Facebook 和 Google，或是生產力 IdP，如 Okta 和 Active Directory 同盟服務 (ADFS)。

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 憑證。

Amazon Cognito 使用者集區功能

功能	描述
OIDC identity provider	Issue ID tokens to authenticate users
Authorization server	Issue access tokens to authorize user access to APIs
SAML 2.0 service provider	Transform SAML assertions into ID and access tokens
OIDC relying party	Transform OIDC tokens into ID and access tokens
Social provider relying party	Transform ID tokens from Apple, Facebook, Amazon, or Google to your own ID and access tokens
Authentication frontend service	Sign up, manage, and authenticate users with managed login
API support for your own UI	Create, manage and authenticate users through authentication API requests in supported AWS SDKs ¹
Multi-factor authentication	Use SMS messages, TOTP, or your user's device as an additional authentication factor ¹
Security monitoring & response	Secure against malicious activity and insecure passwords ¹
Customize authentication flows	Build your own authentication mechanism, or add custom steps to existing flows ²

Groups	Create logical groupings of users, and a hierarchy of IAM role claims when you pass tokens to identity pools
Customize tokens	Customize your ID and access tokens with new, modified, and suppressed claims
Customize user attributes	Assign values to user attributes and add your own custom attributes

1 聯合身分使用者無法使用 功能。

2 聯合身分和受管登入使用者無法使用 功能。

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

身分集區是您指派給使用者或訪客並授權接收臨時 AWS 登入資料的一組唯一識別符或身分。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。身分集區為身分建立的字符可以從 AWS Security Token Service () 擷取臨時工作階段憑證 AWS STS。

若要補充已驗證的身分，您也可以設定身分集區以在沒有 IdP 身分驗證的情況下授權 AWS 存取。您可以使用 提供自訂身分驗證。 [開發人員驗證的身分](#) 您也可以使用 [未經驗證的身分](#)，將臨時 AWS 登入資料授予訪客使用者。

使用身分集區時，您有兩種方式可與 中的 IAM 政策整合 AWS 帳戶。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，會 AWS STS 標記他們的工作階段 `Department: marketing`。您的 Amazon S3 儲存貯體將根據 [aws:PrincipalTag](#) 條件允許讀取操作，該條件需要的 `Department` 標籤值為 `marketing`。

Amazon Cognito 身分池功能

功能	描述
Amazon Cognito user pool relying party	Exchange an ID token from your user pool for web identity credentials from AWS STS
SAML 2.0 service provider	Exchange SAML assertions for web identity credentials from AWS STS
OIDC relying party	Exchange OIDC tokens for web identity credentials from AWS STS
Social provider relying party	Exchange OAuth tokens from Amazon, Facebook, Google, Apple, and Twitter for web identity credentials from AWS STS
Custom relying party	With AWS credentials, exchange claims in any format for web identity credentials from AWS STS
Unauthenticated access	Issue limited-access web identity credentials from AWS STS without authentication
Role-based access control	Choose an IAM role for your authenticated user based on their claims, and configure your roles to only be assumed in the context of your identity pool
Attribute-based access control	Convert claims into principal tags for your AWS STS temporary session, and use IAM policies to filter resource access based on principal tags

如需身分集區的詳細資訊，請參閱 [Amazon Cognito 身分集區入門](#) 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 identity provider	Issue OIDC ID tokens to authenticate app users	✓	
User directory	Store user profiles for authentication	✓	
Authorize API access	Issue access tokens to authorize user access to APIs (including user profile self-service API operations), databases, and other resources that accept OAuth scopes	✓	
IAM web identity authorization	Generate tokens that you can exchange with AWS STS for temporary AWS credentials		✓
SAML 2.0 service provider & OIDC identity provider	Issue customized OIDC tokens based on claims from a SAML 2.0 identity provider	✓	

OIDC relying party & OIDC identity provider	Issue customized OIDC tokens based on claims from an OIDC identity provider	✓
OAuth 2.0 relying party & OIDC identity provider	Issue customized OIDC tokens based on scopes from OAuth 2.0 social providers like Apple and Google	✓
SAML 2.0 service provider & credentials broker	Issue temporary AWS credentials based on claims from a SAML 2.0 identity provider	✓
OIDC relying party & credentials broker	Issue temporary AWS credentials based on claims from an OIDC identity provider	✓
Social provider relying party & credentials broker	Issue temporary AWS credentials based on JSON web tokens from developer applications with social providers like Apple and Google	✓
Amazon Cognito user pool relying party & credentials broker	Issue temporary AWS credentials based on JSON web tokens from Amazon Cognito user pools	✓

Custom relying party & credentials broker	Issue temporary AWS credentials to arbitrary identities, authorized by developer IAM credentials	✓
Authentication frontend service	Sign up, manage, and authenticate users with managed login	✓
API support for your own authentication UI	Create, manage and authenticate users through API requests in supported AWS SDKs ¹	✓
MFA	Use SMS messages, TOTP, or your user's device as an additional authentication factor ¹	✓
Security monitoring & response	Protect against malicious activity and insecure passwords ¹	✓
Customize authentication flows	Build your own authentication mechanism, or add custom steps to existing flows ¹	✓
User groups	Create logical groupings of users, and a hierarchy of IAM role claims when you pass tokens to identity pools	✓

Customize tokens	Customize your ID and access tokens with new, modified, and suppressed claims and scopes	✓
AWS WAF web ACLs	Monitor and control requests to your authentication front end with AWS WAF	✓
Customize user attributes	Assign values to user attributes and add your own custom attributes	✓
Unauthenticated access	Issue limited-access web identity credentials from AWS STS without authentication	✓
Role-based access control	Choose an IAM role for your authenticated user based on their claims, and configure your role trust to limit access to web identity users	✓
Attribute-based access control	Transform user claims into principal tags for your AWS STS temporary session, and use IAM policies to filter resource access based on principal tags	✓

1 功能不適用於聯合身分使用者。

Amazon Cognito 入門

如需使用者集區應用程式的範例，請參閱 [使用者集區入門](#)。

如需身分集區的簡介，請參閱 [Amazon Cognito 身分集區入門](#)。

如需使用者集區和身分集區之引導式設定體驗的連結，請參閱 [Amazon Cognito 的引導設定選項](#)。

若要開始使用 AWS SDK，請參閱 [AWS 開發人員工具](#)。如需 Amazon Cognito 特定的開發人員資源，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 [入門 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 AWS 可讓提供服務，包括 Amazon Cognito，具有非常高的可用性和備援，同時將延遲降至最低。

若要查看 Amazon Cognito 目前是否可在任何中使用 AWS 區域，請參閱 [AWS 依區域提供服務](#)。

若要了解區域 API 服務端點，請參閱 [AWS 區域和端點](#) Amazon Web Services 一般參考。

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](#)。

常見 Amazon Cognito 術語和概念

Amazon Cognito 提供 Web 和行動應用程式的登入資料。它會從身分和存取管理中常見的術語中提取和建置。提供許多通用身分和存取術語的指南。部分範例如下：

- IDPro 知識內文中的 [術語](#)
- [AWS Identity Services](#)

- NIST CSRC 的 [詞彙表](#)

下列清單說明 Amazon Cognito 獨有或 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

此清單中的術語並非 Amazon Cognito 特有的術語，且受到身分和存取管理從業者的廣泛認可。下列不是詳細的術語清單，而是本指南中特定 Amazon Cognito 內容的指南。

存取權杖

JSON Web 權杖 (JWT)，其中包含實體存取資訊系統 [之授權](#) 的相關資訊。

應用程式、應用程式

一般而言，行動應用程式。在本指南中，應用程式通常是連接到 Amazon Cognito 的 Web 應用程式或行動應用程式的速記。

屬性型存取控制 (ABAC)

應用程式根據使用者屬性決定資源存取權的模型，例如其職稱或部門。用於強制執行 ABAC 的 Amazon Cognito 工具會在使用者集區中包含 ID 字符，並在身分集區中包含 [主體標籤](#)。

身分驗證

為存取資訊系統而建立真實身分的程序。Amazon Cognito 接受第三方身分提供者的身分驗證證明，同時也是軟體應用程式的身分驗證提供者。

授權

將許可授予資源的程序。使用者集區 [存取字符](#) 包含應用程式可用來允許使用者和系統存取資源的資訊。

授權伺服器

產生 [JSON Web 字符](#) 的 OAuth 或 OpenID Connect (OIDC) 系統。Amazon Cognito 使用者集區 [受管授權伺服器](#) 是使用者集區中兩種身分驗證和授權方法的授權伺服器元件。使用者集區也支援 [SDK 身分驗證](#) 中的 API 挑戰-回應流程。

機密應用程式、伺服器端應用程式

使用者從遠端連線的應用程式，在應用程式伺服器上使用程式碼並存取秘密。這通常是 Web 應用程式。

Identity provider (IdP) (身分提供者 (IdP))

存放和驗證使用者身分的服務。Amazon Cognito 可以向[外部供應商](#)請求身分驗證，並成為應用程式的 IdP。

JSON Web 字符 (JWT)

JSON 格式的文件，其中包含有關已驗證使用者的宣告。ID 權杖驗證使用者、存取權杖授權使用者，以及重新整理權杖更新憑證。Amazon Cognito 會從[外部供應商](#)接收字符，並對應用程式或發出字符 AWS STS。

Machine-to-machine(M2M) 授權

授權請求給non-user-interactive機器實體 API 端點的程序，例如 Web 伺服器應用程式層。使用者集區在[存取權杖](#)中使用 OAuth 2.0 範圍的用戶端憑證授予中提供 M2M 授權。

多重要素驗證 (MFA)

使用者在提供使用者名稱和密碼之後，提供其他身分驗證的要求。Amazon Cognito 使用者集區具有[本機使用者的](#) MFA 功能。

OAuth 2.0 (社交) 供應商

使用者集區或身分集區的 IdP，提供 [JWT](#) 存取和重新整理權杖。Amazon Cognito 使用者集區會在使用者驗證後自動與社交供應商互動。

OpenID Connect (OIDC) 供應商

將 IdP 擴展 [OAuth](#) 規格以提供 ID 字符的使用者集區或身分集區。Amazon Cognito 使用者集區會在使用者驗證後自動與 OIDC 供應商互動。

Passkey、WebAuthn

一種身分驗證形式，其中使用者裝置上的密碼編譯金鑰或通行金鑰可提供其身分驗證證明。使用者驗證硬體或軟體驗證器中是否存在生物識別或 PIN 碼機制。通行密鑰是網路釣魚持久性，並且繫結至特定網站/應用程式，提供安全的無密碼體驗。Amazon Cognito 使用者集區支援使用通行密鑰登入。

無密碼

一種身分驗證形式，使用者不需要輸入密碼。無密碼登入的方法包括傳送到電子郵件地址和電話號碼的一次性密碼 (OTPs)，以及通行密鑰。Amazon Cognito 使用者集區支援使用 OTPs 和通行密鑰登入。

公有應用程式

一種獨立於裝置上的應用程式，其程式碼存放在本機，無法存取秘密。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 互動組態的元件。

角色型存取控制 (RBAC)

根據使用者的功能指定授予存取權的模型。Amazon Cognito 身分集區實作 RBAC，並區分 IAM 角色。

服務供應商 (SP)、依賴方 (RP)

依賴 IdP 來宣告使用者可信任的應用程式。Amazon Cognito 做為外部 IdPs SP，以及做為應用程式型 SPs IdP。

SAML 供應商

使用者集區或身分集區的 IdP，可產生使用者傳遞至 Amazon Cognito 的數位簽署聲明文件。

通用唯一識別符 (UUID)

套用至物件的 128 位元標籤。Amazon Cognito UUIDs 在每個使用者集區或身分集區中都是唯一的，但不符合特定的 UUID 格式。

使用者目錄

使用者及其屬性的集合，可將該資訊提供給其他系統。Amazon Cognito 使用者集區是使用者目錄，也是從外部使用者目錄整合使用者的工具。

使用者集區

當您在本指南中看到下列清單中的術語時，它們會參考使用者集區的特定功能或組態。

自適應身分驗證

[進階安全性](#)的功能，可偵測潛在的惡意活動，並對[使用者設定檔](#)套用額外的安全性。

進階安全功能

選用元件，可新增工具以確保使用者安全。

應用程式用戶端

將使用者集區的設定定義為一個應用程式的 IdP 的元件。

回呼 URL、重新導向 URI、傳回 URL

[應用程式用戶端](#)中的設定，以及使用者集區[授權伺服器](#)請求中的參數。回呼 URL 是[應用程式中](#)已驗證使用者的初始目的地。

以選擇為基礎的身分驗證

使用者集區的 API 身分驗證形式，其中每個使用者都有一組可供他們使用的登入選項。他們的選擇可能包括使用者名稱和密碼，包含或不包含 MFA、通行密鑰登入，或使用電子郵件或簡訊一次性密碼進行無密碼登入。您可以透過請求身分驗證選項清單或宣告偏好的選項，為使用者塑造選擇程序。

與[用戶端型身分驗證](#)比較。

用戶端型身分驗證

使用使用者集區 API 和應用程式後端建立的身分驗證形式，使用 AWS SDKs 建置。在宣告式身分驗證中，您的應用程式會獨立判斷使用者應執行的登入類型，以及預先輸入的請求。

與[以選擇為基礎的身分驗證](#)進行比較。

憑證洩漏

[進階安全性](#)的功能，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

判斷已符合先決條件以允許新使用者登入的程序。確認通常透過電子郵件地址或電話號碼[驗證](#)完成。

自訂身分驗證

使用 [Lambda 觸發程序](#) 的身分驗證程序延伸，可定義其他使用者挑戰和回應。

裝置身分驗證

以使用信任裝置 ID 的登入來取代 [MFA](#) 的身分驗證程序。

網域、使用者集區網域

託管受[管登入頁面](#)的 Web 網域 AWS。您可以在您擁有的網域中設定 DNS，或在 AWS 擁有的網域中使用識別子網域字首。

基本計劃

具有使用者集區中最新開發的[功能計畫](#)。Essentials 計劃不包含 [Plus 計劃](#)中的自動學習安全功能。

外部供應商、第三方供應商

與使用者集區具有信任關係的 IdP。使用者集區可做為外部提供者與您的應用程式之間的中繼實體，使用 SAML 2.0、OIDC 和社交提供者管理身分驗證程序。使用者集區會將外部供應商身分驗證結果合併為單一 IdP，讓您的應用程式可以使用單一 OIDC 依賴方程式庫來處理許多使用者。

特徵計劃

您可以為使用者集區選取的功能群組。特徵計劃在 AWS 帳單中具有不同的成本。新使用者集區預設為 [Essentials 計劃](#)。

目前的計劃

- [Lite 計劃](#)
- [基本計劃](#)
- [Plus 計劃](#)

聯合身分使用者、外部使用者

使用者集區中由[外部供應商](#)驗證的使用者。

託管 UI（傳統）、託管 UI 頁面

使用者集區網域上身分驗證前端、依賴方和身分提供者服務的早期版本。託管 UI 具有一組基本功能，以及簡化的外觀和風格。您可以將託管 UI 品牌套用到上傳標誌影像檔案，以及具有一組預先決定的 CSS 樣式的檔案。與[受管登入](#)比較。

Lambda 觸發程序

中的函數 AWS Lambda，使用者集區可以在使用者身分驗證程序的重點自動叫用。您可以使用 Lambda 觸發來自訂身分驗證結果。

本機使用者

使用者集區[使用者目錄](#)中的使用者[描述](#)檔，不是透過與[外部供應商](#)的身分驗證所建立。

連結的使用者

來自[外部供應商](#)的使用者，其身分會與[本機使用者](#)合併。

Lite 計劃

具有最初使用使用者集區啟動之功能的特徵[計劃](#)。Lite 計劃不包含 [Essentials 計劃](#)中的新功能或 [Plus 計劃](#)中的自動學習安全功能。

受管授權伺服器、託管 UI 授權伺服器、授權伺服器

[受管登入](#)的元件，託管服務，以與[使用者集區網域](#)上的 IdPs 和應用程式互動。[託管的 UI](#) 與其提供的使用者互動功能中的受管登入不同，但具有相同的授權伺服器功能。

受管登入、受管登入頁面

[使用者集區網域](#)上的一組網頁，用於託管服務以進行使用者身分驗證。這些服務包含以 [IdP](#) 身分運作的函數、第三方 IdPs [的依賴方](#)，以及使用者互動式身分驗證 UI 的伺服器。當您為使用者集區設定網域時，Amazon Cognito 會將所有受管登入頁面上線。

您的應用程式匯入 OIDC 程式庫，可叫用使用者的瀏覽器，並將其導向受管登入 UI 以進行註冊、登入、密碼管理和其他身分驗證操作。身分驗證之後，OIDC 程式庫可以處理身分驗證請求的結果。

受管登入身分驗證

使用[使用者集區網域](#)上的服務登入，並使用使用者互動瀏覽器頁面或 HTTPS API 請求完成。應用程式使用 OpenID Connect (OIDC) 程式庫處理受管登入身分驗證。此程序包括使用[外部供應商](#)登入、使用互動式受管登入頁面的本機使用者登入，以及 [M2M 授權](#)。使用傳統[託管 UI](#) 進行身分驗證也屬於此術語。

與 [AWS SDK 身分驗證](#) 比較。

Plus 計劃

具有使用者集區中最新開發和進階安全功能的功能[計畫](#)。

SDK 身分驗證、AWS SDK 身分驗證

一組身分驗證和授權 API 操作，您可以使用 AWS SDK 將其新增至應用程式後端。此身分驗證模型需要您自己的自訂登入機制。API 可以登入[本機使用者](#)和[連結的使用者](#)。

與[受管登入身分驗證](#) 比較。

威脅防護

在使用者集區中，威脅防護是指旨在緩解對您的身分驗證和授權機制的威脅的技術。自適應身分驗證、已洩露憑證偵測和 IP 地址封鎖清單屬於威脅防護類別。

字符自訂

權杖產生前 [Lambda 觸發](#)的結果，可在執行時間修改使用者的 ID 或存取權杖。

使用者集區、Amazon Cognito 身分提供者、**cognito-idp**、Amazon Cognito 使用者集區

針對使用 OIDC IdPs 的應用程式，具有身分驗證和授權服務 AWS 的資源。

驗證

確認使用者擁有電子郵件地址或電話號碼的程序。使用者集區會將代碼傳送給已輸入新電子郵件地址或電話號碼的使用者。當他們將程式碼提交至 Amazon Cognito 時，他們會驗證訊息目的地的擁有權，並可以從使用者集區接收其他訊息。此外，請參閱[確認](#)。

使用者設定檔、使用者帳戶

使用者 [目錄中的使用者](#) 項目。所有使用者，包括來自第三方 IdPs 的使用者，在其使用者集區中都有設定檔。

身分集區

當您在本指南中看到下列清單中的術語時，它們會參考身分集區的特定功能或組態。

存取控制的屬性

在身分集區中實作 [屬性型存取控制](#)。身分集區會將使用者屬性做為標籤套用至使用者憑證。

基本（傳統）身分驗證

您可以在此驗證程序自訂 [使用者登入](#) 資料的請求。

開發人員驗證的身分

使用開發人員 [登入資料授權身分集區使用者](#) 登入資料的身分驗證程序。 [???](#)

開發人員登入資料

身分集區管理員的 IAM API 金鑰。

增強型身分驗證

身分驗證流程，會選取 IAM 角色，並根據您在身分集區中定義的邏輯套用主體標籤。

Identity

將應用程式使用者及其[使用者登入](#)資料連結至與身分集區具有信任關係之外部[使用者目錄](#)中設定檔的 [UUID](#)。

身分集區、Amazon Cognito 聯合身分、Amazon Cognito 身分、**cognito-identity**

針對使用[臨時 AWS 登入資料](#)的應用程式，具有身分驗證和授權服務 AWS 的資源。

未驗證的身分

未使用身分集區 IdP 登入的使用者。您可以在使用者進行身分驗證之前，允許使用者為單一 IAM 角色產生有限的使用者登入資料。

使用者登入資料

使用者在身分集區身分驗證後收到的暫時 AWS API 金鑰。

入門 AWS

開始使用 Amazon Cognito 之前，請先自行設定一些必要的 AWS 資源。如果您已經可以登入 AWS 帳戶，您可以略過本節。如果您要尋找有關使用 AWS 登入資料註冊和登入的資訊，請繼續閱讀。擁有具有足夠 AWS Identity and Access Management (IAM) 許可的登入資料之後，您就可以開始使用[使用者集區](#)和[身分集區](#)。

註冊 AWS 帳戶

如果您沒有 AWS 帳戶，請完成下列步驟來建立一個。

註冊 AWS 帳戶

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

註冊程序的一部分包括接聽電話或文字訊息，並在電話鍵盤上輸入驗證碼。

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

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

建立具有管理存取權的使用者

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

保護您的 AWS 帳戶根使用者

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

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

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

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

建立具有管理存取權的使用者

1. 啟用 IAM Identity Center。

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

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

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

以具有管理存取權的使用者身分登入

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

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

指派存取權給其他使用者

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

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

2. 將使用者指派至群組，然後對該群組指派單一登入存取權。

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

使用者集區入門

您有一個應用程式需要身分驗證和存取控制。您可以在 OpenID Connect (OIDC) 架構中用於單一登入 (SSO)。Amazon Cognito 具有使用 AWS SDK 處理應用程式後端身分驗證邏輯的工具，以及用於叫用用戶端中的瀏覽器來存取受管授權伺服器。

Amazon Cognito 主控台會引導您從偏好的應用程式架構檢視建立使用者集區。從那裡，您可以繼續新增功能，例如使用外部[社交](#)或 [SAML 2.0](#) 身分提供者 (IdPs) 的聯合登入。Amazon Cognito 主控台內的應用程式模型倚賴將 OIDC 程式庫新增至您的專案並叫用瀏覽器。

當您努力擴展功能集並納入更多 Amazon Cognito 元件時，請閱讀 [Amazon Cognito 使用者集區](#) 一章，了解您可以使用使用者集區執行的所有操作的完整說明。

本章和 Amazon Cognito 主控台內的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，以使用更多可供您使用的選項。然後，您可以更新您的應用程式以採用新功能並與 IdPs 互動。

如果您不想使用[受管登入頁面](#)，您可以使用 AWS SDK 或 建立具有自訂建置身分驗證介面的應用程式 AWS Amplify。您以這種方式建置的應用程式會與[使用者集區 API](#) 互動，且僅適用於驗證[本機使用者](#)。繼續在 [了解此身分驗證模型其他應用程式選項](#)。

主題

- [在 Amazon Cognito 主控台中建立新的應用程式](#)
- [其他應用程式選項](#)
- [將更多功能和安全性選項新增至您的使用者集區](#)

在 Amazon Cognito 主控台中建立新的應用程式

使用者集區會將身分驗證選項新增至軟體應用程式。如需最簡單的入門體驗，請前往 Amazon Cognito 主控台，並依照其中的指示操作。其中的建立程序不僅會引導您設定使用者集區資源，還會引導您設定應用程式的初始部分。

當您準備好開始時，請導覽至 [Amazon Cognito 主控台](#)，然後選取 [按鈕](#) 以建立新的使用者集區。設定程序將引導您完成組態和程式設計語言選項。

身分驗證概念的其他資源

- [使用 Amazon Cognito 使用者集區進行身分驗證](#)

- [了解 API、OIDC 和受管登入頁面身分驗證](#)
- [身分驗證如何與 Amazon Cognito 搭配使用](#)
- [將 Amazon Cognito 身分驗證和授權與 Web 和行動應用程式整合](#)

為您的應用程式建立 Amazon Cognito 資源

1. 導覽至 [Amazon Cognito 主控台](#)。若要將許可指派給 IAM 主體，以便他們可以建立和管理 Amazon Cognito 資源，請參閱 [AWS Amazon Cognito 的 受管政策](#)。AmazonCognitoPowerUser 政策足以建立使用者集區。
2. 從使用者集區功能表中選取建立使用者集區，或選取五分鐘內免費開始使用。
3. 在定義應用程式下，選擇最適合您要為其建立身分驗證和授權服務之應用程式案例的應用程式類型。
4. 在為您的應用程式命名中，輸入描述性名稱或繼續使用預設名稱。
5. 您必須在支援建立使用者集區後無法變更的設定選項下進行一些基本選擇。
 - a. 在登入識別符的選項下，告訴我們您希望如何在使用者登入時識別使用者。您可以偏好使用者產生的使用者名稱、電子郵件地址或電話號碼。您也可以允許多個選項的組合。Amazon Cognito 接受您在[受管登入](#)表單的使用者名稱欄位中在此處設定的選項。
 - b. 在註冊的必要屬性下，告訴我們當使用者註冊新帳戶時，您要收集哪些使用者資訊。在受管登入頁面中，Amazon Cognito 會顯示所有必要屬性的提示。

登入識別碼的選項會影響您所需的屬性。使用者名稱需要每個使用者的電子郵件或電話屬性，以便他們可以在電子郵件或簡訊中接收密碼重設代碼。電子郵件需要電子郵件屬性，而電話號碼需要電話號碼屬性。

6. 在新增傳回 URL 下，輸入使用者完成身分驗證後應用程式的重新導向路徑。此位置應該是應用程式中使用 OpenID Connect (OIDC) 程式庫來處理使用者身分驗證結果的路由。測試應用程式的傳回 URL 範例為 `https://localhost:3000/callback`。在 Amazon Cognito 主控台中 NodeJS 應用程式範例中，此路由使用 [openid-client](#) 來收集存取字符，並將其兌換為使用者資訊。您可以在建立資源後瀏覽開發平台的範例。
7. 選擇建立您的應用程式。Amazon Cognito 會使用應用程式類型的預設設定來建立使用者集區和應用程式用戶端。您可以在建立初始資源後設定其他選項，例如[外部身分提供者](#)和[多重要素驗證 \(MFA\)](#)。
8. 在設定應用程式頁面上，您可以立即取得應用程式的程式碼範例。若要探索您的新使用者集區，請向下捲動並選取前往概觀。

9. 若要在相同的使用者集區中新增更多應用程式，請導覽至應用程式用戶端選單，並新增新的應用程式用戶端。這將重複以應用程式為中心的建立程序，但只會將新的應用程式用戶端新增至現有的使用者集區。

使用此程序建立使用者集區和一或多個應用程式用戶端之後，您可以使用受管登入開始測試身分驗證操作。這些快速入門選項開放給公有自我註冊。我們建議您使用主控台程序建立測試環境，然後將最終設計移至生產環境。花時間熟悉 Amazon Cognito 的功能。然後，若要移至生產工作負載，請製作自訂組態，並使用 AWS CloudFormation 和 等自動化工具進行部署 AWS Cloud Development Kit (AWS CDK)。

Amazon Cognito 會在此程序中進行您無法反轉的一些預設組態。如需您無法變更的使用者集區設定，以及您可以在主控台中選擇的選項的詳細資訊，請參閱 [更新使用者集區和應用程式用戶端組態](#)。

設定	Effect	如何變更	其他資訊
Client secret (用戶端密碼)	在身分驗證請求中需要用戶端秘密雜湊。	使用傳統 Web 應用程式或Machine-to-machine應用程式設定檔建立新的應用程式用戶端。	使用應用程式用戶端的應用程式特定設定
偏好的使用者名稱	使用者集區不接受 preferred_username 屬性做為別名。	使用 AWS SDK 以程式設計方式建立使用者集區。	自訂登入屬性
區分大小寫	使用者集區使用者名稱不區分大小寫，例如JohnD視為與相同的使用者johnd。	使用 AWS SDK 以程式設計方式建立使用者集區。	使用者集區大小寫區分

其他應用程式選項

您可能擁有要與 Amazon Cognito 身分驗證整合的現有應用程式 UI。相較於 Amazon Cognito 使用者集區，您甚至可能擁有功能較低的現有身分驗證頁面。您可以使用適用於各種程式設計語言的 Amazon Cognito 整合 AWS SDKs將身分驗證元件新增至此類型的應用程式。一些範例如下。

如果您在 Amazon Cognito 主控台中為此目的建立使用者集區，則可能不需要擁有託管互動式登入頁面和 OpenID Connect (OIDC) 服務的[使用者集區網域](#)。在主控台中建立使用者集區的程序會自動為您產生網域。您可以從使用者集區的網域索引標籤中刪除此網域。其他選項包括使用 API 請求 AWS SDKs 和 CLI 中的自動設定選項，為您的應用程式以程式設計方式建立 AWS Amplify Amazon Cognito 資源。如需詳細資訊，請參閱[將 Amazon Cognito 身分驗證和授權與 Web 和行動應用程式整合](#)。

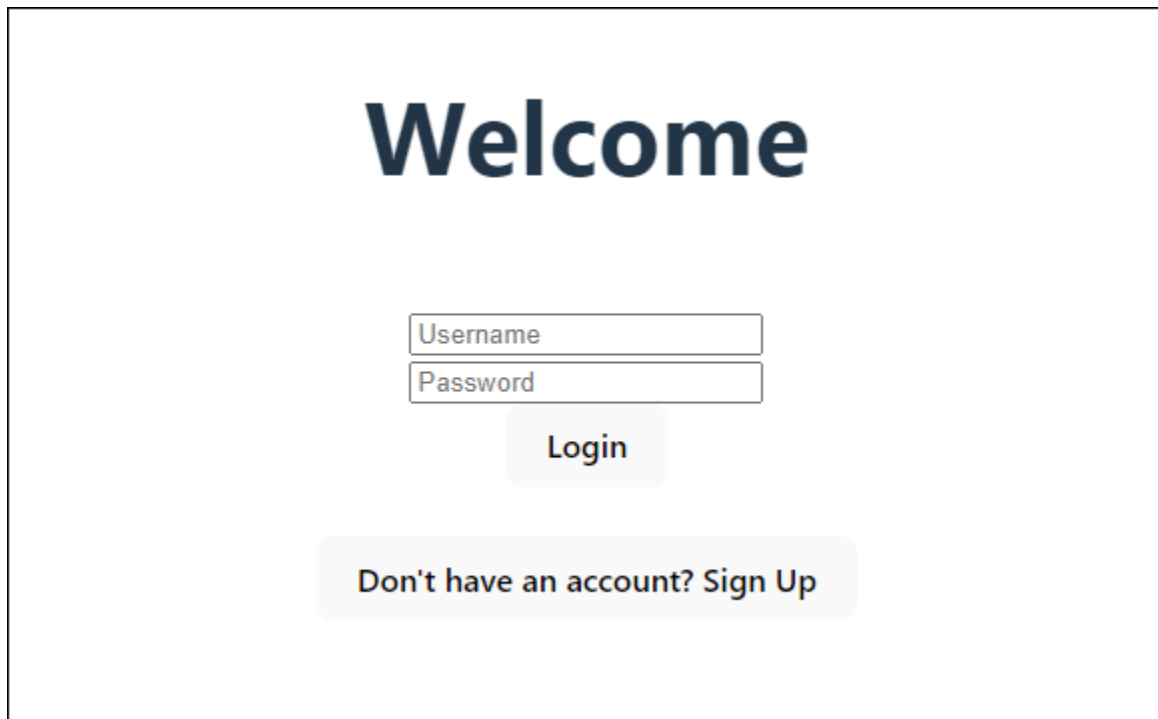
主題

- [設定 React 單頁應用程式範例](#)
- [使用 Flutter 設定範例 Android 應用程式](#)

設定 React 單頁應用程式範例

在本教學課程中，您將建立 React 單一頁面應用程式，您可以在其中測試使用者註冊、確認和登入。React 是適用於 Web 和行動應用程式的 JavaScript 型程式庫，著重於使用者介面 (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 開發 Web 應用程式的經驗，[請從 GitHub 下載範例應用程式](#)。

下列螢幕擷取畫面是您將建立之應用程式中的初始身分驗證頁面。



若要設定此應用程式，您的使用者集區必須符合下列要求：

- 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。使用者名稱要求：未選取使用者名稱區分大小寫。
- 不需要多重要素驗證 (MFA)。MFA 強制執行：選用 MFA。
- 您的使用者集區會使用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。
- 電子郵件是唯一必要的屬性。必要屬性：電子郵件。
- 使用者可以在使用者集區中註冊自己。自我註冊：選取啟用自我註冊。
- 您的初始應用程式用戶端是公有用戶端，允許使用使用者名稱和密碼登入。應用程式類型：公有用戶端、身分驗證流程：ALLOW_USER_PASSWORD_AUTH。

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。開發人員環境需求如下：

1. Node.js 已安裝並更新。
2. 已安裝節點套件管理員 (npm)，並至少更新至 10.2.3 版。
3. 環境可在網頁瀏覽器的 TCP 連接埠 5173 上存取。

建立 React Web 應用程式範例

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. 建立新的 React 服務。

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

3. 從 GitHub 上的 AWS 程式碼範例儲存庫複製 `cognito-developer-guide-react-example` [專案資料夾](#)。

```
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/
```

4. 導覽至專案中的 src 目錄。

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

5. 編輯config.json和取代下列值：

- a. YOUR_AWS_REGION 將取代為 AWS 區域 程式碼。例如：us-east-1。
 - b. YOUR_COGNITO_USER_POOL_ID 將取代為您指定用於測試的使用者集區的 ID。例如：us-east-1_EXAMPLE。使用者集區必須位於您在上一個步驟中輸入 AWS 區域 的中。
 - c. YOUR_COGNITO_APP_CLIENT_ID 將取代為您指定用於測試的應用程式用戶端 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. 如果您想要從 以外的 IP 存取範例應用程式localhost，請編輯package.json並將該行變更為 "dev": "vite", "dev": "vite --host 0.0.0.0",。
 7. 安裝您的應用程式。

```
npm install
```

8. 啟動應用程式。

```
npm run dev
```

9. 在 http://localhost:5173或 的 Web 瀏覽器中存取應用程式http://[IP address]:5173。
10. 使用有效的電子郵件地址註冊新使用者。
11. 從電子郵件訊息擷取確認碼。在應用程式中輸入確認碼。
12. 使用您的使用者名稱和密碼登入。

使用 建立 React 開發人員環境 Amazon Lightsail

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

使用 Lightsail，您可以快速建立已預先設定此範例應用程式的先決條件的小型伺服器執行個體。您可以使用瀏覽器型用戶端將 SSH 連線到執行個體，並在公有或私有 IP 地址連線到 Web 伺服器。

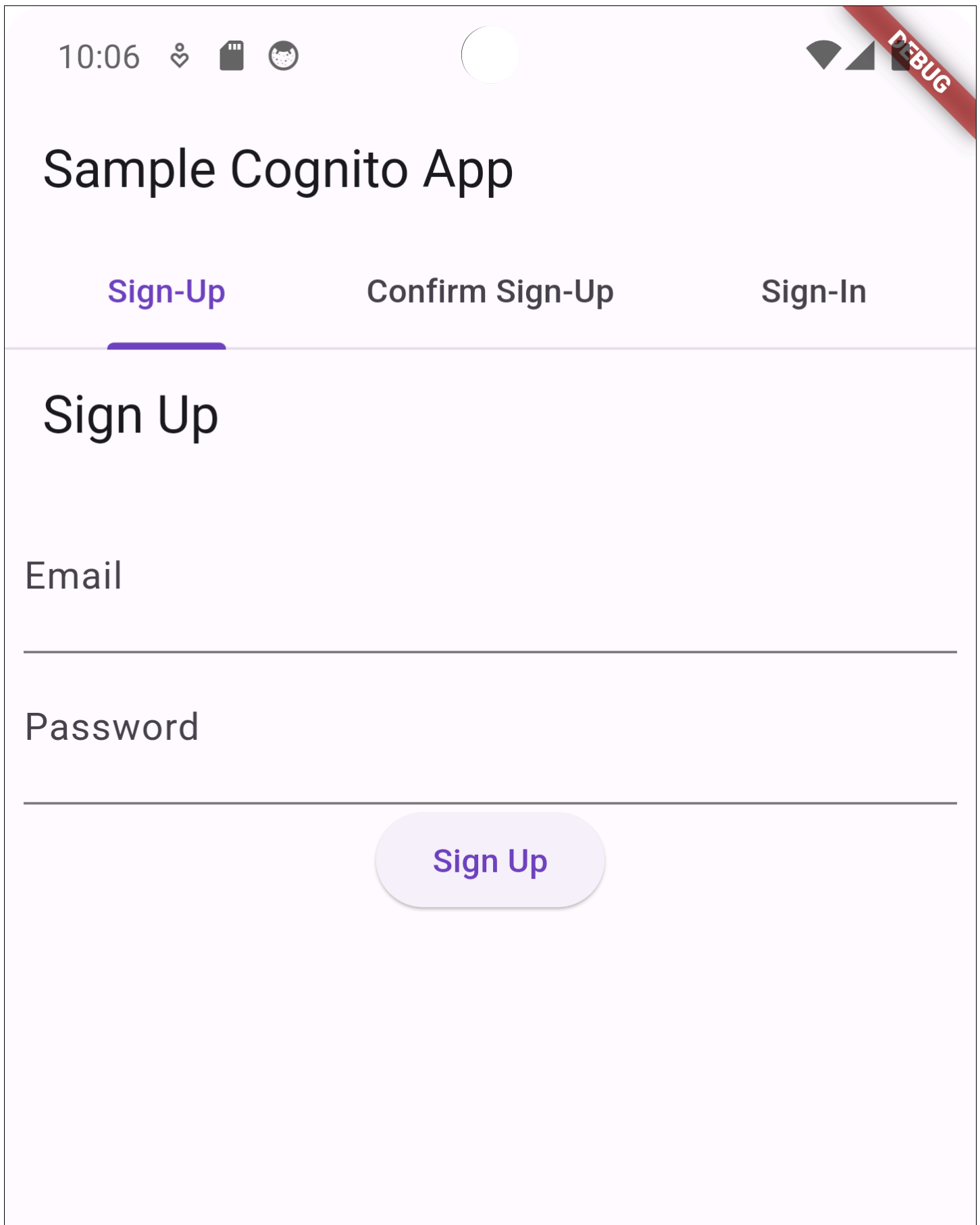
為此範例應用程式建立Lightsail執行個體

1. 前往 [Lightsail 主控台](#)。如果出現提示，請輸入您的 AWS 登入資料。
2. 選擇 建立執行個體。
3. 針對選取平台，選擇 Linux/Unix。
4. 針對選取藍圖，選擇 Node.js。
5. 在識別您的執行個體下，為您的開發環境提供易記的名稱。
6. 選擇 建立執行個體。
7. Lightsail 建立執行個體之後，請從 Connect 索引標籤中選取執行個體，然後選擇使用 SSH 連線。
8. SSH 工作階段會在瀏覽器視窗中開啟。執行 `node -v` 和 `npm -v` 以確認您的執行個體已佈建 Node.js 和最低 npm 版本 10.2.3。
9. 繼續[設定您的 React 應用程式](#)。

使用 Flutter 設定範例 Android 應用程式

在本教學課程中，您將在 Android Studio 中建立行動應用程式，您可以在其中模擬裝置並測試使用者註冊、確認和登入。此範例應用程式會在 Flutter 中為 Android 建立基本 Amazon Cognito 使用者集區行動用戶端。如果您已經有使用 Flutter 開發行動應用程式的經驗，[請從 GitHub 下載範例應用程式](#)。

下列螢幕擷取畫面顯示虛擬 Android 裝置上執行的應用程式。



若要設定此應用程式，您的使用者集區必須符合下列要求：

- 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。使用者名稱要求：未選取使用者名稱區分大小寫。
- 不需要多重要素驗證 (MFA)。MFA 強制執行：選用 MFA。
- 您的使用者集區會使用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。
- 電子郵件是唯一必要的屬性。必要屬性：電子郵件。
- 使用者可以在使用者集區中註冊自己。自我註冊：選取啟用自我註冊。
- 您的初始應用程式用戶端是公有用戶端，允許使用使用者名稱和密碼登入。應用程式類型：公有用戶端、身分驗證流程：ALLOW_USER_PASSWORD_AUTH。

建立應用程式

建立範例 Android 應用程式

1. 安裝 [Android Studio](#) 和 [命令列工具](#)。
2. 在 Android Studio 中，安裝 [Flutter 外掛程式](#)。
3. [在此範例應用程式中](#)，從 `cognito_flutter_mobile_app` 目錄的內容建立新的 Android Studio 專案。
 - 編輯 `assets/config.json` 並使用使用者集區 `<<YOUR_USER_POOL_ID>>` 和應用程式用戶端的 `<<YOUR_CLIENT_ID>>` ID 取代 `和`。IDs
4. 安裝 [Flutter](#)。
 - a. 將 Flutter 新增至 PATH 變數。
 - 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元件。

```
flutter doctor
```

 - 如果有任何元件遺失，請執行 `flutter doctor -v` 了解如何修正問題。
 - d. 變更至新 Flutter 專案的目錄並安裝相依性。
 - 執行 `flutter pub add amazon_cognito_identity_dart_2`。

- e. 執行 `flutter pub add flutter_secure_storage`。
5. 建立虛擬 Android 裝置。
 1. 在 Android Studio GUI 中，使用裝置 [管理員建立新裝置](#)。
 2. 在 CLI 中，執行 `flutter emulators --create --name android-device`。
6. 啟動您的虛擬 Android 裝置。
 1. 在 Android Studio GUI 中，選取虛擬裝置旁的開
始
示。
 2. 在 CLI 中，執行 `flutter emulators --launch android-device`。
7. 在虛擬裝置上啟動應用程式。
 1. 在 Android Studio GUI 中，選取部
署
示。
 2. 在 CLI 中，執行 `flutter run`。
8. 導覽至您在 Android Studio 中執行的虛擬裝置。
9. 使用有效的電子郵件地址註冊新使用者。
10. 從電子郵件訊息擷取確認碼。在應用程式中輸入確認碼。
11. 使用您的使用者名稱和密碼登入。

圖

圖

將更多功能和安全性選項新增至您的使用者集區

遵循教學課程完成範例應用程式後，您可以擴大使用者集區實作的範圍。或者，如果您未建立測試應用程式，請根據您的偏好設定建立新的使用者集區。您可以自訂其他應用程式的使用者集區功能，或[新增外部身分提供者](#)。當您規劃將 Amazon Cognito 使用者集區放入生產應用程式時，您可以評估[其他範例和教學課程](#)。

如果您的下一個優先順序是在使用者集區中檢查和套用應用程式安全選項，請參閱 [Amazon Cognito 使用者集區的安全最佳實務](#)。

Amazon Cognito 具有功能計劃，可在您選擇加入更高層時新增功能和安全性選項。您可以從 Lite 計劃開始，使用 Essentials 計劃新增進階身分驗證和授權選項，以及使用 Plus 計劃新增自動原因安全防護機制。如需詳細資訊，請參閱[使用者集區功能計劃](#)。

以下是一些額外的 Amazon Cognito 使用者集區功能：

- [將品牌套用到受管登入頁面](#)
- [將 MFA 新增到使用者集區](#)
- [具有威脅防護的進階安全性](#)
- [使用 Lambda 觸發程序來自訂使用者集區工作流程](#)
- [使用 Amazon Pinpoint 進行使用者集區分析](#)

如需 Amazon Cognito 身分驗證和授權模型的概觀，請參閱 [身分驗證如何與 Amazon Cognito 搭配使用](#)。

若要在使用者集區身分驗證成功 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 SDKs 之外，您也可以使用 [來管理您的使用者集區 AWS Command Line Interface](#)。

主題

- [將社交登入新增至您的使用者集區](#)
- [新增 SAML 2.0 身分提供者](#)

將社交登入新增至您的使用者集區

讓使用者能夠透過現有的公有或社交登入您的應用程式，身分提供者可以改善他們的身分驗證體驗。Amazon Cognito 使用者集區與 Facebook、Google、Amazon 和 Apple 等熱門的社交身分提供者 (IdPs) 整合，為您的使用者提供他們已經熟悉的便捷登入選項。

當您設定社交登入時，您會為使用者提供僅為您的應用程式建立專用帳戶的替代方案。這可以改善轉換率，並讓註冊程序更順暢。從使用者的角度來看，他們可以套用現有的社交登入資料以快速進行身分驗證，而不需要記住另一個使用者名稱和密碼。

在使用者集區中設定社交 IdP 需要幾個關鍵步驟。您必須向社交提供者註冊應用程式，才能取得用戶端 ID 和秘密。然後，您可以將社交 IdP 組態新增至使用者集區，指定您要請求的範圍，以及您要從

IdP 屬性映射的使用者集區屬性。在執行時間，Amazon Cognito 會處理與提供者的字符交換、映射使用者屬性，並以共用使用者集區格式將字符發行到您的應用程式。

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 應用程式，您會看到不同的選項。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. 從導覽列中，選擇產品，然後選擇從 Facebook Login 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 包含使用者集區網域與/oauth2/idpresponse端點。

```
https://your_user_pool_domain/oauth2/idpresponse
```

15. 選擇儲存變更。

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應用程式和服務，然後選擇使用 Amazon 登入。

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. 在左側導覽列上，選擇 APIs 和服務，然後選擇聲明同意畫面。
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` 和自訂網域的根。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的測試使用者，然後選擇儲存並繼續。
10. 再次展開左側導覽列，選擇 APIs 和服務，然後選擇登入資料。
11. 選擇建立憑證，然後選擇 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 Authorized JavaScript origins (授權的 JavaScript 來源) 下方，選擇 ADD URI (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。
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 [使用社交 IdP 設定您的使用者集區](#) 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 做為身分提供者之後，您將需要此值[使用社交 IdP 設定您的使用者集區](#)。
 3. 選擇 Continue (繼續)，然後選擇 Register (註冊)。
13. 選擇您剛從識別符頁面建立的服務 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，選取您先前建立做為主要應用程式 ID 的應用程式 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，選擇下載以下載私有金鑰、記下顯示的金鑰 ID，然後選擇完成。您在 [使用社交 IdP 設定您的使用者集區](#) 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。

使用 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入 AWS 登入資料。
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. 選擇社交和外部供應商選單。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 – 輸入上一節產生的應用程式用戶端 ID 和應用程式用戶端秘密。
 - Facebook – 輸入上一節產生的應用程式用戶端 ID 和應用程式用戶端秘密，然後選擇 API 版本（例如 2.12 版）。我們建議選擇最新的可能版本 - 每個 Facebook API 都有生命週期和棄用日期。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 和私有金鑰。
7. 輸入您要使用的已授權範圍名稱。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 的登入頁面後遇到失敗（這些可能來自 Amazon Cognito 的內部故障或開發人員撰寫的任何內容）。
 - 服務 ID 識別符會跨使用者集區和/或其他身分驗證服務使用。
 - 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。
 - 開發人員會刪除使用者，然後使用者再次登入，而不從其 Apple ID 設定檔中移除應用程式。
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. 選擇建立。
10. 從應用程式用戶端功能表中，選擇清單中的其中一個應用程式用戶端，並編輯託管 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.auth.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

Amazon Cognito 會取消未在 5 分鐘內完成的身分驗證請求，並將使用者重新導向至受管登入。此頁面會顯示「Something went wrong」錯誤訊息。

新增 SAML 2.0 身分提供者

您的應用程式使用者可以使用 SAML 2.0 身分提供者 (IdP) 登入。當您的客戶是組織的內部客戶或連結的企業時，您可以透過 IdPs 社交 IdP 選擇 SAML 2.0 IdPs。當社交 IdP 允許所有使用者註冊帳戶時，SAML IdP 更可能與組織控制的使用者目錄配對。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 身分提供者](#)。

您必須更新 SAML 身分提供者並設定您的使用者集區。如需有關如何將使用者集區新增為 SAML 2.0 身分提供者之依賴方或應用程式的資訊，請參閱 SAML 身分提供者的文件。

您還必須向 SAML 身分提供者提供聲明消費者服務 (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](#) 的網域選單中找到使用者集區的網域字首和區域值。

對於某些 SAML 身分提供者，您也需要提供服務提供者 (SP) urn，也稱為對象 URI 或 SP 實體 ID，格式為：

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

您可以在 [Amazon Cognito 主控台](#) 的使用者集區的概觀儀表板中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。這不需要您的應用程式擷取或剖析 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身分提供者接收 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 登入資料。
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. 選擇社交和外部供應商選單。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)。識別符告知 Amazon Cognito 應檢查使用者登入時輸入的電子郵件地址。然後，它會引導他們前往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)。您必須設定 SAML 2.0 身分提供者，將登出回應傳送至設定受管登入時建立的 `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 身分提供者預期有簽署的登出請求，您也需要使用 SAML IdP 設定 Amazon Cognito 提供的簽署憑證。

SAML IdP 會處理已簽署的登出請求，並將您的使用者從 Amazon Cognito 工作階段登出。

- 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

如果您的提供者有公有端點，而不是上傳檔案，建議您輸入中繼資料文件 URL。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。

- 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
```

- 選擇建立。

Amazon Cognito 身分集區入門

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。您的身分集區可以從下列類型的身分驗證服務引入身分：

- 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

使用者向供應商進行身分驗證並對身分集區提供授權後，就會取得臨時 AWS 登入資料。使用者的登入資料具有您為存取其他而定義的許可 AWS 服務。

主題

- [在 Amazon Cognito 中建立身分集區](#)
- [設定 SDK](#)
- [整合身分提供者](#)
- [取得憑證](#)

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。下列程序是一般指南，可在 主控台中建立新的身分集區。您也可以[直接跳到 主控台](#)，並遵循引導式體驗和內嵌說明內容。

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。若要將許可指派給您的 IAM 主體，以便他們可以建立和管理 Amazon Cognito 資源，請參閱 [AWS Amazon Cognito 的 受管政策](#)。此 AmazonCognitoPowerUser 政策足以建立身分集區。
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. 如果您在 中已有要使用的角色 AWS 帳戶，您可以選擇使用現有的 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 連接身分提供者 中，輸入您在 設定身分池信任 中選擇的身分提供者 (IdP) 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。當基本流程使用中時，您可以略過為 IdP 所做的角色選取，並直接呼叫 [AssumeRoleWithWebIdentity](#)。如需詳細資訊，請參閱[身分集區身分驗證流程](#)。

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 身分集區，請設定 適用於 Java 的 AWS SDK、AWS Amplify 或 適用於 .NET 的 SDK。如需詳細資訊，請參閱下列主題。

- 在《適用於 JavaScript 的 AWS SDK 開發人員指南》中 [為 JavaScript 設定 SDK](#)
- 《Amplify 開發人員中心》中的 [Amplify 文件](#)
- 《適用於 .NET 的 SDK 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 [身分集區第三方身分提供者](#)。

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 (未經驗證) 和已驗證並收到字符的使用者提供臨時 AWS 登入資料。透過這些 AWS 登入資料，您的應用程式可以透過 Amazon API Gateway AWS 安全地存取位於 AWS 或外部的後端。請參閱 [取得憑證](#)。

Amazon Cognito 的引導設定選項

您可能想要在結構化的引導式體驗中評估 Amazon Cognito 的功能。以下是一些外部資源，可為使用者集區和身分集區提供量身打造的體驗。

完成研討會

AWS 研討會工作室[會舉辦研討會](#)，引導您完成大部分 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、身分集區和安全組態。

從範例新增應用程式碼

本指南中的[程式碼範例](#)章節具有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具有涵蓋個別操作的簡短程式碼片段，以及各種程式設計語言中 end-to-end 範例應用程式的較長範例。

使用 建立完整堆疊應用程式 AWS Amplify

[AWS Amplify](#) 是 AWS 服務，適用於想要開發和託管應用程式和使用者介面的開發人員。Amazon Cognito 是 Amplify 的身分驗證元件。當您將身分驗證新增至應用程式時，Amplify 可以自動部署 Amazon Cognito 使用者集區和身分集區資源。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](#)。

GitHub 上的更多 Amazon Cognito 應用程式資源

- [使用 .NET for Amazon Cognito 的身分驗證流程範例](#)
- [Amazon Cognito 無密碼驗證](#)
- [具有 Amazon Verified Permissions 的 PetStore 範例](#)
- [使用 ABAC + 身分集區存取 AWS 資源的 React 應用程式範例](#)
- [使用 AWS CDK 進行機器授權的 Amazon Cognito 和 API Gateway](#)
- [使用 Amazon Cognito、API Gateway 和 IAM 建置精細授權](#)
- [CloudFront authorization@edge](#)

更多研討會

- [使用 Amazon Cognito 和 WebAuthn 實作無密碼身分驗證](#)
- [Amazon Cognito 研討會](#)

- [Amazon Cognito 故障診斷研討會](#)
- [使用 Amazon Cognito 和驗證許可進行身分驗證和授權](#)
- [Amazon Cognito JWT Deep Dive](#)

部落格文章

- [使用 Amazon CloudFront 代理保護 Amazon Cognito 的公有用戶端 Amazon CloudFront](#)
- [如何使用 Azure AD 設定 Amazon Cognito 進行聯合身分驗證](#)
- [簡化 Web 應用程式身分驗證：使用 Amazon Cognito 使用者集區聯合 AD FS 的指南](#)

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

Amazon Cognito 的實作是 AWS Management Console 或 AWS SDK 管理工具和應用程式中 SDK 程式庫的混合。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。

您可以使用 Amazon Cognito 使用者集區建立的最低努力整合是使用[受管登入](#)。受管登入是 ready-to-use Web 型登入應用程式，用於快速測試和部署 Amazon Cognito 使用者集區。使用受管登入的使用者集區身分驗證需要引導使用者前往託管登入頁面的 OpenID Connect (OIDC) 程式庫。在這一系列的使用者互動和重新導向 Web 端點中，Amazon Cognito 會處理身分驗證流程，包括第三方登入、多重要素驗證 (MFA)，並選擇身分驗證流程。您的應用程式只需要處理 Amazon Cognito 在回應中傳回的身分驗證結果。

您也可以將 AWS SDK 新增至應用程式、自訂建置身分驗證介面，以及叫用 API 操作來驗證和授權使用者。[AWS Amplify](#) 是 AWS 服務用於建置完整堆疊應用程式的，後端為 Amazon Cognito 身分驗證。

例如，您的應用程式可能會叫用受管登入以進行使用者登入，然後從您的應用程式碼呼叫權杖端點，以交換使用者的權杖授權碼。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。身分集區沒有與使用者集區相同的受管身分驗證選項，若要存取應用程式中的 AWS 登入資料，請在匯入的 SDK 模組中實作身分集區操作。若要開始使用您自己的自訂應用程式程式碼，請造訪[AWS SDKs](#) 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。如需可用的平台 AWS SDKs，請參閱[使用 AWS SDK 進行身分驗證](#)。AWS CLI 是適用於 Amazon Cognito 和其他的命令列開發套件 AWS 服務，是開始熟悉 Amazon Cognito API 操作及其語法的寶貴位置。

Note

有些 Amazon Cognito 元件只能使用 API 進行設定。例如，您只能透過使用請求來更新 [CreateUserPool](#) 或 [UpdateUserPool](#) API 請求中 [UserPool](#) 類別的 LambdaConfig 屬性，藉此設定使用者集區[自訂簡訊或電子郵件寄件者](#) 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。最終類別的 API 操作會執行您在機密伺服器端用戶端使用自己的 AWS 登入資料授權的使用者操作。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[了解 API、OIDC 和受管登入頁面身分驗證](#)。

主題

- [使用 進行身分驗證 AWS Amplify](#)
- [使用 AWS SDK 進行身分驗證](#)
- [身分驗證如何與 Amazon Cognito 搭配使用](#)
- [搭配 AWS SDK 使用此服務](#)
- [使用 Amazon Verified Permissions 進行授權](#)

使用 進行身分驗證 AWS Amplify

AWS Amplify 是建置 Web 和行動應用程式的完整解決方案。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。

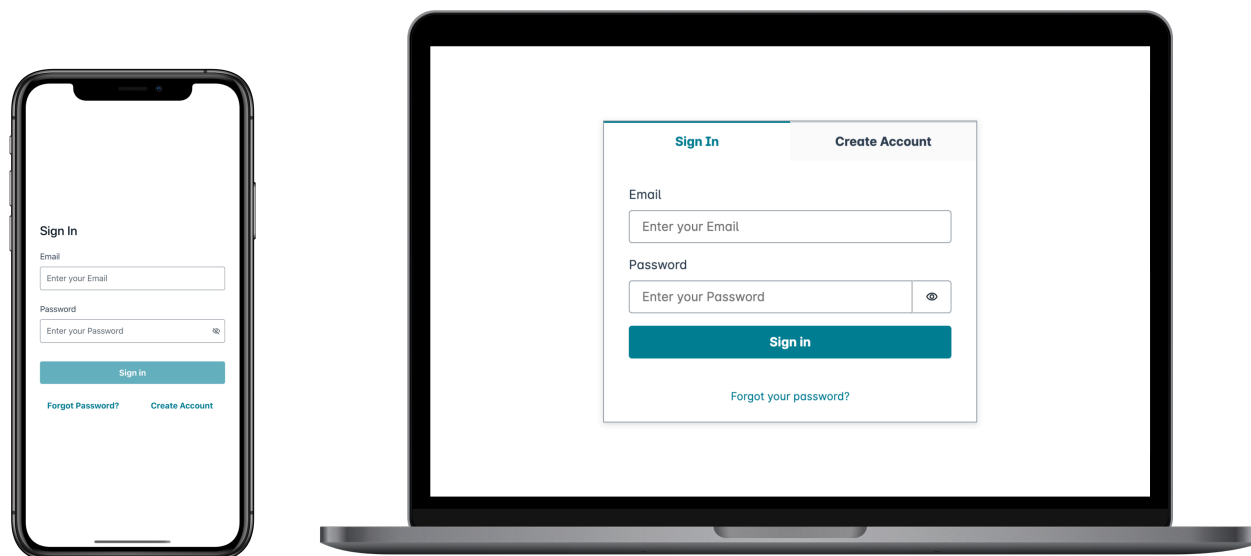
- [React 的 Amplify 身分驗證](#)
- [React Native 的 Amplify 身分驗證](#)
- [Swift \(iOS\) 的 Amplify 身分驗證](#)
- [Android 的 Amplify 身分驗證](#)
- [Flutter 的 Amplify 身分驗證](#)

Amplify 程式庫是開放原始碼，可在 [GitHub](#) 上取得。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)

[使用者集區受管登入](#) 可以滿足 Web 或行動應用程式身分驗證前端的基本需求。若要在受管登入可容納的參數之外自訂您的使用者介面 (UI)，請自訂建置應用程式。[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

若要使用安全後端建置與 Amazon Cognito 互動的自有身分微服務，請使用您選擇的語言 AWS SDK 連線至 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 適用於 .NET 的 SDK](#)
- [AWS 適用於 C++ 的 SDK](#)
- [AWS 適用於 Go 的 SDK](#)
- [AWS 適用於 Java V2 的 SDK](#)
- [AWS 適用於 JavaScript 的 SDK](#)
- [AWS 適用於 PHP V3 的 SDK](#)
- [AWS 適用於 Python 的 SDK](#)
- [AWS 適用於 Ruby V3 的 SDK](#)

身分驗證如何與 Amazon Cognito 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 Web 字符 JWTs)。

當您的客戶使用使用者集區字符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 登入資料。

透過使用者集區登入，您可以使用 AWS SDK 完全實作身分驗證和授權。如果您不想建置自己的使用者介面 (UI) 元件，您可以叫用第三方身分提供者 (IdP) 的預先建置 Web UI（受管登入）或登入頁面。

本主題概述您的應用程式可以與 Amazon Cognito 互動的一些方式，以使用 ID 字符進行身分驗證、使用存取字符進行授權，以及 AWS 服務使用身分集區憑證進行存取。

主題

- [使用受管登入進行使用者集區身分驗證](#)
- [使用 AWS SDK 的使用者集區 API 身分驗證和授權](#)
- [使用第三方身分提供者進行使用者集區身分驗證](#)
- [身分集區身分驗證](#)

使用受管登入進行使用者集區身分驗證

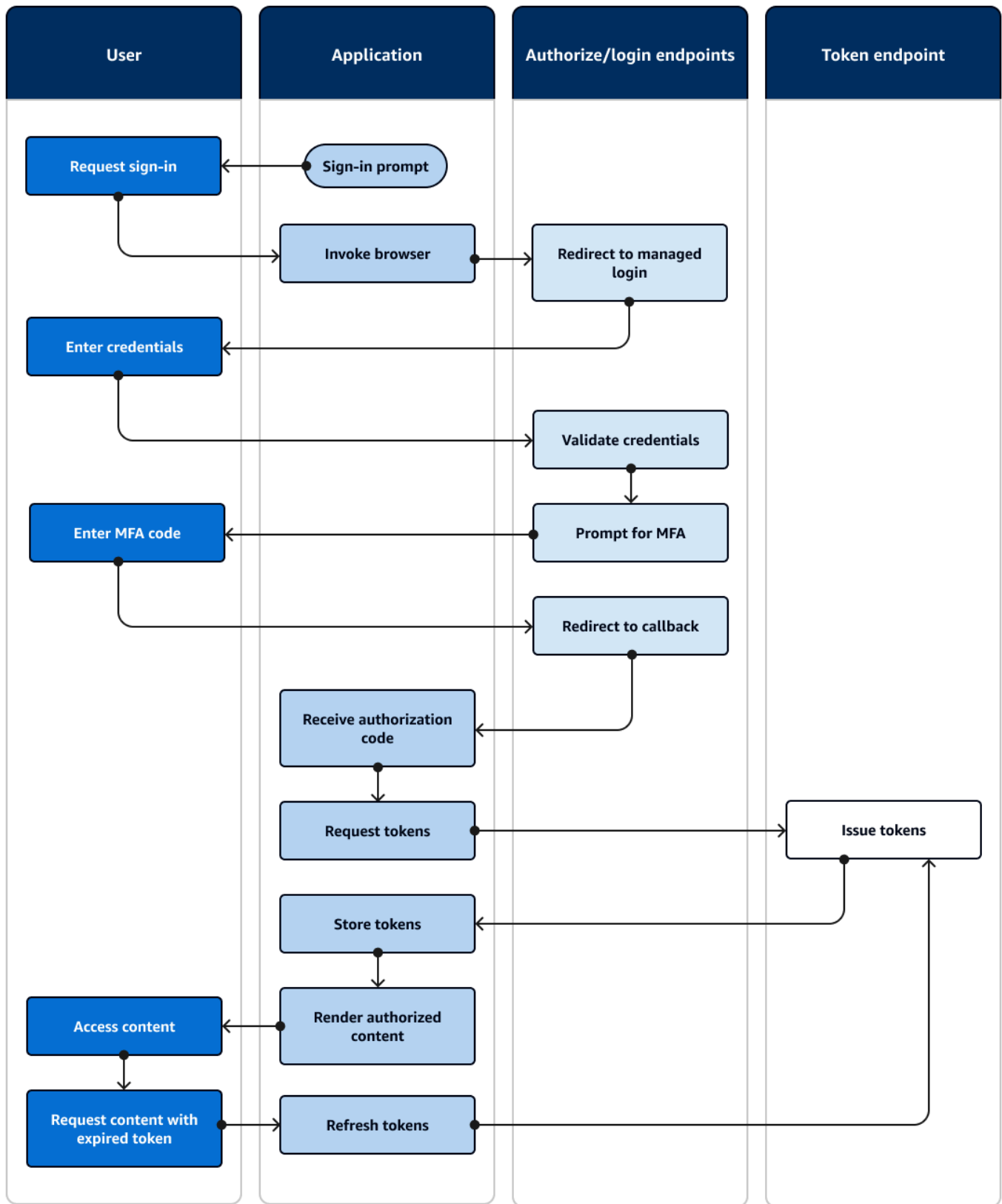
[受管登入](#)是連結至使用者集區和應用程式用戶端的網站。它可以為您的使用者執行登入、註冊和密碼重設操作。具有受管登入元件以進行身分驗證的應用程式可能需要較少的開發人員實作工作。應用程式可以略過 UI 元件進行身分驗證，並在使用者的瀏覽器中叫用受管登入網頁。

應用程式會使用 Web 或應用程式重新導向位置收集使用者的 JWTs。實作受管登入的應用程式可以連線到使用者集區進行身分驗證，就像是 OpenID Connect (OIDC) IdP。

受管登入適合應用程式需要 OIDC 授權伺服器的身分驗證服務，但不立即需要自訂身分驗證、身分集區整合或使用屬性自助服務等功能。當您想要使用這些進階選項時，您可以使用 SDK 的使用者集區元件來實作這些選項。

受管登入和第三方 IdP 身分驗證模型主要依賴 OIDC 實作，最適合具有 OAuth 2.0 範圍的進階授權模型。

下圖說明受管登入身分驗證的典型登入工作階段。



受管登入身分驗證流程

1. 使用者存取您的應用程式。
2. 他們選取「登入」連結。
3. 應用程式會將使用者導向至使用者集區網域的受管登入頁面中的登入提示。
4. 他們輸入使用者名稱和密碼。
5. 使用者集區會驗證使用者的登入資料，並判斷使用者已啟用多重要素驗證 (MFA)。
6. 受管登入頁面會提示使用者輸入 MFA 代碼。
7. 使用者輸入其 MFA 代碼。
8. 您的使用者集區會將使用者重新導向至應用程式 URL。
9. 應用程式會從受管登入附加至[回呼 URL](#)的 URL 請求參數收集授權碼。
10. 應用程式會請求具有授權碼的字符。
11. 字符端點會將 JWTs 傳回至應用程式。
12. 應用程式會解碼、驗證和存放或快取使用者的 JWTs。
13. 應用程式會顯示請求的存取控制元件。
14. 使用者檢視其內容。
15. 稍後，使用者的存取權杖已過期，他們請求檢視受存取控制的元件。
16. 應用程式會判斷使用者的工作階段應持續存在。它會使用重新整理權杖從權杖端點請求新權杖。

變體和自訂

您可以使用整個使用者集區的[品牌編輯器](#)，或在任何[應用程式用戶端](#)層級自訂受管登入頁面的外觀和風格。您也可以使用自己的身分提供者、範圍、使用者屬性的存取和進階安全組態來[設定應用程式用戶端](#)。

相關資源

- [使用者集區受管登入](#)
- [範圍、M2M 和具有資源伺服器的 APIs](#)
- [使用者集區端點和受管登入參考](#)

使用 AWS SDK 的使用者集區 API 身分驗證和授權

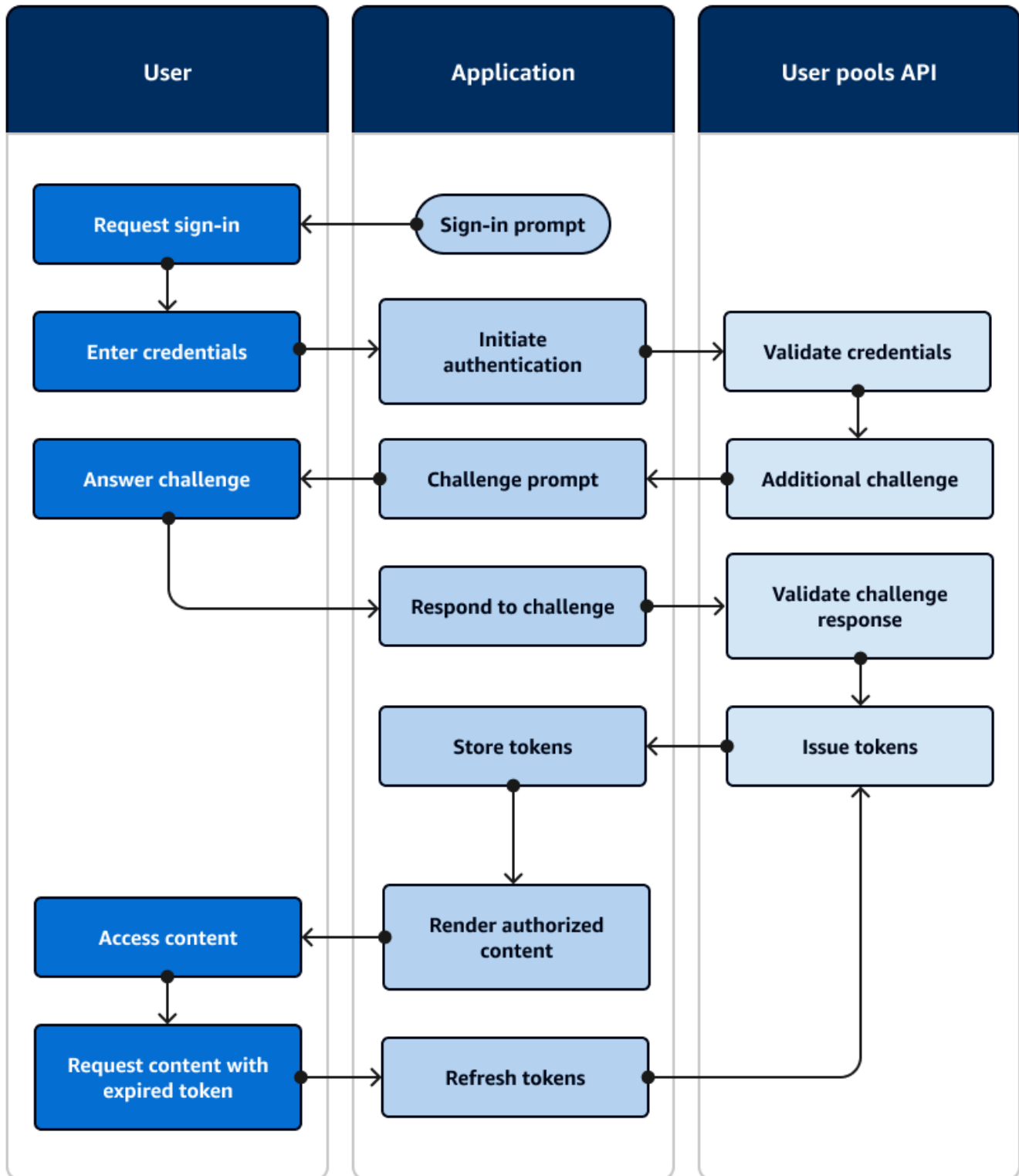
AWS 已在[各種開發人員架構](#)中為 Amazon Cognito 使用者集區或 Amazon Cognito 身分提供者開發元件。這些 SDKs 內建的方法會呼叫 [Amazon Cognito 使用者集區 API](#)。相同的使用者集區 API 命名空間具有設定使用者集區和使用者的身分驗證的操作。如需更徹底的概觀，請參閱 [了解 API、OIDC 和受管登入頁面身分驗證](#)。

API 身分驗證適用於您的應用程式具有現有 UI 元件的模型，主要依賴使用者集區做為使用者目錄。此設計會將 Amazon Cognito 新增為較大應用程式中的元件。它需要程式設計邏輯來處理複雜的挑戰和回應鏈。

此應用程式不需要實作完整的 OpenID Connect (OIDC) 依賴方實作。反之，它可以解碼和使用 JWTs。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 開發套件建置身分驗證。

具有自訂 OAuth 範圍的 API 身分驗證較不傾向於外部 API 授權。若要從 API 身分驗證將自訂範圍新增至存取字符，請使用在執行時間修改字符[產生權杖前 Lambda 觸發程序](#)。

下圖說明 API 身分驗證的典型登入工作階段。



API 身分驗證流程

1. 使用者存取您的應用程式。
2. 他們選取「登入」連結。
3. 他們輸入使用者名稱和密碼。
4. 應用程式會叫用發出 [InitiateAuth](#) API 請求的方法。請求會將使用者的登入資料傳遞給使用者集區。
5. 使用者集區會驗證使用者的登入資料，並判斷使用者已啟用多重要素驗證 (MFA)。
6. 使用者集區會回應請求 MFA 程式碼的挑戰。
7. 應用程式會產生提示，從使用者收集 MFA 代碼。
8. 應用程式會叫用發出 [RespondToAuthChallenge](#) API 請求的方法。請求會傳遞使用者的 MFA 代碼。
9. 使用者集區會驗證使用者的 MFA 代碼。
10. 使用者集區會以使用者的 JWTs 回應。
11. 應用程式會解碼、驗證和存放或快取使用者的 JWTs。
12. 應用程式會顯示請求的存取控制元件。
13. 使用者檢視其內容。
14. 稍後，使用者的存取權杖已過期，他們請求檢視受存取控制的元件。
15. 應用程式會判斷使用者的工作階段應持續存在。它會再次使用重新整理權杖叫用 [InitiateAuth](#) 方法，並擷取新的權杖。

變體和自訂

您可以使用其他挑戰來增強此流程，例如，您自己的自訂身分驗證挑戰。您可以自動限制密碼已洩露或非預期登入特性可能表示嘗試惡意登入的使用者存取。此流程看起來與註冊、更新使用者屬性和重設密碼的操作大致相同。這些流程大部分都有重複的公有（用戶端）和機密（伺服器端）API 操作。

相關資源

- [Amazon Cognito 使用者集區 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](#)
- [了解 API、OIDC 和受管登入頁面身分驗證](#)

使用第三方身分提供者進行使用者集區身分驗證

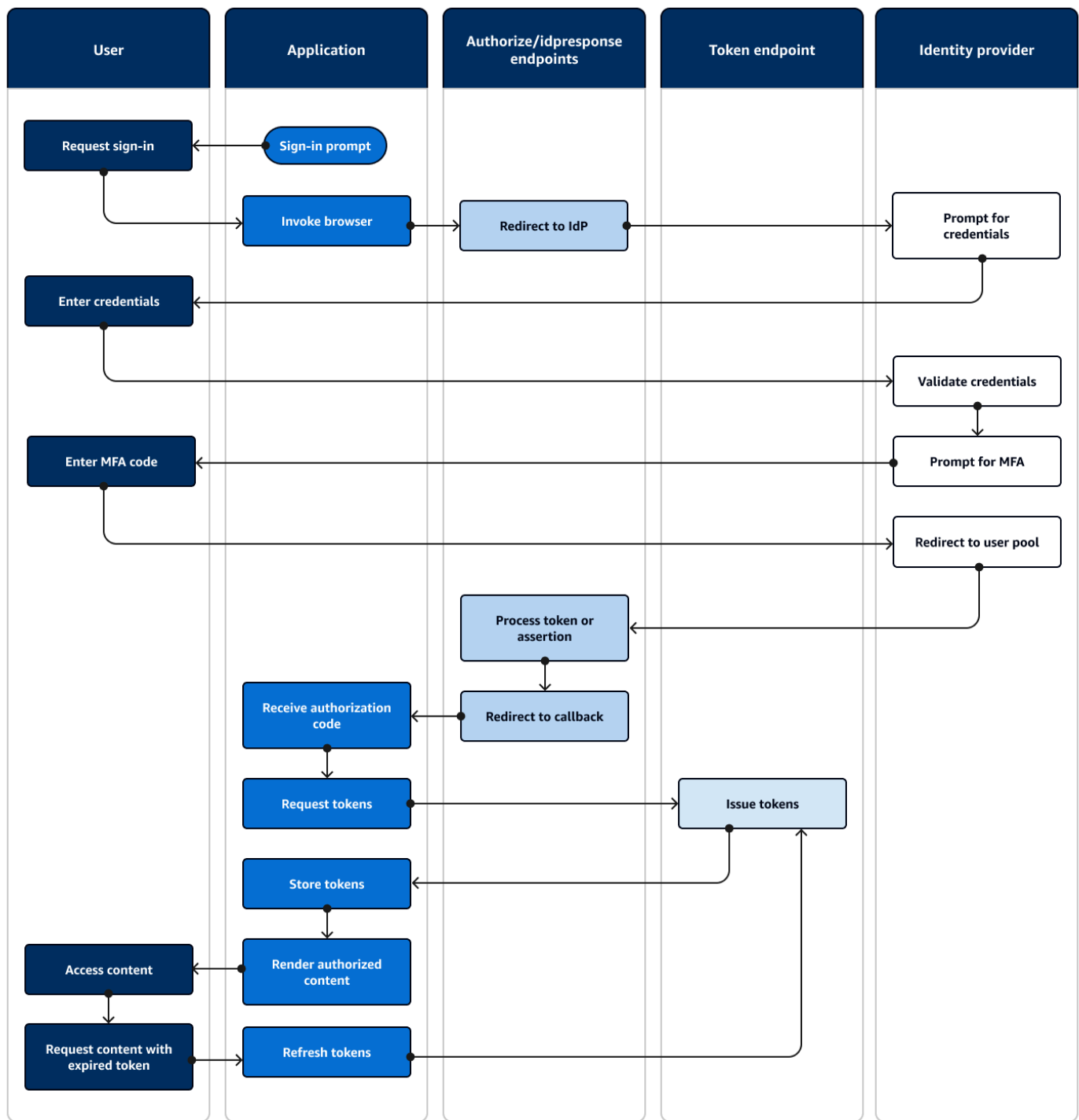
使用外部身分提供者 (IdP) 或聯合身分驗證登入是與[受管登入](#)類似的模型。您的應用程式是使用者集區的 OIDC 依賴方，而您的使用者集區可做為 IdP 的傳遞。IdP 可以是消費者使用者目錄，例如 Facebook 或 Google，也可以是 SAML 2.0 或 OIDC 企業目錄，例如 Azure。

您的應用程式不會在使用者的瀏覽器中受管登入，而是在使用者集區[授權伺服器](#)上叫用重新導向端點。從使用者的檢視中，他們會選擇應用程式中的登入按鈕。然後，其 IdP 會提示他們登入。如同受管登入身分驗證，應用程式會在應用程式中的重新導向位置收集 JWTs。

使用第三方 IdP 進行身分驗證，適合使用者在註冊應用程式時可能不想取得新密碼的模型。您可以輕鬆將第三方身分驗證新增至實作受管登入身分驗證的應用程式。實際上，受管登入和第三方 IdPs 會從您在使用者瀏覽器中叫用內容的次要變化產生一致的身分驗證結果。

與受管登入身分驗證一樣，聯合身分驗證最適合具有 OAuth 2.0 範圍的進階授權模型。

下圖說明聯合身分驗證的典型登入工作階段。



聯合身分驗證流程

1. 使用者存取您的應用程式。
2. 他們選取「登入」連結。

3. 應用程式會引導使用者使用其 IdP 登入提示。
4. 他們輸入使用者名稱和密碼。
5. IdP 會驗證使用者的登入資料，並判斷使用者已啟用多重要素驗證 (MFA)。
6. IdP 會提示使用者輸入 MFA 代碼。
7. 使用者輸入其 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. 如果使用者通過授權碼，則使用者集區會無提示地交換 IdP 字符的程式碼。使用者集區會驗證 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. 應用程式會從使用者集區附加至[回呼 URL](#) 的 URL 請求參數收集授權碼。
11. 應用程式會請求具有授權碼的字符。
12. 字符端點會將 JWTs 傳回應用程式。
13. 應用程式會解碼、驗證和存放或快取使用者的 JWTs。
14. 應用程式會顯示請求的存取控制元件。
15. 使用者檢視其內容。
16. 稍後，使用者的存取權杖已過期，他們請求檢視受存取控制的元件。
17. 應用程式會判斷使用者的工作階段應持續存在。它會使用重新整理權杖從權杖端點請求新權杖。

變體和自訂

您可以在[受管登入](#)中啟動聯合身分驗證，使用者可以從指派給[應用程式用戶端](#)的 IdPs 清單中選擇。受管登入也可以提示輸入電子郵件地址，並自動將使用者的請求路由到對應的 SAML IdP。使用第三方身分提供者進行身分驗證不需要使用者與受管登入互動。您的應用程式可以將請求參數新增至使用者的[授權伺服器請求](#)，並讓使用者以無提示方式重新導向至其 IdP 登入頁面。

相關資源

- [使用第三方身分提供者登入使用者集區](#)
- [範圍、M2M 和具有資源伺服器的 APIs](#)
- [使用者集區端點和受管登入參考](#)

身分集區身分驗證

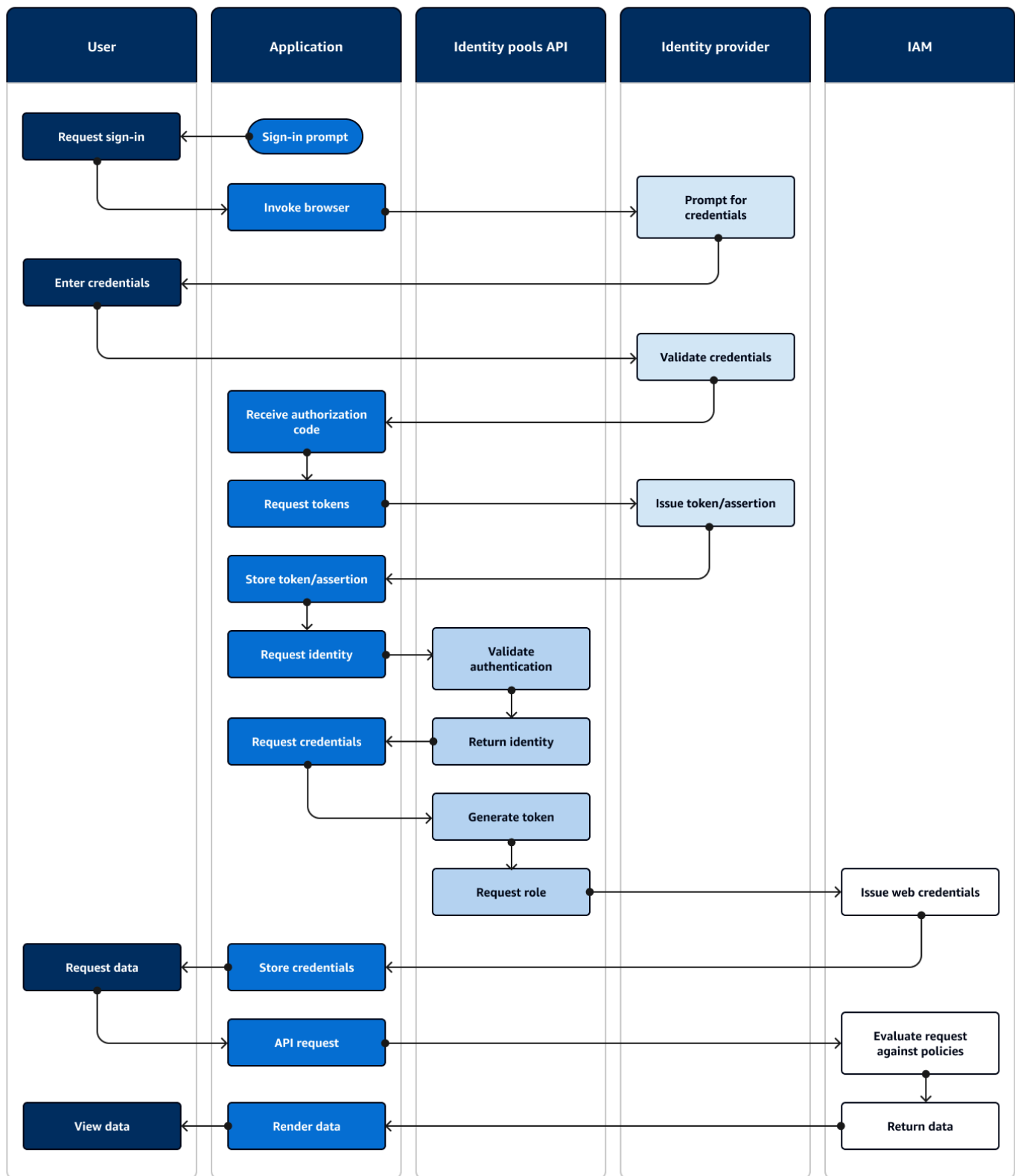
身分集區是應用程式元件，與函數、API 命名空間和 SDK 模型中的使用者集區不同。當使用者集區提供字符型身分驗證和授權時，身分集區會提供 AWS Identity and Access Management (IAM) 的授權。

您可以將一組 IdPs 指派給身分集區，並使用它們登入使用者。使用者集區緊密整合為身分集區 IdPs 並為身分集區提供存取控制的最多選項。同時，身分集區的身分驗證選項種類繁多。使用者集區會加入 SAML、OIDC、社交、開發人員和訪客身分來源，做為從身分集區暫時 AWS 登入資料的路由。

使用身分集區的身分驗證是外部的，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身分驗證之後，它會將驗證傳遞至身分集區，並接收臨時工作階段作為回報。

使用身分集區的身分驗證適合一種模型，您可以在其中 AWS 服務 使用 IAM 授權強制執行 中應用程式資產和資料的存取控制。與[使用者集區中的 API 身分驗證](#)一樣，成功的應用程式會針對您要存取的每個服務包含 AWS SDKs，以保障使用者的利益。AWS SDKs 會將身分集區身分驗證的登入資料套用為 API 請求的簽章。

下圖說明使用 IdP 進行身分集區身分驗證的典型登入工作階段。



身分集區身分驗證流程

1. 使用者存取您的應用程式。
2. 他們選取「登入」連結。
3. 應用程式會引導使用者使用其 IdP 登入提示。
4. 他們輸入使用者名稱和密碼。
5. IdP 會驗證使用者的登入資料。
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. 如果使用者傳遞授權碼，應用程式會交換 IdP 字符的程式碼。
8. 應用程式會解碼、驗證和存放或快取使用者的 JWTs 或聲明。
9. 應用程式會叫用提出 [GetId](#) API 請求的方法。它會傳遞使用者的字符或聲明，並請求身分 ID。
10. 身分集區會針對設定的身分提供者驗證權杖或聲明。
11. 身分集區會傳回身分 ID。
12. 應用程式會叫用發出 [GetCredentialsForIdentity](#) API 請求的方法。它會傳遞使用者的字符或聲明，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 角色的宣告。身分集區會根據使用者的請求和 IdP 身分集區組態中的角色選擇條件來決定角色。
14. AWS Security Token Service (AWS STS) 會從身分集區回應 [AssumeRoleWithWebIdentity](#) 請求。回應包含具有 IAM 角色之暫時工作階段的 API 登入資料。
15. 應用程式會存放工作階段登入資料。
16. 使用者在應用程式中採取動作，需要中受存取保護的資源 AWS。
17. 應用程式會將暫時登入資料做為 [簽章](#) 套用至所需的 API 請求 AWS 服務。
18. IAM 會評估連接到登入資料中角色的政策。它將它們與請求進行比較。
19. 會 AWS 服務 傳回請求的資料。
20. 應用程式會在使用者的界面中轉譯資料。
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身分驗證，請在問題字符/聲明步驟之後插入先前的其中一個使用者集區概觀。開發人員身分驗證會以 [開發人員登入](#) 資料簽署的請求取代請求身分之前的所有步驟。訪客身分驗證也會直接跳至請求身分、不會驗證身分驗證，並傳回 [有限存取](#) IAM 角色的登入資料。

相關資源

- [Amazon Cognito 身分集區](#)
- [使用者 IAM 角色](#)
- [身分集區身分驗證流程](#)

搭配 AWS SDK 使用此服務

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

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

SDK 文件	代碼範例
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。

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。

您的應用程式可以在 [IsAuthorizedWithToken](#) 或 [BatchIsAuthorizedWithToken](#) API 請求中提供使用者的身分或存取權杖給已驗證的許可。這些 API 操作接受您的使用者做為 Resource，Principal 並對他們想要存取 Action 的 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. 權杖中的 token_use 宣告值與您傳遞給的參數相符 IsAuthorizedWithToken。access 如果您將其傳遞至 accessToken 參數，且 id 將其傳遞至 identityToken 參數，則 token_use 宣告必須是。
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。如果 Verified Permissions 符合允許的動作且不符合明確 Deny 動作，Allow 則會使用 回應您的請求；否則，它會使用 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 Verified Permissions 的使用者宣告，請使用 自訂存取和身分字符中的內容[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。

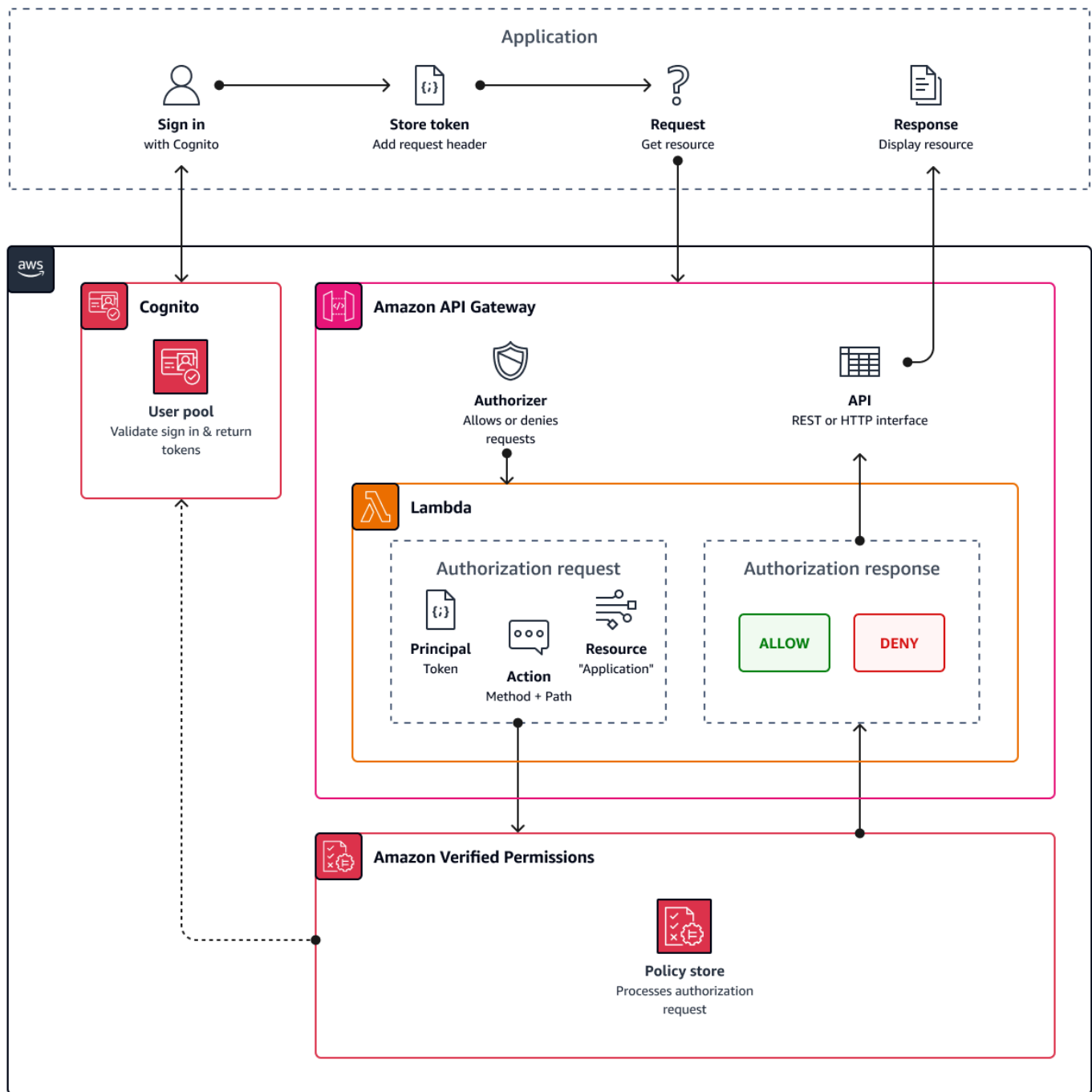
其他資源

- [將 Amazon Cognito 權杖映射至已驗證許可結構描述](#)
- [使用 Amazon Verified Permissions 和 Amazon Cognito 授權 API Gateway APIs](#)
- [研討會：使用 Amazon Cognito 和驗證許可進行身分驗證和授權](#)

具有已驗證許可的 API 授權

您的 ID 或存取權杖可以授權請求使用 Verified Permissions 後端 Amazon API Gateway REST APIs。您可以使用使用者集區和 API 的立即連結來建立 [政策存放區](#)。透過 [使用 API Gateway 設定和身分來源](#) 啟動選項，Verified Permissions 會將使用者集區身分來源新增至政策存放區，並將 Lambda 授權方新增至 API。當您的應用程式將使用者集區承載符記傳遞至 API 時，Lambda 授權方會叫用驗證許可。授權方以委託人身分傳遞字符，並以請求路徑和方法做為動作。

下圖說明具有 Verified Permissions 的 API Gateway API 的授權流程。如需詳細資訊，請參閱《Amazon Verified Permissions 使用者指南》中的 [API 連結政策存放區](#)。



Verified Permissions 會圍繞[使用者集區群組](#)建立 API 授權。由於 ID 和存取權杖都包含 `cognito:groups` 宣告，因此您的政策存放區可以在各種應用程式內容中管理 APIs 的角色型存取控制 (RBAC)。

選擇政策存放區設定

在政策存放區上設定身分來源時，您必須選擇是否要處理存取或 ID 字符。此決策對政策引擎的運作方式很重要。ID 字符包含使用者屬性。存取權杖包含使用者存取控制資訊：[OAuth 範圍](#)。雖然這兩種字符類型都有群組成員資訊，但我們通常建議使用 Verified Permissions 政策存放區的 RBAC 存取字符。存取權杖會新增至群組成員資格，其範圍有助於授權決策。存取字符中的宣告會成為授權請求中的[內容](#)。

當您將使用者集區設定為身分來源時，也必須設定使用者和群組實體類型。實體類型是您可以在 Verified Permissions 政策中參考的主體、動作和資源識別符。政策存放區中的實體可以有成員關係，其中一個實體可以是父實體的成員。使用 成員資格，您可以參考委託人群組、動作群組和資源群組。如果是使用者集區群組，您指定的使用者實體類型必須是群組實體類型的成員。當您設定 [API 連結政策存放區](#)或遵循 Verified Permissions 主控台內的引導設定時，您的政策存放區會自動具有此父成員關係。

ID 字符可以結合 RBAC 與屬性型存取控制 (ABAC)。建立 [API 連結政策存放區](#)之後，您可以使用[使用者屬性](#)和群組成員資格來增強政策。ID 字符中的屬性宣告會成為授權請求中的[主體屬性](#)。您的政策可以根據委託人屬性進行授權決策。

您也可以設定政策存放區，以接受權杖，其中包含符合您所提供可接受應用程式用戶端清單的 aud 或 client_id 宣告。

角色型 API 授權的範例政策

下列範例政策是透過為 [PetStore](#) 範例 REST API 設定 Verified Permissions 政策存放區所建立。

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

Verified Permissions 會在下列情況下，從您的應用程式將 Allow 決策傳回給授權請求：

1. 您的應用程式在 Authorization 標頭中傳遞 ID 或存取權杖做為承載權杖。
2. 您的應用程式已透過包含字串的 cognito:groups 宣告傳遞權杖 MyGroup。
3. 例如，您的應用程式向 <https://myapi.example.com/pets> 或提出 HTTP GET 請求 <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 請求以外的條件下產生驗證許可的授權請求。您可以將應用程式中的任何存取控制決策提交到您的政策存放區。例如，您可以在任何請求傳輸網路之前，使用屬性型存取控制來補充 Amazon DynamoDB 或 Amazon S3 安全性，以減少配額使用。

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
);
```

以下是使用者集區中 ID `us-east-1_Example` 為 且 ID 為 或使用者 ID 的使用者委託人範例 `973db890-092c-49e4-a9d0-912a4c0a20c7`。

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

當您想要在 Verified Permissions 政策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
);
```

屬性型存取控制

Verified Permissions 為您的應用程式的授權，以及 Amazon Cognito 身分集區登入 AWS 資料的[存取控制功能屬性](#)，都是以屬性為基礎的存取控制 (ABAC) 形式。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	流程	結果
Amazon Verified Permissions	Returns an <code>##</code> or <code>###</code> decision from analysis of a user pool JWT.	Access to application resources succeeds or fails based on Cedar policy evaluation.
Amazon Cognito identity pools (attributes for access control)	Assigns 工作階段標籤 to your user based on their attributes. IAM policy conditions can check tags <code>##</code> or <code>###</code> user access to AWS 服務.	A tagged session with temporary AWS credentials for an IAM role.

Amazon Cognito AWS SDKs的程式碼範例

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 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

- [使用 AWS SDKs的 Amazon Cognito 身分程式碼範例](#)
 - [使用 AWS SDKs的 Amazon Cognito 身分基本範例](#)
 - [使用 AWS SDKs的 Amazon Cognito 身分動作](#)
 - [CreateIdentityPool 搭配 AWS SDK 或 CLI 使用](#)
 - [DeleteIdentityPool 搭配 AWS SDK 或 CLI 使用](#)
 - [搭配使用 DescribeIdentityPool 與 CLI](#)
 - [GetCredentialsForIdentity 搭配 AWS SDK 使用](#)
 - [搭配使用 GetIdentityPoolRoles 與 CLI](#)
 - [ListIdentityPools 搭配 AWS SDK 或 CLI 使用](#)
 - [搭配使用 SetIdentityPoolRoles 與 CLI](#)
 - [搭配使用 UpdateIdentityPool 與 CLI](#)
 - [使用 AWS SDKs Amazon Cognito 身分案例](#)
 - [建立 Amazon Textract Explorer 應用程式](#)
- [使用 AWS SDKs的 Amazon Cognito 身分提供者程式碼範例](#)
 - [使用 AWS SDKs的 Amazon Cognito 身分提供者基本範例](#)
 - [Hello Amazon Cognito](#)
 - [使用 AWS SDKs的 Amazon Cognito 身分提供者動作](#)
 - [AdminCreateUser 搭配 AWS SDK 或 CLI 使用](#)
 - [AdminGetUser 搭配 AWS SDK 或 CLI 使用](#)
 - [AdminInitiateAuth 搭配 AWS SDK 或 CLI 使用](#)
 - [AdminRespondToAuthChallenge 搭配 AWS SDK 或 CLI 使用](#)
 - [AdminSetUserPassword 搭配 AWS SDK 或 CLI 使用](#)
 - [AssociateSoftwareToken 搭配 AWS SDK 或 CLI 使用](#)
 - [ConfirmDevice 搭配 AWS SDK 或 CLI 使用](#)

- [ConfirmForgotPassword 搭配 AWS SDK 或 CLI 使用](#)
- [ConfirmSignUp 搭配 AWS SDK 或 CLI 使用](#)
- [CreateUserPool 搭配 AWS SDK 或 CLI 使用](#)
- [CreateUserPoolClient 搭配 AWS SDK 或 CLI 使用](#)
- [DeleteUser 搭配 AWS SDK 或 CLI 使用](#)
- [ForgotPassword 搭配 AWS SDK 或 CLI 使用](#)
- [InitiateAuth 搭配 AWS SDK 或 CLI 使用](#)
- [ListUserPools 搭配 AWS SDK 或 CLI 使用](#)
- [ListUsers 搭配 AWS SDK 或 CLI 使用](#)
- [ResendConfirmationCode 搭配 AWS SDK 或 CLI 使用](#)
- [RespondToAuthChallenge 搭配 AWS SDK 或 CLI 使用](#)
- [SignUp 搭配 AWS SDK 或 CLI 使用](#)
- [UpdateUserPool 搭配 AWS SDK 或 CLI 使用](#)
- [VerifySoftwareToken 搭配 AWS SDK 或 CLI 使用](#)
- [使用 AWS SDKs Amazon Cognito 身分提供者案例](#)
 - [使用 SDK 透過 Lambda 函數自動確認已知的 Amazon Cognito 使用者 AWS](#)
 - [使用 SDK 透過 Lambda 函數自動遷移已知的 Amazon Cognito 使用者 AWS](#)
 - [使用 SDK 向需要 MFA 的 Amazon Cognito 使用者集區註冊使用者 AWS](#)
 - [使用 AWS SDK 進行 Amazon Cognito 使用者身分驗證後，使用 Lambda 函數寫入自訂活動資料](#)
- [使用 SDK 的 Amazon Cognito Sync 程式碼範例 AWS SDKs](#)
 - [使用 AWS SDKs 的 Amazon Cognito Sync 基本範例](#)
 - [使用 AWS SDKs 的 Amazon Cognito Sync 動作](#)
 - [ListIdentityPoolUsage 搭配 AWS SDK 使用](#)

使用 AWS SDKs 的 Amazon Cognito 身分程式碼範例

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

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

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

- [使用 AWS SDKs 的 Amazon Cognito 身分基本範例](#)
 - [使用 AWS SDKs 的 Amazon Cognito 身分動作](#)
 - [CreateIdentityPool 搭配 AWS SDK 或 CLI 使用](#)
 - [DeleteIdentityPool 搭配 AWS SDK 或 CLI 使用](#)
 - [搭配使用 DescribeIdentityPool 與 CLI](#)
 - [GetCredentialsForIdentity 搭配 AWS SDK 使用](#)
 - [搭配使用 GetIdentityPoolRoles 與 CLI](#)
 - [ListIdentityPools 搭配 AWS SDK 或 CLI 使用](#)
 - [搭配使用 SetIdentityPoolRoles 與 CLI](#)
 - [搭配使用 UpdateIdentityPool 與 CLI](#)
 - [使用 AWS SDKs Amazon Cognito 身分案例](#)
 - [建立 Amazon Textract Explorer 應用程式](#)

使用 AWS SDKs 的 Amazon Cognito 身分基本範例

下列程式碼範例示範如何搭配 AWS SDKs 使用 Amazon Cognito Identity 的基本概念。

範例

- [使用 AWS SDKs 的 Amazon Cognito 身分動作](#)
 - [CreateIdentityPool 搭配 AWS SDK 或 CLI 使用](#)
 - [DeleteIdentityPool 搭配 AWS SDK 或 CLI 使用](#)
 - [搭配使用 DescribeIdentityPool 與 CLI](#)
 - [GetCredentialsForIdentity 搭配 AWS SDK 使用](#)
 - [搭配使用 GetIdentityPoolRoles 與 CLI](#)
 - [ListIdentityPools 搭配 AWS SDK 或 CLI 使用](#)
 - [搭配使用 SetIdentityPoolRoles 與 CLI](#)
 - [搭配使用 UpdateIdentityPool 與 CLI](#)

使用 AWS SDKs 的 Amazon Cognito 身分動作

下列程式碼範例示範如何使用 AWS SDKs 執行個別 Amazon Cognito Identity 動作。每個範例均包含 GitHub 的連結，您可以在連結中找到設定和執行程式碼的相關說明。

這些摘錄會呼叫 Amazon Cognito Identity API，是必須在內容中執行之大型程式的程式碼摘錄。您可以在 [使用 AWS SDKs Amazon Cognito 身分案例](#) 中查看內容中的動作。

下列範例僅包含最常使用的動作。如需完整清單，請參閱 [Amazon Cognito 身分 API 參考](#)。

範例

- [CreateIdentityPool 搭配 AWS SDK 或 CLI 使用](#)
- [DeleteIdentityPool 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 DescribeIdentityPool 與 CLI](#)
- [GetCredentialsForIdentity 搭配 AWS SDK 使用](#)
- [搭配使用 GetIdentityPoolRoles 與 CLI](#)
- [ListIdentityPools 搭配 AWS SDK 或 CLI 使用](#)
- [搭配使用 SetIdentityPoolRoles 與 CLI](#)
- [搭配使用 UpdateIdentityPool 與 CLI](#)

CreateIdentityPool 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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 MyIdentityPool 的身分集區。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_aaaaaaaa",ClientId="3n4b5urk1ft4f13mg5e62d9ado",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》中的 [CreateIdentityPool](#)。

Java

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.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.CognitoIdentityProviderException;

/**
 * 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》中的 [CreateIdentityPool](#)。

PowerShell

PowerShell V4 的工具

範例 1：建立新的身分集區，允許未驗證的身分。

```
New-CGIIIdentityPool -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 Cmdlet Reference (V4) 中的 [CreateIdentityPool](#)。

PowerShell V5 的工具

範例 1：建立新的身分集區，允許未驗證的身分。

```
New-CGIIIdentityPool -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 Cmdlet Reference (V5) 中的 [CreateIdentityPool](#)。

Swift

SDK for Swift

Note

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

```
import AWSCognitoIdentity

/// 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? {
    do {
        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
    } catch {
        print("ERROR: createIdentityPool:", dump(error))
        throw error
    }
}

```

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

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。

DeleteIdentityPool 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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》中的 [DeleteIdentityPool](#)。

Java

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.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 identityPoolId = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

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

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

            DeleteIdentityPoolRequest identityPoolRequest =
DeleteIdentityPoolRequest.builder()
                .identityPoolId(identityPoolId)
                .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》中的 [DeleteIdentityPool](#)。

PowerShell

PowerShell V4 的工具

範例 1：刪除特定身分集區。

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V4) 中的 [DeleteIdentityPool](#)。

PowerShell V5 的工具

範例 1：刪除特定身分集區。

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

- 如需 API 詳細資訊，請參閱 AWS Tools for PowerShell Cmdlet Reference (V5) 中的 [DeleteIdentityPool](#)。

Swift

SDK for Swift

Note

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

```
import AWSCognitoIdentity

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

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

```
    } catch {
        print("ERROR: deleteIdentityPool:", dump(error))
        throw error
    }
}
```

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

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。

搭配使用 **DescribeIdentityPool** 與 CLI

下列程式碼範例示範如何使用 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_11111111",
      "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
```

```
        "ServerSideTokenCheck": false
    }
  ]
}
```

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

PowerShell

PowerShell V4 的工具

範例 1：依特定身分集區 ID 擷取其相關資訊。

```
Get-CGIIIdentityPool -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 Cmdlet Reference (V4) 中的 [DescribeIdentityPool](#)。

PowerShell V5 的工具

範例 1：依特定身分集區 ID 擷取其相關資訊。

```
Get-CGIIIdentityPool -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 Cmdlet Reference (V5) 中的 [DescribeIdentityPool](#)。

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。

GetCredentialsForIdentity 搭配 AWS SDK 使用

以下程式碼範例顯示如何使用 GetCredentialsForIdentity。

Java

適用於 Java 2.x 的 SDK

Note

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

```
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.CognitoIdentityProviderException;

/**
 * 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》中的 [GetCredentialsForIdentity](#)。

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。

搭配使用 **GetIdentityPoolRoles** 與 CLI

下列程式碼範例示範如何使用 `GetIdentityPoolRoles`。

CLI

AWS CLI

取得身分集區角色

此範例會取得身分集區角色。

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》中的 [GetIdentityPoolRoles](#)。

PowerShell

PowerShell V4 的工具

範例 1：取得特定身分集區角色的相關資訊。

```
Get-CGIIIdentityPoolRole -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 《Cmdlet Reference (V4)》中的 [GetIdentityPoolRoles](#)。

PowerShell V5 的工具

範例 1：取得特定身分集區之角色的相關資訊。

```
Get-CGIIIdentityPoolRole -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 《Cmdlet Reference (V5)》中的 [GetIdentityPoolRoles](#)。

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。

ListIdentityPools 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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》中的 [ListIdentityPools](#)。

Java

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.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.CognitoIdentityProviderException;

/**
 * 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》中的 [ListIdentityPools](#)。

PowerShell

PowerShell V4 的工具

範例 1：擷取現有身分集區的清單。

```
Get-CGIIIdentityPoolList
```

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 Cmdlet Reference (V4) 中的 [ListIdentityPools](#)。

PowerShell V5 的工具

範例 1：擷取現有身分集區的清單。

```
Get-CGIIIdentityPoolList
```

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 Cmdlet Reference (V5) 中的 [ListIdentityPools](#)。

Swift

SDK for Swift

Note

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

```
import AWSCognitoIdentity

/// 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? {
    let listPoolsInput = ListIdentityPoolsInput(maxResults: 25)
    // Use "Paginated" to get all the objects.
    // This lets the SDK handle the 'nextToken' field in
    "ListIdentityPoolsOutput".
    let pages = cognitoIdentityClient.listIdentityPoolsPaginated(input:
listPoolsInput)

    do {
        for try await page in pages {
            guard let identityPools = page.identityPools else {
                print("ERROR: listIdentityPoolsPaginated returned nil
contents.")
                continue
            }

            /// 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.

            for pool in identityPools {
                if pool.identityPoolName == name {
                    return pool.identityPoolId!
                }
            }
        } catch {
            print("ERROR: getIdentityPoolID:", dump(error))
            throw error
        }

        return nil
    }
}
```

取得現有身分集區的 ID 或建立它 (如果尚不存在)。

```
import AWSCognitoIdentity
```

```
/// 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.

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

        return poolId
    } catch {
        print("ERROR: getOrCreateIdentityPoolID:", dump(error))
        throw error
    }
}
```

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

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。

搭配使用 **SetIdentityPoolRoles** 與 CLI

下列程式碼範例示範如何使用 SetIdentityPoolRoles。

CLI

AWS CLI

設定身分集區角色

下列 `set-identity-pool-roles` 範例會設定身分集區角色。

```
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》中的 [SetIdentityPoolRoles](#)。

PowerShell

PowerShell V4 的工具

範例 1：將特定身分集區設定為具有未驗證的 IAM 角色。

```
Set-CGIIIdentityPoolRole -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 Cmdlet Reference (V4) 中的 [SetIdentityPoolRoles](#)。

PowerShell V5 的工具

範例 1：將特定身分集區設定為具有未驗證的 IAM 角色。

```
Set-CGIIIdentityPoolRole -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 Cmdlet Reference (V5) 中的 [SetIdentityPoolRoles](#)。

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。

搭配使用 **UpdateIdentityPool** 與 CLI

下列程式碼範例示範如何使用 `UpdateIdentityPool`。

CLI

AWS CLI

更新身分集區

此範例會更新身分集區。它會將名稱設定為 MyIdentityPool。它將 Cognito 新增為身分提供者。它不允許未驗證的身分。

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_11111111",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_11111111",
      "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
      "ServerSideTokenCheck": false
    }
  ]
}
```

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

PowerShell

PowerShell V4 的工具

範例 1：更新一些身分集區屬性，在此情況下為身分集區的名稱。


```
Update-CGIIIdentityPool -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 Cmdlet Reference (V4) 中的 [UpdateIdentityPool](#)。

PowerShell V5 的工具

範例 1：更新一些身分集區屬性，在此情況下為身分集區的名稱。

```
Update-CGIIIdentityPool -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 Cmdlet Reference (V5) 中的 [UpdateIdentityPool](#)。

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 AWS SDKs Amazon Cognito 身分案例

下列程式碼範例示範如何在 Amazon Cognito Identity AWS SDKs 中實作常見案例。這些案例說明如何透過呼叫 Amazon Cognito Identity 中的多個函數或與其他函數結合來完成特定任務 AWS 服務。每個案例均包含完整原始碼的連結，您可在連結中找到如何設定和執程式碼的相關指示。

案例的目標是獲得中等水平的經驗，協助您了解內容中的服務動作。

範例

- [建立 Amazon Textract Explorer 應用程式](#)

建立 Amazon Textract Explorer 應用程式

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

JavaScript

適用於 JavaScript (v3) 的 SDK

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

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例中使用的服務

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK for Python (Boto3)

顯示如何使用 適用於 Python (Boto3) 的 AWS SDK 搭配 Amazon Textract 來偵測文件映像中的文字、表單和資料表元素。輸入影像和 Amazon Textract 輸出會顯示在 Tkinter 應用程式中，可讓您探索偵測到的元素。

- 將文件影像提交到 Amazon Textract，並探索偵測到元素的輸出。
- 將影像直接傳送至 Amazon Textract 或透過 Amazon Simple Storage Service (Amazon S3) 儲存貯體。
- 使用非同步 API 可以在任務完成時啟動將通知發布到 Amazon Simple Notification Service (Amazon SNS) 主題的任務。
- 輪詢 Amazon Simple Queue Service (Amazon SQS) 佇列以取得任務完成訊息並顯示結果。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例中使用的服務

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

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

使用 AWS SDKs 的 Amazon Cognito 身分提供者程式碼範例

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

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

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

SDK for C++

Note

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

CMakeLists.txt CMake 檔案的程式碼。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS 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 適用於 C++ 的 AWS SDK API 參考中的 [ListUserPools](#)。

Go

SDK for Go V2

Note

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

```
package main  
  
import (  
    "context"  
    "fmt"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/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() {  
    ctx := context.Background()  
    sdkConfig, err := config.LoadDefaultConfig(ctx)
```

```
if err != nil {
    fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
    fmt.Println(err)
    return
}
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(ctx)
    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\t%v: %v\n", *pool.Name, *pool.Id)
    }
}
}
```

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

Java

SDK for Java 2.x

Note

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


```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
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 參考中的 [ListUserPools](#)。

JavaScript

SDK for JavaScript (v3)

Note

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

```
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 適用於 JavaScript 的 AWS SDK API 參考中的 [ListUserPools](#)。

Python

SDK for Python (Boto3)

Note

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

```
import boto3

# Create a Cognito Identity Provider client
cognitoidp = boto3.client("cognito-idp")

# Initialize a paginator for the list_user_pools operation
paginator = cognitoidp.get_paginator("list_user_pools")

# Create a PageIterator from the paginator
page_iterator = paginator.paginate(MaxResults=10)

# Initialize variables for pagination
user_pools = []

# Handle pagination
for page in page_iterator:
    user_pools.extend(page.get("UserPools", []))

# Print the list of user pools
print("User Pools for the account:")
if user_pools:
    for pool in user_pools:
        print(f"Name: {pool['Name']}, ID: {pool['Id']}")
else:
    print("No user pools found.")
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [ListUserPools](#)。

Ruby

SDK for Ruby

Note

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

```
require 'aws-sdk-cognitoidentityprovider'
require 'logger'

# CognitoManager is a class responsible for managing AWS Cognito operations
# such as listing all user pools in the current AWS account.
class CognitoManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all user pools associated with the AWS account.
  def list_user_pools
    paginator = @client.list_user_pools(max_results: 10)
    user_pools = []
    paginator.each_page do |page|
      user_pools.concat(page.user_pools)
    end

    if user_pools.empty?
      @logger.info('No Cognito user pools found.')
    else
      user_pools.each do |user_pool|
        @logger.info("User pool ID: #{user_pool.id}")
        @logger.info("User pool name: #{user_pool.name}")
      end
    end
  end
end
```

```

        @logger.info("User pool status: #{user_pool.status}")
        @logger.info('---')
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  cognito_client = Aws::CognitoIdentityProvider::Client.new
  manager = CognitoManager.new(cognito_client)
  manager.list_user_pools
end

```

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

程式碼範例

- [使用 AWS SDKs 的 Amazon Cognito 身分提供者基本範例](#)
 - [Hello Amazon Cognito](#)
 - [使用 AWS SDKs 的 Amazon Cognito 身分提供者動作](#)
 - [AdminCreateUser 搭配 AWS SDK 或 CLI 使用](#)
 - [AdminGetUser 搭配 AWS SDK 或 CLI 使用](#)
 - [AdminInitiateAuth 搭配 AWS SDK 或 CLI 使用](#)
 - [AdminRespondToAuthChallenge 搭配 AWS SDK 或 CLI 使用](#)
 - [AdminSetUserPassword 搭配 AWS SDK 或 CLI 使用](#)
 - [AssociateSoftwareToken 搭配 AWS SDK 或 CLI 使用](#)
 - [ConfirmDevice 搭配 AWS SDK 或 CLI 使用](#)
 - [ConfirmForgotPassword 搭配 AWS SDK 或 CLI 使用](#)
 - [ConfirmSignUp 搭配 AWS SDK 或 CLI 使用](#)
 - [CreateUserPool 搭配 AWS SDK 或 CLI 使用](#)
 - [CreateUserPoolClient 搭配 AWS SDK 或 CLI 使用](#)
 - [DeleteUser 搭配 AWS SDK 或 CLI 使用](#)
 - [ForgotPassword 搭配 AWS SDK 或 CLI 使用](#)
 - [InitiateAuth 搭配 AWS SDK 或 CLI 使用](#)

- [ListUserPools 搭配 AWS SDK 或 CLI 使用](#)
- [ListUsers 搭配 AWS SDK 或 CLI 使用](#)
- [ResendConfirmationCode 搭配 AWS SDK 或 CLI 使用](#)
- [RespondToAuthChallenge 搭配 AWS SDK 或 CLI 使用](#)
- [SignUp 搭配 AWS SDK 或 CLI 使用](#)
- [UpdateUserPool 搭配 AWS SDK 或 CLI 使用](#)
- [VerifySoftwareToken 搭配 AWS SDK 或 CLI 使用](#)
- [使用 AWS SDKs Amazon Cognito 身分提供者案例](#)
 - [使用 SDK 透過 Lambda 函數自動確認已知的 Amazon Cognito 使用者 AWS](#)
 - [使用 SDK 透過 Lambda 函數自動遷移已知的 Amazon Cognito 使用者 AWS](#)
 - [使用 SDK 向需要 MFA 的 Amazon Cognito 使用者集區註冊使用者 AWS](#)
 - [使用 AWS SDK 進行 Amazon Cognito 使用者身分驗證後，使用 Lambda 函數寫入自訂活動資料](#)

使用 AWS SDKs 的 Amazon Cognito 身分提供者基本範例

下列程式碼範例示範如何搭配 AWS SDKs 使用 Amazon Cognito Identity Provider 的基本概念。

範例

- [Hello Amazon Cognito](#)
- [使用 AWS SDKs 的 Amazon Cognito 身分提供者動作](#)
 - [AdminCreateUser 搭配 AWS SDK 或 CLI 使用](#)
 - [AdminGetUser 搭配 AWS SDK 或 CLI 使用](#)
 - [AdminInitiateAuth 搭配 AWS SDK 或 CLI 使用](#)
 - [AdminRespondToAuthChallenge 搭配 AWS SDK 或 CLI 使用](#)
 - [AdminSetUserPassword 搭配 AWS SDK 或 CLI 使用](#)
 - [AssociateSoftwareToken 搭配 AWS SDK 或 CLI 使用](#)
 - [ConfirmDevice 搭配 AWS SDK 或 CLI 使用](#)
 - [ConfirmForgotPassword 搭配 AWS SDK 或 CLI 使用](#)
 - [ConfirmSignUp 搭配 AWS SDK 或 CLI 使用](#)
 - [CreateUserPool 搭配 AWS SDK 或 CLI 使用](#)
 - [CreateUserPoolClient 搭配 AWS SDK 或 CLI 使用](#)

- [DeleteUser 搭配 AWS SDK 或 CLI 使用](#)
- [ForgotPassword 搭配 AWS SDK 或 CLI 使用](#)
- [InitiateAuth 搭配 AWS SDK 或 CLI 使用](#)
- [ListUserPools 搭配 AWS SDK 或 CLI 使用](#)
- [ListUsers 搭配 AWS SDK 或 CLI 使用](#)
- [ResendConfirmationCode 搭配 AWS SDK 或 CLI 使用](#)
- [RespondToAuthChallenge 搭配 AWS SDK 或 CLI 使用](#)
- [SignUp 搭配 AWS SDK 或 CLI 使用](#)
- [UpdateUserPool 搭配 AWS SDK 或 CLI 使用](#)
- [VerifySoftwareToken 搭配 AWS SDK 或 CLI 使用](#)

Hello Amazon Cognito

下列程式碼範例顯示如何開始使用 Amazon Cognito。

C++

SDK for C++

Note

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

CMakeLists.txt CMake 檔案的程式碼。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS 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 適用於 C++ 的 AWS SDK API 參考中的 [ListUserPools](#)。

Go

SDK for Go V2

Note

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

```
package main

import (
    "context"
    "fmt"
    "log"
```

```
"github.com/aws/aws-sdk-go-v2/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() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    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(ctx)
        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\t%v: %v\n", *pool.Name, *pool.Id)
        }
    }
}
```

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

Java

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
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 參考中的 [ListUserPools](#)。

JavaScript

SDK for JavaScript (v3)

Note

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

```
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 適用於 JavaScript 的 AWS SDK API 參考中的 [ListUserPools](#)。

Python

SDK for Python (Boto3)

Note

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

```
import boto3

# Create a Cognito Identity Provider client
cognitoidp = boto3.client("cognito-idp")

# Initialize a paginator for the list_user_pools operation
paginator = cognitoidp.get_paginator("list_user_pools")

# Create a PageIterator from the paginator
page_iterator = paginator.paginate(MaxResults=10)

# Initialize variables for pagination
```

```
user_pools = []

# Handle pagination
for page in page_iterator:
    user_pools.extend(page.get("UserPools", []))

# Print the list of user pools
print("User Pools for the account:")
if user_pools:
    for pool in user_pools:
        print(f"Name: {pool['Name']}, ID: {pool['Id']}")
else:
    print("No user pools found.")
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [ListUserPools](#)。

Ruby

SDK for Ruby

Note

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

```
require 'aws-sdk-cognitoidentityprovider'
require 'logger'

# CognitoManager is a class responsible for managing AWS Cognito operations
# such as listing all user pools in the current AWS account.
class CognitoManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end
end
```

```
# Lists and prints all user pools associated with the AWS account.
def list_user_pools
  paginator = @client.list_user_pools(max_results: 10)
  user_pools = []
  paginator.each_page do |page|
    user_pools.concat(page.user_pools)
  end

  if user_pools.empty?
    @logger.info('No Cognito user pools found.')
  else
    user_pools.each do |user_pool|
      @logger.info("User pool ID: #{user_pool.id}")
      @logger.info("User pool name: #{user_pool.name}")
      @logger.info("User pool status: #{user_pool.status}")
      @logger.info('---')
    end
  end
end

end

end

if $PROGRAM_NAME == __FILE__
  cognito_client = Aws::CognitoIdentityProvider::Client.new
  manager = CognitoManager.new(cognito_client)
  manager.list_user_pools
end
```

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

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 AWS SDKs 的 Amazon Cognito 身分提供者動作

下列程式碼範例示範如何使用 AWS SDKs 執行個別 Amazon Cognito 身分提供者動作。每個範例均包含 GitHub 的連結，您可以在連結中找到設定和執行程式碼的相關說明。

這些摘錄會呼叫 Amazon 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。您可以在 [使用 AWS SDKs Amazon Cognito 身分提供者案例](#) 中查看內容中的動作。

下列範例僅包含最常使用的動作。如需完整清單，請參閱 [Amazon Cognito 身分提供者 API 參考](#)。

範例

- [AdminCreateUser 搭配 AWS SDK 或 CLI 使用](#)
- [AdminGetUser 搭配 AWS SDK 或 CLI 使用](#)
- [AdminInitiateAuth 搭配 AWS SDK 或 CLI 使用](#)
- [AdminRespondToAuthChallenge 搭配 AWS SDK 或 CLI 使用](#)
- [AdminSetUserPassword 搭配 AWS SDK 或 CLI 使用](#)
- [AssociateSoftwareToken 搭配 AWS SDK 或 CLI 使用](#)
- [ConfirmDevice 搭配 AWS SDK 或 CLI 使用](#)
- [ConfirmForgotPassword 搭配 AWS SDK 或 CLI 使用](#)
- [ConfirmSignUp 搭配 AWS SDK 或 CLI 使用](#)
- [CreateUserPool 搭配 AWS SDK 或 CLI 使用](#)
- [CreateUserPoolClient 搭配 AWS SDK 或 CLI 使用](#)
- [DeleteUser 搭配 AWS SDK 或 CLI 使用](#)
- [ForgotPassword 搭配 AWS SDK 或 CLI 使用](#)
- [InitiateAuth 搭配 AWS SDK 或 CLI 使用](#)
- [ListUserPools 搭配 AWS SDK 或 CLI 使用](#)
- [ListUsers 搭配 AWS SDK 或 CLI 使用](#)
- [ResendConfirmationCode 搭配 AWS SDK 或 CLI 使用](#)
- [RespondToAuthChallenge 搭配 AWS SDK 或 CLI 使用](#)
- [SignUp 搭配 AWS SDK 或 CLI 使用](#)
- [UpdateUserPool 搭配 AWS SDK 或 CLI 使用](#)
- [VerifySoftwareToken 搭配 AWS SDK 或 CLI 使用](#)

AdminCreateUser 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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：

- [進行 Amazon Cognito 使用者身分驗證後，可使用 Lambda 函數撰寫自訂活動資料](#)

CLI

AWS CLI

建立使用者

下列admin-create-user範例會建立具有指定設定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》中的 [AdminCreateUser](#)。

Go

SDK for Go V2

 Note

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

```
import (
    "context"
    "errors"
    "log"

    "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"
)

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(ctx context.Context, userPoolId
    string, userName string, userEmail string) error {
    _, err := actor.CognitoClient.AdminCreateUser(ctx,
        &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，請參閱適用於 Go 的 AWS SDK 《API 參考》中的 [AdminCreateUser](#)。

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。

AdminGetUser 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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

適用於 .NET 的 SDK

Note

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

```
/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [AdminGetUser](#)。

C++

SDK for C++

Note

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

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

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《適用於 C++ 的 AWS SDK API 參考》中的 [AdminGetUser](#)。

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_aaaaaaaaaa --
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》中的 [AdminGetUser](#)。

Java

SDK for Java 2.x

Note

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

```
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 [AdminGetUser](#)。

JavaScript

SDK for JavaScript (v3)

Note

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

```
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《適用於 JavaScript 的 AWS SDK API 參考》中的 [AdminGetUser](#)。

Kotlin

SDK for Kotlin

Note

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

```
suspend fun 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}")
    }
}
```

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

Python

SDK for Python (Boto3)

Note

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

```

class 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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [AdminGetUser](#)。

Swift

SDK for Swift

Note

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

```
import AWSClientRuntime
```

```
import AWSCognitoIdentityProvider

/// Get information about a specific user in a user pool.
///
/// - Parameters:
///   - cipClient: The Amazon Cognito Identity Provider client to use.
///   - userName: The user to retrieve information about.
///   - userPoolId: The user pool to search for the specified user.
///
/// - Returns: `true` if the user's information was successfully
///   retrieved. Otherwise returns `false`.
func adminGetUser(cipClient: CognitoIdentityProviderClient, userName: String,
                  userPoolId: String) async -> Bool {
    do {
        let output = try await cipClient.adminGetUser(
            input: AdminGetUserInput(
                userPoolId: userPoolId,
                username: userName
            )
        )

        guard let userStatus = output.userStatus else {
            print("*** Unable to get the user's status.")
            return false
        }

        print("User status: \(userStatus)")
        return true
    } catch {
        return false
    }
}
```

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

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。

AdminInitiateAuth 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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

適用於 .NET 的 SDK

Note

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

```
/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [AdminInitiateAuth](#)。

C++

SDK for C++

Note

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

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

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《適用於 C++ 的 AWS SDK API 參考》中的 [AdminInitiateAuth](#)。

CLI

AWS CLI

以管理員身分登入使用者

下列 admin-initiate-auth 範例會在使用者 diego@example.com 中登入。此範例也包含威脅防護的中繼資料和 Lambda 觸發的 ClientMetadata。使用者已設定為 TOTP MFA，並收到挑戰，要求他們先從驗證器應用程式提供程式碼，才能完成身分驗證。

```

aws cognito-idp admin-initiate-auth \
  --user-pool-id us-west-2_EXAMPLE \
  --client-id 1example23456789 \
  --auth-flow ADMIN_USER_PASSWORD_AUTH \
  --auth-parameters USERNAME=diego@example.com,PASSWORD="My@Example
$Password3!",SECRET_HASH=ExampleEncodedClientIdSecretAndUsername= \
  --context-data="{\"EncodedData\": \"abc123example\", \"HttpHeaders\":
[{\\"headerName\": \"UserAgent\", \"headerValue\": \"Mozilla/5.0 (Windows NT
6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0\"}], \"IpAddress\":
\\\"192.0.2.1\\\", \"ServerName\": \"example.com\", \"ServerPath\": \"/login/\"} \"
  --client-metadata="{\"MyExampleKey\": \"MyExampleValue\"}"

```

輸出：

```

{
  "ChallengeName": "SOFTWARE_TOKEN_MFA",
  "Session": "AYABeExample...",
  "ChallengeParameters": {
    "FRIENDLY_DEVICE_NAME": "MyAuthenticatorApp",
    "USER_ID_FOR_SRP": "diego@example.com"
  }
}

```

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的 [管理員身分驗證流程](#)。

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

Java

SDK for Java 2.x

Note

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

```
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》中的 [AdminInitiateAuth](#)。

JavaScript

SDK for JavaScript (v3)

Note

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

```
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《適用於 JavaScript 的 AWS SDK API 參考》中的 [AdminInitiateAuth](#)。

Kotlin

SDK for Kotlin

Note

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

```
suspend fun 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
    }
}
```

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

Python

SDK for Python (Boto3)

Note

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

```
class 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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [AdminInitiateAuth](#)。

Swift

SDK for Swift

Note

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

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

/// Begin an authentication session.
///
/// - Parameters:
///   - cipClient: The `CognitoIdentityProviderClient` to use.
///   - clientId: The app client ID to use.
///   - userName: The username to check.
///   - password: The user's password.
///   - userPoolId: The user pool to use.
///
/// - Returns: The session token associated with this authentication
///   session.
func initiateAuth(cipClient: CognitoIdentityProviderClient, clientId: String,
                  userName: String, password: String,
                  userPoolId: String) async -> String? {
    var authParams: [String: String] = [:]

    authParams["USERNAME"] = userName
    authParams["PASSWORD"] = password

    do {
        let output = try await cipClient.adminInitiateAuth(
            input: AdminInitiateAuthInput(
                authFlow:
CognitoIdentityProviderClientTypes.AuthFlowType.adminUserPasswordAuth,
                authParameters: authParams,
                clientId: clientId,
                userPoolId: userPoolId
            )
        )
    }
```

```
        guard let challengeName = output.challengeName else {
            print("*** Invalid response from the auth service.")
            return nil
        }

        print("=====> Response challenge is \(challengeName)")

        return output.session
    } catch _ as UserNotFoundException {
        print("*** The specified username, \(userName), doesn't exist.")
        return nil
    } catch _ as UserNotConfirmedException {
        print("*** The user \(userName) has not been confirmed.")
        return nil
    } catch {
        print("*** An unexpected error occurred.")
        return nil
    }
}
```

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

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。

AdminRespondToAuthChallenge 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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

適用於 .NET 的 SDK

Note

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

```
/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [AdminRespondToAuthChallenge](#)。

C++

SDK for C++

Note

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

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

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《適用於 C++ 的 AWS SDK API 參考》中的 [AdminRespondToAuthChallenge](#)。

CLI

AWS CLI

回應身分驗證挑戰

根據您的身分驗證流程、使用者集區組態和使用者的設定，有許多方法可以回應不同的身分驗證挑戰。下列 `admin-respond-to-auth-challenge` 範例提供 `diego@example.com` 的 TOTP MFA 代碼並完成登入。此使用者集區已開啟裝置記住，因此身分驗證結果也會傳回新的裝置金鑰。

```

aws cognito-idp admin-respond-to-auth-challenge \
  --user-pool-id us-west-2_EXAMPLE \
  --client-id 1example23456789 \
  --challenge-name SOFTWARE_TOKEN_MFA \
  --challenge-
responses USERNAME=diego@example.com,SOFTWARE_TOKEN_MFA_CODE=000000 \

```

```
--session AYABeExample...
```

輸出：

```
{
  "ChallengeParameters": {},
  "AuthenticationResult": {
    "AccessToken": "eyJra456defEXAMPLE",
    "ExpiresIn": 3600,
    "TokenType": "Bearer",
    "RefreshToken": "eyJra123abcEXAMPLE",
    "IdToken": "eyJra789ghiEXAMPLE",
    "NewDeviceMetadata": {
      "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "DeviceGroupKey": "-ExAmPlE1"
    }
  }
}
```

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的[管理員身分驗證流程](#)。

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

Java

SDK for Java 2.x

Note

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

```
// 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》中的 [AdminRespondToAuthChallenge](#)。

JavaScript

SDK for JavaScript (v3)

Note

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

```
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《適用於 JavaScript 的 AWS SDK API 參考》中的 [AdminRespondToAuthChallenge](#)。

Kotlin

SDK for Kotlin

Note

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

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponses0b = mutableMapOf<String, String>()
    challengeResponses0b["USERNAME"] = userName
    challengeResponses0b["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
```

```

        clientId = clientIdVal
        challengeResponses = challengeResponses0b
        session = sessionVal
    }

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

```

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

Python

SDK for Python (Boto3)

Note

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

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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [AdminRespondToAuthChallenge](#)。

Swift

SDK for Swift

Note

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

```

import AWSClientRuntime
import AWSCognitoIdentityProvider

/// Respond to the authentication challenge received from Cognito after
/// initiating an authentication session. This involves sending a current
/// MFA code to the service.
///
/// - Parameters:
///   - cipClient: The `CognitoIdentityProviderClient` to use.

```

```

    /// - userName: The user's username.
    /// - clientId: The app client ID.
    /// - userPoolId: The user pool to sign into.
    /// - mfaCode: The 6-digit MFA code currently displayed by the user's
    ///   authenticator.
    /// - session: The authentication session to continue processing.
    func adminRespondToAuthChallenge(cipClient: CognitoIdentityProviderClient,
    userName: String,
                                clientId: String, userPoolId: String,
    mfaCode: String,
                                session: String) async {
        print("=====> SOFTWARE_TOKEN_MFA challenge is generated...")

        var challengeResponsesOb: [String: String] = [:]
        challengeResponsesOb["USERNAME"] = userName
        challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

        do {
            let output = try await cipClient.adminRespondToAuthChallenge(
                input: AdminRespondToAuthChallengeInput(
                    challengeName:
CognitoIdentityProviderClientTypes.ChallengeNameType.softwareTokenMfa,
                    challengeResponses: challengeResponsesOb,
                    clientId: clientId,
                    session: session,
                    userPoolId: userPoolId
                )
            )

            guard let authenticationResult = output.authenticationResult else {
                print("*** Unable to get authentication result.")
                return
            }

            print("=====> Authentication result (JWTs are redacted):")
            print(authenticationResult)
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")

```



```
        return
    } catch _ as UserNotConfirmedException {
        print("*** The user \(userName) has not been confirmed.")
        return
    } catch let error as NotAuthorizedException {
        print("*** Unauthorized access. Reason: \(error.properties.message ??
"<unknown>")")
    } catch {
        print("*** Error responding to the MFA challenge.")
        return
    }
}
```

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

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。

AdminSetUserPassword 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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：

- [進行 Amazon Cognito 使用者身分驗證後，可使用 Lambda 函數撰寫自訂活動資料](#)

CLI

AWS CLI

將使用者密碼設定為管理員

下列admin-set-user-password範例會永久設定 diego@example.com 的密碼。

```
aws cognito-idp admin-set-user-password \
  --user-pool-id us-west-2_EXAMPLE \
  --username diego@example.com \
  --password MyExamplePassword1! \
  --permanent
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的[密碼、密碼復原和密碼政策](#)。

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

Go

SDK for Go V2

 Note

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

```
import (
    "context"
    "errors"
    "log"

    "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"
)

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(ctx context.Context, userPoolId
    string, userName string, password string) error {
    _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
        &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，請參閱適用於 Go 的 AWS SDK 《API 參考》中的 [AdminSetUserPassword](#)。

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。

AssociateSoftwareToken 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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

適用於 .NET 的 SDK

Note

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

```
/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [AssociateSoftwareToken](#)。

C++

SDK for C++

Note

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

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

```

    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《適用於 C++ 的 AWS SDK API 參考》中的 [AssociateSoftwareToken](#)。

CLI

AWS CLI

為 MFA 驗證器應用程式產生私密金鑰

下列 `associate-software-token` 範例會為已登入並收到存取權杖的使用者產生 TOTP 私有金鑰。產生的私有金鑰可以手動輸入到驗證器應用程式中，或者應用程式可以將它轉譯為使用者可以掃描的 QR 碼。

```
aws cognito-idp associate-software-token \  
  --access-token eyJra456defEXAMPLE
```

輸出：

```
{  
  "SecretCode": "QWERTYUIOP123456EXAMPLE"  
}
```

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的 [TOTP 軟體字符 MFA](#)。

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

Java

SDK for Java 2.x

Note

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

```
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》中的 [AssociateSoftwareToken](#)。

JavaScript

SDK for JavaScript (v3)

Note

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

```
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《適用於 JavaScript 的 AWS SDK API 參考》中的 [AssociateSoftwareToken](#)。

Kotlin

SDK for Kotlin

Note

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

```
suspend fun 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
}
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Kotlin API 參考》中的 [AssociateSoftwareToken](#)。

Python

SDK for Python (Boto3)

Note

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


```
class 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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [AssociateSoftwareToken](#)。

Swift

SDK for Swift

Note

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

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

/// Request and display an MFA secret token that the user should enter
/// into their authenticator to set it up for the user account.
///
/// - Parameters:
///   - cipClient: The `CognitoIdentityProviderClient` to use.
///   - authSession: The authentication session to request an MFA secret
///     for.
///
/// - Returns: A string containing the MFA secret token that should be
///   entered into the authenticator software.
func getSecretForAppMFA(cipClient: CognitoIdentityProviderClient,
authSession: String?) async -> String? {
    do {
        let output = try await cipClient.associateSoftwareToken(
            input: AssociateSoftwareTokenInput(
                session: authSession
            )
        )

        guard let secretCode = output.secretCode else {
            print("*** Unable to get the secret code")
            return nil
        }
    }
}
```

```
        print("=====> Enter this token into Google Authenticator:
\\(secretCode)")
        return output.session
    } catch _ as SoftwareTokenMFANotFoundException {
        print("*** The specified user pool isn't configured for MFA.")
        return nil
    } catch {
        print("*** An unexpected error occurred getting the secret for the
app's MFA.")
        return nil
    }
}
```

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

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。

ConfirmDevice 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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

適用於 .NET 的 SDK

Note

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

```
/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [ConfirmDevice](#)。

CLI

AWS CLI

確認使用者裝置

下列confirm-device範例會為目前使用者新增記住的裝置。

```

aws cognito-idp confirm-device \
  --access-token eyJra456defEXAMPLE \
  --device-key us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
  --device-secret-verifier-
config PasswordVerifier=TXlWZXJpZmllc1N0cmLuZw,Salt=TXlTUlBTYWx0

```

輸出：

```
{
```

```
"UserConfirmationNecessary": false
}
```

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的[在使用者集區中使用使用者裝置](#)。

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

JavaScript

SDK for JavaScript (v3)

Note

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

```
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《適用於 JavaScript 的 AWS SDK API 參考》中的 [ConfirmDevice](#)。

Python

SDK for Python (Boto3)

Note

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

```
class 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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [ConfirmDevice](#)。

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。

ConfirmForgotPassword 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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 函數自動遷移已知使用者](#)

CLI

AWS CLI

確認忘記的密碼

此範例會確認使用者名稱 `diego@example.com` 的忘記密碼。

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》中的 [ConfirmForgotPassword](#)。

Go

SDK for Go V2

 Note

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

```
import (
    "context"
    "errors"
    "log"

    "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"
)

type CognitoActions struct {
    CognitoClient *cognitoidentityprovider.Client
}

// ConfirmForgotPassword confirms a user with a confirmation code and a new
// password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
    string, code string, userName string, password string) error {
    _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
        &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，請參閱適用於 Go 的 AWS SDK 《API 參考》中的 [ConfirmForgotPassword](#)。

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。

ConfirmSignUp 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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

適用於 .NET 的 SDK

Note

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

```
/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [ConfirmSignUp](#)。

C++

SDK for C++

Note

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

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

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《適用於 C++ 的 AWS SDK API 參考》中的 [ConfirmSignUp](#)。

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 API 參考》中的 [ConfirmSignUp](#)。

Java

SDK for Java 2.x

Note

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

```
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》中的 [ConfirmSignUp](#)。

JavaScript

SDK for JavaScript (v3)

Note

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

```
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《適用於 JavaScript 的 AWS SDK API 參考》中的 [ConfirmSignUp](#)。

Kotlin

SDK for Kotlin

Note

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

```
suspend fun 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")
    }
}
```

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

Python

SDK for Python (Boto3)

Note

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

```
class 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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [ConfirmSignUp](#)。

Swift

SDK for Swift

Note

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

```
import AWSClientRuntime
```



```

import AWSCognitoIdentityProvider

/// Submit a confirmation code for the specified user. This is the code as
/// entered by the user after they've received it by email or text
/// message.
///
/// - Parameters:
///   - cipClient: The `CognitoIdentityProviderClient` to use.
///   - clientId: The app client ID the user is signing up for.
///   - userName: The username of the user whose code is being sent.
///   - code: The user's confirmation code.
///
/// - Returns: `true` if the code was successfully confirmed; otherwise
/// `false`.
func confirmSignUp(cipClient: CognitoIdentityProviderClient, clientId:
String,
                  userName: String, code: String) async -> Bool {
    do {
        _ = try await cipClient.confirmSignUp(
            input: ConfirmSignUpInput(
                clientId: clientId,
                confirmationCode: code,
                username: userName
            )
        )

        print("=====> \(userName) has been confirmed.")
        return true
    } catch {
        print("=====> \(userName)'s code was entered incorrectly.")
        return false
    }
}

```

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

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。

CreateUserPool 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 CreateUserPool。

CLI

AWS CLI

建立最低限度設定的使用者集區

此範例會使用預設值建立名為 `MyUserPool` 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_aaaaaaaaaa",
    "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》中的 [CreateUserPool](#)。

Java

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
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》中的 [CreateUserPool](#)。

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。

CreateUserPoolClient 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 CreateUserPoolClient。

CLI

AWS CLI

建立使用者集區用戶端

下列create-user-pool-client範例會建立新的使用者集區用戶端，其中包含用戶端秘密、明確的讀取和寫入屬性、使用使用者名稱密碼和 SRP 流程登入、使用三個 IdPs 登入、存取 OAuth 範圍的子集、PinPoint 分析，以及延伸的身分驗證工作階段有效性。

```
aws cognito-idp create-user-pool-client \
  --user-pool-id us-west-2_EXAMPLE \
  --client-name MyTestClient \
  --generate-secret \
  --refresh-token-validity 10 \
  --access-token-validity 60 \
  --id-token-validity 60 \
  --token-validity-units AccessToken=minutes,IdToken=minutes,RefreshToken=days \
  --read-attributes email phone_number email_verified phone_number_verified \
  --write-attributes email phone_number \
  --explicit-auth-flows ALLOW_USER_PASSWORD_AUTH ALLOW_USER_SRP_AUTH ALLOW_REFRESH_TOKEN_AUTH \
  --supported-identity-providers Google Facebook MyOIDC \
  --callback-urls https://www.amazon.com https://example.com http://localhost:8001 myapp://example \
  --allowed-o-auth-flows code implicit \
  --allowed-o-auth-scopes openid profile aws.cognito.signin.user.admin solar-system-data/asteroids.add \
  --allowed-o-auth-flows-user-pool-client \
  --analytics-configuration ApplicationArn=arn:aws:mobiletargeting:us-west-2:767671399759:apps/thisisanexamplepinpointapplicationid,UserDataShared=TRUE \
  --prevent-user-existence-errors ENABLED \
  --enable-token-revocation \
  --enable-propagate-additional-user-context-data \
  --auth-session-validity 4
```

輸出：

```
{
  "UserPoolClient": {
    "UserPoolId": "us-west-2_EXAMPLE",
    "ClientName": "MyTestClient",
    "ClientId": "123abc456defEXAMPLE",
    "ClientSecret": "this1234is5678my91011example1213client1415secret",
    "LastModifiedDate": 1726788459.464,
    "CreationDate": 1726788459.464,
    "RefreshTokenValidity": 10,
    "AccessTokenValidity": 60,
    "IdTokenValidity": 60,
    "TokenValidityUnits": {
      "AccessToken": "minutes",
```



```
        "IdToken": "minutes",
        "RefreshToken": "days"
    },
    "ReadAttributes": [
        "email_verified",
        "phone_number_verified",
        "phone_number",
        "email"
    ],
    "WriteAttributes": [
        "phone_number",
        "email"
    ],
    "ExplicitAuthFlows": [
        "ALLOW_USER_PASSWORD_AUTH",
        "ALLOW_USER_SRP_AUTH",
        "ALLOW_REFRESH_TOKEN_AUTH"
    ],
    "SupportedIdentityProviders": [
        "Google",
        "MyOIDC",
        "Facebook"
    ],
    "CallbackURLs": [
        "https://example.com",
        "https://www.amazon.com",
        "myapp://example",
        "http://localhost:8001"
    ],
    "AllowedOAuthFlows": [
        "implicit",
        "code"
    ],
    "AllowedOAuthScopes": [
        "aws.cognito.signin.user.admin",
        "openid",
        "profile",
        "solar-system-data/asteroids.add"
    ],
    "AllowedOAuthFlowsUserPoolClient": true,
    "AnalyticsConfiguration": {
        "ApplicationArn": "arn:aws:mobiletargeting:us-
west-2:123456789012:apps/thisisanexamplepinpointapplicationid",
```

```

        "RoleArn": "arn:aws:iam::123456789012:role/aws-service-role/cognito-
        idp.amazonaws.com/AWSServiceRoleForAmazonCognitoIdp",
        "UserDataShared": true
    },
    "PreventUserExistenceErrors": "ENABLED",
    "EnableTokenRevocation": true,
    "EnablePropagateAdditionalUserContextData": true,
    "AuthSessionValidity": 4
}
}

```

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的[應用程式用戶端的應用程式特定設定](#)。

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

Java

SDK for Java 2.x

Note

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

```

import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientResponse;

/**
 * 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》中的 [CreateUserPoolClient](#)。

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。

DeleteUser 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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 函數自動遷移已知使用者](#)
- [進行 Amazon Cognito 使用者身分驗證後，可使用 Lambda 函數撰寫自訂活動資料](#)

C++

SDK for C++

Note

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

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

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 適用於 C++ 的 AWS SDK API Reference 中的 [DeleteUser](#)。

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》中的 [DeleteUser](#)。

Go

SDK for Go V2

 Note

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

```
import (
    "context"
    "errors"
    "log"

    "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"
)

type CognitoActions struct {
    CognitoClient *cognitoidentityprovider.Client
}

// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
    string) error {
    _, err := actor.CognitoClient.DeleteUser(ctx,
        &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 適用於 Go 的 AWS SDK API Reference 中的 [DeleteUser](#)。

JavaScript

SDK for JavaScript (v3)

Note

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

```
/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
 * provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
    try {
        const client = new CognitoIdentityProviderClient({ region });
        const response = await client.send(
            new DeleteUserCommand({ AccessToken: accessToken }),
        );
        return [response, null];
    } catch (err) {
        return [null, err];
    }
};
```

- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API Reference 中的 [DeleteUser](#)。

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。

ForgotPassword 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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》中的 [ForgotPassword](#)。

Go

SDK for Go V2

 Note

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

```
import (
    "context"
    "errors"
    "log"

    "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"
)

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(ctx context.Context, clientId string,
    userName string) (*types.CodeDeliveryDetailsType, error) {
    output, err := actor.CognitoClient.ForgotPassword(ctx,
        &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，請參閱適用於 Go 的 AWS SDK 《API 參考》中的 [ForgotPassword](#)。

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。

InitiateAuth 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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 函數自動遷移已知使用者](#)
- [使用需要 MFA 的使用者集區註冊使用者](#)
- [進行 Amazon Cognito 使用者身分驗證後，可使用 Lambda 函數撰寫自訂活動資料](#)

.NET

適用於 .NET 的 SDK

Note

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

```
/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [InitiateAuth](#)。

CLI

AWS CLI

登入使用者

下列initiate-auth範例使用基本的使用者名稱密碼流程登入使用者，沒有額外的挑戰。

```

aws cognito-idp initiate-auth \
  --auth-flow USER_PASSWORD_AUTH \
  --client-id 1example23456789 \
  --analytics-metadata AnalyticsEndpointId=d70b2ba36a8c4dc5a04a0451aEXAMPLE \
  --auth-parameters USERNAME=testuser,PASSWORD=[Password] --user-context-
data EncodedData=mycontextdata --client-metadata MyTestKey=MyTestValue

```

輸出：

```
{
```

```
"AuthenticationResult": {
  "AccessToken": "eyJra456defEXAMPLE",
  "ExpiresIn": 3600,
  "TokenType": "Bearer",
  "RefreshToken": "eyJra123abcEXAMPLE",
  "IdToken": "eyJra789ghiEXAMPLE",
  "NewDeviceMetadata": {
    "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "DeviceGroupKey": "-v7w9UcY6"
  }
}
```

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的[身分驗證](#)。

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

Go

SDK for Go V2

Note

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

```
import (
    "context"
    "errors"
    "log"

    "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"
)

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(ctx context.Context, clientId string, userName
string, password string) (*types.AuthenticationResultType, error) {
    var authResult *types.AuthenticationResultType
    output, err := actor.CognitoClient.InitiateAuth(ctx,
    &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《適用於 Go 的 AWS SDK API 參考》中的 [InitiateAuth](#)。

JavaScript

SDK for JavaScript (v3)

Note

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

```
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《適用於 JavaScript 的 AWS SDK API 參考》中的 [InitiateAuth](#)。

Python

SDK for Python (Boto3)

Note

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

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 "
            f"{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_auth['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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [InitiateAuth](#)。

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。

ListUserPools 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 ListUserPools。

.NET

適用於 .NET 的 SDK (v4)

Note

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

```

/// <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 適用於 .NET 的 AWS SDK API 參考中的 [ListUserPools](#)。

CLI

AWS CLI

若要列出使用者集區

下列 `list-user-pools` 範例列出目前 CLI 登入 AWS 資料帳戶中的 3 個可用使用者集區。

```
aws cognito-idp list-user-pools \
    --max-results 3
```

輸出：

```
{
  "NextToken": "[Pagination token]",
  "UserPools": [
    {
      "CreationDate": 1681502497.741,
      "Id": "us-west-2_EXAMPLE1",
      "LambdaConfig": {
        "CustomMessage": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
        "PreSignUp": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
        "PreTokenGeneration": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
```

```
        "PreTokenGenerationConfig": {
            "LambdaArn": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
            "LambdaVersion": "V1_0"
        },
        "LastModifiedDate": 1681502497.741,
        "Name": "user pool 1"
    },
    {
        "CreationDate": 1686064178.717,
        "Id": "us-west-2_EXAMPLE2",
        "LambdaConfig": {
        },
        "LastModifiedDate": 1686064178.873,
        "Name": "user pool 2"
    },
    {
        "CreationDate": 1627681712.237,
        "Id": "us-west-2_EXAMPLE3",
        "LambdaConfig": {
            "UserMigration": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction"
        },
        "LastModifiedDate": 1678486942.479,
        "Name": "user pool 3"
    }
]
}
```

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的[Amazon Cognito 使用者集區](#)。

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

Go

SDK for Go V2

 Note

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

```
package main

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/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() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    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(ctx)
        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 適用於 Go 的 AWS SDK API 參考中的 [ListUserPools](#)。

Java

SDK for Java 2.x

Note

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

```

import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
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 參考中的 [ListUserPools](#)。

Rust

SDK for Rust

Note

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

```
async fn show_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 詳細資訊，請參閱《AWS SDK for Rust API 參考》中的 [ListUserPools](#)。

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。

ListUsers 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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

適用於 .NET 的 SDK

Note

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

```
/// <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 適用於 .NET 的 AWS SDK API Reference 中的 [ListUsers](#)。

CLI

AWS CLI

範例 1：列出具有伺服器端篩選條件的使用者

下列list-users範例列出請求的使用者集區中的 3 個使用者，其電子郵件地址開頭為testuser。

```
aws cognito-idp list-users \  
  --user-pool-id us-west-2_EXAMPLE \  
  --filter email^="testuser\" \  
  --max-items 3
```

輸出：

```
{  
  "PaginationToken": "efgh5678EXAMPLE",  
  "Users": [  
    {  
      "Attributes": [  
        {  
          "Name": "sub",  
          "Value": "eaad0219-2117-439f-8d46-4db20e59268f"  
        },  
        {  
          "Name": "email",  
          "Value": "testuser@example.com"  
        }  
      ],  
      "Enabled": true,  
      "UserCreateDate": 1682955829.578,  
      "UserLastModifiedDate": 1689030181.63,  
      "UserStatus": "CONFIRMED",  
      "Username": "testuser"  
    },  
    {  
      "Attributes": [  
        {  
          "Name": "sub",  
          "Value": "3b994cfd-0b07-4581-be46-3c82f9a70c90"  
        },  
        {  
          "Name": "email",  
          "Value": "testuser2@example.com"  
        }  
      ],  
      "Enabled": true,  
      "UserCreateDate": 1684427979.201,
```

```

        "UserLastModifiedDate": 1684427979.201,
        "UserStatus": "UNCONFIRMED",
        "Username": "testuser2"
    },
    {
        "Attributes": [
            {
                "Name": "sub",
                "Value": "5929e0d1-4c34-42d1-9b79-a5ecacfe66f7"
            },
            {
                "Name": "email",
                "Value": "testuser3@example.com"
            }
        ],
        "Enabled": true,
        "UserCreateDate": 1684427823.641,
        "UserLastModifiedDate": 1684427823.641,
        "UserStatus": "UNCONFIRMED",
        "Username": "testuser3@example.com"
    }
]
}

```

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的[管理和搜尋使用者](#)。

範例 2：列出具有用戶端篩選條件的使用者

下列list-users範例列出三個具有 屬性之使用者的屬性，在此例中是其電子郵件地址，其中包含電子郵件網域 "@example.com"。如果其他屬性包含此字串，則也會顯示這些屬性。第二個使用者沒有符合查詢的屬性，且會從顯示的輸出中排除，但不會從伺服器回應中排除。

```

aws cognito-idp list-users \
  --user-pool-id us-west-2_EXAMPLE \
  --max-items 3 \
  --query Users\[.*\].Attributes\[.*Value\.contains\(\@\.\'example.com\'\) \]

```

輸出：

```

[
  [
    {
      "Name": "email",

```

```
        "Value": "admin@example.com"
    },
    [
    [
        {
            "Name": "email",
            "Value": "operator@example.com"
        }
    ]
    ]
]
```

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的[管理和搜尋使用者](#)。

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

Java

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
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 Reference 中的 [ListUsers](#)。

JavaScript

SDK for JavaScript (v3)

Note

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

```
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 適用於 JavaScript 的 AWS SDK API Reference 中的 [ListUsers](#)。

Kotlin

SDK for Kotlin

Note

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

```
suspend fun 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}")
    }
}
}

```

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

Python

SDK for Python (Boto3)

Note

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

```

class 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 詳細資訊，請參閱 AWS SDK for Python (Boto3) API Reference 中的 [ListUsers](#)。

Swift

SDK for Swift

Note

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

```
do {
    let output = try await cognitoClient.listUsers(
        input: ListUsersInput(
            userPoolId: poolId
        )
    )

    guard let users = output.users else {
        print("No users found.")
    }
```



```
        return
    }

    print("\(users.count) user(s) found.")
    for user in users {
        print("  \(user.username ?? "<unknown>")")
    }
} catch _ as UnauthorizedException {
    print("*** Please authenticate with AWS before using this command.")
    return
} catch _ as ResourceNotFoundException {
    print("*** The specified User Pool was not found.")
    return
} catch {
    print("*** An unexpected type of error occurred.")
    return
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Swift API reference 中的 [ListUsers](#)。

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。

ResendConfirmationCode 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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

適用於 .NET 的 SDK

Note

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

```

    /// <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《適用於 .NET 的 AWS SDK API 參考》中的 [ResendConfirmationCode](#)。

C++

SDK for C++

Note

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

```
Aws::Client::ClientConfiguration clientConfig;
```

```

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

        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《適用於 C++ 的 AWS SDK API 參考》中的 [ResendConfirmationCode](#)。

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》中的[ResendConfirmationCode](#)。

Java

SDK for Java 2.x

Note

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

```
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》中的 [ResendConfirmationCode](#)。

JavaScript

SDK for JavaScript (v3)

Note

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

```
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《適用於 JavaScript 的 AWS SDK API 參考》中的 [ResendConfirmationCode](#)。

Kotlin

SDK for Kotlin

Note

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

```
suspend fun 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))
    }
}
```

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

Python

SDK for Python (Boto3)

Note

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

```

class 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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [ResendConfirmationCode](#)。

Swift

SDK for Swift

Note

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

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

/// Requests a new confirmation code be sent to the given user's contact
/// method.
///
/// - Parameters:
///   - cipClient: The `CognitoIdentityProviderClient` to use.
///   - clientId: The application client ID.
///   - userName: The user to resend a code for.
///
/// - Returns: `true` if a new code was sent successfully, otherwise
///   `false`.
func resendConfirmationCode(cipClient: CognitoIdentityProviderClient,
                           clientId: String,
                           userName: String) async -> Bool {
    do {
        let output = try await cipClient.resendConfirmationCode(
            input: ResendConfirmationCodeInput(
                clientId: clientId,
                username: userName
            )
        )

        guard let deliveryMedium = output.codeDeliveryDetails?.deliveryMedium
    else {
```



```

        print("*** Unable to get the delivery method for the resent
code.")
        return false
    }

    print("=====> A new code has been sent by \$(deliveryMedium)")
    return true
} catch {
    print("*** Unable to resend the confirmation code to user
\$(userName).")
    return false
}
}
}

```

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

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。

RespondToAuthChallenge 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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

範例 1：回應 NEW_PASSWORD_REQUIRED 挑戰

下列 respond-to-auth-challenge 範例會回應 NEW_PASSWORD_REQUIRED 傳回的 initiate-auth 挑戰。它會為使用者 設定密碼 jane@example.com。

```

aws cognito-idp respond-to-auth-challenge \
  --client-id 1example23456789 \
  --challenge-name NEW_PASSWORD_REQUIRED \

```

```
--challenge-responses USERNAME=jane@example.com,NEW_PASSWORD=[Password] \  
--session AYABeEv5Hk1EXAMPLE
```

輸出：

```
{  
  "ChallengeParameters": {},  
  "AuthenticationResult": {  
    "AccessToken": "ACCESS_TOKEN",  
    "ExpiresIn": 3600,  
    "TokenType": "Bearer",  
    "RefreshToken": "REFRESH_TOKEN",  
    "IdToken": "ID_TOKEN",  
    "NewDeviceMetadata": {  
      "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "DeviceGroupKey": "-wt2ha1Zd"  
    }  
  }  
}
```

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的[身分驗證](#)。

範例 2：回應 SELECT_MFA_TYPE 挑戰

下列respond-to-auth-challenge範例選擇 TOTP MFA 做為目前使用者的 MFA 選項。系統會提示使用者選取 MFA 類型，然後提示使用者輸入其 MFA 代碼。

```
aws cognito-idp respond-to-auth-challenge \  
  --client-id 1example23456789  
  --session AYABeEv5Hk1EXAMPLE  
  --challenge-name SELECT_MFA_TYPE  
  --challenge-responses USERNAME=testuser,ANSWER=SOFTWARE_TOKEN_MFA
```

輸出：

```
{  
  "ChallengeName": "SOFTWARE_TOKEN_MFA",  
  "Session": "AYABeEv5Hk1EXAMPLE",  
  "ChallengeParameters": {  
    "FRIENDLY_DEVICE_NAME": "transparent"  
  }  
}
```

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的[新增 MFA](#)。

範例 3：回應 Software_TOKEN_MFA 挑戰

下列respond-to-auth-challenge範例提供 TOTP MFA 程式碼並完成登入。

```
aws cognito-idp respond-to-auth-challenge \  
  --client-id 1example23456789 \  
  --session AYABeEv5Hk1EXAMPLE \  
  --challenge-name SOFTWARE_TOKEN_MFA \  
  --challenge-responses USERNAME=testuser,SOFTWARE_TOKEN_MFA_CODE=123456
```

輸出：

```
{  
  "AuthenticationResult": {  
    "AccessToken": "eyJra456defEXAMPLE",  
    "ExpiresIn": 3600,  
    "TokenType": "Bearer",  
    "RefreshToken": "eyJra123abcEXAMPLE",  
    "IdToken": "eyJra789ghiEXAMPLE",  
    "NewDeviceMetadata": {  
      "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",  
      "DeviceGroupKey": "-v7w9UcY6"  
    }  
  }  
}
```

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的[新增 MFA](#)。

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

JavaScript

SDK for JavaScript (v3)

Note

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

```
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《適用於 JavaScript 的 AWS SDK API 參考》中的 [RespondToAuthChallenge](#)。

Python

SDK for Python (Boto3)

Note

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

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_auth['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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [RespondToAuthChallenge](#)。

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。

SignUp 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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 函數自動遷移已知使用者](#)

- [使用需要 MFA 的使用者集區註冊使用者](#)

.NET

適用於 .NET 的 SDK

Note

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

```
/// <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 的詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [SignUp](#)。

C++

SDK for C++

Note

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

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

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

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》中的 [SignUp](#)。

Go

SDK for Go V2

 Note

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

```
import (
    "context"
    "errors"
    "log"

    "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"
)

type CognitoActions struct {
    CognitoClient *cognitoidentityprovider.Client
}

// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
string, password string, userEmail string) (bool, error) {
    confirmed := false
    output, err := actor.CognitoClient.SignUp(ctx,
        &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 的詳細資訊，請參閱《適用於 Go 的 AWS SDK API 參考》中的 [SignUp](#)。

Java

SDK for Java 2.x

Note

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

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

JavaScript

SDK for JavaScript (v3)

Note

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

```
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 的詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK API 參考》中的 [SignUp](#)。

Kotlin

SDK for Kotlin

 Note

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

```
suspend fun 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 的詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [SignUp](#)。

Python

SDK for Python (Boto3)

Note

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

```
class 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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [SignUp](#)。

Swift

SDK for Swift

Note

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

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

/// Create a new user in a user pool.
///
/// - Parameters:
///   - cipClient: The `CognitoIdentityProviderClient` to use.
///   - clientId: The ID of the app client to create a user for.
///   - userName: The username for the new user.
///   - password: The new user's password.
///   - email: The new user's email address.
///
/// - Returns: `true` if successful; otherwise `false`.
func signUp(cipClient: CognitoIdentityProviderClient, clientId: String,
userName: String, password: String, email: String) async -> Bool {
    let emailAttr = CognitoIdentityProviderClientTypes.AttributeType(
        name: "email",
        value: email
    )

    let userAttrsList = [emailAttr]

    do {
        _ = try await cipClient.signUp(
            input: SignUpInput(
                clientId: clientId,
                password: password,
                userAttributes: userAttrsList,
                username: userName
            )
        )
    }
}
```

```
        print("=====> User \(userName) signed up.")
    } catch _ as AWSCognitoIdentityProvider.UsernameExistsException {
        print("*** The username \(userName) already exists. Please use a
different one.")
        return false
    } catch let error as AWSCognitoIdentityProvider.InvalidPasswordException
{
        print("*** Error: The specified password is invalid. Reason:
\(error.properties.message ?? "<none available>").")
        return false
    } catch _ as AWSCognitoIdentityProvider.ResourceNotFoundException {
        print("*** Error: The specified client ID (\(clientId)) doesn't
exist.")
        return false
    } catch {
        print("*** Unexpected error: \(error)")
        return false
    }

    return true
}
```

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

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。

UpdateUserPool 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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 函數自動遷移已知使用者](#)
- [進行 Amazon Cognito 使用者身分驗證後，可使用 Lambda 函數撰寫自訂活動資料](#)

CLI

AWS CLI

更新使用者集區

下列update-user-pool範例會使用每個可用組態選項的範例語法來修改使用者集區。若要更新使用者集區，您必須指定所有先前設定的選項，否則它們會重設為預設值。

```
aws cognito-idp update-user-pool --user-pool-id us-west-2_EXAMPLE \
    --policies PasswordPolicy=
    \{MinimumLength=6,RequireUppercase=true,RequireLowercase=true,RequireNumbers=true,Require
    \
    --deletion-protection ACTIVE \
    --lambda-config PreSignUp="arn:aws:lambda:us-
west-2:123456789012:function:cognito-test-presignup-
function",PreTokenGeneration="arn:aws:lambda:us-
west-2:123456789012:function:cognito-test-pretoken-function" \
    --auto-verified-attributes "phone_number" "email" \
    --verification-message-template \{"SmsMessage\":"\"Your code is
{####}\"",\"EmailMessage\":"\"Your code is {####}\"",\"EmailSubject\":"\"Your
verification code\"",\"EmailMessageByLink\":"\"Click {##here##} to verify
your email address.\"",\"EmailSubjectByLink\":"\"Your verification link\"",
\"DefaultEmailOption\":"\"CONFIRM_WITH_LINK\""\} \
    --sms-authentication-message "Your code is {####}" \
    --user-attribute-update-settings
    AttributesRequireVerificationBeforeUpdate="email","phone_number" \
    --mfa-configuration "OPTIONAL" \
    --device-
configuration ChallengeRequiredOnNewDevice=true,DeviceOnlyRememberedOnUserPrompt=true
    \
    --email-configuration SourceArn="arn:aws:ses:us-
west-2:123456789012:identity/admin@example.com",ReplyToEmailAddress="amdin
+noreply@example.com",EmailSendingAccount=DEVELOPER,From="admin@amazon.com",Configuration
configuration-set" \
    --sms-configuration SnsCallerArn="arn:aws:iam::123456789012:role/service-
role/SNS-SMS-Role",ExternalId="12345",SnsRegion="us-west-2" \
    --admin-create-user-config
    AllowAdminCreateUserOnly=false,InviteMessageTemplate=\{SMSMessage=\\"Welcome
{username}. Your confirmation code is {####}\"",EmailMessage=\\"Welcome
{username}. Your confirmation code is {####}\"",EmailSubject=\\"Welcome to
MyMobileGame\""\} \
    --user-pool-tags "Function"="MyMobileGame","Developers"="Berlin" \
```

```
--admin-create-user-config
AllowAdminCreateUserOnly=false,InviteMessageTemplate={SMSMessage=\\\\"Welcome
{username}. Your confirmation code is {####}\\",EmailMessage=\\\\"Welcome
{username}. Your confirmation code is {####}\\",EmailSubject=\\\\"Welcome to
MyMobileGame\\"\\} \
--user-pool-add-ons AdvancedSecurityMode="AUDIT" \
--account-recovery-setting RecoveryMechanisms=
\\[{Priority=1,Name="verified_email"}\
{Priority=2,Name="verified_phone_number"}\]
```

此命令不會產生輸出。

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的[更新使用者集區組態](#)。

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

Go

SDK for Go V2

Note

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

```
import (
    "context"
    "errors"
    "log"

    "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"
)

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(ctx context.Context, userPoolId
    string, triggers ...TriggerInfo) error {
    output, err := actor.CognitoClient.DescribeUserPool(ctx,
        &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(ctx,
        &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，請參閱適用於 Go 的 AWS SDK 《API 參考》中的 [UpdateUserPool](#)。

JavaScript

SDK for JavaScript (v3)

Note

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

```

/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,

```

```
    },
  });

  const response = await cognitoClient.send(command);
  return [response, null];
} catch (err) {
  return [null, err];
}
};
```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [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。

VerifySoftwareToken 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 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

適用於 .NET 的 SDK

Note

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

```
/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [VerifySoftwareToken](#)。

C++

SDK for C++

Note

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

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

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《適用於 C++ 的 AWS SDK API 參考》中的 [VerifySoftwareToken](#)。

CLI

AWS CLI

確認註冊 TOTP 驗證器

下列 `verify-software-token` 範例會完成目前使用者的 TOTP 註冊。

```
aws cognito-idp verify-software-token \
  --access-token eyJra456defEXAMPLE \
  --user-code 123456
```

輸出：

```
{
  "Status": "SUCCESS"
}
```

如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的 [將 MFA 新增至使用者集區](#)。

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

Java

SDK for Java 2.x

Note

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

```
// 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》中的 [VerifySoftwareToken](#)。

JavaScript

SDK for JavaScript (v3)

Note

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

```
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《適用於 JavaScript 的 AWS SDK API 參考》中的 [VerifySoftwareToken](#)。

Kotlin

SDK for Kotlin

Note

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

```
// 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}")
    }
}
```

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

Python

SDK for Python (Boto3)

Note

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

```

class 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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [VerifySoftwareToken](#)。

Swift

SDK for Swift

Note

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

```
import AWSClientRuntime
import AWSCognitoIdentityProvider

/// Confirm that the user's TOTP authenticator is configured correctly by
/// sending a code to it to check that it matches successfully.
///
/// - Parameters:
///   - cipClient: The `CognitoIdentityProviderClient` to use.
///   - session: An authentication session previously returned by an
///     `associateSoftwareToken()` call.
///   - mfaCode: The 6-digit code currently displayed by the user's
///     authenticator, as provided by the user.
func verifyTOTP(cipClient: CognitoIdentityProviderClient, session: String?,
mfaCode: String?) async {
    do {
        let output = try await cipClient.verifySoftwareToken(
            input: VerifySoftwareTokenInput(
                session: session,
                userCode: mfaCode
            )
        )

        guard let tokenStatus = output.status else {
            print("*** Unable to get the token's status.")
            return
        }
    }
}
```

```
    }
    print("=====> The token's status is: \(tokenStatus)")
} catch _ as SoftwareTokenMFANotFoundException {
    print("*** The specified user pool isn't configured for MFA.")
    return
} catch _ as CodeMismatchException {
    print("*** The specified MFA code doesn't match the expected value.")
    return
} catch _ as UserNotFoundException {
    print("*** The specified username doesn't exist.")
    return
} catch _ as UserNotConfirmedException {
    print("*** The user has not been confirmed.")
    return
} catch {
    print("*** Error verifying the MFA token!")
    return
}
}
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Swift 的 SDK API 參考》中的 [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 AWS SDKs Amazon Cognito 身分提供者案例

下列程式碼範例示範如何在 Amazon Cognito Identity Provider AWS SDKs 中實作常見案例。這些案例說明如何透過呼叫 Amazon Cognito Identity Provider 內的多個函數或與其他函數結合，來完成特定任務 AWS 服務。每個案例均包含完整原始碼的連結，您可在連結中找到如何設定和執程式碼的相關指示。

案例的目標是獲得中等水平的經驗，協助您了解內容中的服務動作。

範例

- [使用 SDK 透過 Lambda 函數自動確認已知的 Amazon Cognito 使用者 AWS](#)
- [使用 SDK 透過 Lambda 函數自動遷移已知的 Amazon Cognito 使用者 AWS](#)
- [使用 SDK 向需要 MFA 的 Amazon Cognito 使用者集區註冊使用者 AWS](#)

- [使用 AWS SDK 進行 Amazon Cognito 使用者身分驗證後，使用 Lambda 函數寫入自訂活動資料](#)

使用 SDK 透過 Lambda 函數自動確認已知的 Amazon Cognito 使用者 AWS

下列程式碼範例示範如何使用 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 觸發條件的 Lambda 函數。
- 使用 Amazon Cognito 註冊使用者。
- Lambda 函數會掃描 DynamoDB 資料表，並自動確認已知使用者。
- 以新使用者身分登入，然後清除資源。

Go

SDK for Go V2

Note

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

在命令提示中執行互動式案例。

```
import (
    "context"
    "errors"
    "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"
)

// 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(ctx context.Context, 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(
        ctx, 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(ctx context.Context, 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(ctx, 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(ctx, 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(ctx context.Context, 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(ctx, 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(ctx context.Context, stackName string) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            runner.resources.Cleanup(ctx)
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Printf("Welcome\n")

    log.Println(strings.Repeat("-", 88))

    stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
    if err != nil {
        panic(err)
    }
    runner.resources.userPoolId = stackOutputs["UserPoolId"]
    runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])

    runner.AddPreSignUpTrigger(ctx, stackOutputs["UserPoolId"],
        stackOutputs["AutoConfirmFunctionArn"])
    runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
    userName, password := runner.SignUpUser(ctx, stackOutputs["UserPoolClientId"],
        stackOutputs["TableName"])
    runner.helper.ListRecentLogEvents(ctx, stackOutputs["AutoConfirmFunction"])
    runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
        runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password))
}

```

```
runner.resources.Cleanup(ctx)

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

使用 Lambda 函數來處理 PreSignUp 觸發條件。

```
import (
    "context"
    "log"
    "os"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

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() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        log.Panicln(err)
    }
    h := handler{
        dynamoClient: dynamodb.NewFromConfig(sdkConfig),
    }
    lambda.Start(h.HandleRequest)
}

```

建立執行一般任務的 struct。

```

import (
    "context"
    "log"
    "strings"
    "time"
    "user_pools_and_lambda_triggers/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

```

```
// IScenarioHelper defines common functions used by the workflows in this
// example.
type IScenarioHelper interface {
    Pause(secs int)
    GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
        error)
    PopulateUserTable(ctx context.Context, tableName string)
    GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
    AddKnownUser(ctx context.Context, tableName string, user actions.User)
    ListRecentLogEvents(ctx context.Context, 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(ctx context.Context, stackName
    string) (actions.StackOutputs, error) {
    return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, 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(ctx, tableName)
    if err != nil {
        panic(err)
    }
}

// GetKnownUsers gets the users from the known users table in a structured
// format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
    (actions.UserList, error) {
    knownUsers, err := helper.dynamoActor.Scan(ctx, 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(ctx context.Context, 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(ctx, 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(ctx context.Context,
    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(ctx, functionName)
    if err != nil {
        panic(err)
    }
    log.Printf("Getting some recent events from log stream %v\n",
        *logStream.LogStreamName)
    events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
        *logStream.LogStreamName, 10)
    if err != nil {
        panic(err)
    }
    for _, event := range events {
        log.Printf("\t\t%v", *event.Message)
    }
    log.Println(strings.Repeat("-", 88))
}
```

建立包裝 Amazon Cognito 動作的 struct。

```
import (
    "context"
    "errors"
    "log"

    "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"
)

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(ctx context.Context, userPoolId
string, triggers ...TriggerInfo) error {
    output, err := actor.CognitoClient.DescribeUserPool(ctx,
&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(ctx,
&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(ctx context.Context, clientId string, userName
string, password string, userEmail string) (bool, error) {
    confirmed := false
    output, err := actor.CognitoClient.SignUp(ctx,
&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(ctx context.Context, clientId string, userName
string, password string) (*types.AuthenticationResultType, error) {
    var authResult *types.AuthenticationResultType
    output, err := actor.CognitoClient.InitiateAuth(ctx,
        &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(ctx context.Context, clientId string,
userName string) (*types.CodeDeliveryDetailsType, error) {
    output, err := actor.CognitoClient.ForgotPassword(ctx,
        &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(ctx context.Context, clientId
string, code string, userName string, password string) error {
_, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
&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(ctx context.Context, userAccessToken
string) error {
_, err := actor.CognitoClient.DeleteUser(ctx,
&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(ctx context.Context, userPoolId
string, userName string, userEmail string) error {
```

```

_, err := actor.CognitoClient.AdminCreateUser(ctx,
&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(ctx context.Context, userPoolId
string, userName string, password string) error {
_, err := actor.CognitoClient.AdminSetUserPassword(ctx,
&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 struct。

```
import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// 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(ctx context.Context, 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(ctx, &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(ctx context.Context, tableName string) (UserList,
error) {
    var userList UserList
    output, err := actor.DynamoClient.Scan(ctx, &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(ctx context.Context, 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(ctx, &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 Logs 動作的 struct。

```

import (
    "context"
    "fmt"
    "log"

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

```

```

)

type CloudWatchLogsActions struct {
    CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
    functionName string) (types.LogStream, error) {
    var logStream types.LogStream
    logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
    output, err := actor.CwlClient.DescribeLogStreams(ctx,
        &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(ctx context.Context, 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(ctx,
        &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
}

```

建立包裝 AWS CloudFormation 動作的結構。

```

import (
    "context"
    "log"

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

// 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(ctx context.Context, stackName
    string) StackOutputs {
    output, err := actor.CfnClient.DescribeStacks(ctx,
        &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
}

```

清除資源。

```
import (
    "context"
    "log"
    "user_pools_and_lambda_triggers/actions"

    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// 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(ctx context.Context) {
    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(ctx, 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(ctx, 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《適用於 Go 的 AWS SDK API 參考》中的下列主題。
 - [DeleteUser](#)
 - [InitiateAuth](#)
 - [SignUp](#)
 - [UpdateUserPool](#)

JavaScript

SDK for JavaScript (v3)

Note

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

設定互動式 "Scenario" 執行。JavaScript (v3) 範例會共用案例執行器，以簡化複雜的範例。完整的原始程式碼位於 GitHub。

```
import { AutoConfirm } from "./scenario-auto-confirm.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {
  errors: [],
  users: [
    {
      UserName: "test_user_1",
      Email: "test_email_1@example.com",
    },
    {
      UserName: "test_user_2",
      Email: "test_email_2@example.com",
    },
    {
      UserName: "test_user_3",
      Email: "test_email_3@example.com",
    },
  ],
};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 * class
 * that simplifies running a series of steps.
 */
```

```
export const scenarios = {
  // Demonstrate automatically confirming known users in a database.
  "auto-confirm": AutoConfirm(context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { parseScenarioArgs } from "@aws-doc-sdk-examples/lib/scenario/index.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Cognito user pools and triggers",
    description:
      "Demonstrate how to use the AWS SDKs to customize Amazon Cognito authentication behavior.",
  });
}
```

此案例展示如何自動確認已知使用者。它會協調範例步驟。

```
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";

import {
  getStackOutputs,
  logCleanUpReminder,
  promptForStackName,
  promptForStackRegion,
  skipWhenErrors,
} from "./steps-common.js";
import { populateTable } from "./actions/dynamodb-actions.js";
import {
  addPreSignUpHandler,
  deleteUser,
  getUser,
  signIn,
  signUpUser,
```

```

} from "./actions/cognito-actions.js";
import {
  getLatestLogStreamForLambda,
  getLogEvents,
} from "./actions/cloudwatch-logs-actions.js";

/**
 * @typedef {{
 *   errors: Error[],
 *   password: string,
 *   users: { Username: string, UserEmail: string }[],
 *   selectedUser?: string,
 *   stackName?: string,
 *   stackRegion?: string,
 *   token?: string,
 *   confirmDeleteSignedInUser?: boolean,
 *   TableName?: string,
 *   UserPoolClientId?: string,
 *   UserPoolId?: string,
 *   UserPoolArn?: string,
 *   AutoConfirmHandlerArn?: string,
 *   AutoConfirmHandlerName?: string
 * }} State
 */

const greeting = new ScenarioOutput(
  "greeting",
  (/** @type {State} */ state) => `This demo will populate some users into the \
database created as part of the "${state.stackName}" stack. \
Then the AutoConfirmHandler will be linked to the PreSignUp \
trigger from Cognito. Finally, you will choose a user to sign up.` ,
  { skipWhen: skipWhenErrors },
);

const logPopulatingUsers = new ScenarioOutput(
  "logPopulatingUsers",
  "Populating the DynamoDB table with some users.",
  { skipWhenErrors: skipWhenErrors },
);

const logPopulatingUsersComplete = new ScenarioOutput(
  "logPopulatingUsersComplete",
  "Done populating users.",
  { skipWhen: skipWhenErrors },
);

```



```
);

const populateUsers = new ScenarioAction(
  "populateUsers",
  async (** @type {State} */ state) => {
    const [_, err] = await populateTable({
      region: state.stackRegion,
      tableName: state.TableName,
      items: state.users,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTrigger = new ScenarioOutput(
  "logSetupSignUpTrigger",
  "Setting up the PreSignUp trigger for the Cognito User Pool.",
  { skipWhen: skipWhenErrors },
);

const setupSignUpTrigger = new ScenarioAction(
  "setupSignUpTrigger",
  async (** @type {State} */ state) => {
    const [_, err] = await addPreSignUpHandler({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      handlerArn: state.AutoConfirmHandlerArn,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTriggerComplete = new ScenarioOutput(
  "logSetupSignUpTriggerComplete",
```

```

(
  /** @type {State} */ state,
) => `The lambda function "${state.AutoConfirmHandlerName}" \
has been configured as the PreSignUp trigger handler for the user pool
"${state.UserPoolId}".`,
{ skipWhen: skipWhenErrors },
);

const selectUser = new ScenarioInput(
  "selectedUser",
  "Select a user to sign up.",
  {
    type: "select",
    choices: (/** @type {State} */ state) => state.users.map((u) => u.UserName),
    skipWhen: skipWhenErrors,
    default: (/** @type {State} */ state) => state.users[0].UserName,
  },
);

const checkIfUserAlreadyExists = new ScenarioAction(
  "checkIfUserAlreadyExists",
  async (/** @type {State} */ state) => {
    const [user, err] = await getUser({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      username: state.selectedUser,
    });

    if (err?.name === "UserNotFoundException") {
      // Do nothing. We're not expecting the user to exist before
      // sign up is complete.
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    if (user) {
      state.errors.push(
        new Error(
          `The user "${state.selectedUser}" already exists in the user pool
"${state.UserPoolId}".`,

```

```

        ),
    );
}
},
{
    skipWhen: skipWhenErrors,
},
);

const createPassword = new ScenarioInput(
    "password",
    "Enter a password that has at least eight characters, uppercase, lowercase, numbers and symbols.",
    { type: "password", skipWhen: skipWhenErrors, default: "Abcd1234!" },
);

const logSignUpExistingUser = new ScenarioOutput(
    "logSignUpExistingUser",
    (/** @type {State} */ state) => `Signing up user "${state.selectedUser}".`,
    { skipWhen: skipWhenErrors },
);

const signUpExistingUser = new ScenarioAction(
    "signUpExistingUser",
    async (/** @type {State} */ state) => {
        const signUp = (password) =>
            signUpUser({
                region: state.stackRegion,
                userPoolClientId: state.UserPoolClientId,
                username: state.selectedUser,
                email: state.users.find((u) => u.UserName === state.selectedUser)
                    .UserEmail,
                password,
            });

        let [_, err] = await signUp(state.password);

        while (err?.name === "InvalidPasswordException") {
            console.warn("The password you entered was invalid.");
            await createPassword.handle(state);
            [_, err] = await signUp(state.password);
        }

        if (err) {

```

```
        state.errors.push(err);
    }
},
{ skipWhen: skipWhenErrors },
);

const logSignUpExistingUserComplete = new ScenarioOutput(
    "logSignUpExistingUserComplete",
    (/** @type {State} */ state) =>
        `${state.selectedUser} was signed up successfully.`,
    { skipWhen: skipWhenErrors },
);

const logLambdaLogs = new ScenarioAction(
    "logLambdaLogs",
    async (/** @type {State} */ state) => {
        console.log(
            "Waiting a few seconds to let Lambda write to CloudWatch Logs...\n",
        );
        await wait(10);

        const [logStream, logStreamErr] = await getLatestLogStreamForLambda({
            functionName: state.AutoConfirmHandlerName,
            region: state.stackRegion,
        });
        if (logStreamErr) {
            state.errors.push(logStreamErr);
            return;
        }

        console.log(
            `Getting some recent events from log stream "${logStream.logStreamName}"`,
        );
        const [logEvents, logEventsErr] = await getLogEvents({
            functionName: state.AutoConfirmHandlerName,
            region: state.stackRegion,
            eventCount: 10,
            logStreamName: logStream.logStreamName,
        });
        if (logEventsErr) {
            state.errors.push(logEventsErr);
            return;
        }
    }
);
```

```
    console.log(logEvents.map((ev) => `\t${ev.message}`).join(""));
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUser = new ScenarioOutput(
  "logSignInUser",
  (/** @type {State} */ state) => `Let's sign in as ${state.selectedUser}`,
  { skipWhen: skipWhenErrors },
);

const signInUser = new ScenarioAction(
  "signInUser",
  async (/** @type {State} */ state) => {
    const [response, err] = await signIn({
      region: state.stackRegion,
      clientId: state.UserPoolClientId,
      username: state.selectedUser,
      password: state.password,
    });

    if (err?.name === "PasswordResetRequiredException") {
      state.errors.push(new Error("Please reset your password."));
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    state.token = response?.AuthenticationResult?.AccessToken;
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUserComplete = new ScenarioOutput(
  "logSignInUserComplete",
  (/** @type {State} */ state) =>
    `Successfully signed in. Your access token starts with:
    ${state.token.slice(0, 11)}`,
  { skipWhen: skipWhenErrors },
);
```

```

const confirmDeleteSignedInUser = new ScenarioInput(
  "confirmDeleteSignedInUser",
  "Do you want to delete the currently signed in user?",
  { type: "confirm", skipWhen: skipWhenErrors },
);

const deleteSignedInUser = new ScenarioAction(
  "deleteSignedInUser",
  async (/** @type {State} */ state) => {
    const [_, err] = await deleteUser({
      region: state.stackRegion,
      accessToken: state.token,
    });

    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      skipWhenErrors(state) || !state.confirmDeleteSignedInUser,
  },
);

const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State} */ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
);

export const AutoConfirm = (context) =>
  new Scenario(
    "AutoConfirm",
    [
      promptForStackName,
      promptForStackRegion,
    ],
  );

```

```

    getStackOutputs,
    greeting,
    logPopulatingUsers,
    populateUsers,
    logPopulatingUsersComplete,
    logSetupSignUpTrigger,
    setupSignUpTrigger,
    logSetupSignUpTriggerComplete,
    selectUser,
    checkIfUserAlreadyExists,
    createPassword,
    logSignUpExistingUser,
    signUpExistingUser,
    logSignUpExistingUserComplete,
    logLambdaLogs,
    logSignInUser,
    signInUser,
    logSignInUserComplete,
    confirmDeleteSignedInUser,
    deleteSignedInUser,
    logCleanUpReminder,
    logErrors,
  ],
  context,
);

```

這些是與其他案例共用的步驟。

```

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { getCfnOutputs } from "@aws-doc-sdk-examples/lib/sdk/cfn-outputs.js";

export const skipWhenErrors = (state) => state.errors.length > 0;

export const getStackOutputs = new ScenarioAction(
  "getStackOutputs",
  async (state) => {
    if (!state.stackName || !state.stackRegion) {
      state.errors.push(

```

```

        new Error(
            "No stack name or region provided. The stack name and \
region are required to fetch CFN outputs relevant to this example.",
        ),
    );
    return;
}

    const outputs = await getCfnOutputs(state.stackName, state.stackRegion);
    Object.assign(state, outputs);
},
);

export const promptForStackName = new ScenarioInput(
    "stackName",
    "Enter the name of the stack you deployed earlier.",
    { type: "input", default: "PoolsAndTriggersStack" },
);

export const promptForStackRegion = new ScenarioInput(
    "stackRegion",
    "Enter the region of the stack you deployed earlier.",
    { type: "input", default: "us-east-1" },
);

export const logCleanUpReminder = new ScenarioOutput(
    "logCleanUpReminder",
    "All done. Remember to run 'cdk destroy' to teardown the stack.",
    { skipWhen: skipWhenErrors },
);

```

具有 Lambda 函數之 PreSignUp 觸發條件的處理常式。

```

import type { PreSignUpTriggerEvent, Handler } from "aws-lambda";
import type { UserRepository } from "../user-repository";
import { DynamoDBUserRepository } from "../user-repository";

export class PreSignUpHandler {
    private userRepository: UserRepository;

    constructor(userRepository: UserRepository) {
        this.userRepository = userRepository;
    }
}

```



```
}

private isPreSignUpTriggerSource(event: PreSignUpTriggerEvent): boolean {
  return event.triggerSource === "PreSignUp_SignUp";
}

private getEventUserEmail(event: PreSignUpTriggerEvent): string {
  return event.request.userAttributes.email;
}

async handlePreSignUpTriggerEvent(
  event: PreSignUpTriggerEvent,
): Promise<PreSignUpTriggerEvent> {
  console.log(
    `Received presignup from ${event.triggerSource} for user
'${event.userName}'`,
  );

  if (!this.isPreSignUpTriggerSource(event)) {
    return event;
  }

  const eventEmail = this.getEventUserEmail(event);
  console.log(`Looking up email ${eventEmail}.`);
  const storedUserInfo =
    await this.userRepository.getUserInfoByEmail(eventEmail);

  if (!storedUserInfo) {
    console.log(
      `Email ${eventEmail} not found. Email verification is required.`,
    );
    return event;
  }

  if (storedUserInfo.UserName !== event.userName) {
    console.log(
      `UserEmail ${eventEmail} found, but stored UserName
'${storedUserInfo.UserName}' does not match supplied UserName
'${event.userName}'. Verification is required.`,
    );
  } else {
    console.log(
      `UserEmail ${eventEmail} found with matching UserName
${storedUserInfo.UserName}. User is confirmed.`,
    );
  }
}
```

```

    );
    event.response.autoConfirmUser = true;
    event.response.autoVerifyEmail = true;
  }
  return event;
}
}

const createPreSignUpHandler = (): PreSignUpHandler => {
  const tableName = process.env.TABLE_NAME;
  if (!tableName) {
    throw new Error("TABLE_NAME environment variable is not set");
  }

  const userRepository = new DynamoDBUserRepository(tableName);
  return new PreSignUpHandler(userRepository);
};

export const handler: Handler = async (event: PreSignUpTriggerEvent) => {
  const preSignUpHandler = createPreSignUpHandler();
  return preSignUpHandler.handlePreSignUpTriggerEvent(event);
};

```

CloudWatch Logs 動作的模組。

```

import {
  CloudWatchLogsClient,
  GetLogEventsCommand,
  OrderBy,
  paginateDescribeLogStreams,
} from "@aws-sdk/client-cloudwatch-logs";

/**
 * Get the latest log stream for a Lambda function.
 * @param {{ functionName: string, region: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").LogStream | null,
  unknown]>}
 */
export const getLatestLogStreamForLambda = async ({ functionName, region }) => {
  try {
    const logGroupName = `/aws/lambda/${functionName}`;

```

```

const cwlClient = new CloudWatchLogsClient({ region });
const paginator = paginateDescribeLogStreams(
  { client: cwlClient },
  {
    descending: true,
    limit: 1,
    orderBy: OrderBy.LastEventTime,
    logGroupName,
  },
);

for await (const page of paginator) {
  return [page.logStreams[0], null];
}
} catch (err) {
  return [null, err];
}
};

/**
 * Get the log events for a Lambda function's log stream.
 * @param {{
 *   functionName: string,
 *   logStreamName: string,
 *   eventCount: number,
 *   region: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").OutputLogEvent[]
 * | null, unknown]>}
 */
export const getLogEvents = async ({
  functionName,
  logStreamName,
  eventCount,
  region,
}) => {
  try {
    const cwlClient = new CloudWatchLogsClient({ region });
    const logGroupName = `aws/lambda/${functionName}`;
    const response = await cwlClient.send(
      new GetLogEventsCommand({
        logStreamName: logStreamName,
        limit: eventCount,
        logGroupName: logGroupName,

```

```
    }),  
    );  
  
    return [response.events, null];  
  } catch (err) {  
    return [null, err];  
  }  
};
```

Amazon Cognito 動作的模組。

```
import {  
  AdminGetUserCommand,  
  CognitoIdentityProviderClient,  
  DeleteUserCommand,  
  InitiateAuthCommand,  
  SignUpCommand,  
  UpdateUserPoolCommand,  
} from "@aws-sdk/client-cognito-identity-provider";  
  
/**  
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool  
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config  
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-provider").UpdateUserPoolCommandOutput | null, unknown]>}  
 */  
export const addPreSignUpHandler = async ({  
  region,  
  userPoolId,  
  handlerArn,  
}) => {  
  try {  
    const cognitoClient = new CognitoIdentityProviderClient({  
      region,  
    });  
  
    const command = new UpdateUserPoolCommand({  
      UserPoolId: userPoolId,  
      LambdaConfig: {  
        PreSignUp: handlerArn,  
      },  
    });
```

```
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Attempt to register a user to a user pool with a given username and password.
 * @param {{
 *   region: string,
 *   userPoolClientId: string,
 *   username: string,
 *   email: string,
 *   password: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-provider").SignUpCommandOutput | null, unknown]>}
 */
export const signUpUser = async ({
  region,
  userPoolClientId,
  username,
  email,
  password,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const response = await cognitoClient.send(
      new SignUpCommand({
        ClientId: userPoolClientId,
        Username: username,
        Password: password,
        UserAttributes: [{ Name: "email", Value: email }],
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

```

    }
};

/**
 * Sign in a user to Amazon Cognito using a username and password authentication
 * flow.
 * @param {{ region: string, clientId: string, username: string, password:
 * string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
 * provider").InitiateAuthCommandOutput | null, unknown]>}
 */
export const signIn = async ({ region, clientId, username, password }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new InitiateAuthCommand({
        AuthFlow: "USER_PASSWORD_AUTH",
        ClientId: clientId,
        AuthParameters: { USERNAME: username, PASSWORD: password },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Retrieve an existing user from a user pool.
 * @param {{ region: string, userPoolId: string, username: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
 * provider").AdminGetUserCommandOutput | null, unknown]>}
 */
export const getUser = async ({ region, userPoolId, username }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new AdminGetUserCommand({
        UserPoolId: userPoolId,
        Username: username,
      }),
    );
    return [response, null];
  } catch (err) {

```

```

    return [null, err];
  }
};

/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

```

DynamoDB 動作的模組。

```

import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

/**
 * Populate a DynamoDB table with provide items.
 * @param {{ region: string, tableName: string, items: Record<string,
unknown>[] }} config
 * @returns {Promise<[import("@aws-sdk/lib-dynamodb").BatchWriteCommandOutput |
null, unknown]>}
 */
export const populateTable = async ({ region, tableName, items }) => {
  try {
    const ddbClient = new DynamoDBClient({ region });

```

```
const docClient = DynamoDBDocumentClient.from(ddbClient);
const response = await docClient.send(
  new BatchWriteCommand({
    RequestItems: {
      [tableName]: items.map((item) => ({
        PutRequest: {
          Item: item,
        },
      })),
    },
  });
return [response, null];
} catch (err) {
  return [null, err];
}
};
```

- 如需 API 詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK API 參考》中的下列主題。
 - [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 SDK 透過 Lambda 函數自動遷移已知的 Amazon Cognito 使用者 AWS

下列程式碼範例示範如何使用 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 觸發條件的 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 資料表，並自動將已知使用者遷移至使用者集區。
- 執行忘記密碼流程，重設已遷移使用者的密碼。
- 以新使用者身分登入，然後清除資源。

Go

SDK for Go V2

 Note

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

在命令提示中執行互動式案例。

```
import (
    "context"
    "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(ctx context.Context, 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(
        ctx, 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(ctx context.Context, 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(ctx, 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(ctx, 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(ctx context.Context, 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(ctx, 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(ctx, 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(ctx, 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(ctx context.Context, stackName string) {
defer func() {
if r := recover(); r != nil {
log.Println("Something went wrong with the demo.")
runner.resources.Cleanup(ctx)
}
}()

log.Println(strings.Repeat("-", 88))
log.Printf("Welcome\n")

log.Println(strings.Repeat("-", 88))

stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
if err != nil {
panic(err)
}
runner.resources.userPoolId = stackOutputs["UserPoolId"]

runner.AddMigrateUserTrigger(ctx, stackOutputs["UserPoolId"],
stackOutputs["MigrateUserFunctionArn"])
runner.resources.triggers = append(runner.resources.triggers,
actions.UserMigration)
resetNeeded, user := runner.SignInUser(ctx, stackOutputs["TableName"],
stackOutputs["UserPoolClientId"])
if resetNeeded {
runner.helper.ListRecentLogEvents(ctx, stackOutputs["MigrateUserFunction"])
runner.ResetPassword(ctx, stackOutputs["UserPoolClientId"], user)
}

runner.resources.Cleanup(ctx)

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

```

使用 Lambda 函數來處理 MigrateUser 觸發條件。

```
import (
    "context"
    "log"
    "os"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
)

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"`
    Email string `dynamodbav:"Email"`
}

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 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() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        log.Panicln(err)
    }
    h := handler{
        dynamoClient: dynamodb.NewFromConfig(sdkConfig),
    }
    lambda.Start(h.HandleRequest)
}

```

建立執行一般任務的 struct。

```

import (
    "context"
    "log"
    "strings"
    "time"
    "user_pools_and_lambda_triggers/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
// example.
type IScenarioHelper interface {
    Pause(secs int)
}

```



```

    GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
    error)
    PopulateUserTable(ctx context.Context, tableName string)
    GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
    AddKnownUser(ctx context.Context, tableName string, user actions.User)
    ListRecentLogEvents(ctx context.Context, 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(ctx context.Context, stackName
    string) (actions.StackOutputs, error) {

```

```
    return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, 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(ctx, tableName)
    if err != nil {
        panic(err)
    }
}

// GetKnownUsers gets the users from the known users table in a structured
format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
(actions.UserList, error) {
    knownUsers, err := helper.dynamoActor.Scan(ctx, 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(ctx context.Context, 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(ctx, 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(ctx context.Context,
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(ctx, functionName)
if err != nil {
    panic(err)
}
log.Printf("Getting some recent events from log stream %v\n",
*logStream.LogStreamName)
events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
*logStream.LogStreamName, 10)
if err != nil {
    panic(err)
}
for _, event := range events {
    log.Printf("\t%v", *event.Message)
}
log.Println(strings.Repeat("-", 88))
}

```

建立包裝 Amazon Cognito 動作的 struct。

```

import (
    "context"
    "errors"
    "log"

    "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"
)

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(ctx context.Context, userPoolId
    string, triggers ...TriggerInfo) error {
    output, err := actor.CognitoClient.DescribeUserPool(ctx,
        &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(ctx,
        &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(ctx context.Context, clientId string, userName
string, password string, userEmail string) (bool, error) {
    confirmed := false
    output, err := actor.CognitoClient.SignUp(ctx,
    &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(ctx context.Context, clientId string, userName
string, password string) (*types.AuthenticationResultType, error) {
    var authResult *types.AuthenticationResultType
    output, err := actor.CognitoClient.InitiateAuth(ctx,
    &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(ctx context.Context, clientId string,
    userName string) (*types.CodeDeliveryDetailsType, error) {
    output, err := actor.CognitoClient.ForgotPassword(ctx,
        &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(ctx context.Context, clientId
    string, code string, userName string, password string) error {
    _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
        &cognitoidentityprovider.ConfirmForgotPasswordInput{
            ClientId:      aws.String(clientId),
            ConfirmationCode: aws.String(code),
            Password:      aws.String(password),
        })
    return err
}

```

```

    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(ctx context.Context, userAccessToken
string) error {
  _, err := actor.CognitoClient.DeleteUser(ctx,
&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(ctx context.Context, userPoolId
string, userName string, userEmail string) error {
  _, err := actor.CognitoClient.AdminCreateUser(ctx,
&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(ctx context.Context, userPoolId
string, userName string, password string) error {
    _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
    &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 struct。

```
import (
```



```
"context"
"fmt"
"log"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
"github.com/aws/aws-sdk-go-v2/service/dynamodb"
"github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// 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(ctx context.Context, 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(ctx, &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(ctx context.Context, tableName string) (UserList,
    error) {
    var userList UserList
    output, err := actor.DynamoClient.Scan(ctx, &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(ctx context.Context, 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(ctx, &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 Logs 動作的 struct。

```

import (
    "context"
    "fmt"
    "log"

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

type CloudWatchLogsActions struct {
    CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
    functionName string) (types.LogStream, error) {

```

```

var logStream types.LogStream
logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
output, err := actor.CwlClient.DescribeLogStreams(ctx,
&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(ctx context.Context, 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(ctx,
&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
}

```

建立包裝 AWS CloudFormation 動作的結構。

```

import (
    "context"
    "log"

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

// 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(ctx context.Context, stackName
    string) StackOutputs {
    output, err := actor.CfnClient.DescribeStacks(ctx,
        &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
}

```

清除資源。

```

import (
    "context"
    "log"

```

```

"user_pools_and_lambda_triggers/actions"

"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// 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(ctx context.Context) {
    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(ctx, 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(ctx, 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《適用於 Go 的 AWS SDK 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 SDK 向需要 MFA 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

適用於 .NET 的 SDK

Note

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

```
namespace 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.WriteLine("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($"Confirmation code sent to {userName}.");
Console.WriteLine("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.WriteLine("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.WriteLine("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.WriteLine("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《適用於 .NET 的 AWS SDK API 參考》中的下列主題。
 - [AdminGetUser](#)
 - [AdminInitiateAuth](#)
 - [AdminRespondToAuthChallenge](#)
 - [AssociateSoftwareToken](#)
 - [ConfirmDevice](#)
 - [ConfirmSignUp](#)
 - [InitiateAuth](#)
 - [ListUsers](#)
 - [ResendConfirmationCode](#)
 - [RespondToAuthChallenge](#)
 - [SignUp](#)

- [VerifySoftwareToken](#)

C++

SDK for C++

 Note

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

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

//! 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《適用於 C++ 的 AWS SDK API 參考》中的下列主題。

- [AdminGetUser](#)
- [AdminInitiateAuth](#)
- [AdminRespondToAuthChallenge](#)
- [AssociateSoftwareToken](#)
- [ConfirmDevice](#)
- [ConfirmSignUp](#)
- [InitiateAuth](#)
- [ListUsers](#)
- [ResendConfirmationCode](#)
- [RespondToAuthChallenge](#)
- [SignUp](#)
- [VerifySoftwareToken](#)

Java

SDK for Java 2.x

Note

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

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.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

SDK for JavaScript (v3)

Note

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

為了獲得最佳體驗，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 { logger } 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);
    logger.log("Signing up.");
    await signUp({ clientId, username, password, email });
    logger.log(`Signed up. A confirmation email has been sent to: ${email}.`);
    logger.log(
      `Run 'confirm-sign-up ${username} <code>' to confirm your account.`
    );
  } catch (err) {
    logger.error(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 { logger } 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);
    logger.log("Confirming user.");
    await confirmSignUp({ clientId, username, code });
    logger.log(
      `User confirmed. Run 'admin-initiate-auth ${username} <password>' to sign
in.` ,
    );
  } catch (err) {
    logger.error(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 { logger } 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);

    logger.log("Signing in.");
    const { ChallengeName, Session } = await adminInitiateAuth({
      clientId,
      userPoolId,
      username,
      password,
    });

    if (ChallengeName === "MFA_SETUP") {
      logger.log("MFA setup is required.");
      return handleMfaSetup(Session, username);
    }
  }
};
```

```
    if (ChallengeName === "SOFTWARE_TOKEN_MFA") {
      handleSoftwareTokenMfa(Session);
      logger.log(`Run 'admin-respond-to-auth-challenge ${username} <totp>'`);
    }
  } catch (err) {
    logger.error(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 { logger } 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);  
  
    logger.log("Successfully authenticated.");  
  } catch (err) {  
    logger.error(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 { logger } 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);

    logger.log("Verifying TOTP.");
    await verifySoftwareToken(totp);
    logger.log("TOTP Verified. Run 'admin-initiate-auth' again to sign-in.");
  } catch (err) {
    logger.error(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《適用於 JavaScript 的 AWS SDK API 參考》中的下列主題。
 - [AdminGetUser](#)
 - [AdminInitiateAuth](#)
 - [AdminRespondToAuthChallenge](#)
 - [AssociateSoftwareToken](#)
 - [ConfirmDevice](#)
 - [ConfirmSignUp](#)
 - [InitiateAuth](#)
 - [ListUsers](#)
 - [ResendConfirmationCode](#)
 - [RespondToAuthChallenge](#)
 - [SignUp](#)
 - [VerifySoftwareToken](#)

Kotlin

SDK for Kotlin

Note

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

```
/**
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 in0b = Scanner(System.`in`)
val userName = in0b.nextLine()
println(userName)

println("**** Enter your password")
val password: String = in0b.nextLine()

println("**** Enter your email")
val email = in0b.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 = in0b.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 = in0b.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 = in0b.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 = in0b.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 challengeResponses0b = mutableMapOf<String, String>()
    challengeResponses0b["USERNAME"] = userName
    challengeResponses0b["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponses0b
            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 SDK for Kotlin API reference 中的下列主題。

- [AdminGetUser](#)
- [AdminInitiateAuth](#)
- [AdminRespondToAuthChallenge](#)
- [AssociateSoftwareToken](#)
- [ConfirmDevice](#)
- [ConfirmSignUp](#)
- [InitiateAuth](#)
- [ListUsers](#)
- [ResendConfirmationCode](#)
- [RespondToAuthChallenge](#)
- [SignUp](#)
- [VerifySoftwareToken](#)

Python

SDK for Python (Boto3)

Note

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

建立包裝案例中所用 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_auth['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 AWS SDK for Python (Boto3) API Reference 中的下列主題。

- [AdminGetUser](#)
- [AdminInitiateAuth](#)
- [AdminRespondToAuthChallenge](#)
- [AssociateSoftwareToken](#)
- [ConfirmDevice](#)
- [ConfirmSignUp](#)
- [InitiateAuth](#)
- [ListUsers](#)
- [ResendConfirmationCode](#)
- [RespondToAuthChallenge](#)
- [SignUp](#)
- [VerifySoftwareToken](#)

Swift

SDK for Swift

Note

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

Package.swift 檔。

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "http-config",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/aws-labs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
        // or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "cognito-scenario",
```

```

        dependencies: [
            .product(name: "AWSS3", package: "aws-sdk-swift"),
            .product(name: "ArgumentParser", package: "swift-argument-
parser")
        ],
        path: "Sources")
    ]
)

```

Swift 程式碼檔案。

```

// An example demonstrating various features of Amazon Cognito. Before running
// this Swift 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 example performs the following functions:
//
// 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.

```



```

import ArgumentParser
import Foundation

import AWSClientRuntime
import AWSCognitoIdentityProvider

struct ExampleCommand: ParsableCommand {
    @Argument(help: "The application clientId.")
    var clientId: String
    @Argument(help: "The user pool ID to use.")
    var poolId: String
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "cognito-scenario",
        abstract: ""
        Demonstrates various features of Amazon Cognito.
        "",
        discussion: ""
        ""
    )

    /// Prompt for an input string of at least a minimum length.
    ///
    /// - Parameters:
    ///   - prompt: The prompt string to display.
    ///   - minLength: The minimum number of characters to allow in the
    ///     response. Default value is 0.
    ///
    /// - Returns: The entered string.
    func stringRequest(_ prompt: String, minLength: Int = 1) -> String {
        while true {
            print(prompt, terminator: "")
            let str = readLine()

            guard let str else {
                continue
            }
            if str.count >= minLength {
                return str
            } else {
                print("*** Response must be at least \(minLength) character(s)
long.")
            }
        }
    }
}

```

```

    }
  }
}

/// Ask a yes/no question.
///
/// - Parameter prompt: A prompt string to print.
///
/// - Returns: `true` if the user answered "Y", otherwise `false`.
func yesNoRequest(_ prompt: String) -> Bool {
    while true {
        let answer = stringRequest(prompt).lowercased()
        if answer == "y" || answer == "n" {
            return answer == "y"
        }
    }
}

/// Get information about a specific user in a user pool.
///
/// - Parameters:
///   - cipClient: The Amazon Cognito Identity Provider client to use.
///   - userName: The user to retrieve information about.
///   - userPoolId: The user pool to search for the specified user.
///
/// - Returns: `true` if the user's information was successfully
///   retrieved. Otherwise returns `false`.
func adminGetUser(cipClient: CognitoIdentityProviderClient, userName: String,
                  userPoolId: String) async -> Bool {
    do {
        let output = try await cipClient.adminGetUser(
            input: AdminGetUserInput(
                userPoolId: userPoolId,
                username: userName
            )
        )

        guard let userStatus = output.userStatus else {
            print("*** Unable to get the user's status.")
            return false
        }

        print("User status: \(userStatus)")
        return true
    }
}

```

```

        } catch {
            return false
        }
    }

    /// Create a new user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The ID of the app client to create a user for.
    ///   - userName: The username for the new user.
    ///   - password: The new user's password.
    ///   - email: The new user's email address.
    ///
    /// - Returns: `true` if successful; otherwise `false`.
    func signUp(cipClient: CognitoIdentityProviderClient, clientId: String,
        userName: String, password: String, email: String) async -> Bool {
        let emailAttr = CognitoIdentityProviderClientTypes.AttributeType(
            name: "email",
            value: email
        )

        let userAttrsList = [emailAttr]

        do {
            _ = try await cipClient.signUp(
                input: SignUpInput(
                    clientId: clientId,
                    password: password,
                    userAttributes: userAttrsList,
                    username: userName
                )
            )

            print("====> User \(userName) signed up.")
        } catch _ as AWSIdentityProvider.UsernameExistsException {
            print("*** The username \(userName) already exists. Please use a
different one.")
            return false
        } catch let error as AWSIdentityProvider.InvalidPasswordException {
            print("*** Error: The specified password is invalid. Reason:
\((error.properties.message ?? "<none available>").")")

```

```

        return false
    } catch _ as AWSCognitoIdentityProvider.ResourceNotFoundException {
        print("*** Error: The specified client ID (\(clientId)) doesn't
exist.")
        return false
    } catch {
        print("*** Unexpected error: \(error)")
        return false
    }

    return true
}

/// Requests a new confirmation code be sent to the given user's contact
/// method.
///
/// - Parameters:
///   - cipClient: The `CognitoIdentityProviderClient` to use.
///   - clientId: The application client ID.
///   - userName: The user to resend a code for.
///
/// - Returns: `true` if a new code was sent successfully, otherwise
///   `false`.
func resendConfirmationCode(cipClient: CognitoIdentityProviderClient,
clientId: String,
                           userName: String) async -> Bool {
    do {
        let output = try await cipClient.resendConfirmationCode(
            input: ResendConfirmationCodeInput(
                clientId: clientId,
                username: userName
            )
        )

        guard let deliveryMedium = output.codeDeliveryDetails?.deliveryMedium
    else {
        print("*** Unable to get the delivery method for the resent
code.")
        return false
    }

    print("=====> A new code has been sent by \(deliveryMedium)")
    return true
} catch {

```

```

        print("*** Unable to resend the confirmation code to user
\\(userName).")
        return false
    }
}

/// Submit a confirmation code for the specified user. This is the code as
/// entered by the user after they've received it by email or text
/// message.
///
/// - Parameters:
///   - cipClient: The `CognitoIdentityProviderClient` to use.
///   - clientId: The app client ID the user is signing up for.
///   - userName: The username of the user whose code is being sent.
///   - code: The user's confirmation code.
///
/// - Returns: `true` if the code was successfully confirmed; otherwise
`false`.
func confirmSignUp(cipClient: CognitoIdentityProviderClient, clientId:
String,
                  userName: String, code: String) async -> Bool {
    do {
        _ = try await cipClient.confirmSignUp(
            input: ConfirmSignUpInput(
                clientId: clientId,
                confirmationCode: code,
                username: userName
            )
        )

        print("=====> \\(userName) has been confirmed.")
        return true
    } catch {
        print("=====> \\(userName)'s code was entered incorrectly.")
        return false
    }
}

/// Begin an authentication session.
///
/// - Parameters:
///   - cipClient: The `CognitoIdentityProviderClient` to use.
///   - clientId: The app client ID to use.
///   - userName: The username to check.

```

```

/// - password: The user's password.
/// - userPoolId: The user pool to use.
///
/// - Returns: The session token associated with this authentication
/// session.
func initiateAuth(cipClient: CognitoIdentityProviderClient, clientId: String,
                  userName: String, password: String,
                  userPoolId: String) async -> String? {
    var authParams: [String: String] = [:]

    authParams["USERNAME"] = userName
    authParams["PASSWORD"] = password

    do {
        let output = try await cipClient.adminInitiateAuth(
            input: AdminInitiateAuthInput(
                authFlow:
CognitoIdentityProviderClientTypes.AuthFlowType.adminUserPasswordAuth,
                authParameters: authParams,
                clientId: clientId,
                userPoolId: userPoolId
            )
        )

        guard let challengeName = output.challengeName else {
            print("*** Invalid response from the auth service.")
            return nil
        }

        print("=====> Response challenge is \(challengeName)")

        return output.session
    } catch _ as UserNotFoundException {
        print("*** The specified username, \(userName), doesn't exist.")
        return nil
    } catch _ as UserNotConfirmedException {
        print("*** The user \(userName) has not been confirmed.")
        return nil
    } catch {
        print("*** An unexpected error occurred.")
        return nil
    }
}

```

```

    /// Request and display an MFA secret token that the user should enter
    /// into their authenticator to set it up for the user account.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - authSession: The authentication session to request an MFA secret
    ///     for.
    ///
    /// - Returns: A string containing the MFA secret token that should be
    ///   entered into the authenticator software.
    func getSecretForAppMFA(cipClient: CognitoIdentityProviderClient,
    authSession: String?) async -> String? {
        do {
            let output = try await cipClient.associateSoftwareToken(
                input: AssociateSoftwareTokenInput(
                    session: authSession
                )
            )

            guard let secretCode = output.secretCode else {
                print("*** Unable to get the secret code")
                return nil
            }

            print("=====> Enter this token into Google Authenticator:
            \(secretCode)")
            return output.session
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return nil
        } catch {
            print("*** An unexpected error occurred getting the secret for the
            app's MFA.")
            return nil
        }
    }

    /// Confirm that the user's TOTP authenticator is configured correctly by
    /// sending a code to it to check that it matches successfully.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - session: An authentication session previously returned by an
    ///     `associateSoftwareToken()` call.

```

```

    /// - mfaCode: The 6-digit code currently displayed by the user's
    ///       authenticator, as provided by the user.
    func verifyTOTP(cipClient: CognitoIdentityProviderClient, session: String?,
mfaCode: String?) async {
        do {
            let output = try await cipClient.verifySoftwareToken(
                input: VerifySoftwareTokenInput(
                    session: session,
                    userCode: mfaCode
                )
            )

            guard let tokenStatus = output.status else {
                print("*** Unable to get the token's status.")
                return
            }
            print("=====> The token's status is: \(tokenStatus)")
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user has not been confirmed.")
            return
        } catch {
            print("*** Error verifying the MFA token!")
            return
        }
    }
}

/// Respond to the authentication challenge received from Cognito after
/// initiating an authentication session. This involves sending a current
/// MFA code to the service.
///
/// - Parameters:
///   - cipClient: The `CognitoIdentityProviderClient` to use.
///   - userName: The user's username.
///   - clientId: The app client ID.
///   - userPoolId: The user pool to sign into.

```



```

    /// - mfaCode: The 6-digit MFA code currently displayed by the user's
    ///       authenticator.
    /// - session: The authentication session to continue processing.
    func adminRespondToAuthChallenge(cipClient: CognitoIdentityProviderClient,
    userName: String,
                                clientId: String, userPoolId: String,
    mfaCode: String,
                                session: String) async {
        print("=====> SOFTWARE_TOKEN_MFA challenge is generated...")

        var challengeResponsesOb: [String: String] = [:]
        challengeResponsesOb["USERNAME"] = userName
        challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

        do {
            let output = try await cipClient.adminRespondToAuthChallenge(
                input: AdminRespondToAuthChallengeInput(
                    challengeName:
CognitoIdentityProviderClientTypes.ChallengeNameType.softwareTokenMfa,
                    challengeResponses: challengeResponsesOb,
                    clientId: clientId,
                    session: session,
                    userPoolId: userPoolId
                )
            )

            guard let authenticationResult = output.authenticationResult else {
                print("*** Unable to get authentication result.")
                return
            }

            print("=====> Authentication result (JWTs are redacted):")
            print(authenticationResult)
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")

```

```

        return
    } catch let error as UnauthorizedException {
        print("*** Unauthorized access. Reason: \(error.properties.message ??
"<unknown>")")
    } catch {
        print("*** Error responding to the MFA challenge.")
        return
    }
}

/// Called by ``main()`` to run the bulk of the example.
func runAsync() async throws {
    let config = try await
CognitoIdentityProviderClient.CognitoIdentityProviderClientConfiguration(region:
region)
    let cipClient = CognitoIdentityProviderClient(config: config)

    print("""
        This example collects information about a user, then creates that
user in the
        specified user pool. Then, it enables Multi-Factor Authentication
(MFA) for that
        user by associating an authenticator application (such as Google
Authenticator
        or a password manager that supports TOTP). Then, the user uses a
code from their
        authenticator application to sign in.

        """)

    let userName = stringRequest("Please enter a new username: ")
    let password = stringRequest("Enter a password: ")
    let email = stringRequest("Enter your email address: ", minLength: 5)

    // Submit the sign-up request to AWS.

    print("==> Signing up user \(userName)...")
    if await signUp(cipClient: cipClient, clientId: clientId,
                    userName: userName, password: password,
                    email: email) == false {
        return
    }

    // Check the user's status. This time, it should come back "unconfirmed".

```

```
        print("==> Getting the status of user \(userName) from the user pool
(should be 'unconfirmed')...")
        if await adminGetUser(cipClient: cipClient, userName: userName,
userPoolId: poolId) == false {
            return
        }

        // Ask the user if they want a replacement code sent, such as if the
        // code hasn't arrived yet. If the user responds with a "yes," send a
        // new code.

        if yesNoRequest("==> A confirmation code was sent to \(userName). Would
you like to send a new code (Y/N)? ") {
            print("==> Sending a new confirmation code...")
            if await resendConfirmationCode(cipClient: cipClient, clientId:
clientId, userName: userName) == false {
                return
            }
        }

        // Ask the user to enter the confirmation code, then send it to Amazon
        // Cognito to verify it.

        let code = stringRequest("==> Enter the confirmation code sent to
\(userName): ")
        if await confirmSignUp(cipClient: cipClient, clientId: clientId,
userName: userName, code: code) == false {
            // The code didn't match. Your application may wish to offer to
            // re-send the confirmation code here and try again.
            return
        }

        // Check the user's status again. This time it should come back
        // "confirmed".

        print("==> Rechecking status of user \(userName) in the user pool (should
be 'confirmed')...")
        if await adminGetUser(cipClient: cipClient, userName: userName,
userPoolId: poolId) == false {
            return
        }
        // Check the challenge mode. Here, it should be "mfaSetup", indicating
        // that the user needs to add MFA before using it. This returns a
```

```
// session that can be used to register MFA, or nil if an error occurs.

let authSession = await initiateAuth(cipClient: cipClient, clientId:
clientId,
                                userName: userName, password:
password,
                                userPoolId: poolId)

if authSession == nil {
    return
}

// Ask Cognito for an MFA secret token that the user should enter into
// their authenticator software (such as Google Authenticator) or
// password manager to configure it for this user account. This
// returns a new session that should be used for the new stage of the
// authentication process.

let newSession = await getSecretForAppMFA(cipClient: cipClient,
authSession: authSession)
if newSession == nil {
    return
}

// Ask the user to enter the current 6-digit code displayed by their
// authenticator. Then verify that it matches the value expected for
// the session.

let mfaCode1 = stringRequest("==> Enter the 6-digit code displayed in
your authenticator: ",
                                minLength: 6)
await verifyTOTP(cipClient: cipClient, session: newSession, mfaCode:
mfaCode1)

// Ask the user to authenticate now that the authenticator has been
// configured. This creates a new session using the user's username
// and password as already entered.

print("\nNow starting the sign-in process for user \(userName)... \n")

let session2 = await initiateAuth(cipClient: cipClient, clientId:
clientId,
                                userName: userName, password: password,
userPoolId: poolId)
guard let session2 else {
```

```

        return
    }

    // Now that we have a new auth session, `session2`, ask the user for a
    // new 6-digit code from their authenticator, and send it to the auth
    // session.

    let mfaCode2 = stringRequest("==> Wait for your authenticator to show a
new 6-digit code, then enter it: ",
                                minLength: 6)
    await adminRespondToAuthChallenge(cipClient: cipClient, userName:
userName,
                                    clientId: clientId, userPoolId: poolId,
                                    mfaCode: mfaCode2, session: session2)
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}

```

- 如需 API 詳細資訊，請參閱《適用於 Swift 的 AWS SDK API 參考》中的下列主題。

- [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 AWS SDK 進行 Amazon Cognito 使用者身分驗證後，使用 Lambda 函數寫入自訂活動資料

下列程式碼範例示範如何在 Amazon Cognito 使用者身分驗證之後，使用 Lambda 函數撰寫自訂活動資料。

- 使用管理員函數將使用者新增至使用者集區。
- 設定使用者集區以呼叫 PostAuthentication 觸發條件的 Lambda 函數。
- 將新使用者登入 Amazon Cognito。
- Lambda 函數會將自訂資訊寫入 CloudWatch Logs 和 DynamoDB 資料表。
- 從 DynamoDB 資料表取得並顯示自訂資料，然後清除資源。

Go

SDK for Go V2

 Note

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

在命令提示中執行互動式案例。

```
import (
```

```

"context"
"errors"
"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"
)

// 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(ctx context.Context, 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(ctx, 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(ctx, 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(ctx, 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(ctx context.Context, 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(

```



```

    ctx, 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(ctx context.Context, 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(ctx, 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(ctx context.Context, 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(ctx, 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(ctx context.Context, stackName string) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            runner.resources.Cleanup(ctx)
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Printf("Welcome\n")

    log.Println(strings.Repeat("-", 88))

    stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
    if err != nil {
        panic(err)
    }
    runner.resources.userPoolId = stackOutputs["UserPoolId"]
    runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
    userName, password := runner.AddUserToPool(ctx, stackOutputs["UserPoolId"],
        stackOutputs["TableName"])

    runner.AddActivityLogTrigger(ctx, stackOutputs["UserPoolId"],
        stackOutputs["ActivityLogFunctionArn"])
    runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password)
    runner.helper.ListRecentLogEvents(ctx, stackOutputs["ActivityLogFunction"])
    runner.GetKnownUserLastLogin(ctx, stackOutputs["TableName"], userName)

    runner.resources.Cleanup(ctx)

    log.Println(strings.Repeat("-", 88))
    log.Println("Thanks for watching!")
    log.Println(strings.Repeat("-", 88))
}

```

使用 Lambda 函數來處理 PostAuthentication 觸發條件。

```
import (
    "context"
    "fmt"
    "log"
    "os"
    "time"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

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)
(event.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() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        log.Panicln(err)
    }
    h := handler{
        dynamoClient: dynamodb.NewFromConfig(sdkConfig),
    }
    lambda.Start(h.HandleRequest)
}

```

建立執行一般任務的 struct。

```

import (
    "context"
    "log"
    "strings"
    "time"
    "user_pools_and_lambda_triggers/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
// example.
type IScenarioHelper interface {
    Pause(secs int)
}

```

```

    GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
    error)
    PopulateUserTable(ctx context.Context, tableName string)
    GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
    AddKnownUser(ctx context.Context, tableName string, user actions.User)
    ListRecentLogEvents(ctx context.Context, 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(ctx context.Context, stackName
    string) (actions.StackOutputs, error) {

```

```

    return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, 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(ctx, tableName)
    if err != nil {
        panic(err)
    }
}

// GetKnownUsers gets the users from the known users table in a structured
format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
(actions.UserList, error) {
    knownUsers, err := helper.dynamoActor.Scan(ctx, 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(ctx context.Context, 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(ctx, 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(ctx context.Context,
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(ctx, functionName)
if err != nil {
    panic(err)
}
log.Printf("Getting some recent events from log stream %v\n",
*logStream.LogStreamName)
events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
*logStream.LogStreamName, 10)
if err != nil {
    panic(err)
}
for _, event := range events {
    log.Printf("\t%v", *event.Message)
}
log.Println(strings.Repeat("-", 88))
}

```

建立包裝 Amazon Cognito 動作的 struct。

```

import (
    "context"
    "errors"
    "log"

    "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"
)

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(ctx context.Context, userPoolId
    string, triggers ...TriggerInfo) error {
    output, err := actor.CognitoClient.DescribeUserPool(ctx,
        &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(ctx,
        &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(ctx context.Context, clientId string, userName
string, password string, userEmail string) (bool, error) {
    confirmed := false
    output, err := actor.CognitoClient.SignUp(ctx,
    &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(ctx context.Context, clientId string, userName
string, password string) (*types.AuthenticationResultType, error) {
    var authResult *types.AuthenticationResultType
    output, err := actor.CognitoClient.InitiateAuth(ctx,
    &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(ctx context.Context, clientId string,
    userName string) (*types.CodeDeliveryDetailsType, error) {
    output, err := actor.CognitoClient.ForgotPassword(ctx,
        &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(ctx context.Context, clientId
    string, code string, userName string, password string) error {
    _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
        &cognitoidentityprovider.ConfirmForgotPasswordInput{
            ClientId:      aws.String(clientId),
            ConfirmationCode: aws.String(code),
            Password:     aws.String(password),
        })
    return err
}

```

```

    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(ctx context.Context, userAccessToken
string) error {
  _, err := actor.CognitoClient.DeleteUser(ctx,
&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(ctx context.Context, userPoolId
string, userName string, userEmail string) error {
  _, err := actor.CognitoClient.AdminCreateUser(ctx,
&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(ctx context.Context, userPoolId
string, userName string, password string) error {
    _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
    &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 struct。

```
import (
```

```
"context"
"fmt"
"log"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
"github.com/aws/aws-sdk-go-v2/service/dynamodb"
"github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// 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(ctx context.Context, 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(ctx, &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(ctx context.Context, tableName string) (UserList,
    error) {
    var userList UserList
    output, err := actor.DynamoClient.Scan(ctx, &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(ctx context.Context, 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(ctx, &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 Logs 動作的 struct。

```

import (
    "context"
    "fmt"
    "log"

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

type CloudWatchLogsActions struct {
    CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
    functionName string) (types.LogStream, error) {

```



```

var logStream types.LogStream
logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
output, err := actor.CwlClient.DescribeLogStreams(ctx,
&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(ctx context.Context, 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(ctx,
&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
}

```

建立包裝 AWS CloudFormation 動作的結構。

```

import (
    "context"
    "log"

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

// 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(ctx context.Context, stackName
    string) StackOutputs {
    output, err := actor.CfnClient.DescribeStacks(ctx,
        &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
}

```

清除資源。

```

import (
    "context"
    "log"

```

```

"user_pools_and_lambda_triggers/actions"

"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// 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(ctx context.Context) {
    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(ctx, 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(ctx, 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《適用於 Go 的 AWS SDK 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。

使用 SDK 的 Amazon Cognito Sync 程式碼範例 AWS SDKs

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

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

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

- [使用 AWS SDKs 的 Amazon Cognito Sync 基本範例](#)
 - [使用 AWS SDKs 的 Amazon Cognito Sync 動作](#)
 - [ListIdentityPoolUsage 搭配 AWS SDK 使用](#)

使用 AWS SDKs 的 Amazon Cognito Sync 基本範例

下列程式碼範例示範如何使用 Amazon Cognito Sync 與 AWS SDKs 的基本概念。

範例

- [使用 AWS SDKs 的 Amazon Cognito Sync 動作](#)
 - [ListIdentityPoolUsage 搭配 AWS SDK 使用](#)

使用 AWS SDKs 的 Amazon Cognito Sync 動作

下列程式碼範例示範如何使用 AWS SDKs 執行個別 Amazon Cognito Sync 動作。每個範例均包含 GitHub 的連結，您可以在連結中找到設定和執行程式碼的相關說明。

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

- [ListIdentityPoolUsage 搭配 AWS SDK 使用](#)

ListIdentityPoolUsage 搭配 AWS SDK 使用

下列程式碼範例示範如何使用 ListIdentityPoolUsage。

Rust

SDK for Rust

Note

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

```
async fn show_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 詳細資訊，請參閱《AWS SDK for Rust API 參考》中的 [ListIdentityPoolUsage](#)。

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 使用者集區可搭配多租用戶應用程式運作，這些應用程式會產生必須保留在 Amazon Cognito 配額內的請求量。若要在客戶群成長時擴展此容量，您可以[購買額外的配額容量](#)。

Note

Amazon Cognito [配額](#)會依 AWS 帳戶 和 套用 AWS 區域。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，並確保配額符合您應用程式中的預期數量和預期租用戶數量。

本節說明您可以實作的方法，以在相同 區域中的 Amazon Cognito 資源與 之間分隔租用戶 AWS 帳戶。您也可以將租用戶分割到多個 AWS 帳戶 或 區域，並為每個租用戶提供自己的配額。多區域多租用戶的其他優點包括最高層級的隔離、全球分散式使用者最短的網路傳輸時間，以及組織中現有的分發模型。

單一區域多租用戶也可以為您的客戶和管理員帶來優勢。

下列清單涵蓋具有共用資源的多租用戶的一些優點。

多租用戶的優點

常用使用者目錄

多租用戶支援客戶在多個應用程式中擁有帳戶的模型。您可以將[第三方供應商的身分連結至單一一致的使用者集區設定檔](#)。如果使用者設定檔對其租用戶是唯一的，則具有單一使用者集區的任何多租用戶策略都有一個使用者管理進入點。

常見安全性

在共用使用者集區中，您可以為安全性建立單一標準，並將相同的[威脅防護](#)、[多重要素驗證 \(MFA\)](#) 和[AWS WAF](#)標準套用至所有租用戶。由於 AWS WAF Web ACL 必須與您建立關聯的資源 AWS 區域 位於相同的 中，因此多租用戶會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，您必須套用在資源之間複寫組態的操作標準。

常見自訂

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 觸發組態](#)和身分集區中的 [Amazon Cognito 事件](#)可能會變得複雜。Lambda 函數必須與使用者集區或身分集區 AWS

區域位於相同的 `Region` 中。共用 Lambda 函數可以強制執行自訂身分驗證流程、使用者遷移、權杖產生和區域內其他函數的標準。

常見訊息

Amazon Simple Notification Service (Amazon SNS) 需要在區域中進行其他組態，才能將 [SMS 訊息](#) 傳送給使用者。您可以使用 Amazon Simple Email Service (Amazon SES) 驗證的身分和區域內包含的網域來傳送電子郵件訊息。

透過多租用戶，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 `Region` 都提供 Amazon SNS 和 Amazon SES AWS 區域，因此在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，您可以取得單一 Lambda 函數的常見自訂，以管理您的訊息傳遞。

[受管登入](#) 會在瀏覽器中設定工作階段 Cookie，以便識別已驗證的使用者。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。

有兩種方式可以防止使用託管 UI 工作階段 Cookie 跨應用程式用戶端登入。

- 將您的使用者分隔為每個租戶的使用者集區。
- 將託管 UI 登入取代為 Amazon Cognito 使用者集區 API 登入。

主題

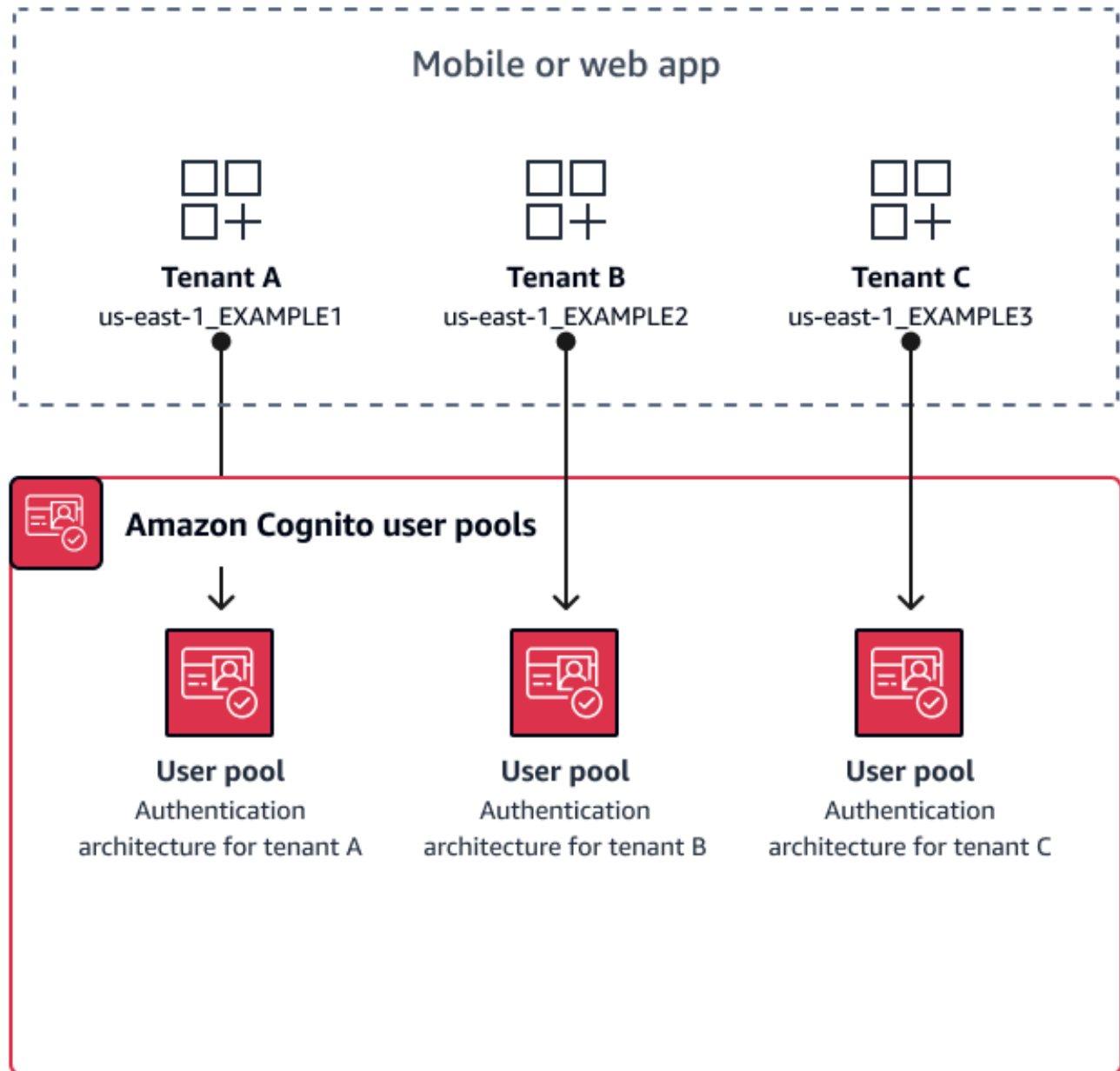
- [使用者集區多租戶最佳實務](#)
- [應用程式用戶端多租用戶最佳實務](#)
- [使用者群組多租戶最佳實務](#)
- [自訂屬性多租戶最佳實務](#)
- [自訂範圍多租戶最佳實務](#)
- [多重租用安全建議](#)

使用者集區多租戶最佳實務

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。依使用者集區隔離租用戶可讓您靈活地進行使用者到租用戶的映射。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，您可以為每個租戶獨立設定託管 UI，並將使用者重新導向至其租戶特定的應用程式執行個體。您也可以使用此方法與後端服務整合，例如 [Amazon API Gateway](#)。

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用戶

當隔離和自訂是您的主要考量時。在具有多個使用者集區的架構中，使用者與租戶之間的關係可能很複雜。請考慮您有兩個教育租用戶的範例。同一個使用者可能是一個應用程式中的有限存取學生，以及另一個應用程式中具有高層級許可的老師。您可能需要在一個應用程式中使用 MFA，但不需要另一個應用程式，或具有不同的密碼政策。由於本機使用者可以使用受管登入在使用者集區中登入多個應用程式用戶端，因此當您希望多個租用戶使用受管登入登入時，使用者集區多租用戶也是理想的選擇。

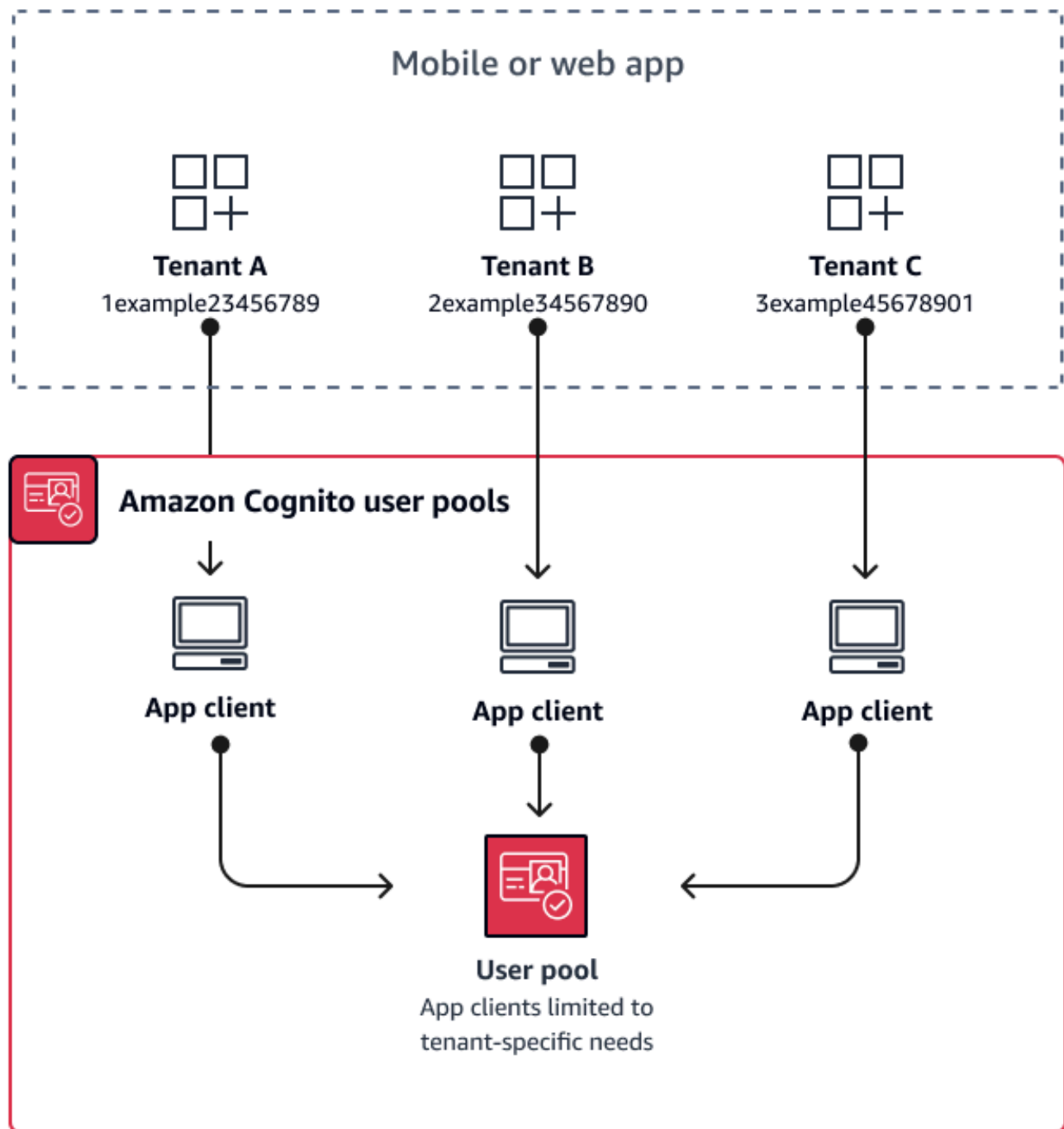
努力程度

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獲得一致且可預測的結果，您必須將 Amazon Cognito 資源與您的自動化工具整合，並在身分驗證架構變得越來越複雜時維持基準。當您想要為應用程式建立單一起點時，您必須建置使用者界面 (UI) 元素，以擷取將使用者路由至正確資源的初始決策。

應用程式用戶端多租用戶最佳實務

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用戶，您可以將任何使用者指派給租用戶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[身分提供者 \(IdPs\)](#) 指派給應用程式用戶端，因此租戶應用程式用戶端可以允許使用租戶特定的 IdP 登入。當使用者存在於多個租用戶時，您可以將其設定檔與多個 IdPs 連結，以獲得一致的使用者體驗。

下圖顯示共用使用者集區中具有專用應用程式用戶端的每個租用戶。



何時實作應用程式用戶端多租用戶

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 觸發條件、密碼政策，以及電子郵件和簡訊的內容和傳遞方法。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

程式用戶端多租用戶非常適合使用app-client-specific IdPs或 Amazon Cognito 使用者集區 API 登入。應用程式用戶端多租用戶也非常適合one-to-many環境。

努力程度

應用程式用戶端多租用戶需要中等努力。應用程式用戶端多租用戶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。在應用程式用戶端多租用架構中，避免需要隔離的託管 UI 登入。您可以使用內建的應用程式用戶端邏輯，將行動應用程式或連結分發至 Web 應用程式，也可以建置初始 UI 元素來判斷使用者的租用。工作量較低，因為您不需要標準化和維護多個使用者集區和身分集區的組態。

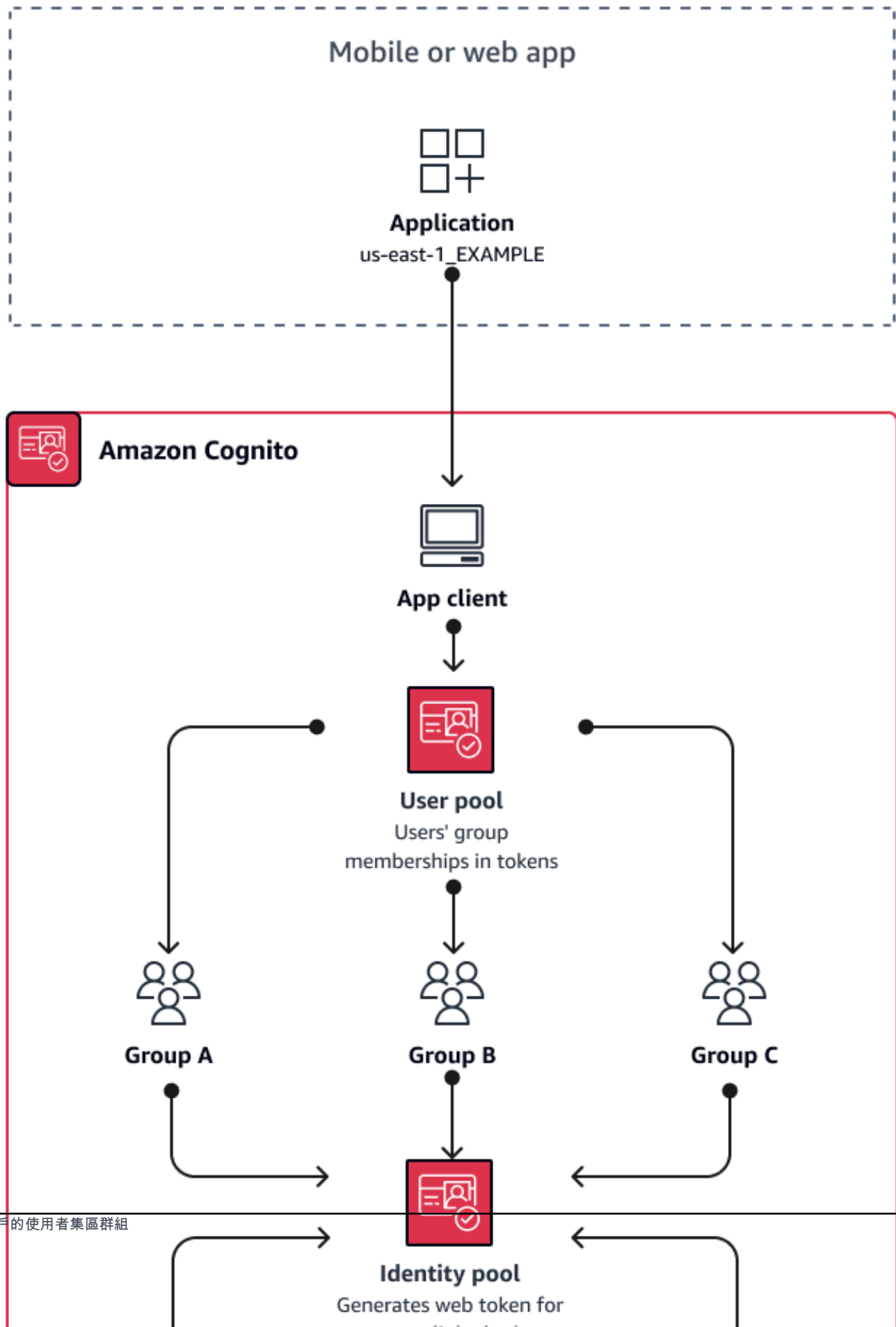
使用者群組多租戶最佳實務

當您的架構需要具有身分集區的 Amazon Cognito 使用者集區時，群組型多租戶最有效。

使用者集區 [ID 和存取權杖](#) 包含cognito:groups宣告。此外，ID 字符包含 cognito:roles和 cognito:preferred_role宣告。當您應用程式中的身分驗證主要結果是來自身分集區的臨時 AWS 憑證時，您的使用者群組成員資格可以判斷他們收到的 [IAM 角色](#) 和許可。

例如，請考慮三個租用戶，每個租用戶都會將應用程式資產存放在自己的 Amazon S3 儲存貯體中。將每個租用戶的使用者指派給相關聯的群組，為群組設定偏好的角色，並授予該角色對其儲存貯體的讀取存取權。

下圖顯示與使用者集區中的專用群組共用應用程式用戶端和使用者集區的租戶，這些群組會決定他們是否符合 IAM 角色的資格。



何時實作群組多租用戶

當存取 AWS 資源是您的主要考量時。Amazon Cognito 使用者集區中的群組是角色型存取控制 (RBAC) 的機制。您可以在使用者集區中設定多個群組，並以群組優先順序進行複雜的 RBAC 決策。身分集區可以為具有最高優先順序的角色、群組宣告中的任何角色，或使用者字符中的其他宣告指派登入資料。

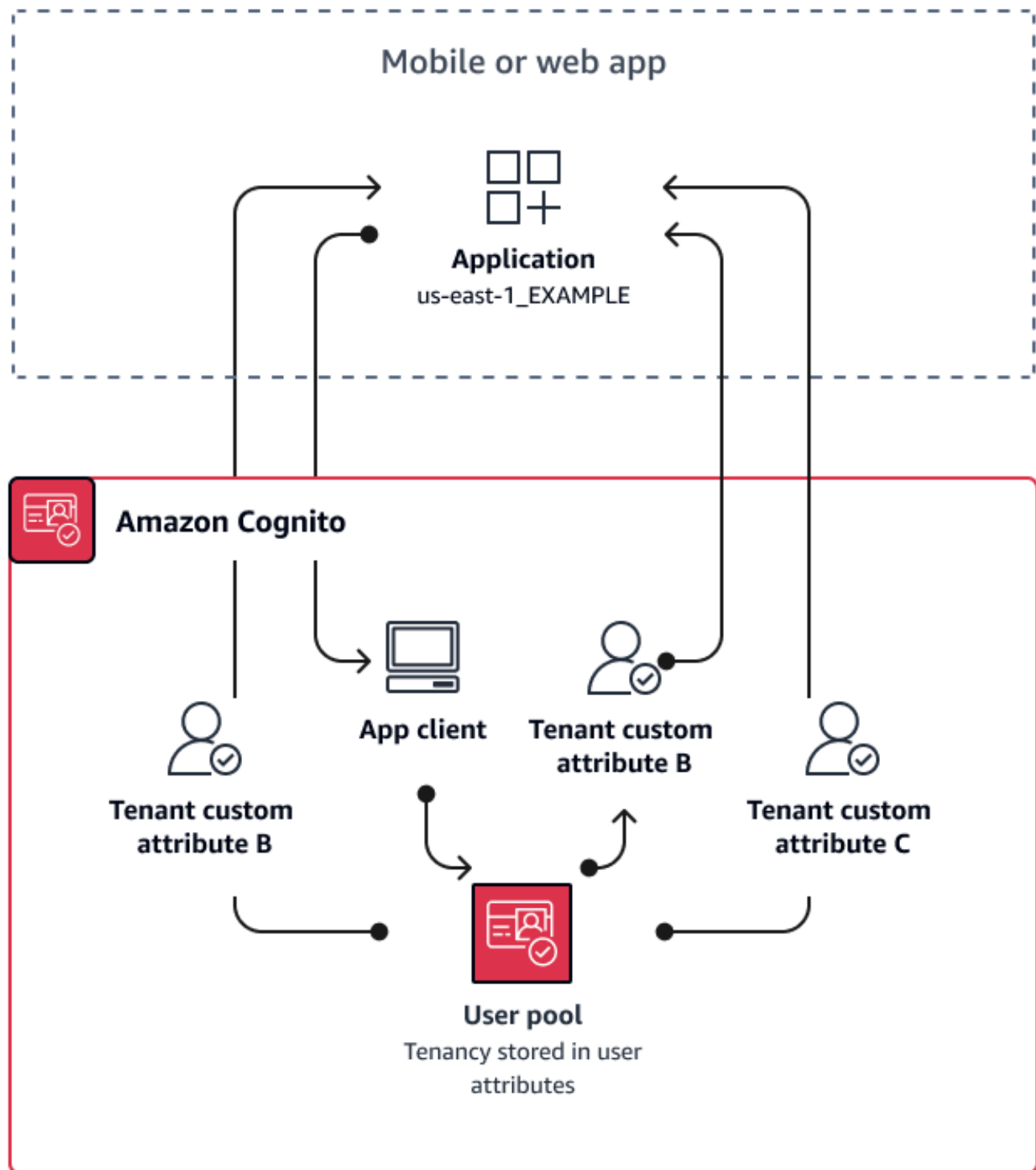
努力程度

單獨使用群組成員資格維持多租戶的工作量很低。不過，若要將使用者集區群組的角色擴展到 IAM 角色選擇的內建容量之外，您必須建置應用程式邏輯，以處理使用者字符中的群組成員資格，並決定在用戶端中該怎麼做。您可以將 Amazon Verified Permissions [IsAuthorizedWithToken](#) API 操作中目前未處理群組識別符，但您可以[開發自訂程式碼](#)來剖析字符的內容，包括群組成員宣告。

自訂屬性多租戶最佳實務

Amazon Cognito 支援具有您選擇的名稱的[自訂屬性](#)。當自訂屬性區分共用使用者集區中的使用者租用時，一個很有用的情況是。當您為使用者指派等屬性的值時`custom:tenantID`，您的應用程式可以相應地指派租戶特定資源的存取權。定義租用戶 ID 的自訂屬性應為不可變或唯讀的應用程式用戶端。

下圖顯示共用應用程式用戶端和使用者集區的租戶，以及使用者集區中指出其所屬租戶的自訂屬性。



當自訂屬性決定租用時，您可以分發單一應用程式或登入 URL。使用者登入後，您的應用程式可以處理 `custom:tenantID` 宣告，決定要載入的資產、要套用的品牌，以及要顯示的功能。對於來自使用者

屬性的進階存取控制決策，請在 Amazon Verified Permissions 中將使用者集區設定為身分提供者，並從 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用戶

當租用為表面層級時。租戶屬性可以為品牌和配置結果做出貢獻。當您想要在租用戶之間實現顯著隔離時，自訂屬性不是最佳選擇。必須在使用者集區或應用程式用戶端層級設定的租用戶之間的任何差異，例如 MFA 或託管 UI 品牌，都需要您以自訂屬性不提供的方式建立租用戶之間的區別。使用身分集區，您甚至可以從使用者 ID 字符中的自訂屬性宣告中選擇 IAM 角色。

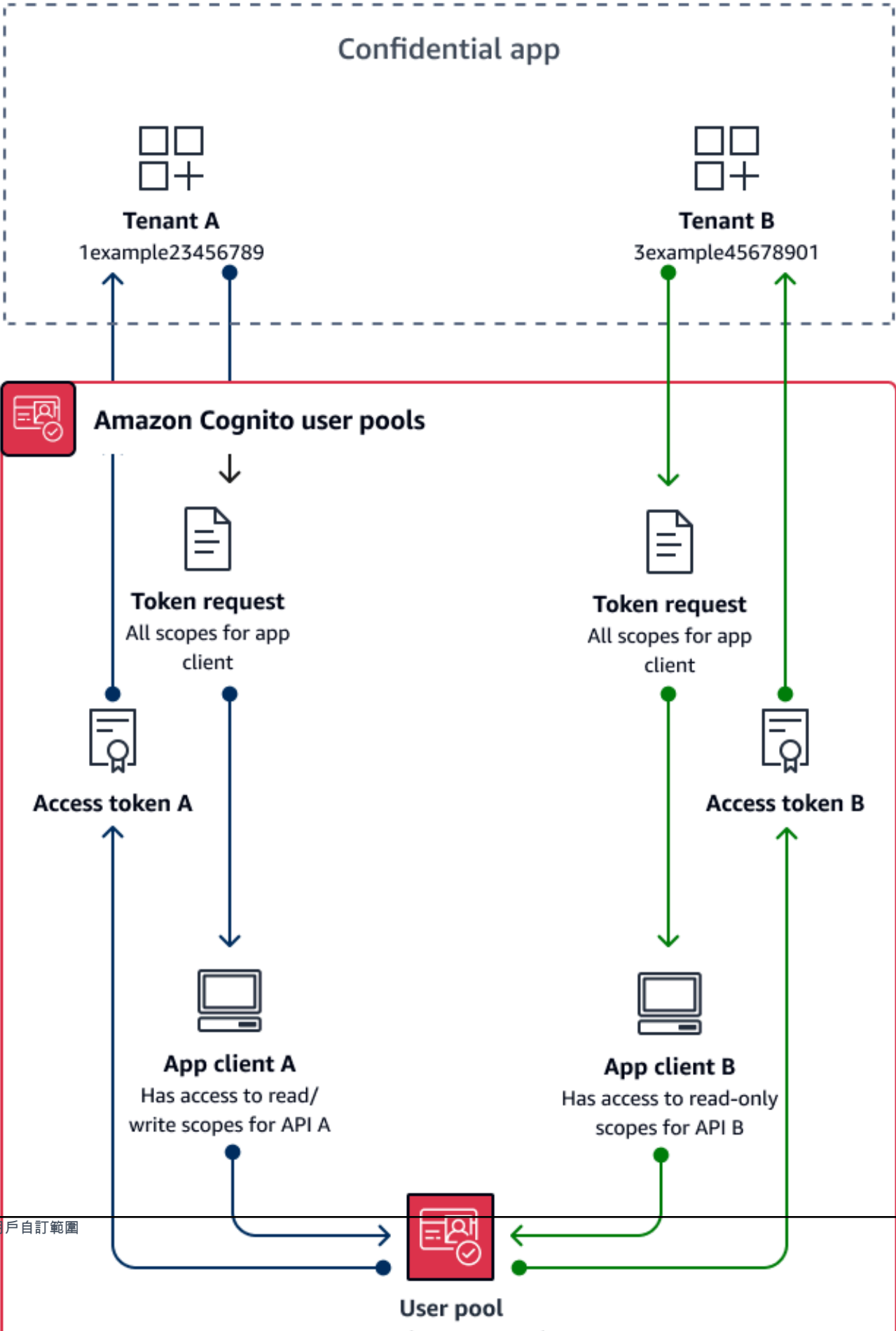
努力程度

由於自訂屬性多租用戶會在您的應用程式上轉移租用戶型授權決策的責任，因此工作量通常很高。如果您已經熟悉剖析 OIDC 宣告的用戶端組態，或在 Amazon Verified Permissions 中，這種方法可能需要最低程度的工作量。

自訂範圍多租戶最佳實務

Amazon Cognito 支援[資源伺服器的](#)自訂 OAuth 2.0 範圍。您可以在具有自訂範圍的machine-to-machine(M2M) 授權模型的使用者集區中實作應用程式用戶端多重租用。範圍型多租戶透過在您的應用程式用戶端或應用程式組態中定義存取權，減少實作 M2M 多租戶所需的工作量。

下圖說明自訂範圍多租用戶的一個選項。它會顯示每個租用戶，其中包含可存取使用者集區中相關範圍的專用應用程式用戶端。



何時實作自訂範圍多租用戶

當您在機密用戶端中使用用戶端憑證的 M2M 授權時。最佳實務是建立應用程式用戶端獨有的資源伺服器。自訂範圍多租用可以是請求相依或用戶端相依。

請求相依

實作應用程式邏輯，僅請求符合您租用戶需求的範圍。例如，應用程式用戶端可能可以發出 API A 和 API B 的讀取和寫入存取權，但租用戶應用程式 A 僅請求 API A 的讀取範圍和表示租用的範圍。此模型允許租用戶之間更複雜的共用範圍組合。

用戶端相依

請求授權請求中指派給應用程式用戶端的所有範圍。若要這樣做，請省略 scope 請求中的 請求參數 [權杖端點](#)。此模型可讓應用程式用戶端存放您要新增至自訂範圍的存取指標。

在任何一種情況下，您的應用程式都會接收具有範圍的存取權杖，這些範圍會指出其所依賴之資料來源的權限。範圍也可以向您的應用程式呈現其他資訊：

- 指定租用
- 有助於請求記錄
- 指出應用程式有權查詢 APIs
- 通知作用中客戶的初始檢查。

努力程度

自訂範圍多租用戶需要與應用程式規模相關的不同程度工作。您必須設計應用程式邏輯，讓您的應用程式剖析存取權杖並提出適當的 API 請求。

例如，資源伺服器範圍的格式為 [resource server identifier]/[name]。資源伺服器識別符不太可能與租戶範圍的授權決策相關，需要一致剖析範圍名稱。

資源範例

下列 AWS CloudFormation 範本會使用一個資源伺服器和應用程式用戶端，為自訂範圍多租用戶建立使用者集區。

```
AWSTemplateFormatVersion: "2010-09-09"
Description: A sample template illustrating scope-based multi-tenancy
Resources:
```

```

MyUserPool:
  Type: "AWS::Cognito::UserPool"
MyUserPoolDomain:
  Type: AWS::Cognito::UserPoolDomain
  Properties:
    UserPoolId: !Ref MyUserPool
    # Note that the value for "Domain" must be unique across all of AWS.
    # In production, you may want to consider using a custom domain.
    # See: https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-
pools-add-custom-domain.html#cognito-user-pools-add-custom-domain-adding
    Domain: !Sub "example-userpool-domain-${AWS::AccountId}"
MyUserPoolResourceServer:
  Type: "AWS::Cognito::UserPoolResourceServer"
  Properties:
    Identifier: resource1
    Name: resource1
    Scopes:
      - ScopeDescription: Read-only access
        ScopeName: readScope
    UserPoolId: !Ref MyUserPool
MyUserPoolTenantBatch1ResourceServer:
  Type: "AWS::Cognito::UserPoolResourceServer"
  Properties:
    Identifier: TenantBatch1
    Name: TenantBatch1
    Scopes:
      - ScopeDescription: tenant1 identifier
        ScopeName: tenant1
      - ScopeDescription: tenant2 identifier
        ScopeName: tenant2
    UserPoolId: !Ref MyUserPool
MyUserPoolClientTenant1:
  Type: "AWS::Cognito::UserPoolClient"
  Properties:
    AllowedOAuthFlows:
      - client_credentials
    AllowedOAuthFlowsUserPoolClient: true
    AllowedOAuthScopes:
      - !Sub "${MyUserPoolTenantBatch1ResourceServer}/tenant1"
      - !Sub "${MyUserPoolResourceServer}/readScope"
    GenerateSecret: true
    UserPoolId: !Ref MyUserPool
Outputs:
  UserPoolClientId:

```

```
Description: User pool client ID
Value: !Ref MyUserPoolClientTenant1
UserPoolDomain:
Description: User pool domain
Value: !Sub "https://${MyUserPoolDomain}.auth.${AWS::Region}.amazoncognito.com"
```

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 Verified Permissions 驗證應用程式中的租用。在您允許應用程式中的使用者請求之前，建置可檢查使用者集區、應用程式用戶端、群組或自訂屬性權利的政策。使用 Amazon Cognito 使用者集區 AWS 建立 Verified Permissions [身分來源](#)。Verified Permissions 具有多租用戶管理的[其他指引](#)。
- 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](#)。
- 針對識別租用戶的使用者設定檔屬性，使用不可變或唯讀的自訂屬性。您只能在建立使用者或在使用者集區中註冊的使用者時，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。
- 在租戶的外部 IdP 和應用程式用戶端之間使用 1：1 映射，以防止未經授權的跨租戶存取。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，以授予使用者存取其他的權限 AWS 服務。

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提供者 (IdP) 聯合。使用者集區可管理處理權杖的開銷，包括透過 Facebook、Google、Amazon 及 Apple 社交登入傳回的權杖，以及從 OpenID Connect (OIDC) 及 SAML IdP 傳回的權杖。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。

使用身分集區，您的使用者可以取得臨時 AWS 憑證來存取 AWS 服務，例如 Amazon S3 和 DynamoDB。身分集區支援匿名訪客使用者，以及透過第三方 IdP 的聯合身分。

主題

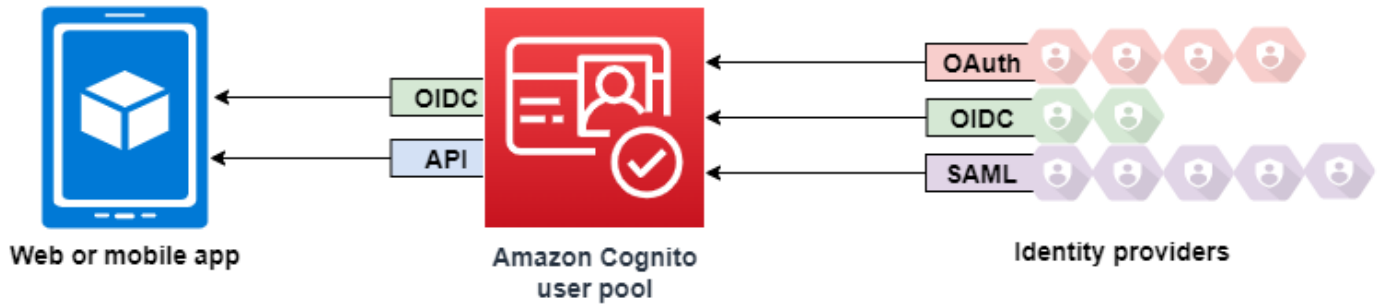
- [以使用者集區進行身分驗證](#)
- [使用使用者集區字符存取後端資源](#)
- [以使用者集區透過 API Gateway 和 Lambda 存取資源](#)
- [使用使用者集區和身分集區存取 AWS 服務](#)
- [以身分集區進行第三方身分驗證以及存取 AWS 服務](#)
- [使用 Amazon Cognito 存取 AWS AppSync 資源](#)

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提供者 (IdP) 聯合。使用者集區可管理處理字符的開銷，包括透過 Facebook、Google、Amazon 及 Apple 社交登入傳回的字符，以及從 OpenID Connect (OIDC) 及 SAML IdP 傳回的字符。

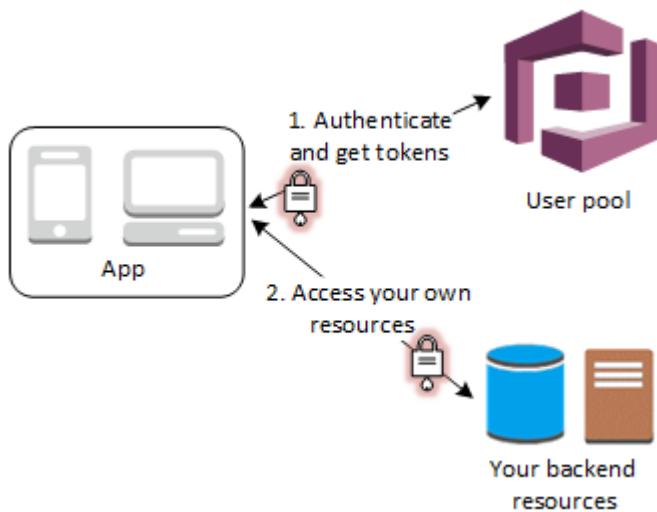
身分驗證成功後，您的 Web 或行動應用程式就會收到來自 Amazon Cognito 的使用者集區權杖。您可以使用這些字符來擷取登入資料，以允許應用程式存取其他 AWS 服務，或者您可以選擇使用這些 AWS 登入資料來控制對伺服器端資源或 Amazon API Gateway 的存取。

如需詳細資訊，請參閱 [範例身分驗證工作階段](#) 和 [了解使用者集區 JSON Web 字符 JWTs](#)。



使用使用者集區字符存取後端資源

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[範圍](#)、[M2M](#) 和 [具有資源伺服器的 APIs](#)。

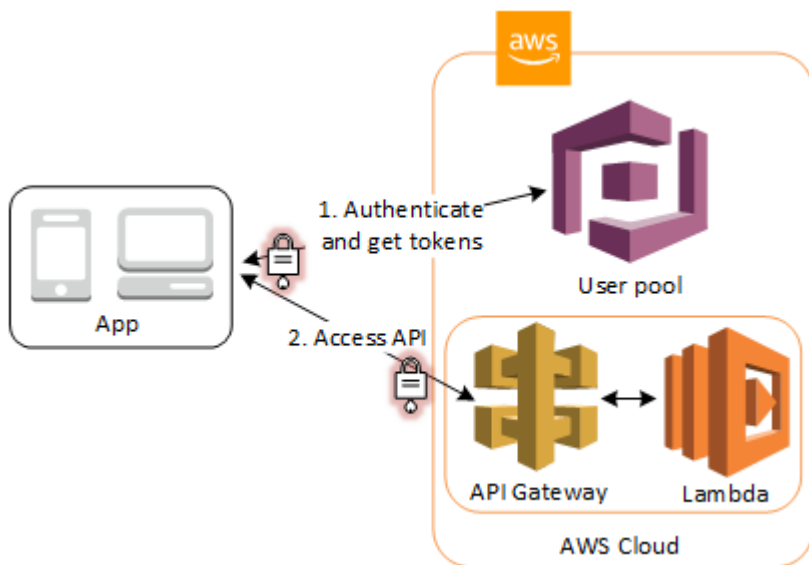
如需有關使用者集區身分驗證的詳細資訊，請參閱 [範例身分驗證工作階段](#) 和 [了解使用者集區 JSON Web 字符 JWTs](#)。

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。

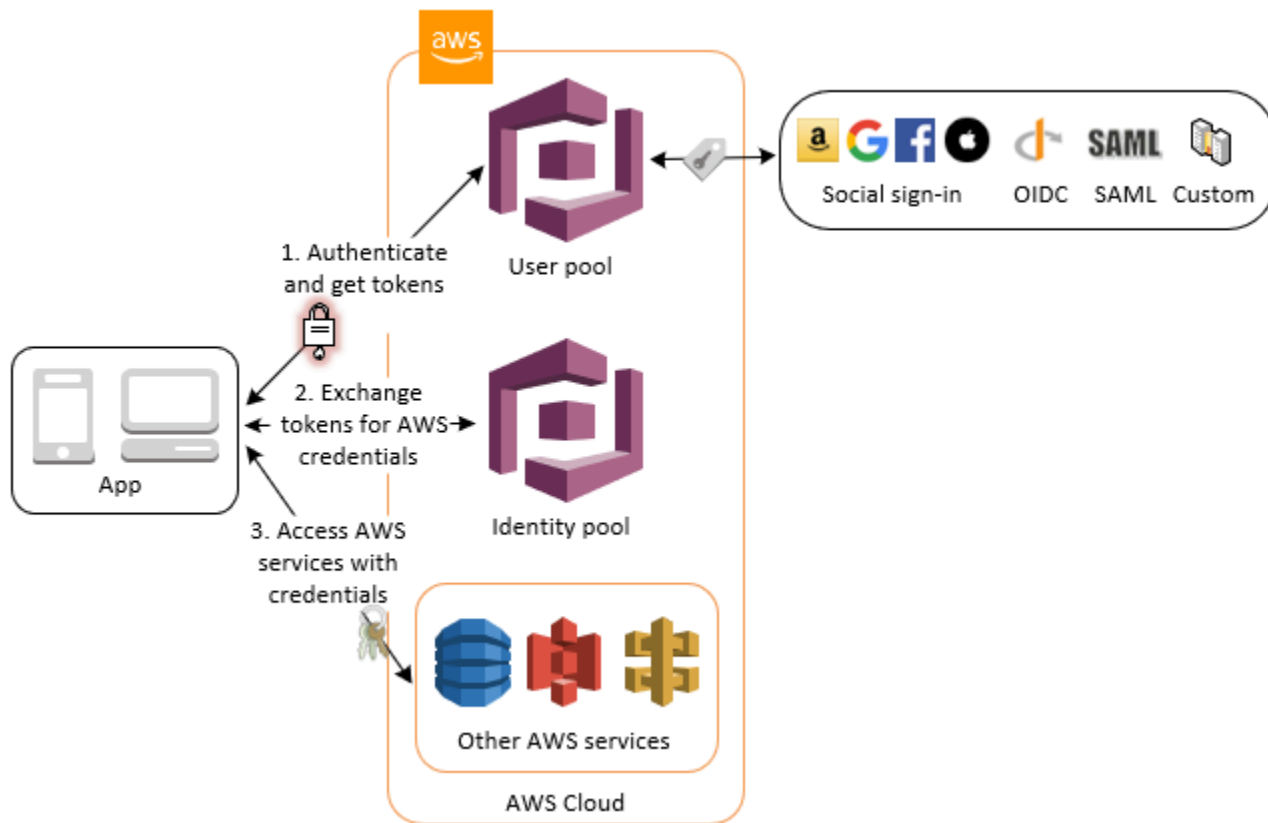
您可以使用使用者集區中的群組，藉由將群組成員資格對應至 IAM 角色來使用 API Gateway 控制許可。使用者隸屬的群組，將會包含在使用者集區於應用程式使用者登入時所提供的 ID 權杖中。如欲了解使用者集區群組的詳細資訊，請參閱[新增群組至使用者集區](#)。

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](#)。



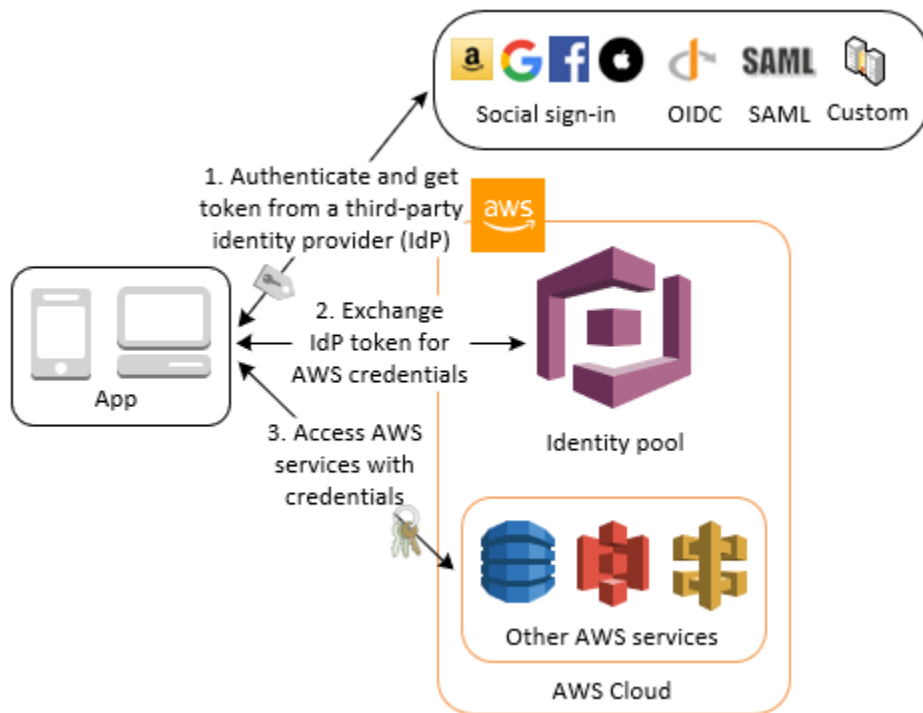
使用使用者集區和身分集區存取 AWS 服務

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。您可以使用身分集區交換其他 AWS 服務的臨時存取權。如需詳細資訊，請參閱[登入後 AWS 服務 使用身分集區存取](#) 和 [Amazon Cognito 身分集區入門](#)。



以身分集區進行第三方身分驗證以及存取 AWS 服務

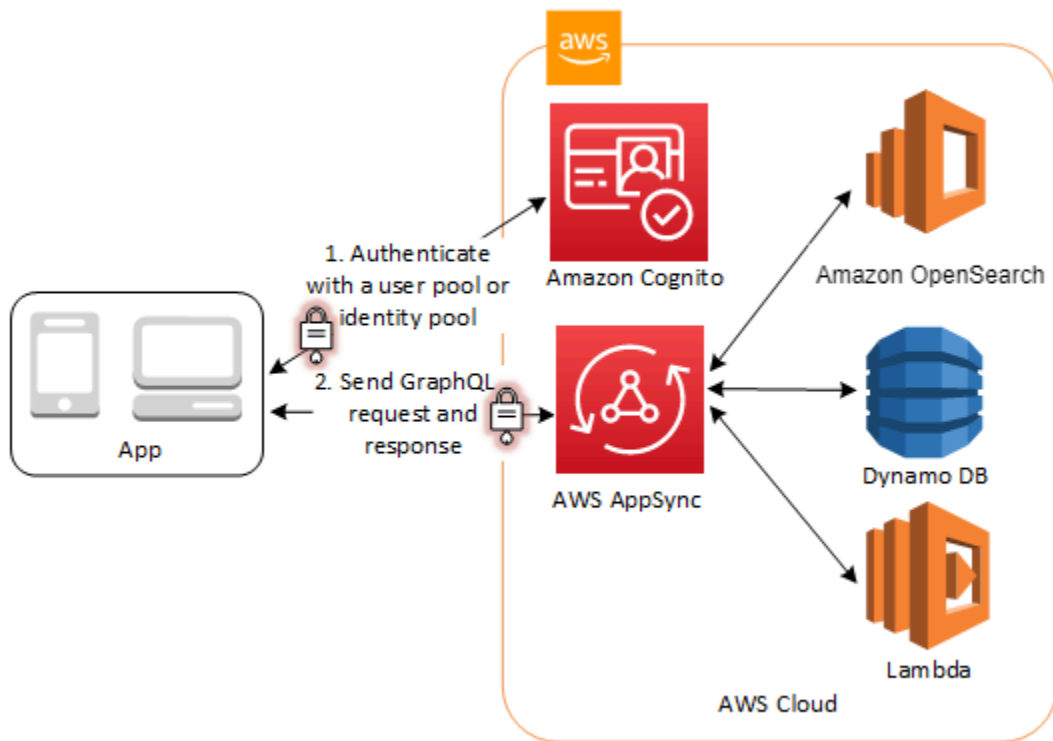
您可以讓使用者透過身分集區存取 AWS 服務。身分集區需使用者的 IdP 字串，且須由第三方身分提供者經身分驗證 (若為匿名訪客則無)。做為交換，身分集區會授予臨時 AWS 登入資料，供您用來存取其他 AWS 服務。如需詳細資訊，請參閱[Amazon Cognito 身分集區入門](#)。



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予使用者對 AWS AppSync 資源的存取權。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](#)。

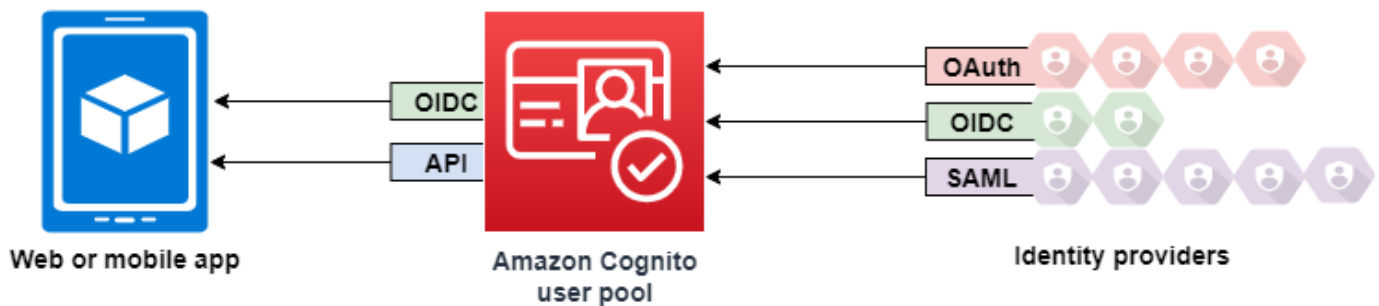
您也可以使用從身分集區收到的 IAM 登入資料來簽署對 AWS AppSync GraphQL API 的請求。請見 [AWS_IAM 授權](#)。



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

- [功能](#)
- [使用者集區功能計劃](#)
- [Amazon Cognito 使用者集區的安全最佳實務](#)
- [使用 Amazon Cognito 使用者集區進行身分驗證](#)
- [使用第三方身分提供者登入使用者集區](#)
- [使用者集區受管登入](#)
- [使用 Lambda 觸發程序來自訂使用者集區工作流程](#)
- [管理使用者集區中的使用者](#)
- [了解使用者集區 JSON Web 字元 \(JWTs\)](#)
- [成功登入後存取資源](#)
- [範圍、M2M 和具有資源伺服器的 APIs](#)
- [設定使用者集區功能](#)
- [使用 Amazon Cognito 使用者集區安全性功能](#)
- [使用者集區端點和受管登入參考](#)

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身分驗證下的註冊功能表中更新自助註冊，或在 [CreateUserPool](#) 或 [UpdateUserPool](#) API 請求中更新 [AllowAdminCreateUserOnly](#) 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。您可以向任何人開啟這些註冊操作，也可以使用用戶端秘密或 AWS 登入資料進行授權。
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。執行 just-in-time AWS Lambda 函數，在現有目錄中查詢您的新使用者，並從現有資料填入其使用者設定檔。

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](#)
- [了解 API、OIDC 和受管登入頁面身分驗證](#)

- [使用 AWS SDKs 的 Amazon Cognito 身分提供者程式碼範例](#)

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 Amazon Cognito 託管的受管登入頁面登入，或透過 Amazon Cognito 使用者集區 API 使用自訂建置的使用者身分驗證服務登入。自訂建置的前端後方的應用程式層可以使用數種方法中的任一種來授權後端請求，以確認合法請求。

使用者可以設定和簽署使用者名稱和密碼、通行金鑰，以及電子郵件和簡訊一次性密碼。您可以提供包含外部使用者目錄的合併登入、登入後的多重要素驗證 (MFA)、信任記住的裝置，以及您設計的自訂身分驗證流程。

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 社交登入來登入和匯入客戶使用者資料。Amazon Cognito 支援透過 OAuth 2.0 登入 Google、Facebook、Amazon 和 Apple。
2. 使用 SAML 和 OIDC 登入來登入和匯入工作和學校使用者資料。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

- [使用第三方身分提供者登入使用者集區](#)
- [將聯合身分使用者連結至現有的使用者描述檔](#)

機器對機器授權

有些工作階段並不是人機互動。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。若要使用 OAuth 2.0 範圍為機器對機器授權產生存取權杖，您可以新增產生[用戶端憑證授權](#)的應用程式用戶端。

相關主題

- [範圍、M2M 和具有資源伺服器的 APIs](#)

受管登入

當您不想建置使用者介面時，您可以向使用者提供自訂的受管登入頁面。受管登入是一組用於註冊、登入、多重要素驗證 (MFA) 和密碼重設的網頁。您可以將受管登入新增至現有網域，或在 AWS 子網域中使用字首識別符。

相關主題

- [使用者集區受管登入](#)
- [設定使用者集區網域](#)

安全

您的本機使用者可以使用簡訊或電子郵件訊息中的代碼，或產生多重要素驗證 (MFA) 代碼的應用程式，提供額外的身分驗證因素。您可以建置機制來設定和處理應用程式中的 MFA，也可以讓受管登入進行管理。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。

如果傳送至使用者集區的網路流量可能是惡意的，您可以監控它並使用 AWS WAF Web ACLs 採取行動。

相關主題

- [將 MFA 新增到使用者集區](#)
- [具有威脅防護的進階安全性](#)
- [將 AWS WAF Web ACL 與使用者集區建立關聯](#)

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。

您可以上傳自訂 CSS 和標誌，為您的使用者提供熟悉的外觀和風格。

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](#)
- [將品牌套用到受管登入頁面](#)

監控與分析

Amazon Cognito 使用者集區會記錄 API 請求，包括受管登入的請求 AWS CloudTrail。您可以在 Amazon CloudWatch Logs 中檢閱效能指標、使用 Lambda 觸發將自訂日誌推送至 CloudWatch、監控電子郵件和簡訊交付，以及在 Service Quotas 主控台中監控 API 請求磁碟區。

使用 Plus [功能計劃](#)，您可以監控使用者身分驗證嘗試是否有自動化學習技術的入侵指標，並立即修復風險。這些進階安全功能也會將使用者活動記錄到您的使用者集區，也可以選擇性地記錄到 Amazon S3、CloudWatch Logs 或 Amazon Data Firehose。

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

- [Amazon Cognito 登入 AWS CloudTrail](#)
- [追蹤 CloudWatch 和 Service Quotas 中的配額和使用量](#)
- [從 Amazon Cognito 使用者集區匯出日誌](#)
- [使用 Amazon Pinpoint 進行使用者集區分析](#)

Amazon Cognito 身分池整合

Amazon Cognito 的另一半是身分池。身分集區提供憑證，可授權和監控來自您使用者的 API 請求 AWS 服務，例如 Amazon DynamoDB 或 Amazon S3。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](#)

使用者集區功能計劃

了解成本是準備實作 Amazon Cognito 使用者集區身分驗證的關鍵步驟。Amazon Cognito 具有使用者集區的功能計劃。每個計劃都有一組功能和每個作用中使用者的每月成本。每個功能計劃都會解鎖存取比之前更多的功能。

使用者集區具有您可以開啟和關閉的各種功能。例如，您可以開啟多重要素驗證 (MFA) 並關閉第三方身分提供者 (IdPs) 登入。有些變更需要您切換功能計劃。使用者集區的下列特性會決定每月 AWS 向您收取用量的費用。

- 您選擇的功能
- 您的應用程式對使用者集區 API 發出的每秒請求數
- 每月具有身分驗證、更新或查詢活動的使用者數量，也稱為[每月作用中使用者](#)或 MAUs
- 來自第三方 SAML 2.0 或 OpenID Connect (OIDC) IdPs 每月作用中使用者數量
- 為 machine-to-machine 授權執行用戶端憑證授予的應用程式用戶端和使用者集區數量

如需使用者集區定價的最新資訊，請參閱 [Amazon Cognito 定價](#)。

特徵計劃選擇適用於一個使用者集區。相同 中的不同使用者集區 AWS 帳戶 可以有不同的計劃選擇。您無法將個別的功能計劃套用至使用者集區中的應用程式用戶端。新使用者集區的預設計劃選擇是 Essentials。

您可以隨時在功能計劃之間切換，以符合應用程式的需求。計劃之間的某些變更需要您關閉作用中的功能。如需詳細資訊，請參閱[關閉功能以變更功能計劃](#)。

使用者集區功能計劃

Lite

Lite 是一種低成本的功能計劃，適用於每月作用中使用者數量較低的使用者集區。該計劃足以用於具有基本身分驗證功能的使用者目錄。它包含登入功能和傳統託管 UI，這是較精簡、較不自訂的受管登入前身。Lite 計劃不包含許多較新的功能，例如存取金鑰自訂和通行金鑰驗證。

基本概念

Essentials 具有所有最新的使用者集區身分驗證功能。無論您的登入頁面是受管登入或自訂建置，此計劃都會為您的應用程式新增新選項。Essentials 具有進階身分驗證功能，例如以[選擇為基礎的登入](#)和[電子郵件 MFA](#)。

Plus

此外，包含 Essentials 計畫中的所有內容，並新增進階安全功能來保護您的使用者。監控使用者登入、註冊和密碼管理請求是否有入侵指標。例如，使用者集區可以偵測使用者是從非預期的位置登入，還是使用屬於公開違規的密碼。

具有 Plus 計畫的使用者集區會產生使用者活動詳細資訊和風險評估的日誌。當您將日誌匯出到外部服務時，您可以將自己的用量和安全性分析套用至這些日誌。

Note

先前，某些使用者集區功能已包含在進階安全功能定價結構中。此結構中包含的功能現在在 Essentials 或 Plus 計劃下。

主題

- [選取功能計劃](#)
- [依計劃列出的功能](#)
- [基本計劃功能](#)
- [Plus 計劃功能](#)
- [關閉功能以變更功能計劃](#)

選取功能計劃

AWS Management Console

選擇功能計劃

1. 前往 [Amazon Cognito 主控台](#)。如果出現提示，請輸入您的 AWS 登入資料。
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. 選取設定選單並檢閱功能計劃索引標籤。
5. 檢閱 Lite、Essentials 和 Plus 計劃中可供您使用的功能。
6. 若要變更您的計劃，請選取切換到基本設定，或切換到 Plus。若要切換到 Lite 計劃，請選擇其他計劃，然後與 Lite 比較。

7. 在下一個畫面上，檢閱您的選擇，然後選取確認。

CLI/API/SDK

[CreateUserPool](#) 和 [UpdateUserPool](#) 操作會在 `UserPoolTier` 參數中設定您的功能計劃。當您未指定的值時 `UserPoolTier`，您的使用者集區會預設為 `Essentials`。如果您將 `AdvancedSecurityMode` 設定為 `AUDIT` 或 `ENFORCED`，則使用者集區層必須是 `PLUS`，並且在未指定 `PLUS` 時預設為。

如需語法，請參閱 [CreateUserPool 中的範例](#)。如需各種程式設計語言的 AWS SDKs 中的此函數連結，請參閱 [CreateUserPool 中的](#)。

```
"UserPoolTier": "PLUS"
```

在 AWS CLI 中，此選項是 `--user-pool-tier` 引數。

```
--user-pool-tier PLUS
```

如需詳細資訊，請參閱 AWS CLI 命令參考中的 [create-user-pool](#) 和 [update-user-pool](#)。

依計劃列出的功能

使用者集區中的功能和計劃

功能	描述	特徵計劃
防止不安全的密碼	在執行時間檢查純文字密碼是否有入侵指標	Plus
防止惡意登入嘗試	在執行時間检查工作階段屬性是否有入侵指標	Plus
記錄和分析使用者活動	產生使用者身分驗證工作階段屬性和風險分數的日誌	Plus
匯出使用者活動日誌	將使用者工作階段和風險日誌推送至外部 AWS 服務	Plus

功能	描述	特徵計劃
使用視覺化編輯器自訂受管登入頁面	在 Amazon Cognito 主控台中使用視覺化編輯器，將品牌和風格套用至受管登入頁面	基本 + Plus
具有電子郵件一次性代碼的 MFA	請求或要求本機使用者在使用者名稱驗證後提供額外的電子郵件訊息登入因素	基本 + Plus
在執行時間自訂存取權杖範圍和宣告	使用 Lambda 觸發來擴展使用者集區存取字符的授權功能	基本 + Plus
使用一次性代碼的無密碼登入	允許使用者透過電子郵件或簡訊接收一次性密碼作為其第一個身分驗證因素	基本 + Plus
使用硬體或軟體 FIDO2 驗證器的通行金鑰登入	允許使用者使用存放在 FIDO2 驗證器上的密碼編譯金鑰作為其第一個身分驗證因素	基本 + Plus
註冊和登入	執行身分驗證操作，並讓新使用者在您的應用程式中註冊帳戶。	Lite + Essentials + Plus
使用者群組	建立使用者的邏輯分組，並為身分集區操作指派預設 IAM 角色。	Lite + Essentials + Plus
使用社交、SAML 和 OIDC 供應商登入	提供使用者直接登入的選項，或使用他們偏好的供應商登入。	Lite + Essentials + Plus
OAuth 2.0/OIDC 授權伺服器	做為 OIDC 發行者。	Lite + Essentials + Plus
登入頁面	用於身分驗證的網頁託管集合。受管登入可在 Essentials 和 Plus 層中使用。傳統託管 UI 適用於所有功能層。	Lite + Essentials + Plus

功能	描述	特徵計劃
密碼、自訂、重新整理金鑰和 SRP 身分驗證	在您的應用程式中提示使用者輸入使用者名稱和密碼。	Lite + Essentials + Plus
具有用戶端憑證 Machine-to-machine(M2M)	發行存取權杖以授權非人類實體。	Lite + Essentials + Plus
搭配資源伺服器的 API 授權	使用授權存取外部系統的自訂範圍發出存取權杖。	Lite + Essentials + Plus
使用者匯入	從 CSV 檔案設定匯入任務，並在使用者登入時執行 just-in-time 遷移。	Lite + Essentials + Plus
MFA 搭配驗證器應用程式和 SMS 一次性密碼	請求或要求本機使用者在使用者名稱驗證後提供額外的簡訊或驗證器應用程式登入因素	Lite + Essentials + Plus
在執行時間自訂 ID 字符範圍和宣告	使用 Lambda 觸發來擴展使用者集區身分 (ID) 字符的身分驗證功能	Lite + Essentials + Plus
使用 Lambda 觸發器的自訂執行時間動作	使用執行外部動作和影響身分驗證的 Lambda 函數，在執行時間自訂登入程序	Lite + Essentials + Plus
使用 CSS 自訂受管登入頁面	下載 CSS 範本並變更受管登入頁面中的某些樣式	Lite + Essentials + Plus

基本計劃功能

Essentials 功能計劃具有 Amazon Cognito 使用者集區的大多數最佳和最新功能。當您從 Lite 切換到 Essentials 計劃時，您會取得受管登入頁面的新功能、使用電子郵件訊息一次性密碼的多重要素驗證、增強型密碼政策和自訂存取字符。若要隨時 up-to-date 掌握最新的使用者集區功能，請為您的使用者集區選擇 Essentials 計劃。

以下各節提供您可以使用 Essentials 計畫新增至應用程式的功能的簡短概觀。如需詳細資訊，請參閱下列頁面。

其他資源

- 存取權杖自訂：[產生權杖前 Lambda 觸發程序](#)
- 電子郵件 MFA：[SMS 和電子郵件訊息 MFA](#)
- 密碼歷史記錄：[密碼、帳戶復原和密碼政策](#)
- 增強型 UI：[將品牌套用到受管登入頁面](#)

主題

- [存取權杖自訂](#)
- [電子郵件 MFA](#)
- [防止密碼重複使用](#)
- [受管登入託管登入和授權伺服器](#)
- [以選擇為基礎的身分驗證](#)

存取權杖自訂

使用者集區[存取權杖](#)會授予應用程式許可：[存取 API](#)、從 [userInfo 端點](#)擷取使用者屬性，或建立外部系統的[群組成員](#)資格。在進階案例中，您可能想要使用應用程式在執行時間決定的其他暫時參數，將新增至使用者集區目錄中的預設存取金鑰資料。例如，您可能想要使用 [Amazon Verified Permissions](#) 驗證使用者的 API 許可，並相應地調整存取權杖中的範圍。

Essentials 計畫會新增至[權杖產生前觸發](#)程序的現有函數。透過低階計畫，您可以自訂 ID 權杖，其中包含其他宣告、角色和群組成員資格。Essentials 新增了觸發條件輸入事件的新版本，可自訂存取權杖宣告、角色、群組成員資格和範圍。存取字串自訂可供具有事件第 3 machine-to-machine(M2M) [用戶端憑證授予](#)使用。

自訂存取權杖

1. 選取 Essentials 或 Plus 功能計畫。
2. 為您的觸發建立 Lambda 函數。若要使用我們的範例函數，[請將其設定為 Node.js](#)。
3. 使用[範例程式碼](#)填入您的 Lambda 函數或編寫您自己的程式碼。函數必須處理來自 Amazon Cognito 的請求物件，並傳回您要包含的變更。

4. 將新函數指派為第 [2 版或第 3 版](#) 權杖產生前觸發條件。第 2 版事件會自訂使用者身分的存取權杖。第 3 版會自訂使用者和機器身分的存取權杖。

進一步了解

- [自訂存取權杖](#)
- [如何在 Amazon Cognito 使用者集區中自訂存取權杖](#)

電子郵件 MFA

Amazon Cognito 使用者集區可設定為使用電子郵件作為多重驗證 (MFA) 的第二個因素。使用電子郵件 MFA，Amazon Cognito 可以向使用者傳送電子郵件，其中包含他們必須輸入的驗證碼，以完成身分驗證程序。這會為使用者登入流程新增重要的額外安全層。若要啟用電子郵件型 MFA，使用者集區必須設定為使用 [Amazon SES 電子郵件傳送組態](#)，而非預設的電子郵件組態。

當您的使用者透過電子郵件訊息選取 MFA 時，只要嘗試登入，Amazon Cognito 就會傳送一次性驗證碼到使用者註冊的電子郵件地址。然後，使用者必須將此程式碼提供給使用者集區，以完成身分驗證流程並取得存取權。這可確保即使使用者的使用者名稱和密碼遭到入侵，他們仍必須提供額外的因素，即透過電子郵件傳送的程式碼，才能存取您的應用程式資源。

如需詳細資訊，請參閱 [SMS 和電子郵件訊息 MFA](#)。以下是如何為電子郵件 MFA 設定使用者集區和使用者的概觀。

在 Amazon Cognito 主控台中設定電子郵件 MFA

1. 選取 Essentials 或 Plus 功能計劃。
2. 在使用者集區的登入選單中，編輯多重要素驗證。
3. 選擇您要設定的 MFA 強制執行層級。透過需要 MFA，API 中的使用者會自動收到使用 MFA 設定、確認和登入的挑戰。在需要 MFA 的使用者集區中，受管登入會提示他們選擇和設定 MFA 因素。使用選用 MFA，您的應用程式必須提供使用者設定 MFA 和設定使用者電子郵件 MFA 偏好設定的選項。
4. 在 MFA 方法下，選取電子郵件訊息做為其中一個選項。

進一步了解

- [SMS 和電子郵件訊息 MFA](#)

防止密碼重複使用

根據預設，Amazon Cognito 使用者集區密碼政策會設定密碼長度和字元類型需求，以及暫時密碼過期。Essentials 計劃新增了強制執行密碼歷史記錄的功能。當使用者嘗試重設密碼時，您的使用者集區可能會阻止他們將其設定為先前的密碼。如需設定密碼政策的詳細資訊，請參閱 [新增使用者集區密碼要求](#)。以下是如何使用密碼歷史記錄政策設定使用者集區的概觀。

在 Amazon Cognito 主控台中設定密碼歷史記錄

1. 選取 Essentials 或 Plus 功能計劃。
2. 在使用者集區的身分驗證方法功能表中，找到密碼政策，然後選取編輯。
3. 設定其他可用選項，並設定防止使用先前密碼的值。

進一步了解

- [密碼、帳戶復原和密碼政策](#)

受管登入託管登入和授權伺服器

Amazon Cognito 使用者集區具有支援下列功能的選用網頁：OpenID Connect (OIDC) IdP、服務供應商或依賴第三方 IdPs 以及用於註冊和登入的公有使用者互動頁面。這些頁面統稱為受管登入。當您為使用者集區選擇網域時，Amazon Cognito 會自動啟用這些頁面。當 Lite 計劃具有託管 UI 時，Essentials 計劃會開啟此進階版本的註冊和登入頁面。

受管登入頁面具有乾淨且 up-to-date 界面，具有更多用於自訂品牌和樣式的功能和選項。Essentials 計劃是解除鎖定受管登入存取權的最低計劃層級。

在 Amazon Cognito 主控台中設定受管登入

1. 從設定功能表中，選取基本或增強功能計劃。
2. 從網域功能表中，[將網域指派給](#)使用者集區，然後選取受管登入的品牌版本。
3. 從受管登入功能表的樣式索引標籤下，選擇建立樣式並將樣式指派給應用程式用戶端，或建立新的應用程式用戶端。

進一步了解

- [使用者集區受管登入](#)

以選擇為基礎的身分驗證

在增強型 UI 和 SDK 型 API 操作中，Essentials 層為身分驗證操作引入了新的身分驗證流程。此流程是選擇型身分驗證。以選擇為基礎的身分驗證是一種方法，其中使用者的身分驗證不是從登入方法的應用程式端宣告開始，而是查詢可能的登入方法，接著選擇。您可以將使用者集區設定為支援以選擇為基礎的身分驗證，並解鎖使用者名稱密碼、無密碼和密碼金鑰身分驗證。在 API 中，這是USER_AUTH流程。

在 Amazon Cognito 主控台中設定以選擇為基礎的身分驗證

1. 選取 Essentials 或 Plus 功能計劃。
2. 在使用者集區的登入功能表中，編輯以選擇為基礎的登入選項。選取並設定您要在以選擇為基礎的身分驗證中啟用的身分驗證方法。
3. 在使用者集區的身分驗證方法功能表中，編輯登入操作的組態。

進一步了解

- [使用 Amazon Cognito 使用者集區進行身分驗證](#)

Plus 計劃功能

Plus 功能計劃具有 Amazon Cognito 使用者集區的進階安全功能。這些功能會在執行時間記錄和分析使用者內容，以找出裝置、位置、請求資料和密碼中的潛在安全問題。然後，它們透過自動回應來封鎖或新增安全性防護到使用者帳戶，以降低潛在風險。您也可以將安全日誌匯出至 Amazon S3、Amazon Data Firehose 或 Amazon CloudWatch Logs 以進行進一步分析。

當您從 Essentials 切換到 Plus 計劃時，您會取得 Essentials 中的所有功能，以及後續的其他功能。這包括威脅防護集的安全選項，也稱為進階安全功能。若要將使用者集區設定為自動適應身分驗證前端中的威脅，請為您的使用者集區選擇 Plus 計劃。

以下各節提供您可以使用 Plus 計劃新增至應用程式的功能的簡短概觀。如需詳細資訊，請參閱下列頁面。

其他資源

- 自適應身分驗證：[使用自適應身分驗證](#)
- 已遭入侵的登入資料：[使用遭盜用憑證偵測](#)
- 日誌匯出：[從 Amazon Cognito 使用者集區匯出日誌](#)

主題

- [威脅防護：自適應身分驗證](#)
- [威脅防護：已遭入侵的憑證偵測](#)
- [威脅防護：使用者活動記錄](#)

威脅防護：自適應身分驗證

Plus 計劃包含自適應身分驗證功能。當您啟用此功能時，您的使用者集區會對每個使用者身分驗證工作階段進行風險評估。從產生的風險評分中，您可以封鎖身分驗證或推送 MFA 給以高於您確定閾值的風險層級登入的使用者。透過自適應身分驗證，您的使用者集區和應用程式會自動封鎖或設定 MFA，以供您懷疑帳戶遭到攻擊的使用者使用。您也可以提供使用者集區中風險評等的意見回饋，以調整未來的評等。

在 Amazon Cognito 主控台中設定自適應身分驗證

1. 選取 Plus 功能計劃。
2. 在使用者集區的威脅防護選單中，編輯威脅防護下的標準和自訂身分驗證。
3. 將標準或自訂身分驗證的強制執行模式設定為全功能。
4. 在自適應身分驗證下，設定不同風險層級的自動風險回應。

進一步了解

- [使用自適應身分驗證](#)
- [在應用程式中收集威脅防護的資料](#)

威脅防護：已遭入侵的憑證偵測

Plus 計劃包含遭盜用憑證偵測功能。此功能可防止使用不安全的密碼，以及此實務建立之意外應用程式存取的威脅。當您允許使用者使用使用者名稱和密碼登入時，他們可能會重複使用已在別處使用的密碼。該密碼可能已洩漏，或只是經常猜測。透過憑證洩露偵測，您的使用者集區會讀取使用者提交的密碼，並將其與密碼資料庫進行比較。如果操作導致決定密碼可能洩露，您可以將使用者集區設定為封鎖登入，然後為應用程式中的使用者啟動密碼重設。

遭入侵憑證偵測可能會在新使用者註冊、現有使用者登入，以及使用者嘗試重設密碼時，對不安全的密碼做出反應。透過此功能，您的使用者集區可以防止或警告使用者在任何地方輸入不安全的密碼登入。

在 Amazon Cognito 主控台中設定遭盜用憑證偵測

1. 選取 Plus 功能計劃。
2. 在使用者集區的威脅防護選單中，編輯威脅防護下的標準和自訂身分驗證。
3. 將標準或自訂身分驗證的強制執行模式設定為全功能。
4. 在遭入侵的登入資料下，設定您要檢查的身分驗證操作類型，以及您要從使用者集區自動回應。

進一步了解

- [使用遭盜用憑證偵測](#)

威脅防護：使用者活動記錄

Plus 計劃新增了記錄功能，可提供使用者身分驗證嘗試的安全性分析和詳細資訊。您可以查看風險評估、使用者 IP 地址、使用者代理程式，以及連接到您應用程式之裝置的其他資訊。您可以使用內建的威脅防護功能對此資訊採取行動，也可以在自己的系統中分析日誌並採取適當的動作。您可以將日誌從威脅防護匯出至 Amazon S3、CloudWatch Logs 或 Amazon DynamoDB。

在 Amazon Cognito 主控台中設定使用者活動記錄

1. 選取 Plus 功能計劃。
2. 在使用者集區的威脅防護選單中，編輯威脅防護下的標準和自訂身分驗證。
3. 將標準或自訂身分驗證的強制執行模式設定為僅限稽核。這是日誌的最低設定。您也可以在全功能模式中啟用它，並設定其他威脅防護功能。
4. 若要將您的日誌匯出到另一個 AWS 服務以進行第三方分析，請前往使用者集區的日誌串流功能表，並設定匯出目的地。

進一步了解

- [匯出使用者身分驗證事件](#)
- [從 Amazon Cognito 使用者集區匯出日誌](#)

關閉功能以變更功能計劃

功能計劃會將組態選項新增至您的使用者集區。您只能在相關功能計劃作用中時設定和使用這些功能。例如，您可以在 Plus 和 Essentials 計畫中設定存取權杖自訂，但無法在 Lite 計畫中設定。若要停用這

些功能，您必須停用每個作用中元件。Amazon Cognito 主控台的設定選單中的切換到選項會通知您在變更功能計劃之前必須停用的功能。透過本章，您可以了解停用對使用者集區組態所做的變更，以及如何個別關閉這些功能。

存取權杖自訂

若要切換到不包含存取權杖自訂的計劃，您必須從使用者集區中移除權杖產生前 [Lambda 觸發條件](#)。若要在不自訂存取權杖的情況下新增產生權杖前觸發條件，請將新函數指派給觸發條件，並針對 `V1_0` 事件進行設定。這些第一版觸發事件只能處理對 ID 字符的變更。

若要手動停用存取權杖自訂，請移除權杖產生前觸發條件，並新增第 1 版觸發條件。

威脅防護

若要在沒有威脅防護的情況下切換到計劃，請從使用者集區的威脅防護功能表中停用所有功能。

日誌匯出

若要在沒有日誌匯出的情況下切換到計劃，請從使用者集區の日誌串流功能表停用它。您的使用者集區不會再產生本機或匯出的使用者活動日誌。您也可以傳送 [SetLogDeliveryConfiguration](#) API 請求，移除 `EventSource` 值為的任何組態 `UserActivity`。

電子郵件 MFA

若要切換到沒有電子郵件 MFA 的計劃，請前往使用者集區的登入功能表。編輯多重要素驗證，並取消選取電子郵件訊息做為其中一個可用的 MFA 方法。

Amazon Cognito 使用者集區的安全最佳實務

此頁面說明當您想要防範常見威脅時，您可以實作的安全最佳實務。您選擇的組態將取決於每個應用程式的使用案例。我們建議您至少將最低權限套用至管理操作，並採取動作來保護應用程式和使用者秘密。您可以採取的另一個進階但有效的步驟是設定 AWS WAF Web ACLs 並將其套用至您的使用者集區。

在網路層級保護您的使用者集區

AWS WAF Web ACLs 可以保護您使用 Amazon Cognito 建置之身分驗證機制的效能和成本。透過 Web ACLs，您可以在 API 和受管登入請求前實作護欄。Web ACLs 會建立網路和應用程式層篩選條件，以根據您設計的規則捨棄流量或需要 CAPTCHA。在請求符合 Web ACL 規則中的資格之前，不會將請求傳遞到您的 Amazon Cognito 資源。如需詳細資訊，請參閱 [AWS WAF Web ACLs](#)。

防止簡訊濫用

當您允許在使用者集區中公開註冊時，您可以使用 Amazon Cognito 在簡訊中傳送的代碼來設定帳戶驗證。SMS 訊息可以與不需要的活動相關聯，並增加您的 AWS 帳單。設定您的基礎設施在發生詐騙時能夠彈性地傳送簡訊。如需詳細資訊，請參閱部落格中的下列文章 [AWS](#)。

- [使用 Amazon Cognito 使用者集區降低使用者註冊詐騙和 SMS 輸出的風險](#)
- [防禦 SMS Pumping：協助對抗人工膨脹流量的新 AWS 功能](#)

了解公有身分驗證

Amazon Cognito 使用者集區具有客戶身分和存取管理 (CIAM) 功能，可支援一般大眾可以註冊使用者帳戶並存取應用程式的使用案例。當使用者集區允許自助註冊時，可以從公有網際網路請求使用者帳戶。自助式請求來自 [SignUp](#) 和 [InitiateAuth](#) 等 API 操作，以及來自使用者與受管登入的互動。您可以設定使用者集區，以減輕可能來自公有請求的濫用，或完全停用公有身分驗證操作。

以下是您可以在使用者集區和應用程式用戶端中管理公有和內部身分驗證請求的一些方式。

影響公有使用者集區存取的使用者集區設定範例

設定	可用的選項	在 上設定	對公有身分驗證的影響	主控台設定	API 操作和參數
自助式註冊	允許使用者註冊 帳戶或以管理員身分建立使用者帳戶。	使用者集區	防止公開註冊	註冊 – 自助式註冊	CreateUserPool 、 UpdateUserPool AdminCreateUserConfig – AllowAdminCreateUserOnly
管理員確認	傳送確認碼給新使用者，或要求管理員確認。	使用者集區	防止在沒有管理員動作的情況下確認註冊	註冊 – Cognito 協助的驗證和確認	CreateUserPool 、 UpdateUserPool

設定	可用的選項	在 上設定	對公有身分驗證的影響	主控台設定	API 操作和參數
					AccountRecoverySettings – admin_only
使用者公開	在登入和密碼重設或防止公開時傳遞「找不到使用者」訊息。	使用者集區	防止猜測登入名稱、電子郵件地址或電話號碼	應用程式用戶端 – 防止使用者存在錯誤	CreateUserPoolClient 、 UpdateUserPoolClient PreventUserExistenceErrors
Client secret (用戶端密碼)	需要或不需要註冊、登入、密碼重設時的秘密雜湊	應用程式用戶端	防止來自未經授權來源的身分驗證請求	應用程式用戶端 – 用戶端秘密	CreateUserPoolClient GenerateSecret
Web ACLs	啟用或停用身分驗證請求的網路防火牆	使用者集區	根據管理員定義的請求特性和 IP 地址規則來限制或防止存取	AWS WAF – WAF 設定	AssociateWebACLResourceArn

設定	可用的選項	在 上設定	對公有身分驗證的影響	主控台設定	API 操作和參數
外部 IdP	允許第三方 IdPs、使用者集區目錄或兩者中的使用者登入	應用程式用戶端	從註冊和登入中排除 本機使用者 或 聯合身分使用者 。	應用程式用戶端 – 身分提供者	CreateUserPoolClient 、 UpdateUserPoolClient Supported Identity Providers
授權伺服器	託管或不託管用於身分驗證的公有網頁	使用者集區	關閉公有網頁並僅允許以 SDK 為基礎的身分驗證	網域	CreateUserPoolDomain 建立任何使用者集區網域都會提供公有網頁。
威脅防護	啟用或停用監控惡意活動或不安全密碼的跡象	使用者集區或應用程式用戶端	當使用者顯示入侵指標時，可以自動封鎖登入或要求 MFA	威脅防護 – 保護設定	SetRiskConfiguration 的參數會SetRiskConfiguration 定義您的威脅防護設定。

使用用戶端秘密保護機密用戶端

用戶端秘密是與[應用程式用戶端](#)相關聯的選用字串。對具有用戶端秘密的應用程式用戶端的所有身分驗證請求都必須包含從使用者名稱、用戶端 ID 和用戶端秘密產生的[秘密雜湊](#)。不知道用戶端秘密的人會從頭關閉您的應用程式。

不過，用戶端秘密有限制。如果您在公有用戶端軟體中嵌入用戶端秘密，您的用戶端秘密會開放檢查。這可讓您在應用程式用戶端中建立使用者、提交密碼重設請求，以及執行其他操作。只有在應用程式是唯一可存取秘密的實體時，才必須實作用戶端秘密。一般而言，這在伺服器端機密用戶端應用程式中是可行的。對於需要用戶端秘密的[M2M 應用程式](#)也是如此。將用戶端秘密存放在加密的本機儲存或中 AWS Secrets Manager。切勿讓您的用戶端秘密出現在公有網際網路上。

保護其他秘密

您使用 Amazon Cognito 使用者集區的身分驗證系統可能會處理私有資料、密碼和 AWS 登入資料。以下是處理應用程式可能存取之秘密的一些最佳實務。

密碼

使用者登入您的應用程式時，可能會輸入密碼。Amazon Cognito 具有重新整理權杖，您的應用程式可以用來繼續過期的使用者工作階段，而無需新的密碼提示。請勿在本機儲存體中放置任何密碼或密碼雜湊。設計您的應用程式將密碼視為不透明，並僅將密碼傳遞至您的使用者集區。

最佳實務是使用 [WebAuthn 通行金鑰](#) 實作無密碼身分驗證。如果您必須實作密碼，請使用[安全遠端密碼 \(SRP\) 身分驗證流程](#)和[多重要素驗證 \(MFA\)](#)。

AWS 登入資料

管理身分驗證和使用集區管理操作需要使用 AWS 登入資料進行身分驗證。若要在應用程式中實作這些操作，請授予[暫時 AWS 登入](#)資料的安全存取權。僅將登入資料存取權授予在您控制之伺服器元件上執行的應用程式。請勿將具有 AWS 登入資料的應用程式放在公有版本控制系統上，例如 GitHub。請勿在公有用戶端應用程式中編碼 AWS 登入資料。

PKCE 程式碼驗證器

[Code Exchange](#) 或 [PKCE 的證明金鑰](#)適用於 OpenID Connect (OIDC) 授權碼授予您的使用者集區授權伺服器。應用程式在請求授權碼時，會與您的使用者集區共用程式碼驗證器秘密。若要交換字符的授權碼，用戶端必須再次確認他們知道代碼驗證者。此實務可防止使用截獲的授權碼發出權杖。

用戶端必須在每個授權請求中產生新的隨機程式碼驗證程式。使用靜態或可預測程式碼驗證器表示攻擊者只需要攔截硬式編碼驗證器和授權碼。設計您的應用程式，使其不會向使用者公開程式碼驗證器值。

使用者集區管理最低權限

IAM 政策可以定義主體對 Amazon Cognito 使用者集區管理和身分驗證操作的存取層級。例如：

- 向 Web 伺服器授予使用管理 API 操作進行身分驗證的許可。
- 對於在 中管理使用者集區 AWS IAM Identity Center 的使用者 AWS 帳戶，授予使用者集區維護和報告的許可。

Amazon Cognito 中的資源精細程度僅限於[兩種資源類型](#)，用於 IAM 政策用途：使用者集區和身分集區。請注意，您無法套用管理個別應用程式用戶端的許可。設定使用者集區，了解您授予的許可在所有應用程式用戶端中都有效。當您的組織有多個應用程式租用戶，而且您的安全模型需要在租用戶之間分離管理責任時，請[實作每個使用者集區一個租用戶的多租用戶](#)。

雖然您可以建立具有等使用者身分驗證操作許可的 IAM 政策 `InitiateAuth`，但這些許可沒有作用。[公有和字符授權的 API 操作](#) 不受 IAM 許可的約束。在可用的使用者集區身分驗證操作中，您只能將許可授予管理伺服器端操作，例如 `AdminInitiateAuth`。

您可以使用最低權限 Action 清單來限制使用者集區管理層級。下列範例政策適用於可以管理 IdPs、資源伺服器、應用程式用戶端和使用者集區網域，但不能管理使用者或使用者集區的管理員。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UserPoolClientAdministrator",
      "Action": [
        "cognito-idp:CreateIdentityProvider",
        "cognito-idp:CreateManagedLoginBranding",
        "cognito-idp:CreateResourceServer",
        "cognito-idp:CreateUserPoolDomain",
        "cognito-idp>DeleteIdentityProvider",
        "cognito-idp>DeleteResourceServer",
        "cognito-idp>DeleteUserPoolDomain",
        "cognito-idp:DescribeIdentityProvider",
        "cognito-idp:DescribeManagedLoginBranding",
        "cognito-idp:DescribeManagedLoginBrandingByClient",

```

```

        "cognito-idp:DescribeResourceServer",
        "cognito-idp:DescribeUserPool",
        "cognito-idp:DescribeUserPoolClient",
        "cognito-idp:DescribeUserPoolDomain",
        "cognito-idp:GetIdentityProviderByIdentifier",
        "cognito-idp:GetUICustomization",
        "cognito-idp:ListIdentityProviders",
        "cognito-idp:ListResourceServers",
        "cognito-idp:ListUserPoolClients",
        "cognito-idp:ListUserPools",
        "cognito-idp:SetUICustomization",
        "cognito-idp:UpdateIdentityProvider",
        "cognito-idp:UpdateManagedLoginBranding",
        "cognito-idp:UpdateResourceServer",
        "cognito-idp:UpdateUserPoolClient",
        "cognito-idp:UpdateUserPoolDomain"
    ],
    "Effect": "Allow",
    "Resource": "arn:aws:cognito-idp:us-west-2:123456789012:userpool/us-
west-2_EXAMPLE"
}
]
}

```

下列範例政策會將使用者和群組管理和身分驗證授予伺服器端應用程式。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UserAdminAuthN",
      "Action": [
        "cognito-idp:AdminAddUserToGroup",
        "cognito-idp:AdminConfirmSignUp",
        "cognito-idp:AdminCreateUser",
        "cognito-idp:AdminDeleteUser",
        "cognito-idp:AdminDeleteUserAttributes",
        "cognito-idp:AdminDisableProviderForUser",
        "cognito-idp:AdminDisableUser",
        "cognito-idp:AdminEnableUser",
        "cognito-idp:AdminForgetDevice",
        "cognito-idp:AdminGetDevice",
        "cognito-idp:AdminGetUser",

```

```

    "cognito-idp:AdminInitiateAuth",
    "cognito-idp:AdminLinkProviderForUser",
    "cognito-idp:AdminListDevices",
    "cognito-idp:AdminListGroupsForUser",
    "cognito-idp:AdminListUserAuthEvents",
    "cognito-idp:AdminRemoveUserFromGroup",
    "cognito-idp:AdminResetUserPassword",
    "cognito-idp:AdminRespondToAuthChallenge",
    "cognito-idp:AdminSetUserMFAPreference",
    "cognito-idp:AdminSetUserPassword",
    "cognito-idp:AdminSetUserSettings",
    "cognito-idp:AdminUpdateAuthEventFeedback",
    "cognito-idp:AdminUpdateDeviceStatus",
    "cognito-idp:AdminUpdateUserAttributes",
    "cognito-idp:AdminUserGlobalSignOut",
    "cognito-idp:AssociateSoftwareToken",
    "cognito-idp:ListGroups",
    "cognito-idp:ListUsers",
    "cognito-idp:ListUsersInGroup",
    "cognito-idp:RevokeToken",
    "cognito-idp:UpdateGroup",
    "cognito-idp:VerifySoftwareToken"
  ],
  "Effect": "Allow",
  "Resource": "arn:aws:cognito-idp:us-west-2:123456789012:userpool/us-
west-2_EXAMPLE"
}
]
}

```

保護和驗證字符

字符可以包含群組成員資格和使用者屬性的內部參考，您可能不想向最終使用者揭露這些屬性。請勿在本機儲存體中存放 ID 和存取權杖。重新整理字符會使用只有您的使用者集區可以存取的金鑰進行加密，並且對使用者和應用程式而言是不透明的。當使用者登出，或當您判斷基於安全考量而不需要保留使用者的工作階段時，[請撤銷重新整理權杖](#)。

使用存取權杖僅授權存取獨立驗證權杖有效和未過期的系統。如需驗證資源，請參閱[驗證 JSON Web 權杖](#)。

決定您要信任的身分提供者

當您使用 [SAML](#) 或 [OIDC](#) 身分提供者 (IdPs) 設定使用者集區時，IdPs 可以建立新的使用者、設定使用者屬性，以及存取您的應用程式資源。SAML 和 OIDC 提供者通常用於 business-to-business (B2B) 或企業案例，其中您或您的直接客戶控制提供者的成員資格和組態。

[社交提供者](#) 會將使用者帳戶提供給網際網路上的任何人，並且比企業提供者更少。只有在您準備好允許公有客戶登入和存取應用程式中的資源時，才能在您的應用程式用戶端中啟用社交 IdPs。

了解範圍對存取使用者設定檔的影響

您可以向使用者集區授權伺服器請求身分驗證請求中的存取控制範圍。這些範圍可以授予使用者外部資源的存取權，而且可以授予使用者檢視和修改自己的使用者設定檔的存取權。設定您的應用程式用戶端，以支援應用程式操作所需的最低範圍。

`aws.cognito.signin.user.admin` 範圍存在於 SDK 身分驗證使用 [InitiateAuth](#) 等操作發行的所有存取字符中。它被指定用於應用程式中的使用者設定檔自助服務操作。您也可以向授權伺服器請求此範圍。權杖授權的操作需要此範圍，例如 [UpdateUserAttributes](#) 和 [GetUser](#)。這些操作的效果受限於您應用程式用戶端的讀取和寫入許可。

`openid`、`email`、`profile` 和 `phone` 範圍會授權請求給使用者集區授權伺服器上 [userInfo](#) 端點的。它們定義端點可以傳回的屬性。在沒有其他 `openid` 範圍的情況下請求時，範圍會傳回所有可用的屬性，但當您在請求中請求更多範圍時，回應會縮小至其他範圍所代表的屬性。此 `openid` 範圍也會指出 ID 權杖的請求；當您從對的請求中省略此範圍時 [授權端點](#)，Amazon Cognito 只會發出存取權杖，並在適用時發出重新整理權杖。如需詳細資訊，請參閱《》中的 OpenID Connect 範圍 [應用程式用戶端條款](#)。

清理使用者屬性的輸入

最終可能做為交付方法和使用者名稱的使用者屬性，例如 `email`，具有 [格式限制](#)。其他屬性可以有字串、布林值或數字資料類型。字串屬性值支援各種輸入。設定您的應用程式以防止嘗試將不需要的資料寫入您的使用者目錄或 Amazon Cognito 提供給使用者的訊息。將使用者提交的字串屬性值提交至 Amazon Cognito 之前，請對應用程式中的使用者提交字串屬性值執行用戶端驗證。

使用者集區會根據您指定的屬性映射，將屬性從 IdPs [映射](#) 到您的使用者集區。僅將安全且可預測的 IdP 屬性映射至使用者集區字串屬性。

使用 Amazon Cognito 使用者集區進行身分驗證

Amazon Cognito 包含幾種對您的使用者進行身分驗證的方法。使用者可以使用密碼和 WebAuthn 通行金鑰登入。Amazon Cognito 可以在電子郵件或簡訊中傳送一次性密碼給他們。您可以實作 Lambda 函數來協調自己的一系列挑戰和回應。這些是身分驗證流程。在身分驗證流程中，使用者提供秘密，Amazon Cognito 會驗證秘密，然後發出 JSON Web 字符 (JWTs) 供應用程式使用 OIDC 程式庫處理。在本章中，我們將討論如何在各種應用程式環境中為各種身分驗證流程設定使用者集區和應用程式用戶端。您將了解使用受管登入託管登入頁面的選項，以及在 AWS SDK 中建置自己的邏輯和前端。

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。

為了驗證使用者的身分，Amazon Cognito 支援除了電子郵件和簡訊一次性密碼和通行金鑰等密碼之外，還包含挑戰類型的身分驗證流程。

主題

- [實作身分驗證流程](#)
- [使用使用者集區進行身分驗證的須知事項](#)
- [範例身分驗證工作階段](#)
- [設定受管登入的身分驗證方法](#)
- [在 SDKs 中 AWS 管理身分驗證方法](#)
- [身分驗證流程](#)
- [API 和 SDK 身分驗證的授權模型](#)

實作身分驗證流程

無論您是實作[受管登入](#)，還是使用 AWS 開發套件進行身分驗證的[自訂建置應用程式前端](#)，都必須為您要實作的身分驗證類型設定應用程式用戶端。下列資訊說明[應用程式用戶端](#)和應用程式中身分驗證流程的設定。

App client supported flows

您可以在 Amazon Cognito 主控台或使用 AWS SDK 中的 API，為您的應用程式用戶端設定支援的流程。將應用程式用戶端設定為支援這些流程之後，您可以在應用程式中部署它們。

下列程序會使用 Amazon Cognito 主控台設定應用程式用戶端的可用身分驗證流程。

為身分驗證流程設定應用程式用戶端（主控台）

1. 登入 AWS 並導覽至 [Amazon Cognito 使用者集區主控台](#)。選擇使用者集區或建立新的集區。
2. 在您的使用者集區組態中，選取應用程式用戶端選單。選擇應用程式用戶端或建立新的用戶端。
3. 在應用程式用戶端資訊下，選取編輯。
4. 在應用程式用戶端流程下，選擇您要支援的身分驗證流程。

設定應用程式用戶端進行身分驗證流程 (API/SDK)

若要使用 Amazon Cognito API 設定應用程式用戶端的可用身分驗證流程，請在 [CreateUserPoolClient](#) 或 [UpdateUserPoolClient](#) 請求ExplicitAuthFlows中設定的值。以下是將安全遠端密碼 (SRP) 和選擇型身分驗證佈建給用戶端的範例。

```
"ExplicitAuthFlows": [  
    "ALLOW_USER_AUTH",  
    "ALLOW_USER_SRP_AUTH"  
]
```

當您設定應用程式用戶端支援的流程時，您可以指定下列選項和 API 值。

應用程式用戶端流程支援

身分驗證流程	相容性	主控台	API
以選擇為基礎的身分驗證	伺服器端、用戶端	在登入時選取身分驗證類型	ALLOW_USER_AUTH
使用持久性密碼登入	用戶端	使用使用者名稱和密碼登入	ALLOW_USER_PASSWORD_AUTH
使用持久性密碼和完全承載登入	伺服器端、用戶端	使用安全的遠端密碼 (SRP) 登入	ALLOW_USER_SRP_AUTH

身分驗證流程	相容性	主控台	API
重新整理權杖	伺服器端、用戶端	從現有的已驗證工作階段取得新的使用者字串	ALLOW_REFRESH_TOKEN_AUTH
伺服器端身分驗證	伺服器端	使用伺服器端管理登入資料登入	ALLOW_ADMIN_USER_PASSWORD_AUTH
自訂身分驗證	伺服器端和用戶端自訂建置的應用程式。與受管登入不相容。	使用來自 Lambda 觸發程序的自訂身分驗證流程登入	ALLOW_CUSTOM_AUTH

Implement flows in your application

受管登入會自動在登入頁面中提供您設定的身分驗證選項。在自訂建置的應用程式中，使用初始流程的宣告開始身分驗證。

- 若要從使用者的流程選項清單中選擇，請使用USER_AUTH流程宣告以[選擇為基礎的身分驗證](#)。此流程具有用戶端型身分驗證流程中無法使用的可用身分驗證方法，例如[通行金鑰](#)和[無密碼](#)身分驗證。
- 若要選擇您的身分驗證流程，請使用應用程式[用戶端中可用的任何其他流程宣告用戶端型身分驗證](#)。

當您登入使用者時，[InitiateAuth](#) 或 [AdminInitiateAuth](#) 請求的內文必須包含 AuthFlow 參數。

以選擇為基礎的身分驗證：

```
"AuthFlow": "USER_AUTH"
```

使用 SRP 的用戶端型身分驗證：

```
"AuthFlow": "USER_SRP_AUTH"
```


使用使用者集區進行身分驗證的須知事項

在設計具有 Amazon Cognito 使用者集區的身分驗證模型時，請考慮以下資訊。

受管登入和託管 UI 中的身分驗證流程

[受管登入](#)和傳統託管 UI 有不同的身分驗證選項。您只能在受管登入中執行無密碼和通行金鑰身分驗證。

自訂身分驗證流程僅適用於 AWS SDK 身分驗證

您無法使用受管登入或傳統託管 UI 執行自訂身分驗證流程，或使用 [Lambda 觸發程序進行自訂身分驗證](#)。自訂身分驗證可用於 [SDK AWS SDKs 身分驗證](#)。

外部身分提供者 (IdP) 登入的受管登入

您無法透過[使用 AWS SDKs 進行身分驗證中的第三方 IdPs](#) 登入使用者。您必須實作受管登入或傳統託管 UI、重新導向至 IdPs，然後在應用程式中使用 OIDC 程式庫處理產生的身分驗證物件。如需受管登入的詳細資訊，請參閱 [使用者集區受管登入](#)。

無密碼身分驗證對其他使用者功能的影響

在使用者集區和應用程式用戶端中使用[一次性密碼](#)或[通行金鑰](#)啟用無密碼登入，會影響使用者建立和遷移。當無密碼登入處於作用中狀態時：

1. 管理員可以建立沒有密碼的使用者。預設邀請訊息範本會變更為不再包含{###}密碼預留位置。如需詳細資訊，請參閱[建立使用者帳戶為管理員](#)。
2. 對於以 SDK 為基礎的 [SignUp](#) 操作，使用者註冊時不需要提供密碼。受管登入和託管 UI 在註冊頁面中需要密碼，即使允許無密碼身分驗證。如需詳細資訊，請參閱[註冊及確認使用者帳戶](#)。
3. 如果從 CSV 檔案匯入的使用者屬性包含可用的無密碼登入選項的電子郵件地址或電話號碼，則可以使用無密碼選項立即登入，而無需重設密碼。如需詳細資訊，請參閱[從 CSV 檔案將使用者匯入使用者集區](#)。
4. 無密碼身分驗證不會叫用[使用者遷移 Lambda 觸發程序](#)。
5. 使用無密碼第一要素登入的使用者無法將[多重要素驗證 \(MFA\)](#) 要素新增至其工作階段。只有密碼型身分驗證流程支援 MFA。

通行密鑰依賴方 URLs 不能在公有字尾清單中

您可以使用您擁有的網域名稱，例如 `www.example.com`，做為通行密鑰組態中的依賴方 (RP) ID。此組態旨在支援在您擁有的網域上執行的自訂建置應用程式。[公有字尾清單](#)或 PSL 包含受保護的高階網域。當您嘗試將 RP URL 設定為 PSL 上的網域時，Amazon Cognito 會傳回錯誤。

主題

- [身分驗證工作階段流程持續時間](#)
- [失敗登入嘗試的鎖定行為](#)

身分驗證工作階段流程持續時間

視使用者集區的功能而定，在應用程式從 Amazon Cognito 擷取字符 `RespondToAuthChallenge` 之前 `InitiateAuth`，您最終可以回應多個挑戰。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。受管登入會將多因素身分驗證的工作階段持續時間設定為 3 分鐘，密碼重設碼則為 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. 對於 SMS 和電子郵件 MFA 代碼，將身分驗證流程工作階段持續時間的值變更為您想要的有效持續時間，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。

User pools 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。

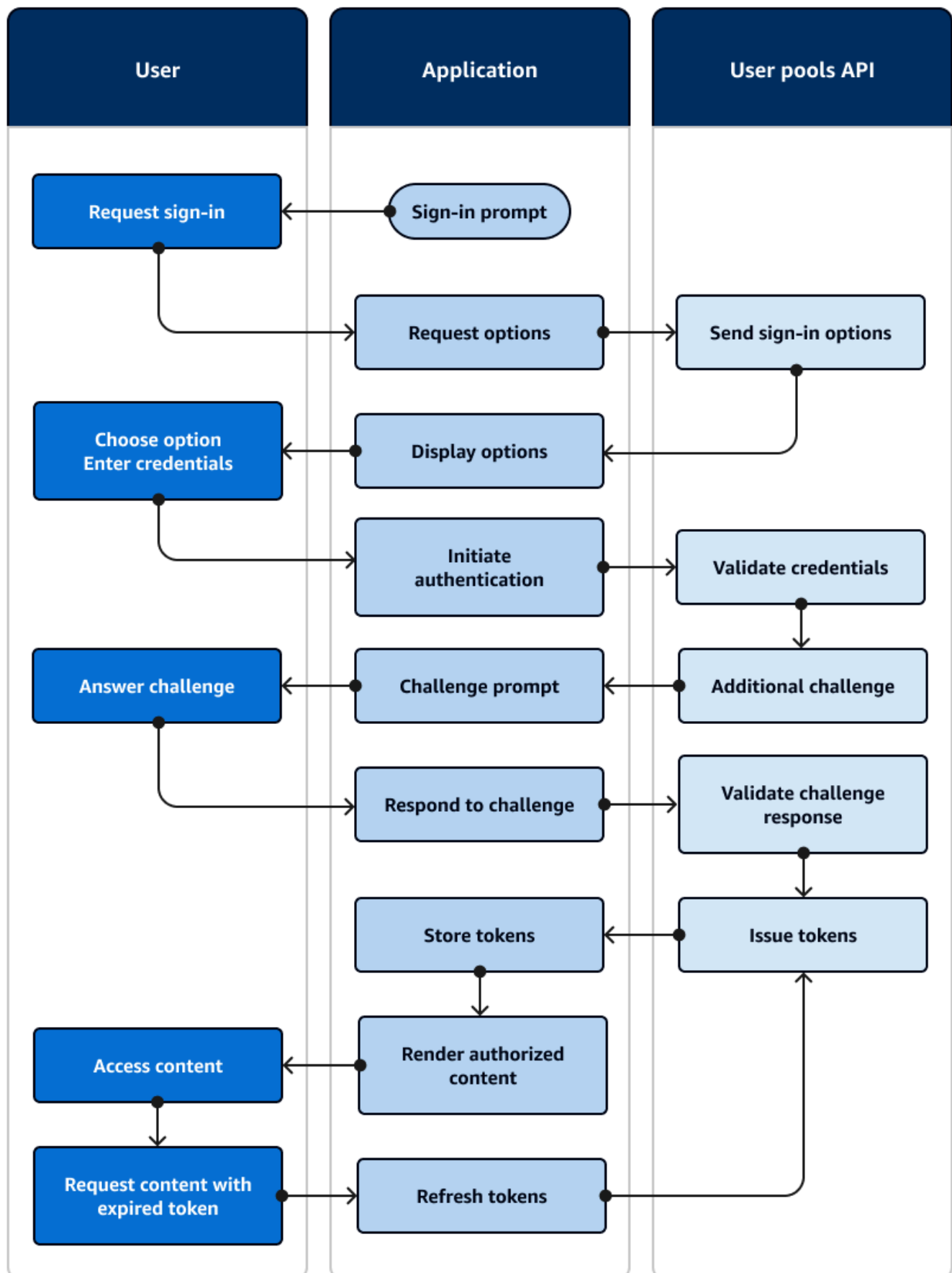
如需應用程式用戶端的詳細資訊，請參閱 [使用應用程式用戶端的應用程式特定設定](#)。

失敗登入嘗試的鎖定行為

使用密碼進行五次未驗證或 IAM 授權的登入嘗試後，Amazon Cognito 會鎖定您的使用者一秒鐘。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。

範例身分驗證工作階段

下圖和step-by-step指南說明使用者登入應用程式的典型案例。範例應用程式會向使用者顯示數個登入選項。他們透過輸入登入資料、提供其他身分驗證因素並登入來選取一個。



使用登入頁面將應用程式想像成，使用者可以使用使用者名稱和密碼登入、在電子郵件訊息中請求一次性代碼，或選擇指紋選項。

1. 登入提示：您的應用程式會顯示含有登入按鈕的主畫面。
2. 請求登入：使用者選取登入。從 Cookie 或快取中，您的應用程式會擷取其使用者名稱，或提示他們輸入。
3. 請求選項：您的應用程式使用USER_AUTH流程的 InitiateAuth API 請求請求使用者的登入選項，為使用者請求可用的登入方法。
4. 傳送登入選項：Amazon Cognito 以 PASSWORD、EMAIL_OTP和 回應WEB_AUTHN。回應包含工作階段識別符，供您在下一個回應中重播。
5. 顯示選項：您的應用程式會顯示使用者輸入使用者名稱和密碼、取得一次性代碼或掃描指紋的 UI 元素。
6. 選擇選項/輸入登入資料：使用者輸入使用者名稱和密碼。
7. 啟動身分驗證：您的應用程式會使用 RespondToAuthChallenge API 請求來提供使用者的登入資訊，以確認使用者名稱-密碼登入，並提供使用者名稱和密碼。
8. 驗證登入資料：Amazon Cognito 會確認使用者的登入資料。
9. 其他挑戰：使用者已使用驗證器應用程式設定多重要素驗證。Amazon Cognito 傳回SOFTWARE_TOKEN_MFA挑戰。
10. 挑戰提示：您的應用程式會顯示從使用者的驗證器應用程式請求以時間為基礎的一次性密碼 (TOTP) 的表單。
11. 答案挑戰：使用者提交 TOTP。
12. 回應挑戰：在另一個RespondToAuthChallenge請求中，您的應用程式會提供使用者的 TOTP。
13. 驗證挑戰回應：Amazon Cognito 會確認使用者的程式碼，並判斷您的使用者集區已設定為不會對目前使用者發出其他挑戰。
14. 問題字符：Amazon Cognito 會傳回 ID、存取和重新整理 JSON Web 字符 (JWTs)。使用者的初始身分驗證已完成。
15. 存放權杖：您的應用程式會快取使用者的權杖，以便其可以參考使用者資料、授權存取資源，並在權杖過期時更新權杖。
16. 轉譯授權內容：您的應用程式會根據使用者的身分和角色來決定使用者的 資源存取權，並交付應用程式內容。
17. 存取內容：使用者已登入並開始使用應用程式。
18. 使用過期權杖請求內容：稍後，使用者會請求需要授權的資源。使用者的快取字符已過期。
19. 重新整理字符：您的應用程式使用使用者儲存的重新整理字符提出InitiateAuth請求。

20.問題字符：Amazon Cognito 會傳回新 ID 並存取 JWTs。使用者的工作階段會安全地重新整理，而不需要額外的登入資料提示。

您可以使用[AWS Lambda 觸發來自訂使用者驗證](#)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。您可以使用[使用者遷移身分驗證流程](#)來讓使用者遷移成為可能，而無需您的使用者重設密碼。

設定受管登入的身分驗證方法

當您想要使用者登入、登出或重設密碼時，您可以叫用[受管登入頁面](#)，也就是用於使用者集區身分驗證的 Web 前端。在此模型中，您的應用程式匯入 OIDC 程式庫，以使用使用者集區受管登入頁面處理瀏覽器型身分驗證嘗試。使用者可用的身分驗證形式取決於使用者集區和應用程式用戶端的組態。在應用程式用戶端中實作ALLOW_USER_AUTH流程，Amazon Cognito 會提示使用者從可用的選項中選取登入方法。實作ALLOW_USER_PASSWORD_AUTH並指派 SAML 供應商，您的登入頁面會提示使用者輸入使用者名稱和密碼，或與其 IdP 連線。

Amazon Cognito 使用者集區主控台可協助您開始設定應用程式的受管登入身分驗證。當您建立新的使用者集區時，請指定您要開發的平台，主控台會提供範例，讓您實作具有入門程式碼的 OIDC 和 OAuth 程式庫，以實作登入和登出流程。您可以使用許多 OIDC 依賴方實作來建置受管登入。我們建議您盡可能使用[經認證的 OIDC 依賴方程式庫](#)。如需詳細資訊，請參閱[使用者集區入門](#)。

一般而言，OIDC 依賴方程式庫會預先檢查使用者集區的.well-known/openid-configuration端點，以判斷發行者 URLs例如字符端點和授權端點。最佳實務是，在您可以選擇的地方實作此自動探索行為。手動設定發行者端點可能會導致錯誤。例如，您可以變更使用者集區網域。的路徑openid-configuration不會連結到您的使用者集區網域，因此自動探索服務端點的應用程式會自動取得您的網域變更。

受管登入的使用者集區設定

您可能想要允許使用應用程式的多個供應商登入，或者您可能想要使用 Amazon Cognito 做為獨立的使用者目錄。您也可以收集使用者屬性、設定和提示 MFA，或要求電子郵件地址做為使用者名稱。您無法直接編輯受管登入和託管 UI 中的欄位。反之，使用者集區的組態會自動設定受管登入身分驗證流程的處理。

下列使用者集區組態項目會決定 Amazon Cognito 在受管登入和託管 UI 中呈現給使用者的身分驗證方法。

User pool options (Sign-in menu)

下列選項位於 Amazon Cognito 主控台中使用者集區的登入選單中。

Cognito 使用者集區登入選項

有使用者名稱的選項。您的受管登入和託管 UI 頁面僅接受您所選取格式的使用者名稱。例如，當您使用電子郵件作為唯一登入選項設定使用者集區時，您的受管登入頁面只會接受電子郵件格式的使用者名稱。

必要屬性

當您在使用者集區中視需要設定屬性時，受管登入會在使用者註冊時提示使用者該屬性的值。

選擇型登入的選項

在 中具有身分驗證方法的設定[以選擇為基礎的身分驗證](#)。在這裡，您可以開啟或關閉身分驗證方法，例如[通行金鑰](#)和[無密碼](#)。這些方法僅適用於具有 Lite 層以上[受管登入網域](#)和[功能計劃](#)的使用者集區。

多重要素驗證

受管登入和託管 UI 會處理 [MFA](#) 的註冊和身分驗證操作。當您的使用者集區需要 MFA 時，您的登入頁面會自動提示使用者設定其額外因素。他們也會提示具有 MFA 組態的使用者使用 MFA 代碼完成身分驗證。當您的使用者集區中的 MFA 關閉或選用時，您的登入頁面不會提示設定 MFA。

使用者帳戶復原

使用者集區的自助式[帳戶復原](#)設定會決定您的登入頁面是否顯示使用者可重設密碼的連結。

User pool options (Domain menu)

下列選項位於 Amazon Cognito 主控台中使用者集區的網域選單中。

網域

您選擇的使用者集區網域會設定使用者在叫用其瀏覽器進行身分驗證時所開啟連結的路徑。

品牌版本

您選擇的品牌版本會決定您的使用者集區網域是否顯示受管登入或託管 UI。

User pool options (Social and external providers menu)

下列選項位於 Amazon Cognito 主控台中使用者集區的社交和外部提供者選單中。

提供者

您新增至使用者集區的身分提供者 (IdPs) 可為使用者集區中的每個應用程式用戶端保持作用中或非作用中狀態。

App client options

下列選項位於 Amazon Cognito 主控台中使用者集區的應用程式用戶端選單中。若要檢閱這些選項，請從清單中選擇應用程式用戶端。

快速設定指南

快速設定指南具有適用於各種開發人員環境的程式碼範例。它們包含將受管登入身分驗證與您的應用程式整合所需的程式庫。

應用程式用戶端資訊

編輯此組態，為目前應用程式用戶端表示的應用程式設定指派的 IdPs。在受管登入頁面上，Amazon Cognito 會顯示使用者的選擇。這些選擇取決於指派的方法和 IdP。例如，如果您指派名為 MySAML 和本機使用者集區登入的 SAML 2.0 IdP，您的受管登入頁面會顯示身分驗證方法提示和的按鈕MySAML。

身分驗證設定

編輯此組態，為您的應用程式設定身分驗證方法。在受管登入頁面上，Amazon Cognito 會顯示使用者的選擇。這些選擇取決於使用者集區做為 IdP 的可用性，以及您指派的方法。例如，如果您指派以選擇為基礎的ALLOW_USER_AUTH身分驗證，您的受管登入頁面會顯示可用的選項，例如輸入電子郵件地址並使用通行金鑰登入。受管登入頁面也會轉譯所指派 IdPs 按鈕。

登入頁面

使用此索引標籤中可用的選項，設定受管登入或託管 UI 使用者互動頁面的視覺化效果。如需詳細資訊，請參閱[將品牌套用到受管登入頁面](#)。

在 SDKs 中 AWS 管理身分驗證方法

Amazon Cognito 使用者集區中的使用者可以使用各種初始登入選項或因素登入。對於某些因素，使用者可以使用多重要素驗證 (MFA) 進行追蹤。這些首要因素包括使用者名稱和密碼、一次性密碼、通行

密鑰和自訂身分驗證。如需詳細資訊，請參閱[身分驗證流程](#)。當您的應用程式具有內建 UI 元件並匯入 AWS SDK 模組時，您必須建置應用程式邏輯以進行身分驗證。您必須從兩種主要方法中選擇一種，並從該方法中選擇您要實作的身分驗證機制。

您可以實作以用戶端為基礎的身分驗證，其中您的應用程式或用戶端會事先宣告身分驗證的類型。您的另一個選項是選擇型身分驗證，您的應用程式會收集使用者名稱，並請求使用者可用的身分驗證類型。您可以根據您的需求，在相同的應用程式中一起實作這些模型，或在應用程式用戶端之間分割。每個方法都有其獨有的功能，例如以用戶端為基礎的自訂身分驗證和以選擇為基礎的無密碼身分驗證。

在使用使用者集區 API 的 AWS SDK 實作執行身分驗證的自訂應用程式中，您必須建構 API 請求，以符合使用者集區組態、應用程式用戶端組態和用戶端偏好設定。以 AuthFlow 的開頭的 `InitiateAuth` 工作階段會以 `USER_AUTH` 開始以選擇為基礎的身分驗證。Amazon Cognito 會以偏好的身分驗證方法或選項清單的挑戰來回應您的 API。以 AuthFlow 開頭的工作階段 `CUSTOM_AUTH` 會直接使用 Lambda 觸發程序進行自訂身分驗證。

有些身分驗證方法固定為兩種流程類型的其中之一，而有些方法在這兩種流程中都可用。

主題

- [以選擇為基礎的身分驗證](#)
- [用戶端型身分驗證](#)

以選擇為基礎的身分驗證

您的應用程式可以在以選擇為基礎的身分驗證中請求下列身分驗證方法。在 [InitiateAuth](#) 或 [AdminInitiateAuth](#) 的 `PREFERRED_CHALLENGE` 參數中宣告這些選項，或在 [RespondToAuthChallenge](#) 或 [AdminRespondToAuthChallenge](#) 的 `ChallengeName` 參數中宣告這些選項。

1. EMAIL_OTP 和 SMS_OTP

[使用一次性密碼的無密碼登入](#)

2. WEB_AUTHN

[使用 WebAuthn 通行金鑰進行無密碼登入](#)

3. PASSWORD

[使用持久性密碼登入](#)

[使用持久性密碼和安全承載登入](#)

登入後的 MFA

若要在其 API 內容中檢閱這些選項，請參閱 [RespondToAuthChallenge](#) ChallengeName 中的。

以選擇為基礎的登入會發出挑戰來回應您的初始請求。此挑戰會驗證請求的選項是否可用，或提供可用的選項清單。您的應用程式可以向使用者顯示這些選擇，然後輸入他們偏好的登入方法的登入資料，並在挑戰回應中繼續進行身分驗證。

您可以在身分驗證流程中使用下列以選擇為基礎的選項。此類型的所有請求都需要您的應用程式先收集使用者名稱，或從快取中擷取。

1. USERNAME 僅請求具有 AuthParameters 的選項。Amazon Cognito 傳回 SELECT_CHALLENGE 挑戰。從那裡，您的應用程式可以提示使用者選取挑戰，並將此回應傳回至您的使用者集區。
2. 請求使用 AuthParameters 的偏好挑戰 PREFERRED_CHALLENGE，以及偏好挑戰的參數，如果有的話。例如，如果您請求 PREFERRED_CHALLENGE 的 PASSWORD_SRP，您也必須包含 SRP_A。如果您的使用者、使用者集區和應用程式用戶端都已針對偏好的挑戰設定，Amazon Cognito 會回應該挑戰的下一個步驟，例如 PASSWORD_VERIFIER 流程或 EMAIL_OTP 和 PASSWORD_SRP SMS_OTP 流程中的 [CodeDeliveryDetails](#)。如果無法使用偏好的挑戰，Amazon Cognito 會回應 SELECT_CHALLENGE 和可用的挑戰清單。
3. 先在中簽署使用者，然後請求其選擇型身分驗證選項。具有登入使用者的存取權杖的 [GetUserAuthFactors](#) 請求會傳回其可用的選擇型身分驗證因素及其 MFA 設定。使用此選項，使用者可以先使用使用者名稱和密碼登入，然後啟用不同形式的身分驗證。您也可以使用此操作，為已使用偏好挑戰登入的使用者檢查其他選項。

若要設定您的應用程式用戶端進行以選擇為基礎的身分驗證，請將 ALLOW_USER_AUTH 新增至允許的身分驗證流程。您還必須選擇想要在使用者集區組態中允許的以選擇為基礎的因素。下列程序說明如何選擇以選擇為基礎的身分驗證因素。

Amazon Cognito console

在使用者集區中設定以選擇為基礎的身分驗證選項

1. 登入 AWS 並導覽至 [Amazon Cognito 使用者集區主控台](#)。選擇使用者集區或建立新的集區。
2. 在您的使用者集區組態中，選取登入功能表。尋找以選擇為基礎的登入選項，然後選擇編輯。
3. 密碼選項一律可用。這包括 PASSWORD 和 PASSWORD_SRP 流程。選取您要新增至使用者選項的其他選項。您可以新增的 PasskeyWEB_AUTHN、的電子郵件訊息一次性密碼 EMAIL_OTP，以及的簡訊一次性密碼 SMS_OTP。

4. 選擇儲存變更。

API/SDK

下列部分 [CreateUserPool](#) 或 [UpdateUserPool](#) 請求內文會設定選擇型身分驗證的所有可用選項。

```
"Policies": {
  "SignInPolicy": {
    "AllowedFirstAuthFactors": [
      "PASSWORD",
      "WEB_AUTHN",
      "EMAIL_OTP",
      "SMS_OTP"
    ]
  }
},
```

用戶端型身分驗證

用戶端型身分驗證支援下列身分驗證流程。在 [InitiateAuth](#) 或 [AdminInitiateAuth](#) 的 AuthFlow 參數中宣告這些選項。

1. USER_PASSWORD_AUTH 和 ADMIN_USER_PASSWORD_AUTH

[使用持久性密碼登入](#)

[登入後的 MFA](#)

此身分驗證流程等同於PASSWORD以選擇為基礎的身分驗證。

2. USER_SRP_AUTH

[使用持久性密碼和安全承載登入](#)

[登入後的 MFA](#)

此身分驗證流程等同於PASSWORD_SRP以選擇為基礎的身分驗證。

3. REFRESH_TOKEN_AUTH

[重新整理權杖](#)

此身分驗證流程僅適用於以用戶端為基礎的身分驗證。

4. CUSTOM_AUTH

自訂身分驗證

此身分驗證流程僅適用於以用戶端為基礎的身分驗證。

透過以用戶端為基礎的身分驗證，Amazon Cognito 會假設您已決定使用者在開始身分驗證流程之前，要如何進行身分驗證。判斷使用者想要提供的登入因素的邏輯必須使用預設設定或自訂提示來決定，然後在向使用者集區提出的第一個請求中宣告。InitiateAuth 請求會宣告直接對應至其中一個所列選項AuthFlow的登入，例如 USER_SRP_AUTH。透過此宣告，請求也包含開始身分驗證的參數，例如 USERNAME、SECRET_HASH和 SRP_A。Amazon Cognito 可能會針對 PASSWORD_VERIFIER SRP 或 TOTP MFA SOFTWARE_TOKEN_MFA的密碼登入等其他挑戰追蹤此請求。

若要設定應用程式用戶端進行用戶端型身分驗證，ALLOW_USER_AUTH請將 以外的任何身分驗證流程新增至允許的身分驗證流程。範例為

ALLOW_USER_PASSWORD_AUTH、ALLOW_CUSTOM_AUTH、ALLOW_REFRESH_TOKEN_AUTH。若要允許用戶端型身分驗證流程，不需要額外的使用者集區組態。

身分驗證流程

使用 Amazon Cognito 使用者集區的身分驗證程序，最適合描述為使用者進行初始選擇、提交登入資料，以及回應其他挑戰的流程。當您在應用程式中實作受管登入身分驗證時，Amazon Cognito 會管理這些提示和挑戰的流程。當您在應用程式後端使用 AWS SDK 實作流程時，您必須建構請求的邏輯、提示使用者輸入，以及回應挑戰。

身為應用程式管理員，您的使用者特性、安全需求和授權模型有助於判斷您希望如何允許使用者登入。自問以下問題。

- 我想要允許使用者使用來自[其他身分提供者 \(IdPs\)](#)登入資料登入嗎？
- [使用者名稱和密碼](#)是否有足夠的身分驗證？
- 是否可以攔截我的使用者名稱密碼身分驗證請求？我想要我的應用程式傳輸密碼，還是[使用雜湊和 salt 交涉身分驗證](#)？
- 我想要允許使用者略過密碼項目，[並接收登入的一次性密碼](#)嗎？
- 我想要允許使用者使用[指紋、臉部或硬體安全金鑰](#)登入？
- 如果有的話，我何時需要[多重要素驗證 \(MFA\)](#)？
- 是否要[保留使用者工作階段，而不重新提示登入資料](#)？

- 我想要將[授權模型延伸](#)到 Amazon Cognito 的內建功能之外嗎？

當您有這些問題的答案時，您可以了解如何啟用相關功能，並在應用程式提出的身分驗證請求中實作這些功能。

在您為使用者設定登入流程後，您可以使用請求 [GetUserAuthFactors](#) API 操作來檢查 MFA 和[以選擇為基礎的](#)身分驗證因素的目前狀態。此操作需要使用登入使用者的存取權杖進行授權。它會傳回使用者身分驗證因素和 MFA 設定。

主題

- [使用第三方 IdPs 登入](#)
- [使用持久性密碼登入](#)
- [使用持久性密碼和安全承載登入](#)
- [使用一次性密碼的無密碼登入](#)
- [使用 WebAuthn 通行金鑰進行無密碼登入](#)
- [登入後的 MFA](#)
- [重新整理權杖](#)
- [自訂身分驗證](#)
- [使用者遷移身分驗證流程](#)

使用第三方 IdPs 登入

Amazon Cognito 使用者集區可做為 IdPs 之間身分驗證工作階段的中繼代理程式，例如 Sign in with Apple、Login with Amazon 和 OpenID Connect (OIDC) 服務。此程序也稱為聯合登入或聯合身分驗證。聯合身分驗證不會利用您可以在應用程式用戶端中建置的任何身分驗證流程。相反地，您可以將設定的使用者集區 IdPs 指派給應用程式用戶端。當使用者在受管登入中選取其 IdP，或您的應用程式調用重新導向至其 IdP 登入頁面的工作階段時，就會發生聯合登入。

使用聯合登入，您可以將主要和 MFA 身分驗證因素委派給使用者的 IdP。除非您將聯合身分使用者連結至本機使用者，否則 Amazon Cognito 不會將本節中的其他進階流程新增至該使用者。[???](#)未連結的聯合身分使用者具有使用者名稱，但它們是映射屬性資料的存放區，通常不會用於與瀏覽器型流程無關的登入。

實作資源

- [使用第三方身分提供者登入使用者集區](#)

使用持久性密碼登入

在 Amazon Cognito 使用者集區中，每個使用者都有使用者名稱。這可能是電話號碼、電子郵件地址，或選擇或管理員提供的識別符。此類型的使用者可以使用其使用者名稱及其密碼登入，並選擇性地提供 MFA。使用者集區可以使用公有或 IAM 授權的 API 操作和 SDK 方法執行使用者名稱密碼登入。您的應用程式可以直接將密碼傳送到您的使用者集區以進行身分驗證。您的使用者集區會回應額外的挑戰或 JSON Web 字符 (JWTs)，這是身分驗證成功的結果。

Activate password sign-in

若要使用使用者名稱和密碼啟用[用戶端型身分驗證](#)，請將您的應用程式用戶端設定為允許。在 Amazon Cognito 主控台中，導覽至使用者集區組態中應用程式下的應用程式用戶端選單。若要允許用戶端行動或原生應用程式的純密碼登入，請編輯應用程式用戶端，然後在身分驗證流程下選擇使用使用者名稱和密碼登入：ALLOW_USER_PASSWORD_AUTH。若要允許伺服器端應用程式的純密碼登入，請編輯應用程式用戶端，然後選擇使用伺服器端管理登入資料登入：ALLOW_ADMIN_USER_PASSWORD_AUTH。

若要使用使用者名稱和密碼啟用以[選擇為基礎的身分驗證](#)，請將您的應用程式用戶端設定為允許。編輯您的應用程式用戶端，然後選擇選擇型登入：ALLOW_USER_AUTH。

Edit app client information [Info](#)

App clients create integration between your app and your user pool. App clients can use their own subset of authentication flows, token characteristics, and security from your user pool.

App client

Configure app clients. App clients are the user pool authentication resources attached to your app. Select an app client to configure the permitted authentication actions for an app.

App client name [Info](#)

Enter a friendly name for your app client.

my-test-app-client

App client names are limited to 128 characters or less. Names may only contain alphanumeric characters, spaces, and the following special characters: + , . @ -

Authentication flows [Info](#)

Choose authentication flows that your app will support. Refresh token authentication is always enabled. We have populated options based on your app type.

- ☒ **Choice-based sign-in: ALLOW_USER_AUTH**
Your user pool responds to sign-in requests with a list of available methods. Users can choose options like one-time passwords, biometric devices and security keys, and password-based sign-in with MFA.
- ☒ **Sign in with username and password: ALLOW_USER_PASSWORD_AUTH**
Users can sign in with a username and password. This method sends the username and password directly to your user pool.
- ☐ **Sign in with secure remote password (SRP): ALLOW_USER_SRP_AUTH**
Users can sign in with username and password. Your application uses SRP libraries in server-side or client-side sign-in operations to pass a password hash and verifier.
- ☒ **Sign in with server-side administrative credentials: ALLOW_ADMIN_USER_PASSWORD_AUTH**
Users can sign in with username and password in server-side authentication operations. This feature is not supported in HostedUI.
- ☐ **Sign in with custom authentication flows from Lambda triggers: ALLOW_CUSTOM_AUTH**
Users can sign in, optionally with username and password, and respond to custom challenges that you design in Lambda functions.
- ☒ **Get new user tokens from existing authenticated sessions: ALLOW_REFRESH_TOKEN_AUTH**
Your application can store a longer-lived refresh token that renews user sessions without additional user prompts.

若要驗證以選擇為基礎的身分驗證流程中是否提供密碼身分驗證，請導覽至登入功能表，並檢閱以選擇為基礎的登入選項下的 區段。如果在可用選項下顯示密碼，您可以使用純密碼身分驗證登入。密碼選項包含純和 SRP 使用者名稱密碼身分驗證變體。

Edit options for choice-based sign-in [Info](#)

With the USER_AUTH sign-in flow, users can choose their primary sign-in factor from a list of options like password, passwordless, and passkey. Choose the types of authentication that you want to allow for users' first authentication prompt.

Available choices [Info](#)

Choose the types of authentication that you want to allow users to choose in the choice-based flow.

Enabled options

Password

Additional choices [Info](#)

Configure the authentication factors that you want users to be able to choose in prompt-based authentication. Users must register any factors that they want to choose for sign-in.

- ☐ Passkey
- ☐ Email message one-time password
- ☐ SMS message one-time password

在 [CreateUserPoolClient](#) 或 [UpdateUserPoolClient](#) 請求中 `ExplicitAuthFlows`，使用您偏好的 `username-and-password` 驗證選項來設定。

```
"ExplicitAuthFlows": [
  "ALLOW_USER_PASSWORD_AUTH",
  "ALLOW_ADMIN_USER_PASSWORD_AUTH",
  "ALLOW_USER_AUTH"
]
```

在 [CreateUserPool](#) 或 [UpdateUserPool](#) 請求中，`Policies` 使用您要支援的以選擇為基礎的身分驗證流程來設定。中的 `PASSWORD` 值 `AllowedFirstAuthFactors` 包含純密碼和 SRP 身分驗證流程選項。

```
"Policies": {
  "SignInPolicy": {
    "AllowedFirstAuthFactors": [
      "PASSWORD",
      "EMAIL_OTP",
      "WEB_AUTHN"
    ]
  }
}
```

Choice-based sign-in with a password

若要使用使用者名稱密碼身分驗證將使用者登入應用程式，請設定 [AdminInitiateAuth](#) 或 [InitiateAuth](#) 請求的內文，如下所示。如果目前使用者符合使用者名稱密碼身分驗證的資格，則此登入請求會成功或繼續下一個挑戰。否則，它會回應可用主要要素驗證挑戰的清單。這組參數是登入所需的最低限度。還有其他參數可用。

```
{
  "AuthFlow": "USER_AUTH",
```

```
"AuthParameters": {
  "USERNAME" : "testuser",
  "PREFERRED_CHALLENGE" : "PASSWORD",
  "PASSWORD" : "[User's password]"
},
"ClientId": "lexample23456789"
}
```

您也可以省略 PREFERRED_CHALLENGE 值，並接收包含使用者合格登入因素清單的回應。

```
{
  "AuthFlow": "USER_AUTH",
  "AuthParameters": {
    "USERNAME" : "testuser"
  },
  "ClientId": "lexample23456789"
}
```

如果您未提交偏好的挑戰，或提交的使用者不符合他們偏好的挑戰資格，Amazon Cognito 會在 中傳回選項清單 AvailableChallenges。當 AvailableChallenges 包含 ChallengeName 的時 PASSWORD，您可以使用 [RespondToAuthChallenge](#) 或 [AdminRespondToAuthChallenge](#) 挑戰回應繼續驗證，格式如下。您必須傳遞 Session 參數，將挑戰回應與 API 回應關聯至您的初始登入請求。這組參數是登入所需的最低限度。還有其他參數可用。

```
{
  "ChallengeName": "PASSWORD",
  "ChallengeResponses": {
    "USERNAME" : "testuser",
    "PASSWORD" : "[User's Password]"
  },
  "ClientId": "lexample23456789",
  "Session": "[Session ID from the previous response]"
}
```

Amazon Cognito 會使用字符或其他必要的挑戰，例如多重要素驗證 (MFA) 來回應合格且成功的偏好挑戰請求和 PASSWORD 挑戰回應。

Client-based sign-in with a password

若要使用使用者名稱密碼身分驗證將使用者登入用戶端應用程式，請設定 [InitiateAuth](#) 請求的內文，如下所示。這組參數是登入所需的最低限度。還有其他參數可用。


```
{
  "AuthFlow": "USER_PASSWORD_AUTH",
  "AuthParameters": {
    "USERNAME" : "testuser",
    "PASSWORD" : "[User's password]"
  },
  "ClientId": "1example23456789"
}
```

若要使用使用者名稱-密碼身分驗證將使用者登入伺服器端應用程式，請設定 [AdminInitiateAuth](#) 請求的內文，如下所示。您的應用程式必須使用 AWS 登入資料簽署此請求。這組參數是登入所需的最低限度。還有其他參數可用。

```
{
  "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
  "AuthParameters": {
    "USERNAME" : "testuser",
    "PASSWORD" : "[User's password]"
  },
  "ClientId": "1example23456789"
}
```

Amazon Cognito 會使用字符或其他必要的挑戰來回應成功的請求，例如多重要素驗證 (MFA)。

使用持久性密碼和安全承載登入

使用者集區中的另一種使用者名稱密碼登入方法形式是使用安全遠端密碼 (SRP) 通訊協定。此選項會傳送使用者集區可驗證的密碼 — 密碼雜湊和 salt — 知識證明。在對 Amazon Cognito 的請求中沒有可讀取的秘密資訊，您的應用程式是唯一可處理使用者輸入密碼的實體。SRP 身分驗證涉及數學計算，最好是由您可以在 SDK 中匯入的現有元件完成。SRP 通常在用戶端應用程式中實作，例如行動應用程式。如需通訊協定的詳細資訊，請參閱 [Stanford SRP 首頁](#)。[Wikipedia](#) 也有資源和範例。[有多種公有程式庫](#)可用於執行身分驗證流程的 SRP 計算。

Amazon Cognito 身分驗證的 initiate-challenge-respond 序列會使用 SRP 驗證使用者及其密碼。您必須將使用者集區和應用程式用戶端設定為支援 SRP 身分驗證，然後在應用程式中實作登入請求和挑戰回應的邏輯。您的 SRP 程式庫可以產生隨機數字和計算值，向使用者集區證明您擁有使用者的密碼。您的應用程式會將這些計算值填入 Amazon Cognito 使用者集區 API 操作 AuthParameters 和 SDK 方法中的 JSON 格式和 ChallengeParameters 欄位，以進行身分驗證。

Activate SRP sign-in

若要使用使用者名稱和 SRP 啟用[用戶端型身分驗證](#)，請將您的應用程式用戶端設定為允許。在 Amazon Cognito 主控台中，導覽至使用者集區組態中應用程式下的應用程式用戶端選單。若要允許用戶端行動或原生應用程式的 SRP 登入，請編輯應用程式用戶端，然後在身分驗證流程下選擇使用安全遠端密碼 (SRP) 登入：ALLOW_USER_SRP_AUTH。

若要使用使用者名稱和 SRP 啟用以[選擇為基礎的身分驗證](#)，請編輯您的應用程式用戶端，然後選擇以選擇為基礎的登入：ALLOW_USER_AUTH。

Edit app client information [Info](#)

App clients create integration between your app and your user pool. App clients can use their own subset of authentication flows, token characteristics, and security from your user pool.

App client

Configure app clients. App clients are the user pool authentication resources attached to your app. Select an app client to configure the permitted authentication actions for an app.

App client name [Info](#)

Enter a friendly name for your app client.

my-test-app-client

App client names are limited to 128 characters or less. Names may only contain alphanumeric characters, spaces, and the following special characters: + = , . @ -

Authentication flows [Info](#)

Choose authentication flows that your app will support. Refresh token authentication is always enabled. We have populated options based on your app type.

☒ Choice-based sign-in: ALLOW_USER_AUTH

Your user pool responds to sign-in requests with a list of available methods. Users can choose options like one-time passwords, biometric devices and security keys, and password-based sign-in with MFA.

☐ Sign in with username and password: ALLOW_USER_PASSWORD_AUTH

Users can sign in with a username and password. This method sends the username and password directly to your user pool.

☒ Sign in with secure remote password (SRP): ALLOW_USER_SRP_AUTH

Users can sign in with username and password. Your application uses SRP libraries in server-side or client-side sign-in operations to pass a password hash and verifier.

☐ Sign in with server-side administrative credentials: ALLOW_ADMIN_USER_PASSWORD_AUTH

Users can sign in with username and password in server-side authentication operations. This feature is not supported in HostedUI.

☐ Sign in with custom authentication flows from Lambda triggers: ALLOW_CUSTOM_AUTH

Users can sign in, optionally with username and password, and respond to custom challenges that you design in Lambda functions.

☒ Get new user tokens from existing authenticated sessions: ALLOW_REFRESH_TOKEN_AUTH

Your application can store a longer-lived refresh token that renews user sessions without additional user prompts.

若要驗證 SRP 身分驗證是否在您的選擇型身分驗證流程中可用，請導覽至登入功能表，並檢閱選擇型登入選項下的 區段。如果在可用選項下顯示密碼，您可以使用 SRP 身分驗證登入。密碼選項包含純文字和 SRP 使用者名稱密碼驗證變體。

Edit options for choice-based sign-in [Info](#)

With the USER_AUTH sign-in flow, users can choose their primary sign-in factor from a list of options like password, passwordless, and passkey. Choose the types of authentication that you want to allow for users' first authentication prompt.

Available choices [Info](#)

Choose the types of authentication that you want to allow users to choose in the choice-based flow.

Enabled options

Password

Additional choices [Info](#)

Configure the authentication factors that you want users to be able to choose in prompt-based authentication. Users must register any factors that they want to choose for sign-in.

☐ Passkey

☐ Email message one-time password

☐ SMS message one-time password

在 [CreateUserPoolClient](#) 或 [UpdateUserPoolClient](#) 請求中 ExplicitAuthFlows，使用您偏好的 username-and-password 驗證選項來設定。

```
"ExplicitAuthFlows": [
```

```
"ALLOW_USER_SRP_AUTH",
"ALLOW_USER_AUTH"
]
```

在 [CreateUserPool](#) 或 [UpdateUserPool](#) 請求中，Policies 使用您要支援的以選擇為基礎的身分驗證流程來設定。中的 PASSWORD 值 AllowedFirstAuthFactors 包含純文字密碼和 SRP 身分驗證流程選項。

```
"Policies": {
  "SignInPolicy": {
    "AllowedFirstAuthFactors": [
      "PASSWORD",
      "EMAIL_OTP",
      "WEB_AUTHN"
    ]
  }
}
```

Choice-based sign-in with SRP

若要使用 SRP 以使用者名稱密碼身分驗證將使用者登入應用程式，請設定 [AdminInitiateAuth](#) 或 [InitiateAuth](#) 請求的內文，如下所示。如果目前使用者符合使用者名稱密碼身分驗證的資格，則此登入請求會成功或繼續下一個挑戰。否則，它會回應可用主要要素驗證挑戰的清單。這組參數是登入所需的最低限度。還有其他參數可用。

```
{
  "AuthFlow": "USER_AUTH",
  "AuthParameters": {
    "USERNAME" : "testuser",
    "PREFERRED_CHALLENGE" : "PASSWORD_SRP",
    "SRP_A" : "[g^a % N]"
  },
  "ClientId": "lexample23456789"
}
```

您也可以省略 PREFERRED_CHALLENGE 值，並接收包含使用者合格登入因素清單的回應。

```
{
  "AuthFlow": "USER_AUTH",
  "AuthParameters": {
    "USERNAME" : "testuser"
  },
}
```

```
"ClientId": "1example23456789"
}
```

如果您未提交偏好的挑戰，或提交的使用者不符合他們偏好的挑戰資格，Amazon Cognito 會在 中傳回選項清單 AvailableChallenges。當 AvailableChallenges 包含 ChallengeName 的時 PASSWORD_SRP，您可以使用 [RespondToAuthChallenge](#) 或 [AdminRespondToAuthChallenge](#) 挑戰回應繼續驗證，格式如下。您必須傳遞 Session 參數，將挑戰回應與 API 回應關聯至您的初始登入請求。這組參數是登入所需的最低限度。還有其他參數可用。

```
{
  "ChallengeName": "PASSWORD_SRP",
  "ChallengeResponses": {
    "USERNAME" : "testuser",
    "SRP_A" : "[g^a % N]"
  },
  "ClientId": "1example23456789",
  "Session": "[Session ID from the previous response]"
}
```

Amazon Cognito 會以挑戰來回應合格的偏好挑戰請求和 PASSWORD_SRP 挑戰回應 PASSWORD_VERIFIER。您的用戶端必須完成 SRP 計算，並回應 [RespondToAuthChallenge](#) 或 [AdminRespondToAuthChallenge](#) 請求中的挑戰。

```
{
  "ChallengeName": "PASSWORD_VERIFIER",
  "ChallengeResponses": {
    "PASSWORD_CLAIM_SIGNATURE" : "string",
    "PASSWORD_CLAIM_SECRET_BLOCK" : "string",
    "TIMESTAMP" : "string"
  },
  "ClientId": "1example23456789",
  "Session": "[Session ID from the previous response]"
}
```

在成功的 PASSWORD_VERIFIER 挑戰回應上，Amazon Cognito 會發出字符或其他必要的挑戰，例如多重要素驗證 (MFA)。

Client-based sign-in with SRP

SRP 身分驗證在用戶端身分驗證中比在伺服器端更為常見。不過，您可以將 SRP 身分驗證與 [InitiateAuth](#) 和 [AdminInitiateAuth](#) 搭配使用。若要將使用者登入應用程式，請設定

InitiateAuth或 AdminInitiateAuth請求的內文，如下所示。這組參數是登入所需的最低限度。還有其他參數可用。

用戶端SRP_A從產生器模數 N_g 產生，提高到秘密隨機整數 a 的力量。

```
{
  "AuthFlow": "USER_SRP_AUTH",
  "AuthParameters": {
    "USERNAME" : "testuser",
    "SRP_A" : "[g^a % N]"
  },
  "ClientId": "1example23456789"
}
```

Amazon Cognito 會以 PASSWORD_VERIFIER 挑戰回應。您的用戶端必須完成 SRP 計算，並回應 [RespondToAuthChallenge](#) 或 [AdminRespondToAuthChallenge](#) 請求中的挑戰。

```
{
  "ChallengeName": "PASSWORD_VERIFIER",
  "ChallengeResponses": {
    "PASSWORD_CLAIM_SIGNATURE" : "string",
    "PASSWORD_CLAIM_SECRET_BLOCK" : "string",
    "TIMESTAMP" : "string"
  },
  "ClientId": "1example23456789",
  "Session": "[Session ID from the previous response]"
}
```

在成功的PASSWORD_VERIFIER挑戰回應上，Amazon Cognito 會發出字符或其他必要的挑戰，例如多重要素驗證 (MFA)。

使用一次性密碼的無密碼登入

密碼可能會遺失或遭竊。您可能只想驗證您的使用者是否可以存取已驗證的電子郵件地址、電話號碼或驗證器應用程式。解決方案是無密碼登入。您的應用程式可以提示使用者輸入使用者名稱、電子郵件地址或電話號碼。然後，Amazon Cognito 會產生一次性密碼 (OTP)，這是他們必須確認的代碼。成功的程式碼會完成身分驗證。

無密碼身分驗證流程與使用者集區中所需的多重要素驗證 (MFA) 不相容。如果您的使用者集區中是選用的 MFA，則已啟用 MFA 的使用者無法使用無密碼第一個因素登入。在 MFA 選用使用者集區中沒有

MFA 偏好設定的使用者可以使用無密碼因素登入。如需詳細資訊，請參閱[使用者集區 MFA 的須知事項](#)。

當使用者在無密碼身分驗證中正確輸入在簡訊或電子郵件訊息中收到的代碼時，除了驗證使用者之外，您的使用者集區還會將使用者的未驗證電子郵件地址或電話號碼屬性標記為已驗證。無論您將使用者集區設定為[自動驗證](#)電子郵件地址或電話號碼CONFIRMED，使用者狀態也會從 UNCONFIRMED 變更為。

使用無密碼登入的新選項

當您在使用者集區中啟用無密碼身分驗證時，它會變更某些使用者流程的運作方式。

1. 使用者可以在沒有密碼的情況下註冊，並在登入時選擇無密碼因素。您也可以建立沒有密碼的使用者做為管理員。
2. 您使用 [CSV 檔案匯入](#)的使用者可以使用無密碼因素立即登入。他們不需要在登入前設定密碼。
3. 沒有密碼的使用者可以在沒有 PreviousPassword 參數的情況下提交 [ChangePassword](#) API 請求。

使用 OTPs 自動登入

使用電子郵件或簡訊 OTPs 註冊和確認其使用者帳戶的使用者，可以使用與其確認訊息相符的無密碼因素自動登入。在受管登入 UI 中，確認其帳戶且有資格使用確認碼交付方法進行 OTP 登入的使用者，會在提供確認碼後自動繼續第一次登入。在具有 AWS SDK 的自訂應用程式中，將下列參數傳遞至 [InitiateAuth](#) 或 [AdminInitiateAuth](#) 操作。

- [ConfirmSignUp](#) API 回應中的 Session 參數做為Session請求參數。
- 的 [AuthFlow](#)USER_AUTH。

您可以傳遞 [PREFERRED_CHALLENGE](#) EMAIL_OTP或 SMS_OTP，但這不是必要的。Session 參數提供身分驗證證明，當您傳遞有效的工作階段程式碼AuthParameters時，Amazon Cognito 會忽略。

登入操作會傳回回應，指出成功的身分驗證 [AuthenticationResult](#)，如果下列條件為 true，則不會有額外的挑戰。

- Session 代碼有效且未過期。
- 使用者符合 OTP 身分驗證方法的資格。

Activate passwordless sign-in

主控台

若要啟用無密碼登入，請將您的使用者集區設定為允許使用一或多個無密碼類型的主要登入，然後將您的應用程式用戶端設定為允許USER_AUTH流程。在 Amazon Cognito 主控台中，導覽至使用者集區組態中身分驗證下的登入選單。編輯以選擇為基礎的登入選項，然後選擇電子郵件訊息一次性密碼或簡訊一次性密碼。您可以啟用這兩個選項。儲存您的變更。

導覽至應用程式用戶端選單，然後選擇應用程式用戶端或建立新的用戶端。選取編輯，然後選擇在登入時選取身分驗證類型：ALLOW_USER_AUTH。

API/SDK

在使用者集區 API 中，SignInPolicy在 [CreateUserPool](#) 或 [UpdateUserPool](#) 請求中使用適當的無密碼選項進行設定。

```
"SignInPolicy": {
  "AllowedFirstAuthFactors": [
    "EMAIL_OTP",
    "SMS_OTP"
  ]
}
```

ExplicitAuthFlows 使用 [CreateUserPoolClient](#) 或 [UpdateUserPoolClient](#) 請求中的必要選項來設定您的應用程式用戶端。

```
"ExplicitAuthFlows": [
  "ALLOW_USER_AUTH"
]
```

Sign in with passwordless

無密碼登入沒有您可以在 [InitiateAuth](#) AuthFlow 和 [AdminInitiateAuth](#) 中指定的[用戶端型](#)登入。OTP 身分驗證僅適用於以 [選擇為基礎的](#) AuthFlow USER_AUTH，您可以在其中請求偏好的登入選項，或從使用者的 [AvailableChallenges](#) 選擇無密碼選項。若要將使用者登入應用程式，請設定 InitiateAuth或 AdminInitiateAuth請求的內文，如下所示。這組參數是登入所需的最低限度。還有其他參數可用。

在此範例中，我們不知道使用者想要以哪種方式登入。如果我們新增PREFERRED_CHALLENGE參數，而且使用者可以使用偏好的挑戰，Amazon Cognito 會回應該挑戰。

```
{
  "AuthFlow": "USER_AUTH",
  "AuthParameters": {
    "USERNAME" : "testuser"
  },
  "ClientId": "1example23456789"
}
```

在此範例中，您可以改為將 "PREFERRED_CHALLENGE": "EMAIL_OTP" 或 "PREFERRED_CHALLENGE": "SMS_OTP" 新增至 AuthParameters。如果使用者符合該偏好方法的資格，您的使用者集區會立即傳送代碼到使用者的電子郵件地址或電話號碼，並傳回 "ChallengeName": "EMAIL_OTP" 或 "ChallengeName": "SMS_OTP"。

如果您未指定偏好的挑戰，Amazon Cognito 會以 AvailableChallenges 參數回應。

```
{
  "AvailableChallenges": [
    "EMAIL_OTP",
    "SMS_OTP",
    "PASSWORD"
  ],
  "Session": "[Session ID]"
}
```

此使用者有資格使用電子郵件訊息 OTP、簡訊 OTP 和 username-password 進行無密碼登入。您的應用程式可以提示使用者進行選擇，或根據內部邏輯進行選擇。接著會繼續進行 [RespondToAuthChallenge](#) 或 [AdminRespondToAuthChallenge](#) 請求，以選取挑戰。假設使用者想要使用電子郵件訊息 OTP 完成無密碼身分驗證。

```
{
  "ChallengeName": "SELECT_CHALLENGE",
  "ChallengeResponses": {
    "USERNAME" : "testuser",
    "ANSWER" : "EMAIL_OTP"
  },
  "ClientId": "1example23456789",
  "Session": "[Session ID from the previous response]"
}
```

Amazon Cognito 會回應 EMAIL_OTP 挑戰，並將代碼傳送到您使用者已驗證的電子郵件地址。然後，您的應用程式必須再次回應此挑戰。

如果您以 EMAIL_OTP 身分請求，這也會是下一個挑戰回應 PREFERRED_CHALLENGE。

```
{
  "ChallengeName": "EMAIL_OTP",
  "ChallengeResponses": {
    "USERNAME" : "testuser",
    "EMAIL_OTP_CODE" : "123456"
  },
  "ClientId": "1example23456789",
  "Session": "[Session ID from the previous response]"
}
```

使用 WebAuthn 通行金鑰進行無密碼登入

通行密鑰是安全的，並對使用者施加相對較低的工作量。Passkey 登入使用身分驗證器，這是使用者可以進行身分驗證的外部裝置。一般密碼會讓使用者暴露在網路釣魚、密碼猜測和憑證遭竊等漏洞中。使用通行密鑰，您的應用程式可以受益於行動電話和其他連接至或內建於資訊系統的裝置上的進階安全措施。常見的通行金鑰登入工作流程會從呼叫您密碼或登入資料管理員的裝置呼叫開始，例如 iOS Keychain 或 Google Chrome 密碼管理員。裝置登入資料管理員會提示他們選取通行金鑰，並使用現有的登入資料或裝置解鎖機制進行授權。現代手機具有臉部掃描器、指紋掃描器、解鎖模式和其他機制，有些可同時滿足您已知的事項，以及具有強式身分驗證原則的事項。如果使用生物識別特徵進行通行金鑰身分驗證，通行金鑰代表您的身分。

您可能想要使用指紋、臉部或安全金鑰身分驗證取代密碼。這是 passkey 或 WebAuthn 身分驗證。應用程式開發人員通常會允許使用者在第一次使用密碼登入後註冊生物識別裝置。透過 Amazon Cognito 使用者集區，您的應用程式可以為使用者設定此登入選項。通行金鑰身分驗證不符合多重要素身分驗證 (MFA) 的資格。

無密碼身分驗證流程與使用者集區中所需的多重要素驗證 (MFA) 不相容。如果您的使用者集區中是選用的 MFA，則已啟用 MFA 的使用者無法使用無密碼第一個因素登入。在 MFA 選用使用者集區中沒有 MFA 偏好設定的使用者可以使用無密碼因素登入。如需詳細資訊，請參閱[使用者集區 MFA 的須知事項](#)。

什麼是通行密鑰？

Passkeys 無需記住複雜的密碼或輸入 OTPs 即可簡化使用者體驗。Passkey 是以[全球資訊網協會 \(W3C\)](#) 和 FIDO (快速線上身分) Alliance 所草擬的 WebAuthn 和 CTAP2 標準為基礎。瀏覽器和平台實作這些標準，為 Web 或行動應用程式提供 APIs 以啟動通行金鑰註冊或身分驗證程序，也為使用者提供 UI 以選擇通行金鑰驗證器並與之互動。

當使用者向網站或應用程式註冊驗證器時，驗證器會建立公有/私有金鑰對。WebAuthn 瀏覽器和平台會將公有金鑰提交至網站或應用程式後端的應用程式。驗證器會保留有關使用者和應用程式的私有金鑰、金鑰 IDs 和中繼資料。當使用者想要在已註冊應用程式中使用其已註冊的驗證器進行驗證時，應用程式會產生隨機挑戰。此挑戰的回應是該應用程式和使用者的驗證器私有金鑰所產生的挑戰數位簽章，以及相關的中繼資料。瀏覽器或應用程式平台會收到數位簽章，並將其傳遞給應用程式後端。然後，應用程式會使用存放的公有金鑰驗證簽章。

Note

您的應用程式不會收到使用者提供給身分驗證器的任何身分驗證秘密，也不會收到私有金鑰的相關資訊。

以下是目前市場上驗證者的一些範例和功能。驗證者可能符合任何或所有這些類別。

- 有些驗證者會在授予存取權之前，使用 PIN 碼、臉部或指紋的生物識別輸入或密碼等因素執行使用者驗證，確保只有合法使用者才能授權動作。其他驗證器沒有任何使用者驗證功能，有些可在應用程式不需要時略過使用者驗證。
- 有些驗證器，例如 YubiKey 硬體字符，是可攜式的。它們會透過 USB、藍牙或 NFC 連線與裝置通訊。有些驗證者是本機身分，並繫結至平台，例如 PC 上的 Windows Hello 或 iPhone 上的 Face ID。裝置繫結身分驗證器可以由使用者攜帶，如果夠小，就像行動裝置一樣。有時候，使用者可以透過無線通訊，將硬體驗證器與許多不同的平台連線。例如，桌面瀏覽器中的使用者在掃描 QR 碼時，可以使用其智慧型手機做為通行金鑰驗證器。
- 有些平台繫結的通行金鑰會同步至雲端，以便從多個位置使用。例如，iPhones 上的 Face ID 通行金鑰會將通行金鑰中繼資料與其 iCloud Keychain 中的使用者 Apple 帳戶同步。這些通行金鑰會在 Apple 裝置之間授予無縫身分驗證，而不是要求使用者獨立註冊每個裝置。軟體型驗證器應用程式，例如 1Password、Dashlane 和 Bitwarden 同步通行金鑰，涵蓋使用者已安裝應用程式的所有平台。

在 WebAuthn 術語中，網站和應用程式依賴各方。每個通行金鑰都與特定的依賴方 ID 相關聯，這是一個統一的識別符，代表接受通行金鑰身分驗證的網站或應用程式。開發人員必須仔細選取其依賴方 ID，才能擁有正確的身分驗證範圍。典型的依賴方 ID 是 Web 伺服器的根網域名稱。具有此依賴方 ID 規格的通行金鑰可以驗證該網域和子網域。當使用者想要存取的網站 URL 與依賴方 ID 不相符時，瀏覽器和平台會拒絕通行金鑰身分驗證。同樣地，對於行動應用程式，只有在應用程式在依存方 ID 指示的路徑上提供的 `.well-known` 關聯檔案中存在應用程式路徑時，才能使用通行金鑰。

通行金鑰是可探索的。瀏覽器或平台可以自動識別和使用它們，而無需使用者輸入使用者名稱。當使用者造訪支援通行金鑰身分驗證的網站或應用程式時，可以從瀏覽器或平台已知的通行金鑰清單中選擇，也可以掃描 QR 碼。

Amazon Cognito 如何實作通行金鑰身分驗證？

通行金鑰是一種選擇加入功能，除了 Lite 之外，所有[功能計劃](#)皆可使用。它僅適用於以[選擇為基礎的身分驗證流程](#)。透過[受管登入](#)，Amazon Cognito 會處理通行金鑰身分驗證的邏輯。您也可以[在 SDKs 中使用 AWS Amazon Cognito 使用者集區 API](#)，在您的應用程式後端執行通行金鑰身分驗證。

Amazon Cognito 會辨識使用兩種非對稱密碼編譯演算法之一建立的密碼金鑰：ES256(-7) 和 RS256(-257)。大多數驗證程式都支援這兩種演算法。根據預設，使用者可以設定任何類型的驗證器，例如硬體字符、手機和軟體驗證器應用程式。Amazon Cognito 目前不支援[認證](#)強制執行。

在您的使用者集區中，您可以將使用者驗證設定為偏好或必要。在未提供值的 API 請求中，此設定預設為偏好，在 Amazon Cognito 主控台中預設會選取偏好。當您將使用者驗證設定為偏好時，使用者可以設定沒有使用者驗證功能的驗證器，而且註冊和身分驗證操作可以在沒有使用者驗證的情況下成功。若要在通行金鑰註冊和身分驗證中強制使用者驗證，請將此設定變更為必要。

您在通行金鑰組態中設定的依賴方 (RP) ID 是重要的決策。當您未另外指定，且您的[網域品牌版本](#)為受管登入時，您的使用者集區預設為預期[自訂網域名稱](#)為 RP ID。如果您沒有自訂網域且未另外指定，您的使用者集區會預設為[字首網域](#)的 RP ID。您也可以將 RP ID 設定為不在公有字尾清單 (PSL) 中的任何網域名稱。您的 RP ID 項目適用於受管登入和 SDK 身分驗證中的通行金鑰註冊和身分驗證。Passkey 只能在行動應用程式中使用 Amazon Cognito 找到 .well-known 關聯檔案，並將您的 RP ID 做為網域。最佳實務是在網站或應用程式可公開使用之前，先判斷並設定依賴方 ID 的值。如果您變更 RP ID，您的使用者必須使用新的 RP ID 再次註冊。

每個使用者可以註冊最多 20 個通行密鑰。他們至少登入您的使用者集區一次後，才能註冊通行金鑰。受管登入會從通行金鑰註冊中移除大量工作。當您為使用者集區和應用程式用戶端啟用通行金鑰驗證時，具有受管登入網域的使用者集區會提醒最終使用者在註冊新使用者帳戶後註冊通行金鑰。您也可以隨時叫用使用者的瀏覽器，將使用者導向受管登入頁面以進行通行金鑰註冊。使用者必須提供使用者名稱，Amazon Cognito 才能啟動通行金鑰身分驗證。受管登入會自動處理。登入頁面會提示輸入使用者名稱，驗證使用者至少已註冊一個通行金鑰，然後提示輸入通行金鑰登入。同樣地，以 SDK 為基礎的應用程式必須提示使用者名稱，並在身分驗證請求中提供使用者名稱。

當您使用通行金鑰設定使用者集區身分驗證，且您具有自訂網域和字首網域時，RP ID 會預設為自訂網域的完整網域名稱 (FQDN)。若要在 Amazon Cognito 主控台中將字首網域設定為 RP ID，請刪除您的自訂網域，或將字首網域的 FQDN 輸入為第三方網域。

Activate passkey sign-in

主控台

若要使用通行金鑰啟用登入，請設定您的使用者集區以允許使用一或多個無密碼類型的主要登入，然後設定您的應用程式用戶端以允許USER_AUTH流程。在 Amazon Cognito 主控台中，導覽至使用者集區組態中身分驗證下的登入選單。編輯以選擇為基礎的登入選項，並將 Passkey 新增至可用選項清單。

導覽至身分驗證方法選單並編輯通行密鑰。

- 使用者驗證是使用者集區是否需要通行金鑰裝置，以執行目前使用者是否獲得通行金鑰授權的其他檢查的設定。若要鼓勵使用者設定具有使用者驗證但不需要的裝置，請選取偏好。若要僅支援具有使用者驗證的裝置，請選取需要。如需詳細資訊，請參閱 <https://w3.org/usage/proofkey/> 的[使用者驗證](#)。
- 用於依賴方 ID 的網域是您的應用程式將傳入使用者通行密鑰註冊請求的識別符。它設定與使用者通行密鑰發行者的信任關係目標。您的依賴方 ID 可以是：使用者集區的網域，如果

Cognito 網域

使用者集區的 Amazon Cognito [字首網域](#)。

自訂網域

使用者集區的[自訂網域](#)。

第三方網域

不使用使用者集區受管登入頁面的應用程式網域。此設定通常與沒有[網域](#)的使用者集區相關聯，並使用 SDK AWS 和後端中的使用者集區 API 執行身分驗證。

導覽至應用程式用戶端選單，然後選擇應用程式用戶端或建立新的用戶端。選取編輯，然後在身分驗證流程下，選擇在登入時選取身分驗證類型：ALLOW_USER_AUTH。

API/SDK

在使用者集區 API 中，SignInPolicy使用 [CreateUserPool](#) 或 [UpdateUserPool](#) 請求中的適當通行金鑰選項來設定。通行金鑰身分驗證WEB_AUTHN的選項必須至少隨附一個其他選項。通行金鑰註冊需要現有的身分驗證工作階段。

```
"SignInPolicy": {
  "AllowedFirstAuthFactors": [
    "PASSWORD",
    "WEB_AUTHN"
```

```
]
}
```

在 [SetUserPoolMfaConfig](#) 請求的 `WebAuthnConfiguration` 參數中設定您的使用者驗證偏好設定和 RP ID。 `RelyingPartyId` 是通行金鑰身分驗證結果的預期目標，可以是您的使用者集區字首或自訂網域，或是您選擇的網域。

```
"WebAuthnConfiguration": {
  "RelyingPartyId": "example.auth.us-east-1.amazoncognito.com",
  "UserVerification": "preferred"
}
```

`ExplicitAuthFlows` 使用 [CreateUserPoolClient](#) 或 [UpdateUserPoolClient](#) 請求中的必要選項來設定您的應用程式用戶端。

```
"ExplicitAuthFlows": [
  "ALLOW_USER_AUTH"
]
```

Register a passkey (managed login)

受管登入會處理使用者註冊通行密鑰。當您的使用者集區中啟用通行金鑰身分驗證時，Amazon Cognito 會在註冊新使用者帳戶時提示使用者設定通行金鑰。

Amazon Cognito 不會在使用者已註冊且未設定通行金鑰時，或如果您將其帳戶建立為管理員時，提示他們設定通行金鑰。處於此狀態的使用者必須使用密碼或無密碼 OTP 等另一個因素登入，才能註冊通行金鑰。

註冊通行密鑰

1. 將使用者導向您的 [登入頁面](#)。

```
https://auth.example.com/oauth2/authorize/?
client_id=1example23456789&response_type=code&scope=email+openid
+phone&redirect_uri=https%3A%2F%2Fwww.example.com
```

2. 處理來自使用者的身分驗證結果。在此範例中，Amazon Cognito `www.example.com` 會使用您的應用程式交換字符的授權碼將其重新導向至。
3. 將使用者導向至 `register-passkey` 頁面。使用者將擁有維護其登入工作階段的瀏覽器 Cookie。通行金鑰 URL 接受 `client_id` 和 `redirect_uri` 參數。Amazon Cognito 僅允許已驗證的

使用者存取此頁面。使用密碼、電子郵件 OTP 或簡訊 OTP 登入您的使用者，然後調用符合下列模式的 URL。

您也可以將其他[授權端點](#)參數新增至此請求，例如 `response_type` 和 `scope`。

```
https://auth.example.com/passkeys/add?  
client_id=1example23456789&redirect_uri=https%3A%2F%2Fwww.example.com
```

Register a passkey (SDK)

您可以使用 [PublicKeyCreationOptions](#) 物件中的中繼資料註冊通行金鑰登入資料。您可以使用登入使用者的登入資料產生此物件，並在 API 請求中將其呈現給其通行金鑰發行者。發行者將傳回 [RegistrationResponseJSON](#) 物件，以確認通行金鑰註冊。

若要開始通行金鑰註冊程序，請使用現有的登入選項登入使用者。使用目前使用者的存取權杖授權 [StartWebAuthnRegistration](#) API 請求。以下是範例 `GetWebAuthnRegistrationOptions` 請求的內文。

```
{  
  "AccessToken": "eyJra456defEXAMPLE"  
}
```

來自使用者集區的回應包含 `PublicKeyCreationOptions` 物件。在 API 請求中將此物件呈現給使用者的發行者。它提供公有金鑰和依賴方 ID 等資訊。發行者將回應 `RegistrationResponseJSON` 物件。

在 [CompleteWebAuthnRegistration](#) API 請求中呈現註冊回應，再次使用使用者的存取權杖進行授權。當您的使用者集區以空白內文回應 HTTP 200 回應時，即會註冊使用者的通行金鑰。

Sign in with a passkey

無密碼登入沒有您可以在 [InitiateAuth](#) AuthFlow 和 [AdminInitiateAuth](#) 中指定的。反之，您必須宣告 AuthFlow 的 `USER_AUTH` 並請求登入選項，或從使用者集區的回應中選擇無密碼選項。若要將使用者登入應用程式，請設定 `InitiateAuth` 或 `AdminInitiateAuth` 請求的內文，如下所示。這組參數是登入所需的最低限度。還有其他參數可用。

在此範例中，我們知道使用者想要使用通行金鑰登入，並新增 `PREFERRED_CHALLENGE` 參數。

```
{  
  "AuthFlow": "USER_AUTH",  
  "AuthParameters": {
```

```
"USERNAME" : "testuser",
"PREFERRED_CHALLENGE" : "WEB_AUTHN"
},
"ClientId": "lexample23456789"
}
```

Amazon Cognito 會以 WEB_AUTHN 挑戰回應。您的應用程式必須回應此挑戰。使用使用者的通行金鑰提供者啟動登入請求。它會傳回 [AuthenticationResponseJSON](#) 物件。

```
{
  "ChallengeName": "WEB_AUTHN",
  "ChallengeResponses": {
    "USERNAME" : "testuser",
    "CREDENTIAL" : "{AuthenticationResponseJSON}"
  },
  "ClientId": "lexample23456789",
  "Session": "[Session ID from the previous response]"
}
```

登入後的 MFA

您可以設定使用使用者名稱密碼流程完成登入的使用者，以提示使用來自電子郵件訊息、簡訊或程式碼產生應用程式的一次性密碼進行其他驗證。MFA 不同於無密碼登入，這是具有一次性密碼的第一個身分驗證因素，或不包含 MFA 的 WebAuthn 通行金鑰。使用者集區中的 MFA 是一種挑戰回應模型，使用者會先示範他們知道密碼，然後示範他們有權存取其已註冊的第二因素裝置。

實作資源

- [將 MFA 新增到使用者集區](#)

重新整理權杖

當您想要讓使用者在不重新輸入登入資料的情況下保持登入狀態時，重新整理字符是您的應用程式必須保留使用者工作階段的工具。應用程式可以將重新整理字符呈現到您的使用者集區，並交換為新的 ID 和存取字符。透過權杖重新整理，您可以確保登入的使用者仍處於作用中狀態、取得更新的屬性資訊，以及更新存取控制權利，而無需使用者介入。

實作資源

- [重新整理權杖](#)

自訂身分驗證

您可能想要為未列於此處的使用者設定身分驗證方法。您可以使用 Lambda 觸發程序進行自訂身分驗證。在一系列 Lambda 函數中，Amazon Cognito 發出挑戰、詢問使用者必須回答的問題、檢查答案的準確性，然後判斷是否應該發出另一個挑戰。問題和答案可能包括安全問題、對 CAPTCHA 服務的請求、對外部 MFA 服務 API 的請求，或這些項目的序列。

實作資源

- [自訂身分驗證挑戰 Lambda 觸發程序](#)

自訂身分驗證流程

Amazon Cognito 使用者集區也能讓您使用自訂身分驗證流程，協助您使用 AWS Lambda 觸發程序來建立以挑戰/回應為基礎的身分驗證模型。

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](#)。

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](#)。

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 觸發程序匯入使用者](#)。

API 和 SDK 身分驗證的授權模型

當您使用使用者集區身分驗證開始開發應用程式時，您必須決定適合您應用程式類型的 API 授權模型。授權模型是一種系統，用於提供授權，以使用 Amazon Cognito 使用者集區 API 和 SDK 整合中的身分驗證元件提出請求。Amazon Cognito 有三種授權模式：IAM 授權、公有和字符授權。

使用 IAM 授權請求時，授權來自請求 Authorization 標頭中一組 AWS IAM 登入資料的簽章。對於伺服器端應用程式，此做法會使用 IAM 授權保護身分驗證操作。對於公有（未驗證）身分驗證請求，不需要授權。這適用於分發給使用者的用戶端應用程式。透過權杖授權的操作，通常與公有操作結合實作，授權來自工作階段權杖或包含在請求 Authorization 標頭中的存取權杖。Amazon Cognito 身分驗證通常會要求您依序實作兩個或多個 API 操作，而您使用的 API 操作取決於應用程式的特性。應用程式分發給使用者的公有用戶端會使用公有操作，其中登入請求不需要授權。字符授權的操作會在公有應用程式中繼續使用者工作階段。伺服器端用戶端，其中應用程式邏輯託管在遠端系統上，使用登入請求的 IAM 授權來保護身分驗證操作。後面的 API 操作對及其對應的 SDK 方法會映射到可用的授權模型。

每個公有身分驗證操作都有某種形式的伺服器端對等項目，例如 [UpdateUserAttributes](#) 和 [AdminUpdateUserAttributes](#)。雖然用戶端操作是由使用者啟動且需要確認，但伺服器端操作假設變更是由使用者集區管理員遞交，且變更會立即生效。在此範例中，Amazon Cognito 會傳送包含確認碼的訊息給使用者，而使用者的存取權杖會授權提交程式碼的 [VerifyUserAttribute](#) 請求。伺服器端應用程式可以立即設定任何屬性的值，但是在用於登入時，變更電子郵件地址和電話號碼的值 [會受到特殊考量](#)。

若要比較 API 身分驗證並查看 API 操作及其授權模型的完整清單，請參閱 [了解 API、OIDC 和受管登入頁面身分驗證](#)。

Client-side (public) authentication

以下是用戶端應用程式中典型的一系列請求

1. 公有 [InitiateAuth](#) 操作會提交主要登入資料，例如使用者名稱和密碼。
2. 權杖授權的 [RespondToAuthChallenge](#) 操作會從InitiateAuth回應和挑戰的答案提交工作階段權杖，例如 MFA。工作階段字串授權表示請求屬於not-yet-complete身分驗證週期的一部分。
3. 權杖授權的 [ConfirmDevice](#) 操作會提交存取權杖，並執行將記住的裝置新增至使用者設定檔的寫入操作。存取字串授權表示在使用者完成身分驗證後，適用於使用者自助式操作的請求。

如需詳細資訊，請參閱[用戶端身分驗證選項](#)及[了解 API、OIDC 和受管登入頁面身分驗證](#)。

Server-side authentication

以下是伺服器端操作的典型請求序列。每個請求都有[AWS 簽章第 4 版](#)授權標頭，該標頭使用發行給應用程式伺服器的 IAM 機器登入資料簽署。

1. [AdminInitiateAuth](#) 操作會提交主要登入資料，例如使用者名稱和密碼。
2. [AdminRespondToAuthChallenge](#) 操作會將答案提交至挑戰，例如 MFA。
3. [AdminUpdateDeviceStatus](#) 操作會將來自AdminInitiateAuth[回應](#)的裝置金鑰設定為記住。

如需詳細資訊，請參閱[伺服器端身分驗證選項](#)及[了解 API、OIDC 和受管登入頁面身分驗證](#)。

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。

主題

- [伺服器端身分驗證選項](#)
- [用戶端身分驗證選項](#)
- [了解 API、OIDC 和受管登入頁面身分驗證](#)
- [依授權模型分組的 API 操作清單](#)

伺服器端身分驗證選項

Web 應用程式和其他伺服器端應用程式會在用戶端載入遠端顯示應用程式的遠端伺服器上實作身分驗證，例如瀏覽器或 SSH 工作階段。伺服器端應用程式通常具有下列特性。

- 它們內建於安裝在伺服器上的應用程式，使用 Java、Ruby 或 Node.js 等語言。
- 他們連線到可能有用戶端秘密的使用者集區[應用程式用戶端](#)，稱為機密用戶端。
- 他們可以存取 AWS 登入資料。
- 他們叫用[受管登入](#)進行身分驗證，或在使用者集區 API 中使用 IAM 授權的操作搭配 AWS SDK。
- 它們為內部客戶提供服務，並可能為公眾客戶提供服務。

使用使用者集區 API 的伺服器端操作可以使用密碼、一次性密碼或通行密鑰作為主要登入因素。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發出 [AdminInitiateAuth](#) API 請求。此操作需要具有包含 `cognito-idp:AdminInitiateAuth` 和許可的 AWS 登入資料 `cognito-idp:AdminRespondToAuthChallenge`。操作會傳回所需的挑戰或身分驗證結果。
- 當應用程式收到挑戰時，會發出 [AdminRespondToAuthChallenge](#) API 請求。AdminRespondToAuthChallenge API 操作也需要 AWS 登入資料。

如需使用 AWS 登入資料簽署 Amazon Cognito API 請求的詳細資訊，請參閱《AWS 一般參考》中的 [Signature 第 4 版簽署程序](#)。

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

由於後端管理員實作使用管理員身分驗證流程，因此流程不支援記住的裝置。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。

用戶端身分驗證選項

行動應用程式和其他用戶端應用程式類型安裝在使用者的裝置上，並在本機執行身分驗證和使用者的邏輯。它們通常具有下列特性。

- 它們內建於 React 原生、Flutter 和 Swift 等語言，並部署到使用者裝置。
- 它們連接到沒有用戶端秘密的使用者集區[應用程式用戶端](#)，稱為公有用戶端。

- 他們無法存取會授權 IAM 授權 API 請求的 AWS 登入資料。
- 他們叫用[受管登入](#)進行身分驗證，或在使用者集區 API 中使用公有和字符授權的操作搭配 AWS SDK。
- 它們為公眾客戶提供服務，並允許任何人註冊和登入。

使用使用者集區 API 的用戶端操作可以使用密碼、一次性密碼或通行金鑰作為主要登入因素。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)。

了解 API、OIDC 和受管登入頁面身分驗證

Amazon Cognito 使用者集區是多種身分驗證技術的組合。他們倚賴外部身分提供者 (IdPs)。它們是使用 OpenID Connect (OIDC) SDKs 實作身分驗證之應用程式的 IdPs。它們以類似 OIDC 身分驗證的 JSON Web 權杖 (JWTs) 發行者身分驗證，但在屬於 AWS SDKs 的 API 方法中提供身分驗證。它們也可以是應用程式的安全進入點。

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. 您的受管登入頁面和傳統託管 UI 包括[受管登入使用者互動端點](#)，以及處理 IdP 和依賴方角色的[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。若要快速開始使用 Amazon Cognito 使用者集區的身分驗證和授權功能，包括註冊、登入、密碼管理和多重要素驗證 (MFA) 的頁面，請使用受管登入的內建使用者介面。

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 `/.well-known/jwks.json`。您的應用程式會在使用 OIDC 或 OAuth 2.0 開發人員程式庫驗證權杖或擷取使用者設定檔資料時，使用這些端點。
2. [Amazon Cognito 使用者集區 API](#) 是一組工具，可供您的 Web 或行動應用程式在您自己的自訂前端收集登入資訊後驗證使用者。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。存取權杖自訂會為您的 AWS 帳單增加成本。

- c. 重新整理字符，可授權新 ID 和存取字符的請求，並重新整理使用者身分和存取控制屬性。

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。

1. 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，以及設定託管受管登入頁面的使用者集區網域。
2. 它會為本機和連結的使用者執行註冊、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。如需包含受管登入之使用者集區端點的詳細資訊，請參閱 [使用者集區端點和受管登入參考](#)。

您的聯合身分使用者可以從受管登入開始並選取其 IdP，或者您可以略過受管登入，並將使用者直接傳送到您的 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。

具有受管登入頁面的範例案例

1. 您的使用者選擇您在應用中建立的「建立帳戶」按鈕。
2. 受管登入會為您的使用者提供您已註冊開發人員登入資料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 身分驗證頁面。您的使用者登入並選擇授權 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. 應用程式中的 OIDC 程式庫會與 交換授權碼，[權杖端點](#)並存放使用者集區發出的 ID 字符、存取字符和重新整理字符。您的應用程式使用 OIDC 程式庫來管理使用者的字符，並維護該使用者的持久性工作階段。

使用者集區 API 和受管登入頁面支援各種案例，如本指南所述。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。

依授權模型分組的 API 操作清單

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 [IAM 授權的管理操作](#) 和 [IAM 授權的使用者操作](#) 的 IAM 政策中指派許可。

IAM 授權的管理操作

IAM 授權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組態，就像您在 [AWS Management Console](#) 中所做的一樣。

例如，若要修改 [UpdateUserPool](#) API 請求中的使用者集區，您必須提供 AWS 登入資料和 IAM 許可可以更新資源。

若要在 AWS Command Line Interface (AWS CLI) 或 AWS SDK 中授權這些請求，請使用將 IAM 登入資料新增至請求的環境變數或用戶端組態來設定您的環境。如需詳細資訊，請參閱《[AWS 使用您的 AWS 登入資料存取 AWS 一般參考](#)》。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 登入資料來授權或簽署這些請求。如需詳細資訊，請參閱[簽署 AWS API 請求](#)。

IAM 授權的管理操作

[AddCustomAttributes](#)

[CreateGroup](#)

[CreateIdentityProvider](#)

[CreateResourceServer](#)

[CreateUserImportJob](#)

[CreateUserPool](#)

[CreateUserPoolClient](#)

[CreateUserPoolDomain](#)

IAM 授權的管理操作

[DeleteGroup](#)

[DeleteIdentityProvider](#)

[DeleteResourceServer](#)

[DeleteUserPool](#)

[DeleteUserPoolClient](#)

[DeleteUserPoolDomain](#)

[DescribeIdentityProvider](#)

[DescribeResourceServer](#)

[DescribeRiskConfiguration](#)

[DescribeUserImportJob](#)

[DescribeUserPool](#)

[DescribeUserPoolClient](#)

[DescribeUserPoolDomain](#)

[GetCSVHeader](#)

[GetGroup](#)

[GetIdentityProviderByIdentifier](#)

[GetSigningCertificate](#)

[GetUICustomization](#)

[GetUserPoolMfaConfig](#)

[ListGroups](#)

[ListIdentityProviders](#)

IAM 授權的管理操作

[ListResourceServers](#)

[ListTagsForResource](#)

[ListUserImportJobs](#)

[ListUserPoolClients](#)

[ListUserPools](#)

[ListUsers](#)

[ListUsersInGroup](#)

[SetRiskConfiguration](#)

[SetUICustomization](#)

[SetUserPoolMfaConfig](#)

[StartUserImportJob](#)

[StopUserImportJob](#)

[TagResource](#)

[UntagResource](#)

[UpdateGroup](#)

[UpdateIdentityProvider](#)

[UpdateResourceServer](#)

[UpdateUserPool](#)

[UpdateUserPoolClient](#)

[UpdateUserPoolDomain](#)

IAM 授權的使用者操作

IAM 授權的使用者操作註冊、登入、管理登入資料、修改和檢視您的使用者。

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，請使用將 IAM 登入資料新增至請求的環境變數或用戶端組態來設定伺服器端應用程式環境。如需詳細資訊，請參閱《》中的[AWS 使用您的 AWS 登入資料存取 AWS](#) 一般參考。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 登入資料來授權或簽署這些請求。如需詳細資訊，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。如需詳細資訊，請參閱[運算私密雜湊值](#)。

IAM 授權的使用者操作

[AdminAddUserToGroup](#)

[AdminConfirmSignUp](#)

[AdminCreateUser](#)

[AdminDeleteUser](#)

[AdminDeleteUserAttributes](#)

[AdminDisableProviderForUser](#)

[AdminDisableUser](#)

[AdminEnableUser](#)

[AdminForgetDevice](#)

[AdminGetDevice](#)

[AdminGetUser](#)

IAM 授權的使用者操作

[AdminInitiateAuth](#)

[AdminLinkProviderForUser](#)

[AdminListDevices](#)

[AdminListGroupsWithUser](#)

[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，您不需要設定任何登入資料。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，您不需要設定任何登入資料。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	AccessTok en	Session (工 作階段)
<u>RespondToAuthChallenge</u>		✓
<u>ChangePassword</u>	✓	
<u>GetUser</u>	✓	
<u>StartWebAuthnRegistration</u>	✓	
<u>CompleteWebAuthnRegistration</u>	✓	
<u>DeleteWebAuthnCredentia</u>	✓	
<u>ListWebAuthnCredentia</u>	✓	
<u>UpdateUserAttributes</u>	✓	
<u>DeleteUserAttributes</u>	✓	
<u>DeleteUser</u>	✓	
<u>ConfirmDevice</u>	✓	
<u>ForgetDevice</u>	✓	
<u>GetDevice</u>	✓	
<u>ListDevices</u>	✓	
<u>UpdateDeviceStatus</u>	✓	
<u>GetUserAttributeVerificationCode</u>	✓	

權杖授權的使用者操作	AccessTok en	Session (工 作階段)
VerifyUserAttribute	✓	
SetUserSettings	✓	
SetUserMFAPreferen ce	✓	
GlobalSignOut	✓	
UpdateAuthEventFee dback		✓
AssociateSoftwareT oken	✓	✓
VerifySoftwareToken	✓	✓
RevokeToken¹		
GetTokensFromRefre shToken¹		

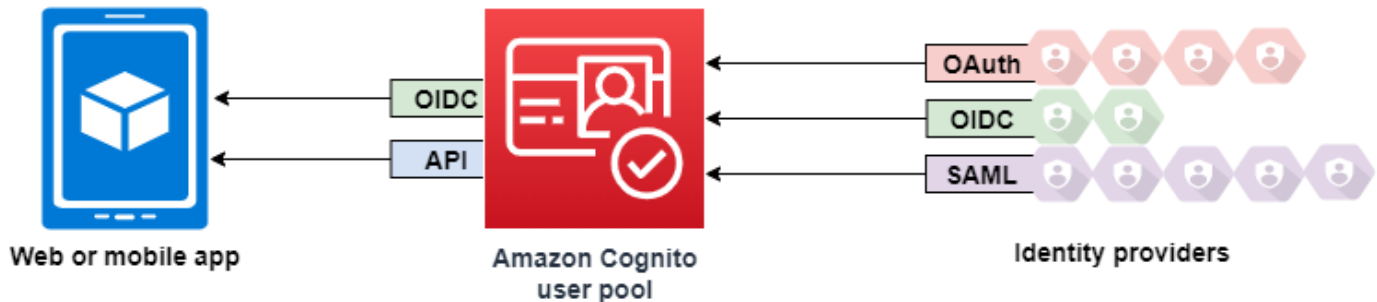
¹ RevokeToken並將GetTokensFromRefreshToken重新整理字符作為授權參數。重新整理權杖用作授權權杖，並作為目標資源。

使用第三方身分提供者登入使用者集區

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提供者 (IdP) 聯合。使用者集區可管理處理字符的開銷，包括透過 Facebook、Google、Amazon 及 Apple 社交登入傳回的字符，以及從 OpenID Connect (OIDC) 及 SAML IdP 傳回的字符。Amazon Cognito 使用內建的託管 Web UI，可以為所有 IdP 其已驗證身分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 與社交、SAML 或 OpenID (OIDC) IdP 連線時，您的使用者集區會充當多個服務供應商與您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)。您的 IdP 會將 OIDC ID 權杖或 SAML 聲明傳遞至 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 [了解使用者集區 JSON Web 字符 JWTs](#)。

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[受管登入](#)的頁面。如需詳細資訊，請參閱[使用您自己的網域進行受管登入](#)。
- 您無法使用如 [InitiateAuth](#) 或 [AdminInitiateAuth](#) 等 API 操作登入聯合身分使用者。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，它會無提示地重新導向到您的 IdP，繞過受管登入。否則，它會重新導向至受管登入 [登入端點](#)。
- 當受管登入將工作階段重新導向至聯合身分 IdP 時，Amazon Cognito 會在請求 Amazon/Cognito 中包含 `user-agent` 標頭。
- 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 會為您新增至使用者集區的每個 OIDC、SAML 和社交 IdP 建立[使用者群組](#)。群組的名稱格式為 `[user pool ID]_[IdP name]`，例如 `us-east-1_EXAMPLE_MYSSO` 或 `us-east-1_EXAMPLE_Google`。每個唯一自動產生的 IdP 使用者設定檔都會自動新增至此群組。[連結的使用者](#)不會自動新增至此群組，但您可以個別程序將其設定檔新增至群組。
- 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 時，您的變更最多可能需要一分鐘才會出現在受管登入中。
- Amazon Cognito 在本身和您的 IdP 之間最多支援 20 個 HTTP 重新導向。
- 當您的使用者使用受管登入登入時，其瀏覽器會存放加密的登入工作階段 Cookie，該 Cookie 會記錄其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，受管登入會重複使用任何未過期的現有工作階段，而使用者會驗證而不再提供登入資料。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 IdP 指派給任何應用程式用戶端，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](#)
- [搭配使用者集區使用社交身分提供者](#)
- [搭配使用者集區使用 SAML 身分提供者](#)
- [搭配使用者集區使用 OIDC 身分提供者](#)
- [將 IdP 屬性映射至設定檔和字符](#)
- [將聯合身分使用者連結至現有的使用者描述檔](#)

為您的使用者集區配置身分提供者

透過使用者集區，您可以透過各種外部身分提供者 (IdPs) 實作登入。本指南的本節說明如何在 Amazon Cognito 主控台中，使用使用者集區設定這些身分提供者。或者，您可以使用使用者集區 API 和 AWS SDK，以程式設計方式新增使用者集區身分提供者。如需詳細資訊，請參閱 [CreateIdentityProvider](#)。

支援的身分提供者選項包括 Facebook、Google 和 Amazon 等社交提供者，以及 OpenID Connect (OIDC) 和 SAML 2.0 提供者。開始使用之前，請先為 IdP 設定管理登入資料。對於每種類型的提供者，您將需要註冊應用程式、取得必要的登入資料，然後在使用者集區中設定提供者詳細資訊。然後，您的使用者可以從連線的身分提供者註冊並登入您的應用程式及其現有帳戶。

驗證下的社交和外部提供者功能表會新增和更新使用者集區 IdPs。如需詳細資訊，請參閱[使用第三方身分提供者登入使用者集區](#)。

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](#)。如果出現提示，請輸入您的 AWS 登入資料。
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. 選擇社交和外部供應商選單。
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 [搭配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。

- 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。

- 選擇儲存變更。
- 從應用程式用戶端功能表中，從清單中選擇應用程式用戶端，然後選取編輯。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。
- 選擇儲存變更。

如需社交 IdPs 的詳細資訊，請參閱 [搭配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

- 前往 [Amazon Cognito 主控台](#)。如果出現提示，請輸入您的 AWS 登入資料。
- 從導覽選單中，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](#)。

4. 選擇社交和外部供應商選單，然後選取新增身分提供者。
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. 選擇建立。
16. 從應用程式用戶端功能表中，從清單中選擇應用程式用戶端，然後選取編輯。若要將新的 SAML 身分提供者新增至應用程式用戶端，請導覽至登入頁面索引標籤，然後選取在受管登入頁面上編輯組態。
17. 選擇儲存變更。

如需 OIDC IdPs 的詳細資訊，請參閱 [搭配使用者集區使用 OIDC 身分提供者](#)。

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。若要了解如何取得適用於第三方 SAML IdP 的中繼資料文件，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](#)。如果出現提示，請輸入您的 AWS 登入資料。
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. 選擇社交和外部提供者選單，然後選取新增身分提供者。
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 SAML 2.0 身分提供者，將登出回應傳送至 Amazon Cognito 在您設定受管登入時建立的<https://mydomain.auth.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 用來驗證已簽署登出請求之公有金鑰的憑證，請執行下列動作：
 - 前往使用者集區的社交和外部供應商選單。
 - 選取您的 SAML 供應商、
 - 選擇檢視簽署憑證。

如需 SAML IdP 的詳細資訊，請參閱[搭配使用者集區使用 SAML 身分提供者](#)。

搭配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。您必須啟用受管登入，才能與支援的社交身分提供者整合。當 Amazon Cognito 建置您的受管登入頁面時，它會建立 Amazon Cognito 和 OIDC 和社交 IdPs 用來交換資訊的 OAuth 2.0 端點。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

- [設定社交 IdP 開發人員帳戶和應用程式](#)
- [使用社交 IdP 設定您的使用者集區](#)
- [測試社交 IdP 組態](#)

設定社交 IdP 開發人員帳戶和應用程式

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。

Facebook

如需有關中繼開發人員帳戶和身分驗證組態的最新資訊，請參閱[中繼應用程式開發](#)。

如何向 Facebook/Meta 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)。

- 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)。
- 在左側導覽列中，選擇 Settings (設定) 然後 Basic (基本)。
- 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。
- 從頁面底部選擇 + Add Platform (+ 新增平台)。
- 選擇 Website (網站)。
- 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.auth.us-east-1.amazoncognito.com/login?
response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com
```

- 選擇儲存變更。
- 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.auth.us-east-1.amazoncognito.com
```

- 選擇儲存變更。
- 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)。
- 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.auth.us-east-1.amazoncognito.com/oauth2/idpresponse
```

- 選擇儲存變更。

Login with Amazon

如需 Login with Amazon 開發人員帳戶和身分驗證組態的最新資訊，請參閱 [Login with Amazon 文件](#)。

如何使用 Login with Amazon 註冊應用程式

- [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.auth.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.auth.us-east-1.amazoncognito.com/oauth2/idpresponse
```

11. 選擇儲存。

Google

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](#)。

如何向 Google 註冊應用程式

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 Authorized JavaScript origins (授權的 JavaScript 來源) 下方，選擇 ADD URI (新增 URI)。輸入您的使用者集區網域。

```
https://mydomain.auth.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.auth.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。

Sign in with Apple

如需設定「使用 Apple 登入」的最新資訊，請參閱《Apple 開發人員文件》中的[設定您的環境以使用 Apple 登入](#)。

如何使用 Sign in with Apple (SIWA) 註冊應用程式

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 [使用社交 IdP 設定您的使用者集區](#) 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 [使用社交 IdP 設定您的使用者集區](#) 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.auth.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.auth.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 [使用社交 IdP 設定您的使用者集區](#) 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。

使用社交 IdP 設定您的使用者集區

使用 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 登入資料。
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. 選擇社交和外部提供者選單，然後選取新增身分提供者。
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. 選擇建立。
 10. 從應用程式用戶端功能表中，從清單中選擇應用程式用戶端，然後選取編輯。若要將新的社交身分提供者新增至應用程式用戶端，請導覽至登入頁面索引標籤，然後選取受管登入頁面上的編輯組態。
 11. 選擇儲存變更。

測試社交 IdP 組態

在您的應用程式中，您必須在使用者的用戶端中叫用瀏覽器，以便他們可以使用其社交供應商登入。在您完成上述章節中的設定程序後，向您的社交供應商測試登入。下列範例 URL 會使用字首網域載入使用者集區的登入頁面。

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

此連結是當您前往應用程式用戶端功能表、選取應用程式用戶端、導覽至登入頁面索引標籤，然後選取檢視登入頁面時，Amazon Cognito 會引導您前往的頁面。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](#)。如需應用程式用戶端的詳細資訊，包括用戶端 IDs 和回呼 URLs，請參閱[使用應用程式用戶端的應用程式特定設定](#)。

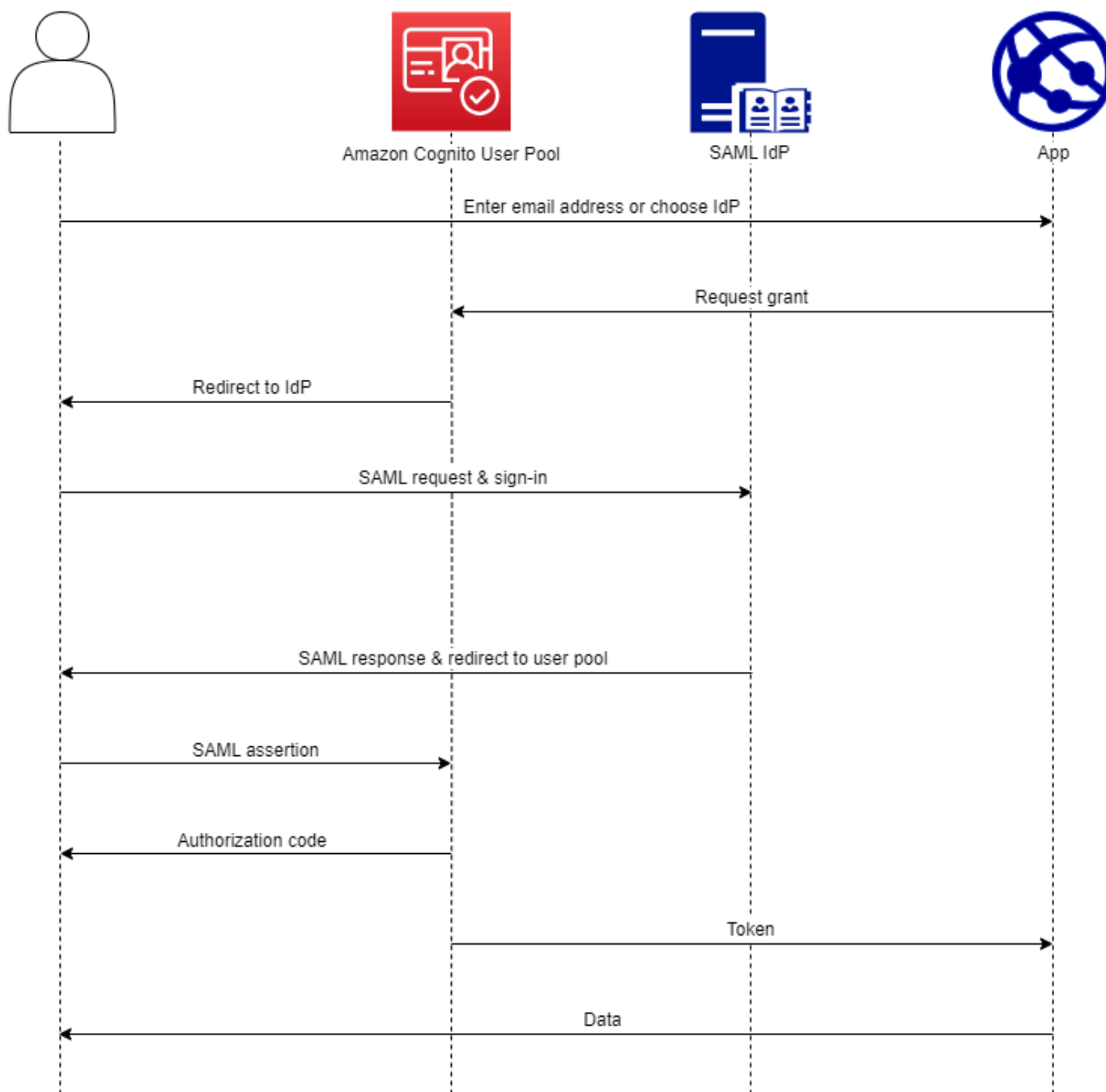
下列範例連結[授權端點](#)會使用 identity_provider 查詢參數，從設定無提示重新導向至社交供應商。此 URL 會略過具有受管登入的互動式使用者集區登入，並直接前往 IdP 登入頁面。

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/
authorize?identity_provider=Facebook|Google|LoginWithAmazon|
SignInWithApple&response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com
```

搭配使用者集區使用 SAML 身分提供者

您可以選擇讓您的 Web 和行動應用程式使用者透過 SAML 身分提供者 (IdP) 登入，例如 [Microsoft Active Directory Federation Services \(ADFS\)](#) 或 [Shibboleth](#)。您必須選擇支援 [SAML 2.0 標準](#) 的 SAML IdP。

透過 受管登入，Amazon Cognito 會驗證本機和第三方 IdP 使用者，並發出 JSON Web 字串 JWTs)。使用 Amazon Cognito 發行的字串，您可以將多個身分來源合併到所有應用程式的通用 OpenID Connect (OIDC) 標準。Amazon Cognito 可以將第三方供應商的 SAML 聲明處理為該 SSO 標準。您可以在 中 AWS Management Console、透過 AWS CLI 或使用 Amazon Cognito 使用者集區 API 建立和管理 SAML IdP。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 身分提供者](#)。



 Note

透過第三方 IdP 登入的聯合身分是 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

您必須設定 SAML IdP 以接受請求，並將回應傳送到您的使用者集區。SAML IdP 的文件將包含有關如何將使用者集區新增為 SAML 2.0 IdP 依賴方或應用程式的資訊。以下文件提供您必須為 SP 實體 ID 和聲明消費者服務 (ACS) URL 提供的值。

使用者集區 SAML 值快速參考

SP 實體 ID

```
urn:amazon:cognito:sp:us-east-1_EXAMPLE
```

ACS URL

```
https://Your user pool domain/saml2/idpresponse
```

您必須設定使用者集區以支援您的身分提供者。新增外部 SAML IdP 的高階步驟如下所示。

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 身分提供者](#)。
3. 將 IdP 指派給您的應用程式用戶端。請參閱 [使用應用程式用戶端的應用程式特定設定](#)。

主題

- [Amazon Cognito 使用者集區的 SAML IdP 相關須知](#)
- [SAML 使用者名稱區分大小寫](#)
- [設定您的第三方 SAML 身分提供者](#)

- [在使用者集區中新增和管理 SAML 身分提供者](#)
- [Amazon Cognito 使用者集區中的 SAML 工作階段啟動](#)
- [使用單一登入來登入 SAML 使用者](#)
- [SAML 簽署和加密](#)
- [SAML 身分提供者名稱和識別符](#)

Amazon Cognito 使用者集區的 SAML IdP 相關須知

實作 SAML 2.0 IdP 隨附一些要求和限制。當您實作 IdP 時，請參閱本節。您還可以找到有助於在 SAML 與使用者集區聯合期間疑難排解錯誤的資訊。

Amazon Cognito 會為您處理 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 SP 啟動和 IdP 啟動的單一登入 (SSO)，如 [SAML V2.0 技術概觀](#) 第 5.1.2 和 5.1.4 節所述。

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 供應商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。

維護轉送狀態參數

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 字符是 Amazon Cognito 維護之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](#)。

識別 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.auth.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](#)。

沒有重播的聲明

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 ID 是 SP 實體 ID

您必須在服務供應商 (SP) 中提供您的 IdP 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。

映射所有必要屬性

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 [將 IdP 屬性映射至設定檔和字符](#)。

聲明格式具有特定需求

您的 SAML IdP 必須在 SAML 聲明中包含下列宣告。

- NameID 宣告。Amazon Cognito 會將 SAML 聲明與目的地使用者建立關聯NameID。如果NameID變更，Amazon Cognito 會將聲明視為新使用者的聲明。您在 IdP 組態NameID中設定為的屬性必須具有持久性值。若要將 SAML 使用者指派給使用者集區中的一致使用者描述檔，請從具有未變更值的 屬性指派您的NameID宣告。

```
<saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:persistent">
```

```
carlos
</saml2:NameID>
```

IdP NameID宣告Format中的 `urn:oasis:names:tc:SAML:1.1:nameid-format:persistent` 表示您的 IdP 正在傳遞不變的值。Amazon Cognito 不需要此格式宣告，`urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified` 如果您的 IdP 未指定 NameID 宣告的格式，則會指派 的格式。此行為符合 [SAML 2.0 規格的第 2.2.2 節](#) 複雜類型 NameIDType。

- 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>
```

- 對於 SP 起始的單一登入，具有原始 SAML 請求 ID InResponseTo 值的 Response 元素。

```
<saml2p:Response Destination="https://mydomain.auth.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 聲明不得包含 InResponseTo 值。

- 具有使用者集區 `saml2/idpresponse` 端點 Recipient 值的 SubjectConfirmationData 元素，對於 SP 起始的 SAML，則為符合原始 SAML 請求 ID InResponseTo 的值。

```
<saml2:SubjectConfirmationData InResponseTo="_dd0a3436-bc64-4679-a0c2-
cb4454f04184" NotOnOrAfter="Date-time stamp" Recipient="https://mydomain.auth.us-
east-1.amazoncognito.com/saml2/idpresponse"/>
```

SP 啟動的登入請求

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 請求包含使用者集區的相關資訊，包括您的 ACS 端點。您可以選擇性地將密碼編譯簽章套用至這些請求。

簽署請求並加密回應

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非對稱金鑰對和簽署憑證。您設定為支援加密 SAML 回應的每個外部 SAML IdP 會導致 Amazon Cognito 為該提供者產生新的金鑰對和加密憑證。若要使用公有金鑰檢視和下載憑證，請在 Amazon Cognito 主控台的社交和外部供應商選單中選擇您的 IdP。

若要建立對來自使用者集區的 SAML 請求的信任，請將使用者集區 SAML 2.0 簽署憑證的副本提供給 IdP。如果您未將 IdP 設定為接受已簽署的請求，IdP 可能會忽略您的使用者集區簽署的 SAML 請求。

1. Amazon Cognito 會將數位簽章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) 請求，而且您可以將使用者集區設定為簽署任何 SAML 外部 IdP 的單一登入 (SSO) 請求。當您提供憑證的副本時，IdP 可以驗證使用者 SAML 請求的完整性。
2. 您的 SAML IdP 可以使用加密憑證加密 SAML 回應。當您使用 SAML 加密設定 IdP 時，您的 IdP 只能傳送加密的回應。

編碼非英數字元

Amazon Cognito 不接受 4 位元組 UTF-8 字元#，例如 # 或您的 IdP 傳遞為屬性值。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？](#)。

中繼資料端點必須位於 HTTP 或 HTTPS 的標準 TCP 連接埠

Amazon Cognito 僅接受標準 TCP 連接埠 80 上的 SAML 提供者中繼資料 URLs，HTTP 則為 443。作為安全最佳實務，在具有 `https://` 字首的 TLS 加密 URL 上託管 SAML 中繼資料。輸入格式 URLs `https://www.example.com/saml2/metadata.xml`。 `http://www.example.com/saml2/metadata.xml` Amazon Cognito 主控台僅接受以 `https://` 為首的 URLs。您也可以使用 [CreateIdentityProvider](#) 和 [UpdateIdentityProvider](#) 設定 IdP 中繼資料。

具有 IdP 起始 SAML 的應用程式用戶端只能使用 SAML 登入

當您啟用支援應用程式用戶端 IdP 起始登入的 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` 做為 HTTP POST 請求。使用者集區不接受具有 HTTP GET 繫結的登出回應。

中繼資料簽署憑證輪換

當您使用 URL 提供中繼資料時，Amazon Cognito 會快取 SAML 中繼資料長達六個小時。執行任何中繼資料簽署憑證輪換時，請將中繼資料來源設定為發佈原始憑證和新憑證至少六小時。當 Amazon Cognito 從中繼資料 URL 重新整理快取時，它會將每個憑證視為有效，而且您的 SAML IdP 可以使用新憑證開始簽署 SAML 聲明。在此期間過後，您可以從已發佈的中繼資料中移除原始憑證。

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 身分提供者 (IdP) 會在使用者的 SAML 聲明中將唯一的傳遞 `NameId` 給 Amazon Cognito。Amazon Cognito 會依 `NameId` 宣告識別 SAML 聯合身分使用者。無論您使用者集區的區分大小寫設定為何，Amazon Cognito 都會在傳遞其唯一且區分大小寫的 `NameId` 宣告時，從 SAML IdP 辨識傳回的聯合身分使用者。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

- [設定您的第三方 SAML 身分提供者](#)
- [在使用者集區中新增和管理 SAML 身分提供者](#)
- [Amazon Cognito 使用者集區中的 SAML 工作階段啟動](#)
- [使用單一登出來登出 SAML 使用者](#)
- [SAML 簽署和加密](#)
- [SAML 身分提供者名稱和識別符](#)

設定您的第三方 SAML 身分提供者

當您想要將 SAML 身分提供者 (IdP) 新增至使用者集區時，您必須在 IdP 的管理界面中進行一些組態更新。本節說明如何格式化您必須提供給 IdP 的值。您也可以了解如何擷取靜態或作用中 URL 中繼資料文件，以識別 IdP 及其對使用者集區的 SAML 宣告。

若要設定第三方 SAML 2.0 身分提供者 (IdP) 解決方案以使用 Amazon Cognito 使用者集區的聯合，您必須設定 SAML IdP 以重新導向至下列聲明消費者服務 (ACS)

URL : <https://mydomain.auth.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](#) 的使用者集區的網域選單中找到您的使用者集區網域路徑。

有些 SAML IdPs 要求您提供 urn，也稱為對象 URI 或 SP 實體 ID，格式為

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。

您還必須設定 SAML IdP，為使用者集區中指定為必要屬性的任何屬性提供值。一般而言，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 聲明中提供某種形式的 email 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使用 Amazon Cognito 使用者集區設定聯合的最佳位置。如需最新資訊，請直接參閱供應商的文件。

若要簽署 SAML 請求，您必須設定 IdP 以信任由使用者集區簽署憑證簽署的請求。若要接受加密的 SAML 回應，您必須設定 IdP 來加密對使用者集區的所有 SAML 回應。您的提供者將擁有設定這些功能的文件。如需 Microsoft 的範例，請參閱[設定 Microsoft Entra SAML 字符加密](#)。

Note

Amazon Cognito 只需要您的身分提供者中繼資料文件。您的提供者也可能提供 SAML 2.0 與 IAM 或 聯合的自訂組態資訊 AWS IAM Identity Center。若要了解如何設定 Amazon Cognito 整合，請尋找擷取中繼資料文件的一般指示，並管理使用者集區中的其餘組態。

解決方案	其他資訊
Microsoft Entra ID	聯合中繼資料
Okta	如何下載 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 身分提供者
Ping Identity (PingFederate)	從 PingFederate 匯出 SAML 中繼資料
JumpCloud	SAML 組態備註
SecureAuth	SAML 應用程式整合

在使用者集區中新增和管理 SAML 身分提供者

將身分提供者設定為使用 Amazon Cognito 之後，您可以將其新增至使用者集區和應用程式用戶端。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 供應商。

AWS Management Console

您可以使用 AWS Management Console 來建立和刪除 SAML 身分提供者 IdPs)。

建立 SAML IdP 之前，您必須擁有從第三方 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. 選擇社交和外部提供者選單，然後選取新增身分提供者。
5. 選擇 SAML IdP。
6. 輸入供應商名稱。您可以在 identity_provider 請求參數中將此易記的名稱傳遞給 [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 SAML 2.0 IdP，將登出回應傳送至設定受管登入時建立的 `https://mydomain.auth.us-east-1.amazonaws.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 預期有簽署的登出請求，您還必須將來自使用者集區的簽署憑證提供給 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。作為安全最佳實務，請選擇僅接受 SP 起始的 SAML 聲明。如果您已準備好讓環境安全地接受未經請求的 SAML 登入工作階段，請選擇接受 SP 起始和 IdP 起始的 SAML 聲明。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

如果您的提供者有公有端點，而不是上傳檔案，我們建議您輸入中繼資料文件 URL。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. 在 SAML 提供者和使用者集區之間映射屬性，將 SAML 提供者屬性映射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 使用簡單的名稱，例如 email，有些則使用如下的名稱。

```
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 #### XML>` 包含字元 的任何執行個體"，您必須新增 \ 做為逸出字元：`\`。

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 #### XML>` 包含字元 的任何執行個體"，您必須新增 \ 做為逸出字元：`\`。

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 IdP 的相關資訊

- 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 需要符合 SAML 回應中此 URN 的對象限制值。將 IdP 設定為針對 IdP-to-SP 回應訊息使用下列 POST 繫結端點。

```
https://mydomain.auth.us-east-1.amazoncognito.com/saml2/idpresponse
```

- 您的 SAML IdP 必須在 SAML 聲明中填入 NameID 和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 下載公有金鑰的副本，您的 IdP 可用來驗證 SAML 登出請求，請選擇使用者集區的社交和外部供應商選單，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

- 登入 [Amazon Cognito 主控台](#)。
- 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。
- 選擇社交和外部供應商選單。
- 在您要刪除的 SAML IdP 旁邊，選取選項按鈕。
- 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) 的單一登入 (SSO) 和 IdP 啟動的 SSO。作為最佳安全實務，請在使用者集區中實作 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 起始的流程進行身分驗證時，他們必須先向 Amazon Cognito 提出請求，並重新導向至 IdP 進行身分驗證。

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 啟動的身分驗證時，Amazon Cognito 無法驗證是否已請求其收到的 SAML 回應，因為 Amazon Cognito 不會使用 SAML 請求啟動身分驗證。在 SP 起始的 SSO 中，Amazon Cognito 會設定狀態參數，以針對原始請求驗證 SAML 回應。使用 SP 起始的登入，您也可以防止跨網站請求偽造 (CSRF)。

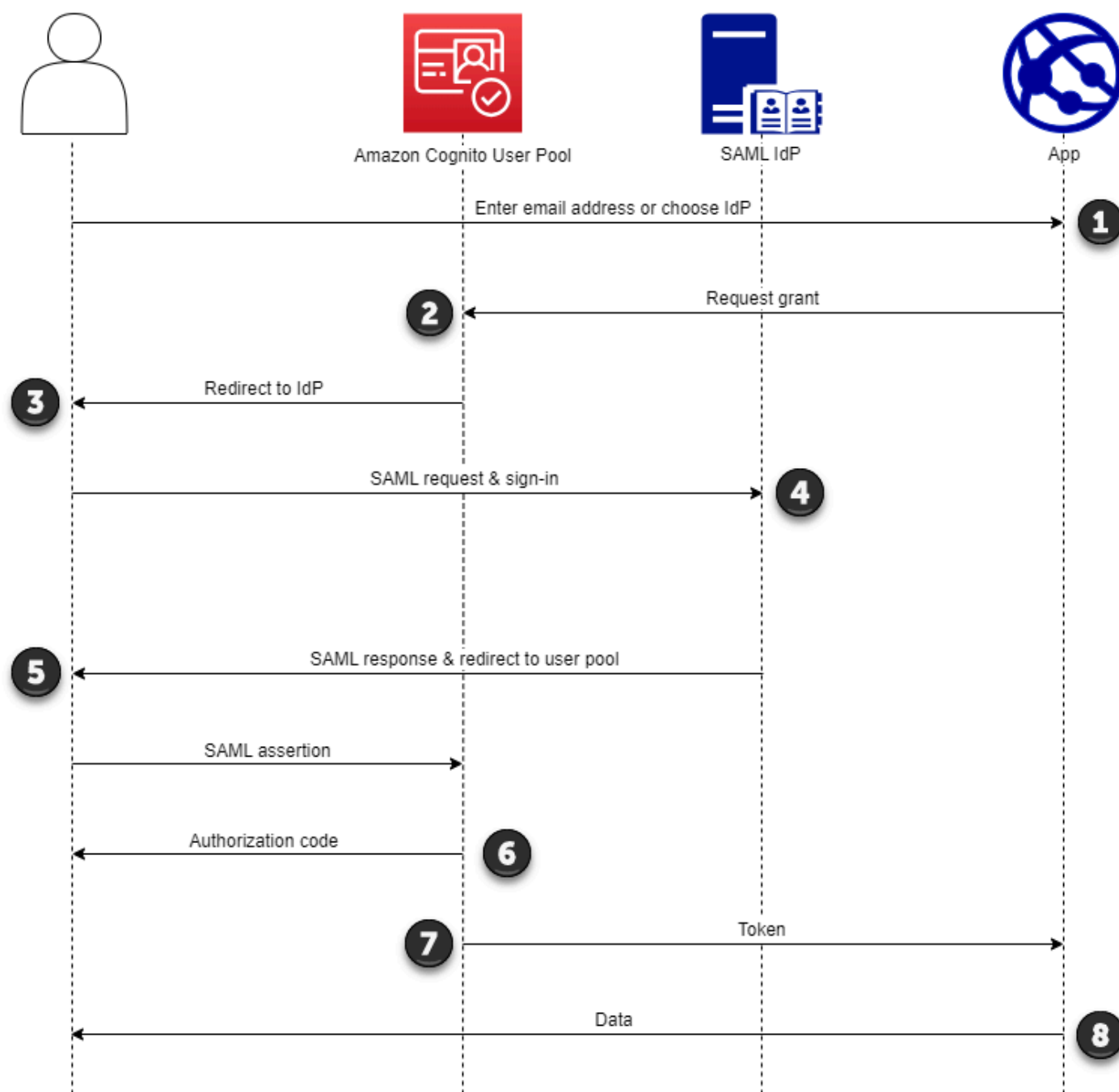
主題

- [實作 SP 初始化的 SAML 登入](#)
- [實作 IdP 起始的 SAML 登入](#)

實作 SP 初始化的 SAML 登入

最佳實務是實作使用者集區的service-provider-initiated(SP 起始) 登入。Amazon Cognito 會啟動使用者的工作階段，並將其重新導向至 IdP。使用此方法，您可以對誰提出登入請求擁有最大的控制權。您也可以特定情況下允許 IdP 起始的登入。

下列程序顯示使用者如何透過 SAML 供應商完成由 SP 起始的使用者集區登入。



1. 您的使用者會在登入頁面輸入其電子郵件地址。若要判斷使用者重新導向至其 IdP，您可以在自訂建置應用程式中收集其電子郵件地址，或在 Web 檢視中叫用受管登入。

您可以設定受管登入頁面以顯示 IdPs 清單，或提示輸入電子郵件地址，並將其與 SAML IdP 的識別符相符。若要提示輸入電子郵件地址，請在 Foundation 中編輯您的受管登入品牌樣式，找到身分驗證行為，然後在提供者顯示下，將顯示樣式設定為網域搜尋輸入。

2. 您的應用程式會叫用您的使用者集區重新導向端點，並請求具有對應至應用程式的用戶端 ID 和對應至使用者的 IdP ID 的工作階段。
3. Amazon Cognito 會使用 SAML 請求，[選擇性地在](#) AuthnRequest 元素中將您的使用者重新導向至 IdP。
4. IdP 會以互動方式驗證使用者，或在瀏覽器 Cookie 中使用記住的工作階段進行驗證。
5. IdP 會在您的使用者集區 SAML 回應端點的 POST 承載中使用[選擇性加密的](#) SAML 聲明來重新導向您的使用者。

 Note

Amazon Cognito 會在 5 分鐘內取消未收到回應的工作階段，並將使用者重新導向至受管登入。當您的使用者遇到此結果時，會收到 Something went wrong 錯誤訊息。

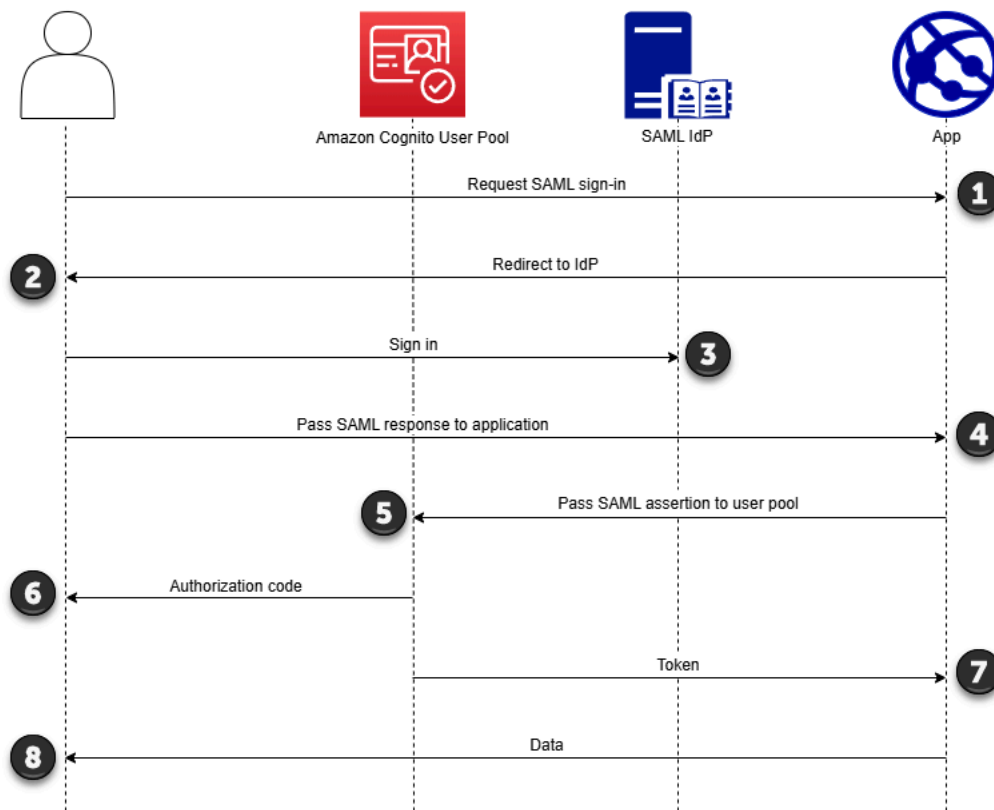
6. 在驗證 SAML 聲明並從回應中的宣告[映射使用者屬性](#)之後，Amazon Cognito 會在內部建立或更新使用者集區中的使用者設定檔。一般而言，您的使用者集區會將授權碼傳回至使用者的瀏覽器工作階段。
7. 您的使用者會向應用程式顯示其授權碼，以交換 JSON Web 字符 (JWTs 的程式碼)。
8. 您的應用程式接受並處理使用者的 ID 字符作為身分驗證，使用其存取字符產生對資源的授權請求，並存放其重新整理字符。

當使用者驗證並接收授權碼授予時，使用者集區會傳回 ID、存取和重新整理權杖。ID 字符是 OIDC 型身分管理的身分驗證物件。存取權杖是具有 [OAuth 2.0](#) 範圍的授權物件。重新整理權杖是在使用者目前的權杖過期時產生新 ID 和存取權杖的物件。您可以在使用者集區應用程式用戶端中設定使用者字符的持續時間。

您也可以選擇重新整理權杖的持續時間。使用者的重新整理字符過期後，他們必須再次登入。如果使用者透過 SAML IdP 進行身分驗證，則使用者的工作階段持續時間是透過其權杖過期設定，而不是其工作階段與其 IdP 過期。您的應用程式必須存放每個使用者的重新整理字符，並在過期時續約其工作階段。受管登入會在瀏覽器 Cookie 中維護使用者工作階段，有效期為 1 小時。

實作 IdP 起始的 SAML 登入

當您為 IdP 起始的 SAML 2.0 登入設定身分提供者時，您可以將 SAML 聲明呈現給使用者集區網域中的 `saml2/idpresponse` 端點，而不需要在啟動工作階段[授權端點](#)。具有此組態的使用者集區接受來自所請求應用程式用戶端所支援之使用者集區外部身分提供者的 IdP 起始 SAML 聲明。



1. 使用者使用您的應用程式請求 SAML 登入。
2. 您的應用程式會叫用瀏覽器，或將使用者重新導向至其 SAML 供應商的登入頁面。
3. IdP 會以互動方式驗證使用者，或在瀏覽器 Cookie 中使用記住的工作階段進行驗證。
4. IdP 會在使用者 POST 內文中使用 SAML 聲明或回應將使用者重新導向到您的應用程式。
5. 您的應用程式會將 SAML 聲明新增至使用者集區 `saml2/idpresponse` 端點請求的 POST 內文。
6. Amazon Cognito 會向您的使用者發出授權碼。
7. 您的使用者會向應用程式顯示其授權碼，以交換 JSON Web 字符 (JWTs) 的程式碼。
8. 您的應用程式接受並處理使用者的 ID 字符作為身分驗證，使用其存取字符產生對資源的授權請求，並存放其重新整理字符。

下列步驟說明使用 IdP 起始的 SAML 2.0 供應商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. 在使用者集區中建立 SAML 2.0 IdP。

3. 設定您的 IdP 以支援 IdP 啟動。IdP 起始的 SAML 引入了其他 SSO 供應商不受約束的安全考量。因此，您無法將非 SAML IdPs，包括使用者集區本身，新增至使用 SAML 供應商搭配 IdP 起始登入的任何應用程式用戶端。
4. 將 IdP 起始的 SAML 供應商與使用者集區中的應用程式用戶端建立關聯。
5. 引導您的使用者前往 SAML IdP 的登入頁面，並擷取 SAML 聲明。
6. 使用您的使用者 SAML 聲明將使用者導向您的使用者集區saml2/idpresponse端點。
7. 接收 JSON Web 字符 JWTs)。

若要在使用者集區中接受未經請求的 SAML 聲明，您必須考慮其對應用程式安全性的影響。當您接受 IdP 起始的請求時，可能會要求欺騙和 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 聲明。

此外，您的 SAML 聲明不得包含InResponseTo宣告，且必須在前 6 分鐘內發出。

您必須使用 IdP 起始的 SAML 將請求提交到您的 /saml2/idpresponse。對於 SP 起始和受管的登入授權請求，您必須提供參數，將請求的應用程式用戶端、範圍、重新導向 URI 和其他詳細資訊識別為HTTP GET請求中的查詢字串參數。不過，對於 IdP 起始的 SAML 聲明，請求的詳細資訊必須格式化為HTTP POST請求內文中的RelayState參數。請求內文也必須包含您的 SAML 聲明做為SAMLResponse參數。

以下是 IdP 起始 SAML 供應商的請求和回應範例。

```
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%3Dexample23456789%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

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 身分提供者。
2. 如有關聯，請將所有社交和 OIDC 身分提供者與您的應用程式用戶端取消關聯。
3. 導覽至使用者集區的社交和外部供應商選單。
4. 編輯或新增 SAML 供應商。
5. 在 IdP 起始的 SAML 登入下，選擇接受 SP 起始和 IdP 起始的 SAML 聲明。
6. 選擇儲存變更。

API/CLI

為 IdP 起始的 SAML 設定 IdP

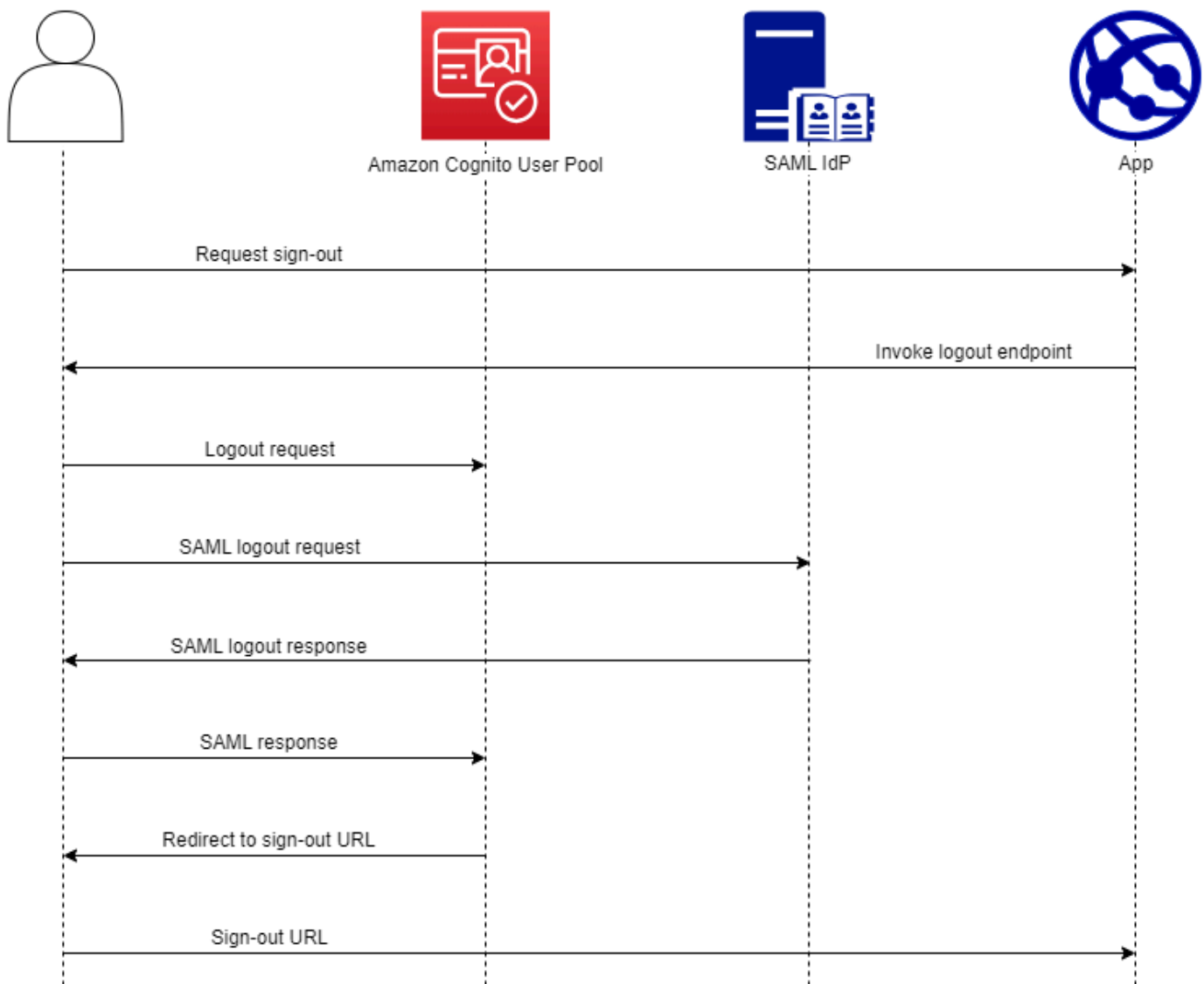
使用 [CreateIdentityProvider](#) 或 [UpdateIdentityProvider](#) API 請求中的 IDPInit 參數設定 IdP 起始的 SAML。以下是支援 ProviderDetails IdP 起始 SAML 的 IdP 範例。

```
"ProviderDetails": {
  "MetadataURL" : "https://myidp.example.com/saml/metadata",
  "IDPSignout" : "true",
  "RequestSigningAlgorithm" : "rsa-sha256",
  "EncryptedResponses" : "true",
  "IDPInit" : "true"
}
```

使用單一登出來登出 SAML 使用者

Amazon Cognito 支援 SAML 2.0 [單一登出](#) (SLO)。透過 SLO，您的應用程式可以在使用者從使用者集區登出時，從其 SAML 身分提供者 (IdPs) 登出使用者。如此一來，當使用者想要再次登入您的應用程式時，他們必須使用其 SAML IdP 進行驗證。否則，他們可能已備妥 IdP 或使用者集區瀏覽器 Cookie，這些 Cookie 會傳遞到您的應用程式，而不需要他們提供登入資料。

當您將 SAML IdP 設定為支援登出流程時，Amazon Cognito 會使用已簽署的 SAML 登出請求將使用者重新導向至 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您的使用者集區/logout端點時，Amazon Cognito 會使用下列請求將 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]
  
```

您的使用者接著LogoutResponse會從其 IdP 使用 返回您的saml2/logout端點。您的 IdP 必須在 HTTP POST請求LogoutResponse中傳送。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 供應商可能會傳送AuthnStatement一個LogoutResponse以上的。此類型的回應AuthnStatement中第一個 sessionIndex中的 必須符合最初驗證使用者的 SAML 回應sessionIndex中的。如果 sessionIndex 位於任何其他 中AuthnStatement，Amazon Cognito 將無法辨識工作階段，也不會將您的使用者登出。

AWS Management Console

設定 SAML 登出

1. 建立[使用者集區](#)、[應用程式用戶端](#)和 SAML IdP。
2. 當您建立或編輯 SAML 身分提供者時，請在身分提供者資訊下，勾選標題為新增登出流程的方塊。
3. 從使用者集區的社交和外部供應商選單中，選擇您的 IdP 並找到簽署憑證。
4. 選擇下載為 .crt。
5. 將 SAML 提供者設定為支援 SAML 單一登出和請求簽署，並上傳使用者集區簽署憑證。您的 IdP 必須重新導向至使用者集區網域/saml2/logout中的。

API/CLI

設定 SAML 登出

使用 [CreateIdentityProvider](#) 或 [UpdateIdentityProvider](#) API 請求的 IDPSignout 參數設定單一登出。以下是支援 SAML 單一登出ProviderDetails的 IdP 範例。

```
"ProviderDetails": {
  "MetadataURL" : "https://myidp.example.com/saml/metadata",
  "IDPSignout" : "true",
  "RequestSigningAlgorithm" : "rsa-sha256",
  "EncryptedResponses" : "true",
  "IDPInit" : "true"
}
```

SAML 簽署和加密

SAML 2.0 登入是以應用程式的使用者為基礎建置，作為其身分驗證流程中請求和回應的承載。您可能想要確保使用者不會在傳輸中讀取或修改這些 SAML 文件。若要達成此目的，請將 SAML 簽署和加密新增至使用者集區中的 SAML 身分提供者 IdPs)。透過 SAML 簽署，您的使用者集區會將簽章新增至 SAML 登入和登出請求。使用您的使用者集區公有金鑰，IdP 可以驗證其是否正在接收未修改的 SAML 請求。然後，當您的 IdP 回應並將 SAML 聲明傳遞給使用者的瀏覽器工作階段時，IdP 可以加密該回應，讓使用者無法檢查自己的屬性和權利。

透過 SAML 簽署和加密，使用者集區 SAML 操作期間的所有密碼編譯操作都必須使用 user-pool-provided 金鑰產生簽章和加密文字。Amazon Cognito 目前，您無法將使用者集區設定為使用外部金鑰簽署請求或接受加密聲明。

Note

您的使用者集區憑證有效期為 10 年。每年一次，Amazon Cognito 會為您的使用者集區產生新的簽署和加密憑證。當您請求簽署憑證時，Amazon Cognito 會傳回最新的憑證，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 聲明。您先前的憑證在整個期間內持續有效，而且憑證之間的公有金鑰不會變更。最佳實務是每年更新供應商組態中的憑證。

主題

- [從 IdP 接受加密的 SAML 回應](#)
- [簽署 SAML 請求](#)

從 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 會將公有/私有 RSA 金鑰對和憑證指派給您在使用者集區中設定的每個外部 SAML 提供者。當您為使用者集區 SAML 供應商啟用回應加密時，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 聲明之前，您的 SAML IdP 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其 SAML IdP。

2. 您的使用者集區會使用 SAML 登入請求，將使用者[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伴隨此請求使用簽章，以啟用 IdP 的完整性驗證。當您想要簽署 SAML 請求時，您必須設定 IdP 以接受您的使用者集區已使用簽署憑證中的公有金鑰簽署的請求。
3. SAML IdP 會在您的使用者中登入並產生 SAML 回應。IdP 會使用公有金鑰加密回應，並將您的使用者重新導向至您的使用者集區/saml2/idpresponse端點。IdP 必須加密 SAML 2.0 規格定義的回應。如需詳細資訊，請參閱 Element <EncryptedAssertion> [OASIS 安全性聲明標記語言 \(SAML\) V2.0 的聲明和通訊協定](#)。
4. 您的使用者集區會使用私有金鑰解密 SAML 回應中的加密文字，並登入您的使用者。

Important

當您在使用者集區中啟用 SAML IdP 的回應加密時，IdP 必須使用提供者特定的公有金鑰來加密所有回應。Amazon Cognito 不接受來自您設定為支援加密之 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 都會收到自己的金鑰對。

AWS Management Console

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. 當您建立或編輯 SAML 身分提供者時，請在簽署請求和加密回應下，勾選標題為需要此提供者加密 SAML 聲明的核取方塊。
3. 從使用者集區的社交和外部提供者功能表中，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 SAML IdP。設定 SAML IdP 以使用憑證中的金鑰加密 SAML 回應。

API/CLI

設定 SAML 回應加密

使用 [CreateIdentityProvider](#) 或 [UpdateIdentityProvider](#) API 請求的 EncryptedResponses 參數設定回應加密。以下是支援請求簽署 ProviderDetails 的 IdP 範例。

```
"ProviderDetails": {
```

```
"MetadataURL" : "https://myidp.example.com/saml/metadata",
"IDPSignout" : "true",
"RequestSigningAlgorithm" : "rsa-sha256",
"EncryptedResponses" : "true",
"IDPInit" : "true"
}
```

若要從您的使用者集區取得加密憑證，請提出 [DescribeIdentityProvider](#) API 請求，並在回應參數 `ActiveEncryptionCertificate` 中擷取的值 `ProviderDetails`。儲存此憑證並將其做為來自使用者集區的登入請求的加密憑證提供給 IdP。

簽署 SAML 請求

向 IdP 證明 SAML 2.0 請求完整性的能力是 Amazon Cognito SP 起始的 SAML 登入的安全性優勢。每個具有網域的使用者集區都會收到使用者集區 X.509 簽署憑證。透過此憑證中的公有金鑰，使用者集區會將密碼編譯簽章套用至使用者集區在使用者選取 SAML IdP 時產生的登出請求。您可以選擇性地設定應用程式用戶端來簽署 SAML 登入請求。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符合您提供的使用者集區憑證中的公有金鑰。

AWS Management Console

設定 SAML 請求簽署

1. 建立 [使用者集區](#)、[應用程式用戶端](#) 和 SAML IdP。
2. 當您建立或編輯 SAML 身分提供者時，請在簽署請求和加密回應下，勾選標題為簽署 SAML 請求給此提供者的核取方塊。
3. 從使用者集區的社交和外部提供者功能表中，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 SAML IdP。設定 SAML IdP 以驗證傳入 SAML 請求的簽章。

API/CLI

設定 SAML 請求簽署

使用 [CreateIdentityProvider](#) 或 [UpdateIdentityProvider](#) API 請求的 `RequestSigningAlgorithm` 參數設定請求簽署。以下是支援請求簽署 `ProviderDetails` 的 IdP 範例。

```
"ProviderDetails": {
```

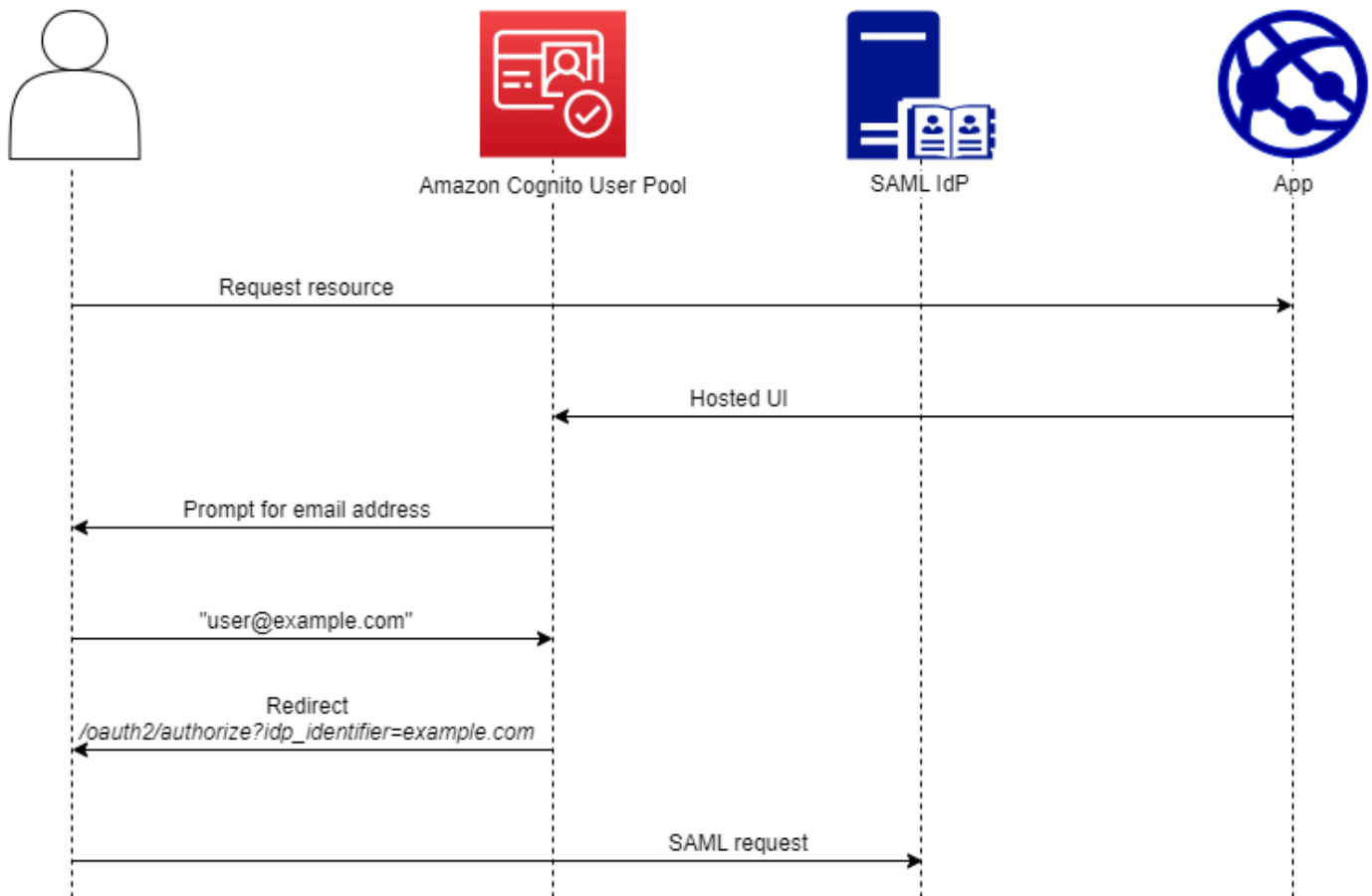
```

"MetadataURL" : "https://myidp.example.com/saml/metadata",
"IDPSignout" : "true",
"RequestSigningAlgorithm" : "rsa-sha256",
"EncryptedResponses" : "true",
"IDPInit" : "true"
}

```

SAML 身分提供者名稱和識別符

當您命名 SAML 身分提供者 (IdPs) 並指派 IdP 識別符時，您可以自動化向該提供者發出的 SP 起始登入和登出請求流程。如需提供者名稱字串限制的相關資訊，請參閱 [CreateIdentityProvider](#) 的 `ProviderName` 屬性。



您也可以為 SAML 供應商選擇最多 50 個識別符。識別符是使用者集區中 IdP 的易記名稱，而且在使用者集區中必須是唯一的。如果您的 SAML 識別符符合使用者的電子郵件網域，受管登入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。由於同一個組織可以擁有多個網域，因此單一 IdP 可以有多個識別符。

無論您是否使用電子郵件網域識別符，都可以在多租戶應用程式中使用識別符，將使用者重新導向至正確的 IdP。當您想要完全略過受管登入時，您可以自訂呈現給使用者的連結，讓他們[授權端點](#)直接透過重新導向至其 IdP。若要使用識別符登入您的使用者並重新導向至其 IdP，請在其初始授權請求的請求參數 `idp_identifier=myidp.example.com` 中包含 格式的識別符。

將使用者傳遞至 IdP 的另一個方法是使用下列 URL 格式的 IdP `identity_provider` 名稱填入 參數。

```
https://mydomain.auth.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 SAML IdP 登入後，IdP 會使用 HTTP POST 內文中的 SAML 回應將使用者重新導向至 `/saml2/idpresponse` 端點。Amazon Cognito 會處理 SAML 聲明，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。您的使用者以此方式完成身分驗證後，他們只會與 IdP 和應用程式的網頁互動。

使用網域格式的 IdP 識別符，受管登入會在登入時請求電子郵件地址，然後在電子郵件網域符合 IdP 識別符時，將使用者重新導向至其 IdP 的登入頁面。例如，您建置的應用程式需要兩個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，您已設定了兩個 IdP，每家公司各一個，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`。

在您的應用程式中，叫用應用程式用戶端的受管登入，以提示每個使用者輸入其電子郵件地址。Amazon Cognito 從電子郵件地址衍生網域、將網域與 IdP 與網域識別符建立關聯，並將您的使用者重新導向至正確的 IdP，並將請求重新導向至包含 `idp_identifier` 請求參數[授權端點](#)的。例如，如果使用者輸入 `bob@exampleA.co.uk`，則與互動的下一個頁面是位於的 IdP 登入頁面 `https://auth.exampleA.co.uk/sso/saml`。

您也可以獨立實作相同的邏輯。在您的應用程式中，您可以建置自訂表單來收集使用者輸入，並根據您自己的邏輯將其與正確的 IdP 建立關聯。您可以為每個應用程式租用戶產生自訂入口網站，其中每個連結都會連結到請求參數中具有租用戶識別符的授權端點。

若要收集電子郵件地址並剖析受管登入中的網域，請為您指派給應用程式用戶端的每個 SAML IdP 指派至少一個識別符。根據預設，受管登入畫面會顯示您指派給應用程式用戶端的每個 IdPs 的按鈕。不過，如果您已成功指派識別符，您的傳統託管 UI 登入頁面如下所示。

Note

在傳統託管 UI 中，當您將識別符指派給 IdPs 時，應用程式用戶端的登入頁面會自動提示電子郵件地址。在受管登入體驗中，您必須在品牌編輯器中啟用此行為。在身分驗證行為設定類別中，選取供應商顯示標題下的網域搜尋輸入。

受管登入中的網域剖析需要您使用網域做為 IdP 識別符。如果您為應用程式用戶端的每個 SAML IdPs 指派任何類型的識別符，則該應用程式的受管登入將不再顯示 IdP 選擇按鈕。當您想要使用電子郵件剖析或自訂邏輯來產生重新導向時，請新增 SAML 的 IdP 識別符。當您想要產生無提示重新導向，也希望受管登入頁面顯示 IdPs 清單時，請不要指派識別符，並在授權請求中使用 `identity_provider` 請求參數。

- 如果您只將一個 SAML IdP 指派給應用程式用戶端，受管登入頁面會顯示按鈕以使用該 IdP 登入。
- 如果您為應用程式用戶端啟用的每個 SAML IdP 指派識別符，則受管登入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 IdPs 但沒有將識別符指派給所有 IdP，受管登入頁面會顯示一個按鈕，以使用每個指派的 IdP 登入。
- 如果您將識別符指派給 IdPs 且希望受管登入頁面顯示選取的 IdP 按鈕，請將沒有識別符的新 IdP 新增至您的應用程式用戶端，或建立新的應用程式用戶端。您也可以刪除現有的 IdP，並在沒有識別符的情況下再次新增它。如果您建立新的 IdP，SAML 使用者將建立新的使用者設定檔。此重複作用中使用者可能會在您變更 IdP 組態的月份產生帳單影響。

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](#)。

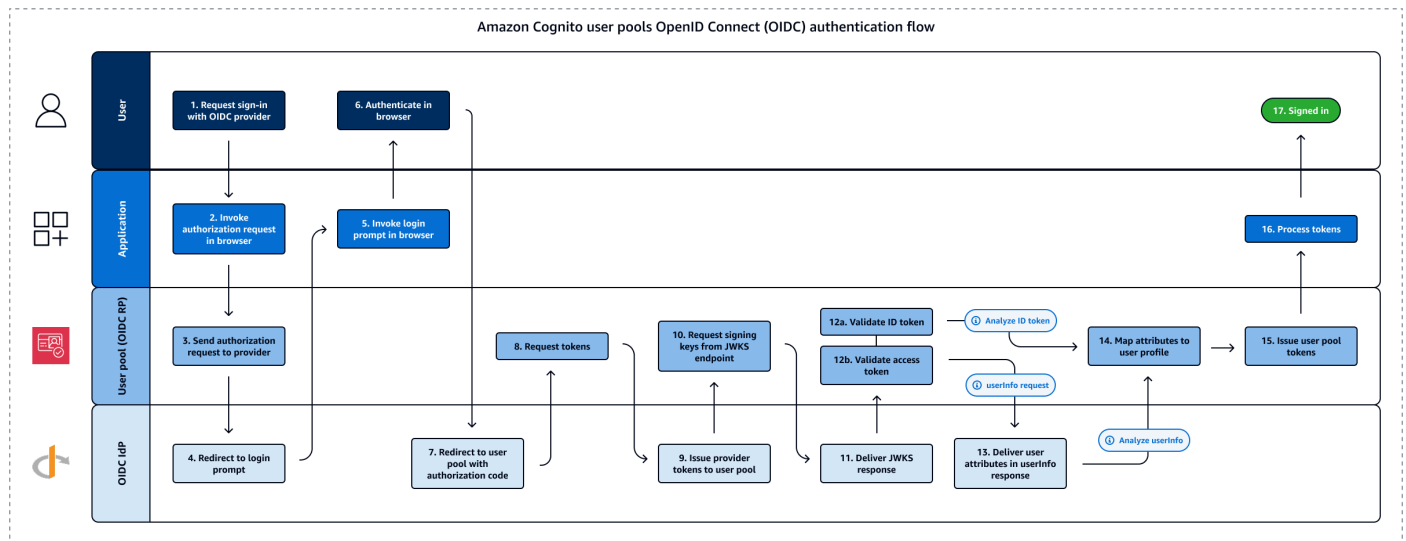
搭配使用者集區使用 OIDC 身分提供者

使用者可以使用其現有帳戶從 OpenID Connect (OIDC) 身分提供者 (IdPs) 登入您的應用程式。透過 OIDC 提供者，獨立單一登入系統的使用者可以在應用程式接收使用者集區共用格式的 OIDC 字符時提供現有的登入資料。若要設定 OIDC IdP，請將您的 IdP 設定為 RP 來處理使用者集區，並將您的應用程式設定為 IdP 來處理使用者集區。Amazon Cognito 做為多個 OIDC IdPs 與您的應用程式之間的中繼步驟。您的使用者集區會將屬性映射規則套用至提供者直接傳遞至使用者集區的 ID 和存取字符中的

宣告。然後，Amazon Cognito 會根據映射的使用者屬性，以及您透過 [Lambda 觸發](#) 對身分驗證流程所做的任何其他調整來發出新權杖。

使用 OIDC IdP 登入的使用者不需要提供新的登入資料或資訊，即可存取您的使用者集區應用程式。您的應用程式可以無提示地將其重新導向至其 IdP 以進行登入，並將使用者集區作為背景工具，以標準化應用程式的字符格式。若要進一步了解 IdP 重新導向，請參閱 [授權端點](#)。

如同其他第三方身分提供者，您必須向 OIDC 提供者註冊應用程式，並取得要連線至使用者集區之 IdP 應用程式的相關資訊。使用者集區 OIDC IdP 需要用戶端 ID、用戶端秘密、您要請求的範圍，以及提供者服務端點的相關資訊。您的使用者集區可以從探索端點探索提供者 OIDC 端點，也可以手動輸入。您還必須檢查提供者 ID 字符，並在 IdP 和使用使用者集區中的屬性之間建立屬性映射。



如需此身分驗證流程的詳細資訊 [OIDC 使用者集區 IdP 身分驗證流程](#)，請參閱。

Note

Amazon Cognito 使用者集區提供透過第三方 (聯合身分) 登入。此功能獨立於與 Amazon Cognito 身分集區的 OIDC 聯合。

您可以透過 或使用使用者集區 API 方法 [CreateIdentityProvider](#) AWS Management Console AWS CLI，將 OIDC IdP 新增至 中的使用者集區。

主題

- [先決條件](#)
- [向 OIDC IdP 註冊應用程式](#)

- [新增 OIDC IdP 到您的使用者集區](#)
- [測試 OIDC IdP 組態](#)
- [OIDC 使用者集區 IdP 身分驗證流程](#)

先決條件

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` 用戶端身分驗證。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` 用戶端身分驗證。如需用戶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 `jwtks_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 `jwtks_uri` 發佈金鑰 ID `kid` 宣告並在其權杖中包含 `kid` 宣告。
 - 呈現具有有效根 CA 信任鏈的未過期公有金鑰。

向 OIDC IdP 註冊應用程式

將 OIDC IdP 新增至使用者集區組態並將其指派給應用程式用戶端之前，請在 IdP 中設定 OIDC 用戶端應用程式。您的使用者集區是使用 IdP 管理身分驗證的依賴方應用程式。

向 OIDC IdP 註冊

1. 向 OIDC IdP 建立開發人員帳戶。

OIDC IdPs 的連結

OIDC IdP	安裝方法	OIDC 探索 URL
Salesforce	Salesforce 做為 OpenID Connect 身分提供者	<code>https://MyDomainName.my.salesforce.com</code>

OIDC IdP	安裝方法	OIDC 探索 URL
		<code>/.well-known/openid-configuration</code>
OneLogin	連接已啟用 OIDC 的應用程式	<code>https://<i>your-domain</i>.onelogin.com / oidc/2/.well-known/openid-configuration</code>
JumpCloud	使用 OIDC 的 SSO	<code>https://oauth.id.jumpcloud.com/.well-known/openid-configuration</code>
Okta	安裝 Okta 身分提供者	<code>https://<i>Your Okta subdomain</i>.okta.com /.well-known/openid-configuration</code>
Microsoft Entra ID	Microsoft 身分平台上的 OpenID Connect	<code>https://login.microsoftonline.com/ <i>{tenant}</i>/v2.0</code> 的值 <i>tenant</i> 可以包含租戶 ID、organizations、common 或 consumers。

- 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.auth.us-east-1.amazoncognito.com/oauth2/idpresponse
```

- 選取您希望使用者目錄與使用者集區共用的[範圍](#)。OIDC IdPs 需要範圍 `openid` 才能提供任何使用者資訊。需要 `email` 範圍才能授予 `email` 和 `email_verified` [宣告](#) 的存取權。OIDC 規格中的其他範圍 `profile` 適用於所有使用者屬性，`phone` 適用於 `phone_number` 和 `phone_number_verified`。
- OIDC IdP 為您提供用戶端 ID 和用戶端秘密。請記下這些值，並將其新增至您稍後新增至使用者集區的 OIDC IdP 組態。

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.auth.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.auth.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。

新增 OIDC IdP 到您的使用者集區

設定 IdP 之後，您可以將使用者集區設定為使用 OIDC IdP 處理身分驗證請求。

Amazon Cognito console

在主控台中新增 OIDC IdP

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. 選擇社交和外部提供者選單，然後選取新增身分提供者。
5. 選擇 OpenID Connect IdP。
6. 輸入唯一的供應商名稱。
7. 輸入 IdP 用戶端 ID。這是您在 OIDC IdP 中建置的應用程式用戶端 ID。您提供的用戶端 ID 必須是您已設定 回呼 URL 的 OIDC 供應商 `https://[your user pool domain]/oauth2/idpresponse`。
8. 輸入 IdP 用戶端秘密。這必須是上一個步驟中相同應用程式用戶端的用戶端秘密。
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 IdP 可能會提示使用者在登入時同意提供這些屬性給您的應用程式。

10. 選擇屬性請求方法。IdPs 可能需要對其 userInfo 端點的請求格式化為 GET 或 POST。例如，Amazon Cognito userInfo 端點需要 HTTP GET 請求。
11. 針對您希望使用者集區決定 IdP 中金鑰 OIDC 聯合端點路徑的方式，選擇設定方法。一般而言，IdPs 會在發行者基礎 URL 託管 `well-known/openid-configuration` 端點。如果您的提供者發生這種情況，自動填入發行者 URL 選項會為您顯示該基本 URL、嘗試從該處存取 `well-known/openid-configuration` 路徑，並讀取其中列出的端點。您可能有非典型的端點路徑，或希望透過替代代理將請求傳遞至一或多個端點。在此情況下，請選取手動輸入，並指定 `authorization`、`userInfo`、`token` 和 `jwt_uri` 端點的路徑。

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。

12. 在 OpenID Connect 提供者和使用者集區之間的映射屬性下設定屬性映射規則。使用者集區屬性是 Amazon Cognito 使用者設定檔中的目的地屬性，而 OpenID Connect 屬性是您希望 Amazon Cognito 在 ID 金鑰宣告或 `userInfo` 回應中找到的來源屬性。Amazon Cognito 會自動將目的地使用者設定檔 `username` 中的 OIDC 宣告子 映射至。

如需詳細資訊，請參閱 [將 IdP 屬性映射至設定檔和字符](#)。

13. 選擇新增身分提供者。
14. 從應用程式用戶端功能表中，從清單中選擇應用程式用戶端。導覽至登入頁面索引標籤，然後在受管登入頁面組態下，選取編輯。尋找身分提供者並新增新的 OIDC IdP。
15. 選擇儲存變更。

API/CLI

請參閱 [CreateIdentityProvider](#) 中範例二的 OIDC 組態。您可以修改此語法，並將其用作 [create-identity-provider](#) `UpdateIdentityProvider` 的 `CreateIdentityProvider`、或 `--cli-input-json` 輸入檔案的請求內文。

測試 OIDC IdP 組態

在您的應用程式中，您必須在使用者的用戶端中叫用瀏覽器，以便他們可以使用其 OIDC 供應商登入。在您完成上述章節中的設定程序後，向您的供應商測試登入。下列範例 URL 會使用字首網域載入使用者集區的登入頁面。

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```


此連結是當您前往應用程式用戶端功能表、選取應用程式用戶端、導覽至登入頁面索引標籤，然後選取檢視登入頁面時，Amazon Cognito 會引導您前往的頁面。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](#)。如需應用程式用戶端的詳細資訊，包括用戶端 IDs 和回呼 URLs，請參閱 [使用應用程式用戶端的應用程式特定設定](#)。

下列範例連結 [授權端點](#) 會使用 `identity_provider` 查詢參數，MyOIDCIp 從 設定供應商的無提示重新導向。此 URL 會使用受管登入略過互動式使用者集區登入，並直接前往 IdP 登入頁面。

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
identity_provider=MyOIDCIp&response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com
```

OIDC 使用者集區 IdP 身分驗證流程

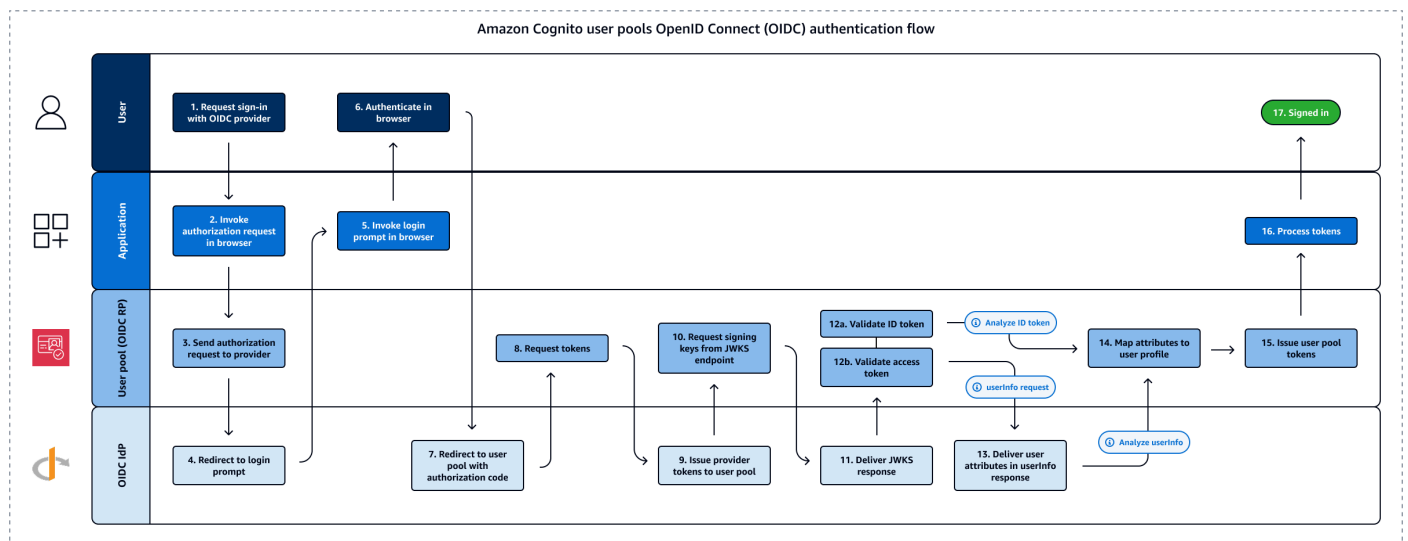
透過 OpenID Connect (OIDC) 登入，您的使用者集區會使用您的身分提供者 (IdP) 自動化授權碼登入流程。您的使用者使用其 IdP 完成登入後，Amazon Cognito 會在外部供應商的 `oauth2/idpresponse` 端點收集其程式碼。使用產生的存取權杖，您的使用者集區會查詢 IdP `userInfo` 端點以擷取使用者屬性。然後，您的使用者集區會將收到的屬性與您設定的屬性映射規則進行比較，並相應地填入使用者的設定檔和 ID 字符。

您在 OIDC 供應商組態中請求的 OAuth 2.0 範圍會定義 IdP 提供給 Amazon Cognito 的使用者屬性。作為最佳安全實務，僅請求對應至您要映射至使用者集區的屬性的範圍。例如，如果您的使用者集區請求 `openid profile`，您會取得所有可能的屬性，但如果您請求 `openid email phone_number`，則只會取得使用者的電子郵件地址和電話號碼。您可以設定從 [OIDC IdPs 請求](#) 的範圍，與您在 [應用程式用戶端](#) 和使用使用者集區身分驗證請求中授權和請求的範圍不同。

當您的使用者使用 OIDC IdP 登入您的應用程式時，您的使用者集區會執行下列身分驗證流程。

1. 使用者存取您的受管登入頁面，並選擇使用其 OIDC IdP 登入。
2. 您的應用程式會將使用者的瀏覽器導向使用者集區的授權端點。
3. 您的使用者集區會將請求重新導向至 OIDC IdP 的授權端點。
4. 您的 IdP 會顯示登入提示。
5. 在您的應用程式中，使用者的工作階段會顯示 OIDC IdP 的登入提示。
6. 使用者輸入其 IdP 的登入資料，或為已驗證的工作階段提供 Cookie。
7. 使用者驗證後，OIDC IdP 會使用授權碼重新導向至 Amazon Cognito。

8. 您的使用者集區會交換 ID 和存取權杖的授權碼。當您使用範圍設定 IdP 時，Amazon Cognito 會收到存取字串 `openid`。ID 字串中的宣告和 `userInfo` 回應是由 IdP 組態中的其他範圍決定，例如 `profile` 和 `email`。
9. 您的 IdP 會發出請求的權杖。
- 10 您的使用者集區會從 IdP 組態中的發行者 URLs 決定 IdP `jwks_uri` 端點的路徑，並從 JSON Web 金鑰集 (JWKS) 端點請求權杖簽署金鑰。
- 11 IdP 會從 JWKS 端點傳回簽署金鑰。
- 12 您的使用者集區會從字串中的簽章和過期資料驗證 IdP 字串。
- 13 您的使用者集區會使用存取權杖授權 IdP `userInfo` 端點的請求。IdP 會根據存取權杖範圍以使用者資料回應。
- 14 您的使用者集區會將 IdP 的 ID 字串和 `userInfo` 回應與使用者集區中的屬性映射規則進行比較。它會將映射的 IdP 屬性寫入使用者集區設定檔屬性。
- 15 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。
- 16 您的應用程式會處理使用者集區字串並登入使用者。



Note

Amazon Cognito 會取消未在 5 分鐘內完成的身分驗證請求，並將使用者重新導向至受管登入。此頁面會顯示「Something went wrong」錯誤訊息。

OIDC 是位於 OAuth 2.0 頂端的身分層級，指定由 IdP 向 OIDC 用戶端應用程式 (依賴方) 發出的 JSON 格式 (JWT) 身分字。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

當您的使用者透過第三方 OIDC 供應商完成登入時，受管登入會從 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，請檢查權杖 kid 宣告中的簽署金鑰 ID 是否於提供者 jwks_uri 端點列出。Amazon Cognito 會針對其處理的每個 IdP ID 字，從 IdP 組態中的 JWKS 端點重新整理簽署金鑰。
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杖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。

將 IdP 屬性映射至設定檔和字符

身分提供者 (IdP) 服務，包括 Amazon Cognito，通常可以記錄使用者的詳細資訊。您可能想知道他們服務的公司、如何聯絡他們，以及其他識別資訊。但是，這些屬性採用的格式在提供者之間有差異。例如，使用使用者集區設定來自三個不同廠商的三個 IdPs，並檢查每個廠商的範例 SAML 聲明、ID 字符或 `userInfo` 承載。一個會將使用者的電子郵件地址表示為 `email`，另一個表示為 `emailaddress`，第三個表示為 `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`。

將 IdPs 與使用者集區整合的主要優點是能夠將各種屬性名稱映射到具有一致、可預測、共用屬性名稱的單一 OIDC 字符結構描述。如此一來，您的開發人員就不需要維護處理複雜種類的單一登入事件的邏輯。此格式合併是屬性映射。使用者集區屬性映射會將 IdP 屬性名稱指派給對應的使用者集區屬性名稱。例如，您可以將使用者集區設定為將 `emailaddress` 宣告的值寫入標準使用者集區屬性 `email`。

每個使用者集區 IdP 都有單獨的屬性映射結構描述。若要指定 IdP 的屬性映射，請在 Amazon Cognito 主控台、AWS SDK 或使用者集區 REST API 中設定使用者集區身分提供者。

對應的須知事項

在您開始設定使用者屬性映射之前，請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杖，OIDC 提供者則會顯示存取和 ID 權杖。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 傳遞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。Amazon Cognito 將此屬性值加上 IdP 的名稱，例如 `MyOIDCIdP_[sub]`。當您希望聯合身分使用者具有與外部使用者目錄中的屬性完全相符的屬性時，請將該屬性對應至等 Amazon Cognito 登入屬性 `preferred_username`。

身分提供者	username 資源屬性
Facebook	id
Google	sub
登入 Amazon	user_id
使用 Apple 登入	sub
SAML 供應商	NameID
OpenID Connect (OIDC) 供應商	sub

- 當使用者集區不區分大小寫時，Amazon Cognito 會以聯合身分使用者自動產生的使用者名稱將使用者名稱來源屬性轉換為小寫。以下是區分大小寫的使用者集區的範例使用者名稱：MySAML_TestUser@example.com。以下是不區分大小寫的使用者集區的相同使用者名稱：MySAML_testuser@example.com。

在不區分大小寫的使用者集區中，處理使用者名稱的 Lambda 觸發必須考慮對使用者名稱來源屬性的任何混合大小寫宣告進行此修改。若要將您的 IdP 連結至與目前使用者集區具有不同區分大小寫設定的使用者集區，請建立新的使用者集區。

- 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 主控台中建立可變自訂屬性，請在註冊功能表中選取新增自訂屬性時，為您新增的屬性啟用可變核取方塊。或者，若您使用 [CreateUserPool](#) API 操作來建立使用者集區，可將每個屬性的 Mutable 參數設為 true。如果您的 IdP 傳送對應不可變屬性的值，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 傳送映射不可寫入屬性的值，Amazon Cognito 不會設定屬性值並繼續驗證。

- 當 IdP 屬性包含多個值時，Amazon Cognito 會將所有值扁平化為以方括號字元 [和 括住的單一逗號分隔字串]。Amazon Cognito URL 表單編碼包含非英數字元的值 -，但 .、*、和 除外_。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 來指定使用者集區的 IdP 屬性映射。

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

- 登入 [Amazon Cognito 主控台](#)。如果出現提示，請輸入您的 AWS 登入資料。
- 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。
- 選擇社交和外部供應商選單。
- 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[搭配使用者集區使用社交身分提供者](#)。

- 針對您需要對應的每個屬性，完成下列步驟：
 - 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。
 - 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。
 - 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)。
- 選擇儲存變更。

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 登入資料。
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. 選擇社交和外部供應商選單。
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。有些 IdP 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
```

- 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。

為您的使用者集區 (AWS CLI 和 AWS API) 指定身分提供者屬性映射

[CreateIdentityProvider](#) 或 [UpdateIdentityProvider](#) 的下列請求內文 phone_number 會依該順序將 SAML 提供者 "MyIdP" 屬性 emailaddress、birthdate 和 映射 phone 至使用者集區屬性 birthdate、email 和 。這是 SAML 2.0 供應商的完整請求內文，您的請求內文會根據 IdP 類型和特定詳細資訊而有所不同。屬性映射位於 AttributeMapping 參數中。

```
{
  "AttributeMapping": {
    "email" : "emailaddress",
    "birthdate" : "birthdate",
    "phone_number" : "phone"
  },
  "IdpIdentifiers": [
    "IdP1",
```

```
    "pdxsaml"
  ],
  "ProviderDetails": {
    "IDPInit": "true",
    "IDPSignout": "true",
    "EncryptedResponses" : "true",
    "MetadataURL": "https://auth.example.com/sso/saml/metadata",
    "RequestSigningAlgorithm": "rsa-sha256"
  },
  "ProviderName": "MyIdP",
  "ProviderType": "SAML",
  "UserPoolId": "us-west-2_EXAMPLE"
}
```

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
```

- API/SDK : [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`

- API/SDK : [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>`

- API/SDK : [DescribeIdentityProvider](#)

將聯合身分使用者連結至現有的使用者描述檔

通常，同一個使用者擁有的設定檔可以具有多個您已連線至您使用者集區的身分提供者 (IdP)。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](#) 告訴 Amazon Cognito 將聯合目錄中使用者的唯一 ID 視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。您可以在 API `AdminLinkProviderForUser` 請求中，在計劃的預備任務或中，建立本機設定檔，並在其第一次登入之前隨時將其連結至您的聯合身分使用者[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。這樣，您可以使用存取陳述式設定本機設定檔，並將其身分陳述式與您的提供者保持在最新狀態。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的使用者具有其建立所用的使用者名稱，以及包含其連結身分詳細資訊的 `identities` 屬性。

Important

由於 `AdminLinkProviderForUser` 允許具有外部聯合身分的使用者以使用者集區中現有使用者的身分登入，因此只能搭配應用程式擁有者所信任的外部 IdP 和供應者屬性使用是至關重要的。

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。每個客戶都透過 Active Directory Federation Services (ADFS) 登入到您的應用程式。您的 IT 管理員 Carlos 在您的每個客戶網域中都有一個帳戶。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 IdP 向 Amazon Cognito 出示 Carlos 在其 SAML 聲明的 email 宣告中的電子郵件地址 `msp_carlos@example.com`。您在使用者集區中建立使用者名稱為 Carlos 的使用者。下列 AWS Command Line Interface (AWS CLI) 命令會從 IdP ADFS1、ADFS2 與 ADFS3 連結 Carlos 的身分。

Note

您可以根據特定的屬性宣告連結使用者。這種能力是 OIDC 和 SAML IdP 所獨有的。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。
- 您可以從最多五個 IdP 屬性宣告將使用者連結至每個 IdP，如 AdminLinkProviderForUser API 請求 SourceUser 中的 ProviderAttributeName 參數所定義。例如，如果您已將至少一位使用

者連結至來源屬性 email、phone、given_name、department 和 location，則您只能將這五個屬性之一上的其他使用者連結。

- 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 [AdminSetUserPassword](#) API 請求中，為自動建立的聯合身分使用者設定檔設定密碼。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。他們還可以在已進行權杖驗證的 API 請求 (如 [ChangePassword](#) 和 [UpdateUserAttributes](#)) 中修改其密碼和屬性。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。若要移除使用者不再需要的設定檔，請使用對等參數進行 [AdminDisableProviderForUser](#) API 請求。

請參閱 [AdminLinkProviderForUser](#) 以瞭解 AWS 開發套件中的其他命令語法與範例。

使用者集區受管登入

您可以選擇 Web 網域來託管使用者集區的服務。當您新增網域時，Amazon Cognito 使用者集區會取得下列函數，統稱為受管登入。

- 做為與 OAuth 2.0 和 OpenID Connect (OIDC) 搭配使用之應用程式身分提供者 (IdP) [的授權伺服器](#)。授權伺服器會[路由請求](#)、[發出和管理 JSON Web 權杖 \(JWTs\)](#)，並提供[使用者屬性資訊](#)。
- ready-to-use 使用者介面 (UI)，用於登入、登出和密碼管理等身分驗證操作。受管登入頁面可做為身分驗證服務的 Web 前端。
- SAML 2.0 IdPs、OIDC IdPs、Facebook、Login with Amazon、Login with Apple 和 Google 的服務提供者 (SP) 或依賴方 (RP)。

與受管登入共用一些功能的另一個選項是傳統託管 UI。傳統託管 UI 是受管登入服務的第一代版本。託管 UI IdP 和 RP 服務通常具有與受管登入相同的特性，但登入頁面的設計更簡單，功能更少。例如，傳統託管 UI 中無法使用通行金鑰登入。在 Lite [功能計劃](#) 中，傳統託管 UI 是您使用者集區網域服務的唯一選項。

受管登入頁面是 Web 介面的集合，用於使用者集區中的基本註冊、登入、多重驗證和密碼重設活動。當您想要讓使用者選擇登入選項時，他們也會將使用者連線至一或多個第三方身分提供者 (IdPs)。當您想要驗證和授權使用者時，您的應用程式可以在使用者的瀏覽器中叫用您的受管登入頁面。

您可以使用自訂標誌、背景和樣式，讓受管登入使用者體驗符合您的品牌。對於可能套用至受管登入 UI 的品牌，您有兩個選項：受管登入的品牌編輯器，以及託管 UI 的託管 UI（傳統）品牌。

品牌編輯器

Amazon Cognito 主控台中具有 up-to-date 身分驗證選項和視覺化編輯器的更新使用者體驗。

託管 UI 品牌

Amazon Cognito 使用者集區先前採用者的熟悉使用者體驗。託管 UI 的品牌是檔案型系統。若要將品牌套用到託管 UI 頁面，您可以上傳標誌影像檔案和設定數個預定 CSS 樣式選項值的檔案。

品牌編輯器不適用於使用者集區的所有功能計劃。如需詳細資訊，請參閱 [使用者集區功能計劃](#)。

如需建構受管登入和託管 UI 服務請求的詳細資訊，請參閱 [使用者集區端點和受管登入參考](#)。

Note

Amazon Cognito 受管登入不支援具有自訂身分驗證 [挑戰 Lambda 觸發條件的自訂身分驗證](#)。

主題

- [受管登入當地語系化](#)
- [使用 設定受管登入 AWS Amplify](#)
- [使用 Amazon Cognito 主控台設定受管登入](#)
- [檢視您的登入頁面](#)
- [自訂身分驗證頁面](#)
- [受管登入和託管 UI 的須知事項](#)
- [設定使用者集區網域](#)

- [將品牌套用到受管登入頁面](#)

受管登入當地語系化

受管登入預設為使用者互動頁面中的英文語言。您可以針對您選擇的語言，在本機顯示受管登入頁面。可用的語言是 中可用的語言 AWS Management Console。在您分發給使用者的連結中，新增lang查詢參數，如下列範例所示。

```
https://<your domain>/oauth2/authorize?lang=es&response_type=code&client_id=<your app client id>&redirect_uri=<your relying-party url>
```

Amazon Cognito 在具有 lang 參數的初始請求之後，會在使用者的瀏覽器中以使用者的語言偏好設定設定 Cookie。設定 Cookie 後，語言選擇會持續存在，而不會顯示或要求您在請求中包含 參數。例如，在使用者使用 lang=de 參數提出登入請求後，其受管登入頁面會以德文轉譯，直到他們清除 Cookie 或使用新的當地語系化參數提出新請求，例如 lang=en。

當地語系化僅適用於受管登入。您必須加入 Essentials 或 Plus [功能計劃](#)，並已指派您的網域使用[受管登入品牌](#)。

您的使用者在受管登入中進行的選擇不適用於[自訂電子郵件或簡訊寄件者觸發](#)。當您實作這些觸發條件時，您必須使用其他機制，例如 locale 屬性，來判斷使用者的偏好語言。

提供下列語言。

受管登入語言

語言	代碼
德文	de
英文	zh
西班牙文	es
法文	fr
印度尼西亞語	id
義大利文	it
日文	ja

語言	代碼
韓文	ko
葡萄牙文 (巴西)	pt-BR
簡體中文	zh-CN
繁體中文	zh-TW

使用 設定受管登入 AWS Amplify

如果您使用 AWS Amplify 將身分驗證新增至 Web 或行動應用程式，您可以在 Amplify 命令列界面 (CLI) 和 Amplify 架構中的程式庫中設定受管登入頁面。若要將身分驗證新增至您的應用程式，請將 Auth 類別新增至您的專案。然後，在您的應用程式中，使用 Amplify 用戶端程式庫驗證使用者集區使用者。

您可以叫用受管登入頁面進行身分驗證，也可以透過重新導向至 IdP 的授權端點來聯合使用者。使用者成功向供應商進行身分驗證後，Amplify 會在您的使用者集區中建立新的使用者，並將使用者的字符傳遞至您的應用程式。

下列範例示範如何使用 AWS Amplify，在應用程式中設定社交供應商的受管登入。

- [AWS Amplify JavaScript 的身分驗證。](#)
- [AWS Amplify Swift 的身分驗證。](#)
- [AWS Amplify Flutter 的身分驗證。](#)
- [AWS Amplify Android 的身分驗證。](#)

使用 Amazon Cognito 主控台設定受管登入

受管登入和託管 UI 的第一個需求是使用者集區網域。在使用者集區主控台中，導覽至使用者集區的網域索引標籤，並新增 Cognito 網域或自訂網域。您也可以在建立新使用者集區的過程中選擇網域。如需詳細資訊，請參閱[設定使用者集區網域](#)。當您的使用者集區中的網域處於作用中狀態時，所有應用程式用戶端都會提供該網域上的公有身分驗證頁面。

當您建立或修改使用者集區網域時，您可以為網域設定品牌版本。此品牌版本是受管登入或託管 UI（傳統）的選擇。您選擇的品牌版本適用於在您的網域使用登入服務的所有應用程式用戶端。

下一個步驟是從使用者集區的[應用程式用戶端](#)索引標籤建立應用程式用戶端。在建立應用程式用戶端的過程中，Amazon Cognito 會要求您提供有關應用程式的資訊，然後提示您選取傳回 URL。傳回 URL 也稱為依賴方 (RP) URL、重新導向 URI 和回呼 URL。這是應用程式執行的 URL，例如 `https://www.example.com` 或 `myapp://example`。

在使用者集區中以品牌樣式設定網域和應用程式用戶端之後，您的受管登入頁面即可在網際網路上使用。

檢視您的登入頁面

在 Amazon Cognito 主控台中，選擇應用程式用戶端選單下應用程式用戶端登入頁面索引標籤中的檢視登入頁面按鈕。此按鈕會使用下列基本參數，帶您前往使用者集區網域中的登入頁面。

- 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

當您想要測試受管登入頁面的基本功能時，檢視登入頁面按鈕很有用。您的登入頁面將與您指派給[使用者集區網域](#)的品牌版本相符。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，檢視登入頁面連結自動產生的參數無法完全符合您應用程式的需求。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 [使用者集區端點和受管登入參考](#)。

登入網頁使用以下 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 relying-party url>
```

您可以從使用者集區中的網域功能表查詢使用者集區網域字串。在應用程式用戶端選單中，您可以識別應用程式用戶端 IDs、其回呼 URLs、其允許的範圍和其他組態。

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。

隱含授予的範例請求

您可以使用下列 URL 來檢視您的登入網頁，以取得 的隱含程式碼授予 `response_type=token`。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.auth.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://auth.example.com/
#id_token=eyJraaBcDeF1234567890&access_token=eyJraGhIjKlM1112131415&expires_in=3600&token_type=
```

自訂身分驗證頁面

在過去，Amazon Cognito 僅託管具有傳統託管 UI 的登入頁面，這是一種簡單的設計，可授予對身分驗證網頁的通用外觀。您可以使用標誌影像自訂 Amazon Cognito 使用者集區，並使用指定一些 CSS 樣式值的檔案調整某些樣式。稍後，Amazon Cognito 推出了受管登入，這是一種更新的託管身分驗證服務。受管登入是使用品牌編輯器更新 look-and-feel。品牌編輯器是無程式碼視覺化編輯器，以及比託管 UI 自訂體驗更大的選項套件。受管登入也引進了自訂背景影像和深色模式佈景主題。

您可以在使用者集區中的受管登入和託管 UI 品牌體驗之間切換。若要進一步了解如何自訂受管登入頁面，請參閱 [將品牌套用到受管登入頁面](#)。

受管登入和託管 UI 的須知事項

一小時受管登入和託管 UI 工作階段 Cookie

當使用者使用您的登入頁面或第三方提供者登入時，Amazon Cognito 會在其瀏覽器中設定 Cookie。使用此 Cookie，使用者可以使用相同的身分驗證方法重新登入一小時。當他們使用瀏覽器 Cookie 登入時，會取得持續您應用程式用戶端組態中指定持續時間的新權杖。使用者屬性或身分驗證因素的變更不會影響其使用瀏覽器 Cookie 重新登入的能力。

使用工作階段 Cookie 進行身分驗證不會將 Cookie 持續時間重設為額外一小時。如果使用者在最近成功的互動式身分驗證後超過一小時嘗試存取您的登入頁面，則必須再次登入。

確認使用者帳戶和驗證使用者屬性

對於使用者集區[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，受管登入和託管 UI 最有效。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。當您改為將使用者確認為使用者集區管理員時，您的登入頁面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](#)。

檢視您對組態所做的變更

如果您變更頁面樣式但未立即顯示，請等待幾分鐘，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字符 AWS Lambda。請參閱 GitHub 上的[解碼和驗證 Amazon Cognito JWT 權杖](#)。

AWS WAF Web ACLs

您可以設定使用者集區，使用 AWS WAF Web ACLs 中的規則來保護提供登入頁面和授權伺服器的網域。目前，只有當您受管登入品牌版本為託管 UI（傳統）時，您設定的規則才會套用到這些頁面。如需詳細資訊，請參閱[AWS WAF Web ACLs 和 Amazon Cognito 的須知事項](#)。

TLS 版本控制

受管登入和託管 UI 頁面需要傳輸中加密。Amazon Cognito 提供的使用者集區網域要求使用者的瀏覽器交涉最低 TLS 版本為 1.2。自訂網域支援使用 TLS 1.2 版的瀏覽器連線。託管 UI（傳統）不需要自訂網域的 TLS 1.2，但較新的受管登入需要自訂網域和字首網域的 TLS 1.2 版。由於 Amazon Cognito 會管理網域服務的組態，因此您無法修改使用者集區網域的 TLS 需求。

CORS 政策

受管登入和託管 UI 都不支援自訂的跨來源資源共用 (CORS) 原始伺服器政策。CORS 政策會防止使用者在其請求中傳遞身分驗證參數。而是在您的應用程式前端實作 CORS 政策。Amazon Cognito 會將 Access-Control-Allow-Origin: * 回應標頭傳回給下列端點的請求。

1. [權杖端點](#)
2. [撤銷端點](#)
3. [userInfo 端點](#)

Cookie

受管登入和託管 UI 會在使用者的瀏覽器中設定 Cookie。Cookie 符合網站未設定第三方 Cookie 之某些瀏覽器的要求。它們的範圍僅限於您的使用者集區端點，並包含下列項目：

- 每個請求的 XSRF-TOKEN Cookie。
- 重新導向使用者時的工作階段一致性 csrf-state Cookie。
- 當您的瀏覽器不支援 SameSite 屬性時，Amazon Cognito 會將工作階段一致性的 csrf-state-legacy Cookie 讀取為備用 Cookie。
- cognito 工作階段 Cookie，可保留成功的登入嘗試一小時。
- 在受管登入中保留使用者[語言當地語系化](#)選擇的 lang Cookie。
- 當使用者在受管登入頁面之間導覽時，保留必要資料的 page-data Cookie。

在 iOS 中，您可以[封鎖所有 Cookie](#)。此設定與受管登入或託管 UI 不相容。若要使用可能啟用此設定的使用者，請使用 AWS SDK 將使用者集區身分驗證建置到原生 iOS 應用程式。在此案例中，您可以建置自己的工作階段儲存體，而這些儲存體不是以 Cookie 為基礎。

受管登入版本變更的影響

請考慮下列新增網域和設定受管登入版本的效果。

- 當您使用受管登入或託管 UI（傳統）品牌新增字首網域時，最多可能需要 60 秒才能使用登入頁面。
- 當您使用受管登入或託管 UI（傳統）品牌新增自訂網域時，最多可能需要五分鐘的時間才能使用登入頁面。
- 當您變更網域的品牌版本時，可能需要最多四分鐘的時間，新的品牌版本才會提供登入頁面。
- 當您在受管登入和託管 UI（傳統）品牌之間切換時，Amazon Cognito 不會維護使用者工作階段。他們必須使用新的界面再次登入。

預設樣式

當您在 中建立應用程式用戶端時 AWS Management Console，Amazon Cognito 會自動將品牌樣式指派給您的應用程式用戶端。當您使用 [CreateUserPoolClient](#) 操作以程式設計方式建立應用程式用戶端時，不會建立品牌樣式。除非您使用 [CreateManagedLoginBranding](#) 請求建立受管登入，否則無法使用 AWS SDK 建立的應用程式用戶端。

受管登入身分驗證提示逾時

Amazon Cognito 會取消未在五分鐘內完成的身分驗證請求，並將使用者重新導向至受管登入。此頁面人會顯示「Something went wrong」錯誤訊息。

設定使用者集區網域

設定網域是設定使用者集區的選用部分。使用者集區網域託管使用者身分驗證、與第三方供應商的聯合，以及 OpenID Connect (OIDC) 流程的功能。它具有受管登入，這是預先建置的介面，用於註冊、登入和密碼復原等金鑰操作。它還託管標準 OpenID Connect (OIDC) 端點，例如[授權](#)、[userInfo](#) 和 [字符合](#)，用於machine-to-machine(M2M) 授權和其他 OIDC 和 OAuth 2.0 身分驗證和授權流程。

使用者在與使用者集區相關聯的網域使用受管登入頁面進行身分驗證。您有兩個設定此網域的選項：您可以使用預設的 Amazon Cognito 託管網域，也可以設定您擁有的自訂網域。

自訂網域選項有更多彈性、安全性和控制的選項。例如，熟悉的組織擁有網域可以鼓勵使用者信任，並讓登入程序更直覺。不過，自訂網域方法需要額外的額外負荷，例如管理 SSL 憑證和 DNS 組態。

端點 URLs 和 `/.well-known/jwks.json` 字符簽署金鑰 `/.well-known/openid-configuration` 的 OIDC 探索端點不會託管在您的網域上。如需詳細資訊，請參閱[身分提供者和依賴方端點](#)。

了解如何設定和管理使用者集區的網域，是將身分驗證整合到您的應用程式的重要步驟。使用使用者集區 API 和 AWS SDK 登入可以是設定網域的替代方案。以 API 為基礎的模型會直接在 API 回應中交付權杖，但對於使用使用者集區擴充功能做為 OIDC IdP 的實作，您必須設定網域。如需使用者集區中可用身分驗證模型的詳細資訊，請參閱[了解 API、OIDC 和受管登入頁面身分驗證](#)。

主題

- [使用者集區網域的須知事項](#)
- [使用 Amazon Cognito 字首網域進行受管登入](#)
- [使用您自己的網域進行受管登入](#)

使用者集區網域的須知事項

使用者集區網域是應用程式中 OIDC 依賴方和 UI 元素的服務點。當您規劃為使用者集區實作網域時，請考慮下列詳細資訊。

預留條款

您無法在 Amazon Cognito aws 字首網域的名稱 `cognito` 中使用文字 `amazon`、`或`。

探索端點位於不同的網域

使用者集區[探索端點](#) `.well-known/openid-configuration` 和 `.well-known/jwks.json` 不在您的使用者集區自訂或字首網域上。這些端點的路徑如下。

- `https://cognito-idp.Region.amazonaws.com/your user pool ID/.well-known/openid-configuration`
- `https://cognito-idp.Region.amazonaws.com/your user pool ID/.well-known/jwks.json`

網域變更的有效時間

Amazon Cognito 最多可能需要一分鐘的時間才能啟動或更新字首網域的品牌版本。自訂網域的變更最多可能需要五分鐘才能傳播。新的自訂網域最多可能需要一小時才能傳播。

同時自訂和字首網域

您可以使用自訂網域和擁有的前綴網域來設定使用者集區 AWS。由於使用者集區[探索端點](#)託管在不同網域，因此它們僅提供自訂網域。例如，您的 `openid-configuration` 會提供 "authorization_endpoint" 的單一值 `https://auth.example.com/oauth2/authorize`。

當您在使用者集區中同時有自訂網域和字首網域時，您可以使用自訂網域搭配 OIDC 供應商的完整功能。具有此組態之使用者集區中的字首網域沒有探索或 token-signing-key 端點，應相應地使用。

自訂網域偏好做為通行密鑰的依賴方 ID

當您使用[通行金鑰](#)設定使用者集區身分驗證時，您必須設定依賴方 (RP) ID。當您有自訂網域和字首網域時，您只能將 RP ID 設定為自訂網域。若要在 Amazon Cognito 主控台中將字首網域設定為 RP ID，請刪除您的自訂網域，或將字首網域的完整網域名稱 (FQDN) 輸入為第三方網域。

請勿在網域階層的不同層級使用自訂網域

您可以設定不同的使用者集區，在相同的頂層網域 (TLD) 中具有自訂網域，例如 `auth.example.com`：// `https://www...https://https://https://www./https://https://https://https://https://auth2.example.comhttps://` 受管登入工作階段 Cookie 對自訂網域和所有子網域有效，例如 `*.auth.example.com`。因此，您應用程式的任何使用者都不應該存取任何父系網域和子網域的受管登入。當自訂網域使用相同的 TLD 時，請將它們保持在相同的子網域層級。

假設您有一個具有自訂網域 `auth.example.com` 的使用者集區。然後，您可以建立另一個使用者集區，並指派自訂網域 `uk.auth.example.com`。使用者使用 `auth.example.com` 登入，並取得其瀏覽器提供給萬用字元路徑 `*.auth.example.com` 中任何網站的 Cookie。然後，他們會嘗試登入

uk.auth.example.com。它們會將無效的 Cookie 傳遞給您的使用者集區網域，並收到錯誤，而不是登入提示。相反地，具有適用於 *.auth.example.com 的 Cookie 的使用者在 auth2.example.com 上啟動登入工作階段沒有問題。

品牌版本

建立網域時，您可以設定品牌版本。您的選項是較新的受管登入體驗和傳統託管 UI 體驗。此選項適用於在您的網域託管服務的所有應用程式用戶端。

使用 Amazon Cognito 字首網域進行受管登入

受管登入的預設體驗託管在 AWS 擁有的網域上。這種方法進入的障礙很低，選擇字首名稱且處於作用中狀態，但沒有自訂網域的信任啟發功能。Amazon Cognito 網域選項與自訂網域選項之間沒有成本差異。唯一的差別是您引導使用者前往的網址中的網域。對於第三方 IdP 重新導向和用戶端憑證流程的情況，託管網域幾乎沒有可見效果。自訂網域更適合使用者使用受管登入登入，並且會與不符合應用程式網域的身分驗證網域互動的情況。

託管的 Amazon Cognito 網域具有您選擇的字首，但託管於根網域 amazoncognito.com。以下是範例：

```
https://cognitoexample.auth.ap-south-1.amazoncognito.com
```

所有字首網域都遵循此格式：*prefix*.auth.AWS ## *code*amazoncognito.com。 [自訂網域](#)使用者集區可以在您擁有的任何網域上託管受管登入或託管 UI 頁面。

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使用者集區網域 AWS Management Console，請參閱 [建立應用程式用戶端](#)。

主題

- [先決條件](#)
- [設定 Amazon Cognito 網域字首](#)
- [驗證您的登入頁面](#)

先決條件

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](#)。

設定 Amazon Cognito 網域字首

您可以使用 AWS Management Console 或 AWS CLI 或 API 來設定使用者集區網域。

Amazon Cognito console

設定網域

1. 導覽至品牌下的網域選單。
2. 在網域旁，選擇動作，然後選取建立 Cognito 網域。如果您已設定使用者集區字首網域，請選擇刪除 Cognito 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。如需設定自訂網域的資訊，請參閱 [使用您自己的網域進行受管登入](#)。
4. 選擇品牌版本。您的品牌版本適用於該網域中的所有使用者互動頁面。您的使用者集區可以託管所有應用程式用戶端的受管登入或託管 UI 品牌。

Note

您可以擁有自訂網域和字首網域，但 Amazon Cognito 僅提供自訂網域的 `/.well-known/openid-configuration` 端點。

5. 選擇 Create (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> --managed-login-version 2`

- 使用者集區 API 操作：[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>`

- 使用者集區 API 操作：[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>`

- 使用者集區 API 操作：[DeleteUserPoolDomain](#)

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。

使用您自己的網域進行受管登入

設定應用程式用戶端之後，您可以使用[受管登入](#)的網域服務的自訂網域來設定使用者集區。使用自訂網域，使用者可以使用您自己的 Web 地址登入您的應用程式，而不是預設amazoncognito.com[字首網](#)

域。自訂網域使用熟悉的網域名稱來改善使用者對應用程式的信任，尤其是當根網域與託管應用程式的網域相符時。自訂網域可以提高組織安全需求的合規性。

自訂網域有一些先決條件，包括使用者集區、應用程式用戶端和您擁有的 Web 網域。自訂網域還需要自訂網域的 SSL 憑證，在美國東部 AWS Certificate Manager（維吉尼亞北部）使用 (ACM) 管理。Amazon Cognito 會建立 Amazon CloudFront 分佈，並使用您的 ACM 憑證在傳輸中受到保護。由於您擁有網域，因此您必須建立 DNS 記錄，將流量導向自訂網域的 CloudFront 分佈。

這些元素準備就緒後，您可以透過 Amazon Cognito 主控台或 API 將自訂網域新增至使用者集區。這包括指定網域名稱和 SSL 憑證，然後使用提供的別名目標更新您的 DNS 組態。進行這些變更後，您可以驗證登入頁面是否可在您的自訂網域存取。

建立自訂網域最不費力的方式是使用 Amazon Route 53 中的公有託管區域。Amazon Cognito 主控台只需幾個步驟即可建立正確的 DNS 記錄。開始之前，請考慮為您擁有的網域或子網域[建立 Route 53 託管區域](#)。

主題

- [將自訂網域新增到使用者集區](#)
- [先決條件](#)
- [步驟 1：輸入您的自訂網域名稱](#)
- [步驟 2：新增別名目標和子網域](#)
- [步驟 3：驗證您的登入頁面](#)
- [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。

先決條件

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。建議您的子網域名稱進行身分驗證。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。

- 由美國東部（維吉尼亞北部）的 ACM 管理的公有 SSL/TLS 憑證。憑證必須位於 us-east-1 中，因為憑證將與全球服務 CloudFront 中的分佈相關聯。
- 支援伺服器名稱指示 (SNI) 的瀏覽器用戶端。Amazon Cognito 指派給自訂網域的 CloudFront 分佈需要 SNI。您無法變更此設定。如需 CloudFront 分佈中 SNI 的詳細資訊，請參閱《Amazon CloudFront 開發人員指南》中的[使用 SNI 提供 HTTPS 請求](#)。
- 允許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 會為受管登入頁面設定數個必要的 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，但使用者集區組態的變更可能會導致受管登入 Cookie 的大小增加。自訂網域前面的 Application Load Balancer (ALB) 等中繼服務可能會強制執行最大標頭大小或總 Cookie 大小。如果您的應用程式也設定自己的 Cookie，使用者的工作階段可能會超過這些限制。為了避免大小限制衝突，建議您的應用程式不要在託管使用者集區網域服務的子網域上設定 Cookie。
- 更新 Amazon CloudFront 分佈的許可。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 CloudFront 中動作的詳細資訊，請參閱[將以身分為基礎的政策 \(IAM 政策\) 用於 CloudFront](#)。

Amazon Cognito 一開始會使用您的 IAM 許可來設定 CloudFront 分發，但分發是由 AWS 管理。您無法變更 Amazon Cognito 與您的使用者集區相關聯之 CloudFront 分發的組態。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. 導覽至品牌下的網域選單。
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自訂網域，請選擇刪除自訂網域，然後再建立新的自訂網域。
3. 在網域旁邊，選擇動作，然後選取建立自訂網域。如果您已經設定自訂網域，請選擇刪除自訂網域以刪除現有網域，然後再建立新的自訂網域。
4. 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。
5. 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 憑證才有資格搭配 Amazon Cognito 自訂網域使用，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](#)。

6. 選擇品牌版本。您的品牌版本適用於該網域中的所有使用者互動頁面。您的使用者集區可以託管所有應用程式用戶端的受管登入或託管 UI 品牌。

Note

您可以擁有自訂網域和字首網域，但 Amazon Cognito 僅提供自訂網域的 `/.well-known/openid-configuration` 端點。

7. 選擇 Create (建立)。
8. Amazon Cognito 會將您返回網域功能表。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

下列 [CreateUserPoolDomain](#) 請求內文會建立自訂網域。

```
{
  "Domain": "auth.example.com",
  "UserPoolId": "us-east-1_EXAMPLE",
  "ManagedLoginVersion": 2,
  "CustomDomainConfig": {
    "CertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
  }
}
```

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 中沒有公有託管區域，請使用您自訂網域的父亲系根建立一個託管區域。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》中的[建立公有託管區域](#)。
 - 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，並在父託管區域中設定委派，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父亲系網域，請新增 DNS 記錄。使用下列屬性建立父系網域的 DNS 記錄：
 - 記錄名稱：保留空白。
 - 記錄類型：A。
 - 別名：不啟用。
 - 值：輸入您選擇的目標。此記錄必須解析為某個項目，但記錄的值對 Amazon Cognito 來說並不重要。
 - TTL：設定為您偏好的 TTL 或保留為預設值。
 - 路由政策：選擇簡易路由。
5. 選擇建立記錄。以下是網域 *example.com* 的範例記錄：

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。開始授權 (SOA) 記錄不足以驗證父網域的 DNS 記錄。

6. 為您的自訂網域新增具有下列屬性的另一個 DNS 記錄：

- 記錄名稱：您的自訂網域字首，例如為 `auth` 建立記錄 `auth.example.com`。
- 記錄類型：A。
- 別名：啟用。
- 將流量路由到：選擇別名到 Cloudfront 分佈。輸入您先前記錄的別名目標，例如 `123example.cloudfront.net`。
- 路由政策：選擇簡易路由。

7. 選擇建立記錄。

 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 [GetChange](#) API 方法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，並在 開啟 Amazon Cognito 主控台 <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. 選擇網域功能表。
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。

將品牌套用到受管登入頁面

您可能想要在身分驗證服務和應用程式之間提供一致的使用者體驗。您可以使用 AWS SDK 中的自訂表單和後端 API 操作，或使用受管登入來完成此目標。受管登入和傳統託管 UI 是應用程式元件的 Web 前端，可提供使用者集區的身分驗證。若要將受管身分驗證服務與應用程式 UX 同步，您有兩個自訂選項：品牌編輯器和託管 UI 品牌。您可以在 Amazon Cognito 主控台和使用使用者集區 API 操作中選擇您偏好的體驗。

品牌編輯器

[品牌編輯器](#)是最新使用者集區 UI 體驗、[受管登入](#)的最新自訂選項。品牌編輯器是適用於受管登入資產和樣式的無程式碼視覺化編輯器，以及一組適用於大量組態選項的程式設計組態 API 操作。您使用[網域](#)和受管登入設定的使用者集區會自動轉譯您登入頁面的品牌設計工具版本。

託管 UI（傳統）品牌

[託管 UI（傳統）品牌體驗](#)有兩個選項：使用一組固定樣式選項修改層疊樣式表 (CSS) 檔案，以及新增自訂標誌影像。您可以在 Amazon Cognito 主控台或使用 [SetUICustomization](#) API 操作設定這些選項。在服務啟動時，Amazon Cognito 只有此選項。您使用[網域](#)和託管 UI 品牌版本設定的使用者集區會自動轉譯登入頁面的傳統版本。您的[功能計劃](#)可能也僅支援託管 UI。

Note

品牌編輯器和傳統品牌體驗會修改託管身分驗證服務的視覺化屬性。目前，您無法修改顯示在受管登入頁面上的文字，除了將當地語系化套用至多種語言的其中一種。如需當地語系化的詳細資訊，請參閱 [受管登入當地語系化](#)。

選擇品牌體驗並指派樣式

在 Amazon Cognito 主控台中，新使用者集區預設為受管登入品牌體驗。您在受管登入可用之前設定的使用者集區將具有託管 UI（傳統）品牌。您可以在受管登入和託管 UI 品牌之間切換。當您變更品牌版本時，Amazon Cognito 會立即將變更套用至使用者集區網域的使用者互動頁面。透過受管登入和託管 UI，您的使用者集區可以為每個應用程式用戶端設定樣式。

每個應用程式用戶端可以有不同的品牌樣式，但使用者集區網域提供受管登入或託管 UI。樣式是套用至應用程式用戶端的一組自訂設定。您可以為每個使用者集區設定一個[自訂網域](#)和一個[字首網域](#)。您可以將不同的品牌版本指派給自訂網域和字首網域。不過，當您也有自訂網域時，字首網域無法完全運

作，OIDC .well-known 探索端點只會顯示自訂網域路徑。對於使用此組態的使用者集區中不需要端點探索 (openid-configuration) 的操作，您只能使用字首網域。由於使用者集區的這些屬性，您可以有效地為每個使用者集區選擇一個品牌版本。

您可以在將網域設定為受管登入品牌版本的使用者集區中，將樣式指派給應用程式用戶端。樣式是由影像檔案、顯示選項、CSS 值組成的一組視覺化設定。當您將樣式指派給應用程式用戶端時，Amazon Cognito 會立即將您的更新推送到您的使用者互動登入頁面。Amazon Cognito 會使用您選擇的品牌版本和您已套用的自訂來轉譯使用者互動頁面。

更新和刪除樣式

建立樣式時，您可以將其連結至應用程式用戶端。若要變更應用程式用戶端的樣式指派，您必須先刪除原始樣式。目前，您無法在樣式之間複製設定。您必須以程式設計方式執行此操作。若要在樣式和應用程式用戶端之間複寫設定，請使用 [DescribeManagedLoginBranding](#) API 操作取得樣式的設定，並使用 [CreateManagedLoginBranding](#) 或 [UpdateManagedLoginBranding](#) 套用。您無法變更應用程式用戶端的指派樣式，只能刪除原始樣式並設定新的樣式。如需使用 API 和 SDK 操作管理樣式的詳細資訊，請參閱 [受管登入品牌 API 和 SDK 操作](#)。

Note

建立或更新品牌樣式的程式設計請求，其請求大小不得超過 2 MB。如果您的請求大於此限制，請將您的請求分成多個參數群組 `UpdateManagedLoginBranding` 的請求，但不超過請求大小上限。這些請求不會導致未指定的參數設為預設值，因此您可以傳送部分請求，而不會影響現有的設定。

您可以從受管登入功能表中刪除 Amazon Cognito 主控台內的樣式。在樣式下，選擇您要刪除的樣式，然後選擇刪除樣式。

在高階，將品牌指派給網域的程序包含下列步驟。

1. [建立網域並設定品牌版本](#)。
2. 建立品牌樣式並將其指派給應用程式用戶端。

將樣式指派給應用程式用戶端

1. 在使用者集區的網域功能表中，建立網域並將品牌版本設定為受管登入。
2. 導覽至受管登入選單。在樣式下，選擇建立樣式。

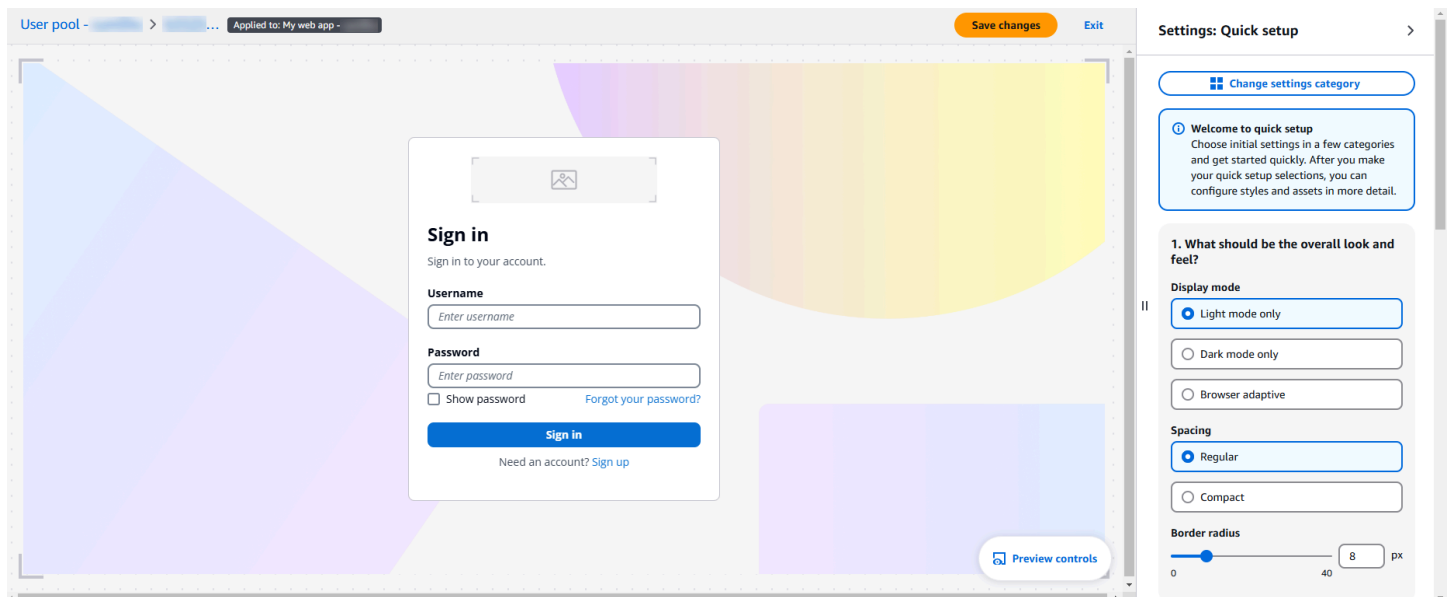
3. 選擇您要為其指派樣式的應用程式用戶端，或建立新的[應用程式用戶端](#)。
4. 若要開始設定您的品牌設定，請選擇啟動品牌編輯器。

主題

- [品牌編輯器和自訂受管登入](#)
- [自訂託管 UI（傳統）品牌](#)

品牌編輯器和自訂受管登入

品牌編輯器是受管登入網頁的視覺化設計和編輯工具。它內建於 Amazon Cognito 主控台。在品牌編輯器中，您可以從登入頁面的預覽開始，並可以進入快速設定選項或使用進階選項的詳細檢視。您可以修改和預覽樣式參數，或新增自訂背景影像和標誌。您可以設定淺色模式和深色模式。



若要開始，請建立您可以套用至使用者集區或應用程式用戶端的樣式。

開始使用品牌編輯器

1. 從[網域索引標籤建立](#)網域，或更新現有的網域。在品牌版本下，將網域設定為使用受管登入。
2. 刪除現有的應用程式用戶端樣式，如果有的話。
 - a. 在應用程式用戶端功能表中，選取您的應用程式用戶端。
 - b. 在受管登入樣式下，選取指派給應用程式用戶端的 style。
 - c. 選擇刪除樣式。確認您的選擇。

- 導覽至使用者集區中的受管登入選單。如果您尚未，請依照提示選取包含受管登入[的功能計劃](#)。如果您想要查看品牌編輯器而不進行變更，也可以選取預覽此功能。
- 在樣式下，選擇建立樣式。
- 選擇您要為其指派樣式的應用程式用戶端，然後選取建立。您也可以建立新的應用程式用戶端。
- Amazon Cognito 主控台會啟動品牌編輯器。
- 選擇您要開始編輯的標籤，或選取啟動編輯器並輸入[快速設定](#)。下列索引標籤可供使用：

預覽版

了解您目前的選擇在受管登入頁面中的外觀。

基礎

設定整體佈景主題、設定外部身分提供者的連結，以及樣式表單欄位。

元件

設定標頭、頁尾和個別 UI 元素的樣式。

- 若要選擇初始設定，請輸入快速設定。選取變更設定類別，然後選擇快速設定。當您選取繼續時，品牌編輯器會啟動一組基本選項供您設定。

文字和當地語系化

您無法在品牌編輯器中修改或當地語系化文字。而是將lang查詢參數新增至您分發給使用者的 URL。此參數可讓您的受管登入頁面當地語系化為多種可用語言之一。如需詳細資訊，請參閱[受管登入當地語系化](#)。

快速設定

啟動品牌編輯器按鈕會為您的受管登入組態載入視覺化編輯器，您可以在其中從各種主要自訂選項中選擇。當您進行選擇時，Amazon Cognito 會在預覽視窗中轉譯您的受管登入變更。若要返回詳細設定選單，請選取變更設定類別按鈕。

整體外觀和感覺應該是什麼？

設定受管登入的基本佈景主題設定。

顯示模式

為您的受管登入選擇淺色模式、深色模式或適應性體驗。當 Amazon Cognito 轉譯受管登入時，適應性設定會延遲到使用者的瀏覽器偏好設定。選擇瀏覽器自適應模式時，您可以選擇不同的顏色和標誌影像，用於淺色和深色模式。

Spacing

設定頁面中元素之間的預設間距。

邊界半徑

設定元素外部邊界的四捨五入深度。

頁面背景看起來如何？

背景類型

顯示影像核取方塊指出您想要背景影像還是設定純色背景顏色。

1. 若要使用影像，請選取顯示影像，然後選擇淺色和深色模式的背景影像。您也可以為影像未涵蓋的背景區域設定深色模式和淺色模式頁面背景顏色。
2. 若要僅對背景使用顏色，請取消選取顯示影像，然後選擇淺色模式和深色模式頁面背景顏色。

表單看起來應該如何？

設定受管登入的形式元素設定。表單項目的範例包括登入和程式碼提示。

水平對齊

設定表單欄位的水平對齊。

表單標誌

設定標誌影像的位置。

標誌影像

選擇要包含在淺色和深色模式表單元素中的標誌影像檔案。若要上傳影像，請選取標誌影像下拉式清單，選擇新增資產，然後新增標誌檔案。

主要品牌顏色

為淺色和深色模式設定佈景主題顏色。此顏色將作為背景顏色套用到分類為主要的所有元素。

標頭看起來應該如何？

選擇是否要在受管登入頁面中包含 標頭。標頭可以包含標誌影像。

標頭標誌

設定 標頭中標誌影像的位置。

標誌影像

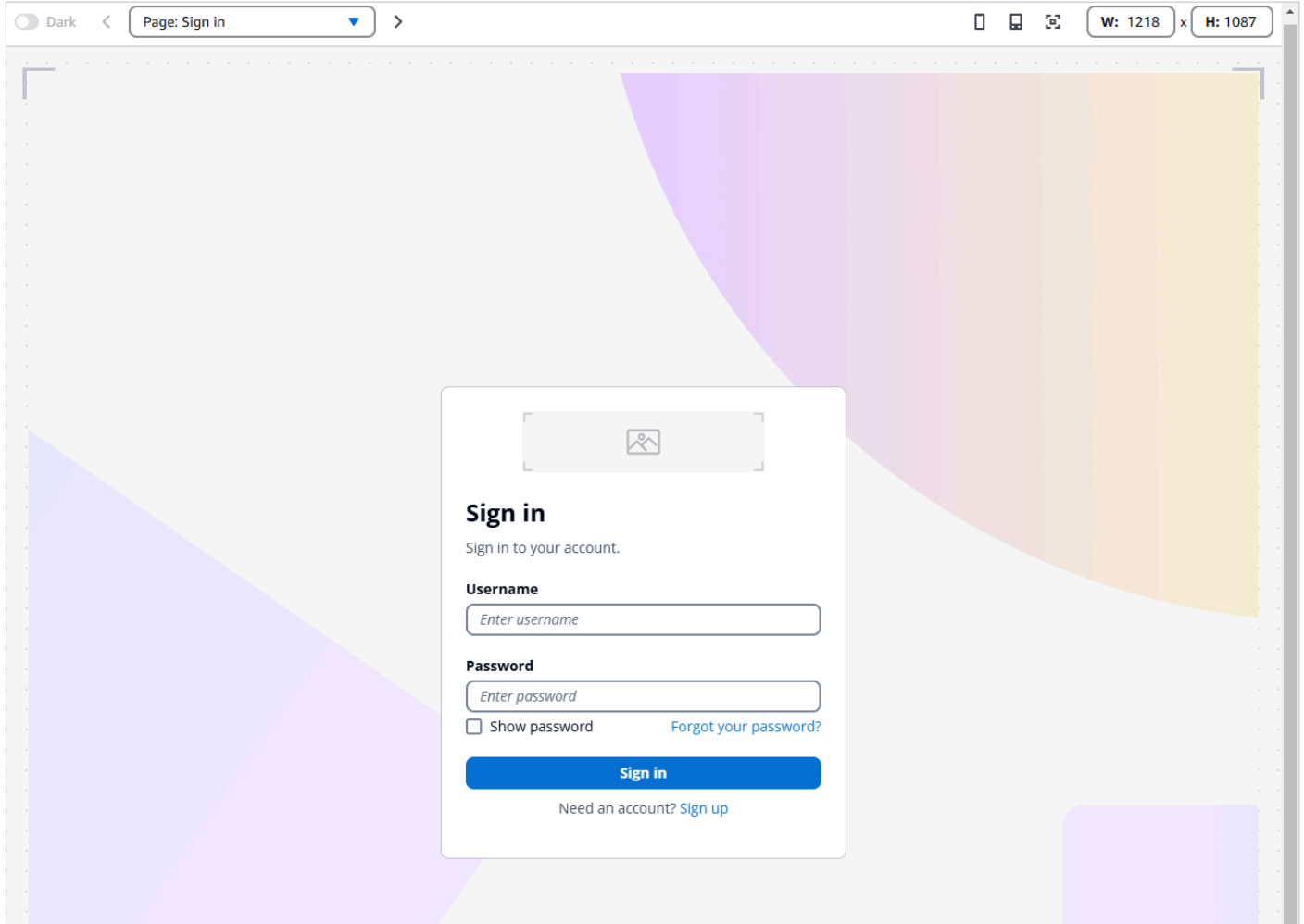
選擇要包含在 標頭中的標誌位置和標誌影像檔案。若要上傳影像，請選取標誌影像下拉式清單，選擇新增資產，然後新增標誌檔案。

標頭背景顏色

設定標頭背景的淺色和深色模式顏色。

詳細設定

在詳細設定檢視中，您可以修改基礎和元件中的個別元件。預覽標籤會顯示目前內容中受管登入的預覽，其中包含您的自訂。

[Amazon Cognito](#) > [User pools](#) > [User pool -](#) > [Managed login](#) > [Style:](#)Style: [Info](#)[Delete style](#)[Launch branding designer](#)**General information** [Info](#)**Assigned app client**
My web app - **Branding customizations**
Cognito default settings**Last customized time**
November 11, 2024 at 11:19 PST[Preview](#)[Foundation](#)[Components](#)

若要輸入元件的視覺化編輯器，請在元件的圖磚中選擇編輯圖示。從佈景主題 Studio 編輯器，您可以使用變更設定類別按鈕在元件之間切換。

基礎

應用程式樣式

設定受管登入組態的基本概念。此類別具有整體佈景主題、文字間距以及頁面標頭和頁尾的設定。

顯示模式

為您的受管登入頁面選擇淺色模式、深色模式或適應性體驗。選擇瀏覽器自適應模式時，您可以選擇不同的顏色和標誌影像，用於淺色和深色模式。

Spacing

設定頁面中元素之間的預設間距。

身分驗證行為

為將使用者連接到外部身分提供者 (IdPs) 按鈕設定樣式。本節包含網域搜尋輸入選項，讓受管登入提示使用者取得電子郵件地址，並將其與其 [SAML 身分提供者識別符](#) 進行比對。

表單行為

設定輸入表單的樣式：輸入位置、顏色和元素對齊。

元件

按鈕

Amazon Cognito 在受管登入頁面上呈現的按鈕樣式。

分隔符號

受管登入元素之間分隔線和邊界的樣式，例如輸入表單和外部供應商登入選擇器。

下拉式清單

下拉式功能表的樣式。

網頁圖示

Amazon Cognito 為標籤和書籤圖示提供的影像樣式。

焦點環

表示目前選取輸入之醒目提示的樣式。

表單容器

繫結表單之元素的樣式。

全域頁尾

Amazon Cognito 在受管登入頁面底部顯示的頁尾樣式。

全域標頭

Amazon Cognito 顯示在受管登入頁面頂端的標頭樣式。

指示

錯誤和成功訊息的樣式。

選項控制

核取方塊、多選和其他輸入提示的樣式。

頁面背景

受管登入整體背景的樣式。

輸入

表單欄位輸入提示的樣式。

連結

受管登入頁面中超連結的樣式。

頁面的文字

頁面內文字的樣式。

欄位的文字

表單輸入周圍文字的樣式。

受管登入品牌 API 和 SDK 操作

您也可以使用 API 操作 [CreateManagedLoginBranding](#) 和 [UpdateManagedLoginBranding](#)，將品牌套用到受管登入樣式。這些操作非常適合為另一個應用程式用戶端或使用者集區建立相同或稍微修改的品牌樣式版本。使用 API 操作 [DescribeManagedLoginBranding](#) 查詢現有樣式的受管登入品牌，然後視需要修改輸出並將其套用至其他資源。

UpdateManagedLoginBranding 操作不會變更套用您樣式的應用程式用戶端。它只會更新指派給應用程式用戶端的現有樣式。若要完全取代應用程式用戶端的樣式，請使用 [DeleteManagedLoginBranding](#) 刪除現有樣式，並使用 指派新樣式CreateManagedLoginBranding。在 Amazon Cognito 主控台中，也是如此：您必須刪除現有樣式並建立新的樣式。

在 API 或 SDK 請求中設定受管登入品牌，需要將您的設定內嵌在轉換為Document資料類型的 JSON 檔案中。以下是您可以新增的影像和產生程式設計請求以設定品牌樣式的指引。

影像資產

[CreateManagedLoginBranding](#) 和 [UpdateManagedLoginBranding](#) 包含 Assets 參數。此參數是 base64 編碼二進位格式的影像檔案陣列。

Note

建立或更新品牌樣式的程式設計請求，其請求大小不得超過 2 MB。請求中的資產可能會超過此限制。如果是這種情況，請將您的請求分成多個UpdateManagedLoginBranding請求，這些請求的參數群組不超過請求大小上限。這些請求不會導致未指定的參數設為預設值，因此您可以傳送部分請求，而不會影響現有的設定。

有些資產對您可以提交的檔案類型有限制。

資產	已接受的副檔名
FAVICON_ICO	ico
FAVICON_SVG	svg
EMAIL_GRAPHIC	png、svg、jpeg
SMS_GRAPHIC	png、svg、jpeg
AUTH_APP_GRAPHIC	png、svg、jpeg
PASSWORD_GRAPHIC	png、svg、jpeg
PASSKEY_GRAPHIC	png、svg、jpeg

資產	已接受的副檔名
PAGE_HEADER_LOGO	png、svg、jpeg
PAGE_HEADER_BACKGROUND	png、svg、jpeg
PAGE_FOOTER_LOGO	png、svg、jpeg
PAGE_FOOTER_BACKGROUND	png、svg、jpeg
PAGE_BACKGROUND	png、svg、jpeg
FORM_BACKGROUND	png、svg、jpeg
FORM_LOGO	png、svg、jpeg
IDP_BUTTON_ICON	ico、svg

SVG 類型的檔案支援下列屬性和元素。

Attributes

accent-height, accumulate, additive, alignment-baseline, ascent, attributename, attributetype, azimuth, basefrequency, baseline-shift, begin, bias, by, class, clip, clip-path, clip-rule, color, color-interpolation, color-interpolation-filters, color-profile, color-rendering, cx, cy, d, dx, dy, diffuseconstant, direction, display, divisor, dur, edgemode, elevation, end, fill, fill-opacity, fill-rule, filter, filterunits, flood-color, flood-opacity, font-family, font-size, font-size-adjust, font-stretch, font-style, font-variant, font-weight, fx, fy, g1, g2, glyph-name, glyphref, gradientunits, gradienttransform, height, href, id, image-rendering, in, in2, k, k1, k2, k3, k4, kerning, keypoints, keysplines, keytimes, lang, lengthadjust, letter-spacing, kernelmatrix, kernelunitlength, lighting-color, local, marker-end, marker-mid, marker-start, markerheight, markerunits, markerwidth, maskcontentunits, maskunits, max, mask, media, method, mode, min, name, numoctaves, offset, operator, opacity, order, orient, orientation, origin, overflow, paint-order, path, pathlength, patterncontentunits, patterntransform, patternunits, points, preservealpha, preserveaspectratio, r, rx, ry, radius, reffx, reffy, repeatcount, repeatdur, restart, result, rotate, scale, seed, shape-rendering, specularconstant, specularexponent, spreadmethod, stddeviation, stitchtiles, stop-color, stop-opacity, stroke-dasharray, stroke-dashoffset, stroke-linecap, stroke-linejoin, stroke-miterlimit, stroke-opacity, stroke, stroke-width, style, surfacscale, tabindex, targetx, targety, transform,

```
text-anchor, text-decoration, text-rendering, textlength, type, u1, u2, unicode,
values, viewBox, visibility, vert-adv-y, vert-origin-x, vert-origin-y, width, word-
spacing, wrap, writing-mode, xchannelselector, ychannelselector, x, x1, x2, xmlns,
y, y1, y2, z, zoomandpan
```

Elements

```
svg, a, altglyph, altglyphdef, altglyphitem, animatecolor, animatemotion,
animatetransform, audio, canvas, circle, clippath, defs, desc, ellipse, filter,
font, g, glyph, glyphref, hkern, image, line, lineargradient, marker, mask,
metadata, mpath, path, pattern, polygon, polyline, radialgradient, rect, stop,
style, switch, symbol, text, textpath, title, tref, tspan, video, view, vkern,
feBlend, feColorMatrix, feComponentTransfer, feComposite, feConvolveMatrix,
feDiffuseLighting, feDisplacementMap, feDistantLight, feFlood, feFuncA, feFuncB,
feFuncG, feFuncR, feGaussianBlur, feMerge, feMergeNode, feMorphology, feOffset,
fePointLight, feSpecularLighting, feSpotLight, feTile, feTurbulence
```

受管登入品牌操作的工具

Amazon Cognito 會管理受管登入品牌設定物件的 [JSON-Schema 格式](#) 檔案。以下是如何以程式設計方式更新您的品牌樣式。

在使用者集區 API 中更新品牌

1. 在 Amazon Cognito 主控台中，從使用者集區的受管登入功能表中建立預設的受管登入品牌樣式。將其指派給應用程式用戶端。
2. 記錄您為其建立樣式的應用程式用戶端 ID，例如 1example23456789。
3. 使用 [DescribeManagedLoginBrandingByClient](#) API 請求擷取品牌樣式的設定，並將 ReturnMergedResources 設定為 true。以下是請求內文的範例。

```
{
  "ClientId": "1example23456789",
  "ReturnMergedResources": true,
  "UserPoolId": "us-east-1_EXAMPLE"
}
```

4. DescribeManagedLoginBrandingByClient 使用自訂修改 的輸出。
 - a. 回應內文會包裝在不屬於建立和更新操作語法的 ManagedLoginBranding 元素中。移除 JSON 物件的這個頂層。

- b. 若要取代映像，請將 Bytes 值取代為每個映像檔案的 Base64-encoded 二進位資料。
 - c. 若要更新設定，請修改 Settings 物件的輸出，並將其包含在下一個請求中。Amazon Cognito 會忽略 Settings 物件中任何不在 API 回應中收到結構描述中的值。
5. 在 [CreateManagedLoginBranding](#) 或 [UpdateManagedLoginBranding](#) 請求中使用更新的回應內文。如果此請求的大小超過 2 MB，請將它分成多個請求。這些操作適用於原始設定保持不變的 PATCH 模型，除非您另有指定。

自訂託管 UI（傳統）品牌

您可以使用 AWS Management Console 或 AWS CLI 或 API，為託管 UI 指定傳統自訂設定。您可以上傳要顯示在應用程式中的自訂標誌影像。您也可以將一些層疊樣式表 (CSS) 選項套用至 UI 的外觀和風格。

您可以自訂 UI 預設值，並使用特定設定覆寫個別[應用程式用戶端](#)。Amazon Cognito 會將預設組態套用至每個沒有用戶端層級設定的應用程式用戶端。

在 Amazon Cognito 主控台和 API 請求中，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 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。

在傳統品牌中指定自訂標誌

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 [SetUICustomization](#) ImageFile 中設定，請將檔案轉換為 Base64-encoded 的文字字串，或在 中提供檔案路徑 AWS CLI，並讓 Amazon Cognito 為您編碼。

在傳統品牌中指定 CSS 自訂

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。
- color 是文字顏色，必須是[合法的 CSS 顏色值](#)。

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。
- color 是文字顏色，必須是[合法的 CSS 顏色值](#)。

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。
- color 是 IdP 區段標頭的文字顏色，例如使用您的公司 ID 登入。必須是[合法的 CSS 顏色值](#)。

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。
- background-color 是具有透明區段之日誌的背景顏色。必須是[合法的 CSS 顏色值](#)。

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。

在 中使用傳統品牌自訂託管 UI AWS Management Console

您可以使用 AWS Management Console 為您的應用程式指定 UI 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_callback>`
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. 從[網域索引標籤建立網域](#)，或更新現有的網域。在品牌版本下，將網域設定為使用託管 UI (傳統)。
4. 選擇受管登入功能表。
5. 若要為所有應用程式用戶端自訂 UI 設定，請在託管 UI 設定下尋找樣式，然後選取編輯。
6. 若要自訂一個應用程式用戶端的 UI 設定，請前往應用程式用戶端功能表，然後選取您要修改的應用程式用戶端，然後尋找託管 UI (傳統) 樣式，然後選取覆寫。選擇 Edit (編輯)。
7. 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)。
8. 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。

在使用者集區 API 和 中使用傳統品牌自訂託管 UI AWS CLI

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 從映像檔案：`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 二進位文字：`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 Lambda 觸發程序來自訂使用者集區工作流程

Amazon Cognito 使用 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。您的函數可以修改身分驗證流程的預設行為、提出 API 請求來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	作業	描述
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

使用者集區流程	作業	描述
	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”	自訂驗證接受或拒絕登入請求
	the section called “身分驗證後”	記錄事件以進行自訂分析
	the section called “產生權杖前”	增加或抑制權杖宣告
註冊	the section called “註冊前”	執行用於接受或拒絕註冊請求的自訂驗證
	the section called “確認後”	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”	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”	執行訊息的進階自訂及當地語系化
權杖建立	the section called “產生權杖前”	在 ID 和存取權杖中新增或移除屬性
電子郵件和簡訊第三方供應商	the section called “自訂寄件者”	透過第三方供應商傳送簡訊和電子郵件

主題

- [Lambda 觸發程序的須知事項](#)
- [新增使用者集區 Lambda 觸發條件](#)
- [使用者集區 Lambda 觸發程序事件](#)
- [使用者集區 Lambda 觸發程序的常用參數](#)
- [用戶端中繼資料](#)
- [將 API 操作連線至 Lambda 觸發程序](#)
- [將 Lambda 觸發程序連接至使用者集區功能操作](#)

- [註冊前 Lambda 觸發程序](#)
- [確認後 Lambda 觸發程序](#)
- [身分驗證前 Lambda 觸發程序](#)
- [身分驗證後 Lambda 觸發程序](#)
- [自訂身分驗證挑戰 Lambda 觸發程序](#)
- [產生權杖前 Lambda 觸發程序](#)
- [遷移使用者 Lambda 觸發程序](#)
- [自訂訊息 Lambda 觸發程序](#)
- [自訂寄件者 Lambda 觸發程序](#)

Lambda 觸發程序的須知事項

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](#)。若要在中自訂存取權杖[產生權杖前 Lambda 觸發程序](#)，您必須使用 Lite 以外的功能計劃來設定使用者集區，並更新 Lambda 觸發組態以使用事件版本 2。
- 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[錯誤處理和自動重試 AWS Lambda](#)。

- 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。

受管登入會傳回 Lambda 觸發程序在登入提示上方產生為錯誤文字的錯誤。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 中新增 Lambda 函數，AWS 帳戶 做為使用者集區的觸發條件。您必須使用 [CreateUserPool](#) 和 [UpdateUserPool](#) API 操作，或其在 AWS CloudFormation 和 中的對等項目來新增跨帳戶觸發 AWS CLI。您無法在 中新增跨帳戶函數 AWS Management Console。
- 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 觸發條件

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. 選擇延伸項目功能表並尋找 Lambda 觸發條件。
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. 選取指派 Lambda 函數，然後在與您的使用者集區 AWS 區域 相同的 中選取函數。

Note

如果您的 AWS Identity and Access Management (IAM) 登入資料具有更新 Lambda 函數的許可，Amazon Cognito 會新增 Lambda 資源型政策。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。如需詳細資訊，請參閱[the section called “須知事項”](#)。

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。如果您的函數在不修改的情況下傳回輸入事

件，Amazon Cognito 會繼續進行預設行為。以下顯示所有 Lambda 觸發輸入事件常見的參數。如需觸發條件特定的事件語法，請檢閱本指南 區段中每個觸發條件的事件結構描述。

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 做為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。

用戶端中繼資料

您可以在 API 操作和[權杖端點](#)請求中將自訂參數提交至 Lambda 觸發函數。使用用戶端中繼資料，您的應用程式可以收集請求來源環境的其他資訊。當您將用戶端中繼資料傳遞至 Lambda 函數時，他們可以處理其他資料，並在記錄或自訂身分驗證流程時使用它。用戶端中繼資料是您選擇的字串對，並以 JSON 鍵值格式設計。

用戶端中繼資料範例使用案例

- 在註冊時將地理位置資料傳遞至[註冊前觸發](#)條件，並防止從不需要的位置登入。
- 將租戶 ID 資料傳遞至[自訂挑戰觸發條件](#)，並向來自不同業務單位的客戶發出不同的挑戰。
- 將使用者的字符傳遞至[字符產生前觸發](#)程序，並產生代表 M2M 請求的委託人日誌。如需請求範例，請參閱[具有基本授權的用戶端憑證](#)。

以下是將用戶端中繼資料傳遞至註冊前觸發程序的範例。

SignUp request

以下是範例 [SignUp](#) 請求，其中包含 Amazon Cognito 傳遞給註冊前觸發條件的用戶端中繼資料。

```
POST HTTP/1.1
Host: cognito-idp.us-east-1.amazonaws.com
X-Amz-Date: 20230613T200059Z
Accept-Encoding: gzip, deflate, br
X-Amz-Target: AWSCognitoIdentityProviderService.SignUp
User-Agent: <UserAgentString>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
  Signature=<Signature>
Content-Length: <PayloadSizeBytes>

{
  "ClientId": "1example23456789",
  "Username": "mary_major",
  "Password": "<Password>",
  "SecretHash": "<Secret hash>",
  "ClientMetadata": {
    "IpAddress" : "192.0.2.252",
    "GeoLocation" : "Netherlands (Kingdom of the) [NL]"
  }
  "UserAttributes": [
    {
      "Name": "name",
      "Value": "Mary"
    },
    {
      "Name": "email",
      "Value": "mary_major@example.com"
    },
    {
      "Name": "phone_number",
      "Value": "+12065551212"
    }
  ],
}
```

Lambda trigger input event

請求會在您的註冊前函數中產生下列請求內文。

```
{
  "callerContext": {
    "awsSdkVersion": "aws-sdk-unknown-unknown",
    "clientId": "1example23456789"
  },
  "region": "us-west-2",
  "request": {
    "clientMetadata": {
      "GeoLocation": "Netherlands (Kingdom of the) [NL]",
      "IpAddress": "192.0.2.252"
    },
    "userAttributes": {
      "email": "mary_major@example.com",
      "name": "Mary",
      "phone_number": "+12065551212"
    },
    "validationData": null
  },
  "response": {
    "autoConfirmUser": false,
    "autoVerifyEmail": false,
    "autoVerifyPhone": false
  },
  "triggerSource": "PreSignUp_SignUp",
  "userName": "mary_major2",
  "userPoolId": "us-west-2_EXAMPLE",
  "version": "1"
}
```

machine-to-machine(M2M) 用戶端登入資料的用戶端中繼資料

您可以在 M2M 請求中傳遞[用戶端中繼資料](#)。用戶端中繼資料是來自使用者或應用程式環境的其他資訊，有助於的結果[產生權杖前 Lambda 觸發程序](#)。在具有使用者主體的身分驗證操作中，您可以將用戶端中繼資料傳遞至 [AdminRespondToAuthChallenge](#) 和 [RespondToAuthChallenge](#) API 請求內文中的權杖產生前觸發器。由於應用程式會執行流程來產生 M2M 的存取字符，並對 提出直接請求[權杖端點](#)，因此具有不同的模型。在用戶端登入資料的字符請求 POST 內文中，將具有用戶端中繼資料物件 URL 編碼 (x-www-form-urlencoded) 的 `aws_client_metadata` 參數傳遞至字串。如需請求範例，請參閱 [具有基本授權的用戶端憑證](#)。以下是傳遞鍵值對 的範例參數 `{"environment": "dev", "language": "en-US"}`。

```
aws_client_metadata=%7B%22environment%22%3A%20%22dev%22,%20%22language%22%3A%20%22en-US%22%7D
```

臨時使用者屬性：**validationData**

有些身分驗證操作也有 validationData 參數。如同用戶端中繼資料，這是將 Amazon Cognito 不會自動收集的外部資訊傳遞至 Lambda 觸發程序的機會。驗證資料欄位旨在為您的 Lambda 函數提供註冊和登入操作中的其他使用者內容。[SignUp](#) 和 [AdminCreateUser](#) 會傳遞 validationData 至 [註冊前觸發](#) 條件。[InitiateAuth](#) 和 [AdminInitiateAuth](#) ClientMetadata 會將輸入事件 validationData 中的 API 請求內文傳遞至身分 [驗證前](#) 和 [遷移使用者](#) 觸發。

若要將 API 操作映射至它們可以傳遞用戶端中繼資料的函數，請參閱以下觸發來源區段。

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、受管登入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 [了解 API、OIDC 和受管登入頁面身分驗證](#)。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](#)
- [受管登入中 Amazon Cognito 本機使用者的 Lambda 觸發條件](#)
- [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

API 操作	Lambda 觸發程序	觸發程序來源
	產生權杖前	TokenGeneration_NewPasswordChallenge
	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 CustomEmailSender_Authentication
	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

API 操作	Lambda 觸發程序	觸發程序來源
ConfirmForgotPassword	確認後	PostConfirmation_ConfirmForgotPassword
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
GetTokensFromRefreshToken	產生權杖前	TokenGeneration_Authentication

受管登入中 Amazon Cognito 本機使用者的 Lambda 觸發條件

下表說明當本機使用者使用受管登入登入使用者集區時，Amazon Cognito 可以叫用之 Lambda 觸發條件的來源字串。

受管登入中的本機使用者觸發來源

受管登入 URI	Lambda 觸發程序	觸發程序來源
/signup	註冊前	PreSignUp_SignUp
	自訂訊息	CustomMessage_SignUp

受管登入 URI	Lambda 觸發程序	觸發程序來源
	自訂電子郵件寄件者	CustomEmailSender_SignUp
	自訂簡訊寄件者	CustomSMSSender_SignUp
	確認後	PostConfirmation_ConfirmSignUp
/login	身分驗證前	PreAuthentication_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
		CustomEmailSender_Authentication
	自訂簡訊寄件者	CustomSMSSender_Authentication

受管登入 URI	Lambda 觸發程序	觸發程序來源
/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

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受管登入來登入，或者您可以對產生請求[授權端點](#)，以無提示方式將他們重新導向至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

登入事件	Lambda 觸發程序	觸發程序來源
	產生權杖前	TokenGeneration_HostedAuth
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。

觸發條件	triggerSource 值	事件
確認後	PostConfirmation_ConfirmSignUp	註冊後確認。
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 會從受管登入頁面驗證使用者。
產生權杖前	TokenGeneration_Authentication	使用者身分驗證或權杖重新整理完成。

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。

自訂寄件者觸發條件

觸發條件	triggerSource 值	事件
自訂寄件者	CustomEmailSender_SignUp CustomSmsSender_SignUp	當使用者在您的使用者集區中註冊時。
自訂寄件者	CustomEmailSender_AdminCreateUser CustomSmsSender_AdminCreateUser	當您以管理員身分建立使用者時，Amazon Cognito 會傳送臨時密碼給他們。
自訂寄件者	CustomEmailSender_ForgotPassword	當您的使用者請求重設密碼時。

觸發條件	triggerSource 值	事件
	CustomSmsSender_ForgotPassword	
自訂寄件者	CustomEmailSender_UpdateUserAttribute CustomSmsSender_UpdateUserAttribute	當使用者變更其電子郵件地址或電話號碼，且 Amazon Cognito 傳送驗證碼時。
自訂寄件者	CustomEmailSender_VerifyUserAttribute CustomSmsSender_VerifyUserAttribute	當使用者新增電子郵件地址或電話號碼，且 Amazon Cognito 傳送驗證碼時。
自訂寄件者	CustomEmailSender_Authentication CustomSmsSender_Authentication	已設定 SMS 或電子郵件 MFA 或 OTP 的使用者登入時。
自訂寄件者	CustomEmailSender_AccountTakeOverNotification	當您的威脅防護設定對使用者的登入嘗試採取自動動作，且風險層級的動作包含通知。

註冊前 Lambda 觸發程序

您可能想要在具有自助註冊選項的使用者集區中自訂註冊程序。預先註冊觸發程序的一些常見用途是執行新使用者的自訂分析和記錄、套用安全性和控管標準，或將第三方 IdP 中的使用者連結至[合併的使用者設定檔](#)。您可能也有信任的使用者，他們不需要進行[驗證和確認](#)。

在 Amazon Cognito 完成建立新的[本機](#)或[聯合身分](#)使用者之前，它會啟用註冊前 Lambda 函數。您的使用者集區會在使用 [SignUp](#) 進行自助註冊時叫用此觸發，或使用信任的[身分提供者](#)首次登入，以及使用 [AdminCreateUser](#) 建立使用者時叫用此觸發。在註冊過程中，您可以使用此函數來分析具有自訂邏輯的登入事件，以及修改或拒絕新使用者。

主題

- [註冊前 Lambda 觸發程序參數](#)
- [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 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。在 [AdminCreateUser](#) 或 [SignUp](#) API 請求的 ValidationData 參數中，將此資訊傳送至 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 觸發程式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，將此資料傳遞至您的 Lambda 函數：[AdminCreateUser](#)、[AdminRespondToAuthChallenge](#)、[ForgotPassword](#) 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](#)。

註冊前範例：自動確認來自己註冊網域的使用者

這是 Lambda 觸發程式碼範例。在 Amazon Cognito 處理註冊請求之前，會立即叫用註冊前觸發程序。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

```
export const handler = async (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。若要防止發生這種情況，您可以使用這些使用者集區 [ListUsers API](#)，查看是否有現有的使用者已經使用新使用者的電話號碼或電子郵件地址做為別名。

Node.js

```
exports.handler = (event, context, callback) => {
  // 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 to Amazon Cognito
  callback(null, event);
};
```

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

```
export const 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。您的使用者集區會在 [ConfirmSignUp](#)、[AdminConfirmSignUp](#) 和 [ConfirmForgotPassword](#) 上叫用您的後置確認函數。當使用者在[受管登入](#)中確認註冊或密碼重設時，也會執行此觸發。

主題

- [確認後 Lambda 觸發程序參數](#)
- [確認後範例](#)

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。

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

除非使用者集區應用程式用戶端 `PreventUserExistenceErrors` 的設定設為 `ENABLED`，否則當使用者不存在時，此 Lambda 觸發不會啟用 `ENABLED`。現有身分驗證工作階段的續約也不會啟用此觸發。

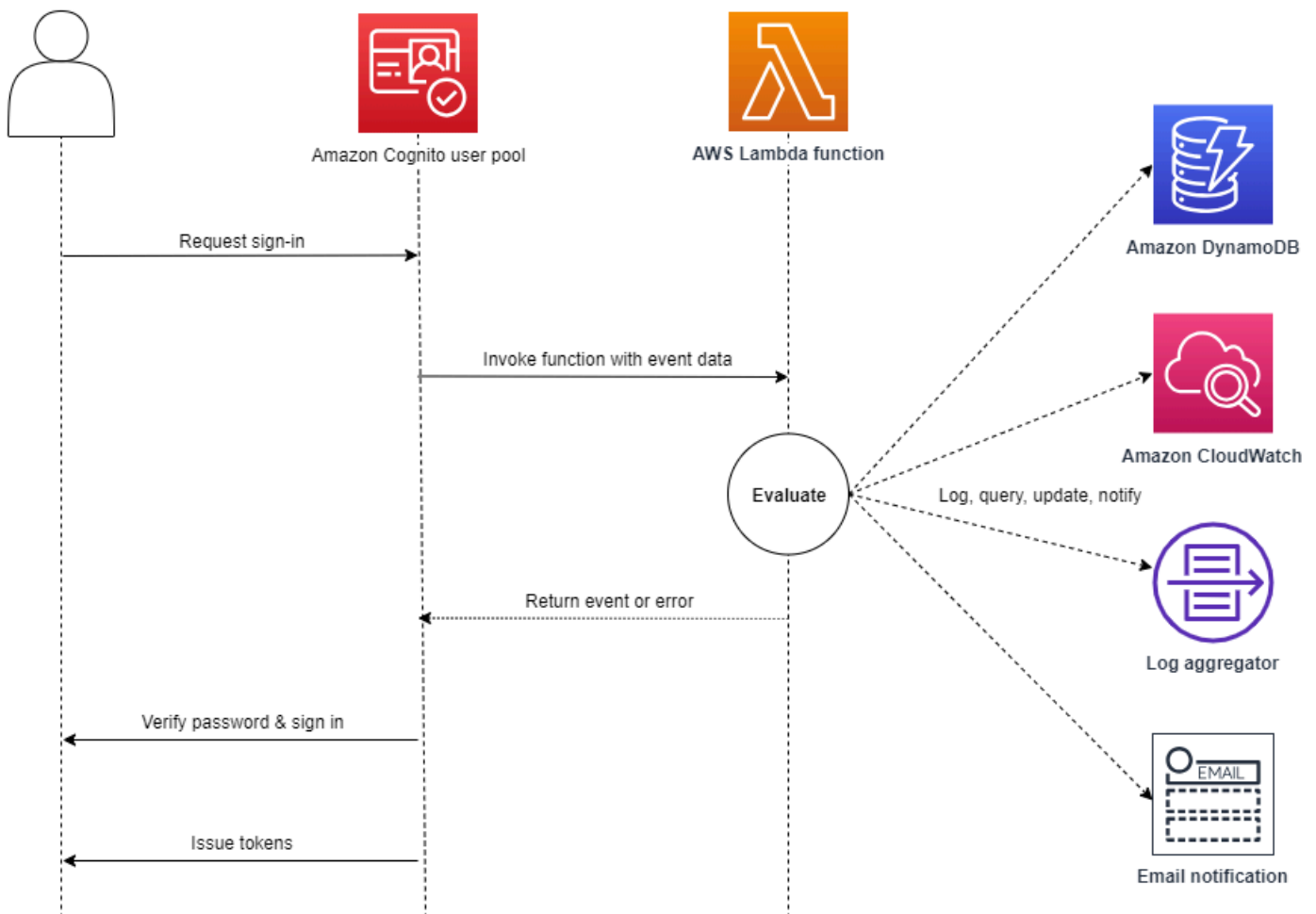
主題

- [流程概觀](#)
- [身分驗證前 Lambda 觸發程序參數](#)
- [身分驗證前範例](#)

流程概觀

Amazon Cognito pre authentication trigger

Evaluate and authorize user sign-in



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。

如需詳細資訊，請參閱[範例身分驗證工作階段](#)。

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 不會處理函數在回應中傳回的任何新增資訊。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

身分驗證後觸發不會變更使用者的身分驗證流程。Amazon Cognito 會在身分驗證完成後，在使用者收到字符之前叫用此 Lambda。當您想要新增身分驗證事件的自訂後製處理時，請新增身分驗證後觸發條件，例如記錄日誌或下次登入時將反映的使用者設定檔調整。

未將請求內文傳回 Amazon Cognito 的身分驗證後 Lambda 仍可能導致身分驗證無法完成。如需詳細資訊，請參閱[Lambda 觸發程序的須知事項](#)。

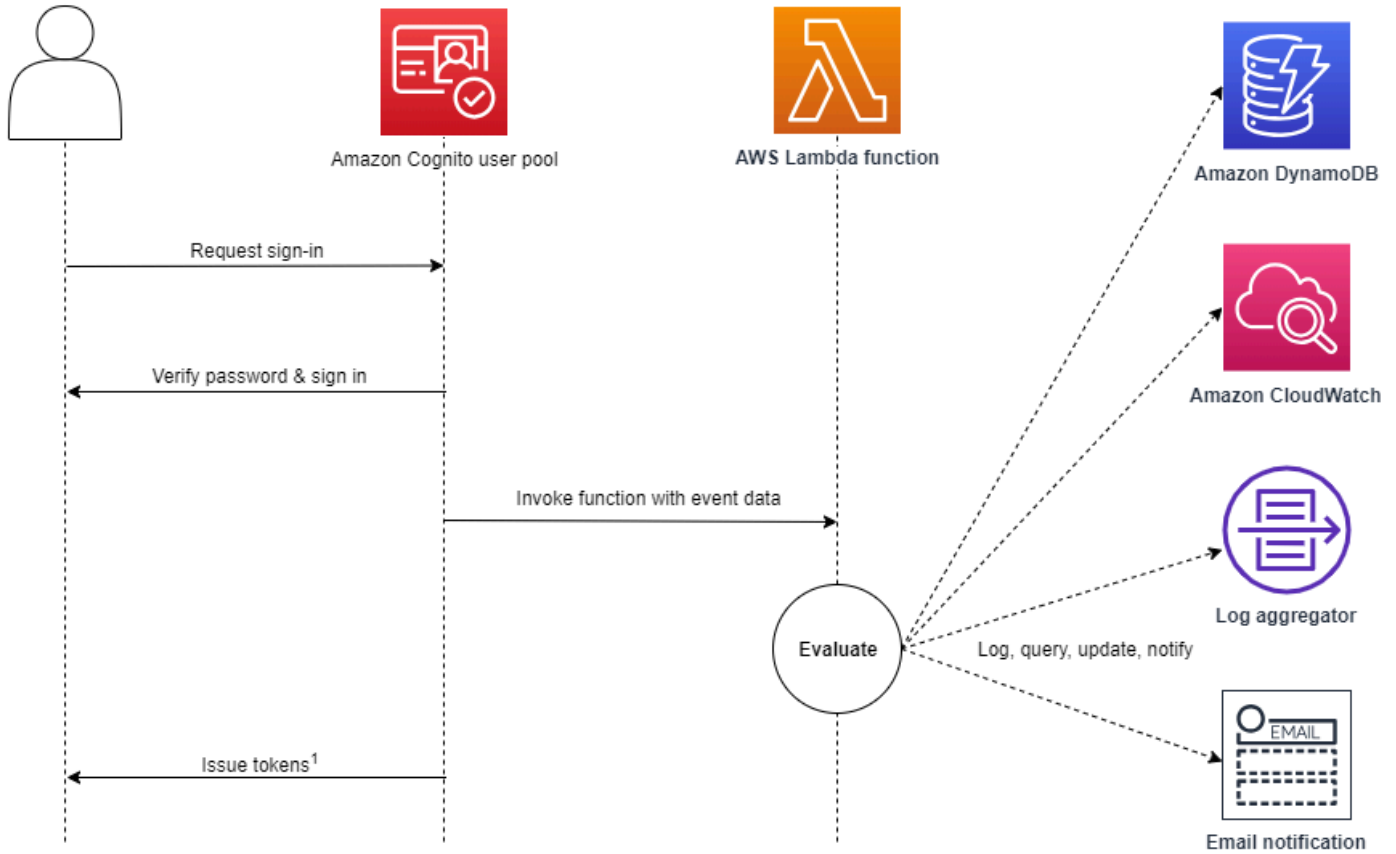
主題

- [身分驗證流程概觀](#)
- [身分驗證後 Lambda 觸發程序參數](#)
- [身分驗證後範例](#)

身分驗證流程概觀

Amazon Cognito post authentication trigger

Report sign-in results



¹ This trigger doesn't have any effect on sign-in outcomes or token contents.

如需詳細資訊，請參閱[範例身分驗證工作階段](#)。

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

當您為 Amazon Cognito 使用者集區建置身分驗證流程時，您可能會發現您想要將身分驗證模型延伸到內建流程之外。自訂挑戰觸發程序的常見使用案例之一是實作使用者名稱、密碼和多重要素驗證 (MFA) 以外的其他安全檢查。自訂挑戰是您可以使用 Lambda 支援的程式設計語言產生的任何問題和回應。例如，您可能想要要求使用者解決 CAPTCHA 或回答安全問題，才能進行身分驗證。另一個潛在需求是整合特殊身分驗證因素或裝置。或者，您可能已經開發了使用硬體安全金鑰或生物識別裝置來驗證使用者身分的軟體。自訂挑戰的身分驗證成功定義是 Lambda 函數接受為正確的任何答案：固定字串，例如外部 API 的滿意回應。

您可以使用自訂挑戰開始身分驗證，並完全控制身分驗證程序，或者您可以在應用程式收到自訂挑戰之前執行使用者名稱密碼身分驗證。

自訂身分驗證挑戰 Lambda 觸發條件：

定義

啟動挑戰序列。決定您是否要啟動新的挑戰、將身分驗證標記為完成，或停止身分驗證嘗試。

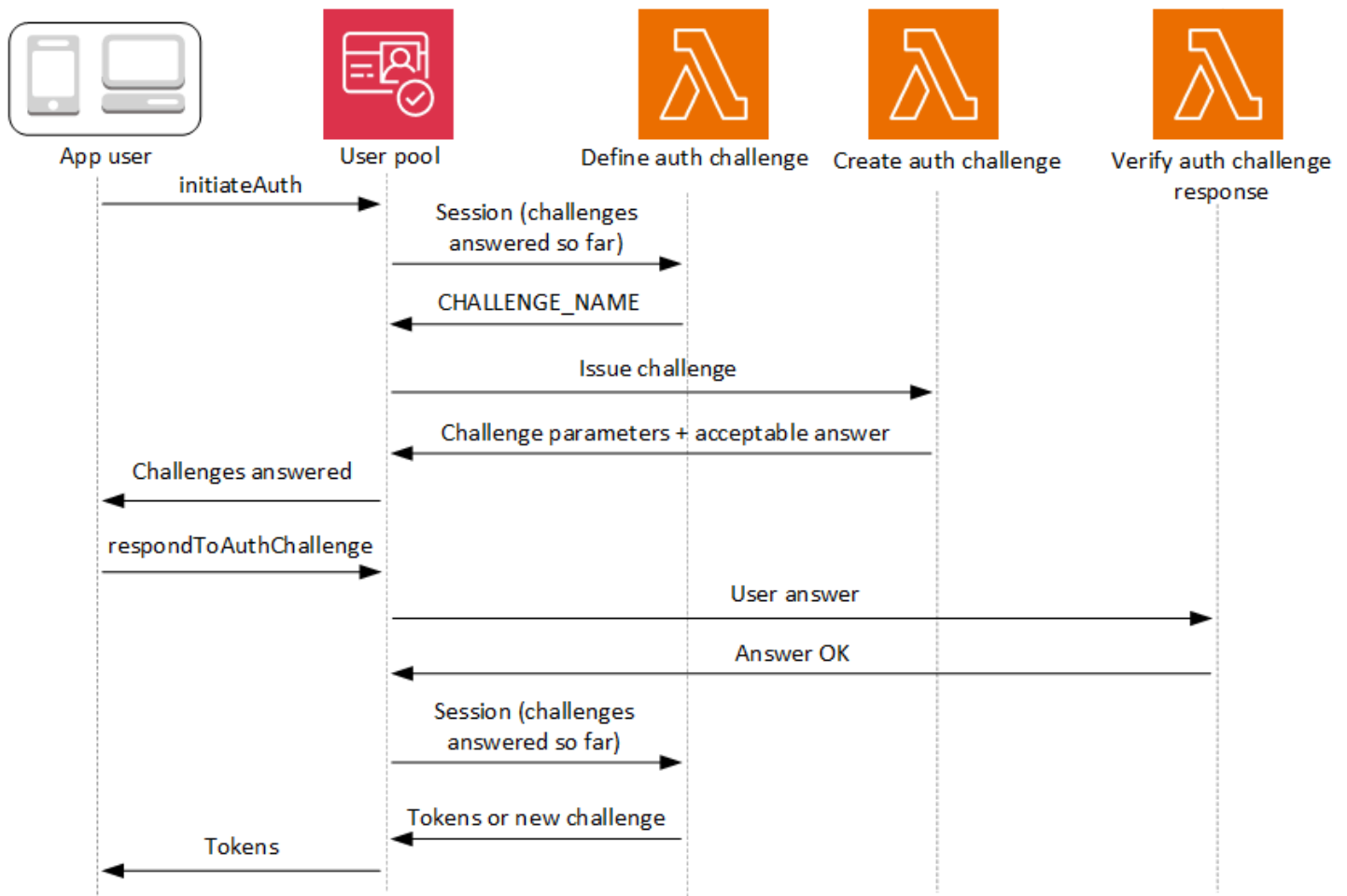
建立

向應用程式發出使用者必須回答的問題。此函數可能會顯示安全問題或 CAPTCHA 的連結，您的應用程式應該向使用者顯示。

驗證

知道預期的回答，並將其與應用程式在挑戰回應中提供的答案進行比較。函數可能會呼叫 CAPTCHA 服務的 API，以擷取使用者嘗試解決方案的預期結果。

這三個 Lambda 函數會串連在一起，以呈現完全在您的控制範圍內且屬於您自己的設計的身分驗證機制。由於自訂身分驗證需要用戶端和 Lambda 函數中的應用程式邏輯，因此您無法在受管登入中處理自訂身分驗證。此身分驗證系統需要額外的開發人員工作。您的應用程式必須使用使用者集區 API 執行身分驗證流程，並使用自訂建立的登入界面來處理產生的挑戰，該界面會在自訂身分驗證挑戰的中心呈現問題。



如需實作自訂身分驗證的詳細資訊，請參閱 [自訂身分驗證流程與挑戰](#)

API 操作 [InitiateAuth](#) 或 [AdminInitiateAuth](#) 與 [RespondToAuthChallenge](#) 或

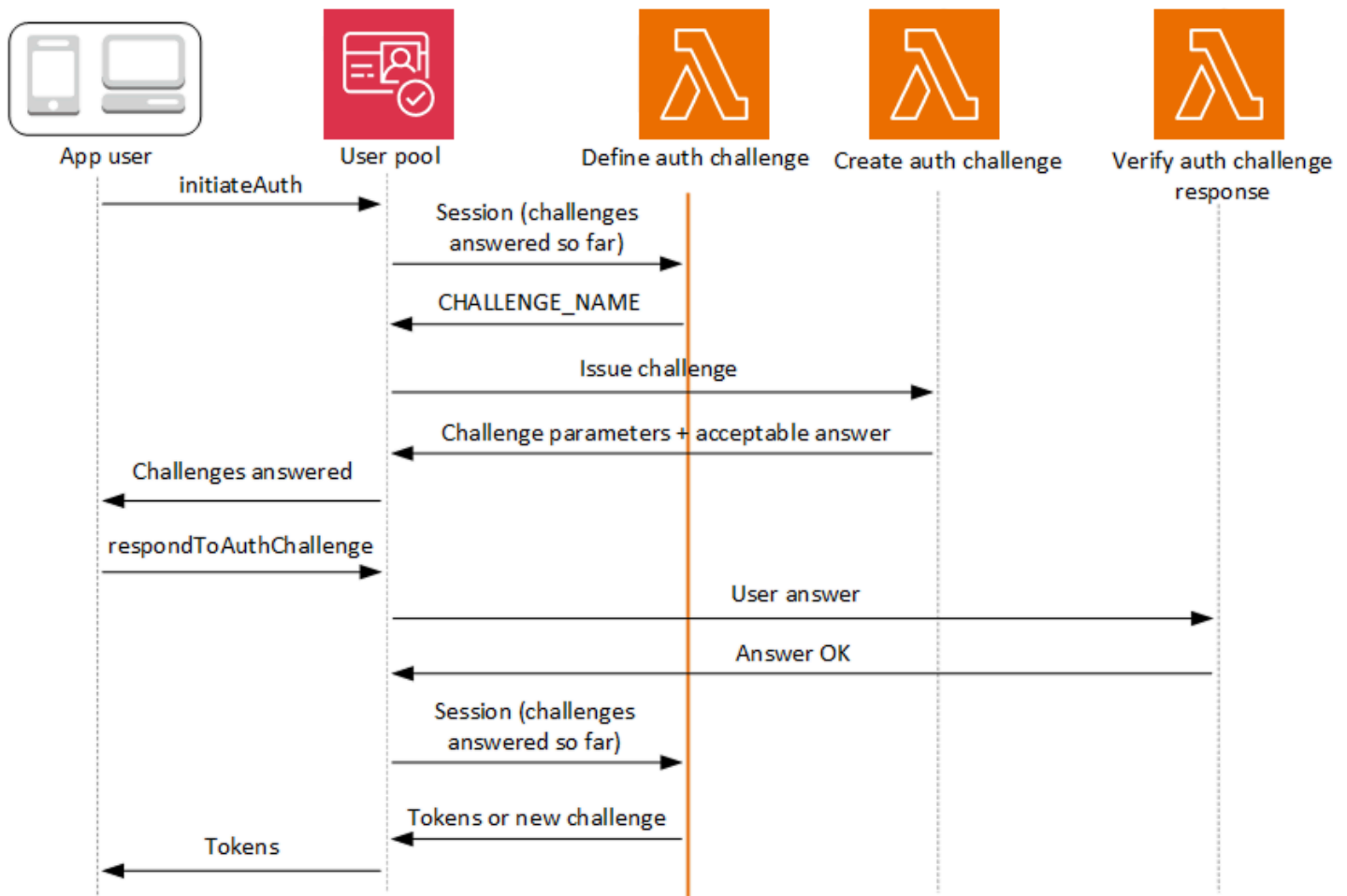
[AdminRespondToAuthChallenge](#) 之間的身分驗證。在此流程中，使用者身分驗證會透過回答連續挑戰，直到驗證失敗或者使用者發出字符為止。挑戰回應可能是新的挑戰。在此情況下，您的應用程式會視需要回應新挑戰的次數。當定義身分驗證挑戰函數分析到目前為止的結果、判斷所有挑戰都已回答，並傳回時，就會成功進行身分驗證 `IssueTokens`。

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

定義身分驗證挑戰觸發程序是一種 Lambda 函數，可在自訂身分驗證流程中維護挑戰序列。它宣告挑戰序列成功或失敗，並在序列尚未完成時設定下一個挑戰。



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。

以下是使用 SRP 流程進行自訂身分驗證之前啟動 InitiateAuth 請求的範例。

```
{
  "AuthFlow": "CUSTOM_AUTH",
  "ClientId": "lexample23456789",
  "AuthParameters": {
    "CHALLENGE_NAME": "SRP_A",
    "USERNAME": "testuser",
    "SRP_A": "[SRP_A]",
    "SECRET_HASH": "[secret hash]"
  }
}
```

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

下列挑戰類型之

— : CUSTOM_CHALLENGE、SRP_A、PASSWORD_VERIFIER、SMS_MFA、EMAIL_OTPSOFTWARE_TO或 ADMIN_NO_SRP_AUTH。

當您定義身分驗證挑戰函式為已設定多重要素驗證的使用者發出PASSWORD_VERIFIER挑戰時，Amazon Cognito 會追蹤該挑戰並顯示 SMS_MFA、EMAIL_OTP或 SOFTWARE_TOKEN_MFA挑戰。這些是多重要素驗證碼的提示。在您的函數中，包含處理來自 SMS_MFA、EMAIL_OTP和 SOFTWARE_TOKEN_MFA挑戰的輸入事件。您不需要在定義身分驗證挑戰函數中調用任何 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。當使用者使用 `SRP_A` 和 `PASSWORD_VERIFIER` 挑戰完成 `SRP` 身分驗證時，此函數會傳遞給他們 `CUSTOM_CHALLENGE`，以叫用建立身分驗證挑戰觸發程序。結合我們的[建立身分驗證挑戰範例](#)，此序列為挑戰 3 提供 `CAPTCHA` 挑戰，並為挑戰 4 提供安全性問題。

使用者解決 `CAPTCHA` 並回答安全問題後，此函數會確認您的使用者集區可以發出字符。不需要 `SRP` 身分驗證；您也可以將 `CAPTCHA` 和安全問題設定為挑戰一和挑戰二。如果您的定義身分驗證挑戰函數未宣告 `SRP` 挑戰，則使用者的成功完全取決於其對自訂提示的回應。

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" &&
```

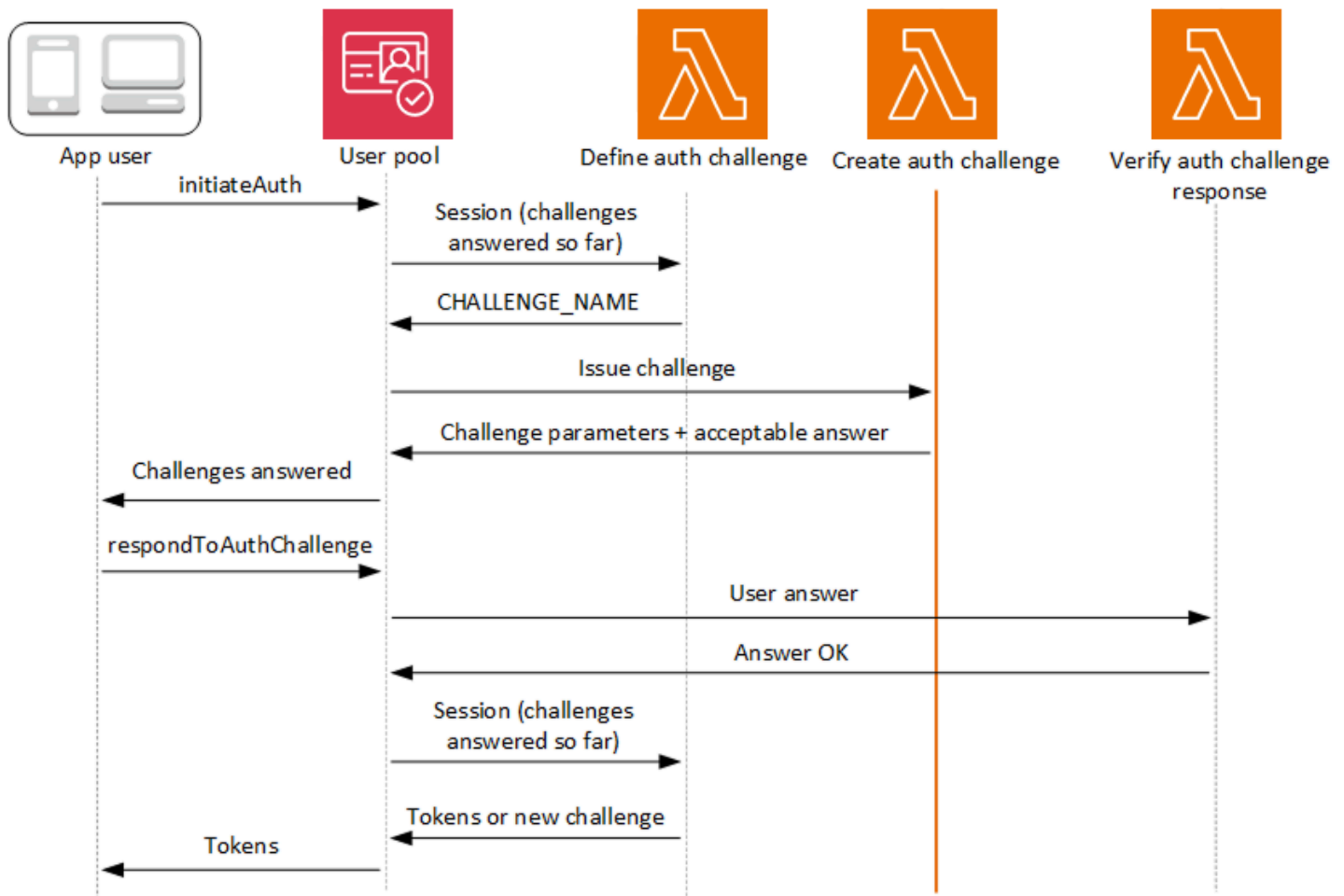
```
    event.request.session[2].challengeResult === true
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "CUSTOM_CHALLENGE";
  } else if (
    event.request.session.length === 4 &&
    event.request.session[3].challengeName === "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

建立身分驗證挑戰觸發程序是一種 Lambda 函數，其中包含定義身分驗證挑戰觸發程序所宣告的每個挑戰的詳細資訊。它會處理定義身分驗證挑戰觸發程序宣告的挑戰名稱 `publicChallengeParameters`，並傳回應用程式必須呈現給使用者的。然後，此函數為您的使用者集區提供對挑戰的答案 `privateChallengeParameters`，您的使用者集區會傳遞給驗證身分驗證挑戰觸發器。在您的定義身分驗證挑戰觸發程序管理挑戰序列的位置，您的建立身分驗證挑戰觸發程序會管理挑戰內容。



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

此函數有兩個自訂挑戰，對應於[定義身分驗證挑戰範例中的](#)挑戰序列。前兩個挑戰是 SRP 身分驗證。對於第三個挑戰，此函數會在挑戰回應中傳回 CAPTCHA URL 到您的應用程式。您的應用程式會在指定的 URL 轉譯 CAPTCHA，並傳回使用者的輸入。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。

對於第四個挑戰，此函數會傳回安全性問題。您的應用程式會呈現問題，並提示使用者回答。使用者解決這兩個自訂挑戰後，定義身分驗證挑戰觸發程序會確認您的使用者集區可以發出權杖。

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";
  }

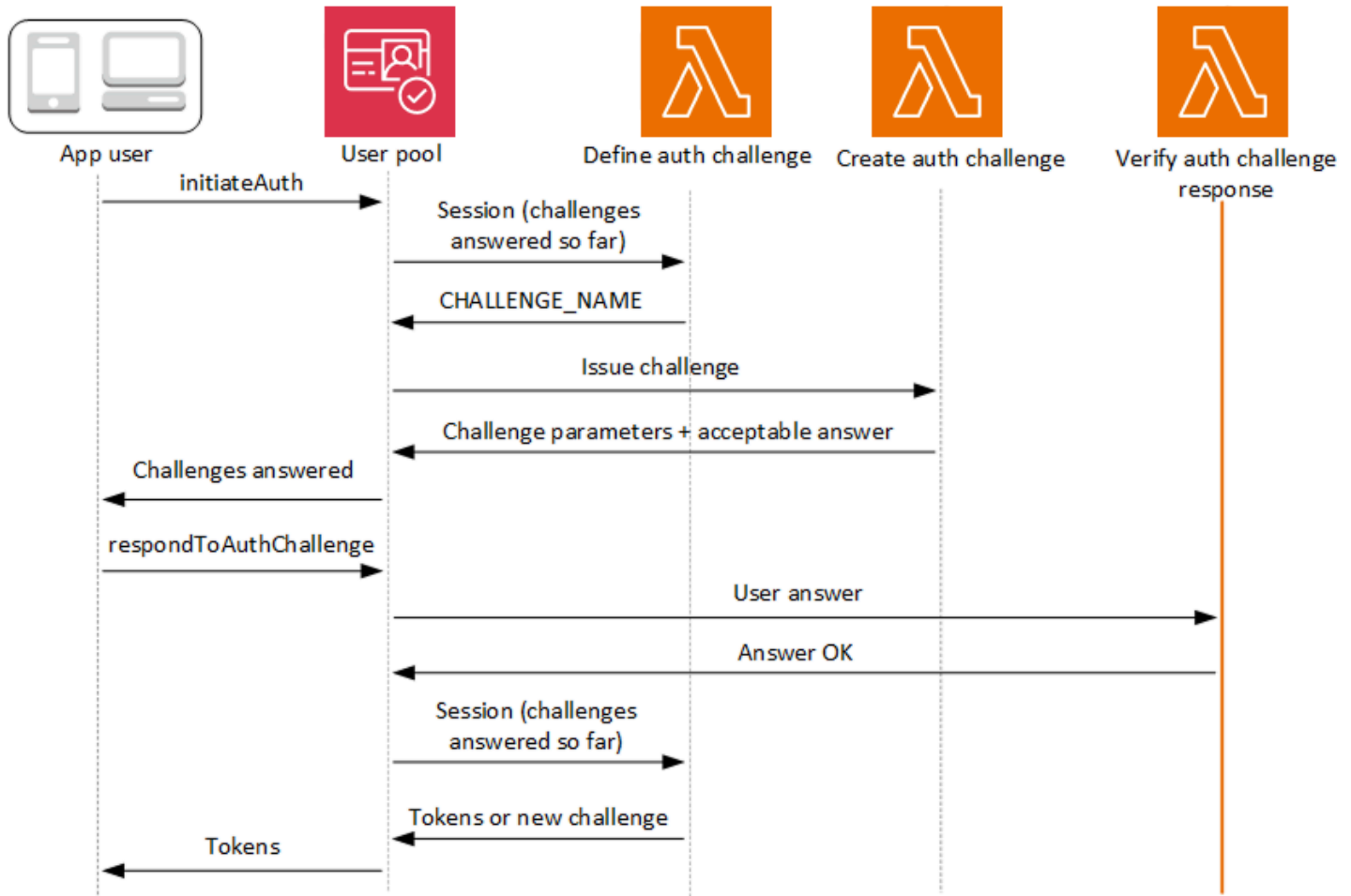
  if (event.request.session.length === 3) {
    event.response.publicChallengeParameters = {};
    event.response.privateChallengeParameters = {};
    event.response.publicChallengeParameters.securityQuestion =
      "Who is your favorite team mascot?";
    event.response.privateChallengeParameters.answer = "Peccy";
  }

  return event;
};

export { handler };
```

確認驗證挑戰回應 Lambda 觸發程序

驗證身分驗證挑戰觸發程序是一種 Lambda 函數，可將使用者提供的回應與已知答案進行比較。此函數會告知使用者集區使用者是否正確回答挑戰。當驗證身分驗證挑戰觸發條件以 `answerCorrect` 的回應時 `true`，身分驗證序列可以繼續。



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

此驗證驗證挑戰函數會檢查使用者對挑戰的回應是否符合預期的回應。使用者的答案是由您應用程式的輸入定義，而偏好的答案則由 [於建立身分驗證挑戰觸發](#) 回應的回應 `privateChallengeParameters.answer` 中定義。正確答案和給定的答案都是此函數輸入事件的一部分。

在此範例中，如果使用者的回應符合預期的回應，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 Essentials 或 Plus 功能計劃的使用者集區中，您可以使用存取權杖自訂產生第 2 版或 V2_0 觸發事件，以及針對 machine-to-machine (M2M) 用戶端憑證授予使用存取權杖自訂產生第 3 版或 V3_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 或 V3_0 事件是單一請求，其中包含 Amazon Cognito 將寫入身分和存取字符的資料。若要自訂這兩個權杖，您必須更新函數以使用觸發程序第 2 版或第 3 版，並在相同的回應中傳送這兩個權杖的資料。

Amazon Cognito 會將第 2 版事件回應套用至從使用者身分驗證存取字符，其中人工使用者已將登入資料提供給您的使用者集區。第 3 版事件回應適用於來自使用者身分驗證和機器身分驗證的存取權杖，其中自動化系統使用應用程式用戶端秘密授權存取權杖請求。除了產生的存取權杖之外，第 2 版和第 3 版的事件也相同。

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將不同版本的權杖產生前觸發事件交付至 Lambda 函數。V1_0 觸發會交付參數以修改 ID 字符。V2_0 或 V3_0 觸發程序會提供下列項目的參數。

1. V1_0 觸發的函數。
2. 自訂存取權杖的功能。
3. 將複雜資料類型傳遞至 ID 和存取權杖宣告值的能力：
 - 字串
 - Number
 - Boolean
 - 字串、數字、布林值或任何這些值的組合陣列
 - JSON

Note

在 ID 字符中，您可以將複雜的物件填入宣告值，但 `phone_number_verified`、`updated_at`、`email_verified` 和 `除外address`。

使用者集區預設會交付 V1_0 事件。若要將使用者集區設定為傳送 V2_0 事件，請在 Amazon Cognito 主控台中設定觸發時，選擇基本功能的觸發事件版本 + 使用者身分的存取權杖自訂。若要產生 V3_0 事件，請選擇基本功能 + 使用者和機器身分的存取權杖自訂。您也可以可以在 [UpdateUserPool](#) 或 [CreateUserPool](#) API 請求 LambdaVersion 的 [LambdaConfig](#) 參數中設定的值。事件版本 1、2 和 3 可在 Essentials 和 Plus 功能計劃中使用。第 3 版事件的 M2M 操作具有與每月作用中使用者 (MAU) 公式不同的定價結構。如需詳細資訊，請參閱 [Amazon Cognito 定價](#)。

Note

在 2024 年 11 月 22 日或之前在 1800 GMT 使用進階安全功能選項操作，且保留在 Lite 功能層的使用者集區，可以存取事件版本一和兩個產生權杖前觸發。沒有進階安全功能的使用者集區可以存取事件版本 1。第 3 版僅適用於 Essentials 和 Plus。

宣告和範圍參考

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。下表說明 Lambda 函數可以 and 無法修改的宣告，以及影響宣告存在或值的觸發事件參數。

取得	預設 字符 類型	可以 新增 嗎？	可以 修改 嗎？	可以 隱藏 嗎？	事件參數 - 新增或 修改	事件參數 - 抑制	身分類型	事件版本
不在使用者集區字符結構描述中的任何宣告	無	是	是	N/A	claimsToOverride	claimsToSuppress	使用者、機器 ¹	全部 ²
scope	存取	是	是	是	scopesToOverride	scopesToSuppress	使用者、機器 ¹	v2_0, v3_0
cognito:groups	ID、存取	是	是	是	groupsToOverride	claimsToSuppress	使用者	全部 ²
cognito:preferred_role	ID	是	是	是	preferredRole	claimsToSuppress	使用者	全部
cognito:roles	ID	是	是	是	iamRolesToOverride	claimsToSuppress	使用者	全部
cognito:username	ID	否	否	否	N/A	N/A	使用者	N/A
具有cognito的任何其他宣告	無	否	否	否	N/A	N/A	N/A	N/A
username	存取	否	否	否	N/A	N/A	使用者	v2_0, v3_0

取得	預設 字符 類型	可以 新增 嗎？	可以 修改 嗎？	可以 隱藏 嗎？	事件參數 - 新增或 修改	事件參數 - 抑制	身分類型	事件版本
sub	ID、 存取	否	否	否	N/A	N/A	使用者	N/A
標準 OIDC 屬 性	ID	是	是	是	claimsTo/ dd0r0ver: ide	claimsTo/ uppress	使用者	全部
custom: 屬性	ID	是	是	是	claimsTo/ dd0r0ver: ide	claimsTo/ uppress	使用者	全部
dev: 屬 性	ID	否	否	是	N/A	claimsTo/ uppress	使用者	全部
identiti s	ID	否	否	否	N/A	N/A	使用者	N/A
aud ⁴	ID	否	否	否	N/A	N/A	使用者、 機器	N/A
client_id	存取	否	否	否	N/A	N/A	使用者、 機器	N/A
event_id	存取	否	否	否	N/A	N/A	使用者、 機器	N/A
device_k y	存取	否	否	否	N/A	N/A	使用者	N/A
version	存取	否	否	否	N/A	N/A	使用者、 機器	N/A
acr	ID、 存取	否	否	否	N/A	N/A	使用者、 機器	N/A

取得	預設 字符 類型	可以 新增 嗎？	可以 修改 嗎？	可以 隱藏 嗎？	事件參數 - 新增或 修改	事件參數 - 抑制	身分類型	事件版本
amr	ID、 存取	否	否	否	N/A	N/A	使用者、 機器	N/A
at_hash	ID	否	否	否	N/A	N/A	使用者、 機器	N/A
auth_time	ID、 存取	否	否	否	N/A	N/A	使用者、 機器	N/A
azp	ID、 存取	否	否	否	N/A	N/A	使用者、 機器	N/A
exp	ID、 存取	否	否	否	N/A	N/A	使用者、 機器	N/A
iat	ID、 存取	否	否	否	N/A	N/A	使用者、 機器	N/A
iss	ID、 存取	否	否	否	N/A	N/A	使用者、 機器	N/A
jti	ID、 存取	否	否	否	N/A	N/A	使用者、 機器	N/A
nbf	ID、 存取	否	否	否	N/A	N/A	使用者、 機器	N/A
nonce	ID、 存取	否	否	否	N/A	N/A	使用者、 機器	N/A
origin_jti	ID、 存取	否	否	否	N/A	N/A	使用者、 機器	N/A
token_use	ID、 存取	否	否	否	N/A	N/A	使用者、 機器	N/A

¹ 機器身分的存取字符僅適用於觸發輸入事件v3_0的。事件第 3 版僅適用於 Essentials 和 Plus 功能層。Lite 層上的使用者集區可以接收v1_0事件。Lite 層上具有進階安全功能的使用者集區可以接收v1_0和v2_0事件。

² 將產生字符前觸發條件設定為僅限 v1_0 ID 字符的事件版本，v2_0適用於 ID 和存取字符，v3_0適用於具有機器身分功能的 ID 和存取字符。

³ 若要隱藏 cognito:preferred_role和 cognito:roles宣告，請將 cognito:groups新增至claimsToSuppress。

⁴ 您可以新增 aud宣告來存取字符，但其值必須符合目前工作階段的應用程式用戶端 ID。您可以從 event.callerContext.clientId 的請求事件中衍生出用戶端 ID。

自訂身分權杖

透過產生字符前 Lambda 觸發程序的所有事件版本，您可以從使用者集區自訂身分 (ID) 字符的內容。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 [了解身分 \(ID\) 字符](#)。

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

透過產生字符前 Lambda 觸發程序的事件版本 2 和 3，您可以從使用者集區自訂存取字符的內容。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。對於具有用戶端憑證授權的machine-to-machine(M2M) 授權，Amazon Cognito 只會在您的使用者集區設定為第 3 版 (V3_0) 事件時調用權杖產生前觸發條件。如需存取權杖的詳細資訊，請參閱 [了解存取權杖](#)。

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. 選擇延伸項目功能表並尋找 Lambda 觸發條件。
4. 新增或編輯產生權杖前觸發程序。
5. 在指派 Lambda 函數下選擇一個 Lambda 函數。
6. 為使用者身分選擇基本功能 + 存取權杖自訂的觸發事件版本，或為使用者和機器身分選擇基本功能 + 存取權杖自訂。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 [UpdateUserPool](#) API 請求。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[更新使用者集區和應用程式用戶端組態](#)。

在請求的 LambdaVersion 參數中包含以下內容。LambdaVersion 的值 V2_0 會導致您的使用者集區新增參數，並將變更套用至存取權杖。LambdaVersion 的值 V3_0 會產生與相同的事件 V2_0，但會導致您的使用者集區也套用變更至 M2M 存取權杖。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": "V3_0"
},
```

machine-to-machine(M2M) 用戶端登入資料的用戶端中繼資料

您可以在 M2M 請求中傳遞[用戶端中繼資料](#)。用戶端中繼資料是來自使用者或應用程式環境的其他資訊，有助於的結果[產生權杖前 Lambda 觸發程序](#)。在使用者主體的身分驗證操作中，您可以將用戶端中繼資料傳遞至 [AdminRespondToAuthChallenge](#) 和 [RespondToAuthChallenge](#) API 請求內文中的權杖產生前觸發條件。由於應用程式會執行流程來產生 M2M 的存取字符，並對 提出直接請求[權杖端點](#)，因此具有不同的模型。在用戶端登入資料的字符請求 POST 內文中，將具有用戶端中繼資料物件 URL 編碼 (x-www-form-urlencoded) 的 `aws_client_metadata` 參數傳遞至字串。如需請求範例，請參閱 [具有基本授權的用戶端憑證](#)。以下是傳遞鍵/值對 的範例參數 `{"environment": "dev", "language": "en-US"}`。

```
aws_client_metadata=%7B%22environment%22%3A%20%22dev%22,%20%22language%22%3A%20%22en-US%22%7D
```

其他 資源

- [如何在 Amazon Cognito 使用者集區中自訂存取權杖](#)

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](#)
- [權杖產生前事件第 2 版範例：新增具有複雜物件的宣告](#)
- [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	從 Amazon Cognito 受管登入頁面進行身分驗證期間呼叫。
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_ClientCredentials	在 M2M 用戶端憑證授予後呼叫。您的使用者集區只會在您的事件版本為 時傳送此事件V3_0。
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 one

第 1 版權杖可以在 ID 權杖中設定群組成員資格、IAM 角色和新宣告。群組成員資格覆寫也適用於存取權杖中的cognito:groups宣告。

```
{
  "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"
        }
      }
    }
  }
}

```

Versions two and three

第 2 版和第 3 版請求事件會新增自訂存取權杖的欄位。使用者集區會套用第 3 版事件的變更，以存取機器身分的字符。這些版本也新增對回應物件中複雜claimsToOverride資料類型的支援。您的 Lambda 函數可以傳回值為 的下列資料類型claimsToOverride：

- 字串
- Number
- Boolean
- 字串、數字、布林值或任何這些值的組合陣列
- 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

名稱	描述	最小觸發事件版本
groupConfiguration	包含目前群組組態的輸入物件。物件包含 groupsToOverride 、 iamRolesToOverride 和 preferredRole 。	1
groupsToOverride	您的用戶所屬的 使用者集區 。	1
iamRolesToOverride	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 不包含來自其傳遞至權杖產生前函數的請求中的 AdminInitiateAuth 和 InitiateAuth API 操作中的 ClientMetadata 參數的資料。	1
scopes	存取字符範圍。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
<code>claimsOverrideDetails</code>	A container for all elements in a <code>V1_0</code> trigger event.	1
<code>claimsAndScopeOverrideDetails</code>	或 <code>V2_0</code> / <code>V3_0</code> 觸發事件中所有元素的容器。	2
<code>idTokenGeneration</code>	您要在使用者 ID 權杖中覆寫、新增或抑制的宣告。此 ID 字符的父項自訂值只會出現在事件版本 2 及更高版本中，但子元素會出現在版本 1 事件中。	2
<code>accessTokenGeneration</code>	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。此存取字符自訂值的父項只會出現在事件版本 2 及更高版本中。	2
<code>claimsToAddOrOverride</code>	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 <code>groupOverrideDetails</code> 。 在事件版本 2 和更新版本中，此元素會出現在 <code>accessTokenGeneration</code> 和 <code>idTokenGeneration</code> 。	1 [*]
<code>claimsToSuppress</code>	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。 在事件版本 2 和更新版本中，此元素會出現在 <code>accessTokenGeneration</code> 和 <code>idTokenGeneration</code> 。	1
<code>groupOverrideDetails</code>	包含目前群組組態的輸出物件。物件包含 <code>groupsToOverride</code> 、 <code>iamRolesToOverride</code> 和 <code>preferredRole</code> 。 您的函數將使用您提供的物件取代 <code>groupOverrideDetails</code> 物件。如果您在回應中提供空的或空物	1

名稱	描述	最小觸發事件版本
	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 <code>groupConfiguration</code> 物件值複製到回應中的 <code>groupOverrideDetails</code> 物件。然後將其傳回服務。 Amazon Cognito ID 權杖和存取權杖都包含 <code>cognito:groups</code> 宣告。您的 <code>groupOverrideDetails</code> 物件取代存取權杖以及 ID 權杖中的 <code>cognito:groups</code> 宣告。群組覆寫是第 1 版事件可以對存取權杖進行的唯一變更。	
<code>scopesToAdd</code>	您要新增至使用者存取字串中 <code>scope</code> 宣告的範圍清單。您無法新增包含一或多個空白字元的範圍值。	2
<code>scopesToSuppress</code>	您要從使用者存取權杖中的 <code>scope</code> 宣告中移除的範圍清單。	2

* 第一版事件的回應物件可以傳回字串。回應第 2 版和第 3 版事件的物件可能會傳回[複雜的物件](#)。

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": []
}
}

```

權杖產生前事件第 2 版範例：新增具有複雜物件的宣告

此範例會對使用者的權杖進行下列修改。

1. 將數字、字串、布林值和 JSON 類型的宣告新增至 ID 字符。這是第 2 版觸發事件提供給 ID 字符的唯一變更。
2. 將數字、字串、布林值和 JSON 類型的宣告新增至存取權杖。
3. 將三個範圍新增至存取權杖。
4. 隱藏 ID 和存取權杖中的 email 宣告。
5. 隱藏存取字符中的 `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"]
        },
        "accessTokenGeneration": {
            "claimsToAddOrOverride": claims,
            "claimsToSuppress": ["email"],
            "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::XXXXXXXXXXXX:role/sns_callerA",
          "arn:aws:iam::XXXXXXXXXXXX:role/sns_callerB",
          "arn:aws:iam::XXXXXXXXXXXX:role/sns_callerC",
        ],
        preferredRole: "arn:aws:iam::XXXXXXXXXXXX: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 觸發程序匯入使用者](#)。

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 ¹	登入時的使用者遷移。
UserMigration_ForgotPassword	忘記密碼流程時的使用者遷移。

¹ 當使用者使用[無密碼登入](#)進行身分驗證時，Amazon Cognito 不會叫用此觸發。

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	需求
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。 如果請求和使用使用者集區符合別名要求，則函數的回應必須將收到的 <code>username</code> 參數指派給別名屬性。此外，回應必須將您自己的值指派給 <code>username</code> 屬性。如果您的使用者集區不符合將接收的 <code>username</code> 映射至別名所需要的條件，則回應中的 <code>username</code> 參數必須與請求完全匹配，或省略。  Note <code>username</code> 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，並且您的用戶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戶端應用程式。

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

```
exports.handler = (event, context, callback) => {
  var user;

  if (event.triggerSource == "UserMigration_Authentication") {
    // authenticate the user with your existing user directory service
    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";
      context.succeed(event);
    } else {
      // Return error to Amazon Cognito
      callback("Bad password");
    }
  } else if (event.triggerSource == "UserMigration_ForgotPassword") {
    // Lookup the user in your existing user directory service
    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";
      context.succeed(event);
    } else {
      // Return error to Amazon Cognito
      callback("Bad password");
    }
  } else {
    // Return error to Amazon Cognito
    callback("Bad triggerSource " + event.triggerSource);
  }
}
```

```
};
```

自訂訊息 Lambda 觸發程序

當您擁有要傳送給使用者的電子郵件和簡訊的外部標準時，或您想要在執行時間將自己的邏輯套用至使用者訊息的格式時，請將自訂訊息觸發條件新增至您的使用者集區。自訂訊息 Lambda 會在您的使用者集區傳送所有電子郵件和簡訊之前收到內容。您的 Lambda 函數接著有機會修改訊息內容和主旨。

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。

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` 觸發來源之自訂訊息 Lambda 函數的輸入事件包含使用者名稱和驗證碼。由於管理員建立的使用者必須同時收到其使用者名稱和密碼，因此函數的回應必須包含使用者名稱和密碼的預留位置變數。訊息的預留位置是 `request.usernameParameter` 和 `request.codeParameter` 的值。這些值通常是 `{username}` 和 `{#####}`；最佳實務是參考輸入值，而不是硬式編碼變數名稱。

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。

triggerSource 值	事件
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請求不包括傳遞至 [AdminInitiateAuth](#) 和 [InitiateAuth](#) 中的 ClientMetadata 參數的 API 操作的資料。若要將此資料傳遞至您的 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.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。如果使用者集區的 `EmailSendingAccount` 屬性為 `DEVELOPER`，那麼您只能使用 `emailSubject` 參數。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.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",
  "region": "us-west-2",
  "userPoolId": "us-west-2_EXAMPLE",
  "userName": "test-user",
  "callerContext": {
    "awsSdkVersion": "aws-sdk-unknown-unknown",
    "clientId": "1example23456789"
  },
  "triggerSource": "CustomMessage_SignUp",
  "request": {
    "userAttributes": {
      "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
      "cognito:user_status": "CONFIRMED",
      "email_verified": "true",
      "phone_number_verified": "true",
      "phone_number": "+12065551212",
      "email": "test-user@example.com"
    },
    "codeParameter": "{####}",
    "linkParameter": "{##Click Here##}",
    "usernameParameter": "None"
  },
  "response": {
    "smsMessage": "None",
```

```
"emailMessage": "None",
"emailSubject": "None"
}
}
```

管理員建立使用者的自訂訊息範例

Amazon Cognito 傳送到此範例自訂訊息 Lambda 函數的請求，其triggerSource值為 CustomMessage_AdminCreateUser，使用者名稱和臨時密碼。函數`${event.request.codeParameter}`會從請求中的臨時密碼，以及`${event.request.usernameParameter}`請求中的使用者名稱填入。

您的自訂訊息必須將 codeParameter和 的值插入回應物件emailMessage中的 usernameParametersmsMessage和 。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

Lambda 會在使用者集區中觸發 CustomEmailSender 和 CustomSMSSender 支援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。當 Amazon Cognito 傳送邀請、MFA 代碼、確認碼、驗證碼和臨時密碼給使用者時，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 會使用 AWS KMS 客戶受管金鑰和來加密這些秘密 AWS Encryption SDK。AWS Encryption SDK 是用戶端加密程式庫，可協助您加密和解密一般資料。

Note

若要將使用者集區設定為使用這些 Lambda 觸發條件，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。

必要的資源

Amazon Cognito 不會在傳送給自訂寄件者觸發條件的事件中以純文字傳送使用者的代碼。Lambda 函數必須解密事件中的程式碼。下列概念是您的函數必須用來取得可交付給使用者的程式碼的加密架構。

AWS KMS

AWS KMS 是建立和控制 AWS KMS 金鑰的受管服務。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。使用自訂寄件者觸發，您的 AWS Lambda 函數可以透過您選擇的方法和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。

此觸發程序適用於您可能想要更好地控制使用者集區傳送電子郵件訊息的方式的情況。您的 Lambda 函數可以自訂對 Amazon SES API 操作的呼叫，例如當您想要管理多個已驗證的身分或跨時 AWS 區域。您的函數也可能將訊息重新導向至另一個交付媒體或第三方服務。

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 () 中建立 [對稱加密金鑰](#) AWS KMS。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。[AWS Encryption SDK](#) 是您函數中 AWS KMS 操作的實用工具。
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_Authentication	使用者登入，Amazon Cognito 會傳送多重要素驗證 (MFA) 代碼。
CustomEmailSender_ForgotPassword	使用者請求代碼以重置其密碼。
CustomEmailSender_ResendCode	使用者請求替換帳戶確認碼。
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

type

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。

Note

Amazon Cognito 會在具有下列觸發來源的事件中 ClientMetadata 傳送至自訂電子郵件觸發函數：

- CustomEmailSender_ForgotPassword
- CustomEmailSender_SignUp
- CustomEmailSender_Authentication

Amazon Cognito 不會 ClientMetadata 以來源 傳送觸發事件 CustomEmailSender_AccountTakeOverNotification。

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 Lambda 函數必須解譯事件並解密程式碼，然後傳遞訊息內容。典型的 函數會組合電子郵件訊息，並將其導向至第三方 SMTP 轉送。

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. 建立或更新使用者集區的 IAM 主體會針對 Amazon Cognito 用來加密程式碼的 KMS 金鑰建立一次性授權。為您的 KMS 金鑰授予此主體 `CreateGrant` 許可。若要讓此範例 KMS 金鑰政策有效，更新使用者集區的管理員必須使用 IAM 角色 的擔任角色工作階段登入 `arn:aws:iam::111222333444:role/my-example-role`。

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": {
      "AWS": "arn:aws:iam::111222333444:role/my-example-role"
    },
    "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 命令會授予 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。在 Amazon Cognito 將秘密傳送至自訂寄件者 Lambda 函數之前，Amazon Cognito 會使用 AWS Encryption SDK 來加密秘密。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. 將 AWS Encryption SDK 新增至 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. 開啟您的 函數並新增 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[更新使用者集區和應用程式用戶端組態](#)。

```
#Send this parameter in an 'aws cognito-idp update-user-pool' CLI command,
including any existing
#user pool configurations. This snippet also includes a pre sign-up trigger for
syntax reference. The pre sign-up trigger
#doesn't have a role in custom sender triggers.

--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 觸發條件update-user-pool AWS CLI，請從 省略 CustomSMSSender或 CustomEmailSender 參數--lambda-config，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)。

部署此函數

1. 在您的開發人員工作區中安裝最新版本的 NodeJS。
2. 在工作區中建立新的 NodeJS 專案。
3. 使用 初始化您的專案 `npm init -y`。
4. 建立 Lambda 函數的指令碼：`touch index.mjs`。
5. 將下列範例的內容貼到 `index.mjs`。
6. 下載專案相依性 AWS Encryption SDK：`npm install @aws-crypto/client-node`。
7. 將專案目錄壓縮為檔案：`zip -r my_deployment_package.zip ..`。
8. [將 ZIP 檔案部署到您的 函數](#)。

此範例函數會解密程式碼，對於註冊事件，則模擬傳送電子郵件訊息到使用者的電子郵件地址。

```
import { KmsKeyringNode, buildClient, CommitmentPolicy } from '@aws-crypto/client-node';

// Configure the encryption SDK client with the KMS key from the environment variables
const { encrypt, decrypt } = buildClient(
  CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT
);

const generatorKeyId = process.env.KEY_ID;
const keyIds = [process.env.KEY_ARN];
const keyring = new KmsKeyringNode({ generatorKeyId, keyIds });

// Example function to simulate sending email.
// This example logs message details to CloudWatch Logs from your Lambda function.
// Update this function with custom logic that sends an email message to 'emailaddress'
// with body 'message'.
const sendEmail = async (emailAddress, message) => {
  // Log the destination with the email address masked.
  console.log(`Simulating email send to ${emailAddress.replace(/^[^@.]/g, '*')}`);
  // Log the message with the code masked.
  console.log(`Message content: ${message.replace(/\b\d{6,8}\b/g, '*****')}`);
  // Simulate API delay
```

```
    await new Promise(resolve => setTimeout(resolve, 100));
    console.log('Email sent successfully');
    return true;
};

export const handler = async (event) => {
  try {
    // Decrypt the secret code using encryption SDK
    let plainTextCode;
    if (event.request.code) {
      const { plaintext, messageHeader } = await decrypt(keyring,
Buffer.from(event.request.code, 'base64'));
      plainTextCode = Buffer.from(plaintext).toString('utf-8');
    }

    // Handle different trigger sources
    if (event.triggerSource == 'CustomEmailSender_SignUp') {
      const emailAddress = event.request.userAttributes.email;
      const message = `Welcome! Your verification code is: ${plainTextCode}`;
      await sendEmail(emailAddress, message);
    }
    else if (event.triggerSource == 'CustomEmailSender_ResendCode') {
      // Handle resend code
    }
    else if (event.triggerSource == 'CustomEmailSender_ForgotPassword') {
      // Handle forgot password
    }
    else if (event.triggerSource == 'CustomEmailSender_UpdateUserAttribute') {
      // Handle update attribute
    }
    else if (event.triggerSource == 'CustomEmailSender_VerifyUserAttribute') {
      // Handle verify attribute
    }
    else if (event.triggerSource == 'CustomEmailSender_AdminCreateUser') {
      // Handle admin create user
    }
    else if (event.triggerSource == 'CustomEmailSender_Authentication') {
      // Handle authentication
    }
    else if (event.triggerSource ==
'CustomEmailSender_AccountTakeOverNotification') {
      // Handle account takeover notification
    }
  }
}
```

```
        return;
    } catch (error) {
        console.error('Error in custom email sender:', error);
        throw error;
    }
};
```

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。透過自訂寄件者觸發，您的 AWS Lambda 函數可以透過您選擇的方法和提供者傳送簡訊通知給使用者。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簡訊。

此觸發程序適用於您可能想要更妥善控制使用者集區傳送簡訊方式的情況。您的 Lambda 函數可以自訂對 Amazon SNS API 操作的呼叫，例如當您想要管理多個起始 IDs 或跨時 AWS 區域。您的函數也可能將訊息重新導向至另一個交付媒體或第三方服務。

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 () 中建立 [對稱加密金鑰](#) AWS KMS。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。[AWS Encryption SDK](#) 是您函數中 AWS KMS 操作的實用工具。
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	使用者請求新的程式碼以確認其註冊。
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。

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 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. 建立或更新使用者集區的 IAM 主體會針對 Amazon Cognito 用來加密程式碼的 KMS 金鑰建立一次性授權。為您的 KMS 金鑰授予此主體 `CreateGrant` 許可。若要讓此範例 KMS 金鑰政策有效，更新使用者集區的管理員必須使用 IAM 角色的擔任角色工作階段登入 `arn:aws:iam::111222333444:role/my-example-role`。

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": {
      "AWS": "arn:aws:iam::111222333444:role/my-example-role"
    },
    "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 命令會授予 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。在 Amazon Cognito 將秘密傳送至自訂寄件者 Lambda 函數之前，Amazon Cognito 會使用 AWS Encryption SDK 來加密秘密。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. 將 AWS Encryption SDK 新增至 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. 開啟您的 函數並新增 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[更新使用者集區和應用程式用戶端組態](#)。

```
#Send this parameter in an 'aws cognito-idp update-user-pool' CLI command,
including any existing
#user pool configurations. This snippet also includes a pre sign-up trigger for
syntax reference. The pre sign-up trigger
#doesn't have a role in custom sender triggers.

--lambda-config "PreSignUp=lambda-arn, \
                  CustomSMSSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
                  CustomEmailSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
                  \
                  KMSKeyID=key-id"
```

若要移除具有 的自訂寄件者 Lambda 觸發條件update-user-pool AWS CLI，請省略 CustomSMSSender或 CustomEmailSender 參數--lambda-config，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 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)。

部署此函數

1. 在您的開發人員工作區中安裝最新版本的 NodeJS。

2. 在工作區中建立新的 NodeJS 專案。
3. 使用 初始化您的專案 `npm init -y`。
4. 建立 Lambda 函數的指令碼：`touch index.mjs`。
5. 將下列範例的內容貼到 `index.mjs`。
6. 下載專案相依性：`npm install @aws-crypto/client-node`。
7. 將專案目錄壓縮為檔案：`zip -r my_deployment_package.zip ..`。
8. [將 ZIP 檔案部署到您的 函數](#)。

```
import { KmsKeyringNode, buildClient, CommitmentPolicy } from '@aws-crypto/client-node';

// Configure the encryption SDK client with the KMS key from the environment variables
const { encrypt, decrypt } = buildClient(
  CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT
);

const generatorKeyId = process.env.KEY_ID;
const keyIds = [process.env.KEY_ARN];
const keyring = new KmsKeyringNode({ generatorKeyId, keyIds });

// Example function to simulate sending SMS.
// This example logs message details to CloudWatch Logs from your Lambda function.
// Update this function with custom logic that sends an SMS message to 'phoneNumber'
// with body 'message'.
const sendSMS = async (phoneNumber, message) => {
  // Log the destination with the phone number masked.
  console.log(`Simulating SMS send to ${phoneNumber.replace(/[^\+]/g, '*')}`);
  // Log the message with the code masked.
  console.log(`Message content: ${message.replace(/\b\d{6,8}\b/g, '*****')}`);
  // Simulate API delay
  await new Promise(resolve => setTimeout(resolve, 100));
  console.log('SMS sent successfully');
  return true;
};

export const handler = async (event) => {
  try {
    // Decrypt the secret code using encryption SDK
    let plainTextCode;
    if (event.request.code) {
```

```
const { plaintext, messageHeader } = await decrypt(keyring,
Buffer.from(event.request.code, 'base64'));
plainTextCode = Buffer.from(plaintext).toString('utf-8');
}

// Handle different trigger sources
if (event.triggerSource == 'CustomSMSSender_SignUp') {
    const phoneNumber = event.request.userAttributes.phone_number;
    const message = `Welcome! Your verification code is: ${plainTextCode}`;
    await sendSMS(phoneNumber, message);
}
else if (event.triggerSource == 'CustomSMSSender_ResendCode') {
    // Handle resend code
}
else if (event.triggerSource == 'CustomSMSSender_ForgotPassword') {
    // Handle forgot password
}
else if (event.triggerSource == 'CustomSMSSender_UpdateUserAttribute') {
    // Handle update attribute
}
else if (event.triggerSource == 'CustomSMSSender_VerifyUserAttribute') {
    // Handle verify attribute
}
else if (event.triggerSource == 'CustomSMSSender_AdminCreateUser') {
    // Handle admin create user
}
return;
} catch (error) {
    console.error('Error in custom SMS sender:', error);
    throw error;
}
};
```

主題

- [使用自訂 SMS 寄件者函數評估 SMS 訊息功能](#)

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 SMS 寄件者 Lambda 函數來測試 Amazon Cognito SMS 簡訊的測試環境，請參閱 [GitHub 上的 aws-samples 程式庫](#) 中的 [amazon-cognito-user-pool-development-and-testing-with-sms-redirected-to-email](#)。儲存庫包含可建立新的使用者集區的 AWS CloudFormation 範本，或使用您已擁有的使用者集區。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，請復原 CloudFormation 堆疊，或從使用者集區中移除自訂 SMS 寄件者函數分配。

Important

請勿使用 [amazon-cognito-user-pool-development-and-testing-with-sms-redirected-to-email](#) 中的範本建置生產環境。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。

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。

管理 Amazon Cognito 使用者集區中的使用者涉及各種組態選項和管理任務。使用者集區可以擴展到數百萬個使用者。此規模的使用者目錄需要同樣可擴展且可重複的管理工具。您可能想要建立許多使用者設定檔、管理非作用中的使用者、產生控管和合規報告，或設定使用者執行大部分工作的自助式工具。建立使用者集區後，您可以控制使用者註冊和確認其帳戶的方式，包括要求電子郵件或電話號碼驗證。管理員也可以直接建立使用者帳戶，並自訂歡迎訊息和密碼要求。

使用者集區具有使用者群組，您可以根據使用者的群組成員資格來管理對資源的存取。您可以將 IAM 角色指派給這些群組，AWS 服務以使用身分集區管理對的存取。ID 和存取權杖中都存在使用者的群組成員資格。透過此資訊，您可以在應用程式中的執行時間或使用 Amazon Verified Permissions 等政策引擎進行存取控制決策。

使用者集區通常有許多使用者。您會經常發現自己正在搜尋和更新使用者帳戶。Amazon Cognito 主控台和 API 支援根據使用者名稱、電子郵件和電話號碼等標準屬性來查詢使用者。管理員也可以重設密碼、停用帳戶和檢視使用者事件歷史記錄。

若要遷移現有的使用者資料，Amazon Cognito 可以選擇從 CSV 檔案匯入使用者，並使用 [Lambda 觸發](#)程序在使用者第一次登入時自動遷移使用者。這些選項支援使用者從其他使用者目錄轉換到您的使用者集區。

您可以使用使用者集區中的使用者管理功能，對使用者生命週期和身分驗證體驗進行精細控制。自助註冊、管理員建立的帳戶、群組和遷移工具的組合，使得 Amazon Cognito 使用者集區成為靈活的使用者目錄。

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](#)
- [密碼、帳戶復原和密碼政策](#)
- [將使用者匯入使用者集區](#)
- [使用使用者屬性](#)

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。接受註冊且具有應用程式用戶端的使用者集區，無論是否有受管登入，都會為網際網路上知道您可公開探索之應用程式用戶端 ID 和註冊請求的使用者建立使用者設定檔。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

使用者集區的註冊功能表包含使用者集區中使用者註冊和管理建立的一些設定。

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時設定必要的屬性。
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。在自我註冊處於非作用中狀態的使用者集區中，[SignUp](#) API 請求會傳回 `NotAuthorizedException`，且受管登入不會顯示註冊連結。

對於您計劃以管理員身分建立使用者的使用者集區，您可以在管理員設定的登入功能表臨時密碼中的設定中設定其臨時密碼的持續時間。

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。在訊息範本下的身分驗證方法選單中自訂此訊息範本。

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 主控台的應用程式用戶端選單中建立具有用戶端秘密的應用程式用戶端。

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。

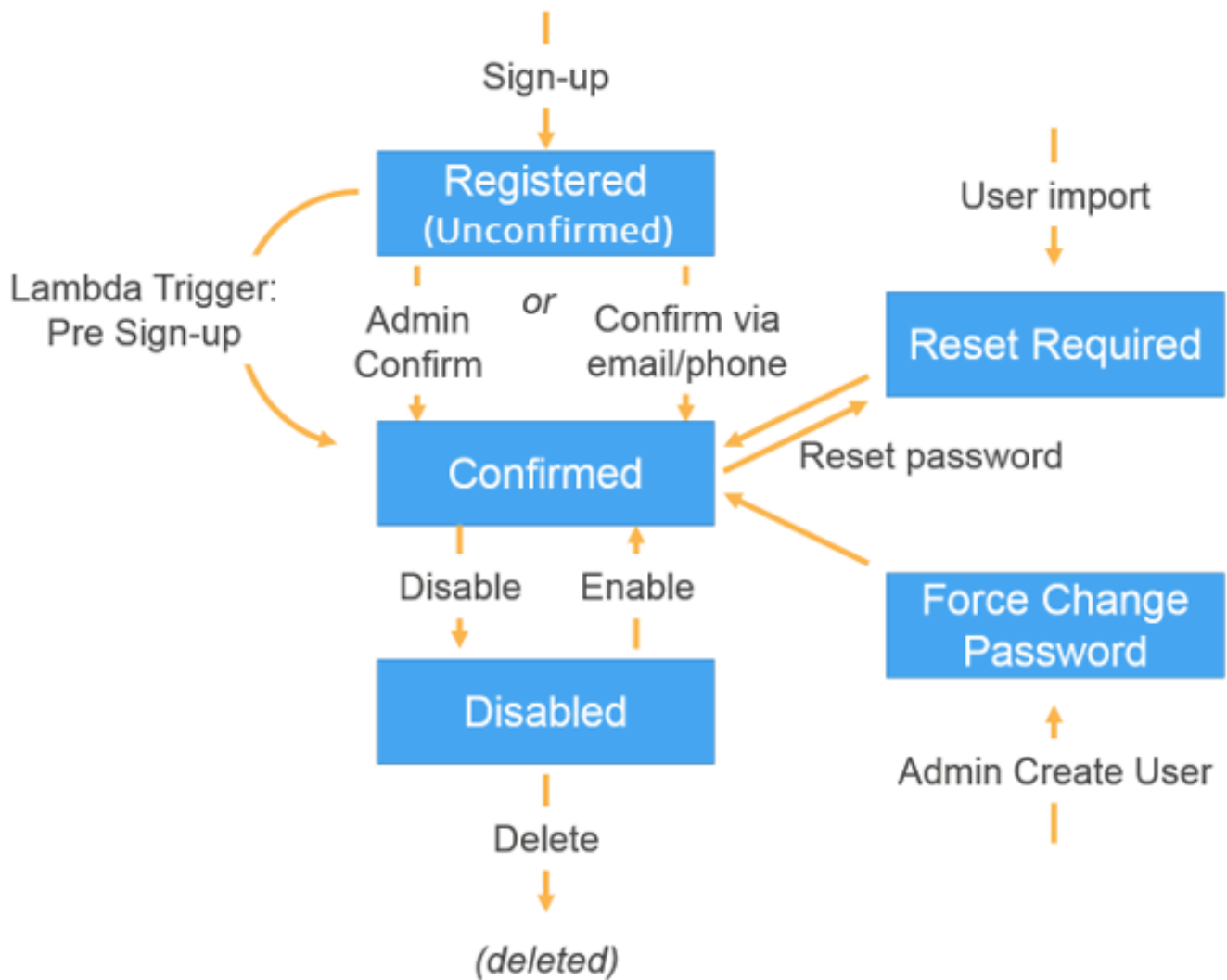
註冊時的密碼

Amazon Cognito 會在所有使用者註冊時要求他們提供密碼，但下列情況除外。如果符合所有這些條件，您可以在註冊操作中省略密碼。

1. 您的使用者集區和應用程式用戶端中已啟用[無密碼登入](#)。
2. 您的應用程式是使用 AWS SDK 中的身分驗證模組自訂建置的。受管登入和託管 UI 一律需要密碼。
3. 使用者提供您所允許之無密碼登入方法的屬性值，例如電子郵件或簡訊一次性密碼 OTPs)。例如，如果您允許使用電子郵件和電話 OTP 登入，使用者可以提供電話號碼或電子郵件地址，但如果您只允許使用電子郵件登入，則必須提供電子郵件地址。
4. 您的使用者集區[會自動驗證](#)使用者可以與無密碼登入搭配使用的屬性。
5. 對於任何指定的 [SignUp](#) 請求，使用者不提供[密碼](#)參數的值。

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。

其他 資源

- [使用 Amazon Cognito 偵測和修復非作用中使用者帳戶](#)

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给合宜的人士，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，且您的使用者代碼或連結不再有效，請呼叫 [ResendConfirmationCode](#) API 操作以產生並傳送新代碼或連結。

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，您可以使用 AdminGetUser API 操作、admin-get-user 命令與 AWS CLI，或其中一個 AWS SDKs 中的對應動作。

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。

確認和驗證使用者屬性的其他動作

下列使用者活動會驗證使用者屬性。您不需要將這些屬性設定為自動驗證：列出的動作在所有情況下都會將其標記為已驗證。

電子郵件地址

1. 使用電子郵件一次性密碼 (OTP) 成功完成[無密碼身分驗證](#)。
2. 使用電子郵件 OTP 成功完成[多重驗證 \(MFA\)](#)。

電話號碼

1. 使用 SMS OTP 成功完成[無密碼身分驗證](#)。
2. 使用 SMS OTP 成功完成 [MFA](#)。

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。請完成 中的下列步驟 AWS Management Console，以設定您的使用者集區，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 登入資料。
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. 選擇註冊功能表，並找到屬性驗證和使用者帳戶確認。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。此程式碼的有效期為 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 主控台中或使用 IAM 驗證 API 操作 [AdminConfirmSignUp](#) 來確認新的使用者帳戶。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。若要以系統管理員身分確認建立的使用者，請使用 `Permanent` 將 [AdminSetUserPassword](#) 請求設定為 `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。您必須登入 AWS Management Console 或使用 AWS 登入資料登入您的 API 請求，以確認帳戶。
 - a. 若要在 Amazon Cognito 主控台中確認使用者，請導覽至使用者功能表，選擇您要確認的使用者，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，請在 中建立 [AdminConfirmSignUp](#) API 請求或 [admin-confirm-sign-up](#) 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的名稱，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 SecretHash 參數的詳細概觀，請參閱 AWS 知識中心的 [「無法驗證用戶端 <client-id>」錯誤的秘密雜湊 Amazon Cognito ?](#)。

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

在您以多個登入名稱設定的使用者集區中，使用者可以在登入時輸入電話號碼或電子郵件地址做為使用者名稱。當他們在您的應用程式中更新其電子郵件地址或電話號碼時，Amazon Cognito 可以立即傳送訊息給他們，其中包含可驗證其擁有新屬性值的代碼。若要啟用這些驗證碼的自動傳送，請參閱 [設定電子郵件或電話驗證](#)。

收到驗證碼的使用者必須在 [VerifyUserAttribute](#) 請求中提供該程式碼回 Amazon Cognito。提供程式碼後，其屬性會標示為已驗證。一般而言，當使用者更新其電子郵件地址或電話號碼時，您會想要先驗證他們是否擁有新值，然後才能使用它來登入和接收訊息。使用者集區具有可設定的選項，可決定使用者是否必須驗證其電子郵件地址或電話號碼的更新。

此選項是使用者集區屬性 `AttributesRequireVerificationBeforeUpdate`。在 [CreateUserPool](#) 或 [UpdateUserPool](#) 請求中設定，或在 Amazon Cognito 主控台的註冊功能表中等待更新時，使用 設定保持原始屬性值作用中。

您的使用者集區如何處理電子郵件地址的更新，且電話號碼已連線至使用者集區的使用者名稱組態。使用者集區使用者名稱可以位於使用者名稱屬性組態中，其中登入名稱為電子郵件地址、電話號碼或兩者。它們也可以位於別名屬性組態中，其中 `username` 屬性是登入名稱，以及電子郵件地址、電話號碼或偏好的使用者名稱作為替代登入名稱。如需詳細資訊，請參閱 [自訂登入屬性](#)。

您也可以使用自訂訊息 Lambda 觸發條件來自訂驗證訊息。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經驗證時，您的應用程式應該通知使用者他們必須驗證 屬性，並提供按鈕或連結供使用者輸入其驗證碼。

下表說明當使用者變更其登入屬性的值時，`AttributesRequireVerificationBeforeUpdate` 和別名設定如何決定結果。

使用者名稱組態	使用者必須驗證新屬性的行為	使用者不需要驗證新屬性的行為
使用者名稱屬性	原始屬性會保持已驗證、符合登入資格，且為原始值。當使用者驗證新值時，Amazon Cognito 會更新屬性值、將其標記為已驗證，並使其符合登入資格。	Amazon Cognito 會將屬性更新為新值。新值符合登入資格。當使用者驗證新值時，Amazon Cognito 會將其標記為已驗證。
別名屬性	原始屬性會保持已驗證、符合登入資格，且為原始值。當使用者驗證新值時，Amazon Cognito 會更新屬性值、將其標記為已驗證，並使其符合登入資格。	Amazon Cognito 會將屬性更新為新值。原始或新的屬性值都不符合登入資格。當使用者驗證新值時，Amazon Cognito 會更新屬性值、將其標記為已驗證，並使其符合登入資格。

範例 1

使用者 1 使用電子郵件地址登入您的應用程式，`user1@example.com` 並具有使用者名稱 `user1` (別名屬性)。您的使用者集區已設定為驗證登入屬性的更新，並自動傳送驗證訊息。他們請求將其電子郵件地址更新為 `user1+foo@example.com`。他們會在 收到驗證電子郵

件user1+foo@example.com，並且只能使用電子郵件地址 再次登入user1@example.com。稍後，他們輸入其驗證碼，並且只能使用電子郵件地址 再次登入user1+foo@example.com。

範例 2

使用者 2 使用電子郵件地址登入您的應用程式，user2@example.com並具有使用者名稱（別名屬性）。您的使用者集區設定為不驗證登入屬性的更新，並自動傳送驗證訊息。他們請求將其電子郵件地址更新為 user2+bar@example.com。他們會在 收到驗證電子郵件user2+bar@example.com，無法再次登入。稍後，他們輸入其驗證碼，並且只能使用電子郵件地址 再次登入user2+bar@example.com。

範例 3

使用者 3 使用電子郵件地址登入您的應用程式，user3@example.com但沒有使用者名稱（使用者名稱屬性）。您的使用者集區設定為不驗證登入屬性的更新，並自動傳送驗證訊息。他們請求將其電子郵件地址更新為 user3+baz@example.com。他們會收到位於 的驗證電子郵件user3+baz@example.com，但可以立即登入，而不需使用驗證碼採取其他動作。

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、CLI AWS Management Console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建立的使用者帳戶已處於確認狀態，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 ForgotPasswordAPI 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 會限制 AWS 帳戶持續發生硬退信的電子郵件數量。

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

您可以在身分驗證方法選單下選擇電子郵件或電話驗證的設定。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。如果您未曾從 Amazon Cognito 或任何其他 傳送 SMS 訊息 AWS 服務，Amazon SNS 可能會將您的帳戶放在 SMS 沙盒中。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 主控台的訊息範本選單中編輯驗證訊息範本時，可以選擇驗證類型的程式碼或連結。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，您也可以利用預先註冊 Lambda 觸發程序或 [AdminConfirmSignUp](#) API 操作來自動確認使用者。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呼叫 [GetUser](#)，查看是否有電子郵件地址正在等待驗證。如果確實需要驗證，應用程式必須呼叫 [GetUserAttributeVerificationCode](#) 以啟動電子郵件驗證流程。然後它必須藉由呼叫 [VerifyUserAttribute](#) 提交驗證碼。

您可以調整 AWS 帳戶 和個別訊息的簡訊花費配額。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 會在 AWS 區域 您建立使用者集區的 中使用 Amazon SNS 資源傳送 SMS 訊息，或在下表的舊版 Amazon SNS 替代區域中傳送簡訊。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 Amazon SNS SMS 訊息 AWS 區域 的](#)。

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)
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 ssue2@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 [AdminUpdateUserAttributes](#) API 請求中將 email_verified 或 phone_number_verified 設定為 true，則他們可以在確認 Amazon Cognito 傳送給他們的代碼之前登入。

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 [AdminUpdateUserAttributes](#) API 請求中將 email_verified 或 phone_number_verified 設定為 true，則他們可以在確認 Amazon Cognito 傳送給他們的代碼之前登入。

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 登入資料。

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. 在註冊功能表中，選擇屬性驗證和使用者的帳戶確認下的編輯。
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 Save changes (儲存變更)。

如欲要求使用 Amazon Cognito API 進行屬性更新驗證，您可以設定 [UpdateUserPool](#) 請求中的 `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。若要授予該許可，您可以建立 AWS Identity and Access Management (IAM) 角色。在 SMS 下的 Amazon Cognito 主控台的身分驗證方法選單中，選擇編輯以設定角色。

設定驗證和邀請訊息

使用 Amazon Cognito，您可以自訂簡訊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，您可以選擇程式碼型或一鍵式連結驗證，以符合應用程式的需求。本主題討論如何在 Amazon Cognito 主控台中個人化多重要素驗證 (MFA) 和驗證通訊。

在訊息範本選單中，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

Note

當使用者註冊或重新傳送確認碼時，Amazon Cognito 會在驗證訊息中傳送連結與您的連結型範本。屬性更新和密碼重設操作的電子郵件會使用程式碼範本。

- 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

簡訊和電子郵件驗證訊息範本只會在您選擇要求電話號碼和電子郵件驗證時顯示。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預留位置插入訊息。Amazon Cognito 會將預留位置取代為對應的值。您可以在任何類型的訊息範本中參考通用範本預留位置，雖然這些值不會出現在所有訊息類型中。

通用範本預留位置

描述	權杖	訊息類型
驗證碼	{####}	驗證、確認和 MFA 訊息
臨時密碼	{####}	忘記密碼和邀請訊息
使用者名稱	{username}	邀請和進階安全訊息

威脅**防護**的其中一個可用自動化回應是通知使用者 Amazon Cognito 偵測到潛在惡意活動。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。

進階安全功能會新增下列預留位置，您可以將這些預留位置插入訊息範本：

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}
意見回饋權杖	{feedback-token}

自訂 SMS 訊息

若要自訂多重要素驗證 (MFA) 的 SMS 訊息，請從 Amazon Cognito 使用者集區主控台內的訊息範本選單編輯 MFA 訊息。

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 會為 SMS 訊息設定 140 個 UTF-8 字元的長度上限，包括驗證碼。

自訂 SMS 驗證訊息

若要自訂電話號碼驗證的簡訊，請從使用者集區的訊息範本選單編輯驗證訊息範本。

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訊息內容，請在使用者集區的訊息範本選單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

您可以在訊息範本選單中編輯邀請訊息範本，自訂 Amazon Cognito 透過簡訊或電子郵件訊息傳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. 選擇身分驗證方法選單。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)。若要使用已驗證網域的電子郵件地址，請在 AWS AWS Command Line Interface 或 API 中設定電子郵件設定。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": "stmt1234567891234",
      "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

使用者集區不只是客戶身分和存取管理 (CIAM) 使用者目錄，網際網路上的任何人都可以在應用程式中註冊使用者設定檔。您可以停用自助式註冊。您可能已經認識您的客戶，並只想認可已事先獲得授權的客戶。您可以透過[私有 SAML 2.0 或 OIDC 身分提供者](#)、[匯入使用者](#)、在[註冊時篩選使用者](#)，或透過使用管理 API 操作建立使用者，在應用程式周圍放置手動身分驗證護欄。管理建立使用者的工作流程可以是程式設計的、在使用者註冊到另一個系統後佈建使用者，也可以是 Amazon Cognito 主控台中的 case-by-case 或測試。

當您以管理員身分建立使用者時，Amazon Cognito 會為其設定臨時密碼，並傳送歡迎或邀請訊息。他們可以遵循邀請訊息中的連結，第一次登入、設定密碼並確認其帳戶。以下頁面說明如何建立新的使用者並設定歡迎訊息。如需使用使用者集區 API 和 AWS SDK 或 CDK 建立使用者的詳細資訊，請參閱 [AdminCreateUser](#)。

建立使用者集區之後，您可以使用 AWS Management Console 以及 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。

以下是管理員如何在使用者集區中管理使用者的一些範例。

- 在 Amazon Cognito 主控台或使用 AdminCreateUser API 操作建立新的使用者設定檔。
- 將 username-and-password、無密碼、通行金鑰和自訂 [身分驗證流程](#) 提供給使用者集區和應用程式用戶端。
- 設定使用者屬性值。
- 建立自訂屬性。
- 在 AdminCreateUser API 請求中設定不可變 [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、建立沒有密碼的使用者，或允許 Amazon Cognito 自動產生密碼。
- 建立新的使用者並自動確認其帳戶、驗證其電子郵件地址，或驗證其電話號碼。
- 透過自訂訊息、自訂簡訊寄件者和自訂電子郵件寄件者等 AWS Management Console 或 Lambda 觸發條件，為新使用者指定自訂簡訊和電子郵件邀請[訊息](#)。[??? ???](#)
- 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。
- 在建立使用者時，[隱藏](#)傳送邀請訊息。
- 為新使用者帳戶指定最長 90 天的過期時間限制。
- 允許使用者自行登入，或要求僅限由管理員新增新使用者。

管理員也可以使用伺服器端應用程式中的 AWS 登入資料登入使用者。如需詳細資訊，請參閱[API 和 SDK 身分驗證的授權模型](#)。

使用者身分驗證流程和建立使用者

使用者管理建立的選項會根據使用者集區的組態而有所不同。身分驗證流程或可供使用者用於登入和 MFA 的方法，可以變更您建立使用者的方式，以及您傳送給他們的訊息。以下是一些可在使用者集區中使用的身分驗證流程。

- 使用者名稱和密碼
- 通行密鑰
- 使用第三方 IdPs 登入
- 使用電子郵件和簡訊一次性密碼 (OTPs 無密碼)
- 使用電子郵件、SMS 和驗證程式應用程式 OTPs 多重驗證
- 使用 Lambda 觸發器進行自訂身分驗證

如需如何設定這些登入因素的詳細資訊，請參閱 [使用 Amazon Cognito 使用者集區進行身分驗證](#)。

建立不含密碼的使用者

如果您已啟用使用者集區的無密碼登入，則可以建立沒有密碼的使用者。若要建立沒有密碼的使用者，您必須為可用的無密碼登入因素提供屬性值。例如，如果您的使用者集區中有電子郵件 OTP 無密碼登入，您可以建立沒有密碼和電子郵件地址屬性的使用者。如果只有新使用者可用的身分驗證流程需要密碼，例如密碼金鑰或使用者名稱密碼，您必須為每個新使用者建立或產生臨時密碼。

建立沒有密碼的新使用者

- 選擇不要在 Amazon Cognito 主控台中設定密碼 Amazon Cognito
- 省略或保留空白 AdminCreateUser API 請求的 TemporaryPassword 參數

沒有密碼的使用者會自動確認

一般而言，新使用者會取得臨時密碼，並在您建立時進入 `FORCE_CHANGE_PASSWORD` 狀態。當您建立沒有密碼的使用者時，使用者會立即進入 `CONFIRMED` 狀態。您無法重新傳送確認碼給 `CONFIRMED` 狀態的這些使用者。

沒有密碼的使用者的邀請訊息變更。

根據預設，Amazon Cognito 會傳送[邀請訊息](#)給新使用者，該訊息會指出當您建立沒有密碼的使用者 `Your username is {userName} and your password is {####}` 時，訊息會指出 `Your username is {userName}`。自訂您的邀請訊息，以反映您是否將設定使用者的密碼。省略無 `{####}` 密碼身分驗證模型中的密碼變數。

當無密碼因素可用時，您無法自動產生密碼

如果您已將使用者集區設定為支援電子郵件或電話 OTP 無密碼登入，則無法自動產生密碼。對於每個將擁有密碼的使用者，您必須在建立其設定檔時設定臨時密碼。

無密碼使用者必須具有所有必要屬性的值

當您在沒有密碼的情況下建立使用者時，只有在使用者為使用者集區中標記為必要的所有屬性提供值時，您的請求才會成功。這適用於任何必要的屬性，而不只是 OTP 交付所需的電話號碼和電子郵件屬性。

建立稍後將提供必要屬性值的使用者

您可能想要在使用者集區中要求屬性，但在管理建立使用者之後，在應用程式中的使用者互動期間收集這些屬性。當管理員使用臨時密碼建立使用者時，可以省略必要屬性的值。您無法省略無密碼使用者的必要屬性值。

缺少必要屬性值和臨時密碼的使用者，會在第一次登入時收到 [NEW_PASSWORD_REQUIRED](#) 挑戰。然後，他們可以在 `requiredAttributes` 參數中提供缺少必要屬性的值。只有在所有必要屬性都可[變](#)時，您才能建立具有密碼且沒有必要屬性的使用者。使用者只能從其登入的應用程式用戶端[寫入](#)必要的屬性，才能完成具有 `NEW_PASSWORD_REQUIRED` 挑戰和必要屬性值的登入。

當您為管理員建立的使用者設定永久密碼時，其狀態會變更為 `CONFIRMED` 而且您的使用者集區不會在第一次登入時提示他們輸入新密碼或必要的屬性。

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. 選擇身分驗證方法選單，並尋找密碼政策。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. 選擇註冊功能表並尋找自助註冊。選擇 Edit (編輯)。
4. 選擇是否要 Enable self-registration (啟用自我註冊)。自我註冊通常用於需要在您的使用者集區中註冊新使用者的公有應用程式用戶端，而無需分發用戶端秘密或 AWS Identity and Access Management (IAM) API 登入資料。

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. 選擇訊息範本選單，然後選擇驗證訊息、邀請訊息或 MFA 訊息，然後選擇編輯。
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。若要確認帳戶為管理員，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設定檔。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. 選擇使用者功能表，然後選擇建立使用者。
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。若要隱藏邀請訊息，請選擇不傳送邀請。

Note

在傳送邀請訊息之前，請在使用者集區的身分驗證方法選單中，使用 Amazon Simple Notification Service 和 Amazon Simple Email Service 設定寄件 AWS 區域 者和 。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)。如果使用者集區中有[無密碼登入](#)可用，則無法使用產生密碼的選項。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. 選擇使用者選單，然後選擇使用者的使用者名稱項目。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。

Amazon Cognito 會為您新增至使用者集區的每個 OIDC、SAML 和社交[身分提供者](#) (IdP) 建立使用者群組。群組的名稱格式為 [user pool ID]_[IdP name]，例如 us-east-1_EXAMPLE_MYSSO 或 us-east-1_EXAMPLE_Google。每個唯一自動產生的 IdP 使用者設定檔都會自動新增至此群組。[連結的使用者](#)不會自動新增至此群組，但您可以個別程序將其設定檔新增至群組。

您可以從 AWS Management Console、APIs 和 CLI 在使用者集區中建立和管理群組。身為開發人員（使用 AWS 登入資料），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。
- 您的應用程式在透過身分集區為使用者請求 AWS 登入資料時，也可以在 [GetCredentialsForIdentity](#) CustomRoleARN 參數指定角色 ARN，以從可用的角色中選擇。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量受限於 [Amazon Cognito 服務配額](#)。
- 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 登入資料。
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. 選擇群組功能表，然後選擇建立群組。
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

使用者集區可以包含數百萬使用者。使用此大小的資料集對管理員來說是一項挑戰。Amazon Cognito 具有尋找和修改使用者設定檔的工具。尋找使用者的首要方法是 Amazon Cognito 主控台的使用者功能表，以及 [ListUsers](#)。在擷取使用者相關資訊的方法中，這些選項不會影響成本，例如 [AdminGetUser](#)。

本指南的本節提供在使用者集區中尋找和更新使用者設定檔的相關資訊。

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](#)。如果出現提示，請輸入您的 AWS 登入資料。
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. 選擇使用者選單，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](#)。如果出現提示，請輸入您的 AWS 登入資料。
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. 選擇使用者選單，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。

啟用、停用和刪除使用者帳戶

您可以刪除未使用的使用者設定檔，或者，如果您想要暫時防止存取，請停用它們。使用者可以刪除自己的帳戶，但只有使用者集區管理員才能啟用和停用使用者帳戶。

刪除的影響

使用者無法使用已刪除的使用者帳戶登入，若要重新取得存取權，必須註冊或再次建立。

停用帳戶的影響

當您停用使用者帳戶時，Amazon Cognito 會自動使所有已驗證的工作階段失效、停用使用者帳戶以進行登入，以及[撤銷其存取權和重新整理權杖](#)。當使用者嘗試登入您停用的帳戶User is not enabled時，Amazon Cognito 會傳回包含訊息的invalid_request錯誤。此行為不會隨著應用程式用戶端的[使用者存在揭露設定](#)而變更。您可以停用本機使用者帳戶和聯合使用者帳戶的本機設定檔。當使用者使用受管登入或傳統託管 UI 登入時，您會停用其帳戶，然後嘗試使用維護其已驗證工作階段的瀏覽器 Cookie 再次登入時，Amazon Cognito 會將他們重新導向至登入頁面。

啟用帳戶的效果

啟用帳戶後，使用者可以立即登入帳戶。使用者帳戶預設為啟用。使用者的屬性和密碼與其帳戶停用前保持不變。無論您停用使用者帳戶或分別撤銷重新整理權杖，應用程式撤銷的權杖都會在您啟用擁有權杖的使用者帳戶後保持無效。

Delete a user account (console)

刪除使用者帳戶

1. 前往 [Amazon Cognito 主控台](#)。如果出現提示，請輸入您的 AWS 登入資料。
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. 選擇使用者選單，然後選取清單中使用者使用者名稱旁的選項按鈕。
5. 選擇 刪除。
6. 選擇停用使用者存取。

7. 選擇刪除。

Delete a user account (API)

使用者可以使用自助access-token-authorized [DeleteUser](#) API 操作來刪除其帳戶。以下是擴充DeleteUser請求內文。

```
{
  "AccessToken": "eyJra456defEXAMPLE"
}
```

管理員可以使用 IAM 授權的 [AdminDeleteUser](#) API 操作刪除使用者帳戶。以下是擴充AdminDeleteUser請求內文。

```
{
  "Username": "testuser",
  "UserPoolId": "us-west-2_EXAMPLE"
}
```

Disable a user account (console)

停用使用者帳戶

1. 前往 [Amazon Cognito 主控台](#)。如果出現提示，請輸入您的 AWS 登入資料。
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. 選擇使用者選單，然後在清單中選取使用者的使用者名稱。
5. 在使用者詳細資訊頁面上，選擇動作、停用使用者存取。
6. 在此建立的對話方塊中，選擇停用。

Disable a user account (API)

管理員可以使用 IAM 授權 [AdminDisableUser](#) API 操作停用使用者帳戶。以下是擴充AdminDisableUser請求內文。

```
{
  "Username": "testuser",
  "UserPoolId": "us-west-2_EXAMPLE"
}
```

```
}
```

Enable a user account (console)

啟用使用者帳戶

1. 前往 [Amazon Cognito 主控台](#)。如果出現提示，請輸入您的 AWS 登入資料。
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. 選擇使用者選單，然後在清單中選取使用者的使用者名稱。
5. 在使用者詳細資訊頁面上，選擇動作、啟用使用者存取。
6. 在此建立的對話方塊中，選擇啟用。

Enable a user account (API)

管理員可以使用 IAM 授權的 [AdminEnableUser](#) API 操作啟用使用者帳戶。以下是擴充AdminEnableUser請求內文。

```
{
  "Username": "testuser",
  "UserPoolId": "us-west-2_EXAMPLE"
}
```

搜尋使用者屬性

如果您已經建立使用者集區，即可從 中的 Users AWS 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 搜尋使用者 AWS Management Console

如果您已經建立使用者集區，即可從 中的 Users AWS Management Console(使用者) 面板進行搜尋。

AWS Management Console 搜尋一律是字首 (「開頭」) 搜尋。

在 Amazon Cognito 主控台中搜尋使用者

1. 前往 [Amazon Cognito 主控台](#)。系統可能會提示您輸入 AWS 登入資料。
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. 選擇使用者選單，然後在搜尋欄位中輸入使用者名稱。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"
}
```

密碼、帳戶復原和密碼政策

登入使用者集區的所有使用者，甚至是[聯合身分使用者](#)，都會將密碼指派給其使用者設定檔。[本機使用者](#)和[連結使用者](#)在登入時必須提供密碼。聯合身分使用者不會使用使用者集區密碼，而是使用其身分提供者 (IdP) 登入。您可以允許使用者重設自己的密碼、以管理員身分重設或變更密碼，以及[設定密碼複雜性和歷史記錄的政策](#)。

Amazon Cognito 不會以純文字儲存使用者密碼。反之，它會以使用者特定的 salt 儲存每個使用者密碼的雜湊。因此，您無法從使用者集區中的使用者設定檔擷取現有的密碼。根據最佳實務，請勿將純文字使用者密碼存放在任何地方。當使用者忘記密碼時，請執行密碼重設。

密碼重設和復原

使用者忘記密碼。您可能希望他們能夠自行重設密碼，或者您可能想要要求管理員為其重設密碼。Amazon Cognito 使用者集區具有兩種模型的選項。本指南的此部分涵蓋使用者集區設定和用於密碼重設的 API 操作。

[ForgotPassword](#) API 操作和受管登入選項 忘記您的密碼？會傳送程式碼給使用者，當他們確認擁有正確的程式碼時，讓他們有機會使用 [ConfirmForgotPassword](#) 設定新密碼。這是自助式密碼復原模型。

復原未驗證的使用者

您可以傳送復原訊息給已驗證其電子郵件地址或電話號碼的使用者。如果他們沒有已確認的復原電子郵件或電話，使用者集區管理員可以將其電子郵件地址或電話號碼標記為已驗證。在 Amazon Cognito 主控台中編輯使用者屬性，然後選取將電話號碼標記為已驗證或將電子郵件地址標記為已驗證旁的核取方塊。您也可以[在 AdminUpdateUserAttributes 請求中將 email_verified 或 phone_number_verified 設為 true](#)。對於新使用者，[ResendConfirmationCode](#) API 操作會將新代碼傳送到其電子郵件地址或電話號碼，他們可以完成自助式確認和驗證。

以管理員身分重設密碼

[AdminSetUserPassword](#) 和 [AdminResetUserPassword](#) API 操作是管理員啟動的密碼重設方法。會 [AdminSetUserPassword](#) 設定暫時或永久密碼，並以與 [AdminResetUserPassword](#) 傳送密碼重設碼給使用者 [ForgotPassword](#)。

設定密碼重設和復原

Amazon Cognito 會自動從您在主控台中建立使用者集區時所選擇的必要屬性和登入選項中選取您的帳戶復原選項。您可以修改這些預設設定。

使用者偏好的 MFA 方法會影響他們可以用來復原密碼的方法。偏好的 MFA 為電子郵件訊息的使用者無法透過電子郵件接收密碼重設碼。偏好的 MFA 是 SMS 訊息的使用者無法透過 SMS 接收密碼重設碼。

當使用者不符合您偏好的密碼重設方法時，[您的密碼復原](#)設定必須提供替代選項。例如，您的復原機制可能有電子郵件做為首要任務，而電子郵件 MFA 可能是使用者集區中的選項。在此情況下，請新增簡訊帳戶復原做為第二個選項，或使用管理 API 操作來重設這些使用者的密碼。

Note

使用者無法在相同的電子郵件地址或電話號碼接收 MFA 和密碼重設代碼。如果他們使用來自 MFA 電子郵件訊息的一次性密碼 (OTPs)，則必須使用 SMS 訊息進行帳戶復原。如果他們使用來自 MFA 簡訊的 OTPs，則必須使用電子郵件訊息進行帳戶復原。在具有 MFA 的使用者集區中，如果使用者擁有電子郵件地址的屬性，但沒有電話號碼，或電話號碼沒有電子郵件地址，則可能無法完成自助式密碼復原。

若要防止使用者無法使用此組態在使用者集區中重設密碼的狀態，請視需要設定 email 和 phone_number 屬性。[???或者](#)，您可以設定程序，在使用者註冊或管理員建立使用者設定檔時，一律收集和設定這些屬性。當使用者具有這兩個屬性時，Amazon Cognito 會自動將密碼重設代碼傳送至不是使用者 MFA 因素的目的地。

下列程序會在使用者集區中設定自助式帳戶復原。

Configure self-service password reset (API/SDK)

AccountRecoverySetting 參數是使用者集區參數，可設定使用者在 [ForgotPassword](#) API 請求中或在受管登入中選取忘記密碼？時可用來復原密碼的方法。ForgotPassword 會將復原碼傳送至已驗證的電子郵件或已驗證的電話號碼。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程式碼交付目的地。

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。此設定的優先順序取決於1最高優先順序。Amazon Cognito 只會將驗證傳送至其中一個指定的方法。下列範例 AccountRecoverySetting 會將電子郵件地址設定為帳戶復原碼的主要目的地，如果使用者沒有電子郵件地址屬性，則會回到簡訊。

```
"AccountRecoverySetting": {
  "RecoveryMechanisms": [
    {
      "Name": "verified_email",
```

```

        "Priority": 1
      },
      {
        "Name": "verified_phone_number",
        "Priority": 2
      }
    ]
  }
}

```

該值 `admin_only` 會關閉自助式帳戶復原，而不是要求使用者聯絡其管理員重設密碼。您無法搭配任何其他帳戶復原機制使用 `admin_only`。下列 e

```

"AccountRecoverySetting": {
  "RecoveryMechanisms": [
    {
      "Name": "admin_only",
      "Priority": 1
    }
  ]
}

```

如果您未指定 `AccountRecoverySetting`，Amazon Cognito 會先將復原碼傳送至已驗證的電話號碼，如果使用者沒有電話號碼屬性，則會傳送至已驗證的電子郵件地址。

如需的詳細資訊 `AccountRecoverySetting`，請參閱 [CreateUserPool](#) 和 [UpdateUserPool](#)。

Configure self-service password reset (console)

從使用者集區的登入功能表設定帳戶復原和密碼重設選項。

設定使用者帳戶復原

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. 選擇登入選單。尋找使用者帳戶復原，然後選擇編輯
5. 若要允許使用者重設自己的密碼，請選擇啟用自助式帳戶復原。
6. 為使用者集區傳送給使用者的密碼復原碼設定交付方法。在使用者帳戶復原訊息的交付方法下，選取可用的選項。最佳實務是，選擇具有次要傳送訊息方法的選項，例如電子郵件，如果可用，否則為簡訊。透過次要交付方法，Amazon Cognito 可以要求使用者使用不同的媒體進行密碼重設，而非 MFA。

7. 選取儲存變更。

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。

新增使用者集區密碼要求

強式、複雜的密碼是使用者集區的安全最佳實務。特別是在開放給網際網路的應用程式，較弱的密碼可能會將使用者的登入資料公開給猜測密碼並嘗試存取資料的系統。密碼越複雜，猜測的難度就越高。Amazon Cognito 為具有安全意識的管理員提供額外的工具，例如[威脅防護](#)和 [AWS WAF Web ACLs](#)，但您的密碼政策是使用者目錄安全的核心元素。

Amazon Cognito 使用者集區中本機使用者的密碼不會自動過期。最佳實務是在外部系統中記錄使用者密碼重設的時間、日期和中繼資料。透過密碼存留期的外部日誌，您的應用程式或 Lambda 觸發程序可以查詢使用者的密碼存留期，並在指定期間後要求重設。

您可以將使用者集區設定為需要符合您安全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。它們也包含大寫、數字和特殊字元的組合。

透過 Essentials 或 Plus 功能層，您也可以設定密碼重複使用的政策。您可以防止使用者將其密碼重設為符合其目前密碼或最多 23 個額外先前密碼的新密碼，最多總共 24 個。

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需求步驟，或存取現有的使用者集區並導覽至身分驗證方法選單。
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。在 Amazon Cognito 驗證密碼是否包含最低必要字元之後，您使用者的密碼可以包含任何類型的額外字元，最長可達密碼長度上限。

- 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 密碼的持續時間，請在 [CreateUserPool](#) 或 [UpdateUserPool](#) API 請求中設定 [TemporaryPasswordValidityDays](#) 的值。
 7. 如果可用，請設定防止使用先前密碼的值。若要使用此功能，請在使用者集區中選擇 Essentials 或 Plus [功能層](#)。此參數的值是當使用者重設密碼時，無法比對新密碼的先前密碼數目。

若要重設過期使用者帳戶的存取權，請執行下列其中一項操作：

- 刪除使用者描述檔並建立新的描述檔。
- 在 [AdminSetUserPassword](#) API 請求中設定新的永久密碼。
- 在 [AdminResetUserPassword](#) API 請求中產生新的確認碼。

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描述檔內容的 CSV 檔來大量遷移使用者。下列章節將說明這些方法。

其他 資源

- [將使用者遷移至 Amazon Cognito 使用者集區的方法](#)
- [AWS re : Inforce 2023 - 遷移至 Amazon Cognito](#)

主題

- [透過使用者遷移 Lambda 觸發程序匯入使用者](#)

- [從 CSV 檔案將使用者匯入使用者集區](#)

透過使用者遷移 Lambda 觸發程序匯入使用者

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. 使用者會開啟您的應用程式，並使用 Amazon Cognito 使用者集區 API 或透過受管登入來登入。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 SDK 建置的自訂登入 UI，您的應用程式必須搭配 `ADMIN_USER_PASSWORD_AUTH` 流程使用 [InitiateAuth](#) `USER_PASSWORD_AUTH` 或 [AdminInitiateAuth](#)。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。您也可以受管登入中使用忘記密碼流程的內建頁面。

其他資源

- [將使用者遷移至 Amazon Cognito 使用者集區的方法](#)

從 CSV 檔案將使用者匯入使用者集區

當您有外部身分存放區，以及為新的本機使用者準備使用者集區的時間時，從逗號分隔值 (CSV) 檔案匯入的大量使用者，對於遷移到 Amazon Cognito 使用者集區而言，可以是低工作量、低成本的選

項。CSV 檔案匯入是下載和填入範本檔案的程序，然後在匯入任務中將檔案交給您的使用者集區。您可以使用 CSV 匯入來快速建立測試使用者。您也可以以程式設計方式將讀取 API 請求填入檔案至外部身分存放區，接著將其詳細資訊和屬性剖析為檔案的寫入操作。

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。您的使用者在使用此方法匯入時處於 RESET_REQUIRED 狀態。

從 CSV 匯入使用者的最低努力方法是在您的使用者集區中啟用[無密碼登入](#)。透過電子郵件地址和電話號碼屬性以及正確的使用者集區組態，使用者可以在匯入任務完成後，立即使用電子郵件或簡訊一次性密碼 (OTPs) 登入。如需詳細資訊，請參閱[需要匯入的使用者重設密碼](#)。

您也可以使用將 Permanent 參數設定為的 [AdminSetUserPassword](#) API 請求來設定使用者的密碼 true。CSV 匯入不會導致使用者集區中的每月計費作用中使用者 MAUs)。不過，密碼重設操作會產生 MAUs。若要在匯入具有密碼的大量使用者時管理成本，而這些使用者可能不會立即處於作用中狀態，請設定您的應用程式，在使用者登入並收到 RESET_REQUIRED 挑戰時提示他們輸入新密碼。

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。

建立使用者匯入任務的步驟

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。

其他 資源

- [Cognito 使用者設定檔匯出參考架構](#)，用於在使用者集區之間匯出使用者帳戶

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 可用來將匯入任務的結果寫入 CloudWatch Logs。

Note

在 Amazon Cognito 主控台中建立匯入任務時，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，並在 <https://console.aws.amazon.com/iam/> : //www. 開啟 IAM 主控台。
2. 為 建立新的 IAM 角色 AWS 服務。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** 取代為使用者集 AWS 區域 區的，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. 選擇信任關係標籤。
5. 選擇編輯信任政策。
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. 選擇更新政策。
8. 請記下 角色 ARN。當您建立匯入任務時，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](#)。系統可能會提示您輸入 AWS 登入資料。
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. 選擇使用者選單。
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[使用使用者屬性](#)。

- 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,phone_number,email_verified
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 () 來建立和執行使用者集區匯入任務AWS CLI。

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](#)。系統可能會提示您輸入 AWS 登入資料。
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. 選擇使用者選單。
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. 在上傳 CSV 下，選擇選擇檔案並連接您準備的 CSV 檔案。
9. 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。

10. 如果您已建立任務但尚未開始，可以稍後再開始。在匯入使用者下的使用者選單中，選擇匯入任務，然後選取開始。您也可以從 AWS SDK 提交 [StartUserImportJob](#) API 請求。
11. 在匯入使用者下的 使用者 選單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：

```
{
  "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。在您執行匯入任務之後，您的使用者集區不會保留匯入檔案中的資訊。完成或過期後，Amazon Cognito 會刪除您上傳的 CSV 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，並在 <https://console.aws.amazon.com/cloudwatch/> 開啟 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

如果您的使用者集區只提供密碼型登入，使用者必須在匯入後重設密碼。第一次登入時，他們可以輸入任何密碼。Amazon Cognito 會向他們發出提示，讓他們在 API 回應中輸入來自您應用程式的登入請求中輸入新密碼。

如果您的使用者集區具有無密碼身分驗證因素，Amazon Cognito 會預設為匯入使用者的身分驗證因素。系統不會提示他們輸入新密碼，而且可以立即使用無密碼電子郵件或 SMS OTP 登入。您也可以提示使用者設定密碼，讓他們可以完成其他登入方法，例如使用者名稱密碼和密碼金鑰。下列條件適用於使用者匯入後的無密碼登入。

1. 您必須匯入具有對應至可用無密碼登入因素之屬性的使用者。如果使用者可以使用電子郵件地址登入，您必須匯入 email 屬性。如果是電話號碼，您必須匯入 phone_number 屬性。如果兩者皆是，請匯入任一屬性的值。
2. 一般而言，使用者匯入 RESET_REQUIRED 的狀態必須重設密碼。如果它們能夠使用無密碼因素登入，Amazon Cognito 會將其狀態設定為 CONFIRMED。

如需無密碼身分驗證的詳細資訊，包括如何設定以及如何在應用程式中建構身分驗證流程，請參閱 [使用 Amazon Cognito 使用者集區進行身分驗證](#)。

下列程序說明匯入 CSV 檔案 RESET_REQUIRED 後，中具有本機使用者的自訂登入機制使用者體驗。如果您的使用者使用受管登入登入，請指示他們選取忘記密碼？選項、提供電子郵件或文字訊息中的代碼，並設定密碼。

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. 您的應用程式發出 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。

使用使用者屬性

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。您也可以在中將自訂屬性新增至使用者集區定義 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。

在將使用者屬性字串值提交至使用者集區之前，請清理這些值的輸入。分析提議使用者屬性值的一種方法是使用 Lambda 觸發程序，例如[預先註冊](#)。

 Note

某些文件和標準將屬性稱為成員。

主題

- [標準屬性](#)
- [使用者名稱和偏好的使用者名稱](#)
- [自訂登入屬性](#)

- [自訂屬性](#)
- [屬性許可和範圍](#)

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：

- name
- family_name
- given_name
- middle_name
- nickname
- preferred_username
- profile
- picture
- website
- gender
- birthdate
- zoneinfo
- locale
- updated_at
- address
- email
- phone_number
- sub

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。

標準屬性具有預先定義的屬性，您可以在 [DescribeUserPool API 回應](#) 的 `SchemaAttributes` 參數中檢視這些屬性。您可以為這些屬性設定自訂值，例如資料類型、可變性和長度限制。若要修改標準屬性，請在 [CreateUserPool Schema 參數](#) 中設定其自訂值。結構描述也是您設定必要屬性的位置。當您在 Amazon Cognito 主控台中建立使用者集區時，無法修改標準屬性的屬性。

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為所需屬性提供值，管理員可以使用 [AdminCreateUser](#) API。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 許可的管理員可以變更使用者的電子郵件地址，並將其標記為已驗證。使用 [AdminUpdateUserAttributes](#) API 或 [admin-update-user-attributes](#) AWS Command Line Interface (AWS CLI) 命令將電子郵件地址標記為已驗證。管理員可以使用此命令將 `email_verified` 屬性變更為 `true`。您也可以在 Amazon Cognito 主控台的使用者選單中編輯使用者，將電子郵件地址標記為已驗證。

值必須是符合 @ 符號和網域標準電子郵件格式 [的有效電子郵件地址字串](#)，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 許可的管理員可以變更使用者的電話號碼，並將其標記為已驗證。使用 [AdminUpdateUserAttributes](#) API 或 [admin-update-user-attributes](#) AWS CLI 命令將電話號碼標記為已驗證。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 Amazon Cognito 主控台的使用者選單中編輯使用者，將電話號碼標記為已驗證。

 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，在確認使用者後，您可以向 [UpdateUserAttributes](#) API 操作請求並新增屬性值。

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. 前往 中的 [Amazon Cognito](#) AWS Management Console。如果主控台提示您，請輸入您的 AWS 登入資料。
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. 選擇註冊功能表。
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。如需 username 屬性的字串限制的相關資訊，請參閱 [SignUp](#) API 請求的 username 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
Users have multiple sign-in attributes	Yes ¹	No ²
Users must verify email address or phone number before they can sign in with it	Yes	No
Sign up users with duplicate email addresses or phone	Yes	No

您的要求	別名屬性	使用者名稱屬性
numbers and prevent UsernameExistsException errors ³		
Can assign the same email address or phone number attribute value to more than one user	Yes ⁴	No

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)

屬性是當使用者有使用者名稱，但也可以使用該屬性登入時的別名。當您想要允許使用者在登入表單的使用者名稱欄位中選擇其使用者名稱和其他屬性值時，請設定別名。username 屬性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 參考中的 [CreateUserPool](#)。

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加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。

Phone number or email address (console)

建立使用者集區時，您必須將電子郵件地址和電話號碼設定為別名屬性。

在 Amazon Cognito 主控台中建立具有使用者名稱別名的使用者集區

1. 前往 AWS Management Console 中的 [Amazon Cognito](#)。如果主控台提示您，請輸入您的 AWS 登入資料。
2. 使用開始使用或建立使用者集區按鈕建立新的使用者集區。
3. 在定義您的應用程式中選擇應用程式設定。
4. 在登入識別碼選項下的設定選項中，選取使用者名稱旁的核取方塊，以及至少一個其他選項：電子郵件和電話號碼。
5. 選擇別名屬性作為註冊的必要屬性。在受管登入註冊表單中，Amazon Cognito 會提示新使用者提供必要屬性的值。
6. 在新增傳回 URL 下，設定受管登入登入後重新導向的應用程式回呼 URL。
7. 選擇 Create (建立)。

Phone number or email address (API/SDK)

使用 [CreateUserPool](#) API 操作建立新的使用者集區。設定 `AliasAttributes` 參數，如下所示。如果您只想要電話號碼別名，您可以移除 `email` 項目，或者如果您只想要電子郵件地址別名，則可以移除 `phone_number` 項目。

```
"AliasAttributes": [  
    "email",  
    "phone_number"  
],
```

Preferred username (API/SDK)

Amazon Cognito 主控台會建立不含 `preferred_username` 做為別名的使用者集區。若要使用別名建立使用者集區，請在 AWS SDK `preferred_username` 中使用 [CreateUserPool](#) API 請求設定使用者集區。若要支援在註冊時建立偏好的使用者名稱屬性，請將 `preferred_username` 設定為必要的屬性。在受管登入註冊表單中，Amazon Cognito 會提示新使用者提供必要屬性的值。您可以在 Amazon Cognito 主控台中將 `preferred_username` 設定為必要屬性，但這不會將其做為別名提供。

將 設定為別名

在 `CreateUserPool` 請求的 `AliasAttributes` 參數中將 `preferred_username` 設定為別名，如下所示。從清單中移除您不想做為別名屬性的任何值。

```
"AliasAttributes": [
  "email",
  "phone_number",
  "preferred_username"
],
```

視需要設定

在受管登入註冊表單中，Amazon Cognito 會提示新使用者提供必要屬性的值。在 [CreateUserPool](#) 請求的 `SchemaAttributes` 參數中 `preferred_username` 視需要設定。

若要將偏好的使用者名稱設定為必要的屬性，請如下所示進行設定。下列範例會修改的預設結構描述 `preferred_username`，以將其設定為必要。其他結構描述參數，例如 `AttributeDataType` (預設為 `string`) 和 `StringAttributeConstraints` (預設為長度 1-99 個字元) 會假設預設值。

```
"Schema": [
  {
    "Name": "preferred_username",
    "Required": true
  }
]
```


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。

當您同時選取電子郵件地址和電話號碼做為使用者名稱屬性時，使用者可以使用一個或另一個登入，即使它們為兩個屬性提供值。登入使用者名稱是根據您在 [SignUp](#) Username 參數中傳遞的值。

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。

Username attributes (console)

下列程序會建立具有電子郵件地址或電話號碼使用者名稱屬性的使用者集區。Amazon Cognito 主控台中使用者名稱屬性的程序差異在於，您也不會將使用者名稱設定為登入屬性。

在 Amazon Cognito 主控台中建立具有使用者名稱屬性的使用者集區

1. 前往 AWS Management Console 中的 [Amazon Cognito](#)。如果主控台提示您，請輸入您的 AWS 登入資料。
2. 使用開始使用或建立使用者集區按鈕建立新的使用者集區。
3. 在定義您的應用程式中選擇應用程式設定。
4. 在登入識別碼選項下的設定選項中，選取您的使用者名稱屬性：電子郵件、電話號碼或兩者。不勾選使用者名稱。
5. 最佳實務是將使用者名稱屬性選取為註冊的必要屬性。在受管登入註冊表單中，Amazon Cognito 會提示新使用者提供必要屬性的值。如果您未視需要設定使用者名稱屬性，Amazon

Cognito 不會提示新使用者為其提供值。在這種情況下，您必須設定應用程式，為每個使用者收集和提交電子郵件地址或電話號碼，然後才能登入。

6. 在新增傳回 URL 下，設定應用程式回呼 URL 以在受管登入後重新導向。
7. 選擇 Create (建立)。

Username attributes (API/SDK)

在 [CreateUserPool](#) 請求中，設定 UsernameAttributes 參數，如下所示。若要僅允許使用電子郵件地址使用者名稱登入，請在此清單中email指定。若要僅允許使用電話號碼使用者名稱登入，請phone_number單獨指定。此參數會將使用者名稱覆寫為登入選項。

```
"UsernameAttributes": [  
  "email",  
  "phone_number"  
],
```

當您設定使用者名稱屬性時，您的 可以提出 [SignUp](#) API 請求，以傳遞 username 參數中的電子郵件地址或電話號碼。以下是具有使用者名稱屬性的程式碼SignUp API 操作行為。

- 如果username字串是有效的電子郵件地址格式，例如 user@example.com，則使用者集區會自動將 username值填入使用者的email屬性。
- 如果username字串是有效的電話號碼格式，例如 +12065551212，則使用者集區會自動以 username值填入使用者的phone_number屬性。
- 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。
- 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。
- SignUp API 會以 [UUID](#) 為您的使用者填入username屬性。此 UUID 的值與使用者身分字符中的 sub 宣告相同。

您可以使用電子郵件地址或電話號碼來取代 [ListUsers](#) 操作以外的所有 APIs 中的使用者名稱。在 ListUsers API 請求中，您可以指定 email或Filter的 phone_number。如果您依 篩選username，則必須提供 UUID 使用者名稱，而不是電子郵件地址或電話號碼。

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。自訂屬性的名稱必須符合 [SchemaAttributeType](#) Name 參數中所述的規則表達式模式。

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。如需更多資訊，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。

您也可以在建立使用者集區時，於 [CreateUserPool](#) 的 SchemaAttributes 屬性中新增開發人員屬性。開發人員屬性帶有 dev: 字首。您只能使用 AWS 登入資料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. 前往 中的 [Amazon Cognito](#) AWS Management Console。如果主控台提示您，請輸入您的 AWS 登入資料。
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. 選擇註冊功能表，然後在自訂屬性區段中，選擇新增自訂屬性。
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 Save changes (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。根據預設，當您建立新的應用程式用戶端時，您會授予應用程式所有標準和自訂屬性的讀取和寫入許可。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。

最佳實務是，在建立應用程式用戶端時指定屬性讀取和寫入許可。授予應用程式用戶端存取您應用程式操作所需的最低使用者屬性集。

Note

[DescribeUserPoolClient](#) 只有在您設定非預設值的應用程式用戶端許可時，才會針對 ReadAttributes 和 WriteAttributes 傳回值。

若要更新屬性許可 (AWS Management Console)

1. 前往 中的 [Amazon Cognito](#) AWS Management Console。如果主控台提示您，請輸入您的 AWS 登入資料。
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. 選擇應用程式用戶端選單，然後從清單中選擇應用程式用戶端。
5. 在屬性許可索引標籤中，選擇編輯。

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。

對於每個應用程式用戶端，您可以將屬性標記為可讀或可寫入。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](#) 請求會在應用程式用戶端宣告中包含存取權杖；Amazon Cognito 只會針對您的應用程式用戶端能夠讀取的屬性傳回值。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。這雖然可行，但並不實用，因為使用者只能登入一次。在第一次登入嘗試之後，Amazon Cognito 會拒絕嘗試，因為映射規則到現在無法寫入的屬性。
- `AdminCreateUser`：您為不可變屬性提供值。

具有範圍的屬性許可

在您使用 AWS SDK 或 CDK、REST API 或 設定的使用者集區中 AWS CLI，您可以使用 OIDC 範圍設定應用程式用戶端讀取或寫入存取權 `oidc:profile`。`oidc:profile` 授予下列標準屬性的讀取或寫入存取權：

- `name`
- `family_name`
- `given_name`
- `middle_name`
- `nickname`

- preferred_username
- profile
- picture
- website
- gender
- birthdate
- zoneinfo
- locale

此清單是 OIDC 標準屬性減去 email、sub、phone_number 和 address，如 [OIDC 規格第 2.4 節](#) 所定義。如需可指派至應用程式用戶端的範圍相關資訊，請參閱 [範圍、M2M 和具有資源伺服器的 APIs](#)。

若要設定您的應用程式用戶端寫入 oidc:profile 範圍內的屬性，請在 [CreateUserPoolClient](#) 或 [UpdateUserPoolClient](#) API 請求中，將 [WriteAttributes](#) 的值設定為，oidc:profile 以及您想要允許應用程式修改的任何其他屬性。同樣地，若要授予這些屬性的讀取存取權，請將 oidc:profile 新增至 [ReadAttributes](#) 的值。

您可以在使用者集區建立之後，變更屬性的許可和範圍。

了解使用者集區 JSON Web 字符 JWTs)

權杖是身分驗證的成品，您的應用程式可用來證明 OIDC 身分驗證並請求存取資源。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 會以 [base64url](#) 編碼字串的形式發出權杖。您可以將任何 Amazon Cognito ID 或存取權杖從 [解碼 base64url](#) 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

- [了解身分 \(ID\) 字符](#)
- [了解存取權杖](#)

- [重新整理權杖](#)
- [使用字符撤銷結束使用者工作階段](#)
- [驗證 JSON Web 字符](#)
- [管理使用者集區字符過期和快取](#)

了解身分 (ID) 字符

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 字符](#)。

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

當您的使用者使用受管登入登入時，Amazon Cognito 會設定 1 小時內有效的工作階段 Cookie。如果您在應用程式中使用受管登入進行身分驗證，並為您的存取和 ID 字符指定少於 1 小時的最短持續時間，您的使用者仍會擁有有效的工作階段，直到 Cookie 過期為止。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 `jwt_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 字符預設承載

這是來自 ID 字符的範例承載。它包含有關已驗證使用者的宣告。如需 OpenID Connect (OIDC) 標準宣告的詳細資訊，請參閱 [OIDC 標準宣告](#) 清單。您可以使用 新增自己的設計宣告 [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符 ([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 [RevokeToken](#) API 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以 為字首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 字符](#)。

了解存取權杖

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。

透過 Essentials 或 Plus [功能計劃](#)，您也可以實作權杖產生前 Lambda 觸發程序，在執行時間將範圍新增至您的存取權杖。如需詳細資訊，請參閱[產生權杖前 Lambda 觸發程序](#)。

具有 openid範圍的使用者存取權杖是向 請求使用者屬性詳細資訊的許可[userInfo 端點](#)。來自userInfo端點的資訊量衍生自存取字符中的其他範圍：例如，profile針對所有使用者資料，email針對其電子郵件地址。

具有 aws.cognito.signin.user.admin範圍的使用者存取權杖是讀取和寫入使用者屬性、列出身分驗證因素、設定多重驗證 (MFA) 偏好設定和管理記住裝置的許可。存取字符授予此範圍之屬性的存取層級，符合您指派給應用程式用戶端的屬性讀取/寫入許可。

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 字符](#)。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 字符，如果您使用受管登入，請勿指定少於一小時的最小值。受管登入會設定一小時有效的瀏覽器 Cookie。如果您將存取權杖持續時間設定為不到一小時，這不會影響受管登入 Cookie 的有效性，且使用者在初次登入後一小時無需額外登入資料即可重新驗證。

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 `jwtks_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](#)。

存取字符預設承載

這是一個存取權杖酬載範例。如需詳細資訊，請參閱 [JWT 宣告](#)。您可以使用 新增自己的設計宣告 [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符 ([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.us-east-1.amazonaws.com/us-east-1_EXAMPLE`

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 [RevokeToken](#) API 作業撤銷使用者權杖時，會參考 origin_jti 宣告。當您撤銷權杖時，Amazon Cognito 不會再驗證具有相同origin_jti值的存取和 ID 權杖。

token_use

權杖的用途。在存取權杖中，其值為 access。

scope

發給登入使用者的 OAuth 2.0 範圍清單。範圍定義權杖提供給userInfo端點上外部 APIs、使用者自助式操作和使用者資料的存取權。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

使用者集區中的使用者名稱。

其他 資源

- [如何在 Amazon Cognito 使用者集區中自訂存取權杖](#)

存取權杖簽章

以 `.well-known/jwks.json` 端點公告的金鑰簽署的存取權杖簽章會驗證權杖標頭和承載的完整性。當您使用存取權杖授權存取外部 APIs 時，請一律設定您的 API 授權方，以針對簽署該權杖的金鑰驗證此簽章。如需詳細資訊，請參閱[驗證 JSON Web 字符](#)。

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。

使用重新整理權杖取得新的存取和身分權杖

Amazon Cognito 會發出重新整理權杖，以使用受管登入授權碼流程和 API 操作或 SDK 方法成功進行身分驗證。重新整理權杖會傳回新的 ID 和存取權杖，以及選擇性的新重新整理權杖。您可以透過下列方式使用重新整理權杖。

GetTokensFromRefreshToken

[GetTokensFromRefreshToken](#) API 操作會從有效的重新整理權杖發出新的 ID 和存取權杖。如果您已啟用重新整理權杖輪換，也會收到新的重新整理權杖。

InitiateAuth 和 AdminInitiateAuth

[AdminInitiateAuth](#) 或 [InitiateAuth](#) API 操作包含 REFRESH_TOKEN_AUTH 身分驗證流程。在此流程中，您會傳遞重新整理權杖，並取得新的 ID 和存取權杖。您無法在啟用 [重新整理權杖輪換](#) 的應用程式用戶端 REFRESH_TOKEN_AUTH 中使用 進行身分驗證。

OAuth 字符端點

具有 [網域](#) 之使用者集區的 [權杖端點](#) 具有 refresh_token 授予類型，可從有效的重新整理權杖發出新的 ID、存取和選擇性（使用 [重新整理權杖輪換](#)）重新整理權杖。

重新整理權杖輪換

透過重新整理字符輪換，您可以選擇設定您的使用者集區使原始重新整理字符失效，並在每次字符重新整理時發出新的重新整理字符。啟用此設定時，所有形式的字符重新整理中的每個成功請求都會傳回新的 ID、存取和重新整理字符。新的重新整理權杖在原始重新整理權杖的剩餘持續時間內有效。您可以設定 [應用程式用戶端](#) 來輪換重新整理權杖或轉移原始重新整理權杖。若要允許短暫重試，您也可以為原始重新整理字符設定最長 60 秒的寬限期。

重新整理字符輪換的須知事項

- 啟用重新整理權杖輪換後，新的宣告會從使用者集區新增至 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 JWTs 的大小。
- 重新整理字符輪換與身分驗證流程不相容 REFRESH_TOKEN_AUTH。若要實作重新整理字符輪換，您必須在應用程式用戶端中停用此身分驗證流程，並設計應用程式以使用 [GetTokensFromRefreshToken](#) API 操作或同等 SDK 方法提交字符重新整理請求。
- 重新整理字符輪換非作用中時，您可以使用 GetTokensFromRefreshToken 或 完成字符重新整理請求 REFRESH_TOKEN_AUTH。

- 當使用者集區中的[裝置記住](#)處於作用中狀態時，您必須在GetTokensFromRefreshToken請求中提供裝置金鑰。如果您的使用者在初始身分驗證請求中沒有應用程式提交的已確認裝置金鑰，Amazon Cognito 會發出新的金鑰。若要重新整理此組態中的權杖，無論您在身分驗證回應中指定了權杖，AuthParameters還是收到了新的權杖，都必須提供裝置金鑰。
- 您可以在GetTokensFromRefreshToken請求中傳遞ClientMetadata至權杖產生前 Lambda 觸發程序。此資料會傳遞至觸發條件的輸入事件，提供您可以在 Lambda 函數的自訂邏輯中使用的其他內容。

作為安全最佳實務，請在應用程式用戶端上啟用重新整理字符輪換。

Enable refresh token rotation (console)

下列程序會為您的應用程式用戶端開啟或關閉重新整理字符輪換。此程序需要現有的應用程式用戶端。若要進一步了解如何建立應用程式用戶端，請參閱 [使用應用程式用戶端的應用程式特定設定](#)。

啟用重新整理字符輪換

1. 前往 [Amazon Cognito 主控台](#)。如果出現提示，請輸入您的 AWS 登入資料。
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. 導覽至應用程式用戶端選單，然後選取現有的應用程式用戶端。
5. 從頁面的應用程式用戶端資訊區段中選取編輯。
6. 在進階安全組態下，找到啟用重新整理字符輪換選項。
7. 若要啟用輪換，請選取核取方塊。若要停用輪換，請取消選取核取方塊。
8. 在重新整理權杖輪換寬限期下，輸入您要設定為延遲的秒數，最多 60 秒，之後才會撤銷輪換的重新整理權杖。

Enable refresh token rotation (API)

在 [CreateUserPoolClient](#) 或 [UpdateUserPoolClient](#) API 請求中設定重新整理權杖輪換。下列部分請求內文會開啟重新整理字符輪換，並將寬限期設定為 10 秒。

```
"RefreshTokenRotation" : {  
  "Feature" : "ENABLED",  
  "RetryGracePeriodSeconds" : 10  
}
```

API 和 SDK 字符重新整理

有兩種方式可以使用重新整理權杖，透過使用者集區 API 取得新的 ID 和存取權杖，取決於重新整理權杖輪換是否作用中。在重新整理字符輪換作用中的應用程式用戶端中，使用 [GetTokensFromRefreshToken](#) API 操作。在未重新整理字符輪換的應用程式用戶端中，使用 [AdminInitiateAuth](#) 或 [InitiateAuth](#) API 操作 REFRESH_TOKEN_AUTH 的流程。

Note

使用者可以使用 [受管登入](#) 或您使用 AWS SDKs 和 Amazon Cognito API 操作建置的自訂應用程式中的使用者集區進行身分驗證。REFRESH_TOKEN_AUTH 流程和 [GetTokensFromRefreshToken](#) 可以為受管登入使用者完成權杖重新整理。自訂應用程式中的字符重新整理不會影響受管登入工作階段。這些工作階段是在瀏覽器 Cookie 中設定，有效期為一小時。[GetTokensFromRefreshToken](#) 回應會發出新的 ID、存取權和選擇性重新整理權杖，但不會續約受管登入工作階段 Cookie。REFRESH_TOKEN_AUTH 在啟用重新整理字符輪換的應用程式用戶端中無法使用。

GetTokensFromRefreshToken

[GetTokensFromRefreshToken](#) 會從您使用重新整理權杖授權的請求傳回新的 ID、存取和重新整理權杖。以下是的範例請求內文 [GetTokensFromRefreshToken](#)。您可以在對此操作的請求中，將用戶端中繼資料提交至 Lambda 觸發程序。

```
{
  "RefreshToken": "eyJjd123abcEXAMPLE",
  "ClientId": "1example23456789",
  "ClientSecret": "myappclientsecret123abc",
  "ClientMetadata": {
    "MyMetadataKey" : "MyMetadataValue"
  },
}
```

AdminInitiateAuth/InitiateAuth

若要在重新整理權杖輪換非作用中時使用重新整理權杖，請使用 [AdminInitiateAuth](#) 或 [InitiateAuth](#) API 操作。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。

以下是使用 `InitiateAuth` 或 `AdminInitiateAuth` API 進行權杖重新整理的範例請求內文。

```
{
  "AuthFlow": "REFRESH_TOKEN_AUTH",
  "ClientId": "1example23456789",
  "UserPoolId": "us-west-2_EXAMPLE",
  "AuthParameters": {
    "REFRESH_TOKEN": "eyJjd123abcEXAMPLE",
    "SECRET_HASH": "kT5acwCVrbD6JexhW3EQwnRSe6fLuPTRkEQ50athqv8="
  }
}
```

OAuth 權杖重新整理

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。

對字符端點的請求可在重新整理字符輪換作用中以及處於非作用中狀態的應用程式用戶端中使用。當重新整理字符輪換作用中時，字符端點會傳回新的重新整理字符。

以下是具有重新整理字符的範例請求。

```
POST /oauth2/token HTTP/1.1
Host: auth.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic ZGpj0Th1M2ppZWRTaTI4M2V1OTI4OmFiY2RLZjAxMjM0NTY3ODkw
Content-Length: **

client_id=1example23456789&grant_type=refresh_token&refresh_token=eyJjd123abcEXAMPLE
```

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[使用字符撤銷結束使用者工作階段](#)。

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。

若要將使用者登出所有目前的登入工作階段，請使用 [GlobalSignOut](#) 或 [AdminUserGlobalSignOut](#) API 請求撤銷其所有字符。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。由於受管登入工作階段 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，而不需要額外的登入資料提示。登出受管登入使用者後，將其重新導向至 [登出端點](#)，Amazon Cognito 會在其中清除其工作階段 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。隨著時間的推移，您的使用者可能會想要取消授權一些已使用重新整理字符保持登入的應用程式。若要從單一工作階段登出您的使用者，請撤銷其重新整理權杖。當您的使用者想要將自己登出所有已驗證的工作階段時，請產生 [GlobalSignOut](#) API 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。

不過，您可以產生 [AdminUserGlobalSignOut](#) API 請求，以您的 AWS 登入資料授權從其所有裝置登出任何使用者。管理員應用程式必須使用 AWS 開發人員登入資料呼叫此 API 操作，並將使用者集區 ID 和使用者的使用者名稱做為參數傳遞。AdminUserGlobalSignOutAPI 可以將使用者集區中的任何使用者登出。

如需您可以使用 AWS 登入資料或使用者的存取權杖來授權之請求的詳細資訊，請參閱 [依授權模型分組的 API 操作清單](#)。

使用字符撤銷結束使用者工作階段

您可以使用下列方法撤銷重新整理權杖和最終使用者工作階段。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。

RevokeToken 操作

[RevokeToken](#) 會撤銷指定重新整理字符的所有存取字符，包括互動式登入的初始存取字符。此操作不會影響任何使用者的其他重新整理字符，或那些其他重新整理字符的 ID 和存取字符子系。

撤銷端點

[撤銷端點](#) 會撤銷指定的重新整理權杖，以及重新整理權杖產生的所有 ID 和存取權杖。此端點也會從互動式登入撤銷初始存取字符。此端點的請求不會影響任何使用者的其他重新整理字符，或那些其他重新整理字符的 ID 和存取字符子系。

GlobalSignOut 操作

[GlobalSignOut](#) 是使用者使用存取權杖授權的自助式操作。此操作會撤銷所有請求使用者的重新整理、ID 和存取權杖。

AdminUserGlobalSignOut 操作

[AdminUserGlobalSignOut](#) 是管理員使用 IAM 登入資料授權的伺服器端操作。此操作會撤銷所有目標使用者的重新整理、ID 和存取權杖。

撤銷權杖的須知事項

- 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。
- 使用者集區 JWTs 會與建立字符時指派的簽章和過期時間獨立。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。
- 建立新使用者集區用戶端時，撤銷權杖功能會預設為啟用。
- 您只能在啟用字符撤銷的應用程式用戶端中撤銷重新整理字符。
- 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。
- 當您在之前啟用的應用程式用戶端中停用字符撤銷時，撤銷的字符不會再次變為作用中。
- 當您[停用使用者帳戶](#)（撤銷重新整理和存取權杖）時，如果您再次啟用使用者帳戶，則撤銷的權杖不會變成作用中。
- 當您使用 AWS Management Console、AWS CLI 或 API 建立新的使用者集區用戶端時 AWS，預設會啟用字符撤銷。

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 AWS CLI 啟用字符撤銷。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`。

若要建立或修改已啟用 Amazon Cognito API 或 AWS SDK 的字符撤銷的應用程式用戶端，請在 [CreateUserPoolClient](#) 或 [UpdateUserPoolClient](#) API 請求中包含下列參數。

```
"EnableTokenRevocation": true
```

若要在 Amazon Cognito 主控台中設定權杖撤銷，請從使用者集區中的應用程式用戶端功能表中選取應用程式用戶端。選取應用程式用戶端資訊中的編輯按鈕，並在進階組態下啟用或停用字串撤銷。

撤銷權杖

您可以使用 [RevokeToken](#) API 請求撤銷重新整理權杖，例如使用 [aws cognito-idp revoke-token](#) CLI 命令。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。

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 字串

JSON Web 字串 (JWTs) 可以輕鬆解碼、讀取和修改。修改後的存取權杖會產生權限提升的風險。修改後的 ID 字串會產生模擬的風險。您的應用程式信任您的使用者集區做為權杖發行者，但如果使用者在傳輸中攔截權杖該怎麼辦？您必須確保您的應用程式收到與 Amazon Cognito 發行相同的字串。

Amazon Cognito 會發出權杖，這些權杖使用 OpenID Connect (OIDC) 規格的一些完整性和機密性功能。使用者集區字串表示過期時間、發行者和數位簽章等物件的有效性。簽章是 . 分隔 JWT 的第三個和最後一個區段，是權杖驗證的關鍵元件。惡意使用者可以修改權杖，但如果您的應用程式擷取公有金

鑰並比較簽章，則不相符。從 OIDC 身分驗證處理 JWTs 的任何應用程式，都必須在每次登入時執行此驗證操作。

在此頁面上，我們會針對 JWTs 的驗證提出一些一般和特定建議。應用程式開發涵蓋各種程式設計語言和平台。由於 Amazon Cognito 實作的 OIDC 足以接近公有規格，因此您選擇的開發人員環境中任何信譽良好的 JWT 程式庫都可以處理您的驗證需求。

這些步驟描述有關驗證使用者集區 JSON web 權杖 (JWT)。

主題

- [先決條件](#)
- [使用 aws-jwt-verify 驗證權杖](#)
- [了解和檢查權杖](#)

先決條件

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 SDKs 可在您的應用程式中提供 Amazon Cognito 使用者集區字符處理和管理工具。AWS Amplify 包含擷取和重新整理 Amazon Cognito 字符的函數。

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](#)
- [使用 AWS SDKs 的 Amazon Cognito 身分提供者程式碼範例](#)
- 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 aws-jwt-verify 驗證權杖

在 Node.js 應用程式中，AWS 建議程式[aws-jwt-verify](#)庫驗證使用者傳遞至應用程式的字符中的參數。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`。

如需可在 Node.js 應用程式或 AWS Lambda 授權方中使用的詳細資訊和範例程式碼，請參閱 GitHub 上的 [aws-jwt-verify](#)。

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。kid 是使用者集區所持有之 2048 位元 RSA 私有簽署金鑰的截斷參考。

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

簽章無法像標頭和承載一樣分解 base64url。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 base64url 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。如果您的使用者向您的應用程式提供 base64url 編碼的 JWT，且其格式不是 [JSON Header].[JSON Payload].[Signature]，則它不是有效的 Amazon Cognito 字符，您可以將其捨棄。

下列範例應用程式會使用 驗證使用者集區字串aws-jwt-verify。

```
// cognito-verify.js
// Usage example: node cognito-verify.js eyJra789ghiEXAMPLE

const { CognitoJwtVerifier } = require('aws-jwt-verify');

// Replace with your Amazon Cognito user pool ID
const userPoolId = 'us-west-2_EXAMPLE';

async function verifyJWT(token) {
  try {
    const verifier = CognitoJwtVerifier.create({
      userPoolId,
      tokenUse: 'access', // or 'id' for ID tokens
      clientId: 'lexample23456789', // Optional, only if you need to verify the token
      audience
    });

    const payload = await verifier.verify(token);
    console.log('Decoded JWT:', payload);
  } catch (err) {
    console.error('Error verifying JWT:', err);
  }
}

// Example usage
if (process.argv.length < 3) {
  console.error('Please provide a JWT token as an argument. ');
  process.exit(1);
}

const MyToken = process.argv[2];
verifyJWT(MyToken);
```

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 解碼使用者集區 JWTs。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。

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. 如果您的存取權杖包含 aws.cognito.signin.user.admin 宣告，請傳送請求至 API (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。

管理使用者集區字符過期和快取

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](#)。
- 從受管登入頁面請求隱含授予。
- 在 Amazon Cognito API 請求 (例如 [InitiateAuth](#)) 中驗證本機使用者。

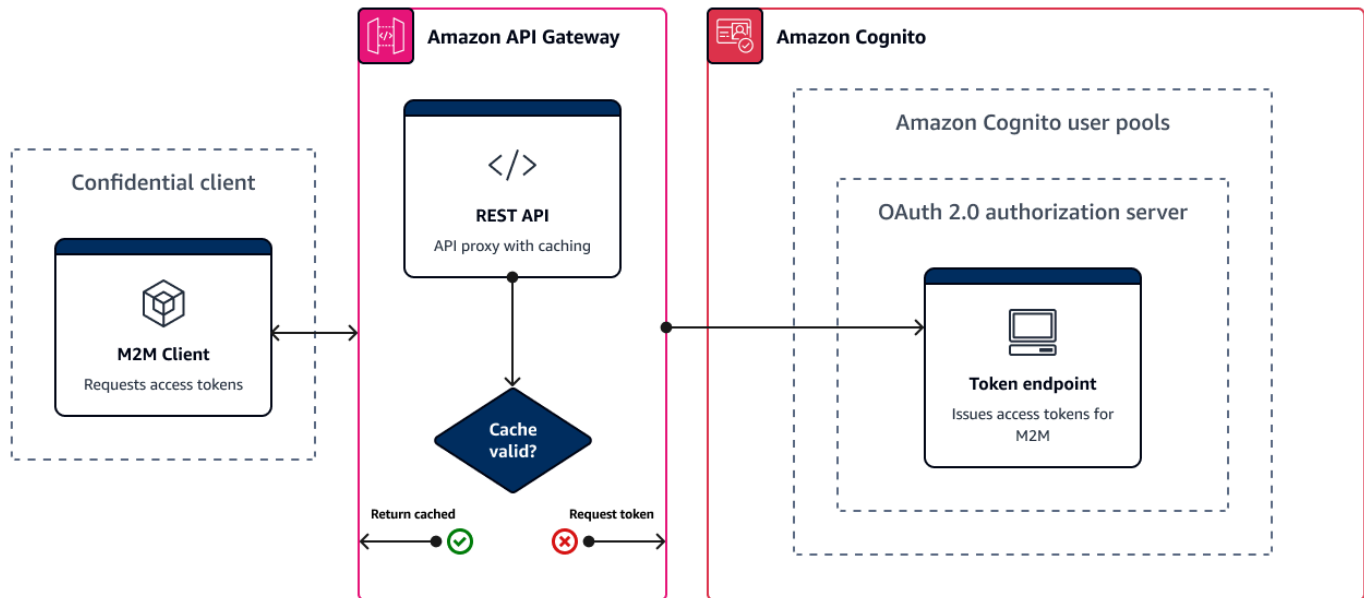
您可以設定使用者集區，將權杖設定為在幾分鐘、幾小時或幾天後過期。為了確保應用程式的效能和可用性，請在字符生命週期的大約 75% 內使用 Amazon Cognito 字符，然後只擷取新的字符。您為應用程式建立的快取解決方案可讓權杖保持可用，並在請求率過高時防止 Amazon Cognito 拒絕請求。用戶端應用程式必須將權杖儲存在記憶體快取中。伺服器端應用程式可以新增加密的快取機制來儲存權杖。

當您的使用者集區產生大量的使用者或機器對機器活動時，您可能會遇到 Amazon Cognito 針對您可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](#) 包含權杖端點，該端點服務用戶端憑證和受管登入授權碼請求。[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 快取機器對機器存取權杖

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 快取是 OAuth 2.0 的 [用戶端憑證授與](#) 的理想選擇，一種頻繁的大容量授與類型，可產生存取權杖以授權機器對機器和微服務工作階段。如果流量激增導致您的微服務水平擴展，您可以在超過使用者集區或應用程式用戶端 AWS 請求速率限制的磁碟區中使用相同的用戶端登入資料，最終會有許多系統。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。

快取金鑰是您在請求內文的 `scope` 參數中請求的 OAuth 範圍和請求中的 `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。

您也可以使用 [ElastiCache \(Redis OSS\)](#) 實作權杖快取。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 查詢字串參數。輸入查詢字串 名稱 `scope`，然後選取必要和快取。
 - ii. 將標頭新增至 HTTP 請求標頭。輸入的請求標頭名稱 `Authorization`，然後選取必要和快取。
4. 在 Stages (階段) 中，設定快取。
 - a. 選擇您要修改的階段，然後選擇從階段詳細資訊中編輯。

- b. 在其他設定、快取設定下，開啟佈建 API 快取選項。
- c. 選擇 Cache capacity (快取容量)。更高的快取容量可改善效能，但需額外付費。
- d. 清除 需要授權 核取方塊。選取繼續。
- e. API Gateway 只會將快取政策套用至階段層級的 GET 方法。您必須將快取政策覆寫套用至 POST 方法。

展開您設定的階段，然後選取 POST 方法。若要為 方法建立快取設定，請選擇建立覆寫。

- f. 啟用啟用方法快取選項。
 - g. 輸入 3600 秒的快取time-to-live (TTL)。選擇儲存。
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。

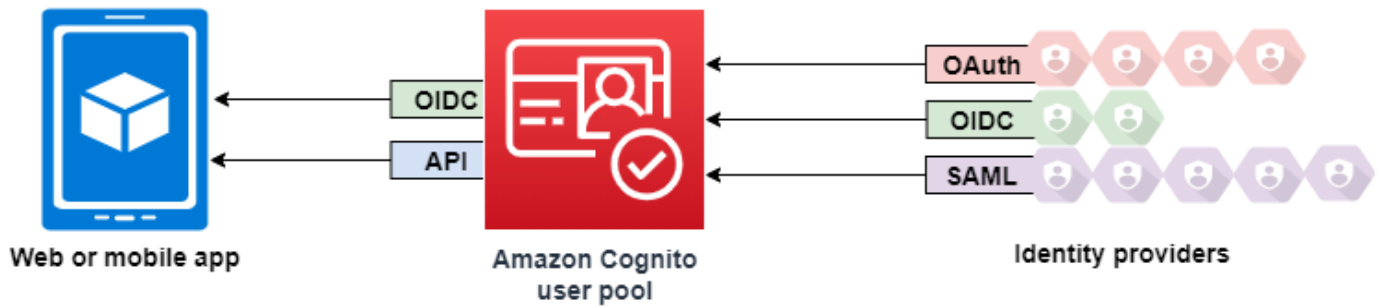
成功登入後存取資源

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提供者 (IdP) 聯合。使用者集區可管理處理字符的開銷，包括透過 Facebook、Google、Amazon 及 Apple 社交登入傳回的字符，以及從 OpenID Connect (OIDC) 及 SAML IdP 傳回的字符。如需詳細資訊，請參閱[了解使用者集區 JSON Web 字符 JWTs](#)。

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。您可以使用使用者集區字符來：

- 擷取 AWS 憑證，以授權 Amazon DynamoDB 和 Amazon S3 AWS 服務 等 中的應用程式資源請求。
- 提供暫時、可撤銷的身分驗證證明。
- 將身分資料填入應用程式中的使用者設定檔。
- 在使用者集區目錄中授權對登入使用者設定檔的變更。
- 使用存取權杖授權使用者資訊的請求。
- 使用存取字符，授權對受存取保護外部 APIs 後方資料的請求。
- 使用 Amazon Verified Permissions 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 [範例身分驗證工作階段](#) 和 [了解使用者集區 JSON Web 字符 JWTs](#)。



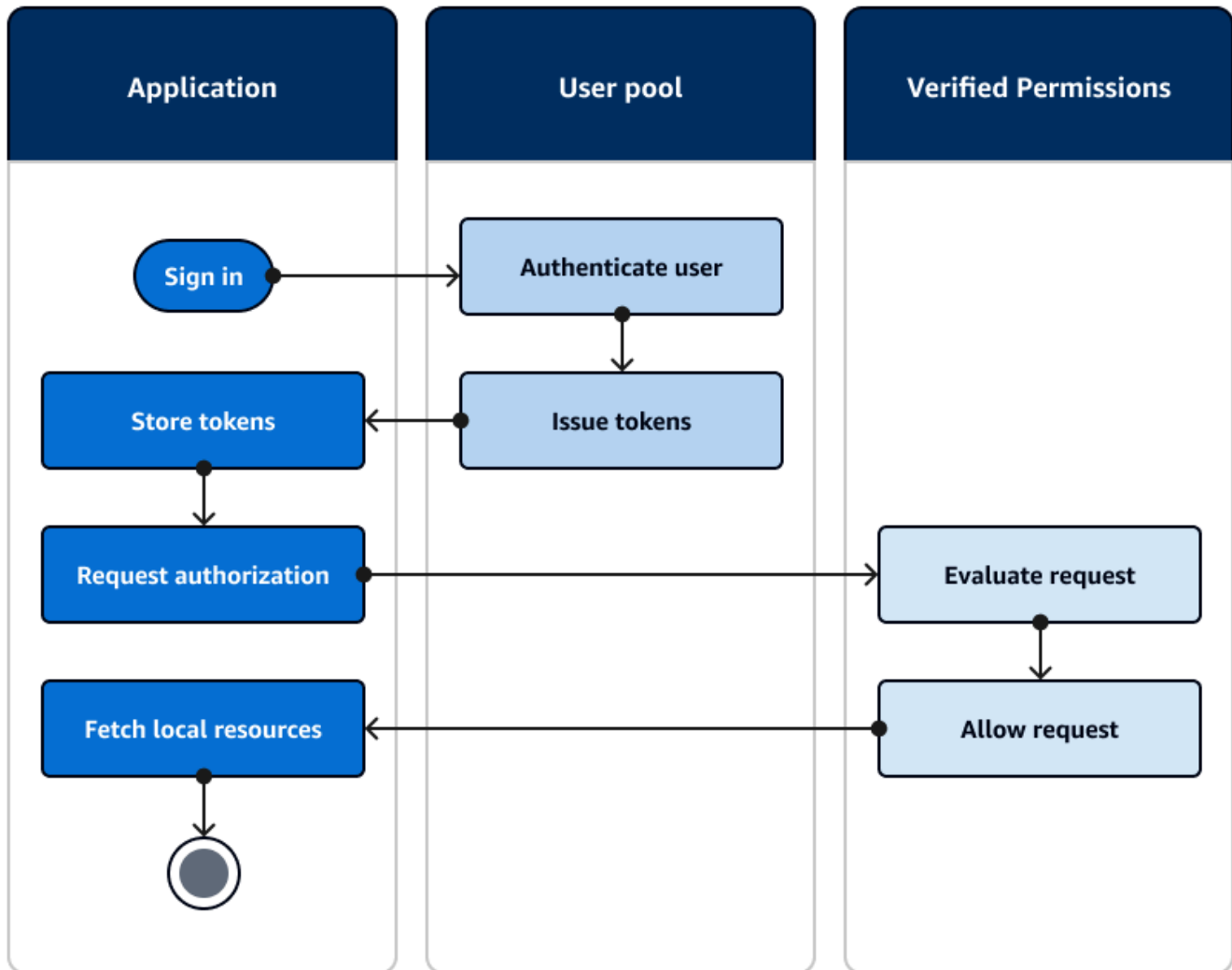
主題

- [使用 Amazon Verified Permissions 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 Verified Permissions 授權存取用戶端或伺服器資源

您的應用程式可以將字符從登入使用者傳遞至 [Amazon Verified Permissions](#)。Verified Permissions 是您建置之應用程式的可擴展性、精細許可管理和授權服務。Amazon Cognito 使用者集區可以是 Verified Permissions 政策存放區的身分來源。Verified Permissions 會從使用者集區權杖中的委託人及其屬性，對請求的動作和資源進行授權決策，例如 `premium_badge.png`，例如 `GetPhoto`。

下圖顯示您的應用程式如何在授權請求中將使用者的權杖傳遞至驗證許可。



開始使用 Amazon Verified Permissions

將使用者集區與 Verified Permissions 整合後，您可以取得所有 Amazon Cognito 應用程式之精細授權的中央來源。這樣就不需要微調安全邏輯，否則您必須在所有應用程式之間進程式碼編寫和複寫。如需使用 Verified Permissions 授權的詳細資訊，請參閱 [使用 Amazon Verified Permissions 進行授權](#)。

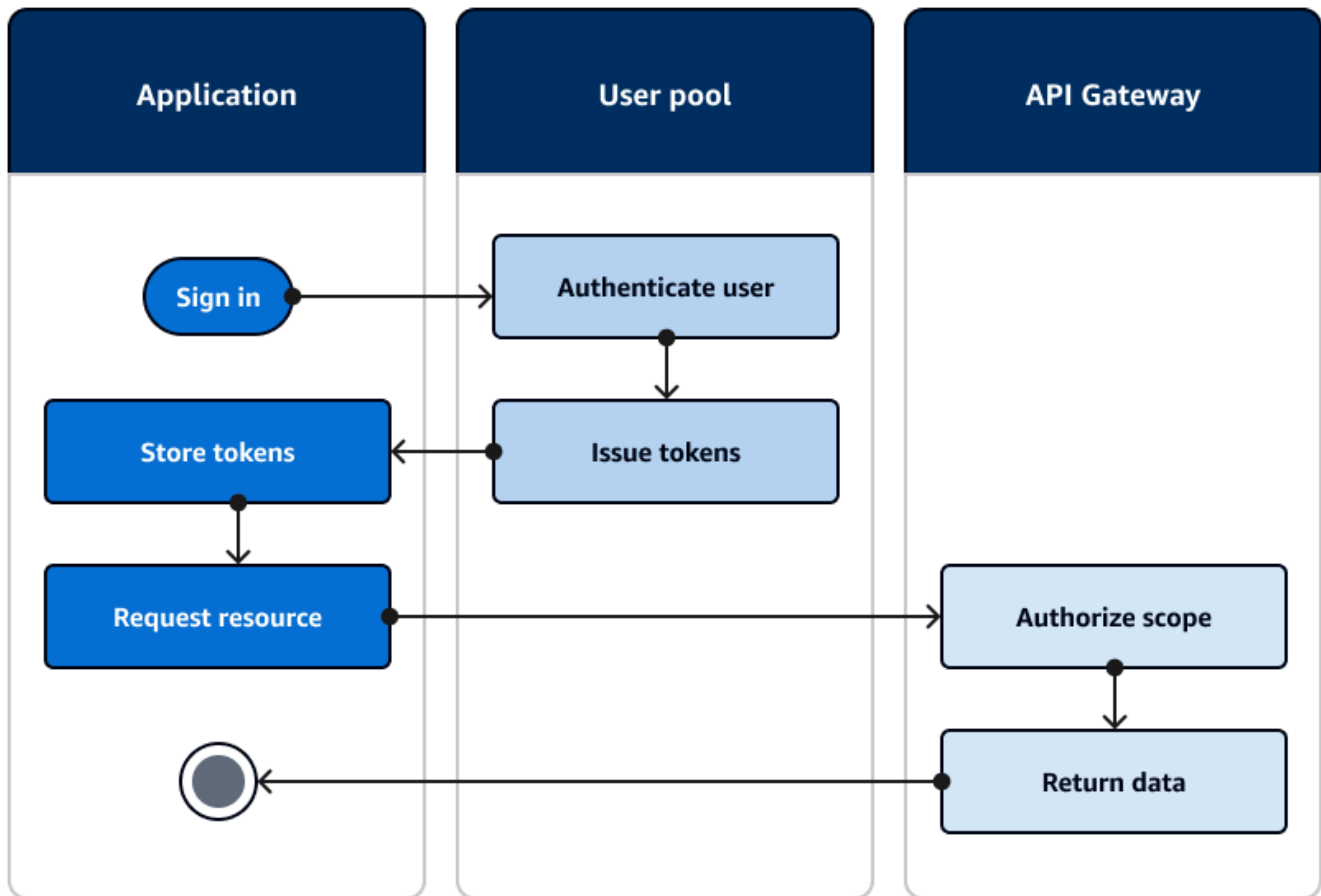
Verified Permissions 授權請求需要 AWS 登入資料。您可以實作下列一些技術，將登入資料安全地套用至授權請求。

- 操作可在伺服器後端存放秘密的 Web 應用程式。
- 取得已驗證的身分集區登入資料。
- 透過 `access-token-authorized` 的代理使用者請求，並將憑證附加 AWS 到請求。

登入後使用 API Gateway 存取資源

Amazon Cognito 使用者集區字符的常見用途是授權對 [API Gateway REST API](#) 的請求。存取字符中的 OAuth 2.0 範圍可以授權方法和路徑，例如 HTTP GET 的 /app_assets。ID 字符可以做為 API 的一般身分驗證，也可以將使用者屬性傳遞給後端服務。API Gateway 還有其他自訂授權選項，例如[適用於 HTTP APIs JWT 授權方](#)，以及可套用更精細邏輯的 [Lambda 授權方](#)。

下圖說明在存取字符中使用 OAuth 2.0 範圍存取 REST API 的應用程式。



您的應用程式必須從已驗證的工作階段中收集字符，並將其新增為承載字符到請求中的 Authorization 標頭。設定您為 API、路徑和方法設定的授權方，以評估權杖內容。API Gateway 只有在請求符合您為授權方設定的條件時，才會傳回資料。

API Gateway API 可以核准從應用程式存取的一些潛在方式如下：

- 存取權杖有效、未過期，且包含正確的 OAuth 2.0 範圍。[REST API 的 Amazon Cognito 使用者集區授權方](#)是一種常見的實作，進入障礙很低。您也可以評估對這類授權方的請求內文、查詢字串參數和標頭。
- ID 字符有效且尚未過期。當您將 ID 字符傳遞給 Amazon Cognito 授權方時，您可以在應用程式伺服器上對 ID 字符內容執行額外的驗證。
- 存取或 ID 字符中的群組、宣告、屬性或角色符合您在 Lambda 函數中定義的需求。[Lambda 授權方](#)會剖析請求標頭中的權杖，並評估權杖是否有授權決策。您可以在函數中建構自訂邏輯，或向[Amazon Verified Permissions](#) 提出 API 請求。

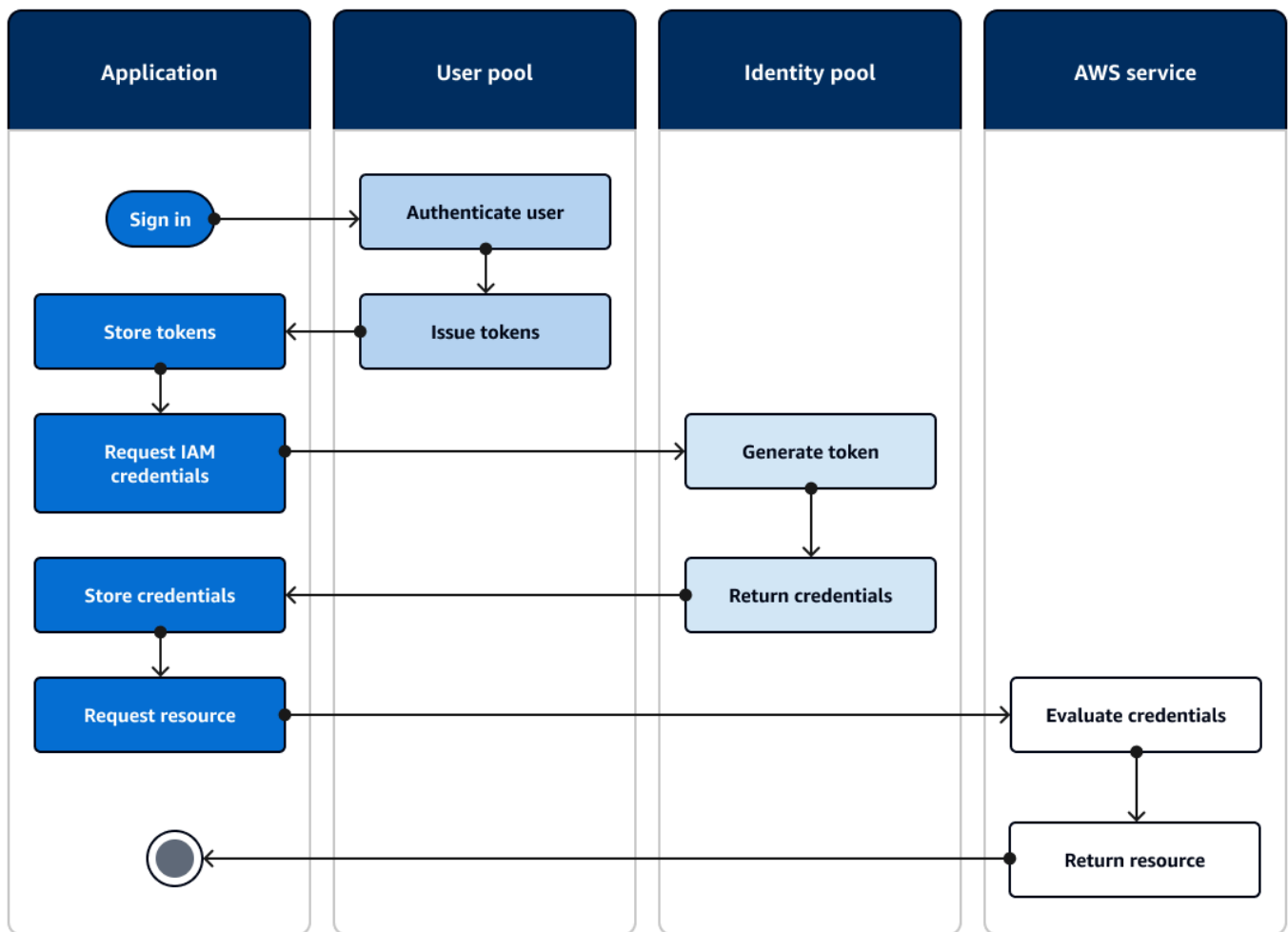
您也可以使用來自使用者集區的字符來授權對 [AWS AppSync GraphQL API](#) 的請求。

登入後 AWS 服務 使用身分集區存取

您的使用者使用使用者集區登入後，他們可以 AWS 服務 使用從身分集區發出的暫時 API 登入資料來存取。

您的 Web 或行動應用程式會從使用者集區接收字符。當您將使用者集區設定為身分集區的身分提供者時，身分集區會交換臨時 AWS 憑證的字符。這些登入資料的範圍可以限定為 IAM 角色及其政策，讓使用者存取一組有限的 AWS 資源。如需詳細資訊，請參閱[身分集區身分驗證流程](#)。

下圖顯示應用程式如何使用使用者集區登入、擷取身分集區登入資料，以及向 請求資產 AWS 服務。



您可以使用身分集區登入資料來：

- 使用使用者自己的登入資料，向 Amazon Verified Permissions 提出精細的授權請求。
- 連線至授權與 IAM 連線的 Amazon API Gateway REST API 或 AWS AppSync GraphQL API。
- 連線至授權與 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。

身分集區不會僅在已驗證的工作階段中搭配使用者集區運作。他們也直接接受來自第三方身分提供者的身分驗證，並且可以為未驗證的訪客使用者產生憑證。

如需使用身分集區與使用者集區群組來控制 AWS 資源存取的詳細資訊，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](#)。此外，如需身分集區和 的詳細資訊 AWS Identity and Access Management，請參閱 [身分集區身分驗證流程](#)。

使用 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](#)。

使用 設定身分集區 AWS Management Console

下列程序說明如何使用 AWS Management Console 將身分集區與一或多個使用者集區和用戶端應用程式整合。

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. 您可以從該 IdP 為使用者提供您在設定已驗證角色時設定的預設角色，或者您可以使用 規則選擇角色。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，請輸入來自使用者身分驗證的來源宣告、您要用來將宣告與規則比較的運算子、將導致與此角色選擇相符的值，以及當角色指派相符時您要指派的角色。選取 新增另一項 以根據不同的條件建立其他規則。
 - ii. 如果您選擇在權杖中使用 preferred_role 宣告選擇角色，Amazon Cognito 會為您使用者 cognito:preferred_role 宣告中的角色發出登入資料。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>
```

您可以從使用者集區 ID 衍生 **<region>** 的值。例如，如果使用者集區 ID 為 us-east-1_EXAMPLE1，則 **<region>** 為 us-east-1。如果使用者集區 ID 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(serviceConfiguration,
    userPoolConfiguration: userPoolConfiguration, forKey: "UserPool")
let pool = AWSCognitoIdentityUserPool(forKey: "UserPool")
```

```
let credentialsProvider = AWSCognitoCredentialsProvider(regionType: .USEast1,
  identityPoolId: "YOUR_IDENTITY_POOL_ID", identityProviderManager:pool)
```

範圍、M2M 和具有資源伺服器的 APIs

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[使用者集區受管登入](#)。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。使用者集區範圍位於存取權杖 scope 宣告中。如需 Amazon Cognito 存取權杖中宣告的詳細資訊，請參閱[了解存取權杖](#)。

使用 Amazon Cognito，存取字符中的範圍可以授權存取外部 APIs 或使用者屬性。您可以向本機使用者、聯合身分使用者或機器身分發出存取權杖。

主題

- [API 授權](#)
- [Machine-to-machine\(M2M\) 授權](#)
- [關於範圍](#)
- [關於資源伺服器](#)

API 授權

以下是您可以使用 Amazon Cognito 權杖授權 APIs 請求的一些方式：

存取權杖

將 Amazon Cognito 授權方新增至 REST API 方法請求組態時，請將授權範圍新增至授權方組態。透過此組態，您的 API 接受 Authorization 標頭中的存取字符，並檢閱它們是否有接受的範圍。

ID 權杖

當您將有效的 ID 字符傳遞至 REST API 中的 Amazon Cognito 授權方時，API Gateway 會接受請求並將 ID 字符內容傳遞至 API 後端。

Amazon Verified Permissions

在已驗證許可中，您可以選擇建立 [API 連結政策存放區](#)。Verified Permissions 會建立並指派 Lambda 授權方，以處理來自請求 Authorization 標頭的 ID 或存取權杖。此 Lambda 授權方會將權杖傳遞至您的政策存放區，其中 Verified Permissions 會將權杖與政策進行比較，並將允許或拒絕決策傳回給授權方。

其他 資源

- [在 API Gateway 中控制和管理對 REST API 的存取](#)
- [使用 Amazon Verified Permissions 進行授權](#)

Machine-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是在應用程式伺服器上執行並連線至遠端 APIs [機密用戶端](#)。它們的操作在沒有使用者互動的情況下進行：排程任務、資料串流或資產更新。當這些用戶端使用存取字符授權其請求時，他們會執行機器到機器或 M2M 授權。在 M2M 授權中，共用秘密會取代存取控制中的使用者登入資料。

使用 M2M 授權存取 API 的應用程式必須具有用戶端 ID 和用戶端秘密。在您的使用者集區中，您必須建置支援用戶端憑證授予的應用程式用戶端。若要支援用戶端登入資料，您的應用程式用戶端必須擁有用戶端秘密，而且您必須擁有使用者集區網域。在此流程中，您的機器身分會直接從 請求存取字符 [權杖端點](#)。您只能在用戶端憑證授予的存取權杖中，從 [資源伺服器](#) 授權自訂範圍。如需設定應用程式用戶端的詳細資訊，請參閱 [使用應用程式用戶端的應用程式特定設定](#)。

來自用戶端憑證授予的存取權杖是您想要允許機器身分從 API 請求的操作的可驗證陳述式。若要進一步了解存取權杖如何授權 API 請求，請繼續閱讀。如需範例應用程式，請參閱 [Amazon Cognito 和 API Gateway 型機器，以使用 AWS CDK 進行機器授權](#)。

M2M 授權的計費模式與每月作用中使用者 (MAUs) 的計費方式不同。當使用者身分驗證為每位作用中使用者帶來成本時，M2M 帳單會反映作用中的用戶端登入資料應用程式用戶端和字符請求總量。如需詳細資訊，請參閱 [Amazon Cognito 定價](#)。若要控制 M2M 授權的成本，請最佳化存取字符的持續時間，以及應用程式提出的字符請求數量。如需在 M2M 授權中使用 API Gateway 快取來減少新字符請求的方法，[管理使用者集區字符過期和快取](#) 請參閱。

如需最佳化 Amazon Cognito 操作以為您的 AWS 帳單增加成本的相關資訊，請參閱 [管理成本](#)。

machine-to-machine(M2M) 用戶端登入資料的用戶端中繼資料

您可以在 M2M 請求中傳遞[用戶端中繼資料](#)。用戶端中繼資料是來自使用者或應用程式環境的其他資訊，有助於的結果[產生權杖前 Lambda 觸發程序](#)。在使用者主體的身分驗證操作中，您可以將用戶端中繼資料傳遞至 [AdminRespondToAuthChallenge](#) 和 [RespondToAuthChallenge](#) API 請求內文中的權杖產生前觸發條件。由於應用程式會執行流程來產生 M2M 的存取字符，並直接請求 [權杖端點](#)，因此有不同的模型。在用戶端登入資料的字符請求 POST 內文中，將具有用戶端中繼資料物件 URL 編碼 (x-www-form-urlencoded) 的 `aws_client_metadata` 參數傳遞至字串。如需請求範例，請參閱 [具有基本授權的用戶端憑證](#)。以下是傳遞鍵/值對的範例參數 `{"environment": "dev", "language": "en-US"}`。

```
aws_client_metadata=%7B%22environment%22%3A%20%22dev%22,%20%22language%22%3A%20%22en-US%22%7D
```

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 使用者集區會發出存取權杖，其中包含使用者集區預留 API 範圍、自訂範圍和 OpenID Connect (OIDC) 範圍。

使用者集區保留的 API 範圍

此 `aws.cognito.signin.user.admin` 範圍授權 Amazon Cognito 使用者集區 API 中目前使用者的自助式操作。它授權存取字符的持有人使用 [GetUser](#) 和 [UpdateUserAttributes](#) API 操作來查詢和更新承載的所有資訊。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。您可以在本頁面中尋找有關自訂範圍的詳細資訊。

OpenID Connect (OIDC) 範圍

當您使用使用者集區授權伺服器驗證使用者時，包括使用受管登入時，您必須請求範圍。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。OIDC 範圍授權您的應用程式從使用者集區 [userInfo 端點](#) 區的讀取使用者資訊。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。您可以授權應用程式用戶端發行具有下列 OIDC 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 OIDC 範圍 email，例如 profile、和 phone 時，ID 字符和 [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。

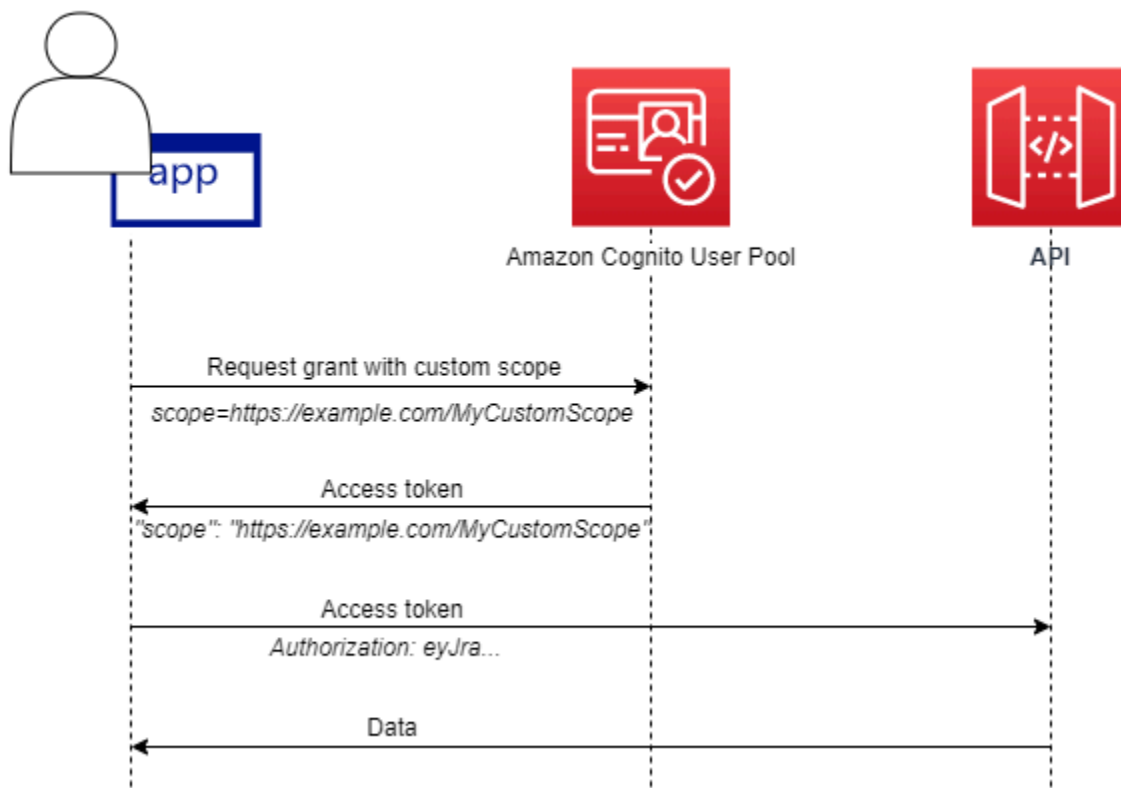
自訂範圍由您定義，並擴充使用者集區的授權功能，以納入與查詢和修改使用者及其屬性無關的用途。例如，如果您有用於照片的資源伺服器，它可能會定義兩個範圍：`photos.read` 用於照片的讀取存取權，`photos.write` 用於寫入/刪除存取權。您可以設定 API 接受存取權杖以進行授權，並在 `scope` 宣告中對存取權杖的 HTTP GET 請求授予 `photos.read`，以及對權杖的 HTTP POST 請求授予 `photos.write`。這些都是自訂範圍。

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 字符](#)。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。在受管登入中，您也可以從隱含授予請求存取字符中的自訂範圍。

Note

由於其設計是作為 IdP 與使用者集區進行人為互動驗證，因此 [InitiateAuth](#) 和 [AdminInitiateAuth](#) 請求只會在存取權杖中產生一個具有單一值 `aws.cognito.signin.user.admin` 的 scope 宣告。

管理資源伺服器 and 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。您可以在包含受管登入的 OAuth 2.0 授權伺服器進行身分驗證流程期間，選擇使用者存取字符的範圍。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 來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 為您的使用者集區定義資源伺服器。

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. 選擇品牌下的網域功能表，並尋找資源伺服器。
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. 選擇建立。

您可以在資源伺服器下的網域選單中，在自訂範圍欄中檢閱您的自訂範圍。您可以從應用程式下的應用程式用戶端選單，為應用程式用戶端啟用自訂範圍。選取應用程式用戶端，找到登入頁面，然後選擇編輯。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](#)

設定使用者集區功能

在先前的章節中，您可能已在 Amazon Cognito 主控台的指導下設定了一些功能。本節中的頁面深入探討使用者集區部分核心功能的詳細組態需求。有關應用程式用戶端、電子郵件和簡訊組態、記住使用者裝置等選項的重要參考資訊。

主題

- [更新使用者集區和應用程式用戶端組態](#)
- [使用應用程式用戶端的應用程式特定設定](#)
- [在使用者集區中使用使用者裝置](#)
- [使用 Amazon Pinpoint 進行使用者集區分析](#)
- [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](#)

更新使用者集區和應用程式用戶端組態

當您想要變更使用者集區或應用程式用戶端中的設定時，只要按幾下滑鼠，就可以在 Amazon Cognito 主控台中套用更新。您可以導覽使用者集區設定中的功能型索引標籤，並更新本指南其他部分所述的欄位。

許多組織會以程式設計方式管理資源 AWS CloudFormation、建置在 AWS SDKs 或 CDK 上的應用程式，以及其他自動化軟體。當這是您的資源管理模型時，您必須特別注意如何將資源進行階段變更。

API 操作 [UpdateUserPool](#) 和 [UpdateUserPoolClient](#) 會更新現有的使用者集區或應用程式用戶端。每個都隨附 API 參考中的警告：如果您未提供屬性的值，Amazon Cognito 會將它設定為預設值。當您只使用一個參數提交更新請求時，Amazon Cognito 會將該參數設定為您選擇的值，並將所有其他參數設定為預設值。這可以重設組態，包括您的屬性結構描述、Lambda 觸發條件，以及您的電子郵件和簡訊組態。

此外，某些設定會在您建立使用者集區或應用程式用戶端之後鎖定在 中，除非您建立新的資源，否則無法變更這些設定。

主題

- [您無法變更的設定](#)
- [SMS 組態](#)
- [使用 AWS SDK AWS CDK、或 REST API 更新使用者集區](#)

您無法變更的設定

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。

Note

先前，您無法變更使用者集區的名稱。這已變更。您現在可以將新的易記名稱指派給使用者集區。

使用者集區 ID

API 參數名稱：[Id/UserPoolId](#)

使用者集區 ID，例如 `us-east-1_EXAMPLE`，是由 Amazon Cognito 自動產生且無法變更。

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[使用應用程式用戶端的應用程式特定設定](#)。若要變更新用戶端密碼，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：[結構描述](#)

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[使用使用者屬性](#)。若要變更所需的屬性，請建立新的使用者集區。

自訂屬性（刪除）

API 參數名稱：[結構描述](#)

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[使用使用者屬性](#)。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。

SMS 組態

在使用者集區中啟用簡訊之後，您就無法停用它們。

- 如果您在建立使用者集區時選擇設定簡訊，則在完成設定後無法停用簡訊。
- 您可以在您建立的使用者集區中啟用簡訊，但之後無法停用簡訊。
- Amazon Cognito 可以使用 SMS 訊息進行使用者帳戶邀請和復原、屬性驗證和多重要素驗證 (MFA)。啟用簡訊後，您可以隨時開啟或關閉這些功能的簡訊。
- SMS 訊息組態包含您委派給 Amazon Cognito 以使用 Amazon SNS 傳送訊息的 IAM 角色。您可以隨時變更指派的角色。

使用 AWS SDK AWS CDK、或 REST API 更新使用者集區

在 Amazon Cognito 主控台中，您可以一次變更一個參數來變更使用者集區設定。例如，若要新增 Lambda 觸發條件，請選擇新增 Lambda 觸發條件，然後選擇函數和觸發條件類型。Amazon Cognito 使用者集區 API 的結構方式是更新使用者集區和應用程式用戶端的操作，需要使用者集區的一組完整參數。不過，主控台會使用其他使用者集區設定，以透明方式自動化此更新操作。

有時，您可能會發現在 中其他位置變更 AWS 帳戶 可能會導致更新在與您要變更的設定無關時產生錯誤。AWS WAF 例如，已刪除的 Amazon SES 身分或 IAM 許可的變更。如果其中一個目前參數不再存在

效，則在您修正設定之前，無法更新設定。當您遇到此類錯誤時，請檢查錯誤回應並驗證其提及的設定。

[AWS Cloud Development Kit \(AWS CDK\)](#)、[Amazon Cognito 使用者集區 REST API](#) 和 [AWS SDKs](#) 是 Amazon Cognito 資源自動化和程式設計組態的工具。使用這些工具的請求也必須像 Amazon Cognito 主控台一樣，在請求內文中更新具有完整資源組態的設定。在高階，您必須執行下列程序。

1. 從描述現有資源組態的 操作擷取輸出。
2. 使用設定變更來修改輸出。
3. 在更新資源的 操作中傳送修改後的組態。

以下程序以 [UpdateUserPool](#) API 操作更新您的組態。同樣的方法，使用不同的輸入欄位，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. 以 [DescribeUserPool](#) 擷取您使用者集區的現有狀態。
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
 - 以 [UpdateUserPoolDomain](#) API 操作更新此欄位。
 - Domain
 - 以 [UpdateUserPoolDomain](#) API 操作更新此欄位。
 - 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。

您也可以使用 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具有空白值的 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
```

使用應用程式用戶端的應用程式特定設定

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，如果您將應用程式設定為使用 AWS 登入資料簽署已驗證的 API 請求，您必須保護登入資料免遭使用者檢查。

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。

您可以在應用程式用戶端層級套用下列使用者集區功能的設定：

1. [分析](#)
2. [受管登入](#) IdPs、授予類型、回呼 URLs和自訂
3. [資源伺服器](#)和自訂範圍
4. [威脅防護](#)
5. [屬性讀取和寫入許可](#)
6. [權杖過期和撤銷](#)
7. [身分驗證流程](#)

應用程式用戶端類型

在 Amazon Cognito 中建立應用程式用戶端時，您可以根據標準 OAuth 用戶端類型 public client (公有用戶端) 和 confidential client (機密用戶端) 預先填入選項。以 client secret (用戶端密碼) 設定 confidential client (機密用戶端)。如需有關用戶端類型的詳細資訊，請參閱 [IETF RFC 6749 #2.1](#)。

Public client (公有用戶端)

公有用戶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，因此沒有用戶端密碼。

Confidential client (機密用戶端)

機密用戶端具有伺服器端資源，這些資源可透過 client secret (用戶端密碼) 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。您的應用程式用戶端必須具有用戶端密碼才能執行 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。

Note

當您為應用程式類型選取傳統 Web 應用程式和 Machine-to-machine 應用程式選項時，Amazon Cognito 主控台會建立具有用戶端秘密的應用程式用戶端。選擇其中一個選項來產生用戶端秘密，或使用 [CreateUserPoolClient](#) 以程式設計方式建立用戶端，並將 GenerateSecret 設定為 true。

您可以在公有應用程式中使用機密用戶端和用戶端密碼。使用 Amazon CloudFront 代理在傳輸中新增 SECRET_HASH。如需詳細資訊，請參閱 AWS 部落格中的[使用 Amazon CloudFront 代理保護 Amazon Cognito 的公有用戶端](#)。

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。如需更多資訊，請參閱[了解身分 \(ID\) 字符合](#)。

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。如需更多資訊，請參閱[了解存取權杖](#)。

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。如需更多資訊，請參閱[重新整理權杖](#)。

您可以從 [Amazon Cognito 主控台](#) 中使用使用者集區的應用程式用戶端選單，為每個應用程式用戶端設定這些字符的過期時間。

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

您的使用者集區可能有許多客戶端，每個客戶端都有自己的應用程式客戶端和 IdP。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。在機器對機器 (M2M) 授權等情況下，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許可時，Amazon Cognito 會將讀取和寫入許可授予所有使用者集區屬性。
- 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，您的應用程式用戶端可以將值寫入不可變屬性。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使用者名稱和密碼進行身分驗證、電子郵件和簡訊 OTPs、通行密鑰驗證程式、使用 Lambda 觸發程序進行自訂身分驗證，以及權杖重新整理。作為最佳安全實務，在自訂建置應用程式中使用 SRP 身分驗證進行使用者名稱和密碼身分驗證。

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/scope`。請參閱 [範圍、M2M 和具有資源伺服器的 APIs](#)。

預設重新導向 URI

使用第三方 IdPs 取代使用者身分驗證請求中的 `redirect_uri` 參數。使用 [CreateUserPoolClient](#) 或 [UpdateUserPoolClient](#) API 請求的 `DefaultRedirectURI` 參數來設定此應用程式用戶端設定。此 URL 也必須是應用程式用戶端 `CallbackURLs` 的成員。下列情況下，Amazon Cognito 會將已驗證的工作階段重新導向至此 URL：

1. 您的應用程式用戶端已指派一個[身分提供者](#)，並定義多個[回呼 URLs](#)。您的使用者集區在不包含 `redirect_uri` 參數時，會將身分驗證請求重新導向至[授權伺服器](#)至預設重新導向 URI。
2. 您的應用程式用戶端已指派一個[身分提供者](#)，並已定義一個[回呼 URLs](#)。在此案例中，您不需要定義預設回呼 URL。不包含 `redirect_uri` 參數重新導向至一個可用回呼 URL 的請求。

身分提供者

您可以選擇部分或所有使用者集區外部身分提供者 (IdP) 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 IdP 的授權連結，並將其顯示在受管登入頁面上。您可以指派多個 IdP，但您必須至少啟用一個。如需使用外部 IdP 的詳細資訊，請參閱[使用第三方身分提供者登入使用者集區](#)。

OpenID Connect 範圍

選擇下列一或多個 OAuth 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。
- `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](#) 的存取權限，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

用戶端憑證授予是用於機器對機器 (M2M) 的通訊。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 [範圍、M2M 和具有資源伺服器的 APIs](#)。

用戶端登入資料會授與 AWS 帳單的額外費用。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 登入資料。
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。這兩個選項都會提示您使用應用程式特定的設定來設定應用程式用戶端。
4. 選擇反映應用程式架構的應用程式類型。
5. 使用易記識別符命名您的應用程式。
6. 輸入傳回 URL。
7. 選擇 Create app client (建立應用程式用戶端)。您可以在建立應用程式用戶端後變更進階選項。
8. Amazon Cognito 會將您傳回應用程式用戶端詳細資訊。若要存取應用程式的範例程式碼，請從快速設定指南索引標籤中選取平台。

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 [CreateUserPoolClient](#) API 請求。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 AWS CLI 中輸入下列命令：

```
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，會 AWS CLI 傳回確認：

```
{  
  "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](#)

列出使用者集區 (AWS CLI 和 AWS API) 中的所有應用程式用戶端資訊

```
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使用者時，您可以將使用者來自[威脅防護](#)的活動日誌與其每個裝置建立關聯，並選擇性地允許您的使用者在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，您可以使用信任的裝置身分驗證流程來設定應用程式。在此流程中，您的應用程式可以向使用者提供在沒有 MFA 的情況下登入的選擇，直到稍後為止，取決於應用程式的安全需求或使用者的偏好設定。在該期間結束時，您的應用程式必須將裝置狀態變更為不記住，且使用者必須使用 MFA 登入，直到確認他們想要記住裝置為止。例如，您的應用程式可能會提示您的使用者信任裝置 30、60 或 90 天。您可以將此日期存放在自訂屬性中，並在該日期變更其裝置的記住狀態。然後，您必須重新提示使用者提交 MFA 代碼，並將裝置設定為在身分驗證成功後再次記住。

1. 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。如需詳細資訊，請參閱[使用裝置登入](#)。

在裝置追蹤下，設定您的使用者集區以記住使用者集區的登入選單中的裝置。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。

您也可以在此[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。如需詳細資訊，請參閱[了解 API、OIDC 和受管登入頁面身分驗證](#)。

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 (JWT) 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杖、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. 您也可以將 `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。對於使用裝置的 SRP 身分驗證，請為您的使用者產生新的 SRP 秘密：新的高熵密碼 `DeviceSecret`、`salt` 和相關聯的密碼驗證器。這些值與用於使用者 SRP 身分驗證的密碼、`salt` 和驗證器不同。它們僅用於裝置身分驗證，並且只會存放在裝置上。為使用者裝置產生 SRP 秘密的函數可在各種 SDKs 提供的 [SRP 程式庫](#) 中使用。

```
PASSWORD_CLAIM_SECRET_BLOCK = SECRET_BLOCK
TIMESTAMP = "Tue May 27 00:09:40 UTC 2025"
k = SHA256(N || g) as a non-negative integer in big-endian
u = SHA256(SRP_A || SRP_B) as a non-negative integer in big-endian
x = SHA256(salt || SHA256(DeviceGroupKey || DeviceKey || ":" || DeviceSecret)) as a
  non-negative integer in big-endian
S_USER = (SRP_B - k * g^x)^(a + u * x) % N
K_USER = HKDF_HMAC_SHA256(salt=u, ikm=S_USER, info="Caldera Derived Key", length=16
  bytes)
```



```
PASSWORD_CLAIM_SIGNATURE = Base64(HMAC_SHA256(key=K_USER, message=(DeviceGroupKey ||  
DeviceKey || PASSWORD_CLAIM_SECRET_BLOCK || TIMESTAMP)))
```

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。在受管登入中，您可以請求 `aws.cognito.signin.user.admin` 範圍，但受管登入會自動將裝置資訊新增至進階安全使用者日誌，而且不會提供記住裝置。

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 登入資料簽署 [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杖。
- 在機密用戶端應用程式中，使用 AWS 登入資料簽署 [AdminListDevices](#) API 請求，並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杖。
- 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杖。

使用 Amazon Pinpoint 進行使用者集區分析

Note

支援結束通知：2026 年 10 月 30 日，AWS 將結束對 Amazon Pinpoint 的支援。2026 年 10 月 30 日之後，您將無法再存取 Amazon Pinpoint 主控台或 Amazon Pinpoint 資源（端點、客群、行銷活動、旅程和分析）。如需詳細資訊，請參閱 [Amazon Pinpoint 終止支援](#)。注意：與 SMS、語音、行動推播、OTP 和電話號碼驗證相關的 APIs 不受此變更影響，並受 AWS 最終使用者傳訊支援。

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 主控台的應用程式用戶端選單中的應用程式用戶端分析組態中選擇與 Amazon Pinpoint 共用使用者屬性資料，Amazon Pinpoint 會為使用者電子郵件地址和電話號碼建立其他端點。 Amazon Cognito

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

下表顯示 Amazon Cognito 和 Amazon Pinpoint 之間符合下列其中一個條件的 AWS 區域 映射。

- 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)
- Middle East (Bahrain)
- 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)
- 亞太地區 (馬來西亞)

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
<code>ap-northeast-1</code>	<code>us-east-1</code>

Amazon Cognito 使用者集區區域	Amazon Pinpoint 專案區域
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-east-1、eu-central-1
eu-west-1	us-east-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-east-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

對於上述資料表中的所有 AWS 區域，Amazon Cognito 只能使用與您使用者集區相同區域中的 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. 選擇應用程式用戶端選單，然後選取您要更新的應用程式用戶端。
4. 在 Pinpoint 分析下的分析索引標籤中，選擇啟用。
5. 選擇 Pinpoint Region (Pinpoint 區域)。
6. 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](#)。

7. 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](#)。

8. 選擇儲存變更。

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

您可以使用具有自訂使用者介面 (UI) 的應用程式，與 AWS SDK 整合的操作。您無法為透過受管登入登入的聯合身分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？](#)。

Amazon Cognito 使用者集區的電子郵件設定

應用程式中的某些事件可能會導致 Amazon Cognito 傳送電子郵件給您的使用者。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：

- 內建於 Amazon Cognito 服務的[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
Forgot password	ForgotPassword , AdminResetUserPassword	Email, SMS	code	No	N/A
Invitation	AdminCreateUser	Email, SMS	code	Yes	邀請訊息
Self-registration	SignUp , ResendConfirmationCode	Email, SMS	code, link	Yes	驗證訊息
Email address or phone number verification	UpdateUserAttributes , AdminUpdateUserAttributes , GetUserAttributeVerificationCode	Email, SMS	code	Yes	驗證訊息
Multi-factor authentication (MFA)	AdminInitiateAuth , InitiateAuth	Email ¹ , SMS, authenticator app	code	Yes ²	MFA 訊息

1 需要進階安全功能和 [Amazon SES 電子郵件組態](#)。

2 適用於簡訊和電子郵件訊息。

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](#)。

若要進一步了解電子郵件 MFA，請參閱 [SMS 和電子郵件訊息 MFA](#)。

Amazon Cognito 可能會阻止在短時間內將額外的電子郵件或簡訊傳送至單一目的地。如果您認為使用者集區受到影響，請設定和檢閱[訊息傳遞錯誤的日誌](#)，然後聯絡您的客戶團隊。

預設電子郵件組態

Amazon Cognito 可以使用其預設電子郵件組態來為您處理電子郵件交付。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。

當您使用預設組態時，您可以使用由 AWS 管理的 Amazon SES 資源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**。如果稍後無法交付的電子郵件地址變成可交付，當使用者集區設定為使用預設組態時，您無法控制其從禁止名單中移除。電子郵件地址可以無限期保留在 AWS 受管禁止名單中。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

AWS 區域 您建立使用者集區的 對使用 Amazon SES 設定電子郵件訊息有三個需求之一。您可以從與使用者集區位於相同區域的 Amazon SES、包括相同區域或一或多個遠端區域的多個區域傳送電子郵件訊息。為了獲得最佳效能，當您有 選項時，請在與使用者集區相同的區域中傳送具有 Amazon SES 驗證身分的電子郵件訊息。

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 AWS 區域 與使用者集區相同的 中，傳送具有已驗證身分的電子郵件訊息。在沒有自訂FROM電子郵件地址的預設電子郵件組態中，Amazon Cognito 會在相同區域中使用no-reply@verificationemail.com已驗證的身分。

回溯相容

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)

此功能支援您可能在服務啟動時為了符合 Amazon Cognito 需求而建立之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FROM電子郵件地址的預設電子郵件組態中，Amazon Cognito 會在相同區域中使用no-reply@verificationemail.com已驗證的身分。

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 中，傳送具有已驗證身分 AWS 區域 的電子郵件訊息。當 Amazon SES 在可使用 Amazon Cognito 的區域中無法使用時，就會發生此組態。

替代區域中已驗證身分的 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[授予使用預設電子郵件組態的許可](#)。

在其中一些區域中，Amazon Cognito 會在兩個替代區域之間分割電子郵件訊息，以用於 的預設電子郵件組態COGNITO_DEFAULT。在這些情況下，若要使用自訂FROM電子郵件地址，每個替代區域中已驗證身分的 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[授予使用預設電子郵件組態的許可](#)。DEVELOPER 在這些區域中使用的 Amazon

SES 電子郵件組態，您必須在第一個列出的區域中使用已驗證的身分，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，設定歐洲（法蘭克福）的已驗證身分來信任 `cognito-idp.me-central-1.amazonaws.com`。在沒有自訂 FROM 電子郵件地址的預設電子郵件組態中，Amazon Cognito 在每個區域中使用 `no-reply@verificationemail.com` 已驗證的身分。

Note

在下列條件組合下，您必須在區域元素中使用萬用字元指定 [EmailConfiguration](#) 的 SourceArn 參數，格式為

`arn:${Partition}:ses:*:${Account}:identity/${IdentityName}`。這可讓您的使用者集區在兩者 AWS 帳戶 的 中，以相同的已驗證身分傳送電子郵件訊息 AWS 區域。

- 您的 EmailSendingAccount 是 COGNITO_DEFAULT。
- 您想要使用自訂 FROM 地址。
- 您的使用者集區會在替代區域中傳送電子郵件。
- 您的使用者集區在下列 Amazon SES 支援區域的表格中指定了第二個¹備用區域。Amazon SES

如果您以程式設計方式使用 AWS SDK 建立使用者集區，Amazon Cognito API 或 CLI AWS CDK、或 AWS CloudFormation- 您的使用者集區會使用 [EmailConfiguration](#) SourceArn 參數為使用者集區指定的 Amazon SES 身分傳送電子郵件訊息。Amazon SES 身分必須佔用支援的 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 AWS 區域 的。

使用者集區區域	區域選項	Amazon SES 支援的區域
美國東部 (維吉尼亞北部)	回溯相容	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	Amazon SES 支援的區域
美國東部 (俄亥俄)	回溯相容	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)	回溯相容	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)	回溯相容	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)
加拿大西部 (卡加利)	替代區域	加拿大 (中部)、美國西部 (加利佛尼亞北部) ¹
亞太區域 (東京)	回溯相容	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)
亞太區域 (香港)	替代區域	亞太區域 (新加坡)、亞太區域 (東京) ¹
亞太區域 (首爾)	回溯相容	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)
亞太地區 (馬來西亞)	替代區域	亞太區域 (雪梨)、亞太區域 (新加坡) ¹
亞太區域 (孟買)	回溯相容	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)

使用者集區區域	區域選項	Amazon SES 支援的區域
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)	回溯相容	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)	回溯相容	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)	回溯相容	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)	回溯相容	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)
歐洲 (法蘭克福)	回溯相容	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)

使用者集區區域	區域選項	Amazon SES 支援的區域
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 會在每個區域中具有相同電子郵件地址的已驗證身分之間分配電子郵件訊息。若要使用自訂FROM地址，EmailConfiguration請使用格式為的 SourceArn 參數來設定 `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 中使用 Amazon SES 時 AWS 區域，它會將您的 放置在該區域的 Amazon SES 沙盒 AWS 帳戶 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。當您的 AWS 帳戶 保留在沙盒中時，請不要將 Amazon SES 組態用於生產中的應用程式。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 從沙盒中移除您的，請參閱 [《Amazon SES Simple Email Service 開發人員指南》中的移出 Amazon SES 沙盒](#)。

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。您授予的許可，以及您用來授予許可的程序，取決於您使用的是預設電子郵件組態或 Amazon SES 組態。

授予使用預設電子郵件組態的許可

只有在您將使用者集區設定為使用 Cognito 傳送電子郵件或EmailSendingAccount設定為 時，才完成此步驟COGNITO_DEFAULT。

使用預設電子郵件組態，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。

- 預設地址 no-reply@verificationemail.com。
- 來自 Amazon SES 中已驗證電子郵件地址或網域的自訂 FROM 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。這些許可是透過 Amazon SES 中地址或網域的[傳送授權政策](#)所授予。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。不過，如果您在主控台之外設定使用者集區，例如使用 AWS CLI 或 Amazon Cognito API，則必須使用 [Amazon SES 主控台](#) 或 [PutIdentityPolicy](#) API 連接政策。

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](#)》。您也可以參閱《[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](#)。如果您的使用者集區區域至少符合 Amazon SES 區域中的一個值，請在下列範例中使用 全域服務主體設定您的傳送授權政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "stmt1234567891234",
      "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 不適用於可使用 Amazon Cognito AWS 區域的所有選擇加入。中東（阿拉伯聯合大公國）是範例，只能在歐洲（法蘭克福）() 傳送已驗證身分的電子郵件eu-central-1。在具有預設電子郵件組態的使用者集區中，Amazon Cognito 也會在兩個區域中分別傳送具有已驗證身分的電子郵件訊息。如果是中東（阿拉伯聯合大公國），則額外區域為歐洲（倫敦）。您必須更新兩個區域中的傳送授權政策。

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](#)。如果您的區域標示為備用區域，請使用區域服務主體設定您的傳送授權政策，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 會使用設定組態之使用者工作階段的 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 [存取 AWS 資源管理](#)。

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 電子郵件地址的值。

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 登入資料。
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. 選擇身分驗證方法選單、尋找電子郵件組態、選擇編輯。
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](#)。

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
Forgot password	ForgotPassword , AdminResetUserPassword	Email, SMS	code	No	N/A
Invitation	AdminCreateUser	Email, SMS	code	Yes	邀請訊息
Self-registration	SignUp , ResendConfirmationCode	Email, SMS	code, link	Yes	驗證訊息
Email address or phone	UpdateUserAttributes , AdminUpdateUserAttributes	Email, SMS	code	Yes	驗證訊息

活動	API 操作	傳遞選項	格式選項	可自訂	訊息範本
number verification	GetUserAttributeVerificationCode				
Multi-factor authentic ation (MFA)	AdminInitiateAuth , InitiateAuth	SMS, authenticator app	code	Yes ¹	MFA 訊息

¹ 適用簡訊。

SMS 訊息的 Amazon SNS 費用。如需詳細資訊，請參閱 [Amazon SNS 定價](#)。

若要進一步了解 MFA，請參閱 [SMS 和電子郵件訊息 MFA](#)。

Amazon Cognito 可能會阻止在短時間內將額外的電子郵件或簡訊傳送至單一目的地。如果您認為使用者集區受到影響，請設定和檢閱[訊息傳遞錯誤的日誌](#)，然後聯絡您的客戶團隊。

最佳實務

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。

如需保護簡訊交付架構的相關資訊，請參閱下列連結。

- [使用 Amazon Cognito 使用者集區降低使用者註冊詐騙和 SMS 幫浦的風險](#)
- [防禦 SMS Pumping：協助對抗人工膨脹流量的新 AWS 功能](#)

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。您第一次設定 Amazon SNS 傳送特定 SMS 文

字訊息時 AWS 區域，Amazon SNS 會將您的 放置在該區域的 SMS 沙盒 AWS 帳戶 中。Amazon SNS 會使用沙盒來防止詐騙和濫用，並符合合規要求。當您 AWS 帳戶 的 位於沙盒中時，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。當您的 AWS 帳戶 保留在沙盒中時，請勿將 Amazon SNS 組態用於生產中的應用程式。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 Amazon SNS SMS 訊息 AWS 區域 的](#)
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，您可以從使用者集區的身分驗證方法選單，在 SMS 下設定 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 只有在符合下列條件時才能擔任該角色：

- 擔任角色操作代表 `aws:SourceArn` 條件中的使用者集區。
- 擔任角色操作代表 `aws:SourceAccount` 條件所 AWS 帳戶 設定 中的使用者集區。
- 擔任角色操作會在 `sts:externalId` 條件中包含外部 ID。

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

```

        "Service": "cognito-idp.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
        "StringEquals": {
            "sts:ExternalId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE2222",
            "aws:SourceAccount": "111122223333"
        },
        "ArnLike": {
            "aws:SourceArn": "arn:aws:cognito-idp:us-west-2:111122223333:userpool/us-west-2_EXAMPLE"
        }
    }
}
]
}

```

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 中或使用 AWS Management Console [DescribeUserPool](#) API 請求查詢使用者集區的 ARNs。

若要傳送 SMS 訊息以進行[多重要素驗證](#)，您的 IAM 角色信任政策必須具有 `sts:ExternalId` 條件。此條件的值必須符合使用者集區的 [SmsConfiguration](#) `ExternalId` 屬性。當您在 Amazon Cognito 主控台中建立使用者集區的過程中建立 IAM 角色時，Amazon Cognito 會在角色和使用者集區設定中為您設定外部 ID。當您使用現有的 IAM 角色時，情況並非如此。

您必須更新 [UpdateUserPool](#) API 請求中的使用者集區 `ExternalId` 參數，並使用具有相同值 `sts:externalId` 的條件更新 IAM 角色信任政策。若要了解如何使用 API 以保留原始組態的方式更新使用者集區，請參閱 [更新使用者集區和應用程式用戶端組態](#)。

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 Amazon SNS SMS 訊息 AWS 區域的

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。Amazon Cognito 在可使用 Amazon Cognito 的任何 AWS 區域中，除了亞太區域（首爾）以外，您可以在 AWS 區域建立使用者集區的中使用 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。您也可以現在現有使用者集區的身分驗證方法選單中選擇 SMS 下的編輯。

在啟動時，對於某些 Amazon Cognito AWS 區域會在替代區域中使用 Amazon SNS 資源傳送簡訊。若要設定您的偏好區域，請使用您使用者集區的 [SmsConfigurationType](#) 物件的 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)。您可能已增加與 Amazon Cognito 搭配使用 AWS 區域 的花費限制。在您變更 Amazon SNS SMS 訊息 AWS 區域 之前，請在 AWS 支援中心開啟配額增加案例，以提高您在新區域中的限制。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)
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)
加拿大 (中部)	加拿大 (中部)、美國東部 (維吉尼亞北部)
加拿大西部 (卡加利)	加拿大西部 (卡加利)

Amazon Cognito 區域	Amazon SNS 區域
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)
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)
歐洲 (蘇黎世)	歐洲 (蘇黎世)
亞太地區 (馬來西亞)	亞太區域 (新加坡)
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)
亞太區域 (香港)	亞太區域 (新加坡)
亞太區域 (首爾)	亞太區域 (東京)
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)
Middle East (Bahrain)	Middle East (Bahrain)

Amazon Cognito 區域	Amazon SNS 區域
中東 (阿拉伯聯合大公國)	中東 (阿拉伯聯合大公國)
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)

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](#)。

如果您在下列情況下操作 AWS 區域，則必須開立 支援 票證以取得原始身分。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)

當您在同一 中有多個起始身分時 AWS 區域，Amazon SNS 會依下列優先順序選擇起始身分類型：短碼、10DLC、免付費電話號碼。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 重複此步驟。

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. 選擇身分驗證方法選單。
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盒狀態 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 AWS 區域中設定原始身分，其中包含您希望 Amazon Cognito 使用的 Amazon SNS 資源後，您可以在您的 AWS 帳戶保留在 SMS 沙盒中時驗證美國電話號碼。Amazon Cognito 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

返回您要建立或[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。

使用 Amazon Cognito 使用者集區安全性功能

您可能想要保護您的應用程式免於網路入侵、密碼猜測、使用者模擬，以及惡意註冊和登入。Amazon Cognito 使用者集區安全功能的組態可以是安全架構中的關鍵元件。應用程式的安全是客戶的責任「雲端安全」，如 AWS [共同的責任模型](#) 中所述。本章中的工具有助於您的應用程式安全設計符合這些目標。

設定使用者集區時必須做出的重要決策是是否允許公開註冊和登入。有些使用者集區選項，例如機密用戶端、使用者的管理建立和確認，以及沒有網域的使用者集區，都會受到較小程度的網際網路攻擊。不過，常見的使用案例是接受來自網際網路上任何人註冊的公有用戶端，並將所有操作直接傳送到您的使用者集區。在任何組態中，尤其是在這些公有組態的情況下，我們建議您以安全功能來規劃和部署使用者集區。當不需要的來源建立新的作用中使用者或嘗試利用現有使用者時，安全性不足也可能會影響您的 AWS 帳單。

MFA 和威脅防護適用於[本機使用者](#)。第三方 IdPs 負責[聯合身分使用者](#)的安全狀態。

使用者集區安全功能

多重要素驗證 (MFA)

請求使用者集區透過電子郵件（使用 Essentials 或 Plus 功能計劃）或 SMS 訊息，或從驗證器應用程式傳送的程式碼，以確認使用者集區登入。

威脅防護

監控登入是否有風險指標，並套用 MFA 或區塊登入。新增自訂宣告和範圍以存取字符。透過電子郵件傳送 MFA 代碼。

AWS WAF Web ACLs

檢查使用者[集區端點和身分驗證 API](#)的傳入流量，是否有網路和應用程式層上不必要的活動。

區分大小寫

防止建立電子郵件地址或慣用使用者名稱與其他使用者相同的使用者，但字元案例除外。

刪除保護

防止自動化系統意外刪除您的使用者集區。需要額外確認 中的使用者集區刪除 AWS Management Console。

使用者存在錯誤

防止在使用者集區中公開發現有的使用者名稱和別名。傳回一般錯誤以回應身分驗證失敗，無論使用者名稱是否有效。

主題

- [將 MFA 新增到使用者集區](#)
- [具有威脅防護的進階安全性](#)
- [將 AWS WAF Web ACL 與使用者集區建立關聯](#)
- [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

MFA 會將您具有身分驗證因素的項目新增至您所知的初始因素，通常是使用者名稱和密碼。您可以選擇簡訊、電子郵件訊息或以時間為基礎的一次性密碼 (TOTP) 作為其他因素，以登入具有密碼作為其主要身分驗證因素的使用者。

多重要素驗證 (MFA) 可提高應用程式中[本機使用者](#)的安全性。如果是[聯合身分使用者](#)，Amazon Cognito 會將所有身分驗證程序委派給 IdP，而不會為他們提供其他身分驗證因素。

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。

透過自適應身分驗證，您可以將使用者集區設定為需要額外的身分驗證因素，以回應增加的風險層級。若要將適應性身分驗證新增到使用者集區，請參閱 [具有威脅防護的進階安全性](#)。

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。若要登入，每個使用者必須至少設定一個 MFA 因素。需要 MFA 時，您必須在使用者加入中包含 MFA 設定，以便您的使用者集區允許他們登入。

當您將 MFA 設定為必要時，受管登入會提示使用者設定 MFA。當您在使用者集區中將 MFA 設定為選用時，受管登入不會提示使用者。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。

主題

- [使用者集區 MFA 的須知事項](#)
- [使用者 MFA 偏好設定](#)
- [使用者執行時間的 MFA 邏輯詳細資訊](#)
- [設定使用者集區以進行多重驗證](#)
- [SMS 和電子郵件訊息 MFA](#)
- [TOTP 軟體權杖 MFA](#)

使用者集區 MFA 的須知事項

設定 MFA 之前，請考慮下列事項：

- 使用者可以擁有 MFA 或使用無密碼因素登入。
 - 您無法在支援[一次性密碼](#)或[通行金鑰](#)的使用者集區中將 MFA 設定為必要。
 - 當使用者集區中需要 MFA AllowedFirstAuthFactors 時 WEB_AUTHN，您無法將 EMAIL_OTP、或 SMS_OTP 新增至。在 Amazon Cognito 主控台中，您無法編輯以選擇為基礎的登入選項，以包含無密碼因素。
 - 當使用者集區中需要 MFA 時，[選擇型登入](#)只會在所有應用程式用戶端中提供 PASSWORD 和 PASSWORD_SRP 因素。如需使用者名稱密碼流程的詳細資訊，請參閱本指南的身分驗證章節[使用持久性密碼和安全承載登入](#)中的 [使用持久性密碼登入](#)和。
 - 在 MFA 為選用的使用者集區中，設定 MFA 因素的使用者只能在選擇型登入中使用使用者名稱密碼驗證流程登入。這些使用者符合所有[用戶端型登入](#)流程的資格。
- 使用者偏好的 MFA 方法會影響他們可以用來復原密碼的方法。偏好的 MFA 為電子郵件訊息的使用者無法透過電子郵件接收密碼重設碼。偏好的 MFA 是 SMS 訊息的使用者無法透過 SMS 接收密碼重設碼。

當使用者不符合您偏好的密碼重設方法時，[您的密碼復原](#)設定必須提供替代選項。例如，您的復原機制可能以電子郵件為第一優先，而電子郵件 MFA 可能是使用者集區中的選項。在此情況下，請新增簡訊帳戶復原做為第二個選項，或使用管理 API 操作來重設這些使用者的密碼。

[UpdateUserPool](#) 的請求內文範例說明 AccountRecoverySetting，當無法重設密碼時，使用者可以透過簡訊恢復。

- 使用者無法在相同的電子郵件地址或電話號碼接收 MFA 和密碼重設代碼。如果他們使用來自 MFA 電子郵件訊息的一次性密碼 (OTPs)，則必須使用 SMS 訊息進行帳戶復原。如果他們使用來自 MFA 簡訊的 OTPs，則必須使用電子郵件訊息進行帳戶復原。在具有 MFA 的使用者集區中，如果使用者擁有電子郵件地址的屬性，但沒有電話號碼，或是電話號碼但沒有電子郵件地址，則可能無法完成自助式密碼復原。

若要防止使用者無法使用此組態在使用者集區中重設密碼的狀態，請視需要設定 email 和 phone_number 屬性。[???或者](#)，您可以設定程序，在使用者註冊或管理員建立使用者設定檔時，一律收集和設定這些屬性。當使用者具有這兩個屬性時，Amazon Cognito 會自動將密碼重設代碼傳送至不是使用者 MFA 因素的目的地。

- 當您在使用者集區中啟用 MFA 並選擇簡訊或電子郵件訊息作為第二個因素時，您可以將訊息傳送到尚未在 Amazon Cognito 中驗證的電話號碼或電子郵件屬性。使用者完成 MFA 後，Amazon Cognito 會將其 phone_number_verified 或 email_verified 屬性設定為 true。

- 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 [失敗登入嘗試的鎖定行為](#) 中所述。
- 如果您的帳戶位於 中的簡訊沙盒中 AWS 區域，其中包含使用者集區的 Amazon Simple Notification Service (Amazon SNS) 資源，您必須先在 Amazon SNS 中驗證電話號碼，才能傳送簡訊。如需詳細資訊，請參閱[Amazon Cognito 使用者集區的簡訊設定](#)。
- 若要變更使用者的 MFA 狀態以回應具有威脅防護的偵測到事件，請在 Amazon Cognito 使用者集區主控台中啟用 MFA 並將其設定為選用。如需詳細資訊，請參閱[具有威脅防護的進階安全性](#)。
- 電子郵件和簡訊要求使用者分別擁有電子郵件地址和電話號碼屬性。您可以在使用者集區中將 email 或 phone_number 設定為必要屬性。在此情況下，除非使用者提供電話號碼，否則無法完成註冊。如果您未將這些屬性設定為必要，但想要執行電子郵件或簡訊 MFA，請在使用者註冊時提示他們輸入電子郵件地址或電話號碼。最佳實務是將您的使用者集區設定為自動傳送訊息給使用者，以[驗證這些屬性](#)。

如果使用者已成功透過簡訊或電子郵件訊息收到臨時密碼，並在 [VerifyUserAttribute](#) API 請求中傳回該代碼，Amazon Cognito 會將電話號碼或電子郵件地址計為已驗證。或者，您的團隊可以設定電話號碼，並使用執行 [AdminUpdateUserAttributes](#) API 請求的管理應用程式將其標記為已驗證。

- 如果您已將 MFA 設定為必要，並啟用多個身分驗證因素，Amazon Cognito 會提示新使用者選取他們要使用的 MFA 因素。使用者必須擁有設定簡訊 MFA 的電話號碼，以及設定電子郵件訊息 MFA 的電子郵件地址。如果使用者未為任何可用訊息型 MFA 定義 屬性，Amazon Cognito 會提示他們設定 TOTP MFA。選擇 MFA 因素 (SELECT_MFA_TYPE) 和設定所選因素 (MFA_SETUP) 的提示，會做為 [InitiateAuth](#) 和 [AdminInitiateAuth](#) API 操作的挑戰回應。

使用者 MFA 偏好設定

使用者可以設定多個 MFA 因素。只有一個可以處於作用中狀態。您可以在使用者集區設定或從使用者提示中選擇使用者的有效 MFA 偏好設定。當使用者集區設定及其自己的使用者層級設定符合下列條件時，使用者集區會提示使用者輸入 MFA 代碼：

1. 您可以在使用者集區中將 MFA 設定為選用或必要。
2. 使用者具有有效的 phone_number 或 email 屬性，或已設定 TOTP 的驗證器應用程式。
3. 至少一個 MFA 因素處於作用中狀態。
4. 一個 MFA 因素設定為偏好。

使用者集區設定及其對 MFA 選項的影響

使用者集區的組態會影響使用者可以選擇的 MFA 方法。以下是一些使用者集區設定，會影響使用者設定 MFA 的能力。

- 在 Amazon Cognito 主控台的登入功能表中的多重要素驗證組態中，您可以將 MFA 設定為選用或必要，或將其關閉。此設定的 API 對等項目是 `CreateUserPool`、`UpdateUserPool` 和的 [MfaConfiguration](#) 參數 `SetUserPoolMfaConfig`。

此外，在多重要素驗證組態中，MFA 方法設定會決定使用者可以設定的 MFA 要素。此設定的 API 對等項目是 [SetUserPoolMfaConfig](#) 操作。

- 在登入選單的使用者帳戶復原下，您可以設定使用者集區傳送訊息給忘記密碼的使用者的方式。使用者的 MFA 方法不能具有與忘記密碼代碼的使用者集區交付方法相同的 MFA 交付方法。忘記密碼交付方法的 API 參數是 `CreateUserPool` 和的 [AccountRecoverySetting](#) 參數 `UpdateUserPool`。

例如，當復原選項為僅限電子郵件時，使用者無法設定電子郵件 MFA。這是因為您無法啟用電子郵件 MFA 並將復原選項設定為僅在相同的使用者集區中傳送電子郵件。當您將此選項設定為電子郵件時，否則 SMS 會是優先順序復原選項，但當使用者不符合電子郵件訊息復原的資格時，您的使用者集區可能會回到 SMS 訊息。在此案例中，使用者可以將電子郵件 MFA 設定為偏好，而且只有在他們嘗試重設密碼時才會收到簡訊。

- 如果您只將一個 MFA 方法設為可用，則不需要管理使用者 MFA 偏好設定。
- 作用中的 SMS 組態會自動讓 SMS 訊息成為使用者集區中可用的 MFA 方法。

在使用者集區中包含您自己的 Amazon SES 資源的作用中 [電子郵件組態](#)，以及 Essentials 或 Plus 功能計劃，會自動讓電子郵件訊息成為使用者集區中可用的 MFA 方法。

- 當您在使用者集區中將 MFA 設定為必要時，使用者無法啟用或停用任何 MFA 方法。您只能設定偏好的方法。
- 當您在使用者集區中將 MFA 設定為選用時，受管登入不會提示使用者設定 MFA，但會在使用者擁有偏好的 MFA 方法時提示他們輸入 MFA 代碼。
- 當您在全功能模式下啟用 [威脅防護](#) 並設定適應性驗證回應時，MFA 必須是使用者集區中的選用項目。具有適應性身分驗證的回應選項之一，是針對其登入嘗試評估為包含風險層級的使用者，要求 MFA。

主控台的註冊功能表中的必要屬性設定會決定使用者是否必須提供電子郵件地址或電話號碼才能註冊您的應用程式。當使用者具有對應的屬性時，電子郵件和簡訊會變成合格的 MFA 因素。的 [結構描述](#) 參數會視需要 `CreateUserPool` 設定屬性。

- 當您在使用者集區中將 MFA 設定為必要，且使用者使用受管登入時，Amazon Cognito 會提示他們從使用者集區的可用方法中選取 MFA 方法。受管登入會處理電子郵件地址或電話號碼的集合，以及 TOTP 的設定。下圖示範 Amazon Cognito 提供給使用者的選項背後的邏輯。

設定使用者的 MFA 偏好設定

您可以在具有存取金鑰授權的自助式模型中，或在具有管理 API 操作的管理員受管模型中，為使用者設定 MFA 偏好設定。這些操作會啟用或停用 MFA 方法，並將多種方法之一設定為偏好選項。您的使用者設定 MFA 偏好設定後，Amazon Cognito 會在登入時提示他們從偏好的 MFA 方法提供代碼。尚未設定偏好設定的使用者會收到提示，在 SELECT_MFA_TYPE 挑戰中選擇偏好的方法。

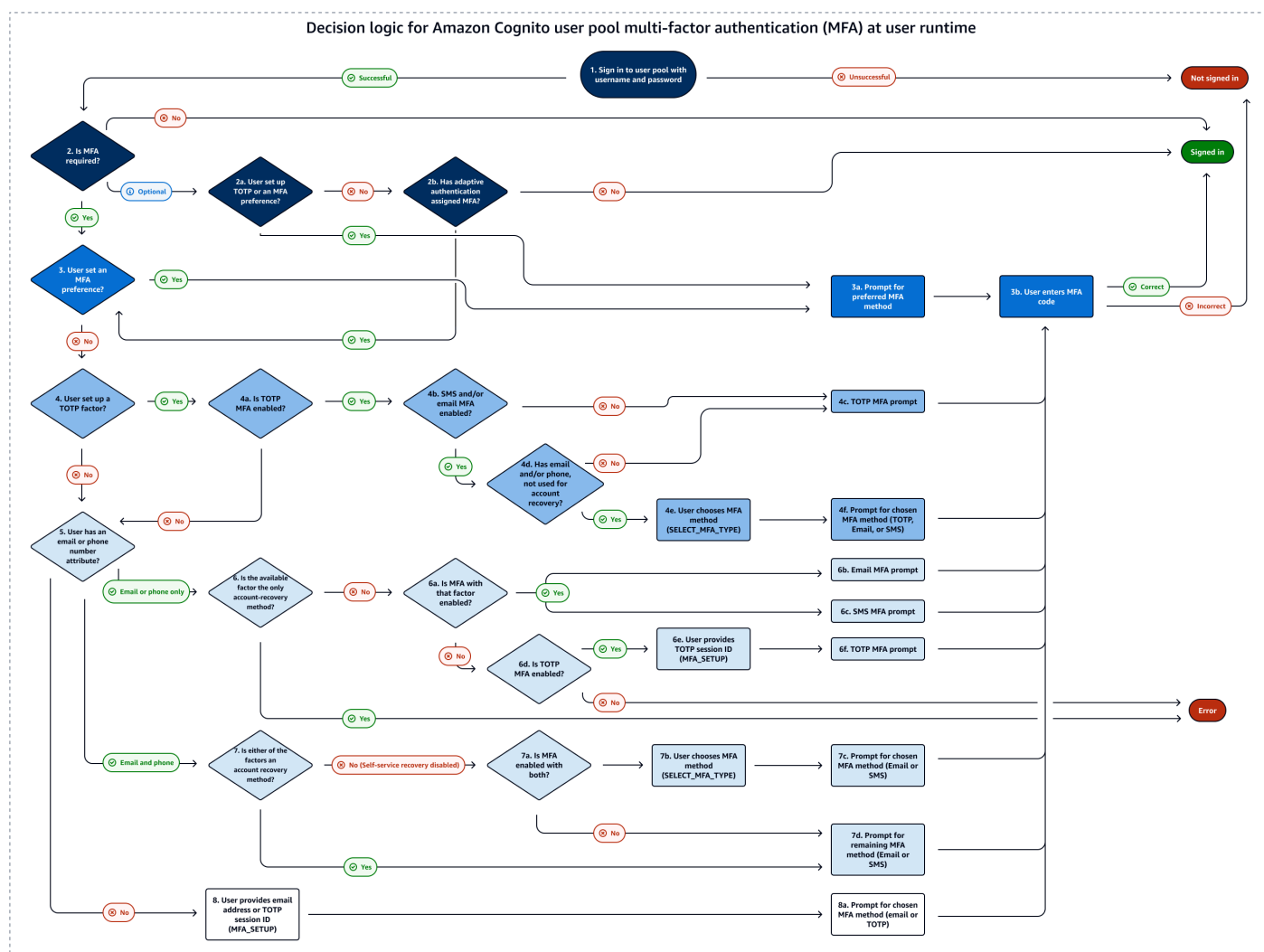
- 在使用者自助式模型或公有應用程式中，使用登入使用者的存取字符授權的 [SetUserMfaPreference](#) 會設定 MFA 組態。
- 在管理員受管或機密應用程式中，使用管理 AWS 登入資料授權的 [AdminSetUserPreference](#) 會設定 MFA 組態。

您也可以從 Amazon Cognito 主控台的使用者選單設定使用者 MFA 偏好設定。如需 Amazon Cognito 使用者集區 API 中公有和機密身分驗證模型的詳細資訊，請參閱 [了解 API、OIDC 和受管登入頁面身分驗證](#)。

使用者執行時間的 MFA 邏輯詳細資訊

為了判斷使用者登入時要採取的步驟，您的使用者集區會評估使用者 MFA 偏好設定、[使用者屬性](#)、[使用者集區 MFA 設定](#)、[威脅防護](#) 動作和 [自助式帳戶復原](#) 設定。然後，它會登入使用者、提示他們選擇 MFA 方法、提示他們設定 MFA 方法，或提示他們輸入 MFA。若要設定 MFA 方法，使用者必須提供 [電子郵件地址或電話號碼](#)，或 [註冊 TOTP 驗證器](#)。他們也可以設定 MFA 選項，並 [事先註冊偏好的選項](#)。下圖列出使用者集區組態在初次註冊後立即嘗試登入時的詳細效果。

此處說明的邏輯適用於以 SDK 為基礎的應用程式和 [受管登入](#)，但在受管登入中不可見。當您對 MFA 進行故障診斷時，請從使用者的結果回溯到促成決策的使用者設定檔和使用集區組態。



下列清單對應至決策邏輯圖表中的編號，並詳細說明每個步

驟。

指出成功的身分驗證和流程的結

論。 

表示身分驗證失敗。

1. 使用者會在您的登入畫面中顯示其使用者名稱或使用者名稱和密碼。如果他們沒有出示有效的登入資料，則會拒絕他們的登入請求。
2. 如果他們成功進行使用者名稱密碼身分驗證，請判斷 MFA 是必要、選用還是關閉。如果關閉，正確的使用者名稱和密碼會導致身分驗證成功。



- a. 如果 MFA 是選用的，請判斷使用者先前是否已設定 TOTP 驗證器。如果他們已設定 TOTP，則提示 TOTP MFA。如果他們成功回應 MFA 挑戰，則會登入。



- b. 判斷威脅防護的適應性身分驗證功能是否需要使用者設定 MFA。如果尚未指派 MFA，則會登入使用者。



3. 如果需要 MFA 或自適應身分驗證已指派 MFA，請判斷使用者是否已將 MFA 因素設定為已啟用且偏好。如果有，請提示 MFA 使用該因素。如果他們成功回應 MFA 挑戰，則會登入。



4. 如果使用者尚未設定 MFA 偏好設定，請判斷使用者是否已註冊 TOTP 驗證器。

- a. 如果使用者已註冊 TOTP 驗證器，請判斷使用者集區中是否提供 TOTP MFA（在使用者先前設定驗證器之後，可以停用 TOTP MFA）。
- b. 判斷使用者集區中是否也提供電子郵件訊息或簡訊 MFA。
- c. 如果電子郵件或簡訊 MFA 都無法使用，請提示使用者輸入 TOTP MFA。如果他們成功回應 MFA 挑戰，則會登入。



- d. 如果電子郵件或簡訊 MFA 可用，請判斷使用者是否有對應的 phone_number email 或 屬性。如果是這樣，則任何不是自助式帳戶復原主要方法且已啟用 MFA 的屬性都可以供他們使用。
- e. 向使用者提示包含 TOTP 和可用簡訊或電子郵件 MFA 因素 MFAS_CAN_SELECT 的選項 SELECT_MFA_TYPE 的挑戰。
- f. 提示使用者輸入他們為回應 SELECT_MFA_TYPE 挑戰而選取的因素。如果他們成功回應 MFA 挑戰，則會登入。



5. 如果使用者尚未註冊 TOTP 驗證器，或者他們目前有 但 TOTP MFA 已停用，請判斷使用者是否有 email 或 phone_number 屬性。
6. 如果使用者只有電子郵件地址或電話號碼，請判斷該屬性是否也是使用者集區實作來傳送帳戶復原訊息以進行密碼重設的方法。如果為 true，則無法完成需要 MFA 的登入，Amazon Cognito 會傳回錯誤。若要啟用此使用者的登入，您必須新增非復原屬性或為其註冊 TOTP 驗證器。



- a. 如果他們有可用的非復原電子郵件地址或電話號碼，請判斷是否啟用對應的電子郵件或 SMS MFA 因素。

- b. 如果他們有未復原的電子郵件地址屬性，且已啟用電子郵件 MFA，請提示他們提出 EMAIL_OTP 挑戰。如果他們成功回應 MFA 挑戰，則會登入。



- c. 如果他們有未復原的電話號碼屬性，且已啟用 SMS MFA，請提示他們提出 SMS_MFA 挑戰。如果他們成功回應 MFA 挑戰，則會登入。



- d. 如果他們沒有符合已啟用電子郵件或 SMS MFA 因素資格的屬性，請判斷是否已啟用 TOTP MFA。如果停用 TOTP MFA，則無法在需要 MFA 的情況下完成登入，Amazon Cognito 會傳回錯誤。若要啟用此使用者的登入，您必須新增非復原屬性或為其註冊 TOTP 驗證器。



 Note

如果使用者具有 TOTP 驗證器但停用 TOTP MFA，則此步驟已評估為否。

- e. 如果已啟用 TOTP MFA，請在 MFAS_CAN_SETUP選項SOFTWARE_TOKEN_MFA中使用 向使用者提出MFA_SETUP挑戰。若要完成此挑戰，您必須為使用者分別註冊 TOTP 驗證器，並使用回應"ChallengeName": "MFA_SETUP", "ChallengeResponses": {"USERNAME": "[username]", "SESSION": "[Session ID from VerifySoftwareToken]}".
- f. 使用者使用來自 [VerifySoftwareToken](#) 請求的工作階段字符回應MFA_SETUP挑戰後，提示他們提出SOFTWARE_TOKEN_MFA挑戰。如果他們成功回應 MFA 挑戰，則會登入。



- 7. 如果使用者同時擁有電子郵件地址和電話號碼，請判斷是否有任何屬性是密碼重設的帳戶復原訊息的主要方法。

- a. 如果停用自助式帳戶復原，任一屬性都可以用於 MFA。判斷是否已啟用電子郵件和簡訊 MFA 因素之一或兩者。
- b. 如果兩個屬性都啟用為 MFA 因素，請提示使用者SELECT_MFA_TYPE挑戰MFAS_CAN_SELECT選項 SMS_MFA和 EMAIL_OTP。
- c. 提示他們輸入為了回應SELECT_MFA_TYPE挑戰而選取的因素。如果他們成功回應 MFA 挑戰，則會登入。



- d. 如果只有一個屬性是合格的 MFA 因素，請提示它們對剩餘的因素提出挑戰。如果他們成功回應 MFA 挑戰，則會登入。



此結果會在下列案例中發生。

- i. 當它們具有 email 和 phone_number 屬性時，會啟用簡訊和電子郵件 MFA，主要帳戶復原方法是透過電子郵件或簡訊。
 - ii. 當它們具有 email 和 phone_number 屬性時，只會啟用簡訊 MFA 或電子郵件 MFA，並停用自助式帳戶復原。
8. 如果使用者尚未註冊 TOTP 驗證器，且沒有 email 或 phone_number 屬性，請提示他們提出 MFA_SETUP 挑戰。中的清單 MFAS_CAN_SETUP 包含使用者集區中非主要帳戶復原選項的所有已啟用 MFA 因素。他們可以使用 ChallengeResponses 來回應電子郵件或 TOTP MFA 的挑戰。若要設定 SMS MFA，請分別新增電話號碼屬性並重新啟動身分驗證。

對於 TOTP MFA，使用 回應 "ChallengeName": "MFA_SETUP", "ChallengeResponses": {"USERNAME": "[username]", "SESSION": "[Session ID from VerifySoftwareToken]"}。

對於電子郵件 MFA，請以 回應 "ChallengeName": "MFA_SETUP", "ChallengeResponses": {"USERNAME": "[username]", "email": "[user's email address]"}。

- a. 提示他們輸入為了回應 SELECT_MFA_TYPE 挑戰而選取的因素。如果他們成功回應 MFA 挑戰，則會登入。



設定使用者集區以進行多重驗證

您可以在 Amazon Cognito 主控台或使用 [SetUserPoolMfaConfig](#) API 操作和 SDK 方法設定 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. 選擇登入選單。尋找多重要素驗證，然後選擇編輯。

5. 選擇您想要與使用者集區搭配使用的 MFA 強制執行方法。

Edit multi-factor authentication (MFA) [Info](#)

Amazon Cognito has additional authentication factors with SMS messages, email message, and time-based one-time passwords (TOTP).

Multi-factor authentication
Configure secure access to your app by enforcing multi-factor authentication (MFA) during the user sign-in process. MFA settings are applied to all app clients.

MFA enforcement [Info](#)

☐ **Require MFA - Recommended**
Users must provide an additional authentication factor when signing in.

☒ **Optional MFA**
Users can sign in with a single authentication factor, and can choose to add additional authentication factors.

☐ **No MFA**
Users can only sign in with a single authentication factor. This is the least secure option.

MFA methods [Info](#)
Choose the MFA methods that are allowed in your user pool. TOTP-based MFA offers a higher level of security. Recipient message and data rates apply.

☐ **Authenticator apps**
Users can authenticate with a TOTP from an authenticator app such as Authy or Google Authenticator.

☒ **SMS message**
Users can authenticate with a code sent by SMS message to a verified phone number. SMS messages are charged separately by Amazon SNS. [Learn more about pricing](#) [↗](#) This option must be selected because SMS is configured.

☐ **Email message**
Users can authenticate with a code sent in an email message. Email messages are charged separately by Amazon SES. [Learn more about pricing](#) [↗](#)

[Cancel](#) [Save changes](#)

- Require MFA (需要 MFA)。** 您使用者集區中的所有使用者都必須使用額外的簡訊、電子郵件或以時間為基礎的一次性密碼 (TOTP) 程式碼登入，做為額外的身分驗證因素。
 - 選用 MFA。** 您可以提供使用者註冊其他登入因素的選項，但仍允許尚未設定 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 [具有威脅防護的進階安全性](#)。
 - No MFA (沒有 MFA)。** 您的使用者無法註冊額外的登入要素。
- 選擇您應用程式中支援的 MFA methods (MFA 方法)。您可以將電子郵件訊息、簡訊或產生 TOTP 的 Authenticator 應用程式設定為第二個因素。
 - 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。在使用者集區的身分驗證方法選單中，尋找 SMS 並選擇編輯。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](#)。

如果您使用電子郵件訊息做為第二個因素，且您尚未將原始身分設定為搭配 Amazon Simple Email Service (Amazon SES) 用於電子郵件訊息，請在 主控台中建立一個。您必須選擇使用 SES 傳送電子郵件選項。在使用者集區的身分驗證方法功能表中，找到電子郵件，然後選擇編輯。從清單中可用的已驗證身分中選取 FROM 電子郵件地址。如果您選擇已驗證的網域，例如 example.com，您還必須在已驗證的網域中設定 FROM 寄件者名稱，例如 admin-noreply@example.com。

8. 選擇儲存變更。

SMS 和電子郵件訊息 MFA

SMS 和電子郵件 MFA 訊息確認使用者可以存取訊息目的地，然後才能登入。他們確認他們不僅可以存取密碼，還可以存取原始使用者的簡訊或電子郵件收件匣。Amazon Cognito 請求使用者提供短碼，您的使用者集區在成功提供使用者名稱和密碼後傳送。

在使用者將電子郵件地址或電話號碼新增至其設定檔之後，簡訊和電子郵件訊息 MFA 不需要額外的組態。Amazon Cognito 可以將訊息傳送到未經驗證的電子郵件地址和電話號碼。當使用者完成第一個 MFA 時，Amazon Cognito 會將他們的電子郵件地址或電話號碼標記為已驗證。

當具有 MFA 的使用者在應用程式中輸入使用者名稱和密碼時，MFA 身分驗證就會開始。您的應用程式會在叫用 [InitiateAuth](#) 或 [AdminInitiateAuth](#) API 請求的 SDK 方法中提交這些初始參數。API 回應 `ChallengeParameters` 中的 `CODE_DELIVERY_DESTINATION` 一個值，指出授權碼的傳送位置。在您的應用程式中，顯示會提示使用者檢查其電話的表單，並包含程式碼的輸入元素。當他們輸入程式碼時，請在 `challenge-response` API 請求中提交，以完成登入程序。

使用 MFA 的使用者在[受管登入](#)頁面中使用使用者名稱和密碼登入後，系統會自動提示他們輸入 MFA 代碼。

使用者集區會使用 中的 Amazon Simple Notification Service (Amazon SNS) 資源傳送 MFA 和其他 Amazon Cognito 通知的 SMS 訊息 AWS 帳戶。Amazon SNS 同樣地，使用者集區會使用您帳戶中的 Amazon Simple Email Service (Amazon SES) 資源傳送電子郵件訊息。這些連結的服務會在您的訊息交付 AWS 帳單上產生自己的成本。他們也有在生產磁碟區傳送訊息的其他要求。如需詳細資訊，請參閱下列連結：

- [Amazon Cognito 使用者集區的簡訊設定](#)
- [全球簡訊定價](#)
- [Amazon Cognito 使用者集區的電子郵件設定](#)
- [Amazon SES 定價](#)

SMS 和電子郵件訊息 MFA 的考量

- 若要允許使用者使用電子郵件 MFA 登入，您的使用者集區必須具有下列組態選項：
 1. 您的使用者集區中有 Plus 或 Essentials 功能計劃。如需詳細資訊，請參閱[使用者集區功能計劃](#)。
 2. 您的使用者集區會使用您自己的 Amazon SES 資源傳送電子郵件訊息。如需詳細資訊，請參閱[Amazon SES 電子郵件組態](#)。
- MFA 代碼適用於您為應用程式用戶端設定的身分驗證流程工作階段持續時間。

當您編輯應用程式用戶端時，請在應用程式用戶端選單的 Amazon Cognito 主控台中設定身分驗證流程工作階段的持續時間。您也可以使用 `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[範例身分驗證工作階段](#)。

- 當使用者成功從 Amazon Cognito 傳送到未驗證電話號碼或電子郵件地址的簡訊或電子郵件訊息中提供代碼時，Amazon Cognito 會將對應的屬性標記為已驗證。
- 若要讓使用者對與 MFA 相關聯的電話號碼或電子郵件地址值進行自助式變更，他們必須使用存取字符合登入並授權請求。如果他們無法存取目前的電話號碼或電子郵件地址，則無法登入。您的團隊必須使用 [AdminUpdateUserAttributes](#) API 請求中的管理員 AWS 登入資料來變更這些值。
- 在使用者集區中[設定 SMS](#) 之後，您就無法停用 SMS 訊息做為可用的 MFA 因素。

TOTP 軟體權杖 MFA

當您在使用者集區中設定 TOTP 軟體字符合 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 受管登入來登入使用者，您的使用者會提交其使用者名稱和密碼，然後在其他登入頁面上提交 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](#) 來設定 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，且使用者在受管登入中註冊應用程式時，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。

Important

當您的 AWS WAF Web ACL 與使用者集區相關聯，且 Web ACL 中的規則顯示 CAPTCHA 時，這可能會導致受管登入 TOTP 註冊發生無法復原的錯誤。若要建立具有 CAPTCHA 動作且不會影響受管登入 TOTP 的規則，請參閱 [設定您的 AWS WAF Web ACL 以進行受管登入 TOTP MFA](#)。如需 AWS WAF Web ACL 和 Amazon Cognito 的相關資訊，請參閱[將 AWS WAF Web ACL 與使用者集區建立關聯](#)。

若要使用 AWS SDK 和 [Amazon Cognito 使用者集區 API](#) 在自訂建置的 UI 中實作 TOTP MFA，請參閱 [為使用者設定 TOTP 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。
 - 使用 SignUp API 操作或透過受管登入在使用者集區中註冊的使用者，會在使用者完成註冊時收到一次性字符。
 - 在您建立使用者且使用者設定其初始密碼後，Amazon Cognito 會從受管登入向使用者發出一性字符。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 不會向管理員建立的使用者 (這些使用者使用 [InitiateAuth](#) 或 [AdminInitiateAuth](#) API 操作) 發放一次性權杖。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，則使用者必須先設定 MFA，才能使用受管登入來登入。您可以使用來自 MFA_SETUP 的 session 回應值挑戰 [AssociateSoftwareToken](#) 請求中的 [InitiateAuth](#) 或 [AdminInitiateAuth](#) 以取代存取字符。
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 只接受來自使用 HMAC-SHA1 雜湊函數產生程式碼之驗證器應用程式的 TOTPs。使用 SHA-256 雜湊產生的代碼會傳回 Code mismatch 錯誤。

為使用者設定 TOTP 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。您的應用程式或受管登入會在 MFA 挑戰回應中向 Amazon Cognito 顯示 TOTP。

在某些情況下，受管登入會提示新使用者設定 TOTP 驗證器。如需詳細資訊，請參閱 [使用者執行時間的 MFA 邏輯詳細資訊](#)。

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 軟體字符 MFA 時，請呼叫 [AssociateSoftwareToken](#)，為該使用者帳戶傳回唯一產生的共用私密金鑰代碼。您可以使用存取權杖或工作階段字串授權 `AssociateSoftwareToken`。
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。當您解決此問題後，請再次嘗試 [VerifySoftwareToken](#) 操作。

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。每次登入都應呼叫 [RespondToAuthChallenge](#)，以取得對新 TOTP 驗證挑戰的回應。
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，請呼叫 [SetUserMFAPreference](#) 修改您的使用者，使其不使用 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 TOTP MFA 挑戰，請使用 [SetUserMFAPreference](#) 更新使用者的 MFA 偏好設定。
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。

設定您的 AWS WAF Web ACL 以進行受管登入 TOTP MFA

當您的 AWS WAF Web ACL 與使用者集區相關聯，且 Web ACL 中的規則顯示 CAPTCHA 時，這可能會導致受管登入和受管登入 TOTP 註冊時發生無法復原的錯誤。AWS WAF CAPTCHA 規則只會對受管登入和傳統託管 UI 中的 TOTP MFA 產生此影響。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 CAPTCHA 回應使用者集區在背景中提出的 [AssociateSoftwareToken](#) 和 [VerifySoftwareToken](#) API 請求時，就會產生此錯誤。若要建立具有 CAPTCHA 動作且不會影響受管登入頁面中 TOTP 的規則，請在規則中排除 AssociateSoftwareToken VerifySoftwareToken 和的 x-amzn-cognito-operation-name 標頭值。

下列螢幕擷取畫面顯示範例 AWS WAF 規則，會將 CAPTCHA 動作套用至沒有 AssociateSoftwareToken 或 x-amzn-cognito-operation-name 標頭值的所有請求 VerifySoftwareToken。

If a request matches all the statements (AND)

NOT Statement 1

- Field to match
 - Single header (x-amzn-cognito-operation-name)
- Positional constraint
 - Exactly matches string
- Search string
 - AssociateSoftwareToken
- Text transformations
 - None (Priority 0)

AND

NOT Statement 2

- Field to match
 - Single header (x-amzn-cognito-operation-name)
- Positional constraint
 - Exactly matches string
- Search string
 - VerifySoftwareToken
- Text transformations
 - None (Priority 0)

Then

Action
The action to take when a web request matches the rule statement.

如需 AWS WAF Web ACLs 和 Amazon Cognito 的詳細資訊，請參閱 [將 AWS WAF Web ACL 與使用者集區建立關聯](#)。

具有威脅防護的進階安全性

建立使用者集區後，您可以在 Amazon Cognito 主控台的導覽功能表中存取威脅防護。您可以開啟威脅防護功能，並自訂為因應不同風險而採取的動作。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，威脅防護會將指標發佈至 Amazon CloudWatch。您可以在 Amazon Cognito 產生第一個事件後看到指標。請參閱 [檢視威脅防護指標](#)。

威脅防護先前稱為進階安全功能，是一組用於使用者集區中不需要活動的監控工具，以及用於自動關閉潛在惡意活動的組態工具。威脅防護對於標準和自訂身分驗證操作有不同的組態選項。例如，您可能想要傳送通知給具有可疑自訂身分驗證登入的使用者，其中您已設定其他安全因素，但使用基本的使用者名稱密碼身分驗證將使用者封鎖在相同的風險層級。

威脅防護可在 Plus 功能計劃中取得。如需詳細資訊，請參閱 [使用者集區功能計劃](#)。

下列使用者集區選項是威脅防護的元件。

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。您可以在使用者集區中的 username-and-password 標準身分驗證流程中檢查登入資料是否洩露。Amazon Cognito 不會在安全遠端密碼 (SRP) 或自訂身分驗證中偵測洩露的登入資料。

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。在這兩種情況下，Amazon Cognito 都會產生使用者活動日誌，您可以在其中找到事件的詳細資訊。

進一步了解

[使用遭盜用憑證偵測](#)

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。您可以監控使用者活動，並自動回應使用者名稱密碼和 SRP 中偵測到的風險等級，以及自訂身分驗證。

當您啟用威脅防護時，Amazon Cognito 會為使用者活動指派風險分數。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。

進一步了解

[使用自適應身分驗證](#)

IP 地址允許清單和拒絕清單

在全函式模式下使用 Amazon Cognito 威脅保護時，您可以建立 IP 地址永遠封鎖和永遠允許例外狀況。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。

IP 地址允許清單和封鎖清單的須知事項

- 您必須以 CIDR 格式表示 Always 封鎖和 Always 允許，例如 192.0.2.0/24、24 位元遮罩或 192.0.2.252/32、單一 IP 地址。
- 始終封鎖 IP 範圍內 IP 地址的裝置無法使用 SDK 型或受管登入應用程式註冊或登入，但他們可以使用第三方 IdPs 登入。
- 一律允許 和 一律封鎖清單不會影響權杖重新整理。
- Amazon Cognito 不會將自適應身分驗證 MFA 規則套用至來自 Always allow IP range 的裝置，但會套用遭盜用憑證規則。

日誌匯出

威脅防護會記錄使用者對使用者集區的身分驗證請求的精細詳細資訊。這些日誌具有威脅評估、使用者資訊和工作階段中繼資料，例如位置和裝置。您可以建立這些日誌的外部封存，以進行保留和分析。Amazon Cognito 使用者集區會將威脅防護日誌匯出至 Amazon S3、CloudWatch Logs 和 Amazon Data Firehose。如需詳細資訊，請參閱[檢視和匯出使用者事件歷史記錄](#)。

進一步了解

[匯出威脅防護使用者活動日誌](#)

主題

- [威脅防護的考量和限制](#)
- [在使用者集區中開啟威脅防護](#)
- [威脅防護強制執行概念](#)
- [標準身分驗證和自訂身分驗證的威脅防護](#)

- [威脅防護先決條件](#)
- [設定威脅防護](#)
- [使用遭盜用憑證偵測](#)
- [使用自適應身分驗證](#)
- [在應用程式中收集威脅防護的資料](#)

威脅防護的考量和限制

驗證流程之間的威脅防護選項不同

Amazon Cognito 支援使用身分驗證流程 和 進行自適應身分驗證USER_PASSWORD_AUTH和憑證洩露偵測ADMIN_USER_PASSWORD_AUTH。您只能為 啟用適應性身分驗證USER_SRP_AUTH。您無法搭配聯合登入使用威脅防護。

一律封鎖 IPs 有助於請求配額

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](#)。

威脅防護不會套用速率限制

有些惡意流量具有大量請求的特性，例如分散式阻斷服務 (DDoS) 攻擊。Amazon Cognito 套用至傳入流量的風險評分是依請求而定，不會考慮請求量。大量事件中的個別請求可能會因應用程式層原因而收到風險分數和自動回應，這些原因與其在大量攻擊中的角色無關。若要在使用者集區中實作對抗容積攻擊的防禦，請新增 AWS WAF Web ACLs。如需詳細資訊，請參閱[將 AWS WAF Web ACL 與使用者集區建立關聯](#)。

威脅防護不會影響 M2M 請求

用戶端憑證授與適用於機器對機器 (M2M) 授權，而不會與使用者帳戶連線。威脅防護只會監控使用者集區中的使用者帳戶和密碼。若要使用 M2M 活動實作安全功能，請考慮 AWS WAF 監控請求率和內容的功能。如需詳細資訊，請參閱[將 AWS WAF Web ACL 與使用者集區建立關聯](#)。

在使用者集區中開啟威脅防護

Amazon Cognito user pools console

啟用使用者集區的威脅防護

1. 前往 [Amazon Cognito 主控台](#)。如果出現提示，請輸入您的 AWS 登入資料。

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. 如果您尚未啟用，請從設定功能表啟用 Plus 功能計劃。
5. 選擇威脅防護選單，然後選取啟用。
6. 選擇儲存變更。

API

在 [CreateUserPool](#) 或 [UpdateUserPool](#) API 請求中將您的功能計劃設定為 Plus。下列部分範例請求內文會將威脅防護設定為全功能模式。如需完整的範例請求，請參閱[範例](#)。

```
"UserPoolAddOns": {
  "AdvancedSecurityMode": "ENFORCED"
}
```

威脅防護是監控使用者操作是否有帳戶接管跡象並自動回應安全受影響使用者帳戶的功能的集體術語。您可以在使用者使用標準和自訂身分驗證流程登入時，將威脅防護設定套用至使用者。

威脅防護[會產生日誌](#)，詳細說明使用者的登入、登出和其他活動。您可以將這些日誌匯出至第三方系統。如需詳細資訊，請參閱[檢視和匯出使用者事件歷史記錄](#)。

威脅防護強制執行概念

威脅防護會在僅限稽核模式下開始，使用者集區會監控使用者活動、指派風險層級，以及產生日誌。根據最佳實務，在啟用完整函數模式之前，請先以僅限稽核模式執行兩週或更長時間。全功能模式包含一組自動反應，可偵測出有風險的活動和洩露的密碼。使用僅限稽核模式，您可以監控 Amazon Cognito 正在執行的威脅評估。您也可以[提供意見回饋](#)，以訓練誤報和誤報的功能。

您可以在使用者集區層級設定威脅防護強制執行，以涵蓋使用者集區中的所有應用程式用戶端，以及在個別應用程式用戶端層級。應用程式用戶端威脅防護組態會覆寫使用者集區組態。若要設定應用程式用戶端的威脅防護，請從 Amazon Cognito 主控台中使用者集區的應用程式用戶端選單導覽至應用程式用戶端設定。您可以在其中使用用戶端層級設定，並設定應用程式用戶端獨有的強制執行。

此外，您可以針對標準和自訂身分驗證類型分別設定威脅防護。

標準身分驗證和自訂身分驗證的威脅防護

您可以設定威脅防護的方式取決於您在使用者集區和應用程式用戶端中進行的身分驗證類型。下列每種類型的身份驗證都可以有自己的強制執行模式和自動回應。

標準身分驗證

標準身分驗證是使用者登入、登出和密碼管理，以及使用者名稱-密碼流程和受管登入。Amazon Cognito 威脅防護會在操作使用受管登入或使用下列 API AuthFlow 參數登入時，監控操作是否有風險指標：

[InitiateAuth](#)

USER_PASSWORD_AUTH、USER_SRP_AUTH。遭入侵的登入資料功能無法存取USER_SRP_AUTH登入中的密碼，而且不會使用此流程監控或處理事件。

[AdminInitiateAuth](#)

ADMIN_USER_PASSWORD_AUTH、USER_SRP_AUTH。遭入侵的登入資料功能無法存取USER_SRP_AUTH登入中的密碼，而且不會使用此流程監控或處理事件。

您可以將標準身分驗證的強制執行模式設定為僅限稽核或完整函數。若要停用標準身分驗證的威脅監控，請將威脅防護設定為無強制執行。

自訂身分驗證

自訂身分驗證是使用[自訂挑戰 Lambda 觸發條件](#)的使用者登入。您無法在受管登入中執行自訂身分驗證。Amazon Cognito 威脅防護會在操作使用 InitiateAuth和 CUSTOM_AUTH的 API AuthFlow 參數登入時，監控操作是否有風險指標AdminInitiateAuth。

您可以將自訂身分驗證的強制執行模式設定為僅限稽核、完整函數或無強制執行。無強制執行選項會停用自訂身分驗證的威脅監控，而不會影響其他威脅防護功能。

威脅防護先決條件

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](#)。

設定威脅防護

請依照這些指示來設定使用者集區威脅防護。

 Note

若要在 Amazon Cognito 使用者集區主控台中為應用程式用戶端設定不同的威脅防護組態，請從應用程式用戶端功能表中選取應用程式用戶端，然後選擇使用用戶端層級設定。

AWS Management Console

設定使用者集區的威脅防護

1. 前往 [Amazon Cognito 主控台](#)。如果出現提示，請輸入您的 AWS 登入資料。
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. 選擇威脅防護選單，然後選取啟用。
5. 選擇您要設定的威脅防護方法：標準和自訂身分驗證。您可以為自訂和標準身分驗證設定不同的強制執行模式，但它們會在完整函數模式中共用自動回應的組態。
6. 選擇 Edit (編輯)。
7. 選擇強制執行模式。若要立即開始回應偵測到的風險，請選取完整函數，並針對遭到入侵的登入資料和適應性驗證設定自動回應。若要在使用者層級日誌和 CloudWatch 中收集資訊，請選取僅稽核。

我們建議您在啟用動作之前，將威脅防護保持在稽核模式兩週。在此期間，Amazon Cognito 可以了解應用程式使用者的使用模式，並提供事件意見回饋來調整回應。

8. 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。
- 9. 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](#)。

Adaptive authentication messages

Customize the messages sent to users when adaptive authentication triggers a notification. Adaptive authentication messages use [Amazon SES](#).

Email configuration

Configure the [Amazon SES](#) verified identity used to send adaptive authentication messages. [Learn more](#)

SES Region [Info](#)

Choose an AWS Region to use with SES in this user pool. For best performance, you should configure SES and your user pool in the same Region.

US East (N. Virginia) ▼

FROM email address [Info](#)

Choose an email address that you have verified with Amazon SES.

▼

↺

FROM sender name - optional [Info](#)

Enter a friendly name for the email sender in the format "John Stiles <johnstiles@example.com>."

REPLY-TO email address - optional [Info](#)

If you set an invalid reply-to address, sending restrictions may be imposed on your account.

▼ Email templates

Risk detected, sign-in allowed

Email subject [Reset to default](#)

New sign-in attempt

Email message - Text [Reset to default](#) Email message - HTML [Reset to default](#)

We observed an unrecognized sign-in to your

<!DOCTYPE html>

- 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](#)。
10. 展開 IP 地址例外狀況，以建立永遠允許或永遠封鎖清單，列出一律允許或封鎖的 IPv4 或 IPv6 地址範圍，無論威脅防護風險評估為何。以 [CIDR notation](#) (CIDR 表示法) 指定 IP 地址範圍 (如 192.168.100.0/24)。
11. 選擇儲存變更。

API (user pool)

若要設定使用者集區的威脅防護組態，請傳送包含UserPoolId參數但不含ClientId參數的 [SetRiskConfiguration](#) API 請求。以下是使用者集區的範例請求內文。此風險組態會根據風險嚴重性採取一系列遞增的動作，並通知所有風險層級的使用者。它會將遭入侵的憑證區塊套用至註冊操作。

若要強制執行此組態，您必須在個別的 [CreateUserPool](#) 或 [UpdateUserPool](#) API 請求ENFORCED中AdvancedSecurityMode將 設定為 。如需{username}此範例中預留位置範本的詳細資訊，請參閱 [設定驗證和邀請訊息](#)。

```
{
  "AccountTakeoverRiskConfiguration": {
    "Actions": {
      "HighAction": {
        "EventAction": "MFA_REQUIRED",
        "Notify": true
      },
      "LowAction": {
        "EventAction": "NO_ACTION",
        "Notify": true
      },
      "MediumAction": {
        "EventAction": "MFA_IF_CONFIGURED",
        "Notify": true
      }
    },
    "NotifyConfiguration": {
      "BlockEmail": {
        "Subject": "You have been blocked for suspicious activity",
        "TextBody": "We blocked {username} at {login-time} from {ip-address}."
      },
      "From": "admin@example.com",
      "MfaEmail": {
        "Subject": "Suspicious activity detected, MFA required",
        "TextBody": "Unexpected sign-in from {username} on device {device-name}. You must use MFA."
      },
      "NoActionEmail": {
        "Subject": "Suspicious activity detected, secure your user account",
        "TextBody": "We noticed suspicious sign-in activity by {username} from {city}, {country} at {login-time}. If this was not you, reset your password."
      }
    }
  }
}
```



```

        "ReplyTo": "admin@example.com",
        "SourceArn": "arn:aws:ses:us-west-2:123456789012:identity/
admin@example.com"
    },
    "CompromisedCredentialsRiskConfiguration": {
        "Actions": {
            "EventAction": "BLOCK"
        },
        "EventFilter": [ "SIGN_UP" ]
    },
    "RiskExceptionConfiguration": {
        "BlockedIPRangeList": [ "192.0.2.0/24", "198.51.100.0/24" ],
        "SkippedIPRangeList": [ "203.0.113.0/24" ]
    },
    "UserPoolId": "us-west-2_EXAMPLE"
}

```

API (app client)

若要設定應用程式用戶端的威脅防護組態，請傳送包含 `UserPoolId` 參數和 `ClientId` 參數的 [SetRiskConfiguration](#) API 請求。以下是應用程式用戶端的範例請求內文。此風險組態比使用者集區組態更嚴重，會封鎖高風險項目。它也會將遭盜用憑證區塊套用至註冊、登入和密碼重設操作。

若要強制執行此組態，您必須在個別的 [CreateUserPool](#) 或 [UpdateUserPool](#) API 請求 ENFORCED 中 AdvancedSecurityMode 將設定為 `ENFORCED`。如需 {username} 此範例中預留位置範本的詳細資訊，請參閱 [設定驗證和邀請訊息](#)。

```

{
  "AccountTakeoverRiskConfiguration": {
    "Actions": {
      "HighAction": {
        "EventAction": "BLOCK",
        "Notify": true
      },
      "LowAction": {
        "EventAction": "NO_ACTION",
        "Notify": true
      },
      "MediumAction": {
        "EventAction": "MFA_REQUIRED",
        "Notify": true
      }
    }
  }
}

```



```

    },
    "NotifyConfiguration": {
      "BlockEmail": {
        "Subject": "You have been blocked for suspicious activity",
        "TextBody": "We blocked {username} at {login-time} from {ip-address}."
      },
      "From": "admin@example.com",
      "MfaEmail": {
        "Subject": "Suspicious activity detected, MFA required",
        "TextBody": "Unexpected sign-in from {username} on device {device-name}.
You must use MFA."
      },
      "NoActionEmail": {
        "Subject": "Suspicious activity detected, secure your user account",
        "TextBody": "We noticed suspicious sign-in activity by {username} from
{city}, {country} at {login-time}. If this was not you, reset your password."
      },
      "ReplyTo": "admin@example.com",
      "SourceArn": "arn:aws:ses:us-west-2:123456789012:identity/
admin@example.com"
    }
  },
  "ClientId": "1example23456789",
  "CompromisedCredentialsRiskConfiguration": {
    "Actions": {
      "EventAction": "BLOCK"
    },
    "EventFilter": [ "SIGN_UP", "SIGN_IN", "PASSWORD_CHANGE" ]
  },
  "RiskExceptionConfiguration": {
    "BlockedIPRangeList": [ "192.0.2.1/32", "192.0.2.2/32" ],
    "SkippedIPRangeList": [ "192.0.2.3/32", "192.0.2.4/32" ]
  },
  "UserPoolId": "us-west-2_EXAMPLE"
}

```

使用遭盜用憑證偵測

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 會在受管登入和 Amazon Cognito API 中檢查使用使用者名稱和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 主控台的威脅防護選單中，您可以設定遭入侵的登入資料。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。

當您選擇 Allow sign-in (允許登入) 時，您可以檢閱 Amazon CloudWatch Logs 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[檢視威脅防護指標](#)。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 檢查使用 [AdminInitiateAuth](#) API 搭配 ADMIN_USER_PASSWORD_AUTH 流程的登入，以及使用 [InitiateAuth](#) API 搭配 USER_PASSWORD_AUTH 流程的登入，以查看是否有洩漏的憑證。

若要將憑證洩漏保護新增到使用者集區，請參閱 [具有威脅防護的進階安全性](#)。

使用自適應身分驗證

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。此風險分數是根據應用程式提供的裝置和使用因素，以及 Amazon Cognito 從請求衍生的其他因素。Amazon Cognito 風險評估的一些因素包括 IP 地址、使用者代理程式，以及與其他登入嘗試的地理位置距離。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 會將有關登入嘗試、其風險等級和失敗挑戰的指標發佈至 Amazon CloudWatch。如需詳細資訊，請參閱[檢視威脅防護指標](#)。

若要將適應性身分驗證新增到使用者集區，請參閱 [具有威脅防護的進階安全性](#)。

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](#)
- [檢視和匯出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

從 Amazon Cognito 主控台內的威脅防護選單中，您可以選擇適應性身分驗證的設定，包括在不同風險層級要採取的動作，以及自訂傳送給使用者的通知訊息。您可以將全域威脅防護組態指派給所有應用程式用戶端，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 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。

選項	動作
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、登入和重設密碼時，您可以收集使用者工作階段的相關資訊，並將其傳遞給 Amazon Cognito 威脅防護。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，並使用 [ContextData](#) 和 [UserContextData](#) 參數將其提交至 Amazon Cognito 威脅防護。內容資料收集程式庫包含在 AWS SDKs 中。如需詳細資訊，請參閱 [將 Amazon Cognito 身分驗證和授權與 Web 和行動應用程式整合](#)。ContextData 如果您有 Plus 功能計劃，您可以提交。如需詳細資訊，請參閱 [設定威脅防護](#)。

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 未經驗證的 API 操作時，您可以提交 UserContextData 至 Amazon Cognito 威脅防護。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：

- 您的使用者集區位於 Plus 功能計劃中。如需詳細資訊，請參閱 [使用者集區功能計劃](#)。

- 您的應用程式用戶端具有用戶端密碼。如需詳細資訊，請參閱[使用應用程式用戶端的應用程式特定設定](#)。
- 您已在應用程式用戶端中啟動 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：

- 您的使用者僅透過如 [AdminInitiateAuth](#) 等已驗證 API 操作登入，且您使用 ContextData 參數。
- 您只希望未經驗證的 API 操作傳送裝置指紋，而不是 IP 地址給 Amazon Cognito 威脅防護。

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. 選擇應用程式用戶端選單。
4. 選擇或建立應用程式用戶端。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. 在您的應用程式用戶端的 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。

若要在 Amazon Cognito API 中設定應用程式用戶端以接受使用者內容資料，請在 [CreateUserPoolClient](#) 或 [UpdateUserPoolClient](#) 請求中將 `EnablePropagateAdditionalUserContextData` 設定為 `true`。如需有關如何在 Web 或行動應用程式中使用威脅防護的資訊，請參閱 [在應用程式中收集威脅防護的資料](#)。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。下面是使用 JavaScript SDK 方法 `getData` 的範例。

```
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 威脅防護只會儲存並指派風險分數給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 適用於 Android 的 SDK : [GetUserContextData](#)
- AWS Amplify 適用於 iOS 的 SDK : [userContextData](#)
- JavaScript : [amazon-cognito-advanced-security-data.min.js](#)

檢視和匯出使用者事件歷史記錄

當您啟用威脅防護時，Amazon Cognito 會為使用者產生每個身分驗證事件的日誌。根據預設，您可以在 Amazon Cognito 主控台的使用者功能表中或使用 [AdminListUserAuthEvents](#) API 操作來檢視使用者日誌。您也可以將這些事件匯出至外部系統，例如 CloudWatch Logs、Amazon S3 或 Amazon Data Firehose。匯出功能可讓您自己的安全分析系統更容易存取應用程式中使用者活動的安全資訊。

主題

- [檢視使用者事件歷史記錄 \(AWS Management Console\)](#)
- [檢視使用者事件歷史記錄 \(API/CLI\)](#)

- [匯出使用者身分驗證事件](#)

檢視使用者事件歷史記錄 (AWS Management Console)

若要查看使用者的登入歷史記錄，您可以從 Amazon Cognito 主控台的使用者選單中選擇使用者。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[了解使用者集區 JSON Web 字符 JWTs](#)。

檢視使用者事件歷史記錄 (API/CLI)

您可以使用 Amazon Cognito API 操作 [AdminListUserAuthEvents](#) 查詢使用者事件歷史記錄，或使用 [admin-list-user-auth-events](#) 查詢 AWS Command Line Interface (AWS CLI)。

AdminListUserAuthEvents request

的下列請求內文會 AdminListUserAuthEvents 傳回一個使用者的最新活動日誌。

```
{
```



```
"UserPoolId": "us-west-2_EXAMPLE",
"Username": "myexampleuser",
"MaxResults": 1
}
```

admin-list-user-auth-events request

下列 請求會admin-list-user-auth-events傳回一個使用者的最新活動日誌。

```
aws cognito-idp admin-list-user-auth-events --max-results 1 --username myexampleuser
--user-pool-id us-west-2_EXAMPLE
```

Response

Amazon Cognito 會將相同的 JSON 回應內文傳回給這兩個請求。以下是找不到包含風險因素之受管登入事件的範例回應：

```
{
  "AuthEvents": [
    {
      "EventId": "[event ID]",
      "EventType": "SignIn",
      "CreationDate": "[Timestamp]",
      "EventResponse": "Pass",
      "EventRisk": {
        "RiskDecision": "NoRisk",
        "CompromisedCredentialsDetected": false
      },
      "ChallengeResponses": [
        {
          "ChallengeName": "Password",
          "ChallengeResponse": "Success"
        }
      ],
      "EventContextData": {
        "IpAddress": "192.168.2.1",
        "DeviceName": "Chrome 125, Windows 10",
        "Timezone": "-07:00",
        "City": "Bellevue",
        "Country": "United States"
      }
    }
  ],
}
```



```
"NextToken": "[event ID]#[Timestamp]"
}
```

匯出使用者身分驗證事件

設定您的使用者集區，將使用者事件從威脅防護匯出到外部系統。支援的外部系統 – Amazon S3、CloudWatch Logs 和 Amazon Data Firehose – 可能會為您傳送或擷取的資料增加 AWS 帳單成本。如需詳細資訊，請參閱[匯出威脅防護使用者活動日誌](#)。

AWS Management Console

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. 選擇日誌串流功能表。選擇 Edit (編輯)。
5. 在記錄狀態下，選取啟用使用者活動日誌匯出旁的核取方塊。
6. 在記錄目的地下，選擇您要處理日誌的 AWS 服務：CloudWatch 日誌群組、Amazon Data Firehose 串流或 S3 儲存貯體。
7. 您的選擇會將對應的資源類型填入資源選取器。從清單中選擇日誌群組、串流或儲存貯體。您也可以選取建立按鈕，以導覽至所選服務的 AWS Management Console，並建立新的資源。
8. 選取儲存變更。

API

為您的使用者活動日誌選擇一種目的地類型。

以下是將 Firehose 串流設定為日誌目的地的範例 SetLogDeliveryConfiguration 請求內文。

```
{
  "LogConfigurations": [
    {
      "EventSource": "userAuthEvents",
      "FirehoseConfiguration": {
        "StreamArn": "arn:aws:firehose:us-west-2:123456789012:deliverystream/example-user-pool-activity-exported"
      },
      "LogLevel": "INFO"
    }
  ]
}
```

```
],  
  "UserPoolId": "us-west-2_EXAMPLE"  
}
```

以下是將 Amazon S3 儲存貯體設定為日誌目的地SetLogDeliveryConfiguration的請求內文範例。

```
{  
  "LogConfigurations": [  
    {  
      "EventSource": "userAuthEvents",  
      "S3Configuration": {  
        "BucketArn": "arn:aws:s3:::amzn-s3-demo-logging-bucket"  
      },  
      "LogLevel": "INFO"  
    }  
  ],  
  "UserPoolId": "us-west-2_EXAMPLE"  
}
```

以下是將 CloudWatch 日誌群組設定為日誌目的地的SetLogDeliveryConfiguration請求內文範例。

```
{  
  "LogConfigurations": [  
    {  
      "EventSource": "userAuthEvents",  
      "CloudWatchLogsConfiguration": {  
        "LogGroupArn": "arn:aws:logs:us-west-2:123456789012:log-group:D0C-EXAMPLE-LOG-GROUP"  
      },  
      "LogLevel": "INFO"  
    }  
  ],  
  "UserPoolId": "us-west-2_EXAMPLE"  
}
```

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，從使用者功能表中選擇使用者，然後選取提供事件意見回饋。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)。

主控台會在使用者選單的使用者詳細資訊中列出登入歷史記錄。如果您選取一個項目，可以將事件標記為有效或無效。您也可以透過使用者集區 API 操作 [AdminUpdateAuthEventFeedback](#)，以及透過 AWS CLI 命令 [admin-update-auth-event-feedback](#) 提供意見回饋。

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

透過威脅防護，Amazon Cognito 可以通知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。

 Note

Amazon Cognito 只會在其動作產生自動風險回應時傳送通知訊息給使用者：封鎖登入、允許登入、將 MFA 設定為選用，或需要 MFA。有些請求可能獲指派風險層級，但不會產生自適應身分驗證自動風險回應；因此，您的使用者集區不會傳送通知。例如，可能會使用風險評分記錄不正確的密碼，但 Amazon Cognito 的回應是登入失敗，而不是套用自適應身分驗證規則。

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)。

Automatic risk response Info					
Risk level	Allow sign-in	Optional MFA	Require MFA	Block sign-in	Notify user
Low risk	☒	☐	☐	☐	☒
Medium risk	☐	☒	☐	☐	☒
High risk	☐	☐	☒	☐	☒

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，請在威脅防護組態中開啟來自自適應身分驗證訊息的電子郵件範本。若要進一步了解電子郵件範本，請參閱 [訊息範本](#)。

在應用程式中收集威脅防護的資料

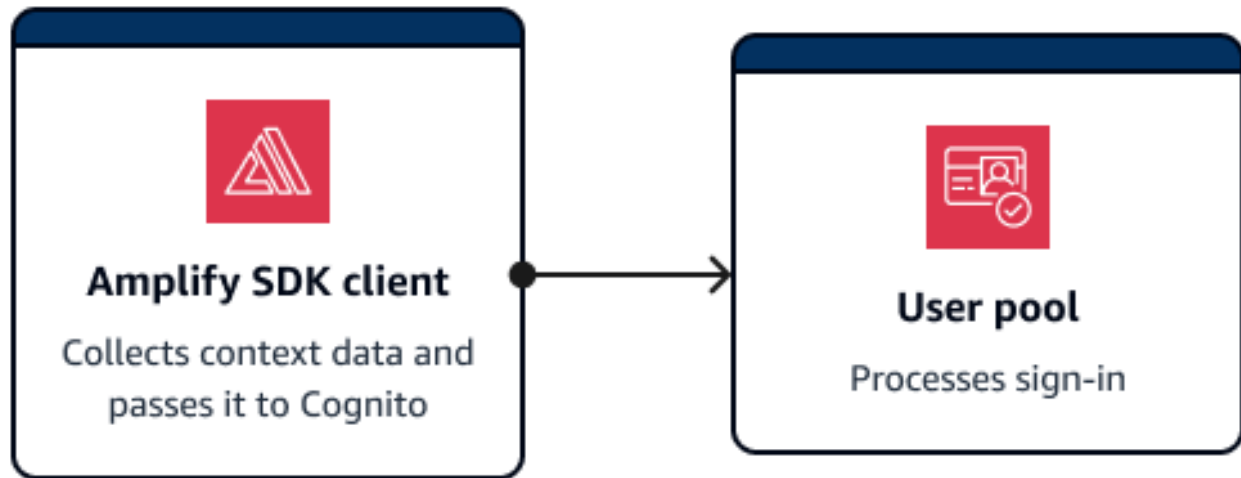
Amazon Cognito [適應性身分驗證](#) 會從使用者登入嘗試的內容詳細資訊中，評估嘗試帳戶接管的風險等級。您的應用程式必須將內容資料新增至 API 請求，Amazon Cognito 威脅防護才能更準確地評估風險。內容資料是 IP 地址、瀏覽器代理程式、裝置資訊和請求標頭等資訊，提供使用者如何連接到使用者集區的相關內容資訊。

向 Amazon Cognito 提交此內容的應用程式的中心責任是對使用者集區的身分驗證請求中的 `EncodedData` 參數。若要將此資料新增至您的請求，您可以使用可自動產生此資訊的開發套件來實作 Amazon Cognito，也可以實作用於 JavaScript、iOS 或 Android 的模組來收集此資料。直接向 Amazon Cognito 提出請求的僅限用戶端應用程式必須實作 AWS Amplify SDKs。具有中繼伺服器或 API 元件的用戶端伺服器應用程式必須實作個別的 SDK 模組。

在下列案例中，您的身分驗證前端會管理使用者內容資料收集，而不需要任何額外的組態：

- 受管登入會自動收集內容資料並將其提交至威脅防護。
- 所有 AWS Amplify 程式庫的身分驗證方法都內建內容資料收集。

使用 Amplify 在僅限用戶端應用程式中提交使用者內容資料



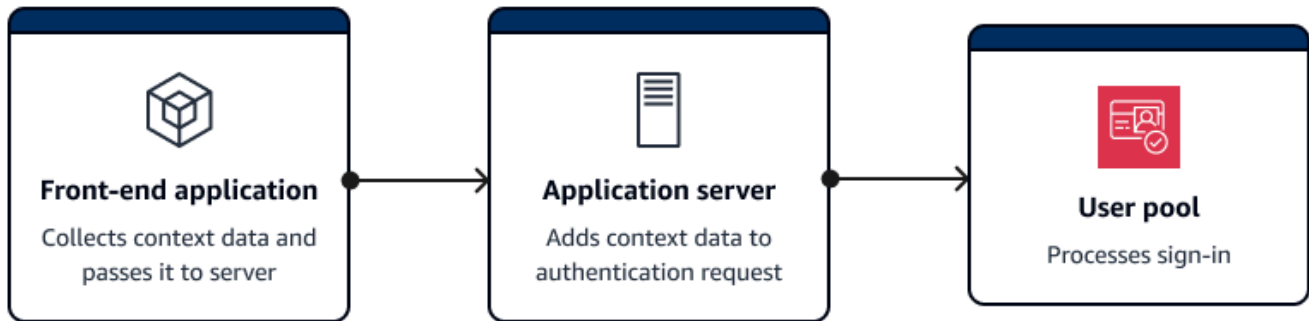
Amplify SDKs支援直接使用 Amazon Cognito 進行身分驗證的行動用戶端。這類用戶端會向 Amazon Cognito 公有 API 操作提出直接 API 請求。Amplify 用戶端預設會自動收集威脅防護的內容資料。

使用 JavaScript 來 Amplify 應用程式是例外狀況。它們需要新增 [JavaScript 模組](#) 來收集使用者內容資料。

一般而言，此組態中的應用程式會使用未經驗證的 API 操作，例如 [InitiateAuth](#) 和 [RespondToAuthChallenge](#)。[UserContextData](#) 物件有助於更準確地評估這些操作的風險。Amplify SDKs會將裝置和工作階段資訊新增至的 `EncodedData` 參數 `UserContextData`。

在用戶端伺服器應用程式中收集內容資料

有些應用程式具有前端層，可收集使用者身分驗證資料，以及向 Amazon Cognito 提交身分驗證請求的應用程式後端層。這是微服務支援的 Web 伺服器和應用程式中常見的架構。在這些應用程式中，您必須匯入公有內容資料收集程式庫。



一般而言，此組態中的應用程式伺服器會使用已驗證的 API 操作，例如 [AdminInitiateAuth](#) 和 [AdminRespondToAuthChallenge](#)。[ContextData](#) 物件可協助 Amazon Cognito 更準確地評估這些操作的風險。的內容ContextData是您的前端傳送到伺服器的編碼資料，以及來自使用者 HTTP 請求到伺服器的其他詳細資訊。這些額外的內容詳細資訊，例如 HTTP 標頭和 IP 地址，為您的應用程式伺服器提供使用者環境的特性。

您的應用程式伺服器也可能使用未經驗證的 API 操作登入，例如 [InitiateAuth](#) 和 [RespondToAuthChallenge](#)。[UserContextData](#) 物件會在這些操作中通知威脅防護風險分析。可用公有內容資料收集程式庫中的操作會將安全資訊新增至身分驗證請求中的 `EncodedData` 參數。此外，將使用者集區設定為接受其他內容資料，並將使用者的來源 IP 新增至的 `IpAddress` 參數 `UserContextData`。

將內容資料新增至用戶端伺服器應用程式

1. 在前端應用程式中，使用 [iOS、Android 或 JavaScript 模組](#) 從用戶端收集編碼的內容資料。
2. 將編碼的資料和身分驗證請求的詳細資訊傳遞給您的應用程式伺服器。
3. 在您的應用程式伺服器中，從 HTTP 請求擷取使用者的 IP 地址、相關的 HTTP 標頭、請求的伺服器名稱和請求的路徑。將這些值填入 Amazon Cognito API 請求的 [ContextData](#) 參數。
4. 使用 SDK 模組收集的編碼裝置資料填入 API 請求ContextData中的 `EncodedData` 參數。將此內容資料新增至身分驗證請求。

用戶端伺服器應用程式的內容資料程式庫

JavaScript

`amazon-cognito-advanced-security-data.min.js` 模組會收集您可以傳遞給應用程式伺服器的 `EncodedData`。

將 `amazon-cognito-advanced-security-data.min.js` 模組新增至您的 JavaScript 組態。
將 `<region>` 取代 AWS 區域 為下列清單中的：`us-east-1`、`us-east-2`、`eu-west-2`、`us-west-2`、`eu-west-1` 或 `eu-central-1`。

```
<script src="https://amazon-cognito-assets.<region>.amazoncognito.com/amazon-cognito-advanced-security-data.min.js"></script>
```

若要產生您可以在 `EncodedData` 參數中使用的 `encodedContextData` 物件，請將以下內容新增至您的 JavaScript 應用程式來源：

```
var encodedContextData = AmazonCognitoAdvancedSecurityData.getData(_username,
    _userpoolId, _userPoolClientId);
```

iOS/Swift

若要產生內容資料，iOS 應用程式可以整合 [適用於 iOS 的 Mobile SDK](#) 模組 [AWSCognitoIdentityProviderASF](#)。

若要收集用於威脅防護的編碼內容資料，請將下列程式碼片段新增至您的應用程式：

```
import AWSCognitoIdentityProviderASF

let deviceId = getDeviceId()
let encodedContextData = AWSCognitoIdentityProviderASF.userContextData(
    userPoolId,
    username: username,
    deviceId: deviceId,
    userPoolClientId: userPoolClientId)

/**
 * Reuse DeviceId from keychain or generate one for the first time.
 */
func getDeviceId() -> String {
    let deviceIdKey = getKeyChainKey(namespace: userPoolId, key:
        "AWSCognitoAuthAsfDeviceId")

    if let existingDeviceId = self.keychain.string(forKey: deviceIdKey) {
        return existingDeviceId
    }

    let newDeviceId = UUID().uuidString
    self.keychain.setString(newDeviceId, forKey: deviceIdKey)
```

```
        return newDeviceId
    }

    /**
     * Get a namespaced keychain key given a namespace and key
     */
    func getKeyChainKey(namespace: String, key: String) -> String {
        return "\(namespace).\(key)"
    }
}
```

Android

若要產生內容資料，Android 應用程式可以整合適用於 [Android 模組 aws-android-sdk-cognitoidentityprovider-asf](#) 的 [Mobile SDK](#)。 [aws-android-sdk-cognitoidentityprovider-asf](#)

若要收集用於威脅防護的編碼內容資料，請將下列程式碼片段新增至您的應用程式：

```
UserContextDataProvider provider = UserContextDataProvider.getInstance();
// context here is android application context.
String encodedContextData = provider.getEncodedContextData(context, username,
    userPoolId, userPoolClientId);
```

將 AWS WAF Web ACL 與使用者集區建立關聯

AWS WAF 是 Web 應用程式防火牆。使用 AWS WAF Web 存取控制清單 (Web ACL)，您可以保護您的使用者集區免於對傳統託管 UI、受管登入和 Amazon Cognito API 服務端點提出不必要的請求。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 \(Web ACL\)](#)。

當您有與使用者集區相關聯的 AWS WAF Web ACL 時，Amazon Cognito 會將來自使用者的所選非機密標頭和請求內容轉送至 AWS WAF。會 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，並將回應傳回給 Amazon Cognito。

AWS WAF Web ACLs 和 Amazon Cognito 的須知事項

- 您無法設定 Web ACL 規則以符合使用者集區請求中的個人身分識別資訊 (PII)，例如使用者名稱、密碼、電話號碼或電子郵件地址。此資料將無法供使用 AWS WAF。反之，請將 Web ACL 規則設定為符合標頭、路徑和內文中的工作階段資料，例如 IP 地址、瀏覽器代理程式和請求的 API 操作。
- Web ACL 中的規則條件必須符合使用者對受管登入的第一個請求。他們的第一個請求通常是 [授權端點](#)。授權端點一律會重新導向請求；請勿在 Web ACL 規則中設定符合的 URL 路徑。

- 封鎖的請求 AWS WAF 不會計入任何請求類型的請求率配額。在 API 層級調節 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。傳播時間可以是幾秒鐘到幾分鐘。當您嘗試在 Web ACL 完全傳播之前建立關聯 [WAFUnavailableEntityException](#) 時，AWS WAF 會傳回。
- 您可以將一個 Web ACL 與每個使用者集區建立關聯。
- 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 [過大請求元件處理](#)，了解如何設定如何處理來自 Amazon Cognito AWS WAF 的過大請求。
- 您無法將使用 AWS WAF [Fraud Control 帳戶接管預防 \(ATP\)](#) 的 Web ACL 與 Amazon Cognito 使用者集區建立關聯。您可於新增 AWS-AmazonManagedRulesATPRuleSet 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。
- 當您的 AWS WAF Web ACL 與使用者集區相關聯，且 Web ACL 中的規則顯示 CAPTCHA 時，這可能會導致受管登入 TOTP 註冊發生無法復原的錯誤。若要建立具有 CAPTCHA 動作且不會影響受管登入 TOTP 的規則，請參閱 [設定您的 AWS WAF Web ACL 以進行受管登入 TOTP MFA](#)。

AWS WAF 會檢查對下列端點的請求。

受管登入和傳統託管 UI

對 [使用者集區端點和受管登入參考](#) 中所有端點的請求。

公有 API 操作

從您的應用程式請求不使用 AWS 登入資料進行授權的 Amazon Cognito API。此包括如 [InitiateAuth](#)、[RespondToAuthChallenge](#) 和 [GetUser](#) 的 API 作業。在範圍內的 API 操作 AWS WAF 不需要使用 AWS 登入資料進行身分驗證。其未經身分驗證，或使用工作階段字串或存取字串進行授權。如需詳細資訊，請參閱[依授權模型分組的 API 操作清單](#)。

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。

- 您可以自訂受管登入請求的錯誤碼和回應內文。您只能為使用者提供 CAPTCHA，以便在受管登入中解決此問題。

- 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 SDKs 會傳回 `ForbiddenException` 錯誤給產生區塊或 CAPTCHA 回應的請求。

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 和 AWS WAF 許可。如需 AWS WAF 許可的相關資訊，請參閱《AWS WAF 開發人員指南》中的 [AWS WAF API 許可](#)。

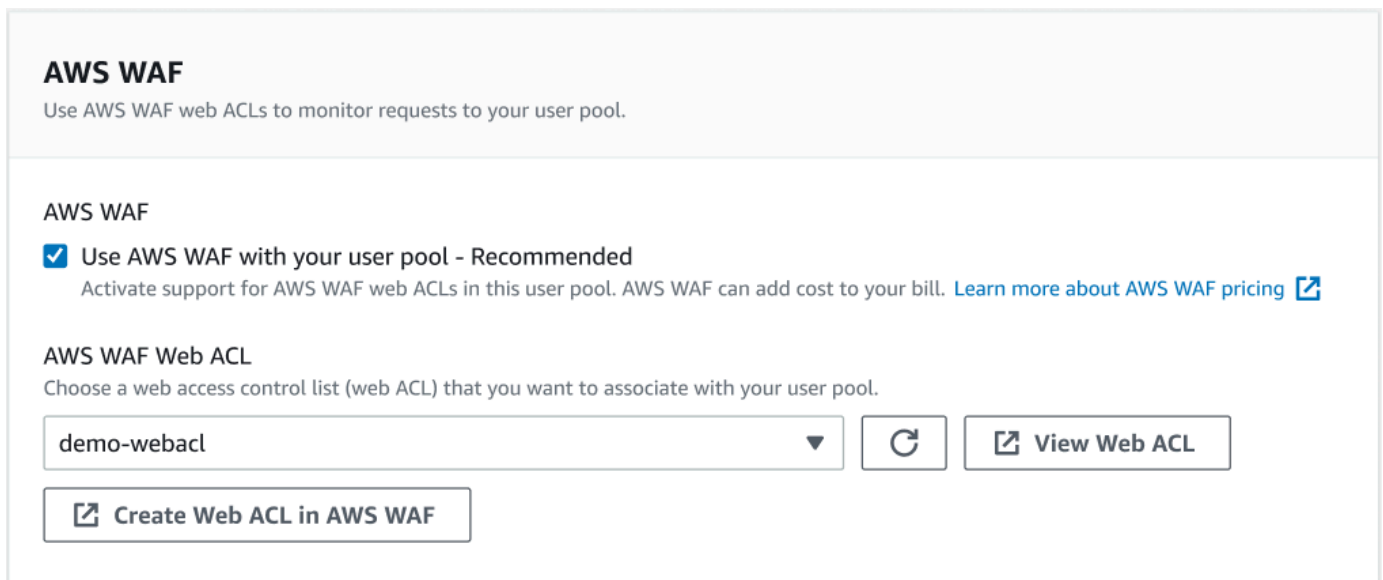
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowWebACLUserPool",
      "Effect": "Allow",
      "Action": [
        "cognito-idp:ListResourcesForWebACL",
        "cognito-idp:GetWebACLForResource",
        "cognito-idp:AssociateWebACL"
      ],
      "Resource": [
        "arn:aws:cognito-idp:*:123456789012:userpool/*"
      ]
    },
    {
      "Sid": "AllowWebACLUserPoolWAFv2",
      "Effect": "Allow",
      "Action": [
        "wafv2:ListResourcesForWebACL",
        "wafv2:AssociateWebACL",
        "wafv2:DisassociateWebACL",
        "wafv2:GetWebACLForResource"
      ],
      "Resource": "arn:aws:wafv2:*:123456789012:*/webacl/*/*"
    },
    {
      "Sid": "DisassociateWebACL1",
      "Effect": "Allow",
      "Action": "wafv2:DisassociateWebACL",
```

```
"Resource": "*"
},
{
  "Sid": "DisassociateWebACL2",
  "Effect": "Allow",
  "Action": [
    "cognito-idp:DisassociateWebACL"
  ],
  "Resource": [
    "arn:aws:cognito-idp:*:123456789012:userpool/*"
  ]
}
]
```

雖然您必須授予 IAM 許可，但列出的動作僅限許可，不會對應至任何 [API 操作](#)。

AWS WAF 為您的使用者集區啟用 並關聯 Web ACL

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. 選擇安全區段中的AWS WAF索引標籤。
4. 選擇編輯。
5. 選取 AWS WAF 搭配您的使用者集區使用。



AWS WAF
Use AWS WAF web ACLs to monitor requests to your user pool.

AWS WAF

☒ Use AWS WAF with your user pool - Recommended
Activate support for AWS WAF web ACLs in this user pool. AWS WAF can add cost to your bill. [Learn more about AWS WAF pricing](#)

AWS WAF Web ACL
Choose a web access control list (web ACL) that you want to associate with your user pool.

demo-webacl ▼   View Web ACL

 Create Web ACL in AWS WAF

6. 選擇您已建立的 AWS WAF Web ACL，或選擇在 中建立 Web ACL AWS WAF，以在 的新 AWS WAF 工作階段中建立一個 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，請從 AWS WAF API 使用 [AssociateWebACL](#)。Amazon Cognito 沒有關聯 Web ACL 的單獨 API 操作。

測試和記錄 AWS WAF Web ACLs

當您將規則動作設定為 Web ACL 中的計數時，會將請求 AWS WAF 新增至符合規則的請求計數。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《開發人員指南》中的[測試和調校您的 AWS WAF 保護](#)。AWS WAF

您也可以設定 AWS WAF 將請求標頭記錄到 Amazon CloudWatch Logs 日誌群組、Amazon Simple Storage Service (Amazon S3) 儲存貯體或 Amazon Data 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。受管登入請求僅包含 x-amzn-cognito-client-id 標頭。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 Web ACLs 可在所有使用者集區[功能計劃](#)中使用。的安全功能可 AWS WAF 補充 Amazon Cognito 威脅防護。您可於使用者集區中啟動這兩項功能。AWS WAF 會單獨用於檢查使用者集區請求的帳單。如需詳細資訊，請參閱[AWS WAF 定價](#)。

記錄 AWS WAF 請求資料需由您以日誌為目標的服務額外計費。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

您在 中建立的 Amazon Cognito AWS Management Console 使用者集區預設不區分大小寫。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。如需唯一識別符區分大小寫和 SAML IdP 的詳細資訊，請參閱 [實作 SP 初始化的 SAML 登入](#)。

建立區分大小寫的使用者集區

如果您使用 [CreateUserPool](#) 等 AWS Command Line Interface (AWS CLI) 和 API 操作建立資源，則必須將布林值 `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`。您在 Amazon Cognito 主控台中建立的使用者集區會區分大小寫。若要產生區分大小寫的使用者集區，您必須使用 `CreateUserPool` 操作。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。

在的登入選單和 [DescribeUserPool](#) 的 `UsernameConfiguration` 屬性中，您可以檢閱帳戶中每個使用者集區的區分大小寫設定。AWS Management Console

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](#)。AWS Lambda 函數可以在不區分大小寫的新使用者集區中建立使用者。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。您還可以針對 [ForgotPassword](#) 事件啟用遷移使用者 Lambda 觸發程序。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。這些衝突在區分大小寫的使用者集區中是唯一的，但在不區分大小寫的使用者集區中是相同的。

如需如何在 Amazon Cognito 使用者集區之間使用遷移使用者 Lambda 觸發程序的詳細資訊，請參閱 AWS 部落格中的 [將使用者遷移至 Amazon Cognito 使用者集區](#)。

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。當您在 中刪除使用者集區時 AWS

Management Console，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。

Delete user pool ? ×

Before you delete this user pool, first make sure no services or apps rely on it.

 If you delete this user pool, and your app still relies on it, any sign-in and sign-up attempts will fail.

- To delete this user pool, permit Amazon Cognito to also take the following prerequisite actions.

Deactivate deletion protection
- To confirm deletion, enter testUserPool in the field.

testUserPool

Cancel Delete

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。若要在您透過 AWS CLI 或 AWS SDK 建立的使用者集區中使用此功能，請將 DeletionProtection 參數設定為 True。

您可以在 Amazon Cognito 主控台的設定選單中的刪除保護容器中啟用或停用刪除保護狀態。

設定刪除保護

- 前往 [Amazon Cognito 主控台](#)。系統可能會提示您輸入 AWS 登入資料。
- 選擇 User 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](#)。
- 選擇設定選單，然後導覽至刪除保護索引標籤。選取啟用或停用。

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，預設為 `PreventUserExistenceErrors` LEGACY 或 停用。在 Amazon Cognito 主控台中，預設 `PreventUserExistenceErrors` 會選取防止使用者存在錯誤 - ENABLED 的設定。若要更新您的 `PreventUserExistenceErrors` 組態，請執行下列其中一項操作：

- 在 [UpdateUserPoolClient](#) API 請求 LEGACY 中變更介於 ENABLED 和 `PreventUserExistenceErrors` 之間的值。
- 在 Amazon Cognito 主控台中編輯您的應用程式用戶端，並變更防止使用者在所選 () 和取消選取 () 之間存在錯誤的狀態 LEGACY。ENABLED

當此屬性的值為 時 LEGACY，當使用者嘗試使用不存在於使用者集區中的使用者名稱登入時，您的應用程式用戶端會傳回 `UserNotFoundException` 錯誤回應。

當此屬性的值為 時 ENABLED，您的應用程式用戶端不會揭露使用者集區中使用者帳戶不存在的 `UserNotFoundException` 錯誤。當您提交使用者名稱不存在的請求時，的 `PreventUserExistenceErrors` 組態 ENABLED 具有以下效果：

- Amazon Cognito 會以非特定資訊回應 API 請求，否則其回應可能會揭露有效的使用者存在。
- Amazon Cognito 會傳回一般身分驗證失敗回應給忘記密碼請求，以及傳回給具有身分驗證流程的身分驗證請求，但 [選擇型身分驗證](#) (USER_AUTH) 除外，例如 USER_SRP_AUTH 或 CUSTOM_AUTH。錯誤回應會告訴您使用者名稱或密碼不正確。
- Amazon Cognito 會使用使用者集區允許的挑戰類型中的隨機選擇來回應以選擇為基礎的身分驗證請求。您的使用者集區可能會傳回通行金鑰、一次性密碼或密碼挑戰。
- Amazon Cognito 帳戶確認和密碼復原 APIs 會傳回回應，指出程式碼已傳送至模擬交付媒體，而不是部分表示使用者的聯絡資訊。

下列資訊詳細說明當 `PreventUserExistenceErrors` 設定為 時，使用者集區操作的行為 ENABLED。

身分驗證和使用者建立操作

您可以在使用者名稱密碼和安全遠端密碼 (SRP) 身分驗證中設定錯誤回應。您也可以自訂使用自訂身分驗證傳回的錯誤。以選擇為基礎的身分驗證不受您的 `PreventUserExistenceErrors` 組態影響。

身分驗證流程中的使用者存在公開詳細資訊

以選擇為基礎的身分驗證

在以 `USER_AUTH` 選擇為基礎的身分驗證流程中，Amazon Cognito 會根據您的使用者集區組態和使用者的屬性，從可用的主要身分驗證因素傳回挑戰。此身分驗證流程可以傳回密碼、安全遠端密碼 (SRP)、WebAuthn (通行金鑰)、簡訊一次性密碼 (OTP) 或電子郵件 OTP 挑戰。使用 `PreventUserExistenceErrors` 作用中時，Amazon Cognito 會對不存在的使用者發出挑戰，以完成一或多個可用的身分驗證形式。使用 `PreventUserExistenceErrors` 非作用中時，Amazon Cognito 會傳回 `UserNotFound` 例外狀況。

使用者名稱和密碼身分驗證

當 `PreventUserExistenceErrors` 處於作用中 `Incorrect username or password` 狀態時 `USER_PASSWORD_AUTH`，身分驗證流程 `ADMIN_USER_PASSWORD_AUTH`、和會 `USER_AUTH` 傳回 `NotAuthorizedException` 具有訊息的 `PASSWORD` 流程。當 `PreventUserExistenceErrors` 處於非作用中狀態時，這些流程會傳回 `UserNotFoundException`。

安全遠端密碼 (SRP) 型身分驗證

最佳實務是，僅在沒有電子郵件地址、電話號碼 `USER_SRP_AUTH` 或偏好的使用者名稱 [別名屬性](#) 的使用者集區 `USER_AUTH` 中實作 `PreventUserExistenceErrors` 或 `PASSWORD_SRP` 流程。具有別名屬性的使用者可能不會受到 SRP 身分驗證流程中的使用者存在抑制。使用者名稱-密碼身分驗證流程 — `ADMIN_USER_PASSWORD_AUTH`、`USER_PASSWORD_AUTH` 和 `USER_AUTHPASSWORD` 挑戰 — 完全禁止來自別名屬性的使用者存在。

當有人嘗試使用應用程式用戶端不知道的使用者名稱登入 SRP 時，Amazon Cognito 會在第一個步驟中傳回模擬回應，如 [RFC 5054](#) 中所述。Amazon Cognito 會針對相同的使用者名稱和使用集區組合，傳回 [UUID](#) 格式的相同 salt 和內部使用者 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。如需 SRP 身分驗證實作的詳細資訊，請參閱 [使用持久性密碼和安全承載登入](#)。

Note

如果您使用以驗證為基礎的別名屬性，而且不可變的使用者名稱未格式化為 [UUID](#)，則可以使用使用者名稱和密碼身分驗證來模擬一般回應。

自訂身分驗證挑戰 Lambda 觸發程序

當使用者嘗試使用身分驗證流程登入，但找不到其使用者名稱時，Amazon Cognito 會叫用自訂CUSTOM_AUTH身分驗證挑戰 Lambda 觸發條件。[???輸入事件](#)包含名為 `布林值` 參數 `UserNotFound`，任何不存在的使用者的值 `true` 為 `。此參數會出現在使用者集區傳送至建立、定義和驗證組成自訂身分驗證架構之驗證挑戰 Lambda 函數的請求事件中。當您在 Lambda 函數的邏輯中檢查此指標時，您可以模擬不存在之使用者的自訂身分驗證挑戰。`

身分驗證前 Lambda 觸發程序

當使用者嘗試登入但找不到使用者名稱時，Amazon Cognito 會叫用身分[驗證前觸發](#)條件。輸入事件包含 `UserNotFound` 參數，值 `true` 為 `，適用於任何不存在的使用者。`

下列清單說明 `PreventUserExistenceErrors` 對使用者帳戶建立的影響。

使用者建立流程中的使用者存在公開詳細資訊

SignUp

`SignUp` 操作一律會在已使用使用者名稱 `UsernameExistsException` 時傳回。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[透過使用者遷移 Lambda 觸發程序匯入使用者](#)。

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"  
  }  
}
```

- 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 lexample23456789 --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`。

使用者集區端點和受管登入參考

Amazon Cognito 有兩種使用者集區身分驗證模型：使用使用者集區 API 和使用 OAuth 2.0 授權伺服器。當您想要在應用程式後端使用 AWS SDK 擷取 OpenID Connect (OIDC) 權杖時，請使用 API。當您想要將使用者集區實作為 OIDC 供應商時，請使用 授權伺服器。授權伺服器新增[聯合登入](#)、[具有存取權杖範圍的 API 和 M2M 授權](#)，以及[受管登入](#)等功能。您可以單獨使用 API 和 OIDC 模型，或在使用者集區層級或[應用程式用戶端](#)層級設定。本節是 OIDC 模型實作的參考。如需兩個身分驗證模型的詳細資訊，請參閱 [了解 API、OIDC 和受管登入頁面身分驗證](#)。

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。其中包括受管登入，您的使用者可以在其中註冊和登入 ([登入端點](#))，以及登出 ([登出端點](#))。如需這些資源的詳細資訊，請參閱 [使用者集區受管登入](#)。

這些頁面也包括公用網路資源，可讓您的使用者集區與第三方 SAML、OpenID Connect (OIDC) 及 OAuth 2.0 身分提供者 (IdP) 通訊。若要使用聯合身分提供者登入使用者，您的使用者必須起始互動式受管登入[登入端點](#)或 OIDC 的請求[授權端點](#)。授權端點會將您的使用者重新導向到您的受管登入頁面或 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。

除了受管登入之外，Amazon Cognito 還與適用於 Android、iOS、JavaScript 等 SDKs 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

請勿鎖定 Amazon Cognito domains 的終端實體或中繼 Transport Layer Security (TLS) 憑證。AWS 會管理所有使用者集區端點和字首網域的所有憑證。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。當您將應用程式鎖定到中繼或分葉憑證時，當 AWS 輪換憑證時，您的應用程式可能會失敗，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

- [使用者互動式受管登入和傳統託管 UI 端點](#)
- [身分提供者和依賴方端點](#)

- [OAuth 2.0 授予](#)
- [在授權碼授予中使用 PKCE](#)
- [受管登入和聯合錯誤回應](#)

使用者互動式受管登入和傳統託管 UI 端點

當您將網域新增至使用者集區時，Amazon Cognito 會啟用本節中的受管登入端點。它們是您的使用者可以完成使用者集區核心身分驗證操作的網頁。包括密碼管理、多重要素驗證 (MFA) 以及屬性驗證的頁面。

組成受管登入的網頁是與客戶互動使用者工作階段的前端 Web 應用程式。您的應用程式必須在使用者的瀏覽器中叫用受管登入。Amazon Cognito 不支援以程式設計方式存取本章中的網頁。[身分提供者和依賴方端點](#) 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](#) 重新導向至受管登入或 IdP 登入頁面，也必須在使用者的瀏覽器中開啟。

所有使用者集區端點都接受來自 IPv4 和 IPv6 來源 IP 地址的流量。

本指南中的主題詳細說明常用的受管登入和傳統託管 UI 端點。受管登入與託管 UI 之間的差異可見，無法運作。除了 `之外/passkeys/add`，所有路徑都會在兩個版本的受管登入品牌之間共用。

在您將網域指派給使用者集區時，Amazon Cognito 還會提供後續網頁。

受管登入端點

端點 URL	描述	如何存取
<code>https://##### /login</code>	登入使用者集區本機和聯合使用者。	從 授權端點 、 <code>/logout</code> 和 <code>/confirmforgotPassword</code> 這類端點重新導向。請參閱 登入端點 。
<code>https://##### /logout</code>	登出使用者集區使用者。	直接連結。請參閱 登出端點 。
<code>https://##### /confirmUser</code>	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。
<code>https://##### /signup</code>	註冊一個新使用者。當您的使用者選擇 Sign up (註冊)，/	使用與 <code>/oauth2/authorize</code> 相同的參數直接連結。

端點 URL	描述	如何存取
	login 頁面會將他們導向到 /signup。	
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。
https://##### domain/passkeys/add	註冊新的 通行金鑰 。僅適用於受管登入。	• 在支援通行金鑰身分驗證的應用程式用戶端確認之後的註冊流程中。 • 使用與 /oauth2/authorize 相同的參數直接連結。

主題

- [受管登入端點：/login](#)
- [受管登入登出端點：/logout](#)

受管登入端點： /login

登入端點是身分驗證伺服器和來自的重新導向目的地[授權端點](#)。當您未指定身分提供者時，這是受管登入的進入點。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

登入端點是受管登入的元件。在您的應用程式中，叫用重新導向至登入端點的聯合和受管登入頁面。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](#)，它會沿著您在初始請求中提供的所有參數傳遞。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
```

範例 – 回應

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。

受管登入登出端點：`/logout`

`/logout` 端點是重新導向端點。它會登出使用者，並重新導向至您應用程式用戶端的授權登出 URL 或 `/login` 端點。GET 請求中對 `/logout` 端點的可用參數是根據 Amazon Cognito 受管登入使用案例量身打造。

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的瀏覽器中調用此和其他受管登入端點。

若要將使用者重新導向至受管登入以再次登入，請將 `redirect_uri` 參數新增至您的請求。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`。

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。

透過 SAML 2.0 IdPs 的單一登出 (SLO)，Amazon Cognito 會先將使用者重新導向至您在 IdP 組態中定義的 SLO 端點。在您的 IdP 將使用者重新導向回 `之後saml2/logout`，Amazon Cognito 會以另一個重新導向來回應 `redirect_uri` 或 `logout_uri` 從您的請求。如需詳細資訊，請參閱[使用單一登出來登出 SAML 使用者](#)。

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 將使用者從工作階段登出，請將他們導向該供應商的登出頁面。

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[使用應用程式用戶端的應用程式特定設定](#)。

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[使用應用程式用戶端的應用程式特定設定](#)。

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。

nonce

(選用) 可新增至請求的隨機值。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)。

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。若要將使用者重新導向至 `/login` 端點以重新驗證權杖並將其傳遞至應用程式，請新增 `redirect_uri` 參數。如果您的 `/logout` 端點請求中同時包含 `logout_uri` 和 `redirect_uri` 參數，Amazon Cognito 會覆寫 `redirect_uri` 參數，並僅處理 `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，Amazon Cognito 會在 `/oauth2/logout` 端點重新導向您的使用者時將其值傳回您的應用程式。

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。若要在 `state` 參數中傳遞符合此格式的字串，請將字串編碼為 base64，然後在應用程式中將其解碼。

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。

請求範例

範例 – 登出並將使用者重新導向至用戶端

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另一個使用者身分登入

當請求省略 `logout_uri` 但以其他方式將構成格式正確的請求的參數提供給授權端點時，Amazon Cognito 會將使用者重新導向至受管登入。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。

您新增至登出請求的其他參數必須在的清單中[請求參數](#)。例如，登出端點不支援具有 `identity_provider` 或 `idp_identifier` 參數的自動 IdP 重新導向。對登出端點的請求 `redirect_uri` 中的參數不是登出 URL，而是您要傳遞至授權端點的 post-sign-in URL。

以下是將使用者登出、重新導向至登入頁面，並在登入 `https://www.example.com` 後提供授權碼給的範例請求。

```
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
```

身分提供者和依賴方端點

聯合端點是使用者集區端點，用於使用者集區使用的其中一個身分驗證標準。其中包括 SAML ACS URLs、OIDC 探索端點，以及做為身分提供者和依賴方的使用者集區角色的服務端點。聯合端點會啟動身分驗證流程、從 IdPs 接收身分驗證證明，以及向用戶端發出權杖。它們會與 IdPs、應用程式和管理員互動，但不會與使用者互動。

此頁面之後的完整頁面主題包含 OAuth 2.0 和 OIDC 提供者端點的詳細資訊，這些端點會在您將網域新增至使用者集區時變成可用。下表是所有聯合端點的清單。

[使用者集區網域](#)的範例如下：

1. 字首網域： `mydomain.auth.us-east-1.amazoncognito.com`
2. 自訂網域： `auth.example.com`

使用者集區聯合端點

端點 URL	描述	如何存取
<code>https://##### /oauth2/authorize</code>	將使用者重新導向至受管登入或使用其 IdP 登入。	在客戶瀏覽器中調用以開始使用者身分驗證。請參閱 授權端點 。
<code>https://##### /oauth2/token</code>	根據授權碼或用戶憑證請求傳回權杖。	應用程式請求擷取字符。請參閱 權杖端點 。

端點 URL	描述	如何存取
<code>https://##### /oauth2/userInfo</code>	根據存取權杖中的 OAuth 2.0 範圍和使用者的身分傳回使用者屬性。	應用程式請求擷取使用者設定檔。請參閱 userInfo 端點 。
<code>https://##### /oauth2/revoke</code>	撤銷重新整理權杖和關聯的存取權杖。	應用程式請求撤銷權杖。請參閱 撤銷端點 。
<code>https://cognito-idp.###.amazonaws.com/##### ID/.well-known/openid-configuration</code>	使用者集區的 OIDC 架構目錄。	應用程式請求尋找使用者集區發行者中繼資料。
<code>https://cognito-idp.###.amazonaws.com/##### ID/.well-known/jwks.json</code>	可用於驗證 Amazon Cognito 權杖的公有金鑰。	應用程式請求驗證 JWTs。
<code>https://Your user pool domain/oauth2/idpresponse</code>	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。
<code>https://##### /saml2/idpresponse</code>	與 SAML 2.0 身分提供者整合的聲明消費者回應 (ACS) URL。	從 SAML 2.0 IdP 重新導向為 ACS URL，或 IdP 起始登入的起始點 ¹ 。
<code>https://##### domain/saml2/logout</code>	與 SAML 2.0 身分提供者整合的 單一登出 (SLO) URL。	從 SAML 2.0 IdP 重新導向為單一登出 (SLO) URL。僅接受 POST 繫結。

¹ 如需 IdP 起始 SAML 登入的詳細資訊，請參閱 [實作 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

- [重新導向和授權端點](#)

- [字符發行者端點](#)
- [使用者屬性端點](#)
- [字符撤銷端點](#)
- [IdP SAML 聲明端點](#)

重新導向和授權端點

`/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](#)。

授權端點會重新導向至受管登入或 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 授權使用者之後，身分驗證伺服器會將瀏覽器重新引導至此 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。

如果您的應用程式將 `state` 參數新增至請求，則當 `/oauth2/authorize` 端點重新導向您的使用者時，Amazon Cognito 會將其值傳回到您的應用程式。

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。若要在 `state` 參數中傳遞符合此格式的字串，請將字串編碼為 base64，然後在應用程式中將其解碼。

identity_provider

選用。

新增此參數以略過受管登入，並將您的使用者重新導向至提供者登入頁面。`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 `identity_provider` 值 Facebook、LoginWithAmazon、Google 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

選用。

新增此參數以透過 `identity_provider` 名稱的替代名稱重新導向至供應商。您可以從 Amazon Cognito 主控台的社交和外部提供者功能表輸入 SAML 2.0 和 OIDC IdPs 的識別符。

scope

選用。

可以合併任何系統預留範圍或與用戶端相關聯的自訂範圍。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

選用。

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

選用。

您從產生的金鑰碼交換 (PKCE) 挑戰證明code_verifier。如需詳細資訊，請參閱[在授權碼授予中使用 PKCE](#)。

僅當您指定 code_challenge_method 參數時才需要。

nonce

選用。

可新增至請求的隨機值。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)。

lang

選用。

您要顯示使用者互動頁面的語言。受管登入頁面可以當地語系化，但託管 UI（傳統）頁面無法。如需詳細資訊，請參閱[受管登入當地語系化](#)。

login_hint

選用。

您要傳遞給授權伺服器的使用者名稱提示。您可以從使用者收集使用者名稱、電子郵件地址或電話號碼，並允許目的地提供者預先填入使用者的登入名稱。當您將login_hint參數和沒有idp_identifier或identity_provider 參數提交至oauth2/authorize端點時，受管登入會以您的提示值填入使用者名稱欄位。您也可以將此參數傳遞至 [登入端點](#)並自動填入使用者名稱值。

當您的授權請求調用重新導向至 OIDC IdPs或 Google 時，Amazon Cognito 會將 login_hint 參數新增至該第三方授權方的請求。您無法將登入提示轉送至 SAML、Apple、Login With Amazon 或 Facebook (Meta) IdPs。

prompt

選用。

OIDC 參數，可控制現有工作階段的身分驗證行為。僅適用於受管登入品牌版本，不適用於傳統託管 UI。如需 OIDC 規格的詳細資訊，請參閱[身分驗證請求](#)。值 `none` 和 `login` 會影響使用者集區身分驗證行為。

使用者透過第三方供應商選取身分驗證時，Amazon Cognito 會將 `prompt` 以外的所有值轉送 `none` 到您的 IdPs。當使用者存取的 URL 包含 `identity_provider` 或 `idp_identifier` 參數時，或當授權伺服器將他們重新導向至 [登入端點](#)，並從可用的按鈕中選取和 IdP 時，就會發生這種情況。

提示參數值

prompt=none

對於具有有效驗證工作階段的使用者，Amazon Cognito 會無提示地繼續身分驗證。使用此提示，使用者可以在使用者集區中的不同應用程式用戶端之間進行無提示驗證。如果使用者尚未驗證，授權伺服器會傳回 `login_required` 錯誤。

prompt=login

Amazon Cognito 要求使用者重新驗證，即使他們有現有的工作階段。當您想要再次驗證使用者的身分時，請傳送此值。具有現有工作階段的已驗證使用者可以返回登入，而不會使該工作階段失效。當具有現有工作階段的使用者再次登入時，Amazon Cognito 會為其指派新的工作階段 Cookie。此參數也可以轉送到您的 IdPs。接受此參數的 IdPs 也會向使用者請求新的身分驗證嘗試。

prompt=select_account

此值不會影響本機登入，且必須在重新導向至 IdPs 請求中提交。當包含在授權請求中時，此參數會 `prompt=select_account` 新增至 IdP 重新導向目的地的 URL 路徑。當 IdPs 支援此參數時，他們會請求使用者選取要登入的帳戶。

prompt=consent

此值不會影響本機登入，且必須在重新導向至 IdPs 請求中提交。當包含在授權請求中時，此參數會 `prompt=consent` 新增至 IdP 重新導向目的地的 URL 路徑。當 IdPs 支援此參數時，他們會在重新導向回您的使用者集區之前請求使用者同意。

當您從請求中省略 `prompt` 參數時，受管登入會遵循預設行為：使用者必須登入，除非其瀏覽器具有有效的受管登入工作階段 Cookie。您可以結合的多個值 `prompt` 與空格字元分隔符號，例如 `prompt=login consent`。

範例：授權碼授予

這是授權碼授予的範例請求。

下列請求會啟動工作階段，以擷取您的使用者在`redirect_uri`目的地傳送到您應用程式的授權碼。此工作階段會請求使用者屬性的範圍，以及存取 Amazon Cognito 自助式 API 操作的範圍。

```
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
```

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。授權碼的有效時間為五分鐘。

```
HTTP/1.1 302 Found
Location: https://www.example.com?code=a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111&state=abcdefg
```

範例：使用 PKCE 授予授權碼

此範例流程會使用 [PKCE](#) 執行授權碼授予。

此請求會新增`code_challenge`參數。若要完成交換字符的程式碼，您必須在對`/oauth2/token`端點的請求中包含`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...
```

授權伺服器會使用授權碼和狀態重新導向回您的應用程式。您的應用程式會處理授權碼，並將其交換為字符。

```
HTTP/1.1 302 Found
```

Location: `https://www.example.com?code=a1b2c3d4-5678-90ab-cdef-EXAMPLE11111&state=abcdefg`

範例：需要使用 重新驗證 `prompt=login`

下列請求會新增 `prompt=login` 參數，要求使用者再次驗證，即使他們有現有的工作階段。

```
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&
prompt=login
```

授權伺服器會重新導向至[登入端點](#)，需要重新驗證。

```
HTTP/1.1 302 Found Location: https://mydomain.auth.us-east-1.amazoncognito.com/
login?response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com&state=abcdefg&scope=openid+profile
+aws.cognito.signin.user.admin&prompt=login
```

範例：使用 進行無提示身分驗證 `prompt=none`

下列請求會新增 `prompt=none` 參數，以無提示方式檢查使用者是否有有效的工作階段。

```
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&
prompt=none
```

當沒有有效的工作階段時，授權伺服器會將錯誤傳回至重新導向 URI

```
HTTP/1.1 302 Found Location: https://www.example.com?error=login_required&state=abcdefg
```

當有效的工作階段存在時，授權伺服器會傳回授權碼。

```
HTTP/1.1 302 Found Location: https://www.example.com?
code=AUTHORIZATION_CODE&state=abcdefg
```

範例：無 **openid** 範圍的字符（隱含）授予

此範例流程會產生隱含授予，並將 JWTs 直接傳回至使用者在工作階段。

請求用於授權伺服器的隱含授予。它會請求存取字符中的範圍，以授權使用者設定檔自助式操作。

```
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
```

授權伺服器只會使用存取字符重新導向回您的應用程式。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。

```
HTTP/1.1 302 Found
Location: https://example.com/
callback#access_token=eyJra456defEXAMPLE&token_type=bearer&expires_in=3600&state=STATE
```

範例：具有 **openid** 範圍的字符（隱含）授予

此範例流程會產生隱含授予，並將權杖傳回至使用者的瀏覽器。

請求用於授權伺服器的隱含授予。它會請求存取字符中的範圍，以授權存取使用者屬性和自助式操作。

```
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
```

授權伺服器會使用存取字符和 ID 字符（因為包含 **openid** 範圍）重新導向回您的應用程式：

```
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。負面請求隨附 HTTP 錯誤代碼和描述，您可以用來更正請求參數。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，身分驗證伺服器會將錯誤重新導向至用戶端的 `redirect_uri` 並在 URL 參數中附加錯誤訊息。以下是格式不正確的範例。
 - 請求不包含 `response_type` 參數。
 - 授權請求提供 `code_challenge` 參數，但不提供 `code_challenge_method` 參數。
 - `code_challenge_method` 參數的值不是 S256。

以下是對格式不正確之範例請求的回應。

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_request
```

- 如果用戶端在 token 中請求 `code` 或 `response_type`，但沒有這些請求的許可，Amazon Cognito 授權伺服器會傳回 `unauthorized_client` 用戶端的 `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
```

- 如果伺服器中有任何非預期的錯誤，身分驗證伺服器 `server_error` 會傳回用戶端的 `redirect_uri`。由於 HTTP 500 錯誤不會傳送到用戶端，因此錯誤不會顯示在使用者的瀏覽器中。授權伺服器會傳回下列錯誤。

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=server_error
```

- 當 Amazon Cognito 透過聯合身分驗證第三方 IdPs 時，Amazon Cognito 可能會遇到連線問題，如下所示：
 - 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
```

- 如果在呼叫 `jwks_uri` 端點進行 ID 字符驗證時發生連線逾時，身分驗證伺服器會將錯誤重新導向至用戶端的 `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
```

- 透過與第三方 IdPs 聯合進行身分驗證時，供應商可能會傳回錯誤回應。這可能是由於組態錯誤或其他原因，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，身分驗證伺服器會使用下列 `redirect_uri` 任一訊息，將錯誤重新導向至用戶端的：

```
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
```

字符發行者端點

OAuth 2.0 [權杖端點](#)會向想要完成授權碼和用戶端憑證授予流程的應用程式/`/oauth2/token`發出 JSON Web 權杖 (JWTs)。這些字符是使用使用者集區進行身分驗證的最終結果。其中包含有關使用者 (ID 字符)、使用者存取層級 (存取字符) 以及使用者保留登入工作階段 (重新整理字符) 的權利的資訊。依賴第三方程式庫的 OpenID Connect (OIDC) 會處理來自此端點的請求和回應承載。字符提供可驗證的身分驗證證明、設定檔資訊，以及存取後端系統的機制。

您的使用者集區 OAuth 2.0 授權伺服器會從權杖端點發出 JSON Web 權杖 (JWTs) 至下列工作階段類型：

1. 已完成授權碼授予請求的使用者。成功兌換程式碼會傳回 ID、存取和重新整理權杖。
2. 已完成用戶端憑證授予的機器對機器 (M2M) 工作階段。使用用戶端秘密成功授權傳回存取權杖。
3. 先前已登入並收到重新整理字符的使用者。重新整理權杖身分驗證會傳回新的 ID 和存取權杖。

Note

在受管登入或透過聯合身分使用授權碼授予登入的使用者，一律可以從字符端點重新整理其字符。使用 API 操作登入，InitiateAuth並在使用者集區中未啟用[記住的裝置](#)時，AdminInitiateAuth使用權杖端點重新整理權杖的使用者。如果記住的裝置處於作用中狀態，請使用應用程式用戶端的[相關 API 或 SDK 字符重新整理操作來重新整理字符](#)。

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。為了應用程式安全，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 PKCE 的詳細資訊，請參閱 [IETF RFC 7636](#)。

您可以在進一步了解使用者集區應用程式用戶端及其授予類型、用戶端秘密、允許的範圍和用戶端 IDs[使用應用程式用戶端的應用程式特定設定](#)。您可以在進一步了解 M2M 授權、用戶端憑證授予，以及具有存取權杖範圍的授權[範圍](#)、M2M 和具有資源伺服器的 APIs。

若要從使用者存取權杖擷取有關使用者的資訊，請將其傳遞給您的 [userInfo 端點](#)或 [GetUser](#) API 請求。存取權杖必須包含這些請求的適當範圍，

對字符端點格式化 POST 請求

`/oauth2/token` 端點僅支援 HTTPS POST。此端點不是使用者互動的。在您的應用程式中使用 [OpenID Connect \(OIDC\) 程式庫](#)處理字符請求。

權杖端點支援 `client_secret_basic` 和 `client_secret_post` 身分驗證。如需 OIDC 規格的詳細資訊，請參閱[用戶端身分驗證](#)。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

您可以在請求的標頭中將下列參數傳遞至字符端點。

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。您也可以將請求的內文加入 `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` 具有用戶端秘密的應用程式用戶端的授權標頭 `abcdef01234567890`，使用字串 Base64-encoded 版本 `djc98u3jiedmi283eu928:abcdef01234567890`：

```
Authorization: Basic ZGpjOTI4M2ppZWRTaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw
```

Content-Type

將此參數的值設為 `'application/x-www-form-urlencoded'`。

內文中的請求參數

以下是您可以在權杖端點的請求內文中以 `x-www-form-urlencoded` 格式請求的參數。

grant_type

「必要」。

您要請求的 OIDC 授予類型。

必須是 `authorization_code`、`refresh_token` 或 `client_credentials`。在以下條件下，您可以從字符端點請求自訂範圍的存取字符：

- 您已在應用程式用戶端組態中啟用請求的範圍。
- 您使用用戶端秘密設定應用程式用戶端。

- 您可以在應用程式用戶端中啟用用戶端憑證授予。

 Note

字符端點只會在 `grant_type` 為 `refresh_token` 時傳回重新整理字符 `authorization_code`。

client_id

「選用」。當您在 *Authorization* 標頭中提供應用程式用戶端 ID 時，不需要。

使用者集區中應用程式用戶端的 ID。指定驗證您使用者的相同應用程式用戶端。

如果用戶端是公有且沒有秘密或授權 `client_secret` 中有，您必須提供此參數 `client_secret_post`。

client_secret

「選用」。當您在 *Authorization* 標頭中提供用戶端秘密，以及應用程式用戶端沒有秘密時，則不需要。

如果應用程式用戶端有一個 `client_secret_post` 授權的應用程式用戶端秘密。

scope

「選用」。

可以是與您的應用程式用戶端相關聯的任何範圍的組合。Amazon Cognito 會忽略請求中對請求應用程式用戶端不允許的範圍。如果您未提供此請求參數，授權伺服器會傳回存取字符 `scope` 宣告，其中包含您在應用程式用戶端組態中啟用的所有授權範圍。您可以請求請求應用程式用戶端允許的任何範圍：標準範圍、資源伺服器的自訂範圍，以及 `aws.cognito.signin.user.admin` 使用者自助服務範圍。

redirect_uri

「選用」。client-credentials 授予不需要。

必須是在 `/oauth2/authorize` 中用來取得 `authorization_code` 的相同 `redirect_uri`。

如果 `grant_type` 是 `refresh_token`，您必須提供此參數 `authorization_code`。

refresh_token

「選用」。僅在使用者已有重新整理權杖，並希望取得新的 ID 和存取權杖時使用。

若要為使用者工作階段產生新的存取和 ID 字串，請將 `refresh_token` 的值設定為請求的應用程式用戶端發出的 `refresh_token` 有效重新整理字串。

當重新整理權杖輪換作用中時，傳回具有新 ID 和存取權杖的新重新整理權杖，否則僅傳回 ID 和存取權杖。

code

「選用」。僅在授權碼授予中需要。

授權碼授予的授權碼。如果您的授權請求包含 `grant_type` 的 `authorization_code`，您必須提供此參數 `authorization_code`。

aws_client_metadata

「選用」。

您要在產生權杖前 [Lambda 觸發程序 machine-to-machine\(M2M\)](#) 授權流程中傳遞至的資訊。您的應用程式可以收集工作階段的相關內容資訊，並在此參數中傳遞。當您 `aws_client_metadata` 以 URL 編碼的 JSON 格式傳遞時，Amazon Cognito 會在輸入事件中將其包含在觸發器 Lambda 函數中。您的權杖前觸發事件版本或全域 Lambda 觸發版本必須設定為第 3 版或更新版本。雖然 Amazon Cognito 接受授權碼和用戶端憑證 M2M 流程中對此端點的請求，但您的使用者集區只會從用戶端憑證請求傳遞 `aws_client_metadata` 至權杖產生前觸發。

code_verifier

選用。只有在您在初始授權請求中提供 `code_challenge_method` 和 `code_challenge` 參數時才需要。

您的應用程式在 [PKCE](#) 的授權碼授予請求中計算 `code_challenge` 的產生程式碼驗證器。

將授權碼交換成權杖

以下請求會在使用授權碼授予進行身分驗證後成功產生 ID、存取和重新整理權杖。請求會以 `Authorization` 標頭中的 `client_secret_basic` 格式傳遞用戶端秘密。

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token&
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpj0Th1M2ppZWRTaTI4M2V1OTI4OmFiY2RLZjAxMjM0NTY3ODkw

grant_type=authorization_code&
client_id=1example23456789&
```

```
code=AUTHORIZATION_CODE&
redirect_uri=com.myclientapp://myclient/redirect
```

回應會向使用者發出新的 ID、存取和重新整理權杖，以及額外的中繼資料。

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_token": "eyJra1example",
  "id_token": "eyJra2example",
  "refresh_token": "eyJj3example",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

具有基本授權的用戶端憑證

來自 M2M 應用程式的下列請求會請求用戶端憑證授予。由於用戶端登入資料需要用戶端秘密，因此請求會使用衍生自應用程式用戶端 ID 和秘密的 Authorization 標頭進行授權。請求會產生具有兩個請求範圍的存取權杖。請求也包含提供 IP 地址資訊的用戶端中繼資料，以及向代表此授予的使用者發出的權杖。Amazon Cognito 會將用戶端中繼資料傳遞至權杖產生前 Lambda 觸發器。

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpj0Th1M2ppZWRTaTI4M2V10TI40mFiY2RLZjAxMjM0NTY3ODkw

grant_type=client_credentials&
client_id=1example23456789&
scope=resourceServerIdentifier1%2Fscope1%20resourceServerIdentifier2%2Fscope2&
&aws_client_metadata=%7B%22onBehalfOfToken%22%3A%22eyJra789ghiEXAMPLE%22,
%20%22ClientIpAddress%22%3A%22192.0.2.252%22%7D
```

Amazon Cognito 會將下列輸入事件傳遞給權杖產生前 Lambda 觸發器。

```
{
  version: '3',
  triggerSource: 'TokenGeneration_ClientCredentials',
  region: 'us-east-1',
  userPoolId: 'us-east-1_EXAMPLE',
  userName: 'ClientCredentials',
```

```

callerContext: {
  awsSdkVersion: 'aws-sdk-unknown-unknown',
  clientId: '1example23456789'
},
request: {
  userAttributes: {},
  groupConfiguration: null,
  scopes: [
    'resourceServerIdentifier1/scope1',
    'resourceServerIdentifier2/scope2'
  ],
  clientMetadata: {
    'onBehalfOfToken': 'eyJra789ghiEXAMPLE',
    'ClientIpAddress': '192.0.2.252'
  }
},
response: { claimsAndScopeOverrideDetails: null }
}

```

回應會傳回存取字符。用戶端憑證授予適用於machine-to-machine(M2M) 授權，且僅傳回存取權杖。

```

HTTP/1.1 200 OK
Content-Type: application/json
{
  "access_token": "eyJra1example",
  "token_type": "Bearer",
  "expires_in": 3600
}

```

具有 POST 內文授權的用戶端憑證

下列 client-credentials 授予請求在請求內文中包含 client_secret 參數，且不包含 Authorization 標頭。此請求使用 client_secret_post 授權語法。請求會產生具有請求範圍的存取權杖。請求也包含提供 IP 地址資訊的用戶端中繼資料，以及向代表此授予的使用者發出的權杖。Amazon Cognito 會將用戶端中繼資料傳遞至權杖產生前 Lambda 觸發器。

```

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%2Fmy_custom_scope&
client_secret=9example87654321&
aws_client_metadata=%7B%22onBehalfOfToken%22%3A%22eyJra789ghiEXAMPLE%22,%20%22ClientIpAddress%22%3A%22192.0.2.252%22%7D
```

Amazon Cognito 會將下列輸入事件傳遞給權杖產生前 Lambda 觸發器。

```
{
  version: '3',
  triggerSource: 'TokenGeneration_ClientCredentials',
  region: 'us-east-1',
  userPoolId: 'us-east-1_EXAMPLE',
  userName: 'ClientCredentials',
  callerContext: {
    awsSdkVersion: 'aws-sdk-unknown-unknown',
    clientId: '1example23456789'
  },
  request: {
    userAttributes: {},
    groupConfiguration: null,
    scopes: [
      'resourceServerIdentifier1/my_custom_scope'
    ],
    clientMetadata: {
      'onBehalfOfToken': 'eyJra789ghiEXAMPLE',
      'ClientIpAddress': '192.0.2.252'
    }
  },
  response: { claimsAndScopeOverrideDetails: null }
}
```

回應會傳回存取字符。用戶端憑證授予適用於machine-to-machine(M2M) 授權，且僅傳回存取權杖。

```
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"
}
```

使用 PKCE 授予授權碼

下列範例請求會使用 [PKCE](#) 完成授權請求，其中包含授權碼授予請求中的 `code_challenge_method` 和 `code_challenge` 參數。

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpj0Th1M2ppZWRTaTI4M2V1OTI4OmFiY2RLZjAxMjM0NTY3ODkw

grant_type=authorization_code&
client_id=1example23456789&
code=AUTHORIZATION_CODE&
code_verifier=CODE_VERIFIER&
redirect_uri=com.myclientapp://myclient/redirect
```

回應會從應用程式成功的 PKCE 驗證傳回 ID、存取和重新整理權杖。

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_token": "eyJra1example",
  "id_token": "eyJra2example",
  "refresh_token": "eyJj3example",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

無需重新整理權杖輪換的權杖重新整理

下列範例請求會將重新整理權杖提供給[重新整理權杖輪換](#)為非作用中的應用程式用戶端。由於應用程式用戶端具有用戶端秘密，因此請求會提供 `Authorization` 標頭。

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpj0Th1M2ppZWRtaTI4M2V1OTI4OmFiY2RLZjAxMjM0NTY3ODkw

grant_type=refresh_token&
client_id=1example23456789&
refresh_token=eyJj3example
```

回應會傳回新的 ID 和存取權杖。

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_token": "eyJra1example",
  "id_token": "eyJra2example",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

使用重新整理權杖輪換進行權杖重新整理

下列範例請求提供重新整理權杖給啟用[重新整理權杖輪換](#)的應用程式用戶端。由於應用程式用戶端具有用戶端秘密，因此請求會提供 Authorization 標頭。

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpj0Th1M2ppZWRtaTI4M2V1OTI4OmFiY2RLZjAxMjM0NTY3ODkw

grant_type=refresh_token&
client_id=1example23456789&
refresh_token=eyJj3example
```

回應會傳回新的 ID、存取和重新整理權杖。

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "access_token": "eyJra1example",
  "id_token": "eyJra2example",
```



```
{
  "refresh_token": "eyJj4example",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

負向回應的範例

格式不正確的請求會從字端點產生錯誤。以下是字端點產生錯誤時回應內文的一般映射。

```
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。

使用者屬性端點

當 OIDC 發出包含使用者屬性的 ID 字符時，OAuth 2.0 會實作 `/oauth2/userInfo` 端點。已驗證的使用者或用戶端會收到具有 `scopes` 宣告的存取字符。此宣告會決定授權伺服器應傳回的屬性。當應用程式向 `userInfo` 端點顯示存取字符時，授權伺服器會傳回回應內文，其中包含存取字符範圍所設定邊界內的使用者屬性。您的應用程式可以從 `userInfo` 端點擷取有關使用者的資訊，只要它擁有至少具有 `openid` 範圍宣告的有效存取字符。

`userInfo` 端點是 OpenID Connect (OIDC) [userInfo 端點](#)。當服務提供者顯示權杖端點發出的存取權杖時，它會回應使用者屬性。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。當您的使用者驗證該 IdP 時，Amazon Cognito 會以無提示方式與 IdP token 端點交換授權碼。您的使用者集區會傳遞 IdP 存取權杖，以授權從 IdP `userInfo` 端點擷取使用者資訊。

使用者存取權杖中的範圍取決於身分驗證請求中的 `scopes` 請求參數，或權杖產生前 [Lambda 觸發](#) 程序新增的範圍。您可以解碼存取權杖並檢查 `scope` 宣告，以查看它們包含的存取控制範圍。以下是會影響從 `userInfo` 端點傳回之資料的一些範圍組合。預留的 Amazon Cognito 範圍 `aws.cognito.signin.user.admin` 不會影響從此端點傳回的資料。

存取字符中的範例範圍及其對 **userInfo** 回應的影響

openid

傳回包含應用程式用戶端可讀取之所有使用者屬性的回應。

openid profile

傳回使用者屬性

`name`、`family_name`、`given_name`、`middle_name`、`nickname`、`preferred_username`、`profile`、`zoneinfo`、`locale` 和 `updated_at`。也會傳回 [自訂屬性](#)。在沒有每個屬性讀取存取權的應用程式用戶端中，此範圍的回應是應用程式用戶端具有讀取存取權之規格內的所有屬性。

openid email

傳回基本設定檔資訊和 `email` 和 `email_verified` 屬性。

openid phone

傳回基本設定檔資訊和 `phone_number` 和 `phone_number_verified` 屬性。

GET /oauth2/userInfo

您的應用程式會直接產生對此端點的請求，而不是透過瀏覽器。

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](#)
- [範例 – 請求](#)
- [範例 – 正面回應](#)
- [負面回應範例](#)

標頭中的請求參數

Authorization: Bearer *<access_token>*

在授權標頭欄位中傳遞存取權杖。

必要。

範例 – 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
```

範例 – 正面回應

```
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

範例 – 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

請求缺少必要的參數、包含不支援的參數值，或格式不正確。

範例 – 錯誤的字符

```
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

存取權杖已過期、已撤銷、格式錯誤或無效。

字符撤銷端點

在工作階段中保留重新整理字符的使用者具有類似瀏覽器 Cookie 的內容。只要重新整理權杖有效，他們可以續約現有的工作階段。您的應用程式可以使用重新整理權杖來取得新的有效權杖，而不是在其 ID 或存取權杖過期後提示使用者登入。不過，您可能會從外部判斷使用者的工作階段應該結束，或者使用者可能會選擇忘記目前的工作階段。此時，您可以撤銷該重新整理字符，使其無法再保留其工作階段。

`/oauth2/revoke` 端點會撤銷 Amazon Cognito 最初以您提供的重新整理字符發出的使用者存取字符。此端點也會撤銷重新整理字符本身，以及來自相同重新整理字符的所有後續存取和身分字符。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

如果您的應用程式用戶端具有用戶端秘密，應用程式必須透過基本 HTTP 授權在授權標頭 `client_secret` 中傳遞其 `client_id` 和 `client_secret`。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) 用戶端想要撤銷的重新整理字符。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

(選用) 您要撤銷之字符的應用程式用戶端 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

此撤銷請求會撤銷沒有用戶端秘密之應用程式用戶端的重新整理字符。記下請求內文中的 `client_id` 參數。

```
POST /oauth2/revoke HTTP/1.1
Host: mydomain.auth.us-east-1.amazoncognito.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded
token=2YotnFZFEjr1zCsicMWpAA&
client_id=1example23456789
```

此撤銷請求會撤銷具有用戶端秘密之應用程式用戶端的重新整理字符。請注意 `Authorization` 標頭，其中包含編碼的用戶端 ID 和用戶端秘密，但請求內文 `client_id` 中沒有。

```
POST /oauth2/revoke HTTP/1.1
Host: 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`。
- 如果用戶端憑證無效，您會收到 HTTP 401 和錯誤 `invalid_client`。
- 如果權杖已被撤銷，或者用戶端提交無效的權杖，您會收到 HTTP 200 OK。

IdP SAML 聲明端點

/saml2/idpresponse 接收 SAML 聲明。在 service-provider-initiated (SP 啟動) 登入中，您的應用程式不會直接與此端點互動，您的 SAML 2.0 身分提供者 (IdP) 會使用其 SAML 回應重新導向您的使用者。對於 SP 起始的登入，請使用的路徑將 IdP 設定為 saml2/idpresponse 聲明消費者服務 (ACS) URL。如需工作階段啟動的詳細資訊，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，當您使用 SAML 2.0 供應商登入使用者後，會在應用程式中叫用對此端點的請求。您的使用者在其瀏覽器中使用 IdP 登入，然後您的應用程式會收集 SAML 聲明並將其提交至此端點。您必須透過 HTTPS 在 HTTP POST 請求內文中提交 SAML 聲明。POST 請求的內文必須是 SAMLResponse 參數和 Relaystate 參數。如需詳細資訊，請參閱 [實作 IdP 起始的 SAML 登入](#)。

saml2/idpresponse 端點最多可接受 100,000 個字元的 SAML 聲明。

POST /saml2/idpresponse

若要在 IdP 起始的登入中使用 /saml2/idpresponse 端點，請使用參數產生 POST 請求，以提供使用者集區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。
- 他們最終想要使用的回呼 URL。
- 他們想要在使用者存取權杖中請求的 OAuth 2.0 範圍。
- 啟動登入請求的 IdP。

IdP 起始的請求內文參數

SAMLResponse

來自與使用者集區中有效應用程式用戶端和 IdP 組態相關聯之 IdP 的 Base64-encoded SAML 聲明。

RelayState

RelayState 參數包含您要傳遞給 oAuth2/authorize 端點的請求參數。如需這些參數的詳細資訊，請參閱 [授權端點](#)。

response_type

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之身分提供者的名稱。

`idp_identifier`

您要重新導向使用者之身分提供者的識別符。

`scope`

您希望使用者從授權伺服器請求的 OAuth 2.0 範圍。

具有正面回應的範例請求

範例 – POST 請求

下列請求適用於應用程式用戶端 MySAMLIdP 中來自 IdP 之使用者的授權碼授予 `example23456789`。使用者 `https://www.example.com` 會使用其授權碼重新導向至 `https://www.example.com`，該授權碼可以交換為包含具有 OAuth 2.0 範圍 `openid`、`email` 和 `phone` 之存取權杖的權杖。

```
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%3Dexample23456789%26redirect_uri%3Dhttps%3A%2F%2Fwww.example.com%26response_type%3Dcode%26scope%3Demail%2Bopenid%2Bphone
```

範例 – 回應

以下是對上一個請求的回應。

```
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 [範例：授權碼授予](#)。Amazon Cognito 在授權碼授予中支援 [Code Exchange \(PKCE\) 的驗證金鑰](#)。

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 從身分驗證取得這三種字符類型，但 API 不會發行具有 `aws.cognito.signin.user.admin` 以外範圍的存取權杖。

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 [範例：無openid範圍的字符（隱含）授予](#)。

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

用戶端憑證是機器對機器存取的僅授權授予。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[範圍、M2M 和具有資源伺服器的 APIs](#)。

用戶端登入資料會授與 AWS 帳單的額外費用。如需詳細資訊，請參閱[Amazon Cognito 定價](#)。

重新整理權杖

您可以直接從 請求重新整理權杖授予[權杖端點](#)。此授予會傳回新的 ID 和存取權杖，以換取有效的重新整理權杖。

如需這些授權及其實作的詳細資訊，請參閱[Amazon Cognito 中的如何使用 OAuth 2.0：了解安全部落格中不同的 OAuth 2.0 授權](#)。AWS

在授權碼授予中使用 PKCE

Amazon Cognito 在授權碼授予中支援 Code Exchange (PKCE) 身分驗證的驗證金鑰。PKCE 是公有用戶端 2.0 OAuth 授權碼授予的延伸。PKCE 可防止兌換攔截的授權碼。

Amazon Cognito 如何使用 PKCE

若要開始使用 PKCE 進行身分驗證，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 用來比較請求初始授權的用戶端與交換權杖授權碼的用戶端的秘密值。

您的應用程式必須將 SHA256 雜湊套用至程式碼驗證器字串，並將結果編碼為 base64。將雜湊字串傳遞至 [授權端點](#) 做為請求內文中的 code_challenge 參數。當您的應用程式交換權杖的授權碼時，它必須以純文字包含程式碼驗證器字串，做為 請求內文中的 code_verifier 參數[權杖端點](#)。Amazon Cognito 會在程式碼驗證器上執行相同的 hash-and-encode 操作。Amazon Cognito 只會在判斷程式碼驗證器產生與授權請求中收到相同的程式碼挑戰時，才會傳回 ID、存取和重新整理權杖。

使用 PKCE 實作授權授予流程

1. 開啟 Amazon Cognito [主控台](#)。如果出現提示，請輸入您的 AWS 登入資料。
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，系統會提示您在精靈期間設定應用程式用戶端和設定受管登入。
 - a. 如果您建立新的使用者集區，請設定應用程式用戶端，並在引導式設定期間設定受管登入。
 - b. 如果您設定現有的使用者集區，如果您尚未新增[網域](#)和[公有應用程式用戶端](#)。

4. 產生隨機英數字串，通常是全域唯一識別符 ([UUID](#))，以為 PKCE 建立程式碼挑戰。此字串是您將在請求中提交至的 `code_verifier` 參數值[權杖端點](#)。
5. 使用 SHA256 演算法雜湊 `code_verifier` 字串。將雜湊操作的結果編碼為 base64。此字串是您將在請求中提交至的 `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}")
```

以下是來自 Python 指令碼的範例輸出：

```
code challenge: Eh0mg-0Zv7BAyo-tdv_vYamx1bo0YDu1DklyXoMDtLg
code verifier: 9D-aW_iygXrgQcWJd0y0tNVMPsXSchIc2xceDhvYVdGLCBk-
JWFTmBNjvKSd0rjTTYaz0FbUmrFERjWx6oKtK2b6z_x4_gHBD1r4K1mRFgyE8yA-05-_v7Dxf3EIYJH
```

6. 使用 PKCE 完成具有授權碼授予請求的受管登入。以下是 URL 範例：

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com&code_challenge=Eh0mg-0Zv7BAyo-
tdv_vYamx1bo0YDu1DklyXoMDtLg&code_challenge_method=S256
```

7. 收集授權碼，並使用權杖端點將其兌換為權杖。以下是範例請求：

```
POST /oauth2/token HTTP/1.1
```

```
Host: mydomain.auth.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-
JWFTmBNjvKSd0rjTTYaz0FbUmrFERrjWx6oKtK2b6z_x4_gHBDlr4K1mRFGyE8yA-05-_v7Dxf3EIYJH
```

8. 檢閱回應。它將包含 ID、存取和重新整理權杖。如需使用 Amazon Cognito 使用者集區字符的詳細資訊，請參閱 [了解使用者集區 JSON Web 字符 JWTs](#)。

受管登入和聯合錯誤回應

受管登入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 會在受管登入中顯示錯誤。

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 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。身分集區會為您的應用程式使用者產生臨時 AWS 登入資料，無論他們是否已登入或您尚未識別他們。使用 AWS Identity and Access Management (IAM) 角色和政策，您可以選擇要授予使用者的權限層級。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 的請求 AWS 服務

將 [API 請求簽署](#) 至 [AWS 服務](#) 例如 Amazon Simple Storage Service (Amazon S3) 和 Amazon DynamoDB。使用 Amazon Pinpoint 和 Amazon CloudWatch 等服務分析使用者活動。

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

交換 ID 或存取權杖、使用者集區權杖、SAML 聲明或 AWS 憑證的社交提供者 OAuth 權杖。

驗證您自己的身分

執行您自己的使用者驗證，並使用開發人員 AWS 登入資料為您的使用者發出登入資料。

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。當您這麼做時，您的使用者可以透過您的使用者集區 IdPs 進行身分驗證、將其宣告合併為常見的 OIDC 身分字符，以及交換該字符做為 AWS 登入資料。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成一個 [AssumeRoleWithWebIdentity](#) API 請求作為短期憑證。

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：[將 Login with Amazon 設定為身分集區 IdP](#)、[將 Facebook 設定為身分集區 IdP](#)、[將 Google 設定為身分集區 IdP](#)、[設定使用 Apple 登入做為身分集區 IdP](#)、Twitter。
- [Amazon Cognito 使用者集區](#)
- [將 OIDC 供應商設定為身分集區 IdP](#)
- [將 SAML 供應商設定為身分集區 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

- [身分集區主控台概觀](#)
- [身分集區身分驗證流程](#)
- [IAM 角色](#)
- [Amazon Cognito 身分集區的安全最佳實務](#)
- [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](#)
- [AWS 服務 使用臨時登入資料存取](#)
- [身分集區第三方身分提供者](#)
- [開發人員驗證的身分](#)
- [將未驗證的使用者切換為已驗證的使用者](#)

身分集區主控台概觀

Amazon Cognito 身分集區為訪客（未經驗證）和已驗證並收到字符的使用者提供臨時 AWS 登入資料。身分集區是連結至外部身分提供者的使用者識別符存放區。

了解身分集區的功能和選項的一種方法是在 Amazon Cognito 主控台中建立一個。您可以探索不同設定對身分驗證流程、角色型和屬性型存取控制和訪客存取的影響。從那裡，您可以繼續進行本指南的後續章節，並將適當的元件新增至您的應用程式，以便實作身分集區身分驗證。

主題

- [建立 身分集區](#)
- [使用者 IAM 角色](#)
- [已進行身分驗證和未進行身分驗證的身分](#)
- [啟用或停用訪客存取](#)
- [變更與身分類型相關聯的角色](#)
- [編輯身分提供者](#)
- [刪除身分集區](#)
- [從身分集區刪除身分](#)
- [使用 Amazon Cognito Sync 與身分集區](#)

建立 身分集區

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. 如果您在 中已有要使用的角色 AWS 帳戶，您可以選擇使用現有的 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 連接身分提供者 中，輸入您在 設定身分池信任 中選擇的身分提供者 (IdP) 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。當基本流程使用中時，您可以略過為 IdP 所做的角色選取，並直接呼叫 [AssumeRoleWithWebIdentity](#)。如需詳細資訊，請參閱[身分集區身分驗證流程](#)。
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[身分集區第三方身分提供者](#)。
- 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 身分集區訪客存取（未驗證的身分）為未向身分提供者進行身分驗證的使用者提供唯一的識別符和 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。如需詳細資訊，請參閱 [Amazon Cognito 身分集區入門](#)。

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. 如果您在 中已有要使用的角色 AWS 帳戶，您可以選擇使用現有的 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。此角色有附加的政策，可指定 AWS 服務 該角色可存取哪些。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。透過修改政策或將不同的角色指派給身分類型，您可以控制 AWS 服務 可以存取的身分類型。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. 如果您在 中已有要使用的角色 AWS 帳戶，您可以選擇使用現有的 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[身分集區第三方身分提供者](#)。

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 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 [實作 Amazon Cognito Sync 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. 如果您在 中已有要使用的角色 AWS 帳戶，您可以選擇使用現有的 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 [實作 Amazon Cognito Sync 串流](#)。

設定 Amazon Cognito 事件

Amazon Cognito Events 可讓您執行 AWS Lambda 函數，以回應 Amazon Cognito Sync 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 [使用 Amazon Cognito Events 自訂工作流程](#)。

若要進一步了解 AWS Lambda，請參閱 [AWS Lambda](#)。

身分集區身分驗證流程

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 也會為您的應用程式提供暫時、有限權限的登入資料，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。

完成其中一個流程後，您就可以存取角色的存取政策 AWS 服務 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成品：

供應商	身分驗證成品
Amazon Cognito 使用者集區	ID 權杖
OpenID Connect (OIDC)	ID 權杖

供應商	身分驗證成品
SAML 2.0	SAML 聲明
社交供應商	存取權杖

強化 (簡化) 身分驗證流程

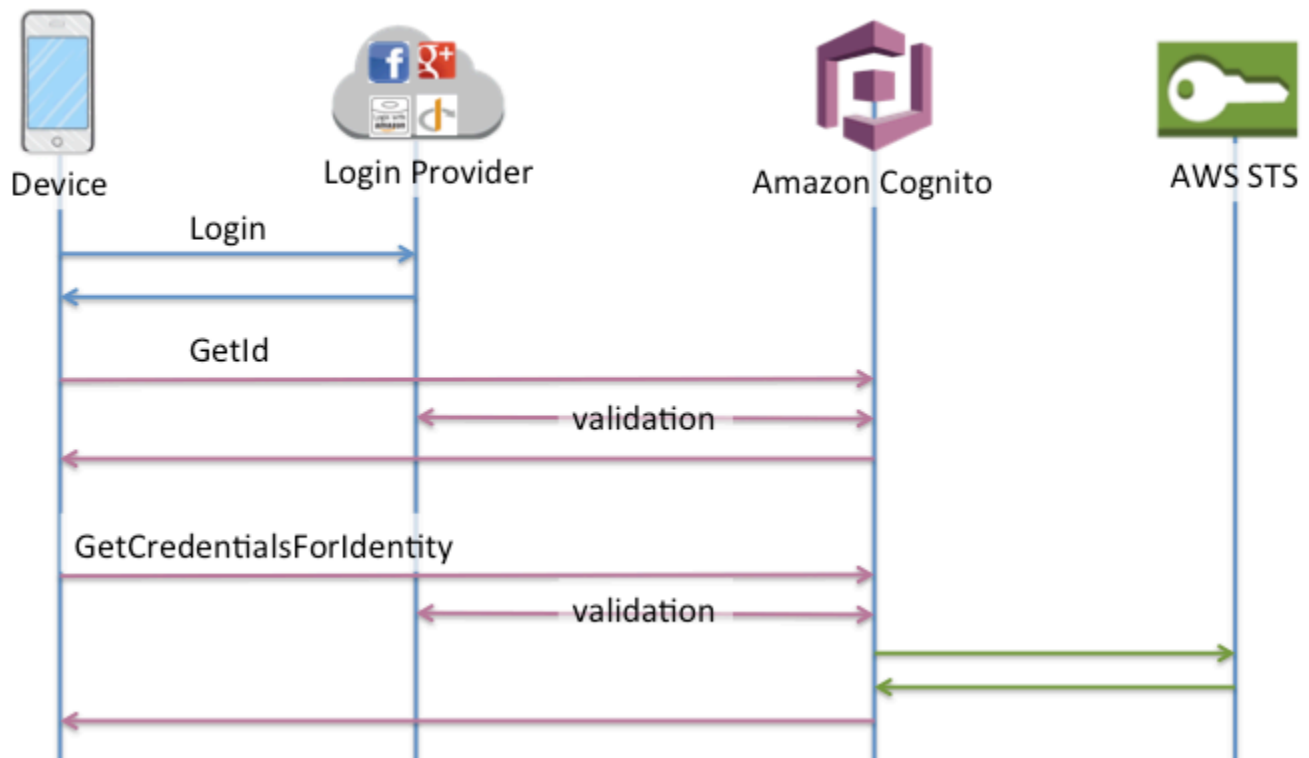
當您使用增強型身分驗證流程時，您的應用程式會先在 [GetId](#) 請求中提供來自授權 Amazon Cognito 使用者集區或第三方身分提供者的身分驗證證明。

1. 您的應用程式會在 [GetID](#) 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分驗證證明 – JSON Web 權杖或 SAML 聲明。
2. 您的身分集區會傳回身分 ID。
3. 您的應用程式在 [GetCredentialsForIdentity](#) 請求中結合身分 ID 與相同的身分驗證證明。
4. 您的身分集區會傳回 AWS 登入資料。
5. 您的應用程式使用暫時登入資料簽署 AWS API 請求。

增強型身分驗證會管理身分集區組態中 IAM 角色選取和憑證擷取的邏輯。您可以設定身分集區以選取預設角色，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。增強型身分驗證的 AWS 登入資料有效期為一小時。

增強型身分驗證中的操作順序

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. 您的應用程式會在 [GetID](#) 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分驗證證明 – JSON Web 權杖或 SAML 聲明。
2. 您的身分集區會傳回身分 ID。
3. 您的應用程式在 [GetOpenIdToken](#) 請求中結合身分 ID 與相同的身分驗證證明。
4. GetOpenIdToken 會傳回身分集區發行的新 OAuth 2.0 字符。
5. 您的應用程式會在 [AssumeRoleWithWebIdentity](#) 請求中顯示新的權杖。
6. AWS Security Token Service (AWS STS) 會傳回 AWS 登入資料。
7. 您的應用程式使用暫時登入資料簽署 AWS API 請求。

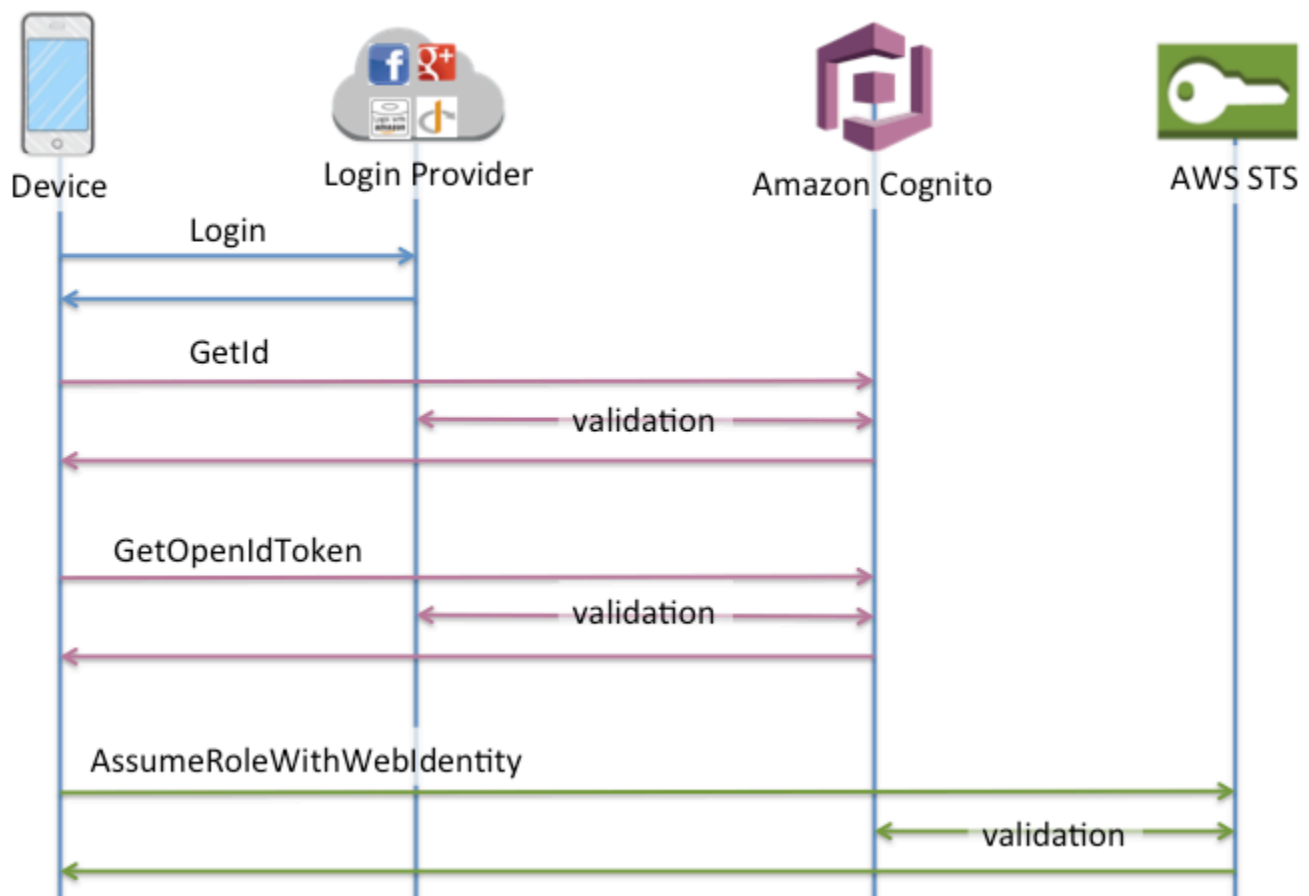
基本工作流程可讓您更精密控制自己分配給使用者的憑證。強化身分驗證流程的 [GetCredentialsForIdentity](#) 請求會根據存取權杖的內容請求角色。傳統工作流程中的 [AssumeRoleWithWebIdentity](#) 請求會授予您的應用程式更大的能力，以針對您已設定足夠信任政策的任何 AWS Identity and Access Management 角色請求登入資料。您也可以請求自訂角色工作階段持續時間。

您可以在沒有角色映射的使用者集區中使用基本身分驗證流程登入。這種類型的身分集區沒有預設驗證或未驗證的角色，也沒有設定角色型或屬性型存取控制。當您GetOpenIdToken嘗試使用角色映射的身分集區時，您會收到下列錯誤。

Basic (classic) flow is not supported with RoleMappings, please use enhanced flow.

基本身分驗證中的操作順序

1. GetId
2. GetOpenIdToken
3. AssumeRoleWithWebIdentity



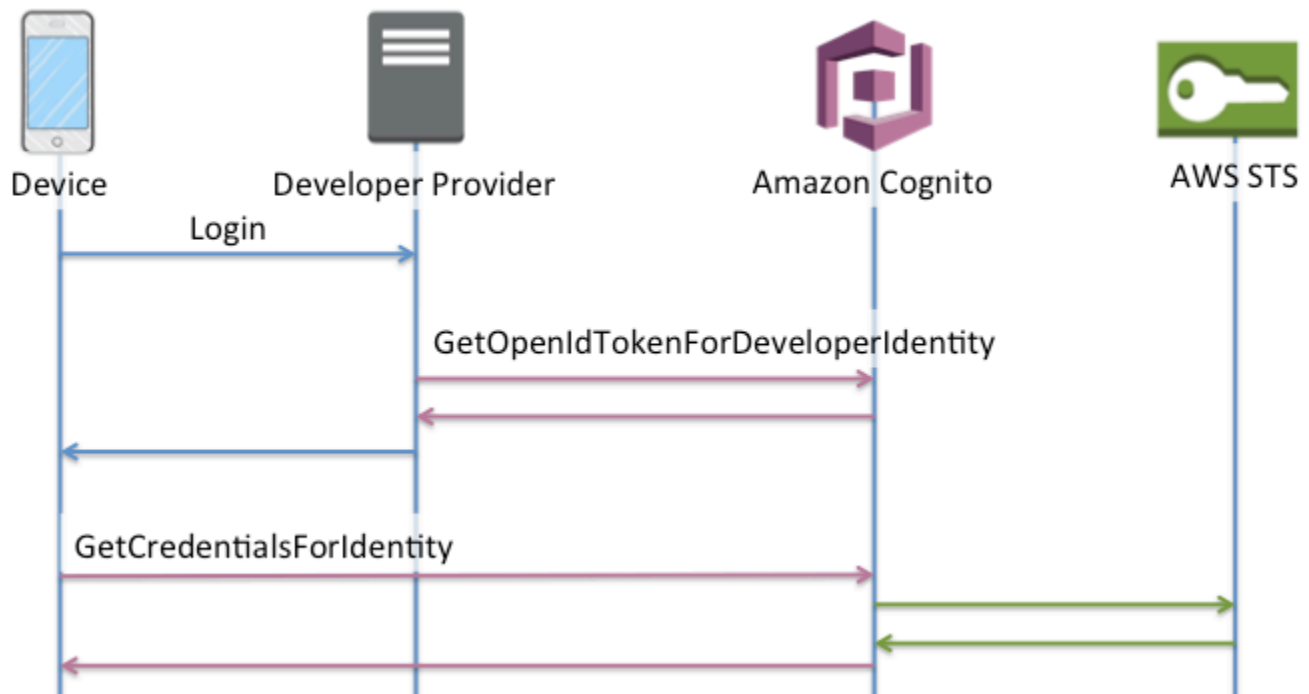
開發人員驗證的身分驗證流程

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

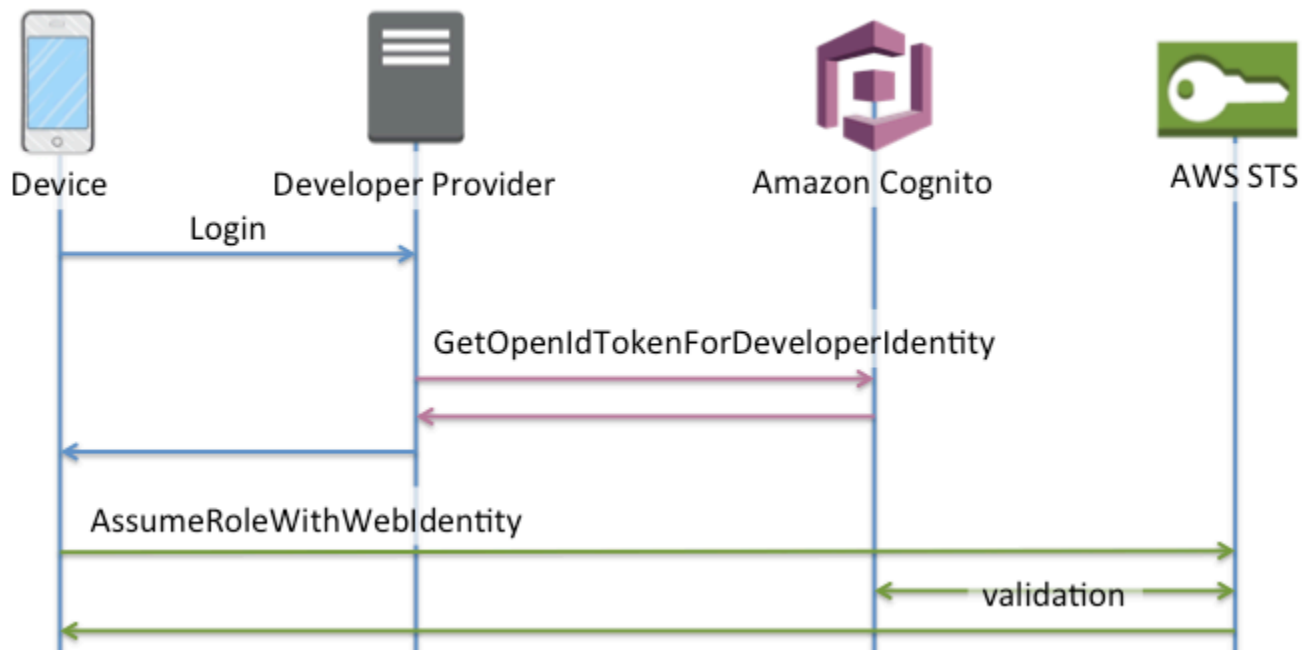
使用開發人員供應商進行增強型身分驗證的操作順序

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](#)



與開發人員供應商進行基本身分驗證的操作順序

1. 在身分集區外部實作邏輯，以登入並產生開發人員提供者識別符。
2. 擷取儲存的伺服器端 AWS 登入資料。
3. 在以授權 AWS 憑證簽署的 [GetOpenIdTokenForDeveloperIdentity](#) API 請求中傳送開發人員提供者識別符。
4. 使用 [AssumeRoleWithWebIdentity](#) 請求應用程式登入資料。



我應該使用哪一種身分驗證流程？

增強型流程是開發人員工作量最低的最安全選擇：

- 增強的流程可減少 API 請求的複雜性、大小和速率。
- 您的應用程式不需要向 提出其他 API 請求 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

當您建立新的身分集區時，預設不要啟用基本（傳統）身分驗證，這是最佳實務。若要實作基本身分驗證，請先評估 Web 身分 IAM 角色的信任關係。然後在您的用戶端中建置角色選取的邏輯，並保護用戶端不受使用者修改。

基本身分驗證流程會將 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的工作階段，並發出權杖供您交換登入資料 AWS STS。使用者可以將基本身分驗證中的字符交換為信任您的身分集區和的任何 IAM 角色 `amr`，或已驗證/未驗證狀態。

同樣地，請了解開發人員身分驗證是身分提供者身分驗證驗證的捷徑。Amazon Cognito AWS 信任授權 [GetOpenIdTokenForDeveloperIdentity](#) 請求的登入資料，無需對請求內容進行其他驗證。保護授權開發人員身分驗證不被使用者存取的秘密。

API 摘要

GetId

[GetId](#) API 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。瀏覽器中的 AWS Mobile SDKs 和 AWS 適用於 JavaScript 的 SDK 具有可為您處理此快取的登入資料提供者。

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 ID 後，即可呼叫 [GetCredentialsForIdentity](#) API。此操作在功能上等同於呼叫 [GetOpenIdToken](#)，然後呼叫 [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 主控台來執行，或是透過 [SetIdentityPoolRoles](#) 操作手動執行。

GetOpenIdToken

在您建立身分 ID 之後，提出 [GetOpenIdToken](#) API 請求。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。您可以提交此字符做為 [AssumeRoleWithWebIdentity](#) 請求中的 `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。行動 SDK 中的 AWS 供應商或瀏覽器中適用於 JavaScript 的 AWS SDK 會為您執行此操作。SDKs

`GetOpenIdTokenForDeveloperIdentity`

使用開發人員驗證的身分時，[GetOpenIdTokenForDeveloperIdentity](#) 操作會從裝置取代 [GetId](#) 和 [GetOpenIdToken](#) 的使用。由於您的應用程式使用 AWS 登入資料簽署此 API 操作的請求，Amazon Cognito 信任請求中提供的使用者識別符是有效的。開發人員身分驗證會取代 Amazon Cognito 對外部供應商執行的字符驗證。

此 API 的承載包含 logins 映射。此映射必須包含開發人員供應商的金鑰，以及做為系統中使用者識別符的值。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

利用開發人員驗證的身分，Amazon Cognito 可以透過 [MergeDeveloperIdentities](#) API 來支援隱含合併及明確合併。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 權杖之後，您可以透過對 AWS Security Token Service () 的 [AssumeRoleWithWebIdentity](#) API 請求，將此權杖交換為臨時 AWS 憑證 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許可。修改它們以符合您的需求。

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 (US-West)	<code>https://cognito-identity.us-gov-west-1.amazonaws.com/.well-known/jwks_uri</code>
China (Beijing)	<code>https://cognito-identity.cn-north-1.amazonaws.com.cn/.well-known/jwks_uri</code>
Opt-in Regions like Europe (Milan) and Africa (Cape Town)	<code>https://cognito-identity. <i>Region</i>.amazonaws.com/.well-known/jwks_uri</code>

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 角色運作的方式如下：當使用者登入您的應用程式時，Amazon Cognito 會為使用者產生暫時 AWS 憑證。這些暫時登入資料會與特定 IAM 角色建立關聯。您可以使用 IAM 角色定義一組存取 AWS 資源的許可。

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](#)
- [角色信任和許可](#)

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](#)。

呈現給 的權杖是由身分集區 AWS STS 產生，該集區會將使用者集區、社交或 OIDC 提供者權杖或 SAML 聲明轉譯為自己的權杖。身分集區權杖包含身分集區 ID 的 aud 宣告。

下列範例角色信任政策允許聯合身分服務主體 `cognito-identity.amazonaws.com` 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）身分驗證中 IAM 角色的信任政策

您必須套用至少一個條件，以限制與身分集區搭配使用之角色的信任政策。當您為身分集區建立或更新角色信任政策時，如果您嘗試儲存變更而沒有至少一個限制來源身分的條件索引鍵，IAM 會傳回錯誤。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集區來源身分的條件。如需完整清單，請參閱 [AWS Web 聯合身分的可用金鑰](#)。

在基本或傳統身分集區的身分驗證中，AWS STS 如果具有正確的信任政策，您可以使用擔任任何 IAM 角色。Amazon Cognito 身分集區的 IAM 角色信任服務主體 `cognito-identity.amazonaws.com` 擔任該角色。此組態不足以保護您的 IAM 角色免於意外存取資源。此類型的角色必須將其他條件套用至角色信任政策。如果沒有以下至少一個條件，則無法為身分集區建立或修改角色。

`cognito-identity.amazonaws.com:aud`

將角色限制為來自一或多個身分集區的操作。Amazon Cognito 會在身分集區字符中指出 `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 會在身分集區字符中指出 `amr` 宣告中的身分驗證狀態。

cognito-identity.amazonaws.com:sub

依 [UUID](#) 將角色限制為一或多個使用者。此 UUID 是身分集區中的使用者身分 ID。此值不是來自使用者原始身分提供者sub的值。Amazon Cognito 會在身分集區字符的sub宣告中指出此 UUID。

增強型流程身分驗證需要 IAM 角色與身分集區位於相同的 AWS 帳戶 中，但在基本身分驗證中並非如此。

其他考量事項適用於擔任[跨帳戶 IAM 角色](#)的 Amazon Cognito 身分集區。這些角色的信任政策必須接受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條件值中身分集區的使用者。

身分集區針對身分發行的字符，包含身分集區 AWS 帳戶 來源的相關資訊。當您在 [AssumeRoleWithWebIdentity](#) API 請求中呈現身分集區字符時，會 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 判斷請求是跨帳戶，則會檢查角色信任政策是否有 aud條件。如果角色信任政策中沒有這類條件，則擔任角色呼叫會失敗。如果請求不是跨帳戶，AWS STS 不會強制執行此限制。最佳實務是，一律將此類型的條件套用至身分集區角色的信任政策。

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 Amazon Cognito Web 聯合的 IAM 政策變數詳細資訊，請參閱 AWS Identity and Access Management 《使用者指南》中的 [IAM 和 AWS STS 條件內容索引鍵](#)。

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 受管工作階段政策](#)。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 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. PolicyArns.member.N 參數，其值為 AWS 受管政策，可在 Amazon CloudWatch 中授予其他許可。

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 Data 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 Amazon Location Service
遊戲開發	Amazon GameLift 伺服器
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 AI Amazon Textract ¹ Amazon Transcribe Amazon Translate
管理與治理	Amazon CloudWatch Amazon CloudWatch Logs
聯網與內容交付	Amazon API Gateway
安全、身分與合規	Amazon Cognito 使用者集區

類別	服務
儲存	Amazon Simple Storage Service

1 對於下表 AWS 服務 中的 ，內嵌政策會授予動作子集。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 ReEncryptTo ReEncryptFrom GenerateDataKey GenerateDataKeyPair GenerateDataKeyPair GenerateDataKeyPairWithoutPlaintext GenerateDataKeyWithoutPlaintext
Amazon SageMaker AI	InvokeEndpoint
Amazon Textract	DetectDocumentText AnalyzeDocument
Amazon Sumerian	View*
Amazon Location Service	SearchPlaceIndex* GetPlace CalculateRoute* *Geofence

AWS 服務	未經身分驗證的強化流程使用者的最大許可
	*Geofences
	DevicePosition

若要授予 AWS 服務 超過此清單的存取權，請在您的身分集區中啟用基本（傳統）身分驗證流程。如果您的使用者看到未驗證使用者獲指派 IAM 角色的政策 AWS 服務 所允許的NotAuthorizedException錯誤，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。

訪客使用者的內嵌工作階段政策

Amazon Cognito 會先在 IAM 憑證的請求中套用內嵌政策。內嵌工作階段政策會限制使用者的有效許可，無法包含下列清單以外的任何 AWS 服務 的存取。您也必須在套用至使用者 IAM 角色的政策 AWS 服務 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。如需內嵌政策和其他工作階段政策的淨效果概觀，請參閱 [未經身分驗證的使用者可存取的服務](#)。

```
{
  "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:*",
        "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:*"

```

訪客的 AWS 受管工作階段政策

Amazon Cognito 也會將 AWS 受管政策作為工作階段政策套用到未驗證訪客的增強流程工作階段。此政策會使用政策 來限制未驗證使用者許可的範圍AmazonCognitoUnAuthedIdentitiesSessionPolicy。

您也必須在連接到未驗證 IAM 角色的政策中授予此許可。使用者對擔任角色工作階段的有效許可是指派給其角色的 IAM 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](#)。

如需此 AWS 受管政策和其他工作階段政策的淨效果概觀，請參閱 [未經身分驗證的使用者可存取的服務](#)。

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",
      "geo:GetMap*",
      "geo:SearchPlaceIndex*",
      "geo:GetPlace",
      "geo:CalculateRoute*",
      "geo:*Geofence",
      "geo:*Geofences",
      "geo:*DevicePosition*",
      "kms:Encrypt",
      "kms:Decrypt",
      "kms:ReEncryptTo",
      "kms:ReEncryptFrom",
      "kms:GenerateDataKey",
      "kms:GenerateDataKeyPair",
```

```

        "kms:GenerateDataKeyPairWithoutPlaintext",
        "kms:GenerateDataKeyWithoutPlaintext"
    ],
    "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:::amzn-s3-demo-bucket/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:::amzn-s3-demo-bucket/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:::amzn-s3-demo-bucket` 中其他物件的存取權。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket"],
      "Condition": {"StringLike": {"s3:prefix": ["${cognito-identity.amazonaws.com:sub}/*"]}}
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/${cognito-identity.amazonaws.com:sub}/*"]
    }
  ]
}
```

[Amazon S3 Access Grants](#) 可實現類似的存取模型。

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}"]
        }
    }
}
]
}

```

授予身分許可可以叫用 Lambda 函數

下列存取政策授予身分許可可以叫用 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 身分集區的安全最佳實務

Amazon Cognito 身分集區為您的應用程式提供暫時 AWS 登入資料。AWS 帳戶 通常同時包含應用程式使用者所需的資源，以及私有後端資源。組成 AWS 登入資料的 IAM 角色和政策可以授予任何這些資源的存取權。

身分集區組態的主要最佳實務是確保您的應用程式可以在沒有過多或非預期權限的情況下完成工作。為了防止安全設定錯誤，請在您要發佈至生產環境的每個應用程式啟動之前檢閱這些建議。

主題

- [IAM 組態最佳實務](#)
- [身分集區組態最佳實務](#)

IAM 組態最佳實務

當訪客或已驗證的使用者在您的應用程式中啟動需要身分集區憑證的工作階段時，您的應用程式會擷取 IAM 角色的臨時 AWS 憑證。登入資料可能適用於預設角色、身分集區組態中規則選擇的角色，或是應用程式選擇的自訂角色。透過指派給每個角色的許可，您的使用者即可存取您的 AWS 資源。

如需一般 IAM 最佳實務的詳細資訊，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已驗證的使用者。AWS STS 也要求跨帳戶基本身分驗證請求有兩個特定條件：`cognito-identity.amazonaws.com:aud`和`cognito-identity.amazonaws.com:amr`。最佳實務是在信任身分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` 宣告必須符合信任的身分集區 ID。
- `cognito-identity.amazonaws.com:amr`：身分集區字符中的 `amr` 宣告必須經過身分驗證或未經驗證。使用此條件時，您只能將角色的存取權保留給未驗證的訪客，也可以保留給已驗證的使用者。您可以進一步精簡此條件的值，將角色限制為來自特定提供者的使用者，例如 `graph.facebook.com`。

下列範例角色信任政策會在下列條件下授予角色的存取權：

```
{
  "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"：使用者必須經過身分驗證。訪客使用者無法擔任該角色。

套用最低權限許可

當您使用已驗證存取或訪客存取的 IAM 政策設定許可時，只會授予執行特定任務所需的特定許可，或最低權限許可。下列範例 IAM 政策在套用至角色時，會授予 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"]
    }
  ]
}

```

身分集區組態最佳實務

身分集區有彈性的選項可產生 AWS 登入資料。當您的應用程式可以使用最安全的方法時，請勿採取設計捷徑。

了解訪客存取的影響

未驗證的訪客存取允許使用者在登入 AWS 帳戶 之前從 擷取資料。知道您的身分集區 ID 的任何人都可以請求未經驗證的登入資料。您的身分集區 ID 不是機密資訊。當您啟用訪客存取時，您授予未經驗證工作階段的 AWS 許可可供所有人使用。

最佳實務是在使用者驗證後，才停用訪客存取並擷取必要的資源。如果您的應用程式在登入之前需要存取 資源，請採取下列預防措施。

- 熟悉[未驗證角色的自動限制](#)。
- 監控和調整未驗證 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。
- 只有當您確信會將 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。

預設使用增強型身分驗證

透過基本（傳統）身分驗證，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[縮小範圍政策](#)，為未驗證的身分提供額外的安全性。增強型流程是開發人員工作量最低的最安全選擇。若要進一步了解這些選項，請參閱 [身分集區身分驗證流程](#)。

基本流程可以公開用戶端邏輯，這些邏輯會進入 AWS 對登入資料 STS API 請求的角色選擇和組合。增強型流程會隱藏身分集區自動化背後的邏輯和擔任角色請求。

當您設定基本身分驗證時，請將 [IAM 最佳實務](#) 套用至您的 IAM 角色及其許可。

安全地使用開發人員供應商

開發人員驗證的身分是伺服器端應用程式的身分集區功能。身分集區進行開發人員身分驗證所需的唯一身分驗證證據是身分集區開發人員的 AWS 憑證。身分集區不會對您在此身分驗證流程中呈現的開發人員提供者識別符的有效性強制執行任何限制。

最佳實務是，僅在下列條件下實作開發人員提供者：

- 若要建立使用開發人員驗證憑證的責任，請設計您的開發人員供應商名稱和識別符，以指示身分驗證來源。例如："Logins" : {"MyCorp provider" : "[*provider application ID*]"}。
- 避免長期使用者登入資料。將伺服器端用戶端設定為使用服務連結角色請求身分，例如 [EC2 執行個體描述檔](#) 和 [Lambda 執行角色](#)。
- 避免在相同的身分集區中混合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 身分集區實施。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

- [透過 Amazon Cognito 身分集區使用屬性進行存取控制](#)
- [使用屬性進行存取控制的政策範例](#)
- [關閉以存取控制屬性 \(主控台\)](#)
- [預設供應商對應](#)

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：

- [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策略範例](#)。

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"

供應商	權杖類型	委託人標籤值	範例
Google	ID 權杖	aud(client ID) 和 sub(user ID)	"620493171733-eebk7c0hcp5lj3e1tlqp1gntt3k0rncv.apps.googleusercontent.com"、"109220063452404746097"
SAML	聲明	"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"、"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name"	"auth0 5e28d196f8f55a0eaaa95de3"、"user123@gmail.com"
Apple	ID 權杖	aud(client ID) 和 sub(user ID)	"com.amazonaws.ec2-54-80-172-243.compute-1.client"、"001968.a6ca34e9c1e742458a26cf8005854be9.0733"
Amazon	存取權杖	aud (Client ID on Amzn Dev Ac)、user_id(user ID)	"amzn1.application-oa2-client.9d70d9382d3446108aaee3dd763a0fa6"、"amzn1.account.AGHNIFJQMFSBG3G6XCPVB35ORQAA"

供應商	權杖類型	委託人標籤值	範例
標準 OIDC 供應商	ID 權杖和存取權杖	aud (as client_id)、sub (as user ID)	"620493171733-eebk7c0hcp5lj3e1tlqp1gntt3k0rncv.apps.googleusercontent.com"、"109220063452404746097"
Twitter	存取權杖	aud (app ID; app Secret)、sub (user ID)	"DfwifTtKEX1FiIBRnOTIR0CFK;Xgj5xb8xlrIVCPjXgLldkW7fXmw cJJrFvnoK9gwZkLexo1y5z1"、"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 身分集區會為您的已驗證使用者指派一組臨時、有限權限的登入資料，以存取您的 AWS 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：

- 如果已設定的 [GetCredentialsForIdentity](#) CustomRoleArn 參數符合 cognito:roles 宣告中的角色，則使用此參數。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、已設定 cognito:roles 宣告，且未在 GetCredentialsForIdentity 的呼叫中指定 CustomRoleArn，則會使用主控台內的 Role resolution (角色解析) 設定或是 AmbiguousRoleResolution 欄位 (在 [SetIdentityPoolRoles](#) API 的 RoleMappings 參數中) 來判斷所要指派的角色。

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[使用使用者屬性](#)。

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 AmbiguousRoleResolutionRoleMapping [類型的](#) 欄位中 (指定在 RoleMappingsSetIdentityPoolRoles [API 的](#) 參數中) 沒有規則相符時，所要指派的角色。

若要在 Amazon Cognito 主控台中將規則型映射新增至身分提供者，請新增或更新 IdP，然後選取在角色選擇下選擇具有規則的角色。您可以從該處新增規則，將提供者宣告映射至 IAM 角色。

您可以使用 [RoleMapping](#) 類型的 RulesConfiguration 欄位，在 AWS CLI 或 API 中為身分提供者設定規則型映射。您可以在 [SetIdentityPoolRoles](#) API 的 RoleMappings 參數中指定此欄位。

例如，下列 AWS CLI 命令新增規則，將角色指派給您 Sacramento 位置中經過 OIDC IdP 身分驗證 `arn:aws:iam::123456789012:role/Sacramento_team_S3_admin` 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)。

您可以使用主控台內的 Role resolution (角色解析) 設定，以及 RoleMappingsSetIdentityPoolRoles [API 的](#) 參數，來指定當無法從權杖判定正確角色時的預設行為。

取得憑證

您可以使用 Amazon Cognito 將臨時、有限權限的登入資料交付到您的應用程式，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。在 AWS Security Token Service () 中使用 Web 聯合身分支援交換登入資料AWS STS。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，登入資料物件中的身分會使用 交換登入資料 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 SDK 或 AWS 的本機config檔案中設定您的身分集區字符 AWS CLI，請新增web_identity_token_file設定檔項目。請參閱《AWS SDKs和工具參考指南》中的[擔任角色登入資料提供者](#)。

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。為了獲得最佳結果，請使用內建的身分集區整合啟動您的專案 AWS Amplify。

AWS 使用身分集區取得和設定登入資料的開發套件資源

- 《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)
- 《適用於 JavaScript 的 AWS SDK 開發人員指南》中的[使用 Amazon Cognito Identity 驗證使用者](#)
- 《適用於 .NET 的 AWS SDK 開發人員指南》中的[Amazon Cognito 登入資料提供者](#)
- 《適用於 Go 的 AWS SDK 開發人員指南》中的[以程式設計方式指定登入資料](#)
- 《AWS SDK for Java 2.x 開發人員指南》中的在[程式碼中提供臨時憑證](#)
- 《適用於 PHP 的 AWS SDK 開發人員指南》中的[assumeRoleWithWebIdentityCredentialProvider](#) 提供者
- 在 適用於 Python (Boto3) 的 AWS SDK 文件中[使用 Web 身分提供者指派角色](#)
- 《適用於 Rust 的 AWS SDK 開發人員指南》中的[指定您的登入資料和預設區域](#)

下列各節提供部分 AWS SDKs中的範例程式碼。

Android

您可以使用 Amazon Cognito 將臨時、有限權限的登入資料交付到您的應用程式，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。若要提供 AWS 登入資料給應用程式，請遵循下列步驟。

若要在 Android 應用程式中使用 Amazon Cognito 身分集區，請設定 AWS Amplify。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 層級 11) 開始，如果您在主要應用程式執行緒上執行網路 I/O，您的應用程式將會自動失敗，並擲出 [NetworkOnMainThreadException](#)。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 將臨時、有限權限的登入資料交付到您的應用程式，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。若要提供 AWS 登入資料給應用程式，請完成下列步驟。

若要在 iOS 應用程式中使用 Amazon Cognito 身分集區，請設定 AWS Amplify。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 將臨時、有限權限的登入資料交付到您的應用程式，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。若要提供 AWS 登入資料給應用程式，請遵循下列步驟。

若要在 iOS 應用程式中使用 Amazon Cognito 身分集區，請設定 AWS Amplify。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.loginFacebook` 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 將臨時、有限權限的登入資料交付到您的應用程式，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。若要提供 AWS 登入資料給應用程式，請遵循下列步驟。

[AWS SDK for Unity](#) 現在是 [適用於 .NET 的 SDK](#) 的一部分。若要在 中開始使用 Amazon Cognito 適用於 .NET 的 SDK，請參閱《適用於 .NET 的 AWS SDK 開發人員指南》中的 [Amazon Cognito 登入資料提供者](#)。或者，請參閱 [Amplify 開發中心](#)，了解使用 建置應用程式的選項 AWS Amplify。

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 將臨時、有限權限的登入資料交付到您的應用程式，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。若要提供 AWS 登入資料給應用程式，請遵循下列步驟。

[AWS SDK for Xamarin](#) 現在是 [適用於 .NET 的 SDK](#) 的一部分。若要在 中開始使用 Amazon Cognito 適用於 .NET 的 SDK，請參閱《適用於 .NET 的 AWS SDK 開發人員指南》中的 [Amazon Cognito 登入資料提供者](#)。或者，請參閱 [Amplify 開發中心](#)，了解使用 建置應用程式的選項 AWS Amplify。

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();
```

AWS 服務 使用臨時登入資料存取

使用身分集區成功進行身分驗證的結果是一組 AWS 登入資料。透過這些登入資料，您的應用程式可以向受 IAM 身分驗證保護 AWS 的資源提出請求。使用您可以新增至應用程式以存取身分集區 API 操作的各種 AWS SDKs，您可以提出未驗證的 API 請求來產生臨時登入資料。然後，您可以將其他 SDKs AWS 服務 新增至用戶端，並使用這些臨時登入資料簽署請求。授予臨時憑證角色的 IAM 許可必須允許您從其他 服務請求的操作。

在您設定 Amazon Cognito 登入資料提供者並擷取 AWS 登入資料後，請建立 AWS 服務 用戶端。以下是 AWS SDK 文件的一些範例。

AWS 用於建立用戶端的 SDK 資源

- 《適用於 C++ 的 AWS SDK 開發人員指南》中的[AWS 用戶端組態](#)
- 《適用於 Go 的 AWS SDK 開發人員指南》中的[搭配 使用 適用於 Go 的 AWS SDK V2 AWS 服務](#)
- 《AWS SDK for Java 2.x 開發人員指南》中的[設定 HTTP 用戶端](#)
- 《適用於 JavaScript 的 AWS SDK 開發人員指南》中的[建立和呼叫服務物件](#)
- 在 適用於 Python (Boto3) 的 AWS SDK 文件中[建立用戶端](#)
- 《適用於 Rust 的 AWS SDK 開發人員指南》中的[建立服務用戶端](#)
- 《適用於 Swift 的 AWS SDK 開發人員指南》中的[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 Amazon Cognito 身分集區，請設定 AWS Amplify。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符，以及 AWS Mobile SDK 的臨時有限權限 AWS 登入資料。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 身分集區，請設定 AWS Amplify。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符，以及 AWS Mobile SDK 的臨時有限權限 AWS 登入資料。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 身分集區，請設定 AWS Amplify。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符，以及 AWS Mobile SDK 的臨時有限權限 AWS 登入資料。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符，以及 AWS Mobile SDK 的臨時、有限權限 AWS 登入資料。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。

Unity

[AWS SDK for Unity](#) 現在是 [適用於 .NET 的 SDK](#) 的一部分。若要在 中開始使用 Amazon Cognito 適用於 .NET 的 SDK，請參閱《適用於 .NET 的 AWS SDK 開發人員指南》中的 [Amazon Cognito 登入資料提供者](#)。或者，請參閱 [Amplify 開發中心](#)，了解使用 建置應用程式的選項 AWS Amplify。

```
// 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符，以及 AWS Mobile SDK 的臨時、有限權限 AWS 登入資料。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。

Xamarin

[AWS SDK for Xamarin](#) 現在是 [適用於 .NET 的 SDK](#) 的一部分。若要在 中開始使用 Amazon Cognito 適用於 .NET 的 SDK，請參閱《適用於 .NET 的 AWS SDK 開發人員指南》中的 [Amazon Cognito 登入資料提供者](#)。或者，請參閱 [Amplify 開發中心](#)，了解使用 建置應用程式的選項 AWS Amplify。

```
// 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符，以及 AWS Mobile SDK 的臨時、有限權限 AWS 登入資料。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。

身分集區第三方身分提供者

透過 Amazon Cognito 身分集區，您可以與各種外部身分提供者 (IdPs) 整合，以在應用程式中透過聯合身分驗證提供臨時 AWS 憑證。透過設定您的身分集區以使用這些外部 IdPs，您可以透過 Amazon Cognito 使用者集區、社交提供者、OIDC 提供者或 SAML 提供者進行身分驗證，為您的使用者授權存取後端 AWS 資源。本節涵蓋設定和整合 IdPs 與您的 Amazon Cognito 身分集區的步驟。

使用 `logins` 屬性，您便可設定從身分提供者 (IdP) 收到的憑證。您也可以將身分集區與多個 IdPs 建立關聯。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。

下列指示將引導您使用 Amazon Cognito 身分池支援的 IdP 進行身分驗證。

主題

- [將 Facebook 設定為身分集區 IdP](#)
- [將 Login with Amazon 設定為身分集區 IdP](#)
- [將 Google 設定為身分集區 IdP](#)
- [設定使用 Apple 登入做為身分集區 IdP](#)
- [將 OIDC 供應商設定為身分集區 IdP](#)
- [將 SAML 供應商設定為身分集區 IdP](#)

將 Facebook 設定為身分集區 IdP

Amazon Cognito 身分集區可與 Facebook 搭配使用，為您的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：

Note

Amazon Cognito 身分集區聯合與 [Facebook Limited Login](#) 不相容。如需如何設定適用於 iOS 的 Facebook Login 而不超過有限登入許可集的詳細資訊，請參閱適用於 [iOS 的 Facebook Login - 開發人員的 Meta 快速入門](#)。

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

如需如何將 JavaScript Web 應用程式與 Facebook 登入整合的詳細資訊，請參閱 [Facebook 入門指南](#)。

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的存取權杖來驗證使用者、產生唯一識別符，並視需要授予使用者對其他 AWS 資源的存取權。

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 工作階段物件包含 OAuth 權杖，Amazon Cognito 會使用此權杖來為已驗證的最終使用者產生 AWS 登入資料。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

設定登入映射後，請呼叫 `refresh` 或 `get` 以擷取 AWS 登入資料。

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](#) 參考資料頁面，並選擇 `initWithRegionType: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)。

 Note

Amazon Cognito 身分集區聯合與 [Facebook Limited Login](#) 不相容。如需如何設定適用於 iOS 的 Facebook Login 而不超過有限登入許可集的詳細資訊，請參閱適用於 [iOS 的 Facebook Login - 開發人員的 Meta 快速入門](#)。

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 AWSIdentityProviderManager 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 [AWSIdentityProviderManager](#) 參考資料頁面，並選擇 initWithRegionType: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的存取權杖來驗證使用者、產生唯一識別符，並視需要授予使用者對其他 AWS 資源的存取權。

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 開發套件會取得 Amazon Cognito 用來為已驗證的最終使用者產生 AWS 憑證的 OAuth 字符合。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：將使用者集區與 Cognito 身分整合」。

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 <LoginResult> () {
        HandleSuccess = loginResult = > {
            var accessToken = loginResult.AccessToken;
            credentials.AddLogin("graph.facebook.com", accessToken.Token);
        }
    });
}
```



```
        //open new activity
    },
    HandleCancel = () = > {
        //throw error message
    },
    HandleError = loginError = > {
        //throw error message
    }
});
LoginManager.Instance.LogInWithReadPermissions(this, new List < string > {
    "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
        }
    });
}
```

將 Login with Amazon 設定為身分集區 IdP

Amazon Cognito 身分集區可與 Login with Amazon 搭配使用，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

對 Amazon 登入進行身分驗證之後，您可以在 TokenListener 介面的 onSuccess 方法中，將權杖傳遞至 Amazon Cognito 憑證供應商。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

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：

```
- (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]
    }
}
```

使用 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'
    }
})
```

```
});
```

將 Google 設定為身分集區 IdP

Amazon Cognito 身分集區與 Google 搭配使用，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，以便達到最佳的整合。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](#)。

如需如何將 Google 整合到您的 Android 應用程式的詳細資訊，請參閱 Google Identity 文件中的[使用 Sign in with Google 驗證使用者](#)。

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，以便達到最佳的整合。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，以便達到最佳的整合。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 GTM0Auth2Authentication 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: GTM0Auth2Authentication!, 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，以便達到最佳的整合。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 JavaScript Web 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. 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.
    });
  }
}
```

```
}  
}
```

設定使用 Apple 登入做為身分集區 IdP

Amazon Cognito 身分集區可與 Sign in with Apple 搭配使用，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」的最新資訊，請參閱《Apple 開發人員文件》中的[設定您的環境以使用 Apple 登入](#)。

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 with 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 Android 使用者介面，請遵循 Apple 文件中的在 [Web 上顯示 Sign in with Apple 按鈕](#)。
- 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的 ID 字符來驗證使用者、產生唯一識別符，並視需要授予使用者存取其他 AWS 資源的權限。


```
@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 字符來驗證使用者、產生唯一識別符，並視需要授予使用者存取其他 AWS 資源的權限。

```
(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 字符來驗證使用者、產生唯一識別符，並視需要授予使用者存取其他 AWS 資源的權限。

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 Sign in with Apple : JavaScript

Apple 不提供支援適用於 JavaScript 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 JavaScript 使用者介面，請遵循 Apple 文件中的在 [Web 上顯示 Sign in with Apple 按鈕](#)。
- 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的 ID 字符合來驗證使用者、產生唯一識別符，並視需要授予使用者存取其他 AWS 資源的權限。

```
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.  
    });  
}
```

將 OIDC 供應商設定為身分集區 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。使用 Amazon Cognito，您可以將身分與透過設定的 OpenID Connect 提供者連結[AWS Identity and Access Management](#)。

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 AWS 帳戶中的 IAM IdP 選擇一個 OIDC 身分提供者。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。

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

請參閱供應商的文件，了解如何登入和接收 ID 字符。

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。

Note

身分集區會短暫維護 OIDC IdP 簽署金鑰的快取。如果您的提供者變更其簽署金鑰，Amazon Cognito 可能會傳回 `NoKeyFound` 錯誤，直到此快取重新整理為止。如果您遇到此錯誤，請等待約十分鐘讓身分集區重新整理簽署金鑰。

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 }
```

JavaScript

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({  
  IdentityPoolId: 'IDENTITY_POOL_ID',  
  Logins: {  
    'login.provider.com': token  
  }  
});
```

將 SAML 供應商設定為身分集區 IdP

透過 Amazon Cognito 身分集區，您可以透過 SAML 2.0 向身分提供者 (IdPs) 驗證使用者。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 帳戶中的 IAM IdP 選擇一個 SAML 身分提供者。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。透過許多 IdP，您便可以指定 URL，以供 IdP 用於從 XML 文件讀取依賴方資訊和憑證。對於 AWS，您可以使用 <https://signin.aws.amazon.com/static/saml-metadata.xml>。下一個步驟是從 IdP 設定 SAML 聲明回應，以填入 AWS 所需的宣告。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，如果 SAML 聲明符合 SAML 中繼資料中的任何憑證，則在登入時，您的身分集區會判斷 SAML 聲明是否有效。

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 聯合，請判斷使用者起始 login. AWS federation 使用 IdP 起始登入的 URL。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 Logins 對應中接受 SAML 判斷提示的 API 作業包括 [GetId](#)、[GetCredentialsForIdentity](#)、[GetOpenIdToken](#)，以及 [GetOpenIdTokenForDeveloperIdentity](#)。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。

開發人員驗證的身分

除了透過 [將 Facebook 設定為身分集區 IdP](#)、[將 Google 設定為身分集區 IdP](#)、[將 Login with Amazon 設定為身分集區 IdP](#) 和 [設定使用 Apple 登入做為身分集區 IdP](#) 的 Web 聯合身分，Amazon Cognito 還支援開發人員驗證的身分。透過開發人員驗證身分，您可以透過自己的現有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 AWS 資源。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 AWS 部落格中的 [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

[GetOpenIdTokenForDeveloperIdentity](#) API 操作可以啟動增強型和基本身分驗證的開發人員身分驗證。此 API 會使用管理登入資料來驗證請求。Logins 映射是身分集區開發人員提供者名稱，就像與自訂識別符 `login.mydevprovider` 配對。

範例：


```
"Logins": {
  "login.mydevprovider": "my developer identifier"
}
```

增強型身分驗證

呼叫 [GetCredentialsForIdentity](#) API 操作，並使用來自 的字符名稱cognito-identity.amazonaws.com和值的Logins映射GetOpenIdTokenForDeveloperIdentity。

範例：

```
"Logins": {
  "cognito-identity.amazonaws.com": "eyJra12345EXAMPLE"
}
```

GetCredentialsForIdentity 使用開發人員驗證的身分，會傳回身分集區之預設驗證角色的臨時憑證。

基本身分驗證

呼叫 [AssumeRoleWithWebIdentity](#) API 操作，並請求任何已[定義適當信任關係](#)RoleArn之 IAM 角色的。將 的值WebIdentityToken設定為從 取得的字符GetOpenIdTokenForDeveloperIdentity。

如需開發人員驗證身分身分驗證流程及其與外部提供者身分有何不同的相關資訊，請參閱 [身分集區身分驗證流程](#)。

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。

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串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，此類別會延伸 [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，此類別會延伸 [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 [GetOpenIdTokenForDeveloperIdentity](#) 來取得權杖。必須使用 AWS 開發人員登入資料從後端叫用此 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。如果攻擊者取得此字符，他們可以在字符持續時間內將其交換為最終使用者的 AWS 憑證。

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 * 151);
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。若要將自訂身分連接至使用者的社交身分 (Login with Amazon、Sign in with Apple、Facebook 或 Google)，請在呼叫 [GetOpenIdTokenForDeveloperIdentity](#) 時，將身分提供者權杖新增至呼叫對應。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，會呼叫 `getIdentityId` 方法並傳回無效值。

```
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 [AWSIdentityProviderHelper](#) `logins` 方法，使其能夠依據目前的身分供應商，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。

```

- (AWSIdentityProvider *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSIdentityProvider taskWithResult:nil];
    }else if (/*logic to determine if user is Facebook*/){
        return [AWSIdentityProvider 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

[AWSCognitoCredentialsProviderHelper](#) `logins` 方法，使其能夠依據目前的身分供應商，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，建議的作法是從 `AmazonCognitoEnhancedIdentityProvider` (而不是從 `AbstractCognitoIdentityProvider`) 擴展您的身分提供者，並呼叫父系 `RefreshAsync` 方法 (而不是自己的方法)，以免使用者未向您自己的後端驗證身分。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。

將未驗證的使用者切換為已驗證的使用者

Amazon Cognito 身分集區可支援已驗證和未驗證的使用者。未經驗證的使用者即使未透過任何身分提供者 (IdPs) 登入，也可以存取您的 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` 介面來通知您的應用程式。請在該介面中實作 `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 適用於 JavaScript 的 AWS SDK 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.IdentityChangedEventArgs 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.IdentityChangedEventArgs 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](#)。如同 Amazon Cognito Sync，AWS AppSync 是一項跨裝置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](#)
- [跨用戶端同步資料](#)
- [處理事件回呼](#)
- [實作推送同步](#)
- [實作 Amazon Cognito Sync 串流](#)
- [使用 Amazon Cognito Events 自訂工作流程](#)

Amazon Cognito Sync 入門

⚠ 如果您第一次使用 Amazon Cognito Sync，請改用 [AWS AppSync](#)。如同 Amazon Cognito Sync，AWS AppSync 是一項跨裝置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 是一種 AWS 服務和用戶端程式庫，可啟用應用程式相關使用者資料的跨裝置同步。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 [Amazon Cognito 身分集區入門](#)。

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。如需詳細資訊，請參閱 [跨用戶端同步資料](#)。

跨用戶端同步資料

 如果您第一次使用 Amazon Cognito Sync，請改用 [AWS AppSync](#)。如同 Amazon Cognito Sync，AWS AppSync 是一項跨裝置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。登入資料提供者會取得臨時 AWS 登入資料，讓您的應用程式能夠存取您的 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[處理事件回呼](#)。

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 AWS Cognito 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[處理事件回呼](#)。

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[處理事件回呼](#)。

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[處理事件回呼](#)。

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[處理事件回呼](#)。

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[處理事件回呼](#)。

處理事件回呼

 如果您第一次使用 Amazon Cognito Sync，請改用 [AWS AppSync](#)。如同 Amazon Cognito Sync，AWS AppSync 是一項跨裝置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。

身為 Amazon Cognito Sync 開發人員，您可以實作各種回呼來處理不同的同步事件和案例。Android SDK 中的 SyncCallback 介面會設定資料集同步的通知，包括 onSuccess() 何時成功下載資料集、onFailure() 何時發生例外狀況，以及 onConflict() 解決本機和遠端資料之間的衝突。

在 iOS 開發套件中，您可以註冊類似通知，例如，AWSCognitoDidStartSynchronizeNotification 並設定處理常式，例如

`AWSCognitoRecordConflictHandler` 以解決衝突。JavaScript、Unity 和 Xamarin 平台具有類似的回呼機制。當您實作這些回呼時，您的應用程式可以正常地處理使用 Amazon Cognito Sync 時可能發生的各種同步事件和案例。

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 = AWSognito.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 = AWSognito.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;
}
```

實作推送同步

 如果您第一次使用 Amazon Cognito Sync，請改用 [AWS AppSync](#)。如同 Amazon Cognito Sync，AWS AppSync 是一項跨裝置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 角色設定為具有完整的 Amazon SNS 存取權，或建立具有完整 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 AWS Cognito 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 AWS Cognito 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) {

    NotificationCenter.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。

實作 Amazon Cognito Sync 串流

 如果您第一次使用 Amazon Cognito Sync，請改用 [AWS AppSync](#)。如同 Amazon Cognito Sync，AWS AppSync 是一項跨裝置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。

使用 Amazon Cognito Events 自訂工作流程

 如果您第一次使用 Amazon Cognito Sync，請改用 [AWS AppSync](#)。如同 Amazon Cognito Sync，AWS AppSync 是一項跨裝置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 Events 可讓您執行 AWS Lambda 函數，以回應 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 UpdateRecords AWS 帳戶登入資料（開發人員登入資料）呼叫，則不會叫用您的 Lambda 函數。

在 中建立 函數 AWS Lambda

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 主控台新增此許可（請參閱[使用資源型政策](#)）[AWS Lambda](#)，或使用 Lambda [AddPermission](#) 操作。

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);
};
```

Amazon Cognito 的安全

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，該架構專為滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您之間의 共同責任。[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 Cloud 中執行 AWS 服務的基礎設施。AWS 也為您提供可安全使用的服務。在[AWS 合規計畫](#)中，第三方稽核人員會定期測試和驗證我們的安全有效性。若要了解適用於 Amazon Cognito 的合規計畫，請參閱[AWS 合規計畫的服務範圍](#)。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。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。您也會了解如何使用其他 AWS 服務來協助您監控和保護 Amazon Cognito 資源。

目錄

- [Amazon Cognito 的資料保護](#)
- [Amazon Cognito 的 Identity and Access Management](#)
- [在 Amazon Cognito 中記錄和監控](#)
- [Amazon Cognito 的合規驗證](#)
- [Amazon Cognito 的恢復能力](#)
- [Amazon Cognito 的基礎設施安全](#)
- [Amazon Cognito 使用者集區中的組態與漏洞分析](#)
- [AWS Amazon Cognito 的 受管政策](#)

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 全球基礎設施。您負責維護在此基礎設施上託管內容的控制權。此內容包含您使用之 AWS 服務的安全組態和管理任務。如需有關資料隱私權的詳細資訊，請參閱[資料隱私權常見問答集](#)。

基於資料保護目的，我們建議您保護 AWS 帳戶登入資料，並使用 AWS Identity and Access Management (IAM) 設定個別使用者帳戶。如此一來，每個使用者都只會獲得授予完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。
- 使用設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案，以及服務中的所有 AWS 預設安全控制。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Simple Storage Service (Amazon Simple Storage Service (Amazon S3)) 的個人資料。

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 Amazon Cognito 或使用主控台 AWS CLI、API 或 AWS SDKs 的其他 AWS 服務時。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

Amazon Cognito 是受管服務，受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及如何 AWS 保護基礎設施的資訊，請參閱[AWS 雲端安全](#)。若要使用基礎設施安全的最佳實務設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的[基礎設施保護](#)。

您可以使用 AWS 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[依授權模型分組的 API 操作清單](#)。

 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) 是 AWS 服務，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員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。IAM 是您可以免費使用 AWS 服務的。

主題

- [目標對象](#)
- [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](#)

目標對象

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，取決於您在 Amazon Cognito 中執行的工作。

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

身分驗證是您 AWS 使用身分憑證登入的方式。您必須以 AWS 帳戶根使用者身分、IAM 使用者身分或擔任 IAM 角色來驗證（登入 AWS）。

您可以使用透過身分來源提供的憑證，以聯合身分 AWS 身分登入。AWS IAM Identity Center (IAM Identity Center) 使用者、您的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料，都是聯合身分的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使用聯合 AWS 身分存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需登入的詳細資訊 AWS，請參閱 AWS 登入《使用者指南》中的 [如何登入您的 AWS 帳戶](#)。

如果您以 AWS 程式設計方式存取，AWS 會提供軟體開發套件 (SDK) 和命令列界面 (CLI)，以使用您的憑證以密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，則必須自行簽署請求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱《IAM 使用者指南》中的 [適用於 API 請求的 AWS Signature 第 4 版](#)。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如，AWS 建議您使用多重驗證 (MFA) 來提高帳戶的安全性。如需更多資訊，請參閱《AWS IAM Identity Center 使用者指南》中的 [多重要素驗證](#) 和《IAM 使用者指南》中的 [IAM 中的 AWS 多重要素驗證](#)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取帳戶中的所有 AWS 服務和資源。此身分稱為 AWS 帳戶 Theroot 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的 [需要根使用者憑證的任務](#)。

聯合身分

最佳實務是，要求人類使用者，包括需要管理員存取權的使用者，使用臨時 AWS 服務憑證與身分提供者聯合來存取。

聯合身分是您企業使用者目錄、Web 身分提供者、AWS Directory Service、Identity Center 目錄，或 AWS 服務是透過身分來源提供的登入資料存取的任何使用者。當聯合身分存取時 AWS 帳戶，它們會擔任角色，而角色會提供臨時登入資料。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連接並同步到您自己的身分來源中的一組使用者 AWS 帳戶和群組，以便在所有和應用程式中使用。如需 IAM Identity Center 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的[什麼是 IAM Identity Center？](#)。

IAM 使用者和群組

[IAM 使用者](#)是中的身分 AWS 帳戶，具有單一人員或應用程式的特定許可。建議您盡可能依賴臨時憑證，而不是擁有建立長期憑證（例如密碼和存取金鑰）的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱[IAM 使用者指南](#)中的為需要長期憑證的使用案例定期輪換存取金鑰。

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

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供臨時憑證。如需更多資訊，請參閱《IAM 使用者指南》中的[IAM 使用者的使用案例](#)。

IAM 角色

[IAM 角色](#)是中具有特定許可 AWS 帳戶的身分。它類似 IAM 使用者，但不與特定的人員相關聯。若要暫時在中擔任 IAM 角色 AWS Management Console，您可以從[使用者切換至 IAM 角色（主控台）](#)。您可以透過呼叫 AWS CLI 或 AWS API 操作或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的[擔任角色的方法](#)。

使用臨時憑證的 IAM 角色在下列情況中非常有用：

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

訊，請參閱《[IAM 使用者指南](#)》中的為第三方身分提供者 (聯合) 建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的[許可集](#)。

- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。不過，對於某些 AWS 服務，您可以將政策直接連接到資源 (而不是使用角色做為代理)。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的[IAM 中的跨帳戶資源存取](#)。
- 跨服務存取 – 有些 AWS 服務 使用其他 中的功能 AWS 服務。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
- 轉送存取工作階段 (FAS) – 當您使用 IAM 使用者或角色在其中執行動作時 AWS，您會被視為委託人。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。只有在服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《[轉發存取工作階段](#)》。
- 服務角色 – 服務角色是服務擔任的 [IAM 角色](#)，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[建立角色以委派許可權給 AWS 服務](#)。
- 服務連結角色 – 服務連結角色是連結至的一種服務角色 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 – 您可以使用 IAM 角色來管理在 EC2 執行個體上執行之應用程式的臨時登入資料，以及提出 AWS CLI 或 AWS API 請求。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並將其提供給其所有應用程式，您可以建立連接至執行個體的執行個體描述檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得臨時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的[使用 IAM 角色來授予許可權給 Amazon EC2 執行個體上執行的應用程式](#)。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策是 中的物件，AWS 當與身分或資源相關聯時，會定義其許可。當委託人 (使用者、根使用者或角色工作階段) 發出請

求時，會 AWS 評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數政策會以 JSON 文件 AWS 的形式存放在 中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 [JSON 政策概觀](#)。

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該政策的使用者可以從 AWS Management Console AWS CLI、或 API AWS 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的 [透過客戶管理政策定義自訂 IAM 許可](#)。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策是獨立的政策，您可以連接到 中的多個使用者、群組和角色 AWS 帳戶。受管政策包括 AWS 受管政策和客戶受管政策。如需了解如何在受管政策及內嵌政策之間選擇，請參閱《IAM 使用者指南》中的 [在受管政策和內嵌政策間選擇](#)。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中 [指定主體](#)。委託人可以包含帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 AWS WAF 和 Amazon VPC 是支援 ACLs 的服務範例。如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的 [存取控制清單 \(ACL\) 概觀](#)。

其他政策類型

AWS 支援其他較不常見的政策類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 Principal 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 [IAM 實體許可界限](#)。
- 服務控制政策 SCPs) – SCPs 是 JSON 政策，可指定中組織或組織單位 (OU) 的最大許可 AWS Organizations。AWS Organizations 是一種用於分組和集中管理您企業擁有 AWS 帳戶之多個的服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的許可，包括每個實體 AWS 帳戶根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的 [服務控制政策](#)。
- 資源控制政策 (RCP) - RCP 是 JSON 政策，可用來設定您帳戶中資源的可用許可上限，採取這種方式就不需要更新附加至您所擁有的每個資源的 IAM 政策。RCP 會限制成員帳戶中資源的許可，並可能影響身分的有效許可，包括 AWS 帳戶根使用者，無論它們是否屬於您的組織。如需 Organizations 和 RCPs 的詳細資訊，包括支援 RCPs AWS 服務的清單，請參閱 AWS Organizations 《使用者指南》中的 [資源控制政策 \(RCPs\)](#)。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過撰寫程式的方式建立角色或聯合使用者的暫時工作階段時，做為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的 [工作階段政策](#)。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解如何 AWS 在涉及多種政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的 [政策評估邏輯](#)。

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
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

若要全面了解 Amazon Cognito 和其他 AWS 服務如何與大多數 IAM 功能搭配使用，請參閱 [《AWS IAM 使用者指南》](#) 中的 [與 IAM 搭配使用的服務](#)。

Amazon Cognito 身分型政策

支援身分型政策：是

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱 [《IAM 使用者指南》](#) 中的 [透過客戶管理政策定義自訂 IAM 許可](#)。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱 [《IAM 使用者指南》](#) 中的 [IAM JSON 政策元素參考](#)。

Amazon 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

支援資源型政策：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](#)。委託人可以包含帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

如需啟用跨帳戶存取權，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，做為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當委託人和資源位於不同位置時 AWS 帳戶，信任帳戶中的 IAM 管理員也必須授予委託人實體（使用者或角色）存取資源的許可。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM 中的快帳戶資源存取](#)。

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內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 Action 元素描述您可以用來允許或拒絕政策中存取的動作。政策動作通常具有與相關聯 AWS API 操作相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看 Amazon 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 AWS 登入資料簽署 Amazon Cognito API 請求時，您可以在 AWS Identity and Access Management (IAM) 政策中限制這些請求。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。如需已簽署 API 請求的詳細資訊，請參閱[簽署 AWS API 請求](#)。

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 [依授權模型分組的 API 操作清單](#)。

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內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用 [條件運算子](#) 的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵，AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，會使用邏輯 OR 操作 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 [IAM 政策元素：變數和標籤](#)。

AWS 支援全域條件金鑰和服務特定的條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的 [AWS 全域條件內容索引鍵](#)。

若要查看 Amazon 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 AWS 中，這些屬性稱為標籤。您可以將標籤連接至 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

如需根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的 [條件元素](#) 中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的 [使用 ABAC 授權定義許可](#)。如要查看含有設定 ABAC 步驟的教學課程，請參閱 IAM 使用者指南中的 [使用屬性型存取控制 \(ABAC\)](#)。

將暫時憑證與 Amazon Cognito 搭配使用

支援臨時憑證：是

當您使用臨時登入資料登入時，有些 AWS 服務 無法運作。如需詳細資訊，包括哪些 AWS 服務 使用臨時登入資料，請參閱 [《AWS 服務 IAM 使用者指南》中的 使用 IAM](#)。

如果您 AWS Management Console 使用使用者名稱和密碼以外的任何方法登入，則會使用臨時登入資料。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立臨時登入資料。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的 [從使用者切換至 IAM 角色 \(主控台\)](#)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱 [IAM 中的暫時性安全憑證](#)。

Amazon 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，您會被視為委託人。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫 的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。只有在服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱[轉發存取工作階段](#)。

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。如需詳細資訊，請參閱《IAM 使用者指南》中的[建立角色以委派許可權給 AWS 服務](#)。

如需 Amazon Cognito 服務角色的詳細資訊，請參閱 [啟用推播同步](#) 和 [實作推送同步](#)。

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 的一種服務角色 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理 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 來執行任務。若要授

予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

如需了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的[建立 IAM 政策 \(主控台\)](#)。

如需 Amazon 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 受管政策並轉向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，進一步減少許可。如需更多資訊，請參閱 IAM 使用者指南中的 [AWS 受管政策](#) 或 [任務職能的 AWS 受管政策](#)。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 [IAM 中的政策和許可](#)。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。如果透過特定 使用服務動作，您也可以使用條件來授予其存取權 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 [IAM JSON 政策元素：條件](#)。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 [使用 IAM Access Analyzer 驗證政策](#)。

- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_configure-api-require.html 中的透過 MFA 的安全 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 [IAM 安全最佳實務](#)。

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內的 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和檢視中 Amazon Cognito 資源的詳細資訊 AWS 帳戶。如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許最低主控台許可。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

為了確保使用者和角色仍然可以使用 Amazon Cognito 主控台，請將 Amazon Cognito `ConsoleAccess` 或 `ReadOnly` AWS 受管政策連接到實體。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。此政策包含在主控台或使用或 AWS CLI AWS API 以程式設計方式完成此動作的許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
```

```

        "Effect": "Allow",
        "Action": [
            "iam:GetUserPolicy",
            "iam:ListGroupsForUser",
            "iam:ListAttachedUserPolicies",
            "iam:ListUserPolicies",
            "iam:GetUser"
        ],
        "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
        "Sid": "NavigateInConsole",
        "Effect": "Allow",
        "Action": [
            "iam:GetGroupPolicy",
            "iam:GetPolicyVersion",
            "iam:GetPolicy",
            "iam:ListAttachedGroupPolicies",
            "iam:ListGroupPolicies",
            "iam:ListPolicyVersions",
            "iam:ListPolicies",
            "iam:ListUsers"
        ],
        "Resource": "*"
    }
]
}

```

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](#)
- [我未獲得執行 iam:PassRole 的授權](#)

- [我是管理員，並且想要允許其他人存取 Amazon Cognito](#)
- [我想要允許 AWS 帳戶外的人員存取我的 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 管理員。您的管理員提供您的簽署憑證。

我未獲得執行 iam:PassRole 的授權

如果您收到錯誤，告知您無權執行 `iam:PassRole` 動作，則必須更新您的政策，您才能將角色傳遞至 Amazon Cognito。

有些 AWS 服務可讓您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為 marymajor 的 IAM 使用者嘗試使用主控台在 Amazon 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 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，您必須將許可授予需要存取的人員或應用程式。如果您使用 AWS IAM Identity Center 來管理人員和應用程式，您可以將許可集指派給使用者或群組，以定義其存

取層級。許可集會自動建立 IAM 政策，並將其指派給與該人員或應用程式相關聯的 IAM 角色。如需詳細資訊，請參閱AWS IAM Identity Center 《使用者指南》中的[許可集](#)。

如果您不是使用 IAM Identity Center，則必須為需要存取的人員或應用程式建立 IAM 實體（使用者或角色）。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。授予許可後，請將登入資料提供給使用者或應用程式開發人員。他們將使用這些登入資料來存取 AWS。若要進一步了解如何建立 IAM 使用者、群組、政策和許可，請參閱《IAM [使用者指南](#)》中的 [IAM 身分](#)和[政策和許可](#)。

我想要允許 AWS 帳戶外的人員存取我的 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](#)。
- 若要了解如何提供您擁有 AWS 帳戶 的資源存取權，請參閱《[IAM 使用者指南](#)》中的在您擁有 AWS 帳戶 的另一個 中為 IAM 使用者提供存取權。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的[將存取權提供給第三方 AWS 帳戶 擁有](#)。
- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的[將存取權提供給在外部進行身分驗證的使用者 \(聯合身分\)](#)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 [IAM 中的跨帳戶資源存取](#)。

使用 Amazon Cognito 的服務連結角色

Amazon Cognito 使用 AWS Identity and Access Management (IAM) [服務連結角色](#)。服務連結角色是一種獨特的 IAM 角色類型，具有允許 AWS 服務 擔任該角色的信任政策。服務連結角色是由 Amazon Cognito 預先定義，並包含該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 Pinpoint 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 `AWSServiceRoleForAmazonCognitoIdpEmailService` 動作：

- 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

AWSServiceRoleForAmazonCognitoIdp 服務連結角色信任下列服務可擔任該角色：

- `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 AWSServiceRoleForAmazonCognitoIdp 動作

- 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、AWS CLI 或 Amazon Cognito API 中的電子郵件傳遞時，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

您無法在其中編輯 AmazonCognitoIdp 或 AmazonCognitoIdpEmailService 服務連結角色 AWS Identity and Access Management。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](#)」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。在可刪除 AmazonCognitoIdp 或 AmazonCognitoIdpEmailService 服務連結角色之前，您必須對每個使用此角色的使用者集區執行下列任一動作：

- 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。

請記得在 AWS 區域 具有使用 角色之使用者集區的每個 中執行 動作。

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 並開啟位於 的 Amazon Cognito 主控台<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 並開啟位於 的 Amazon Cognito 主控台<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 來刪除 AmazonCognitoIdp 或 AmazonCognitoIdpEmailService 服務連結角色。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 在提供服務的所有 AWS 區域 中支援服務連結角色。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 解決方案可靠性、可用性和效能的重要部分。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 – 使用 CloudTrail，您可以從 Amazon Cognito 主控台以及對 Amazon Cognito API 操作的程式碼呼叫擷取 API 呼叫。例如，當使用者進行身分驗證時，CloudTrail 可以記錄詳細資訊，例如請求中的 IP 地址、提出請求的人員，以及提出請求的時間。
- Amazon CloudWatch Logs — 透過 CloudWatch Logs，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，您可以近乎即時地監控、報告，以及在發生事件時採取自動動作。例如，您可以在提供的指標上建立 CloudWatch 儀表板來監控 Amazon Cognito 使用者集區，或者您可以在提供的指標上建立 CloudWatch 警示，以便在違反設定的閾值時通知您。

- Amazon CloudWatch Logs Insights – 使用 CloudWatch Logs Insights，您可以設定 CloudTrail 將事件傳送到 CloudWatch，以監控 Amazon Cognito CloudTrail 日誌檔案。

主題

- [監控和管理成本](#)
- [從 Amazon Cognito 使用者集區匯出日誌](#)
- [追蹤 CloudWatch 和 Service Quotas 中的配額和使用量](#)
- [Amazon Cognito 登入 AWS CloudTrail](#)

監控和管理成本

如同任何其他一樣 AWS 服務，請務必了解 Amazon Cognito 組態和用量對 AWS 帳單的影響。在準備將使用者集區部署到生產環境時，請設定活動和資源消耗的監控和防護措施。當您知道要查看何處以及哪些動作會產生額外的成本時，您可以在帳單中設定意外的預防措施。

Amazon Cognito 會針對您的用量收取下列維度的費用。

- 使用者集區每月作用中使用者 MAUs) — 費率因[功能計劃](#)而異
- 使用 OIDC 或 SAML 聯合登入的使用者集區 MAUs
- 使用用戶端憑證授予的作用中使用者集區應用程式用戶端和機器對機器 (M2M) 授權的請求磁碟區
- 對於某些類別的使用者集區 APIs，購買的用量超過預設配額

此外，使用者集區的功能，例如電子郵件訊息、簡訊和 Lambda 觸發，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

產品啟動和開放至新使用者群等大量事件可能會增加您的 MAU 計數，並產生成本影響。預先估計新使用者計數，並在活動發生時監看。您可能會發現您想要在購買額外的配額容量時容納磁碟區，或使用額外的安全措施來控制磁碟區。

您可以在 [AWS 帳單與成本管理 主控台](#) 中檢視和報告您的 AWS 成本。您可以在帳單和付款區段中找到 Amazon Cognito 的最新費用。在依服務計費下，篩選 Cognito 以檢視您的用量。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 請求率，請在 Service Quotas 主控台中檢閱使用率指標。例如，用戶端登入資料請求會顯示為 ClientAuthentication 請求的速率。在您的帳單中，這些請求會與產生請求的應用程式用戶端相關聯。透過此資訊，您可以公平分配成本給[多租戶架構中的租戶](#)。

若要取得一段時間內的 M2M 請求計數，您也可以將[AWS CloudTrail 事件傳送至 CloudWatch Logs](#) 進行分析。使用用戶端憑證授予來查詢 CloudTrail Token_POST 事件。下列 CloudWatch Insights 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 會根據使用者計數、功能用量和請求量計費。以下是在 Amazon Cognito 中管理成本的一些秘訣，

請勿啟用非作用中的使用者

讓使用者處於作用中狀態的典型操作是登入、註冊和密碼重設。如需更完整的清單，請參閱[每月作用中使用者](#)。Amazon Cognito 不會將非作用中使用者計入您的帳單。避免任何將使用者設為作用中的操作。使用 [ListUsers](#) 操作查詢使用者，而不是 [AdminGetUser](#) API 操作。請勿對非作用中的使用者執行使用者集區操作的大量管理測試。

連結聯合身分使用者

使用 SAML 2.0 或 OpenID Connect (OIDC) 身分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](#)。連結的使用者可以以本機使用者身分登入，其中包含與其聯合身分使用者一起提供的屬性和存取權。來自 SAML 或 OIDC IdPs 的使用者，在一個月內，只會以本機使用者身分使用連結的本機帳戶登入。

管理請求率

如果您的使用者集區接近配額上限，您可以考慮購買額外的容量來處理磁碟區。您可能可以減少應用程式中的請求量。如需詳細資訊，請參閱[最佳化配額限制的請求率](#)。

只在您需要新權杖時請求新權杖

具有用戶端憑證授予的機器對機器 (M2M) 授權可以達到大量的字符請求。每個新的字符請求都會影響您的請求率配額和帳單大小。若要最佳化成本，請在應用程式的設計中包含權杖過期設定和權杖處理。

- [快取存取權杖](#)，以便在應用程式請求新的權杖時，它會收到先前發行權杖的快取版本。當您實作此方法時，快取代理會保護請求存取字符的應用程式，而不會察覺先前取得的字符過期。快取字符非常適合短期微服務，例如 Lambda 函數和 Docker 容器。
- 在考量權杖過期的應用程式中實作權杖處理機制。在先前的權杖即將過期之前，請勿請求新的權杖。最佳實務是在字符生命週期的 75% 左右重新整理字符。此實務可將字符持續時間最大化，同時確保應用程式中的使用者連續性。

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發行具有適當有效期間的存取權杖。自訂字符持續時間最適合長期 APIs 和伺服器，這些伺服器可以持續管理憑證請求的頻率。

ListUsers，而非 AdminGetUser

若要查詢使用者集區中使用者的屬性，請盡可能使用 [ListUsers](#) API 操作和相關聯的 [SDK](#) 方法。[AdminGetUser](#) 將使用者標示為當月作用中，並對用來計算使用者集區帳單的每月作用中使用者 (MAUs) 做出貢獻。

刪除未使用的用戶端登入資料應用程式用戶端

M2M 授權會根據兩個因素計費：字符請求的速率，以及執行用戶端憑證授予的應用程式用戶端數量。當 M2M 授權的應用程式用戶端未使用時，請刪除它們或移除其發出用戶端憑證的授權。如需管理應用程式用戶端組態的詳細資訊，請參閱 [使用應用程式用戶端的應用程式特定設定](#)。

管理功能計劃

當您在使用者集區中選擇[功能計劃](#)時，帳單費率會套用至使用者集區中的所有 MAUs。如果您有不需更高層級功能計劃之功能的使用者，請將他們分開到另一個使用者集區。

從 Amazon Cognito 使用者集區匯出日誌

您可以設定使用者集區，將一些額外活動的詳細日誌傳送至另一個活動 AWS 服務，例如 CloudWatch 日誌群組。這些日誌比中的日誌更精細 AWS CloudTrail，對於疑難排解您的使用者集區以及使用[進階安全功能](#)分析使用者登入活動非常有用。當您想要串流簡訊和電子郵件通知錯誤的日誌時，您的使用者集區會將 ERROR 層級日誌傳送至 CloudWatch 日誌群組。當您想要串流使用者登入活動的日誌時，您的使用者集區會將 INFO 層級日誌傳送至日誌群組、Amazon Data Firehose 串流或 Amazon S3 儲存貯體。您可以在使用者集區中結合這兩個選項。

主題

- [日誌匯出須知事項](#)

- [匯出電子郵件和簡訊傳遞錯誤](#)
- [匯出威脅防護使用者活動日誌](#)

日誌匯出須知事項

成本影響

Amazon Data Firehose、Amazon S3 和 CloudWatch Logs 會產生資料擷取和擷取的成本。您的記錄組態可能會影響您的 AWS 帳單。如需詳細資訊，請參閱下列內容：

- Amazon CloudWatch 定價中的[已修訂日誌](#)。
- [Amazon Data Firehose 定價](#)
- [Amazon S3 定價](#)

使用者活動日誌匯出包含安全評估，是使用者集區[進階安全功能的](#)函數。Amazon Cognito 只會在進階安全功能處於作用中狀態時產生這些日誌。這些功能會增加使用者集區中每個每月作用中使用者 (MAU) 的成本。如需詳細資訊，請參閱 [Amazon Cognito 定價](#)。

使用者活動日誌是INFO關卡

匯出的使用者活動日誌僅處於INFO錯誤層級，並提供身分驗證活動統計和安全性分析的資訊。WARNING 和 ERROR錯誤層級的訊息，例如限流錯誤，不會包含在匯出的日誌中。

盡最大努力交付

最好從 Amazon Cognito 傳遞日誌。您的使用者集區提供的日誌數量，以及 CloudWatch Logs、Amazon S3 和 Firehose 的服務配額可能會影響日誌的交付。

現有的外部日誌不受影響

這些記錄選項不會取代或變更使用者集區的下列日誌函數。

1. CloudTrail 日誌，內容包含像是註冊和登入等例行使用者活動。
2. 使用 CloudWatch 指標大規模分析使用者活動。

此外，您也可以從 CloudWatch Logs [使用 Lambda 觸發程序來自訂使用者集區工作流程](#)中找到來自 [在 CloudWatch 主控台中檢視使用者集區匯入結果](#)和 的日誌。Amazon Cognito 和 Lambda 將這些日誌存放在與您為使用者活動日誌指定的日誌群組不同的日誌群組中。

僅適用於使用者集區

身分集區中不存在日誌匯出功能。

需要使用者許可和服務連結角色

設定日誌匯出的 AWS 主體必須具有修改目標資源的許可，如以下主題所述。Amazon Cognito 會代表您建立[服務連結角色](#)，並擔任該角色將日誌交付至目標資源。

如需有關從 Amazon Cognito 傳送日誌的授權模型的詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的[從 啟用記錄 AWS 服務](#)。

日誌層級專屬於日誌類型

訊息傳遞日誌類型為 `ERROR` 錯誤層級 `userNotification` 為 `進階安全使用者活動` 日誌類型為 `INFO` 錯誤層級 `userAuthEvents` 為 `進階安全使用者活動`。您可以合併兩個 成員 `LogConfigurations`，一個 `userNotification` 用於 CloudWatch Logs，另一個 `userAuthEvents` 用於 Firehose、Amazon S3 或 CloudWatch Logs。

您無法將使用者活動日誌傳送至多個目的地。您無法將使用者通知日誌傳送至 CloudWatch Logs 以外的任何目的地。

不同的組態選項

您只能使用 Amazon Cognito 使用者集區 API 或 AWS SDK 設定使用者通知日誌。您可以使用 API 或在 Amazon Cognito 主控台中設定進階安全使用者活動日誌。若要設定兩者，請使用 API，如[SetLogDeliveryConfiguration](#) 的範例請求所示。

大型資源型政策所需的其他組態

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[從特定 AWS 服務啟用記錄](#)。

在 Amazon S3 中自動建立資料夾

當您設定將威脅防護日誌匯出至 Amazon S3 儲存貯體時，Amazon Cognito 可能會在儲存貯體中建立 `AWSLogs` 資料夾。並非所有情況下都會建立該資料夾，而且組態可以在不建立的情況下成功。

匯出電子郵件和簡訊傳遞錯誤

對於電子郵件和簡訊傳遞錯誤，您可以從使用者集區傳遞錯誤層級的使用者通知日誌。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，使用者通知記錄非常有用。此日誌匯出選項與[使用者活動匯出](#)不同，不需要 Plus 功能計劃。

您可以在 [SetLogDeliveryConfiguration](#) API 請求中使用 Amazon Cognito 使用者集區 API 設定詳細通知日誌。您可以在 [GetLogDeliveryConfiguration](#) API 請求中檢視使用者集區的記錄組態。以下是請求內文的範例。

```
{
  "LogConfigurations": [
    {
      "CloudWatchLogsConfiguration": {
        "LogGroupArn": "arn:aws:logs:us-west-2:123456789012:log-group:example-user-pool-exported"
      },
      "EventSource": "userNotification",
      "LogLevel": "ERROR"
    }
  ],
  "UserPoolId": "us-west-2_EXAMPLE"
}
```

您必須使用具有下列許可的 AWS 登入資料來授權這些請求。

```
{
  "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"
  }
}

```

匯出威脅防護使用者活動日誌

具有 Plus 功能計劃和威脅防護日誌的使用者活動事件的使用者集區：使用使用者集區的使用者登入、登出和其他身分驗證操作的詳細資訊和安全性評估。您可能想要在自己的日誌管理系統中檢閱使用者活動日誌，或建立封存。您可以將此資料匯出至 Amazon CloudWatch Logs 日誌群組、Amazon Data Firehose 串流或 Amazon Simple Storage Service (Amazon S3) 儲存貯體。從那裡，您可以將此資料擷取到其他系統，以適合您的操作程序的方式來分析、標準化或以其他方式處理資料。若要匯出此類型

的資料，您的使用者集區必須位於 Plus 功能計劃中，而且[進階安全功能](#)必須在您的使用者集區中處於作用中狀態。

透過這些使用者活動日誌中的資訊，您可以檢視使用者登入和帳戶管理活動的設定檔。根據預設，Amazon Cognito 會將這些事件擷取到以使用者集區為基礎的儲存體。下列範例是登入且評估為沒有風險因素的使用者的範例事件。您可以使用 AdminListUserAuthEvents API 操作擷取此資訊。以下是輸出範例：

```
{
  "AuthEvents": [
    {
      "EventId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "EventType": "SignIn",
      "CreationDate": "2024-06-27T10:49:59.139000-07:00",
      "EventResponse": "Pass",
      "EventRisk": {
        "RiskDecision": "NoRisk",
        "CompromisedCredentialsDetected": false
      },
      "ChallengeResponses": [
        {
          "ChallengeName": "Password",
          "ChallengeResponse": "Success"
        }
      ],
      "EventContextData": {
        "IpAddress": "192.0.2.1",
        "DeviceName": "Chrome 126, Windows 10",
        "Timezone": "-07:00",
        "City": "null",
        "Country": "United States"
      }
    }
  ],
  "NextToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222#2024-06-27T17:49:59.139Z"
}
```

您可以在 Amazon Cognito 主控台或 [SetLogDeliveryConfiguration](#) API 操作中啟用使用者活動的日誌匯出。

AWS Management Console

1. 如果您還沒有要使用的日誌群組，請建立 [S3 儲存貯體](#)、[Firehose 串流](#)或 [CloudWatch 日誌群組](#)。
2. 登入 [Amazon Cognito 主控台](#)。
3. 選擇 User Pools (使用者集區)。
4. 從清單中選擇現有的使用者集區，或[建立使用者集區](#)。
5. 選擇進階安全索引標籤。尋找匯出使用者活動日誌，然後選擇編輯
6. 在記錄狀態下，選取啟用使用者活動日誌匯出旁的核取方塊。
7. 在記錄目的地下，選擇您要處理日誌 AWS 服務的：CloudWatch 日誌群組、Amazon Data Firehose 串流或 S3 儲存貯體。
8. 您的選擇會將對應的資源類型填入資源選取器。從清單中選擇日誌群組、串流或儲存貯體。您也可以選取建立按鈕，以導覽至所選服務的 AWS Management Console，並建立新的資源。
9. 選取儲存變更。

API

為您的使用者活動日誌選擇一種目的地類型。

以下是將 Firehose 串流設定為日誌目的地的範例SetLogDeliveryConfiguration請求內文。

```
{
  "LogConfigurations": [
    {
      "EventSource": "userAuthEvents",
      "FirehoseConfiguration": {
        "StreamArn": "arn:aws:firehose:us-west-2:123456789012:deliverystream/example-user-pool-activity-exported"
      },
      "LogLevel": "INFO"
    }
  ],
  "UserPoolId": "us-west-2_EXAMPLE"
}
```

以下是將 Amazon S3 儲存貯體設定為日誌目的地SetLogDeliveryConfiguration的請求內文範例。


```
{
  "LogConfigurations": [
    {
      "EventSource": "userAuthEvents",
      "S3Configuration": {
        "BucketArn": "arn:aws:s3:::amzn-s3-demo-logging-bucket"
      },
      "LogLevel": "INFO"
    }
  ],
  "UserPoolId": "us-west-2_EXAMPLE"
}
```

以下是將 CloudWatch 日誌群組設定為日誌目的地的SetLogDeliveryConfiguration請求內文範例。

```
{
  "LogConfigurations": [
    {
      "EventSource": "userAuthEvents",
      "CloudWatchLogsConfiguration": {
        "LogGroupArn": "arn:aws:logs:us-west-2:123456789012:log-group:DOC-EXAMPLE-LOG-GROUP"
      },
      "LogLevel": "INFO"
    }
  ],
  "UserPoolId": "us-west-2_EXAMPLE"
}
```

設定日誌交付的使用者必須是使用者集區管理員，並具有下列其他許可：

Amazon S3

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ManageUserPoolLogs",
      "Action": [
        "cognito-idp:SetLogDeliveryConfiguration",

```

```

        "cognito-idp:GetLogDeliveryConfiguration",
    ],
    "Resource": [
        "*"
    ],
    "Effect": "Allow"
},
{
    "Sid": "ManageLogsS3",
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogDelivery",
        "s3:PutBucketPolicy",
        "s3:GetBucketPolicy"
    ],
    "Resource": "*"
}
]
}

```

CloudWatch Logs

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ManageUserPoolLogs",
            "Action": [
                "cognito-idp:SetLogDeliveryConfiguration",
                "cognito-idp:GetLogDeliveryConfiguration",
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "ManageLogsCWL",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs>DeleteLogDelivery",
            ],

```

```

        "logs:ListLogDeliveries",
        "logs:PutResourcePolicy",
        "logs:DescribeResourcePolicies",
        "logs:DescribeLogGroups"
    ],
    "Resource": [
        "*"
    ],
    "Effect": "Allow"
}
]
}
```

Amazon Data Firehose

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ManageUserPoolLogs",
      "Action": [
        "cognito-idp:SetLogDeliveryConfiguration",
        "cognito-idp:GetLogDeliveryConfiguration",
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    },
    {
      "Sid": "ManageUserPoolLogsFirehose",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "iam:CreateServiceLinkedRole",
        "firehose:TagDeliveryStream"
      ],
      "Resource": "*"
    }
  ]
}
```

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_ACTIVITY",
  "logLevel": "INFO",
  "message": {
    "version": "1",
    "eventId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "eventType": "SignUp",
    "userSub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "userName": "test-user",
    "userPoolId": "us-west-2_EXAMPLE",
    "clientId": "lexample23456789",
    "creationDate": "Wed Jul 17 17:25:55 UTC 2024",
    "eventResponse": "InProgress",
    "riskLevel": "",
    "riskDecision": "PASS",
    "challenges": [],
    "deviceName": "Other, Other",
    "ipAddress": "192.0.2.1",
    "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
    "idpName": "",
    "compromisedCredentialDetected": "false",
    "city": "Seattle",
    "country": "United States",
    "eventFeedbackValue": "",
    "eventFeedbackDate": "",
    "eventFeedbackProvider": "",
    "hasContextData": "true"
  },
  "logSourceId": {
    "userPoolId": "us-west-2_EXAMPLE"
  }
}
```

追蹤 CloudWatch 和 Service Quotas 中的配額和使用量

您可以使用 Amazon CloudWatch 或 Service Quotas 來監控 Amazon Cognito 使用者集區。您也可以 Service Quotas 中監控身分集區使用情況。CloudWatch 可收集原始資料，將這些資料轉換為可讀取且幾近即時的指標。在 CloudWatch 中，您可以設定留意特定閾值的警示，當滿足這些閾值時傳送通知或採取動作。若要針對服務配額建立 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 CloudWatch 中保留期間的詳細資訊，請造訪 [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 CloudWatch 中保留期間的詳細資訊，請造訪 [Amazon CloudWatch 常見問答集頁面](#)。

如果您登入 AWS 帳戶的設定為 CloudWatch 跨帳戶可觀測性的監控帳戶，您可以使用該監控帳戶來視覺化服務配額，並為連結至該監控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 跨帳戶觀察功能](#)。

主題

- [CloudWatch 中的使用者集區指標](#)
- [Service Quotas 中的指標](#)

CloudWatch 中的使用者集區指標

使用者集區會以指標形式向 CloudWatch 報告使用者活動統計資料。從 CloudWatch，您可以分析使用者集區中的身分驗證活動量和配額用量。透過這些指標中的資訊，您可以為值得注意的事件設定警示，並視需要調整您的使用者集區組態。當使用者活動記錄具有使用者集區中使用者活動的詳細記錄時，CloudWatch 指標具有彙總統計資料和效能指標。

下表列出 Amazon Cognito 使用者集區可用的指標。Amazon Cognito 會將指標發佈至命名空間 AWS/Cognito 和 AWS/Usage。如需詳細資訊，請參閱和《Amazon CloudWatch 使用者指南》中的 [Namespaces](#)。

如需追蹤配額和用量的詳細資訊，請參閱 [追蹤配額使用量](#) 和 [追蹤每月作用中使用者 MAUs](#)。

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法是使用 CLI 中的 get-metric-data 或 AWS get-metric-statistics 命令。

指標	描述	命名空間
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。 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請求總數，請使用 CloudWatch Math 表達式並從 Sum 統計數字中減去 Sample Count 統計數字。 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	AWS/Cognito

指標	描述	命名空間
	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	AWS/Cognito

指標	描述	命名空間
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身分驗證請求總數，請使用 CloudWatch Math 表達式並從 Sample Count 統計數字中減去 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。 請注意，此指標中不包含重新整理 Amazon Cognito 權杖的	AWS/Cognito

指標	描述	命名空間
	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	
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	AWS/Cognito

指標	描述	命名空間
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。若要計算失敗的重新整理 Amazon Cognito 權杖請求總數，請使用 CloudWatch Math 表達式並從 Sample Count 統計數字中減去 Sum 統計數字。 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	AWS/Cognito

指標	描述	命名空間
	單位：計數	
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	AWS/Cognito

指標	描述	命名空間
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。若要計算失敗的聯合身分請求總數，請使用 CloudWatch Math 表達式並從 Sample Count 統計數字中減去 Sum 統計數字。 指標維度：UserPool、UserPoolClient、IdentityProvider 單位：計數	AWS/Cognito

指標	描述	命名空間
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	AWS/Cognito
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。 系統會針對 AWS 帳戶和區域中所有使用者集區的每個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	AWS/Usage

指標	描述	命名空間
ThrottleCount	提供與類別相關的調節呼叫總數。 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	AWS/Usage

檢視威脅防護指標

您的使用者集區發佈的指標具有有關威脅防護設定對使用者身分驗證活動之影響的統計資訊。您可能想知道有多少使用者嘗試使用遭入侵的登入資料登入。您也可以了解有多少百分比的登入活動被評估為具有一定程度的風險。Amazon Cognito 會在 Amazon CloudWatch 中將威脅防護功能的指標發佈至您的帳戶。Amazon Cognito 會依風險層級和請求層級將威脅防護指標分組在一起。

若要將內容新增至風險分析，您可以在[使用者集區或匯出的資料來源中檢視個別使用者登入嘗試的相關資訊](#)。

若要在 CloudWatch 主控台中檢視指標

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
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	命名空間
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：使用者集區的識別符。 RiskLevel：高 (預設)、中或低。	AWS/Cognito
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：使用者集區的識別符。 RiskLevel：高、中或低。	AWS/Cognito
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：使用者集區的識別符。	AWS/Cognito

指標	描述	指標維度	命名空間
		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：使用者集區的識別符。	AWS/Cognito
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：使用者集區的識別符。	AWS/Cognito

Amazon Cognito 提供您兩個預先定義的指標群組，以準備在 CloudWatch 中進行分析。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。

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 AWS 的服務名稱。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 CloudWatch 主控台追蹤指標

您可以使用 CloudWatch 追蹤和收集 Amazon Cognito 使用者集區指標。CloudWatch 儀表板會顯示您使用的每個 AWS 服務的指標。您可以使用 CloudWatch 建立指標警示。警示可設為傳送通知或變更您正在監控的特定資源。若要在 CloudWatch 中檢視服務配額指標，請完成下列步驟。

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。

為配額建立 CloudWatch 警示

Amazon Cognito 提供對應至 `CallCount` 和 `ThrottleCount` APIs AWS 服務配額的 CloudWatch 用量指標。如需在 CloudWatch 中追蹤用量的詳細資訊，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。若要了解如何使用 Service Quotas 主控台設定 CloudWatch 警示，請參閱[Service Quotas 和 CloudWatch 警示](#)。

Service Quotas 中的指標

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。對於部分配額類型，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。您可以在這裡將圖表新增至儀表板，或透過動作選單選取 View in metrics (在指標中檢視)，這會帶您前往 CloudWatch 主控台中的相關指標。

Amazon Cognito 登入 AWS CloudTrail

Amazon Cognito 已與服務整合 AWS CloudTrail，此服務提供 Amazon Cognito 中使用者、角色或 AWS 服務所採取動作的記錄。CloudTrail 將 Amazon Cognito 的 API 呼叫子集擷取為事件，包括來自 Amazon Cognito 主控台的呼叫，以及來自對 Amazon Cognito API 操作發出的程式碼呼叫。如果您建立追蹤，就可以選擇將 CloudTrail 事件傳遞至 Amazon S3 儲存貯體，包括 Amazon Cognito 的事件。即使您未設定追蹤，依然可以透過 CloudTrail 主控台內的 Event history (事件歷史記錄) 檢視最新

事件。您可以利用 CloudTrail 所收集的資訊來判斷向 Amazon Cognito 發出的請求，以及發出請求的 IP 地址、人員、時間和其他詳細資訊。

若要進一步了解 CloudTrail，包括如何設定及啟用，請參閱 [《AWS CloudTrail 使用者指南》](#)。

您也可以針對特定 CloudTrail 事件建立 Amazon CloudWatch 警示。例如，您可以設定 CloudWatch 在身分集區組態變更時觸發警示。如需詳細資訊，請參閱[建立 CloudTrail 事件的 CloudWatch 警示：範例](#)。

主題

- [Amazon Cognito 傳送至 CloudTrail 的資訊](#)
- [使用 Amazon CloudWatch Logs Insights 分析 Amazon Cognito CloudTrail 事件](#)
- [Amazon Cognito 事件範例](#)

Amazon Cognito 傳送至 CloudTrail 的資訊

當您建立時，CloudTrail 會開啟 AWS 帳戶。當 Amazon Cognito 中發生支援的事件活動時，該活動會與事件歷史記錄中的其他 AWS 服務事件一起記錄在 CloudTrail 事件中。您可以在 AWS 帳戶中檢視、搜尋和下載最近的事件。如需詳細資訊，請參閱[使用 CloudTrail 事件歷史記錄檢視事件](#)。

若要不持續記錄您 AWS 帳戶中的事件，包括 Amazon Cognito 的事件，請建立追蹤。CloudTrail 追蹤將日誌檔案傳遞至 Amazon S3 儲存貯體。根據預設，當您在主控台建立線索時，線索會套用到所有區域。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案傳送到您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱：

- [建立追蹤的概觀](#)
- [CloudTrail 支援的服務和整合](#)
- [設定 CloudTrail 的 Amazon SNS 通知](#)
- [接收多個區域的 CloudTrail 日誌檔案](#)和[接收多個帳戶的 CloudTrail 日誌檔案](#)

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 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 會使用值 `HIDDEN_FOR_SECURITY_REASONS` 隱藏 CloudTrail 事件中的某些私有欄位。如需 Amazon Cognito 未填入事件的欄位範例，請參閱 [Amazon Cognito 事件範例](#)。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。

使用者集區事件

Amazon Cognito 支援將 [User Pool Actions](#) (使用者集區動作) 頁面上列出的所有動作記錄為 CloudTrail 日誌檔案中的事件。Amazon Cognito 會將使用者集區事件記錄到 CloudTrail 做為管理事件。

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 或受管登入頁面提供資源的端點](#) 提出請求。API 請求具有 `AwsApiCall` `eventType`，端點請求具有 `AwsServiceEvent` `eventType`。

Amazon Cognito 會將下列請求記錄到您的受管登入服務，做為 CloudTrail 中的事件。

Hosted UI (classic) events

CloudTrail 中的託管 UI (傳統) 事件

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 <code>/oauth2/idpresponse</code> 端點提交 IdP 權杖。
SAML2Response_POST , Beta_SAML2Response_POST	使用者向您的 <code>/saml2/idpresponse</code> 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 。

作業	描述
Token_POST , Beta-Token_POST	使用者向您的 權杖端點 提交授權碼。
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。

作業	描述
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。
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。

Managed login events

CloudTrail 中的受管登入事件

作業	描述
login_POST	使用者提交登入資料到您的 登入端點 。
login_continue_POST	已經登入一次的使用者選擇再次登入。
forgotPassword_POST	使用者重設密碼。
selectChallenge_POST	使用者在提交使用者名稱或登入資料後回應身分驗證挑戰。
confirmUser_GET	使用者會在 確認或驗證電子郵件訊息 中開啟連結。
mfa_back_POST	使用者會在 MFA 提示後選擇返回按鈕。
mfa_options_POST	使用者選取 MFA 選項。
mfa_phone_register_POST	使用者提交要註冊為 MFA 因素的電話號碼。此操作會導致 Amazon Cognito 將 MFA 代碼傳送到其電話號碼。
mfa_phone_verify_POST	使用者提交傳送到其電話號碼的 MFA 代碼。
mfa_phone_resendCode_POST	使用者提交請求，以將 MFA 代碼重新傳送至其電話號碼。
mfa_totp_POST	使用者提交 TOTP MFA 代碼。

作業	描述
signup_POST	使用者提交資訊到您的/signup受管登入頁面。
signup_confirm_POST	使用者從電子郵件或簡訊提交確認碼。
verifyCode_POST	使用者提交一次性密碼 (OTP) 以進行無密碼身分驗證。
passkeys_add_POST	使用者提交註冊新通行金鑰登入資料的請求。
passkeys_add_GET	使用者導覽至可註冊通行金鑰的頁面。
login_passkey_POST	使用者使用通行金鑰登入。

Note

Amazon Cognito 會在 CloudTrail 日誌中，記錄特定使用者請求的 UserSub 而不是 UserName。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。

身分集區事件

資料事件

Amazon Cognito 會將下列 Amazon Cognito 身分事件記錄至 CloudTrail 作為資料事件。[資料事件](#)是 CloudTrail 預設不會記錄的大容量資料平面 API 操作。資料事件需支付額外的費用。

- [GetCredentialsForIdentity](#)
- [GetId](#)
- [GetOpenIdToken](#)
- [GetOpenIdTokenForDeveloperIdentity](#)
- [UnlinkIdentity](#)

若要為這些 API 操作產生 CloudTrail 日誌，您必須在您的線索中啟動資料事件，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 會將剩餘的 Amazon Cognito 身分池 API 操作記錄為管理事件。根據預設，CloudTrail 會記錄管理事件 API 操作。

如需 Amazon Cognito 記錄到 CloudTrail 的 Amazon Cognito 身分池 API 操作清單，請參閱 [Amazon Cognito 身分池 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 CloudTrail 的 Amazon Cognito Sync API 操作清單，請參閱 [Amazon Cognito Sync API 參考](#)。

使用 Amazon CloudWatch Logs Insights 分析 Amazon Cognito CloudTrail 事件

您可以使用 Amazon CloudWatch Logs Insights 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到 CloudWatch Logs 時，CloudTrail 只會傳送符合您追蹤設定的事件。

若要查詢或研究 Amazon Cognito CloudTrail 事件，請在 CloudTrail 主控台中，確定您在追蹤設定中選取管理事件選項，以便您可以監控對 AWS 資源執行的管理操作。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 事件範例

Amazon Cognito 會將 AWS CloudTrail 使用者身分驗證活動和管理活動的相關資訊記錄到。這同時適用於使用者集區和身分集區。例如，您可以在相同的線索中看到 GetId 和 UpdateIdentityPool 事件，或 UpdateAuthEventFeedback 和 SetRiskConfiguration 事件。您也會看到與使用者集區 API 中的操作不對應的託管 UI 活動的使用者集區日誌。本節提供一些您可能會看到的日誌範例。若要了解任何操作的 CloudTrail 事件結構描述，請產生該操作的請求，並檢閱它在線索中建立的事件。

追蹤能以日誌檔案的形式將事件傳遞至您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一或多個日誌專案。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。CloudTrail 日誌檔案並非依公有 API 呼叫追蹤記錄的堆疊排序，因此不會以任何特定順序出現。

主題

- [託管 UI 註冊的 CloudTrail 事件範例](#)
- [SAML 請求的範例 CloudTrail 事件](#)
- [向權杖端點發出請求的 CloudTrail 事件範例](#)
- [CreateIdentityPool 的 CloudTrail 事件範例](#)
- [GetCredentialsForIdentity 的 CloudTrail 事件範例](#)
- [GetId 的 CloudTrail 事件範例](#)
- [GetOpenIdToken 的 CloudTrail 事件範例](#)
- [GetOpenIdTokenForDeveloperIdentity 的 CloudTrail 事件範例](#)
- [UnlinkIdentity 的 CloudTrail 事件範例](#)

託管 UI 註冊的 CloudTrail 事件範例

以下範例 CloudTrail 事件示範使用者透過託管的 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.auth.us-west-2.amazoncognito.com",
    "userPoolId": "us-west-2_EXAMPLE"
  },
  "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.auth.us-west-2.amazoncognito.com",
"userPoolId": "us-west-2_EXAMPLE"
},
"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.auth.us-west-2.amazoncognito.com",
    "userPoolId": "us-west-2_EXAMPLE"
  },
  "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.auth.us-west-2.amazoncognito.com",
    "userPoolId": "us-west-2_EXAMPLE"
},
"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 請求的範例 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.auth.us-west-2.amazoncognito.com",
    "userPoolId": "us-west-2_EXAMPLE"
  },
  "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"
}

```

向權杖端點發出請求的 CloudTrail 事件範例

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.auth.us-west-2.amazoncognito.com",
    "userPoolId": "us-west-2_EXAMPLE"
},
"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.auth.us-west-2.amazoncognito.com",
    "userPoolId": "us-west-2_EXAMPLE"
  },
  "requestID": "4f871256-6825-488a-871b-c2d9f55caff2",
  "eventID": "473e5cbc-a5b3-4578-9ad6-3dfdc8a6d34",
  "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.auth.us-west-2.amazoncognito.com",
    "userPoolId": "us-west-2_EXAMPLE"
  },
  "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"
}
```

CreateIdentityPool 的 CloudTrail 事件範例

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": "0000000000000000"
    }
  },
  "responseElements": {
    "identityPoolName": "TestPool",
    "identityPoolId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
    "allowUnauthenticatedIdentities": true,
  }
}
```

```

    "supportedLoginProviders": {
      "graph.facebook.com": "0000000000000000"
    },
    "requestID": "15cc73a1-0780-460c-91e8-e12ef034e116",
    "eventID": "f1d47f93-c708-495b-bff1-cb935a6064b2",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
  }
}

```

GetCredentialsForIdentity 的 CloudTrail 事件範例

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_aaaaaaa":
"HIDDEN_DUE_TO_SECURITY_REASONS"
    },
    "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
  },
  "responseElements": {
    "credentials": {
      "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
      "sessionToken": "aAaAaAaAaAab1111111111EXAMPLE",
      "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"
  }

```

GetId 的 CloudTrail 事件範例

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_aaaaaaaa":
"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"
}

```

GetOpenIdToken 的 CloudTrail 事件範例

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_aaaaaaaa":
"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"
  }

```

GetOpenIdTokenForDeveloperIdentity 的 CloudTrail 事件範例

下列範例是請求 GetOpenIdTokenForDeveloperIdentity 動作的日誌項目。

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAIEXAMPLE:johns-AssumedRoleSession",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/johns-AssumedRoleSession",
    "accountId": "111122223333",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAIEXAMPLE",
        "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"
}

```

UnlinkIdentity 的 CloudTrail 事件範例

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_aaaaaaaa":
"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_aaaaaaaa"]
  },
}

```

```
{
  "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"
}
```

Amazon Cognito 的合規驗證

在多個合規計畫中，第三方稽核人員會評估 Amazon Cognito 的安全與 AWS 合規。這些計畫包括 SOC、PCI、FedRAMP、HIPAA 等等。

如需特定合規計畫範圍內 AWS 的服務清單，請參閱[AWS 合規計畫範圍內的服務合規](#)計畫的服務。如需一般資訊，請參閱 [AWS 合規計畫](#)。

您可以使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱[下載報告 in AWS Artifact](#)

您使用 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。
- [HIPAA 安全與合規架構白皮書](#) – 此白皮書說明公司如何使用 AWS 來建立符合 HIPAA 規範的應用程式。
- [AWS 合規資源](#) – 此工作手冊和指南集合可能適用於您的產業和位置。
- 使用《AWS Config 開發人員指南》中的[規則來評估資源](#) AWS Config；評估資源組態符合內部實務、產業準則和法規的程度。
- [AWS Security Hub](#) – AWS 此服務提供 內安全狀態的全方位檢視 AWS，可協助您檢查是否符合安全產業標準和最佳實務。

Amazon Cognito 的恢復能力

AWS 全球基礎設施是以 AWS 區域和可用區域為基礎建置。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 AWS 區域和可用區域的詳細資訊，請參閱 [AWS 全球基礎設施](#)。

主題

- [區域資料考量](#)

區域資料考量

Amazon Cognito 使用者集區各自建立在一個 AWS 區域中，而且只會將使用者設定檔資料存放在該區域中。使用者集區可以根據選用功能的設定方式，將使用者資料傳送至不同的 AWS 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 Simple Email Service (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 區域進行路由。
- 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

Amazon Pinpoint 適用於北美洲、歐洲、亞洲和大洋洲的數個 AWS 區域。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

Amazon Cognito 是受管服務，受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及如何 AWS 保護基礎設施的資訊，請參閱[AWS 雲端安全](#)。若要使用基礎設施安全的最佳實務設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的[基礎設施保護](#)。

您可以使用 AWS 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 處理基本安全任務，例如訪客作業系統 (OS) 和資料庫修補、防火牆組態和災難復原。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

若要新增許可給使用者、群組和角色，使用 AWS 受管政策比自行撰寫政策更容易。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。這些政策涵蓋常見的使用案例，並可在您的帳戶中使用 AWS。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 受管政策。您無法變更 AWS 受管政策中的許可。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 受管政策移除許可，因此政策更新不會破壞您現有的許可。

此外，AWS 支援跨多個服務之任務函數的受管政策。例如，ReadOnlyAccess AWS 受管政策提供所有 AWS 服務和資源的唯讀存取權。當服務啟動新功能時，會為新操作和資源 AWS 新增唯讀許可。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](#)。

AWS 授予 Amazon Cognito 存取權的受管 IAM 政策

- AmazonCognitoPowerUser - 存取及管理身分集區和使用者集區所有層面的許可。若要檢視此政策的許可，請參閱 [AmazonCognitoPowerUser](#)。
- AmazonCognitoReadOnly - 身分集區和使用者集區唯讀存取的許可。若要檢視此政策的許可，請參閱 [AmazonCognitoReadOnly](#)。
- AmazonCognitoDeveloperAuthenticatedIdentities - 讓您的身分驗證系統與 Amazon Cognito 整合的許可。若要檢視此政策的許可，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。

具有未驗證訪客存取權的身分集區會將額外的 AWS 受管政策作為[工作階段政策](#)套用至未驗證的使用者。此 AWS 受管政策沒有預期的管理用途。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](#)。

AWS Amazon Cognito 授予訪客使用者的受管 IAM 政策

- AmazonCognitoUnAuthedIdentitiesSessionPolicy - 結合內嵌工作階段政策，會限制 IAM 管理員可授予身分集區訪客使用者的許可。Amazon Cognito 會自動將此政策套用至訪客工作階段。如需詳細資訊，請參閱[訪客的 AWS 受管工作階段政策](#)。

受 AWS 管政策的 Amazon Cognito 更新

檢視自此服務開始追蹤 Amazon Cognito AWS 受管政策更新以來的詳細資訊。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
AmazonCognitoPowerUser –變更	Amazon Cognito 新增了動作，以允許使用適用於 Amazon Cognito 使用者集區管理進階使用者的 AWS End User Messaging SMS API 操作 DescribeAccountAttributes 。	2025 年 2 月 27 日
AmazonCognitoUnAuthenticatedIdentitiesSessionPolicy –變更	Amazon Cognito 新增了動作，以允許 AWS Key Management Service 在身分集區中使用未經驗證（訪客）的使用者。	2024 年 10 月 30 日
AmazonCognitoUnAuthenticatedIdentitiesSessionPolicy –變更	Amazon Cognito 新增了動作，以允許在身分集區中為未驗證（訪客）的使用者使用 Amazon Location Service。	2024 年 8 月 9 日
AmazonCognitoUnAuthenticatedIdentitiesSessionPolicy –新政策	針對身分集區中訪客使用者的權限縮小範圍新增 AWS 受管政策。	2023 年 7 月 14 日
AmazonCognitoPowerUser 和 AmazonCognitoReadOnly –變更	新增了許可，以允許進階使用者檢視和管理 AWS WAF Web ACLs 與 Amazon Cognito 使用者集區的關聯。 新增新許可，以允許唯讀使用者檢視 AWS WAF Web ACLs 與 Amazon Cognito 使用者集區的關聯。	2022 年 7 月 19 日

變更	描述	日期
AmazonCognitoPowerUser –變更	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 –變更	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 資源的 AWS 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，因此您可以將相同的標籤指派給來自不同服務的資源。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 帳單與成本管理 儀表板上啟用這些標籤。AWS 使用成本分配標籤來分類您的成本，並向您交付每月成本分配報告。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：_ . : / = + - @
- 鍵和值會區分大小寫
- 請不要使用 aws：做為鍵的字首；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](#)。如果出現提示，請輸入您的 AWS 登入資料。
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. 選擇設定選單，並找到標籤索引標籤。
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)。

若要標記身分集區，請導覽至身分集區功能表，然後選取或建立身分集區。在身分集區屬性索引標籤中，尋找標籤。選擇 Add tag (新增標籤)。

AWS CLI 範例

AWS CLI 提供的命令可協助您管理指派給 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 配額代表一個中一個 AWS 區域的最大請求量 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的請求量。AWS 一次只能在一個區域中授予配額增加請求。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](#)
- [管理 API 請求速率配額](#)
- [Amazon Cognito 使用者集區 API 操作類別和請求率配額](#)
- [Amazon Cognito 身分集區 \(聯合身分\) API 操作請求率配額](#)
- [資源數量和大小的配額](#)

了解 API 請求率配額

配額分類

Amazon Cognito 會強制執行 API 操作的最大請求率。如需 Amazon Cognito 提供 API 操作的詳細資訊，請參閱[使用者集區](#)和[身分集區](#)的 API 參考指南。對於使用者集區，這些操作會分組為常見使用案例的類別，例如 `UserAuthentication` 或 `UserCreation`。如需依類別的使用者集區 API 操作清單，請參閱 [Amazon Cognito 使用者集區 API 操作類別和請求率配額](#)。

在 [Service Quotas 主控台](#) 中，您可以依類別使用者集區和身分集區追蹤配額用量。如果 Amazon Cognito 使用者集區的請求率或超過配額，您可以購買額外的容量。您可以依類別追蹤使用者集區配額用量，並在 [Service Quotas 主控台](#) 中購買配額增加。

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請求速率配額，會 AWS 測量 AWS 帳戶 一個區域中來自 中所有使用者集區或身分集區的所有請求的彙總速率。

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。

UserAuthentication 類別包含 Amazon Cognito 使用者集區 API 中的四個操作：AdminInitiateAuth、AdminRespondToAuthChallenge、InitiateAuth 和 RespondToAuthChallenge。此外，託管 UI 中的使用者身分驗證有助於此配額。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。此提高的配額可因應應用程式中設定的多個身分驗證挑戰。此配額足以涵蓋大多數使用案例。在您的應用程式對身分驗證挑戰做出最多三個回應之後，額外的請求會計入 UserAuthentication 類別配額。[多重要素驗證 \(MFA\)](#)、[裝置身分驗證](#) 和 [自訂身分驗證](#) 都是您可能在使用者集區中設計的挑戰提示範例。

例如，如果您的 UserAuthentication 類別配額為 80 RPS，您可以呼叫 RespondToAuthChallenge 或以高達 240 RPS ($3 * 80 \text{ RPS}$) AdminRespondToAuthChallenge 的速率呼叫 或 。如果您的使用者集區每次身分驗證提示四輪挑戰，且每秒有 70 名使用者登入，則總計 RespondToAuthChallenge 為 280 RPS ($70 * 4$)，高於配額 40 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。由於此值超過設定為 80 RPS 乘以 30 RPS 的類別配額，Amazon Cognito 會調節您應用程式的請求。

每月作用中使用者

當 Amazon Cognito 計算使用者集區帳單時，它會向您收取每月作用中使用者 (MAU) 的費率。在規劃增加配額請求時，請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。當您使用 SAML 或 OIDC 聯合[將聯合身分使用者連結至本機使用者](#)時，無論使用者是直接登入或透過聯合身分登入 Enterprise MAU，本機使用者都會計入企業目錄 MAU 或 。如需詳細資訊，請參閱 [Amazon Cognito 定價](#)。

- 使用者的註冊或管理建立。[使用者 CSV 匯入](#)不會計入您的 MAU 計數。

- 使用者帳戶確認或屬性驗證。
- 登入和挑戰回應。您使用目前登入使用者的存取權杖授權的操作不會計入您的 MAU 計數；不過，由於登入會產生存取權杖，因此這些操作會指出相關聯的使用者是 MAU。
- 登出和字符撤銷。
- 密碼自助式重設，並以管理員身分設定使用者密碼。將使用者密碼重設為管理員 ([AdminResetUserPassword](#)) 不會計入您的 MAU 計數。
- 變更使用者屬性或群組成員資格。
- 以管理員身分查詢使用者的詳細屬性。

Note

類別 以管理員身分查詢使用者的詳細屬性包括 API 操作 [AdminGetUser](#)，但不包括 [ListUsers](#)。大型使用者集區中的詳細user-by-user使用者查詢可能會對您的 AWS 帳單產生重大影響。若要避免額外費用，請使用 收集使用者資料，ListUsers或將使用者資訊存放在外部資料庫中。

任何作用中的使用者，或是任何未在日曆月內作用中的使用者，都不需要支付額外的工作階段費用。在 Lite、Essentials 和 Plus 的可用選項之間變更使用者集區功能計畫的月份中，該月的帳單是根據每個層級中的每月作用中使用者 (MAUs) 總和計算，而當使用者處於作用中狀態時，每個 MAU 都會指派給最高價格的指派層級。例如：

1. 在月初，您的使用者集區已加入 Plus 功能計畫。
2. 使用者 A 會在當月第一天登入。
3. 使用者 B 會在當月的第一天和最後一天登入。
4. 在每月的第十天，您將功能計畫切換為 Essentials。
5. 使用者 C 在當月最後一天登入。

在此案例中，使用者 A 和使用者 B 是 Plus MAUs而使用者 C 是 Essentials MAU。

Lite MAU

當使用者集區在 Lite 功能計畫中時，每月至少作用中一次的使用者，以及當使用者集區在 Essentials 或 Plus 計畫中時從未作用中的使用者。

基本 MAU

當使用者集區在 Essentials 功能計劃中時，每月至少作用中一次的使用者，以及當使用者集區在 Plus 計劃中時從未作用中的使用者。

Plus MAU

當使用者集區在 Plus 計劃中時，每個月至少作用中一次的使用者。

如需詳細資訊，請參閱[使用者集區功能計劃](#)。

管理 API 請求速率配額

確認配額需求

Important

如果您增加 UserAuthentication、UserCreation 或 等類別的 Amazon Cognito 配額 AccountRecovery，您可能需要增加其他的配額 AWS 服務。例如，如果這些服務的請求速率配額不足，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。

如果您考慮八小時期間內使用者登入頻率的差異，假設峰值與平均值的比例為 3:1，則所需 UserAuthentication 配額為 200 個 RPS。

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。

最佳化配額限制的請求率

由於提高 API 速率限制會增加 AWS 帳單成本，因此在請求提高配額之前，請考慮調整用量模型。以下是最佳化請求率之應用程式架構的一些範例。

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[要建置的工具 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。

在用戶端驗證 JSON Web 字符 JWTs)

應用程式必須先驗證 JWT 權杖，才能信任該權杖。您可以在用戶端驗證字符的簽章和有效性，而無需向使用者集區傳送 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 JWTs

重複使用存取權杖，直到過期為止。如需 API Gateway 中具有字符快取的範例架構，請參閱[管理使用者集區字符過期和快取](#)。而不是產生 API 請求來查詢使用者資訊、快取 ID 字符直到過期，以及從快取讀取使用者屬性。

如需在 中使用 API 請求率的詳細資訊 AWS，請參閱[管理和監控工作負載中的 API 限流](#)。如需最佳化 Amazon Cognito 操作以為您的 AWS 帳單增加成本的相關資訊，請參閱[管理成本](#)。

追蹤配額使用量

Amazon Cognito 在 Amazon CloudWatch 中產生 CallCount 和 ThrottleCount 指標，供每個 API 操作類別在帳戶層級套用。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 使用量指標](#)。

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 CloudWatch 指標追蹤使用情況

您可以使用 CloudWatch 追蹤和收集 Amazon Cognito 使用者集區使用率指標。CloudWatch 儀表板會顯示您使用 AWS 服務的每個的指標。您可以使用 CloudWatch 建立指標警示，以傳送通知或變更您正在監控的特定資源。如需 CloudWatch 指標的詳細資訊，請參閱[追蹤 CloudWatch 使用量指標](#)。

透過 Service Quotas 指標追蹤使用率

Amazon Cognito 使用者集區與 Service Quotas 整合，Service Quotas 是一種主控台界面，可顯示並管理您的服務配額用量。在 Service Quotas 主控台中，您可以查詢特定配額的值、檢視監控資訊、要求增加配額，或設定 CloudWatch 警示。在您的帳戶處於作用中狀態一段時間後，您可以檢視資源使用率的圖表。

[Amazon Cognito 使用者集區](#)和 [Amazon Cognito](#)身分集區的 Service Quotas 主控台 中的套用帳戶層級配額值欄會顯示您目前的配額。Service Quotas 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數 (RPS) 配額會顯示其目前的用量。Service Quotas 主控台也可以導覽至 CloudWatch 指標，以更仔細地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 MAUs)

使用者集區中的每月作用中使用者 (MAUs) 數量會為您的規劃增加請求率配額提供重要資料。您可以比較 API 請求率與您在指定期間內作用中的使用者數量。有了這些知識，您可以計算應用程式的作用中使用者增加將如何影響用量模型中的配額。例如，假設您在美國西部（奧勒岡）的合併應用程式一個月產生 200 萬名作用中使用者，而您的UserAuthentication類別偶爾會收到調節錯誤，預設配額為每秒 120 個請求 (RPS)。在上個月，在您成功的廣告行銷活動之前，您有 100 萬MAUs，而且您的應用程式從未超過 80 個 RPS。如果您預期由於新的電視台而出現類似的峰值，您可以購買額外的 40 RPS，以容納調整後配額為 160 RPS 的下一個百萬使用者。

檢閱您的 MAUs

存取 [AWS Billing 主控台](#)並檢閱最近的帳單。在依服務收費的情況下，您可以在 Cognito 上進行篩選，以檢視該計費期間的 MAUs明細。

請求提高配額

Amazon Cognito 具有您每秒可在每個使用者集區和身分集區中執行的最大操作數量配額 AWS 區域。您可以購買增加可調整的 Amazon Cognito 使用者集區 API 請求率配額。檢查您目前的配額，並從 Service Quotas 主控台或透過 Service Quotas API 操作 `ListAWSDefaultServiceQuotas` 和 `購買增加RequestServiceQuotaIncrease`。

- 若要使用 Service Quotas 主控台購買增加配額，請參閱 [Service Quotas 使用者指南](#) 中的 [請求增加 API 配額](#)。
- AWS 目標是在 10 天內完成配額增加請求。不過，數個考量可能會導致請求處理時間超過 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。您必須購買增加的配額容量。如需提高配額定價，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) 也會套用至中的所有使用者集區 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。如果您有不到一百萬 MAUs，且 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 [最佳化配額限制的請求率](#)。

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、InitiateAuth、RespondToAuthChallenge	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 • RespondToAuthChallenge¹ • AdminInitiateAuth • AdminRespondToAuthChallenge¹	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
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 • RespondToAuthChallenge¹ • AdminRespondToAuthChallenge¹ • 受管登入密碼重設	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 • AdminListGroupsWithUser • 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 • AdminSetUserMFAPreference • AdminSetUserSettings • AdminUpdateDeviceStatus • UpdateDeviceStatus • UpdateAuthEventFeedback • ConfirmDevice • SetUserMFAPreference • 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。	產生憑證以用於授權機器對機器請求的操作	150	否

¹ A RespondToAuthChallenge或的AdminRespondToAuthChallenge回應ChallengeNameNEW_PASSWORD_REQUIRED會計入 UserAccountRecovery類別。所有其他挑戰回應都會計入 UserAuthentication類別。

² 登入期間的每個託管 UI 操作都會對配額貢獻一個請求。例如，登入並提供 MFA 程式碼的使用者會提供 2 個請求。授權碼授予中的字符兌換會受到與 UserAuthentication 類別中配額相同的額外配額分配。

³ 此類別中的任何個別操作都有限制，可防止單一使用者集區以高於 5 RPS 的速率呼叫操作。

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 憑證。	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	將索引標籤套用於身分池。	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 AWS 區域中身分集區的最低請求率配額 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，例如每個區域的使用者集區，會套用至每個區域中的 Amazon Cognito 資源 AWS 區域。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。
每個使用者集區的受管登入品牌樣式	10	否	N/A
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
每天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

資源	配額	可調整	最大配額
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
身分提供者名稱中的字元數	32	否	N/A
SAML 回應中的字元	100,000	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
每個使用者的 Passkey/WebAuthn 驗證器	20	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

資源	配額	可調整	最大配額
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。一個交易中現有和新增的宣告加上存取和身分字元範圍的數量，必須加總到小於或等於此配額的數字。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。此限制會在 UTC 每天重設為 0900。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元	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

資源	配額
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
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
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
單一系列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。

資源	配額	可調整	最大配額
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 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
AWS WAF 受管登入中的 Web ACLs。	您現在可以將 AWS WAF Web ACL 規則套用至具有受管登入品牌版本的使用者集區應用程式用戶端。	2025 年 6 月 24 日
已更新 Lambda 觸發程序範例。	更新自訂電子郵件和簡訊寄件者 Lambda 觸發條件的範例函數，以與 Node.js 22.x 相容。此範例現在也可供測試使用。	2025 年 5 月 19 日
新prompt參數。	您現在可以使用 prompt 參數更好地控制現有受管登入工作階段的重新驗證。您也可以將此參數的值傳遞給第三方供應商。	2025 年 5 月 15 日
M2M 請求的用戶端中繼資料。	您現在可以在用戶端登入資料或machine-to-machine(M2M) 請求中傳遞用戶端中繼資料。Amazon Cognito 會將 M2M 用戶端中繼資料傳遞至權杖產生前 Lambda 觸發程序。	2025 年 4 月 29 日
重新整理權杖輪換。	您現在可以取得新的重新整理字符，並在重新整理字符請求中使原始字符失效。	2025 年 4 月 22 日

[Amazon Cognito 現已在亞太區域（馬來西亞）提供 AWS 區域。](#)

您現在可以在亞太區域（馬來西亞）區域中建立 Amazon Cognito 資源。

2025 年 3 月 7 日

[機器身分的存取權杖自訂。](#)

產生權杖前 Lambda 觸發程序現在有一個第 3 版事件，可修改用於 machine-to-machine (M2M) 授權的用戶端憑證授予中的存取權杖宣告和範圍。

2025 年 3 月 3 日

[更新 AmazonCognitoPowerUser AWS 受管政策的相關資訊。](#)

在 Amazon Cognito 使用者集區進階使用者的 AWS 受管政策中新增 AWS End User Messaging SMS 操作。

2025 年 2 月 27 日

[更新 OpenID Connect \(OIDC\) 整合的概觀。](#)

新增圖表，說明 Amazon Cognito 如何向 OIDC 身分提供者進行身分驗證。

2025 年 2 月 25 日

[新增 MFA 邏輯的相關資訊。](#)

新增圖表，說明 Amazon Cognito 如何在執行時間將您的使用者集區多重要素驗證 (MFA) 設定套用至使用者。

2025 年 2 月 25 日

[新增 Amazon Cognito 使用者集區安全最佳實務。](#)

新增了有關保護秘密以及遵循使用者集區組態中安全最佳實務的頁面。

2025 年 2 月 25 日

[使用者集區的入門資源更新。](#)

Amazon Cognito 使用者集區的入門體驗具有新的主控台設計和應用程式選項。

2024 年 11 月 21 日

[具有功能計劃的新定價模型。](#)

更新了使用者集區的帳單模型。進階安全功能現在是威脅防護。進階安全功能授權中的元件現在已包含在 Essentials 和 Plus 功能計劃中。

2024 年 11 月 21 日

新的受管登入功能。	已啟動受管登入，即託管 UI 的更新。	2024 年 11 月 21 日
新的身分驗證方法和新的身分驗證流程。	您現在可以使用通行金鑰和一次性密碼登入 Amazon Cognito 使用者集區。	2024 年 11 月 21 日
已更新的相關資訊AmazonCognitoUnAuthedIdentitiesSessionPolicy 。	將 AWS 受管政策中的 AWS Key Management Service 操作從內嵌政策縮減未驗證身分的範圍至 AWS 受管政策。	2024 年 11 月 1 日
新增login_hint 參數。	您現在可以將使用者名稱提示新增至託管 UI、OIDC IdPs 和 Google IdPs 的授權請求。	2024 年 10 月 3 日
電子郵件 MFA 的新進階安全功能。	您現在可以透過具有進階安全功能的電子郵件訊息傳送多重要素驗證 (MFA) 代碼。	2024 年 9 月 12 日
新的內容和頁面變更。	修改標題、移除不需要的內容、新增以案例為基礎的簡介、將使用者集區 OIDC 和託管 UI 端點參考移至使用者集區區段。	2024 年 9 月 9 日
已更新的相關資訊AmazonCognitoUnAuthedIdentitiesSessionPolicy 。	身分集區中未驗證身分縮小範圍的 AWS 受管政策現在允許 Amazon Location Service。	2024 年 8 月 9 日
使用 Lambda 觸發和增強型威脅偵測進行自訂身分驗證的新威脅預防。	您現在可以使用威脅防護分析自訂身分驗證登入，並套用適應性身分驗證回應。威脅防護現在也會分析登入流量，找出嘗試之間不可能的地理距離。	2024 年 8 月 8 日

新的進階安全功能，用於防止密碼重複使用和使用者的活動日誌匯出。	您現在可以匯出使用者活動日誌，並使用 Amazon Cognito 使用者集區中的進階安全功能設定密碼歷史記錄政策。	2024 年 8 月 6 日
Amazon Cognito 現已在加拿大西部（卡加利）和亞太區域（香港）提供 AWS 區域。	您現在可以在加拿大西部（卡加利）和亞太區域（香港）區域中建立 Amazon Cognito 資源。	2024 年 7 月 9 日
改善應用程式行為的描述，以實現進階安全性	更新進階安全性自適應身分驗證的裝置內容資料相關資訊。	2024 年 6 月 10 日
新增對權杖前 Lambda 觸發程序中複雜物件的支援	您現在可以將陣列和 JSON 物件新增至 ID 和存取字宣告。	2024 年 5 月 30 日
更新 Verified Permissions 和 Amazon Cognito 的相關資訊。	Amazon Verified Permissions 現在與 Amazon Cognito 更直接整合。	2024 年 5 月 15 日
多區域 Amazon SES 驗證的身分。	在一些 AWS 區域沒有 Amazon SES 的中，Amazon Cognito 使用者集區會在兩個遠端區域之間負載平衡電子郵件。	2024 年 5 月 10 日
新增 M2M 授權和管理成本的相關資訊。	了解如何搭配 Amazon Cognito 使用者集區使用 machine-to-machine(M2M) 使用案例的用戶端憑證授權。	2024 年 5 月 9 日
Amazon Cognito 現已在歐洲（西班牙）和亞太區域（海德拉巴）提供 AWS 區域。	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）提供 AWS 區域。	您現在可以在亞太區域（墨爾本）區域中建立 Amazon Cognito 資源。	2024 年 4 月 4 日

在 Flutter for Amazon Cognito 使用者集區中新增範例 Android 應用程式。	您可以從 GitHub 上的 Flutter 應用程式範例建置 Amazon Cognito 的入門行動應用程式。	2024 年 4 月 4 日
新的入門內容	擴展內容以開始使用、常見案例、多租戶最佳實務，以及在登入後存取資源。	2024 年 4 月 1 日
Amazon Cognito 現已在歐洲（蘇黎世）提供 AWS 區域。	您現在可以在歐洲（蘇黎世）區域中建立 Amazon Cognito 資源。	2024 年 3 月 14 日
Amazon Cognito 現已在中東（阿拉伯聯合大公國）提供 AWS 區域。	您現在可以在中東（阿拉伯聯合大公國）區域中建立 Amazon Cognito 資源。	2024 年 3 月 8 日
新的 SAML 功能和改善的內容。	您現在可以簽署 SAML 請求、加密 SAML 回應，以及設定 IdP 起始的 SAML SSO。	2024 年 2 月 1 日
配額增加可用。	您現在可以購買 Amazon Cognito 請求率配額的額外容量。	2024 年 1 月 25 日
Amazon Cognito 身分集區支援 Service Quotas 中的請求率。	您現在可以監控 Amazon Cognito 身分集區的每秒請求 (RPS) 配額，並在 Service Quotas 主控台中增加請求。	2023 年 12 月 19 日
新增了用於自訂存取權杖內容的新功能。	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 範圍的內容。	對 使用應用程式用戶端的應用程式特定設定 和 範圍、M2M 和具有資源伺服器的 APIs 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

改善有關裝置和裝置身分驗證的內容。	有關使用裝置金鑰和裝置 SRP 身分驗證的新內容。	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
取消強調 LOGIN 端點的直接存取。	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）提供 AWS 區域。	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
已引進有關使用 Amazon Verified Permissions 授權 Amazon Cognito 的資訊。	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 CloudWatch Logs 的新功能。	現在您可以將電子郵件和簡訊傳遞錯誤記錄到 CloudWatch 日誌群組。	2023 年 8 月 1 日
更新身分集區訪客使用者的 AWS 受管政策相關資訊。	身分集區訪客使用者的許可縮小範圍現在包含內嵌工作階段政策和 AWS 受管工作階段政策。	2023 年 5 月 16 日
Amazon Cognito 身分集區的內容改善和新的主控台指示。	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
服務首頁和使用集區首頁的新增和改進。	已更新 Amazon Cognito 和 使用者集區 的概觀頁面。	2023 年 5 月 16 日

使用者集區字符文件的一般改進。	更新範例權杖，並新增驗證權杖的新資訊。	2023 年 2 月 16 日
您現在可以在其中記錄 Amazon Cognito 身分集區資料事件 AWS CloudTrail。	CloudTrail 支援在記錄資料事件的線索中選擇 Amazon Cognito 身分集區大容量 API 操作。	2023 年 2 月 15 日
已更新 Lambda 觸發器範例和描述。	Lambda 觸發程序範例已更新至 JavaScript 版本 3。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 受管政策套用至未驗證的工作階段。	使用增強流程進行身分驗證的身分集區使用者現在已將額外的 AWS 受管政策套用至其工作階段。	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
新增使用 Amazon Cognito 使用者集區進行 API 模型和身分驗證的相關資訊。	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) 提供 AWS 區域。	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保護的資訊。	當您使用 建立新的使用者集區時 AWS Management Console，它現在預設會受到保護，不會遭到刪除。	2022 年 10 月 20 日

新增託管 UI 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託管 UI。	2022 年 9 月 8 日
新增 AWS WAF 和 Amazon Cognito 的相關資訊。	您現在可以將 AWS WAF Web ACL 與 Amazon Cognito 使用者集區建立關聯。	2022 年 8 月 3 日
新增更多範例 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
新增兩步驟屬性驗證的相關資訊。	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
已更新聯合文件。新的 IP 地址傳播功能。	已更新用於設定使用者集區社交 IdP 的演練。已新增有關聯合身分使用者描述檔與屬性映射的資訊。新增有關裝置指紋的新資訊，以實現進階安全性。	2022 年 5 月 31 日
在沒有與託管 UI 互動的情況下登入聯合身分使用者	新增了有關如何將應用程式加入書籤的新頁面，以便 Amazon Cognito 無提示地將使用者導向聯合登入。	2022 年 5 月 29 日
Amazon Cognito 使用者集區的區域內簡訊和電子郵件訊息	您現在可以將 Amazon Simple Notification Service 用於簡訊，並將 Amazon Simple Email Service 用於與您使用者集區 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	新增並釐清資源和請求率配額。	2022 年 1 月 10 日

新的 Amazon Cognito 使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 RevokeToken 操作，為使用者 撤銷重新整理權杖 。	2021 年 6 月 10 日
多租戶最佳實務	新增多租戶應用程式的最佳實務。	2021 年 3 月 4 日
存取控制的屬性	Amazon Cognito Identity Pools 提供存取控制 (AFAC) 的屬性，讓客戶授予使用者對 AWS 資源的存取權。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 觸發和自訂電子郵件寄件者 Lambda 觸發	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 權杖更新	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量、要求提高配額，並建立 CloudWatch 警示來監控您的配額使用量。在此變更中，已更新 Amazon Cognito 使用者集區的可用 CloudWatch 指標一節以反映新資訊。新章節名稱為：在 CloudWatch 和 Service Quotas 中追蹤配額和使用量	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
US AWS GovCloud 支援 Amazon Cognito	AWS GovCloud (US) 區域現在支援 Amazon Cognito。	2020 年 5 月 13 日
Amazon Cognito Pinpoint 文件更新	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
新的 Amazon Cognito 專用安全章節	安全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 身分集區現在支援使用 Apple 登入	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 Facebook API 版本控制	新增 Facebook API 版本選擇。	2020 年 4 月 3 日

使用者名稱案例不敏感更新	新增了在建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
有關的新資訊 AWS Amplify	新增使用 AWS Amplify SDKs 和程式庫將 Amazon Cognito 與您的 Web 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
使用者集區觸發條件的新屬性	Amazon Cognito 現在會在傳遞給大多數使用者集區觸發 AWS Lambda 器函數的事件資訊中包含 <code>clientMetadata</code> 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
已更新限制	ListUsers API 動作的調節限制已更新。	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
Amazon Cognito 使用者集區的 Amazon SES 電子郵件設定 Amazon Cognito	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
標記支援	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
已更新限制	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
測試電子郵件的文件更新	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 進階安全性	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 UI 的自訂網域	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 使用者集區 OIDC 身分提供者	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 遷移觸發程序	新增有關 Lambda Migration 觸發功能的頁面	2018 年 4 月 8 日
Amazon Cognito 開發人員指南更新	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 進階安全 Beta 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	更新範例，示範如何搭配使用者集區使用 AWS Lambda 觸發。	2016 年 11 月 27 日
文件更新	已更新 iOS 程式碼範例。	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	新增與 的整合 AWS CloudTrail。	2016 年 2 月 18 日

事件與 Lambda 的整合	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 支援	可支援 OpenID Connect 供應商。	2014 年 11 月 23 日
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。