Amazon Cognito

开发人员指南



Amazon Cognito: 开发人员指南

Copyright © 2023 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能并非如此。

Table of Contents

什么是 Amazon Cognito？ 1
用户池 1
身份池 2
Amazon Cognito 的功能 3
用户池 3
身份池 4
Amazon Cognito 用户群体和身份池的比较 5
Amazon Cognito 入门 7
区域可用性 8
Amazon Cognito 的定价 8
使用 Amazon Cognito 控制台 8
使用 AWS SDK 8
Amazon Cognito 入门 10
注册一个 AWS 账户 10
创建管理用户 11
Amazon Cognito 常见场景 12
使用用户池进行身份验证 12
访问您的服务器端资源 13
使用 API Gateway 和 Lambda 来访问资源 13
使用用户池和身份池访问 AWS 服务 14
借助第三方进行身份验证并使用身份池访问 AWS 服务 14
使用 Amazon Cognito 访问 AWS AppSync 资源 15
教程 16
创建用户池 16
相关资源 16
创建身份池 16
相关资源 18
清除 AWS 资源 18
与应用程序集成 20
使用 AWS Amplify 框架进行 Amazon Cognito 身份验证 20
使用 amazon-cognito-identity-js 的身份验证 21
使用 AWS 开发工具包进行身份验证 21
使用 Amazon Verified Permissions 进行授权 21
Amazon Cognito 用户的示例策略 23
多租户应用程序最佳实践 25
基于用户池的多租户 25
基于应用程序客户端的多租户 25
基于组的多租户 26
基于自定义属性的多租户 26
多租户安全建议 26
Amazon Cognito 用户池 27
功能 27
注册 28
登录 28
托管 UI 29
安全性 29
自定义客户体验 29
监控和分析 29
Amazon Cognito 身份池集成 30
用户池入门 30
第 1 步 创建用户池 30
第 2 步 添加应用程序客户端并设置托管 UI 31
第 3 步 向用户池添加社交登录（可选） 33
第 4 步 将使用 SAML 身份提供商的登录添加到用户池（可选） 37

后续步骤 39
身份验证 39
用户池身份验证流程 41
使用 API 和端点 47
用户群体 API 身份验证 48
更新用户群体 53
使用 Amazon Cognito API 或 AWS CLI 更新用户群体 54
使用托管 UI 54
使用 AWS Amplify 设置托管 UI 55
使用 Amazon Cognito 控制台设置托管 UI 55
配置应用程序客户端 58
配置域 61
自定义内置网页 67
如何使用托管 UI 71
定义资源服务器 82
通过第三方添加登录 85
联合登录在 Amazon Cognito 用户群体中的工作方式 85
应用程序作为 Amazon Cognito 的服务提供者的责任 85
关于 Amazon Cognito 用户群体第三方登录需要了解的事项 86
添加社交身份提供商 86
添加 SAML 提供商 92
添加 OIDC 提供商 102
指定属性映射 107
将联合用户与现有用户配置文件关联 109
使用 Lambda 触发器 111
重要注意事项 112
添加用户池触发器 113
用户池 Lambda 触发器事件 114
用户池 Lambda 触发器通用参数 114
按事件分类的 Lambda 触发器源 115
按函数分类的 Lambda 触发器源 118
注册前 Lambda 触发器 120
确认后 Lambda 触发器 125
身份验证前 Lambda 触发器 128
身份验证后 Lambda 触发器 131
质询 Lambda 触发器 134
令牌生成前 Lambda 触发器 143
迁移用户 Lambda 触发器 147
自定义消息 Lambda 触发器 151
自定义发件人 Lambda 触发器 155
使用 Amazon Pinpoint 分析 164
查找 Amazon Cognito 和 Amazon Pinpoint 区域映射 165
将您的应用程序与 Amazon Pinpoint 集成 167
管理用户 167
注册并确认用户账户 168
以管理员身份创建用户 175
向用户池添加组 178
管理和搜索用户 180
恢复用户账户 183
将用户导入一个用户池 183
电子邮件设置 194
默认电子邮件功能 194
Amazon SES 电子邮件配置 194
配置电子邮件账户 196
短信设置 199
首次在 Amazon Cognito 用户池中设置 SMS 消息 199
使用令牌 203

使用 ID 令牌 204
使用访问令牌 206
使用刷新令牌 208
撤消令牌 209
验证 JSON Web 令牌 211
缓存令牌 214
在登录后访问资源 215
访问服务器端资源 13
使用 API Gateway 和 Lambda 访问资源 216
使用身份池访问 AWS 资源 217
使用安全功能 219
添加 MFA 220
添加高级安全 227
AWS WAF Web ACL 236
区分大小写 238
管理用户泄露 239
用户池控制台参考 243
用户和组 243
属性 243
密码要求 250
允许用户注册 250
删除保护 251
电子邮件或电话验证 251
消息自定义 253
标签 257
设备 258
应用程序客户端 258
触发 260
分析 260
域名 261
UI 自定义 262
身份提供商 262
属性映射 266
旧版控制台 267
Amazon Cognito 身份池 288
身份池入门 289
在 Amazon Cognito 中创建一个身份池 289
设置 SDK 291
集成身份提供商 291
获取凭证 291
使用身份池 291
用户 IAM 角色 293
经过身份验证和未经身份验证的身份 293
激活或停用访客访问权限 293
更改与身份类型关联的角色 294
编辑身份提供者 295
删除身份池 296
从身份池删除身份 296
将 Amazon Cognito Sync 与身份池一起使用 297
身份池概念 299
身份池身份验证流程 300
IAM 角色 305
角色信任和权限 312
将属性用于访问控制 313
使用属性对 Amazon Cognito 身份池进行访问控制 314
示例：将属性用于访问控制策略 315
关闭访问控制属性 316

默认提供商映射 317
使用基于角色的访问控制 318
为角色映射创建角色 318
授予传递角色权限 318
使用令牌向用户分配角色 319
使用基于规则的映射向用户分配角色 319
基于规则的映射中使用的令牌声明 320
基于角色的访问控制的最佳实践 321
获取凭证 322
访问 AWS 服务 328
身份池外部身份提供商 330
Facebook 330
Login with Amazon 336
Google 339
Sign in with Apple 348
Open ID Connect 提供商 352
SAML 身份提供商 355
经开发人员验证的身份 357
了解身份验证流程 357
定义开发人员提供商名称并将其与身份池关联 357
实施身份提供商 358
更新登录映射（仅限 Android 和 iOS） 363
获取令牌（服务器端） 364
连接到现有社交身份 365
支持在提供商之间转换 365
切换身份 368
Android 368
iOS – objective-C 368
iOS – swift 368
JavaScript 369
Unity 369
Xamarin 370
Amazon Cognito Sync 371
Amazon Cognito Sync 入门 371
设置 Amazon Cognito 中的身份池 371
存储和同步数据 371
同步数据 372
初始化 Amazon Cognito Sync 客户端 372
了解数据集 373
在数据集中读取并写入数据 375
使用同步存储同步本地数据 376
处理回调 378
Android 378
iOS - Objective-C 380
iOS - Swift 382
JavaScript 384
Unity 386
Xamarin 388
推送同步 389
创建 Amazon Simple Notification Service（Amazon SNS）应用程序 390
在 Amazon Cognito 控制台中启用推送同步 390
在您的应用程序中使用推送同步：Android 391
在您的应用程序中使用推送同步：iOS – Objective-C 392
在您的应用程序中使用推送同步：iOS – Swift 394
Amazon Cognito Streams 396
Amazon Cognito Events 397
代码示例 401

Amazon Cognito 身份示例 401
操作 402
跨服务示例 405
Amazon Cognito 身份提供者示例 407
操作 407
场景 454
Amazon Cognito 同步示例 488
操作 488
安全性 490
数据保护 490
数据加密 491
身份和访问管理 491
受众 491
使用身份进行身份验证 492
使用策略管理访问 493
Amazon Cognito 如何与 IAM 配合使用 494
基于身份的策略示例 500
问题排查 503
使用服务相关角色 504
日志记录和监控 507
跟踪 CloudWatch 和 Service Quotas 中的配额和使用情况 507
使用 AWS CloudTrail 记录 Amazon Cognito API 调用 513
合规性验证 530
故障恢复能力 530
区域数据注意事项 530
基础设施安全性 531
配置和漏洞分析 531
AWS 托管策略 531
策略更新 532
为资源添加标签 534
支持的资源 534
标签限制 534
使用控制台管理标签 534
AWS CLI 示例 535
分配标签 535
查看标签 536
删除标签 536
在创建资源时应用标签 537
API 操作 537
适用于用户池标签的 API 操作 537
适用于身份池标签的 API 操作 537
配额 539
了解 API 请求速率配额 539
配额分类 539
使用特殊请求速率处理 Amazon Cognito 用户池 API 操作 539
Monthly active users (月活跃用户数) 540
管理 API 请求速率配额 540
确定配额要求 540
优化配额 541
跟踪配额使用量 541
请求增加配额 541
用户群体请求速率配额 542
身份池请求速率配额 547
资源数量和大小的配额 547
API 参考 551
用户群体端点参考 551
托管 UI 端点参考 552

联合身份验证端点参考 556
托管 UI 和联合身份验证错误响应 568
用户池 API 参考 569
身份池 API 参考 570
Cognito 同步 API 参考 570
文档历史记录 571

什么是 Amazon Cognito？

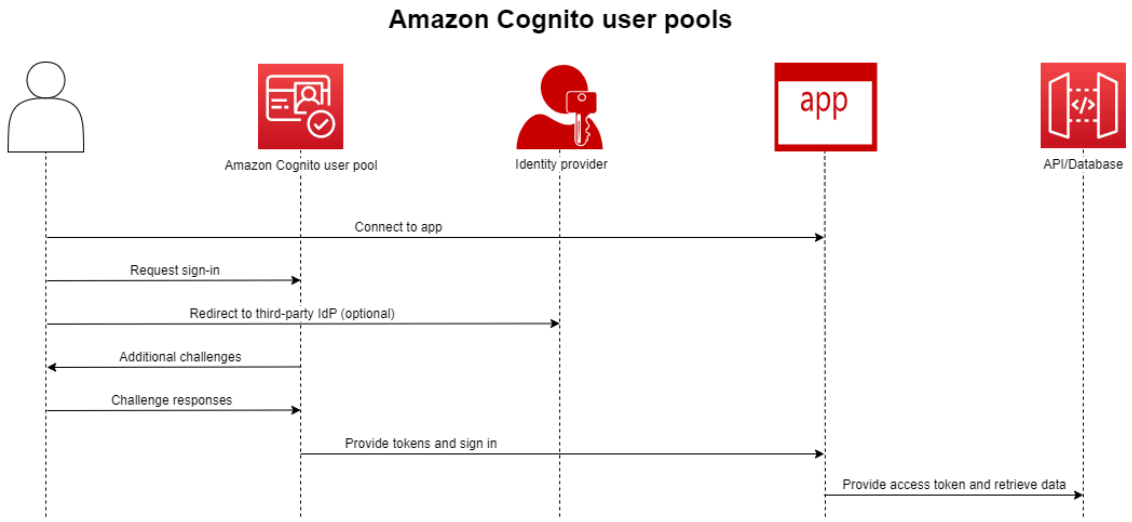
Amazon Cognito 是 Web 和移动应用程序的身份平台。它是用户目录、身份验证服务器以及 OAuth 2.0 访问令牌和 AWS 凭证的授权服务。使用 Amazon Cognito，您可以对内置用户目录、企业目录以及 Google 和 Facebook 等使用者身份提供者中的用户进行身份验证和授权。

主题

- 用户池 (p. 1)
- 身份池 (p. 2)
- Amazon Cognito 的功能 (p. 3)
- Amazon Cognito 用户群体和身份池的比较 (p. 5)
- Amazon Cognito 入门 (p. 7)
- 区域可用性 (p. 8)
- Amazon Cognito 的定价 (p. 8)
- 使用 Amazon Cognito 控制台 (p. 8)
- 将此服务与 AWS SDK 结合使用 (p. 8)

随后的两个组件构成了 Amazon Cognito。它们根据用户的访问需求独立或协同运行。

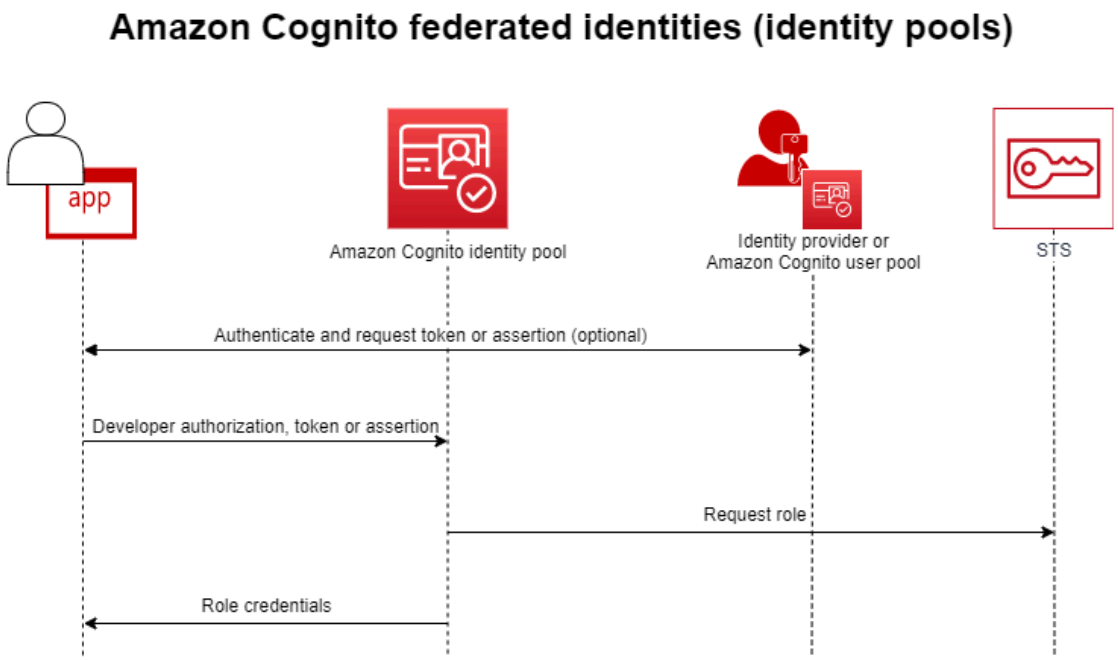
用户池



当您想要对您的应用程序或 API 进行身份验证和授权时，请创建用户群体。用户群体是一个用户目录，既有自助服务，也有管理员驱动的用户创建、管理和身份验证。用户群体可以是独立的目录和 OIDC 身份提供者（IdP），也可以是员工身份和客户身份的第三方提供者的中间服务提供者（SP）。组织的 SAML 2.0 和 OIDC IdP 将员工身份引入 Cognito 和您的应用程序。OAuth 2.0 的公有身份存储 Amazon、Google、Apple 和 Facebook 引入了客户身份。

用户群体不要求与身份池集成。从用户群体中，您可以直接向应用程序、Web 服务器或 API 颁发经过身份验证的 JSON Web 令牌（JWT）。

身份池



当您想要授权经过身份验证的用户或匿名用户访问您的 AWS 资源时，请设置 Amazon Cognito 身份池。身份池为您的应用程序颁发 AWS 凭证，以便向用户提供资源。您可以使用可信身份提供者（如用户群体或 SAML 2.0 服务）对用户进行身份验证。此身份提供者还可以选择为访客用户颁发凭证。身份池同时使用基于角色和基于属性的访问控制来管理您的用户访问 AWS 资源的授权。

身份池不要求与用户群体集成。身份池可以直接接受来自员工和使用者身份提供者的经过身份验证的声明。

将 Amazon Cognito 用户群体和身份池一起使用

在本主题开头的图表中，您使用 Amazon Cognito 对用户进行身份验证，然后向他们授予对 AWS 服务的访问权限。

1. 应用程序用户通过用户群体登录并接收 OAuth 2.0 令牌。
2. 应用程序将用户群体令牌及身份池交换为临时 AWS 凭证，您可以将这些凭证用于 AWS API 和 AWS Command Line Interface（AWS CLI）。
3. 应用程序将凭证会话分配给用户，并提供对诸如 Amazon S3 和 Amazon DynamoDB 等 AWS 服务的授权访问权限。

有关使用身份池和用户群体的更多示例，请参阅常见的 Amazon Cognito 场景。

在 Amazon Cognito 中，责任共担模式的云安全 义务符合 SOC 1-3、PCI DSS、ISO 27001 的要求且符合 HIPAA-BAA 的条件。您可以将 Amazon Cognito 中的云中安全性设计为符合 SOC1-3、ISO 27001 和 HIPAA-BAA 的要求，但不符合 PCI DSS。有关更多信息，请参阅范围内的 AWS 服务。另请参阅区域数据注意事项。

Amazon Cognito 的功能

用户池

Amazon Cognito 用户群体是一个用户目录。利用用户群体，您的用户可以通过 Amazon Cognito 登录您的 Web 或移动应用程序，也可以通过第三方 IdP 进行联合身份验证。联合用户和本地用户在您的用户群体中拥有用户配置文件。

本地用户是那些已注册或您直接在用户群体中创建的用户。您可以在 AWS Management Console、AWS SDK 或 AWS Command Line Interface（AWS CLI）中管理和自定义这些用户配置文件。

Amazon Cognito 用户群体接受来自第三方 IdP 的令牌和断言，并将用户属性收集到它发布给应用程序的 JWT 中。您可以在一组 JWT 上对应用程序进行标准化，而 Amazon Cognito 则处理与 IdP 的互动，同时将其声明映射到中央令牌格式。

Amazon Cognito 用户群体可以是独立的 IdP。Amazon Cognito 以 OpenID Connect（OIDC）标准为基础生成用于身份验证和授权的 JWT。当您登录本地用户时，用户群体对这些用户具有权限。对本地用户进行身份验证时，您可以访问以下功能。

- 实现您自己的 Web 前端，此前端调用 Amazon Cognito 用户群体 API 来对用户进行验证、授权和管理。
- 为用户设置多重身份验证（MFA）。Amazon Cognito 支持基于时间的一次性密码（TOTP）和短信 MFA。
- 可防止来自受恶意控制的用户账户的访问。
- 创建您自己的自定义多步骤身份验证流程。
- 在另一个目录中查找用户并将其迁移到 Amazon Cognito。

Amazon Cognito 用户群体还可以扮演双重角色，即 IdP 的服务提供者（SP）和应用程序的 IdP。Amazon Cognito 用户群体可以连接到如 Facebook 和 Google 等使用者 IdP，也可以连接到如 Okta 和 Active Directory Federation Services（ADFS）等员工 IdP。

有了 Amazon Cognito 用户群体颁发的 OAuth 2.0 和 OpenID Connect（OIDC）令牌，您可以

- 在应用程序中接受 ID 令牌，该令牌可对用户进行身份验证，并提供设置用户配置文件所需的信息
- 在您的 API 中接受访问令牌，此令牌具有用于对用户的 API 调用进行授权的 OIDC 范围。
- 从 Amazon Cognito 身份池中检索 AWS 凭证。

Amazon Cognito 用户群体的功能

功能	描述
OIDC IdP	Issue ID tokens to authenticate users
Authorization server	Issue access tokens to authorize user access to APIs
SAML 2.0 SP	Transform SAML assertions into ID and access tokens
OIDC SP	Transform OIDC tokens into ID and access tokens
OAuth 2.0 SP	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 the hosted UI
API support for your own UI	Create, manage and authenticate users through API requests in supported AWS SDKs[1]
MFA	Use SMS messages, TOTPs, 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[1]
Groups	Create logical groupings of users, and a hierarchy of IAM role claims when you pass tokens to identity pools
Customize ID tokens	Customize your ID tokens with new, modified, and suppressed claims
Customize user attributes	Assign values to user attributes and add your own custom attributes

[1] 功能仅适用于本地用户。

有关用户池的更多信息，请参阅用户池入门 (p. 30)和 Amazon Cognito 用户池 API 参考。

身份池

身份池是您分配给用户或访客并授权接收临时 AWS 凭证的唯一标识符或身份的集合。当您以来自 SAML 2.0、OpenID Connect（OIDC）或 OAuth 2.0 社交身份提供者（IdP）的可信声明的形式向身份池提供身份验证证明时，您将用户与身份池中的身份相关联。身份池为身份创建的令牌可以从 AWS Security Token Service（AWS STS）检索临时会话凭证。

为了补充经过身份验证的身份，您还可以配置一个身份池来授权 AWS 访问，而无需 IdP 身份验证。您可以提供自己的自定义身份验证证明，也可以不提供身份验证。您可以使用未经身份验证的身份向任何请求临时 AWS 凭证的应用程序用户授予此类凭证。身份池还接受声明，并根据您自己的自定义模式，使用经过开发人员验证的身份颁发凭证。

使用 Amazon Cognito 身份池，您可以通过两种方式在您的 AWS 账户中与 IAM policy 集成。您可以同时使用这两个功能，也可以单独使用。

基于角色的访问控制

当用户将声明传递到身份池时，Amazon Cognito 会选择它请求的 IAM 角色。要根据您的需求自定义角色的权限，您可以对每个角色应用 IAM policy。例如，如果您的用户证明自己在市场营销部门工作，则他们将获得相应角色的凭证，该角色的策略是根据营销部门访问需求量身定制的。Amazon Cognito 可以请求原定设置角色、基于查询用户声明的规则来请求角色，或者基于用户在用户群体中的组成员资格来请求角色。您还可以配置角色信任策略，以便 IAM 仅信任您的身份池来生成临时会话。

访问控制属性

您的身份池读取用户声明中的属性，并将它们映射到用户临时会话中的主体标签。然后，您可以配置基于 IAM 资源的 IAM policy，以基于您的身份池中携带会话标签的 IAM 主体允许或拒绝访问资源。例如，如果您的用户证明自己在市场营销部门工作，则 AWS STS 为他们的会话添加标签 `Department: marketing`。Amazon S3 桶允许基于 aws:PrincipalTag 条件的读取操作，该条件要求 `Department` 标签的值为 `marketing`。

Amazon Cognito 身份池的功能

功能	描述
Amazon Cognito user pool SP	Exchange an ID token from your user pool for web identity credentials from AWS STS
SAML 2.0 SP	Exchange SAML assertions for web identity credentials from AWS STS
OIDC SP	Exchange OIDC tokens for web identity credentials from AWS STS
OAuth 2.0 SP	Exchange OAuth tokens from Amazon, Facebook, Google, Apple, and Twitter for web identity credentials from AWS STS
Custom SP	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

有关身份池的更多信息，请参阅Amazon Cognito 身份池入门（联合身份）(p. 289)和 Amazon Cognito 身份池 API 参考。

Amazon Cognito 用户群体和身份池的比较

功能	描述	用户池	身份池
OIDC IdP	Issue OIDC ID tokens to authenticate app users	✓	
API authorization server	Issue access tokens to authorize user access to APIs, databases, and other resources that accept OAuth 2.0 authorization scopes	✓	
IAM web identity authorization server	Generate tokens that you can exchange with AWS STS for temporary AWS credentials		✓
SAML 2.0 SP & OIDC IdP	Issue customized OIDC tokens based on claims from a SAML 2.0 IdP	✓	

OIDC SP & OIDC IdP	Issue customized OIDC tokens based on claims from an OIDC IdP	✓	
OAuth 2.0 SP & OIDC IdP	Issue customized OIDC tokens based on scopes from OAuth 2.0 social providers like Apple and Google	✓	
SAML 2.0 SP & credentials broker	Issue temporary AWS credentials based on claims from a SAML 2.0 IdP		✓
OIDC SP & credentials broker	Issue temporary AWS credentials based on claims from an OIDC IdP		✓
OAuth 2.0 SP & credentials broker	Issue temporary AWS credentials based on scopes from OAuth 2.0 social providers like Apple and Google		✓
Amazon Cognito user pool SP & credentials broker	Issue temporary AWS credentials based on OIDC claims from an Amazon Cognito user pool		✓
Custom SP & credentials broker	Issue temporary AWS credentials based on developer IAM authorization		✓
Authentication frontend service	Sign up, manage, and authenticate users with the hosted UI	✓	
API support for your own authentication UI	Create, manage and authenticate users through API requests in supported AWS SDKs[1]	✓	
MFA	Use SMS messages, TOTPs, 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]	✓	
Groups	Create logical groupings of users, and a hierarchy of IAM role claims when you pass tokens to identity pools	✓	
Customize ID tokens	Customize your ID tokens with new, modified, and suppressed claims	✓	
AWS WAF web ACLs	Monitor and control requests to your authentication environment 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 roles to only be assumed in the context of your identity pool		✓
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] 功能仅适用于本地用户。

Amazon Cognito 入门

有关首要任务和开始位置的指南，请参阅 Amazon Cognito 入门 (p. 10)。

有关视频、文章、文档和示例应用程序，请参阅 Amazon Cognito 开发人员资源。

您需要一个AWS账户才能使用 Amazon Cognito。有关更多信息，请参阅使用 Amazon Cognito 控制台 (p. 8)。

区域可用性

Amazon Cognito 已在全球多个AWS区域推出。在每个区域中，Amazon Cognito 分布在多个可用区内。这些可用区的物理位置是相互隔离的，但可通过私有、低延迟、高吞吐量和高度冗余的网络连接联合在一起。这些可用区使AWS能够提供极高水平的可用性和冗余的服务（包括 Amazon Cognito），同时最大程度地减少延迟。

有关 Amazon Cognito 当前可用的所有区域的列表，请参阅《Amazon Web Services 一般参考》中的 AWS 区域和端点。要详细了解每个区域中可用的可用区数量，请参阅 AWS 全球基础设施。

Amazon Cognito 的定价

有关 Amazon Cognito 定价的信息，请参阅 Amazon Cognito 定价。

使用 Amazon Cognito 控制台

您可以使用 Amazon Cognito 控制台创建和管理用户池和身份池。

本指南为 Amazon Cognito 控制台中常见的 Amazon Cognito 用户群体任务提供了分步演练。

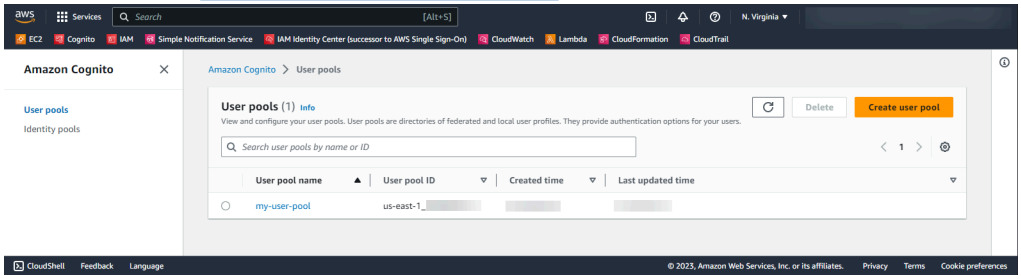
使用 Amazon Cognito 控制台

1. 您需要注册AWS账户才能使用 Amazon Cognito。
2. 转到 Amazon Cognito 控制台。系统可能会提示您输入 AWS 凭证。
3. 要创建或编辑用户池，请从左侧导航窗格中选择 User Pools（用户池）。

   有关更多信息，请参阅用户池入门 (p. 30)。
4. 要创建或编辑身份池，请选择身份池。您将被导向到原始控制台以获取 Amazon Cognito 身份池。

   有关更多信息，请参阅Amazon Cognito 身份池入门（联合身份） (p. 289)。

Amazon Cognito 控制台是AWS Management Console的一部分，后者提供了有关账户和账单的信息。有关更多信息，请参阅使用 AWS Management Console。



将此服务与 AWS SDK 结合使用

AWS 软件开发工具包（SDK）适用于许多常用编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

软件开发工具包文档	代码示例
AWS SDK for C++	AWS SDK for C++ 代码示例
AWS SDK for Go	AWS SDK for Go 代码示例
AWS SDK for Java	AWS SDK for Java 代码示例
AWS SDK for JavaScript	AWS SDK for JavaScript 代码示例
AWS SDK for Kotlin	AWS SDK for Kotlin 代码示例
AWS SDK for .NET	AWS SDK for .NET 代码示例
AWS SDK for PHP	AWS SDK for PHP 代码示例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 代码示例
AWS SDK for Ruby	AWS SDK for Ruby 代码示例
AWS SDK for Rust	AWS SDK for Rust 代码示例
AWS SDK for Swift	AWS SDK for Swift 代码示例

可用性示例

找不到所需的内容？ 通过使用此页面底部的 Provide feedback（提供反馈）链接请求代码示例。

Amazon Cognito 入门

本节介绍首要 Amazon Cognito 任务以及从何处入手。有关 Amazon Cognito 的概述，请参阅什么是 Amazon Cognito？ (p. 1)。

Amazon Cognito 的两个主要组件是用户池和身份池。用户池是为您的 Web 和移动应用程序用户提供注册和登录选项的用户目录。身份池提供AWS凭证以向用户授予对其他AWS服务的访问权限。您可以单独使用或配合使用用户池和身份池。

首要任务和开始位置
使用用户池添加注册和登录
1. 使用用户池创建用户目录。 2. 添加应用程序以启用托管 UI。 3. 向用户池添加社交登录。 4. 向用户池添加通过基于 SAML 的身份提供商 (IdP) 进行的登录。 5. 向用户池添加通过 OpenID Connect (OIDC) IdP 进行的登录。 6. 安装用户池开发工具包。 7. 自定义内置托管 Web UI 登录页面和注册页面。 8. 配置用户池安全功能。 9. 使用 Lambda 触发器自定义用户池工作流程。 10. 使用 Amazon Pinpoint 分析收集数据和目标活动。
在用户池中管理用户
• 注册并确认用户账户。 • 以管理员身份创建用户账户。 • 管理和搜索用户账户。 • 向用户池添加组。 • 将用户导入到用户池。
访问资源
常见的 Amazon Cognito 场景： • 使用用户池进行身份验证。 • 通过用户池访问后端资源。 • 通过用户池访问 API Gateway 和 Lambda。 • 通过用户池和身份池访问AWS服务。 • 通过第三方和身份池访问AWS服务。 • 通过用户池或身份池访问 AWS AppSync 资源。

注册一个 AWS 账户

如果您还没有 AWS 账户，请完成以下步骤来创建一个。

注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

   当您注册 AWS 账户 时，系统将会创建一个 AWS 账户根用户。根用户有权访问该账户中的所有 AWS 服务和资源。作为安全最佳实践，请 为管理用户分配管理访问权限，并且只使用根用户执行 需要根用户访问权限的任务。

AWS注册过程完成后，会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择 My Account (我的账户) 来查看当前的账户活动并管理您的账户。

创建管理用户

注册 AWS 账户 后，创建一个管理用户，以避免使用根用户执行日常任务。

保护您的 AWS 账户根用户

1. 选择 Root user（根用户）并输入您的 AWS 账户 电子邮件地址，以账户拥有者身份登录 AWS Management Console。在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。
2. 对您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅《IAM 用户指南》中的为 AWS 账户根用户启用虚拟 MFA 设备（控制台）。

创建管理用户

- 对于您的日常管理任务，请在 AWS IAM Identity Center (successor to AWS Single Sign-On) 中为管理用户授予管理访问权限。

  有关说明，请参阅《AWS IAM Identity Center (successor to AWS Single Sign-On) 用户指南》中的入门。

作为管理用户登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录 URL。

  要获取使用 IAM Identity Center 用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的 登录 AWS 访问门户。

Amazon Cognito 常见场景

本主题介绍使用 Amazon Cognito 的六个常见场景。

Amazon Cognito 的两个主要组件是用户池和身份池。用户池是为您的 Web 和移动应用程序用户提供注册和登录选项的用户目录。身份池提供AWS凭证以向用户授予对其他AWS服务的访问权限。

用户池是 Amazon Cognito 中的用户目录。您的应用程序用户可以通过用户池直接登录，也可以通过第三方身份提供商（IdP）联合。用户池管理处理以下令牌的开销：从通过 Facebook、Google、亚马逊和 Apple 进行的社交登录返回的令牌以及从 OpenID Connect（OIDC）和 SAML IdP 返回的令牌。无论您的用户是直接登录还是通过第三方登录，用户池的所有成员都有一个可通过开发工具包访问的目录配置文件。

借助身份池，您的用户可以获取临时AWS凭证来访问AWS服务，如 Amazon S3 和 DynamoDB。身份池支持匿名来访用户以及通过第三方 IdP 的联合身份验证。

主题

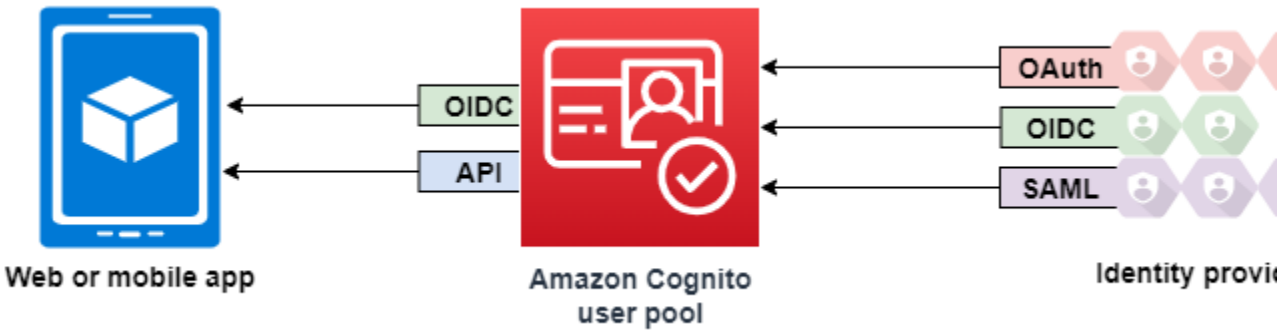
- 使用用户池进行身份验证 (p. 12)
- 使用用户池访问您的服务器端资源 (p. 13)
- 将 API Gateway 和 Lambda 与用户池结合使用来访问资源 (p. 13)
- 使用用户池和身份池访问 AWS 服务 (p. 14)
- 借助第三方进行身份验证并使用身份池访问 AWS 服务 (p. 14)
- 使用 Amazon Cognito 访问 AWS AppSync 资源 (p. 15)

使用用户池进行身份验证

您可以允许您的用户使用用户池进行身份验证。您的应用程序用户可以通过用户池直接登录，也可以通过第三方身份提供商（IdP）联合。用户池管理处理以下令牌的开销：从通过 Facebook、Google、亚马逊和 Apple 进行的社交登录返回的令牌以及从 OpenID Connect（OIDC）和 SAML IdP 返回的令牌。

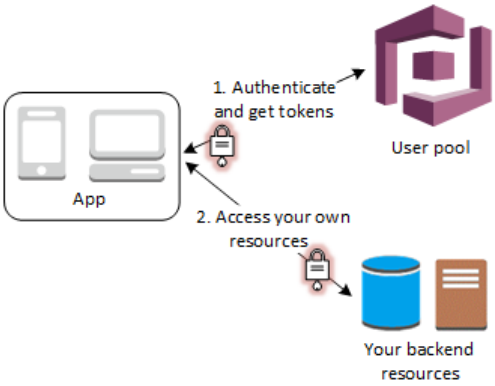
成功进行身份验证后，您的 Web 或移动应用程序将收到来自 Amazon Cognito 的用户池令牌。可以使用这些令牌检索允许您的应用程序访问其他AWS服务的AWS凭证，也可以选择使用它们来控制对您的服务器端资源或 Amazon API Gateway 的访问。

有关更多信息，请参阅用户池身份验证流程 (p. 41)和将令牌与用户池结合使用 (p. 203)。



使用用户池访问您的服务器端资源

成功进行用户池登录后，您的 Web 或移动应用程序将收到来自 Amazon Cognito 的用户池令牌。您可以使用这些令牌控制对您的服务器端资源的访问。您也可以创建用户池组来管理权限以及表示不同类型的用户。有关使用组控制资源访问权限的更多信息，请参阅向用户池添加组 (p. 178)。



在为用户群体配置域后，Amazon Cognito 会预调配一个托管 Web UI，您可使用此 UI 为应用程序添加注册页和登录页。使用此 OAuth 2.0 基础，您可以创建自己的资源服务器，从而使您的用户能够访问受保护的资源。有关更多信息，请参阅为您的用户池定义资源服务器 (p. 82)。

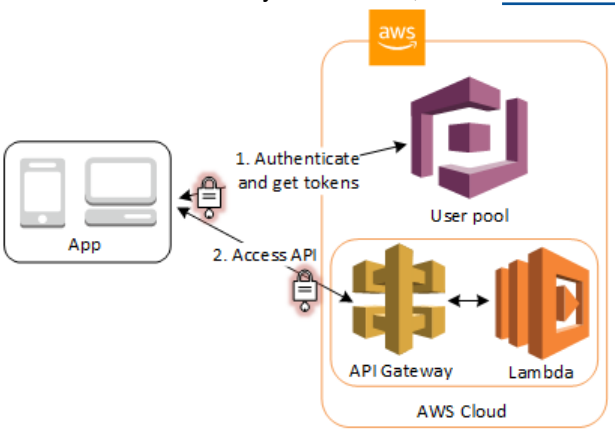
有关用户群体身份验证的更多信息，请参阅用户池身份验证流程 (p. 41)和将令牌与用户池结合使用 (p. 203)。

将 API Gateway 和 Lambda 与用户池结合使用来访问资源

您可以允许用户通过 API Gateway 访问您的 API。API Gateway 会验证来自成功用户池身份验证的令牌，并使用它们向您的用户授予对资源（包括 Lambda 函数）或您自己的 API 的访问权限。

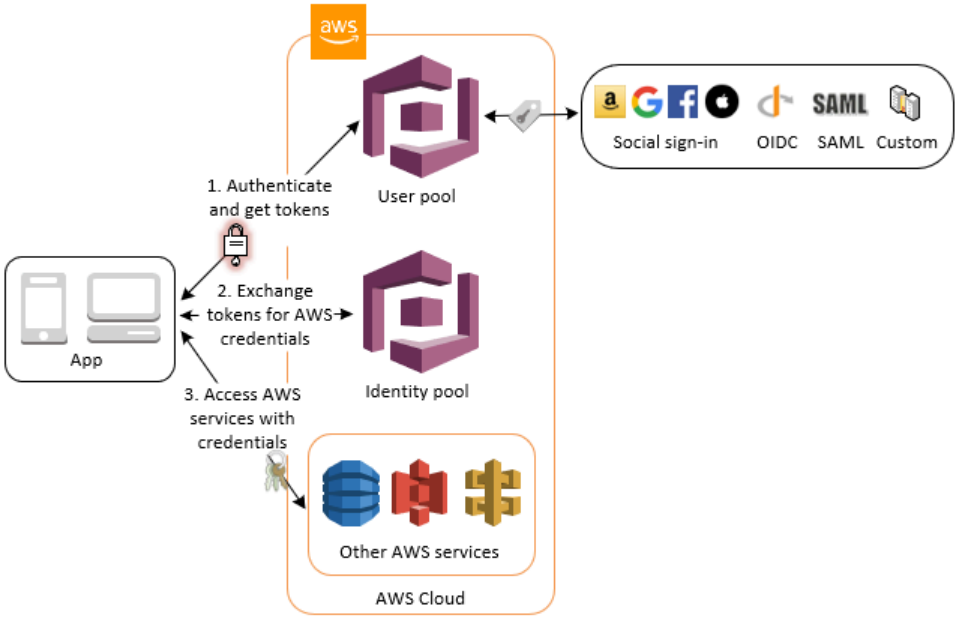
您可以使用用户池中的组控制对 API Gateway 的权限，方法是将组成员资格映射到 IAM 角色。用户所属的组包含在您的应用程序用户登录时用户池提供的 ID 令牌中。有关用户池组的更多信息，请参阅向用户池添加组 (p. 178)。

您可以将您的用户池令牌随请求一起提交到 API Gateway，以便 Amazon Cognito 授权方 Lambda 函数进行验证。有关 API Gateway 的更多信息，请参阅将 API Gateway 与 Amazon Cognito 用户池结合使用。



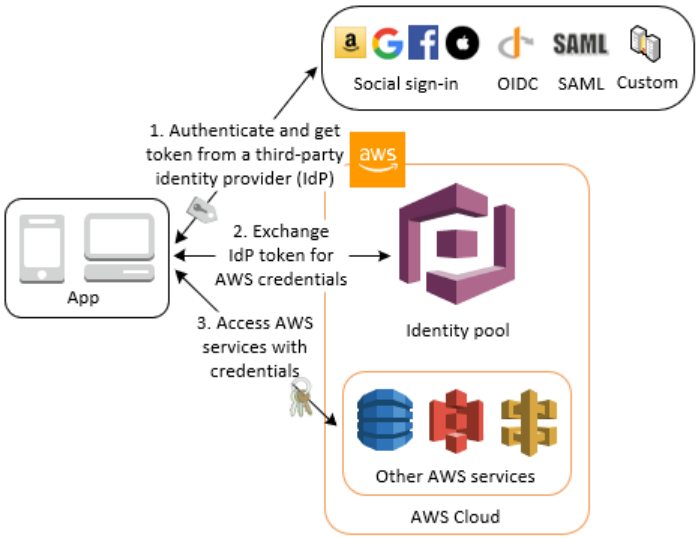
使用用户池和身份池访问 AWS 服务

成功进行用户池身份验证后，您的应用程序将收到来自 Amazon Cognito 的用户池令牌。您可以用它们交换用于使用身份池对其他AWS服务的临时访问令牌。有关更多信息，请参阅在登录后使用身份池访问 AWS 服务 (p. 217)和Amazon Cognito 身份池入门（联合身份） (p. 289)。



借助第三方进行身份验证并使用身份池访问 AWS 服务

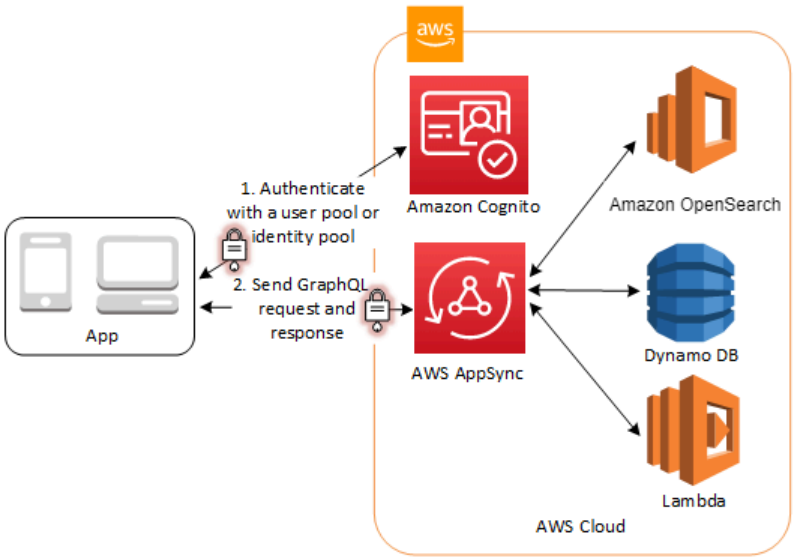
您可以允许您的用户通过身份池访问AWS服务。身份池需要来自由第三方身份提供商进行身份验证的用户的 IdP 令牌 (如果是匿名来访者，则不需要令牌)。作为交换，身份池将授予可用来访问其他AWS服务的临时AWS凭证。有关更多信息，请参阅Amazon Cognito 身份池入门（联合身份） (p. 289)。



使用 Amazon Cognito 访问 AWS AppSync 资源

您可以向用户授予访问权限，让他们能够使用来自成功的 Amazon Cognito 用户群体身份验证的令牌访问 AWS AppSync 资源。有关更多信息，请参阅《AWS AppSync 开发人员指南》中的 AMAZON_COGNITO_USER_POOLS 授权。

还可以使用从身份池收到的 IAM 凭证签署对 AWS AppSync GraphQL API 的请求。请参阅 AWS_IAM 授权。



Amazon Cognito 教程

Amazon Cognito 的两个主要组件是用户池和身份池。用户池是为您的 Web 和移动应用程序用户提供注册和登录选项的用户目录。身份池提供AWS凭证以向用户授予对其他AWS服务的访问权限。

主题

- 教程：创建用户池 (p. 16)
- 教程：创建身份池 (p. 16)
- 教程：清除 AWS 资源 (p. 18)

教程：创建用户池

利用用户池，您的用户可以通过 Amazon Cognito 登录您的 Web 或移动应用程序。

创建用户池

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 在页面右上角，选择 Create a user pool（创建用户池）以开启用户池创建向导。
4. 在 Configure sign-in experience（配置登录体验）中，选择您将与此用户池一起使用的联合提供商。有关更多信息，请参阅通过第三方添加用户池登录。
5. 在 Configure security requirements（配置安全要求）中，选择密码策略、多重身份验证（MFA）要求以及用户账户恢复选项。有关更多信息，请参阅 Amazon Cognito 中的安全性。
6. 在 Configure sign-up experience（配置注册体验）中，确定新用户在注册时如何验证其身份，以及在用户注册流程中应该要求哪些属性或可选属性。有关更多信息，请参阅管理用户池中的用户。
7. 在 Configure message delivery（配置消息传输）中，配置与 Amazon Simple Email Service 和 Amazon Simple Notification Service 的集成，以便向用户发送电子邮件和 SMS 消息以进行注册、账户确认、MFA 和账户恢复。有关更多信息，请参阅 Amazon Cognito 用户池的电子邮件设置和 Amazon Cognito 用户池的 SMS 消息设置。
8. 在 Integrate your app（集成应用程序）中，命名用户池、配置托管 UI 并创建应用程序客户端。有关更多信息，请参阅添加应用程序以启用托管 Web UI。
9. 在 Review and create（审核和创建）屏幕中，查看您的选择并修改任意您希望的选择。如果您对用户池配置感到满意，请选择 Create user pool（创建用户池）继续执行。

相关资源

有关用户池的更多信息，请参阅Amazon Cognito 用户池 (p. 27)。

另请参阅 用户池身份验证流程 (p. 41) 和 将令牌与用户池结合使用 (p. 203)。

教程：创建身份池

借助身份池，您的用户可以获取临时AWS凭证来访问AWS服务，如 Amazon S3 和 DynamoDB。

New console

在控制台中创建新的身份池

1. 登录 Amazon Cognito 控制台并选择身份池。

2. 选择创建身份池。
3. 在配置身份池信任中，选择将您的身份池设置为经过身份验证的访问权限和/或访客访问权限。

   - 如果您选择了经过身份验证的访问权限，请在身份池中选择一个或多个您要设置为经过身份验证的身份来源的身份类型。如果您配置了自定义开发人员提供者，则在创建身份池后无法对其进行修改或删除。

4. 在配置权限中，为身份池中经过身份验证的用户或访客用户选择原定设置 IAM 角色。

   a. 如果您希望 Amazon Cognito 为您创建一个具有基本权限并与您的身份池建立信任关系的新角色，请选择创建新的 IAM 角色。例如，输入 IAM 角色名称以标识您的新角色，例如 `myidentitypool_authenticatedrole`。选择查看策略文档以查看 Amazon Cognito 将分配给新 IAM 角色的权限。

   b. 如果您的 AWS 账户中已经有想要使用的角色，则可以选择使用现有的 IAM 角色。您必须将您的 IAM 角色信任策略配置为包括 `cognito-identity.amazonaws.com`。配置您的角色信任策略，以仅允许 Amazon Cognito 在提供证据证明请求来自您的特定身份池中经过身份验证的用户时，才代入该角色。有关更多信息，请参阅角色信任和权限 (p. 312)。

5. 在连接身份提供者中，输入您在配置身份池信任中选择的身份提供者（IdP）的详细信息。可能会要求您提供 OAuth 应用程序客户端信息、选择 Amazon Cognito 用户群体、选择 IAM IdP 或输入开发人员提供者的自定义标识符。

   a. 为每个 IdP 选择角色设置。您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。使用 Amazon Cognito 用户群体 IdP，还可以选择令牌中包含 preferred_role 声明的角色。有关 `cognito:preferred_role` 声明的更多信息，请参阅将优先级值分配到组 (p. 179)。

      i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。

      ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。

   b. 您可以为每个 IdP 配置访问控制属性。访问控制属性将用户声明映射到 Amazon Cognito 应用于其临时会话的主体标签。您可以构建 IAM policy，以根据应用于用户会话的标签来筛选用户访问权限。

      i. 如果不应用主体标签，请选择非活动。

      ii. 要基于 `sub` 和 `aud` 声明应用主体标签，请选择使用原定设置映射。

      iii. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。

6. 在配置属性中，在身份池名称下输入名称。
7. 在基本（经典）身份验证下，选择是否要激活基本流程。在基本流程处于活动状态的情况下，您可以绕过为 IdP 所做的角色选择，直接调用 AssumeRoleWithWebIdentity。有关更多信息，请参阅身份池（联合身份）身份验证流程 (p. 300)。
8. 如果要将标签应用到身份池，请在标签下选择添加标签。
9. 在查看并创建中，确认您为新身份池所做的选择。选择编辑以返回向导并更改任何设置。完成后，选择创建身份池。

AWS Management Console

创建身份池

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 Manage Identity Pools（管理身份池）。
3. 选择 Create new identity pool（创建新身份池）。
4. 为身份池输入一个名称。

5. 要启用未经身份验证的身份，请从 Unauthenticated identities（未经身份验证的身份）的可折叠部分中选择 Enable access to unauthenticated identities（启用访问未经身份验证的身份）。
6. 选择 Create Pool（创建池）。
7. 系统将提示您访问您的 AWS 资源。

   选择 Allow（允许）以创建两个与您的身份池关联的原定设置角色，一个用于未经身份验证的用户，另一个用于经过身份验证的用户。这些默认角色会向 Amazon Cognito Sync 提供身份池访问权限。您可以在 IAM 控制台中修改与身份池关联的角色。
8. 记下您的身份池 ID 号。您将使用它来设置允许您的应用程序用户访问其它 AWS 服务（如 Amazon Simple Storage Service 或 DynamoDB）的策略

相关资源

有关身份池的更多信息，请参阅Amazon Cognito 身份池（联合身份）(p. 288)。

有关使用具有身份池的 Amazon S3 示例，请参阅通过浏览器将照片上载到 Amazon S3。

教程：清除 AWS 资源

New console

删除身份池

1. 从 Amazon Cognito 控制台中选择身份池。选中要删除的身份池旁边的单选按钮。
2. 选择删除。
3. 输入或粘贴身份池的名称，然后选择删除。

AWS Management Console

删除身份池

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 Manage Identity Pools（管理身份池）。
3. 选择要删除的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
4. 在控制面板页面的右上角，选择 Edit identity pool (编辑身份池)。此时将显示 Edit identity pool（编辑身份池）页。
5. 向下滚动并选择 Delete identity pool（删除身份池）以将其展开。
6. 选择 Delete identity pool（删除身份池）。
7. 选择 Delete pool（删除池）。

删除用户池

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 从导航窗格中选择 User Pools（用户池）。
3. 如果您为用户群体分配了自定义域，请选择要删除的用户群体，导航到应用程序集成选项卡，选择操作，然后选择删除自定义域。
4. 返回到用户群体列表并选中要删除的身份池旁边的单选按钮。
5. 选择删除。
6. 如果您的用户群体中的删除保护处于活动状态，请选中停用删除保护复选框。

7. 如果您已经为用户群体分配 Cognito 域，请选中删除 Cognito 域复选框。
8. 输入或粘贴您的用户群体的名称，然后选择删除。

将 Amazon Cognito 身份验证和授权与 Web 和移动应用程序集成

主题

- 使用 AWS Amplify 框架进行 Amazon Cognito 身份验证 (p. 20)
- 使用 amazon-cognito-identity-js 的身份验证 (p. 21)
- 使用 AWS 开发工具包进行身份验证 (p. 21)
- 使用 Amazon Verified Permissions 进行授权 (p. 21)

当新用户发现您的程序时，或当现有用户返回使用您的应用程序时，他们的第一个任务是注册或登录。当您将应用程序与 Amazon Cognito 应用程序客户端集成时，您可以调用 API 操作来对用户进行身份验证和授权。在典型的工作流中，应用程序将使用 Amazon Cognito API 在用户群体中创建新用户、检索用户群体令牌以及从身份池中获取临时凭证。要将 Amazon Cognito 与您的 Web 应用程序或移动应用程序集成，请使用 AWS 开发工具包和库。

Amazon Cognito 提供 AWS Management Console集成和托管用户界面等可视化工具。Amazon Cognito 也可以在您的应用程序代码中使用。在使用 Amazon Cognito 进行身份验证和授权之前，请选择应用程序平台并准备代码以与服务集成。AWS CLI 是一款适用于 Amazon Cognito 和其他 AWS 服务的基于文本的交互式 SDK，是开始自行熟悉 Amazon Cognito API 的重要位置。

> Note
>
> Amazon Cognito 的某些组件只能使用 API 进行配置。例如，您只能使用用于更新 CreateUserPool 或 UpdateUserPool API 请求中 UserPool 类的 `LambdaConfig` 属性的请求，来设置用户群体自定义短信或电子邮件发件人 Lambda 触发器。

有关在应用程序中使用 Amazon Cognito 的更多信息，请参阅使用 AWS SDK 的 Amazon Cognito 的代码示例。

使用 AWS Amplify 框架进行 Amazon Cognito 身份验证

AWS Amplify 向 Web 和移动开发人员提供服务和库。借助 Amplify，您可以构建与由 AWS 服务组成的后端环境集成的应用程序。要预置后端环境，以及将 AWS 服务与客户端代码集成，请使用 Amplify Framework。该框架提供了一个交互式命令行界面 (CLI)，可帮助您为按类别组织的功能配置 AWS 资源。这些类别包括分析、存储和身份验证等。该框架还为 Web 和移动平台（包括 iOS、Android 及 JavaScript）提供了高级开发工具包和库。JavaScript 框架包括 React、React Native、Angular、Ionic 和 Vue。每个开发工具包和库都包含身份验证操作，您可以使用这些操作来实施 Amazon Cognito 驱动的身份验证工作流。

要使用 Amplify Framework 框架向应用程序添加身份验证功能，请参阅您平台的 Amplify 授权文档：

- 适用于 JavaScript 的 Amplify 身份验证
- 适用于 iOS 的 Amplify 身份验证
- 适用于 Android 的 Amplify 身份验证

- 放大 Flutter 的身份验证

使用 amazon-cognito-identity-js 的身份验证

适用于 JavaScript 的 Amazon Cognito Identity SDK 使得应用程序可以在 Amazon Cognito 用户群体中注册本地用户和对其进行身份验证。应用程序还可以使用此开发工具包在 Amazon Cognito 用户池中查看、删除和更新用户属性。amazon-cognito-identity-js 程序包提供了示例代码，使得经过了身份验证的用户可以更改其密码。该程序包还提供了示例代码，可以为未经身份验证的用户启动并完成找回遗忘密码的流程。

适用于 JavaScript 的 Amazon Cognito Identity 开发工具包。

使用 AWS 开发工具包进行身份验证

要使用安全的后端构建您自己的身份微服务以与 Amazon Cognito 进行交互，请使用您选择的语言的 AWS SDK 连接到 Amazon Cognito 用户群体和 Amazon Cognito 身份池 API。

有关各个 API 操作的详细信息，请参阅 Amazon Cognito 用户池 API 参考和 Amazon Cognito API 参考。这些文档包含另请参阅部分，其中提供了在支持的平台上使用各种 SDK 的资源。

- AWS 命令行界面
- 适用于 .NET 的 AWS SDK
- 适用于 C++ 的 AWS SDK
- 适用于 Go 的 AWS SDK
- 适用于 Java V2 的 AWS SDK
- 适用于 JavaScript 的 AWS SDK
- 适用于 PHP V3 的 AWS SDK
- 适用于 Python 的 AWS SDK
- 适用于 Ruby V3 的 AWS SDK

使用 Amazon Verified Permissions 进行授权

Amazon Verified Permissions 是针对您构建的应用程序的授权服务。当您将 Amazon Cognito 用户群体添加为身份源时，应用程序可以将用户群体访问权限或身份（ID）令牌传递给 Verified Permissions，以便做出允许或拒绝决定。Verified Permissions 根据您使用 Cedar 策略语言编写的策略，来考虑用户的属性和请求上下文。请求上下文可以包括所请求的文档、映像或其他资源的标识符，以及用户想要对该资源采取的操作。

您的应用程序可以在 IsAuthorizedWithToken API 请求中向 Verified Permissions 提供用户的身份或访问令牌。`IsAuthorizedWithToken` API 还接受用户针对他们想要访问的 `Resource` 所请求的 `Action`，以及有助于做出精细访问决策的其他自定义 `Context`。

当应用程序在 `IsAuthorizedWithToken` API 请求中提供令牌时，Verified Permissions 将执行以下验证。

1. 您的用户群体是针对所请求的策略存储而配置的 Verified Permissions 身份来源。
2. 访问令牌或身份令牌中的 `client_id` 或 `aud` 声明分别与您提供给 Verified Permissions 的用户群体应用程序客户端 ID 相匹配。要验证此声明，您必须在 Verified Permissions 身份来源中配置客户端 ID 验证。
3. 您的令牌未过期。
4. 令牌中的签名来自用户群体中已发布的 JSON Web 密钥（JWK）。您可以在 `https://cognito-idp.`*`Region`*`.amazonaws.com/`*`your user pool ID`*`/.well-known/jwks.json` 中查看您的 JWK。

已撤销的令牌和已删除的用户

Verified Permissions 仅验证它从您的身份来源和用户令牌到期时间所了解的信息。Verified Permissions 不检查令牌是否撤销或用户是否存在。如果您从用户群体中撤销了用户的令牌或删除了用户的配置文件，则 Verified Permissions 在令牌到期之前仍会认为该令牌有效。

策略评估

将您的用户群体配置为策略存储的身份来源。将您的应用程序配置为在对 Verified Permissions 的请求中提交用户的令牌。对于每个请求，Verified Permissions 将令牌中的声明与策略进行比较。Verified Permissions 策略类似于 AWS 中的 IAM policy。该策略会声明主体、资源 和操作。如果您对于 `Allow` 的请求与允许的操作匹配，但与显式拒绝操作不匹配，则 Verified Permissions 将对您的请求进行响应；否则，它将以 `Deny` 进行响应。有关更多信息，请参阅《Amazon Verified Permissions 用户指南》中的 Amazon Verified Permissions 策略。

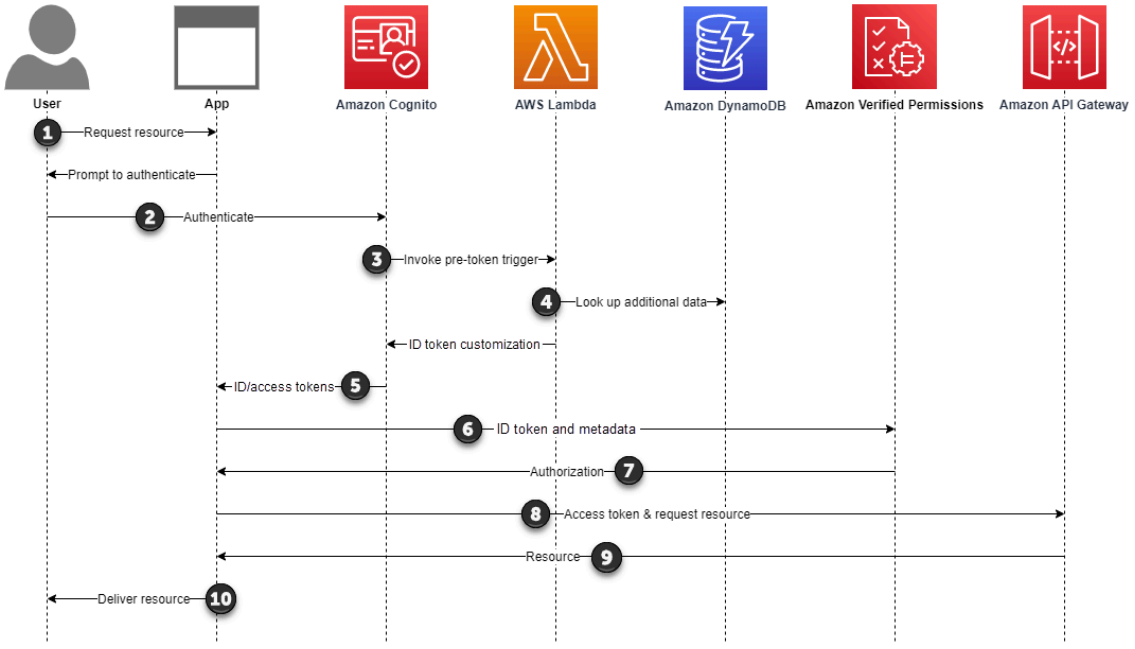
自定义令牌

要更改、添加和删除您想要向 Verified Permissions 提供的用户声明，请使用令牌生成前 Lambda 触发器 (p. 143)自定义身份令牌中的内容。使用令牌生成前触发器，您可以在令牌中添加和修改声明。例如，您可以在数据库中查询其他用户属性，并将这些属性编码为您的 ID 令牌。

> **Note**
>
> 由于 Verified Permissions 处理声明的方式，请勿在令牌生成前函数中添加名为 `cognito`、`dev` 或 `custom` 的声明。如果您提供的这些保留的声明前缀不是采用以冒号分隔的格式（例如 `cognito:username`），而是采用完整的声明名称，则您的授权请求会失败。
> Verified Permissions 不处理 `cognito:groups` 声明。

授权示例

以下是使用自定义 ID 令牌和从 Amazon API Gateway 检索资源的 Amazon Cognito 和 Verified Permissions 的用户群体流程示例。



授权流程示例

1. 用户在应用程序中请求资源，而应用程序必须从 API 中检索该资源。

2. 用户登录到您的 Amazon Cognito 用户群体。
3. 用户群体将用户的详细信息发送到用户群体令牌生成前 Lambda 触发器 (p. 143)。使用令牌生成前触发器是可选的。
4. Lambda 函数会在 Amazon DynamoDB 中查找用户，并将声明添加到用户的 ID 令牌中。
5. Amazon Cognito 将访问权限和 ID 令牌返回给应用程序。
6. 应用程序在对 Verified Permissions 的请求中提供用户的 ID 令牌和元数据。
7. Verified Permissions 处理用户的声明，并将其与您组织的策略存储中的策略进行比较。当它发现一个不覆盖 Deny 的 Allow 条件时，Verified Permissions 会以其 Allow 决定进行响应。
8. 应用程序处理 Allow 响应，并将用户的访问令牌提供给您的 API Gateway Amazon Cognito 授权者。在本例中，访问令牌具有双重作用：也即，它对 Verified Permissions 的精细访问控制决策和 API Gateway 的基于作用域的访问控制决策都具有重要贡献。
9. 您的 API 接受用户的令牌，并将请求的资源返回给应用程序。
10. 应用程序向用户显示所请求的资源。
11. 当用户请求其他受保护的资源时，应用程序将返回授权步骤，并将新的请求提供给 Verified Permissions。

Amazon Cognito 用户的示例策略

以下示例使用 Cedar 策略语言，以允许通过一个用户群体应用程序客户端进行身份验证的财务用户读写 example_image.png。John 是应用程序中的用户，他从应用程序客户端接收 ID 令牌，并在 GET 请求中将其传递到需要授权的 URL https://example.com/images/example_image.png。John 的 ID 令牌拥有用户群体应用程序客户端 ID 1234567890example 的 aud 声明。令牌生成前 Lambda 函数还插入了一个新声明 costCenter，对于 John 来说，值为 Finance1234。

```
permit (
   principal,
   actions in [Action::"readFile", "writeFile"],
   resource == ExampleCorp::File::"a8f3f048-ded7-4a95-89ca-e283644e1898" //
 example_image.png
)
when {
   principal.aud == "1234567890example" &&
   principal.costCenter like "Finance*"
};
```

以下请求正文会导致 Allow 响应。

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

当您要在 Verified Permissions 策略中指定主体时，请使用以下格式：

```
principal == namespace::User::"[user pool ID]"|"[user sub]"
```

以下是 ID 为 us-east-1_Example（带有子项）或用户 ID 为 973db890-092c-49e4-a9d0-912a4c0a20c7 的用户群体中用户的主体语句示例。

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

基于属性的访问控制

使用 Verified Permissions 对应用程序进行授权和适用于 AWS 凭证的 Amazon Cognito 身份池的访问控制属性功能都采用基于属性的访问权限控制（ABAC）形式。以下是 Verified Permissions 和 Amazon Cognito ABAC 的功能比较。在 ABAC 中，系统检查实体的属性，并根据您定义的条件做出授权决策。

服务	过程	结果
Amazon Verified Permissions	Returns an `Allow####` or `##` 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 会话标签 to your user based on their attributes. IAM policy conditions can check tags `Allow####` or `##` user access to AWS 服务.	A tagged session with temporary AWS credentials for an IAM role.

多租户应用程序最佳实践

您可以使用 Amazon Cognito 用户池保护小型多租户应用程序，其中租户数量和预期容量与相关 Amazon Cognito 服务配额相匹配。

> Note
>
> Amazon Cognito 配额根据各个 AWS 账户和区域应用。这些配额在应用程序中的所有租户之间共享。请查看 Amazon Cognito 服务配额，并确保配额符合您应用程序中的预期容量和预期租户数量。

有四种方法可以保护多租户应用程序：用户池、应用程序客户端、组或自定义属性。

主题

- 基于用户池的多租户 (p. 25)
- 基于应用程序客户端的多租户 (p. 25)
- 基于组的多租户 (p. 26)
- 基于自定义属性的多租户 (p. 26)
- 多租户安全建议 (p. 26)

基于用户池的多租户

利用此设计，您可以为应用程序中的每个租户创建一个用户池。此方法为每个租户提供最大程度的隔离。您可以为每个租户实施不同的配置。按用户池隔离租户为您在用户到租户映射中提供了灵活性。您可以为同一用户创建多个配置文件。但是，每个用户必须为他们可以访问的每个租户单独注册。使用此方法，可以单独为每个租户设置托管 UI，并将用户重新导向到您应用程序的租户特定实例。您还可以使用此方法与 API Gateway 等后端服务集成。

您可以在以下场景下使用基于用户池的多租户：

- 您的应用程序对于每个租户具有不同的配置。例如，每个租户的数据驻留要求、密码策略和 MFA 配置可能有所不同。
- 您的应用程序具有复杂的用户到租户角色映射。例如，某个用户可以在租户 A 中是“学生”，也可以在租户 B 中是“教师”。
- 您的应用程序使用原定设置 Amazon Cognito 托管 UI 作为本地用户进行身份验证的主要方法。本地用户仅存在于您的用户群体目录中，无需通过外部 IdP 进行联合身份验证。
- 您的应用程序具有孤立的多租户应用程序，其中每个租户都可以获取应用程序基础设施的完整实例以供其使用。

工作量级别

使用此方法的开发和操作工作量很大。您必须在使用 Amazon Cognito API 操作和自动化工具的应用程序中构建租户登录和管理组件。这些组件是为每个租户创建所需资源所必需的组件。您还需要实施与租户匹配的用户界面。此外，您必须向应用程序添加逻辑，以便用户注册和登录其相应租户的用户池。

基于应用程序客户端的多租户

通过基于应用程序客户端的多租户，您可以将同一用户映射到多个租户，而无需重新创建用户配置文件。您可以为每个租户创建一个应用程序客户端，并使租户外部 IdP 成为此应用程序客户端唯一可以使用的身份提供商。有关更多信息，请参阅配置用户池应用程序客户端。

托管 UI 在浏览器中设置会话 Cookie，以便它识别已经过身份验证的用户。当您向多个应用程序客户端对用户群体中的本地用户 进行身份验证时，用户的会话 Cookie 会在同一用户群体中针对所有应用程序客户端对用户进行身份验证。本地用户仅存在于您的用户群体目录中，无需通过外部 IdP 进行联合身份验证。会话 Cookie 的有效期为 1 小时。您无法更改会话 Cookie 的持续时间。

您可以在以下场景下使用基于应用程序客户端的多租户：

- 您的应用程序在所有租户中具有相同的配置。例如，数据驻留和密码策略在所有租户中都相同。
- 您的应用程序在用户与租户之间具有一对多映射。例如，一个用户可以使用同一个配置文件访问多个租户。
- 您有一个仅限联合身份验证的多租户应用程序，其中租户将始终使用外部 IdP 登录您的应用程序。
- 您有一个 B2B 多租户应用程序，且租户后端服务将使用客户端凭证授权访问您的服务。在这种情况下，您可以为每个租户创建应用程序客户端，并与租户后端服务共享客户端 ID 和密钥，以便进行计算机到计算机身份验证。

工作量级别

使用此方法的开发工作量很大。您必须实施与租户匹配的逻辑和用户界面，以便将用户与其租户的应用程序客户端进行匹配。

基于组的多租户

通过基于组的多租户，您可以将 Amazon Cognito 用户池组与租户相关联。这样，您就可以通过基于角色的访问控制 (RBAC) 来使用其他功能。有关更多信息，请参阅基于角色的访问控制。

基于自定义属性的多租户

通过基于自定义属性的多租户，您可以将租户标识数据（如 `tenant_id`）作为自定义属性存储在用户的配置文件中。然后，您可以在应用程序和后端服务中处理所有多租户逻辑。通过此方法，您可以对所有用户使用统一的注册和登录体验。要在应用程序中确定用户的租户，您可以检查此自定义属性。

多租户安全建议

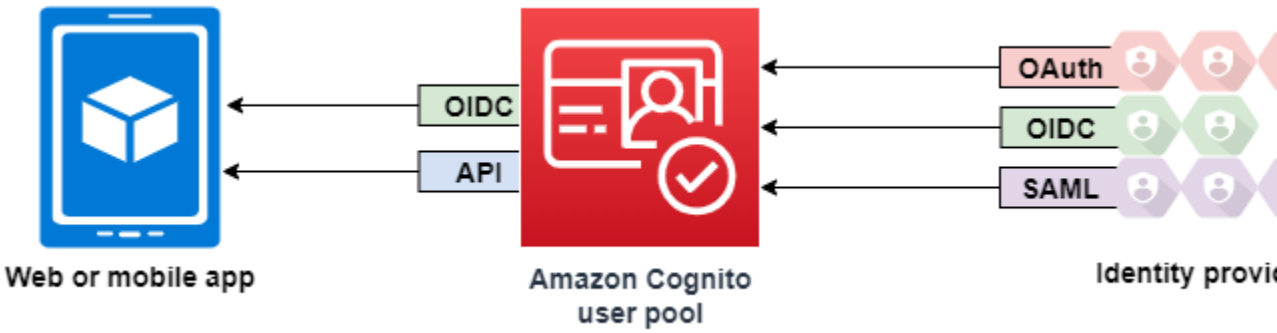
为了帮助提高应用程序的安全性，我们有下列建议：

- 仅使用经过验证的电子邮件地址，根据域匹配授权用户访问租户。仅信任经过您的应用程序验证或者或者外部 IdP 提供了验证证明的电子邮件地址和电话号码。有关设置这些权限的更多详细信息，请参阅属性权限和范围。
- 对标识租户的用户配置文件属性使用不可变 或只读属性。管理员必须能够更改这些属性。此外，向应用程序客户端授予对属性的只读访问权。
- 在外部 IdP 与应用程序客户端之间使用 1:1 映射，以防止未经授权的跨租户访问。已通过外部 IdP 身份验证且具有有效 Amazon Cognito 会话 Cookie 的用户，可以访问信任相同 IdP 的其他租户应用程序。
- 在应用程序中实施与租户匹配的授权逻辑时，请限制用户，使他们不能修改用于授权用户访问租户的条件。此外，如果使用外部 IdP 进行联合身份验证，请限制租户身份提供商管理员，使其无法修改用户访问权限。

Amazon Cognito 用户池

Amazon Cognito 用户群体是用于 Web 和移动应用程序身份验证和授权的用户目录。从应用程序的角度来看，Amazon Cognito 用户群体是 OpenID Connect（OIDC）身份提供者（IdP）。用户群体为安全性、身份联合验证、应用程序集成和用户体验自定义添加了多层附加功能。

例如，您可以验证用户的会话是否来自可信来源。您可以将 Amazon Cognito 目录与外部身份提供者相结合。使用您的首选 AWS SDK，您可以选择最适合应用程序的 API 授权模型。您还可以添加用于修改或彻底修改 Amazon Cognito 原定设置行为的 AWS Lambda 函数。



主题

- 功能 (p. 27)
- 用户池入门 (p. 30)
- 使用用户池进行身份验证 (p. 39)
- 使用 Amazon Cognito 用户群体 API 和用户群体端点 (p. 47)
- 更新用户群体配置 (p. 53)
- 使用 Amazon Cognito 托管 UI 进行注册和登录 (p. 54)
- 通过第三方添加用户池登录 (p. 85)
- 使用 Lambda 触发器自定义用户池工作流 (p. 111)
- 将 Amazon Pinpoint 分析与 Amazon Cognito 用户池结合使用 (p. 164)
- 管理用户池中的用户 (p. 167)
- Amazon Cognito 用户池的电子邮件设置 (p. 194)
- Amazon Cognito 用户池的短信设置 (p. 199)
- 将令牌与用户池结合使用 (p. 203)
- 在成功进行用户池身份验证后访问资源 (p. 215)
- 使用 Amazon Cognito 用户池安全功能 (p. 219)
- 用户池参考（AWS Management Console） (p. 243)

功能

Amazon Cognito 用户群体具有以下功能。

注册

Amazon Cognito 用户群体具有用户驱动、管理员驱动和编程方法，可将用户配置文件添加到您的用户群体。Amazon Cognito 用户群体支持以下注册模式。您可以在应用程序中使用这些模型的任意组合。

1. 用户可以在应用程序中输入其信息，并创建用户群体原生的用户配置文件。您可以调用 API 注册操作在用户群体中注册用户。您可以向任何人开放这些注册操作，也可以使用客户端密钥或 AWS 凭证对他们进行授权。
2. 您可以将用户重定向到第三方 IdP，他们可以授权该第三方 IdP 将其信息传递给 Amazon Cognito。Amazon Cognito 将 OIDC ID 令牌、OAuth 2.0 `userInfo` 数据和 SAML 2.0 断言处理到用户群体的用户配置文件中。您可以根据属性映射规则控制您希望 Amazon Cognito 接收的属性。
3. 您可以跳过公共注册或联合身份验证注册，并根据自己的数据来源和模式来创建用户。在 Amazon Cognito 控制台或 API 中直接添加用户。从 CSV 文件导入用户。运行即时 AWS Lambda 函数，此函数在现有目录中查找您的新用户，并从现有数据填充用户配置文件。

用户注册后，您可以将他们添加到 Amazon Cognito 在访问令牌和 ID 令牌中列出的组。在将 ID 令牌传递给身份池时，还可以将用户群体组链接到 IAM 角色。

相关主题

- 管理用户池中的用户 (p. 167)
- 使用 Amazon Cognito 用户群体 API 和用户群体端点 (p. 47)
- 使用 AWS SDK 的 Amazon Cognito 身份提供者的代码示例 (p. 407)

登录

Amazon Cognito 可以是应用程序的独立用户目录和身份提供者（IdP）。您的用户可以使用由 Amazon Cognito 托管的 UI 登录，也可以通过 Amazon Cognito 用户群体 API 使用您自己的 UI 登录。前端自定义 UI 后面的应用程序层可以使用多种方法中的任何一种来授权后端的请求，以确认有效的请求。

要使用外部目录（可选择与 Amazon Cognito 内置的用户目录结合使用）登录用户，您可以添加以下集成。

1. 使用 OAuth 2.0 社交登录进行登录并导入使用者用户数据。Amazon Cognito 支持通过 OAuth 2.0 登录 Google、Facebook、Amazon 和 Apple。
2. 使用 SAML 和 OIDC 登录进行登录并导入企业用户数据。您也可以将 Amazon Cognito 配置为接受来自任何 SAML 或 OpenID Connect（OIDC）身份提供者（IdP）的声明。
3. 将外部用户配置文件链接到原生用户配置文件。关联的用户可以使用第三方用户身份登录，并获得您分配给内置目录中的用户的访问权限。

相关主题

- 通过第三方添加用户池登录 (p. 85)
- 将联合用户与现有用户配置文件关联 (p. 109)

机器对机器授权

有些会话不是人机互动。您可能需要一个能够通过自动化流程向 API 授权请求的服务账户。要为具有 OAuth 2.0 范围的机器对机器授权生成访问令牌，您可以添加用于生成客户端凭证授权的应用程序客户端。

相关主题

- 为您的用户池定义资源服务器 (p. 82)

托管 UI

如果您不想构建用户界面，则可以向用户提供自定义 Amazon Cognito 托管 UI。托管 UI 是一组用于注册、登录、多重身份验证（MFA）和密码重置的网页。您可以将托管 UI 添加到现有域中，或者在 AWS 子域中使用前缀标识符。

相关主题

- 使用 Amazon Cognito 托管 UI 进行注册和登录 (p. 54)
- 配置用户池域 (p. 61)

安全性

您的本地用户可以通过 SMS 消息中的代码或用于生成多重身份验证（MFA）代码的应用程序，提供额外的身份验证要素。您可以构建在应用程序中设置和处理 MFA 的机制，也可以让托管 UI 管理 MFA。当您的用户从可信设备登录时，Amazon Cognito 用户群体可以绕过 MFA。

如果您不想最初就要求用户提供 MFA，则可以有条件地提出要求。借助高级安全功能，Amazon Cognito 可以检测潜在的恶意活动，并要求用户设置 MFA 或阻止登录。

如果用户群体的网络流量可能是恶意的，则可以对其进行监控并使用 AWS WAF Web ACL 采取措施。

相关主题

- 向用户池添加 MFA (p. 220)
- 向用户池添加高级安全 (p. 227)
- 将 AWS WAF Web ACL 与用户池相关联 (p. 236)

自定义客户体验

在用户注册、登录或配置文件更新的大多数阶段，您可以自定义 Amazon Cognito 处理请求的方式。使用 Lambda 触发器，您可以根据自定义条件修改 ID 令牌或拒绝注册请求。您可以创建自己的自定义身份验证流程。

您可以上传自定义 CSS 和徽标，为用户提供熟悉的托管 UI 外观。

相关主题

- 使用 Lambda 触发器自定义用户池工作流 (p. 111)
- 自定义身份验证质询 Lambda 触发器 (p. 134)
- 自定义内置登录网页和注册网页 (p. 67)

监控和分析

Amazon Cognito 用户群体将 API 请求（包括对托管 UI 的请求）记录到 AWS CloudTrail。您可以在 Amazon CloudWatch Logs 中查看性能指标，使用 Lambda 触发器将自定义日志推送到 CloudWatch，以及在服务限额控制台中监控 API 请求量。

还可以将 API 请求中的设备和会话数据记录到 Amazon Pinpoint 活动中。借助 Amazon Pinpoint，您可以根据对用户活动的分析，从应用程序发送推送通知。

相关主题

- 使用 AWS CloudTrail 记录 Amazon Cognito API 调用 (p. 513)

- 跟踪 CloudWatch 和 Service Quotas 中的配额和使用情况 (p. 507)
- 将 Amazon Pinpoint 分析与 Amazon Cognito 用户池结合使用 (p. 164)

Amazon Cognito 身份池集成

Amazon Cognito 的另一半是身份池。身份池提供凭证，用于授权和监控用户对 AWS 服务（例如 Amazon DynamoDB 或 Amazon S3）的 API 请求。您可以构建基于身份的访问策略，根据您在用户群体中对用户进行分类的方式来保护您的数据。身份池还可以接受来自各种身份提供者的令牌和 SAML 2.0 断言，与用户群体身份验证无关。

相关主题

- 在登录后使用身份池访问 AWS 服务 (p. 217)
- Amazon Cognito 身份池（联合身份） (p. 288)

用户池入门

首次使用 Amazon Cognito 控制台按照以下步骤设置和配置用户群体。使用本指南开始测试 Amazon Cognito 的功能。这些过程为新用户群体建议原定设置，以便您开始使用。有关更多信息，请参阅Amazon Cognito 入门 (p. 10)。

主题

- 第 1 步 创建用户池 (p. 30)
- 第 2 步 添加应用程序客户端并设置托管 UI (p. 31)
- 第 3 步 向用户池添加社交登录（可选） (p. 33)
- 第 4 步 将使用 SAML 身份提供商的登录添加到用户池（可选） (p. 37)
- 后续步骤 (p. 39)

第 1 步 创建用户池

使用 Amazon Cognito 用户池，您可以创建和维护用户目录，并将注册信息和登录信息添加到移动应用程序或 Web 应用程序。

创建用户池

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 在页面右上角，选择 Create a user pool（创建用户池）以开启用户池创建向导。
4. 在 Configure sign-in experience（配置登录体验）中，选择要用于此用户池的联合提供商。有关更多信息，请参阅通过第三方添加用户池登录。

   Note

   默认情况下，Make user name case sensitive（使用户名区分大小写）选项已关闭。我们建议您不要激活此选项。当用户名不区分大小写时，用户可以使用 **`username`** 或 **`Username`** 登录。Make user name case sensitive（使用户名区分大小写）选项还控制 `preferred_username` 和 `email` 别名的区分大小写设置。当用户名区分大小写时，您必须采取额外的安全预防措施。有关更多信息，请参阅用户池区分大小写 (p. 238)。
5. 在 Configure security requirements（配置安全要求）中，选择密码策略、多重身份验证（MFA）要求以及用户账户恢复选项。有关更多信息，请参阅 Amazon Cognito 中的安全性。

6. 在 Configure sign-up experience（配置注册体验）中，确定新用户在注册时如何验证其身份，以及在用户注册流程中应该要求哪些属性或可选属性。有关更多信息，请参阅管理用户池中的用户。
7. 在 Configure message delivery（配置消息传输）中，配置与 Amazon Simple Email Service (Amazon SES) 和 Amazon Simple Notification Service (Amazon SNS) 的集成，以便向用户发送电子邮件和 SMS 消息以进行注册、账户确认、MFA 和账户恢复。有关更多信息，请参阅 Amazon Cognito 用户池的电子邮件设置和 Amazon Cognito 用户池的 SMS 消息设置。
8. 在 Integrate your app（集成应用程序）中，命名用户池、配置托管 UI 并创建应用程序客户端。有关更多信息，请参阅添加应用程序以启用托管 Web UI。
9. 在 Review and create（审核和创建）屏幕中，查看您的选择并修改任意您希望的选择。如果您对用户池配置感到满意，请选择 Create user pool（创建用户池）继续执行。

下一个步骤

第 2 步 添加应用程序客户端并设置托管 UI (p. 31)

第 2 步 添加应用程序客户端并设置托管 UI

在创建用户池后，您可以创建一个应用程序，以便使用内置网页来对用户进行注册和登录。

在用户池中创建应用程序

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。如果创建新的用户池，系统将在向导期间提示您设置应用程序客户端并配置托管 UI。
4. 选择用户池的 App integration（应用程序集成）选项卡。
5. 选择在 Domain（域）旁边的 Actions（操作），然后选择 Create custom domain（创建自定义域）或 Create Cognito domain（创建 Cognito 域）。如果您已配置用户池域，请先选择 Delete Cognito domain（删除 Cognito 域）或 Delete custom domain（删除自定义域），然后再创建新的自定义域。
6. 输入可用的域前缀，将其与 Cognito domain（Cognito 域）结合使用。有关设置 Custom domain（自定义域）的信息，请参阅将您自己的域用于托管 UI。
7. 选择 Create（创建）。
8. 导航回用于同一用户池的 App integration（应用程序集成）选项卡，然后查找 App clients（应用程序客户端）。选择 Create an app client（创建应用程序客户端）。
9. 选择 Application type（应用程序类型）。系统将根据您的选择提供一些建议的设置。使用托管 UI 的应用程序是 Public client（公有客户端）。
10. 输入 App client name（应用程序客户端名称）。
11. 在此练习中，选择 Don't generate client secret（不生成客户端密钥）。客户端密钥由机密应用程序使用，该应用程序由集中式应用程序对用户进行身份验证。在此练习中，您将向用户展示托管 UI 登录页面，无需客户端密钥。
12. 选择您将允许用于您的应用程序的 Authentication flows（身份验证流程）。确保已选择 `USER_SRP_AUTH`。
13. 根据需要自定义 token expiration（令牌的到期时间）、Advanced security configuration（高级安全配置）和 Attribute read and write permissions（属性读取和写入权限）。有关更多信息，请参阅配置应用程序客户端设置。
14. 为您的应用程序客户端 Add a callback URL（添加回调 URL）。在托管 UI 身份验证后，您将被引导到该地址。在您能够在应用程序中实施注销之前，您无需添加 Allowed sign-out URL（允许的注销 URL）。

    对于 iOS 或 Android 应用程序，您可以使用类似 `myapp://` 的回调 URL。
15. 为应用程序客户端选择 Identity providers（身份提供商）。至少，启用 Cognito user pool（Cognito 用户池）作为提供商。

Note

要使用外部身份提供者 (IdP)（如 Facebook、Amazon、Google 和 Apple）以及通过 OpenID Connect (OIDC) 或 SAML IdP 进行登录，请首先按照通过第三方添加用户群体登录中所示对其进行配置，然后返回到 App client settings（应用程序客户端设置）页面以启用它们。

16. 选择 OAuth 2.0 Grant Types（OAuth 2.0 授予类型）。选择 Authorization code grant（授权代码授予）以返回随后更换为用户池令牌的授权代码。由于令牌绝不会直接向终端用户公开，因此它们不太可能被泄露。但是，后端需要自定义应用程序以将授权代码换成用户池令牌。出于安全原因，对于移动应用程序，强烈建议您将授权代码授予流程与代码交换的证明密钥（PKCE）一起使用。

    选择 Implicit grant (隐式授予) 以便从 Amazon Cognito 将用户池 JSON Web 令牌（JWT）返回给您。当没有可用于将授权代码更换为令牌的后端时，您可以使用此流程。它对于调试令牌也很有帮助。

    Note

    您可以同时启用 Authorization code grant（授权代码授予）和 Implicit code grant（隐式代码授予），然后按需使用每个授予。
    只有在您的应用程序需要代表自己而不是代表用户请求访问令牌时才选择 Client credentials（客户端凭证）。
17. 除非您希望明确地排除一个项目，否则请选择所有 OpenID Connect scopes（OpenID Connect 范围）。
18. 选择您已配置的任意 Custom scopes（自定义范围）。自定义范围通常用于机密客户端。
19. 选择 Create（创建）。

查看您的登录页面

在您的 App client（应用程序客户端）页面上，选择 View hosted UI（查看托管 UI）打开一个新的浏览器选项卡，以转至预先填充了应用程序客户端 ID、范围、授权和回调 URL 参数的登录页面。

您可以使用以下 URL 手动查看托管 UI 登录网页。记下 `response_type`。在此示例中，response_type=code 表示授权代码授予。

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

您可以使用针对隐式代码授予的以下 URL 查看托管 UI 登录网页，其中，response_type=token。成功登录后，Amazon Cognito 会将用户池令牌返回到您的 Web 浏览器的地址栏。

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

您可以在响应中的 `#idtoken=` 参数后面找到 JSON Web Token (JWT) 身份令牌。

以下 URL 是来自隐式授予请求的示例响应。您的身份令牌字符串长得多。

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

使用 RS256 算法对 Amazon Cognito 用户池令牌进行签名。您可以使用 AWS Lambda 解码和验证用户池令牌，请参阅 AWS GitHub 网站上的 Decode and verify Amazon Cognito JWT tokens。

您的域将显示在 Domain name（域名）页面上。您的应用程序客户端 ID 和回调 URL 将显示在 General settings（常规设置）页面上。如果您在控制台中所做的更改没有立即显示，请等待几分钟，然后刷新浏览器。

下一步

第 3 步 向用户池添加社交登录（可选）(p. 33)

第 3 步 向用户池添加社交登录（可选）

您可以允许应用程序用户通过社交身份提供商（IdP）（如 Facebook、Google、亚马逊和 Apple）进行登录。无论您的用户是直接登录还是通过第三方登录，所有用户都在用户池中有一个配置文件。如果您不想添加通过社交登录身份提供商进行的登录，请跳过此步骤。

步骤 1：向社交 IdP 注册

在使用 Amazon Cognito 创建社交 IdP 之前，必须向社交 IdP 注册应用程序才能接收客户端 ID 和客户端密钥。

向 Facebook 注册应用程序

1. 创建 Facebook 开发人员账户。
2. 使用 Facebook 凭证登录。
3. 在 My Apps（我的应用程序）菜单上，选择 Create New App（创建新的应用程序）。
4. 输入 Facebook 应用程序的名称，然后选择 Create App ID（创建应用程序 ID）。
5. 在左侧导航栏上，选择 Settings（设置），然后选择 Basic（基本）。
6. 记下 App ID（应用程序 ID）和 App Secret（应用程序密钥）。您将在下一节中使用它们。
7. 从页面底部选择 + Add Platform（+ 添加平台）。
8. 选择 Save changes（保存更改）。
9. 为 App Domains（应用程序域）输入用户池域。

```
https://your_user_pool_domain
```

10. 选择 Save changes（保存更改）。
11. 从导航栏中，选择 Products（产品），然后选择 Facebook Login（Facebook 登录）中的 Set up（设置）。
12. 从导航栏中，选择 Facebook Login（Facebook 登录），然后选择 Settings（设置）。

    在 Valid OAuth Redirect URIs（有效的 OAuth 重定向 URI）中输入重定向 URL。重定向 URL 将包含具有 `/oauth2/idpresponse` 端点的用户池域。

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

13. 选择 Save changes（保存更改）。

向 Amazon 注册应用程序

1. 创建 Amazon 开发人员账户。
2. 使用 Amazon 凭证登录。
3. 您需要创建一个 Amazon 安全配置文件才能接收 Amazon 客户端 ID 和客户端密钥。

   从页面顶部的导航栏中选择 Apps and Services（应用程序和服务），然后选择 Login with Amazon。
4. 选择 Create a Security Profile（创建安全配置文件）。
5. 输入 Security Profile Name（安全配置文件名称）、Security Profile Description（安全配置文件描述）和 Consent Privacy Notice URL（同意隐私声明 URL）。
6. 选择 Save（保存）。

7. 选择 Client ID（客户端 ID）和 Client Secret（客户端密钥）以显示客户端 ID 和密钥。您将在下一节中使用它们。
8. 将鼠标悬停在齿轮图标上并选择 Web Settings（Web 设置），然后选择 Edit（编辑）。
9. 将用户池域输入到 Allowed Origins（允许的源）中。

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

10. 将具有 `/oauth2/idpresponse` 端点的用户池域输入到 Allowed Return URLs（允许的返回 URL）中。

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

11. 选择 Save（保存）。

向 Google 注册应用程序

有关 Google Cloud 平台中的 OAuth 2.0 的更多信息，请参阅 Google Workspace for Developers 文档中的了解身份验证和授权。

1. 创建 Google 开发人员账户。
2. 登录到 Google Cloud Platform 控制台。
3. 从顶部导航栏中，选择 Select a project（选择项目）。如果您在 Google 平台中已经有项目，则此菜单会改为显示您的原定设置项目。
4. 选择 NEW PROJECT（新建项目）。
5. 输入产品的名称，然后选择 CREATE（创建）。
6. 在左侧导航栏上，选择 APIs and Services（API 和服务），然后选择 Oauth consent screen（Oauth 同意屏幕）。
7. 输入应用程序信息，App domain（应用程序域）、Authorized domains（已授权的域）和 Developer contact information（开发人员联系信息）。例如，您的 Authorized domains（已授权的域）必须包括 `amazoncognito.com` 和自定义域的根，例如 `example.com`。选择 SAVE AND CONTINUE（保存并继续）。
8. 1. 在 Scopes（范围）下，选择 Add or remove scopes（添加或删除范围），然后至少选择以下 OAuth 范围。

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

9. 在 Test users（测试用户）下，选择 Add users（添加用户）。输入您的电子邮件地址和任何其他授权测试用户，然后选择 SAVE AND CONTINUE（保存并继续）。
10. 再次展开左侧导航栏，选择 APIs and Services（API 和服务），然后选择 Credentials（凭证）。
11. 依次选择 CREATE CREDENTIALS（创建凭证）和 OAuth client ID（OAuth 客户端 ID）。
12. 选择 Application type（应用程序类型）并为客户端提供 Name（名称）。
13. 在 Authorized JavaScript origins（已授权的 JavaScript 源）下，选择 ADD URI（添加 URI）。输入用户群体域。

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

14. 在 Authorized redirect URIs（已授权的重新导向 URI）下，选择 ADD URI（添加 URI）。输入指向用户群体域的 `/oauth2/idpresponse` 端点的路径。

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

15. 选择 CREATE（创建）。

16. 安全地存储 Google 在 Your client ID（您的客户端 ID）和 Your client secret（您的客户端密钥）下显示的值。当您添加 Google IdP 时，请向 Amazon Cognito 提供这些值。

向 Apple 注册应用程序

有关设置“通过 Apple 登录”的更多信息，请参阅 Apple 开发人员文档中的配置环境以通过 Apple 登录。

1. 创建 Apple 开发人员账户。
2. 使用 Apple 凭证登录。
3. 在左侧导航栏上，选择 Certificates, Identifiers & Profiles（证书、标识符和配置文件）。
4. 在左侧导航栏上，选择 Identifiers (标识符)。
5. 在 Identifiers (标识符) 页面上，选择 + 图标。
6. 在 Register a New Identifier (注册新标识符) 页面上，选择 App IDs (应用程序 ID)，然后选择 Continue (继续)。
7. 在 Select a type（选择类型）页面上，选择 App（应用程序），然后选择 Continue（继续）。
8. 在 Register an App ID（注册应用程序 ID）页面上，执行以下操作：
    1. 在 Description（说明）下，输入说明。
    2. 在 App ID Prefix（应用程序 ID 前缀）下，输入 Bundle ID（捆绑包 ID）。记下 App ID Prefix（应用程序 ID 前缀）下的值。在步骤 2：将社交 IdP 添加到用户池 (p. 90)中选择 Apple 作为身份提供商后，您将使用此值。
    3. 在 Capabilities（功能）下，选择 Sign In with Apple，然后选择 Edit（编辑）。
    4. 在 Sign in with Apple: App ID Configuration（通过 Apple 登录：应用程序 ID 配置）页面上，选择将应用程序设置为主应用程序或与其他应用程序 ID 分组在一起，然后选择 Save（保存）。
    5. 选择 Continue（继续）。
9. 在 Confirm your App ID（确认您的应用程序 ID）页面上，选择 Register（注册）。
10. 在 Identifiers (标识符) 页面上，选择 + 图标。
11. 在 Register a New Identifier（注册新标识符）页面上，选择 Services IDs（服务 ID），然后选择 Continue（继续）。
12. 在 Register an Services ID（注册服务 ID）页面上，执行以下操作：
    1. 在 Description（描述）下方，键入描述。
    2. 在 Identifier（标识符）下方，键入标识符。记下此服务 ID，因为在步骤 2：将社交 IdP 添加到用户池 (p. 90)中选择 Apple 作为身份提供商后需要此值。
    3. 选择 Continue（继续），然后选择 Register（注册）。
13. 从 Identifiers（标识符）页中选择您刚创建的 Services ID（服务 ID）。
    1. 选择 Sign In with Apple（使用苹果账号登录），然后选择 Configure（配置）。
    2. 在 Web Authentication Configuration（Web 身份验证配置）页上，选择您先前创建的应用程序 ID 作为 Primary App ID（主应用程序 ID）。
    3. 在 Website URLs（网站 URL）旁边选择 + 图标。
    4. 在 Domains and subdomains（域名和子域）下，输入不带 `https://` 前缀的用户群体域。

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

    5. 在 Return URLs（返回 URL）下，输入指向用户群体域的 `/oauth2/idpresponse` 端点的路径。

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

    6. 选择 Next（下一步），然后选择 Done（完成）。您不需要验证域。
    7. 选择 Continue（继续），然后选择 Save（保存）。

14. 在左侧导航栏上，选择 Keys (密钥)。
15. 在 Keys（密钥）页面上，选择 + 图标。
16. 在 Register a New Key（注册新密钥）页面上，执行以下操作：
    1. 在 Key Name（密钥名称）下，输入密钥名称。
    2. 选择 Sign In with Apple，然后选择 Configure（配置）。
    3. 在 Configure Key（配置密钥）页上，选择您先前创建的应用程序 ID 作为 Primary App ID（主应用程序 ID）。选择 Save（保存）。
    4. 选择 Continue（继续），然后选择 Register（注册）。
17. 在 Download Your Key（下载您的密钥）页面上，选择 Download（下载）以下载私有密钥并记下显示的 Key ID（密钥 ID），然后选择 Done（完成）。在步骤 2：将社交 IdP 添加到用户池 (p. 90)中选择 Apple 作为身份提供商后，您将需要此私有密钥和在此页面上显示的 Key ID（密钥 ID）值。

步骤 2：将社交 IdP 添加到用户池

在本节中，您使用上一节中的客户端 ID 和客户端密钥在用户池中配置社交 IdP。

使用 AWS Management Console配置用户池社交身份提供商

1. 转到 Amazon Cognito 控制台。系统可能会提示您输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 Sign-in experience（登录体验）选项卡。找到 Federated sign-in（联合登录），然后选择 Add an identity provider（添加身份提供商）。
5. 选择一个社交身份提供商：Facebook、Google、Login with Amazon 或 Sign in with Apple。
6. 根据您选择的社交身份提供商，从以下步骤中进行选择：
   - Google 和 Login with Amazon – 输入在上一部分中生成的 app client ID（应用程序客户端 ID）和 app client secret（应用程序客户端密钥）。
   - Facebook – 输入在上一部分中生成的 app client ID（应用程序客户端 ID）和 app client secret（应用程序客户端密钥），然后选择 API 版本（例如，版本 2.12）。我们建议选择最新的可用版本，因为每个 Facebook API 版本都有一个生命周期和一个弃用日期。Facebook 的范围和属性可能因 API 版本而异。我们建议您使用 Facebook 测试您的社交身份登录，以确保联合会按预期运行。
   - Sign in with Apple（通过 Apple 登录）– 输入在上一部分中生成的 Services ID（服务 ID）、Team ID（团队 ID）、Key ID（密钥 ID）和 private key（私有密钥）。
7. 输入要使用的 Authorized scopes（授权范围）的名称。范围定义了您要通过应用程序访问的用户属性（如 `name` 和 `email`）。对于 Facebook，这些属性应用逗号分隔。对于 Google 和 Login with Amazon，则应采用空格分隔。对于 Sign in with Apple，选中要访问的范围的复选框。

社交身份提供商	示例范围
Facebook	`public_profile, email`
Google	`profile email openid`
Login with Amazon	`profile postal_code`
Sign in with Apple	`email name`

您的应用程序用户需要同意向您的应用程序提供这些属性。关于社交服务提供商范围的更多信息，请参阅 Google、Facebook 和 Login with Amazon 或 Sign in with Apple 的文档。

对于 Sign in with Apple，下面提供了可能不会返回范围的用户场景：

- 终端用户离开 Apple 登录页面后出现故障（可能来自 Amazon Cognito 内部的故障或开发人员编写的任何内容）
- 跨用户池和/或其它身份验证服务使用服务 ID 标识符
- 开发人员在用户登录后添加额外的范围。用户仅在进行身份验证和刷新令牌时检索新信息。
- 开发人员删除用户，然后用户再次登录，而没有从其 Apple ID 个人资料中删除该应用程序

8. 请将身份提供商的属性映射到您的用户池。有关更多信息，请参阅指定适用于用户池的身份提供程序属性映射。
9. 选择 Create（创建）。
10. 从 App client integration（应用程序客户端集成）选项卡上的列表中选择一个 App clients（应用程序客户端），然后选择 Edit hosted UI settings（编辑托管 UI 设置）。将新的社交身份提供商添加到 Identity providers（身份提供商）下的应用程序客户端。
11. 选择 Save changes（保存更改）。

步骤 3：测试社交 IdP 配置

可以通过使用前两节中的元素来创建登录 URL。使用此 URL 测试社交 IdP 配置。

```
https://<your_user_pool_domain>/login?
response_type=code&client_id=<your_client_id>&redirect_uri=https://www.example.com
```

您可以在用户池 Domain name（域名）控制台页上找到您的域。client_id 位于 App client settings（应用程序客户端设置）页上。对于 redirect_uri 参数，使用您的回调 URL。这是页面的 URL，在页面中，您的用户在身份验证成功后将被重定向。

Note

Amazon Cognito 会取消未在 5 分钟内完成的身份验证请求，并将用户重定向到托管 UI。页面随即显示 `Something went wrong` 错误消息。

下一步

第 4 步 将使用 SAML 身份提供商的登录添加到用户池（可选）(p. 37)

第 4 步 将使用 SAML 身份提供商的登录添加到用户池（可选）

您可以允许您的应用程序用户通过 SAML 身份提供商（IdP）进行登录。无论您的用户是直接登录还是通过第三方登录，所有用户都在用户池中有一个配置文件。如果您不想添加通过 SAML 身份提供商进行的登录，请跳过此步骤。

您需要更新您的 SAML 身份提供商并配置您的用户池。请参阅 SAML 身份提供商的文档，了解有关如何将您的用户池添加为 SAML 2.0 身份提供商的依赖方或应用程序的信息。

请参阅 SAML 身份提供商的文档，了解有关如何将您的用户池添加为 SAML 2.0 身份提供商的依赖方或应用程序的信息。在用户群体域中为您的 SAML 身份提供者中的 SAML 2.0 POST 绑定配置以下端点。有关用户群体域的更多信息，请参阅 配置用户池域 (p. 61)。

```
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 控制台的 Domain name（域名）选项卡上找到用户池的域前缀和区域值。

对于一些 SAML 身份提供商，您还需要采用以下格式提供 SP urn/受众 URI/SP 实体 ID：

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

您可以在 Amazon Cognito 控制台的 General settings（常规设置）选项卡上找到用户池 ID。

您还应配置 SAML 身份提供商，以便为您的用户池中需要的任何属性提供属性值。通常情况下，email 是用户池的必需属性。在这种情况下，SAML 身份提供商应在 SAML 断言中提供一个 email 值（声明）。

Amazon Cognito 用户池支持 SAML 2.0 与 POST 绑定端点联合。这使您的应用程序不必检索或分析 SAML 断言响应，因为用户池直接通过用户代理从身份提供商接收 SAML 响应。

在您的用户池中配置 SAML 2.0 身份提供商

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 Sign-in experience（登录体验）选项卡。找到 Federated sign-in（联合登录），然后选择 Add an identity provider（添加身份提供商）。
5. 选择 SAML 社交身份提供商。
6. 输入以逗号分隔的 Identifiers（标识符）。标识符将告知 Amazon Cognito 应该检查用户登录时输入的电子邮件地址，然后将它们引导到与其域名对应的提供商。
7. 如果您希望 Amazon Cognito 在用户注销时向您的提供商发送已签名的注销请求，请选择 Add sign-out flow（添加注销流程）。您必须配置 SAML 2.0 身份提供商以向配置托管 UI 时创建的 `https://<your Amazon Cognito domain>/saml2/logout` 端点发送注销响应。此 `saml2/logout` 端点使用 POST 绑定。

   Note

   如果选择此选项，并且您的 SAML 身份提供商需要已签名的注销请求，则您还需要为您的 SAML IdP 配置 Amazon Cognito 提供的签名证书。
   SAML IdP 将处理已签名的注销请求并从 Amazon Cognito 会话中注销您的用户。
8. 选择 Metadata document source（元数据文档源）。如果您的身份提供商在公有 URL 上提供 SAML 元数据，则可以选择 Metadata document URL（元数据文档 URL），然后输入该公有 URL。否则，请选择 Upload metadata document（上载元数据文档），然后选择您之前从提供商下载的元数据文件。

   Note

   我们建议您输入元数据文档 URL（如果您的提供商具有公有端点），而不是上载文件，因为这将允许 Amazon Cognito 自动刷新元数据。通常，元数据刷新操作每 6 小时执行一次或在元数据过期前执行（以时间较早者为准）。
9. 选择 Map attributes between your SAML provider and your app（在 SAML 提供商和应用程序之间映射属性）将 SAML 提供程序属性映射到用户池中的用户配置文件。在属性映射中包含用户池必需属性。

   例如，当您选择 User pool attribute（用户池属性）email 时，按照您的身份提供商提供的 SAML 断言中显示的内容，输入 SAML 属性名称。您的身份提供商可能会提供示例 SAML 断言以供参考。一些身份提供商使用简单名称（如 email），另一些则使用以下示例 URL 格式的属性名称：

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

10. 选择 Create（创建）。

有关更多信息，请参阅向用户池添加 SAML 身份提供商 (p. 92)。

后续步骤

现在您已创建用户池，您可以在 Amazon Cognito 中探索更多功能。

深入了解以下用户池功能：

- 自定义内置登录网页和注册网页 (p. 67)
- 向用户池添加 MFA (p. 220)
- 向用户池添加高级安全 (p. 227)
- 使用 Lambda 触发器自定义用户池工作流 (p. 111)
- 将 Amazon Pinpoint 分析与 Amazon Cognito 用户池结合使用 (p. 164)

有关 Amazon Cognito 身份验证和授权案例的概览，请参阅Amazon Cognito 常见场景 (p. 12)。

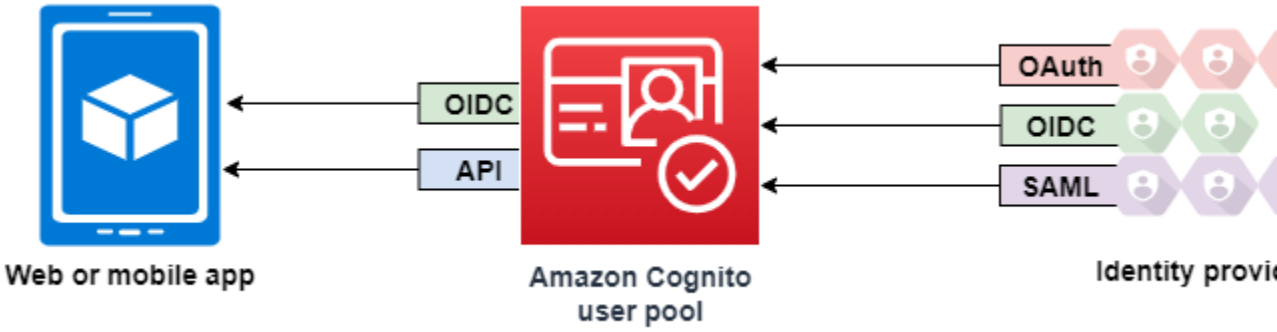
要在成功的用户池身份验证后访问其他 AWS 服务，请参阅 在登录后使用身份池访问 AWS 服务 (p. 217)。

除了 AWS Management Console和本节前面部分提到的用户池开发工具包之外，您还可以使用 AWS Command Line Interface与您的用户池配置文件进行交互。

使用用户池进行身份验证

您的应用程序用户可以通过用户池直接登录，也可以通过第三方身份提供商（IdP）联合。用户池管理处理以下令牌的开销：从通过 Facebook、Google、亚马逊和 Apple 进行的社交登录返回的令牌以及从 OpenID Connect（OIDC）和 SAML IdP 返回的令牌。

在成功验证身份后，Amazon Cognito 会将用户池令牌返回到您的应用程序。您可以使用这些令牌向您的用户授予对您的服务器端资源或 Amazon API Gateway 的访问权限。或者，您可以用它们交换用于访问其他AWS服务的AWS凭证。



您的 Web 或移动应用程序的用户池令牌处理和管理在客户端上通过 Amazon Cognito SDK 进行。同样的，如果存在有效的（非过期的）刷新令牌，Mobile SDK for iOS 和 Mobile SDK for Android 会自动刷新您的 ID 令牌和访问令牌，而且 ID 令牌和访问令牌剩余有效时间至少有 5 分钟。有关 SDK 以及适用于 JavaScript、Android 和 iOS 的示例代码的信息，请参阅 Amazon Cognito 用户池 SDK。

在您的应用程序用户成功登录后，Amazon Cognito 会创建会话并返回经过身份验证的用户的 ID 令牌、访问令牌和刷新令牌。

JavaScript

```
// Amazon Cognito creates a session which includes the id, access, and refresh tokens
 of an authenticated user.

var authenticationData = {
        Username : 'username',
        Password : 'password',
    };
    var authenticationDetails = new
 AmazonCognitoIdentity.AuthenticationDetails(authenticationData);
    var poolData = { UserPoolId : 'us-east-1_ExaMPle',
        ClientId : '1example23456789'
    };
    var userPool = new AmazonCognitoIdentity.CognitoUserPool(poolData);
    var userData = {
        Username : 'username',
        Pool : userPool
    };
    var cognitoUser = new AmazonCognitoIdentity.CognitoUser(userData);
    cognitoUser.authenticateUser(authenticationDetails, {
        onSuccess: function (result) {
            var accessToken = result.getAccessToken().getJwtToken();

            /* Use the idToken for Logins Map when Federating User Pools with identity
 pools or when passing through an Authorization Header to an API Gateway Authorizer */
            var idToken = result.idToken.jwtToken;
        },

        onFailure: function(err) {
            alert(err);
        },

});
```

Android

```
// Session is an object of the type CognitoUserSession, and includes the id, access,
 and refresh tokens for a user.

String idToken = session.getIdToken().getJWTToken();
String accessToken = session.getAccessToken().getJWT();
```

iOS - swift

```
// AWSCognitoIdentityUserSession includes id, access, and refresh tokens for a user.

- (AWSTask<AWSCognitoIdentityUserSession *> *)getSession;
```

iOS - objective-C

```
// AWSCognitoIdentityUserSession includes the id, access, and refresh tokens for a
 user.

[[user getSession:@"username" password:@"password" validationData:nil scopes:nil]
 continueWithSuccessBlock:^id _Nullable(AWSTask<AWSCognitoIdentityUserSession *> *
 _Nonnull task) {
    // success, task.result has user session
    return nil;
}];
```

主题

- 用户池身份验证流程 (p. 41)

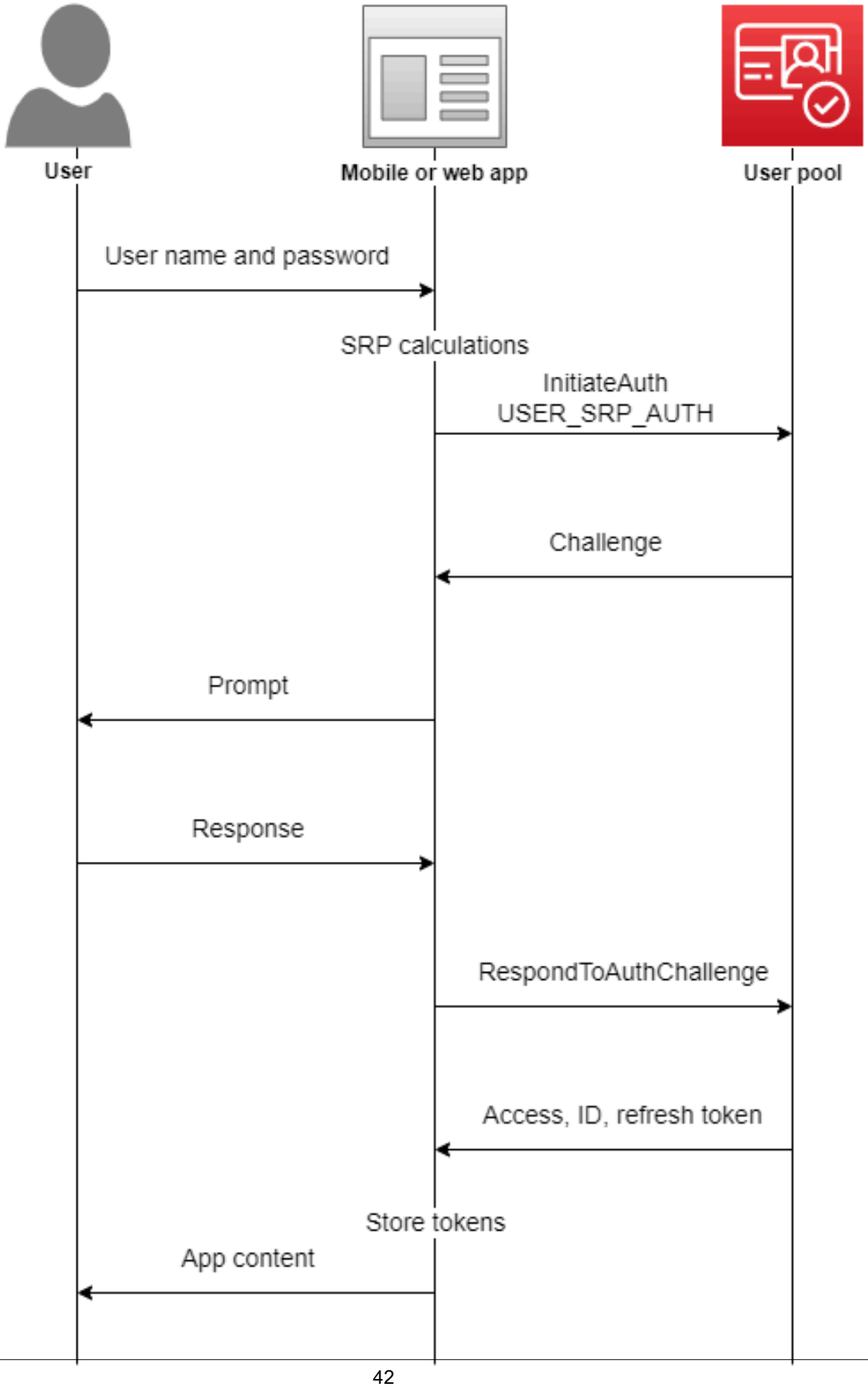
用户池身份验证流程

Amazon Cognito 包含多种对用户进行身份验证的方法。所有用户群体（无论您是否具有域）都可以在用户群体 API 中对用户进行身份验证。如果您向用户群体添加域，则可以使用用户群体端点。用户群体 API 支持针对 API 请求的各种授权模型和请求流程。

为了验证用户的身份，除了密码之外，Amazon Cognito 还支持包含新质询类型的身份验证流程。Amazon Cognito 身份验证通常要求您按以下顺序实施两个 API 操作：

1. `InitiateAuth`
2. `RespondToAuthChallenge`

用户通过应答连续的质询进行身份验证，直到身份验证失败或 Amazon Cognito 向用户发放令牌。您可以在包含不同质询的流程中对 Amazon Cognito 重复这些步骤，以支持任何自定义身份验证流程。


User
Mobile or web app
User pool
User name and password
SRP calculations
InitiateAuth
USER_SRP_AUTH
Challenge
Prompt
Response
RespondToAuthChallenge
Access, ID, refresh token
Store tokens
App content

通常，您的应用程序会生成一条提示，要求从用户收集信息，然后在 API 请求中将该信息提交给 Amazon Cognito。考虑使用用户池中的 `InitiateAuth` 流程，您已在其中为用户配置了多重身份验证 (MFA)。

1. 应用程序会提示您的用户输入用户名和密码。
2. 您可以将用户名和密码作为参数包含在 `InitiateAuth` 中。
3. Amazon Cognito 会返回 `SMS_MFA` 质询和会话标识符。
4. 应用程序会提示您的用户输入手机上收到的 MFA 代码。
5. 您将该代码和会话标识符包含在 `RespondToAuthChallenge` 请求中。

根据用户池的功能，您最终可能会在您的应用程序从 Amazon Cognito 检索令牌之前，响应 `InitiateAuth` 的几个质询。Amazon Cognito 在对每个请求的响应中都包含一个会话字符串。要将您的 API 请求合并到身份验证流程中，请在每个后续请求中包含来自上一个请求的响应中的会话字符串。默认情况下，在会话字符串过期之前，您的用户有三分钟时间完成每项质询。要调整此时段，请更改您的应用程序客户端的 Authentication flow session duration（身份验证流程会话持续时间）。以下过程介绍如何在应用程序客户端配置中更改此设置。

> **Note**
>
> 身份验证流程会话持续时间设置适用于使用 Amazon Cognito 用户群体 API 进行身份验证。Amazon Cognito 托管 UI 将多重身份验证的会话持续时间设置为 3 分钟，并将密码重置代码的会话持续时间设置为 8 分钟。

Amazon Cognito console

配置应用程序客户端身份验证流程会话持续时间 (AWS Management Console)

1. 在您的用户群体中的 App integration（应用程序集成）选项卡上，从 App clients and analytics（应用程序客户端和分析）容器中选择您的应用程序客户端的名称。
2. 在应用程序客户端信息容器中选择编辑。
3. 将 Authentication flow session duration（身份验证流程会话持续时间）的值更改为 SMS MFA 代码所需的有效期，以分钟为单位。这还会更改任何用户在您的应用程序客户端中必须完成任何身份验证质询的时间。
4. 选择 Save changes（保存更改）。

Amazon Cognito API

配置应用程序客户端身份验证流程会话持续时间 (Amazon Cognito API)

1. 使用 `DescribeUserPoolClient` 请求中您的现有用户群体设置准备 `UpdateUserPoolClient` 请求。您的 `UpdateUserPoolClient` 请求必须包含所有现有的应用程序客户端属性。
2. 将 `AuthSessionValidity` 的值更改为 SMS MFA 代码所需的有效期，以分钟为单位。这还会更改任何用户在您的应用程序客户端中必须完成任何身份验证质询的时间。

有关应用程序客户端的更多信息，请参阅配置用户池应用程序客户端 (p. 258)。

您可以使用 AWS Lambda 触发器来自定义用户进行身份验证的方式。作为身份验证流程的一部分，这些触发器将发布并验证自己的质询。

您还可以在安全后端服务器上对用户使用管理员身份验证流程。您可以使用用户迁移身份验证流来进行用户迁移，而无需用户重置其密码。

失败登录尝试的 Amazon Cognito 锁定行为

在使用密码五次登录尝试失败后，Amazon Cognito 会将您的用户锁定一秒钟。然后，每多一次失败的尝试，锁定持续时间将增加一倍，最长约为 15 分钟。在锁定期内进行的尝试会产生 `Password attempts exceeded` 异常，不会影响后续锁定期的持续时间。对于累计 n 次的失败登录尝试（不包括 `Password`

attempts exceeded 异常），Amazon Cognito 会将您的用户锁定 2^(n-5) 秒。要将锁定重置为其初始状态，您的用户在锁定后连续 15 分钟的任何时间内都不得发起任何登录尝试。此行为随时可能会发生变化。此行为不适用于自定义质询，除非它们还执行基于密码的身份验证。

主题

- 客户端身份验证流程 (p. 44)
- 服务器端身份验证流程 (p. 44)
- 自定义身份验证流程 (p. 45)
- 内置身份验证流程和质询 (p. 45)
- 自定义身份验证流程和质询 (p. 45)
- 在自定义身份验证流程中使用 SRP 密码验证 (p. 46)
- 管理员身份验证流程 (p. 46)
- 用户迁移身份验证流程 (p. 47)

客户端身份验证流程

以下过程适用于您使用 AWS Amplify 或者 AWS SDK 创建的用户客户端应用程序：

1. 用户将他们的用户名和密码输入到应用程序中。
2. 应用程序使用用户的用户名和安全远程密码（SRP）详细信息调用 InitiateAuth 操作。

   此 API 操作返回身份验证参数。

   Note

   该应用程序使用 AWS 开发工具包内置的 Amazon Cognito SRP 功能生成 SRP 详细信息。
3. 应用程序调用 RespondToAuthChallenge 操作。如果调用成功，则 Amazon Cognito 返回用户的令牌，并且身份验证流程完成。

   如果 Amazon Cognito 需要另一个质询，则对 RespondToAuthChallenge 的调用不返回任何令牌。相反，调用会返回一个会话。
4. 如果 RespondToAuthChallenge 返回一个会话，应用程序将再次调用 RespondToAuthChallenge，这次使用会话和质询响应（例如，MFA 代码）。

服务器端身份验证流程

如果您没有用户应用程序，而是使用 Java、Ruby 或 Node.js 安全后端或服务器端应用程序，则可以对 Amazon Cognito 用户池使用经过身份验证的服务器端 API。

对于服务器端应用程序，用户池身份验证与客户端应用程序的身份验证类似，但以下情况除外：

- 服务器端应用程序调用 AdminInitiateAuth API 操作（而不是 InitiateAuth）。此操作需要 AWS 凭证，此凭证拥有的权限包括 cognito-idp:AdminInitiateAuth 和 cognito-idp:AdminRespondToAuthChallenge。该操作返回必需的身份验证参数。
- 服务器端应用程序获得身份验证参数后，它就会调用 AdminRespondToAuthChallenge API 操作（而不是 RespondToAuthChallenge）。AdminRespondToAuthChallenge API 操作仅在您提供了 AWS 凭证时才会成功。

有关使用 AWS 凭证对 Amazon Cognito API 请求签名的更多信息，请参阅《AWS 一般参考》中的签名版本 4 签名过程。

AdminInitiateAuth 和 AdminRespondToAuthChallenge API 操作无法接受用于管理员登录的用户名和密码用户凭证，除非您通过执行以下操作之一明确启用它们：

- 调用 CreateUserPoolClient 或 UpdateUserPoolClient 时，在 ExplicitAuthFlow 参数中包含 ALLOW_ADMIN_USER_PASSWORD_AUTH（以前称为 ADMIN_NO_SRP_AUTH）。
- 将 ALLOW_ADMIN_USER_PASSWORD_AUTH 添加到应用程序客户端的身份验证流程列表中。在您的用户池中的 App integration（应用程序集成）选项卡上的 App clients and analytics（应用程序客户端和分析）下配置应用程序客户端。有关更多信息，请参阅配置用户池应用程序客户端 (p. 258)。

自定义身份验证流程

Amazon Cognito 用户池实现了使用自定义身份验证流程，这可以帮助您使用 AWS Lambda 触发器创建基于质询/响应的身份验证模型。

> Note
>
> 您不能在自定义身份验证流中使用高级安全功能。有关更多信息，请参阅向用户池添加高级安全 (p. 227)。

自定义身份验证流程可以自定义质询和响应周期，以满足不同需求。该流程首先调用 InitiateAuth API 操作，该调用指示将使用的身份验证类型并提供了所有初始身份验证参数。Amazon Cognito 将使用以下类型的信息之一响应 InitiateAuth 调用：

- 用户质询及会话和参数。
- 错误（如果用户未能通过身份验证）
- ID、访问和刷新令牌（如果 InitiateAuth 调用中提供的参数足以使用户登录）。（通常，用户或应用程序必须首先应答质询，但这必须由您的自定义代码决定。）

如果 Amazon Cognito 使用质询响应 InitiateAuth 调用，则应用程序将收集更多输入并调用 RespondToAuthChallenge 操作。此调用提供质询响应并将其传回会话。Amazon Cognito 对 RespondToAuthChallenge 的响应类似于对 InitiateAuth 调用的响应。如果用户已登录，Amazon Cognito 会提供令牌，如果用户未登录，则 Amazon Cognito 会提供另一个质询或错误。如果 Amazon Cognito 返回另一质询，则序列重复，应用程序调用 RespondToAuthChallenge 直到用户成功登录或返回错误。有关 InitiateAuth 和 RespondToAuthChallenge API 操作的详细信息，请参阅 API 文档。

内置身份验证流程和质询

Amazon Cognito 包含内置 AuthFlow 和 ChallengeName 值，以便标准身份验证流程可以通过安全远程密码（SRP）协议验证用户名和密码。AWS 开发工具包具有对 Amazon Cognito 这些流程的内置支持。

该流程通过将 USER_SRP_AUTH 作为 AuthFlow 发送到 InitiateAuth 开始。您还在 AuthParameters 中发送 USERNAME 和 SRP_A 值。如果 InitiateAuth 调用成功，则响应将在质询参数中包括 PASSWORD_VERIFIER 作为 ChallengeName 和 SRP_B。然后，应用程序将使用 ChallengeResponses 中的 PASSWORD_VERIFIER ChallengeName 和必要参数调用 RespondToAuthChallenge。如果 RespondToAuthChallenge 调用成功并且用户登录，则 Amazon Cognito 将发布令牌。如果您为用户激活了多重验证 (MFA)，Amazon Cognito 返回 SMS_MFA 的 ChallengeName。该应用程序可以通过对 RespondToAuthChallenge 的另一次调用来提供必要的代码。

自定义身份验证流程和质询

应用程序可以启动自定义身份验证流程，具体方法是：调用 InitiateAuth 并将 CUSTOM_AUTH 用作 Authflow。借助自定义身份验证流程，三个 Lambda 触发器控制响应的质询和验证。

- DefineAuthChallenge Lambda 触发器将以前的质询和响应的会话数组作为输入。然后，它生成下一个质询名称和布尔值，指示用户是否通过身份验证并且应被授予令牌。此 Lambda 触发器是一个状态机，可通过质询控制用户的路径。
- CreateAuthChallenge Lambda 触发器将质询名称作为输入并生成质询和参数以评估响应。当 DefineAuthChallenge 返回 CUSTOM_CHALLENGE 作为下一次质询时，身份验证流程调用

CreateAuthChallenge。CreateAuthChallenge Lambda 触发器在质询元数据参数中传递下一个类型的质询。

- VerifyAuthChallengeResponse Lambda 函数会评估响应并返回布尔值以表明响应是否有效。

自定义身份验证流程还可以使用内置质询的组合，例如 SRP 密码验证和通过短信进行的 MFA。它可以使用自定义质询，如验证码或秘密问题。

在自定义身份验证流程中使用 SRP 密码验证

如果您希望将 SRP 包含在自定义身份验证流程中，则您必须开始使用 SRP。

- 要在自定义流程中启动 SRP 密码验证，应用程序将 CUSTOM_AUTH 作为 Authflow 来调用 InitiateAuth。在 AuthParameters 映射 中，来自应用程序的请求包括 SRP_A:（SRP A 值）和 CHALLENGE_NAME: SRP_A。
- CUSTOM_AUTH 流会使用 challengeName: SRP_A 和 challengeResult: true 的初始会话调用 DefineAuthChallenge Lambda 触发器。您的 Lambda 函数使用 challengeName: PASSWORD_VERIFIER、issueTokens: false 和 failAuthentication: false 作出响应。
- 接下来，该应用程序必须使用 challengeName: PASSWORD_VERIFIER 和 challengeResponses 映射中 SRP 所需的其它参数调用 RespondToAuthChallenge。
- 如果 Amazon Cognito 验证了密码，RespondToAuthChallenge 使用 challengeName: PASSWORD_VERIFIER 和 challengeResult: true 的第二个会话调用 DefineAuthChallenge Lambda 触发器。此时，DefineAuthChallenge Lambda 触发器可以使用 challengeName: CUSTOM_CHALLENGE 响应来开启自定义质询。
- 如果为用户启用了 MFA，则在 Amazon Cognito 验证密码后，您的用户将被要求设置 MFA 或使用 MFA 登录。

> **Note**
>
> Amazon Cognito 托管的登录网页无法激活 自定义身份验证质询 Lambda 触发器 (p. 134)。

有关 Lambda 触发器的更多信息，包括示例代码，请参阅使用 Lambda 触发器自定义用户池工作流 (p. 111)。

管理员身份验证流程

身份验证的最佳实践是使用 自定义身份验证流程 (p. 45) 中所述的 API 操作，以使用 SRP 进行密码验证。AWS 开发工具包使用该方法，这一方法帮助它们使用 SRP。但是，如果您希望避免 SRP 计算，还可以使用另一组专为在安全后端服务器上使用的管理员 API 操作。对于这些后端管理员实施，请使用 AdminInitiateAuth 代替 InitiateAuth。另外，请使用 AdminRespondToAuthChallenge 代替 RespondToAuthChallenge。由于您能够以明文形式提交密码，因此在使用这些操作时不必进行 SRP 计算。示例如下：

```
AdminInitiateAuth Request {
    "AuthFlow":"ADMIN_USER_PASSWORD_AUTH",
    "AuthParameters":{
        "USERNAME":"<username>",
        "PASSWORD":"<password>"
        },
    "ClientId":"<clientId>",
    "UserPoolId":"<userPoolId>"
}
```

这些管理员身份验证操作要求提供开发人员凭证，并使用 AWS 签名版本 4 (SigV4) 签名过程。这些操作在 Node.js 等标准 AWS SDK 中可用，便于用于 Lambda 函数。要使用这些操作并让它们接受明文密码，您必须在控制台中为应用程序激活这些操作。或者，您可以在调用 CreateUserPoolClient 或 UpdateUserPoolClient 时为 ExplicitAuthFlow 参数传递

ADMIN_USER_PASSWORD_AUTH。InitiateAuth 和 RespondToAuthChallenge 操作不接受 ADMIN_USER_PASSWORD_AUTH AuthFlow。

在 AdminInitiateAuth 响应 ChallengeParameters 中，USER_ID_FOR_SRP 属性（如果存在）包含用户的实际用户名而不是别名（如电子邮件地址或电话号码）。在 AdminRespondToAuthChallenge 调用的 ChallengeResponses 中，您必须在 USERNAME 参数中传递此用户名。

Note

由于后端管理员实施使用管理员身份验证流程，因此流不支持设备跟踪。在您启用设备跟踪时，管理员身份验证成功，但任何对刷新访问令牌的调用均会失败。

用户迁移身份验证流程

用户迁移 Lambda 触发器可帮助您将用户从旧式用户管理系统迁移到您的用户池。如果选择 USER_PASSWORD_AUTH 身份验证流程，则用户在用户迁移过程中无需重置密码。此流程在身份验证期间通过加密的 SSL 连接向服务发送用户的密码。

所有用户均完成迁移后，请切换为更安全的 SRP 流程。SRP 流程不通过网络发送任何密码。

要了解有关 Lambda 触发器的更多信息，请参阅使用 Lambda 触发器自定义用户池工作流 (p. 111)。

有关使用 Lambda 触发器迁移用户的更多信息，请参阅利用用户迁移 Lambda 触发器将用户导入用户池 (p. 183)。

使用 Amazon Cognito 用户群体 API 和用户群体端点

当您想要注册、登录和管理用户群体中的用户时，有两种选择。

1. 您的用户群体端点包括托管 UI 和联合身份验证端点。它们构成了一个公共网页包，当您为用户群体选择域时，Amazon Cognito 会激活这些网页。要快速开始使用 Amazon Cognito 用户群体的身份验证和授权功能，包括注册、登录、密码管理和多重身份验证（MFA）页面，请使用托管 UI 的内置用户界面。其他用户群体端点便于使用第三方身份提供者（IdP）进行身份验证。他们执行的服务包括以下各项。
   a. 服务提供者回调端点，用于处理来自您的 IdP 的经身份验证的声明，例如 saml2/idpresponse 和 oauth2/idpresponse。当 Amazon Cognito 是您的应用程序和 IdP 之间的中间服务提供者（SP）时，回调端点代表服务。
   b. 提供有关您的环境的信息的端点，例如 oauth2/userInfo 和 jwks.json。您的应用程序在使用 AWS SDK 和 OAuth 2.0 库验证令牌或检索用户配置文件数据时使用这些端点。
2. Amazon Cognito 用户群体 API 是一组用于您的 Web 或移动应用程序的工具，它在您自己的自定义前端中收集登录信息，以验证用户身份。用户群体 API 身份验证生成以下 JSON Web 令牌。
   a. 带有来自您的用户的可验证属性声明的身份令牌。
   b. 一种访问令牌，用于授权用户针对 AWS 服务端点创建经令牌授权的 API 请求。

      Note

      来自用户群体 API 身份验证的访问令牌仅包含 aws.cognito.signin.user.admin 范围。相反，如果您想生成具有额外范围的访问令牌（例如，授权对第三方 API 的请求），请通过用户群体端点对用户进行身份验证。

您可以将通常通过用户群体端点登录的联合用户与其配置文件位于用户群体本地 的用户相关联。本地用户仅存在于您的用户群体目录中，无需通过外部 IdP 进行联合身份验证。如果您在 AdminLinkProviderForUser API 请求中将他们的联合身份链接到本地用户，则他们可以使用用户群体 API 进行登录。有关更多信息，请参阅将联合用户与现有用户配置文件关联 (p. 109)。

Amazon Cognito 用户群体 API 有双重用途。它创建和配置您的 Amazon Cognito 用户群体资源。例如，您可以创建用户群体、添加 AWS Lambda 触发器以及配置您的托管 UI 域。用户群体 API 还为本地用户和关联的用户执行注册、登录和其他用户操作。

使用 Amazon Cognito 用户群体 API 的示例场景

1. 用户选择了您在应用程序中创建的“Create an account”（创建账户）按钮。他们输入电子邮件地址和密码。
2. 您的应用程序发送了 SignUp API 请求，并在用户群体中创建新用户。
3. 应用程序提示用户输入电子邮件确认代码。用户输入他们在电子邮件中收到的代码。
4. 您的应用程序发送带有用户确认码的 ConfirmSignUp API 请求。
5. 应用程序提示您的用户输入用户名和密码，而用户输入其信息。
6. 您的应用程序发送 InitiateAuth API 请求并存储 ID 令牌、访问令牌和刷新令牌。应用程序调用 OIDC 库来管理用户的令牌并为该用户维护持久会话。

在 Amazon Cognito 用户群体 API 中，您无法登录通过 IdP 进行联合身份验证的用户。您必须通过用户群体端点对这些用户进行身份验证。有关包含托管 UI 的用户群体端点的信息，请参阅用户群体联合身份验证端点和托管 UI 参考 (p. 551)。联合用户可以在托管 UI 中开始登录并选择其 IdP，您也可以跳过托管 UI，将用户直接发送到您的 IdP 以进行登录。当您的 API 发送请求到 对端点授权 (p. 557) 且带有 IdP 参数时，Amazon Cognito 会以静默方式将用户重定向到 IdP 登录页面。

用户群体端点示例场景

1. 用户选择了您在应用程序中创建的“Create an account”（创建账户）按钮。
2. 您可以向用户提供您注册开发人员凭证的社交身份提供者列表。您的用户选择了 Apple。
3. 您的应用程序向 对端点授权 (p. 557) 发出请求，提供者名称为 `SignInWithApple`。
4. 用户的浏览器打开 Apple OAuth 授权页面。用户选择允许 Amazon Cognito 读取其个人资料信息。
5. Amazon Cognito 确认 Apple 访问令牌并查询用户的 Apple 个人资料。
6. 用户向您的应用程序出示 Amazon Cognito 授权代码。
7. 应用程序与 令牌端点 (p. 562) 交换授权码并存储 ID 令牌、访问令牌和刷新令牌。应用程序调用 OIDC 库来管理用户的令牌并为该用户维护持久会话。

用户群体 API 和用户群体端点支持各种场景，如本指南中所述。以下部分探讨了用户群体 API 如何进一步划分为支持您的注册、登录和资源管理要求的类。

Amazon Cognito 用户群体经过身份验证和未经身份验证的 API 操作

Amazon Cognito 用户群体 API 既是资源管理接口，也是面向用户的身份验证和授权接口，结合了其操作中遵循的授权模型。根据 API 操作，您可能需要使用 IAM 凭证、访问令牌、会话令牌、客户端密钥或者前面这些内容的组合提供授权。对于许多用户身份验证和授权操作，您可以选择请求的经过身份验证和未经身份验证的版本。对于分发给用户的应用程序（例如移动应用程序），最佳安全实践是提供未经身份验证的操作；您无需在代码中包含任何密钥。

您只能在 IAM policy 中为 经过 IAM 身份验证的管理操作 (p. 48) 和 经过 IAM 身份验证的用户操作 (p. 50) 分配权限。

经过 IAM 身份验证的管理操作

经过 IAM 身份验证的管理操作会修改和查看您的用户群体和应用程序客户端配置，就像您在 AWS Management Console 中所做的那样。

例如，要在 UpdateUserPool API 请求中修改您的用户群体，您必须提供 AWS 凭证和 IAM 权限才能更新资源。

要在 AWS Command Line Interface（AWS CLI）或 AWS SDK 中授权这些请求，请使用将 IAM 凭证添加到请求的环境变量或客户端配置来配置您的环境。有关更多信息，请参阅《AWS 一般参考》中的使用 AWS 凭

证访问 AWS。对于 Amazon Cognito 用户群体 API，您也可以直接向服务端点发送请求。您必须使用嵌入到请求标头中的 AWS 凭证授权或签署 这些请求。有关更多信息，请参阅签署 AWS API 请求。

经过 IAM 身份验证的管理操作

AddCustomAttributes

CreateGroup

CreateIdentityProvider

CreateResourceServer

CreateUserImportJob

CreateUserPool

CreateUserPoolClient

CreateUserPoolDomain

DeleteGroup

DeleteIdentityProvider

DeleteResourceServer

DeleteUserPool

DeleteUserPoolClient

DeleteUserPoolDomain

DescribeIdentityProvider

DescribeResourceServer

DescribeRiskConfiguration

DescribeUserImportJob

DescribeUserPool

DescribeUserPoolClient

DescribeUserPoolDomain

GetCSVHeader

GetGroup

GetIdentityProviderByIdentifier

GetSigningCertificate

GetUICustomization

GetUserPoolMfaConfig

ListGroups

ListIdentityProviders

ListResourceServers

经过 IAM 身份验证的管理操作

ListTagsForResource

ListUserImportJobs

ListUserPoolClients

ListUserPools

ListUsers

ListUsersInGroup

SetRiskConfiguration

SetUICustomization

SetUserPoolMfaConfig

StartUserImportJob

StopUserImportJob

TagResource

UntagResource

UpdateGroup

UpdateIdentityProvider

UpdateResourceServer

UpdateUserPool

UpdateUserPoolClient

UpdateUserPoolDomain

经过 IAM 身份验证的用户操作

经过 IAM 身份验证的用户操作注册、登录、管理凭证、修改和查看您的用户。

例如，您可以有一个服务器端应用程序层，为 Web 前端提供支持。您的服务器端应用程序是可以信任的 OAuth 机密客户端，具有对您 Amazon Cognito 资源的特权访问权限。要在应用程序中注册用户，您的服务器可以在 AdminCreateUser API 请求中包含 AWS。有关 OAuth 客户端类型的更多信息，请参阅 The OAuth 2.0 Authorization Framework（OAuth 2.0 授权框架）中的客户端类型。

要在 AWS CLI 或 AWS SDK 中授权这些请求，请使用将 IAM 凭证添加到请求的环境变量或客户端配置来配置服务器端应用程序的环境。有关更多信息，请参阅《AWS 一般参考》中的使用 AWS 凭证访问 AWS。对于 Amazon Cognito 用户群体 API，您也可以直接向服务端点发送请求。您必须使用嵌入到请求标头中的 AWS 凭证授权或签署 这些请求。有关更多信息，请参阅签署 AWS API 请求。

如果您的应用程序客户端有客户端密钥，则您必须提供 IAM 凭证，并在 AuthParameters 中提供 SecretHash 参数或 SECRET_HASH 值（取决于操作）。有关更多信息，请参阅计算密钥哈希值 (p. 172)。

经过 IAM 身份验证的用户操作

AdminAddUserToGroup

经过 IAM 身份验证的用户操作

AdminConfirmSignUp

AdminCreateUser

AdminDeleteUser

AdminDeleteUserAttributes

AdminDisableProviderForUser

AdminDisableUser

AdminEnableUser

AdminForgetDevice

AdminGetDevice

AdminGetUser

AdminInitiateAuth

AdminLinkProviderForUser

AdminListDevices

AdminListGroupsForUser

AdminListUserAuthEvents

AdminRemoveUserFromGroup

AdminResetUserPassword

AdminRespondToAuthChallenge

AdminSetUserMFAPreference

AdminSetUserPassword

AdminSetUserSettings

AdminUpdateAuthEventFeedback

AdminUpdateDeviceStatus

AdminUpdateUserAttributes

AdminUserGlobalSignOut

未经身份验证的用户操作

未经身份验证的用户操作注册、登录以及为您的用户启动密码重置。当您希望 Internet 上的任何人都可以注册并登录您的应用程序时，请使用未经身份验证（公开）的 API 操作。

例如，要在您的应用程序中注册用户，您可以分发 OAuth 公共客户端，其中不提供对密钥的任何特权访问。您可以使用未经身份验证的 API 操作 SignUp 注册此用户。

要在使用 AWS SDK 开发的公共客户端中发送这些请求，您无需配置任何凭证。对于没有额外授权的 Amazon Cognito 用户群体 API，您也可以直接向服务端点发送请求。

如果您的应用程序客户端有客户端密钥，则您必须在 AuthParameters 中提供 SecretHash 参数或 SECRET_HASH 值（取决于操作）。有关更多信息，请参阅计算密钥哈希值 (p. 172)。

未经身份验证的用户操作
SignUp
ConfirmSignUp
ResendConfirmationCode
ForgotPassword
ConfirmForgotPassword
InitiateAuth

经过令牌身份验证的用户操作

经过令牌身份验证的用户操作在用户已登录或开始登录流程之后，注销、管理其凭证、修改和查看用户。如果您不想在应用程序中分发密钥，并且想要使用用户自己的凭证授权请求，请使用经过令牌身份验证的 API 操作。如果用户已完成登录，则您必须使用访问令牌来授权其经过令牌身份验证的 API 请求。如果您的用户正在登录流程中，则您必须使用 Amazon Cognito 在对先前请求的响应中返回的会话令牌，授权其经过令牌身份验证的 API 请求。

例如，在公共客户端中，您可能希望更新用户的配置文件，限制用户仅对自己的配置文件具有写入权限。要进行此更新，您的客户端可以在 UpdateUserAttributes API 请求中包含用户的访问令牌。

要在使用 AWS SDK 开发的公共客户端中发送这些请求，您无需配置任何凭证。在您的请求中包含 AccessToken 或 Session 参数。对于 Amazon Cognito 用户群体 API，您也可以直接向服务端点发送请求。要向服务端点授权请求，请在请求的 POST 正文中包含访问令牌或会话令牌。

要签署经令牌验证的操作的 API 请求，请将访问令牌作为 Authorization 标头包含在请求中，格式为 Bearer *<Base64-encoded access token>*。

经过令牌身份验证的用户操作	AccessToken	会话
RespondToAuthChallenge		✓
ChangePassword	✓	
GetUser	✓	
UpdateUserAttributes	✓	
DeleteUserAttributes	✓	
DeleteUser	✓	
ConfirmDevice	✓	
ForgetDevice	✓	
GetDevice	✓	
ListDevices	✓	
UpdateDeviceStatus	✓	

经过令牌身份验证的用户操作	AccessToken	会话
GetUserAttributeVerificationCode	✓	
VerifyUserAttribute	✓	
SetUserSettings	✓	
SetUserMFAPreference	✓	
GlobalSignOut	✓	
AssociateSoftwareToken	✓	✓
UpdateAuthEventFeedback		✓
VerifySoftwareToken	✓	✓
RevokeToken[1]		

[1] RevokeToken 获取刷新令牌作为参数。刷新令牌用作授权令牌和目标资源。

更新用户群体配置

要在 AWS Management Console中更改 Amazon Cognito 用户群体的设置，请浏览用户群体设置中基于功能的选项卡，并按照本指南其他部分的描述更新字段。创建用户群体后，您无法更改一些设置。如果您想要更改以下设置，您必须创建新的用户群体或应用程序客户端。

用户池名称

分配给用户群体的友好名称。要更改用户群体的名称，请创建新的用户群体。

Amazon Cognito 用户群体登录选项

用户登录时可以作为用户名传递的属性。创建用户群体时，您可以选择允许使用用户名、电子邮件地址、电话号码或首选用户名进行登录。要更改用户群体登录选项，请创建新的用户群体。

Make user name case sensitive（使用户名区分大小写）

当您创建的用户名与其他用户名（字母大小写除外）匹配时，Amazon Cognito 可以将其视为同一用户或唯一用户。有关更多信息，请参阅用户池区分大小写 (p. 238)。要更改区分大小写，请创建新的用户群体。

必需的属性

当用户注册时或当您创建属性时，用户必须为这些属性提供值。有关更多信息，请参阅用户池属性 (p. 243)。要更改必需的属性，请创建新的用户群体。

客户端密钥

创建应用程序客户端时，可以生成客户端密钥，以便只有受信任的来源才能向用户群体发出请求。有关更多信息，请参阅配置用户池应用程序客户端 (p. 258)。要更改客户端密钥，请在同一用户群体中创建新的应用程序客户端。

自定义属性

具有自定义名称的属性。您可以更改用户自定义属性的值，但不能从用户群体中删除自定义属性。有关更多信息，请参阅用户池属性 (p. 243)。如果达到自定义属性的最大数量并且您想要修改列表，请创建新的用户群体。

使用 Amazon Cognito API 或 AWS CLI 更新用户群体

您可以使用自动化工具 [如 Amazon Cognito API 或 AWS Command Line Interface (AWS CLI)] 更改 Amazon Cognito 用户群体的配置。以下过程使用 UpdateUserPool API 操作更新配置。使用不同输入字段的相同方法适用于 UpdateUserPoolClient。

> **Important**
>
> 如果您未为现有参数提供值，Amazon Cognito 将它们设置为默认值。例如，如果您有现有的 `LambdaConfig`，然后提交具有空 `LambdaConfig` 的 `UpdateUserPool`，则会删除为用户群体触发器分配的所有 Lambda 函数。当您想自动更改用户群体配置时，请相应地进行规划。

1. 使用 DescribeUserPool 捕获用户群体的现有状态。
2. 设置 `DescribeUserPool` 的输出的格式以与 `UpdateUserPool` 的请求参数匹配。从输出 JSON 中删除以下顶级字段及其子对象。
   - `Arn`
   - `CreationDate`
   - `CustomDomain`
     - 使用 UpdateUserPoolDomain API 操作更新此字段。
   - `Domain`
     - 使用 UpdateUserPoolDomain API 操作更新此字段。
   - `EmailConfigurationFailure`
   - `EstimatedNumberOfUsers`
   - `Id`
   - `LastModifiedDate`
   - `Name`
   - `SchemaAttributes`
   - `SmsConfigurationFailure`
   - `Status`
3. 确认生成的 JSON 与 `UpdateUserPool` 的请求参数匹配。
4. 修改任何您想要在生成的 JSON 中更改的参数。
5. 提交 `UpdateUserPool` API 请求，同时将您修改后的 JSON 作为请求输入。

您还可以在 AWS CLI 中，在 `update-user-pool` 的 `--cli-input-json` 参数中使用这一修改后的 `DescribeUserPool` 输出。

或者，运行以下 AWS CLI 命令，为 `update-user-pool` 生成已接受输入字段为空值的 JSON。然后，您可以使用用户群体中的现有值填充这些字段。

```
aws cognito-idp update-user-pool --generate-cli-skeleton --output json
```

运行以下命令以为应用程序客户端生成相同的 JSON 对象。

```
aws cognito-idp update-user-pool-client --generate-cli-skeleton --output json
```

使用 Amazon Cognito 托管 UI 进行注册和登录

Amazon Cognito 托管 UI 为您提供 OAuth 2.0 合规授权服务器。它提供了终端用户流的默认实现，例如注册、身份验证等。您可以通过更改用户池配置来自定义用户流，例如添加多重身份验证 (MFA)。您的

应用程序将重定向到托管 UI，它将处理用户流。可以通过提供品牌特定徽标以及自定义托管 UI 元素的设计来定制用户体验。Amazon Cognito 托管 UI 还允许您添加功能，让终端用户可使用社交服务提供商（Facebook、Amazon、Google 和 Apple）、任何 OpenID Connect（OIDC）合规和 SAML 提供商登录。

主题

- 使用 AWS Amplify 设置托管 UI (p. 55)
- 使用 Amazon Cognito 控制台设置托管 UI (p. 55)
- 配置用户池应用程序客户端 (p. 58)
- 配置用户池域 (p. 61)
- 自定义内置登录网页和注册网页 (p. 67)
- 使用托管 UI 注册和登录 (p. 71)
- 为您的用户池定义资源服务器 (p. 82)

使用 AWS Amplify 设置托管 UI

如果您使用 AWS Amplify 向 Web 或移动应用程序添加身份验证功能，则可以使用 AWS Amplify 框架中的命令行界面 (CLI) 和库来设置托管 UI。要向应用程序添加身份验证功能，您可以使用 AWS Amplify CLI 将 `Auth` 类别添加到项目中。然后，在客户端代码中，您可以使用 AWS Amplify 库，通过 Amazon Cognito 用户池对用户进行身份验证。

您可以显示预构建的托管 UI，也可以通过重定向到社交登录提供商（例如 Facebook、Google、Amazon 或 Apple）的 OAuth 2.0 终端节点联合用户身份。用户成功通过社交服务提供商身份验证之后，如果需要，AWS Amplify 在您的用户池中创建一个新用户，并向您的应用程序提供用户的 OIDC 令牌。

有关更多信息，请参阅您使用的应用平台的 AWS Amplify 框架文档：

- 适用于 JavaScript 的 AWS Amplify 身份验证。
- 适用于 iOS 的 AWS Amplify 身份验证。
- 适用于 Android 的 AWS Amplify 身份验证。

使用 Amazon Cognito 控制台设置托管 UI

创建应用程序客户端

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 App integration（应用程序集成）选项卡。
5. 选择 App clients（应用程序客户端）下的 Create an app client（创建应用程序客户端）。
6. 选择 App type（应用程序类型）：Public client（公有客户端）、Confidential client（机密客户端），或者 Other（其它）。Public client（公有客户端）通常在用户的设备上运行，并使用未经身份验证和令牌身份验证的 API。Confidential client（机密客户端）通常在您通过客户端密钥和 API 凭证信任的中央服务器上的应用程序运行，并使用授权标头和 AWS Identity and Access Management 凭证对请求进行签名。如果您的使用案例与预配置的应用程序的 Public client（公有客户端）或 Confidential client（机密客户端）应用程序设置不同，请选择 Other（其它）。
7. 输入 App client name（应用程序客户端名称）。
8. 选择您希望在应用程序客户端中允许的 Authentication flows（身份验证流程）。
9. 配置 Authentication flow session duration（身份验证流程会话持续时间）。这是您的用户在会话令牌过期之前必须完成每个身份验证质询的时间。

10. （可选）配置令牌的到期时间。

    a. 指定应用程序客户端的 Refresh token expiration（刷新令牌的过期时间）。默认值为 30 天。您可以将其更改为 1 小时到 10 年之间的任何值。

    b. 指定应用程序客户端的 Access token expiration（访问令牌的过期时间）。默认值为 1 小时。您可以将其更改为 5 分钟到 24 小时之间的任何值。

    c. 指定应用程序客户端的 ID token expiration（ID 令牌的过期时间）。默认值为 1 小时。您可以将其更改为 5 分钟到 24 小时之间的任何值。

    Important

    如果您使用托管 UI 并将令牌生命周期配置为不到一小时，用户将能够根据其会话 Cookie 持续时间使用令牌，该 Cookie 当前固定在一小时。

11. 选择 Generate client secret（生成客户端密钥），让 Amazon Cognito 为您生成一个客户端密钥。客户端密钥通常与机密客户端关联。
12. 是否为此应用程序客户端选择 Enable token revocation（启用令牌撤消）。这会增加令牌的大小。有关更多信息，请参阅撤消令牌。
13. 是否为此应用程序客户端选择 Prevent error messages that reveal user existence（防止暴露用户存在的错误消息）。Amazon Cognito 将响应不存在的用户的登录请求，并显示一条指出用户名或密码不正确的通用消息。
14. （可选）为此应用程序客户端配置 Attribute read and write permissions（属性读取和写入权限）。您的应用程序客户端有权限读取和写入用户池属性架构的有限子集。
15. 选择 Create（创建）。
16. 记下 Client id（客户端 ID）。这将识别注册和登录请求中的应用程序客户端。

配置应用程序

1. 在 App integration（应用程序集成）选项卡的 App clients（应用程序客户端）下，选择您的应用程序客户端。检查当前 Hosted UI（托管 UI）信息。
2. 在 Allowed callback URL(s)（允许的回调 URL）下选择 Add a callback URL（添加回调 URL）。回调 URL 是在用户成功登录之后将被重新导向的地址。
3. 在 Allowed sign-out URL(s)（允许的注销 URL）下选择 Add a sign-out URL（添加注销 URL）。注销 URL 是在您的用户注销后会被重新导向的地址。
4. 从 Identity providers（身份提供商）列表中添加至少一个列出的选项。
5. 在 OAuth 2.0 grant types（OAuth 2.0 授予类型）下，选择 Authorization code grant（授权代码授予）以返回随后更换为用户池令牌的授权代码。由于令牌绝不会直接向终端用户公开，因此它们不太可能被泄露。但是，后端需要自定义应用程序以将授权代码换成用户池令牌。出于安全原因，对于移动应用程序，强烈建议您将授权代码授予流程与代码交换的证明密钥（PKCE）一起使用。
6. 在 OAuth 2.0 grant types（OAuth 2.0 授予类型）下，选择 Implicit grant（隐式授予），以便从 Amazon Cognito 将用户池 JSON Web 令牌（JWT）返回给您。当没有可用于将授权代码更换为令牌的后端时，您可以使用此流程。它对于调试令牌也很有帮助。
7. 您可以同时启用 Authorization code（授权代码）和 Implicit code（隐式代码），然后按需使用每个授予。如果未选中 Authorization code（授权代码）或 Implicit code（隐式代码）授权，并且您的应用程序客户端具有客户端密钥，您可以启用 Client credentials（客户端凭证）授权。只有在您的应用程序需要代表自己而不是代表用户请求访问令牌时，才选择 Client credentials（客户端凭证）。
8. 选择您要为此应用程序客户端授权的 OpenID Connect scopes（OpenID Connect 范围）。
9. 选择 Save changes（保存更改）。

配置域

1. 导航到用户池的 App integration（应用程序集成）选项卡。

2. 选择在 Domain（域）旁边的 Actions（操作），然后选择 Create custom domain（创建自定义域）或 Create Cognito domain（创建 Cognito 域）。如果您已配置用户池域，请先选择 Delete Cognito domain（删除 Cognito 域）或 Delete custom domain（删除自定义域），然后再创建新的自定义域。
3. 输入可用的域前缀，将其与 Cognito domain（Cognito 域）结合使用。有关设置 Custom domain（自定义域）的信息，请参阅将您自己的域用于托管 UI
4. 选择 Create（创建）。

查看您的登录页面

在 Amazon Cognito 控制台中，请在 App integration（应用程序集成）选项卡中的 App clients and analytics（应用程序客户端和分析）下，选择应用程序客户端配置中的 View Hosted UI（查看托管 UI）按钮。此按钮会将您转到托管 UI 中的登录页面，其中包含以下基本参数。

- 应用程序客户端 id
- 授权代码授予请求
- 对您为当前应用程序客户端激活的所有范围的请求
- 当前应用程序客户端列表中的第一个回调 URL

当您想要测试托管 UI 的基本功能时，View hosted UI（查看托管 UI）按钮很有用。您可以使用其他和修改后的参数自定义登录 URL。在大多数情况下，View hosted UI（查看托管 UI）链接的自动生成的参数不完全符合您的应用程序需求。在这些情况下，您必须自定义应用程序在用户登录时调用的 URL。有关登录参数键和值的更多信息，请参阅 用户群体联合身份验证端点和托管 UI 参考 (p. 551)。

托管 UI 登录网页使用以下 URL 格式。记下 `response_type`。在此示例中，response_type=code 表示授权代码授予。

```
https://<your_domain>/oauth2/authorize?
response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
```

当您使用自定义参数导航到 `/oauth2/authorize` 端点时，Amazon Cognito 会将您重定向到 `/oauth2/login` 端点，或者在您具有 `identity_provider` 或 `idp_identifier` 参数时，静默地将您重定向到 IdP 登录页面。有关绕过托管 UI 的示例 URL，请参阅 SAML 会话在 Amazon Cognito 用户池中启动 (p. 100)。

您可以使用针对隐式代码授予的以下 URL 查看托管 UI 登录网页，其中，response_type=token。成功登录后，Amazon Cognito 会将用户池令牌返回到您的 Web 浏览器的地址栏。

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

您可以在响应中的 `#idtoken=` 参数后面找到 JSON Web Token (JWT) 身份令牌。

下面是来自隐式授予请求的示例响应。您的身份令牌字符串长得多。

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

如果未立即显示对托管 UI 页面的更改，请等待几分钟，然后刷新页面。使用 RS256 算法对 Amazon Cognito 用户池令牌进行签名。您可以使用 AWS Lambda 解码和验证用户群体令牌，请参阅 GitHub 上的解码和验证 Amazon Cognito JWT 令牌。

您的域将显示在 Domain name（域名）页面上。您的应用程序客户端 ID 和回调 URL 将显示在 App client settings（应用程序客户端设置）页面上。

Note

Amazon Cognito 托管的 UI 不支持自定义身份验证流程。

Amazon Cognito 托管 UI 不支持自定义跨源资源共享 (CORS) 源策略。托管 UI 中的 CORS 策略将阻止用户在其请求中传递身份验证参数。应在应用程序的 Web 前端中实施 CORS 策略。Amazon Cognito 会对发往以下 OAuth 端点的请求返回 `Access-Control-Allow-Origin: *` 响应标头。

1. 令牌端点 (p. 562)
2. 撤消端点 (p. 567)
3. UserInfo 终端节点 (p. 566)

配置用户池应用程序客户端

在创建用户池后，您可以配置一个应用程序客户端，以便使用内置网页来注册和登录用户。For terminology, see App Client Settings Terminology (p. 60)。

配置应用程序客户端（AWS Management Console）

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 App integration（应用程序集成）选项卡。
5. 选择 App clients（应用程序客户端）下的 Create an app client（创建应用程序客户端）。
6. 选择 App type（应用程序类型）：Public client（公有客户端）、Confidential client（机密客户端），或者 Other（其它）。Public client（公有客户端）通常在用户的设备上运行，并使用未经身份验证和令牌身份验证的 API。Confidential client（机密客户端）通常在您通过客户端密钥和 API 凭证信任的中央服务器上的应用程序运行，并使用授权标头和 AWS Identity and Access Management 凭证对请求进行签名。如果您的使用案例与预配置的应用程序的 Public client（公有客户端）或 Confidential client（机密客户端）应用程序设置不同，请选择 Other（其它）。
7. 输入 App client name（应用程序客户端名称）。
8. 选择您希望在应用程序客户端中允许的 Authentication flows（身份验证流程）。有关更多信息，请参阅用户池身份验证流程。
9. 配置 Authentication flow session duration（身份验证流程会话持续时间）。这是您的用户在会话令牌过期之前必须完成每个身份验证质询的时间。
10. （可选）配置令牌的到期时间。
    a. 指定应用程序客户端的 Refresh token expiration（刷新令牌的过期时间）。默认值为 30 天。您可以将其更改为 1 小时到 10 年之间的任何值。
    b. 指定应用程序客户端的 Access token expiration（访问令牌的过期时间）。默认值为 1 小时。您可以将其更改为 5 分钟到 24 小时之间的任何值。
    c. 指定应用程序客户端的 ID token expiration（ID 令牌的过期时间）。默认值为 1 小时。您可以将其更改为 5 分钟到 24 小时之间的任何值。

       Important

       如果您使用托管 UI 并将令牌生命周期配置为不到一小时，用户将能够根据其会话 Cookie 持续时间使用令牌，该 Cookie 当前固定在一小时。
11. 选择 Generate client secret（生成客户端密钥），让 Amazon Cognito 为您生成一个客户端密钥。客户端密钥通常与机密客户端关联。

12. 是否为此应用程序客户端选择 Enable token revocation（启用令牌撤消）。这将增加 Amazon Cognito 发出的令牌的大小。有关更多信息，请参阅撤消令牌。
13. 是否为此应用程序客户端选择 Prevent error messages that reveal user existence（防止暴露用户存在的错误消息）。Amazon Cognito 将响应不存在的用户的登录请求，并显示一条指出用户名或密码不正确的通用消息。
14. （可选）为此应用程序客户端配置 Attribute read and write permissions（属性读取和写入权限）。您的应用程序客户端具有仅读取和写入用户池属性架构的有限子集的权限。有关更多信息，请参阅属性权限和范围。
15. 选择 Create（创建）。
16. 记下 Client id（客户端 ID）。这将识别注册和登录请求中的应用程序客户端。

配置应用程序客户端（AWS CLI 和 AWS API）

您可以使用 AWS CLI 创建、更新、描述和删除您的用户池应用程序客户端。

在这些示例中，将您的用户池和应用程序客户端 ID 值替换为“*MyUserPoolID*”和“*MyAppClientID*”。同样，您的参数值可能不同于这些示例中使用的值。

Note

回调和注销 URL 采用 JSON 格式，以防止 CLI 将它们视为远程参数文件：

```
--callback-urls "["https://example.com"]"
--logout-urls "["https://example.com"]"
```

更新用户池应用程序客户端（AWS CLI 和 AWS API）

在 AWS CLI 中，输入以下命令：

```
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，则 AWS CLI 会返回确认：

```
{
    "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
    }
```

```
}
```

请参阅 AWS CLI 命令参考了解更多信息：update-user-pool-client。

AWS API：UpdateUserPoolClient

创建用户池应用程序客户端（AWS CLI 和 AWS API）

```
aws cognito-idp create-user-pool-client --user-pool-id MyUserPoolID --client-name myApp
```

请参阅 AWS CLI 命令参考了解更多信息：create-user-pool-client。

AWS API：CreateUserPoolClient

获取有关用户池应用程序客户端的信息（AWS CLI 和 AWS API）

```
aws cognito-idp describe-user-pool-client --user-pool-id MyUserPoolID --client-
id MyClientID
```

请参阅 AWS CLI 命令参考了解更多信息：describe-user-pool-client。

AWS API：DescribeUserPoolClient

列出一个用户池中所有应用程序客户端的信息（AWS CLI 和 AWS API）

```
aws cognito-idp list-user-pool-clients --user-pool-id "MyUserPoolID" --max-results 3
```

请参阅 AWS CLI 命令参考了解更多信息：list-user-pool-clients。

AWS API：ListUserPoolClients

删除用户池应用程序客户端（AWS CLI 和 AWS API）

```
aws cognito-idp delete-user-pool-client --user-pool-id "MyUserPoolID" --client-id
 "MyAppClientID"
```

请参阅 AWS CLI 命令参考了解更多信息：delete-user-pool-client。

AWS API：DeleteUserPoolClient

应用程序客户端设置术语

以下术语和定义可帮助您配置应用程序客户端。

启用身份提供商

您可以选择身份提供商（IdP）来对用户进行身份验证。该服务可以由您的用户池或第三方（如 Facebook）来执行。要使用 IdP，您需要先启用它。您可以启用多个 IdP，但必须至少启用一个。有关使用外部 IdP 的更多信息，请参阅通过第三方添加用户池登录 (p. 85)。

回调 URL

回调 URL 指示在用户成功登录之后将被重新导向到哪里。选择至少一个回调 URL。回调 URL 必须：

- 是绝对 URI。
- 已预先向客户端注册。
- 不包含片段组件。

请参阅 OAuth 2.0 – 重新导向端点。

Amazon Cognito 要求使用 HTTPS 而不是 HTTP，但 http://localhost（仅用于测试目的）除外。

应用程序回调 URL（如 `myapp://example`）也受支持。

注销 URL

注销 URL 指示在您的用户注销后会被重定向到哪里。

允许的 OAuth 流程

Authorization code grant (授权代码授予) 流启动代码授予流，这会提供授权代码作为响应。此代码可用来与令牌端点 (p. 562)交换访问令牌。由于令牌绝不会直接向终端用户公开，因此它们不太可能被泄露。但是，后端需要自定义应用程序以将授权代码换成用户池令牌。

> **Note**
>
> 出于安全原因，对于移动应用程序，强烈建议您仅将授权代码授予流程与 PKCE 一起使用。

Implicit grant（隐式授予）流允许客户端直接从对端点授权 (p. 557)获取访问令牌（根据范围，可选择获取 ID 令牌）。如果您的应用程序无法启动 Authorization code grant（授权代码授予）流，请选择此流。有关更多信息，请参阅 OAuth 2.0 规范。

您可以同时激活 Authorization code grant（授权代码授予）和 Implicit code grant（隐式代码授予），然后按需使用每个授予。

Client credentials (客户端凭证) 流在机器到机器的通信中使用。您可以使用此流来请求访问令牌，从而访问自己的资源。当您的应用程序代表自己而不是代表用户来请求令牌时，选择此流。

> **Note**
>
> 由于您没有以用户身份调用客户端凭证流程，因此只能使用此流程填充访问令牌的自定义范围。自定义范围就是您为自己的资源服务器定义的范围。请参阅为您的用户池定义资源服务器 (p. 82)。

允许的 OAuth 范围

选择以下一个或多个 `OAuth` 范围来指定可以为访问令牌请求的访问权限。

- `phone` 范围授予对 `phone_number` 和 `phone_number_verified` 声明的访问权限。此范围只能通过 `openid` 范围来请求。
- `email` 范围授予对 `email` 和 `email_verified` 声明的访问权限。此范围只能通过 `openid` 范围来请求。
- `openid` 范围声明您要检索与 OpenID Connect 规范一致的范围。除非您请求 `openid` 范围，否则 Amazon Cognito 不会返回 ID 令牌。
- `aws.cognito.signin.user.admin` 范围授予对需要访问令牌的 Amazon Cognito 用户池 API 操作的访问权限，例如 UpdateUserAttributes 和 VerifyUserAttribute。
- `profile` 范围授予对客户端可读取的所有用户属性的访问权限。此范围只能通过 `openid` 范围来请求。

允许的自定义范围

自定义范围是您在 Resource Servers (资源服务器) 中为自己的资源服务器定义的范围。格式为：*##-###-###/##*。请参阅为您的用户池定义资源服务器 (p. 82)。

有关 OAuth 范围的更多信息，请参阅标准 OIDC 范围列表。

配置用户池域

设置应用程序客户端之后，您可以配置注册和登录网页的地址。您可以使用 Amazon Cognito 托管域并选择可用域前缀，或者也可以使用您自己的 Web 地址作为自定义域。

要使用 AWS Management Console 添加应用程序客户端和 Amazon Cognito 托管域，请参阅添加应用程序以启用托管 Web UI。

Note

您不能在域前缀中使用文本 `aws`、`amazon` 或 `cognito`。

主题

- 将 Amazon Cognito 域用于托管 UI (p. 62)
- 将您自己的域用于托管 UI (p. 63)

将 Amazon Cognito 域用于托管 UI

设置客户端应用程序之后，您可以配置注册和登录网页的地址。您可以将托管 Amazon Cognito 域与您自己的域前缀一起使用。

要使用 AWS Management Console 添加应用程序客户端和 Amazon Cognito 托管域，请参阅添加应用程序以启用托管 Web UI。

主题

- 先决条件 (p. 62)
- 步骤 1：配置托管用户池域 (p. 62)
- 步骤 2：验证登录页面 (p. 63)

先决条件

在开始之前，您需要：

- 用户池和应用程序客户端。有关更多信息，请参阅 用户池入门 (p. 30)。

步骤 1：配置托管用户池域

要配置托管用户池域

您可以使用 AWS Management Console、AWS CLI 或 API 来配置用户池域。

Amazon Cognito console

配置域

1. 导航到用户池的 App integration（应用程序集成）选项卡。
2. 选择在 Domain（域）旁边的 Actions（操作），然后选择 Create custom domain（创建自定义域）或 Create Cognito domain（创建 Cognito 域）。如果您已配置用户池域，请先选择 Delete Cognito domain（删除 Cognito 域）或 Delete custom domain（删除自定义域），然后再创建新的自定义域。
3. 输入可用的域前缀，将其与 Cognito domain（Cognito 域）结合使用。有关设置 Custom domain（自定义域）的信息，请参阅将您自己的域用于托管 UI
4. 选择 Create（创建）。

CLI/API

使用以下命令可以创建域前缀并将其分配到您的用户池。

配置用户池域

- AWS CLI: `aws cognito-idp create-user-pool-domain`

示例: `aws cognito-idp create-user-pool-domain --user-pool-id <user_pool_id> --domain <domain_name>`

- AWS API：CreateUserPoolDomain

获取有关域的信息

- AWS CLI: `aws cognito-idp describe-user-pool-domain`

  示例: `aws cognito-idp describe-user-pool-domain --domain <domain_name>`
- AWS API：DescribeUserPoolDomain

如需删除域

- AWS CLI: `aws cognito-idp delete-user-pool-domain`

  示例: `aws cognito-idp delete-user-pool-domain --domain <domain_name>`
- AWS API：DeleteUserPoolDomain

步骤 2：验证登录页面

- 验证登录页面是否可从您的 Amazon Cognito 托管域访问。

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

您的域显示在 Amazon Cognito 控制台的 Domain name (域名) 页面上。您的应用程序客户端 ID 和回调 URL 将显示在 App client settings（应用程序客户端设置）页面上。

将您自己的域用于托管 UI

在设置应用程序客户端后，您可以使用适用于 Amazon Cognito 托管 UI 和身份验证 API 终端节点的自定义域配置用户池。利用自定义域，您可以使用您自己的 Web 地址以允许用户登录您的应用程序。

主题

- 将自定义域添加到用户池 (p. 63)
- 更改自定义域的 SSL 证书 (p. 66)

将自定义域添加到用户池

要将自定义域添加到用户池，请在 Amazon Cognito 控制台中指定域名，并提供您使用 AWS Certificate Manager（ACM）管理的证书。在添加域后，Amazon Cognito 提供了一个要添加到 DNS 配置的别名目标。

先决条件

在开始之前，您需要：

- 用户池和应用程序客户端。有关更多信息，请参阅用户池入门 (p. 30)。
- 您拥有的 Web 域。其父域必须在 DNS 中有有效的 A 记录。父域可以是域的根，也可以是域层次结构中上一级的子域。例如，如果您的自定义域是 auth.xyz.example.com，Amazon Cognito 必须能够将 xyz.example.com 解析为 IP 地址。为了防止意外影响客户基础设施，Amazon Cognito 不支持对自定义域使用顶级域 (TLD)。有关更多信息，请参阅域名。

- 能够为自定义域创建子域。我们建议使用 auth 作为子域。例如：*auth.example.com*。

  Note

  如果您没有通配符证书，则可能需要为自定义域的子域获取新证书。
- 由 ACM 管理的安全套接字层（SSL）证书。

  Note

  在请求或导入证书之前，您必须在 ACM 控制台中将AWS区域更改为美国东部（弗吉尼亚北部）。
- 要设置自定义域名或更新其证书，您必须有更新 Amazon CloudFront 分配的权限。为此，您可以为 AWS 账户 中的用户附加以下 IAM policy 声明：

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Sid": "AllowCloudFrontUpdateDistribution",
            "Effect": "Allow",
            "Action": [
                "cloudfront:updateDistribution"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

有关在 CloudFront 中授权操作的更多信息，请参阅对 CloudFront 使用基于身份的策略 (IAM policy)。

Amazon Cognito 最初使用您的 IAM 权限来配置 CloudFront 分配，但分配由 AWS 管理。您无法更改 Amazon Cognito 将其与您的用户群体相关联的 CloudFront 分配的配置。例如，您无法更新安全策略中支持的 TLS 版本。

步骤 1：输入自定义域名

您可以使用 Amazon Cognito 控制台或 API 将域添加到用户池。

Amazon Cognito console

将域从 Amazon Cognito 控制台添加到用户池：

1. 登录 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User pools（用户池）。
3. 选择要向其添加域的用户池。
4. 选择 App integration（应用程序集成）选项卡。
5. 选择在 Domain（域）旁边的 Actions（操作），然后选择 Create custom domain（创建自定义域）。

   Note

   如果您已配置用户池域，请先选择 Delete Cognito domain（删除 Cognito 域）或 Delete custom domain（删除自定义域）删除现有域，然后再创建新的自定义域。
6. 对于 Custom domain（自定义域），请输入您希望与 Amazon Cognito 一起使用的域的 URL。您的域名只能包含小写字母、数字和连字符。请勿对第一个或最后一个字符使用连字符。使用句点来分隔子域名。

7. 对于 ACM certificate（ACM 证书），请选择要用于您的域的 SSL 证书。只有美国东部（弗吉尼亚北部）的 ACM 证书才有资格与 Amazon Cognito 自定义域一起使用，无论用户池的 AWS 区域 如何。

   如果您没有可用证书，则可以使用 ACM 在美国东部（弗吉尼亚北部）预置一个证书。有关更多信息，请参阅《AWS Certificate Manager 用户指南》中的入门。
8. 选择 Create（创建）。
9. Amazon Cognito 将您返回到 App integration（应用程序集成）选项卡。此时将显示标题为 Create an alias record in your domain's DNS（在域名的 DNS 中创建别名记录）的消息。记下控制台中显示的 Domain（域）和 Alias target（别名目标）。在下一步骤中，将使用它们将流量指向您的自定义域。

API

使用 Amazon Cognito API 将域添加到用户池：

- 使用 CreateUserPoolDomain 操作。

步骤 2：添加别名目标和子域

在本步骤中，您将通过域名服务器 (DNS) 服务提供商设置一个别名，该别名指回到上一个步骤中的别名目标。如果您将 Amazon Route 53 用于 DNS 地址解析，请选择使用 Route 53 添加别名目标和子域部分。

将别名目标和子域添加到当前 DNS 配置

- 如果您没有将 Route 53 用于 DNS 地址解析，则必须使用您的 DNS 服务提供商配置工具将上一个步骤中的别名目标添加到域 DNS 记录中。您的 DNS 提供商还需要为您的自定义域设置子域。

使用 Route 53 添加别名目标和子域

1. 登录 Route 53 控制台。如果出现提示，请输入 AWS 凭证。
2. 如果您在 Route 53 中没有托管区域，则必须按以下步骤创建一个托管区域。否则，请继续步骤 3。
   a. 选择 Create Hosted Zone（创建托管区域）。
   b. 从 Domain Name（域名）列表中输入自定义域（例如，*myapp.auth.example.com*）的父域（例如，*auth.example.com*）。
   c. {可选}对于 Comment（注释），输入有关该托管区域的注释。
   d. 选择 Public hosted zone（公有托管区域）的托管区域 Type（类型）以允许公共客户端解析您的自定义域。不支持选择 Private hosted zone（私有托管区域）。
   e. 根据需要应用 Tags（标签）。
   f. 选择 Create Hosted Zone（创建托管区域）。

      Note

      您还可以为自定义域创建新的托管区域，并且可以在父托管区域中创建委托集，将查询指向子域托管区域。此方法为您的托管区域提供了更大的灵活度和安全性。有关更多信息，请参阅通过 Amazon Route 53 为托管域创建子域。
3. 在 Hosted Zones（托管区域）页面上，选择您的托管区域的名称。
4. 选择 Create record（创建记录）。为父域添加 DNS 记录，然后选择 Create records（创建记录）。以下是域 *auth.example.com* 的示例记录。

   ```
   auth.example.com. 60 IN A 198.51.100.1
   ```

Note

Amazon Cognito 验证您的自定义域的父域是否有 DNS 记录，以防止意外劫持生产域。如果您没有父域的 DNS 记录，则当您尝试设置自定义域时，Amazon Cognito 将返回错误消息。

5. 再次选择 Create Record（创建记录）。
6. 输入与您的自定义域相匹配的 Record name（记录名称）（例如，*myapp*）以创建 *myapp.auth.example.com* 记录。
7. 将先前记下的别名目标名称输入到 Alias Target（别名目标）。
8. 启用 Alias（别名）选项。
9. 选择 Route traffic to（将流量路由至）、Alias to CloudFront distribution（别名到 CloudFront 分配）。输入由 Amazon Cognito 在您创建自定义域时提供的 Alias target（别名目标）。
10. 选择 Create Records（创建记录）。

Note

新记录可能需要大约 60 秒钟才能传播到所有 Route 53 DNS 服务器。您可以使用 Route 53 GetChange API 方法验证更改是否已传播。

步骤 3：验证登录页面

- 验证登录页面是否可从您的自定义域访问。

  通过在浏览器中输入此地址，使用您的自定义域和子域进行登录。这是包含子域 *auth* 的自定义域 *example.com* 的示例 URL：

```
https://myapp.auth.example.com/login?
response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
```

更改自定义域的 SSL 证书

如果需要，您可以使用 Amazon Cognito 更改应用于自定义域的证书。

通常，在使用 ACM 进行常规证书续订后，此操作是不必要的。当您续订 ACM 中的现有证书时，证书的 ARN 保持不变，并且您的自定义域将自动使用新证书。

但是，如果您将现有证书替换为新证书，ACM 将为新证书提供一个新 ARN。要将新证书应用于自定义域，您必须将此 ARN 提供给 Amazon Cognito。

在提供新证书后，Amazon Cognito 需要长达 1 小时才能将它分配给自定义域。

开始前的准备工作

您必须先将证书添加到 Amazon Cognito，然后才能更改 ACM 中的证书。有关更多信息，请参阅《AWS Certificate Manager 用户指南》中的入门。
将您的证书添加到 ACM 时，您必须选择美国东部（弗吉尼亚北部）作为AWS区域。

您可以使用 Amazon Cognito 控制台或 API 更改证书。

AWS Management Console

更新 Amazon Cognito 控制台的证书：

1. 登录AWS Management Console并打开 Amazon Cognito 控制台 (https://console.aws.amazon.com/cognito/home)。
2. 选择 User Pools（用户池）。

3. 选择要更新其证书的用户池。
4. 选择 App integration（应用程序集成）选项卡。
5. 选择 Actions（操作）、Edit ACM certificate（编辑 ACM 证书）。
6. 选择您希望与自定义域关联的新证书。
7. 选择 Save changes（保存更改）。

API

如要更新证书（Amazon Cognito API）

- 使用 UpdateUserPoolDomain 操作。

自定义内置登录网页和注册网页

您可以使用 AWS Management Console、AWS CLI 或 API 为内置应用程序 UI 体验指定自定义设置。您可以上传要显示在应用程序中的自定义徽标图像。您还可以使用 Cascading 样式表（CSS）来自定义 UI 的外观。

您可以为单个客户端 (具有特定的 `clientId`) 或为所有客户端 (通过将 `clientId` 设置为 `ALL`) 指定应用程序 UI 自定义设置。如果您指定`ALL`，则将对之前未设置 UI 自定义项的所有客户端使用默认配置。如果您为某个特定客户端指定了 UI 自定义设置，则它无法再回退到 `ALL` 配置。

设置 UI 定制设置的请求的大小不得超过 135 KB。在极少数情况下，请求标头、您的 CSS 文件和徽标的大小总和可能超过 135 KB。Amazon Cognito 对图像文件进行 Base64 编码。这会将 100KB 大小的图像增加到 130KB，保留 5KB 用于请求标头和 CSS。如果请求太大，则 AWS Management Console 或您的 `SetUICustomization` API 请求会返回 `request parameters too large` 错误。请将您的徽标图像调整为不超过 100 KB，将 CSS 文件调整为不超过 3 KB。您无法单独设置 CSS 和徽标定制设置。

Note

要自定义 UI，必须为用户群体设置域。

为应用程序指定自定义徽标

Amazon Cognito 将您的自定义徽标放在 登录端点 (p. 552) 的输入字段上方居中。

为自定义托管 UI 徽标选择可缩放至 350 x 178 像素的 PNG、JPG 或 JPEG 文件。徽标文件的大小不得超过 100KB，或者在 Amazon Cognito 编码为 Base64 之后不超过 130KB。要在 API 的 SetUICustomization 中设置 `ImageFile`，请将文件转换为 Base64 编码的文本字符串，或者在 AWS CLI 中提供文件路径并让 Amazon Cognito 为您编码。

为应用程序指定 CSS 自定义项

您可以为托管应用程序页面自定义 CSS，但存在以下限制：

- 您可以使用以下任意 CSS 类名：
  - `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`
- 属性值可以包含 HTML，但以下值除外：`@import`、`@supports`、`@page` 或者 `@media` 语句以及 Javascript。

您可以自定义以下 CSS 属性。

Labels
  - font-weight 是 100 的倍数 (从 100 到 900)。

输入字段
  - width 是以占包含块大小的百分比形式表示的宽度。
  - height 是输入字段的高度，以像素 (px) 为单位。
  - color 是文本颜色。它可以是任何标准 CSS 颜色值。
  - background-color 是输入字段的背景色。它可以是任何标准 CSS 颜色值。
  - border 是标准 CSS 边框值，用于指定您的应用程序窗口边框的宽度、透明度和颜色。宽度可以是从 1px 到 100px 的任何值。透明度可以是完全透明或不透明。颜色可以是任何标准颜色值。

文本描述
  - padding-top 是文本描述上方的填充量。
  - padding-bottom 是文本描述下方的填充量。
  - display 可以是 `block` 或 `inline`。
  - font-size 是文本描述的字体大小。

提交按钮
  - font-size 是按钮文本的字体大小。
  - font-weight 是按钮文本的字体粗细：`bold`、`italic` 或 `normal`。
  - 页边距是一个由四个值组成的字符串，用于指示按钮的上边距、右边距、下边距和左边距大小。
  - font-size 是文本描述的字体大小。
  - width 是按钮文本的宽度，以占包含块大小的百分比形式表示。
  - height 是按钮的高度，以像素 (px) 为单位。
  - color 是按钮文本颜色。它可以是任何标准 CSS 颜色值。
  - background-color 是按钮的背景色。它可以是任何标准颜色值。

横幅
  - 填充是一个由四个值组成的字符串，用于指示横幅的上边距、右边距、下边距和左边距大小。
  - background-color 是横幅的背景色。它可以是任何标准 CSS 颜色值。

提交按钮悬停
  - color 是您将鼠标指针悬停在按钮上方时按钮的前景色。它可以是任何标准 CSS 颜色值。
  - background-color 是您将鼠标指针悬停在按钮上方时按钮的背景色。它可以是任何标准 CSS 颜色值。

身份提供商按钮悬停
  - color 是您将鼠标指针悬停在按钮上方时按钮的前景色。它可以是任何标准 CSS 颜色值。

- background-color 是您将鼠标指针悬停在按钮上方时按钮的背景色。它可以是任何标准 CSS 颜色值。

密码校验无效

- color 是 "Password check not valid" 消息的文本颜色。它可以是任何标准 CSS 颜色值。

背景

- background-color 是应用程序窗口的背景色。它可以是任何标准 CSS 颜色值。

错误消息

- 页边距是一个由四个值组成的字符串，用于指示上边距、右边距、下边距和左边距大小。
- padding 是边距大小。
- font-size 是字体大小。
- width 是错误消息的宽度，以占包含块大小的百分比形式表示。
- background 是错误消息的背景色。它可以是任何标准 CSS 颜色值。
- 边框是一个由三个值组成的字符串，用于指定边框的宽度、透明度和颜色。
- color 是错误消息文本颜色。它可以是任何标准 CSS 颜色值。
- box-sizing 用于向浏览器指示应包含的大小属性 (宽度和高度)。

身份提供商按钮

- height 是按钮的高度，以像素 (px) 为单位。
- width 是按钮文本的宽度，以占包含块大小的百分比形式表示。
- text-align 是文本对齐设置。它可以是 `left`、`right` 或 `center`。
- margin-bottom 是下边距设置。
- color 是按钮文本颜色。它可以是任何标准 CSS 颜色值。
- background-color 是按钮的背景色。它可以是任何标准 CSS 颜色值。
- border-color 是按钮边框的颜色。它可以是任何标准 CSS 颜色值。

身份提供商描述

- padding-top 是描述上方的填充量。
- padding-bottom 是描述下方的填充量。
- display 可以是 `block` 或 `inline`。
- font-size 是描述的字体大小。

法律文本

- color 是文本颜色。它可以是任何标准 CSS 颜色值。
- font-size 是字体大小。

  Note

  自定义 Legal text（法律文本）时，您自定义的是 We won't post to any of your accounts without asking first（如果未先询问，我们不会发布到您的任何账户）的消息，该消息显示在登录页面的社交身份提供商下方。

徽标

- max-width 是以占包含块大小的百分比形式表示的最大宽度。
- max-height 是以占包含块大小的百分比形式表示的最大高度。

输入字段聚焦

- border-color 是输入字段的颜色。它可以是任何标准 CSS 颜色值。
- outline 是输入字段的边框宽度 (以像素为单位)。

社交按钮

- height 是按钮的高度，以像素 (px) 为单位。
- text-align 是文本对齐设置。它可以是 `left`、`right` 或 `center`。
- width 是按钮文本的宽度，以占包含块大小的百分比形式表示。
- margin-bottom 是下边距设置。

密码校验有效

- color 是 "Password check valid" 消息的文本颜色。它可以是任何标准 CSS 颜色值。

为用户池指定应用程序 UI 自定义设置（AWS Management Console）

您可以使用 AWS Management Console为应用程序指定 UI 自定义设置。

> Note
>
> 通过利用您的用户池的特定信息构建以下 URL 并将它键入到浏览器中，您可以查看具有自定义项的托管 UI： `https://<your_domain>/login?response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>`
> 在控制台中进行的更改出现之前，您可能必须等待长达 1 分钟才能刷新浏览器。
> 您的域显示在 App integration（应用程序集成）选项卡中，该选项卡位于 Domain（域）下方。您的应用程序客户端 ID 和回调 URL 在 App clients（应用程序客户端）下显示。

指定应用程序 UI 自定义设置

1. 登录 Amazon Cognito 控制台。
2. 在导航窗格中，选择 User Pools（用户池），然后选择要编辑的用户池。
3. 选择 App integration（应用程序集成）选项卡。
4. 要为所有应用程序客户端自定义 UI 设置，请找到 Hosted UI customization（托管 UI 自定义），然后选择 Edit（编辑）。
5. 要为一个应用程序客户端自定义 UI 设置，请找到应用程序客户端，并选择要修改的应用程序客户端，然后找到托管 UI 自定义并选择编辑。要将应用程序客户端从用户池默认自定义切换为特定于客户端的自定义，请选择 Use client-level settings（使用客户端级别设置）。
6. 要上载自己的徽标图像文件，请选择 Choose file（选择文件）或 Replace current file（替换当前文件）。
7. 要自定义托管 UI CSS，请下载 CSS template.css，然后使用您想要自定义的值修改模板。只有模板中包含的密钥才能与托管 UI 一起使用。添加的 CSS 密钥不会反映在您的 UI 中。自定义 CSS 文件后，请选择 Choose file（选择文件）或 Replace current file（替换当前文件）来上载您的自定义 CSS 文件。

为用户池指定应用程序 UI 自定义设置（AWS CLI 和 AWS API）

使用以下命令可为您的用户池指定应用程序 UI 自定义项。

使用以下 API 操作，获取用户群体的内置应用程序 UI 的 UI 定制设置。

- AWS CLI: `aws cognito-idp get-ui-customization`
- AWS API：GetUICustomization

使用以下 API 操作，设置用户群体的内置应用程序 UI 的 UI 定制设置。

- AWS CLI，从图像文件：`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 及编码为 Base64 二进制文本的图像：`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

使用托管 UI 注册和登录

在您为用户群体和应用程序客户端配置和自定义 Amazon Cognito 托管 UI 后，您的应用程序可以将其呈现给用户。托管 UI 支持多个 Amazon Cognito 身份验证操作，包括以下示例。

- 在应用程序中注册为新用户
- 验证电子邮件地址或电话号码
- 设置多重身份验证 (MFA)
- 使用本地用户名和密码登录
- 使用第三方身份提供者 (IdP) 登录
- 重置密码

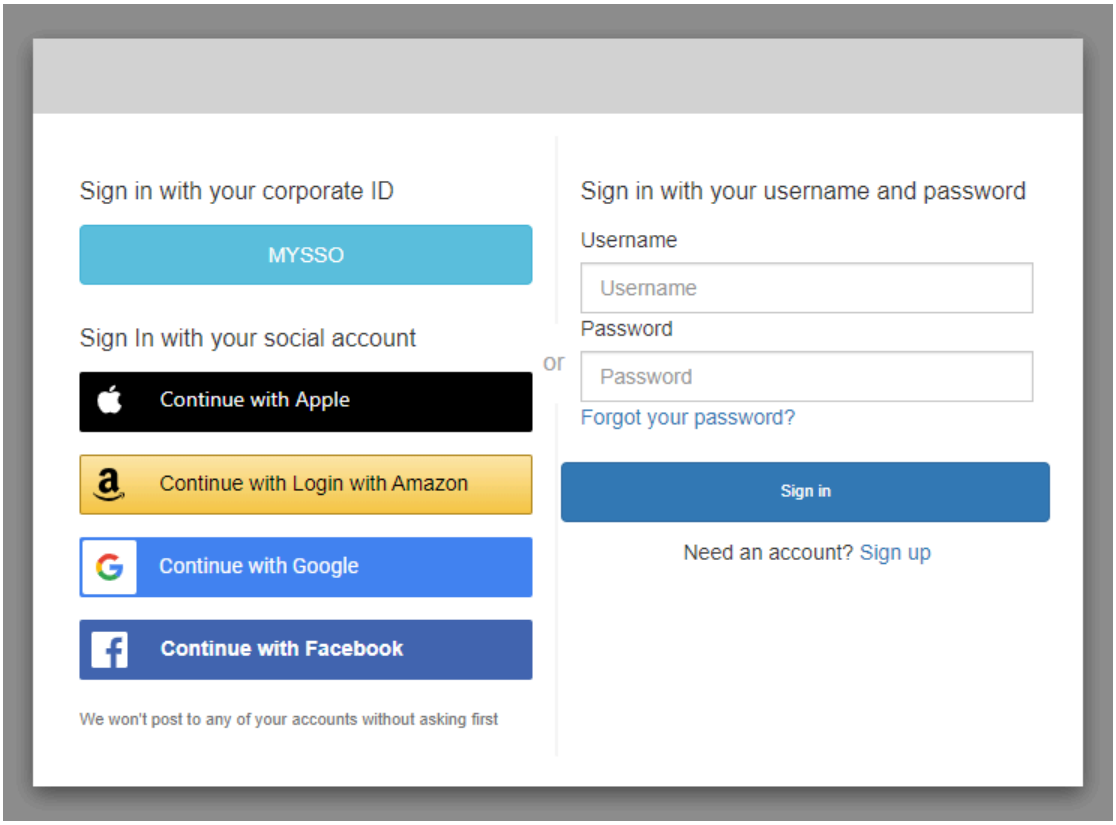
Amazon Cognito 托管 UI 从 登录端点 (p. 557)开始。登录页面的 URL 是您为用户群体选择的域与反映您希望发出的 OAuth 2.0 授予、您的应用程序客户端、您的应用程序的路径和您希望请求的 OpenID Connect (OIDC) 范围的参数的组合。

```
https://<your user pool domain>/authorize?client_id=<your app client
 ID>&response_type=<code/token>&scope=<scopes to request>&redirect_uri=<your callback URL>
```

以下 URL 将上面的占位符字段替换为示例值。

```
https://auth.example.com/authorize? /
client_id=1example23456789 /
&response_type=code /
&scope=aws.cognito.signin.user.admin+email+openid+profile /
&redirect_uri=https%3A%2F%2Faws.amazon.com
```

Amazon Cognito 托管 UI 的登录页面提供了通过用户群体或您分配给您的用户请求的应用程序客户端的任何身份提供者 (IdP) 登录的选项。它还包括用于在用户群体中注册新用户账户或重置已忘记的密码的链接。



主题

- 如何在 Amazon Cognito 托管 UI 中注册新账户 (p. 72)
- 如何使用 Amazon Cognito 托管 UI 登录 (p. 77)
- 如何使用 Amazon Cognito 托管 UI 重置密码 (p. 80)

如何在 Amazon Cognito 托管 UI 中注册新账户

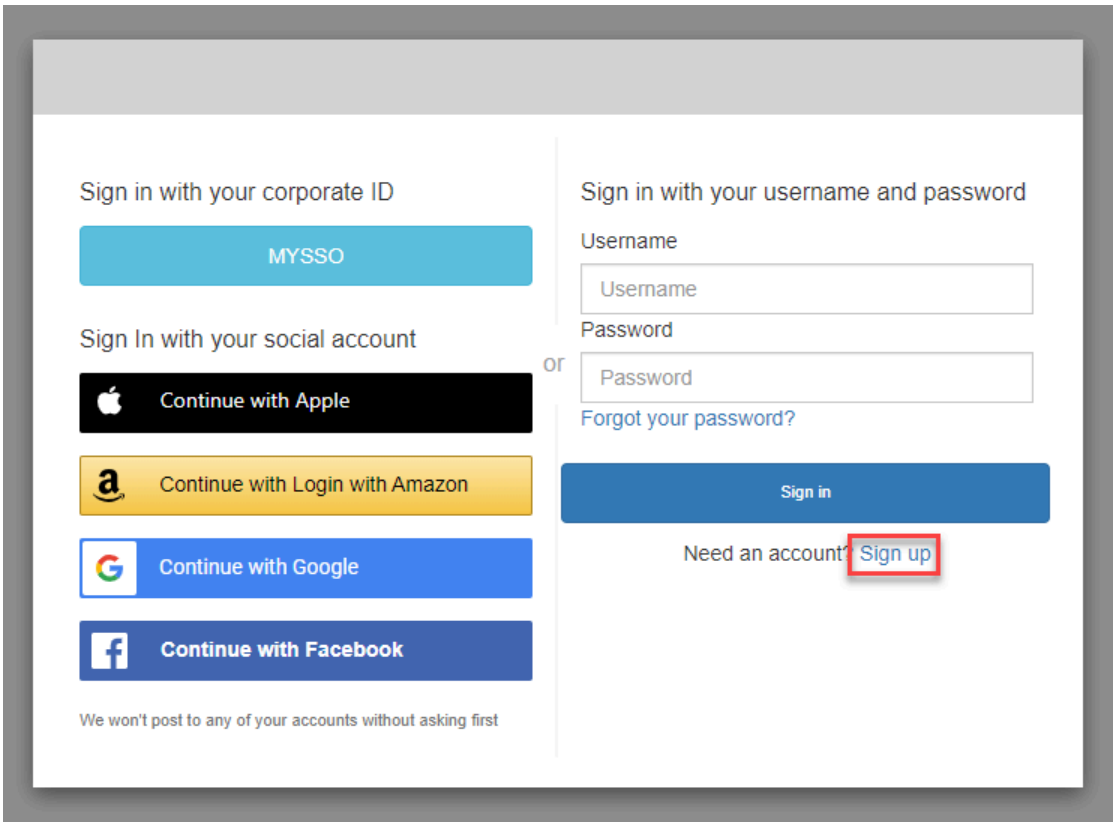
本指南向您介绍如何在使用 Amazon Cognito 的应用程序中注册用户账户。

> Note
>
> 当您登录到使用 Amazon Cognito 托管用户界面 (UI) 的应用程序时，可能会看到应用程序所有者在本指南介绍的基本配置之外自定义的页面。

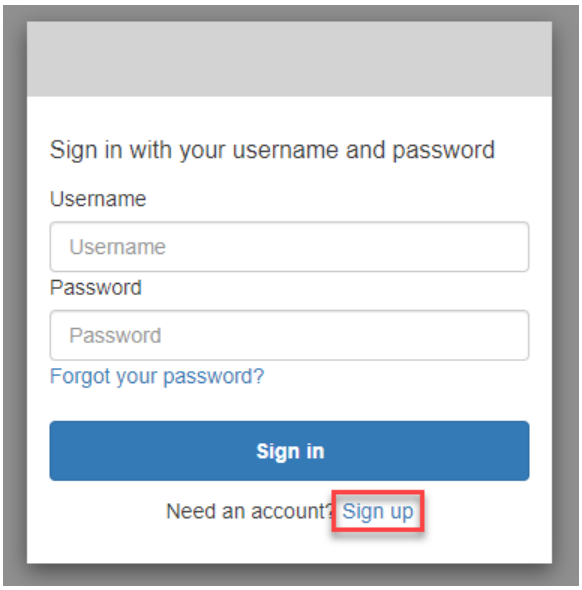
1. 如果您打算使用用户名和密码通过 Amazon Cognito 登录，而不是通过应用程序所有者列出的第三方登录提供商之一登录，请从登录页面中选择 Sign up（注册）。

   如果您的登录提供商不是 Amazon Cognito，则在您选择第三方提供商的按钮后，注册即完成。根据应用程序所有者选择的选项，您可能可以选择用于登录的提供商，也可能只看到输入用户名和密码的提示。

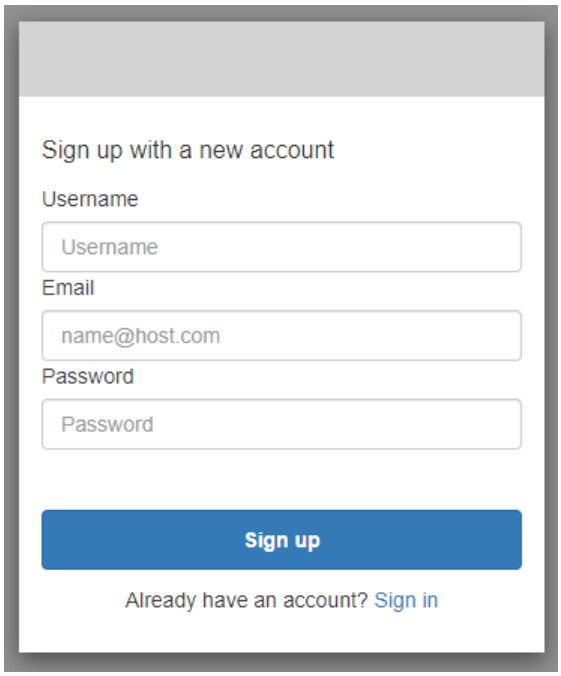
With multiple sign-in providers



With only Amazon Cognito as a sign-in provider

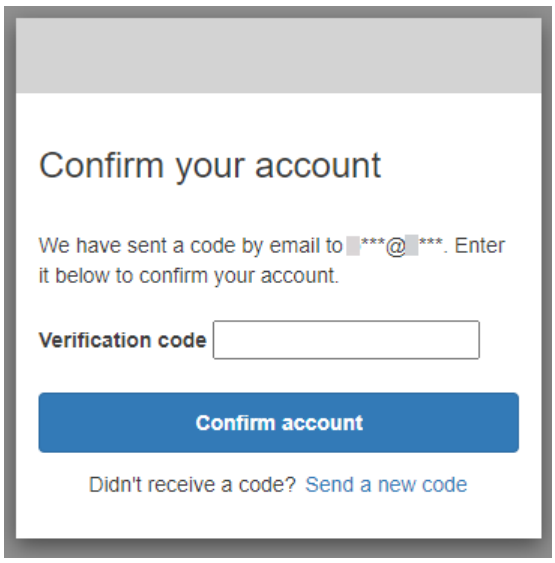


2. 在 Sign up with a new account（使用新账户注册）页面上，应用程序所有者要求提供注册所需的信息。他们可能会要求提供用户名、电子邮件地址或电话号码。输入所需信息，然后选择密码。



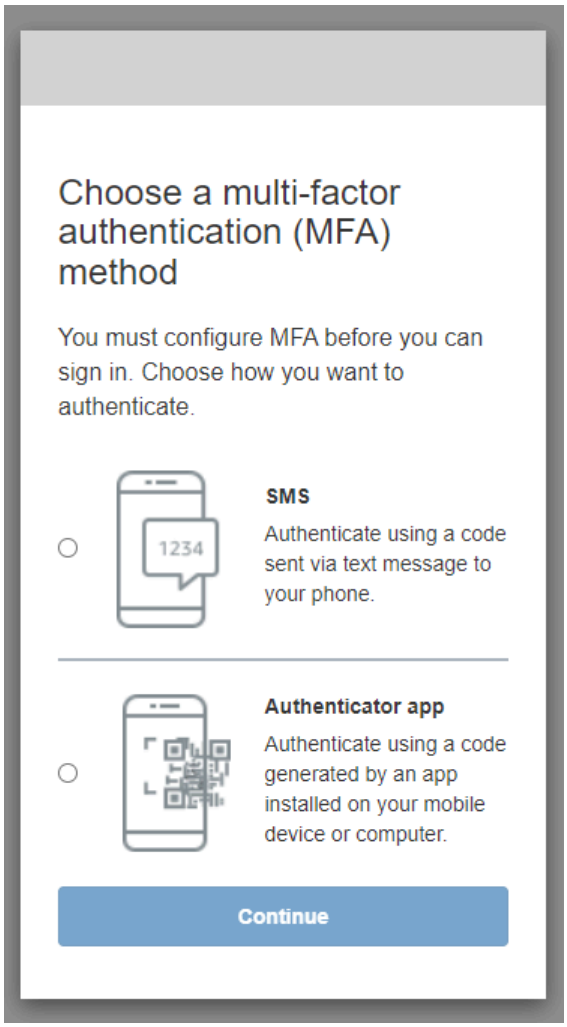
3. 在 Confirm your account（确认您的账户）页面上，应用程序所有者可能会要求您确认账户，以验证您是否可以通过您提供的电子邮件地址或电话号码接收消息。

   您将在电子邮件或 SMS 文本消息中收到代码。在表单中输入代码以确认您输入了正确的联系信息。



4. 应用程序所有者可能会要求您设置多重身份验证 (MFA)。您可能会看到选择 MFA 方法的提示，或者您的应用程序可能会跳到下一步。

   在 Choose a multi-factor authentication (MFA) method [选择多重身份验证(MFA)方法] 页面上，选择 MFA 方法。如果您选择 SMS，则会在 SMS 文本消息中收到 MFA 密码。如果您选择 Authenticator app（身份验证器应用程序），则必须在设备上安装应用程序才能生成基于时间的 MFA 密码。您必须在 3 分钟内做出选择。

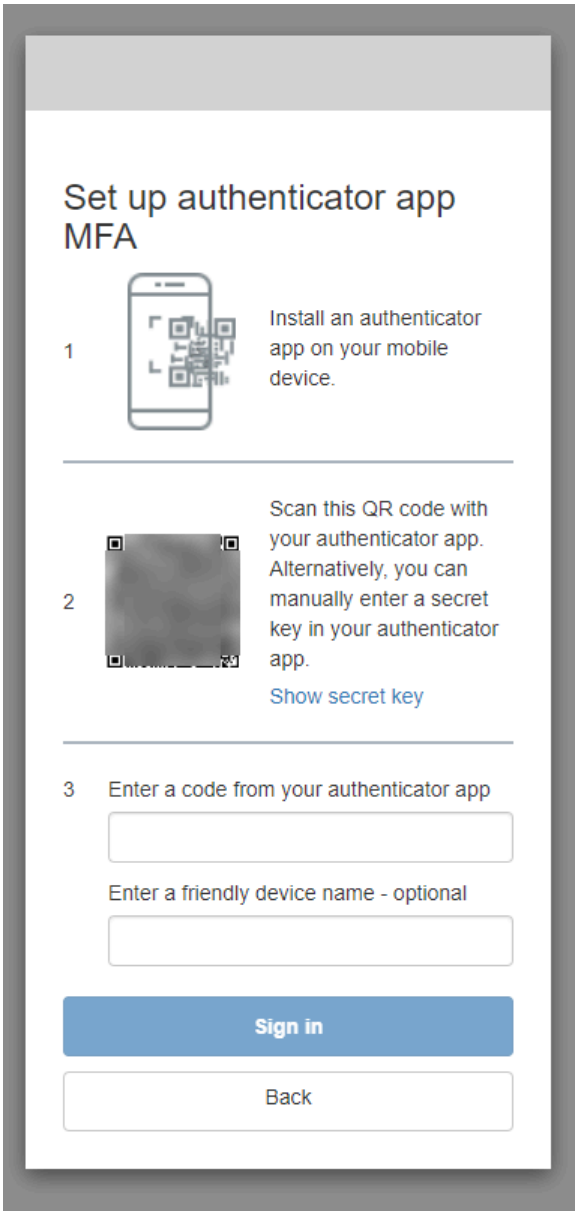


5. Amazon Cognito 要求您提供来自身份验证器应用程序或 SMS 文本消息的代码。在 3 分钟内输入您收到的代码。

   Authenticator app

   1. 打开您下载的身份验证器应用程序。
   2. 用摄像头扫描页面上的 QR 代码。您可能需要授权应用程序使用您的摄像头。

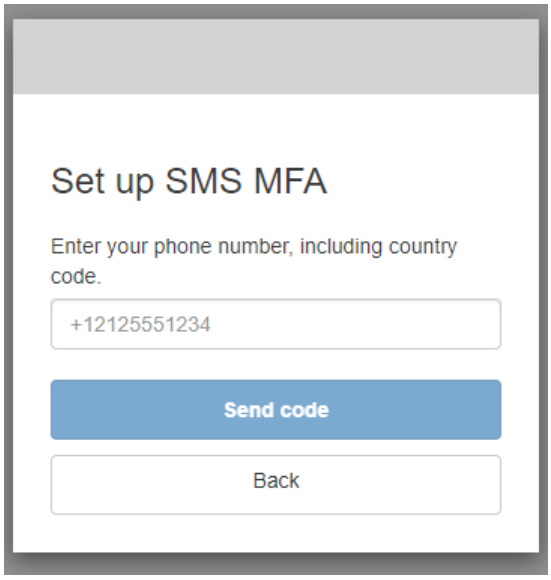
      如果您无法扫描 QR 代码，请选择 Show secret key（显示密钥）来显示可手动输入到身份验证器应用程序中的代码。
   3. 您的身份验证器应用程序开始显示每隔几秒钟更改一次的代码。输入应用程序中的当前代码。
   4. （可选）在 Set up authenticator app MFA（设置身份验证器应用程序 MFA）页面上，为您的设备选择名称。当您登录时，Amazon Cognito 会要求您提供具有您在此处提供的名称的设备中的代码。



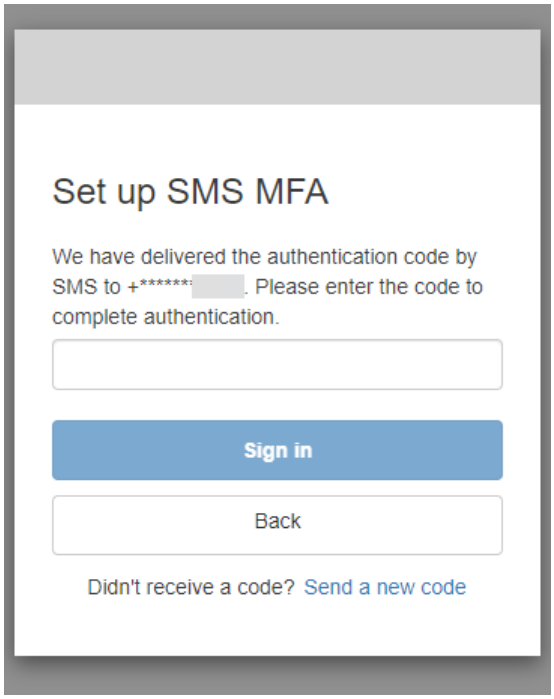
SMS text message

1. 如果应用程序所有者尚未收集您的电话号码，Amazon Cognito 会请求您提供电话号码。

   在 Set up SMS MFA（设置 SMS MFA）页面上，输入一个包含 + 符号和国家/地区代码的电话号码，例如 +12125551234。



2. 您会收到包含代码的 SMS 消息。在 Set up SMS MFA（设置 SMS MFA）页面上，输入代码。如果您没有收到代码但想重试，请选择 Send a new code（发送新代码）。选择 Back（返回）来输入新的电话号码。



6. 当您首次注册并确认您的详细信息时，Amazon Cognito 会在您完成此过程后授予对您的应用程序的访问权限。

如何使用 Amazon Cognito 托管 UI 登录

本指南向您介绍如何登录到使用 Amazon Cognito 的应用程序。

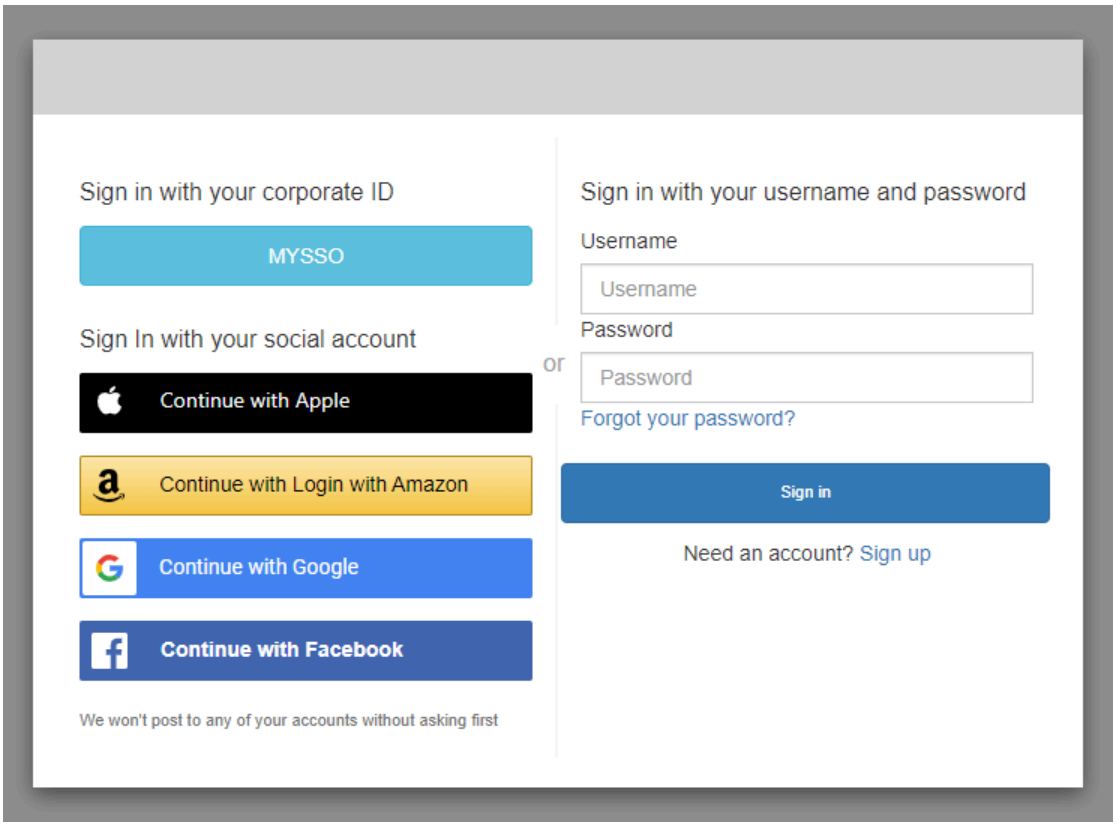
Note

当您登录到使用 Amazon Cognito 托管用户界面 (UI) 的应用程序时，可能会看到应用程序所有者在本指南介绍的基本配置之外自定义的页面。

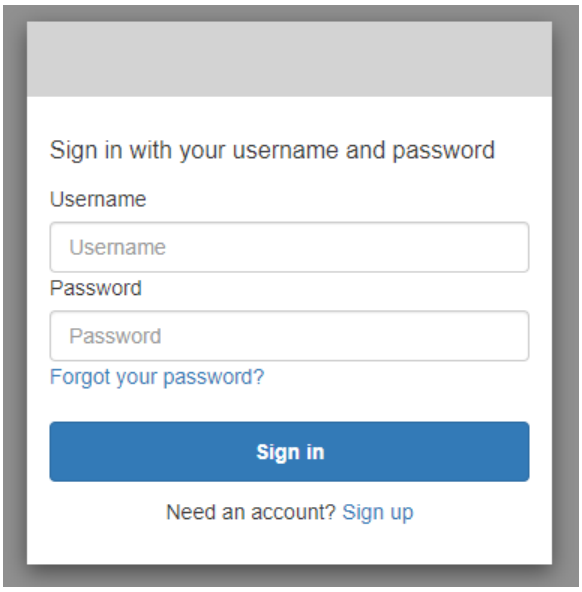
1. 根据应用程序所有者选择的选项，您可能可以选择用于登录的提供商，也可能只看到输入用户名和密码的提示。当您从此页面使用用户名和密码登录时，Amazon Cognito 是您的登录提供商。否则，您的登录提供商将由您选择的按钮表示。

   您可以在此处选择提供商，或者输入用户名和密码，然后立即获得对您的应用程序的访问权限。如果 Amazon Cognito 是您的登录提供商，则应用程序所有者可能还需要多重身份验证。

   With multiple sign-in providers

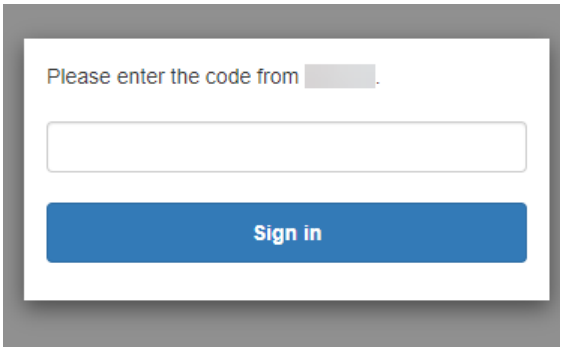


With only Amazon Cognito as a sign-in provider

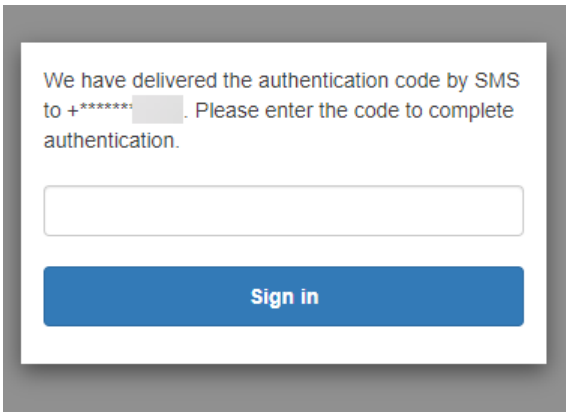


2. 您可能在应用程序中注册时已设置了 MFA。输入您在 SMS 消息中收到的或显示在身份验证器应用程序中的 MFA 代码。您必须在 3 分钟内输入此代码。

   With an authenticator app



   With an SMS code



3. 在您登录并完成 MFA 后，Amazon Cognito 会授予对您的应用程序的访问权限。

如何使用 Amazon Cognito 托管 UI 重置密码

本指南向您介绍如何在使用 Amazon Cognito 的应用程序中重置密码。

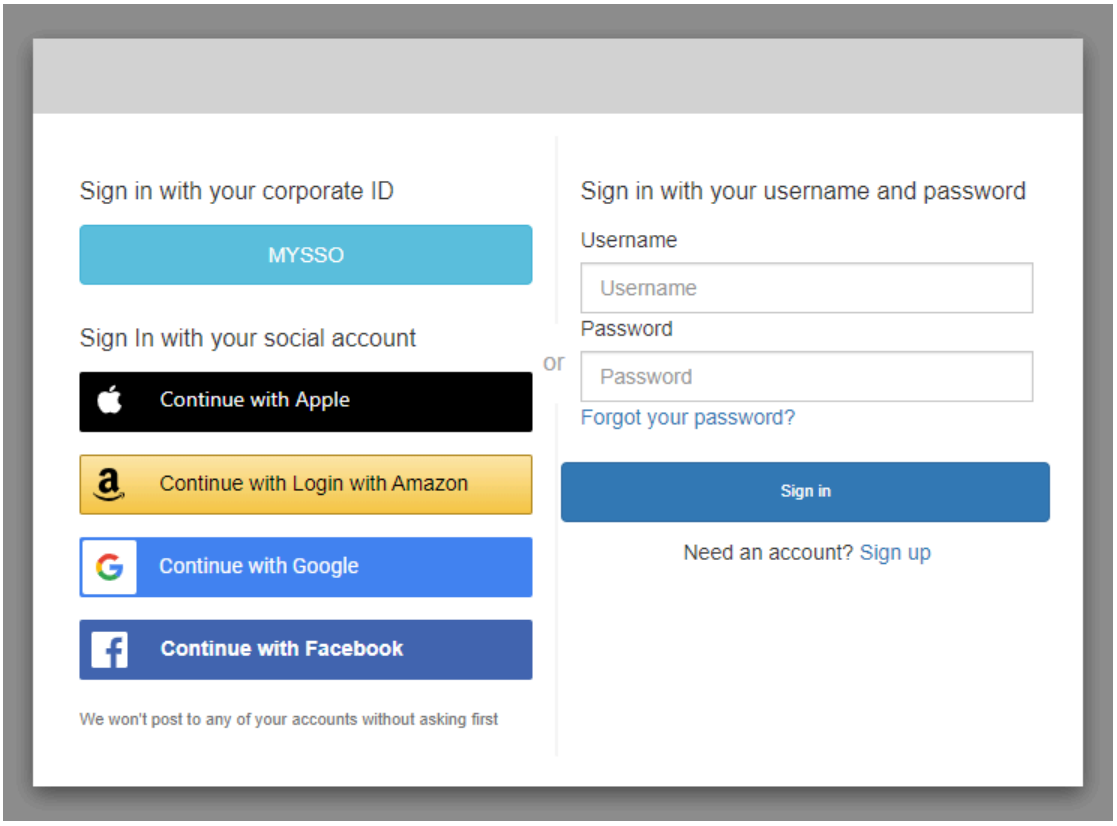
Note

当您登录到使用 Amazon Cognito 托管用户界面 (UI) 的应用程序时，可能会看到应用程序所有者在本指南介绍的基本配置之外自定义的页面。

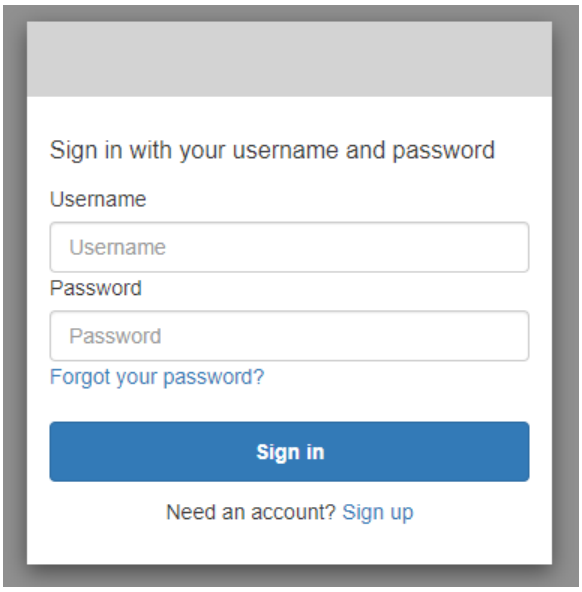
1. 根据应用程序所有者选择的选项，您可能可以选择用于登录的提供商，也可能只看到输入用户名和密码的提示。当您从此页面使用用户名和密码登录时，Amazon Cognito 是您的登录提供商。否则，您的登录提供商将由您选择的按钮表示。

   如果您通常从登录页面选择提供商，但您的密码无效，请通过提供商按照过程重置密码。如果 Amazon Cognito 是您的登录提供商，请选择 Forgot your password? (忘记密码？)

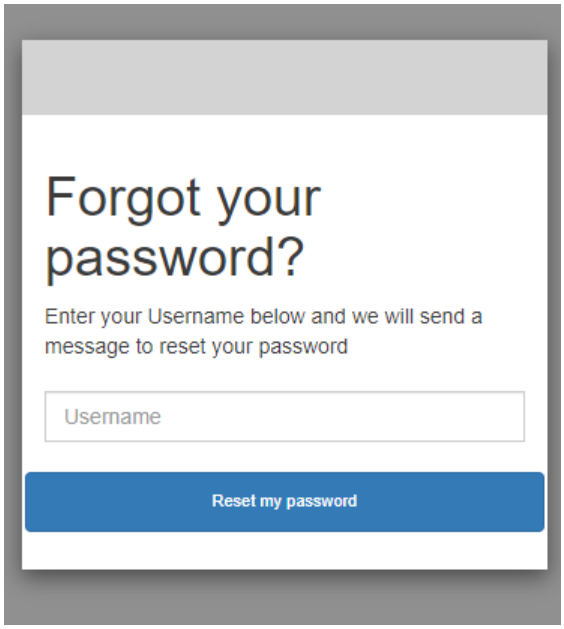
   With multiple sign-in providers



With only Amazon Cognito as a sign-in provider

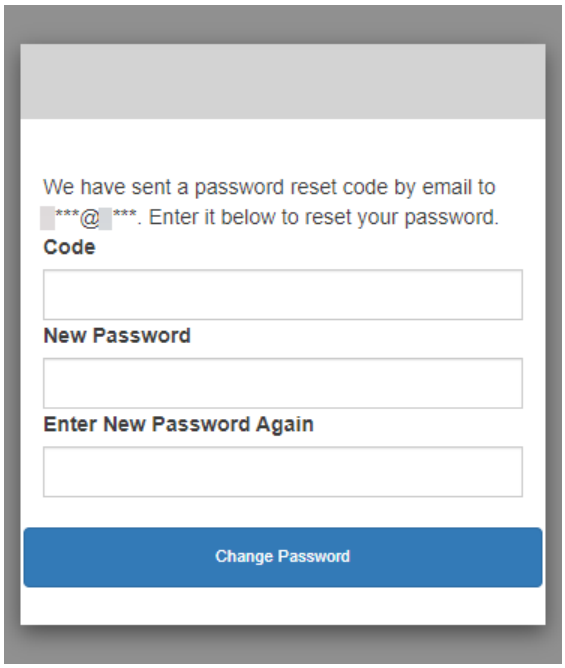


2. 在 Forgot your password? (忘记密码?) 页面上，Amazon Cognito 会提示您输入用于登录的信息。这可能是您的用户名、电子邮件地址或电话号码。



3. Amazon Cognito 将以电子邮件或 SMS 文本消息形式向您发送代码。

   输入您收到的代码，然后在提供的字段中输入两次新密码。您必须在 8 分钟内输入您的重置码。



4. 更改密码后，返回到登录页面并使用新密码登录。

为您的用户池定义资源服务器

在为用户群体配置域后，Amazon Cognito 会自动预置一个托管 Web UI，其中包含您的应用程序可提供给用户的注册页和登录页。有关更多信息，请参阅 第 2 步 添加应用程序客户端并设置托管 UI (p. 31)。

资源服务器 是 OAuth 2.0 API 服务器。为了保护访问受保护的资源，它会验证您的应用程序的访问令牌并授权访问您的 API。它根据令牌签名验证颁发者，根据令牌到期时间验证有效性，并根据令牌声明中的范围验证访问级别。

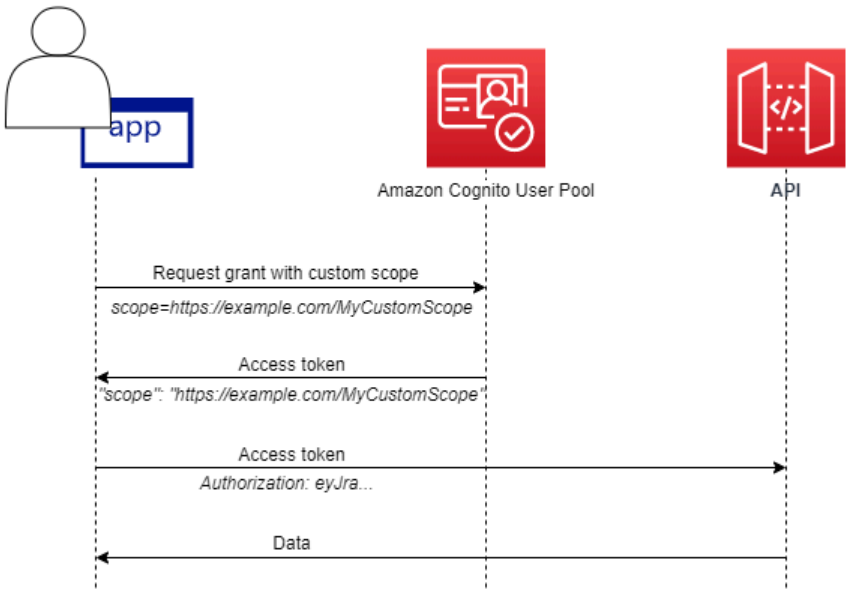
资源服务器 API 可能会授予对数据库中信息的访问权限，或者控制您的 IT 资源。Amazon Cognito 访问令牌可以授权访问支持 OAuth 2.0 的 API。Amazon API Gateway REST API 具有对使用 Amazon Cognito 访问令牌进行授权的内置支持。应用程序会将 API 调用中的访问令牌传递到资源服务器。当应用程序在访问请求中传递令牌时，它应该将访问令牌视为不透明的。资源服务器将检查访问令牌以确定是否应授予访问权限。

> Note
>
> 您的资源服务器在处理访问令牌内的任何声明之前必须验证访问令牌的签名和到期日期。有关验证和使用用户池令牌的更多信息，请参阅《AWS 博客》中的 Integrating Amazon Cognito User Pools with API Gateway。Amazon API Gateway 是用于检查访问令牌和保护您资源的一个很好的选择。有关 API Gateway Lambda 授权方的更多信息，请参阅使用 API Gateway Lambda 授权方。

范围是应用程序可请求的对资源的访问权限的级别。例如，如果您有一个照片资源服务器，它可能会定义两个范围：一个用于对照片的读取访问，另一个用于写入/删除访问。当应用程序发出请求访问权限的 API 调用并传递访问令牌时，该令牌将有一个或多个嵌入到其中的范围。

概述

利用 Amazon Cognito，应用程序开发人员可以创建自己的 OAuth2.0 资源服务器并将自定义范围与它们相关联。将自定义范围与应用程序客户端相关联，然后您的应用程序可以在来自 令牌端点 (p. 562) 的 OAuth2.0 授权代码授予、隐式授予和客户端凭证授予中请求这些范围。Amazon Cognito 在访问令牌中传递 `scope` 声明中的自定义范围。客户端可对其资源服务器使用访问令牌，然后服务器基于令牌中给出的范围做出授权决定。有关访问令牌范围的更多信息，请参阅将令牌与用户池结合使用 (p. 203)。



要获得具有自定义范围的访问令牌，您的应用程序必须向 令牌端点 (p. 562) 发出请求以兑换授权代码或请求客户端凭证授予。在托管 UI 中，您还可以通过隐式授予在访问令牌中请求自定义范围。

> **Note**
>
> 由于它们是为人机交互式身份验证而设计的，InitiateAuth 和 AdminInitiateAuth 请求生成不包含任何自定义范围的访问令牌。

管理资源服务器和自定义范围

在创建资源服务器时，您必须提供资源服务器名称和资源服务器标识符。对于您在资源服务器中创建的每个范围，您都必须提供范围名称和描述。

示例：

- `Name`：资源服务器的易记名称，如 `Weather API` 或 `Photo API`。
- `Identifier`：资源服务器的唯一标识符。这可能是您的资源服务器所在的 HTTPS 端点。例如，`https://my-weather-api.example.com`。
- `Scope Name`：范围名称。例如 `weather.read`
- `Scope Description`：范围的简要说明。例如，`Retrieve weather information`。

当客户端应用程序在任一 OAuth2.0 流中请求自定义范围时，它必须请求范围的完整标识符，即 *`resourceServerIdentifier`*`/`*`scopeName`*。例如，如果资源服务器标识符是 `https://myphotosapi.example.com`，范围名称是 `photos.read`，则客户端应用程序必须在运行时请求 `https://myphotosapi.example.com/photos.read`。

从资源服务器中删除一个范围不会删除其与所有客户端的关联；删除范围反而只会使其处于非活动状态。如果客户端应用程序在运行时请求已删除的范围，则该范围将被忽略，并且不会包含在访问令牌中。如果之后重新添加了该范围，它将会再次包含在访问令牌中。

如果从客户端中删除了某个范围，则会删除客户端与范围之间的关联。如果客户端在运行时请求了不允许使用的范围，则会返回错误并且不会颁发访问令牌。

您可以使用 AWS Management Console、API 或 CLI 为您的用户池定义资源服务器和范围。

为您的用户池定义资源服务器（AWS Management Console）

您可以使用 AWS Management Console为您的用户池定义资源服务器。

定义资源服务器

1. 登录 Amazon Cognito 控制台。
2. 在导航窗格中，选择 User Pools（用户池），然后选择要编辑的用户池。
3. 选择 App integration（应用程序集成）选项卡，然后查找 Resource servers（资源服务器）。
4. 选择 Create a resource server（创建资源服务器）。
5. 输入 Resource server name（资源服务器名称）。例如，`Photo Server`。
6. 输入 Resource server identifier（资源服务器识符）。例如，`com.example.photos`。
7. 输入您的资源的 Custom scopes（自定义范围），例如 `read` 和 `write`。
8. 对于每个 Scope name（范围名称），输入一个 Description（描述），如 `view your photos` 和 `update your photos`。
9. 选择 Create（创建）。

您的自定义范围可以在 App integration（应用程序集成）选项卡中进行审查，该选项卡位于 Custom scopes（自定义范围）列中的 Resource servers（资源服务）下。可以使用 App integration（应用程序集成）选项卡为应用程序客户端启用自定义范围，该选项卡位于 App clients（应用程序客户端）下。选择应用程序客户端，查找 Hosted UI settings（托管 UI 设置），然后选择 Edit（编辑）。添加 Custom scopes（自定义范围），然后选择 Save changes（保存更改）。

为您的用户池定义资源服务器（AWS CLI 和 AWS API）

使用以下命令可为您的用户池指定资源服务器设置。

创建资源服务器

- AWS CLI: `aws cognito-idp create-resource-server`
- AWS API：CreateResourceServer

获取有关您的资源服务器设置的信息

- AWS CLI: `aws cognito-idp describe-resource-server`
- AWS API：DescribeResourceServer

列出用户池的所有资源服务器的相关信息

- AWS CLI: `aws cognito-idp list-resource-servers`
- AWS API：ListResourceServers

删除资源服务器

- AWS CLI: `aws cognito-idp delete-resource-server`
- AWS API：DeleteResourceServer

更新资源服务器的设置

- AWS CLI: `aws cognito-idp update-resource-server`

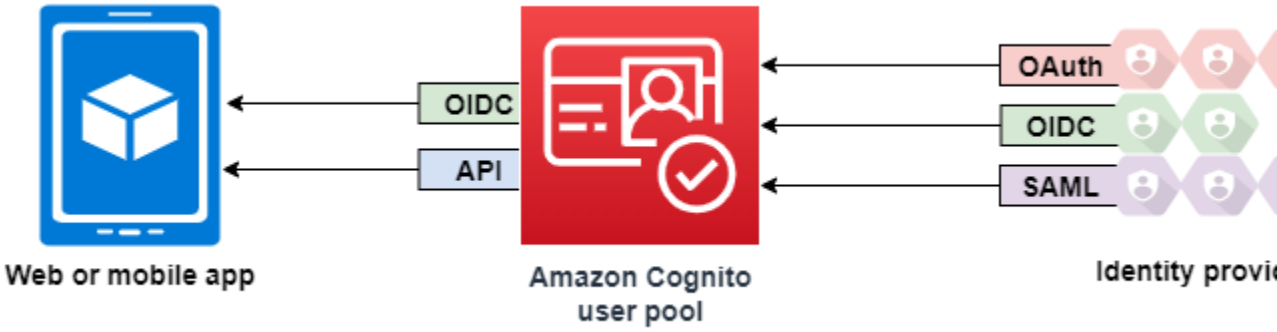
- AWS API：UpdateResourceServer

通过第三方添加用户池登录

您的应用程序用户可以通过用户池直接登录，也可以通过第三方身份提供商（IdP）联合。用户池管理处理以下令牌的开销：从通过 Facebook、Google、亚马逊和 Apple 进行的社交登录返回的令牌以及从 OpenID Connect（OIDC）和 SAML IdP 返回的令牌。利用内置托管 Web UI，Amazon Cognito 为来自所有 IdP 的经过身份验证的用户提供令牌处理和管理。这样，后端系统可以基于一组用户池令牌实现标准化。

联合登录在 Amazon Cognito 用户群体中的工作方式

通过第三方（联合身份验证）进行登录可在 Amazon Cognito 用户池中实现。此功能不依赖于通过 Amazon Cognito 身份池（联合身份）实现的联合身份验证。



Amazon Cognito 是用户目录和 OAuth 2.0 身份提供者 (IdP)。当您以本地用户身份登录 Amazon Cognito 目录时，您的用户群体是应用程序的 IdP。本地用户仅存在于您的用户群体目录中，无需通过外部 IdP 进行联合身份验证。

当您将 Amazon Cognito 连接到社交、SAML 或 OpenID Connect (OIDC) IdP 时，您的用户群体充当多个服务提供商和应用程序之间的桥梁。对于您的 IdP 而言，Amazon Cognito 是服务提供商 (SP)。您的 IdP 将 OIDC ID 令牌或 SAML 断言传递给 Amazon Cognito。Amazon Cognito 会在令牌或断言中读取有关您用户的声明，并将这些声明映射到用户群体目录中的新用户配置文件。

然后，Amazon Cognito 在其自己的目录中为联合用户创建用户配置文件。Amazon Cognito 根据来自 IdP 的声明向用户添加属性。对于 OIDC 和社交身份提供者，则向 IDP 运营的公有 `userinfo` 端点添加属性。当映射的 IdP 属性发生变化时，用户的属性会在用户群体中发生变化。您还可以添加更多属性，这些属性独立于 IdP 中的属性。

Amazon Cognito 为联合用户创建配置文件后，它会更改其功能并将自己显示为应用程序的 IdP（现在是 SP）。Amazon Cognito 是 OIDC 和 OAuth 2.0 IdP 的组合。它生成访问令牌、ID 令牌和刷新令牌。有关令牌的更多信息，请参阅将令牌与用户池结合使用 (p. 203)。

您必须设计一个与 Amazon Cognito 集成的应用程序，以便对用户进行身份验证和授权，无论是联合用户还是本地用户。

应用程序作为 Amazon Cognito 的服务提供者的责任

验证和处理令牌中的信息

在大多数情况下，Amazon Cognito 将经过身份验证的用户重新导向到它附加了授权码的应用程序 URL。您的应用程序将此代码交换为访问权限、ID 和刷新令牌。然后，它必须检查令牌的有效性，并根据令牌中的声明向用户提供信息。

使用 Amazon Cognito API 请求响应身份验证事件

您的应用程序必须与 Amazon Cognito 用户群体 API 和身份验证 API 端点集成。身份验证 API 会将用户进行登录和注销，并管理令牌。用户群体 API 具有多种操作，用于管理您的用户群体、用户以及身份验证环境的安全性。当应用程序收到来自 Amazon Cognito 的响应时，它必须知道下一步该怎么做。

关于 Amazon Cognito 用户群体第三方登录需要了解的事项

- 如果您希望用户使用联合提供商登录，则必须选择域。这将设置 Amazon Cognito 托管 UI 以及托管 UI 和 OIDC 端点。有关更多信息，请参阅将您自己的域用于托管 UI (p. 63)。
- 您不能使用 API 操作（如 InitiateAuth 和 AdminInitiateAuth）以联合用户身份登录。联合用户只能使用登录端点 (p. 552)或对端点授权 (p. 557)进行登录。
- 对端点授权 (p. 557) 是重定向端点。如果您在请求中提供 `idp_identifier` 或 `identity_provider` 参数，它将绕过托管 UI，以静默方式重定向到您的 IdP。否则，它会重定向到托管 UI 登录端点 (p. 552)。有关示例，请参阅 在企业控制面板中为 Amazon Cognito 应用程序添加书签 (p. 101)。
- 当托管 UI 将会话重定向到联合 IdP 时，Amazon Cognito 会在请求中包含 `user-agent` 标头 `Amazon/Cognito`。
- Amazon Cognito 从固定标识符和 IdP 名称的组合中派生联合用户配置文件的 `username` 属性。要生成符合自定义要求的用户名，请创建到 `preferred_username` 属性的映射。有关更多信息，请参阅有关映射的需知信息 (p. 107)。

  示例：`MyIDP_bob@example.com`
- Amazon Cognito 将有关联合用户身份的信息记录到属性中，并在 ID 令牌中记录一个称为 `identities` 的声明。此声明包含用户的提供商以及提供商提供的唯一 ID。您无法直接在用户配置文件中更改 `identities` 属性。有关如何关联联合用户的更多信息，请参阅将联合用户与现有用户配置文件关联 (p. 109)。
- 当您在 UpdateIdentityProvider API 请求中更新 IdP 时，您的更改最多可能需要一分钟才能显示在托管 UI 中。
- Amazon Cognito 支持在其自身与您的 IdP 之间最多 20 个 HTTP 重定向。
- 当您的用户使用托管 UI 登录时，用户的浏览器会存储一个加密的登录会话 cookie，用于记录用户登录时使用的客户端和提供者。如果用户尝试使用相同的参数再次登录，则托管 UI 会重复使用任何未过期的现有会话，并且用户无需再次提供凭证即可进行身份验证。如果用户使用不同的 IdP 再次登录，包括切换到本地用户群体登录或从本地用户群体登录进行切换，则他们必须提供凭证并生成新的登录会话。

  您可以将任何用户群体 IdP 分配给任何应用程序客户端，并且用户只能使用您分配给其应用程序客户端的 IdP 进行登录。

主题

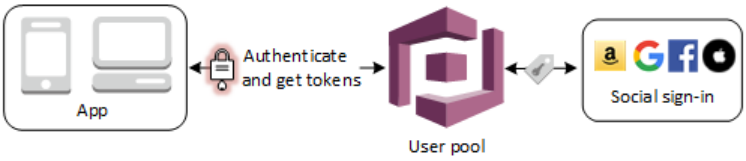
- 向用户池添加社交身份提供商 (p. 86)
- 向用户池添加 SAML 身份提供商 (p. 92)
- 向用户池添加 OIDC 身份提供商 (p. 102)
- 指定适用于用户池的身份提供商属性映射 (p. 107)
- 将联合用户与现有用户配置文件关联 (p. 109)

向用户池添加社交身份提供商

您的 Web 和移动应用程序用户可以通过社交身份提供商 (IdP)（例如 Facebook、Google、Amazon 和 Apple）进行登录。利用内置托管 Web UI，Amazon Cognito 将为所有经过身份验证的用户提供令牌处理和管理。这样，后端系统可以基于一组用户池令牌实现标准化。您必须启用托管 UI 才能与受支持的社交身份提

供商集成。当 Amazon Cognito 构建您的托管 UI 时，它会创建 Amazon Cognito 和您的 OIDC 以及社交 IdP 用来交换信息的 OAuth 2.0 终端节点。有关更多信息，请参阅 Amazon Cognito 用户池 Auth API 参考。

您可以在 AWS Management Console 中添加社交 IdP，也可以使用 AWS CLI 或 Amazon Cognito API。



Note

通过第三方（联合身份验证）进行登录可在 Amazon Cognito 用户池中实现。此功能不依赖于通过 Amazon Cognito 身份池（联合身份）实现的联合身份验证。

主题

- 先决条件 (p. 87)
- 步骤 1：向社交 IdP 注册 (p. 87)
- 步骤 2：将社交 IdP 添加到用户池 (p. 90)
- 步骤 3：测试社交 IdP 配置 (p. 91)

先决条件

在开始之前，您需要：

- 具有应用程序客户端和用户池域的用户池。有关更多信息，请参阅创建用户池。
- 社交 IdP。

步骤 1：向社交 IdP 注册

在使用 Amazon Cognito 创建社交 IdP 之前，必须向社交 IdP 注册应用程序才能接收客户端 ID 和客户端密钥。

向 Facebook 注册应用程序

1. 创建 Facebook 开发人员账户。
2. 使用 Facebook 凭证登录。
3. 在 My Apps（我的应用程序）菜单上，选择 Create New App（创建新的应用程序）。
4. 输入 Facebook 应用程序的名称，然后选择 Create App ID（创建应用程序 ID）。
5. 在左侧导航栏上，选择 Settings（设置），然后选择 Basic（基本）。
6. 记下 App ID（应用程序 ID）和 App Secret（应用程序密钥）。您将在下一节中使用它们。
7. 从页面底部选择 + Add Platform（+ 添加平台）。
8. 选择 Website（网站）。
9. 在 Website（网站）下，请在您的应用程序登录页面上将路径输入到 Site URL（站点 URL）中。

```
https://<your_user_pool_domain>/login?
response_type=code&client_id=<your_client_id>&redirect_uri=https://www.example.com
```

10. 选择 Save changes（保存更改）。
11. 在您的用户池域的根目录中，将路径输入到 App Domains（应用程序域）。

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

12. 选择 Save changes（保存更改）。
13. 从导航栏中，选择 Add Product（添加产品），然后选择 Facebook Login（Facebook 登录）中的 Set up（设置）。
14. 从导航栏中，选择 Facebook Login（Facebook 登录），然后选择 Settings（设置）。

    将指向用户群体域的 `/oauth2/idpresponse` 端点的路径输入到 Valid OAuth Redirect URIs（有效的 OAuth 重新导向 URI）中。

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

15. 选择 Save changes（保存更改）。

向 Amazon 注册应用程序

1. 创建 Amazon 开发人员账户。
2. 使用 Amazon 凭证登录。
3. 您需要创建一个 Amazon 安全配置文件才能接收 Amazon 客户端 ID 和客户端密钥。

   从页面顶部的导航栏中选择 Apps and Services（应用程序和服务），然后选择 Login with Amazon。
4. 选择 Create a Security Profile（创建安全配置文件）。
5. 输入 Security Profile Name（安全配置文件名称）、Security Profile Description（安全配置文件描述）和 Consent Privacy Notice URL（同意隐私声明 URL）。
6. 选择 Save（保存）。
7. 选择 Client ID（客户端 ID）和 Client Secret（客户端密钥）以显示客户端 ID 和密钥。您将在下一节中使用它们。
8. 将鼠标悬停在齿轮图标上并选择 Web Settings（Web 设置），然后选择 Edit（编辑）。
9. 将用户池域输入到 Allowed Origins（允许的源）中。

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

10. 将具有 `/oauth2/idpresponse` 端点的用户池域输入到 Allowed Return URLs（允许的返回 URL）中。

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

11. 选择 Save（保存）。

向 Google 注册应用程序

有关 Google Cloud 平台中的 OAuth 2.0 的更多信息，请参阅 Google Workspace for Developers 文档中的了解身份验证和授权。

1. 创建 Google 开发人员账户。
2. 登录到 Google Cloud Platform 控制台。
3. 从顶部导航栏中，选择 Select a project（选择项目）。如果您在 Google 平台中已经有项目，则此菜单会改为显示您的原定设置项目。
4. 选择 NEW PROJECT（新建项目）。
5. 输入产品的名称，然后选择 CREATE（创建）。

6. 在左侧导航栏上，选择 APIs and Services（API 和服务），然后选择 Oauth consent screen（Oauth 同意屏幕）。
7. 输入应用程序信息，App domain（应用程序域）、Authorized domains（已授权的域）和 Developer contact information（开发人员联系信息）。例如，您的 Authorized domains（已授权的域）必须包括 `amazoncognito.com` 和自定义域的根，例如 `example.com`。选择 SAVE AND CONTINUE（保存并继续）。
8. 1. 在 Scopes（范围）下，选择 Add or remove scopes（添加或删除范围），然后至少选择以下 OAuth 范围。

   1. `.../auth/userinfo.email`
   2. `.../auth/userinfo.profile`
   3. openid
9. 在 Test users（测试用户）下，选择 Add users（添加用户）。输入您的电子邮件地址和任何其他授权测试用户，然后选择 SAVE AND CONTINUE（保存并继续）。
10. 再次展开左侧导航栏，选择 APIs and Services（API 和服务），然后选择 Credentials（凭证）。
11. 依次选择 CREATE CREDENTIALS（创建凭证）和 OAuth client ID（OAuth 客户端 ID）。
12. 选择 Application type（应用程序类型）并为客户端提供 Name（名称）。
13. 在 Authorized JavaScript origins（已授权的 JavaScript 源）下，选择 ADD URI（添加 URI）。输入用户群体域。

    ```
    https://<your-user-pool-domain>
    ```
14. 在 Authorized redirect URIs（已授权的重新导向 URI）下，选择 ADD URI（添加 URI）。输入指向用户群体域的 `/oauth2/idpresponse` 端点的路径。

    ```
    https://<your-user-pool-domain>/oauth2/idpresponse
    ```
15. 选择 CREATE（创建）。
16. 安全地存储 Google 在 Your client ID（您的客户端 ID）和 Your client secret（您的客户端密钥）下显示的值。当您添加 Google IdP 时，请向 Amazon Cognito 提供这些值。

向 Apple 注册应用程序

有关设置“通过 Apple 登录”的更多信息，请参阅 Apple 开发人员文档中的配置环境以通过 Apple 登录。

1. 创建 Apple 开发人员账户。
2. 使用 Apple 凭证登录。
3. 在左侧导航栏上，选择 Certificates, Identifiers & Profiles（证书、标识符和配置文件）。
4. 在左侧导航栏上，选择 Identifiers (标识符)。
5. 在 Identifiers (标识符) 页面上，选择 + 图标。
6. 在 Register a New Identifier (注册新标识符) 页面上，选择 App IDs (应用程序 ID)，然后选择 Continue (继续)。
7. 在 Select a type（选择类型）页面上，选择 App（应用程序），然后选择 Continue（继续）。
8. 在 Register an App ID（注册应用程序 ID）页面上，执行以下操作：

   1. 在 Description（说明）下，输入说明。
   2. 在 App ID Prefix（应用程序 ID 前缀）下，输入 Bundle ID（捆绑包 ID）。记下 App ID Prefix（应用程序 ID 前缀）下的值。在步骤 2：将社交 IdP 添加到用户池 (p. 90)中选择 Apple 作为身份提供商后，您将使用此值。
   3. 在 Capabilities（功能）下，选择 Sign In with Apple，然后选择 Edit（编辑）。
   4. 在 Sign in with Apple: App ID Configuration（通过 Apple 登录：应用程序 ID 配置）页面上，选择将应用程序设置为主应用程序或与其他应用程序 ID 分组在一起，然后选择 Save（保存）。

5. 选择 Continue（继续）。

9. 在 Confirm your App ID（确认您的应用程序 ID）页面上，选择 Register（注册）。
10. 在 Identifiers (标识符) 页面上，选择 + 图标。
11. 在 Register a New Identifier（注册新标识符）页面上，选择 Services IDs（服务 ID），然后选择 Continue（继续）。
12. 在 Register an Services ID（注册服务 ID）页面上，执行以下操作：

    1. 在 Description（描述）下方，键入描述。
    2. 在 Identifier（标识符）下方，键入标识符。记下此服务 ID，因为在步骤 2：将社交 IdP 添加到用户池 (p. 90)中选择 Apple 作为身份提供商后需要此值。
    3. 选择 Continue（继续），然后选择 Register（注册）。
13. 从 Identifiers（标识符）页中选择您刚创建的 Services ID（服务 ID）。

    1. 选择 Sign In with Apple（使用苹果账号登录），然后选择 Configure（配置）。
    2. 在 Web Authentication Configuration（Web 身份验证配置）页上，选择您先前创建的应用程序 ID 作为 Primary App ID（主应用程序 ID）。
    3. 在 Website URLs（网站 URL）旁边选择 + 图标。
    4. 在 Domains and subdomains（域名和子域）下，输入不带 `https://` 前缀的用户群体域。

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

    5. 在 Return URLs（返回 URL）下，输入指向用户群体域的 `/oauth2/idpresponse` 端点的路径。

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

    6. 选择 Next（下一步），然后选择 Done（完成）。您不需要验证域。
    7. 选择 Continue（继续），然后选择 Save（保存）。
14. 在左侧导航栏上，选择 Keys (密钥)。
15. 在 Keys（密钥）页面上，选择 + 图标。
16. 在 Register a New Key（注册新密钥）页面上，执行以下操作：

    1. 在 Key Name（密钥名称）下，输入密钥名称。
    2. 选择 Sign In with Apple，然后选择 Configure（配置）。
    3. 在 Configure Key（配置密钥）页上，选择您先前创建的应用程序 ID 作为 Primary App ID（主应用程序 ID）。选择 Save（保存）。
    4. 选择 Continue（继续），然后选择 Register（注册）。
17. 在 Download Your Key（下载您的密钥）页面上，选择 Download（下载）以下载私有密钥并记下显示的 Key ID（密钥 ID），然后选择 Done（完成）。在步骤 2：将社交 IdP 添加到用户池 (p. 90)中选择 Apple 作为身份提供商后，您将需要此私有密钥和在此页面上显示的 Key ID（密钥 ID）值。

步骤 2：将社交 IdP 添加到用户池

使用 AWS Management Console配置用户池社交 IdP

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 Sign-in experience（登录体验）选项卡。找到 Federated sign-in（联合登录），然后选择 Add an identity provider（添加身份提供商）。
5. 选择一个社交 IdP：Facebook、Google（谷歌）、Login with Amazon（使用亚马逊账号登录）或 Sign in with Apple（使用苹果账号登录）。

6. 根据您选择的社交 IdP，从以下步骤中进行选择：

   - Google 和 Login with Amazon – 输入在上一部分中生成的 app client ID（应用程序客户端 ID）和 app client secret（应用程序客户端密钥）。
   - Facebook – 输入在上一部分中生成的 app client ID（应用程序客户端 ID）和 app client secret（应用程序客户端密钥），然后选择 API 版本（例如，版本 2.12）。我们建议选择最新的可用版本，因为每个 Facebook API 版本都有一个生命周期和一个弃用日期。Facebook 的范围和属性可能因 API 版本而异。我们建议您使用 Facebook 测试您的社交身份登录，以确保联合身份认证会按预期运行。
   - Sign In with Apple – 输入在上一部分中生成的 Services ID（服务 ID）、Team ID（团队 ID）、Key ID（密钥 ID）和 private key（私有密钥）。
7. 输入要使用的 Authorized scopes（授权范围）的名称。范围定义了您要通过应用程序访问的用户属性（如 `name` 和 `email`）。对于 Facebook，这些属性应用逗号分隔。对于 Google 和 Login with Amazon，则应采用空格分隔。对于 Sign in with Apple，选中要访问的范围的复选框。

社交身份提供商	示例范围
Facebook	`public_profile, email`
Google	`profile email openid`
Login with Amazon	`profile postal_code`
Sign in with Apple	`email name`

   您的应用程序用户需要同意向您的应用程序提供这些属性。关于社交服务提供商范围的更多信息，请参阅 Google、Facebook 和 Login with Amazon 或 Sign in with Apple 的文档。

   对于 Sign in with Apple，下面提供了可能不会返回范围的用户场景：

   - 终端用户离开 Apple 登录页面后出现故障（可能来自 Amazon Cognito 内部的故障或开发人员编写的任何内容）
   - 跨用户池和/或其它身份验证服务使用服务 ID 标识符
   - 在最终用户登录之前，开发人员添加了其他范围（未检索到新信息）
   - 开发人员删除用户，然后用户再次登录，而没有从其 Apple ID 个人资料中删除该应用程序
8. 请将 IdP 的属性映射到您的用户池。有关更多信息，请参阅指定适用于用户池的身份提供程序属性映射。
9. 选择 Create（创建）。
10. 从 App client integration（应用程序客户端集成）选项卡上的列表中选择一个 App clients（应用程序客户端），然后选择 Edit hosted UI settings（编辑托管 UI 设置）。将新的社交 IdP 添加到 Identity providers（身份提供商）下的应用程序客户端。
11. 选择 Save changes（保存更改）。

步骤 3：测试社交 IdP 配置

可以通过使用前两节中的元素来创建登录 URL。使用此 URL 测试社交 IdP 配置。

```
https://<your_user_pool_domain>/login?
response_type=code&client_id=<your_client_id>&redirect_uri=https://www.example.com
```

您可以在用户池 Domain name（域名）控制台页上找到您的域。client_id 位于 App client settings（应用程序客户端设置）页上。对于 redirect_uri 参数，使用您的回调 URL。这是页面的 URL，在页面中，您的用户在身份验证成功后将被重定向。

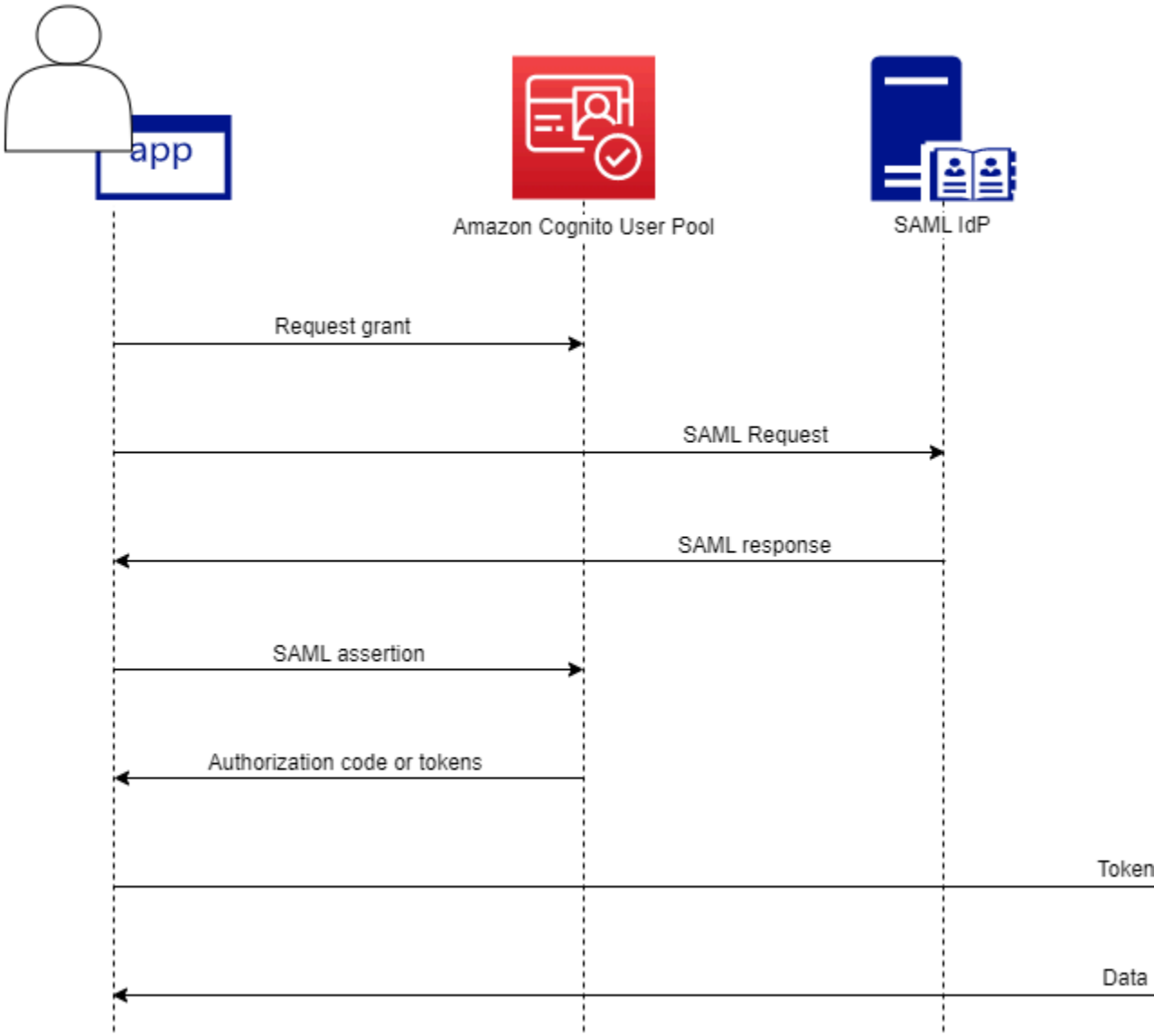
Note

Amazon Cognito 会取消未在 5 分钟内完成的身份验证请求，并将用户重定向到托管 UI。页面随即显示 `Something went wrong` 错误消息。

向用户池添加 SAML 身份提供商

可以选择允许 Web 和移动应用程序用户通过 SAML 身份提供程序 (IdP)（如 Microsoft Active Directory 联合身份验证服务 (ADFS) 或 Shibboleth）登录。您必须选择支持 SAML 2.0 标准的 SAML IdP。

利用内置托管 Web UI，Amazon Cognito 将为所有经过身份验证的用户提供令牌处理和管理。这样，后端系统可以基于一组用户池令牌实现标准化。您可以通过 AWS CLI 或使用 Amazon Cognito API 调用在 AWS Management Console 中创建和管理 SAML IdP。要开始使用控制台，请参阅使用 AWS Management Console将“通过基于 SAML 的身份提供商登录”添加到用户池。



Note

通过第三方（联合身份验证）进行登录可在 Amazon Cognito 用户池中实现。此功能不依赖于通过 Amazon Cognito 身份池（联合身份）实现的联合身份验证。

您需要更新 SAML IdP 并配置用户池以支持此提供商。请参阅 SAML IdP 的文档，了解有关如何将您的用户池添加为 SAML 2.0 IdP 的依赖方或应用程序的信息。

Amazon Cognito 用户群体支持 SAML 2.0 与 POST 绑定端点联合身份验证。这使您的应用程序不必检索或分析 SAML 断言响应，因为用户池直接通过用户代理从 IdP 接收 SAML 响应。您的用户池代表您的应用程序充当服务提供商（SP）。Amazon Cognito 支持 SP 发起的单点登录（SSO），如 SAML V2.0 技术概览的第 5.1.2 部分中所述。

关于 Amazon Cognito 用户群体中的 SAML IdP 需要了解的事项

1. Amazon Cognito 和您的 SAML IdP 使用 `relayState` 参数维护会话信息。
   a. Amazon Cognito 支持大于 80 个字节的 `relayState` 值。虽然 SAML 规范规定 `relayState` 值“长度不得超过 80 个字节”，但目前的行业惯例往往偏离这种行为。因此，拒绝超过 80 个字节的 `relayState` 值将破坏许多标准 SAML 提供商集成。
   b. RelayState 令牌是对 Amazon Cognito 维护的状态信息的不透明引用。Amazon Cognito 不保证 `relayState` 参数的内容。不要解析其内容，以免您的应用程序依赖解析结果。有关更多信息，请参阅 SAML 2.0 规范。
2. 您的 SAML 身份提供者要求您设置断言使用者端点。您的 IdP 使用 SAML 断言将您的用户重定向到此端点。在用户群体域中为您的 SAML 身份提供者中的 SAML 2.0 POST 绑定配置以下端点。您可以在 Amazon Cognito 控制台的应用程序集成选项卡上，找到用户群体的 <yourDomainPrefix> 和 <Region> 的值。

   有关用户群体域的更多信息，请参阅 配置用户池域 (p. 61)。

   ```
   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
   ```

   a. Amazon Cognito `saml2/idpresponse` 端点不接受加密的 SAML 断言。
   b. 您无法向您的 Amazon Cognito `saml2/idpresponse` 端点重复或重放 SAML 断言。重放的 SAML 断言的断言 ID 与早期 IdP 响应的 ID 重复。
3. 您必须向 IdP 提供服务提供商（SP）`urn`，也称为受众 URI 或 SP 实体 ID。用户群体的受众 URI 采用以下格式。

   ```
   urn:amazon:cognito:sp:<your user pool ID>
   ```

   您可以在 Amazon Cognito 控制台的用户群体概览下找到 <yourUserPoolID> 的值。
4. 配置 SAML IdP，为用户群体中根据需要设置的任何属性提供值。例如，`email` 是用户群体的通用必需属性。在您的用户可以登录之前，SAML IdP 断言必须包含映射到用户群体属性 `email` 的声明。有关属性映射的更多信息，请参阅指定适用于用户池的身份提供商属性映射 (p. 107)。
5. SAML IdP 必须在 SAML 响应中包含以下声明。
   - 一项 `AudienceRestriction` 声明，所具有的 `Audience` 值将您的用户群体 SP 实体 ID 设置为响应的目标。

     ```
     <saml:AudienceRestriction>
             <saml:Audience> urn:amazon:cognito:sp:<your user pool ID>
     </saml:AudienceRestriction>
     ```

- 一项 Response 声明，具有原始 SAML 请求 ID 的 InResponseTo 值。

```
<saml2p:Response Destination="https://your user pool domain>/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">
```

- 一项 SubjectConfirmationData 声明，具有用户群体 saml2/idpresponse 端点的 Recipient 值，以及与原始 SAML 请求 ID 匹配的 InResponseTo 值。

```
<saml2:SubjectConfirmationData InResponseTo="_dd0a3436-bc64-4679-a0c2-cb4454f04184"
 NotOnOrAfter="Date-time stamp" Recipient="https://your user pool domain>/saml2/
idpresponse"/>
```

6. 当 对端点授权 (p. 557) 将用户定向到您的 IdP 登录页面时，Amazon Cognito 会在 HTTP GET 请求的 URL 参数中包括 SAML 请求。SAML 请求包含有关用户群体和 saml2/idpresponse 端点的信息。Amazon Cognito 不对 SAML 请求进行签名。
7. Amazon Cognito 将加密签名应用于用户传递到 IdP 的注销端点的注销请求。要与这些注销请求建立信任，您可以向 IdP 提供用户群体 SAML 2.0 签名证书的副本。在 Amazon Cognito 控制台的登录体验选项卡中检索签名证书。

主题

- SAML 用户池 IdP 身份验证流程 (p. 94)
- 选择 SAML 身份提供商名称 (p. 95)
- 为用户池创建和管理 SAML 身份提供商（AWS Management Console）(p. 96)
- 为用户池创建和管理 SAML 身份提供商（AWS CLI 和 AWS API）(p. 98)
- 将第三方 SAML 身份提供商与 Amazon Cognito 用户池集成 (p. 99)
- SAML 会话在 Amazon Cognito 用户池中启动 (p. 100)

SAML 用户池 IdP 身份验证流程

您可以直接从您的用户池集成基于 SAML 的 IdP。

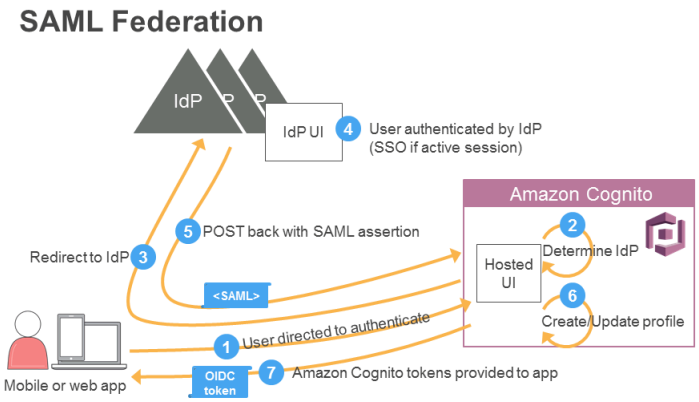
1. 应用程序通过将用户定向到由 AWS 托管的 UI 来开始注册和登录过程。移动应用程序可以使用 Web 视图显示由 AWS 托管的页。
2. 通常，您的用户池将通过用户的电子邮件地址确定用户的 IdP。

   或者，如果您的应用程序在将用户定向到用户池之前已收集信息，它可以通过查询参数向 Amazon Cognito 提供这些信息。
3. 您的用户将使用 SAML 请求重定向到 IdP。
4. IdP 将对用户进行身份验证（如有必要）。如果 IdP 发现用户具有活动会话，则 IdP 将跳过身份验证以提供单一登录 (SSO) 体验。
5. IdP 以 POST 方式将 SAML 断言发送到 Amazon Cognito 服务。

   Note

   Amazon Cognito 会取消未在 5 分钟内完成的身份验证请求，并将用户重定向到托管 UI。页面随即显示 Something went wrong 错误消息。
6. 在验证 SAML 断言并从断言中收集用户属性（声明）后，Amazon Cognito 在用户池中内部创建或更新用户配置文件。Amazon Cognito 将 OIDC 令牌返回到当前已登录用户的应用程序。

下图显示了此过程的身份验证流程：



当用户进行身份验证后，用户池将返回 ID、访问权限和刷新令牌。ID 令牌是用于身份管理的标准 OIDC 令牌，而访问令牌是标准 OAuth 2.0 令牌。ID 和访问令牌将在一小时后过期。您的应用程序可使用刷新令牌来获取新令牌，而无需让用户重新进行身份验证。

作为开发人员，您可以选择刷新令牌的到期时间，从而更改用户需要重新进行身份验证的频率。如果作为联合身份用户的用户已通过外部 IdP 进行身份验证，则应用程序会将 Amazon Cognito 令牌与刷新令牌结合使用来决定用户重新进行身份验证之前经过的时间，而无论外部 IdP 的令牌何时到期。当 ID 和访问令牌过期后，用户池将自动使用刷新令牌来获取新的 ID 和访问令牌。如果刷新令牌也已过期，服务器会通过 AWS 托管的应用程序中的页面自动启动身份验证。

SAML 用户名区分大小写

当联合身份用户尝试登录时，SAML 身份提供商 (IdP) 在用户的 SAML 断言中将 IdP 目录中的唯一 `NameId` 传递给 Amazon Cognito。Amazon Cognito 通过其 `NameId` 声明识别 SAML 联合身份用户。无论您的用户池的区分大小写设置如何，Amazon Cognito 都要求来自 SAML IdP 的联合身份用户传递唯一且区分大小写的 `NameId` 声明。如果您将 `email` 等属性映射到 `NameId`，并且您的用户更改其电子邮件地址，他们将无法登录您的应用程序。

从具有不会改变的值的 IdP 属性映射 SAML 断言中的 `NameId`。

例如，Carlos 在您的不区分大小写的用户池中具有来自 Active Directory 联合身份验证服务 (ADFS) SAML 断言的用户配置文件，该断言传递了`Carlos@example.com` 的 `NameId` 值。下次 Carlos 尝试登录时，您的 ADFS IdP 会传递`carlos@example.com` 的 `NameId` 值。由于 `NameId` 的大小写必须完全匹配，登录不成功。

如果您的用户在其 `NameID` 更改后无法登录，请从您的用户池中删除他们的用户配置文件。Amazon Cognito 将在用户下次登录时创建新的用户配置文件。

选择 SAML 身份提供商名称

选择您的 SAML 身份提供者（IdP）的名称和标识符。SAML 提供者的名称最多可以包含 32 个字符，并且必须符合以下正则表达式：

```
[\p{L}\p{M}\p{S}\p{N}\p{P}\p{Z}]+|[^_\p{Z}][\p{L}\p{M}\p{S}\p{N}\p{P}][^_
\p{Z}]+
```

您还可以为 SAML 提供商选择标识符。标识符将唯一解析为与用户群体关联的 IdP。通常，每个标识符均对应于 SAML IdP 所代表的一个组织域。对于多个组织共享的多租户应用程序，您可以使用标识符将用户重定向到正确的 IdP。由于同一个组织可以拥有多个域，因此您可以提供多个标识符。要使用标识符让用户登录，请使用idp_identifier 参数将他们的会话定向到您的应用程序客户端的 对端点授权 (p. 557)。

您可以为每个 SAML 提供商关联最多 50 个标识符。标识符在用户池中必须唯一。

您可以创建指向您的 Amazon Cognito 用户池 /authorize 终端节点的链接，以静默方式将您的用户重定向到您的 IdP 的登录页面。在对用户群体 /authorize 端点的请求中包含一个标识符作为 idp_identifier 参数。或者，您可以使用 IdP 的名称填充参数 identity_provider。采用以下 URL 格式：

```
https://your_Amazon_Cognito_userpool_domain/authorize?
response_type=code&identity_provider=your-SAML-IdP-name&client_id=your-client-
id&redirect_uri=https://your_application_redirect_url
```

用户使用您的 SAML IdP 登录后，他们会透明地将 SAML 响应提交到您的 saml2/idpresponse 终端节点。Amazon Cognito 通过重定向到您的应用程序客户端回调 URL 来回复 IdP 响应。用户没有看到与您的托管 UI 的任何交互。

以域格式创建 IdP 标识符以收集用户电子邮件地址并将其定向到与其域对应的 IdP 的登录页面。例如，假设您为两家不同公司的员工构建了一个应用程序供其使用。第一家公司 AnyCompany A 拥有 exampleA.com 和 exampleA.co.uk。第二家公司 AnyCompany B 拥有 exampleB.com。对于此示例，您设置了两个 IdP，一个公司对应一个 IdP，如下所示：

- 对于 IdP A，您定义标识符 exampleA.com 和 exampleA.co.uk。
- 对于 IdP B，您定义标识符 exampleB.com。

在您的应用程序中，您可以提示用户输入其电子邮件地址。您的应用程序从他们的电子邮件地址派生用户的域，并通过在对 /authorize 端点的调用的 idp_identifier 参数中提供域来将他们重定向到正确的 IdP。例如，如果用户输入 bob@exampleA.co.uk，则用户将重定向到 IdP A。

Amazon Cognito 托管 UI 可以收集联合身份用户的电子邮件地址并解析域。为此，请为您分配给应用程序客户端的每个 SAML IdP 至少分配一个标识符。默认情况下，托管 UI 登录屏幕为您已分配到应用程序客户端的每个 IdP 显示一个按钮。如果您已成功分配标识符，则托管 UI 登录页面将如下图所示。

要使用此策略，您必须使用域作为 IdP 标识符。Amazon Cognito 从用户输入的电子邮件地址中提取电子邮件域。然后，Amazon Cognito 将电子邮件域作为请求中的 IdPIdentifier 参数添加到 /authorize 终端节点。您的用户在选择上图中显示的 Sign in（登录）按钮后会看到 IdP 登录页面。

- 如果您仅将一个 SAML IdP 分配给您的应用程序客户端，则托管 UI 登录页面会显示一个用于使用该 IdP 登录的按钮。
- 如果您为针对应用程序客户端激活的每个 SAML IdP 分配一个标识符，则托管 UI 登录页面中会出现一个用于输入电子邮件地址的框。
- 如果您有多个 IdP，并且没有为所有 IdP 分配标识符，则托管页面将包含 IdP 列表。
- 如果您已向 IdP 分配了标识符，并且希望托管 UI 显示一系列 IdP 按钮，请向您的应用程序客户端添加一个没有标识符的新 IdP，或者创建新的应用程序客户端。您也可以删除现有 IdP，然后在没有标识符的情况下重新添加它；请注意，这将在您的目录中创建新用户。

如果您有自定义 UI，则解析域名，以便它匹配您在添加 IdP 时分配的标识符。有关 IdP 设置的更多信息，请参阅为用户池配置身份提供商 (p. 262)。

为用户池创建和管理 SAML 身份提供商（AWS Management Console）

您可以使用 AWS Management Console 创建和删除 SAML 身份提供商 (IdP)。

在创建 SAML IdP 之前，您需要使用从第三方 IdP 处获得的 SAML 元数据文档。有关如何获取或生成所需的 SAML 元数据文档的说明，请参阅将第三方 SAML 身份提供商与 Amazon Cognito 用户池集成 (p. 99)。

在您的用户池中配置 SAML 2.0 IdP

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。

3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 Sign-in experience（登录体验）选项卡。找到 Federated sign-in（联合登录），然后选择 Add an identity provider（添加身份提供商）。
5. 选择一个 SAML IdP。
6. 输入以逗号分隔的 Identifiers（标识符）。标识符将告知 Amazon Cognito 应该检查用户登录时输入的电子邮件地址，然后将它们引导到与其域名对应的提供商。
7. 如果您希望 Amazon Cognito 在用户注销时向您的提供商发送已签名的注销请求，请选择 Add sign-out flow（添加注销流程）。您必须配置 SAML 2.0 IdP 以向配置托管 UI 时创建的 `https://<your Amazon Cognito domain>/saml2/logout` 终端节点发送注销响应。此 `saml2/logout` 端点使用 POST 绑定。

   Note

   如果选择此选项，并且您的 SAML IdP 需要已签名的注销请求，则您还需要为您的 SAML IdP 配置 Amazon Cognito 提供的签名证书。
   SAML IdP 将处理已签名的注销请求并从 Amazon Cognito 会话中注销您的用户。
8. 选择 Metadata document source（元数据文档源）。如果您的 IdP 在公有 URL 上提供 SAML 元数据，则可以选择 Metadata document URL（元数据文档 URL），然后输入该公有 URL。否则，请选择 Upload metadata document（上载元数据文档），然后选择您之前从提供商下载的元数据文件。

   Note

   我们建议您输入元数据文档 URL（如果您的提供商具有公有端点），而不是上载文件，因为这将允许 Amazon Cognito 自动刷新元数据。通常，元数据刷新操作每 6 小时执行一次或在元数据过期前执行（以时间较早者为准）。
9. Map attributes between your SAML provider and your app（在 SAML 提供商和应用程序之间映射属性）将 SAML 提供程序属性映射到用户池中的用户配置文件。在属性映射中包含用户池必需属性。

   例如，当您选择 User pool attribute（用户池属性）`email` 时，按照您的 IdP 提供的 SAML 断言中显示的内容，输入 SAML 属性名称。如果 IdP 提供了示例 SAML 断言，您可以使用这些示例断言帮助您查找名称。一些 IdP 使用简单名称 (如 `email`)，另一些则使用类似于下面的名称：

   ```
   http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
   ```
10. 选择 Create（创建）。

    Note

    如果您在使用 HTTPS 元数据终端节点 URL 创建 SAML IdP 时看见 `InvalidParameterException`（例如，“Error retrieving metadata from *<metadata endpoint>* (从 <元数据终端节点> 检索元数据时出错)”），请确保元数据终端节点已正确设置 SSL，并且存在与之关联的有效 SSL 证书。

设置 SAML IdP 以添加用户池作为信赖方

- 用户池服务提供商 URN 为：`urn:amazon:cognito:sp:<user_pool_id>`。Amazon Cognito 向 SAML IdP 发出 `AuthnRequest` 以发布 SAML 断言以及对此 URN 的受众限制。您的 IdP 将以下 POST 绑定终端节点用于 IdP 到 SP 的响应消息：`https://Your user pool domain/saml2/idpresponse`。
- 确保您的 SAML IdP 在 SAML 断言中为用户池填充 `NameID` 和任何所需属性。`NameID` 用于在用户池中唯一地标识您的 SAML 联合身份用户。使用持久性 SAML 名称 ID 格式。

向您的 SAML 2.0 IDP 提供签名证书

- 要从 Amazon Cognito 获取您的 IdP 可用于验证 SAML 注销请求的公有密钥的副本，请在您的用户群体的登录体验选项卡中的签名证书下选择查看签名证书或以 .crt 格式下载证书。

您可以使用 Amazon Cognito 控制台删除在用户池中设置的任何 SAML 提供商。

删除 SAML 提供商

1. 登录 Amazon Cognito 控制台。
2. 在导航窗格中，选择 User Pools（用户池），然后选择要编辑的用户池。
3. 选择 Sign-in experience（登录体验）选项卡并找到 Federated sign-in（联合登录）。
4. 选中要删除的 SAML IdP 旁边的单选按钮。
5. 当系统提示您 Delete identity provider（删除身份提供商）时，请输入 SAML 提供商的名称以确认删除，然后选择 Delete（删除）。

为用户池创建和管理 SAML 身份提供商（AWS CLI 和 AWS API）

使用以下命令可创建和管理 SAML 身份提供商 (IdP)。

创建 IdP 并上传元数据文档

- AWS CLI: `aws cognito-idp create-identity-provider`

  带元数据文件的示例：`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>* 包含任何引号 (")，则必须对其进行转义 (\")。

  带元数据 URL 的示例：`aws cognito-idp create-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-type SAML --provider-details MetadataURL=<metadata_url> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`
- AWS API：CreateIdentityProvider

为 IdP 上传新的元数据文档

- AWS CLI: `aws cognito-idp update-identity-provider`

  带元数据文件的示例：`aws cognito-idp update-identity-provider --user-pool-id <user_pool_id> --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：

```
{
    "MetadataFile": "<SAML metadata XML>"
}
```

Note

如果 *<SAML metadata XML>* 包含任何引号 (")，则必须对其进行转义 (\")。

带元数据 URL 的示例：`aws cognito-idp update-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-details MetadataURL=<metadata_url> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API：UpdateIdentityProvider

获取有关特定 IdP 的信息

- AWS CLI: `aws cognito-idp describe-identity-provider`

  `aws cognito-idp describe-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1`
- AWS API：DescribeIdentityProvider

列出有关所有 IdP 的信息

- AWS CLI: `aws cognito-idp list-identity-providers`

  示例：`aws cognito-idp list-identity-providers --user-pool-id <user_pool_id> --max-results 3`
- AWS API：ListIdentityProviders

删除 IdP

- AWS CLI: `aws cognito-idp delete-identity-provider`

  `aws cognito-idp delete-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1`
- AWS API：DeleteIdentityProvider

将第三方 SAML 身份提供商与 Amazon Cognito 用户池集成

要配置第三方 SAML 2.0 身份提供者（IdP）解决方案以使用 Amazon Cognito 用户群体的联合身份验证，您必须配置 SAML IdP 以重定向到以下 URL：`https://Your user pool domain/saml2/idpresponse`。如果您的用户群体具有 Amazon Cognito 域，则可以在 Amazon Cognito 控制台中您的用户群体的 App integration（应用程序集成）选项卡中找到用户群体域路径。

对于一些 SAML IdP，您必须以 `urn:amazon:cognito:sp:<yourUserPoolID>` 格式提供 urn/受众 URI/SP 实体 ID。您可以在 Amazon Cognito 控制台的 General settings（常规设置）选项卡上找到用户池 ID。

您还必须配置 SAML IdP，为您用户池中需要的任意属性提供属性值。通常，`email` 是用户池的必需属性，这种情况下 SAML IdP 必须在 SAML 断言中提供 `email` 值（声明）。

Note

Amazon Cognito 不接受您的 IdP 作为属性值传递的 4 字节 UTF-8 字符（例如 # 或#）。您可以将字符编码为 Base64，将其作为文本传递，然后在应用程序中对其进行解码。
在以下示例中，将不接受属性声明：

```
<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>
```

与上述示例不同，将接受以下属性声明：

```
<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>
```

以下链接将帮助您配置第三方 SAML 2.0 IdP 解决方案以使用 Amazon Cognito 用户池的联合身份验证。

Note

Amazon Cognito 包含 IdP 支持，因此您只需访问以下提供商站点即可获取 SAML 元数据文档。您可以在提供商网站上查看有关与 AWS 集成的详细说明，但您不需要这些说明。

解决方案	更多信息
Microsoft Active Directory 联合身份验证服务 (AD FS)	您可以下载 ADFS 联合身份验证服务器的 SAML 元数据文档。以下是一个示例元数据 URL。 `https://<yourservername>/FederationMetadata/2007-06/FederationMetadata.xml`
Okta	在 Okta 中将 Amazon Cognito 用户群体配置为应用程序后，您可以在 Okta 控制面板的应用程序部分中找到元数据文档。选择您的应用程序，选择登录选项卡，然后找到元数据详细信息。以下是一个示例元数据 URL。 `https://<prefix>.okta.com/app/<App ID>/sso/saml/metadata`
Auth0	您可以从 Auth0 控制面板获取元数据文档。选择客户端，然后选择设置。选择显示高级设置并找到 SAML 元数据 URL。以下是一个示例元数据 URL。 `https://<your-domain-prefix>.auth0.com/samlp/metadata/<your-Auth0-client-ID>`
Ping Identity	下载 PingFederate 元数据 XML 文件的说明位于 PingIdentity 知识库的从 PingFederate 导出 SAML 元数据中。

SAML 会话在 Amazon Cognito 用户池中启动

Amazon Cognito 支持由服务提供商启动（SP 启动）的单点登录 (SSO)。SAML V2.0 技术概述的第 5.1.2 节描述了 SP 启动的 SSO。Amazon Cognito 是您的应用程序的身份提供商 (IdP)。该应用程序是为经过身份验证的用户检索令牌的服务提供程序 (SP)。但是，当您使用第三方 IdP 对用户进行身份验证时，Amazon Cognito 就是 SP。使用 SAML IdP 进行身份验证的 Amazon Cognito 用户必须始终首先向 Amazon Cognito 发出请求并重定向到 IdP 进行身份验证。

对于某些企业使用案例，对内部应用程序的访问从企业 IdP 托管的控制面板上的书签开始。当用户选择书签时，IdP 会生成一个 SAML 响应并将其发送到 SP 以向应用程序验证用户身份。

Amazon Cognito 不支持 IdP 启动的 SSO。除非 Amazon Cognito 通过 SAML 请求启动身份验证，否则 Amazon Cognito 无法验证它是否已请求收到的 SAML 响应。

在企业控制面板中为 Amazon Cognito 应用程序添加书签

您可以在 SAML 或 OIDC IdP 控制面板中创建书签，提供对通过 Amazon Cognito 用户池进行身份验证的 Web 应用程序的 SSO 访问。您可以通过不要求用户使用托管 UI 登录的方式链接到 Amazon Cognito。为此，请根据以下模板构建一个指向 Amazon Cognito 用户池的 对端点授权 (p. 557) 的书签链接：

```
https://your_Amazon_Cognito_userpool_domain/authorize?
response_type=code&identity_provider=your-SAML-IdP-name&client_id=your-client-
id&redirect_uri=https://your_application_redirect_url
```

> Note
>
> 您还可以在授权终端节点请求中使用 `idp_identifier` 参数而不是 `identity_provider` 参数。IdP 标识符是在用户池中创建身份提供商时可以配置的替代名称。请参阅选择 SAML 身份提供商名称 (p. 95)。

当您在对 `/authorize` 的请求中使用适当的参数时，Amazon Cognito 会静默开始 SP 启动的登录流程并将您的用户重定向到使用您的 IdP 登录。

下图显示了模拟 IdP 启动的 SSO 的身份验证流程。您的用户可以通过公司门户网站中的链接向 Amazon Cognito 进行身份验证。

上传或链接来自提供商的元数据文档，以便在用户池中添加 SAML IdP。创建一个使用您的 SAML IdP 进行登录的应用程序客户端，并将您的应用程序的 URL 作为授权回调 URL。有关应用程序客户端的更多信息，请参阅配置用户池应用程序客户端 (p. 258)。

在尝试使用此解决方案向应用程序进行身份验证之前，请测试由 SP 发起的从托管 UI 向您的应用程序的登录。有关如何在 Amazon Cognito 中配置 SAML IdP 的更多信息，请参阅将第三方 SAML 身份提供商与 Amazon Cognito 用户池集成 (p. 99)。

满足这些要求后，为您的 `/authorize` 终端节点创建一个书签，其中包含 `identity_provider` 或 `idp_identifier` 参数。上图说明了随后的过程。

1. 您的用户登录 SSO IdP 控制面板。用户有权访问的企业应用程序将填充此控制面板。
2. 您的用户选择指向应用程序的链接，该应用程序向 Amazon Cognito 进行身份验证。在许多 SSO 门户中，您都可以添加自定义应用程序链接。可用于创建指向 SSO 门户中公共 URL 的链接的任何功能都可以使用。
3. SSO 门户中的自定义应用程序链接将用户定向到用户池 对端点授权 (p. 557)。该链接包括 `response_type`、`client_id`、`redirect_uri` 和 `identity_provider` 的参数。`identity_provider` 参数是您在用户池中提供 IdP 的名称。您还可以使用 `idp_identifier` 参数而不是 `identity_provider` 参数。用户通过包含 `idp_identifier` 或 `identity_provider` 参数的链接访问您的托管 UI。此用户绕过登录页面并直接导航以使用您的 IdP 进行身份验证。有关命名 SAML IdP 的更多信息，请参阅选择 SAML 身份提供商名称 (p. 95)。

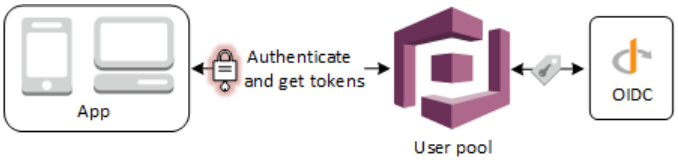
   示例 URL：

   ```
   https://your_Amazon_Cognito_userpool_domain/authorize?
   response_type=code&identity_provider=your-SAML-IdP-name&client_id=your-
   client-id&redirect_uri=https://your_application_redirect_url
   ```
4. Amazon Cognito 使用 SAML 请求将用户会话重定向到您的 IdP。
5. 您的用户在登录控制面板时可能已收到 IdP 的会话 Cookie。您的 IdP 使用此 Cookie 以静默方式验证用户并使用 SAML 响应将其重定向到 Amazon Cognito `idpresponse` 终端节点。如果不存在活动的会话，则 IdP 在发布 SAML 响应之前将对用户重新进行身份验证。
6. Amazon Cognito 会验证 SAML 响应，并根据 SAML 断言创建或更新用户配置文件。

7. Amazon Cognito 使用授权码将用户重定向到您的内部应用程序。您将内部应用程序 URL 配置为应用程序客户端的授权重定向 URL。
8. 您的应用程序用授权码交换 Amazon Cognito 令牌。有关更多信息，请参阅令牌端点 (p. 562)。

向用户池添加 OIDC 身份提供商

您可允许已具有 OpenID Connect (OIDC) 身份提供者 (IdP) 账户的用户跳过注册步骤，并使用现有账户登录您的应用程序。利用内置托管 Web UI，Amazon Cognito 将为所有经过身份验证的用户提供令牌处理和管理。这样，后端系统可以基于一组用户池令牌实现标准化。



Note

通过第三方（联合身份验证）进行登录可在 Amazon Cognito 用户池中实现。此功能不依赖于通过 Amazon Cognito 身份池（联合身份）实现的联合身份验证。

您可以在AWS Management Console中、使用 AWS CLI 或使用用户池 API 方法 CreateIdentityProvider 向用户池添加 OIDC IdP。

主题

- 先决条件 (p. 102)
- 步骤 1：向 OIDC IdP 注册 (p. 102)
- 步骤 2：将 OIDC IdP 添加到用户池 (p. 104)
- 步骤 3：测试 OIDC IdP 配置 (p. 106)
- OIDC 用户池 IdP 身份验证流程 (p. 106)

先决条件

在开始之前，您需要：

- 具有应用程序客户端和用户池域的用户池。有关更多信息，请参阅创建用户池。
- 具有以下配置的 OIDC IdP：
  - 支持 `client_secret_post` 客户端身份验证。Amazon Cognito 不在 IdP 的 OIDC 发现端点上检查 `token_endpoint_auth_methods_supported` 声明。Amazon Cognito 不支持 `client_secret_basic` 客户端身份验证。有关客户端验证的更多信息，请参阅 OpenID Connect 文档中的客户端身份验证。
  - 仅对 OIDC 端点使用 HTTPS，例如 `openid_configuration`、`userInfo` 和 `jwks_uri`。
  - 仅为 OIDC 端点使用 TCP 端口 80 和 443。
  - 只能使用 HMAC-SHA、ECDSA 或 RSA 算法对 ID 令牌进行签名。

步骤 1：向 OIDC IdP 注册

在使用 Amazon Cognito 创建 OIDC IdP 之前，必须向 OIDC IdP 注册应用程序才能接收客户端 ID 和客户端密钥。

向 OIDC IdP 注册

1. 使用 OIDC IdP 创建开发人员账户。

链接到 OIDC IdP

OIDC IdP	如何安装	OIDC 发现 URL
Salesforce	安装 Salesforce 身份提供商	https://login.salesforce.com
Ping Identity	安装 Ping Identity 身份提供商	https://*## Ping ###*:9031/idp/userinfo.openid 例如：`https://pf.company.com:9031/idp/userinfo.openid`
Okta	安装 Okta 身份提供商	https://*## Okta ##*.oktapreview.com 或者 `https://`*`Your Okta subdomain`*`.okta.com`
Microsoft Azure Active Directory (Azure AD)	安装 Microsoft Azure AD 身份提供商	`https://login.microsoftonline.com/`*`{tenant}`*`/v2.0`
Google	安装 Google 身份提供商	https://accounts.google.com **Note** Amazon Cognito 提供 Google 作为集成社交登录 IdP。建议使用集成 IdP。请参阅向用户池添加社交身份提供商 (p. 86)。

2. 向 OIDC IdP 注册具有 `/oauth2/idpresponse` 端点的用户池域 URL。这将确保 OIDC IdP 之后在对用户进行身份验证时通过 Amazon Cognito 接受此 URL。

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

3. 向 Amazon Cognito 用户池注册回调 URL。这是成功身份验证后 Amazon Cognito 将您的用户重定向到的页面的 URL。

```
https://www.example.com
```

4. 选择 scopes（范围）。范围 openid 为必填字段。需要 email（电子邮件）范围来授予对 email 和 email_verified 声明的访问权限。
5. OIDC IdP 为您提供客户端 ID 和客户端密钥。您在用户池中设置 OIDC IdP 时将使用它们。

示例：使用 Salesforce 作为用户池的 OIDC IdP

当您要在与 OIDC 兼容的 IdP（如 Salesforce）和您的用户池之间建立信任时，请使用 OIDC IdP。

1. 在 Salesforce 开发人员网站上创建账户。
2. 通过在上一步中设置的开发人员账户登录。
3. 请在 Salesforce 页面上，执行以下操作之一：
   - 如果您使用的是 Lightning Experience，请选择设置齿轮图标，然后选择 Setup Home（设置主页）。
   - 如果您使用的是 Salesforce Classic 并且在用户界面标题中看到 Setup（设置），请选择它。

- 如果您使用的是 Salesforce Classic 但没有在用户界面标题中看到 Setup（设置），请从顶部导航栏中选择您的姓名，然后从下拉列表中选择 Setup（设置）。

4. 在左侧导航栏上，选择 Company Settings（公司设置）。
5. 在导航栏上，选择 Domain（域），输入一个域，然后选择 Create（创建）。
6. 在左侧导航栏上，选择在 Platform Tools（平台工具）下的 Apps（应用程序）。
7. 选择 App Manager（应用程序管理器）。
8. a. 选择 New connected app（新连接的应用程序）。
   b. 完成必填句段。

      在 Start URL（启动 URL）下，在 `/authorize` 终端节点处输入使用您的 Salesforce IdP 登录的用户池域的 URL。当您的用户访问您连接的应用程序时，Salesforce 会将他们定向到此 URL 以完成登录。然后 Salesforce 将用户重定向到与应用程序客户端关联的回调 URL。

      ```
      https://<your_user_pool_domain>/authorize?
      response_type=code&client_id=<your_client_id>&redirect_uri=https://
      www.example.com&identity_provider=CorpSalesforce
      ```

   c. 启用 OAuth settings（OAuth 设置），然后在 Callback URL（回调 URL）中输入用户池域的 `/oauth2/idpresponse` 终端节点的 URL。这是 Salesforce 发出授权码的 URL，Amazon Cognito 会用该代码交换 OAuth 令牌。

      ```
      https://<your_user_pool_domain>/oauth2/idpresponse
      ```

9. 选择 scopes（范围）。您必须包含范围 openid。要授予对 email 和 email_verified 声明的访问权限，请添加 email（电子邮件）范围。通过空格分隔范围。
10. 选择 Create（创建）。

    在 Salesforce 中，客户端 ID 称为 Consumer Key（使用者密钥），客户端密钥为 Consumer Secret（使用者私有密钥）。记下您的客户端 ID 和客户端密钥。您将在下一节中使用它们。

步骤 2：将 OIDC IdP 添加到用户池

在本节中，配置用户池以通过 OIDC IdP 处理基于 OIDC 的身份验证请求。

添加 OIDC IdP（Amazon Cognito 控制台）

添加 OIDC IdP

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 从导航菜单中选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 Sign-in experience（登录体验）选项卡。找到 Federated sign-in（联合登录），然后选择 Add an identity provider（添加身份提供商）。
5. 选择一个 OpenID Connect IdP。
6. 在 Provider name（提供商名称）中输入一个唯一名称。
7. 将您从提供商那里收到的客户端 ID 输入到 Client ID（客户端 ID）。
8. 将您从提供商那里收到的客户端密钥输入到 Client secret（客户端密钥）。
9. 为该提供商输入 Authorized scopes（授权范围）。范围定义了应用程序将向您的提供商请求的用户属性组（例如 `name` 和 `email`）。根据 OAuth 2.0 规范，范围必须以空格分隔。

   您的用户需要同意向您的应用程序提供这些属性。
10. 请选择一个 Attribute request method（属性请求方法），以便为 Amazon Cognito 提供 HTTP 方法（GET 或 POST），它必须使用该方法从提供商运营的 userInfo 端点中获取用户的详细信息。

11. 请选择 Setup method（设置方法）并通过 Auto fill through issuer URL（自动填充发布者 URL）或 Manual input（手动输入）检索 OpenID Connect 端点。当您的提供商具有公有 `.well-known/openid-configuration` 端点且 Amazon Cognito 可以在其中检索 `authorization`、`token`、`userInfo` 和 `jwks_uri` 端点的 URL 时，使用 Auto fill through issuer URL（自动填充发布者 URL）。
12. 请输入发布者 URL 或 IdP 中的 `authorization`、`token`、`userInfo` 和 `jwks_uri` 端点 URL。

    Note

    URL 应该以 `https://` 开头，并且不应以下斜杠 `/` 结尾。只有端口号 443 和 80 可用于此 URL。例如，Salesforce 使用以下 URL：`https://login.salesforce.com`
    如果选择自动填充，则发现文档必须对以下值使用 HTTPS：`authorization_endpoint`、`token_endpoint`、`userinfo_endpoint` 和 `jwks_uri`。否则，登录将失败。
13. 默认情况下，sub OIDC 声明将映射到用户池 Username（用户名）属性中。您可以将其他 OIDC 声明映射到用户池属性。输入 OIDC 声明，然后从下拉列表中选择对应的用户池属性。例如，声明 email 通常会映射到用户池属性 Email（电子邮件）。
14. 请将 IdP 的属性映射到您的用户池。有关更多信息，请参阅指定适用于用户池的身份提供程序属性映射。
15. 选择 Create（创建）。
16. 从 App client integration（应用程序客户端集成）选项卡上的列表中选择一个 App clients（应用程序客户端），然后选择 Edit hosted UI settings（编辑托管 UI 设置）。将新的 OIDC IdP 添加到 Identity providers（身份提供商）下的应用程序客户端。
17. 选择 Save changes（保存更改）。

添加 OIDC IdP (AWS CLI)

- 请参阅 CreateIdentityProvider API 方法的参数描述。

```
aws cognito-idp create-identity-provider
--user-pool-id string
--provider-name string
--provider-type OIDC
--provider-details map


--attribute-mapping string
--idp-identifiers (list)
--cli-input-json string
--generate-cli-skeleton string
```

使用此提供商详细信息映射：

```
{
  "client_id": "string",
  "client_secret": "string",
  "authorize_scopes": "string",
  "attributes_request_method": "string",
  "oidc_issuer": "string",

  "authorize_url": "string",
```

```
  "token_url": "string",
  "attributes_url": "string",
  "jwks_uri": "string"
}
```

步骤 3：测试 OIDC IdP 配置

可以通过使用上两节中的元素并使用这些元素测试 OIDC IdP 配置来创建授权 URL。

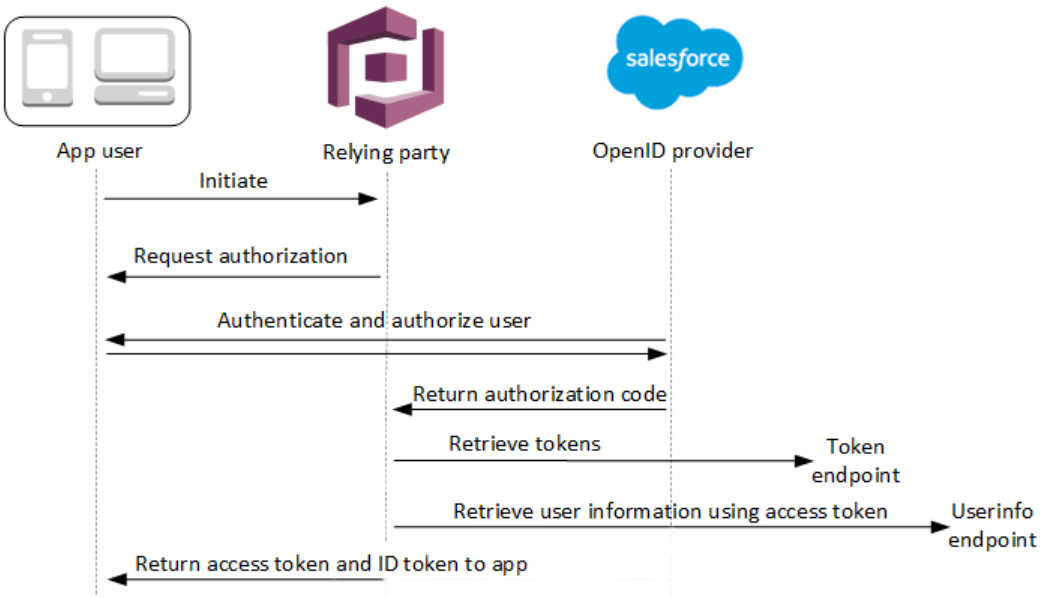
```
https://<your_user_pool_domain>/oauth2/authorize?
response_type=code&client_id=<your_client_id>&redirect_uri=https://www.example.com
```

您可以在用户池 Domain name（域名）控制台页上找到您的域。您可以在 General settings（常规设置）页面上找到 client_id。对于 redirect_uri 参数，使用您的回调 URL。这是页面的 URL，在页面中，您的用户在身份验证成功后将被重定向。

OIDC 用户池 IdP 身份验证流程

当您的用户使用 OIDC IdP 登录您的应用程序时，他们会经过以下身份验证流程。

1. 您的用户将登录 Amazon Cognito 内置登录页面，并获得通过 OIDC IdP（如 Salesforce）登录的选项。
2. 您的用户将重定向到 OIDC IdP 的 `authorization` 终端节点。
3. 在您的用户经过身份验证后，OIDC IdP 将使用授权代码重新导向至 Amazon Cognito。
4. Amazon Cognito 与 OIDC IdP 交换此授权代码以获得访问令牌。
5. Amazon Cognito 在您的用户池中创建或更新用户账户。
6. Amazon Cognito 颁发应用程序持有者令牌，可能包括身份令牌、访问令牌和刷新令牌。



Note

Amazon Cognito 会取消未在 5 分钟内完成的身份验证请求，并将用户重定向到托管 UI。页面随即显示 `Something went wrong` 错误消息。

OIDC 是基于 OAuth 2.0 的身份层，它指定 IdP 向 OIDC 客户端应用程序（信赖方）颁发的 JSON 格式的 (JWT) 身份令牌。有关将 Amazon Cognito 添加为 OIDC 信赖方的信息，请参阅适用于您 OIDC IdP 的文档。

当用户进行身份验证后，用户池将返回 ID、访问权限和刷新令牌。ID 令牌是用于身份管理的标准 OIDC 令牌，而访问令牌是标准 OAuth 2.0 令牌。

指定适用于用户池的身份提供商属性映射

您可以使用 AWS Management Console、AWS CLI 或 API 来指定用户池的身份提供商 (IdP) 的属性映射。

有关映射的需知信息

在使用映射之前，请查看下面重要的详细信息：

- 当联合用户登录到您的应用程序时，您的用户群体需要的每个用户群体属性都必须存在一个映射。例如，如果您的用户群体需要 `email` 属性来进行注册，则将此属性映射到 IdP 中的对等属性。
- 默认情况下，映射的电子邮件地址未经验证。您无法使用一次性代码验证映射的电子邮件地址。而是要映射 IdP 的属性来获取验证状态。例如，Google 和大多数 OIDC 提供商都包含 `email_verified` 属性。
- 对于每个映射的用户池属性，最大值长度（2048 个字符）必须足够大，才能容纳 Amazon Cognito 从 IdP 处获取的值。否则，当用户登录到您的应用程序时，Amazon Cognito 会报告错误。如果您将自定义属性映射到 IdP 令牌，请将长度设置为 2048 个字符。
- Amazon Cognito 从联合 IdP 传递的特定声明中派生出联合用户配置文件中的 `username` 属性，如下所示。Amazon Cognito 在此属性值前面加上 IdP 的名称。当您希望联合用户拥有与第三方目录中的属性完全匹配的属性时，请将该属性映射到另一个 Amazon Cognito 属性，如 `preferred_username`。

身份提供商	**`username`** 资源属性
Facebook	`id`
Google	`sub`
Login with Amazon	`user_id`
Sign in with Apple	`sub`
SAML 提供商	`NameID`
OpenID Connect (OIDC) 提供商	`sub`

- 当用户登录您的应用程序时，Amazon Cognito 必须能够更新映射的用户池属性。用户通过某个 IdP 登录时，Amazon Cognito 将使用来自该 IdP 的最新信息更新映射的属性。Amazon Cognito 将更新每个映射的属性，即使当前值已经与最新信息匹配也会更新。如果 Amazon Cognito 无法更新属性，则会报告错误。要确保 Amazon Cognito 可以更新属性，请检查以下要求：
  - 您从 IdP 映射的所有用户群体自定义属性都必须为可变的。您可以随时更新可变的自定义属性。相比之下，只有在首次创建用户配置文件时，才能为用户的不可变 自定义属性设置值。要在 Amazon Cognito 控制台中创建可变的自定义属性，请在 Sign-up experience（注册体验）选项卡中选择 Add custom attributes（添加自定义属性）时，为您添加的属性激活 Mutable（可变）复选框。或者，如果您使用 CreateUserPool API 操作创建用户池，则可将这些属性中的每个属性的 `Mutable` 参数设置为 `true`。
  - 在应用程序的应用程序客户端设置中，映射的属性必须可写。您可以在 Amazon Cognito 控制台的 App clients (应用程序客户端) 页面中设置哪些属性为可写属性。或者，如果您使用 `CreateUserPoolClient` API 操作创建应用程序客户端，则可以将这些属性添加到 `WriteAttributes` 数组。
- 如果 IdP 属性包含多个值，Amazon Cognito URL 形式对包含非字母数字字符的值（不包括“.”、“-”、“*”和“_”字符）进行编码。在您的应用程序中使用这些值之前，您必须解码这些值。

指定适用于用户池的身份提供商属性映射（AWS Management Console）

您可以使用 AWS Management Console为您的用户池的 IdP 指定属性映射。

> Note
>
> 只有当陈述存在于传入令牌中时，Amazon Cognito 才会将传入陈述映射到用户池属性。如果之前映射的声明不再存在于传入令牌中，则不会被删除或更改。如果您的应用程序需要映射已删除的声明，则可以使用预身份验证 Lambda 触发器在身份验证期间删除自定义属性，并允许从传入令牌重新填充这些属性。

指定社交 IdP 属性映射

1. 登录 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 在导航窗格中，选择 User Pools（用户池），然后选择要编辑的用户池。
3. 选择 Sign-in experience（登录体验）选项卡并找到 Federated sign-in（联合登录）。
4. 选择 Add an identity provider（添加身份提供商），或者选择您已配置的 Facebook、Google、Amazon 或 Apple IdP。找到 Attribute mapping（属性映射），然后选择 Edit（编辑）。

   有关添加社交 IdP 的更多信息，请参阅向用户池添加社交身份提供商 (p. 86)。
5. 对于需要映射的每个属性，请完成以下步骤：

   a. 从 User pool attribute（用户池属性）列中选择属性。这是分配给您的用户池中用户配置文件的属性。自定义属性在标准属性之后列出。

   b. 从 ***<provider>*** attribute（<provider> 属性）列中选择属性。这将是从提供商目录传递的属性。在下拉列表中提供来自社交服务提供商的已知属性。

   c. 要在 IdP 和 Amazon Cognito 之间映射其它属性，请选择 Add another attribute（添加其它属性）。
6. 选择 Save changes（保存更改）。

指定 SAML 提供商属性映射

1. 登录 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 在导航窗格中，选择 User Pools（用户池），然后选择要编辑的用户池。
3. 选择 Sign-in experience（登录体验）选项卡并找到 Federated sign-in（联合登录）。
4. 选择 Add an identity provider（添加身份提供商），或者选择您已配置的 SAML IdP。找到 Attribute mapping（属性映射），然后选择 Edit（编辑）。有关添加 SAML IdP 的更多信息，请参阅向用户池添加 SAML 身份提供商 (p. 92)。
5. 对于需要映射的每个属性，请完成以下步骤：

   a. 从 User pool attribute（用户池属性）列中选择属性。这是分配给您的用户池中用户配置文件的属性。自定义属性在标准属性之后列出。

   b. 从 SAML attribute（SAML 属性）列中选择属性。这将是从提供商目录传递的属性。

      您的 IdP 可能会提供示例 SAML 断言以供参考。一些 IdP 使用简单名称（如 `email`），另一些则使用类似于下面 URL 格式的属性名称：

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

   c. 要在 IdP 和 Amazon Cognito 之间映射其它属性，请选择 Add another attribute（添加其它属性）。
6. 选择 Save changes（保存更改）。

指定适用于用户池的身份提供商属性映射（AWS CLI 和 AWS API）

使用以下命令为您的用户池指定 IdP 属性映射。

在提供商创建时指定属性映射

- AWS CLI: `aws cognito-idp create-identity-provider`

  带元数据文件的示例：`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>* 包含任何引号 (")，则必须对其进行转义 (\")。

  带元数据 URL 的示例：`aws cognito-idp create-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-type SAML --provider-details MetadataURL=<metadata_url> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`
- AWS API：CreateIdentityProvider

指定现有 IdP 的属性映射

- AWS CLI: `aws cognito-idp update-identity-provider`

  示例：`aws cognito-idp update-identity-provider --user-pool-id <user_pool_id> --provider-name <provider_name> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`
- AWS API：UpdateIdentityProvider

获取有关特定 IdP 的属性映射的信息

- AWS CLI: `aws cognito-idp describe-identity-provider`

  示例：`aws cognito-idp describe-identity-provider --user-pool-id <user_pool_id> --provider-name <provider_name>`
- AWS API：DescribeIdentityProvider

将联合用户与现有用户配置文件关联

通常，同一个用户拥有的配置文件具有您已连接到用户群体的多个身份提供者 (IdP)。Amazon Cognito 可以将用户的每次出现与目录中的同一个用户配置文件关联。这样，一个拥有多个 IdP 用户的人可以在您的应用程序中获得一致的体验。AdminLinkProviderForUser 告诉 Amazon Cognito 将联合目录中的用户的唯一 ID 识别为用户群体中的用户。当您具有零个或多个与用户配置文件相关联的联合身份时，用户群体中的一个用户将计为一个每月活跃用户 (MAU) 以进行计费。

当联合用户登录用户群体时，Amazon Cognito 会创建一个以您的 IdP 名称为前缀的即时用户配置文件。在用户群体中搜索用户时，您可以找到链接的本地用户和自动创建的联合用户。在其令牌中，关联用户的身份声明源于他们的联合身份验证配置文件。他们的访问权限声明源于他们的本地配置文件。

> **Important**
>
> 由于 `AdminLinkProviderForUser` 允许具有外部联合身份的用户以用户群体中的现有用户身份进行登录，因此，至关重要的是，它只能与应用程序拥有者已信任的外部 IdP 和提供者属性一起使用。

例如，如果您是托管服务提供商 (MSP)，具有一个与多个客户共享的应用程序。每个客户都通过 Active Directory Federation Services (ADFS) 登录您的应用程序。您的 IT 管理员 Carlos 在每个客户的域中都有一个账户。您希望 Carlos 在每次登录时都能被识别为应用程序管理员，而不考虑 IdP。

您的 ADFS IdP 在 Carlos 的 SAML 断言的 `email` 声明中向 Amazon Cognito 提供 Carlos 的电子邮件地址 `msp_carlos@example.com`。您使用用户名 `Carlos` 在用户群体中创建一个用户。以下 AWS Command Line Interface (AWS CLI) 命令将 Carlos 的来自 IdP ADFS1、ADFS2 和 ADFS3 的身份相关联。

> **Note**
>
> 您可以根据特定属性声明关联用户。此功能对 SAML IdP 来说是独有的。对于其他提供者类型，您必须基于固定来源属性来进行关联。有关更多信息，请参阅 AdminLinkProviderForUser。
> 当您将 OIDC 或社交 IdP 与用户配置文件关联时，必须将 `ProviderAttributeName` 设置为 `Cognito_Subject`。`ProviderAttributeValue` 必须是用户在 IdP 中的唯一标识符。

```
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
```

用户群体中的用户配置文件 `Carlos` 现在具有以下 `identities` 属性。

```
[{
    "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
}]
```

有关关联联合用户需了解的事项

- 您最多可以将五个联合用户与每个用户配置文件关联。
- 您可以将联合用户与现有联合用户配置文件或本地用户关联。
- 您无法在 AWS Management Console中将提供者与用户配置文件关联。
- 您的用户 ID 令牌在 `identities` 声明中含包所有关联的提供者。
- 您可以在 AdminSetUserPassword API 请求中为自动创建的联合用户配置文件设置密码。然后，该用户的状态从 `EXTERNAL_PROVIDER` 更改为 `CONFIRMED`。处于此状态的用户能够以联合用户身份登录，并像关联的本地用户一样在 API 中启动身份验证流程。他们还可以在经过令牌身份验证的 API 请求中修改其密码和属性，如 ChangePassword 和 UpdateUserAttributes。作为最佳安全实践，为了让用户与您的外部 IdP 保持同步，请勿在联合用户配置文件上设置密码。相反，使用 `AdminLinkProviderForUser` 将用户与本地配置文件关联。
- 当用户通过 IdP 进行登录时，Amazon Cognito 会将用户属性填充到关联的本地用户配置文件中。Amazon Cognito 处理来自 OIDC IdP 的 ID 令牌中的身份声明，还会检查 OAuth 2.0 和 OIDC 提供者的 `userInfo` 端点。Amazon Cognito 将 ID 令牌中的信息优先级置于来自 `userInfo` 的信息之上。

有关 AWS SDK 中的其他命令语法和示例，请参阅 AdminLinkProviderForUser。

使用 Lambda 触发器自定义用户池工作流

您可以创建一个 Lambda 函数，然后在用户池操作 [如用户注册、确认和登录（身份验证）] 期间使用 Lambda 触发器触发该函数。您可以添加身份验证质询、迁移用户和自定义验证消息。

Lambda 触发器可以自定义用户在您的用户群体中启动操作后 Amazon Cognito 向用户提供的响应。例如，您可以阻止原本会成功的用户登录，也可以向您的 ID 令牌添加声明。他们还可以对您的 AWS 环境、外部 API、数据库或身份存储运行即时操作。例如，迁移用户触发器可以将外部操作与 Amazon Cognito 中的更改相结合：您可以在外部目录中查找用户信息，然后根据该外部信息设置新用户的属性。

当您为用户群体分配 Lambda 触发器时，Amazon Cognito 会中断其原定设置流程，以从您的函数请求信息。Amazon Cognito 生成 JSON 事件并将其传递给您的函数。该事件包含有关您的用户旨在创建用户账户、登录、重置密码或更新属性的请求的信息。然后，您的函数有机会采取行动，或者将事件原封不动地发回。

下表总结了使用 Lambda 触发器自定义用户池操作的一些方法：

用户池流	运算	描述
自定义身份验证流程	定义身份验证质询	确定自定义身份验证流中的下一个挑战
	创建身份验证质询	在自定义身份验证流中创建挑战
	验证身份验证质询响应	确定响应在自定义身份验证流中是正确的
身份验证事件	the section called “身份验证前 Lambda 触发器” (p. 128)	自定义验证以接受或拒绝登录请求
	the section called “身份验证后 Lambda 触发器” (p. 131)	记录自定义分析的事件

用户池流	运算	描述
	the section called “令牌生成前 Lambda 触发器” (p. 143)	增加或隐藏令牌声明
注册	the section called “注册前 Lambda 触发器” (p. 120)	执行接受或拒绝注册请求的自定义验证
	the section called “确认后 Lambda 触发器” (p. 125)	为自定义分析添加自定义欢迎消息或事件日志记录
	the section called “迁移用户 Lambda 触发器” (p. 147)	将用户从现有用户目录迁移到用户池
消息	the section called “自定义消息 Lambda 触发器” (p. 151)	执行消息的高级自定义和本地化
令牌创建	the section called “令牌生成前 Lambda 触发器” (p. 143)	添加或删除 ID 令牌中的属性
电子邮件和 SMS 第三方提供商	the section called “自定义发件人 Lambda 触发器” (p. 155)	使用第三方提供商发送 SMS 和电子邮件

主题

- 重要注意事项 (p. 112)
- 添加用户池 Lambda 触发器 (p. 113)
- 用户池 Lambda 触发器事件 (p. 114)
- 用户池 Lambda 触发器通用参数 (p. 114)
- 将 API 操作连接到 Lambda 触发器 (p. 115)
- 将 Lambda 触发器连接到用户群体功能操作 (p. 118)
- 注册前 Lambda 触发器 (p. 120)
- 确认后 Lambda 触发器 (p. 125)
- 身份验证前 Lambda 触发器 (p. 128)
- 身份验证后 Lambda 触发器 (p. 131)
- 自定义身份验证质询 Lambda 触发器 (p. 134)
- 令牌生成前 Lambda 触发器 (p. 143)
- 迁移用户 Lambda 触发器 (p. 147)
- 自定义消息 Lambda 触发器 (p. 151)
- 自定义发件人 Lambda 触发器 (p. 155)

重要注意事项

在使用 Lambda 函数之前，请考虑以下信息：

- 除了自定义发件人 Lambda 触发器之外，Amazon Cognito 会同步调用 Lambda 函数。Amazon Cognito 调用您的 Lambda 函数时，函数必须在 5 秒内响应。如果并非如此，并且可以重试调用，则 Amazon Cognito 会重试调用。3 次尝试失败后，该函数将超时。您无法更改此 5 秒钟超时值。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 编程模型。

  Amazon Cognito 不会重试返回调用错误且 HTTP 状态代码为 500-599 的函数调用。这些代码表示配置问题导致 Lambda 无法启动该函数。有关更多信息，请参阅 AWS Lambda 中的错误处理和自动重试。

- 如果您删除某个 Lambda 触发器，必须更新用户池中的相应触发器。例如，如果您删除身份验证后触发器，则必须在相应用户池中将 Post authentication（身份验证后）触发器设置为 none（无）。
- 如果您的用户使用 Amazon Cognito 托管 UI，则他们可以看到 Lambda 触发器在查询参数（由 Amazon Cognito 添加到回调 URL）中生成的错误，但自定义发件人 Lambda 触发器除外。最佳做法是，仅在 Lambda 函数中生成您希望用户看到的错误。在 Lambda 触发器本身中记录任何敏感或调试信息。
- 您可以将另一个 AWS 账户中的 Lambda 函数添加为用户群体的触发器。您必须使用 CreateUserPool 和 UpdateUserPool API 操作或它们在 AWS CloudFormation 和 AWS CLI 中的等效操作添加跨账户触发器。您无法在 AWS Management Console中添加跨账户函数。
- 当您在 Amazon Cognito 控制台中添加 Lambda 触发器时，Amazon Cognito 会向您的函数添加一个基于资源的策略，允许您的用户群体调用该函数。当您在 Amazon Cognito 控制台之外创建 Lambda 触发器（包括跨账户函数）时，您必须向 Lambda 函数的基于资源的策略添加权限。您添加的权限必须允许 Amazon Cognito 代表您的用户群体调用函数。您可以从 Lambda 控制台添加权限或者使用 Lambda AddPermission API 操作。

  Lambda 基于资源的策略示例

  以下 Lambda 基于资源的策略示例授予 Amazon Cognito 有限调用 Lambda 函数的能力。Amazon Cognito 只能在代表 `aws:SourceArn` 中的用户池和 `aws:SourceAccount` 条件中的账户时才能调用函数。

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

添加用户池 Lambda 触发器

使用控制台添加用户池 Lambda 触发器

1. 使用 Lambda 控制台创建 Lambda 函数。有关 Lambda 函数的更多信息，请参阅《AWS Lambda 开发人员指南》https://docs.aws.amazon.com/lambda/latest/dg/。
2. 转到 Amazon Cognito 控制台，然后选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 User pool properties（用户池属性）选项卡，并找到 Lambda triggers（Lambda 触发器）。
5. 选择 Add a Lambda trigger（添加 Lambda 触发器）。
6. 基于您希望自定义的身份验证阶段，选择 Lambda 触发器 Category（类别）。

7. 选择 Assign Lambda function（分配 Lambda 函数），然后在同一个 AWS 区域 中选择一个函数作为您的用户池。

   Note

   如果您的 AWS Identity and Access Management (IAM) 凭证有权更新 Lambda 函数，Amazon Cognito 将添加基于 Lambda 资源的策略。通过此政策，Amazon Cognito 可以调用您选择的函数。如果登录凭证没有足够的 IAM 权限，则必须单独更新基于资源的策略。有关更多信息，请参阅the section called “重要注意事项” (p. 112)。
8. 选择 Save changes（保存更改）。
9. 您可以在 Lambda 控制台中使用 CloudWatch 记录您的 Lambda 函数。有关更多信息，请参阅访问适用于 Lambda 的 CloudWatch Logs。

用户池 Lambda 触发器事件

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，Lambda 函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。此事件显示了 Lambda 触发器通用参数：

JSON

```
{
    "version": "string",
    "triggerSource": "string",
    "region": AWSRegion,
    "userPoolId": "string",
    "userName": "string",
    "callerContext":
        {
            "awsSdkVersion": "string",
            "clientId": "string"
        },
    "request":
        {
            "userAttributes": {
                "string": "string",
                ....
            }
        },
    "response": {}
}
```

用户池 Lambda 触发器通用参数

version

您的 Lambda 函数的版本号。

triggerSource

触发 Lambda 函数的事件的名称。有关每个 triggerSource 的说明，请参阅将 Lambda 触发器连接到用户群体功能操作 (p. 118)。

区域

AWS 区域，作为 `AWSRegion` 实例。

userPoolId

用户池的 ID。

userName

当前用户的用户名。

callerContext

有关请求和代码环境的元数据。它包含 awsSdkVersion 和 clientId 字段。

awsSdkVersion

生成请求的 AWS SDK 版本。

clientId

用户群体应用程序客户端的 ID。

request

您的用户的 API 请求的详细信息。它包括以下字段以及触发器特定的任何请求参数。例如，Amazon Cognito 发送到预身份验证触发器的事件也将包含一个 `userNotFound` 参数。当您的用户尝试使用未注册的用户名登录时，您可以处理此参数的值以执行自定义操作。

userAttributes

用户属性名称和值的一个或多个键值对，例如 `"email": "john@example.com"`。

response

此参数在原始请求中不包含任何信息。Lambda 函数必须将整个事件返回给 Amazon Cognito，并将任何返回参数添加到 `response`。要查看您的函数可以包含哪些返回参数，请参阅要使用的触发器的文档。

将 API 操作连接到 Lambda 触发器

以下部分介绍 Amazon Cognito 从您的用户群体中的活动调用的 Lambda 触发器。

当您的应用程序通过 Amazon Cognito 用户群体 API、托管 UI 或用户群体端点登录用户时，Amazon Cognito 会根据会话上下文调用您的 Lambda 函数。有关 Amazon Cognito 用户群体 API 和用户群体端点的更多信息，请参阅使用 Amazon Cognito 用户群体 API 和用户群体端点 (p. 47)。以下各节中的表格描述了导致 Amazon Cognito 调用函数的事件，以及 Amazon Cognito 在请求中包含的 `triggerSource` 字符串。

主题

- Amazon Cognito API 中的 Lambda 触发器 (p. 115)
- 托管 UI 中的 Amazon Cognito 本地用户的 Lambda 触发器 (p. 117)
- 针对联合身份用户的 Lambda 触发器 (p. 117)

Amazon Cognito API 中的 Lambda 触发器

下表描述了 Lambda 触发器的源字符串，当您的应用程序创建、登录或更新本地用户时，Amazon Cognito 可以调用这些触发器。

Amazon Cognito API 中的本地用户触发器来源

API 操作	Lambda 触发器	触发器源
AdminCreateUser	注册前	`PreSignUp_AdminCreateUser`
	令牌生成前	`TokenGeneration_NewPasswordChallenge`
	自定义消息	`CustomMessage_AdminCreateUser`
	自定义电子邮件发件人	`CustomEmailSender_AdminCreateUser`

API 操作	Lambda 触发器	触发器源
	自定义 SMS 发送人	CustomSMSSender_AdminCreateUser
SignUp	注册前	PreSignUp_SignUp
	自定义消息	CustomMessage_SignUp
	自定义电子邮件发件人	CustomEmailSender_SignUp
	自定义 SMS 发送人	CustomSMSSender_SignUp
ConfirmSignUp AdminConfirmSignUp	确认后	PostConfirmation_ConfirmSignUp
InitiateAuth AdminInitiateAuth	身份验证前	PreAuthentication_Authentication
	定义身份验证质询	DefineAuthChallenge_Authentication
	创建身份验证质询	CreateAuthChallenge_Authentication
	令牌生成前	TokenGeneration_Authentication TokenGeneration_AuthenticateDevice TokenGeneration_RefreshTokens
	迁移用户	UserMigration_Authentication
	自定义消息	CustomMessage_Authentication
	自定义电子邮件发件人	CustomEmailSender_AccountTakeOverNot
	自定义 SMS 发送人	CustomSMSSender_Authentication
ForgotPassword	迁移用户	UserMigration_ForgotPassword
	自定义消息	CustomMessage_ForgotPassword
	自定义电子邮件发件人	CustomEmailSender_ForgotPassword
	自定义 SMS 发送人	CustomSMSSender_ForgotPassword
ConfirmForgotPassword	确认后	PostConfirmation_ConfirmForgotPassw
UpdateUserAttributes AdminUpdateUserAttributes	自定义消息	CustomMessage_UpdateUserAttribute
	自定义电子邮件发件人	CustomEmailSender_UpdateUserAttribut
	自定义 SMS 发送人	CustomSMSSender_UpdateUserAttribute
VerifyUserAttributes	自定义消息	CustomMessage_VerifyUserAttribute
	自定义电子邮件发件人	CustomEmailSender_VerifyUserAttribut
	自定义 SMS 发送人	CustomSMSSender_VerifyUserAttribute

托管 UI 中的 Amazon Cognito 本地用户的 Lambda 触发器

下表描述了 Lambda 触发器的源字符串，当本地用户使用托管 UI 登录您的用户群体时，Amazon Cognito 可以调用这些触发器。

托管 UI 中的原生用户触发器源

托管 UI URI	Lambda 触发器	触发器源
/signup	注册前	PreSignUp_SignUp
	自定义消息	CustomMessage_SignUp
	自定义电子邮件发件人	CustomEmailSender_SignUp
	自定义 SMS 发送人	CustomSMSSender_SignUp
/confirmuser	确认后	PostConfirmation_ConfirmSignUp
/login	身份验证前	PreAuthentication_Authentication
	定义身份验证质询	DefineAuthChallenge_Authentication
	创建身份验证质询	CreateAuthChallenge_Authentication
	令牌生成前	TokenGeneration_Authentication TokenGeneration_AuthenticateDevice TokenGeneration_RefreshTokens
	迁移用户	UserMigration_Authentication
	自定义消息	CustomMessage_Authentication
	自定义电子邮件发件人	CustomEmailSender_AccountTakeOverNot
	自定义 SMS 发送人	CustomSMSSender_Authentication
/forgotpassword	迁移用户	UserMigration_ForgotPassword
	自定义消息	CustomMessage_ForgotPassword
	自定义电子邮件发件人	CustomEmailSender_ForgotPassword
	自定义 SMS 发送人	CustomSMSSender_ForgotPassword
/confirmforgotpassword	确认后	PostConfirmation_ConfirmForgotPasswo

针对联合身份用户的 Lambda 触发器

您可以使用以下 Lambda 触发器，为使用联合身份提供商登录的用户自定义用户池工作流。

> **Note**
>
> 联合用户可以使用 Amazon Cognito 托管 UI 进行登录，也可以生成对 对端点授权 (p. 557) 的请求，以静默方式将他们重新导向到其身份提供者登录页面。您无法使用 Amazon Cognito 用户群体 API 登录联合用户。

联合身份用户触发器源

登录事件	Lambda 触发器	触发器源
首次登录	注册前	PreSignUp_ExternalProvider
	确认后	PostConfirmation_ConfirmSignUp
	令牌生成前	TokenGeneration_HostedAuth
后续登录	身份验证前	PreAuthentication_Authentication
	身份验证后	PostAuthentication_Authentication
	令牌生成前	TokenGeneration_HostedAuth

联合身份登录不会在您的用户池中调用任何 自定义身份验证质询 Lambda 触发器 (p. 134)、迁移用户 Lambda 触发器 (p. 147)、自定义消息 Lambda 触发器 (p. 151) 或者自定义发件人 Lambda 触发器 (p. 155)。

将 Lambda 触发器连接到用户群体功能操作

每个 Lambda 触发器都在您的用户群体中发挥功能作用。例如，触发器可以修改您的注册流程，或添加自定义身份验证质询。Amazon Cognito 发送到 Lambda 函数的事件可以反映构成该函数角色的多个操作之一。例如，当您的用户注册时以及当您创建用户时，Amazon Cognito 会调用预注册触发器。同一功能角色的每个不同案例都有其自身的 `triggerSource` 值。您的 Lambda 函数可以根据调用该函数的操作以不同的方式处理传入事件。

当事件对应于触发器源时，Amazon Cognito 还会调用所有分配的函数。例如，当用户登录到您分配了迁移用户和预身份验证触发器的用户群体时，他们会同时激活这两个触发器。

注册、确认和登录 (身份验证) 触发器

触发器	triggerSource 值	事件
注册前	PreSignUp_SignUp	注册前。
注册前	PreSignUp_AdminCreateUser	在管理员创建新用户时做好注册准备。
注册前	PreSignUp_ExternalProvider	适用于外部身份提供商的注册前。
确认后	PostConfirmation_ConfirmSignUp	注册后确认。
确认后	PostConfirmation_ConfirmForgotPassword	忘记密码后确认。
身份验证前	PreAuthentication_Authentication	身份验证前。
身份验证后	PostAuthentication_Authentication	身份验证后。

自定义身份验证质询触发器

触发器	triggerSource 值	事件
定义身份验证质询	DefineAuthChallenge_Authentication	定义身份验证质询。
创建身份验证质询	CreateAuthChallenge_Authentication	创建身份验证质询。

触发器	triggerSource 值	事件
验证身份验证质询	VerifyAuthChallengeResponse	验证身份验证质询响应。

令牌生成前触发器

触发器	triggerSource 值	事件
令牌生成前	TokenGeneration_HostedAuth	Amazon Cognito 从您托管的 UI 登录页面对用户进行身份验证。
令牌生成前	TokenGeneration_Authentication	用户身份验证流程完成。
令牌生成前	TokenGeneration_NewPasswordChallenge	管理员创建用户。当用户必须更改临时密码时，Amazon Cognito 调用此项。
令牌生成前	TokenGeneration_AuthenticateDevice	记住用户设备身份验证。
令牌生成前	TokenGeneration_RefreshTokens	用户尝试刷新身份和访问令牌时调用。

迁移用户触发器

触发器	triggerSource 值	事件
用户迁移	UserMigration_Authentication	用户登录时进行迁移。
用户迁移	UserMigration_ForgotPassword	忘记密码流程中的用户迁移。

自定义消息触发器

触发器	triggerSource 值	事件
自定义消息	CustomMessage_SignUp	用户在您的用户池中注册时的自定义消息。
自定义消息	CustomMessage_AdminCreateUser	当您创建用户作为管理员并且 Amazon Cognito 向他们发送临时密码时的自定义消息。
自定义消息	CustomMessage_ResendCode	现有用户请求新的确认代码时的自定义消息。
自定义消息	CustomMessage_ForgotPassword	用户请求重置密码时的自定义消息。
自定义消息	CustomMessage_UpdateUserAttribute	用户更改其电子邮件地址或电话号码并且 Amazon Cognito 发送验证代码时的自定义消息。
自定义消息	CustomMessage_VerifyUserAttribute	用户添加电子邮件地址或电话号码并且 Amazon Cognito 发送验证代码时的自定义消息。
自定义消息	CustomMessage_Authentication	配置了 SMS MFA 的用户登录时的自定义消息。

注册前 Lambda 触发器

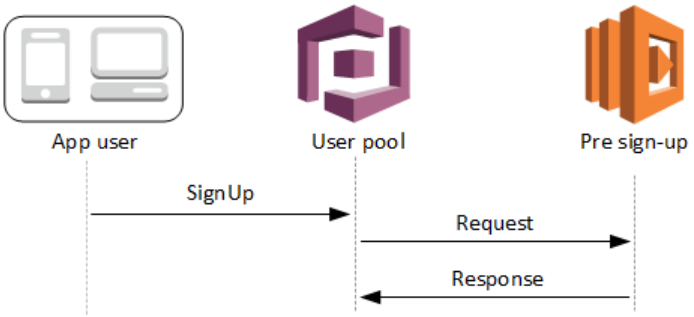
在 Amazon Cognito 注册新用户之前不久，它会激活预注册 AWS Lambda 函数。在注册过程中，您可以使用此函数执行自定义验证，并根据验证结果接受或拒绝注册请求。

主题

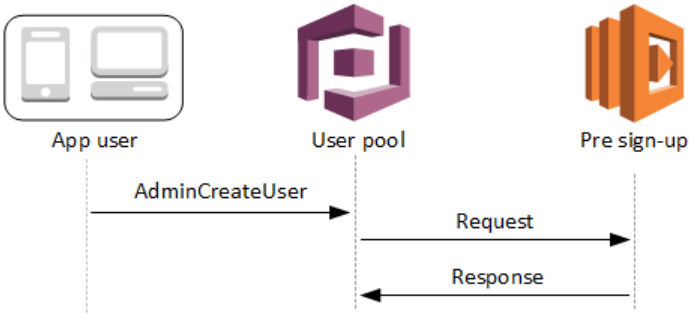
- 注册前 Lambda 流程 (p. 120)
- 注册前 Lambda 触发器参数 (p. 120)
- 注册教程 (p. 122)
- 注册前示例：从注册的域自动确认用户 (p. 122)
- 注册前示例：自动确认和自动验证所有用户 (p. 124)
- 注册前示例：如果用户名少于五个字符，则拒绝注册 (p. 125)

注册前 Lambda 流程

客户端注册流程



服务器注册流程



请求包括来自客户端的验证数据。此数据来自传递给用户池 SignUp 和 AdminCreateUser API 方法的 `ValidationData` 值。

注册前 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "userAttributes": {
```

```
            "string": "string",
            . . .
        },
        "validationData": {
            "string": "string",
            . . .
         },
        "clientMetadata": {
            "string": "string",
            . . .
         }
    },

    "response": {
        "autoConfirmUser": "boolean",
        "autoVerifyPhone": "boolean",
        "autoVerifyEmail": "boolean"
    }
}
```

注册前请求参数

userAttributes

表示用户属性的一个或多个名称/值对。属性名称是键。

validationData

在用户注册请求中包含验证数据的一个或多个名称/值对。设置验证数据，然后在用户注册请求中从客户端传递。您可以使用 InitiateAuth 和 AdminInitiateAuth API 操作中的 ClientMetadata 参数将此数据传递给 Lambda 函数。

clientMetadata

一个或多个键值对，您可以将其作为自定义输入内容提供给为注册前触发器指定的 Lambda 函数。您可以使用以下 API 操作中的 ClientMetadata 参数将此数据传递给 Lambda 函数：AdminCreateUser、AdminRespondToAuthChallenge、ForgotPassword 和 SignUp。

注册前响应参数

在响应中，如果您想要自动确认用户，则您可以将 `autoConfirmUser` 设置为 `true`。您可以将 `autoVerifyEmail` 设置为 `true`，以自动验证用户的电子邮件。您可以将 `autoVerifyPhone` 设置为 `true`，以自动验证用户的电话号码。

> **Note**
>
> `AdminCreateUser` API 触发注册前 Lambda 时，Amazon Cognito 会忽略响应参数 `autoVerifyPhone`、`autoVerifyEmail` 和 `autoConfirmUser`。

autoConfirmUser

设置为 `true` 以自动确认用户，否则设置为 `false`。

autoVerifyEmail

设置为 `true` 可以设置为所注册用户已通过验证的电子邮件地址，否则为 `false`。如果 `autoVerifyEmail` 设置为 `true`，则 `email` 属性必须具有有效的非空值。否则将出现错误，用户将无法完成注册。

如果选择 `email` 属性作为别名，则在设置了 `autoVerifyEmail` 时将为用户的电子邮件地址创建别名。如果已存在具有该电子邮件地址的别名，则别名将移动到新用户，以前用户的电子邮件地址将标记为未验证。有关更多信息，请参阅自定义登录属性 (p. 246)。

autoVerifyPhone

设置为 `true` 可以设置为所注册用户已通过验证的电话号码，否则为 `false`。如果 `autoVerifyPhone` 设置为 `true`，则 `phone_number` 属性必须具有有效的非空值。否则将出现错误，用户将无法完成注册。

如果选择 `phone_number` 属性作为别名，则在设置了 `autoVerifyPhone` 时将为用户的电话号码创建别名。如果已存在具有该电话号码的别名，则别名将移动到新用户，以前用户的电话号码将标记为未验证。有关更多信息，请参阅自定义登录属性 (p. 246)。

注册教程

注册前 Lambda 函数在用户注册前触发。请参阅这些适用于 JavaScript、Android 和 iOS 的 Amazon Cognito 注册教程。

平台	教程
JavaScript 身份开发工具包	通过 JavaScript 注册用户
Android 身份开发工具包	通过 Android 注册用户
iOS 身份开发工具包	通过 iOS 注册用户

注册前示例：从注册的域自动确认用户

您可以使用注册前 Lambda 触发器添加自定义逻辑，以验证注册您的用户池的新用户。这是一个示例 JavaScript 程序，演示如何注册新用户。它将在身份验证过程中调用注册前 Lambda 触发器。

JavaScript

```
var attributeList = [];
var dataEmail = {
    Name : 'email',
    Value : '...' // your email here
};
var dataPhoneNumber = {
    Name : 'phone_number',
    Value : '...' // your phone number here with +country code and no delimiters in
 front
};

var dataEmailDomain = {
    Name: "custom:domain",
    Value: "example.com"
}
var attributeEmail =
new AmazonCognitoIdentity.CognitoUserAttribute(dataEmail);
var attributePhoneNumber =
new AmazonCognitoIdentity.CognitoUserAttribute(dataPhoneNumber);
var attributeEmailDomain =
new AmazonCognitoIdentity.CognitoUserAttribute(dataEmailDomain);

attributeList.push(attributeEmail);
attributeList.push(attributePhoneNumber);
attributeList.push(attributeEmailDomain);

var cognitoUser;
userPool.signUp('username', 'password', attributeList, null, function(err, result){
    if (err) {
```

```
        alert(err);
        return;
    }
    cognitoUser = result.user;
    console.log('user name is ' + cognitoUser.getUsername());
});
```

这是一个示例 Lambda 触发器，在注册前使用用户池注册前 Lambda 触发器调用。它使用自定义属性 custom:domain 自动确认来自特定电子邮件域的新用户。任何不在自定义域中的新用户都将添加到用户池，但不会自动确认。

Node.js

```
exports.handler = (event, context, callback) => {
    // Set the user pool autoConfirmUser flag after validating the email domain
    event.response.autoConfirmUser = false;

    // Split the email address so we can compare domains
    var address = event.request.userAttributes.email.split("@")

    // This example uses a custom attribute "custom:domain"
    if (event.request.userAttributes.hasOwnProperty("custom:domain")) {
        if ( event.request.userAttributes['custom:domain'] === address[1]) {
            event.response.autoConfirmUser = true;
        }
    }

    // Return to Amazon Cognito
    callback(null, event);
};
```

Python

```
def lambda_handler(event, context):
    # It sets the user pool autoConfirmUser flag after validating the email domain
    event['response']['autoConfirmUser'] = False

    # Split the email address so we can compare domains
    address = event['request']['userAttributes']['email'].split('@')

    # This example uses a custom attribute 'custom:domain'
    if 'custom:domain' in event['request']['userAttributes']:
        if event['request']['userAttributes']['custom:domain'] == address[1]:
            event['response']['autoConfirmUser'] = True

    # Return to Amazon Cognito
    return event
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
    "request": {
        "userAttributes": {
            "email": "testuser@example.com",
            "custom:domain": "example.com"
        }
```

```
    },
    "response": {}
}
```

注册前示例：自动确认和自动验证所有用户

此示例确认所有用户并将用户的 email 和 phone_number 属性设置为“已验证”（如果该属性存在）。此外，如果启用了别名，当设置了自动验证时，将为 phone_number 和 email 创建别名。

> **Note**
>
> 如果已存在具有相同电话号码的别名，则别名将移动到新用户，以前用户的 phone_number 将标记为未验证。电子邮件地址也是如此。要防止出现这种情况，可以使用用户池 ListUsers API 查看现有用户是否已在使用新用户的电话号码或电子邮件地址作为别名。

Node.js

```
const handler = async (event) => {
  // Confirm the user
  event.response.autoConfirmUser = true;
  // Set the email as verified if it is in the request
  if (event.request.userAttributes.hasOwnProperty("email")) {
    event.response.autoVerifyEmail = true;
  }

  // Set the phone number as verified if it is in the request
  if (event.request.userAttributes.hasOwnProperty("phone_number")) {
    event.response.autoVerifyPhone = true;
  }

  return event;
};

export { handler };
```

Python

```
def lambda_handler(event, context):
    # Confirm the user
    event['response']['autoConfirmUser'] = True

    # Set the email as verified if it is in the request
    if 'email' in event['request']['userAttributes']:
        event['response']['autoVerifyEmail'] = True

    # Set the phone number as verified if it is in the request
    if 'phone_number' in event['request']['userAttributes']:
        event['response']['autoVerifyPhone'] = True

    # Return to Amazon Cognito
    return event
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
```

```
  "request": {
    "userAttributes": {
      "email": "user@example.com",
      "phone_number": "+12065550100"
    }
  },
  "response": {}
}
```

注册前示例：如果用户名少于五个字符，则拒绝注册

此示例检查注册请求中用户名的长度。如果用户输入的名称长度少于五个字符，则该示例将返回错误。

Node.js

```
exports.handler = (event, context, callback) => {
    // Impose a condition that the minimum length of the username is 5 is imposed on
 all user pools.
    if (event.userName.length < 5) {
        var error = new Error("Cannot register users with username less than the
 minimum length of 5");
        // Return error to Amazon Cognito
        callback(error, event);
    }
    // Return to Amazon Cognito
    callback(null, event);
};
```

Python

```
def lambda_handler(event, context):
    if len(event['userName']) < 5:
        raise Exception("Cannot register users with username less than the minimum
 length of 5")
    # Return to Amazon Cognito
    return event
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
  "userName": "rroe",
  "response": {}
}
```

确认后 Lambda 触发器

Amazon Cognito 在确认新用户后调用此触发器，让您能够发送自定义消息或添加自定义逻辑。例如，您可以使用此触发器来收集新的用户数据。

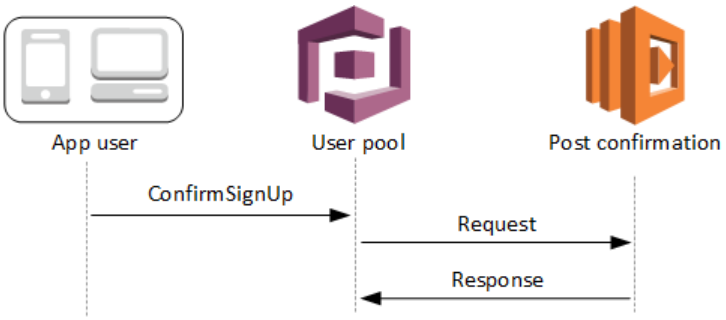
请求包含已确认用户的当前属性。

主题

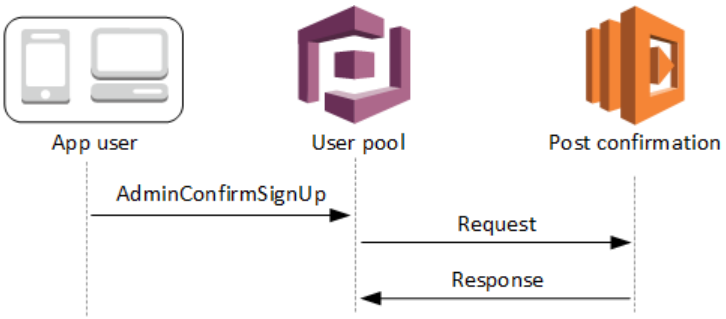
- 确认后 Lambda 流程 (p. 126)
- 确认后 Lambda 触发器参数 (p. 126)
- 用户确认教程 (p. 127)
- 确认后示例 (p. 127)

确认后 Lambda 流程

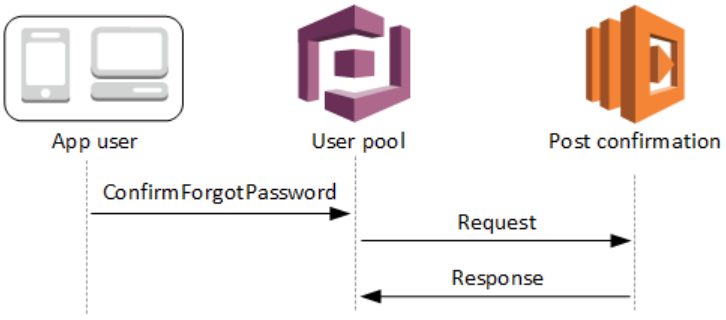
客户端确认注册流程



服务器确认注册流程



确认忘记密码流程



确认后 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
```

```
            "userAttributes": {
                "string": "string",
                . . .
            },
            "clientMetadata": {
             "string": "string",
             . . .
            }
        },
    "response": {}
}
```

确认后请求参数

userAttributes

表示用户属性的一个或多个键值对。

clientMetadata

一个或多个键值对，您可以将其作为自定义输入内容提供给为确认后触发器指定的 Lambda 函数。您可以使用以下 API 操作中的 ClientMetadata 参数将此数据传递给 Lambda 函数：AdminConfirmSignUp、ConfirmForgotPassword、ConfirmSignUp 和 SignUp。

确认后响应参数

预计响应中没有其他返回信息。

用户确认教程

确认后 Lambda 函数在 Amazon Cognito 确认新用户后触发。请参阅这些适用于 JavaScript、Android 和 iOS 的用户确认教程。

平台	教程
JavaScript 身份开发工具包	通过 JavaScript 确认用户
Android 身份开发工具包	通过 Android 确认用户
iOS 身份开发工具包	通过 iOS 确认用户

确认后示例

此示例 Lambda 函数将使用 Amazon SES 向用户发送确认电子邮件。有关更多信息，请参阅 Amazon Simple Email Service 开发人员指南。

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 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
    "request": {
        "userAttributes": {
            "email": "user@example.com",
            "email_verified": true
        }
    },
    "response": {}
}
```

身份验证前 Lambda 触发器

当用户尝试登录时，Amazon Cognito 会调用此触发器，以便您可以创建用于执行准备操作的自定义验证。例如，您可以拒绝身份验证请求或将会话数据记录到外部系统。

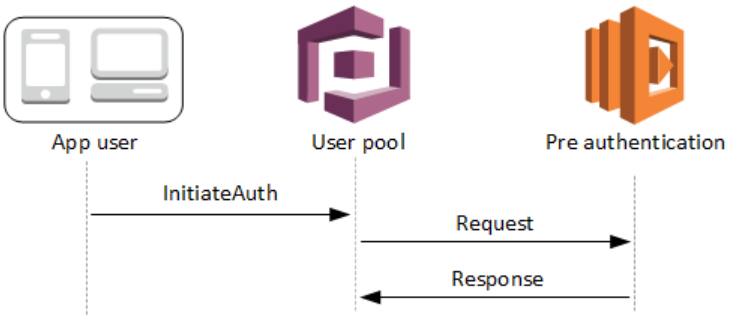
> **Note**
>
> 当用户不存在或在您的用户群体中已经有会话时，此触发器不会激活。

主题

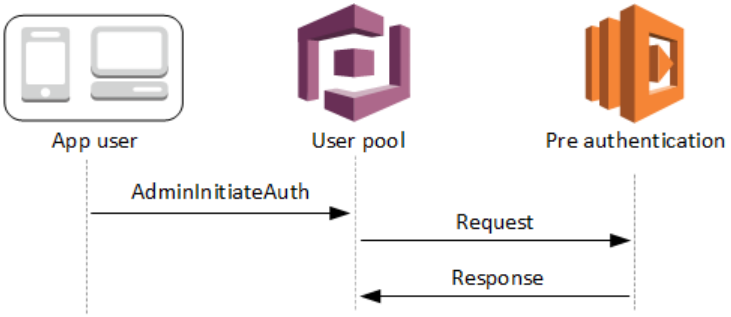
- 身份验证前 Lambda 流程 (p. 129)
- 身份验证前 Lambda 触发器参数 (p. 129)
- 身份验证前示例 (p. 130)

身份验证前 Lambda 流程

客户端身份验证流程



服务器身份验证流程



该请求包含来自 `ClientMetadata` 值的客户端验证数据，该值由应用程序传递到用户池 InitiateAuth 和 AdminInitiateAuth API 操作。

有关更多信息，请参阅用户池身份验证流程 (p. 41)。

身份验证前 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "validationData": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
```

```
    },
    "response": {}
}
```

身份验证前请求参数

userAttributes

表示用户属性的一个或多个名称/值对。

userNotFound

当您将用户池客户端的 `PreventUserExistenceErrors` 设置为 `ENABLED` 时，Amazon Cognito 将填充此布尔值。

validationData

一个或多个键/值对，包含用户的登录请求中的验证数据。要将此数据传递给 Lambda 函数，请在 InitiateAuth 和 AdminInitiateAuth API 操作中使用 ClientMetadata 参数。

身份验证前响应参数

Amazon Cognito 不需要响应中任何额外的返回信息。您的函数可以返回错误以拒绝登录尝试，或者使用 API 操作来查询和修改资源。

身份验证前示例

此示例函数阻止用户使用特定的应用程序客户端登录到您的用户群体。由于预身份验证 Lambda 函数不会在您的用户有现有会话时调用，因此，此函数仅阻止使用您想要屏蔽的应用程序客户端 ID 的新会话。

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 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
    "callerContext": {
        "clientId": "<user pool app client id to be blocked>"
    },
    "response": {}
}
```

身份验证后 Lambda 触发器

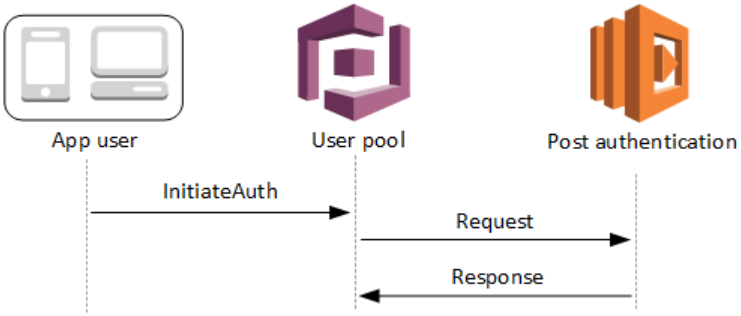
由于 Amazon Cognito 在登录用户后调用此触发器，您可以在 Amazon Cognito 对用户进行身份验证后添加自定义逻辑。

主题

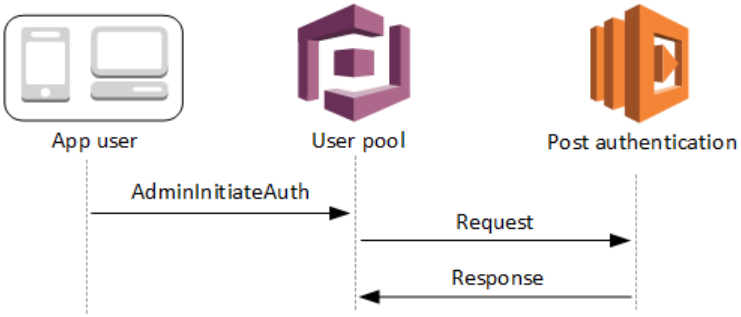
- 身份验证后 Lambda 流程 (p. 131)
- 身份验证后 Lambda 触发器参数 (p. 131)
- 身份验证教程 (p. 132)
- 身份验证后示例 (p. 132)

身份验证后 Lambda 流程

客户端身份验证流程



服务器身份验证流程



有关更多信息，请参阅 用户池身份验证流程 (p. 41)。

身份验证后 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
         },
         "newDeviceUsed": boolean,
         "clientMetadata": {
            "string": "string",
            . . .
           }
        },
    "response": {}
}
```

身份验证后请求参数

newDeviceUsed

此标记指示用户是否已在新设备上登录。Amazon Cognito 仅在用户池的记住的设备值设置为 `Always` 或 `User Opt-In` 时设置此标记。

userAttributes

表示用户属性的一个或多个名称/值对。

clientMetadata

一个或多个键值对，您可以将其作为自定义输入内容提供给为身份验证后触发器指定的 Lambda 函数。要将此数据传递给 Lambda 函数，您可以在 AdminRespondToAuthChallenge 和 RespondToAuthChallenge API 操作中使用 ClientMetadata 参数。在传递到身份验证后函数的请求中，Amazon Cognito 不包括 AdminInitiateAuth 和 InitiateAuth API 操作的 ClientMetadata 参数中传递的数据。

身份验证后响应参数

Amazon Cognito 不需要响应中任何额外的返回信息。您的函数可以使用 API 操作来查询和修改资源，或者将事件元数据记录到外部系统。

身份验证教程

Amazon Cognito 登录用户之后，将会立即激活身份验证后 Lambda 函数。请参阅这些适用于 JavaScript、Android 和 iOS 的登录教程。

平台	教程
JavaScript 身份开发工具包	通过 JavaScript 登录用户
Android 身份开发工具包	通过 Android 登录用户
iOS 身份开发工具包	通过 iOS 登录用户

身份验证后示例

此身份验证后示例 Lambda 函数将成功登录数据发送到 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

```
from __future__ import print_function
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'])

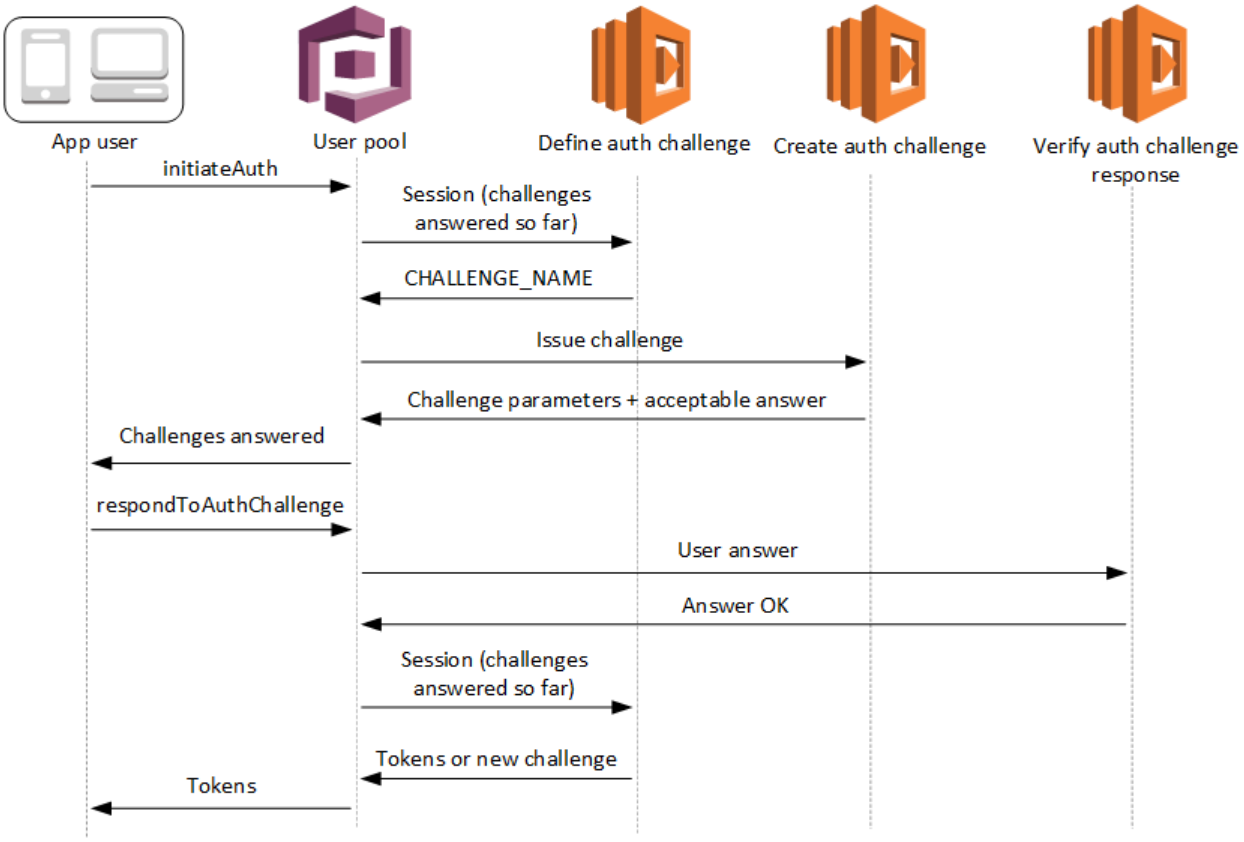
    # Return to Amazon Cognito
    return event
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
  "triggerSource": "testTrigger",
  "userPoolId": "testPool",
  "userName": "testName",
  "callerContext": {
      "clientId": "12345"
  },
  "response": {}
}
```

自定义身份验证质询 Lambda 触发器



在用户池自定义身份验证流程中，这些 Lambda 触发器将发出并验证自己的质询。

定义身份验证质询

Amazon Cognito 调用此触发器以启动自定义身份验证流程。

创建身份验证质询

Amazon Cognito 将在定义身份验证质询之后调用此触发器以创建自定义质询。

验证身份验证质询响应

Amazon Cognito 调用此触发器，以验证终端用户对自定义质询的响应是否有效。

您可以使用这些质询 Lambda 触发器引入新的质询类型。例如，这些质询类型可能包含 CAPTCHA 或动态质询问题。

您可以使用用户池 InitiateAuth 和 RespondToAuthChallenge API 方法将身份验证泛化为两个常见步骤。

在此流程中，用户通过回答连续的质询进行身份验证，直到身份验证失败或用户获得令牌。这两个 API 调用可重复执行以包含不同的质询。

Note

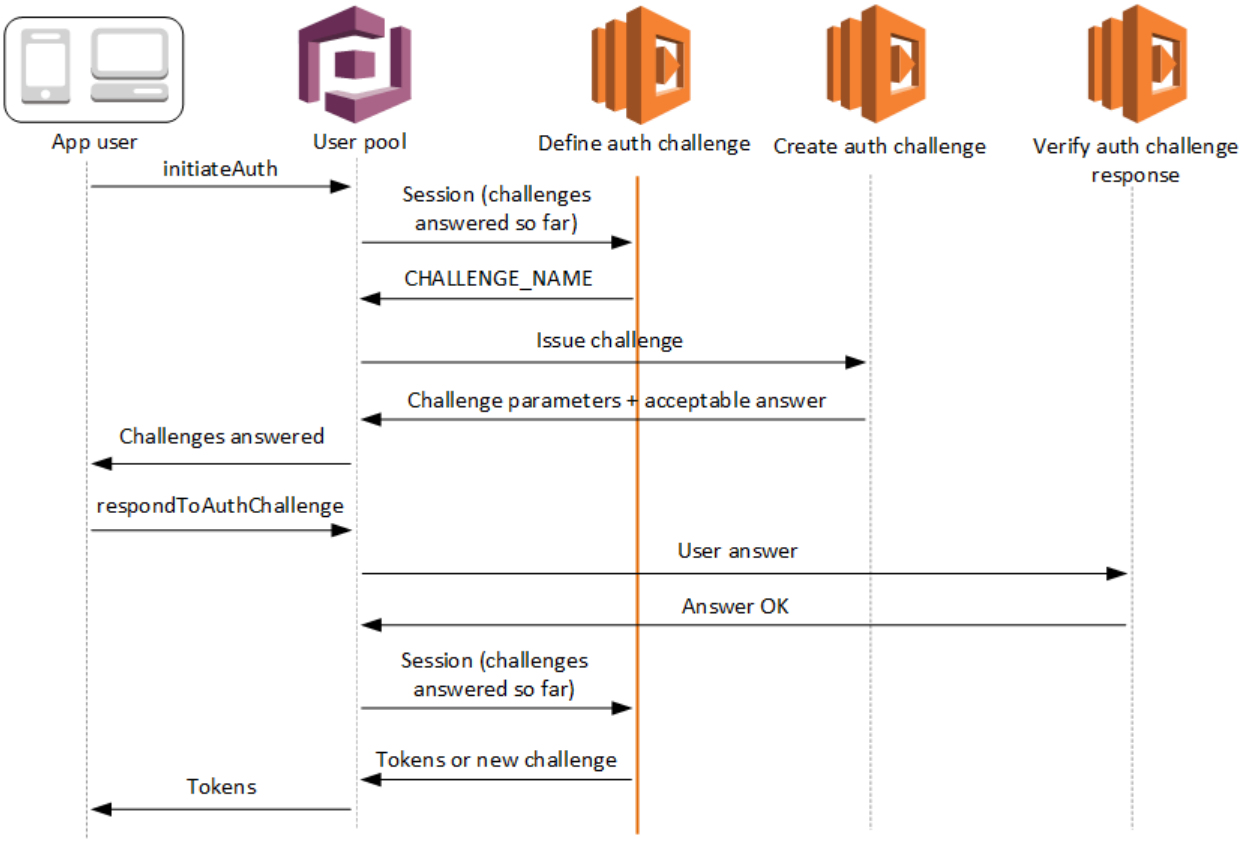
Amazon Cognito 托管的 UI 不支持自定义身份验证流程。

主题

- 定义身份验证质询 Lambda 触发器 (p. 135)
- 创建身份验证质询 Lambda 触发器 (p. 138)

- 验证身份验证质询响应 Lambda 触发器 (p. 141)

定义身份验证质询 Lambda 触发器



定义身份验证质询

Amazon Cognito 调用此触发器以启动自定义身份验证流程。

此 Lambda 触发器的请求包括 `session`。`session` 参数是一个数组，包含在当前身份验证流程中向用户显示的所有质询。请求还包含相应的结果。`session` 数组按照时间顺序存储质询详细信息 (`ChallengeResult`)。质询 `session[0]` 表示用户收到的第一个质询。

您可以让 Amazon Cognito 在发出自定义质询之前验证用户密码。过程概述如下：

1. 您的应用程序使用 `AuthParameters` 映射来调用 `InitiateAuth` 或 `AdminInitiateAuth`，以此来启动登录。参数必须包括 `CHALLENGE_NAME: SRP_A`，以及 `SRP_A` 和 `USERNAME` 的值。
2. Amazon Cognito 使用包含 `challengeName: SRP_A` 和 `challengeResult: true` 的初始会话，调用您定义的身份验证质询 Lambda 触发器。
3. 在收到这些输入后，您的 Lambda 函数发出 `challengeName: PASSWORD_VERIFIER`、`issueTokens: false`、`failAuthentication: false` 响应。
4. 如果密码验证成功，Amazon Cognito 会使用包含 `challengeName: PASSWORD_VERIFIER` 和 `challengeResult: true` 的新会话再次调用您的 Lambda 函数。
5. 为了启动您的自定义质询，Lambda 函数发出 `challengeName: CUSTOM_CHALLENGE`、`issueTokens: false` 和 `failAuthentication: false` 响应。如果您不想启动包含密码验证的自定义身份验证流程，可以使用 `AuthParameters` 映射（包括 `CHALLENGE_NAME: CUSTOM_CHALLENGE`）启动登录。

6. 质询循环将一直重复到所有质询得到应答。

主题

- 定义身份验证质询 Lambda 触发器参数 (p. 136)
- 定义身份验证质询示例 (p. 137)

定义身份验证质询 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
                . . .
        },
        "session": [
            ChallengeResult,
            . . .
        ],
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "challengeName": "string",
        "issueTokens": boolean,
        "failAuthentication": boolean
    }
}
```

定义身份验证质询请求参数

当 Amazon Cognito 调用您的 Lambda 函数时，Amazon Cognito 提供以下参数：

userAttributes

表示用户属性的一个或多个名称/值对。

userNotFound

一个布尔值，当您的用户池客户端将 `PreventUserExistenceErrors` 设置为 `ENABLED` 时，Amazon Cognito 将填充该值。值 `true` 表示用户 ID（用户名、电子邮件地址以及其他详细信息）不匹配任何现有用户。当 `PreventUserExistenceErrors` 设置为 `ENABLED` 时，该服务不会向应用程序通知不存在的用户。我们建议您的 Lambda 函数保持相同的用户体验并考虑延迟。这样，不论用户是否存在，调用方都不会检测到不同的行为。

session

`ChallengeResult` 元素的数组。每个数组包含以下元素：

challengeName

以下质询类型之一：`CUSTOM_CHALLENGE`、`SRP_A`、`PASSWORD_VERIFIER`、`SMS_MFA`、`DEVICE_SRP_AUTH`、`DEVICE_PASSWORD_VERIFIER` 或 `ADMIN_NO_SRP_AUTH`。

当您的定义身份验证质询功能向已设置多重身份验证的用户发出 PASSWORD_VERIFIER 质询时，Amazon Cognito 会随后提出 SMS_MFA 质询。在您的函数中，包括对来自 SMS_MFA 质询的输入事件的处理。您无需从定义身份验证质询函数中调用 SMS_MFA 质询。

Important

在函数确定用户是否已成功通过身份验证以及是否应向其颁发令牌时，请始终检查定义身份验证质询函数中的 challengeName 以及它是否与预期值匹配。

challengeResult

如果用户成功完成质询，则设置为 true，否则设置为 false。

challengeMetadata

您的自定义质询的名称。仅当 challengeName 为 CUSTOM_CHALLENGE 时使用。

clientMetadata

一个或多个键值对，您可以将其作为自定义输入内容提供给为定义身份验证质询触发器指定的 Lambda 函数。要将此数据传递给 Lambda 函数，您可以使用 AdminRespondToAuthChallenge 和 RespondToAuthChallenge API 操作中的 ClientMetadata 参数。调用“定义身份验证质询”函数的请求不包括在 AdminInitiateAuth 和 InitiateAuth API 操作的 ClientMetadata 参数中传递的数据。

定义身份验证质询响应参数

在响应中，您可以返回身份验证流程的下一阶段。

challengeName

一个字符串，其中包含下一质询的名称。如果您希望向您的用户显示新的质询，请在此处指定质询名称。

issueTokens

如果您确定用户已充分完成了身份验证质询，则设置为 true。如果用户没有充分满足质询条件，则设置为 false。

failAuthentication

如果您想要终止当前的身份验证流程，则设置为 true。要继续当前的身份验证流程，请设置为 false。

定义身份验证质询示例

此示例针对身份验证定义一系列质询，并仅在用户成功完成所有质询后发布令牌。

Node.js

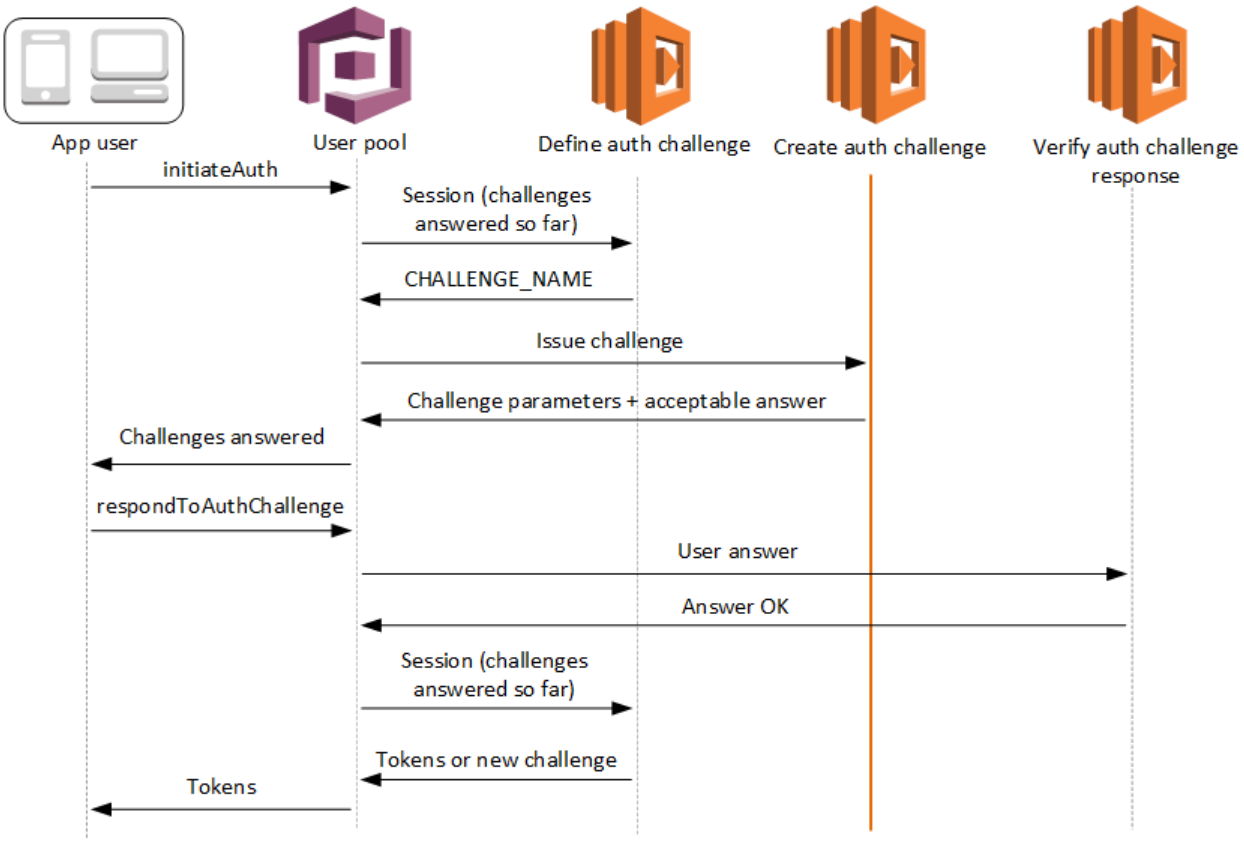
```
const handler = async (event) => {
  if (
    event.request.session.length == 1 &&
    event.request.session[0].challengeName == "SRP_A"
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "PASSWORD_VERIFIER";
  } else if (
    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 }
```

创建身份验证质询 Lambda 触发器



创建身份验证质询

如果指定自定义质询作为定义身份验证质询 触发器的一部分，则 Amazon Cognito 会在定义身份验证质询之后调用此触发器。它将创建一个自定义身份验证流程。

系统调用此 Lambda 触发器来创建要向用户显示的质询。此 Lambda 触发器的请求包括 `challengeName` 和 `session`。`challengeName` 是一个字符串，是向用户显示的下一质询的名称。此属性的值在定义身份验证质询 Lambda 触发器中设置。

质询循环将一直重复到所有质询得到应答。

主题

- 创建身份验证质询 Lambda 触发器参数 (p. 139)
- 创建身份验证质询示例 (p. 140)

创建身份验证质询 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "challengeName": "string",
        "session": [
            ChallengeResult,
            . . .
        ],
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "publicChallengeParameters": {
            "string": "string",
            . . .
        },
        "privateChallengeParameters": {
            "string": "string",
            . . .
        },
        "challengeMetadata": "string"
    }
}
```

创建身份验证质询请求参数

userAttributes

表示用户属性的一个或多个名称/值对。

userNotFound

当为您的用户池客户端将 `PreventUserExistenceErrors` 设置为 `ENABLED` 时，将填充此布尔值。

challengeName

新质询的名称。

session

会话元素是一组 `ChallengeResult` 元素，其中，每个元素包含以下元素：

challengeName

质询类型。以下值之一：`"CUSTOM_CHALLENGE"`、`"PASSWORD_VERIFIER"`、`"SMS_MFA"`、`"DEVICE_SRP_AUTH"`、`"DEVICE_PA` 或 `"ADMIN_NO_SRP_AUTH"`。

challengeResult

如果用户成功完成质询，则设置为 `true`，否则设置为 `false`。

challengeMetadata

您的自定义质询的名称。仅当 `challengeName` 为 `"CUSTOM_CHALLENGE"` 时使用。

clientMetadata

一个或多个键值对，您可以将其作为自定义输入内容提供给为创建身份验证质询触发器指定的 Lambda 函数。您可以在 AdminRespondToAuthChallenge 和 RespondToAuthChallenge API 操作中使用 ClientMetadata 参数，将此数据传递给 Lambda 函数。调用创建身份验证质询函数的请求不包括在 AdminInitiateAuth 和 InitiateAuth API 操作的 ClientMetadata 参数中传递的数据。

创建身份验证质询响应参数

publicChallengeParameters

客户端应用程序要在向用户显示的质询中使用的一个或多个键/值对。此参数应包含所有必要信息，以向用户准确显示质询。

privateChallengeParameters

此参数仅由验证身份验证质询响应 Lambda 触发器使用。此参数应包含所需的所有信息，以验证用户对质询的响应。也就是说，`publicChallengeParameters` 参数包含向用户显示的问题，`privateChallengeParameters` 包含问题的有效答案。

challengeMetadata

您的自定义质询的名称（如果是自定义质询）。

创建身份验证质询示例

CAPTCHA 作为针对用户的质询而创建。CAPTCHA 图像的 URL 作为 `captchaUrl` 添加到公有质询参数中，并且预期答案添加到私有质询参数中。

Node.js

```
const handler = async (event) => {
  if (event.request.challengeName !== "CUSTOM_CHALLENGE") {
    return event;
  }

  if (event.request.session.length === 2) {
    event.response.publicChallengeParameters = {};
```

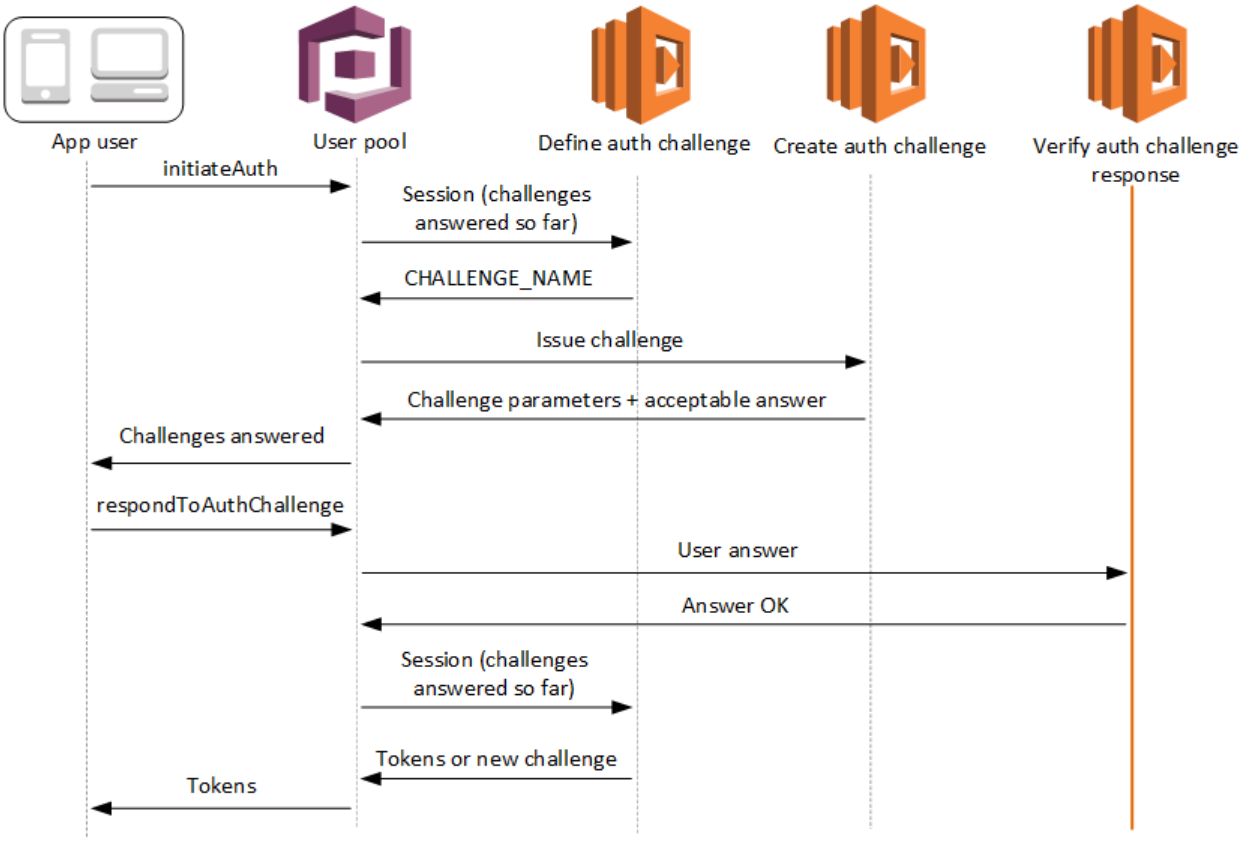
```
    event.response.privateChallengeParameters = {};
    event.response.publicChallengeParameters.captchaUrl = "url/123.jpg";
    event.response.privateChallengeParameters.answer = "5";
  }

  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 }
```

验证身份验证质询响应 Lambda 触发器



验证身份验证质询响应

Amazon Cognito 调用此触发器，以验证用户对自定义身份验证质询的响应是否有效。它是用户池自定义身份验证流程的一部分。

此触发器的请求包括 `privateChallengeParameters` 和 `challengeAnswer` 参数。创建身份验证质询 Lambda 触发器返回 `privateChallengeParameters` 值，并包含用户的预期响应。`challengeAnswer` 参数包含用户对质询的响应。

响应包含 `answerCorrect` 属性。如果用户成功完成质询，Amazon Cognito 会将属性值设置为 `true`。如果用户未成功完成质询，Amazon Cognito 会将属性值设置为 `false`。

质询循环将一直重复，直至用户应答所有质询。

主题

- 验证身份验证质询 Lambda 触发器参数 (p. 142)
- 验证身份验证质询响应示例 (p. 143)

验证身份验证质询 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "privateChallengeParameters": {
            "string": "string",
            . . .
        },
        "challengeAnswer": "string",
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "answerCorrect": boolean
    }
}
```

验证身份验证质询请求参数

userAttributes

此参数包含表示用户属性的一个或多个名称/值对。

userNotFound

当 Amazon Cognito 将您用户池客户端的 `PreventUserExistenceErrors` 设置为 `ENABLED` 时，Amazon Cognito 将填充此布尔值。

privateChallengeParameters

此参数来自创建身份验证质询触发器。为了确定用户是否通过了质询，Amazon Cognito 将参数与用户的 challengeAnswer 进行比较。

此参数包含所需的所有信息，以验证用户对质询的响应。该信息包括 Amazon Cognito 向用户提出的问题 (`publicChallengeParameters`)，以及问题的有效回答 (`privateChallengeParameters`)。只有验证身份验证质询响应 Lambda 触发器使用此参数。

challengeAnswer

此参数值是来自用户对质询响应的应答。

clientMetadata

此参数包含一个或多个键值对，您可以将其作为自定义输入内容提供给用于验证身份验证质询触发器的 Lambda 函数。要将此数据传递给 Lambda 函数，请在 AdminRespondToAuthChallenge 和 RespondToAuthChallenge API 操作中使用 ClientMetadata 参数。在传递到验证身份验证质询函数的请求中，Amazon Cognito 不包括 AdminInitiateAuth 和 InitiateAuth API 操作的 ClientMetadata 参数中传递的数据。

验证身份验证质询响应参数

answerCorrect

如果用户成功完成质询，Amazon Cognito 将此参数设置为 `true`。如果用户未成功完成质询，Amazon Cognito 将此参数设置为 `false`。

验证身份验证质询响应示例

在此示例中，Lambda 函数检查用户对质询的响应是否与预期响应一致。如果用户的响应与预期响应一致，Amazon Cognito 将 `answerCorrect` 参数设置为 `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 };
```

令牌生成前 Lambda 触发器

由于 Amazon Cognito 在令牌生成之前调用此触发器，您可以自定义身份令牌声明。

您可以使用此 Lambda 触发器在 Amazon Cognito 生成身份令牌之前自定义身份令牌。您可以使用此触发器来在身份令牌中添加新声明、更新声明或者隐藏声明。要使用此功能，可以从 Amazon Cognito 用户群体控制台关联 Lambda 函数或通过 AWS Command Line Interface(AWS CLI) 更新用户群体。

您无法修改以下声明：

- `acr`
- `amr`
- `aud`
- `at_hash`
- `auth_time`
- `azp`
- `cognito:username`

- exp
- iat
- identities
- iss
- jti
- nbf
- nonce
- origin_jti
- sub
- token_use

主题

- 令牌生成前 Lambda 触发器源 (p. 144)
- 令牌生成前 Lambda 触发器参数 (p. 144)
- 令牌生成前示例：添加新声明并隐藏现有声明 (p. 146)
- 令牌生成前示例：修改用户的组成员资格 (p. 146)

令牌生成前 Lambda 触发器源

triggerSource 值	事件
TokenGeneration_HostedAuth	通过 Amazon Cognito 托管 UI 登录页进行身份验证时调用。
TokenGeneration_Authentication	用户身份验证流完成之后调用。
TokenGeneration_NewPasswordChallenge	管理员创建用户之后调用。当用户必须更改临时密码时调用此流。
TokenGeneration_AuthenticateDevice	用户设备身份验证结束时调用。
TokenGeneration_RefreshTokens	用户尝试刷新身份和令牌时调用。

令牌生成前 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

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

令牌生成前请求参数

groupConfiguration

包含当前组配置的输入对象。对象包括 `groupsToOverride`、`iamRolesToOverride` 和 `preferredRole`。

groupsToOverride

与接收身份令牌的用户相对应的组名称的列表。

iamRolesToOverride

与这些组对应的当前 AWS Identity and Access Management (IAM) 角色的列表。

preferredRole

指示首选 IAM 角色的字符串。

clientMetadata

一个或多个键值对，您可以指定它们并将它们作为自定义输入提供给 Lambda 函数以用于令牌生成前的触发器。要将此数据传递给 Lambda 函数，请在 AdminRespondToAuthChallenge 和 RespondToAuthChallenge API 操作中使用 ClientMetadata 参数。Amazon Cognito 在传递到令牌生成前函数的请求中，不包括 AdminInitiateAuth 和 InitiateAuth API 操作的 ClientMetadata 参数中传递的数据。

令牌生成前响应参数

claimsToAddOrOverride

要添加或覆盖的声明的一个或多个键值对的映射。对于与组相关的声明，请改用 groupOverrideDetails。

claimsToSuppress

包含您希望 Amazon Cognito 在身份令牌中隐藏的声明的列表。

Note

如果您的函数同时隐藏并替换了声明值，则 Amazon Cognito 会隐藏声明。

groupOverrideDetails

包含当前组配置的输出对象。对象包括 `groupsToOverride`、`iamRolesToOverride` 和 `preferredRole`。

您的函数将 groupOverrideDetails 对象替换为您提供的对象。如果您在响应中提供空的或空对象，则 Amazon Cognito 将隐藏组。要保持现有组配置不变，请将请求的 groupConfiguration 对象的值复制到响应中的 groupOverrideDetails 对象。然后将其传回服务。

Amazon Cognito ID 令牌和访问令牌都包含 `cognito:groups` 声明。在访问令牌和 ID 令牌中，您的 groupOverrideDetails 对象将替换 `cognito:groups` 声明。

令牌生成前示例：添加新声明并隐藏现有声明

此示例使用令牌生成前 Lambda 函数添加新声明和隐藏现有声明。

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 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的测试事件：由于该代码示例不处理任何请求参数，因此您可以使用带有空请求的测试事件。有关常见请求参数的更多信息，请参阅用户池 Lambda 触发器事件 (p. 114)。

JSON

```
{
  "request": {},
  "response": {}
}
```

令牌生成前示例：修改用户的组成员资格

此示例使用令牌生成前 Lambda 函数修改用户的组成员资格。

Node.js

```
const handler = async (event) => {
  event.response = {
    claimsOverrideDetails: {
      groupOverrideDetails: {
        groupsToOverride: ["group-A", "group-B", "group-C"],
        iamRolesToOverride: [
          "arn:aws:iam::XXXXXXXXXXX:role/sns_callerA",
          "arn:aws:iam::XXXXXXXXX:role/sns_callerB",
          "arn:aws:iam::XXXXXXXXXX:role/sns_callerC",
        ],
        preferredRole: "arn:aws:iam::XXXXXXXXXXX:role/sns_caller",
      },
    },
  };

  return event;
};

export { handler };
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
  "request": {},
  "response": {}
}
```

迁移用户 Lambda 触发器

如果用户在使用密码登录时或在使用忘记密码流程时不在用户池中，Amazon Cognito 会调用此触发器。Lambda 函数成功返回后，Amazon Cognito 将在用户池中创建用户。有关利用用户迁移 Lambda 触发器进行身份验证流程的详细信息，请参阅利用用户迁移 Lambda 触发器将用户导入用户池 (p. 183)。

要在用户登录时或在忘记密码流程中，将用户从您的现有用户目录迁移到 Amazon Cognito 用户池，请使用此 Lambda 触发器。

主题

- 迁移用户 Lambda 触发器源 (p. 147)
- 迁移用户 Lambda 触发器参数 (p. 148)
- 示例：使用现有密码迁移用户 (p. 150)

迁移用户 Lambda 触发器源

triggerSource 值	事件
`UserMigration_Authentication`	登录时的用户迁移。
`UserMigration_ForgotPassword`	在忘记密码流程中迁移用户。

迁移用户 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

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
    }
}
```

迁移用户请求参数

userName

用户在登录时输入的用户名。

password

用户在登录时输入的密码。Amazon Cognito 不会在由忘记密码流程发起的请求中发送此值。

validationData

一个或多个键/值对，其中包含用户登录请求中的验证数据。要将此数据传递给 Lambda 函数，您可以在 InitiateAuth 和 AdminInitiateAuth API 操作中使用 ClientMetadata 参数。

clientMetadata

一个或多个键/值对，您可以将其作为自定义输入内容提供给迁移用户触发器的 Lambda 函数。要将此数据传递给 Lambda 函数，您可以在 AdminRespondToAuthChallenge 和 ForgotPassword API 操作中使用 ClientMetadata 参数。

迁移用户响应参数

userAttributes

此字段为必填。

该字段必须包含一个或多个名称/值对，Amazon Cognito 将其存储在用户池的用户配置文件中并用作用户属性。您可以同时包括标准的和自定义的用户属性。自定义属性需要使用 `custom:` 前缀，以便与标准属性区分开来。有关更多信息，请参阅自定义属性。

Note

要在忘记密码流程中重置密码，用户必须拥有经过验证的电子邮件地址或经过验证的电话号码。Amazon Cognito 将包含重置密码代码的消息发送到用户属性中的电子邮件地址或电话号码。

属性	要求
创建用户池时标记为必需的所有属性	如果迁移过程中缺少任何必需的属性，Amazon Cognito 将使用默认值。
`username`	如果您在为用户池配置了用户名登录之外，还配置了别名登录，并且用户输入了有效的别名值作为用户名，则此属性是必需的。该别名值可以是电子邮件地址、首选用户名或电话号码。 如果请求和用户池满足别名要求，则函数的响应必须将收到的 `username` 参数分配给别名属性，此外，响应必须将您自己的值分配给 `username` 属性。如果您的用户池不符合所需的条件，无法将收到的 `username` 发送到别名，则响应中的 `username` 参数必须与请求完全匹配，否则就会被忽略。 Note 在用户池中，`username` 必须唯一。

finalUserStatus

您可以将此参数设置为 `CONFIRMED` 以自动确认用户，这样他们就可以使用之前的密码登录。当您将用户设置为 `CONFIRMED` 时，他们无需执行额外的操作即可登录。如果您未将此属性设置为 `CONFIRMED`，则它会设置为 `RESET_REQUIRED`。

若 `finalUserStatus` 设置为 `RESET_REQUIRED`，则意味着用户在迁移之后，必须在登录时立即更改密码，并且您的客户端应用程序必须在身份验证中处理 `PasswordResetRequiredException`。

Note

使用 Lambda 触发器迁移期间，Amazon Cognito 不强制执行您为用户池配置的密码强度策略。如果密码不符合您配置的密码策略，Amazon Cognito 仍会接受密码，以便它继续迁移用户。要强制实施密码强度策略并拒绝不符合策略的密码，请验证代码中的密码强度。然后，如果密码不符合策略，则将 finalUserStatus 设置为 `RESET_REQUIRED`。

messageAction

您可以将此参数设置为 `SUPPRESS`，以拒绝发送 Amazon Cognito 通常会向新用户发送的欢迎消息。如果您的函数未返回此参数，Amazon Cognito 会发送欢迎消息。

desiredDeliveryMediums

您可以将此参数设置为 `EMAIL` 以通过电子邮件发送欢迎消息，或者设置为 `SMS` 以通过 SMS 发送欢迎消息。如果您的函数未返回此参数，Amazon Cognito 通过 SMS 发送欢迎消息。

forceAliasCreation

如果您将此参数设置为 `TRUE`，并且 UserAttributes 参数中的电话号码或电子邮件地址已作为其他用户的别名存在，则 API 调用会将该别名从以前的用户迁移到新创建的用户。以前的用户无法再使用该别名登录。

如果您将此参数设置为 `FALSE` 而且别名存在，Amazon Cognito 不会迁移用户，并向客户端应用程序返回错误。

如果您不返回此参数，Amazon Cognito 会假定其值为“false”。

enableSMSMFA

将此参数设置为 `true`，要求迁移的用户完成 SMS 短信多重身份验证 (MFA) 才能登录。您的用户池必须启用 MFA。请求参数中的用户属性必须包含电话号码，否则该用户的迁移将失败。

示例：使用现有密码迁移用户

此示例 Lambda 函数使用现有密码迁移用户，并隐藏 Amazon Cognito 发送的欢迎消息。

Node.js

```
const validUsers = {
  belladonna: { password: "Test123", emailAddress: "bella@example.com" },
};

// Replace this mock with a call to a real authentication service.
const authenticateUser = (username, password) => {
  if (validUsers[username] && validUsers[username].password === password) {
    return validUsers[username];
  } else {
    return null;
  }
};

const lookupUser = (username) => {
  const user = validUsers[username];

  if (user) {
    return { emailAddress: user.emailAddress };
  } else {
    return null;
  }
};

const handler = async (event) => {
  if (event.triggerSource == "UserMigration_Authentication") {
    // Authenticate the user with your existing user directory service
    const user = authenticateUser(event.userName, event.request.password);
    if (user) {
      event.response.userAttributes = {
        email: user.emailAddress,
        email_verified: "true",
      };
      event.response.finalUserStatus = "CONFIRMED";
      event.response.messageAction = "SUPPRESS";
    }
  } else if (event.triggerSource == "UserMigration_ForgotPassword") {
    // Look up the user in your existing user directory service
    const user = lookupUser(event.userName);
    if (user) {
      event.response.userAttributes = {
        email: user.emailAddress,
        // Required to enable password-reset code to be sent to user
        email_verified: "true",
      };
      event.response.messageAction = "SUPPRESS";
    }
  }

  return event;
};
```

```
export { handler };
```

自定义消息 Lambda 触发器

Amazon Cognito 在发送电子邮件或电话验证消息或多重验证 (MFA) 代码前调用此触发器。您可以使用自定义消息触发器动态自定义消息。您可以在 Amazon Cognito 控制台的 Message Customizations（消息自定义）选项卡中编辑静态自定义消息。

该请求包括 `codeParameter`。这是一个字符串，用作 Amazon Cognito 传递给用户的代码的占位符。将 `codeParameter` 字符串插入消息正文中用于显示验证码的位置。Amazon Cognito 在收到此响应后，Amazon Cognito 将 `codeParameter` 字符串替换为实际验证码。

> **Note**
>
> 具有 `CustomMessage_AdminCreateUser` 触发器源的自定义消息 Lambda 函数将返回用户名和验证码。由于管理员创建的用户必须同时收到其用户名和代码，因此来自您的函数的响应必须同时包含 `request.usernameParameter` 和 `request.codeParameter`。

主题

- 自定义消息 Lambda 触发器源 (p. 151)
- 自定义消息 Lambda 触发器参数 (p. 151)
- 用于注册的自定义消息示例 (p. 153)
- 管理员创建用户的自定义消息示例 (p. 154)

自定义消息 Lambda 触发器源

triggerSource 值	事件
`CustomMessage_SignUp`	自定义消息 – 在注册后发送确认代码。
`CustomMessage_AdminCreateUser`	自定义消息 – 向新用户发送临时密码。
`CustomMessage_ResendCode`	自定义消息 – 向现有用户重新发送确认代码。
`CustomMessage_ForgotPassword`	自定义消息 – 针对“忘记密码”请求发送确认代码。
`CustomMessage_UpdateUserAttribute`	自定义消息 – 当用户的电子邮件或电话号码发生更改时，此触发器自动向用户发送验证码。不可用于其他属性。
`CustomMessage_VerifyUserAttribute`	自定义消息 – 当用户针对新的电子邮件或电话号码手动请求验证码时，此触发器向用户发送验证码。
`CustomMessage_Authentication`	自定义消息 – 在身份验证过程中发送 MFA 代码。

自定义消息 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
```

```
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        }
        "codeParameter": "####",
        "usernameParameter": "string",
        "clientMetadata": {
            "string": "string",
            . . .
        }
    },
    "response": {
        "smsMessage": "string",
        "emailMessage": "string",
        "emailSubject": "string"
    }
}
```

自定义消息请求参数

userAttributes

表示用户属性的一个或多个名称/值对。

codeParameter

一个字符串，用作自定义消息中验证码的占位符。

usernameParameter

用户名。Amazon Cognito 在管理员创建的用户发出的请求中包含此参数。

clientMetadata

一个或多个键值对，您可以将其作为自定义输入内容提供给为自定义消息触发器指定的 Lambda 函数。调用自定义消息函数的请求不包括在 AdminInitiateAuth 和 InitiateAuth API 操作的 ClientMetadata 参数中传递的数据。要将此数据传递给 Lambda 函数，您可以在以下 API 操作中使用 ClientMetadata 参数：

- AdminResetUserPassword
- AdminRespondToAuthChallenge
- AdminUpdateUserAttributes
- ForgotPassword
- GetUserAttributeVerificationCode
- ResendConfirmationCode
- SignUp
- UpdateUserAttributes

自定义消息响应参数

在响应中，指定要在发送给用户的消息中使用的自定义文本。

smsMessage

要发送给用户的自定义 SMS 消息。必须包含您在请求中收到的 `codeParameter` 值。

emailMessage

发送给用户的自定义电子邮件。您可以在 `emailMessage` 参数中使用 HTML 格式。必须包含您在请求中收到的 `codeParameter` 值作为变量 `{####}`。只有在用户池的 `EmailSendingAccount` 属性为 `DEVELOPER` 时，Amazon Cognito 才可

以使用emailMessage 参数。如果用户池的 EmailSendingAccount 属性不是 DEVELOPER 且返回了 emailMessage 参数，Amazon Cognito 会生成 400 错误代码 com.amazonaws.cognito.identity.idp.model.InvalidLambdaResponseException。当您选择使用 Amazon Simple Email Service (Amazon SES) 发送电子邮件时，用户池的 EmailSendingAccount 属性为 DEVELOPER。否则，该值为 COGNITO_DEFAULT。

emailSubject

自定义消息的主题行。只有在用户池的 EmailSendingAccount 属性为 DEVELOPER 时，您才可以使用 emailSubject 参数。如果用户池的 EmailSendingAccount 属性不是 DEVELOPER 且 Amazon Cognito 返回了 emailSubject 参数，Amazon Cognito 会生成 400 错误代码 com.amazonaws.cognito.identity.idp.model.InvalidLambdaResponseException。当您选择使用 Amazon Simple Email Service (Amazon SES) 发送电子邮件时，用户池的 EmailSendingAccount 属性为 DEVELOPER。否则，该值为 COGNITO_DEFAULT。

用于注册的自定义消息示例

当服务要求应用程序向用户发送验证码时，此示例 Lambda 函数自定义电子邮件或 SMS 消息。

Amazon Cognito 可以在多个事件中调用 Lambda 触发器：注册后、重新发送验证码时、恢复忘记的密码时或验证用户属性时。响应包括电子邮件和 SMS 消息。该消息必须包含代码参数 "####"。此参数是用户收到的验证码的占位符。

电子邮件的最大长度为 20000 个 UTF-8 字符。此长度包括验证码。您可以在这些电子邮件中使用 HTML 标签。

SMS 消息的最大长度为 140 个 UTF-8 个字符。此长度包括验证码。

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 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
  "version": 1,
  "triggerSource": "CustomMessage_SignUp/CustomMessage_ResendCode/
CustomMessage_ForgotPassword/CustomMessage_VerifyUserAttribute",
  "region": "<region>",
  "userPoolId": "<userPoolId>",
  "userName": "<userName>",
  "callerContext": {
      "awsSdk": "<calling aws sdk with version>",
      "clientId": "<apps client id>",
```

```
      ...
  },
  "request": {
      "userAttributes": {
          "phone_number_verified": false,
          "email_verified": true,
           ...
      },
      "codeParameter": "####"
  },
  "response": {
      "smsMessage": "<custom message to be sent in the message with code parameter>"
      "emailMessage": "<custom message to be sent in the message with code parameter>"
      "emailSubject": "<custom email subject>"
  }
}
```

管理员创建用户的自定义消息示例

具有 CustomMessage_AdminCreateUser 触发器的自定义消息 Lambda 函数将返回用户名和验证码。出于这个原因，请在消息正文中包含 request.usernameParameter 和 request.codeParameter。

代码参数值 #### 是临时密码的占位符，“username”是用户收到的用户名的占位符。

电子邮件的最大长度为 20000 个 UTF-8 字符。此长度包括验证码。您可以在这些电子邮件中使用 HTML 标签。SMS 消息的最大长度为 140 个 UTF-8 个字符。此长度包括验证码。

响应包括电子邮件和 SMS 消息。

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 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

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>"
    }
}
```

自定义发件人 Lambda 触发器

Amazon Cognito 用户池提供 Lambda 触发器 `CustomEmailSender` 和 `CustomSMSSender`，用于激活第三方电子邮件和 SMS 通知。您可以选择 SMS 和电子邮件提供商，通过 Lambda 函数代码向用户发送通知。当 Amazon Cognito 需要向用户发送确认码、验证码或临时密码等通知时，这些事件会触发您配置的 Lambda 函数。Amazon Cognito 将代码和临时密码（密钥）发送到您激活的 Lambda 函数。Amazon Cognito 使用 AWS KMS 客户托管密钥和 AWS Encryption SDK加密这些密钥。AWS Encryption SDK 是一个客户端加密库，可帮助您加密和解密通用数据。

> **Note**
>
> 要配置用户池以使用这些 Lambda 触发器，您可以使用 AWS CLI 或开发工具包。无法从 Amazon Cognito 控制台进行这些配置。

CustomEmailSender (p. 156)

Amazon Cognito 调用此触发器向用户发送电子邮件通知。

CustomSMSSender (p. 160)

Amazon Cognito 调用此触发器向用户发送 SMS 通知。

资源

以下资源可以帮助您使用 `CustomEmailSender` 和 `CustomSMSSender` 触发器。

AWS KMS

AWS KMS 是用于创建和控制 AWS KMS 钥匙的托管服务。这些密钥加密您的数据。有关更多信息，请参阅什么是 AWS Key Management Service？。

KMS 密钥

KMS 密钥是加密密钥的逻辑表示形式。KMS 密钥包含元数据，如密钥 ID、创建日期、描述和密钥状态。KMS 密钥还包含用于加密和解密数据的密钥材料。有关更多信息，请参阅 AWS KMS 密钥。

对称 KMS 密钥

对称 KMS 密钥是一个 256 位加密密钥，它不会退出 AWS KMS 而不加密。要使用对称 KMS 密钥，您必须调用 AWS KMS。Amazon Cognito 使用对称密钥。加密和解密使用同一密钥。有关更多信息，请参阅对称 KMS 密钥。

自定义电子邮件发件人 Lambda 触发器

Amazon Cognito 调用自定义电子邮件发件人触发器，以便第三方提供商可以从您的 AWS Lambda 函数代码向您的用户发送电子邮件通知。Amazon Cognito 将电子邮件消息事件作为请求发送到 Lambda 函数。然后，您的函数的自定义代码必须处理和传递消息。

> Note
>
> 目前，您无法在 Amazon Cognito 控制台中分配自定义电子邮件发件人触发器。您可以在 `CreateUserPool` 或 `UpdateUserPool` API 请求中使用 `LambdaConfig` 参数分配触发器。

要设置此触发器，请执行以下步骤：

1. 在 AWS Key Management Service (AWS KMS) 中创建加密密钥。Amazon Cognito 生成密钥（临时密码和授权代码），然后使用此 KMS 密钥对其进行加密。然后，您可以在 Lambda 函数中使用 AWS Encryption SDK 来解密代码并以明文形式将其发送给用户。
2. 创建一个您要分配为自定义发件人触发器的 Lambda 函数。向您分配给 Lambda 函数的执行角色授予对您的 KMS 密钥的 `kms:Decrypt` 权限。
3. 授予 Amazon Cognito 服务主体 `cognito-idp.amazonaws.com` 调用 Lambda 函数以及为 KMS 密钥创建授权的访问权限。
4. 编写 Lambda 函数代码，将您的消息定向到自定义传递方法或第三方提供商。
5. 更新用户池，使其使用自定义发件人 Lambda 触发器。

> Important
>
> 当您为用户池配置自定义电子邮件发件人或自定义 SMS 发件人功能时，请配置对称 AWS KMS key 以提高安全性。Amazon Cognito 使用您配置的 KMS 密钥加密代码或临时密码。Amazon Cognito 将 base64 编码的密文发送到您的 Lambda 函数。有关更多信息，请参阅对称 KMS 密钥。

自定义电子邮件发件人 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "type": "customEmailSenderRequestV1",
        "code": "string",
        "clientMetadata": {
            "string": "string",
             . . .
            },
        "userAttributes": {
            "string": "string",
            . . .
         }
}
```

自定义电子邮件发件人请求参数

type

请求版本。对于自定义电子邮件发件人事件，此字符串的值始终为 `customEmailSenderRequestV1`。

代码

您的函数可以解密并发送给您的用户的加密代码。

clientMetadata

一个或多个键值对，您可以将它们作为自定义输入提供给自定义电子邮件发件人 Lambda 函数触发器。要将此数据传递给 Lambda 函数，您可以在 AdminRespondToAuthChallenge 和 RespondToAuthChallenge API 操作中使用 ClientMetadata 参数。在传递到身份验证后函数的请求中，Amazon Cognito 不包括 AdminInitiateAuth 和 InitiateAuth API 操作的 ClientMetadata 参数中传递的数据。

userAttributes

表示用户属性的一个或多个键值对。

自定义电子邮件发件人响应参数

Amazon Cognito 不需要自定义电子邮件发件人响应中有任何其他返回信息。您的函数可以使用 API 操作来查询和修改资源，或者将事件元数据记录到外部系统。

激活自定义电子邮件发件人 Lambda 触发器

要设置使用自定义逻辑为您的用户池发送电子邮件消息的自定义电子邮件发件人触发器，请按如下方式激活触发器。以下步骤会将自定义电子邮件触发器和/或自定义 SMS 触发器分配给您的用户群体。在您添加自定义电子邮件发件人触发器后，Amazon Cognito 将始终向您的 Lambda 函数发送用户属性（包括电子邮件地址）和一次性代码，而本来会使用 Amazon Simple Email Service 发送电子邮件。

Important

Amazon Cognito HTML 会转义用户临时密码中的保留字符，例如 <（`<`）和 `>`（>）等。这些字符可能出现在 Amazon Cognito 发送到您的自定义电子邮件发件人函数的临时密码中，但不会出现在临时验证码中。要发送临时密码，您的 Lambda 函数在解密密码之后必须取消对这些字符的转义，然后再将消息发送给您的用户。

1. 在 AWS KMS 中创建加密密钥。此密钥加密 Amazon Cognito 生成的临时密码和授权代码。然后，您可以在自定义发件人 Lambda 函数中解密这些密钥，将其以明文形式发送给用户。
2. 授予 Amazon Cognito 服务主体 `cognito-idp.amazonaws.com` 使用 KMS 密钥加密代码的访问权限。

   将以下基于资源的策略应用于您的 KMS 密钥。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cognito-idp.amazonaws.com"
        },
        "Action": "kms:CreateGrant",
        "Resource": "arn:aws:kms:us-
west-2:111222333444:key/1example-2222-3333-4444-999example",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "111222333444"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:cognito-idp:us-
west-2:111222333444:userpool/us-east-1_EXAMPLE"
            }
```

```
        }
    }]
}
```

3. 为自定义发件人触发器创建 Lambda 函数。Amazon Cognito 使用 AWS Encryption SDK 加密密钥、临时密码和授权用户 API 请求的代码。
   - 为您的 Lambda 函数分配一个 IAM 角色，该角色至少具有对您的 KMS 密钥的 `kms:Decrypt` 权限。
4. 授予 Amazon Cognito 服务主体 `cognito-idp.amazonaws.com` 访问权限，以调用 Lambda 函数。

   以下 AWS CLI 命令授予 Amazon Cognito 调用 Lambda 函数的权限。

```
       aws lambda add-permission --function-name lambda_arn --statement-id
 "CognitoLambdaInvokeAccess" --action lambda:InvokeFunction --principal cognito-
idp.amazonaws.com
```

5. 编写 Lambda 函数代码以发送消息。Amazon Cognito 使用 AWS Encryption SDK 加密密钥，然后 Amazon Cognito 会将密钥发送给自定义发件人 Lambda 函数。在您的函数中，解密密钥并处理任何相关的元数据。然后将代码、您自己的自定义消息和目标电话号码发送到传送消息的自定义 API。
6. 将 AWS Encryption SDK 添加到 Lambda 函数。有关更多信息，请参阅 AWS Encryption SDK 编程语言。要更新 Lambda 包，请完成以下步骤：
   a. 在 AWS Management Console 中将您的 Lambda 函数作为.zip 文件导出。
   b. 打开您的函数并添加 AWS Encryption SDK。有关更多信息和下载链接，请参阅 AWS Encryption SDK Developer Guide（《Crypto SDK 开发人员指南》）中的 AWS Encryption SDK 编程语言。
   c. 压缩您的函数及 SDK 依赖项，然后将函数上传到 Lambda。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的将 Lambda 函数部署为 .zip 文件归档。
7. 更新用户群体，添加自定义发件人 Lambda 触发器。在 `UpdateUserPool` API 请求中包含 `CustomSMSSender` 或 `CustomEmailSender` 参数。`UpdateUserPool` API 操作需要更新用户群体的所有参数和 您要更改的参数。如果您没有提供所有相关参数，Amazon Cognito 会将任何缺失参数的值设置为其原定设置。如以下示例所示，包括您想要添加到或保留在用户群体中的所有 Lambda 函数的条目。有关更多信息，请参阅更新用户群体配置 (p. 53)。

```
   #Send this parameter in an 'aws cognito-idp update-user-pool' CLI command,
including any existing
   #user pool configurations.

     --lambda-config "PreSignUp=lambda-arn, \
                      CustomSMSSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
                      CustomEmailSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
                      KMSKeyID=key-id"
```

要使用 `update-user-pool` AWS CLI 删除自定义发件人 Lambda 触发器，请在 `--lambda-config` 中省略 `CustomSMSSender` 或 `CustomEmailSender` 参数，并包括要与用户群体结合使用的所有其他触发器。

要使用 `UpdateUserPool` API 请求删除自定义发件人 Lambda 触发器，请在包含其余用户群体配置的请求正文中省略 `CustomSMSSender` 或 `CustomEmailSender` 参数。

代码示例

以下 Node.js 示例在您的自定义电子邮件发件人 Lambda 函数中处理电子邮件事件。此示例假设您的函数定义了两个环境变量。

KEY_ALIAS

要用于加密和解密用户代码的 KMS 密钥的别名。

KEY_ARN

要用于加密和解密用户代码的 KMS 密钥的 Amazon 资源名称 (ARN)。

```
const AWS = require('aws-sdk');
const b64 = require('base64-js');
const encryptionSdk = require('@aws-crypto/client-node');
//Configure the encryption SDK client with the KMS key from the environment variables.
const { encrypt, decrypt } =
 encryptionSdk.buildClient(encryptionSdk.CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT);
const generatorKeyId = process.env.KEY_ALIAS;
const keyIds = [ process.env.KEY_ARN ];
const keyring = new encryptionSdk.KmsKeyringNode({ generatorKeyId, keyIds })
exports.handler = async (event) => {
 //Decrypt the secret code using encryption SDK.
 let plainTextCode;
 if(event.request.code){
  const { plaintext, messageHeader } = await decrypt(keyring,
 b64.toByteArray(event.request.code));
  plainTextCode = plaintext
 }
 //PlainTextCode now contains the decrypted secret.
 if(event.triggerSource == 'CustomEmailSender_SignUp'){
  //Send an email message to your user via a custom provider.
  //Include the temporary password in the message.
 }
 else if(event.triggerSource == 'CustomEmailSender_ResendCode'){
 }
 else if(event.triggerSource == 'CustomEmailSender_ForgotPassword'){
 }
 else if(event.triggerSource == 'CustomEmailSender_UpdateUserAttribute'){
 }
 else if(event.triggerSource == 'CustomEmailSender_VerifyUserAttribute'){
 }
 else if(event.triggerSource == 'CustomEmailSender_AdminCreateUser'){
 }
 else if(event.triggerSource == 'CustomEmailSender_AccountTakeOverNotification'){
 }
 return;
};
```

自定义电子邮件发件人 Lambda 触发器源

下表显示了 Lambda 代码中自定义电子邮件触发器源的触发事件。

TriggerSource value	事件
CustomEmailSender_SignUp	用户注册，Amazon Cognito 随即发送欢迎消息。
CustomEmailSender_ForgotPassword	用户请求代码以重置其密码。
CustomEmailSender_ResendCode	用户请求更换代码以重置其密码。
CustomEmailSender_UpdateUserAttribute	用户更新电子邮件地址或电话号码属性，而 Amazon Cognito 将发送代码用于验证该属性。
CustomEmailSender_VerifyUserAttribute	用户创建新电子邮件地址或电话号码属性，而 Amazon Cognito 将发送代码用于验证该属性。

TriggerSource value	事件
CustomEmailSender_AdminCreateUser	您在用户池中创建新用户，而 Amazon Cognito 向其发送临时密码。
CustomEmailSender_AccountTakeOverNotification	Amazon Cognito 检测到接管用户账户的尝试并向用户发送通知。

自定义 SMS 发件人 Lambda 触发器

Amazon Cognito 调用自定义 SMS 发件人触发器，以便第三方提供商可以从您的 AWS Lambda 函数代码向您的用户发送 SMS 通知。Amazon Cognito 将 SMS 消息事件作为请求发送到 Lambda 函数。然后，您的函数的自定义代码必须处理和传递消息。

> Note
>
> 目前，您无法在 Amazon Cognito 控制台中分配自定义 SMS 发件人触发器。您可以在 `CreateUserPool` 或 `UpdateUserPool` API 请求中使用 `LambdaConfig` 参数分配触发器。

要设置此触发器，请执行以下步骤：

1. 在 AWS Key Management Service (AWS KMS) 中创建加密密钥。Amazon Cognito 生成密钥（临时密码和授权代码），然后使用此 KMS 密钥对其进行加密。然后，您可以在 Lambda 函数中使用 AWS Encryption SDK 来解密代码并以明文形式将其发送给用户。
2. 创建一个您要分配为自定义 SMS 发件人触发器的 Lambda 函数。向您分配给 Lambda 函数的执行角色授予对您的 KMS 密钥的 `kms:Decrypt` 权限。
3. 授予 Amazon Cognito 服务主体 `cognito-idp.amazonaws.com` 调用 Lambda 函数以及为 KMS 密钥创建授权的访问权限。
4. 编写 Lambda 函数代码，将您的 SMS 消息定向到自定义传递方法或第三方提供商。
5. 更新用户池，使其使用自定义发件人 Lambda 触发器。

> Important
>
> 当您配置自定义电子邮件或自定义 SMS 发件人功能时，创建新的对称 AWS KMS key。Amazon Cognito 使用您配置的 KMS 密钥加密代码或临时密码。Amazon Cognito 将 base64 编码的加密文字发送到您的 Lambda 函数。有关更多信息，请参阅对称 KMS 密钥。

自定义 SMS 发件人 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "type": "customSMSSenderRequestV1",
        "code": "string",
        "clientMetadata": {
            "string": "string",
             . . .
            },
        "userAttributes": {
            "string": "string",
            . . .
         }
}
```

自定义 SMS 发件人请求参数

type

请求版本。对于自定义 SMS 发件人事件，此字符串的值始终为 `customSMSSenderRequestV1`。

代码

您的函数可以解密并发送给您的用户的加密代码。

clientMetadata

一个或多个键值对，您可以将它们作为自定义输入提供给自定义 SMS 发件人 Lambda 函数触发器。要将此数据传递给 Lambda 函数，您可以在 AdminRespondToAuthChallenge 和 RespondToAuthChallenge API 操作中使用 ClientMetadata 参数。在传递到身份验证后函数的请求中，Amazon Cognito 不包括 AdminInitiateAuth 和 InitiateAuth API 操作的 ClientMetadata 参数中传递的数据。

userAttributes

表示用户属性的一个或多个键值对。

自定义 SMS 发件人响应参数

Amazon Cognito 不需要响应中任何额外的返回信息。您的函数可以使用 API 操作来查询和修改资源，或者将事件元数据记录到外部系统。

激活自定义 SMS 发件人 Lambda 触发器

您可以设置自定义 SMS 发件人触发器，使用自定义逻辑为您的用户群体发送 SMS 消息。以下步骤会将自定义 SMS 触发器和/或自定义电子邮件触发器分配给您的用户群体。添加自定义 SMS 发件人触发器后，Amazon Cognito 始终向您的 Lambda 函数发送用户属性（包括电话号码）和一次性代码，而不是采用默认使用 Amazon Simple Notification Service 发送 SMS 消息的行为。

Important

Amazon Cognito HTML 会转义用户临时密码中的保留字符，例如 <（`<`）和 >（>）等。这些字符可能出现在 Amazon Cognito 发送到您的自定义电子邮件发件人函数的临时密码中，但不会出现在临时验证码中。要发送临时密码，您的 Lambda 函数在解密密码之后必须取消对这些字符的转义，然后再将消息发送给您的用户。

1. 在 AWS KMS 中创建加密密钥。此密钥加密 Amazon Cognito 生成的临时密码和授权代码。然后，您可以在自定义发件人 Lambda 函数中解密这些密钥，将其以明文形式发送给用户。
2. 授予 Amazon Cognito 服务主体 `cognito-idp.amazonaws.com` 使用 KMS 密钥加密代码的访问权限。

   将以下基于资源的策略应用于您的 KMS 密钥。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cognito-idp.amazonaws.com"
        },
        "Action": "kms:CreateGrant",
        "Resource": "arn:aws:kms:us-
west-2:111222333444:key/1example-2222-3333-4444-999example",
        "Condition": {
            "StringEquals": {
```

```
                "aws:SourceAccount": "111222333444"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:cognito-idp:us-
west-2:111222333444:userpool/us-east-1_EXAMPLE"
            }
        }
    }]
}
```

3. 为自定义发件人触发器创建 Lambda 函数。Amazon Cognito 使用 AWS Encryption SDK 加密密钥、临时密码和授权用户 API 请求的代码。

   - 为您的 Lambda 函数分配一个 IAM 角色，该角色至少具有对您的 KMS 密钥的 `kms:Decrypt` 权限。

4. 授予 Amazon Cognito 服务主体 `cognito-idp.amazonaws.com` 访问权限，以调用 Lambda 函数。

   以下 AWS CLI 命令授予 Amazon Cognito 调用 Lambda 函数的权限。

```
       aws lambda add-permission --function-name lambda_arn --statement-id
 "CognitoLambdaInvokeAccess" --action lambda:InvokeFunction --principal cognito-
idp.amazonaws.com
```

5. 编写 Lambda 函数代码以发送消息。Amazon Cognito 使用 AWS Encryption SDK 加密密钥，然后 Amazon Cognito 会将密钥发送给自定义发件人 Lambda 函数。在您的函数中，解密密钥并处理任何相关的元数据。然后将代码、您自己的自定义消息和目标电话号码发送到传送消息的自定义 API。
6. 将 AWS Encryption SDK 添加到 Lambda 函数。有关更多信息，请参阅 AWS Encryption SDK 编程语言。要更新 Lambda 包，请完成以下步骤：

   a. 在 AWS Management Console 中将您的 Lambda 函数作为.zip 文件导出。
   b. 打开您的函数并添加 AWS Encryption SDK。有关更多信息和下载链接，请参阅 AWS Encryption SDK Developer Guide（《Crypto SDK 开发人员指南》）中的 AWS Encryption SDK 编程语言。
   c. 压缩您的函数及 SDK 依赖项，然后将函数上传到 Lambda。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的将 Lambda 函数部署为 .zip 文件归档。

7. 更新用户群体，添加自定义发件人 Lambda 触发器。在 `UpdateUserPool` API 请求中包含 `CustomSMSSender` 或 `CustomEmailSender` 参数。`UpdateUserPool` API 操作需要更新用户群体的所有参数和 您要更改的参数。如果您没有提供所有相关参数，Amazon Cognito 会将任何缺失参数的值设置为其原定设置。如以下示例所示，包括您想要添加到或保留在用户群体中的所有 Lambda 函数的条目。有关更多信息，请参阅更新用户群体配置 (p. 53)。

```
   #Send this parameter in an 'aws cognito-idp update-user-pool' CLI command,
 including any existing
   #user pool configurations.

     --lambda-config "PreSignUp=lambda-arn, \
                      CustomSMSSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
                      CustomEmailSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
                      KMSKeyID=key-id"
```

要使用 `update-user-pool` AWS CLI 删除自定义发件人 Lambda 触发器，请在 `--lambda-config` 中省略 `CustomSMSSender` 或 `CustomEmailSender` 参数，并包括要与用户群体结合使用的所有其他触发器。

要使用 `UpdateUserPool` API 请求删除自定义发件人 Lambda 触发器，请在包含其余用户群体配置的请求正文中省略 `CustomSMSSender` 或 `CustomEmailSender` 参数。

代码示例

以下 Node.js 示例在您的自定义 SMS 发件人 Lambda 函数中处理 SMS 消息事件。此示例假设您的函数定义了两个环境变量。

KEY_ALIAS

要用于加密和解密用户代码的 KMS 密钥的别名。

KEY_ARN

要用于加密和解密用户代码的 KMS 密钥的 Amazon 资源名称 (ARN)。

```
const AWS = require('aws-sdk');
const b64 = require('base64-js');
const encryptionSdk = require('@aws-crypto/client-node');
//Configure the encryption SDK client with the KMS key from the environment variables.
const { encrypt, decrypt } =
 encryptionSdk.buildClient(encryptionSdk.CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT);
const generatorKeyId = process.env.KEY_ALIAS;
const keyIds = [ process.env.KEY_ARN ];
const keyring = new encryptionSdk.KmsKeyringNode({ generatorKeyId, keyIds })
exports.handler = async (event) => {
 //Decrypt the secret code using encryption SDK.
 let plainTextCode;
 if(event.request.code){
  const { plaintext, messageHeader } = await decrypt(keyring,
 b64.toByteArray(event.request.code));
  plainTextCode = plaintext
 }
 //PlainTextCode now contains the decrypted secret.
 if(event.triggerSource == 'CustomSMSSender_SignUp'){
  //Send an SMS message to your user via a custom provider.
  //Include the temporary password in the message.
 }
 else if(event.triggerSource == 'CustomSMSSender_ResendCode'){
 }
 else if(event.triggerSource == 'CustomSMSSender_ForgotPassword'){
 }
 else if(event.triggerSource == 'CustomSMSSender_UpdateUserAttribute'){
 }
 else if(event.triggerSource == 'CustomSMSSender_VerifyUserAttribute'){
 }
 else if(event.triggerSource == 'CustomSMSSender_AdminCreateUser'){
 }
 else if(event.triggerSource == 'CustomSMSSender_AccountTakeOverNotification'){
 }
 return;
};
```

主题

- 使用自定义 SMS 发件人函数评估 SMS 消息功能 (p. 163)
- 自定义 SMS 发件人 Lambda 触发器源 (p. 159)

使用自定义 SMS 发件人函数评估 SMS 消息功能

自定义 SMS 发件人 Lambda 函数接受您的用户池将发送的 SMS 消息，并且该函数根据您的自定义逻辑传送内容。Amazon Cognito 将 自定义 SMS 发件人 Lambda 触发器参数 (p. 160) 发送到您的函数。您的函数可以用这些信息做您想做的事。例如，您可以将代码发送到 Amazon Simple Notification Service (Amazon SNS) 主题。Amazon SNS 主题订阅者可以是 SMS 消息、HTTPS 终端节点或电子邮件地址。

要使用自定义 SMS 发件人 Lambda 函数为 Amazon Cognito SMS 消息传递创建测试环境，请参阅 GitHub 上 aws-samples 库中的 amazon-cognito-user-pool-development-and-testing-with-sms-redirected-to-email。存储库包含 AWS CloudFormation 可以创建新的用户池或使用已有用户池的模板。这些模板创建 Lambda 函数和 Amazon SNS 主题。模板分配为自定义 SMS 发件人触发器的 Lambda 函数将您的 SMS 消息重定向到 Amazon SNS 主题的订阅者。

当您将此解决方案部署到用户池时，对于 Amazon Cognito 通常通过 SMS 消息发送的所有消息，Lambda 函数将改为发送到中央电子邮件地址。使用此解决方案自定义和预览 SMS 消息，并测试促使 Amazon Cognito 发送 SMS 消息的用户池事件。完成测试后，回滚 CloudFormation 堆栈，或从您的用户池中删除自定义 SMS 发件人功能分配。

Important

请勿使用 amazon-cognito-user-pool-development-and-testing-with-sms-redirected-to-email 中的模板来构建生产环境。解决方案中的自定义 SMS 发件人 Lambda 函数模拟 SMS 消息，但 Lambda 函数将它们全部发送到一个中央电子邮件地址。在生产 Amazon Cognito 用户池中发送 SMS 消息之前，您必须完成 Amazon Cognito 用户池的短信设置 (p. 199)中显示的要求。

自定义 SMS 发件人 Lambda 触发器源

下表显示了 Lambda 代码中自定义 SMS 触发器源的触发事件。

TriggerSource value	事件
CustomSMSSender_SignUp	用户注册，Amazon Cognito 随即发送欢迎消息。
CustomSMSSender_ForgotPassword	用户请求代码以重置其密码。
CustomSMSSender_ResendCode	用户请求更换代码以重置其密码。
CustomSMSSender_VerifyUserAttribute	用户创建新电子邮件地址或电话号码属性，而 Amazon Cognito 将发送代码用于验证该属性。
CustomSMSSender_UpdateUserAttribute	用户更新电子邮件地址或电话号码属性，而 Amazon Cognito 将发送代码用于验证该属性。
CustomSMSSender_Authentication	配置了 SMS 多重验证 (MFA) 的用户将登录。
CustomSMSSender_AdminCreateUser	您在用户池中创建新用户，而 Amazon Cognito 向其发送临时密码。

将 Amazon Pinpoint 分析与 Amazon Cognito 用户池结合使用

Amazon Pinpoint 用户群体与 Amazon Pinpoint 集成，为 Amazon Cognito 用户群体提供分析并丰富 Amazon Pinpoint 活动的用户数据。Amazon Pinpoint 提供分析和有针对性的市场活动，以使用推送通知推动用户与移动应用程序的交互。通过 Amazon Cognito 用户池中的 Amazon Pinpoint 分析支持，您可以在 Amazon Pinpoint 控制台中跟踪用户池注册、登录、失败的身份验证、日活跃用户（DAU）和月活跃用户（MAU）。您可以深入查看不同日期范围或属性的数据，例如设备平台、设备区域设置和应用程序版本。

您还可以为应用程序设置自定义属性。然后可以使用这些属性在 Amazon Pinpoint 上对用户进行细分，并向他们发送有针对性的推送通知。如果您在 Amazon Cognito 控制台的 Analytics（分析）选项卡中选择 Share user attribute data with Amazon Pinpoint（与 Amazon Pinpoint 分享用户属性数据），Amazon Pinpoint 将会为用户电子邮件地址和电话号码创建其他端点。

当您使用 Amazon Cognito 控制台在用户群体中激活 Amazon Pinpoint 分析时，您还会创建一个服务相关角色，Amazon Cognito 在向 Amazon Pinpoint 发出针对用户群体的 API 请求时代入

该角色。添加分析配置的 IAM 主体必须具有 CreateServiceLinkedRole 权限。该服务相关角色为 AWSServiceRoleForAmazonCognitoIdp。有关更多信息，请参阅对 Amazon Cognito 使用服务相关角色 (p. 504)。

当您在 Amazon Cognito API 中向应用程序客户端应用 `AnalyticsConfiguration` 时，您可以为 Amazon Pinpoint 分配自定义 IAM 角色和外部 ID 来代入该角色。此角色必须信任 `cognito-idp` 服务主体，如果角色信任策略需要外部 ID，则它必须与您的 `AnalyticsConfiguration` 相匹配。您必须针对 Amazon Pinpoint 项目向该角色授予 `cognito-idp:Describe*` 权限和以下权限。

- `mobiletargeting:UpdateEndpoint`
- `mobiletargeting:PutEvents`

Amazon Cognito 和 Amazon Pinpoint 区域可用性

下表显示了符合以下条件之一的 Amazon Cognito 与 Amazon Pinpoint 之间的 AWS 区域 映射。

- 您只能在美国东部（弗吉尼亚州北部）（us-east-1）区域使用 Amazon Pinpoint 项目。
- 您可以在相同的区域或者在美国东部（弗吉尼亚州北部）（us-east-1）区域使用 Amazon Pinpoint 项目

默认情况下，Amazon Cognito 只能向位于相同 AWS 区域 中的 Amazon Pinpoint 项目发送分析。此规则的例外情况是下表中的区域，以及 Amazon Pinpoint 不可用的区域。

> **Note**
>
> Amazon Pinpoint 已在欧洲地区（米兰）和中东（巴林）推出。这些地区的 Amazon Cognito 用户群体不支持分析。

表中显示了您构建 Amazon Cognito 用户群体的区域与 Amazon Pinpoint 中对应区域的关系。您必须在可用区域中配置 Amazon Pinpoint 项目，才能将其与 Amazon Cognito 集成。

Amazon Cognito 用户群体区域	Amazon Pinpoint 项目所在区域
ap-northeast-1	us-east-1
ap-northeast-2	us-east-1
ap-south-1	us-east-1、ap-sou1
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

区域映射示例

- 如果您在 ap-northest-1 中创建用户群体，则可以在 us-east-1 中创建您的 Amazon Pinpoint 项目。
- 如果您在 ap-south-1 中创建用户群体，则可以在 us-east-1 或 ap-south-1 中创建您的 Amazon Pinpoint 项目。

> **Note**
>
> 对于上表以外的所有 AWS 区域，Amazon Cognito 只能使用与您的用户群体位于同一区域的 Amazon Pinpoint 项目。如果 Amazon Pinpoint 在您构建用户群体的区域中不可用且未在表中列出，那么 Amazon Cognito 在该区域不支持 Amazon Pinpoint 分析。有关详细的 AWS 区域 信息，请参阅 Amazon Pinpoint 端点和配额。

指定 Amazon Pinpoint 分析设置（AWS Management Console）

您可以配置 Amazon Cognito 用户群体以向 Amazon Pinpoint 发送分析数据。Amazon Cognito 仅为本地用户将分析数据发送到 Amazon Pinpoint。将您的用户群体配置为与 Amazon Pinpoint 项目关联后，您必须在 API 请求中包含 `AnalyticsMetadata`。有关更多信息，请参阅将您的应用程序与 Amazon Pinpoint 集成 (p. 167)。

指定分析设置

1. 转到 Amazon Cognito 控制台。系统可能会提示您输入 AWS 凭证。
2. 选择 User Pools（用户群体）并从列表中选择一个现有的用户群体。
3. 选择 App integration（应用程序集成）选项卡。
4. 在 App clients and analytics（应用程序客户端和分析）下，从列表中选择现有的 App client name（应用程序客户端名称）。
5. 在 Pinpoint analytics（Pinpoint 分析）下，选择 Enable（启用）。
6. 选择 Pinpoint Region（Pinpoint 区域）。
7. 选择 Amazon Pinpoint project（Amazon Pinpoint 项目）或者选择 Create Amazon Pinpoint project（创建 Amazon Pinpoint 项目）。

   > **Note**
   >
   > Amazon Pinpoint 项目 ID 是 Amazon Pinpoint 项目特有的由 32 个字符组成的字符串。它在 Amazon Pinpoint 控制台中列出。
   > 您可以将多个 Amazon Cognito 应用程序映射到单个 Amazon Pinpoint 项目。但是，每个 Amazon Cognito 应用程序只能映射到一个 Amazon Pinpoint 项目。
   > 在 Amazon Pinpoint 中，每个项目都应该是单个应用程序。例如，如果游戏开发员有两款游戏，每款游戏应该是单独的 Amazon Pinpoint 项目，即使这两款游戏使用同一 Amazon Cognito 用户池。有关 Amazon Pinpoint 项目的更多信息，请参阅在 Amazon Pinpoint 中创建项目。
8. 在 User data sharing（用户数据共享）下，如果您希望 Amazon Cognito 发送电子邮件地址和电话号码到 Amazon Pinpoint 并为用户创建额外的端点，则选择 Share user data with Amazon Pinpoint（与 Amazon Pinpoint 共享用户数据）。用户验证其电子邮件地址和电话号码后，Amazon Cognito 只在它们可用于用户账户时才会与 Amazon Pinpoint 共享。

   > **Note**
   >
   > 端点唯一地标识可以使用 Amazon Pinpoint 向其发送推送通知的用户设备。有关端点的更多信息，请参阅《Amazon Pinpoint 开发人员指南》中的添加端点。
9. 选择 Save changes（保存更改）。

指定 Amazon Pinpoint 分析设置（AWS CLI 和 AWS API）

使用以下命令为您的用户池指定 Amazon Pinpoint 分析设置。

在创建应用程序时为用户池的现有客户端应用程序指定分析设置

- AWS CLI: `aws cognito-idp create-user-pool-client`
- AWS API：CreateUserPoolClient

为用户池的现有客户端应用程序更新分析设置

- AWS CLI: `aws cognito-idp update-user-pool-client`
- AWS API：UpdateUserPoolClient

> Note
>
> 在您使用`ApplicationArn`时，Amazon Cognito 支持区域内集成

将您的应用程序与 Amazon Pinpoint 集成

您可以在用户群体 API 中针对 Amazon Cognito 本地用户将分析元数据发布到 Amazon Pinpoint。

本地用户

注册了账户或在您的用户群体中创建的用户，而不是通过第三方身份提供者 (IdP) 登录的用户。

用户群体 API

您可以使用具有自定义用户界面 (UI) 的应用程序，将其与 AWS SDK 集成的操作。您无法为通过托管 UI 登录的联合身份用户或本地用户传递分析元数据。有关用户群体 API 操作的列表，请参阅 Amazon Cognito API 参考。

在您将用户群体配置为发布到活动后，Amazon Cognito 会将以下 API 操作的元数据传递给 Amazon Pinpoint。

- `AdminInitiateAuth`
- `AdminRespondToAuthChallenge`
- `ConfirmForgotPassword`
- `ConfirmSignUp`
- `ForgotPassword`
- `InitiateAuth`
- `ResendConfirmationCode`
- `RespondToAuthChallenge`
- `SignUp`

要将有关用户会话的元数据传递到 Amazon Pinpoint 活动，请在 API 请求的 `AnalyticsMetadata` 参数中包含 `AnalyticsEndpointId` 值。有关 JavaScript 示例，请参阅 AWS 知识中心 中的为什么 Amazon Cognito 用户群体分析未显示在 Amazon Pinpoint 控制面板上？。

管理用户池中的用户

创建用户群体后，您可以创建、确认和管理用户账户。借助 Amazon Cognito 用户池组，您可以通过将 IAM 角色映射到组来管理您的用户及其对资源的访问。

利用用户迁移 Lambda 触发器，您可以将用户导入用户池。此方法使用户能够在首次登录您的用户池时从您的现有用户目录无缝迁移到用户池。

主题

- 注册并确认用户账户 (p. 168)
- 以管理员身份创建用户账户 (p. 175)
- 向用户池添加组 (p. 178)
- 管理和搜索用户账户 (p. 180)
- 恢复用户账户 (p. 183)
- 将用户导入一个用户池 (p. 183)

注册并确认用户账户

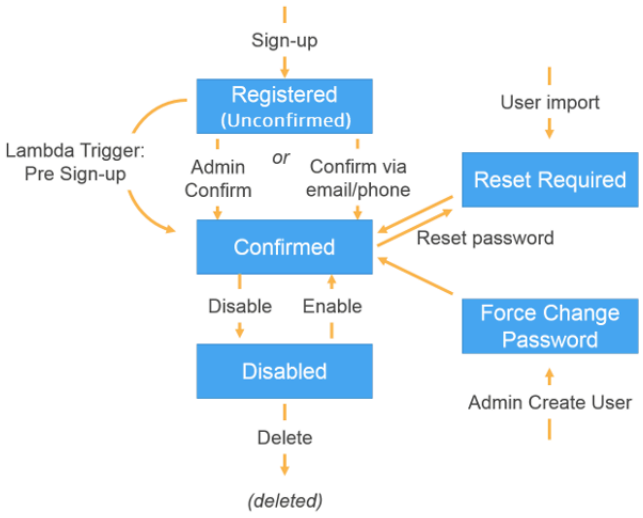
可通过以下任一方法将用户账户添加到您的用户池中：

- 用户在您用户池的客户端应用程序中进行注册。这可以是移动应用程序或 Web 应用程序。
- 您可以将用户账户导入到用户池中。有关更多信息，请参阅通过 CSV 文件将用户导入用户池中 (p. 184)。
- 您可以在用户池中创建用户账户并邀请用户登录。有关更多信息，请参阅以管理员身份创建用户账户 (p. 175)。

自行注册的用户必须得到确认才可登录。导入和创建的用户已经过确认，但他们必须在首次登录时创建自己的密码。以下部分将介绍确认过程以及电子邮件和电话验证。

用户账户确认概览

下图阐明了确认过程：



用户账户可以处于以下任一状态：

已注册（未确认）

用户已成功注册，但在用户账户得到确认之前无法登录。在此状态下，用户已启用，但未得到确认。

自行注册的新用户由此状态开始。

已确认

用户账户已确认，用户可以登录。当用户输入代码或点击电子邮件链接以确认其用户账户时，系统将自动验证电子邮件或电话号码。代码或链接的有效期为 24 小时。

如果管理员或预注册 Lambda 触发器确认了用户账户，则可能没有与该账户关联的经验证的电子邮件或电话号码。

需要重置密码

用户账户已确认，但用户必须请求代码并重置其密码，然后才可以登录。

由管理员或开发人员导入的用户账户以此状态开始。

强制更改密码

用户账户已确认，用户可以使用临时密码进行登录。但在首次登录时，用户必须将其密码更改为新值，然后才能执行任何其他操作。

由管理员或开发人员创建的用户账户以此状态开始。

已禁用

在可以删除用户账户之前，必须禁用该用户的登录访问权限。

在注册时验证联系人信息

当新用户注册您的应用程序时，您可能希望他们提供至少一种联系方式。例如，利用用户的联系人信息，您可以：

- 在用户选择重置其密码时发送临时密码。
- 在更新用户的个人信息或财务信息后向用户发送通知。
- 发送促销信息（例如，特别优惠或折扣）。
- 发送账户摘要或账单提醒。

对于像这样的使用案例，将消息发送到经过验证的目的地非常重要。否则，您可能会将消息发送到错误键入的无效电子邮件地址或电话号码。或者更糟糕的是，您可能会将敏感信息发送给冒充您的用户的坏人。

为了帮助确保您仅将消息发送给正确的人员，请配置您的 Amazon Cognito 用户池以使用户在注册时必须提供以下内容：

a. 一个电子邮件地址或电话号码。
b. Amazon Cognito 发送到该电子邮件地址或电话号码的验证码。如果 24 小时后，您的用户代码或链接不再有效，请调用 ResendConfirmationCode API 操作以生成并发送新代码或链接。

通过提供验证码，用户可以证明他们有权访问收到该代码的邮箱或手机。在用户提供该代码后，Amazon Cognito 将通过以下方式更新用户池中的用户相关信息：

- 将用户的状态设置为 `CONFIRMED`。
- 更新用户的属性以指示已验证电子邮件地址或电话号码。

要查看此信息，您可以使用 Amazon Cognito 控制台。或者，您可以使用 `AdminGetUser` API 操作、带 `admin-get-user` 的 AWS CLI 命令或某个 AWS 开发工具包中的相应操作。

如果用户具有经验证的联系方式，Amazon Cognito 会在用户请求重置密码时自动向其发送消息。

配置您的用户池以要求电子邮件或手机验证

验证用户的电子邮件地址和电话号码时，确保可以联系到用户。在 AWS Management Console中完成以下步骤以配置用户群体，要求用户确认其电子邮件地址或电话号码。

Note

如果您的账户中还没有用户群体，请参阅用户池入门 (p. 30)。

配置用户池

1. 导航到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 从导航窗格中选择 User Pools（用户池）。从列表中选择一个现有用户池，或创建一个用户池。
3. 选择 Sign-up experience（登录体验）选项卡，然后找到 Attribute verification and user account confirmation（属性验证和用户账户确认）。选择编辑。
4. 选择是否激活 Cognito-assisted verification and confirmation（Cognito 辅助验证和确认），从而使 Amazon Cognito 向用户注册或创建用户配置文件时选择的用户联系人属性发送消息。Amazon Cognito 为用户发送提供代码或链接的消息，在用户确认收到该代码或链接后，系统将验证其属性并确认用户配置文件以进行登录。

   Note

   您还可以禁用 Cognito-assisted verification and confirmation（Cognito 辅助验证和确认）并使用经过身份验证的 API 操作或 Lambda 触发器验证属性并确认用户。
   如果您选择此选项，则 Amazon Cognito 不会在用户注册时发送验证码。如果您要使用自定义身份验证流程验证至少一种联系方式，而不使用来自 Amazon Cognito 的验证码，请选择此选项。例如，您可以使用一个预注册 Lambda 触发器，该触发器将自动验证属于特定域的电子邮件地址。
   如果您不验证用户的联系人信息，用户可能无法使用您的应用程序。请记住，用户需要经验证的联系人信息才能：

   - Reset their passwords（重置密码）– 当用户在您的应用程序中选择调用 `ForgotPassword` API 操作的选项时，Amazon Cognito 会将临时密码发送到用户的电子邮件地址或电话号码。Amazon Cognito 仅在用户具有至少一个经验证的联系方式时发送此密码。
   - Sign in by using an email address or phone number as an alias（通过使用电子邮箱地址或电话号码作为别名进行登录）– 如果您将用户池配置为允许这些别名，则用户只能在别名经过验证后使用别名进行登录。有关更多信息，请参阅自定义登录属性 (p. 246)。

5. 选择您的 Attributes to verify（要验证的属性）：

   发送 SMS 消息，验证电话号码

   Amazon Cognito 将在用户注册时通过 SMS 消息发送验证码。如果您通常通过 SMS 消息与用户通信，请选择此选项。例如，如果您要发送交付通知、约会确认或提醒，则将需要使用经过验证的电话号码。确认账户时，用户电话号码将成为已验证属性；您必须采取其它操作来验证用户电子邮件地址并与其进行通信。

   发送电子邮件消息，验证电子邮件地址

   Amazon Cognito 将在用户注册时通过电子邮件发送验证码。如果您通常通过电子邮件与用户通信，请选择此选项。例如，如果您要发送账单、订单摘要或特别优惠，则将需要使用经过验证的电子邮件地址。确认账户时，用户电子邮件地址将成为已验证属性；您必须采取其它操作来验证用户电话号码并与其进行通信。

   如果电话号码可用，则发送 SMS 消息，否则发送电子邮件

   如果您不要求所有用户都拥有相同的经验证的联系方式，请选择此选项。在这种情况下，您的应用程序中的注册页面可能会要求用户仅验证其首选联系方式。当 Amazon Cognito 发送验证码时，会将该代码发送到来自您应用程序的 `SignUp` 请求中提供的联系方式。如果用户同时提供了电子邮件地址和电话号码，并且您的应用程序在 `SignUp` 请求中提供了这两种联系方式，Amazon Cognito 将仅向电话号码发送验证码。

   如果您要求用户同时验证电子邮件地址和电话号码，则选择此选项。Amazon Cognito 将在用户注册时验证一种联系方式，而且您的应用程序必须在用户登录后验证另一种联系方式。有关更多信息，请参阅在您要求用户确认电子邮件地址和电话号码的情况下 (p. 171)。

6. 选择 Save changes（保存更改）。

使用电子邮件或电话验证的身份验证流程

如果您的用户池要求用户验证其联系人信息，则当用户注册时，您的应用程序必须促进以下流程：

1. 用户通过输入用户名称、电话号码和/或电子邮件地址及其他可能属性，在您的应用程序中进行注册。
2. Amazon Cognito 服务接收来自应用程序的注册请求。验证该请求包含注册所需的所有属性后，该服务将完成注册过程并向用户的手机（通过 SMS 消息）或电子邮件发送确认码。代码的有效期为 24 小时
3. 该服务向应用程序返回信息，表示注册过程已完成且用户账户正等待确认。响应中包含关于确认代码所发送到位置的信息。此时，用户账户处于未确认状态，而且用户的电子邮件地址和电话号码未经验证。
4. 现在，应用程序会提示用户输入确认代码。用户无需立即输入代码。但是，用户只有在输入确认代码后才可登录。
5. 用户在应用程序中输入确认代码。
6. 应用程序调用 `ConfirmSignUp` 将代码发送到 Amazon Cognito 服务，该服务将验证代码并在代码正确时将用户账户设置为已确认状态。成功确认用户账户之后，Amazon Cognito 服务会自动将用于确认（电子邮件地址或电话号码）的属性标记为已验证。除非此属性的值发生更改，否则用户无需再次进行验证。
7. 此时，用户账户处于已确认状态，用户可以登录。

在您要求用户确认电子邮件地址和电话号码的情况下

Amazon Cognito 在用户注册时仅验证一种联系方式。如果 Amazon Cognito 必须在验证电子邮件地址或电话号码之间进行选择，则会选择通过 SMS 消息发送验证码来验证电话号码。例如，如果您将用户池配置为允许用户验证电子邮件地址或电话号码，并且您的应用程序在注册时提供了这两个属性，则 Amazon Cognito 将仅验证电话号码。在用户验证其电话号码后，Amazon Cognito 会将用户的状态设置为 `CONFIRMED`，并允许用户登录您的应用程序。

在用户登录后，您的应用程序会提供相应选项来验证在注册期间未验证的联系方式。为了验证第二种联系方式，您的应用程序将调用 `VerifyUserAttribute` API 操作。请注意，此操作需要 `AccessToken` 参数，而 Amazon Cognito 只为经过身份验证的用户提供访问令牌。因此，您只能在用户登录后验证第二种联系方式。

如果您要求用户同时验证电子邮件地址和电话号码，请执行以下操作：

1. 配置用户池以允许用户验证电子邮件地址或电话号码。
2. 在应用程序的注册流程中，要求用户提供电子邮件地址和电话号码。调用 `SignUp` API 操作，并为 `UserAttributes` 参数提供电子邮件地址和电话号码。此时，Amazon Cognito 会向用户的手机发送一个验证码。
3. 在应用程序界面中，会显示一个确认页面以供用户输入验证码。通过调用 `ConfirmSignUp` API 操作来确认用户。此时，用户的状态为 `CONFIRMED`，并且用户的电话号码已验证，但电子邮件地址未验证。
4. 显示登录页，并通过调用 `InitiateAuth` API 操作对用户进行身份验证。对用户进行身份验证后，Amazon Cognito 将向您的应用程序返回访问令牌。
5. 调用 `GetUserAttributeVerificationCode` API 操作。在请求中指定以下参数：

   - `AccessToken` – Amazon Cognito 在用户登录时返回的访问令牌。
   - `AttributeName` – 将 `"email"` 指定为属性值。

   Amazon Cognito 将向用户的电子邮件地址发送一个验证码。
6. 显示一个确认页面以供用户输入验证码。当用户提交代码时，请调用 `VerifyUserAttribute` API 操作。在请求中指定以下参数：

   - `AccessToken` – Amazon Cognito 在用户登录时返回的访问令牌。

- `AttributeName` – 将 `"email"` 指定为属性值。
- `Code` – 用户提供的验证码。

此时，电子邮件地址已验证。

允许用户在您的应用程序中注册但以用户群体管理员身份进行确认

您可能不希望您的用户群体自动在用户群体中发送验证消息，但仍希望允许任何人注册账户。例如，该模型为人工审查新的注册请求以及批量验证和处理注册留出了空间。您可以在 Amazon Cognito 控制台中或使用经 IAM 身份验证的 API 操作 AdminConfirmSignUp 来确认新的用户账户。无论您的用户群体是否发送验证消息，您都能以管理员身份确认用户账户。

您只能使用此方法确认用户的自助注册。要以管理员身份确认您创建的用户，请创建 AdminSetUserPassword API 请求并将 `Permanent` 设置为 `True`。

1. 用户通过输入用户名称、电话号码和/或电子邮件地址及其他可能属性，在您的应用程序中进行注册。
2. Amazon Cognito 服务接收来自应用程序的注册请求。验证该请求包含注册所需的所有属性之后，该服务将完成注册过程，并向应用程序返回信息，表示注册已完成且正在等待确认。此时，用户账户处于未确认状态。账户经过确认后，用户才可登录。
3. 确认用户的账户。您必须使用 AWS 凭证登录 AWS Management Console 或签署 API 请求才能确认账户。
   a. 要在 Amazon Cognito 控制台中确认用户，请导航到用户选项卡，选择要确认的用户，然后从操作菜单中选择确认。
   b. 要在 AWS API 或 CLI 中确认用户，请在 AWS CLI 中创建 AdminConfirmSignUp API 请求或 admin-confirm-sign-up。
4. 此时，用户账户处于已确认状态，用户可以登录。

计算密钥哈希值

作为最佳实践，请将客户端密钥分配给您的机密应用程序客户端。当您向应用程序客户端分配客户端密钥时，您的 Amazon Cognito 用户群体 API 请求必须包括一个哈希值，用于包含请求正文中的客户端密钥。为了验证您对以下列表中 API 操作的客户端密钥的了解，请将客户端密钥与应用程序客户端 ID 和用户的用户名连接起来，然后对该字符串进行 base64 编码。

以下 Amazon Cognito 用户群体 API 接受 `SecretHash` 参数中的客户端密钥哈希值。

- ConfirmForgotPassword
- ConfirmSignUp
- ForgotPassword
- ResendConfirmationCode
- SignUp

此外，以下 API 接受 `SECRET_HASH` 参数中的客户端密钥哈希值，这可以是在身份验证参数中或在质询响应中。

API 操作	SECRET_HASH 的父参数
InitiateAuth	AuthParameters
AdminInitiateAuth	AuthParameters

API 操作	SECRET_HASH 的父参数
RespondToAuthChallenge	ChallengeResponses
AdminRespondToAuthChallenge	ChallengeResponses

密钥哈希值是 Base 64 编码的加密哈希消息身份验证代码（HMAC，Hash Message Authentication Code），使用用户群体客户端的私有密钥、用户名以及消息中的客户端 ID 进行计算。以下伪代码显示如何计算此值。在此伪代码中，+ 表示串联，HMAC_SHA256 代表使用 HmacSHA256 生成 HMAC 值的函数，Base64 代表生成哈希输出的 Base-64 编码版本的函数。

```
Base64 ( HMAC_SHA256 ( "Client Secret Key", "Username" + "Client Id" ) )
```

有关如何计算和使用 SecretHash 参数的详细概述，请参阅 AWS 知识中心内的如何解决 Amazon Cognito 用户群体 API 中的“无法验证客户端 <client-id> 的密钥哈希”错误？

您可以在服务器端应用程序代码中使用以下代码示例：

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)
```

无需验证电子邮件或电话号码即可确认用户账户

预注册 Lambda 触发器可用于在注册时自动确认用户账户，而无需提供确认码或者验证电子邮件或电话号码。通过此方法进行确认的用户可立即登录，而无需接收代码。

您还可通过此触发器将用户的电子邮件或电话号码标记为已验证。

> Note
>
> 虽然这种方法对刚入门的用户而言很方便，但我们建议至少自动验证电子邮件或电话号码之一。否则，如果用户忘记密码，可能就无法进行恢复。

如果您不要求用户在注册时接收并输入确认码，也不在预注册 Lambda 触发器中自动验证电子邮件和电话号码，则您将承担对于该用户账户不具备经验证的电子邮件地址或电话号码的风险。用户可以稍后验证电子邮件地址或电话号码。但是，如果用户忘记自己的密码且没有经验证的电子邮件地址或电话号码，则用户将被锁定而无法使用账户，因为忘记密码流程需要经验证的电子邮件或电话号码以便向用户发送验证码。

当用户更改其电子邮件或电话号码时应进行验证

当用户在应用程序中更新其电子邮件地址或电话号码时，如果您将用户群体配置为自动验证该属性，Amazon Cognito 会立即向用户发送带有验证码的消息。然后，用户必须将验证消息中的代码提供给您的应用程序。然后，您的应用程序在 VerifyUserAttribute API 请求中提交代码，以完成对新属性值的验证。

如果用户群体不要求用户验证更新的电子邮件地址或电话号码，Amazon Cognito 会立即更改更新的 `email` 或 `phone_number` 属性的值，并将该属性标记为未验证。您的用户无法使用未经验证的电子邮件或电话号码登录。他们必须先完成对更新值的验证，然后才能将该属性用作登录别名。

如果您的用户群体要求用户验证更新的电子邮件地址或电话号码，则 Amazon Cognito 会将属性保持为已验证状态并设置为其原始值，直到您的用户验证新属性值为止。如果属性是登录的别名，您的用户可以使用原始属性值登录，直到验证过程将属性更改为新值。有关如何配置用户群体以要求用户验证更新的属性的更多信息，请参阅配置电子邮件或电话验证。

您可以使用自定义消息 Lambda 触发器自定义验证消息。有关更多信息，请参阅自定义消息 Lambda 触发器 (p. 151)。当用户的电子邮件地址或电话号码未经验证时，您的应用程序应通知用户必须验证该属性，并为用户提供一个按钮或链接以验证其新的电子邮件地址或电话号码。

针对由管理员或开发人员创建的用户账户的确认和验证过程

由管理员或开发人员创建的用户账户已经处于已确认状态，所以用户无需输入确认代码。Amazon Cognito 服务向这些用户发送的邀请消息包含用户名和临时密码。用户需要在登录前更改密码。有关更多信息，请参阅自定义电子邮件和 SMS 消息 (p. 177)中的 以管理员身份创建用户账户 (p. 175)和使用 Lambda 触发器自定义用户池工作流 (p. 111)中的自定义消息触发器。

针对导入的用户账户的确认和验证过程

使用 AWS Management Console、CLI 或 API 中的用户导入功能创建的用户账户（请参阅通过 CSV 文件将用户导入用户池中 (p. 184)）已处于已确认状态，因此用户无需输入确认代码。没有发送邀请消息。但是，导入的用户账户要求用户首先调用 `ForgotPassword` API 来请求代码，然后通过调用 `ConfirmForgotPassword` API 来使用发送的代码创建密码，之后方可登录。有关更多信息，请参阅要求导入的用户重置密码 (p. 193)。

导入用户账户时，用户的电子邮件或电话号码必须已标记为已验证，从而用户无需验证即可登录。

在测试应用程序时发送电子邮件

当用户在您的用户群体的客户端应用程序中创建和管理其账户时，Amazon Cognito 将向用户发送电子邮件。如果您将用户池配置为要求电子邮件验证，Amazon Cognito 将在以下情况下发送电子邮件：

- 用户注册。
- 用户更新其电子邮件地址。
- 用户执行一项调用 `ForgotPassword` API 操作的操作。
- 您以管理员身份创建用户账户。

根据发起电子邮件递送的操作，电子邮件将包含验证码或临时密码。您的用户必须接收这些电子邮件并理解消息。否则，他们可能无法登录并使用您的应用程序。

要确保电子邮件成功发送并且邮件看起来正确，请测试应用程序中从 Amazon Cognito 启动电子邮件传送的操作。例如，通过使用应用程序中的注册页面或通过使用 `SignUp` API 操作，您可以通过使用测试电子邮件地址进行注册来启动电子邮件传送。在通过此方式进行测试时，请记住以下几点：

> **重要提示**
>
> 当您使用电子邮件地址测试从 Amazon Cognito 启动电子邮件传送的操作时，请勿使用虚假的电子邮件地址（没有邮箱的电子邮件地址）。使用真实的电子邮件地址才能接收来自 Amazon Cognito 的电子邮件，而不会创建查无此人的邮件。
> 在 Amazon Cognito 未能将电子邮件传送到收件人邮箱时会产生查无此人的邮件，如果邮箱不存在，则始终会发生这种情况。
> Amazon Cognito 会限制持续产生查无此人的邮件的AWS账户可发送的电子邮件数量。

当您测试启动电子邮件传送的操作时，请使用下列电子邮件地址之一以防止出现查无此人的邮件：

- 您拥有的用于测试的电子邮件账户的地址。当您使用自己的电子邮件地址时，您将收到 Amazon Cognito 发送的电子邮件。利用此电子邮件，您可以使用验证码来测试应用程序中的注册体验。如果您为用户池自定义了电子邮件，则可检查自定义项看起来是否正确。
- 邮箱模拟器地址：success@simulator.amazonses.com。如果您使用模拟器地址，Amazon Cognito 将成功发送电子邮件，但您无法查看。当您不需要使用验证码并且不需要检查电子邮件时，此选项很有用。
- 添加了任意标签的邮箱模拟器地址（如 success+user1@simulator.amazonses.com 或 success+user2@simulator.amazonses.com）。Amazon Cognito 可成功向这些地址发送电子邮件，但您无法查看其发送的电子邮件。当您希望通过向用户池添加多个测试用户来测试注册过程，并且每个测试用户都具有一个唯一的电子邮件地址时，此选项很有用。

以管理员身份创建用户账户

创建用户池之后，您可以使用AWS Management Console以及 AWS Command Line Interface 或 Amazon Cognito API 创建用户。您可以为用户池中的新用户创建配置文件，并通过 SMS 或电子邮件向用户发送带有注册说明的欢迎消息。

开发人员和管理员可以执行以下任务：

- 通过使用 AWS Management Console或调用 `AdminCreateUser` API 来创建新的用户配置文件。
- 指定临时密码或允许 Amazon Cognito 自动生成一个密码。
- 指定是否将提供的电子邮件地址和电话号码标记已针对新用户进行验证。
- 通过 AWS Management Console或自定义消息 Lambda 触发器为新用户指定自定义 SMS 和电子邮件邀请消息。有关更多信息，请参阅使用 Lambda 触发器自定义用户池工作流 (p. 111)。
- 指定是否通过 SMS、电子邮件或两者发送邀请消息。
- 通过调用 `AdminCreateUser` API，并为 `RESEND` 参数指定 `MessageAction`，向现有用户重新发送欢迎消息。

  > **Note**
  >
  > 目前，无法使用 AWS Management Console执行此操作。
- 创建用户时禁止发送邀请消息。

- 为用户账户指定到期时间限制（最多 90 天）。
- 允许用户自行注册或要求只能由管理员添加新用户。

针对由管理员或开发人员创建的用户的身份验证流程

针对这些用户的身份验证流程包括提交新密码并提供必须属性的任何缺失值的额外步骤。下面列出了相关步骤；步骤 5、6 和 7 特定于这些用户。

1. 用户通过提交其用户名和密码首次开始登录。
2. 开发工具包调用 `InitiateAuth(Username, USER_SRP_AUTH)`。
3. Amazon Cognito 返回具有加密盐和私有密钥块的 `PASSWORD_VERIFIER` 质询。
4. 开发工具包执行 SRP 计算并调用 `RespondToAuthChallenge(Username, <SRP variables>, PASSWORD_VERIFIER)`。
5. Amazon Cognito 返回 `NEW_PASSWORD_REQUIRED` 质询。此质询的主体包括用户的当前属性，以及用户池中当前在用户配置文件中没有值的任何必需属性。有关更多信息，请参阅 RespondToAuthChallenge。
6. 系统提示用户输入新密码和必需属性的任何缺失值。
7. 开发工具包调用 `RespondToAuthChallenge(Username, <New password>, <User attributes>)`。
8. 如果用户需要 MFA 的第二安全要素，则 Amazon Cognito 将返回 SMS_MFA 质询并提交代码。
9. 用户成功更改自己的密码并选择性地提供属性值或完成 MFA 之后，用户将登录，并且系统将发出令牌。

用户满足所有质询后，Amazon Cognito 服务会将用户标记为已确认，并为用户发出 ID 令牌、访问令牌和刷新令牌。有关更多信息，请参阅将令牌与用户池结合使用 (p. 203)。

在 AWS Management Console 中创建新用户

您可以设置用户密码要求、配置发送给用户的邀请和验证消息，以及使用 Amazon Cognito 控制台添加新用户。

设置密码策略并启用自行注册

您可以配置设置来降低密码复杂性，以及用户是否可以在用户池中使用公有 API 进行注册。

配置密码策略

1. 导航到 Amazon Cognito 控制台，选择 User Pools（用户池）。
2. 从列表中选择一个现有用户池，或创建一个用户池。
3. 选择 Sign-in experience（登录体验）选项卡，然后查找 Password policy（密码策略）。选择编辑。
4. 选择 Custom（自定义）的 Password policy mode（密码策略模式）。
5. 选择 Password minimum length（密码最小长度）。有关密码长度要求的限制，请参阅用户池资源配额。
6. 选择 Password complexity（密码复杂性）要求。
7. 选择管理员设置的密码应在多长时间内有效。
8. 选择 Save changes（保存更改）。

允许自助注册

1. 导航到 Amazon Cognito 控制台，选择 User Pools（用户池）。
2. 从列表中选择一个现有用户池，或创建一个用户池。

3. 选择 Sign-up experience（注册体验）选项卡，然后查找 Self-service sign-up（自助注册）。选择 Edit（编辑）。
4. 选择是否 Enable self-registration（启用自助注册）。自助注册通常用于需要在用户池中注册新用户而不分发客户端密钥或 AWS Identity and Access Management (IAM) API 凭证的公有应用程序客户端。

   禁用自助注册

   如果您不启用自助注册，则必须通过使用 IAM API 凭证的管理 API 操作或通过联合提供商登录来创建新用户。
5. 选择 Save changes（保存更改）。

自定义电子邮件和 SMS 消息

自定义用户消息

当您邀请用户登录、注册用户账户或登录并提示用户进行多重身份验证（MFA）时，您可以自定义 Amazon Cognito 发送给用户的消息。

Note

当您在用户池中创建用户并邀请他们登录时将发送 Invitation message（邀请消息）。Amazon Cognito 将初始登录信息发送到用户的电子邮件地址或电话号码。
当用户在您的用户池中注册用户账户时将发送 Verification message（验证消息）。Amazon Cognito 向用户发送代码。当用户向 Amazon Cognito 提供代码时，他们会验证自己的联系人信息并确认自己的账户以进行登录。验证代码的有效期为 24 小时。
当您在用户池中启用 SMS MFA，并且已配置 SMS MFA 的用户登录并提示输入 MFA 时，将发送 MFA message（MFA 消息）。

1. 导航到 Amazon Cognito 控制台，选择 User Pools（用户池）。
2. 从列表中选择一个现有用户池，或创建一个用户池。
3. 选择 Messaging（消息收发）选项卡，然后查找 Message templates（消息模板）。选择 Verification messages（验证消息）、Invitation messages（邀请消息）或 MFA messages（MFA 消息），然后选择 Edit（编辑）。
4. 自定义所选消息类型的消息。

   Note

   自定义消息时，必须包括消息模板中的所有变量。如果变量（例如，{####}）不包括在内，您的用户将没有足够的信息来完成消息操作。
   有关更多信息，请参阅消息模板。
5. a. Verification messages（验证消息）

      i. 选择用于 Email（电子邮件）消息的 Verification type（验证类型）。Code（代码）验证将发送用户必须输入的数字代码。Link（链接）验证将发送一个链接，用户可以点击该链接以验证其联系人信息。用于 Link（链接）消息变量中的文本显示为超链接文本。例如，使用变量 {##Click here##} 的消息模板在电子邮件中显示为单击此处。
      ii. 输入用于 Email（电子邮件）消息的 Email subject（电子邮件主题）。
      iii. 输入用于 Email（电子邮件）消息的自定义 Email message（电子邮件消息）模板。您可以使用 HTML 自定义此模板。
      iv. 输入用于 SMS 消息的自定义 SMS message（SMS 消息）模板。
      v. 选择 Save changes（保存更改）。

   b. Invitation messages（邀请消息）

      i. 输入用于 Email（电子邮件）消息的 Email subject（电子邮件主题）。
      ii. 输入用于 Email（电子邮件）消息的自定义 Email message（电子邮件消息）模板。您可以使用 HTML 自定义此模板。

iii. 输入用于 SMS 消息的自定义 SMS message（SMS 消息）模板。
iv. 选择 Save changes（保存更改）。

c. MFA messages（MFA 消息）

i. 输入用于 SMS 消息的自定义 SMS message（SMS 消息）模板。
ii. 选择 Save changes（保存更改）。

创建用户

创建用户

您可以从 Amazon Cognito 控制台为用户池创建新用户。通常，用户可以在设置密码后登录。要使用电子邮件地址登录，用户必须验证 `email` 属性。要使用电话号码登录，用户必须验证 `phone_number` 属性。要以管理员身份确认账户，您还可以使用 AWS CLI 或 API 或者使用联合身份提供商创建用户配置文件。有关更多信息，请参阅 Amazon Cognito API 参考。

1. 导航到 Amazon Cognito 控制台，选择 User Pools（用户池）。
2. 从列表中选择一个现有用户池，或创建一个用户池。
3. 选择 Users（用户）选项卡，然后选择 Create user（创建用户）。
4. 检查 User pool sign-in and security requirements（用户池登录和安全要求）以获取有关密码要求、可用的账户恢复方法和用户池的别名属性的指导。
5. 选择您希望如何发送 Invitation message（邀请消息）。选择 SMS 消息和/或电子邮件消息。

   Note

   在您可以发送邀请消息之前，请在用户池的 Messaging（消息）选项卡中使用 Amazon Simple Notification Service 和 Amazon Simple Email Service 配置发件人和 AWS 区域。收件人消息和数据费率适用。Amazon SES 单独向您收取电子邮件消息费用，Amazon SNS 单独向您收取 SMS 消息费用。
6. 选择用于新用户的 Username（用户名）。
7. 选择您是要为用户 Create a password（创建密码），还是让 Amazon Cognito Generate a password（生成密码）。任何临时密码都必须遵守用户池密码策略。
8. 选择 Create（创建）。
9. 选择 Users（用户）选项卡，然后选择用户的 User name（用户名）条目。添加和编辑 User attributes（用户属性）和 Group memberships（组成员资格）。查看 User event history（用户事件历史记录）。

向用户池添加组

借助对 Amazon Cognito 用户池中组的支持，您可以创建和管理组、将用户添加到组以及从组中删除用户。使用组可创建用户集合以管理其权限或表示不同类型的用户。您可以向组分配一个 AWS Identity and Access Management（IAM）角色以定义组成员的权限。

您可以使用组以在用户池中创建用户集合，这通常用于为这些用户设置权限。例如，您可以为作为您网站和应用程序的读者、贡献者或编辑者的用户创建单独的组。通过使用与组关联的 IAM 角色，您还可以为那些不同的组设置不同的权限，从而只有贡献者可以将内容放置在 Amazon S3 中，并且只有编辑者可以通过 Amazon API Gateway 中的 API 发布内容。

您可以通过 AWS Management Console、API 和 CLI 创建并管理用户池中的组。作为开发人员 (使用 AWS 凭证)，您可以创建、读取、更新、删除并列出用户池的组。您还可以将用户添加到组和从组中删除用户。

在用户池中使用组不会产生额外费用。有关更多信息，请参阅 Amazon Cognito 定价。

向组分配 IAM 角色

您可以使用组通过 IAM 角色控制资源的权限。IAM 角色包含信任策略和权限策略。角色的信任策略指定谁可使用该角色。权限策略指定组成员可以访问的操作和资源。在您创建 IAM 角色时，请设置角色的信任策略以允许您的组用户担任该角色。请在角色的权限策略中，指定您希望组具有的权限。

在 Amazon Cognito 中创建组时，可以通过提供角色的 ARN 指定 IAM 角色。当组成员使用 Amazon Cognito 登录时，他们可以从身份池接收临时凭证。他们的权限由关联的 IAM 角色确定。

单个用户可处于多个组中。作为开发人员，当一个用户位于多个组中时，您可以使用以下选项自动选择 IAM 角色：

- 您可以为每个组分配优先级值。将选择优先级较高（值较低）的组，并应用其关联的 IAM 角色。
- 通过身份池请求用户的 AWS 凭证时，您的应用程序还可以从可用角色中进行选择，只需在 GetCredentialsForIdentity `CustomRoleARN` 参数中指定角色 ARN。指定的 IAM 角色必须与适用于用户的角色相匹配。

将优先级值分配到组

一个用户可属于多个组。在用户的访问令牌和 ID 令牌中，`cognito:groups` 声明包含用户所属的所有组的列表。`cognito:roles` 断言包含与这些组对应的角色列表。

由于一个用户可以属于多个组，因此可为每个组分配一个优先级。这是一个非负数值，用于指定该组相对于用户池中用户所属其它组的优先级。零是代表最高优先级的值。具有较低优先级值的组优先于具有较高或空优先级值的组。如果一个用户属于两个或更多组，则具有最低优先级值的组的 IAM 角色将应用于用户 ID 令牌中的 `cognito:preferred_role` 声明。

两个组可以具有相同的优先级值。如果发生这种情况，则两个组之间不存在优先情况。如果具有相同优先级值的两个组还具有相同的角色 ARN，则该角色将用于每个组中用户的 ID 令牌的 `cognito:preferred_role` 陈述。如果两个组具有不同的角色 ARN，则不会在用户的 ID 令牌中设置 `cognito:preferred_role` 断言。

使用组控制使用 Amazon API Gateway 的权限

您可以使用用户池中的组控制使用 Amazon API Gateway 的权限。用户所属的组包含在 `cognito:groups` 声明中的用户池的 ID 令牌和访问令牌中。您可以通过请求向 Amazon API Gateway 提交 ID 或访问令牌，并使用 Amazon Cognito 用户池授权方获取 REST API。有关更多信息，请参阅《API Gateway 开发人员指南》中的使用 Amazon Cognito 用户池作为授权方控制对 REST API 的访问。

您还可以使用自定义 JWT 授权方授权访问 Amazon API Gateway HTTP API。有关更多信息，请参阅《API Gateway 开发人员指南》中的使用 JWT 授权方控制对 HTTP API 的访问。

组的限制

用户组受以下限制的约束：

- 您可以创建的组的数量受 Amazon Cognito 服务限制 (p. 539)的限制。
- 不能对组进行嵌套。
- 不能搜索组中的用户。
- 不能按名称搜索组，但可以列出组。
- 只能删除没有成员的组。

在 AWS Management Console 中创建新组

使用以下过程创建新组。

创建新组。

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 Groups（组）选项卡，然后选择 Create group（创建组）。
5. 在 Create a group（创建组）页面的 Group name（组名称）中，为您的新组输入一个易记名称。
6. 您可以选择使用以下任意字段提供有关此组的其它信息：
   - Description（说明）– 输入有关这个新组将用于什么的详细信息。
   - Precedence（优先顺序）– Amazon Cognito 根据给定用户所属群组具有的较低优先级值，评估并应用所有群组权限。将选择优先级较低的组，并应用其关联的 IAM 角色。有关更多信息，请参阅将优先级值分配到组 (p. 179)。
   - IAM role（IAM 角色）– 当您需要控制对资源的权限时，您可以为组分配 IAM 角色。如果您要将用户池与身份池集成，并且身份池配置为从令牌中选择角色，则 IAM role (IAM 角色) 设置将确定在用户的 ID 令牌中分配哪个角色。有关更多信息，请参阅向组分配 IAM 角色 (p. 179)。
   - Add users to this group（将用户添加到此组）– 创建后将现有用户添加为该组的成员。
7. 选择 Create（创建）以确认。

管理和搜索用户账户

创建用户池后，您可以使用AWS Management Console以及 AWS Command Line Interface 或 Amazon Cognito API 查看和管理用户。本主题将介绍如何使用 AWS Management Console查看和搜索用户。

查看用户属性

请使用以下过程在 Amazon Cognito 控制台中查看用户属性。

查看用户属性

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 Users（用户）选项卡，然后在列表中选择用户。
5. 在用户详细信息页面，您可以在 User attributes（用户属性）中查看哪些属性与用户关联。

重置用户的密码

请使用以下过程在 Amazon Cognito 控制台重置用户的密码。

重置用户的密码

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 Users（用户）选项卡，然后在列表中选择用户。
5. 在用户详细信息页面上，选择 Actions（操作）、Reset password（重置密码）。
6. 在 Reset password（重置密码）对话框中，查看信息，准备就绪后，选择 Reset（重置）。

   该操作会立即导致向用户发送确认代码，并通过将用户状态更改为 `RESET_REQUIRED` 来禁用用户的当前密码。Reset password（重置密码）代码的有效期为 1 小时。

搜索用户属性

如果您已创建用户池，则可以在 AWS Management Console 的 Users（用户）面板中搜索。您还可以使用 Amazon Cognito ListUsers API，该 API 接受 Filter 参数。

您可以搜索以下任何标准属性。自定义属性不可搜索。

- username（区分大小写）
- email
- phone_number
- name
- given_name
- family_name
- preferred_username
- cognito:user_status（在控制台中称为Status（状态））（区分大小写）
- status（在控制台中称为Enabled（已启用））（区分大小写）
- sub

> **Note**
>
> 您还可以使用客户端筛选条件列出用户。服务器端筛选条件匹配的属性不超过 1 个。对于高级搜索，请使用客户端筛选条件，其中包含 AWS Command Line Interface 中 `list-users` 操作的 `--query` 参数。当您使用客户端筛选条件时，ListUsers 会返回一个不包含或包含多个用户的分页列表。您可以连续接收多个结果为零的页面。对返回的每个分页令牌重复查询，直到您收到一个空的分页令牌值，然后查看合并结果。
> 有关服务器端和客户端筛选的详细信息，请参阅《AWS Command Line Interface 用户指南》中的筛选 AWS CLI 输出。

使用 AWS Management Console 搜索用户

如果您已创建用户池，则可以在 AWS Management Console 的 Users（用户）面板中搜索。

AWS Management Console搜索始终为前缀 (“starts with”) 搜索。

在 Amazon Cognito 控制台中搜索用户

1. 转到 Amazon Cognito 控制台。系统可能会提示您输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 Users（用户）选项卡，然后在搜索字段中输入用户的用户名。请注意，某些属性值区分大小写 [例如，Username（用户名）]。

   您还可以通过调整搜索筛选条件来查找用户，将范围缩小到其它用户属性，如 Email（电子邮件）、Phone number（电话号码）或 Last name（姓）。

使用 `ListUsers` API 搜索用户

要从应用程序中搜索用户，请使用 Amazon Cognito ListUsers API。此 API 使用以下参数：

- `AttributesToGet`：一组字符串，其中每个字符串均为将针对搜索结果中的每位用户返回的用户属性的名称。如果字符串组为空，将返回所有属性。

- `Filter`：筛选条件字符串，格式为 `"AttributeName Filter-Type "AttributeValue""`。筛选条件字符串中的引号必须使用反斜杠 (\) 字符进行转义。例如，`"family_name = \"Reddy\""`。如果筛选条件字符串为空，`ListUsers` 将返回用户池中的所有用户。
  - `AttributeName`：要搜索的属性的名称。一次只能搜索一个属性。

    Note

    您只能搜索标准属性。自定义属性不可搜索。这是因为只有索引属性可搜索，而自定义属性不可索引。
  - `Filter-Type`：对于精确匹配，请使用 =，例如 `given_name = "Jon"`。对于前缀 (“starts with”) 匹配，请使用 ^=，例如 `given_name ^= "Jon"`。
  - `AttributeValue`：必须为每位用户匹配的属性值。
- `Limit`：要返回的最大用户数。
- `PaginationToken`：可从之前的搜索中获取更多结果的令牌。Amazon Cognito 会在一小时后让分页令牌过期。
- `UserPoolId`：应对其执行搜索的用户池的用户池 ID。

所有搜索都区分大小写。搜索结果按以 `AttributeName` 字符串命名的属性进行升序排列。

使用 `ListUsers` API 的示例

以下示例将返回所有用户并包括所有属性。

```
{
    "AttributesToGet": [],
    "Filter": "",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

以下示例将返回电话号码以“+1312”开头的所有用户并包括所有属性。

```
{
    "AttributesToGet": [],
    "Filter": "phone_number ^= \"+1312\"",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

以下示例将返回姓氏为“Reddy”的前 10 位用户。对于每个用户，搜索结果包含用户的名字、电话号码和电子邮件地址。如果用户池中有 10 个以上相匹配的用户，则响应将包含一个分页标记。

```
{
    "AttributesToGet": [
        "given_name", "phone_number", "email"
    ],
    "Filter": "family_name = \"Reddy\"",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

如果上一示例返回分页标记，则以下示例将返回与同一筛选条件字符串相匹配的接下来的 10 位用户。

```
{
    "AttributesToGet": [
        "given_name", "phone_number", "email"
    ],
    "Filter": "family_name = \"Reddy\"",
    "Limit": 10,
    "PaginationToken": "pagination_token_from_previous_search",
    "UserPoolId": "us-east-1_samplepool"
}
```

恢复用户账户

通过 AccountRecoverySetting 参数，您可以自定义用户在调用 ForgotPassword API 时可用于恢复其密码的方法。ForgotPassword 会将恢复码发送到已验证的电子邮件或已验证的电话号码。恢复代码的有效期为 1 小时。当您为用户池指定 AccountRecoverySetting 时，Amazon Cognito 会根据您设置的优先级选择代码发送目标。

当您定义 AccountRecoverySetting 并且用户配置了 SMS MFA 时，不能将 SMS 用作账户恢复机制。此设置的优先级已确定，其中 1 为最高优先级。Cognito 仅向指定方法之一发送验证。

例如，当管理员不希望用户自行恢复账户时，可以使用值 admin_only，这会改为要求用户联系管理员来重置账户。您不能将 admin_only 与任何其他帐户恢复机制一起使用。

如果您未指定 AccountRecoverySetting，Amazon Cognito 会使用旧机制来确定恢复密码的方法。在这种情况下，Cognito 首先使用经验证手机。如果找不到用户的经验证手机，Cognito 会退回，接下来使用经验证电子邮件地址。

有关 AccountRecoverySetting 的更多信息，请参阅《Amazon Cognito 身份提供商 API 参考》中的 CreateUserPool 和 UpdateUserPool。

忘记密码行为

我们允许用户在 1 小时内进行 5 到 20 次的密码重置代码请求或输入尝试，作为忘记密码和确认忘记密码操作的一部分。确切的值取决于与请求关联的风险参数。请注意，这种行为可能会发生变化。

将用户导入一个用户池

您可以使用以下两种方式将用户从现有用户目录或用户数据库导入或迁移到 Amazon Cognito 用户池中。您可以利用用户迁移 Lambda 触发器，在用户首次使用 Amazon Cognito 登录时迁移用户。借助这种方法，用户可以继续使用其现有的密码，不必在迁移到用户池后重置密码。或者，您可以上传 CSV 文件 (包含所有用户的用户配置文件属性)，批量迁移用户。以下各部分分别介绍了这两种方法。

主题

- 利用用户迁移 Lambda 触发器将用户导入用户池 (p. 183)
- 通过 CSV 文件将用户导入用户池中 (p. 184)

利用用户迁移 Lambda 触发器将用户导入用户池

使用这种方法，当用户首次登录您的应用程序或请求重置密码时，您可以将用户从现有用户目录无缝迁移到用户池。向您的用户池添加一个 迁移用户 Lambda 触发器 (p. 147) 函数，它会接收有关尝试登录的用户的元数据，并从外部身份源返回用户配置文件信息。有关此 Lambda 触发器的详细信息以及示例代码（包括请求和响应参数），请参阅迁移用户 Lambda 触发器参数 (p. 148)。

在开始迁移用户之前，请在您的 AWS 账户中创建一个用户迁移 Lambda 函数，并将该 Lambda 函数设置为您的用户池中的用户迁移触发器。向您的 Lambda 函数添加授权策略，该策略仅允许 Amazon Cognito 服务

账户主体 cognito-idp.amazonaws.com 调用该 Lambda 函数，并且只能在您自己的用户池的上下文中进行。有关更多信息，请参阅对 AWS Lambda 使用基于资源的策略（Lambda 函数策略）。

登录流程

1. 用户打开您的应用程序并使用 Amazon Cognito 用户池 API 或通过 Amazon Cognito 托管 UI 登录。有关如何使用 Amazon Cognito API 协助登录的更多信息，请参阅将 Amazon Cognito 身份验证和授权与 Web 和移动应用程序集成 (p. 20)。
2. 您的应用程序将用户名和密码发送至 Amazon Cognito。如果您的应用程序具有使用 AWS 开发工具包构建的自定义登录 UI，则您的应用程序必须将 InitiateAuth 或 AdminInitiateAuth 与 USER_PASSWORD_AUTH 或 ADMIN_USER_PASSWORD_AUTH 流程一起使用。当您的应用使用其中一个流程时，开发工具包会将密码发送到服务器。

   Note

   在添加用户迁移触发器之前，请在您的应用程序客户端的设置中激活 USER_PASSWORD_AUTH 或 ADMIN_USER_PASSWORD_AUTH 流程。您必须使用这些流程而不是默认 USER_SRP_AUTH 流程。Amazon Cognito 必须向您的 Lambda 函数发送密码，以便它可以验证您的用户在另一个目录中的身份验证。SRP 会在您的 Lambda 函数中隐藏您用户的密码。
3. Amazon Cognito 检查提交的用户名是否与用户池中的用户名或别名匹配。您可以将用户的电子邮件地址、电话号码或首选用户名设置为用户池中的别名。如果用户不存在，Amazon Cognito 会将参数（包括用户名和密码）发送到您的 迁移用户 Lambda 触发器 (p. 147) 函数。
4. 您的 迁移用户 Lambda 触发器 (p. 147) 函数使用您的现有用户目录或用户数据库检查用户，或验证用户身份。该函数返回 Amazon Cognito 存储在用户池的用户配置文件中的用户属性。仅当提交的用户名与别名属性匹配时，您才能返回 username 参数。如果您希望用户继续使用其现有密码，您的函数将在 Lambda 响应中将属性 finalUserStatus 设置为 CONFIRMED。您的应用程序必须返回 迁移用户 Lambda 触发器参数 (p. 148) 中显示的所有 "response" 参数。

   Important

   不要在您的用户迁移 Lambda 代码中记录整个请求事件对象。此请求事件对象包括用户的密码。如果您不清理日志，密码将显示在 CloudWatch Logs 中。
5. Amazon Cognito 在您的用户池中创建用户配置文件，并将令牌返回您的应用程序客户端。
6. 您的应用程序执行令牌接收，接受用户身份验证，然后继续处理请求的内容。

迁移用户后，请使用 USER_SRP_AUTH 进行登录。安全远程密码 (SRP) 协议不会通过网络发送密码，并为您在迁移期间使用的 USER_PASSWORD_AUTH 流程提供安全优势。

如果迁移期间出现错误（包括客户端设备或网络问题），您的应用程序会从 Amazon Cognito 用户池 API 接收错误响应。发生这种情况时，Amazon Cognito 可能会也可能不会在您的用户池中创建用户账户。然后，用户应尝试再次登录。如果登录反复失败，请尝试在您的应用程序中使用忘记密码流程重置用户密码。

忘记密码流程还会使用 UserMigration_ForgotPassword 事件源调用您的 迁移用户 Lambda 触发器 (p. 147) 函数。由于用户在请求密码重置时没有提交密码，因此 Amazon Cognito 在发送到您的 Lambda 函数的事件中不包含密码。您的函数只能在现有用户目录中查找用户并返回属性，以添加到用户池中的用户配置文件中。Amazon Cognito 会接收来自您的函数的响应，并通过电子邮件或 SMS 向用户发送密码重置代码。在用户验证代码后，他们可以在您的应用程序中设置新密码。如果您的应用程序具有自定义 UI，请构建支持忘记密码流程的页面。Amazon Cognito 托管 UI 具有用于忘记密码流程的内置页面。

通过 CSV 文件将用户导入用户池中

您可以将用户导入 Amazon Cognito 用户池中。用户信息从一个特殊格式的 .csv 文件导入。导入过程会设置所有用户属性的值，不过 password 除外。不支持导入密码，因为安全妥善做法要求密码不能为纯文本，而我们不支持导入哈希。这意味着，用户必须在首次登录时更改密码。因此，当使用此方法导入时，您的用户将处于 RESET_REQUIRED 状态。

您可以使用 AdminSetUserPassword API 请求，将 Permanent 参数设置为 true 来设置用户的密码。

Note

每个用户的创建日期就是将该用户导入用户池中的日期。创建日期不是导入的属性之一。

基本步骤如下：

1. 在 AWS Identity and Access Management（IAM）控制台中创建 Amazon CloudWatch Logs 角色。
2. 创建用户导入 .csv 文件。
3. 创建并运行用户导入任务。
4. 上传用户导入 .csv 文件。
5. 启动并运行用户导入任务。
6. 使用 CloudWatch 检查事件日志。
7. 要求导入的用户重置密码。

主题

- 创建 CloudWatch Logs IAM 角色 (p. 185)
- 创建用户导入 CSV 文件 (p. 186)
- 创建并运行 Amazon Cognito 用户池导入任务 (p. 188)
- 在 CloudWatch 控制台中查看用户池导入结果 (p. 192)
- 要求导入的用户重置密码 (p. 193)

创建 CloudWatch Logs IAM 角色

如果您使用的是 Amazon Cognito CLI 或 API，则需要创建一个 CloudWatch IAM 角色。以下过程描述了如何创建 IAM 角色，Amazon Cognito 可以使用该角色将导入作业的结果写入 CloudWatch Logs。

Note

在 Amazon Cognito 控制台中创建导入作业时，您可以同时创建 IAM 角色。当您选择 Create a new IAM role（创建新 IAM 角色）时，Amazon Cognito 会自动对该角色应用相应的信任策略和 IAM policy。

为用户群体导入创建 CloudWatch Logs IAM 角色（AWS CLI、API）

1. 登录AWS Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 为 AWS 服务 创建新 IAM 角色。有关详细说明，请参阅《AWS Identity and Access Management 用户指南》中的为 AWS 服务 创建一个角色。
   a. 当您为 Trusted entity type（可信实体类型）选择 Use case（使用案例）时，请选择任意服务。Amazon Cognito 目前未在服务使用案例中列出。
   b. 在 Add permissions（添加权限）屏幕中，选择 Create policy（创建策略）并插入以下策略声明。将 *REGION* 替换为您用户群体的 AWS 区域，例如 `us-east-1`。将 *ACCOUNT* 替换为您的 AWS 账户 ID，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. 由于您在创建角色时没有选择 Amazon Cognito 作为可信实体，因此您现在必须手动编辑该角色的信任关系。在 IAM 控制台的导航窗格中选择 Roles（角色），然后选择您创建的新角色。
4. 选择 Trust relationships (信任关系) 选项卡。
5. 选择 Edit trust policy（编辑信任策略）。
6. 将以下策略声明粘贴到 Edit trust policy（编辑信任策略）中，替换任何现有文本：

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "cognito-idp.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }
```

7. 选择 Update policy（更新策略）。
8. 记下角色 ARN。您在创建导入作业时需要此 ARN。

创建用户导入 CSV 文件

您必须先创建逗号分隔值（CSV，Comma-Separated Value）文件，在其中包含要导入的用户及其属性，然后才能将现有用户导入用户群体中。从用户群体中，您可以检索其标头反映了您的用户群体的属性架构的用户导入文件。然后，您可以插入符合 设置 CSV 文件的格式 (p. 187) 中的格式要求的用户信息。

下载 CSV 文件标头（控制台）

使用以下步骤下载 CSV 标头文件。

下载 CSV 文件标头

1. 转到 Amazon Cognito 控制台。系统可能会提示您输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 Users（用户）选项卡。
5. 在 Import users（导入用户）部分中，选择 Create an import job（创建导入作业）。
6. 在 Upload CSV（上传 CSV）下，选择 template.csv 链接并下载 CSV 文件。

下载 CSV 文件标头（AWS CLI）

要获取一系列正确的标头，请运行以下 CLI 命令，其中，*USER_POOL_ID* 是要向其中导入用户的用户池的用户池标识符。

```
aws cognito-idp get-csv-header --user-pool-id "USER_POOL_ID"
```

示例响应:

```
{
    "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"
}
```

设置 CSV 文件的格式

下载的用户导入 CSV 标头文件类似于以下字符串。它还包括您已添加到用户群体的所有自定义属性。

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,picture,we
```

编辑 CSV 文件，以使文件包含此标头和用户的属性值，并根据以下规则设置格式：

> **Note**
>
> 有关属性值（如电话号码的正确格式）的更多信息，请参阅用户池属性 (p. 243)。

- 文件的第一行是已下载的包含用户属性名称的标头行。
- CSV 文件中列的顺序不重要。
- 第一行之后的每一行都包含用户的属性值。
- 标头中的所有列都必须存在，但您不需要在每一列中提供值。
- 以下属性为必需属性：
  - cognito:username
  - cognito:mfa_enabled
  - email_verified 或 phone_number_verified
    - 每个用户至少有一个自动验证属性必须为 `true`。自动验证的属性是新用户加入您的用户群体时，Amazon Cognito 自动向其发送验证码的电子邮件地址或电话号码。
    - 用户池必须至少有一个自动验证属性，要么是 email_verified，要么是 phone_number_verified。如果用户池没有自动验证属性，则导入任务不会启动。
    - 如果用户池只有一个自动验证属性，则该属性必须针对每个用户进行验证。例如，如果用户池只有 phone_number 为自动验证属性，则每个用户的 phone_number_verified 值都必须为 `true`。

Note

对于重置其密码的用户，用户必须拥有经过验证的电子邮件或电话号码。Amazon Cognito 将包含重置密码代码的消息发送到 CSV 文件中指定的电子邮件或电话号码。如果将消息发送到电话号码，则通过 SMS 消息发送。有关更多信息，请参阅在注册时验证联系人信息 (p. 169)。

  - email（如果 email_verified 为 `true`）
  - phone_number（如果 phone_number_verified 为 `true`）
  - 创建用户池时标记为必需的所有属性
- 字符串式的属性值不 应该用引号括起来。
- 如果属性值包含逗号，则您必须在逗号前使用反斜杠 (\)。这是因为 CSV 文件中的字段使用逗号分隔。
- CSV 文件内容应采用不含字节顺序标记的 UTF-8 格式。
- cognito:username 字段是必填项，并且在用户池中必须是唯一的。它可以是任何 Unicode 字符串。但是，它不能包含空格或制表符。
- birthdate 值（如果存在）必须采用 *mm/dd/yyyy* 格式。也就是说，如果生日日期为 1985 年 2 月 1 日，则必须编码为 **02/01/1985**。
- cognito:mfa_enabled 字段为必填字段。如果您已将用户池设置为需要进行多重验证 (MFA)，则所有用户的此字段都必须为 `true`。如果您已将 MFA 设置为关闭，则所有用户的此字段都必须为 `false`。如果您已将 MFA 设置为可选，则此字段可以是 `true` 或 `false`，但不能为空。
- 最大长度为 16000 个字符。
- CSV 文件的最大大小为 100MB。
- 文件中的最大行（用户）数为 500000。此最大值不包括标题行。
- updated_at 字段值应为纪元时间（用秒表示），例如：**1471453471**。
- 属性值中的所有前导空格或尾部空格均应去除。

以下列表是没有自定义属性的用户群体的 CSV 导入文件示例。您的用户群体架构可能与此示例有所不同。在这种情况下，您必须在从用户群体下载的 CSV 模板中提供测试值。

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,picture,we
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
```

创建并运行 Amazon Cognito 用户池导入任务

本部分介绍如何使用 Amazon Cognito 控制台和 AWS Command Line Interface（AWS CLI）创建并运行用户群体导入作业。

主题

- 从 CSV 文件导入用户（控制台）(p. 188)
- 导入用户（AWS CLI）(p. 189)

从 CSV 文件导入用户（控制台）

以下过程介绍了如何从 CSV 文件导入用户。

从 CSV 文件导入用户（控制台）

1. 转到 Amazon Cognito 控制台。系统可能会提示您输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。

4. 选择 Users（用户）选项卡。
5. 在 Import users（导入用户）部分中，选择 Create an import job（创建导入作业）。
6. 在 Create import job（创建导入作业）页面上，输入 Job name（作业名称）。
7. 选择 Create a new IAM role（创建新的 IAM 角色）或者 Use an existing IAM role（使用现有 IAM 角色）。

   a. 如果您选择 Create a new IAM role（创建新的 IAM 角色），请输入新角色的名称。Amazon Cognito 将自动创建具有正确权限和信任关系的角色。创建导入作业的 IAM 主体必须具有创建 IAM 角色的权限。

   b. 如果您选择 Use an existing IAM role（使用现有 IAM 角色），请从 IAM role selection（IAM 角色选择）下的列表中选择一个角色。此角色必须具有 创建 CloudWatch Logs IAM 角色 (p. 185) 中所述的权限和信任策略。
8. 选择 Create job（创建作业）可提交作业，但稍后再启动。选择 Create and start job（创建并启动作业）可提交您的作业并立即启动。
9. 如果您创建了作业但未启动作业，则可以稍后再启动。在 Users（用户）选项卡下的 Import users（导入用户）中，选择导入作业，然后选择 Start（开始）。您也可以从 AWS SDK 提交 StartUserImportJob API 请求。
10. 在 Users（用户）选项卡下的 Import users（导入用户）中，监控用户导入作业的进度。如果您的作业不成功，则可以选择 Status（状态）值。如需更多详细信息，请选择 View the CloudWatch logs for more details（查看 CloudWatch Logs 以获取详细信息），在 CloudWatch Logs 控制台中查看任意问题。

导入用户（AWS CLI）

以下 CLI 命令可用于将用户导入到用户池：

- `create-user-import-job`
- `get-csv-header`
- `describe-user-import-job`
- `list-user-import-jobs`
- `start-user-import-job`
- `stop-user-import-job`

要获取这些命令的命令行选项列表，请使用 `help` 命令行选项。例如：

```
aws cognito-idp get-csv-header help
```

创建用户导入任务

创建 CSV 文件后，请通过运行以下 CLI 命令创建用户导入作业，其中，*JOB_NAME* 是您为作业选择的名称，*USER_POOL_ID* 是将新用户添加到的用户群体的用户群体 ID，*ROLE_ARN* 是您在 创建 CloudWatch Logs IAM 角色 (p. 185) 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"
```

响应中返回的 *PRE_SIGNED_URL* 在 15 分钟内有效。在此之后，它将过期，而您必须创建新的用户导入任务以获取新的 URL。

Example 示例响应:

```
{
    "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
    }
}
```

用户导入任务的状态值

在对用户导入命令的响应中，您将看到以下 Status 值当中的其中一个值：

- Created – 任务已创建但未启动。
- Pending – 转换状态。您已启动任务，但它尚未开始导入用户。
- InProgress – 任务已启动，正在导入用户。
- Stopping – 您已停止任务，但任务尚未停止导入用户。
- Stopped – 您已停止任务，且任务已停止导入用户。
- Succeeded – 任务已成功完成。
- Failed – 任务因错误而停止。
- Expired – 您创建了一个任务，但未在 24-48 小时内启动任务。与任务关联的所有数据已删除，且任务无法启动。

上传 CSV 文件

使用以下 curl 命令将包含用户数据的 CSV 文件上传到您从 create-user-import-job 命令的响应中获取的预签名 URL。

```
curl -v -T "PATH_TO_CSV_FILE" -H "x-amz-server-side-encryption:aws:kms" "PRE_SIGNED_URL"
```

在此命令的输出中，查找 "We are completely uploaded and fine" 这一短语。此短语表示文件已成功上传。

描述用户导入任务

要获取用户导入任务的描述，请使用以下命令，其中，*USER_POOL_ID* 是用户池 ID，*JOB_ID* 是创建用户导入任务时返回的任务 ID。

```
aws cognito-idp describe-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

Example 示例响应:

```
{
    "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
    }
}
```

在上一示例输出中，*PRE_SIGNED_URL* 是您上传 CSV 文件的目标 URL。*ROLE_ARN* 是您创建角色时收到的 CloudWatch Logs 角色 ARN。

列出用户导入任务

要列出用户导入任务，请使用以下命令：

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 2
```

Example 示例响应:

```
{
    "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"
}
```

任务按创建日期 (从近到远) 排列。第二项任务之后的 *PAGINATION_TOKEN* 字符串表示此列表命令还有其他结果。要列出更多结果，请使用 `--pagination-token` 选项，如下所示：

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 10 --
pagination-token "PAGINATION_TOKEN"
```

启动用户导入任务

要启动用户导入任务，请使用以下命令：

```
aws cognito-idp start-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

每个账户每次只能有一个导入任务处于活动状态。

Example 示例响应:

```
{
    "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
    }
}
```

停止用户导入任务

要停止正在进行的用户导入任务，请使用以下命令。停止任务后，无法重新启动该任务。

```
aws cognito-idp stop-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

Example 示例响应:

```
{
    "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 控制台中查看用户池导入结果

您可以在 Amazon CloudWatch 控制台中查看导入任务的结果。

主题

- 查看结果 (p. 192)
- 解析结果 (p. 193)

查看结果

以下步骤介绍了如何查看用户池导入结果。

查看用户池导入结果的步骤

1. 登录AWS Management Console并打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 选择 Logs (日志)。
3. 为用户池导入任务选择日志组。日志组名称的形式为 `/aws/cognito/userpools/USER_POOL_ID/USER_POOL_NAME`。
4. 为刚运行的用户导入任务选择日志。日志名称的形式为 *JOB_ID*/*JOB_NAME*。日志中的结果按行号引用用户。日志中不会写入用户数据。对于每个用户，都将出现类似于以下内容的行：

   - `[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).`

解析结果

已成功导入的用户的状态设置为“PasswordReset”。

在以下情况下，将不会导入用户，但导入任务将继续：

- 自动验证属性未设置为 `true`。
- 用户数据与架构不匹配。
- 由于内部错误，无法导入用户。

在以下情况下，导入任务将失败：

- Amazon CloudWatch Logs 角色无法被承担，因为没有正确的访问策略，或已删除。
- 用户池已删除。
- Amazon Cognito 无法解析 .csv 文件。

要求导入的用户重置密码

每个导入的用户第一次登录时，系统都会要求其输入新密码。以下过程描述了您导入 CSV 文件后，本地用户在自定义应用程序中的用户体验。如果您的用户使用托管 UI 登录，Amazon Cognito 会在其首次登录时提示他们设置新密码。

要求导入的用户重置密码

1. 用户尝试登录。应用程序调用 `InitiateAuth` 来提交其用户名和密码。
2. 启用了 `PreventUserExistenceErrors` 时，Amazon Cognito 返回 `NotAuthorizedException`。否则返回 `PasswordResetRequiredException`。
3. 您的应用程序发出 `ForgotPassword` API 请求并重置用户的密码。

   a. 应用程序在 `ForgotPassword` API 请求中提交用户名。
   b. Amazon Cognito 向经过验证的电子邮件或电话发送代码。目标取决于您在 CSV 文件中为 `email_verified` 和 `phone_number_verified` 提供的值。对 `ForgotPassword` 请求的响应指明了代码的目标。

      Note

      必须将您的用户群体配置为验证电子邮件或电话号码。有关更多信息，请参阅注册并确认用户账户 (p. 168)。

c. 您的应用程序向用户显示一条消息，以检查发送代码的位置，并提示用户输入代码和新密码。
d. 用户在应用程序中输入代码和新密码。
e. 应用程序在 `ConfirmForgotPassword` API 请求中提交代码和新密码。
f. 您的应用程序重定向用户以进行登录。

Amazon Cognito 用户池的电子邮件设置

用户池客户端应用程序中的某些事件可能导致 Amazon Cognito 向用户发送电子邮件。例如，如果您将用户池配置为需要电子邮件验证，则当用户在应用程序中注册新账户或重置其密码时，Amazon Cognito 会发送电子邮件。根据发起电子邮件递送的操作，电子邮件将包含验证码或临时密码。

为处理电子邮件递送，您可以使用以下任一选项：

- 默认电子邮件功能 (p. 194)，内置于 Amazon Cognito 服务中。
- 您的 Amazon Simple Email Service (Amazon SES) 配置 (p. 194)。

这些设置是可逆的。您可以更新您的用户池以在这些设置之间切换。

默认电子邮件功能

Amazon Cognito 可以使用其默认电子邮件功能为您处理电子邮件递送。当您使用默认选项时，Amazon Cognito 会限制您的用户池每天可以发送的电子邮件数量。有关服务限制的更多信息，请参阅 Amazon Cognito 中的限额 (p. 539)。对于典型的生产环境，默认的电子邮件限制低于所需的递送量。要启用更高的送达量，您可使用您的 Amazon SES 电子邮件配置。

当您使用默认功能时，您将使用 AWS 管理的 Amazon SES 资源来发送电子邮件消息。Amazon SES 将返回查无此人的邮件的电子邮件地址添加到账户级禁止列表或全局禁止列表。如果无法送达的电子邮件地址稍后变为可送达，则当您的用户池配置为使用默认功能时，您无法控制将其从禁止列表中删除的行为。电子邮件地址可以无限期地保留在 AWS 管理的禁止列表上。要管理无法送达的电子邮件地址，请将 Amazon SES 电子邮件配置与账户级别的禁止列表一起使用，如下一节中所述。

使用默认电子邮件配置时，您可以使用以下任一电子邮件地址作为 FROM 地址：

- 默认电子邮件地址 no-reply@verificationemail.com。
- 自定义电子邮件地址。在可以使用您自己的电子邮件地址之前，您必须向 Amazon SES 验证此地址，并且向 Amazon Cognito 授予使用此地址的权限。

Amazon SES 电子邮件配置

您的应用程序需要的递送量可能高于默认选项所提供的递送量。要增加可能的递送量，请将您的 Amazon SES 资源与用户池一起使用来向用户发送电子邮件。当您使用自己的 Amazon SES 配置发送电子邮件消息时，您还可以监控您的电子邮件发送活动。

在可以使用您的 Amazon SES 配置之前，您必须向 Amazon SES 验证一个或多个电子邮件地址或域。将经验证的电子邮件地址或已验证域的地址，用作分配给用户池的 FROM 电子邮件地址。当 Amazon Cognito 向您的用户发送电子邮件时，它会以您的名义调用 Amazon SES 并使用您的电子邮件地址。

当您使用 Amazon SES 配置时，以下条件适用：

- 您的用户池的电子邮件传送限制与适用于您的 AWS 账户 中的 Amazon SES 验证电子邮件地址的限制相同。

- 您可以使用 Amazon SES 中覆盖全局禁止列表的账户级禁止列表来管理发送到无法送达的电子邮件地址的邮件。当您使用账户级禁止列表时，退回的电子邮件消息会影响您的账户作为发件人的声誉。有关更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的使用 Amazon SES 账户级禁止列表。

Amazon SES 电子邮件配置区域

当您选择 AWS 区域而其中包含要用于 Amazon Cognito 电子邮件的 Amazon SES 资源时，您可以选择创建用户池时使用的相同区域。对于部分区域中的 Amazon Cognito 用户池，您还可以使用以下备用区域中的 Amazon SES 资源：美国东部（弗吉尼亚北部）、美国西部（俄勒冈）或欧洲（爱尔兰）。

为了使您的电子邮件操作更快、更可靠，请使用创建用户池所用的 AWS 区域。Amazon Cognito 中支持跨区域 Amazon SES 配置，这为您创建的用户池资源提供了连续性，这些资源是为了遵守服务启动时的 Amazon Cognito 要求而创建的。您在此期间创建的用户池资源只能使用有限数量 AWS 区域中的 Amazon SES 资源。

> Note
>
> 在AWS Management Console中，切换到新的 Amazon Cognito 控制台体验后，您只能使用同一区域中的 Amazon SES 资源。

如果您使用 AWS Command Line Interface、API 或 AWS CloudFormation 创建 Amazon Cognito 用户池资源，则用户池会使用 EmailConfigurationType 对象的 `SourceArn` 参数为您的用户池指定的 Amazon SES 身份，发送电子邮件。Amazon SES 身份必须位于支持的 AWS 区域中。如果您的 `EmailSendingAccount` 是 `COGNITO_DEFAULT` 而且您没有指定 `SourceArn` 参数，则 Amazon Cognito 使用您创建用户池所在区域中的资源，从 `no-reply@verificationemail.com` 发送电子邮件。

下表显示了您可以将 Amazon SES 身份用于 Amazon Cognito 的 AWS 区域。

Amazon Cognito 用户池 AWS 区域	Amazon Simple Email Service 支持的 AWS 区域
美国东部（弗吉尼亚州北部）	美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
美国东部（俄亥俄州）	美国东部（俄亥俄）、美国东部（弗吉尼亚北部）、欧洲（爱尔兰）
美国西部（北加利福尼亚）	美国西部（北加利福尼亚）
美国西部（俄勒冈州）	美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
加拿大（中部）	加拿大（中部）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
亚太地区（东京）	亚太地区（东京）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
亚太地区（首尔）	亚太地区（首尔）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
亚太地区（孟买）	亚太地区（孟买）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
亚太地区（新加坡）	亚太地区（新加坡）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
亚太地区（悉尼）	亚太地区（悉尼）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）

Amazon Cognito 用户池 AWS 区域	Amazon Simple Email Service 支持的 AWS 区域
欧洲（爱尔兰）	美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
欧洲（伦敦）	欧洲（伦敦）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
欧洲（巴黎）	欧洲（巴黎）
欧洲（法兰克福）	欧洲（法兰克福）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
欧洲（斯德哥尔摩）	欧洲（斯德哥尔摩）
欧洲（米兰）	Europe (Milan)
中东（巴林）	中东（巴林）
南美洲（圣保罗）	南美洲（圣保罗）

为您的用户池配置电子邮件

完成以下步骤为用户池配置电子邮件设置。根据您要使用的设置，您可能需要 Amazon SES、AWS Identity and Access Management（IAM）和 Amazon Cognito 中的 IAM 权限。

> Note
>
> 您在这些步骤中创建的资源无法跨 AWS 账户进行共享。例如，您不能为一个账户中的用户池配置用户池，然后将其用于另一个账户中的 Amazon SES 电子邮件地址。如果您在多个账户中使用 Amazon Cognito，请为每个账户中重复这些步骤。

步骤 1：使用 Amazon SES 验证电子邮件地址或域

在配置您的用户池之前，如果您要执行以下任一操作，则必须使用 Amazon SES 验证一个或多个电子邮件地址或域：

- 使用您自己的电子邮件地址作为 FROM 地址
- 使用您的 Amazon SES 配置处理电子邮件送达

通过验证您的电子邮件地址或域，您确认您拥有该电子邮件地址，这有助于防止未经授权的使用。

有关使用 Amazon SES 验证电子邮件地址的更多信息，请参阅 Amazon Simple Email Service 开发人员指南中的验证电子邮件地址。有关使用 Amazon SES 验证域的信息，请参阅验证域。

步骤 2：将您的账户移出 Amazon SES 沙盒

如果您使用的是默认 Amazon Cognito 电子邮件功能，则跳过此步骤。

当您首次在任意 AWS 区域中使用 Amazon SES 时，它会将您的 AWS 账户放置在该区域的 Amazon SES 沙盒中。Amazon SES 使用沙盒防止欺诈和滥用。如果您使用您的 Amazon SES 配置来处理电子邮件送达，则必须将您的 AWS 账户移出沙盒，然后 Amazon Cognito 才能向用户发送电子邮件。

在沙盒中，Amazon SES 会对您可以发送的电子邮件数量和可以发送电子邮件的位置施加限制。您可以仅向已通过 Amazon SES 验证的地址和域发送电子邮件，也可以将其发送到 Amazon SES 邮箱模拟器地址。当您的 AWS 账户保留在沙盒中时，请勿对生产环境中的应用程序使用您的 Amazon SES 配置。在这种情况下，Amazon Cognito 无法将邮件发送到您用户的电子邮件地址。

要将您的 AWS 账户移出沙盒，请参阅《Amazon Simple Email Service 开发人员指南》中的移出 Amazon SES 沙盒。

步骤 3：授予 Amazon Cognito 电子邮件权限

您可能需要向 Amazon Cognito 授予特定权限，然后它才能向您的用户发送电子邮件。您授予的权限以及用于授予权限的过程取决于您使用的是默认电子邮件功能还是您的 Amazon SES 配置。

授予权限以使用默认的电子邮件功能

只有当您使用的是默认 Amazon Cognito 电子邮件功能时才可跳过此步骤。

如果将您的用户池配置为使用默认 Amazon Cognito 的电子邮件功能，您可以使用以下任一地址作为 FROM 地址，Amazon Cognito 会通过该地址向您的用户发送电子邮件：

- 默认地址
- Amazon SES 中的自定义地址必须是已验证的电子邮件地址，或已验证域中的电子邮件地址。

如果您使用自定义地址，Amazon Cognito 需要额外权限，以便使用此地址向您的用户发送电子邮件。这些权限通过发送授权策略授予，并在 Amazon SES 中会将此策略附加到地址或域。如果您使用 Amazon Cognito 控制台向您的用户池添加自定义地址，此策略会自动附加到 Amazon SES 验证的电子邮件地址。然而，如果您在此控制台外部配置您的用户池（例如，使用 AWS CLI 或 Amazon Cognito API），则必须使用 Amazon SES 控制台或 PutIdentityPolicy API 附加此策略。

> **Note**
>
> 您只能使用 AWS CLI 或 Amazon Cognito API 在已验证的域中配置 FORM 地址。

发送授权策略根据使用 Amazon Cognito 调用 Amazon SES 的账户资源，来允许或拒绝访问。有关基于资源的策略的更多信息，请参阅 IAM 用户指南。在 Amazon SES 开发人员指南中可以找到基于资源的策略示例。

Example 发送授权策略

以下示例发送授权策略授予 Amazon Cognito 使用经 Amazon SES 验证的身份的有限能力。Amazon Cognito 在代表 `aws:SourceArn` 中的用户池和 `aws:SourceAccount` 条件中的账户时才能发送电子邮件。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "stmnt1234567891234",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "email.cognito-idp.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "aws:SourceArn": "<your user pool ARN>"
                }
```

```
            }
        }
    ]
}
```

在此示例中，“Sid”值为唯一标识语句的任意字符串。

有关策略语法的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的 Amazon SES 发送授权策略。

有关更多示例，请参阅《Amazon Simple Email Service 开发人员指南》中的 Amazon SES 发送授权策略示例。

Example

授予权限以使用您的 Amazon SES 配置

如果您将用户池配置为使用您的 Amazon SES 配置，Amazon Cognito 在向用户发送电子邮件时，需要额外的权限代表您调用 Amazon SES。此授权将通过 IAM 服务授予。

当您使用此选项配置用户池时，Amazon Cognito 将创建一个服务相关角色，这是您 AWS 账户中的一种 IAM 角色类型。此角色包含允许 Amazon Cognito 访问 Amazon SES 并使用您的地址发送电子邮件的权限。

在 Amazon Cognito 可以创建此角色之前，您用来设置用户池的 IAM 权限必须包含 `iam:CreateServiceLinkedRole` 操作。有关使用 IAM 中权限的更多信息，请参阅《IAM 用户指南》中的 AWS 资源的访问权限管理。

有关 Amazon Cognito 创建的服务相关角色的更多信息，请参阅对 Amazon Cognito 使用服务相关角色 (p. 504)。

步骤 4：配置用户池

如果您要将您的用户池配置为使用以下内容，请完成以下步骤：

- 自定义 FROM 地址（显示为电子邮件发件人）。
- 自定义 REPLY-TO 地址，用于接收您的用户发送到您的 FROM 地址的邮件
- Amazon SES 配置

如果您要使用默认 Amazon Cognito 电子邮件功能和地址，则跳过此过程。

配置用户池以使用自定义电子邮件地址

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 Messaging（消息收发）选项卡，查找 Email configuration（电子邮件配置），选择 Edit（编辑）。
5. 在 Edit email configuration（编辑电子邮件配置）页面中，选择 Send email from Amazon SES（使用 Amazon SES 发送电子邮件）或 Send email with Amazon Cognito（使用 Amazon Cognito 发送电子邮件）。仅当您选择 Send email from Amazon SES（使用 Amazon SES 发送电子邮件）时，您才可以自定义 SES Region（SES 区域）、Configuration Set（配置集） 和 FROM sender name（FROM 发件人姓名）。
6. 要使用自定义 FROM 地址，请完成以下步骤：

    a. 在 SES region（SES 区域）下，选择包含验证的电子邮件地址的区域。

    b. 在 FROM email address（FROM 电子邮件地址）下，选择您的电子邮件地址。使用您已通过 Amazon SES 验证的电子邮件地址。

c. （可选）在 Configuration set（配置集）下，选择 Amazon SES 使用的配置集。进行此更改并保存可创建服务相关角色。

d. （可选）在 FROM sender address（FROM 发件人地址）下，输入电子邮件地址。您可以仅提供电子邮件地址，也可以同时提供电子邮件地址和格式为 `Jane Doe <janedoe@example.com>` 的易记名称。

e. （可选）在 REPLY-TO email address（REPLY-TO 电子邮箱地址）下，输入要用来接收用户发送到您的 FROM 地址的邮件的电子邮件地址。

7. 选择 Save changes（保存更改）。

相关主题

- 自定义电子邮件验证消息 (p. 255)
- 自定义用户邀请消息 (p. 256)

Amazon Cognito 用户池的短信设置

您的用户池的某些 Amazon Cognito 事件可能会导致 Amazon Cognito 向您的用户发送短信。例如，如果您将用户池配置为需要电话验证，则当用户在应用程序中注册新账户或重置其密码时，Amazon Cognito 会发送短信。根据发起短信的操作，短信中将包含验证码、临时密码或欢迎消息。

Amazon Cognito 使用 Amazon Simple Notification Service (Amazon SNS) 传送短信。如果这是您首次通过 Amazon Cognito 或 Amazon SNS 发送短信，Amazon SNS 会将您放在沙盒环境。在沙盒环境中，您可以对应用程序的 SMS 文本消息进行测试。在沙盒中，只能将消息发送给经过验证的电话号码。

Note

由于全球范围内未经请求的短信流量巨大，一些政府在短信发送者和接收者之间设置了障碍。当您使用短信进行 MFA 和用户更新时，必须采取额外的步骤来确保您的短信已送达。您还必须监控您的用户可能居住的国家/地区的短信相关法规，并保持短信配置处于最新状态。有关更多信息，请参阅《Amazon Simple Notification Service 开发人员指南》中的移动文本消息（SMS）。
使用短信对用户进行身份验证和验证不是安全最佳做法。电话号码可能会变更所有者，而可能无法可靠地代表您拥有的 用户 MFA 要素。而是在应用程序中或使用第三方 IdP 实现 TOTP MFA。您还可以使用自定义身份验证质询 Lambda 触发器 (p. 134)创建其他自定义身份验证要素。

首次在 Amazon Cognito 用户池中设置 SMS 消息

Amazon Cognito 使用 Amazon SNS 向您的用户池发送短信。您还可以使用自定义 SMS 发件人 Lambda 触发器 (p. 160)，通过自己的资源发送 SMS 消息。在特定 AWS 区域中首次设置 Amazon SNS 发送 SMS 短信时，Amazon SNS 会将您的 AWS 账户放在该区域的 SMS 沙盒中。Amazon SNS 使用沙盒防止欺诈和滥用，并满足合规性要求。当您的 AWS 账户位于沙盒中时，Amazon SNS 施加了一些限制。例如，您最多可以向 10 个已通过 Amazon SNS 验证的电话号码发送短信。当您的 AWS 账户仍保留在沙盒中时，请勿对生产环境中的应用程序使用您的 Amazon SNS 配置。当您位于沙盒中时，Amazon Cognito 无法向用户的电话号码发送消息。

向用户池用户发送 SMS 文本消息

1. 准备一个 IAM 角色，Amazon Cognito 可以使用该角色通过 Amazon SNS 发送 SMS 消息 (p. 200)
2. 为 Amazon SNS SMS 消息选择 AWS 区域 (p. 200)
3. 获取源身份以将 SMS 消息发送到美国电话号码 (p. 201)
4. 确认您位于 SMS 沙盒中 (p. 202)
5. 将您的账户移出 Amazon SNS 沙盒 (p. 202)
6. 在 Amazon SNS 中验证 Amazon Cognito 的电话号码 (p. 202)
7. 在 Amazon Cognito 中完成用户群体设置 (p. 203)

准备一个 IAM 角色，Amazon Cognito 可以使用该角色通过 Amazon SNS 发送 SMS 消息

当您从用户群体发送 SMS 消息时，Amazon Cognito 将代入您的账户中的 IAM 角色。Amazon Cognito 使用分配给该角色的 `sns:Publish` 权限向您的用户发送 SMS 消息。在 Amazon Cognito 控制台中，您可以从 SMS 下用户群体的 Messaging（消息收发）选项卡设置 IAM role selection（IAM 角色选择），或者在用户群体创建向导过程中进行此选择。

以下示例 IAM 角色信任策略授予 Amazon Cognito 用户群体有限代入角色的能力。Amazon Cognito 只能在代表 `aws:SourceArn` 条件中的用户群体和 `aws:SourceAccount` 条件中的 AWS 账户时代入角色。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "cognito-idp.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "<your account number>"
            },
            "ArnLike": {
                "aws:SourceArn": "<your user pool ARN>"
            }
        }
    }]
}
```

您可以在 `aws:SourceArn` 条件的值中指定准确的用户群体 ARN 或通配符 ARN。在 AWS Management Console 中或使用 DescribeUserPool API 请求查找您的用户群体的 ARN。

有关 IAM 角色和信任策略的更多信息，请参阅《AWS Identity and Access Management 用户指南》中的角色术语和概念。

为 Amazon SNS SMS 消息选择 AWS 区域

在某些 AWS 区域，您可以选择包含要用于 Amazon Cognito SMS 消息的 Amazon SNS 资源的区域。除亚太地区（首尔）以外，在提供 Amazon Cognito 的任意 AWS 区域，您可以在创建用户池的 AWS 区域中使用 Amazon SNS 资源。在有多个区域可供选择时，为了使您的 SMS 消息收发更快且更可靠，请使用与您的用户池位于相同区域中的 Amazon SNS 资源。

> **Note**
>
> 在 AWS Management Console 中，您只能在切换到新的 Amazon Cognito 控制台体验后更改 SMS 资源的区域。

在新建用户池向导的 Configure message delivery（配置消息传输）步骤中，为 SMS 资源选择区域。您还可以在现有用户池的 Messaging（消息收发）选项卡中，在 SMS 下选择 Edit（编辑）。

在启动时，对于一些 AWS 区域，Amazon Cognito 在备用区域中使用 Amazon SNS 资源发送 SMS 消息。要设置首选区域，请使用您用户池 SmsConfigurationType 对象的 `SnsRegion` 参数。当您通过下表以编程方式在 Amazon Cognito 区域中创建 Amazon Cognito 用户池资源并且您不提供 `SnsRegion` 参数时，您的用户池可以使用旧 Amazon SNS 区域中的 Amazon SNS 资源发送 SMS 消息。

亚太地区（首尔）AWS 区域中的 Amazon Cognito 用户池必须使用您在亚太地区（东京）区域中的 Amazon SNS 配置。

Amazon SNS 将所有新账户的支出配额设定为每月 1.00 美元 (USD)。您可能已经提高了与 Amazon Cognito 一起使用的 AWS 区域 中的支出限额。在您更改 Amazon SNS SMS 消息的 AWS 区域 之前，请在 AWS 支持中心创建一个配额增加案例，以增加您在新区域中的限额。有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南中的请求对 Amazon SNS 提升您的每月 SMS 支出配额。

您可以使用相应 Amazon SNS 区域中的 Amazon SNS 资源为下表中的任何 Amazon Cognito 区域发送 SMS 消息。

Amazon Cognito 区域	Amazon SNS 区域
美国东部（俄亥俄州）	美国东部（俄亥俄）、美国东部（弗吉尼亚北部）
亚太地区（孟买）	亚太地区（孟买）、亚太地区（新加坡）
加拿大（中部）	加拿大（中部）、美国东部（弗吉尼亚北部）
欧洲地区（法兰克福）	欧洲（法兰克福）、欧洲（爱尔兰）
欧洲地区（伦敦）	欧洲（伦敦）、欧洲（爱尔兰）
亚太地区（首尔）	亚太地区（东京）
美国东部（弗吉尼亚州北部）	美国东部（弗吉尼亚州北部）
美国西部（北加利福尼亚）	美国西部（北加利福尼亚）
美国西部（俄勒冈州）	美国西部（俄勒冈州）
亚太地区（新加坡）	亚太地区（新加坡）
亚太地区（悉尼）	亚太地区（悉尼）
亚太地区（东京）	亚太地区（东京）
欧洲地区（爱尔兰）	欧洲地区（爱尔兰）
欧洲地区（巴黎）	欧洲地区（巴黎）
欧洲地区（斯德哥尔摩）	欧洲地区（斯德哥尔摩）
欧洲地区（米兰）	Europe (Milan)
中东（巴林）	中东（巴林）
南美洲（圣保罗）	南美洲（圣保罗）

获取源身份以将 SMS 消息发送到美国电话号码

无论您是构建 SMS 沙盒测试环境还是生产环境，如果您计划向美国电话号码发送短信，则必须获取源身份。

自 2021 年 6 月 1 日起，美国运营商要求提供源身份才能向美国电话号码发送短信。如果您没有源身份，则必须获取一个。请参阅《Amazon Pinpoint 用户指南》中的申请号码了解如何获取源身份。

如果您在以下 AWS 区域开展运营，则必须提交 AWS Support 票证来获取源身份。有关说明，请参阅《Amazon Simple Notification Service 开发人员指南》中的针对 SMS 消息收发请求支持。

- 美国东部（俄亥俄州）
- 欧洲地区（斯德哥尔摩）
- 欧洲地区（巴黎）

- Europe (Milan)
- Middle East (Bahrain)
- 南美洲（圣保罗）
- 美国西部（北加利福尼亚）

当您在同一个 AWS 区域 中有多个源身份时，Amazon SNS 按以下优先顺序选择源身份类型：短代码、10DLC、免费电话号码。无法更改此优先级。有关更多信息，请参阅 Amazon SNS 常见问题。

确认您位于 SMS 沙盒中

按照以下过程确认您是否在 SMS 沙盒中。对于您拥有生产 Amazon Cognito 用户池的每个 AWS 区域，重复此步骤。

在 Amazon Cognito 控制台中查看 SMS 沙盒状态

确认您位于 SMS 沙盒中

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 Messaging（消息收发）选项卡。
5. 在 SMS configuration（SMS 配置）部分，展开 Move to Amazon SNS production environment（迁移到 Amazon SNS 生产环境）。如果您的账户位于 SMS 沙盒中，您将看到以下消息：

   ```
   You are currently in the SMS Sandbox and cannot send SMS messages to
   unverified numbers.
   ```

   如果您没有看到此消息，则表明有人已经在您的账户中设置了 SMS 消息。跳至在 Amazon Cognito 中完成用户群体设置 (p. 203)。
6. 选择消息中的 Amazon SNS 链接。这将在新选项卡中打开 Amazon SNS 控制台。
7. 验证您是否位于沙盒环境中。控制台消息指示您的沙盒状态和 AWS 区域，如下所示：

   ```
   This account is in the SMS sandbox in US East (N. Virginia).
   ```

将您的账户移出 Amazon SNS 沙盒

如果正在测试应用程序，并且只需向管理员可以验证的电话号码发送 SMS 消息，请跳过此步骤。

要在生产环境中使用您的应用程序，请将账户移出 SMS 沙盒并放入生产环境。在包含您希望 Amazon Cognito 使用的 Amazon SNS 资源的 AWS 区域中配置源身份之后，您可以在 AWS 账户仍然位于 SMS 沙盒中时测试美国号码。当您的 Amazon SNS 环境投入生产时，您无需在 Amazon SNS 中验证用户电话号码即可向用户发送 SMS 消息。

有关详细说明，请参阅《Amazon Simple Notification Service 开发人员指南》中的移出 SMS 沙盒。

在 Amazon SNS 中验证 Amazon Cognito 的电话号码

如果您已将账户移出 SMS 沙盒，则跳过此步骤。

当您位于 SMS 沙盒中时，您可以使用 Amazon SNS 向任何已验证的号码发送消息。

要验证电话号码，请执行以下操作：

1. 在 Amazon SNS 控制台的 Text messaging (SMS)（文本消息 (SMS)）部分，添加 Sandbox destination phone number（沙盒目标电话号码）。

2. 在您提供的电话号码上接收带有密码的 SMS 消息。
3. 在 Amazon SNS 控制台上，输入 SMS 消息中的 Verification code（验证代码）。

有关详细说明，请参阅《Amazon Simple Notification Service 开发人员指南》中的在 SMS 沙盒中添加并验证电话号码。

> **Note**
>
> 当您在 SMS 沙盒中时，Amazon SNS 限制可以验证的目标电话号码的数量。请参阅《Amazon Simple Notification Service 开发人员指南》中的 SMS 沙盒。

在 Amazon Cognito 中完成用户群体设置

返回您在其中创建 (p. 30)或者编辑 (p. 169)用户池的浏览器选项卡。完成过程。当您成功将 SMS 配置添加到用户群体后，Amazon Cognito 会向内部电话号码发送测试消息，以验证您的配置有效。Amazon SNS 会对每条测试 SMS 消息收费。

将令牌与用户池结合使用

通过令牌验证用户的身份并授予对资源的访问权限。令牌中的声明是有关您的用户的信息。ID 令牌包含有关用户身份的声明，例如他们的用户名、姓氏和电子邮件地址。访问令牌包含类似于 `scope` 的声明，经过身份验证的用户可以使用它们访问第三方 API、Amazon Cognito 用户自助 API 操作和 UserInfo 终端节点 (p. 566)。访问令牌和 ID 令牌均包含 `cognito:groups` 声明，其中包含用户在用户群体中的组成员资格。

Amazon Cognito 还有令牌，您可以使用它来获取新的令牌或撤销现有令牌。刷新令牌 (p. 208)来检索新的 ID 令牌和访问令牌。撤消令牌 (p. 209)来撤消刷新令牌允许的用户访问权限。

Amazon Cognito 以 Base64 编码字符串的形式发布令牌。您可以将任何 Amazon Cognito ID 或访问令牌从 Base64 解码为纯文本 JSON。Amazon Cognito 刷新令牌已加密，Amazon Cognito 管理员或用户无法读取。

使用令牌进行身份验证

当用户登录您的应用程序时，Amazon Cognito 会验证登录信息。如果登录成功，Amazon Cognito 会创建会话并返回经过身份验证的用户的 ID 令牌、访问令牌和刷新令牌。您可以使用这些令牌向您的用户授予对您的服务器端资源或 Amazon API Gateway 的访问权限。或者，您可以用它们交换用于访问其他AWS服务的临时AWS凭证。



存储令牌

您的应用程序必须能够存储不同大小的令牌。令牌大小变化的原因可能包括但不限于其它声明、编码算法的更改以及加密算法的更改等。当您在用户群体中启用令牌吊销时，Amazon Cognito 会向 JSON Web 令牌添加其他声明，从而增加令牌大小。新 `origin_jti` 和 `jti` 声明已添加到访问和 ID 令牌中。有关令牌撤消的更多信息，请参阅撤消令牌。

> **Important**
>
> 最佳实践是在应用程序环境中保护传输和存储中的所有令牌。令牌可以包含有关用户的个人识别信息，以及有关用于用户群体的安全模型的信息。

主题

- 使用 ID 令牌 (p. 204)
- 使用访问令牌 (p. 206)
- 使用刷新令牌 (p. 208)
- 撤消令牌 (p. 209)
- 验证 JSON Web 令牌 (p. 211)
- 缓存令牌 (p. 214)

使用 ID 令牌

ID 令牌是一个 JSON Web 令牌（JWT），其中包含有关经身份验证的用户的身份声明，如 `name`、`email` 和 `phone_number`。您可以在应用程序中使用此身份信息。此外，ID 令牌还可用于针对资源服务器或服务器应用程序对用户进行身份验证。您还可以将应用程序外部的 ID 令牌用于 Web API 操作。在这些情况下，您必须先验证 ID 令牌的签名，然后才能信任 ID 令牌内的任何声明。请参阅验证 JSON Web 令牌 (p. 211)。

您可以将 ID 令牌过期时间设置为 5 分钟到 1 天之间的任何值。您可以按应用程序客户端设置此值。

> **Important**
>
> 当您的用户使用托管 UI 或联合身份提供商 (IdP, Identity Provider) 登录时，Amazon Cognito 会设置有效期为 1 小时的会话 Cookie。如果您使用托管 UI 或联合身份验证，并为访问权限和 ID 令牌指定的不到 1 小时的最短持续时间，则您的用户在 Cookie 过期前仍将拥有有效的会话。如果用户的令牌在一小时的会话期间过期，则用户可以刷新他们的令牌，而无需重新身份验证。

ID 令牌标头

标头包含两部分信息：密钥 ID (`kid`) 和算法 (`alg`)。

```
{
"kid" : "1234example=",
"alg" : "RS256"
}
```

`kid`

密钥 ID。其值指示用于保护令牌的 JSON Web Signature (JWS) 的密钥。您可以在 `jwks_uri` 端点上查看您的用户群体签名密钥 ID。

有关 `kid` 参数的更多信息，请参阅密钥标识符 (kid) 标头参数。

`alg`

Amazon Cognito 用于保护访问令牌的加密算法。用户池使用 RS256 加密算法，这是一种采用 SHA-256 的 RSA 签名。

有关 `alg` 参数的更多信息，请参阅算法（alg）标头参数。

ID 令牌负载

这是来自 ID 令牌的示例负载。它包含有关经过身份验证的用户的声明。有关 OpenID Connect（OIDC）标准声明的更多信息，请参阅 OIDC 标准声明。

```
<header>.
```

```
{
   "sub":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "cognito:groups":[
      "my-test-group"
   ],
   "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",
   "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

经过身份验证的用户的唯一标识符 (UUID) 或主题。用户名在您的用户群体中可能不是唯一的。sub 声明是识别给定用户的最佳方法。

cognito:groups

以您的用户为成员的用户群体组的名称数组。

cognito:preferred_role

与用户的最高优先级用户群体组关联的 IAM 角色的 ARN。

iss

发行令牌的身份提供者。声明采用以下格式。

```
https://cognito-idp.<Region>.amazonaws.com/<your user pool ID>
```

cognito:username

用户群体中用户的用户名。

nonce

nonce 声明来自同名的参数，您可以将其添加到 OAuth 2.0 authorize 端点的请求中。添加参数时，nonce 声明包含在 Amazon Cognito 颁发的 ID 令牌中，您可以使用它来防范重播攻击。如果在您的请求中未提供 nonce 值，当您通过第三方身份提供商进行身份验证时，Amazon Cognito 会自动生成并验证随机数，然后将其作为 nonce 声明添加到 ID 令牌中。Amazon Cognito 中的 nonce 声明的实现基于 OIDC 标准。

origin_jti

与用户的刷新令牌关联的令牌撤消标识符。Amazon Cognito 在检查您是否通过 撤消端点 (p. 567) 或 RevokeToken API 操作撤消了用户的令牌时引用了 origin_jti 声明。当您撤消令牌时，Amazon Cognito 会使所有具有相同 origin_jti 值的访问令牌和 ID 令牌失效。

cognito:roles

与您的用户组关联的 IAM 角色的名称的数组。

aud

对用户进行身份验证的用户群体应用程序客户端。Amazon Cognito 在访问令牌 `client_id` 声明中呈现相同的值。

token_use

令牌的预期用途。在 ID 令牌中，其值为 `id`。

auth_time

您的用户完成身份验证的身份验证时间，采用 Unix 时间格式。

exp

您的用户令牌的过期时间，采用 Unix 时间格式。

iat

Amazon Cognito 颁发您的用户令牌的时间，采用 Unix 时间格式。

jti

JWT 的唯一标识符。

ID 令牌可包含 OIDC 标准声明中所定义的 OIDC 标准声明。ID 令牌还可包含您在用户池中定义的自定义属性。无论属性类型如何，Amazon Cognito 都会将自定义属性值作为字符串写入 ID 令牌。

Note

用户池自定义属性始终以 custom: 为前缀。

ID 令牌签名

ID 令牌的签名根据 JWT 令牌的标头和负载计算。在您接受应用程序收到的任何 ID 令牌中的声明之前，请验证该令牌的签名。有关更多信息，请参阅“验证 JSON Web 令牌”。验证 JSON Web 令牌 (p. 211)。

使用访问令牌

用户池访问令牌包含有关经身份验证的用户声明、用户组列表以及范围列表。访问令牌的目的是授权执行 API 操作。您的用户群体接受访问令牌以授权用户执行自助操作。例如，您可以使用访问令牌授予用户访问权限以添加、更改或删除用户属性。

通过访问令牌中的 OAuth 2.0 范围（派生自您添加到用户群体的自定义范围），您可以授权用户从 API 检索信息。例如，Amazon API Gateway 支持使用 Amazon Cognito 访问令牌进行授权。您可以使用用户群体中的信息填充 REST API 授权方，也可以使用 Amazon Cognito 作为 HTTP API 的 JSON 网络令牌（JWT）授权方。要生成具有自定义范围的访问令牌，您必须通过用户群体公有端点请求访问令牌。

您的用户的访问令牌是允许从 UserInfo 终端节点 (p. 566) 中请求有关您的用户属性的更多信息的权限。您的用户的访问令牌也是读取和写入用户属性的权限。访问令牌授予的属性访问级别取决于分配给应用程序客户端的权限以及在令牌中授予的范围。

访问令牌是 JSON Web 令牌 (JWT)。访问令牌的标头与 ID 令牌具有相同的结构。Amazon Cognito 使用与签署 ID 令牌的密钥所不同的密钥对访问令牌进行签名。访问密钥 ID (`kid`) 声明的值与来自同一用户会话的 ID 令牌中 `kid` 声明的值不匹配。在您的应用程序代码中，单独验证 ID 令牌和访问令牌。在验证签名之前，请勿信任访问令牌中的声明。有关更多信息，请参阅验证 JSON Web 令牌 (p. 211)。您可以将访问令牌过期时间设置为 5 分钟到 1 天之间的任何值。您可以按应用程序客户端设置此值。

Important

对于访问令牌和 ID 令牌，如果使用托管 UI，请勿将最小值指定少于一小时。Amazon Cognito HostedUI 使用有效期为一小时的 Cookie。如果您输入的最短时间不到一个小时，则无法获得较短的过期时间。

访问令牌标头

标头包含两部分信息：密钥 ID (kid) 和算法 (alg)。

```
{
"kid" : "1234example="
"alg" : "RS256",
}
```

kid

密钥 ID。其值指示用于保护令牌的 JSON Web Signature (JWS) 的密钥。您可以在 jwks_uri 端点上查看您的用户群体签名密钥 ID。

有关 kid 参数的更多信息，请参阅密钥标识符 (kid) 标头参数。

alg

Amazon Cognito 用于保护访问令牌的加密算法。用户池使用 RS256 加密算法，这是一种采用 SHA-256 的 RSA 签名。

有关 alg 参数的更多信息，请参阅算法（alg）标头参数。

访问令牌有效负载

这是来自访问令牌的示例负载。有关更多信息，请参阅 JWT 声明。

```
<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

经过身份验证的用户的唯一标识符 (UUID) 或主题。用户名在您的用户群体中可能不是唯一的。sub 声明是识别给定用户的最佳方法。

cognito:groups

以您的用户为成员的用户群体组的名称数组。

iss

发行令牌的身份提供者。声明采用以下格式。

`https://cognito-idp.<Region>.amazonaws.com/<your user pool ID>`

client_id

对用户进行身份验证的用户群体应用程序客户端。Amazon Cognito 在 ID 令牌 aud 声明中呈现相同的值。

origin_jti

与用户的刷新令牌关联的令牌撤消标识符。Amazon Cognito 在检查您是否通过 撤消端点 (p. 567) 或 RevokeToken API 操作撤消了用户的令牌时引用了 `origin_jti` 声明。当您撤消令牌时，Amazon Cognito 会使所有具有相同 `origin_jti` 值的访问令牌和 ID 令牌失效。

token_use

令牌的预期用途。在访问令牌中，其值为 `access`。

scope

OAuth 2.0 范围的列表，这些范围用于定义令牌提供的访问权限。令牌端点 (p. 562) 中的令牌可以包含您的应用程序客户端支持的任何范围。来自 Amazon Cognito API 登录的令牌仅包含范围 `aws.cognito.signin.user.admin`。

auth_time

您的用户完成身份验证的身份验证时间，采用 Unix 时间格式。

exp

您的用户令牌的过期时间，采用 Unix 时间格式。

iat

Amazon Cognito 颁发您的用户令牌的时间，采用 Unix 时间格式。

jti

JWT 的唯一标识符。

访问令牌签名

访问令牌的签名根据 JWT 令牌的标头和负载计算。在 Web API 中，当您在应用程序外部使用时，您必须始终先验证此签名，然后才能接受该令牌。有关更多信息，请参阅 JWT 令牌。

使用刷新令牌

您可以使用刷新令牌来检索新的 ID 令牌和访问令牌。默认情况下，刷新令牌会在您的应用程序用户登录用户池的 30 天后过期。当您为用户池创建应用程序时，您可以将应用程序的刷新令牌到期时间设置为介于 60 分钟和 10 年之间的任何值。

如果存在有效的（未过期）刷新令牌，则 Mobile SDK for iOS、Mobile SDK for Android、Amplify for iOS、Android 和 Flutter 会自动刷新您的 ID 和访问令牌。ID 和访问令牌的剩余有效期至少为 2 分钟。如果刷新令牌已过期，您的应用程序用户必须通过再次登录用户池来重新进行身份验证。如果访问令牌和 ID 令牌的最小值设置为 5 分钟，并且您正在使用 SDK，则刷新令牌将持续用于检索新访问和 ID 令牌。您会看到预期行为的最小值设置为 7 分钟，而不是 5 分钟。

只要用户在针对新账户的 `UnusedAccountValidityDays` 时间限制之前至少登录一次，用户账户本身就不会过期。

启动新的刷新令牌 (API)

使用 API 或托管 UI 来启动刷新令牌的身份验证。

要结合使用刷新令牌与用户池 API 获取新的 ID 令牌和访问令牌，请使用 AdminInitiateAuth 或 InitiateAuth API 操作。为 AuthFlow 参数传递 REFRESH_TOKEN_AUTH。授权参数 AuthParameters 是密钥-值映射，其中密钥为 "REFRESH_TOKEN"，值为实际刷新令牌。在您的 API 请求通过所有质询后，Amazon Cognito 会返回新的 ID 和访问令牌。

Note

要更改使用托管 UI 登录的用户的令牌，请使用 InitiateAuth API 操作。

撤消刷新令牌

您可以撤销属于用户的刷新令牌。有关撤消令牌的更多信息，请参阅撤消令牌 (p. 209)。

Note

撤消刷新令牌会撤消所有用刷新令牌发布的令牌。

当您使用 GlobalSignOut 和 AdminUserGlobalSignOut API 操作撤消用户的所有令牌时，用户可以从他们当前登录的所有设备中注销。用户注销后，发生下列情况：

- 用户的刷新令牌无法用于为用户获取新的令牌。
- 用户的访问令牌无法用于用户池服务。
- 用户必须重新进行身份验证以获取新的令牌。会话 Cookie 不会自动过期。作为最佳实践，应用程序应将用户重新导向到注销端点，以强制浏览器清除会话 Cookie。

应用程序可以使用 GlobalSignOut API 来允许单个用户自行从所有设备注销。通常，应用程序会将此选项显示为一个选择，例如 Sign out from all devices（从所有设备注销）。该应用程序必须使用用户未过期、未撤消的有效访问令牌调用此 API 操作。此操作无法用于允许一个用户注销另一个用户。

应用程序可以使用 AdminUserGlobalSignOut API 来允许管理员将一个用户从所有设备注销。管理员应用程序必须通过 AWS 开发人员凭证调用此 API 操作，并传递用户群体 ID 和用户的用户名作为参数。AdminUserGlobalSignOut API 可以在用户池中注销任何用户。

撤消令牌

您可以使用 AWS API 为用户撤消刷新令牌。撤消刷新令牌后，先前由该刷新令牌颁发的所有访问令牌都将无效。向用户颁发的其他刷新令牌不受影响。

Note

JWT 令牌是独立的，具有在创建令牌时分配的签名和过期时间。已撤消令牌无法与任何需要令牌的 Amazon Cognito API 调用一起使用。但是，如果使用任何验证令牌签名和过期的 JWT 库进行验证，已撤消令牌仍然有效。

您可以在已启用令牌撤消的情况下撤消用户池客户端的刷新令牌。当您创建新的用户池客户端时，默认会启用令牌撤消。

启用令牌撤消

在您为现有的用户池客户端撤消令牌前，必须启用令牌撤销。您可以使用 AWS CLI 或 AWS API 为现有用户池客户端启用令牌撤销。为此，请调用 `aws cognito-idp describe-user-pool-client` CLI 命令或 DescribeUserPoolClient API 操作以从应用程序客户端检索当前设置。然后调用 `aws cognito-idp update-user-pool-client` CLI 命令或 UpdateUserPoolClient API 操作。包括来自您的应用客户端的当前设置并将 EnableTokenRevocation 参数设置为 `true`。

当您使用AWS Management Console、AWS CLI 或 AWS API 创建新的用户池客户端时，默认会启用令牌撤消。

启用令牌吊销后，Amazon Cognito JSON Web 令牌中会添加新的声明。`origin_jti` 和 `jti` 声明已添加到访问和 ID 令牌中。这些声明增加应用程序客户端访问和 ID 令牌的大小。

以下 JSON 示例显示请求使用 `CreateUserPoolClient` API 来启用令牌撤消。

```
{
    "AccessTokenValidity": 123,
    "AllowedOAuthFlows": [
        "string"
    ],
    "AllowedOAuthFlowsUserPoolClient": true,
    "AllowedOAuthScopes": [
        "string"
    ],
    "AnalyticsConfiguration": {
        "ApplicationArn": "string",
        "ApplicationId": "string",
        "ExternalId": "string",
        "RoleArn": "string",
        "UserDataShared": false
    },
    "CallbackURLs": [
        "string"
    ],
    "ClientName": "string",
    "DefaultRedirectURI": "string",
    "ExplicitAuthFlows": [
        "string"
    ],
    "GenerateSecret": true,
    "IdTokenValidity": 123,
    "LogoutURLs": [
        "string"
    ],
    "PreventUserExistenceErrors": "string",
    "ReadAttributes": [
        "string"
    ],
    "RefreshTokenValidity": 456,
    "SupportedIdentityProviders": [
        "string"
    ],
    "TokenValidityUnits": {
        "AccessToken": "string",
        "IdToken": "string",
        "RefreshToken": "string"
    },
    "UserPoolId": "string",
    "WriteAttributes": [
        "string"
    ],
    "EnableTokenRevocation": true
}
```

撤消令牌

您可以使用 `RevokeToken` API 操作来撤消刷新令牌。您也可以使用 `aws cognito-idp revoke-token` CLI 命令来撤消令牌。最后，您还可以使用撤消端点来撤消令牌。此端点在您将域添加到用户池后可用。您可以在 Amazon Cognito 托管域或您的自定义域上使用撤消端点。

> **Note**
>
> 要撤消刷新令牌，您必须使用与获取令牌所用的相同客户端 ID。

验证 JSON Web 令牌

这些步骤描述了验证用户池 JSON Web 令牌（JWT）的过程。

主题

- 先决条件 (p. 211)
- 使用 aws-jwt-verify 验证令牌 (p. 211)
- 了解和检查令牌 (p. 211)

先决条件

您的库、开发工具包或软件框架可能已处理本部分中的任务。AWSSDK 提供在应用程序中处理和管理 Amazon Cognito 用户群体令牌的工具。AWS Amplify 包括检索和刷新 Amazon Cognito 令牌的功能。

有关更多信息，请参阅以下页面。

- 将 Amazon Cognito 身份验证和授权与 Web 和移动应用程序集成 (p. 20)
- 使用 AWS SDK 的 Amazon Cognito 身份提供者的代码示例
- Amplify Dev Center（Amplify 开发中心）中的 Advanced workflows（高级工作流）

许多库可用于解码和验证 JSON Web 令牌 (JWT)。如果您要手动处理用于服务器端 API 处理的令牌，或者您使用的是其他编程语言，则这些库可以为您提供帮助。请参阅用于处理 JWT 令牌库的 OpenID Foundation 列表。

使用 aws-jwt-verify 验证令牌

在 Node.js 应用程序中，AWS 建议使用 aws-jwt-verify 库来验证用户传递给您的应用程序的令牌中的参数。使用 `aws-jwt-verify`，您可以使用要为一个或多个用户群体验证的声明值填充 `CognitoJwtVerifier`。它可以检查的一些值包括以下内容。

- 该访问令牌或 ID 令牌的格式不正确或已过期，但具有有效的签名。
- 这些访问令牌来自正确的用户群体和应用程序客户端。
- 该访问令牌声明包含正确的 OAuth 2.0 范围。
- 对访问令牌和 ID 令牌进行签名的密钥与用户群体的 JWKS URI 中的签名密钥 `kid` 匹配。

  JWKS URI 包含有关对用户令牌进行签名的私有密钥的公有信息。您可以在以下位置找到用户群体的 JWKS URI：`https://cognito-idp.<Region>.amazonaws.com/<userPoolId>/.well-known/jwks.json`。

有关更多信息和您可以在 Node.js 应用程序或 AWS Lambda 授权方中使用的示例代码，请参阅 GitHub 上的 aws-jwt-verify。

了解和检查令牌

在将令牌检查与应用程序集成之前，请考虑 Amazon Cognito 如何组装 JWT。从用户群体中检索示例令牌。解码并详细检查它们，以了解它们的特征，并确定要验证的内容和时间。例如，您可能希望检查一个场景中的组成员资格，而在另一个场景中，您可能想要检查范围。

以下部分描述在准备应用程序时手动检查 Amazon Cognito JWT 的过程。

确认 JWT 的结构

JSON Web 令牌 (JWT) 包括三个部分，各部分之间有一个 .（圆点）分隔符。

标头

Amazon Cognito 用来对令牌进行签名的密钥 ID `kid` 和 RSA 算法 `alg`。Amazon Cognito 使用 `alg` (`RS256`) 对令牌进行签名。

有效负载

令牌声明。在 ID 令牌中，声明包括用户属性和有关用户群体 `iss` 和应用程序客户端 `aud` 的信息。在访问令牌中，有效负载包括范围、组成员资格、用户群体身份 (`iss`) 和应用程序客户端 (`client_id`)。

签名

签名不是像标头和有效负载那样的可解码 base64。它是一个 RSA256 标识符，派生自签名密钥和参数（您可以在 JWKS URI 上观察到）。

标头和有效负载是以 base64 编码的 JSON。您可以通过解码为起始字符 `eyJ` 的开头字符 `{` 来识别它们。如果用户向您的应用程序提供了以 base64 编码的 JWT，但其格式不是 `[JSON Header].[JSON Payload].[Signature]`，则它不是有效的 Amazon Cognito 令牌，您可以将其丢弃。

验证 JWT

JWT 签名是标头和负载的哈希组合。Amazon Cognito 为每个用户池生成两对 RSA 加密密钥。一个私有密钥对访问令牌进行签名，另一个私有密钥对 ID 令牌进行签名。

验证 JWT 令牌的签名

1. 解码 ID 令牌。

   OpenID Foundation 还维护用于处理 JWT 令牌的库列表。

   您也可以使用 AWS Lambda 解码用户群体 JWT。有关更多信息，请参阅使用 AWS Lambda 解码并验证 Amazon Cognito JWT 令牌。
2. 将本地密钥 ID (`kid`) 与公有 `kid` 进行比较。

   a. 下载并存储适用于用户池的对应的公有 JSON Web Key（JWK）。它可作为 JSON Web Key Set (JWKS) 的一部分提供。您可以通过为您的环境构建以下 `jwks_uri` URL 来找到它：

   ```
   https://cognito-idp.<Region>.amazonaws.com/<userPoolId>/.well-known/jwks.json
   ```

   有关更多 JWK 和 JWK 集的更多信息，请参阅 JSON Web Key (JWK)。

   > **Note**
   >
   > Amazon Cognito 可能会轮换用户群体中的签名密钥。最佳做法是使用 `kid` 作为缓存密钥在应用程序中缓存公有密钥，并定期刷新缓存。将您的应用程序收到的令牌中的 `kid` 与缓存进行比较。
   > 如果您收到的令牌的颁发者是正确的，但 `kid` 不同，则 Amazon Cognito 可能已经轮换了签名密钥。从您的用户群体 `jwks_uri` 端点刷新缓存。

   这是个 `jwks.json` 文件示例：

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

密钥 ID（**kid**）

kid 是一个提示，指示哪个密钥用于保护令牌的 JSON Web 签名（JWS）。

算法（**alg**）

alg 标头参数表示用于保护 ID 令牌的加密算法。用户池使用 RS256 加密算法，这是一种采用 SHA-256 的 RSA 签名。有关 RSA 的更多信息，请参阅 RSA 加密。

密钥类型（**kty**）

kty 参数标识与密钥结合使用的加密算法系列，例如，在本示例中为“RSA”。

RSA 指数（**e**）

e 参数包含 RSA 公有密钥的指数值。它表示为采用 Base64urlUInt 编码的值。

RSA 模数（**n**）

n 参数包含 RSA 公有密钥的模数值。它表示为采用 Base64urlUInt 编码的值。

使用（**use**）

use 参数描述了公有密钥的预期用途。在本示例中，use 值 sig 表示签名。

b. 搜索与您 JWT 的 kid 相匹配的 kid 的公有 JSON Web 密钥。

3. 使用 JWT 库将颁发者的签名与令牌中的签名进行比较。发布者签名来自在 jwks.json 中的 kid 公有密钥（RSA 模数 "n"），该公有密钥与令牌 kid 匹配。您可能首先需要将 JWK 转换为 PEM 格式。以下示例采用 JWT 和 JWK 格式，并且使用 Node.js 库 jsonwebtoken 来验证 JWT 签名：

   Node.js

```
var jwt = require('jsonwebtoken');
var jwkToPem = require('jwk-to-pem');
var pem = jwkToPem(jwk);
jwt.verify(token, pem, { algorithms: ['RS256'] }, function(err, decodedToken) {
});
```

验证声明

验证 JWT 声明

1. 通过以下方法之一，验证令牌是否未过期。

   a. 对令牌解码并将 exp 声明与当前时间进行比较。

   b. 如果访问令牌包含 aws.cognito.signin.user.admin 声明，请向 GetUser 等 API 发送请求。如果令牌已过期，您使用访问令牌进行授权的 API 请求会返回错误。

   c. 在针对UserInfo 终端节点 (p. 566)的请求中提供您的访问令牌。如果您的令牌已过期，则请求会返回错误。

2. ID 令牌中的 aud 声明和访问令牌中的 client_id 声明应与在 Amazon Cognito 用户池中创建的应用程序客户端 ID 匹配。

3. 发布者 (iss) 声明应与您的用户池匹配。例如，在 us-east-1 区域中创建的用户池将有下列 iss 值：

   https://cognito-idp.us-east-1.amazonaws.com/*<userpoolID>*.
4. 检查 token_use 声明。

   - 如果您在 Web API 操作中只接受访问令牌，则其值必须为 access。
   - 如果您只使用 ID 令牌，则其值必须为 id。
   - 如果您同时使用 ID 令牌和访问令牌，则 token_use 声明必须为 id 或 access。

您现在可以信任该令牌内的声明。

缓存令牌

每次您想要获取新的 JSON Web 令牌（JWT）时，应用程序都必须成功完成以下请求之一。

- 从令牌端点 (p. 562)请求客户端凭证或授权代码授予。
- 从您的托管 UI 请求隐式授权。
- 在 Amazon Cognito API 请求（例如 InitiateAuth）中对本地用户进行身份验证。

您可以将用户群体配置为将令牌设置为在数分钟、数小时或数天后过期。要确保应用程序的性能和可用性，请使用 Amazon Cognito 令牌直到它们过期，并且仅在那时检索新令牌。您为应用程序构建的缓存解决方案可保持令牌可用，并可防止 Amazon Cognito 在您的请求速率过高时拒绝请求。客户端应用程序必须将令牌存储在内存缓存中。服务器端应用程序可以添加加密的缓存机制来存储令牌。

当您的用户群体生成大量用户或机器到机器活动时，您可能会遇到 Amazon Cognito 对您可以发出的令牌请求数量设置的限制。要减少您向 Amazon Cognito 端点发出的请求数量，可以安全地存储和重复使用身份验证数据，也可以实施指数回退和重试。

身份验证数据来自两类端点。Amazon Cognito OAuth 2.0 端点包括令牌端点，该端点为客户端凭证和托管 UI 授权代码请求提供服务。服务端点应答用户群体 API 请求，例如 InitiateAuth 和 RespondToAuthChallenge。每种类型的请求都有自己的限制。有关 限制的更多信息，请参阅 Amazon Cognito 中的限额 (p. 539)。

使用 Amazon API Gateway 缓存机器到机器访问令牌

借助 API Gateway 令牌缓存，您的应用程序可以横向缩减对大于 Amazon Cognito OAuth 端点的默认请求速率配额的事件的响应。

您可以缓存访问令牌，以便您的应用程序仅在缓存的令牌过期时才请求新的访问令牌。否则，您的缓存端点会从缓存中返回一个令牌。这会阻止对 Amazon Cognito API 端点的其他调用。当您使用 Amazon API Gateway 作为 令牌端点 (p. 562)的代理时，您的 API 会响应大多数原本会计入您的请求配额的请求，从而避免由于速率限制而导致的请求失败。

以下基于 API Gateway 的解决方案提供了低延迟、低代码/无代码的令牌缓存实施。API Gateway API 在传输中加密，也可以选择静态加密。API Gateway 缓存非常适合 OAuth 2.0 客户端凭证授予，这是一种经常性的高容量授予类型，它会生成访问令牌以授权机器到机器和微服务会话。在诸如导致微服务横向扩展的流量激增之类的事件中，您最终可以让许多系统使用相同的客户端凭证，其数量超过您的用户群体或应用程序客户端的 AWS 请求速率限制。要保持应用程序可用性和低延迟，缓存解决方案是此类情况下的最佳实践。

在此解决方案中，您在 API 中定义缓存，以便为您要在应用程序中请求的 OAuth 范围和应用程序客户端的每种组合存储单独的访问令牌。当您的应用程序发出与缓存键匹配的请求时，您的 API 会使用 Amazon Cognito 向与缓存键匹配的第一个请求颁发的访问令牌进行响应。当您的缓存键持续时间到期时，API 会将请求转发到令牌端点并缓存新的访问令牌。

> Note
>
> 您的缓存键持续时间必须短于应用程序客户端的访问令牌持续时间。

缓存键是您在 scope URL 参数中请求的 OAuth 范围和请求中的 Authorization 标头的组合。Authorization 标头包含您的应用程序客户端 ID 和客户端密钥。您无需在应用程序中实施其他逻辑即可实施此解决方案。您只必须更新配置以更改用户群体令牌端点的路径。

您还可以使用 ElastiCache for Redis 实施令牌缓存。要使用 AWS Identity and Access Management (IAM) 策略进行精细控制，请考虑 Amazon DynamoDB 缓存。

Note

在 API Gateway 中缓存需要支付额外费用。有关更多详细信息，请参阅定价。

使用 API Gateway 设置缓存代理

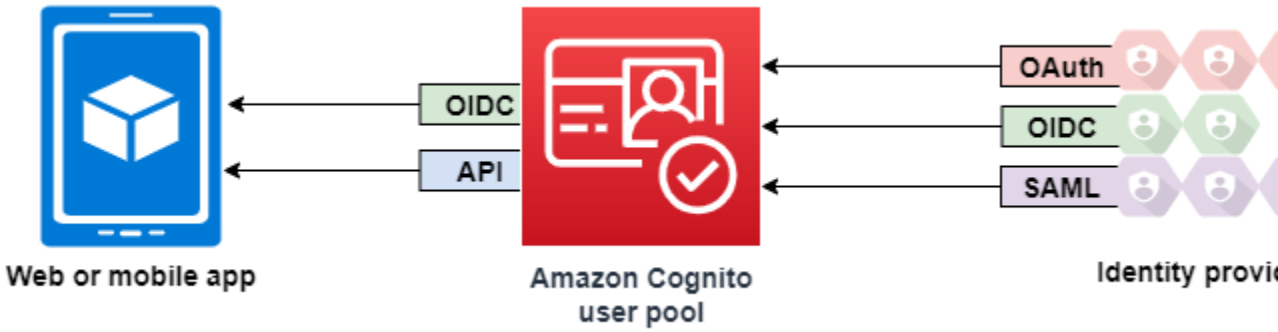
1. 打开 API Gateway 控制台，然后创建一个 REST API。
2. 在 Resources（资源）中，创建一个 POST 方法。
   a. 选择 HTTP Integration type（集成类型）。
   b. 选择 Use HTTP proxy integration（使用 HTTP 代理集成）。
   c. 输入 Endpoint URL（端点 URL）`https://<your user pool domain>/oauth2/token`。
3. 在 Resources（资源）中，配置缓存键。
   a. 编辑 POST 方法的 Method request（方法请求）。
   b. 将您的 scope 参数和 Authorization 标头设置为缓存键。
      i. 向 URL query string parameters（URL 查询字符串参数）中添加一个查询字符串，并为 scope 字符串选择 Caching（缓存）。
      ii. 向 HTTP request headers（HTTP 请求标头）中添加一个标头，并为 Authorization 标头选择 Caching（缓存）。
4. 在 Stages（阶段）中，配置缓存。
   a. 选择您想要修改的阶段。
   b. 在 Settings（设置）下，选择 Enable API cache（启用 API 缓存）。
   c. 选择 Cache capacity（缓存容量）。
   d. 为 Cache time-to-live (TTL)[缓存生存时间(TTL)]选择至少 3600 秒。
   e. 清除需要授权复选框。
5. 在 Stages（阶段）中，注意 Invoke URL（调用 URL）。
6. 更新您的应用程序，以将令牌请求发布到 API 的 Invoke URL（调用 URL）而不是用户群体的 /oauth2/token 端点。

在成功进行用户池身份验证后访问资源

您的应用程序用户可以通过用户池直接登录，也可以通过第三方身份提供商（IdP）联合。用户池管理处理以下令牌的开销：从通过 Facebook、Google、亚马逊和 Apple 进行的社交登录返回的令牌以及从 OpenID Connect（OIDC）和 SAML IdP 返回的令牌。有关更多信息，请参阅将令牌与用户池结合使用 (p. 203)。

成功进行身份验证后，您的应用程序将收到来自 Amazon Cognito 的用户池令牌。您可以使用这些令牌来检索允许您的应用程序访问其他AWS服务的AWS凭证。还可以选择使用它们来控制对您自己的服务器端资源或 Amazon API Gateway 的访问权限。

有关更多信息，请参阅 用户池身份验证流程 (p. 41) 和 将令牌与用户池结合使用 (p. 203)：

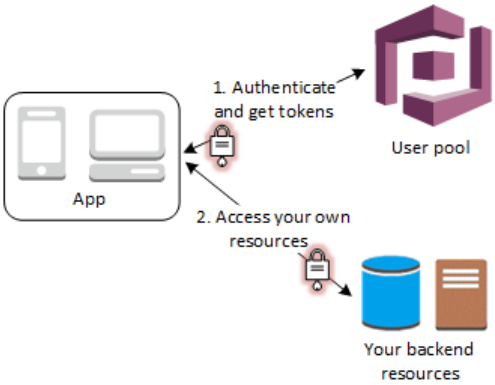


主题

- 在登录后访问服务器端资源 (p. 13)
- 在登录后使用 API Gateway 和 Lambda 访问资源 (p. 216)
- 在登录后使用身份池访问 AWS 服务 (p. 217)

在登录后访问服务器端资源

成功进行身份验证后，您的 Web 或移动应用程序将收到来自 Amazon Cognito 的用户池令牌。您可以使用这些令牌控制对您的服务器端资源的访问。您也可以创建用户池组来管理权限以及表示不同类型的用户。有关使用组控制资源访问权限的更多信息，请参阅向用户池添加组 (p. 178)。

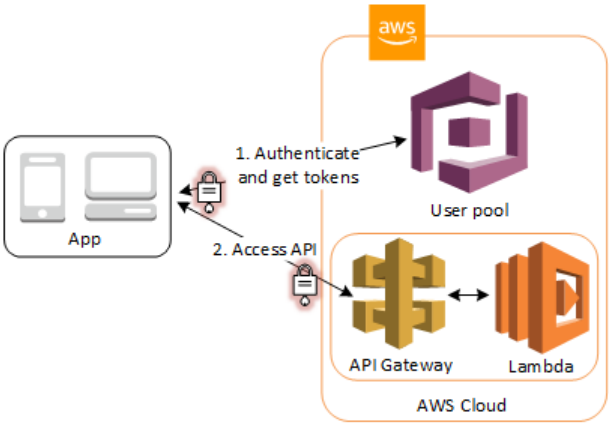


在为用户池配置域后，Amazon Cognito 会预置一个托管 Web UI，您可使用此 UI 为应用程序添加注册页和登录页。使用此 OAuth 2.0 基础，您可以创建自己的资源服务器，以便您的用户能够访问受保护的资源。有关更多信息，请参阅为您的用户池定义资源服务器 (p. 82)。

有关用户群体身份验证的更多信息，请参阅用户池身份验证流程 (p. 41)和将令牌与用户池结合使用 (p. 203)。

在登录后使用 API Gateway 和 Lambda 访问资源

用户可以通过 API Gateway 访问您的 API。API Gateway 会验证来自成功用户池身份验证的令牌，并使用它们向您的用户授予对资源（包括 Lambda 函数）或您自己的 API 的访问权限。



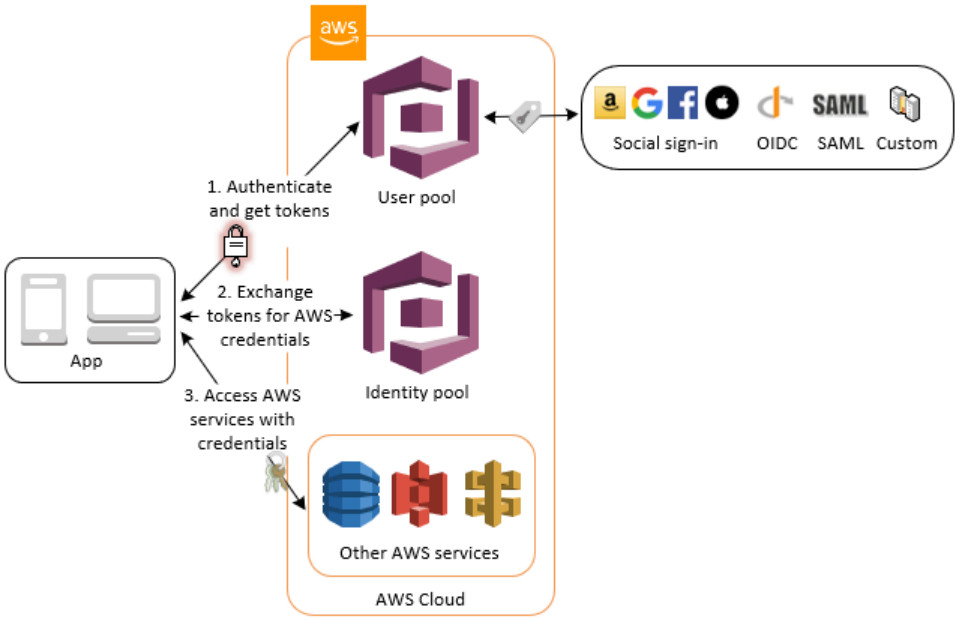
您可以使用用户池中的组控制对 API Gateway 的权限，方法是将组成员资格映射到 IAM 角色。用户所属的组包含在您的 Web 或移动应用程序用户登录时用户池提供的 ID 令牌中。有关用户池组的更多信息，请参阅向用户池添加组 (p. 178)。

您可以将您的用户池令牌随请求一起提交到 API Gateway，以便 Amazon Cognito 授权方 Lambda 函数进行验证。有关 API Gateway 的更多信息，请参阅将 API Gateway 与 Amazon Cognito 用户池结合使用。

在登录后使用身份池访问 AWS 服务

您可以允许用户通过用户池登录，然后使用身份池访问AWS服务。

成功进行身份验证后，您的 Web 或移动应用程序将收到来自 Amazon Cognito 的用户池令牌。您可以使用这些令牌来检索允许您的应用程序访问其他AWS服务的AWS凭证。有关更多信息，请参阅Amazon Cognito 身份池入门（联合身份） (p. 289)。



有关结合使用身份池和用户池组控制AWS资源访问权限的更多信息，请参阅向用户池添加组 (p. 178)和使用基于角色的访问控制 (p. 318)。有关身份池和 AWS Identity and Access Management 的更多信息，另请参阅身份池概念（联合身份） (p. 299)。

使用 AWS Management Console 设置用户池

创建 Amazon Cognito 用户池并记下每个客户端应用程序的用户池 ID 和应用程序客户端 ID。有关创建用户池的更多信息，请参阅用户池入门 (p. 30)。

使用 AWS Management Console 设置身份池

以下过程介绍了如何使用 AWS Management Console 将一个身份池与一个或多个用户池和客户端应用程序集成。

AWS Management Console

配置身份池

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 Federated identities（联合身份）。
3. 选择想要针对其启用 Amazon Cognito 用户池作为提供商的身份池的名称。
4. 在 Dashboard（控制面板）页上，选择 Edit identity pool（编辑身份池）。
5. 展开 Authentication providers（身份验证提供商）部分。
6. 选择 Cognito。
7. 输入 User Pool ID（用户池 ID）。
8. 输入 App Client ID（应用程序客户端 ID）。这必须与您在控制台的 User pools（用户池）部分中创建应用程序时接收到的客户端应用程序 ID 一致。
9. 如果您有其它应用程序或用户池，请选择 Add Another Provider（添加其它提供商），并针对每个用户池中的每个应用程序输入 User Pool ID（用户池 ID）和 App Client ID（应用程序客户端 ID）。
10. 如果您没有更多应用程序或用户池要添加，请选择 Save Changes（保存更改）。如果成功，Dashboard（控制面板）页面上将显示 Changes saved successfully（已成功保存更改）消息。

将用户池与身份池集成

对您的应用程序用户进行身份验证后，将用户的身份令牌添加到凭证提供程序中的登录映射中。提供商名称取决于 Amazon Cognito 用户池 ID。结构如下所示：

```
cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>
```

<region> 的值与用户群体 ID 中的区域相同。例如，cognito-idp.us-east-1.amazonaws.com/us-east-1_123456789。

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)
```

使用 Amazon Cognito 用户池安全功能

可以向用户池添加多重验证 (MFA) 以保护用户的身份。MFA 增加了第二种身份验证要素，这样用户群体可以不单单依靠用户名和密码。您可以使用 SMS 文本消息或基于时间的一次性密码 (TOTP) 作为用户登录所需的第二个因素。也可以使用自适应身份验证，其基于风险的模型可预测您可能何时需要另一身份验证要素。用户池高级安全性 功能包括自适应身份验证和防止盗用凭证的保护措施。

主题

- 向用户池添加 MFA (p. 220)

- 向用户池添加高级安全 (p. 227)
- 将 AWS WAF Web ACL 与用户池相关联 (p. 236)
- 用户池区分大小写 (p. 238)
- 管理用户存在错误响应 (p. 239)

向用户池添加 MFA

多重验证 (MFA) 可增强应用程序的安全性。它将您具有的某种 身份验证要素添加到您已知的 用户名和密码要素中。您可以选择将 SMS 文本消息或基于时间的一次性密码 (TOTP) 作为用户登录的第二安全要素。

> Note
>
> 新用户首次登录应用程序时，Amazon Cognito 会发出 OAuth 2.0 令牌，即使您的用户池要求使用 MFA 也是如此。您的用户首次登录时的第二个身份验证因素是他们对 Amazon Cognito 发送给他们的验证消息的确认。如果您的用户池要求使用 MFA，Amazon Cognito 会提示您的用户注册一个额外的登录因素，以便在第一次之后的每次登录尝试期间使用。

借助自适应身份验证，可以将用户池配置为响应增加的风险级别需要第二安全要素身份验证。要向用户池添加自适应身份验证，请参阅 向用户池添加高级安全 (p. 227)。

将用户池的 MFA 设置为 `required` 时，所有用户都必须完成 MFA 才能登录。要登录，每个用户至少设置一个 MFA 安全要素，例如 SMS 或 TOTP。当您将 MFA 设置为 `required` 时，您必须在用户登录中包含 MFA 设置，以便您的用户池允许他们登录。

如果您激活 SMS 作为 MFA 安全要素，则可以要求用户提供电话号码并在注册过程中进行验证。如果您将 MFA 设置为 `required` 并且只支持 SMS 作为要素，则用户必须提供电话号码。没有电话号码的用户将需要您的支持才能将电话号码添加到其个人资料中，然后才能登录。您可以将未经验证的电话号码用于 SMS MFA。MFA 成功后，这些号码的状态将为已验证。

如果您已将 MFA 设置为 `required` 并且激活了 SMS 和 TOTP 作为支持的验证方法，Amazon Cognito 会提示没有电话号码的新用户设置 TOTP MFA。如果您已将 MFA 设置为 `required` 而且您唯一激活的 MFA 方法是 TOTP，Amazon Cognito 会提示所有新用户在第二次登录时设置 TOTP MFA。Amazon Cognito 生成质询以设置 TOTP MFA，作为对 InitiateAuth 和 AdminInitiateAuth API 操作的响应。

在五次尝试提交 MFA 代码均未成功后，Amazon Cognito 会开始如用户池身份验证流程 (p. 41)中所描述的指数超时锁定过程。

主题

- 先决条件 (p. 220)
- 配置多重验证 (p. 221)
- SMS 文本消息 MFA (p. 222)
- TOTP 软件令牌 MFA (p. 222)

先决条件

在设置 MFA 之前，请考虑以下情况：

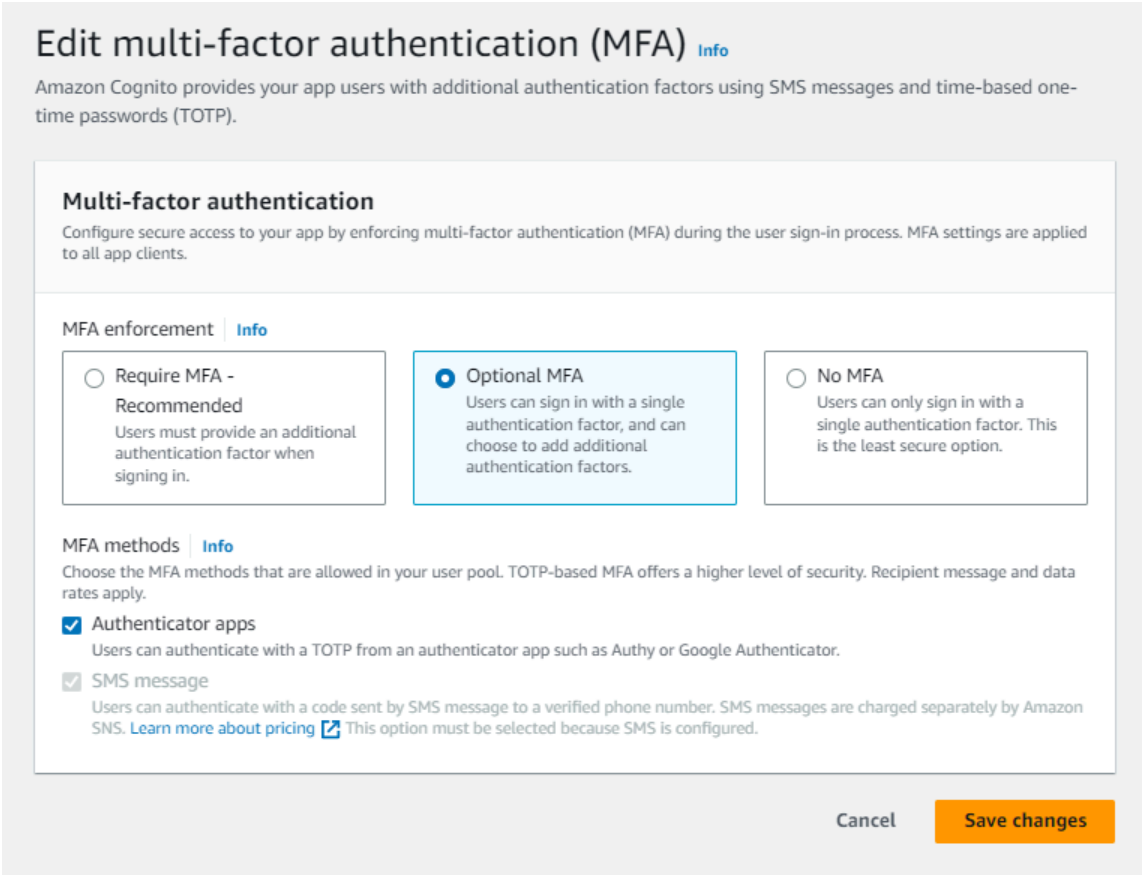
- 当您在用户池中激活 MFA 并选择 SMS text message（SMS 文本消息）作为第二安全要素时，您可以向尚未在 Amazon Cognito 中验证的电话号码属性发送短信。用户完成短信 MFA 后，Amazon Cognito 会将其 `phone_number_verified` 属性设置为 `true`。
- 如果您的账户所在的 SMS 沙盒，位于包含用户池的 Amazon Simple Notification Service (Amazon SNS) 资源的 AWS 区域中，您必须在 Amazon SNS 中验证电话号码，然后才能发送短信。有关更多信息，请参阅 Amazon Cognito 用户池的短信设置 (p. 199)。
- 高级安全功能要求您已激活 MFA 并在 Amazon Cognito 用户池控制台中将其设置为可选。有关更多信息，请参阅向用户池添加高级安全 (p. 227)。

配置多重验证

您可以在 Amazon Cognito 控制台中配置 MFA。

在 Amazon Cognito 控制台中配置 MFA

1. 登录 Amazon Cognito 控制台。
2. 选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 Sign-in experience（登录体验）选项卡。找到 Multi-factor authentication（多重验证），然后选择 Edit（编辑）
5. 选择您希望用于用户池的 MFA enforcement（MFA 执行）方法。



   a. Require MFA（需要 MFA）。用户池中的所有用户必须使用其他 SMS 代码或基于时间的一次性密码（TOTP）安全要素进行登录。
   b. Optional MFA（可选 MFA）– 您可以为用户提供选项来注册额外的登录安全要素，但仍允许未配置 MFA 的用户登录。如果您使用自适应身份验证，请选择此选项。有关自适应身份验证的更多信息，请参阅向用户池添加高级安全 (p. 227)。
   c. No MFA（无 MFA）。您的用户无法注册其他登录安全要素。
6. 选择您在应用程序中支持的 MFA methods（MFA 方法）。您可以将 SMS message（SMS 消息）或可生成 TOTP 的 Authenticator apps（身份验证器应用程序）设置为第二个登录因素。我们建议您实施基于 TOTP 的 MFA，这样可以使用 SMS 消息找回账户的密码。
7. 如果使用 SMS 文本消息作为第二安全要素，并且没有配置 IAM 角色与 Amazon Simple Notification Service (Amazon SNS) 一起使用 SMS 消息，您可以在控制台中创建一个角色。在用户池的

Messaging（消息收发）选项卡下，找到SMS 并选择 Edti（编辑）。您还可以使用允许 Amazon Cognito 代表您向用户发送短信的现有角色。有关更多信息，请参阅 IAM 角色。

8. 选择 Save changes（保存更改）。

SMS 文本消息 MFA

如果用户登录时 MFA 处于启用状态，他们首先要输入并提交其用户名和密码。客户端应用程序将收到指示授权代码发送位置的 `getMFA` 响应。客户端应用程序应向用户指明在哪里查找该代码，例如代码将被发送到哪个电话号码。接下来，它提供一个用于输入代码的表单。最后，客户端应用程序提交代码以完成登录过程。目的地被屏蔽，这将隐藏电话号码除最后 4 位以外的所有数字。如果一个应用程序正在使用 Amazon Cognito 托管 UI，则应用程序会显示一个页面，供用户输入 MFA 代码。

SMS 文本消息授权代码对您为应用程序客户端设置的 Authentication flow session duration（身份验证流程会话持续时间）有效。

当您在 App clients and analytics（应用程序客户端和分析）下修改应用程序客户端时，在 Amazon Cognito 控制台的 App integration（应用程序集成）选项卡中设置身份验证流程会话的持续时间。您还可以在 `CreateUserPoolClient` 或 `UpdateUserPoolClient` API 请求中设置身份验证流程会话持续时间。有关更多信息，请参阅用户池身份验证流程 (p. 41)。

如果用户无法再访问其发送短信 MFA 代码的设备，则它们必须请求客户服务部门帮助。具备必要 AWS 账户权限的管理员可以更改用户的电话号码，但只能通过 AWS CLI 或 API 进行。

当用户成功通过短信 MFA 流程时，其电话号码也将标记为“已验证”。

> **Note**
>
> SMS for MFA 是单独收费的。(向电子邮件地址发送验证代码是不收费的。) 有关 Amazon SNS 定价的信息，请参阅 Worldwide SMS 定价。有关提供 SMS 消息发送服务的最新国家/地区列表，请参阅支持的区域和国家/地区。
>
> **Important**
>
> 要确保能够发送短信来验证电话号码，或针对短信 MFA 发送短信，您必须请求增加 Amazon SNS 的支出限额。
> Amazon Cognito 使用 Amazon SNS 向用户发送短信。Amazon SNS 发送短信的数量取决于支出限额。您可以为AWS账户和单条消息指定支出限额，且该限额仅适用于发送短信的费用。
> 每个账户的默认支出限额（如果未指定）为每月 1.00 USD。如果您想提高此限额，请在 AWS Support Center 提交 SNS 限额提高案例。在新限制值中，输入所需的每月支出限额。在 Use Case Description（使用案例描述）字段中，说明您要请求提高每月的 SMS 支出限额。

要向用户池添加 MFA，请参阅 向用户池添加 MFA (p. 220)。

TOTP 软件令牌 MFA

当您在用户群体中设置了 TOTP 软件令牌 MFA 时，您的用户通过用户名和密码登录，然后使用 TOTP 完成身份验证。用户设置并验证用户名和密码后，就可以为 MFA 激活 TOTP 软件令牌。如果应用程序使用 Amazon Cognito 托管 UI 来登录用户，则用户提交用户名和密码，然后在额外的登录页面上提交 TOTP 密码。

您可以在 Amazon Cognito 控制台中为用户池激活 TOTP MFA，也可以使用 Amazon Cognito API 操作来激活。在用户池级别，您可以通过调用 SetUserPoolMfaConfig 配置 MFA 和启用 TOTP MFA。

> **Note**
>
> 如果您没有为用户池激活 TOTP 软件令牌 MFA，则 Amazon Cognito 无法使用此令牌进行关联或验证用户。在这种情况下，用户会收到 `SoftwareTokenMFANotFoundException` 异常，并带有说明 `Software Token MFA has not been enabled by the userPool`。如果您稍后为用户池停用了软件令牌 MFA，则以前已关联并验证 TOTP 令牌的用户可以继续将其用于 MFA。

为用户配置 TOTP 是一个多步骤过程，在此过程中，用户将收到一个秘密代码，它们通过输入一次性密码来验证该代码。接下来，您可以为用户启用 TOTP MFA，或将 TOTP 设置为用户的首选 MFA 方法。

当您将用户群体配置为需要 TOTP MFA 并且用户在托管 UI 中注册应用程序时，Amazon Cognito 会自动执行用户流程。Amazon Cognito 提示您的用户选择 MFA 方法，显示 QR 代码来设置身份验证器应用程序，并验证他们的 MFA 注册。在您允许用户在 SMS 和 TOTP MFA 之间进行选择的用户群体中，Amazon Cognito 还为您的用户提供了方法选择。有关托管 UI 注册体验的更多信息，请参阅 如何在 Amazon Cognito 托管 UI 中注册新账户 (p. 72)。

> Important
>
> 当您将 AWS WAF Web ACL 与用户群体关联并且 Web ACL 中的规则提供 CAPTCHA 时，这可能会导致托管 UI TOTP 注册中出现不可恢复的错误。要创建具有 CAPTCHA 操作且不影响托管 UI TOTP 的规则，请参阅 为托管 UI TOTP MFA 配置 AWS WAF Web ACL (p. 224)。有关 AWS WAF Web ACL 和 Amazon Cognito 的更多信息，请参阅 将 AWS WAF Web ACL 与用户池相关联 (p. 236)。

要在使用 Amazon Cognito API 的自定义 UI 中实施 TOTP MFA，请参阅 在 Amazon Cognito 用户群体 API 中为用户配置 MFA (p. 223)。

要向用户池添加 MFA，请参阅 向用户池添加 MFA (p. 220)。

TOTP MFA 注意事项和限制

1. Amazon Cognito 通过可生成 TOTP 代码的身份验证器应用程序支持软件令牌 MFA。Amazon Cognito 不支持基于硬件的 MFA。
2. 当您的用户池要求尚未进行配置的用户提供 TOTP 时，您的用户将收到一个一次性访问令牌，应用程序可使用该令牌为用户激活 TOTP MFA。在用户注册了额外的 TOTP 登录安全要素之前，后续的登录尝试将失败。
   - 通过 `SignUp` API 操作或通过托管 UI 登录用户池的用户，在完成注册时将收到一次性的令牌。
   - 在您创建用户并且用户设置初始密码后，Amazon Cognito 会从托管 UI 向用户颁发一次性令牌。如果您为用户设置了永久密码，则 Amazon Cognito 会在用户首次登录时颁发一次性令牌。
   - 对于由管理员创建且使用 InitiateAuth 或 AdminInitiateAuth API 操作登录的用户，Amazon Cognito 不会向其颁发一次性令牌。在您的用户成功完成质询来设置初始密码后，或者如果您为用户设置永久密码，Amazon Cognito 会立即向用户提出质询来设置 MFA。
3. 如果用户池中需要 MFA 的用户已收到一次性访问令牌但尚未设置 TOTP MFA，则在设置 MFA 之前，用户无法使用托管 UI 登录。您可以不使用访问令牌，而是使用来自 `MFA_SETUP` 质询的 `session` 响应值，在 AssociateSoftwareToken 请求中进行 InitiateAuth 或 AdminInitiateAuth。
4. 如果您的用户已设置 TOTP，则他们可以将其用于 MFA，即使您以后停用用户池的 TOTP。

在 Amazon Cognito 用户群体 API 中为用户配置 MFA

当用户首次登录时，您的应用程序使用他们的一次性访问令牌生成 TOTP 私钥，并以文本或二维码格式将其呈现给用户。您的用户配置其身份验证器应用程序并为后续登录尝试提供 TOTP。您的应用程序或托管 UI 在 MFA 质询响应中向 Amazon Cognito 提交 TOTP。

主题

- 关联 TOTP 软件令牌 (p. 223)
- 验证 TOTP 令牌 (p. 224)
- 使用 TOTP MFA 登录 (p. 224)
- 删除 TOTP 令牌 (p. 224)

关联 TOTP 软件令牌

要关联 TOTP 令牌，请向用户发送一个必须使用一次性密码来验证的秘密代码。关联令牌需要执行三个步骤。

1. 当您的用户选择 TOTP 软件令牌 MFA 时，调用 AssociateSoftwareToken 来返回为用户账户生成的唯一共享私有密钥代码。您可以使用访问令牌或会话字符串授权 AssutatSoftwAreToken。
2. 您的应用程序向用户呈现私钥或您从私钥生成的二维码。用户必须将密钥输入到一个生成 TOTP 的应用程序（如 Google Authenticator）中。您可以使用 libqrencode 生成二维码。
3. 您的用户用身份验证器应用程序（例如 Google Authenticator）输入密钥或扫描二维码，该应用程序开始生成代码。

验证 TOTP 令牌

接下来验证 TOTP 令牌。要求您的用户提供示例代码并将其提供给 Amazon Cognito 服务，以确认用户成功生成 TOTP 代码，如下所示。

1. 您的应用程序会提示用户输入代码，以证明他们已正确设置了身份验证器应用程序。
2. 用户的身份验证器应用程序显示一个临时密码。身份验证器应用程序根据您提供给用户的密钥生成密码。
3. 用户输入他们的临时密码。您的应用程序在 `VerifySoftwareToken` API 请求中将临时密码传递给 Amazon Cognito。
4. Amazon Cognito 保留与用户关联的密钥，并生成 TOTP 并将其与用户提供的 TOTP 进行比较。如果匹配，`VerifySoftwareToken` 返回 `SUCCESS` 响应。
5. Amazon Cognito 将 TOTP 要素与用户关联起来。
6. 如果 `VerifySoftwareToken` 操作返回 `ERROR` 响应，请确保用户的时钟正确且没有超过最大重试次数。Amazon Cognito 接受在尝试前后 30 秒之内的 TOTP 令牌，以容许轻微的时钟偏差。解决问题后，请再次尝试 VerifySoftwareToken 操作。

使用 TOTP MFA 登录

此时，您的用户可使用基于时间的一次性密码登录。过程如下所述。

1. 用户将输入其用户名和密码来登录客户端应用程序。
2. TOTP MFA 质询被调用，您的应用程序提示用户输入临时密码。
3. 您的用户从关联的 TOTP 生成应用程序获取临时密码。
4. 您的用户在您的客户端应用程序中输入 TOTP 代码。您的应用程序通知 Amazon Cognito 服务以验证该代码。每次登录时，都应调用 RespondToAuthChallenge，以获取对新的 TOTP 身份验证质询的响应。
5. 如果令牌通过 Amazon Cognito 验证，则登录成功，您的用户可以继续完成身份验证流程。

删除 TOTP 令牌

最后，您的应用程序应允许用户删除 TOTP 令牌：

1. 客户端应用程序应要求用户输入其密码。
2. 如果密码正确，则删除 TOTP 令牌。

> **Note**
>
> 当前，您无法删除用户的 TOTP 软件令牌。要替换用户的软件令牌，请关联并验证新的软件令牌。要为用户停用 TOTP MFA，请调用 SetUserMFAPreference 来修改用户，使其不使用 MFA 或仅使用 SMS MFA。

为托管 UI TOTP MFA 配置 AWS WAF Web ACL

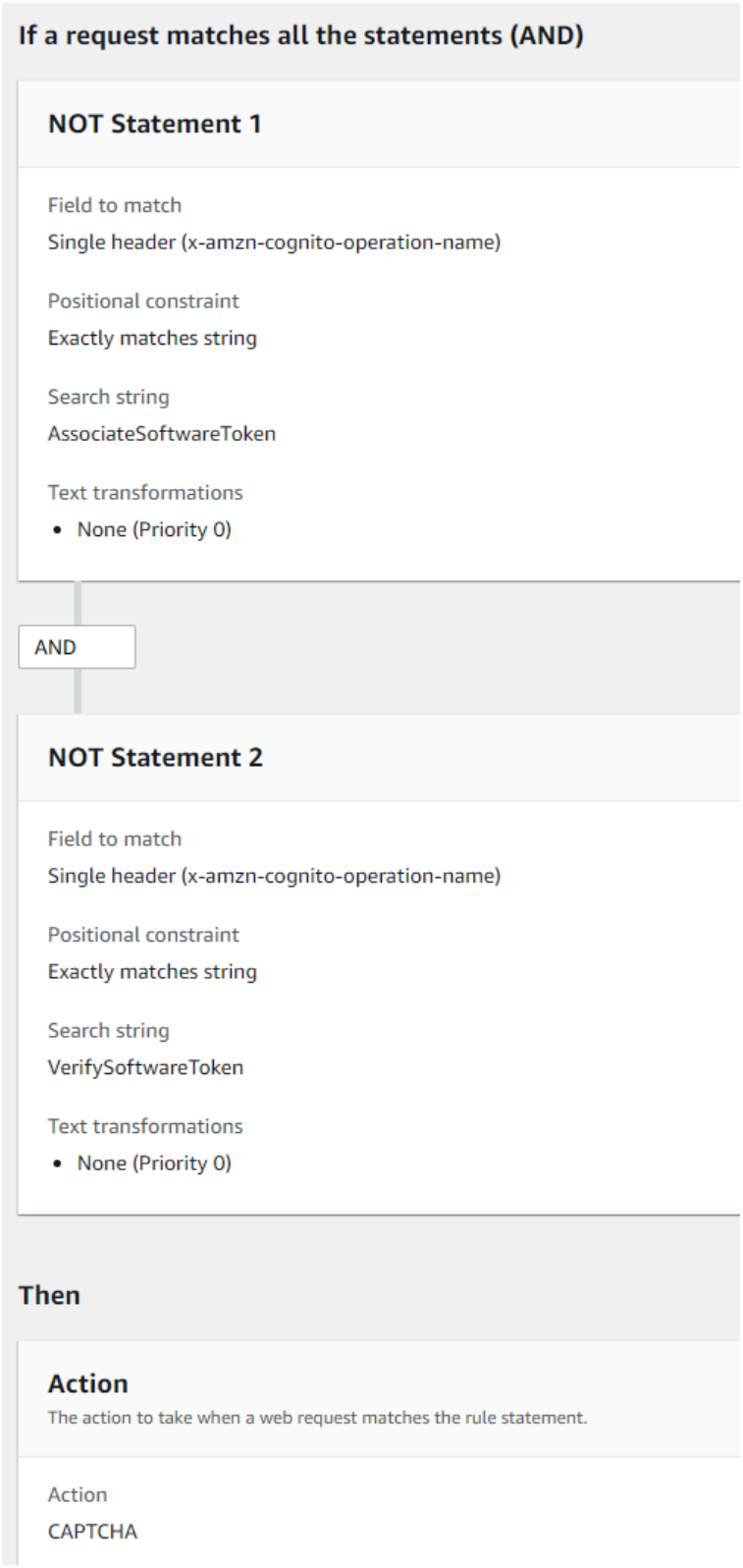
当您将 AWS WAF Web ACL 与用户群体关联并且 Web ACL 中的规则提供 CAPTCHA 时，这可能会导致托管 UI TOTP 注册中出现不可恢复的错误。AWS WAFCAPTCHA 规则仅以这种方式影响托管 UI 中的 TOTP MFA。SMS MFA 不受影响。

当 CAPTCHA 规则不允许用户完成 TOTP MFA 设置时，Amazon Cognito 会显示以下错误。

Request not allowed due to WAF captcha.（由于 WAF captcha，不允许请求。）

当 AWS WAF 提示提供 CAPTCHA 来响应您的用户群体在后台发出的 AssociateSoftwareToken 和 VerifySoftwareToken API 请求时，会出现此错误。要创建具有 CAPTCHA 操作且不影响托管 UI TOTP 的规则，请从您的规则中的 CAPTCHA 操作中排除 `AssociateSoftwareToken` 和 `VerifySoftwareToken` 的 `x-amzn-cognito-operation-name` 标头值。

以下屏幕截图显示一个示例 AWS WAF 规则，该规则将 CAPTCHA 操作应用于所有没有 `AssociateSoftwareToken` 或 `VerifySoftwareToken` 的 `x-amzn-cognito-operation-name` 标头值的请求。



有关 AWS WAF Web ACL 和 Amazon Cognito 的更多信息，请参阅 将 AWS WAF Web ACL 与用户池相关联 (p. 236)。

向用户池添加高级安全

在创建用户池之后，即可访问 Amazon Cognito 控制台中导航栏上的 Advanced security (高级安全)。可以打开用户池高级安全功能并自定义为响应不同风险要执行的操作。或者，您可以使用审计模式收集与检测到的风险相关的指标，而无需应用任何安全缓解措施。在审核模式下，高级安全功能会将指标发布到 Amazon CloudWatch。在 Amazon Cognito 生成其第一个高级安全事件后，您可以看到高级安全指标。请参阅查看高级安全指标 (p. 234)。

高级安全功能包括已泄露凭证检测 和自适应身份验证。

已泄露的凭证

用户将密码重复用于多个用户账户。Amazon Cognito 的已泄露凭证功能可编译公开泄露的用户名和密码数据，并将用户的凭证与泄露的凭证列表进行比较。已泄露凭证的检测还检查常猜测的密码。

您可以选择用于提示检查已泄露凭证的用户操作，以及您希望 Amazon Cognito 采取的应对措施。对于登录、注册和密码更改事件，Amazon Cognito 可以禁止登录或允许登录。在这两种情况下，Amazon Cognito 都会生成用户活动日志，您可以在其中找到有关该事件的更多信息。

自适应身份验证

Amazon Cognito 可以查看来自用户登录请求的位置和设备信息，并应用自动响应来保护用户群体中的用户账户免受可疑活动的侵害。

当您激活高级安全时，Amazon Cognito 会为用户活动分配风险评分。您可以为可疑活动指定自动响应：您可以需要 MFA、禁止登录，或者只记录活动详细信息和风险评分。您还可以自动发送电子邮件，将可疑活动通知用户，以便他们可以重置密码或采取其他自助操作。

主题

- 注意事项和限制 (p. 227)
- 先决条件 (p. 228)
- 配置高级安全功能 (p. 228)
- 检查盗用凭证 (p. 230)
- 使用自适应身份验证 (p. 230)
- 查看高级安全指标 (p. 234)
- 通过应用程序激活用户池高级安全功能 (p. 235)

注意事项和限制

- Amazon Cognito 高级安全功能适用其他定价。请参阅 Amazon Cognito 定价页面。
- Amazon Cognito 支持在以下标准身份验证流中使用高级安全功能：`USER_PASSWORD_AUTH`、`ADMIN_USER_PASSWORD_AUTH`、`USER_SRP_AUTH` 和 `ADMIN_USER_SRP_AUTH`。无法在 `CUSTOM_AUTH` 流和 自定义身份验证质询 Lambda 触发器 (p. 134) 中或者通过联合登录使用高级安全功能。
- 使用完整功能模式下的 Amazon Cognito 高级安全功能，您可以创建始终阻止和始终允许 IP 地址例外。对于来自 Always block（始终阻止）例外列表中 IP 地址的会话，自适应身份验证不会向其分配风险级别，该会话也无法登录您的用户池。
- 在您的用户池中阻止来自 Always block（始终阻止）例外列表中 IP 地址的请求，可以帮助您的用户池保持在请求速率配额以内。Amazon Cognito 高级安全功能无法防止分布式拒绝服务 (DDoS) 攻击。有关如何阻止 Amazon Cognito 端点上不需要流量的信息，请参阅 AWS 安全博客 中的 Protect public clients for Amazon Cognito by using an Amazon CloudFront proxy（使用 Amazon CloudFront 代理保护 Amazon Cognito 的公共客户端）。

先决条件

在开始之前，您需要：

- 用户池和应用程序客户端。有关更多信息，请参阅 用户池入门 (p. 30)。
- 在 Amazon Cognito 控制台中将多重身份验证（MFA）设置为 Optional (可选),以使用基于风险的自适应身份验证功能。有关更多信息，请参阅向用户池添加 MFA (p. 220)。
- 如果您使用电子邮件通知，请转到 Amazon SES 控制台配置并验证要用于通知电子邮件的电子邮件地址或域。有关 Amazon SES 的更多信息，请参阅在 Amazon SES 中验证身份。

配置高级安全功能

您可以在 AWS Management Console 中配置 Amazon Cognito 高级安全功能。

为用户池配置高级安全功能

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 App integration（应用程序集成）选项卡。找到 Advanced security（高级安全），然后选择 Enable（启用）。如果之前启用了高级安全，请选择 Edit（编辑）。
5. 选择 Full function（完整功能）以配置对盗用凭证和自适应身份验证的高级安全响应。选择 Audit only（仅审计）以收集信息并将用户池数据发送到 CloudWatch。高级安全定价适用于 Audit only（仅审计）和 Full function（完整功能）模式。有关更多信息，请参阅 Amazon Cognito 定价。

   建议在启用操作之前，先将高级安全功能保持在审核模式下两周。在此期间，Amazon Cognito 可以了解应用程序用户的使用模式。
6. 如果您已选择 Audit only（仅审计），请选择 Save changes（保存更改）。如果您已选择 Full function（完整功能）：

   a. 选择是否要进行 Custom（自定义）操作或使用 Cognito defaults（Cognito 默认设置）响应可疑 Compromised credentials（盗用凭证）。Cognito defaults（Cognito 默认设置）：

      i. 检测 Sign-in（登录）、Sign-up（注册）和 Password change（密码更改）中的盗用凭证。
      ii. 使用 Block sign-in（阻止登录）操作响应盗用凭证。

   b. 如果您为 Compromised credentials（盗用凭证）选择了 Custom（自定义）操作，请选择 Amazon Cognito 将用于 Event detection（事件检测）的用户池操作，以及您希望 Amazon Cognito 执行的 Compromised credentials responses（盗用凭证响应）。您可以使用可疑盗用凭证进行 Block sign-in（阻止登录）或 Allow sign-in（允许登录）。

   c. 在 Adaptive authentication（自适应身份验证）下，选择如何响应恶意登录尝试。选择是否要进行 Custom（自定义）操作或使用 Cognito defaults（Cognito 默认设置）响应可疑恶意活动。当您选择 Cognito defaults（Cognito 默认设置）时，Amazon Cognito 会阻止所有风险级别的登录，并且不会通知用户。

   d. 如果您针对 Adaptive authentication（自适应身份验证）已选择 Custom（自定义操作），请根据严重性级别选择 Amazon Cognito 对检测到的风险执行的 Automatic risk response（自动风险响应）操作。当您针对风险级别分配响应时，您无法为较高风险级别分配限制性较小的响应。您可以为风险级别分配以下响应：

      i. Allow sign-in（允许登录）– 不采取任何预防性操作。
      ii. Optional MFA（可选 MFA）– 如果用户配置了 MFA，Amazon Cognito 将始终要求用户在登录时提供其它 SMS 或基于时间的一次性密码（TOTP）因素。如果用户没有配置 MFA，他们可以继续正常登录。
      iii. Require MFA（需要 MFA）– 如果用户配置了 MFA，Amazon Cognito 将始终要求用户在登录时提供其它短信或 TOTP 因素。如果用户没有配置 MFA，Amazon Cognito 将提示他们设置

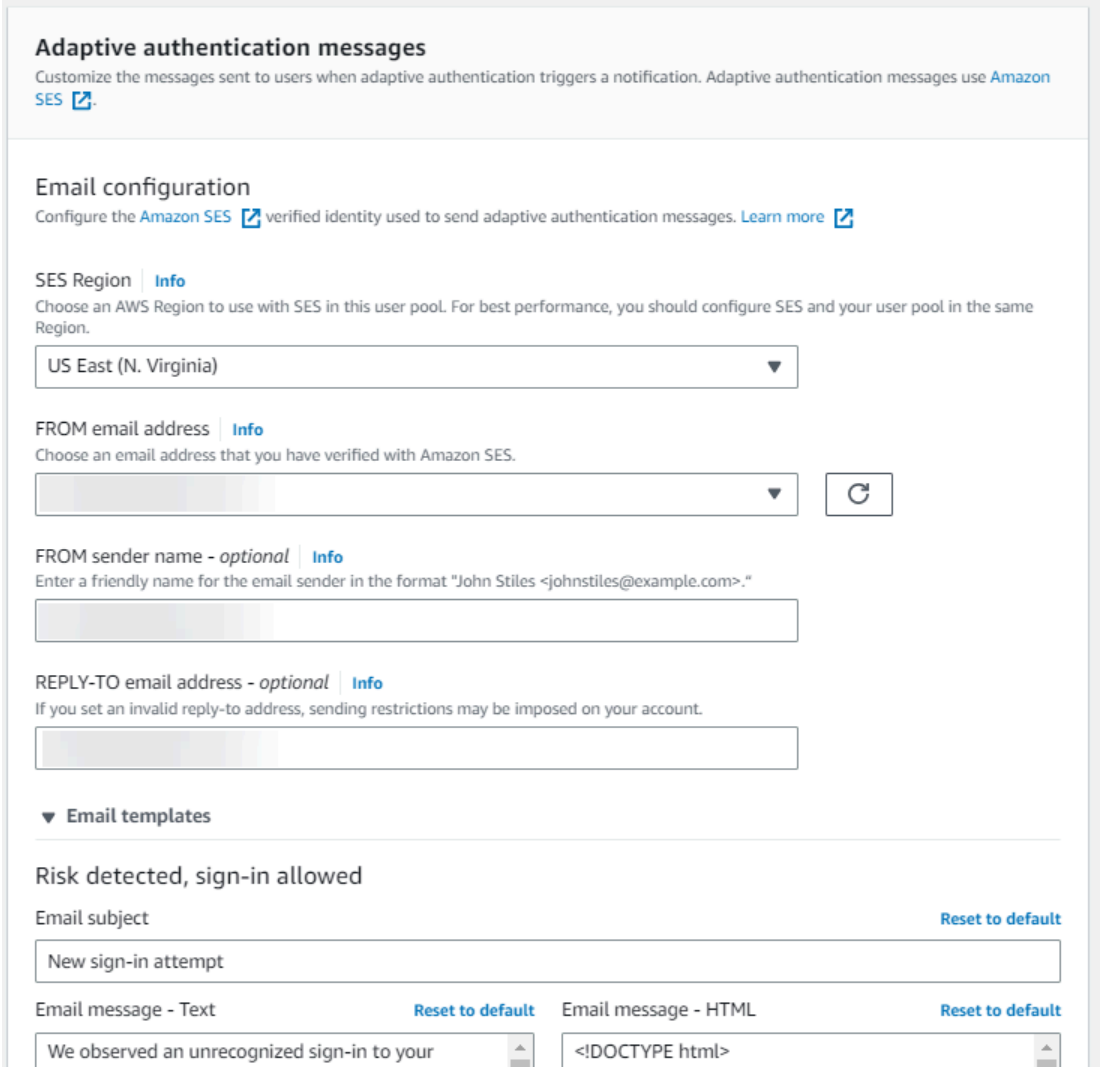
MFA。在您自动要求用户使用 MFA 之前，请在应用程序中配置一种机制来捕获 SMS MFA 的电话号码，或为 TOTP MFA 注册身份验证器应用程序。

iv. Block sign-in（阻止登录）– 阻止用户登录。

v. Notify user（通知用户）– 向用户发送电子邮件，其中包含有关 Amazon Cognito 检测到的风险以及您所采取的响应的信息。您可以为发送的消息自定义电子邮件消息模板。

7. 如果您在上一步骤中选择了 Notify user（通知用户），您可以自定义电子邮件发送设置和电子邮件消息模板以进行自适应身份验证。

   a. 在 Email configuration（邮件配置）下，选择您希望与自适应身份验证一起使用的 SES Region（SES 区域）、FROM email address（FROM 电子邮件地址）、FROM sender name（FROM 发件人名称）和 REPLY-TO email address（REPLY-TO 电子邮件地址）。有关将用户池电子邮件消息与 Amazon Simple Email Service 集成的更多信息，请参阅 Amazon Cognito 用户池的电子邮件设置。



   b. 展开 Email templates（电子邮件模板）以自定义包含 HTML 和纯文本版本电子邮件消息的自适应身份验证通知。要了解有关电子邮件消息模板的更多信息，请参阅消息模板 (p. 254)。

8. 展开 IP address exceptions（IP 地址异常）以创建 Always-allow（始终允许）或者 Always-block（始终阻止）的 IPv4 或 IPv6 地址清单，该清单将始终允许或阻止其 IPv4 或 IPv6 地址范围，无论高级安全风险评估如何。用 CIDR 表示法指定 IP 地址范围（例如，192.168.100.0/24）。

9. 选择 Save changes（保存更改）。

检查盗用凭证

Amazon Cognito 可以检测用户的用户名和密码是否已在别处遭盗用。这种情况可能出现在用户在多个站点重复使用凭证时或它们使用不安全的密码时。Amazon Cognito 在托管 UI 中和通过 Amazon Cognito API 检查使用用户名和密码登录的本地用户。本地用户仅存在于您的用户群体目录中，无需通过外部 IdP 进行联合身份验证。

从 Amazon Cognito 控制台的 App integration（应用程序集成）选项卡的 Advanced security（高级安全功能）中，您可以配置 Compromised credentials（已泄漏的凭证）。配置 Event detection（事件检测）以选择要监控的用户事件，以查看是否有已泄漏的凭证。配置 Compromised credentials responses（已泄露凭证响应），以选择在检测到已泄露的凭证时是允许还是阻止用户。Amazon Cognito 可以在登录、注册和密码更改期间检查是否有已泄露的凭证。

当您选择 Allow sign-in（允许登录）时，您可以查看 Amazon CloudWatch Logs 以监控 Amazon Cognito 对用户事件进行的评估。有关更多信息，请参阅查看高级安全指标 (p. 234)。当您选择 Block sign-in（禁止登录）时，Amazon Cognito 会阻止使用已泄露的凭证的用户登录。当 Amazon Cognito 阻止用户登录时，它将用户的 `UserStatus` 设置为 `RESET_REQUIRED`。具有 `RESET_REQUIRED` 状态的用户必须先更改密码，然后才能再次登录。

> Note
>
> 当前，对于采用安全远程密码（SRP）流程的登录操作，Amazon Cognito 不会检查是否有已泄露的凭证。SRP 在登录期间发送经过哈希处理的密码证明。Amazon Cognito 无法在内部访问密码，因此，它只能评估您的客户端以纯文本形式传递给它的密码。
> Amazon Cognito 会检查使用 AdminInitiateAuth API（具有 `ADMIN_USER_PASSWORD_AUTH` 流）和 InitiateAuth API（具有 `USER_PASSWORD_AUTH` 流）的登录，查看是否存在泄露的凭证。

要向用户池添加遭盗用凭证保护，请参阅 向用户池添加高级安全 (p. 227)。

使用自适应身份验证

借助自适应身份验证，可以将用户池配置为阻止可疑登录，或为响应增加的风险级别添加第二安全要素身份验证。对于每次登录尝试，Amazon Cognito 都会生成一个风险分数来表示登录请求来自遭盗用源的可能性。该风险评分基于包括设备和用户信息在内的因素。当 Amazon Cognito 检测到用户会话中存在风险且用户尚未选择多重身份验证 (MFA) 方法时，自适应身份验证可以为用户群体中的用户开启或要求 MFA。当您为用户激活 MFA 时，无论您如何配置自适应身份验证，他们都会收到在身份验证期间提供或设置第二因素的质询。从用户的角度来看，您的应用程序可以帮助他们设置 MFA，也可以选择让 Amazon Cognito 阻止他们再次登录，直到他们配置了附加因素。

Amazon Cognito 将向 Amazon CloudWatch 发布登录尝试、其风险级别和失败的质询。有关更多信息，请参阅查看高级安全指标 (p. 234)。

要向用户池添加自适应身份验证，请参阅 向用户池添加高级安全 (p. 227)。

主题

- 自适应身份验证概览 (p. 230)
- 将用户设备和会话数据添加到 API 请求 (p. 231)
- 查看用户事件历史记录 (p. 233)
- 提供事件反馈 (p. 233)
- 发送通知消息 (p. 234)

自适应身份验证概览

在 Amazon Cognito 控制台的应用程序集成选项卡中的高级安全上，您可以选择自适应身份验证的设置，包括在不同风险等级下采取什么操作，以及向用户发送的通知消息的自定义设置。您可以为所有应用程序客户端分配全局高级安全配置，但将客户端级配置应用于各个应用程序客户端。

Amazon Cognito 自适应身份验证为每个用户会话分配以下风险级别之一：高、中、低或无风险。

将 Enforcement method（执行方法）从 Audit-only（仅限审计）更改为 Full-function（完整功能）时，请仔细斟酌您的选项。您对风险等级应用的自动响应会影响 Amazon Cognito 为具有相同特征的后续用户会话分配的风险等级。例如，在您选择不采取任何操作或 Allow（允许）之后，对于 Amazon Cognito 最初评估为高风险的用户会话，Amazon Cognito 会认为类似会话的风险较低。

对于每个风险级别，您可以选择以下选项：

选项	操作
Allow（允许）	用户无需额外安全要素即可登录。
Optional MFA（可选 MFA）	已配置第二安全要素的用户需要完成第二安全要素质询才能登录。用于 SMS 的电话号码和 TOTP 软件令牌是可供使用的第二个安全要素。未配置第二个因素的用户只能使用一组凭证登录。
Require MFA（需要 MFA）	已配置第二安全要素的用户需要完成第二安全要素质询才能登录。Amazon Cognito 阻止未配置第二个安全要素的用户登录。
Block（阻止）	Amazon Cognito 阻止指定风险级别下的所有登录尝试。

Note

您无需验证手机号码即可将其用于 SMS 来作为第二个身份验证要素。

将用户设备和会话数据添加到 API 请求

当您使用 API 对用户进行注册、登录和重置密码时，可以收集用户会话的相关信息并将其传递给 Amazon Cognito 高级安全功能。此信息包括用户的 IP 地址和唯一的设备标识符。

您可能在用户和 Amazon Cognito 之间有中间网络设备，例如代理服务或应用程序服务器。您可以收集用户的上下文数据并将其传递给 Amazon Cognito，以便自适应身份验证根据用户端点（而不是服务器或代理）的特征来计算风险。如果您的客户端应用程序直接调用 Amazon Cognito API 操作，自适应身份验证会自动记录源 IP 地址。但是，它不会记录其他设备信息（例如 `user-agent`），除非您也收集设备指纹。

使用 Amazon Cognito 上下文数据收集库生成此数据，然后使用 ContextData 和 UserContextData 参数将它提交到 Amazon Cognito 高级安全功能。上下文数据收集库包含在 AWS SDK 中。有关更多信息，请参阅将 Amazon Cognito 与 Web 和移动应用程序集成。如果您已在用户群体中激活了高级安全功能，则可以提交 `ContextData`。有关更多信息，请参阅配置高级安全功能。

当您从应用程序服务器调用以下经过 Amazon Cognito 身份验证的 API 操作时，请在 `ContextData` 参数中传递用户设备的 IP。此外，请传递服务器名称、服务器路径和编码的设备指纹数据。

- AdminInitiateAuth
- AdminRespondToAuthChallenge

当您调用 Amazon Cognito 未经身份验证的 API 操作时，您可以将 `UserContextData` 提交到 Amazon Cognito 高级安全功能。此数据在 `EncodedData` 参数中包括设备指纹。如果您符合以下条件，也可以在 `UserContextData` 中提交 `IpAddress` 参数：

- 您已在用户群体中激活了高级安全功能。有关更多信息，请参阅配置高级安全功能。
- 您的应用程序客户端具有客户端密钥。有关更多信息，请参阅配置用户群体应用程序客户端。
- 您已在应用程序客户端中激活 Accept additional user context data（接受其他用户上下文数据）。有关更多信息，请参阅 接受额外的用户上下文数据（AWS Management Console）（p. 232）。

您的应用程序可以在以下 Amazon Cognito 未经验证的 API 操作中，使用编码的设备指纹数据和用户设备的 IP 地址填充 `UserContextData` 参数。

- InitiateAuth
- RespondToAuthChallenge
- SignUp
- ConfirmSignUp
- ForgotPassword
- ConfirmForgotPassword
- ResendConfirmationCode

接受额外的用户上下文数据（AWS Management Console）

激活 Accept additional user context data（接受其他用户上下文数据）功能后，用户群体在 `UserContextData` 参数中接受 IP 地址。在以下情况下，您无需激活此功能：

- 您的用户只能使用经过身份验证的 API 操作（如 AdminInitiateAuth）进行登录，而您使用 `ContextData` 参数。
- 您只希望未经身份验证的 API 操作向 Amazon Cognito 高级安全功能发送设备指纹，而不是 IP 地址。

在 Amazon Cognito 控制台中按如下方式更新应用程序客户端，以添加对其他用户上下文数据的支持。

1. 登录 Amazon Cognito 控制台。
2. 在导航窗格中，选择 Manage your User Pools（管理您的用户池），然后选择要编辑的用户池。
3. 选择 App integration（应用程序集成）选项卡。
4. 在 App clients and analytics（应用程序客户端和分析）下，选择或创建应用程序客户端。有关更多信息，请参阅配置用户群体应用程序客户端。
5. 从 App client information（应用程序客户端信息）容器中选择 Edit（编辑）。
6. 在应用程序客户端的 Advanced authentication settings（高级身份验证设置）下，选择 Accept additional user context data（接受其他用户上下文数据）。
7. 选择 Save changes（保存更改）。

要将应用程序客户端配置为在 Amazon Cognito API 中接受用户上下文数据，请在 CreateUserPoolClient 或 UpdateUserPoolClient 请求中将 `EnablePropagateAdditionalUserContextData` 设置为 `true`。有关从 Web 或移动应用程序激活高级安全功能的信息，请参阅从应用程序中激活用户群体高级安全功能。当您的应用程序从服务器调用 Amazon Cognito 时，从客户端收集用户上下文数据。以下是使用 JavaScript 开发工具包方法 `getData` 的示例。

```
var encodedData =
 AmazonCognitoAdvancedSecurityData.getData(username, userPoolId, clientId);
```

在设计应用程序以使用自适应身份验证时，我们建议您将最新的 Amazon Cognito 开发工具包集成到应用程序中。最新版本的开发工具包收集设备指纹信息，如设备 ID、模型和时区。有关 Amazon Cognito SDK 的更多信息，请参阅安装用户池 SDK。Amazon Cognito 高级安全功能只为应用程序以正确格式提交的事件保存和分配风险评分。如果 Amazon Cognito 返回错误响应，请检查您的请求是否包含有效的密钥哈希值，以及 `IPaddress` 参数是否为有效的 IPv4 或 IPv6 地址。

`ContextData` 和 `UserContextData` 资源

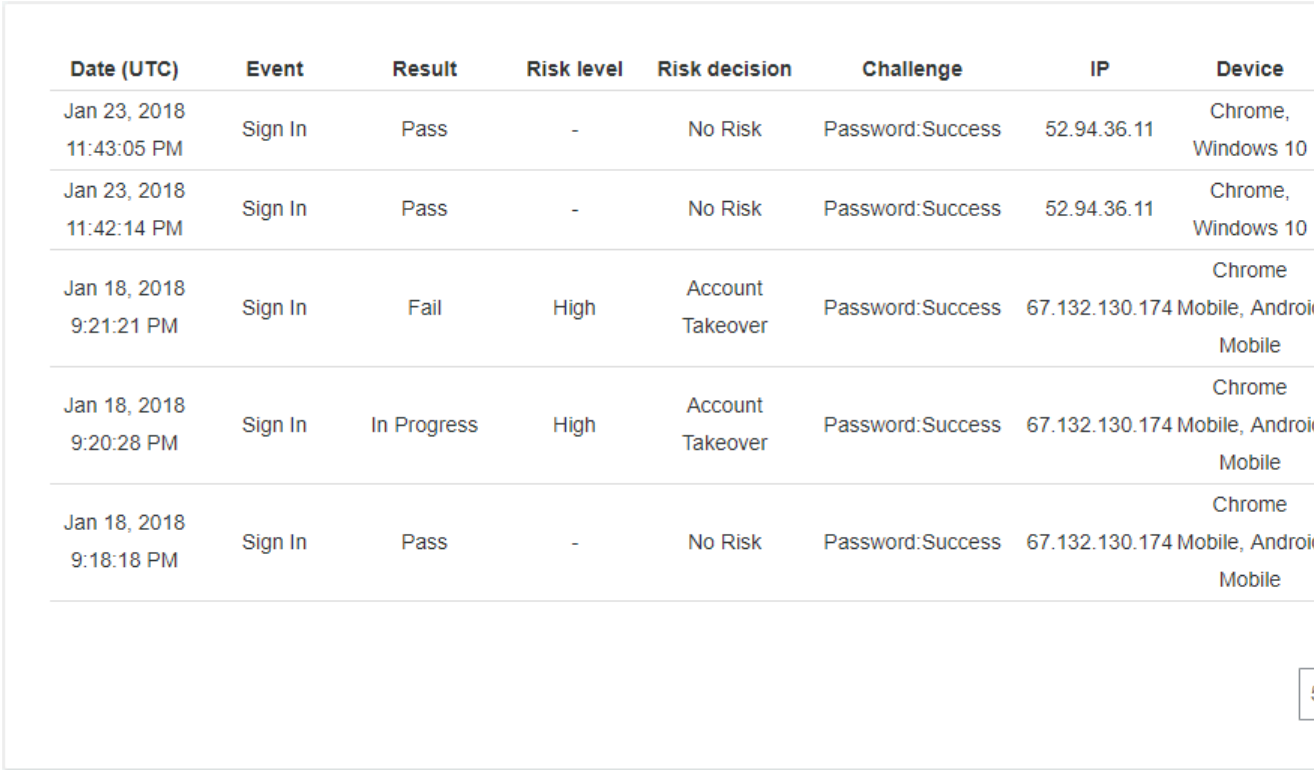
- 适用于 Android 的 AWS Amplify SDK：GetUserContextData
- 适用于 iOS 的 AWS Amplify SDK：userContextData
- JavaScript：amazon-cognito-advanced-security-data.min.js

查看用户事件历史记录

Note

在新的 Amazon Cognito 控制台中，您可以在 Users（用户）选项卡中查看用户事件历史记录。

要查看某个用户的登录历史记录，您可以在 Amazon Cognito 控制台的 Users（用户）选项卡中选择该用户。Amazon Cognito 会将用户事件历史记录保留两年。



Date (UTC)	Event	Result	Risk level	Risk decision	Challenge	IP	Device
Jan 23, 2018 11:43:05 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10
Jan 23, 2018 11:42:14 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10
Jan 18, 2018 9:21:21 PM	Sign In	Fail	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Androi Mobile
Jan 18, 2018 9:20:28 PM	Sign In	In Progress	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Androi Mobile
Jan 18, 2018 9:18:18 PM	Sign In	Pass	-	No Risk	Password:Success	67.132.130.174	Chrome Mobile, Androi Mobile

每个登录事件都有一个事件 ID。该事件还包含对应的上下文数据，如位置、设备详细信息和风险检测结果。您可以使用 Amazon Cognito API 操作 AdminListUserAutEvents 或者使用 AWS Command Line Interface (AWS CLI) 通过 admin-list-user-auth-events 来查询用户事件历史记录。

您还可以将事件 ID 与 Amazon Cognito 在记录事件时发放的令牌关联起来。ID 和访问令牌在其有效负载中包含此事件 ID。Amazon Cognito 还将刷新令牌的使用与原始事件 ID 相关联。您可以通过原始事件 ID 追溯到导致颁发 Amazon Cognito 令牌的登录事件的事件 ID。您可以跟踪系统中的令牌在特定身份验证事件中的使用。有关更多信息，请参阅将令牌与用户池结合使用 (p. 203)。

提供事件反馈

事件反馈实时影响风险评估，并随着时间的推移改进风险评估算法。您可以通过 Amazon Cognito 控制台和 API 操作提供有关登录尝试有效性的反馈。

Note

您的事件反馈会影响 Amazon Cognito 为具有相同特征的后续用户会话分配的风险等级。

在 Amazon Cognito 控制台中，从 Users（用户）选项卡中选择一个用户，然后选择 Provide event feedback（提供事件反馈）。您可以查看事件详细信息，并选择 Set as valid（设为有效）或 Set as invalid（设为无效）。

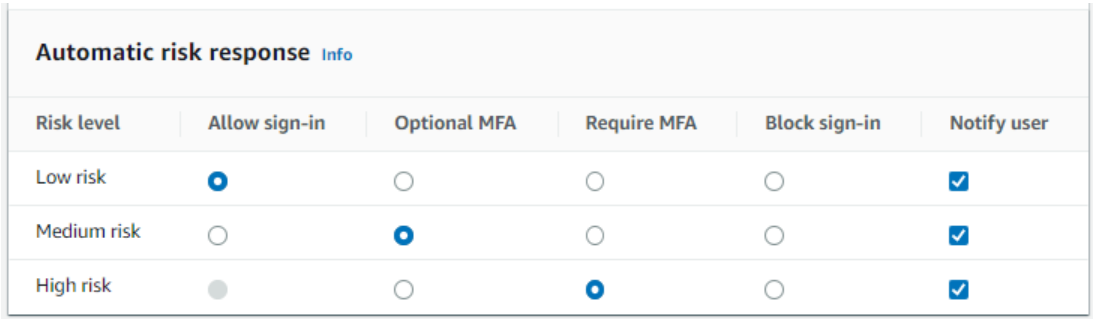
控制台的 Users and groups（用户和组）选项卡中列出了登录历史记录。您可以选择某个条目来将事件标记为有效或无效。您还可以通过用户池 API 操作 AdminUpdateAuthEventFeedback、AWS CLI 命令 admin-update-auth-event-feedback 提供反馈。

当您在 Amazon Cognito 控制台中选择 Set as valid（设为有效）或在 API 中提供 `valid` 的 `FeedbackValue` 值时，您告诉 Amazon Cognito 您信任某个用户会话（Amazon Cognito 已在其中评估了某种风险等级）。当您在 Amazon Cognito 控制台中选择 Set as invalid（设为无效）或在 API 中提供 `invalid` 的 `FeedbackValue` 值时，您告诉 Amazon Cognito 您不信任某个用户会话，或者您不认为 Amazon Cognito 评估的风险等级足够高。

发送通知消息

借助高级安全保护措施，Amazon Cognito 可以通知您的用户存在有风险的登录尝试。Amazon Cognito 还可以提示用户选择链接以指示登录是有效还是无效。Amazon Cognito 使用此反馈来提高用户池的风险检测准确性。

在 Automatic risk response（自动风险响应）部分，对低、中或高风险案例选择 Notify Users（通知用户）。



无论您的用户是否验证了电子邮件地址，Amazon Cognito 都会向他们发送电子邮件通知。

您可以自定义通知电子邮件消息，并提供这些消息的纯文本和 HTML 版本。要自定义您的电子邮件通知，请在高级安全配置中，从 Adaptive authentication messages（自适应身份验证消息）中打开 Email templates（电子邮件模板）。要了解有关电子邮件模板的更多信息，请参阅 消息模板 (p. 254)。

查看高级安全指标

Amazon Cognito 将高级安全功能的指标发布到您在 Amazon CloudWatch 中的账户。Amazon Cognito 同时按风险级别和请求级别分组高级安全指标。

在 CloudWatch 控制台中查看指标

1. 访问 https://console.aws.amazon.com/cloudwatch/ 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Metrics (指标)。
3. 选择 Amazon Cognito。
4. 选择一组聚合指标，如 By Risk Classification（按风险分类）。
5. All metrics（所有指标）选项卡显示该选择的所有指标。您可执行以下操作：
   - 要对表进行排序，请使用列标题。
   - 要为指标绘制图表，请选中该指标旁的复选框。要选择所有指标，请选中表的标题行中的复选框。
   - 要按资源进行筛选，请选择资源 ID，然后选择 Add to search（添加到搜索）。
   - 要按指标进行筛选，请选择指标名称，然后选择 Add to search（添加到搜索）。

指标	描述	指标维度
CompromisedCredentialRisk	Amazon Cognito 在其中检测到泄露凭证的请求。	Operation：操作类型。仅有的维度是 `PasswordChange`、`SignIn` 或 `SignUp`。

指标	描述	指标维度
		UserPoolId：用户池标识符。 RiskLevel：高 (默认)、中、低。
AccountTakeoverRisk	Amazon Cognito 在其中检测到账户接管风险的请求。	Operation：操作类型。仅有的维度是 `PasswordChange`、`SignIn` 或 `SignUp`。 UserPoolId：用户池标识符。 RiskLevel：高、中、低。
OverrideBlock	因开发人员提供的配置而被 Amazon Cognito 阻止的请求。	Operation：操作类型。仅有的维度是 `PasswordChange`、`SignIn` 或 `SignUp`。 UserPoolId：用户池标识符。 RiskLevel：高、中、低。
Risk	Amazon Cognito 标记为有风险的请求。	Operation：操作类型，例如 `PasswordChange`、`SignIn` 或 `SignUp`。 UserPoolId：用户池标识符。
NoRisk	Amazon Cognito 在其中没有识别出任何风险的请求。	Operation：操作类型，例如 `PasswordChange`、`SignIn` 或 `SignUp`。 UserPoolId：用户池标识符。

Amazon Cognito 为您提供两组预定义指标以供在 CloudWatch 中分析。By Risk Classification（按风险分类）标识 Amazon Cognito 认定为有风险的请求的风险级别。By Request Classification（按请求分类）体现了按请求级别聚合的指标。

汇总指标组	描述
按风险分类	被 Amazon Cognito 标识为有风险的请求。
按请求分类	按请求汇总的指标。

通过应用程序激活用户池高级安全功能

在为用户池配置高级安全功能后，必须在 Web 或移动应用程序中激活这些功能。

对 JavaScript 使用高级安全功能

1. 将适用于 JavaScript 的 Amazon Cognito Identity SDK 添加到您的应用程序。
2. 在 CognitoUserPool.js 中，将 `AdvancedSecurityDataCollectionFlag` 设置为 `true`。将 `UserPoolId` 设置为用户群体 ID。
3. 将此源引用添加到您的应用程序的 JavaScript 文件中。使用 AWS 区域 替换 `<region>`，例如 `us-east-1`。

```
<script src="https://amazon-cognito-assets.<region>.amazoncognito.com/amazon-cognito-
advanced-security-data.min.js"></script>
```

对 Android 使用高级安全功能

1. 使用适用于 Android 的 AWS Amplify 创建您的应用程序。有关更多信息，请参阅《AWS Amplify 开发中心》中的项目设置。
2. 使用 `userContextDataProvider`，在身份验证请求中包含用户和设备信息。

有关在旧版 Android SDK 中添加用户上下文数据的信息，请参阅 aws-android-sdk-cognitoidentityprovider-asf。

对 iOS 使用高级安全功能

1. 使用适用于 Swift 或 Flutter 的 AWS Amplify 创建您的应用程序。有关更多信息，请参阅《AWS Amplify 开发中心》中的 Swift 项目设置和 Flutter 项目设置。
2. 在身份验证请求中包含用户和设备信息。有关要与 InitiateAuth API 操作一起使用的示例，请参阅 GitHub 上 InitiateAuthInput+Amplify.swift 中的 `userContextData`。

有关在旧版 iOS SDK 中添加用户上下文数据的信息，请参阅AWSCognitoIdentityProviderASF。

将 AWS WAF Web ACL 与用户池相关联

AWS WAF 是一个 Web 应用程序防火墙。使用 AWS WAF Web 访问控制列表 (Web ACL)，您可以保护您的用户池免受对托管 UI 和 Amazon Cognito API 服务端点的不必要请求的影响。Web ACL 使您可以对用户池响应的所有 HTTPS Web 请求进行精细控制。有关 AWS WAF Web ACL 的更多信息，请参阅 AWS WAF 开发人员指南 中的管理和使用 Web 访问控制列表 (web ACL)。

当您将 AWS WAF Web ACL 与用户群体关联时，Amazon Cognito 会将来自用户的选定非机密请求标头和内容转发到 AWS WAF。AWS WAF 会检查请求的内容，将其与您在 Web ACL 中指定的规则进行比较，然后向 Amazon Cognito 返回响应。

AWS WAF Web ACL 和 Amazon Cognito 须知信息

- 当您创建 Web ACL 时，Web ACL 要经过一小段时间才能完全传播并可供 Amazon Cognito 使用。传播时间可以从几秒钟到几分钟不等。当您尝试在 Web ACL 完全传播之前关联它时，AWS WAF 返回一个 `WAFUnavailableEntityException`。
- 您可以将一个 Web ACL 与一个用户池关联。
- 您的请求可能会导致负载超出 AWS WAF 可以检查的负载限制。请参阅 AWS WAF 开发人员指南 中的 超大请求组件处理，了解如何配置 AWS WAF 处理来自 Amazon Cognito 的超大请求的方式。
- 您无法将使用 AWS WAF 欺诈控制账户接管预防 (ATP) 的 Web ACL 与 Amazon Cognito 用户池关联。在添加 `AWS-AWSManagedRulesATPRuleSet` 托管规则组时实施 ATP 功能。在将您的 Web ACL 与用户池关联之前，请确保该 Web ACL 不使用此托管规则组。
- 当您将 AWS WAF Web ACL 与用户群体关联并且 Web ACL 中的规则提供 CAPTCHA 时，这可能会导致托管 UI TOTP 注册中出现不可恢复的错误。要创建具有 CAPTCHA 操作且不影响托管 UI TOTP 的规则，请参阅 为托管 UI TOTP MFA 配置 AWS WAF Web ACL (p. 224)。

AWS WAF 检查对以下端点的请求。

托管 UI

对 用户群体联合身份验证端点和托管 UI 参考 (p. 551) 中所有端点的请求。

公有 API 操作

从您的应用程序向 Amazon Cognito API 发出的请求不使用 AWS 凭证进行授权。这包括 InitiateAuth、RespondToAuthChallenge 和 GetUser 等 API 操作。AWS WAF 范围内的 API 操作不需要使用 AWS 凭证进行身份验证。它们未经身份验证，或者是使用会话字符串或访问令牌授权的。有关更多信息，请参阅Amazon Cognito 用户群体经过身份验证和未经身份验证的 API 操作 (p. 48)。

您可以使用以下规则操作在 Web ACL 中配置规则：Count（计数）、Allow（允许）、Block（阻止），或者提供一个验证码以响应符合规则的请求。有关更多信息，请参阅 AWS WAF 开发人员指南 中的 AWS WAF 规则。根据规则操作，您可以自定义 Amazon Cognito 返回给用户的响应。

Important

自定义错误响应的选项取决于您发出 API 请求的方式。

- 您可以自定义托管 UI 请求的错误代码和响应正文。您只能在托管 UI 中为用户提供验证码以解决此问题。
- 对于您使用 Amazon Cognito 用户群体 API 提出的请求，可以自定义收到阻止响应的请求的响应正文。您也可以指定 400–499 范围内的自定义错误代码。
- AWS Command Line Interface (AWS CLI) 和 AWS SDK 返回一个 `ForbiddenException` 错误，以请求生成 Block（阻止）或 CAPTCHA（验证码）响应。

将 Web ACL 与您的用户池相关联

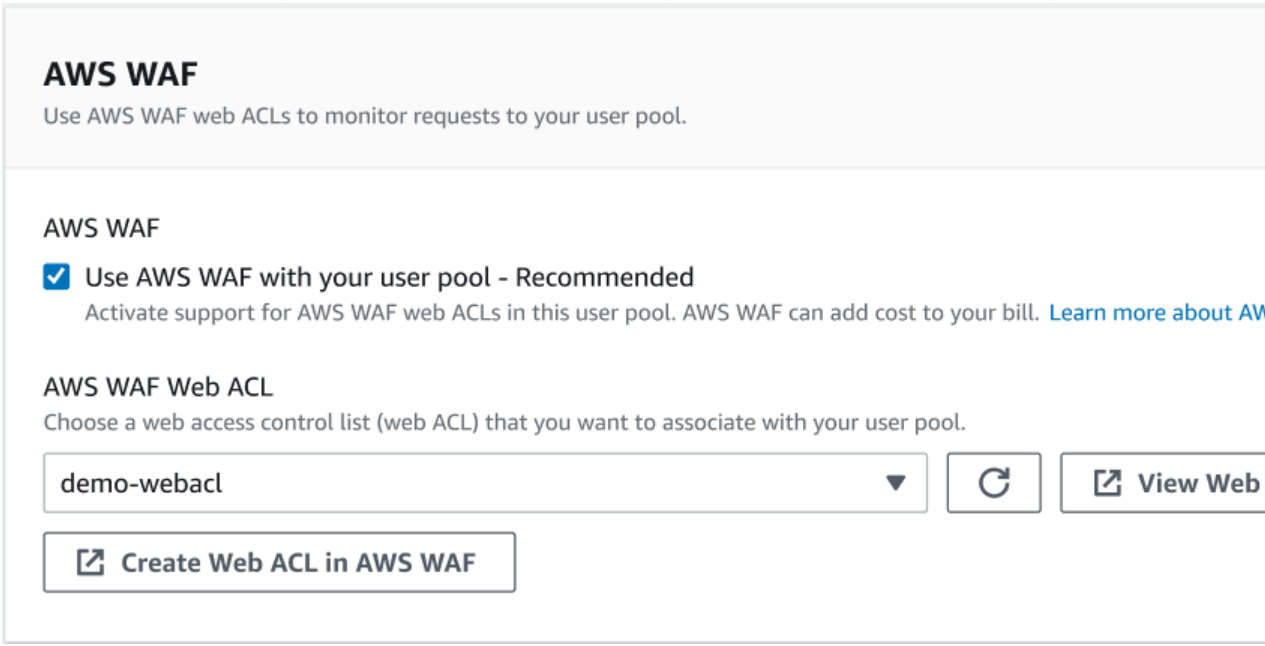
要在您的用户池中使用 Web ACL，AWS Identity and Access Management (IAM) 主体必须拥有以下 Amazon Cognito 权限。有关 AWS WAF 权限的信息，请参阅 AWS WAF 开发人员指南 中的 AWS WAF API 权限。

- `cognito-idp:AssociateWebACL`
- `cognito-idp:DisassociateWebACL`
- `cognito-idp:GetWebACLForResource`
- `cognito-idp:ListResourcesForWebACL`

尽管您必须授予 IAM 权限，但列出的操作仅用于说明权限，不对应于 API 操作。

为您的用户池激活 AWS WAF 并关联一个 Web ACL

1. 登录 Amazon Cognito 控制台。
2. 在导航窗格中，选择 User Pools（用户池），然后选择要编辑的用户池。
3. 选择 User pool properties（用户池属性）选项卡。
4. 选择 AWS WAF 旁边的 Edit（编辑）。
5. 在 AWS WAF 下，选择 Use AWS WAF with your user pool（将 WAF 与用户池结合使用）。



6. 选择一个您已经创建的 AWS WAF Web ACL，或者选择 Create web ACL in AWS WAF（在 WAF 中创建 Web ACL），以在AWS Management Console的新 AWS WAF 会话中创建一个 Web ACL。
7. 选择 Save changes（保存更改）。

要在 AWS Command Line Interface 或 SDK 中以编程方式将 Web ACL 与用户群体相关联，请使用 AWS WAF API 中的 AssociateWebACL。Amazon Cognito 没有可关联 Web ACL 的单独 API 操作。

测试和记录 AWS WAF Web ACL

当您在 Web ACL 中将规则操作设置为 Count（计数）时，AWS WAF 会将请求添加到与规则匹配的请求计数中。要使用您的用户池测试 Web ACL，请将规则操作设置为 Count（计数），并考虑与每条规则匹配的请求数量。例如，如果您要设置为 Block（阻止）操作的规则与您确定为正常用户流量的大量请求相匹配，则您可能需要重新配置规则。有关更多信息，请参阅 AWS WAF 开发人员指南 中的测试和调整您的 AWS WAF 保护。

您还可以配置 AWS WAF 以将请求标头记录到 Amazon CloudWatch Logs 日志组、Amazon Simple Storage Service (Amazon S3) 存储桶或 Amazon Kinesis Data Firehose。您可以通过 `x-amzn-cognito-client-id` 和 `x-amzn-cognito-operation-name` 识别您使用用户群体 API 发出的 Amazon Cognito 请求。托管的 UI 请求仅包含 `x-amzn-cognito-client-id` 标头。有关更多信息，请参阅 AWS WAF 开发人员指南 中的记录 Web ACL 流量。

AWS WAF Web ACL 不受 Amazon Cognito 高级安全功能定价的约束。AWS WAF 的安全功能可以补充 Amazon Cognito 高级安全功能。您可以在用户池中激活这两项功能。AWS WAF 对用户池请求的检查单独收费。有关更多信息，请参阅 AWS WAF 定价。

记录 AWS WAF 请求数据将由您的日志的目标服务收取额外费用。有关更多信息，请参阅 AWS WAF 开发人员指南 中的记录 Web ACL 流量信息的定价。

用户池区分大小写

默认情况下，您在 AWS Management Console 中创建的 Amazon Cognito 用户池不区分大小写。当用户池不区分大小写时，user@example.com 和 User@example.com 指的是同一个用户。当用户群体中的用户名不区分大小写时，`preferred_username` 和 `email` 属性也不区分大小写。

要考虑用户池区分大小写设置，请根据备用用户属性识别应用程序代码中的用户。由于用户名、首选用户名或电子邮件地址属性的大小写在不同的用户配置文件中可能有所不同，因此请改为参阅 sub 属性。您还可以在用户池中创建不可变的自定义属性，并为每个新用户配置文件中的属性分配自己的唯一标识符值。首次创建用户时，可以将值写入您创建的不可改变的自定义属性。

Note

无论您的用户池的区分大小写设置如何，Amazon Cognito 都要求来自 SAML 或 OIDC 身份提供商 (IdP) 的联合身份用户传递唯一且区分大小写的 `NameId` 或 `sub` 声明。有关唯一标识符区分大小写和 SAML IdP 的更多信息，请参阅SAML 用户池 IdP 身份验证流程 (p. 94)。

创建区分大小写的用户池

如果您使用 AWS Command Line Interface (AWS CLI) 和 API 操作（如 CreateUserPool）创建资源，则必须将布尔型 `CaseSensitive` 参数设置为 `false`。此设置将创建不区分大小写的用户池。如果您不指定值，`CaseSensitive` 默认为 `true`。此默认设置与您在 AWS Management Console 中创建的用户池的默认行为相反。2020 年 2 月 12 日之前，无论平台如何，用户池都默认区分大小写。

您可以使用 AWS Management Console 的 Sign-in experience（登录体验）选项卡或 DescribeUserPool API 操作来查看您账户中每个用户池的区分大小写设置。

迁移到新的用户池

由于用户配置文件之间存在潜在冲突，您无法将 Amazon Cognito 用户池从区分大小写更改为不区分大小写。相反，请将用户迁移到新的用户池。您必须构建迁移代码才能解决与大小写相关的冲突。此代码必须返回唯一的新用户，或者在检测到冲突时拒绝登录尝试。在新的不区分大小写的用户池中，分配一个 迁移用户 Lambda 触发器 (p. 147)。AWS Lambda 函数可以在新的不区分大小写的用户池中创建用户。当用户使用不区分大小写的用户池登录失败时，Lambda 函数会从区分大小写的用户池中查找并复制该用户。您还可以在 ForgotPassword 事件上激活迁移用户 Lambda 触发器。Amazon Cognito 会将用户信息和事件元数据从登录或密码恢复操作传递到您的 Lambda 函数。当函数在不区分大小写的用户群体中创建新用户时，您可以使用事件数据来管理用户名和电子邮件地址之间的冲突。这些冲突是用户名和电子邮件地址之间的冲突，这些名称和地址在不区分大小写的用户群体中是唯一的，但在区分大小写的用户群体中则相同。

有关如何在 Amazon Cognito 用户池之间使用迁移用户 Lambda 触发器的更多信息，请参阅 AWS 博客中的将用户迁移到 Amazon Cognito 用户池。

管理用户存在错误响应

Amazon Cognito 支持自定义由用户群体返回的错误响应。自定义错误响应可用于用户创建和身份验证、密码恢复和确认操作。

使用用户池应用程序客户端的 `PreventUserExistenceErrors` 设置，以启用或禁用用户存在相关错误。当您创建新的用户群体时，默认情况下 `PreventUserExistenceErrors` 为 `false`。当此值为 `true` 时，Amazon Cognito 会使用非特定信息响应 API 请求，否则其响应可能会泄露存在有效的用户。在 Amazon Cognito API 中，通过 UpdateUserPoolClient 请求更新此值。在 AWS Management Console 中，创建或编辑您的应用程序客户端以激活 Prevent user existence errors（防止用户已存在错误）。

当您防止在用户群体中出现用户存在错误时，Amazon Cognito 登录和忘记密码 API 将返回一个通用的身份验证故障响应。错误响应告知您用户名或密码不正确。Amazon Cognito 账户确认和密码恢复 API 返回一个响应，指示代码已发送到模拟的传输媒介。当状态为`ENABLED`并且用户不存在时错误响应工作。下面是当 `PreventUserExistenceErrors` 设置为 `ENABLED` 时，Amazon Cognito 操作的详细行为。

用户创建和身份验证操作

您可以将用户名密码身份验证 或安全远程密码（SRP，Secure Remote Password）身份验证 用于以下操作。您还可以对使用自定义身份验证返回的错误进行自定义。

- `AdminInitiateAuth`

- AdminRespondToAuthChallenge
- InitiateAuth
- RespondToAuthChallenge

以下列表演示了如何在用户身份验证操作中自定义错误响应。

用户名和密码身份验证

要使用 ADMIN_USER_PASSWORD_AUTH 和 USER_PASSWORD_AUTH 登录用户，请在 AdminInitiateAuth 或 InitiateAuth API 请求中包含用户名和密码。在用户名或密码不正确时，Amazon Cognito 返回一个通用 NotAuthorizedException 错误。

基于安全远程密码（SRP）的身份验证

要使用 USER_SRP_AUTH 登录用户，请在 AdminInitiateAuth 或 InitiateAuth API 请求中包含用户名和 SRP_A 参数。作为响应，Amazon Cognito 根据 SRP 标准，向用户返回 SRP_B 和“salt”。如果找不到用户，Amazon Cognito 会在第一步中返回一个模拟响应，如 RFC 5054 中所述。Amazon Cognito 始终针对相同的用户名和用户群体组合返回相同的加密盐以及通用唯一标识符（UUID）格式的内部用户 ID。当您发送带有密码证明的 RespondToAuthChallenge API 时，Amazon Cognito 在用户名或密码不正确时返回一个通用 NotAuthorizedException 错误。

Note

如果您使用基于验证的别名属性，并且不可改变的用户名格式不是 UUID，则可以模拟使用用户名和密码身份验证的通用响应。

自定义身份验证质询 Lambda 触发器

如果您使用自定义身份验证质询 Lambda 触发器并启用错误响应，则 LambdaChallenge 将返回一个名为 UserNotFound 的布尔值参数。然后它在 DefineAuthChallenge、VerifyAuthChallenge 和 CreateAuthChallenge Lambda 触发器请求后传递。您可以使用此触发器来模拟不存在用户的自定义授权质询。如果您为不存在的用户调用预身份验证 Lambda 触发器，则 Amazon Cognito 将返回UserNotFound。

以下列表演示了如何在用户创建操作中自定义错误响应。

SignUp

当已使用用户名时，SignUp 操作返回 UsernameExistsException。如果在您的应用程序中注册用户时，您不希望 Amazon Cognito 为电子邮件地址和电话号码返回 UsernameExistsException 错误，请使用基于验证的别名属性。有关别名的更多信息，请参阅用户群体属性的自定义登录属性部分。

有关 Amazon Cognito 如何阻止使用 SignUp API 请求来发现用户群体中用户的示例，请参阅 在注册时防止出现电子邮件地址和电话号码的 UsernameExistsException 错误 (p. 240)。

导入的用户

如果 PreventUserExistenceErrors 已启用，则在对导入的用户进行身份验证期间，将返回通用 NotAuthorizedException 错误，指示用户名或密码不正确，而不是返回 PasswordResetRequiredException。请参阅要求导入的用户重置密码，了解更多信息。

迁移用户 Lambda 触发器

当 Lambda 触发器在原始事件上下文中设置了空响应时，Amazon Cognito 将为不存在的用户返回模拟响应。有关更多信息，请参阅迁移用户 Lambda 触发器。

在注册时防止出现电子邮件地址和电话号码的 UsernameExistsException 错误

以下示例演示了在用户群体中配置别名属性时，如何在对 SignUp API 请求的响应中，防止重复的电子邮件地址和电话号码生成 UsernameExistsException 错误。您必须在创建用户群体时使用电子邮件地址或电话号码作为别名属性。有关更多信息，请参阅用户群体属性的自定义登录属性 部分。

1. Jie 注册了一个新的用户名，还提供了电子邮件地址 `jie@example.com`。Amazon Cognito 将向其电子邮件地址发送一个代码。

   示例 AWS CLI 命令

   ```
   aws cognito-idp sign-up --client-id 1234567890abcdef0 --username jie --password PASSWORD
    --user-attributes Name="email",Value="jie@example.com"
   ```

   响应示例

   ```
   {
       "UserConfirmed": false,
       "UserSub": "<subId>",
       "CodeDeliveryDetails": {
           "AttributeName": "email",
           "Destination": "j****@e****",
           "DeliveryMedium": "EMAIL"
       }
   }
   ```

2. Jie 提供了发送过来的代码，确认其拥有该电子邮件地址。这样就完成了用户注册。

   示例 AWS CLI 命令

   ```
   aws cognito-idp confirm-sign-up --client-id 1234567890abcdef0 --username=jie --
   confirmation-code xxxxxx
   ```

3. Shirley 注册了一个新的用户账户并提供了电子邮件地址 `jie@example.com`。Amazon Cognito 不会返回 `UsernameExistsException` 错误，而是向 Jie 的电子邮件地址发送确认代码。

   示例 AWS CLI 命令

   ```
   aws cognito-idp sign-up --client-id 1234567890abcdef0 --username shirley --password
    PASSWORD --user-attributes Name="email",Value="jie@example.com"
   ```

   响应示例

   ```
   {
       "UserConfirmed": false,
       "UserSub": "<new subId>",
       "CodeDeliveryDetails": {
           "AttributeName": "email",
           "Destination": "j****@e****",
           "DeliveryMedium": "EMAIL"
       }
   }
   ```

4. 在另一种情况下，Shirley 拥有对 `jie@example.com` 的所有权。Shirley 收到了 Amazon Cognito 发送到 Jie 电子邮件地址的代码，并尝试确认该账户。

   示例 AWS CLI 命令

   ```
   aws cognito-idp confirm-sign-up --client-id 1234567890abcdef0 --username=shirley --
   confirmation-code xxxxxx
   ```

   响应示例

```
An error occurred (AliasExistsException) when calling the ConfirmSignUp operation: An
 account with the email already exists.
```

尽管已将 `jie@example.com` 分配给现有用户，Amazon Cognito 不会对 Shirley 的 `aws cognito-idp sign-up` 请求返回错误。在 Amazon Cognito 返回错误响应之前，Shirley 必须证明对该电子邮件地址的所有权。在具有别名属性的用户群体中，此行为会阻止使用公共 SignUp API 来检查是否存在具有给定电子邮件地址或电话号码的用户。

此行为与 Amazon Cognito 向使用现有用户名的 SignUp 请求返回的响应不同，如以下示例所示。尽管 Shirley 从此回复中得知已经存在具有用户名 `jie` 的用户，但他们并不知道与该用户关联的任何电子邮件地址或电话号码。

示例 CLI 命令

```
aws cognito-idp sign-up --client-id 1example23456789 --username jie --password PASSWORD
      --user-attributes Name="email",Value="shirley@example.com"
```

响应示例

```
An error occurred (UsernameExistsException) when calling the SignUp operation: User already
 exists
```

密码重置操作

当您防止出现用户存在错误时，Amazon Cognito 会对用户密码重置操作返回以下响应。

ForgotPassword

当找不到用户、用户已停用或没有经过验证的传送机制来恢复其密码时，Amazon Cognito 会为用户返回 `CodeDeliveryDetails` 以及模拟的传递媒介。模拟的传递媒介由用户池的输入用户名格式和验证设置决定。

ConfirmForgotPassword

Amazon Cognito 为不存在或已禁用的用户返回 `CodeMismatchException` 错误。如果在使用 `ForgotPassword` 时不请求代码，Amazon Cognito 将返回 `ExpiredCodeException` 错误。

确认操作

当您防止出现用户存在错误时，Amazon Cognito 会对用户确认和验证操作返回以下响应。

ResendConfirmationCode

Amazon Cognito 为已禁用或不存在的用户返回 `CodeDeliveryDetails`。Amazon Cognito 会向现有用户的电子邮件或电话发送确认码。

ConfirmSignUp

如果代码已过期，则将返回 `ExpiredCodeException`。当用户未被授权时，Amazon Cognito 返回 `NotAuthorizedException`。如果代码与服务器期望的代码不匹配，则 Amazon Cognito 返回 `CodeMismatchException`。

用户池参考（AWS Management Console）

您可以根据应用程序的需要自定义用户池设置。本节介绍设置的每个类别，并为您提供有关属性、策略、电子邮件和电话验证、多重验证、应用程序、触发器和信任设备的详细信息。

主题

- 导入和创建用户与组 (p. 243)
- 用户池属性 (p. 243)
- 添加用户池密码要求 (p. 250)
- 配置注册和管理性创建用户 (p. 250)
- 用户池删除保护 (p. 251)
- 配置电子邮件或电话验证 (p. 251)
- 配置 SMS 和电子邮件验证消息以及用户邀请消息 (p. 253)
- 将成本分配标签添加到您的用户池 (p. 257)
- 指定用户池设备跟踪设置 (p. 258)
- 配置用户池应用程序客户端 (p. 258)
- 配置用户池 Lambda 触发器 (p. 260)
- 配置用户池分析 (p. 260)
- 为用户池添加域名 (p. 261)
- 自定义内置应用程序 UI 以便注册用户和让用户登录 (p. 262)
- 为用户池配置身份提供商 (p. 262)
- 为用户池配置属性映射 (p. 266)
- Amazon Cognito 用户群体旧版控制台过程 (p. 267)

导入和创建用户与组

在用户和组选项卡上，您可以导入用户、创建用户、搜索用户以及创建组并将用户分配到其中。有关更多信息，请参阅：

- 通过 CSV 文件将用户导入用户池中 (p. 184)
- 向用户池添加组 (p. 178)
- 以管理员身份创建用户账户 (p. 175)

用户池属性

属性是各种条目的信息，用于帮助您标识单个用户，如名称、电子邮件和电话号码。新的用户池有一组默认标准属性。您也可以在 AWS Management Console 中为您的用户池定义添加自定义属性。本主题将详细介绍这些属性，并为您提供有关如何设置用户池的提示。

请勿将所有与用户相关的信息都应存储在属性中。例如，将频繁变化的用户数据（如使用情况统计数据或游戏分数）保存在单独的数据存储（如 Amazon Cognito Sync 或 Amazon DynamoDB）中。

主题

- 标准属性 (p. 244)
- 用户名和首选用户名 (p. 245)
- 自定义登录属性 (p. 246)
- 自定义属性 (p. 248)
- 属性权限和范围 (p. 249)

标准属性

Amazon Cognito 根据 OpenID Connect 规范为所有用户分配了以下组标准属性。

- address
- birthdate
- email
- family_name
- gender
- given_name
- locale
- middle_name
- name
- nickname
- phone_number
- picture
- preferred_username
- profile
- sub
- updated_at
- website
- zoneinfo

除 sub 外，原定设置情况下，对于所有用户，标准属性都是可选的。要将某个属性设置为必需属性，请在用户池创建过程中，选择属性旁边的 Required（必需）复选框。Amazon Cognito 为每个用户的 sub 属性分配一个唯一的用户标识符值。

> **Note**
>
> 如果您将某个标准属性标记为 Required（必需），则用户必须为该属性提供一个值才能注册。要创建用户而不是为所需属性提供值，管理员可以使用 AdminCreateUser API。创建用户池后，您无法在必需属性和非必需属性之间切换属性。

默认情况下，标准和自定义属性值可以是任何不超过 2048 个字符的字符串，但是某些属性值（例如 updated_at）有格式限制。只能验证 email（电子邮件）和 phone（电话）。

> **Note**
>
> 一些文档和标准将属性称为成员。

标准属性详细信息

标准属性详细信息

email

用户和管理员可以验证电子邮件地址值。

具有适当 AWS 账户 权限的管理员可以更改用户的电子邮件，并将其标记为已验证。使用 AdminUpdateUserAttributes API 或 admin-update-user-attributes AWS Command Line Interface (AWS CLI) 命令将电子邮件地址标记为已验证。使用此命令，管理员可以将 email_verified 属性更改为 true。您也可以在 AWS Management Console 的 Users（用户）选项卡中编辑用户，以将电子邮件地址标记为已验证。

phone_number

如果 SMS 多重验证 (MFA) 处于活动状态，用户必须提供电话号码。有关更多信息，请参阅向用户池添加 MFA (p. 220)。

用户和管理员可以验证电话号码值。

具有适当 AWS 账户 权限的管理员可以更改用户的电话号码，并将其标记为已验证。使用 AdminUpdateUserAttributes API 或 admin-update-user-attributes (AWS CLI) 命令将电话号码标记为已验证。使用此命令，管理员可以将 `phone_number_verified` 属性更改为 `true`。您也可以在 AWS Management Console的 Users（用户）选项卡中编辑用户，以将电话号码标记为已验证。

Important

电话号码必须遵循以下格式规则：电话号码必须以加号 (**+**) 开头，后面紧跟国家/地区代码。电话号码只能包含 **+** 号和数字。先删除电话号码中的任何其他字符，如圆括号、空格或短划线 (**-**)，然后再将该值提交给服务。例如，美国境内的电话号码必须遵循以下格式：**+14325551212**。

preferred_username

您可以将 `preferred_username` 选择为必需或别名，但不能同时选择这两者。如果 `preferred_username` 是一个别名，您可以使用 UpdateUserAttributes API 在确认用户后添加属性值。

sub

根据 `sub` 属性对用户编制索引和进行搜索。`sub` 属性是每个用户群体中的唯一用户标识符。用户可以更改如 `username` 和 `email` 等属性。`sub` 属性具有固定的值。有关查找用户的更多信息，请参阅管理和搜索用户账户 (p. 180)。

查看必需属性

通过以下过程可查看给定用户池的必需属性。

Note

在创建用户池后，您无法更改必需属性。

查看必需属性

1. 转到 AWS Management Console中的 Amazon Cognito。如果控制台提示您，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 Sign-up experience（注册体验）选项卡。
5. 在 Required attributes（必需属性）部分中，查看用户池的必需属性。

用户名和首选用户名

`username` 值是一个单独的属性，与 `name` 属性不同。每个用户都有 `username` 属性。Amazon Cognito 会自动为联合用户生成用户名。您必须提供 `username` 属性以在 Amazon Cognito 目录中创建本地用户。创建用户后，您将无法更改 `username` 属性的值。

开发人员可以使用 `preferred_username` 属性为用户提供一个他们可以更改的用户名。有关更多信息，请参阅自定义登录属性 (p. 246)。

如果您的应用程序不需要用户名，就不必要求用户提供用户名。您的应用程序可以在后台为用户创建唯一的用户名。如果您希望用户使用电子邮件地址和密码注册和登录，这非常有用。有关更多信息，请参阅自定义登录属性 (p. 246)。

在用户池中，username 必须是唯一的。username 可重复使用，但只能是在您已将其删除且不再使用它的情况下。

自定义登录属性

创建用户群体时，如果您希望用户能够使用电子邮件地址或电话号码作为其用户名进行注册和登录，则可以设置用户名属性。或者，您可以设置别名属性为用户提供选项：用户可以在注册时包含多个属性，然后使用用户名、首选用户名、电子邮件地址或电话号码登录。

> **Important**
>
> 创建用户池后，您无法更改此设置。

如何在别名属性和用户名属性之间进行选择

您的要求	别名属性	用户名属性
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
Prevent `UsernameExistsException` when user signs up	Yes	No
Can assign the same sign-in attribute to more than one user	Yes[3]	No

[1] 可用的登录属性包括用户名、电子邮件地址、电话号码和首选用户名。

[2] 可以使用电子邮件地址或电话号码进行登录。

[3] 只有最后验证了该属性的用户才能使用该属性登录。

选项 1：多个登录属性（别名属性）

如果您想允许用户在登录时选择输入其用户名或其他属性值，则可以激活别名。默认情况下，用户使用他们的用户名和密码登录。用户名是一个固定值，用户无法更改。如果您将某个属性标记为别名，用户就可以使用该属性代替用户名来登录。您可以将电子邮件地址、电话号码和首选用户名属性标记为别名。例如，如果您选择电子邮件地址和电话号码作为用户群体的别名，该用户群体中的用户就可以将用户名、电子邮件地址或电话号码与密码一起使用进行登录。

要选择别名属性，请在创建用户群体时选择 User Name（用户名）和至少一个其他登录选项。

> **Note**
>
> 将用户池配置为不区分大小写时，用户可以使用小写或大写字母进行注册或使用别名登录。有关更多信息，请参阅《Amazon Cognito 用户池 API 参考》中的 CreateUserPool。

如果您选择电子邮件地址作为别名，Amazon Cognito 不接受与有效电子邮件地址格式匹配的用户名。同样，如果您选择电话号码作为别名，Amazon Cognito 将不接受与有效的电话号码格式相匹配的用户群体的用户名。

> **Note**
>
> 在用户池中，别名值必须是唯一的。如果您为电子邮件地址或电话号码配置别名，那么提供的值只能在一个账户中处于已验证状态。在注册期间，如果您的用户提供电子邮件地址或电话号码作

为别名值，而另一用户已使用该别名值，注册将成功。然而，当用户尝试使用此电子邮件（或电话号码）确认账户并输入有效的代码时，Amazon Cognito 会返回 AliasExistsException 错误。该错误向用户指出，已存在使用此电子邮件地址（或电话号码）的账户。此时，用户可以放弃新账户的创建，并尝试重置旧账户的密码。如果用户继续创建新账户，您的应用程序必须使用 forceAliasCreation 选项调用 ConfirmSignUp API。ConfirmSignUp 和 forceAliasCreation 结合会将别名从以前的账户移至新创建的账户，并在以前的账户中将此属性标记为未经验证。

只有在您的用户验证电话号码和电子邮件地址后，电话号码和电子邮件地址才会成为用户的活动别名。如果您将电子邮件地址和电话号码用作别名，我们建议您选择对其进行自动验证。

选择别名属性以防止用户注册时出现电子邮件地址和电话号码属性的 UsernameExistsException 错误。

激活 preferred_username 属性，以便您的用户可以更改他们用来登录的用户名，而他们的 username 属性值不会更改。如果您想设置这种用户体验，请提交新的 username 值作为 preferred_username，并选择 preferred_username 作为别名。这样，用户就可以使用输入的新值登录。如果选择 preferred_username 作为别名，您的用户只有在确认账户时才能提供该值。他们在注册期间无法提供该值。

当用户使用用户名注册时，您可以选择他们是否可以使用以下一个或多个别名登录。

- 经过验证的电子邮件地址
- 经过验证的电话号码
- 首选用户名

用户注册后可以更改这些别名。

Important

当您的用户群体支持使用别名登录，并且您想要向用户授权或查找用户时，请不要通过用户的任何登录属性来识别您的用户。固定值的用户标识符 sub 是用户身份的唯一一致指标。

在创建用户池时包括以下步骤，以便用户可以使用别名登录。

配置用户群体以便用户可以使用首选用户名登录

1. 转到 AWS Management Console中的 Amazon Cognito。如果控制台提示您，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 在页面右上角，选择 Create a user pool（创建用户池）以开启用户池创建向导。
4. 在 Configure sign-in experience（配置登录体验）中，选择您想将其与用户池关联的身份 Provider types（提供商类型）。
5. 在 Cognito user pool sign-in options（Cognito 用户池登录选项）下，选择 User name（用户名）、Email（电子邮件）和 Phone number（电话号码）的任意组合。
6. 在用户名要求下，选择允许用户使用首选用户名登录，以便用户可以设置登录时要使用的备用用户名。
7. 选择 Next（下一步），然后完成向导中的所有步骤。

选项 2：将电子邮件地址或电话号码作为登录属性（用户名属性）

当用户使用电子邮件地址或电话号码作为其用户名进行注册时，您可以选择他们是否可以仅使用电子邮件地址、仅使用电话号码或其中之一进行注册。

要选择用户名属性，请在创建用户群体时不要选择用户名作为登录选项。

电子邮件地址或电话号码必须是唯一的，并且不能已被其他用户使用。它不必经过验证。用户使用电子邮件地址或电话号码注册之后，将无法使用相同的电子邮件地址或电话号码创建新账户。如果需要，用户只能重复使用现有账户并重置账户的密码。但是，用户可以将电子邮件地址或电话号码更改为新的电子邮件地址或电话号码。如果电子邮件地址或电话号码未被使用，它将成为新的用户名。

Note

如果用户使用电子邮件地址作为用户名进行注册，则可以将用户名更改为另一个电子邮件地址，但他们无法将用户名更改为电话号码。如果用户使用电话号码进行注册，他们可以将用户名更改为另一个电话号码，但他们无法将用户名更改为电子邮件地址。

在用户池创建过程中使用以下步骤设置使用电子邮件地址或电话号码注册和登录。

配置用户池使用电子邮件地址或电话号码注册和登录

1. 转到 AWS Management Console中的 Amazon Cognito。如果控制台提示您，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 在页面右上角，选择 Create a user pool（创建用户池）以开启用户池创建向导。
4. 在 Cognito user pool sign-in options（Cognito 用户群体登录选项）下，选择 Email（电子邮件）和 Phone number（电话号码）的任意组合，这表示用户可用于登录的属性。
5. 选择 Next（下一步），然后完成向导中的剩余步骤。

Note

您不需要将电子邮件地址或电话号码标记为用户池的必需属性。

在应用程序中实施选项 2

1. 调用 `CreateUserPool` API，以创建用户池。将 `UserNameAttributes` 参数设置为 `phone_number`、`email` 或 `phone_number | email`。
2. 调用 `SignUp` API 并在 API 的 `username` 参数中传递电子邮件地址或电话号码。此 API 可执行以下操作：
   - 如果 `username` 字符串采用有效的电子邮件格式，用户池将使用 `username` 值自动填充用户的 `email` 属性。
   - 如果 `username` 字符串采用有效的电话号码格式，用户池将使用 `phone_number` 值自动填充用户的 `username` 属性。
   - 如果 `username` 字符串格式不是电子邮件或电话号码格式，`SignUp` API 将返回异常。
   - `SignUp` API 为用户生成一个持久性 UUID，并在内部将其作为不可变的用户名属性。此 UUID 与用户身份令牌中的 `sub` 声明具有相同的值。
   - 如果 `username` 字符串包含已被使用的电子邮件地址或电话号码，`SignUp` API 将返回异常。

您可以使用电子邮件地址或电话号码作为别名，在除 `ListUsers` API 之外的所有 API 中代替用户名。调用 `ListUsers` 时，可以通过 `email` 或 `phone_number` 属性搜索。如果通过 `username` 进行搜索，您必须提供实际的用户名，而不是别名。

自定义属性

您可以将最多 50 个自定义属性添加到您的用户池。您可以为自定义属性指定一个最小和/或最大长度。但是，任何自定义属性的最大长度不能超过 2048 个字符。

每个自定义属性都具有以下特性：

- 可以将其定义为一个字符串或数字。Amazon Cognito 仅将自定义属性值作为字符串写入 ID 令牌。
- 您不能要求用户为属性提供值。
- 将其添加到用户池后，您将无法删除或更改它。
- 属性名称的字符长度在 Amazon Cognito 接受的限制范围内。有关更多信息，请参阅Amazon Cognito 中的限额 (p. 539)。

- 它可能是可以改变的，也可能是不可改变的。在创建用户时，您只能将值写入不可改变属性。如果您的应用程序客户端具有该属性的写入权限，您可以更改可变属性的值。参阅 属性权限和范围 (p. 249) 了解更多信息。

  Note

  在您的代码和 使用基于角色的访问控制 (p. 318) 的规则设置中，自定义属性需要使用 `custom:` 前缀，以便将它们与标准属性区分开来。

通过以下过程创建新的自定义属性。

使用控制台添加自定义属性

1. 转到 AWS Management Console中的 Amazon Cognito。如果控制台提示您，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 Sign-up experience（注册体验）选项卡，然后在 Custom attributes（自定义属性）部分，选择 Add custom attributes（添加自定义属性）。
5. 在存储库的 Add custom attributes（添加自定义属性）页面上，提供有关新属性的以下详细信息：
   - 输入 Name（名称）。
   - 选择 Type（类型）（字符串或数字）。
   - 输入最小字符串长度或数字值。
   - 输入最大字符串长度或数字值。
   - 如果您想授予用户在设置初始值后更改自定义属性值的权限，请选择 Mutable（可变）。
6. 选择 Save changes（保存更改）。

属性权限和范围

对于每个应用程序客户端，您可以为每个用户属性设置读取和写入权限。这样，您可以控制为了读取和修改您为用户存储的每个属性，任何应用程序所具有的访问权限。例如，您可以设置一个自定义属性，用于指明用户是否为付费客户。您的应用程序可能能够看到此属性，但无法直接更改它。相反，您可以使用管理工具或后台进程更新此属性。您可以通过 Amazon Cognito 控制台、Amazon Cognito API 或 AWS CLI 设置用户属性的权限。默认情况下，任何新的自定义属性都不可用，直到您为其设置读取和写入权限。在 Amazon Cognito 控制台中创建新的应用程序客户端时，默认情况下，您将授予应用程序对所有标准和自定义属性的读写权限。要限制您的应用程序，只允许其访问其所需信息量，请选择应用程序需要的范围子集。

通过控制台更新属性权限

1. 转到 AWS Management Console中的 Amazon Cognito。如果控制台提示您，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 App integration（应用程序集成）选项卡，然后在 App clients（应用程序客户端）部分，从列表中选择应用程序客户端。
5. 在 Attribute read and write permissions（属性读取和写入权限）部分，选择 Edit（编辑）。
6. 在 Edit attribute read and write permissions（编辑设置属性读取和写入权限）页面上，配置读取和写入权限，然后选择 Save changes（保存更改）。

使用自定义属性对每个应用程序客户端重复这些步骤。

可以针对每个应用程序将属性标记为可读或可写。这对于标准属性和自定义属性均适用。应用程序可以读取您标记为可读的属性，也可以写入您标记为可写的属性。如果应用程序尝试更新不可写的属性，就会收到 `NotAuthorizedException` 异常。当应用程序调用 GetUser 时，只能收到该应用程序可读的属性。来自

应用程序的用户 ID 令牌仅包含与可读属性相对应的声明。用户池中的必需属性始终为可写属性。如果您使用 AWS CLI 或管理 API 来设置可写属性但不提供必需属性，就会收到 `InvalidParameterException` 异常。

您可以在创建用户池后更改属性权限和范围。

添加用户池密码要求

要为您的应用程序用户创建强密码，请将最小密码长度指定为至少八个字符。还需要大写、数字和特殊字符。复杂的密码难于猜测。我们建议您将其作为最佳安全实践。

密码中可使用以下字符：

- 大写和小写基本拉丁语字母
- 数字
- 在下一部分中列出的特殊字符。
- 非前导、非结尾的空格字符。

创建密码策略

您可以在 AWS Management Console 中的 Sign-in experience（登录体验）选项卡下的 Password policy（密码策略）中指定以下要求。

- 密码最小长度至少为 6 个字符，最多 99 个字符。这是您希望用户能够设置的最短密码。Amazon Cognito 密码最多可包括 256 个字符。
- 您的用户必须创建包含至少 1 种以下类型字符的密码。
  - 数字
  - Special character（特殊字符），源自以下字符集。非前导、非结尾的空格字符也被视为特殊字符。

    ```
    ^ $ * . [ ] { } ( ) ? " ! @ # % & / \ , > < ' : ; | _ ~ ` = + -
    ```

  - Uppercase Basic Latin letter（大写基本拉丁字母）
  - Lowercase Basic Latin letter（小写基本拉丁字母）

配置注册和管理性创建用户

您可以在用户群体中配置管理性创建用户策略。此配置决定用户是否可以注册，或者您是否希望创建这些用户。它还设置了用户无法再确认其用户账户之前的时间。

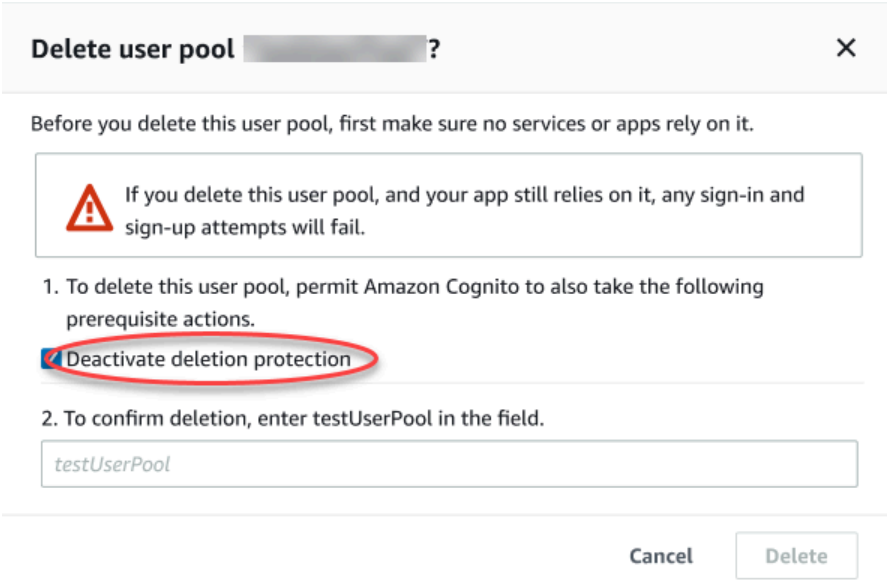
- 如果您希望用户能够使用未经身份验证的 `SignUp` API 生成新账户，请在注册体验选项卡的自助注册下，将自行注册设置为已启用。如果您禁用自行注册，则只能使用您通过具有 `AdminCreateUser` 权限的 IAM 凭证签署的 `AdminCreateUser` API 请求创建新用户。在禁用自行注册的用户群体中，注册 API 请求返回 `NotAuthorizedException`。
- 在登录体验选项卡的密码策略下，为管理员设置的临时密码过期时间设置一个值。在此期限过后，您通过 `AdminCreateUser` API 请求创建的新用户将无法登录和设置新密码。从管理员或用户创建账户时开始计算，原定设置为 7 天。最大设置为 365 天。要更新 Amazon Cognito 用户群体 API 中的密码持续时间，请在您的 CreateUserPool 或 UpdateUserPool API 请求中为 UnusedAccountValidityDays 设置一个值。

  要重置过期用户账户的访问权限，请更新自动验证的属性或发送 `AdminResetUserPassword` API 请求。

  > **Note**
  >
  > 用户登录后，账户将永不过期。

用户池删除保护

要使您的管理员不会意外删除用户池，请激活删除保护。启用删除保护后，必须先确认要删除用户池，然后才能将其删除。在 AWS Management Console 中删除用户池时，可以同时停用删除保护。当您接受停用删除保护的提示并确认要删除时（如下图所示），Amazon Cognito 会删除您的用户池。



如果您想通过 Amazon Cognito API 请求删除用户池，必须先在 UpdateUserPool 请求中将 `DeletionProtection` 更改为 `Inactive`。如果您不停用删除保护，Amazon Cognito 会返回 `InvalidParameterException` 错误。停用删除保护后，您可以在 deleteUserPool 请求中删除用户池。

当您在 AWS Management Console 中创建新的用户群体时，Amazon Cognito 默认情况下会激活 Deletion protection（删除保护）。当您使用 `CreateUserPool` API 创建用户群体时，默认情况下未激活删除保护。要在使用 AWS CLI 或 AWS SDK 创建的用户群体中使用此功能，请将 `DeletionProtection` 参数设置为 `True`。

您可以在 Amazon Cognito 控制台的 User pool settings（用户群体设置）选项卡上的 Deletion protection（删除保护）容器中激活或停用删除保护状态。

配置删除保护

1. 转到 Amazon Cognito 控制台。系统可能会提示您输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 User pool settings（用户池设置）选项卡。找到 Deletion Protection（删除保护），然后选择 Activate（激活）或 Deactivate（停用）。
5. 在下一个对话框中确认您的选择。

配置电子邮件或电话验证

您可以在消息收发选项卡下选择电子邮件或电话验证的设置。有关多重验证 (MFA) 的详细信息，请参阅SMS 文本消息 MFA (p. 222)。

Amazon Cognito 使用 Amazon SNS 发送 SMS 消息。如果您之前没有从 Amazon Cognito 或任何其他 AWS 服务发送 SMS 消息，Amazon SNS 可能会将您的账户放在 SMS 沙盒中。我们建议您在将账户从沙盒移到生产环境之前，向已验证的电话号码发送测试消息。此外，如果您计划向美国的目标电话号码发送短信，则

必须从 Amazon Pinpoint 获取源 ID 或发件人 ID。要为 Amazon Cognito 用户群体配置 SMS 消息，请参阅 Amazon Cognito 用户池的短信设置 (p. 199)。

Amazon Cognito 可以自动验证电子邮件地址或电话号码。要进行此验证，Amazon Cognito 将发送验证码或验证链接。对于电子邮件地址，Amazon Cognito 可以通过电子邮件发送代码或链接。当您在 Amazon Cognito 控制台的消息收发选项卡中编辑验证消息模板时，您可以选择验证类型，即代码或链接。有关更多信息，请参阅自定义电子邮件验证消息 (p. 255)。

对于电话号码，Amazon Cognito 以 SMS 文本消息的形式发送代码。

Amazon Cognito 必须验证电话号码或电子邮件地址来确认用户，帮助他们恢复忘记的密码。或者，您也可以通过预注册 Lambda 触发器自动确认用户，或者使用 AdminConfirmSignUp API 操作。有关更多信息，请参阅注册并确认用户账户 (p. 168)。

验证代码或链接的有效期为 24 小时。

如果您选择要求通过电子邮件地址或电话号码进行验证，则在用户注册时，Amazon Cognito 将自动发送验证代码或链接。如果用户池已配置了 自定义 SMS 发件人 Lambda 触发器 (p. 160) 或 自定义电子邮件发件人 Lambda 触发器 (p. 156)，则会改为调用该函数。

注意

- Amazon SNS 会另外收取用于验证电话号码的 SMS 文本消息费用。发送电子邮件不收费。有关 Amazon SNS 定价的信息，请参阅 Worldwide SMS 定价。有关提供 SMS 消息收发服务的最新国家/地区列表，请参阅支持的区域和国家/地区。
- 当您在应用程序中测试从 Amazon Cognito 生成电子邮件的操作时，请使用 Amazon Cognito 可以发送到而不会查无此人的邮件的真实电子邮件地址。有关更多信息，请参阅the section called “在测试应用程序时发送电子邮件” (p. 174)。
- 忘记密码流程要求通过用户的电子邮件或电话号码来验证用户。

Important

如果用户同时注册了电话号码和电子邮件地址，并且用户群体设置需要验证这两个属性，那么 Amazon Cognito 会通过 SMS 消息将验证码发送到电话号码上。Amazon Cognito 未验证电子邮件地址，因此您的应用程序必须调用 GetUser 以查看电子邮件地址是否在等待验证。如果确实需要验证，应用程序必须调用 GetUserAttributeVerificationCode 以启动电子邮件验证流程。然后，它必须通过调用 VerifyUserAttribute 来提交验证码。

您可以调整 AWS 账户和各条消息的 SMS 消息支出限额。该限额仅适用于发送 SMS 消息的费用。有关更多信息，请参阅 Amazon SNS 常见问题中的什么是账户级别和消息级别支出配额以及它们如何运作？

Amazon Cognito 使用 Amazon SNS 资源发送 SMS 消息，这些资源位于您创建用户池的 AWS 区域，或者位于下表中的旧版 Amazon SNS 备用区域。亚太地区（首尔）区域中的 Amazon Cognito 用户池例外。这些用户池使用您在亚太地区（东京）区域中的 Amazon SNS 配置。有关更多信息，请参阅为 Amazon SNS SMS 消息选择 AWS 区域 (p. 200)。

Amazon Cognito 区域	旧版 Amazon SNS 备用区域
美国东部（俄亥俄州）	美国东部（弗吉尼亚州北部）
亚太地区（孟买）	亚太地区（新加坡）
亚太地区（首尔）	亚太地区（东京）
加拿大（中部）	美国东部（弗吉尼亚州北部）
欧洲（法兰克福）	欧洲（爱尔兰）
欧洲（伦敦）	欧洲（爱尔兰）

示例：如果您的 Amazon Cognito 用户池位于亚太地区（孟买）区域，并且您增加了在 ap-southeast-1 区域中的支出限额，则可能不希望请求单独增加 ap-south-1 的支出限额。而是可以使用亚太地区（新加坡）中的 Amazon SNS 资源。

验证对于电子邮件地址和电话号码的更新

在用户更改电子邮件地址或电话号码属性的值后，这些属性可以立即变为活动但未经验证的状态。Amazon Cognito 还可以要求您的用户在 Amazon Cognito 更新属性之前验证新值。当您要求用户首先验证新值时，他们可以使用原始值进行登录和接收消息，直到他们验证新值。

当您的用户可以使用其电子邮件地址或电话号码作为用户群体中的登录别名时，他们的已更新属性的登录名取决于您是否要求验证已更新的属性。当您要求用户验证已更新的属性时，用户可以使用原始属性值登录，直到他们验证新值。如果您不要求用户验证已更新的属性，则在验证新值之前，用户无法使用新属性值或原始属性值登录或接收消息。

例如，您的用户群体允许使用电子邮件地址别名登录，并要求用户在更新时验证其电子邮件地址。Sue 以 `sue@example.com` 身份登录，想将她的电子邮件地址更改为 `sue2@example.com`，但是不小心输入了 **s**`sue2@example.com`。Sue 没有收到验证电子邮件，所以她无法验证 `ssue2@example.com`。Sue 以 `sue@example.com` 身份登录，然后在您的应用程序中重新提交表单，以将她的电子邮件地址更新为 `sue2@example.com`。她收到此电子邮件，向您的应用程序提供验证码，然后开始以 `sue2@example.com` 身份登录。

当用户更新其电子邮件地址或电话号码时，需要进行属性验证

1. 登录 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 在导航窗格中，选择 User Pools（用户池），然后选择要编辑的用户池。
3. 在 Sign-up experience（注册体验）选项卡中，在 Attribute verification and user account confirmation（属性验证和用户账户确认）下，选择 Edit（编辑）。
4. 选择 Keep original attribute value active when an update is pending（等待更新时，保持原始属性值处于活动状态）。
5. 在 Active attribute values when an update is pending（等待更新时的活动属性值）下，选择您希望在 Amazon Cognito 更新值之前要求用户验证的属性。
6. 选择 Save changes（保存更改）。

如果要求使用 Amazon Cognito API 进行属性更新验证，您可以在 UpdateUserPool 请求中设置 `RequireAttributesVerifiedBeforeUpdate` 参数。

授权 Amazon Cognito 代表您发送消息。

要代表您向您的用户发送短信，Amazon Cognito 需要具有您的权限。要授予该权限，您可以创建 AWS Identity and Access Management (IAM) 角色。在 Amazon Cognito 控制台的消息收发选项卡的“SMS”下，选择编辑以设置角色。

配置 SMS 和电子邮件验证消息以及用户邀请消息

在消息收发选项卡的消息模板下，您可以自定义：

- 您的 SMS 短信多重身份验证（MFA）消息
- 您的 SMS 和电子邮件验证消息
- 电子邮件的验证类型–代码或链接
- 您的用户邀请消息
- 流经用户池的电子邮件的 FROM 和 REPLY-TO 电子邮件地址

Note

只有在验证选项卡上选择要求进行电话号码和电子邮件验证之后，才会显示 SMS 和电子邮件验证消息模板。同样，只有 MFA 设置为 required（必填）或 optional（可选）时，才会显示 SMS MFA 消息模板。

主题

- 消息模板 (p. 254)
- 自定义 SMS 消息 (p. 255)
- 自定义电子邮件验证消息 (p. 255)
- 自定义用户邀请消息 (p. 256)
- 自定义您的电子邮件地址 (p. 256)
- 授权 Amazon Cognito 代表您发送 Amazon SES 电子邮件（通过自定义 FROM 电子邮件地址）(p. 256)

消息模板

您可以使用消息模板，利用占位符（将被相应值替换）在消息中插入一个字段。

模板占位符

描述	Token
验证代码	{####}
临时密码	{####}
用户名称	{username}

Note

您不能在验证电子邮件中使用 {username} 占位符。您可以在邀请电子邮件中使用您通过 AdminCreateUser 操作生成的 {username} 占位符。这些邀请电子邮件使用两个占位符：{username} 是用户名，{####} 是临时密码。

您可以使用高级安全模板占位符执行以下操作：

- 包括某个事件的特定详细信息，例如 IP 地址、城市、国家/地区、登录时间、设备名称。Amazon Cognito 高级安全功能可以分析这些详细信息。
- 验证一键式链接是否有效。
- 使用事件 ID、反馈令牌和用户名构建您自己的一键式链接。

Note

要生成一键式链接并在高级安全电子邮件模板中使用了 {one-click-link-valid} 和 {one-click-link-invalid} 占位符，您必须已经为用户池配置了域。

高级安全模板占位符

描述	Token
IP 地址	{ip-address}

描述	Token
城市	{city}
国家/地区	{country}
登录时间	{login-time}
设备名称	{device-name}
一键式链接有效	{one-click-link-valid}
一键式链接无效	{one-click-link-invalid}
事件 ID	{event-id}
反馈令牌	{feedback-token}

自定义 SMS 消息

Note

在新的 Amazon Cognito 控制台体验中，您可以在 Message templates（消息模板）标题下的 Messaging（消息收发）选项卡中自定义 SMS 信息。

您可以通过编辑 Do you want to customize your SMS messages?（是否要自定义您的 SMS 消息？）标题下的模板，来自定义用于多重身份验证（MFA）的 SMS 消息。

Important

您的自定义消息必须包含 {####} 占位符。该占位符会在消息发送之前替换为身份验证代码。

消息的最大长度为 140 个 UTF-8 字符，其中包括身份验证代码。

自定义 SMS 验证消息

您可以通过编辑是否要自定义您的 SMS 验证消息？标题下的模板来自定义用于电话号码验证的 SMS 消息。

Important

您的自定义消息必须包含 {####} 占位符。该占位符会在消息发送之前替换为验证代码。

消息的最大长度为 140 个 UTF-8 字符，其中包括验证代码。

自定义电子邮件验证消息

要使用 Amazon Cognito 验证用户池中用户的电子邮件地址，您可以向用户发送一封电子邮件，其中包含用户可以点击的链接或可以输入的代码。

要自定义用于电子邮件地址验证消息的电子邮件主题和消息内容，请编辑用户群体的消息收发选项卡中的验证消息模板。当您编辑验证消息模板时，您可以选择验证类型，即代码或链接。

当您选择代码作为验证类型时，您的自定义消息必须包含 {####} 占位符。发送消息时，验证代码会替换占位符。

当您选择链接作为验证类型时，您的自定义消息必须包含格式为 {##Verify Your Email##} 的占位符。您可以更改占位符之间的文本字符串，例如 {##Click here##}。标题为 Verify Your Email（验证您的电子邮件）的验证链接将替换此占位符。

电子邮件验证消息的链接将您的用户定向到类似于以下示例的 URL。

```
https://<your user pool domain>/confirmUser/?
client_id=abcdefg12345678&user_name=emailtest&confirmation_code=123456
```

消息的最大长度为 20000 个 UTF-8 字符，其中包括验证代码（如果有）。您可以在此消息中使用 HTML 标签来格式化内容。

自定义用户邀请消息

您可以通过编辑消息收发选项卡中的邀请消息模板，自定义 Amazon Cognito 通过 SMS 或电子邮件发送给新用户的用户邀请消息。

> **Important**
>
> 您的自定义消息必须包含 `{username}` 和 `{####}` 占位符。当 Amazon Cognito 发送邀请消息时，它会将这些占位符替换为您用户的用户名和密码。

SMS 消息的最大长度为 140 个 UTF-8 字符，其中包括验证代码。电子邮件的最大长度为 20000 个 UTF-8 字符，其中包括验证代码。您可以在电子邮件中使用 HTML 标签来格式化内容。

自定义您的电子邮件地址

默认情况下，Amazon Cognito 通过 no-reply@verificationemail.com 向用户池中的用户发送电子邮件。您可以选择指定自定义 FROM 和 REPLY-TO 电子邮件地址来代替 no-reply@verificationemail.com。

自定义 FROM 和 REPLY-TO 电子邮件地址

1. 导航到 Amazon Cognito 控制台，选择 User Pools（用户池）。
2. 从列表中选择一个现有用户池，或创建一个用户池。
3. 选择 Messaging（消息收发）选项卡。在 Email（电子邮件）下，选择 Edit（编辑）。
4. 选择 SES Region（SES 区域）。
5. 从在您所选 SES Region（SES 区域）中已经过 Amazon SES 验证的电子邮件地址列表中，选择 FROM email address（发件人电子邮件地址）。要使用来自已验证域的电子邮件地址，请在 AWS Command Line Interface 或者 AWS API 中配置电子邮件设置。有关更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的在 Amazon SES 中验证电子邮件地址和域。
6. 从您所选 SES Region（SES 区域）中配置集的列表中，选择 Configuration set（配置集）。
7. 为您的电子邮件消息输入易记且格式为 `John Stiles <johnstiles@example.com>` 的 FROM sender name（FROM 发件人名称）。
8. 要自定义 REPLY-TO 电子邮件地址，请在 REPLY-TO email address（REPLY-TO 电子邮件地址）字段中输入有效的电子邮件地址。

授权 Amazon Cognito 代表您发送 Amazon SES 电子邮件（通过自定义 FROM 电子邮件地址）

您可以将 Amazon Cognito 配置为从自定义 FROM 电子邮件地址而不是默认地址发送电子邮件。要使用自定义地址，您必须授予 Amazon Cognito 权限，才能从经过 Amazon SES 验证的身份发送电子邮件。大多数情况下，您可以创建发送授权策略来授予权限。有关更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的使用 Amazon SES 的发送授权。

当您将用户池配置为使用 Amazon SES 处理电子邮件时，Amazon Cognito 会在您的账户中创建 `AWSServiceRoleForAmazonCognitoIdpEmailService` 角色来授予对 Amazon SES 的访问权限。使用 `AWSServiceRoleForAmazonCognitoIdpEmailService` 服务相关角色时无需发送授权策略。只需在

用户池中使用默认电子邮件功能和经过验证的 Amazon SES 身份作为 FROM 地址时，才需要添加发送授权策略。

有关 Amazon Cognito 创建的服务相关角色的更多信息，请参阅对 Amazon Cognito 使用服务相关角色 (p. 504)。

以下示例发送授权策略授予 Amazon Cognito 使用经 Amazon SES 验证的身份的有限能力。Amazon Cognito 在代表 `aws:SourceArn` 中的用户池和 `aws:SourceAccount` 条件中的账户时才能发送电子邮件。有关更多示例，请参阅《Amazon Simple Email Service 开发人员指南》中的 Amazon SES 发送授权策略示例。

Note

在此示例中，“Sid”值为唯一标识语句的任意字符串。有关策略语法的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的 Amazon SES 发送授权策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "stmnt1234567891234",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "email.cognito-idp.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "aws:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
}
```

当您从下拉菜单中选择 Amazon SES 身份时，Amazon Cognito 控制台会为您添加相似策略。如果您使用 CLI 或 API 配置用户池，则必须将与前例结构相同的策略附加到您的 Amazon SES 身份。

将成本分配标签添加到您的用户池

在用户群体属性选项卡的标签下，您可以添加成本分配标签，以对 AWS 成本进行分类和跟踪。当您将标签应用于 AWS 资源（例如 Amazon Cognito 用户池）时，您的 AWS 成本分配报告将包含按标签汇总的使用情况和成本。您可以应用代表业务类别（例如成本中心、应用程序名称或拥有者）的标签，以便跨多项服务组织您的成本。有关更多信息，请参阅 AWS Billing 用户指南中的使用成本分配标签。

要添加标签，请选择管理标签，然后选择添加标签或添加其他。按照标签限制中列出的限制，输入键和值。选择 Save changes（保存更改）以保存您的标签。

Important

为了使标签显示在您的账单报告上，您必须在账单控制台中激活您应用的标签。有关更多信息，请参阅《AWS Billing 用户指南》中的激活用户定义的成本分配标签。

指定用户池设备跟踪设置

作为进一步提高安全性的一种方式，您可以跟踪用户登录的设备。本主题介绍如何在AWS Management Console中将设备跟踪添加到您的 Amazon Cognito 用户池。

设置记忆设备

借助 Amazon Cognito 用户池，您可以选择让 Amazon Cognito 记住用于访问您的应用程序的设备，并在用户池中将这些记住的设备与应用程序用户关联。如果您设置了多重验证 (MFA)，还可以选择使用记忆设备来停止向用户发送代码。您必须使用 USER_SRP_AUTH 身份验证流程才能使用设备跟踪功能。要使用记住的设备替换 MFA，您还必须为您的用户群体启用 MFA。

在用户群体的登录体验选项卡的设备跟踪下配置设备记忆功能。通过 Amazon Cognito 控制台设置记忆设备功能时，有三种选项供您选择：Always (始终)、User Opt-In (用户选择加入) 和 No (否)。

- 请勿记住（原定设置）- 不记住设备。
- 始终记住 - 记住应用程序的用户使用的每台设备。
- User Opt-In (用户选择加入) - 只有当您的用户选择记住设备时，该设备才会被记住。

如果选择了始终记住或用户选择加入，那么对于每台设备，系统都会在用户首次使用该设备登录时，为其分配一个设备标识符（键和密钥）。除了识别设备，此密钥不会用于其他任何地方，但会被服务跟踪。

如果您选择始终记住，则在用户身份验证流程中，Amazon Cognito 将使用设备标识符（键和密钥）在每个用户使用设备登录时对该设备进行身份验证。

如果您选择用户选择加入，则只有当应用程序的用户选择记住设备时，您才能记住该设备。当用户使用新设备登录时，请求启动跟踪的响应将指示是否应该提示用户记住他们的设备。您必须创建用户界面来提示用户。如果用户选择记住设备，则系统会将设备状态更新为“已记住”状态。

AWS Mobile SDK 还提供了其他 API，可用来查看记住的设备（ListDevices、GetDevice）、将设备标记为“已记住”或“未记住”（UpdateDeviceStatus），以及停止跟踪设备（ForgetDevice）。在 REST API 中，还提供了这些 API 的管理员版本，这些版本的 API 具备提升的权限，适用于任何用户。它们的 API 名称分别为 AdminListDevices、AdminGetDevice，等等。它们不通过开发工具包提供。

使用记忆设备停止多重身份验证（MFA）

如果您选择了始终记住或用户选择加入，则还可以选择为应用程序的用户在记住的设备上停止 MFA 质询。要使用此功能，您必须为您的用户池启用 MFA。有关更多信息，请参阅向用户池添加 MFA (p. 220)。

> Note
>
> 如果设备记住功能设置为始终，并且允许用户权限的信任设备绕过 MFA 设置为是，则将忽略基于风险的 MFA 中用于中/高风险的 MFA 设置。

配置用户池应用程序客户端

应用程序是用户池中的一个实体，它有权限调用未经身份验证的 API 操作。未经身份验证的 API 操作是指没有经过身份验证的用户的操作。示例包括注册、登录和处理忘记密码的操作。要调用这些 API 操作，您需要应用程序客户端 ID 和可选的客户端密钥。您有责任确保任何应用程序客户端 ID 或密钥的安全，以便只有经过授权的客户端应用程序才能调用这些未经身份验证的操作。

您可以为一个用户池创建多个应用程序。通常，一个应用程序对应一个应用程序的平台。例如，您可以为服务器端应用程序和其他 Android 应用程序创建一个应用程序。每个应用程序都有各自的应用程序客户端 ID。

在 Amazon Cognito 中创建应用程序客户端时，您可以根据标准 OAuth 客户端类型公有客户端和机密客户端预填充选项。使用客户端密钥配置机密客户端

公有客户端

公有客户端在浏览器或移动设备上运行。由于它没有可信的服务器端资源，所以它没有客户端密钥。

机密客户端

机密客户端拥有可以信任的服务器端资源，使用客户端密钥进行未经身份验证的 API 操作。该应用程序可在后端服务器上作为守护进程或 Shell 脚本运行。

客户端密钥

客户端密钥是一个固定字符串，您的应用程序必须在发送到应用程序客户端的所有 API 请求中使用该字符串。您的应用程序客户端必须有客户端密钥才能执行 `client_credentials` 授权。

您在创建应用程序后无法更改密钥。如果要轮换密钥，您可以创建一个具备新私有密钥的新应用程序。您也可以删除应用程序，以便阻止使用该应用程序客户端 ID 的应用程序的访问。

您可以将机密客户端和客户端密钥用于公有应用程序。使用 Amazon CloudFront 代理添加传输中的 SECRET_HASH。有关更多信息，请参阅 AWS 博客上的 Protect public clients for Amazon Cognito by using an Amazon CloudFront proxy（使用 Amazon CloudFront 代理保护 Amazon Cognito 的公共客户端）。

创建应用程序客户端（AWS Management Console）

创建应用程序客户端（控制台）

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 App integration（应用程序集成）选项卡。
5. 选择 App clients（应用程序客户端）下的 Create an app client（创建应用程序客户端）。
6. 选择 App type（应用程序类型）：Public client（公有客户端）、Confidential client（机密客户端），或者 Other（其它）。
7. 输入 App client name（应用程序客户端名称）。
8. 选择生成客户端密钥，让 Amazon Cognito 为您生成一个客户端密钥。客户端密钥通常与机密客户端关联。
9. 选择您希望在应用程序客户端中允许的 Authentication flows（身份验证流程）。
10. 配置 Authentication flow session duration（身份验证流程会话持续时间）。这是您的用户在会话令牌过期之前必须完成每个身份验证质询的时间。
11. （可选）如果您要配置令牌过期时间，请完成以下步骤：
    a. 指定应用程序客户端的 Refresh token expiration（刷新令牌的过期时间）。默认值为 30 天。您可以将其更改为 1 小时到 10 年之间的任何值。
    b. 指定应用程序客户端的 Access token expiration（访问令牌的过期时间）。默认值为 1 小时。您可以将其更改为 5 分钟到 24 小时之间的任何值。
    c. 指定应用程序客户端的 ID token expiration（ID 令牌的过期时间）。默认值为 1 小时。您可以将其更改为 5 分钟到 24 小时之间的任何值。

       Important

       如果您使用托管 UI 并将令牌生命周期配置为不到一小时，用户将能够根据其会话 Cookie 持续时间使用令牌，该 Cookie 当前固定在一小时。
12. 是否为此应用程序客户端选择 Enable token revocation（启用令牌撤消）。这将增加 Amazon Cognito 发出的令牌的大小。
13. 选择您是否将针对此应用程序客户端防止用户已存在错误。Amazon Cognito 将响应不存在的用户的登录请求，并显示一条指出用户名或密码不正确的通用消息。
14. 如果要在此应用程序客户端中使用托管 UI，请配置托管 UI 设置。

a. 输入一个或多个允许的回调 URL。这些是您希望 Amazon Cognito 在用户完成身份验证后重定向他们的 Web 或应用程序 URL。
b. 输入一个或多个允许的注销 URL。这些是您希望应用程序在对注销端点 (p. 554)的请求中接受的 URL。
c. 选择一个或多个您希望用户能够登录应用程序的身份提供者。您可以选择现有 IdP 的任意组合。您可以单独使用您的用户群体或使用您在用户群体中配置的一个或多个第三方 IdP，对用户进行身份验证。
d. 选择您希望应用程序客户端接受的 OAuth 2.0 授权类型。
   - 选择授权码授权以将代码传递给您的应用程序，该应用程序可以向令牌端点 (p. 562)兑换令牌。
   - 选择隐式授权，以将 ID 令牌和访问令牌直接传递给您的应用程序。隐式授予流程将令牌直接透露给您的用户。
   - 选择客户端凭证以将访问令牌传递给您的应用程序，这不是基于它对用户凭证的了解，而是基于它对客户端密钥的了解。客户端凭证授予流程与授权码和隐式授予流程是互斥的。
e. 选择您要授权与应用程序客户端结合使用的 OpenID Connect 范围。您可以通过用户群体 API 生成仅限 `aws.cognito.signin.user.admin` 范围的访问令牌。对于其他范围，您必须从令牌端点 (p. 562)请求访问令牌。
f. 选择您要授权与应用程序客户端结合使用的自定义范围。自定义范围最常用于授权对第三方 API 的访问权限。

15. 为此应用程序客户端配置属性读取和写入权限。您的应用程序客户端有权限读取和写入用户群体属性模式的全部或有限子集。
16. 选择 Create app client（创建应用程序客户端）。
17. 记下 Client id（客户端 ID）。这将识别注册和登录请求中的应用程序客户端。

在用户池中创建和更新应用程序客户端（API、AWS CLI）

请执行以下任一操作。更新应用程序客户端时，必须在请求中包含整个应用程序客户端配置。Amazon Cognito 会将空白配置设置设为原定设置值，即使您之前分配了值也是如此。

- API – 使用 CreateUserPoolClient 和 UpdateUserPoolClient 操作。
- AWS CLI – 在命令行处，运行 `create-user-pool-client` 和 `update-user-pool-client` 命令。

配置用户池 Lambda 触发器

您可以使用 AWS Lambda 触发器自定义工作流和 Amazon Cognito 用户体验。您可以创建以下 Lambda 触发器：注册前、身份验证前、自定义消息、身份验证后、确认后、定义身份验证质询、创建身份验证质询、验证身份验证质询响应和用户迁移。

可以使用用户迁移 Lambda 触发器轻松地将用户从现有用户管理系统迁移到您的用户池。

有关每个 Lambda 触发器的示例，请参阅使用 Lambda 触发器自定义用户池工作流 (p. 111)。

Note

自定义消息 AWS Lambda 触发器是自定义电子邮件和 SMS 消息的一种高级方式。有关更多信息，请参阅 使用 Lambda 触发器自定义用户池工作流 (p. 111)。

配置用户池分析

使用 Amazon Pinpoint 分析，您可以跟踪 Amazon Cognito 用户池的注册、登录、失败的身份验证、日活跃用户（DAU）和月活跃用户（MAU）。您还可以使用 AWS Mobile SDK for Android 或 AWS Mobile SDK for iOS 设置特定于您的应用程序的用户属性。然后可以使用这些属性在 Amazon Pinpoint 中对用户进行细分，并向他们发送有针对性的推送通知。

在应用程序集成选项卡的应用程序客户端和分析下，您可以导航到现有应用程序客户端或创建新的应用程序客户端。在应用程序客户端的配置中，您可以指定要与应用程序结合使用的 Amazon Pinpoint 项目。有关更多信息，请参阅将 Amazon Pinpoint 分析与 Amazon Cognito 用户池结合使用 (p. 164)。

Note

Amazon Pinpoint 在北美、欧洲、亚洲和大洋洲的多个AWS区域中均已推出。Amazon Pinpoint 区域包括 Amazon Pinpoint API。如果 Amazon Cognito 支持 Amazon Pinpoint 区域，Amazon Cognito 会将事件发送到同一 Amazon Pinpoint 区域中的 Amazon Pinpoint 项目。如果区域不受 Amazon Pinpoint 支持，Amazon Cognito 将仅支持在 us-east-1 中发送事件。有关 Amazon Pinpoint 区域的详细信息，请参阅 Amazon Pinpoint 端点和配额和将 Amazon Pinpoint 分析与 Amazon Cognito 用户池结合使用。

添加分析和市场活动

1. 选择添加分析和活动。
2. 从列表中选择 Cognito app client (Cognito 应用程序客户端)。
3. 要将 Amazon Cognito 应用程序映射到 Amazon Pinpoint 项目，请从列表中选择该 Amazon Pinpoint 项目。

   Note

   Amazon Pinpoint 项目 ID 是 Amazon Pinpoint 项目特有的由 32 个字符组成的字符串。它在 Amazon Pinpoint 控制台中列出。
   您可以将多个 Amazon Cognito 应用程序映射到单个 Amazon Pinpoint 项目。但是，每个 Amazon Cognito 应用程序只能映射到一个 Amazon Pinpoint 项目。
   在 Amazon Pinpoint 中，每个项目都应该是单个应用程序。例如，如果游戏开发员有两款游戏，每款游戏应该是单独的 Amazon Pinpoint 项目，即使这两款游戏使用同一 Amazon Cognito 用户池。
4. 如果您希望 Amazon Cognito 向 Amazon Pinpoint 发送电子邮件地址和电话号码，以便为用户创建额外的端点，请选择 Share user attribute data with Amazon Pinpoint (与 Amazon Pinpoint 共享用户属性数据)。

   Note

   端点唯一地标识可以使用 Amazon Pinpoint 向其发送推送通知的用户设备。有关端点的更多信息，请参阅《Amazon Pinpoint 开发人员指南》中的 Adding endpoints to Amazon Pinpoint（向 Amazon Pinpoint 添加端点）。
5. 输入您已经创建的 IAM 角色，或者选择 Create new role (创建新角色) 以在 IAM 控制台中创建一个新角色。
6. 选择 Save changes（保存更改）。
7. 要指定额外的应用程序映射，请选择添加应用程序映射。
8. 选择 Save changes（保存更改）。

为用户池添加域名

在 App integration（应用程序集成）选项卡中，创建一个 Amazon 拥有的前缀域或为您的用户池使用自定义域。您应用程序的前缀域为 `https://<domain_prefix>.auth.<region>.amazoncognito.com`。

您的应用程序的完整 URL 类似于此示例：

```
https://example.auth.us-east-1.amazoncognito.com/login?redirect_uri=https://
www.google.com&response_type=code&client_id=<client_id_value>
```

有关更多信息，请参阅配置用户池域 (p. 61)。

Important

您必须先指定应用程序客户端设置（例如回调和重定向 URL），然后才能访问应用程序的托管 UI。有关更多信息，请参阅配置用户池应用程序客户端 (p. 258)。

配置域

1. 导航到用户池的 App integration（应用程序集成）选项卡。
2. 选择在 Domain（域）旁边的 Actions（操作），然后选择 Create custom domain（创建自定义域）或 Create Cognito domain（创建 Cognito 域）。如果您已配置用户池域，请先选择 Delete Cognito domain（删除 Cognito 域）或 Delete custom domain（删除自定义域），然后再创建新的自定义域。
3. 输入可用的域前缀，将其与 Cognito domain（Cognito 域）结合使用。您的前缀不能包含字符串 `aws`、`amazon` 或者 `cognito`。有关设置 Custom domain（自定义域）的信息，请参阅将您自己的域用于托管 UI
4. 选择 Create（创建）。

自定义内置应用程序 UI 以便注册用户和让用户登录

在 App integration（应用程序集成）选项卡上，借助 Hosted UI customization（托管 UI 自定义），您可以将自己的自定义项添加到默认应用程序 UI。

您还可以为应用程序客户端自定义托管 UI。从 App integration（应用程序集成）选项卡上选择应用程序客户端，并找到 Hosted UI customization（托管 UI 自定义）。选择 Use client-level settings（使用客户端级别设置）以便仅进入该应用程序客户端的 UI 自定义屏幕。

有关每个自定义字段的详细信息，请参阅自定义内置登录网页和注册网页 (p. 67)。

Note

通过利用您的用户池的特定信息构建以下 URL 并将它键入到浏览器中，您可以查看具有自定义项的托管 UI： `https://<your_domain>/login?response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>`
在控制台中进行的更改出现之前，您可能必须等待长达 1 分钟才能刷新浏览器。
您的域显示在 Domain name（域名）选项卡上。您的应用程序客户端 ID 和回调 URL 将显示在 General settings（常规设置）选项卡上。
您还可以为启用了托管 UI 的应用程序客户端启动托管 UI。在 App integration（应用程序集成）选项卡上选择应用程序客户端，并找到 Hosted UI settings（托管 UI 设置）。选择 View hosted UI（查看托管 UI）。

自定义内置应用程序 UI

1. 要上载自己的徽标图像文件，请选择 Choose file（选择文件）或 Replace current file（替换当前文件）。
2. 要为托管 UI 自定义 CSS，请下载 CSS template.css 然后使用您希望自定义的值修改模板。只有模板中包含的密钥才能与托管 UI 一起使用。添加的 CSS 密钥不会反映在您的 UI 中。自定义 CSS 后，请选择 Choose file（选择文件）或 Replace current file（替换当前文件）来上载您的自定义 CSS。
3. 选择 Save changes（保存更改）。

为用户池配置身份提供商

在登录体验选项卡的联合身份提供者登录下，您可以将身份提供者（IdP）添加到您的用户群体。有关更多信息，请参阅通过第三方添加用户池登录 (p. 85)。

主题

- 使用社交 IdP 设置用户登录 (p. 263)

- 使用 OIDC IdP 设置用户登录 (p. 264)
- 使用 SAML IdP 设置用户登录 (p. 265)

使用社交 IdP 设置用户登录

您可以使用联合身份验证，将 Amazon Cognito 用户池与社交身份提供商（如 Facebook、Google 和 Login with Amazon）集成起来。

要添加社交身份提供商，您首先要通过该身份提供商创建一个开发人员账户。在拥有开发人员账户后，您应向身份提供商注册您的应用程序。该身份提供商将为您的应用程序创建应用程序 ID 和应用程序密钥，然后您在您的 Amazon Cognito 用户池中配置这些值。

- Google Identity Platform
- Facebook for Developers
- Login with Amazon
- 通过 Apple 登录

将用户登录与社交 IdP 集成

1. 登录 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 在导航窗格中，选择 User Pools（用户池），然后选择要编辑的用户池。
3. 选择 Sign-in experience（登录体验）选项卡并找到 Federated sign-in（联合登录）。
4. 选择 Add an identity provider（添加身份提供商），或者选择已配置的身份提供商（例如 Facebook、Google、Amazon 或 Apple），找到 Identity provider information（身份提供商信息），然后选择 Edit（编辑）。有关添加社交身份提供商的更多信息，请参阅向用户池添加社交身份提供商 (p. 86)。
5. 根据您选择的 IdP，完成以下步骤之一，从而输入社交身份提供商的信息：

   Facebook、Google 和 Login with Amazon

   输入您创建客户端应用程序时收到的应用程序密钥。

   Sign In with Apple

   输入您向 Apple 提供的服务 ID，以及创建应用程序客户端时收到的团队 ID、密钥 ID 和私有密钥。
6. 对于 Authorize scopes（授权范围），输入要映射到用户池属性的社交身份提供商范围的名称。范围定义了您要通过应用程序访问的用户属性（如名称和电子邮件）。输入范围时，根据您选择的 IdP 使用以下准则：

   - Facebook – 以英文逗号分隔范围。例如：

     ```
     public_profile, email
     ```
   - Google、Login with Amazon 和 Sign In with Apple – 以空格分隔范围。例如：
     - Google: `profile email openid`
     - Login with Amazon: `profile postal_code`
     - Sign In with Apple: `name email`

     Note

     对于 Sign in with Apple（控制台），请使用复选框以选择范围。
7. 选择 Save changes（保存更改）。
8. 从 App client integration（应用程序客户端集成）选项卡上的列表中选择一个 App clients（应用程序客户端），然后选择 Edit hosted UI settings（编辑托管 UI 设置）。将新的社交身份提供商添加到 Identity providers（身份提供商）下的应用程序客户端。

9. 选择 Save changes（保存更改）。

有关社交 IdP 的更多信息，请参阅 向用户池添加社交身份提供商 (p. 86)。

使用 OIDC IdP 设置用户登录

您可以将用户登录与 OpenID Connect (OIDC) 身份提供商 (IdP) 集成，例如 Salesforce 或 Ping Identity。

向用户群体添加 OIDC 提供者

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 从导航菜单中选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 Sign-in experience（登录体验）选项卡。找到 Federated sign-in（联合登录），然后选择 Add an identity provider（添加身份提供商）。
5. 选择 OpenID Connect 身份提供商。
6. 在 Provider name（提供商名称）中输入一个唯一名称。
7. 将您从提供商那里收到的客户端 ID 输入到 Client ID（客户端 ID）。
8. 将您从提供商那里收到的客户端密钥输入到 Client secret（客户端密钥）。
9. 为该提供商输入 Authorized scopes（授权范围）。范围定义了应用程序将向您的提供商请求的用户属性组（例如 `name` 和 `email`）。根据 OAuth 2.0 规范，范围必须以空格分隔。

   您的用户必须同意向您的应用程序提供这些属性。
10. 请选择一个 Attribute request method（属性请求方法），以便为 Amazon Cognito 提供 HTTP 方法（GET 或 POST），Amazon Cognito 使用该方法从提供商运营的 userInfo 端点中获取用户的详细信息。
11. 请选择 Setup method（设置方法）并通过 Auto fill through issuer URL（自动填充发布者 URL）或 Manual input（手动输入）检索 OpenID Connect 端点。当您的提供商具有公有 `.well-known/openid-configuration` 端点且 Amazon Cognito 可以在其中检索 `authorization`、`token`、`userInfo` 和 `jwks_uri` 端点的 URL 时，使用 Auto fill through issuer URL（自动填充发布者 URL）。
12. 请输入发布者 URL 或 IdP 中的 `authorization`、`token`、`userInfo` 和 `jwks_uri` 端点 URL。

    Note

    您只能将端口号 443 和 80 用于发现、自动填充和手动输入的 URL。如果您的 OIDC 提供商使用任何非标准 TCP 端口，则用户登录失败。
    发布者 URL 必须以 `https://` 开头，而且不得以 `/` 字符结尾。例如，Salesforce 使用以下 URL：
    `https://login.salesforce.com`
    与您的发布者 URL 关联的 `openid-configuration` 文档必须为以下值提供 HTTPS URL：`authorization_endpoint`、`token_endpoint`、`userinfo_endpoint` 和 `jwks_uri`。同样，当您选择 Manula input（手动输入）时，只能输入 HTTPS URL。
13. 默认情况下，sub OIDC 声明将映射到用户池 Username（用户名）属性中。您可以将其他 OIDC 声明映射到用户池属性。输入 OIDC 声明，然后从下拉列表中选择对应的用户池属性。例如，声明 email 通常会映射到用户池属性 Email（电子邮件）。
14. 请将身份提供商的其它属性映射到您的用户池。有关更多信息，请参阅指定适用于用户池的身份提供程序属性映射。
15. 选择 Create（创建）。
16. 从 App client integration（应用程序客户端集成）选项卡上的列表中选择 App clients（应用程序客户端）的其中一个，然后选择 Edit hosted UI settings（编辑托管 UI 设置）。将新的 OIDC 身份提供商添加到 Identity providers（身份提供商）下的应用程序客户端。
17. 选择 Save changes（保存更改）。

有关 OIDC IdP 的更多信息，请参阅向用户池添加 OIDC 身份提供商 (p. 102)。

使用 SAML IdP 设置用户登录

您可以使用 Amazon Cognito 用户池的联合身份验证与 SAML 身份提供商（IdP）集成。您可以通过上传文件或输入元数据文档端点 URL 来提供元数据文档。有关获取第三方 SAML IdP 元数据文档的信息，请参阅将第三方 SAML 身份提供商与 Amazon Cognito 用户池集成 (p. 99)。

在您的用户池中配置 SAML 2.0 身份提供商

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 Sign-in experience（登录体验）选项卡。找到 Federated sign-in（联合登录），然后选择 Add an identity provider（添加身份提供商）。
5. 选择 SAML 身份提供商。
6. 输入以逗号分隔的 Identifiers（标识符）。标识符指示 Amazon Cognito 检查用户登录电子邮件地址，然后将用户引导到与其域对应的提供商。
7. 如果您希望 Amazon Cognito 在用户注销时向您的提供商发送已签名的注销请求，请选择 Add sign-out flow（添加注销流程）。配置 SAML 2.0 身份提供商，向您配置托管 UI 时 Amazon Cognito 创建的 `https://<your Amazon Cognito domain>/saml2/logout` 端点发送注销响应。此 `saml2/logout` 端点使用 POST 绑定。

   Note

   如果选择此选项，并且您的 SAML 身份提供商需要已签名的注销请求，则您还需要为您的 SAML IdP 配置 Amazon Cognito 提供的签名证书。
   SAML IdP 将处理已签名的注销请求并从 Amazon Cognito 会话中注销您的用户。
8. 选择 Metadata document source（元数据文档源）。如果您的身份提供商在公有 URL 上提供 SAML 元数据，则可以选择 Metadata document URL（元数据文档 URL），然后输入该公有 URL。否则，请选择 Upload metadata document（上载元数据文档），然后选择您之前从提供商下载的元数据文件。

   Note

   如果您的提供商具有公有端点，我们建议您输入元数据文档 URL，而不是上载文件。如果您使用 URL，Amazon Cognito 会自动刷新元数据。通常，元数据刷新操作每 6 小时执行一次或在元数据过期前执行（以时间较早者为准）。
9. Map attributes between your SAML provider and your app（在 SAML 提供商和应用程序之间映射属性）将 SAML 提供程序属性映射到用户池中的用户配置文件。在属性映射中包含用户池必需属性。

   例如，当您选择 User pool attribute（用户池属性）`email` 时，按照您的身份提供商提供的 SAML 断言中显示的内容，输入 SAML 属性名称。您的身份提供商可能会提供示例 SAML 断言以供参考。一些身份提供商使用简单名称（如 `email`），另一些则使用类似于下面 URL 格式的属性名称：

   ```
   http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
   ```
10. 选择 Create（创建）。

    Note

    如果您在使用 HTTPS 元数据端点 URL 创建 SAML IdP 时看见 `InvalidParameterException`，请确保元数据端点已正确设置 SSL，并且存在与之关联的有效 SSL 证书。这种例外的一个例子是“Error retrieving metadata from *<metadata endpoint>*”（从 <元数据端点> 检索元数据时出错）。

设置 SAML IdP 以添加签名证书

- 要获取包含 IdP 用于验证已签名注销请求的公有密钥的证书，请在联合身份验证控制台页面上的身份提供者下，在 SAML 对话框上的使用中的 SAML 提供者下选择显示签名证书对话框。

有关 SAML IdP 的更多信息，请参阅 向用户池添加 SAML 身份提供商 (p. 92)。

为用户池配置属性映射

在登录体验选项卡下，在您在用户群体中配置的每个身份提供者（IdP）的配置中，您可以将 IdP 属性或断言映射到用户群体属性。有关更多信息，请参阅指定适用于用户池的身份提供商属性映射 (p. 107)。

Note

目前，仅 Facebook `id`、Google `sub`、Login with Amazon `user_id` 和 Sign in with Apple `sub` 属性可映射到 Amazon Cognito 用户池 `username` 属性。

Note

用户池中的属性必须足够大，以包含映射的身份提供商属性的值，否则用户登录时将发生错误。如果将自定义属性映射到身份提供商令牌，则应将其大小设置为最大 2048 个字符。
您必须为您的用户池所需的任何属性创建映射。

为您的用户池指定社交身份提供商属性映射。

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 从导航菜单中选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 Sign-in experience（登录体验）选项卡。找到联合身份提供者登录。
5. 选择 Add an identity provider（添加身份提供商），或者选择您已配置的身份提供商，如 Facebook、Google、Amazon 或 Apple。找到 Attribute mapping（属性映射），然后选择 Edit（编辑）。有关添加社交身份提供商的更多信息，请参阅向用户池添加社交身份提供商 (p. 86)。
6. 对于需要映射的每个属性，请完成以下步骤：
   a. 从 User pool attribute（用户池属性）列中选择属性。这是分配给您的用户池中用户配置文件的属性。自定义属性在标准属性之后列出。
   b. 从 ***<provider>*** attribute（<provider> 属性）列中选择属性。这将是从提供商目录传递的属性。在下拉列表中提供来自社交服务提供商的已知属性。
   c. 要在 IdP 和 Amazon Cognito 之间映射其它属性，请选择 Add another attribute（添加其它属性）。
7. 选择 Save changes（保存更改）。

为您的用户池指定 SAML 身份提供商属性映射

1. 登录 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 在导航窗格中，选择 User Pools（用户池），然后选择要编辑的用户池。
3. 选择 Sign-in experience（登录体验）选项卡并找到 Federated sign-in（联合登录）。
4. 选择 Add an identity provider（添加身份提供商），或者选择您已配置的 SAML 身份提供商。找到 Attribute mapping（属性映射），然后选择 Edit（编辑）。有关添加 SAML 身份提供商的更多信息，请参阅向用户池添加 SAML 身份提供商 (p. 92)。
5. 对于需要映射的每个属性，请完成以下步骤：
   a. 从 User pool attribute（用户池属性）列中选择属性。这是分配给您的用户池中用户配置文件的属性。自定义属性在标准属性之后列出。
   b. 从 SAML attribute（SAML 属性）列中选择属性。这将是从提供商目录传递的属性。

您的身份提供商可能会提供示例 SAML 断言以供参考。一些身份提供商使用简单名称（如 `email`），另一些则使用类似于下面 URL 格式的属性名称：

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

c. 要在 IdP 和 Amazon Cognito 之间映射其它属性，请选择 Add another attribute（添加其它属性）。

6. 选择 Save changes（保存更改）。

Amazon Cognito 用户群体旧版控制台过程

以下过程说明如何在旧版 Amazon Cognito 控制台中执行常见操作。旧版控制台不再可用。AWS 在此维护这些程序，以便为过去用过旧版控制台的客户提供归档参考。

开始使用

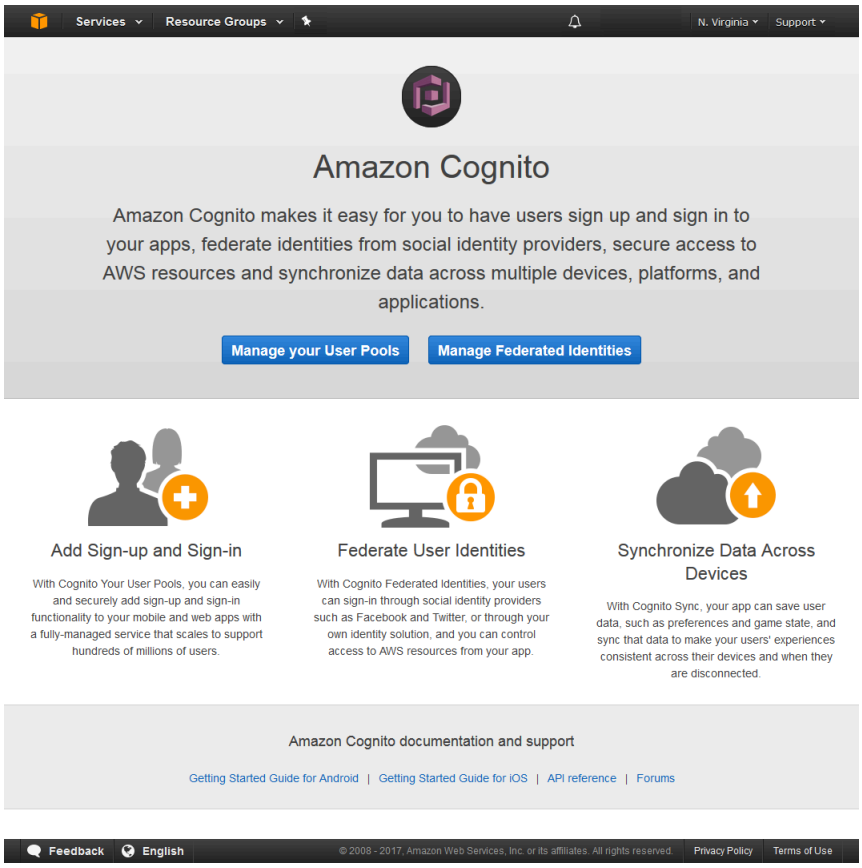
如要使用 Amazon Cognito 控制台

1. 您需要注册AWS账户才能使用 Amazon Cognito。
2. 转到 Amazon Cognito 控制台。系统可能会提示您输入 AWS 凭证。
3. 要创建或编辑用户池，请选择管理您的用户池。

   有关更多信息，请参阅用户池入门 (p. 30)。
4. 要创建或编辑身份池，请选择管理联合身份。

   有关更多信息，请参阅Amazon Cognito 身份池入门（联合身份） (p. 289)。

   Amazon Cognito 控制台是AWS Management Console的一部分，后者提供了有关账户和账单的信息。有关更多信息，请参阅使用 AWS Management Console。



创建用户池

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 Manage User Pools（管理用户池）。
3. 选择 Create a user pool（创建用户池）。
4. 为您的用户池输入一个名称，然后选择 Review defaults（查看默认值）以保存该名称。
5. 在 Review（审查）页面上，选择 Create pool（创建池）。

删除用户池

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 Manage User Pools（管理用户池）。
3. 选择在上一步中创建的用户池。
4. 在 App integration（应用程序集成）下的 Domain name（域名）页面，选择 Delete domain（删除域）。
5. 提示确认时，选择 Delete domain（删除域）。
6. 转到 General Settings（常规设置）页面。
7. 选择页面右上角的 Delete pool（删除池）。
8. 提示进行确认时，请输入 **`delete`** 并选择 Delete pool（删除池）。

将标签添加到用户池

1. 在 https://console.aws.amazon.com/cognito 打开 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 Manage User Pools（管理用户池）。
3. 在 Your User Pools（您的用户池）页面上，选择要添加标签的用户池。
4. 在导航菜单中，选择 Tags（标签）。
5. 选择 Add tag（添加标签）即可添加您的第一个标签。如果您之前已将标签分配给此用户池，请选择 Add another tag（添加其它标签）。
6. 为 Tag Key（标签键）和 Tag Value（标签值）输入一个值。
7. 对于每个要添加的其他标签，请选择 Add another tag（添加其他标签）。
8. 添加完标签后，选择 Save changes（保存更改）。

使用应用程序

在用户群体中创建应用程序

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 Manage User Pools（管理用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 在页面左侧的导航栏上，选择 General settings（常规设置）下的 App clients（应用程序客户端）。
5. 选择 Add an app client（添加应用程序客户端）。
6. 输入应用程序客户端的名称。
7. 在此练习中，请清除选项 Generate client secret（生成客户端密钥）。由于使用具有客户端身份验证的客户端密钥（例如此练习中使用的 JavaScript）并不安全，因此不建议将其用于生产应用程序客户端。客户端密钥仅限由具有以下特点的应用程序使用：拥有可保护客户端密钥的服务器端身份验证组件。
8. 选择 Create app client（创建应用程序客户端）。
9. 记下 App client ID（应用程序客户端 ID）。
10. 选择 Return to pool details（返回池详细信息）。
11. 从控制台页面左侧的导航栏中选择 App client settings（应用程序客户端设置）。
12. 选择 Cognito User Pool（Cognito 用户池）作为其中一个 Enabled Identity Providers（启用的身份提供商）。

    Note

    要使用外部身份提供商（IdP）（如 Facebook、亚马逊、Google 和 Apple）以及通过 OpenID Connect（OIDC）或 SAML IdP 进行登录，请首先按下文所述的方式对它们进行配置，然后返回到 App client settings (应用程序客户端设置) 页面将其启用。
13. 输入回调 URL，供 Amazon Cognito 授权服务器在用户通过身份验证之后进行调用。对于 Web 应用程序，URL 应以 `https://` 开头，例如 `https://www.example.com`。

    对于 iOS 或 Android 应用程序，您可以使用类似 `myapp://` 的回调 URL。
14. 输入注销 URL。
15. 选择 Authorization code grant（授权代码授予）以返回随后更换为用户池令牌的授权代码。由于令牌绝不会直接向终端用户公开，因此它们不太可能被泄露。但是，后端需要自定义应用程序以将授权代码换成用户池令牌。出于安全原因，对于移动应用程序，强烈建议您将授权代码授予流程与代码交换的证明密钥（PKCE）一起使用。
16. 在 Allowed OAuth Flows (允许的 OAuth 流程) 下，选择 Implicit grant (隐式授予) 以便从 Amazon Cognito 将用户池 JSON Web 令牌（JWT）返回给您。当没有可用于将授权代码更换为令牌的后端时，您可以使用此流程。它对于调试令牌也很有帮助。

Note

您可以同时启用 Authorization code grant（授权代码授予）和 Implicit code grant（隐式代码授予），然后按需使用每个授予。

17. 除非您希望明确地排除一个允许的 OAuth 范围，否则请选中所有 Allowed OAuth scopes（允许的 OAuth 范围）的复选框。

    Note

    只有在您的应用程序需要代表自己而不是代表用户请求访问令牌时才选择 Client credentials (客户端凭证)。

18. 选择 Save changes（保存更改）。
19. 在 Domain name（域名）页面上，键入可用的域前缀。
20. 记下完整的域地址。
21. 选择 Save changes（保存更改）。

创建应用程序客户端

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 Manage User Pools（管理用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 在页面左侧的导航栏上，选择在 General settings（常规设置）下的 App clients（应用程序客户端）。
5. 选择 Add an app client（添加应用程序客户端）。
6. 输入应用程序的名称。
7. 除非您的授权流程有要求，否则应清除选项Generate client secret（生成客户端密钥）。客户端密钥由具有以下特点的应用程序使用：拥有可保护客户端密钥的服务器端组件。
8. （可选）更改令牌过期设置。
9. 选择 Auth Flows Configuration（身份验证流配置）选项。有关更多信息，请参阅用户池身份验证流程。
10. 选择 Security configuration（安全配置）。我们建议您选择 Enabled (已启用)。
11. （可选）选择 Set attribute read and write permissions（设置属性读取和写入权限）。有关更多信息，请参阅属性权限和范围。
12. 选择 Create app client（创建应用程序客户端）。
13. 记下 App client id（应用程序客户端 ID）。
14. 选择 Return to pool details（返回池详细信息）。

为用户池指定应用程序客户端设置

1. Note

   只有在您编辑现有用户池时才会显示 General settings（常规设置）选项卡。

   如果您希望使用内置托管页面注册和登录用户，或者希望使用 OAuth2.0 流，则必须在 General settings（常规设置）选项卡上，至少为应用程序配置一个身份提供商（IdP）。有关更多信息，请参阅配置用户池应用程序客户端 (p. 58)。

2. 在 Enabled Identity Providers（启用的身份提供商）中，为您在 App Clients（应用程序客户端）选项卡中配置的应用程序选择您要使用的身份提供商。
3. 输入您所需的 Callback URLs (回调 URL)（以逗号分隔）。这些 URL 应用到所有选定的身份提供商。

   Note

   您必须先在控制台中或者使用 AWS CLI 或 API 来注册 URL，然后才能在应用程序中使用它们。

4. 输入您所需的 Sign out URLs (注册 URL) (以逗号分隔)。

   Note

   您必须先在控制台中或者使用 CLI 或 API 来注册 URL，然后才能在应用程序中使用它们。

5. 在 OAuth 2.0 下，从以下选项中进行选择。有关更多信息，请参阅 应用程序客户端设置术语 (p. 60) 和 OAuth 2.0 规范。

   - 对于 Allowed OAuth Flows (允许的 OAuth 流程)，选择 Authorized code grant (授权代码授予) 和 Implicit grant (隐式授予)。只有在您的应用程序需要代表自己而不是代表用户请求访问令牌时才选择 Client credentials (客户端凭证)。
   - 对于 Allowed OAuth Scopes (允许的 OAuth 范围)，选择您所需的范围。每个范围是包含一个或多个标准属性的集合。
   - 对于 Allowed Custom Scopes (允许的自定义范围)，从您定义的任何自定义范围中选择所需的范围。自定义范围在 Resource Servers (资源服务器) 选项卡中进行定义。有关更多信息，请参阅为您的用户池定义资源服务器 (p. 82)。

指定分析设置

1. 登录 Amazon Cognito 控制台。
2. 在导航窗格中，选择 Manage User Pools (管理用户池)，然后选择要编辑的用户池。
3. 选择分析选项卡。
4. 选择添加分析和活动。
5. 从列表中选择 Cognito app client (Cognito 应用程序客户端)。
6. 要将 Amazon Cognito 应用程序映射到 Amazon Pinpoint 项目，请从列表中选择该 Amazon Pinpoint 项目。

   Note

   Amazon Pinpoint 项目 ID 是 Amazon Pinpoint 项目特有的由 32 个字符组成的字符串。它在 Amazon Pinpoint 控制台中列出。
   您可以将多个 Amazon Cognito 应用程序映射到单个 Amazon Pinpoint 项目。但是，每个 Amazon Cognito 应用程序只能映射到一个 Amazon Pinpoint 项目。
   在 Amazon Pinpoint 中，每个项目都应该是单个应用程序。例如，如果游戏开发员有两款游戏，每款游戏应该是单独的 Amazon Pinpoint 项目，即使这两款游戏使用同一 Amazon Cognito 用户池。有关 Amazon Pinpoint 项目的更多信息，请参阅在 Amazon Pinpoint 中创建项目。

7. 如果您希望 Amazon Cognito 向 Amazon Pinpoint 发送电子邮件地址和电话号码，以便为用户创建额外的端点，请选择 Share user attribute data with Amazon Pinpoint (与 Amazon Pinpoint 共享用户属性数据)。验证账户电话号码和电子邮件地址后，只有在它们可用于用户账户时，它们才会与 Amazon Pinpoint 共享。

   Note

   端点唯一地标识可以使用 Amazon Pinpoint 向其发送推送通知的用户设备。有关端点的更多信息，请参阅《Amazon Pinpoint 开发人员指南》中的添加端点。

8. 选择 Save changes（保存更改）。
9. 要指定额外的应用程序映射，请选择添加其他应用程序映射。
10. 选择 Save changes（保存更改）。

通过控制台更新属性权限

1. 转到 AWS Management Console中的 Amazon Cognito。如果控制台提示您，请输入 AWS 凭证。
2. 选择 Manage User Pools（管理用户池）。

3. 在 Your User Pools（您的用户池）页面上，选择要配置的用户池。
4. 在左侧导航栏中，选择 App clients（应用程序客户端）。
5. 为要更新的应用程序客户端选择 Show Details（显示详细信息）。
6. 在页面底部，选择 Set attribute read and write permissions（设置属性读取和写入权限），然后配置读取和写入权限。
7. 选择 Save app client changes（保存应用程序客户端更改）。

使用托管 UI

配置域

1. 登录 Amazon Cognito 控制台。
2. 在导航窗格中，选择 Manage your User Pools（管理您的用户池），然后选择要编辑的用户池。
3. 选择域名选项卡。
4. 在前缀域名框中键入要使用的域前缀。
5. 选择 Check availability（检查可用性）以确认域前缀可用。
6. 选择 Save changes（保存更改）。

指定域名

> Note
>
> 只有在您编辑现有用户池时才会显示 Domain name（域名）选项卡。

在域名选项卡上，您可以输入自己的前缀域名。您应用程序的域为 `https://<domain_prefix>.auth.<region>.amazoncognito.com`。

您的应用程序的完整 URL 类似于此示例：

```
https://example.auth.us-east-1.amazoncognito.com/login?redirect_uri=https://
www.google.com&response_type=code&client_id=<client_id_value>
```

有关更多信息，请参阅配置用户池域 (p. 61)。

> Important
>
> 您必须先指定应用程序客户端设置 (例如回调和重定向 URL)，然后才能访问应用程序的 URL。有关更多信息，请参阅配置用户池应用程序客户端 (p. 258)。

为用户池指定域名

1. 在 Prefix domain name（前缀域名）框中输入您所需的域名。
2. 选择 Check availability（检查可用性）验证您所需的前缀域名是否可用。
3. 如果前缀域名可供使用，请选择 Save changes（保存更改）。

更新域证书

1. 登录AWS Management Console并打开 Amazon Cognito 控制台 (https://console.aws.amazon.com/cognito/home)。
2. 选择 Manage User Pools（管理用户池）。
3. 在 Your User Pools（您的用户池）页面上，选择要向其添加域的用户池。

4. 在左侧导航菜单上，选择 Domain name（域名）。
5. 在 Your own domain (您自己的域) 下，对于 AWS managed certificate (亚马逊云科技托管证书)，选择您的新证书。
6. 选择 Save changes（保存更改）。

指定应用程序 UI 自定义设置

1. Note

   通过利用您的用户池的特定信息构建以下 URL 并将它键入到浏览器中，您可以查看具有自定义项的托管 UI： `https://<your_domain>/login?response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>`
   在控制台中进行的更改出现之前，您可能必须等待长达 1 分钟才能刷新浏览器。
   您的域显示在 Domain name（域名）选项卡上。您的应用程序客户端 ID 和回调 URL 将显示在 General settings（常规设置）选项卡上。

   登录 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 在导航窗格中，选择 Manage User Pools（管理用户池），然后选择要编辑的用户池。
3. 选择 UI 自定义项选项卡。
4. 在 App client to customize (用于自定义的应用程序客户端) 下，选择要从您之前在 App clients 选项卡中创建的应用程序客户端的下拉菜单自定义的应用程序。
5. 要上传您自己的徽标图像文件，请选择 Choose a file (选择文件) 或者将文件拖动到 Logo (optional) (日志（可选）) 框中。
6. 在 CSS customizations (optional)（CSS 自定义项（可选））下，您可以通过更改 CSS 属性的默认值来自定义应用程序的外观。

定义资源服务器

1. 登录 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 在导航窗格中，选择 Manage User Pools（管理用户池），然后选择要编辑的用户池。
3. 选择资源服务器选项卡。
4. 选择 Add a resource server（添加资源服务器）。
5. 输入您的资源服务器的名称。例如，`Photo Server`。
6. 输入您的资源服务器的标识符。例如，`com.example.photos`。
7. 输入您的资源的自定义范围的名称（例如 `read` 和 `write`）。
8. 对于每个 scope name（范围名称），输入一个 description（描述），如 `view your photos` 和 `update your photos`。
9. 选择 Save changes（保存更改）。

   您定义的每个自定义范围都将显示在 App client settings（应用程序客户端设置）选项卡的 OAuth2.0 Allowed Custom Scopes（OAuth2.0 允许的自定义范围）下面；例如 `com.example.photos/read`。

使用身份提供者

使用 AWS Management Console配置用户群体社交 IdP

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 Manage User Pools（管理用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。

4. 在左侧导航栏上，选择 Identity providers（身份提供商）。
5. 选择一个社交 IdP：Facebook、Google（谷歌）、Login with Amazon（使用亚马逊账号登录）或 Apple（苹果）。
6. 根据您选择的社交 IdP，从以下步骤中进行选择：

   - Google 和 Login with Amazon – 输入在上一部分中生成的 app client ID（应用程序客户端 ID）和 app client secret（应用程序客户端密钥）。
   - Facebook – 输入在上一部分中生成的 app client ID（应用程序客户端 ID）和 app client secret（应用程序客户端密钥），然后选择 API 版本（例如，版本 2.12）。我们建议选择最新的可用版本，因为每个 Facebook API 版本都有一个生命周期和一个弃用日期。Facebook 的范围和属性可能因 API 版本而异。我们建议您使用 Facebook 测试您的社交身份登录，以确保联合身份认证会按预期运行。
   - Sign In with Apple – 输入在上一部分中生成的 Services ID（服务 ID）、Team ID（团队 ID）、Key ID（密钥 ID）和 private key（私有密钥）。
7. 输入要授权的范围的名称。范围定义了您要通过应用程序访问的用户属性（如 `name` 和 `email`）。对于 Facebook，这些属性应用逗号分隔。对于 Google 和 Login with Amazon，则应采用空格分隔。对于 Sign in with Apple，选中要访问的范围的复选框。

社交身份提供商	示例范围
Facebook	`public_profile, email`
Google	`profile email openid`
Login with Amazon	`profile postal_code`
Sign in with Apple	`email name`

   您的应用程序用户需要同意向您的应用程序提供这些属性。关于范围的更多信息，请参阅 Google、Facebook 和 Login with Amazon 或 Sign in with Apple 的文档。

   对于 Sign in with Apple，下面提供了可能不会返回范围的用户场景：

   - 终端用户离开 Apple 登录页面后出现故障（可能来自 Amazon Cognito 内部的故障或开发人员编写的任何内容）
   - 跨用户池和其它身份验证服务使用服务 ID 标识符
   - 在最终用户登录之前，开发人员添加了其他范围（未检索到新信息）
   - 开发人员删除用户，然后用户再次登录，而没有从其 Apple ID 个人资料中删除该应用程序
8. 为您要配置的社交 IdP 选择 Enable（启用）。
9. 从导航栏中选择 App client settings（应用程序客户端设置）。
10. 选择社交 IdP 作为用户池应用程序的 Enabled Identity Providers（启用的身份提供商）之一。
11. 在用户池应用程序的 Callback URL(s)（回调 URL）中输入回调 URL。这是您的用户在身份验证成功后将被重新导向的 URL。

    ```
    https://www.example.com
    ```
12. 选择 Save changes（保存更改）。
13. 在 Attribute mapping（属性映射）选项卡上，至少为必需属性添加映射，通常是 `email`，如下所示：

    a. 选中此复选框以选择 Facebook、Google 或 Amazon 属性名称。您还可以键入未在 Amazon Cognito 控制台中列出的其他属性的名称。
    b. 从下拉列表中选择目标用户池属性。
    c. 选择 Save changes（保存更改）。

d. 选择 Go to summary（转到摘要页面）。

将用户登录与社交 IdP 集成

1. 选择一个社交身份提供商，例如 Facebook、Google、Login with Amazon 或 Sign In with Apple。
2. 根据您选择的 IdP，完成以下步骤之一，从而输入社交身份提供商的信息：

   Facebook、Google 和 Login with Amazon

   输入您创建客户端应用程序时收到的应用程序密钥。

   Sign In with Apple

   输入您向 Apple 提供的服务 ID，以及创建应用程序客户端时收到的团队 ID、密钥 ID 和私有密钥。
3. 对于 App secret（应用程序密钥），输入您创建客户端应用程序时收到的应用程序密钥。
4. 对于 Authorize scopes（授权范围），输入要映射到用户池属性的社交身份提供商范围的名称。范围定义了您要通过应用程序访问的用户属性（如名称和电子邮件）。输入范围时，根据您选择的 IdP 使用以下准则：

   - Facebook – 以英文逗号分隔范围。例如：

     ```
     public_profile, email
     ```
   - Google、Login with Amazon 和 Sign In with Apple – 以空格分隔范围。例如：
     - Google: `profile email openid`
     - Login with Amazon: `profile postal_code`
     - Sign In with Apple: `name email`

       Note

       对于 Sign in with Apple（控制台），请使用复选框以选择范围。

   您的用户需要同意向应用程序提供这些属性。关于每个社交身份提供商范围的更多信息，请参阅 Google、Facebook 和 Login with Amazon 或 Sign In with Apple 的文档。
5. 选择 Enable Facebook（启用 Facebook）、Enable Google（启用谷歌）、Enable Login with Amazon（启用使用亚马逊账号登录）或 Enable Sign in with Apple（启用使用苹果账号登录）。

在您的用户群体中配置 SAML 2.0 IdP

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 Manage User Pools（管理用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 在左侧导航栏上，选择 Identity providers（身份提供商）。
5. 选择 SAML 以打开 SAML 对话框。
6. 在 Metadata document（元数据文档）下，通过 SAML IdP 上载元数据文档。还可以输入指向该元数据文档的 URL。有关更多信息，请参阅将第三方 SAML 身份提供商与 Amazon Cognito 用户池集成 (p. 99)。

   Note

   建议您提供端点 URL（如果它是一个公有端点），而不是上传文件，因为这将允许 Amazon Cognito 自动刷新元数据。通常，元数据刷新操作每 6 小时执行一次或在元数据过期前执行（以时间较早者为准）。
7. 输入 SAML Provider name（提供商名称）。有关 SAML 命名的更多信息，请参阅选择 SAML 身份提供商名称 (p. 95)。

8. 输入要使用的任何可选 SAML Identifiers（标识符）。
9. 如果您希望用户在用户池注销时，同时从 Amazon Cognito 和 SAML IdP 注销，请选择 Enable IdP sign out flow（启用 IdP 注销流程）。

   如果启用此流程，在调用 Logout-endpoint 时，将会向 SAML IdP 发送已签名的注销请求。

   配置此端点以处理来自 IdP 的注销响应。此终端节点使用 POST 绑定。

   ```
   https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/logout
   ```

   Note

   如果选择此选项，并且您的 SAML IdP 需要已签名的注销请求，则您还需要为您的 SAML IdP 配置 Amazon Cognito 提供的签名证书。
   SAML IdP 将处理已签名的注销请求并从 Amazon Cognito 会话中注销您的用户。
10. 选择 Create provider（创建提供商）。
11. 在 Attribute mapping（属性映射）选项卡上，至少为必需属性添加映射，通常是 email，如下所示：

    a. 按照您的 IdP 提供的 SAML 断言中的显示，输入 SAML 属性名称。您的 IdP 可能会提供示例 SAML 断言以供参考。一些 IdP 使用简单名称（如 email），另一些则使用类似于下面 URL 格式的属性名称：

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

    b. 从下拉列表中选择目标用户池属性。
12. 选择 Save changes（保存更改）。
13. 选择 Go to summary（转到摘要页面）。

删除 SAML 提供者

1. 登录 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证
2. 在导航窗格中，选择 Manage your User Pools（管理您的用户池），然后选择要编辑的用户池。
3. 从 Federation（联合身份验证）控制台页面中选择 Identity providers（身份提供商）。
4. 选择 SAML 以显示 SAML 身份提供商。
5. 选中要删除的提供商旁边的复选框。
6. 选择 Delete provider（删除提供商）。

为您的用户群体指定 SAML 身份提供者属性映射

1. 选择 SAML 选项卡。
2. 对于需要映射的每个属性，请完成以下步骤：

   a. 选择 Add SAML attribute（添加 SAML 属性）。
   b. 在 SAML attribute（SAML 属性）字段中，输入要映射的 SAML 属性的名称。
   c. 对于 User pool attribute（用户池属性），从下拉列表中选择要映射到 SAML 属性的用户池属性。
   d. 选择 Save changes（保存更改）。

添加 OIDC IdP

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. Manage User Pools（管理用户池）。

3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 在左侧导航栏上，选择 Identity providers（身份提供商）。
5. 选择 OpenID Connect。
6. 在 Provider name（提供商名称）中输入一个唯一名称。
7. 在 Client ID（客户端 ID）中输入上一节中的 OIDC IdP 的客户端 ID。
8. 在 Client secret（客户端密钥）中输入上一节中的客户端密钥。
9. 在下拉列表中，选择用于将 userinfo 端点中的用户的详细信息提取到 Attributes request method（属性请求方法）的 HTTP 方法（GET 或 POST）。
10. 输入要授权的范围的名称。范围定义了您要通过应用程序访问的用户属性（如 `name` 和 `email`）。根据 OAuth 2.0 规范，范围由空格分隔。

    您的应用程序用户需要同意向您的应用程序提供这些属性。
11. 输入 IdP 的 URL，然后选择 Run discovery（运行发现）。

    Note

    URL 应该以 `https://` 开头，并且不应以下斜杠 `/` 结尾。只有端口号 443 和 80 可用于此 URL。例如，Salesforce 使用以下 URL：`https://login.salesforce.com`

    - 如果 Run discovery（运行发现）不成功，则需要提供 Authorization endpoint（授权终端节点）、Token endpoint（令牌终端节点）、Userinfo endpoint（用户信息终端节点）和 Jwks uri（JSON Web Key 的位置）。

      Note

      如果提供商使用发现进行联合登录，则发现文档必须对以下值使用 HTTPS：`authorization_endpoint`、`token_endpoint`、`userinfo_endpoint` 和 `jwks_uri`。否则，登录将失败。
12. 选择 Create provider（创建提供商）。
13. 在左侧导航栏上，选择 App client settings（应用程序客户端设置）。
14. 选择在上一步中设置的 OIDC 提供商作为 Enabled Identity Providers（启用身份提供商）之一。
15. 输入回调 URL，供 Amazon Cognito 授权服务器在用户通过身份验证之后进行调用。这是页面的 URL，在页面中，您的用户在身份验证成功后将被重定向。

    ```
    https://www.example.com
    ```
16. 在 Allowed OAuth Flows（允许的 OAuth 流程）下，启用 Authorization code grant（授权代码授予）和 Implicit code grant（隐式代码授予）。

    除非您希望明确地排除一个允许的 OAuth 范围，否则请选中所有 Allowed OAuth scopes（允许的 OAuth 范围）的复选框。
17. 选择 Save changes（保存更改）。
18. 在左侧导航栏的 Attribute mapping（属性映射）选项卡上，将 OIDC 声明的映射添加到用户池属性。

    a. 作为默认值，OIDC 声明 sub 将映射到用户池属性 Username（用户名）。您可以将其他 OIDC 声明映射到用户池属性。输入 OIDC 声明，然后从下拉列表中选择对应的用户池属性。例如，声明 email 通常会映射到用户池属性 Email（电子邮件）。

    b. 在下拉列表中，选择目标用户池属性。

    c. 选择 Save changes（保存更改）。

    d. 选择 Go to summary（转到摘要页面）。

社交提供者属性映射

Note

只有在您编辑现有用户池时才会显示 Attribute mapping（属性映射）选项卡。

在属性映射选项卡上，您可以将身份提供商 (IdP) 属性或断言映射到用户池属性。有关更多信息，请参阅指定适用于用户池的身份提供商属性映射 (p. 107)。

Note

目前，仅 Facebook `id`、Google `sub`、Login with Amazon `user_id` 和 Sign in with Apple `sub` 属性可映射到 Amazon Cognito 用户池 `username` 属性。

Note

用户池中的属性必须足够大，以包含映射的身份提供商属性的值，否则用户登录时将发生错误。如果将自定义属性映射到身份提供商令牌，则应将其大小设置为最大 2048 个字符。
您必须为您的用户池所需的任何属性创建映射。

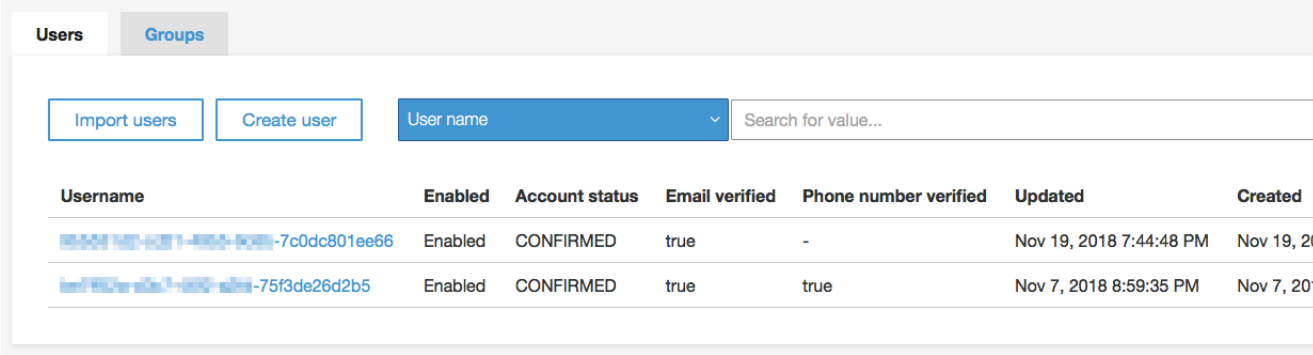
指定社交 IdP 属性映射

1. 登录 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证
2. 在导航窗格中，选择 Manage your User Pools（管理您的用户池），然后选择要编辑的用户池。
3. 选择 Attribute mapping（属性映射）选项卡。
4. 选择 Facebook、Google、Amazon 或 Apple 选项卡。
5. 对于需要映射的每个属性，请完成以下步骤：
   a. 选中 Capture（截取）复选框。
   b. 对于 User pool attribute（用户池属性），在下拉列表中，选择要映射到社交 IdP 属性的用户池属性。
   c. 如果您需要更多属性，请选择 Add Facebook attribute（添加 Facebook 属性）[或者 Add Google attribute（添加 Google 属性）、Add Amazon attribute（添加 Amazon 属性）或 Add Apple attribute（添加 Apple 属性）] 并完成以下步骤：
      i. 在 Facebook attribute（Facebook 属性）、Google attribute（Google 属性）、Amazon attribute（Amazon 属性）或 Apple attribute（Apple 属性）字段中，输入要映射的属性的名称。
      ii. 在 User pool attribute（用户池属性）下拉列表中，选择要将社交 IdP 属性映射到的用户池属性。
   d. 选择 Save changes（保存更改）。

使用用户

创建用户

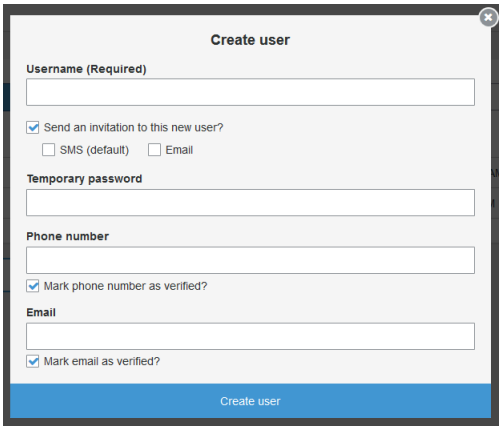
Users and groups（用户和组）选项卡中的 Users（用户）选项卡具有一个 Create user（创建用户）按钮。



选择 Create user（创建用户）后，系统将会显示一个 Create user（创建用户）对话框，在此对话框中可以输入关于新用户的信息。仅 Username（用户名）字段为必填字段。

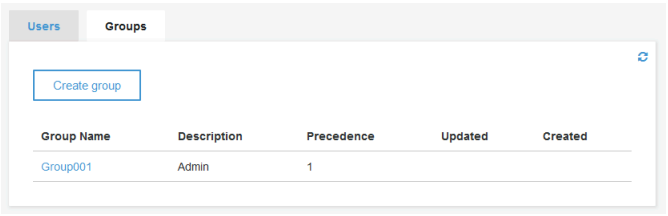
Note

对于您在 中使用 Create userAWS Management Console（创建用户）表单创建的用户账户，只能在 AWS Management Console中设置表单中显示的属性。其他属性必须使用 AWS Command Line Interface 或 Amazon Cognito API 进行设置，即使您已将它们标记为必需属性。

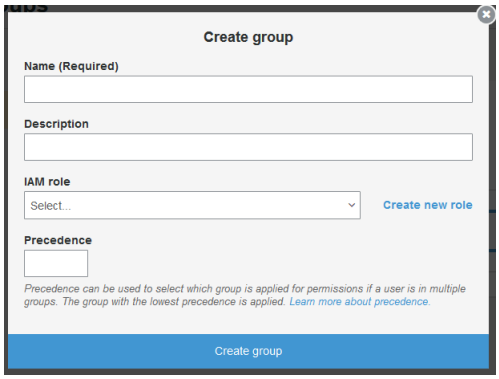


创建组

Users and groups 选项卡中的 Groups 选项卡具有一个 Create group 按钮。



在选择 Create group (创建组) 时，将显示一个 Create group (创建组) 表单。您将在此表格中输入组的信息。仅 Name(名称) 字段为必填字段。如果您要将用户池与身份池集成，并且身份池配置为从令牌中选择角色，则 IAM role (IAM 角色) 设置将确定在用户的 ID 令牌中分配哪个角色。如果您尚未定义角色，请选择 Create new role (创建新角色)。如果您有多个组，并且可将您的用户分配到多个组，则可以为每个组设置 Precedence (优先级) 值。优先级值可以是任何非负整数。零是代表最高优先级的值。



查看用户属性

您可以在 AWS Management Console中执行多项操作：

- 您可以查看 Pool details（池详细信息），并编辑用户池属性、密码策略、MFA 设置、应用程序和触发器。有关更多信息，请参阅用户池参考（AWS Management Console）(p. 243)。
- 您可以查看用户池中的用户，并进一步了解更多详细信息。
- 您还可以查看关于用户池中个别用户的详细信息。
- 您也可以搜索用户池中的用户。

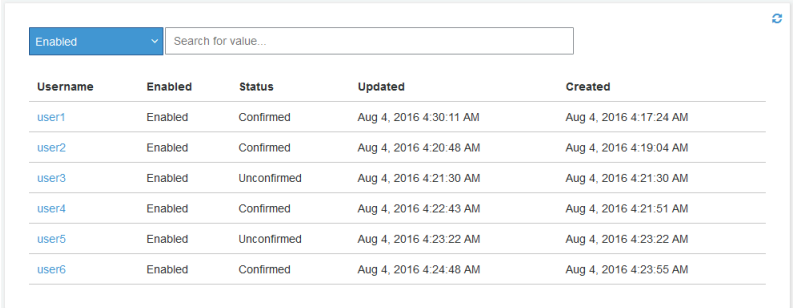
查看用户属性

1. 在 AWS Management Console 中的 Amazon Cognito 主页上，选择 Manage your user identities（管理您的用户身份）。
2. 从 Your User Pools（您的用户池）页中选择用户池。
3. 选择 User and Groups（用户和组）以查看用户信息。
4. 选择用户名称以显示关于单个用户的更多信息。在此屏幕中，您可以执行以下任一操作：
   - Add user to group（将用户添加到组）
   - Reset user password（重置用户密码）
   - Confirm user（确认用户）
   - Enable or disable MFA（启用或禁用 MFA）
   - Delete user（删除用户）

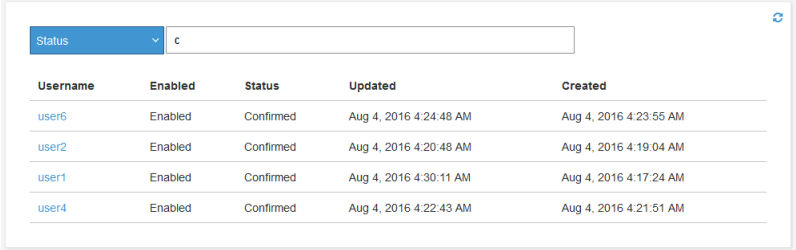
搜索用户

以下所有示例均使用同一个用户池。

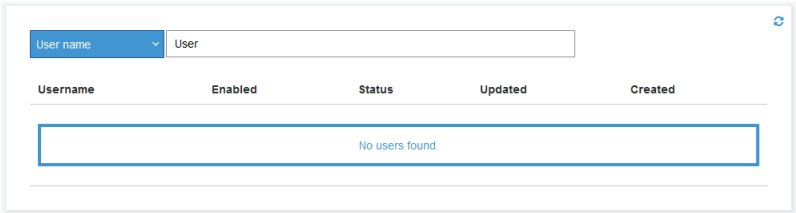
例如，如果您要列出所有用户，则将搜索框留空。



如果您要搜索所有已确认的用户，则从下拉菜单中选择Status（状态）。在搜索框中，输入单词“confirmed”的第一个字母。



请注意，某些属性值区分大小写，如 User name（用户名）。



使用控制台添加用户群体 Lambda 触发器

1. 使用 Lambda 控制台创建 Lambda 函数。有关 Lambda 函数的更多信息，请参阅https://docs.aws.amazon.com/lambda/latest/dg/《AWS Lambda 开发人员指南》。
2. 转到 Amazon Cognito 控制台。选择 Manage User Pools（管理用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 在您的用户池中，从导航栏中选择 Triggers（触发器）选项卡。
5. 选择 Lambda 触发器，例如注册前或身份验证前。然后，从 Lambda function（Lambda 函数）下拉列表中选择 Lambda 函数。
6. 选择 Save changes（保存更改）。
7. 您可以在 Lambda 控制台中使用 Amazon CloudWatch 记录您的 Lambda 函数。有关更多信息，请参阅访问适用于 Lambda 的 Amazon CloudWatch Logs。

重置用户的密码

1. 在 AWS Management Console 中的 Amazon Cognito 主页上，选择 Manage your user identities（管理您的用户身份）。
2. 从 Your User Pools（您的用户池）页中选择用户池。
3. 选择User and Groups （用户和组）以查看用户信息。
4. 在列表中选择要为其重置密码的用户。

   Reset user password（重置用户密码）操作会立即向用户发送确认代码，并通过将用户状态更改为 `RESET_REQUIRED` 来禁用用户的当前密码。Enable MFA（启用 MFA）操作会导致在用户尝试登录时立即向其发送确认代码。Reset user password（重置用户密码）代码的有效期为 1 小时。MFA 代码的有效期为 3 分钟。

下载用户导入 CSV 文件标头

1. 导航到 Amazon Cognito 控制台，选择 Manage User Pools (管理用户池)，然后选择要向其中导入用户的用户池。
2. 选择 Users（用户）选项卡。
3. 选择 Import users（导入用户）。

4. 选择 Download CSV header（下载 CSV 标头）以获取包含标头行（您必须在 CSV 文件中添加这一内容）的 CSV 文件。

从 CSV 文件导入用户

1. 选择创建导入任务。
2. 输入 Job name（任务名称）。任务名称可以包含大写和小写字母 (a-z、A-Z)、数字 (0-9) 和以下特殊字符：+ = , . @ -。
3. 如果这是您首次创建用户导入任务，则AWS Management Console将自动为您创建 IAM 角色。否则，您可以从 IAM Role (IAM 角色) 列表中选择现有角色，也可以让AWS Management Console为您创建新角色。
4. 选择 Upload CSV（上传 CSV）并选择要从中导入用户的 CSV 文件。
5. 请选择 Create job (创建任务)。
6. 要启动任务，请选择开始。

查看必需属性

1. 转到 AWS Management Console中的 Amazon Cognito。如果控制台提示您，请输入 AWS 凭证。
2. 选择 Manage User Pools（管理用户池）。
3. 在 Your User Pools（您的用户池）页面上，选择一个用户池。
4. 在左侧的导航菜单中，选择 Attributes（属性）。
5. 在 Which standard attributes are required?（哪些标准属性为必需属性？）下，查看用户池的必需属性。

配置用户池使用别名登录

1. 转到 AWS Management Console中的 Amazon Cognito。如果控制台提示您，请输入 AWS 凭证。
2. 选择 Manage User Pools（管理用户池）。
3. 在页面右上角，选择 Create a user pool（创建用户池）。
4. 在页面左上角，选择 Review（查看）。
5. 在 Attributes（属性）选项卡中，在 How do you want your end users to sign-in?（您希望您的最终用户如何注册和登录？）中选择 Username（用户名）。然后选择以下任一选项：

   - Also allow sign in with verified email address（还允许使用经验证的电子邮件地址登录）：此选项允许用户使用其电子邮件地址登录。
   - Also allow sign in with verified phone number（还允许使用经验证的电话号码登录）：此选项允许用户使用其电话号码登录。
   - Also allow sign in with preferred username（还允许使用首选用户名登录）：此选项允许用户使用首选的用户名登录。用户可以更改此用户名。
6. 选择 Next step（下一步）保存，然后完成向导中的所有步骤。

添加自定义属性

1. 转到 AWS Management Console中的 Amazon Cognito。如果控制台提示您，请输入 AWS 凭证。
2. 选择 Manage User Pools（管理用户池）。
3. 在 Your User Pools（您的用户池）页面上，选择要配置的用户池。
4. 在左侧的导航栏中，选择 Attributes（属性）。
5. 在 Do you want to add custom attributes?（是否要添加自定义属性？）下，选择 Add custom attribute（添加自定义属性）。

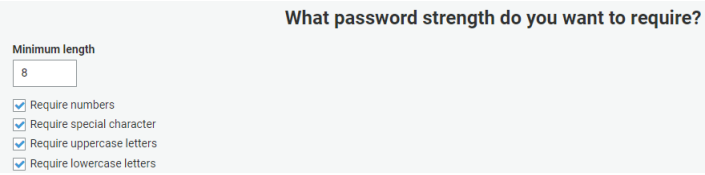
6. 提供有关新属性的以下详细信息：
   - 输入 Name（名称）。
   - 选择 Type（类型）（字符串或数字）。
   - 输入最小字符串长度或数字值。
   - 输入最大字符串长度或数字值。
   - 如果您想授予用户在设置初始值后更改自定义属性值的权限，请选择 Mutable（可变）。
7. 选择 Save changes（保存更改）。

消息和验证

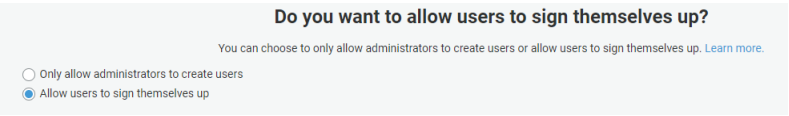
设置密码策略并启用自行注册

Policies（策略）选项卡包含以下相关设置：

- 指定所需的密码强度。



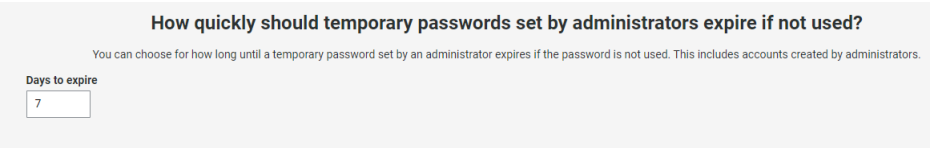
- 指定是否允许用户自己注册。默认情况下，设定此选项。



- 为新账户指定用户账户到期时间限制（以天为单位）。从用户账户的创建时间开始计算，默认设置为 7 天。最大设置为 90 天。账户到期后，用户将无法登录账户，直至管理员更新用户配置文件。

  Note

  用户登录后，账户将永不过期。

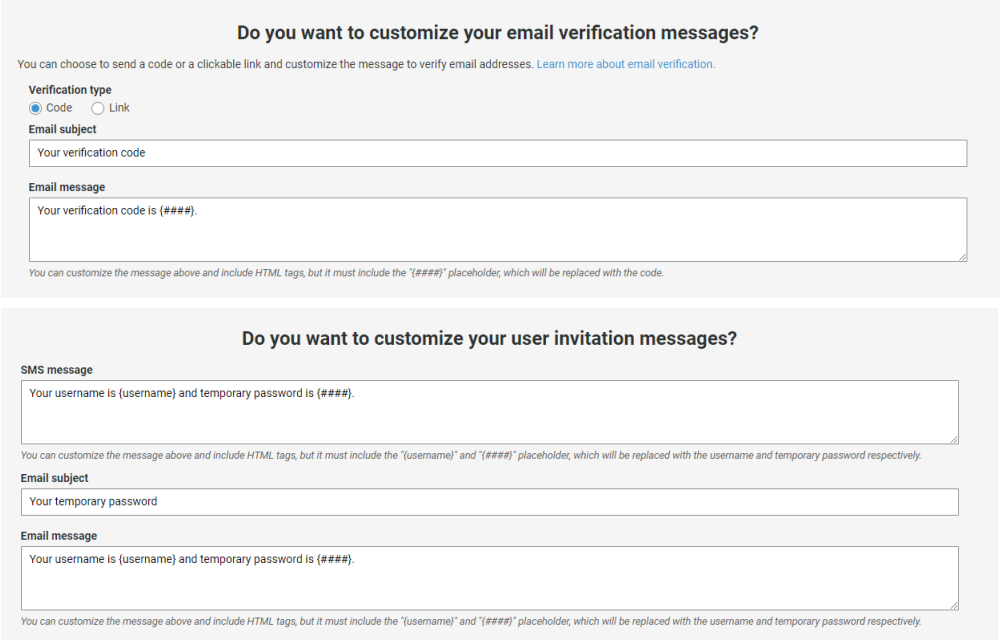


自定义消息

Message Customizations（消息自定义）选项卡包含用于指定自定义电子邮件验证和用户邀请消息的模板。

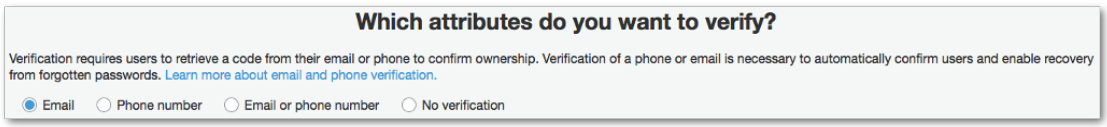
对于电子邮件验证消息或用户邀请消息，消息的最大长度为 2048 个 UTF-8 字符，包括验证代码或临时密码。对于 SMS 验证消息或用户邀请消息，消息的最大长度为 140 个 UTF-8 字符，包括验证代码或临时密码。

验证代码的有效期为 24 小时。



配置用户验证

1. 登录AWS Management Console并打开 Amazon Cognito 控制台 (https://console.aws.amazon.com/cognito)。如果出现提示，请输入 AWS 凭证。
2. 选择 Manage User Pools（管理用户池）。
3. 在 Your User Pools（您的用户池）页面上，选择要配置的用户池。
4. 在左侧导航菜单中，选择 MFA and verifications（MFA 和验证）。
5. 在 Which attributes do you want to verify?（您要验证哪些属性？）下，选择以下选项之一：



Email（电子邮件）

如果您选择此选项，Amazon Cognito 将在用户注册时通过电子邮件发送验证码。如果您通常通过电子邮件与用户通信，请选择此选项。例如，如果您要发送账单、订单摘要或特别优惠，则将需要使用经过验证的电子邮件地址。

Phone number（电话号码）

如果您选择此选项，Amazon Cognito 将在用户注册时通过 SMS 消息发送验证码。如果您通常通过 SMS 消息与用户通信，请选择此选项。例如，如果您要发送交付通知、约会确认或提醒，则将需要使用经过验证的电话号码。

Email or phone number（电子邮件或电话号码）

如果您不要求所有用户都拥有相同的经验证的联系方式，请选择此选项。在这种情况下，您的应用程序中的注册页面可能会要求用户仅验证其首选联系方式。当 Amazon Cognito 发送验证码时，会将该代码发送到来自您应用程序的 `SignUp` 请求中提供的联系方式。如果用户同时提供了电子邮件地址和电话号码，并且您的应用程序在 `SignUp` 请求中提供了这两种联系方式，Amazon Cognito 将仅向电话号码发送验证码。

如果您要求用户同时验证电子邮件地址和电话号码，则选择此选项。Amazon Cognito 将在用户注册时验证一种联系方式，而且您的应用程序必须在用户登录后验证另一种联系方式。有关更多信息，请参阅在您要求用户确认电子邮件地址和电话号码的情况下 (p. 171)。

None（无）

如果您选择此选项，则 Amazon Cognito 不会在用户注册时发送验证码。如果您要使用自定义身份验证流程验证至少一种联系方式，而不使用来自 Amazon Cognito 的验证码，请选择此选项。例如，您可以使用一个预注册 Lambda 触发器，该触发器将自动验证属于特定域的电子邮件地址。

如果您不验证用户的联系人信息，用户可能无法使用您的应用程序。请记住，用户需要经验证的联系人信息才能：

- Reset their passwords（重置密码）– 当用户在您的应用程序中选择调用 `ForgotPassword` API 操作的选项时，Amazon Cognito 会将临时密码发送到用户的电子邮件地址或电话号码。Amazon Cognito 仅在用户具有至少一个经验证的联系方式时发送此密码。
- Sign in by using an email address or phone number as an alias（通过使用电子邮箱地址或电话号码作为别名进行登录）– 如果您将用户池配置为允许这些别名，则用户只能在别名经过验证后使用别名进行登录。有关更多信息，请参阅自定义登录属性 (p. 246)。

6. 选择保存更改。

配置用户池以使用自定义电子邮件地址

1. 在 https://console.aws.amazon.com/cognito 打开 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 Manage User Pools（管理用户池）。
3. 在 Your User Pools（您的用户池）页面上，选择要配置的用户池。
4. 在左侧导航菜单中，选择 Message customizations（管理自定义）。
5. 如果您希望使用自定义 FROM 地址，选择 Add custom FROM address (添加自定义 FROM 地址) 并完成以下步骤：

   a. 对于 SES region (SES 区域)，选择包含验证的电子邮件地址的区域。

   b. 对于 Source ARN（源 ARN），请选择您的电子邮件地址。使用您已在所选区域使用 Amazon SES 进行验证的电子邮件地址。

   c. 对于 FROM email address（FROM 电子邮件地址），选择您的电子邮件地址。您可以仅提供电子邮件地址，也可以同时提供电子邮件地址和格式为 `Jane Doe <janedoe@example.com>` 的易记名称。

6. 在 Do you want to send emails through your Amazon SES Configuration?（是否要通过 Amazon SES 配置发送电子邮件？）下，选择 Yes - Use Amazon SES（是 - 使用 Amazon SES）或 No - Use Cognito (Default)（否 -使用 Cognito (默认)）。

   如果您选择使用 Amazon SES，Amazon Cognito 将在您保存更改后创建服务相关角色。

7. 如果您希望使用自定义 REPLY-TO 地址，请选择 Add custom REPLY-TO address（添加自定义 REPLY-TO 地址）。然后，输入电子邮件地址，您将使用该地址接收用户发送到您的 FROM 地址的邮件。
8. 当您完成设置您的电子邮件账户选项时，选择 Save changes（保存更改）。

确认您位于 SMS 沙盒中

1. 在 https://console.aws.amazon.com/cognito 打开 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 创建新用户池 (p. 30)或者编辑现有用户池 (p. 169)。
3. 如果您的账户位于 SMS 沙盒中，您将在 Amazon Cognito 中看到以下消息：

```
You are currently in a Sandbox environment in Amazon SNS.
```

如果您没有看到此消息，则表明有人已经在您的账户中设置了 SMS 消息。跳至在 Amazon Cognito 中完成用户群体设置 (p. 203)。

4. 在消息中，要在新选项卡中打开 Amazon SNS 控制台，请选择 Amazon SNS 链接。
5. 验证您是否位于沙盒环境中。控制台消息指示您的沙盒状态和 AWS 区域，如下所示：

```
This account is in the SMS sandbox in US East (N. Virginia).
```

配置用户池使用电子邮件地址或电话号码注册和登录

1. 转到 AWS Management Console中的 Amazon Cognito。如果控制台提示您，请输入 AWS 凭证。
2. 选择 Manage User Pools（管理用户池）。
3. 在页面右上角，选择 Create a user pool（创建用户池）。
4. 在页面左上角，选择 Review（查看）。
5. 在 Attributes（属性）选项卡中的 How do you want your end users to sign-in?（您希望您的最终用户如何登录？）下，选择 Email address or phone number（电子邮件地址或电话号码）。然后选择以下任一选项：
   - Allow email addresses（允许使用电子邮件地址）：允许用户使用电子邮件作为用户名注册。
   - Allow phone numbers（允许使用电话号码）：允许用户使用电话号码作为用户名注册。
   - Allow both email addresses and phone number (users can choose one)（允许同时使用电子邮件地址和电话号码（用户可选择其中一个））：允许用户在注册过程中使用电子邮件地址或电话号码作为用户名。
6. 选择 Next step（下一步）保存，然后完成向导中的所有步骤。

自定义 FROM 和 REPLY-TO 电子邮件地址

1. 导航到 Amazon Cognito 控制台，选择用户池。
2. 选择 Message customizations（消息自定义）。
3. 在 Message customizations（消息自定义）中，选择 SES Region（SES 区域）。
4. 在 FROM email address ARN（FROM 电子邮件地址 ARN）字段中，从列表中选择已验证电子邮件地址。选择 FROM email address ARN（FROM 电子邮件地址 ARN）字段下的验证 SES 身份链接，验证 Amazon Simple Email Service 身份。有关更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的在 Amazon SES 中验证电子邮件地址和域。
5. 要自定义 REPLY-TO 电子邮件地址，请在 REPLY-TO email address（REPLY-TO 电子邮件地址）字段中输入有效的电子邮件地址。

使用安全功能

在 Amazon Cognito 控制台中配置 MFA

1. 从左侧导航栏中，选择 MFA and verifications（MFA 和验证）。
2. 选择 MFA 处于 Off（关闭）、Optional（可选）还是Required（必需）状态。
3. 在按用户激活 MFA 时，或者在您使用基于风险的自适应身份验证时，选择 Optional（可选）。有关自适应身份验证的更多信息，请参阅向用户池添加高级安全 (p. 227)。
4. 选择要在应用程序中支持的第二安全要素。用户可以使用 SMS text message（短信）或 Time-based One-time Password（基于时间的一次性密码）作为第二安全要素。我们建议使用 TOTP，这样您可以使用 SMS 作为密码恢复机制，而不是作为身份验证安全要素。

5. 如果您使用 SMS 文本消息作为第二安全要素并且未定义具有此权限的 IAM 角色，则可在控制台中创建一个此角色。选择 Create role（创建角色）以创建允许 Amazon Cognito 代表您向用户发送 SMS 消息的 IAM 角色。有关更多信息，请参阅 IAM 角色。
6. 选择 Save changes（保存更改）。

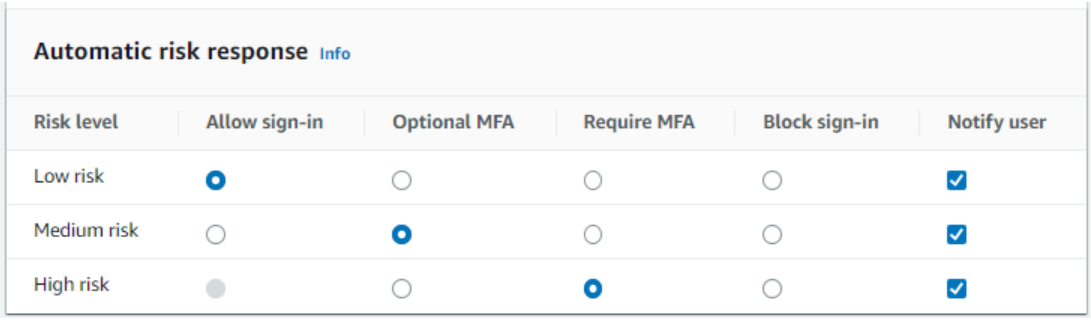
为用户池配置高级安全功能

1. 从左侧导航栏中，选择 Advanced security（高级安全功能）。
2. 对于 Do you want to enable advanced security features for this user pool?（是否要为此用户池启用高级安全功能？），选择 Yes（是）以启用高级安全功能。或者，选择 Audit only（仅审核）来收集信息并将用户池数据发送到 CloudWatch。

   建议在启用操作之前，先将高级安全功能保持在审核模式下两周。在此期间，Amazon Cognito 可以了解应用程序用户的使用模式。
3. 从下拉列表中，选择 What app client do you want to customize settings for?（您希望自定义哪个应用程序客户端的设置？）默认情况下，对于所有应用程序客户端将设置保留为全局。
4. 对于 Which action do you want to take with the compromised credentials?（对于对遭盗用的凭证，您希望采取什么操作？），选择 Allow（允许）或 Block use（阻止使用）。
5. 选择 Customize when compromised credentials are blocked（自定义何时阻止遭盗用的凭证），以选择哪些事件应发起遭盗用的凭证检查：

   - 登录
   - 注册
   - 密码更改
6. 在 How do you want to use adaptive authentication for sign-in attempts rated as low, medium and high risk?（当登录尝试被评为低、中和高风险时，您希望如何使用自适应身份验证？）下选择如何响应恶意登录尝试。可以允许或阻止登录尝试，或者在允许登录前要求额外的质询。

   要在检测到异常登录尝试时发送电子邮件通知，请选择 Notify users（通知用户）。



7. 如果在上一步中选择了 Notify users（通知用户），则可通过使用 Notification message customization（通知消息自定义）表单自定义电子邮件通知消息。
8. 选择 Customize（自定义）以自定义包含 HTML 和纯文本版本的电子邮件消息的自适应身份验证通知。要了解有关电子邮件消息模板的更多信息，请参阅消息模板 (p. 254)。
9. 键入不管高级安全风险评估如何都要 Always allow（始终允许）或 Always block（始终阻止）的任何 IP 地址。用 CIDR 表示法指定 IP 地址范围（例如，192.168.100.0/24）。
10. 选择 Save changes（保存更改）。

Amazon Cognito 身份池（联合身份）

Amazon Cognito 身份池是联合身份的目录，您可以用它交换 AWS 凭证。身份池会为应用程序的用户生成临时 AWS 凭证，无论他们是已登录还是您尚未识别他们。使用 AWS Identity and Access Management（IAM）角色和策略，您可以选择要向用户授予的权限级别。用户能够以访客身份开始，然后检索您保留在 AWS 服务中的资产。然后，他们可以通过第三方身份提供者登录，以解锁对您提供给注册会员的资产的访问权限。第三方身份提供者可以是使用者（社交）OAuth 2.0 提供者（如 Apple 或 Google）、自定义 SAML 或 OIDC 身份提供者，也可以是您自己的设计的自定义身份验证方案，也称为开发人员提供者。

2023 年 4 月 19 日，Amazon Cognito 推出了新的身份池 AWS Management Console体验。目前，您可以在原始身份池控制台和新设计之间进行选择。在本指南中，当您在控制台中查看身份池操作的说明时，可以在原始控制台体验和新控制台体验的说明之间切换。

Amazon Cognito 身份池的功能

签署对 AWS 服务的请求

签署对 AWS 服务 [如 Amazon Simple Storage Service（Amazon S3）和 Amazon DynamoDB] 的 API 请求。使用如 Amazon Pinpoint 和 Amazon CloudWatch 等服务分析用户活动。

使用基于资源的策略筛选请求

对于用户对资源的访问权限进行精细控制。将用户声明转换为 IAM 会话标签，并构建 IAM policy 以向用户的不同子集授予资源访问权限。

分配访客访问权限

对于尚未登录的用户，请将您的身份池配置为生成具有较窄访问权限范围的 AWS 凭证。通过单点登录提供者对用户进行身份验证以提升其访问权限。

根据用户特征分配 IAM 角色

为所有经身份验证的用户分配一个 IAM 角色，或者根据每个用户的声明选择角色。

接受各种身份提供者

将 OIDC 身份或访问令牌、用户群体令牌、SAML 断言或社交提供者 OAuth 令牌交换为 AWS 凭证。

验证您自己的身份

自行进行用户验证，并使用您的开发人员 AWS 凭证为您的用户颁发凭证。

您可能已经有一个 Amazon Cognito 用户群体，用于为您的应用程序提供身份验证和授权服务。您可以将用户群体设置为身份池的身份提供者（IdP）。当您这样做时，用户可以通过用户群体 IdP 进行身份验证，将其声明合并为通用 OIDC 身份令牌，然后将该令牌交换为 AWS 凭证。然后，您的用户可以在签名的请求中向 AWS 服务提供其凭证。

还可以将来自任何身份提供者的经身份验证的声明直接提供给身份池。Amazon Cognito 将来自 SAML、OAuth 和 OIDC 提供者的用户声明自定义为 AssumeRoleWithWebIdentity API 请求以获取短期凭证。

Amazon Cognito 用户群体类似于支持 SSO 的应用程序的 OIDC 身份提供者。对于具有最适合 IAM 授权的资源依赖关系的任何应用程序，身份池充当其 AWS 身份提供者。

Amazon Cognito 身份池支持以下身份提供商：

- 公有提供者：Login with Amazon（身份池）(p. 336)、Facebook（身份池）(p. 330)、Google（身份池）(p. 339)、Sign in with Apple（身份池）(p. 348)、Twitter。
- Amazon Cognito 用户池 (p. 27)

- OpenID Connect 提供程序（身份池）(p. 352)
- SAML 身份提供商（身份池）(p. 355)
- 经开发人员验证的身份（身份池）(p. 357)

有关 Amazon Cognito 身份池区域可用性的信息，请参阅AWS服务区域可用性。

有关 Amazon Cognito 身份池的更多信息，请参阅以下主题。

主题

- Amazon Cognito 身份池入门（联合身份）(p. 289)
- 使用身份池（联合身份）(p. 291)
- 身份池概念（联合身份） (p. 299)
- 将属性用于访问控制 (p. 313)
- 使用基于角色的访问控制 (p. 318)
- 获取凭证 (p. 322)
- 访问 AWS 服务 (p. 328)
- 身份池（联合身份）外部身份提供商 (p. 330)
- 经开发人员验证的身份（身份池）(p. 357)
- 将未经身份验证的用户切换为经过身份验证的用户（身份池）(p. 368)

Amazon Cognito 身份池入门（联合身份）

借助 Amazon Cognito 身份池，您可以创建唯一身份并为用户分配权限。您的身份池可以包括：

- Amazon Cognito 用户池中的用户
- 通过外部身份提供商（例如 Facebook、Google、Apple 或 OIDC 或 SAML 身份提供商）进行身份验证的用户。
- 通过您自己的现有身份验证流程进行身份验证的用户

借助身份池，您可以获取带有您定义的权限的临时 AWS 凭证，以直接访问其它 AWS 服务 或通过 Amazon API Gateway 访问资源。

主题

- 在 Amazon Cognito 中创建一个身份池 (p. 289)
- 设置 SDK (p. 291)
- 集成身份提供商 (p. 291)
- 获取凭证 (p. 291)

在 Amazon Cognito 中创建一个身份池

您可以通过 Amazon Cognito 控制台创建身份池，或者，您还可以使用 AWS Command Line Interface（CLI）或 Amazon Cognito API。

New console

在控制台中创建新的身份池

1. 登录 Amazon Cognito 控制台并选择身份池。
2. 选择创建身份池。

3. 在配置身份池信任中，选择将您的身份池设置为经过身份验证的访问权限和/或访客访问权限。

   - 如果您选择了经过身份验证的访问权限，请在身份池中选择一个或多个您要设置为经过身份验证的身份来源的身份类型。如果您配置了自定义开发人员提供者，则在创建身份池后无法对其进行修改或删除。

4. 在配置权限中，为身份池中经过身份验证的用户或访客用户选择原定设置 IAM 角色。

   a. 如果您希望 Amazon Cognito 为您创建一个具有基本权限并与您的身份池建立信任关系的新角色，请选择创建新的 IAM 角色。例如，输入 IAM 角色名称以标识您的新角色，例如 `myidentitypool_authenticatedrole`。选择查看策略文档以查看 Amazon Cognito 将分配给新 IAM 角色的权限。

   b. 如果您的 AWS 账户中已经有想要使用的角色，则可以选择使用现有的 IAM 角色。您必须将您的 IAM 角色信任策略配置为包括 `cognito-identity.amazonaws.com`。配置您的角色信任策略，以仅允许 Amazon Cognito 在提供证据证明请求来自您的特定身份池中经过身份验证的用户时，才代入该角色。有关更多信息，请参阅角色信任和权限 (p. 312)。

5. 在连接身份提供者中，输入您在配置身份池信任中选择的身份提供者（IdP）的详细信息。可能会要求您提供 OAuth 应用程序客户端信息、选择 Amazon Cognito 用户群体、选择 IAM IdP 或输入开发人员提供者的自定义标识符。

   a. 为每个 IdP 选择角色设置。您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。使用 Amazon Cognito 用户群体 IdP，还可以选择令牌中包含 preferred_role 声明的角色。有关 `cognito:preferred_role` 声明的更多信息，请参阅将优先级值分配到组 (p. 179)。

      i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。

      ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。

   b. 您可以为每个 IdP 配置访问控制属性。访问控制属性将用户声明映射到 Amazon Cognito 应用于其临时会话的主体标签。您可以构建 IAM policy，以根据应用于用户会话的标签来筛选用户访问权限。

      i. 如果不应用主体标签，请选择非活动。

      ii. 要基于 `sub` 和 `aud` 声明应用主体标签，请选择使用原定设置映射。

      iii. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。

6. 在配置属性中，在身份池名称下输入名称。
7. 在基本（经典）身份验证下，选择是否要激活基本流程。在基本流程处于活动状态的情况下，您可以绕过为 IdP 所做的角色选择，直接调用 AssumeRoleWithWebIdentity。有关更多信息，请参阅身份池（联合身份）身份验证流程 (p. 300)。
8. 如果要将标签应用到身份池，请在标签下选择添加标签。
9. 在查看并创建中，确认您为新身份池所做的选择。选择编辑以返回向导并更改任何设置。完成后，选择创建身份池。

Original console

在控制台中创建新的身份池

1. 登录到 Amazon Cognito 控制台，选择 Manage identity pools（管理身份池），然后选择 Create new identity pool（创建新的身份池）。
2. 为身份池键入一个名称。
3. 要启用未经身份验证的身份，请从 Unauthenticated identities（未经身份验证的身份）的可折叠部分中选择 Enable access to unauthenticated identities（启用访问未经身份验证的身份）。

4. 如果需要，请在身份验证提供商部分中配置身份验证提供商。
5. 选择 Create Pool（创建池）。

   Note

   有效身份池至少需要一个身份。
6. 系统将提示您访问您的AWS资源。

   选择 Allow（允许）以创建两个与您的身份池关联的默认角色，一个用于未经身份验证的用户，另一个用于经过身份验证的用户。这些默认角色会向 Amazon Cognito Sync 提供身份池访问权限。您可以在 AWS Identity and Access Management（IAM）控制台中修改与身份池关联的角色。

设置 SDK

要使用 Amazon Cognito 身份池，请设置 AWS Amplify、AWS SDK for Java 或 AWS SDK for .NET。有关更多信息，请参阅以下主题。

- 《AWS SDK for Java 开发人员指南》中的设置适用于 JavaScript 的 SDK
- 《Amplify 开发中心》中的 Amplify 文档
- 《AWS SDK for .NET 开发人员指南》中的 Amazon Cognito 凭证提供者

集成身份提供商

Amazon Cognito 身份池（联合身份）通过 Amazon Cognito 用户群体、联合身份提供者（包括 Amazon、Facebook、Google、Apple 和 SAML 身份提供者）以及未经身份验证的身份支持用户身份验证。此功能还支持 经开发人员验证的身份（身份池）(p. 357)，这让您能够通过自己的后端身份验证流程注册并对用户进行身份验证。

要了解有关使用 Amazon Cognito 用户池创建自己的用户目录的更多信息，请参阅Amazon Cognito 用户池 (p. 27)和在登录后使用身份池访问 AWS 服务 (p. 217)。

要了解有关使用外部身份提供商的更多信息，请参阅 身份池（联合身份）外部身份提供商 (p. 330)。

要了解有关集成自己的后端身份验证流程的更多信息，请参阅经开发人员验证的身份（身份池）(p. 357)。

获取凭证

Amazon Cognito 身份池为来宾用户（未经身份验证）及经过身份验证并收到了令牌的用户提供临时的AWS凭证。借助这些 AWS 凭证，您的应用程序可以通过 Amazon API Gateway 在 AWS 内部或 AWS 外部安全地访问后端。请参阅获取凭证 (p. 322)。

使用身份池（联合身份）

Amazon Cognito 身份池为来宾用户（未经身份验证）和经过身份验证并收到令牌的用户提供临时AWS凭证。身份池是用于存储特定于您的账户的用户身份数据的存储区。

New console

在控制台中创建新的身份池

1. 登录 Amazon Cognito 控制台并选择身份池。
2. 选择创建身份池。
3. 在配置身份池信任中，选择将您的身份池设置为经过身份验证的访问权限和/或访客访问权限。

- 如果您选择了经过身份验证的访问权限，请在身份池中选择一个或多个您要设置为经过身份验证的身份来源的身份类型。如果您配置了自定义开发人员提供者，则在创建身份池后无法对其进行修改或删除。

4. 在配置权限中，为身份池中经过身份验证的用户或访客用户选择原定设置 IAM 角色。

   a. 如果您希望 Amazon Cognito 为您创建一个具有基本权限并与您的身份池建立信任关系的新角色，请选择创建新的 IAM 角色。例如，输入 IAM 角色名称以标识您的新角色，例如 `myidentitypool_authenticatedrole`。选择查看策略文档以查看 Amazon Cognito 将分配给新 IAM 角色的权限。

   b. 如果您的 AWS 账户中已经有想要使用的角色，则可以选择使用现有的 IAM 角色。您必须将您的 IAM 角色信任策略配置为包括 `cognito-identity.amazonaws.com`。配置您的角色信任策略，以仅允许 Amazon Cognito 在提供证据证明请求来自您的特定身份池中经过身份验证的用户时，才代入该角色。有关更多信息，请参阅角色信任和权限 (p. 312)。

5. 在连接身份提供者中，输入您在配置身份池信任中选择的身份提供者（IdP）的详细信息。可能会要求您提供 OAuth 应用程序客户端信息、选择 Amazon Cognito 用户群体、选择 IAM IdP 或输入开发人员提供者的自定义标识符。

   a. 为每个 IdP 选择角色设置。您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。使用 Amazon Cognito 用户群体 IdP，还可以选择令牌中包含 preferred_role 声明的角色。有关 `cognito:preferred_role` 声明的更多信息，请参阅将优先级值分配到组 (p. 179)。

      i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。

      ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。

   b. 您可以为每个 IdP 配置访问控制属性。访问控制属性将用户声明映射到 Amazon Cognito 应用于其临时会话的主体标签。您可以构建 IAM policy，以根据应用于用户会话的标签来筛选用户访问权限。

      i. 如果不应用主体标签，请选择非活动。

      ii. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。

      iii. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。

6. 在配置属性中，在身份池名称下输入名称。
7. 在基本（经典）身份验证下，选择是否要激活基本流程。在基本流程处于活动状态的情况下，您可以绕过为 IdP 所做的角色选择，直接调用 AssumeRoleWithWebIdentity。有关更多信息，请参阅身份池（联合身份）身份验证流程 (p. 300)。
8. 如果要将标签应用到身份池，请在标签下选择添加标签。
9. 在查看并创建中，确认您为新身份池所做的选择。选择编辑以返回向导并更改任何设置。完成后，选择创建身份池。

Original console

在控制台中创建新的身份池

1. 登录到 Amazon Cognito 控制台，选择 Manage identity pools（管理身份池），然后选择 Create new identity pool（创建新的身份池）。
2. 为身份池键入一个名称。
3. 要启用未经身份验证的身份，请从 Unauthenticated identities（未经身份验证的身份）的可折叠部分中选择 Enable access to unauthenticated identities（启用访问未经身份验证的身份）。
4. 如果需要，请在身份验证提供商部分中配置身份验证提供商。

5. 选择 Create Pool（创建池）。

   Note

   有效身份池至少需要一个身份。

6. 系统将提示您访问您的AWS资源。

   选择 Allow（允许）以创建两个与您的身份池关联的默认角色，一个用于未经身份验证的用户，另一个用于经过身份验证的用户。这些默认角色会向 Amazon Cognito Sync 提供身份池访问权限。您可以在 IAM 控制台中修改与身份池关联的角色。有关使用 Amazon Cognito 控制台的更多说明，请参阅使用 Amazon Cognito 控制台 (p. 8)。

用户 IAM 角色

IAM 角色为您的用户定义访问AWS资源的权限，例如Amazon Cognito Sync (p. 371)。您的应用程序用户将担任您创建的角色。您可以为经过身份验证和未经身份验证的用户指定不同角色。要了解有关 IAM 角色的更多信息，请参阅 IAM 角色 (p. 305)。

经过身份验证和未经身份验证的身份

Amazon Cognito 身份池同时支持经过身份验证和未经身份验证的身份。经过身份验证的身份属于已通过任何受支持的身份提供商进行身份验证的用户。未经身份验证的身份通常属于来宾用户。

- 要使用公共登录提供商配置经过身份验证的身份，请参阅 身份池（联合身份）外部身份提供商 (p. 330)。
- 要配置您自己的后端身份验证流程，请参阅 经开发人员验证的身份（身份池） (p. 357)。

激活或停用访客访问权限

Amazon Cognito 身份池访客访问权限（未经身份验证的身份）针对每个未通过身份提供者进行身份验证的用户提供唯一标识符和 AWS 凭证。如果应用程序允许未登录的用户进行访问，则您可以针对未经身份验证的身份激活访问权限。要了解更多信息，请参阅 Amazon Cognito 身份池入门（联合身份） (p. 289)。

New console

更新身份池中的访客访问权限

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 找到访客访问权限。在目前不支持访客访问的身份池中，状态为非活动。

   a. 如果访客访问权限处于活动状态并且您想将其停用，请选择停用。
   b. 如果访客访问权限处于非活动状态而您想要将其激活，请选择编辑。

      - 为身份池中的访客用户选择原定设置 IAM 角色。

        A. 如果您希望 Amazon Cognito 为您创建一个具有基本权限并与您的身份池建立信任关系的新角色，请选择创建新的 IAM 角色。例如，输入 IAM 角色名称以标识您的新角色，例如 `myidentitypool_authenticatedrole`。选择查看策略文档以查看 Amazon Cognito 将分配给新 IAM 角色的权限。
        B. 如果您的 AWS 账户中已经有想要使用的角色，则可以选择使用现有的 IAM 角色。您必须将您的 IAM 角色信任策略配置为包括 `cognito-identity.amazonaws.com`。配置您的角色信任策略，以仅允许 Amazon Cognito 在提供证据证明请求来自您的特定身份池中经过身份验证的用户时，才代入该角色。有关更多信息，请参阅角色信任和权限 (p. 312)。

C. 选择 Save changes（保存更改）。
D. 要激活访客访问权限，请在用户访问权限选项卡中选择激活。

Original console

在 Amazon Cognito 控制台中选择 Manage identity pools（管理身份池）。

1. 选择想要针对其启用或禁用未经身份验证的身份的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在控制面板页面的右上角，选择 Edit identity pool（编辑身份池）。此时将显示 Edit identity pool（编辑身份池）页。
3. 向下滚动并选择 Unauthenticated identities（未经验证的身份）以将其展开。
4. 选中复选框以启用或禁用对未经身份验证的身份的访问权限。
5. 选择 Save Changes（保存更改）。

更改与身份类型关联的角色

身份池中的每个身份要么经过身份验证，要么未经过身份验证。经过身份验证的身份属于通过公共登录提供商（Amazon Cognito 用户池、Login with Amazon、Sign in with Apple、Facebook、Google、SAML 或任何 OpenID Connect 提供商）或开发人员提供商（自己的后端身份验证流程）验证身份的用户。未经身份验证的身份通常属于来宾用户。

每个身份类型都有一个分配的角色。该角色附加了一个策略，指明角色可以访问的 AWS 服务。Amazon Cognito 接收请求后，服务将确定身份类型、确定分配给该身份类型的角色，并使用附加到该角色的策略进行响应。通过修改策略或向身份类型分配不同的角色，您可以控制身份类型可以访问的 AWS 服务。要查看或修改与身份池中与角色关联的策略，请参阅 AWS IAM 控制台。

New console

更改身份池的原定设置经过身份验证或未经身份验证的角色

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 找到访客访问权限或经过身份验证的访问权限。在当前未为该访问类型配置的身份池中，状态为非活动。选择 Edit（编辑）。
4. 为身份池中的访客用户或经过身份验证的用户选择原定设置 IAM 角色。
   a. 如果您希望 Amazon Cognito 为您创建一个具有基本权限并与您的身份池建立信任关系的新角色，请选择创建新的 IAM 角色。例如，输入 IAM 角色名称以标识您的新角色，例如 `myidentitypool_authenticatedrole`。选择查看策略文档以查看 Amazon Cognito 将分配给新 IAM 角色的权限。
   b. 如果您的 AWS 账户中已经有想要使用的角色，则可以选择使用现有的 IAM 角色。您必须将您的 IAM 角色信任策略配置为包括 `cognito-identity.amazonaws.com`。配置您的角色信任策略，以仅允许 Amazon Cognito 在提供证据证明请求来自您的特定身份池中经过身份验证的用户时，才代入该角色。有关更多信息，请参阅角色信任和权限 (p. 312)。
5. 选择 Save changes（保存更改）。

Original console

您可以使用 Amazon Cognito 身份池（联合身份）控制台更改与身份类型关联的角色。在 Amazon Cognito 控制台中选择 Manage identity pools（管理身份池）。

1. 选择想要针对其修改角色的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。

2. 在控制面板页面的右上角，选择 Edit identity pool（编辑身份池）。此时将显示 Edit identity pool（编辑身份池）页。
3. 使用 Unauthenticated role（未经身份验证的角色）和 Authenticated role（经过身份验证的角色）旁边的下拉菜单更改角色。在 AWS IAM 控制台中，选择 Create new role（创建新角色）以创建或修改与各个身份类型关联的角色。有关更多信息，请参阅 IAM 角色。

编辑身份提供者

如果您允许用户通过使用者身份提供者（例如，Amazon Cognito 用户群体、Login with Amazon、通过 Apple 登录、Facebook 或 Google）进行身份验证，则您可以在 Amazon Cognito 身份池（联合身份）控制台中指定应用程序标识符。上述操作会将应用程序 ID (由公共登录提供商提供) 与身份池关联。

您还可以从此页面为每个提供商配置身份验证规则。每个提供商最多可以有 25 个规则。规则按您为各个提供商保存的顺序应用。有关更多信息，请参阅使用基于角色的访问控制 (p. 318)。

Warning

更改身份池中关联的 IdP 应用程序 ID 可防止现有用户通过该身份池进行身份验证。有关更多信息，请参阅身份池（联合身份）外部身份提供商 (p. 330)。

New console

更新身份池身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 找到身份提供者。选择要编辑的身份提供者。如果要添加新的 IdP，请选择添加身份提供者。
   - 如果您选择添加身份提供者，请选择要添加的身份类型之一。
4. 要更改应用程序 ID，请在身份提供者信息中选择编辑。
5. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请在角色设置中选择编辑。
   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。使用 Amazon Cognito 用户群体 IdP，还可以选择令牌中包含 preferred_role 声明的角色。有关 `cognito:preferred_role` 声明的更多信息，请参阅将优先级值分配到组 (p. 179)。
     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。
     ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。
6. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请在访问控制属性中选择编辑。
   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 `sub` 和 `aud` 声明应用主体标签，请选择使用原定设置映射。
   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。
7. 选择 Save changes（保存更改）。

Original console

在 Amazon Cognito 控制台中选择 Manage identity pools（管理身份池）。

1. 选择想要针对其启用外部提供商的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在控制面板页面的右上角，选择 Edit identity pool（编辑身份池）。此时将显示 Edit identity pool（编辑身份池）页。
3. 向下滚动并选择 Authentication providers（身份验证提供商）以将其展开。
4. 选择相应提供商的选项卡，然后输入与该身份验证提供商相关的所需信息。

删除身份池

您不能撤消身份池删除。删除身份池后，所有依赖该身份池的应用程序和用户将停止工作。

New console

删除身份池

1. 从 Amazon Cognito 控制台中选择身份池。选中要删除的身份池旁边的单选按钮。
2. 选择删除。
3. 输入或粘贴身份池的名称，然后选择删除。

Original console

在 Amazon Cognito 控制台中选择 Manage identity pools（管理身份池）。

1. 选择要删除的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在控制面板页面的右上角，选择 Edit identity pool（编辑身份池）。此时将显示 Edit identity pool（编辑身份池）页。
3. 向下滚动并选择 Delete identity pool（删除身份池）以将其展开。
4. 选择 Delete identity pool（删除身份池）。
5. 选择 Delete pool（删除池）。

Warning

选择删除按钮后，您将永久删除身份池和其中包含的所有用户数据。删除身份池将导致使用身份池的应用程序和其他服务停止工作。

从身份池删除身份

当您从身份池中删除身份时，您会删除 Amazon Cognito 为该联合用户存储的身份信息。当用户再次请求凭证时，如果身份池仍然信任用户的身份提供者，则用户会收到新的身份 ID。您无法撤消此操作。

New console

删除身份

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择身份浏览器选项卡。
3. 选中要删除的身份旁边的复选框，然后选择删除。确认您要删除这些身份，然后选择删除。

Original console

在 Amazon Cognito 控制台中选择 Manage identity pools（管理身份池）。

1. 选择包含要删除的身份的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在控制面板页面的左侧导航窗格中，选择 Identity browser（身份浏览器）。此时将显示 Identities（身份）页。
3. 在 Identities（身份）页面上，输入要删除的身份 ID，然后选择 Search（搜索）。
4. 在 Identity details（身份详细信息）页面上，选择 Delete identity（删除身份）按钮，然后选择 Delete（删除）。

将 Amazon Cognito Sync 与身份池一起使用

Amazon Cognito Sync 是一种 AWS 服务和客户端库，用于跨设备同步与应用程序相关的用户数据。Amazon Cognito 可以跨移动设备和 Web 同步用户配置文件数据，无需使用您自己的后端。客户端库在本地缓存数据，因此，您的应用程序可以读取和写入数据，无论设备是否处于连接状态，都是如此。设备处于在线状态时，您可以同步数据。如果您设置推送同步，您可在更新可用时立即通知其他设备。

管理数据集

如果您在应用程序中实施了 Amazon Cognito Sync 功能，则 Amazon Cognito 身份池控制台允许您手动创建和删除各个身份的数据集和记录。对于您在 Amazon Cognito 身份池控制台中对身份的数据集或记录做出的任何更改，只有当您在控制台中选择 Synchronize（同步）后才会保存。直到身份调用 Synchronize（同步）后，终端用户才能看到更改。一旦刷新特定身份的列表数据集页面，从其它设备同步的有关各个身份的数据即会显示。

为身份创建数据集

Amazon Cognito Sync 将数据集与一个身份关联起来。您可以在数据集中填充该身份所代表的用户的身份信息，然后将该信息同步到用户的所有设备。

New console

将数据集和数据集记录添加到身份

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择身份浏览器选项卡。
3. 选择要编辑的身份。
4. 在数据集中，选择创建数据集。
5. 输入数据集名称并选择创建数据集。
6. 如果您想向数据集添加记录，请从身份详细信息中选择您的数据集。在记录中，选择创建记录。
7. 输入记录的键和值。选择 Confirm（确认）。重复此操作以添加更多记录。

Original console

在 Amazon Cognito 控制台中选择 Manage identity pools（管理身份池）。

1. 选择包含想要为其创建数据集的身份的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在控制面板页面的左侧导航窗格中，选择 Identity browser（身份浏览器）。此时将显示 Identities（身份）页。
3. 在 Identities（身份）页面上，输入想要为其创建数据集的身份 ID，然后选择 Search（搜索）。
4. 在该身份的 Identity details（身份详细信息）页面上，选择 Create dataset（创建数据集）按钮、输入数据集名称，然后选择 Create and edit dataset（创建和编辑数据集）。
5. 在 Current dataset（当前数据集）页面上，选择 Create record（创建记录）以创建记录来存储在该数据集中。

6. 为该数据集输入一个密钥（有效的 JSON 值或要存储的值），然后选择 Format as JSON（JSON 格式）来格式化输入的值以便于阅读，并确认值是格式正确的 JSON。完成后，选择 Save Changes（保存更改）。
7. 选择 Synchronize（同步）来同步数据集。您做出的更改在您选择 Synchronize（同步）后才会保存，并且在身份调用 Synchronize（同步）后才对用户可见。要放弃未同步的更改，请选择您想要放弃的更改，然后选择放弃更改。

删除与身份关联的数据集

New console

从身份中删除数据集及其记录

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择身份浏览器选项卡。
3. 选择包含要删除的数据集的身份。
4. 在数据集中，选择要删除的数据集旁边的单选按钮。
5. 选择删除。查看您的选择，然后再次选择删除。

Original console

在 Amazon Cognito 控制台中选择 Manage identity pools（管理身份池）。

1. 选择包含想要删除其数据集的身份的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在控制面板页面的左侧导航窗格中，选择 Identity browser（身份浏览器）。此时将显示 Identities（身份）页。
3. 在 Identities（身份）页面上，输入包含想要删除的数据集的身份 ID，然后选择 Search（搜索）。
4. 在 Identity details（身份详细信息）页面上，选中想要删除的数据集旁边的复选框，选择 Delete selected（删除选定项），然后选择 Delete（删除）。

批量发布数据

批量发布可用于将存储在 Amazon Cognito Sync 存储中的数据导出到 Amazon Kinesis Stream。有关如何批量发布所有流的说明，请参阅 Amazon Cognito Streams (p. 396)。

激活推送同步

Amazon Cognito 会自动跟踪身份和设备之间的关联。通过使用推送同步功能，可以确保在身份数据发生更改时通知给定身份的每个实例。推送同步可以确保，只要身份的数据集发生更改，与该身份关联的所有设备就会收到一个静音推送通知，通知它们所发生的更改。

您可以在 Amazon Cognito 控制台中激活推送同步。

New console

激活推送同步

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择身份池属性选项卡。
3. 在推送同步中，选择编辑
4. 选择激活与身份池的推送同步。

5. 选择您在当前 AWS 区域中创建的 Amazon Simple Notification Service（Amazon SNS）平台应用程序之一。Amazon Cognito 向您的平台应用程序发布推送通知。选择创建平台应用程序以导航到 Amazon SNS 控制台并创建一个新的应用程序。
6. 要发布到平台应用程序，Amazon Cognito 将代入您的 AWS 账户中的 IAM 角色。如果您希望 Amazon Cognito 为您创建一个具有基本权限并与您的身份池建立信任关系的新角色，请选择创建新的 IAM 角色。例如，输入 IAM 角色名称以标识您的新角色，例如 `myidentitypool_authenticatedrole`。选择查看策略文档以查看 Amazon Cognito 将分配给新 IAM 角色的权限。
7. 如果您的 AWS 账户中已经有想要使用的角色，则可以选择使用现有的 IAM 角色。您必须将您的 IAM 角色信任策略配置为包括 `cognito-identity.amazonaws.com`。将您的角色信任策略配置为仅允许 Amazon Cognito 在提供证据证明请求来自您的特定身份池中经过身份验证的用户时代入该角色。有关更多信息，请参阅角色信任和权限 (p. 312)。
8. 选择 Save changes（保存更改）。

Original console

在 Amazon Cognito 控制台中选择 Manage identity pools（管理身份池）。

1. 选择您需要启用推送同步的身份池的名称。此时将显示身份池的 Dashboard page（控制面板）页。
2. 在控制面板页面的右上角，选择 Edit identity pool（编辑身份池）。此时将显示 Edit identity pool（编辑身份池）页。
3. 向下滚动并选择 Push synchronization（推送同步）以将其展开。
4. 在 Service role（服务角色）下拉列表中，选择授予 Amazon Cognito 发送 Amazon Simple Notification Service（Amazon SNS）通知的权限的 IAM 角色。在 AWS IAM 控制台中，选择 Create role（创建角色）以创建或修改与您身份池关联的角色。
5. 选择一个平台应用程序，然后选择 Save Changes（保存更改）。

设置 Amazon Cognito Streams

Amazon Cognito Streams 让开发人员能够控制和了解他们存储在 Amazon Cognito Sync 中的数据。开发人员现在可以配置 Kinesis 流以接收数据形式的事件。Amazon Cognito 可以实时向您拥有的 Kinesis 流推送每个数据集更改。有关如何在 Amazon Cognito 控制台中设置 Amazon Cognito Streams 的说明，请参阅Amazon Cognito Streams (p. 396)。

设置 Amazon Cognito Events

Amazon Cognito Events 使您能够运行 AWS Lambda 函数以响应 Amazon Cognito Sync 中的重要事件。当数据集得到同步时，Amazon Cognito Sync 会引发同步触发事件。当用户更新数据时，您可以使用同步触发事件采取行动。有关从控制台设置 Amazon Cognito Events 的说明，请参阅Amazon Cognito Events (p. 397)。

要了解有关 AWS Lambda 的更多信息，请参阅AWS Lambda。

身份池概念（联合身份）

您可以使用 Amazon Cognito 身份池，为用户创建唯一的身份，并通过身份提供商对其进行身份验证。借助身份，您可以获取权限受限的临时 AWS 凭证以访问其他 AWS 服务。Amazon Cognito 身份池支持公有身份提供商（Amazon、Apple、Facebook 和 Google），以及未经身份验证的身份。此外，它还支持已经过开发人员验证的身份，这让您能够通过自己的后端身份验证流程注册用户并对用户进行身份验证。

有关 Amazon Cognito 身份池区域可用性的信息，请参阅AWS服务区域可用性。有关 Amazon Cognito 身份池概念的更多信息，请参阅以下主题。

主题

- 身份池（联合身份）身份验证流程 (p. 300)
- IAM 角色 (p. 305)
- 角色信任和权限 (p. 312)

身份池（联合身份）身份验证流程

Amazon Cognito 可帮助您为终端用户创建在多个设备和平台间保持一致的唯一标识符。此外，Amazon Cognito 还可以为应用程序提供具有有限权限的临时凭证来访问AWS资源。此页面介绍有关 Amazon Cognito 中的身份验证如何工作的基础知识，并解释了身份池中身份的生命周期。

外部提供商身份验证流程

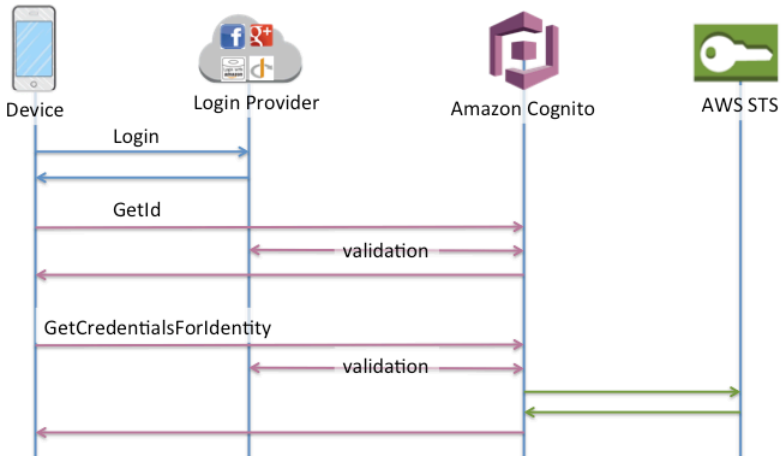
使用 Amazon Cognito 进行身份验证的用户将通过一个多步骤流程来引导启动其凭证。Amazon Cognito 提供两个不同的通过公有提供商进行身份验证的流程：增强型流程和基本流程。

完成其中的一个流程后，您便可以访问由您角色的访问策略定义的其他 AWS 服务。默认情况下，Amazon Cognito 控制台创建可访问 Amazon Cognito Sync 存储和 Amazon Mobile Analytics 的角色。有关如何授予额外访问权限的更多信息，请参阅 IAM 角色 (p. 305)。

增强型（简化的）身份验证流程

当您使用增强型身份验证流程时，您的应用程序首先显示 `GetID` 请求中来自已授权 Amazon Cognito 用户池或第三方身份提供商的 ID 令牌。应用程序将该令牌交换为身份池中的身份 ID。然后，身份 ID 将用于 `GetCredentialsForIdentity` 请求中的相同身份提供商令牌。增强型工作流通过在后台执行 `GetOpenIdToken` 和 `AssumeRoleWithWebIdentity` 为您简化凭证检索。`GetCredentialsForIdentity` 返回 AWS API 凭证。

1. `GetId`
2. `GetCredentialsForIdentity`

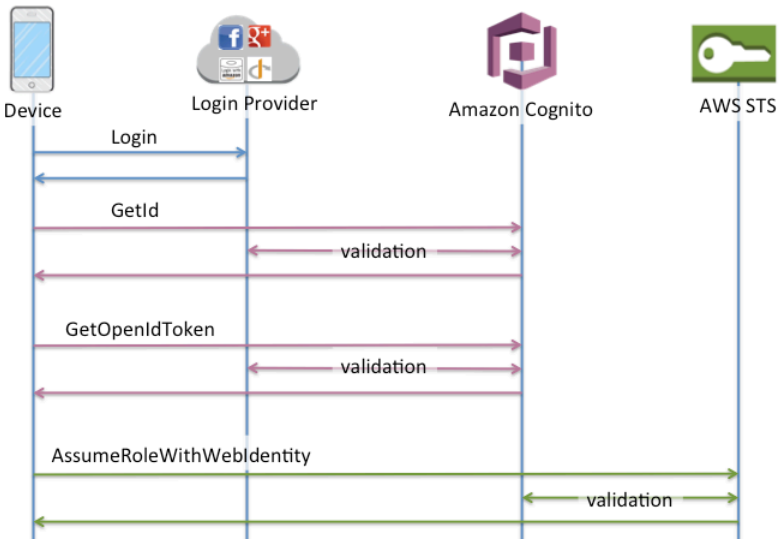


基本（经典）工作流程

当您使用基本身份验证流程时，您的应用程序首先显示 GetID 请求中来自已授权 Amazon Cognito 用户群体或第三方身份提供者的 ID 令牌。应用程序将该令牌交换为身份池中的身份 ID。然后，身份 ID 将用于 GetOpenIdToken 请求中的相同身份提供者令牌。`GetOpenIdToken` 返回由身份池颁发的新 OAuth 2.0 令牌。然后，您可以使用 AssumeRoleWithWebIdentity 请求中的新令牌来检索 AWS API 凭证。通过基本型工

作流，您可以更精细地控制分发给用户的凭证。增强型身份验证流程的 `GetCredentialsForIdentity` 请求根据访问令牌的内容请求角色。经典工作流的 `AssumeRoleWithWebIdentity` 请求使您的应用程序能够更好地请求任何配置了足够信任策略的 AWS Identity and Access Management 角色的凭证。

1. `GetId`
2. `GetOpenIdToken`
3. `AssumeRoleWithWebIdentity`

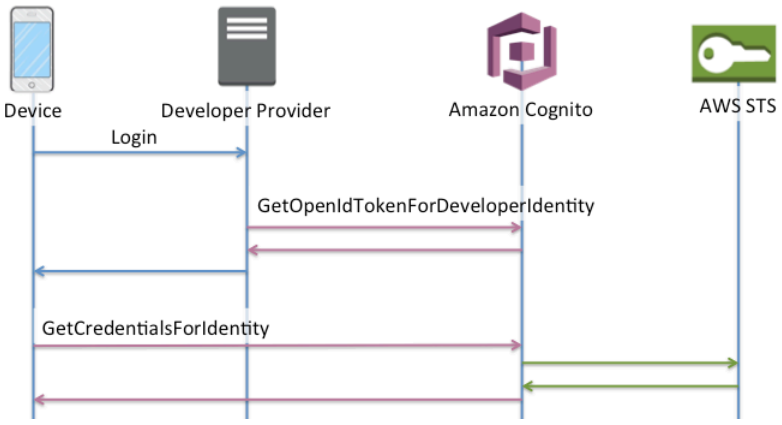


已经过开发人员验证的身份验证流程

使用经开发人员验证的身份（身份池）(p. 357)时，客户端将使用包括 Amazon Cognito 外部代码的不同身份验证流程，在您自己的身份验证系统中验证用户。Amazon Cognito 外部代码如此处所示。

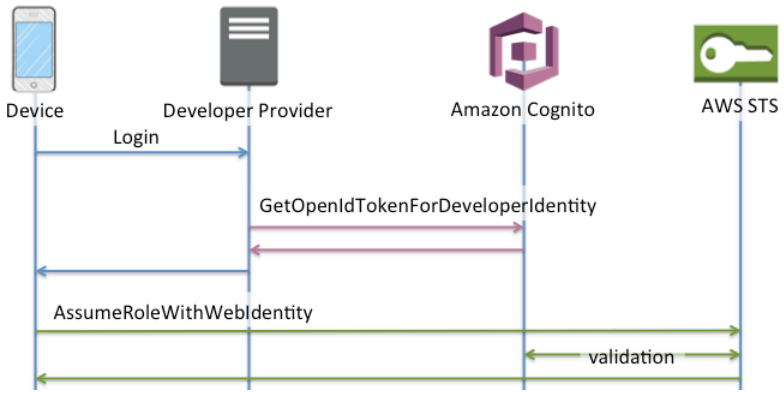
增强型身份验证流程

1. 通过开发人员提供商登录（Amazon Cognito 外部代码）
2. 验证用户登录（Amazon Cognito 外部代码）
3. GetOpenIdTokenForDeveloperIdentity
4. GetCredentialsForIdentity



基本身份验证流程

1. 通过开发人员提供商登录（Amazon Cognito 外部代码）
2. 验证用户登录（Amazon Cognito 外部代码）
3. GetOpenIdTokenForDeveloperIdentity
4. AssumeRoleWithWebIdentity



我应该使用哪种身份验证流程？

对于大多数客户，增强型流程是正确的选择，因为它与基本流程相比可提供很多优势：

- 用来在设备上获取凭证的网络调用少一个。
- 所有调用都面向 Amazon Cognito，这意味着它的网络连接也少一个。
- 角色不再需要嵌入到应用程序中，只需一个身份池 ID 和区域即可开始引导启动凭证。

自 2015 年 2 月起，Amazon Cognito 控制台显示使用增强型流程的示例代码。此外，如果您的身份池没有使用增强型流程所需的角色关联，控制台会显示一条通知。

以下是支持增强型流程的最低开发工具包版本：

SDK（最低版本）

- AWS SDK for iOS（2.0.14）
- AWS SDK for Android（2.1.8）
- AWS SDK for JavaScript（2.1.7）
- AWS SDK for Unity（1.0.3）
- AWS SDK for Xamarin (3.0.0.5)

如果您要使用的角色不只是在控制台中创建新身份池时配置的两个默认角色，请使用基本流程。

API 摘要

GetId

`GetId` API 调用是在 Amazon Cognito 中建立新身份所必需的第一个调用。

未经身份验证的访问

Amazon Cognito 能够在您的应用程序中授权未经身份验证的访客访问。如果您的身份池中已启用此功能，用户可以随时通过 `GetId` API 请求新的身份 ID。该应用程序应缓存此身份 ID，以对

Amazon Cognito 发出后续调用。AWS Mobile SDK 和 AWS SDK for JavaScript in the Browser 拥有可为您处理此缓存的凭证提供商。

经过身份验证的访问

在您将应用程序配置为支持公共登录提供商（Facebook、Google+、Login with Amazon 或 Sign in with Apple）后，用户也可以提供用于在这些提供商中识别他们的令牌（OAuth 或 OpenID Connect）。在对 `GetId` 的调用中使用时，Amazon Cognito 创建一个经过身份验证的新身份，或者是返回已与该特定登录关联的身份。Amazon Cognito 通过向提供商验证令牌并确保以下各项来实现此目的：

- 令牌有效且来自已配置的提供商。
- 令牌未过期。
- 令牌与使用该提供商创建的应用程序标识符（如 Facebook 应用程序 ID）匹配。
- 令牌与用户标识符匹配。

GetCredentialsForIdentity

在建立身份 ID 后，即可调用 `GetCredentialsForIdentity` API。此 API 在功能上等同于调用 `GetOpenIdToken` 随后调用 `AssumeRoleWithWebIdentity`。

要让 Amazon Cognito 代表您调用 `AssumeRoleWithWebIdentity`，您的身份池必须具有与之关联的 IAM 角色。您可以通过 Amazon Cognito 控制台，也可以手动通过 SetIdentityPoolRoles 操作实现此目的。

GetOpenIdToken

在建立身份 ID 后，即可发出 `GetOpenIdToken` API 请求。在您的第一个请求后缓存身份 ID，然后使用 `GetOpenIdToken` 启动该身份的后续基本（经典）会话。

对 `GetOpenIdToken` API 请求的响应是 Amazon Cognito 生成的令牌。您可以将此令牌作为 `AssumeRoleWithWebIdentity` 请求中的 `WebIdentityToken` 参数提交。

在提交 OpenID 令牌之前，请在您的应用程序中进行验证。您可以使用 SDK 中的 OIDC 库或类似于 aws-jwt-verify 的库来确认 Amazon Cognito 颁发了令牌。OpenID 令牌的签名密钥 ID 或 `kid` 是 Amazon Cognito 身份 jwks_uri 文档中列出的密钥之一。这些密钥可能会发生变化。验证 Amazon Cognito 身份令牌的函数应定期更新它在 jwks_uri 文档中的密钥列表。Amazon Cognito 在 jwks_uri 缓存控制响应标头中设置刷新时长，而 `max-age` 当前设置为 30 天。

未经身份验证的访问

要为未经身份验证的身份获取令牌，您只需身份 ID 本身。无法为经身份验证的身份或已停用的身份获取未经身份验证的令牌。

经过身份验证的访问

如果您有一个经过身份验证的身份，您必须为已与该身份关联的登录名传递至少一个令牌。在 `GetOpenIdToken` 调用期间，所有传入的令牌都必须通过之前提到的同一验证；如果有任何令牌失败，整个调用都会失败。`GetOpenIdToken` 调用的响应中还包括身份 ID。这是因为您传入的身份 ID 可能不是返回的那个身份 ID。

关联登录名

如果您为尚未与任何身份关联的登录名提交令牌，该登录名将视为已“关联”到关联身份。您只能为每个公共提供商链接一个登录名。如果尝试将多个登录名链接到一个公共提供商，将导致 `ResourceConflictException` 错误响应。如果登录名只链接到一个现有身份，则 `GetOpenIdToken` 返回的身份 ID 将与传入的相同。

合并身份

如果您为目前未链接到给定身份，但链接到另一身份的登录名传入令牌，这两个身份将合并。两个身份一旦合并，其中一个身份会成为所有关联登录名的父项/所有者，另一个身份会被禁用。在这种情况下，将返回父项/所有者的身份 ID。如果此值不同，则必须更新本地缓存。AWS Mobile SDKs 或 AWS SDK for JavaScript in the Browser 中的提供商为您执行此操作。

GetOpenIdTokenForDeveloperIdentity

使用已经过开发人员验证的身份时，`GetOpenIdTokenForDeveloperIdentity` API 将替代设备使用的 `GetId` 和 `GetOpenIdToken`。由于此 API 调用由您的AWS凭证签署，Amazon Cognito 会相信此 API 调用中提供的用户标识符有效。这将替代 Amazon Cognito 对外部提供商执行的令牌验证。

此 API 的负载包含一个登录映射，其中必须包含您的开发人员提供商密钥，以及作为系统中用户标识符的值。如果用户标识符尚未关联到现有身份，Amazon Cognito 会创建新的身份，并为该身份返回新身份 ID 以及 OpenID Connect 令牌。如果用户标识符已关联，Amazon Cognito 返回预先存在的身份 ID 和 OpenID Connect 令牌。在您的第一个请求后缓存开发人员身份 ID，然后使用 `GetOpenIdTokenForDeveloperIdentity` 启动该身份的后续基本（经典）会话。

对 `GetOpenIdTokenForDeveloperIdentity` API 请求的响应是 Amazon Cognito 生成的令牌。您可以将此令牌作为 `AssumeRoleWithWebIdentity` 请求中的 `WebIdentityToken` 参数提交。

在提交 OpenID Connect 令牌之前，请在您的应用程序中进行验证。您可以使用 SDK 中的 OIDC 库或类似于 aws-jwt-verify 的库来确认 Amazon Cognito 颁发了令牌。OpenID Connect 令牌的签名密钥 ID 或 `kid` 是 Amazon Cognito 身份 jwks_uri 文档中列出的密钥之一。这些密钥可能会发生变化。验证 Amazon Cognito 身份令牌的函数应定期更新它在 jwks_uri 文档中的密钥列表。Amazon Cognito 在 jwks_uri `cache-control` 响应标头中设置刷新时长，`max-age` 当前设置为 30 天。

关联登录名

与外部提供商一样，提供尚未与身份关联的额外登录名会将这些登录名隐式关联到该身份。如果您将一个外部提供者登录名链接到一个身份，用户可以对该提供者使用外部提供者身份验证流程。但是，他们无法在调用 `GetId` 或 `GetOpenIdToken` 时使用登录映射中的开发人员提供者名称。

合并身份

借助已经过开发人员验证的身份，Amazon Cognito 同时支持通过 MergeDeveloperIdentities API 进行隐式合并和显式合并。通过显式合并，您可以使用系统中的用户标识符将两个身份标记为单个身份。如果您提供了源和目标用户标识符，Amazon Cognito 会将其合并。您下次为任一用户标识符请求 OpenID Connect 令牌时，系统都会返回同一身份 ID。

AssumeRoleWithWebIdentity

拥有 OpenID Connect 令牌后，您就可以通过 AWS Security Token Service (STS) 中的 `AssumeRoleWithWebIdentity` API 调用将此令牌换为临时 AWS 凭证。这个调用与直接使用 Facebook、Google+、Login with Amazon 或 Sign in with Apple 的情况并无不同。唯一的例外是您传递 Amazon Cognito 令牌，而不是其他某个公共提供商的令牌。

由于可以创建的身份数量没有限制，所以您务必要了解向用户授予的权限。为应用程序设置不同的角色：一个用于未经身份验证的用户，一个用于经过身份验证的用户。默认情况下，当您首次设置身份池时，Amazon Cognito 控制台会为您创建这些角色。这两个角色的访问策略将完全相同：它将授予用户访问 Amazon Cognito Sync 的权限，并且这些用户可将事件提交到 Amazon Mobile Analytics。您可以修改这些角色以满足您的需求。

了解有关 角色信任和权限 (p. 312) 的更多信息。

† 原定设置 Amazon Cognito 身份 jwks_uri 文档包含有关在大多数 AWS 区域 中用于签署身份池令牌的密钥的信息。以下区域有不同的 jwks_uri 文档。

Amazon Cognito Identity JSON web key URIs in other AWS 区域

AWS 区域	jwks_uri 文档的路径
AWS GovCloud (US-West)	`https://cognito-identity.us-gov-west-1.amazonaws.com/.well-known/jwks_uri`

AWS 区域	jwks_uri 文档的路径
China (Beijing)	`https://cognito-identity.cn-north-1.amazonaws.com.cn/.well-known/jwks_uri`
Opt-in Regions like Europe (Milan) and Africa (Cape Town)	`https://cognito-identity.Region.amazonaws.com/.well-known/jwks_uri`

您还可以从颁发者推断出 jwks_uri，或者从 Amazon Cognito 推断出在 OpenID 令牌中收到的 `iss`。OIDC 标准发现端点 `<issuer>/.well-known/openid-configuration` 列出了您的令牌的 jwks_uri 的路径。

IAM 角色

在创建身份池时，系统会提示您更新用户代入的 IAM 角色。IAM 角色的工作方式如下：当用户登录应用程序时，Amazon Cognito 为用户生成临时 AWS 凭证。这些临时凭证与特定 IAM 角色相关联。通过 IAM 角色，您可以定义一组权限，用来访问您的 AWS 资源。

您可以为经过身份验证的用户和未经身份验证的用户指定默认 IAM 角色。此外，您可以定义规则，以便基于用户 ID 令牌中的声明为每个用户选择角色。有关更多信息，请参阅使用基于角色的访问控制 (p. 318)。

默认情况下，Amazon Cognito 控制台将创建可提供访问 Amazon Mobile Analytics 和 Amazon Cognito Sync 的权限的角色。或者，您也可以选择使用现有的 IAM 角色。

修改 IAM 角色以允许或限制对其他服务的访问。为此，请登录 IAM 控制台。然后，选择 Roles（角色），选择一个角色。Permissions（权限）选项卡中会列出选定角色所附加的策略。您可以选择相应的 Manage Policy（管理策略）链接自定义访问策略。要了解如何使用和定义策略的更多信息，请参阅 IAM 策略概述。

> Note
>
> 作为最佳实践，定义策略时应遵循授予最低权限的原则。换言之，策略只包含用户执行其任务所需的权限。有关更多信息，请参阅 IAM 用户指南中的授予最低权限。
> 请记住，未经验证的身份会被未登录应用的用户所利用。通常情况下，为未经验证的身份分配的权限应该比为经过验证的身份分配的权限更严格。

主题

- 设置信任策略 (p. 305)
- 访问策略 (p. 306)

设置信任策略

Amazon Cognito 使用 IAM 角色为应用程序的用户生成临时凭证。对权限的访问由角色的信任关系控制。了解有关 角色信任和权限 (p. 312) 的更多信息。

跨身份池重复使用角色

要跨多个身份池重复使用某个角色，由于它们共享一个通用权限集，您可以添加多个身份池，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"
    ]
```

```
}
```

限制对特定身份的访问权限

要创建限制为一组特定应用用户的策略，请检查 cognito-identity.amazonaws.com:sub 的值：

```
"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"
    ]
}
```

限制对特定提供商的访问权限

要创建仅限于已使用特定提供商 (可能是您自己的登录提供商) 登录的用户的策略，请检查 cognito-identity.amazonaws.com:amr 的值：

```
"ForAnyValue:StringLike": {
    "cognito-identity.amazonaws.com:amr": "login.myprovider.myapp"
}
```

例如，一个仅信任 Facebook 的应用程序将具有以下 amr 子句：

```
"ForAnyValue:StringLike": {
    "cognito-identity.amazonaws.com:amr": "graph.facebook.com"
}
```

访问策略

您附加到某个角色的权限适用于代入该角色的所有用户。为区分用户的访问权限，请使用策略条件和变量。有关更多信息，请参阅 IAM policy 元素：变量和标签。您可以在访问策略中使用 sub 条件，将操作限制到 Amazon Cognito 身份 ID。请谨慎使用此选项，特别是对于未经身份验证的身份，因为这些身份缺少一致的用户 ID。有关使用 Amazon Cognito 进行 Web 联合身份验证的 IAM policy 变量的更多信息，请参阅《AWS Identity and Access Management 用户指南》中的 AWS STSIAM 和条件上下文密钥。

为提供更好的安全保护，Amazon Cognito 使用 GetCredentialsForIdentity，对在增强型流程中分配给未经身份验证用户的凭证应用缩小范围策略。缩小范围策略可向您对未经身份验证的角色应用的 IAM policy 添加 内联会话策略 (p. 308) 和 AWS 托管式会话策略 (p. 309)。由于您必须在角色的 IAM policy 和会话策略中授予访问权限，因此，范围缩小策略限制了用户对以下服务列表以外的服务的访问权限。

> **Note**
>
> 在基本（经典）流程中，您可以发出自己的 AssumeRoleWithWebIdentity API 请求，并可以将这些限制应用于请求。作为最佳安全实践，请勿向未经身份验证的用户分配超出此缩小范围策略的任何权限。

在使用增强型流程的成功请求中，Amazon Cognito 在后台发出 AssumeRoleWithWebIdentity API 请求。在此请求的参数中，Amazon Cognito 包括以下内容。

1. 用户的身份 ID。
2. 用户所需要代入的 IAM 角色的 ARN。
3. 一个 policy 参数，添加内联会话策略。
4. 一个 PolicyArns.member.N 参数，其值是在 Amazon CloudWatch 中授予额外权限的 AWS 托管式策略。

未经身份验证的用户可以访问的服务

当您使用增强型流程时，Amazon Cognito 对您的用户会话应用的范围缩小策略会阻止用户会话使用下表中列出的服务以外的任何服务。对于服务子集，仅允许特定操作。

类别	服务
分析	Amazon Kinesis Data Firehose Amazon Kinesis Data Analytics
应用程序集成	Amazon Simple Queue Service
AR 和 VR	Amazon Sumerian[1]
业务应用程序	Amazon Mobile Analytics Amazon Simple Email Service
计算	AWS Lambda
加密和 PKI	AWS Key Management Service[1]
数据库	Amazon DynamoDB Amazon SimpleDB
前端 Web 和移动	AWS AppSync Amazon Location Service Amazon Simple Notification Service Amazon Pinpoint
游戏开发	Amazon GameLift
物联网 (IoT)	AWS IoT
Machine Learning	Amazon CodeWhisperer Amazon Comprehend Amazon Lex Amazon Machine Learning Amazon Personalize Amazon Polly Amazon Rekognition Amazon SageMaker[1] Amazon Textract[1] Amazon Transcribe Amazon Translate
管理与治理	Amazon CloudWatch

类别	服务
	Amazon CloudWatch Logs
联网和内容分发	Amazon API Gateway
安全性、身份与合规性	Amazon Cognito Identity Amazon Cognito 用户池 Amazon Cognito Sync
存储	Amazon Simple Storage Service

[1] 对于下表中的 AWS 服务，内联策略授予了一部分操作的权限。该表显示了每个服务中的可用操作。

AWS 服务	未经身份验证的增强型流程用户的最大权限

AWS Key Management Service	`Encrypt`
	`Decrypt`
	`ReEncrypt`
	`GenerateDataKey`

Amazon SageMaker	`InvokeEndpoint`
Amazon Textract	`DetectDocumentText` `AnalyzeDocument`
Amazon Sumerian	`View*`

要授予对超出此列表之外的 AWS 服务的访问权限，请在身份池中激活基本（经典）身份验证流程。如果您的用户发现，在分配给未经身份验证用户的 IAM 角色的策略中，其允许的 AWS 服务中出现了 `NotAuthorizedException` 错误，请评估您是否可以在使用案例中删除该服务。如果不能，请切换到基本流程。

内联会话策略

内联会话策略对用户的有效权限进行限制，不能包括对以下列表之外的任何 AWS 服务 的访问权限。您还必须在应用到用户的 IAM 角色的策略中，将权限授予这些 AWS 服务。对于代入角色的会话，用户的有效权限是分配给其角色的策略与其会话策略的交集。有关更多信息，请参阅《AWS Identity and Access Management 用户指南》中的会话策略。

Amazon Cognito 将以下内联策略添加到原定设置情况下启用的 AWS 区域中的用户会话。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:*",
                "logs:*",
                "dynamodb:*",
                "kinesis:*",
```

```
                "mobileanalytics:*",
                "s3:*",
                "ses:*",
                "sns:*",
                "sqs:*",
                "lambda:*",
                "machinelearning:*",
                "execute-api:*",
                "iot:*",
                "gamelift:*",
                "scs:*",
                "cognito-identity:*",
                "cognito-idp:*",
                "lex:*",
                "polly:*",
                "comprehend:*",
                "translate:*",
                "transcribe:*",
                "rekognition:*",
                "mobiletargeting:*",
                "firehose:*",
                "appsync:*",
                "personalize:*",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "sagemaker:InvokeEndpoint",
                "cognito-sync:*",
                "sumerian:View*",
                "codewhisperer:*",
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument",
                "sdb:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

对于所有其他区域，内联范围缩小策略包括原定设置区域中列出的所有内容，以下 `Action` 语句除外。

```
                "cognito-sync:*",
                "sumerian:View*",
                "codewhisperer:*",
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument",
                "sdb:*"
```

AWS 托管式会话策略

Amazon Cognito 还会在增强流程中将 AWS 托管式策略 `AmazonCognitoUnauthenticatedIdentities` 附加到未经身份验证的用户。您还必须在附加到未经身份验证的 IAM 角色的策略中授予此权限。

`AmazonCognitoUnauthenticatedIdentities` 托管式策略具有以下权限。

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

```
        "Action": [
            "rum:PutRumEvents",
            "polly:*",
            "comprehend:*",
            "translate:*",
            "transcribe:*",
            "rekognition:*",
            "mobiletargeting:*",
            "firehose:*",
            "personalize:*",
            "sagemaker:InvokeEndpoint"
        ],
        "Resource": "*"
    }]
}
```

访问策略示例

在此部分中，您可以找到示例 Amazon Cognito 访问策略，这些策略仅授予您的身份完成特定操作所需的权限。您可以在可能的情况下使用策略变量进一步限制给定标识 ID 的权限。例如，使用 ${cognito-identity.amazonaws.com:sub}。有关更多信息，请参阅AWS移动博客上的 nderstanding Amazon Cognito Authentication Part 3: Roles and Policies。

> **Note**
>
> 作为安全性最佳实践，策略应仅包括用户执行其任务所需的权限。这意味着您应该尽可能始终为对象限定单个身份的访问范围。

向身份授予对 Amazon S3 中单个对象的读取访问权限

以下访问策略向身份授予读取权限，以便从给定的 S3 存储桶中检索单个对象。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/assets/my_picture.jpg"]
    }
  ]
}
```

向身份授予对 Amazon S3 中身份特定路径的读写访问权限

以下访问策略通过将前缀映射到 ${cognito-identity.amazonaws.com:sub} 变量来授予读取和写入权限，以访问 S3 存储桶中的特定前缀“文件夹”。

利用此策略，通过 ${cognito-identity.amazonaws.com:sub} 插入的身份（例如 us-east-1:12345678-1234-1234-1234-123456790ab）将能够在 arn:aws:s3:::mybucket/us-east-1:12345678-1234-1234-1234-123456790ab 中获取、放置和列出对象。但是，不会授予身份访问 arn:aws:s3:::mybucket 中的其他对象的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["s3:ListBucket"],
```

```
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket"],
      "Condition": {"StringLike": {"s3:prefix": ["${cognito-identity.amazonaws.com:sub}/
*"]}}
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/${cognito-identity.amazonaws.com:sub}/*"]
    }
  ]
}
```

为 Amazon DynamoDB 分配身份细粒度访问权限

以下访问策略使用 Amazon Cognito 环境变量，为 DynamoDB 资源提供细粒度访问控制。这些变量按身份 ID 授予对 DynamoDB 中项目的访问权限。有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的使用 IAM 策略条件实现精细访问控制。

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

向身份授予调用 Lambda 函数的权限

以下访问策略向身份授予调用 Lambda 函数的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": [
                "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
            ]
        }
    ]
```

```
 }
```

向身份授予将记录发布到 Kinesis Data Stream 的权限

以下访问策略允许身份将 PutRecord 操作与任何 Kinesis Data Streams 结合使用。它可以应用于需要将数据记录添加到账户中所有流的用户。有关更多信息，请参阅《Amazon Kinesis Data Streams 开发人员指南》中的使用 IAM 控制对 Amazon Kinesis Data Streams 资源的访问。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kinesis:PutRecord",
            "Resource": [
                "arn:aws:kinesis:us-east-1:111122223333:stream/stream1"
            ]
        }
    ]
}
```

向身份授予访问 Amazon Cognito 同步存储中其数据的权限

以下访问策略仅向身份授予访问 Amazon Cognito Sync 存储中其自己数据的权限。

```
{
  "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}/*"]
      }]
  }
```

角色信任和权限

这些角色的区别在于其信任关系。下面是未经身份验证角色的示例信任策略：

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

```
}
```

此策略向来自 cognito-identity.amazonaws.com（OpenID Connect 令牌的发布者）的联合身份用户授予代入该角色的权限。此外，策略限制令牌的 aud (在此示例中为身份池 ID) 匹配身份池。最后，策略指定的由 Amazon Cognito GetOpenIdToken API 操作发布的令牌的多值 amr 声明的数组成员之一具有值 unauthenticated。

当 Amazon Cognito 创建令牌时，它将令牌的 amr 设置为 unauthenticated 或 authenticated。如果 amr 是 authenticated，则令牌包括身份验证期间使用的所有提供商。这意味着，您可以创建一个角色，它只信任通过 Facebook 登录的用户，这只需将 amr 条件更改为如下所示即可：

```
"ForAnyValue:StringLike": {
  "cognito-identity.amazonaws.com:amr": "graph.facebook.com"
}
```

在更改角色的信任关系或尝试跨身份池使用角色时，请务必谨慎。如果您未正确配置角色来信任身份池，则 STS 结果中会出现类似以下内容的异常：

```
AccessDenied -- Not authorized to perform sts:AssumeRoleWithWebIdentity
```

如果您看到此消息，请仔细检查身份池和身份验证类型是否具有正确的角色。

将属性用于访问控制

访问控制属性是基于属性的访问权限控制（ABAC）的 Amazon Cognito 身份池实现。您可以通过基于用户属性的 Amazon Cognito 身份池使用 IAM 策略来控制对AWS资源的访问。可以从社交和企业身份提供商那里获得这些属性。您可以将提供商的访问权限和 ID 令牌或 SAML 断言中的属性映射到可在 IAM 权限策略中引用的标签。

您可以选择默认映射或在 Amazon Cognito 身份池中创建自己的自定义映射。默认映射让您可以根据一组固定的用户属性编写 IAM 策略。自定义映射允许您选择 IAM 权限策略中引用的一组自定义用户属性。Amazon Cognito 控制台中的 Attribute names (属性名称) 已映射到 Tag key for principal (委托人的标签密钥)，这些是 IAM 权限策略中引用的标签。

例如，假设您拥有一个具有免费和付费会员资格的媒体流式传输服务。您可以将媒体文件存储在 Amazon S3 中，并使用免费或高级标签对其贴标签。您可以将属性用于访问控制，以允许访问基于用户会员级别（这是用户配置文件的一部分）的免费和付费内容。您可以将成员资格属性映射到委托人的标签密钥，以传递给 IAM 权限策略。通过这种方式，您可以创建单个权限策略，并根据会员级别的值和内容文件上的标签有条件地允许对高级内容的访问。

主题

- 使用属性对 Amazon Cognito 身份池进行访问控制 (p. 314)
- 示例：将属性用于访问控制策略 (p. 315)
- 关闭访问控制属性（控制台）(p. 316)
- 默认提供商映射 (p. 317)

使用属性来控制访问有若干优势：

- 使用属性进行访问控制时，权限管理会更高效。您可以创建使用用户属性的基本权限策略，而不必为不同的任务功能创建多个策略。
- 不管您何时为应用程序添加或删除资源或用户，都无需更新策略。权限策略只向具有匹配用户属性的用户授予访问权限。例如，您可能需要根据用户的任务标题控制对某些 S3 存储桶的访问权限。在这种情况

下，您可以创建权限策略，以便仅允许定义的任务标题中的用户访问这些文件。有关更多信息，请参阅 IAM 教程：将 SAML 会话标签用于 ABAC。

- 属性可以作为委托人标签传递给策略，该策略基于这些属性的值允许或拒绝权限。

使用属性对 Amazon Cognito 身份池进行访问控制

使用属性进行访问控制之前，请确保满足以下先决条件：

- AWS 账户
- 用户池
- 身份池
- 设置 SDK (p. 291)
- 已集成身份提供商
- 凭据

要使用属性进行访问控制，您设置为数据来源的声明将设置您选择的标签键的值。Amazon Cognito 会将标签键和值应用于您的用户会话。您的 IAM policy 可以根据 `${aws:PrincipalTag/tagkey}` 条件评估用户的访问权限。IAM 会根据策略评估用户标签的值。

New console

在控制台中配置访问控制属性

1. 登录 Amazon Cognito 控制台并选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 找到身份提供者。选择要编辑的身份提供者。如果要添加新的 IdP，请选择添加身份提供者。
4. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请在访问控制属性中选择编辑。
   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。
   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。
5. 选择 Save changes（保存更改）。

Original console

要使用属性进行访问控制，您必须配置 Tag Key for Principal (委托人的标签密钥) 和 Attribute (属性) 名称。在 Tag Key for Principal (委托人的标签密钥) 中，该值用于匹配权限策略中的 PrincipalTag 条件。Attribute name (属性名称) 中的值是将在策略中评估其值的属性名称。

使用属性对身份池进行访问控制

1. 打开 Amazon Cognito 控制台。
2. 选择 Manage Identity Pools（管理身份池）。
3. 在控制面板上，选择您要使用属性进行访问控制的身份池名称。
4. 选择编辑身份池。
5. 展开 Authentication providers（身份验证提供商）部分。
6. 在 Authentication providers (身份验证提供商) 部分中，选择您想要使用的提供商选项卡。

7. 在 Attributes for access control (访问控制属性) 中，选择 Default attribute mappings (默认属性映射) 或 Custom attribute mappings (自定义属性映射)。每个提供商的默认映射不同。有关更多信息，请参阅默认提供商映射 (p. 317)以获取访问控制的属性。
8. 如果选择 Custom attribute mappings (自定义属性映射)，请完成以下步骤。

   1. 在 Tag Key for Principal (委托人的标签密钥) 中，输入您的自定义文本。最大长度为 128 个字符。
   2. 在 Attribute name (属性名称) 中，输入来自提供商令牌或 SAML 断言的属性名称。您可以从提供者的开发人员指南中获取 IdP 的属性名称。属性名称最多可包含 256 个字符。此外，所有属性的总字符限制为 460 字节。
   3. （可选）添加其他提供商。您可以在控制台中来为 Amazon Cognito 用户池、OIDC 和 SAML 提供商添加多个提供商。例如，您可以将两个 Amazon Cognito 用户池作为两个独立的身份提供商添加。Amazon Cognito 将每个选项卡视为不同的 IdP。您可以为每个 IdP 单独配置访问控制属性。
   4. 要完成操作，请使用 IAM 控制台创建包含默认映射的许可策略，或您在 Tag Key for Principal (委托人的标签密钥) 中提供的自定义文本映射。有关在 IAM 中创建权限策略的教程，请参阅《IAM 用户指南》中的 IAM 教程：根据标签定义访问AWS资源的权限。

示例：将属性用于访问控制策略

考虑这样一种场景：某公司法律部门的员工需要列出存储桶中属于其部门并按其安全级别分类的所有文件。假定此员工从身份提供商获得的令牌包含以下陈述。

声明

```
{ .
  .
"sub" : "57e7b692-4f66-480d-98b8-45a6729b4c88",
"department" : "legal",
"clearance" : "confidential",
 .
 .
}
```

这些属性可以映射到标签，并在 IAM 权限策略中作为委托人标签引用。现在，您可以通过更改身份提供商端的用户配置文件来管理访问权限。或者，您可以使用名称或标签来更改资源端的属性，而无需更改策略本身。

以下权限策略有两个作用：

- 允许列表访问以与用户部门名称匹配的前缀结尾的所有 S3 桶。
- 允许对这些存储桶中的文件进行读取访问，只要文件上的清理标签与用户的清理属性匹配。

权限策略

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

信任策略决定谁可担任此角色。信任关系策略允许使用 `sts:AssumeRoleWithWebIdentity` 和 `sts:TagSession` 来允许访问。它添加了一部分条件，以将策略限制为您创建的身份池，并确保该策略适用于经身份验证的角色。

信任策略

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

关闭访问控制属性（控制台）

按照以下操作步骤来停用访问控制属性。

New console

在控制台中停用访问控制属性

1. 登录 Amazon Cognito 控制台并选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 找到身份提供者。选择要编辑的身份提供者。
4. 在访问控制属性中选择编辑。

5. 如果不应用主体标签，请选择非活动。
6. 选择 Save changes（保存更改）。

Original console

禁用访问控制属性

1. 打开 Amazon Cognito 控制台。
2. 选择 Manage Identity Pools（管理身份池）。
3. 在控制面板上，选择您要禁用其访问控制属性的身份池名称。
4. 选择页面右上角的 Edit identity pool (编辑身份池)。
5. 展开 Authentication providers（身份验证提供商）部分。
6. 在 Authenticated role selection (身份验证角色选择) 中，选择 Disable (禁用)。
7. 要完成，请滚动至页面底部，然后选择 Save Changes (保存更改)。

默认提供商映射

下表提供 Amazon Cognito 支持的身份验证提供商的默认映射信息。

Provider	令牌类型	委托人标签值	示例
Amazon Cognito 用户池	ID 令牌	aud（客户端 ID）和 sub（用户 ID）	"6jk8ltokc7ac9es6jrtg9q572f"，"57e7b6
Facebook	访问令牌	aud（app_id），sub（user_id）	“4[illegible]844718097981”，“1121772169923
Google	ID 令牌	aud（客户端 ID）和 sub（用户 ID）	“620493171733-eebk7c0hcp5lj3e1tlqp1gntt3k0rncv.app
SAML	断言	“http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier”，“http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name”	"auth0|5e28d196f8f55a0eaaa95de3"，"user12
Apple	ID 令牌	aud（客户端 ID）和 sub（用户 ID）	“com.amazonaws.ec2-54-80-172-243.c
Amazon	访问令牌	aud（Amzn Dev Ac 上的客户端 ID），user_id（用户 ID）	“amzn1.application-oa2-client.9d70d9382d3446108aaee3dd763
标准 OIDC 提供商	ID 令牌和访问令牌	aud（作为 client_id）和 sub（作为用户 ID）	“620493171733-eebk7c0hcp5lj3e1tlqp1gntt3k0rncv.app
Twitter	访问令牌	aud（应用程序 ID；应用程序密钥），sub（用户 ID）	"DfwifTtKEX1FiIBRnOTIR0CFK;Xgj5xb
DevAuth	映射	不适用	"tag1", "tag2"

Note

Tag Key for Principal (委托人的标签密钥) 和 Attribute (属性) 名称的默认属性映射选项会自动填充。您无法更改默认映射。

使用基于角色的访问控制

Amazon Cognito 身份池为经过身份验证的用户分配了一组可访问 AWS 资源的具有有限权限的临时凭证。每个用户的权限通过您创建的 IAM 角色进行控制。您可以定义规则，以便基于用户 ID 令牌中的声明为每个用户选择角色。您可以为经过身份验证的用户定义一个默认角色。您也可以为未经身份验证的来宾用户定义一个具有有限权限的单独的 IAM 角色。

为角色映射创建角色

请务必为每个角色添加适当的信任策略，这样只能由 Amazon Cognito 针对您的身份池中经过身份验证的用户担任。下面是此类信任策略的示例：

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

此策略允许来自 `cognito-identity.amazonaws.com` (OpenID Connect 令牌的发布者) 的联合身份用户担任该角色。此外，策略限制令牌的 `aud` (在此示例中为身份池 ID) 匹配身份池。最后，策略指定的由 Amazon Cognito `GetOpenIdToken` API 操作发布的令牌的多值 `amr` 声明的数组成员之一具有值 `authenticated`。

授予传递角色权限

要允许用户为角色设置超过该用户在身份池上的现有权限的权限，您需授予用户 `iam:PassRole` 权限以将角色传递给 `set-identity-pool-roles` API。例如，如果用户无法写入 Amazon S3，但用户在身份池上设置的 IAM 角色可向 Amazon S3 授予写入权限，则用户只能在已向该角色授予 `iam:PassRole` 权限的情况下设置该角色。以下示例策略介绍了如何提供 `iam:PassRole` 权限。

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

```
            "Sid": "Stmt1",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myS3WriteAccessRole"
            ]
        }
    ]
}
```

在此策略示例中，将 `iam:PassRole` 权限授予了角色 `myS3WriteAccessRole`。使用角色的 Amazon Resource Name（ARN）指定该角色。您必须将此策略附加到您的用户。有关更多信息，请参阅使用管理的策略。

> **Note**
>
> Lambda 函数使用基于资源的策略，该策略直接附加至 Lambda 函数本身。创建调用 Lambda 函数的规则时，您未传递角色，因此创建规则的用户无需 `iam:PassRole` 权限。有关 Lambda 函数授权的更多信息，请参阅管理权限：使用 Lambda 函数策略。

使用令牌向用户分配角色

对于通过 Amazon Cognito 用户池登录的用户，角色可在用户池分配的 ID 令牌中进行传递。角色将显示在 ID 令牌的以下声明中：

- `cognito:preferred_role` 声明是角色 ARN。
- `cognito:roles` 声明是一个以逗号分隔的字符串，其中包含一组允许的角色 ARN。

声明按如下方式设置：

- `cognito:preferred_role` 声明设置为组中具有最大 (最小) `Precedence` 值的角色。如果只有一个允许的角色，则 `cognito:preferred_role` 设置为该角色。如果存在多个角色且没有一个角色具有最高优先级，则此声明未设置。
- 如果至少存在一个角色，则 `cognito:roles` 声明已设置。

使用令牌分配角色时，如果存在多个可向用户分配的角色，Amazon Cognito 身份池（联合身份）将按以下方式选择角色：

- 使用 GetCredentialsForIdentity `CustomRoleArn` 参数（在参数已设置且与 `cognito:roles` 声明中的角色相匹配的情况下）。如果此参数与 `cognito:roles` 中的角色不匹配，则拒绝访问。
- 如果 `cognito:preferred_role` 声明已设置，请使用它。
- 如果 `cognito:preferred_role` 声明未设置、`cognito:roles` 声明已设置但未在 `GetCredentialsForIdentity` 的调用中指定 `CustomRoleArn`，则使用控制台中的 Role resolution（角色解析）设置或 `AmbiguousRoleResolution` 字段（在 SetIdentityPoolRoles API 的 `RoleMappings` 参数中）来确定要分配的角色。

使用基于规则的映射向用户分配角色

规则允许您将身份提供商令牌的声明映射到 IAM 角色。

每个规则指定一个令牌声明（例如 Amazon Cognito 用户池的 ID 令牌中的用户属性）、匹配类型、值和 IAM 角色。匹配类型可以是 `Equals`、`NotEqual`、`StartsWith` 或 `Contains`。如果用户拥有声明的匹配

值，则该用户可在获取凭证后担任该角色。例如，您可以创建一个规则，为 custom:dept 自定义属性值为 Sales 的用户分配一个特定的 IAM 角色。

Note

在规则设置中，自定义属性需要使用 custom: 前缀，以便将它们与标准属性区分开来。

规则按顺序进行评估，并使用第一条匹配规则的 IAM 角色，除非指定 CustomRoleArn 以覆盖顺序。有关 Amazon Cognito 用户池中用户属性的更多信息，请参阅用户池属性 (p. 243)。

您可以在身份池（联合身份）控制台中为身份验证提供商设置多条规则。按顺序应用规则。您可以拖动规则以更改其顺序。第一条匹配规则优先。如果匹配类型为 NotEqual 且声明不存在，则不会对规则进行评估。如果没有规则匹配，则角色解析设置应用到使用经过身份验证的原定设置角色或拒绝请求。

在 API 和 CLI 中，您可以指定在以下情况下要分配的角色：AmbiguousRoleResolutionRoleMapping 类型 (在 SetIdentityPoolRolesRoleMappings API 的 参数中指定) 的 字段中没有规则匹配。

您可以使用 RoleMapping 类型的 RulesConfiguration 字段，在 AWS CLI 或 API 中为 OpenID Connect (OIDC) 和 SAML 身份提供商设置基于规则的映射。您可以在 SetIdentityPoolRoles API 的 RoleMappings 参数中指定此字段。AWS Management Console当前不允许为 OIDC 或 SAML 提供商添加规则。

例如，以下 AWS CLI 命令添加规则，将角色 arn:aws:iam::123456789012:role/Sacramento_team_S3_admin 分配给位于萨克拉门托位置由 OIDC IdP arn:aws:iam::123456789012:oidc-provider/myOIDCIdP 进行身份验证的用户：

```
aws cognito-identity set-identity-pool-roles --region us-east-1 --cli-input-json file://
role-mapping.json
```

role-mapping.json 的内容：

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

对于每个用户池或为某个身份池配置的其他身份验证提供商，您可以创建最多 25 条规则。此限制不可调整。有关更多信息，请参阅 Amazon Cognito 中的配额。

基于规则的映射中使用的令牌声明

Amazon Cognito

Amazon Cognito ID 令牌以 JSON Web Token（JWT）表示。令牌包含有关经过身份验证的用户的身份声明，例如 `name`、`family_name` 和 `phone_number`。有关标准声明的更多信息，请参阅 OpenID Connect 规范。除了标准声明之外，以下是特定于 Amazon Cognito 的额外声明：

- `cognito:groups`
- `cognito:roles`
- `cognito:preferred_role`

Amazon

以下声明以及这些声明可能的值可与 Login with Amazon 配合使用：

- `iss`：www.amazon.com
- `aud`：应用程序 ID
- `sub`：Login with Amazon 令牌中的 `sub`

Facebook

以下声明以及这些声明可能的值可与 Facebook 配合使用：

- `iss`：graph.facebook.com
- `aud`：应用程序 ID
- `sub`：Facebook 令牌中的 `sub`

Google

Google 令牌包含 OpenID Connect 规范中的标准声明。OpenID 令牌中的所有声明均可用于基于规则的映射。请参阅 Google 的 OpenID Connect 网站，了解 Google 令牌中包含的声明。

Apple

Apple 令牌包含 OpenID Connect 规范中的标准声明。请参阅 Apple 文档中的使用 Sign in with Apple 对用户进行身份验证，详细了解有关 Apple 令牌提供的声明。Apple 的令牌并不总是包含 `email`。

OpenID

OpenID 令牌中的所有声明均可用于基于规则的映射。有关标准声明的更多信息，请参阅 OpenID Connect 规范。请参阅您的 OpenID 提供商文档，了解其中包含的任何额外声明。

SAML

根据收到的 SAML 断言分析声明。SAML 断言中包含的所有声明均可在基于规则的映射中使用。

基于角色的访问控制的最佳实践

> **Important**
>
> 如果最终用户可修改您映射到角色的声明，则任何最终用户均可担任您的角色并相应地设置策略。仅将无法由最终用户直接设置的声明映射到具有提升的权限的角色。在 Amazon Cognito 用户池中，您可以为每个用户属性设置每个应用程序的读取和写入权限。

> **Important**
>
> 如果您在 Amazon Cognito 用户池中为组设置角色，这些角色将通过用户的 ID 令牌进行传递。要使用这些角色，您还必须为身份池中选择的通过身份验证的角色设置 Choose role from token（使用令牌选择角色）。
> 您可以使用控制台中的 Role resolution（角色解析）设置以及 SetIdentityPoolRoles API 的 `RoleMappings` 参数，指定无法确定令牌中的正确角色时的默认行为。

获取凭证

您可以使用 Amazon Cognito 将具有有限权限的临时凭证传输到应用程序，以便您的用户访问AWS资源。本部分介绍如何获取凭证以及如何从身份池检索 Amazon Cognito 身份。

Amazon Cognito 同时支持经过身份验证和未经身份验证的身份。未经身份验证的用户的身份未经过验证，因此，该角色很适合您的应用程序的来宾用户或用户身份验证与否无关紧要的情形。经过身份验证的用户可以通过第三方身份提供商或证实其身份的用户池登录到您的应用程序。确保您的资源的权限范围适当，让未经身份验证的用户无权访问这些资源。

Amazon Cognito 身份并不是凭证。可以在 AWS Security Token Service (AWS STS) 中使用 Web 联合身份验证支持来用它们交换凭证。建议使用 `AWS.CognitoIdentityCredentials` 来为您的应用程序用户获得AWS凭证。然后，使用 AWS STS 用凭证对象中的身份交换凭证。

> **Note**
>
> 如果您的身份池是在 2015 年 2 月前创建的，则必须将角色与身份池重新关联，以便在没有角色作为参数的情况下使用 `AWS.CognitoIdentityCredentials` 构造函数。为此，请打开 Amazon Cognito 控制台，选择 Manage identity pools（管理身份池）、选择您的身份池，然后选择 Edit identity Pool（编辑身份池），指定您的经过身份验证的角色和未经身份验证的角色，然后保存更改。

Web 身份凭证提供者是 AWS SDK 中原定设置凭证提供者链的一部分。要在 AWS SDK 或 AWS CLI 的本地 `config` 文件中设置身份池令牌，请添加 `web_identity_token_file` 配置文件条目。请参阅《AWS SDK 和工具参考指南》中的代入角色凭证提供者。

要详细了解如何在 SDK 中填充 Web 身份凭证，请参阅《SDK 开发人员指南》。为了获得最佳结果，请使用内置到 AWS Amplify 中的身份池集成开始您的项目。

用于通过身份池获取和设置凭证的 AWS SDK 资源

- 《Amplify 开发人员中心》的身份池联合身份验证（Android）
- 《Amplify 开发人员中心》的身份池联合身份验证（iOS）
- 《AWS SDK for JavaScript 开发人员指南》中的使用 Amazon Cognito 身份对用户进行身份验证
- 《AWS SDK for .NET 开发人员指南》中的 Amazon Cognito 凭证提供者
- 《AWS SDK for Go 开发人员指南》中的以编程方式指定凭证
- 《AWS SDK for Java 2.x 开发人员指南》中的在代码中提供临时凭证
- 《AWS SDK for PHP 开发人员指南》中的 assumeRoleWithWebIdentityCredentialProvider 提供者
- AWS SDK for Python (Boto3) 文档中的代入 Web 身份提供者的角色
- 《AWS SDK for Rust 开发人员指南》中的指定您的凭证和原定设置区域

以下几节提供一些旧版 AWS SDK 中的示例代码。

Android

您可以使用 Amazon Cognito 将具有有限权限的临时凭证传输到应用程序，以便您的用户访问 AWS 资源。Amazon Cognito 同时支持经过身份验证和未经身份验证的身份。要向应用程序提供AWS凭证，请按照以下步骤操作。

New console

要在 Android 应用程序中使用 Amazon Cognito 身份池，请设置 AWS Amplify。有关更多信息，请参阅《Amplify 开发中心》中的身份验证。

Original console

1. 在 Amazon Cognito 控制台中选择 Manage identity pools（管理身份池），创建一个身份池并复制启动代码段。
2. 如果您尚未完成此操作，请将 Amplify Android 或旧版 AWS Mobile SDK for Android 添加到您的项目中。以下说明适用于 AWS Mobile SDK for Android。
3. 包括以下导入语句：

```
import com.amazonaws.auth.CognitoCachingCredentialsProvider;
import com.amazonaws.regions.Regions;
```

4. 使用 Amazon Cognito 控制台生成的代码段初始化 Amazon Cognito 凭证提供程序。IDENTITY_POOL_ID 的值将特定于您的账户：

```
CognitoCachingCredentialsProvider credentialsProvider = new
 CognitoCachingCredentialsProvider(
  getApplicationContext(), // Context
  "IDENTITY_POOL_ID", // Identity Pool ID
  Regions.US_EAST_1 // Region
);
```

5. 将初始化的 Amazon Cognito 凭证提供程序传递到要使用的AWS客户端的构造器中。所需的代码取决于要初始化的服务。客户端将使用此提供程序获取用于访问 AWS 资源的凭证。

   Note

   如果您在 2015 年 2 月之前创建了身份池，您必须将您的角色与身份池重新关联，才能在没有角色作为参数的情况下使用此构造函数。为此，请打开 Amazon Cognito 控制台，选择 Manage Federated Identities（管理联合身份）、选择您的身份池，然后选择 Edit identity Pool（编辑身份池），指定您的经过身份验证的角色和未经身份验证的角色，然后保存更改。

检索 Amazon Cognito 身份

如果您允许未经身份验证的用户，则可以立即检索终端用户的唯一 Amazon Cognito 标识符（身份 ID）。如果您正在对用户进行身份验证，则可以在设置完凭证提供程序中的登录令牌后检索身份 ID：

```
String identityId = credentialsProvider.getIdentityId();
Log.d("LogTag", "my ID is " + identityId);
```

Note

请勿在应用程序的主线程中调用 `getIdentityId()`、`refresh()` 或 `getCredentials()`。自 Android 3.0（API 级别 11）起，如果您在应用程序的主线程上执行网络 I/O，您的应用程序将自动失败并引发 NetworkOnMainThreadException。您必须使用 `AsyncTask` 将您的代码移至后台线程。有关更多信息，请参阅 Android 文档。您也可以调用 `getCachedIdentityId()` 以检索 ID，但前提是已缓存在本地。否则，该方法将返回 null 值。

iOS - Objective-C

您可以使用 Amazon Cognito 将具有有限权限的临时凭证传输到应用程序，以便您的用户访问AWS资源。Amazon Cognito 身份池同时支持经过身份验证和未经身份验证的身份。要向应用程序提供 AWS 凭证，请完成以下步骤。

New console

要在 iOS 应用程序中使用 Amazon Cognito 身份池，请设置 AWS Amplify。有关更多信息，请参阅《Amplify 开发中心》中的 Swift 身份验证和 Flutter 身份验证。

Original console

1. 在 Amazon Cognito 控制台中选择 Manage identity pools（管理身份池），创建一个身份池并复制启动代码段。
2. 如果您尚未执行此操作，请将适用于 Swift 的 Amplify 库或 Flutter 或旧版 AWS Mobile SDK for iOS 添加到您的项目中。以下说明适用于 AWS Mobile SDK for iOS。
3. 在您的源代码中，请包括 AWSCore 标头：

```
#import <AWSCore/AWSCore.h>
```

4. 使用 Amazon Cognito 控制台生成的代码段初始化 Amazon Cognito 凭证提供程序。IDENTITY_POOL_ID 的值将特定于您的账户：

```
AWSCognitoCredentialsProvider *credentialsProvider =
 [[AWSCognitoCredentialsProvider alloc]
initWithRegionType:AWSRegionUSEast1 identityPoolId:@"IDENTITY_POOL_ID"];
AWSServiceConfiguration *configuration = [[AWSServiceConfiguration alloc]
initWithRegion:AWSRegionUSEast1 credentialsProvider:credentialsProvider];
AWSServiceManager.defaultServiceManager.defaultServiceConfiguration =
 configuration;
```

> **Note**
>
> 如果您在 2015 年 2 月之前创建了身份池，您必须将您的角色与身份池重新关联，才能在没有角色作为参数的情况下使用此构造函数。为此，请打开 Amazon Cognito 控制台，选择 Manage identity pools（管理身份池）、选择您的身份池，然后选择 Edit identity Pool（编辑身份池），指定您的经过身份验证的角色和未经身份验证的角色，然后保存更改。

检索 Amazon Cognito 身份

如果您允许未经身份验证的用户或者已设置完凭证提供程序中的登录令牌（如果您正在对用户进行身份验证），则可以立即检索终端用户的唯一 Amazon Cognito 标识符（身份 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` 为异步调用。如果您已在提供程序上设置身份 ID，则可以调用 `credentialsProvider.identityId` 以检索已缓存在本地的身份。但是，如果您未在提供程序上设置身份 ID，则调用 `credentialsProvider.identityId` 将返回 `nil`。有关更多信息，请参阅 Amplify iOS SDK 参考。

iOS - Swift

您可以使用 Amazon Cognito 将具有有限权限的临时凭证传输到应用程序，以便您的用户访问 AWS 资源。Amazon Cognito 同时支持经过身份验证和未经身份验证的身份。要向应用程序提供AWS凭证，请按照以下步骤操作。

New console

要在 iOS 应用程序中使用 Amazon Cognito 身份池，请设置 AWS Amplify。有关更多信息，请参阅《Amplify 开发中心》中的 Swift 身份验证。

Original console

1. 在 Amazon Cognito 控制台中选择 Manage identity pools（管理身份池），创建一个身份池并复制启动代码段。
2. 如果您尚未执行此操作，请将适用于 Swift 的 Amplify 库或 Flutter 或旧版 AWS Mobile SDK for iOS 添加到您的项目中。以下说明适用于 AWS Mobile SDK for iOS。
3. 在您的源代码中，请包括 `AWSCore` 标头：

```
import AWSCore
```

4. 使用 Amazon Cognito 控制台生成的代码段初始化 Amazon Cognito 凭证提供程序。`IDENTITY_POOL_ID` 的值将特定于您的账户：

```
let credentialsProvider = AWSCognitoCredentialsProvider(regionType: .USEast1,
 identityPoolId: "IDENTITY_POOL_ID")
let configuration = AWSServiceConfiguration(region: .USEast1, credentialsProvider:
 credentialsProvider)
AWSServiceManager.default().defaultServiceConfiguration = configuration
```

> **Note**
>
> 如果您在 2015 年 2 月之前创建了身份池，您必须将您的角色与身份池重新关联，才能在没有角色作为参数的情况下使用此构造函数。为此，请打开 Amazon Cognito 控制台，选择 Manage identity pools（管理身份池）、选择您的身份池，然后选择 Edit identity Pool（编辑身份池），指定您的经过身份验证的角色和未经身份验证的角色，然后保存更改。

检索 Amazon Cognito 身份

如果您允许未经身份验证的用户或者已设置完凭证提供程序中的登录令牌（如果您正在对用户进行身份验证），则可以立即检索终端用户的唯一 Amazon Cognito 标识符（身份 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` 为异步调用。如果您已在提供程序上设置身份 ID，则可以调用 `credentialsProvider.identityId` 以检索已缓存在本地的身份。但是，如果您未在提供程序上设置身份 ID，则调用 `credentialsProvider.identityId` 将返回 `nil`。有关更多信息，请参阅 Amplify iOS SDK 参考。

JavaScript

如果您尚未创建身份池，则在使用 AWS.CognitoIdentityCredentials 之前，先在 Amazon Cognito 控制台中创建一个身份池。

通过您的身份提供商配置身份池后，您可以使用 AWS.CognitoIdentityCredentials 验证用户身份。要将应用程序凭证配置为使用 AWS.CognitoIdentityCredentials，则为 credentials 或基于每个服务配置设置 AWS.Config 属性。以下示例使用 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;

});
```

可选的 Logins 属性是身份提供商名称到这些提供商身份令牌的映射。您如何从身份提供商获得令牌的方式取决于您使用的提供商。例如，如果 Facebook 是您的身份提供商之一，则您可以使用来自 Facebook 软件开发工具包的 FB.login 函数获取身份提供商令牌：

```
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.');
    }
});
```

检索 Amazon Cognito 身份

如果您允许未经身份验证的用户或者已设置完凭证提供程序中的登录令牌（如果您正在对用户进行身份验证），则可以立即检索终端用户的唯一 Amazon Cognito 标识符（身份 ID）：

```
var identityId = AWS.config.credentials.identityId;
```

Unity

您可以使用 Amazon Cognito 将具有有限权限的临时凭证传输到应用程序，以便您的用户访问 AWS 资源。Amazon Cognito 同时支持经过身份验证和未经身份验证的身份。要向应用程序提供AWS凭证，请按照以下步骤操作。

New console

适用于 Unity 的 AWS SDK 现在是 AWS SDK for .NET 的一部分。要开始使用 AWS SDK for .NET 中的 Amazon Cognito，请参阅《AWS SDK for .NET 开发人员指南》中的 Amazon Cognito 凭证提供者。或者，请参阅 Amplify 开发中心，了解使用 AWS Amplify 构建应用程序的选项。

Original console

1. 在 Amazon Cognito 控制台中选择 Manage identity pools（管理身份池），创建一个身份池并复制启动代码段。
2. 适用于 Unity 的 AWS 移动 SDK 现在是 AWS SDK for .NET 的一部分。如果您尚未执行此操作，请下载 AWS SDK for .NET 程序包并将其导入您的项目。您可以通过菜单资产 > 导入程序包 > 导入程序包执行此操作。
3. 将控制台中的启动器代码段粘贴到调用 Amazon Cognito 所用的脚本中。`IDENTITY_POOL_ID` 的值将特定于您的账户：

```
CognitoAWSCredentials credentials = new CognitoAWSCredentials (
      "IDENTITY_POOL_ID",    // Cognito identity Pool ID
      RegionEndpoint.USEast1 // Region
    );
```

4. 将初始化的 Amazon Cognito 凭证传递到要使用的AWS客户端的构造器中。所需的代码取决于要初始化的服务。客户端将使用此提供程序获取用于访问 AWS 资源的凭证。

   Note

   如果您在 2015 年 2 月之前创建了身份池，您必须将您的角色与身份池重新关联，才能在没有角色作为参数的情况下使用此构造函数。为此，请打开 Amazon Cognito 控制台，选择 Manage identity pools（管理身份池）、选择您的身份池，然后选择 Edit identity Pool（编辑身份池），指定您的经过身份验证的角色和未经身份验证的角色，然后保存更改。

检索 Amazon Cognito 身份

如果您允许未经身份验证的用户或者已设置完凭证提供程序中的登录令牌（如果您正在对用户进行身份验证），则可以立即检索终端用户的唯一 Amazon Cognito 标识符（身份 ID）：

```
credentials.GetIdentityIdAsync(delegate(AmazonCognitoIdentityResult<string> result) {
    if (result.Exception != null) {
        //Exception!
    }
    string identityId = result.Response;
});
```

Xamarin

您可以使用 Amazon Cognito 将具有有限权限的临时凭证传输到应用程序，以便您的用户访问 AWS 资源。Amazon Cognito 同时支持经过身份验证和未经身份验证的身份。要向应用程序提供AWS凭证，请按照以下步骤操作。

New console

适用于 Xamarin 的 AWS SDK 现在是 AWS SDK for .NET 的一部分。要开始使用 AWS SDK for .NET 中的 Amazon Cognito，请参阅《AWS SDK for .NET 开发人员指南》中的 Amazon Cognito 凭证提供者。或者，请参阅 Amplify 开发中心，了解使用 AWS Amplify 构建应用程序的选项。

Original console

1. 在 Amazon Cognito 控制台中选择 Manage identity pools（管理身份池），创建一个身份池并复制启动代码段。
2. AWS SDK for .NET 现在包括适用于 Xamarin 的 AWS 移动 SDK。如果您尚未完成此操作，请将 AWS SDK for .NET 添加到您的项目中。有关更多信息，请参阅什么是 AWS SDK for .NET 和 Xamarin？
3. 包括以下使用语句：

```
using Amazon.CognitoIdentity;
```

4. 将控制台中的启动器代码段粘贴到调用 Amazon Cognito 所用的脚本中。IDENTITY_POOL_ID 的值将特定于您的账户：

```
CognitoAWSCredentials credentials = new CognitoAWSCredentials (
  "IDENTITY_POOL_ID",    // Cognito identity Pool ID
  RegionEndpoint.USEast1 // Region
);
```

5. 将初始化的 Amazon Cognito 凭证传递到要使用的AWS客户端的构造器中。所需的代码取决于要初始化的服务。客户端将使用此提供程序获取用于访问 AWS 资源的凭证。

Note

注意：如果您在 2015 年 2 月之前创建了身份池，您必须将您的角色与身份池重新关联，才能在没有角色作为参数的情况下使用此构造函数。为此，请打开 Amazon Cognito 控制台，选择 Manage identity pools（管理身份池）、选择您的身份池，然后选择 Edit identity Pool（编辑身份池），指定您的经过身份验证的角色和未经身份验证的角色，然后保存更改。

检索 Amazon Cognito 身份

如果您允许未经身份验证的用户或者已设置完凭证提供程序中的登录令牌（如果您正在对用户进行身份验证），则可以立即检索终端用户的唯一 Amazon Cognito 标识符（身份 ID）：

```
var identityId = await credentials.GetIdentityIdAsync();
```

访问 AWS 服务

在配置 Amazon Cognito 凭证提供者并检索 AWS 凭证后，您可以创建 AWS 服务 客户端。

用于创建客户端的 AWS SDK 资源

- 《AWS SDK for C++ 开发人员指南》中的 AWS 客户端配置
- 《AWS SDK for Go 开发人员指南》中的将 AWS SDK for Go V2 与 AWS 服务 结合使用。
- 《AWS SDK for Java 2.x 开发人员指南》中的配置 HTTP 客户端
- 《AWS SDK for JavaScript 开发人员指南》中的创建和调用服务对象
- AWS SDK for Python (Boto3) 文档中的创建客户端
- 《AWS SDK for Rust 开发人员指南》中的创建服务客户端

- 《AWS SDK for Swift 开发人员指南》中的使用别名。

下面的代码段初始化 Amazon DynamoDB 客户端：

Android

要在 Android 应用程序中使用 Amazon Cognito 身份池，请设置 AWS Amplify。有关更多信息，请参阅《Amplify 开发中心》中的身份验证。

```
// Create a service client with the provider
AmazonDynamoDB client = new AmazonDynamoDBClient(credentialsProvider);
```

凭证提供程序与 Amazon Cognito 通信，为经过身份验证和未经身份验证的用户检索唯一标识符，并为 AWS Mobile SDK 检索具有有限权限的临时 AWS 凭证。检索到的凭证的有效期为 1 小时，提供程序会在凭证过期时进行刷新。

iOS - Objective-C

要在 iOS 应用程序中使用 Amazon Cognito 身份池，请设置 AWS Amplify。有关更多信息，请参阅《Amplify 开发中心》中的 Swift 身份验证和 Flutter 身份验证。

```
// 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];
```

凭证提供程序与 Amazon Cognito 通信，为经过身份验证和未经身份验证的用户检索唯一标识符，并为 AWS Mobile SDK 检索具有有限权限的临时 AWS 凭证。检索到的凭证的有效期为 1 小时，提供程序会在凭证过期时进行刷新。

iOS - Swift

要在 iOS 应用程序中使用 Amazon Cognito 身份池，请设置 AWS Amplify。有关更多信息，请参阅《Amplify 开发中心》中的 Swift 身份验证。

```
// 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")
```

凭证提供程序与 Amazon Cognito 通信，为经过身份验证和未经身份验证的用户检索唯一标识符，并为 AWS Mobile SDK 检索具有有限权限的临时 AWS 凭证。检索到的凭证的有效期为 1 小时，提供程序会在凭证过期时进行刷新。

JavaScript

```
// Create a service client with the provider
var dynamodb = new AWS.DynamoDB({region: 'us-west-2'});
```

凭证提供程序与 Amazon Cognito 通信，为经过身份验证和未经身份验证的用户检索唯一标识符，并为 AWS Mobile SDK 检索具有有限权限的临时 AWS 凭证。检索到的凭证的有效期为 1 小时，提供程序会在凭证过期时进行刷新。

Unity

适用于 Unity 的 AWS SDK 现在是 AWS SDK for .NET 的一部分。要开始使用 AWS SDK for .NET 中的 Amazon Cognito，请参阅《AWS SDK for .NET 开发人员指南》中的 Amazon Cognito 凭证提供者。或者，请参阅 Amplify 开发中心，了解使用 AWS Amplify 构建应用程序的选项。

```
// create a service client that uses credentials provided by Cognito
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials, REGION);
```

凭证提供程序与 Amazon Cognito 通信，为经过身份验证和未经身份验证的用户检索唯一标识符，并为 AWS Mobile SDK 检索具有有限权限的临时 AWS 凭证。检索到的凭证的有效期为 1 小时，提供程序会在凭证过期时进行刷新。

Xamarin

适用于 Xamarin 的 AWS SDK 现在是 AWS SDK for .NET 的一部分。要开始使用 AWS SDK for .NET 中的 Amazon Cognito，请参阅《AWS SDK for .NET 开发人员指南》中的 Amazon Cognito 凭证提供者。或者，请参阅 Amplify 开发中心，了解使用 AWS Amplify 构建应用程序的选项。

```
// create a service client that uses credentials provided by Cognito
var client = new AmazonDynamoDBClient(credentials, REGION)
```

凭证提供程序与 Amazon Cognito 通信，为经过身份验证和未经身份验证的用户检索唯一标识符，并为 AWS Mobile SDK 检索具有有限权限的临时 AWS 凭证。检索到的凭证的有效期为 1 小时，提供程序会在凭证过期时进行刷新。

身份池（联合身份）外部身份提供商

使用 `logins` 属性，您可以设置从身份提供商 (IdP) 处收到的凭证。此外，您还可以将一个身份池与多个 IdP 相关联。例如，您可以在 `logins` 属性中设置 Facebook 和 Google 令牌，以将唯一的 Amazon Cognito 身份与两个 IdP 登录相关联。用户可以使用任一账户进行身份验证，但 Amazon Cognito 返回同一用户标识符。

以下说明指导您使用 Amazon Cognito 身份池支持的 IdP 进行身份验证。

主题

- Facebook（身份池）（p. 330）
- Login with Amazon（身份池）（p. 336）
- Google（身份池）（p. 339）
- Sign in with Apple（身份池）（p. 348）
- OpenID Connect 提供程序（身份池）（p. 352）
- SAML 身份提供商（身份池）（p. 355）

Facebook（身份池）

Amazon Cognito 身份池与 Facebook 集成以针对移动应用程序用户提供联合身份验证。本部分介绍如何使用 Facebook 作为 IdP 来注册和设置应用程序。

设置 Facebook

在验证 Facebook 用户并与 Facebook API 交互之前，先要向 Facebook 注册您的应用程序。

Facebook 开发人员门户可帮助您设置应用程序。在将 Facebook 集成到您的 Amazon Cognito 身份池之前，请执行以下过程：

设置 Facebook

1. 在 Facebook 开发人员门户中，使用 Facebook 凭证登录。
2. 从 Apps（应用程序）菜单中，选择 Add a New App（添加新应用程序）。
3. 选择一个平台，然后完成快速启动流程。

Android

有关如何将 Android 应用程序与 Facebook 登录集成的更多信息，请参阅 Facebook 入门指南。

iOS - Objective-C

有关如何将 OS Objective-C 应用程序与 Facebook 登录集成的更多信息，请参阅 Facebook 入门指南。

iOS - Swift

有关如何将 iOS Swift 应用程序与 Facebook 登录集成的更多信息，请参阅 Facebook 入门指南。

JavaScript

有关如何将 JavaScript Web 应用程序与 Facebook 登录集成的更多信息，请参阅 Facebook 入门指南。

Unity

有关如何将 Unity 应用程序与 Facebook 登录集成的更多信息，请参阅 Facebook 入门指南。

Xamarin

要添加 Facebook 身份验证，请首先按照以下相应流程将 Facebook 开发工具包集成到您的应用程序中。Amazon Cognito 身份池使用 Facebook 访问令牌生成与 Amazon Cognito 身份关联的唯一用户标识符。

- 由 Xamarin 开发的 Facebook iOS 开发工具包
- 由 Xamarin 开发的 Facebook Android 开发工具包

在 Amazon Cognito 身份池控制台中配置身份提供者

使用以下过程配置身份提供者。

New console

添加 Facebook 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 Facebook。
5. 输入您在 Meta for Developers 中创建的 OAuth 项目的应用程序 ID。有关更多信息，请参阅《Meta for Developers 文档》中的 Facebook 登录。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。

    - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。

i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。

ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。

7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。

   a. 如果不应用主体标签，请选择非活动。

   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。

   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。

8. 选择 Save changes（保存更改）。

Original console

1. 在 Amazon Cognito 控制台主页中选择 Manage Identity Pools（管理身份池）。
2. 选择想要在其中启用 Facebook 作为外部提供商的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
3. 在控制面板页面的右上角，选择 Edit identity pool (编辑身份池)。此时将显示 Edit identity pool（编辑身份池）页。
4. 向下滚动并选择 Authentication providers（身份验证提供商）以展开这一部分。
5. 选择 Facebook 选项卡。
6. 选择 Unlock（解锁）。
7. 输入您从 Facebook 获取的 Facebook 应用程序 ID，然后选择保存更改。

使用 Facebook

Android

要添加 Facebook 身份验证，请先按照 Facebook 指南将 Facebook 开发工具包集成到应用程序中。然后，将 Login with Facebook（使用 Facebook 登录）按钮添加到 Android 用户界面。Facebook 开发工具包使用会话对象跟踪其状态。Amazon Cognito 使用来自此会话对象的访问令牌对用户进行身份验证、生成唯一标识符，并根据需要授予用户对其他 AWS 资源的访问权限。

使用 Facebook 开发工具包对用户进行身份验证后，将会话令牌添加到 Amazon Cognito 凭证提供程序中。

Facebook SDK 4.0 或更高版本：

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("graph.facebook.com", AccessToken.getCurrentAccessToken().getToken());
credentialsProvider.setLogins(logins);
```

Facebook SDK 4.0 之前的版本：

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("graph.facebook.com", Session.getActiveSession().getAccessToken());
credentialsProvider.setLogins(logins);
```

Facebook 登录流程在其开发工具包中初始化一个单例会话。Facebook 会话对象包含 Amazon Cognito 用于为通过身份验证的终端用户生成 AWS 凭证的 OAuth 令牌。Amazon Cognito 还使用该令牌检查用户数据

库，以确定是否存在与此特定 Facebook 身份匹配的用户。如果用户已存在，则 API 会返回现有的标识符。否则，API 将返回新的标识符。客户端开发工具包会自动在本地设备上缓存标识符。

Note

设置登录映射后，调用 `refresh` 或 `get` 以检索 AWS 凭证。

iOS - Objective-C

要添加 Facebook 身份验证，请先按照 Facebook 指南将 Facebook 开发工具包集成到应用程序中。然后，向用户界面添加“用 Facebook 登录”按钮。Facebook 开发工具包使用会话对象跟踪其状态。Amazon Cognito 使用来自此会话对象的访问令牌对用户进行身份验证，并将其绑定到唯一的 Amazon Cognito 身份池（联合身份）。

要向 Amazon Cognito 提供 Facebook 访问令牌，请实施 AWSIdentityProviderManager 协议。

在实施 `logins` 方法时，返回一个包含 `AWSIdentityProviderFacebook` 的词典。此词典充当键，而经过身份验证的 Facebook 用户的当前访问令牌充当值，如以下代码示例所示。

```
- (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"}]];
    }
}
```

当您实例化 `AWSCognitoCredentialsProvider` 时，在构造函数中传递实施 `AWSIdentityProviderManager` 作为 `identityProviderManager` 的值的类。有关更多信息，请转到 AWSCognitoCredentialsProvider 参考页面，并选择 initWithRegionType:identityPoolId:identityProviderManager。

iOS - Swift

要添加 Facebook 身份验证，请先按照 Facebook 指南将 Facebook 开发工具包集成到应用程序中。然后，向用户界面添加“用 Facebook 登录”按钮。Facebook 开发工具包使用会话对象跟踪其状态。Amazon Cognito 使用来自此会话对象的访问令牌对用户进行身份验证，并将其绑定到唯一的 Amazon Cognito 身份池（联合身份）。

要向 Amazon Cognito 提供 Facebook 访问令牌，请实施 AWSIdentityProviderManager 协议。

在实施 `logins` 方法时，返回一个包含 `AWSIdentityProviderFacebook` 的词典。此词典充当键，而经过身份验证的 Facebook 用户的当前访问令牌充当值，如以下代码示例所示。

```
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"]))
    }
}
```

当您实例化 `AWSCognitoCredentialsProvider` 时，在构造函数中传递实施 `AWSIdentityProviderManager` 作为 `identityProviderManager` 的值的

类。有关更多信息，请转到 AWSCognitoCredentialsProvider 参考页面，并选择 initWithRegionType:identityPoolId:identityProviderManager。

JavaScript

要添加 Facebook 身份验证，请按照适用于 Web 的 Facebook 登录中的说明操作，并在您的网站上添加 Login with Facebook （使用 Facebook 登录）按钮。Facebook 开发工具包使用会话对象跟踪其状态。Amazon Cognito 使用来自此会话对象的访问令牌对用户进行身份验证、生成唯一标识符，并根据需要授予用户对其他 AWS 资源的访问权限。

使用 Facebook 开发工具包对用户进行身份验证后，将会话令牌添加到 Amazon Cognito 凭证提供程序中。

```
FB.login(function (response) {

  // Check if the user logged in successfully.
  if (response.authResponse) {

    console.log('You are now logged in.');

    // Add the Facebook access token to the Amazon Cognito credentials login map.
    AWS.config.credentials = new AWS.CognitoIdentityCredentials({
      IdentityPoolId: 'IDENTITY_POOL_ID',
      Logins: {
        'graph.facebook.com': response.authResponse.accessToken
      }
    });

    // Obtain AWS credentials
    AWS.config.credentials.get(function(){
        // Access AWS resources here.
    });

  } else {
    console.log('There was a problem logging you in.');
  }

});
```

Facebook SDK 获取 Amazon Cognito 用于为通过身份验证的终端用户生成 AWS 凭证的 OAuth 令牌。Amazon Cognito 还可以使用该令牌检查用户数据库，以确定是否存在与此特定 Facebook 身份匹配的用户。如果用户已存在，则 API 会返回现有的标识符。否则，将返回新的标识符。标识符由本地设备上的客户端开发工具包自动缓存。

> **Note**
>
> 设置登录映射后，请调用 `refresh` 或 `get` 以获取凭证。有关代码示例，请参阅 JavaScript 自述文件中的“使用案例 17：将用户池与 Cognito 身份集成”。

Unity

要添加 Facebook 身份验证，请先按照 Facebook 指南将 Facebook 开发工具包集成到应用程序中。Amazon Cognito 使用 FB 对象中的 Facebook 访问令牌生成与 Amazon Cognito 身份关联的唯一用户标识符。

使用 Facebook 开发工具包对用户进行身份验证后，将会话令牌添加到 Amazon Cognito 凭证提供程序：

```
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);
}
```

使用 FB.AccessToken 之前，调用 FB.Login() 并确保 FB.IsLoggedIn 为 True。

Xamarin

Xamarin for Android：

```
public void InitializeFacebook() {
    FacebookSdk.SdkInitialize(this.ApplicationContext);
    callbackManager = CallbackManagerFactory.Create();
    LoginManager.Instance.RegisterCallback(callbackManager, new FacebookCallback &lt;
 LoginResult &gt; () {
      HandleSuccess = loginResult = &gt; {
        var accessToken = loginResult.AccessToken;
        credentials.AddLogin("graph.facebook.com", accessToken.Token);
        //open new activity
      },
      HandleCancel = () = &gt; {
        //throw error message
      },
      HandleError = loginError = &gt; {
        //throw error message
      }
    });
    LoginManager.Instance.LogInWithReadPermissions(this, new List &lt; string &gt; {
      "public_profile"
    });
  }
```

Xamarin for iOS：

```
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
    }
  });
}
```

Login with Amazon（身份池）

Amazon Cognito 与 Login with Amazon（使用亚马逊账号登录）集成，以针对移动应用程序和 Web 应用程序用户提供联合身份验证。本部分介绍如何使用 Login with Amazon（使用亚马逊账号登录）作为身份提供商 (IdP) 来注册和设置应用程序。

在开发人员门户中设置 Login with Amazon（使用亚马逊账号登录）用于 Amazon Cognito。有关更多信息，请参阅 Login with Amazon（使用亚马逊账号登录）常见问题中的设置 Login with Amazon（使用亚马逊账号登录）。

Note

要将 Login with Amazon（使用亚马逊账号登录）集成到 Xamarin 应用程序中，请按照 Xamarin 入门指南中的说明操作。

Note

您无法在 Unity 平台上原生集成 Login with Amazon（使用亚马逊账号登录）。请使用 Web 视图并完成浏览器登录流程。

设置 Login with Amazon（使用亚马逊账号登录）

实施 Login with Amazon（使用亚马逊账号登录）

在 Amazon 开发人员门户中，您可以设置 OAuth 应用程序以便与您的身份池集成，查找 Login with Amazon（使用亚马逊账号登录）文档并下载开发工具包。选择 Developer console（开发人员控制台），然后在开发人员门户中选择 Login with Amazon（使用亚马逊账号登录）。您可以为您的应用程序创建一个安全配置文件，然后在您的应用程序中构建 Login with Amazon（使用亚马逊账号登录）身份验证机制。请参阅获取凭证 (p. 322)，了解有关如何将 Login with Amazon（使用亚马逊账号登录）身份验证与应用程序集成的更多信息

Amazon 针对您的新安全配置文件发布 OAuth 2.0 client ID（客户端 ID）。您可以在安全配置文件的 Web Settings（Web 设置）选项卡上找到 client ID（客户端 ID）。在您的身份池中的 Login with Amazon（使用亚马逊账号登录）IdP 的 Amazon App ID（Amazon 应用程序 ID）字段中输入 client ID（客户端 ID）。

在 Amazon Cognito 控制台中配置外部提供商

New console

添加 Login with Amazon 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 Login with Amazon。
5. 输入您在 Login with Amazon 中创建的 OAuth 项目的应用程序 ID。有关更多信息，请参阅 Login with Amazon 文档。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。

   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。

i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。

ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。

7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。

   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。
   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。

8. 选择 Save changes（保存更改）。

Original console

在 Amazon Cognito 控制台主页中选择 Manage Identity Pools（管理身份池）。

1. 选择您要启用 Login with Amazon（使用亚马逊账号登录）作为外部提供商的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在控制面板页面的右上角，选择 Edit identity pool (编辑身份池)。此时将显示“Edit identity pool”（编辑身份池）页。
3. 向下滚动并选择 Authentication providers（身份验证提供商）以展开这一部分。
4. 选择 Amazon 选项卡。
5. 选择 Unlock（解锁）。
6. 输入从 Amazon 获取的 Amazon 应用程序 ID，然后选保存更改。

使用 Login with Amazon：Android

在对 Amazon 登录进行身份验证后，您可以在 TokenListener 接口的 onSuccess 方法中将令牌传递给 Amazon Cognito 凭证提供程序。代码如下所示：

```
@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

在对 Amazon 登录进行身份验证后，您可以在 AMZNAccessTokenDelegate 的 requestDidSucceed 方法中将令牌传递给 Amazon Cognito 凭证提供程序：

```
- (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

在对 Amazon 登录进行身份验证后，您可以在 AMZNAccessTokenDelegate 的 requestDidSucceed 方法中将令牌传递给 Amazon Cognito 凭证提供程序：

```
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

在用户通过 Login with Amazon 进行身份验证并重定向回网站后，系统会在查询字符串中提供 Login with Amazon access_token。将此令牌传递到凭证登录映射。

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
   IdentityPoolId: 'IDENTITY_POOL_ID',
   Logins: {
       'www.amazon.com': 'Amazon Access Token'
   }
});
```

使用 Login with Amazon：Xamarin

Xamarin for Android

```
AmazonAuthorizationManager manager = new AmazonAuthorizationManager(this, Bundle.Empty);

var tokenListener = new APIListener {
  Success = response => {
    // Get the auth token
    var token = response.GetString(AuthzConstants.BUNDLE_KEY.Token.Val);
    credentials.AddLogin("www.amazon.com", token);
  }
};

// Try and get existing login
manager.GetToken(new[] {
  "profile"
}, tokenListener);
```

Xamarin for iOS

在 AppDelegate.cs 中，插入以下内容：

```
public override bool OpenUrl (UIApplication application, NSUrl url, string
 sourceApplication, NSObject annotation)
{
    // Pass on the url to the SDK to parse authorization code from the url
    bool isValidRedirectSignInURL = AIMobileLib.HandleOpenUrl (url, sourceApplication);
```

```
    if(!isValidRedirectSignInURL)
        return false;

    // App may also want to handle url
    return true;
}
```

然后，在 ViewController.cs 中执行以下操作：

```
public override void ViewDidLoad ()
{
    base.LoadView ();

    // Here we create the Amazon Login Button
    btnLogin = UIButton.FromType (UIButtonType.RoundedRect);
    btnLogin.Frame = new RectangleF (55, 206, 209, 48);
    btnLogin.SetTitle ("Login using Amazon", UIControlState.Normal);
    btnLogin.TouchUpInside += (sender, e) => {
        AIMobileLib.AuthorizeUser (new [] { "profile"}, new AMZNAuthorizationDelegate ());
    };
    View.AddSubview (btnLogin);
}

// Class that handles Authentication Success/Failure
public class AMZNAuthorizationDelegate : AIAuthenticationDelegate
{
  public override void RequestDidSucceed(ApiResult apiResult)
    {
      // Your code after the user authorizes application for requested scopes
      var token = apiResult["access_token"];
      credentials.AddLogin("www.amazon.com",token);
    }

    public override void RequestDidFail(ApiError errorResponse)
    {
      // Your code when the authorization fails
      InvokeOnMainThread(() => new UIAlertView("User Authorization Failed",
 errorResponse.Error.Message, null, "Ok", null).Show());
    }
}
```

Google（身份池）

Amazon Cognito 与 Google 集成，以针对移动应用程序用户提供联合身份验证。本部分介绍如何使用 Google 作为 IdP 来注册和设置应用程序。

Android

> **Note**
>
> 如果您的应用程序使用 Google 并且可在多个移动平台上使用，则应将 Google 配置为 OpenID Connect 提供商 (p. 352)。将所有创建的客户端 ID 添加为附加受众值，以便更好地集成。要了解有关 Google 跨客户端身份模式的更多信息，请参阅跨客户端身份。

设置 Google

要激活 Google Sign-in for Android，请为应用程序创建 Google Developers 控制台项目。

1. 转到 Google Developers 控制台并创建一个新项目。
2. 选择 APIs & Services（API 和服务），然后选择 OAuth consent screen（OAuth 同意屏幕）。自定义 Google 在征求用户同意以便与您的应用共享个人资料数据时向用户显示的信息。

3. 选择 Credentials（凭证），然后选择 Create credentials（创建凭证）。选择 OAuth client ID（OAuth 客户端 ID）。选择 Android 作为 Application type（应用程序类型）。为开发应用程序的每个平台创建单独的客户端 ID。
4. 从 Credentials（凭证）中，选择 Manage service accounts（管理服务账户）。选择 Create service account（创建服务账户）。输入服务账户详细信息，然后选择 Create and continue（创建并继续）。
5. 授予服务账户对项目的访问权限。根据应用程序的要求授予用户访问服务账户的权限。
6. 选择您的新服务账户，选择 Keys（密钥）选项卡，然后选择 Add key（添加密钥）。创建并下载新的 JSON 密钥。

有关如何使用 Google Developers 控制台的更多信息，请参阅 Google Cloud 文档中的创建和管理项目。

有关将 Google 集成到 Android 应用程序的更多信息，请参阅 Google Identity 文档中的 Google Sign-In for Android。

New console

添加 Google 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 Google。
5. 输入您在 Google Cloud Platform 上创建的 OAuth 项目的客户端 ID。有关更多信息，请参阅《Google Cloud Platform 控制台帮助》中的设置 OAuth 2.0。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。
   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。
     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。
     ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。
7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。
   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 `sub` 和 `aud` 声明应用主体标签，请选择使用原定设置映射。
   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。
8. 选择 Save changes（保存更改）。

Original console

在 Amazon Cognito 控制台中配置外部提供商

在 Amazon Cognito 控制台主页中选择 Manage Identity Pools（管理身份池）：

1. 选择要将 Google 作为外部提供商激活的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在控制面板页面的右上角，选择 Edit identity pool (编辑身份池)。此时将显示“Edit identity pool”（编辑身份池）页。

3. 向下滚动并选择 Authentication providers（身份验证提供商）。“Edit identity pool”（编辑身份池）页面将展开以显示其他身份验证提供商选项。
4. 选择 Google 选项卡。
5. 选择 Unlock（解锁）。
6. 输入您在 Google Cloud 控制台中创建的 Google 客户端 ID，然后选择 Save Changes（保存更改）。

使用 Google

要在应用程序中启用 Login with Google，请按照适用于 Android 的 Google 文档中的说明执行操作。当用户登录时，他们向 Google 请求 OpenID Connect 身份验证令牌。然后，Amazon Cognito 将使用该令牌对用户进行身份验证并生成一个唯一标识符。

以下示例代码显示如何从 Google Play 服务检索身份验证令牌：

```
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

如果您的应用程序使用 Google 并且可在多个移动平台上使用，则应将 Google 配置为 OpenID Connect 提供商 (p. 352)。将所有创建的客户端 ID 添加为附加受众值，以便更好地集成。要了解有关 Google 跨客户端身份模式的更多信息，请参阅跨客户端身份。

设置 Google

要启用 Google Sign-in for iOS，请为应用程序创建 Google Developers 控制台项目。

1. 转到 Google Developers 控制台并创建一个新项目。
2. 选择 APIs & Services（API 和服务），然后选择 OAuth consent screen（OAuth 同意屏幕）。自定义 Google 在征求用户同意以便与您的应用共享个人资料数据时向用户显示的信息。
3. 选择 Credentials（凭证），然后选择 Create credentials（创建凭证）。选择 OAuth client ID（OAuth 客户端 ID）。选择 iOS 作为 Application type（应用程序类型）。为开发应用程序的每个平台创建单独的客户端 ID。
4. 从 Credentials（凭证）中，选择 Manage service accounts（管理服务账户）。选择 Create service account（创建服务账户）。输入服务账户详细信息，然后选择 Create and continue（创建并继续）。
5. 授予服务账户对项目的访问权限。根据应用程序的要求授予用户访问服务账户的权限。
6. 选择您的新服务账户。选择 Keys（密钥）选项卡，然后选择 Add key（添加密钥）。创建并下载新的 JSON 密钥。

有关如何使用 Google Developers 控制台的更多信息，请参阅 Google Cloud 文档中的创建和管理项目。

有关将 Google 集成到 iOS 应用程序的更多信息，请参阅 Google Identity 文档中的 Google Sign-In for iOS。

New console

添加 Google 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。

2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 Google。
5. 输入您在 Google Cloud Platform 上创建的 OAuth 项目的客户端 ID。有关更多信息，请参阅《Google Cloud Platform 控制台帮助》中的设置 OAuth 2.0。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。

   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。

     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。

     ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。
7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。

   a. 如果不应用主体标签，请选择非活动。

   b. 要基于 `sub` 和 `aud` 声明应用主体标签，请选择使用原定设置映射。

   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。
8. 选择 Save changes（保存更改）。

Original console

在 Amazon Cognito 控制台主页中选择 Manage Identity Pools（管理身份池）：

在 Amazon Cognito 控制台中配置外部提供商

1. 选择想要在其中启用 Google 作为外部提供商的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在控制面板页面的右上角，选择 Edit identity pool (编辑身份池)。此时将显示“Edit identity pool”（编辑身份池）页。
3. 向下滚动并选择 Authentication providers（身份验证提供商）以展开这一部分。
4. 选择 Google 选项卡。
5. 选择 Unlock（解锁）。
6. 输入从 Google 获取的 Google 客户端 ID，然后选择 Save Changes（保存更改）。

使用 Google

要在应用程序中启用“用 Google 登录”，请按照适用于 iOS 的 Google 文档中的说明执行操作。成功通过身份验证后，将生成一个 OpenID Connect 身份验证令牌，供 Amazon Cognito 用于对用户进行身份验证并生成一个唯一标识符。

成功通过身份验证后，将生成一个 `GTMOAuth2Authentication` 对象，其中包含一个 `id_token`，供 Amazon Cognito 用于对用户进行身份验证并生成一个唯一标识符：

```
- (void)finishedWithAuth: (GTMOAuth2Authentication *)auth error: (NSError *) error {
        NSString *idToken = [auth.parameters objectForKey:@"id_token"];
        credentialsProvider.logins = @{ @(AWSCognitoLoginProviderKeyGoogle): idToken };
    }
```

iOS - Swift

> **Note**
>
> 如果您的应用程序使用 Google 并且可在多个移动平台上使用，则应将 Google 配置为 OpenID Connect 提供商 (p. 352)。将所有创建的客户端 ID 添加为附加受众值，以便更好地集成。要了解有关 Google 跨客户端身份模式的更多信息，请参阅跨客户端身份。

设置 Google

要启用 Google Sign-in for iOS，请为应用程序创建 Google Developers 控制台项目。

1. 转到 Google Developers 控制台并创建一个新项目。
2. 选择 APIs & Services（API 和服务），然后选择 OAuth consent screen（OAuth 同意屏幕）。自定义 Google 在征求用户同意以便与您的应用共享个人资料数据时向用户显示的信息。
3. 选择 Credentials（凭证），然后选择 Create credentials（创建凭证）。选择 OAuth client ID（OAuth 客户端 ID）。选择 iOS 作为 Application type（应用程序类型）。为开发应用程序的每个平台创建单独的客户端 ID。
4. 从 Credentials（凭证）中，选择 Manage service accounts（管理服务账户）。选择 Create service account（创建服务账户）。输入服务账户详细信息，然后选择 Create and continue（创建并继续）。
5. 授予服务账户对项目的访问权限。根据应用程序的要求授予用户访问服务账户的权限。
6. 选择您的新服务账户，选择 Keys（密钥）选项卡，然后选择 Add key（添加密钥）。创建并下载新的 JSON 密钥。

有关如何使用 Google Developers 控制台的更多信息，请参阅 Google Cloud 文档中的创建和管理项目。

有关将 Google 集成到 iOS 应用程序的更多信息，请参阅 Google Identity 文档中的 Google Sign-In for iOS。

在 Amazon Cognito 控制台主页中选择 Manage Identity Pools（管理身份池）：

在 Amazon Cognito 控制台中配置外部提供商

1. 选择想要在其中启用 Google 作为外部提供商的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在控制面板页面的右上角，选择 Edit identity pool (编辑身份池)。此时将显示“Edit identity pool”（编辑身份池）页。
3. 向下滚动并选择 Authentication providers（身份验证提供商）以展开这一部分。
4. 选择 Google 选项卡。
5. 选择 Unlock（解锁）。
6. 输入从 Google 获取的 Google 客户端 ID，然后选择 Save Changes（保存更改）。

使用 Google

要在应用程序中启用“用 Google 登录”，请按照适用于 iOS 的 Google 文档中的说明执行操作。成功通过身份验证后，将生成一个 OpenID Connect 身份验证令牌，供 Amazon Cognito 用于对用户进行身份验证并生成一个唯一标识符。

成功的身份验证会生成一个包含 `id_token` 的 `GTMOAuth2Authentication` 对象。Amazon Cognito 使用此令牌对用户进行身份验证并生成一个唯一标识符：

```
func finishedWithAuth(auth: GTMOAuth2Authentication!, error: NSError!) {
    if error != nil {
      print(error.localizedDescription)
    }
    else {
```

```
      let idToken = auth.parameters.objectForKey("id_token")
      credentialsProvider.logins = [AWSCognitoLoginProviderKey.Google.rawValue: idToken!]
    }
}
```

JavaScript

Note

如果您的应用程序使用 Google 并且可在多个移动平台上使用，则应将 Google 配置为 OpenID Connect 提供商 (p. 352)。将所有创建的客户端 ID 添加为附加受众值，以便更好地集成。要了解有关 Google 跨客户端身份模式的更多信息，请参阅跨客户端身份。

设置 Google

要为 JavaScript Web 应用程序启用 Google Sign-in，您必须为应用程序创建 Google Developers 控制台项目。

1. 转到 Google Developers 控制台并创建一个新项目。
2. 选择 APIs & Services（API 和服务），然后选择 OAuth consent screen（OAuth 同意屏幕）。自定义 Google 在征求用户同意以便与您的应用共享个人资料数据时向用户显示的信息。
3. 选择 Credentials（凭证），然后选择 Create credentials（创建凭证）。选择 OAuth client ID（OAuth 客户端 ID）。选择 Web application（Web 应用程序）作为 Application type（应用程序类型）。为开发应用程序的每个平台创建单独的客户端 ID。
4. 从 Credentials（凭证）中，选择 Manage service accounts（管理服务账户）。选择 Create service account（创建服务账户）。输入服务账户详细信息，然后选择 Create and continue（创建并继续）。
5. 授予服务账户对项目的访问权限。根据应用程序的要求授予用户访问服务账户的权限。
6. 选择您的新服务账户，选择 Keys（密钥）选项卡，然后选择 Add key（添加密钥）。创建并下载新的 JSON 密钥。

有关如何使用 Google Developers 控制台的更多信息，请参阅 Google Cloud 文档中的创建和管理项目。

有关如何将 Google 集成到 Web 应用程序的更多信息，请参阅 Google Identity 文档中的使用 Google 登录。

在 Amazon Cognito 控制台中配置外部提供商

New console

添加 Google 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 Google。
5. 输入您在 Google Cloud Platform 上创建的 OAuth 项目的客户端 ID。有关更多信息，请参阅《Google Cloud Platform 控制台帮助》中的设置 OAuth 2.0。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。

   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。

     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。

ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。

7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。

   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。
   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。

8. 选择 Save changes（保存更改）。

Original console

在 Amazon Cognito 控制台主页中选择 Manage Identity Pools（管理身份池）：

1. 选择想要在其中启用 Google 作为外部提供商的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在控制面板页面的右上角，选择 Edit identity pool (编辑身份池)。此时将显示“Edit identity pool”（编辑身份池）页。
3. 向下滚动并选择 Authentication providers（身份验证提供商）以展开这一部分。
4. 选择 Google 选项卡。
5. 选择 Unlock（解锁）。
6. 输入从 Google 获取的 Google 客户端 ID，然后选择 Save Changes（保存更改）。

使用 Google

要在应用程序中启用“用 Google 登录”，请按照适用于 Web 的 Google 文档中的说明执行操作。

成功通过身份验证后，将生成一个响应对象，其中包含一个 id_token，供 Amazon Cognito 用于对用户进行身份验证并生成一个唯一标识符：

```
function signinCallback(authResult) {
  if (authResult['status']['signed_in']) {

     // Add the Google access token to the Amazon Cognito credentials login map.
     AWS.config.credentials = new AWS.CognitoIdentityCredentials({
        IdentityPoolId: 'IDENTITY_POOL_ID',
        Logins: {
           'accounts.google.com': authResult['id_token']
        }
     });

     // Obtain AWS credentials
     AWS.config.credentials.get(function(){
        // Access AWS resources here.
     });
  }
}
```

Unity

设置 Google

要对 Unity 应用程序启用 Google Sign-in，请为应用程序创建 Google Developers 控制台项目。

1. 转到 Google Developers 控制台并创建一个新项目。

2. 选择 APIs & Services（API 和服务），然后选择 OAuth consent screen（OAuth 同意屏幕）。自定义 Google 在征求用户同意以便与您的应用共享个人资料数据时向用户显示的信息。
3. 选择 Credentials（凭证），然后选择 Create credentials（创建凭证）。选择 OAuth client ID（OAuth 客户端 ID）。选择 Web application（Web 应用程序）作为 Application type（应用程序类型）。为开发应用程序的每个平台创建单独的客户端 ID。
4. 对于 Unity，为 Android 另外创建一个 OAuth client ID（OAuth 客户端 ID），为 iOS 再创建一个。
5. 从 Credentials（凭证）中，选择 Manage service accounts（管理服务账户）。选择 Create service account（创建服务账户）。输入服务账户详细信息，然后选择 Create and continue（创建并继续）。
6. 授予服务账户对项目的访问权限。根据应用程序的要求授予用户访问服务账户的权限。
7. 选择您的新服务账户，选择 Keys（密钥）选项卡，然后选择 Add key（添加密钥）。创建并下载新的 JSON 密钥。

有关如何使用 Google Developers 控制台的更多信息，请参阅 Google Cloud 文档中的创建和管理项目。

在 IAM 控制台中创建 OpenID 提供商

1. 在 IAM 控制台中创建 OpenID 提供商。有关如何设置 OpenID 提供商的信息，请参阅使用 OpenID Connect 身份提供商 (p. 352)。
2. 当系统提示您输入提供商 URL 时，请输入 "`https://accounts.google.com`"。
3. 当系统提示您在 Audience（受众）字段中输入一个值时，请输入您在之前步骤中创建的三个客户端 ID 中的任意一个。
4. 选择提供商名称并使用另外两个客户端 ID 添加另外两个受众。

在 Amazon Cognito 控制台中配置外部提供商

New console

在 Amazon Cognito 控制台主页中选择 Manage Identity Pools（管理身份池）：

添加 Google 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 Google。
5. 输入您在 Google Cloud Platform 上创建的 OAuth 项目的客户端 ID。有关更多信息，请参阅《Google Cloud Platform 控制台帮助》中的设置 OAuth 2.0。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。
   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。
     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。
     ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。
7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。
   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 `sub` 和 `aud` 声明应用主体标签，请选择使用原定设置映射。

c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。

8. 选择 Save changes（保存更改）。

Original console

在 Amazon Cognito 控制台主页中选择 Manage Identity Pools（管理身份池）：

1. 选择想要在其中启用 Google 作为外部提供商的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在控制面板页面的右上角，选择 Edit identity pool (编辑身份池)。此时将显示“Edit identity pool”（编辑身份池）页。
3. 向下滚动并选择 Authentication providers（身份验证提供商）以展开这一部分。
4. 选择 Google 选项卡。
5. 选择 Unlock（解锁）。
6. 输入从 Google 获取的 Google 客户端 ID，然后选择 Save Changes（保存更改）。

安装 Unity Google 插件

1. 将适用于 Unity 的 Google Play Games 插件添加到 Unity 项目。
2. 在 Unity 中，从 Windows 菜单，使用适用于 Android 和 iOS 平台的三个 ID 配置插件。

使用 Google

以下示例代码显示如何从 Google Play 服务检索身份验证令牌：

```
void Start()
{
  PlayGamesClientConfiguration config = new PlayGamesClientConfiguration.Builder().Build();
  PlayGamesPlatform.InitializeInstance(config);
  PlayGamesPlatform.DebugLogEnabled = true;
  PlayGamesPlatform.Activate();
  Social.localUser.Authenticate(GoogleLoginCallback);
}

void GoogleLoginCallback(bool success)
{
  if (success)
  {
    string token = PlayGamesPlatform.Instance.GetIdToken();
    credentials.AddLogin("accounts.google.com", token);
  }
  else
  {
    Debug.LogError("Google login failed. If you are not running in an actual Android/iOS
 device, this is expected.");
  }
}
```

Xamarin

Note

Amazon Cognito 本身并不支持 Xamarin 平台上的 Google。目前，进行集成需要使用 Web 视图来完成浏览器登录流程。要了解 Google 集成如何与其他开发工具包配合使用，请选择另一个平台。

要在应用程序中启用 Login with Google，请对用户进行身份验证并从其获取 OpenID Connect 令牌。Amazon Cognito 使用此令牌生成与 Amazon Cognito 身份关联的唯一用户标识符。遗憾的是，适用于 Xamarin 的 Google SDK 包不允许您检索 OpenID Connect 令牌，因此，请使用替代客户端或 Web 视图中的 Web 流程。

拥有令牌后，您可以在 `CognitoAWSCredentials` 中对其进行设置：

```
credentials.AddLogin("accounts.google.com", token);
```

Note

如果您的应用程序使用 Google 并且可在多个移动平台上使用，则应将 Google 配置为 OpenID Connect 提供商 (p. 352)。将所有创建的客户端 ID 添加为附加受众值，以便更好地集成。要了解有关 Google 跨客户端身份模式的更多信息，请参阅跨客户端身份。

Sign in with Apple（身份池）

Amazon Cognito 与 Sign in with Apple 集成，以针对移动应用程序和 Web 应用程序用户提供联合身份验证。本节介绍如何使用 Sign in with Apple（使用苹果账号登录）作为身份提供商 (IdP) 来注册和设置应用程序。

要将 Sign in with Apple 作为身份验证提供商添加到身份池，您必须完成两个过程。首先，在应用程序中集成 Sign in with Apple（使用苹果账号登录），然后在身份池中配置 Sign in with Apple（使用苹果账号登录）。

设置 Sign in with Apple

要将 Sign in with Apple（使用苹果账号登录）配置为 IdP，您必须向 Apple 注册您的应用程序才能接收客户端 ID。

1. 创建 Apple 开发人员账户。
2. 使用 Apple 凭证登录。
3. 在左侧导航窗格中，选择 Certificates, IDs & Profiles（证书、ID 和配置文件）。
4. 在左侧导航窗格中，选择Identifiers（标识符）。
5. 在 Identifiers（标识符）页面上，选择 + 图标。
6. 在 Register a New Identifier（注册新标识符）页面上，选择 App IDs（应用程序 ID），然后选择 Continue（继续）。
7. 在 Register an App ID（注册应用程序 ID）页面上，执行以下操作：
   a. 在 Description（描述）下方，键入描述。
   b. 在 Bundle ID（服务包 ID）下，键入标识符。记下此 Bundle ID（捆绑包 ID），因为您需要此值才能将 Apple 配置为身份池中的提供商。
   c. 在 Capabilities（功能）下，选择 Sign In with Apple，然后选择 Edit（编辑）。
   d. 在通过 Apple 登录：应用程序 ID 配置页面上，为应用程序选择适当的设置。然后选择 Save（保存）。
   e. 选择 Continue（继续）。
8. 在 Confirm your App ID（确认您的应用程序 ID）页面上，选择 Register（注册）。
9. 如果要将 Sign in with Apple 与本机 iOS 应用程序集成，请继续执行步骤 10。步骤 11 适用于您希望与 Sign in with Apple JS 集成的应用程序。
10. 在 Identifiers（标识符）页面上，选择 App IDs（应用程序 ID）菜单，然后选择 Services IDs（服务 ID）。选择 + 图标。
11. 在 Register a New Identifier（注册新标识符）页面上，选择 Services IDs（服务 ID），然后选择 Continue（继续）。
12. 在 Register an Services ID（注册服务 ID）页面上，执行以下操作：

a. 在 Description（描述）下方，键入描述。
b. 在 Identifier（标识符）下方，键入标识符。记下服务 ID，因为您需要此值才能将 Apple 配置为身份池中的提供商。
c. 选择 Sign In with Apple（使用苹果账号登录），然后选择 Configure（配置）。
d. 在 Web Authentication Configuration（Web 身份验证配置）页面上，选择 Primary App ID（主应用程序 ID）。在 Website URLs（网站 URL）下，选择 + 图标。对于 Domains and Subdomains（域和子域），输入应用程序的域名。在 Return URL（返回 URL）中，输入回调 URL，授权在用户通过 Sign in with Apple（使用苹果账号登录）进行身份验证后重定向用户。
e. 选择 Next（下一步）。
f. 选择 Continue（继续），然后选择 Register（注册）。

13. 在左侧导航窗格中，选择 Keys（密钥）。
14. 在 Keys（密钥）页面上，选择 + 图标。
15. 在 Register a New Key（注册新密钥）页面上，执行以下操作：

a. 在 Key Name（密钥名称）下方，键入密钥名称。
b. 选择 Sign In with Apple，然后选择 Configure（配置）。
c. 在 Configure Key（配置密钥）页面上，选择 Primary App ID（主应用程序 ID），然后选择 Save（保存）。
d. 选择 Continue（继续），然后选择 Register（注册）。

Note

要将 Sign in with Apple 与本机 iOS 应用程序集成，请参阅通过 Sign in with Apple 实施用户身份验证。
要在本机 iOS 以外的平台中集成 Sign in with Apple（使用苹果账号登录），请参阅使用 Apple JS 登录。

在 Amazon Cognito 联合身份控制台中配置外部提供商

使用以下过程可以配置外部提供商。

New console

添加通过 Apple 登录身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择通过 Apple 登录。
5. 输入您使用 Apple Developer 创建的 OAuth 项目的服务 ID。有关更多信息，请参阅通过 Apple 登录文档中的使用通过 Apple 登录对用户进行身份验证。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。

   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。

     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。
     ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。

7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。

   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。
   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。
8. 选择 Save changes（保存更改）。

Original console

1. 在 Amazon Cognito 控制台主页中选择 Manage Identity Pools（管理身份池）。
2. 选择要启用 Apple 作为外部提供商的身份池的名称。
3. 在控制面板的右上角，选择 Edit identity pool（编辑身份池）。
4. 向下滚动并选择 Authentication providers（身份验证提供商）以展开这一部分。
5. 选择 Apple 选项卡。
6. 输入从 https://developer.apple.com 获得的捆绑包 ID。然后选择 Save changes（保存更改）。
7. 如果您在本机 iOS 应用程序中使用 Sign in with Apple，请输入您从 developer.apple.com 获得的 BundleID。如果您在 Web 或其他应用程序中使用 Sign in with Apple（使用苹果账号登录），请输入服务 ID。然后选择 Save changes（保存更改）。

在 Amazon Cognito 联合身份 CLI 中以 Sign in with Apple 作为提供商的示例

此示例创建一个名为 MyIdentityPool 的身份池，并使用 Sign in with Apple（使用苹果账号登录）作为 IdP。

```
aws cognito-identity create-identity-pool --identity-pool-name MyIdentityPool
--supported-login-providers appleid.apple.com="sameple.apple.clientid"
```

有关更多信息，请参阅创建身份池

生成 Amazon Cognito 身份 ID

此示例生成（或检索）Amazon Cognito ID。这是一个公有 API，因此您不需要任何凭证即可调用此 API。

```
aws cognito-identity get-id --identity-pool-id SampleIdentityPoolId --logins
appleid.apple.com="SignInWithAppleIdToken"
```

有关更多信息，请参阅 get-id。

获取 Amazon Cognito 身份 ID 的凭证

此示例返回用于提供的身份 ID 和 Sign in with Apple 登录的凭证。这是一个公有 API，因此您不需要任何凭证即可调用此 API。

```
aws cognito-identity get-credentials-for-identity --identity-id
SampleIdentityId --logins appleid.apple.com="SignInWithAppleIdToken"
```

有关更多信息，请参阅 get-credentials-for-identity

使用 Sign in with Apple：Android

Apple 不提供支持 Sign in with Apple for Android 的开发工具包。您可以改为在 Web 视图中使用 Web 流。

- 要在应用程序中配置 Sign in with Apple，请按照 Apple 文档中的配置您的 Web 页面以使用 Sign in with Apple 操作。
- 要将 Sign in with Apple 按钮添加到 Android 用户界面，请按照 Apple 文档中的显示和配置 Sign in with Apple 按钮操作。
- 要使用 Sign in with Apple（使用苹果账号登录）安全地对用户进行身份验证，请按照 Apple 文档中的使用 Sign in with Apple（使用苹果账号登录）对用户进行身份验证操作。

Sign in with Apple 使用会话对象跟踪其状态。Amazon Cognito 使用来自此会话对象的 ID 令牌对用户进行身份验证、生成唯一标识符，并根据需要授予用户对其他AWS资源的访问权限。

```
@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 为原生 iOS 应用程序中的 Sign in with Apple 提供了开发工具包支持。要在本机 iOS 设备中使用 Sign in with Apple 实施用户身份验证，请按照 Apple 文档中的使用 Sign in with Apple 实施用户身份验证操作。

Amazon Cognito 使用 ID 令牌对用户进行身份验证、生成唯一标识符，并根据需要授予用户对其他 AWS 资源的访问权限。

```
(void)finishedWithAuth: (ASAuthorizationAppleIDCredential *)auth error: (NSError *) error {
        NSString *idToken = [ASAuthorizationAppleIDCredential
 objectForKey:@"identityToken"];
        credentialsProvider.logins = @{ "appleid.apple.com": idToken };
    }
```

所用 Sign in with Apple：iOS - Swift

Apple 为原生 iOS 应用程序中的 Sign in with Apple 提供了开发工具包支持。要在本机 iOS 设备中使用 Sign in with Apple 实施用户身份验证，请按照 Apple 文档中的使用 Sign in with Apple 实施用户身份验证操作。

Amazon Cognito 使用 ID 令牌对用户进行身份验证、生成唯一标识符，并根据需要授予用户对其他 AWS 资源的访问权限。

有关如何在 iOS 中设置 Sign in with Apple（使用苹果账号登录）的更多信息，请参阅设置 Sign in with Apple（使用苹果账号登录）

```
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 不提供支持 Sign in with Apple for JavaScript 的开发工具包。您可以改为在 Web 视图中使用 Web 流。

- 要在应用程序中配置 Sign in with Apple，请按照 Apple 文档中的配置您的 Web 页面以使用 Sign in with Apple 操作。
- 要将 Sign in with Apple 按钮添加到 JavaScript 用户界面，请按照 Apple 文档中的显示和配置 Sign in with Apple 按钮操作。
- 要通过 Sign in with Apple（使用苹果账号登录）安全地对用户进行身份验证，请按照 Apple 文档中的配置您的 Web 页面以使用苹果账号登录操作。

Sign in with Apple 使用会话对象跟踪其状态。Amazon Cognito 使用来自此会话对象的 ID 令牌对用户进行身份验证、生成唯一标识符，并根据需要授予用户对其他AWS资源的访问权限。

```
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.
     });
}
```

使用 Sign in with Apple：Xamarin

我们没有支持 Sign in with Apple for Xamarin 的开发工具包。您可以改为在 Web 视图中使用 Web 流。

- 要在应用程序中配置 Sign in with Apple，请按照 Apple 文档中的配置您的 Web 页面以使用 Sign in with Apple 操作。
- 要将 Sign in with Apple 按钮添加到 Xamarin 用户界面，请按照 Apple 文档中的显示和配置 Sign in with Apple 按钮操作。
- 要通过 Sign in with Apple（使用苹果账号登录）安全地对用户进行身份验证，请按照 Apple 文档中的配置您的 Web 页面以使用苹果账号登录操作。

Sign in with Apple 使用会话对象跟踪其状态。Amazon Cognito 使用来自此会话对象的 ID 令牌对用户进行身份验证、生成唯一标识符，并根据需要授予用户对其他AWS资源的访问权限。

拥有令牌后，您可以在 `CognitoAWSCredentials` 中对其进行设置：

```
credentials.AddLogin("appleid.apple.com", token);
```

OpenID Connect 提供程序（身份池）

OpenID Connect 是许多登录提供程序支持的身份验证开放标准。Amazon Cognito 支持您将身份与您通过 AWS Identity and Access Management 配置的 OpenID Connect 提供程序相关联。

添加 OpenID Connect 提供商

有关如何创建 OpenID Connect 提供者的信息，请参阅《AWS Identity and Access Management 用户指南》中的创建 OpenID Connect（OIDC）身份提供者。

将提供商与 Amazon Cognito 关联

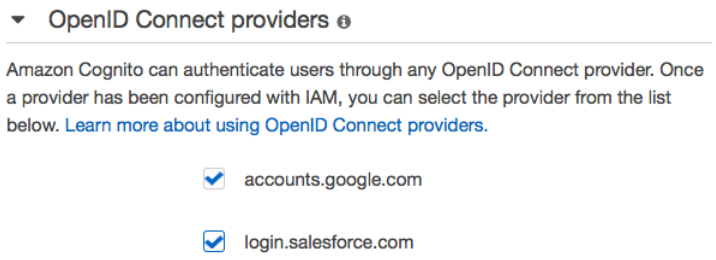
New console

添加 OIDC 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 OpenID Connect（OIDC）。
5. 从您的 AWS 账户的 IAM IdP 中选择 OIDC 身份提供者。如果您想添加新的 SAML 提供者，请选择创建新的提供者以导航到 IAM 控制台。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。
   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。
     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。
     ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。
7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。
   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。
   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。
8. 选择 Save changes（保存更改）。

Original console

在 IAM 控制台中创建 OpenID Connect 提供程序后，您可以将其与身份池关联。您配置的所有提供程序都将显示在 Amazon Cognito 控制台中 Edit Identity Pool（编辑身份池）屏幕上的 OpenID Connect Providers（OpenID Connect 提供程序）标题下。



您可以将多个 OpenID Connect 提供商与一个身份池关联。

使用 OpenID Connect

请参阅提供商的文档，了解如何登录并接收 ID 令牌。

拥有令牌后，将此令牌添加到登录映射。使用提供程序的 URI 作为密钥。

验证 OpenID Connect 令牌

首次与 Amazon Cognito 集成时，您可能会收到 `InvalidToken` 异常。务必要了解 Amazon Cognito 如何验证 OpenID Connect (OIDC) 令牌。

> **Note**
>
> 如此处 (https://tools.ietf.org/html/rfc7523) 所述，Amazon Cognito 留出 5 分钟的宽限期来处理系统之间的任何时钟偏差。

1. `iss` 参数必须与登录映射使用的密钥匹配（如 login.provider.com）。
2. 签名必须有效。签名必须可通过 RSA 公有密钥进行验证。
3. 证书公钥的指纹与您在创建 OIDC 提供程序时在 IAM 中设置的指纹相匹配。
4. 如果存在 `azp` 参数，请针对 OIDC 提供程序中列出的客户端 ID 检查此值。
5. 如果不存在 `azp` 参数，请针对 OIDC 提供程序中列出的客户端 ID 检查 `aud` 参数。

jwt.io 网站是用于解码令牌以验证这些值的宝贵资源。

Android

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("login.provider.com", token);
credentialsProvider.setLogins(logins);
```

iOS - Objective-C

```
credentialsProvider.logins = @{ "login.provider.com": token }
```

iOS - Swift

要向 Amazon Cognito 提供 OIDC ID 令牌，请实施 `AWSIdentityProviderManager` 协议。

在实现 `logins` 方法时，返回包含您配置的 OIDC 提供程序名称的词典。此词典充当键，而经过身份验证的用户的当前 ID 令牌充当值，如以下代码示例所示。

```
class OIDCProvider: NSObject, AWSIdentityProviderManager {
    func logins() -> AWSTask<NSDictionary> {
        let completion = AWSTaskCompletionSource<NSString>()
        getToken(tokenCompletion: completion)
        return completion.task.continueOnSuccessWith { (task) -> AWSTask<NSDictionary>? in
            //login.provider.name is the name of the OIDC provider as setup in the Amazon
 Cognito console
            return AWSTask(result:["login.provider.name":task.result!])
        } as! AWSTask<NSDictionary>

    }

    func getToken(tokenCompletion: AWSTaskCompletionSource<NSString>) -> Void {
        //get a valid oidc token from your server, or if you have one that hasn't expired
 cached, return it

        //TODO code to get token from your server
        //...

        //if error getting token, set error appropriately
        tokenCompletion.set(error:NSError(domain: "OIDC Login", code: -1 , userInfo:
 ["Unable to get OIDC token" : "Details about your error"]))
        //else
```

```
        tokenCompletion.set(result:"result from server id token")
    }
}
```

当您实例化 `AWSCognitoCredentialsProvider` 时，在构造函数中将实施 AWSIdentityProviderManager 的类作为 identityProviderManager 的值传递。有关更多信息，请转到 AWSCognitoCredentialsProvider 参考页面，并选择 initWithRegionType:identityPoolId:identityProviderManager。

JavaScript

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
 IdentityPoolId: 'IDENTITY_POOL_ID',
 Logins: {
    'login.provider.com': token
 }
});
```

Unity

```
credentials.AddLogin("login.provider.com", token);
```

Xamarin

```
credentials.AddLogin("login.provider.com", token);
```

SAML 身份提供商（身份池）

Amazon Cognito 支持通过身份提供商 (IdP) 并运用安全断言标记语言 2.0（SAML 2.0）进行身份验证。您可以使用支持 SAML 和 Amazon Cognito 的 IdP，为用户提供简单的引导流程。您支持 SAML 的 IdP 指定了用户可以承担的 IAM 角色。这样，不同的用户可以获得不同的权限集。

配置 SAML IdP 身份池

以下步骤介绍了如何配置身份池以使用基于 SAML 的 IdP。

> **Note**
>
> 在配置身份池以支持 SAML 提供商前，您必须先在 IAM 控制台中配置 SAML IdP。有关更多信息，请参阅 IAM 用户指南中的将第三方 SAML 解决方案提供商与 AWS 集成。

New console

添加 SAML 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 SAML。
5. 从您的 AWS 账户的 IAM IdP 中选择 SAML 身份提供者。如果您想添加新的 SAML 提供者，请选择创建新的提供者以导航到 IAM 控制台。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。

   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。

i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。

ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。

7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。

   a. 如果不应用主体标签，请选择非活动。

   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。

   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。

8. 选择 Save changes（保存更改）。

Original console

配置身份池以支持 SAML 提供商

1. 登录到 Amazon Cognito 控制台，选择 Manage Identity Pools（管理身份池），然后选择 Create new identity pool（创建新的身份池）。
2. 在Authentication providers（身份验证提供商）部分中，选择 SAML 选项卡。
3. 选择 SAML 提供商的 ARN，然后选择 Create Pool（创建池）。

配置 SAML IdP

创建 SAML 提供程序后，配置 SAML IdP，以在 IdP 和 AWS 之间添加信赖方信任。对于许多 IdP，您可以指定一个 URL，IdP 可以使用该 URL 从 XML 文档中读取依赖方信息和证书。对于 AWS，您可以使用 https://signin.aws.amazon.com/static/saml-metadata.xml。下一步是配置来自 IdP 的 SAML 断言响应，以填充 AWS 所需的声明。有关申请配置的详细信息，请参阅针对身份验证响应配置 SAML 断言。

使用 SAML 自定义用户角色

将 SAML 与 Amazon Cognito 身份结合使用时，可针对终端用户自定义角色。Amazon Cognito 只支持对基于 SAML 的 IdP 使用增强流程 (p. 300)。您无需为身份池指定经过身份验证或未经身份验证的角色，即可使用基于 SAML 的 IdP。`https://aws.amazon.com/SAML/Attributes/Role` 声明属性指定一个或多个逗号分隔的角色和提供商 ARN 对。这些是用户可以担任的角色。您可以配置 SAML IdP 以根据 IdP 提供的用户属性信息填充角色属性。如果您在 SAML 断言中收到多个角色，请在调用 `getCredentialsForIdentity` 时填充可选的 `customRoleArn` 参数。如果 `customRoleArn` 角色与 SAML 断言中的声明中的角色匹配，则用户将承担此角色。

使用 SAML IdP 对用户进行身份验证

要与基于 SAML 的 IdP 进行联合，请确定用户启动登录的 URL。AWS 联合使用 IdP 启动的登录。在 AD FS 2.0 中，URL 采用 `https://<fqdn>/adfs/ls/IdpInitiatedSignOn.aspx?loginToRp=urn:amazon:webservices` 格式。

要在 Amazon Cognito 中添加对 SAML IdP 的支持，请首先使用 SAML 身份提供商从 iOS 或 Android 应用程序对用户进行身份验证。您用于与 SAML IdP 集成和向其进行身份验证的代码特定于 SAML 提供商。对用户进行身份验证后，您可以使用 Amazon Cognito API 向 Amazon Cognito 身份提供生成的 SAML 断言。

您无法在身份池 API 请求的 `Logins` 映射中重复或重放 SAML 断言。重放的 SAML 断言的断言 ID 与早期 API 请求的 ID 重复。可以在 `Logins` 映射中接受 SAML 断言的 API 操作包括 GetId、GetCredentialsForIdentity、GetOpenIdToken 和 GetOpenIDTokenForDeveloperIdentity。您可以

在身份池身份验证流程中对于每个 API 请求重放 SAML 断言 ID 一次。例如，您可以在 GetId 请求和后续 GetCredentialsForIdentity 请求中提供相同的 SAML 断言，但不能在第二个 GetId 请求中提供相同的 SAML 断言。

Android

如果您使用 Android 开发工具包，则可以使用 SAML 断言填充登录映射，如下所示。

```
Map logins = new HashMap();
logins.put("arn:aws:iam::aws account id:saml-provider/name", "base64 encoded assertion
 response");
// Now this should be set to CognitoCachingCredentialsProvider object.
CognitoCachingCredentialsProvider credentialsProvider = new
 CognitoCachingCredentialsProvider(context, identity pool id, region);
credentialsProvider.setLogins(logins);
// If SAML assertion contains multiple roles, resolve the role by setting the custom role
credentialsProvider.setCustomRoleArn("arn:aws:iam::aws account id:role/customRoleName");
// This should trigger a call to the Amazon Cognito service to get the credentials.
credentialsProvider.getCredentials();
```

iOS

如果您使用的是 iOS 开发工具包，您可以在 AWSIdentityProviderManager 中提供 SAML 断言，如下所示。

```
- (AWSTask<NSDictionary<NSString*,NSString*> *> *) logins {
    //this is hardcoded for simplicity, normally you would asynchronously go to your SAML
 provider
    //get the assertion and return the logins map using a AWSTaskCompletionSource
    return [AWSTask taskWithResult:@{@"arn:aws:iam::aws account id:saml-provider/
name":@"base64 encoded assertion response"}];
}

// If SAML assertion contains multiple roles, resolve the role by setting the custom role.
// Implementing this is optional if there is only one role.
- (NSString *)customRoleArn {
    return @"arn:aws:iam::accountId:role/customRoleName";
}
```

经开发人员验证的身份（身份池）

除了通过 Facebook（身份池）(p. 330)、Google（身份池）(p. 339)、Login with Amazon（身份池）(p. 336) 和 Sign in with Apple（身份池）(p. 348) 的 Web 身份联合验证之外，Amazon Cognito 还支持经开发人员验证的身份。借助经开发人员验证的身份，您可以注册并通过自己现有的身份验证流程对用户进行身份验证，同时仍然使用 Amazon Cognito 同步用户数据并访问 AWS 资源。使用经开发人员验证的身份涉及最终用户设备、身份验证后端和 Amazon Cognito 之间的交互。有关更多详细信息，请参阅 AWS 博客中的了解 Amazon Cognito 身份验证第 2 部分：已经过开发人员验证的身份。

了解身份验证流程

有关经开发人员验证的身份的身份验证流程以及它与外部提供者身份验证流程的不同之处的信息，请参阅身份池（联合身份）身份验证流程 (p. 300)。

定义开发人员提供商名称并将其与身份池关联

要使用经开发人员验证的身份，您需要与开发人员提供者关联的身份池。为此，请按照以下步骤操作：

New console

添加自定义开发人员提供者

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择自定义开发人员提供者。
5. 输入开发人员提供者名称。添加开发人员提供者后，无法更改或删除它。
6. 选择 Save changes（保存更改）。

Original console

1. 登录 Amazon Cognito 控制台。
2. 创建新的身份池，并在该过程中，在 Authentication Providers（身份验证提供商）的 Custom（自定义）选项卡中定义开发人员提供商名称。
3. 或者，编辑现有身份池，并在 Authentication Providers（身份验证提供商）的 Custom（自定义）选项卡中定义开发人员提供商名称。

注意：一旦设置提供商名称，便无法进行更改。

有关使用 Amazon Cognito 控制台的更多说明，请参阅使用 Amazon Cognito 控制台 (p. 8)。

实施身份提供商

Android

要使用经开发人员验证的身份，请实施自己的身份提供者类，该类可扩展 `AWSAbstractCognitoIdentityProvider`。您的身份提供商类应返回包含令牌作为属性的响应对象。

以下是身份提供者的基本示例。

```
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;
    }

  }
}
```

要使用此身份提供商，您必须将其传递到 CognitoCachingCredentialsProvider。示例如下：

```
DeveloperAuthenticationProvider developerProvider = new
 DeveloperAuthenticationProvider( null, "IDENTITYPOOLID", context, Regions.USEAST1);
CognitoCachingCredentialsProvider credentialsProvider = new
 CognitoCachingCredentialsProvider( context, developerProvider, Regions.USEAST1);
```

iOS – objective-C

要使用经开发人员验证的身份，请实施自己的身份提供者类，该类可扩展 AWSCognitoCredentialsProviderHelper。您的身份提供商类应返回包含令牌作为属性的响应对象。

```
@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
```

要使用此身份提供商，请将其传递到 AWSCognitoCredentialsProvider，如下例所示：

```
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];
```

如果您想同时支持未经身份验证的身份和经开发人员验证的身份，请在 logins 实施中覆盖 AWSCognitoCredentialsProviderHelper 方法。

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else{
        return [super logins];
    }
}
```

如果您想支持经开发人员验证的身份和社交提供者，您必须管理在 AWSCognitoCredentialsProviderHelper 的 logins 实施中谁是当前的提供者。

```
- (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

要使用经开发人员验证的身份，请实施自己的身份提供者类，该类可扩展 AWSCognitoCredentialsProviderHelper。您的身份提供商类应返回包含令牌作为属性的响应对象。

```
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)
}
```

要使用此身份提供商，请将其传递到 AWSCognitoCredentialsProvider，如下例所示：

```
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
```

如果您想同时支持未经身份验证的身份和经开发人员验证的身份，请在 logins 实施中覆盖 AWSCognitoCredentialsProviderHelper 方法。

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else {
        return super.logins()
    }
}
```

如果您想支持经开发人员验证的身份和社交提供者，您必须管理在 AWSCognitoCredentialsProviderHelper 的 logins 实施中谁是当前的提供者。

```
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

从后端获取身份 ID 和会话令牌后，您要将它们传递到 AWS.CognitoIdentityCredentials 提供者。以下为示例。

```
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

要使用经开发人员验证的身份，您必须扩展 CognitoAWSCredentials 并覆盖 RefreshIdentity 方法，以从后端检索用户身份 ID 和令牌，并将它们返回。下面是可通过“example.com”联系假想后端的身份提供者的简单示例：

```
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);
    }
}
```

上面的代码使用线程调度程序对象调用协同程序。如果您在项目中无法执行上述操作，您可以在场景中使用以下脚本：

```
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

要使用经开发人员验证的身份，您必须扩展 `CognitoAWSCredentials` 并覆盖 `RefreshIdentity` 方法，以从后端检索用户身份 ID 和令牌，并将它们返回。下面是可通过“example.com”联系假想后端的身份提供者的基本示例：

```
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;
    }
}
```

更新登录映射（仅限 Android 和 iOS）

Android

使用身份验证系统成功对用户进行身份验证后，请使用开发人员提供者名称和开发人员用户标识符更新登录映射。此标识符是一个字母数字字符串，可在身份验证系统中唯一标识用户。请确保在更新登录映射后调用 `refresh` 方法，因为 `identityId` 可能已更改：

```
HashMap<String, String> loginsMap = new HashMap<String, String>();
```

```
loginsMap.put(developerAuthenticationProvider.getProviderName(), developerUserIdentifier);

credentialsProvider.setLogins(loginsMap);
credentialsProvider.refresh();
```

iOS – objective-C

如果没有凭证或者凭证已过期，则 iOS 开发工具包仅调用 `logins` 方法，以获取最新登录映射。如果您要强制 SDK 获取新的凭证（例如，最终用户从未经身份验证变为经过身份验证并且您想要经过身份验证的用户的凭证），则对 `credentialsProvider` 调用 `clearCredentials`。

```
[credentialsProvider clearCredentials];
```

iOS – swift

如果没有凭证或者凭证已过期，则 iOS 开发工具包仅调用 `logins` 方法，以获取最新登录映射。如果您要强制开发工具包获取新的凭证 (例如，最终用户从未经身份验证变为经过身份验证并且您想要经过身份验证的用户的凭证)，则在 `clearCredentials` 上调用 `credentialsProvider`。

```
credentialsProvider.clearCredentials()
```

获取令牌（服务器端）

您可通过调用 GetOpenIdTokenForDeveloperIdentity 获取令牌。您必须使用AWS开发人员凭证从后端调用此 API。不得从客户端开发工具包调用它。API 接收 Cognito 身份池 ID；包含身份提供者名称作为密钥及标识符作为值的登录映射；以及可选 Cognito 身份 ID（例如，您让一个未经过身份验证的用户变成了经身份验证的用户）。标识符可以是用户的用户名、电子邮件地址或数值。API 通过为用户提供唯一 Cognito ID 及为最终用户提供 OpenID Connect 令牌来响应您的调用。

对于由 `GetOpenIdTokenForDeveloperIdentity` 返回的令牌，您需要注意以下事项：

- 您可以指定令牌的自定义过期时间，以便缓存。如果您不提供任何自定义过期时间，则令牌的有效期为 15 分钟。
- 您可以设置的最大令牌持续时间为 24 小时。
- 请留意延长令牌持续时间所带来的安全方面的问题。如果攻击者获取了此令牌，则他们可以将令牌换成AWS凭证，供终端用户在令牌持续时间使用。

以下 Java 代码段显示了如何初始化 Amazon Cognito 客户端，以及如何检索经开发人员验证的身份的令牌。

```
// authenticate your end user as appropriate
// ....

// if authenticated, initialize a cognito client with your AWS developer credentials
AmazonCognitoIdentity identityClient = new AmazonCognitoIdentityClient(
  new BasicAWSCredentials("access_key_id", "secret_access_key")
);

// create a new request to retrieve the token for your end user
GetOpenIdTokenForDeveloperIdentityRequest request =
  new GetOpenIdTokenForDeveloperIdentityRequest();
request.setIdentityPoolId("YOUR_COGNITO_IDENTITY_POOL_ID");

request.setIdentityId("YOUR_COGNITO_IDENTITY_ID"); //optional, set this if your client has
 an
                                                   //identity ID that you want to link to
 this
```

```
                                                        //developer account

// set up your logins map with the username of your end user
HashMap<String,String> logins = new HashMap<>();
logins.put("YOUR_IDENTITY_PROVIDER_NAME","YOUR_END_USER_IDENTIFIER");
request.setLogins(logins);

// optionally set token duration (in seconds)
request.setTokenDuration(60 * 15l);
GetOpenIdTokenForDeveloperIdentityResult response =
  identityClient.getOpenIdTokenForDeveloperIdentity(request);

// obtain identity id and token to return to your client
String identityId = response.getIdentityId();
String token = response.getToken();

//code to return identity id and token to client
//...
```

按照上述步骤操作，您应该能够将经开发人员验证的身份集成到应用程序中。如有任何问题或疑问，请随时在我们的论坛上发帖。

连接到现有社交身份

当您使用经开发人员验证的身份时，您必须从后端链接所有提供者。要将自定义身份连接到用户的社交身份（Login with Amazon、Sign in with Apple、Facebook 或 Google），请在调用 GetOpenIdTokenForDeveloperIdentity 时将身份提供商令牌添加到登录映射。要实现上述目标，当您从客户端开发工具包调用后端来对最终用户进行身份验证时，您还需要传递最终用户的社交提供商令牌。

例如，如果您想将自定义身份链接到 Facebook，在调用 `GetOpenIdTokenForDeveloperIdentity` 时，除了身份提供商标识符之外，您还需要将 Facebook 令牌添加到登录映射。

```
logins.put("YOUR_IDENTITY_PROVIDER_NAME","YOUR_END_USER_IDENTIFIER");
logins.put("graph.facebook.com","END_USERS_FACEBOOK_ACCESSTOKEN");
```

支持在提供商之间转换

Android

您的应用程序可能需要支持未经身份验证的身份或使用公有提供者（Login with Amazon、通过 Apple 登录、Facebook 或 Google）的经过身份验证的身份，以及经开发人员验证的身份。经开发人员验证的身份与其他身份（未经身份验证的身份和使用公共提供者的经过身份验证的身份）的主要区别在于 identityId 和令牌的获取方式。对于其他身份，移动应用程序将直接与 Amazon Cognito 进行交互，而不是与身份验证系统联系。因此，移动应用程序应该能够支持两个不同的流程，具体取决于应用程序用户的选择。对此，您必须对自定义身份提供者做出一些更改。

`refresh` 方法检查登录映射。如果映射不为空并且有带开发人员提供者名称的密钥，请调用您的后端。否则，调用 getIdentityId 方法并返回 null。

```
public String refresh() {

   setToken(null);

   // If the logins map is not empty make a call to your backend
   // to get the token and identityId
   if (getProviderName() != null &&
      !this.loginsMap.isEmpty() &&
      this.loginsMap.containsKey(getProviderName())) {
```

```
      /**
       * This is where you would call your backend
       **/

      // now set the returned identity id and token in the provider
      update(identityId, token);
      return token;

   } else {
      // Call getIdentityId method and return null
      this.getIdentityId();
      return null;
   }
}
```

同样，getIdentityId 方法也有两个流程，具体取决于登录映射的内容：

```
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

您的应用程序可能需要支持未经身份验证的身份或使用公有提供者（Login with Amazon、通过 Apple 登录、Facebook 或 Google）的经过身份验证的身份，以及经开发人员验证的身份。为此，请覆盖 AWSCognitoCredentialsProviderHelper logins 方法，以便根据当前的身份提供商返回正确的登录映射。此示例说明如何能够在未经身份验证的身份、Facebook 和经开发人员验证的身份之间切换。

```
- (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];
    }
}
```

当您从未经身份验证转换为经过身份验证时，您应该调用 `[credentialsProvider clearCredentials];` 以强制开发工具包获取经过身份验证的新凭证。当您在两个经过身份验证的提供者之间切换并且不想将这两个提供者链接起来时（例如，您没有在登录词典中为多个提供者提供令牌），请调用 `[credentialsProvider clearKeychain];`。上述操作将清除凭证和身份，并强制开发工具包获取新的。

iOS – swift

您的应用程序可能需要支持未经身份验证的身份或使用公有提供者（Login with Amazon、通过 Apple 登录、Facebook 或 Google）的经过身份验证的身份，以及经开发人员验证的身份。为此，请覆盖 AWSCognitoCredentialsProviderHelper `logins` 方法，以便根据当前的身份提供商返回正确的登录映射。此示例说明如何能够在未经身份验证的身份、Facebook 和经开发人员验证的身份之间切换。

```
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()
    }
}
```

当您从未经身份验证转换为经过身份验证时，您应该调用 `credentialsProvider.clearCredentials()` 以强制开发工具包获取经过身份验证的新凭证。当您在两个经过身份验证的提供商之间切换并且不想将这两个提供商链接起来时 (即，您没有在登录词典中为多个提供商提供令牌)，您应该调用 `credentialsProvider.clearKeychain()`。上述操作将清除凭证和身份，并强制开发工具包获取新的。

Unity

您的应用程序可能需要支持未经身份验证的身份或使用公有提供者（Login with Amazon、通过 Apple 登录、Facebook 或 Google）的经过身份验证的身份，以及经开发人员验证的身份。经开发人员验证的身份与其他身份（未经身份验证的身份和使用公共提供者的经过身份验证的身份）的主要区别在于 identityId 和令牌的获取方式。对于其他身份，移动应用程序将直接与 Amazon Cognito 进行交互，而不是与身份验证系统联系。移动应用程序应该能够支持两个不同的流程，具体取决于应用程序用户的选择。对此，您必须对自定义身份提供商做出一些更改。

在 Unity 中，我们建议您从 AmazonCognitoEnhancedIdentityProvide 而不是 AbstractCognitoIdentityProvider 扩展身份提供商，并调用父 RefreshAsync 方法而不是您自己的方法，以防用户未通过您自己的后端进行身份验证。如果用户已经过身份验证，则您可以使用之前介绍的相同的流程。

Xamarin

您的应用程序可能需要支持未经身份验证的身份或使用公有提供者（Login with Amazon、通过 Apple 登录、Facebook 或 Google）的经过身份验证的身份，以及经开发人员验证的身份。经开发人员验证的身份与其他身份（未经身份验证的身份和使用公共提供者的经过身份验证的身份）的主要区别在于 identityId 和令牌的获取方式。对于其他身份，移动应用程序将直接与 Amazon Cognito 进行交互，而不是与身份验证系统联

系。移动应用程序应该能够支持两个不同的流程，具体取决于应用程序用户的选择。对此，您必须对自定义身份提供者做出一些更改。

将未经身份验证的用户切换为经过身份验证的用户（身份池）

Amazon Cognito 身份池同时支持经过身份验证的用户和未经身份验证的用户。即使未经身份验证的用户不通过任何身份提供商（IdP）登录，这些用户也有权访问您的AWS资源。此级别的访问可用于向尚未登录的用户显示内容。即使每个未经身份验证的用户尚未单独登录和经过身份验证，这些用户在身份池中也都具有唯一的身份。

本节介绍了用户如何选择从使用未经身份验证的身份登录切换为使用经过身份验证的身份登录。

Android

用户能够以未经身份验证的来宾的身份登录您的应用程序。最终，他们可能决定使用某个受支持的 IdP 登录。Amazon Cognito 将确保旧身份保留与新身份相同的唯一标识符，并确保配置文件数据自动合并。

应用程序会通过 `IdentityChangedListener` 界面收到配置文件合并的消息。在界面中实施 `identityChanged` 方法以接收这些消息：

```
@override
public void identityChanged(String oldIdentityId, String newIdentityId) {
    // handle the change
}
```

iOS – objective-C

用户能够以未经身份验证的来宾的身份登录您的应用程序。最终，他们可能决定使用某个受支持的 IdP 登录。Amazon Cognito 将确保旧身份保留与新身份相同的唯一标识符，并确保配置文件数据自动合并。

`NSNotificationCenter` 通知应用程序配置文件合并的消息：

```
[[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

用户能够以未经身份验证的来宾的身份登录您的应用程序。最终，他们可能决定使用某个受支持的 IdP 登录。Amazon Cognito 将确保旧身份保留与新身份相同的唯一标识符，并确保配置文件数据自动合并。

`NSNotificationCenter` 通知应用程序配置文件合并的消息：

```
[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

最初未经身份验证的用户

用户最初通常具有未经身份验证的角色。对于此角色，您可以设置配置对象的凭证属性，而不设置登录属性。在这种情况下，您的默认配置可能如下所示：

```
// set the default config object
var creds = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'us-east-1:1699ebc0-7900-4099-b910-2df94f52a030'
});
AWS.config.credentials = creds;
```

切换为经过身份验证的用户

当未经身份验证的用户登录 IdP 并且您拥有令牌时，您可以通过调用可更新凭证对象和添加登录令牌的自定义函数，来将用户从未经身份验证的用户切换为经过身份验证的用户：

```
// 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;
}
```

您还可以创建 `CognitoIdentityCredentials` 对象。在这种情况下，必须重置任何现有服务对象的凭证属性，以反映更新的凭证配置信息。请参阅使用全局配置对象。

有关 `CognitoIdentityCredentials` 对象的更多信息，请参阅 AWS SDK for JavaScript API 参考中的 AWS.CognitoIdentityCredentials。

Unity

用户能够以未经身份验证的来宾的身份登录您的应用程序。最终，他们可能决定使用某个受支持的 IdP 登录。Amazon Cognito 将确保旧身份保留与新身份相同的唯一标识符，并确保配置文件数据自动合并。

您可以订阅 `IdentityChangedEvent`，以接收配置文件合并的通知：

```
credentialsProvider.IdentityChangedEvent += delegate(object sender,
 CognitoAWSCredentials.IdentityChangedArgs e)
{
    // handle the change
```

```
    Debug.log("Identity changed from " + e.OldIdentityId + " to " + e.NewIdentityId);
};
```

Xamarin

用户能够以未经身份验证的来宾的身份登录您的应用程序。最终，他们可能决定使用某个受支持的 IdP 登录。Amazon Cognito 将确保旧身份保留与新身份相同的唯一标识符，并确保配置文件数据自动合并。

```
credentialsProvider.IdentityChangedEvent += delegate(object sender,
 CognitoAWSCredentials.IdentityChangedArgs e){
    // handle the change
    Console.WriteLine("Identity changed from " + e.OldIdentityId + " to " +
 e.NewIdentityId);
};
```

Amazon Cognito Sync

> 如果您是 Amazon Cognito Sync 的新用户，请使用 AWS AppSync。像 Amazon Cognito Sync 一样，AWS AppSync 是一种在设备之间同步应用程序数据的服务。
> 它允许同步用户数据，如应用程序首选项或游戏状态。它还通过允许多个用户实时同步和协作处理共享的数据，来扩展这些功能。

Amazon Cognito Sync 是一种 AWS 服务和客户端库，用于跨设备同步与应用程序相关的用户数据。Amazon Cognito 可以跨移动设备和 Web 同步用户配置文件数据，无需使用您自己的后端。客户端库在本地缓存数据，因此，您的应用程序可以读取和写入数据，无论设备是否处于连接状态，都是如此。设备处于在线状态时，您可以同步数据。如果您设置推送同步，您可在更新可用时立即通知其他设备。

有关 Amazon Cognito 身份区域可用性的信息，请参阅AWS服务区域可用性。

要了解有关 Amazon Cognito Sync 的更多信息，请参阅以下主题。

主题

- Amazon Cognito Sync 入门 (p. 371)
- 同步数据 (p. 372)
- 处理回调 (p. 378)
- 推送同步 (p. 389)
- Amazon Cognito Streams (p. 396)
- Amazon Cognito Events (p. 397)

Amazon Cognito Sync 入门

> 如果您是 Amazon Cognito Sync 的新用户，请使用 AWS AppSync。像 Amazon Cognito Sync 一样，AWS AppSync 是一种在设备之间同步应用程序数据的服务。
> 它允许同步用户数据，如应用程序首选项或游戏状态。它还通过允许多个用户实时同步和协作处理共享的数据，来扩展这些功能。

Amazon Cognito Sync 是一种AWS服务和客户端库，用于跨设备同步与应用程序相关的用户数据。您可以使用它来跨移动设备和 Web 应用程序同步用户配置文件数据。客户端库在本地缓存数据，因此，您的应用程序可以读取和写入数据，无论设备是否处于连接状态，都是如此。设备处于在线状态时，您可以同步数据，如果您设置推送同步，则在更新可用时会立即通知其他设备。

设置 Amazon Cognito 中的身份池

Amazon Cognito Sync 需要一个 Amazon Cognito 身份池来提供用户身份。您需要先设置身份池，然后才能使用 Amazon Cognito Sync。要创建身份池并安装开发工具包，请参阅Amazon Cognito 身份池入门（联合身份）(p. 289)。

存储和同步数据

设置身份池并安装开发工具包后，您就可以开始存储并在设备之间同步数据了。有关更多信息，请参阅同步数据 (p. 372)。

同步数据

> 如果您是 Amazon Cognito Sync 的新用户，请使用 AWS AppSync。像 Amazon Cognito Sync 一样，AWS AppSync 是一种在设备之间同步应用程序数据的服务。
> 它允许同步用户数据，如应用程序首选项或游戏状态。它还通过允许多个用户实时同步和协作处理共享的数据，来扩展这些功能。

借助 Amazon Cognito，您可将用户数据保存在包含键/值对的数据集中。Amazon Cognito 将此数据与您身份池中的身份相关联，这样您的应用程序就可以跨登录和设备访问它。要在 Amazon Cognito 服务和终端用户设备之间同步此数据，请调用 synchronize 方法。每个数据集的最大大小为 1MB。您最多可以将 20 个数据集与一个身份关联。

Amazon Cognito Sync 客户端会为身份数据创建一个本地缓存。您的应用程序在读取和写入键时，它与此本地缓存通信。此通信保证您在设备上所做的所有更改都能在设备上立即可用，即使您处于离线状态也是如此。调用 synchronize 方法后，服务的更改将拉取到设备上，并且所有本地更改都会推送到该服务。此时，这些更改可供其他设备同步。

初始化 Amazon Cognito Sync 客户端

要初始化 Amazon Cognito Sync 客户端，您必须先创建凭证提供程序。凭证提供程序获取临时 AWS 凭证，以使应用程序能够访问 AWS 资源。您还必须导入必要的标头文件。使用以下步骤初始化 Amazon Cognito Sync 客户端。

Android

1. 按照获取凭证 (p. 322)中的说明创建凭证提供程序。
2. 如下所示导入 Amazon Cognito 程序包：`import com.amazonaws.mobileconnectors.cognito.*;`
3. 初始化 Amazon Cognito Sync。传入 Android 应用程序上下文、身份池 ID、AWS 区域和初始化的 Amazon Cognito 凭证提供程序，如下所示：

```
CognitoSyncManager client = new CognitoSyncManager(
    getApplicationContext(),
    Regions.YOUR_REGION,
    credentialsProvider);
```

iOS - Objective-C

1. 按照获取凭证 (p. 322)中的说明创建凭证提供程序。
2. 导入 `AWSCore` 和 `Cognito`，并初始化 `AWSCognito`如下所示：

```
#import <AWSiOSSDKv2/AWSCore.h>
#import <AWSCognitoSync/Cognito.h>

AWSCognito *syncClient = [AWSCognito defaultCognito];
```

3. 如果您正在使用 CocoaPods，请将 `<AWSiOSSDKv2/AWSCore.h>` 替换为 `AWSCore.h`。请遵循与 Amazon Cognito 导入相同的语法。

iOS - Swift

1. 按照获取凭证 (p. 322)中的说明创建凭证提供程序。

2. 导入并初始化 AWSCognito，如下所示：

```
import AWSCognito
let syncClient = AWSCognito.default()!
```

JavaScript

1. 下载 Amazon Cognito Sync Manager for JavaScript。
2. 在您的项目中添加 Sync Manager 库。
3. 按照获取凭证 (p. 322)中的说明创建凭证提供程序。
4. 如下所示初始化 Sync Manager：

```
var syncManager = new AWS.CognitoSyncManager();
```

Unity

1. 按照获取凭证 (p. 322)中的说明创建 CognitoAWSCredentials 的实例。
2. 创建 CognitoSyncManager 的实例。传递 CognitoAwsCredentials 对象和 AmazonCognitoSyncConfig，并至少包括区域集，如下所示：

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =
 REGION };
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

Xamarin

1. 按照 获取凭证 (p. 322) 中的说明创建 CognitoAWSCredentials 的实例。
2. 创建 CognitoSyncManager 的实例。传递 CognitoAwsCredentials 对象和 AmazonCognitoSyncConfig，并至少包括区域集，如下所示：

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =
 REGION };
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

了解数据集

Amazon Cognito 将用户配置文件数据组织到数据集中。每个数据集可以包含高达 1MB 的键值对形式的数据。数据集是您可以同步的最细粒度实体。在调用 synchronize 方法之前，在数据集上执行的读取和写入操作只会对本地存储产生影响。Amazon Cognito 使用唯一字符串标识数据集。您可以创建新数据集或打开现有数据集，如下所示。

Android

```
Dataset dataset = client.openOrCreateDataset("datasetname");
```

要删除数据集，首先要调用方法以将其从本地存储中删除，然后调用 synchronize 方法从 Amazon Cognito 中删除数据集，如下所示：

```
dataset.delete();
dataset.synchronize(syncCallback);
```

iOS - Objective-C

```
AWSCognitoDataset *dataset = [syncClient openOrCreateDataset:@"myDataSet"];
```

要删除数据集，首先要调用方法以将其从本地存储中删除，然后调用 synchronize 方法从 Amazon Cognito 中删除数据集，如下所示：

```
[dataset clear];
[dataset synchronize];
```

iOS - Swift

```
let dataset = syncClient.openOrCreateDataset("myDataSet")!
```

要删除数据集，首先要调用方法以将其从本地存储中删除，然后调用 synchronize 方法如下所示以从 Amazon Cognito 中删除数据集：

```
dataset.clear()
dataset.synchronize()
```

JavaScript

```
syncManager.openOrCreateDataset('myDatasetName', function(err, dataset) {
   // ...
});
```

Unity

```
string myValue = dataset.Get("myKey");
dataset.Put("myKey", "newValue");
```

要从数据集中删除键，请如下所示使用 Remove：

```
dataset.Remove("myKey");
```

Xamarin

```
Dataset dataset = syncManager.OpenOrCreateDataset("myDatasetName");
```

要删除数据集，首先要调用方法以将其从本地存储中删除，然后调用 synchronize 方法从 Amazon Cognito 中删除数据集，如下所示：

```
dataset.Delete();
dataset.SynchronizeAsync();
```

在数据集中读取并写入数据

Amazon Cognito 数据集的功能与字典一样，包含可以通过键访问的值。您可以读取、添加或修改数据集的键和值，就像数据集是字典一样，如下所示。

请注意，在您调用同步方法之前，写入数据集的值仅对本地缓存的数据副本有影响。

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

要从数据集中删除键，请如下所示使用 remove 方法：

```
dataset.remove("myKey");
```

iOS - Objective-C

要从数据集中删除键，请如下所示使用 removeObjectForKey：

```
[dataset removeObjectForKey:@"myKey"];
```

iOS - Swift

要从数据集中删除键，请如下所示使用 removeObjectForKey：

```
dataset.removeObjectForKey("myKey")
```

Unity

要从数据集中删除键，请如下所示使用 Remove：

```
dataset.Remove("myKey");
```

Xamarin

您可以使用 Remove 从数据集中删除键：

```
dataset.Remove("myKey");
```

使用同步存储同步本地数据

Android

synchronize 方法将本地缓存的数据与存储在 Amazon Cognito Sync 存储空间中的数据进行比较。从 Amazon Cognito Sync 存储空间中拉取远程更改；如果出现任何冲突，则调用冲突解决方法；设备上的更新值将推送到该服务。要同步数据集，请调用其 synchronize 方法：

```
dataset.synchronize(syncCallback);
```

synchronize 方法收到 SyncCallback 接口的实现，如下所述。

synchronizeOnConnectivity() 方法尝试在连接可用时进行同步。如果连接立即可用，则 synchronizeOnConnectivity() 的行为类似于 synchronize()。否则，它会监控连接更改，并在连接可用时立即执行同步。如果多次调用 synchronizeOnConnectivity()，则只会保持最近一次同步请求，并只会触发最近一次回调。如果数据集或回调收集到垃圾，则此方法不会执行同步，且不会触发回调。

要了解有关数据集同步和不同回调的更多信息，请参阅处理回调 (p. 378)。

iOS - Objective-C

synchronize 方法将本地缓存的数据与存储在 Amazon Cognito Sync 存储空间中的数据进行比较。从 Amazon Cognito Sync 存储空间中拉取远程更改；如果出现任何冲突，则调用冲突解决方法；设备上的更新值将推送到该服务。要同步数据集，请调用其 synchronize 方法：

synchronize 方法是异步的，它会返回 AWSTask 对象以处理响应：

```
[[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 方法尝试在设备具备连接时进行同步。首先，synchronizeOnConnectivity 将检查连接状态，如果设备处于在线状态，则立即调用 synchronize，并返回与此次尝试关联的 AWSTask 对象。

如果设备处于离线状态，synchronizeOnConnectivity 1) 计划在设备下次变为在线状态时进行同步；2) 返回一个无结果的 AWSTask。计划的同步仅在该数据集对象的生命周期内有效。如果在连接恢复之前退出应用程序，则数据不会进行同步。如果您希望在计划同步期间发生事件时收到通知，则必须添加在 AWSCognito 中找到的通知的观察者。

要了解有关数据集同步和不同回调的更多信息，请参阅处理回调 (p. 378)。

iOS - Swift

synchronize 方法将本地缓存的数据与存储在 Amazon Cognito Sync 存储空间中的数据进行比较。从 Amazon Cognito Sync 存储空间中拉取远程更改；如果出现任何冲突，则调用冲突解决方法；设备上的更新值将推送到该服务。要同步数据集，请调用其 synchronize 方法：

synchronize 方法是异步的，它会返回 AWSTask 对象以处理响应：

```
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 方法尝试在设备具备连接时进行同步。首先，synchronizeOnConnectivity 将检查连接状态，如果设备处于在线状态，则立即调用 synchronize，并返回与此次尝试关联的 AWSTask 对象。

如果设备处于离线状态，synchronizeOnConnectivity 1) 计划在设备下次变为在线状态时进行同步；2) 返回一个无结果的 AWSTask 对象。计划的同步仅在该数据集对象的生命周期内有效。如果在连接恢复之

前退出应用程序，则数据不会进行同步。如果您希望在计划同步期间发生事件时收到通知，则必须添加在 AWSCognito 中找到的通知的观察者。

要了解有关数据集同步和不同回调的更多信息，请参阅处理回调 (p. 378)。

JavaScript

synchronize 方法将本地缓存的数据与存储在 Amazon Cognito Sync 存储空间中的数据进行比较。从 Amazon Cognito Sync 存储空间中拉取远程更改；如果出现任何冲突，则调用冲突解决方法；设备上的更新值将推送到该服务。要同步数据集，请调用其 synchronize 方法：

```
dataset.synchronize();
```

要了解有关数据集同步和不同回调的更多信息，请参阅处理回调 (p. 378)。

Unity

synchronize 方法将本地缓存的数据与存储在 Amazon Cognito Sync 存储空间中的数据进行比较。从 Amazon Cognito Sync 存储空间中拉取远程更改；如果出现任何冲突，则调用冲突解决方法；设备上的更新值将推送到该服务。要同步数据集，请调用其 synchronize 方法：

```
dataset.Synchronize();
```

同步将以异步方式运行，最终会调用您可以在数据集中指定的几个回调之一。

要了解有关数据集同步和不同回调的更多信息，请参阅处理回调 (p. 378)。

Xamarin

synchronize 方法将本地缓存的数据与存储在 Amazon Cognito Sync 存储空间中的数据进行比较。从 Amazon Cognito Sync 存储空间中拉取远程更改；如果出现任何冲突，则调用冲突解决方法；设备上的更新值将推送到该服务。要同步数据集，请调用其 synchronize 方法：

```
dataset.SynchronizeAsync();
```

要了解有关数据集同步和不同回调的更多信息，请参阅处理回调 (p. 378)。

处理回调

> 如果您是 Amazon Cognito Sync 的新用户，请使用 AWS AppSync。像 Amazon Cognito Sync 一样，AWS AppSync 是一种在设备之间同步应用程序数据的服务。
> 它允许同步用户数据，如应用程序首选项或游戏状态。它还通过允许多个用户实时同步和协作处理共享的数据，来扩展这些功能。

本部分介绍如何处理回调。

Android

SyncCallback 接口

通过实施 SyncCallback 接口，您可以在应用程序上接收有关数据集同步的通知。然后，您的应用程序可以在删除本地数据、合并未经身份验证的和经过身份验证的配置文件以及解决同步冲突方面制定有效决策。您应该实施接口所需的以下方法：

- onSuccess()
- onFailure()
- onConflict()
- onDatasetDeleted()
- onDatasetsMerged()

请注意，如果您不想指定所有回调，也可以使用类 DefaultSyncCallback，它会为所有回调提供默认的空实施。

onSuccess

从同步存储空间成功下载数据集后，将触发 onSuccess() 回调。

```
@Override
public void onSuccess(Dataset dataset, List<Record> newRecords) {
}
```

onFailure

如果同步过程中出现异常，则会调用 onFailure()。

```
@Override
public void onFailure(DataStorageException dse) {
}
```

onConflict

如果在本地存储和同步存储空间修改同一键，则会产生冲突。onConflict() 方法可处理冲突解决方法。如果您没有实施此方法，则 Amazon Cognito Sync 客户端将默认为使用最近的更改。

```
@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

删除数据集后，Amazon Cognito 客户端将使用 SyncCallback 接口确认是否还应删除本地缓存的数据集副本。实施 onDatasetDeleted() 方法，以告知客户端开发工具包该如何处理本地数据。

```
@Override
```

```
public boolean onDatasetDeleted(Dataset dataset, String datasetName) {
    // return true to delete the local copy of the dataset
    return true;
}
```

onDatasetMerged

当两个以前未连接的身份链接在一起后，它们的所有数据集都将合并。可通过 onDatasetsMerged() 方法向应用程序发送有关合并的通知：

```
@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

在同步调用过程中，Amazon Cognito 客户端将发出大量 NSNotification 事件。您可以进行注册，以通过标准 NSNotificationCenter 监控这些通知：

```
[NSNotificationCenter defaultCenter]
  addObserver:self
  selector:@selector(myNotificationHandler:)
  name:NOTIFICATION_TYPE
  object:nil];
```

Amazon Cognito 支持五种通知类型，如下所列。

AWSCognitoDidStartSynchronizeNotification

同步操作开始时调用。userInfo 包含主数据集，即正在进行同步的数据集的名称。

AWSCognitoDidEndSynchronizeNotification

在同步操作完成 (无论是否成功) 时调用。userInfo 包含主数据集，即正在进行同步的数据集的名称。

AWSCognitoDidFailToSynchronizeNotification

同步操作失败时调用。userInfo 包含主数据集 (即正在进行同步的数据集的名称) 和关键错误 (包含导致失败的错误)。

AWSCognitoDidChangeRemoteValueNotification

本地更改成功推送到 Amazon Cognito 时调用。userInfo 包含主数据集 (即正在进行同步的数据集的名称) 和关键键 (包含已推送的记录键的 NSArray)。

AWSCognitoDidChangeLocalValueFromRemoteNotification

本地值由于同步操作而更改时调用。userInfo 包含主数据集 (即正在进行同步的数据集的名称) 和关键键 (包含已更改的记录键的 NSArray)。

冲突解决方法处理程序

在同步操作过程中，如果在本地存储和同步存储空间修改同一键，则会产生冲突。如果您尚未设置冲突解决方法处理程序，则 Amazon Cognito 将默认为选择最近的更新。

通过实施和分配 AWSCognitoRecordConflictHandler，您可以更改默认的冲突解决方法。AWSCognitoConflict 输入参数冲突包含本地缓存数据和同步存储空间中冲突记录的 AWSCognitoRecord 对象。使用 AWSCognitoConflict，您可以通过 [conflict resolveWithLocalRecord] 解决与本地记录的冲突，通过 [conflict resolveWithRemoteRecord] 解决与远程记录的冲突，或通过 [conflict resolveWithValue:value] 解决与全新值的冲突。此方法返回无将阻止同步继续进行，并且下一次同步过程开始时会再次出现冲突。

您可以在客户端层面设置冲突解决方法处理程序：

```
client.conflictHandler = ^AWSCognitoResolvedConflict* (NSString *datasetName,
 AWSCognitoConflict *conflict) {
    // always choose local changes
    return [conflict resolveWithLocalRecord];
};
```

或在数据集层面：

```
dataset.conflictHandler = ^AWSCognitoResolvedConflict* (NSString *datasetName,
 AWSCognitoConflict *conflict) {
    // override and always choose remote changes
    return [conflict resolveWithRemoteRecord];
};
```

数据集删除处理程序

删除数据集后，Amazon Cognito 客户端将使用 AWSCognitoDatasetDeletedHandler 来确认是否还应删除本地缓存的数据集副本。如果未实施 AWSCognitoDatasetDeletedHandler，则将自动清除本地数据。如果您希望在清除前保留本地数据的副本或保留本地数据，则实施 AWSCognitoDatasetDeletedHandler。

您可以在客户端层面设置数据集删除处理程序：

```
client.datasetDeletedHandler = ^BOOL (NSString *datasetName) {
    // make a backup of the data if you choose
    ...
    // delete the local data (default behavior)
    return YES;
};
```

或在数据集层面：

```
dataset.datasetDeletedHandler = ^BOOL (NSString *datasetName) {
    // override default and keep the local data
    return NO;
};
```

数据集合并处理程序

当两个以前未连接的身份链接在一起后，它们的所有数据集都将合并。可通过 DatasetMergeHandler 向应用程序发送有关合并的通知。该处理程序将收到根数据集名称以及标记为根数据集合并的数据集名称数组。

如果未实施 DatasetMergeHandler，这些数据集将被忽略，但将继续占用最多 20 个身份总数据集中的空间。

您可以在客户端层面设置数据集合并处理程序：

```
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];
    }
};
```

或在数据集层面：

```
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

在同步调用过程中，Amazon Cognito 客户端将发出大量 NSNotification 事件。您可以进行注册，以通过标准 NSNotificationCenter 监控这些通知：

```
NSNotificationCenter.defaultCenter().addObserver(observer: self,
   selector: "myNotificationHandler",
   name:NOTIFICATION_TYPE,
   object:nil)
```

Amazon Cognito 支持五种通知类型，如下所列。

AWSCognitoDidStartSynchronizeNotification

同步操作开始时调用。userInfo 包含主数据集，即正在进行同步的数据集的名称。

AWSCognitoDidEndSynchronizeNotification

在同步操作完成 (无论是否成功) 时调用。userInfo 包含主数据集，即正在进行同步的数据集的名称。

AWSCognitoDidFailToSynchronizeNotification

同步操作失败时调用。userInfo 包含主数据集 (即正在进行同步的数据集的名称) 和关键错误 (包含导致失败的错误)。

AWSCognitoDidChangeRemoteValueNotification

本地更改成功推送到 Amazon Cognito 时调用。userInfo 包含主数据集 (即正在进行同步的数据集的名称) 和关键键 (包含已推送的记录键的 NSArray)。

AWSCognitoDidChangeLocalValueFromRemoteNotification

本地值由于同步操作而更改时调用。userInfo 包含主数据集 (即正在进行同步的数据集的名称) 和关键键 (包含已更改的记录键的 NSArray)。

冲突解决方法处理程序

在同步操作过程中，如果在本地存储和同步存储空间修改同一键，则会产生冲突。如果您尚未设置冲突解决方法处理程序，则 Amazon Cognito 将默认为选择最近的更新。

通过实施和分配 AWSCognitoRecordConflictHandler，您可以更改默认的冲突解决方法。AWSCognitoConflict 输入参数冲突包含本地缓存数据和同步存储空间中冲突记录的 AWSCognitoRecord 对象。使用 AWSCognitoConflict，您可以通过 [conflict resolveWithLocalRecord] 解决与本地记录的冲突，通过 [conflict resolveWithRemoteRecord] 解决与远程记录的冲突，或通过 [conflict resolveWithValue:value] 解决与全新值的冲突。此方法返回无将阻止同步继续进行，并且下一次同步过程开始时会再次出现冲突。

您可以在客户端层面设置冲突解决方法处理程序：

```
client.conflictHandler = {
    (datasetName: String?, conflict: AWSCognitoConflict?) -> AWSCognitoResolvedConflict? in
    return conflict.resolveWithLocalRecord()
}
```

或在数据集层面：

```
dataset.conflictHandler = {
    (datasetName: String?, conflict: AWSCognitoConflict?) -> AWSCognitoResolvedConflict? in
    return conflict.resolveWithLocalRecord()
}
```

数据集删除处理程序

删除数据集后，Amazon Cognito 客户端将使用 AWSCognitoDatasetDeletedHandler 来确认是否还应删除本地缓存的数据集副本。如果未实施 AWSCognitoDatasetDeletedHandler，则将自动清除本地数据。如果您希望在清除前保留本地数据的副本或保留本地数据，则实施 AWSCognitoDatasetDeletedHandler。

您可以在客户端层面设置数据集删除处理程序：

```
client.datasetDeletedHandler = {
      (datasetName: String!) -> Bool in
      // make a backup of the data if you choose
      ...
      // delete the local data (default behaviour)
      return true
}
```

或在数据集层面：

```
dataset.datasetDeletedHandler = {
      (datasetName: String!) -> Bool in
      // make a backup of the data if you choose
      ...
      // delete the local data (default behaviour)
      return true
}
```

数据集合并处理程序

当两个以前未连接的身份链接在一起后，它们的所有数据集都将合并。可通过 DatasetMergeHandler 向应用程序发送有关合并的通知。该处理程序将收到根数据集名称以及标记为根数据集合并的数据集名称数组。

如果未实施 DatasetMergeHandler，这些数据集将被忽略，但将继续占用最多 20 个身份总数据集中的空间。

您可以在客户端层面设置数据集合并处理程序：

```
client.datasetMergedHandler = {
    (datasetName: String!, datasets: [AnyObject]!) -> Void in
    for nameObject in datasets {
        if let name = nameObject as? String {
            let merged = AWSCognito.defaultCognito().openOrCreateDataset(name)
            merged.clear()
            merged.synchronize()
        }
    }
}
```

或在数据集层面：

```
dataset.datasetMergedHandler = {
    (datasetName: String!, datasets: [AnyObject]!) -> Void in
    for nameObject in datasets {
        if let name = nameObject as? String {
            let merged = AWSCognito.defaultCognito().openOrCreateDataset(name)
            // do something with the data if it differs from existing dataset
              ...
              // now delete it
            merged.clear()
            merged.synchronize()
        }
    }
}
```

JavaScript

同步回调

在数据集上执行 synchronize() 时，您可以有选择性地指定用于处理以下各种状态的回调：

```
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()

从同步存储空间成功更新数据集后，将触发 onSuccess() 回调。如果您未定义回调，则同步成功完成后将保持静默状态。

```
onSuccess: function(dataset, newRecords) {
   console.log('Successfully synchronized ' + newRecords.length + ' new records.');
}
```

onFailure()

如果同步过程中出现异常，则会调用 onFailure()。如果您未定义回调，则同步失败后将无提示。

```
onFailure: function(err) {
   console.log('Synchronization failed.');
   console.log(err);
}
```

onConflict()

如果在本地存储和同步存储空间修改同一键，则会产生冲突。onConflict() 方法可处理冲突解决方法。如果您未实施此方法，则在发生冲突时，同步将中止。

```
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()

删除数据集后，Amazon Cognito 客户端将使用 onDatasetDeleted() 回调决定是否还应删除本地缓存的数据集副本。默认情况下，不会删除该数据集。

```
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()

当两个以前未连接的身份链接在一起后，它们的所有数据集都将合并。可通过 onDatasetsMerged() 回调向应用程序发送有关合并的通知。

```
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

打开或创建数据集之后，您可以为其设置不同的回调，以在使用 Synchronize 方法时触发。以下就是将回调注册到数据集的方法：

```
dataset.OnSyncSuccess += this.HandleSyncSuccess;
dataset.OnSyncFailure += this.HandleSyncFailure;
dataset.OnSyncConflict = this.HandleSyncConflict;
dataset.OnDatasetMerged = this.HandleDatasetMerged;
dataset.OnDatasetDeleted = this.HandleDatasetDeleted;
```

请注意，SyncSuccess 和 SyncFailure 使用 += 而不是 =，因此您可以向其订阅多个回调。

OnSyncSuccess

从云成功更新数据集后，将触发 OnSyncSuccess 回调。如果您未定义回调，则同步成功完成后将保持静默状态。

```
private void HandleSyncSuccess(object sender, SyncSuccessEvent e)
{
    // Continue with your game flow, display the loaded data, etc.
}
```

OnSyncFailure

如果同步过程中出现异常，则会调用 OnSyncFailure。如果您未定义回调，则同步失败后将无提示。

```
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

如果在本地存储和同步存储空间修改同一键，则会产生冲突。OnSyncConflict 回调可处理冲突解决方法。如果您未实施此方法，则在发生冲突时，同步将中止。

```
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

删除数据集后，Amazon Cognito 客户端将使用 OnDatasetDeleted 回调决定是否还应删除本地缓存的数据集副本。默认情况下，不会删除该数据集。

```
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

当两个以前未连接的身份链接在一起后，它们的所有数据集都将合并。可通过 OnDatasetsMerged 回调向应用程序发送有关合并的通知。

```
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

打开或创建数据集之后，您可以为其设置不同的回调，以在使用 Synchronize 方法时触发。以下就是将回调注册到数据集的方法：

```
dataset.OnSyncSuccess += this.HandleSyncSuccess;
dataset.OnSyncFailure += this.HandleSyncFailure;
dataset.OnSyncConflict = this.HandleSyncConflict;
dataset.OnDatasetMerged = this.HandleDatasetMerged;
dataset.OnDatasetDeleted = this.HandleDatasetDeleted;
```

请注意，SyncSuccess 和 SyncFailure 使用 += 而不是 =，因此您可以向其订阅多个回调。

OnSyncSuccess

从云成功更新数据集后，将触发 OnSyncSuccess 回调。如果您未定义回调，则同步成功完成后将保持静默状态。

```
private void HandleSyncSuccess(object sender, SyncSuccessEventArgs e)
{
    // Continue with your game flow, display the loaded data, etc.
}
```

OnSyncFailure

如果同步过程中出现异常，则会调用 OnSyncFailure。如果您未定义回调，则同步失败后将无提示。

```
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

如果在本地存储和同步存储空间修改同一键，则会产生冲突。OnSyncConflict 回调可处理冲突解决方法。如果您未实施此方法，则在发生冲突时，同步将中止。

```
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

删除数据集后，Amazon Cognito 客户端将使用 `OnDatasetDeleted` 回调决定是否还应删除本地缓存的数据集副本。默认情况下，不会删除该数据集。

```
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

当两个以前未连接的身份链接在一起后，它们的所有数据集都将合并。可通过 `OnDatasetsMerged` 回调向应用程序发送有关合并的通知。

```
public bool HandleDatasetMerged(Dataset localDataset, List<string> mergedDatasetNames)
{
    foreach (string name in mergedDatasetNames)
    {
        Dataset mergedDataset = syncManager.OpenOrCreateDataset(name);

            //Implement your merge logic here

        mergedDataset.OnSyncSuccess += lambda;
        mergedDataset.SynchronizeAsync(); //Asnchronously fetch the dataset
    }

    // returning true allows the Synchronize to continue and false stops it
    return false;
}
```

推送同步

如果您是 Amazon Cognito Sync 的新用户，请使用 AWS AppSync。像 Amazon Cognito Sync 一样，AWS AppSync 是一种在设备之间同步应用程序数据的服务。

它允许同步用户数据，如应用程序首选项或游戏状态。它还通过允许多个用户实时同步和协作处理共享的数据，来扩展这些功能。

Amazon Cognito 会自动跟踪身份和设备之间的关联。使用推送同步功能，您可以确保在身份数据发生更改后向给定身份的每个实例发送通知。推送同步可以确保，无论特定身份的同步存储数据何时发生更改，与该身份关联的所有设备都会收到一个静音推送通知，通知它们所发生的更改。

Note

推送同步不支持 JavaScript、Unity 或 Xamarin。

您必须首先设置用于推送同步的账户，并在 Amazon Cognito 控制台中启用推送同步，然后才可使用推送同步。

创建 Amazon Simple Notification Service（Amazon SNS）应用程序

为支持的平台创建并配置 Amazon SNS 应用程序，如 SNS 开发人员指南中所述。

在 Amazon Cognito 控制台中启用推送同步

您可以通过 Amazon Cognito 控制台启用推送同步。从控制台主页：

1. 单击您需要启用推送同步的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在Dashboard（控制面板）页的右上角，单击 Manage Identity Pools（管理身份池）。此时将显示 Federated Identities（联合身份）页。
3. 向下滚动并单击 Push synchronization（推送同步）以将其展开。
4. 在 Service role（服务角色）下拉菜单中，选择授予 Cognito 发送 SNS 通知的权限的 IAM 角色。在 AWS IAM 控制台中，单击 Create role (创建角色) 以创建或修改与您身份池关联的角色。
5. 选择一个平台应用程序，然后单击 Save Changes（保存更改）。
6. 为应用程序授予 SNS 访问权限

在 AWS Identity and Access Management 控制台中，将您的 IAM 角色配置为具有完整 Amazon SNS 访问权限，或创建一个具有完整 Amazon SNS 访问权限的新角色。以下示例角色信任策略授予 Amazon Cognito Sync 有限代入 IAM 角色的能力。Amazon Cognito Sync 只能在代表 `aws:SourceArn` 条件中的身份池和 `aws:SourceAccount` 条件中的账户时才能代入该角色。

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

```
    ]
}
```

要了解有关 IAM 角色的更多信息，请参阅角色（委托和联合）。

在您的应用程序中使用推送同步：Android

您的应用程序需要导入 Google Play 服务。您可以通过 Android SDK Manager 下载最新版本的 Google Play 开发工具包。按照 Android 有关 Android 实施的文档注册您的应用程序并接收来自 GCM 的注册 ID。收到注册 ID 之后，您需要向 Amazon Cognito 注册设备，如以下代码段所示：

```
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);
}
```

现在，您可以订阅设备以接收来自特定数据集的更新：

```
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);
    }
}
```

要停止接收来自数据集的推送通知，只需调用 unsubscribe 方法即可。要订阅 `CognitoSyncManager` 对象中的所有数据集（或特定子集），请使用 `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);
    }
}
```

实施 Android BroadcastReceiver 对象时，您可以检查已修改数据集的最新版本，并决定您的应用程序是否需要再次同步。

```
@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() {
            // ...
        });
    }

}
```

推送通知负载中提供以下键：

- `source`：cognito-sync。这可以作为通知之间的区分因素。
- `identityPoolId`：身份池 ID。这可用于验证或获取其他信息，但从接收方的角度来看，这并不是不可或缺的。
- `identityId`：池中的身份 ID。
- `datasetName`：已更新的数据集的名称。这可用于 openOrCreateDataset 调用。
- `syncCount`：远程数据集的同步计数。您可以使用此方法来确保本地数据集已过期，并且传入同步是新的。

在您的应用程序中使用推送同步：iOS – Objective-C

要获取应用程序的设备令牌，请参阅 Apple 有关注册远程通知的文档。收到来自 APN 的作为 NSData 对象的设备令牌之后，您需要立即使用同步客户端的 `registerDevice:` 方法向 Amazon Cognito 注册，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;
      }
    ];
```

在调试模式下，设备将向 APN 沙盒注册；在发布模式下，设备将向 APN 注册。要接收来自特定数据集的更新，请使用 `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;
      }
    ];
```

要停止接收来自数据集的推送通知，只需调用 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;
      }
    ];
```

要订阅 AWSCognito 对象中的所有数据集，请调用 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;
      }
    ];
```

在调用 subscribeAll 之前，请确保在每个数据集上至少同步一次，以便数据集存在于服务器上。

要对推送通知做出反应，您需要在应用程序委托上实施 didReceiveRemoteNotification 方法：

```
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary
 *)userInfo
    {
        [[NSNotificationCenter defaultCenter]
 postNotificationName:@"CognitoPushNotification" object:userInfo];
    }
```

如果您使用通知处理程序发布通知，则可以在您拥有数据集句柄的应用程序中的其他位置响应通知。如果您按照如下方式订阅通知...

```
[[NSNotificationCenter defaultCenter] addObserver:self
 selector:@selector(didReceivePushSync:)
    name: :@"CognitoPushNotification" object:nil];
```

...则可以按照如下所示处理通知：

```
- (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;
            }];
        }
    }
```

推送通知负载中提供以下键：

- source：cognito-sync。这可以作为通知之间的区分因素。
- identityPoolId：身份池 ID。这可用于验证或获取其他信息，但从接收方的角度来看，这并不是不可或缺的。
- identityId：池中的身份 ID。
- datasetName：已更新的数据集的名称。这可用于 openOrCreateDataset 调用。
- syncCount：远程数据集的同步计数。您可以使用此方法来确保本地数据集已过期，并且传入同步是新的。

在您的应用程序中使用推送同步：iOS – Swift

要获取应用程序的设备令牌，请参阅 Apple 有关注册远程通知的文档。收到来自 APN 的作为 NSData 对象的设备令牌之后，您需要立即使用同步客户端的 registerDevice: 方法向 Amazon Cognito 注册，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
})
```

在调试模式下，设备将向 APN 沙盒注册；在发布模式下，设备将向 APN 注册。要接收来自特定数据集的更新，请使用 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
})
```

要停止接收来自数据集的推送通知，请调用 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
```

```
})
```

要订阅 AWSCognito 对象中的所有数据集，请调用 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
})
```

在调用 subscribeAll 之前，请确保在每个数据集上至少同步一次，以便数据集存在于服务器上。

要对推送通知做出反应，您需要在应用程序委托上实施 didReceiveRemoteNotification 方法：

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo:
 [NSObject : AnyObject],
  fetchCompletionHandler completionHandler: (UIBackgroundFetchResult) -> Void) {

 NSNotificationCenter.defaultCenter().postNotificationName("CognitoPushNotification",
 object: userInfo)
})
```

如果您使用通知处理程序发布通知，则可以在您拥有数据集句柄的应用程序中的其他位置响应通知。如果您按照如下方式订阅通知...

```
NSNotificationCenter.defaultCenter().addObserver(observer:self,
   selector:"didReceivePushSync:",
   name:"CognitoPushNotification",
   object:nil)
```

...则可以按照如下所示处理通知：

```
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
          }
        }
    }
}
```

推送通知负载中提供以下键：

- source：cognito-sync。这可以作为通知之间的区分因素。
- identityPoolId：身份池 ID。这可用于验证或获取其他信息，但从接收方的角度来看，这并不是不可或缺的。

- `identityId`：池中的身份 ID。
- `datasetName`：已更新的数据集的名称。这可用于 `openOrCreateDataset` 调用。
- `syncCount`：远程数据集的同步计数。您可以使用此方法来确保本地数据集已过期，并且传入同步是新的。

Amazon Cognito Streams

> 如果您是 Amazon Cognito Sync 的新用户，请使用 AWS AppSync。像 Amazon Cognito Sync 一样，AWS AppSync 是一种在设备之间同步应用程序数据的服务。它允许同步用户数据，如应用程序首选项或游戏状态。它还通过允许多个用户实时同步和协作处理共享的数据，来扩展这些功能。

Amazon Cognito Streams 让开发人员能够控制和了解他们存储在 Amazon Cognito 中的数据。现在，开发人员可以配置 Kinesis 流，以便在数据更新和同步时接收事件。Amazon Cognito 可以实时向您拥有的 Kinesis 流推送每个数据集更改。

使用 Amazon Cognito Streams，您可以将所有的同步数据移动到 Kinesis，然后将其流式传输到数据仓库工具（如 Amazon Redshift），供进一步分析。要了解有关 Kinesis 的更多信息，请参阅 Amazon Kinesis 入门。

配置流

您可以在 Amazon Cognito 控制台中设置 Amazon Cognito Streams。要在 Amazon Cognito 控制台中启用 Amazon Cognito Streams，您需要选择要将它发布到哪个 Kinesis 流，以及授权 Amazon Cognito 将事件放入选定流中的 IAM 角色。

从控制台主页：

1. 单击要为其设置 Amazon Cognito 流的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在 Dashboard（控制面板）页的右上角，单击 Manage Identity Pools（管理身份池）。出现“Manage Federated Identities”（管理联合身份）页。
3. 向下滚动，并单击 Cognito Streams（Cognito 流），以将其展开。
4. 在 Stream name（流名称）下拉菜单中，选择一个现有 Kinesis 流的名称。或者，单击 Create stream（创建流）以创建一个流，输入流的名称和分片数量。要了解分片，以及获取有关如何估算流需要的分片数的帮助，请参阅 Kinesis 开发人员指南。
5. 在 Publish role (发布角色) 下拉菜单中，选择授予 Amazon Cognito 发布流的权限的 IAM 角色。在 AWS IAM 控制台中，单击 Create role (创建角色) 以创建或修改与您身份池关联的角色。
6. 在 Stream status（流状态）下拉菜单中，选择 Enabled（启用）以启用流更新。单击 Save Changes（保存更改）。

成功配置 Amazon Cognito Streams 之后，此身份池中数据集的所有后续更新都会发送到此流中。

流内容

发送到流的每个记录都代表一次同步。以下是一个发送到流的记录示例：

```
{
    "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
}
```

对大于 Kinesis 最大有效负载大小 (1 MB) 的更新，Amazon Cognito 将包括预签名 Amazon S3 URL，其中包含更新的完整内容。

配置 Amazon Cognito Streams 之后，如果您删除 Kinesis 流或更改角色信任权限，使得 Amazon Cognito Sync 不再代入该角色，则您将禁用 Amazon Cognito 流。您必须重新创建 Kinesis 流或修复角色，然后必须重新启用此流。

批量发布

配置 Amazon Cognito Streams 之后，您能够对身份池中的现有数据执行批量发布操作。通过控制台或直接通过 API 启动批量发布操作之后，Amazon Cognito 会开始将此数据发布到接收更新的同一流中。

Amazon Cognito 不保证在使用批量发布操作时发送到流的数据具有唯一性。您可能会收到两个相同的更新，一个来自更新操作，一个属于批量发布。在处理来自流的记录时，请记住这一点。

要批量发布所有流，请执行“配置流”下的步骤 1-6，然后单击“Start bulk publish”（开始批量发布）。任何特定时间都只能有一个正在进行的批量发布操作，且每 24 小时只能有一次成功的批量发布请求。

Amazon Cognito Events

> 如果您是 Amazon Cognito Sync 的新用户，请使用 AWS AppSync。像 Amazon Cognito Sync 一样，AWS AppSync 是一种在设备之间同步应用程序数据的服务。
> 它允许同步用户数据，如应用程序首选项或游戏状态。它还通过允许多个用户实时同步和协作处理共享的数据，来扩展这些功能。

Amazon Cognito Events 使您能够执行 AWS Lambda 函数以响应 Amazon Cognito 中的重要事件。当数据集得到同步时，Amazon Cognito 会引发同步触发事件。当用户更新数据时，您可以使用同步触发事件采取行动。该函数可以评估并有选择性地操作数据，然后，数据才会存储到云中并同步到用户的其他设备。这有利于在来自设备的数据同步到用户的其他设备之前对其进行验证，或者基于传入数据更新数据集中的其他值，如在玩家达到新级别时颁发奖励。

以下步骤将引导您设置每次 Amazon Cognito 数据集同步时都会执行的 Lambda 函数。

> Note
>
> 使用 Amazon Cognito Events 时，您只能使用从 Amazon Cognito 身份获取的凭证。如果您有关联的 Lambda 函数，但您通过 AWS 账户凭证（开发人员凭证）调用 `UpdateRecords`，则不会调用 Lambda 函数。

在 AWS Lambda 中创建函数

要将 Lambda 与 Amazon Cognito 集成，您首先需要在 Lambda 中创建函数。为此，请执行以下操作：

在 Amazon Cognito 中选择 Lambda 函数

1. 打开 Lambda 控制台。
2. 单击“Create a Lambda function”（创建 Lambda 函数）。
3. 在“Select blueprint”（选择蓝图）屏幕上，搜索并选择“cognito-sync-trigger”。
4. 在“Configure event sources”（配置事件源）屏幕上，将“Event source type”（事件源类型）设置保留为“Cognito Sync Trigger”，并选择您的身份池。单击“Next”（下一步）。

   Note

   在控制台外配置 Amazon Cognito Sync 触发器时，您必须添加 Lambda 基于资源的权限才能允许 Amazon Cognito 调用该函数。您可以从 Lambda 控制台（请参阅使用 AWS Lambda 的基于资源的策略），或通过使用 Lambda AddPermission 操作添加此权限。
   Lambda 基于资源的策略示例
   以下 AWS Lambda 基于资源的策略授予 Amazon Cognito 有限调用 Lambda 函数的能力。Amazon Cognito 只能在代表 `aws:SourceArn` 条件中的身份池和 `aws:SourceAccount` 条件中的账户时才能调用该函数。

   ```
   {
       "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”（配置函数）屏幕上，输入函数的名称和描述。将“Runtime”（运行时）设置保留为“Node.js”。在我们的示例中保留原来的代码。默认示例没有更改正在同步的数据。它只记录发生了 Amazon Cognito 同步触发事件这一事实。将“Handler name”（处理程序名称）设置保留为“index.handler”。对于 Role (角色)，选择一个授权您的代码访问 AWS Lambda 的 IAM 角色。要修改角色，请参阅 IAM 控制台。将“Advanced”（高级）设置保留不变。单击“Next”（下一步）。
6. 在“Review”（查看）屏幕上，查看详细信息，并单击“Create function”（创建函数）。下一页面将显示您的新 Lambda 函数。

现在，Lambda 中已经写入了相应的函数，您需要选择该函数作为 Amazon Cognito Sync 触发事件的处理程序。以下步骤将指导您完成此过程。

从控制台主页：

1. 单击要为其设置 Amazon Cognito Events 的身份池的名称。此时将显示身份池的“Dashboard”（控制面板）页。
2. 在“Dashboard”（控制面板）页的右上角，单击“Manage Federated Identities”（管理联合身份）。出现“Manage Federated Identities”（管理联合身份）页。

3. 向下滚动，并单击“Cognito Events”（Cognito 事件），以将其展开。
4. 在 Sync Trigger (同步触发) 下拉菜单中，选择您希望在同步事件发生时触发的 Lambda 函数。
5. 单击“Save Changes”（保存更改）。

现在，您的 Lambda 函数将在每次数据集同步时执行。下一部分将介绍如何在数据同步时读取和修改函数中的数据。

为同步触发编写 Lambda 函数

同步触发器遵循服务提供商接口使用的编程模式。Amazon Cognito 将按照以下 JSON 格式向您的 Lambda 函数提供输入。

```
{
  "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 预计函数的返回值与输入格式相同。

编写同步触发器事件的函数时，应注意以下事项：

- Amazon Cognito 在 UpdateRecords 期间调用 Lambda 函数时，该函数必须在 5 秒内响应。如果没有，则 Amazon Cognito Sync 服务将生成 `LambdaSocketTimeoutException` 异常。您无法增加此超时值。
- 如果您遇到了 `LambdaThrottledException` 异常，请再次尝试同步操作以更新记录。
- Amazon Cognito 提供数据集中出现的所有记录，以作为函数的输入。
- 应用程序用户更新中将 op 字段设置为 `replace` 的记录。删除的记录将 op 字段设置为 `remove`。
- 您可以修改任何记录，即使应用程序用户没有更新该记录也是如此。
- 除 datasetRecords 之外的所有字段均为只读。请勿更改它们。如果您更改这些字段，则无法更新记录。
- 要修改记录的值，只需更新该值并将 op 设置为 `replace`。
- 要删除记录，请将 op 设置为 `remove`，或将该值设置为空。
- 要添加记录，请将新记录添加到 datasetRecords 数组。
- Amazon Cognito 更新记录时，Amazon Cognito 会忽略响应中任何省略的记录。

示例 Lambda 函数

以下示例 Lambda 函数显示如何访问、修改和删除数据。

```
console.log('Loading function');

exports.handler = function(event, context) {
```

```
    console.log(JSON.stringify(event, null, 2));

    //Check for the event type
    if (event.eventType === 'SyncTrigger') {

        //Modify value for a key
        if('SampleKey1' in event.datasetRecords){
            event.datasetRecords.SampleKey1.newValue = 'ModifyValue1';
            event.datasetRecords.SampleKey1.op = 'replace';
        }

        //Remove a key
        if('SampleKey2' in event.datasetRecords){
            event.datasetRecords.SampleKey2.op = 'remove';
        }

        //Add a key
        if(!('SampleKey3' in event.datasetRecords)){
            event.datasetRecords.SampleKey3={'newValue':'ModifyValue3', 'op' : 'replace'};
        }

    }
    context.done(null, event);
};
```

使用 AWS SDK 的 Amazon Cognito 的代码示例

以下代码示例展示如何将 Amazon Cognito 与 AWS 软件开发工具包（SDK）一起使用。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

代码示例

- 使用 AWS SDK 的 Amazon Cognito 身份的代码示例 (p. 401)
  - 使用 AWS SDK 的 Amazon Cognito 身份的操作 (p. 402)
    - 使用指定名称创建一个新的 Amazon Cognito 身份 (p. 402)
    - 删除具有指定名称的 Amazon Cognito 身份 (p. 403)
    - 列出 Amazon Cognito 身份池 (p. 404)
  - 使用 AWS SDK 的 Amazon Cognito 身份的跨服务示例 (p. 405)
    - 构建 Amazon Transcribe 应用程序 (p. 406)
    - 创建 Amazon Textract 浏览器应用程序 (p. 406)
- 使用 AWS SDK 的 Amazon Cognito 身份提供者的代码示例 (p. 407)
  - 使用 AWS SDK 的 Amazon Cognito 身份提供者的操作 (p. 407)
    - 使用 AWS SDK 确认 Amazon Cognito 用户 (p. 408)
    - 使用 AWS SDK 确认 MFA 设备可通过 Amazon Cognito 进行跟踪 (p. 411)
    - 使用 AWS SDK 删除 Amazon Cognito 用户 (p. 414)
    - 使用 AWS SDK 获取令牌以将 MFA 应用程序与 Amazon Cognito 用户关联 (p. 415)
    - 使用 AWS SDK 获取有关 Amazon Cognito 用户的信息 (p. 418)
    - 使用 AWS SDK 列出 Amazon Cognito 用户群体 (p. 422)
    - 使用 AWS SDK 列出 Amazon Cognito 用户 (p. 424)
    - 使用 AWS SDK 重新发送 Amazon Cognito 确认码 (p. 427)
    - 使用 AWS SDK 响应 Amazon Cognito SRP 身份验证质询 (p. 431)
    - 使用 AWS SDK 响应 Amazon Cognito 身份验证质询 (p. 434)
    - 使用 AWS SDK 向 Amazon Cognito 注册用户 (p. 438)
    - 使用 AWS SDK 通过 Amazon Cognito 跟踪的设备开始身份验证 (p. 443)
    - 使用 AWS SDK 通过 Amazon Cognito 和管理员凭证开始身份验证 (p. 446)
    - 使用 AWS SDK 向 Amazon Cognito 用户验证 MFA 应用程序 (p. 450)
  - 使用 AWS SDK 的 Amazon Cognito 身份提供者的场景 (p. 454)
    - 使用 AWS SDK 向需要 MFA 的 Amazon Cognito 用户群体注册用户 (p. 454)
- 使用 AWS SDK 的 Amazon Cognito 同步的代码示例 (p. 488)
  - 使用 AWS SDK 的 Amazon Cognito 同步的操作 (p. 488)
    - 使用 AWS SDK 获取向 Amazon Cognito 注册的身份池列表 (p. 488)

使用 AWS SDK 的 Amazon Cognito 身份的代码示例

以下代码示例展示如何将 Amazon Cognito 身份与 AWS 软件开发工具包 (SDK) 一起使用。

操作是大型程序的代码摘录，必须在上下文中运行。虽然操作向您展示了如何调用单个服务函数，但您可以在其相关场景和跨服务示例中查看操作的上下文。

跨服务示例是指跨多个 AWS 服务 工作的示例应用程序。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

代码示例

- 使用 AWS SDK 的 Amazon Cognito 身份的操作 (p. 402)
  - 使用指定名称创建一个新的 Amazon Cognito 身份 (p. 402)
  - 删除具有指定名称的 Amazon Cognito 身份 (p. 403)
  - 列出 Amazon Cognito 身份池 (p. 404)
- 使用 AWS SDK 的 Amazon Cognito 身份的跨服务示例 (p. 405)
  - 构建 Amazon Transcribe 应用程序 (p. 406)
  - 创建 Amazon Textract 浏览器应用程序 (p. 406)

使用 AWS SDK 的 Amazon Cognito 身份的操作

以下代码示例演示如何使用 AWS SDK 执行各个 Amazon Cognito 身份操作。这些代码节选调用了 Amazon Cognito 身份 API，是必须在上下文中运行的较大型程序的代码节选。每个示例都包含一个指向 GitHub 的链接，您可以在其中找到有关设置和运行代码的说明。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon Cognito 身份 API 参考。

示例

- 使用指定名称创建一个新的 Amazon Cognito 身份 (p. 402)
- 删除具有指定名称的 Amazon Cognito 身份 (p. 403)
- 列出 Amazon Cognito 身份池 (p. 404)

使用指定名称创建一个新的 Amazon Cognito 身份

以下代码示例显示如何找到具有给定名称的 Amazon Cognito 身份，如果找不到则创建该身份。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 列出身份池 (p. 404)
- 列出身份池 (p. 404)

Swift

SDK for Swift

Note

这是预览版 SDK 的预发布文档。本文档随时可能更改。

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

创建新的身份池。

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

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

        return poolId
    }
```

- 有关更多信息，请参阅 AWS SDK for Swift 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的以下主题。
  - CreateIdentityPool
  - ListIdentityPools

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

删除具有指定名称的 Amazon Cognito 身份

以下代码示例显示如何删除给定了 ID 的 Amazon Cognito 身份。

Swift

SDK for Swift

> **Note**
>
> 这是预览版 SDK 的预发布文档。本文档随时可能更改。

> **Note**
>
> 在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

删除指定的身份池。

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

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

- 有关更多信息，请参阅 AWS SDK for Swift 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的 DeleteIdentityPool。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

列出 Amazon Cognito 身份池

以下代码示例显示如何获取 Amazon Cognito 身份池的列表。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 创建 身份池 (p. 402)
- 创建 身份池 (p. 402)

Swift

SDK for Swift

Note

这是预览版 SDK 的预发布文档。本文档随时可能更改。

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

根据给定的身份池名称查找其 ID。

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

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

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

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

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

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

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

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

            token = output.nextToken
        } while token != nil

        return nil
    }
```

获取现有身份池的 ID，如果它不存在，则创建它。

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

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

        return poolId
    }
```

- 有关更多信息，请参阅 AWS SDK for Swift 开发人员指南。
- 有关 API 详细信息，请参阅《AWS SDK for Swift API 参考》中的以下主题。
  - CreateIdentityPool
  - ListIdentityPools

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 的 Amazon Cognito 身份的跨服务示例

以下示例应用程序使用 AWS SDK 将 Amazon Cognito 身份与其他 AWS 服务 相结合。每个示例都包含一个指向 GitHub 的链接，其中包含了有关如何设置和运行应用程序的说明。

示例

- 构建 Amazon Transcribe 应用程序 (p. 406)

- 创建 Amazon Textract 浏览器应用程序 (p. 406)

构建 Amazon Transcribe 应用程序

以下代码示例显示了如何使用 Amazon Transcribe 在浏览器中转录并显示录音。

JavaScript

SDK for JavaScript (v3)

创建一个使用 Amazon Transcribe 在浏览器中转录和显示录音的应用程序。该应用程序使用两个 Amazon Simple Storage Service（Amazon S3）桶，一个用于托管应用程序代码，另一个用于存储转录。该应用程序使用 Amazon Cognito 用户池对您的用户进行身份验证。已经过身份验证的用户有 AWS Identity and Access Management (IAM) 权限来访问所需的 AWS 服务。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

此示例也可在 AWS SDK for JavaScript v3 开发人员指南中找到。

本示例中使用的服务

- Amazon Cognito Identity
- Amazon S3
- Amazon Transcribe

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

创建 Amazon Textract 浏览器应用程序

以下代码示例展示如何通过交互式应用程序探索 Amazon Textract 输出。

JavaScript

SDK for JavaScript (v3)

展示如何使用 AWS SDK for JavaScript 构建 React 应用程序，该应用程序使用 Amazon Textract 从文档图像中提取数据并在交互式网页中显示该数据。此示例在 Web 浏览器中运行，需要经过身份验证的 Amazon Cognito 身份才能获得凭证。它使用 Amazon Simple Storage Service (Amazon S3) 进行存储；对于通知，它将轮询订阅 Amazon Simple Notification Service (Amazon SNS) 主题的 Amazon Simple Queue Service (Amazon SQS) 队列。

有关完整的源代码以及如何设置和运行的说明，请参阅 GitHub 上的完整示例。

本示例中使用的服务

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

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 的 Amazon Cognito 身份提供者的代码示例

以下代码示例显示如何将 Amazon Cognito 身份提供者与 AWS 软件开发工具包 (SDK) 一起使用。

操作是大型程序的代码摘录，必须在上下文中运行。虽然操作向您展示了如何调用单个服务函数，但您可以在其相关场景和跨服务示例中查看操作的上下文。

场景是展示如何通过在同一服务中调用多个函数来完成特定任务的代码示例。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

代码示例

- 使用 AWS SDK 的 Amazon Cognito 身份提供者的操作 (p. 407)
  - 使用 AWS SDK 确认 Amazon Cognito 用户 (p. 408)
  - 使用 AWS SDK 确认 MFA 设备可通过 Amazon Cognito 进行跟踪 (p. 411)
  - 使用 AWS SDK 删除 Amazon Cognito 用户 (p. 414)
  - 使用 AWS SDK 获取令牌以将 MFA 应用程序与 Amazon Cognito 用户关联 (p. 415)
  - 使用 AWS SDK 获取有关 Amazon Cognito 用户的信息 (p. 418)
  - 使用 AWS SDK 列出 Amazon Cognito 用户群体 (p. 422)
  - 使用 AWS SDK 列出 Amazon Cognito 用户 (p. 424)
  - 使用 AWS SDK 重新发送 Amazon Cognito 确认码 (p. 427)
  - 使用 AWS SDK 响应 Amazon Cognito SRP 身份验证质询 (p. 431)
  - 使用 AWS SDK 响应 Amazon Cognito 身份验证质询 (p. 434)
  - 使用 AWS SDK 向 Amazon Cognito 注册用户 (p. 438)
  - 使用 AWS SDK 通过 Amazon Cognito 跟踪的设备开始身份验证 (p. 443)
  - 使用 AWS SDK 通过 Amazon Cognito 和管理员凭证开始身份验证 (p. 446)
  - 使用 AWS SDK 向 Amazon Cognito 用户验证 MFA 应用程序 (p. 450)
- 使用 AWS SDK 的 Amazon Cognito 身份提供者的场景 (p. 454)
  - 使用 AWS SDK 向需要 MFA 的 Amazon Cognito 用户群体注册用户 (p. 454)

使用 AWS SDK 的 Amazon Cognito 身份提供者的操作

以下代码示例演示如何使用 AWS SDK 执行各个 Amazon Cognito 身份提供者操作。这些代码节选调用了 Amazon Cognito 身份提供者 API，是必须在上下文中运行的较大型程序的代码节选。每个示例都包含一个指向 GitHub 的链接，您可以在其中找到有关设置和运行代码的说明。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon Cognito 身份提供者 API 参考。

示例

- 使用 AWS SDK 确认 Amazon Cognito 用户 (p. 408)
- 使用 AWS SDK 确认 MFA 设备可通过 Amazon Cognito 进行跟踪 (p. 411)
- 使用 AWS SDK 删除 Amazon Cognito 用户 (p. 414)
- 使用 AWS SDK 获取令牌以将 MFA 应用程序与 Amazon Cognito 用户关联 (p. 415)
- 使用 AWS SDK 获取有关 Amazon Cognito 用户的信息 (p. 418)
- 使用 AWS SDK 列出 Amazon Cognito 用户群体 (p. 422)

- 使用 AWS SDK 列出 Amazon Cognito 用户 (p. 424)
- 使用 AWS SDK 重新发送 Amazon Cognito 确认码 (p. 427)
- 使用 AWS SDK 响应 Amazon Cognito SRP 身份验证质询 (p. 431)
- 使用 AWS SDK 响应 Amazon Cognito 身份验证质询 (p. 434)
- 使用 AWS SDK 向 Amazon Cognito 注册用户 (p. 438)
- 使用 AWS SDK 通过 Amazon Cognito 跟踪的设备开始身份验证 (p. 443)
- 使用 AWS SDK 通过 Amazon Cognito 和管理员凭证开始身份验证 (p. 446)
- 使用 AWS SDK 向 Amazon Cognito 用户验证 MFA 应用程序 (p. 450)

使用 AWS SDK 确认 Amazon Cognito 用户

以下代码示例显示如何确认 Amazon Cognito 用户。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户群体注册用户 (p. 454)

.NET

AWS SDK for .NET

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
    /// <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></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;
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 ConfirmSignUp。

C++

适用于 C++ 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
        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;
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 ConfirmSignUp。

Java

SDK for Java 2.x

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
    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);
```

```
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 ConfirmSignUp。

JavaScript

SDK for JavaScript (v3)

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

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

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

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

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 ConfirmSignUp。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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")
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 ConfirmSignUp。

Python

适用于 Python (Boto3) 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 ConfirmSignUp。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 确认 MFA 设备可通过 Amazon Cognito 进行跟踪

以下代码示例显示如何确认 MFA 设备可通过 Amazon Cognito 进行跟踪。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户群体注册用户 (p. 454)

.NET

AWS SDK for .NET

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
    /// <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;
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 ConfirmDevice。

JavaScript

SDK for JavaScript (v3)

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

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

  const command = new ConfirmDeviceCommand({
    DeviceKey: deviceKey,
    AccessToken: accessToken,
    DeviceSecretVerifierConfig: {
      PasswordVerifier: passwordVerifier,
      Salt: salt
    },
  });

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

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 ConfirmDevice。

Python

适用于 Python (Boto3) 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 ConfirmDevice。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 删除 Amazon Cognito 用户

以下代码示例显示如何删除 Amazon Cognito 用户。

C++

SDK for C++

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
        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;
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 DeleteUser。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 获取令牌以将 MFA 应用程序与 Amazon Cognito 用户关联

以下代码示例显示如何获取令牌，以将 MFA 应用程序与 Amazon Cognito 用户关联。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户群体注册用户 (p. 454)

.NET

AWS SDK for .NET

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>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.Write("Enter the following token into the authenticator:
{secretCode}");

        return tokenResponse.Session;
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 AssociateSoftwareToken。

C++

适用于 C++ 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
        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;
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 AssociateSoftwareToken。

Java

SDK for Java 2.x

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
    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();
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 AssociateSoftwareToken。

JavaScript

SDK for JavaScript (v3)

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
const associateSoftwareToken = async (session) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);
  const command = new AssociateSoftwareTokenCommand({
    Session: session,
  });

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

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 AssociateSoftwareToken。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 AssociateSoftwareToken。

Python

适用于 Python (Boto3) 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 AssociateSoftwareToken。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 获取有关 Amazon Cognito 用户的信息

以下代码示例显示如何获取有关 Amazon Cognito 用户的信息。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户群体注册用户 (p. 454)

.NET

AWS SDK for .NET

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
   /// <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></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;
   }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 AdminGetUser。

C++

适用于 C++ 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
       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;
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 AdminGetUser。

Java

SDK for Java 2.x

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
    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);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 AdminGetUser。

JavaScript

SDK for JavaScript (v3)

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

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

  const command = new AdminGetUserCommand({
    UserPoolId: userPoolId,
    Username: username,
```

```
  });

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

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 AdminGetUser。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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}")
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 AdminGetUser。

Python

适用于 Python (Boto3) 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 AdminGetUser。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 列出 Amazon Cognito 用户群体

以下代码示例显示如何列出 Amazon Cognito 用户群体。

.NET

AWS SDK for .NET

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
    /// <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;
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 ListUserPools。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client.list_user_pools().max_results(10).send().await?;
    if let Some(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!("  Status:          {:?}", pool.status());
            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(())
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 ListUserPools。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 列出 Amazon Cognito 用户

以下代码示例显示如何列出 Amazon Cognito 用户。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户群体注册用户 (p. 454)

.NET

AWS SDK for .NET

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
/// <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;
}
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 ListUsers。

Java

SDK for Java 2.x

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
    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);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 ListUsers。

JavaScript

SDK for JavaScript (v3)

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
const listUsers = async ({ userPoolId }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new ListUsersCommand({
    UserPoolId: userPoolId,
  });

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

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 ListUsers。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 ListUsers。

Python

适用于 Python (Boto3) 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python（Boto3）API 参考》中的 ListUsers。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 重新发送 Amazon Cognito 确认码

以下代码示例显示如何重新发送 Amazon Cognito 确认码。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户群体注册用户 (p. 454)

.NET

AWS SDK for .NET

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
    /// <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></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;
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 ResendConfirmationCode。

C++

适用于 C++ 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
        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;
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 ResendConfirmationCode。

Java

SDK for Java 2.x

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
    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);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 ResendConfirmationCode。

JavaScript

SDK for JavaScript (v3)

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
const resendConfirmationCode = async ({ clientId, username }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new ResendConfirmationCodeCommand({
    ClientId: clientId,
    Username: username
  });

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

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 ResendConfirmationCode。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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))
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 ResendConfirmationCode。

Python

适用于 Python (Boto3) 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 ResendConfirmationCode。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 响应 Amazon Cognito SRP 身份验证质询

以下代码示例显示如何响应 Amazon Cognito SRP 身份验证质询。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户群体注册用户 (p. 454)

.NET

AWS SDK for .NET

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
   /// <summary>
   /// Respond to an 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>
   /// <returns>An async Task.</returns>
   public async Task<AuthenticationResultType> RespondToAuthChallengeAsync(string
userName, string clientId, string mfaCode, string session)
   {
       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 RespondToAuthChallengeRequest
       {
           ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
           ClientId = clientId,
           ChallengeResponses = challengeResponses,
           Session = session
       };

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

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 RespondToAuthChallenge。

JavaScript

SDK for JavaScript (v3)

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
const respondToAuthChallenge = async ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = createClientForDefaultRegion(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);
};
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 RespondToAuthChallenge。

Python

适用于 Python (Boto3) 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

登录跟踪的设备。要完成登录，客户端必须正确响应安全远程密码 (SRP) 质询。

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

```
   def sign_in_with_tracked_device(
           self, user_name, password, device_key, device_group_key,
device_password,
           aws_srp):
       """
       Signs in to Amazon Cognito as a user who has a tracked device. Signing in
       with a tracked device lets a user sign in without entering a new MFA code.

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

       For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

       :param user_name: The user that is associated with the device.
       :param password: The user's password.
       :param device_key: The key of a tracked device.
       :param device_group_key: The group key of a tracked device.
       :param device_password: The password that is associated with the device.
       :param aws_srp: A class that helps with SRP calculations. The scenario
                       associated with this example uses the warrant package.
       :return: The result of the authentication. When successful, this contains
an
                access token for the user.
       """
       try:
           srp_helper = aws_srp.AWSSRP(
               username=user_name, password=device_password,
               pool_id='_', client_id=self.client_id, client_secret=None,
               client=self.cognito_idp_client)

           response_init = self.cognito_idp_client.initiate_auth(
               ClientId=self.client_id, AuthFlow='USER_PASSWORD_AUTH',
               AuthParameters={
                   'USERNAME': user_name, 'PASSWORD': password, 'DEVICE_KEY':
device_key})
           if response_init['ChallengeName'] != 'DEVICE_SRP_AUTH':
               raise RuntimeError(
                   f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}.")

           auth_params = srp_helper.get_auth_params()
           auth_params['DEVICE_KEY'] = device_key
           response_auth = self.cognito_idp_client.respond_to_auth_challenge(
               ClientId=self.client_id,
               ChallengeName='DEVICE_SRP_AUTH',
               ChallengeResponses=auth_params
           )
           if response_auth['ChallengeName'] != 'DEVICE_PASSWORD_VERIFIER':
               raise RuntimeError(
                   f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                   f"{response_init['ChallengeName']}.")

           challenge_params = response_auth['ChallengeParameters']
           challenge_params['USER_ID_FOR_SRP'] = device_group_key + device_key
           cr = srp_helper.process_challenge(challenge_params, {'USERNAME':
user_name})
           cr['USERNAME'] = user_name
           cr['DEVICE_KEY'] = device_key
           response_verifier = self.cognito_idp_client.respond_to_auth_challenge(
               ClientId=self.client_id,
               ChallengeName='DEVICE_PASSWORD_VERIFIER',
```

```
                ChallengeResponses=cr)
            auth_tokens = response_verifier['AuthenticationResult']
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
user_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return auth_tokens
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 RespondToAuthChallenge。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 响应 Amazon Cognito 身份验证质询

以下代码示例显示如何响应 Amazon Cognito 身份验证质询。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户群体注册用户 (p. 454)

.NET

AWS SDK for .NET

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
   public async Task<AuthenticationResultType>
AdminRespondToAuthChallengeAsync(string userPoolId, string userName, string
clientId, string mfaCode, string session)
   {
       var challengeResponses = new Dictionary<string, string>();
       challengeResponses.Add("USERNAME", userName);
       challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

       var request = new AdminRespondToAuthChallengeRequest
       {
           ClientId = clientId,
           UserPoolId = userPoolId,
           ChallengeResponses = challengeResponses,
           Session = session
       };

       var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(request);
       return response.AuthenticationResult;
   }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 AdminRespondToAuthChallenge。

C++

适用于 C++ 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
        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;
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 AdminRespondToAuthChallenge。

Java

SDK for Java 2.x

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 AdminRespondToAuthChallenge。

JavaScript

SDK for JavaScript (v3)

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
const adminRespondToAuthChallenge = async ({
  userPoolId,
  clientId,
  username,
  totp,
  session,
}) => {
  const client = createClientForDefaultRegion(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);
};
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 AdminRespondToAuthChallenge。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

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

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

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

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 AdminRespondToAuthChallenge。

Python

适用于 Python (Boto3) 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

通过提供关联的 MFA 应用程序生成的代码来响应 MFA 质询。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 AdminRespondToAuthChallenge。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 向 Amazon Cognito 注册用户

以下代码示例显示如何向 Amazon Cognito 注册用户。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户群体注册用户 (p. 454)

.NET

AWS SDK for .NET

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
    /// <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;
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的注册。

C++

适用于 C++ 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
        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;
       }
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的注册。

Java

SDK for Java 2.x

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
   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);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的注册。

JavaScript

SDK for JavaScript (v3)

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
const signUp = async ({ clientId, username, password, email }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

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

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的注册。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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")
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的注册。

Python

适用于 Python (Boto3) 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的注册。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 通过 Amazon Cognito 跟踪的设备开始身份验证

以下代码示例显示如何通过 Amazon Cognito 跟踪的设备开始身份验证。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户群体注册用户 (p. 454)

.NET

AWS SDK for .NET

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
    /// <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 call to InitiateAuthAsync.</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;
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 InitiateAuth。

JavaScript

SDK for JavaScript (v3)

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

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

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

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

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 InitiateAuth。

Python

适用于 Python (Boto3) 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

登录跟踪的设备。要完成登录，客户端必须正确响应安全远程密码 (SRP) 质询。

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

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

    def sign_in_with_tracked_device(
            self, user_name, password, device_key, device_group_key,
device_password,
            aws_srp):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA code.

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

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name, password=device_password,
                pool_id='_', client_id=self.client_id, client_secret=None,
                client=self.cognito_idp_client)

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id, AuthFlow='USER_PASSWORD_AUTH',
                AuthParameters={
                    'USERNAME': user_name, 'PASSWORD': password, 'DEVICE_KEY':
device_key})
            if response_init['ChallengeName'] != 'DEVICE_SRP_AUTH':
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}.")

            auth_params = srp_helper.get_auth_params()
            auth_params['DEVICE_KEY'] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName='DEVICE_SRP_AUTH',
                ChallengeResponses=auth_params
            )
            if response_auth['ChallengeName'] != 'DEVICE_PASSWORD_VERIFIER':
```

```
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}.")

            challenge_params = response_auth['ChallengeParameters']
            challenge_params['USER_ID_FOR_SRP'] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {'USERNAME':
user_name})
            cr['USERNAME'] = user_name
            cr['DEVICE_KEY'] = device_key
            response_verifier = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName='DEVICE_PASSWORD_VERIFIER',
                ChallengeResponses=cr)
            auth_tokens = response_verifier['AuthenticationResult']
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
user_name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return auth_tokens
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 InitiateAuth。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 通过 Amazon Cognito 和管理员凭证开始身份验证

以下代码示例显示如何使用 Amazon Cognito 和管理员凭证开始身份验证。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户群体注册用户 (p. 454)

.NET

AWS SDK for .NET

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
   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.USER_PASSWORD_AUTH,
```

```
        };

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

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 AdminInitiateAuth。

C++

适用于 C++ 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
        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;
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 AdminInitiateAuth。

Java

SDK for Java 2.x

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
   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;
   }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 AdminInitiateAuth。

JavaScript

SDK for JavaScript (v3)

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

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

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

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

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 AdminInitiateAuth。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 AdminInitiateAuth。

Python

适用于 Python (Boto3) 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 AdminInitiateAuth。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 向 Amazon Cognito 用户验证 MFA 应用程序

以下代码示例显示如何向 Amazon Cognito 用户验证 MFA 应用程序。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户群体注册用户 (p. 454)

.NET

AWS SDK for .NET

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
   /// <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;
   }
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的 VerifySoftwareToken。

C++

适用于 C++ 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
       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;
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的 VerifySoftwareToken。

Java

SDK for Java 2.x

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 VerifySoftwareToken。

JavaScript

SDK for JavaScript (v3)

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
const verifySoftwareToken = async (totp) => {
  const client = createClientForDefaultRegion(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);
};
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的 VerifySoftwareToken。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

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

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的 VerifySoftwareToken。

Python

适用于 Python (Boto3) 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 VerifySoftwareToken。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 的 Amazon Cognito 身份提供者的场景

以下代码示例向您展示如何通过 AWS SDK 实施 Amazon Cognito 身份提供者中的常见场景。这些场景向您展示了如何通过在 Amazon Cognito 身份提供者中调用多个函数来完成特定任务。每个场景都包含一个指向 GitHub 的链接，其中包含了有关如何设置和运行代码的说明。

示例

- 使用 AWS SDK 向需要 MFA 的 Amazon Cognito 用户群体注册用户 (p. 454)

使用 AWS SDK 向需要 MFA 的 Amazon Cognito 用户群体注册用户

以下代码示例显示了如何：

- 使用用户名、密码和电子邮件地址注册和确认用户。
- 通过将 MFA 应用程序与用户关联来设置多重身份验证。
- 使用密码和 MFA 代码登录。

.NET

AWS SDK for .NET

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
global using Amazon.CognitoIdentityProvider;
global using Amazon.CognitoIdentityProvider.Model;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


namespace 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>()
            .AddTransient<UiMethods>()
            )
            .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>();
        var uiMethods = host.Services.GetRequiredService<UiMethods>();

        // clientId - The app client Id value that you get from the AWS CDK script.
        string clientId = configuration["ClientId"]; // "*** REPLACE WITH CLIENT ID
 VALUE FROM CDK SCRIPT";

        // poolId - The pool Id that you get from the AWS CDK script.
        string poolId = configuration["PoolId"]; // "*** REPLACE WITH POOL ID VALUE
 FROM CDK SCRIPT";
        var userName = configuration["UserName"];
        var password = configuration["Password"];
        var email = configuration["Email"];
        var userPoolId = configuration["UserPoolId"];
```

```
        // If the username wasn't set in the configuration file,
        // get it from the user now.
        if (userName is null)
        {
            do
            {
                Console.Write("Username: ");
                userName = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(userName));
        }
        Console.WriteLine($"\nUsername: {userName}");

        // If the password wasn't set in the configuration file,
        // get it from the user now.
        if (password is null)
        {
            do
            {
                Console.Write("Password: ");
                password = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(password));
        }

        // If the email address wasn't set in the configuration file,
        // get it from the user now.
        if (email is null)
        {
            do
            {
                Console.Write("Email: ");
                email = Console.ReadLine();
            } while (string.IsNullOrEmpty(email));
        }

        // Now sign up the user.
        Console.WriteLine($"\nSigning up {userName} with email address: {email}");
        await cognitoWrapper.SignUpAsync(clientId, userName, password, email);

        // Add the user to the user pool.
        Console.WriteLine($"Adding {userName} to the user pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        uiMethods.DisplayTitle("Get confirmation code");
        Console.WriteLine($"Conformation code sent to {userName}.");
        Console.Write("Would you like to send a new code? (Yes/No) ");
        var answer = Console.ReadLine();

        if (answer.ToLower() == "YES")
        {
            await cognitoWrapper.ResendConfirmationCodeAsync(clientId, userName);
            Console.WriteLine("Sending a new confirmation code");
        }

        Console.Write("Enter confirmation code (from Email): ");
        string 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);

        var authResponse = await cognitoWrapper.InitiateAuthAsync(clientId,
userName, password);
```

```
        var mySession = authResponse.Session;

        var newSession = await
 cognitoWrapper.AssociateSoftwareTokenAsync(mySession);

        Console.Write("Enter the 6-digit code displayed in Google Authenticator:
 ");
        string myCode = Console.ReadLine();

        // Verify the TOTP and register for MFA.
        await cognitoWrapper.GetAdminUserAsync(newSession, myCode);
        Console.Write("Re-enter the 6-digit code displayed in your authenticator");
        string mfaCode = Console.ReadLine();

        var session2 = await cognitoWrapper.AdminInitiateAuthAsync(clientId,
 userPoolId, userName, password);
        await cognitoWrapper.RespondToAuthChallengeAsync(userName, clientId,
 mfaCode, session2);
    }
}


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;
   }


   public async Task<AuthenticationResultType>
AdminRespondToAuthChallengeAsync(string userPoolId, string userName, string
clientId, string mfaCode, string session)
   {
       var challengeResponses = new Dictionary<string, string>();
       challengeResponses.Add("USERNAME", userName);
       challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

       var request = new AdminRespondToAuthChallengeRequest
       {
           ClientId = clientId,
           UserPoolId = userPoolId,
           ChallengeResponses = challengeResponses,
           Session = session
       };

       var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(request);
       return response.AuthenticationResult;
   }


   /// <summary>
   /// Respond to an 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>
   /// <returns>An async Task.</returns>
   public async Task<AuthenticationResultType> RespondToAuthChallengeAsync(string
userName, string clientId, string mfaCode, string session)
   {
       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 RespondToAuthChallengeRequest
       {
           ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
           ClientId = clientId,
           ChallengeResponses = challengeResponses,
           Session = session
       };
```

```
        var response = await
_cognitoService.RespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult}");
        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>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.Write("Enter the following token into the authenticator:
{secretCode}");

        return tokenResponse.Session;
    }


    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.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 call to InitiateAuthAsync.</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></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></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></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;
    }

}


namespace CognitoBasics;

/// <summary>
/// Some useful methods to make screen display easier.
/// </summary>
public class UiMethods
{
    public readonly string SepBar = new string('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the scenario.
    /// </summary>
    public void DisplayOverview()
    {
        Console.Clear();
        DisplayTitle("Welcome to the Amazon Cognito Demo");

        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t 1. Signs up a user.");
        Console.WriteLine("\t 2. Gets the user's confirmation status.");
        Console.WriteLine("\t 3. Resends the confirmation code if the user
 requested another code.");
        Console.WriteLine("\t 4. Confirms that the user signed up.");
        Console.WriteLine("\t 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”).");
```

```
        Console.WriteLine("\t 6. Invokes the AssociateSoftwareToken method to
 generate a TOTP MFA private key. This can be used with Google Authenticator.");
        Console.WriteLine("\t 7. Invokes the VerifySoftwareToken method to verify
 the TOTP and register for MFA.");
        Console.WriteLine("\t 8. Invokes the AdminInitiateAuth to sign in again.
 This results in being prompted to submit a TOTP (Response: “ChallengeName”:
 “SOFTWARE_TOKEN_MFA”).");
        Console.WriteLine("\t 9. Invokes the AdminRespondToAuthChallenge to get
 back a token.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue.");
        _ = Console.ReadLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to pad with spaces.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的以下主题。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

C++

适用于 C++ 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
        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();
}
```

- 有关 API 详细信息，请参阅《AWS SDK for C++ API 参考》中的以下主题。
  - AdminGetUser

- AdminInitiateAuth
- AdminRespondToAuthChallenge
- AssociateSoftwareToken
- ConfirmDevice
- ConfirmSignUp
- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

Java

SDK for Java 2.x

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
/**
 * 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 = "\n" +
            "Usage:\n" +
```

```
            "    <clientId> <poolId>\n\n" +
            "Where:\n" +
            "    clientId - The app client Id value that you can get from the AWS
 CDK script.\n\n" +
            "    poolId - The pool Id that you can get from the AWS CDK script. \n
\n" ;

        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)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .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);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的以下主题。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

JavaScript

SDK for JavaScript (v3)

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

为了获得最佳体验，请克隆 GitHub 存储库并运行此示例。以下代码代表完整示例应用程序的示例。

```
import { log } from "../../../../libs/utils/util-log.js";
import { signUp } from "../../../actions/sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "../../../../libs/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);
    const clientId = getSecondValuesFromEntries(FILE_USER_POOLS)[0];
    validateClient(clientId);
    log(`Signing up.`);
    await signUp({ clientId, username, password, email });
    log(`Signed up. An confirmation email has been sent to: ${email}.`);
    log(`Run 'confirm-sign-up ${username} <code>' to confirm your account.`);
  } catch (err) {
    log(err);
  }
};

export { signUpHandler };

const signUp = async ({ clientId, username, password, email }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};

import { log } from "../../../../libs/utils/util-log.js";
import { confirmSignUp } from "../../../actions/confirm-sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "../../../../libs/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);
    const clientId = getSecondValuesFromEntries(FILE_USER_POOLS)[0];
    validateClient(clientId);
    log(`Confirming user.`);
    await confirmSignUp({ clientId, username, code });
    log(
      `User confirmed. Run 'admin-initiate-auth ${username} <password>' to sign
 in.`
    );
  } catch (err) {
    log(err);
  }
};

export { confirmSignUpHandler };

const confirmSignUp = async ({ clientId, username, code }) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

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

  return client.send(command);
};

import qrcode from "qrcode-terminal";
import { log } from "../../../../libs/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 "../../../../libs/utils/util-csv.js";

const handleMfaSetup = async (session, username) => {
  const { SecretCode, Session } = await associateSoftwareToken(session);

  // Store the Session for use with 'VerifySoftwareToken'.
  process.env.SESSION = Session;

  console.log(
    "Scan this code in your preferred authenticator app, then run 'verify-software-
token' to finish the setup."
  );
  qrcode.generate(
    `otpauth://totp/${username}?secret=${SecretCode}`,
    { small: true },
    console.log
  );
};

const handleSoftwareTokenMfa = (session) => {
  // Store the Session for use with 'AdminRespondToAuthChallenge'.
```

```
  process.env.SESSION = session;
};

const validateClient = (id) => {
  if (!id) {
    throw new Error(
      `User pool client id is missing. Did you run 'create-user-pool'?`
    );
  }
};

const validateId = (id) => {
  if (!id) {
    throw new Error(`User pool id is missing. Did you run 'create-user-pool'?`);
  }
};

const validateUser = (username, password) => {
  if (!(username && password)) {
    throw new Error(
      `Username and password must be provided as arguments to the 'admin-initiate-
auth' command.`
    );
  }
};

const adminInitiateAuthHandler = async (commands) => {
  const [_, username, password] = commands;

  try {
    validateUser(username, password);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    validateId(userPoolId);
    validateClient(clientId);

    log("Signing in.");
    const { ChallengeName, Session } = await adminInitiateAuth({
      clientId,
      userPoolId,
      username,
      password,
    });

    if (ChallengeName === "MFA_SETUP") {
      log("MFA setup is required.");
      return handleMfaSetup(Session, username);
    }

    if (ChallengeName === "SOFTWARE_TOKEN_MFA") {
      handleSoftwareTokenMfa(Session);
      log(`Run 'admin-respond-to-auth-challenge ${username} <totp>'`);
    }
  } catch (err) {
    log(err);
  }
};

export { adminInitiateAuthHandler };

const adminInitiateAuth = async ({
  clientId,
  userPoolId,
  username,
  password,
}) => {
```

```
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

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

  return client.send(command);
};

import { log } from "../../../../libs/utils/util-log.js";
import { adminRespondToAuthChallenge } from "../../../actions/admin-respond-to-
auth-challenge.js";
import { getFirstEntry } from "../../../../libs/utils/util-csv.js";
import { FILE_USER_POOLS } from "./constants.js";

const verifyUsername = (username) => {
  if (!username) {
    throw new Error(
      `Username is missing. It must be provided as an argument to the 'admin-
respond-to-auth-challenge' command.`
    );
  }
};

const verifyTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) is missing. It must be provided as an
 argument to the 'admin-respond-to-auth-challenge' command.`
    );
  }
};

const storeAccessToken = (token) => {
  process.env.AccessToken = token;
};

const adminRespondToAuthChallengeHandler = async (commands) => {
  const [_, username, totp] = commands;

  try {
    verifyUsername(username);
    verifyTotp(totp);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    const session = process.env.SESSION;

    const { AuthenticationResult } = await adminRespondToAuthChallenge({
      clientId,
      userPoolId,
      username,
      totp,
      session,
    });

    storeAccessToken(AuthenticationResult.AccessToken);

    log("Successfully authenticated.");
  } catch (err) {
    log(err);
  }
};
```

```
export { adminRespondToAuthChallengeHandler };

const respondToAuthChallenge = async ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = createClientForDefaultRegion(CognitoIdentityProviderClient);

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};

import { log } from "../../../../libs/utils/util-log.js";
import { verifySoftwareToken } from "../../../actions/verify-software-token.js";

const validateTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) must be provided to the 'validate-
software-token' command.`
    );
  }
};
const verifySoftwareTokenHandler = async (commands) => {
  const [_, totp] = commands;

  try {
    validateTotp(totp);

    log("Verifying TOTP.");
    await verifySoftwareToken(totp);
    log("TOTP Verified. Run 'admin-initiate-auth' again to sign-in.");
  } catch (err) {
    console.log(err);
  }
};

export { verifySoftwareTokenHandler };

const verifySoftwareToken = async (totp) => {
  const client = createClientForDefaultRegion(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);
};
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的以下主题。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
/**
 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 inOb = Scanner(System.`in`)
    val userName = inOb.nextLine()
    println(userName)

    println("*** Enter your password")
    val password: String = inOb.nextLine()

    println("*** Enter your email")
    val email = inOb.nextLine()

    println("*** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("*** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("*** Conformation code sent to $userName. Would you like to send a new
 code? (Yes/No)")
    val ans = inOb.nextLine()

    if (ans.compareTo("Yes") == 0) {
        println("*** Sending a new confirmation code")
        resendConfirmationCode(clientId, userName)
    }
    println("*** Enter the confirmation code that was emailed")
    val code = inOb.nextLine()
    confirmSignUp(clientId, code, userName)

    println("*** Rechecking the status of $userName in the user pool")
    getAdminUser(userName, poolId)

    val authResponse = checkAuthMethod(clientId, userName, password, poolId)
    val mySession = authResponse.session
    val newSession = getSecretForAppMFA(mySession)
    println("*** Enter the 6-digit code displayed in Google Authenticator")
    val myCode = inOb.nextLine()

    // Verify the TOTP and register for MFA.
    verifyTOTP(newSession, myCode)
    println("*** Re-enter a 6-digit code displayed in Google Authenticator")
    val mfaCode: String = inOb.nextLine()
    val authResponse1 = checkAuthMethod(clientId, userName, password, poolId)
    val session2 = authResponse1.session
```

```
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun checkAuthMethod(clientIdVal: String, userNameVal: String, passwordVal:
 String, userPoolIdVal: String): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

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

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

suspend fun resendConfirmationCode(clientIdVal: String?, userNameVal: String?) {
    val codeRequest = ResendConfirmationCodeRequest {
        clientId = clientIdVal
        username = userNameVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}

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

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

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

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(sessionVal: String?, codeVal: String?) {
    val tokenRequest = VerifySoftwareTokenRequest {
        userCode = codeVal
```

```
        session = sessionVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest = AssociateSoftwareTokenRequest {
        session = sessionVal
    }

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

suspend fun confirmSignUp(clientIdVal: String?, codeVal: String?, userNameVal:
 String?) {
    val signUpRequest = ConfirmSignUpRequest {
        clientId = clientIdVal
        confirmationCode = codeVal
        username = userNameVal
    }

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

suspend fun getAdminUser(userNameVal: String?, poolIdVal: String?) {
    val userRequest = AdminGetUserRequest {
        username = userNameVal
        userPoolId = poolIdVal
    }

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

suspend fun signUp(clientIdVal: String?, userNameVal: String?, passwordVal:
 String?, emailVal: String?) {
    val userAttrs = AttributeType {
        name = "email"
        value = emailVal
    }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest = SignUpRequest {
        userAttributes = userAttrsList
```

```
        username = userNameVal
        clientId = clientIdVal
        password = passwordVal
    }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的以下主题。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Python

适用于 Python (Boto3) 的 SDK

Note

在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

除了前面列出的步骤外，此示例还注册了一个 MFA 设备以通过 Amazon Cognito进行跟踪，并向您展示如何使用来自被跟踪设备的密码和信息进行登录。这样，就无需输入新的 MFA 代码。

```
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()
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的以下主。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers

- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 AWS SDK 的 Amazon Cognito 同步的代码示例

以下代码示例展示如何将 Amazon Cognito 同步与 AWS 软件开发工具包 (SDK) 一起使用。

操作是大型程序的代码摘录，必须在上下文中运行。虽然操作向您展示了如何调用具体服务函数，但您可以在其相关场景和跨服务示例中查看操作的上下文。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

代码示例

- 使用 AWS SDK 的 Amazon Cognito 同步的操作 (p. 488)
    - 使用 AWS SDK 获取向 Amazon Cognito 注册的身份池列表 (p. 488)

使用 AWS SDK 的 Amazon Cognito 同步的操作

以下代码示例演示如何使用 AWS SDK 执行各个 Amazon Cognito 同步操作。这些代码节选调用了 Amazon Cognito 同步 API，是必须在上下文中运行的较大型程序的代码节选。每个示例都包含一个指向 GitHub 的链接，您可以在其中找到有关设置和运行代码的说明。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon Cognito 同步 API 参考。

示例

- 使用 AWS SDK 获取向 Amazon Cognito 注册的身份池列表 (p. 488)

使用 AWS SDK 获取向 Amazon Cognito 注册的身份池列表

以下代码示例显示如何列出 Amazon Cognito 身份池。

Rust

SDK for Rust

> **Note**
>
> 本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

> **Note**
>
> 在 GitHub 上查看更多内容。在 AWS 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
async fn show_pools(client: &Client) -> Result<(), Error> {
```

```
    let response = client
        .list_identity_pool_usage()
        .max_results(10)
        .send()
        .await?;

    if let Some(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(())
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API 参考》中的 ListIdentityPoolUsage。

有关 AWS 软件开发工具包开发人员指南和代码示例的完整列表，请参阅 将此服务与 AWS SDK 结合使用 (p. 8)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

Amazon Cognito 中的安全性

AWS 十分重视云安全性。作为 AWS 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。

安全性是AWS和您的共同责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云的安全性 – AWS 负责保护在 AWS 云中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。作为 AWS 合规性计划 的一部分，第三方审计人员将定期测试和验证安全性。要了解适用于 Amazon Cognito 的合规性计划，请参阅合规性计划范围内的 AWS 服务。
- 云中的安全性 - 您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

该文档帮助您了解如何在使用 Amazon Cognito 时应用责任共担模式。它说明了如何配置 Amazon Cognito 以实现您的安全性和合规性目标。您还会了解如何使用其他AWS服务以帮助您监控和保护 Amazon Cognito 资源。

目录

- Amazon Cognito 中的数据保护 (p. 490)
- 适用于 Amazon Cognito 的 Identity and Access Management (p. 491)
- Amazon Cognito 中的日志记录和监控 (p. 507)
- Amazon Cognito 的合规性验证 (p. 530)
- Amazon Cognito 中的恢复能力 (p. 530)
- Amazon Cognito 中的基础设施安全性 (p. 531)
- Amazon Cognito 用户池中的配置和漏洞分析 (p. 531)
- Amazon Cognito 的 AWS 托管式策略 (p. 531)

Amazon Cognito 中的数据保护

AWS 责任共担模式应用于在 Amazon Cognito（Amazon Cognito）中保护数据。如该模式中所述，AWS 负责保护运行所有 AWS 云的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。此内容包括您所使用的AWS服务的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。

出于数据保护目的，我们建议您保护 AWS 账户凭证并使用 AWS Identity and Access Management (IAM) 设置单独的用户账户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用 Multi-Factor Authentication (MFA)。
- 使用 SSL/TLS 与 AWS 资源进行通信。
- 使用 AWS CloudTrail 设置 API 和用户活动日志记录。
- 使用 AWS 加密解决方案以及 AWS 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Simple Storage Service（Amazon S3）中的个人数据。

我们强烈建议您切勿将敏感的可识别信息（例如您客户的账号）放入自由格式字段（例如 Name（名称）字段）。这包括使用控制台、API、AWS CLI 或 AWS SDK 处理 Amazon Cognito 或其他AWS服务时。您输入到 Amazon Cognito 或其他服务中的任何数据都可能被选取以包含在诊断日志中。当您向外部服务器提供 URL 时，请勿在 URL 中包含凭证信息来验证您对该服务器的请求。

数据加密

数据加密通常分为两类：静态加密和传输中加密。

静态加密

Amazon Cognito 中的数据按照行业标准进行静态加密。

传输中加密

对 Amazon Cognito 的所有请求都必须使用传输层安全协议（TLS）发出。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

> Note
>
> Amazon Cognito 在内部加密客户内容，并且不支持客户提供的密钥。

适用于 Amazon Cognito 的 Identity and Access Management

AWS Identity and Access Management (IAM) 是一项 AWS 服务，可以帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）使用 Cognito 资源。IAM 是一项无需额外费用即可使用的 AWS 服务。

主题

- 受众 (p. 491)
- 使用身份进行身份验证 (p. 492)
- 使用策略管理访问 (p. 493)
- Amazon Cognito 如何与 IAM 配合使用 (p. 494)
- 适用于 Amazon Cognito 的基于身份的策略示例 (p. 500)
- Amazon Cognito 身份和访问问题排查 (p. 503)
- 对 Amazon Cognito 使用服务相关角色 (p. 504)

受众

使用 AWS Identity and Access Management（IAM）的方式因您可以在 Cognito 中执行的操作而异。

服务用户 – 如果您使用 Cognito 服务来完成工作，则您的管理员会为您提供所需的凭证和权限。当您使用更多 Cognito 功能来完成工作时，您可能需要更多权限。了解如何管理访问权限可帮助您向管理员请求适合的权限。如果您无法访问 Cognito 中的一项功能，请参阅Amazon Cognito 身份和访问问题排查 (p. 503)。

服务管理员 – 如果您在公司负责管理 Cognito 资源，则您可能具有 Cognito 的完全访问权限。您有责任确定您的服务用户应访问哪些 Cognito 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Cognito 搭配使用的更多信息，请参阅Amazon Cognito 如何与 IAM 配合使用 (p. 494)。

IAM 管理员 - 如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 Cognito 的访问的详细信息。要查看您可在 IAM 中使用的 Cognito 基于身份的策略示例，请参阅适用于 Amazon Cognito 的基于身份的策略示例 (p. 500)。

使用身份进行身份验证

身份验证是您使用身份凭证登录 AWS 的方法。您必须作为 AWS 账户根用户、IAM 用户或通过代入 IAM 角色进行身份验证（登录到 AWS）。

您可以使用通过身份源提供的凭证以联合身份登录到 AWS。AWS IAM Identity Center (successor to AWS Single Sign-On)(IAM Identity Center) 用户、您公司的单点登录身份验证以及您的 Google 或 Facebook 凭证都是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当您使用联合身份验证访问 AWS 时，您就是在间接代入角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录到 AWS 的更多信息，请参阅《AWS 登录 用户指南》中的 如何登录到您的 AWS 账户。

如果您以编程方式访问 AWS，则 AWS 将提供软件开发工具包 (SDK) 和命令行界面（CLI），以便使用您的凭证对您的请求进行加密签名。如果您不使用 AWS 工具，则必须自行对请求签名。有关使用推荐的方法自行对请求签名的更多信息，请参阅《AWS 一般参考》中的 签名版本 4 签名流程。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如，AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center (successor to AWS Single Sign-On) 用户指南》中的 多重身份验证 和《IAM 用户指南》中的 在 AWS 中使用多重身份验证（MFA）。

AWS 账户根用户

当您创建 AWS 账户时，最初使用的是一个对账户中所有 AWS 服务和资源拥有完全访问权限的登录身份。此身份称为 AWS 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅 AWS Account Management 参考指南 中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）结合使用联合身份验证和身份提供程序，以使用临时凭证来访问 AWS 服务。

联合身份是来自企业用户目录、Web 身份提供程序、AWS Directory Service、Identity Center 目录的用户，或任何使用通过身份源提供的凭证来访问 AWS 服务 的用户。当联合身份访问 AWS 账户时，他们代入角色，而角色提供临时凭证。

要集中管理访问权限，我们建议您使用 AWS IAM Identity Center (successor to AWS Single Sign-On)。您可以在 IAM Identity Center 中创建用户和组，也可以连接并同步到您自己的身份源中的一组用户和组以跨所有 AWS 账户和应用程序使用。有关 IAM Identity Center 的信息，请参阅《AWS IAM Identity Center (successor to AWS Single Sign-On) 用户指南》中的什么是 IAM Identity Center？

IAM 用户和组

IAM 用户是AWS 账户内对某个人员或应用程序具有特定权限的一个身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色是AWS 账户中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 AWS Management Console 中暂时代入 IAM 角色。您可以调用 AWS CLI 或 AWS API 操作或使用自定义 URL 以担任角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- Federated user access（联合用户访问）– 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的 为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center (successor to AWS Single Sign-On) 用户指南》中的 权限集。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户访问 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 – 某些 AWS 服务使用其它 AWS 服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Simple Storage Service（Amazon S3）中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 主体权限 – 当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为主体。策略向主体授予权限。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中触发另一个操作。在这种情况下，您必须具有执行这两个操作的权限。要查看某个操作是否需要策略中的其他相关操作，请参阅《服务授权参考》中的 Amazon Cognito 的操作、资源和条件键。
  - 服务角色 – 服务角色是服务代表您在您的账户中执行操作而担任的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的 创建向 AWS 服务 委派权限的角色。
  - 服务相关角色 – 服务相关角色是与 AWS 服务关联的一种服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 AWS 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅 IAM 用户指南中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您将创建策略并将其附加到 AWS 身份或资源，以控制 AWS 中的访问。策略是 AWS 中的对象；在与身份或资源相关联时，策略定义它们的权限。在主体（用户、根用户或角色会话）发出请求时，AWS 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 AWS 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南 中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM policy。然后，管理员可以向角色添加 IAM policy，并且用户可以代入角色。

IAM policy 定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 AWS Management Console、AWS CLI 或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的 创建 IAM policy。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管式策略是可以附加到AWS 账户中的多个用户、组和角色的独立策略。托管式策略包括 AWS 托管式策略和客户托管式策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅 IAM 用户指南 中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。主体可以包括账户、用户、角色、联合身份用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用来自 IAM 的 AWS 托管式策略。

访问控制列表 (ACL)

访问控制列表 (ACL) 控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Simple Storage Service（Amazon S3）、AWS WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的访问控制列表 (ACL) 概览。

其它策略类型

AWS 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 – 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅 IAM 用户指南中的 IAM 实体的权限边界。
- 服务控制策略 (SCP) – SCP 是 JSON 策略，指定了组织或组织单位 (OU) 在 AWS Organizations 中的最大权限。AWS Organizations 服务可以分组和集中管理您的企业拥有的多个 AWS 账户 账户。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体（包括每个 AWS 账户根用户）的权限。有关 Organizations 和 SCP 的更多信息，请参阅 AWS Organizations 用户指南中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解 AWS 如何确定在涉及多种策略类型时是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

Amazon Cognito 如何与 IAM 配合使用

在使用 IAM 管理对 Cognito 的访问权限之前，您应该了解哪些 IAM 功能可用于 Cognito。

可与 Amazon Cognito 一起使用的 IAM 功能

IAM 功能	Cognito 支持
基于身份的策略 (p. 495)	是
基于资源的策略 (p. 495)	否
策略操作 (p. 496)	是
策略资源 (p. 497)	是
策略条件键 (p. 498)	是
ACL (p. 499)	否
ABAC（策略中的标签） (p. 499)	部分
临时凭证 (p. 499)	是
主体权限 (p. 500)	否
服务角色 (p. 500)	是
服务相关角色 (p. 500)	是

要大致了解 Amazon Cognito 和其他AWS服务如何与大多数 IAM 功能一起使用，请参阅《IAM 用户指南》中的与 IAM 一起使用的AWS服务。

适用于 Cognito 的基于身份的策略

支持基于身份的策略	是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM policy。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

适用于 Cognito 的基于身份的策略示例

要查看 Cognito 基于身份的策略的示例，请参阅适用于 Amazon Cognito 的基于身份的策略示例 (p. 500)。

Cognito 内基于资源的策略

支持基于资源的策略。	否

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在

什么条件下执行。您必须在基于资源的策略中指定主体。主体可以包括账户、用户、角色、联合身份用户或 AWS 服务。

要启用跨账户存取，您可以将整个账户或其它账户中的 IAM 实体指定为基于资源的策略中的主体。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当主体和资源处于不同的AWS 账户中时，则信任账户中的 IAM 管理员还必须授予主体实体（用户或角色）对资源的访问权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。

Cognito 的策略操作

支持策略操作	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么 条件 下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行相关操作的权限。

如查看 Cognito 操作的列表，请参阅《服务授权参考》中 Amazon Cognito 定义的操作。

Cognito 中的策略操作在操作前使用以下前缀：

```
cognito-identity
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "cognito-identity:action1",
      "cognito-identity:action2"
         ]
```

签名与未签名的 API

使用 AWS 凭证签署 Amazon Cognito API 请求时，您可以在 AWS Identity and Access Management (IAM) 策略中限制它们。您必须使用 AWS 凭证签署的 API 请求包括使用 `AdminInitiateAuth` 进行服务器端登录，以及创建、查看或修改您的 Amazon Cognito 资源的操作（如 `UpdateUserPool`）。有关已签署 API 请求的更多信息，请参阅签署 AWS API 请求。

由于 Amazon Cognito 是一款消费者身份产品，适用于您想要向公众开放的应用程序，因此您可以访问以下未签署的 API。您的应用程序针对您的用户和潜在用户发出这些 API 请求。有些 API 不需要事先授权，例如 `InitiateAuth` 可用于启动新的身份验证会话。有些 API 使用访问令牌或会话密钥进行授权，例如 `VerifySoftwareToken` 可为已有经过身份验证的会话的用户完成 MFA 设置。未签署但已授权的 Amazon Cognito 用户池 API 支持请求语法中的 `Session` 或 `AccessToken` 参数，如 Amazon Cognito API 参考中所示。未签署的 Amazon Cognito 身份 API 支持 `IdentityId` 参数，如 Amazon Cognito 联合身份 API 参考中所示。

有关 Amazon Cognito 用户群体 API 操作的授权模式和角色的更多信息，请参阅Amazon Cognito 用户群体经过身份验证和未经身份验证的 API 操作 (p. 48)。

Amazon Cognito 身份池 API 操作

- `GetId`

- GetOpenIdToken
- GetCredentialsForIdentity
- UnlinkIdentity

Amazon Cognito 用户群体 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

要查看 Cognito 基于身份的策略的示例，请参阅适用于 Amazon Cognito 的基于身份的策略示例 (p. 500)。

Cognito 的策略资源

支持策略资源	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符（*）指示语句应用于所有资源。

```
"Resource": "*"
```

Amazon Resource Name (ARN)

适用于 Amazon Cognito 联合身份的 ARN

在 Amazon Cognito 身份池（联合身份）中，您可以使用如下例所示的 Amazon Resource Name（ARN）格式限制 IAM 用户对特定身份池的访问权限。有关 ARN 的更多信息，请参阅 IAM 标识符。

```
arn:aws:cognito-identity:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

Amazon Cognito Sync 的 ARN

在 Amazon Cognito Sync 中，客户还可以按身份池 ID、身份 ID 和数据集名称限制访问。

对于在身份池上运行的 API，身份池 ARN 格式与 Amazon Cognito 联合身份的相同，但前者的服务名称是 `cognito-sync`，而不是 `cognito-identity`：

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

对于在单个身份池上运行的 API (例如 `RegisterDevice`)，您可以通过以下 ARN 格式引用单个身份：

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/identity/IDENTITY_ID
```

有关在数据集上运行的 API (例如 `UpdateRecords` 和 `ListRecords`)，您可以使用以下 ARN 格式引用单个数据集：

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/identity/IDENTITY_ID/
dataset/DATASET_NAME
```

适用于 Amazon Cognito 用户池的 ARN

对于 Amazon Cognito 中的您的用户群体，您可以使用以下 ARN 格式限制用户对特定用户群体的访问权限：

```
arn:aws:cognito-idp:REGION:ACCOUNT_ID:userpool/USER_POOL_ID
```

有关 Cognito 资源类型及其 ARN 的列表，请参阅《服务授权引用》中 Amazon Cognito 定义的资源。要了解您可以使用哪些操作指定每个资源的 ARN，请参阅 Amazon Cognito 定义的操作。

要查看 Cognito 基于身份的策略的示例，请参阅适用于 Amazon Cognito 的基于身份的策略示例 (p. 500)。

Cognito 的策略条件键

支持特定于服务的策略条件键	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则 AWS 使用逻辑 OR 运算来评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM policy 元素：变量和标签。

AWS支持全局条件键和特定于服务的条件键。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的 AWS 全局条件上下文键。

有关 Cognito 条件键的列表，请参阅《服务授权参考》中 Amazon Cognito 的条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon Cognito 定义的操作。

要查看 Cognito 基于身份的策略的示例，请参阅适用于 Amazon Cognito 的基于身份的策略示例 (p. 500)。

Cognito 中的访问控制列表（ACL）

支持 ACL	否

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

使用 Cognito 的基于属性的访问控制（ABAC）

支持 ABAC（策略中的标签）	部分

基于属性的访问控制（ABAC）是一种授权策略，该策略基于属性来定义权限。在 AWS 中，这些属性称为标签。您可以将标签附加到 IAM 实体（用户或角色）以及AWS资源。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为 Yes（是）。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为 Partial（部分）。

有关 ABAC 的更多信息,请参阅《IAM 用户指南》中的什么是 ABAC？。要查看包含设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问控制（ABAC）。

将临时凭证用于 Cognito

支持临时凭证	是

某些 AWS 服务在您使用临时凭证登录时无法正常工作。有关更多信息，包括 AWS 服务与临时凭证配合使用，请参阅 IAM 用户指南中的使用 IAM 的 AWS 服务。

如果您不使用用户名和密码而用其它方法登录到AWS Management Console，则使用临时凭证。例如，当您使用贵公司的单点登录（SSO）链接访问AWS时，该过程将自动创建临时凭证。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的切换到角色（控制台）。

您可以使用 AWS CLI 或者 AWS API 创建临时凭证。之后，您可以使用这些临时凭证访问AWS。AWS建议您动态生成临时凭证，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

Cognito 的跨服务主体权限

支持主体权限	否

当您使用 IAM 用户或角色在AWS中执行操作时，您将被视为主体。策略向主体授予权限。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中触发另一个操作。在这种情况下，您必须具有执行这两个操作的权限。要查看某个操作是否需要策略中的其他相关操作，请参阅《服务授权参考》中的 Amazon Cognito 的操作、资源和条件键。

Cognito 的服务角色

支持服务角色	是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的 创建向 AWS 服务 委派权限的角色。

有关 Cognito 服务角色的详细信息，请参阅激活推送同步 (p. 298)和推送同步 (p. 389)。

> Warning
>
> 更改服务角色的权限可能会破坏 Cognito 的功能。仅当 Cognito 提供相关指导时才编辑服务角色。

Cognito 的服务相关角色

支持服务相关角色	是

服务相关角色是一种与 AWS 服务相关的服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 AWS 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理 Cognito 服务相关角色的详细信息，请参阅对 Amazon Cognito 使用服务相关角色 (p. 504)。

适用于 Amazon Cognito 的基于身份的策略示例

默认情况下，用户和角色没有创建或修改 Cognito 资源的权限。他们也无法使用 AWS Management Console、AWS Command Line Interface（AWS CLI）或 AWS API 执行任务。要用户用户在其需要的资源上执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的 创建 IAM policy。

有关 Cognito 定义的操作和资源类型的详细信息，包括每种资源类型的 ARN 格式，请参阅服务授权参考 中的 Amazon Cognito 的操作、资源和条件键。

主题

- 策略最佳实践 (p. 501)
- 使用 Cognito 控制台 (p. 501)
- 允许用户查看他们自己的权限 (p. 501)
- 限制对特定身份池的控制台访问权限 (p. 502)
- 允许池中的所有身份访问特定数据集 (p. 502)

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 Cognito 资源。这些操作可能会使 AWS 账户 产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- AWS 托管策略及转向最低权限许可入门 - 要开始向用户和工作负载授予权限，请使用 AWS 托管策略来为许多常见使用场景授予权限。您可以在 AWS 账户 中找到这些策略。我们建议通过定义特定于您的使用场景的 AWS 客户管理型策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略或工作职能的 AWS 托管策略。
- 应用最低权限 – 在使用 IAM policy 设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM policy 中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果通过特定 AWS 服务（例如 AWS CloudFormation）使用服务操作，您还可以使用条件来授予对服务操作的访问权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM policy，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM policy语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。
- Require multi-factor authentication（MFA）[需要多重身份验证（MFA）] – 如果您所处的场景要求您的 AWS 账户 中有 IAM 用户或根用户，请启用 MFA 来提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实践的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

> Note
>
> 当您查看和修改 Amazon Cognito 资源时，Amazon Cognito 控制台的原始版本和新版本具有不同的底层行为。如果您仅在条件 `aws:ViaAWSService` 为 true 时授予对 `cognito-idp` 服务前缀下的操作的权限，受影响的 IAM 主体可能在原始控制台中对 Amazon Cognito 资源有效，但在 Amazon Cognito 控制台中无效。要在 Amazon Cognito 控制台中有效，请不要在您的 IAM policy 中对 Amazon Cognito 权限设置 `aws:ViaAWSService` 条件。

使用 Cognito 控制台

要访问 Amazon Cognito 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关您AWS 账户中 Cognito 资源的详细信息。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于只需要调用 AWS CLI 或 AWS API 的用户，无需为其提供最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍可使用 Cognito 控制台，请同时将 Cognito `ConsoleAccess` 或 `ReadOnly` AWS 管理策略添加到实体。有关更多信息，请参阅 IAM 用户指南中的为用户添加权限。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上完成此操作或者以编程方式使用 AWS CLI 或 AWS API 所需的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
```

```
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

限制对特定身份池的控制台访问权限

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

允许池中的所有身份访问特定数据集

```
{
```

```
  "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 身份和访问问题排查

以下信息可帮助您诊断和修复在使用 Cognito 和 IAM 时可能遇到的常见问题。

主题

- 我没有在 Cognito 中执行操作的权限 (p. 503)
- 我无权执行 iam:PassRole (p. 503)
- 我是管理员并希望允许其他人访问 Cognito (p. 504)
- 我想要允许我的AWS账户之外的用户访问我的 Cognito 资源 (p. 504)

我没有在 Cognito 中执行操作的权限

如果您收到一个错误，表明您无权执行某个操作，则必须更新策略以允许您执行该操作。

当 `mateojackson` IAM 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不拥有虚构 `cognito-identity:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: cognito-
identity:GetWidget on resource: my-example-widget
```

在此情况下，必须更新 `mateojackson` 用户的策略，以允许使用 `cognito-identity:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。

如果您需要帮助，请联系您的 AWS 管理员。管理员是向您提供登录凭证的人。

我无权执行 iam:PassRole

如果您收到一个错误，表明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给 Cognito。

有些 AWS 服务允许您将现有角色传递到该服务，而不是创建新服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 Cognito 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。管理员是向您提供登录凭证的人。

我是管理员并希望允许其他人访问 Cognito

要允许其他人访问 Cognito，您必须为需要访问权限的人员或应用程序创建一个 IAM 实体（用户或角色）。它们将使用该实体的凭证访问 AWS。然后，您必须将策略附加到实体，以便在 Cognito 中向其授予正确的权限。

要立即开始使用，请参阅 IAM 用户指南中的创建您的第一个 IAM 委派用户和组。

我想要允许我的AWS账户之外的用户访问我的 Cognito 资源

您可以创建一个角色，以便其它账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表 (ACL) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 Cognito 是否支持这些功能，请参阅Amazon Cognito 如何与 IAM 配合使用 (p. 494)。
- 要了解如何为您拥有的 AWS 账户 中的资源提供访问权限，请参阅 IAM 用户指南中的为您拥有的另一个 AWS 账户 中的 IAM 用户提供访问权限。
- 要了解如何为第三方AWS 账户提供您的资源的访问权限，请参阅 IAM 用户指南中的为第三方拥有的AWS 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南中的为经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅 IAM 用户指南中的 IAM 角色与基于资源的策略有何不同。

对 Amazon Cognito 使用服务相关角色

Amazon Cognito 使用 AWS Identity and Access Management（IAM）服务相关角色。服务相关角色是一种独特类型的 IAM 角色，它与 Amazon Cognito 直接相关。服务相关角色由 Amazon Cognito 预定义，并包含服务代表您调用其他AWS服务所需的所有权限。

服务相关角色可让您更轻松地设置 Amazon Cognito，因为您不必手动添加必要的权限。Amazon Cognito 定义其服务相关角色的权限，除非另外定义，否则只有 Amazon Cognito 可以代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其它 IAM 实体的权限策略。

只有在首先删除相关资源后，才能删除服务相关角色。这将保护您的 Amazon Cognito 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其它服务的信息，请参阅使用 IAM 的 AWS 服务并查找 Service-Linked Role（服务相关角色）列中显示为 Yes（是）的服务。请选择 Yes 与查看该服务的服务相关角色文档的链接。

Amazon Cognito 的服务相关角色权限

Amazon Cognito 使用下列服务相关角色：

- AWSServiceRoleForAmazonCognitoIdpEmailService — 允许 Amazon Cognito 用户池服务使用您的 Amazon SES 身份来发送电子邮件。

- AWSServiceRoleForAmazonCognitoIdp — 允许 Amazon Cognito 用户池为您的 Amazon Pinpoint 项目发布事件和配置端点。

AWSServiceRoleForAmazonCognitoIdpEmailService

AWSServiceRoleForAmazonCognitoIdpEmailService 服务相关角色信任以下服务代入该角色：

- email.cognito-idp.amazonaws.com

角色权限策略允许 Amazon Cognito 对指定资源完成以下操作：

AWSServiceRoleForAmazonCognitoIdpEmailService 允许的操作：

- 操作：ses:SendEmail 和 ses:SendRawEmail
- 资源: *

此策略拒绝 Amazon Cognito 对指定资源完成以下操作的功能：

拒绝的操作

- 操作：ses:List*
- 资源: *

凭借这些权限，Amazon Cognito 只能使用 Amazon SES 中经过验证的电子邮件地址向用户发送电子邮件。当您的用户在客户端应用程序中针对用户池执行特定操作（如注册或重置密码）时，Amazon Cognito 将向用户发送电子邮件。

必须配置权限，允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅 IAM 用户指南中的服务相关角色权限。

AWSServiceRoleForAmazonCognitoIdp

AWSServiceRoleForAmazonCognitoIdp 服务相关角色信任以下服务代入角色：

- email.cognito-idp.amazonaws.com

角色权限策略允许 Amazon Cognito 对指定资源完成以下操作：

AWSServiceRoleForAmazonCognitoIdp 允许的操作：

- 操作：cognito-idp:Describe
- 资源: *

有了此权限，Amazon Cognito 可以为您调用 Describe Amazon Cognito API 操作。

> **Note**
>
> 当您将 Amazon Cognito 与采用 createUserPoolClient 和 updateUserPoolClient 的 Amazon Pinpoint 集成时，资源权限将作为内联策略添加到 SLR 中。内联策略将提供 mobiletargeting:UpdateEndpoint 和 mobiletargeting:PutEvents 权限。这些权限允许 Amazon Cognito 发布事件并为与 Cognito 集成的 Pinpoint 项目配置端点。

创建适用于 Amazon Cognito 的服务相关角色

无需手动创建服务相关角色。当您在AWS Management Console、AWS CLI 或 Amazon Cognito API 中将用户池配置为使用 Amazon SES 配置处理邮件送达时，Amazon Cognito 会为您创建与服务相关的角色。

如果删除此服务相关角色，然后需要再次创建，可以使用相同流程在账户中重新创建此角色。当您将用户池配置为使用 Amazon SES 配置去处理邮件送达时，Amazon Cognito 会为您创建与服务相关的角色。

在 Amazon Cognito 可以创建此角色之前，您用来设置用户池的 IAM 权限必须包含 `iam:CreateServiceLinkedRole` 操作。有关更新 IAM 中权限的更多信息，请参阅《IAM 用户指南》中的更改 IAM 用户的权限。

编辑适用于 Amazon Cognito 的服务相关角色

您无法在 AWS Identity and Access Management 中编辑 Amazon CognitoIDP 或 AmazonCognitoIDPEmailService 服务相关角色。在创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。不过，您可以使用 IAM 编辑角色的说明。有关更多信息，请参阅 IAM 用户指南中的编辑服务相关角色。

删除适用于 Amazon Cognito 的服务相关角色

如果您不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。如果您删除角色，则只应保留 Amazon Cognito 主动监控或维护的实体。在删除 AmazonCognitoIdp 或 AmazonCognitoIdpEmailService 服务相关角色之前，您必须对使用该角色的每个用户池执行以下任一操作：

- 删除该用户池。
- 更新用户池中的电子邮件设置以使用默认的电子邮件功能。默认设置不使用服务相关角色。

请记住在具有使用该角色的用户池的每个 AWS 区域 中执行该操作。

> **Note**
>
> 如果在您尝试删除资源时，Amazon Cognito 服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，请等待几分钟后重试。

删除 Amazon Cognito 用户池

1. 登录AWS Management Console并打开 Amazon Cognito 控制台（https://console.aws.amazon.com/cognito）。
2. 选择 Manage User Pools（管理用户池）。
3. 在 Your User Pools (您的用户池) 页面上，选择要删除的用户池。
4. 选择 Delete pool（删除池）。
5. 在 Delete user pool (删除用户池) 窗口中，键入 **`delete`**，然后选择 Delete pool (删除池)。

更新 Amazon Cognito 用户池以使用默认电子邮件功能

1. 登录AWS Management Console并打开 Amazon Cognito 控制台（https://console.aws.amazon.com/cognito）。
2. 选择 Manage User Pools（管理用户池）。
3. 在 Your User Pools (您的用户池) 页面上，选择要更新的用户池。
4. 在左侧导航菜单中，选择 Message customizations（管理自定义）。
5. 在 Do you want to send emails through your Amazon SES Configuration? (是否要通过 Amazon SES 配置发送电子邮件?) 下，选择 No - Use Cognito (Default) (否 -使用 Cognito (默认))。
6. 当您完成设置您的电子邮件账户选项时，选择 Save changes（保存更改）。

使用 IAM 手动删除服务相关角色

使用 IAM 控制台、AWS CLI 或 AWS API 来删除 AmazonCognitoIdp 或 AmazonCognitoIdpEmailService 服务相关角色。有关更多信息，请参阅 IAM 用户指南中的删除服务相关角色。

Amazon Cognito 服务相关角色支持的区域

Amazon Cognito 支持在已推出该服务的所有 AWS 区域中使用服务相关角色。有关更多信息，请参阅AWS 区域 和端点。

Amazon Cognito 中的日志记录和监控

监控是保持 Amazon Cognito 和您的其他AWS解决方案的可靠性、可用性和性能的重要方面。Amazon Cognito 目前支持下列 AWS 服务，这样您便可以监控您的组织和组织内部的活动。

- AWS CloudTrail – 借助 CloudTrail，您可以捕获来自 Amazon Cognito 控制台的 API 调用，以及从代码调用到 Amazon Cognito API 运营的 API 操作。例如，在用户进行身份验证时，CloudTrail 会记录请求中的 IP 地址、发出请求的人以及发出请求的时间等详细信息。
- Amazon CloudWatch 指标 – 利用 CloudWatch 指标，您可以几乎实时地监控和报告，并在发生事件时采取自动措施。例如，您可以基于提供的指标创建 CloudWatch 控制面板来监控 Amazon Cognito 用户池，也可以基于提供的指标创建 CloudWatch 警报，以便在超过设定的阈值时通知您。
- Amazon CloudWatch Logs Insights – 通过 CloudWatch Logs Insights，您可以将 CloudTrail 配置为将事件发送到 CloudWatch，从而监控 Amazon Cognito CloudTrail 日志文件。

主题

- 跟踪 CloudWatch 和 Service Quotas 中的配额和使用情况 (p. 507)
- 使用 AWS CloudTrail 记录 Amazon Cognito API 调用 (p. 513)

跟踪 CloudWatch 和 Service Quotas 中的配额和使用情况

您可以使用 Amazon CloudWatch 或使用 Service Quotas 监控 Amazon Cognito 用户池。CloudWatch 收集和处理原始数据，并将数据处理为便于读取的近乎实时的指标。在 CloudWatch 中，可以设置用于监测特定阈值的告警，并在达到相应阈值时发送通知或执行操作。要针对 Service Quotas 创建 CloudWatch 告警，请参阅创建 CloudWatch 告警。Amazon Cognito 指标每隔五分钟提供一次。有关 CloudWatch 中的保留期的更多信息，请访问 Amazon CloudWatch 常见问题页面。

您可以使用 Service Quotas 查看和管理 Amazon Cognito 用户池配额使用情况。Service Quotas 控制台具有三个功能：查看服务配额、请求提高服务配额以及查看当前利用率。您可以使用第一个功能来查看配额，并查看配额是否可调节。您可以使用第二个功能请求增加 Service Quotas。您可以使用最后一个功能查看配额利用率。此功能仅在您的账户已激活一段时间后才可用。有关在 Service Quotas 控制台中查看配额的更多信息，请参阅查看 Service Quotas。

> **Note**
>
> Amazon Cognito 指标每 5 分钟提供一次。有关 CloudWatch 中的保留期的更多信息，请访问 Amazon CloudWatch 常见问题页面。

主题

- Amazon Cognito 用户池指标 (p. 508)
- Amazon Cognito 用户池的维度 (p. 512)
- 使用 Service Quotas 控制台来跟踪指标 (p. 512)
- 使用 CloudWatch 控制台跟踪指标 (p. 513)
- 为配额创建 CloudWatch 告警 (p. 513)

Amazon Cognito 用户池指标

下表列出了对 Amazon Cognito 用户池可用的指标。

Note

控制台中不会显示在过去两周内没有任何新数据点的指标。当您在控制台中 All metrics (所有指标) 选项卡的搜索框中输入其指标名称或维度名称时，它们也不会显示。此外，它们不会在 list-metrics 命令的结果中返回。检索这些指标的最佳方法是在 AWS CLI 中使用 `get-metric-data` 或者 `get-metric-statistics` 命令。

指标	描述
`SignUpSuccesses`	提供向 Amazon Cognito 用户池发出的成功用户注册请求的总数。一个成功的用户注册请求会产生值 1，而一个不成功的请求会产生值 0。受限制的请求也会被视为不成功的请求，因此，一个受限制的请求也将产生计数 0。 要查找成功的用户注册请求的百分比，请对此指标使用 `Average` 统计数据。要计算用户注册请求的总数，请对此指标使用 `Sample Count` 统计数据。要计算成功的用户注册请求的总数，请对此指标使用 `Sum` 统计数据。要计算失败的用户注册请求的总数，请使用 CloudWatch `Math` 表达式并从 `Sample Count` 统计数据中减去 `Sum` 统计数据。 为每个用户池客户端的用户池发布此指标。如果用户注册由管理员执行，则以 `Admin` 身份将指标与用户池客户端一起发布。 请注意，不会针对用户导入和用户迁移案例发出此指标。 指标维度：`UserPool`、`UserPoolClient` 单位：计数
`SignUpThrottles`	提供向 Amazon Cognito 用户池发出的受限的用户注册请求的总数。当用户注册请求受到限制时，将发布计数 1。 要计算受限制的用户注册请求的总数，请对此指标使用 `Sum` 统计数据。 为每个客户端的每个用户池发布此指标。如果受限制的请求由管理员发出，则以 `Admin` 身份将指标与用户池客户端一起发布。 指标维度：`UserPool`、`UserPoolClient` 单位：计数
`SignInSuccesses`	提供向 Amazon Cognito 用户池发出的成功的用户身份验证请求的总数。在向用户颁发身份验证令牌时，用户身份验证将被视为成功。一个成功的身份验证会产生值 1，而一个不成功的请求会产生值 0。受限制的请求也会被视为不成功的请求，因此，一个受限制的请求也将产生计数 0。

指标	描述
	要查找成功的用户身份验证请求的百分比，请对此指标使用 Average 统计数据。要计算用户身份验证请求的总数，请对此指标使用 Sample Count 统计数据。要计算成功的用户身份验证请求的总数，请对此指标使用 Sum 统计数据。要计算失败的用户身份验证请求的总数，请使用 CloudWatch Math 表达式并从 Sample Count 统计数据中减去 Sum 统计数据。 为每个客户端的每个用户池发布此指标。如果请求中提供了无效的用户池客户端，则指标中的相应用户池客户端值将包含固定值 Invalid，而不是请求中发送的实际无效值。 请注意，Amazon Cognito 令牌刷新请求并未包含在此指标中。有一个用于提供 Refresh 令牌统计数据的单独指标。 指标维度：UserPool、UserPoolClient 单位：计数
SignInThrottles	提供向 Amazon Cognito 用户池发出的受限制的用户身份验证请求的总数。当身份验证请求受到限制时，将发布计数 1。 要计算受限制的用户身份验证请求的总数，请对此指标使用 Sum 统计数据。 为每个客户端的每个用户池发布此指标。如果请求中提供了无效的用户池客户端，则指标中的相应用户池客户端值将包含固定值 Invalid，而不是请求中发送的实际无效值。 Amazon Cognito 令牌刷新请求并未包含在此指标中。有一个用于提供 Refresh 令牌统计数据的单独指标。 指标维度：UserPool、UserPoolClient 单位：计数

指标	描述
TokenRefreshSuccesses	提供向 Amazon Cognito 用户池发出的成功的 Amazon Cognito 令牌刷新请求的总数。一个成功的 Amazon Cognito 令牌刷新请求会产生值 1，而一个不成功的请求会产生值 0。受限制的请求也会被视为不成功的请求，因此，一个受限制的请求也将产生计数 0。 要查找成功的 Amazon Cognito令牌刷新请求的百分比，请对此指标使用 Average 统计数据。要计算 Amazon Cognito 令牌刷新请求的总数，请对此指标使用 Sample Count 统计数据。要计算成功的 Amazon Cognito 令牌刷新请求的总数，请对此指标使用Sum统计数据。要计算失败的 Amazon Cognito 令牌刷新请求的总数，请使用 CloudWatch Math 表达式并从 Sample Count 统计数据中减去 Sum 统计数据。 按每个用户池客户端发布此指标。如果请求中有无效的用户池客户端，则用户池客户端值包含固定值 Invalid。 指标维度：UserPool、UserPoolClient 单位：计数
TokenRefreshThrottles	提供向 Amazon Cognito 用户池发出的受限制 Amazon Cognito 令牌刷新请求的总数。当 Amazon Cognito 令牌刷新请求受到限制时，将发布计数 1。 要计算受限制的 Amazon Cognito 令牌刷新请求的总数，请对此指标使用 Sum 统计数据。 为每个客户端的每个用户池发布此指标。如果请求中提供了无效的用户池客户端，则指标中的相应用户池客户端值将包含固定值 Invalid，而不是请求中发送的实际无效值。 指标维度：UserPool、UserPoolClient 单位：计数

指标	描述
FederationSuccesses	提供向 Amazon Cognito 用户池发出的成功的联合身份验证请求的总数。当 Amazon Cognito 向用户颁发身份验证令牌时，身份联合验证被视为成功。一个成功的联合身份验证请求会产生值 1，而一个不成功的请求会产生值 0。节流的请求以及生成授权码但没有令牌的请求所生成的值为 0。 要查找成功的联合身份验证请求的百分比，请对此指标使用 Average 统计数据。要计算联合身份验证请求的总数，请对此指标使用 Sample Count 统计数据。要计算成功的联合身份验证请求的总数，请对此指标使用 Sum 统计数据。要计算失败的联合身份验证请求的总数，请使用 CloudWatch Math 表达式并从 Sample Count统计数据中减去 Sum 统计数据。 指标维度：UserPool、UserPoolClient、IdentityProvider 单位：计数
FederationThrottles	提供向 Amazon Cognito 用户池发出的受限制的联合身份验证请求的总数。当联合身份验证请求受到限制时，将发布计数 1。 要计算受限制的联合身份验证请求的总数，请对此指标使用 Sum 统计数据。 指标维度：UserPool、UserPoolClient、IdentityProvider 单位：计数
CallCount	提供客户发出的与类别相关的调用总数。此指标包括所有调用，如受限制的调用、失败的调用和成功的调用。 此指标在用量 nameSpace 中提供。 类别配额针对账户和区域中所有用户池的每个AWS账户强制实施。 您可以使用此指标的 Sum 统计数据计算调用总数。 指标维度：服务、类型、资源、类 单位：计数

指标	描述
ThrottleCount	提供与类别相关的受限调用总数。 此指标在用量 nameSpace 中提供。 此指标已在账户级别发布。 您可以使用此指标的 Sum 统计数据计算某个类别中的调用总数。 指标维度：服务、类型、资源、类 单位：计数

Amazon Cognito 用户池的维度

以下维度用于优化由 Amazon Cognito 发布的用量指标。维度仅适用于 CallCount 和 ThrottleCount 指标。

维度	描述
服务	包含该资源的 AWS 服务的名称。对于 Amazon Cognito 用量指标，此维度的值为 Cognito user pool。
类型	正在报告的实体的类型。Amazon Cognito 用量指标的唯一有效值为 API。
资源	正在运行的资源的类型。唯一的有效值是类别名。
类	所跟踪的资源的类。Amazon Cognito 不使用类维度。

使用 Service Quotas 控制台来跟踪指标

您可以使用“服务限额”从一个中心位置查看和管理您的 Amazon Cognito 用户群体配额。您可以使用 Service Quotas 控制台查看具体配额的详细信息、监控配额利用率以及请求增加配额。对于某些限额类型，您可以创建 CloudWatch 警报来跟踪您的限额利用率。要详细了解您可以跟踪哪些 Amazon Cognito 指标，请参阅跟踪配额使用量 (p. 541)。

要查看 Amazon Cognito 用户池 Service Quotas 利用率，请完成以下步骤。

1. 打开 Service Quotas 控制台。
2. 在导航窗格中，选择 AWS services (亚马逊云科技服务)。
3. 从 AWS services (亚马逊云科技服务) 列表中，在搜索字段中输入 Amazon Cognito 用户池。此时将显示服务配额页面。
4. 选择支持 CloudWatch 监控的配额。例如，选择 Rate of UserAuthentication requests。
5. 向下滚动到 Monitoring (监控)。此部分仅显示支持 CloudWatch 监控的配额。
6. 在 Monitoring (监控) 中，您可以在图表中查看当前服务配额利用率。
7. 在 Monitoring (监控) 中，选择 1 小时、3 小时、12 小时、1 天、3 天或 1 周。
8. 选择图表中的任意区域，以查看服务配额利用率百分比。在此处，您可以将图形添加到控制面板或使用操作菜单选择 View in metrics (在指标中查看)，这会显示 CloudWatch 控制台中的相关指标。

使用 CloudWatch 控制台跟踪指标

您可以使用 CloudWatch 来跟踪和收集 Amazon Cognito 用户池指标。CloudWatch 控制面板将显示您所使用的每个AWS服务的指标。您可以使用 CloudWatch 来创建指标告警。可以将警报设置为向您发送通知或更改您正在监控的特定资源。要在 CloudWatch 中查看服务配额指标，请完成以下步骤。

1. 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Metrics (指标)。
3. 在 All metrics (所有指标) 中，选择一个指标和维度。
4. 选中指标旁边的复选框。指标将出现在图表中。

> Note
>
> 控制台中不会显示在过去两周内没有任何新数据点的指标。当您在控制台的 All metrics (全部指标) 选项卡的搜索框中输入指标名称或维度名称时，它们也不会显示，并且 list-metrics 命令的结果中不会返回它们。检索这些指标的最佳方法是在 AWS CLI 中使用 `get-metric-data` 或者 `get-metric-statistics` 命令。

为配额创建 CloudWatch 告警

Amazon Cognito 提供 CloudWatch 使用量指标，这些指标与 `CallCount` 和 `ThrottleCount` API 的AWS服务配额对应。有关在 CloudWatch 中跟踪使用量的更多信息，请参阅跟踪配额使用量 (p. 541)。

在 Service Quotas 控制台中，您可以创建告警，以便在您的使用量接近服务配额时提示您。要了解如何使用 Service Quotas 控制台设置 CloudWatch 告警，请参阅 Service Quotas 和 CloudWatch 告警。

使用 AWS CloudTrail 记录 Amazon Cognito API 调用

Amazon Cognito 与 AWS CloudTrail 集成，后者是一项服务，提供 Amazon Cognito 中由用户、角色或AWS服务所采取操作的记录。CloudTrail 将对 Amazon Cognito 的 API 调用子集作为事件捕获，包括来自 Amazon Cognito 控制台的调用和对 Amazon Cognito API 操作的代码调用。如果您创建跟踪，则可以选择将 CloudTrail 事件传送到 Amazon S3 存储桶，包括 Amazon Cognito 的事件。如果您不配置跟踪，则仍可在 CloudTrail 控制台中的 Event history（事件历史记录）中查看最新事件。使用 CloudTrail 收集的信息，您可以确定向 Amazon Cognito 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息（包括如何对其进行配置和激活），请参阅《AWS CloudTrail 用户指南》。

您还可以针对特定 CloudTrail 事件创建 Amazon CloudWatch 告警。例如，您可以设置 CloudWatch，以在身份池配置发生更改时触发警报。有关更多信息，请参阅针对 CloudTrail 事件创建 CloudWatch 告警：示例。

主题

- CloudTrail 中的 Amazon Cognito 信息 (p. 513)
- 了解 Amazon Cognito 登录事件 (p. 516)
- 使用 Amazon CloudWatch Logs Insights 分析 Amazon Cognito CloudTrail 事件 (p. 529)

CloudTrail 中的 Amazon Cognito 信息

在您创建帐户时，您的 AWS 帐户中已启用 CloudTrail。当 Amazon Cognito 中发生受支持的事件活动时，该活动将记录在 CloudTrail 事件中，并与其他 AWS 服务事件一同保存在 Event history (事件历史记录) 中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录AWS账户中的事件（包括 Amazon Cognito 的事件），请创建跟踪记录。CloudTrail 跟踪可将日志文件传送至 Amazon S3 存储桶。原定设置情况下，在控制台中创建跟踪时，此跟踪应用于所有区域。此跟踪记录在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Simple Storage Service（Amazon S3）桶。此外，您可以配置其他AWS服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其它 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

Amazon Cognito User Pools

Amazon Cognito 支持将用户池操作页面上列出的所有操作作为 CloudTrail 日志文件中的事件进行记录。Amazon Cognito 将用户群体事件作为管理事件记录到 CloudTrail。

Amazon Cognito 用户群体 CloudTrail 条目中的 `eventType` 字段告诉您，应用程序是向 Amazon Cognito 用户群体 API 发出了请求，还是向为 OpenID Connect、SAML 2.0 或托管 UI 提供资源的端点发出了请求。API 请求的 `AwsApiCall` 为 `eventType`，端点请求的 `AwsServiceEvent` 为 `eventType`。

Amazon Cognito 将以下对于托管 UI 的托管 UI 请求作为事件记录到 CloudTrail 中。

CloudTrail 中的托管 UI 操作

运算	描述
`Login_GET`, `CognitoAuthentication`	用户查看或向您的 登录端点 (p. 552) 提交凭证。
`OAuth2_Authorize_GET`	用户查看您的 对端点授权 (p. 557)。
`OAuth2Response_GET`, `OAuth2Response_POST`	用户向您的 `/oauth2/idpresponse` 端点提交 IdP 令牌。
`SAML2Response_POST`	用户向您的 `/saml2/idpresponse` 端点提交 IdP SAML 断言。
`Login_OIDC_SAML_POST`	用户在您的 登录端点 (p. 552) 中输入用户名并与 IdP 标识符匹配。
`Token_POST`	用户向您的 令牌端点 (p. 562) 提交授权码。
`Signup_GET`, `Signup_POST`	用户向您的 `/signup` 端点提交注册信息。
`Confirm_GET`, `Confirm_POST`	用户在托管 UI 中提交确认代码。
`ResendCode_POST`	用户在托管 UI 中提交重新发送确认代码的请求。
`ForgotPassword_GET`, `ForgotPassword_POST`	用户向您的 `/forgotPassword` 端点提交重置密码的请求。

运算	描述
ConfirmForgotPassword_GET, ConfirmForgotPassword_POST	用户向您的 /confirmForgotPassword 端点提交代码以确认其 ForgotPassword 请求。
ResetPassword_GET, ResetPassword_POST	用户在托管 UI 中提交新密码。
Mfa_GET, Mfa_POST	用户在托管 UI 中提交多重身份验证 (MFA) 代码。
MfaOption_GET, MfaOption_POST	用户在托管 UI 中选择其首选 MFA 方法。
Logout	用户在您的 /logout 端点注销。
SAML2Logout_POST	用户在您的 /saml2/logout 端点注销。
Error_GET	用户在托管 UI 中查看错误页面。
UserInfo_GET, UserInfo_POST	用户或 IdP 与您的 UserInfo 终端节点 (p. 566) 交换信息。
Confirm_With_Link_GET	用户根据 Amazon Cognito 在电子邮件中发送的链接提交确认。
Event_Feedback_GET	用户向 Amazon Cognito 提交有关高级安全功能事件的反馈。

Note

Amazon Cognito 在 CloudTrail 日志中记录特定用户请求的 UserSub，但不记录 UserName。通过调用 ListUsers API，并使用主题筛选条件，您可以找到给定 UserSub 的用户。

Amazon Cognito 身份池

数据事件

Amazon Cognito 将以下 Amazon Cognito 身份事件作为数据事件记录到 CloudTrail 中。数据事件是大容量数据面板 API 操作，CloudTrail 原定设置情况下不记录这些操作。记录数据事件将收取额外费用。

- GetCredentialsForIdentity
- GetId
- GetOpenIdToken
- GetOpenIdTokenForDeveloperIdentity
- UnlinkIdentity

要为这些 API 操作生成 CloudTrail 日志，您必须在跟踪记录中激活数据事件，并为 Cognito identity pools（Cognito 身份池）选择事件选择器。有关更多信息，请参阅 AWS CloudTrail 用户指南中的记录数据事件以便跟踪。

您还可以使用以下 CLI 命令将身份池事件选择器添加到您的跟踪记录中。

```
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 将剩余的 Amazon Cognito 身份池 API 操作记录为管理事件。原定设置情况下，CloudTrail 会记录管理事件 API 操作。

有关 Amazon Cognito 记录到 CloudTrail 的 Amazon Cognito 身份池 API 操作的列表，请参阅 Amazon Cognito 身份池 API 参考。

Amazon Cognito Sync

Amazon Cognito 将所有 Amazon Cognito 同步 API 操作记录为管理事件。有关 Amazon Cognito 记录到 CloudTrail 的 Amazon Cognito 同步 API 操作的列表，请参阅 Amazon Cognito 同步 API 参考。

了解 Amazon Cognito 登录事件

跟踪可以将事件作为日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

主题

- 托管 UI 注册的示例 CloudTrail 事件 (p. 516)
- SAML 请求的示例 CloudTrail 事件 (p. 521)
- 令牌端点请求的示例 CloudTrail 事件 (p. 522)
- CreateIdentityPool 的示例 CloudTrail 事件 (p. 525)
- GetCredentialsForIdentity 的示例 CloudTrail 事件 (p. 525)
- GetId 的示例 CloudTrail 事件 (p. 526)
- GetOpenIdToken 的示例 CloudTrail 事件 (p. 527)
- GetOpenIdTokenForDeveloperIdentity 的示例 CloudTrail 事件 (p. 527)
- UnlinkIdentity 的示例 CloudTrail 事件 (p. 528)

托管 UI 注册的示例 CloudTrail 事件

以下示例 CloudTrail 事件演示了当用户通过托管 UI 注册时 Amazon Cognito 记录的信息。

当新用户导航到应用程序的登录页面时，Amazon Cognito 会记录以下事件。

```
{
    "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"
}
```

当新用户从应用程序的登录页面选择 Sign up（注册）时，Amazon Cognito 会记录以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:21:43Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Signup_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "response_type":
            [
                "code"
```

```
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "7a63e7c2-b057-4f3d-a171-9d9113264fff",
    "eventID": "5e7b27a0-6870-4226-adb4-f86cd51ac5d8",
    "readOnly": true,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

当新用户选择用户名、输入电子邮件地址并从应用程序的登录页面选择密码时，Amazon Cognito 会记录以下事件。Amazon Cognito 不会将有关用户身份的识别信息记录到 CloudTrail。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:22:05Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Signup_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 302
        },
        "requestParameters":
        {
            "password":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "requiredAttributes[email]":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "response_type":
            [
                "code"
            ],
            "_csrf":
```

```
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ],
            "username":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "9ad58dd8-3517-4aa8-96a5-d17a01df9eb4",
    "eventID": "c75eb7a5-eb8c-43d1-8331-f4412e756e69",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

当新用户在注册后访问托管 UI 中的用户确认页面时，Amazon Cognito 会记录以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:22:06Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Confirm_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "response_type":
            [
                "code"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
```

```
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "58a5b170-3127-45bb-88cc-3e652d779e0b",
    "eventID": "7f87291a-6d50-409a-822f-e3a5ec7e60da",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

当用户在托管 UI 的用户确认页面中输入 Amazon Cognito 通过电子邮件发送给他们的代码时，Amazon Cognito 会记录以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:23:32Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Confirm_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 302
        },
        "requestParameters":
        {
            "confirm":
            [
                ""
            ],
            "deliveryMedium":
            [
                "EMAIL"
            ],
            "sub":
            [
                "704b1e47-34fe-40e9-8c41-504997494531"
            ],
            "code":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "destination":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
```

```
            ],
            "response_type":
            [
                "code"
            ],
            "_csrf":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "cognitoAsfData":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ],
            "username":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "9764300a-ed35-4f87-8a0f-b18b3fe2b11e",
    "eventID": "e24ac6e5-2f70-4c6e-ad4e-2f08a547bb36",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

SAML 请求的示例 CloudTrail 事件

当使用您的 SAML IdP 进行身份验证的用户将 SAML 断言提交给您的 `/saml2/idpresponse` 端点时，Amazon Cognito 会记录以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-06T00:50:57Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "SAML2Response_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
```

```
        {
            "status": 302
        },
        "requestParameters":
        {
            "RelayState":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "SAMLResponse":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "4f6f15d1-c370-4a57-87f0-aac4817803f7",
    "eventID": "9824b50f-d9d1-4fb8-a2c1-6aa78ca5902a",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "625647942648",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

令牌端点请求的示例 CloudTrail 事件

以下是来自对 令牌端点 (p. 562) 的请求的示例事件。

当已通过身份验证并收到授权代码的用户将代码提交到您的 `/oauth2/token` 端点时，Amazon Cognito 会记录以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T22:12:30Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "code":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "grant_type":
```

```
            [
                "authorization_code"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "f257f752-cc14-4c52-ad5b-152a46915238",
    "eventID": "0bd1586d-cd3e-4d7a-abaf-fd8bfc3912fd",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

当您的后端系统向您的 /oauth2/token 端点提交访问令牌的 client_credentials 请求时，Amazon Cognito 会记录以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T21:07:05Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "grant_type":
            [
                "client_credentials"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
```

```
    },
    "requestID": "4f871256-6825-488a-871b-c2d9f55caff2",
    "eventID": "473e5cbc-a5b3-4578-9ad6-3dfdcb8a6d34",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

当您的应用程序与您的 /oauth2/token 端点交换刷新令牌以获取新 ID 和访问令牌时，Amazon Cognito 会记录以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T22:16:40Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "refresh_token":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "grant_type":
            [
                "refresh_token"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_aaaaaaaaa"
    },
    "requestID": "2829f0c6-a3a9-4584-b046-11756dfe8a81",
    "eventID": "12bd3464-59c7-44fa-b8ff-67e1cf092018",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
```

```
    },
    "eventCategory": "Management"
}
```

CreateIdentityPool 的示例 CloudTrail 事件

以下示例是为进行 CreateIdentityPool 操作而发出的请求的日志条目。该请求由名为 Alice 的 IAM 用户发出。

```
{
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "['EXAMPLE_KEY_ID']",
        "userName": "Alice"
    },
    "eventTime": "2016-01-07T02:04:30Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "CreateIdentityPool",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "USER_AGENT",
    "requestParameters": {
        "identityPoolName": "TestPool",
        "allowUnauthenticatedIdentities": true,
        "supportedLoginProviders": {
            "graph.facebook.com": "000000000000000"
        }
    },
    "responseElements": {
        "identityPoolName": "TestPool",
        "identityPoolId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "allowUnauthenticatedIdentities": true,
        "supportedLoginProviders": {
            "graph.facebook.com": "000000000000000"
        }
    },
    "requestID": "15cc73a1-0780-460c-91e8-e12ef034e116",
    "eventID": "f1d47f93-c708-495b-bff1-cb935a6064b2",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

GetCredentialsForIdentity 的示例 CloudTrail 事件

以下示例是为进行 GetCredentialsForIdentity 操作而发出的请求的日志条目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetCredentialsForIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off command/
cognito-identity.get-credentials-for-identity",
    "requestParameters": {
```

```
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
            "sessionToken": "aAaAaAaAaAaAab1111111111EXAMPLE",
            "expiration": "Jan 19, 2023 5:55:08 PM"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "659dfc23-7c4e-4e7c-858a-1abce884d645",
    "eventID": "6ad1c766-5a41-4b28-b5ca-e223ccb00f0d",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```


GetId 的示例 CloudTrail 事件

以下示例是为进行 GetId 操作而发出的请求的日志条目。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:05Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetId",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off command/
cognito-identity.get-id",
    "requestParameters": {
        "identityPoolId": "us-east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE",
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "dc28def9-07c8-460a-a8f3-3816229e6664",
    "eventID": "c5c459d9-40ec-41fd-8f6b-57865d5a9975",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
```

```
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

GetOpenIdToken 的示例 CloudTrail 事件

以下示例是为进行 GetOpenIdToken 操作而发出的请求的日志条目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetOpenIdToken",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off command/
cognito-identity.get-open-id-token",
    "requestParameters": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "a506ba18-10d7-4fdb-9548-a8187b2e38bb",
    "eventID": "19ffc1a6-6ed8-4580-a4e1-3062c5ce6457",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

GetOpenIdTokenForDeveloperIdentity 的示例 CloudTrail 事件

以下示例是为进行 GetOpenIdTokenForDeveloperIdentity 操作而发出的请求的日志条目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA1EXAMPLE:johns-AssumedRoleSession",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/johns-AssumedRoleSession",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA1EXAMPLE",
```

```
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "attributes": {
                "creationDate": "2023-01-19T16:53:14Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetOpenIdTokenForDeveloperIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "27.0.3.154",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off command/
cognito-identity.get-open-id-token-for-developer-identity",
    "requestParameters": {
        "tokenDuration": 900,
        "identityPoolId": "us-east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE",
        "logins": {
            "JohnsDeveloperProvider": "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "b807df87-57e7-4dd6-b90c-b06f46a61c21",
    "eventID": "f26fed91-3340-4d70-91ae-cdf555547b76",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```


UnlinkIdentity 的示例 CloudTrail 事件

以下示例是为进行 UnlinkIdentity 操作而发出的请求的日志条目。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "UnlinkIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off command/
cognito-identity.unlink-identity",
    "requestParameters": {
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
```

```
        "loginsToRemove": ["cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa"]
    },
    "responseElements": null,
    "requestID": "99c2c8e2-9c29-416f-bb17-b650a5cbada9",
    "eventID": "d8e26126-202a-43c2-b458-3f225efaedc7",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

使用 Amazon CloudWatch Logs Insights 分析 Amazon Cognito CloudTrail 事件

您可以使用 Amazon CloudWatch Logs Insights 搜索和分析您的 Amazon Cognito CloudTrail 事件。在配置您的跟踪以将事件发送到 CloudWatch Logs 时，CloudTrail 只发送符合您跟踪记录设置的事件。

如需查询或研究您的 Amazon Cognito CloudTrail 事件，请在 CloudTrail 控制台中确保您在跟踪记录设置中选择 Management events (管理事件) 选项，以便您可以监控对AWS资源执行的管理操作。当您想要识别账户中的错误、异常活动或异常用户行为时，可以在跟踪记录设置中选择 Insights 事件选项。

Amazon Cognito 查询的示例

您可以在 Amazon CloudWatch 控制台中使用下列查询。

常规查询

查找 25 个最近添加的日志事件。

```
fields @timestamp, @message | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com"
```

获取 25 个最近添加的录入事件（包含异常）的列表。

```
fields @timestamp, @message | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com" and @message like /Exception/
```

异常和错误查询

查找最近添加的 25 个含错误代码 `NotAuthorizedException` 的录入事件以及 Amazon Cognito 用户池 `sub`。

```
fields @timestamp, additionalEventData.sub as user | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com" and errorCode= "NotAuthorizedException"
```

查找具有 `sourceIPAddress` 和相应 `eventName` 的记录数。

```
filter eventSource = "cognito-idp.amazonaws.com"
| stats count(*) by sourceIPAddress, eventName
```

查找触发 `NotAuthorizedException` 错误的前 25 个 IP 地址。

```
filter eventSource = "cognito-idp.amazonaws.com" and errorCode= "NotAuthorizedException"
| stats count(*) as count by sourceIPAddress, eventName
| sort count desc | limit 25
```

找到调用 ForgotPassword API 的前 25 个 IP 地址。

```
filter eventSource = "cognito-idp.amazonaws.com" and eventName = 'ForgotPassword'
| stats count(*) as count by sourceIPAddress
| sort count desc | limit 25
```

Amazon Cognito 的合规性验证

作为多个AWS合规性计划的一部分，第三方审计员将评估 Amazon Cognito 的安全性和合规性。其中包括 SOC、PCI、FedRAMP、HIPAA 及其它。

有关特定合规性计划范围内的 AWS 服务列表，请参阅合规性计划范围内的 AWS 服务。有关常规信息，请参阅 AWS 合规性计划。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅在 AWS Artifact 中下载报告。

您使用 Amazon Cognito 的合规性责任取决于您数据的敏感度、贵公司的合规性目标以及适用的法律法规。AWS提供以下资源来帮助满足合规性：

- 安全性与合规性 Quick Start 指南 – 这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署关注安全性和合规性的基准环境的步骤。
- 《设计符合 HIPAA 安全性和合规性要求的架构》白皮书 – 此白皮书介绍公司如何使用 AWS 创建符合 HIPAA 标准的应用程序。
- AWS 合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- AWS Config 开发人员指南中的使用规则评估资源 – AWS Config；评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- AWS Security Hub – 此 AWS 服务提供了 AWS 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。

Amazon Cognito 中的恢复能力

AWS全球基础设施围绕AWS区域和可用区构建。区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅 AWS 全球基础设施。

主题

- 区域数据注意事项 (p. 530)

区域数据注意事项

每个 Amazon Cognito 用户池均在一个AWS区域中创建，并且他们仅将用户配置文件数据存储在该区域中。用户池可以将用户数据发送到其他 AWS 区域，具体取决于可选功能的配置方式。

- 如果默认的 no-reply@verificationemail.com 电子邮件地址设置用于通过 Amazon Cognito 用户池路由电子邮件地址验证，则电子邮件将通过与关联用户池相同的区域路由。

- 如果其他电子邮件地址用于通过 Amazon Cognito 用户池配置 Amazon Simple Email Service (Amazon SES)，则该电子邮件地址将通过与 Amazon SES 中的电子邮件地址关联的 AWS 区域路由。
- 除非在配置电子邮件或电话验证中另有说明，否则来自 Amazon Cognito 用户池的短信通过同一区域 Amazon SNS 路由。
- 如果 Amazon Pinpoint 分析用于 Amazon Cognito 用户池，则事件数据将路由到美国东部（弗吉尼亚北部）区域。

  Note

  Amazon Pinpoint 在北美、欧洲、亚洲和大洋洲的多个AWS区域中均已推出。Amazon Pinpoint 区域包括 Amazon Pinpoint API。如果 Amazon Cognito 支持 Amazon Pinpoint 区域，Amazon Cognito 会将事件发送到同一 Amazon Pinpoint 区域中的 Amazon Pinpoint 项目。如果区域不受 Amazon Pinpoint 支持，Amazon Cognito 将仅支持在 us-east-1 中发送事件。有关 Amazon Pinpoint 区域的详细信息，请参阅 Amazon Pinpoint 端点和配额和将 Amazon Pinpoint 分析与 Amazon Cognito 用户池结合使用。

Amazon Cognito 中的基础设施安全性

作为一项托管式服务，Amazon Cognito 由 Amazon Web Services：安全流程概览白皮书中所述的 AWS 全球网络安全流程提供保护。

您可以使用AWS发布的 API 调用通过网络访问 Amazon Cognito。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service (AWS STS) 生成临时安全凭证来对请求进行签名。

Amazon Cognito 用户池中的配置和漏洞分析

AWS 负责处理基本安全任务，如来宾操作系统 (OS) 和数据库补丁、防火墙配置和灾难恢复等。这些流程已通过相应第三方审核和认证。有关更多详细信息，请参阅以下资源：

- Amazon Cognito 的合规性验证 (p. 530)
- 责任共担模式

Amazon Cognito 的 AWS 托管式策略

要向用户、组和角色添加权限，与自己编写策略相比，使用 AWS 托管策略更简单。创建仅为团队提供所需权限的 IAM 客户托管策略需要时间和专业知识。要快速入门，您可以使用我们的 AWS 托管策略。这些策略涵盖常见使用案例，可在您的 AWS 账户中使用。有关 AWS 托管策略的更多信息，请参阅 IAM 用户指南中的 AWS 托管策略。

AWS 服务负责维护和更新 AWS 托管策略。您无法更改 AWS 托管策略中的权限。服务偶尔会向 AWS 托管式策略添加额外权限以支持新功能。此类更新会影响附加策略的所有身份（用户、组和角色）。当启动新功能或新操作可用时，服务最有可能会更新 AWS 托管式策略。服务不会从 AWS 托管式策略中删除权限，因此策略更新不会破坏您的现有权限。

此外，AWS 还支持跨多种服务的工作职能的托管策略。例如，ReadOnlyAccess AWS 托管策略提供对所有 AWS 服务和资源的只读访问权限。当服务启动新功能时，AWS 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅 IAM 用户指南中的适用于工作职能的 AWS 托管策略。

您可以使用 IAM 控制台提供的大量策略授予对 Amazon Cognito 的访问权限：

- `AmazonCognitoPowerUser`– 用于对身份池和用户池所有方面进行访问和管理的权限。要查看此策略的权限，请参阅 AmazonCognitoPowerUser。
- `AmazonCognitoReadOnly` – 用于对身份池和用户池进行只读访问的权限。要查看此策略的权限，请参阅 AmazonCognitoReadOnly。
- `AmazonCognitoDeveloperAuthenticatedIdentities` – 用于将身份验证系统与 Amazon Cognito 集成的权限。要查看此策略的权限，请参阅 AmazonCognitoDeveloperAuthenticatedIdentities。

这些策略由 Amazon Cognito 团队维护，因此，即使添加了新 API，用户也将继续拥有相同级别的访问权限。

> **Note**
>
> 由于创建新的身份池时还需要创建 IAM 角色，您希望用来创建新身份池的所有用户也都必须具有管理员策略。

Amazon Cognito 更新了 AWS 托管式策略

查看有关自此服务开始跟踪这些更改起，Amazon Cognito 的 AWS 托管式策略更新的详细信息。有关此页面更改的自动提示，请订阅 Amazon Cognito 文档历史记录页面上的 RSS 源。

更改	说明	日期
`AmazonCognitoPowerUser` 和 `AmazonCognitoReadOnly` – 现有策略更新	添加了新权限，以允许高级用户查看和管理 AWS WAF Web ACL 与 Amazon Cognito 用户池的关联。 添加了新权限，以允许只读用户查看 AWS WAF Web ACL 与 Amazon Cognito 用户池的关联。	2022 年 7 月 19 日
`AmazonCognitoPowerUser` – 对现有策略的更新	添加了一项新权限，允许 Amazon Cognito 调用 Amazon Simple Notification Service `PutIdentityPolicy` 和 `ListConfigurationSets` 操作。 此更改允许 Amazon Cognito 用户池更新 Amazon SES 发送授权策略，并在您配置用户池中发送电子邮件时应用 Amazon SES 配置集。	2021 年 11 月 17 日

更改	说明	日期
`AmazonCognitoPowerUser` – 对现有策略的更新	添加了一项新权限，允许 Amazon Cognito 调用 Amazon Simple Notification Service 的 `GetSMSSandboxAccountStatus` 操作。 此更改允许 Amazon Cognito 用户池决定您是否需要退出 Amazon Simple Notification Service 沙盒，以便通过用户池向所有终端用户发送消息。	2021 年 6 月 1 日
Amazon Cognito 开启跟踪更改	Amazon Cognito 为其 AWS 托管式策略开启了跟踪更改。	2021 年 3 月 1 日

为 Amazon Cognito 资源贴标签

标签是您或 AWS 分配给 AWS 资源的元数据标签。每个标签均包含一个键 和一个值。对于您分配的标签，需要定义键和值。例如，您可以将键定义为 `stage`，将一个资源的值定义为 `test`。

标签可帮助您：

- 标识和整理您的 AWS 资源。许多 AWS 服务支持标签，因此，您可以将同一标签分配给来自不同服务的资源，以指示这些资源是相关的。这有助于您指示哪些资源是相关的。例如，您可以将相同的标签分配给您分配至 Amazon DynamoDB 表的 Amazon Cognito 用户池。
- 跟踪您的 AWS 成本。您可以在 AWS Billing and Cost Management 控制面板上激活标签。AWS 使用成本分配标签对您的成本进行分类，并向您提供每月成本分配报告。有关更多信息，请参阅《AWS Billing 用户指南》中的使用成本分配标签。
- 根据分配给资源的标签控制对资源的访问。您可以通过在 AWS Identity and Access Management（IAM）策略的条件中指定标签键和值来控制访问权限。例如，您可以允许用户更新用户群体，但前提是该用户群体具有 `owner` 标签且值为该用户的名称。有关更多信息，请参阅《IAM 用户指南》中的使用标签控制访问。

您可以使用 AWS Command Line Interface 或 Amazon Cognito API 来添加、编辑或删除用户池和身份池的标签。您还可以使用 Amazon Cognito 控制台来管理用户池的标签。

有关使用标签的提示，请参阅 AWS Answers 博客上的 AWS tagging strategies 文章。

以下各部分提供有关 Amazon Cognito 标签的更多信息。

Amazon Cognito 中支持的资源

Amazon Cognito 中的以下资源支持贴标签：

- 用户池
- 身份池

标签限制

以下限制适用于 Amazon Cognito 资源上的标签：

- 您可以分配给资源的最大标签数量 - 50
- 最大密钥长度 – 128 个 Unicode 字符
- 最大值长度 – 256 个 Unicode 字符
- 键和值的有效字符 – a-z、A-Z、0-9、空格和以下字符：_ . : / = + - 和 @
- 键和值区分大小写
- 请不要使用 `aws:` 作为键的前缀；它保留为供 AWS 使用

使用 Amazon Cognito 控制台管理标签

您可以使用 Amazon Cognito 控制台来管理分配给您的用户池的标签。

将标签添加到用户池

1. 导航到 Amazon Cognito 控制台。如果出现提示，请输入 AWS 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 User pool properties（用户池属性）选项卡并找到 Tags（标签）。
5. 选择 Add tags（添加标签）即可添加您的第一个标签。如果您之前已将标签分配给此用户池，请在 Manage tags（管理标签）中选择 Add another（添加其它）。
6. 为 Tag key（标签键）和 Tag value（标签值）指定值。
7. 对于每个要添加的其它标签，请选择 Add another（添加其它）。
8. 添加完标签后，选择 Save changes（保存更改）。

在 Manage tags（管理标签）页面上，您还可以编辑任何现有标签的键和值。要删除标签，请选择 Remove（删除）。

AWS CLI 示例

AWS CLI 提供帮助您管理分配给 Amazon Cognito 用户池和身份池的标签。

分配标签

使用以下命令可将标签分配给现有的用户池和身份池。

Example 适用于用户池的 **`tag-resource`** 命令

使用 `cognito-idp` 命令集中的 `tag-resource`将标签分配给用户池：

```
$ aws cognito-idp tag-resource \
> --resource-arn user-pool-arn \
> --tags Stage=Test
```

此命令包含以下参数：

- `resource-arn` - 您向其应用标签的用户池的 Amazon Resource Name（ARN）。要查找 ARN，请在 Amazon Cognito 控制台中选择该用户池，并查看 General settings（常规设置）选项卡上的 Pool ARN（池 ARN）值。
- `tags` – 以格式 *`key`*=*`value`* 表示的标签的键值对。

要一次分配多个标签，请以逗号分隔的列表形式指定它们：

```
$ aws cognito-idp tag-resource \
> --resource-arn user-pool-arn \
> --tags Stage=Test,CostCenter=80432,Owner=SysEng
```

Example 适用于身份池的 **`tag-resource`** 命令

使用 `cognito-identity` 命令集中的 `tag-resource` 将标签分配给身份池：

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
```

```
> --tags Stage=Test
```

此命令包含以下参数：

- `resource-arn` - 您向其应用标签的身份池的 Amazon Resource Name（ARN）。要查找 ARN，请在 Amazon Cognito 控制台中选择身份池，并选择 Edit identity pool (编辑身份池)。然后，在 Identity pool ID（身份池 ID）中，选择 Show ARN（显示 ARN）。
- `tags` – 以格式 *key*=*value* 表示的标签的键值对。

要一次分配多个标签，请以逗号分隔的列表形式指定它们：

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
> --tags Stage=Test,CostCenter=80432,Owner=SysEng
```

查看标签

使用以下命令可查看您已分配给用户池和身份池的标签。

Example 适用于用户池的 **list-tags-for-resource** 命令

使用 `cognito-idp` 命令集中的 `list-tags-for-resource` 查看分配给用户池的标签：

```
$ aws cognito-idp list-tags-for-resource --resource-arn user-pool-arn
```

Example 适用于身份池的 **list-tags-for-resource** 命令

使用 `cognito-identity` 命令集中的 `list-tags-for-resource` 查看分配给身份池的标签：

```
$ aws cognito-identity list-tags-for-resource --resource-arn identity-pool-arn
```

删除标签

使用以下命令从用户池和身份池中删除标签。

Example 适用于用户池的 **untag-resource** 命令

使用 `cognito-idp` 命令集中的 `untag-resource` 删除用户池中的标签：

```
$ aws cognito-idp untag-resource \
> --resource-arn user-pool-arn \
> --tag-keys Stage CostCenter Owner
```

对于 `--tag-keys` 参数，请指定一个或多个标签键。请勿包含标签值。用空格分隔键。

Example 适用于身份池的 **untag-resource** 命令

使用 `cognito-identity` 命令集中的 `untag-resource` 删除身份池中的标签：

```
$ aws cognito-identity untag-resource \
> --resource-arn identity-pool-arn \
> --tag-keys Stage CostCenter Owner
```

对于 `--tag-keys` 参数，请指定一个或多个标签键。请勿包含标签值。

> **Important**
>
> 删除用户或身份池后，与已删除池相关的标签在删除后的 30天内仍然可以在控制台或 API 调用中显示。

在创建资源时应用标签

使用以下命令可在创建用户池或身份池时分配标签。

Example **`create-user-pool`** 命令以及标签

当您使用 `create-user-pool` 命令创建用户时，您可以使用 `--user-pool-tags` 参数指定标签：

```
$ aws cognito-idp create-user-pool \
> --pool-name user-pool-name \
> --user-pool-tags Stage=Test,CostCenter=80432,Owner=SysEng
```

标签的键值对必须采用格式 *key*=*value*。如果要添加多个标签，请以逗号分隔的列表形式指定它们。

Example **`create-identity-pool`** 命令以及标签

当您使用 `create-identity-pool` 命令创建用户池时，您可以使用 `--identity-pool-tags` 参数指定标签：

```
$ aws cognito-identity create-identity-pool \
> --identity-pool-name identity-pool-name \
> --allow-unauthenticated-identities \
> --identity-pool-tags Stage=Test,CostCenter=80432,Owner=SysEng
```

标签的键值对必须采用格式 *key*=*value*。如果要添加多个标签，请以逗号分隔的列表形式指定它们。

使用 Amazon Cognito API 管理标签

您可以在 Amazon Cognito API 中使用以下操作来管理用户池和身份池的标签。

适用于用户池标签的 API 操作

使用以下 API 操作来分配、查看和删除用户池的标签。

- TagResource
- ListTagsForResource
- UntagResource
- CreateUserPool

适用于身份池标签的 API 操作

使用以下 API 操作来分配、查看和删除身份池的标签。

- TagResource
- ListTagsForResource

- UntagResource
- CreateIdentityPool

Amazon Cognito 中的限额

Amazon Cognito 对您可以在账户中执行的最大操作数具有原定设置限额，以前称为限制。Amazon Cognito 对于 Amazon Cognito 资源的最大数量和规模也具有限额。

每个 Amazon Cognito 配额表示在一个 AWS 区域、一个 AWS 账户 中最大的请求量。例如，您的应用程序可以对美国东部（弗吉尼亚州北部）中的所有用户群体，最高以默认配额（RPS）速率为 `UserAuthentication` 操作发出 API 请求。您的应用程序在亚太地区（东京）可以对在其自己区域中的用户群体生成相同数量的请求。AWS 一次只能批准一个区域中的配额增加请求。在美国东部（弗吉尼亚州北部）成功增加限额对您在亚太地区（东京）的最大请求速率没有任何效果。

主题

- 了解 API 请求速率配额 (p. 539)
- 管理 API 请求速率配额 (p. 540)
- Amazon Cognito 用户池 API 操作类别和请求速率配额 (p. 542)
- Amazon Cognito 身份池（联合身份）API 操作请求速率配额 (p. 547)
- 资源数量和大小的配额 (p. 547)

了解 API 请求速率配额

配额分类

Amazon Cognito 为 API 操作设置了最大请求速率。有关 Amazon Cognito 提供的 API 操作的更多信息，请参阅 Amazon Cognito API 和端点参考 (p. 551)，对于用户群体，这些操作分为几个常见使用案例类别，例如 `UserAuthentication` 或 `UserCreation`。有关按类别列出的 API 操作列表，请参阅 Amazon Cognito 用户池 API 操作类别和请求速率配额 (p. 542)。您可以在 Service Quotas 控制台中，按类别跟踪用户群体配额使用情况以及请求提升配额。

操作限额定义如下：某类别内所有操作的每秒最大请求数 (RPS)。Amazon Cognito 用户池服务将配额应用于每个类别下的所有操作。例如，类别 `UserCreation`，包括四项操作：`SignUp`、`ConfirmSignUp`、`AdminCreateUser` 和 `AdminConfirmSignUp`。该类别分配有组合配额 50 RPS。如果多个操作同时发生，此类别中的每个操作最多可以单独调用 50 RPS，也可以组合调用。

> **Note**
>
> 类别配额仅适用于用户群体。Amazon Cognito 将每个身份池配额应用于单个操作。对于按类别和按操作的请求速率配额，AWS 将来自所有用户群体或身份池的所有请求汇总到在一个 AWS 区域 的您的 AWS 账户 中。

使用特殊请求速率处理 Amazon Cognito 用户池 API 操作

操作配额针对组合请求总数在类别级别进行衡量和实施，但 `AdminRespondToAuthChallenge` 和 `RespondToAuthChallenge` 操作除外，在这两个操作中，将应用特殊处理规则。

UserAuthentication 类别包括四项操作：AdminInitiateAuth、InitiateAuth、AdminRespondToAuthChallenge 和 RespondToAuthChallenge。InitiateAuth 和 AdminInitiateAuth 操作按照类别配额进行衡量和执行。匹配的 RespondToAuthChallenge 和 AdminRespondToAuthChallenge 操作需依据单独的配额，该配额是 UserAuthentication 类别限制的三倍。此提升的配额适应开发人员应用程序中设置的多重身份验证质询。此配额足以涵盖大多数使用案例。如果超出每次身份验证调用三倍的阈值，超出率会计入 UserAuthentication 类别限额。

例如，如果 UserAuthentication 类别的配额为 80 RPS，您可以调用 RespondToAuthChallenge 或 AdminRespondToAuthChallenge 最多 240 RPS（80 RPS x 3）。如果将应用程序设置为每次身份验证调用有四轮质询，并且您每秒有 70 次登录，则 RespondToAuthChallenge 总数为 280 RPS（70 x 4），比配额高出 40 RPS。将额外的 40 RPS 添加到 70 个 InitiateAuth 调用，使 UserAuthentication 类别的总使用量为 110 RPS（40 + 70）。该值比设置为 80 RPS 的类别配额多出 30 RPS，因此该应用程序受限。

Monthly active users (月活跃用户数)

月活跃用户（MAU）是计算 Amazon Cognito 用户池账单的基准。MAU 还用于确定您的用户池是否符合增加配额资格。如果在一个日历月内有与该用户相关的身份操作，则该用户将计为 MAU。使用户变为活动状态的活动包括以下各项。

- 注册和以管理员身份创建用户
- 登录
- 用户账户确认或属性验证
- 密码重置
- 更改用户属性、组成员资格或 MFA 首选项
- 查询用户的详细属性
- 用户激活、停用或删除

管理 API 请求速率配额

确定配额要求

> **Important**
>
> 如果您增加例如 UserAuthentication、UserCreation 或 AccountRecovery 等类别的 Amazon Cognito 配额，您可能需要增加其它服务的配额。例如，如果这些服务的请求率配额不足，Amazon Cognito 使用 Amazon Simple Notification Service（Amazon SNS）或 Amazon Simple Email Service（Amazon SES）发送的邮件可能会发送失败。

要计算配额要求，请确定在特定时间段内将与您的应用程序交互的活跃用户数。例如，如果您的应用程序预计在八小时内平均有 100 万个活跃用户登录，则您必须平均每秒对 35 个用户进行身份验证。

此外，如果您假设平均用户会话为两个小时，并且令牌配置为在一个小时后过期，则每个用户必须在此会话期间刷新其令牌一次。为支持此负载，UserAuthentication 类别所需的平均配额为 70 RPS。

如果考虑八小时内用户登录频率的变化，假设峰值与平均值比率为 3:1，则所需的 UserAuthentication 配额为 200 RPS。

> **Note**
>
> 如果您为每个用户操作调用多个操作，则必须在类别级别对各个操作调用速率进行求和。

优化配额

请按照以下方法之一处理峰值调用速率。

在回退等待期之后重试尝试

> 您可以在每次 API 调用时捕获错误，然后在回退期之后重试尝试。您可以根据业务需求和负载调整回退算法。Amazon SDK 具有内置重试逻辑。有关更多信息，请参阅用于在 AWS 上进行构建的工具。

使用外部数据库处理频繁更新的属性

> 如果您的应用程序需要对用户池进行多次调用以读取或写入自定义属性，可使用外部存储。您可以使用首选数据库存储自定义属性，也可以在登录期间使用缓存层来加载用户配置文件。您可以在需要时从缓存中引用此配置文件，而无需从用户池重新加载用户配置文件。

在客户端验证 JWT 令牌

> 应用程序必须在信任 JWT 令牌之前验证这些令牌。您可以在客户端验证令牌的签名和有效性，而无需向用户池发送 API 调用。验证令牌后，您可以信任令牌中的陈述并使用陈述，而不是执行更多 `getUser` API 调用。有关更多信息，请参阅 验证 JSON Web 令牌。

使用等候室限制 Web 应用程序的流量

> 如果您预计在有时限的活动（例如参加考试或参加实时活动）期间会有大量用户登录，则可以使用自我限制机制优化请求流量。例如，您可以设置一个等候室，用户可以在其中等待直到会话可用，从而允许您在有可用容量时处理请求。请参阅 AWS 虚拟等候室解决方案以获取等候室的参考架构。

跟踪配额使用量

Amazon Cognito 为每个 API 操作类别生成账户级别的 Amazon CloudWatch 中的 `CallCount` 和 `ThrottleCount` 指标。您可以使用 `CallCount` 跟踪客户发出的与类别相关的调用总数。您可以使用 `ThrottleCount` 跟踪与类别相关的节流调用总数。您可以使用 `CallCount` 和 `ThrottleCount` 指标以及 `Sum` 统计数据计算类别中的调用总数。有关更多信息，请参阅 CloudWatch 使用情况指标。

监控服务配额时，utilization（利用率）是使用中的服务配额的百分比。例如，如果配额值为 200 个资源，正在使用 150 个资源，则利用率为 75%。Usage（使用量）是指服务配额中正在使用的资源或操作的数量。

通过 CloudWatch 指标跟踪使用量

您可以使用 CloudWatch 跟踪和收集 Amazon Cognito 用户池利用率指标。CloudWatch 控制面板将显示您所使用的每个 AWS 服务 的指标。借助 CloudWatch，您可以创建指标告警以通知您或更改您正在监控的特定资源。有关 CloudWatch 使用量指标的更多信息，请参阅跟踪您的 CloudWatch 使用量指标 (p. 507)。

通过 Service Quotas 指标跟踪利用率

Amazon Cognito 用户池已与 Service Quotas 集成，后者是一个基于浏览器的界面，可用于查看和管理您的服务配额使用量。在 Service Quotas 控制台中，您可以查找特定配额的值、查看监控信息、请求增加配额或设置 CloudWatch 告警。在您的账户处于活动状态一段时间后，您可以查看资源利用率图。

有关在 Service Quotas 控制台中查看配额的更多信息，请参阅查看 Service Quotas。

请求增加配额

Amazon Cognito 对于您可以在账户中执行的用户群体操作的最大数量具有限额。您可以在 Amazon Cognito 中请求增加可调 API 请求速率配额。要请求增加配额，请使用 Service Quotas 控制台、服务限制提高表单或 `RequestServiceQuotaIncrease` 或 `ListAWSDefaultServiceQuotas` API 操作。

- 要使用 Service Quotas 控制台请求增加配额，请参阅《Service Quotas 用户指南》中的请求增加 API 配额。
- 如果配额在 Service Quotas 中尚不可用，请使用服务限制提高表单。

> **Important**
>
> 只能增加可调配额。根据当前的利用率、现有或预期的 MAU 以及优化最佳实践的实施情况来评估配额增加请求。当您提交提高限额请求时，请尽可能提供有关当前使用情况、预计使用量和优化方法的更多信息。如果限额提升会提高您的请求速率，您当前的平均请求速率应已接近当前限额的最高速率。请参阅 Amazon Cognito 用户池 API 操作类别和请求速率配额 (p. 542)，了解有关可调配额的更多信息。
> 考虑一个示例场景，您的应用程序正在增长到 500 万个 MAU。您当前在 UserAuthentication 类别操作中的平均利用率很高：为 60%，或每秒 72 个请求，而当前的配额为 120 个。对于超过前 200 万个 MAU 的 300 万个 MAU 中的每一个 MAU，您可以为每秒 40 个请求申请提升 UserAuthentication 限额，从而使您的限额为每秒最多 240 个请求。

Amazon Cognito 用户池 API 操作类别和请求速率配额

由于 Amazon Cognito 的重叠 API 操作类具有不同的授权模型，因此每个操作都属于一个类别。每个类别对所有成员 API 操作都有自己的共同使用配额，跨您账户中一个 AWS 区域 的所有用户群体。您只能请求提升可调整类别配额。有关更多信息，请参阅请求增加配额 (p. 541)。配额调整适用于您的账户在单个区域中的用户群体。对于每个用户群体，Amazon Cognito 将某些类别[1] 中的操作限制为每秒 5 次请求（RPS，Requests Per Second）。Default quota (RPS) [默认配额（RPS）] 还适用于账户中的所有用户群体。

> **Note**
>
> 每个类别的配额按每月活跃用户数（MAU，Monthly Active User）计。不超过 200 万个 MAU 的 AWS 账户 可以在默认配额内操作。如果您的 MAU 少于 100 万，则不应请求提升配额。

类别操作配额适用于一个 AWS 区域 的所有用户群体中的所有用户。Amazon Cognito 还为您的应用程序可以针对一个用户生成的请求数量保留了配额。您必须限制每个用户的 API 请求数，如下表所示。

Amazon Cognito 用户群体中每个用户的请求速率配额

运算	每个用户每秒操作数
读取用户配置文件 示例：`AdminInitiateAuth`、`GetUser`、`GetDevice`。	10
写入用户配置文件 示例：`UpdateUserAttributes`、`SetUserSettings`、`AdminLinkProviderForUser`。	10

您必须限制每个类别的 API 请求数，如下表所示。

Amazon Cognito 用户群体中每个类别的请求速率配额

类别	描述	默认配额（RPS）[2]	可调整	提高配额的资格[3]
UserAuthentication • InitiateAuth • RespondToAuthChallenge • AdminInitiateAuth • AdminRespondToAuthChallenge	对用户进行身份验证（登录）的操作。 这些操作受使用特殊请求速率处理 Amazon Cognito 用户池 API 操作 (p. 539)的制约。	120	是	每增加 100 万个 MAU 就可获得 40 RPS
UserCreation • SignUp • ConfirmSignUp • AdminCreateUser • AdminConfirmSignUp	用于创建或确认 Amazon Cognito 本地用户的操作。这是由您的 Amazon Cognito 用户池直接创建和验证的用户。	50	是	每增加 100 万个 MAU 就可获得 10 RPS
UserFederation Amazon Cognito 托管式操作，用于通过第三方 IdP 将用户联合到 Cognito。	通过第三方身份提供商将用户联合（身份验证）到您的 Amazon Cognito 用户池的操作。	25	是	每增加 100 万个 MAU 就可获得 10 RPS
UserAccountRecovery • ChangePassword • ConfirmForgotPassword • ForgotPassword • AdminResetUserPassword • AdminSetUserPassword	恢复用户账户或者更改或更新用户密码的操作。	30	否	不适用
UserRead • AdminGetUser • GetUser	从用户池中检索用户的操作。	120	是	每增加 100 万个 MAU 就可获得 40 RPS
UserUpdate • AdminAddUserToGroup • AdminDeleteUserAttributes • AdminUpdateUserAttributes • AdminDeleteUser • AdminDisableUser • AdminEnableUser • AdminLinkProviderForUser • AdminDisableProviderForUser • VerifyUserAttribute • DeleteUser • DeleteUserAttributes	用于管理用户和用户属性的操作。	25	否	不适用

类别	描述	默认配额（RPS）[2]	可调整	提高配额的资格[3]
• UpdateUserAttributes • AdminUserGlobalSignOut • GlobalSignOut • AdminRemoveUserFromGroup				
`UserToken` • RevokeToken	令牌管理的操作	120	是	每增加 100 万个 MAU 就可获得 40 RPS
`UserResourceRead` • AdminGetDevice • AdminListGroupsForUser • AdminListDevices • GetDevice • ListDevices • GetUserAttributeVerificationCode • ResendConfirmationCode • AdminListUserAuthEvents	从 Amazon Cognito 检索用户资源信息（如记忆设备或组成员资格）的操作。	50	是	每增加 100 万个 MAU 就可获得 20 RPS
`UserResourceUpdate` • AdminForgetDevice • AdminUpdateAuthEventFeedback • AdminSetUserMFAPreference • AdminSetUserSettings • AdminUpdateDeviceStatus • UpdateDeviceStatus • UpdateAuthEventFeedback • ConfirmDevice • SetUserMFAPreference • SetUserSettings • VerifySoftwareToken • AssociateSoftwareToken • ForgetDevice	用于更新用户的资源信息（如记忆设备或组成员资格）的操作。	25	否	不适用
`UserList` • ListUsers • ListUsersInGroup	返回用户列表的操作。	30	否	不适用
`UserPoolRead` • DescribeUserPool • ListUserPools	读取用户池的操作。	15	否	不适用

类别	描述	默认配额（RPS）[2]	可调整	提高配额的资格[3]
`UserPoolUpdate` • CreateUserPool • UpdateUserPool • DeleteUserPool	创建、更新或删除用户池的操作。	15	否	不适用
`UserPoolResourceRead` • DescribeIdentityProvider • DescribeResourceServer • DescribeUserImportJob • DescribeUserPoolDomain • GetCSVHeader • GetGroup • GetSigningCertificate • GetIdentityProviderByIdentifier • GetUserPoolMfaConfig • ListGroups • ListIdentityProviders • ListResourceServers • ListTagsForResource • ListUserImportJobs • DescribeRiskConfiguration • GetUICustomization	从用户池中检索有关资源信息（如组或资源服务器）的操作。[1]	20	否	不适用

类别	描述	默认配额（RPS）[2]	可调整	提高配额的资格[3]
UserPoolResourceUpdate • AddCustomAttributes • CreateGroup • CreateIdentityProvider • CreateResourceServer • CreateUserImportJob • CreateUserPoolDomain • DeleteGroup • DeleteIdentityProvider • DeleteResourceServer • DeleteUserPoolDomain • SetUserPoolMfaConfig • StartUserImportJob • StopUserImportJob • UpdateGroup • UpdateIdentityProvider • UpdateResourceServer • UpdateUserPoolDomain • SetRiskConfiguration • SetUICustomization • TagResource • UntagResource	修改用户池中的资源（如组或资源服务器）的操作。[1]	15	否	不适用
UserPoolClientRead • DescribeUserPoolClient[1] • ListUserPoolClients	检索用户池客户端的相关信息的操作。	15	否	不适用
UserPoolClientUpdate • CreateUserPoolClient[1] • DeleteUserPoolClient • UpdateUserPoolClient	创建、更新和删除用户池客户端的操作。	15	否	不适用
ClientAuthentication client_credentials 授予 TOKEN 端点类型请求。	生成用于授权计算机到计算机请求的凭证的操作	150	否	不适用

[1]此类别中的任何单独操作都有一项约束，阻止单个用户池以高于 5 RPS 的速率调用操作。

[2]所有少于 200 万个 MAU 的用户池的请求速率。

[3]当您拥有 100 万至 1000 万个 MAU 时，将考虑提高请求速率。根据当前的利用率、现有或预期的 MAU 以及优化最佳实践的实施情况来评估配额增加请求。Amazon Cognito 保留拒绝限额增加请求的权利，以防止滥用并保护所有客户的服务可用性。如果您有 1000 万个以上的 MAU，我们建议您与 AWS 解决方案构架师合作以针对 Amazon Cognito 配额优化的方式设计解决方案。请联系您的 AWS 账户团队寻求帮助。

Amazon Cognito 身份池（联合身份）API 操作请求速率配额

运算	描述	默认配额（RPS）	可调整	提高配额的资格
GetId	检索身份池的身份 ID。	25	是	请联系您的账户团队。
GetOpenIdToken	在经典工作流中从身份池中检索 OpenID 令牌。	200	是	请联系您的账户团队。
GetCredentialsForIdentity	检索增强型工作流中的身份池的 AWS 凭证。	200	是	请联系您的账户团队。
GetOpenIdTokenForDeveloperIdentity	从开发人员工作流中的身份池中检索 OpenID 令牌。	50	是	请联系您的账户团队。
ListIdentities	检索身份池中的身份 ID 列表。	5	是	请联系您的账户团队。
DeleteIdentities	从身份池中删除一个或多个注册的身份。	10	是	请联系您的账户团队。
TagResource	将标签应用于身份池。	5	是	请联系您的账户团队。
UntagResource	从身份池中移除标签。	5	是	请联系您的账户团队。
ListTagsForResource	显示应用于身份池的标签列表。	10	是	请联系您的账户团队。

资源数量和大小的配额

资源配额是 Amazon Cognito 中资源、输入字段、持续时间和其他杂项功能的最大数量或大小。

您可以在 Service Quotas 控制台中或者通过提高服务限制表调整某些资源配额。要通过 Service Quotas 控制台请求增加配额，请参阅《Service Quotas 用户指南》中的请求增加配额。如果配额在 Service Quotas 中尚不可用，请使用服务限制提高表单。

> **Note**
>
> 在 AWS 账户 级别的资源配额，例如每个账户的用户群体数，适用于各 AWS 区域 中的 Amazon Cognito 资源。例如，您在美国东部（弗吉尼亚州北部）中有 1000 个用户群体，在欧洲地区（斯德哥尔摩）另外有 1000 个。

下表显示了默认资源配额以及它们是否可调整。

Amazon Cognito 用户池资源配额

资源	配额	可调整	最大配额
每个用户池的应用程序端	1000	是	10000
每个账户的用户池	1000	是	10000
每个用户池的身份提供商	300	是	1000
每个用户池的资源服务器	25	是	300
每个用户池的用户数	40000000	是	请联系您的账户团队。
每个用户池的自定义属性	50	否	不适用
每个属性的字符数	2,048 字节	否	不适用
自定义属性名称的字符数	20	否	不适用
密码策略中必需的最少密码字符	6–99	否	不适用
按 AWS 账户[1] 每天发送的电子邮件	50	否	不适用
电子邮件主题中的字符数	140	否	不适用
电子邮件消息中的字符数	20000	否	不适用
SMS 验证消息中的字符数	140	否	不适用
密码中的字符数	256	否	不适用
身份提供商名称的字符数	40	否	不适用
每个身份提供商的标识符数	50	否	不适用
链接到用户的身份数	5	否	不适用
每个应用程序客户端的回调 URL 数	100	否	不适用
每个应用程序客户端的注销 URL 数	100	否	不适用
每个资源服务器的范围	100	否	不适用
每个应用程序客户端的范围	50	否	不适用
每个账户的自定义域	4	否	不适用

资源	配额	可调整	最大配额
每个用户都可以属于的组	100	否	不适用
每个用户池的组数	10000	否	不适用

[1] 仅当您使用 Amazon Cognito 用户群体的原定设置电子邮件功能时，此限额才适用。要提高电子邮件送达量，请配置您的用户池以使用 Amazon SES 电子邮件配置。有关更多信息，请参阅Amazon Cognito 用户池的电子邮件设置 (p. 194)。

Amazon Cognito 用户池会话验证参数

令牌	配额
ID 令牌	5 分钟 – 1 天
刷新令牌	1 小时 – 3650 天
访问令牌	5 分钟 – 1 天
托管 UI 会话 Cookie	1 小时
身份验证会话令牌	3 分钟 – 15 分钟

Amazon Cognito 用户群体代码安全资源限额（不可调整）

资源	配额
注册确认码有效期	24 小时
用户属性验证码有效期	24 小时
多重身份验证 (MFA) 代码有效期	3–15 分钟
忘记密码代码有效期	1 小时
每用户最大 `ConfirmForgotPassword` 调用数	每小时 5 至 20 次尝试，具体取决于风险评分
每用户最大 `ResendConfirmationCode` 调用数	每小时 5 次尝试次数
每用户最大 `ConfirmSignUp` 调用数	每小时 15 次尝试次数
每用户最大 `ChangePassword` 调用数	每小时 5 次尝试次数
每用户最大 `GetUserAttributeVerificationCode` 调用数	每小时 5 次尝试次数
每用户最大 `VerifyUserAttribute` 调用数	每小时 15 次尝试次数

Amazon Cognito 用户群体用户导入任务资源限额（不可调整）

资源	配额
每个用户池的用户导入任务	1000

资源	配额
每个用户导入 CSV 行的最大字符数	16000
最大 CSV 文件大小	100MB
每个 CSV 文件的最大用户数	500,000

Amazon Cognito 身份池（联合身份）资源配额

资源	配额	可调整	最大配额
每个账户的身份池	1000	否	不适用
每个身份池的 Amazon Cognito 用户池提供商数	50	是	1000
身份池名称的字符长度	128 字节	否	不适用
登录提供商名称的字符长度	2,048 字节	否	不适用
每个身份池的身份	无限	否	不适用
可以为其指定角色映射的身份提供商数	10	否	不适用
单个列表或查找调用的结果数	60	否	不适用
基于角色的访问控制（RBAC）规则	25	否	不适用

Amazon Cognito Sync 资源配额

资源	配额	可调整	最大配额
每个身份的数据集	20	是	请联系您的账户团队。
每个数据集的记录数	1024	是	请联系您的账户团队。
单个数据集的大小	1MB	是	请联系您的账户团队。
数据集名称的字符数	128 字节	否	不适用
请求成功后批量发布的等待时间	24 小时	否	不适用

Amazon Cognito API 和端点参考

以下参考文献描述了 Amazon Cognito 各项功能的服务端点。Amazon Cognito 用户群体有以下选项：具有用户群体域的用户群体端点和用户群体 API。有关用于 Amazon Cognito 用户群体 API 的 API 操作类的细分，请参阅使用 Amazon Cognito 用户群体 API 和用户群体端点 (p. 47)。

有关按 AWS 区域列出的用户群体 API 的服务端点的列表，请参阅《AWS 一般参考》中的服务端点。

主题

- 用户群体联合身份验证端点和托管 UI 参考 (p. 551)
- Amazon Cognito 用户池 API 参考 (p. 569)
- Amazon Cognito 身份池（联合身份）API 参考 (p. 570)
- Amazon Cognito Sync API 参考 (p. 570)

用户群体联合身份验证端点和托管 UI 参考

当您向用户群体分配域时，Amazon Cognito 会激活此处列出的公开网页。您的域用作所有应用程序客户端的中央接入点。它们包括托管 UI，您的用户可以在其中注册、登录（登录端点 (p. 552)）和注销（注销端点 (p. 554)）。有关这些资源的标签的更多信息，请参阅 使用 Amazon Cognito 托管 UI 进行注册和登录 (p. 54)。

这些页面还包括允许您的用户群体与第三方 SAML、OpenID Connect (OIDC) 和 OAuth 2.0 身份提供者 (IdP) 进行通信的公有 Web 资源。要使用联合身份提供者登录用户，您的用户必须向交互式托管 UI 登录端点 (p. 552)或 OIDC 对端点授权 (p. 557)发起请求。授权端点将您的用户重定向到托管 UI 或 IdP 登录页面。

您的应用程序还可以使用 Amazon Cognito 用户群体 API 登录本地用户。本地用户仅存在于您的用户群体目录中，无需通过外部 IdP 进行联合身份验证。

除了托管 UI 和联合身份验证端点之外，Amazon Cognito 还集成了适用于 Android、iOS、JavaScript 等的 SDK。这些 SDK 提供了使用 Amazon Cognito API 服务端点执行用户群体 API 操作的工具。有关服务端点的更多信息，请参阅 Amazon Cognito 身份端点和限额。

> Warning
>
> 请不要固定 Amazon Cognito 域的终端实体或中间传输层安全性协议（TLS）凭证。AWS 管理所有用户群体端点和前缀域的所有凭证。支持 Amazon Cognito 证书的信任链中的证书颁发机构 (CA) 会动态轮换和续订。当您将应用程序固定到中间证书或叶证书时，如果 AWS 轮换证书，应用程序可能会失败而不发出通知。
> 而应将应用程序固定到所有可用的 Amazon 根证书。有关更多信息，请参阅《AWS Certificate Manager 用户指南》的证书固定中的最佳做法和建议。

主题

- 托管 UI 端点参考 (p. 552)
- OAuth 2.0、OpenID Connect 和 SAML 2.0 联合身份验证端点参考 (p. 556)
- 托管 UI 和联合身份验证错误响应 (p. 568)

托管 UI 端点参考

当您向用户群体添加域时，Amazon Cognito 会激活本节中的托管 UI 端点。这些端点是您的用户可以在其中完成用户群体的核心身份验证操作的网页。它们包括用于密码管理、多重身份验证（MFA）和属性验证的页面。有关托管 UI 中的用户体验的更多信息，请参阅使用托管 UI 注册和登录 (p. 71)。

本指南中的主题详细描述了常用的托管 UI 端点。您向用户群体分配域时，Amazon Cognito 创建以下端点。

托管 UI 端点

终端节点 URL	描述	如何访问此端点
https://*******/login	登录用户群体本地用户和联合用户。	直接链接。请参阅登录端点 (p. 552)。
https://*******/logout	注销用户群体用户。	直接链接。请参阅注销端点 (p. 554)。
https://*******/confirmUser	确认已选择电子邮件链接来验证其用户账户的用户。	电子邮件中用户选择的链接。
https://*******/signup	注册一个新用户。当您的用户选择 Sign up（注册）时，`/login` 页面会将他们定向到 `/signup`。	与 `/login` 具有相同参数的直接链接。
https://*******/confirm	在用户群体向已注册的用户发送确认码后，提示您的用户输入验证码。	仅从 `/signup` 重定向。
https://*******/forgotPassword	提示您的用户输入其用户名并发送密码重置代码。在您的用户选择 Forget your password?（忘记密码？）时，`/login` 页面会将他们定向到 `/forgotPassword`。	1. 从 `/login` 中的忘记密码链接。 2. 与 `/login` 具有相同参数的直接链接。
https://*******/confirmforgotPassword	提示您的用户输入其密码重置代码和新的密码。在您的用户选择 Reset your password（重置密码）时，`/forgotPassword` 页面会将他们定向到 `/confirmforgotPassword`。	仅从 `/forgotPassword` 重定向。
https://*******/resendcode	向已在用户群体中注册的用户发送新的确认码。	仅从 `/confirm` 中的发送新代码链接重定向。

主题

- 登录端点 (p. 552)
- 注销端点 (p. 554)

登录端点

`/login` 终端节点让用户登录。它加载登录页面，并向用户显示为客户端配置的身份验证选项。

GET /login

`/login` 端点仅支持对用户的初始请求执行 `HTTPS GET`。用户池客户端通过系统浏览器发出此请求。JavaScript 的系统浏览器包括 Chrome 或 Firefox。Android 浏览器包括自定义 Chrome 选项卡。iOS 浏览器包括 Safari View 控件。

请求参数

client_id

您的应用程序的应用程序客户端 ID。要获取应用程序客户端 ID，请在用户池中注册该应用程序。有关更多信息，请参阅 配置用户池应用程序客户端 (p. 258)。

必需。

redirect_uri

身份验证成功后用户重定向到的 URI。它应在指定的 `response_type` 的 `client_id` 上配置。

必需。

response_type

OAuth 响应类型，对于代码授予流，它可能是 `code`，对于隐式流，它可能是 `token`。

必需。

state

当应用程序向请求添加 state 参数时，如果 `/oauth2/login` 端点重新导向您的用户，则 Amazon Cognito 将此参数的值返回给您的应用程序。

将此值添加到您的请求中以防止 CSRF 攻击。

不能将 `state` 参数的值设置为 URL 编码的 JSON 字符串。要在 `state` 参数中传递与此格式匹配的字符串，请将该字符串编码为 Base64，然后在应用程序中对其进行解码。

可选但不推荐。

scope

可以是任何系统预留范围或与客户端关联的自定义范围的组合。范围必须以空格分隔。系统预留范围为 `openid`、`email`、`phone`、`profile` 和 `aws.cognito.signin.user.admin`。必须为应用程序客户端激活您请求的任何范围，否则 Amazon Cognito 将忽略它。

如果客户端不请求任何范围，则身份验证服务器使用与客户端关联的所有范围。

只有在请求 `openid` 范围时，才返回 ID 令牌。如果请求了 `aws.cognito.signin.user.admin` 范围，则访问令牌只能用于 Amazon Cognito 用户池。只有在同时请求了 `phone` 范围时，才能请求 `email`、`profile` 和 `openid` 范围。这些范围控制进入 ID 令牌中的声明。

可选。

code_challenge_method

用于生成质询的方法。PKCE RFC 定义两个方法：S256 和 plain；但是，Amazon Cognito 身份验证服务器仅支持 S256。

可选。

code_challenge

从 `code_verifier` 生成的质询。

仅在指定 code_challenge_method 时必需。

示例请求：提示用户登录

此示例显示登录屏幕。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/login?
                response_type=code&
                client_id=ad398u21ijw3s9w3939&
                redirect_uri=https://YOUR_APP/redirect_uri&
                state=STATE&
                scope=openid+profile+aws.cognito.signin.user.admin
```

示例响应

身份验证服务器重定向到您的应用程序并提供授权代码和状态。服务器必须在查询字符串参数中返回代码和状态，而不是在片段中。

```
HTTP/1.1 302 Found
                    Location: https://YOUR_APP/redirect_uri?
code=AUTHORIZATION_CODE&state=STATE
```

用户发起的登录请求

在用户加载 /login 端点后，他们可以输入用户名和密码并选择登录。这样做时，将生成一个与 GET 请求具有相同标头请求参数的 HTTPS POST 请求，以及包含用户名、密码和设备指纹的请求正文。

注销端点

/logout 端点是重定向端点。该端点将注销用户并重定向到您的应用程序客户端的授权退出 URL，或重定向到 /login 端点。/logout 端点的 GET 请求中的可用参数是针对 Amazon Cognito 托管 UI 使用案例量身定制的。

要将用户重定向到托管 UI 以再次登录，请在请求中添加 redirect_uri 参数。带 redirect_uri 参数的 logout 请求还必须包含对登录端点 (p. 552)的后续请求的参数，例如 client_id、response_type 和 scope。

要将用户重定向到您选择的页面，请将允许的注销 URL 添加到您的应用程序客户端。在用户的对 logout 端点的请求中，添加 logout_uri 和 client_id 参数。如果 logout_uri 的值是应用程序客户端的允许的注销 URL 之一，则 Amazon Cognito 会将用户重定向到该 URL。

GET /logout

/logout 终端节点只支持 HTTPS GET。用户池客户端通常通过浏览器发出此请求。浏览器在 Android 中通常是自定义 Chrome 标签页，在 iOS 中是 Safari 视图控件。

请求参数

client_id

您的应用程序的应用程序客户端 ID。要获取应用程序客户端 ID，您必须在用户池中注册该应用程序。有关更多信息，请参阅配置用户池应用程序客户端 (p. 258)。

必需。

logout_uri

使用 logout_uri 参数将用户重新导向到自定义注销页面。将其值设置为应用程序客户端 sign-out URL（注销 URL），您要在用户退出后将其重新导向到此 URL。仅将 logout_uri 与 client_id 参数一起使用。有关更多信息，请参阅配置用户池应用程序客户端 (p. 58)。

您也可以使用 logout_uri 参数将用户重定向到另一个应用程序客户端的登录页面。将其他应用程序客户端的登录页面设置为您的应用程序客户端中的 Allowed callback URL（允许的回调 URL）。在对 `/logout` 终端节点的请求中，将 logout_uri 参数的值设置为 URL 编码的登录页面。

Amazon Cognito 要求在您对 `/logout` 终端节点的请求中使用 logout_uri 或 redirect_uri 参数。logout_uri 参数会将您的用户重定向到另一个网站。

redirect_uri

使用 redirect_uri 将用户重新导向到登录页以进行身份验证。将其值设置为应用程序客户端 Allowed callback URL（允许的回调 URL），您要在用户再次登录后将其重新导向到此 URL。添加您要传递给 `/login` 端点的 client_id、scope、state 和 response_type 参数。

Amazon Cognito 要求在您对 `/logout` 终端节点的请求中使用 logout_uri 或 redirect_uri 参数。当您要将用户重定向到 `/login` 终端节点以重新验证身份并将令牌传递给您的应用程序时，请添加 redirect_uri 参数。

response_type

您希望在用户登录后从 Amazon Cognito 接收的 OAuth 2.0 响应。`code` 和 `token` 是 response_type 参数的有效值。

在您使用 redirect_uri 参数时是必需的。

state

当应用程序向请求添加 state 参数时，如果 `/oauth2/logout` 端点重新导向您的用户，则 Amazon Cognito 将此参数的值返回给您的应用程序。

将此值添加到您的请求中以防止 CSRF 攻击。

不能将 `state` 参数的值设置为 URL 编码的 JSON 字符串。要在 `state` 参数中传递与此格式匹配的字符串，请将该字符串编码为 Base64，然后在应用程序中对其进行解码。

如果您使用 redirect_uri 参数，强烈推荐使用此参数。

scope

在您使用 redirect_uri 参数将其注销后，您希望从 Amazon Cognito 请求的 OAuth 2.0 范围。Amazon Cognito 使用您对 `/logout` 终端节点的请求中的 scope 参数将您的用户重定向到 `/login` 终端节点。

在您使用 redirect_uri 参数时是可选的。如果不包括 scope 参数，Amazon Cognito 会使用 scope 参数将您的用户重定向到 `/login` 终端节点。当 Amazon Cognito 重定向用户并自动填充 `scope` 时，该参数包括应用程序客户端的所有授权范围。

示例请求

示例 1：注销并将用户重新导向到客户端

此示例将清除现有会话并将用户重定向到客户端。像这样带有 `logout_uri` 参数的示例请求，还需要 `client_id` 参数。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?
   client_id=ad398u21ijw3s9w3939&
   logout_uri=https://myclient/logout
```

示例 2：注销并提示用户以其他用户身份登录

此示例清除现有会话并显示登录屏幕。该示例使用您在对 对端点授权 (p. 557)发出请求时使用的相同参数。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?
   response_type=code&
   client_id=ad398u21ijw3s9w3939&
   redirect_uri=https://YOUR_APP/redirect_uri&
   state=STATE&
   scope=openid+profile+aws.cognito.signin.user.admin
```

OAuth 2.0、OpenID Connect 和 SAML 2.0 联合身份验证端点参考

当您向用户群体添加域时，Amazon Cognito 会激活本节中的端点。联合身份验证端点不是用户交互式的。它们为您的应用程序执行服务角色，以与第三方 OAuth 2.0、OIDC 和 SAML 2.0 身份提供者（IdP）进行通信。

本指南中的主题描述几个常用的 OAuth 2.0 和 OIDC 端点。您向用户群体分配域时，Amazon Cognito 创建以下端点。

用户群体联合身份验证端点

终端节点 URL	描述	如何访问此端点
https://*#######*/oauth2/authorize	将用户重定向到托管 UI 或使用其 IdP 登录。	直接链接。请参阅对端点授权 (p. 557)。
https://*#######*/oauth2/token	根据授权码或客户端凭证请求返回令牌。	直接链接。请参阅令牌端点 (p. 562)。
https://*#######*/oauth2/userInfo	根据访问令牌中的 OAuth 2.0 范围和用户身份返回用户属性。	直接链接。请参阅UserInfo 终端节点 (p. 566)。
https://*#######*/oauth2/revoke	撤消刷新令牌和关联的访问令牌。	直接链接。请参阅撤消端点 (p. 567)。
https://cognito-idp.*Region*.amazonaws.com/*your user pool ID*/.well-known/openid-configuration	用户群体的 OIDC 架构的目录。	直接链接。
https://cognito-idp.*Region*.amazonaws.com/*your user pool ID*/.well-known/jwks.json	您可以用来验证 Amazon Cognito 令牌的公有密钥。	直接链接。
https://*#######*/oauth2/idpresponse	社交身份提供者必须使用授权码将用户重新导向到此端点。Amazon Cognito 在验证您的联合用户时将代码兑换为令牌。	已从 OIDC IdP 登录重定向为 IdP 客户端回调 URL。
https://*Your user pool domain*/saml2/idpresponse	要对 SAML 2.0 联合用户进行身份验证，您的身份提供者必须使用 SAML 响应将用户重新导向到此端点。	已从 SAML 2.0 IdP 重定向为单点登录 URL。

有关 OpenID Connect 和 OAuth 标准的更多信息，请参阅 OpenID Connect 1.0 和 OAuth 2.0。

主题

- 对端点授权 (p. 557)
- 令牌端点 (p. 562)
- UserInfo 终端节点 (p. 566)
- 撤消端点 (p. 567)

对端点授权

`/oauth2/authorize` 端点是支持两个重定向目标的重定向端点。如果您在 URL 中包括 `identity_provider` 或 `idp_identifier` 参数，则它会静默地将用户重定向到该身份提供者 (IdP) 的登录页面。否则，它会使用您在请求中包括的相同 URL 参数重定向到 登录端点 (p. 552)。

要使用 authorize 端点，请在 `/oauth2/authorize` 上使用为您的用户群体提供有关以下用户群体详细信息的信息的参数调用用户的浏览器。

- 您希望登录到的应用程序客户端。
- 您希望最终到达的回调 URL。
- 您希望在用户的访问令牌中请求的 OAuth 2.0 范围。
- （可选）您希望用于登录的第三方 IdP。

您还可以提供 Amazon Cognito 用来验证传入声明的 `state` 和 `nonce` 参数。

GET /oauth2/authorize

`/oauth2/authorize` 终端节点只支持 `HTTPS GET`。您的应用程序通常在用户的浏览器中发起此请求。您只能通过 HTTPS 向 `/oauth2/authorize` 端点发出请求。

您可以在 OpenID Connect (OIDC) 标准的 Authorization Endpoint（授权端点）中详细了解授权端点的定义。

请求参数

response_type

响应类型。必须为 `code` 或 `token`。

`response_type` 为 `code` 的成功请求返回授权代码授予。授权代码授予是 Amazon Cognito 附加到重定向 URL 的 `code` 参数。您的应用程序可以将包含 令牌端点 (p. 562)的代码交换为访问权限、ID 和刷新令牌。作为安全最佳实践，以及要为您的用户接收刷新令牌，请在您的应用程序中使用授权代码授予。

`response_type` 为 `token` 的成功请求返回隐式授予。隐式授予是 Amazon Cognito 附加到您的重定向 URL 的 ID 和访问令牌。隐式授予的安全性较差，因为它会向用户公开令牌和潜在的识别信息。您可以在应用程序客户端的配置中停用对隐式授予的支持。

必需。

client_id

客户端 ID。

`client_id` 的值必须是您在其中发出请求的用户群体中应用程序客户端的 ID。您的应用程序客户端必须支持由 Amazon Cognito 本地用户或至少一个第三方 IdP 登录。

必需。

redirect_uri

在 Amazon Cognito 授权用户之后，身份验证服务器将浏览器重定向到的 URL。

重定向统一资源标识符 (URI) 必须具有以下属性：

- 必须是绝对 URI。
- 您必须已经将 URI 预注册到客户端。
- 不能包含片段组件。

请参阅 OAuth 2.0 – 重定向端点。

Amazon Cognito 要求您的重新导向 URI 使用 HTTPS，但 `http://localhost` 除外，您可以将其设置为回调 URL 以进行测试。

Amazon Cognito 还支持应用程序回调 URL，如 `myapp://example`。

必需。

state

当应用程序向请求添加 state 参数时，如果 `/oauth2/authorize` 端点重新导向您的用户，则 Amazon Cognito 将此参数的值返回给您的应用程序。

将此值添加到您的请求中以防止 CSRF 攻击。

不能将 `state` 参数的值设置为 URL 编码的 JSON 字符串。要在 `state` 参数中传递与此格式匹配的字符串，请将该字符串编码为 Base64，然后在应用程序中对其进行解码。

可选但不推荐。

identity_provider

添加此参数以绕过托管 UI 并将您的用户重定向到提供商登录页面。identity_provider 参数的值是出现在您的用户池中的身份提供商 (IdP) 的名称。

- 对于社交提供商，您可以使用以下 identity_provider 值：Facebook、Google（谷歌）、LoginWithAmazon（使用亚马逊账号登录）和 SignInWithApple（使用苹果账号登录）。
- 对于 Amazon Cognito 用户池，使用值 COGNITO。
- 对于 SAML 2.0 和 OpenID Connect (OIDC) 身份提供者 (IdP)，使用您在用户群体中分配给 IdP 的名称。

可选。

idp_identifier

添加此参数以重定向到具有 identity_provider 名称的替代名称的提供程序。您可以从 Amazon Cognito 控制台的 Sign-in experience（登录体验）选项卡中为您的 SAML 2.0 和 OIDC IdP 输入标识符。

可选。

scope

可以是任何系统预留范围或与客户端关联的自定义范围的组合。范围必须以空格分隔。系统预留范围为 `openid`、`email`、`phone`、`profile` 和 `aws.cognito.signin.user.admin`。使用的任意范围必须与客户端关联，否则将在运行时忽略。

如果客户端不请求任何范围，则身份验证服务器使用与客户端关联的所有范围。

如果请求 `openid` 范围，则只返回 ID 令牌。如果请求 `aws.cognito.signin.user.admin` 范围，则访问令牌只能用于 Amazon Cognito 用户池。如果同时请求了 phone 范围，则只能请求 `email`、`profile` 和 `openid` 范围。这些范围控制进入 ID 令牌中的声明。

可选。

code_challenge_method

用于生成质询的方法。PKCE RFC 定义两个方法：S256 和 plain；但是，Amazon Cognito 身份验证服务器仅支持 S256。

可选。

code_challenge

从 `code_verifier` 生成的质询。

仅当您指定 `code_challenge_method` 参数时是必需的。

nonce

您可以添加到请求中的随机值。您提供的 nonce 值包含在 Amazon Cognito 发出的 ID 令牌中。为了防范重播攻击，您的应用程序可以检查 ID 令牌中的 `nonce` 声明并将其与您生成的声明进行比较。有关 `nonce` 声明的更多信息，请参阅《OpenID Connect 标准》中的 ID token validation（ID 令牌验证）。

具有正向响应的示例请求

授予授权代码

示例请求

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
                                response_type=code&
                                client_id=ad398u21ijw3s9w3939&
                                redirect_uri=https://YOUR_APP/redirect_uri&
                                state=STATE&
                                scope=openid+profile+aws.cognito.signin.user.admin
```

示例响应

Amazon Cognito 身份验证服务器使用授权代码和状态重新导向回您的应用程序。代码和状态必须在查询字符串参数中返回，而不是在片段中。查询字符串是 Web 请求的一部分，显示在“?”字符之后；该字符串可以包含一个或多个使用“&”字符分隔的参数。片段是 Web 请求中显示在“#”字符后的部分，用于指定文档的子部分。

Note

响应返回有效期为 5 分钟的一次性使用代码。

```
HTTP/1.1 302 Found
                    Location: https://YOUR_APP/redirect_uri?
code=AUTHORIZATION_CODE&state=STATE
```

具有 PKCE 的授权代码授予

示例请求

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
                            response_type=code&
                            client_id=ad398u21ijw3s9w3939&
                            redirect_uri=https://YOUR_APP/redirect_uri&
```

```
                        state=STATE&
                        scope=aws.cognito.signin.user.admin&
                        code_challenge_method=S256&
                        code_challenge=CODE_CHALLENGE
```

示例响应

身份验证服务器使用授权代码和状态重定向回您的应用程序。代码和状态必须在查询字符串参数中返回，而不是在片段中。

```
HTTP/1.1 302 Found
                    Location: https://YOUR_APP/redirect_uri?
code=AUTHORIZATION_CODE&state=STATE
```

不带 openid 范围的令牌授予

示例请求

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
                        response_type=token&
                        client_id=ad398u21ijw3s9w3939&
                        redirect_uri=https://YOUR_APP/redirect_uri&
                        state=STATE&
                        scope=aws.cognito.signin.user.admin
```

示例响应

Amazon Cognito 授权服务器使用访问令牌重新导向回您的应用程序。由于未请求 openid 范围，Amazon Cognito 不会返回 ID 令牌。此外，Amazon Cognito 不会在此流程中返回刷新令牌。Amazon Cognito 在片段中返回访问令牌和状态，而不是在查询字符串中。

```
HTTP/1.1 302 Found
                       Location: https://YOUR_APP/
redirect_uri#access_token=ACCESS_TOKEN&token_type=bearer&expires_in=3600&state=STATE
```

具有 openid 范围的令牌授予

示例请求

```
                       GET
                           https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/
authorize?
                           response_type=token&
                           client_id=ad398u21ijw3s9w3939&
                           redirect_uri=https://YOUR_APP/redirect_uri&
                           state=STATE&
                           scope=aws.cognito.signin.user.admin+openid+profile
```

示例响应

授权服务器重定向回您的应用程序，带有访问令牌和 ID 令牌 (因为包括了 openid 范围)。

```
HTTP/1.1 302 Found
                          Location: https://YOUR_APP/
redirect_uri#id_token=ID_TOKEN&access_token=ACCESS_TOKEN&token_type=bearer&expires_in=3600&state=STATE
```

负向响应的示例

以下是负向响应的示例：

- 如果 client_id 和 redirect_uri 有效，但请求参数格式不正确，身份验证服务器会将该错误重定向到客户端的 redirect_uri 并在 URL 参数中附加错误消息。格式错误的示例包括：如果响应提供 code_challenge 而不是 code_challenge_method，或者 code_challenge_method 不是“S256”，则请求不包含 response_type 参数。

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?
  error=invalid_request
  ```
- 如果客户端在 response_type 中请求 code 或 token，但没有对于这些请求的权限，则 Amazon Cognito 授权服务器将 unauthorized_client 返回到客户端的 redirect_uri，如下所示：

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?
  error=unauthorized_client
  ```
- 如果客户端请求范围未知、格式错误或者无效，则 Amazon Cognito 授权服务器会将 invalid_scope 返回到客户端的 redirect_uri，如下所示：

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_scope
  ```
- 如果服务器中有意外的错误，则授权服务器会将 server_error 返回到客户端的 redirect_uri。由于 HTTP 500 错误不会发送到客户端，所以不在浏览器中向用户显示错误。应生成以下错误：

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=server_error
  ```
- 当 Amazon Cognito 通过联合第三方 IdP 进行身份验证时，Amazon Cognito 可能会遇到以下连接问题：
  - 如果从 IdP 处请求令牌时连接超时，身份验证服务器会将该错误重定向到客户端的 redirect_uri，如下所示：

    ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Timeout+occurred+in+calling+IdP
    +token+endpoint
    ```
  - 如果在调用 jwks 端点进行 id_token 验证时连接超时，身份验证服务器会将错误重新导向到客户端的 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
    ```
- 通过联合第三方 IdP 进行身份验证时，提供程序可能会因配置错误或其他原因返回错误响应，如下所示：
  - 如果从其他提供商处收到错误响应，身份验证服务器会将该错误重定向到客户端的 redirect_uri，如下所示：

    ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=[IdP name]+Error+-+[status
    code]+error getting token
    ```
  - 如果从 Google 收到错误响应，身份验证服务器会将错误重新导向到客户端的 redirect_uri，如下所示：HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Google+Error+-+[status code]+[Google
    provided error code]
- 如果 Amazon Cognito 在与外部 IdP 建立连接时遇到通信协议异常，身份验证服务器会将该错误重定向到客户端的 redirect_uri，并显示以下错误消息：

- `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`

令牌端点

生成发送到 `/oauth2/token` 端点的 POST 请求，以便为用户或服务获取 JSON Web 令牌（JWT，JSON Web Token）。当您向用户群体添加域时，Amazon Cognito 会激活专用于您的用户群体的 OAuth 2.0 令牌端点。在基于用户的模型中，您的应用程序向令牌端点发送授权代码，用于交换 ID、访问权限和刷新令牌。在机器对机器的模型中，您的应用程序向令牌端点发送客户端密钥，用于交换访问令牌。

要从用户的访问令牌中检索有关用户的信息，您可以将其传递给 UserInfo 终端节点 (p. 566) 或 GetUser API 请求。

POST /oauth2/token

`/oauth2/token` 终端节点只支持 `HTTPS POST`。您的应用程序直接对此端点发出请求，而不通过用户浏览器。

令牌端点支持 `client_secret_basic` 和 `client_secret_post` 身份验证。有关 OpenID Connect 规范的更多信息，请参阅客户端身份验证。有关 OpenID Connect 规范中令牌端点的更多信息，请参阅令牌端点。

标头中的请求参数

Authorization

如果向客户端发布了密钥，则客户端可以在授权标头中将其 `client_id` 和 `client_secret` 作为 `client_secret_basic` HTTP 授权传递。您还可以在请求正文中包含 `client_id` 和 `client_secret` 作为 `client_secret_post` 授权。

授权标头字符串是 基本 `Base64Encode(client_id:client_secret)`。以下示例是具有客户端密钥 `abcdef01234567890` 的应用程序客户端 `djc98u3jiedmi283eu928` 的授权标头，其使用字符串 `djc98u3jiedmi283eu928:abcdef01234567890` 的 Base64 编码版本。

```
Authorization: Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw
```

Content-Type

必须始终为 `'application/x-www-form-urlencoded'`。

正文中的请求参数

grant_type

授予类型。

必须为 `authorization_code`、`refresh_token` 或 `client_credentials`。当在应用程序客户端中已启用请求的范围、已配置客户端密钥并且具有允许的 `client_credentials` 授予时，您可以通过令牌端点为自定义范围请求访问令牌。

必需。

client_id

用户群体中应用程序客户端的 ID。必须指定对用户进行身份验证的同一个应用程序客户端。

如果客户端是公有的且没有密钥，则为必需的；或者，如果想要使用 client_secret_post 授权，则使用 client_secret。

client_secret

对用户进行身份验证的应用程序客户端的客户端密钥。如果您的应用程序客户端具有客户端密钥，并且您未发送 Authorization 标头，则为必需的。

scope

可以是与应用程序客户端关联的任何自定义范围的组合。必须为应用程序客户端激活您请求的任何范围，否则 Amazon Cognito 将忽略它。如果客户端不请求任何范围，则身份验证服务器使用与客户端关联的所有自定义范围。

可选。仅当 grant_type 为 client_credentials 时使用。

redirect_uri

必须是用于在 /oauth2/authorize 中获取 authorization_code 的相同 redirect_uri。

仅当 grant_type 为 authorization_code 时必需。

refresh_token

要为用户的会话生成新的访问令牌和 ID 令牌，请将您的 /oauth2/token 请求中的 refresh_token 参数的值设置为同一个应用程序客户端中以前颁发的刷新令牌。

代码

如果 grant_type 为 authorization_code，则必需。

code_verifier

证明密钥。

如果 grant_type 为 authorization_code 并且使用 PKCE 请求了授权代码，则必需。

具有正向响应的示例请求

为获取令牌交换授权代码

示例请求

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token&
                            Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                            grant_type=authorization_code&
                            client_id=1example23456789&
                            code=AUTHORIZATION_CODE&
                            redirect_uri=com.myclientapp://myclient/redirect
```

示例响应

```
HTTP/1.1 200 OK

                           Content-Type: application/json

                           {
                            "access_token":"eyJra1example",
                            "id_token":"eyJra2example",
                            "refresh_token":"eyJj3example",
                            "token_type":"Bearer",
                            "expires_in":3600
```

```
                            }
```

Note

仅当 grant_type 为 authorization_code 时，令牌终端节点才返回 refresh_token。

为获取访问令牌交换客户端凭证

示例请求

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
                            Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                            grant_type=client_credentials&
                            client_id=1example23456789&

 scope=resourceServerIdentifier1/scope1 resourceServerIdentifier2/scope2
```

示例响应

```
HTTP/1.1 200 OK
                            Content-Type: application/json

                            {
                            "access_token":"eyJra1example",
                            "token_type":"Bearer",
                            "expires_in":3600
                            }
```

为获取令牌交换具有 PKCE 的授权代码授予

示例请求

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token
                                Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                                grant_type=authorization_code&
                                client_id=1example23456789&
                                code=AUTHORIZATION_CODE&
                                code_verifier=CODE_VERIFIER&
                                redirect_uri=com.myclientapp://myclient/redirect
```

示例响应

```
HTTP/1.1 200 OK
                            Content-Type: application/json

                            {
                            "access_token":"eyJra1example",
                            "id_token":"eyJra2example",
                            "refresh_token":"eyJj3example",
                            "token_type":"Bearer",
                            "expires_in":3600
                            }
```

Note

仅当 grant_type 为 authorization_code 时，令牌终端节点才返回 refresh_token。

为获取令牌交换刷新令牌

示例请求

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
                            Content-Type='application/x-www-form-urlencoded'&

 Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

                            grant_type=refresh_token&
                            client_id=1example23456789&
                            refresh_token=eyJj3example
```

示例响应

```
HTTP/1.1 200 OK
                            Content-Type: application/json

                            {
                            "access_token":"eyJra1example",
                            "id_token":"eyJra2example",
                            "token_type":"Bearer",
                            "expires_in":3600
                                  }
```

Note

仅当 grant_type 为 authorization_code 时，令牌终端节点才返回 refresh_token。

负向响应的示例

错误响应示例

```
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

请求缺少必需的参数、包括不支持的参数值 (除了 unsupported_grant_type 之外) 或者格式错误。例如，grant_type 是 refresh_token，但未包括 refresh_token。

invalid_client

客户端身份验证失败。例如，客户端的授权标头中包含 client_id 和 client_secret，但没有这样的客户端带有 client_id 和 client_secret。

invalid_grant

已撤销刷新令牌。

授权代码已使用或不存在。

应用程序客户端对请求的范围内的所有属性都没有读取权限。例如，您的应用程序请求 email 范围，应用程序客户端可以读取 email 属性，但不能读取 email_verified。

unauthorized_client

客户端不允许代码授予流或刷新令牌。

unsupported_grant_type

如果 grant_type 是 authorization_code、refresh_token 或 client_credentials 之外的任意内容，则返回。

UserInfo 终端节点

userInfo 端点是 OpenID Connect（OIDC）userInfo 端点。当服务提供商出示您的 令牌端点 (p. 562) 颁发的访问令牌时，它会使用用户属性进行响应。用户访问令牌中的范围定义了 userInfo 端点在其响应中返回的用户属性。openid 范围必须是访问令牌声明的范围之一。

Amazon Cognito 颁发访问令牌来响应用户群体 API 请求，如 InitiateAuth。由于它们不包含任何范围，因此 userInfo 端点不接受这些访问令牌。而是必须由您出示来自令牌端点的访问令牌。

您的 OAuth 2.0 第三方身份提供者（IdP）还托管 userInfo 端点。当您的用户使用该 IdP 进行身份验证时，Amazon Cognito 会以静默方式与 IdP token 端点交换授权代码，并从 IdP userInfo 端点检索用户信息。

GET /oauth2/userInfo

您的应用程序直接对此端点发出请求，而不通过浏览器。

有关更多信息，请参阅 OpenID Connect（OIDC）规范中的 UserInfo 端点。

主题

- 标头中的请求参数 (p. 566)
- 示例请求 (p. 566)
- 示例正向响应 (p. 566)
- 示例负向响应 (p. 567)

标头中的请求参数

Authorization: Bearer *<access_token>*

使用授权标头字段传递访问令牌。

必需。

示例请求

```
GET https://<your-user-pool-domain>/oauth2/userInfo
                Authorization: Bearer <access_token>
```

示例正向响应

```
HTTP/1.1 200 OK
                        Content-Type: application/json;charset=UTF-8
```

```
{
   "sub": "248289761001",
   "name": "Jane Doe",
   "given_name": "Jane",
   "family_name": "Doe",
   "preferred_username": "j.doe",
   "email": "janedoe@example.com"
}
```

有关 OIDC 声明的列表，请参阅标准声明。

示例负向响应

请求无效

```
HTTP/1.1 400 Bad Request
                WWW-Authenticate: error="invalid_request",
                  error_description="Bad OAuth2 request at UserInfo Endpoint"
```

invalid_request

请求缺少必需的参数、包括不支持的参数值或者格式错误。

令牌无效

```
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

访问令牌已过期、已撤销、格式不正确或无效。

撤消端点

/oauth2/revoke 端点撤销 Amazon Cognito 最初使用您提供的刷新令牌颁发的用户访问令牌，以及来自同一刷新令牌的所有后续访问令牌和身份令牌。端点撤销令牌后，您无法使用已撤销的访问令牌访问 Amazon Cognito 令牌进行身份验证的 API。

POST /oauth2/revoke

/oauth2/revoke 终端节点只支持 HTTPS POST。用户池客户端直接对此终端节点发出请求，而不通过系统浏览器。

标头中的请求参数

Authorization

如果应用程序客户端具有客户端密钥，则客户端必须通过基本 HTTP 授权在其授权标头中传递 client_id 和 client_secret。密钥是基本 Base64Encode(client_id:client_secret)。

Content-Type

必须始终为 'application/x-www-form-urlencoded'。

正文中的请求参数

token

客户端要撤消的刷新令牌。请求还撤销 Amazon Cognito 使用此刷新令牌颁发的所有访问令牌。

必需。

client_id

您要撤消的令牌的应用程序客户端 ID。

如果客户端是公有的且没有密钥，则为必需。

撤消请求示例

示例 1：为没有客户端密钥的应用程序客户端撤消令牌

```
POST /oauth2/revoke HTTP/1.1
    Host: https://mydomain.auth.us-east-1.amazoncognito.com
    Accept: application/json
    Content-Type: application/x-www-form-urlencoded
    token=2YotnFZFEjr1zCsicMWpAA&
    client_id=djc98u3jiedmi283eu928
```

示例 2：为具有客户端密钥的应用程序客户端撤消令牌

```
POST /oauth2/revoke HTTP/1.1
    Host: https://mydomain.auth.us-east-1.amazoncognito.com
    Accept: application/json
    Content-Type: application/x-www-form-urlencoded
    Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
    token=2YotnFZFEjr1zCsicMWpAA
```

撤消错误响应

成功的响应包含空正文。错误响应是一个带有 `error` 字段和 `error_description` 字段（在某些情况下）的 JSON 对象。

终端节点错误

- 如果令牌在请求中不存在或者应用程序客户端禁用了该功能，则您会收到 HTTP 400 和错误 `invalid_request`。
- 如果 Amazon Cognito 在撤销请求中发送的令牌不是刷新令牌，则您会收到 HTTP 400 和错误 `unsupported_token_type`。
- 如果客户端凭证无效，则您会收到 HTTP 401 和错误 `invalid_client`。
- 如果令牌已撤销或客户端提交的令牌无效，您会收到 HTTP 200 OK。

托管 UI 和联合身份验证错误响应

托管 UI 中的登录过程或联合登录可能会返回错误。以下是一些可能导致身份验证以错误结束的情况。

- 用户执行了您的用户群体无法完成的操作。

- Lambda 触发器不以预期的语法响应。
- 您的身份提供者（IdP）返回错误。
- Amazon Cognito 无法验证您的用户提供的属性信息。
- 您的 IdP 未发送与所需属性对应的声明。

当 Amazon Cognito 遇到错误时，它通过以下方式之一进行通信。

1. Amazon Cognito 发送的重定向 URL 的请求参数中存在错误。
2. Amazon Cognito 在托管 UI 中显示错误。

Amazon Cognito 附加到请求参数的错误具有以下格式。

```
https://<Callback URL>/?error_description=error+description&error=error+name
```

如果您帮助用户在无法执行操作时提交错误信息，则要求他们捕获 URL 和文本或页面的屏幕截图。

OIDC 和社交身份提供者错误消息

您的身份提供者可能会返回错误。当 OIDC 或 OAuth 2.0 IdP 返回符合标准的错误时，Amazon Cognito 将您的用户重定向到回调 URL，并将提供者错误响应添加到错误请求参数中。Amazon Cognito 将提供者名称和 HTTP 错误代码添加到现有的错误字符串中。

以下 URL 是从返回错误的 IdP 重定向到 Amazon Cognito 的示例。

```
https://www.amazon.com/?error_description=LoginWithAmazon+Error+-+400+invalid_request+The
+request+is+missing+a+required+parameter+%3A+client_secret&error=invalid_request
```

由于 Amazon Cognito 仅返回它从提供者处收到的内容，因此您的用户可能会看到这些信息的一部分。

如果用户在通过您的 IdP 进行初始登录时遇到问题，IdP 会直接向用户发送任何错误消息。当 Amazon Cognito 向您的 IdP 生成验证用户会话的请求时，它会向您的用户中继一条错误消息。Amazon Cognito 中继来自以下端点的 OAuth 和 OIDC IdP 错误消息。

`/token`

Amazon Cognito 交换 IdP 授权代码以获得访问令牌。

`/.well-known/openid-configuration`

Amazon Cognito 发现了通往发行者端点的路径。

`/.well-known/jwks.json`

为验证用户的 JSON 网络令牌（JWT），Amazon Cognito 发现您的 IdP 用来签署令牌的 JSON 网络密钥（JWT）。

由于 Amazon Cognito 不会启动到可能返回 HTTP 错误的 SAML 2.0 提供者的出站会话，因此，用户在与 SAML 2.0 IdP 会话期间出现的错误不包括这种形式的提供者错误消息。

Amazon Cognito 用户池 API 参考

有了 Amazon Cognito 用户群体，您可以在 Web 和移动应用程序中注册和登录用户。您可以更改经过身份验证的用户的密码，并为未经身份验证的用户启动忘记密码流程。有关更多信息，请参阅 用户池身份验证流程 (p. 41) 和 将令牌与用户池结合使用 (p. 203)：

Amazon Cognito 用户群体 API 包括多种操作，可以查看和修改您的用户群体及用户，以及执行用户身份验证和授权。有关合并到 Amazon Cognito 用户群体 API 中的 API 操作类别的描述，请参阅 使用 Amazon Cognito 用户群体 API 和用户群体端点 (p. 47)。

有关 Amazon Cognito 用户群体 API 操作和语法的详细列表，请参阅 Amazon Cognito 用户群体 API 参考。《Amazon Cognito 用户群体 API 参考》中的每个页面都链接到了参考材料，提供有关各种 AWS SDK 的语法和示例。

Amazon Cognito 身份池（联合身份）API 参考

有了 Amazon Cognito 身份池，您的 Web 和移动应用程序用户便可以获得权限有限的临时AWS凭证，使他们能够访问其他AWS服务。

有关完整的身份池（联合身份）API 参考，请参阅 Amazon Cognito API 参考。

Amazon Cognito Sync API 参考

Amazon Cognito Sync 是一种AWS服务和客户端库，用于跨设备同步与应用程序相关的用户数据。

有关 Amazon Cognito Sync API 参考的更多信息，请参阅 Amazon Cognito Sync API 参考。

Amazon Cognito 文档历史记录

下表介绍 Amazon Cognito 文档的重要补充部分。我们还会根据您发送的反馈对文档进行频繁的小幅更新。要提交反馈，请找到 Amazon Cognito 文档中任何页面底部的反馈链接。

更改	说明	日期
对服务主页和用户群体主页的添加和改进。	更新了 Amazon Cognito 和 Amazon Cognito 用户群体的概述。	2023 年 5 月 16 日
对用户群体令牌文档进行了总体改进。	更新了示例令牌，添加了有关验证令牌的新信息。请参阅将令牌与用户池结合使用 (p. 203)。	2023 年 2 月 16 日
您现在可以在 AWS CloudTrail 中记录 Amazon Cognito 身份池数据事件。	CloudTrail 支持在记录数据事件的跟踪记录中选择 Amazon Cognito 身份池大容量 API 操作。请参阅Amazon Cognito 身份池 (p. 515)。	2023 年 2 月 15 日
Amazon Cognito 身份池将 AWS 托管式策略应用于未经身份验证的会话。	使用增强流程进行身份验证的身份池用户现在可以对其会话应用额外的 AWS 托管式策略。请参阅访问策略 (p. 306)。	2023 年 1 月 31 日
更新了 Lambda 触发器示例和描述。	Lambda 触发器示例已更新到 JavaScript 版本 3。您现在可以直接将 Lambda 触发器与 API 操作相关。请参阅使用 Lambda 触发器自定义用户池工作流 (p. 111)。	2023 年 1 月 31 日
添加了代码示例。	本指南现在包含各种编程语言的 Amazon Cognito 应用程序的示例代码。请参阅使用 AWS SDK 的 Amazon Cognito 的代码示例。	2023 年 1 月 23 日
添加了有关 API 模型和使用 Amazon Cognito 用户群体进行身份验证的信息。	Amazon Cognito 用户群体具有多种用于请求授权的 API 接口和格式。参阅 使用 Amazon Cognito 用户群体 API 和用户群体端点 (p. 47) 了解更多信息。	2022 年 12 月 15 日
Amazon Cognito 现已在欧洲地区（米兰）AWS 区域 推出。	您现在可以在欧洲地区（米兰）区域创建 Amazon Cognito 用户群体。	2022 年 12 月 6 日
添加了有关用户群体删除保护的信息。	当您使用 AWS Management Console 创建新的用户群体时，默认会启用删除保护。	2022 年 10 月 20 日
添加了托管 UI 的用户指南以及有关托管 UI 中的 TOTP MFA 的信息。	您的用户现在可以在 Amazon Cognito 托管 UI 中注册 TOTP MFA 设备。现在，您可以在使用托管 UI 注册和登录 (p. 71)中预览默认的托管 UI。	2022 年 9 月 8 日

更改	说明	日期
添加有关 AWS WAF 和 Amazon Cognito 的信息。	现在，您可以将一个 AWS WAF Web ACL 与 Amazon Cognito 用户池相关联。请参阅将 AWS WAF Web ACL 与用户池相关联。	2022 年 8 月 3 日
添加了更多示例 AWS CloudTrail 事件。	Amazon Cognito 现在会将联合身份验证和托管 UI 请求记录到您的跟踪。	2022 年 6 月 15 日
添加了有关两步属性验证的信息。	现在，您可以选择您的用户是否必须验证新的电子邮件地址或电话号码，然后才能使用该地址或电话号码登录。请参阅验证对于电子邮件地址和电话号码的更新。	2022 年 6 月 9 日
更新了联合身份验证文档。新的 IP 地址传播功能。	更新了用于设置用户群体社交 IdP 的演练。添加了有关联合用户配置文件和属性映射的信息。添加了有关 将用户设备和会话数据添加到 API 请求 (p. 231)的新信息。	2022 年 5 月 31 日
使联合身份用户登录，而无需与托管 UI 交互	添加了一个关于如何在企业控制面板中为 Amazon Cognito 应用程序添加书签 (p. 101)以便 Amazon Cognito 静默地将用户定向到联合登录。	2022 年 3 月 29 日
Amazon Cognito 用户池的区域内 SMS 和电子邮件消息	现在，您可以在您的用户池所在的同一 AWS 区域 中可以将 Amazon Simple Notification Service 用于 SMS 消息，将 Amazon Simple Email Service 用于电子邮件消息。	2022 年 3 月 14 日
配额页面更新	添加并澄清了资源和Amazon Cognito 中的限额 (p. 539)的请求费率配额。	2022 年 1 月 10 日
全新 Amazon Cognito 用户池控制台体验	更新了在更新后的 Amazon Cognito 控制台中创建和管理用户池的说明。	2021 年 11 月 18 日
RevokeToken API 和撤销端点	您可以使用 RevokeToken 操作为用户撤消刷新令牌 (p. 209)。	2021 年 6 月 10 日
向 GitHub 发布指南 Markdown 格式	与所有AWS文档一样，本指南现在提供了 Markdown 格式，可供查看和发表评论，网址为 https://github.com/awsdocs/amazon-cognito-developer-guide。	2021 年 3 月 23 日
多租户最佳实践	在文档中添加了多租户应用程序最佳实践。	2021 年 3 月 4 日

更改	说明	日期
访问控制属性	Amazon Cognito 身份池提供访问控制属性（AFAC），作为客户授予用户访问AWS资源的权限一种方法。授权可以根据用户用来与 Amazon Cognito 联合的身份提供商提供的用户属性执行。	2021 年 1 月 15 日
自定义 SMS 发件人 Lambda 触发器和自定义电子邮件发件人 Lambda 触发器	您可以利用自定义 SMS 发件人 Lambda 触发器和自定义电子邮件发件人 Lambda 触发器使第三方提供商从 Lambda 函数代码中向用户发送电子邮件和 SMS 通知。	2020 年 11 月
Amazon Cognito 令牌更新	在访问令牌、ID 令牌和刷新令牌中添加了更新的过期信息。	2020 年 10 月 29 日
Amazon Cognito Service Quotas	Service Quotas 可用于 Amazon Cognito 类别配额。您可以使用 Service Quotas 控制台查看配额使用情况，请求增加配额和创建 CloudWatch 告警来监控您的配额使用情况。在此次更改中，更新了“Amazon Cognito 用户池的可用 CloudWatch 指标”章节，反映最新信息。新章节的名称为：在 CloudWatch 和 Service Quotas 中跟踪配额和使用情况	2020 年 10 月 29 日
Amazon Cognito 配额分类	配额分类可用于帮助您监控配额使用情况并请求增加配额。配额根据常见使用案例分为不同的类别。	2020 年 8 月 17 日
Amazon Cognito Pinpoint 文档更新	添加了新的服务相关角色。更新了有关“将 Amazon Pinpoint 分析与 Amazon Cognito 用户池结合使用”的说明。	2020 年 5 月 13 日
美国 AWS GovCLoud 支持 Amazon Cognito	AWS GovCloud (US) 区域现在支持 Amazon Cognito。	2020 年 5 月 13 日
新 Amazon Cognito 专用安全性章节	“安全性”一章可帮助您的组织深入了解AWS服务的内置安全性和可配置安全性。我们的新章节提供有关云及云中安全性的信息。	2020 年 4 月 30 日
Amazon Cognito 身份池现在支持 Sign in with Apple	Sign in with Apple 功能现已在所有运营 Amazon Cognito 的区域推出，cn-north-1 region 除外。	2020 年 4 月 7 日
新的 Facebook API 版本控制	在 Facebook API 中添加了版本选择。	2020 年 4 月 3 日
用户名不区分大小写更新	添加了关于在创建用户池之前启用用户名不区分大小写 (p. 30)的建议。	2020 年 2 月 11 日

更改	说明	日期
有关 AWS Amplify 的新信息	添加了有关使用 AWS Amplify SDK 和库将 Amazon Cognito 与 Web 或移动应用程序集成 (p. 20)的信息。删除了在 AWS Amplify 之前使用 Amazon Cognito SDK 的信息。	2019 年 11 月 22 日
用户池触发器的新属性	Amazon Cognito 的事件信息中现在包含一个 `clientMetadata` 参数，该参数会传递给大多数用户池触发器的 AWS Lambda 函数。您可以使用此参数来通过其他数据强化自定义身份验证工作流。	2019 年 10 月 4 日
更新限制	更新了对 ListUsers API 操作的限制。有关更多信息，请参阅Amazon Cognito 中的限额 (p. 539)。	2019 年 6 月 25 日
新限制	用户池中的软性限制现在包含对用户数量的限制。有关更多信息，请参阅Amazon Cognito 中的限额 (p. 539)。	2019 年 6 月 17 日
Amazon Cognito 用户池的 Amazon SES 电子邮件设置	您可以配置用户池，以便 Amazon Cognito 使用您的 Amazon SES 配置向用户发送电子邮件。此设置允许 Amazon Cognito 以比其他可能的方式更高的送达量发送电子邮件。有关更多信息，请参阅Amazon Cognito 用户池的电子邮件设置 (p. 194)。	2019 年 4 月 8 日
标记支持	添加了有关标记 Amazon Cognito 资源 (p. 534)的信息。	2019 年 3 月 26 日
更改自定义域的证书	如果您使用自定义域托管 Amazon Cognito 托管 UI，则可以根据需要更改此域的 SSL 证书。有关更多信息，请参阅更改自定义域的 SSL 证书 (p. 66)。	2018 年 12 月 19 日
新限制	添加了针对每个用户可以属于的最大组数的新限制。有关更多信息，请参阅Amazon Cognito 中的限额 (p. 539)。	2018 年 12 月 14 日
已更新限制	更新了用户池的软限制。有关更多信息，请参阅Amazon Cognito 中的限额 (p. 539)。	2018 年 12 月 11 日

更改	说明	日期
用于验证电子邮件地址和电话号码的文档更新	添加了有关配置用户池以在用户注册您的应用程序时要求进行电子邮件或电话验证的信息。有关更多信息，请参阅在注册时验证联系人信息 (p. 169)。	2018 年 11 月 20 日
用于测试电子邮件的文档更新	添加了有关在测试应用程序时从 Amazon Cognito 启动电子邮件传送的指导信息。有关更多信息，请参阅在测试应用程序时发送电子邮件 (p. 174)。	2018 年 11 月 13 日
Amazon Cognito 高级安全性	新增的安全功能使开发人员能够保护他们的应用程序和用户免受恶意的自动程序攻击，确保用户账户拒绝被盗用的凭据，并根据计算的登录尝试风险自动调整登录所需的难度。	2018 年 6 月 14 日
自定义域用于 Amazon Cognito 托管 UI	允许开发人员将自己的完全自定义域用于 Amazon Cognito 用户池中的托管 UI。	2018 年 4 月 6 日
Amazon Cognito 用户池 OIDC 身份提供商	已添加“通过 OpenID Connect (OIDC) 身份提供商 (如 Salesforce 或 Ping Identity) 登录用户池”。	2018 年 5 月 17 日
Amazon Cognito 开发人员指南更新	添加了顶层内容“什么是 Amazon Cognito”和“Amazon Cognito 入门”。还添加了常见场景并重新组织了用户池 TOC。添加了新的“Amazon Cognito 用户池入门”部分。	2018 年 4 月 6 日
Amazon Cognito Lambda 迁移触发器	添加了介绍 Lambda 迁移触发器功能的页面	2018 年 2 月 8 日
Amazon Cognito 高级安全性测试版	添加的新安全功能，可让开发人员保护应用和用户远离恶意自动程序，确保已在 Internet 上泄露的凭证无法登录用户账户，并根据登录尝试计算出风险，据此自动调整登录所需的难度。	2017 年 11 月 28 日
Amazon Pinpoint 集成	添加了使用 Amazon Pinpoint 为您的 Amazon Cognito 用户池应用程序提供分析并丰富 Amazon Pinpoint 活动用户数据的功能。有关更多信息，请参阅将 Amazon Pinpoint 分析与 Amazon Cognito 用户池结合使用 (p. 164)。	2017 年 9 月 26 日

更改	说明	日期
Amazon Cognito 用户池的联合身份验证和内置应用程序 UI 功能	添加了允许用户通过 Facebook、Google、Login with Amazon 或 SAML 身份提供商登录用户池的功能。添加了可自定义的内置应用程序 UI 和带自定义声明的 OAuth 2.0 支持。	2017 年 8 月 10 日
HIPAA 和 PCI 合规性相关的功能变更	添加了允许用户使用电话号码或电子邮件地址作为用户名的功能。	2017 年 6 月 7 日
用户组和基于角色的访问控制功能	添加了创建和管理用户组的管理功能。管理员可以根据组成员资格和管理员创建的规则将 IAM 角色分配给用户。有关更多信息，请参阅向用户池添加组 (p. 178)和使用基于角色的访问控制 (p. 318)。	2016 年 12 月 15 日
文档更新	更新了 经开发人员验证的身份（身份池）(p. 357) 中的 iOS 代码示例。	2016 年 11 月 18 日
文档更新	添加了有关用户账户的确认流程的信息。有关更多信息，请参阅注册并确认用户账户 (p. 168)。	2016 年 11 月 9 日
创建用户账户功能	添加了通过 Amazon Cognito 控制台和 API 创建用户账户的管理功能。有关更多信息，请参阅以管理员身份创建用户账户 (p. 175)。	2016 年 10 月 6 日
文档更新	更新了显示如何将 AWS Lambda 触发器与用户池结合使用的示例。有关更多信息，请参阅使用 Lambda 触发器自定义用户池工作流 (p. 111)。	2016 年 9 月 27 日
用户导入功能	添加了 Cognito 用户池的批量导入功能。使用此功能将用户从现有身份提供商迁移到 Amazon Cognito 用户池。有关更多信息，请参阅通过 CSV 文件将用户导入用户池中 (p. 184)。	2016 年 9 月 1 日
Cognito 用户池公开发布	添加了 Cognito 用户池功能。借助此功能，使用用户池创建和维护用户目录，并将注册信息和登录信息添加到移动应用程序或 Web 应用程序中。有关更多信息，请参阅Amazon Cognito 用户池 (p. 27)。	2016 年 7 月 28 日

更改	说明	日期
SAML 支援	使用身份提供商通过安全断言标记语言 2.0 (SAML 2.0) 添加了对身份验证的支持。有关更多信息，请参阅SAML 身份提供商（身份池）(p. 355)。	2016 年 6 月 23 日
CloudTrail 集成	添加了与 AWS CloudTrail 的集成。有关更多信息，请参阅使用 AWS CloudTrail 记录 Amazon Cognito API 调用 (p. 513)。	2016 年 2 月 18 日
事件与 Lambda 的集成	使您能够执行 AWS Lambda 函数以响应 Amazon Cognito 中的重要事件。有关更多信息，请参阅Amazon Cognito Events (p. 397)。	2015 年 4 月 9 日
Amazon Kinesis 的数据流	提供了对数据流的控制和了解。有关更多信息，请参阅Amazon Cognito Streams (p. 396)。	2015 年 3 月 4 日
推送同步	启用了对无提示推送同步的支持有关更多信息，请参阅Amazon Cognito Sync (p. 371)。	2014 年 11 月 6 日
OpenID 连接支持	启用了对 OpenID Connect 提供商的支持。有关更多信息，请参阅身份池（联合身份）外部身份提供商 (p. 330)。	2014 年 10 月 23 日
添加了对经过开发人员验证的身份的支持	使拥有自己的身份验证和身份管理系统的开发人员被视为 Amazon Cognito 中的身份提供商。有关更多信息，请参阅经开发人员验证的身份（身份池）(p. 357)。	2014 年 9 月 29 日
Amazon Cognito 正式发布		2014 年 7 月 10 日