aws

开发人员指南

Amazon Cognito

Amazon Cognito: 开发人员指南

Table of Contents

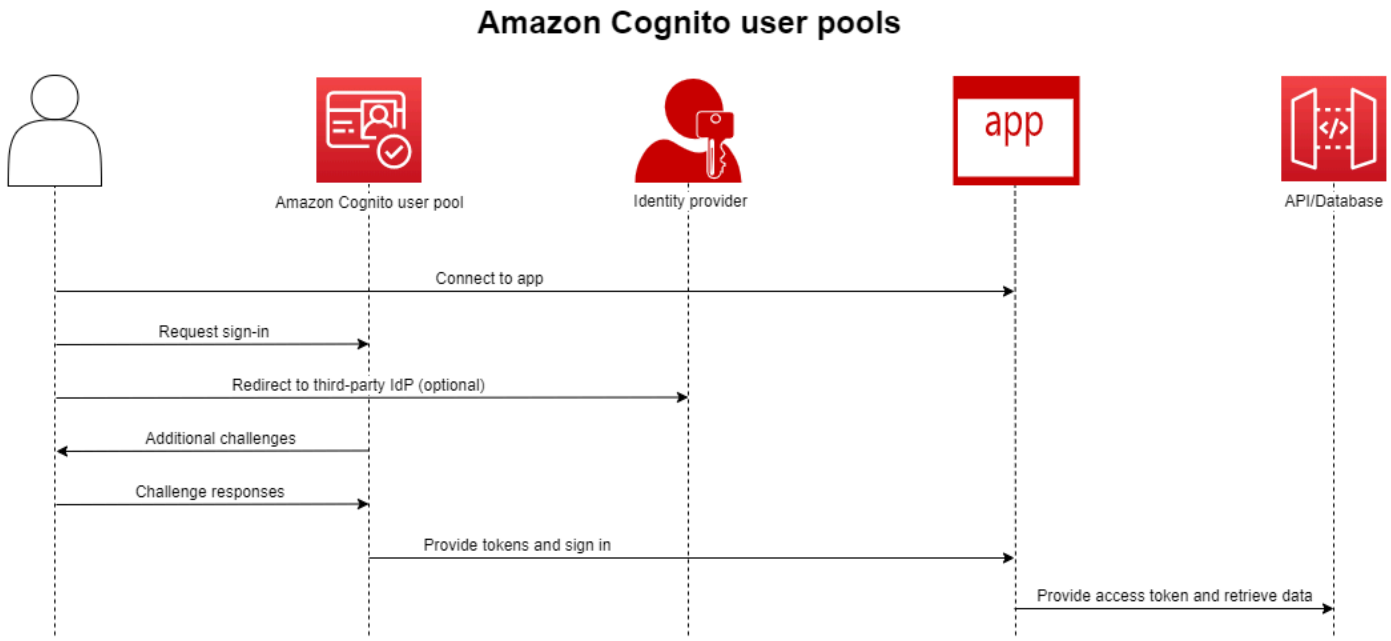
什么是 Amazon Cognito？

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

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。

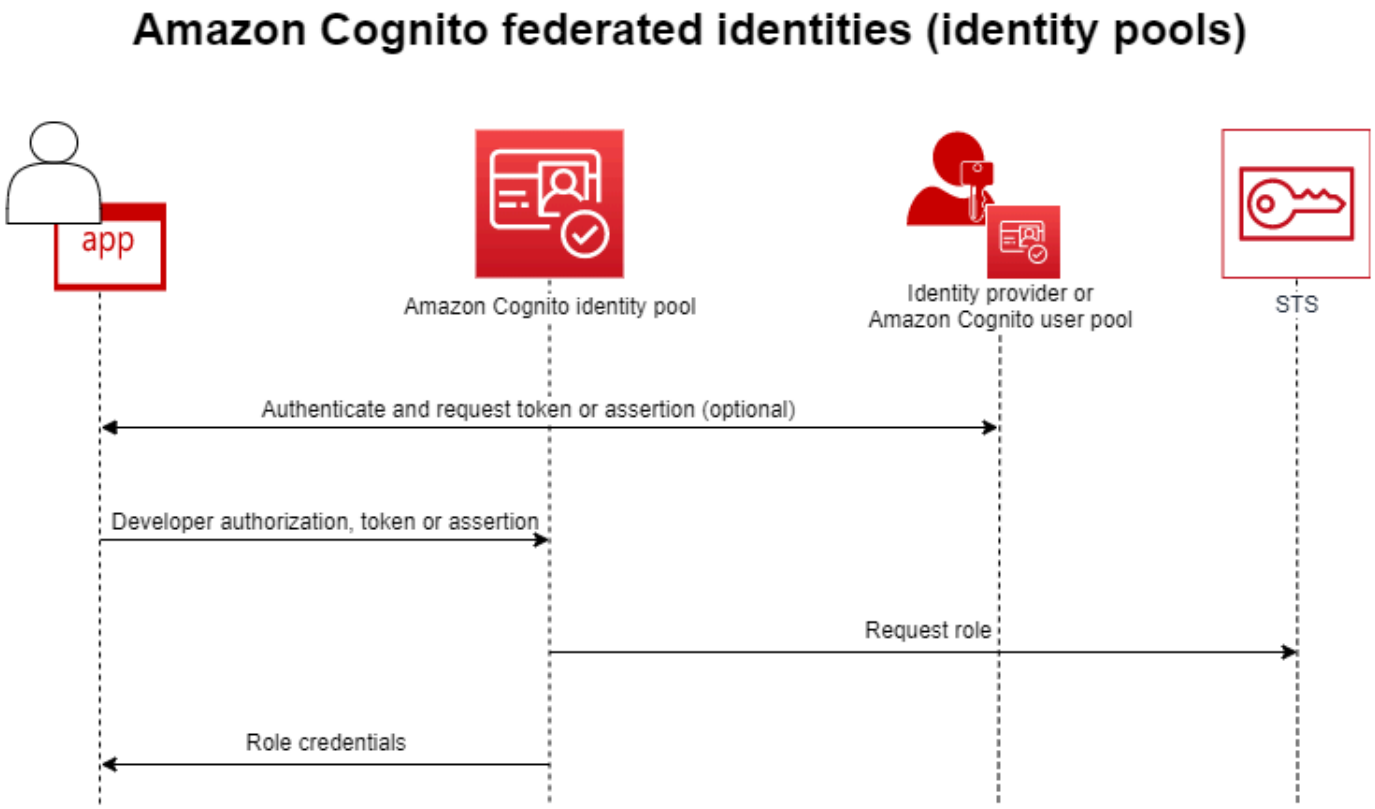
用户池



当您想要对您的应用程序或 API 进行身份验证和授权时，请创建用户池。用户池是一个用户目录，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。用户池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）。您可以在 SAML 2.0 和 OIDC 中为组织的工作人员身份提供单点登录 (SSO) 和带有用户池的 OIDC IdPs 。还可以在应用程序中，为 Amazon、Google、Apple 和 Facebook 等 Auth 2.0 公共身份存储中组织的客户身份提供 SSO。有关客户身份和访问管理（CIAM）的更多信息，请参阅什么是 CIAM？

用户池不要求与身份池集成。从用户池中，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



如果您想授权经过身份验证的用户或匿名用户访问您的 AWS 资源，请设置 Amazon Cognito 身份池。身份池会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（如用户池或 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。身份池使用基于角色和基于属性的访问控制来管理用户访问您的资源的授权。 AWS

身份池不要求与用户池集成。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 用户池和身份池一起使用

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池登录并接收 OAuth 2.0 令牌。
2. 您的应用程序将用户池令牌与身份池交换，以获得可以与 AWS API 和 AWS Command Line Interface (AWS CLI) 一起使用的临时 AWS 证书。

3. 您的应用程序会将凭证会话分配给您的用户，并授予对诸如 Amazon S3 和 Amazon DynamoDB 之 AWS 服务 类的授权访问权限。

有关使用身份池和用户池的更多示例，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 用户池是一个用户目录。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，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池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池中创建的用户。您可以在、 AWS SDK 或 AWS Command Line Interface (AWS CLI) 中 AWS Management Console管理和自定义这些用户配置文件。

Amazon Cognito 用户池接受来自第三方的令牌和断言 IdPs，并将用户属性收集到发布给您的应用程序的 JWT 中。您可以在一组 JWT 上标准化您的应用程序，而 Amazon Cognito 则处理 IdPs与之的互动，将其声明映射到中央令牌格式。

Amazon Cognito 用户池可以是独立的 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，用户池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 用户池 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 用户池还可以充当您的应用程序的服务提供商 (SP) 和您的应用程序的 IdPs IdP 的双重角色。Amazon Cognito 用户池可以连接到 Facebook 和谷歌 IdPs 等消费者，也可以连接 Okta 和 Active Directory 联合服务 (ADFS) IdPs 等员工。

有了 Amazon Cognito 用户池颁发的 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 证书。

Amazon Cognito 用户池的功能

功能	描述
OIDC IdP	发放 ID 令牌对用户进行身份验证
授权服务器	发放访问令牌以授权用户访问 API
SAML 2.0 SP	将 SAML 断言转换为 ID 和访问令牌
OIDC SP	将 OIDC 令牌转换为 ID 和访问令牌
OAuth 2.0 SP	将苹果、Facebook、亚马逊或谷歌的 ID 令牌转换为你自己的 ID 和访问令牌
认证前端服务	使用托管用户界面注册、管理和验证用户
为你自己的用户界面提供 API 支持	通过支持的 AWS 软件开发工具包中的 API 请求创建、管理和验证用户[1]
MFA	使用短信、TOTP 或用户的设备作为额外的身份验证因素[1]
安全监控和响应	抵御恶意活动和不安全的密码[1]
自定义身份验证流程	构建自己的身份验证机制，或在现有流程中添加自定义步骤[1]
组	在将令牌传递给身份池时，创建用户的逻辑分组和 IAM 角色声明的层次结构

自定义 ID 令牌	使用新的、修改过的和禁止的声明自定义您的 ID 令牌
自定义用户属性	为用户属性分配值并添加您自己的自定义属性

[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其接收临时 AWS 证书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。您的身份池为身份创建的令牌可以从 AWS Security Token Service (AWS STS) 检索临时会话证书。

为了补充经过身份验证的身份，您还可以将身份池配置为在没有 IdP 身份验证的情况下授权 AWS 访问。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 AWS 身份向任何请求临时凭证的应用程序用户授予临时证书。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池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，如果您的用户证明他们在市场营销部门工作，则会为他们的会话添加 AWS STS 标签`Department: marketing`。您的 Amazon S3 存储桶允许基于 a ws: PrincipalTag 条件进行读取操作，该条件要求`Department`标签的`marketing`值为。

Amazon Cognito 身份池的功能

功能	描述
亚马逊 Cognito 用户池 SP	使用用户池中的 ID 令牌交换来自的 Web 身份凭证 AWS STS
SAML 2.0 SP	交换来自 Web 身份凭证的 SAML 断言 AWS STS
OIDC SP	使用 OIDC 令牌兑换 Web 身份凭证 AWS STS
OAuth 2.0 SP	使用来自亚马逊、Facebook、谷歌、苹果和 Twitter 的 OAuth 代币来换取来自的网络身份凭证 AWS STS
自定义 SP	使用 AWS 凭证，以任何格式交换来自的 Web 身份凭证的声明 AWS STS
未经身份验证的访问	无需身份验证即可颁发访问受 AWS STS 限的 Web 身份凭证
基于角色的访问控制	根据身份验证用户的声明为其选择一个 IAM 角色，并将您的角色配置为仅在您的身份池环境中担任
基于属性的访问控制	将声明转换为 AWS STS 临时会话的委托人标签，并使用 IAM 策略根据委托人标签筛选资源访问权限

有关身份池的更多信息，请参阅开始使用 Amazon Cognito 身份池和 Amazon Cognito 身份池 API 参考。

Amazon Cognito 用户池和身份池的比较

功能	描述	用户池	身份池

OIDC IdP	发放 OIDC ID 令牌以对应用程序用户进行身份验证	✓	
API 授权服务器	发放访问令牌以授权用户访问接受 OAuth 2.0 授权范围的 API、数据库和其他资源	✓	
IAM 网络身份授权服务器	生成可用于交换临时 AWS 凭证 AWS STS 的令牌		✓
SAML 2.0 SP 和 OIDC IdP	根据来自 SAML 2.0 IdP 的主张发行定制的 OIDC 代币	✓	
OIDC SP 和 OIDC IdP	根据OIDC IdP的索赔发行定制的OIDC代币	✓	
OAuth 2.0 SP 和 OIDC IdP	根据苹果和谷歌等 OAuth 2.0 社交提供商的范围发行自定义 OIDC 代币	✓	
SAML 2.0 SP 和凭证代理	根据 SAML 2.0 IdP 的声明颁发临时 AWS 证书		✓
OIDC SP 和凭证经纪人	根据 OIDC IdP 的声明颁发临时 AWS 证书		✓
OAuth 2.0 SP 和凭证代理	根据苹果和谷歌等 OAuth 2.0 社交提供商的范围颁发临时 AWS 证书		✓

亚马逊 Cognito 用户池 SP 和凭证代理	根据 Amazon Cognito 用户池中的 OIDC 声明颁发临时 AWS 证书		✓
自定义 SP 和凭证代理	根据开发者 IAM 授权颁发临时 AWS 证书		✓
认证前端服务	使用托管用户界面注册、管理和验证用户	✓	
API 支持你自己的身份验证界面	通过支持的 AWS 软件开发工具包中的 API 请求创建、管理和验证用户[1]	✓	
MFA	使用短信、TOTP 或用户的设备作为额外的身份验证因素[1]	✓	
安全监控和响应	防范恶意活动和不安全的密码[1]	✓	
自定义身份验证流程	构建自己的身份验证机制，或在现有流程中添加自定义步骤[1]	✓	
组	在将令牌传递给身份池时，创建用户的逻辑分组和 IAM 角色声明的层次结构	✓	
自定义 ID 令牌	使用新的、修改过的和禁止的声明自定义您的 ID 令牌	✓	
AWS WAF 网页 ACL	通过以下方式监控和控制对身份验证环境的请求 AWS WAF	✓	

自定义用户属性	为用户属性分配值并添加您自己的自定义属性	✓	
未经身份验证的访问	无需身份验证即可颁发访问受 AWS STS 限的 Web 身份凭证		✓
基于角色的访问控制	根据身份验证用户的声明为其选择一个 IAM 角色，并将您的角色配置为仅在您的身份池环境中担任		✓
基于属性的访问控制	将用户声明转换为 AWS STS 临时会话的委托人标签，并使用 IAM 策略根据委托人标签筛选资源访问权限		✓

[1] 功能仅适用于本地用户。

Amazon Cognito 入门

有关用户池应用程序的示例，请参见用户池入门。

有关身份池的介绍，请参阅开始使用 Amazon Cognito 身份池。

有关用户池和身份池引导式设置体验的链接，请参阅亚马逊 Cognito 的指导设置选项。

有关视频、文章、文档和更多示例应用程序，请参阅 Amazon Cognito 开发者资源。

您需要 AWS 账户才能使用 Amazon Cognito。有关更多信息，请参阅 入门 AWS。

区域可用性

亚马逊 Cognito 已在全球多个 AWS 地区上市。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 AWS 能够为包括 Amazon Cognito 在内的服务提供非常高的可用性和冗余性，同时还可以最大限度地减少延迟。

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。

身份验证如何与 Amazon Cognito 用户池和身份池配合使用

当您的客户登录 Amazon Cognito 用户池时，您的应用程序会收到 JSON 网络令牌 (JWT)。

当您的客户使用用户池令牌或其他提供商登录身份池时，您的应用程序将收到临时 AWS 证书。

通过用户池登录，您可以完全使用 AWS SDK 实现身份验证和授权。如果您不想构建自己的用户界面 (UI) 组件，则可以调用预构建的 Web UI（托管 UI）或第三方身份提供商 (IdP) 的登录页面。

本主题概述了您的应用程序与 Amazon Cognito 交互以使用 ID 令牌进行身份验证、使用访问令牌进行授权以及使用身份池 AWS 服务 凭证进行访问的一些方式。

主题

- 使用 AWS SDK 进行用户池 API 身份验证和授权
- 使用托管 UI 进行用户池身份验证
- 使用第三方身份提供商进行用户池身份验证
- 身份池认证

使用 AWS SDK 进行用户池 API 身份验证和授权

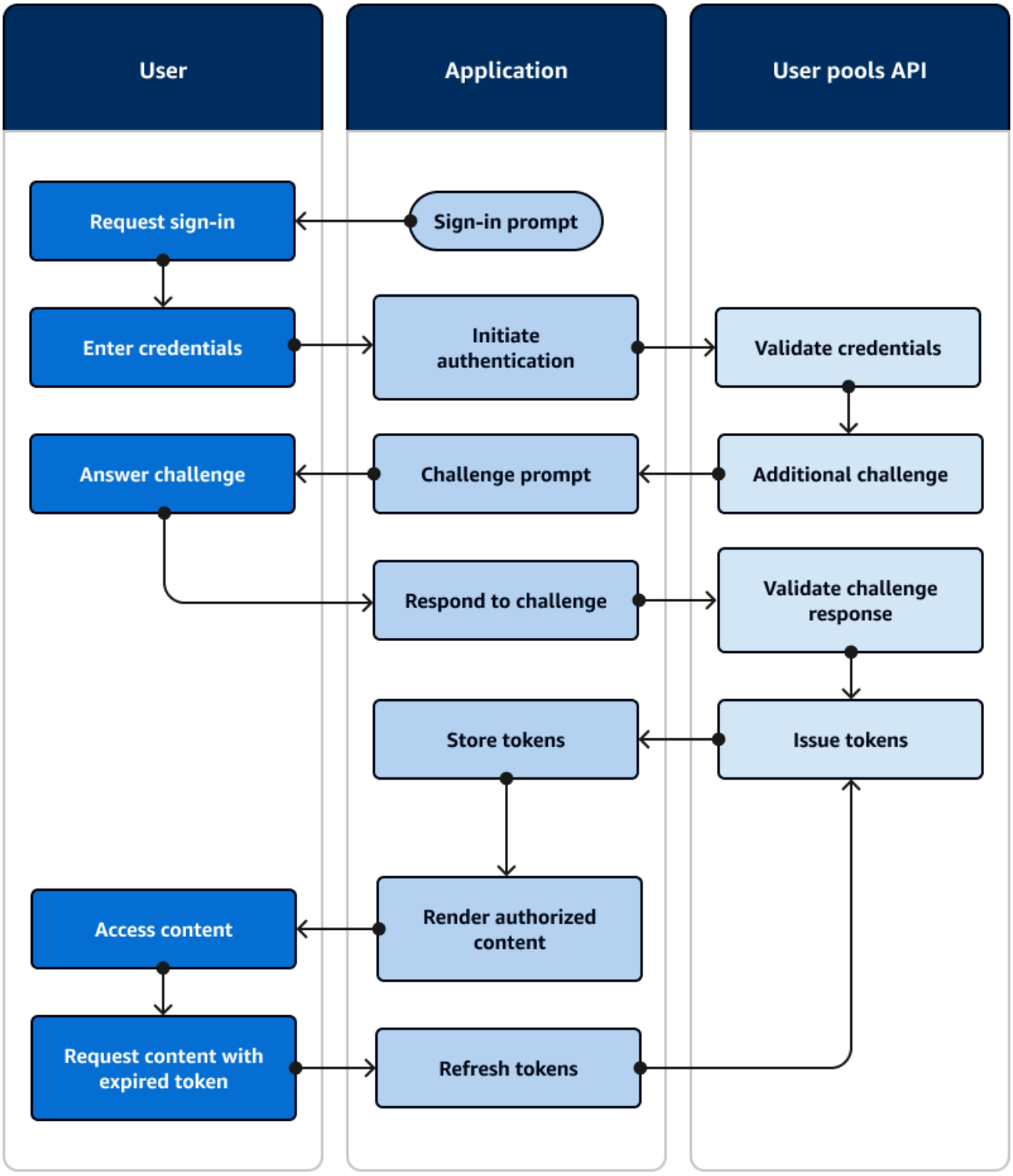
AWS 已在各种开发者框架中为 Amazon Cognito 用户池或 Amazon Cognito 身份提供商开发了组件。这些软件开发工具包中内置的方法调用 Amazon Cognito 用户池 API。同一个用户池 API 命名空间具有用于配置用户池和用户身份验证的操作。有关更全面的概述，请参阅使用 Amazon Cognito 用户池 API 和用户池端点。

API 身份验证适合您的应用程序具有现有 UI 组件且主要依赖用户池作为用户目录的模型。这种设计将 Amazon Cognito 添加为大型应用程序中的一个组件。它需要编程逻辑来处理复杂的挑战和响应链。

此应用程序不需要实现完整的 OpenID Connect (OIDC) 依赖方实现。相反，它能够解码和使用 JWT。如果您想访问本地用户的全套用户池功能，请在您的开发环境中使用 Amazon Cognito SDK 进行身份验证。

使用自定义 OAuth 范围的 API 身份验证不太倾向于外部 API 授权。要通过 API 身份验证向访问令牌添加自定义范围，请在运行时使用修改令牌令牌生成前 Lambda 触发器。

下图说明了 API 身份验证的典型登录会话。


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

API 身份验证流程

1. 用户访问您的应用程序。
2. 他们选择 “登录” 链接。
3. 他们输入用户名和密码。
4. 应用程序调用发出 InitiateAuthAPI 请求的方法。该请求会将用户的凭证传递到用户池。
5. 用户池会验证用户的凭证并确定用户是否已激活多因素身份验证 (MFA)。
6. 用户池通过请求获取 MFA 代码的质询进行响应。
7. 应用程序会生成一个提示，从用户那里收集 MFA 代码。
8. 应用程序调用发出 RespondToAuthChallengeAPI 请求的方法。该请求传递了用户的 MFA 代码。
9. 用户池验证用户的 MFA 代码。
10. 用户池使用用户的 JWT 进行响应。
11. 应用程序解码、验证、存储或缓存用户的 JWT。
12. 应用程序显示请求的访问控制组件。
13. 用户查看他们的内容。
14. 后来，用户的访问令牌已过期，他们请求查看访问控制组件。
15. 应用程序决定用户的会话应该持续下去。它使用刷新令牌再次调用该InitiateAuth方法并检索新令牌。

变体和自定义

您可以通过其他挑战来扩展此流程，例如，您自己的自定义身份验证挑战。您可以自动限制密码已被泄露，或者其意外登录特征可能表明存在恶意登录尝试的用户的访问权限。注册、更新用户属性和重置密码的操作流程大致相同。这些流程中的大多数都有重复的公共（客户端）和机密（服务器端）API 操作。

相关资源

- 亚马逊 Cognito 用户池 API
- 用户池入门
- 将 Amazon Cognito 身份验证和授权与 Web 和移动应用程序集成
- 使用 Amazon Cognito 用户池 API 和用户池端点

使用托管 UI 进行用户池身份验证

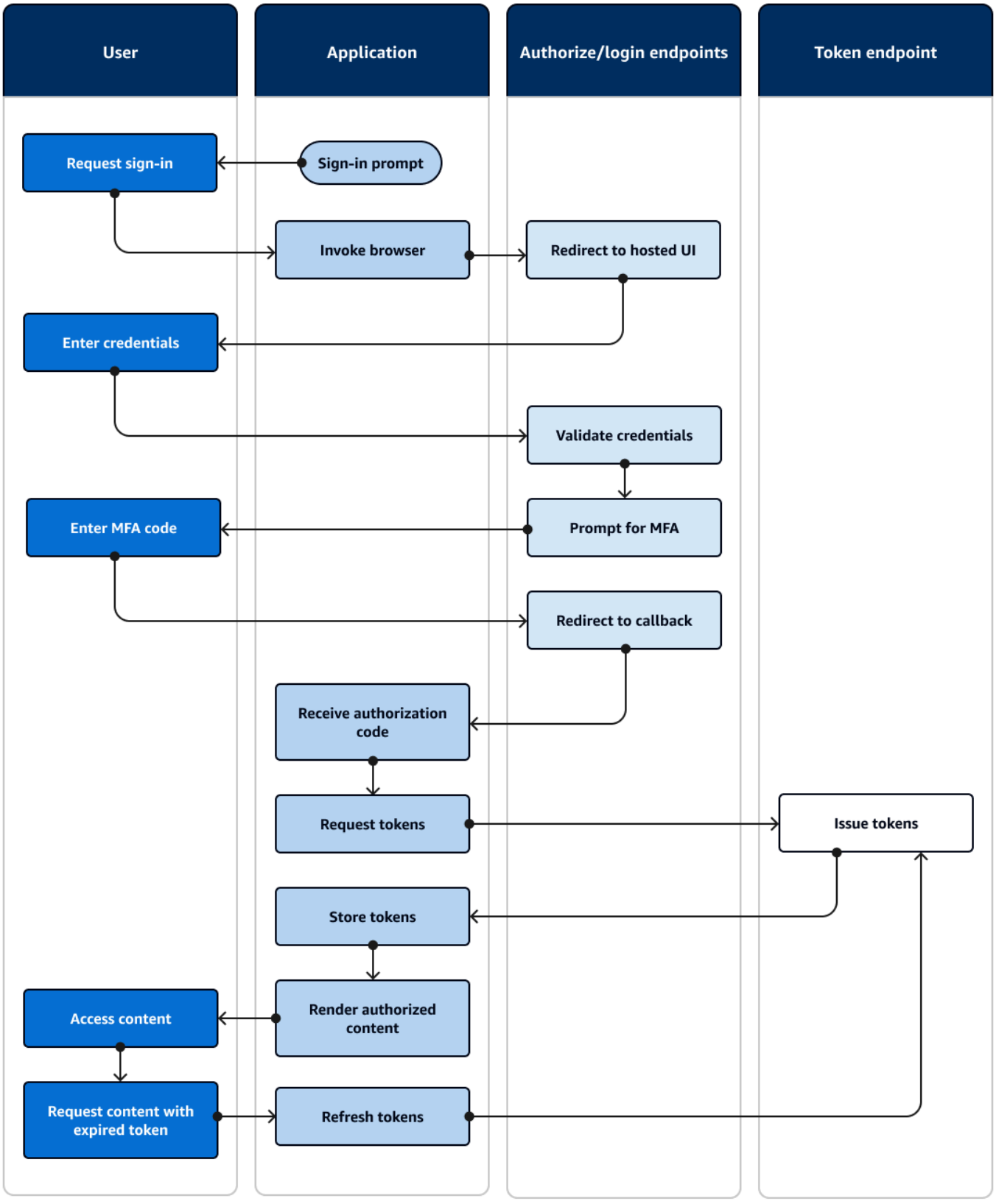
托管用户界面是一个链接到您的用户池和应用程序客户端的网站。它可以为您的用户执行登录、注册和密码重置操作。具有用于身份验证的托管用户界面组件的应用程序可能需要较少的开发人员来实现。应用程序可以跳过用户界面组件进行身份验证，并在用户的浏览器中调用托管用户界面。

应用程序使用网络或应用程序的重定向位置收集用户的 JWT。实现托管用户界面的应用程序可以连接到用户池进行身份验证，就好像它们是 OpenID Connect (OIDC) IdP 一样。

托管 UI 身份验证适合这样的模式：应用程序需要授权服务器，但不需要自定义身份验证、身份池集成或用户属性自助服务等功能。当你想使用其中一些高级选项时，你可以使用 SDK 的用户池组件来实现它们。

托管 UI 和第三方 IdP 身份验证模型主要依赖 OIDC 实现，最适合具有 OAuth 2.0 范围的高级授权模型。

下图说明了托管 UI 身份验证的典型登录会话。


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

托管用户界面身份验证流程

1. 用户访问您的应用程序。
2. 他们选择 “登录” 链接。
3. 该应用程序会将用户引导至托管界面登录提示。
4. 他们输入用户名和密码。
5. 用户池会验证用户的凭证并确定用户是否已激活多因素身份验证 (MFA)。
6. 托管用户界面会提示用户输入 MFA 代码。
7. 用户输入他们的 MFA 代码。
8. 托管用户界面将用户重定向到应用程序。
9. 应用程序从托管 UI 附加到回调 URL 的 URL 请求参数中收集授权码。
10. 应用程序请求带有授权码的令牌。
11. 令牌端点将 JWT 返回到应用程序。
12. 应用程序解码、验证、存储或缓存用户的 JWT。
13. 应用程序显示请求的访问控制组件。
14. 用户查看他们的内容。
15. 后来，用户的访问令牌已过期，他们请求查看访问控制组件。
16. 应用程序决定用户的会话应该持续下去。它使用刷新令牌从令牌端点请求新令牌。

变体和自定义

您可以在任何应用程序客户端中使用 CSS 自定义托管 UI 的外观。您还可以使用自己的身份提供商、作用域、用户属性的访问权限和高级安全配置来配置应用程序客户端。

相关资源

- 设置和使用 Amazon Cognito 托管 UI 和联合身份验证端点
- 使用托管 UI 注册和登录
- 使用资源服务器进行范围、M2M 和 API 授权
- 用户池联合身份验证端点和托管 UI 参考

使用第三方身份提供商进行用户池身份验证

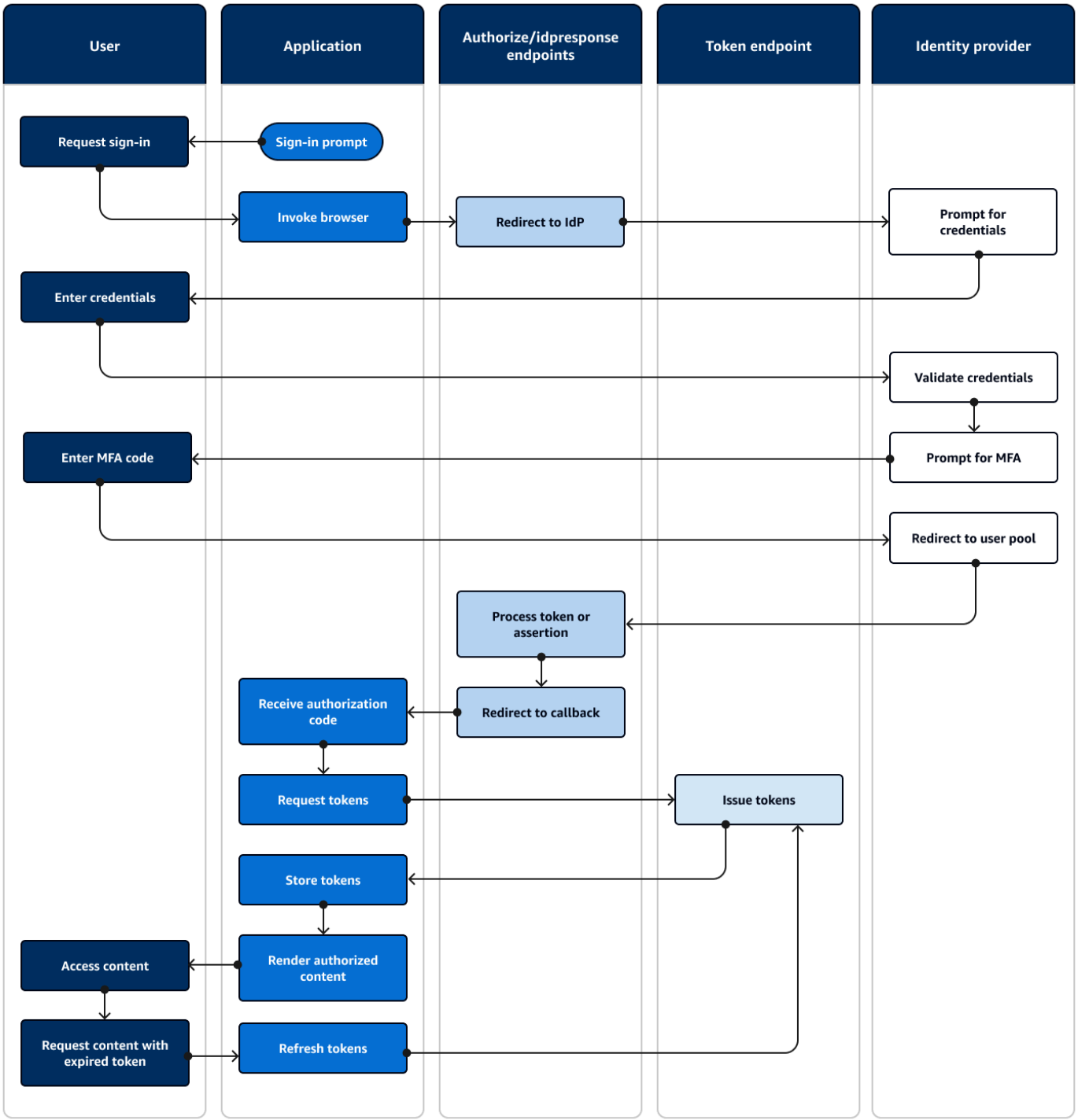
使用外部身份提供商 (IdP) 登录或联合身份验证的模式与托管 UI 类似。您的应用程序是用户池的 OIDC 依赖方，而您的用户池则充当 IdP 的直通方。IdP 可以是像 Facebook 或谷歌这样的消费者用户目录，也可以是 SAML 2.0 或 OIDC 企业目录，比如 Azure。

您的应用程序调用用户池授权服务器上的重定向端点，而不是用户浏览器中的托管用户界面。从用户的角度来看，他们在您的应用程序中选择登录按钮。然后，他们的 IdP 会提示他们登录。与托管 UI 身份验证一样，应用程序在应用程序的重定向位置收集 JWT。

使用第三方 IdP 进行身份验证适合用户在注册您的应用程序时可能不想出新密码的模式。可以毫不费力地将第三方身份验证添加到已实现托管 UI 身份验证的应用程序中。实际上，由于您在用户浏览器中调用的内容存在细微差异，托管用户界面和第三方会 IdPs 产生一致的身份验证结果。

与托管界面身份验证一样，联合身份验证最适合具有 OAuth 2.0 范围的高级授权模型。

下图说明了联合身份验证的典型登录会话。



联合身份验证流程

1. 用户访问您的应用程序。
2. 他们选择 “登录” 链接。

3. 该应用程序会引导用户使用其 IdP 进入登录提示。
4. 他们输入用户名和密码。
5. IdP 会验证用户的凭证并确定用户已激活多因素身份验证 (MFA)。
6. IdP 会提示用户输入 MFA 代码。
7. 用户输入他们的 MFA 代码。
8. IdP 使用 SAML 响应或授权码将用户重定向到用户池。
9. 如果用户传递了授权码，则用户池会默默地将该代码交换 IdP 令牌。用户池会验证 IdP 令牌，并使用新的授权码将用户重定向到应用程序。
10. 应用程序从用户池附加到回调 URL 的 URL 请求参数中收集授权码。
11. 应用程序请求带有授权码的令牌。
12. 令牌端点将 JWT 返回到应用程序。
13. 应用程序解码、验证、存储或缓存用户的 JWT。
14. 应用程序显示请求的访问控制组件。
15. 用户查看他们的内容。
16. 后来，用户的访问令牌已过期，他们请求查看访问控制组件。
17. 应用程序决定用户的会话应该持续下去。它使用刷新令牌从令牌端点请求新令牌。

变体和自定义

您可以在托管界面中启动联合身份验证，用户可以在其中从分配给应用程序客户端 IdPs 的列表中进行选择。托管用户界面还可以提示输入电子邮件地址，并自动将用户的请求路由到相应的 SAML IdP。使用第三方身份提供商进行身份验证不需要用户与托管 UI 进行交互。您的应用程序可以向用户的授权服务器请求添加请求参数，并让用户静默重定向到其 IdP 登录页面。

相关资源

- 通过第三方添加用户池登录
- 示例场景：在企业控制面板中为 Amazon Cognito 应用程序添加书签
- 使用资源服务器进行范围、M2M 和 API 授权
- 用户池联合身份验证端点和托管 UI 参考

身份池认证

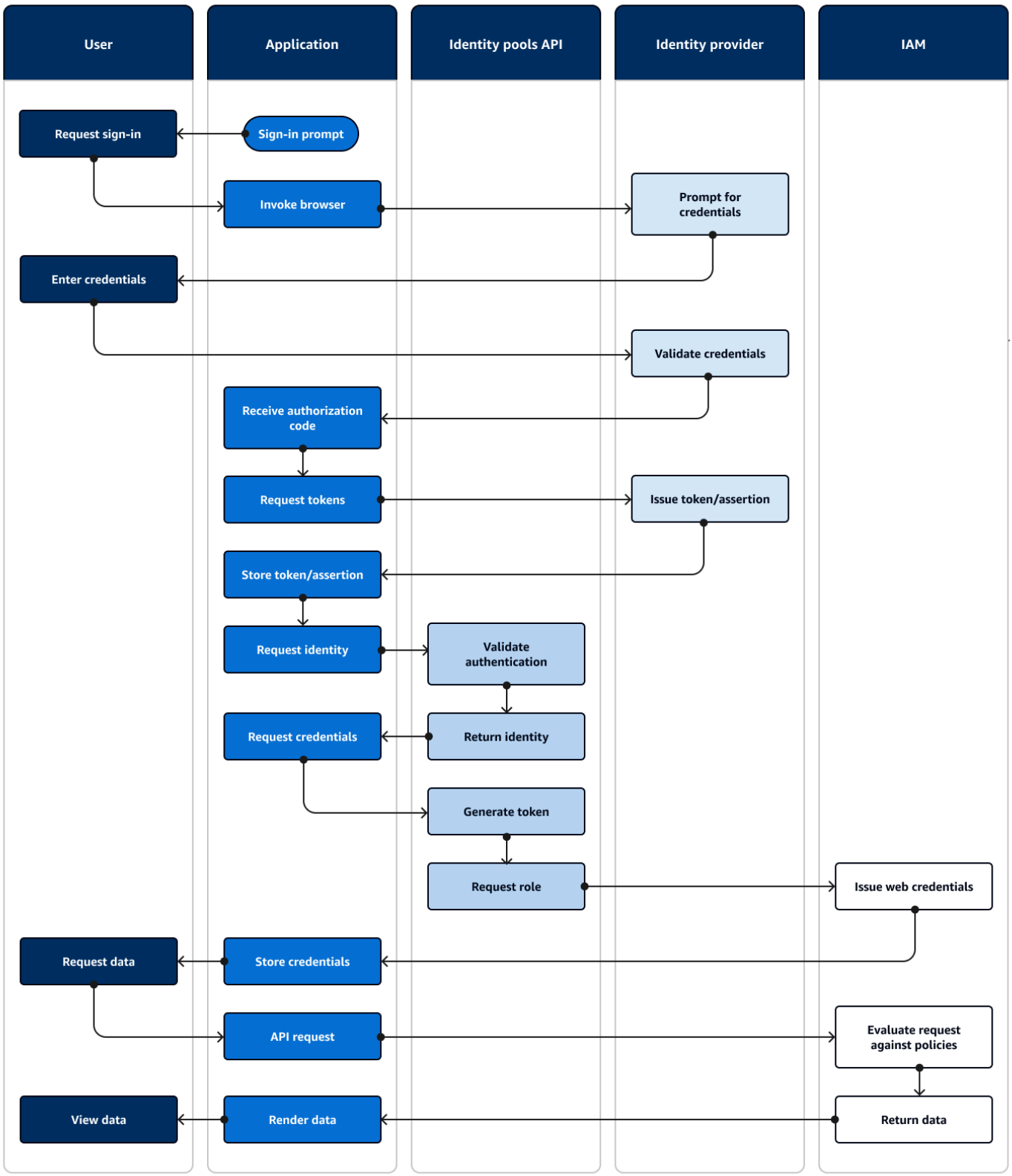
身份池是应用程序的一个组件，在函数、API 命名空间和 SDK 模型上与用户池截然不同。用户池提供基于令牌的身份验证和授权，而身份池则为 AWS Identity and Access Management (IAM) 提供授权。

您可以将一组身份池分配 IdPs 给身份池，并使用这些身份池登录用户。用户池作为身份池紧密集成 IdPs ，为身份池提供了最多的访问控制选项。同时，身份池有多种身份验证选项可供选择。用户池加入 SAML、OIDC、社交、开发者和访客身份源，作为从身份池中获得临时 AWS 证书的路由。

身份池的身份验证是外部的，它遵循前面说明的用户池流程之一，或者您与另一个 IdP 独立开发的流程。在您的应用程序执行初始身份验证后，它会将证明传递给身份池并接收临时会话作为回报。

使用身份池进行身份验证适合您使用 IAM 授权对应用程序资产和数据实施访问控制 AWS 服务 的模式。与用户池中的 API 身份验证一样，成功的应用程序包括您要访问的每项服务的 AWS SDK，以使用户受益。 AWS SDK 将身份池身份验证中的凭据作为签名应用于 API 请求。

下图说明了使用 IdP 进行身份池身份验证的典型登录会话。


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

联合身份验证流程

1. 用户访问您的应用程序。
2. 他们选择 “登录” 链接。
3. 该应用程序会引导用户使用其 IdP 进入登录提示。
4. 他们输入用户名和密码。
5. IdP 会验证用户的证书。
6. IdP 使用 SAML 响应或授权码将用户重定向到应用程序。
7. 如果用户传递了授权码，则应用程序将使用该代码兑换 IdP 令牌。
8. 应用程序解码、验证、存储或缓存用户的 JWT 或断言。
9. 应用程序调用发出 GetIdAPI 请求的方法。它传递用户的令牌或断言并请求身份 ID。
10. 身份池根据配置的身份提供者验证令牌或断言。
11. 身份池返回一个身份 ID。
12. 应用程序调用发出 GetCredentialsForIdentityAPI 请求的方法。它传递用户的令牌或断言并请求一个 IAM 角色。
13. 身份池会生成一个新的 JWT。新的 JWT 包含请求 IAM 角色的声明。身份池根据用户的请求和 IdP 的身份池配置中的角色选择标准来确定角色。
14. AWS Security Token Service (AWS STS) 响应来自身份池的AssumeRoleWithWebIdentity请求。该响应包含带有 IAM 角色的临时会话的 API 证书。
15. 应用程序存储会话凭证。
16. 用户在应用程序中执行的操作需要访问受保护的资源。 AWS
17. 应用程序将临时证书作为签名应用于所需的 API 请求 AWS 服务。
18. IAM 会评估证书中附加到该角色的策略。它将它们与请求进行比较。
19. AWS 服务 返回请求的数据。
20. 应用程序在用户界面中呈现数据。
21. 用户查看数据。

变体和自定义

要使用用户池可视化身份验证，请在问题令牌/断言步骤之后插入之前的用户池概述之一。开发者身份验证将请求身份之前的所有步骤替换为由开发者凭证签名的请求。访客身份验证还会直接跳到请求身份，不验证身份验证，并返回访问受限的 IAM 角色的证书。

相关资源

- Amazon Cognito 身份池
- 用户 IAM 角色
- 身份池概念
- 身份池（联合身份）身份验证流程

亚马逊 Cognito 条款

Amazon Cognito 为网络和移动应用程序提供凭证。它借鉴并建立在身份和访问管理中常见的术语之上。有许多通用身份和访问条款指南可供选择。部分示例包括：

- IDPro 知识体系中的@@ 术语
- AWS 身份服务
- 来自 NIST CSRC 的@@ 词汇表

以下列表描述了亚马逊 Cognito 独有的术语或在亚马逊 Cognito 中具有特定上下文的术语。

主题

- 常规
- 用户池
- 身份池

常规

此列表中的术语并不是 Amazon Cognito 所特有的，在身份和访问管理从业者中得到了广泛认可。以下不是术语的详尽列表，而是本指南中有关其特定 Amazon Cognito 上下文的指南。

App

通常是移动应用程序。在本指南中，应用程序通常是连接到 Amazon Cognito 的 Web 应用程序或移动应用程序的简写。

基于属性的访问控制 (ABAC)

一种模型，在这种模型中，应用程序根据用户的属性（例如其职称或部门）来确定对资源的访问权限。Amazon Cognito 强制执行 ABAC 的工具包括用户池中的 ID 令牌和身份池中的委托人标签。

授权服务器

生成 JSON 网络令牌的基于 Web 的系统。Amazon Cognito 用户池联合终端节点是用户池中两种身份验证和授权方法的授权服务器组件。另一种方法是用户池 API。

机密应用程序、服务器端应用程序

用户远程连接的应用程序，其代码位于应用程序服务器上，并且可以访问机密。这通常是一个 Web 应用程序。

Identity provider (IdP) (身份提供商 (IdP))

一种存储和验证用户身份的服务。Amazon Cognito 可以向外部提供商请求身份验证，并成为应用程序的 IdP。

JSON 网络令牌 (JWT)

一个 JSON 格式的文档，其中包含有关经过身份验证的用户的声明。ID 令牌对用户进行身份验证，访问令牌授权用户，刷新令牌更新凭证。Amazon Cognito从外部提供商那里接收令牌，并向应用程序发放令牌，或者. AWS STS

多重身份验证 (MFA)

要求用户在提供用户名和密码后提供额外的身份验证。Amazon Cognito 用户池为本地用户提供了 MFA 功能。

OAuth 2.0（社交）提供商

提供 JWT 访问和刷新令牌的用户池或身份池的 Id P。在用户进行身份验证后，Amazon Cognito 用户池会自动与社交提供商进行互动。

OpenID Connect (OIDC) 提供商

用户池或身份池的 IdP，用于扩展 OAuth 规范以提供 ID 令牌。在用户进行身份验证后，Amazon Cognito 用户池会自动与 OIDC 提供商进行交互。

公共应用程序

一种在设备上自包含的应用程序，其代码存储在本地，并且无法访问密钥。这通常是一个移动应用程序。

资源服务器

具有访问控制功能的 API。Amazon Cognito 用户池还使用资源服务器来描述用于定义与 API 交互的配置的组件。

基于角色的访问控制 (RBAC)

一种根据用户的职能名称授予访问权限的模型。Amazon Cognito 身份池通过区分 IAM 角色来实现 RBAC。

服务提供商 (SP)、依赖方 (RP)

依赖 IdP 来断言用户值得信赖的应用程序。Amazon Cognito 充当外部服务提供商 IdPs，对基于应用程序的 SP 充当 IdP。

samlProvid

用户池或身份池的 IdP，用于生成经过数字签名的断言文档，您的用户会将其传递给 Amazon Cognito。

通用唯一标识符 (UUID)

应用于对象的 128 位标签。Amazon Cognito UUID 在每个用户池或身份池中都是唯一的。

用户目录

向其他系统提供该信息的用户及其属性的集合。Amazon Cognito 用户池是用户目录，也是用于合并来自外部用户目录的用户的工具。

用户池

当您在本指南中看到以下列表中的术语时，它们指的是用户池的特定功能或配置。

亚马逊 Cognito 用户池 API

一组身份验证和授权 API 操作，您可以使用 AWS SDK 将其添加到您的应用程序中。API 可以登录本地用户和关联用户。

自适应身份验证

一种高级安全功能，可检测潜在的恶意活动并为用户配置文件应用额外的安全保护。

高级安全功能

一个可选组件，用于添加用户安全工具。

应用程序客户端

一种组件，用于将用户池的设置定义为一个应用程序的 IdP。

回调网址，重定向 URI

应用程序客户端中的设置和对用户池联合终端节点的请求中的参数。回传 URL 是您的应用程序中经过身份验证的用户的初始目的地。

已泄露的凭证

一种高级安全功能，可检测攻击者可能知道的用户密码，并对用户配置文件应用额外的安全保护。

确认

确定是否满足了允许新用户登录的先决条件的过程。确认通常通过电子邮件地址或电话号码验证来完成。

自定义身份验证

使用 Lambda 触发器扩展身份验证流程，用于定义其他用户质询和响应。

设备认证

一种身份验证过程，将 MFA 替换为使用可信设备的 ID 的登录。

外部提供商、第三方提供商

与用户池有信任关系的 IdP。

联合用户

用户池中由外部提供商进行身份验证的用户。

联邦终端节点

用户池网域上的一组网页，用于托管与之交互的 IdPs 服务和应用程序。

托管 UI

用户池域上的一组交互式网页，用于托管用户身份验证服务。

Lambda 触发器

其中一种函数 AWS Lambda ，用户池可以在用户身份验证过程的关键时刻自动调用。您可以使用 Lambda 触发器自定义身份验证结果。

本地用户

用户池用户目录中的用户配置文件，不是通过向外部提供商进行身份验证而创建的。

关联用户

来自外部提供商的用户，其身份与本地用户合并。

代币自定义

令牌生成前 Lambda 触发器的结果，该触发器在运行时修改用户的 ID 或访问令牌。

用户池、亚马逊 Cognito 身份提供商**cognito-idp**、Amazon Cognito 用户池

一种 AWS 资源，为使用 OID IdPs C 的应用程序提供身份验证和授权服务。

用户池域

您添加到用户池中的网站名称。该域是托管 UI 和联合终端节点的基本 URL。

验证

确认用户拥有电子邮件地址或电话号码的过程。用户池向输入了新电子邮件地址或电话号码的用户发送验证码。当他们向 Amazon Cognito 提交代码时，他们将验证自己对消息目标的所有权，并且可以从用户池中接收其他消息。另请参阅确认。

用户个人资料、用户账户

用户目录中用户的条目。所有用户的用户池中都有个人资料。

身份池

当您在本指南中看到以下列表中的术语时，它们指的是身份池的特定功能或配置。

访问控制属性

在身份池中实现基于属性的访问控制。身份池将用户属性作为标签应用于用户证书。

基本（经典）身份验证

一种身份验证过程，您可以在其中自定义对用户凭证的请求。

已经过开发人员验证的身份

使用开发者凭证授权身份池用户凭证的身份池用户凭证的身份验证过程。

开发者证书

身份池管理员的 IAM API 密钥。

增强的身份验证

一种身份验证流程，它根据您在身份池中定义的逻辑选择 IAM 角色并应用委托人标签。

求同

一个 UUID，用于将应用程序用户及其用户凭据链接到与身份池存在信任关系的外部用户目录中的个人资料。

身份池、亚马逊 Cognito 联合身份、亚马逊 Cognito 身份、**cognito-identity**

一种 AWS 资源，为使用临时 AWS 证书的应用程序提供身份验证和授权服务。

未经验证的 身份

尚未使用身份池 IdP 登录的用户。您可以允许用户在进行身份验证之前为单个 IAM 角色生成有限的用户证书。

用户凭证

用户在身份池身份验证后收到的临时 AWS API 密钥。

将此服务与 AWS SDK 配合使用

AWS 软件开发套件 (SDK) 可用于许多流行的编程语言。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。

SDK 文档	代码示例
AWS SDK for C++	AWS SDK for C++ 代码示例
AWS CLI	AWS CLI 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 Tools for PowerShell	PowerShell 代码示例工具

SDK 文档	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
适用于 SAP ABAP 的 AWS SDK	适用于 SAP ABAP 的 AWS SDK 代码示例
AWS SDK for Swift	AWS SDK for Swift 代码示例

示例可用性

找不到所需的内容？ 通过使用此页面底部的提供反馈链接请求代码示例。

入门 AWS

在开始使用 Amazon Cognito 之前，请为自己设置一些必需 AWS 的资源。

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

报名参加 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，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。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，请为用户分配管理访问权限，并且只使用根用户来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/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。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. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 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网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

用户池入门

您可以使用本节中的指南来创建您的初始用户池资源。要进行 step-by-step 演练，请从 React JavaScript 开发者环境中的基本 Web 应用程序开始。然后，您可以继续添加诸如托管用户界面（托管用户界面）和与外部社交或 SAML 2.0 身份提供商的联合登录（IdPs）之类的功能。

在您努力扩展功能集并整合 Amazon Cognito 的更多组件时，请阅读 A mazon Cognito 用户池章节，了解您可以对用户池进行的所有操作的完整描述。

本节中的示例用户池和应用程序演示了应用程序资源与 Amazon Cognito 用户池的基本集成。稍后，您可以调整用户池以使用更多可用的选项。然后，您可以更新您的应用程序以采用新的 API 并与托管的 UI 进行交互，以及 IdPs。

本节中的教程使用 SDK 创建具有自定义 UI 和基于 API 的身份 AWS 验证的应用程序。以这种方式构建的应用程序非常适合对本地用户进行身份验证。要从具有预构建用户界面、自动处理某些用户池功能以及对联合用户进行身份验证的应用程序开始，请跳至添加带有托管 UI 的应用程序客户端。

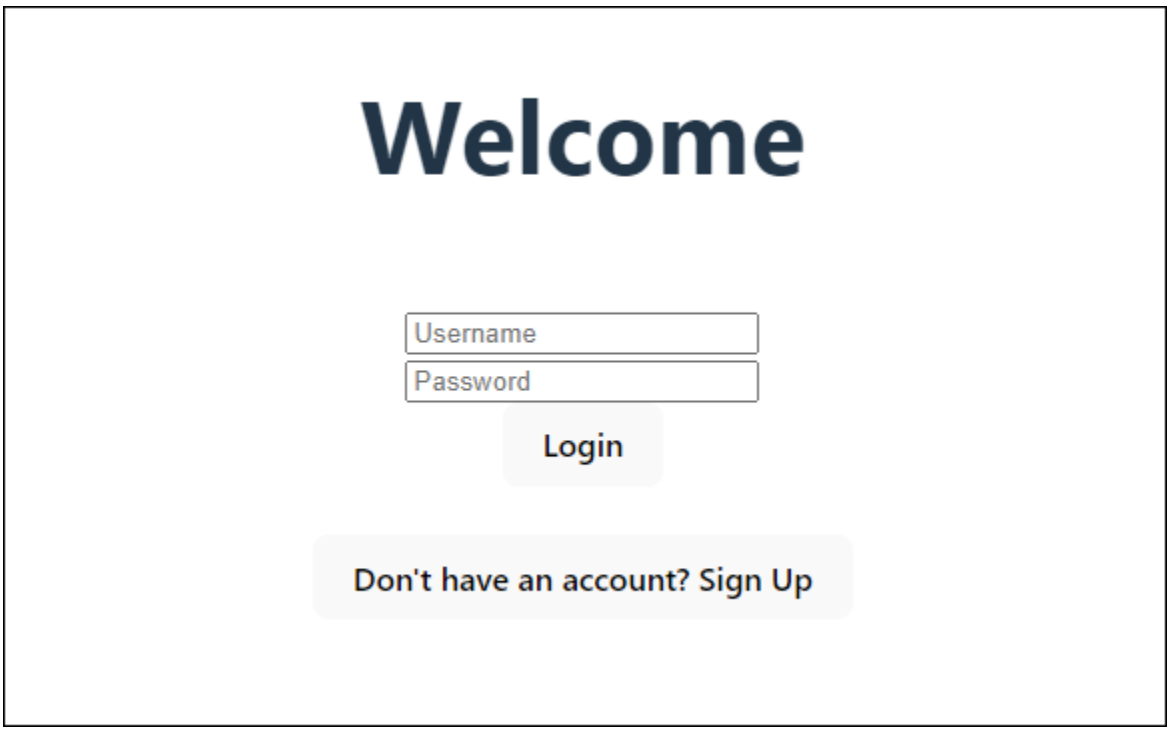
主题

- 设置一个 React 单页应用程序示例
- 使用 Flutter 设置安卓应用示例
- 后续步骤

设置一个 React 单页应用程序示例

在本教程中，您将创建一个 React 单页应用程序，您可以在其中测试用户注册、确认和登录。React 是一个 JavaScript基于 Web 和移动应用程序的库，侧重于用户界面 (UI)。此示例应用程序演示了 Amazon Cognito 用户池的一些基本功能。如果您已经有使用 React 开发 Web 应用程序的经验，请从中下载示例应用程序 GitHub。

以下屏幕截图显示了您要创建的应用程序中的初始身份验证页面。



创建用户池过程为您设置了一个与示例应用程序配合使用的用户池。如果您的用户池满足以下要求，则可以跳过此步骤：

- 用户可以使用自己的电子邮件地址登录。Cognito 用户池登录选项：电子邮件。
- 用户名不区分大小写。用户名要求：未选择 “将用户名区分大小写”。
- 不需要多因素身份验证 (MFA)。MFA 执法：可选 M FA。
- 您的用户池通过电子邮件验证用于确认用户配置文件的属性。要验证的属性：发送电子邮件、验证电子邮件地址。
- 电子邮件是唯一的必填属性。必填属性：电子邮件。
- 用户可以在您的用户池中自行注册。自行注册：选中 “启用自助注册”。
- 您的初始应用程序客户端是允许使用用户名和密码登录的公共客户端。应用程序类型：公共客户端，身份验证流程：ALLOW_USER_PASSWORD_AUTH。

创建用户池

创建新的用户池

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 AWS 凭据。
2. 选择 “创建用户池” 按钮。您可能需要从左侧导航窗格中选择 “用户池” 才能显示此选项。

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“配置登录体验”中，您可以选择要用于此用户池的身份提供商 (IdPs)。有关更多信息，请参阅通过第三方添加用户池登录。

   a. 在“身份验证提供程序”下，对于提供者类型，请确保仅选择 Cognito 用户池。

   b. 对于 Cognito 用户池登录选项，请选择用户名。不要选择任何其他用户名要求。

   c. 将所有其他选项保留为默认值，然后选择“下一步”。
5. 在配置安全要求中，您可以选择密码策略、多因素身份验证 (MFA) 要求和用户帐户恢复选项。有关更多信息，请参阅 使用 Amazon Cognito 用户池安全功能。

   a. 对于密码策略，请确认密码策略模式已设置为 Cognito 默认值。

   b. 在多重身份验证下，对于强制执行 MFA，请选择可选 MF A。

   c. 对于 MFA 方法，请选择身份验证器应用程序和短信。

   d. 要恢复用户帐户，请确认已选中“启用自助服务帐户恢复”，并将用户帐户恢复消息传送方式设置为“仅限电子邮件”。

   e. 将所有其他选项保留为默认值，然后选择“下一步”。
6. 在“配置注册体验”中，您可以确定新用户在注册为新用户时将如何验证其身份，以及在用户注册流程中哪些属性应为必填属性或可选属性。有关更多信息，请参阅 管理用户池中的用户。

   a. 确认已选中“启用自助注册”。此设置会打开您的用户池，让互联网上的任何人都可以注册。这是为示例应用程序而设计的，但在生产环境中应谨慎应用此设置。

   b. 在 Cognito 辅助验证和确认下，确认已选中“允许 Cognito 自动发送消息进行验证和确认”复选框。

   c. 确认“要验证的属性”设置为“发送电子邮件，验证电子邮件地址”。

   d. 在“验证属性更改”下，确认已选择默认选项：选中“待更新时保留原始属性值”，“待更新时处于活动状态”属性值设置为“电子邮件地址”。

   e. 在“必填属性”下，确认基于先前选择的必填属性显示电子邮件。

> **Important**
>
> 对于此示例应用程序，您的用户池不得将 phone_number 设置为必填属性。如果 phone_n umber 显示为必填属性，请查看并更新您之前的选择：
>
> - 可选 MFA，仅限电子邮件用于用户账户恢复消息的传送方式
> - 发送电子邮件，验证要验证的属性的电子邮件地址

f. 将所有其他选项保留为默认值，然后选择 “下一步”。

7. 在配置消息传送中，您可以配置与亚马逊简单电子邮件服务和亚马逊简单通知服务的集成，向您的用户发送电子邮件和短信，用于注册、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 用户池的短信设置。

   a. 对于电子邮件提供商，选择使用 Cognito 发送电子邮件，然后使用 Amazon Cognito 提供的默认电子邮件发件人。此低电子邮件量设置足以进行应用程序测试。在使用亚马逊简单电子邮件服务 (Amazon SES) 验证电子邮件地址并选择 “使用亚马逊 SES 发送电子邮件” 后，您可以返回。

   b. 对于 SMS，请选择创建新的 IAM 角色并输入 I AM 角色名称。这将创建一个向 Amazon Cognito 授予发送短信权限的角色。

   c. 将所有其他选项保留为默认值，然后选择 “下一步”。

8. 在集成您的应用程序中，您可以命名用户池、配置托管用户界面和创建应用程序客户端。有关更多信息，请参阅 添加带有托管 UI 的应用程序客户端。示例应用程序不使用托管 UI。

   a. 在用户池名称下，输入用户池名称。

   b. 不要选择 “使用 Cognito 托管的用户界面”。

   c. 在 “初始应用程序客户端” 下，确认应用程序类型已设置为 “公共客户端”。

   d. 在 “客户密钥” 下，确认已选中 “不生成客户机密钥”。

   e. 输入应用程序客户端名称。

   f. 展开高级应用程序客户端设置。`ALLOW_USER_PASSWORD_AUTH`添加到身份验证流程列表中。

   g. 将所有其他选项保留为默认值，然后选择 “下一步”。

9. 在 “查看并创建” 屏幕中查看您的选择，并根据需要修改任何选择。如果您对用户池配置感到满意，请选择创建用户池以继续。

10. 在用户池页面中，选择您的新用户池。

11. 在 “用户池概述” 下，记下您的用户池 ID。在创建示例应用程序时，您将提供此字符串。

12. 选择 “应用程序集成” 选项卡，然后找到 “应用程序客户端和分析” 部分。选择您的新应用程序客户端。记下您的客户端 ID。

相关资源

- Amazon Cognito 用户群体
- 用户池身份验证流程

- 将令牌与用户池结合使用

创建应用程序

要构建此应用程序，必须设置开发人员环境。开发人员环境要求是：

1. Node.js 已安装并更新。
2. 节点包管理器 (npm) 已安装并更新至至少版本 10.2.3。
3. 可以在网络浏览器中通过 TCP 端口 5173 访问该环境。

创建示例 React Web 应用程序

1. 登录您的开发人员环境并导航到应用程序的父目录。

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

2. 创建一个新的 React 服务。

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

3. 从上的 AWS 代码示例存储库中克隆cognito-developer-guide-react-example项目文件夹 GitHub。

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

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

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

4. 导航到项目中的src目录。

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

5. 编辑config.ts并替换以下值：

   a. YOUR_AWS_REGION用 AWS 区域 代码替换。例如：us-east-1。

   b. YOUR_COGNITO_USER_POOL_ID替换为您指定用于测试的用户池的 ID。例如：us-east-1_EXAMPLE。用户池必须位于您在 AWS 区域 上一步中输入的。
   c. YOUR_COGNITO_APP_CLIENT_ID替换为您指定用于测试的应用程序客户端的 ID。例如：1example23456789。应用程序客户端必须位于上一步中的用户池中。
6. 如果要从以外的 IP 访问示例应用程序localhost，请编辑该行package.json并将其更改"dev": "vite",为"dev": "vite --host 0.0.0.0",。
7. 安装您的应用程序。

```
npm install
```

8. 启动应用程序。

```
npm run dev
```

9. 在 Web 浏览器中访问该应用程序，网址为http://localhost:5173或http://[IP address]:5173。
10. 使用有效的电子邮件地址注册新用户。
11. 从您的电子邮件中检索确认码。在应用程序中输入确认码。
12. 使用您的用户名和密码登录。

使用 Amazon Lightsail 创建 React 开发者环境

开始使用此应用程序的一种快速方法是使用 Amazon Lightsail 创建虚拟云服务器。

借助 Lightsail，您可以快速创建一个小型服务器实例，该实例已预先配置了此示例应用程序的先决条件。您可以使用基于浏览器的客户端通过 SSH 连接到您的实例，并通过公有或私有 IP 地址连接到 Web 服务器。

为此示例应用程序创建 Lightsail 实例

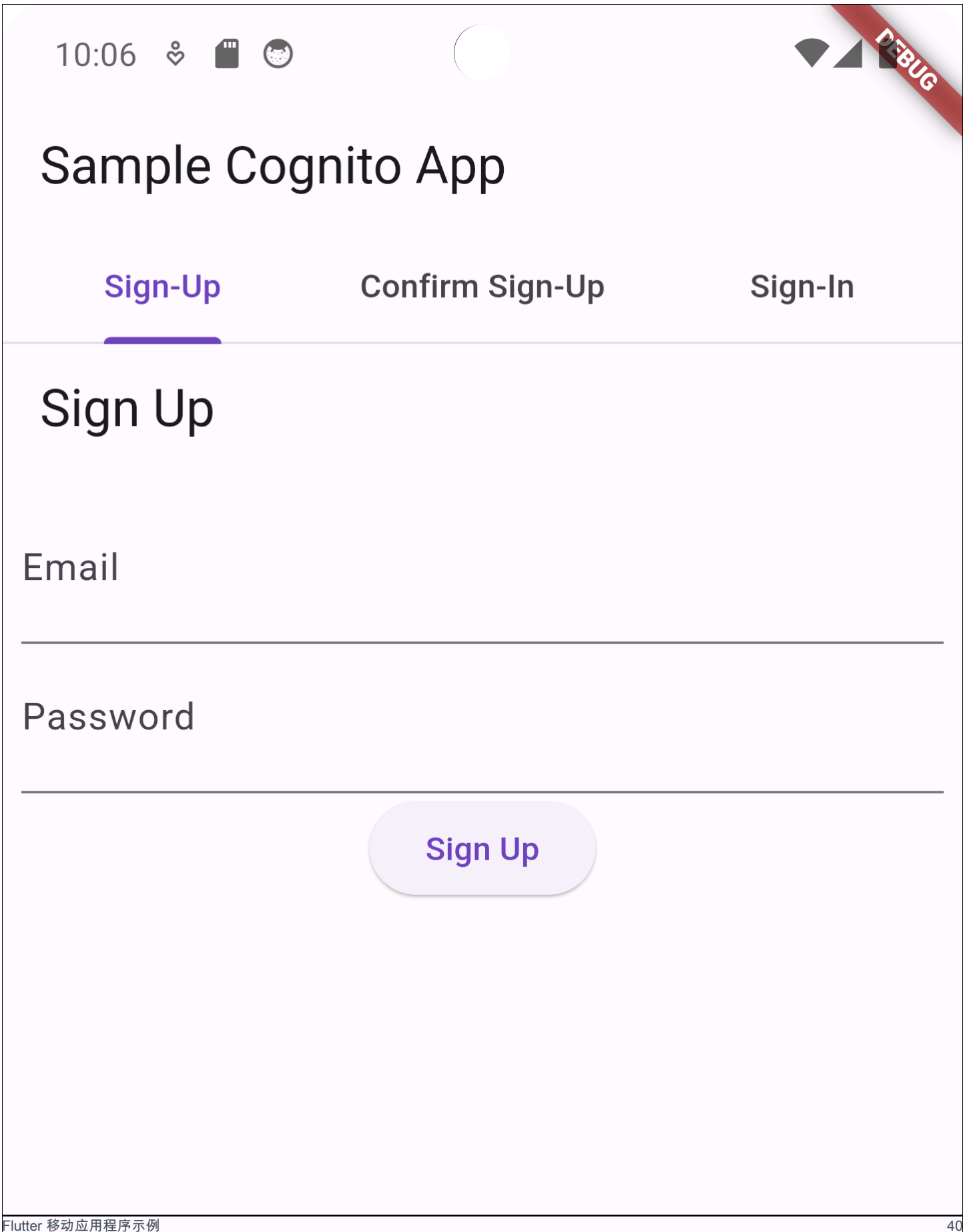
1. 前往 Lightsail 控制台。如果出现提示，请输入您的 AWS 凭据。
2. 选择创建实例。
3. 在 “选择平台” 中，选择 Linux/Unix。
4. 在 “选择蓝图” 中，选择 Node.js。
5. 在 “识别您的实例” 下，为您的开发环境指定一个友好的名称。
6. 选择创建实例。

7. Lightsail 创建您的实例后，将其选中，然后从 “连接” 选项卡中选择 “使用 SSH 连接”。
8. SSH 会话将在浏览器窗口中打开。运行`node -v`和`npm -v`并确认您的实例已配置了 Node.js 且最低的 npm 版本为 10.2.3。
9. 继续配置你的 React 应用程序。

使用 Flutter 设置安卓应用示例

在本教程中，您将在 Android Studio 中创建一个移动应用程序，您可以在其中模拟设备并测试用户的注册、确认和登录。此示例应用程序在 Flutter 中为安卓系统创建了一个基本的 Amazon Cognito 用户池移动客户端。如果您已经有使用 Flutter 开发移动应用程序的经验，请从 GitHub中下载示例应用程序。

以下屏幕截图显示了在虚拟 Android 设备上运行的应用程序。


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

创建用户池过程为您设置了一个与示例应用程序配合使用的用户池。如果您的用户池满足以下要求，则可以跳过此步骤：

- 用户可以使用自己的电子邮件地址登录。Cognito 用户池登录选项：电子邮件。
- 用户名不区分大小写。用户名要求：未选择 “将用户名区分大小写”。
- 不需要多因素身份验证 (MFA)。MFA 执法：可选 M FA。
- 您的用户池通过电子邮件验证用于确认用户配置文件的属性。要验证的属性：发送电子邮件、验证电子邮件地址。
- 电子邮件是唯一的必填属性。必填属性：电子邮件。
- 用户可以在您的用户池中自行注册。自行注册：选中 “启用自助注册”。
- 您的初始应用程序客户端是允许使用用户名和密码登录的公共客户端。应用程序类型：公共客户端，身份验证流程：ALLOW_USER_PASSWORD_AUTH。

创建用户池

创建新的用户池

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 AWS 凭据。
2. 选择 “创建用户池” 按钮。您可能需要从左侧导航窗格中选择 “用户池” 才能显示此选项。
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 “配置登录体验” 中，您可以选择要用于此用户池的身份提供商 (IdPs)。有关更多信息，请参阅通过第三方添加用户池登录。

   a. 在 “身份验证提供程序” 下，对于提供者类型，请确保仅选择 Cognito 用户池。

   b. 对于 Cognito 用户池登录选项，请选择用户名。不要选择任何其他用户名要求。

   c. 将所有其他选项保留为默认值，然后选择 “下一步”。
5. 在配置安全要求中，您可以选择密码策略、多因素身份验证 (MFA) 要求和用户帐户恢复选项。有关更多信息，请参阅 使用 Amazon Cognito 用户池安全功能。

   a. 对于密码策略，请确认密码策略模式已设置为 Cognito 默认值。

   b. 在多重身份验证下，对于强制执行 MFA，请选择可选 MF A。

   c. 对于 MFA 方法，请选择身份验证器应用程序和短信。

   d. 要恢复用户帐户，请确认已选中 “启用自助服务帐户恢复”，并将用户帐户恢复消息传送方式设置为 “仅限电子邮件”。

e. 将所有其他选项保留为默认值，然后选择 “下一步”。

6. 在 “配置注册体验” 中，您可以确定新用户在注册为新用户时将如何验证其身份，以及在用户注册流程中哪些属性应为必填属性或可选属性。有关更多信息，请参阅 管理用户池中的用户。

   a. 确认已选中 “启用自助注册”。此设置会打开您的用户池，让互联网上的任何人都可以注册。这是为示例应用程序而设计的，但在生产环境中应谨慎应用此设置。

   b. 在 Cognito 辅助验证和确认下，确认已选中 “允许 Cognito 自动发送消息进行验证和确认” 复选框。

   c. 确认 “要验证的属性” 设置为 “发送电子邮件，验证电子邮件地址”。

   d. 在 “验证属性更改” 下，确认已选择默认选项：选中 “待更新时保留原始属性值”，“待更新时处于活动状态” 属性值设置为 “电子邮件地址”。

   e. 在 “必填属性” 下，确认基于先前选择的必填属性显示电子邮件。

   > **Important**
   >
   > 对于此示例应用程序，您的用户池不得将 phone_number 设置为必填属性。如果 phone_n umber 显示为必填属性，请查看并更新您之前的选择：
   >
   > - 可选 MFA，仅限电子邮件用于用户账户恢复消息的传送方式
   > - 发送电子邮件，验证要验证的属性的电子邮件地址

   f. 将所有其他选项保留为默认值，然后选择 “下一步”。

7. 在配置消息传送中，您可以配置与亚马逊简单电子邮件服务和亚马逊简单通知服务的集成，向您的用户发送电子邮件和短信，用于注册、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 用户池的短信设置。

   a. 对于电子邮件提供商，选择使用 Cognito 发送电子邮件，然后使用 Amazon Cognito 提供的默认电子邮件发件人。此低电子邮件量设置足以进行应用程序测试。在使用亚马逊简单电子邮件服务 (Amazon SES) 验证电子邮件地址并选择 “使用亚马逊 SES 发送电子邮件” 后，您可以返回。

   b. 对于 SMS，请选择创建新的 IAM 角色并输入 I AM 角色名称。这将创建一个向 Amazon Cognito 授予发送短信权限的角色。

   c. 将所有其他选项保留为默认值，然后选择 “下一步”。

8. 在集成您的应用程序中，您可以命名用户池、配置托管用户界面和创建应用程序客户端。有关更多信息，请参阅 添加带有托管 UI 的应用程序客户端。示例应用程序不使用托管 UI。

a. 在用户池名称下，输入用户池名称。
b. 不要选择 “使用 Cognito 托管的用户界面”。
c. 在 “初始应用程序客户端” 下，确认应用程序类型已设置为 “公共客户端”。
d. 在 “客户密钥” 下，确认已选中 “不生成客户机密钥”。
e. 输入应用程序客户端名称。
f. 展开高级应用程序客户端设置。ALLOW_USER_PASSWORD_AUTH添加到身份验证流程列表中。
g. 将所有其他选项保留为默认值，然后选择 “下一步”。

9. 在 “查看并创建” 屏幕中查看您的选择，并根据需要修改任何选择。如果您对用户池配置感到满意，请选择创建用户池以继续。
10. 在用户池页面中，选择您的新用户池。
11. 在 “用户池概述” 下，记下您的用户池 ID。在创建示例应用程序时，您将提供此字符串。
12. 选择 “应用程序集成” 选项卡，然后找到 “应用程序客户端和分析” 部分。选择您的新应用程序客户端。记下您的客户端 ID。

相关资源

- Amazon Cognito 用户群体
- 用户池身份验证流程
- 将令牌与用户池结合使用

创建应用程序

创建安卓应用示例

1. 安装安卓工作室和命令行工具。
2. 在安卓工作室中，安装 Flutter 插件。
3. 在此示例应用程序中，根据cognito_flutter_mobile_app目录内容创建一个新的 Android Studio 项目。
    - << YOUR CLIENT ID>>使用您之前创建的用户池和应用程序客户端的 ID 编辑assets/config.json并替换<<YOUR USER POOL ID>>和。
4. 安装 Flutter。

a. 将 Flutter 添加到你的 PATH 变量中。

b. 使用以下命令接受许可证。

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

c. 验证您的 Flutter 环境并安装所有缺失的组件。

```
flutter doctor
```

- 如果缺少任何组件，`flutter doctor -v`请运行以了解如何修复问题。

d. 切换到新 Flutter 项目的目录并安装依赖项。

- 运行 `flutter pub add amazon_cognito_identity_dart_2`。

e. 运行 `flutter pub add flutter_secure_storage`。

5. 创建一台虚拟的安卓设备。

1. 在 Android Studio 用户界面中，使用设备管理器创建新设备。
2. 在 CLI 中运行`flutter emulators --create --name android-device`。

6. 启动您的虚拟安卓设备。

1. 在 Android Studio 用户界面中，选择虚拟设备旁边的开始图标。
2. 在 CLI 中运行`flutter emulators --launch android-device`。

7. 在虚拟设备上启动您的应用程序。

1. 在 Android Studio 用户界面中，选择部署图标。
2. 在 CLI 中运行`flutter run`。

8. 在 Android Studio 中导航到你正在运行的虚拟设备。
9. 使用有效的电子邮件地址注册新用户。
10. 从您的电子邮件中检索确认码。在应用程序中输入确认码。
11. 使用您的用户名和密码登录。

后续步骤

按照教程完成示例应用程序后，可以扩大用户池实现的范围。您可以创建其他用户池，为其他应用程序自定义用户池功能，或者添加外部身份提供商。在计划将 Amazon Cognito 用户池放入生产应用程序时，您可以评估其他示例和教程。

以下是 Amazon Cognito 用户池的一些其他功能：

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

有关 Amazon Cognito 身份验证和授权模型的概述，请参阅。身份验证如何与 Amazon Cognito 用户池和身份池配合使用

要在成功进行用户池身份验证 AWS 服务 后访问其他，请参阅登录后 AWS 服务 使用身份池进行访问。

除了使用 AWS Management Console 和用户池 SDK 之外，您还可以使用来管理您的用户池。AWS Command Line Interface

主题

- 创建新的用户池
- 添加带有托管 UI 的应用程序客户端
- 向用户池添加社交登录（可选）
- 将使用 SAML 身份提供商的登录添加到用户池（可选）

创建新的用户池

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。

创建新的用户池

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 AWS 凭据。
2. 选择 “创建用户池” 按钮。您可能需要从左侧导航窗格中选择 “用户池” 才能显示此选项。

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“配置登录体验”中，您可以选择要用于此用户池的身份提供商 (IdPs)。有关更多信息，请参阅通过第三方添加用户池登录。

   a. 在“身份验证提供程序”下，对于提供者类型，请确保仅选择 Cognito 用户池。

   b. 对于 Cognito 用户池登录选项，请选择用户名。不要选择任何其他用户名要求。

   c. 将所有其他选项保留为默认值，然后选择“下一步”。
5. 在配置安全要求中，您可以选择密码策略、多因素身份验证 (MFA) 要求和用户帐户恢复选项。有关更多信息，请参阅 使用 Amazon Cognito 用户池安全功能。

   a. 对于密码策略，请确认密码策略模式已设置为 Cognito 默认值。

   b. 在多重身份验证下，对于强制执行 MFA，请选择可选 MF A。

   c. 对于 MFA 方法，请选择身份验证器应用程序和短信。

   d. 要恢复用户帐户，请确认已选中“启用自助服务帐户恢复”，并将用户帐户恢复消息传送方式设置为“仅限电子邮件”。

   e. 将所有其他选项保留为默认值，然后选择“下一步”。
6. 在“配置注册体验”中，您可以确定新用户在注册为新用户时将如何验证其身份，以及在用户注册流程中哪些属性应为必填属性或可选属性。有关更多信息，请参阅 管理用户池中的用户。

   a. 确认已选中“启用自助注册”。此设置会打开您的用户池，让互联网上的任何人都可以注册。这是为示例应用程序而设计的，但在生产环境中应谨慎应用此设置。

   b. 在 Cognito 辅助验证和确认下，确认已选中“允许 Cognito 自动发送消息进行验证和确认”复选框。

   c. 确认“要验证的属性”设置为“发送电子邮件，验证电子邮件地址”。

   d. 在“验证属性更改”下，确认已选择默认选项：选中“待更新时保留原始属性值”，“待更新时处于活动状态”属性值设置为“电子邮件地址”。

   e. 在“必填属性”下，确认基于先前选择的必填属性显示电子邮件。

> **Important**
>
> 对于此示例应用程序，您的用户池不得将 phone_number 设置为必填属性。如果 phone_n umber 显示为必填属性，请查看并更新您之前的选择：
>
> - 可选 MFA，仅限电子邮件用于用户账户恢复消息的传送方式
> - 发送电子邮件，验证要验证的属性的电子邮件地址

f. 将所有其他选项保留为默认值，然后选择 “下一步”。

7. 在配置消息传送中，您可以配置与亚马逊简单电子邮件服务和亚马逊简单通知服务的集成，向您的用户发送电子邮件和短信，用于注册、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 用户池的短信设置。

   a. 对于电子邮件提供商，选择使用 Cognito 发送电子邮件，然后使用 Amazon Cognito 提供的默认电子邮件发件人。此低电子邮件量设置足以进行应用程序测试。在使用亚马逊简单电子邮件服务 (Amazon SES) 验证电子邮件地址并选择 “使用亚马逊 SES 发送电子邮件” 后，您可以返回。

   b. 对于 SMS，请选择创建新的 IAM 角色并输入 I AM 角色名称。这将创建一个向 Amazon Cognito 授予发送短信权限的角色。

   c. 将所有其他选项保留为默认值，然后选择 “下一步”。

8. 在集成您的应用程序中，您可以命名用户池、配置托管用户界面和创建应用程序客户端。有关更多信息，请参阅 添加带有托管 UI 的应用程序客户端。示例应用程序不使用托管 UI。

   a. 在用户池名称下，输入用户池名称。

   b. 不要选择 “使用 Cognito 托管的用户界面”。

   c. 在 “初始应用程序客户端” 下，确认应用程序类型已设置为 “公共客户端”。

   d. 在 “客户密钥” 下，确认已选中 “不生成客户机密钥”。

   e. 输入应用程序客户端名称。

   f. 展开高级应用程序客户端设置。`ALLOW_USER_PASSWORD_AUTH`添加到身份验证流程列表中。

   g. 将所有其他选项保留为默认值，然后选择 “下一步”。

9. 在 “查看并创建” 屏幕中查看您的选择，并根据需要修改任何选择。如果您对用户池配置感到满意，请选择创建用户池以继续。

10. 在用户池页面中，选择您的新用户池。

11. 在 “用户池概述” 下，记下您的用户池 ID。在创建示例应用程序时，您将提供此字符串。

12. 选择 “应用程序集成” 选项卡，然后找到 “应用程序客户端和分析” 部分。选择您的新应用程序客户端。记下您的客户端 ID。

创建用户池

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 AWS 凭据。

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 使用 Amazon Cognito 用户池安全功能。
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。

   > **Important**
   >
   > 如果您在用户群体中激活用户注册，则互联网上的任何人都可以注册账户并登录您的应用程序。除非您开放您的应用程序供公开注册，否则不要在用户群体中启用自助注册。要更改此设置，请在用户池控制台的 “注册体验” 选项卡中更新自助服务注册，或者更新 CreateUserPool或 UpdateUserPoolAPI AllowAdminCreateUserOnly请求中的值。有关可以在用户群体中设置的安全功能的信息，请参阅使用 Amazon Cognito 用户池安全功能。

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 用户池的短信设置。
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 添加带有托管 UI 的应用程序客户端。
9. 在 “查看并创建” 屏幕中查看您的选择，并根据需要修改任何选择。如果您对用户池配置感到满意，请选择创建用户池以继续。

相关资源

有关用户池的更多信息，请参阅Amazon Cognito 用户群体。

另请参阅：用户池身份验证流程和将令牌与用户池结合使用。

添加带有托管 UI 的应用程序客户端

创建用户池后，您可以为应用程序创建应用程序客户端，以打开托管 UI 的内置网页。在托管用户界面中，用户可以：

- 注册获取用户个人资料。
- 使用第三方身份提供商登录。
- 使用或不使用多重身份验证登录。
- 重置他们的密码。

要为托管界面创建应用程序客户端，请登录

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 AWS 凭据。
2. 选择用户池。
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. 选择域旁边的操作，然后选创建自定义域或创建 Amazon Cognito 域。如果您已配置用户群体域，请先选择删除 Amazon Cognito 域或删除自定义域，然后再创建新的自定义域。
6. 输入可用的域前缀，以与 Amazon 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. 选择您的应用程序允许的身份验证流程。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。在能够在应用程序中实现注销之前，您无需添加允许的注销网址。

    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 Amazon Cognito 用户群体作为提供商。

> **Note**
>
> 要使用外部身份提供商 (IdPs)（例如 Facebook、亚马逊、谷歌和苹果）以及通过 OpenID Connect (OIDC) 或 SAML 登录 IdPs，请先按照添加通过第三方登录的用户池中所示对其进行配置。然后返回应用程序客户端设置页面将其启用。

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. 选择您已配置的任何自定义范围。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

在您的应用程序客户端页面上，选择查看托管用户界面，打开一个新的浏览器选项卡，进入预先填充了应用程序客户端 ID、范围、授权和回传网址参数的登录页面。

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。您可以使用解码和验证用户池令牌。AWS Lambda有关更多信息，请参阅网站上的 Amazon Cognito JWT 代币解码和验证。 AWS GitHub

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。

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。

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）。

   如果你没有现有的Facebook应用程序，你会看到一个不同的选项。选择 Create App (创建应用程序)。
4. 在创建应用程序页面上，为您的应用程序选择一个用例，然后选择下一步。
5. 输入 Facebook 应用程序的名称，然后选择创建应用程序。
6. 在左侧导航栏上，选择应用程序设置，然后选择基本。
7. 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。
8. 从页面底部选择 + 添加平台。
9. 在 “选择平台” 屏幕上，选择您的平台，然后选择 “下一步”。
10. 选择 Save changes（保存更改）。
11. 为 App Domains（应用程序域）输入用户池域。

    ```
    https://your_user_pool_domain
    ```
12. 选择 Save changes（保存更改）。
13. 从导航栏中选择 “产品”，然后选择 “通过 Facebook 登录进行配置”。
14. 从 Facebook 登录的配置菜单中，选择设置。

    在 Valid OAuth Redirect URIs（有效的 OAuth 重定向 URI）中输入重定向 URL。重定向 URL 由您的用户池域和`/oauth2/idpresponse`终端节点组成。

    ```
    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 “应用程序和服务”，然后选择 Lo gin 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默认项目。
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 API 和服务，然后选择 Oauth 同意屏幕。
7. 输入应用程序信息、应用程序域、授权域名和开发者联系信息。您的授权域名必须包含您的自定义域名`amazoncognito.com`和根域。例如：`example.com`。选择 SAVE AND CONTINUE（保存并继续）。
8. 1. 在 “范围” 下，选择 “添加或移除范围”，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“保存并继续”。
10. 再次展开左侧导航栏，选择 API 和服务，然后选择凭据。
11. 选择“创建凭据”，然后选择 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. 在授权 JavaScript 来源下，选择添加 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 在您的客户端 ID 和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“选择类型”页面上，选择“应用程序”，然后选择“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（说明）下，输入说明。
    2. 在 Identifier（标识符）下，输入标识符。请记下此服务 ID，因为在中选择 Apple 作为身份提供商后，您将需要此值步骤 2：将社交 IdP 添加到用户池。
    3. 选择 Continue (继续)，然后选择 Register (注册)。
13. 从 “标识符” 页面选择您刚刚创建的服务 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 “下一步”，然后选择 “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. 在配置密钥页面上，选择您之前创建的应用程序 ID 作为主应用程序 ID。选择保存。
    4. 选择 Continue（继续），然后选择 Register（注册）。
17. 在“下载您的密钥”页面上，选择“下载”以下载私钥，记下显示的密钥 ID，然后选择“完成”。在步骤 2：将社交 IdP 添加到用户池中选择 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。

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。

使用配置用户池社交身份提供商 AWS Management Console

1. 转到 Amazon Cognito 控制台。系统可能会提示您输入 AWS 凭证。
2. 选择用户池。
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 L ogin with Amazon — 输入在上一节中生成的应用程序客户端 ID 和应用程序客户端密钥。
   - Facebook — 输入在上一节中生成的应用程序客户端 ID 和应用程序客户端密钥，然后选择 API 版本（例如，版本 2.12）。我们建议选择可能的最新版本——每个 Facebook API 都有生命周期和弃用日期。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。
   - 使用 Apple 登录 — 输入在上一节中生成的服务 ID、团队 ID、密钥 ID 和私钥。
7. 输入要使用的授权范围的名称。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：

- 最终用户在离开 Apple 的登录页面后遇到故障（这些故障可能源于 Amazon Cognito 中的内部故障或开发者编写的任何内容）。
- 服务 ID 标识符用于用户池和/或其他身份验证服务。
- 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。
- 开发者删除了该用户，然后用户重新登录，而无需从其 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 有关映射的需知信息。
9. 选择创建。
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）。

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://mydomain.us-east-1.amazoncognito.com/login?
response_type=code&client_id=1example23456789&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。

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。

有关更多信息，请参阅 在用户池中使用 SAML 身份提供商。

您必须更新您的 SAML 身份提供商并配置您的用户池。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。

您还必须向 SAML 身份提供商提供断言消费者服务 (ACS) 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。有关用户池域的更多信息，请参阅配置用户池域。

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

您可以在 Amazon Cognito 控制台的域名选项卡上找到您的域名前缀和用户池的区域值。

对于某些 SAML 身份提供商，您还需要提供服务提供商 (SP)urn，也称为受众 URI 或 SP 实体 ID，格式为：

```
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。这使您的应用程序无需检索或解析 SAML 断言响应，因为用户池通过用户代理直接接收来自您的身份提供商的 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，请输入您的 AWS 凭据。
2. 选择用户池。
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）。标识符告诉 Amazon Cognito 它应该检查用户登录时输入的电子邮件地址。然后，它会将他们定向到与其域名相对应的提供商。
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。`saml2/logout`端点使用 POST 绑定。

   > **Note**
   >
   > 如果选择了此选项，并且您的 SAML 身份提供商需要签名的注销请求，则您还需要配置 Amazon Cognito 随您的 SAML IdP 提供的签名证书。
   > SAML IdP 将处理已签名的注销请求，并将您的用户从 Amazon Cognito 会话中注销。

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**
>
> 如果您的提供商有公共端点，我们建议您输入元数据文档 URL，而不是上传文件。这允许 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. 选择创建。

开始使用 Amazon Cognito 身份池

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

使用身份池，您可以获得临时 AWS 证书，这些证书具有您定义的权限，以便直接访问其他资源 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

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。

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 角色中已有要使用的角色，则可以选择使用现有 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。

5. 在 Connect 身份提供商中，输入您在配置身份池信任中选择的身份提供商 (IdPs) 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。

      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。启用基本流程后，您可以绕过为自己选择的角色 IdPs ，AssumeRoleWithWebIdentity直接致电。有关更多信息，请参阅 身份池（联合身份）身份验证流程。

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。

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。

- 在《AWS SDK for Java 开发者指南》 JavaScript中设置 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），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 用户群体和登录后 AWS 服务 使用身份池进行访问。

要了解有关使用外部身份提供商的更多信息，请参阅 身份池外部身份提供商。

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）。

获取凭证

Amazon Cognito 身份池为访客用户（未经身份验证）和已通过身份验证并收到令牌的用户提供临时 AWS 证书。有了这些 AWS 证书，您的应用程序就可以 AWS 通过 Amazon API Gateway 安全地访问内部 AWS 或外部的后端。请参阅 获取凭证。

亚马逊 Cognito 的指导设置选项

您可能需要在结构化的指导性体验中评估 Amazon Cognito 的功能。以下是一些外部资源，可通过用户池和身份池提供量身定制的体验。

完成一场研讨会

AWS 研讨会工作室举办研讨会，引导你完成大部分 Amazon Cognito 功能的设置。这些功能包括用户池 API、用户池托管 UI、身份池和安全配置。

从示例中添加应用程序代码

本指南中的代码示例章节包含可用于用户池和身份池的应用程序代码。代码示例章节的用户池部分包含涵盖各个操作的简短片段和较长的示例， end-to-end 例如使用各种编程语言的应用程序。

使用创建全栈应用程序 AWS Amplify

AWS Amplify AWS 服务 适用于想要开发和托管应用程序和用户界面的开发人员。亚马逊 Cognito 是 Amplify 的身份验证组件。当您在应用程序中添加身份验证时，Amplify 可以自动部署 Amazon Cognito 用户池和身份池资源。另请参阅将 Amazon Cognito 身份验证和授权与 Web 和移动应用程序集成。

更多亚马逊 Cognito 应用程序资源请访问 GitHub

- 适用于亚马逊 Cognito 的.NET 身份验证流程示例
- 亚马逊 Cognito 无密码身份验证
- PetStore带有 Amazon 验证权限的示例
- 使用 ABAC + 身份池访问 AWS 资源的 React 应用程序示例
- 使用 CDK 基于 Amazon Cognito 和 API Gateway 的机器到机器授权 AWS
- 使用 Amazon Cognito、API Gateway 和 IAM 构建精细授权
- CloudFront授权 @edge

更多研讨会

- 使用 Amazon Cognito 实现无密码身份验证和 WebAuthn
- 使用 Amazon Cognito 用户池实现多租户 SaaS 身份
- 亚马逊 Cognito JWT 深度探索

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

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。我们建议您使用AWS Amplify将 Amazon Cognito 与您的网络和移动应用程序集成。AWS Amplify 是一个完整的解决方案，可让前端 Web 和移动开发人员轻松构建、连接和托管全栈应用程序 AWS，并且可以 AWS 服务 随着用例的发展灵活利用其广度。Amplify Auth 主要利用 Amazon Cognito 来构建身份验证功能。

主题

- 使用进行身份验证 AWS Amplify
- 使用 AWS 开发工具包进行身份验证
- 使用 Amazon Verified Permissions 进行授权

Amazon Cognito 的典型实现使用可视化工具和 API 的组合。Amazon Cognito 控制台是用于设置和管理 Amazon Cognito 用户群体和身份池的可视界面。托管用户界面是一个 ready-to-use 基于 Web 的登录应用程序，用于快速测试和部署 Amazon Cognito 用户池。此外，在大多数 Amazon Cognito 部署中，您必须在应用程序中添加代码才能与用户群体和身份池进行交互。例如，应用程序可能会调用托管 UI 进行用户登录，然后从应用程序代码中调用令牌端点，以便将用户的授权码交换为令牌。然后，应用程序必须解释和存储用户的令牌，并在适当的上下文中出示它们以进行身份验证和授权。Amplify 为这些流程添加了带有内置功能的引导式集成工具。

您也可以完全在代码中构建您的 Amazon Cognito 资源。要开始使用自己的自定义应用程序代码，请访问 AWS SDK 的 Amazon Cognito 代码示例。要将 Amazon Cognito 作为 OpenID Connect 身份提供商进行集成，请使用 OpenID Connect 开发工具。

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。有关可用的平台，请参阅使用 AWS 开发工具包进行身份验证。 AWS CLI 这是一款适用于亚马逊 Cognito AWS 服务等的命令行软件开发工具包，是开始熟悉 Amazon Cognito API 的好去处。

> **Note**
>
> Amazon Cognito 的某些组件只能使用 API 进行配置。例如，您只能使用更新或 UpdateUserPoolAPI 请求中UserPool类`LambdaConfig`属性的请求来设置用户池自定义 SMS 或电子邮件发件人 Lambda 触发器。CreateUserPool

Amazon Cognito 用户群体 API 与多个 API 操作类共享其命名空间。一个类配置用户群体及其进程、身份提供商和用户。另一个类包括用户在公共客户端中执行的未经身份验证的操作，旨在登录、注销和管理他们的配置文件。最后一类 API 操作在机密的服务器端客户端中执行您使用自己的 AWS 凭据授权的用户操作。在开始实现应用程序代码之前，您必须知道您想要的应用程序架构。有关更多信息，请参阅使用 Amazon Cognito 用户池 API 和用户池端点。

使用进行身份验证 AWS Amplify

AWS Amplify 是用于构建 Web 和移动应用程序的完整解决方案。借助 Amplify，您可以使用 Amplify 库连接到现有资源，也可以使用 Amplify 命令行界面（CLI）创建和配置新资源。Amplify 还连接了用户界面组件（如身份验证器），以便在您的应用程序中设置和自定义登录和注册体验。

要在前端应用程序中使用 Amplify 身份验证功能，请参阅以下按平台分列的文档。

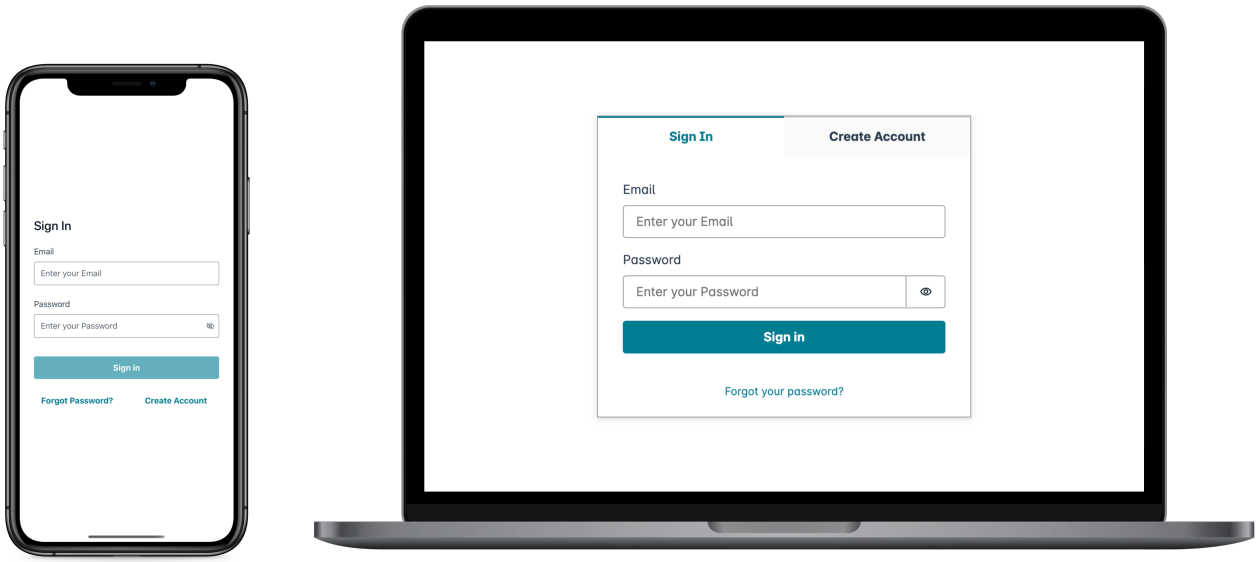
- Amplify 身份验证用于 JavaScript
- 适用于 iOS 的 Amplify 身份验证
- 适用于 Android 的 Amplify 身份验证
- 放大 Flutter 的身份验证

Amplify 库是开源的，可在上使用。GitHub要了解有关 Amplify Auth 如何实现 Amazon Cognito 身份验证的更多信息，请访问以下库。

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

使用 Amplify 创建用户界面（UI）

Amazon Cognito 用户群体托管 UI 可以满足 Web 或移动应用程序的身份验证前端的基本需要。要在托管 UI 可容纳的参数之外自定义您的用户界面（UI），请创建自定义应用程序。Amplify UI 是各种语言的前端组件的可自定义集合。



要开始使用您的自定义身份验证组件，请访问身份验证器组件的以下文档。

- 适用于 Android 的身份验证器
- 适用于 Angular 的身份验证器
- 适用于 Flutter 的身份验证器
- 适用于 React 的身份验证器
- 适用于 React Native 的身份验证器
- 适用于 Swift 的身份验证器
- 适用于 Vue 的身份验证器

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 交互，请使用您选择的语言版本的软件开发工具包连接到 Amazon Cognito 用户池和 Amazon Cognito 身份池 API。 AWS

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

- AWS 适用于 .NET 的 SDK
- AWS 适用于 C++ 的 SDK
- AWS 适用于 Go 的 SDK
- AWS 适用于 Java 的 SDK V2
- AWS 适用于 JavaScript
- AWS 适用于 PHP 的 SDK V3
- AWS Python 软件开发工具包
- AWS 适用于 Ruby V3 的 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或 BatchIsAuthorizedWithTokenAPI 请求中向已验证的权限提供用户的身份IsAuthorizedWithToken或访问令牌。这些 API 操作接受您的用户作为`Principal`并对他们想要访问`Resource`的用户做出授权决定。`Action`其他自定义`Context`可以帮助做出详细的访问决策。

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. 您的令牌中的`token_use`索赔值与您传递给的参数相匹配`IsAuthorizedWithToken`。`token_use`声明必须是`access`您将其传递给`accessToken`参数以及`id`是否将其传递给`identityToken`参数。
5. 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。`Allow`如果您的请求与允许的操作匹配且与显式`Deny`操作不匹配，则验证权限会对您的请求做出响应；否则，它会响应`Deny`。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、添加和删除您要向已验证权限提供的用户声明，请使用自定义访问和身份令牌中的内容令牌生成前 Lambda 触发器。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。

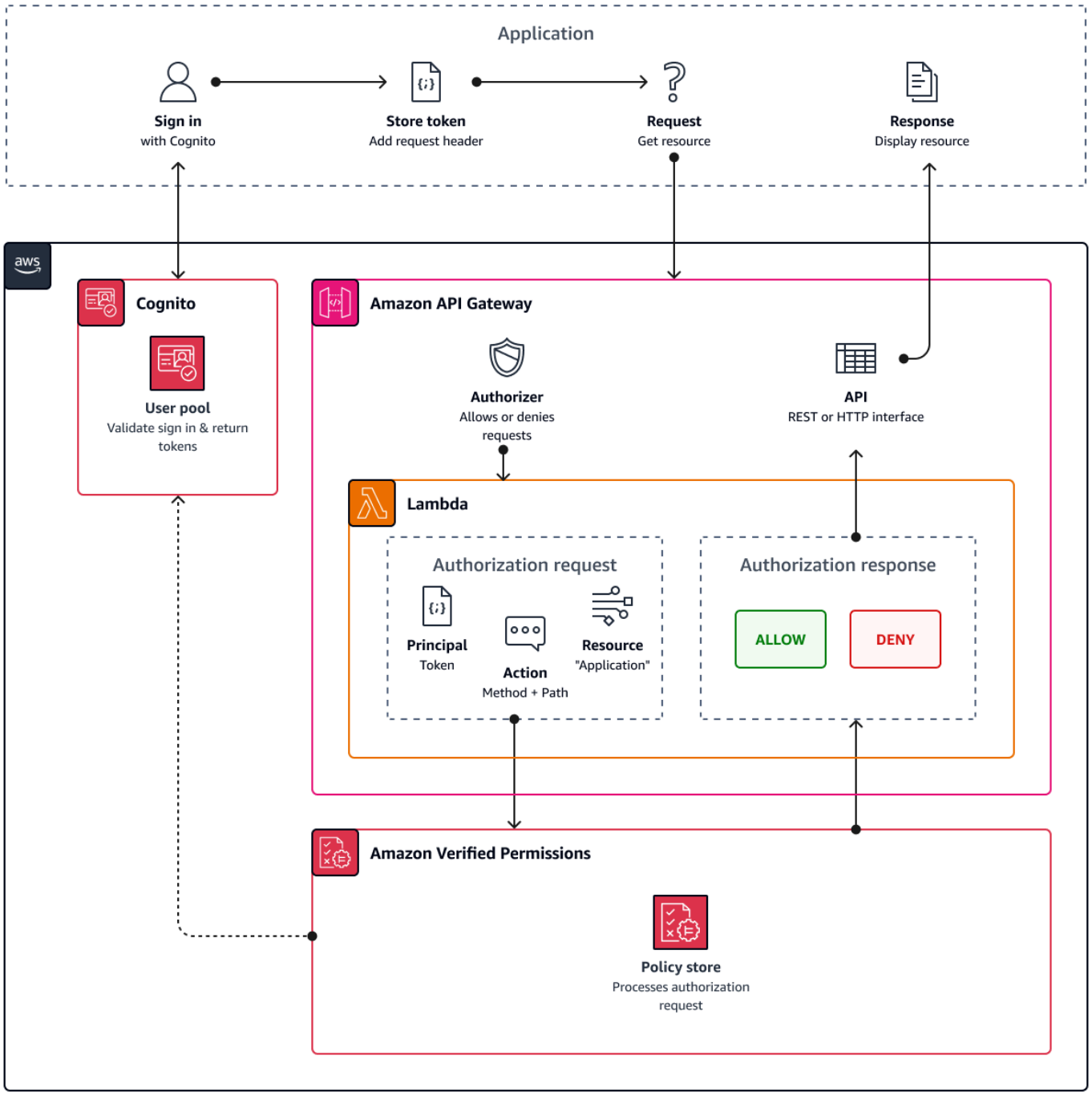
> **Note**
>
> 由于 Verified Permissions 处理声明的方式，请勿在令牌生成前函数中添加名为 `cognito`、`dev` 或 `custom` 的声明。如果您提供的这些保留的声明前缀不是采用以冒号分隔的格式（例如 `cognito:username`），而是采用完整的声明名称，则您的授权请求会失败。

有关 Verified Permissions 如何将 Amazon Cognito 令牌中的声明映射到授权策略的更多信息，请参阅将 Amazon Cognito 令牌映射到 Verified Permissions 架构。

使用已验证权限的 API 授权

您的身份证或访问令牌可以授权向后端 Amazon API Gateway REST API 发出的请求，且权限经过验证。您可以创建一个策略存储库，其中包含指向您的用户池和 API 的即时链接。通过 “使用 Co gnito 和 API Gateway 进行设置” 启动选项，“已验证权限” 会将用户池身份源添加到策略存储中，并向 API 添加一个 Lambda 授权方。当您的应用程序将用户池持有者令牌传递给 API 时，Lambda 授权方会调用已验证的权限。授权方将令牌作为委托人传递，将请求路径和方法作为操作传递。

下图说明了具有已验证权限的 API Gateway API 的授权流程。有关详细明细，请参阅《Amazon 验证权限用户指南》中与 API 关联的策略存储。



经过验证的权限围绕用户池组构建 API 授权。由于 ID 和访问令牌都包含`cognito:groups`声明，因此您的策略存储可以在各种应用程序上下文中为您的 API 管理基于角色的访问控制 (RBAC)。

选择策略存储设置

在策略存储上配置身份源时，必须选择是要处理访问令牌还是 ID 令牌。此决定对您的策略引擎的运作方式具有重要意义。ID 令牌包含用户属性。访问令牌包含用户访问控制信息：O Auth 范围。尽管两种令牌类型都有群组成员资格信息，但我们通常建议使用带有已验证权限策略存储的 RBAC 的访问令牌。访问令牌可增加群组成员资格，其范围可以促成授权决策。访问令牌中的声明成为授权请求中的上下文。

在将用户池配置为身份源时，还必须配置用户和组实体类型。实体类型是您可以在 “已验证权限” 策略中引用的委托人、操作和资源标识符。策略存储中的实体可以具有成员关系，其中一个实体可以是父实体的成员。通过成员资格，您可以引用负责人小组、操作组和资源组。对于用户池群组，您指定的用户实体类型必须是该组实体类型的成员。当您设置与 API 关联的策略存储库或在 “已验证权限” 控制台中按照引导式设置操作时，您的策略存储会自动建立这种父成员关系。

ID 令牌可以将 RBAC 与基于属性的访问控制 (ABAC) 结合使用。创建与 API 关联的策略存储后，您可以使用用户属性和群组成员资格来增强您的政策。ID 令牌中的属性声明成为授权请求中的主体属性。您的策略可以根据委托人属性做出授权决定。

您也可以将策略存储配置为接受与您提供的可接受应用程序客户端列表相匹配的`aud`或`client_id`声明的令牌。

基于角色的 API 授权策略示例

以下示例策略是通过为示PetStore例 REST API 设置已验证权限策略存储而创建的。

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

在以下情况下，已验证的权限会返回对您的应用程序的授权请求的`Allow`决定：

1. 您的应用程序在`Authorization`标头中传递了 ID 或访问令牌作为持有者令牌。
2. 您的应用程序传递了一个带有`cognito:groups`声明的令牌，其中包含该字符串`MyGroup`。
3. 例如，您的应用程序向`https://myapi.example.com/pets`或发出了`HTTP GET`请求`https://myapi.example.com/pets/scrappy`。

Amazon Cognito 用户的示例策略

您的用户池还可以在 API 请求以外的条件下生成对已验证权限的授权请求。您可以将应用程序中的任何访问控制决策提交到策略存储区。例如，您可以在任何请求通过网络之前通过基于属性的访问控制来补充 Amazon DynamoDB 或 Amazon S3 的安全性，从而减少配额使用量。

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

以下请求正文会导致 `Allow` 响应。

```
{
   "accesstoken": "[John's ID 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：

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

```
   action,
   resource
);
```

以下是用户池中 ID 为 sub 或用户 ID us-east-1_Example 的用户的委托人示例973db890-092c-49e4-a9d0-912a4c0a20c7。

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

要在 “已验证权限” 策略中指定用户组时，请使用以下格式：

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

以下是一个示例

基于属性的访问控制

为您的应用程序提供经过验证的权限授权，以及用于 AWS 凭证的 Amazon Cognito 身份池的访问控制属性功能，都是基于属性的访问控制 (ABAC) 的形式。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	根据对用户池 JWT 的分析返回Allow或Deny决策。	根据 Cedar 策略评估，应用程序资源的访问成功或失败。
Amazon Cognito 身份池（用于访问控制的属性）	根据用户的属性为其分配会话标签。IAM 策略条件可以检查标签Allow或Deny用户访问权限 AWS 服务。	带有 IAM 角色临时 AWS 证书的标记会话。

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 配合使用。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

- 使用软件开发工具包的 Amazon Cognito 身份的代码示例 AWS
  - 使用软件开发工具包对 Amazon Cognito 身份执行的操作 AWS
    - CreateIdentityPool与 AWS SDK 或 CLI 配合使用
    - DeleteIdentityPool与 AWS SDK 或 CLI 配合使用
    - DescribeIdentityPool与 AWS SDK 或 CLI 配合使用
    - GetCredentialsForIdentity与 AWS SDK 或 CLI 配合使用
    - GetIdentityPoolRoles与 AWS SDK 或 CLI 配合使用
    - ListIdentityPools与 AWS SDK 或 CLI 配合使用
    - SetIdentityPoolRoles与 AWS SDK 或 CLI 配合使用
    - UpdateIdentityPool与 AWS SDK 或 CLI 配合使用
  - 使用软件开发工具包的 Amazon Cognito 身份的跨服务示例 AWS
    - 构建 Amazon Transcribe 应用程序
    - 创建 Amazon Textract 浏览器应用程序
- 使用软件开发工具包的 Amazon Cognito 身份提供商的代码示例 AWS
  - 使用软件开发工具包对亚马逊 Cognito 身份提供商采取的操作 AWS
    - AdminCreateUser与 AWS SDK 或 CLI 配合使用
    - AdminGetUser与 AWS SDK 或 CLI 配合使用
    - AdminInitiateAuth与 AWS SDK 或 CLI 配合使用
    - AdminRespondToAuthChallenge与 AWS SDK 或 CLI 配合使用
    - AdminSetUserPassword与 AWS SDK 或 CLI 配合使用
    - AssociateSoftwareToken与 AWS SDK 或 CLI 配合使用
    - ConfirmDevice与 AWS SDK 或 CLI 配合使用
    - ConfirmForgotPassword与 AWS SDK 或 CLI 配合使用
    - ConfirmSignUp与 AWS SDK 或 CLI 配合使用

- CreateUserPool与 AWS SDK 或 CLI 配合使用
- CreateUserPoolClient与 AWS SDK 或 CLI 配合使用
- DeleteUser与 AWS SDK 或 CLI 配合使用
- ForgotPassword与 AWS SDK 或 CLI 配合使用
- InitiateAuth与 AWS SDK 或 CLI 配合使用
- ListUserPools与 AWS SDK 或 CLI 配合使用
- ListUsers与 AWS SDK 或 CLI 配合使用
- ResendConfirmationCode与 AWS SDK 或 CLI 配合使用
- RespondToAuthChallenge与 AWS SDK 或 CLI 配合使用
- SignUp与 AWS SDK 或 CLI 配合使用
- UpdateUserPool与 AWS SDK 或 CLI 配合使用
- VerifySoftwareToken与 AWS SDK 或 CLI 配合使用

- 使用软件开发工具包的 Amazon Cognito 身份提供商的场景 AWS
  - 使用软件开发工具包通过 Lambda 函数自动确认已知的亚马逊 Cognito 用户 AWS
  - 使用软件开发工具包使用 Lambda 函数自动迁移已知的亚马逊 Cognito 用户 AWS
  - 使用需要使用软件开发工具包进行 MFA 的 Amazon Cognito 用户池注册用户 AWS
  - 使用软件开发工具包在 Amazon Cognito 用户身份验证后，使用 Lambda 函数编写自定义活动数据 AWS

- 使用软件开发工具包进行 Amazon Cognito 同步的代码示例 AWS
  - 使用软件开发工具包进行 Amazon Cognito 同步的操作 AWS
    - ListIdentityPoolUsage与 AWS SDK 或 CLI 配合使用

使用软件开发工具包的 Amazon Cognito 身份的代码示例 AWS

以下代码示例展示了如何将 Amazon Cognito Identity 与 AWS 软件开发套件 (SDK) 配合使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

跨服务示例是指跨多个 AWS 服务工作的示例应用程序。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

- 使用软件开发工具包对 Amazon Cognito 身份执行的操作 AWS
  - CreateIdentityPool与 AWS SDK 或 CLI 配合使用
  - DeleteIdentityPool与 AWS SDK 或 CLI 配合使用
  - DescribeIdentityPool与 AWS SDK 或 CLI 配合使用
  - GetCredentialsForIdentity与 AWS SDK 或 CLI 配合使用
  - GetIdentityPoolRoles与 AWS SDK 或 CLI 配合使用
  - ListIdentityPools与 AWS SDK 或 CLI 配合使用
  - SetIdentityPoolRoles与 AWS SDK 或 CLI 配合使用
  - UpdateIdentityPool与 AWS SDK 或 CLI 配合使用
- 使用软件开发工具包的 Amazon Cognito 身份的跨服务示例 AWS
  - 构建 Amazon Transcribe 应用程序
  - 创建 Amazon Textract 浏览器应用程序

使用软件开发工具包对 Amazon Cognito 身份执行的操作 AWS

以下代码示例演示了如何使用软件开发工具包执行单个 Amazon Cognito 身份操作。 AWS 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的链接 GitHub，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

- CreateIdentityPool与 AWS SDK 或 CLI 配合使用
- DeleteIdentityPool与 AWS SDK 或 CLI 配合使用
- DescribeIdentityPool与 AWS SDK 或 CLI 配合使用
- GetCredentialsForIdentity与 AWS SDK 或 CLI 配合使用
- GetIdentityPoolRoles与 AWS SDK 或 CLI 配合使用
- ListIdentityPools与 AWS SDK 或 CLI 配合使用
- SetIdentityPoolRoles与 AWS SDK 或 CLI 配合使用
- UpdateIdentityPool与 AWS SDK 或 CLI 配合使用

`CreateIdentityPool`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateIdentityPool`。

CLI

AWS CLI

使用 Cognito 身份池提供者创建身份池

此示例创建了一个名为的身份池 MyIdentityPool。它有一个 Cognito 身份池提供者。不允许使用未经身份验证的身份。

命令:

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

输出：

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateIdentityPool中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <identityPoolName>\s

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

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

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

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

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

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateIdentityPool中的。

PowerShell

用于 PowerShell

示例 1：创建允许未经身份验证的身份的新身份池。

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

输出：

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

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[CreateIdentityPool](#)中的。

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**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用于 S wift 的AWS SDK API 参考CreateIdentityPool中。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteIdentityPool`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteIdentityPool`。

CLI

AWS CLI

删除身份池

以下 `delete-identity-pool` 示例删除指定身份池。

命令:

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

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteIdentityPool中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteIdentityPool {

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

                Usage:
                    <identityPoolId>\s

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

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

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

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

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

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

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DeleteIdentityPool](#)中的。

PowerShell

用于 PowerShell

示例 1：删除特定的身份池。

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

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteIdentityPool中的。

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**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用于 S wift 的AWS SDK API 参考DeleteIdentityPool中。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeIdentityPool`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeIdentityPool`。

CLI

AWS CLI

描述身份池

此示例描述了身份池。

命令:

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

输出：

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeIdentityPool中的。

PowerShell

用于 PowerShell

示例 1：按身份池的 ID 检索有关该身份池的信息。

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

输出：

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

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeIdentityPool中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`GetCredentialsForIdentity`与 AWS SDK 或 CLI 配合使用

以下代码示例演示了如何使用 `GetCredentialsForIdentity`。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

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

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <identityId>\s

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

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

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

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

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

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

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetCredentialsForIdentity](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[将此服务与 AWS SDK 配合使用](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`GetIdentityPoolRoles`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetIdentityPoolRoles`。

CLI

AWS CLI

获取身份池角色

此示例获取身份池角色。

命令:

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

输出：

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考GetIdentityPoolRoles中的。

PowerShell

用于 PowerShell

示例 1：获取有关特定身份池的角色的信息。

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

输出：

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

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考GetIdentityPoolRoles中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListIdentityPools`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListIdentityPools`。

CLI

AWS CLI

列出身份池

此示例列出身份池。最多可列出 20 个身份。

命令:

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

输出：

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListIdentityPools中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

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

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

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

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListIdentityPools中的。

PowerShell

用于 PowerShell

示例 1：检索现有身份池的列表。

```
Get-CGIIdentityPoolList
```

输出：

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

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ListIdentityPools中的。

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**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用于 S wift 的AWS SDK API 参考ListIdentityPools中。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`SetIdentityPoolRoles`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `SetIdentityPoolRoles`。

CLI

AWS CLI

设置身份池角色

以下`set-identity-pool-roles`示例设置身份池角色。

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考SetIdentityPoolRoles中的。

PowerShell

用于 PowerShell

示例 1：将特定的身份池配置为具有未经身份验证的 IAM 角色。

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

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考SetIdentityPoolRoles中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`UpdateIdentityPool`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `UpdateIdentityPool`。

CLI

AWS CLI

更新身份池

此示例更新身份池。它将名称设置为 MyIdentityPool。它将 Cognito 添加为身份提供者。它不允许使用未经身份验证的身份。

命令:

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

输出：

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UpdateIdentityPool](#)中的。

PowerShell

用于 PowerShell

示例 1：更新某些身份池属性，在本例中为身份池的名称。

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

输出：

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

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考UpdateIdentityPool中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包的 Amazon Cognito 身份的跨服务示例 AWS

以下示例应用程序使用 AWS 软件开发工具包将 Amazon Cognito Identity 与其他应用程序组合在一起。 AWS 服务每个示例都包含一个指向的链接 GitHub，您可以在其中找到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
- 创建 Amazon Textract 浏览器应用程序

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

适用于 JavaScript (v3) 的软件开发工具包

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。经过身份验证的用户拥有 AWS Identity and Access Management (IAM) 访问所需 AWS 服务的权限。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

该示例也可在 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 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 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

适用于 JavaScript (v3) 的软件开发工具包

演示如何使用 AWS SDK for JavaScript 来构建 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将其显示在交互式网页中。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。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

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

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包的 Amazon Cognito 身份提供商的代码示例 AWS

以下代码示例展示了如何将 Amazon Cognito 身份提供程序与 AWS 软件开发套件 (SDK) 配合使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

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 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

开始使用

开始使用 Amazon Cognito

以下代码示例显示如何开始使用 Amazon Cognito。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。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。

C MakeLists .txt cMake 文件的代码。

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

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS cognito-idp)

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

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                 # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_cognito.cpp)
```

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

hello_cognito.cpp 源文件的代码。

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

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

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

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

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

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

```
                listUserPoolsRequest.SetNextToken(nextToken);
            }

            Aws::CognitoIdentityProvider::Model::ListUserPoolsOutcome
 listUserPoolsOutcome =
                    cognitoClient.ListUserPools(listUserPoolsRequest);

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

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

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


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

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考ListUserPools中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
```

```
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Go API 参考ListUserPools中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

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

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

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListUserPools中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。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。

```
import {
  paginateListUserPools,
  CognitoIdentityProviderClient,
} from "@aws-sdk/client-cognito-identity-provider";

const client = new CognitoIdentityProviderClient({});

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

  const userPoolNames = [];

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考ListUserPools中的。

代码示例

- 使用软件开发工具包对亚马逊 Cognito 身份提供商采取的操作 AWS
  - AdminCreateUser与 AWS SDK 或 CLI 配合使用
  - AdminGetUser与 AWS SDK 或 CLI 配合使用
  - AdminInitiateAuth与 AWS SDK 或 CLI 配合使用

- AdminRespondToAuthChallenge与 AWS SDK 或 CLI 配合使用
- AdminSetUserPassword与 AWS SDK 或 CLI 配合使用
- AssociateSoftwareToken与 AWS SDK 或 CLI 配合使用
- ConfirmDevice与 AWS SDK 或 CLI 配合使用
- ConfirmForgotPassword与 AWS SDK 或 CLI 配合使用
- ConfirmSignUp与 AWS SDK 或 CLI 配合使用
- CreateUserPool与 AWS SDK 或 CLI 配合使用
- CreateUserPoolClient与 AWS SDK 或 CLI 配合使用
- DeleteUser与 AWS SDK 或 CLI 配合使用
- ForgotPassword与 AWS SDK 或 CLI 配合使用
- InitiateAuth与 AWS SDK 或 CLI 配合使用
- ListUserPools与 AWS SDK 或 CLI 配合使用
- ListUsers与 AWS SDK 或 CLI 配合使用
- ResendConfirmationCode与 AWS SDK 或 CLI 配合使用
- RespondToAuthChallenge与 AWS SDK 或 CLI 配合使用
- SignUp与 AWS SDK 或 CLI 配合使用
- UpdateUserPool与 AWS SDK 或 CLI 配合使用
- VerifySoftwareToken与 AWS SDK 或 CLI 配合使用

- 使用软件开发工具包的 Amazon Cognito 身份提供商的场景 AWS
  - 使用软件开发工具包通过 Lambda 函数自动确认已知的亚马逊 Cognito 用户 AWS
  - 使用软件开发工具包使用 Lambda 函数自动迁移已知的亚马逊 Cognito 用户 AWS
  - 使用需要使用软件开发工具包进行 MFA 的 Amazon Cognito 用户池注册用户 AWS
  - 使用软件开发工具包在 Amazon Cognito 用户身份验证后，使用 Lambda 函数编写自定义活动数据 AWS

使用软件开发工具包对亚马逊 Cognito 身份提供商采取的操作 AWS

以下代码示例演示了如何使用软件开发工具包执行各个 Amazon Cognito 身份提供商操作。 AWS 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的链接 GitHub，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

- AdminCreateUser与 AWS SDK 或 CLI 配合使用
- AdminGetUser与 AWS SDK 或 CLI 配合使用
- AdminInitiateAuth与 AWS SDK 或 CLI 配合使用
- AdminRespondToAuthChallenge与 AWS SDK 或 CLI 配合使用
- AdminSetUserPassword与 AWS SDK 或 CLI 配合使用
- AssociateSoftwareToken与 AWS SDK 或 CLI 配合使用
- ConfirmDevice与 AWS SDK 或 CLI 配合使用
- ConfirmForgotPassword与 AWS SDK 或 CLI 配合使用
- ConfirmSignUp与 AWS SDK 或 CLI 配合使用
- CreateUserPool与 AWS SDK 或 CLI 配合使用
- CreateUserPoolClient与 AWS SDK 或 CLI 配合使用
- DeleteUser与 AWS SDK 或 CLI 配合使用
- ForgotPassword与 AWS SDK 或 CLI 配合使用
- InitiateAuth与 AWS SDK 或 CLI 配合使用
- ListUserPools与 AWS SDK 或 CLI 配合使用
- ListUsers与 AWS SDK 或 CLI 配合使用
- ResendConfirmationCode与 AWS SDK 或 CLI 配合使用
- RespondToAuthChallenge与 AWS SDK 或 CLI 配合使用
- SignUp与 AWS SDK 或 CLI 配合使用
- UpdateUserPool与 AWS SDK 或 CLI 配合使用
- VerifySoftwareToken与 AWS SDK 或 CLI 配合使用

`AdminCreateUser`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AdminCreateUser`。

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：

- 在完成 Amazon Cognito 用户身份验证后使用 Lambda 函数写入自定义活动数据

CLI

AWS CLI

创建用户

以下admin-create-user示例使用指定设置的电子邮件地址和电话号码创建用户。

```
aws cognito-idp admin-create-user \
    --user-pool-id us-west-2_aaaaaaaaa \
    --username diego \
    --user-attributes Name=email,Value=diego@example.com
 Name=phone_number,Value="+15555551212" \
    --message-action SUPPRESS
```

输出：

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AdminCreateUser中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

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



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Go API 参考AdminCreateUser中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AdminGetUser`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AdminGetUser`。

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 的用户池注册用户

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。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>Async task.</returns>
   public async Task<UserStatusType> GetAdminUserAsync(string userName, string
poolId)
   {
       AdminGetUserRequest userRequest = new AdminGetUserRequest
       {
           Username = userName,
           UserPoolId = poolId,
       };
```

```
        var response = await _cognitoService.AdminGetUserAsync(userRequest);

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

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考AdminGetUser中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for C++ API 参考AdminGetUser中的。

CLI

AWS CLI

获取用户

此示例获取有关用户名 jane@example.com 的信息。

命令:

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

输出：

```
{
  "Username": "4320de44-2322-4620-999b-5e2e1c8df013",
  "Enabled": true,
  "UserStatus": "FORCE_CHANGE_PASSWORD",
  "UserCreateDate": 1548108509.537,
  "UserAttributes": [
      {
          "Name": "sub",
          "Value": "4320de44-2322-4620-999b-5e2e1c8df013"
      },
      {
          "Name": "email_verified",
          "Value": "true"
      },
      {
          "Name": "phone_number_verified",
          "Value": "true"
      },
```

```
        {
            "Name": "phone_number",
            "Value": "+01115551212"
        },
        {
            "Name": "email",
            "Value": "jane@example.com"
        }
    ],
    "UserLastModifiedDate": 1548108509.537
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminGetUser](#)中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AdminGetUser中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。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 = ({ userPoolId, username }) => {
  const client = new CognitoIdentityProviderClient({});

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考AdminGetUser中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用AdminGetUser于 K otlin 的AWS SDK API 参考。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用AdminGetUser于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AdminInitiateAuth`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AdminInitiateAuth`。

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 的用户池注册用户

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。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 an admin auth request.
/// </summary>
/// <param name="clientId">The client ID to use.</param>
/// <param name="userPoolId">The ID of the user pool.</param>
/// <param name="userName">The username to authenticate.</param>
/// <param name="password">The user's password.</param>
```

```
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考AdminInitiateAuth中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for C++ API 参考AdminInitiateAuth中的。

CLI

AWS CLI

发起身份验证

此示例使用 ADMIN_NO_SRP_AUTH 流程发起对用户名 jane@example.com 的身份验证

客户端必须启用适用于基于服务器的身份验证（ADMIN_NO_SRP_AUTH）的登录 API。

使用返回值中的会话信息调用 admin-respond-to-auth-challenge。

命令:

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

输出：

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AdminInitiateAuth](#)中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AdminInitiateAuth中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。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 = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考AdminInitiateAuth中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用AdminInitiateAuth于 K otlin 的AWS SDK API 参考。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用AdminInitiateAuth于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AdminRespondToAuthChallenge`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AdminRespondToAuthChallenge`。

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 的用户池注册用户

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。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 admin authentication challenge.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="clientId">The client ID.</param>
    /// <param name="mfaCode">The multi-factor authentication code.</param>
    /// <param name="session">The current application session.</param>
    /// <param name="clientId">The user pool ID.</param>
    /// <returns>The result of the authentication response.</returns>
    public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
        string userName,
```

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

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

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考[AdminRespondToAuthChallenge](#)中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for C++ API 参考[AdminRespondToAuthChallenge](#)中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AdminRespondToAuthChallenge中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。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 = ({
  userPoolId,
  clientId,
  username,
  totp,
  session,
}) => {
  const client = new CognitoIdentityProviderClient({});
  const command = new AdminRespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: totp,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

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

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考AdminRespondToAuthChallenge中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用AdminRespondToAuthChallenge于 K otlin 的AWS SDK API 参考。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用AdminRespondToAuthChallenge于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AdminSetUserPassword`与 AWS SDK 或 CLI 配合使用

以下代码示例演示了如何使用 `AdminSetUserPassword`。

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：

- 在完成 Amazon Cognito 用户身份验证后使用 Lambda 函数写入自定义活动数据

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

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



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

```
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Go API 参考AdminSetUserPassword中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AssociateSoftwareToken`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AssociateSoftwareToken`。

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 的用户池注册用户

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。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>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

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

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

        return tokenResponse.Session;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考AssociateSoftwareToken中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for C++ API 参考AssociateSoftwareToken中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AssociateSoftwareToken中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。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 = (session) => {
```

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考AssociateSoftwareToken中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用AssociateSoftwareToken于 K otlin 的AWS SDK API 参考。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用AssociateSoftwareToken于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ConfirmDevice`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ConfirmDevice`。

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 的用户池注册用户

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for .NET API 参考ConfirmDevice中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。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 = ({ deviceKey, accessToken, passwordVerifier, salt }) => {
  const client = new CognitoIdentityProviderClient({});

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

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考ConfirmDevice中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用ConfirmDevice于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ConfirmForgotPassword`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ConfirmForgotPassword`。

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：

- 使用 Lambda 函数自动迁移已知用户

CLI

AWS CLI

确认忘记的密码

此示例确认忘记了用户名 diego@example.com 的密码。

命令:

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ConfirmForgotPassword中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

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



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

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

- 有关 API 的详细信息，请参阅 AWS SDK for Go API 参考ConfirmForgotPassword中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ConfirmSignUp`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ConfirmSignUp`。

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 的用户池注册用户

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。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>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
string userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

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

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考[ConfirmSignUp](#)中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for C++ API 参考ConfirmSignUp中的。

CLI

AWS CLI

确认注册

此示例确认用户名 diego@example.com 的注册。

命令:

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ConfirmSignUp中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ConfirmSignUp中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。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 = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考ConfirmSignUp中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用ConfirmSignUp于 K otlin 的AWS SDK API 参考。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用ConfirmSignUp于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateUserPool`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateUserPool`。

CLI

AWS CLI

创建最低配置的用户池

此示例创建了一个 MyUserPool 使用默认值命名的用户池。没有必要的属性，也没有应用程序客户端。MFA 和高级安全功能已禁用。

命令:

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

输出：

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

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

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

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

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

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

创建具有两个必要属性的用户池

此示例创建了一个用户池 MyUserPool。该池配置为接受电子邮件作为用户名属性。它还使用 Amazon Simple Email Service 将电子邮件源地址设置为经过验证的地址。

命令:

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

输出：

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

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

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

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

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

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

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateUserPool中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
```

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUserPool {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolName>\s

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

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

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

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

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

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

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateUserPool中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateUserPoolClient`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateUserPoolClient`。

CLI

AWS CLI

创建用户池客户端

此示例创建了一个具有两个显式授权流程的新用户池客户端：USER_PASSWORD_AUTH 和 ADMIN_NO_SRP_AUTH。

命令:

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

输出：

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateUserPoolClient中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientReques
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRespon

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

```
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <clientName> <userPoolId>\s

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

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

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

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

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

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

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateUserPoolClient中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteUser`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteUser`。

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：

- 使用 Lambda 函数自动确认已知用户
- 使用 Lambda 函数自动迁移已知用户
- 在完成 Amazon Cognito 用户身份验证后使用 Lambda 函数写入自定义活动数据

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for C++ API 参考DeleteUser中的。

CLI

AWS CLI

删除用户

此示例删除一个用户。

命令:

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteUser中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

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



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

- 有关 API 的详细信息，请参阅 AWS SDK for Go API 参考DeleteUser中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ForgotPassword`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ForgotPassword`。

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：

- 使用 Lambda 函数自动迁移已知用户

CLI

AWS CLI

强制更改密码

以下`forgot-password`示例向 jane@example.com 发送了一条消息，要求他们更改密码。

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

输出：

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

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ForgotPassword中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

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



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

- 有关 API 的详细信息，请参阅 AWS SDK for Go API 参考ForgotPassword中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`InitiateAuth`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `InitiateAuth`。

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：

- 使用 Lambda 函数自动确认已知用户
- 使用 Lambda 函数自动迁移已知用户
- 向需要 MFA 的用户池注册用户
- 在完成 Amazon Cognito 用户身份验证后使用 Lambda 函数写入自定义活动数据

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。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 initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest
```

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

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

        return response;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考InitiateAuth中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

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



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

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

- 有关 API 的详细信息，请参阅 AWS SDK for Go API 参考InitiateAuth中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。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 = ({ username, password, clientId }) => {
  const client = new CognitoIdentityProviderClient({});

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考InitiateAuth中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。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。

此示例演示了如何使用被跟踪设备开始身份验证。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 的详细信息，请参阅适用InitiateAuth于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListUserPools`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListUserPools`。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for .NET API 参考ListUserPools中的。

CLI

AWS CLI

列出用户池

此示例最多列出 20 个用户池。

命令:

```
aws cognito-idp list-user-pools --max-results 20
```

输出：

```
{
  "UserPools": [
      {
         "CreationDate": 1547763720.822,
         "LastModifiedDate": 1547763720.822,
         "LambdaConfig": {},
         "Id": "us-west-2_aaaaaaaaa",
         "Name": "MyUserPool"
      }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListUserPools中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
```

```
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Go API 参考ListUserPools中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

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

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

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListUserPools中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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?;
    let pools = response.user_pools();
    println!("User pools:");
    for pool in pools {
        println!("  ID:              {}", pool.id().unwrap_or_default());
        println!("  Name:            {}", pool.name().unwrap_or_default());
        println!("  Lambda Config:   {:?}", pool.lambda_config().unwrap());
        println!(
            "  Last modified:   {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!(
            "  Creation date:   {:?}",
            pool.creation_date().unwrap().to_chrono_utc()
        );
        println!();
    }
    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用ListUserPools于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListUsers`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListUsers`。

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 的用户池注册用户

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for .NET API 参考ListUsers中的。

CLI

AWS CLI

列出用户

此示例最多列出 20 个用户。

命令:

```
aws cognito-idp list-users --user-pool-id us-west-2_aaaaaaaaa --limit 20
```

输出：

```
{
  "Users": [
      {
          "Username": "22704aa3-fc10-479a-97eb-2af5806bd327",
          "Enabled": true,
          "UserStatus": "FORCE_CHANGE_PASSWORD",
          "UserCreateDate": 1548089817.683,
          "UserLastModifiedDate": 1548089817.683,
          "Attributes": [
              {
                  "Name": "sub",
                  "Value": "22704aa3-fc10-479a-97eb-2af5806bd327"
              },
              {
                  "Name": "email_verified",
                  "Value": "true"
              },
              {
                  "Name": "email",
                  "Value": "mary@example.com"
              }
          ]
      }
  ]
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListUsers中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUsers {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolId>\s

                Where:
```

```
                    userPoolId - The ID given to your user pool when it's
created.
                """";

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

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

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }

    public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {
        try {
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created "
                        + user.userCreateDate());
            });

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

    // Shows how to list users by using a filter.
    public static void listUsersFilter(CognitoIdentityProviderClient
cognitoClient, String userPoolId) {
```

```
        try {
            String filter = "email = \"tblue@noserver.com\"";
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .filter(filter)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User with filter applied " + user.username()
 + " Status " + user.userStatus()
                        + " Created " + user.userCreateDate());
            });

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

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListUsers中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。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 = ({ userPoolId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ListUsersCommand({
    UserPoolId: userPoolId,
  });

  return client.send(command);
```

```
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考ListUsers中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用ListUsers于 K otlin 的AWS SDK API 参考。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用ListUsers于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ResendConfirmationCode`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ResendConfirmationCode`。

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 的用户池注册用户

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。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>The delivery details.</returns>
```

```
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
_cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
{response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考ResendConfirmationCode中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for C++ API 参考ResendConfirmationCode中的。

CLI

AWS CLI

重新发送确认码

以下 `resend-confirmation-code` 示例向用户 `jane` 发送确认码。

```
aws cognito-idp resend-confirmation-code \
    --client-id 12a3b456c7de890f11g123hijk \
    --username jane
```

输出：

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

```
    }
}
```

有关更多信息，请参阅《Amazon Cognito 开发人员指南》中的注册并确认用户账户。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ResendConfirmationCode中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ResendConfirmationCode中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。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 = ({ clientId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ResendConfirmationCodeCommand({
    ClientId: clientId,
    Username: username,
  });

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

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考ResendConfirmationCode中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用ResendConfirmationCode于 K otlin 的AWS SDK API 参考。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用ResendConfirmationCode于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`RespondToAuthChallenge`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `RespondToAuthChallenge`。

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 的用户池注册用户

CLI

AWS CLI

响应身份验证质询

此示例响应使用 initiate-auth 发起的身份验证质询。这是对 NEW_PASSWORD_REQUIRED 质询的响应。它为用户 jane@example.com 设置了密码。

命令:

```
aws cognito-idp respond-to-auth-challenge --client-id 3n4b5urk1ft4fl3mg5e62d9ado
 --challenge-name NEW_PASSWORD_REQUIRED --challenge-responses
 USERNAME=jane@example.com,NEW_PASSWORD="password" --session "SESSION_TOKEN"
```

输出：

```
{
  "ChallengeParameters": {},
  "AuthenticationResult": {
      "AccessToken": "ACCESS_TOKEN",
      "ExpiresIn": 3600,
      "TokenType": "Bearer",
      "RefreshToken": "REFRESH_TOKEN",
      "IdToken": "ID_TOKEN",
      "NewDeviceMetadata": {
          "DeviceKey": "us-west-2_fec070d2-fa88-424a-8ec8-b26d7198eb23",
          "DeviceGroupKey": "-wt2ha1Zd"
      }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考RespondToAuthChallenge中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。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 = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

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

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考RespondToAuthChallenge中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用RespondToAuthChallenge于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`SignUp`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `SignUp`。

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：

- 使用 Lambda 函数自动确认已知用户
- 使用 Lambda 函数自动迁移已知用户
- 向需要 MFA 的用户池注册用户

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for .NET API 参考SignUp中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for C++ API 参考SignUp中的。

CLI

AWS CLI

注册用户

此示例注册 jane@example.com。

命令:

```
aws cognito-idp sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username jane@example.com --password PASSWORD --user-attributes
 Name="email",Value="jane@example.com" Name="name",Value="Jane"
```

输出：

```
{
  "UserConfirmed": false,
  "UserSub": "e04d60a6-45dc-441c-a40b-e25a787d4862"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考SignUp中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

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



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
```

```
    {Name: aws.String("email"), Value: aws.String(userEmail)},
   },
  })
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    log.Println(*invalidPassword.Message)
   } else {
    log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
   }
  } else {
   confirmed = output.UserConfirmed
  }
  return confirmed, err
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Go API 参考SignUp中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SignUp中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。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 = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

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

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考SignUp中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用SignUp于 K otlin 的AWS SDK API 参考。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用SignUp于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`UpdateUserPool`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `UpdateUserPool`。

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：

- 使用 Lambda 函数自动确认已知用户
- 使用 Lambda 函数自动迁移已知用户
- 在完成 Amazon Cognito 用户身份验证后使用 Lambda 函数写入自定义活动数据

CLI

AWS CLI

更新用户池

此示例向用户池添加标签。

命令:

```
aws cognito-idp update-user-pool --user-pool-id us-west-2_aaaaaaaaa --user-pool-
tags Team=Blue,Area=West
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考UpdateUserPool中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

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

```
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
```

```
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Go API 参考UpdateUserPool中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`VerifySoftwareToken`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `VerifySoftwareToken`。

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 的用户池注册用户

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for .NET API 参考VerifySoftwareToken中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for C++ API 参考VerifySoftwareToken中的。

Java

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考VerifySoftwareToken中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。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 = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

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

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考VerifySoftwareToken中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用VerifySoftwareToken于 K otlin 的AWS SDK API 参考。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。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 的详细信息，请参阅适用VerifySoftwareToken于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包的 Amazon Cognito 身份提供商的场景 AWS

以下代码示例向您展示了如何使用软件开发工具包在 Amazon Cognito 身份提供商中实现常见场景。AWS 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的链接 GitHub，您可以在其中找到有关如何设置和运行代码的说明。

示例

- 使用软件开发工具包通过 Lambda 函数自动确认已知的亚马逊 Cognito 用户 AWS
- 使用软件开发工具包使用 Lambda 函数自动迁移已知的亚马逊 Cognito 用户 AWS
- 使用需要使用软件开发工具包进行 MFA 的 Amazon Cognito 用户池注册用户 AWS
- 使用软件开发工具包在 Amazon Cognito 用户身份验证后，使用 Lambda 函数编写自定义活动数据 AWS

使用软件开发工具包通过 Lambda 函数自动确认已知的亚马逊 Cognito 用户 AWS

以下代码示例显示了如何使用 Lambda 函数自动确认已知的 Amazon Cognito 用户。

- 配置用户池以调用 `PreSignUp` 触发器的 Lambda 函数。

- 将用户注册到 Amazon Cognito
- Lambda 函数会扫描 DynamoDB 表并自动确认已知用户。
- 以新用户身份登录，然后清理资源。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

在命令提示符中运行交互式场景。

```
// AutoConfirm separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type AutoConfirm struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) AutoConfirm {
 scenario := AutoConfirm{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}
```

```
// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
 PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(userPoolId string, functionArn
 string) {
 log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
 Cognito.\n" +
  "This trigger happens when a user signs up, and lets your function take action
 before the main Cognito\n" +
  "sign up processing occurs.\n")
 err := runner.cognitoActor.UpdateTriggers(
  userPoolId,
  actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
 trigger.\n",
  functionArn, userPoolId)
}

// SignUpUser signs up a user from the known user table with a password you
 specify.
func (runner *AutoConfirm) SignUpUser(clientId string, usersTable string)
 (string, string) {
 log.Println("Let's sign up a user to your Cognito user pool. When the user's
 email matches an email in the\n" +
  "DynamoDB known users table, it is automatically verified and the user is
 confirmed.")

 knownUsers, err := runner.helper.GetKnownUsers(usersTable)
 if err != nil {
  panic(err)
 }
 userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
 knownUsers.UserNameList())
 user := knownUsers.Users[userChoice]

 var signedUp bool
 var userConfirmed bool
 password := runner.questioner.AskPassword("Enter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !signedUp {
```

```
  log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
 user.UserEmail)
  userConfirmed, err = runner.cognitoActor.SignUp(clientId, user.UserName,
 password, user.UserEmail)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("Enter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   signedUp = true
  }
 }
 log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

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

 return user.UserName, password
}

// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(clientId string, userName string, password
 string) string {
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 log.Printf("Let's sign in as %v...\n", userName)
 authResult, err := runner.cognitoActor.SignIn(clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 log.Println(strings.Repeat("-", 88))
 return *authResult.AccessToken
}

// Run runs the scenario.
func (runner *AutoConfirm) Run(stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup()
  }
```

```
 }()

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

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

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

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

 runner.resources.Cleanup()

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

使用 Lambda 函数处理 PreSignUp 触发器。

```
const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}
```

```
// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
 error) {
 log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "PreSignUp_SignUp" {
  // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the
 response from this handler.
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserEmail: event.Request.UserAttributes["email"],
 }
 log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
 output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key:       user.GetKey(),
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Error looking up email %v.\n", user.UserEmail)
  return event, err
 }
 if output.Item == nil {
  log.Printf("Email %v not found. Email verification is required.\n",
 user.UserEmail)
  return event, err
```

```
 }

 err = attributevalue.UnmarshalMap(output.Item, &user)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
  return event, err
 }

 if user.UserName != event.UserName {
  log.Printf("UserEmail %v found, but stored UserName '%v' does not match
 supplied UserName '%v'. Verification is required.\n",
   user.UserEmail, user.UserName, event.UserName)
 } else {
  log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.
\n", user.UserEmail, user.UserName)
  event.Response.AutoConfirmUser = true
  event.Response.AutoVerifyEmail = true
 }

 return event, err
}

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

创建一个执行常见任务的结构。

```
// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(stackName string) (actions.StackOutputs, error)
```

```
 PopulateUserTable(tableName string)
 GetKnownUsers(tableName string) (actions.UserList, error)
 AddKnownUser(tableName string, user actions.User)
 ListRecentLogEvents(functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(stackName), nil
}
```

```
// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(tableName string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(tableName string) (actions.UserList,
 error) {
 knownUsers, err := helper.dynamoActor.Scan(tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(tableName string, user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(functionName)
 if err != nil {
  panic(err)
```

```
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

创建一个封装 Amazon Cognito 操作的结构。

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



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
```

```
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
```

```
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(clientId string, userName string, password
 string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(context.TODO(),
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

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



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(clientId string, code string,
 userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(context.TODO(),
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}
```

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



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
```

```
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

创建一个封装 DynamoDB 操作的结构。

```
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
```

```
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(tableName string) error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(context.TODO(),
 &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
```

```
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(tableName string) (UserList, error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(context.TODO(), &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(tableName string, user User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

创建一个封装 CloudWatch 日志操作的结构。

```
type CloudWatchLogsActions struct {
```

```
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(functionName string)
 (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(context.TODO(),
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(functionName string,
 logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(context.TODO(),
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
```

```
}
```

创建一个封装动作的结构。 AWS CloudFormation

```
// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(stackName string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(context.TODO(),
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

清理资源。

```
// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger
```

```
 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(resources.userPoolId,
 triggerList...)
  if err != nil {
```

```
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的以下主题。
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包使用 Lambda 函数自动迁移已知的亚马逊 Cognito 用户 AWS

以下代码示例显示了如何使用 Lambda 函数自动迁移已知的 Amazon Cognito 用户。

- 配置用户池以调用 `MigrateUser` 触发器的 Lambda 函数。
- 使用不在用户池中的用户名和电子邮件地址登录 Amazon Cognito。
- Lambda 函数会扫描 DynamoDB 表并自动将已知用户迁移到该用户池。
- 执行“忘记密码”流程可重置已迁移用户的密码。
- 以新用户身份登录，然后清理资源。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

在命令提示符中运行交互式场景。

```
import (
 "errors"
 "fmt"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// MigrateUser separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type MigrateUser struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) MigrateUser {
 scenario := MigrateUser{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
```

```
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
 MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(userPoolId string, functionArn
 string) {
 log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
 Cognito.\n" +
  "This trigger happens when an unknown user signs in, and lets your function
 take action before Cognito\n" +
  "rejects the user.\n\n")
 err := runner.cognitoActor.UpdateTriggers(
  userPoolId,
  actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
 trigger.\n",
  functionArn, userPoolId)

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

// SignInUser adds a new user to the known users table and signs that user in to
 Amazon Cognito.
func (runner *MigrateUser) SignInUser(usersTable string, clientId string) (bool,
 actions.User) {
 log.Println("Let's sign in a user to your Cognito user pool. When the username
 and email matches an entry in the\n" +
  "DynamoDB known users table, the email is automatically verified and the user
 is migrated to the Cognito user pool.")

 user := actions.User{}
 user.UserName = runner.questioner.Ask("\nEnter a username:")
 user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This
 email will be used to confirm user migration\n" +
  "during this example:")
```

```
 runner.helper.AddKnownUser(usersTable, user)

 var err error
 var resetRequired *types.PasswordResetRequiredException
 var authResult *types.AuthenticationResultType
 signedIn := false
 for !signedIn && resetRequired == nil {
  log.Printf("Signing in to Cognito as user '%v'. The expected result is a
 PasswordResetRequiredException.\n\n", user.UserName)
  authResult, err = runner.cognitoActor.SignIn(clientId, user.UserName, "_")
  if err != nil {
   if errors.As(err, &resetRequired) {
    log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
 DynamoDB known users table.\n"+
     "User migration is started and a password reset is required.",
 user.UserName)
   } else {
    panic(err)
   }
  } else {
   log.Printf("User '%v' successfully signed in. This is unexpected and probably
 means you have not\n"+
    "cleaned up a previous run of this scenario, so the user exist in the Cognito
 user pool.\n"+
    "You can continue this example and select to clean up resources, or manually
 remove\n"+
    "the user from your user pool and try again.", user.UserName)
   runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
   signedIn = true
  }
 }

 log.Println(strings.Repeat("-", 88))
 return resetRequired != nil, user
}

// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(clientId string, user actions.User) {
 wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user
 to Cognito, you must be able to receive a confirmation\n"+
  "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
 if !wantCode {
```

```
 log.Println("To complete this example and successfully migrate a user to
Cognito, you must enter an email\n" +
  "you own that can receive a confirmation code.")
 return
}
codeDelivery, err := runner.cognitoActor.ForgotPassword(clientId, user.UserName)
if err != nil {
 panic(err)
}
log.Printf("\nA confirmation code has been sent to %v.",
*codeDelivery.Destination)
code := runner.questioner.Ask("Check your email and enter it here:")

confirmed := false
password := runner.questioner.AskPassword("\nEnter a password that has at least
eight characters, uppercase, lowercase, numbers and symbols.\n"+
 "(the password will not display as you type):", 8)
for !confirmed {
 log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
 err = runner.cognitoActor.ConfirmForgotPassword(clientId, code, user.UserName,
password)
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   password = runner.questioner.AskPassword("\nEnter another password:", 8)
  } else {
   panic(err)
  }
 } else {
  confirmed = true
 }
}
log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
log.Println("Signing in with your username and password...")
authResult, err := runner.cognitoActor.SignIn(clientId, user.UserName, password)
if err != nil {
 panic(err)
}
log.Printf("Successfully signed in. Your access token starts with: %v...\n",
(*authResult.AccessToken)[:10])
runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
*authResult.AccessToken)

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

```
}

// Run runs the scenario.
func (runner *MigrateUser) Run(stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup()
  }
 }()

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

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

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

 runner.AddMigrateUserTrigger(stackOutputs["UserPoolId"],
 stackOutputs["MigrateUserFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers,
 actions.UserMigration)
 resetNeeded, user := runner.SignInUser(stackOutputs["TableName"],
 stackOutputs["UserPoolClientId"])
 if resetNeeded {
  runner.helper.ListRecentLogEvents(stackOutputs["MigrateUserFunction"])
  runner.ResetPassword(stackOutputs["UserPoolClientId"], user)
 }

 runner.resources.Cleanup()

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

使用 Lambda 函数处理 `MigrateUser` 触发器。

```
const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsMigrateUser)
 (events.CognitoEventUserPoolsMigrateUser, error) {
 log.Printf("Received migrate trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 if event.TriggerSource != "UserMigration_Authentication" {
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName: event.UserName,
 }
 log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
 filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
 expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
 if err != nil {
  log.Printf("Error building expression to query for user '%v'.\n",
 user.UserName)
  return event, err
 }
 output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName:                 aws.String(tableName),
  FilterExpression:          expr.Filter(),
  ExpressionAttributeNames:  expr.Names(),
  ExpressionAttributeValues: expr.Values(),
 })
```

```
 if err != nil {
  log.Printf("Error looking up user '%v'.\n", user.UserName)
  return event, err
 }
 if output.Items == nil || len(output.Items) == 0 {
  log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
  return event, err
 }

 var users []UserInfo
 err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
  return event, err
 }

 user = users[0]
 log.Printf("UserName '%v' found with email %v. User is migrated and must reset
 password.\n", user.UserName, user.UserEmail)
 event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes =
 map[string]string{
  "email":          user.UserEmail,
  "email_verified": "true", // email_verified is required for the forgot password
 flow.
 }
 event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus =
 "RESET_REQUIRED"
 event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

 return event, err
}

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

创建一个执行常见任务的结构。

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

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
```

```
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(tableName string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(tableName string) (actions.UserList,
 error) {
 knownUsers, err := helper.dynamoActor.Scan(tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(tableName string, user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(tableName, user)
 if err != nil {
  panic(err)
 }
}
```

```
// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

创建一个封装 Amazon Cognito 操作的结构。

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



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
```

```
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

```
// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



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

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



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



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

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



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



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
```

```
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

创建一个封装 DynamoDB 操作的结构。

```
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
```

```
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(tableName string) error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
```

```
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(context.TODO(),
 &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(tableName string) (UserList, error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(context.TODO(), &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(tableName string, user User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
```

```
 }
 return err
}
```

创建一个封装 CloudWatch 日志操作的结构。

```
type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(functionName string)
 (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(context.TODO(),
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(functionName string,
 logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(context.TODO(),
 &cloudwatchlogs.GetLogEventsInput{
```

```
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

创建一个封装动作的结构。 AWS CloudFormation

```
// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(stackName string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(context.TODO(),
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

清理资源。

```
// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
```

```
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的以下主题。
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用需要使用软件开发工具包进行 MFA 的 Amazon Cognito 用户池注册用户 AWS

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**
>
> 还有更多相关信息 GitHub。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。

```
namespace CognitoBasics;

public class CognitoBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Cognito.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonCognitoIdentityProvider>()
            .AddTransient<CognitoWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<CognitoBasics>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
```

```
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var cognitoWrapper = host.Services.GetRequiredService<CognitoWrapper>();

        Console.WriteLine(new string('-', 80));
        UiMethods.DisplayOverview();
        Console.WriteLine(new string('-', 80));

        // clientId - The app client Id value that you get from the AWS CDK
script.
        var clientId = configuration["ClientId"]; // "*** REPLACE WITH CLIENT ID
VALUE FROM CDK SCRIPT";

        // poolId - The pool Id that you get from the AWS CDK script.
        var poolId = configuration["PoolId"]!; // "*** REPLACE WITH POOL ID VALUE
FROM CDK SCRIPT";
        var userName = configuration["UserName"];
        var password = configuration["Password"];
        var email = configuration["Email"];

        // If the username wasn't set in the configuration file,
        // get it from the user now.
        if (userName is null)
        {
            do
            {
                Console.Write("Username: ");
                userName = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(userName));
        }
        Console.WriteLine($"\nUsername: {userName}");

        // If the password wasn't set in the configuration file,
        // get it from the user now.
        if (password is null)
        {
            do
            {
                Console.Write("Password: ");
                password = Console.ReadLine();
            }
```

```
            while (string.IsNullOrEmpty(password));
        }

        // If the email address wasn't set in the configuration file,
        // get it from the user now.
        if (email is null)
        {
            do
            {
                Console.Write("Email: ");
                email = Console.ReadLine();
            } while (string.IsNullOrEmpty(email));
        }

        // Now sign up the user.
        Console.WriteLine($"\nSigning up {userName} with email address:
{email}");
        await cognitoWrapper.SignUpAsync(clientId, userName, password, email);

        // Add the user to the user pool.
        Console.WriteLine($"Adding {userName} to the user pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        UiMethods.DisplayTitle("Get confirmation code");
        Console.WriteLine($"Conformation code sent to {userName}.");
        Console.Write("Would you like to send a new code? (Y/N) ");
        var answer = Console.ReadLine();

        if (answer!.ToLower() == "y")
        {
            await cognitoWrapper.ResendConfirmationCodeAsync(clientId, userName);
            Console.WriteLine("Sending a new confirmation code");
        }

        Console.Write("Enter confirmation code (from Email): ");
        var code = Console.ReadLine();

        await cognitoWrapper.ConfirmSignupAsync(clientId, code, userName);

        UiMethods.DisplayTitle("Checking status");
        Console.WriteLine($"Rechecking the status of {userName} in the user
pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);
```

```
        Console.WriteLine($"Setting up authenticator for {userName} in the user
 pool");
        var setupResponse = await cognitoWrapper.InitiateAuthAsync(clientId,
 userName, password);

        var setupSession = await
 cognitoWrapper.AssociateSoftwareTokenAsync(setupResponse.Session);
        Console.Write("Enter the 6-digit code displayed in Google Authenticator:
 ");
        var setupCode = Console.ReadLine();

        var setupResult = await
 cognitoWrapper.VerifySoftwareTokenAsync(setupSession, setupCode);
        Console.WriteLine($"Setup status: {setupResult}");

        Console.WriteLine($"Now logging in {userName} in the user pool");
        var authSession = await cognitoWrapper.AdminInitiateAuthAsync(clientId,
 poolId, userName, password);

        Console.Write("Enter a new 6-digit code displayed in Google
 Authenticator: ");
        var authCode = Console.ReadLine();

        var authResult = await
 cognitoWrapper.AdminRespondToAuthChallengeAsync(userName, clientId, authCode,
 authSession, poolId);
        Console.WriteLine($"Authenticated and received access token:
 {authResult.AccessToken}");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cognito scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }
}


using System.Net;

namespace CognitoActions;

/// <summary>
/// Methods to perform Amazon Cognito Identity Provider actions.
/// </summary>
public class CognitoWrapper
```

```
{
    private readonly IAmazonCognitoIdentityProvider _cognitoService;

    /// <summary>
    /// Constructor for the wrapper class containing Amazon Cognito actions.
    /// </summary>
    /// <param name="cognitoService">The Amazon Cognito client object.</param>
    public CognitoWrapper(IAmazonCognitoIdentityProvider cognitoService)
    {
        _cognitoService = cognitoService;
    }

    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
 ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }


    /// <summary>
    /// Get a list of users for the Amazon Cognito user pool.
    /// </summary>
    /// <param name="userPoolId">The user pool ID.</param>
    /// <returns>A list of users.</returns>
    public async Task<List<UserType>> ListUsersAsync(string userPoolId)
    {
        var request = new ListUsersRequest
        {
            UserPoolId = userPoolId
        };
```

```
        var users = new List<UserType>();

        var usersPaginator = _cognitoService.Paginators.ListUsers(request);
        await foreach (var response in usersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


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

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

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

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


    /// <summary>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }


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

        var tokenResponse = await
_cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
```

```
        var secretCode = tokenResponse.SecretCode;

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

        return tokenResponse.Session;
    }


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

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

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

    /// <summary>
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
    /// <param name="password">The password for the user who is authenticating.</
param>
```

```
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

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

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

        return response;
    }

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

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

```
        }
        return false;
    }


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

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


    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };
```

```
        var response = await
 _cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
 {response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }


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

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

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


    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
```

```
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

}
```

- 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++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。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

适用于 Java 2.x 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRespons
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito_scenario_user_pool_with_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * “ChallengeName”: “MFA_SETUP”).
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
 * key. This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 * MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 * prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
 InvalidKeyException {
        final String usage = """

                Usage:
                    <clientId> <poolId>

                Where:
                    clientId - The app client Id value that you can get from the
 AWS CDK script.
                    poolId - The pool Id that you can get from the AWS CDK
 script.\s
                """;
```

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

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("*** Enter your password");
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();

        System.out.println("1. Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Getting " + userName + " in the user pool");
        getAdminUser(identityProviderClient, userName, poolId);

        System.out
                .println("*** Conformation code sent to " + userName + ". Would
you like to send a new code? (Yes/No)");
        System.out.println(DASHES);

        System.out.println(DASHES);
        String ans = in.nextLine();

        if (ans.compareTo("Yes") == 0) {
```

```
            resendConfirmationCode(identityProviderClient, clientId, userName);
            System.out.println("3. Sending a new confirmation code");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enter confirmation code that was emailed");
        String code = in.nextLine();
        confirmSignUp(identityProviderClient, clientId, code, userName);
        System.out.println("Rechecking the status of " + userName + " in the user
pool");
        getAdminUser(identityProviderClient, userName, poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Invokes the initiateAuth to sign in");
        AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        String mySession = authResponse.session();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to
generate a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient,
mySession);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
```

```
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9.  Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

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

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

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

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

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

```
   // Verify the TOTP and register for MFA.
   public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
       try {
           VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                   .userCode(code)
                   .session(session)
                   .build();

           VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
           System.out.println("The status of the token is " +
verifyResponse.statusAsString());

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

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

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

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

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

        return null;
    }

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

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

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

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

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

    public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
```

```
            String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                    .clientId(clientId)
                    .username(userName)
                    .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

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

    public static void signUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

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

```
    }

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

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

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

- 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

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。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 "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { signUp } from "../../../actions/sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username, password, email) => {
  if (!(username && password && email)) {
    throw new Error(
      `Username, password, and email must be provided as arguments to the 'sign-
up' command.`,
    );
  }
};

const signUpHandler = async (commands) => {
  const [_, username, password, email] = commands;

  try {
    validateUser(username, password, email);
    /**
```

```
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    log(`Signing up.`);
    await signUp({ clientId, username, password, email });
    log(`Signed up. A confirmation email has been sent to: ${email}.`);
    log(`Run 'confirm-sign-up ${username} <code>' to confirm your account.`);
  } catch (err) {
    log(err);
  }
};

export { signUpHandler };

const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};

import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { confirmSignUp } from "../../../actions/confirm-sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username) => {
```

```
  if (!username) {
    throw new Error(
      `Username name is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const validateCode = (code) => {
  if (!code) {
    throw new Error(
      `Verification code is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const confirmSignUpHandler = async (commands) => {
  const [_, username, code] = commands;

  try {
    validateUser(username);
    validateCode(code);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    log(`Confirming user.`);
    await confirmSignUp({ clientId, username, code });
    log(
      `User confirmed. Run 'admin-initiate-auth ${username} <password>' to sign
 in.`,
    );
  } catch (err) {
    log(err);
  }
};

export { confirmSignUpHandler };

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

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

  return client.send(command);
};

import qrcode from "qrcode-terminal";
import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminInitiateAuth } from "../../../actions/admin-initiate-auth.js";
import { associateSoftwareToken } from "../../../actions/associate-software-
token.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";

const handleMfaSetup = async (session, username) => {
  const { SecretCode, Session } = await associateSoftwareToken(session);

  // Store the Session for use with 'VerifySoftwareToken'.
  process.env.SESSION = Session;

  console.log(
    "Scan this code in your preferred authenticator app, then run 'verify-
software-token' to finish the setup.",
  );
  qrcode.generate(
    `otpauth://totp/${username}?secret=${SecretCode}`,
    { small: true },
    console.log,
  );
};

const handleSoftwareTokenMfa = (session) => {
  // Store the Session for use with 'AdminRespondToAuthChallenge'.
  process.env.SESSION = session;
};

const validateClient = (id) => {
  if (!id) {
    throw new Error(
      `User pool client id is missing. Did you run 'create-user-pool'?`,
```

```
    );
  }
};

const validateId = (id) => {
  if (!id) {
    throw new Error(`User pool id is missing. Did you run 'create-user-pool'?`);
  }
};

const validateUser = (username, password) => {
  if (!(username && password)) {
    throw new Error(
      `Username and password must be provided as arguments to the 'admin-
initiate-auth' command.`,
    );
  }
};

const adminInitiateAuthHandler = async (commands) => {
  const [_, username, password] = commands;

  try {
    validateUser(username, password);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    validateId(userPoolId);
    validateClient(clientId);

    log("Signing in.");
    const { ChallengeName, Session } = await adminInitiateAuth({
      clientId,
      userPoolId,
      username,
      password,
    });

    if (ChallengeName === "MFA_SETUP") {
      log("MFA setup is required.");
      return handleMfaSetup(Session, username);
    }

    if (ChallengeName === "SOFTWARE_TOKEN_MFA") {
      handleSoftwareTokenMfa(Session);
```

```
      log(`Run 'admin-respond-to-auth-challenge ${username} <totp>'`);
    }
  } catch (err) {
    log(err);
  }
};

export { adminInitiateAuthHandler };

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

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

  return client.send(command);
};

import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminRespondToAuthChallenge } from "../../../actions/admin-respond-to-
auth-challenge.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";
import { FILE_USER_POOLS } from "./constants.js";

const verifyUsername = (username) => {
  if (!username) {
    throw new Error(
      `Username is missing. It must be provided as an argument to the 'admin-
respond-to-auth-challenge' command.`,
    );
  }
};

const verifyTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) is missing. It must be provided as an
 argument to the 'admin-respond-to-auth-challenge' command.`,
    );
  }
```

```
};

const storeAccessToken = (token) => {
  process.env.AccessToken = token;
};

const adminRespondToAuthChallengeHandler = async (commands) => {
  const [_, username, totp] = commands;

  try {
    verifyUsername(username);
    verifyTotp(totp);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    const session = process.env.SESSION;

    const { AuthenticationResult } = await adminRespondToAuthChallenge({
      clientId,
      userPoolId,
      username,
      totp,
      session,
    });

    storeAccessToken(AuthenticationResult.AccessToken);

    log("Successfully authenticated.");
  } catch (err) {
    log(err);
  }
};

export { adminRespondToAuthChallengeHandler };

const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
```

```
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};

import { log } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { verifySoftwareToken } from "../../../actions/verify-software-token.js";

const validateTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) must be provided to the 'validate-
software-token' command.`,
    );
  }
};
const verifySoftwareTokenHandler = async (commands) => {
  const [_, totp] = commands;

  try {
    validateTotp(totp);

    log("Verifying TOTP.");
    await verifySoftwareToken(totp);
    log("TOTP Verified. Run 'admin-initiate-auth' again to sign-in.");
  } catch (err) {
    console.log(err);
  }
};

export { verifySoftwareTokenHandler };

const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
```

```
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

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

- 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

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。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。

创建一个包含场景中使用的 Amazon Cognito 函数的类。

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

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


    def _secret_hash(self, user_name):
        """
        Calculates a secret hash from a user name and a client secret.
```

```
        :param user_name: The user name to use when calculating the hash.
        :return: The secret hash.
        """
        key = self.client_secret.encode()
        msg = bytes(user_name + self.client_id, "utf-8")
        secret_hash = base64.b64encode(
            hmac.new(key, msg, digestmod=hashlib.sha256).digest()
        ).decode()
        logger.info("Made secret hash for %s: %s.", user_name, secret_hash)
        return secret_hash

    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
Cognito
        to send an email to the specified email address. The email contains a
code that
        can be used to confirm the user.

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

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
```

```
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery


    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
```

```
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
                self.user_pool_id,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users


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

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

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
 this
                 returns an access token that can be used to get AWS credentials.
 Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
 application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
 self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
```

```
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
not "
                        "configured for TOTP MFA. This example requires TOTP
MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


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

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            response.pop("ResponseMetadata", None)
            return response


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


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

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
```

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


    def confirm_mfa_device(
        self,
```

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

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

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


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
code.

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

```
        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
 {response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
```

```
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
 user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
 self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

创建运行场景的类。此示例还注册了一个 MFA 设备以通过 Amazon Cognito 进行跟踪，并向您演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》中的以下主题。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge

- SignUp
- VerifySoftwareToken

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包在 Amazon Cognito 用户身份验证后，使用 Lambda 函数编写自定义活动数据 AWS

以下代码示例显示了在完成 Amazon Cognito 用户身份验证后如何使用 Lambda 函数写入自定义活动数据。

- 使用管理员功能将用户添加到用户池。
- 配置用户池以调用 `PostAuthentication` 触发器的 Lambda 函数。
- 将新用户登录到 Amazon Cognito 控制台。
- Lambda 函数将自定义信息写入 CloudWatch 日志和 DynamoDB 表。
- 从 DynamoDB 表获取并显示自定义数据，然后清理资源。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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。

在命令提示符中运行交互式场景。

```
// ActivityLog separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ActivityLog struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
```

```
 cognitoActor *actions.CognitoActions
}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) ActivityLog {
 scenario := ActivityLog{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
 credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(userPoolId string, tableName string)
 (string, string) {
 log.Println("To facilitate this example, let's add a user to the user pool using
 administrator privileges.")
 users, err := runner.helper.GetKnownUsers(tableName)
 if err != nil {
  panic(err)
 }
 user := users.Users[0]
 log.Printf("Adding known user %v to the user pool.\n", user.UserName)
 err = runner.cognitoActor.AdminCreateUser(userPoolId, user.UserName,
 user.UserEmail)
 if err != nil {
  panic(err)
 }
 pwSet := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !pwSet {
  log.Printf("\nSetting password for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.AdminSetUserPassword(userPoolId, user.UserName,
 password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
```

```
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   pwSet = true
  }
 }

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

 return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
 PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(userPoolId string,
 activityLogArn string) {
 log.Println("Let's add a Lambda function to handle the PostAuthentication
 trigger from Cognito.\n" +
  "This trigger happens after a user is authenticated, and lets your function
 take action, such as logging\n" +
  "the outcome.")
 err := runner.cognitoActor.UpdateTriggers(
  userPoolId,
  actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
 aws.String(activityLogArn)})
 if err != nil {
  panic(err)
 }
 runner.resources.triggers = append(runner.resources.triggers,
 actions.PostAuthentication)
 log.Printf("Lambda function %v added to user pool %v to handle
 PostAuthentication Cognito trigger.\n",
  activityLogArn, userPoolId)

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

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(clientId string, userName string, password
 string) {
```

```
 log.Printf("Now we'll sign in user %v and check the results in the logs and the
 DynamoDB table.", userName)
 runner.questioner.Ask("Press Enter when you're ready.")
 authResult, err := runner.cognitoActor.SignIn(clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Println("Sign in successful.",
  "The PostAuthentication Lambda handler writes custom information to CloudWatch
 Logs.")

 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
 table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(tableName string, userName
 string) {
 log.Println("The PostAuthentication handler also writes login data to the
 DynamoDB table.")
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 users, err := runner.helper.GetKnownUsers(tableName)
 if err != nil {
  panic(err)
 }
 for _, user := range users.Users {
  if user.UserName == userName {
   log.Println("The last login info for the user in the known users table is:")
   log.Printf("\t%+v", *user.LastLogin)
  }
 }
 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *ActivityLog) Run(stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup()
  }
 }()
```

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

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

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

 runner.AddActivityLogTrigger(stackOutputs["UserPoolId"],
 stackOutputs["ActivityLogFunctionArn"])
 runner.SignInUser(stackOutputs["UserPoolClientId"], userName, password)
 runner.helper.ListRecentLogEvents(stackOutputs["ActivityLogFunction"])
 runner.GetKnownUserLastLogin(stackOutputs["TableName"], userName)

 runner.resources.Cleanup()

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

使用 Lambda 函数处理 PostAuthentication 触发器。

```
const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
 format.
type LoginInfo struct {
 UserPoolId string `dynamodbav:"UserPoolId"`
 ClientId   string `dynamodbav:"ClientId"`
 Time       string `dynamodbav:"Time"`
}
```

```
// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string    `dynamodbav:"UserName"`
 UserEmail string    `dynamodbav:"UserEmail"`
 LastLogin LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to
 the logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
 event events.CognitoEventUserPoolsPostAuthentication)
 (events.CognitoEventUserPoolsPostAuthentication, error) {
 log.Printf("Received post authentication trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName:  event.UserName,
  UserEmail: event.Request.UserAttributes["email"],
  LastLogin: LoginInfo{
   UserPoolId: event.UserPoolID,
   ClientId:   event.CallerContext.ClientID,
   Time:       time.Now().Format(time.UnixDate),
  },
 }
 // Write to CloudWatch Logs.
 fmt.Printf("%#v", user)

 // Also write to an external system. This examples uses DynamoDB to demonstrate.
 userMap, err := attributevalue.MarshalMap(user)
```

```
 if err != nil {
  log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
 } else if len(userMap) == 0 {
  log.Printf("User info marshaled to an empty map.")
 } else {
  _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
   Item:      userMap,
   TableName: aws.String(tableName),
  })
  if err != nil {
   log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
  } else {
   log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
  }
 }

 return event, nil
}

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

创建一个执行常见任务的结构。

```
// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(stackName string) (actions.StackOutputs, error)
 PopulateUserTable(tableName string)
 GetKnownUsers(tableName string) (actions.UserList, error)
 AddKnownUser(tableName string, user actions.User)
```

```
 ListRecentLogEvents(functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(stackName string)
 (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(tableName string) {
```

```
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(tableName string) (actions.UserList,
 error) {
 knownUsers, err := helper.dynamoActor.Scan(tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(tableName string, user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
```

```
 events, err := helper.cwlActor.GetLogEvents(functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

创建一个封装 Amazon Cognito 操作的结构。

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



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(userPoolId string,
 triggers ...TriggerInfo) error {
```

```
 output, err := actor.CognitoClient.DescribeUserPool(context.TODO(),
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(context.TODO(),
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(clientId string, userName string, password
 string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(context.TODO(),
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
```

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



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(clientId string, userName string, password
 string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(context.TODO(),
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

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



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(clientId string, code string,
 userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(context.TODO(),
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
```

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



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(userPoolId string, userName string,
 userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(context.TODO(),
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
```

```
func (actor CognitoActions) AdminSetUserPassword(userPoolId string, userName
 string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(context.TODO(),
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

创建一个封装 DynamoDB 操作的结构。

```
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
```

```
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(tableName string) error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(context.TODO(),
 &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
```

```
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(tableName string) (UserList, error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(context.TODO(), &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(tableName string, user User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(context.TODO(), &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

创建一个封装 CloudWatch 日志操作的结构。

```
type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
```

```
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(functionName string)
 (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(context.TODO(),
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(functionName string,
 logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(context.TODO(),
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

创建一个封装动作的结构。 AWS CloudFormation

```
// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(stackName string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(context.TODO(),
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

清理资源。

```
// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger
```

```
 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
```

```
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Go API 参考》中的以下主题。
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包进行 Amazon Cognito 同步的代码示例 AWS

以下代码示例展示了如何将 Amazon Cognito Sync 与 AWS 软件开发套件 (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。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

- 使用软件开发工具包进行 Amazon Cognito 同步的操作 AWS
  - ListIdentityPoolUsage与 AWS SDK 或 CLI 配合使用

使用软件开发工具包进行 Amazon Cognito 同步的操作 AWS

以下代码示例演示了如何使用软件开发工具包执行单个 Amazon Cognito 同步操作。 AWS 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的链接 GitHub，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

- ListIdentityPoolUsage与 AWS SDK 或 CLI 配合使用

`ListIdentityPoolUsage`与 AWS SDK 或 CLI 配合使用

以下代码示例演示了如何使用 `ListIdentityPoolUsage`。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。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?;

    let pools = response.identity_pool_usages();
    println!("Identity pools:");

    for pool in pools {
        println!(
            "  Identity pool ID:    {}",
            pool.identity_pool_id().unwrap_or_default()
        );
```

```
        println!(
            "  Data storage:        {}",
            pool.data_storage().unwrap_or_default()
        );
        println!(
            "  Sync sessions count: {}",
            pool.sync_sessions_count().unwrap_or_default()
        );
        println!(
            "  Last modified:       {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!();
    }

    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用ListIdentityPoolUsage于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

多租户应用程序最佳实践

Amazon Cognito 用户池与多租户应用程序一起运行，这些应用程序生成的大量请求必须保持在 Amazon Cognito 配额之内。要在客户群增长时扩大此容量，您可以购买额外的配额容量。

> **Note**
>
> Amazon Cognito 配额按和适用。 AWS 账户 AWS 区域这些配额在应用程序中的所有租户之间共享。查看 Amazon Cognito 服务配额，确保配额符合应用程序中的预期容量和预期的租户数量。

本节介绍您可以实施的方法来区分同一地区的 Amazon Cognito 资源和。 AWS 账户您也可以将租户分配到多个 AWS 账户 或多个区域，并为每个租户分配自己的配额。多区域多租户的其他优势包括尽可能高的隔离级别、为全球分布的用户提供最短的网络传输时间，以及遵守组织中现有的分发模式。

单区域多租户也可以为您的客户和管理员带来好处。

以下列表介绍了使用共享资源实现多租户的一些优势。

多租户的优势

常用用户目录

多租户支持客户在多个应用程序中拥有账户的模式。您可以将来自第三方提供商的身份关联到单个一致的用户池配置文件中。如果用户配置文件是其租户所独有的，则任何具有单用户池的多租户策略都有一个用户管理入口点。

常见安全性

在共享用户池中，您可以创建单一的安全标准，并将相同的高级安全、多因素身份验证 (MFA) AWS WAF和标准应用于所有租户。由于 AWS WAF Web ACL 必须与您关联的资源 AWS 区域 相同，因此多租户提供对复杂资源的共享访问权限。如果您想在多区域 Amazon Cognito 应用程序中保持一致的安全配置，则必须应用在资源之间复制配置的操作标准。

常用定制

您可以使用自定义用户池和身份池 AWS Lambda。用户池中的 Lambda 触发器和身份池中的 Amazon Cognito 事件的配置可能会变得复杂。Lambda 函数必须与您的用户池或身份池位于同一个 AWS 区域 中。共享 Lambda 函数可以强制执行区域内的自定义身份验证流程、用户迁移、令牌生成和其他功能的标准。

常用消息

亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 需要在某个区域进行额外配置，然后才能向用户发送 SMS 消息。您可以使用经亚马逊简单电子邮件服务 (Amazon SES) 认证的身份和区域内包含的域名发送电子邮件。

借助多租户，您可以在所有租户之间共享此配置和维护开销。由于 Amazon SNS 和 Amazon SES 并非全部可用 AWS 区域，因此在区域之间分配资源需要额外考虑。

使用自定义消息传递提供商时，您可以获得单个 Lambda 函数的通用自定义功能来管理您的消息传送。

托管用户界面在浏览器中设置会话 Cookie，以便识别已通过身份验证的用户。当您在用户池中对本地用户进行身份验证时，他们的会话 Cookie 会针对同一用户池中的所有应用程序客户端对他们进行身份验证。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。

有两种方法可以防止使用托管界面会话 Cookie 跨应用程序客户端登录。

- 将您的用户分成每个租户的用户池。
- 将托管用户界面登录替换为 Amazon Cognito 用户池 API 登录。

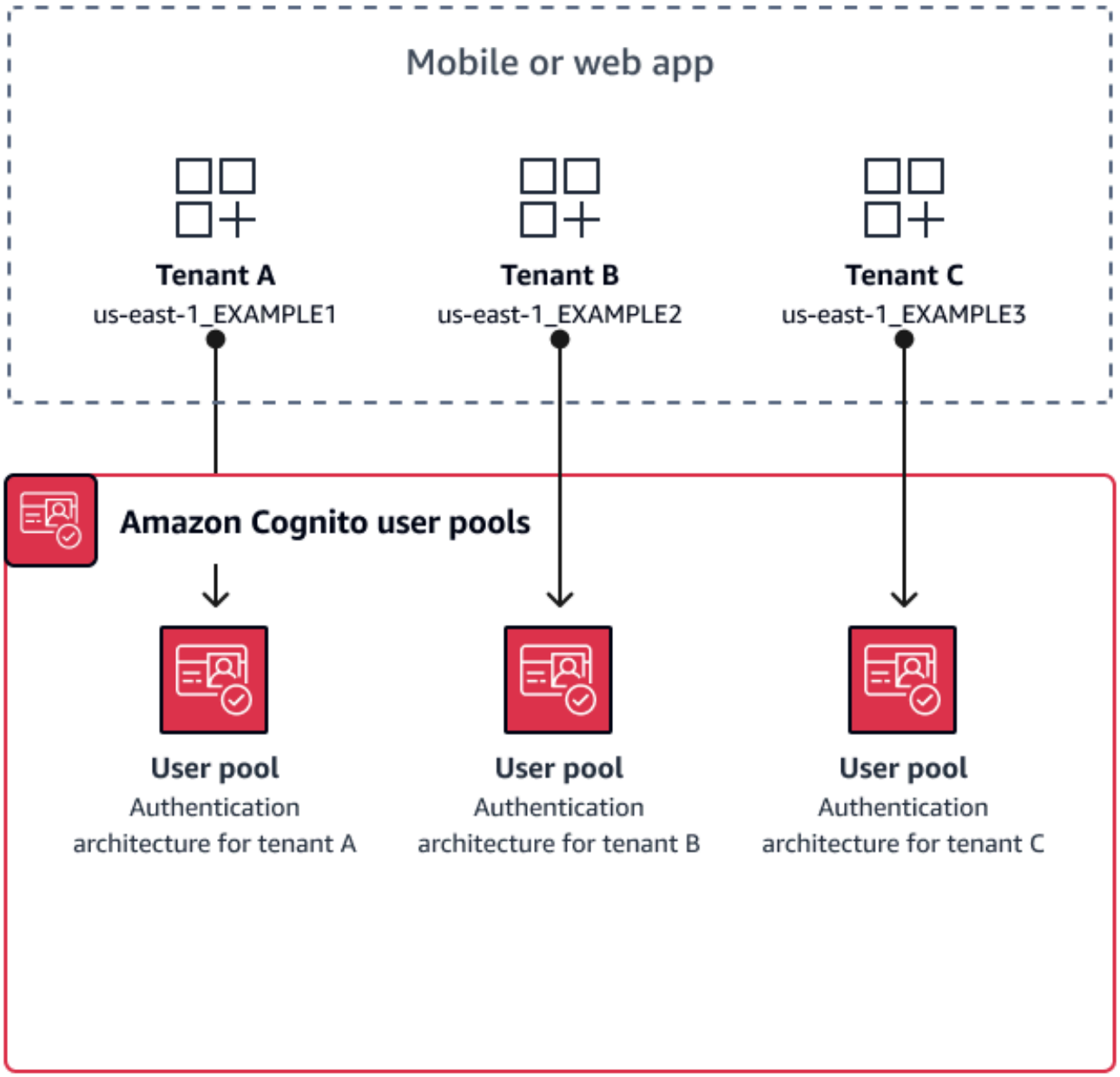
主题

用户池多租户最佳实践

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。通过用户池进行租户隔离使您可以灵活地进行 user-to-tenant 映射。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。

使用这种方法，您可以为每个租户单独设置托管用户界面，并将用户重定向到您的应用程序的租户特定实例。您也可以使用这种方法与 Amazon API Gatew ay 等后端服务集成。

下图显示每个租户都有一个专用用户池。



何时实施用户池多租户

当隔离和定制是您的主要关注点时。在具有多个用户池的架构中，用户和租户之间的关系可能很复杂。举一个例子，你有两个教育租户。同一个用户在一个应用程序中可能是访问受限的学生，而在另一个应用程序中可能是具有较高权限的教师。您可能需要在一个应用程序中使用 MFA，但不要求在另一个应

用程序中使用 MFA，或者有不同的密码策略。由于本地用户可以使用托管 UI 登录用户池中的多个应用程序客户端，因此，当您希望多个租户使用托管 UI 登录时，用户池多租户也是理想的选择。

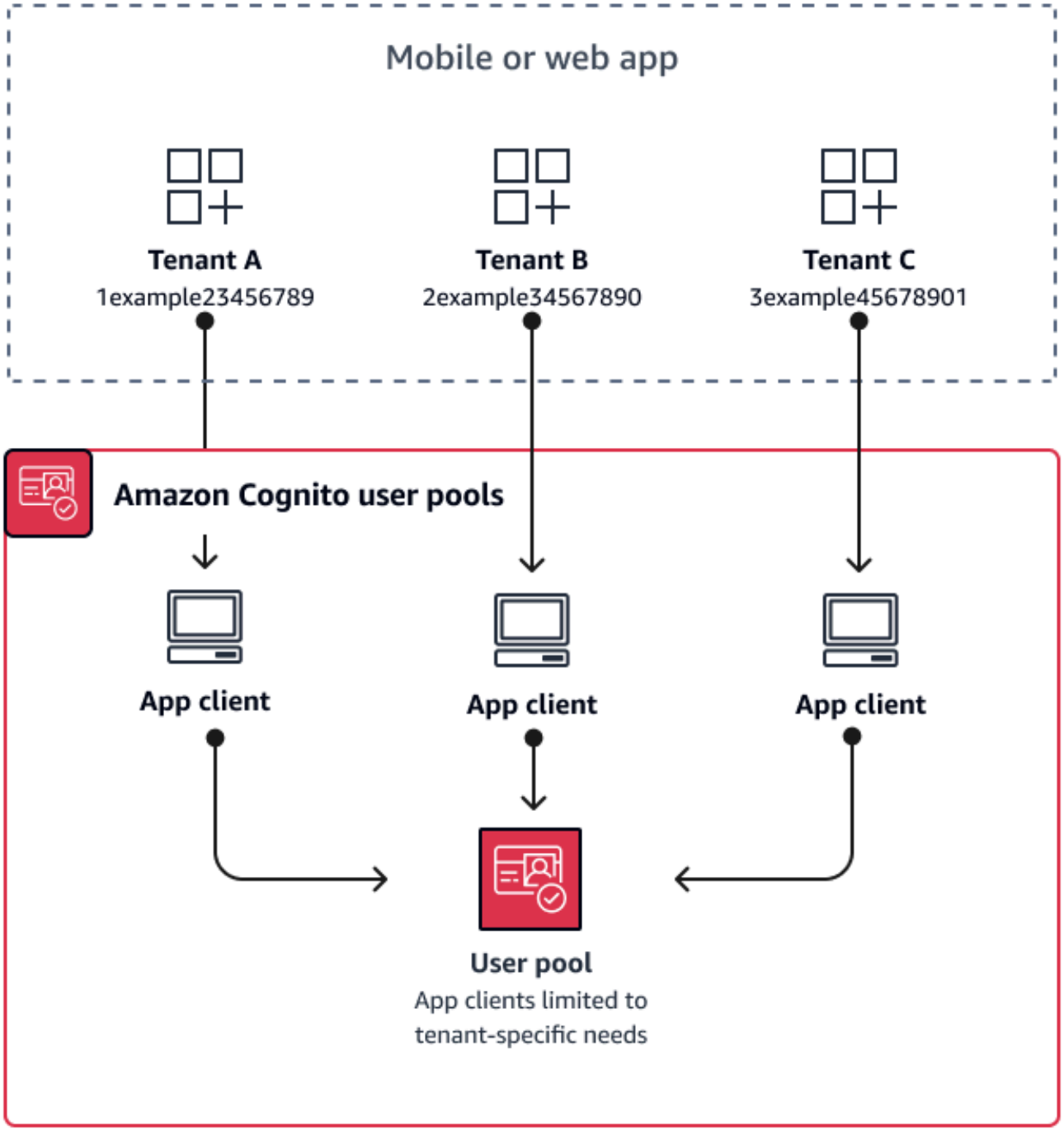
工作量水平

使用此方法的开发和操作工作量很大。为了确保您的应用程序系列获得一致且可预测的结果，您必须将 Amazon Cognito 资源与自动化工具集成，并在身份验证架构变得越来越复杂时保持基准。当你想为应用程序创建单一起点时，你必须构建用户界面 (UI) 元素来捕捉将用户引导到正确资源的初始决策。

应用程序-客户端多租户最佳实践

为应用程序中的每个租户创建一个应用程序客户端。借助应用程序-客户端多租户，您可以将任何用户分配给与租户关联的应用程序客户端，并保留单个用户个人资料。由于您可以将用户池中的任何或所有身份提供者 (IdPs) 分配给应用程序客户端，因此租户应用程序客户端可以允许使用租户特定的 IdP 登录。当用户存在于多个租户中时，您可以将他们的个人资料与多个租户关联起来， IdPs 以获得一致的用户体验。

下图显示了共享用户池中每个租户都有专用应用程序客户端。



何时实施应用程序-客户端多租户

何时您可以为用户池级别的设置选择通用配置，例如 Lambda 触发器、密码策略以及电子邮件和短信的内容和传递方式。由于共享用户池中的用户可以登录任何应用程序客户端，因此应用程序客户端多租

户非常适合使用或 A app-client-specific IdPs mazon Cognito 用户池 API 登录。App-Client 多租户也非常适合您希望允许用户在多个应用程序之间切换的 one-to-many环境。

工作量水平

应用程序-客户端多租户需要付出适度的努力。应用程序-客户端多租户面临的一个主要挑战是租户能否提供托管用户界面 Cookie 并在应用程序之间切换。在应用程序-客户端多租户架构中，在需要隔离的地方避免托管用户界面登录。您可以使用内置的应用程序客户端逻辑分发移动应用程序或指向 Web 应用程序的链接，也可以构建决定用户租约的初始界面元素。工作量较低，因为您无需跨多个用户池和身份池标准化和维护配置。

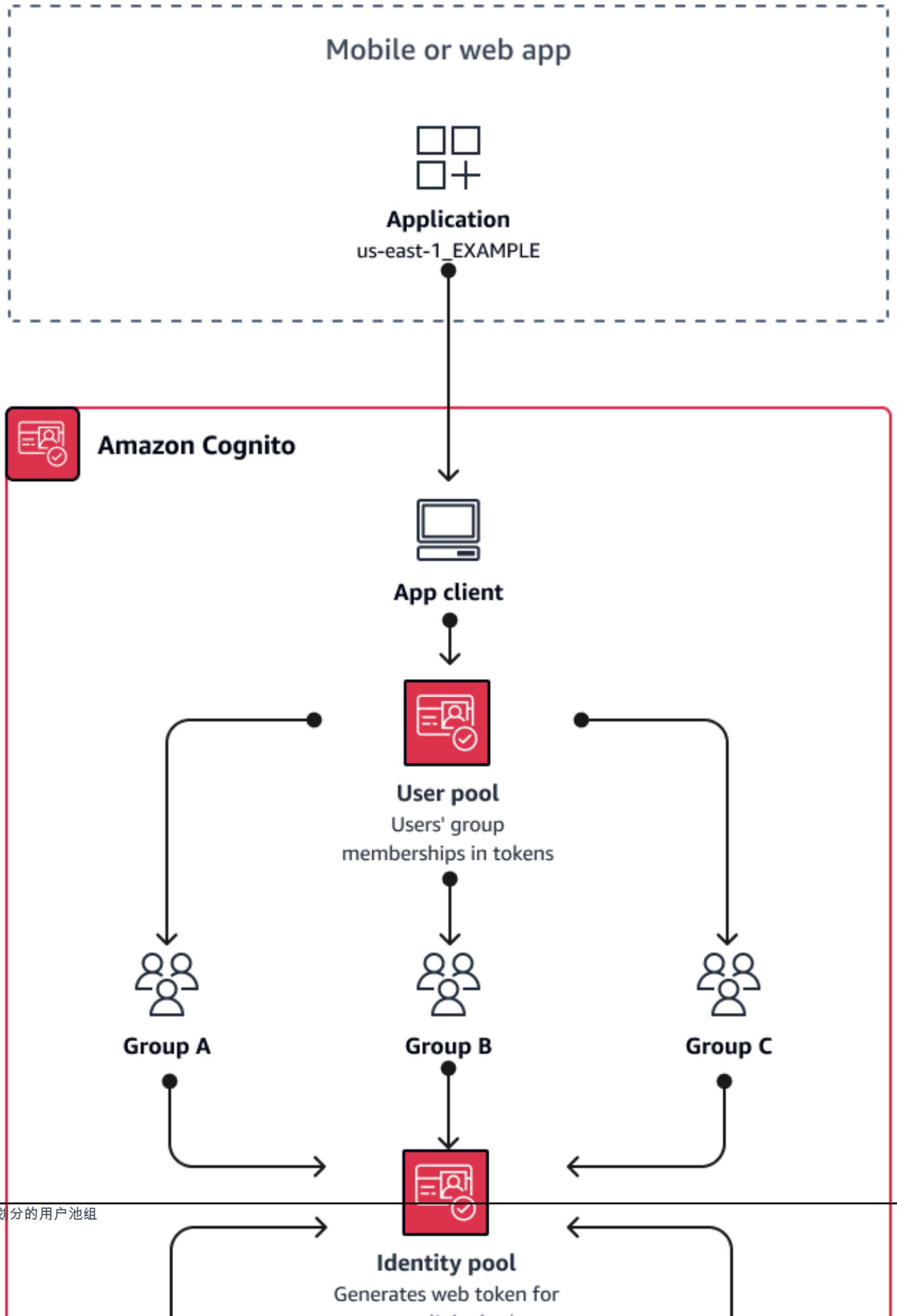
用户池组多租户最佳实践

当您的架构需要带有身份池的 Amazon Cognito 用户池时，基于群组的多租户效果最佳。

用户池 ID 和访问令牌包含`cognito:groups`声明。此外，ID 令牌包含`cognito:roles`和`cognito:preferred_role`声明。当您的应用程序中身份验证的主要结果是来自身份池的临时 AWS 证书时，您的用户的群组成员资格可以决定他们获得的 IAM 角色和权限。

例如，假设三个租户，每个租户都将应用程序资产存储在自己的 Amazon S3 存储桶中。将每个租户的用户分配到关联的群组，为该群组配置首选角色，并授予该角色对其存储桶的读取权限。

下图显示租户共享应用程序客户端和用户池，以及用户池中的专用群组来确定他们是否有资格担任 IAM 角色。


Mobile or web app
Application
us-east-1_EXAMPLE
Amazon Cognito
App client
User pool
Users' group
memberships in tokens
Group A
Group B
Group C
Identity pool
Generates web token for

何时实施群组多租户

当 AWS 资源访问是您的首要考虑时。Amazon Cognito 用户池用户池中的群组是一种基于角色的访问控制 (RBAC) 机制。您可以在用户池中配置多个群组，并根据群组优先级做出复杂的 RBAC 决策。身份池可以为优先级最高的角色、群组中的任何角色或用户令牌中的其他声明分配证书。

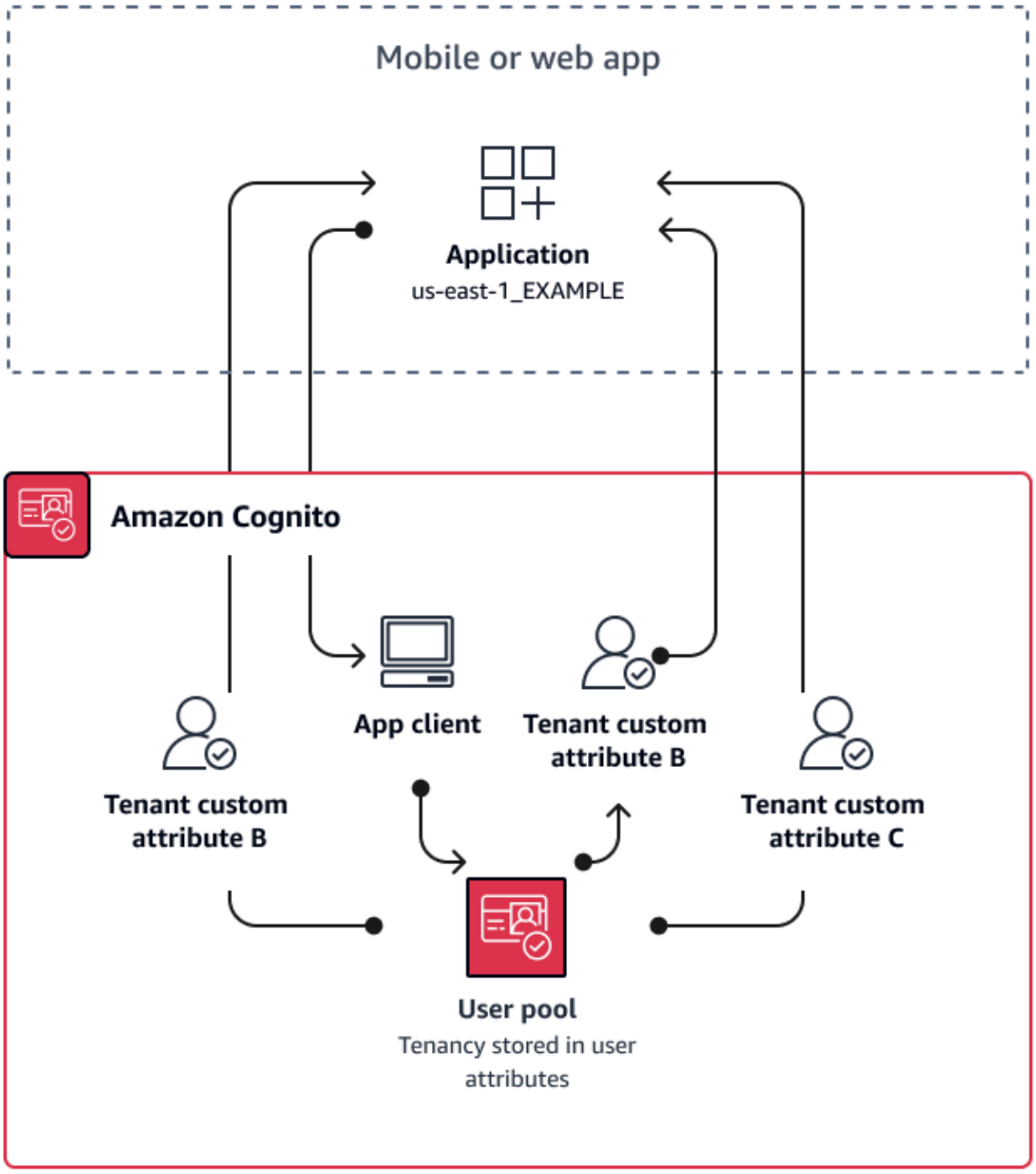
工作量水平

仅凭群组成员资格，维护多租户的工作量很低。但是，要将用户池组的角色扩展到内置的 IAM 角色选择容量之外，您必须构建应用程序逻辑来处理用户令牌中的群组成员资格，并确定要在客户端中执行的操作。您可以将 Amazon 验证权限与您的应用程序集成，以做出客户端授权决策。当前未在 Verified Permissions IsAuthorizedWithTokenAPI 操作中处理群组标识符，但您可以开发解析令牌内容的自定义代码，包括群组成员资格声明。

自定义属性多租户最佳实践

Amazon Cognito 支持使用您选择的名称的自定义属性。自定义属性在一种情况下很有用，那就是它们区分共享用户池中用户的租约。当你为用户分配类似的属性的值时`custom:tenantID`，你的应用程序可以相应地分配对租户特定资源的访问权限。定义租户 ID 的自定义属性对应用程序客户端来说应该是不可变的或只读的。

下图显示租户共享应用程序客户端和用户池，用户池中的自定义属性表示他们所属的租户。



当自定义属性决定租赁时，您可以分发单个应用程序或登录 URL。用户登录后，您的应用程序可以处理`custom:tenantID`索赔，确定要加载哪些资产、要应用的品牌以及要显示的功能。要根据用户属性

做出高级访问控制决策，请在 Amazon Verified Permissions 中将您的用户池设置为身份提供商，然后根据身份或访问令牌的内容生成访问决策。

何时实施自定义属性多租户

当租赁是表面层面时。租户属性可以为品牌和布局结果做出贡献。当你想在租户之间实现显著隔离时，自定义属性并不是最佳选择。必须在用户池或应用程序客户端级别配置的租户之间的任何区别（例如 MFA 或托管 UI 品牌）都要求您以自定义属性无法提供的方式在租户之间进行区分。借助身份池，您甚至可以从用户 ID 令牌中的自定义属性声明中选择 IAM 角色。

工作量水平

由于自定义属性多租户将基于租户的授权决策的职责转移到您的应用程序上，因此工作量往往很高。如果您已经精通解析 OIDC 声明的客户端配置或 Amazon Verified Permissions，那么这种方法可能需要最少的工作量。

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：

- 使用 Amazon 验证权限在您的应用程序中验证租约。在允许用户在应用程序中提出请求之前，请先制定政策，检查用户池、应用程序客户端、群组或自定义属性权限。 AWS 创建了经过验证的权限身份源，同时考虑了 Amazon Cognito 用户池。已验证的权限为多租户管理提供了其他指导。
- 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的自定义属性。只有在创建用户或用户在用户池中注册时，才能设置不可变属性的值。此外，向应用程序客户端授予对属性的只读访问权。
- 在租户的外部 IdP 和应用程序客户端之间使用 1:1 映射，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 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。身份池提供临时 AWS 证书，以授予您的用户访问其他人的权限 AWS 服务。

用户池是 Amazon Cognito 中的用户目录。您的应用程序用户可以直接通过用户池登录，也可以通过第三方身份提供商 (IdP) 进行联合。用户池管理处理通过Facebook、谷歌、亚马逊和苹果进行社交登录以及从OpenID Connect（OIDC）和SAML返回的代币的开销。 IdPs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，您的用户可以获得访问 AWS 服务（例如 Amazon S3 和 DynamoDB）的临时 AWS 证书。身份池支持匿名访客用户，也支持通过第三方进行联合 IdPs。

主题

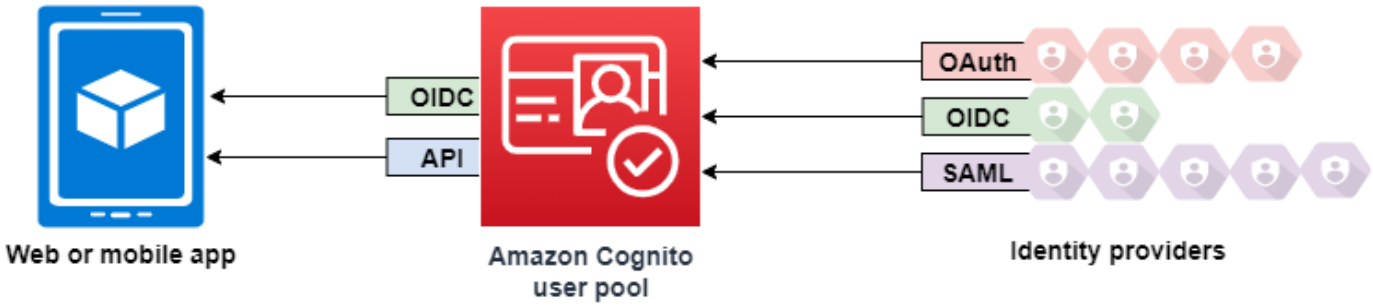
- 使用用户池进行身份验证
- 使用用户池访问您的服务器端资源
- 将 API Gateway 和 Lambda 与用户池结合使用来访问资源
- 使用用户池和身份池访问 AWS 服务
- 借助第三方进行身份验证并使用身份池访问 AWS 服务
- 使用 Amazon Cognito 访问 AWS AppSync 资源

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直接通过用户池登录，也可以通过第三方身份提供商 (IdP) 进行联合。用户池管理处理通过Facebook、谷歌、亚马逊和苹果进行社交登录以及从OpenID Connect（OIDC）和SAML返回的代币的开销。 IdPs

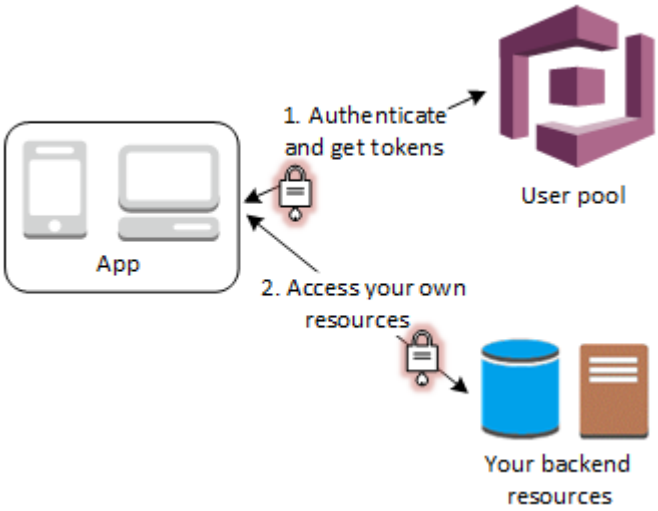
成功进行身份验证后，您的 Web 或移动应用程序将收到来自 Amazon Cognito 的用户池令牌。您可以使用这些令牌来检索允许您的应用程序访问其他 AWS 服务的 AWS 证书，也可以选择使用它们来控制对服务器端资源或 Amazon API Gateway 的访问。

有关更多信息，请参阅用户池身份验证流程和将令牌与用户池结合使用。



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。



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 使用资源服务器进行范围、M2M 和 API 授权。

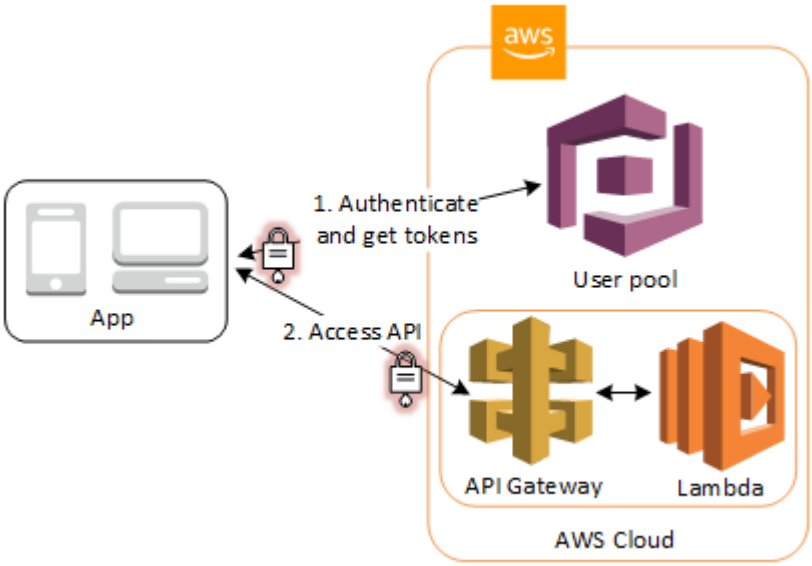
有关用户群体身份验证的更多信息，请参阅用户池身份验证流程和将令牌与用户池结合使用。

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。

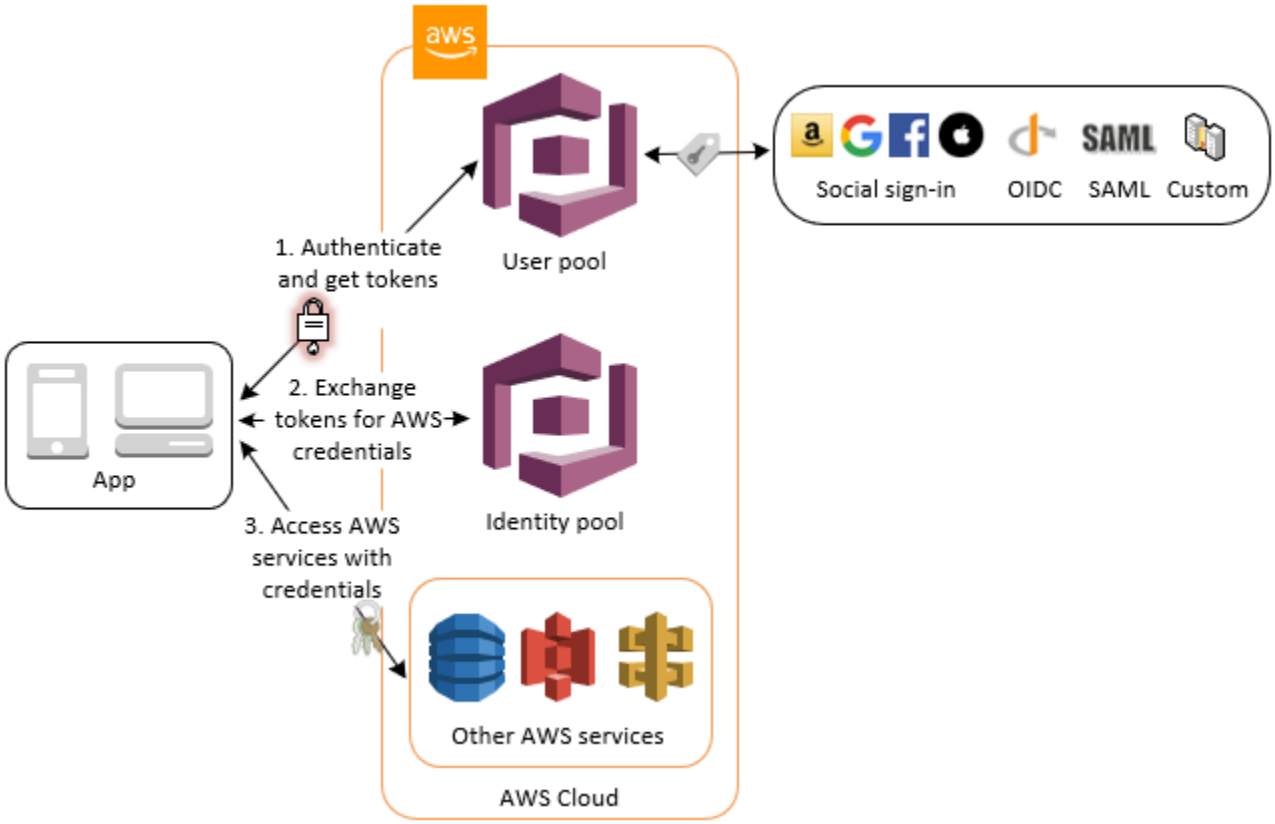
您可以使用用户池中的组控制对 API Gateway 的权限，方法是将组成员资格映射到 IAM 角色。用户所属的组包含在您的应用程序用户登录时用户池提供的 ID 令牌中。有关用户池组的更多信息，请参阅向用户池添加组。

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。



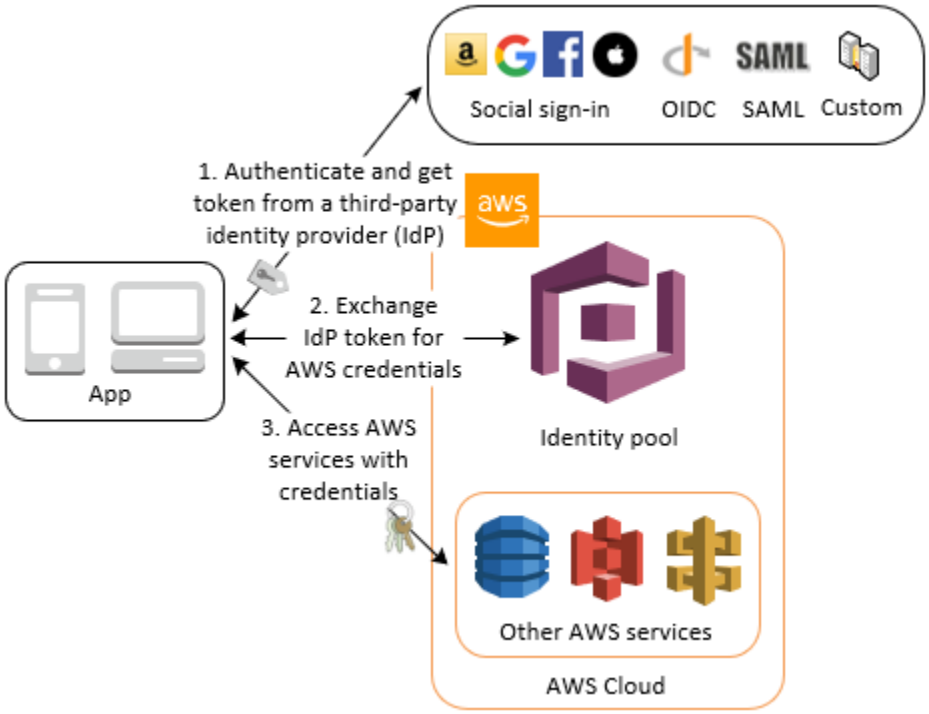
使用用户池和身份池访问 AWS 服务

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。您可以将它们交换为通过身份池临时访问其他 AWS 服务。有关更多信息，请参阅 登录后 AWS 服务 使用身份池进行访问 和 开始使用 Amazon Cognito 身份池。



借助第三方进行身份验证并使用身份池访问 AWS 服务

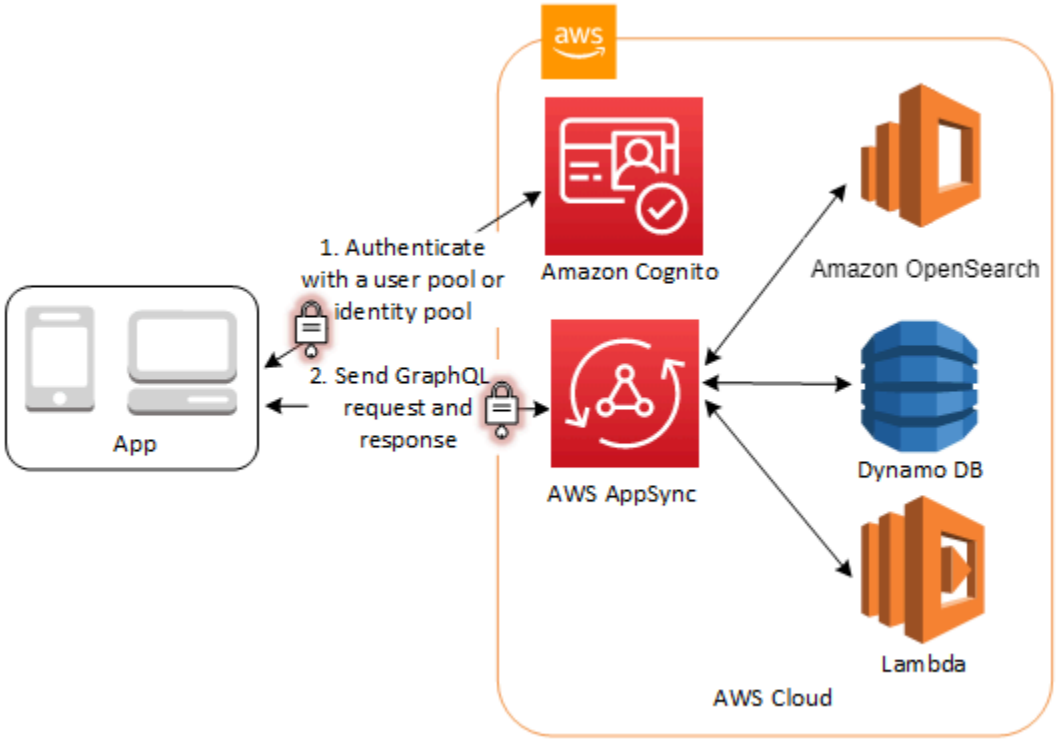
您可以允许您的用户通过身份池访问 AWS 服务。身份池需要来自由第三方身份提供商进行身份验证的用户的 IdP 令牌 (如果是匿名来访者，则不需要令牌)。作为交换，身份池会授予可用于访问其他 AWS 服务的临时 AWS 证书。有关更多信息，请参阅 开始使用 Amazon Cognito 身份池。



使用 Amazon Cognito 访问 AWS AppSync 资源

您可以通过成功的 Amazon Cognito 用户池身份验证获得的令牌向您的用户授予访问 AWS AppSync 资源的权限。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。

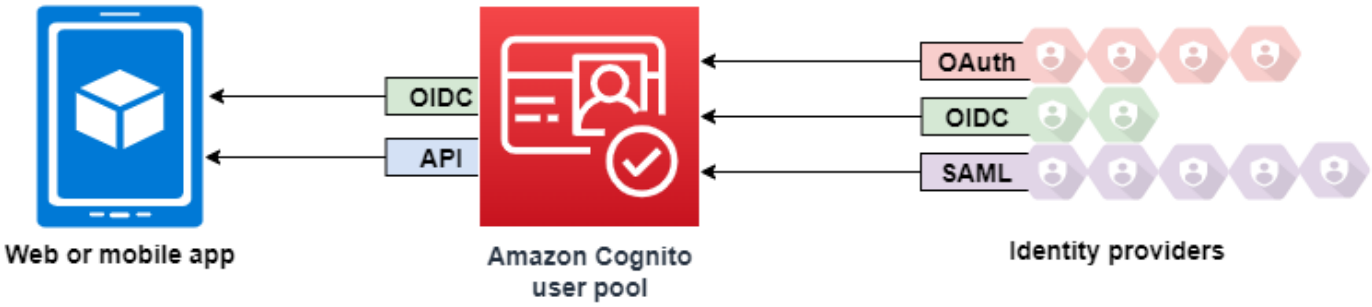
您还可以使用从身份池收到的 IAM 证书签署对 AWS AppSync GraphQL API 的请求。请参阅 AWS_IAM 授权。


aws
Amazon Cognito
1. Authenticate with a user pool or identity pool
2. Send GraphQL request and response
App
AWS AppSync
Amazon OpenSearch
Dynamo DB
Lambda
AWS Cloud

Amazon Cognito 用户群体

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，您可以选择最适合您的应用程序的 API 授权模式。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
- 使用用户池进行身份验证
- 使用 Amazon Cognito 用户池 API 和用户池端点
- 更新用户群体配置
- 设置和使用 Amazon Cognito 托管 UI 和联合身份验证端点
- 使用资源服务器进行范围、M2M 和 API 授权
- 通过第三方添加用户池登录
- 使用 Lambda 触发器自定义用户池工作流
- 将 Amazon Pinpoint 分析与 Amazon Cognito 用户池结合使用
- 管理用户池中的用户
- Amazon Cognito 用户池的电子邮件设置
- Amazon Cognito 用户池的短信设置
- 将令牌与用户池结合使用
- 在成功进行用户池身份验证后访问资源

- 使用 Amazon Cognito 用户池安全功能

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。

> **Important**
>
> 如果您在用户群体中激活用户注册，则互联网上的任何人都可以注册账户并登录您的应用程序。除非您开放您的应用程序供公开注册，否则不要在用户群体中启用自助注册。
> 要更改此设置，请在用户池控制台的 “注册体验” 选项卡中更新自助服务注册，或者更新 CreateUserPool或 UpdateUserPoolAPI AllowAdminCreateUserOnly请求中的值。
> 有关可以在用户群体中设置的安全特征的信息，请参阅使用 Amazon Cognito 用户池安全功能。

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，也可以使用客户密钥或 AWS 凭据对其进行授权。
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。运行一个 just-in-time AWS Lambda 函数，在现有目录中查找您的新用户，并根据现有数据填充他们的用户配置文件。

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

- 使用 Amazon Cognito 用户池 API 和用户池端点
- 使用软件开发工具包的 Amazon Cognito 身份提供商的代码示例 AWS

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
- 将联合用户与现有用户配置文件关联

M achine-to-machine 授权

有些会话不是 human-to-machine 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。要生成访问令牌以通过 OAuth 2.0 范围进行 machine-to-machine 授权，您可以添加生成客户端凭证授权的应用程序客户端。

相关 主题

- 使用资源服务器进行范围、M2M 和 API 授权

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 添加到现有网域中，也可以在 AWS 子域中使用前缀标识符。

相关 主题

- 设置和使用 Amazon Cognito 托管 UI 和联合身份验证端点
- 配置用户池域

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。

如果您的用户池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
- 向用户池添加高级安全
- 将 AWS WAF Web ACL 与用户池关联

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
- 自定义身份验证质询 Lambda 触发器
- 自定义内置登录网页和注册网页

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 L CloudWatch ogs 中查看性能指标， CloudWatch 使用 Lambda 触发器将自定义日志推送到，并在 Service Quotas 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

- 使用记录亚马逊 Cognito API 调用 AWS CloudTrail
- 跟踪和 Service Quotas 中的配额 CloudWatch 和使用情况
- 将 Amazon Pinpoint 分析与 Amazon Cognito 用户池结合使用

Amazon Cognito 身份池集成

Amazon Cognito 的另一半是身份池。身份池提供证书，用于授权和监控您的用户向（例如 Amazon DynamoDB 或 Amazon S3）发出的 API 请求。 AWS 服务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。

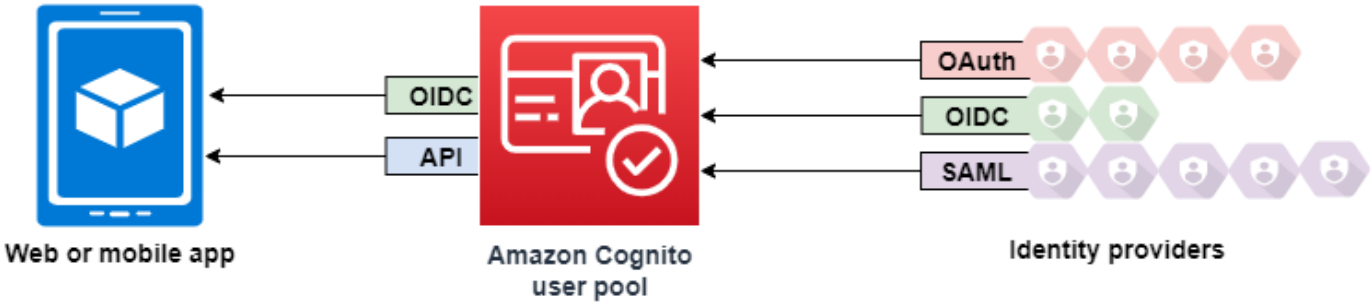
相关 主题

- 登录后 AWS 服务 使用身份池进行访问
- Amazon Cognito 身份池

使用用户池进行身份验证

您的应用程序用户可以直接通过用户池登录，也可以通过第三方身份提供商 (IdP) 进行联合。用户池管理处理通过Facebook、谷歌、亚马逊和苹果进行社交登录以及从OpenID Connect（OIDC）和SAML返回的代币的开销。 IdPs

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，您可以将它们交换为访问其他 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。有关安卓和 iOS 的软件开发工具包和示例代码的信息 JavaScript，请参阅 Amazon Cognito 用户池软件开发工具包。

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
- 用户池应用程序客户端
- 使用用户群体中的用户设备

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：

Public authentication

1. InitiateAuth
2. RespondToAuthChallenge

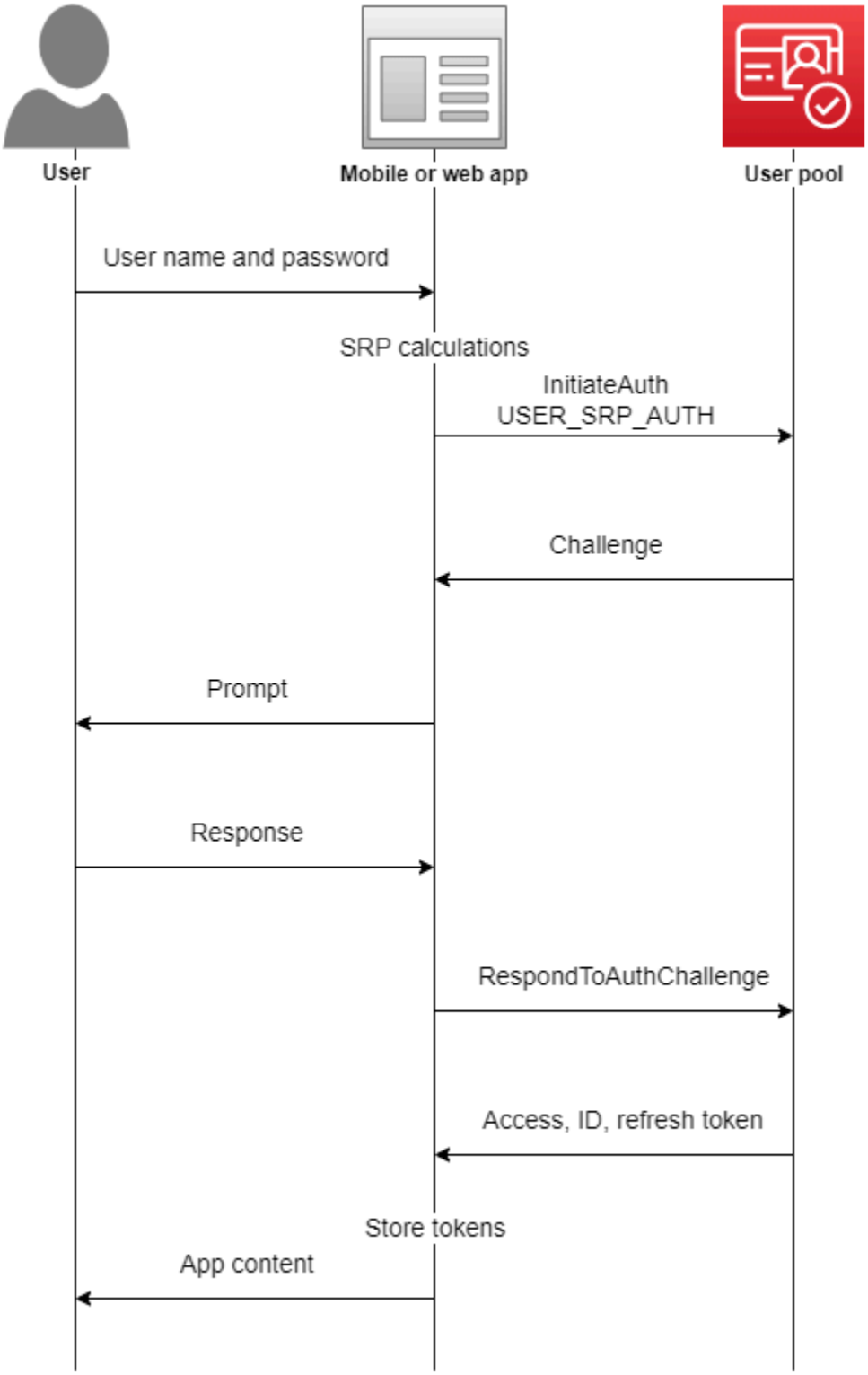
   `InitiateAuth` 和 `RespondToAuthChallenge` 是未经身份验证的 API，用于客户端公共应用程序客户端。

Server-side authentication

1. AdminInitiateAuth
2. AdminRespondToAuthChallenge

   `AdminInitiateAuth` 和 `AdminRespondToAuthChallenge` 需要 IAM 凭证，适用于服务器端机密应用程序客户端。

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。



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。

根据用户池的特征，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 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，请参阅用户池应用程序客户端。

您可以使用 AWS Lambda 触发器来自定义用户身份验证的方式。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

在使用密码进行五次未经身份验证或经 IAM 身份验证的登录尝试失败后，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 n=0 初始状态，用户必须在锁定期到期后才能成功登录，或者在锁定后连续 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
- 服务器端身份验证流程
- 自定义身份验证流程
- 内置身份验证流程和质询
- 自定义身份验证流程和质询
- 在自定义身份验证流程中使用 SRP 密码验证
- 管理员身份验证流程
- 用户迁移身份验证流程

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 特征生成 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`）。此操作需要具有包含`cognito-idp:AdminInitiateAuth`和权限的 AWS 证书`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`)。只有在您提供 AWS 凭证时，`AdminRespondToAuthChallenge`API 操作才会成功。

有关使用 AWS 凭证签署 Amazon Cognito API 请求的更多信息，请参阅AWS 一般参考中的签名版本 4 签名流程。

AdminInitiateAuth和 AdminRespondToAuthChallenge API 操作不能接受管理员登录的 username-and-password 用户凭据，除非您通过以下方式之一明确允许他们这样做：

- 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，请参阅用户池应用程序客户端。

自定义身份验证流程

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

> **Note**
>
> 您不能使用高级安全功能来处理泄露的凭据，也不能使用自定义身份验证流程的自适应身份验证。有关更多信息，请参阅向用户池添加高级安全。

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。这些 AWS 软件开发工具包内置了 Amazon Cognito 对这些流程的支持。

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。

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。

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 中所述的 API 操作，以使用 SRP 进行密码验证。S AWS DK 使用这种方法，这种方法可以帮助他们使用 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。

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。

用户池应用程序客户端

用户池应用程序客户端是用户池中的一项配置，它与一个通过 Amazon Cognito 进行身份验证的移动或 Web 应用程序进行交互。应用程序客户端可以调用经过授权和未经身份验证的 API 操作，并读取或修改用户的部分或全部属性。您的应用程序必须在操作中向应用程序客户端表明自己的身份，才能注册、登录和处理忘记的密码。这些 API 请求必须包括使用应用程序客户端 ID 进行自我识别以及使用可选客户端密钥进行授权的机制。您必须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。此外，如果您将应用程序配置为使用 AWS 凭证签署经过身份验证的 API 请求，则必须保护您的凭据免受用户检查。

您可以为一个用户池创建多个应用程序。应用程序客户端可能链接到应用程序的代码平台，也可能链接到用户池中的单独租户。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。

应用程序客户端类型

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 有关客户端类型的更多信息，请参阅 IETF RFC 6749 #2.1。

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

客户端机密或客户端密码是一个固定字符串，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。有关更多信息，请参阅 IETF RFC 6749 #2.3.1。

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`SECRET_HASH`在途中。有关更多信息，请参阅博客上的 “使用亚马逊 CloudFront 代理保护 Amazon Cognito 的公共客户端” AWS 。

JSON Web 令牌

Amazon Cognito 应用程序客户端可以发放以下类型的 JSON Web 令牌（JWT）。

身份（ID）令牌

一份可验证的声明，表明您的用户是从用户池进行的身份验证。OpenID Connect（OIDC）在 OAuth 2.0 定义的访问和刷新令牌标准中添加了 ID 令牌规范。ID 令牌包含身份信息，例如用户属

性，您的应用程序可以使用这些信息来创建用户个人资料和配置资源。请参阅使用 ID 令牌了解更多信息。

访问令牌

您的用户访问权限的可验证声明。访问令牌包含范围，这是 OIDC 和 OAuth 2.0 的一项特征。您的应用程序可以为后端资源提供范围，并证明您的用户池已授权用户或计算机访问来自 API 的数据或它们自己的用户数据。具有自定义范围的访问令牌（通常来自 M2M 客户端凭证授权）用于授权访问资源服务器。请参阅使用访问令牌了解更多信息。

刷新令牌

一种加密的初始身份验证声明，当您的用户令牌到期时，您的应用程序可以将其提供给您的用户池。刷新令牌请求会返回新的未到期访问令牌和 ID 令牌。请参阅使用刷新令牌了解更多信息。

您可以在 Amazon Cognito 控制台的用户池的应用程序集成选项卡中，为每个应用程序客户端设置这些令牌的到期时间。

应用程序客户端术语

以下术语是 Amazon Cognito 控制台中应用程序客户端的可用属性。

允许回调 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。

允许注销 URL

注销 URL 指示在您的用户注销后会被重定向到哪里。

属性读取和写入权限

您的用户群可能有很多客户，每个客户都有自己的应用程序客户端，并且 IdPs。您可以将应用程序客户端配置为仅对与应用程序相关的用户属性具有读写权限。在 machine-to-machine （M2M）授权之类的情况下，您可以不授予对任何用户属性的访问权限。

属性读取和写入权限配置的注意事项

- 当您创建应用程序客户端并且不自定义属性读写权限时，Amazon Cognito 会向所有用户池属性授予读写权限。
- 您可以授予对不可变自定义属性的写入权限。创建或注册用户时，您的应用程序客户端可以将值写入不可变属性。此后，您将无法为用户的任何不可变自定义属性写入值。
- 应用程序客户端必须拥有对用户池中必要属性的写入权限。Amazon Cognito 控制台会自动将必要属性设置为可写属性。
- 您不能允许应用程序客户端对 email_verified 或 phone_number_verified 拥有写入权限。用户池管理员可以修改这些值。用户只能通过属性验证来更改这些属性的值。

身份验证流程

您的应用程序客户端允许的登录方法。您的应用程序可以支持使用用户名和密码的身份验证、安全远程密码（SRP）、使用 Lambda 触发器的自定义身份验证以及令牌刷新。作为最佳安全实践，请使用 SRP 身份验证作为您的主要登录方法。托管 UI 会自动使用 SRP 将用户登录。

自定义范围

自定义范围是您在资源服务器中为自己的资源服务器定义的范围。格式为 *resource-server-identifier*/s *cop* e。请参阅 使用资源服务器进行范围、M2M 和 API 授权。

默认重定向 URI

将用户身份验证请求中的redirect_uri参数替换为第三方 IdPs。使用CreateUserPoolClient或UpdateUserPoolClientAPI 请求的DefaultRedirectURI参数配置此应用程序客户端设置。此 URL 还必须是您的应用程序客户端的成员。CallbackURLs在以下情况下，Amazon Cognito 将经过身份验证的会话重定向到此 URL：

1. 您的应用程序客户端分配了一个身份提供商并定义了多个回传 URL。如果身份验证请求不包含redirect_uri参数，则您的用户池会将对授权服务器的身份验证请求重定向到默认的重定向 URI。
2. 您的应用程序客户端分配了一个身份提供商并定义了一个回传 URL。在这种情况下，无需定义默认的回调 URL。不包含redirect_uri参数的请求会重定向到一个可用的回传网址。

身份提供者

您可以选择部分或全部用户池外部身份提供商 (IdPs) 来对用户进行身份验证。您的应用程序客户端还可以仅对用户池中的本地用户进行身份验证。当您将 IdP 添加到应用程序客户端时，可以生成指向 IdP 的授权链接，并将其显示在您的托管 UI 登录页面上。您可以分配多个 IdPs，但必须至少分配一个。有关使用外部的更多信息 IdPs，请参阅通过第三方添加用户池登录。

OpenID Connect 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。

- openid 范围声明您要检索 ID 令牌和用户的唯一 ID。它还会请求全部或部分用户属性，具体取决于请求中的其他范围。除非您请求 openid 范围，否则 Amazon Cognito 不会返回 ID 令牌。openid 范围授权结构化 ID 令牌声明，例如过期时间和密钥 ID，并确定您在 UserInfo 端点的响应中收到的用户属性。
  - 当 openid 是您请求的唯一范围时，Amazon Cognito 会使用当前应用程序客户端可以读取的所有用户属性填充 ID 令牌。对仅具有此范围的访问令牌的 userInfo 响应将返回所有用户属性。
  - 当您使用其他范围（例如 phone、email 或 profile）请求 openid 时，ID 令牌和 userInfo 返回用户的唯一 ID 以及由其他范围定义的属性。
- 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。
- 该aws.cognito.signin.user.admin范围允许访问需要访问令牌的 Amazon Cognito 用户池 API 操作，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。

有关范围的更多信息，请参阅标准 OIDC 范围列表。

OAuth 授权类型

OAuth 授权是一种检索用户池令牌的身份验证方法。Amazon Cognito 支持以下类型的授权。要将这些 OAuth 授权集成到您的应用程序中，您必须将域添加到您的用户池中。

授予授权代码

授权码授权会生成一个代码，您的应用程序可以用它与令牌端点交换用户池令牌。当您交换授权码时，您的应用程序会收到 ID、访问权限和刷新令牌。这种 OAuth 流程与隐式授权一样，都是在用

户的浏览器中进行的。授权码授权是 Amazon Cognito 提供的最安全的授权，因为令牌在用户的会话中不可见。相反，您的应用程序会生成返回令牌的请求，并可以将其缓存在受保护存储空间中。有关更多信息，请参阅 IETF RFC 6749 #1.3.1 中的授权码。

> **Note**
>
> 作为公共客户端应用程序的最佳安全实践，只激活授权码授权 OAuth 流程，并实施代码交换证明密钥（PKCE）以限制令牌交换。通过 PKCE，客户端只有在向令牌端点提供与原始身份验证请求中相同的机密时，才能交换授权码。有关 PKCE 的更多信息，请参阅 IETF RFC 7636。

隐式授予

隐式授权直接从对端点授权向用户的浏览器会话提供访问权限和 ID 令牌，但不返回刷新令牌。隐式授权消除了向令牌端点提出单独请求的要求，但与 PKCE 不兼容，也不会返回刷新令牌。该授权适用于无法完成授权码授权的测试场景和应用程序架构。有关更多信息，请参阅 IETF RFC 6749 #1.3.2 中的隐式授权。您可以在应用程序客户端中同时激活授权码授权和隐式授权，然后按需使用每个授权。

客户端凭证授权

客户端凭证授予用于 machine-to-machine (M2M) 通信。授权码和隐式授权向经过身份验证的人类用户发放令牌。客户端凭证授权非交互式系统对 API 的基于范围的授权。您的应用程序可以直接从令牌端点请求客户端凭证并接收访问令牌。有关更多信息，请参阅 IETF RFC 6749 #1.3.4 中的客户端凭证。您只能在具有客户端机密且不支持授权码或隐式授权的应用程序客户端中激活客户端凭证授权。

> **Note**
>
> 由于您没有以用户身份调用客户端凭证流程，因此该授权只能向访问令牌添加自定义范围。自定义范围就是您为自己的资源服务器定义的范围。默认范围（例如 `openid` 和 `profile`）不适用于非人类用户。
>
> 由于 ID 令牌是对用户属性的验证，因此它们与 M2M 通信无关，客户凭证授权也不会发放 ID 令牌。请参阅 使用资源服务器进行范围、M2M 和 API 授权。

客户凭证授予会增加您的 AWS 账单费用。有关更多信息，请参阅 Amazon Cognito 定价。

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，请输入您的 AWS 凭据。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择应用程序集成选项卡。
5. 选择应用程序客户端下的创建应用程序客户端。
6. 选择应用程序类型：公有客户端、机密客户端，或者其它。
7. 输入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身份验证流程。
10. 配置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 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。您可以选择现有的任何组合 IdPs。您可以单独使用您的用户池对用户进行身份验证，也可以使用您在用户池中配置的一个或多个第三方 IdPs 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。您可以通过用户池 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池属性模式的全部或有限子集。

16. 选择创建应用程序客户端。

17. 记下客户端 ID。这将识别注册和登录请求中的应用程序客户端。

AWS CLI

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

> **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"]"
```

有关更多信息，请参阅 AWS CLI 命令参考：create-user-pool-client

Amazon Cognito user pools API

生成 CreateUserPoolClientAPI 请求。必须为所有您不想设置为默认值的参数指定一个值。

更新用户池应用程序客户端（AWS CLI 和 AWS API）

在 AWS CLI，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 返回确认信息：

```
{
    "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
    }
}
```

有关更多信息，请参阅 AWS CLI 命令参考:update-user-pool-client.

AWS API: UpdateUserPoolClient

获取有关用户池应用程序客户端（AWS CLI 和 AWS API）的信息

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

有关更多信息，请参阅 AWS CLI 命令参考:describe-user-pool-client.

AWS API: DescribeUserPoolClient

列出用户池（AWS CLI 和 AWS API）中的所有应用程序客户端信息

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

有关更多信息，请参阅 AWS CLI 命令参考: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"
```

有关更多信息，请参阅 AWS CLI 命令参考：delete-user-pool-client

AWS API: DeleteUserPoolClient

使用用户群体中的用户设备

当您使用 Amazon Cognito 用户群体 API 登录本地用户群体用户时，您可以将来自高级安全特征的用户活动日志与用户的每台设备相关联，并且（可选）如果用户使用的是可信设备，也可以允许用户跳过

多重身份验证（MFA）。对于任何尚未包含设备信息的登录，Amazon Cognito 都会在响应中包含设备密钥。设备密钥的格式为 *Region_UUID*。借助设备密钥、安全远程密码（SRP）库和允许设备身份验证的用户群体，您可以提示应用程序中的用户信任当前设备，而不再在登录时提示输入 MFA 代码。

主题

设置记忆设备

借助 Amazon Cognito 用户群体，您可以将每个用户的设备与唯一的设备标识符（设备密钥）关联起来。当您在登录时提供设备密钥并执行设备身份验证时，您可以利用两个特征。

1. 借助高级安全特征，您可以监控特定设备上的用户活动，以实现安全和分析目的。当用户登录时，应用程序可以选择对每个用户及其设备进行身份验证，同时将设备信息添加到其活动日志中。
2. 记住设备功能还支持可信设备身份验证流程，其中，用户可以选择在与应用程序的安全要求相适应的时间段内不使用 MFA 即可登录。当您想要重新提示用户提交 MFA 代码时，可以更改其设备的记住状态。

   记住的设备只能在 MFA 处于活动状态的用户群体中覆盖 MFA。

当用户使用记住的设备登录时，您必须在其身份验证流程中执行额外的设备身份验证。有关更多信息，请参阅使用设备登录。

在用户群体的登录体验选项卡的设备跟踪下，将用户群体配置为记住设备。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

用户群体不会提示用户在登录时记住设备。

始终记住

当应用程序确认用户的设备时，用户群体将始终记住该设备，并且不会在将来成功登录设备时返回 MFA 质询。

用户选择加入

> 当应用程序确认用户的设备后，用户群体不会自动抑制 MFA 质询。您必须提示用户选择是否要记住设备。

当您选择始终记住或用户选择加入时，每次用户从身份不明的设备登录时，Amazon Cognito 都会生成设备标识符密钥和机密。设备密钥是应用程序在用户执行设备身份验证时发送到用户群体的初始标识符。

对于每个已确认的用户设备，无论是自动记住还是选择加入，您都可以在每次用户登录时使用设备标识符密钥和机密对设备进行身份验证。

您还可以在 CreateUserPool 或 UpdateUserPool API 请求中为用户群体配置记住设备设置。有关更多信息，请参阅 DeviceConfiguration 属性。

Amazon Cognito 用户群体 API 为记住的设备提供了额外的操作。

1. ListDevices 和 AdminListDevices 返回用户的设备密钥列表及其元数据。
2. GetDevice 和 AdminGetDevice 返回单个设备的设备密钥和元数据。
3. UpdateDeviceStatus 和 AdminUpdateDeviceStatus 将用户的设备设置为已记住或未记住。
4. ForgetDevice 和 AdminForgetDevice 将用户已确认的设备从其配置文件中移除。

名称以 `Admin` 开头的 API 操作用于服务器端应用程序，必须使用 IAM 凭证进行授权。有关更多信息，请参阅使用 Amazon Cognito 用户池 API 和用户池端点。

获取设备密钥

每当用户使用用户群体 API 登录并且身份验证参数中未包含设备密钥作为 `DEVICE_KEY` 时，Amazon Cognito 都会在响应中返回新的设备密钥。在公共客户端应用程序中，将设备密钥放在应用程序存储中，以便您可以将其包含在将来的请求中。在机密服务器端应用程序中，使用用户的设备密钥设置浏览器 Cookie 或其它客户端令牌。

应用程序必须确认设备密钥并提供其它信息，然后用户才能使用其可信设备登录。向 Amazon Cognito 生成一个 ConfirmDevice 请求，该请求使用设备密钥、友好名称、密码验证程序和盐来确认用户的设备。如果您将用户群体配置为选择加入设备身份验证，Amazon Cognito 会在响应您的 `ConfirmDevice` 请求时，提示用户必须选择是否记住当前设备。对用户在 UpdateDeviceStatus 请求中选择的内容进行响应。

当您确认用户的设备但未将其设置为记住的设备时，Amazon Cognito 会存储关联，但在您提供设备密钥时继续进行非设备登录。设备可以生成对用户安全和故障排除非常有用的日志。已确认但未记住的设备不会利用登录特征，但会利用安全监控日志特征。当您为应用程序客户端激活高级安全特征并将设备占用空间编码到请求中时，Amazon Cognito 会将用户事件与已确认的设备关联起来。

获取新的设备密钥

1. 使用 InitiateAuth API 请求开始用户的登录会话。
2. 使用 RespondToAuthChallenge 响应所有身份验证质询，直到您收到标记用户的登录会话完成的 JSON Web 令牌（JWT）。
3. 在应用程序中，记录 Amazon Cognito 在其 `RespondToAuthChallenge` 或 `InitiateAuth` 响应的 `NewDeviceMetadata` 中返回的值：`DeviceGroupKey` 和 `DeviceKey`。
4. 为用户生成新的 SRP 密钥：盐和密码验证程序。此功能可在提供 SRP 库的 SDK 中使用。
5. 提示用户输入设备名称，或根据用户的设备特征生成一个名称。
6. 在 ConfirmDevice API 请求中提供用户的访问令牌、设备密钥、设备名称和 SRP 密钥。如果用户群体设置为始终记住设备，则用户的注册已完成。
7. 如果 Amazon Cognito 对于 `ConfirmDevice` 响应了 `"UserConfirmationNecessary": true`，请提示您的用户选择是否要记住该设备。如果用户确认要记住设备，请使用用户的访问令牌、设备密钥和 `"DeviceRememberedStatus": "remembered"` 生成 UpdateDeviceStatus API 请求。
8. 如果您已指示 Amazon Cognito 记住该设备，那么当用户下次登录时，看到的不是 MFA 质询，而是 `DEVICE_SRP_AUTH` 质询。

使用设备登录

将用户的设备配置为记住后，当用户使用相同的设备密钥登录时，Amazon Cognito 不再要求用户提交 MFA 代码。设备身份验证仅用设备身份验证质询取代 MFA 身份验证质询。您不能仅使用设备身份验证登录用户。用户必须首先使用其密码或自定义质询完成身份验证。以下是在记住的设备上对用户进行身份验证的过程。

要在使用自定义身份验证质询 Lambda 触发器的流程中执行设备身份验证，请在 InitiateAuth API 请求中传递 `DEVICE_KEY` 参数。在用户成功完成所有质询并且 `CUSTOM_CHALLENGE` 质询返回的 `issueTokens` 值为 `true` 之后，Amazon Cognito 将返回一个最终 `DEVICE_SRP_AUTH` 质询。

使用设备登录

1. 从客户端存储中检索用户的设备密钥。

2. 使用 InitiateAuth API 请求开始用户的登录会话。选择一个 AuthFlow：USER_SRP_AUTH、REFRESH_TOKEN_AUTH、USER_PASSWORD_AUTH 或 CUSTOM_AUTH。在 AuthParameters 中，将用户的设备密钥添加到 DEVICE_KEY 参数中，并包括所选登录流程所需的其它参数。

   a. 您还可以在对身份验证质询的 PASSWORD_VERIFIER 响应的参数中传递 DEVICE_KEY。

3. 完成质询响应，直到您在响应中收到 DEVICE_SRP_AUTH 质询。
4. 在 RespondToAuthChallenge API 请求中，发送 DEVICE_SRP_AUTH 的 ChallengeName 以及 USERNAME、DEVICE_KEY 和 SRP_A 的参数。
5. Amazon Cognito 以 DEVICE_PASSWORD_VERIFIER 质询进行响应。此质询响应包括 SECRET_BLOCK 和 SRP_B 的值。
6. 使用您的 SRP 库，生成并提交 PASSWORD_CLAIM_SIGNATURE、PASSWORD_CLAIM_SECRET_BLOCK、TIMESTAMP、USERNAME 和 DEVICE_KEY 参数。在其它 RespondToAuthChallenge 请求中提交这些内容。
7. 完成其它质询，直到收到用户的 JWT。

以下伪代码演示如何计算 DEVICE_PASSWORD_VERIFIER 质询响应的值。

```
PASSWORD_CLAIM_SECRET_BLOCK = SECRET_BLOCK
TIMESTAMP = Tue Sep 25 00:09:40 UTC 2018
PASSWORD_CLAIM_SIGNATURE = Base64(SHA256_HMAC(K_USER, DeviceGroupKey + DeviceKey +
 PASSWORD_CLAIM_SECRET_BLOCK + TIMESTAMP))
K_USER = SHA256_HASH(S_USER)
S_USER = (SRP_B - k * gx)(a + ux)
x = SHA256_HASH(salt + FULL_PASSWORD)
u = SHA256_HASH(SRP_A + SRP_B)
k = SHA256_HASH(N + g)
```

查看、更新和忘记设备

您可以使用 Amazon Cognito API 在应用程序中实现以下特征。

1. 显示有关用户的当前设备的信息。
2. 显示用户的所有设备的列表。
3. 忘记设备。
4. 更新设备记住的状态。

授权以下描述中的 API 请求的访问令牌必须包含 `aws.cognito.signin.user.admin` 范围。Amazon Cognito 会将此范围的声明添加到您使用 Amazon Cognito 用户群体 API 生成的所有访问令牌中。第三方 IdP 必须为向 Amazon Cognito 进行身份验证的用户单独管理设备和 MFA。在托管 UI 中，您可以请求 `aws.cognito.signin.user.admin` 范围，但是托管 UI 会自动将设备信息添加到高级安全用户日志中，而不提供记住设备的功能。

显示有关设备的信息

您可以查询有关用户设备的信息，以确定设备当前是否仍在使用中。例如，您可能希望在记住的设备已有 90 天未登录后将其停用。

- 要在公共客户端应用程序中显示用户的设备信息，请在 GetDevice API 请求中提交用户的访问密钥和设备密钥。
- 要在机密客户端应用程序中显示用户的设备信息，请使用 AWS 凭证签署 AdminGetDevice API 请求，然后提交用户的用户名、设备密钥和用户群体。

显示用户的所有设备的列表

您可以显示用户的所有设备及其属性的列表。例如，您可能要验证当前设备是否与记住的设备相匹配。

- 在公共客户端应用程序中，在 ListDevices API 请求中提交用户的访问令牌。
- 在机密客户端应用程序中，使用 AWS 凭证签署 AdminListDevices API 请求，然后提交用户的用户名和用户群体。

忘记设备

您可以删除用户的设备密钥。当您确定您的用户不再使用设备时，或者当您检测到异常活动并希望提示用户再次完成 MFA 时，您可能需要这样做。要稍后再次注册设备，必须生成并存储新的设备密钥。

- 在公共客户端应用程序中，在 ForgetDevice API 请求中提交用户的设备密钥和访问令牌。
- 在机密客户端应用程序中，在 AdminForgetDevice API 请求中提交用户的设备密钥和访问令牌。

使用 Amazon Cognito 用户池 API 和用户池端点

当您想要注册、登录和管理用户池中的用户时，有两种选择。

1. 您的用户池端点包括托管 UI 和联合身份验证端点。它们构成了一个公共网页包，当您为用户池选择域时，Amazon Cognito 会激活这些网页。要快速开始使用 Amazon Cognito 用户池的身份验证和授

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。其他用户池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 用户池 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。用户池 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**
>
> 默认情况下，来自用户池 API 身份验证的访问令牌仅包含 `aws.cognito.signin.user.admin` 作用域。要生成具有额外作用域的访问令牌（例如，授权对第三方 API 的请求），请在通过用户池端点进行身份验证期间请求作用域，或者在 令牌生成前 Lambda 触发器 中添加自定义作用域。自定义访问令牌会增加您的 AWS 账单费用。

您可以将通常通过用户池端点登录的联合用户与其配置文件位于用户池本地 的用户相关联。本地用户仅存在于您的用户池目录中，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池 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。

Amazon Cognito 用户池 API 有双重用途。它创建和配置您的 Amazon Cognito 用户池资源。例如，您可以创建用户池、添加 AWS Lambda 触发器以及配置您的托管 UI 域。用户池 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 用户池 API 的示例场景

1. 用户选择了您在应用程序中创建的“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，并在用户池中创建新用户。
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 用户池 API 中，您无法登录通过 IdP 进行联合身份验证的用户。您必须通过用户池端点对这些用户进行身份验证。有关包含托管 UI 的用户池端点的信息，请参阅用户池联合身份验证端点和托管 UI 参考。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 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。

用户池端点示例场景

1. 用户选择了您在应用程序中创建的“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 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 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。

用户池 API 和用户池端点支持各种场景，如本指南中所述。以下部分探讨了用户池 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 用户池经过身份验证和未经身份验证的 API 操作

Amazon Cognito 用户池 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 和 经过 IAM 身份验证的用户操作 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池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池，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 用户池 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

经过 IAM 身份验证的管理操作

DeleteUserPoolDomain

DescribeIdentityProvider

DescribeResourceServer

DescribeRiskConfiguration

DescribeUserImportJob

DescribeUserPool

DescribeUserPoolClient

DescribeUserPoolDomain

GetCSVHeader

GetGroup

GetIdentityProviderByIdentifier

GetSigningCertificate

GetUICustomization

GetUserPoolMfaConfig

ListGroups

ListIdentityProviders

ListResourceServers

ListTagsForResource

ListUserImportJobs

ListUserPoolClients

ListUserPools

经过 IAM 身份验证的管理操作

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 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 用户池 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。

经过 IAM 身份验证的用户操作

`AdminAddUserToGroup`

`AdminConfirmSignUp`

`AdminCreateUser`

`AdminDeleteUser`

`AdminDeleteUserAttributes`

`AdminDisableProviderForUser`

`AdminDisableUser`

`AdminEnableUser`

`AdminForgetDevice`

`AdminGetDevice`

`AdminGetUser`

`AdminInitiateAuth`

`AdminLinkProviderForUser`

`AdminListDevices`

`AdminListGroupsForUser`

`AdminListUserAuthEvents`

经过 IAM 身份验证的用户操作

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 用户池 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。

未经身份验证的用户操作

SignUp

ConfirmSignUp

未经身份验证的用户操作

ResendConfirmationCode

ForgotPassword

ConfirmForgotPassword

InitiateAuth

经过令牌授权的用户操作

经过令牌授权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，请使用经过令牌授权的 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授权的 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，授权其经过令牌授权的 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 用户池 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。

要签署经过令牌授权的操作的 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>*。

经过令牌授权的用户操作	AccessToken	会话
RespondToAuthChallenge		✓
ChangePassword	✓	
GetUser	✓	

经过令牌授权的用户操作	AccessToken	会话
UpdateUserAttributes	✓	
DeleteUserAttributes	✓	
DeleteUser	✓	
ConfirmDevice	✓	
ForgetDevice	✓	
GetDevice	✓	
ListDevices	✓	
UpdateDeviceStatus	✓	
GetUserAttributeVerificationCode	✓	
VerifyUserAttribute	✓	
SetUserSettings	✓	
SetUserMFAPreference	✓	
GlobalSignOut	✓	
AssociateSoftwareToken	✓	✓
UpdateAuthEventFeedback		✓

经过令牌授权的用户操作	AccessToken	会话
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

要在中更改 Amazon Cognito 用户池的设置 AWS Management Console，请浏览用户池设置中基于功能的选项卡，并按照本指南其他部分所述更新字段。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

API 参数名称：PoolName

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

API 参数名称：AliasAttributes和 UsernameAttributes

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）

API 参数名称：UsernameConfiguration

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。要更改区分大小写，请创建新的用户群体。

客户端密钥

API 参数名称：GenerateSecret

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 用户池应用程序客户端。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。

必需的属性

API 参数名称：架构

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。要更改必需的属性，请创建新的用户群体。

自定义属性

API 参数名称：架构

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。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。

短信配置

在用户池中激活 SMS 消息后，您无法将其停用。

- 如果您在创建用户池时选择配置短信，则在完成设置后无法停用 SMS。
- 您可以在自己创建的用户池中激活 SMS 消息，但之后就无法停用 SMS。
- Amazon Cognito 可以使用短信进行用户账户邀请和恢复、属性验证和多因素身份验证 (MFA)。激活 SMS 消息后，您可以随时为这些功能开启或关闭 SMS 消息。
- 短信配置包括您委托给 Amazon Cognito 的 IAM 角色，用于通过亚马逊 SNS 发送消息。您可以随时更改分配的角色。

使用 AWS SDK 或 REST API 更新用户池 AWS CDK

在 Amazon Cognito 控制台中，您可以一次更改一个参数的用户池设置。例如，要添加 Lambda 触发器，您可以选择添加 Lambda 触发器，然后选择函数和触发器类型。Amazon Cognito 用户池 API 的结构是，用户池和应用程序客户端的更新操作需要用户池的全套参数。但是，控制台会使用您的其他用户池设置透明地自动执行此更新操作。

有时你可能会发现，当更新与你要更改的设置无关时，你 AWS 账户 可能会发现其他地方的更改可能会导致更新生成错误。例如，已删除的 Amazon SES 身份或 IAM 权限的 AWS WAF更改。如果其中一

个当前参数不再有效，则在修复该参数之前，您无法更新设置。遇到此类错误时，请检查错误响应并验证其提及的设置。

Amazon Cognito 用户池 REST API 和AWS 软件开发工具包是用于自动化和编程配置 Amazon Cognito 资源的工具。AWS Cloud Development Kit (AWS CDK)使用这些工具的请求还必须像 Amazon Cognito 控制台一样，在请求正文中使用完整的资源配置来更新设置。总体而言，您必须执行以下过程。

1. 从描述现有资源配置的操作中捕获输出。
2. 更改设置后修改输出。
3. 在更新资源的操作中发送修改后的配置。

以下过程使用 UpdateUserPoolAPI 操作更新您的配置。同样的方法适用于不同的输入字段 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. 使用捕获用户池的现有状态 DescribeUserPool。
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 UpdateUserPoolDomainAPI 操作更新此字段。
   - `Domain`
     - 使用 UpdateUserPoolDomainAPI 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 命令为接受的输入字段生成空值的 JSON。update-user-pool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
```

设置和使用 Amazon Cognito 托管 UI 和联合身份验证端点

带有域的 Amazon Cognito 用户池是符合 OAuth-2.0 标准的授权服务器和用于身份验证的 ready-to-use 托管用户界面 (UI)。授权服务器路由身份验证请求，发布和管理 JSON Web 令牌（JWT），并提供用户属性信息。托管 UI 是一系列 Web 界面，用于处理用户池中的基本注册、登录、多重身份验证和密码重置活动。它也是您与应用程序关联的第三方身份提供商 (IdPs) 进行身份验证的中心中心。当您想要对用户进行身份验证和授权时，应用程序可以调用托管 UI 和授权端点。您可以通过自己的徽标和 CSS 自定义来使托管 UI 用户体验适合您的品牌。有关托管 UI 和授权服务器的组件的更多信息，请参阅用户池联合身份验证端点和托管 UI 参考。

> **Note**
>
> Amazon Cognito 托管 UI 不支持使用自定义身份验证质询 Lambda 触发器进行自定义身份验证。

主题

使用设置托管用户界面 AWS Amplify

如果您使用 AWS Amplify 向 Web 或移动应用程序添加身份验证，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，在您的客户端代码中，您可以使用这些 AWS Amplify 库对 Amazon Cognito 用户池中的用户进行身份验证。

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 端点联合用户身份。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。

以下示例说明 AWS Amplify 如何使用在您的应用程序中使用社交提供商设置托管界面。

- AWS Amplify 的身份验证 JavaScript。
- AWS Amplify Swift 的身份验证。
- AWS Amplify Flutter 的身份验证。
- AWS Amplify 安卓版身份验证。

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，请输入您的 AWS 凭据。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。

4. 选择应用程序集成选项卡。
5. 选择应用程序客户端下的创建应用程序客户端。
6. 选择应用程序类型：公有客户端、机密客户端，或者其它。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。机密客户端通常通过您信任的中央服务器上的应用程序运行，该应用程序具有客户端密钥和 API 凭据，并使用授权标头和 AWS Identity and Access Management 凭据对请求进行签名。如果您的使用案例与预配置的应用程序的公有客户端或机密客户端应用程序设置不同，请选择其它。
7. 输入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身份验证流程。
9. 配置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 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. 选择创建。
16. 记下客户端 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. 在应用程序集成选项卡的应用程序客户端下，选择您的应用程序客户端。检查当前托管 UI信息。
2. 在允许的回调 URL 下选择添加回调 URL。回调 URL 是在用户成功登录之后将被重新导向的地址。
3. 在允许的注销 URL 下选择添加注销 URL。注销 URL 是在您的用户注销后会被重新导向的地址。
4. 从身份提供商列表中添加至少一个列出的选项。
5. 在 OAuth 2.0 授予类型下，选择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 授予类型下，选择隐式授予，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. 您可以同时启用授权代码和隐式代码，然后按需使用每个授予。如果未选中授权代码或隐式代码授权，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，才选择客户端凭证。
8. 选择您要为此应用程序客户端授权的 OpenID Connect 范围。
9. 选择保存更改。

配置域

1. 导航到用户池的应用程序集成选项卡。
2. 选择在域旁边的操作，然后选择创建自定义域或创建 Cognito 域。如果您已配置用户池域，请先选择删除 Cognito 域或删除自定义域，然后再创建新的自定义域。
3. 输入可用的域前缀，将其与 Cognito 域结合使用。有关设置自定义域的信息，请参阅将您自己的域用于托管 UI
4. 选择创建。

查看您的登录页面

在 Amazon Cognito 控制台中，请在应用程序集成选项卡中的应用程序客户端和分析下，选择应用程序客户端配置中的 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，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，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 用户池联合身份验证端点和托管 UI 参考。

托管 UI 登录网页使用以下 URL 格式。此示例使用 `response_type=code` 参数请求授权码授予。

```
https://<your domain>/oauth2/authorize?response_type=code&client_id=<your app client
 id>&redirect_uri=<your callback url>
```

您可以从应用程序集成选项卡中检索您的用户池域字符串。在同一个选项卡中，您可以在应用程序客户端和分析下识别应用程序客户端 ID、其回调 URL、其允许的范围和其他配置。

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。

关于隐式授予的托管 UI 请求示例

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

身份令牌和访问令牌显示为附加到您的重定向 URL 的参数。

以下内容是来自隐式授予请求的示例响应。

```
            https://mydomain.example.com/
#id_token=eyJraaBcDeF1234567890&access_token=eyJraGhIjKlM1112131415&expires_in=3600&token_type=
```

关于 Amazon Cognito 用户池托管 UI 需要了解的事项

托管 UI 和确认用户为管理员

对于用户池本地用户，当您将用户池配置为允许 Cognito 自动发送消息以进行验证和确认时，托管 UI 效果最佳。当您启用此设置时，Amazon Cognito 会向注册的用户发送一条包含确认码的消息。当您改而确认用户为用户池管理员时，在注册后托管 UI 会显示一条错误消息。在这种状态下，Amazon Cognito 已创建新用户，但无法发送验证消息。您仍然可以确认用户为管理员，但他们可能会在遇到错误后联系您的支持部门。有关管理确认的更多信息，请参阅允许用户在您的应用程序中注册但以用户池管理员身份进行确认。

查看您对托管 UI 配置所做的更改

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。

解码用户池令牌

使用 RS256 算法对 Amazon Cognito 用户池令牌进行签名。您可以使用解码和验证用户池令牌 AWS Lambda，请参阅解码和验证 Amazon Cognito JW T 令牌。 GitHub

托管用户界面和 TLS 版本

托管 UI 需要在传输过程中进行加密。Amazon Cognito 提供的用户池域要求的最低 TLS 版本为 1.2。自定义域名支持但不需要 TLS 版本 1.2。由于 Amazon Cognito 管理托管用户界面和授权服务器终端节点的配置，因此您无法修改用户池域的 TLS 要求。

托管 UI 和 CORS 策略

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
2. 撤消端点
3. UserInfo 端点

托管用户界面和授权服务器 Cookie

Amazon Cognito 用户池终端节点在用户的浏览器中设置 Cookie。这些 Cookie 符合某些浏览器的要求，即网站不设置第三方 Cookie。它们的作用域仅限于您的用户池端点，包括以下内容：

- 每个请求都有 XSRF-TOKEN cookie。
- 用于在用户被重定向时保持会话一致性的 csrf-state Cookie。
- 一种cognito会话 Cookie，可将成功的登录尝试保存一小时。

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
- 将您自己的域用于托管 UI

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。

> **Note**
>
> 为了增强您的 Amazon Cognito 应用程序的安全性，用户群体端点的父域将在公共后缀列表 (PSL) 中注册。PSL 可帮助用户的网络浏览器对您的用户群体端点及其设置的 Cookie 建立一致的理解。

用户群体端点父域采用以下格式。

```
auth.Region.amazoncognito.com
auth-fips.Region.amazoncognito.com
```

要使用添加应用程序客户端和 Amazon Cognito 托管域 AWS Management Console，请参阅。创建应用程序客户端

主题

- 先决条件
- 步骤 1：配置托管用户池域
- 步骤 2：验证登录页面

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。

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 或 AWS CLI 或 API 来配置用户池域。

Amazon Cognito console

配置域

1. 导航到用户池的应用程序集成选项卡。
2. 选择域旁边的操作，然后选择创建自定义域或创建 Amazon Cognito 域。如果您已配置用户群体域，请先选择删除 Amazon Cognito 域或删除自定义域，然后再创建新的自定义域。
3. 输入可用的域前缀，以与 Amazon Cognito 域结合使用。有关设置自定义域的信息，请参阅将您自己的域用于托管 UI
4. 选择创建。

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

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 端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
- 更改自定义域的 SSL 证书

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。
- 您拥有的 Web 域。其父域必须具有有效的 DNS A 记录。您可以为该记录分配任何值。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**
>
> 在申请或导入证书之前，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）。

- 一种允许您的用户池授权服务器向用户会话添加 Cookie 的应用程序。Amazon Cognito 为托管用户界面设置了几个必需的 Cookie。这包括 `cognito`、`cognito-fl` 和 `XSRF-TOKEN`。尽管每个 Cookie 都符合浏览器大小限制，但更改用户池配置可能会导致托管用户界面 Cookie 的大小变大。像自定义域前面的应用程序负载均衡器 (ALB) 这样的中间服务可能会强制规定最大标题大小或总数 Cookie 大小。如果您的应用程序还设置了自己的 Cookie，则用户的会话可能会超过这些限制。为避免大小限制冲突，我们建议您的应用程序不要在托管 UI 子域上设置 Cookie。
- 允许更新 Amazon CloudFront 分配。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": [
                "*"
            ]
        }
    ]
}
```

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

Amazon Cognito 最初使用您的 IAM 权限来配置 CloudFront分配，但分配由管理。 AWS您无法更改 Amazon Cognito 与您的用户池关联的 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. 选择用户池。
3. 选择要向其添加域的用户池。
4. 选择应用程序集成选项卡。
5. 选择在域旁边的操作，然后选择创建自定义域。

> **Note**
>
> 如果您已配置用户池域，请先选择删除 Cognito 域或删除自定义域来删除现有域，然后再创建新的自定义域。

6. 对于自定义域，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 证书，请选择要用于您的域的 SSL 证书。无论您的用户池如何，只有美国东部（弗吉尼亚北部）的 ACM 证书才有资格用于 Amazon Cognito 自定义域。 AWS 区域

   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. 选择创建。
9. Amazon Cognito 将您返回到应用程序集成选项卡。此时将显示标题为在域名的 DNS 中创建别名记录的消息。记下控制台中显示的域和别名目标。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，请创建一个以您的自定义域为父域的根域的托管区域。有关更多信息，请参阅

   a. 选择创建托管区域。

   b. 从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. 输入托管区域的描述。

   d. 选择公有托管区域的托管区域类型以允许公共客户端解析您的自定义域。不支持选择私有托管区域。

   e. 根据需要应用标签。

   f. 选择创建托管区域。

   > **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。否则，请创建 A 记录。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. 在托管区域页面上，选择您的托管区域的名称。
4. 如果您还没有 DNS 记录，请为自定义域名的父域名添加 DNS 记录。为父域添加 DNS A 记录，然后选择创建记录。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。就父域验证而言，授权起始记录 (SOA) 记录不足以构成足以验证父域名的 DNS 记录。

5. 为您的自定义域名添加 DNS 记录。例如，您的记录必须指向自定义域别名目标`123example.cloudfront.net`。再次选择创建记录。
6. 输入与您的自定义域相匹配的记录名称（例如，*myapp*）以创建 *myapp.auth.example.com* 记录。
7. 启用别名选项。
8. 选择将流量路由至、别名到 CloudFront 分配。输入由 Amazon Cognito 在您创建自定义域时提供的别名目标。
9. 选择创建记录。

> **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 GetChangeAPI 方法来验证您的更改是否已传播。

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，网址为。https://console.aws.amazon.com/cognito/home
2. 选择用户池。
3. 选择要更新其证书的用户池。
4. 选择应用程序集成选项卡。
5. 选择 操作、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 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_callbac`
> 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

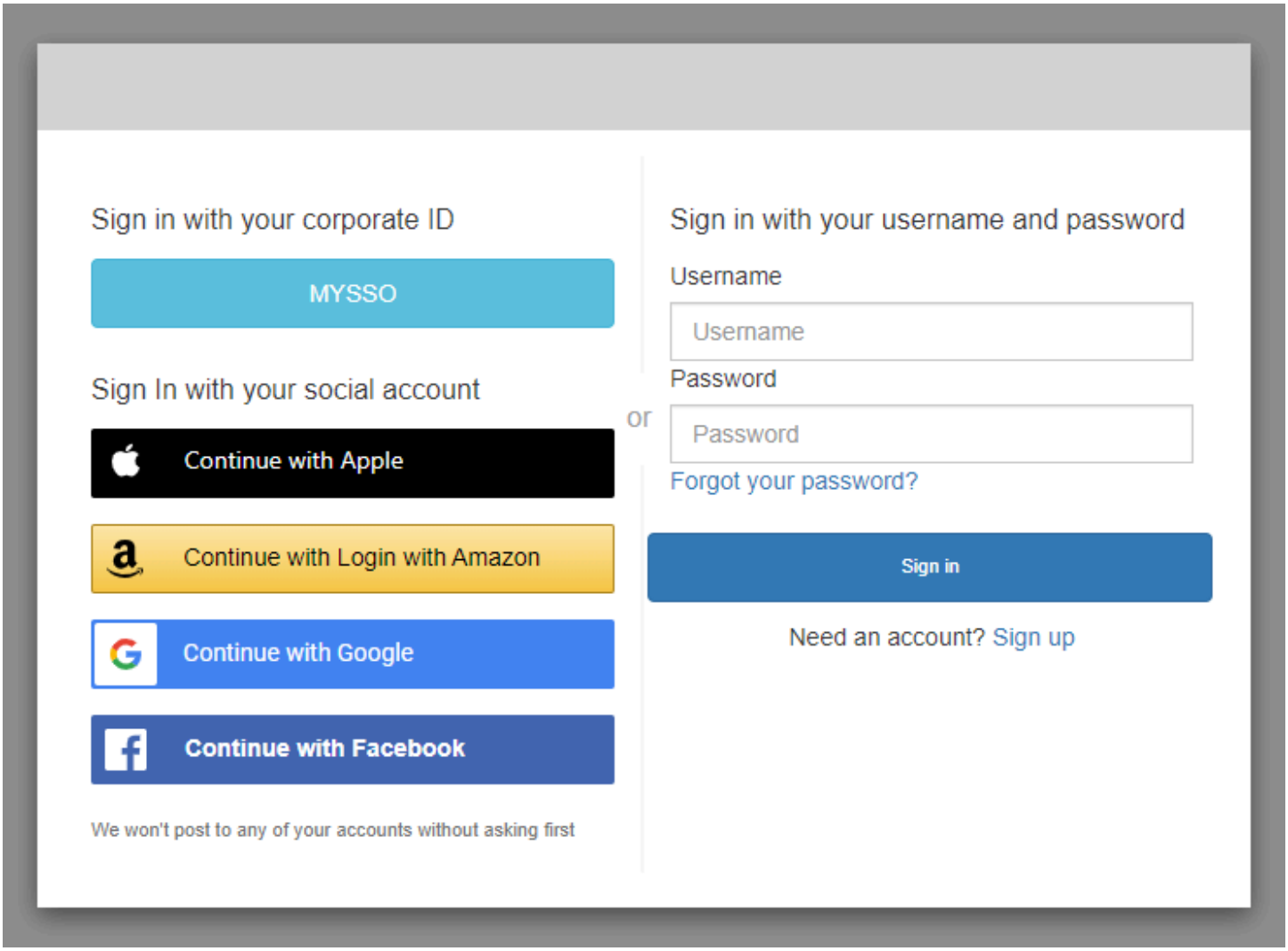
Amazon Cognito 托管 UI 从 登录端点开始。登录页面的 URL 是您为用户群体选择的域与反映您希望发出的 OAuth 2.0 授予、您的应用程序客户端、您的应用程序的路径和您希望请求的 OpenID Connect (OIDC) 范围的参数的组合。

```
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
- 如何使用 Amazon Cognito 托管 UI 登录
- 如何使用 Amazon Cognito 托管 UI 重置密码

如何在 Amazon Cognito 托管 UI 中注册新账户

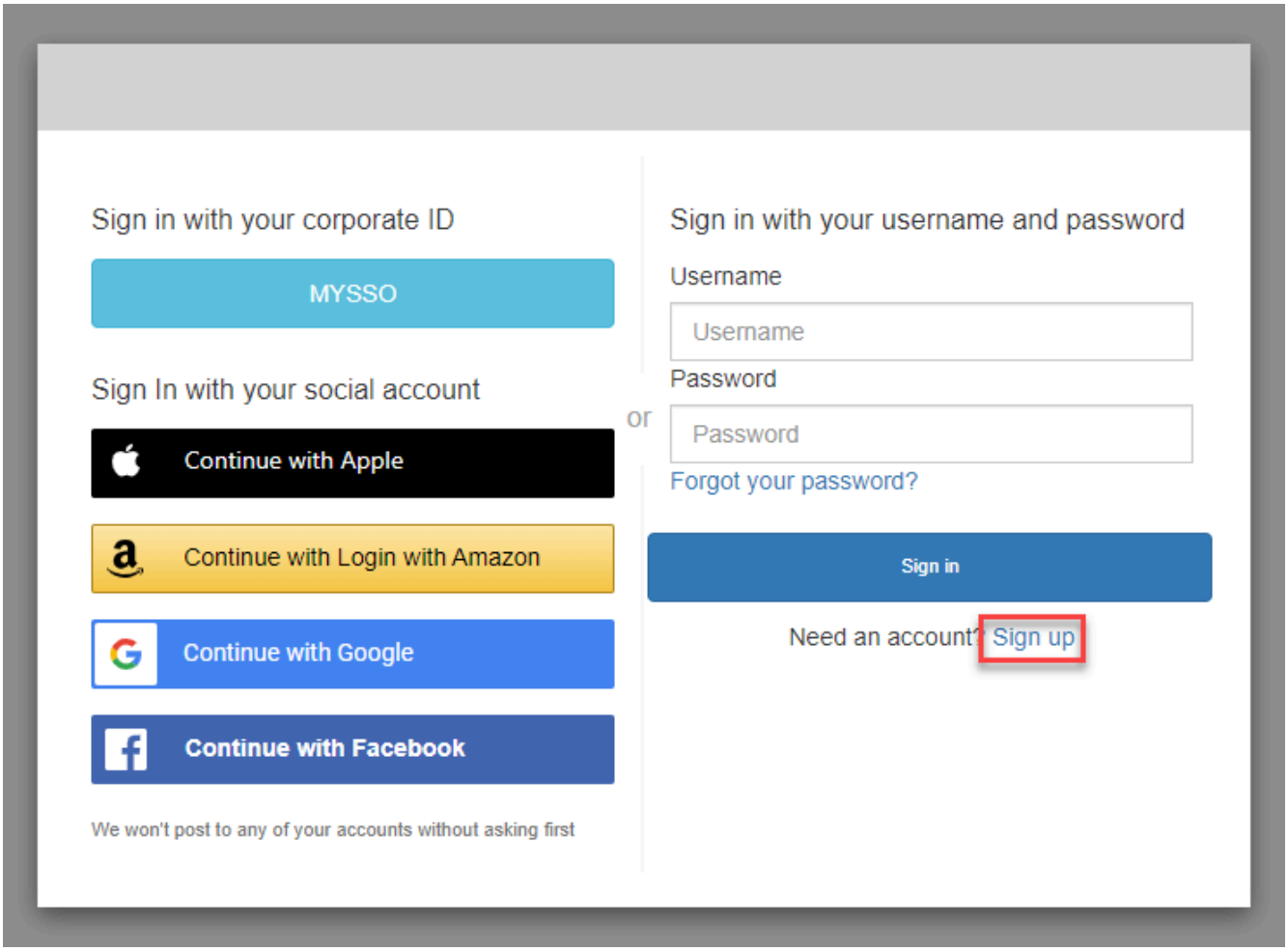
本指南向您介绍如何在使用 Amazon Cognito 的应用程序中注册用户账户。

> **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。

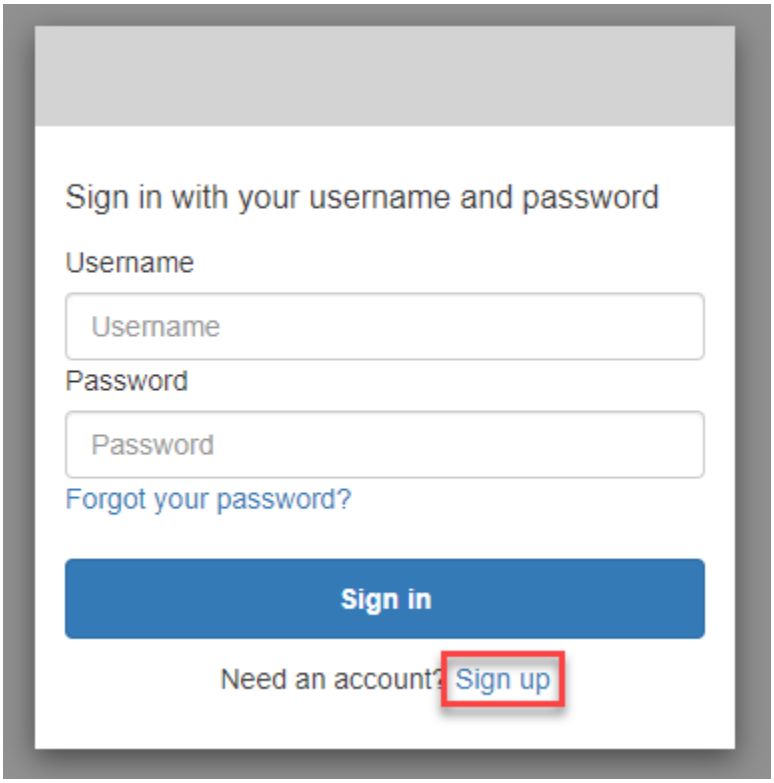
1. 如果您打算使用用户名和密码通过 Amazon Cognito 登录，而不是通过应用程序所有者列出的第三方登录提供商之一登录，请从登录页面中选择 Sign up（注册）。

   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。

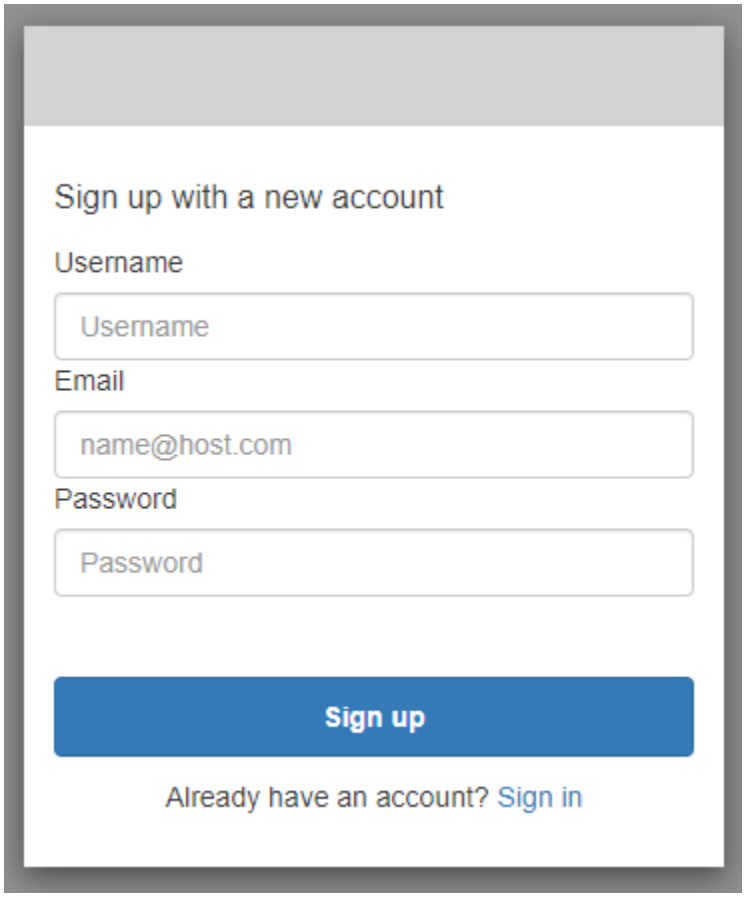
   With multiple sign-in providers



With only Amazon Cognito as a sign-in provider

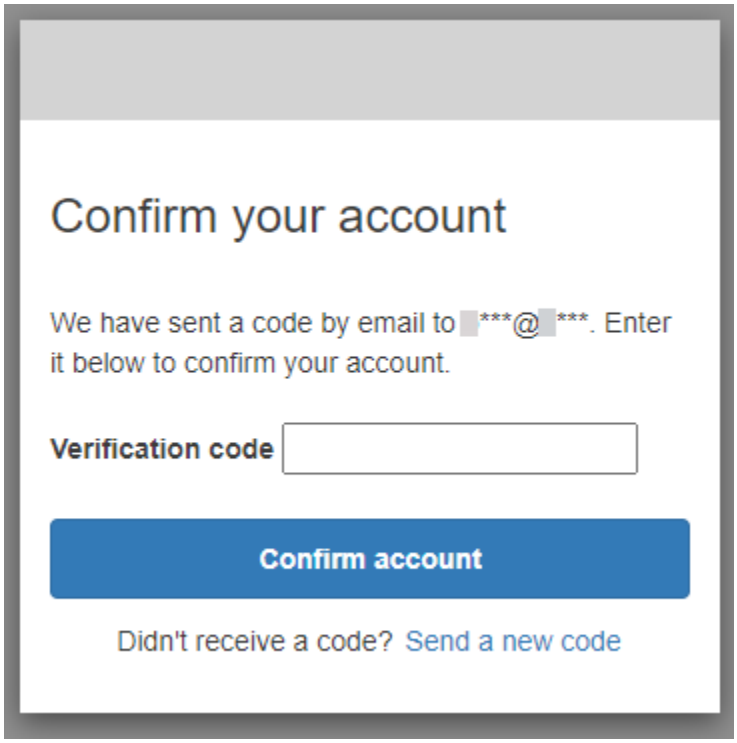


2. 在 Sign up with a new account（使用新账户注册）页面上，应用程序所有者要求提供注册所需的信息。他们可能会要求提供用户名、电子邮件地址或电话号码。输入所需信息，然后选择密码。



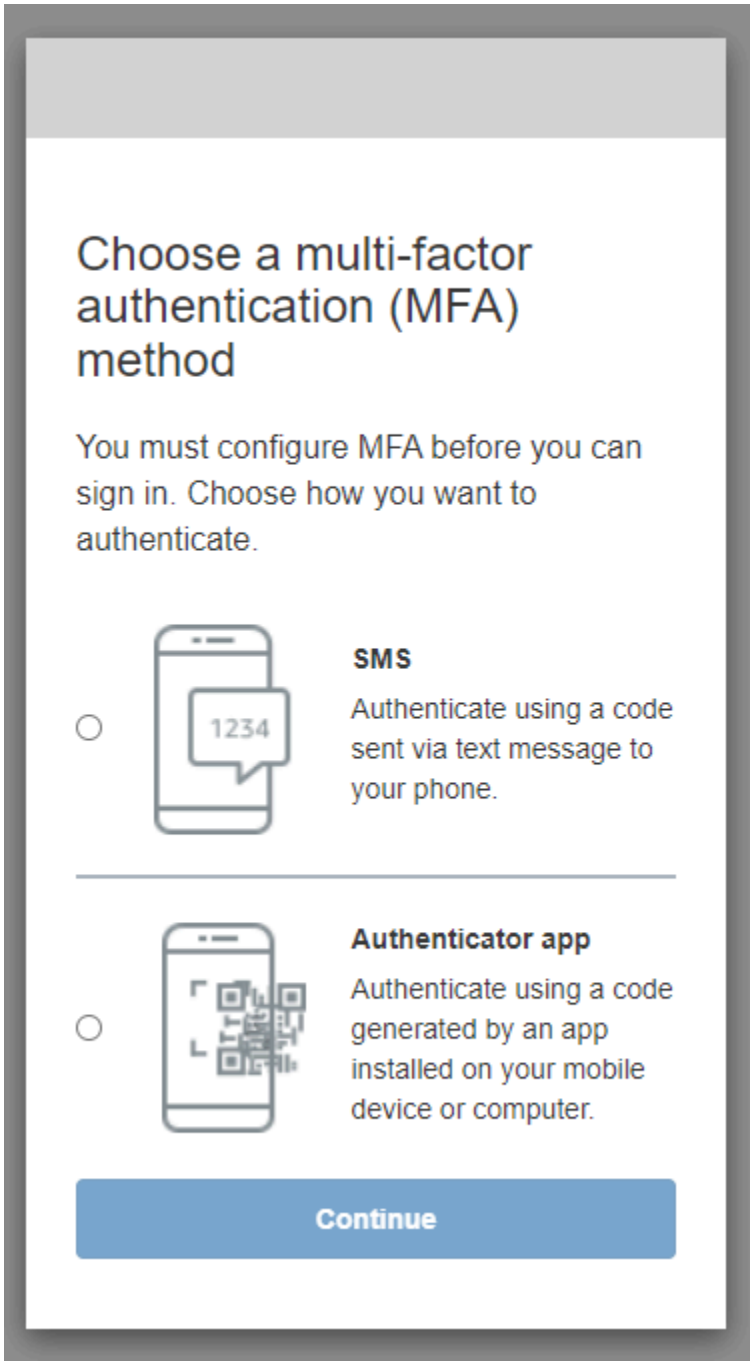
3. 在 Confirm your account（确认您的账户）页面上，应用程序所有者可能会要求您确认账户，以验证您是否可以通过您提供的电子邮件地址或电话号码接收消息。

   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。



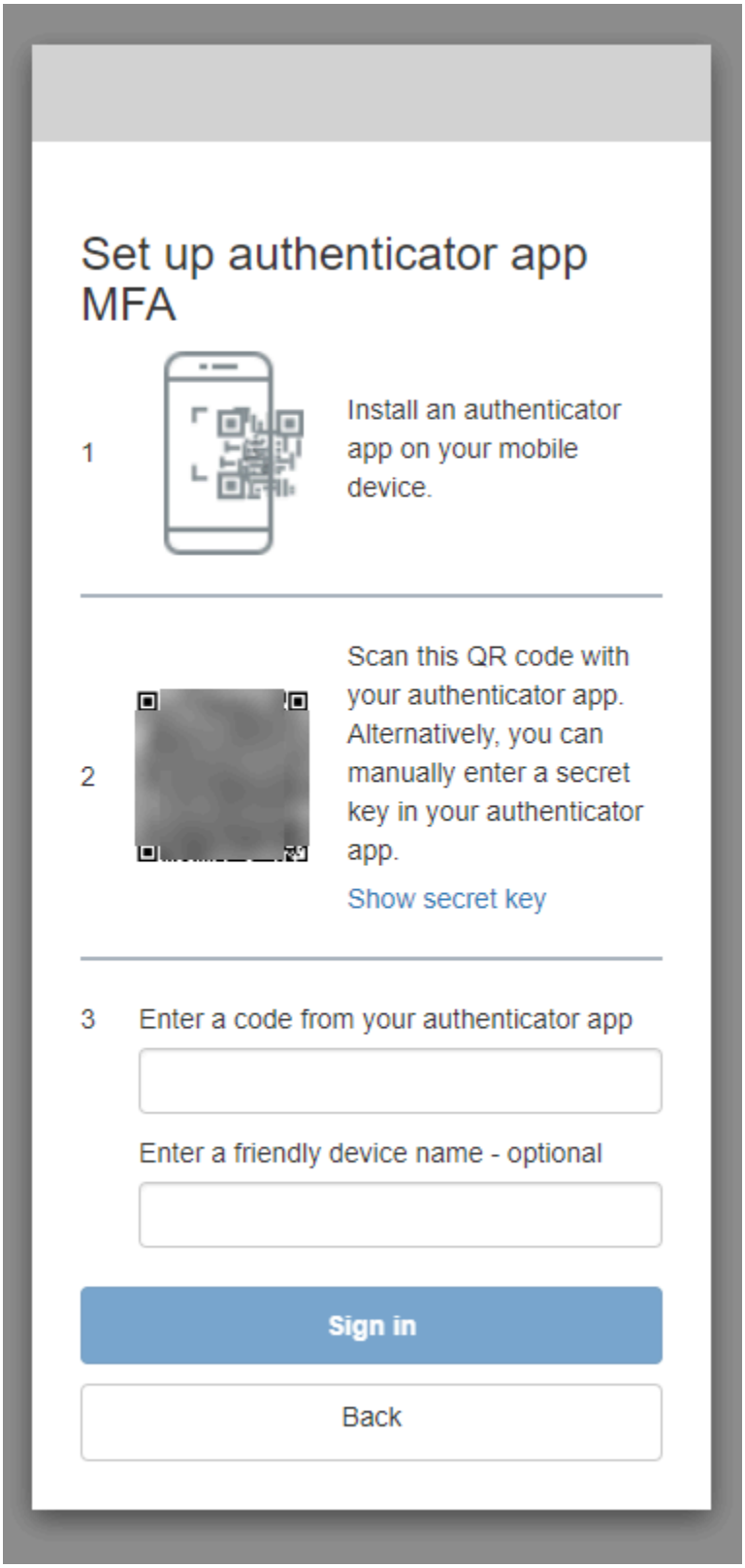
5. Amazon Cognito 要求您提供来自身份验证器应用程序或 SMS 文本消息的代码。在 3 分钟内输入您收到的代码。

   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。

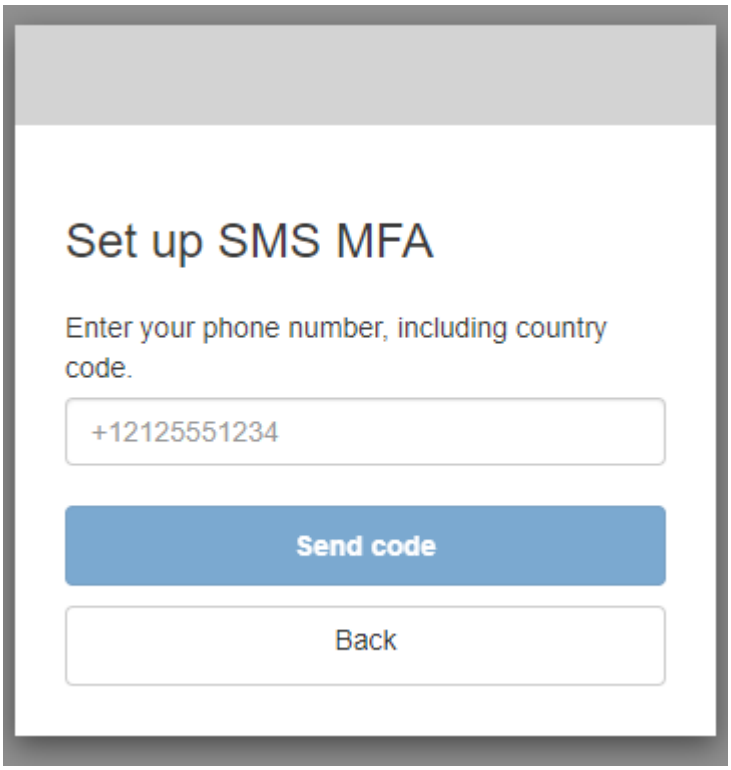
3. 您的身份验证器应用程序开始显示每隔几秒钟更改一次的代码。输入应用程序中的当前代码。
4. （可选）在 Set up authenticator app MFA（设置身份验证器应用程序 MFA）页面上，为您的设备选择名称。当您登录时，Amazon Cognito 会要求您提供具有您在此处提供的名称的设备中的代码。


Set up authenticator app MFA
1
Install an authenticator app on your mobile device.
2
Scan this QR code with your authenticator app. Alternatively, you can manually enter a secret key in your authenticator app.
Show secret key
3
Enter a code from your authenticator app
Enter a friendly device name - optional
Sign in
Back

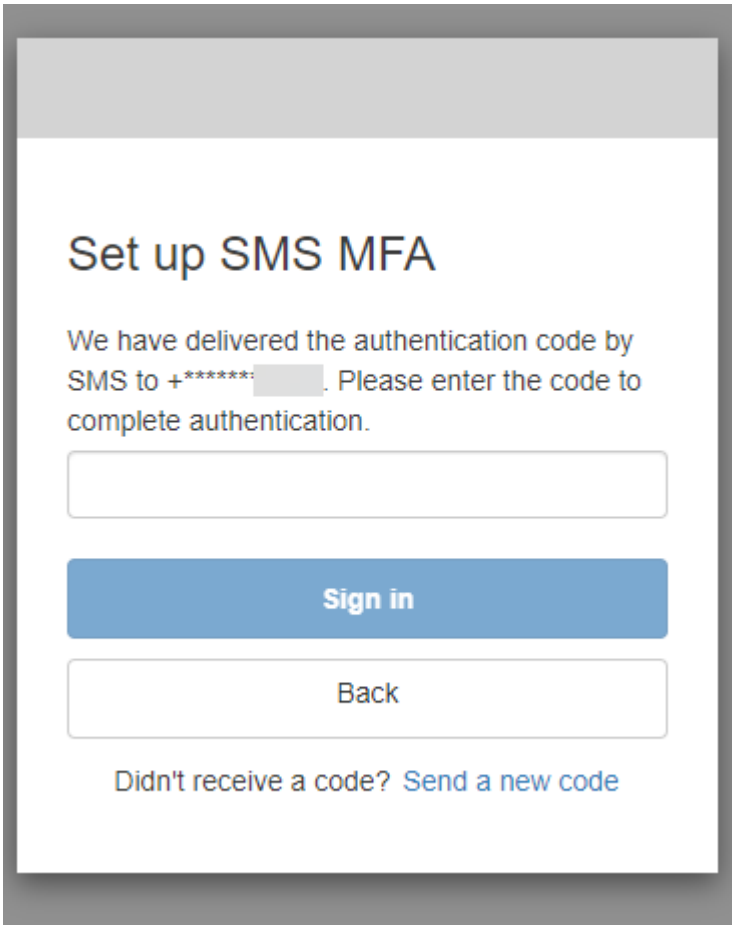
SMS text message

1. 如果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

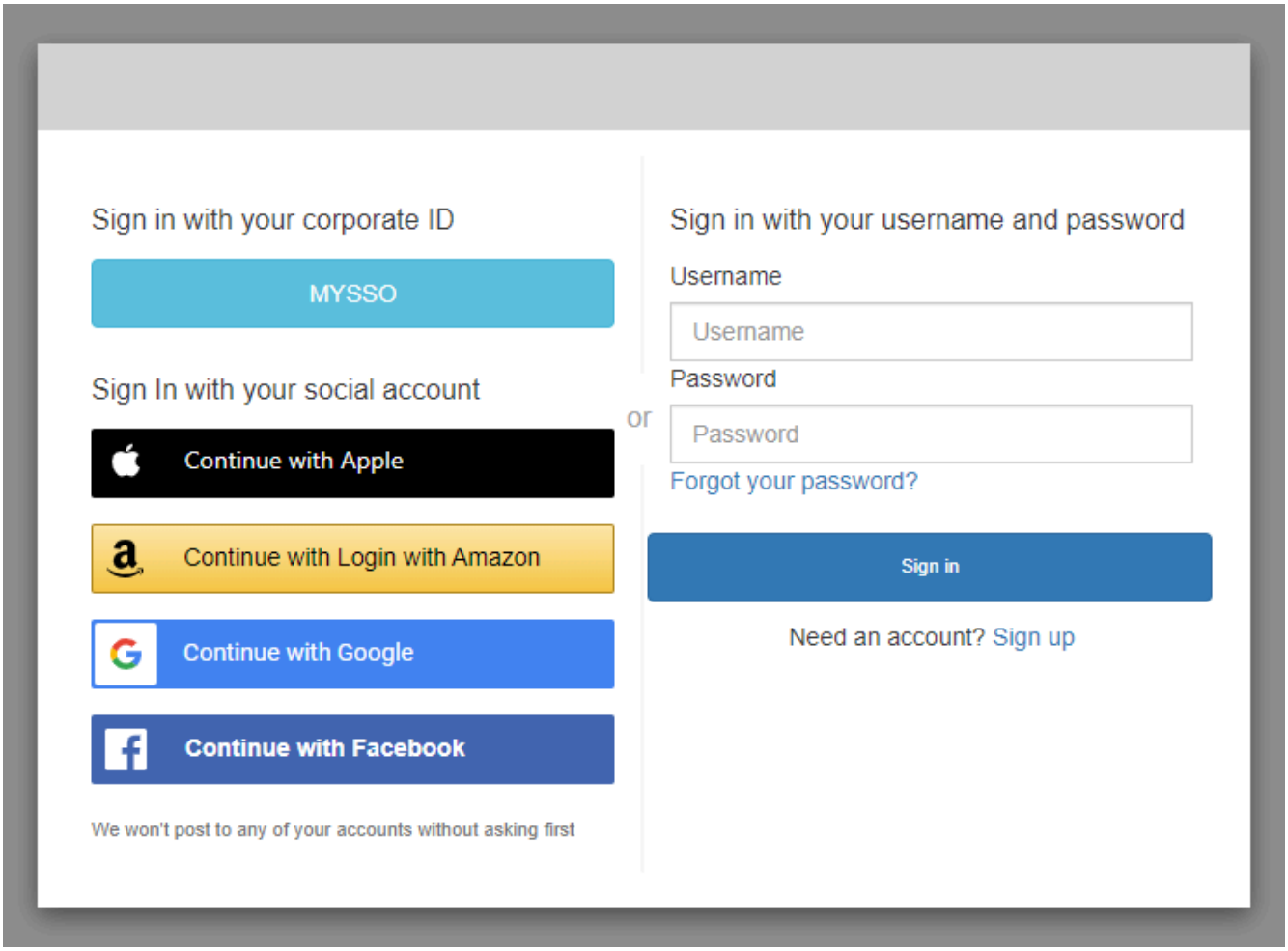
本指南向您介绍如何登录到使用 Amazon Cognito 的应用程序。

> **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。

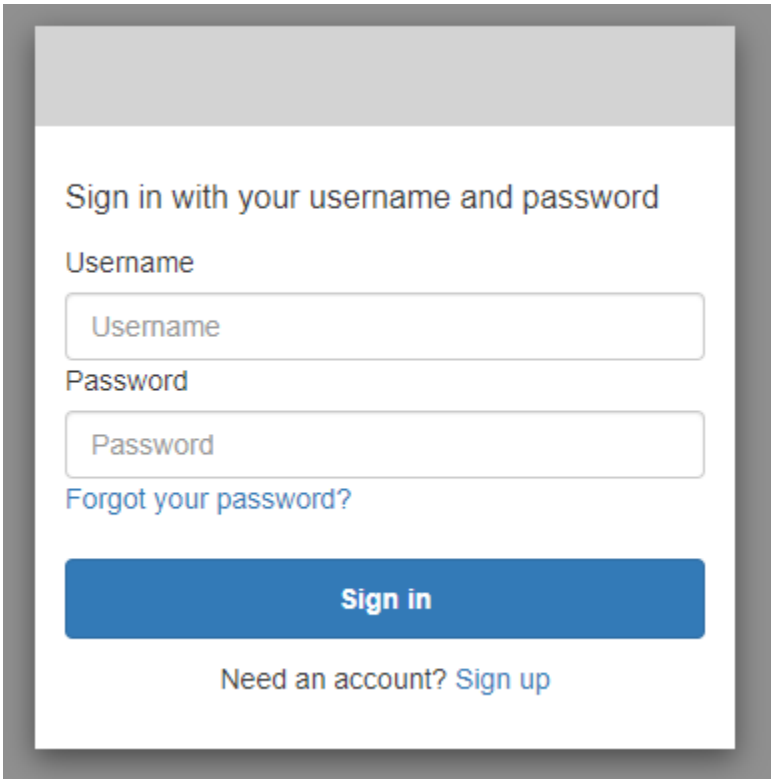
1. 根据应用程序所有者选择的选项，您可能可以选择用于登录的提供商，也可能只看到输入用户名和密码的提示。当您从此页面使用用户名和密码登录时，Amazon Cognito 是您的登录提供商。否则，您的登录提供商将由您选择的按钮表示。

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

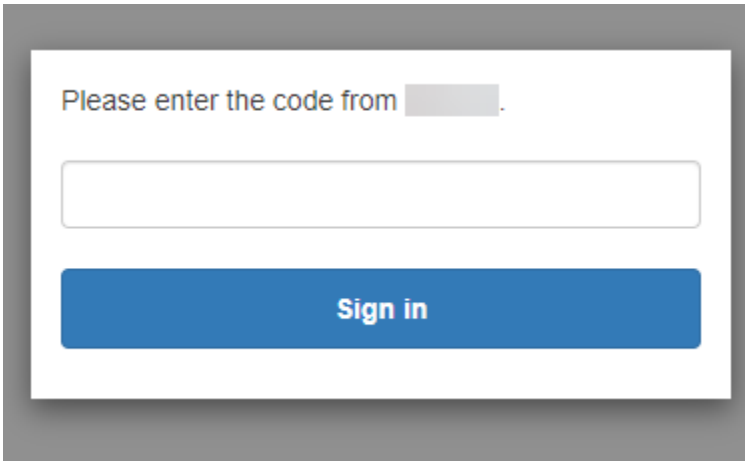


With only Amazon Cognito as a sign-in provider

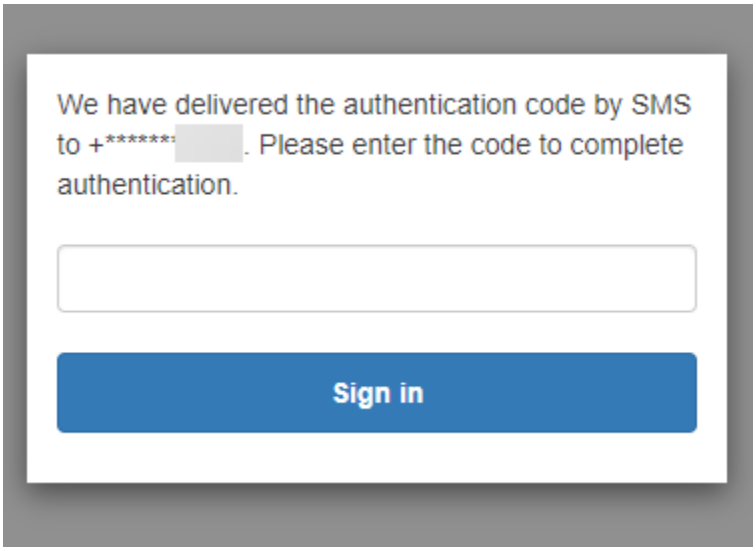


2. 您可能在应用程序中注册时已设置了 MFA。输入您在 SMS 消息中收到的或显示在身份验证器应用程序中的 MFA 代码。您必须在 3 分钟内输入此代码。

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

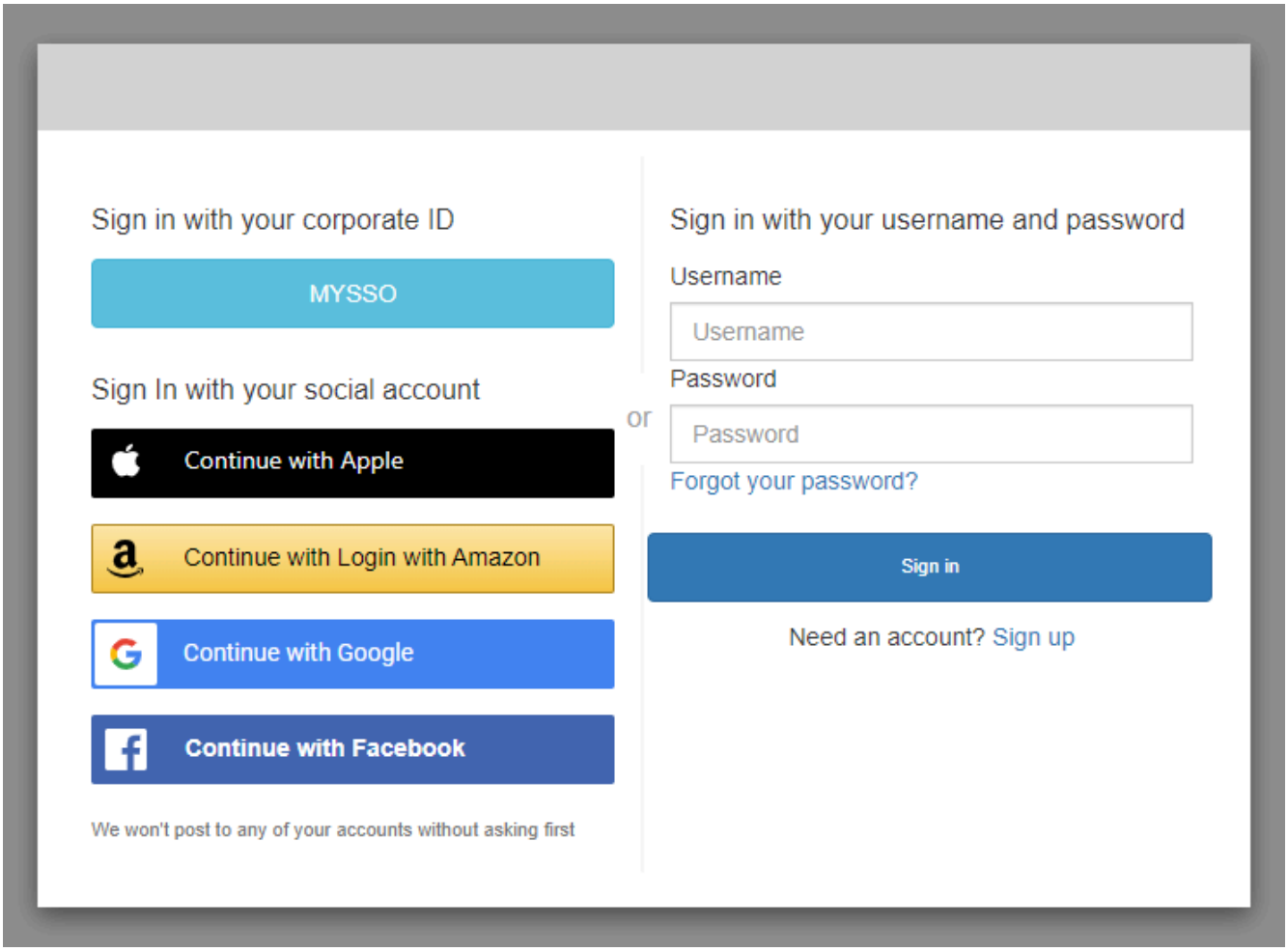
本指南向您介绍如何在使用 Amazon Cognito 的应用程序中重置密码。

> **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。

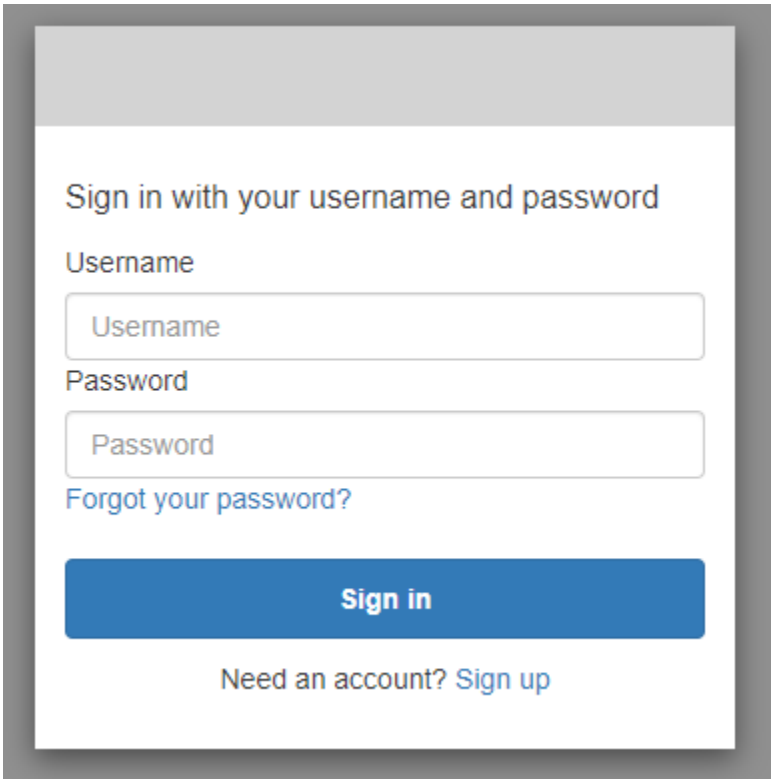
1. 根据应用程序所有者选择的选项，您可能可以选择用于登录的提供商，也可能只看到输入用户名和密码的提示。当您从此页面使用用户名和密码登录时，Amazon Cognito 是您的登录提供商。否则，您的登录提供商将由您选择的按钮表示。

   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？)

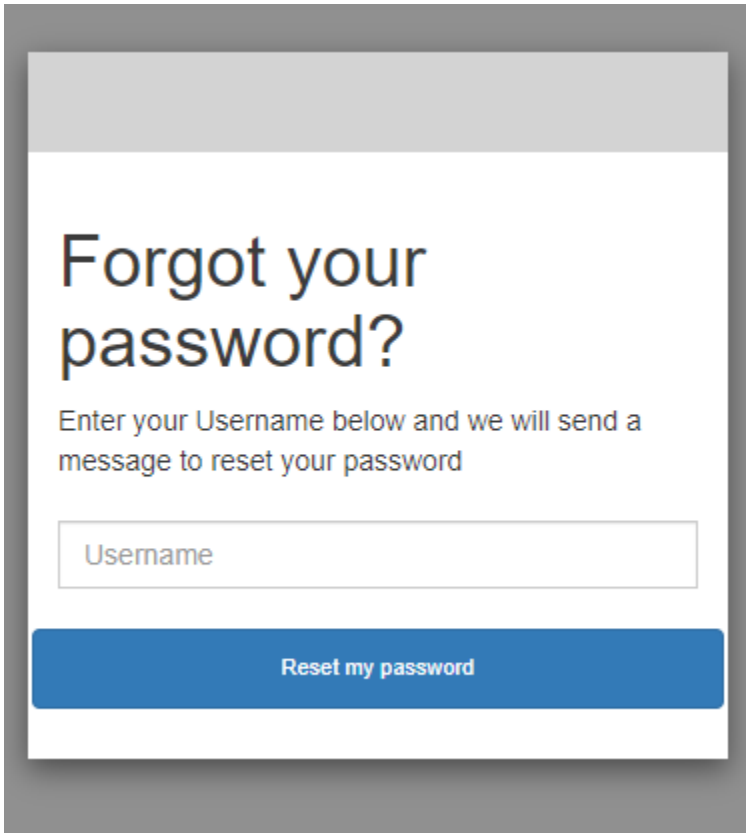
With multiple sign-in providers



With only Amazon Cognito as a sign-in provider

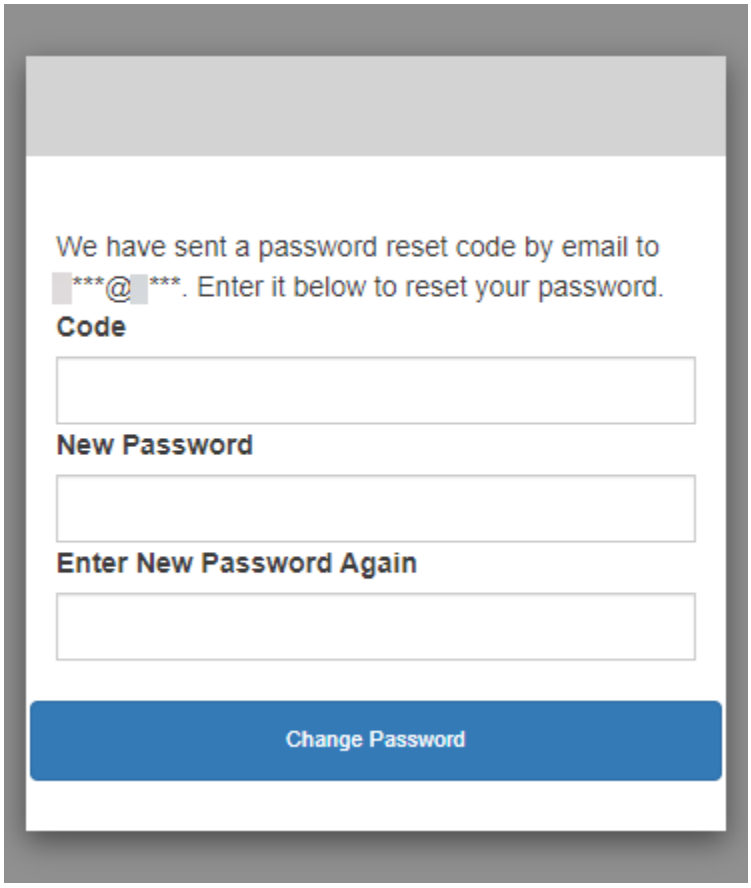


2. 在 Forgot your password? (忘记密码?) 页面上，Amazon Cognito 会提示您输入用于登录的信息。这可能是您的用户名、电子邮件地址或电话号码。



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。

使用资源服务器进行范围、M2M 和 API 授权

在为用户群体配置域后，Amazon Cognito 会自动预置一个 OAuth 2.0 授权服务器和托管 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 添加带有托管 UI 的应用程序客户端。您可以选择希望授权服务器添加到访问令牌中的范围。范围授权访问资源服务器和用户数据。

资源服务器 是 OAuth 2.0 API 服务器。为了保护受访问权限保护的资源，它会验证用户群体中的访问令牌所包含的范围是否授权所请求的方法和它所保护的 API 中的路径。它根据令牌签名验证发放者，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。用户池范围位于访问令牌`scope`声明中。有关 Amazon Cognito 访问令牌中的声明的更多信息，请参阅使用访问令牌。

借助 Amazon Cognito，访问令牌中的范围可以授权访问外部 API 或用户属性。您可以向本地用户、联合用户或计算机身份颁发访问令牌。

M achine-to-machine (M2M) 授权

Amazon Cognito 支持使用机器身份访问 API 数据的应用程序。用户池中的计算机身份是在应用程序服务器上运行并连接到远程 API 的机密客户端。它们的操作无需用户交互：计划任务、数据流或资产更新。当这些客户端使用访问令牌授权其请求时，它们会执行机器对机器或 M2M 授权。在 M2M 授权中，共享密钥取代访问控制中的用户凭证。

通过 M2M 授权访问 API 的应用程序必须具有客户端 ID 和客户端密钥。在您的用户池中，您必须构建支持客户端凭据授予的应用程序客户端。要支持客户端凭据，您的应用程序客户端必须具有客户端密钥，并且必须具有用户池域。在此流程中，您的计算机身份直接从中请求访问令牌令牌端点。您只能在访问令牌中授权来自资源服务器的自定义范围，以获得客户端凭据授权。有关设置应用程序客户端的更多信息，请参阅用户池应用程序客户端。

来自客户端凭证授予的访问令牌是您希望允许计算机身份从 API 请求的操作的可验证声明。要详细了解访问令牌如何授权 API 请求，请继续阅读。有关示例应用程序，请参阅使用 AWS CDK 的基于 Amazon Cognito 和 API Gateway 的机器对机器授权。

M2M 授权的计费模式不同于每月活跃用户 (MAU) 的计费方式。用户身份验证会产生每位活跃用户的费用，而 M2M 计费则反映了活跃的客户凭证、应用程序客户端和令牌请求总量。有关更多信息，请参阅 Amazon Cognito 定价。要控制 M2M 授权的成本，请优化访问令牌的持续时间和应用程序发出的令牌请求数量。缓存令牌有关使用 API Gateway 缓存来减少 M2M 授权中对新令牌的请求的方法，请参阅。

有关优化会增加 AWS 账单成本的 Amazon Cognito 操作的信息，请参阅。管理 成本

关于范围

范围是应用程序可请求的对资源的访问权限的级别。在 Amazon Cognito 访问令牌中，范围由您与用户群体建立的信任提供支持：一个具有已知数字签名的可信访问令牌发放者。用户群体可以生成访问令牌，其范围可以证明您的客户可以管理自己的部分或全部用户个人资料，或者可以从后端 API 检索数据。Amazon Cognito 用户群体使用用户群体预留 API 范围、自定义范围和标准范围来发放访问令牌。

用户群体预留 API 范围

`aws.cognito.signin.user.admin` 范围将授权 Amazon Cognito 用户群体 API。它授权访问令牌持有者通过和 UpdateUserAttributesAPI 操作查询和更新有关用户池用户的所有信息。GetUser当您使用 Amazon Cognito 用户群体 API 对用户进行身份验证时，这是您在访问令牌中收到的唯一范围。这也是您读写已授权应用程序客户端读写的用户属性所需的唯一范围。您也可以在发往 对端点授权 的请求中请求此范围。仅此范围不足以向 UserInfo 端点 请求用户属性。对于同时授权用户群体 API 和

用户 userInfo 请求的访问令牌，您必须在一个 /oauth2/authorize 请求中同时请求 openid 和 aws.cognito.signin.user.admin 这两个范围。

自定义范围

自定义范围授权对资源服务器所保护的外部 API 的请求。您可以使用其他类型的范围请求自定义范围。您可以在此页面中找到有关自定义范围的更多信息。

标准范围

使用用户群体 OAuth 2.0 授权服务器（包括托管 UI）验证用户时，必须请求范围。您可以在 Amazon Cognito 授权服务器中对用户群体本地用户和第三方联合用户进行身份验证。标准 OAuth 2.0 范围授权您的应用从您的用户群体的UserInfo 端点中读取用户信息。在 OAuth 模型中，您可以从 userInfo 端点查询用户属性，从而优化您的应用程序，使其能够处理大量的用户属性请求。userInfo 端点返回权限级别的属性，该级别由访问令牌中的范围决定。您可以授权您的应用程序客户端颁发具有以下标准 OAuth 2.0 范围的访问令牌。

openid

OpenID Connect (OIDC) 查询的最小范围。授权 ID 令牌、唯一标识符声明 sub 以及请求其他范围的能力。

> **Note**
>
> 当您请求 openid 范围而不请求其他范围时，您的用户群体 ID 令牌和 userInfo 响应将包括您的应用程序客户端可以读取的所有用户属性的声明。当您同时请求 openid 和其他标准范围（例如 profile、email 和 phone）时，ID 令牌和 userInfo 响应的内容将受到其他范围的限制。
> 例如，如果发送到 对端点授权 的请求带有参数 scope=openid+email，则将返回带有 sub、email 和 email_verified 的 ID 令牌。来自此请求的访问令牌也将从 UserInfo 端点 返回这些属性。带有参数 scope=openid 的请求将在 ID 令牌中返回所有客户端可以读取的属性，userInfo 响应也是如此。

配置文件

授权应用程序客户端可以读取的所有用户属性。

email

授权用户属性 email 和 email_verified。如果有已明确设置的值，Amazon Cognito 将返回 email_verified。

phone

授权用户属性 phone_number 和 phone_number_verified。

关于资源服务器

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。资源服务器将检查访问令牌以确定是否应授予访问权限。

Amazon Cognito 将来可能会更新用户群体访问令牌的架构。如果您的应用程序在将访问令牌传递给 API 之前分析其内容，则您必须对代码进行设计以接受架构的更新。

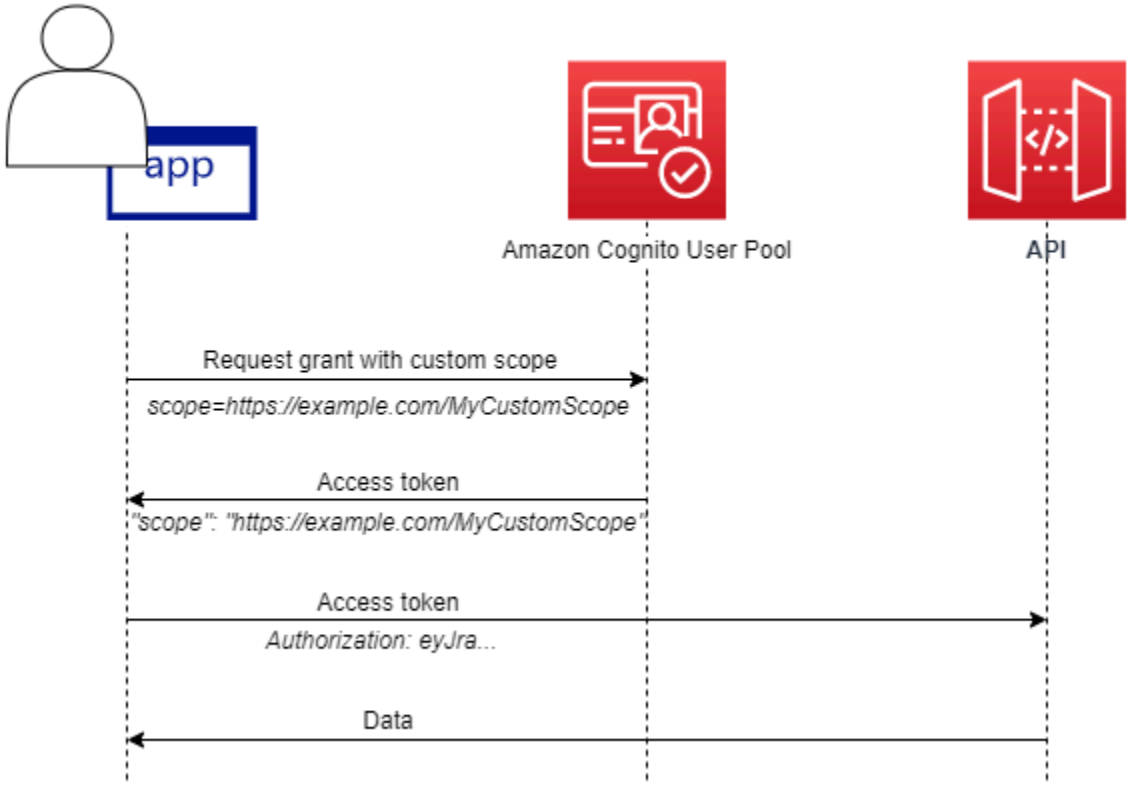
自定义范围由您定义，它会扩展用户群体的授权功能，以包括与查询和修改用户及其属性无关的目的。例如，如果您有一个照片资源服务器，它可能会定义两个范围：photos.read 用于对照片的读取访问，photos.write 用于写入/删除访问。您可以配置 API 以接受用于授权的访问令牌，并授予 HTTP GET 请求使用 scope 声明中的 photos.read 访问令牌，以及授予 HTTP POST 请求使用 photos.write 访问令牌。这些是自定义范围。

> **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。有关验证令牌的更多信息，请参阅验证 JSON Web 令牌。有关在 Amazon API Gateway 中验证和使用用户群体令牌的更多信息，请参阅博客将 Amazon Cognito 用户群体与 API Gateway 集成。API Gateway 是用于检查访问令牌和保护您的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。

概述

利用 Amazon Cognito，您可以创建 OAuth2.0 资源服务器并将自定义范围与它们相关联。访问令牌中的自定义范围可向 API 中的特定操作授权。您可以授权用户群体中的任何应用程序客户端从您的任何资源服务器发布自定义范围。将自定义范围与应用程序客户端相关联，并在来自令牌端点的 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 在访问令牌中将自定义

范围添加到 scope 声明中。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。



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 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**
>
> 因为它们是为以用户池作为 IdP 的人机交互身份验证而设计的 InitiateAuth，AdminInitiateAuth并且请求仅在访问令牌中生成具有单scope一值的声明。aws.cognito.signin.user.admin

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。

- 资源服务器名称：资源服务器的易记名称，如 Solar system object tracker 或 Photo API。

- 资源服务器标识符：资源服务器的唯一标识符。标识符是您希望与 API 关联的任何名称，例如 `solar-system-data`。您可以配置更长的标识符，例如 `https://solar-system-data-api.example.com`，作为对 API URI 路径的更直接引用，但较长的字符串会增加访问令牌的大小。
- 范围名称：`scope` 声明中需要的值。例如，`sunproximity.read`。
- 描述：范围的友好描述。例如，`Check current proximity to sun`。

Amazon Cognito 可以在任何用户的访问令牌中包含自定义范围，无论这些用户是用户群体的本地用户还是与第三方身份提供者的联合身份验证用户。在使用包含托管 UI 的 OAuth 2.0 授权服务器进行身份验证时，您可以为用户的访问令牌选择范围。您的用户的身份验证必须从以 `scope` 作为请求参数之一的对端点授权开始。以下是推荐的资源服务器格式。对于标识符，请使用 API 友好名称。对于自定义范围，请使用它们授权的操作。

```
resourceServerIdentifier/scopeName
```

例如，您在柯伊伯带发现了一颗新的小行星，您想通过 `solar-system-data` API 对其进行注册。授权对小行星数据库进行写操作的范围是 `asteroids.add`。当您请求授权您注册发现的小行星的访问令牌时，请将 `scope` HTTPS 请求参数格式设置为 `scope=solar-system-data/asteroids.add`。

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。而是范围标记为非活动。Amazon Cognito 不会为访问令牌添加非活动的范围，但如果您的应用程序请求访问令牌，则会正常进行。如果您稍后再次将范围添加到资源服务器，则 Amazon Cognito 会再次将其写入访问令牌。如果您请求的范围尚未与应用程序客户端关联，则无论您是否将其从用户群体资源服务器中删除，身份验证都会失败。

您可以使用 AWS Management Console、API 或 CLI 为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）

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

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（AWS CLI 和 AWS API）定义资源服务器

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

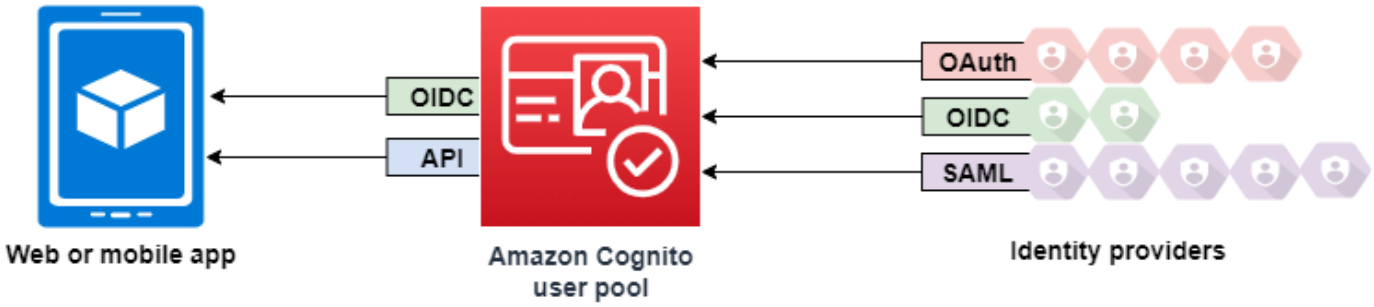
- AWS CLI: `aws cognito-idp update-resource-server`
- AWS API: UpdateResourceServer

通过第三方添加用户池登录

您的应用程序用户可以直接通过用户池登录，也可以通过第三方身份提供商 (IdP) 进行联合。用户池管理处理通过Facebook、谷歌、亚马逊和苹果进行社交登录以及从OpenID Connect（OIDC）和SAML返回的代币的开销。 IdPs借助内置的托管网络用户界面，Amazon Cognito 为所有经过身份验证的用户提供令牌处理和管理。 IdPs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。此特征不依赖于通过 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池目录中，无需通过外部 IdP 进行联合身份验证。

当你将 Amazon Cognito 连接到社交、SAML 或 OpenID Connect (OIDC IdPs) 时，你的用户池充当了多个服务提供商和你的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)。你将 OIDC ID 令牌或 SAML 声明 IdPs 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。

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。
- 您无法使用 InitiateAuth和之类的 API 操作登录联合用户AdminInitiateAuth。联合用户只能使用登录端点或对端点授权进行登录。
- 对端点授权 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。有关示例，请参阅示例场景：在企业控制面板中为 Amazon Cognito 应用程序添加书签。
- 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。

  例如：`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。
- 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。

  您可以将任何用户池分配 IdPs 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 登录。

主题

- 为用户池配置身份提供商
- 在用户池中使用社交身份提供商
- 在用户池中使用 SAML 身份提供商
- 将 OIDC 身份提供商与用户池配合使用
- 指定适用于用户池的身份提供商属性映射
- 将联合用户与现有用户配置文件关联

为用户池配置身份提供商

在联合身份提供商登录下的登录体验选项卡中，您可以将身份提供商 (IdPs) 添加到您的用户池中。有关更多信息，请参阅通过第三方添加用户池登录。

主题

- 使用社交 IdP 设置用户登录
- 使用 OIDC IdP 设置用户登录
- 使用 SAML IdP 设置用户登录

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，请参阅在用户池中使用社交身份提供商。
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）。

有关社交的更多信息 IdPs，请参阅在用户池中使用社交身份提供商。

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 的更多信息 IdPs，请参阅。将 OIDC 身份提供商与用户池配合使用

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 的元数据文档的信息 IdPs，请参阅配置您的第三方 SAML 身份提供商。

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://mydomain.us-east-1.amazoncognito.com/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

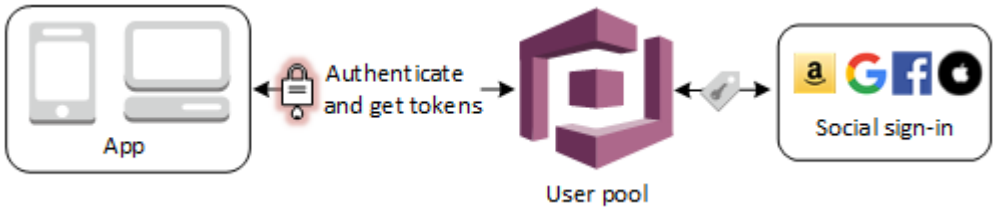
- 要获取包含 IdP 用于验证已签名注销请求的公有密钥的证书，请在联合身份验证控制台页面上的身份提供者下，在 SAML 对话框上的使用中的 SAML 提供者下选择显示签名证书对话框。

有关 SAML 的更多信息， IdPs 请参阅在用户池中使用 SAML 身份提供商。

在用户池中使用社交身份提供商

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 构建您的托管用户界面时，它会创建 OAuth 2.0 终端节点，Amazon Cognito 和您的 OIDC 以及社交活动使用这些终端节点来交换信息。 IdPs 有关更多信息，请参阅 Amazon Cognito 用户池 Auth API 参考。

你可以在中添加社交 IdP AWS Management Console，也可以使用 CLI 或 Amazon Cognito AP AWS I。



> **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。此特征不依赖于通过 Amazon Cognito 身份池（联合身份）实现的联合身份验证。

主题

- 先决条件
- 步骤 1：向社交 IdP 注册
- 步骤 2：将社交 IdP 添加到用户池
- 步骤 3：测试社交 IdP 配置

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

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://mydomain.us-east-1.amazoncognito.com/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://mydomain.us-east-1.amazoncognito.com
   ```
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://mydomain.us-east-1.amazoncognito.com/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默认项目。
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 “授权 JavaScript 来源” 下，选择 “添加 URI”。输入用户群体域。

    ```
    https://mydomain.us-east-1.amazoncognito.com
    ```
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://mydomain.us-east-1.amazoncognito.com/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 “使用 Apple 登录” 的 up-to-date 更多信息，请参阅 Apple 开发者文档中的配置环境以使用 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。

```
mydomain.us-east-1.amazoncognito.com
```

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://mydomain.us-east-1.amazoncognito.com/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

要配置用户池社交 IdP，请使用 AWS Management Console

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 AWS 凭据。
2. 选择用户池。
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://mydomain.us-east-1.amazoncognito.com/login?
response_type=code&client_id=1example23456789&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。

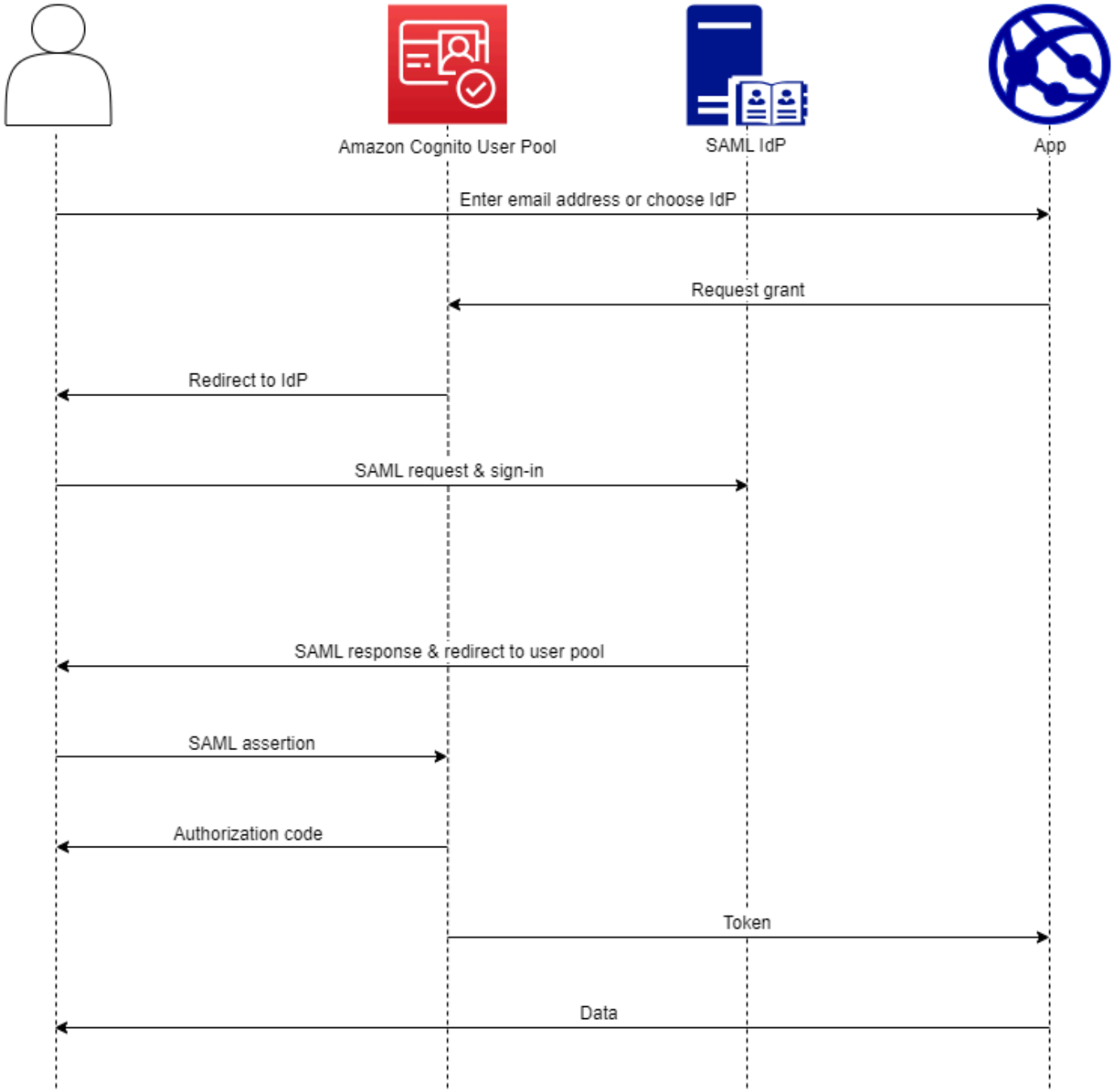
> **Note**
>
> Amazon Cognito 会取消未在 5 分钟内完成的身份验证请求，并将用户重定向到托管 UI。页面随即显示 `Something went wrong` 错误消息。

在用户池中使用 SAML 身份提供商

你可以选择让你的网络和移动应用程序用户通过 SAML 身份提供商 (IdP) 登录，例如微软 Active Directory 联合身份验证服务 (ADFS) 或 Shibboleth。您必须选择支持 SAML 2.0 标准的 SAML IdP。

借助托管的用户界面和联合终端节点，Amazon Cognito 可以对本地和第三方 IdP 用户进行身份验证，并发放 JSON 网络令牌 (JWT)。使用 Amazon Cognito 发行的令牌，您可以将多个身份源整合为适用于所有应用程序的通用 OpenID Connect (OIDC) 标准。Amazon Cognito 可以将您的第三方提供商的 SAML 声明处理为该 SSO 标准。您可以在、通过 AWS CLI或使用 Amazon Cognito 用户池 API 创建

和管理 SAML IdP。 AWS Management Console要在中创建您的第一个 SAML IdP AWS Management Console，请参阅。[在用户池中添加和管理 SAML 身份提供商]



> **Note**
>
> 通过第三方 IdP 登录进行联合是 Amazon Cognito 用户池的一项功能。Amazon Cognito 身份池（有时也称为 Amazon Cognito 联合身份）是一种联合身份的实现，您必须在每个身份池

中单独设置该联合身份。用户池可以是身份池的第三方 IdP。有关更多信息，请参阅 Amazon Cognito 身份池。

IdP 配置的快速参考

您必须将 SAML IdP 配置为接受请求并将响应发送到您的用户池。您的 SAML IdP 文档将包含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。以下文档提供了必须为 SP 实体 ID 和断言使用者服务 (ACS) URL 提供的值。

用户池 SAML 值快速参考

SP 实体 ID

```
urn:amazon:cognito:sp:us-east-1_EXAMPLE
```

ACS 网址

```
https://Your user pool domain/saml2/idpresponse
```

您必须配置您的用户池以支持您的身份提供商。添加外部 SAML IdP 的高级步骤如下。

1. 从您的 IdP 下载 SAML 元数据，或检索元数据终端节点的网址。请参阅 配置您的第三方 SAML 身份提供商。
2. 向您的用户池中添加新的 IdP。上传 SAML 元数据或提供元数据网址。请参阅 在用户池中添加和管理 SAML 身份提供商。
3. 将 IdP 分配给您的应用程序客户端。请参阅 用户池应用程序客户端。

主题

- 关于 Amazon Cognito 用户 IdPs 池中 SAML 的注意事项
- SAML 用户名区分大小写
- 在用户池中添加和管理 SAML 身份提供商
- SAML 会话在 Amazon Cognito 用户池中启动
- 使用 SP 启动的 SAML 登录
- 使用 IDP 发起的 SAML 登录

- SAML 注销流程
- SAML 签名和加密
- SAML 身份提供商名称和标识符
- 配置您的第三方 SAML 身份提供商

关于 Amazon Cognito 用户 IdPs 池中 SAML 的注意事项

Amazon Cognito 会为您处理 SAML 断言

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 SAM 启动和 IDP 启动的单点登录 (SSO)，如 SAML V2.0 技术概述第 5.1.2 和 5.1.4 节中所述。

提供有效的 IdP 签名证书

在用户池中配置 SAML IdP 时，SAML 提供商元数据中的签名证书不得过期。

用户池支持多个签名证书

如果在 SAML 元数据中，您的 SAML IdP 包含多个签名证书，则在登录时，只要与 SAML 元数据中的任何证书匹配，您的用户群体就会确定 SAML 断言有效。每个签名证书的长度不得超过 4,096 个字符。

维护继电器状态参数

Amazon Cognito 和您的 SAML IdP 使用 `relayState` 参数维护会话信息。

1. 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。
2. 该`relayState`令牌不透明地引用了由 Amazon Cognito 维护的状态信息。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。

识别 ACS 端点

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。

```
https://Your user pool domain/saml2/idpresponse
```

```
With an Amazon Cognito domain:
https://mydomain.us-east-1.amazoncognito.com/saml2/idpresponse
With a custom domain:
https://auth.example.com/saml2/idpresponse
```

有关用户群体域的更多信息，请参阅 配置用户池域。

没有重播的断言

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。

用户池 ID 是 SP 实体 ID

您必须向 IdP 提供您在服务提供商 (SP) 中的用户池 IDurn，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:us-east-1_EXAMPLE
```

您可以在 Amazon Cognito 控制台的用户池概述下找到您的用户池 ID。

映射所有必需的属性

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。

断言格式有特定的要求

您的 SAML IdP 必须在 SAML 断言中包含以下声明。

1. 一项NameID索赔。Amazon Cognito 通过将 SAML 断言与目标用户关联起来。NameID如果NameID发生变化，Amazon Cognito 会认为该声明是针对新用户的。您在 IdP 配置NameID中设置的属性必须具有永久值。

   ```
   <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:persistent">
     carlos
   </saml2:NameID>
   ```

2. 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:us-east-1_EXAMPLE
   ```

```
</saml:AudienceRestriction>
```

3. 对于 SP 发起的单点登录，其InResponseTo值为原始 SAML 请求 ID 的Response元素。

```
<saml2p:Response Destination="https://mydomain.us-east-1.amazoncognito.com/
saml2/idpresponse" ID="id123" InResponseTo="_dd0a3436-bc64-4679-
a0c2-cb4454f04184" IssueInstant="Date-time stamp" Version="2.0"
 xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" xmlns:xs="http://
www.w3.org/2001/XMLSchema">
```

> **Note**
>
> IDP 发起的 SAML 断言不得包含值。InResponseTo

4. 一个SubjectConfirmationData元素，其Recipient值为您的用户池saml2/idpresponse终端节点，对于由 SP 发起的 SAML，其InResponseTo值与原始 SAML 请求 ID 相匹配。

```
<saml2:SubjectConfirmationData InResponseTo="_dd0a3436-bc64-4679-a0c2-
cb4454f04184" NotOnOrAfter="Date-time stamp" Recipient="https://mydomain.us-
east-1.amazoncognito.com/saml2/idpresponse"/>
```

SP 发起的登录请求

当 对端点授权 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 请求包含有关您的用户池的信息，包括您的 ACS 终端节点。您可以选择对这些请求应用加密签名。

签署请求并加密响应

每个拥有 SAML 提供商的用户池都会生成一个非对称密钥对和签名证书，Amazon Cognito 将该数字签名分配给 SAML 请求。您配置为支持加密的 SAML 响应的每个外部 SAML IdP 都会导致 Amazon Cognito 为该提供商生成新的密钥对和加密证书。要查看和下载带有公钥的证书，请在 Amazon Cognito 控制台的登录体验选项卡中选择您的 IdP。

要与来自用户池的 SAML 请求建立信任，请向您的 IdP 提供您的用户池 SAML 2.0 签名证书的副本。如果您未将 IdP 配置为接受已签名的请求，则您的 IdP 可能会忽略您的用户池签署的 SAML 请求。

1. Amazon Cognito 将数字签名应用于您的用户传递给您的 IdP 的 SAML 请求。您的用户池签署所有单点注销 (SLO) 请求，您可以将用户池配置为签署任何 SAML 外部 IdP 的单点登录 (SSO) 请求。当您提供证书副本时，您的 IdP 可以验证用户的 SAML 请求的完整性。

2. 您的 SAML IdP 可以使用加密证书加密 SAML 响应。当您配置采用 SAML 加密的 IdP 时，您的 IdP 只能发送加密的响应。

对非字母数字字符进行编码

Amazon Cognito 不接受你的 IdP 作为属性值传递的 4 字节 UTF-8 字符，比如 # 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>
```

元数据端点必须具有有效的传输层安全性

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。有关验证证书的更多信息，请参阅什么是 SSL/TLS 证书？ 。

使用 IDP 启动的 SAML 的应用程序客户端只能使用 SAML 登录

当您激活对支持 IdP 启动登录应用程序客户端的 SAML 2.0 IdP 的支持时，您只能向该应用程序客户端添加其他 SAML IdPs 2.0。您无法将用户池中的用户目录以及所有非 SAML 外部身份提供商添加到以这种方式配置的应用程序客户端。

注销响应必须使用 POST 绑定

终`/saml2/logout`端节点`LogoutResponse`接受`HTTP POST`请求。用户池不接受带有`HTTP GET`绑定的注销响应。

SAML 用户名区分大小写

当联合用户尝试登录时，SAML 身份提供商 (IdP) 会在用户的 SAML 断言中向 Amazon Cognito 传递一个NameId唯一的。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 在通过唯一且区分大小写的声明时，Amazon Cognito 都会识别从 SAML IdP 返回的联合用户。NameId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。

主题

- 在用户池中添加和管理 SAML 身份提供商
- SAML 会话在 Amazon Cognito 用户池中启动
- 使用 SP 启动的 SAML 登录
- 使用 IDP 发起的 SAML 登录
- SAML 注销流程
- SAML 签名和加密
- SAML 身份提供商名称和标识符
- 配置您的第三方 SAML 身份提供商

在用户池中添加和管理 SAML 身份提供商

以下过程演示如何在 Amazon Cognito 用户池中创建、修改和删除 SAML 提供商。

AWS Management Console

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

在创建 SAML IdP 之前，您必须拥有从第三方 IdP 那里获得的 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，请参阅配置您的第三方 SAML 身份提供商。

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. 输入提供商名称。您可以在`identity_provider`请求参数中将此友好名称传递给对端点授权。
7. 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。
8. 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://mydomain.us-east-1.amazoncognito.com/saml2/logout` 终端节点发送注销响应。此 `saml2/logout` 端点使用 POST 绑定。

   > **Note**
   >
   > 如果选择了此选项，并且您的 SAML IdP 需要签名的注销请求，则您还必须向 SAML IdP 提供用户池中的签名证书。
   > 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。

9. 选择您的 IDP 启动的 SAML 登录配置。作为安全最佳实践，请选择 “仅接受 SP 发起的 SAML 断言”。如果您已准备好安全地接受未经请求的 SAML 登录会话，请选择接受 SP 发起和 IDP 发起的 SAML 断言。有关更多信息，请参阅 SAML 会话在 Amazon Cognito 用户池中启动。
10. 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**
>
> 如果您的提供商有公共终端节点，我们建议您输入元数据文档 URL，而不是上传文件。Amazon Cognito 会自动刷新元数据网址中的元数据。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）。

11. 在您的 SAML 提供商和用户池之间映射属性，将 SAML 提供商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。有些 IdPs 使用简单的名称，例如email，而另一些则使用如下所示的名称。

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

12. 选择创建。

API/CLI

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` *`us-east-1_EXAMPLE`* `--provider-name=`*`SAML_provider_1`* `--provider-type SAML --provider-details file:///details.json --attribute-mapping` *`email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`*

  其中 `details.json` 包含：

  ```
  "ProviderDetails": {
        "MetadataFile": "<SAML metadata XML>",
        "IDPSignout" : "true",
        "RequestSigningAlgorithm" : "rsa-sha256",
        "EncryptedResponses" : "true",
  ```

```
      "IDPInit" : "true"
}
```

> **Note**
>
> 如果<SAML metadata XML>包含该字符的任何实例"，则必须添加\为转义字符：\"。

带元数据 URL 的示例：`aws cognito-idp create-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-type SAML --provider-details MetadataURL=https://myidp.example.com/sso/saml/metadata --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API: CreateIdentityProvider

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 us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

  其中 `details.json` 包含：

```
"ProviderDetails": {
      "MetadataFile": "<SAML metadata XML>",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

> **Note**
>
> 如果<SAML metadata XML>包含该字符的任何实例"，则必须添加\为转义字符：\"。

带元数据 URL 的示例：`aws cognito-idp update-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-details MetadataURL=https://myidp.example.com/sso/saml/metadata --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- AWS API: UpdateIdentityProvider

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 us-east-1_EXAMPLE --provider-name=SAML_provider_1`
- AWS API: DescribeIdentityProvider

列出所有相关信息 IdPs

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

  例如：`aws cognito-idp list-identity-providers --user-pool-id us-east-1_EXAMPLE --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 us-east-1_EXAMPLE --provider-name=SAML_provider_1`
- AWS API: DeleteIdentityProvider

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:us-east-1_EXAMPLE`。Amazon Cognito 要求受众限制值必须与 SAML 响应中的此 URN 相匹配。将您的 IdP 配置为使用以下 POST 绑定端点来发送 IdP 到 SP 的响应消息。

  ```
  https://mydomain.us-east-1.amazoncognito.com/saml2/idpresponse
  ```

- 您的 SAML IdP 必须在 SAML 断言中`NameID`填充用户池的所有必需属性。 `NameID`用于在用户池中唯一标识您的 SAML 联合用户。您的 IdP 必须以一致的、区分大小写的格式传递每个用户的 SAML 名称 ID。用户名 ID 值的任何变化都会创建一个新的用户个人资料。

向您的 SAML 2.0 IDP 提供签名证书

- 要从 Amazon Cognito 下载公钥副本，供您的 IdP 用来验证 SAML 注销请求，请选择用户池的登录体验选项卡，选择您的 IdP，然后在查看签名证书下选择下载为.crt。

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。

要删除 SAML 提供者

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. 选择登录体验选项卡，找到联邦身份提供商登录信息。
4. 选择要删除的 SAML IdPs 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）。

SAML 会话在 Amazon Cognito 用户池中启动

Amazon Cognito 支持服务提供商发起的（SP 发起的）单点登录 (SSO) 和 IDP 发起的 SSO。作为最佳安全实践，请在您的用户池中实施 SP 启动的 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。当您的 SAML 2.0 用户使用 SP 启动的流程进行身份验证时，他们必须始终首先向 Amazon Cognito 提出请求，然后重定向到 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。

您可以在用户池中配置 SAML IdP 以支持 IdP 启动的 SSO。当你支持 IDP 发起的身份验证时，Amazon Cognito 无法验证它是否已请求收到的 SAML 响应，因为 Amazon Cognito 不会通过 SAML 请求启动身份验证。在 SP 发起的 SSO 中，Amazon Cognito 会设置状态参数，以验证针对原始请求的 SAML 响应。通过 SP 发起的登录，您还可以防范跨站请求伪造 (CSRF)。

有关如何在不希望用户与用户池托管用户界面交互的环境中构建 SAM 的示例，请参阅。[示例场景：在企业控制面板中为 Amazon Cognito 应用程序添加书签]

主题

示例场景：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，这些控制面板为 Amazon Cognito 用户池提供 SSO 访问网络应用程序的权限。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，请在您的门户中添加一个登录书签，该书签按以下格式重定向到您的 Amazon Cognito 用户池。对端点授权

```
https://mydomain.us-east-1.amazoncognito.com/authorize?
response_type=code&identity_provider=MySAMLIdP&client_id=1example23456789&redire
www.example.com
```

> **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 标识符是您在用户池中创建身份提供商时可以配置的备用名称或电子邮件域。请参阅 SAML 身份提供商名称和标识符。

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。

首先，请在您的用户池中添加一个 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，请参阅用户池应用程序客户端。

在将这种经过身份验证的访问权限部署到您的门户之前，请测试 SP 发起的从托管用户界面登录您的应用程序。有关如何在 Amazon Cognito 中配置 SAML IdP 的更多信息，请参阅配置您的第三方 SAML 身份提供商。

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。

满足要求后，为你的创建一个书签对端点授权，其中包含identity_provider或idp_identifier参数。用户身份验证按以下步骤进行。

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 的链接的任何特征都可以使用。
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。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联合终端节点。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 的更多信息 IdPs，请参阅SAML 身份提供商名称和标识符。

   网址示例

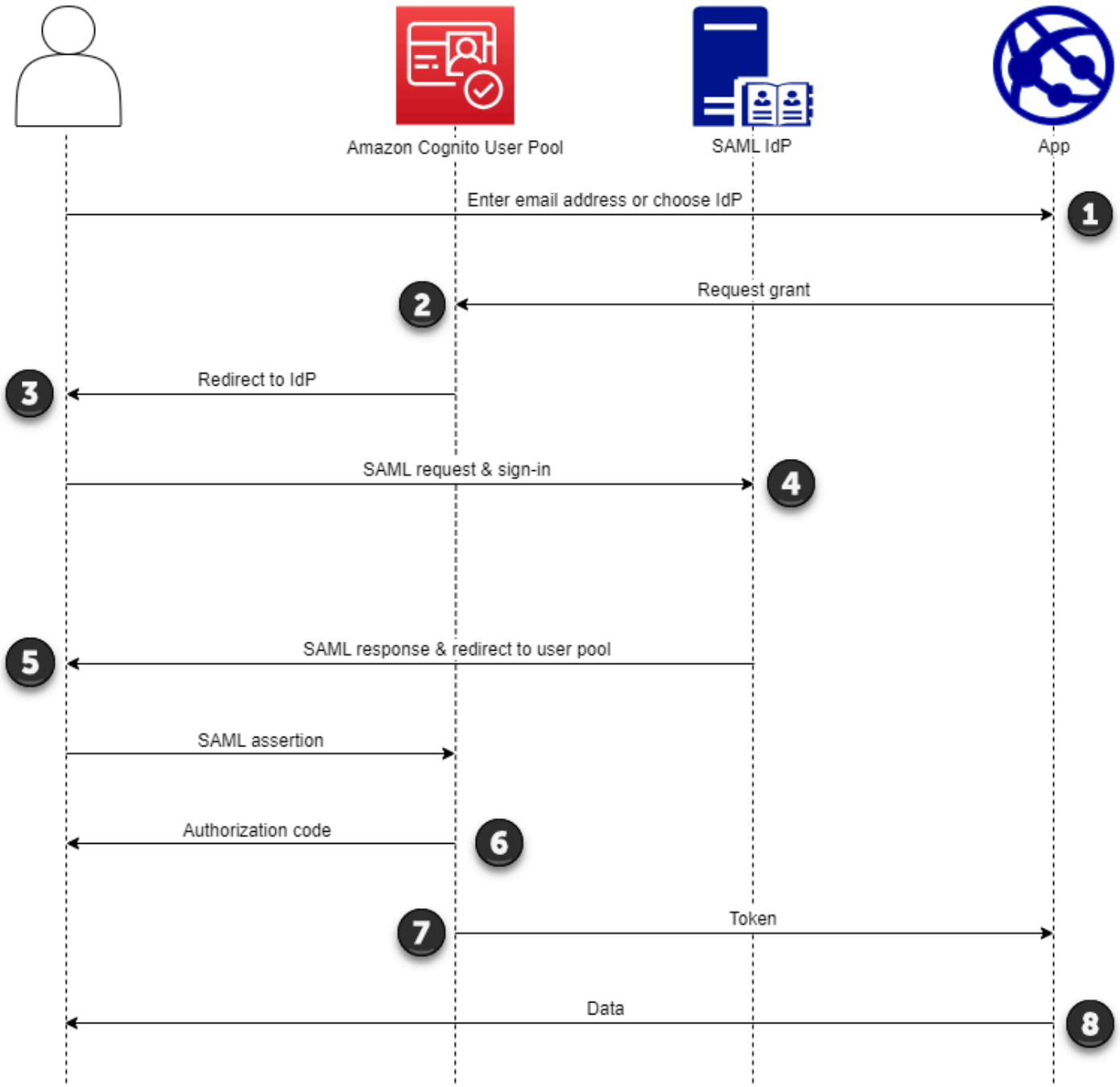
   ```
   https://mydomain.us-east-1.amazoncognito.com/authorize?
   response_type=code&
   identity_provider=MySAMLIdP&
   client_id=1example23456789&
   redirect_uri=https://www.example.com
   ```

4. Amazon Cognito 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。

使用 SP 启动的 SAML 登录

最佳做法是实现 service-provider-initiated （由 SP 发起的）用户池登录。Amazon Cognito 会启动您的用户的会话并将他们重定向到您的 IdP。使用这种方法，您可以最大限度地控制谁提出登录请求。在某些条件下，您也可以允许 IDP 发起的登录。有关更多信息，请参阅 SAML 会话在 Amazon Cognito 用户池中启动。

以下过程显示了用户如何通过 SAML 提供商登录您的用户池。



1. 您的用户在登录页面输入他们的电子邮件地址。要确定您的用户是否重定向到其 IdP，您可以在定制化应用程序中收集他们的电子邮件地址，或者在 Web 视图中调用托管 UI。您可以将托管 UI 配置为显示电子邮件地址列表 IdPs 或仅提示输入电子邮件地址。
2. 您的应用程序调用您的用户池重定向端点，并请求使用与应用程序对应的客户端 ID 和与用户对应的 IdP ID 进行会话。

3. Amazon Cognito 使用元素中的 SAML 请求（可选签名）将您的用户重定向到 IdP。`AuthnRequest`
4. IdP 以交互方式对用户进行身份验证，或者使用浏览器 cookie 中记住的会话对用户进行身份验证。
5. IdP 将您的用户重定向到您的用户池 SAML 响应端点，并在其 POST 有效负载中包含可选加密的 SAML 断言。

> **Note**
>
> Amazon Cognito 会取消在 5 分钟内未收到响应的会话，并将用户重定向到托管用户界面。当您的用户遇到此结果时，他们会收到一条`Something went wrong`错误消息。

6. 在验证 SAML 断言并映射响应中声明中的用户属性后，Amazon Cognito 会在内部创建或更新用户池中的用户个人资料。通常，您的用户池会向用户的浏览器会话返回授权码。
7. 您的用户向您的应用程序出示他们的授权码，您的应用程序会将该代码兑换 JSON 网络令牌 (JWT)。
8. 您的应用程序接受并处理用户的 ID 令牌作为身份验证，使用其访问令牌生成对资源的授权请求，并存储他们的刷新令牌。

当用户进行身份验证并收到授权码授予时，用户池会返回 ID、访问和刷新令牌。ID 令牌是基于 OIDC 的身份管理的身份验证对象。访问令牌是具有 OAuth 2.0 范围的授权对象。刷新令牌是一个在用户当前令牌过期时生成新 ID 和访问令牌的对象。您可以在用户池应用程序客户端中配置用户令牌的持续时间。

您还可以选择刷新令牌的持续时间。用户的刷新令牌到期后，他们必须重新登录。如果他们通过 SAML IdP 进行身份验证，则用户的会话持续时间由其令牌的到期时间而不是他们与 IdP 的会话到期时间来设置。您的应用程序必须存储每位用户的刷新令牌，并在刷新令牌到期时续订他们的会话。托管用户界面在有效期为 1 小时的浏览器 Cookie 中维护用户会话。

使用 IDP 发起的 SAML 登录

在为 IDP 发起的 SAML 2.0 登录配置身份提供商时，您可以向用户池域中的`saml2/idpresponse`终端节点提供 SAML 断言，而无需在上启动会话。对端点授权具有此配置的用户池接受请求的应用程序客户端支持的用户池外部身份提供商发起 IdP 发起的 SAML 断言。以下步骤描述了配置和使用 IDP 启动的 SAML 2.0 提供商登录的整个过程。

1. 创建或指定用户池和应用程序客户端。

2. 在您的用户池中创建一个 SAML 2.0 IdP。
3. 将您的 IdP 配置为支持 IdP 启动。IDP 发起的 SAML 引入了其他 SSO 提供商不受约束的安全注意事项。因此，您无法将非 SAML IdPs（包括用户池本身）添加到任何使用 SAML 提供商并通过 IDP 启动登录的应用程序客户端。
4. 将 IDP 发起的 SAML 提供商与用户池中的应用程序客户端相关联。
5. 将您的用户引导至 SAML IdP 的登录页面并检索 SAML 断言。
6. 使用 SAML 断言将您的用户引导到您的用户池`saml2/idpresponse`终端节点。
7. 接收 JSON 网络令牌 (JWT)。

要接受用户池中未经请求的 SAML 断言，必须考虑其对应用程序安全的影响。当您接受 IdP 发起的请求时，可能会出现请求欺骗和 CSRF 尝试。尽管您的用户池无法验证 IDP 发起的登录会话，但 Amazon Cognito 会验证您的请求参数和 SAML 断言。

此外，您的 SAML 声明不得包含`InResponseTo`索赔，并且必须在前 6 分钟内发出。

您必须使用 IDP 发起的 SAML 向您的提交请求。`/saml2/idpresponse`对于 SP 发起和托管的 UI 授权请求，您必须提供参数，将您请求的应用程序客户端、范围、重定向 URI 和其他详细信息标识为请求中的`HTTP GET`查询字符串参数。但是，对于 IDP 发起的 SAML 断言，您的请求的详细信息必须格式化为请求正文中的`RelayState`参数。`HTTP POST`请求正文还必须包含您的 SAML 断言作为参数。`SAMLResponse`

以下是对 IDP 发起的 SAML 提供商的请求示例。

```
POST /saml2/idpresponse HTTP/1.1
User-Agent: USER_AGENT
Accept: */*
Host: example.auth.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded

SAMLResponse=[Base64-encoded SAML assertion]&RelayState=identity_provider
%3DMySAMLIdP%26client_id%3D1example23456789%26redirect_uri%3Dhttps%3A%2F
%2Fwww.example.com%26response_type%3Dcode%26scope%3Demail%2Bopenid%2Bphone

HTTP/1.1 302 Found
Date: Wed, 06 Dec 2023 00:15:29 GMT
Content-Length: 0
x-amz-cognito-request-id: 8aba6eb5-fb54-4bc6-9368-c3878434f0fb
Location: https://www.example.com?code=[Authorization code]
```

AWS Management Console

为 IdP 启动的 SAML 配置 IdP

1. 创建用户池、应用程序客户端和 SAML 身份提供商。
2. 取消所有社交和 OIDC 身份提供商与您的应用程序客户端的关联（如果有）。
3. 导航到用户池的 “登录体验” 选项卡。
4. 在 “联合身份提供商登录” 下，编辑或添加 SAML 提供商。
5. 在 IDP 发起的 SAML 登录下，选择接受 SP 发起的、以及 IDP 发起的 SAML 断言。
6. 选择保存更改。

API/CLI

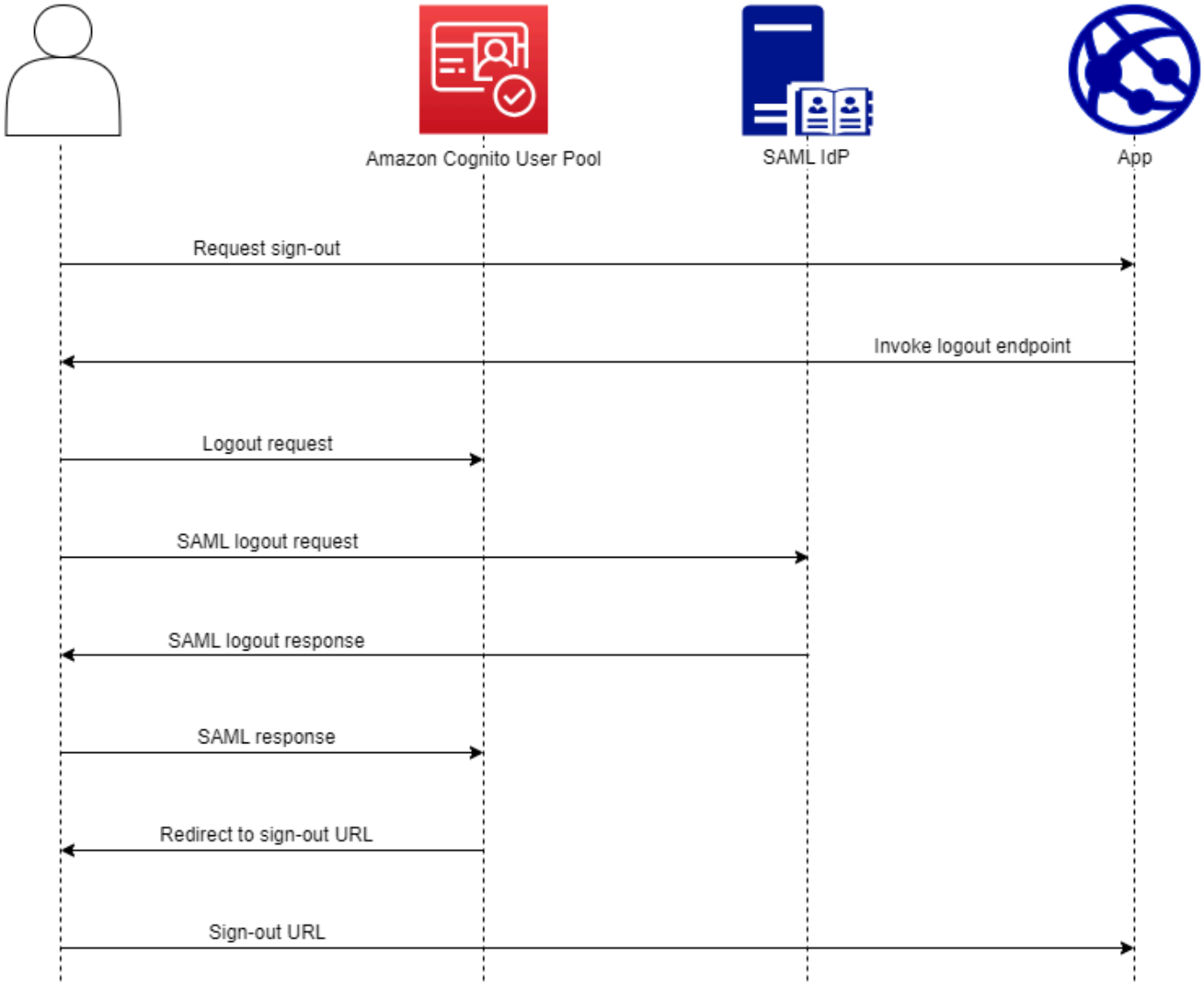
为 IdP 启动的 SAML 配置 IdP

使用CreateIdentityProvider或 UpdateIdentityProviderAPI 请求中的`IDPInit`参数配置 IDP 启动的 SAML。以下是支持 IdP 发起`ProviderDetails`的 SAML 的 IdP 示例。

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML 注销流程

亚马逊 Cognito 支持 SAML 2.0 单点注销。当您将 SAML IdP 配置为支持注销流程时，Amazon Cognito 会将您的用户通过已签名的 SAML 注销请求重定向到您的 IdP。Amazon Cognito 根据你的 IdP 元数据中的`SingleLogoutService`网址确定重定向位置。Amazon Cognito 使用您的用户池签名证书签署退出请求。



当您将具有 SAML 会话的用户定向到您的用户池`/logout`终端节点时，Amazon Cognito 会将您的 SAML 用户通过以下请求重定向到 IdP 元数据中指定的 SLO 终端节点。

```
https://[SingleLogoutService endpoint]?
SAMLRequest=[encoded SAML request]&
RelayState=[RelayState]&
SigAlg=http://www.w3.org/2001/04/xmldsig-more#rsa-sha256&
Signature=[User pool RSA signature]
```

然后，您的用户使用他们的 IdP 返回LogoutResponse到您的saml2/logout终端节点。您的 IdP 必须发送LogoutResponse请求。HTTP POST然后，Amazon Cognito 会将他们从最初的退出请求重定向到重定向目的地。

您的 SAML 提供商可能会发送LogoutResponse包含多个内容AuthnStatement的。此类响应sessionIndexAuthnStatement中的第一个必须与最初对用户进行身份验证的 SAML 响应sessionIndex中的相匹配。如果sessionIndex是其他会话AuthnStatement，Amazon Cognito 将无法识别该会话，您的用户也不会被注销。

AWS Management Console

配置 SAML 注销

1. 创建用户池、应用程序客户端和 SAML IdP。
2. 创建或编辑 SAML 身份提供商时，在“身份提供者信息”下，选中标题为“添加注销流程”的复选框。
3. 在用户池的登录体验选项卡中，在联合身份提供商登录下，选择您的 IdP 并找到签名证书。
4. 选择下载为.crt。
5. 将 SAML 提供商配置为支持 SAML 单点注销和请求签名，然后上传用户池签名证书。您的 IdP 必须重定向到您的用户池域/saml2/logout中。

API/CLI

配置 SAML 注销

使用CreateIdentityProvider或 UpdateIdentityProviderAPI 请求的IDPSignout参数配置单次注销。以下是支持 SAML ProviderDetails 单点注销的 IdP 示例。

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",,
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML 签名和加密

Amazon Cognito 支持签名的 SAML 请求以及用于登录和注销的加密 SAML 响应。用户池 SAML 操作期间的所有加密操作都必须使用 user-pool-provided Amazon Cognito 生成的密钥生成签名和密文。目前，您无法将用户池配置为使用外部密钥签署请求或接受加密断言。

> **Note**
>
> 您的用户池证书有效期为 10 年。Amazon Cognito 每年都会为您的用户池生成一次新的签名和加密证书。当您请求签名证书时，Amazon Cognito 会返回最新的证书，并使用最新的签名证书对请求进行签名。您的 IdP 可以使用任何未过期的用户池加密证书对 SAML 断言进行加密。您之前的证书在整个有效期内继续有效。作为最佳实践，请每年更新提供商配置中的证书。

主题

接受来自你的 IdP 的加密 SAML 响应

当用户登录和注销时，Amazon Cognito 和你的 IdP 可以在 SAML 响应中保密。Amazon Cognito 会向您在用户池中配置的每个外部 SAML 提供商分配一个公共 RSA 密钥对和一个证书。为用户池 SAML 提供者启用响应加密时，必须将证书上传到支持加密 SAML 响应的 IdP。在您的 IdP 开始使用提供的密钥加密所有 SAML 断言之前，您与 SAML IdP 的用户池连接无法正常工作。

以下是加密 SAML 登录流程的概述。

1. 您的用户开始登录并选择他们的 SAML IdP。
2. 您的用户池通过 SAML 登录请求对端点授权将您的用户重定向到他们的 SAML IdP。您的用户池可以选择在此请求中附上签名，该签名允许 IdP 进行完整性验证。要签署 SAML 请求时，必须将您的 IdP 配置为接受您的用户池使用签名证书中的公钥签署的请求。
3. SAML IdP 登录您的用户并生成 SAML 响应。IdP 使用公钥对响应进行加密，并将您的用户重定向到您的用户池终端节点。`/saml2/idpresponse`IdP 必须按照 SAML 2.0 规范的定义对响应进行加密。有关更多信息，请参阅 `Element <EncryptedAssertion>` OASIS 安全断言标记语言 (SAML) V2.0 的断言和协议。
4. 您的用户池使用私钥解密 SAML 响应中的密文并登录您的用户。

> **Important**
>
> 当您为用户池中的 SAML IdP 启用响应加密时，您的 IdP 必须使用该提供商特有的公钥对所有响应进行加密。Amazon Cognito 不接受来自您配置为支持加密的 SAML 外部 IdP 的未加密的 SAML 响应。

用户池中的任何外部 SAML IdP 都可以支持响应加密，并且每个 IdP 都会收到自己的密钥对。

AWS Management Console

配置 SAML 响应加密

1. 创建用户池、应用程序客户端和 SAML IdP。
2. 创建或编辑 SAML 身份提供商时，在“签署请求并加密响应”下，选中标题为“要求该提供商提供加密 SAML 断言”的复选框。
3. 在用户池的登录体验选项卡中，在联合身份提供商登录下，选择您的 SAML IdP，然后选择查看加密证书。
4. 选择下载为.crt，然后将下载的文件提供给您的 SAML IdP。将您的 SAML IdP 配置为使用证书中的密钥加密 SAML 响应。

API/CLI

配置 SAML 响应加密

使用CreateIdentityProvider或 UpdateIdentityProviderAPI 请求的`EncryptedResponses`参数配置响应加密。以下是支持请求签`ProviderDetails`名的 IdP 示例。

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

签署 SAML 请求

能够向你的 IdP 证明 SAML 2.0 请求的完整性是 Amazon Cognito SP 发起的 SAML 登录的一项安全优势。每个拥有域的用户池都会收到一个用户池 X.509 签名证书。使用此证书中的公钥，用户池将加密签名应用于您的用户池在用户选择 SAML IdP 时生成的注销请求。您可以选择将应用程序客户端配置为签署 SAML 登录请求。当您签署 SAML 请求时，您的 IdP 可以检查请求的 XML 元数据中的签名是否与您提供的用户池证书中的公钥相匹配。

AWS Management Console

配置 SAML 请求签名

1. 创建用户池、应用程序客户端和 SAML IdP。
2. 创建或编辑您的 SAML 身份提供商时，在 “签署请求并加密响应” 下，选中标题为 “向该提供商签署 SAML 请求” 的复选框。
3. 在用户池的登录体验选项卡中，在联合身份提供商登录下，选择查看签名证书。
4. 选择下载为.crt，然后将下载的文件提供给您的 SAML IdP。配置您的 SAML IdP 以验证传入的 SAML 请求的签名。

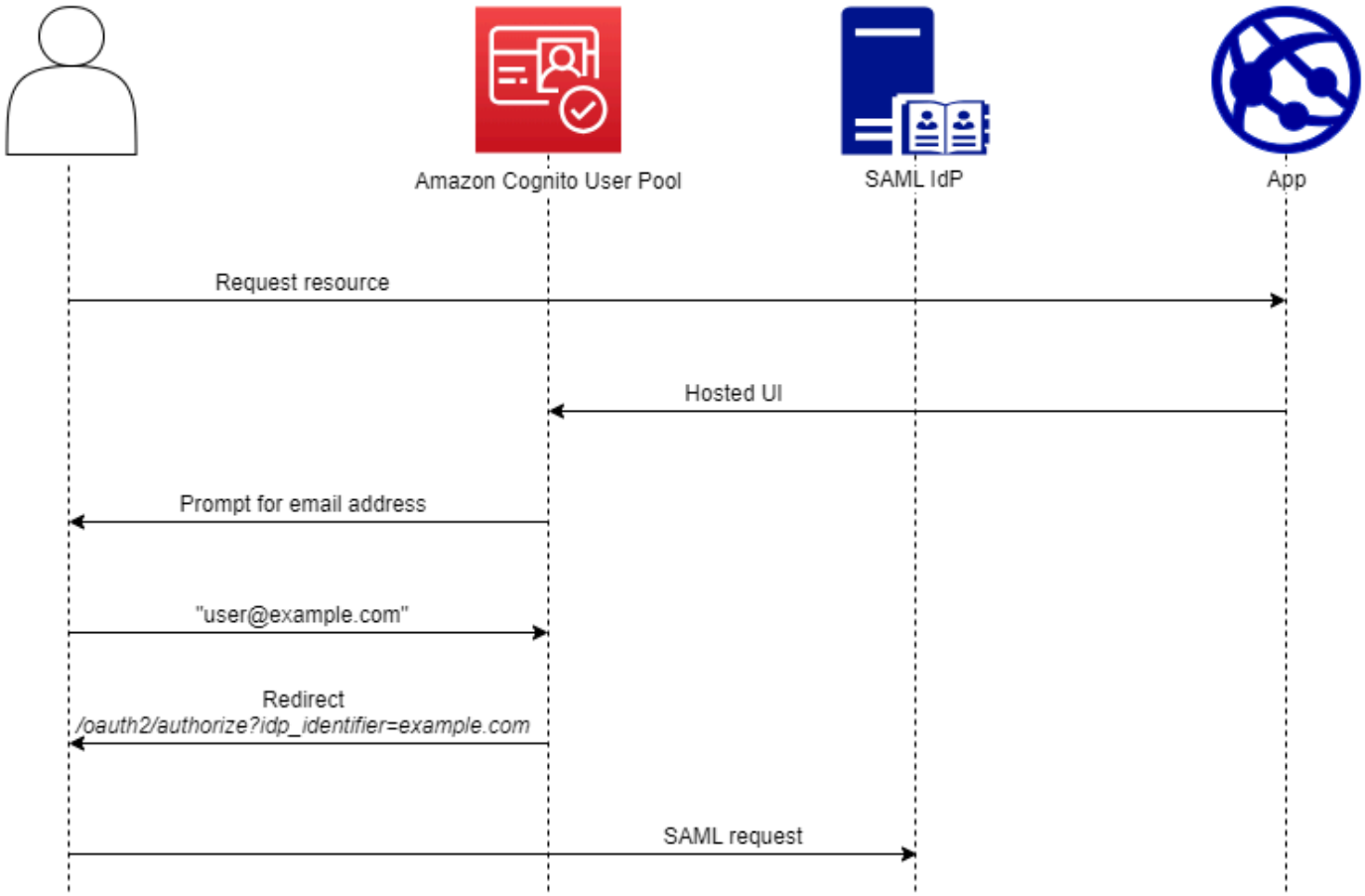
API/CLI

配置 SAML 请求签名

使用CreateIdentityProvider或 UpdateIdentityProviderAPI 请求的`RequestSigningAlgorithm`参数配置请求签名。以下是支持请求签`ProviderDetails`名的 IdP 示例。

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML 身份提供商名称和标识符



当您命名您的 SAML 身份提供商 (IdPs) 并分配 IdP 标识符时，您可以自动将 SP 发起的登录和注销请求流向该提供商。有关提供商名称的字符串约束的信息，请参阅的`ProviderName`属性CreateIdentityProvider。

您还可以为 SAML 提供商选择最多 50 个标识符。标识符是用户池中 IdP 的友好名称，在用户池中必须是唯一的。如果您的 SAML 标识符与用户的电子邮件域名匹配，Amazon Cognito 托管的 UI 会请求每个用户的电子邮件地址，评估其电子邮件地址中的域，然后将他们重定向到与其域名对应的 IdP。由于同一个组织可以拥有多个域，因此单个 IdP 可以有多个标识符。

无论您是否使用电子邮件域标识符，都可以在多租户应用程序中使用标识符将用户重定向到正确的 IdP。当你想完全绕过托管用户界面时，你可以自定义向用户提供的链接，这样他们就可以对端点授权直接重定向到他们的 IdP。要使用标识符登录您的用户并重定向到他们的 IdP，请在其初始授权请求的请求参数`idp_identifier=`*`myidp.example.com`*中加入该格式的标识符。

将用户传递给您的 IdP 的另一种方法是identity_provider使用以下 URL 格式的 IdP 名称填充参数。

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
identity_provider=MySAMLIdP&
client_id=1example23456789&
redirect_uri=https://www.example.com
```

用户使用您的 SAML IdP 登录后，您的 IdP 会将他们重定向到您的终端节点，并在正文中显示 SAML 响应。HTTP POST /saml2/idpresponseAmazon Cognito 会处理 SAML 断言，如果响应中的声明符合预期，则会重定向到您的应用程序客户端回传 URL。在您的用户以这种方式完成身份验证后，他们只与您的 IdP 和您的应用程序的网页进行了交互。

使用域格式的 IdP 标识符，Amazon Cognito 托管的 UI 会在登录时请求电子邮件地址，然后，当电子邮件域与 IdP 标识符匹配时，会将用户重定向到其 IdP 的登录页面。例如，您构建的应用程序需要两家不同公司的员工登录。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。在本示例中，您设置了两个 IdPs，每家公司一个，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，调用应用程序客户端的托管界面，以提示每位用户输入其电子邮件地址。Amazon Cognito 从电子邮件地址派生域，将该域与具有域标识符的 IdP 相关联，然后通过向包含请求参数的请求将您的用户重定向到正确的 IdP。对端点授权 idp_identifier例如，如果用户进入bob@exampleA.co.uk，则他们与之互动的下一个页面是 IdP 登录页面。https://auth.exampleA.co.uk/sso/saml

您也可以独立实现相同的逻辑。在您的应用程序中，您可以构建一个自定义表单，用于收集用户输入并根据自己的逻辑将其与正确的 IdP 关联起来。您可以为每个应用程序租户生成自定义应用程序门户，其中每个租户都链接到请求参数中包含租户标识符的授权端点。

要在托管 UI 中收集电子邮件地址并解析域名，请为分配给应用程序客户端的每个 SAML IdP 分配至少一个标识符。默认情况下，托管用户界面登录屏幕会为你分配给应用程序客户端 IdPs 的每个用户显示一个按钮。但是，如果您成功分配了标识符，则您的托管用户界面登录页面将如下图所示。

在托管 UI 中解析域名需要使用域名作为 IdP 标识符。如果您为应用程序客户端的每个 SAML IdPs 分配了任何类型的标识符，则该应用程序的托管界面将不再显示 IDP 选择按钮。当您打算使用电子邮件

解析或自定义逻辑生成重定向时，请为 SAML 添加 IdP 标识符。如果您想生成静默重定向，并且希望您的托管界面显示列表 IdPs，请不要分配标识符，也不要在授权`identity_provider`请求中使用请求参数。

- 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。
- 如果您为应用程序客户端激活的每个 SAML IdP 分配一个标识符，则在托管 UI 登录页面中会出现用户输入电子邮件地址的提示。
- 如果您有多个 IdPs ，但没有为所有用户分配标识符，则托管用户界面登录页面会显示一个按钮，用于使用每个分配的 IdP 进行登录。
- 如果您为自己分配了标识符， IdPs 并且希望托管界面显示精选的 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，或者创建一个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。如果您创建新的 IdP，则您的 SAML 用户将创建新的用户个人资料。活跃用户的重复可能会对您更改 IdP 配置的当月产生账单影响。

有关 IdP 设置的更多信息，请参阅为用户池配置身份提供商。

配置您的第三方 SAML 身份提供商

要将第三方 SAML 2.0 身份提供商 (IdP) 解决方案配置为与 Amazon Cognito 用户池的联合身份验证配合使用，您必须将 SAML IdP 配置为重定向到以下断言消费者服务 (ACS) 网址：。`https://`*`mydomain.us-east-1.amazoncognito.com`*`/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。

某些 SAML IdPs 要求您在表`urn:amazon:cognito:sp:`*`us-east-1_EXAMPLE`*单中`urn`提供（也称为受众 URI 或 SP 实体 ID）。您可以在 Amazon Cognito 控制台的用户池概述下找到您的用户池 ID。

您还必须将 SAML IdP 配置为用户池中指定为必需属性的任何属性提供值。通常，`email`是用户池的必需属性，在这种情况下，SAML IdP 必须在其 SAML 断言中提供某种形式的`email`声明，并且您必须将声明映射到该提供者的属性。

以下第三方 SAML 2.0 IdP 解决方案的配置信息是开始与 Amazon Cognito 用户池建立联合身份验证的好去处。有关最新信息，请直接查阅提供商的文档。

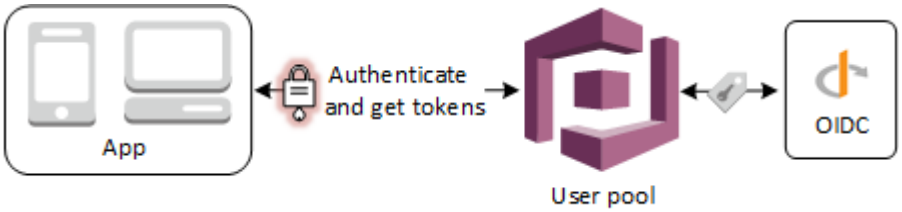
要签署 SAML 请求，您必须将 IdP 配置为信任由您的用户池签名证书签名的请求。要接受加密的 SAML 响应，必须将 IdP 配置为对用户池的所有 SAML 响应进行加密。您的提供商将提供有关配置这些功能的文档。有关微软的示例，请参阅配置微软 Entra SAML 令牌加密。

> **Note**
>
> Amazon Cognito 只需要您的身份提供者元数据文档。您的提供商可能会提供与 SAML 2.0 进行 AWS 账户 联合身份验证的配置信息；这些信息与 Amazon Cognito 集成无关。

解决方案	更多信息
Microsoft Active Directory 联合身份验证服务 (AD FS)	联邦元数据浏览器
Okta	如何下载用于 SAML 应用程序集成的 IdP 元数据和 SAML 签名证书
Auth0	将 Auth0 配置为 SAML 身份提供商
Ping 身份 (PingFederate)	从中导出 SAML 元数据 PingFederate
JumpCloud	SAML 配置注意事项
SecureAuth	SAML 应用程序集成

将 OIDC 身份提供商与用户池配合使用

您可以让已经拥有 OpenID Connect (OIDC) 身份提供商 (IdPs) 账户的用户跳过注册步骤，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。此特征不依赖于通过 Amazon Cognito 身份池（联合身份）实现的联合身份验证。

您可以通过 AWS Management Console、或使用用户池 API 方法将 OIDC IdP 添加到您的用户池中。AWS CLICreateIdentityProvider

主题

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。
  - 在密钥的 `jwks_uri` 处发布密钥 ID `kid` 声明，并在其令牌中包含 `kid` 声明。

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。

与 OIDC 的链接 IdPs

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 在用户池中使用社交身份提供商。

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://mydomain.us-east-1.amazoncognito.com/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://mydomain.us-east-1.amazoncognito.com/authorize?
response_type=code&client_id=<your_client_id>&redirect_uri=https://
www.example.com&identity_provider=CorpSalesforce
```

c. 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://mydomain.us-east-1.amazoncognito.com/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，请输入您的 AWS 凭据。
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 CreateIdentityProviderAPI 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。

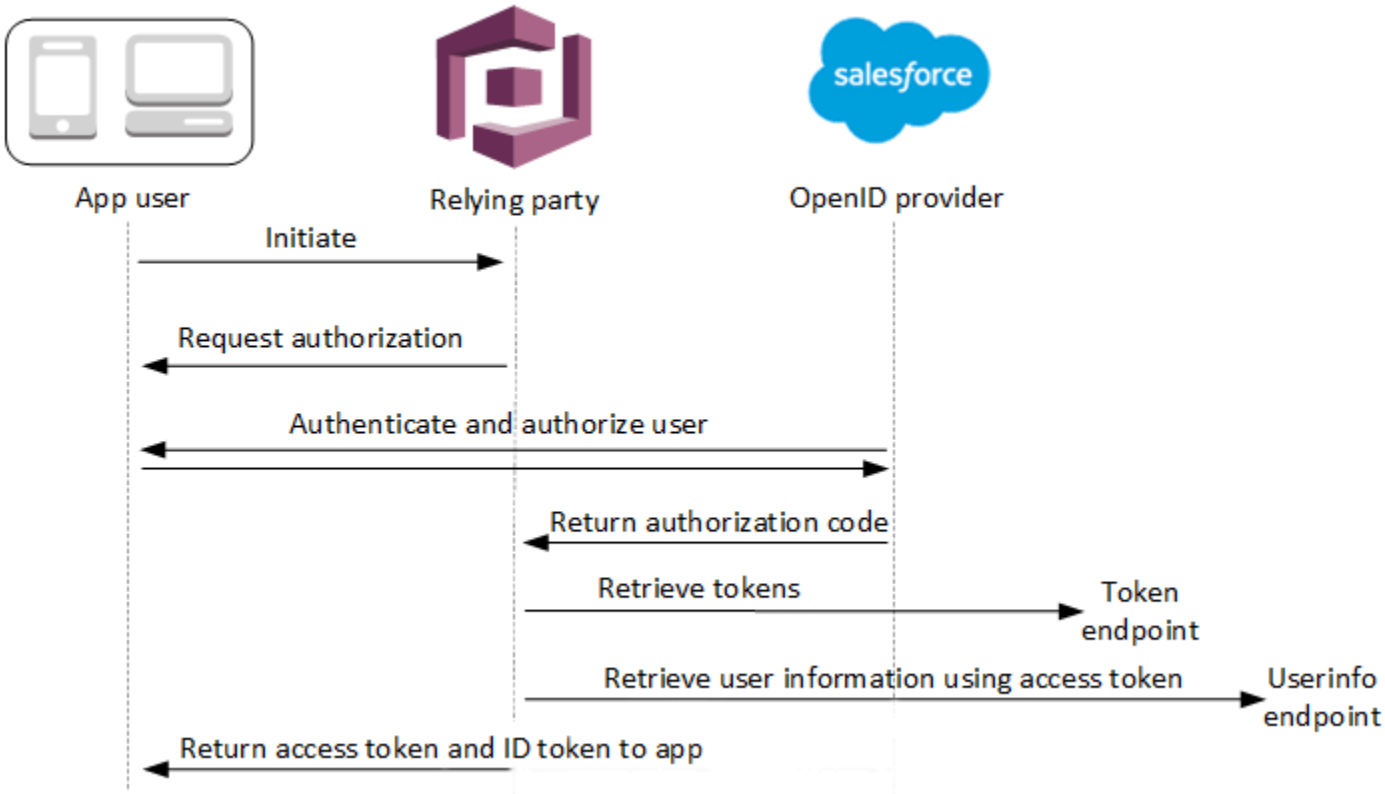
```
https://mydomain.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

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 OAuth 2.0 之上的一个身份层，它指定由 IdPs OIDC 客户端应用程序（依赖方）颁发的 JSON 格式 (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。

当用户使用授权码授予进行身份验证时，用户群体将返回 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。有关您的用户群体应用程序客户端可以支持的授权类型的更多信息，请参阅对端点授权。

用户群体如何处理来自 OIDC 提供者的声明

当用户通过第三方 OIDC 提供者完成登录时，Amazon Cognito 托管 UI 会从 IdP 检索授权码。用户群体会与 IdP 的 `token` 端点交换访问令牌和 ID 令牌的授权码。用户群体不会将这些令牌传递给用户或应用程序，而是使用它们来构建用户配置文件，其中包含用户群体在声明中以其自己的令牌表示的数据。

Amazon Cognito 不会独立验证访问令牌。相反，它会从提供者 `userInfo` 端点请求用户属性信息，如果令牌无效，则该请求会被拒绝。

Amazon Cognito 通过以下检查来验证提供者 ID 令牌：

1. 检查提供者是否使用以下集合中的算法对令牌进行了签名：RSA、HMAC、椭圆曲线。
2. 如果提供者使用非对称签名算法对令牌进行了签名，请检查令牌 `kid` 声明中的签名密钥 ID 是否在提供者 `jwks_uri` 端点上列出。
3. 根据提供者元数据，将 ID 令牌签名与预期的签名进行比较。
4. 将 `iss` 声明与为 IdP 配置的 OIDC 颁发者进行比较。
5. 比较 `aud` 声明是否与在 IdP 上配置的客户端 ID 相匹配，或者，如果 `aud` 声明中有多个值，则声明包含所配置的客户端 ID。
6. 检查 `exp` 声明中的时间戳不早于当前时间。

用户群体会验证 ID 令牌，然后尝试使用提供者访问令牌向提供者 `userInfo` 端点发出请求。此请求检索访问令牌中的范围授权它读取的任何用户配置文件信息。然后，用户群体将搜索您在用户群体中根据需要设置的用户属性。您必须在提供者配置中为必需的属性创建属性映射。用户群体会检查提供者 ID 令牌和 `userInfo` 响应。用户群体将所有与映射规则匹配的声明写入用户群体用户配置文件中的用户属性。用户群体会忽略与映射规则匹配、但不是必需且在提供者的声明中找不到的属性。

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 为用户池的身份提供商 (IdP) 指定属性映射。

有关映射的需知信息

在开始设置用户属性映射之前，请查看以下重要细节。

- 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。
- 您可以将身份提供者（IdP）令牌映射到用户群体中的自定义属性。社交提供者提供访问令牌，而 OIDC 提供者提供访问令牌和 ID 令牌。要映射令牌，请添加一个最大长度为 2048 个字符的自定义属性，向您的应用程序客户端授予对该属性的写入权限，然后将 `access_token` 或 `id_token` 从 IdP 映射到自定义属性。
- 对于每个映射的用户群体属性，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。当令牌长度超过 2048 个字符时，Amazon Cognito 不支持将 IdP 令牌映射到自定义属性。
- Amazon Cognito 从您的联合 IdP 通过的特定声明中衍生出联合用户个人资料中的`username`属性，如下表所示。例如，Amazon Cognito 会在这个属性值前面加上你的 IdP 的名字。`MyOIDCIdP_[sub]`如果您希望您的联合用户拥有与外部用户目录中的属性完全匹配的属性，请将该属性映射到 Amazon Cognito 登录属性，例如。`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`

身份提供商	**username** 资源属性
OpenID Connect (OIDC) 提供者	`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。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 CreateUserPoolAPI 操作创建用户池，则可以将每个属性的`Mutable`参数设置为`true`。如果您的 IdP 发送映射的不可变属性的值，Amazon Cognito 会返回错误并登录失败。
  - 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。如果您的 IdP 为映射的不可写属性发送值，则 Amazon Cognito 不会设置该属性值，而是继续进行身份验证。
- 当 IdP 属性包含多个值时，Amazon Cognito 会将所有值扁平化为一个以逗号分隔的字符串，并且 URL 对包含非字母数字字符（不包括 '、'、'、' 和 .、" 字符）的值进行格式编码。- * _在您的应用程序中使用这些值之前，您必须解码并解析各个值。

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

您可以使用为您的用户 AWS Management Console 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，请输入您的 AWS 凭据。
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，请参阅在用户池中使用社交身份提供商。
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，请输入您的 AWS 凭据。
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，请参阅在用户池中使用 SAML 身份提供商。
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。有些 IdPs 使用简单的名称，例如`email`，而另一些则使用类似于 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）。

为您的用户池（AWS CLI 和 AWS API）指定身份提供商属性映射

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 (\")。

  元数据 URL 示例：

```
aws cognito-idp create-identity-provider \
--user-pool-id us-east-1_EXAMPLE \
--provider-name=SAML_provider_1 \
--provider-type SAML \
--provider-details MetadataURL=https://myidp.example.com/saml/metadata \
```

```
--attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/
emailaddress
```

- AWS API: CreateIdentityProvider

指定现有 IdP 的属性映射

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

  例如：`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`

  例如：`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，同一个用户的个人资料包含多个身份提供商 (IdPs)，您已将其连接到您的用户池。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让 Amazon Cognito 将您的联合目录中用户的唯一 ID 识别为用户池中的用户。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。

当联合用户首次登录您的用户群体时，Amazon Cognito 会查找您已关联到其身份的本地个人资料。如果不存在关联的个人资料，则您的用户群体会创建一个新的个人资料。您可以在联合用户首次登录之前随时在 `AdminLinkProviderForUser` API 请求中创建本地个人资料并将其关联到他们，无论是在计划的预部署任务中还是在计划中的预配置任务中，还是在中。注册前 Lambda 触发器在您的用户登录且 Amazon Cognito 检测到关联的本地个人资料后，您的用户群体会读取用户的声明，并将其与 IdP 的映射规则进行比较。然后，您的用户群体会使用他们登录时映射的声明来更新关联的本地个人资料。通过这种方式，您可以为本地配置文件配置访问权限声明，并将其身份声明保留给您的 up-to-date 提供商。在 Amazon Cognito 将您的联合用户与关联的个

人资料匹配后，他们将始终登录该个人资料。然后，您可以将用户的更多提供者身份关联到同一个人资料，从而为一位客户提供一致的应用体验。要关联之前登录过的联合用户，必须先删除其现有个人资料。您可以通过其格式识别现有个人资料：*[Provider name]*_identifier。例如，LoginWithAmazon_amzn1.account.AFAEXAMPLE。您创建并关联到第三方用户身份的用户具有创建时使用的用户名和包含其关联身份详细信息的identities属性。

> **Important**
>
> 由于AdminLinkProviderForUser允许具有外部联合身份的用户以用户池中的现有用户身份登录，因此必须仅将其与应用程序所有者信任的外部 IdPs 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。

你的 ADFS 在 IdPs 向亚马逊 Cognito 提出的 S msp_carlos@example.com AML email 声明中出示了 Carlos 的电子邮件地址。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) 命令链接了 IdPs ADFS1、ADFS2 和 ADFS3 中卡洛斯的身份。

> **Note**
>
> 您可以根据特定属性声明关联用户。这种能力是 OIDC 和 SAML 所独有的。 IdPs 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。
- 您无法在中将提供商链接到用户个人资料 AWS Management Console。
- 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 API 请求中为自动创建的联合用户配置文件设置密码。AdminSetUserPassword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验证的 API 请求中修改自己的密码和属性，例如和。ChangePasswordUpdateUserAttributes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。

当您得知用户不再使用您已关联到其个人资料的外部用户账户时，您可以取消该用户账户与您的用户群体用户的关联。当您关联用户时，您会在请求中提供用户的属性名称、属性值和提供者名称。要删除用户不再需要的配置文件，请使用等效参数发出 AdminDisableProviderForUserAPI 请求。

AdminLinkProviderForUser有关 AWS 软件开发工具包中的其他命令语法和示例，请参阅。

使用 Lambda 触发器自定义用户池工作流

Amazon Cognito 使用 AWS Lambda 函数来修改用户群体的身份验证行为。您可以将您的用户群体配置为在用户首次注册之前、完成身份验证之后以及两者之间的几个阶段自动调用 Lambda 函数。您的函数可以修改身份验证流程的默认行为，发出 API 请求以修改您的用户群体或其他 AWS 资源，以及与外部系统通信。您的 Lambda 函数中的代码是您自己的。Amazon Cognito 会将事件数据发送到您的函数，等待函数处理数据，而且在大多数情况下，预计会出现一个响应事件，该事件反映您要对会话进行的任何更改。

在请求和响应事件系统中，您可以引入自己的身份验证质询、在用户群体与其他身份存储之间迁移用户、自定义消息以及修改 JSON Web 令牌 (JWT)。

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。他们还可以对您的 AWS 环境、外部 API、数据库或身份存储执行运行时操作。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	操作	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”	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”	记录自定义分析的事件
	the section called “令牌生成前 Lambda 触发器”	增加或隐藏令牌声明
注册	the section called “注册前 Lambda 触发器”	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”	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”	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”	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”	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”	使用第三方提供商发送 SMS 和电子邮件

主题

重要注意事项

在为 Lambda 函数准备用户群体时，请考虑以下各项：

- Amazon Cognito 发送到 Lambda 触发器的事件可能会随着新功能推出而发生变化。响应和请求元素在 JSON 层次结构中的位置可能改变，或者可能会添加元素名称。在 Lambda 函数中，您预期会收到本指南中介绍的输入元素键值对，但是更严格的输入验证可能会导致您的函数失败。
- 您可以从 Amazon Cognito 发送到某些触发器的多个事件版本中选择一个。某些版本可能需要您接受对 Amazon Cognito 定价的更改。有关定价的更多信息，请参阅 Amazon Cognito 定价。要在令牌生成前 Lambda 触发器 中自定义访问令牌，您必须使用高级安全功能配置用户群体，并更新 Lambda 触发器配置以使用事件版本 2。
- Amazon Cognito 会同步调用 Lambda 函数，但 自定义发件人 Lambda 触发器 除外。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。

- 您无法在 Lambda 触发器配置中声明函数版本。默认情况下，Amazon Cognito 用户群体会调用您的函数的最新版本。但是，您可以将函数版本与别名相关联，并将触发器 `LambdaArn` 设置为 CreateUserPool 或 UpdateUserPool API 请求中的别名 ARN。此选项在AWS Management Console 中不可用。要了解有关别名的更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 函数别名。
- 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 Lambda 函数没有向 Amazon Cognito 返回请求和响应参数，或者返回错误，则身份验证事件将无法成功。您可以在函数中返回错误，以阻止用户注册、身份验证、令牌生成或其身份验证流程中任何其他调用 Lambda 触发器的阶段。

  Amazon Cognito 托管 UI 将返回 Lambda 触发器生成的错误作为登录提示上方的错误文本。Amazon Cognito 用户群体 API 以 *`[trigger]`* `failed with error` *`[error text from response]`* 格式返回触发器错误。最佳做法是，仅在 Lambda 函数中生成您希望用户看到的错误。使用输出方法（如 `print()`）将任何敏感或调试信息记录到 CloudWatch Logs 中。有关示例，请参阅注册前示例：如果用户名少于五个字符，则拒绝注册。
- 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”。

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。

region

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 用户池 API 和用户池端点。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

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`
	自定义 SMS 发送人	`CustomSMSSender_AdminCreateUser`

API 操作	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
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

API 操作	Lambda 触发器	触发器源
	自定义电子邮件发件人	CustomEmailSender_AccountTakeOverNotification
	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_ConfirmForgotPassword
UpdateUserAttributes AdminUpdateUserAttributes	自定义消息	CustomMessage_UpdateUserAttribute
	自定义电子邮件发件人	CustomEmailSender_UpdateUserAttribute
	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_VerifyUserAttribute

API 操作	Lambda 触发器	触发器源
	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 中的本地用户触发器源

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

托管 UI URI	Lambda 触发器	触发器源
		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_AccountTakeOverNotification
	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_ConfirmForgotPassword

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 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、迁移用户 Lambda 触发器、自定义消息 Lambda 触发器 或者自定义发件人 Lambda 触发器。

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	赛事
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	赛事
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。
验证身份验证质询	`VerifyAuthChallengeResponse_Authentication`	验证身份验证质询响应。

令牌生成前触发器

触发器	triggerSource 值	赛事
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`	结束用户设备身份验证。
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	赛事
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	赛事
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。

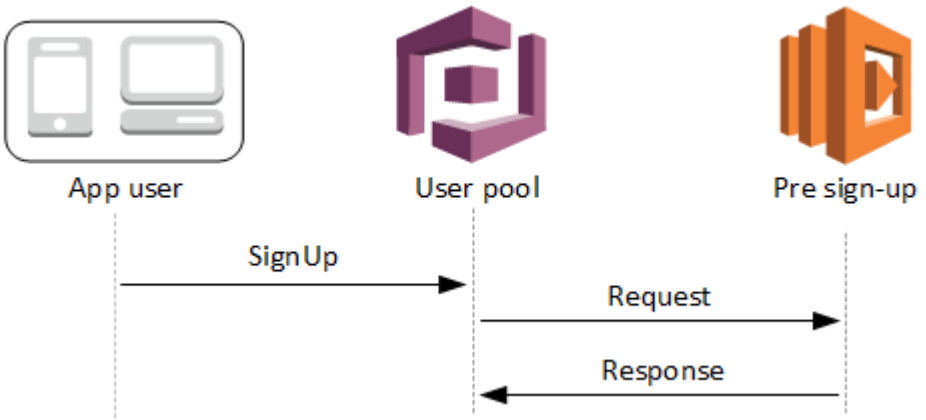
注册前 Lambda 触发器

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。

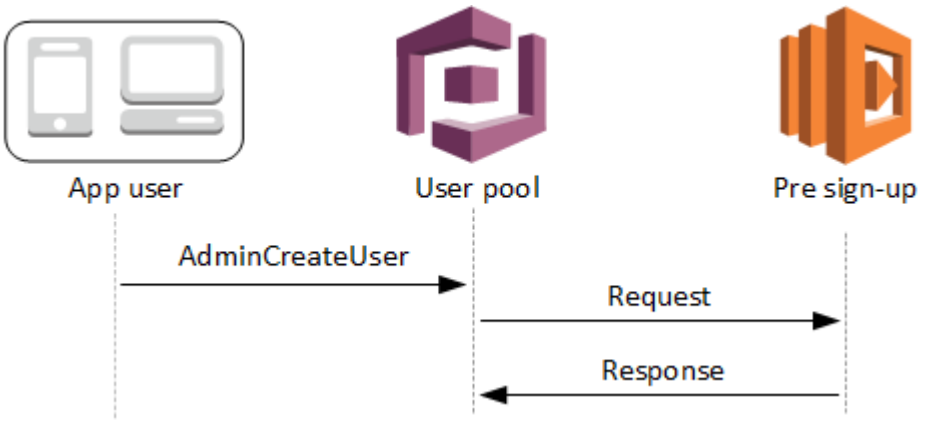
主题

注册前 Lambda 流程

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。这些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

一个或多个包含用户属性数据的键值对，您的应用程序在创建新用户的请求中将这些数据传递给 Amazon Cognito。在您AdminCreateUser或 SignUpAPI 请求的 ValidationData 参数中将此信息发送到您的 Lambda 函数。

Amazon Cognito 不会将你的 ValidationData 数据设置为你创建的用户的属性。 ValidationData 是您为注册前 Lambda 触发器而提供的临时用户信息。

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您的 Lambda 函数：AdminCreateUser、AdminRespondToAuthChallengeForgotPassword、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 函数时，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。

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。

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安卓和 iOS 的 Amazon Cognito 注册教程。

平台	教程
JavaScript 身份软件开发工具包	使用注册用户 JavaScript
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。这是一个演示如何注册新用户的示例 JavaScript 程序。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。为了防止这种情况发生，您可以使用用户池 ListUsers API 来查看是否有现有用户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

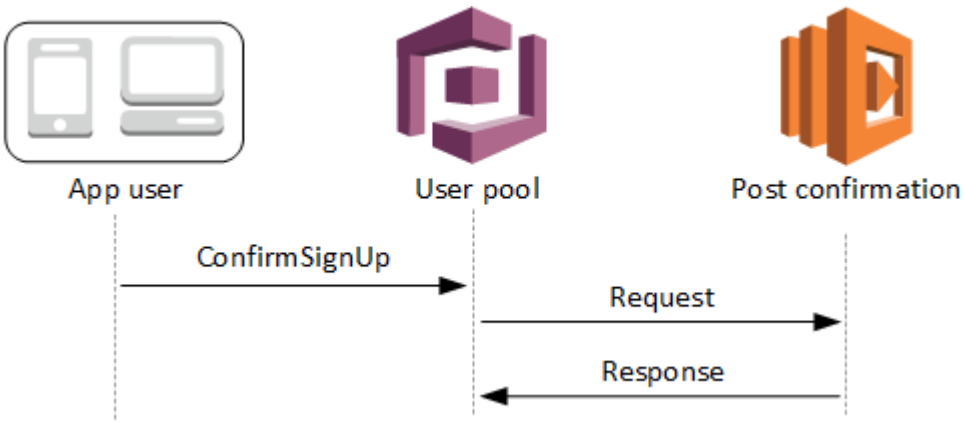
Amazon Cognito 会在注册用户确认其用户账户后调用此触发器。在您的确认后 Lambda 函数中，您可以发送自定义消息或添加自定义 API 请求。例如，您可以查询外部系统并为用户填充其他属性。Amazon Cognito 仅对在您的用户群体中注册的用户调用此触发器，而不会针对您使用管理员凭证创建的用户账户调用此触发器。

请求包含已确认用户的当前属性。

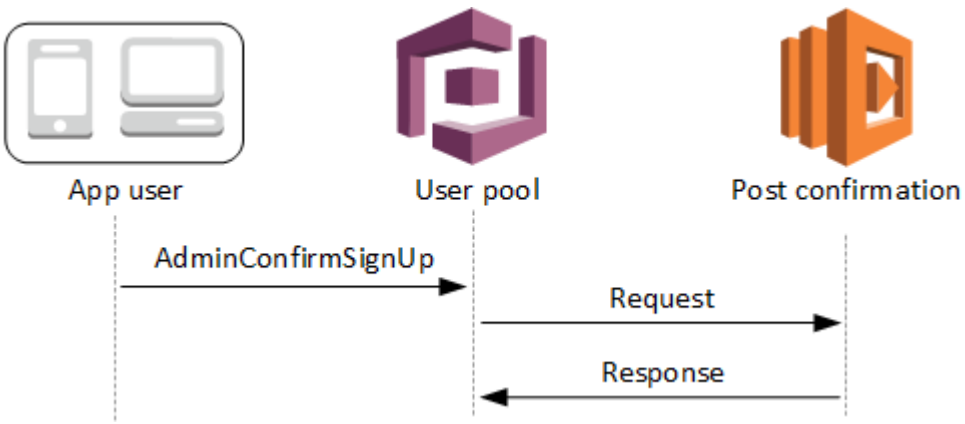
主题

确认后 Lambda 流程

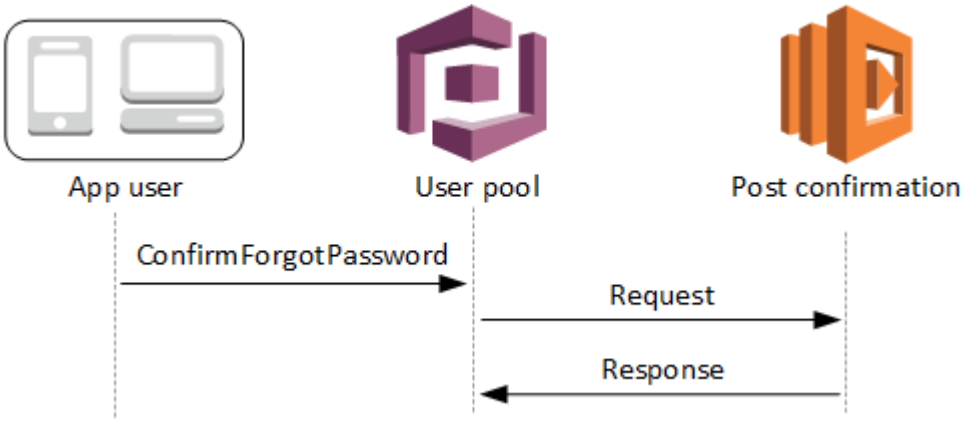
客户端确认注册流程



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

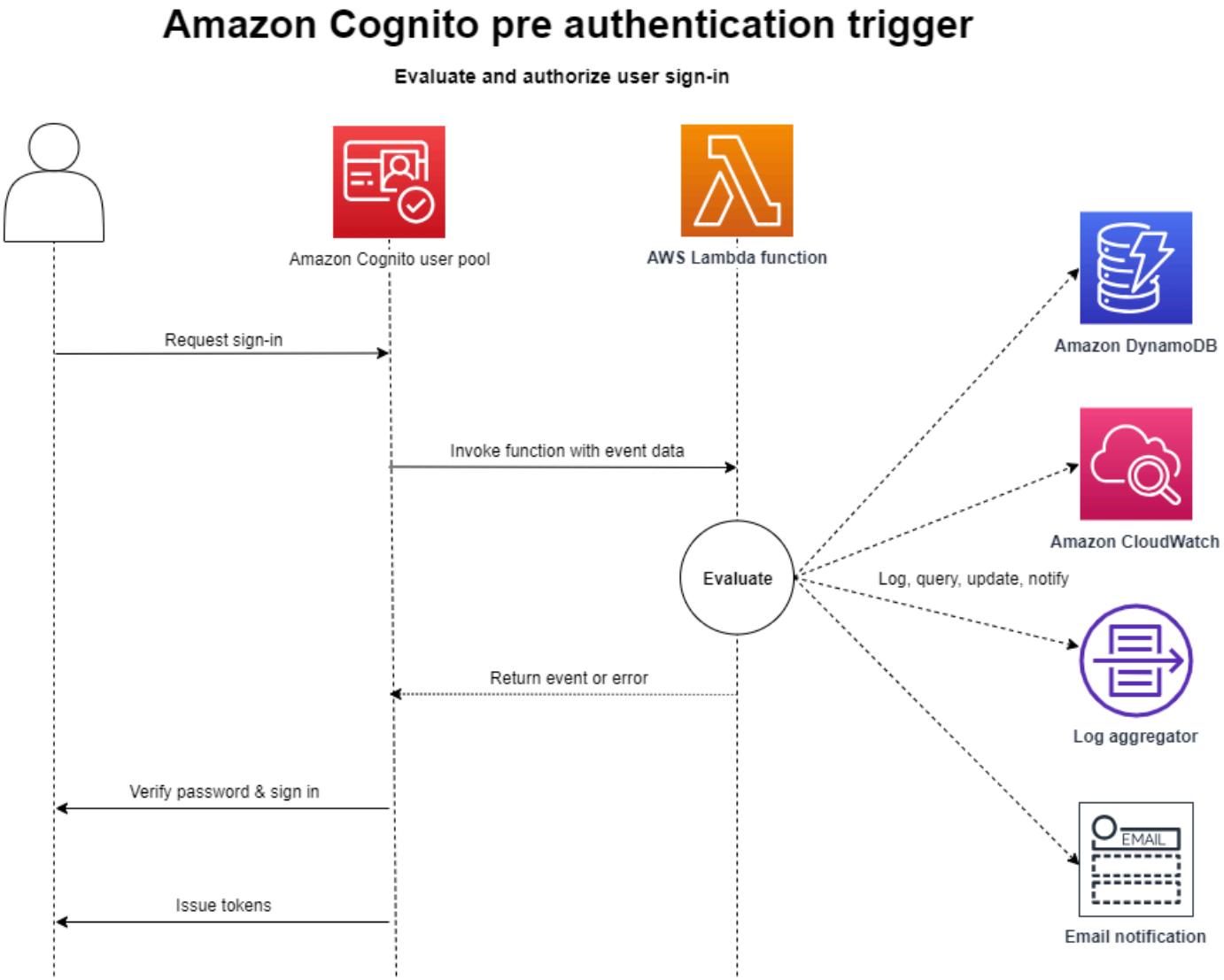
当用户尝试登录时，Amazon Cognito 会调用此触发器，以便您可以创建用于执行准备操作的自定义验证。例如，您可以拒绝身份验证请求或将会话数据记录到外部系统。

> **Note**
>
> 当用户不存在或您的用户群体中已经有会话时，此 Lambda 触发器不会激活。如果将用户群体应用程序客户端的 `PreventUserExistenceErrors` 设置设为 `ENABLED`，则 Lambda 触发器将会激活。

主题

- 身份验证流概述
- 身份验证前 Lambda 触发器参数
- 身份验证前示例

身份验证流概述



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群体 `InitiateAuth` 和 `AdminInitiateAuth` API 操作。

有关更多信息，请参阅用户池身份验证流程。

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": {}
```

```
}
```

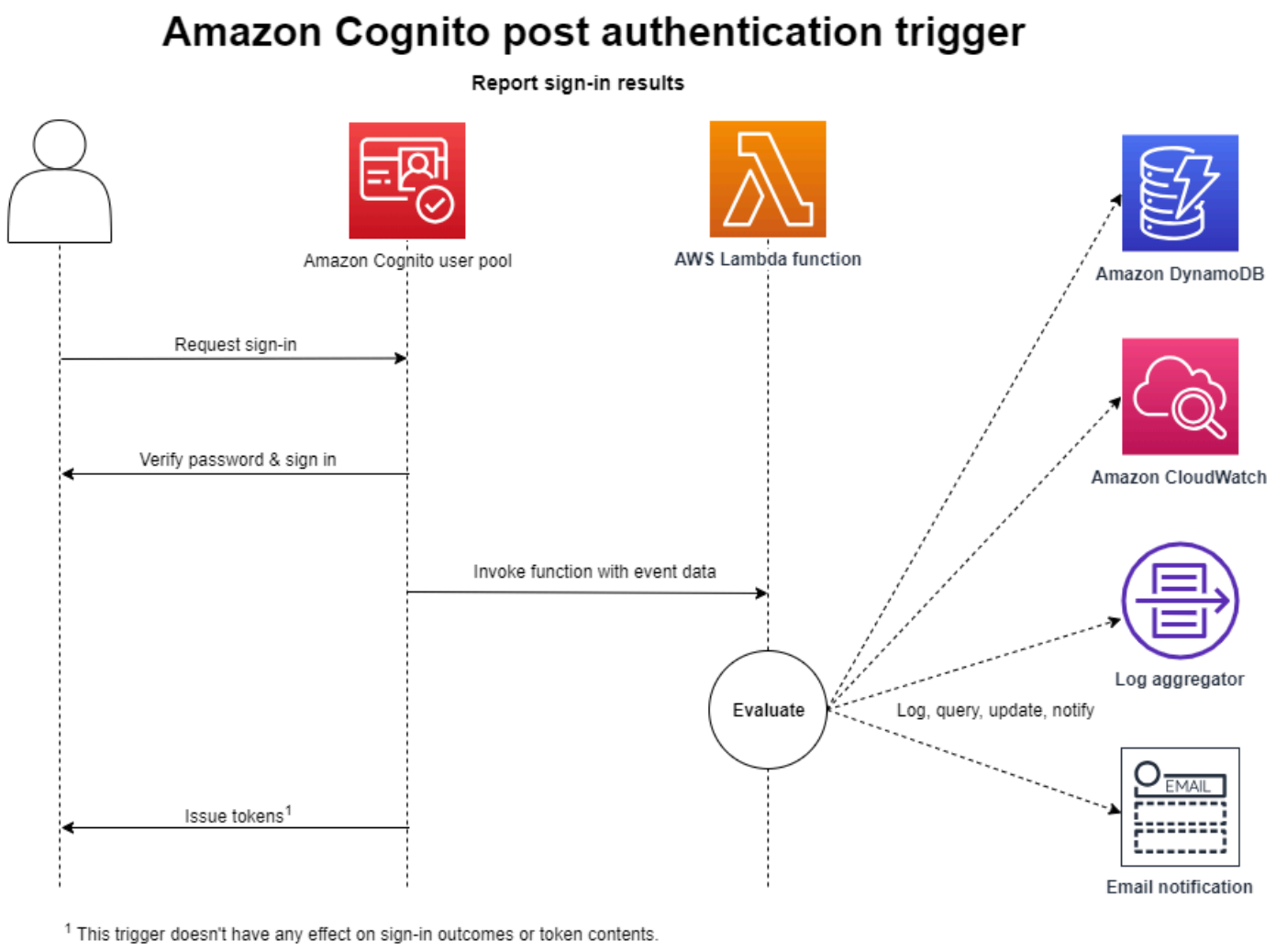
身份验证后 Lambda 触发器

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

- 身份验证流概述
- 身份验证后 Lambda 触发器参数
- 身份验证教程
- 身份验证后示例

身份验证流概述



有关更多信息，请参阅用户池身份验证流程。

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

```
import os
def lambda_handler(event, context):

    # Send post authentication data to Cloudwatch logs
    print ("Authentication successful")
    print ("Trigger function =", event['triggerSource'])
    print ("User pool = ", event['userPoolId'])
    print ("App client ID = ", event['callerContext']['clientId'])
    print ("User ID = ", event['userName'])
```

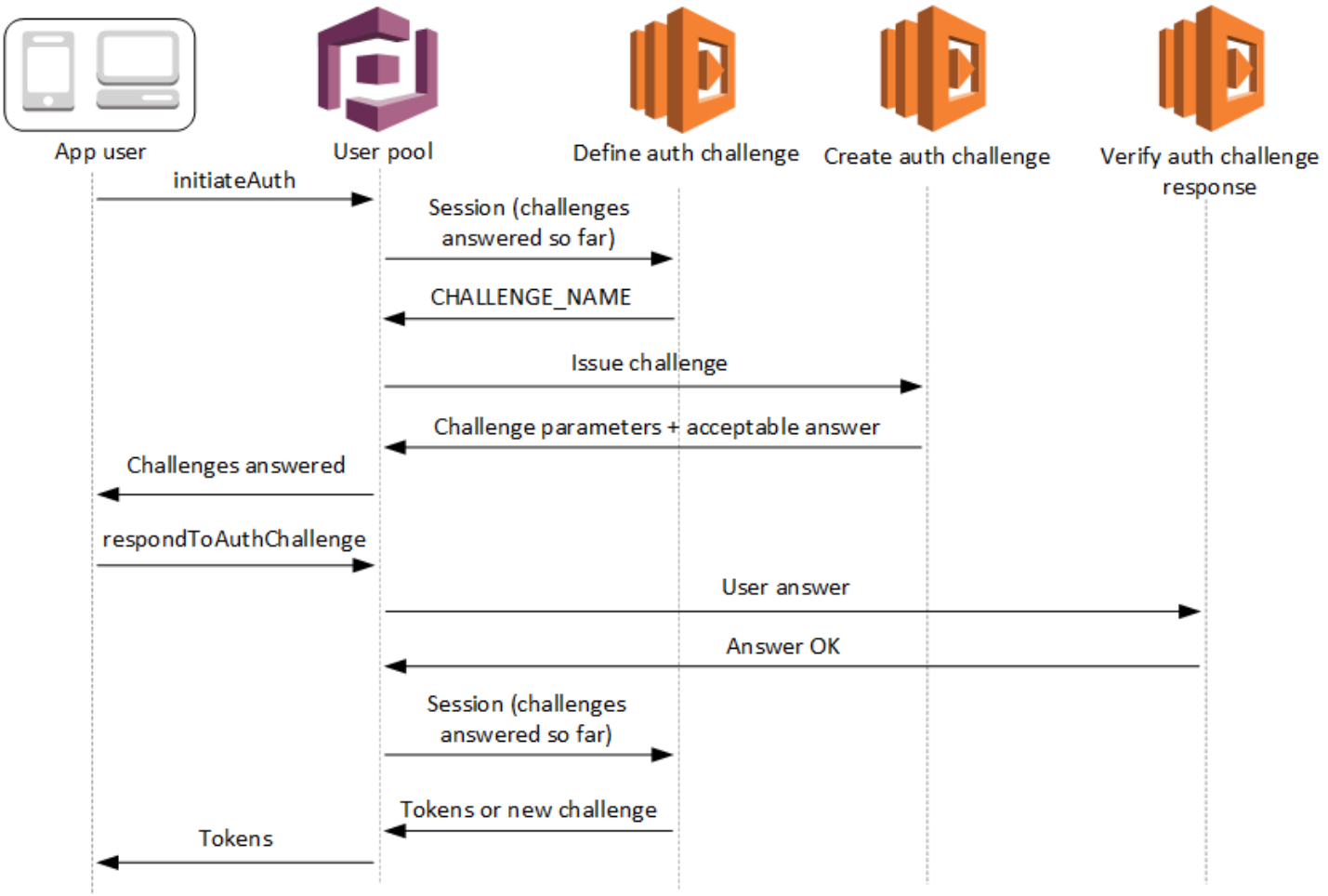
```
    # Return to Amazon Cognito
    return event
```

Amazon Cognito 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。

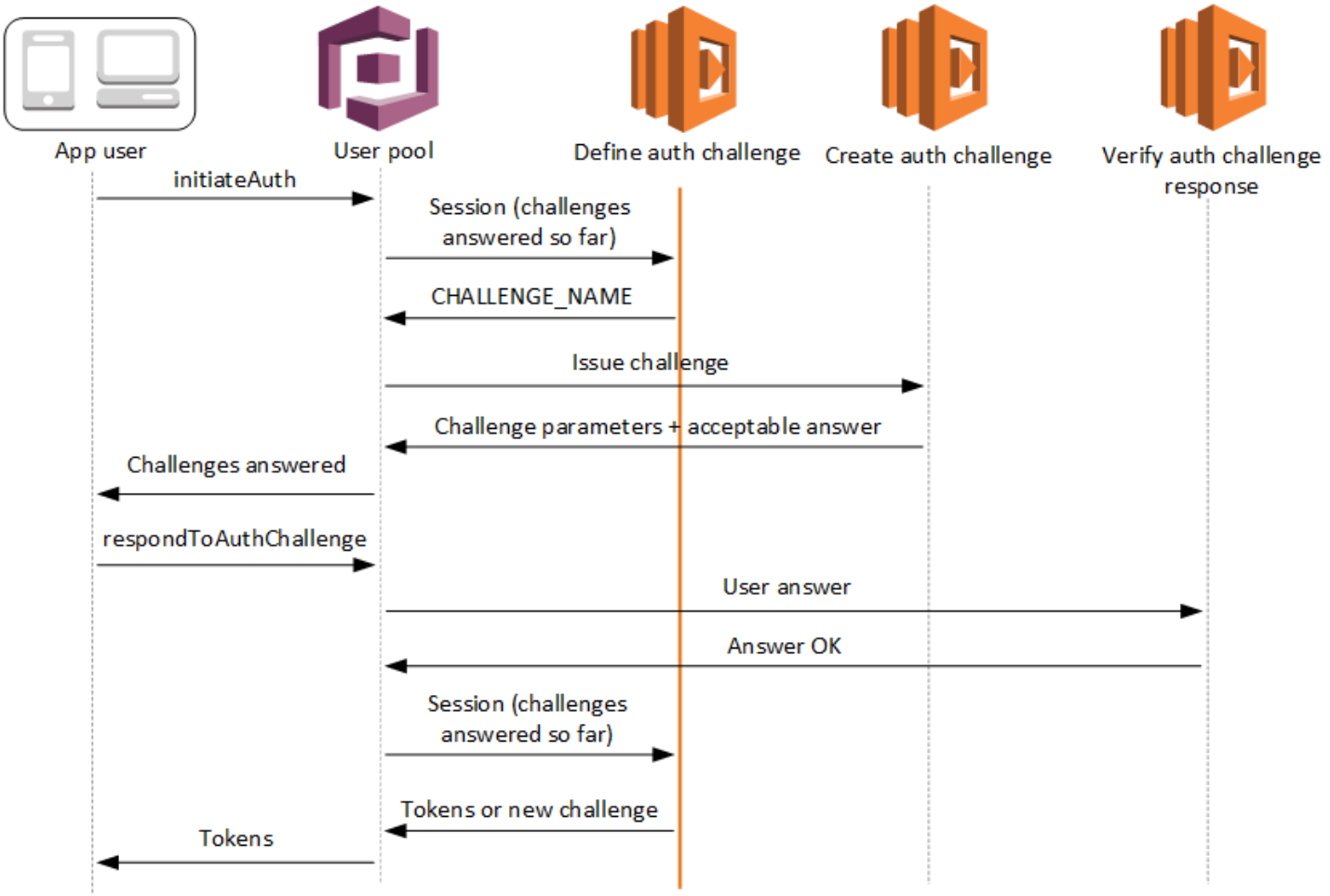
在此流程中，用户通过回答连续的质询进行身份验证，直到身份验证失败或用户获得令牌。这两个 API 调用可重复执行以包含不同的质询。

> **Note**
>
> Amazon Cognito 托管 UI 不支持使用自定义身份验证质询 Lambda 触发器进行自定义身份验证。

主题

- 定义身份验证质询 Lambda 触发器
- 创建身份验证质询 Lambda 触发器
- 验证身份验证质询响应 Lambda 触发器

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。当您在自定义质询流程中执行 SRP 身份验证时，请求频率限额身份验证类别中关联的任何 Lambda 触发器都将运行。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
- 定义身份验证质询示例

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。

会话

`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

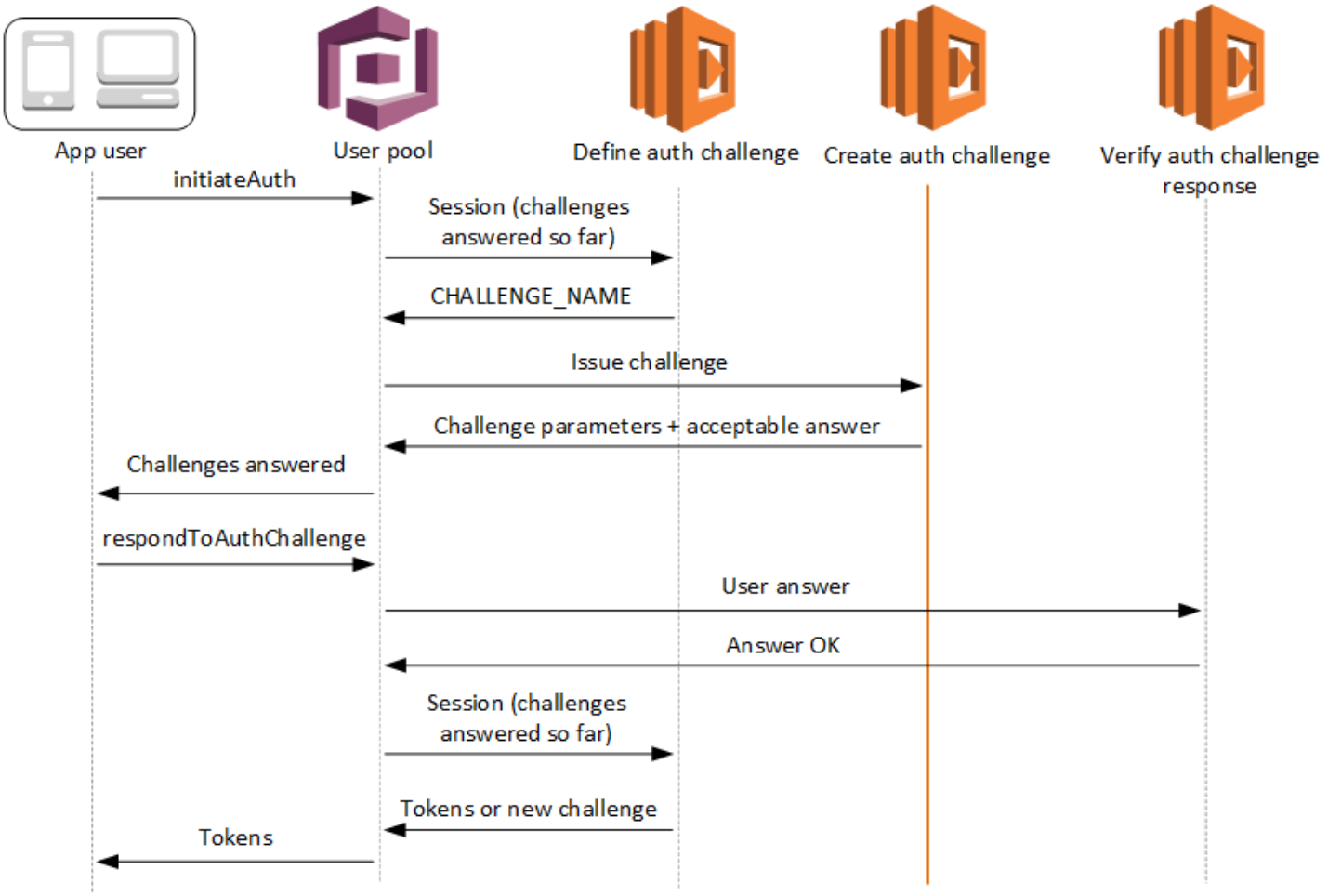
此示例针对身份验证定义一系列质询，并仅在用户成功完成所有质询后发布令牌。

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
- 创建身份验证质询示例

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"、"D 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）。

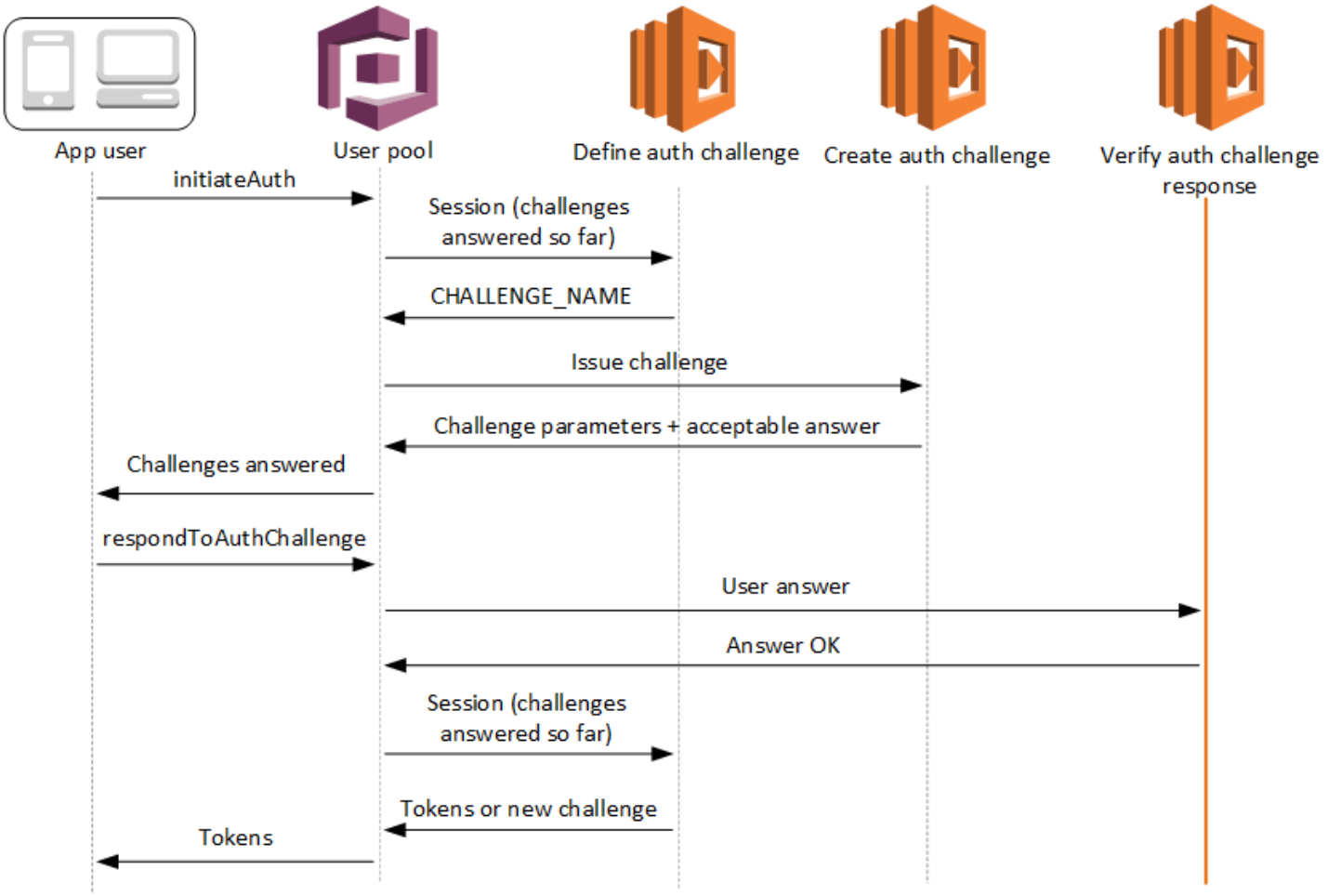
创建身份验证质询示例

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
- 验证身份验证质询响应示例

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 会在令牌生成之前调用此触发器，您可以自定义用户池令牌中的声明。使用版本 1 或 `V1_0` 令牌生成前触发器事件的基本功能，可以自定义身份（ID）令牌。在启用了高级安全功能的用户池中，您可以通过自定义访问令牌来生成版本 2 或 `V2_0` 触发器事件。

Amazon Cognito 将向您的函数发送 `V1_0` 请求，其中包含将写入 ID 令牌的数据。`V2_0` 事件是包含将由 Amazon Cognito 写入身份和访问令牌的数据的单个请求。要自定义这两个令牌，您必须更新函数以使用最新的触发器版本，并在同一个响应中发送两个令牌的数据。

在 Amazon Cognito 向您的应用程序发布身份和访问令牌之前，此 Lambda 触发器可以添加、删除和修改这些令牌中的某些声明。要使用此功能，可以从 Amazon Cognito 用户池控制台关联 Lambda 函数或通过 AWS Command Line Interface（AWS CLI）更新用户池 `LambdaConfig`。

事件版本

您的用户池可以向您的 Lambda 函数提供不同版本的令牌生成前触发事件。`V1_0`触发器提供用于修改 ID 令牌的参数。`V2_0`触发器为以下内容提供参数。

1. `V1_0`触发器的功能。
2. 能够自定义访问令牌。
3. 能够将复杂的数据类型传递给 ID 并访问令牌声明值：
   - String
   - 数字
   - 布尔值
   - 字符串、数字、布尔值的数组或其中任何一个的组合
   - JSON

> **Note**
>
> 在 ID 令牌中，您可以将复杂对象填充到除了phone_number_verified、email_verifiedupdated_at、和address之外的声明值。

默认情况下，用户池传递V1_0事件。要将您的用户池配置为发送V2_0事件，请在 Amazon Cognito 控制台中配置触发器时，选择基本功能+访问令牌自定义的触发事件版本。您也可以在UpdateUserPool 或 CreateUserPool API 请求的LambdaVersionLambdaConfig参数中设置的值。使用V2_0事件自定义访问令牌需要支付额外费用。有关更多信息，请参阅 Amazon Cognito 定价。

排除的声明和范围

Amazon Cognito 限制了您可以在访问令牌和身份令牌中添加、修改或隐藏的声明和范围。如果您的 Lambda 函数尝试为这些声明中的任何一个设置值，Amazon Cognito 会颁发一个具有原始声明值的令牌（如果请求中存在原始声明值）。

共享声明

- acr
- amr
- at_hash
- auth_time
- azp
- exp
- iat
- iss
- jti
- nbf
- nonce
- origin_jti
- sub
- token_use

ID 令牌声明

- identities
- aud
- cognito:username

访问令牌声明

- username
- client_id
- scope

> **Note**
>
> 您可以使用 scopesToAdd 和 scopesToSuppress 响应值更改访问令牌中的范围，但不能直接修改 scope 声明。您不能添加以 aws.cognito 开头的范围，包括用户池的保留范围 aws.cognito.signin.user.admin。

- device_key
- event_id
- version

您不能添加或覆盖带有以下前缀的声明，但可以将其隐藏，或者阻止它们出现在令牌中。

- dev:
- cognito:

您可以向访问令牌添加 aud 声明，但其值必须与当前会话的应用程序客户端 ID 相匹配。您可以从 event.callerContext.clientId 获取请求事件中的客户端 ID。

自定义身份令牌

使用令牌生成前 Lambda 触发器，您可以自定义来自用户池的身份（ID）令牌的内容。ID 令牌提供来自可信身份源的用户属性，用于登录 Web 或移动应用程序。有关 ID 令牌的更多信息，请参阅使用 ID 令牌。

将令牌生成前 Lambda 触发器与 ID 令牌结合使用，可实现以下目的。

- 在运行时更改您的用户从身份池中请求的 IAM 角色。
- 从外部来源添加用户属性。
- 添加或替换现有用户属性值。
- 隐藏用户属性，否则这些属性由于用户获得的授权范围以及您授予应用程序客户端的属性读取权限，会传递给您的应用程序。

自定义访问令牌

使用令牌生成前 Lambda 触发器，您可以自定义来自用户池的访问令牌的内容。访问令牌授权用户从受访问保护的资源（例如 Amazon Cognito 令牌授权的 API 操作和第三方 API）中检索信息。尽管您可以使用客户端凭证授予的 Amazon Cognito 生成访问令牌 machine-to-machine（M2M）授权，但是 M2M 请求不会调用令牌生成前的触发函数，也无法发布自定义访问令牌。有关访问令牌的更多信息，请参阅使用访问令牌。

将令牌生成前 Lambda 触发器与访问令牌结合使用，可实现以下目的。

- 在 `scope` 声明中添加或隐藏 OAuth 2.0 范围。例如，您可以将范围添加到由 Amazon Cognito 用户池 API 身份验证生成的访问令牌中，该身份验证仅分配范围 `aws.cognito.signin.user.admin`。
- 更改用户在用户池组中的成员资格。
- 添加尚不存在于 Amazon Cognito 访问令牌中的声明。
- 隐藏原本会传递到应用程序的声明。

要在用户池中支持访问自定义，必须将用户池配置为生成触发器请求的更新版本。请按照如下所示的过程更新用户池。

AWS Management Console

在令牌生成前 Lambda 触发器中支持访问令牌自定义

1. 转到 Amazon Cognito 控制台，然后选择用户池。
2. 从列表中选择一个现有用户池，或创建一个用户池。
3. 如果尚未激活高级安全功能，请从应用程序集成选项卡中激活该功能。
4. 选择用户池属性选项卡，并找到 Lambda 触发器。
5. 添加或编辑令牌生成前触发器。

6. 在分配 Lambda 函数下选择一个 Lambda 函数。
7. 选择基本功能 + 访问令牌自定义的触发事件版本。此设置会更新 Amazon Cognito 发送给您的函数的请求参数，使该函数包含用于自定义访问令牌的字段。

User pools API

在令牌生成前 Lambda 触发器中支持访问令牌自定义

生成CreateUserPool或 UpdateUserPoolAPI 请求。必须为所有您不想设置为默认值的参数指定一个值。有关更多信息，请参阅 更新用户群体配置。

在请求的 `LambdaVersion` 参数中包含以下内容。`LambdaVersion` 的值为 `V2_0` 会使您的用户池添加用于自定义访问令牌的参数。要调用特定的函数版本，请使用以函数版本作为 `LambdaArn` 值的 Lambda 函数 ARN。

```
"PreTokenGenerationConfig": {
   "LambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction",
   "LambdaVersion": "V2_0"
},
```

主题

- 令牌生成前 Lambda 触发器源
- 令牌生成前 Lambda 触发器参数
- 令牌生成前触发器事件版本 2 示例：添加和隐藏声明、范围及组
- 代币生成前事件版本二示例：添加包含复杂对象的声明
- 令牌生成前事件版本 1 示例：添加新声明并隐藏现有声明
- 令牌生成前事件版本 1 示例：修改用户的组成员资格

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。

triggerSource 值	事件
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。在向用户池添加令牌生成前 Lambda 触发器时，您可以选择触发器版本。此版本决定 Amazon Cognito 是否将请求以及用于自定义访问令牌的附加参数传递给您的 Lambda 函数。

Version 1

版本 1 令牌可以在 ID 令牌中设置群组成员资格、IAM 角色和新的声明。

```
{
    "request": {
        "userAttributes": {"string": "string"},
        "groupConfiguration": {
                "groupsToOverride": [
                    "string",
                    "string"
                ],
                "iamRolesToOverride": [
                    "string",
                    "string"
                ],
                "preferredRole": "string"
        },
        "clientMetadata": {"string": "string"}
    },
    "response": {
        "claimsOverrideDetails": {
            "claimsToAddOrOverride": {"string": "string"},
```

```
            "claimsToSuppress": [
                "string",
                "string"
            ],
            "groupOverrideDetails": {
                "groupsToOverride": [
                    "string",
                    "string"
                ],
                "iamRolesToOverride": [
                    "string",
                    "string"
                ],
                "preferredRole": "string"
            }
        }
    }
}
```

Version 2

版本 2 请求事件添加了自定义访问令牌的字段。它还增加了对响应对象中复杂claimsToOverride数据类型的支持。您的 Lambda 函数可以返回以下类型的数据，其值为：claimsToOverride

- String
- 数字
- 布尔值
- 字符串、数字、布尔值的数组或其中任何一个的组合
- JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string"
        },
        "scopes": ["string", "string"],
        "groupConfiguration": {
            "groupsToOverride": ["string", "string"],
            "iamRolesToOverride": ["string", "string"],
```

```
            "preferredRole": "string"
        },
        "clientMetadata": {
            "string": "string"
        }
    },
    "response": {
        "claimsAndScopeOverrideDetails": {
            "idTokenGeneration": {
                "claimsToAddOrOverride": {
                    "string": [accepted datatype]
                },
                "claimsToSuppress": ["string", "string"]
            },
            "accessTokenGeneration": {
                "claimsToAddOrOverride": {
                    "string": [accepted datatype]
                },
                "claimsToSuppress": ["string", "string"],
                "scopesToAdd": ["string", "string"],
                "scopesToSuppress": ["string", "string"]
            },
            "groupOverrideDetails": {
                "groupsToOverride": ["string", "string"],
                "iamRolesToOverride": ["string", "string"],
                "preferredRole": "string"
            }
        }
    }
}
```

令牌生成前请求参数

名称	描述	最低触发器事件版本
userAttributes	用户池中用户配置文件的属性。	1
groupConfiguration	包含当前组配置的输入对象。对象包括 `groupsToOverride` 、`iamRolesToOverride` 和 `preferredRole` 。	1

名称	描述	最低触发器事件版本
groupsToOverride	您的用户所属的用户池组。	1
iamRolesTo覆盖	您可以将用户池组与 AWS Identity and Access Management (IAM) 角色关联。此元素是您的用户所属组中的所有 IAM 角色的列表。	1
preferredRole	您可以为用户池组设置一个优先级。此元素包含 `groupsToOverride` 元素中具有最高优先级组中的 IAM 角色的名称。	1
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。 要将此数据传递给您的 Lambda 函数，请使用AdminRespondToAuthChallenge和 RespondToAuthChallengeAPI 操作中的 ClientMetadata参数。Amazon Cognito 在传递给令牌生成前函数的请求中不包含来自`ClientMetadata` 参数AdminInitiateAuth和 InitiateAuthAPI 操作的数据。	1
范围	您用户的 OAuth 2.0 范围。访问令牌中的范围是您的用户请求且您授权应用程序客户端发布的用户池标准范围和自定义范围。	2

令牌生成前响应参数

名称	描述	最低触发器事件版本
claimsOverrideDetails	用于存放 `V1_0` 触发器事件中所有元素的容器。	1
claimsAndScopeOverrideDetails	用于存放 `V2_0` 触发器事件中所有元素的容器。	2

名称	描述	最低触发器事件版本
idTokenGeneration	您要在用户的 ID 令牌中覆盖、添加或隐藏的声明。这是 ID 令牌自定义值的父元素，仅出现在版本 2 事件中，但子元素出现在版本 1 事件中。	2
accessTokenGeneration	您要在用户的访问令牌中覆盖、添加或隐藏的声明和范围。这是访问令牌自定义值的父元素，仅出现在版本 2 事件中。	2
claimsToAddOrOverride	您要添加或修改的一个或多个声明及其值的映射。对于与组相关的声明，请改用 `groupOverrideDetails` 。 在版本 2 事件中，此元素同时出现在 `accessTokenGeneration` 和 `idTokenGeneration` 下。	1*
claimsToSuppress	您希望 Amazon Cognito 隐藏的声明列表。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。 在版本 2 事件中，此元素同时出现在 `accessTokenGeneration` 和 `idTokenGeneration` 下。	1
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。	1

名称	描述	最低触发器事件版本
scopesToAdd	您希望在用户的访问令牌中添加到 scope 声明的 OAuth 2.0 范围列表。不能添加包含一个或多个空格字符的范围值。	2
scopesToSuppress	您希望从用户的访问令牌中的 scope 声明移除的 OAuth 2.0 范围列表。	2

* 版本 1 事件的响应对象可以返回字符串。版本 2 事件的响应对象可以返回复杂的对象。

令牌生成前触发器事件版本 2 示例：添加和隐藏声明、范围及组

此示例对用户的令牌进行了以下修改。

1. 在 ID 令牌中将其 family_name 设置为 Doe。
2. 防止 email 和 phone_number 声明出现在 ID 令牌中。
3. 将其 ID 令牌 cognito:roles 声明设置为 "arn:aws:iam::123456789012:role\/sns_callerA","arn:aws:iam::123456789012:role\/sns_callerC","arn:aws:iam::123456789012:role\/sns_callerB"。
4. 将其 ID 令牌 cognito:preferred_role 声明设置为 arn:aws:iam::123456789012:role/sns_caller。
5. 将作用域 openid、email 和 solar-system-data/asteroids.add 添加到访问令牌中。
6. 隐藏访问令牌的作用域 phone_number 和 aws.cognito.signin.user.admin。删除 phone_number 可阻止从 userInfo 中检索用户的电话号码。删除 aws.cognito.signin.user.admin 可阻止用户通过 Amazon Cognito 用户池 API 请求读取和修改自己的个人资料。

   > **Note**
   >
   > 只有当访问令牌中的剩余作用域包括 openid 和至少一个其他标准作用域时，从作用域中删除 phone_number 才会阻止检索用户的电话号码。有关更多信息，请参阅 关于范围。

7. 将其 ID 和访问令牌 cognito:groups 声明设置为 "new-group-A","new-group-B","new-group-C"。

JavaScript

```
export const handler = function(event, context) {
  event.response = {
    "claimsAndScopeOverrideDetails": {
      "idTokenGeneration": {
        "claimsToAddOrOverride": {
          "family_name": "Doe"
    },
        "claimsToSuppress": [
          "email",
          "phone_number"
    ]
      },
      "accessTokenGeneration": {
        "scopesToAdd": [
          "openid",
          "email",
          "solar-system-data/asteroids.add"
        ],
        "scopesToSuppress": [
          "phone_number",
          "aws.cognito.signin.user.admin"
        ]
      },
      "groupOverrideDetails": {
        "groupsToOverride": [
    "new-group-A",
    "new-group-B",
    "new-group-C"
  ],
        "iamRolesToOverride": [
          "arn:aws:iam::123456789012:role/new_roleA",
          "arn:aws:iam::123456789012:role/new_roleB",
          "arn:aws:iam::123456789012:role/new_roleC"
        ],
        "preferredRole": "arn:aws:iam::123456789012:role/new_role",
      }
    }
  };
  // Return to Amazon Cognito
  context.done(null, event);
};
```

Amazon Cognito 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": "2",
    "triggerSource": "TokenGeneration_Authentication",
    "region": "us-east-1",
    "userPoolId": "us-east-1_EXAMPLE",
    "userName": "JaneDoe",
    "callerContext": {
        "awsSdkVersion": "aws-sdk-unknown-unknown",
        "clientId": "1example23456789"
    },
    "request": {
        "userAttributes": {
            "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "cognito:user_status": "CONFIRMED",
            "email_verified": "true",
            "phone_number_verified": "true",
            "phone_number": "+12065551212",
            "family_name": "Zoe",
            "email": "Jane.Doe@example.com"
        },
        "groupConfiguration": {
            "groupsToOverride": ["group-1", "group-2", "group-3"],
            "iamRolesToOverride": ["arn:aws:iam::123456789012:role/sns_caller1",
 "arn:aws:iam::123456789012:role/sns_caller2", "arn:aws:iam::123456789012:role/
sns_caller3"],
            "preferredRole": ["arn:aws:iam::123456789012:role/sns_caller"]
        },
        "scopes": [
            "aws.cognito.signin.user.admin", "openid", "email", "phone"
        ]
    },
    "response": {
        "claimsAndScopeOverrideDetails": []
    }
}
```

代币生成前事件版本二示例：添加包含复杂对象的声明

此示例对用户的令牌进行了以下修改。

1. 将数字、字符串、布尔值和 JSON 类型的声明添加到 ID 令牌中。这是第二版触发器事件对 ID 令牌提供的唯一更改。
2. 将数字、字符串、布尔值和 JSON 类型的声明添加到访问令牌中。
3. 向访问令牌添加三个范围。
4. 隐藏 ID `email` 和访问令牌中的和`sub`声明。
5. 抑制访问令牌中的`aws.cognito.signin.user.admin`作用域。

JavaScript

```
export const handler = function(event, context) {

    var scopes = ["MyAPI.read", "MyAPI.write", "MyAPI.admin"]
    var claims = {}
    claims["aud"]= event.callerContext.clientId;
    claims["booleanTest"] = false;
    claims["longTest"] = 9223372036854775807;
    claims["exponentTest"] = 1.7976931348623157E308;
    claims["ArrayTest"] = ["test", 9223372036854775807, 1.7976931348623157E308,
 true];
    claims["longStringTest"] = "\{\
        \"first_json_block\": \{\
            \"key_A\": \"value_A\",\
            \"key_B\": \"value_B\"\
        \},\
        \"second_json_block\": \{\
            \"key_C\": \{\
                \"subkey_D\": [\
                    \"value_D\",\
                    \"value_E\"\
                ],\
                \"subkey_F\": \"value_F\"\
            \},\
            \"key_G\": \"value_G\"\
        \}\
    \}";
    claims["jsonTest"] = {
     "first_json_block": {
```

```
      "key_A": "value_A",
      "key_B": "value_B"
     },
     "second_json_block": {
      "key_C": {
       "subkey_D": [
        "value_D",
        "value_E"
       ],
       "subkey_F": "value_F"
      },
      "key_G": "value_G"
     }
    };
    event.response = {
        "claimsAndScopeOverrideDetails": {
            "idTokenGeneration": {
                "claimsToAddOrOverride": claims,
                "claimsToSuppress": ["email","sub"]
            },
            "accessTokenGeneration": {
                "claimsToAddOrOverride": claims,
                "claimsToSuppress": ["email","sub"],
                "scopesToAdd": scopes,
                "scopesToSuppress": ["aws.cognito.signin.user.admin"]
            }
        }
    };
    console.info("EVENT response\n" + JSON.stringify(event, (_, v) => typeof v ===
 'bigint' ? v.toString() : v, 2))
    console.info("EVENT response size\n" + JSON.stringify(event, (_, v) => typeof v
 === 'bigint' ? v.toString() : v).length)
    // Return to Amazon Cognito
    context.done(null, event);
};
```

Amazon Cognito 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": "2",
    "triggerSource": "TokenGeneration_HostedAuth",
    "region": "us-west-2",
    "userPoolId": "us-west-2_EXAMPLE",
    "userName": "JaneDoe",
    "callerContext": {
        "awsSdkVersion": "aws-sdk-unknown-unknown",
        "clientId": "1example23456789"
    },
    "request": {
        "userAttributes": {
            "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "cognito:user_status": "CONFIRMED"
            "email_verified": "true",
            "phone_number_verified": "true",
            "phone_number": "+12065551212",
            "email": "Jane.Doe@example.com"
        },
        "groupConfiguration": {
            "groupsToOverride": ["group-1", "group-2", "group-3"],
            "iamRolesToOverride": ["arn:aws:iam::123456789012:role/sns_caller1"],
            "preferredRole": ["arn:aws:iam::123456789012:role/sns_caller1"]
        },
        "scopes": [
            "aws.cognito.signin.user.admin",
            "phone",
            "openid",
            "profile",
            "email"
        ]
    },
    "response": {
        "claimsAndScopeOverrideDetails": []
    }
}
```

令牌生成前事件版本 1 示例：添加新声明并隐藏现有声明

此示例将版本 1 触发器事件与令牌生成前 Lambda 函数结合使用，以添加新声明并隐藏现有声明。

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。

JSON

```
{
  "request": {},
  "response": {}
}
```

令牌生成前事件版本 1 示例：修改用户的组成员资格

此示例将版本 1 触发器事件与令牌生成前 Lambda 函数结合使用，以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。

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

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

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。

triggerSource 值	事件
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 InitiateAuthAPI 操作中的 ClientMetadata 参数中AdminInitiateAuth传递的数据。要将此数据传递给您的 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。有关 Amazon Cognito 适用于这些参数的字符串限制，请参阅。MessageTemplateType

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。只有当用户池的 EmailSendingAccount属性为时，您才能使用该emailSubject参数DEVELOPER。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

Amazon Cognito 向此示例自定义消息 Lambda 函数发送的请求的`triggerSource`值为，用户名`CustomMessage_AdminCreateUser`和临时密码均为。该函数`${event.request.codeParameter}`从请求中的临时密码和`${event.request.usernameParameter}`请求中的用户名填充。

您的自定义消息必须在响应对象`codeParameter`和`emailMessage`中`usernameParameter`插入`smsMessage`和的值。在此示例中，该函数将相同的消息写入响应字段`event.response.smsMessage`和`event.response.emailMessage`。

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

Amazon Cognito 调用此触发器向用户发送电子邮件通知。

CustomSMSSender

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

当您为用户群体分配自定义电子邮件发件人触发器时，如果用户事件要求 Amazon Cognito 发送电子邮件，则它会调用 Lambda 函数，而不是其原定设置行为。使用自定义发件人触发器，您的 AWS Lambda 函数可以通过您选择的方法和提供商向您的用户发送电子邮件通知。您的函数的自定义代码必须处理和传递用户群体中的所有电子邮件。

> **Note**
>
> 目前，您无法在 Amazon Cognito 控制台中分配自定义发件人触发器。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）中创建对称加密密钥 Amazon Cognito 生成密钥（临时密码、验证码和确认码），然后使用此 KMS 密钥对这些密钥进行加密。然后，您可以在 Lambda 函数中使用解密 API 操作来解密这些密钥，并以明文形式将其发送给用户。对于函数中的 AWS KMS 操作，AWS Encryption SDK 是一个有用的工具。
2. 创建一个您要分配为自定义发件人触发器的 Lambda 函数。将您的 KMS 密钥的 `kms:Decrypt` 权限授予 Lambda 函数角色。
3. 授予 Amazon Cognito 服务主体 `cognito-idp.amazonaws.com` 访问权限，以调用 Lambda 函数。
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。要传递用户的验证码或确认码，请对请求中 `code` 参数的值进行 Base64 解码和解密。此操作将生成必须包含在消息中的明文代码或密码。
5. 更新用户池，使其使用自定义发件人 Lambda 触发器。使用自定义发件人触发器更新或创建用户群体的 IAM 主体必须具有为您的 KMS 密钥创建授予的权限。以下 `LambdaConfig` 代码段分配自定义短信和电子邮件发件人函数。

```
"LambdaConfig": {
   "KMSKeyID": "arn:aws:kms:us-
east-1:123456789012:key/a6c4f8e2-0c45-47db-925f-87854bc9e357",
   "CustomEmailSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   },
   "CustomSMSSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   }
```

自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。

```
   #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。

TriggerSource value	事件
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。
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

当您为用户群体分配自定义短信发件人触发器时，如果用户事件要求 Amazon Cognito 发送短信，则它会调用 Lambda 函数，而不是其原定设置行为。使用自定义发件人触发器，您的 AWS Lambda 函数可以通过您选择的方法和提供商向用户发送 SMS 通知。您的函数的自定义代码必须处理和传递用户群体中的所有短信。

> **Note**
>
> 目前，您无法在 Amazon Cognito 控制台中分配自定义发件人触发器。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) 中创建对称加密密钥。Amazon Cognito 生成密钥（临时密码、验证码和确认码），然后使用此 KMS 密钥对这些密钥进行加密。然后，您可以在 Lambda 函数中使用解密 API 操作来解密这些密钥，并以明文形式将其发送给用户。AWS Encryption SDK是对函数进行 AWS KMS 操作的有用工具。
2. 创建一个您要分配为自定义发件人触发器的 Lambda 函数。将您的 KMS 密钥的 `kms:Decrypt` 权限授予 Lambda 函数角色。
3. 授予 Amazon Cognito 服务主体 `cognito-idp.amazonaws.com` 访问权限，以调用 Lambda 函数。

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。要传递用户的验证码或确认码，请对请求中 code 参数的值进行 Base64 解码和解密。此操作将生成必须包含在消息中的明文代码或密码。
5. 更新用户池，使其使用自定义发件人 Lambda 触发器。使用自定义发件人触发器更新或创建用户群体的 IAM 主体必须具有为您的 KMS 密钥创建授予的权限。以下 LambdaConfig 代码段分配自定义短信和电子邮件发件人函数。

```
"LambdaConfig": {
   "KMSKeyID": "arn:aws:kms:us-
east-1:123456789012:key/a6c4f8e2-0c45-47db-925f-87854bc9e357",
   "CustomEmailSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   },
   "CustomSMSSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   }
```

自定义 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。要将此数据传递给您的 Lambda 函数，您可以使用AdminRespondToAuthChallenge和RespondToAuthChallengeAPI 操作中的 ClientMetadata 参数。Amazon Cognito 在传递给身份验证后函数的请求中不包含来自 ClientMetadata 参数AdminInitiateAuth和 InitiateAuthAPI 操作的数据。

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 调用您的 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。在亚马逊 Cognito 将机密发送 AWS Encryption SDK 给自定义发件人 Lambda 函数之前，Amazon Cognito 使用它来加密机密。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 添加到您的 Lambda 函数中。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。

```
   #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"
```

要删除带有的自定义发件人 Lambda 触发器 `update-user-pool` AWS CLI，请省略`CustomSMSSender`或`CustomEmailSender`参数`--lambda-config`，并包含您要在用户池中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
- 自定义 SMS 发件人 Lambda 触发器源

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 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。

要使用自定义短信发送器 Lambda 函数为 Amazon Cognito 短信创建测试环境，请参阅上的 aws-sample amazon-cognito-user-pools development-and-testing-with 库sms-redirected-to-email中的--。GitHub存储库包含可以创建新用户池或使用已有的用户池的 AWS CloudFormation 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，或者从用户池中移除自定义短信发送器功能分配。

> **Important**
>
> 不要使用 amazon-cognito-user-pool-development-and-testing-with- 中的模板sms-redirected-to-email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	用户请求新代码以确认其注册。
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。

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。

Amazon Pinpoint 未在以下区域推出。这些地区的 Amazon Cognito 用户群体不支持分析。

- Europe (Milan)
- 中东（巴林）
- 亚太地区（大阪）

- 以色列（特拉维夫）
- 非洲（开普敦）
- 亚太地区（雅加达）

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。

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？。

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。

> **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）。

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

配置用户创建策略

您的用户池可以允许用户注册，您也可以以管理员身份创建用户。您还可以控制注册后的验证和确认过程由用户掌控的程度。例如，您可能需要审查注册并根据外部验证流程接受注册。此配置，或管理员创建用户策略，还可设置用户无法再确认其用户账户之前的时间长度。

作为软件的客户身份和访问管理（CIAM）平台，Amazon Cognito 可以满足公众客户的需求。接受注册并拥有应用程序客户端的用户池，无论有没有托管 UI，都可以为互联网上知道您的可公开发现的应用程序客户端 ID 并请求注册的任何人创建用户配置文件。注册的用户配置文件可接收访问和身份令

牌，还可以访问您为应用程序授权的资源。在用户池中激活注册之前，请查看您的选项并确保配置符合您的安全标准。请按照以下步骤所述，谨慎设置启用自行注册和 `AllowAdminCreateUserOnly`。

AWS Management Console

用户池的注册体验选项卡和创建用户池向导的配置注册体验步骤包含在用户池中注册和以管理员身份创建用户的一些设置。

配置注册体验

1. 在 Cognito 辅助验证和确认中，选择是否允许 Cognito 自动发送消息以进行验证和确认。启用此设置后，Amazon Cognito 会向新用户发送电子邮件或短信，其中包含他们必须向您的用户池出示的代码。这将确认他们对电子邮件地址或电话号码的所有权，将等效属性设置为已验证，并确认用于登录的用户账户。您选择的要验证的属性将决定验证消息的传送方式和目的地。
2. 在创建用户时，验证属性更改并不重要，但与属性验证有关。可以允许已更改但尚未验证其登录属性的用户继续使用其新属性值或原始属性值登录。有关更多信息，请参阅 当用户更改其电子邮件或电话号码时应进行验证。
3. 必填属性显示用户注册或您创建用户之前必须为其提供值的属性。只能在创建用户池向导中设置必填属性。
4. 自定义属性对用户创建和注册过程很重要，因为只有在首次创建用户时才能为不可变的自定义属性设置值。有关自定义属性的更多信息，请参阅自定义属性。
5. 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，请在自助注册中，选择启用自行注册。如果禁用了自行注册，则只能在 Amazon Cognito 控制台中或通过 AdminCreateUser API 请求，以管理员身份创建新用户。在未启用自行注册的用户池中，SignUp API 请求会返回 `NotAuthorizedException`，托管 UI 也不会显示注册链接。

对于您计划以管理员身份创建用户的用户池，可以在管理员设置的临时密码到期时间下的登录体验选项卡中配置临时密码的有效期。

以管理员身份创建用户的另一个重要元素是邀请消息。创建新用户时，Amazon Cognito 会给他们发送一条包含您的应用程序链接的消息，以便他们可以首次登录。可以在消息模板下的消息收发选项卡中自定义这个消息模板。

您可以使用客户端密钥配置机密应用程序客户端（通常是 Web 应用程序），以防止在没有应用程序客户端密钥的情况下注册。作为安全最佳实践，请勿在公共应用程序客户端（通常是移动应用程序）中分发应用程序客户端密钥。您可以在 Amazon Cognito 控制台的应用程序集成选项卡中使用客户端密钥创建应用程序客户端。

Amazon Cognito user pools API

您可以在 CreateUserPool 或 UpdateUserPool API 请求中以编程方式设置用于在用户池中创建用户的参数。

AdminCreateUserConfig 元素将为用户池的以下属性设置值。

1. 启用自助注册
2. 向管理员创建的新用户发送的邀请消息

如果将以下示例添加到完整的 API 请求正文中，可设置一个用户池（自助注册为非活动状态）和一封基本的邀请电子邮件。

```
"AdminCreateUserConfig": {
      "AllowAdminCreateUserOnly": true,
      "InviteMessageTemplate": {
         "EmailMessage": "Your username is {username} and temporary password is
 {####}.",
         "EmailSubject": "Welcome to ExampleApp",
         "SMSMessage": "Your username is {username} and temporary password is
 {####}."
      }
   }
```

CreateUserPool 或 UpdateUserPool API 请求的以下其他参数用于控制新用户的创建。

AutoVerifiedAttributes

注册新用户时要自动向其发送消息的属性、电子邮件地址或电话号码。

策略

用户池密码策略。

架构

用户池自定义属性。它们对用户的创建和注册过程很重要，因为只有在首次创建用户时才能为不可变的自定义属性设置值。

此参数还将为您的用户池设置必填属性。如果将以下文本插入到完整 API 请求正文的 Schema 元素中，将把 email 属性设为必填属性。

```
{
            "Name": "email",
            "Required": true
}
```

注册并确认用户账户

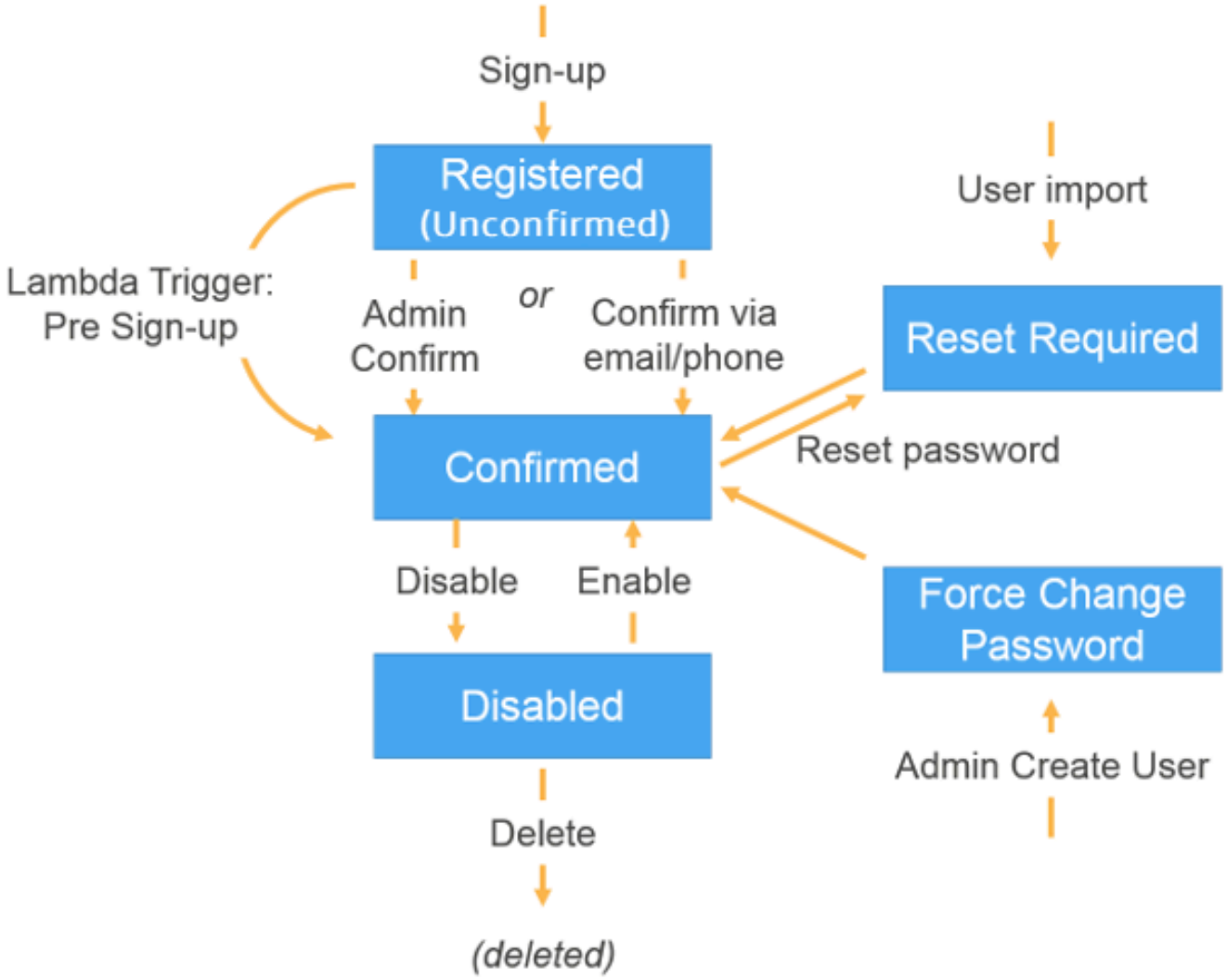
可通过以下任一方法将用户账户添加到您的用户池中：

- 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。
- 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。

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 小时过去并且您的用户的代码或链接不再有效，请调用 ResendConfirmationCodeAPI 操作生成并发送新的代码或链接。

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、带的admin-get-user AWS CLI命令或其中一个 AWS 软件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。完成中的以下步骤， AWS Management Console 将您的用户池配置为要求您的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池，请参阅用户池入门。

配置用户池

1. 导航到 Amazon Cognito 控制台。如果出现提示，请输入您的 AWS 凭据。
2. 从导航窗格中选择用户池。从列表中选择一个现有用户池，或创建一个用户池。
3. 选择登录体验选项卡，然后找到属性验证和用户账户确认。选择编辑。
4. 在 Cognito 辅助验证和确认下，选择是否允许 Cognito 自动发送消息以进行验证和确认。启用此设置后，当用户注册或您创建用户配置文件时，Amazon Cognito 会向您选择的用户联系人属性发送消息。为验证属性并确认登录的用户配置文件，Amazon Cognito 会通过消息向用户发送代码或链接。用户随后必须在您的 UI 中输入相应代码，这样，您的应用才能在 ConfirmSignUp 或 AdminConfirmSignUp API 请求中对其进行确认。

   > **Note**
   >
   > 您还可以禁用 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：

- 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。
- 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。

5. 选择您的要验证的属性：

   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。

6. 选择保存更改。

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池管理员身份进行确认

您可能不希望您的用户池自动在用户池中发送验证消息，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 控制台中或通过 IAM 身份验证的 API 操作确认新的用户账户。AdminConfirmSignUp无论您的用户池是否发送验证消息，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。要确认您以管理员身份创建的用户，请创建一个AdminSetUserPasswordAPI 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 凭据登录 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，请创建 AdminConfirmSignUpAPI 请求，或者admin-confirm-sign-up在 AWS CLI。
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 用户池 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。

应用程序将用户登录到具有密钥哈希值的客户端时，您可以使用任何用户池登录属性的值作为密钥哈希值的用户名元素。应用程序在使用 `REFRESH_TOKEN_AUTH` 的身份验证操作中请求新令牌时，用户名元素的值取决于您的登录属性。如果您的用户池没有将 `username` 用作登录属性，请通过用户的访问令牌或 ID 令牌中的 `sub` 声明设置密钥哈希用户名值。如果 `username` 为登录属性，请通过 `username` 声明设置密钥哈希用户名值。

以下 Amazon Cognito 用户池 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
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），使用用户池客户端的私有密钥、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，请参阅如何解决我的 Amazon Cognito 用户池 API 中的 “无法验证客户端的秘密哈希” 错误？<client-id> 在 AWS 知识中心中。

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，如果您将用户池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，您的应用在 VerifyUserAttributeAPI 请求中提交代码，以完成对新属性值的验证。

如果用户池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。

如果您的用户池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。有关如何配置用户池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。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 消息中的 以管理员身份创建用户账户和使用 Lambda 触发器自定义用户池工作流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（参见通过 CSV 文件将用户导入用户池中）创建的用户账户已处于已确认状态，因此用户无需输入确认码。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。

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

当用户在您的用户池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 限制了持续出现硬退邮件的 AWS 账户可以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。

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。

Amazon Cognito 使用 Amazon SNS 发送 SMS 消息。如果您 AWS 服务 之前没有发送过来自亚马逊 Cognito 或其他任何公司的短信，Amazon SNS 可能会将您的账户置于短信沙箱中。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。

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。

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，您可以使用注册前 Lambda 触发器或使用 API 操作自动确认用户AdminConfirmSignUp。有关更多信息，请参阅 注册并确认用户账户。

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 或 自定义电子邮件发件人 Lambda 触发器，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”。

- 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 尚未验证电子邮件地址，因此您的应用程序必须致电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，则应用程序必须致电GetUserAttributeVerificationCode以启动电子邮件验证流程。然后它必须通过致电提交验证码VerifyUserAttribute。

您可以调整 AWS 账户 和单条消息的短信支出配额。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 使用您创建用户池的地方或下表中传统的 Amazon SNS 备用区域的 Amazon SNS 资源发送短信。 AWS 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 选择 Amazon SNS 短信 AWS 区域。

Amazon Cognito 区域	旧版 Amazon SNS 备用区域
美国东部（俄亥俄）	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）
欧洲地区（法兰克福）	欧洲地区（爱尔兰）
欧洲地区（伦敦）	欧洲地区（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。

当用户更新属性并且您的用户群体验证新的属性值时

- 在确认代码以验证新值之前，用户可以使用原始属性值登录。
- 用户只有在确认代码以验证新值后，才能使用新属性值登录。
- 如果您在 AdminUpdateUserAttributesAPI 请求`true`中`phone_number_verified`将`email_verified`或设置为，则他们可以在确认 Amazon Cognito 发送给他们的代码之前登录。

当用户更新属性而您的用户群体未验证新的属性值时

- 用户无法使用原始属性值登录或接收消息。
- 在确认代码以验证新属性值之前，用户无法使用新属性值登录或接收除确认码之外的消息。
- 如果您在 AdminUpdateUserAttributesAPI 请求`true`中`phone_number_verified`将`email_verified`或设置为，则他们可以在确认 Amazon Cognito 发送给他们的代码之前登录。

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，请输入您的 AWS 凭据。

2. 在导航窗格中，选择 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. 选择保存更改。

要要求使用 Amazon Cognito API 进行属性更新验证，您可以在请求中设置`AttributesRequireVerificationBeforeUpdate`参数。UpdateUserPool

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，您可以创建一个 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

Amazon Cognito 允许您自定义短信和电子邮件验证消息以及用户邀请消息，以增强应用程序的安全性和用户体验。借助 Amazon Cognito，您可以在基于代码的验证或一键式链接验证之间进行选择，以满足您的应用程序需求。本主题讨论如何在 Amazon Cognito 控制台中对多因素身份验证 (MFA) 和验证通信进行个性化设置。

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
- 自定义 SMS 消息
- 自定义电子邮件验证消息
- 自定义用户邀请消息
- 自定义您的电子邮件地址
- 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）

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	令牌
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。您可以在通过AdminCreateUser操作生成的邀请电子邮件中使用{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 `{one-click-link-valid}` 和 `{one-click-link-invalid}` 占位符，您必须已经为用户群体配置了域。

高级安全模板占位符

描述	令牌
IP 地址	`{ip-address}`
城市	`{city}`
Country	`{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，您可以自定义短信

您可以在消息收发选项卡中的消息模板标题下自定义用于多重身份验证（MFA）的短信。

> **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。

Amazon Cognito 规定短信（包括验证码）的最大长度为 140 个 UTF-8 字符。

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。

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。

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。
- 设置用户属性值。
- 创建自定义属性。
- 在 `AdminCreateUser` API 请求中设置不可变自定义属性的值。此功能在 Amazon Cognito 控制台中不可用。
- 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。
- 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。

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. 选择登录体验选项卡，然后查找密码策略。选择编辑。
4. 选择自定义的密码策略模式。
5. 选择密码最小长度。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. 选择保存更改。

允许自助注册

1. 导航到 Amazon Cognito 控制台，选择用户池。
2. 从列表中选择一个现有用户池，或创建一个用户池。
3. 选择注册体验选项卡，然后查找自助注册。选择编辑。
4. 选择是否启用自助注册。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 消息。

1. 导航到 Amazon Cognito 控制台，选择用户池。
2. 从列表中选择一个现有用户池，或创建一个用户池。
3. 选择消息收发选项卡，然后查找消息模板。选择验证消息、邀请消息或 MFA 消息，然后选择编辑。
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. 验证消息

      i. 选择用于电子邮件消息的验证类型。代码验证将发送用户必须输入的数字代码。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. 输入用于电子邮件消息的电子邮件主题。

      iii. 输入用于电子邮件消息的自定义电子邮件消息模板。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 消息模板。

      v. 选择保存更改。

   b. 邀请消息

      i. 输入用于电子邮件消息的电子邮件主题。

ii. 输入用于电子邮件消息的自定义电子邮件消息模板。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 消息模板。

iv. 选择保存更改。

c. MFA 消息

i. 输入用于 SMS 消息的自定义 SMS 消息模板。

ii. 选择保存更改。

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. 选择用户选项卡，然后选择创建用户。
4. 检查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. 选择用于新用户的用户名。
7. 选择您是要为用户创建密码，还是让 Amazon Cognito 生成密码。任何临时密码都必须遵守用户池密码策略。
8. 选择创建。

9. 选择用户选项卡，然后选择用户的用户名条目。添加和编辑用户属性和组成员资格。查看用户事件历史记录。

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。您可以为群组分配 AWS Identity and Access Management (IAM) 角色来定义群组成员的权限。

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、API 和 CLI 在用户池中创建和管理群组。 AWS Management Console作为开发者（使用 AWS 证书），您可以创建、读取、更新、删除和列出用户池的群组。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。
- 在通过身份池为用户请求 AWS 凭证时，您的应用程序还可以在参数中指定角色 ARN，从可用角色中GetCredentialsForIdentityCustomRoleARN进行选择。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：

- 您可以创建的群组数量受到 Amazon Cognito 服务配额的限制。
- 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。

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，请输入您的 AWS 凭据。
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。
   - 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。
   - 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。要检索所有属性，请不要包含 `AttributesToGet` 参数或文本字符串值为 `null` 的请求 `AttributesToGet`。
- `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": null,
    "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": null,
    "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
- 通过 CSV 文件将用户导入用户池中

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 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。

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。
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 函数。
4. 您的 迁移用户 Lambda 触发器 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 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 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。在您的函数完成其调用并将其响应返回给 Amazon Cognito 后，您的用户群体将通过电子邮件或 SMS 发送密码重置代码。在您的应用程序中，提示您的用户输入确认码和新密码，然后在 ConfirmForgotPassword API 请求中将该信息发送给 Amazon Cognito。您还可以在 Amazon Cognito 托管 UI 中使用内置的忘记密码流程页面。

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

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 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,pi
```

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。

- 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。

  - 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,pi
John,,John,Doe,,,,,,,johndoe@example.com,TRUE,,02/01/1985,,,+12345550100,TRUE,123 Any
 Street,,FALSE
```

```
Jane,,Jane,Roe,,,,,,,janeroe@example.com,TRUE,,01/01/1985,,,+12345550199,TRUE,100 Main
 Street,,FALSE
```

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）
- 导入用户（AWS CLI）

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 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 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
- 解析结果

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并输入任意密码时，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. 在您的应用程序中，通过 `InitiateAuth` 使用随机密码以静默方式为当前用户尝试登录。
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。

    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。

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。您也可以在中的用户池定义中添加自定义属性 AWS Management Console。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。

> **Note**
>
> 一些文档和标准将属性称为成员。

主题

标准属性

Amazon Cognito 根据 OpenID Connect 规范为所有用户分配一组标准属性。默认情况下，标准和自定义属性值可以是长度不超过 2048 个字符的任何字符串，但是某些属性值有格式限制。

标准属性是：

- `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，默认情况下，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。只能验证 email 和 phone_number 属性。

> **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。要创建用户而不给出必填属性的值，管理员可以使用 AdminCreateUserAPI。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。

标准属性详细信息和格式限制

birthdate

值必须是一个有效的 10 个字符的日期，格式为 YYYY-MM-DD。

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地址，也可以将其标记为已验证。使用 AdminUpdateUserAttributesAPI 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`。您也可以在的 “用户” 选项卡中编辑用户 ， AWS Management Console 将电子邮件地址标记为已验证。

值必须是有效的电子邮件地址字符串，遵循标准电子邮件格式，带有 @ 符号和域名，长度不超过 2048 个字符。

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。

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，也可以将其标记为已验证。使用 AdminUpdateUserAttributesAPI 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`。您也可以在的 “用户” 选项卡中编辑用户 ， AWS Management Console 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`是别名，则可以向 UpdateUserAttributesAPI 操作发出请求，并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 `phone_number` 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。

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. 前往 Amazon Cognito AWS Management Console如果控制台提示您，请输入您的 AWS 凭据。
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。

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。

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。有关属性的字符串限制的信息，请参阅 SignUpAPI 请求的用户名*username*属性。

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
用户有多个登录属性	是[1]	No[2]
用户必须先验证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	是	不支持
使用重复的电子邮件地址或电话号码注册用户并防止出现UsernameExistsException 错误[3]	是	不支持
可以将相同的电子邮件地址或电话号码属性值分配给多个用户	是[4]	否

[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] 当用户使用可能重复的电子邮件地址或电话号码注册但没有用户名时，您的用户群体不会生成 UsernameExistsException 错误。此行为独立于防止用户名存在错误，此错误适用于登录操作，但不适用于注册操作。

[4] 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CreateUserPool中的。

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，请输入您的 AWS 凭据。
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，请输入您的 AWS 凭据。
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。
- 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。请参阅属性权限和范围了解更多信息。

> **Note**
>
> 在您的代码和 使用基于角色的访问控制 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。

在创建用户池时，您还可以在的属性中添加开发者属性CreateUserPool。SchemaAttributes开发人员属性具有 dev: 前缀。您只能使用 AWS 凭证修改用户的开发者属性。开发人员属性是一项旧版特征，Amazon Cognito 已将其替换为应用程序客户端读写权限。

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. 前往 Amazon Cognito AWS Management Console如果控制台提示您，请输入您的 AWS 凭据。
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。默认情况下，当你创建新的应用程序客户端时，你会向你的应用程序授予所有标准和自定义属性的读写权限。要将应用程序限制为仅使用它所需的信息量，请在应用程序客户端配置中为属性分配特定权限。

最佳做法是在创建应用程序客户端时指定属性的读取和写入权限。向您的应用程序客户端授予访问应用程序运行所需的最少用户属性的访问权限。

> **Note**
>
> DescribeUserPoolClient只有在配置默认权限之外的应用程序客户端权限`WriteAttributes`时，才会返回`ReadAttributes`和的值。

更新属性权限（AWS Management Console）

1. 前往 Amazon Cognito AWS Management Console如果控制台提示您，请输入您的 AWS 凭据。
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。

对于每个应用程序，您可以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。您的应用程序可以检索您标记为可读的属性的值，也可以设置或修改您标记为可写的属性的值。如果您的应用程序尝试为它未获授权写入的属性设置值，Amazon Cognito 会返回 `NotAuthorizedException`。GetUser请求包括带有应用程序客户端声明的访问令牌；Amazon Cognito 仅返回应用程序客户端可以读取的属性的值。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。所有应用程序客户端都可以写入用户群体所需的属性。仅当您还为尚未具有值的必需属性提供值时，才能在 Amazon Cognito 用户群体 API 请求中设置相应属性的值。

自定义属性具有不同的读写权限特征。您可以将它们创建为用户群体的可变或不可变属性，并可以将它们设置为任何应用程序客户端的读或写属性。

在创建用户期间，不可变的自定义属性可以更新一次。您可以使用以下方法填充不可变的属性。

- `SignUp`：用户注册了一个应用程序客户端，该客户端具有对不可变自定义属性的写访问权限。它们为该属性提供了一个值。
- 使用第三方 IdP 登录：用户登录到一个应用程序客户端，该客户端具有对不可变自定义属性的写访问权限。IdP 的用户群体配置有一条规则，可以将提供的声明映射到不可变的属性。
- `AdminCreateUser`：您为不可变属性提供一个值。

有关您可以分配给应用程序客户端的范围的信息，请参阅使用资源服务器进行范围、M2M 和 API 授权。

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

强大、复杂的密码是用户群的最佳安全实践。特别是在对互联网开放的应用程序中，弱密码会将用户的凭据暴露给猜测密码并尝试访问您的数据的系统。密码越复杂，就越难猜出。Amazon Cognito 为注重安全的管理员提供了其他工具，例如高级安全功能和 AWS WAFWeb ACL，但是您的密码策略是用户目录安全的核心要素。

Amazon Cognito 用户池中本地用户的密码不会自动过期。最佳做法是在外部系统中记录用户密码重置的时间、日期和元数据。使用密码使用期限的外部日志，您的应用程序或 Lambda 触发器可以查找用户的密码年限，并在给定时间后要求重置。

您可以将用户池配置为要求符合安全标准的最低密码复杂性。复杂密码的最小长度至少为八个字符。它们还包括大写、数字和特殊字符的组合。

设置用户池密码策略

1. 创建一个用户池并导航到配置安全要求步骤，或访问现有用户池并导航到登录体验选项卡。
2. 导航到密码策略。
3. 选择密码策略模式。Cognito 默认使用推荐的最低设置来配置您的用户池。您也可以选择一项自定义密码策略。
4. 设置密码最小长度。所有用户都必须使用长度大于或等于这个值的密码进行注册或创建。您可以将这个最小值设置为 99，但用户可以设置最长 256 个字符的密码。
5. 在密码要求下配置密码的复杂性规则。选择您希望在每个用户的密码中至少包含一个的字符类型（数字、特殊字符、大写字母和小写字母）。

   密码中可以要求至少包含以下字符之一。在 Amazon Cognito 验证密码中包含所需的最少字符后，您的用户的密码可以包含任何类型的额外字符，但不得超过最大密码长度。

   - 大写和小写基本拉丁字母
   - 数字
   - 以下特殊字符。

     ```
     ^ $ * . [ ] { } ( ) ? " ! @ # % & / \ , > < ' : ; | _ ~ ` = + -
     ```

   - 非前导、非结尾的空格字符。
6. 为管理员设置的临时密码到期时间设置一个值。超过此期限，您通过 Amazon Cognito 控制台或 `AdminCreateUser` 创建的新用户将无法登录和设置新密码。使用临时密码登录后，他们的用户账户永远不会过期。要在 Amazon Cognito 用户池 API 中更新密码时长，请在您

的CreateUserPool或 UpdateUserPoolAPI 请求TemporaryPasswordValidityDays 中为设置一个值。

- 要重置已过期用户账户的访问权限，请执行以下操作之一。
  - 删除用户配置文件并创建新的用户配置文件。
  - 在 AdminSetUserPasswordAPI 请求中设置新的永久密码。
  - 在 AdminResetUserPasswordAPI 请求中生成新的确认码。

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：

- Amazon Cognito 服务中内置的@@ 默认电子邮件配置。
- 您的 Amazon Simple Email Service (Amazon SES) 配置。

创建用户群体后，您可以更改传递选项。

Amazon Cognito 会向您的用户发送电子邮件，其中包含用户可以输入的代码或用户可以选择的 URL 链接。下表显示了可以生成电子邮件的事件。

消息选项

活动	API 操作	传递选项	格式选项	可自定义	消息模板
忘记密码	ForgotPassword	电子邮件、短信	代码	否	不适用
邀请	AdminCreateUser	电子邮件、短信	代码	是	邀请消息
自行注册	SignUp	电子邮件、短信	代码，链接	是	验证消息

活动	API 操作	传递选项	格式选项	可自定义	消息模板
电子邮件地址或电话号码验证	UpdateUserAttributes	电子邮件、短信	代码	是	验证消息
多重身份验证 (MFA)	AdminInitiateAuth, InitiateAuth	短信	代码	是[1]	MFA 消息

[1] 用于短信。

Amazon SES 会对电子邮件收费。有关详情，请参阅 Amazon SES 定价。

默认电子邮件配置

Amazon Cognito 可以使用其默认电子邮件配置来为您处理电子邮件的传送。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。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。

使用默认配置时，您使用由管理的 Amazon SES 资源 AWS 来发送电子邮件。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配置时，您无法控制将其从禁止列表中删除。电子邮件地址可以无限期地保留在 AWS管理的禁止列表中。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

AWS 区域 在 Amazon SES 中配置电子邮件时，创建用户池的位置有三个要求之一。您可以从 Amazon SES 发送电子邮件，这些电子邮件可能与您的用户池位于同一区域、多个区域（包括同一区域），或者一个或多个远程区域。为了获得最佳性能，在您可以选择的情况下，使用与您的用户池位于同一区域的 Amazon SES 验证身份发送电子邮件。

Amazon SES 验证身份的地区要求类别

仅限区域内

您的用户池可以像用户池 AWS 区域 一样发送身份经过验证的电子邮件。在没有自定义FROM电子邮件地址的默认电子邮件配置中，Amazon Cognito 在同一地区使用no-reply@verificationemail.com经过验证的身份。

向后兼容

您的用户池可以在相同 AWS 区域 或以下备用区域之一发送身份经过验证的电子邮件：

- 美国东部（弗吉尼亚州北部）
- 美国西部（俄勒冈州）
- 欧洲地区（爱尔兰）

此功能支持用户池资源的连续性，这些资源可能是在服务启动时创建的，以满足 Amazon Cognito 的要求。该时期的用户池只能在有限的数量内发送身份经过验证的电子邮件 AWS 区域。在没有自定义FROM电子邮件地址的默认电子邮件配置中，Amazon Cognito 在同一地区使用no-reply@verificationemail.com经过验证的身份。

备用区域

您的用户池可以在用户池区域之外的备用 AWS 区域 地址发送身份经过验证的电子邮件。当 Amazon SES 在提供 Amazon Cognito 的地区不可用时，就会发生这种配置。

Amazon SES 针对您在备用地区经过验证的身份的发送授权政策必须信任来源地区的 Amazon Cognito 服务主体。有关更多信息，请参阅 授予使用默认电子邮件配置的权限。

在其中一些区域，Amazon Cognito 在两个备用区域之间拆分电子邮件，默认电子邮件配置为。`COGNITO_DEFAULT`在这些情况下，要使用自定义`FROM`电子邮件地址，Amazon SES 针对您在每个备用区域的已验证身份的发送授权策略必须信任原始地区的 Amazon Cognito 服务主体。有关更多信息，请参阅 授予使用默认电子邮件配置的权限。`DEVELOPER`在这些区域的 Amazon SES 电子邮件配置中，您必须在第一个列出的区域中使用经过验证的身份，并将其配置为信任用户池区域中的 Amazon Cognito 服务主体。例如，在中东（阿联酋）的用户池中，将欧洲（法兰克福）的已验证身份配置为可信`cognito-idp.me-central-1.amazonaws.com`。在没有自定义`FROM`电子邮件地址的默认电子邮件配置中，Amazon Cognito 在每个区域使用`no-reply@verificationemail.com`经过验证的身份。

> **Note**
>
> 在以下条件组合下，必须在 Region 元素中EmailConfiguration使用通配符指定`SourceArn`参数，格式为`arn:${Partition}:ses:*:${Account}:identity/${IdentityName}`。这允许您的用户池发送两者中具有相同已验证身份 AWS 账户 的电子邮件 AWS 区域。
>
> - 你的 EmailSendingAccount 是`COGNITO_DEFAULT`。
> - 你想使用自定义`FROM`地址。
> - 您的用户群在备用区域发送电子邮件。
> - 您的用户池在后面的 Amazon SES 支持区域表中指定了第二个[1]备用区域。

如果您使用软件开发工具包、Amazon Cognito API 或 CLI、或以编程方式创建用户池，则您的用户池将 AWS CDK使用参数 AWS CloudFormation为您的用户池指定的 Amazon SES 身份发送电子邮件。 AWS `SourceArn` EmailConfigurationAmazon SES 身份必须占据支持 AWS 区域的。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。

下表显示了您可以在 Amazon Cognito 上使用 Amazon SES 身份 AWS 区域 的地方。

用户池区域	区域选项	亚马逊 SES 支持的区域
美国东部（弗吉尼亚州北部）	向后兼容	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）	向后兼容	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（加利福尼亚北部）	仅限区域内	美国西部（加利福尼亚北部）
美国西部（俄勒冈）	向后兼容	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）	向后兼容	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)	向后兼容	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）	向后兼容	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）	向后兼容	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（海得拉巴）	备用区域	亚太地区（孟买）、亚太地区（新加坡）[1]
亚太地区（新加坡）	向后兼容	亚太地区（新加坡）、美国东部（弗吉尼亚北部）、美国

用户池区域	区域选项	亚马逊 SES 支持的区域
		西部（俄勒冈）、欧洲（爱尔兰）
亚太地区（悉尼）	向后兼容	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）
亚太地区（大阪）	仅限区域内	亚太地区（大阪）
亚太地区（雅加达）	仅限区域内	亚太地区（雅加达）
亚太地区（墨尔本）	备用区域	亚太地区（悉尼）、亚太地区（新加坡）[1]
欧洲地区（爱尔兰）	向后兼容	美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
欧洲地区（伦敦）	向后兼容	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）
欧洲地区（巴黎）	仅限区域内	欧洲地区（巴黎）
欧洲地区（法兰克福）	向后兼容	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（苏黎世）	备用区域	欧洲（法兰克福）、欧洲（伦敦）[1]
欧洲地区（斯德哥尔摩）	仅限区域内	欧洲地区（斯德哥尔摩）
欧洲地区（米兰）	仅限区域内	欧洲地区（米兰）
欧洲（西班牙）	备用区域	欧洲（巴黎）、欧洲（斯德哥尔摩）[1]

用户池区域	区域选项	亚马逊 SES 支持的区域
中东（巴林）	仅限区域内	中东（巴林）
中东（阿联酋）	备用区域	欧洲（法兰克福）、欧洲（伦敦）[1]
南美洲（圣保罗）	仅限区域内	南美洲（圣保罗）
以色列（特拉维夫）	仅限区域内	以色列（特拉维夫）
非洲（开普敦）	仅限区域内	非洲（开普敦）

[1] 用于具有默认电子邮件配置的用户池。Amazon Cognito 在每个区域使用相同电子邮件地址的经过验证的身份之间分发电子邮件。要使用自定义FROM地址，请EmailConfiguration使用格式为的SourceArn参数进行配置arn:*${Partition}*:ses:*:*${Account}*:identity/*${IdentityName}*。

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 电子邮件配置，请忽略此步骤。

当你第一次在任何地方使用 Amazon SES 时 AWS 区域，它会将你置 AWS 账户 于该地区的 Amazon SES 沙箱中。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。在您 AWS 账户 仍处于沙箱状态时，请不要将您的 Amazon SES 配置用于生产中的应用程序。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 AWS 账户 从沙箱中移除，请参阅《亚马逊简单电子邮件服务开发者指南》中的移出 Amazon SES 沙箱。

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。您授予的权限以及授予权限的过程取决于您使用的是默认电子邮件配置还是 Amazon SES 配置。

授予使用默认电子邮件配置的权限

只有将用户池配置为使用 Cognito 发送电子邮件或设置为`EmailSendingAccount`，才能完成此步骤。`COGNITO_DEFAULT`

使用默认的电子邮件配置，您的用户池可以使用以下任一地址发送电子邮件。

- 默认地址`no-reply@verificationemail.com`。
- 来自您在 Amazon SES 中经过验证的电子邮件地址或域名的自定义发件人地址。

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。这些权限由 Amazon SES 中地址或域名的发送授权策略授予。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。但是，如果您在控制台之外配置用户池，例如使用 AWS CLI 或 Amazon Cognito API，则必须使用 A mazon SES 控制台或 API 附加策略。PutIdentityPolicy

> **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。

Regions with Amazon SES

您在用户池区域或备用区域中的发送授权策略必须允许 Amazon Cognito 服务主体发送电子邮件。有关更多信息，请参阅区域表。如果您的用户池区域与 Amazon SES 区域中的至少一个值匹配，请在以下示例中使用全球服务主体配置您的发送授权策略。

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

Opt-in Regions without Amazon SES

在可用 Amazon Cognito AWS 区域 的地方，Amazon SES 并非在所有可选模式中都可用。中东（阿联酋）就是一个例子，它只能在欧洲（法兰克福）发送身份经过验证的电子邮件（eu-central-1）。在使用默认电子邮件配置的用户池中，Amazon Cognito 还会在两个区域分别发送身份经过验证的电子邮件。就中东（阿联酋）而言，额外的区域是欧洲（伦敦）。您必须更新两个地区的发送授权政策。

您在每个备用区域的发送授权政策必须允许用户池选择加入区域中的 Amazon Cognito 服务主体发送电子邮件。有关更多信息，请参阅区域表。如果您的地区被标记为备用区域，请使用区域服务主体配置您的发送授权策略，如下例所示。根据需要将示例区域标识符 *me-central* -1 替换为所需的区域 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "cognito-idp.me-central-1.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "aws:SourceArn": "<your user pool ARN>"
                }
```

```
            }
        }
    ]
}
```

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。

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。

Amazon Cognito 使用设置配置的用户会话的 AWS 凭证创建您的服务相关角色。此会话的 IAM 权限必须包括 `iam:CreateServiceLinkedRole` 操作。有关 IAM 中权限的更多信息，请参阅 I AM 用户指南中的AWS 资源访问管理。

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。

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

> **Note**
>
> 如果您的已验证身份是一个电子邮件地址，Amazon Cognito 会默认将该电子邮件地址设置为 FROM 和 REPLY-TO 电子邮件地址。但是，如果您的已验证身份是一个域，则必须为 FROM

和 REPLY-TO 电子邮件地址提供一个值。例如，如果您的已验证域是 example.com，则可以将 no-reply@example.com 同时设置为 FROM 和 REPLY-TO 电子邮件地址。

如果您想使用默认的 Amazon Cognito 电子邮件配置和地址，请忽略此步骤。

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，请输入您的 AWS 凭据。
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

- 自定义用户邀请消息

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。

Amazon SNS 对短信收取费用。有关更多信息，请参阅 Amazon SNS 定价。

> **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 触发器创建其他自定义身份验证要素。

Amazon Cognito 会向您的用户发送带有他们可以输入的验证码的短信。下表显示了可以生成短信的事件。

消息选项

活动	API 操作	传递选项	格式选项	可自定义	消息模板
忘记密码	ForgotPassword	电子邮件、短信	代码	否	不适用

活动	API 操作	传递选项	格式选项	可自定义	消息模板
邀请	AdminCreateUser	电子邮件、短信	代码	是	邀请消息
自行注册	SignUp	电子邮件、短信	代码，链接	是	验证消息
电子邮件地址或电话号码验证	UpdateUserAttributes	电子邮件、短信	代码	是	验证消息
多重身份验证 (MFA)	AdminInitiateAuth, InitiateAuth	短信、身份验证器应用程序	代码	是[1]	MFA 消息

[1] 用于短信。

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，通过自己的资源发送 SMS 消息。首次将 Amazon SNS 设置为在特定 AWS 区域地区发送短信时，Amazon SNS 会将 AWS 账户 您置于该地区的短信沙箱中。Amazon SNS 使用沙箱来防止欺诈和滥用行为并满足合规要求。当你进入沙箱 AWS 账户 时，Amazon SNS 会施加一些限制。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。在您 AWS 账户 仍处于沙箱状态时，请勿将您的 Amazon SNS 配置用于生产中的应用程序。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
2. 选择 Amazon SNS 短信 AWS 区域
3. 获取源身份以将 SMS 消息发送到美国电话号码
4. 确认您位于 SMS 沙盒中
5. 将您的账户移出 Amazon SNS 沙盒
6. 在 Amazon SNS 中验证 Amazon Cognito 的电话号码
7. 在 Amazon Cognito 中完成用户群体设置

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。使用 DescribeUserPoolAPI 请求在 AWS Management Console 或中查找用户池的 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。

选择 Amazon SNS 短信 AWS 区域

在某些情况下 AWS 区域，您可以选择包含要用于发送 Amazon Cognito SMS 消息的 Amazon SNS 资源的区域。在任何提供 Amazon Cognito AWS 区域 的地方，除了亚太地区（首尔）之外，您都可以在

创建用户池的地方使用 Amazon SNS 资源。 AWS 区域 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**
>
> 在中 AWS Management Console，只有切换到新的 Amazon Cognito 控制台体验后，您才能更改短信资源的区域。

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）。

对于某些人来说，在发布时 AWS 区域，Amazon Cognito使用其他地区的Amazon SNS资源发送了短信。要设置您的首选区域，请使用您的用户池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）的 Amazon Cognito 用户池 AWS 区域 必须使用您在亚太地区（东京）地区的 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)。在使用 Amazon Cognito 时 AWS 区域 ，你可能已经提高了支出限额。在更改 Amazon SNS 短信的配额之前，请在 AWS 支持中心提出增加配额的案例，以提高您在新区域的限额。 AWS 区域 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）
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）

Amazon Cognito 区域	Amazon SNS 区域
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（弗吉尼亚北部）
美国西部（加利福尼亚北部）	美国西部（加利福尼亚北部）
美国西部（俄勒冈）	美国西部（俄勒冈州）
亚太地区（孟买）	亚太地区（孟买）、亚太地区（新加坡）
亚太地区（海得拉巴）	亚太地区（海得拉巴）
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)
亚太地区（雅加达）	亚太地区（雅加达）
亚太地区（大阪）	亚太地区（大阪）
亚太地区（墨尔本）	亚太地区（墨尔本）
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）	欧洲地区（米兰）
欧洲（西班牙）	欧洲（西班牙）
中东（巴林）	中东（巴林）
南美洲（圣保罗）	南美洲（圣保罗）
以色列（特拉维夫）	以色列（特拉维夫）

Amazon Cognito 区域	Amazon SNS 区域
非洲（开普敦）	非洲（开普敦）
中东（阿联酋）	中东（阿联酋）
欧洲（苏黎世）	欧洲（苏黎世）

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。

如果您从事以下操作 AWS 区域，则必须开 AWS Support 票才能获得发件人身份。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（加利福尼亚北部）

当您的同一个来源身份有多个时 AWS 区域，Amazon SNS 会按以下优先顺序选择来源身份类型：短码、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。对每个有正式版 Amazon Cognito 用户池 AWS 区域 的地方重复此操作。

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。
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。控制台消息会显示您的沙箱状态 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。在中配置了包含您希望 Amazon Cognito 使用的 Amazon SNS 资源的原始身份后，您可以在留在 SMS 沙箱中时 AWS 账户验证美国电话号码。 AWS 区域 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

返回您在其中创建或者编辑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 端点。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。有关用户群体组的更多信息，请参阅 向用户池添加组。

Amazon Cognito 还有刷新令牌，您可以使用它来获取新的令牌或撤销现有令牌。刷新令牌来检索新的 ID 令牌和访问令牌。撤消令牌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，对用户群体用户和管理员不透明，并且只能由您的用户群体读取。

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 会创建会话并为经过身份验证的用户返回 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 Amazon API Gateway 等下游资源和 API 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，以及有关用于用户池的安全模型的信息。

自定义令牌

您可以自定义 Amazon Cognito 传递给应用程序的访问令牌和 ID 令牌。在 令牌生成前 Lambda 触发器中，您可以添加、修改和隐藏令牌声明。令牌生成前触发器是一个 Lambda 函数，Amazon Cognito 会向其发送一组默认声明。声明包括 OAuth 2.0 范围、用户群体组成员资格、用户属性等。然后，该函数可以根据这些信息在运行时进行更改，并将更新的令牌声明返回给 Amazon Cognito。

使用版本 2 事件自定义访问令牌需要支付额外费用。有关更多信息，请参阅 Amazon Cognito 定价。

主题

- 使用 ID 令牌
- 使用访问令牌
- 使用刷新令牌
- 撤消令牌
- 验证 JSON Web 令牌
- 缓存令牌

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。

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 令牌默认有效载荷

这是来自 ID 令牌的有效负载示例。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 标准声明列表。您可以使用添加对自己设计的声明令牌生成前 Lambda 触发器。

```
<header>.{
    "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "cognito:groups": [
        "test-group-a",
        "test-group-b",
        "test-group-c"
    ],
    "email_verified": true,
    "cognito:preferred_role": "arn:aws:iam::111122223333:role/my-test-role",
    "iss": "https://cognito-idp.us-west-2.amazonaws.com/us-west-2_example",
    "cognito:username": "my-test-user",
    "middle_name": "Jane",
    "nonce": "abcdefg",
    "origin_jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "cognito:roles": [
        "arn:aws:iam::111122223333:role/my-test-role"
    ],
    "aud": "xxxxxxxxxxxxexample",
    "identities": [
        {
            "userId": "amzn1.account.EXAMPLE",
            "providerName": "LoginWithAmazon",
            "providerType": "LoginWithAmazon",
            "issuer": null,
            "primary": "true",
            "dateCreated": "1642699117273"
        }
    ],
    "event_id": "64f513be-32db-42b0-b78e-b02127b4f463",
    "token_use": "id",
    "auth_time": 1676312777,
    "exp": 1676316377,
    "iat": 1676312777,
    "jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "email": "my-test-user@example.com"
}
.<token signature>
```

sub

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。组可以是您提供给应用程序的标识符，也可以从身份池为首选 IAM 角色生成请求。

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。有关您的用户群体如何选择此角色声明的更多信息，请参阅将优先级值分配到组。

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>`

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或 API 操作撤销了用户的令牌时会引用该origin_jti声明。RevokeToken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。每个用户群体组可以有一个与之关联的 IAM 角色。此数组显示了您的用户组的所有 IAM 角色，不按优先级排列。有关更多信息，请参阅 向用户池添加组。

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。

identities

用户的 `identities` 属性的内容。该属性包含有关您通过联合登录或通过将联合用户与本地配置文件关联而与用户关联的每个第三方身份提供者配置文件的信息。此信息包含其提供者名称、提供者唯一 ID 和其它元数据。

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前缀为前缀。`custom:`

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。

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 端点 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。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。

访问令牌默认负载

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。您可以使用添加对自己设计的声明令牌生成前 Lambda 触发器。

```
<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或 API 操作撤销了用户的令牌时会引用该origin_jti声明。RevokeToken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 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。

`username`

用户群体中用户的用户名。

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 验证 JSON Web 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。

使用刷新令牌获取新的访问权限和身份令牌

使用 API 或托管 UI 来启动刷新令牌的身份验证。

要使用刷新令牌通过用户池 API 获取新 ID 和访问令牌，请使用AdminInitiateAuth或 InitiateAuthAPI 操作。为 `AuthFlow` 参数传递 `REFRESH_TOKEN_AUTH`。在 `AuthFlow` 的 `AuthParameters` 属性中，将用户的刷新令牌作为 `"REFRESH_TOKEN"` 的值进行传递。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**
>
> 要使用 Amazon Cognito 用户群体 API 刷新托管 UI 用户的令牌，请生成 `InitiateAuth` 请求。

您也可以将刷新令牌提交给用户群体中您在其中配置了域的令牌端点。在请求正文中，加入 `refresh_token` 的 `grant_type` 值和用户刷新令牌的 `refresh_token` 值。

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。

> **Note**
>
> 撤消刷新令牌将撤消 Amazon Cognito 从具有该令牌的刷新请求中发出的所有 ID 和访问令牌。

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。注销用户后，会发生以下影响。

- 用户的刷新令牌无法获取该用户的新令牌。
- 用户的访问令牌无法发出经过令牌授权的 API 请求。
- 用户必须重新进行身份验证以获取新的令牌。由于托管 UI 会话 Cookie 不会自动过期，因此，用户可以使用会话 Cookie 重新进行身份验证，而无需额外提示输入凭证。注销托管 UI 用户后，将这些用户重定向到注销端点，Amazon Cognito 将在其中清除用户的会话 cookie。

使用刷新令牌，您可以将用户的会话长时间保留在您的应用程序中。随着时间推移，您的用户可能希望取消对他们已登录的某些设备的授权，从而不断刷新他们的会话。要将您的用户从单个设备注销，请撤消其刷新令牌。当您的用户想要退出所有经过身份验证的会话时，请生成 GlobalSignOutAPI 请求。应用程序可以为用户提供一个选择，如从所有设备注销。`GlobalSignOut` 接受用户的有效（即未更改、未过期、未撤消的）访问令牌。由于此 API 经过令牌授权，因此一个用户无法使用它来发起另一个用户的注销。

但是，您可以生成一个 AdminUserGlobalSignOutAPI 请求，该请求由您使用您的 AWS 凭据进行授权，以便将任何用户从其所有设备上注销。管理员应用程序必须使用 AWS 开发者凭据调用此 API 操作，并将用户池 ID 和用户名作为参数传递。`AdminUserGlobalSignOut` API 可以在用户池中注销任何用户。

有关您可以使用 AWS 凭证或用户的访问令牌授权的请求的更多信息，请参阅Amazon Cognito 用户池经过身份验证和未经身份验证的 API 操作。

撤消令牌

您可以使用 AWS API 撤消用户的刷新令牌。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`。

使用 AWS Management Console、或 AWS API 创建新的用户池客户端时 AWS CLI，默认情况下会启用令牌撤销。

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。

要创建或修改启用了令牌撤销的应用程序客户端，请在您的CreateUserPoolClient或UpdateUserPoolClientAPI 请求中包含以下参数。

```
"EnableTokenRevocation": true
```

撤消令牌

您可以使用 RevokeTokenAPI 请求撤消刷新令牌，例如使用 `aws cognito-idp revoke-token` CLI 命令。您也可以使用撤消端点撤消令牌。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**
>
> 您撤消刷新令牌的请求必须包括用于获取令牌的客户端 ID。

以下是示例 RevokeToken API 请求的正文。

```
{
   "ClientId": "1example23456789",
   "ClientSecret": "abcdef123456789ghijklexample",
   "Token": "eyJjdHkiOiJKV1QiEXAMPLE"
}
```

以下是对使用自定义域的用户群体的 /oauth2/revoke 端点发出的 cURL 请求示例。

```
curl --location 'auth.mydomain.com/oauth2/revoke' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Authorization: Basic Base64Encode(client_id:client_secret)' \
--data-urlencode 'token=abcdef123456789ghijklexample' \
--data-urlencode 'client_id=1example23456789'
```

除非您的应用程序客户端具有客户端密钥，否则 RevokeToken 操作和 /oauth2/revoke 端点不需要额外授权。

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
- 使用验证令牌 aws-jwt-verify
- 了解和检查令牌

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。 AWS 软件开发工具包为应用程序中的 Amazon Cognito 用户池令牌处理和管理提供了工具。 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
- 使用软件开发工具包的 Amazon Cognito 身份提供商的代码示例 AWS
- 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。

使用验证令牌 aws-jwt-verify

在 Node.js 应用程序中， AWS 建议使用该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`。

有关您可以在 Node.js 应用程序或 AWS Lambda 授权方中使用的更多信息和示例代码，请参阅aws-jwt-verify上 GitHub的。

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。

标题

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。

   您还可以使用 AWS Lambda 解码用户池 JWT。有关更多信息，请参阅使用解码和验证 Amazon Cognito J WT 令牌。 AWS Lambda
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. 如果您的访问令牌包含aws.cognito.signin.user.admin索赔，请向类似的 API 发送请求GetUser。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 端点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：

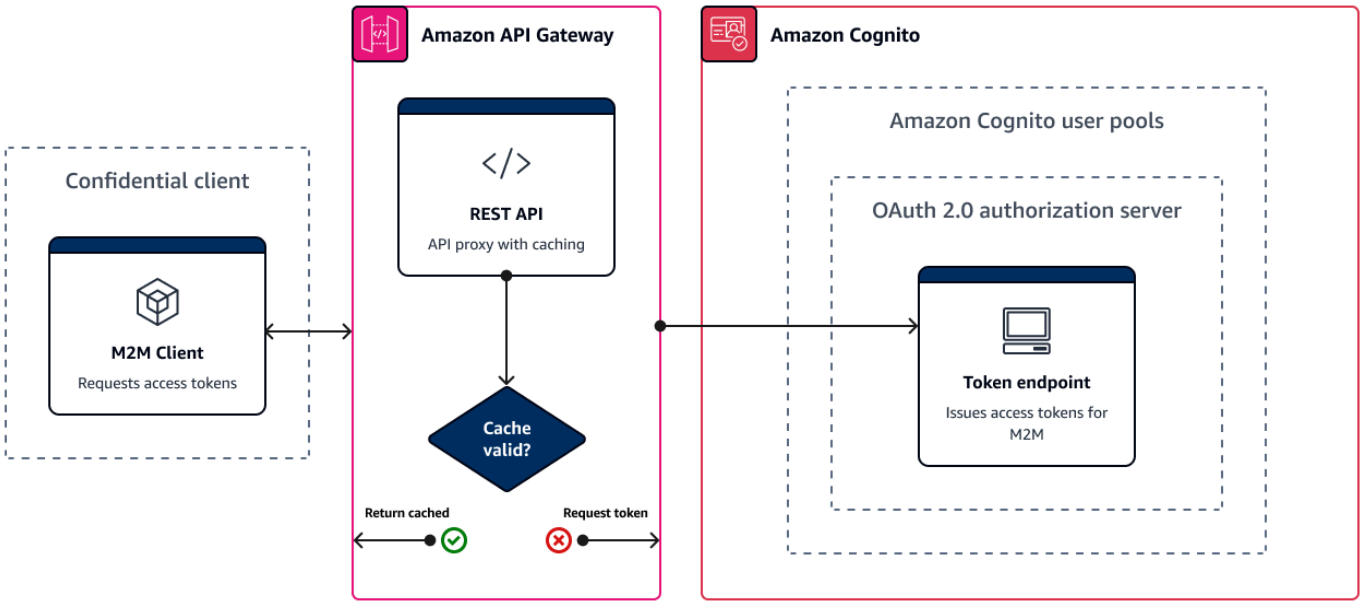
   `https://cognito-idp.us-east-1.amazonaws.com/<userpoolID>`.

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 请求中对本地用户进行身份验证，例如。InitiateAuth

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。

当您的用户池生成大量用户或 machine-to-machine 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定的限制。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。

使用 Amazon API Gateway 缓存 machine-to-machine 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，OAuth 2.0 客户端凭证授予通常是大批量的授予类型，用于生成用于授权 machine-to-machine 和微服务会话的访问令牌。在诸如流量激增导致您的微服务水平扩展的情况下，您最终可能会有许多系统使用相同的客户端凭据，但其数量超过用户池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。

您也可以使用 for Redis ElastiCache 实现令牌缓存。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. 选择至少 3600 秒的缓存 time-to-live (TTL)。
   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。

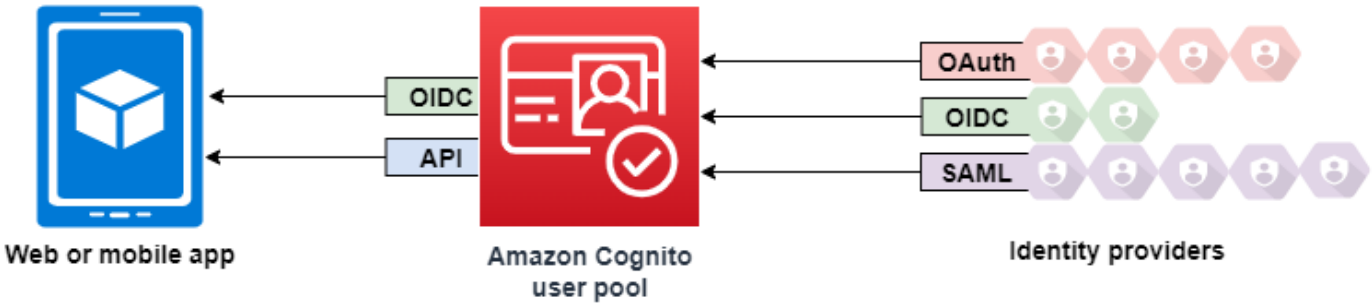
在成功进行用户池身份验证后访问资源

您的应用程序用户可以直接通过用户池登录，也可以通过第三方身份提供商 (IdP) 进行联合。用户池管理处理通过Facebook、谷歌、亚马逊和苹果进行社交登录以及从OpenID Connect（OIDC）和SAML返回的代币的开销。 IdPs有关更多信息，请参阅 将令牌与用户池结合使用。

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。您可以使用用户池令牌来：

- 在 Amazon DynamoDB 和 Amazon S3 AWS 服务 等中检索授权应用程序资源请求的 AWS 证书。
- 提供临时的、可撤销的身份验证证明。
- 将身份数据填充到应用程序中的用户个人资料中。
- 授权更改用户池目录中已登录用户的个人资料。
- 使用访问令牌对用户信息的请求进行授权。
- 使用访问令牌授权对受访问保护的外部 API 后面的数据的请求。
- 使用 Amazon Verified 权限授权访问存储在客户端或服务器上的应用程序资产。

有关更多信息，请参阅 用户池身份验证流程 和 将令牌与用户池结合使用。



主题

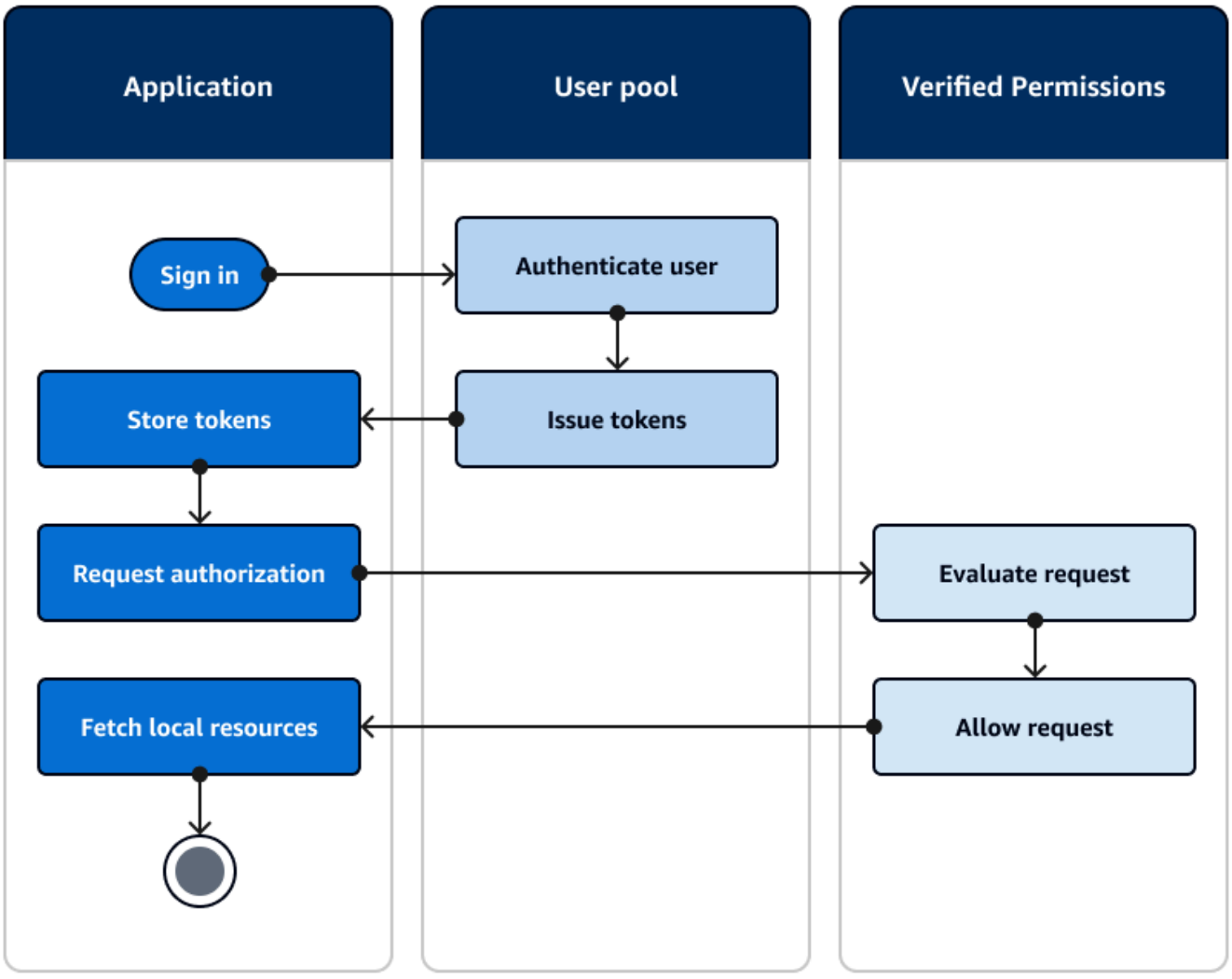
- 使用 Amazon 验证权限授权访问客户端或服务器资源
- 登录后使用 API Gateway 访问资源

- 登录后 AWS 服务 使用身份池进行访问

使用 Amazon 验证权限授权访问客户端或服务器资源

您的应用程序可以将登录用户的令牌传递给 A ma zon 验证权限。Verified Permissions 是一项可扩展、精细的权限管理和授权服务，适用于您构建的自定义应用程序。Amazon Cognito 用户池可以作为已验证权限策略存储的身份来源。Verified Permissions 会根据用户池令牌中的委托人及其属性对请求的操作和资源（`GetPhoto`例如`premium_badge.png`）做出授权决定。

下图显示了您的应用程序如何在授权请求中将用户的令牌传递给已验证的权限。



开始使用 Amazon 验证权限

将用户池与经过验证的权限集成后，您将获得所有 Amazon Cognito 应用程序的精细授权的集中来源。这消除了对精细安全逻辑的需求，否则您将不得不在所有应用程序之间进行编码和复制。有关使用已验证权限进行授权的更多信息，请参阅使用 Amazon Verified Permissions 进行授权。

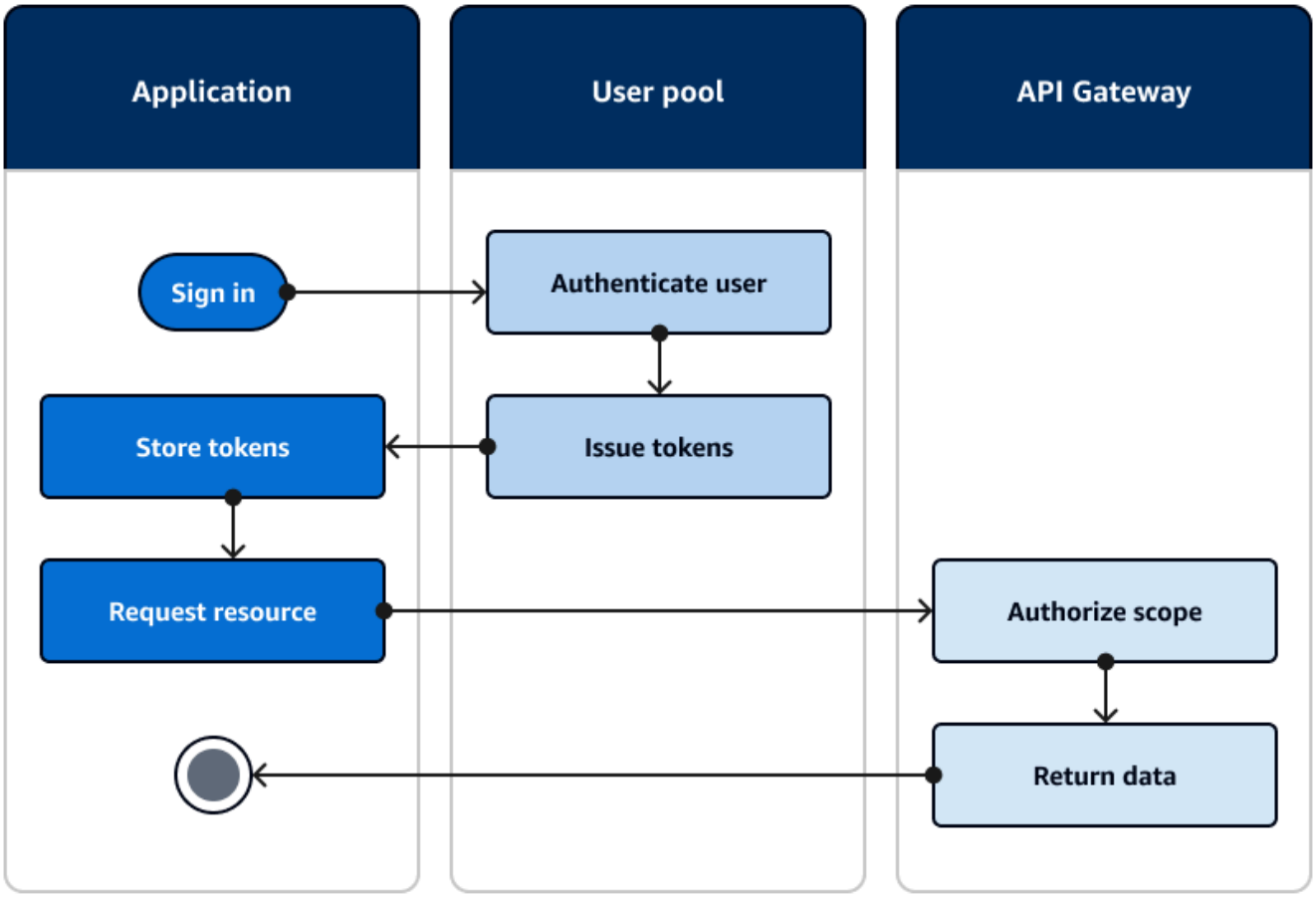
已验证的权限授权请求需要 AWS 凭证。您可以实施以下一些技术来安全地将凭据应用于授权请求。

- 操作可以在服务器后端存储机密的 Web 应用程序。
- 获取经过身份验证的身份池凭证。
- 通过 access-token-authorized API 代理用户请求，并将 AWS 凭据附加到请求中。

登录后使用 API Gateway 访问资源

Amazon Cognito 用户池令牌的常见用途是授权向 API Gate way REST AP I 发出的请求。访问令牌中的 OAuth 2.0 作用域可以对方法和路径进行授权，比如`HTTP GET`。`/app_assets`ID 令牌可以用作 API 的通用身份验证，也可以将用户属性传递给后端服务。API Gateway 还有其他自定义授权选项，比如 HTTP API 的 JWT 授权方和可以应用更精细逻辑的 Lambda 授权者。

下图说明了访问令牌中具有 OAuth 2.0 范围的 REST API 的应用程序。



您的应用程序必须从经过身份验证的会话中收集令牌，并将其作为不记名令牌添加到请求的`Authorization`标头中。配置您为评估令牌内容的 API、路径和方法配置的授权方。只有当请求符合您为授权方设置的条件时，API Gateway 才会返回数据。

API Gateway API 可以通过以下方式批准来自应用程序的访问权限：

- 访问令牌包含正确的 OAuth 2.0 范围。REST API 的 Amazon Cognito 用户池授权者是一种常见的实现，进入门槛很低。您还可以评估向此类授权方发出的请求的正文、查询字符串参数和标头。
- ID 令牌有效且未过期。当您将身份令牌传递给 Amazon Cognito 授权机构时，您可以在应用程序服务器上对 ID 令牌内容进行额外验证。
- 访问或 ID 令牌中的群组、声明、属性或角色符合您在 Lambda 函数中定义的要求。Lambda 授权方解析请求标头中的令牌并对其进行评估以做出授权决策。您可以在函数中构建自定义逻辑，也可以向 Amazon 验证权限发出 API 请求。

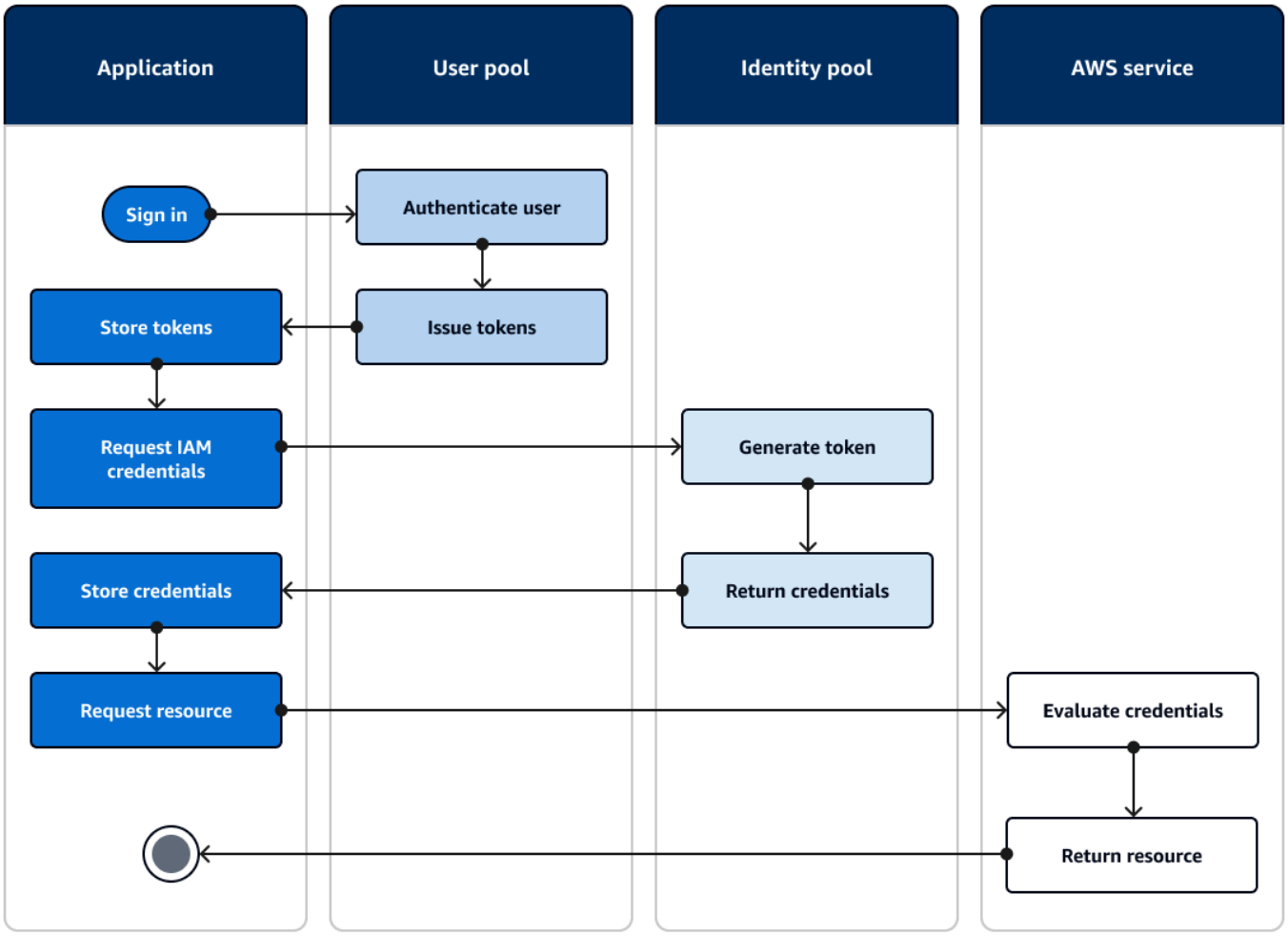
您还可以使用来自用户池的令牌来授权AWS AppSync 向 GraphQL API 发出的请求。

登录后 AWS 服务 使用身份池进行访问

用户使用用户池登录后，他们可以使用身份池颁发的临时 API 凭证进行访问 AWS 服务 。

您的网络或移动应用程序接收来自用户池的令牌。当您将用户池配置为身份池的身份提供者时，身份池会用令牌交换临时 AWS 证书。这些证书的范围可以限于 IAM 角色及其策略，这些策略允许用户访问有限的 AWS 资源。有关更多信息，请参阅 身份池（联合身份）身份验证流程。

下图显示了应用程序如何使用用户池登录、检索身份池凭证以及如何从中请求资产。 AWS 服务



您可以使用身份池凭证来：

- 使用您的用户自己的凭证向 Amazon Verified Permissions 提出精细的授权请求。
- 连接到 Amazon API Gateway REST API 或授权与 IAM 连接的 AWS AppSync GraphQL API。

- 连接到授权与 IAM 连接的数据库后端，例如亚马逊 DynamoDB 或 Amazon RDS。
- 从 Amazon S3 存储桶中检索应用程序资产。
- 使用 Amazon WorkSpaces 虚拟桌面启动会话。

身份池不只在经过身份验证的用户池会话中运行。他们还直接接受来自第三方身份提供商的身份验证，并且可以为未经身份验证的访客用户生成凭证。

有关使用身份池和用户池组来控制对 AWS 资源的访问的更多信息，请参阅向用户池添加组和使用基于角色的访问控制。另外，有关身份池和的更多信息 AWS Identity and Access Management，请参阅身份池概念。

使用设置用户池 AWS Management Console

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。

使用设置身份池 AWS Management Console

以下过程介绍如何使用将 AWS Management Console 身份池与一个或多个用户池和客户端应用程序集成。

添加 Amazon Cognito 用户群体身份提供者 (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 Amazon Cognito 用户群体。
5. 输入用户群体 ID 和应用程序客户端 ID。
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。

    a. 您可以为来自该 IdP 的用户提供您在配置身份验证角色时设置的默认角色，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。

        i. 如果您选择 “选择带规则的角色”，请输入来自用户身份验证的来源声明、用于将声明与规则进行比较的运算符、将导致与该角色选择匹配的值，以及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. 如果您在令牌中选择带有 preferred_role 声明的角色，Amazon Cognito 会为您的用户声明中的角色颁发证书。`cognito:preferred_role`如果不存在首选角色声明，Amazon Cognito 将根据您的角色解析发放凭证。

b. 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。

- 如果不应用主体标签，请选择非活动。
- 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。
- 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. 选择保存更改。

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>
```

您可以<region>从用户池 ID 中推导出值。例如，如果用户池 ID 为`us-east-1_EXAMPLE1`，则<region>为`us-east-1`。如果用户池 ID 为`us-west-2_EXAMPLE2`，则<region>为`us-west-2`。

JavaScript

```
var cognitoUser = userPool.getCurrentUser();

if (cognitoUser != null) {
 cognitoUser.getSession(function(err, result) {
  if (result) {
   console.log('You are now logged in.');

   // Add the User's Id Token to the Cognito credentials login map.
   AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'YOUR_IDENTITY_POOL_ID',
    Logins: {
```

```
     'cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>':
 result.getIdToken().getJwtToken()
    }
   });
  }
 });
}
```

Android

```
cognitoUser.getSessionInBackground(new AuthenticationHandler() {
 @Override
 public void onSuccess(CognitoUserSession session) {
  String idToken = session.getIdToken().getJWTToken();

  Map<String, String> logins = new HashMap<String, String>();
  logins.put("cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>",
 session.getIdToken().getJWTToken());
  credentialsProvider.setLogins(logins);
 }

});
```

iOS - objective-C

```
AWSServiceConfiguration *serviceConfiguration = [[AWSServiceConfiguration alloc]
 initWithRegion:AWSRegionUSEast1 credentialsProvider:nil];
AWSCognitoIdentityUserPoolConfiguration *userPoolConfiguration =
 [[AWSCognitoIdentityUserPoolConfiguration alloc] initWithClientId:@"YOUR_CLIENT_ID"
  clientSecret:@"YOUR_CLIENT_SECRET" poolId:@"YOUR_USER_POOL_ID"];
[AWSCognitoIdentityUserPool
 registerCognitoIdentityUserPoolWithConfiguration:serviceConfiguration
 userPoolConfiguration:userPoolConfiguration forKey:@"UserPool"];
AWSCognitoIdentityUserPool *pool = [AWSCognitoIdentityUserPool
 CognitoIdentityUserPoolForKey:@"UserPool"];
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider
 alloc] initWithRegionType:AWSRegionUSEast1 identityPoolId:@"YOUR_IDENTITY_POOL_ID"
 identityProviderManager:pool];
```

iOS - swift

```
let serviceConfiguration = AWSServiceConfiguration(region: .USEast1,
 credentialsProvider: nil)
```

```
let userPoolConfiguration = AWSCognitoIdentityUserPoolConfiguration(clientId:
 "YOUR_CLIENT_ID", clientSecret: "YOUR_CLIENT_SECRET", poolId: "YOUR_USER_POOL_ID")
AWSCognitoIdentityUserPool.registerCognitoIdentityUserPoolWithConfiguration(serviceConfigura
 userPoolConfiguration: userPoolConfiguration, forKey: "UserPool")
let pool = AWSCognitoIdentityUserPool(forKey: "UserPool")
let credentialsProvider = AWSCognitoCredentialsProvider(regionType: .USEast1,
 identityPoolId: "YOUR_IDENTITY_POOL_ID", identityProviderManager:pool)
```

使用 Amazon Cognito 用户池安全功能

可以向用户池添加多重身份验证（MFA）以保护用户的身份。MFA 增加了第二种身份验证要素，这样用户池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
- 向用户池添加高级安全
- 将 AWS WAF Web ACL 与用户池关联
- 用户池区分大小写
- 用户池删除保护
- 管理用户存在错误响应

向用户池添加 MFA

多重身份验证（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。

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 设置为必需并且激活了 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 设置为必需并且您激活的唯一 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 提出了设置 TOTP MFA 以响应 InitiateAuthAPI 操作的挑战。AdminInitiateAuth

当您将 MFA 设置为必需时，托管 UI 会提示用户设置 MFA。当您在用户池中将 MFA 设置为可选时，托管 UI 不会提示用户。要使用可选的 MFA，您必须在应用程序中构建一个界面，以提示用户选择他们需要设置 MFA，然后引导他们完成 API 输入以验证他们的额外登录因素。

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中所描述的指数超时锁定过程。

主题

- 先决条件
- 配置身份多重验证
- SMS 文本消息 MFA
- TOTP 软件令牌 MFA

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 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`。

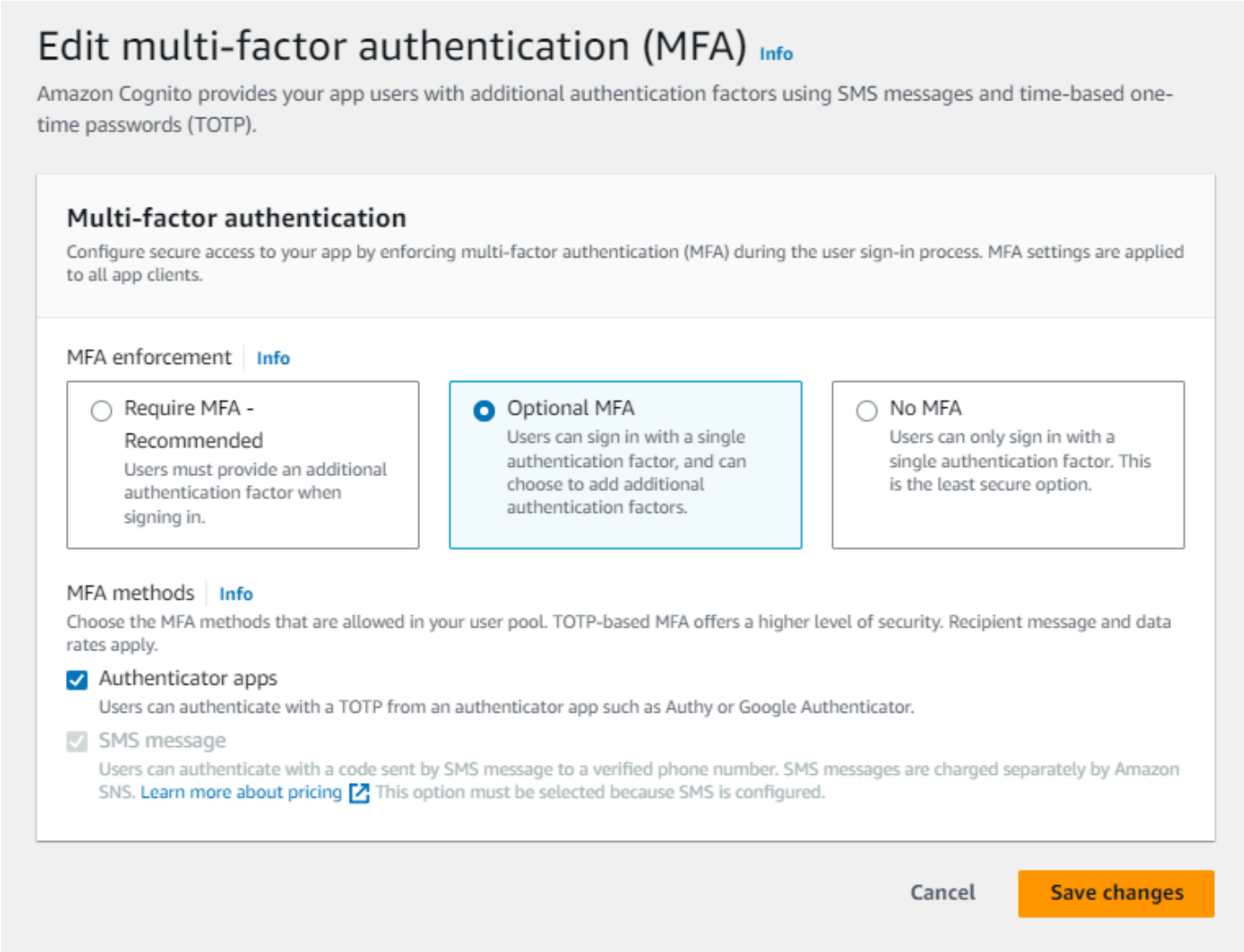
- 如果您的账户位于包含您的用户池的亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 资源的短信沙箱中，则必须先验证亚马逊 SNS 中的电话号码，然后才能发送短信。 AWS 区域 有关更多信息，请参阅 Amazon Cognito 用户池的短信设置。
- 高级安全功能要求您已激活 MFA 并在 Amazon Cognito 用户池控制台中将其设置为可选。有关更多信息，请参阅 向用户池添加高级安全。

配置身份多重验证

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. 选择登录体验选项卡。找到多重身份验证，然后选择编辑
5. 选择您希望用于用户池的MFA 强制执行方法。



a. 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. 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。

c. 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 方法。您可以将 SMS 消息或可生成 TOTP 的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。在用户池的消息收发选项卡下，找到 SMS 并选择编辑。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身份验证流程会话持续时间有效。

当您在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。

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。具有必要 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。

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。可以为 AWS 账户和单条消息指定支出限额，且该限制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。如果您想提高限额，请在 AWS Support 中心提交 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。有关用户池中包含 Amazon SNS 的短信的更多信息，请参阅。Amazon Cognito 用户池的短信设置

TOTP 软件令牌 MFA

当您在用户池中设置了 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，您可以调用SetUserPoolMfaConfig配置 MFA 并启用 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。

当您将用户池配置为需要 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池中，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。

> **Important**
>
> 当你有一个 AWS WAF Web ACL 与用户池相关联，并且你的 Web ACL 中的规则显示了验证码时，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 为托管用户界面 TOTP MFA 配置 AWS WAF 网页 ACL。有关 AWS WAF 网页 ACL 和 Amazon Cognito 的更多信息，请参阅。将 AWS WAF Web ACL 与用户池关联

要在使用 Amazon Cognito API 的自定义 UI 中实施 TOTP MFA，请参阅 在 Amazon Cognito 用户池 API 中为用户配置 MFA。

要向用户池添加 MFA，请参阅 向用户池添加 MFA。

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。
   - Amazon Cognito 不会向使用或 API 操作登录的管理员创建的用户发放一次性令牌。InitiateAuthAdminInitiateAuth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。您可以代替访问令牌，而是在请求中使用`MFA_SETUP`质询InitiateAuth或AssociateSoftwareToken请求AdminInitiateAuth中的`session`响应值。
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。
5. Amazon Cognito 仅接受来自使用 SHA-1 哈希函数生成代码的身份验证器应用程序的 TOTP。使用 SHA-256 哈希生成的代码会返回 `Code mismatch` 错误。

在 Amazon Cognito 用户池 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
- 验证 TOTP 令牌
- 使用 TOTP MFA 登录
- 删除 TOTP 令牌

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，AssociateSoftwareToken调用返回为用户帐户生成的唯一共享密钥代码。您可以使用访问令牌或会话字符串进行授权 AssociateSoftwareToken 。
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，请重试该 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。对于每次登录，都RespondToAuthChallenge应调用以获取对新 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您的用户停用他们的 TOTP 配置。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 mf SetUserapReference 将您的用户修改为不使用 MF A 或仅使用 SMS MFA。

1. 在您的应用程序中为想要重置 MFA 的用户创建界面。在此界面中提示用户输入密码。
2. 如果 Amazon Cognito 返回 TOTP MFA 质询，请使用 mfapReference 更新用户的 MFA 偏好。SetUser
3. 在您的应用程序中，告知您的用户他们已停用 MFA 并提示他们重新登录。

为托管用户界面 TOTP MFA 配置 AWS WAF 网页 ACL

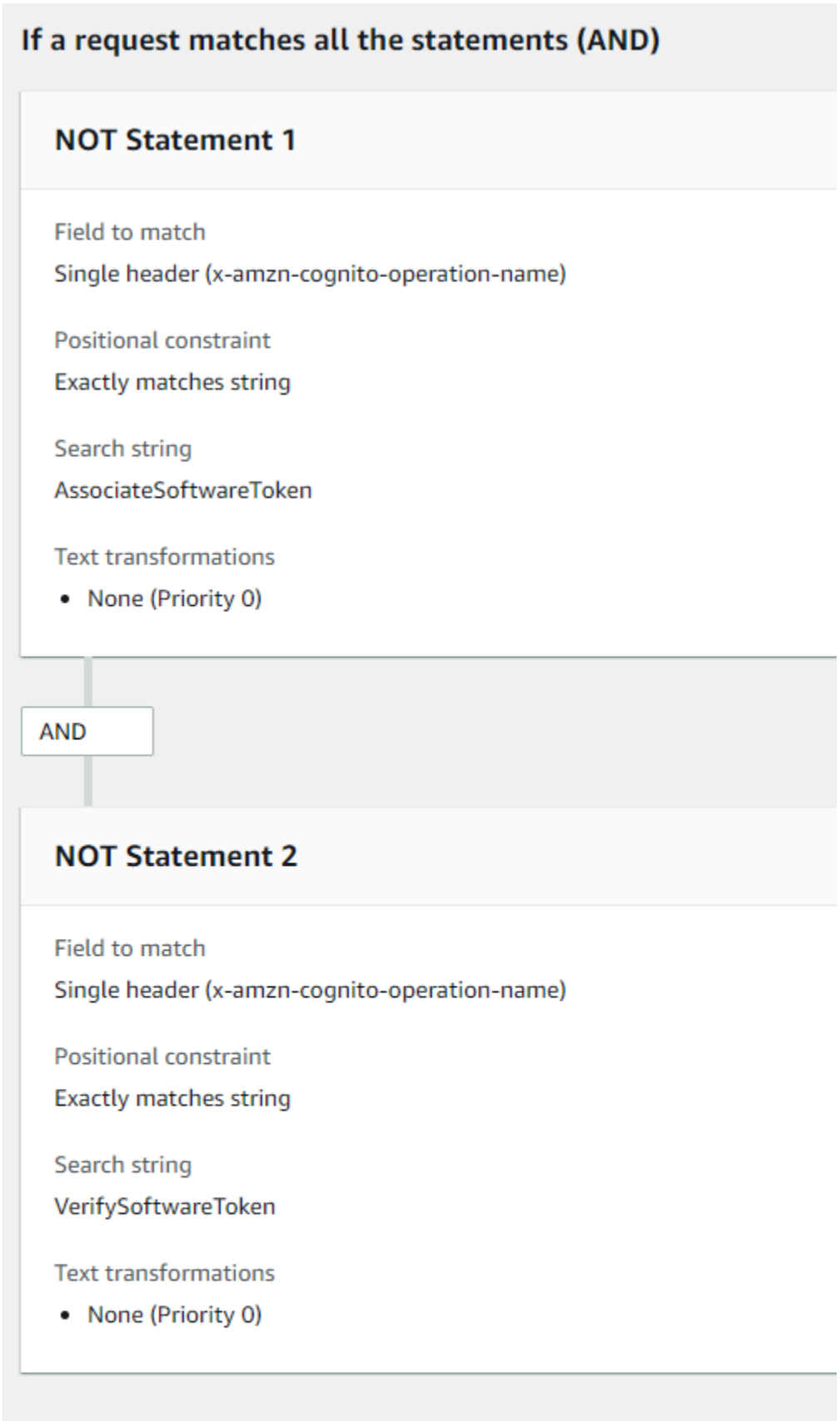
当你有一个 AWS WAF Web ACL 与用户池相关联，并且你的 Web ACL 中的规则显示了验证码时，这可能会导致托管 UI TOTP 注册中出现不可恢复的错误。 AWS WAF 验证码规则仅以这种方式影响托管用户界面中的 TOTP MFA。SMS MFA 不受影响。

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。

由于 WAF captcha，不允许请求。

当你的用户池在后台发出的 VerifySoftwareTokenAPI 请求时 AWS WAF 提示输入验证码时，就会出现此错误。AssociateSoftwareToken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。

以下屏幕截图显示了一个示例 AWS WAF 规则，该规则将验证码操作应用于`x-amzn-cognito-operation-name`标头值不为或的所有请求。`AssociateSoftwareToken` `VerifySoftwareToken`



Action

The action to take when a web request matches the rule statement.

有关 AWS WAF 网页 ACL 和 Amazon Cognito 的更多信息，请参阅。将 AWS WAF Web ACL 与用户池关联

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。在审计模式下，高级安全功能会向 Amazon 发布指标 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。

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池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。

访问令牌自定义

激活高级安全功能时，您可以将用户池配置为接受对版本 2 Lambda 触发器事件的响应。在版本 2 中，您可以在访问令牌中自定义范围和其他声明。这增强了您在用户进行身份验证时创建灵活授权结果的能力。有关更多信息，请参阅自定义访问令牌。

主题

- 注意事项和限制

- 先决条件
- 配置高级安全功能
- 检查已泄露的凭证
- 使用自适应身份验证
- 查看高级安全指标
- 通过应用程序激活用户池高级安全功能

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 通过以下标准身份验证流程支持自适应身份验证和凭证受损检测：、和。USER_PASSWORD_AUTH ADMIN_USER_PASSWORD_AUTH USER_SRP_AUTH无法在 CUSTOM_AUTH 流和 自定义身份验证质询 Lambda 触发器 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。要对用户池中的容量攻击实施防御，请添加 AWS WAF Web ACL。有关更多信息，请参阅 将 AWS WAF Web ACL 与用户池关联。
- 客户凭证授予的目的是在与用户帐户无关的情况下进行 machine-to-machine （M2M）授权。高级安全功能仅监控用户池中的用户账户和密码。要在 M2M 活动中实现安全功能，请考虑 AWS WAF 用于监控请求率和内容的功能。有关更多信息，请参阅 将 AWS WAF Web ACL 与用户池关联。

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。
- 在 Amazon Cognito 控制台中将多重身份验证（MFA）设置为可选,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。
- 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，请输入您的 AWS 凭据。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择应用程序集成选项卡。找到高级安全，然后选择启用。如果之前启用了高级安全，请选择编辑。
5. 选择完整功能以配置对已泄露的凭证和自适应身份验证的高级安全响应。选择 “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仅审计和完整功能模式。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. 如果您已选择仅审计，请选择保存更改。如果您已选择完整功能：

   a. 选择是否要进行自定义操作或使用 Cognito 默认设置响应可疑已泄露的凭证。Cognito 默认设置：

      i. 检测登录、注册和密码更改中的已泄露的凭证。

      ii. 使用禁止登录操作响应已泄露的凭证。

   b. 如果您为已泄露的凭证选择了自定义操作，请选择 Amazon Cognito 将用于事件检测的用户池操作，以及您希望 Amazon Cognito 执行的 已泄露凭证的响应。您可以使用可疑的已泄露凭证进行禁止登录或允许登录。

   c. 在自适应身份验证下，选择如何响应恶意登录尝试。选择是否要进行自定义操作或使用 Cognito 默认设置响应可疑恶意活动。当您选择 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. 如果您针对自适应身份验证已选择自定义操作，请根据严重性级别选择 Amazon Cognito 对检测到的风险执行的 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. 允许登录 – 不采取任何预防性操作。

ii. 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。

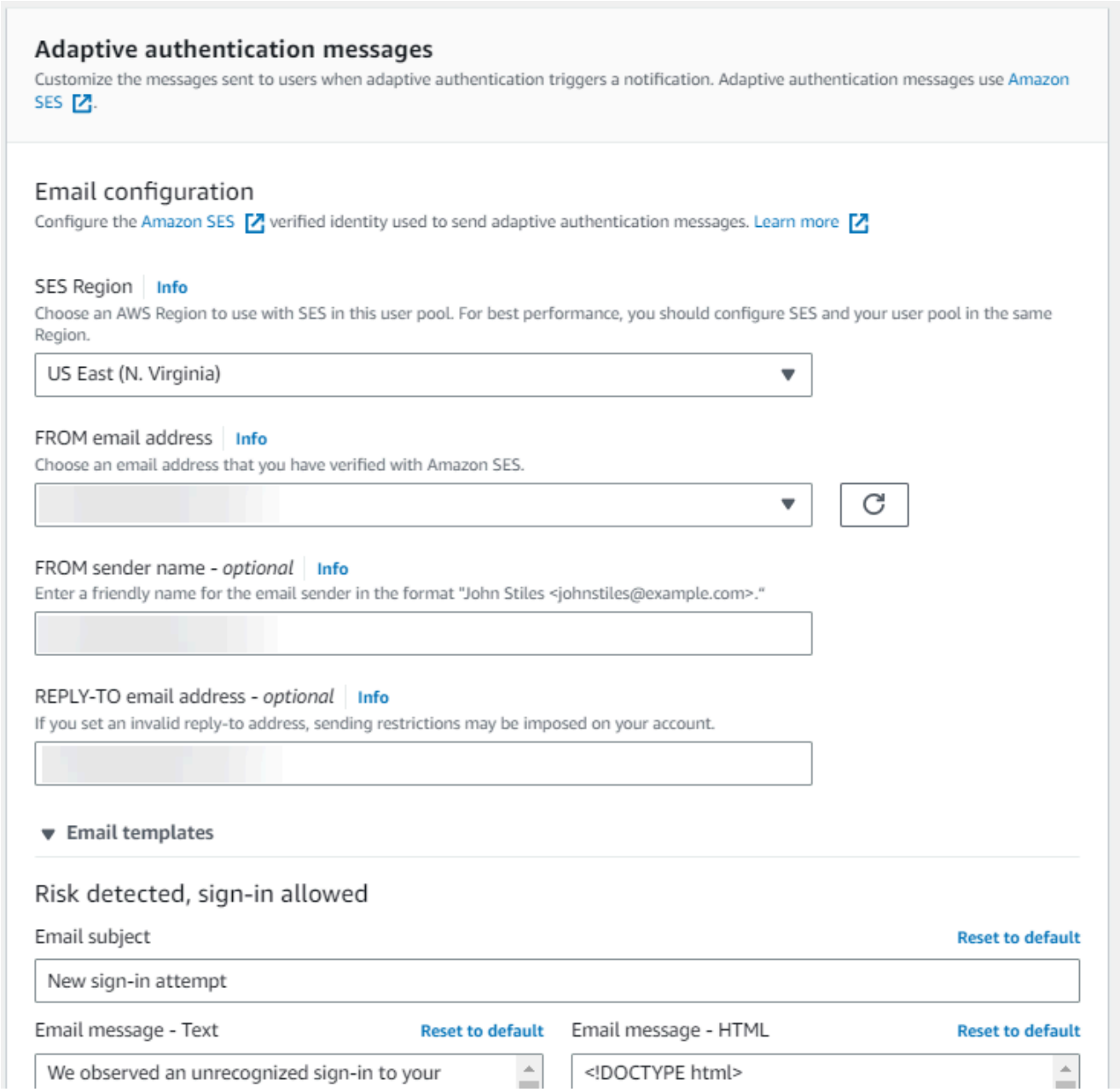
iii. 需要 MFA – 如果用户配置了 MFA，Amazon Cognito 将始终要求用户在登录时提供其它短信或 TOTP 因素。如果用户没有配置 MFA，Amazon Cognito 将提示他们设置 MFA。在您自动要求用户使用 MFA 之前，请在应用程序中配置一种机制来捕获 SMS MFA 的电话号码，或为 TOTP MFA 注册身份验证器应用程序。

iv. 禁止登录 – 阻止用户登录。

v. 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. 在邮件配置下，选择您希望与自适应身份验证一起使用的 SES 区域、FROM 电子邮件地址、FROM 发件人名称和 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。

8. 展开 IP 地址异常以创建始终允许或者始终阻止的 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. 选择保存更改。

检查已泄露的凭证

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池目录中，无需通过外部 IdP 进行联合身份验证。

从 Amazon Cognito 控制台的应用程序集成选项卡的高级安全功能中，您可以配置已泄露的凭证。配置事件检测以选择要监控的用户事件，以查看是否有已泄露的凭证。配置已泄露凭证的响应，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。

选择 “允许登录” 后，您可以查看 Amazon CloudWatch Logs 以监控 Amazon Cognito 对用户事件所做的评估。有关更多信息，请参阅 查看高级安全指标。当您选择禁止登录时，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 会检查使用带流AdminInitiateAuth的 API 和带`ADMIN_USER_PASSWORD_AUTH`流的 InitiateAuthAPI 的登录凭证是否遭到`USER_PASSWORD_AUTH`泄露。

要向用户池添加已泄露的凭证保护，请参阅 向用户池添加高级安全。

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池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 向亚马逊发布了登录尝试、其风险等级和失败的挑战。 CloudWatch有关更多信息，请参阅 查看高级安全指标。

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。

主题

- 自适应身份验证概览
- 将用户设备和会话数据添加到 API 请求
- 查看用户事件历史记录
- 提供事件反馈
- 发送通知消息

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。

将强制执行方法从仅限审计更改为完整功能时，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
允许	用户无需额外安全要素即可登录。
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。

选项	操作
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。
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这些数据，然后使用和参数将其提交给 Amazon Cognito 高级安全部门。ContextDataUserContextData上下文数据收集库包含在 AWS 软件开发工具包中。有关更多信息，请参阅将 Amazon Cognito 与 Web 和移动应用程序集成。如果您已在用户池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：

- 您已在用户池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，请参阅配置用户池应用程序客户端。
- 您已在应用程序客户端中激活 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）。

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）

激活接受其他用户上下文数据功能后，用户池在 `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（例如）登录 AdminInitiateAuth ，并且您使用`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 管理您的用户池，然后选择要编辑的用户池。
3. 选择应用程序集成选项卡。

4. 在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，请参阅配置用户池应用程序客户端。
5. 从应用程序客户端信息容器中选择编辑。
6. 在应用程序客户端的高级身份验证设置下，选择接受其他用户上下文数据。
7. 选择保存更改。

要将您的应用程序客户端配置为接受 Amazon Cognito API 中的用户情境数据，请在CreateUserPoolClient或`EnablePropagateAdditionalUserContextData`UpdateUserPoolClient请求`true`中设置为。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池高级安全功能。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 SDK 方法的示例`getData`。

```
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。

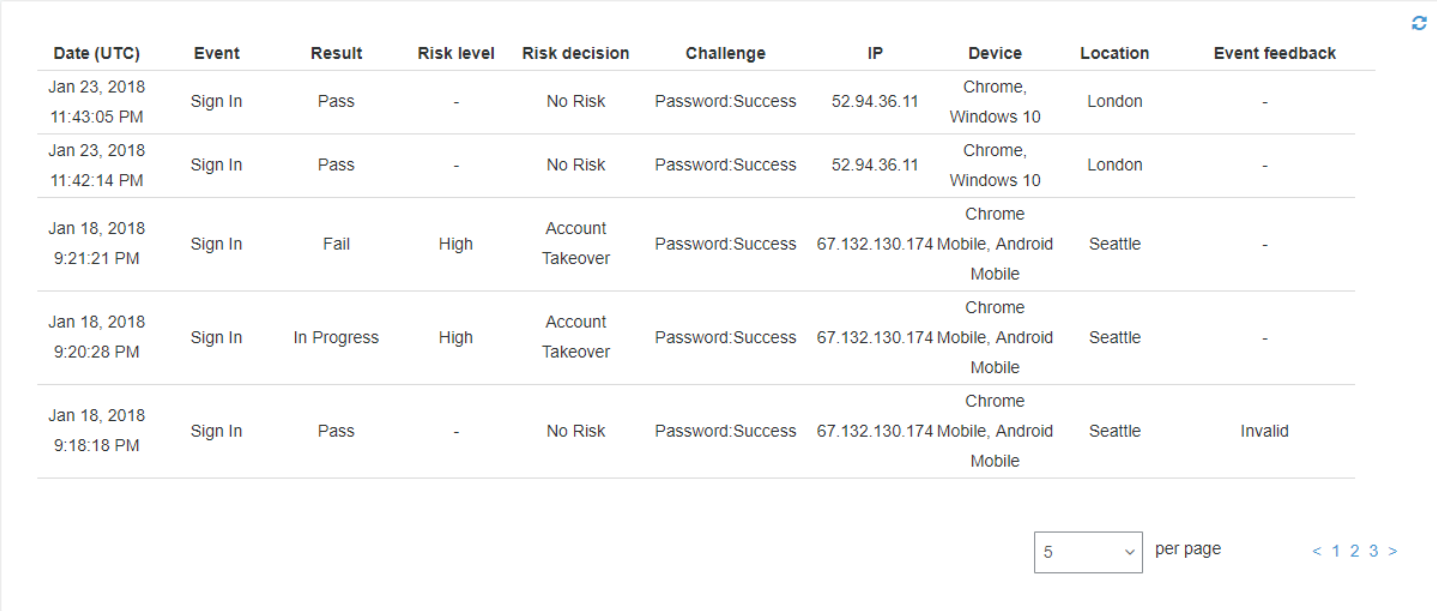
`ContextData` 和 `UserContextData` 资源

- AWS Amplify 适用于 Android 的 SDK：GetUserContextData
- AWS Amplify 适用于 iOS 的 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	Location	Event feedback
Jan 23, 2018 11:43:05 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10	London	-
Jan 23, 2018 11:42:14 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10	London	-
Jan 18, 2018 9:21:21 PM	Sign In	Fail	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	-
Jan 18, 2018 9:20:28 PM	Sign In	In Progress	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	-
Jan 18, 2018 9:18:18 PM	Sign In	Pass	-	No Risk	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	Invalid

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AdminListUserAuthEvents或带有-events 的 AWS Command Line Interface (AWS CLI) 来查询用户事件历史记录。admin-list-user-auth

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。

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，从用户选项卡中选择一个用户，然后选择提供事件反馈。您可以查看事件详细信息，并选择设为有效或设为无效。

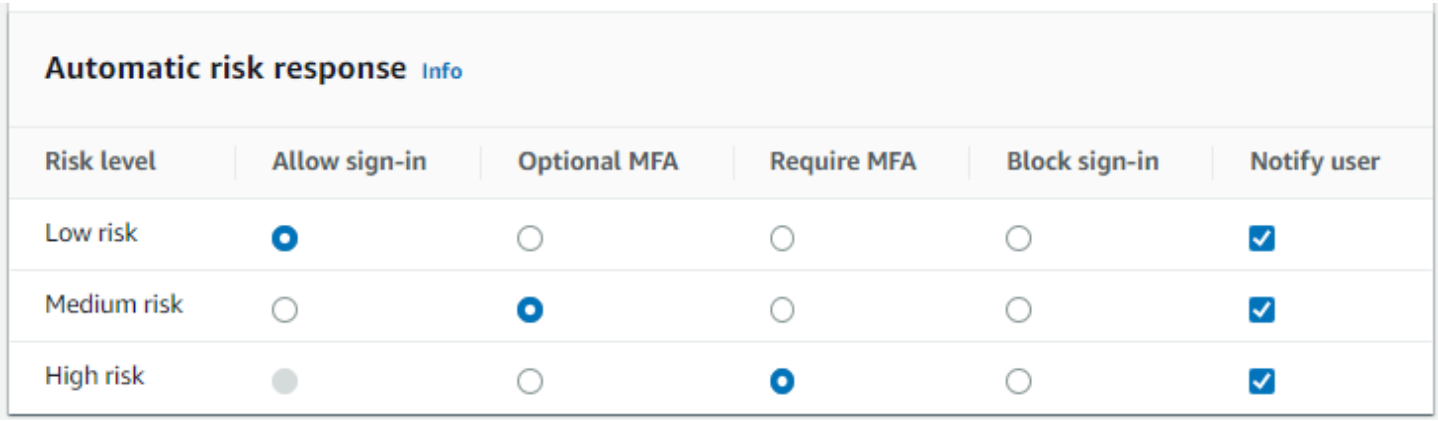
控制台的用户和组选项卡中列出了登录历史记录。您可以选择某个条目来将事件标记为有效或无效。您还可以通过用户池 API 操作和 AWS CLI 命令 admin-update-auth-event-fe AdminUpdateAuthEventFeedbackedback 提供反馈。

当您在 Amazon Cognito 控制台中选择设为有效或在 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。

在自动风险响应部分，对低、中或高风险案例选择通知用户。



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，从自适应身份验证消息中打开电子邮件模板。要了解有关电子邮件模板的更多信息，请参阅 消息模板。

查看高级安全指标

Amazon Cognito 会向您在亚马逊上的账户发布高级安全功能指标。 CloudWatchAmazon 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. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择指标。
3. 选择 Amazon Cognito。

4. 选择一组聚合指标，如按风险分类。
5. 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，然后选择添加到搜索。
   - 要按指标进行筛选，请选择指标名称，然后选择添加到搜索。

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`。 UserPoolId：用户池的标识符。 RiskLevel：高（默认）、中或低。
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：用户池的标识符。 RiskLevel: 高、中或低。
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`。

指标	描述	指标维度
		UserPoolId：用户池的标识符。 RiskLevel: 高、中或低。
Risk	Amazon Cognito 标记为有风险的请求。	Operation：操作类型，例如 `PasswordChange` 、`SignIn` 或 `SignUp`。 UserPoolId：用户池的标识符。
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：用户池的标识符。

Amazon Cognito 为您提供了两组预定义的指标，供您随时进行分析。 CloudWatch按风险分类标识 Amazon Cognito 认定为有风险的请求的风险级别。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。

将高级安全性与 JavaScript

1. 将适用的 Amazon Cognito 身份软件开发工具包 JavaScript添加到您的应用程序中。

2. 在 CognitoUserPool.js 中，设置AdvancedSecurityDataCollectionFlag为。true将 UserPoolId 设置为用户池 ID。
3. 将此源代码引用添加到您的应用程序 JavaScript 文件中。<region>替换为以下 AWS 区域 列表中的：us-east-1、us-east-2、us-west-2eu-west-1、eu-west-2、或eu-central-1。

```
<script src="https://amazon-cognito-assets.<region>.amazoncognito.com/amazon-cognito-advanced-security-data.min.js"></script>
```

对 Android 使用高级安全功能

1. 使用安卓版创建您的应用程序。 AWS Amplify 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 AWS Amplify 适用于 Swift 或 Flutter 的应用程序创建你的应用程序。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。有关在 InitiateAuthAPI 操作中使用的示例，请参阅 InitiateAuthInput+amplify.swift userContextData 中的。 GitHub

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 与用户池关联

AWS WAF 是 Web 应用程序防火墙。借助 AWS WAF 网络访问控制列表 (Web ACL)，您可以保护您的用户池免受对托管用户界面和 Amazon Cognito API 服务终端节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)。

当您的 AWS WAF Web ACL 与用户池关联时，Amazon Cognito 会将您的用户请求中选定的非机密标头和内容转发到。 AWS WAF AWS WAF 检查请求的内容，将其与您在网页 ACL 中指定的规则进行比较，然后向 Amazon Cognito 返回响应。

关于 AWS WAF 网页 ACL 和 Amazon Cognito 的注意事项

- 被阻止的请求 AWS WAF 不计入任何请求类型的请求速率配额。该 AWS WAF 处理程序在 API 级别的限制处理程序之前被调用。
- 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。 AWS WAF `WAFUnavailableEntityException`当您尝试在 Web ACL 完全传播之前将其关联时，将返回。
- 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 开发者指南中的超大请求组件处理，了解如何配置如何 AWS WAF 处理来自 Amazon Cognito 的超大请求。
- 您无法将使用防 AWS WAF 欺诈控制账户盗用 (ATP) 的网络 ACL 与 Amazon Cognito 用户池相关联。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。
- 当你有一个 AWS WAF Web ACL 与用户池相关联，并且你的 Web ACL 中的规则显示了验证码时，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 为托管用户界面 TOTP MFA 配置 AWS WAF 网页 ACL。

AWS WAF 检查对以下端点的请求。

托管 UI

对 用户池联合身份验证端点和托管 UI 参考 中所有端点的请求。

公有 API 操作

您的应用程序向 Amazon Cognito API 发出的不使用 AWS 凭证进行授权的请求。这包括InitiateAuthRespondToAuthChallenge、和 API 操作GetUser。范围内的 API 操作 AWS WAF 不需要使用 AWS 凭据进行身份验证。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 用户池经过身份验证和未经身份验证的 API 操作。

您可以使用以下规则操作在 Web ACL 中配置规则：计数、允许、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 用户池 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 软件开发工具包`ForbiddenException`会向生成区块或验证码响应的请求返回错误。

将 Web ACL 与您的用户池相关联

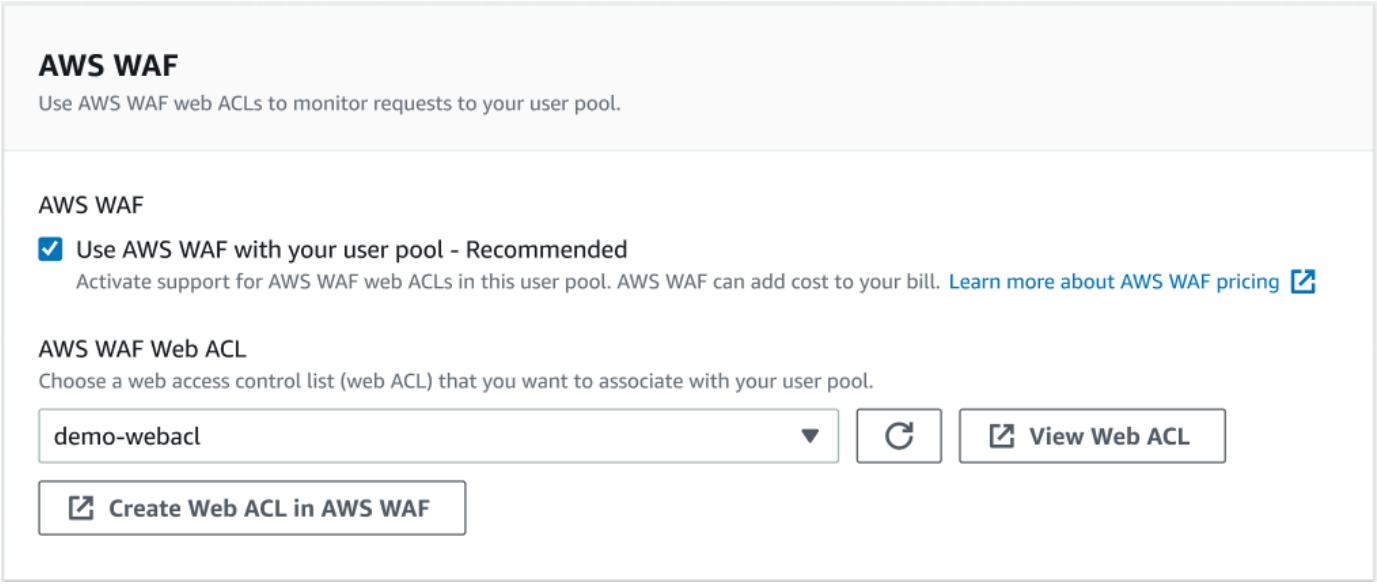
要在您的用户池中使用网页 ACL，您的 AWS Identity and Access Management (IAM) 委托人必须拥有以下 Amazon Cognito 权限。有关 AWS WAF 权限的信息，请参阅《AWS WAF 开发者指南》中的 AWS WAF API 权限。

- `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。

激活 AWS WAF 您的用户池并关联 Web ACL

1. 登录 Amazon Cognito 控制台。
2. 在导航窗格中，选择 用户池，然后选择要编辑的用户池。
3. 选择用户池属性选项卡。
4. 选择 AWS WAF 旁边的编辑。
5. 在下方 AWS WAF，选择 AWS WAF 与您的用户池一起使用。



6. 选择您已经创建的AWS WAF Web ACL，或者选择在中创建 Web ACL AWS WAF，在中的新 AWS WAF 会话中创建一个 Web ACL AWS Management Console。
7. 选择保存更改。

要以编程方式将 Web ACL 与 AWS Command Line Interface 或 SDK 中的用户池关联，请使用 AWS WAF API AssociateWeb中的 ACL。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 C ou nt 时， AWS WAF 会将该请求添加到与该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 开发者指南》中的测试和调整 AWS WAF 保护措施。

您还可以配置 AWS WAF 为将请求标头记录到亚马逊 CloudWatch 日志组、亚马逊简单存储服务 (Amazon S3) 存储桶或亚马逊数据 Firehose。您可以通过 `x-amzn-cognito-client-id` 和 `x-amzn-cognito-operation-name` 识别您使用用户池 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。的安全功能 AWS WAF 补充了 Amazon Cognito 的高级安全功能。您可以在用户池中激活这两项功能。 AWS WAF 单独开具账单，用于检查用户池请求。有关更多信息，请参阅AWS WAF 定价。

日志 AWS WAF 请求数据需要按您定位日志的服务收取额外费用。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，您在中创建的 Amazon Cognito 用户池不区分大小写。 AWS Management Console 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。当用户池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 的更多信息 IdPs，请参阅使用 SP 启动的 SAML 登录。

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（例如）创建资源 CreateUserPool，则必须将布尔`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 或 DescribeUserPoolAPI 操作的 “登录体验” 选项卡来查看账户中每个用户池的区分大小写设置。

迁移到新的用户池

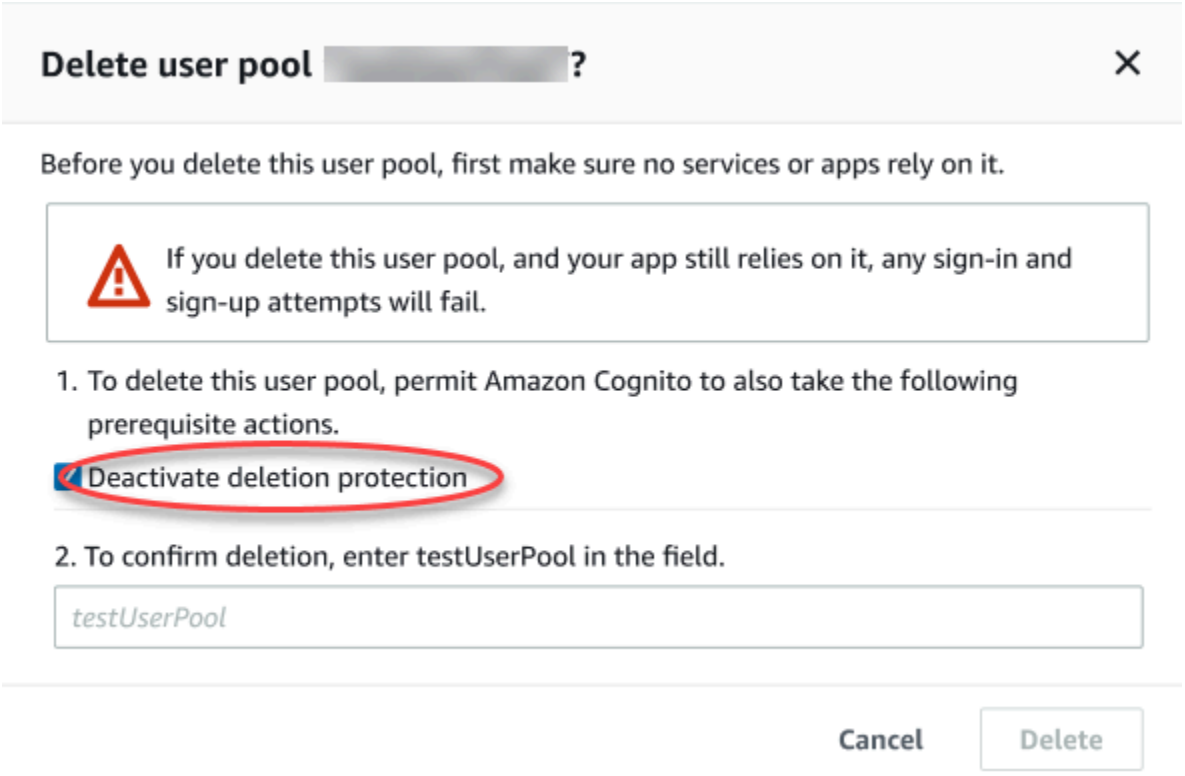
由于用户配置文件之间存在潜在冲突，您无法将 Amazon Cognito 用户池从区分大小写更改为不区分大小写。相反，请将用户迁移到新的用户池。您必须构建迁移代码才能解决与大小写相关的冲突。此代码必须返回唯一的新用户，或者在检测到冲突时拒绝登录尝试。在新的不区分大小写的用户池中，分配一个 迁移用户 Lambda 触发器。该 AWS Lambda 函数可以在新的不区分大小写的用户池中创建用户。当用户使用不区分大小写的用户池登录失败时，Lambda 函数会从区分大小写的用户池中查找并复制该用户。您还可以针对事件激活迁移用户 Lambda 触发

器。ForgotPasswordAmazon 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池中创建新用户时，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池中是唯一的，但在区分大小写的用户池中则相同。

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，请参阅博客中的将用户迁移到 Amazon Cognito 用户池。 AWS

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。

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。当您使用 Amazon Cognito 用户池 API 创建新的应用程序客户端时，默认PreventUserExistenceErrors处于禁用LEGACY状态或禁用状态。在 Amazon Cognito 控制台中，“防止用户存在错误” 选项（设置ENABLED为PreventUserExistenceErrors）默认处于选中状态。要更新您的PreventUserExistenceErrors配置，请执行以下任一操作：

- 在 UpdateUserPoolClientAPI 请求中更改介LEGACY于ENABLED和PreventUserExistenceErrors之间的值。
- 在 Amazon Cognito 控制台中编辑您的应用程序客户端，并在选定 (ENABLED) 和取消选择 () 之间更改防止用户存在错误的状态。LEGACY

当此属性的值为时LEGACY，当用户尝试使用您的用户池中不存在的用户名登录时，您的应用程序客户端会返回UserNotFoundException错误响应。

当此属性的值为时ENABLED，您的应用程序客户端不会以UserNotFoundException错误的方式披露您的用户池中不存在用户帐号。的PreventUserExistenceErrors配置ENABLED具有以下效果：

- Amazon Cognito 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用户。

- Amazon Cognito 登录和忘记密码 API 会返回通用的身份验证失败响应。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 会返回一个响应，表示代码已发送到模拟交付媒体，而不是部分显示用户的联系信息。

以下信息详细说明了设置为时`PreventUserExistenceErrors`用户池操作的行为`ENABLED`。

身份验证和用户创建操作

您可以在 “用户名-密码” 和 “安全远程密码 (SR P)” 身份验证中配置错误响应。您还可以对使用自定义身份验证返回的错误进行自定义。以下 API 执行这些身份验证操作：

- `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。作为回应，Amazon Cognito 会为`SRP_B`用户退货并加盐。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

当用户名已被UsernameExistsException使用时，该SignUp操作始终会返回。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，请参阅自定义登录属性。

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。

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。

   AWS CLI 命令示例

```
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。

   AWS CLI 命令示例

```
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。

   AWS CLI 命令示例

```
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。

   AWS CLI 命令示例

   ```
   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`。

Amazon Cognito 身份池

Amazon Cognito 身份池是联合身份的目录，您可以用它交换 AWS 凭证。身份池会为您的应用程序的用户生成临时 AWS 证书，无论他们已登录还是您尚未识别他们的身份。通过 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。

Amazon Cognito 身份池的功能

签署请求 AWS 服务

将 API 请求签名给亚马逊简单存储服务 (Amazon S3) Servic AWS 服务 e 和亚马逊 DynamoDB。使用亚马逊 Pinpoint 和亚马逊等服务分析用户活动。 CloudWatch

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 ID 或访问令牌、用户池令牌、SAML 断言或社交提供商 OAuth 令牌交换凭证。 AWS

验证您自己的身份

执行您自己的用户验证，并使用您的开发者 AWS 证书为您的用户颁发证书。

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）。当你这样做时，你的用户可以通过你的用户池进行身

份验证 IdPs，将他们的声明整合到一个普通的 OIDC 身份令牌中，然后用该令牌兑换证书。 AWS 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 SAML、OAuth 和 OIDC 提供商的用户声明自定义为短期证书的 API 请求。AssumeRoleWithWebIdentity

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：将 Login with Amazon 设置为身份池 IdP、将 Facebook 设置为身份池 IdP、将 Google 设置为身份池 IdP、将 “通过 Apple 登录” 设置为身份池 IdP、Twitter。
- Amazon Cognito 用户池
- 将 OIDC 提供商设置为身份池 IdP
- 将 SAML 提供商设置为身份池 IdP
- 经开发人员验证的身份（身份池）

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

- 使用身份池（联合身份）
- 身份池概念
- Amazon Cognito 身份池的安全最佳实践
- 将属性用于访问控制
- 使用基于角色的访问控制
- 获取凭证
- 访问 AWS 服务
- 身份池外部身份提供商
- 经开发人员验证的身份（身份池）
- 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）

使用身份池（联合身份）

Amazon Cognito 身份池为访客用户（未经身份验证）和已通过身份验证并收到令牌的用户提供临时 AWS 证书。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。

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 角色中已有要使用的角色，则可以选择使用现有 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。
5. 在 Connect 身份提供商中，输入您在配置身份池信任中选择的身份提供商 (IdPs) 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。
      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。启用基本流程后，您可以绕过为自己选择的角色 IdPs ，AssumeRoleWithWebIdentity直接致电。有关更多信息，请参阅 身份池（联合身份）身份验证流程。
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。

用户 IAM 角色

IAM 角色定义用户访问 AWS 资源的权限，例如Amazon Cognito Sync。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。

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 身份池外部身份提供商。
- 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）。

激活或停用访客访问权限

Amazon Cognito 身份池访客访问（未经身份验证的身份）为未向身份提供者进行身份验证的用户提供唯一标识符和 AWS 证书。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，请参阅开始使用 Amazon Cognito 身份池。

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 角色中已有要使用的角色，则可以选择使用现有 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。
            C. 选择保存更改。
            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。

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。此角色附有策略，规定 AWS 服务 该角色可以访问哪个角色。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。通过修改策略或为身份类型分配不同的角色，您可以控制哪些 AWS 服务身份类型可以访问。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。

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 角色中已有要使用的角色，则可以选择使用现有 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。
5. 选择保存更改。

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。

> **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 身份池外部身份提供商。

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。
     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. 选择保存更改。

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。

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。

> **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。

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。

将 Amazon Cognito Sync 与身份池一起使用

Amazon Cognito Sync 是一个 AWS 服务 和客户端库，它使跨设备同步与应用程序相关的用户数据成为可能。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。

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。选择确认。重复此操作以添加更多记录。

删除与身份关联的数据集

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。

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。

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。

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 角色中已有要使用的角色，则可以选择使用现有 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。
8. 选择保存更改。

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。

设置 Amazon Cognito Events

Amazon Cognito Events 允许您运行 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。

要了解更多信息 AWS Lambda，请参阅AWS Lambda。

身份池概念

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。有了身份，您就可以获得临时的、权限有限的 AWS 凭证来访问其他。 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
- IAM 角色
- 角色信任和权限

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。Amazon Cognito 还会向您的应用程序提供临时的、权限有限的凭证以访问资源。 AWS 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。

完成其中一个流程后，您可以访问由您的角色访问策略定义的其他 AWS 服务 流程。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。

身份池接受来自提供商的以下工件：

提供商	身份验证神器
Amazon Cognito 用户池	ID 令牌

提供商	身份验证神器
OpenID Connect (OIDC)	ID 令牌
SAML 2.0	SAML 断言
社交提供者	访问令牌

增强型（简化的）身份验证流程

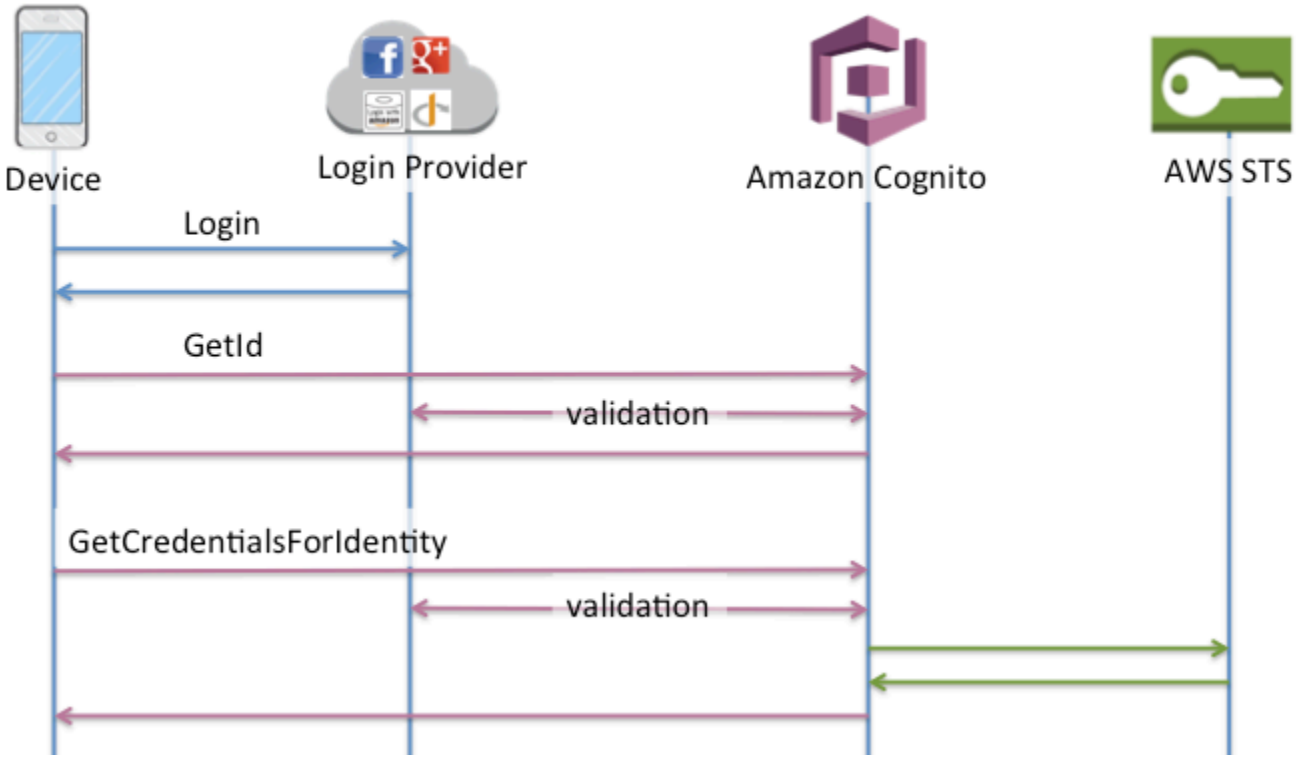
当您使用增强版身份验证流程时，您的应用程序会首先在请求中提供来自授权的 Amazon Cognito 用户池或第三方身份提供商的身份验证证明。GetId

1. 您的应用程序在 GetID 请求中提供来自授权的 Amazon Cognito 用户池或第三方身份提供商的身份验证证明（JSON 网络令牌或 SAML 断言）。
2. 您的身份池会返回一个身份 ID。
3. 您的应用程序在GetCredentialsForIdentity请求中将身份 ID 与相同的身份验证证明相结合。
4. 您的身份池会返回 AWS 证书。
5. 您的应用程序使用临时证书签署 AWS API 请求。

增强型身份验证在您的身份池配置中管理 IAM 角色选择和证书检索的逻辑。您可以将身份池配置为选择默认角色，将基于属性的访问控制 (ABAC) 或基于角色的访问控制 (RBAC) 原则应用于角色选择。来自增强身份验证的 AWS 凭证有效期为一小时。

增强型身份验证中的操作顺序

1. `GetId`
2. `GetCredentialsForIdentity`



基本（经典）工作流程

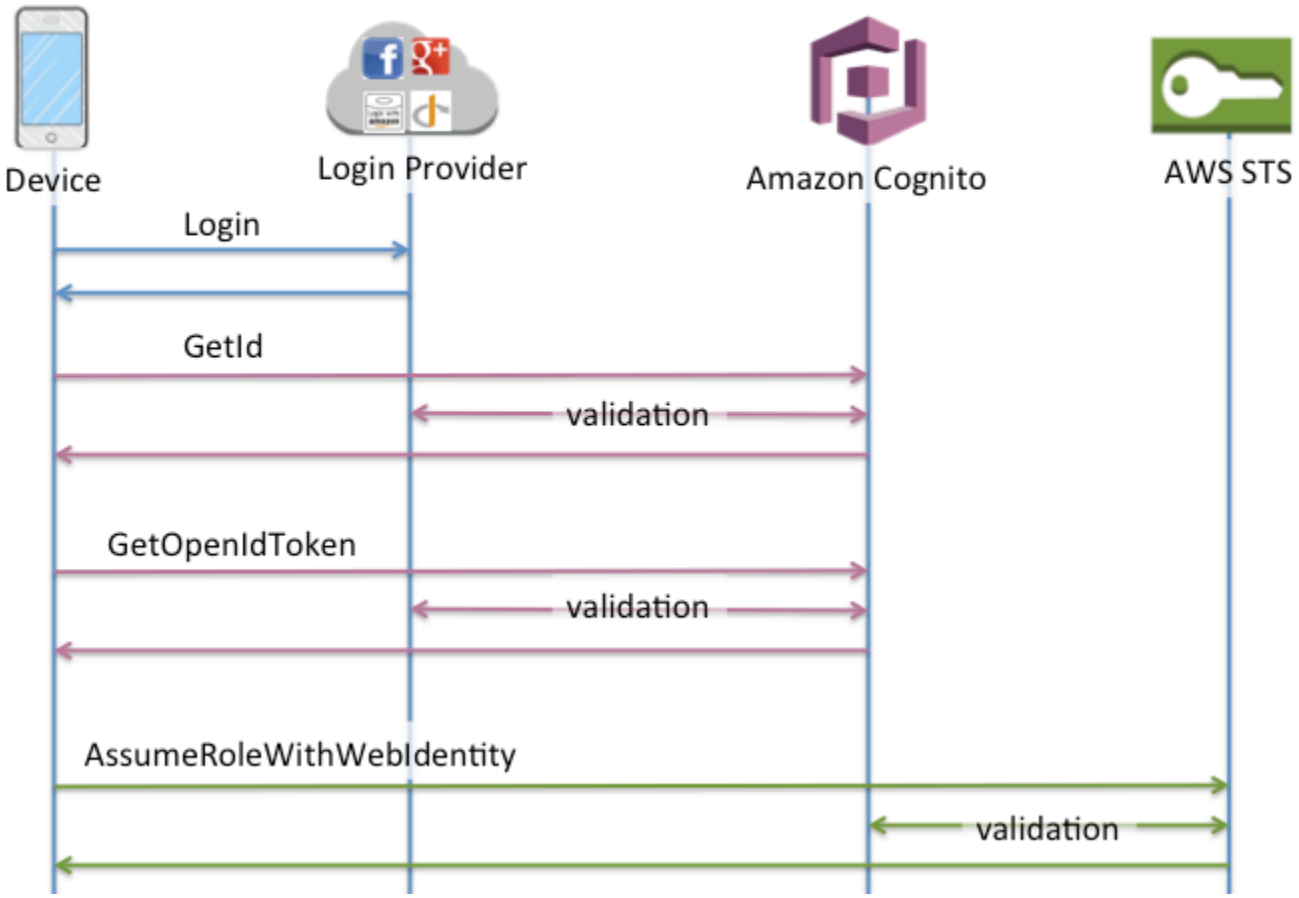
当你使用基本的身份验证流程时，

1. 您的应用程序在 GetID 请求中提供来自授权的 Amazon Cognito 用户池或第三方身份提供商的身份验证证明（JSON 网络令牌或 SAML 断言）。
2. 您的身份池会返回一个身份 ID。
3. 您的应用程序在GetOpenIdToken请求中将身份 ID 与相同的身份验证证明相结合。
4. `GetOpenIdToken`返回由您的身份池颁发的新 OAuth 2.0 令牌。
5. 您的应用程序在AssumeRoleWithWebIdentity请求中显示新令牌。
6. AWS Security Token Service AWS STS) 返回 AWS 凭证。
7. 您的应用程序使用临时证书签署 AWS API 请求。

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。经典工作流程中的`AssumeRoleWithWebIdentity`请求使您的应用程序能够更好地请求您配置了足够信任策略的任何 AWS Identity and Access Management 角色的凭证。您也可以请求自定义角色会话持续时间。

基本身份验证中的操作顺序

1. `GetId`
2. `GetOpenIdToken`
3. `AssumeRoleWithWebIdentity`



已经过开发人员验证的身份验证流程

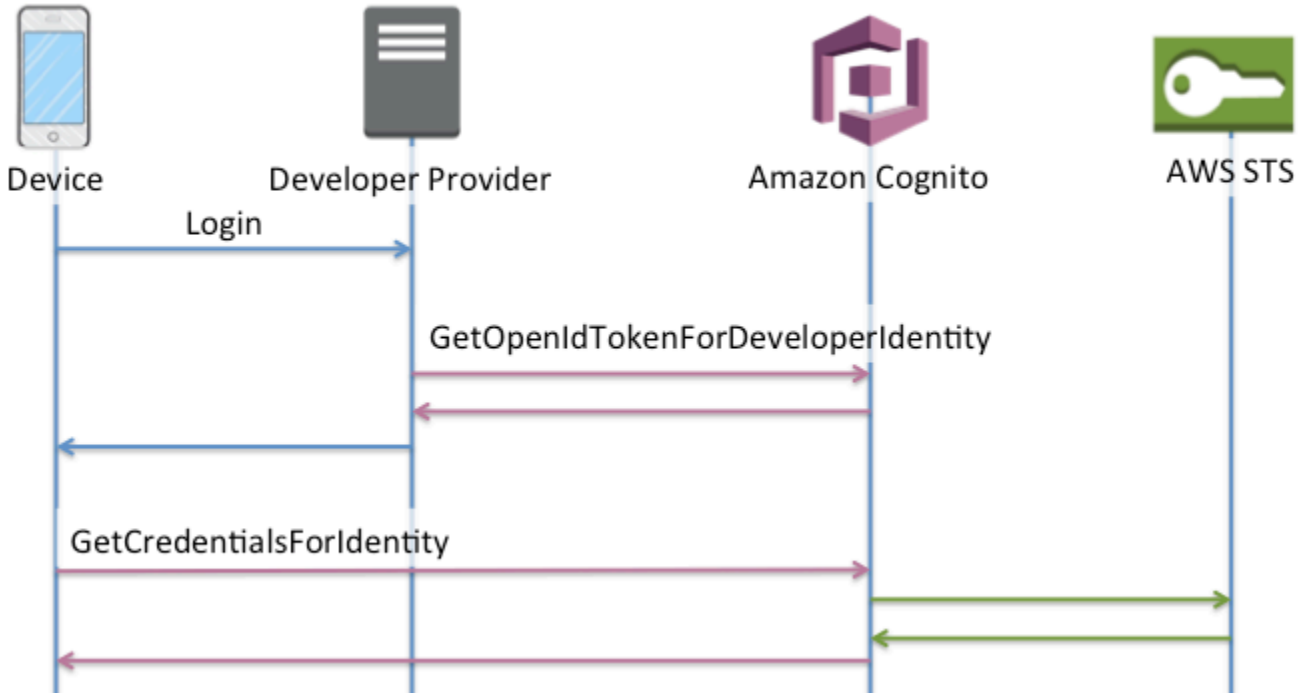
使用经开发人员验证的身份（身份池）时，客户端将使用包括 Amazon Cognito 外部代码的不同身份验证流程，在您自己的身份验证系统中验证用户。Amazon Cognito 外部代码如此处所示。

增强型身份验证流程

开发者提供商的增强身份验证中的操作顺序

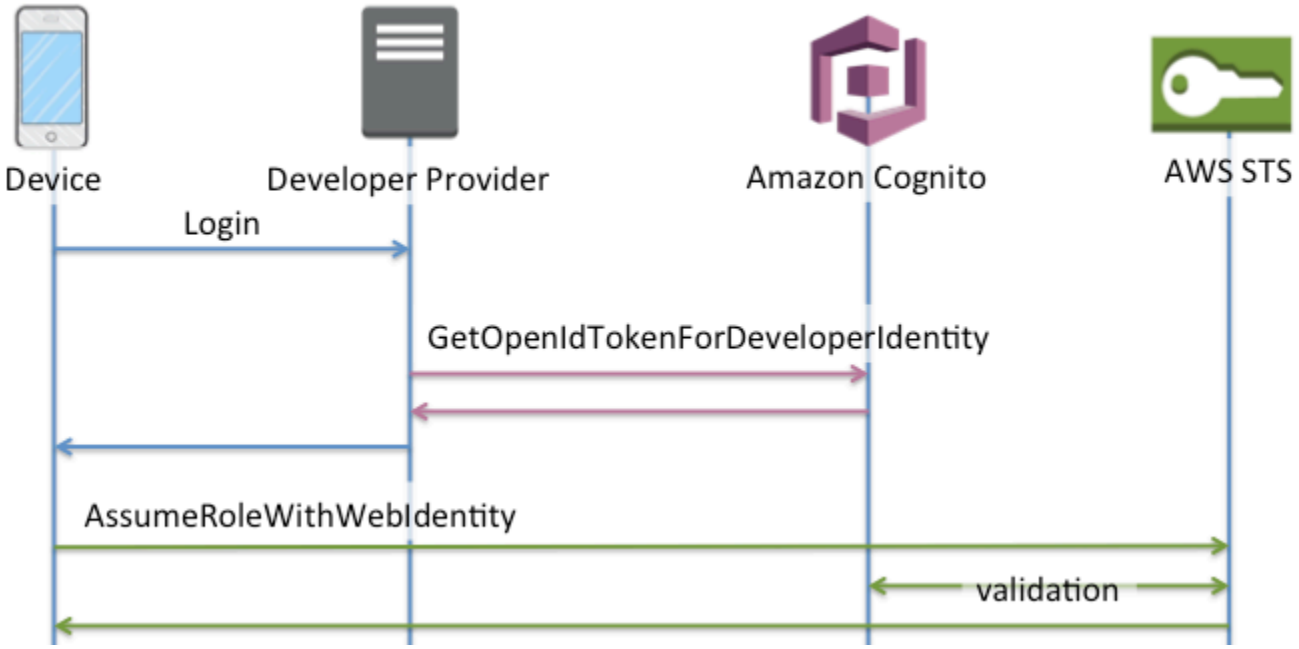
1. 通过开发人员提供商登录（Amazon Cognito 外部代码）
2. 验证用户登录（Amazon Cognito 外部代码）

3. GetOpenIdTokenForDeveloperIdentity
4. GetCredentialsForIdentity



使用开发者提供商进行基本身份验证的操作顺序

1. 在身份池之外实现逻辑以登录并生成开发者-提供商标识符。
2. 检索存储的服务器端 AWS 凭证。
3. 在使用授权 AWS 凭证签名GetOpenIdTokenForDeveloperIdentity的 API 请求中发送开发者提供商标识符。
4. 使用请求应用程序凭证AssumeRoleWithWebIdentity。



我应该使用哪种身份验证流程？

增强的流程是最安全的选择，开发人员的工作量最少：

- 增强的流程降低了 API 请求的复杂性、大小和速率。
- 您的应用程序无需向发出其他 API 请求 AWS STS。
- 您的身份池会评估您的用户应获得的 IAM 角色证书。您无需在客户端中嵌入角色选择逻辑。

> **Important**
>
> 创建新的身份池时，最好不要默认激活基本（经典）身份验证。要实现基本身份验证，请先评估您的 IAM 角色对于 Web 身份的信任关系。然后在您的客户端中构建角色选择逻辑，并保护客户端免受用户修改。

基本身份验证流程将 IAM 角色选择的逻辑委托给您的应用程序。在此流程中，Amazon Cognito 会验证您的用户经过身份验证或未经身份验证的会话，并颁发一个令牌，您可以用该令牌交换证书。 AWS STS用户可以将基本身份验证中的令牌交换为任何信任您的身份池和/或已验证/未经身份验证`amr`状态的 IAM 角色。

同样，请理解开发者身份验证是身份提供者身份验证的捷径。Amazon Cognito 信任授权GetOpenIdTokenForDeveloperIdentity请求的 AWS 凭证，无需对请求内容进行额外验证。保护授权开发者身份验证的密钥不被用户访问。

API 摘要

GetId

GetIdAPI 调用是在 Amazon Cognito 中建立新身份所需的第一个调用。

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 移动版 SDK 和浏览器 JavaScript 中的 S AWS DK 都有凭据提供商，可以为您处理缓存。

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

在您建立身份 ID 之后，即可调用该 GetCredentialsForIdentityAPI。因此，此操作在功能上等同于调用GetOpenIdToken。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 控制台执行此操作，也可以通过操作手动执行此操作。SetIdentityPoolRoles

GetOpenIdToken

在建立身份 ID 后发出 GetOpenIdTokenAPI 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。浏览器中的 AWS 移动 SDK 或 AWS SDK JavaScript 中的提供商会为您执行此操作。

GetOpenIdTokenForDeveloperIdentity

当使用经过开发者身份验证GetId的GetOpenIdToken身份时，该GetOpenIdTokenForDeveloperIdentity操作取代了对设备的使用。由于您的应用程序使用 AWS 凭证对此 API 操作的请求进行签名，因此 Amazon Cognito 相信请求中提供的用户标识符是有效的。开发者身份验证取代 Amazon Cognito 对外部提供商执行的令牌验证。

此 API 的有效载荷包括logins地图。此地图必须包含开发者提供商的密钥和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

对于经过开发者身份验证的身份，Amazon Cognito 支持通过 API 进行隐式合并和显式合并。MergeDeveloperIdentities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

在你获得 OpenID Connect 令牌后，你可以通过 AssumeRoleWithWebIdentityAPI 请求将其换成临时 AWS 证书 AWS Security Token Service ()AWS STS。

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。为您的应用程序设置不同的 IAM 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。Amazon Cognito 控制台可以在您首次设置身份池时创建默认角色。这些角色实际上没有被授予任何权限。修改它们以满足您的需求。

了解有关 角色信任和权限 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 （美国西部）	`https://cognito-identity.us-gov-west-1.amazonaws.com/.well-known/jwks_uri`
中国（北京）	`https://cognito-identity.cn-north-1.amazonaws.com.cn/.well-known/jwks_uri`
欧洲（米兰）和非洲（开普敦）等选择加入区域	`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 角色的工作原理如下：当用户登录您的应用程序时，Amazon Cognito 会为该用户生成临时 AWS 证书。这些临时凭证与特定 IAM 角色相关联。使用 IAM 角色，您可以定义一组访问 AWS 资源的权限。

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。

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

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 的更多信息。

呈现给 AWS STS 的令牌由身份池生成，身份池将用户池、社交或 OIDC 提供商令牌或 SAML 断言转换为自己的令牌。身份池令牌包含一个 aud 声明，即身份池 ID。

以下示例角色信任策略允许联合服务主体cognito-identity.amazonaws.com调用 AWS STS API AssumeRoleWithWebIdentity。仅当 API 请求中的身份池令牌具有以下声明时，请求才会成功。

1. 一个 aud 声明（身份池 ID us-west-2:abcdefg-1234-5678-910a-0e8443553f95）。
2. 一个 amr 声明（authenticated），当用户已登录并且不是访客用户时添加。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "cognito-identity.amazonaws.com"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "cognito-identity.amazonaws.com:aud": "us-
west-2:abcdefg-1234-5678-910a-0e8443553f95"
```

```
                },
                "ForAnyValue:StringLike": {
                    "cognito-identity.amazonaws.com:amr": "authenticated"
                }
            }
        }
    ]
}
```

基本（经典）身份验证中 IAM 角色的信任策略

您必须至少应用一个条件，限制您在身份池中使用的角色的信任策略。当您为身份池创建或更新角色信任策略时，如果您尝试在没有至少一个限制源身份的条件密钥的情况下保存更改，IAM 会返回错误。AWS STS 不允许从身份池到缺少此类条件的 IAM 角色进行跨账户AssumeRoleWithWebIdentity操作。

本主题包括几个限制身份池源身份的条件。有关完整列表，请参阅 AWS Web 联合身份验证的可用密钥。

在使用身份池进行基本身份验证或经典身份验证中， AWS STS 如果任何 IAM 角色具有正确的信任策略，则可以代入该角色。Amazon Cognito 身份池的 IAM 角色信任服务主体 `cognito-identity.amazonaws.com` 担任该角色。此配置不足以保护您的 IAM 角色免受意外访问资源的侵害。此类角色必须对角色信任策略应用附加条件。如果不满足以下至少一个条件，则无法为身份池创建或修改角色。

`cognito-identity.amazonaws.com:aud`

将角色限制为只能从一个或多个身份池执行操作。Amazon Cognito 在身份池令牌中标明了`aud`声明中的源身份池。

`cognito-identity.amazonaws.com:amr`

将角色限制为`authenticated`或`unauthenticated`（访客）用户。Amazon Cognito 会在身份池令牌中的`amr`声明中显示身份验证状态。

`cognito-identity.amazonaws.com:sub`

通过 UUID 将角色限制为一个或多个用户。此 UUID 是用户在身份池中的身份 ID。此值不是来自用户原始身份提供商的`sub`值。Amazon Cognito 在身份池令牌中的`sub`声明中注明了此 UUID。

Enhanced-Flow 身份验证要求 IAM 角色与身份池 AWS 账户 相同，但在基本身份验证中，情况并非如此。

其他注意事项适用于承担跨账户 IAM 角色的 Amazon Cognito 身份池。这些角色的信任策略必须接受cognito-identity.amazonaws.com服务主体，并且必须包含特定cognito-identity.amazonaws.com:aud条件。为防止意外访问您的 AWS 资源，aud条件键在条件值中将角色限制为身份池中的用户。

身份池为身份发放的令牌包含有关身份池来源 AWS 账户 的信息。当您在AssumeRoleWithWebIdentityAPI 请求中提供身份池令牌时， AWS STS 会检查原始身份池是否与 IAM 角色 AWS 账户 相同。如果 AWS STS 确定请求是跨账户的，则会检查角色信任策略是否有aud条件。如果角色信任策略中不存在此类条件，则假设角色调用将失败。如果请求不是跨账户请求，则 AWS STS 不强制执行此限制。作为最佳实践，请务必将此类条件应用于您的身份池角色的信任策略。

其他信托政策条件

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 进行网络联合的 IAM 策略变量的更多信息，请参阅AWS Identity and Access Management 用户指南中的 IAM 和 AWS STS 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 和 AWS 托管会话策略。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。

Amazon Cognito 还可防止经过身份验证和未经身份验证的用户向 Amazon Cognito 身份池和 Amazon Cognito Sync 发出 API 请求。其他人 AWS 服务 可能会限制通过 Web 身份访问服务。

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，其值为在 Amazon 中授予额外权限的AWS 托管策略 CloudWatch。

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 Data Firehose 适用于 Apache Flink 的亚马逊托管服务
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

类别	服务
机器学习	Amazon CodeWhisperer Amazon Comprehend Amazon Lex Amazon Machine Learning Amazon Personalize Amazon Polly Amazon Rekognition 亚马逊 SageMaker [1] Amazon Textract[1] Amazon Transcribe Amazon Translate
管理与治理	Amazon CloudWatch Amazon CloudWatch 日志
联网和内容分发	Amazon API Gateway
安全性、身份与合规性	Amazon Cognito 用户群体
存储	Amazon Simple Storage Service

[1] 对于下表 AWS 服务 中的，内联策略授予操作的子集。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`

AWS 服务	未经身份验证的增强型流程用户的最大权限

	`ReEncrypt`
	
	`GenerateDataKey`

Amazon SageMaker	`InvokeEndpoint`

Amazon Textract	`DetectDocumentText`
	
	`AnalyzeDocument`

Amazon Sumerian	`View*`

要向此列表 AWS 服务 之外的用户授予访问权限，请在您的身份池中激活基本（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

内联会话策略限制您的用户的有效权限，使其无法访问以下列表中 AWS 服务 以外的任何权限。您还必须在应用于用户的 IAM 角色的策略 AWS 服务 中向这些角色授予权限。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 托管会话策略

Amazon Cognito 还通过 AWS 托管式策略 AmazonCognitoUnAuthedIdentitiesSessionPolicy，将未经身份验证的用户的权限范围限制为增强流程中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。

AmazonCognitoUnAuthedIdentitiesSessionPolicy 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

在本部分中，您可以找到示例 Amazon Cognito 访问策略，这些策略仅向您的用户授予完成特定操作所需的最低权限。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。

Amazon Cognito 身份池的安全最佳实践

Amazon Cognito 身份池为您的 AWS 应用程序提供临时证书。 AWS 账户 通常包含应用程序用户所需的资源和私有后端资源。构成 AWS 证书的 IAM 角色和策略可以授予对其中任何资源的访问权限。

身份池配置的主要最佳实践是确保您的应用程序可以在没有过多或意外权限的情况下完成工作。为防止安全配置错误，请在要发布到生产环境的每个应用程序启动之前查看这些建议。

主题

IAM 配置最佳实践

当访客或经过身份验证的用户在您的应用程序中启动需要身份池证书的会话时，您的应用程序会检索 IAM 角色的临时 AWS 证书。这些凭据可能是默认角色、由身份池配置中的规则选择的角色或应用程序选择的自定义角色的证书。分配给每个角色的权限后，您的用户就可以访问您的 AWS 资源。

有关常规 IAM 最佳实践的更多信息，请参阅 AWS Identity and Access Management 用户指南中的 IAM 最佳实践。

在 IAM 角色中使用信任策略条件

IAM 要求身份池的角色至少有一个信任策略条件。例如，此条件可以将角色的作用域设置为仅限经过身份验证的用户。 AWS STS 还要求跨账户基本身份验证请求具有两个特定条件：`cognito-identity.amazonaws.com:aud`和。`cognito-identity.amazonaws.com:amr`作为最佳实践，请在所有信任身份池服务主体的IAM角色中应用这两个条件`cognito-identity.amazonaws.com`。

- cognito-identity.amazonaws.com:aud：身份池令牌中的 a ud 声明必须与可信身份池 ID 相匹配。
- cognito-identity.amazonaws.com:amr：身份池令牌中的 amr 声明必须经过身份验证或未经身份验证。在这种情况下，您只能将角色的访问权限保留给未经身份验证的访客，或者仅向经过身份验证的用户保留访问权限。例如，您可以进一步细化此条件的值，将角色限制为来自特定提供商的用户graph.facebook.com。

以下示例角色信任策略在以下条件下授予对角色的访问权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.amazonaws.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com:aud": "us-east-1:a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com:amr": "authenticated"
        }
      }
    }
  ]
}
```

与身份池相关的元素

- "Federated": "cognito-identity.amazonaws.com"：用户必须来自身份池。
- "cognito-identity.amazonaws.com:aud": "us-east-1:a1b2c3d4-5678-90ab-cdef-example11111"：用户必须来自特定的身份池us-east-1:a1b2c3d4-5678-90ab-cdef-example11111。
- "cognito-identity.amazonaws.com:amr": "authenticated": 必须对用户进行身份验证。访客用户无法担任该角色。

应用最低权限权限

当您使用 IAM 策略为经过身份验证的访问或访客访问设置权限时，请仅授予执行特定任务所需的特定权限或最低权限权限。以下示例 IAM 策略应用于角色时，授予对 Amazon S3 存储桶中单个图像文件的只读访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/assets/my_picture.jpg"]
    }
  ]
}
```

身份池配置最佳实践

身份池为生成 AWS 凭证提供了灵活的选项。当您的应用程序可以使用最安全的方法时，不要使用设计捷径。

了解访客访问的影响

未经身份验证的访客访问权限允许用户在登录 AWS 账户 之前从您那里检索数据。任何知道您的身份池 ID 的人都可以申请未经身份验证的证书。您的身份池 ID 不是机密信息。激活访客访问 AWS 权限后，所有人均可使用您授予未经身份验证的会话的权限。

最佳做法是，停用访客访问权限，仅在用户进行身份验证后才获取所需的资源。如果您的应用程序需要在登录前访问资源，请采取以下预防措施。

- 熟悉对未经身份验证的角色设置的自动限制。
- 监控和调整未经身份验证的 IAM 角色的权限，以满足应用程序的特定需求。
- 授予对特定资源的访问权限。
- 保护您的默认未经身份验证的 IAM 角色的信任策略。
- 只有当您确信自己会将 IAM 角色中的权限授予互联网上的任何人时，才能激活访客访问权限。

默认使用增强型身份验证

通过基本（经典）身份验证，Amazon Cognito 会将选择的 IAM 角色委托给您的应用程序。相比之下，增强型流程使用身份池中的集中式逻辑来确定 IAM 角色。它还通过设置了 IAM 权限上限的范围缩小策略，为未经身份验证的身份提供了额外的安全性。增强的流程是最安全的选择，开发人员的工作量最少。要了解有关这些选项的更多信息，请参阅身份池（联合身份）身份验证流程。

基本流程可以公开用于角色选择和组装 AWS STS API 凭证请求的客户端逻辑。增强的流程将逻辑和代理角色请求隐藏在身份池自动化背后。

配置基本身份验证时，请将 IAM 最佳实践应用于您的 IAM 角色及其权限。

安全地使用开发者提供商

经过开发人员身份验证的身份是服务器端应用程序身份池的一项功能。身份池开发者身份验证所需的唯一身份验证证据是身份池开发者的 AWS 证书。身份池不会对您在此身份验证流程中提供的开发者-提供商标识符的有效性施加任何限制。

作为最佳实践，仅在以下条件下实施开发者提供商：

- 要为使用经过开发者身份验证的证书创建问责制，请设计您的开发者提供商名称和标识符以指明身份验证来源。例如："Logins" : {"MyCorp provider" : "*[provider application ID]*"}。
- 避免使用长期存在的用户凭证。将您的服务器端客户端配置为使用服务相关角色请求身份，例如 EC2 实例配置文件和 Lambda 执行角色。
- 避免在同一个身份池中混合使用内部和外部信任来源。将您的开发者提供商和单点登录 (SSO) 提供商添加到单独的身份池中。

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
- 示例：将属性用于访问控制策略
- 关闭访问控制属性（控制台）
- 默认提供商映射

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
- 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。

您必须准备要将其凭证传递给用户的 IAM 角色。这些角色的信任策略必须允许 Amazon Cognito 为您的用户代入该角色。对于用于访问控制的属性，您还必须允许 Amazon Cognito 将主体标签应用于用户的临时会话。授予使用操作 AssumeRoleWithWebIdentity 代入该角色的权限。授予使用仅限权限操作 `sts:TagSession` 标记用户会话的权限。有关更多信息，请参阅《AWS Identity and Access Management 用户指南》中的在 AWS Security Token Service 中传递会话标签。有关向 Amazon Cognito 服务主体 `cognito-identity.amazonaws.com` 授予 `sts:AssumeRoleWithWebIdentity` 和 `sts:TagSession` 权限的示例信任策略，请参阅示例：将属性用于访问控制策略。

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. 选择保存更改。

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。

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"，"57e7b692-4f66-480d-98b8-45a6729b4c88"
Facebook	访问令牌	aud（app_id），sub（user_id）	“492844718097981”，“112177216992379”
Google	ID 令牌	aud（客户端 ID）和 sub（用户 ID）	“620493171733-eebk7c0hcp5lj3e1tlqp1gntt3k0rncv.apps.googleusercontent.com”，“109220063452404746097”
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"，"user123@gmail.com"
Apple	ID 令牌	aud（客户端 ID）和 sub（用户 ID）	“com.amazonaws.ec2-54-80-172-243.compute-1.client”，“00

Provider	令牌类型	委托人标签值	示例
			1968.a6ca34e9c1e74 2458a26cf8005854be 9.0733”
Amazon	访问令牌	aud（Amzn Dev Ac 上的客户端 ID），user_id（用户 ID）	“amzn1.application -oa2-client.9d70d9 382d34461 08aaee3dd763a0fa6” ，“amzn1.account.AG HNIFJQMFS BG3G6XCPV B35ORQAA”
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- eebk7c0hcp5lj 3e1tlqp1gntt3k0rnc v.apps.googleuserc ontent.com”，“10922 0063452404746097”
Twitter	访问令牌	aud（应用程序 ID；应用程序密钥 ），sub（用户 ID）	"DfwifTtKEX1FiIBRn OTIR0CFK; Xgj5xb8xIrIVCPjXgL IdkW7fXmw cJJrFvnoK9gwZkLexo 1y5z1"，"1269003884 292222976"
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 身份池为您的经过身份验证的用户分配一组临时的、权限有限的凭证，以访问您的资源。 AWS 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：

- 如果GetCredentialsForIdentityCustomRoleArn参数已设置并且与cognito:roles声明中的角色匹配，则使用该参数。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。
- 如果未设置cognito:preferred_role声明，则cognito:roles声明已设置，CustomRoleArn且未在调用中指定GetCredentialsForIdentity，则使用控制台中的角色解析设置或AmbiguousRoleResolution字段（在 SetIdentityPoolRolesAPI 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。

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RoleMapping类型字段中没有匹配规则时要分配的角色，该AmbiguousRoleResolution字段在 SetIdentityPoolRolesAPI 的RoleMappings参数中指定。

您可以使用类型的字段在 AWS CLI 或 API 中为 OpenID Connect (OIDC) 和 SAML 身份提供者设置基于规则的映射。RulesConfiguration RoleMapping您可以在 SetIdentityPoolRolesAPI 的RoleMappings参数中指定此字段。 AWS Management Console 目前不允许您为 OIDC 或 SAML 提供商添加规则。

例如，以下 AWS CLI 命令添加了一条规则，该规则将角色分配arn:aws:iam::123456789012:role/Sacramento_team_S3_admin给萨克拉曼多所在地经过 OIDC IdP 身份验证的用户：arn:aws:iam::123456789012:oidc-provider/myOIDCIdP

```
aws cognito-identity set-identity-pool-roles --region us-east-1 --cli-input-json
 file://role-mapping.json
```

role-mapping.json 的内容：

```
{
    "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角色解析设置和 SetIdentityPoolRolesAPI 的`RoleMappings`参数来指定无法通过令牌确定正确的角色时的默认行为。

获取凭证

您可以使用 Amazon Cognito 为您的应用程序提供临时的、权限有限的证书，以便您的用户可以访问资源。 AWS 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。使用 AWS Security Token Service (AWS STS) 中的 Web 联合身份验证支持将它们交换为凭证。建议使用 `AWS.CognitoIdentityCredentials` 来为您的应用程序用户获得 AWS 凭证。然后使用将凭证对象中的身份交换为证书 AWS STS。

> **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。要在本地`config`文件中为 AWS SDK 或设置身份池令牌 AWS CLI，请添加`web_identity_token_file`个人资料条目。请参阅《软件开发工具包和 AWS 工具参考指南》中的代入角色凭证提供商。

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》。为了获得最佳效果，请使用内置的身份池集成开始您的项目 AWS Amplify。

AWS 用于通过身份池获取和设置凭证的 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）
- 开发者指南中@@ 使用亚马逊 Cognito 身份对用户进行 AWS SDK for JavaScript 身份验证
- 开发者指南中的@@ 亚马逊 Cognito 凭证提供商 AWS SDK for .NET
- 在《 AWS SDK for Go 开发者指南》中@@ 以编程方式指定凭证
- 在《 AWS SDK for Java 2.x 开发人员指南》的代码中提供临时证书
- assumeRoleWithWebIdentityCredentialProvider AWS SDK for PHP 开发者指南中的提供商
- AWS SDK for Python (Boto3) 文档中的代入 Web 身份提供者的角色
- 在 AWS SDK for Rust 开发者指南中@@ 指定您的凭证和默认区域

以下各节提供了一些旧版 AWS SDK 中的示例代码。

Android

您可以使用 Amazon Cognito 为您的应用程序提供临时的、权限有限的证书，以便您的用户可以访问资源。 AWS Amazon Cognito 同时支持经过身份验证和未经身份验证的身份。要为您的应用程序提供 AWS 凭证，请按照以下步骤操作。

要在安卓应用程序中使用 Amazon Cognito 身份池，请进行设置。 AWS Amplify有关更多信息，请参阅《Amplify 开发中心》中的身份验证。

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()`。从 Android 3.0（API 级别 11）开始，NetworkOnMainThreadException如果您在主应用程序线程上执行网络 I/O，您的应用程序将自动失败并抛出。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 为您的应用程序提供临时的、权限有限的证书，以便您的用户可以访问资源。 AWS 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。要为您的应用程序提供 AWS 凭证，请完成以下步骤。

要在 iOS 应用程序中使用 Amazon Cognito 身份池，请进行设置。 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。

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 为您的应用程序提供临时的、有限权限的证书，以便您的用户可以访问资源。 AWS Amazon Cognito 同时支持经过身份验证和未经身份验证的身份。要为您的应用程序提供 AWS 凭证，请按照以下步骤操作。

要在 iOS 应用程序中使用 Amazon Cognito 身份池，请进行设置。 AWS Amplify有关更多信息，请参阅《Amplify 开发中心》中的 Swift 身份验证。

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 为您的应用程序提供临时的、权限有限的证书，以便您的用户可以访问资源。 AWS Amazon Cognito 同时支持经过身份验证和未经身份验证的身份。要为您的应用程序提供 AWS 凭证，请按照以下步骤操作。

适用于 Unity 的AWS SDK 现在是 AWS SDK for .NET 的一部分。要开始使用中的亚马逊 Cognito AWS SDK for .NET，请参阅开发者指南中的亚马逊 Cognito 凭证提供商。 AWS SDK for .NET 或者，请参阅 Amplify 开发者中心，了解用于构建应用程序的选项。 AWS Amplify

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 为您的应用程序提供临时的、有限权限的证书，以便您的用户可以访问资源。 AWS Amazon Cognito 同时支持经过身份验证和未经身份验证的身份。要为您的应用程序提供 AWS 凭证，请按照以下步骤操作。

适用于 Xamarin 的AWS SDK 现在是 AWS SDK for .NET 的一部分。要开始使用中的亚马逊 Cognito AWS SDK for .NET，请参阅开发者指南中的亚马逊 Cognito 凭证提供商。 AWS SDK for .NET 或者，请参阅 Amplify 开发者中心，了解用于构建应用程序的选项。 AWS Amplify

> **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

在配置您的 Amazon Cognito 凭证提供程序并检索 AWS 证书后，您可以创建客户端 AWS 服务 。

AWS 用于创建客户端的 SDK 资源

- AWS 《 AWS SDK for C++ 开发人员指南》中的@@ 客户端配置
- 将 AWS SDK for Go V2 与《 AWS SDK for Go 开发人员指南》 AWS 服务中的搭配使用
- 在《 AWS SDK for Java 2.x 开发人员指南》中@@ 配置 HTTP 客户端
- 在《 AWS SDK for JavaScript 开发者指南》中@@ 创建和调用服务对象
- 在 AWS SDK for Python (Boto3) 文档中@@ 创建客户端
- 在《 AWS SDK for Rust 开发者指南》中@@ 创建服务客户端
- 在《 AWS SDK for Swift 开发者指南》中@@ 使用客户端

下面的代码段初始化 Amazon DynamoDB 客户端：

Android

要在安卓应用程序中使用 Amazon Cognito 身份池，请进行设置。 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);
```

凭证提供者与 Amazon Cognito 通信，检索经过身份验证和未经身份验证的用户的唯一标识符，以及移动 SDK 的临时有限 AWS 权限证书。 AWS 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，请进行设置。 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];
```

凭证提供者与 Amazon Cognito 通信，检索经过身份验证和未经身份验证的用户的唯一标识符，以及移动 SDK 的临时有限 AWS 权限证书。 AWS 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，请进行设置。 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")
```

凭证提供者与 Amazon Cognito 通信，检索经过身份验证和未经身份验证的用户的唯一标识符，以及移动 SDK 的临时有限 AWS 权限证书。 AWS 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'});
```

凭证提供者与 Amazon Cognito 通信，检索经过身份验证和未经身份验证的用户的唯一标识符，以及移动 SDK 的临时 AWS 有限权限证书。 AWS 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。要开始使用中的亚马逊 Cognito AWS SDK for .NET，请参阅开发者指南中的亚马逊 Cognito 凭证提供商。 AWS SDK for .NET 或者，请参阅 Amplify 开发者中心，了解用于构建应用程序的选项。 AWS Amplify

```
// create a service client that uses credentials provided by Cognito
```

```
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials, REGION);
```

凭证提供者与 Amazon Cognito 通信，检索经过身份验证和未经身份验证的用户的唯一标识符，以及移动 SDK 的临时 AWS 有限权限证书。 AWS 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。要开始使用中的亚马逊 Cognito AWS SDK for .NET，请参阅开发者指南中的亚马逊 Cognito 凭证提供商。 AWS SDK for .NET 或者，请参阅 Amplify 开发者中心，了解用于构建应用程序的选项。 AWS Amplify

```
// create a service client that uses credentials provided by Cognito
var client = new AmazonDynamoDBClient(credentials, REGION)
```

凭证提供者与 Amazon Cognito 通信，检索经过身份验证和未经身份验证的用户的唯一标识符，以及移动 SDK 的临时 AWS 有限权限证书。 AWS 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，您可以将一个身份池与多个身份池相关联 IdPs。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。

以下说明将指导您使用 Amazon Cognito 身份池支持的身份池进行身份验证。 IdPs

主题

- 将 Facebook 设置为身份池 IdP
- 将 Login with Amazon 设置为身份池 IdP
- 将 Google 设置为身份池 IdP
- 将 “通过 Apple 登录” 设置为身份池 IdP
- 将 OIDC 提供商设置为身份池 IdP
- 将 SAML 提供商设置为身份池 IdP

将 Facebook 设置为身份池 IdP

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 网络应用程序与 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。

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，并在需要时授予用户访问其他 AWS 资源的权限。

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 会话对象包含一个 OAuth 令牌，Amazon Cognito 使用该令牌为经过身份验证的最终用户 AWS 生成证书。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`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initWithRegion类型: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initWithRegion类型: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，并在需要时授予用户访问其他 AWS 资源的权限。

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 软件开发工具包会获取 OAuth 令牌，亚马逊 Cognito 使用该令牌为经过身份验证的最终用户生成 AWS 证书。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 “用例 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
    }
  });
}
```

将 Login with Amazon 设置为身份池 IdP

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，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）。在您身份池中的“通过 Amazon 登录”IdP 的应用程序 ID 字段中输入安全配置文件 ID。

> **Note**
>
> 在您身份池中的“通过 Amazon 登录”IdP 的应用程序 ID 字段中输入安全配置文件 ID。这与使用客户端 ID 的用户群体不同。

在 Amazon Cognito 控制台中配置外部提供商

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. 选择保存更改。

使用 Login with Amazon：Android

对亚马逊登录进行身份验证后，您可以通过界面的 onSuccess 方法将令牌传递给 Amazon Cognito 凭证提供商。 TokenListener 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

对亚马逊登录进行身份验证后，您可以使用 AMZN 的 requestDidSucceed 方法将令牌传递给 Amazon Cognito 凭证提供商：AccessTokenDelegate

```
- (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());
    }
}
```

将 Google 设置为身份池 IdP

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。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。

如需详细了解如何将 Google 集成到您的安卓应用中，请参阅 Google 身份文档中的使用 Google 登录对用户进行身份验证。

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。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。

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。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。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 网络应用程序启用 Google 登录，请为您的应用程序创建 Google 开发者控制台项目。

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

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。
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

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。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。

将 “通过 Apple 登录” 设置为身份池 IdP

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）。有关设置 “使用 Apple 登录” 的 up-to-date 更多信息，请参阅 Apple 开发者文档中的配置环境以使用 Apple 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。

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 使用此会话对象中的 ID 令牌对用户进行身份验证，生成唯一标识符，并在需要时授予用户访问其他 AWS 资源的权限。

```
@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，并在需要时授予用户访问其他 AWS 资源的权限。

```
(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，并在需要时授予用户访问其他 AWS 资源的权限。

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 “通过 Apple 登录”： JavaScript

Apple 不提供支持 “用苹果登录” 功能的 SDK JavaScript。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。
- 要在 JavaScript 用户界面中添加 “使用 Apple 登录” 按钮，请按照 Apple 文档中的 “显示和配置使用 Apple 按钮登录” 进行操作。
- 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 使用此会话对象中的 ID 令牌对用户进行身份验证，生成唯一标识符，并在需要时授予用户访问其他 AWS 资源的权限。

```
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 使用此会话对象中的 ID 令牌对用户进行身份验证，生成唯一标识符，并在需要时授予用户访问其他 AWS 资源的权限。

拥有令牌后，您可以在 `CognitoAWSCredentials` 中对其进行设置：

```
credentials.AddLogin("appleid.apple.com", token);
```

将 OIDC 提供商设置为身份池 IdP

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

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 IAM 中选择一个 OIDC 身份提供商 IdPs 。 AWS 账户如果您想添加新的 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. 选择保存更改。

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")
    }
}
```

实例化时AWSCognitoCredentialsProvider，在构造函数中将实现的类AWSIdentityProviderManager作为 identityProviderManager的值传递。有关更多信息，请转到AWSCognitoCredentialsProvider参考页面并选择initWithRegion类型:identityPoolId:identityProviderManager。

JavaScript

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
 IdentityPoolId: 'IDENTITY_POOL_ID',
 Logins: {
    'login.provider.com': token
 }
});
```

Unity

```
credentials.AddLogin("login.provider.com", token);
```

Xamarin

```
credentials.AddLogin("login.provider.com", token);
```

将 SAML 提供商设置为身份池 IdP

Amazon Cognito 支持通过安全断言标记语言 2.0 (SAML 2.0IdPs) 对身份提供商 () 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。

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 AWS 账户的 IAM 中选择一个 SAML 身份提供商 IdPs 。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. 选择保存更改。

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。使用 man IdPs y，您可以指定一个 URL，IdP 可以使用该网址从 XML 文档中读取信赖方信息和证书。对于 AWS，你可以使用 https://signin.aws.amazon.com/static/saml-metadata.xml。下一步是配置来自 IdP 的 SAML 断言响应，以填充所需的声明。 AWS 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。

如果在 SAML 元数据中，您的 SAML IdP 包含多个签名证书，则在登录时，只要与 SAML 元数据中的任何证书匹配，您的用户群体就会确定 SAML 断言有效。

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。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 联合，请确定用户启动登录的 URL。 AWS 联盟使用 IDP 发起的登录。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地图中接受 SAML 断言的 API 操作包括GetId、GetCredentialsForIdentityGetOpenIdToken、和 ID。GetOpenTokenForDeveloperIdentity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 将 Facebook 设置为身份池 IdP、将 Google 设置为身份池 IdP、将 Login with Amazon 设置为身份池 IdP 和 将 “通过 Apple 登录” 设置为身份池 IdP 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。使用经过开发人员身份验证的身份，您可以通过自己的现有身份验证流程注册和验证用户，同时仍可以使用 Amazon Cognito 同步用户数据和访问资源。 AWS 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，请参阅博客中的了解 Amazon Cognito 身份验证第 2 部分：经过开发人员身份验证的 AWS 身份。

了解身份验证流程

GetOpenIdTokenForDeveloperIdentityAPI 操作可以为增强身份验证和基本身份验证启动开发者身份验证。此 API 使用管理凭证对请求进行身份验证。该`Logins`地图是身份池开发者提供商的名称，例如与自定义标识符`login.mydevprovider`配对。

例如：

```
"Logins": {
        "login.mydevprovider": "my developer identifier"
    }
```

增强的身份验证

使用包含令牌名称`cognito-identity.amazonaws.com`和值`Logins`的地图调用 GetCredentialsForIdentityAPI 操作`GetOpenIdTokenForDeveloperIdentity`。

例如：

```
"Logins": {
        "cognito-identity.amazonaws.com": "eyJra12345EXAMPLE"
    }
```

`GetCredentialsForIdentity`使用经过开发者身份验证的身份，会返回身份池中默认经过身份验证的角色的临时证书。

基本身份验证

调用 AssumeRoleWithWebIdentityAPI 操作并请求已定义适当信任关系RoleArn的任何 IAM 角色的。将的值设置WebIdentityToken为从中获得的令牌GetOpenIdTokenForDeveloperIdentity。

有关经过开发人员身份验证的身份 authflow 以及它们与外部提供商身份有何区别的信息，请参阅。身份池（联合身份）身份验证流程

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：

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. 选择保存更改。

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。

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）

您可以通过调用获取令牌GetOpenIdTokenForDeveloperIdentity。必须使用 AWS 开发者凭据从您的后端调用此 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。如果攻击者获得此令牌，他们可以在令牌有效期内将其交换为最终用户的 AWS 凭证。

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。要将自定义身份与用户的社交身份（Login with Amazon、使用 Apple 登录、Facebook 或 Google 登录）关联起来，请在致电GetOpenIdTokenForDeveloperIdentity时将身份提供者令牌添加到登录地图中。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，调用该 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，请重写该AWSCognitoCredentialsProviderHelperlogins方法，以便能够根据当前身份提供者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，请重写该AWSCognitoCredentialsProviderHelperlogins方法，以便能够根据当前身份提供者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 中执行此操作的推荐方法是从 AmazonCognitoEnhancedIdentityProvide 而不是扩展您的身份提供商 AbstractCognitoIdentityProvider，并调用父 RefreshAsync 方法而不是您自己的方法，以防用户未使用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
- 同步数据
- 处理回调
- 推送同步
- Amazon Cognito Streams
- Amazon Cognito Events

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，请参阅开始使用 Amazon Cognito 身份池。

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。

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 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。

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。

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。

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。

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。

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。

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 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。

本指南提供了 Amazon Cognito 控制台中常见的 Amazon Cognito 用户池任务的 step-by-step 演练。

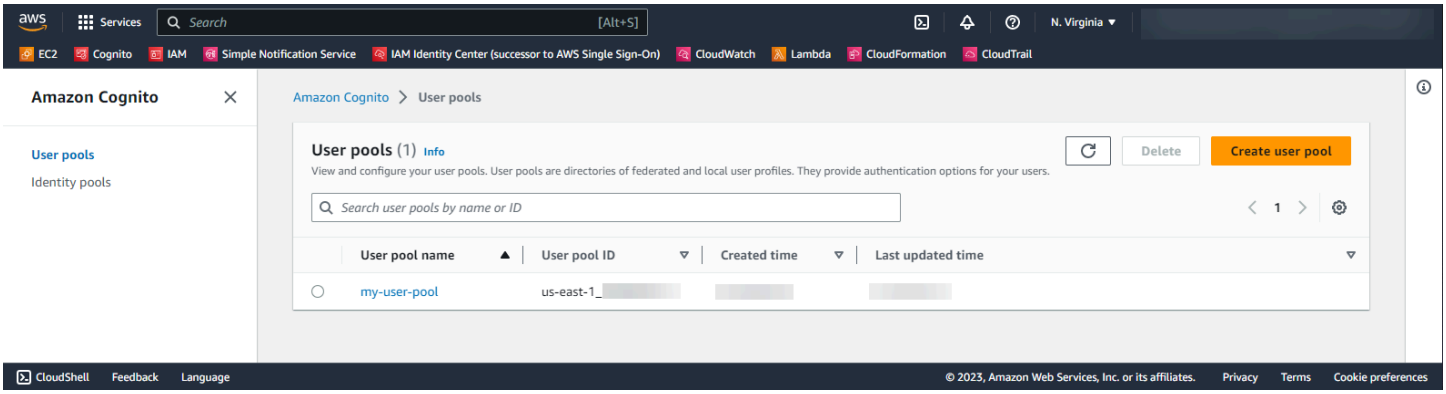
使用 Amazon Cognito 控制台

1. 要使用 Amazon Cognito，你需要注册一个账户。 AWS
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。
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 开始使用 Amazon Cognito 身份池。

Amazon Cognito 控制台是其中的一部分 AWS Management Console，它提供有关您的账户和账单的信息。有关更多信息，请参阅使用 AWS Management Console。



主题

- 用户群体控制台
- 身份池控制台

用户群体控制台

在 Amazon Cognito 控制台的用户群体视图中，从列表中选择一个用户群体以查看详细信息。在详细视图中，控制台顶部的用户群体概述包含有关您的用户群体的基本信息。以下选项卡将您的用户群体配置整理为相关功能。

用户

用户选项卡包含有关用户和从 CSV 文件导入用户的信息。您可以在此选项卡中添加、删除和编辑用户。

参考信息

- 管理用户池中的用户
- 通过 CSV 文件将用户导入用户池中

组

组选项卡包含有关用户组的信息。您可以添加、修改和更改组中的成员资格，也可以更改与组关联的 IAM 角色以实现身份池集成。

参考信息

- 向用户池添加组

登录体验

登录体验选项卡包含有关用户如何登录您的用户群体的信息。此选项卡包含第三方身份提供者、用户名选项、密码策略、多重身份验证（MFA）配置、忘记密码行为和设备记忆等。您可以添加和修改身份提供者，也可以更改用户群体的整体登录行为。

参考信息

- 通过第三方添加用户池登录
- 自定义登录属性
- 添加用户池密码要求
- 向用户池添加 MFA
- 恢复用户账户
- 使用用户群体中的用户设备

注册体验

注册体验选项卡包含有关自助注册、必需属性、验证电话号码和电子邮件地址以及自定义属性的信息。

参考信息

- 注册并确认用户账户
- 用户池属性
- 在注册时验证联系人信息

消息收发

消息收发选项卡包含有关您想用来向用户发送电子邮件和短信的 AWS 服务 以及要向他们发送的消息的格式的信息。

参考信息

- Amazon Cognito 用户池的电子邮件设置
- Amazon Cognito 用户池的短信设置
- 配置 SMS 和电子邮件验证消息以及用户邀请消息

应用程序集成

应用程序集成选项卡包含有关用户群体应用程序客户端、您分配给用户群体服务端点的域、API 资源服务器、托管 UI 和高级安全性的信息。您可以深入研究每个应用程序客户端，以配置以下内容。

1. 令牌设置
2. 回调 URL
3. 身份验证流程
4. 属性权限
5. 特定于应用程序的高级安全和托管 UI 设置
6. Amazon Pinpoint 分析

参考信息

- 用户池应用程序客户端
- 设置和使用 Amazon Cognito 托管 UI 和联合身份验证端点
- 配置用户池域
- 使用资源服务器进行范围、M2M 和 API 授权

- 向用户池添加高级安全
- 将 Amazon Pinpoint 分析与 Amazon Cognito 用户池结合使用

用户群体属性

用户池属性选项卡包含与用户无直接关系的用户池配置信息：Lambda 触发器、 AWS WAF Web ACL 保护、删除保护和资源标签。

参考信息

- 使用 Lambda 触发器自定义用户池工作流
- 将 AWS WAF Web ACL 与用户池关联
- 用户池删除保护
- 为资源添加 AWS 标签

身份池控制台

在 Amazon Cognito 控制台的身份池视图中，从列表中选择一个身份池以查看详细信息。在详细视图中，控制台顶部的身份池概述包含有关您的用户群体的基本信息。以下选项卡将您的用户群体配置整理为相关功能。

用户统计数据

用户统计数据选项卡显示有关在您的身份池中生成了身份的用户的统计信息。您无法在此选项卡中配置任何身份池设置。

身份浏览器

身份浏览器选项卡包含有关用户在您的身份池中生成的个人身份的信息。您可以查看和删除身份。

参考信息

- 开始使用 Amazon Cognito 身份池

用户访问权限

用户访问权限选项卡包含有关您已关联到身份池的身份提供者、开发人员提供者、分配给身份的原定设置 IAM 角色以及未经身份验证的访客访问配置的信息。您可以深入研究每个身份提供者，以配置以下内容。

1. 使用 IAM 角色选择进行基于角色的访问控制
2. 使用用于访问控制的属性进行基于属性的访问控制

参考信息

- 身份池外部身份提供商
- IAM 角色
- 经过身份验证和未经身份验证的身份
- 经开发人员验证的身份（身份池）
- 使用基于角色的访问控制
- 将属性用于访问控制

身份池属性

身份池属性选项卡包含有关其他身份池配置的信息：基本（经典）身份验证和资源标签。

- 身份池（联合身份）身份验证流程
- 为资源添加 AWS 标签

Amazon Cognito 中的安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将此描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 Amazon Cognito 的合规计划，请参阅按合规计划提供的范围内的AWS 服务按合规计划划。
- 云端安全-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。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 Amazon Cognito 资源。

内容

Amazon Cognito 中的数据保护

分担责任模型 AWS 分担责任模型适用于亚马逊 Cognito（Amazon Cognito）中的数据保护。如本模型所述 AWS ，负责保护运行所有 AWS 云的全球基础架构。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) 设置个人用户账户。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数据：

- 对每个账户使用多重身份验证 (MFA)。
- 使用 SSL/TLS 与资源通信。 AWS
- 使用设置 API 和用户活动日志 AWS CloudTrail。
- 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 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（例如名称字段）。这包括您使用控制台、API 或软件开发工具包使用 Amazon Cognito 或其他 AWS 服务时。 AWS CLI AWS 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网址时，请勿在网址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

作为一项托管服务，Amazon Cognito 受到 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用通过网络访问 Amazon Cognito。客户端必须支持以下内容：

- 传输层安全性协议 (TLS)。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密 (PFS) 的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。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 委托人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 用户池和身份池具有经过 IAM 身份验证、未经身份验证和经过令牌授权的 API 操作。未经身份验证和经过令牌授权的 API 操作旨在由您的客户（即您的应用程序的最终用户）使用。未经身份验证和经过令牌授权的 API 操作会进行静态加密和传输中加密。有关更多信息，请参阅 Amazon Cognito 用户池经过身份验证和未经身份验证的 API 操作。

> **Note**
>
> Amazon Cognito 在内部加密您的内容，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) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）来使用 Amazon Cognito 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

受众

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您在 Amazon Cognito 中所做的工作。

服务用户 – 如果您使用 Amazon Cognito 服务来完成任务，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 Amazon 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有助于您向管理员请求适合的权限。如果您无法访问 Amazon Cognito 中的一项功能，请参阅Amazon Cognito 身份和访问问题排查。

服务管理员 – 如果您在公司负责管理 Amazon Cognito 资源，则您可能具有 Amazon 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 Amazon 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 Amazon Cognito 结合使用的更多信息，请参阅Amazon Cognito 如何与 IAM 配合使用。

IAM 管理员 - 如果您是 IAM 管理员，您可能希望了解有关如何编写策略以管理对 Amazon Cognito 的访问权限的详细信息。要查看您可在 IAM 中使用的 Amazon 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。

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 AWS 账户根用户任 IAM 角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （ IAM Identity Center）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。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。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅 IAM 用户指南中的签署 AWS API 请求。

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 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 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。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《IAM 用户指南》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）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C enter 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。 AWS Directory Service当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户 和应用程序中使用。有关 IAM Identity Center 的信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

IAM 用户和群组

I AM 用户是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。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，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

I AM 角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 AWS Management Console 通过切换角色在中临时担任 IAM 角色。您可以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。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：

- 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 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。
- 跨账户存取 – 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，对于某些资源 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 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。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 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。
  - 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。
  - 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分派的 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-服务相关角色是一种链接到的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 A@@ mazon 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证书。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该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

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档

的 AWS 形式存储在中。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。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 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。

IAM 策略定义操作的权限，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。拥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 策略。

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。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。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。委托人可以包括账户、用户、角色、联合用户或 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 IAM 中的 AWS 托管策略。

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。

Amazon S3 和 Amazon VPC 就是支持 ACL 的服务示例。 AWS WAF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。

其他策略类型

AWS 支持其他不太常见的策略类型。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，用于指定组织或组织单位 (OU) 的最大权限。 AWS Organizations AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。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 限制成员账户中的实体（包括每个 AWS 账户根用户实体）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。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，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 Amazon Cognito 的访问权限之前，您应该了解哪些 IAM 功能可用于 Amazon Cognito。

可与 Amazon Cognito 一起使用的 IAM 功能

IAM 功能	Amazon Cognito 支持
基于身份的策略	是
基于资源的策略	否

IAM 功能	Amazon Cognito 支持
策略操作	是
策略资源	是
策略条件键	是
ACL	否
ABAC（策略中的标签）	部分
临时凭证	是
主体权限	否
服务角色	是
服务相关角色	是

要全面了解 Amazon Cognito 和其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中与 IAM 配合使用的AWS 服务。

Amazon Cognito 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 策略。

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 Amazon Cognito 的基于身份的策略示例

要查看 Amazon 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。

Amazon 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。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户存取，您可以将整个账户或其他账户中的 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。当委托人和资源处于不同位置时 AWS 账户，可信账户中的 IAM 管理员还必须向委托人实体（用户或角色）授予访问资源的权限。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。

Amazon 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。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关联操作的权限。

要查看 Amazon Cognito 操作的列表，请参阅《服务授权参考》中的 Amazon Cognito 定义的操作。

Amazon 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

当您使用 AWS 证书签署 Amazon Cognito API 请求时，您可以在 AWS Identity and Access Management (IAM) 策略中对其进行限制。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）。有关已签名的 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 用户池 API 操作的授权模式和角色的更多信息，请参阅Amazon Cognito 用户池经过身份验证和未经身份验证的 API 操作。

Amazon Cognito 身份池 API 操作

- GetId
- GetOpenIdToken
- GetCredentialsForIdentity
- UnlinkIdentity

Amazon Cognito 用户池 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 Amazon 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。

Amazon 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。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 资源名称（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 中的您的用户池，您可以使用以下 ARN 格式限制用户对特定用户池的访问权限：

```
arn:aws:cognito-idp:REGION:ACCOUNT_ID:userpool/USER_POOL_ID
```

有关 Amazon Cognito 资源类型及其 ARN 的列表，请参阅《服务授权引用》中的 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 Amazon 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。

Amazon 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。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，则使用逻辑OR运算来 AWS 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 策略元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

有关 Amazon Cognito 条件键的列表，请参阅《服务授权参考》中的 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 Amazon 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。

Amazon 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 Amazon Cognito 的基于属性的访问权限控制（ABAC）

支持 ABAC（策略中的标签）	部分

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于属性来定义权限。在中 AWS，这些属性称为标签。您可以向 IAM 实体（用户或角色）和许多 AWS 资源附加标签。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权限控制（ABAC）。

将临时凭证用于 Amazon Cognito

支持临时凭证	是

当你使用临时证书登录时，有些 AWS 服务 不起作用。有关更多信息，包括哪些 AWS 服务 适用于临时证书，请参阅 IAM 用户指南中的AWS 服务 与 IAM 配合使用的信息。

如果您使用除用户名和密码之外的任何方法登录，则 AWS Management Console 使用的是临时证书。例如，当您 AWS 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。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 AWS API 手动创建临时证书。然后，您可以使用这些临时证书进行访问 AWS。 AWS 建议您动态生成临时证书，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。

Amazon Cognito 的跨服务主体权限

支持转发访问会话（FAS）	否

当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

Amazon 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 Amazon Cognito 服务角色的详细信息，请参阅激活推送同步和推送同步。

> **Warning**
>
> 更改服务角色的权限可能会破坏 Amazon Cognito 的功能。仅当 Amazon Cognito 提供相关指导时才编辑服务角色。

Amazon 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服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。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 Amazon 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。

适用于 Amazon Cognito 的基于身份的策略示例

原定设置情况下，用户和角色没有创建或修改 Amazon 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 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 策略。

有关 Amazon 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 的格式，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
- 使用 Amazon Cognito 控制台
- 允许用户查看他们自己的权限
- 限制对特定身份池的控制台访问权限
- 允许池中的所有身份访问特定数据集

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 Amazon 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：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管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 策略设置权限时，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 策略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。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 AWS CloudFormation。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 策略，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 策略语言 (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。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。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 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实操。

> **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 Amazon 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的 Amazon Cognito 资源的详细信息。 AWS 账户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。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。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然可以使用 Amazon Cognito 控制台，还需要将亚马逊 Co ConsoleAccess gnito ReadOnly AWS 或托管策略附加到实体。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或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "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

使用以下信息帮助您诊断和修复在使用 Amazon Cognito 和 IAM 时可能遇到的常见问题。

主题

- 我没有在 Amazon Cognito 中执行操作的权限
- 我无权执行 iam：PassRole
- 我是管理员并希望允许其他人访问 Amazon Cognito
- 我想允许 AWS 账户以外的人访问我的 Amazon Cognito 资源

我没有在 Amazon Cognito 中执行操作的权限

如果您收到错误提示，表明您无权执行某个操作，则您必须更新策略以允许执行该操作。

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。您的管理员是提供登录凭证的人。

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 Amazon Cognito。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的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 Amazon 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。您的管理员是提供登录凭证的人。

我是管理员并希望允许其他人访问 Amazon Cognito

要允许其他人访问 Amazon 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 Amazon 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。

我想允许 AWS 账户以外的人访问我的 Amazon Cognito 资源

您可以创建一个角色，以便其他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 Amazon Cognito 是否支持这些功能，请参阅Amazon Cognito 如何与 IAM 配合使用。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）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的 IAM 角色，其信任策略 AWS 服务 允许担任该角色。服务相关角色由 Amazon Cognito 预定义，包括该服务代表您调用 AWS 其他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他 IAM 实体的权限策略。

只有在首先删除相关资源后，您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 服务并查找服务相关角色列中显示为是的服务。选择是和链接，查看该服务的服务相关角色文档。

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 用户池服务使用你的 Amazon SES 身份发送电子邮件。

- AWSServiceRoleForAmazonCognitoIdp— 允许 Amazon Cognito 用户池为你的 Amazon Pinpoint 项目发布事件和配置终端节点。

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：

允许的操作 AWSServiceRoleForAmazonCognitoIdpEmailService：

- 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。

您必须配置权限，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来代入该角色：

- `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：

允许的操作 AWSServiceRoleForAmazonCognitoIdp

- 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

您无需手动创建服务相关角色。当您将用户池配置为使用您的 Amazon SES 配置来处理在 AWS Management Console AWS CLI、或 Amazon Cognito API 中发送电子邮件时，Amazon Cognito 会为您创建服务相关角色。

如果您删除该服务相关角色，然后需要再次创建，您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

您无法在中编辑 AmazonCognitoIdp 或 AmazonCognitoIdpEmailService 与服务相关的角色。 AWS Identity and Access Management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。在删除角色 AmazonCognitoIdp 或 AmazonCognitoIdpEmailService 服务相关角色之前，必须对使用该角色的每个用户池执行以下操作之一：

- 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。

请记住使用该角色 AWS 区域 的用户池在每个用户池中执行操作。

> **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，网址为。https://console.aws.amazon.com/cognito
2. 选择管理用户池。
3. 在您的用户池页面上，选择要删除的用户池。
4. 选择删除池。
5. 在删除用户池窗口中，键入 **delete**，然后选择删除池。

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，网址为。https://console.aws.amazon.com/cognito
2. 选择管理用户池。
3. 在您的用户池页面上，选择要更新的用户池。
4. 在左侧导航菜单中，选择消息自定义。
5. 在是否要通过 Amazon SES 配置发送电子邮件？下，选择否 -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 删除 AmazonCognitoIdp 或 AmazonCognitoIdpEmailService 与服务相关的角色。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 在所有提供服务 AWS 区域 的地方都支持与服务相关的角色。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

监控是维护 Amazon Cognito 和其他 AWS 解决方案的可靠性、可用性和性能的重要组成部分。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 — 通过使用， CloudTrail 您可以捕获来自亚马逊 Cognito 控制台的 API 调用，以及从对亚马逊 Cognito API 操作的代码调用中捕获 API 调用。例如，当用户进行身份验证时，CloudTrail 可以记录诸如请求中的 IP 地址、谁发出请求以及何时发出请求之类的详细信息。
- Amazon CloudWatch Logs — 借助 CloudWatch 日志，您可以向日志组发送精细的用户活动日志。例如，您可以查看详细的用户活动日志，以排查向用户发送电子邮件和短信时遇到的问题。
- Amazon CloudWatch Metrics — 借助 CloudWatch 指标，您可以近乎实时地监控、报告事件并在发生事件时自动采取行动。例如，您可以根据提供的指标创建 CloudWatch 控制面板来监控您的 Amazon Cognito 用户池，也可以根据提供的指标创建 CloudWatch 警报，以便在违反设定阈值时通知您。
- Amazon CloudWatch Logs Insig hts — 借助 CloudWatch Logs Insights，您可以配置 CloudWatch 为将事件发送 CloudTrail 到以监控 Amazon Cognito CloudTrail 日志文件。

主题

监控成本

Amazon Cognito 会根据您的以下使用量收费。

- 用户池每月活跃用户 (MAU)
- 使用 OIDC 或 SAML 联合身份登录的用户池 MAU
- 具有高级安全功能的用户池中的 MAU
- 活跃的用户池应用程序客户端和使用客户端凭证进行机器到机器 (M2M) 授权的请求量
- 某些类别的用户池 API 的购买使用量超过了默认配额

此外，用户池的功能（如电子邮件、SMS 消息和 Lambda 触发器）可能会在依赖服务中产生费用。有关完整概述，请参阅亚马逊 Cognito 定价。

查看和预测成本

您可以在AWS Billing and Cost Management 控制台中查看和报告您的 AWS 成本。您可以在 “账单和付款” 部分找到您最近的 Amazon Cognito 费用。在 “账单”、“按服务计费” 下，筛选Cognito以查看您的使用情况。有关更多信息，请参阅 AWS Billing 用户指南中的查看您的账单。

要监控 API 请求速率，请在 Service Quotas 控制台中查看利用率指标。例如，客户端凭证请求显示为 ClientAuthentication 请求率。在您的账单中，这些请求与生成这些请求的应用程序客户端相关联。有了这些信息，您就可以公平地将成本分配给多租户架构中的租户。

要获取一段时间内的 M2M 请求数量，您还可以将AWS CloudTrail 事件发送到 CloudWatch Logs 进行分析。使用客户端凭证授予 CloudTrail `Token_POST`的事件查询您的事件。以下 CloudWatch Insights 查询返回此计数。

```
filter eventName = "Token_POST" and @message like '"grant_type":["client_credentials"]'
 | stats count(*)
```

管理 成本

Amazon Cognito 根据用户数量、功能使用情况和请求量计费。以下是在亚马逊 Cognito 中管理成本的一些技巧，

不要激活不活跃的用户

使用户处于活动状态的典型操作是登录、注册和密码重置。有关更详尽的列表，请参阅每月活跃用户。Amazon Cognito 不会将不活跃的用户计入您的账单。避免任何使用户处于活动状态的操作。不AdminGetUser使用 API 操作，而是使用该ListUsers操作查询用户。不要对非活跃用户的用户池操作进行大量管理测试。

链接联合用户

使用 SAML 2.0 或 OpenID Connect (OIDC) 身份提供商登录的用户的成本要高于本地用户。您可以将这些用户链接到本地用户配置文件。关联用户可以使用其联合用户附带的属性和访问权限以本地用户身份登录。来自 SAML 或 OIDC IdPs 的用户在一个月内仅使用关联的本地账户登录，则按本地用户计费。

管理请求费率

如果您的用户池已接近配额的上限，则可以考虑购买额外的容量来处理该容量。您也许可以减少应用程序中的请求量。有关更多信息，请参阅 优化配额限制的请求速率。

仅在需要新令牌时才申请新代币

使用客户端凭证授予的机器对机器 (M2M) 授权可以满足大量令牌请求。每个新的令牌请求都会影响您的请求速率配额和账单规模。为了优化成本，请在应用程序设计中包括令牌到期设置和令牌处理。

- 缓存访问令牌，以便当您的应用程序请求新令牌时，它会收到先前发布的令牌的缓存版本。当您实现此方法时，您的缓存代理可以防范那些在不知道先前获取的令牌已过期的情况下请求访问令牌的应用程序。缓存令牌非常适合于 Lambda 函数和 Docker 容器等短期微服务。
- 在您的应用程序中实现令牌处理机制，以考虑令牌过期。在之前的代币到期之前，请勿申请新令牌。评估每个应用程序的机密性和可用性需求，并将用户池应用程序客户端配置为颁发具有适当有效期的访问令牌。自定义令牌持续时间最适合使用寿命较长的 API 和可以持续管理凭证请求频率的服务器。

删除未使用的客户端凭据应用程序客户端

M2M 授权账单基于两个因素：令牌请求率和授予客户凭证的应用程序客户端数量。如果未使用用于 M2M 授权的应用程序客户端，请将其删除或移除其颁发客户端凭据的授权。有关管理应用程序客户端配置的更多信息，请参阅用户池应用程序客户端。

管理高级安全

在用户池中配置高级安全功能时，高级安全计费费率适用于用户池中的所有 MAU。如果您的用户不需要高级安全功能，请将他们分成另一个用户池。

跟踪和 Service Quotas 中的配额 CloudWatch 和使用情况

您可以使用亚马逊 CloudWatch 或服务配额来监控 Amazon Cognito 用户池。您还可以在 Service Quotas 中监控身份池的使用情况。 CloudWatch 收集原始数据并将其处理成可读的、近乎实时的指标。在中 CloudWatch，您可以设置警报，监视某些阈值，并在达到这些阈值时发送通知或采取行动。要为服务配额创建 CloudWatch 警报，请参阅创建 CloudWatch警报。Amazon Cognito 指标每隔五分钟提供一次。有关中保留期限的更多信息 CloudWatch，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 用户池及身份池配额使用情况。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。有关中保留期限的更多信息 CloudWatch，请访问 Amazon CloudWatch 常见问题页面。

如果您登录 AWS 账户 的是设置为 CloudWatch跨账户可观察性的监控账户，则可以使用该监控账户来可视化服务配额，并为与该监控账户关联的源账户中的指标设置警报。有关更多信息，请参阅CloudWatch 跨账户可观察性。

主题

- 记录来自 Amazon Cognito 用户池的其他活动
- Amazon Cognito 用户池指标
- Amazon Cognito 用户池的维度
- 使用 Service Quotas 控制台来跟踪指标
- 使用 CloudWatch 控制台跟踪指标
- 为配额创建 CloudWatch 警报

记录来自 Amazon Cognito 用户池的其他活动

您可以将用户池配置为向日志组发送一些其他活动的详细 CloudWatch 日志。这些日志的粒度比中的日志更精细 AWS CloudTrail，可用于对用户池进行故障排除。激活此功能后，您可以选择希望 Amazon Cognito 将日志发送到哪个日志组。当您想了解您的用户池通过 Amazon SNS 和 Amazon SES 传递的电子邮件和 SMS 消息的状态时，用户活动日志记录非常有用。

目前，您只能从用户池中传递错误级别的用户通知日志。

详细的日志记录不会取代或更改用户池的以下日志功能。

1. CloudTrail 常规用户活动日志，例如注册和登录。
2. 使用 CloudWatch 指标大规模分析用户活动。

另外，您还可以在日志中找到来自用户导入任务和 Lambda 触发器的日志。 CloudWatch Amazon Cognito 和 Lambda 将这些日志存储在与您为详细活动日志指定的日志组不同的日志组中。

您可以在 API 请求中使用 Amazon Cognito 用户池 API 配置详细的SetLogDeliveryConfiguration活动日志。您可以在 GetLogDeliveryConfigurationAPI 请求中查看用户池的日志配置。

您必须使用具有以下权限的 AWS 证书来授权这些请求。

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

```
            {
            "Sid": "ManageUserPoolLogs",
            "Action": [
                "cognito-idp:SetLogDeliveryConfiguration",
                "cognito-idp:GetLogDeliveryConfiguration",
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "CognitoLog",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "CognitoLoggingCWL",
            "Action": [
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

以下是用户池中的一个示例事件。此日志架构可能会发生变化。某些字段可能记录为空值。

```
{
```

```
    "eventTimestamp": "1687297330677",
    "eventSource": "USER_NOTIFICATION",
    "logLevel": "ERROR",
    "message": {
        "details": "String"
    },
    "logSourceId": {
        "userPoolId": "String"
    }
}
```

从 Amazon Cognito 传输日志将尽力而为。您的用户池提供的日志量以及您的日志服务配额可能会影响 CloudWatch 日志的传输。

CloudWatch 启用日志传输后，将收取日志费用。有关更多信息，请参阅 Amazon CloudWatch 定价中的销售日志。

要将日志发送到资源策略大小超过 5120 个字符的日志组，请使用以 `/aws/vendedlogs` 开头的路径配置日志组。有关更多信息，请参阅启用某些 AWS 服务的日志记录。

Amazon Cognito 用户池指标

下表列出了对 Amazon Cognito 用户池可用的指标。Amazon Cognito 的 Amazon CloudWatch 指标命名空间是 `AWS/Cognito`。有关更多信息，请参阅 A mazon CloudWatch 用户指南中的命名空间。

> **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`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。

指标	描述
	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总数，请使用 CloudWatch Math表达式并从Sum统计数据中减去统计 Sample Count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

指标	描述
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。 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总数，请使用 CloudWatch Math表达式并从Sum统计数据中减去统计Sample Count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

指标	描述
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。要计算刷新 Amazon Cognito 令牌的失败请求总数，请使用CloudWatchMath表达式并从Sum统计数据中减去统计数据。Sample Count 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

指标	描述
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。要计算失败的身份联合请求总数，请使用 CloudWatch Math表达式并从Sum统计数据中减去统计Sample Count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

指标	描述
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。 在账户和地区的所有用户池中，每个 AWS 账户都必须使用类别配额。 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
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

借助 Service Quotas，您可以从一个中心位置查看和管理 Amazon Cognito 用户池及身份池配额。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。对于某些配额类型，您可以创建 CloudWatch警报来跟踪您的配额使用情况。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。

要查看 Amazon Cognito 用户池和身份池的服务限额使用情况，请完成以下步骤。

1. 打开服务限额控制台。
2. 在导航窗格中，选择 AWS 服务。
3. 从 AWS 服务列表中，搜索并选择 Amazon Cognito 用户池或 Amazon Cognito 联合身份。此时将显示服务配额页面。
4. 选择支持 CloudWatch 监控的配额。例如，在 Amazon Cognito 用户池中选择 `Rate of UserAuthentication requests`。
5. 向下滚动到监控。此部分仅针对支持 CloudWatch 监控的配额显示。
6. 在监控中，您可以在图表中查看当前服务配额利用率。
7. 在监控)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。在这里，您可以将图表添加到控制面板或使用操作菜单选择在指标中查看，这将带您进入 CloudWatch 控制台中的相关指标。

使用 CloudWatch 控制台跟踪指标

您可以使用跟踪和收集 Amazon Cognito 用户池指标。 CloudWatch CloudWatch 控制面板将显示有关您使用的每项 AWS 服务的指标。您可以使用 CloudWatch 创建指标警报。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。要在中查看服务配额指标 CloudWatch，请完成以下步骤。

1. 打开CloudWatch控制台。
2. 在导航窗格中，选择指标。
3. 在所有指标中，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“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 警报

Amazon Cognito 提供的 CloudWatch 使用量指标与`CallCount`和 `ThrottleCount` API 的 AWS 服务配额相对应。有关在中跟踪使用情况的更多信息 CloudWatch，请参阅跟踪配额使用量。

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。要了解如何使用服务配额控制台设置 CloudWatch 警报，请参阅服务配额和 CloudWatch 警报。

使用记录亚马逊 Cognito API 调用 AWS CloudTrail

Amazon Cognito 与 AWS CloudTrail一项服务集成，该服务提供用户、角色或 AWS 服务在 Amazon Cognito 中采取的操作的记录。 CloudTrail 捕获一部分 Amazon Cognito 的 API 调用作为事件，包括来自亚马逊 Cognito 控制台的调用和对亚马逊 Cognito API 操作的代码调用。如果您创建了跟踪，则可以选择将 CloudTrail 事件传送到 Amazon S3 存储桶，包括 Amazon Cognito 的事件。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。通过收集的信息 CloudTrail，您可以确定向 Amazon Cognito 发出的请求、发出请求的 IP 地址、谁提出了请求、何时提出请求以及其他详细信息。

要了解更多信息 CloudTrail，包括如何配置和激活它，请参阅AWS CloudTrail 用户指南。

您还可以为特定 CloudTrail 事件创建 Amazon CloudWatch 警报。例如，您可以设置 CloudWatch 为在身份池配置发生更改时触发警报。有关更多信息，请参阅为 CloudTrail 事件创建 CloudWatch 警报：示例。

主题

- 亚马逊 Cognito 中的信息 CloudTrail
- 了解 Amazon Cognito 登录事件
- 使用亚马逊日志见解分析 Amazon Cognito CloudTrail 事件 CloudWatch

亚马逊 Cognito 中的信息 CloudTrail

CloudTrail 在您创建时已开启 AWS 账户。当 Amazon Cognito 中出现支持的事件活动时，该活动会与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在自己的 AWS 账户中查看、搜索和下载最近发生的事件。有关更多信息，请参阅使用事件历史查看 CloudTrail 事件。

要持续记录您的 AWS 账户中的事件，包括 Amazon Cognito 的事件，请创建跟踪。 CloudTrail 跟踪将日志文件传输到 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为以下各项配置亚马逊 SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail日志文件

每个事件或日记账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用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅CloudTrail 用户身份元素。

中的机密数据 AWS CloudTrail

由于用户池和身份池会处理用户数据，因此 Amazon Cognito 会使用该值掩盖 CloudTrail 事件中的一些私有字段。HIDDEN_FOR_SECURITY_REASONS有关 Amazon Cognito 未填充到事件的字段的示例，请参阅了解 Amazon Cognito 登录事件。Amazon Cognito 只会掩盖一些通常包含用户信息的字段，例如密码和令牌。Amazon Cognito 不会自动检测或屏蔽您在 API 请求中填充到非私有字段的个人身份信息。

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事件 CloudTrail 记录在日志文件中。Amazon Cognito 将用户池事件记录 CloudTrail 为管理事件。

Amazon Cognito 用户池 CloudTrail 条目中的eventType字段告诉你你的应用程序是向 A mazon Cognito 用户池 API 发出请求，还是向为 OpenID Connect、SAML 2.0 或托管用户界面提供资源的终端节点发出了请求。API 请求的 AwsApiCall 为 eventType，端点请求的 AwsServiceEvent 为 eventType。

Amazon Cognito 将以下托管用户界面请求作为事件记录到您的托管用户界面。 CloudTrail

托管的 UI 操作位于 CloudTrail

操作	描述
Login_GET，CognitoAuthentication	用户查看或向您的 登录端点 提交凭证。
OAuth2_Authorize_GET，Beta_Authorize_GET	用户查看您的 对端点授权。
OAuth2Response_GET，OAuth2Response_POST	用户向您的 /oauth2/idpresponse 端点提交 IdP 令牌。
SAML2Response_POST，Beta_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 中输入用户名并与 IdP 标识符匹配。
Token_POST，Beta_Token_POST	用户向您的 令牌端点 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。

操作	描述
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。
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。
MfaRegister_GET ，MfaRegister_POST	用户在注册 MFA 时，在托管 UI 中提交多重身份验证（MFA）代码。
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 端点 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 会记录特定`UserName`于用户的请求，`UserSub`但不记录在 CloudTrail 日志中。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 将以下亚马逊 Cognito 身份事件记录 CloudTrail 为数据事件。数据事件是大容量数据平面 API 操作， CloudTrail 默认情况下不记录。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，您必须激活跟踪中的数据事件，并为 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。 CloudTrail 默认情况下会记录管理事件 API 操作。

有关 Amazon Cognito 登录 CloudTrail到的 Amazon Cognito 身份池 API 操作的列表，请参阅亚马逊 Co gnito 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 登录 CloudTrail的亚马逊 Cognito Sync API 操作的列表，请参阅亚马逊 Cognito Sy nc 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输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序出现。

主题

- 托管用户界面注册的示例 CloudTrail 事件
- SAML 请求的示例 CloudTrail 事件
- 向令牌端点发出请求 CloudTrail 的事件示例
- 的示例 CloudTrail 事件 CreateIdentityPool
- 的示例 CloudTrail 事件 GetCredentialsForIdentity
- 的示例 CloudTrail 事件 GetId
- 的示例 CloudTrail 事件 GetOpenIdToken
- 的示例 CloudTrail 事件 GetOpenIdTokenForDeveloperIdentity
- 的示例 CloudTrail 事件 UnlinkIdentity

托管用户界面注册的示例 CloudTrail 事件

以下示例 CloudTrail 事件演示了用户通过托管用户界面注册时 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"
}
```

向令牌端点发出请求 CloudTrail 的事件示例

以下是来自对 令牌端点 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"
}
```

的示例 CloudTrail 事件 CreateIdentityPool

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"
}
```

的示例 CloudTrail 事件 GetCredentialsForIdentity

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"
}
```

的示例 CloudTrail 事件 GetId

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"
}
```

的示例 CloudTrail 事件 GetOpenIdToken

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"
}
```

的示例 CloudTrail 事件 GetOpenIdTokenForDeveloperIdentity

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"
}
```

的示例 CloudTrail 事件 UnlinkIdentity

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"
}
```

使用亚马逊日志见解分析 Amazon Cognito CloudTrail 事件 CloudWatch

您可以使用 Amazon Log CloudWatch s Insights 搜索和分析您的 Amazon Cognito CloudTrail 事件。当您将跟踪配置为向 CloudWatch Logs 发送事件时，仅 CloudTrail 发送与您的跟踪设置相匹配的事件。

要查询或研究您的 Amazon Cognito CloudTrail 事件，请在 CloudTrail 控制台中确保在跟踪设置中选择管理事件选项，以便您可以监控对资源执行的管理操作。 AWS 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 控制台中使用以下查询。

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

作为多项合规计划的一部分，第三方审计师对 Amazon Cognito 的安全与 AWS 合规性进行评估。其中包括 SOC、PCI、FedRAMP、HIPAA 及其他。

有关特定合规计划范围内的 AWS 服务列表，请参阅AWS 按合规计划划分的范围内AWS 服务（按合分）。有关常规信息，请参阅 AWS 合规性计划。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

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。
- HIPAA 安全与合规架构白皮书 — 本白皮书描述了各公司如何使用它来 AWS 创建符合 HIPAA 标准的应用程序。
- AWS 合规资源AWS — 此工作簿和指南集合可能适用于您所在的行业和所在地区。
- 使用《AWS Config 开发人员指南》中的规则评估资源 — AWS Config; 评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 此 AWS 服务可全面了解您的安全状态 AWS ，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

区域数据注意事项

Amazon Cognito 用户池均在一个 AWS 区域中创建，并且它们仅在该区域存储用户个人资料数据。用户池可以将用户数据发送到不同的 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。
- 如果使用不同的电子邮件地址来配置带有 Amazon Cognito 用户池的亚马逊简单电子邮件服务 (Amazon SES)，则该电子邮件地址将 AWS 通过与 Amazon SES 中的电子邮件地址关联的区域进行路由。
- 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 已在北美、欧洲、亚洲和大洋洲的多个 AWS 地区上市。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

作为一项托管服务，Amazon Cognito 受到 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用通过网络访问 Amazon Cognito。客户端必须支持以下内容：

- 传输层安全性协议 (TLS)。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密 (PFS) 的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。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 委托人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 处理基本的安全任务，例如客户机操作系统 (OS) 和数据库修补、防火墙配置和灾难恢复。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
- 责任共担模式

AWS 亚马逊 Cognito 的托管策略

要向用户、群组和角色添加权限，使用 AWS 托管策略比自己编写策略要容易得多。创建仅为团队提供所需权限的 IAM 客户管理型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。这些政策涵盖常见用例，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 托管策略中移除权限，因此策略更新不会破坏您的现有权限。

此外，还 AWS 支持跨多个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。当服务启动一项新功能时， 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 托管式策略。

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**
>
> 创建新的身份池时，您可以自动为经过身份验证的用户和访客用户访问权限创建新角色。使用新的 IAM 角色创建身份池的管理员还必须拥有 IAM 权限才能创建角色。

具有未经身份验证的访客访问权限的身份池会将额外的 AWS 托管策略作为会话策略应用于未经身份验证的用户。AmazonCognitoUnAuthedIdentitiesSessionPolicy此 AWS 托管策略没有预期的管理用途。相反，它限制了您可以对身份池增强的身份验证流程中的访客用户应用的权限范围。有关更多信息，请参阅 IAM 角色。

亚马逊 Cognito 更新了托管 AWS 政策

查看自该服务开始跟踪这些更改以来，Amazon Cognito 托 AWS 管政策更新的详细信息。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	描述	日期
AmazonCognitoUnAuthedIdentitiesSessionPolicy – 新策略	添加了用于缩小身份池中访客用户的权限范围的 AWS 托管策略。	2023 年 7 月 14 日
AmazonCognitoPowerUser 和 AmazonCognitoReadOnly – 现有策略更新	添加了新的权限，允许高级用户查看和管理 AWS WAF 网页 ACL 与 Amazon Cognito 用户池的关联。 添加了新的权限，允许只读用户查看 AWS WAF 网页 ACL 与 Amazon Cognito 用户池的关联。	2022 年 7 月 19 日
AmazonCognitoPowerUser – 更新了现有策略	添加了一项新权限，允许 Amazon Cognito 调用 Amazon Simple Notification Service PutIdentityPolicy 和 ListConfigurationSets 操作。	2021 年 11 月 17 日

更改	描述	日期
	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。	
`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 开始跟踪其托 AWS 管策略的变更。	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默认限额，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 配额将最大请求量合而为一 AWS 区域 。 AWS 账户例如，您的应用程序可以对美国东部（弗吉尼亚州北部）中的所有用户池，最高以默认配额（RPS）速率为 `UserAuthentication` 操作发出 API 请求。您在亚太地区（东京）的应用程序可以针对您所在地区的所有用户池生成相同数量的请求。 AWS 一次只能在一个地区批准增加配额的请求。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

配额分类

Amazon Cognito 对 API 操作强制执行最高请求速率。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，对于用户池，这些操作分为几个常见使用案例类别，例如 `UserAuthentication` 或 `UserCreation`。有关按类别划分的用户池 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。

在 S ervice Quotas 控制台中，您可以按类别用户池和身份池跟踪配额使用情况。如果您的 Amazon Cognito 用户池的请求速率或超过配额，则可以购买更多容量。您可以在 Service Quotas 控制台中按类别跟踪用户池配额使用情况和购买配额增加情况。

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池。Amazon Cognito 将每个身份池配额应用于单个操作。对于每个类别和每个操作的请求速率配额， AWS 衡量来自一个区域内所有用户池或身份池的所有请求的总速率。 AWS 账户

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。

该`UserAuthentication`类别包括 Amazon Cognito 用户池 API 中的四个操作：`AdminInitiateAuth`、`InitiateAuth`、`AdminRespondToAuthChallenge`、和。`RespondToAuthChallenge`此外，托管用户界面中的用户身份验证也占据了这一配额。`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。此提升的配额可容纳您的应用程序中设置的多个身份验证挑战。此配额足以涵盖大多数使用案例。在您的应用对身份验证质询做出最多三次响应后，其他请求将计入`UserAuthentication`类别配额。多因素身份验证 (MFA) 、设备身份验证和自定义身份验证都是您可能在用户池中设计的质询提示的示例。

例如，如果该`UserAuthentication`类别的配额为 80 RPS，则可以拨打`RespondToAuthChallenge`或`AdminRespondToAuthChallenge`按高达 240 RPS（3* 80 RPS）的费率拨打。如果您的用户池提示每次身份验证进行四轮质询，并且每秒 70 名用户登录，则总数`RespondToAuthChallenge`为 280 RPS（70 x 4），比配额高出 40 个 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）。由于此值超出设置的 80 RPS 的类别配额 30 RPS，因此 Amazon Cognito 会限制来自您的应用程序的请求。

每月活跃用户

当 Amazon Cognito 计算用户池账单时，它会根据每个月活跃用户 (MA U) 向您收取费率。在计划增加配额请求时，请考虑您当前和预计的 MAU 数量。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

- 退出
- 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

> **Note**
>
> “查询用户的详细属性” 类别包括 API 操作 AdminGetUser，但不包括ListUsers。大型用户池中的详细 user-by-user 查询可能会对您的 AWS 账单产生重大影响。为避免额外收费，请使用外部数据库收集用户数据`ListUsers`或将用户信息存储在外部数据库中。

管理 API 请求速率配额

确定配额要求

> **Important**
>
> 如果您增加`UserAuthentication`、`UserCreation`或等类别的 Amazon Cognito 配额`AccountRecovery`，则可能需要增加其他类别的配额。 AWS 服务例如，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。

如果考虑到八小时内用户登录频 peak-to-average 率的差异，假设比率为 3:1，则需要所需的`UserAuthentication`配额 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。

优化配额限制的请求速率

由于提高 API 速率限制会增加 AWS 账单成本，因此在申请增加配额之前，请考虑调整使用模式。以下是一些优化请求速率的应用程序架构示例。

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，请参阅构建工具 AWS。

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 JSON 网络令牌 (JWT)

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 请求。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。

缓存 JWT

重复使用访问令牌，直到其过期。有关在 API Gateway 中使用令牌缓存的框架示例，请参阅缓存令牌。与其生成 API 请求来查询用户信息，不如将 ID 令牌缓存到期，然后从缓存中读取用户属性。

有关在中使用 API 请求速率的更多信息 AWS，请参阅管理和监控工作负载中的 API 限制。有关优化会增加 AWS 账单成本的 Amazon Cognito 操作的信息，请参阅。管理 成本

跟踪配额使用量

Amazon Cognito 在亚马逊中 CloudWatch 为账户级别的每个 API 操作类别生成`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 使用量指标。

监控服务配额时，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%。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 指标跟踪使用情况

您可以使用跟踪和收集 Amazon Cognito 用户池使用率指标。 CloudWatch CloudWatch仪表板会显示有关您 AWS 服务 使用的每一项的指标。使用 CloudWatch，您可以创建指标警报来通知您或更改您正在监控的特定资源。有关 CloudWatch 指标的更多信息，请参阅跟踪您的 CloudWatch 使用量指标。

通过 Service Quotas 指标跟踪利用率

Amazon Cognito 用户池与 Service Quotas 集成，后者是一个控制台界面，用于显示和管理您的服务配额使用情况。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 警报。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表。

Amazon Cognito 用户池和 Amazon Cognito 身份池的服务配额控制台中的已申请账户级别配额值列显示您的当前配额。利用率列显示您当前的配额使用率。可调整的 Amazon Cognito 用户池 requests-per-second (RPS) 配额会显示其当前使用情况。Service Quotas 控制台还可以将您导航到 CloudWatch 指标，以便仔细查看所选配额指标。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。

跟踪每月活跃用户 (MAU)

用户池中的月活跃用户 (MAU) 数量为你计划增加请求率配额提供了重要数据。您可以将您的 API 请求率与您在给定时间段内的活跃用户数量进行比较。有了这些知识，您就可以计算出应用程序活跃用户的增加将如何影响您的使用模式中的配额。例如，假设您在美国西部（俄勒冈州）的合并应用程序在一个月内产生了 200 万活跃用户，而您的`UserAuthentication`类别偶尔会出现限制错误，默认配额为每秒 120 个请求 (RPS)。在广告活动成功之前的上个月，您有 100 万个 MAU，并且您的申请从未超过 80 RPS。如果您预计新的电视广告会导致类似的激增，则可以额外购买 40 个 RPS，以容纳下一百万用户，调整后的配额为 160 RPS。

查看您的 MAU

访问AWS Billing 控制台并查看最近的账单。在 “按服务计费” 下，您可以在 Cognito 上进行筛选，以查看该账单周期的 MAU 明细。

请求提高限额

Amazon Cognito 在用户池和每个用户池中的身份池中都有每秒可以执行的最大操作数的配额。 AWS 区域您可以购买增加可调整的 Amazon Cognito 用户池 API 请求速率配额。查看您当前的配额，然后通过服务配额控制台或通过服务配额 API 操作`ListAWSDefaultServiceQuotas`购买增量配额`RequestServiceQuotaIncrease`。

- 要使用 Service Quotas 控制台购买增加配额，请参阅《服务配额用户指南》中的申请 API 配额增加。
- AWS 目标是在 10 天内完成增加配额的请求。但是，有几个注意事项可能会导致请求处理时间超过 10 天。例如，某些请求可能需要Amazon Cognito预配置额外的硬件容量，而请求量的季节性增加可能会导致延迟。
- 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。您必须购买增加的配额容量。有关提高配额的定价，请参阅 Amazon Cognito 定价。

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 的所有用户池。您只能请求提升可调整类别配额。有关更多信息，请参阅 请求提高限额。配额调整适用于您的账户在单个区域中的用户池。对于每个用户池，Amazon Cognito 将某些类别[3]中的操作限制为每秒 5 次请求（RPS）。默认配额 (RPS) 还适用于中的所有用户池。 AWS 账户

> **Note**
>
> 每个类别的配额按每月活跃用户数（MAU）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 少于一百万，而 Amazon Cognito 正在限制请求，请考虑优化您的应用程序。有关更多信息，请参阅 优化配额限制的请求速率。

类别操作配额适用于一个 AWS 区域的所有用户池中的所有用户。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 用户池中每个用户的请求速率配额

操作	每个用户每秒操作数
读取用户配置文件 示例：`GetUser`、`GetDevice`	10
写入用户配置文件 示例：`UpdateUserAttributes` 、`SetUserSettings`	10

您必须限制每个类别的 API 请求数，如下表所示。

Amazon Cognito 用户池中每个类别的请求速率配额

类别	描述	默认配额（RPS）	可调整
`UserAuthentication` • InitiateAuth • 使用 `InitiateAuth` 或 令牌端点 刷新令牌 • RespondToAuthChallenge[1] • AdminInitiateAuth • AdminRespondToAuthChallenge[1]	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 的制约。	120	是

类别	描述	默认配额（RPS）	可调整
• 托管 UI 登录和授权码或隐式授权中的 MFA[2]			
`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	可以
`UserFederation` 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。	向用户池联合身份验证端点提交 IdP 响应的操作。生成 IdP 令牌的 OIDC 或社交服务提供商操作以及所有 SAML 请求一起构成此限额。	25	是

类别	描述	默认配额（RPS）	可调整
UserAccountRecovery • ChangePassword • ConfirmForgotPassword • ForgotPassword • AdminResetUserPassword • AdminSetUserPassword • RespondToAuthChallenge[1] • AdminRespondToAuthChallenge[1] • 托管 UI 密码重置	恢复用户账户或者更改或更新用户密码的操作。	30	否
UserRead • AdminGetUser • GetUser	从用户池中检索用户的操作。	120	是

类别	描述	默认配额（RPS）	可调整
UserUpdate • AdminAddUserToGroup • AdminDeleteUserAttributes • AdminUpdateUserAttributes • AdminDeleteUser • AdminDisableUser • AdminEnableUser • AdminLinkProviderForUser • AdminDisableProviderForUser • VerifyUserAttribute • DeleteUser • DeleteUserAttributes • UpdateUserAttributes • AdminUserGlobalSignOut • GlobalSignOut • AdminRemoveUserFromGroup	用于管理用户和用户属性的操作。	25	否
UserToken • RevokeToken	令牌管理的操作	120	是

类别	描述	默认配额（RPS）	可调整
`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	可以

类别	描述	默认配额（RPS）	可调整
`UserResourceUpdate` • AdminForgetDevice • AdminUpdateAuthEventFeedback • AdminSetUsermfapReferen • AdminSetUserSettings • AdminUpdateDeviceStatus • UpdateDeviceStatus • UpdateAuthEventFeedback • ConfirmDevice • SetUsermfapReferen • 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
`UserList` • ListUsers • ListUsersInGroup	返回用户列表的操作。	30	否

类别	描述	默认配额（RPS）	可调整
`UserPoolRead` • DescribeUserPool • ListUserPools	读取用户池的操作。	15	否
`UserPoolUpdate` • CreateUserPool • UpdateUserPool • DeleteUserPool	创建、更新或删除用户池的操作。	15	否

类别	描述	默认配额（RPS）	可调整
`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。[3]	20	否

类别	描述	默认配额（RPS）	可调整
`UserPoolResourceUpdate` • AddCustomAttributes • CreateGroup • CreateIdentityProvider • CreateResourceServer • CreateUserImportJob • CreateUserPoolDomain • DeleteGroup • DeleteIdentityProvider • DeleteResourceServer • DeleteUserPoolDomain • SetUserPoolMfaConfig • StartUserImportJob • StopUserImportJob • UpdateGroup • UpdateIdentityProvider • UpdateResourceServer • UpdateUserPoolDomain	修改用户池中的资源（如组或资源服务器）的操作。[3]	15	否

类别	描述	默认配额（RPS）	可调整
• SetRiskConfiguration • SetUICustomization • TagResource • UntagResource			
UserPoolClientRead • DescribeUserPoolClient • ListUserPoolClients	检索用户池客户端的相关信息的操作。[3]	15	否
UserPoolClientUpdate • CreateUserPoolClient • DeleteUserPoolClient • UpdateUserPoolClient	创建、更新和删除用户池客户端的操作。[3]	15	否
ClientAuthentication client_credentials 向令牌端点发出授权类型请求。	生成用于授权 machine-to-machine 请求的凭证的操作	150	否

[1] A RespondToAuthChallenge 或 a ChallengeName 的AdminRespondToAuthChallenge回复NEW_PASSWORD_REQUIRED计入该UserAccountRecovery类别。所有其他质疑回复都计入该UserAuthentication类别。

[2] 登录期间的每个托管用户界面操作都会为配额提供一个请求。例如，登录并提供 MFA 代码的用户贡献 2 个请求。授权码授予中的代币兑换需要额外的配额分配，比例与您在该类别中的配额相同。`UserAuthentication`

[3] 此类别中的任何单个操作都有一个约束条件，可以防止在单个用户池中以高于 5 RPS 的速率调用该操作。

Amazon Cognito 身份池（联合身份）API 操作请求速率配额

操作	描述	默认配额（RPS）[1]	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`	在增强的工作流程中从 AWS 身份池中检索证书。	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	可以	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。

操作	描述	默认配额（RPS）[1]	可调整	提高配额的资格
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。

[1] 默认配额是您的任意 AWS 区域 身份池中身份池的最低请求速率配额 AWS 账户。在某些区域，您的 RPS 配额可能会更高。

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特征的最大数量或大小。

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**
>
> AWS 账户 级别的资源配额（如每个区域的用户池）适用于每个区域中的 Amazon Cognito 资源。 AWS 区域例如，您在美国东部（弗吉尼亚州北部）中有 1000 个用户池，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	限额	可调整	最大配额
每个用户池的应用程序端	1000	是	10000
每个区域的用户池数	1000	是	10000

资源	限额	可调整	最大配额
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	40,000,000	是	请联系您的账户团队。
令牌生成前 Lambda 触发器中的合并更改总数[1]	5000	是	请联系您的账户团队。
每个用户池的自定义属性	50	不可以	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
每人每天发送的电子邮件 AWS 账户[2]	50	不可以	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

资源	限额	可调整	最大配额
身份提供商名称的字符数	32	否	不适用
每个身份提供商的标识符数	50	不可以	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	不可以	不适用
每个账户的自定义域	4	否	不适用
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] 来自 令牌生成前 Lambda 触发器 的令牌可能会遇到此限额。访问令牌和身份令牌中现有和已添加的声明数量以及范围数量加起来必须小于或等于此限额。被隐藏的声明和范围不计入此限额。

[2] 仅当您使用 Amazon Cognito 用户池的默认电子邮件功能时，此限额才适用。要提高电子邮件发送量，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。

Amazon Cognito 用户池会话验证参数

令牌	限额
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 用户池代码安全资源限额（不可调整）

资源	限额
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
每位用户每小时的最大 ConfirmForgotPassword 和 ForgotPassword 请求数[1]	5–20
每位用户每小时的最大 ResendConfirmationCode 请求数	5
每位用户每小时的最大 ConfirmSignUp 请求数	15
每位用户每小时的最大 ChangePassword 请求数	5
每位用户每小时的最大 GetUserAttributeVerificationCode 请求数	5

资源	限额
每位用户每小时的最大 VerifyUserAttribute 请求数	15

[1] Amazon Cognito 会评估更新密码请求中的风险因素，然后分配一个与评估的风险等级相关的配额。有关更多信息，请参阅 忘记密码行为。

Amazon Cognito 用户池用户导入任务资源限额

资源	限额	可调整	最大配额
每个用户池的用户导入任务	1000	是	请联系您的账户团队。
每个用户导入 CSV 行的最大字符数	16000	否	不适用
最大 CSV 文件大小	100 MB	否	不适用
每个 CSV 文件的最大用户数	500,000	否	不适用

Amazon Cognito 身份池（联合身份）资源配额

资源	限额	可调整	最大配额
每个账户的身份池	1000	是	不适用
每个身份池的 Amazon Cognito 用户池提供商数	50	可以	1000
身份池名称的字符长度	128 字节	否	不适用

资源	限额	可调整	最大配额
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	无限制	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	限额	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	1 MB	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 用户池 API 和用户池端点。

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

- 用户池联合身份验证端点和托管 UI 参考
- Amazon Cognito 用户池 API 参考
- Amazon Cognito 身份池（联合身份）API 参考
- Amazon Cognito Sync API 参考

用户池联合身份验证端点和托管 UI 参考

当您向用户池分配域时，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）和注销（注销端点)。有关这些资源的标签的更多信息，请参阅 设置和使用 Amazon Cognito 托管 UI 和联合身份验证端点。

这些页面还包括公共网络资源，允许您的用户池与第三方 SAML、OpenID Connect (OIDC) 和 OAuth 2.0 身份提供商 () 进行通信。IdPs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 登录端点或 OIDC 对端点授权发起请求。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 用户池 API 登录本地用户。本地用户仅存在于您的用户池目录中，无需通过外部 IdP 进行联合身份验证。

除了托管用户界面和联合终端节点外，Amazon Cognito 还集成了适用于安卓、iOS 等的软件开发工具包。 JavaScript这些 SDK 提供了使用 Amazon Cognito API 服务端点执行用户池 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**
>
> 不要锁定 Amazon Cognito 域的终端实体或中间传输层安全 (TLS) 证书。 AWS 管理所有用户池终端节点和前缀域的所有证书。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，您的应用程序在 AWS 轮换证书时可能会失败，恕不另行通知。
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

当您向用户池添加域时，Amazon Cognito 会激活本节中的托管 UI 端点。它们是您的用户可以在其中完成用户池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。

构成托管 UI 的网页是一个前端 Web 应用程序，用于与客户进行交互式用户会话。您的应用程序必须在用户的浏览器中调用托管 UI。Amazon Cognito 不支持以编程方式访问本章中的网页。可以在应用程序代码中直接查询 OAuth 2.0、OpenID Connect 和 SAML 2.0 联合身份验证端点参考 中那些返回 JSON 响应的联合端点。对端点授权 重定向到托管 UI 或 IdP 登录页面，并且还必须在用户的浏览器中打开。

本指南中的主题详细描述了常用的托管 UI 端点。Amazon Cognito 会在您为用户池分配域时提供以下网页。

托管 UI 端点

端点 URL	描述	如何访问此端点
https://######/login	登录用户池本地用户和联合用户。	从端点重定向，如对端点授权、`/logout` 和 `/forgotPassword` 。请参阅登录端点。

端点 URL	描述	如何访问此端点
https://######/logout	注销用户池用户。	直接链接。请参阅 注销端点。
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。当您的用户选择注册时，/login 页面会将他们定向到 /signup。	与 /oauth2/authorize 具有相同参数的直接链接。
https://######/confirm	在用户池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 /oauth2/authorize 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	向已在用户池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。

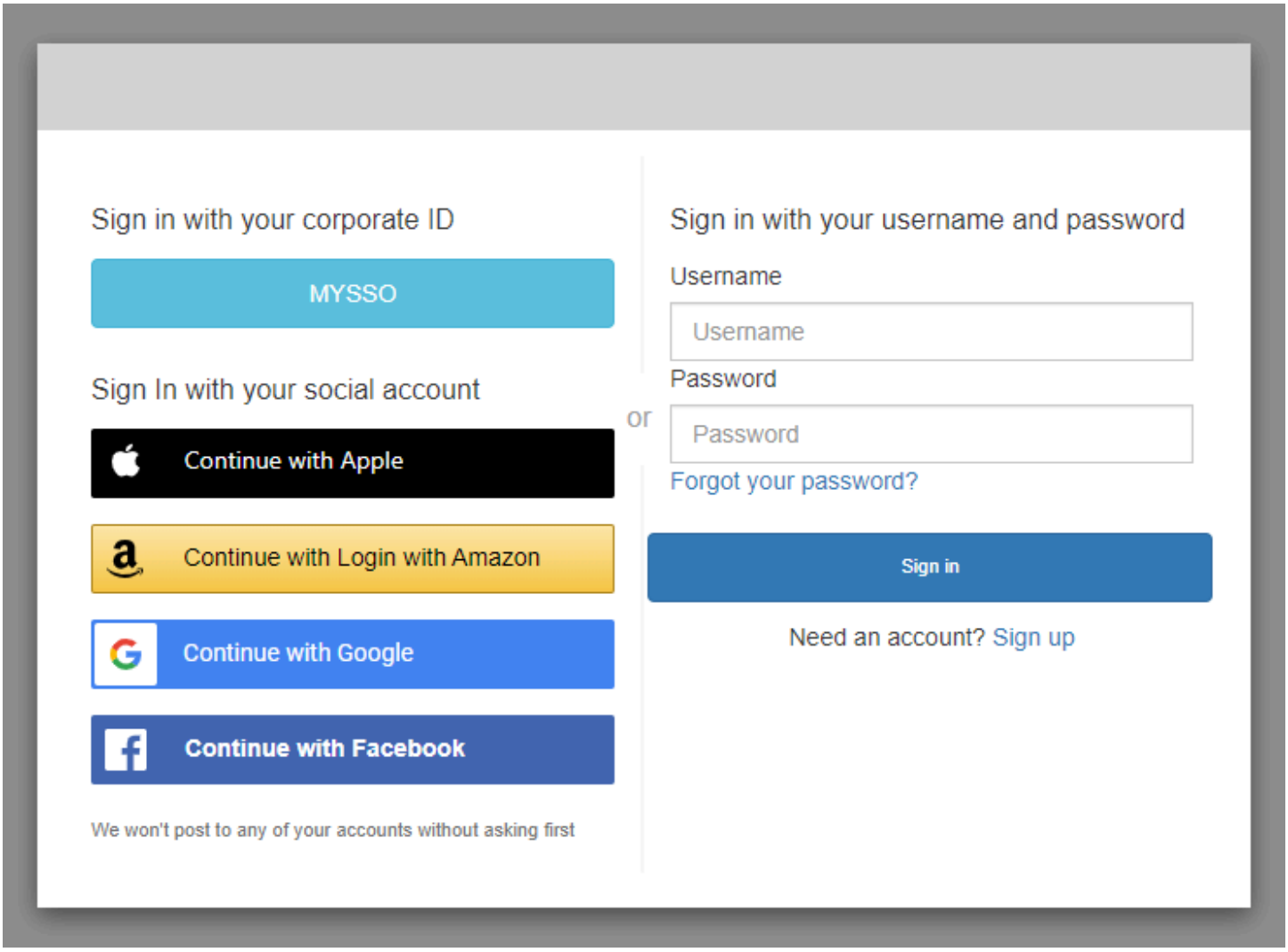
主题

- 登录端点
- 注销端点

登录端点

登录端点是一个身份验证服务器，也是来自对端点授权的重定向目的地。当您不指定身份提供者时，它是托管 UI 的入口点。当您生成到登录端点的重定向时，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。

> **Note**
>
> 登录端点是托管 UI 的组成部分。在您的应用程序中，调用重定向到登录端点的联合身份验证和托管 UI 页面。用户直接访问登录端点并不是最佳实践。



GET /login

/login 端点仅支持对用户的初始请求执行 HTTPS GET。您的应用程序会在 Chrome 或 Firefox 等浏览器中调用该页面。当您/login从重定向到时对端点授权，它会传递您在初始请求中提供的所有参数。登录端点支持授权端点的所有请求参数。您也可以直接访问登录端点。作为最佳实践，应在 /oauth2/authorize 上发起所有用户会话。

示例-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。它会注销用户并重定向到您的应用程序客户端的授权注销网址或终端节点。/login/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的后续请求的参数，例如 `client_id`、`response_type` 和 `scope`。

注销端点是一个前端 Web 应用程序，用于与客户进行交互式用户会话。您的应用程序必须在用户的浏览器中调用此端点和其他托管 UI 端点。

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。

使用 SAML 2.0 的单点注销 (SLO)，Amazon IdPs Cognito 首先将您的用户重定向到您在 IdP 配置中定义的 SLO 终端节点。在您的 IdP 将您的用户重定向回之后，Amazon `saml2/logout` Cognito 会再重定向到您的请求或来自您的请求。`redirect_uri` `logout_uri`有关更多信息，请参阅 SAML 注销流程。

注销端点不会让用户退出 OIDC 或社交身份提供商 ()。IdPs要让用户退出与外部 IdP 的会话，请将他们引导到该提供商的注销页面。

GET /logout

`/logout` 端点只支持 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 用户池应用程序客户端。

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 用户池应用程序客户端。

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允许的回调 URL。在对 `/logout` 端点的请求中，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 端点的请求中使用 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。如果您对 /logout 端点的请求中同时包含 logout_uri 和 redirect_uri 参数，Amazon Cognito 将仅使用 logout_uri 参数，不使用 redirect_uri 参数。

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 端点的请求中使用 logout_uri 或 redirect_uri 参数。要将您的用户重定向到您的/login终端节点以重新进行身份验证并将令牌传递给您的应用程序，请添加 red irect _uri 参数。如果您的终端节点请求中同时包含 logout_uri 和 redirect_uri 参数，则 /logout Amazon Cognito 会覆盖 redirect_uri 参数并专门处理 logout_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

当您的应用程序向请求添加状态参数时，当/oauth2/logout终端节点重定向您的用户时，Amazon Cognito 会将其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。

范围

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 端点的请求中的 scope 参数将您的用户重定向到 /login 端点。

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 端点。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。

请求示例

示例-注销并将用户重定向到客户端

除logout_uri和之外client_id，此端点的所有可能的查询参数都将传递到对端点授权。当请求包含 logout_uri 和 client_id 时，Amazon Cognito 会将用户会话重定向到 logout_uri 值中的 URL，忽略所有其他请求参数。这个 URL 必须是应用程序客户端的授权注销 URL。

以下是注销并重定向到 https://www.example.com/welcome 的请求示例。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?
  client_id=1example23456789&
  logout_uri=https%3A%2F%2Fwww.example.com%2Fwelcome
```

示例-注销并提示用户以其他用户身份登录

如果请求忽略 logout_uri，但以其他方式提供了参数，可以构成对授权端点发出的、格式正确的请求，Amazon Cognito 会将用户重定向到托管 UI 登录。注销端点会将原始请求中的参数附加到重定向目的地。面向注销端点的请求中的参数 redirect_uri 不是注销 URL，而是您要传递给授权端点的登录 URL。

以下是用户注销、重定向到登录页面并在登录https://www.example.com后向其提供授权码的请求示例。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?
  response_type=code&
  client_id=1example23456789&
  redirect_uri=https%3A%2F%2Fwww.example.com&
  state=example-state-value&
  nonce=example-nonce-value&
  scope=openid+profile+aws.cognito.signin.user.admin
```

OAuth 2.0、OpenID Connect 和 SAML 2.0 联合身份验证端点参考

当您向用户池添加域时，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。他们为您的应用程序扮演服务角色，以便与第三方 OAuth 2.0、OIDC 和 SAML 2.0 身份提供商（ ）进行通信。IdPs

本指南中的主题描述几个常用的 OAuth 2.0 和 OIDC 端点。您向用户池分配域时，Amazon Cognito 创建以下端点。

用户池联合身份验证端点

端点 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。	在客户浏览器中调用以开始用户身份验证。请参阅 对端点授权。
https://######/oauth2/token	根据授权码或客户端凭证请求返回令牌。	由应用程序请求检索令牌。请参阅 令牌端点。
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。	应用程序要求检索用户个人资料。请参阅 UserInfo 端点。
https://######/oauth2/revoke	撤消刷新令牌和关联的访问令牌。	应用程序要求撤销令牌。请参阅 撤消端点。
https://cognito-idp.*Region*.amazonaws.com/*your user pool ID*/.well-known/openid-configuration	用户池的 OIDC 架构的目录。	由应用程序请求查找用户池发行者元数据。
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。	由应用程序请求验证 JWT。
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	用于与 SAML 2.0 身份提供商集成的断言消费者响应 (ACS) 网址。	从 SAML 2.0 IdP 重定向为 ACS 网址，或 IdP 发起的登录的起点。[1]

端点 URL	描述	如何访问此端点
https://###### /saml2/logout	用于与 SAML 2.0 身份提供商集成的单点注销 (SLO) 网址。	从 SAML 2.0 IdP 重定向为单点注销 (SLO) 网址。仅接受 POST 绑定。

[1] 有关 IDP 发起的 SAML 登录的更多信息，请参阅。使用 IDP 发起的 SAML 登录

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
- 令牌端点
- UserInfo 端点
- 撤消端点
- saml2/idpresponse 端点

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。

授权端点将会重定向到托管 UI 或 IdP 登录页面。此端点上的用户会话的目的地是您的用户必须直接在其浏览器中与之交互的网页。

要使用 authorize 端点，请在 `/oauth2/authorize` 上使用为您的用户池提供有关以下用户池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` 端点只支持 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）标准的授权端点中详细了解授权端点的定义。

请求参数

`response_type`

（必填）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。

`client_id`

（必填）应用程序客户端 ID。

`client_id` 的值必须是您在其中发出请求的用户池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。

`redirect_uri`

（必填）Amazon Cognito 对用户进行授权后，身份验证服务器重定向浏览器的 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`。

`state`

（可选，推荐）当您的应用程序向请求添加状态参数时，当`/oauth2/authorize`终端节点重定向您的用户时，Amazon Cognito 会将其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，然后在您的应用程序中对其进行解码。

`identity_provider`

（可选）添加此参数可绕过托管界面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。

- 对于社交提供者，您可以使用 identity_provider 值`Facebook`、`Google`、`LoginWithAmazon`和。`SignInWithApple`
- 对于 Amazon Cognito 用户池，请使用该值。`COGNITO`
- 对于 SAML 2.0 和 OpenID Connect (OIDC) 身份提供商 (IdPs)，请使用您在用户池中分配给 IdP 的名称。

`idp_identifier`

（可选）添加此参数以重定向到具有替代名称的 id entity_p rovider 名称的提供商。您可以 IdPs 从 Amazon Cognito 控制台的登录体验选项卡中输入 SAML 2.0 和 OIDC 的标识符。

`scope`

（可选）可以是任何系统保留的作用域或与客户端关联的自定义作用域的组合。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。

`code_challenge_method`

（可选）您用来生成质询的哈希协议。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。

code_challenge

（可选）您从中生成的挑战code_verifier。

仅当您指定 code_challenge_method 参数时是必需的。

nonce

（可选）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）。

回复为正面的请求示例

以下示例说明了向/oauth2/authorize终端节点发出 HTTP 请求的格式。

授予授权代码

这是授权码授权请求的示例。

示例-GET 请求

以下请求启动会话，以检索您的用户在redirect_uri目的地传递给您的应用程序的授权码。此会话请求用户属性的范围以及对 Amazon Cognito 自助服务 API 操作的访问权限。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=openid+profile+aws.cognito.signin.user.admin
```

示例-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。授权码的有效期为五分钟。

```
HTTP/1.1 302 Found
Location: https://www.example.com?code=a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111&state=abcdefg
```

具有 PKCE 的授权代码授予

这是向 P KCE 授予授权码的请求示例。

示例-GET 请求

以下请求为上一个请求添加了一个code_challenge参数。要完成将代码兑换为令牌的过程，您必须在向/oauth2/token终端节点发出的请求中包含该code_verifier参数。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin&
code_challenge_method=S256&
code_challenge=a1b2c3d4...
```

示例-响应

身份验证服务器使用授权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，而不是在片段中返回：

```
HTTP/1.1 302 Found
Location: https://www.example.com?code=a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111&state=abcdefg
```

不带 **openid** 范围的令牌授予

这是一个生成隐式授权并将 JWT 直接返回到用户会话的示例请求。

示例-GET 请求

以下请求是请求您的授权服务器进行隐式授权。来自亚马逊 Cognito 的访问令牌授权自助服务 API 操作。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=token&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin
```

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 在片段中而不是查询字符串中返回访问令牌和状态：

```
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

这是一个生成隐式授权并将 JWT 直接返回到用户会话的示例请求。

示例-GET 请求

以下请求是请求您的授权服务器进行隐式授权。来自 Amazon Cognito 的访问令牌授权访问用户属性和自助服务 API 操作。

```
GET
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=token&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin+openid+profile
```

示例-响应

授权服务器使用访问令牌和 ID 令牌重定向回您的应用程序（因为已包含openid范围）：

```
HTTP/1.1 302 Found
Location: https://
www.example.com#id_token=eyJra67890EXAMPLE&access_token=eyJra12345EXAMPLE&token_type=bearer&exp
```

负向响应的示例

亚马逊 Cognito 可能会拒绝您的请求。否定请求附带一个 HTTP 错误代码和描述，您可以用它来更正请求参数。以下是负面回应的示例。

- 如果client_id和redirect_uri有效，但请求参数的格式不正确，则身份验证服务器会将错误重定向到客户端，redirect_uri并在 URL 参数中附加一条错误消息。以下是格式不正确的示例。
  - 该请求不包含response_type参数。
  - 授权请求提供了一个code_challenge参数，但没有提供code_challenge_method参数。
  - code_challenge_method参数的值不是S256。

  以下是对格式不正确的示例请求的响应。

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_request
```

- 如果客户端请求code或token进入response_type，但无权处理这些请求，Amazon Cognito 授权服务器unauthorized_client将返回到客户端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
```

- 如果服务器中出现任何意外错误，则身份验证服务器server_error将返回到客户端redirect_uri。由于 HTTP 500 错误不会发送到客户端，因此该错误不会显示在用户的浏览器中。授权服务器返回以下错误。

```
HTTP 1.1 302 Found Location: https://client_redirect_uri?error=server_error
```

- 当 Amazon Cognito 通过联合身份验证向第三方 IdPs进行身份验证时，Amazon Cognito 可能会遇到连接问题，例如：
  - 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_uri端点进行 ID 令牌验证时出现连接超时，则身份验证服务器会以错误重定向到客户端，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
```

- 通过联合第三方进行身份验证时 IdPs，提供商可能会返回错误响应。这可能是由于配置错误或其他原因造成的，例如：
  - 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=Google+Error+-+[status code]+[Google-
    provided error code]
    ```

- 当 Amazon Cognito 在连接到外部 IdP 时遇到通信异常时，身份验证服务器会以错误重定向到客户端，并显示以下任`redirect_uri`一消息：
  - ```
    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

`/oauth2/token` 处的 OAuth 2.0 令牌端点会发放 JSON 网络令牌（JWT）。

您的用户池 OAuth 2.0 授权服务器从令牌端点向以下类型的会话发放 JSON 网络令牌 (JWT)：

1. 完成授权码请求的用户授权。成功兑换代码会返回 ID、访问令牌和刷新令牌。
2. 已完成客户端凭证授予的 M achine-to-machine (M2M) 会话。成功使用客户端机密进行授权会返回访问令牌。
3. 之前登录并收到刷新令牌的用户。刷新令牌身份验证会返回新的 ID 和访问令牌。

> **Note**
>
> 在托管用户界面中使用授权码授权或通过联合身份登录的用户可以随时从令牌端点刷新其令牌。使用 API 操作InitiateAuth登录的用户，当记住的设备在您的用户池中未处于活动状态时，AdminInitiateAuth可以使用令牌端点刷新令牌。如果记住的设备处于活动状态，请使用 of REFRESH_TOKEN_AUTH in AuthFlow InitiateAuth 或 AdminInitiateAuth API 请求刷新令牌。

当您向用户池添加域时，令牌端点将公开可用。它接受 HTTP POST 请求。为了确保应用程序安全，请将 PKCE 与您的授权码登录事件一起使用。PKCE 会验证传递授权码的用户是否与经过身份验证的用户相同。有关 PKCE 的更多信息，请参阅 IETF RF C 7636。

要详细了解用户池应用程序客户端及其授权类型、客户端机密、授权范围和客户端 ID，请访问用户池应用程序客户端。要详细了解 M2M 授权、客户凭证授予和访问令牌范围的授权，请访问。使用资源服务器进行范围、M2M 和 API 授权

要从用户的访问令牌中检索有关用户的信息，请将其传递给您的UserInfo 端点或 GetUserAPI 请求。

POST /oauth2/token

/oauth2/token 端点只支持 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djc98u3jiedmi283eu928带有客户端密钥的应用程序客户端的授权标头abcdef01234567890，使用了 Base64 编码版本的字符串：djc98u3jiedmi283eu928:abcdef01234567890

```
Authorization: Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw
```

Content-Type

将此参数的值设置为 'application/x-www-form-urlencoded'。

正文中的请求参数

grant_type

（必填）您要申请的 OIDC 拨款类型。

必须为 authorization_code、refresh_token 或 client_credentials。在以下条件下，您可以从令牌端点请求自定义范围的访问令牌：

- 您在应用程序客户端配置中启用了请求的范围。
- 您使用客户端密钥配置了应用程序客户端。
- 您可以在应用程序客户端中启用客户端凭据授权。

client_id

（可选）用户池中应用程序客户端的 ID。指定对您的用户进行身份验证的同一个应用程序客户端。

如果客户端是公共客户端且没有密钥或已client_secret_post获得授权，则必须提供此参数。client_secret

client_secret

（可选）对您的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

（可选）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，亚马逊 Cognito 将忽略它。如果客户端未请求任何范围，则身份验证服务器会分配您在应用程序客户端配置中授权的所有自定义范围。

仅当 grant_type 为 client_credentials 时使用。

redirect_uri

（可选）必须与进入authorization_code时redirect_uri使用的相同/oauth2/authorize。

如果grant_type是，则必须提供此参数authorization_code。

refresh_token

（可选）要为用户的会话生成新的访问权限和 ID 令牌，请将/oauth2/token请求中的refresh_token参数值设置为先前从同一应用程序客户端发放的刷新令牌。

code

（可选）授权码授予的授权码。如果您的授权请求包含，则必须提供此参数authorization_code。grant_type

code_verifier

（可选）您在 PKCE 的授权码授权请求code_challenge中用来计算的任意值。

得到正面回复的请求示例

为获取令牌交换授权代码

示例-POST 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，令牌端点才返回 refresh_token。

用客户端凭证交换访问令牌：授权标头中的客户端密钥

示例-POST 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
                            }
```

用客户端凭证交换访问令牌：请求正文中的客户端密钥

示例-POST 请求

```
POST /oauth2/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
X-Amz-Target: AWSCognitoIdentityProviderService.Client credentials request
User-Agent: USER_AGENT
```

```
Accept: /
Accept-Encoding: gzip, deflate, br
Content-Length: 177
Referer: http://auth.example.com/oauth2/token
Host: auth.example.com
Connection: keep-alive

grant_type=client_credentials&client_id=1example23456789&scope=my_resource_server_identifier%2F
```

示例-响应

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Tue, 05 Dec 2023 16:11:11 GMT
x-amz-cognito-request-id: 829f4fe2-a1ee-476e-b834-5cd85c03373b

{
"access_token": "eyJra12345EXAMPLE",
"expires_in": 3600,
"token_type": "Bearer"
}
```

为获取令牌交换具有 PKCE 的授权代码授予

示例-POST 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，令牌端点才返回 refresh_token。

为获取令牌交换刷新令牌

示例-POST 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，令牌端点才返回 refresh_token。

负向响应的示例

示例-错误响应

```
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 端点

`userInfo` 端点是 OpenID Connect（OIDC）userInfo 端点。当服务提供商出示您的 令牌端点 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池 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 终端节点交换授权码。`token`您的用户池传递 IdP 访问令牌以授权从 IdP 端点检索用户信息。`userInfo`

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
- 示例-请求
- 示例 — 正面回应
- 负面回应示例

标头中的请求参数

`Authorization: Bearer <access_token>`

在授权标头字段中传递访问令牌。

必需。

示例-请求

```
GET /oauth2/userInfo HTTP/1.1
Content-Type: application/x-amz-json-1.1
Authorization: Bearer eyJra12345EXAMPLE
```

```
User-Agent: [User agent]
Accept: */*
Host: auth.example.com
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

示例 — 正面回应

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: [Integer]
Date: [Timestamp]
x-amz-cognito-request-id: [UUID]
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
Server: Server
Connection: keep-alive
{
    "sub": "[UUID]",
    "email_verified": "true",
    "custom:mycustom1": "CustomValue",
    "phone_number_verified": "true",
    "phone_number": "+12065551212",
    "email": "bob@example.com",
    "username": "bob"
}
```

有关 OIDC 声明的列表，请参阅标准声明。目前，Amazon Cognito 将 `email_verified` 和 `phone_number_verified` 的值返回为字符串。

负面回应示例

示例-错误的请求

```
HTTP/1.1 400 Bad Request
WWW-Authenticate: error="invalid_request",
error_description="Bad OAuth2 request at UserInfo Endpoint"
```

invalid_request

请求缺少必填参数，包含不支持的参数值，或者格式不正确。

示例-不良代币

```
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、格式错误或无效。

撤消端点

/oauth2/revoke终端节点会撤销 Amazon Cognito 最初与您提供的刷新令牌一起发布的用户访问令牌。此端点还会从同一个刷新令牌中撤消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 端点只支持 HTTPS POST。用户池客户端直接对此端点发出请求，而不通过系统浏览器。

标头中的请求参数

Authorization

如果您的应用程序客户端有客户端密钥，则应用程序必须通过基本 HTTP 授权client_secret在授权标头中传递其client_id和。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

（必需）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

（可选）您要撤销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。

端点错误

- 如果令牌在请求中不存在或者应用程序客户端禁用了该特征，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。

saml2/idpresponse 端点

接`/saml2/idpresponse`收 SAML 断言。在 service-provider-initiated （SP 发起的）登录中，您的 SAML 2.0 身份提供商 (IdP) 会使用其 SAML 响应将您的用户重定向到此端点。在 SP 启动的登录中，您的应用程序不会与该端点交互。使用您的路径配置您的 IdP `saml2/idpresponse` 作为断言使用者服务 (ACS) URL。有关会话启动的更多信息，请参阅SAML 会话在 Amazon Cognito 用户池中启动。

在 IdP 发起的登录中，您的用户可以通过您自己的流程使用您的 IdP 登录，并通过 HTTPS 在请求正文中提交 SAML 断言。`HTTP POSTPOST`请求的正文必须是`SAMLResponse`参数和`Relaystate`参数。有关更多信息，请参阅 使用 IDP 发起的 SAML 登录。

帖子 **`/saml2/idpresponse`**

要在 IdP 发起的登录中使用`/saml2/idpresponse`终端节点，请生成一个 POST 请求，其参数可为您的用户池提供有关您的用户会话的信息。

- 他们要登录的应用程序客户端。
- 他们想要最终到达的回传网址。
- 他们想要在用户的访问令牌中请求的 OAuth 2.0 范围。
- 发起登录请求的 IdP。

IDP 发起的请求正文参数

样本响应

来自 IdP 的 Base64 编码的 SAML 断言，该断言与您的用户池中的有效应用程序客户端和 IdP 配置相关联。

RelayState

`RelayState`参数包含您原本要传递到`oauth2/authorize`终端节点的请求参数。有关这些参数的详细信息，请参见对端点授权。

response_type

OAuth 2.0 授权类型。

client_id

应用程序客户端 ID。

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。

identity_provider

您要将用户重定向到的身份提供商的名称。

idp_identifier

您要将用户重定向到的身份提供商的标识符。

范围

您希望用户向授权服务器请求的 OAuth 2.0 范围。

回复为正面的请求示例

示例-POST 请求

以下请求是在应用程序客户端中向来自 IdP `MySAMLIdP` 的用户授予授权码。`1example23456789`用户使用其授权码重定向到`https://www.example.com`，该授权码可以兑换包含具有 OAuth 2.0 范围的访问令牌的令牌`openid`、`email`和。`phone`

```
POST /saml2/idpresponse HTTP/1.1
User-Agent: USER_AGENT
Accept: */*
Host: example.auth.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded

SAMLResponse=[Base64-encoded SAML assertion]&RelayState=identity_provider
%3DMySAMLIdP%26client_id%3D1example23456789%26redirect_uri%3Dhttps%3A%2F
%2Fwww.example.com%26response_type%3Dcode%26scope%3Demail%2Bopenid%2Bphone
```

示例-响应

以下是对先前请求的回应。

```
HTTP/1.1 302 Found
Date: Wed, 06 Dec 2023 00:15:29 GMT
Content-Length: 0
x-amz-cognito-request-id: 8aba6eb5-fb54-4bc6-9368-c3878434f0fb
Location: https://www.example.com?code=[Authorization code]
```

OAuth 2.0 授予

Amazon Cognito 用户池 OAuth 2.0 授权服务器发布令牌，以响应三种类型的 OAuth 2.0 授权授予。您可以为用户池中的每个应用程序客户端设置支持的授权类型。您不能在同一应用程序客户端中启用客户端凭证授予作为隐式 或授权码 授予。隐式和授权码授予的请求从对端点授权开始，而客户端凭证授予的请求从令牌端点开始。

授予授权代码

为了响应您的成功身份验证请求，授权服务器会在回调 URL 的 code 参数中添加一个授权码。然后，您必须与令牌端点交换 ID、访问令牌和刷新令牌的代码。要请求授权码授予，请在请求中将 response_type 设置为 code。有关示例请求，请参阅授予授权代码。

授权码授予是最安全的授权授予形式。它不会直接向您的用户显示令牌内容。相反，您的应用程序负责检索和安全存储用户的令牌。在 Amazon Cognito 中，授权码授予是从授权服务器获取所有三种令牌类型（ID、访问和刷新）的唯一方法。您也可以通过 Amazon Cognito 用户池 API 从身份验证中获取所有三种令牌类型，但是 API 不使用除 aws.cognito.signin.user.admin 以外的范围发布访问令牌。

隐式授予

为了响应成功的身份验证请求，授权服务器会在 access_token 参数中附加一个访问令牌，并在您的回调 URL 的 id_token 参数中附加一个 ID 令牌。隐式授予不要求与令牌端点进行额外的交互。要请求隐式授予，请在请求中将 response_type 设置为 token。隐式授予仅生成 ID 和访问令牌。有关示例请求，请参阅不带 openid 范围的令牌授予。

隐式授予是旧版授权授予。与授权码授予不同，用户可以拦截和检查您的令牌。要防止通过隐式授予交付令牌，请将您的应用程序客户端配置为仅支持授权码授予。

客户端凭证

客户凭证是一种仅限授权的访问授权。 machine-to-machine要获得客户凭证授予，请绕过对端点授权，并直接向令牌端点生成请求。您的应用程序客户端必须具有客户端密钥，并且仅支持客户端凭证授予。为了响应您的成功请求，授权服务器会返回访问令牌。

来自客户端凭证授予的访问令牌是一种包含 OAuth 2.0 范围的授权机制。通常，令牌包含自定义范围声明，这些声明授权 HTTP 操作访问那些访问受保护的 API。有关更多信息，请参阅 使用资源服务器进行范围、M2M 和 API 授权。

客户凭证授予会增加您的 AWS 账单费用。有关更多信息，请参阅 Amazon Cognito 定价。

在授权码中使用 PKCE 可授予 Amazon Cognito 用户池的权限

Amazon Cognito 在授权码授权中支持代码交换证明密钥 (PKCE) 身份验证。PKCE 是 OAuth 2.0 授权码授予公共客户端的扩展。PKCE 防范被拦截的授权码兑换。

亚马逊 Cognito 如何使用 PKCE

要开始使用 PKCE 进行身份验证，您的应用程序必须生成一个唯一的字符串值。此字符串是代码验证器，Amazon Cognito 使用该值将请求初始授权的客户端与使用授权码交换令牌的客户进行比较。

您的应用程序必须对代码验证器字符串应用 SHA256 哈希值，并将结果编码为 base64。将经过哈希处理的字符串对端点授权作为请求正文中的`code_challenge`参数传递给的。当您的应用程序将授权码交换为令牌时，它必须将纯文本的代码验证器字符串作为`code_verifier`参数包含在请求正文中。令牌端点Amazon Cognito 对代码验证器执行相同的 hash-and-encode 操作。Amazon Cognito 仅在确定代码验证器产生的代码质询与授权请求中收到的代码质询相同，才会返回 ID、访问和刷新令牌。

使用 PKCE 实现授权授予流程

1. 打开 Amazon Cognito 控制台。如果出现提示，请输入您的 AWS 凭据。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。如果您创建了用户池，则在向导期间，系统将提示您设置应用程序客户端并配置托管用户界面。

   a. 如果您创建了新的用户池，请在引导式设置期间设置应用程序客户端并配置托管用户界面。

   b. 如果您配置现有用户池，请添加域和公共应用程序客户端（如果尚未配置）。
4. 生成一个随机的字母数字字符串，通常是通用唯一标识符 (UUID)，以便为 PKCE 创建代码挑战。此字符串是您将在向的请求中提交的`code_verifier`参数的值令牌端点。
5. 使用 SHA256 算法对`code_verifier`字符串进行哈希处理。将哈希操作的结果编码为 base64。此字符串是您将在向的请求中提交的`code_challenge`参数的值对端点授权。

   以下Python示例生成 a `code_verifier` 并计算`code_challenge`：

```
#!/usr/bin/env python3

import random
from base64 import urlsafe_b64encode
from hashlib import sha256
from string import ascii_letters
from string import digits

# use a cryptographically strong random number generator source
rand = random.SystemRandom()

code_verifier = ''.join(rand.choices(ascii_letters + digits, k=128))
code_verifier_hash = sha256(code_verifier.encode()).digest()
code_challenge = urlsafe_b64encode(code_verifier_hash).decode().rstrip('=')

print(f"code challenge: {code_challenge}")
print(f"code verifier: {code_verifier}")
```

以下是Python脚本的输出示例：

```
code challenge: Eh0mg-OZv7BAyo-tdv_vYamx1boOYDulDklyXoMDtLg
code verifier: 9D-aW_iygXrgQcWJd0y0tNVMPSXSChIc2xceDhvYVdGLCBk-
JWFTmBNjvKSdOrjTTYazOFbUmrFERrjWx6oKtK2b6z_x4_gHBDlr4K1mRFGyE8yA-05-_v7Dxf3EIYJH
```

6. 使用 PKCE 的授权码授权请求完成托管 UI 登录。以下是 URL 示例：

```
https://mydomain.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com&code_challenge=Eh0mg-OZv7BAyo-
tdv_vYamx1boOYDulDklyXoMDtLg&code_challenge_method=S256
```

7. 收集授权code并使用令牌端点将其兑换为令牌。以下是一个请求示例：

```
POST /oauth2/token HTTP/1.1
Host: mydomain.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 296

redirect_uri=https%3A%2F%2Fwww.example.com&
client_id=1example23456789&
code=7378f445-c87f-400c-855e-0297d072ff03&
```

```
grant_type=authorization_code&
code_verifier=9D-aW_iygXrgQcWJd0y0tNVMPSXSChIc2xceDhvYVdGLCBk-
JWFTmBNjvKSdOrjTTYazOFbUmrFERrjWx6oKtK2b6z_x4_gHBDlr4K1mRFGyE8yA-05-_v7Dxf3EIYJH
```

8. 查看回复。它将包含 ID、访问和刷新令牌。有关使用 Amazon Cognito 用户池令牌的更多信息，请参阅。将令牌与用户池结合使用

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池无法完成的操作。
- 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。

> **Note**
>
> Amazon Cognito 错误描述不是固定字符串，您不应使用依赖于固定模式或格式的逻辑。

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和将令牌与用户池结合使用。

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 用户池 API 和用户池端点。

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。

变更	说明	日期
在令牌前 Lambda 触发器中增加了对复杂对象的支持	现在，您可以将数组和 JSON 对象添加到 ID 和访问令牌声明中。	2024 年 5 月 30 日
更新了有关已验证权限和 Amazon Cognito 的信息。	亚马逊验证权限现在可以更直接地与亚马逊 Cognito 集成。	2024 年 5 月 15 日
经过多区域 Amazon SES 验证的身份。	在某些 AWS 区域 没有 Amazon SES 的情况下，Amazon Cognito 用户池在两个远程区域之间对电子邮件进行负载平衡。	2024年5月10日
添加了有关 M2M 授权和管理成本的信息。	了解如何在 Amazon Cognito 用户池中使用客户凭证授予 machine-to-machine (M2M) 用例。	2024 年 5 月 9 日
Amazon Cognito 现已在欧洲（西班牙）和亚太地区（海得拉巴）上市。 AWS 区域	现在，您可以在欧洲（西班牙）和亚太地区（海得拉巴）地区创建 Amazon Cognito 资源。	2024年4月15日
Amazon Cognito 现已在亚太地区（墨尔本）上市。 AWS 区域	现在，您可以在亚太地区（墨尔本）地区创建 Amazon Cognito 资源。	2024 年 4 月 4 日
在 Flutter 中为亚马逊 Cognito 用户池添加了一个安卓应用程序示例。	你可以通过上面的 Flutter 示例应用程序为亚马逊 Cognito 构建一款入门级移动应用程序。 GitHub	2024 年 4 月 4 日

新的入门内容	扩展了入门、常见场景、多租户最佳实践以及登录后访问资源的内容。	2024 年 4 月 1 日
亚马逊 Cognito 现已在欧洲（苏黎世）上市。 AWS 区域	现在，您可以在欧洲（苏黎世）地区创建 Amazon Cognito 资源。	2024 年 3 月 14 日
亚马逊 Cognito 现已在中东（阿联酋）上市。 AWS 区域	现在，您可以在中东（阿联酋）地区创建 Amazon Cognito 资源。	2024 年 3 月 8 日
新的 SAML 功能和改进的内容。	现在，您可以签署 SAML 请求、加密 SAML 响应并设置 IDP 启动的 SAML SSO。	2024年2月1日
可增加配额。	现在，您可以为 Amazon Cognito 请求速率配额购买更多容量。	2024 年 1 月 25 日
Amazon Cognito 身份池支持服务配额中的请求速率。	现在，您可以监控 Amazon Cognito 身份池的 requests-per-second (RPS) 配额，并在服务配额控制台中请求增加配额。	2023 年 12 月 19 日
添加了一项用于自定义访问令牌内容的新功能。	现在，您可以在用户池访问令牌中添加、修改和删除声明和范围。	2023 年 12 月 12 日
改进了有关应用程序客户端和 OAuth 范围的内容。	澄清对 用户池应用程序客户端 和 使用资源服务器进行范围、M2M 和 API 授权 所做的编辑和更正。删除了旧版控制台指令。	2023 年 11 月 14 日
改进了有关设备和设备身份验证的内容。	有关使用设备密钥和设备 SRP 身份验证的新内容。	2023 年 10 月 18 日

更新了 AWS Management Console 指南。	删除了用户池控制台参考以及相关主题中重新分发的主题，并为 Amazon Cognito 控制台中基于选项卡的整理方式添加了指南。	2023 年 8 月 30 日
不再强调直接访问 LOGIN 端点。	添加了用户池登录端点的直观概述，并强调了从对端点授权开始身份验证。	2023 年 8 月 30 日
Amazon Cognito 现已在亚太地区（大阪）和以色列（特拉维夫）上市。 AWS 区域	现在，您可以在亚太地区（大阪）和以色列（特拉维夫）地区创建 Amazon Cognito 资源。	2023 年 8 月 30 日
介绍了有关使用亚马逊验证权限授权 Amazon Cognito 的信息。	在您的应用程序中，您可以调用 Verified Permissions API 从中央机构生成访问决策。	2023 年 8 月 1 日
在 Amazon CloudWatch Logs 中添加了一项用于记录用户池详细用户活动的新功能。	现在，您可以将电子邮件和短信发送错误 CloudWatch 记录到日志组中。	2023 年 8 月 1 日
更新了有关身份池访客用户的 AWS 托管策略的信息。	身份池访客用户的权限范围现在包括内联会话策略和 AWS 托管会话策略。	2023 年 5 月 16 日
Amazon Cognito 身份池的内容改进和新的控制台说明。	添加了新的控制台演练以反映新的控制台体验，改进了身份池的代码集成详细信息。	2023 年 5 月 16 日
对服务主页和用户池主页进行了新增和改进。	更新了 Amazon Cognito 和用户池的概述页面。	2023 年 5 月 16 日
对用户池令牌文档的总体改进。	更新了示例令牌，添加了有关验证令牌的新信息。	2023 年 2 月 16 日

现在，您可以在中记录 Amazon Cognito 身份池数据事件。 AWS CloudTrail	CloudTrail 支持在记录数据事件的跟踪中选择 Amazon Cognito 身份池的大容量 API 操作。	2023 年 2 月 15 日
更新了 Lambda 触发器示例和描述。	Lambda 触发器示例已更新至 JavaScript 版本 3。您现在可以直接将 Lambda 触发器与 API 操作相关。	2023 年 1 月 31 日
Amazon Cognito 身份池将 AWS 托管策略应用于未经身份验证的会话。	使用增强型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 托管策略。	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。	2023 年 1 月 23 日
添加了有关 API 模型和使用 Amazon Cognito 用户池进行身份验证的信息。	Amazon Cognito 用户池具有多种用于请求授权的 API 接口和格式。	2022 年 12 月 15 日
亚马逊 Cognito 现已在欧洲（米兰）上市。 AWS 区域	您现在可以在欧洲地区（米兰）区域创建 Amazon Cognito 用户池。	2022 年 12 月 6 日
添加了有关用户池删除保护的信息。	当你使用创建新的用户池时 AWS Management Console，默认情况下会保护该用户池不被删除。	2022 年 10 月 20 日
为托管界面添加了用户指南，并在托管用户界面中添加了有关 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，您可以预览默认的托管用户界面。	2022 年 9 月 8 日

添加了有关 AWS WAF 和 Amazon Cognito 的信息。	现在，您可以将 AWS WAF 网页 ACL 与 Amazon Cognito 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。	2022 年 6 月 9 日
更新了联盟文档。新的 IP 地址传播功能。	更新了设置用户池社交 IdPs的演练。添加了有关联合用户配置文件和属性映射的信息。添加了有关设备指纹的新信息，以提高安全性。	2022 年 5 月 31 日
无需与托管用户界面交互即可登录联合用户	添加了有关如何为应用程序添加书签的新页面，以便 Amazon Cognito 以静默方式引导用户进行联合登录。	2022年5月29日
适用于 Amazon Cognito 用户池的区域内短信和电子邮件消息	现在，您可以将亚马逊简单通知服务用于发送短信，将亚马逊简单电子邮件服务用于发送电子邮件，这与您的用户池 AWS 区域 相同。	2022 年 3 月 14 日
配额页面的更新	添加并阐明了资源和请求速率配额。	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	您可以使用该 RevokeToken 操作来撤消用户的刷新令牌。	2021 年 6 月 10 日
多租户最佳实践	为多租户应用程序添加了最佳实践。	2021 年 3 月 4 日
访问控制属性	Amazon Cognito 身份池提供访问控制 (AFAC) 属性，作为客户向用户授予资源访问权限的一种方式。 AWS 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
自定义短信发送器 Lambda 触发器和自定义电子邮件发送器 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 30 日
亚马逊 Cognito 代币更新	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
亚马逊 Cognito 服务配额	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、请求增加配额以及创建 CloudWatch 警报以监控您的配额使用情况。作为此次变更的一部分，Amazon Cognito 用户池的可用 CloudWatch指标部分已更新，以反映新信息。新的栏目名称为：跟踪和 Service Quotas 中的配额 CloudWatch 和使用情况	2020 年 10 月 29 日

亚马逊 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
美国 GovCloud 支持亚马逊 Cognito AWS	AWS GovCloud（美国）地区现已支持 Amazon Cognito。	2020 年 5 月 13 日
亚马逊 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
全新 Amazon Cognito 专门的安全章节	“安全” 章节可以帮助您的组织获得有关 AWS 服务的内置和可配置安全性的深入信息。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
亚马逊 Cognito 身份池现在支持使用苹果登录	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
全新 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的建议。	2020 年 2 月 11 日
有关的新信息 AWS Amplify	添加了有关使用 AWS Amplify 软件开发工具包和库将 Amazon Cognito 与您的网络或移动应用程序集成的信息。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	现在，Amazon Cognito 在事件信息中包含一个`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
更新了限制	ListUsers API 操作的限制已更新。	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。	2019 年 6 月 17 日
亚马逊 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。	2019 年 4 月 8 日
标签支持	添加了有关标记 Amazon Cognito 资源的信息。	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。	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。	2018 年 12 月 14 日
更新了限制	更新了用户池的软限制。	2018 年 12 月 11 日

更新了用于验证电子邮件地址和电话号码的文档	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。	2018 年 11 月 20 日
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。	2018 年 11 月 13 日
亚马逊 Cognito 高级安全	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，确保用户账户拒绝已泄露的凭证，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
亚马逊 Cognito 托管用户界面的自定义域名	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
亚马逊 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4月8日
亚马逊 Cognito 开发者指南更新	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

亚马逊 Cognito 高级安全测试版	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
亚马逊 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。	2017 年 9 月 26 日
Amazon Cognito 用户池的联合和内置应用程序用户界面功能	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
与 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。	2016 年 12 月 15 日
文档更新	更新了显示如何在用户池中使用 AWS Lambda 触发器的示例。	2016年11月27日
文档更新	更新了 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。	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。	2016 年 10 月 6 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。	2016 年 9 月 1 日
Cognito 用户池正式上市	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。	2016 年 7 月 28 日
SAML 支持	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。	2016 年 6 月 23 日
CloudTrail 整合	增加了与的集成 AWS CloudTrail。	2016 年 2 月 18 日
将事件与 Lambda 集成	使您能够在 Amazon Cognito 中执行 AWS Lambda 函数以响应 Amazon Cognito 中的重要事件。	2015 年 4 月 9 日
数据流到 Amazon Kinesis	提供了对数据流的控制和了解。	2015 年 3 月 4 日
OpenID Connect 支持	启用了对 OpenID Connect 提供商的支持。	2014年11月23日
推送同步	启用了对无提示推送同步的支持	2014 年 11 月 6 日

添加了开发者身份验证支持	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。	2014 年 9 月 29 日
亚马逊 Cognito 正式上市		2014 年 7 月 10 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。