
Amazon Identity and Access Management

用户指南

亚马逊云科技

The Amazon logo, a curved orange arrow pointing from the letter 'A' to the letter 'Z', is positioned below the Chinese text.

Amazon Identity and Access Management: 用户指南

Copyright © 2022 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Amazon Web Services 文档中描述的 Amazon Web Services 服务或功能可能因区域而异。要查看适用于中国区域的差异，请参阅[中国的 Amazon Web Services 服务入门](#)。

Table of Contents

什么是 IAM ?	1
IAM 视频介绍	1
IAM 功能	1
访问 IAM	2
IAM 的工作原理	3
条款	4
主体	5
请求	5
身份验证	5
授权	5
操作	6
资源	6
Amazon 中的用户	6
仅限首次访问：您的根用户凭证	7
IAM 用户	7
联合现有用户	9
IAM 中的权限和策略	10
策略和账户	10
策略和用户	10
策略和组	10
联合身份用户和角色	11
基于身份和基于资源的策略	11
什么是 ABAC ?	12
ABAC 与传统 RBAC 模型的对比	12
IAM 外部的安全功能	13
常见任务的快速链接	13
使用 Amazon SDK	15
开始设置	16
使用 IAM 为用户授予对您的 Amazon 资源的访问权限	16
我是否需要注册 IAM ?	17
其他资源	17
开始使用	18
创建 IAM 管理员用户和用户组	19
创建管理员 IAM 用户和用户组（控制台）	19
创建 IAM 用户和用户组 (Amazon CLI)	20
相关资源	22
创建委派用户	22
创建委派的 IAM 用户和用户组（控制台）	19
减少用户组权限	23
IAM 用户如何登录	24
控制台活动所需的权限	25
在 CloudTrail 中记录登录详细信息	25
IAM 控制台搜索	26
使用 IAM 控制台搜索功能	26
IAM 控制台搜索结果中的图标	26
搜索短语示例	27
教程	28
委托对账单控制台的访问权限	28
先决条件	28
步骤 1：激活对您的 Amazon 测试账户的账单数据的访问权限	29
步骤 2：创建授予账单数据权限的 IAM policy	29
步骤 3：将账单策略附加到用户组	30
步骤 4：测试对账单控制台的访问权限	30
相关资源	31

摘要	31
使用角色委托跨 Amazon 账户的访问权限	31
先决条件	33
步骤 1：在生产账户中创建角色	33
步骤 2：对角色授予访问权限	35
步骤 3：通过切换角色测试访问权限	36
相关资源	40
摘要	40
创建客户托管策略	40
先决条件	40
步骤 1：创建策略	40
步骤 2：附加策略	41
步骤 3：测试用户访问权限	41
相关资源	42
摘要	42
使用基于属性的访问控制 (ABAC)	42
教程概述	43
先决条件	43
步骤 1：创建测试用户	44
步骤 2：创建 ABAC 策略	45
步骤 3：创建角色	47
步骤 4：测试创建密钥	48
步骤 5：测试查看密钥	49
步骤 6：测试可扩展性	51
步骤 7：测试更新和删除密钥	52
摘要	53
相关资源	53
将 SAML 会话标签用于 ABAC	53
使用户能够管理其凭证和 MFA 设置	56
先决条件	56
步骤 1：创建策略以强制 MFA 登录	57
步骤 2：将策略附加到您的测试用户组	57
步骤 3：测试您的用户的访问权限	58
相关资源	59
登录到 Amazon	60
以根用户身份登录	60
作为 IAM 用户登录	61
Amazon 账户 ID 及其别名	63
查找您的 Amazon 账户 ID	63
关于账户别名	64
创建、删除和列出 Amazon 账户别名	64
Amazon 登录问题	65
我的凭证不起作用	65
我需要我的 Amazon 账户 ID 或 Amazon 账户别名	66
我忘记了我的 IAM 用户名或密码	66
我忘记了我的 Amazon 账户的根用户密码	67
我无权访问我的 Amazon 账户的电子邮件	67
我需要变更我的 Amazon 账户的信用卡	67
我需要举报 Amazon 账户欺诈活动	67
我需要关闭我的 Amazon 账户	67
身份	68
Amazon 账户根用户	68
IAM 用户	68
IAM 用户组	68
IAM 角色	69
IAM 中的临时凭证	69
何时创建 IAM 用户（而不是角色）	69

何时创建 IAM 角色 (而不是用户)	69
用户	70
Amazon 如何标识 IAM 用户	70
用户和凭证	70
用户和权限	71
用户和账户	71
作为账户服务的用户	72
添加用户	72
控制用户访问控制台	76
IAM 用户如何登录 Amazon	77
管理用户	79
更改用户的权限	82
管理密码	87
访问密钥	96
检索丢失的密码或访问密钥	103
多重验证 (MFA)	104
查找未使用的凭证	138
获取凭证报告	141
将 IAM 与 CodeCommit 结合使用	144
将 IAM 与 Amazon Keyspaces 结合使用	146
管理服务器证书	147
用户组	151
创建用户组	153
管理用户组	153
角色	158
术语和概念	158
常见场景	161
身份提供程序和联合身份验证	170
服务相关角色	208
创建角色	215
使用角色	236
管理角色	260
角色与基于资源的策略	274
标记 IAM 资源	276
选择 Amazon 标签命名约定	276
在 IAM 和 Amazon STS 中进行标记的规则	277
标记 IAM 用户	278
标记 IAM 角色	280
标记客户托管策略	283
标记 IAM 身份提供程序	284
标记实例配置文件	288
标记服务器证书	290
标记虚拟 MFA 设备	291
会话标签	293
临时安全凭证	300
Amazon STS 和 Amazon 区域	301
临时凭证的常见情形	301
请求临时安全凭证	302
将临时凭证用于 Amazon 资源	311
控制临时安全凭证的权限	313
在 Amazon 区域中管理 Amazon STS	331
使用 Amazon STS 接口 VPC 终端节点	334
使用持有者令牌	336
使用临时凭证的示例应用程序	336
有关临时凭证的其他资源	337
Amazon 账户根用户	337
创建或删除 Amazon 账户	338

在 Amazon 账户根用户上启用 MFA	338
创建根用户的访问密钥	338
删除根用户的访问密钥	339
更改根用户的密码	340
保护根用户的凭证	340
转移根用户拥有者	340
使用 CloudTrail 录入事件	340
CloudTrail 中的 IAM 和 Amazon STS 信息	340
记录 IAM 和 Amazon STS API 请求	341
将 API 请求记录到其他 Amazon 服务	341
记录区域登录事件	341
记录用户登录事件	343
记录临时凭证的登录事件	343
CloudTrail 日志中的 IAM API 事件示例	344
CloudTrail 日志中的 Amazon STS API 事件示例	345
CloudTrail 日志中的登录事件示例	351
访问管理	353
访问管理资源	355
策略和权限	355
策略类型	355
策略和根用户	360
JSON 策略概述	360
授予最低权限	363
托管策略与内联策略	364
权限边界	370
身份与资源	379
使用策略控制访问	381
使用标签控制对 IAM 用户和角色的访问	388
使用标签控制对 Amazon 资源的访问	389
示例策略	391
管理 IAM 策略	437
创建 IAM policy	437
验证策略	443
生成策略	443
测试 IAM policy	443
添加或删除身份权限	451
IAM 策略版本控制	458
编辑 IAM 策略	461
删除 IAM 策略	465
使用访问信息优化权限	467
了解策略	483
策略摘要（服务列表）	484
服务摘要（操作列表）	493
操作摘要（资源列表）	496
策略摘要示例	499
所需权限	508
用于管理 IAM 身份的权限	508
在 Amazon Web Services Management Console 中工作的权限	509
授予跨 Amazon 账户权限	509
某种服务访问其他服务的权限	510
所需的操作	510
IAM 的示例策略	510
代码示例	514
IAM 示例	515
操作	517
场景	660
跨服务示例	730

Amazon STS 示例	732
操作	732
场景	737
安全性	747
数据保护	747
IAM 和 Amazon STS 中的数据加密	748
IAM 和 Amazon STS 中的密钥管理	748
IAM 和 Amazon STS 中的互联网络流量保密性	748
日志记录和监控	748
合规性验证	749
故障恢复能力	749
IAM 恢复能力的最佳实践	750
基础设施安全性	751
配置和漏洞分析	751
安全最佳实践和使用案例	751
安全最佳实践	752
商用案例	755
Amazon 托管策略	757
IAMReadOnlyAccess	758
IAMUserChangePassword	758
IAMAccessAnalyzerFullAccess	758
IAMAccessAnalyzerReadOnlyAccess	759
AccessAnalyzerServiceRolePolicy	760
策略更新	761
IAM Access Analyzer	762
识别与外部实体共享的资源	762
验证策略	763
生成策略	763
有关公共和跨账户存取的结果	763
IAM Access Analyzer 结果的工作原理	764
开始使用 IAM Access Analyzer 结果	764
处理结果	766
查看结果	766
筛选结果	768
对结果进行存档	770
解决结果	770
支持的资源类型	770
设置	773
存档规则	773
使用 EventBridge 监控	775
Security Hub 集成	779
使用 Cloudtrail 进行日志记录	782
IAM Access Analyzer 筛选条件键	784
使用服务相关角色	786
预览访问	788
在 Amazon S3 控制台中预览访问权限	788
使用 IAM Access Analyzer API 预览访问	789
IAM Access Analyzer 策略验证	791
在 IAM (控制台) 中验证策略	791
使用 Access Analyzer 验证策略 (Amazon CLI 或者 Amazon API)	792
策略检查参考	792
IAM Access Analyzer 策略生成	852
策略生成的工作原理	852
服务和操作级别的信息	853
需知信息	854
所需权限	855

基于 CloudTrail 活动生成策略 (控制台)	857
使用其他账户中的 Amazon CloudTrail 数据生成策略	859
基于 CloudTrail 活动生成策略 (Amazon CLI)	861
基于 CloudTrail 活动生成策略 (Amazon API)	861
IAM Access Analyzer 配额	862
IAM 故障排除	863
一般性问题	863
我无法登录我的 Amazon 账户	863
我丢失了访问密钥	863
当我向某个 Amazon 服务发送请求时, 收到了“访问被拒绝”	864
当我使用临时安全凭证发送请求时, 收到了“访问被拒绝”	864
策略变量不起作用	865
我所做的更改可能不会立即可见	865
我没有权限执行: iam:DeleteVirtualMFADevice	866
如何安全地创建 IAM 用户?	866
其他资源	867
拒绝访问错误消息	867
拒绝访问示例	867
IAM policy	870
使用可视化编辑器排除故障	870
使用策略摘要排除故障	873
策略管理故障排查	878
JSON 策略文档故障排查	878
FIDO 安全密钥	882
我无法启用我的 FIDO 安全密钥	882
我无法使用 FIDO 安全密钥登录	883
我的 FIDO 安全密钥已丢失或损坏	883
其它问题	883
IAM 角色	883
我无法代入角色	884
我的 Amazon 账户中出现新角色	885
我无法编辑或删除我的 Amazon 账户中的角色	885
我无权执行: iam:PassRole	885
为什么我无法在 12 小时会话中担任角色? (Amazon CLI、Amazon API)	886
当我尝试在 IAM 控制台中切换角色时收到错误	886
我的角色具有一个允许我执行操作的策略, 但出现“访问被拒绝”错误	886
服务未创建角色的默认策略版本	886
控制台中没有服务角色的使用案例	887
IAM 和 Amazon EC2	888
当尝试启动实例时, 我没有在 Amazon EC2 控制台的 IAM 角色列表中看到本应看到的角色	888
我的实例的凭证适用于错误的角色	889
当时尝试调用 AddRoleToInstanceProfile 时, 发生 AccessDenied 错误	889
Amazon EC2: 在我尝试使用角色启动实例时, 出现 AccessDenied 错误	889
我无法访问我的 EC2 实例的临时安全凭证	889
IAM 树形子目录中 info 文档的错误意味着什么?	890
IAM 和 Amazon S3	891
如何授权匿名访问 Amazon S3 存储桶?	891
我以 Amazon 账户根用户的身份登录后, 为什么无法在自己的账户中访问 Amazon S3 存储桶?	891
SAML 2.0 联合身份验证	891
无效的 SAML 响应	892
需要 RoleSessionName	892
没有 AssumeRoleWithSAML 授权	892
无效的 RoleSessionName 字符	893
无效的源身份字符	893
无效的响应签名	893
无法担任角色	893
无法解析元数据	893

无法解析元数据	894
DurationSeconds 超过 MaxSessionDuration	894
在浏览器中查看 SAML 响应	894
参考	896
IAM 标识符	896
易记名称和路径	896
IAM ARN	896
唯一标识符	900
配额、名称要求和字符限制	902
IAM 名称要求	903
IAM 对象配额	903
IAM Access Analyzer 配额	905
IAM 和 STS 字符限制	905
使用 IAM 的服务	908
计算	909
容器	910
存储	910
数据库。	911
开发人员工具	913
安全性、身份与合规性	914
加密和 PKI	916
机器学习	916
管理和治理	918
迁移与传输	921
移动	922
网络和内容分发	922
媒体	924
分析	925
应用程序集成	926
业务应用程序	927
卫星	927
物联网	927
机器人	929
Quantum 计算	929
区块链	929
游戏开发	929
AR 和 VR	930
客户支持	930
客户互动	930
最终用户计算	931
账单和成本管理	932
其他资源	932
策略参考	933
JSON 元素参考	934
策略评估逻辑	972
策略语法	985
工作职能的 Amazon 托管策略	990
全局条件键	998
IAM 条件键	1014
操作、资源和条件键	1025
资源	1026
用户和组	1026
证书 (密码、访问密钥和 MFA 设备)	1026
权限与策略	1026
联合和委托	1027
IAM 和其他 Amazon 产品	1027
将 IAM 与 Amazon EC2 结合使用	1027

将 IAM 与 Amazon S3 结合使用	1027
将 IAM 与 Amazon RDS 结合使用	1027
将 IAM 与 Amazon DynamoDB 结合使用	1027
一般安全实践	1027
一般资源	1028
发出 HTTP 查询请求	1029
终端节点	1029
必须使用 HTTPS	1029
签署 IAM API 请求	1029
文档历史记录	1031

什么是 IAM ?

 [Follow us on Twitter](#)

Amazon Identity and Access Management (IAM) 是一种 Web 服务，可以帮助您安全地控制对 Amazon 资源的访问。可以使用 IAM 来控制谁通过了身份验证（准许登录）并获得授权（拥有权限）来使用资源。

当您创建 Amazon Web Services 账户时，最初使用的是一个对账户中所有 Amazon Web Services 和资源拥有完全访问权限的登录身份。此身份称为 Amazon Web Services 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《Amazon 一般参考》中的[需要根用户凭证的任务](#)。

目录

- [IAM 视频介绍 \(p. 1\)](#)
- [IAM 功能 \(p. 1\)](#)
- [访问 IAM \(p. 2\)](#)
- [了解 IAM 的工作方式 \(p. 3\)](#)
- [Amazon 身份管理概述：用户 \(p. 6\)](#)
- [访问管理概述：权限和策略 \(p. 10\)](#)
- [什么是适用于 Amazon 的 ABAC ? \(p. 12\)](#)
- [IAM 外部的安全功能 \(p. 13\)](#)
- [常见任务的快速链接 \(p. 13\)](#)
- [将 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)

IAM 视频介绍

Amazon 培训和认证提供了介绍 IAM 的 10 分钟视频：

[Amazon Identity and Access Management 简介](#)

IAM 功能

IAM 为您提供以下功能：

对您 Amazon 账户的共享访问权限

您可以向其他人员授予管理和使用您 Amazon 账户中的资源的权限，而不必共享您的密码或访问密钥。

精细权限

您可以针对不同资源向不同人员授予不同权限。例如，您可以允许某些用户完全访问 Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Simple Storage Service (Amazon S3)、Amazon DynamoDB、Amazon Redshift 以及其他 Amazon 服务。对于另一些用户，您可以允许仅针对某些 S3 存储桶的只读访问权限，或是仅管理某些 EC2 实例的权限，或是访问您的账单信息但无法访问任何其他内容的权限。

在 Amazon EC2 上运行的应用程序针对 Amazon 资源的安全访问权限

您可以使用 IAM 功能安全地为 EC2 实例上运行的应用程序提供凭证。这些凭证为您的应用程序提供权限以访问其他 Amazon 资源。示例包括 S3 存储桶和 DynamoDB 表。

多重验证 (MFA)

您可以向您的账户和各个用户添加双重身份验证以实现更高安全性。借助 MFA，您或您的用户不仅必须提供使用账户所需的密码或访问密钥，还必须提供来自经过特殊配置的设备的代码。如果您已将 FIDO 安全密钥与其他服务配合使用，则该密钥应该具有 Amazon 支持的配置。有关更多信息，请参阅[使用 FIDO 安全密钥的受支持配置 \(p. 113\)](#)。

联合身份

您可以允许已在其他位置（例如，在您的企业网络中或通过 Internet 身份提供商）获得密码的用户获取对您 Amazon 账户的临时访问权限。

实现保证的身份信息

如果您使用 [Amazon CloudTrail](#)，则会收到日志记录，其中包括有关对您账户中的资源进行请求的人员的信息。这些信息基于 IAM 身份。

PCI DSS 合规性

IAM 支持由商家或服务提供商处理、存储和传输信用卡数据，而且已经验证符合支付卡行业 (PCI) 数据安全标准 (DSS)。有关 PCI DSS 的更多信息，包括如何请求 Amazon PCI Compliance Package 的副本，请参阅[PCI DSS 第 1 级](#)。

已与很多 Amazon 服务集成

有关使用 IAM 的 Amazon 服务的列表，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)。

最终一致性

IAM 和很多其他 Amazon 服务一样，具备[最终一致性](#)。IAM 通过复制 Amazon 在全球的数据中心内多个服务器上的数据实现高可用性。如果成功请求更改某些数据，则更改会提交并安全存储。不过，更改必须跨 IAM 复制，这需要时间。此类更改包括创建或更新用户、组、角色或策略。在应用程序的关键、高可用性代码路径中，我们不建议进行此类 IAM 更改。而应在不常运行的、单独的初始化或设置例程中进行 IAM 更改。另外，在生产工作流程依赖这些更改之前，请务必验证更改已传播。有关更多信息，请参阅[我所做的更改可能不会立即可见 \(p. 865\)](#)。

免费使用

Amazon Identity and Access Management (IAM) 和 Amazon Security Token Service (Amazon STS) 是为您的 Amazon 账户提供的一项功能，不会另外收费。只有在您使用 IAM 用户或 Amazon STS 临时安全证书访问其他 Amazon 服务时才会向您收取费用。有关其他 Amazon 产品的定价信息，请参阅[Amazon Web Services 定价页面](#)。

访问 IAM

您可以通过以下任何方式使用 Amazon Identity and Access Management。

Amazon Web Services Management Console

控制台是用于管理 IAM 和 Amazon 资源的基于浏览器的界面。有关通过控制台访问 IAM 的更多信息，请参阅[以 IAM 用户或根用户的身份登录到 Amazon Web Services Management Console \(p. 60\)](#)。有关指导您使用控制台的教程，请参阅[创建您的第一个 IAM 管理员用户和用户组 \(p. 19\)](#)。

Amazon 命令行工具

您可以使用 Amazon 命令行工具，在系统的命令行中发出命令以执行 IAM 和 Amazon 任务。与控制台相比，使用命令行更快、更方便。如果要构建执行 Amazon 任务的脚本，命令行工具也会十分有用。

Amazon 提供两组命令行工具：[Amazon Command Line Interface \(Amazon CLI\)](#) 和 [Amazon Tools for Windows PowerShell](#)。有关安装和使用 Amazon CLI 的更多信息，请参阅[Amazon Command Line Interface 用户指南](#)。有关安装和使用 Tools for Windows PowerShell 的信息，请参阅[Amazon Tools for Windows PowerShell 用户指南](#)。

Amazon 软件开发工具包

Amazon 提供的 SDK (开发工具包) 包含各种编程语言和平台 (Java、Python、Ruby、.NET、iOS、Android 等) 的库和示例代码。开发工具包提供了可通过编程方式访问 IAM 和 Amazon 的便捷方式。例如，开发工具包执行以下类似任务：加密签署请求、管理错误以及自动重试请求。有关 Amazon 开发工具包的信息（包括如何下载及安装），请参阅[适用于 Amazon Web Services 的工具](#)页面。

IAM HTTPS API

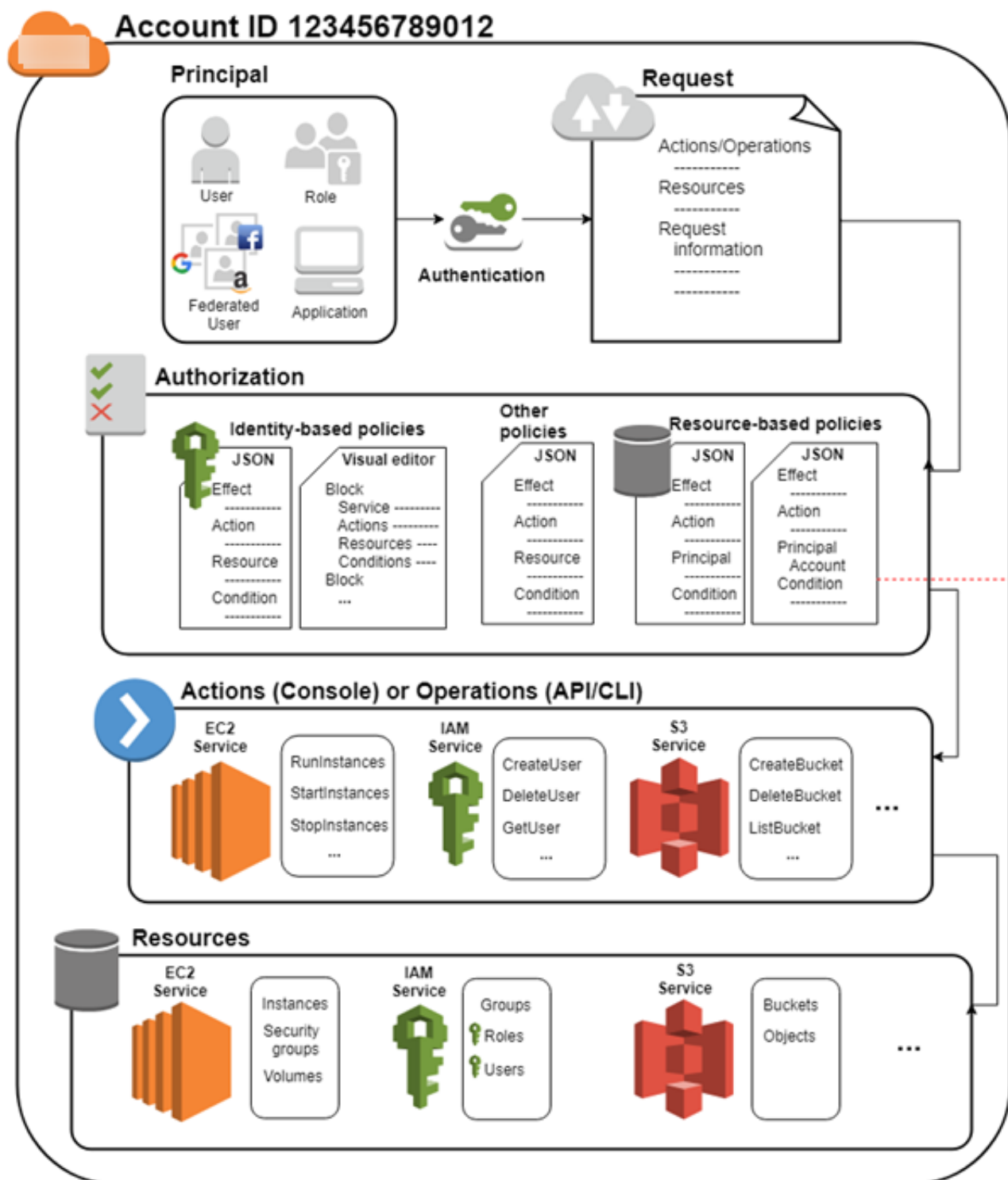
您可以使用 IAM HTTPS API（可让您直接向服务发布 HTTPS 请求）以编程方式访问 IAM 和 Amazon。使用 HTTPS API 时，必须添加代码，才能使用您的凭证对请求进行数字化签名。有关更多信息，请参阅[使用 HTTP 查询请求调用 IAM API \(p. 1029\)](#) 和 [IAM API 参考](#)。

了解 IAM 的工作方式

在创建用户之前，您应该了解 IAM 的工作方式。IAM 提供了控制您的账户的身份验证和授权所需的基础设施。IAM 基础设施包含以下元素：

目录

- [条款 \(p. 4\)](#)
- [主体 \(p. 5\)](#)
- [请求 \(p. 5\)](#)
- [身份验证 \(p. 5\)](#)
- [授权 \(p. 5\)](#)
- [操作 \(p. 6\)](#)
- [资源 \(p. 6\)](#)



条款

了解有关 IAM 术语的更多信息。

IAM 资源

存储在 IAM 中的用户、组、角色、策略和身份提供商对象。与其他 Amazon 服务一样，您可以在 IAM 中添加、编辑和删除资源。

IAM 身份

用于标识和分组的 IAM 资源对象。您可以将策略附加到 IAM 身份。其中包括用户、组和角色。

IAM 实体

Amazon 用于进行身份验证的 IAM 资源对象。其中包括 IAM 用户和角色。

主体

使用 Amazon Web Services 账户、IAM 用户或 IAM 角色登录并向 Amazon 发出请求的人员或应用程序。主体包括联合身份用户和代入的角色。

主体

主体是可请求对 Amazon 资源执行操作的人员或应用程序。主体将作为 Amazon Web Services 账户或 IAM 实体进行身份验证以向 Amazon 发出请求。作为最佳实践，请勿使用您的根用户凭证完成日常工作。而是创建 IAM 实体（用户和角色）。您还可以支持联合身份用户或编程访问以允许应用程序访问您的 Amazon 账户。

请求

在主体尝试使用 Amazon Web Services Management Console、Amazon API 或 Amazon CLI 时，该主体将向 Amazon 发送请求。请求包含以下信息：

- 操作 – 主体希望执行的操作。这可以是 Amazon Web Services Management Console 中的操作或者 Amazon CLI 或 Amazon API 中的操作。
- 资源 – 对其执行操作的 Amazon 资源对象。
- 主体 – 已使用实体（用户或角色）发送请求的人员或应用程序。有关主体的信息包括与主体用于登录的实体关联的策略。
- 环境数据 – 有关 IP 地址、用户代理、SSL 启用状态或当天时间的信息。
- 资源数据 – 与请求的资源相关的数据。这可能包括 DynamoDB 表名称或 Amazon EC2 实例上的标签等信息。

Amazon 将请求信息收集到请求上下文中，后者用于评估和授权请求。

身份验证

主体必须使用其凭证进行身份验证（登录到 Amazon）以向 Amazon 发送请求。某些服务（如 Amazon S3 和 Amazon STS）允许一些来自匿名用户的请求。不过，它们是该规则的例外情况。

要以根用户身份从控制台中进行身份验证，必须使用您的电子邮件地址和密码登录。作为 IAM 用户，请提供您的账户 ID 或别名，然后提供您的用户名和密码。要从 API 或 Amazon CLI 中进行身份验证，您必须提供访问密钥和私有密钥。您可能需要提供额外的安全信息。例如，Amazon 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解有关 Amazon 可验证的 IAM 实体的更多信息，请参阅 [IAM 用户 \(p. 70\)](#) 和 [IAM 角色 \(p. 158\)](#)。

授权

您还必须获得授权（允许）才能完成您的请求。在授权期间，Amazon 使用请求上下文中的值来检查应用于请求的策略。然后，它使用策略来确定是允许还是拒绝请求。大多数策略作为 [JSON 文档 \(p. 360\)](#) 存储在

Amazon 中，并指定主体实体的权限。有[多种类型的策略 \(p. 355\)](#)可影响是否对请求进行授权。要向用户提供访问他们自己账户中的 Amazon 资源的权限，只需基于身份的策略。基于资源的策略常用于授予[跨账户访问 \(p. 509\)](#)。其他策略类型是高级功能，应谨慎使用。

Amazon 检查应用于请求上下文的每个策略。如果一个权限策略包含拒绝的操作，Amazon 将拒绝整个请求并停止评估。这称为显式拒绝。由于请求是默认被拒绝的，因此，只有在适用的权限策略允许请求的每个部分时，Amazon 才会授权请求。单个账户中对于请求的评估逻辑遵循以下一般规则：

- 默认情况下，所有请求都将被拒绝。（通常，始终允许使用 Amazon Web Services 账户根用户凭证创建的访问该账户资源的请求。）
- 任何权限策略（基于身份或基于资源）中的显式允许将覆盖此默认值。
- 组织 SCP、IAM 权限边界或会话策略的存在将覆盖允许。如果存在其中一个或多个策略类型，它们必须都允许请求。否则，将隐式拒绝它。
- 任何策略中的显式拒绝将覆盖任何允许。

要了解有关如何评估所有类型的策略的更多信息，请参阅[策略评估逻辑 \(p. 972\)](#)。如果您需要在另一个账户中发出请求，此其他账户中的策略必须允许访问资源，并且您用于发出请求的 IAM 实体必须具有允许该请求的基于身份的策略。

操作

在对您的请求进行身份验证和授权后，Amazon 将批准请求中的操作。操作是由服务定义的，包括可以对资源执行的操作，例如，查看、创建、编辑和删除该资源。例如，IAM 为用户资源支持大约 40 个操作，包括以下操作：

- CreateUser
- DeleteUser
- GetUser
- UpdateUser

要允许主体执行操作，您必须在应用于主体或受影响的资源的策略中包含所需的操作。要查看各个服务支持的操作、资源类型和条件键的列表，请参阅[Amazon 服务的操作、资源和条件键](#)。

资源

在 Amazon 批准请求中的操作后，可以对您的账户中的相关资源执行这些操作。资源是位于服务中的对象。示例包括 Amazon EC2 实例、IAM 用户和 Amazon S3 存储桶。服务定义了一组可对每个资源执行的操作。如果创建一个请求以对资源执行不相关的操作，则会拒绝该请求。例如，如果您请求删除一个 IAM 角色，但提供一个 IAM 组资源，请求将失败。要查看确定操作影响哪些资源的 Amazon 服务表，请参阅[Amazon 服务的操作、资源和条件键](#)。

Amazon 身份管理概述：用户

为实现更好的安全性和企业，您可以向特定用户（使用自定义权限创建的身份）授予对您的 Amazon 账户的访问权限。通过将现有身份联合到 Amazon 中，可以进一步简化这些用户的访问。

主题

- [仅限首次访问：您的根用户凭证 \(p. 7\)](#)
- [IAM 用户 \(p. 7\)](#)
- [联合现有用户 \(p. 9\)](#)

仅限首次访问：您的根用户凭证

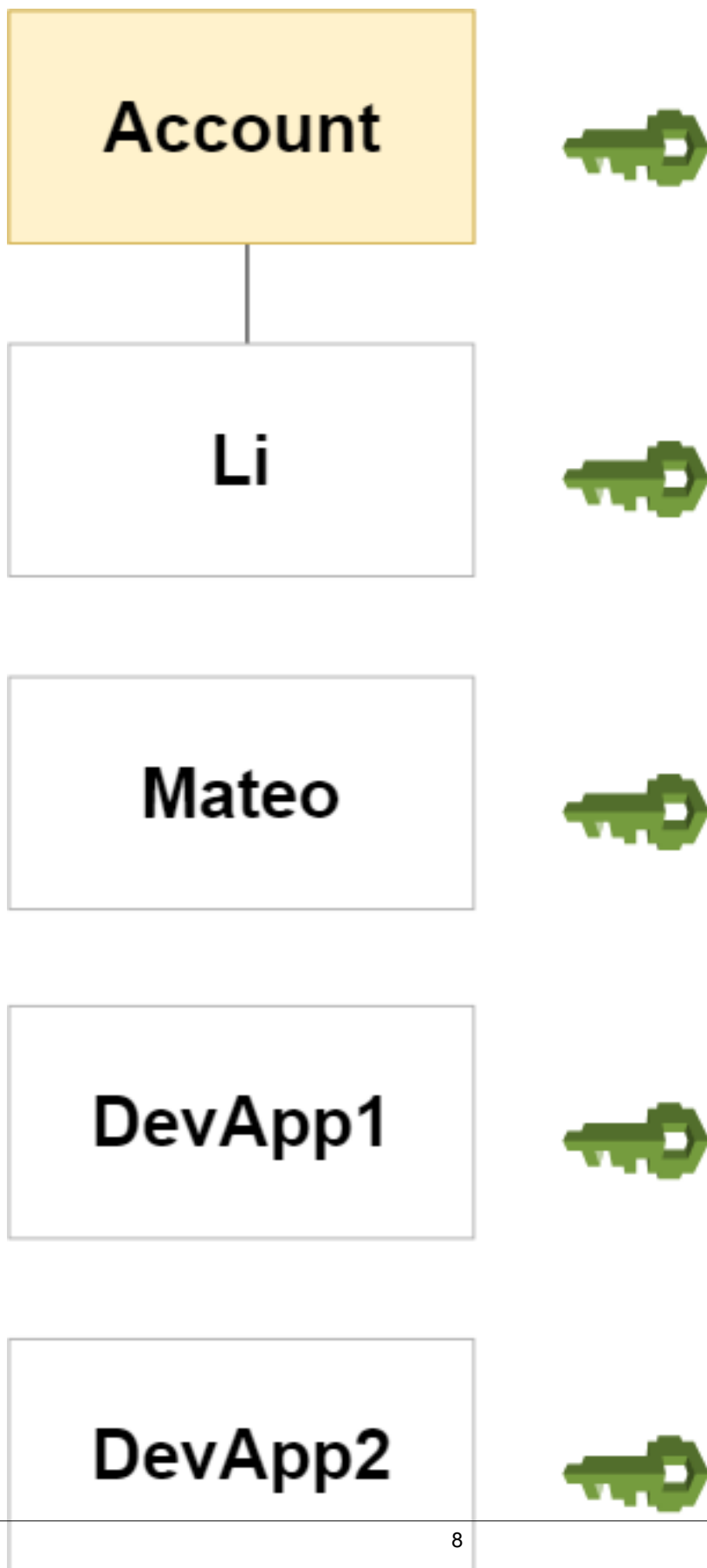
创建 Amazon 账户时，会创建一个用于登录 Amazon 的 Amazon Web Services 账户 根用户身份。您可以使用此身份 (即创建账户时提供的电子邮件地址和密码) 登录 Amazon Web Services Management Console。您的电子邮件地址和密码的组合也称为您的根用户凭证。

使用根用户凭证时，您可以对 Amazon 账户中的所有资源进行完全、无限制的访问，包括访问您的账单信息，您还能更改自己的密码。当您首次设置账户时，需要此访问级别。但是，我们不建议使用根用户凭证进行日常访问。我们特别建议您不要与任何人共享您的根用户凭证，因为如果这样做，他们可对您的账户进行无限制的访问。只有企业中的服务控制策略 (SCP) 才能限制授予根用户的权限。

以下几节说明如何使用 IAM 创建和管理用户身份和权限以提供对您 Amazon 资源的安全、有限访问，适用于您自己以及需要使用您 Amazon 资源的其他人员。

IAM 用户

Amazon Identity and Access Management (IAM) 的“身份”方面可帮助您解决问题“该用户是谁？”(通常称为身份验证)。您可以在账户中创建与企业中的用户对应的各 IAM 用户，不要与他人共享您的根用户凭证。IAM 用户不是单独的账户；它们是您的账户中的用户。每个用户都可以有自己的密码以用于访问 Amazon Web Services Management Console。您还可以为每个用户创建单独的访问密钥，以便用户可以发出编程请求以使用账户中的资源。在下图中，用户 Li、Mateo、DevApp1、DevApp2、TestApp1 和 TestApp2 已添加到单个 Amazon 账户。每个用户都有自己的凭证。



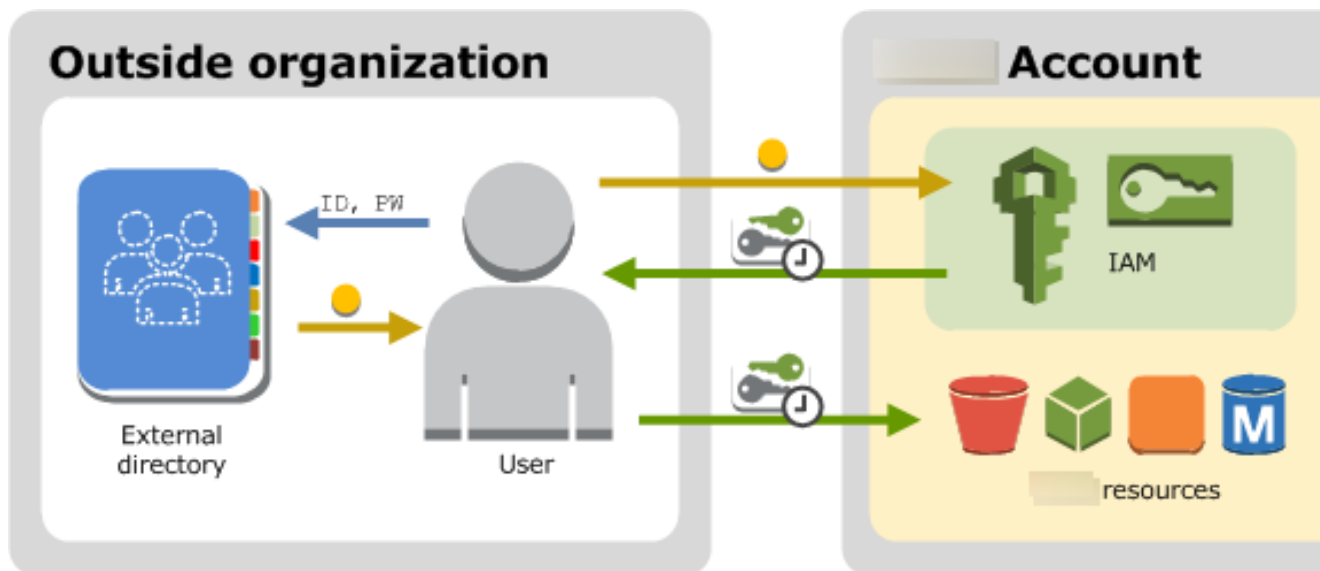
请注意，某些用户实际上是应用程序（例如 DevApp1）。IAM 用户不必表示实际人员；您可以创建 IAM 用户以便为在公司网络中运行并需要 Amazon 访问权限的应用程序生成访问密钥。

我们建议您为自己创建 IAM 用户，然后向自己分配账户的管理权限。您随后可以作为该用户登录以根据需要添加更多用户。

联合现有用户

如果您的企业中的用户已通过某种方法进行身份验证（例如，通过登录到您的公司网络），则不必为他们创建单独的 IAM 用户。相反，您可以在 Amazon 中对这些用户身份进行联合身份验证。

下图介绍用户如何使用 IAM 获取临时 Amazon 安全凭证以访问您 Amazon 账户中的资源。



联合在这些情况下尤其有用：

- 您的用户已在公司目录中拥有身份。

如果您的公司目录与安全断言标记语言 2.0 (SAML 2.0) 兼容，则可以配置公司目录以便为用户提供对 Amazon Web Services Management Console 的单一登录 (SSO) 访问。有关更多信息，请参阅[临时凭证的常见情形 \(p. 301\)](#)。

如果您的公司目录不与 SAML 2.0 兼容，则可以创建身份代理应用程序以便为用户提供对 Amazon Web Services Management Console 的单一登录 (SSO) 访问。有关更多信息，请参阅[使自定义身份代理能够访问 Amazon 控制台 \(p. 201\)](#)。

如果您的公司目录是 Microsoft Active Directory，则可以使用 [Amazon Directory Service](#) 在公司目录与您的 Amazon 账户之间建立信任。

- 您的用户已有 Internet 身份。

如果您创建的移动应用程序或基于 Web 的应用程序可以允许用户通过 Internet 身份提供商 (如 Login with Amazon、Facebook、Google 或任何与 OpenID Connect (OIDC) 兼容的身份提供商) 标识自己，则应用程序可以使用联合访问 Amazon。有关更多信息，请参阅[关于 Web 联合身份验证 \(p. 171\)](#)。

提示

要使用与 Internet 身份提供商的联合身份，我们建议使用 [Amazon Cognito](#)。

访问管理概述：权限和策略

Amazon Identity and Access Management (IAM) 的访问管理部分帮助定义主体实体可在账户内执行的操作。主体实体是指使用 IAM 实体（用户或角色）进行身份验证的人员或应用程序。访问管理通常称为授权。您在 Amazon 中通过创建策略并将其附加到 IAM 身份（用户、用户组或角色）或 Amazon 资源来管理访问权限。策略是 Amazon 中的对象；在与身份或资源相关联时，策略定义它们的权限。在主体使用 IAM 实体（如用户或角色）发出请求时，Amazon 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 Amazon 中存储为 JSON 文档。有关策略类型和用法的更多信息，请参阅[IAM 中的策略和权限](#)。（p. 355）。

策略和账户

如果您管理 Amazon 中的单个账户，则使用策略定义该账户中的权限。如果您管理跨多个账户的权限，则管理用户的权限会比较困难。您可以将 IAM 角色、基于资源的策略或访问控制列表 (ACL) 用于跨账户权限。但是，如果您拥有多个账户，那我们建议您改用该 Amazon Organizations 服务来帮助管理这些权限。有关更多信息，请参阅 Organizations 用户指南 中的[什么是 Amazon Organizations](#)。

策略和用户

IAM 用户是服务中的身份。当您创建 IAM 用户时，他们无法访问您账户中的任何内容，直到您向他们授予权限。向用户授予权限的方法是创建基于身份的策略，这是附加到用户或用户所属组的策略。下面的示例演示一个 JSON 策略，该策略允许用户对 us-east-2 区域内的 123456789012 账户中的 Books 表执行所有 Amazon DynamoDB 操作 (dynamodb:*)。

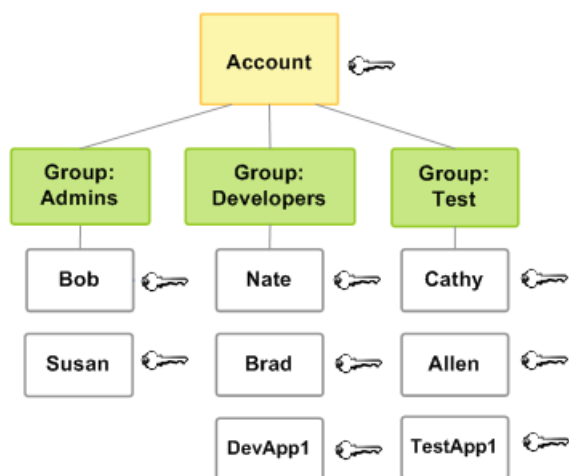
```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "dynamodb:*",
    "Resource": "arn:aws:dynamodb:us-east-2:123456789012:table/Books"
  }
}
```

在将此策略附加到您的 IAM 用户后，该用户将仅具有这些 DynamoDB 权限。大多数用户有多个策略共同代表该用户的权限。

默认情况下会拒绝未显式允许的操作或资源。例如，如果上述策略是附加到用户的唯一策略，则该用户只能对 Books 表执行 DynamoDB 操作。禁止对其他任何表执行操作。同样，不允许用户在 Amazon EC2、Amazon S3 或任何其他 Amazon 服务中执行任何操作。原因是策略中未包含使用这些服务的权限。

策略和组

可以将 IAM 用户组织为 IAM 组，然后将策略附加到组。这种情况下，各用户仍有自己的凭证，但是组中的所有用户都具有附加到组的权限。使用组可更轻松地管理权限，并遵循我们的[IAM 中的安全最佳实践](#)（p. 752）。



用户或组可以附加授予不同权限的多个策略。这种情况下，权限基于策略组合进行计算。不过基本原则仍然适用：如果未向用户授予针对操作和资源的显式权限，则用户没有这些权限。

联合身份用户和角色

联合身份用户无法通过与 IAM 用户相同的方式在您的 Amazon 账户中获得永久身份。要向联合身份用户分配权限，可以创建称为角色的实体，并为角色定义权限。当联合用户登录 Amazon 时，该用户会与角色关联，被授予角色中定义的权限。有关更多信息，请参阅[针对第三方身份提供程序创建角色（联合）](#)（p. 224）。

基于身份和基于资源的策略

基于身份的策略是附加到 IAM 身份（如 IAM 用户、组或角色）的权限策略。基于资源的策略是附加到资源（如 Amazon S3 存储桶或 IAM 角色信任策略）的权限策略。

基于身份的策略控制身份可以在哪些条件下对哪些资源执行哪些操作。基于身份的策略可以进一步分类：

- 托管策略——基于身份的独立策略，可附加到您的 Amazon 账户中的多个用户、组和角色。您可以使用两个类型的托管策略：
 - Amazon 托管策略——由 Amazon 创建和管理的托管策略。如果您刚开始使用策略，建议先使用 Amazon 托管策略。
 - 客户托管策略——您在 Amazon 账户中创建和管理的托管策略。与 Amazon 托管策略相比，客户托管策略可以更精确地控制策略。您可以在可视化编辑器中创建、编辑和验证，它是一项 IAM 策略，也可以直接创建 JSON 策略文档以创建和编辑该策略。有关更多信息，请参阅[创建 IAM policy](#)（p. 437）和[编辑 IAM 策略](#)（p. 461）。
- 内联策略——由您创建和管理的策略，直接嵌入在单个用户、组或角色中。大多数情况下，我们不建议使用内联策略。

基于资源的策略控制指定的主体可以在何种条件下对该资源执行哪些操作。基于资源的策略是内联策略，没有基于资源的托管策略。要启用跨账户访问，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。

IAM 服务仅支持一种类型的基于资源的策略（称为角色信任策略），这种策略附加到 IAM 角色。由于 IAM 角色同时是支持基于资源的策略的身份和资源，因此，您必须同时将信任策略和基于身份的策略附加到 IAM 角色。信任策略定义哪些主体实体（账户、用户、角色和联合身份用户）可以代入该角色。要了解 IAM 角色如何与其他基于资源的策略不同，请参阅[IAM 角色与基于资源的策略有何不同](#)（p. 274）。

要了解哪些服务支持基于资源的策略，请参阅[使用 IAM 的 Amazon 服务](#)（p. 908）。要了解基于资源的策略的更多信息，请参阅[基于身份的策略和基于资源的策略](#)（p. 379）。

什么是适用于 Amazon 的 ABAC？

基于属性的访问控制 (ABAC) 是一种授权策略，该策略基于属性来定义权限。在 Amazon 中，这些属性称为标签。您可以将标签附加到 IAM 资源（包括 IAM 实体（用户和角色））以及 Amazon 资源。您可以为 IAM 委托人创建单个 ABAC 策略或者一小组策略。这些 ABAC 策略可设计为在主体的标签与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。

例如，您可以创建具有 `access-project` 标签键的三个角色。将第一个角色的标签值设置为 `Heart`，第二个为 `Sun`，第三个为 `Lightning`。然后，您可以使用单个策略，在角色和资源标记了 `access-project` 的相同值时允许访问。有关演示如何在 Amazon 中使用 ABAC 的详细教程，请参阅[IAM 教程：根据标签定义访问 Amazon 资源的权限 \(p. 42\)](#)。

ABAC 与传统 RBAC 模型的对比

IAM 中使用的传统授权模型称为基于角色的访问控制 (RBAC)。RBAC 根据用户的工作职能定义权限，在 Amazon 之外称为角色。在 Amazon 中，角色通常是指 IAM 角色，它是您可以代入 IAM 中的身份。IAM 确实包含[任务函数的管式策略 \(p. 990\)](#)，能够将权限分配给 RBAC 模型中的任务函数。

在 IAM 中，您通过为不同工作职能创建不同策略来实施 RBAC。然后，您可将策略附加到身份（IAM 用户、用户组或 IAM 角色）。作为最佳实践，您向工作职能授予所需的最小权限。这称为[授予最小权限 \(p. 754\)](#)。通过列出工作职能可以访问的特定资源来完成此操作。使用传统 RBAC 模型的缺点在于，当员工添加新资源时，您必须更新策略以允许访问这些资源。

例如，假设您的员工在处理三个项目，名为 `Heart`、`Sun` 和 `Lightning`。您可以为每个项目创建一个 IAM 角色。然后，您将策略附加到各个 IAM 角色，定义允许代入该角色的任何用户可以访问的资源。如果员工更换了公司中的工作，您向其分配不同的 IAM 角色。用户或计划可以分配到多个角色。但是，`Sun` 项目可能需要额外的资源，例如新的 Amazon S3 存储桶。在这种情况下，您必须更新附加到 `Sun` 角色的策略来指定新存储桶资源。否则，`Sun` 项目成员不允许访问新的存储桶。

相比传统 RBAC 模型，ABAC 具备以下优势：

- ABAC 权限随着创新扩展。它不再需要管理员更新现有策略以允许对新资源的访问。例如，假设您使用 `access-project` 标签指定了 ABAC 策略。开发人员使用 `access-project = Heart` 标签的角色。当 `Heart` 项目中的员工需要额外的 Amazon EC2 资源时，开发人员可以使用 `access-project = Heart` 标签创建新 Amazon EC2 实例。这样，`Heart` 项目中的任何员工可以启动和停止这些实例，因为其标签值匹配。
- ABAC 需要较少的策略。由于您无需为不同工作职能创建不同策略，需要创建的策略数量减少。这些策略更易于管理。
- 使用 ABAC，团队可以进行更改和扩展。这是因为新资源的权限根据属性自动授予。例如，如果您的公司已经使用 ABAC 支持 `Heart` 和 `Sun` 项目，则可以轻松地添加新 `Lightning` 项目。IAM 管理员创建具有 `access-project = Lightning` 标签的新角色。无需更改策略以支持新项目。有权代入该角色的任何用户可以创建和查看使用 `access-project = Lightning` 标记的实例。此外，团队成员可以从 `Heart` 项目转向 `Lightning` 项目。IAM 管理员将用户分配到不同 IAM 角色。无需更改权限策略。
- 使用 ABAC 可以实现精细权限。在您创建策略时，最佳实践是[授予最小权限 \(p. 754\)](#)。使用传统 RBAC，您必须编写一个策略，仅允许访问特定资源。但是，如果使用 ABAC，您可以允许在所有资源上的操作，但仅在资源标签与主体标签匹配时。
- 通过 ABAC 使用来自您公司目录的员工属性。您可以对基于 SAML 的身份提供商或 Web 身份提供商进行配置，将会话标签传递给 Amazon。当您的员工希望联合身份到 Amazon 中时，其属性将应用到 Amazon 中所得到的主体。然后，您可以使用 ABAC 来允许或拒绝基于这些属性的权限。

有关演示如何在 Amazon 中使用 ABAC 的详细教程，请参阅[IAM 教程：根据标签定义访问 Amazon 资源的权限 \(p. 42\)](#)。

IAM 外部的安全功能

通过 IAM 可以控制对使用 Amazon Web Services Management Console、[Amazon 命令行工具](#)或服务 API 操作（通过使用 [Amazon 开发工具包](#)）执行的任务的访问。某些 Amazon 产品还有其它方法来保护其资源。以下列表提供了一些示例，不过并不详尽。

Amazon EC2

在 Amazon Elastic Compute Cloud 中，需要使用密钥对（对于 Linux 实例）或使用用户名称和密码（对于 Windows 实例）来登录实例。

有关更多信息，请参阅以下文档：

- 适用于 Linux 实例的 Amazon EC2 用户指南中的 [Amazon EC2 Linux 实例入门](#)
- 适用于 Windows 实例的 Amazon EC2 用户指南中的 [Amazon EC2 Windows 实例入门](#)

Amazon RDS

在 Amazon Relational Database Service 中，需要使用与数据库关联的用户名称和密码来登录数据库引擎。

有关更多信息，请参阅 Amazon RDS 用户指南 中的 [Amazon RDS 入门](#)。

Amazon EC2 和 Amazon RDS

在 Amazon EC2 和 Amazon RDS 中，需要使用安全组来控制发送到实例或数据库的流量。

有关更多信息，请参阅以下文档：

- 《适用于 Linux 实例的 Amazon EC2 用户指南》中的 [适用于 Linux 实例的 Amazon EC2 安全组](#)
- 请参阅 Amazon EC2 用户指南（适用于 Windows 实例）中的 [适用于 Windows 实例的 Amazon EC2 安全组](#)。
- Amazon RDS 用户指南中的 [Amazon RDS 安全组](#)

WorkSpaces

在 Amazon WorkSpaces 中，用户使用用户名称和密码登录桌面。

有关更多信息，请参阅 Amazon WorkSpaces 管理指南中的 [WorkSpaces 入门](#)。

Amazon WorkDocs

在 Amazon WorkDocs 中，用户通过使用用户名称和密码进行登录来访问共享文档。

有关更多信息，请参阅 [Amazon WorkDocs 管理指南](#) 中的 Amazon WorkDocs 入门。

这些访问控制方法不是 IAM 的一部分。IAM 允许您控制管理这些 Amazon 产品的方式，方法包括创建或终止 Amazon EC2 实例、设置新 WorkSpaces 桌面等。也就是说，IAM 可帮助您控制通过向 Amazon Web Services 进行请求来执行的任务，并且可帮助您控制对 Amazon Web Services Management Console 的访问。但是，IAM 不会帮助您管理诸如登录操作系统 (Amazon EC2)、数据库 (Amazon RDS)、桌面 (Amazon WorkSpaces) 或协作站点 (Amazon WorkDocs) 等任务的安全性。

当您使用特定 Amazon 产品时，请务必阅读相应文档，了解属于该产品的所有资源的安全选项。

常见任务的快速链接

使用以下链接可获得与 IAM 关联的常见任务的帮助。

作为 IAM 用户登录

请参阅[IAM 用户如何登录 Amazon](#) (p. 77)。

管理 IAM 用户的密码

您需要密码才能访问 Amazon Web Services Management Console (包括对账单信息的访问)。

对于您的 Amazon Web Services 账户 根用户，请参阅 [更改 Amazon 账户根用户密码](#) (p. 87)。

对于 IAM 用户，请参阅 [管理 IAM 用户的密码](#) (p. 90)。

IAM 用户的管理权限

您可以使用策略在您的 Amazon 账户中向 IAM 用户授予权限。IAM 用户在创建时没有任何权限，因此您必须添加权限才能让他们使用 Amazon 资源。

有关更多信息，请参阅[管理 IAM 策略](#) (p. 437)。

列出您 Amazon 账户中的用户并获取有关其凭证的信息

请参阅[获取您 Amazon 账户的凭证报告](#) (p. 141)。

添加 Multi-Factor Authentication (MFA)

要添加虚拟 MFA 设备，请参阅以下内容之一：

- [为您的 Amazon 账户根用户启用虚拟 MFA 设备 \(控制台 \)](#) (p. 108)
- [为 IAM 用户启用虚拟 MFA 设备 \(控制台 \)](#) (p. 107)

要添加 FIDO 安全密钥，请参阅以下内容之一：

- [为 Amazon 账户根用户启用 FIDO 安全密钥 \(控制台 \)](#) (p. 112)
- [为其他 IAM 用户启用 FIDO 安全密钥 \(控制台 \)](#) (p. 111)

要添加硬件 MFA 设备，请参阅以下内容之一：

- [为 Amazon 账户根用户启用硬件 MFA 设备 \(控制台 \)](#) (p. 117).
- [为其他 IAM 用户启用硬件 MFA 设备 \(控制台 \)](#) (p. 117)

获取访问密钥

如果要使用 [Amazon 开发工具包](#)、[Amazon 命令行工具](#)或 API 操作发出 Amazon 请求，则需要访问密钥。

Important

仅当创建访问密钥时，才能查看和下载秘密访问密钥。以后您将无法查看或找回秘密访问密钥。但是，如果您丢失了秘密访问密钥，可以创建新的访问密钥。

对于您的 Amazon 账户，请参阅[管理 Amazon 账户的访问密钥](#)。

对于 IAM 用户，请参阅 [管理 IAM 用户的访问密钥](#) (p. 96)。

为 IAM 资源添加标签

您可以标记以下 IAM 资源：

- IAM 用户
- IAM 角色
- 客户托管策略
- 身份提供商
- 服务器证书
- 虚拟 MFA 设备

要了解有关 IAM 中的标签的更多信息，请参阅 [标记 IAM 资源](#) (p. 276)。

要了解有关使用标签控制对 Amazon 资源的访问权限的信息，请参阅[使用标签控制对 Amazon 资源的访问 \(p. 389\)](#)。

查看所有服务的操作、资源和条件键

这组参考文档可帮助您编写详细的 IAM policy。每个 Amazon 服务都定义了您在 IAM policy 中使用的操作、资源和条件上下文键。要了解更多信息，请参阅[Amazon 服务的操作、资源和条件键](#)。

开始使用 Amazon 的各项功能

这套文档主要介绍 IAM 服务。要了解如何使用 Amazon 以及使用多种服务来解决构建和启动您的第一个项目等问题，请参阅[入门资源中心](#)。

将 IAM 与 Amazon 开发工具包配合使用

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

软件开发工具包文档
Amazon SDK for Java
Amazon SDK for JavaScript
Amazon SDK for Kotlin
Amazon SDK for .NET
Amazon SDK for PHP
Amazon SDK for Python (Boto3)
Amazon SDK for Ruby
Amazon SDK for Rust
Amazon SDK for Swift

有关特定于 IAM 的示例，请参阅[使用 Amazon 开发工具包的 IAM 代码示例 \(p. 514\)](#)。

开始设置 IAM

Amazon Identity and Access Management (IAM) 可以帮助您安全地控制对 Amazon Web Services (Amazon) 和您的账户资源的访问。IAM 也可以使您的账户证书保持私密。利用 IAM，您可以在您的 IAM 账户的伞形结构下创建多个 Amazon 用户，或通过与企业目录的联合身份实现临时访问。在某些情况下，您也可以实现跨 Amazon 账户访问资源。

但是，如果不使用 IAM，则只有两种方法：一种是您必须创建多个 Amazon 账户（即，每个账户都有自己针对 Amazon 产品的计费 and 订阅），另一种是您的员工必须共享一个 Amazon 账户的安全凭证。此外，如果不使用 IAM，您无法控制特定用户或系统可以完成的任务以及他们可能使用哪些 Amazon 资源。

本指南提供了关于 IAM 的概念综述、描述了商用案例并说明了 Amazon 权限和策略。

主题

- [使用 IAM 为用户授予对您的 Amazon 资源的访问权限 \(p. 16\)](#)
- [我是否需要注册 IAM？ \(p. 17\)](#)
- [其他资源 \(p. 17\)](#)

使用 IAM 为用户授予对您的 Amazon 资源的访问权限

以下是您可以使用 IAM 控制您的 Amazon 资源访问权限的几种方式：

访问权限类型	我为什么使用这种方式？	怎样获取更多信息？
Amazon 账户中用户的访问权限	您想要在您的 Amazon 账户的伞形结构下添加用户，并希望使用 IAM 创建用户并管理他们的权限。	要了解如何使用 Amazon Web Services Management Console 在您的 Amazon 账户中创建用户和管理其权限，请参阅 IAM 入门 (p. 18) 。 要了解如何使用 IAM API 或 Amazon Command Line Interface 在您 Amazon 账户下创建用户，请参阅 创建您的第一个 IAM 管理员用户和用户组 (p. 19) 。 有关使用 IAM 用户的更多信息，请参阅 IAM 身份 (用户、用户组和角色) (p. 68) 。
通过您的认证系统和 Amazon 之间的联合身份验证实现非 Amazon 用户访问	您的身份和认证系统中有非 Amazon 用户，他们需要访问您的 Amazon 资源。	要了解如何使用会话令牌通过与您的公司目录之间的联合身份验证授予用户访问您的 Amazon 账户资源的权限，请转至 IAM 临时安全凭证 (p. 300) 。有关 Amazon Security Token Service API 的信息，请转到 Amazon Security Token Service API 参考 。
Amazon 账户间的跨账户访问权限	您想要与其他 Amazon 账户下的用户共享对特定 Amazon 资源的访问权限。	要了解如何使用 IAM 向其他 Amazon 账户授权，请参阅 角色术语和概念 (p. 158) 。

我是否需要注册 IAM？

如果您目前没有 Amazon 账户，则需要创建一个账户才能使用 IAM。您无需专门注册使用 IAM。使用 IAM 不会产生任何费用。

Note

IAM 仅适用于与 IAM 集成的 Amazon 产品。有关支持 IAM 的服务的列表，请参阅 [使用 IAM 的 Amazon 服务 \(p. 908\)](#)。

注册 Amazon

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

在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

其他资源

以下资源可以帮助您使用 IAM 完成任务。

- 管理 Amazon 账户凭证：Amazon 一般参考 中的 [Amazon 安全凭证](#)
- 开始并详细了解有关 [什么是 IAM？ \(p. 1\)](#) 的信息
- 设置命令行接口 (CLI) 以与 IAM 一起使用。有关跨平台 Amazon CLI，请参阅 [Amazon Command Line Interface 文档](#) 和 [IAM CLI 参考](#)。您还可以使用 Windows PowerShell 管理 IAM；请参阅 [Amazon Tools for Windows PowerShell 文档](#) 和 [IAM Windows PowerShell 参考](#)。
- 下载 Amazon 开发工具包以便以编程方式访问 IAM：[用于 Amazon Web Services 的工具](#)
- 获取常见问题解答：[Amazon Identity and Access Management 常见问题](#)
- 获取技术支持：[Amazon Support 中心](#)
- 获取高级技术支持：[Amazon Premium Support 中心](#)
- 查找 Amazon 术语的定义：[Amazon Web Services 术语表](#)
- 获取社群支持：[IAM 开发论坛](#)
- 联系 Amazon：[联系我们](#)

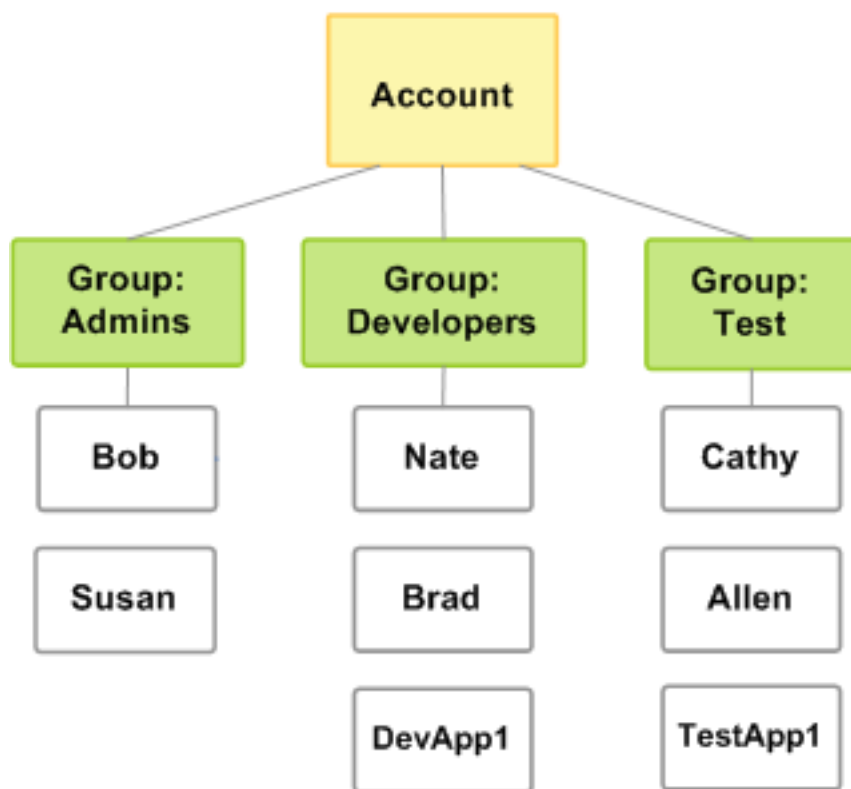
IAM 入门

本主题介绍如何在 Amazon 账户中创建 Amazon Identity and Access Management (IAM) 用户，以准许访问您的 Amazon 资源。首先，您将学习在创建组和用户之前应了解的 IAM 概念；然后，您将详细了解如何使用 Amazon Web Services Management Console 执行必要的任务。第一个任务是设置 Amazon 账户的管理员组。Amazon 账户中，管理员组不是必需的，但我们强烈建议您创建它。

Note

这套文档主要介绍 IAM 服务。要了解如何使用 Amazon 以及使用多种服务来解决构建和启动您的第一个项目等问题，请参阅[入门资源中心](#)。

在下图所示的简单示例中，Amazon 账户有三个组。一个组由一系列具有相似责任的用户组成。在此示例中，一个组为管理员组（名为 Admins）。另外还有一个 Developers 组和一个 Test 组。每个组均包含多个用户。尽管图中并未列明，但每个用户可处于多个组中。您不得将组置于其他组中。您可使用策略向组授予权限。



在随后的流程中，您需要执行下列任务：

- 创建管理员组并向该组提供访问您 Amazon 账户的所有资源的权限。
- 为您自己创建一个用户并将该用户添加到管理员组。
- 为您的用户创建密码，以便可以登录 Amazon Web Services Management Console。

您需要授予管理员组权限，以访问 Amazon 账户内所有可用的资源。可用的资源是指您使用或注册的任何 Amazon 产品。管理员组中的用户也可以访问您的 Amazon 账户信息，Amazon 账户的安全凭证除外。

主题

- [创建您的第一个 IAM 管理员用户和用户组 \(p. 19\)](#)
- [创建您的第一个 IAM 委派用户和组 \(p. 22\)](#)
- [IAM 用户如何登录您的 Amazon 账户 \(p. 24\)](#)
- [IAM 控制台搜索 \(p. 26\)](#)

创建您的第一个 IAM 管理员用户和用户组

Important

如果您发现此页面是因为您正在寻找有关 Product Advertising API 的信息以在您的网站上销售亚马逊产品，请参阅 [Product Advertising API 5.0 文档](#)。

作为 [最佳实践 \(p. 753\)](#)，请勿在不必要时使用 Amazon Web Services 账户根用户执行任务。而是应为需要管理员访问权限的每个人创建新的 IAM 用户。然后，通过将这些用户放入到一个您附加了 AdministratorAccess 托管策略的“管理员”用户组中，使这些用户成为管理员。

之后，管理员用户组中的用户应为 Amazon 账户设置用户组、用户等。所有将来的交互均应通过 Amazon 账户的用户以及他们自己的密钥进行，而不应使用根用户。但是，要执行一些账户和服务管理任务，您必须使用根用户凭证登录。要查看需要您以根用户身份登录的任务，请参阅 [Amazon 需要账户根用户凭证的任务](#)。

创建管理员 IAM 用户和用户组（控制台）

此过程将介绍如何使用 Amazon Web Services Management Console 自行创建 IAM 用户，并将该用户添加到具有已附加托管策略中的管理权限的用户组。

自行创建管理员用户并将该用户添加到管理员用户组（控制台）

1. 选择 Root user（根用户）并输入您的 Amazon Web Services 账户电子邮件地址，以账户拥有者身份登录 [IAM 控制台](#)。在下一页上，输入您的密码。

Note

强烈建议您遵守以下使用 **Administrator** IAM 用户的最佳实践，妥善保存根用户凭证。只在执行少数[账户和服务管理任务](#)时才作为根用户登录。

2. 为您创建的 IAM 管理员用户启用对账单数据的访问，如下所示：
 - a. 在导航栏中，选择账户名称，然后选择 Account（账户）。
 - b. 选择 IAM 用户和角色访问账单信息的权限旁边的编辑。您必须以根用户身份登录，才能在账户页面上显示此部分。
 - c. 选中 Activate IAM Access（激活 IAM 访问权限）旁边的复选框，然后选择 Update（更新）。
 - d. 在导航栏上，选择 Services（服务），然后选择 Security, Identity, & Compliance（安全、身份与合规）下的 IAM 以返回到 IAM 控制面板。
3. 在导航窗格中，选择 Users（用户），然后选择 Add users（添加用户）。
4. 在 Details（详细信息）页面上，执行以下操作：
 - a. 对于 User name，键入 **Administrator**。
 - b. 选中 Amazon Web Services Management Console access（控制台访问）的复选框，选择 Custom password（自定义密码），然后在文本框中键入新密码。
 - c. 默认情况下，Amazon 将强制新用户在首次登录时创建新密码。您可以选择清除 User must create a new password at next sign-in（用户必须在下次登录时创建新密码）旁边的复选框，以允许新用户在登录后重置其密码。
 - d. 选择 Next: Permissions（下一步：权限）。
5. 在 Permissions（权限）页面上，执行以下操作：

- a. 选择 Add user to group。
- b. 选择创建组。
- c. 在 Create group (创建组) 对话框中，对于 Group name (组名称)，键入 **Administrators**。
- d. 选中 AdministratorAccess 策略的复选框。
- e. 选择创建组。
- f. 返回到带有用户组列表的页面，选中您的新用户组所对应的复选框。如果列表中未显示新组，请选择 Refresh (刷新)。
- g. 请选择下一步：标签。
6. (可选) 在 Tags (标签) 页面上，通过以键值对的形式附加标签来向用户添加元数据。有关更多信息，请参阅[标记 IAM 资源 \(p. 276\)](#)。
7. 选择 Next: Review (下一步: 审核)。验证要添加到新用户的用户组成员资格。如果您已准备好继续，请选择 Create user (创建用户)。
8. (可选) 在 Complete (完成) 页面上，您可以下载带有登录信息的 .csv 文件，或者向用户发送带有登录说明的电子邮件。

您可使用此相同的流程创建更多的用户组 and 用户，并允许您的用户访问 Amazon 账户资源。要了解有关使用限制用户对特定 Amazon 资源的权限的策略的信息，请参阅[适用于 Amazon 资源的 Access Management \(p. 353\)](#)和[IAM 基于身份的策略示例 \(p. 391\)](#)。要在创建用户组之后向其中添加其他用户，请参阅[在 IAM 用户组中添加和删除用户 \(p. 154\)](#)。

创建 IAM 用户和用户组 (Amazon CLI)

如果执行了上一节中的步骤，则您已使用 Amazon Web Services Management Console 设置了一个管理员用户组，同时在您的 Amazon 账户中创建了 IAM 用户。此过程显示创建用户组的替代方法。

概述：设置管理员用户组

1. 创建一个用户组并为其提供名称 (例如 Admins)。有关更多信息，请参阅[创建用户组 \(Amazon CLI\) \(p. 20\)](#)。
2. 附加一个策略以便为用户组提供管理权限 (对所有 Amazon 操作和资源的访问权限)。有关更多信息，请参阅[将策略附加到用户组 \(Amazon CLI\) \(p. 21\)](#)。
3. 向用户组至少添加一个用户。有关更多信息，请参阅[在您的 Amazon 账户中创建 IAM 用户 \(p. 72\)](#)。

创建用户组 (Amazon CLI)

本部分将介绍如何在 IAM 系统中创建一个群组。

要求

安装 Amazon Command Line Interface (Amazon CLI)。有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的[安装 Amazon CLI](#)。

创建管理员用户组 (Amazon CLI)

1. 键入 `aws iam create-group` 命令，并使用您为用户组选择的名称。(可选) 您可以包含路径作为该用户组名的一部分。有关路径的更多信息，请参阅[易记名称和路径 \(p. 896\)](#)。名称可包含字母、数字以及以下字符：加号 (+)、等号 (=)、逗号 (,)、句点 (.)、at 符号 (@)、下划线 (_) 和连字符 (-)。名称不区分大小写，且最大长度可为 128 个字符。

在此示例中，您将创建名为 Admins 的组。

```
aws iam create-group --group-name Admins
```

```
{
  "Group": {
    "Path": "/",
    "CreateDate": "2014-06-05T20:29:53.622Z",
    "GroupId": "ABCDEFGHABCDEFGHABCDE",
    "Arn": "arn:aws:iam::123456789012:group/Admins",
    "GroupName": "Admins"
  }
}
```

2. 键入 `aws iam list-groups` 命令以列出您的 Amazon 账户中的用户组并确认该用户组已创建。

```
aws iam list-groups
{
  "Groups": [
    {
      "Path": "/",
      "CreateDate": "2014-06-05T20:29:53.622Z",
      "GroupId": "ABCDEFGHABCDEFGHABCDE",
      "Arn": "arn:aws:iam::123456789012:group/Admins",
      "GroupName": "Admins"
    }
  ]
}
```

响应中包括您的新用户组的 Amazon Resource Name (ARN)。ARN 是 Amazon 用于识别资源的标准格式。ARN 中的 12 位数字是您的 Amazon 账户 ID。您分配至用户组 (Admins) 的易记名称将在用户组 ARN 的末尾显示。

将策略附加到用户组 (Amazon CLI)

本节介绍了如何附加策略，从而允许用户组中的任何用户对 Amazon 账户内的任何资源执行任何操作。您可以通过将名为 `AdministratorAccess` 的 [Amazon 托管策略 \(p. 364\)](#) 附加到 Admins 用户组来执行此操作。有关策略的更多信息，请参阅[适用于 Amazon 资源的 Access Management \(p. 353\)](#)。

添加提供了完整管理员权限的策略 (Amazon CLI)

1. 键入 `aws iam attach-group-policy` 命令以将名为 `AdministratorAccess` 的策略附加到 Admins 用户组。该命令使用名为 `AdministratorAccess` 的 Amazon 托管策略的 ARN。

```
aws iam attach-group-policy --group-name Admins --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

如果命令执行成功，则没有应答。

2. 键入 `aws iam list-attached-group-policies` 命令以确认该策略已附加到 Admins 用户组。

```
aws iam list-attached-group-policies --group-name Admins
```

在响应中列出附加到 Admins 用户组的策略名称。类似如下的响应告诉您名为 `AdministratorAccess` 的策略已附加到 Admins 用户组：

```
{
  "AttachedPolicies": [
    {
      "PolicyName": "AdministratorAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
    }
  ],
}
```

```
"IsTruncated": false  
}
```

您可使用 `aws iam get-policy` 命令来确认特定策略的内容。

Important

在您完成管理员组的设置后，您必须在该用户组中至少添加一位用户。有关向用户组中添加用户的更多信息，请参阅 [在您的 Amazon 账户中创建 IAM 用户 \(p. 72\)](#)。

相关资源

有关 Amazon Web Services 一般参考中的相关信息，请参阅以下资源：

- [需要账户根用户的 Amazon 任务](#)

有关 IAM 用户指南 中的相关信息，请参阅以下资源：

- [使用上次访问的信息优化 Amazon 中的权限 \(p. 467\)](#)
- [IAM 教程：委托对账单控制台的访问权限 \(p. 28\)](#)

创建您的第一个 IAM 委派用户和组

要支持您的 Amazon 账户中的多个用户，您必须委派权限以允许其他人仅执行您要允许的操作。为此，您可以创建一个 IAM 用户组（其中具有这些用户所需的权限），然后在创建必要的组时将 IAM 用户添加到这些组。

创建委派的 IAM 用户和用户组（控制台）

您可以使用 Amazon Web Services Management Console 创建具有委派权限的 IAM 用户组。然后，为其他人创建 IAM 用户并将此用户添加到用户组。

为其他人创建委派用户组 and 用户（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在左侧的导航窗格中，选择策略。

如果这是您首次选择 Policies，则会显示 Welcome to Managed Policies 页面。选择开始使用。

3. 选择 Create policy (创建策略)。
4. 选择 JSON 选项卡，然后在窗口右侧，选择 Import managed policies (导入托管策略)。
5. 在 Import managed policies (导入托管策略) 窗口中，键入 **power** 以缩小策略列表。然后选择 PowerUserAccess 行。
6. 选择 Import (导入) 将策略显示在 JSON 选项卡。
7. 选择 Next: Tags (下一步: 标签)，然后选择 Next: Review (下一步: 审核)。
8. 在 Review policy (查看策略) 页面上，对于 Name (名称)，键入 **PowerUserExampleCorp**。对于描述，键入 **Allows full access to all services except those for user management**。然后，选择创建策略以保存您的工作。
9. 在导航窗格中，选择 Groups (组)，然后选择 Create New Group (创建新组)。
10. 在 User group name (用户组名称) 框中，键入 **PowerUsers**。
11. 在策略列表中，选中 PowerUserExampleCorp 旁边的复选框。

12. 选择创建组。
13. 在导航窗格中，选择 Users (用户)，然后选择 Add users (添加用户)。
14. 对于 User name，键入 **mary.major@examplecorp.com**。
15. 选择 Add another user (添加其他用户) 并键入 **diego.ramirez@examplecorp.com** 作为第二个用户。
16. 选中 Amazon Web Services Management Console access (控制台访问) 旁边的复选框，然后选择 Autogenerated password (自动生成的密码)。默认情况下，Amazon 将强制新用户在首次登录时创建新密码。选中 User must create a new password at next sign-in (用户必须在下次登录时创建新密码) 旁边的复选框。
17. 选择 Next: Permissions (下一步: 权限)。
18. 在 Set permissions (设置权限) 页面上，不要向用户添加权限。您将在用户确认他们已更改密码并登录后添加策略。
19. 选择下一步: 标签。
20. (可选) 通过以键值对的形式附加标签来向用户添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 [标记 IAM 资源 \(p. 276\)](#)。
21. 选择 Next: Review (下一步: 审核) 以查看要添加到新用户的用户组成员资格的列表。如果您准备好继续，请选择 Create users (创建用户)。
22. 下载或复制新用户的密码并安全地将其提供给用户。单独为您的用户提供指向您的 IAM 用户控制台页面的链接及其用户名。
23. 用户确认他们可以成功登录后，如有必要，请在导航窗格中选择 Users (用户)。然后选择其中一个用户的名称。
24. 在 Permissions (权限) 选项卡中，请选择 Add permissions (添加权限)。选择 Add user to group (将用户添加到组)，然后选中 PowerUsers (高级用户) 旁边的复选框。
25. 选择 Next: Review (下一步: 审核)，然后选择 Add permissions (添加权限)。

减少用户组权限

PowerUser 组的成员可以完全访问除提供用户管理操作的少数服务之外的所有服务 (如 IAM 和 Organizations)。经过预定义的不活动时段 (如 90 天) 后，您可以查看用户组成员已访问的服务。然后，您可以减少 PowerUserExampleCorp 策略的权限以仅包含您的团队所需的服务。

有关上次访问的信息的更多信息，请参阅[使用上次访问的信息优化 Amazon 中的权限 \(p. 467\)](#)。

查看上次访问的信息

等待预定义的不活动时段 (如 90 天) 经过。然后，查看您的用户或用户组上次访问的信息，以了解用户上次尝试访问您的 PowerUserExampleCorp 策略允许的服务的时间。

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，请选择 User groups (用户组)，然后选择 PowerUser 组名称。
3. 在用户组摘要页面上，请选择 Access Advisor (访问顾问) 选项卡。

上次访问的信息表显示用户组成员上次尝试访问每个服务的时间 (按时间顺序，从最近的尝试开始)。该表仅包含策略允许的服务。在此情况下，PowerUserExampleCorp 策略允许访问所有 Amazon 服务。

4. 查看此表并生成您的用户组成员最近访问过的服务的列表。

例如，假设在上个月内，您的团队仅访问了 Amazon EC2 和 Amazon S3 服务。但六个月前，他们访问了 Amazon EC2 Auto Scaling 和 IAM。您知道他们正在调查 EC2 Auto Scaling，但您认定不需要这样做。您还知道他们使用 IAM 创建角色以允许 Amazon EC2 访问 S3 存储桶中的数据。因此，您决定减少用户的权限，以仅允许访问 Amazon EC2 和 Amazon S3 服务。

编辑策略以减少权限

在查看上次访问的信息后，可以编辑策略以仅允许访问您的用户所需的服务。

使用数据以仅允许访问必要的服务

1. 在导航窗格中，选择 Policies (策略)，然后选择 PowerUserExampleCorp 策略名称。
2. 选择 Edit policy (编辑策略)，然后选择 JSON 选项卡。
3. 编辑 JSON 策略文档以仅允许所需的服务。

例如，编辑第一个包括 Allow 效果和 NotAction 元素的语句以仅允许 Amazon EC2 和 Amazon S3 操作。为此，请将其替换为具有 FullAccessToSomeServices ID 的语句。您的新策略将类似于以下示例策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FullAccessToSomeServices",
      "Effect": "Allow",
      "Action": [
        "ec2:*",
        "s3:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
        "iam>DeleteServiceLinkedRole",
        "iam:ListRoles",
        "organizations:DescribeOrganization"
      ],
      "Resource": "*"
    }
  ]
}
```

4. 为了支持[授予最低权限 \(p. 754\)](#)的最佳实践，请查看并更正[策略验证 \(p. 443\)](#)期间返回的任何错误、警告或建议。
5. 要进一步减少策略对特定操作和资源的权限，请在 CloudTrail Event history (事件历史记录) 中查看您的事件。在此处，您可以查看有关用户已访问的特定操作和资源的详细信息。有关更多信息，请参阅 Amazon CloudTrail 用户指南中的[在 CloudTrail 控制台中查看 CloudTrail 事件](#)。

IAM 用户如何登录您的 Amazon 账户

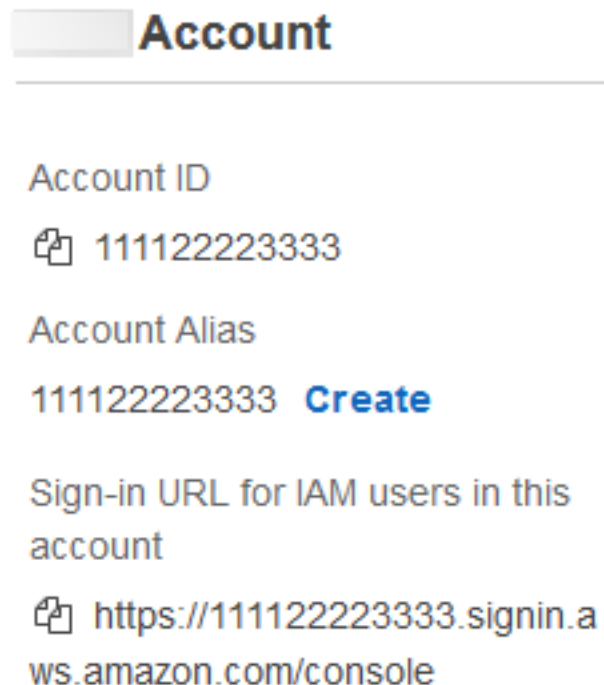
创建 IAM 用户（具有密码）后，这些用户可以登录到 Amazon Web Services Management Console。用户需要您的账户 ID 或别名进行登录。他们也可从包含您的账户 ID 的自定义 URL 登录。

Note

您是否考虑过使用 Amazon IAM Identity Center (successor to Amazon Single Sign-On) (IAM Identity Center)？您可以使用 IAM Identity Center 集中管理对多个 Amazon Web Services 账户的访问权限，并为用户提供受 MFA 保护的单点登录访问权限，可从一个位置访问其分配的所有账户。借助 IAM Identity Center，您可以在 IAM Identity Center 中创建和管理用户身份，或者轻松连接到现有的 SAML 2.0 兼容身份提供者。有关更多信息，请参阅《Amazon IAM Identity Center (successor to Amazon Single Sign-On) 用户指南》中的[什么是 IAM Identity Center？](#)。

在为您的账户创建登录 URL 前，您可以创建账户别名，使得 URL 包含您的账户名称而不是账户 ID。有关更多信息，请参阅[Amazon 账户 ID 及其别名](#) (p. 63)。

您可以在 IAM 控制台控制面板中找到账户的登录 URL。



要为您的 IAM 用户创建登录 URL，请使用以下模式：

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

IAM 用户还可以在以下终端节点登录并手动输入账户 ID 或别名，而不是使用您的自定义 URL：

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

控制台活动所需的权限

您账户中的 IAM 用户只能访问您在策略中专门允许的、适用于他们的 Amazon 资源。例如，可将策略附加到用户、他们所属的组、他们代入的角色或允许他们访问的资源。

要在控制台开展工作，用户必须有权限执行控制台执行的操作（例如列出和创建 Amazon 资源）。有关更多信息，请参阅[适用于 Amazon 资源的 Access Management](#) (p. 353) 和 [IAM 基于身份的策略示例](#) (p. 391)。

如果您账户中的用户需要编程访问，您可以为每个用户创建一个访问密钥对（访问密钥 ID 和秘密访问密钥）。有关更多信息，请参阅[管理访问密钥（控制台）](#) (p. 98)。

在 CloudTrail 中记录登录详细信息

如果您启用 CloudTrail 来记录登录事件，则必须了解 CloudTrail 记录事件的方式。CloudTrail 包括全局和区域日志条目。登录事件记录到 CloudTrail 中的位置取决于您用户登录的方式。有关详细信息，请参阅[使用 CloudTrail 记录 IAM 事件](#)。

IAM 控制台搜索

在浏览 IAM 管理控制台以管理各种 IAM 资源时，您通常需要查找访问密钥。或者，您可能需要浏览深度嵌套的 IAM 资源，以查找所需的内容。使用 IAM 控制台搜索页面作为更快的搜索选项。您可以查找与您的账户、IAM 实体（如用户、组、角色、身份提供商）、策略（按名称）等相关的访问密钥。

IAM 控制台搜索功能可查找以下所有项目：

- 与您的搜索关键字（用户、组、角色、身份提供商和策略）匹配的 IAM 实体名称
- 与搜索关键字匹配的任务

IAM 控制台搜索功能不返回有关 Access Analyzer 的信息。

搜索结果中的每一行都是一个有效链接。例如，您可以在搜索结果中选择用户名称，这样，您将转到该用户的详细信息页面。或者您也可以选择操作链接，例如 Create user，从而进入 Create User 页面。

Note

访问密钥搜索要求您在搜索框中输入完整的访问密钥 ID。搜索结果显示与该键关联的用户。您可在 此处 直接导航到该用户的页面，并且可以在该页面中管理其访问密钥。

使用 IAM 控制台搜索功能

使用 IAM 控制台中的 Search 页面查找与该账户相关的项目。

在 IAM 控制台中搜索项目

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Search。
3. 在 Search（搜索）框中，键入您的搜索关键字。
4. 在搜索结果列表选择一个链接以导航到控制台相应的部分。

IAM 控制台搜索结果中的图标

下面的图标指出了通过搜索找到的项目的类型：

图标	描述
	IAM 用户
	IAM 组
	IAM 角色
	IAM policy

图标	描述
	“创建任务”或“附加策略”等任务
	使用关键字 delete 搜索到的结果

搜索短语示例

您可以在 IAM 搜索中使用以下短语。将各个斜体词分别替换为您要查找的实际 IAM 用户、组、角色、访问密钥、策略或标识提供程序的名称。

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

IAM 教程

以下教程提供了 Amazon Identity and Access Management (IAM) 的常见任务的完整端到端过程。这些过程适用于实验室类型环境，使用了虚构的公司名称、用户名称等。其目的在于提供一般性指导。在未进行仔细审核并进行改进以满足组织环境的独特需求的情况下，这些过程不适合在您的生产环境中直接使用。

教程

- [IAM 教程：委托对账单控制台的访问权限 \(p. 28\)](#)
- [IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限 \(p. 31\)](#)
- [IAM 教程：创建和附加您的第一个客户托管策略 \(p. 40\)](#)
- [IAM 教程：根据标签定义访问 Amazon 资源的权限 \(p. 42\)](#)
- [IAM 教程：允许用户管理其凭证和 MFA 设置 \(p. 56\)](#)

IAM 教程：委托对账单控制台的访问权限

Amazon 账户所有者可以向需要查看或管理 Amazon 账户的 Amazon Billing and Cost Management 数据的特定 IAM 用户委派访问权限。下面的说明将帮助您设置一个预先测试过的方案。此方案会帮助您获得配置账单权限的实践经验，而无需担心影响您的主 Amazon 生产账户。如果您将托管策略附加到 IAM 用户而不是按照本教程操作，则必须首先在[步骤 1 \(p. 29\)](#)中激活对 Amazon Billing and Cost Management 控制台的访问权限。

此工作流程具有四个基本步骤。

步骤 1：激活对您的 Amazon 测试账户的账单数据的访问权限 (p. 29)

如果您创建单个 Amazon 账户，则仅 Amazon 账户所有者 ([Amazon Web Services 账户根用户 \(p. 337\)](#)) 才能查看和管理账单信息。在账户所有者激活 IAM 访问权限并附加向用户或角色提供账单操作的策略之前，IAM 用户无法访问账单数据。要查看需要您以根用户身份登录的其他任务，请参阅[需要账户根用户的 Amazon 任务](#)。

如果您使用 Amazon Organizations [创建成员账户](#)，则默认启用这项功能。

步骤 2：创建授予账单数据权限的 IAM policy (p. 29)

对您的账户启用账单访问权限后，您仍必须向特定 IAM 用户或用户组显式授予账单数据访问权限。您应当使用客户托管策略授予此访问权限。

步骤 3：将账单策略附加到用户组 (p. 30)

将策略附加到用户组时，该用户组的所有成员都会收到与该策略关联的完整访问权限集。在此场景中，您将新账单策略附加到只包含需要账单访问权限的用户的用户组。

步骤 4：测试对账单控制台的访问权限 (p. 30)

完成核心任务后，您即可测试策略。测试确保策略按预期方式运行。

先决条件

创建要在本教程中使用的测试 Amazon 账户。在此账户中创建两个测试用户和两个测试用户组，如下表所示。请确保为每个用户分配密码，以便在后面的步骤 4 中可以登录。

创建用户账户	创建和配置用户组账户	
FinanceManager	BillingFullAccessGroup	FinanceManager
FinanceUser	BillingViewAccessGroup	FinanceUser

步骤 1：激活对您的 Amazon 测试账户的账单数据的访问权限

首先，在 Amazon Billing and Cost Management 控制台中为您的测试用户激活对账单访问权限。

Note

如果您使用 Amazon Organizations [创建成员账户](#)，则默认启用这项功能。

激活 IAM 用户和角色对 Billing and Cost Management 控制台的访问权限

1. 使用您的根账户凭证（您用于创建 Amazon 账户的电子邮件地址和密码）登录 Amazon Web Services Management Console。
2. 在导航栏中，选择账户名称，然后选择 [Account](#)（账户）。
3. 选择 IAM 用户和角色访问账单信息的权限旁边的编辑。
4. 选中 Activate IAM Access（激活 IAM 访问权限）复选框以激活对 Billing and Cost Management 页面的访问权限。
5. 选择 Update（更新）。

您现在可使用 IAM policy 控制用户可访问的页面。

激活 IAM 用户访问权限后，您可附加 IAM policy 以授权或拒绝对特定账单功能的访问。有关使用策略授权 IAM 用户访问 Amazon Billing and Cost Management 功能的更多信息，请参阅 Amazon Billing 用户指南中的 [为账单和成本管理使用基于身份的策略（IAM policy）](#)。

步骤 2：创建授予账单数据权限的 IAM policy

接下来，创建自定义策略，以授予对 Billing and Cost Management 控制台中页面的查看权限和完整访问权限。有关 IAM 权限策略的一般信息，请参阅 [托管策略与内联策略 \(p. 364\)](#)。

创建授予账单数据权限的 IAM policy

1. 以具有管理员凭证的用户身份登录 Amazon Web Services Management Console。根据 IAM 最佳实践，请勿使用根用户凭证登录。有关更多信息，请参阅 [创建您的第一个 IAM 管理员用户和用户组 \(p. 19\)](#)。
2. 访问：<https://console.aws.amazon.com/iam/>，打开 IAM 控制台。
3. 在导航窗格中，选择 Policies（策略），然后选择 Create policy（创建策略）。
4. 在可视化编辑器选项卡上，选择选择服务以开始使用。然后选择 Billing Console（账单控制台）。
5. 按照以下步骤创建两个策略：

完全访问权限

- a. 选中 Manual actions（手动操作）下的 All Billing Console actions（所有账单控制台操作）旁的复选框。您不需要为该策略选择资源或条件。

- b. 请选择下一步：标签。
- c. 请选择下一步：审核。
- d. 在 Review policy (审核策略) 页上的 Name (名称) 旁边键入 **BillingFullAccess**，然后选择 Create policy (创建策略) 以保存该策略。

只读访问权限

- a. 重复步骤 3 和 4 (p. 29)。
- b. 选择选择操作，然后选中读取旁边的复选框。您不需要为该策略选择资源或条件。
- c. 请选择下一步：标签。
- d. 请选择下一步：审核。
- e. 在 Review policy (查看策略) 页面上，对于 Name (名称)，键入 **BillingViewAccess**。然后，选择创建策略以保存该策略。

要查看为用户授予 Billing and Cost Management 控制台访问权限的 IAM policy 中提供的每个权限的说明，请参阅[账单权限说明](#)。

步骤 3：将账单策略附加到用户组

现在您已经有了可用的自定义账单策略，可以将它们附加到之前创建的相应用户组。虽然可以将策略直接附加到用户或角色，但我们（根据 IAM 最佳实践）建议附加到用户组。

将账单策略附加到用户组

1. 在导航窗格中，选择 Policies 以显示您的 Amazon 账户可用的策略的完整列表。要将各策略附加到适当的用户组，请遵循以下步骤：

完全访问权限

- a. 在策略搜索框中键入 **BillingFullAccess**，然后选中策略名称旁边的复选框。
- b. 选择 Actions (操作)，然后选择 Attach (附加)。
- c. 在身份 (用户、用户组和角色) 搜索框中键入 **BillingFullAccessGroup**，选中用户组名称旁边的复选框，然后选择 Attach policy (附加策略)。

只读访问权限

- a. 在策略搜索框中键入 **BillingViewAccess**，然后选中策略名称旁边的复选框。
 - b. 选择 Actions (操作)，然后选择 Attach (附加)。
 - c. 在身份 (用户、用户组和角色) 搜索框中键入 **BillingViewAccessGroup**，选中用户组名称旁边的复选框，然后选择 Attach policy (附加策略)。
2. 注销控制台，然后执行[步骤 4：测试对账单控制台的访问权限](#) (p. 30)。

步骤 4：测试对账单控制台的访问权限

我们建议您以各测试用户身份登录来测试访问权限，以便了解用户可能获得的体验。通过以下步骤，使用两个测试账户登录来查看访问权限有何不同。

使用两个测试用户账户登录以测试账单访问权限

1. 使用 Amazon 账户 ID 或账户别名、您的 IAM 用户名和密码登录到 [IAM 控制台](#)。

Note

为方便起见，Amazon 登录页面使用浏览器 Cookie 记住您的 IAM 用户名和账户信息。如果您之前以其他用户身份登录，请选择页面底部的 Sign-in to a different account (登录到其他账户) 以返回主登录页面。在此处，您可以输入要重新导向到您账户 IAM 用户登录页面的 Amazon 账户 ID 或账户别名。

2. 通过下面提供的步骤使用每个账户登录以比较不同的用户体验。

完全访问权限

- a. 以用户 FinanceManager 的身份登录您的 Amazon 账户。
- b. 在导航栏上，选择 FinanceManager@<##### ID #>，然后选择我的账单控制面板。
- c. 浏览页面并选择不同的按钮以确保您有完全修改权限。

只读访问权限

- a. 以用户 FinanceUser 的身份登录您的 Amazon 账户。
- b. 在导航栏上，选择 FinanceUser@<##### ID #>，然后选择我的账单控制面板。
- c. 浏览页面。注意您是否可以正常显示成本、报告和账单数据。不过，如果您选择一个选项来修改值，会收到 Access Denied 消息。例如，在 Preferences 页面上，选中页面上的任意复选框，然后选择 Save preferences。控制台消息通知您需要 ModifyBilling 权限才能对该页面进行更改。

相关资源

有关 Amazon Billing 用户指南 中的相关信息，请参阅以下资源：

- [激活对账单和成本管理控制台的访问权限](#)
- [允许完全访问 Amazon 服务，但拒绝 IAM 用户访问 Billing and Cost Management 控制台。](#)
- [账单权限说明](#)

有关 IAM 用户指南 中的相关信息，请参阅以下资源：

- [托管策略与内联策略 \(p. 364\)](#)
- [控制用户访问 Amazon Web Services Management Console \(p. 76\)](#)
- [将策略附加到 IAM 用户组 \(p. 155\)](#)

摘要

现在您已成功完成向用户委派对 Billing and Cost Management 控制台的访问权限所需的全部步骤。因此，您已亲眼目睹用户账单控制台的体验。现在，您可以方便地在生产环境中实施此逻辑。

IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限

本教程将指导您如何使用角色来委派对您拥有的不同 Amazon 账户 (Production (生产) 和 Development (开发)) 中的资源的访问权限。您将与另一账户中的用户共享一个账户中的资源。通过以这种方式设置跨账户存取，您不需要在每个账户中创建单个 IAM 用户。此外，用户不必从一个账户注销并登录

另一个账户，以便访问不同的 Amazon 账户中的资源。配置角色后，您将了解如何从 Amazon Web Services Management Console、Amazon CLI 和 API 中使用角色。

Note

IAM 角色和基于资源的策略仅在单个分区内跨账户委派访问权限。例如，假定您在标准 aws 分区的美国西部（加利福尼亚北部）中有一个账户。您在 aws-cn 分区中国（北京）中也有一个账户。您不能使用中国（北京）的账户中 Amazon S3 基于资源的策略，来允许标准 aws 账户中用户的访问权限。

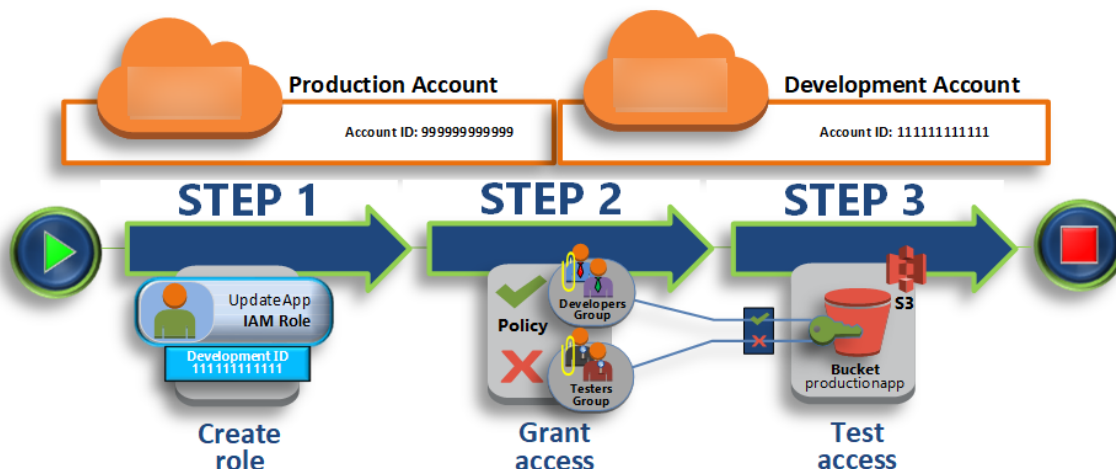
在本教程中，Production（生产）账户管理实时应用程序。开发人员和测试人员使用 Development（开发）账户作为自由测试应用程序的沙盒。在每个账户中，您将应用程序信息存储在 Amazon S3 存储桶中。您管理 Development（开发）账户中的 IAM 用户，该账户中有两个 IAM 用户组：Developer 和 Tester。两个用户组中的用户拥有在 Development 账户中工作的权限，能访问该账户中的资源。有时，开发人员必须更新 Production（生产）账户中的活动应用程序。开发人员将这些应用程序存储在名为 productionapp 的 Amazon S3 存储桶中。

在本教程结束时，您将拥有以下内容：

- Development 账户（可信账户）中能够代入 Production（生产）账户中的特定角色的用户。
- Production（生产）账户（可信账户）中能够访问特定 Amazon S3 存储桶的角色。
- Production（生产）账户中的 productionapp 存储桶。

开发人员可以在 Amazon Web Services Management Console 中使用该角色访问 Production（生产）账户中的 productionapp 存储桶。他们还可以使用通过该角色提供的临时凭证进行了身份验证的 API 调用来访问该存储桶。测试人员同样尝试使用该角色，但会失败。

此工作流程具有三个基本步骤：



步骤 1：在生产账户中创建角色 (p. 33)

首先，您使用 Amazon Web Services Management Console 在 Production（生产）账户（ID 号 999999999999）和 Development（开发）账户（ID 号 111111111111）之间建立信任。可通过创建名为 UpdateApp 的 IAM 角色角色开始。当您创建角色时，将 Development（开发）账户定义为受信任实体，并指定一个权限策略，允许受信任用户更新 productionapp 存储桶。

步骤 2：对角色授予访问权限 (p. 35)

在教程的此步骤中，您将修改 IAM 用户组策略，以便拒绝测试人员访问 UpdateApp 角色。这种情况下，因为 Testers 有 PowerUser 访问权限，我们必须明确 deny（拒绝）其使用该角色的权限。

步骤 3：通过切换角色测试访问权限 (p. 36)

最后，以开发人员的身份使用 `updateApp` 角色更新 Production（生产）账户中的 `productionapp` 存储桶。您可以了解如何通过 Amazon 控制台、Amazon CLI 和 API 访问角色。

先决条件

本教程假定您已准备好以下各项：

- 您可以使用的 Two（两个）单独的 Amazon 账户，一个代表 Development（开发）账户，另一个代表 Production（生产）账户。
- Development（开发）账户中的用户和用户组的创建和配置方式如下：

用户	用户组	权限
David	开发人员	两个用户均可以登录并使用 Development（开发）账户中的 Amazon Web Services Management Console。
Jane	测试人员	

- 您不需要在 Production（生产）账户中创建任何用户或用户组。
- 在 Production（生产）账户中创建 Amazon S3 存储桶。在本教程中，您可以将其称为 `ProductionApp`，但由于 S3 存储桶名称必须全局唯一，您必须使用具有其他名称的存储桶。

步骤 1：在生产账户中创建角色

您可以允许一个 Amazon 账户中的用户访问另一个 Amazon 账户中的资源。为此，请创建一个角色，该角色定义可以访问它的人员以及它向切换到它的用户授予的权限。

在教程的这一步中，您将在 Production（生产）账户中创建角色，并将 Development（开发）账户指定为受信任实体。此外，限制更改角色的权限，只有 `productionapp` 存储桶的读写权限。任何人只要获得使用该角色的权限，就能对 `productionapp` 存储桶进行读写。

在创建角色之前，您需要 Development（开发）Amazon 账户的账户 ID。每个 Amazon 账户具有账户 ID 分配的唯一标识符。

获取开发 Amazon 账户 ID

- 以 Development（开发）账户管理员身份登录 Amazon Web Services Management Console 并在 <https://console.aws.amazon.com/iam/> 上打开 IAM 控制台。
- 在导航栏中，依次选择 Support（支持）和 Support Center（支持中心）。您当前登录的 12 位账户号 (ID) 将显示在 Support Center（支持中心）导航窗格中。对于本文中的情况，您可以将账户 ID 111111111111 用于 Development（开发）账户。但是，如果您在测试环境中使用此方案，则应使用有效的账户 ID。

在可由 Development（开发）账户使用的生产账户中创建角色

- 以 Production（生产）账户管理员身份登录 Amazon Web Services Management Console 并打开 IAM 控制台。
- 创建角色之前，准备好定义角色所需权限的托管策略。您可在以后的步骤中将此策略附加到角色。

您想要设置针对 `productionapp` 存储桶的读写权限。尽管 Amazon 提供了一些 Amazon S3 托管策略，但这些策略并未提供对单个 Amazon S3 存储桶的读写访问权限。您可以创建自己的自定义策略。

在导航窗格中选择策略，然后选择创建策略。

3. 选择 JSON 选项卡，然后复制以下 JSON 策略文档中的文本。将该文本粘贴到 JSON 文本框中，并将资源 ARN (`arn:aws:s3:::productionapp`) 替换为与您的 Amazon S3 存储桶对应的真实资源 ARN。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::productionapp"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::productionapp/*"
    }
  ]
}
```

ListBucket 权限允许用户浏览 productionapp 存储桶中的对象。GetObject、PutObject、DeleteObject 权限允许用户查看、更新和删除 productionapp 存储桶中的内容。

4. 解决[策略验证 \(p. 443\)](#)过程中生成的任何安全警告、错误或常规警告，然后选择 Review policy (查看策略)。

Note

您可以随时在 Visual editor (可视化编辑器) 和 JSON 选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy (查看策略)，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅[调整策略结构 \(p. 871\)](#)。

5. 在 Review (查看) 页面上，输入 **read-write-app-bucket** 作为策略名称。查看策略摘要以查看您的策略授予的权限，然后选择创建策略以保存您的工作。

新策略会显示在托管策略列表中。

6. 在导航窗格中，选择 Roles，然后选择 Create role。
7. 请选择 An Amazon account (Amazon 账户) 角色类型。
8. 对于 Account ID (账户 ID)，键入 Development (开发) 账户 ID。

本教程针对 Development (开发) 账户使用示例账户 ID **111111111111**。您应使用有效的账户 ID。如果使用无效的账户 ID，如 **111111111111**，IAM 不会让您创建新角色。

现在，您不必要求外部 ID 或要求用户拥有多重身份验证 (MFA) 就可以担任该角色。将这些选项保持未选中状态。有关更多信息，请参阅[在 Amazon 中使用多重身份验证 \(MFA\) \(p. 104\)](#)。

9. 选择 Next: Permissions 设置将与角色关联的权限。
10. 选中您之前创建的策略旁边的复选框。

提示

对于 Filter，选择 Customer managed 对列表进行筛选，使其只包含您创建的策略。这会隐藏 Amazon 创建的策略，更容易找到您所需要的策略。

然后选择下一步。

11. (可选) 通过以键值对的形式附加标签来向用户添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 [标记 IAM 资源 \(p. 276\)](#)。
12. (可选) 对于 Description (描述)，输入新角色的描述。
13. 检查角色后，选择 Create role。

UpdateApp 角色显示在角色列表中。

现在，您必须获取该角色的 Amazon Resource Name (ARN)，这是角色的唯一标识。当您修改 Developer 和 Tester 组的策略时，您将指定角色的 ARN 以授予或拒绝权限。

如何获取用于 UpdateApp 的 ARN

1. 在 IAM 控制台的导航窗格中，选择 Roles。
2. 在角色列表中，选择 UpdateApp 角色。
3. 在详细信息窗格的 Summary (摘要) 部分中，复制 Role ARN (角色 ARN) 值。

Production 账户的账户 ID 是 999999999999，因此角色 ARN 是 `arn:aws:iam::999999999999:role/UpdateApp`。确保为您的“生产”账户提供真实的 Amazon 账户 ID。

现在，您已建立 Production 和 Development 账户之间的信任关系。您通过在 Production 账户中创建一个角色来将 Development 账户标识为可信主体，从而实现这一点。您还可以定义切换到 UpdateApp 角色的用户可执行哪些操作。

接下来，修改这些用户组的权限。

步骤 2：对角色授予访问权限

现在，Testers 和 Developers 用户组成员都拥有相应权限，让他们能够自由测试 Development (开发) 账户中的应用程序。使用以下必要步骤添加权限以允许切换到角色。

修改 Developers 用户组以允许他们切换到 UpdateApp 角色

1. 以 Development (开发) 账户中的管理员身份登录，打开 IAM 控制台。
2. 选择 User groups (用户组)，然后选择 Developers。
3. 请选择 Permissions (权限) 选项卡，然后选择 Add permissions (添加权限)，接着选择 Create inline policy (创建内联策略)。
4. 请选择 JSON 选项卡。
5. 添加以下策略语句，允许对生产账户中的 AssumeRole 角色执行 UpdateApp 操作。请确保将 Resource 元素中的 **PRODUCTION-ACCOUNT-ID** 更改为 Production 账户的实际 Amazon 账户 ID。

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

```
"Action": "sts:AssumeRole",
"Resource": "arn:aws:iam::PRODUCTION-ACCOUNT-ID:role/UpdateApp"
}
}
```

Allow 效果显式允许 Developers 组访问生产账户中的 UpdateApp 角色。尝试访问该角色的任何开发人员都会成功。

6. 选择 Review policy (查看策略)。
7. 键入 Policy Name (策略名称)，例如 **allow-assume-s3-role-in-production**。
8. (可选) 对于描述，键入策略的描述。
9. 选择 Create policy (创建策略)。

在大多数环境中，可能不需要执行以下步骤。但是，如果您使用 PowerUserAccess，则某些组可能已能够切换角色。以下过程介绍如何向 Testers 组添加 "Deny" 权限，以确保他们无法代入角色。如果您的环境中不需要此过程，建议不要添加该权限。"Deny" 权限会让整个权限体系更为复杂、难以管理和理解。仅当您没有更好的选择时才使用 "Deny" 权限。

要修改测试人员用户组以拒绝担任 **UpdateApp** 角色的权限

1. 选择 User groups (用户组)，然后选择 Testers。
2. 请选择 Permissions (权限) 选项卡，然后选择 Add permissions (添加权限)，接着选择 Create inline policy (创建内联策略)。
3. 请选择 JSON 选项卡。
4. 添加以下策略语句以拒绝对 AssumeRole 角色执行的 UpdateApp 操作。请确保将 Resource 元素中的 **PRODUCTION-ACCOUNT-ID** 更改为 Production 账户的实际 Amazon 账户 ID。

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

Deny 效果显式拒绝 Testers 组访问生产账户中的 UpdateApp 角色。任何尝试访问该角色的测试人员都会获得一条拒绝访问信息。

5. 选择 Review policy (查看策略)。
6. 键入策略名称，如 **deny-assume-s3-role-in-production**。
7. 选择 Create policy (创建策略)。

Developers 用户组现在有相应权限，可以使用生产账户中的 UpdateApp 角色。Testers 用户组将无法使用 UpdateApp 角色。

接下来，您将了解开发人员 David 如何访问 Production 账户中的 productionapp 存储桶。David 可以从 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 访问存储桶。

步骤 3：通过切换角色测试访问权限

完成本教程的前两个步骤后，您将具有一个可以授予对 Production (生产) 账户中资源的访问权限的角色。您还将在 Development (开发) 账户中有一个用户组，该组中的用户可以使用该角色。此步骤讨论如何对从 Amazon Web Services Management Console、Amazon CLI 和 Amazon API 切换到该角色进行测试。

Important

只有当您以 IAM 用户或联合身份用户身份登录时，才能切换角色。另外，如果启动 Amazon EC2 实例来运行应用程序，则应用程序可以通过实例配置文件承担角色。以 Amazon Web Services 账户根用户身份登录后无法切换至角色。

切换角色（控制台）

如果 David 需要在 Amazon Web Services Management Console 中以 Production（生产）环境中工作，他可以通过使用 Switch Role（切换角色）实现。他指定账户 ID 或别名以及角色名称，他的权限会立即切换为该角色允许的权限。然后，他可以通过控制台使用 productionapp 存储桶，但是无法使用 Production（生产）中的其他任何资源。当 David 使用角色时，他无法使用其在 Development（开发）账户中的高级用户权限。这是因为，一次仅一组权限能够生效。

Important

使用 Amazon Web Services Management Console 切换角色仅适用于不需要 ExternalId 的账户。例如，假定您将您账户的访问权限授予第三方，并在权限策略的 Condition 元素中要求 ExternalId。在此情况下，第三方只能通过使用 Amazon API 或命令行工具访问您的账户。第三方不能使用控制台，因为它不能提供 ExternalId 值。有关此方案的更多信息，请参阅 Amazon 安全博客中的 [如何在向第三方授予对 Amazon 资源的访问权时使用外部 ID \(p. 164\)](#) 和 [如何启用对 Amazon Web Services Management Console 的跨账户存取](#)。

David 可以使用 IAM 提供的两种方法进入 Switch Role（切换角色）页：

- David 收到其管理员提供的一个链接，该链接指向一个预定义的“Switch Role”（切换角色）配置。该链接在 Create role 向导最后一页上或在跨账户角色的 Role Summary 页面上提供给管理员。选择该链接将引导 David 进入已填写 Account ID（账户 ID）和 Role name（角色姓名）字段的 Switch Role（切换角色）页面。David 只需选择 Switch Role（切换角色）。
- 管理员不会通过电子邮件发送该链接，而是发送账户 ID 号和角色名称值。要切换角色，David 必须手动输入这些值。以下步骤对此进行说明。

如何担任角色

1. David 使用其位于 Development（开发）用户组中常规用户登录到 Amazon Web Services Management Console。
2. 他们选择管理员通过电子邮件向其发送的链接。该会将 David 跳转至 Switch Role（切换角色）页面，其中已填写了账户 ID 或别名和角色名称信息。

—或者—

David 在导航栏上选择其姓名 [Identity（身份）菜单]，然后选择 Switch Roles（切换角色）。

如果这是 David 首次通过这种方式尝试访问“Switch Role（切换角色）”页面，则他会首先登录初始 Switch Role（切换角色）页面。此页面提供有关切换角色如何使用户可以跨 Amazon 账户管理资源的其他信息。David 必须选择此页面上的 Switch Role（切换角色）才能完成此过程的其余步骤。

3. 接下来，为了访问角色，David 必须手动键入生产账户 ID 号 (999999999999) 和角色名称 (UpdateApp)。

此外，David 还想监控当前 IAM 中处于活动状态的角色及相关权限。为了跟踪此信息，他在 Display Name（显示名称）文本框中键入 PRODUCTION，并选择红色选项，然后选择 Switch Role（切换角色）。

4. David 现在可以通过 Amazon S3 控制台使用 Amazon S3 存储桶，或使用 UpdateApp 角色有权限的其他任何资源。
5. 完成后，David 可返回其原始权限。为此，他们可以选择导航栏上的 PRODUCTION 角色显示名称，然后选择 Back to David @ 111111111111（返回 David @ 111111111111）。

6. 当 David 下次要切换角色并在导航栏中选择 Identity (身份) 菜单时, 他将看到 PRODUCTION 条目仍在上次的位置。他可以选择该条目立即切换角色, 无需重新输入账户 ID 和角色名称。

切换角色 (Amazon CLI)

如果 David 需要在 Production (生产) 环境中的命令行上工作, 他可以使用 [Amazon CLI](#) 做到这一点。他运行 `aws sts assume-role` 命令并传递角色 ARN 以获取该角色的临时安全凭证。然后, 他在环境变量中配置这些凭证, 使后续的 Amazon CLI 命令能够使用该角色的权限执行。当 David 使用角色时, 他不能同时使用他在 Development (开发) 账户中的高级用户权限, 因为同一时间只有一组权限有效。

请注意, 所有访问密钥和令牌都只是示例, 不能原样照用。请用您的实际环境的适当值替换。

如何担任角色

1. David 打开命令提示符窗口, 运行以下命令确认 Amazon CLI 客户端正在工作:

```
aws help
```

Note

David 的默认环境使用他通过 `David` 命令创建的默认配置文件中的 `aws configure` 用户凭证。有关更多信息, 请参阅《Amazon Command Line Interface 用户指南》中的[配置 Amazon Command Line Interface](#)。

2. 他通过运行以下命令开始角色切换过程, 以切换到 Production (生产) 账户中的 `UpdateApp` 角色。他从创建该角色的管理员处获得了角色 ARN。该命令还需要您提供一个会话名称, 您可以选择任何文本作为该名称。

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

然后 David 在输出中看到以下内容:

```
{
  "Credentials": {
    "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "SessionToken": "AQoDYXdzEGcaEXAMPLE2gsYULo
+Im5ZEXAMPLEeYjs1M2FUIgIJx9tQqNMBEXAMPLE
CvSRyh0FW7jEXAMPLEW+vE/7s1HRpXviG7b+qYf4nD00EXAMPLEmj4wxS04L/
uZEXAMPLECihzFB5lTYLto9dyBgSDy
EXAMPLE9/
g7QRUhZp4bqbEXAMPLENwGPYoj59pFA4lNKCikVgkREXAMPLEjlxQ7y52gekeVEXAMPLEDiB9ST3Uuysg
sKdEXAMPLE1TVastU1A0SKFEXAMPLEiYwCC/Cs8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP
+4eZScEXAMPLESnf87e
NhyDHq6ikBQ==",
    "Expiration": "2014-12-11T23:08:07Z",
    "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
  }
}
```

3. David 在输出的“Credentials” (凭证) 部分中看到了他们所需要的三个部分。

- AccessKeyId
- SecretAccessKey
- SessionToken

David 需要配置 Amazon CLI 环境，以在后续的调用中使用这些参数。有关各种凭证配置方法的信息，请参阅[配置 Amazon Command Line Interface](#)。您不能使用 `aws configure` 命令，因为它不支持捕获会话令牌。但是，您可手动将信息输入到配置文件中。由于这些是到期时间相对较短的临时凭证，将它们添加到您当前的命令行会话环境中是最简单的。

4. 为了将三个值添加到环境，David 将上一步的输出剪切并粘贴到以下命令中。您可能希望剪切并粘贴到简单文本编辑器中，以解决会话令牌输出中的换行问题。即使此处为清晰起见，将输出显示为换行格式，在添加时也必须是个长字符串形式。

Note

以下示例显示了 Windows 环境中的命令，其中“set”是创建环境变量的命令。在 Linux 或 MacOS 计算机上，您将改用“export”命令。该示例的其余部分对于三种环境均有效。

```
set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
set AWS_SESSION_TOKEN=AQoDYXdzEGcaEXAMPLE2gsYULo
+Im5ZEXAMPLEeYjs1M2FUIgIJx9tQqNMBEXAMPLECvS
Ryh0FW7jEXAMPLEW+vE/7s1HRpXviG7b+qYf4nD00EXAMPLEmj4wxS04L/
uZEXAMPLECihzFB5lTYLto9dyBgSDyEXA
MPLEKEY9/
g7QRUhZp4bqbEXAMPLENwGPYoj59pFA4lNKCikVgkREXAMPLEjlxQ7y52gekeVEXAMPLEDiB9ST3UusKd
EXAMPLE1TVastU1A0SKFEXAMPLEiywCC/Cs8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP
+4eZScEXAMPLENhykxiHen
DHq6ikBQ==
```

此时，以下所有命令都在这些凭证确定的角色的权限下运行。对 David 而言就是 UpdateApp 角色。

5. 运行该命令访问 Production 账户中的资源。在此示例中，David 使用以下命令列出其 S3 存储桶的内容。

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

因为 Amazon S3 存储桶名称通常是唯一的，所以无需指定拥有存储桶的账户 ID。要访问其他 Amazon 服务的资源，请参阅该服务的 Amazon CLI 文档，了解引用其资源所需使用的命令和语法。

使用 AssumeRole (Amazon API)

当 David 需要通过代码来更新 Production (生产) 账户时，他执行 AssumeRole 调用来担任 UpdateApp 角色。该调用返回临时凭证，他可以使用这些凭证访问 Production (生产) 账户中的 productionapp 存储桶。使用这些凭证，David 可以调用 API 以更新 productionapp 存储桶。但是，即使他在 Development (开发) 账户中拥有高级用户权限，也无法通过调用 API 来访问 Production (生产) 账户中的任何其他资源。

如何担任角色

1. David 把 AssumeRole 视为应用程序的一个段调用。他们必须指定 UpdateApp ARN：arn:aws:iam::999999999999:role/UpdateApp。

AssumeRole 调用返回的结果包含临时凭证以及 AccessKeyId 和 SecretAccessKey。它还包括一个 Expiration 时间，该时间指示凭证何时到期（届时您必须请求新的凭证）。

2. David 使用那些临时证书调用 s3:PutObject 升级 productionapp 存储段。他们会将凭证作为 AuthParams 参数传递给 API 调用。由于临时角色凭证只有 productionapp 存储桶的读写权限，因此对生产账户的任何其他操作都会被拒绝。

有关代码示例 (使用 Python)，请参阅[切换到 IAM 角色 \(Amazon API\) \(p. 250\)](#)。

相关资源

- 有关 IAM 用户的更多信息，请参阅 [IAM 身份（用户、用户组和角色）](#) (p. 68) 中的用户和组。
- 有关 Amazon S3 存储桶的更多信息，请参阅 Amazon Simple Storage Service 用户指南中的 [创建存储桶](#)。
- 要了解您信任区域之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅 [什么是 IAM Access Analyzer？](#)。

摘要

您已完成跨账户 API 访问教程。您创建了一个角色与另一个账户之间建立信任关系，并规定了受信任实体可以执行哪些操作。然后，您修改了组策略以控制哪些 IAM 用户可以访问该角色。最终，Development（开发）账户的开发人员可以使用临时凭证升级 Production（生产）账户中的 productionapp 存储桶。

IAM 教程：创建和附加您的第一个客户托管策略

在本教程中，您将使用 Amazon Web Services Management Console 创建一个 [客户托管策略](#) (p. 365)，然后将该策略附加到您的 Amazon 账户中的一个 IAM 用户。您创建的策略允许具有只读权限的 IAM 测试用户直接登录 Amazon Web Services Management Console。

此工作流程具有三个基本步骤：

步骤 1：创建策略 (p. 40)

默认情况下，IAM 用户没有权限来执行任何操作。未经过您的允许，他们无法访问 Amazon 管理控制台，也无法管理其中的数据。在该步骤中，您创建一个允许任何附加的用户登录到该控制台的客户托管策略。

步骤 2：附加策略 (p. 41)

将策略附加到用户时，该用户继承与该策略相关的所有访问权限。在此步骤中，您会将新策略附加到测试用户账户。

步骤 3：测试用户访问权限 (p. 41)

附加策略后，您即可以该用户身份登录并测试策略。

先决条件

要执行本教程中的步骤，您必须已具备以下内容：

- 可使用 IAM 用户身份登录的具有管理权限的 Amazon 账户。
- 未分配有如下权限或组成员资格的测试 IAM 用户：

用户名称	组	权限
PolicyUser	<无>	<无>

步骤 1：创建策略

在该步骤中，您创建一个允许任何附加用户（具有 IAM 数据的只读访问权限）登录 Amazon Web Services Management Console 的客户托管策略。

为测试用户创建策略

1. 使用您具有管理员权限的账户，通过以下网址登录到 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Policies (策略)。
3. 在内容窗格中，选择创建策略。
4. 选择 JSON 选项卡，然后复制以下 JSON 策略文档中的文本。将该文本粘贴到 JSON 文本框中。

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

5. 解决[策略验证 \(p. 443\)](#)过程中生成的任何安全警告、错误或常规警告，然后选择 Review policy (查看策略)。

Note

您可以随时在 Visual editor (可视化编辑器) 和 JSON 选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy (查看策略)，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅[调整策略结构 \(p. 871\)](#)。

6. 在 Review (查看) 页面上，键入 **UsersReadOnlyAccessToIAMConsole** 作为策略名称。查看策略摘要以查看您的策略授予的权限，然后选择创建策略以保存您的工作。

将在托管策略列表中显示新策略，并已准备好附加该策略。

步骤 2：附加策略

接下来，您会将刚刚创建的策略附加到测试 IAM 用户。

将策略挂载到测试用户

1. 在 IAM 控制台的导航窗格中，选择 Policies (策略)。
2. 在策略列表顶部的搜索框中，开始键入 **UsersReadOnlyAccessToIAMConsole** 直至您的策略出现。然后，选中列表中 UsersReadOnlyAccessToIAMConsole 旁边的框。
3. 请选择 Actions (操作) 按钮，然后选择 Attach (附加)。
4. 对于 Filter，选择 Users。
5. 在搜索框中，开始键入 **PolicyUser** 直至该用户在列表上可见。然后，选中列表中该用户旁边的框。
6. 选择 Attach Policy (附加策略)。

您已将策略挂载到 IAM 测试用户，这意味着现在该用户具有 IAM 控制台的只读访问权限。

步骤 3：测试用户访问权限

对于本教程，我们建议您以各测试用户身份登录来测试访问权限，以便能了解用户可能获得的体验。

使用测试用户账户登录以测试访问权限

1. 使用您的 PolicyUser 测试用户通过以下网址登录到 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 浏览控制台的各个页面并尝试创建新的用户或组。请注意，PolicyUser 可以显示数据，但无法创建或修改现有 IAM 数据。

相关资源

有关 IAM 用户指南 中的相关信息，请参阅以下资源：

- [托管策略与内联策略 \(p. 364\)](#)
- [控制用户访问 Amazon Web Services Management Console \(p. 76\)](#)

摘要

现在，您已成功完成创建和附加客户托管策略所需的所有步骤。因此，您可以使用测试账户登录 IAM 控制台，以了解用户可能获得的体验。

IAM 教程：根据标签定义访问 Amazon 资源的权限

基于属性的访问控制 (ABAC) 是一种授权策略，该策略基于属性来定义权限。在 Amazon 中，这些属性称为标签。您可以将标签附加到 IAM 资源（包括 IAM 实体（用户和角色））以及 Amazon 资源。您可以定义策略，这些策略使用标签条件键来根据主体的标签向其授予权限。当您使用标签控制对 Amazon 资源的访问时，可通过对 Amazon 策略进行较少更改来实现团队和资源的增长。ABAC 策略比传统 Amazon 策略更加灵活，后者需要您列出每个单独的资源。有关 ABAC 及其相对于传统策略的优势的更多信息，请参阅[什么是适用于 Amazon 的 ABAC？\(p. 12\)](#)。

Note

您必须为每个会话标签传递一个值。Amazon Security Token Service 不支持多值会话标签。

主题

- [教程概述 \(p. 43\)](#)
- [先决条件 \(p. 43\)](#)
- [步骤 1：创建测试用户 \(p. 44\)](#)
- [步骤 2：创建 ABAC 策略 \(p. 45\)](#)
- [步骤 3：创建角色 \(p. 47\)](#)
- [步骤 4：测试创建密钥 \(p. 48\)](#)
- [步骤 5：测试查看密钥 \(p. 49\)](#)
- [步骤 6：测试可扩展性 \(p. 51\)](#)
- [步骤 7：测试更新和删除密钥 \(p. 52\)](#)
- [摘要 \(p. 53\)](#)
- [相关资源 \(p. 53\)](#)
- [IAM 教程：将 SAML 会话标签用于 ABAC \(p. 53\)](#)

教程概述

本教程说明如何创建一个允许具有主体标签的 IAM 角色 角色访问具有匹配标签的资源的策略并测试该策略。当主体向 Amazon 发出请求时，将根据主体标签和资源标签是否匹配来授予其权限。此策略仅允许用户查看或编辑其作业需要的 Amazon 资源。

场景

假设您是一家名为 Example Corporation 的大公司的开发人员主管，同时也是一名经验丰富的 IAM 管理员。您熟悉 IAM 用户、角色和策略的创建和管理。您希望确保您的开发工程师和质量保证团队成员能够访问其所需的资源。您还需要一个随公司发展而扩展的策略。

您选择使用 Amazon 资源标签和 IAM 角色主体标签来为支持 ABAC 策略的服务实施该策略（从 Amazon Secrets Manager 开始）。要了解哪些服务支持基于标签的授权，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)。要了解可以在策略中使用哪些标记条件键以及每个服务的操作和资源，请参阅[Amazon 服务的操作、资源和条件键](#)。您可以对基于 SAML 的身份提供程序或 Web 身份提供程序进行配置，将[会话标签 \(p. 293\)](#)传递给 Amazon。当您的员工希望联合身份到 Amazon 中时，其属性将应用到 Amazon 中所得到的主体。然后，您可以使用 ABAC 来允许或拒绝基于这些属性的权限。要了解 SAML 联合身份中的会话标签与本教程的不同之处，请参阅[IAM 教程：将 SAML 会话标签用于 ABAC \(p. 53\)](#)。

您的工程和质量保证团队成员是 Pegasus 或 Unicorn 项目。您可以选择以下 3 个字符的项目和团队标签值：

- access-project = peg (对于 Pegasus 项目)
- access-project = uni (对于 Unicorn 项目)
- access-team = eng (对于工程团队)
- access-team = gas (对于质量保证团队)

此外，您选择需要 cost-center 成本分配标签以启用自定义 Amazon 账单报告。有关更多信息，请参阅 Amazon Billing and Cost Management 用户指南 中的[使用成本分配标签](#)。

重要决策摘要

- 员工使用 IAM 用户凭证进行登录，然后代入其团队和项目的 IAM 角色 角色。如果您的公司拥有与自己的身份系统，则您可以设置联合身份验证以允许员工在没有 IAM 用户的情况下代入角色。有关更多信息，请参阅[IAM 教程：将 SAML 会话标签用于 ABAC \(p. 53\)](#)。
- 将向所有角色附加同一策略。根据标签允许或拒绝操作。
- 员工可以创建新的资源，但前提是员工必须将相同的标签附加到应用于其角色的资源。这将确保员工能够在创建资源后进行查看。管理员不再需要使用新资源的 ARN 来更新策略。
- 员工可以读取其团队拥有的资源，无论项目如何。
- 员工可以更新和删除其团队和项目拥有的资源。
- IAM 管理员可以为新项目添加新角色。他们可以创建和标记新的 IAM 用户以允许访问相应的角色。管理员不需要编辑策略来支持新项目或团队成员。

在本教程中，您将标记每个资源，标记项目角色，并将策略添加到角色以允许前面所述的行为。生成的策略允许角色对使用同一项目和团队标签标记的资源进行 Create、Read、Update 和 Delete 访问。此策略还允许对使用同一团队标记的资源进行跨项目 Read 访问。

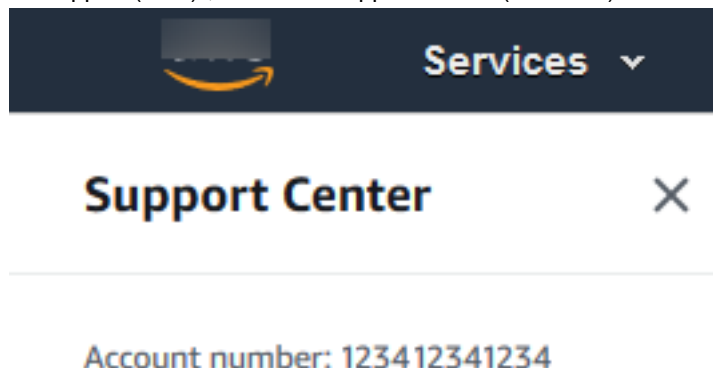
先决条件

要执行本教程中的步骤，您必须已具备以下内容：

- 可使用 IAM 用户身份登录的具有管理权限的 Amazon 账户。如果您有一个新账户并作为 Amazon Web Services 账户 根用户进行登录，则[创建 IAM 管理员用户 \(p. 19\)](#)。

- 您的 12 位账户 ID（用于在步骤 3 中创建角色）。

要使用 Amazon Web Services Management Console 查找 Amazon 账户 ID 号，请在右上角的导航栏上选择 Support（支持），然后选择 Support Center（支持中心）。账户号（ID）将显示在左侧的导航窗格中。



- 体验在 Amazon Web Services Management Console 中创建和编辑 IAM 用户、角色和策略。但是，如果您需要获得帮助以便记住 IAM 管理过程，请访问本教程提供的链接来查看分步说明。

步骤 1：创建测试用户

为了进行测试，请创建四个有权代入具有相同标签的角色的 IAM 用户。这可让您更轻松地向您的团队添加更多用户。在标记用户时，他们会自动获得访问权来代入正确角色。如果用户仅在一个项目和团队中工作，则无需将其添加到角色的信任策略中。

1. 创建以下名为 `access-assume-role` 的客户托管策略。有关创建 JSON 策略的更多信息，请参阅 [创建 IAM 策略 \(p. 438\)](#)。

ABAC 策略：代入任何 ABAC 角色，但前提是用户和角色标签匹配

以下策略允许用户在您的账户中代入任何带有 `access-` 名称前缀的角色。还必须使用与用户相同的项目、团队和成本中心来标记角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "TutorialAssumeRole",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::123456789012:role/access-*",
      "Condition": {
        "StringEquals": {
          "iam:ResourceTag/access-project": "${aws:PrincipalTag/access-project}",
          "iam:ResourceTag/access-team": "${aws:PrincipalTag/access-team}",
          "iam:ResourceTag/cost-center": "${aws:PrincipalTag/cost-center}"
        }
      }
    }
  ]
}
```

要将本教程扩展到大量用户，您可以将策略附加到一个组并将每个用户添加到该组。有关更多信息，请参阅 [创建 IAM 用户组 \(p. 153\)](#) 和 [在 IAM 用户组中添加和删除用户 \(p. 154\)](#)。

2. 创建以下 IAM 用户，附加 `access-assume-role` 权限策略，并添加以下标签。有关创建和标记新用户的更多信息，请参阅[创建 IAM 用户（控制台）](#) (p. 73)。

ABAC 用户

用户名称	用户标签
access-Arnav-peg-eng	<code>access-project = peg</code> <code>access-team = eng</code> <code>cost-center = 987654</code>
access-Mary-peg-qas	<code>access-project = peg</code> <code>access-team = qas</code> <code>cost-center = 987654</code>
access-Saanvi-uni-eng	<code>access-project = uni</code> <code>access-team = eng</code> <code>cost-center = 123456</code>
access-Carlos-uni-qas	<code>access-project = uni</code> <code>access-team = qas</code> <code>cost-center = 123456</code>

步骤 2：创建 ABAC 策略

创建以下名为 `access-same-project-team` 的策略。您将在以后的步骤中将此策略添加到角色。有关创建 JSON 策略的更多信息，请参阅[创建 IAM 策略](#) (p. 438)。

有关可针对本教程调整的其他策略，请参阅以下页面：

- [控制 IAM 主体进行的访问](#) (p. 388)
- [Amazon EC2：允许以编程方式和在控制台中启动或停止用户已标记的 EC2 实例](#) (p. 411)
- [EC2：基于匹配的主体和资源标签来启动或停止实例](#) (p. 412)
- [EC2：基于标签来启动或停止实例](#) (p. 411)
- [IAM：代入具有特定标签的角色](#) (p. 416)

ABAC 策略：仅在主体标签和资源标签匹配时访问 Secrets Manager 资源

以下策略允许主体创建、读取、编辑和删除资源，但前提是已使用与主体相同的键/值对标记这些资源。当主体创建资源时，必须添加 `access-project`、`access-team` 和 `cost-center` 标签以及与主体的标签匹配的值。该策略还允许添加可选的 `Name` 或 `OwnedBy` 标签。

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

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

```
}
```

此策略有何作用？

- `AllActionsSecretsManagerSameProjectSameTeam` 语句允许对所有相关资源执行此服务的所有操作，但前提是资源标签与主体标签匹配。通过将 `"Action": "secretsmanager:*"` 添加到策略，策略将在 Secrets Manager 增长时扩展。如果 Secrets Manager 添加一个新的 API 操作，则您无需将该操作添加到语句。该语句使用三个条件块实施 ABAC。仅在所有三个块都返回 `true` 时允许该请求。
 - 如果资源上存在指定的标签键，并且其值与主体的标签匹配，则此语句的第一个条件块将返回 `true`。对于不匹配的标签或不支持资源标记的操作，此块返回 `false`。要了解此块不允许哪些操作，请参阅 [Amazon Secrets Manager 的操作、资源和条件键](#)。此页面指明对 [Secret \(密钥\) 资源类型](#) 执行的操作支持 `secretsmanager:ResourceTag/tag-key` 条件键。某些 [Secrets Manager 操作](#) 不支持该资源类型，包括 `GetRandomPassword` 和 `ListSecrets`。您必须创建额外语句才能允许这些操作。
 - 如果请求中传递的每个标签键都包含在指定列表中，则第二个条件块将返回 `true`。通过将 `ForAllValues` 和 `StringEquals` 条件运算符结合使用来执行此操作。如果没有传递键或一组键的子集，则条件返回 `true`。对于不允许在请求中传递标签的 `Get*` 操作，这将允许该操作。如果请求者包含列表中没有的标签键，则条件返回 `false`。请求中传递的每个标签键均必须与该列表的一个成员匹配。有关更多信息，请参阅 [使用多个键和值 \(p. 960\)](#)。
 - 在以下情况下，第三个条件块将返回 `true`：请求支持传递标签；所有三个标签都存在；标签与主体标签值匹配。如果请求不支持传递标签，则此块也返回 `true`。这要归功于条件运算符中的 `...IfExists` ([p. 957](#))。如果在支持该块的操作期间未传递任何标签，或者标签键和值不匹配，则该块将返回 `false`。
- `AllResourcesSecretsManagerNoTags` 语句允许 `GetRandomPassword` 和 `ListSecrets` 操作，而第一个语句不允许这两项操作。
- 如果使用与资源相同的 `access-team` 标签来标记主体，则 `ReadSecretsManagerSameTeam` 语句允许只读操作。无论项目或成本中心标签如何，都允许此操作。
- `DenyUntagSecretsManagerReservedTags` 语句拒绝从 Secrets Manager 中删除带有以“access-”开头的键的标签的请求。这些标签用于控制对资源的访问，因此删除这些标签可能会删除权限。
- `DenyPermissionsManagement` 语句拒绝访问创建、编辑或删除 Secrets Manager 的基于资源的策略。这些策略可用于更改密钥的权限。

Important

此策略使用一种策略来允许服务的所有操作，但明确拒绝权限更改操作。拒绝操作将覆盖任何其他允许主体执行该操作的策略。这可能会产生意外结果。作为最佳实践，仅在没有允许该操作的情况下才使用显式拒绝。否则，允许各个操作的列表，默认情况下将拒绝不需要的操作。

步骤 3：创建角色

创建以下 IAM 角色并附加您在上一步中创建的 `access-same-project-team` 策略。有关创建 IAM 角色的更多信息，请参阅 [创建向 IAM 用户委派权限的角色 \(p. 215\)](#)。如果选择使用联合身份而不是 IAM 用户和角色，请参阅 [IAM 教程：将 SAML 会话标签用于 ABAC \(p. 53\)](#)。

ABAC 角色

作业函数	角色标签	角色名称	角色描述
项目 Pegasus 工厂	<code>access-project = peg</code> <code>access-team = eng</code> <code>cost-center = 987654</code>	<code>access-peg-engineering</code>	允许工程师读取所有工程资源以及创建和管理 Pegasus 工程资源。
项目 Pegasus 质量保证	<code>access-project = peg</code> <code>access-team = qas</code>	<code>access-peg-quality-assurance</code>	允许 QA 团队读取所有 QA 资源以及创建和管理所有 Pegasus QA 资源。

作业函数	角色标签	角色名称	角色描述
	<code>cost-center = 987654</code>		
项目 Unicorn 工程	<code>access-project = uni</code> <code>access-team = eng</code> <code>cost-center = 123456</code>	<code>access-uni-engineering</code>	允许工程师读取所有工程资源以及创建和管理 Unicorn 工程资源。
项目 Unicorn 质量保证	<code>access-project = uni</code> <code>access-team = qas</code> <code>cost-center = 123456</code>	<code>access-uni-quality-assurance</code>	允许 QA 团队读取所有 QA 资源以及创建和管理所有 Unicorn QA 资源。

步骤 4：测试创建密钥

附加到角色的权限策略允许员工创建密钥。仅在使用其项目、团队和成本中心标记密钥时允许这样做。通过在 Secrets Manager 中以用户身份登录、代入正确的角色并测试活动，确认您的权限按预期工作。

测试创建带有和不带有所需标签的密钥

1. 在您的主浏览器窗口中，仍以管理员用户身份登录，以便能在 IAM 中查看用户、角色和策略。使用浏览器无痕窗口或单独的浏览器来进行测试。在那里，以 `access-Arn timer-peg-eng` IAM 用户的身份登录并打开 Secrets Manager 控制台，地址：<https://console.aws.amazon.com/secretsmanager/>。
2. 尝试切换到 `access-uni-engineering` 角色。有关在 Amazon Web Services Management Console 中切换角色的更多信息，请参阅[切换到角色（控制台）](#)（p. 243）。

此操作失败，因为 `access-project` 和 `cost-center` 标签值与 `access-Arn timer-peg-eng` 用户和 `access-uni-engineering` 角色的不匹配。

3. 切换到 `access-peg-engineering` 角色。有关在 Amazon Web Services Management Console 中切换角色的更多信息，请参阅[切换到角色（控制台）](#)（p. 243）。
4. 使用以下信息存储新密钥。要了解如何存储密钥，请参阅 Amazon Secrets Manager 用户指南中的[创建基本密钥](#)。

1. 在 Select secret type (选择密钥类型) 部分中，选择 Other type of secrets (其他类型的密钥)。在两个文本框中，输入 `test-access-key` 和 `test-access-secret`。

您必须拥有额外的权限才能保存特定的 Amazon 服务的凭证。例如，要创建 Amazon RDS 数据库凭证，您必须有权描述 RDS 实例、RDS 集群和 Amazon Redshift 集群。

2. 为 Secret name (密钥名称) 字段输入 `test-access-peg-eng`。
3. 从下表添加不同的标签组合，并查看预期行为。
4. 选择 Store (存储) 以尝试创建密钥。当存储失败时，请返回之前的 Secrets Manager 控制台页面，并使用下表中的下一个标签集。允许使用最后一个标签集，并且将成功创建密钥。

test-access-peg-eng 角色的 ABAC 标签组合

access-project 标签值	access-team 标签值	cost-center 标签值	其他标签	预期行为
(无)	(无)	(无)	(无)	已拒绝，因为 <code>access-project</code> 标签值与角色的 <code>peg</code> 值不匹配。

access-project 标签值	access-team 标签值	cost-center 标签值	其他标签	预期行为
uni	eng	987654	(无)	已拒绝，因为 access-project 标签值与角色的 peg 值不匹配。
peg	gas	987654	(无)	已拒绝，因为 access-team 标签值与角色的 eng 值不匹配。
peg	eng	123456	(无)	已拒绝，因为 cost-center 标签值与角色的 987654 值不匹配。
peg	eng	987654	owner = Jane	已拒绝，因为策略不允许额外的标签 owner，即使所有三个所需标签都存在且其值与角色的值匹配也是如此。
peg	eng	987654	Name = Jane	已允许，因为所有三个所需标签都存在且其值与角色的值匹配。此外，允许您包含可选的 Name 标签。

5. 注销并针对以下每个角色和标签值重复此过程的前三个步骤。在此过程的第四步中，测试所选的任何一组缺少的标签、可选标签、不允许的标签和无效的标签值。然后，使用所需的标签创建具有以下标签和名称的密钥。

ABAC 角色和标签

用户名称	角色名称	密钥名称	密钥标签
access-Mary-peg-gas	access-peg-quality-assurance	test-access-peg-gas	access-project = peg access-team = gas cost-center = 987654
access-Saanvi-uni-eng	access-uni-engineering	test-access-uni-eng	access-project = uni access-team = eng cost-center = 123456
access-Carlos-uni-gas	access-uni-quality-assurance	test-access-uni-gas	access-project = uni access-team = gas cost-center = 123456

步骤 5：测试查看密钥

您附加到每个角色的策略允许员工查看标记有其团队名称的所有密钥，而不管其项目如何。通过在 Secrets Manager 中测试您的角色来确认您的权限按预期工作。

测试查看带有或不带有所需标签的密钥

1. 以下列某个 IAM 用户身份登录：

- access-Arnav-peg-eng
- access-Mary-peg-qas
- access-Saanvi-uni-eng
- access-Carlos-uni-qas

2. 切换到匹配的角色：

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

有关在 Amazon Web Services Management Console 中切换角色的更多信息，请参阅[切换到角色（控制台）](#) (p. 243)。

3. 在左侧导航窗格中，选择菜单图标以展开菜单，然后选择 Secrets (密钥)。
4. 无论您当前的角色如何，都应在表中看到所有四个密钥。这是预期情况，因为名为 access-same-project-team 的策略允许对所有资源执行 secretsmanager:ListSecrets 操作。
5. 选择其中一个密钥的名称。
6. 在密钥的详细信息页面上，您的角色标签将确定您是否能查看页面内容。将您的角色名称与您的密钥名称进行比较。如果它们共享同一团队名称，则 access-team 标签匹配。如果它们不匹配，则访问将被拒绝。

每个角色的 ABAC 密钥查看行为

角色名称	密钥名称	预期行为
access-peg-engineering	test-access-peg-eng	已允许
	test-access-peg-qas	已拒绝
	test-access-uni-eng	已允许
	test-access-uni-qas	已拒绝
access-peg-quality-assurance	test-access-peg-eng	已拒绝
	test-access-peg-qas	已允许
	test-access-uni-eng	已拒绝
	test-access-uni-qas	已允许
access-uni-engineering	test-access-peg-eng	已允许
	test-access-peg-qas	已拒绝
	test-access-uni-eng	已允许
	test-access-uni-qas	已拒绝
access-uni-quality-assurance	test-access-peg-eng	已拒绝
	test-access-peg-qas	已允许

角色名称	密钥名称	预期行为
	test-access-uni-eng	已拒绝
	test-access-uni-qas	已允许

- 从页面顶部的痕迹导航中，选择 Secrets (密钥) 以返回密钥列表。使用不同的角色重复此过程中的步骤来测试您是否能查看每个密钥。

步骤 6：测试可扩展性

在基于角色的访问控制 (RBAC) 上使用基于属性的访问控制 (ABAC) 的重要原因是可扩展性。当您的公司向 Amazon 添加新的项目、团队或人员时，您无需更新 ABAC 驱动型策略。例如，假设 Example Company 正在资助一个代号为 Centaur 的新项目。一个名叫 Saanvi Sarkar 的工程师将成为 Centaur 的工程师主管，同时该工程师还参与了 Unicorn 项目。Saanvi 还将审查 Peg 项目的工作。此外，新聘用了几名工程师，其中包括 Nikhil Jayashankar，他仅参与 Centaur 项目。

向 Amazon 添加新项目

- 以 IAM 管理员登录，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
- 在左侧导航窗格中，选择 Roles (角色)，然后添加名为 access-cen-engineering 的 IAM 角色。将 **access-same-project-team** 权限策略附加到角色并添加以下标签：
 - access-project = cen
 - access-team = eng
 - cost-center = 101010
- 在左侧导航窗格中，请选择用户。
- 添加一个名为 access-Nikhil-cen-eng 的新用户，并附加名为 access-assume-role 的策略。
- 使用 [步骤 4：测试创建密钥 \(p. 48\)](#) 和 [步骤 5：测试查看密钥 \(p. 49\)](#) 中的过程。在另一个浏览器窗口中，测试 Nikhil 是否能创建 Centaur 工程密钥并且是否能查看所有工程密钥。
- 在您以管理员身份登录的主浏览器窗口中，选择用户 access-Saanvi-uni-eng。
- 在 Permissions (权限) 选项卡上，删除 access-assume-role 权限策略。
- 添加以下名为 access-assume-specific-roles 的内联策略。有关将内联策略添加到用户的更多信息，请参阅 [为用户或角色嵌入内联策略 \(控制台\) \(p. 452\)](#)。

ABAC 策略：仅代入特定角色

此策略允许 Saanvi 代入 Pegasus 或 Centaur 项目的工程角色。由于 IAM 不支持多值标签，因此有必要创建此自定义策略。不能使用 access-project = peg 和 access-project = cen 标记 Saanvi 用户。此外，Amazon 授权模型不能同时与这两个值匹配。有关更多信息，请参阅 [在 IAM 和 Amazon STS 中进行标记的规则 \(p. 277\)](#)。相反，您必须手动指定她可以代入的两个角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "TutorialAssumeSpecificRoles",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": [
        "arn:aws:iam::123456789012:role/access-peg-engineering",
        "arn:aws:iam::123456789012:role/access-cen-engineering"
      ]
    }
  ]
}
```

```
}
```

- 使用 [步骤 4：测试创建密钥 \(p. 48\)](#) 和 [步骤 5：测试查看密钥 \(p. 49\)](#) 中的过程。在另一个浏览器窗口中，确认 Saanvi 可以代入这两个角色。检查她是否只能根据角色的标签为项目、团队和成本中心创建密钥。此外，确认她能够查看有关工程团队拥有的所有密钥（包括她刚创建的密钥）的详细信息。

步骤 7：测试更新和删除密钥

附加到角色的 `access-same-project-team` 策略允许员工更新和删除任何标记有项目、团队和成本中心的密钥。通过在 Secrets Manager 中测试您的角色来确认您的权限按预期工作。

测试更新和删除带有和不带有所需标签的密钥

- 以下列某个 IAM 用户身份登录：

- `access-Arnav-peg-eng`
- `access-Mary-peg-qas`
- `access-Saanvi-uni-eng`
- `access-Carlos-uni-qas`
- `access-Nikhil-cen-eng`

- 切换到匹配的角色：

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

有关在 Amazon Web Services Management Console 中切换角色的更多信息，请参阅 [切换到角色（控制台） \(p. 243\)](#)。

- 对于每个角色，请尝试更新密钥描述，然后尝试删除以下密钥。有关更多信息，请参阅 Amazon Secrets Manager 用户指南中的 [修改密钥](#) 和 [删除和还原密钥](#)。

每个角色的 ABAC 密钥更新和删除行为

角色名称	密钥名称	预期行为
<code>access-peg-engineering</code>	<code>test-access-peg-eng</code>	已允许
	<code>test-access-uni-eng</code>	已拒绝
	<code>test-access-uni-qas</code>	已拒绝
<code>access-peg-quality-assurance</code>	<code>test-access-peg-qas</code>	已允许
	<code>test-access-uni-eng</code>	已拒绝
<code>access-uni-engineering</code>	<code>test-access-uni-eng</code>	已允许
	<code>test-access-uni-qas</code>	已拒绝
<code>access-peg-quality-assurance</code>	<code>test-access-uni-qas</code>	已允许

摘要

您现已成功完成将标签用于基于属性的访问控制 (ABAC) 所需的所有步骤。您已了解如何定义标记策略。您已将该策略应用于主体和资源。您已创建并应用一个为 Secrets Manager 强制实施该策略的策略。您还了解到，在添加新项目 and 团队成员时，ABAC 可以轻松扩展。因此，您可以使用测试角色登录 IAM 控制台，并体验如何在 Amazon 中将标签用于 ABAC。

Note

您添加的策略仅允许在特定条件下执行操作。如果您将不同的策略应用于具有更广泛权限的用户或角色，则可能不会将操作限制为需要标记。例如，如果您使用 AdministratorAccessAmazon 托管策略向用户授予完全管理权限，则这些策略不会限制该访问。有关在涉及多个策略时如何确定权限的更多信息，请参阅[确定是允许还是拒绝账户内的请求 \(p. 974\)](#)。

相关资源

有关 IAM 用户指南 中的相关信息，请参阅以下资源：

- [什么是适用于 Amazon 的 ABAC？ \(p. 12\)](#)
- [Amazon 全局条件上下文密钥 \(p. 998\)](#)
- [创建您的第一个 IAM 管理员用户和用户组 \(p. 19\)](#)
- [创建 IAM 用户（控制台） \(p. 73\)](#)
- [创建向 IAM 用户委派权限的角色 \(p. 215\)](#)
- [标记 IAM 资源 \(p. 276\)](#)
- [使用标签控制对 Amazon 资源的访问 \(p. 389\)](#)
- [切换到角色（控制台） \(p. 243\)](#)
- [IAM 教程：将 SAML 会话标签用于 ABAC \(p. 53\)](#)

要了解如何监控账户中的标签，请参阅[使用无服务器工作流程和 Amazon CloudWatch Events 监控 Amazon 资源的标签更改](#)。

IAM 教程：将 SAML 会话标签用于 ABAC

基于属性的访问控制 (ABAC) 是一种授权策略，该策略基于属性来定义权限。在 Amazon 中，这些属性称为标签。您可以将标签附加到 IAM 资源（包括 IAM 实体（用户和角色））以及 Amazon 资源。将实体用于向 Amazon 发出请求时，实体成为主体并且包含标签。

您也可以在代入角色或联合身份用户时传递[会话标签 \(p. 293\)](#)。然后，您可以定义策略，这些策略使用标签条件键来根据主体的标签向其授予权限。当您使用标签控制对 Amazon 资源的访问时，可通过对 Amazon 策略进行较少更改来实现团队和资源的增长。ABAC 策略比传统 Amazon 策略更加灵活，后者需要您列出每个单独的资源。有关 ABAC 及其相对于传统策略的优势的更多信息，请参阅[什么是适用于 Amazon 的 ABAC？ \(p. 12\)](#)。

如果您的公司使用基于 SAML 的身份提供程序 (IdP) 管理公司用户身份，则可以在 Amazon 中使用 SAML 属性进行细粒度访问控制。属性包括成本中心标识符、用户电子邮件地址、部门分类和项目分配等。当您将这些属性作为会话标签传递时，可以根据这些会话标签控制对 Amazon 的访问权限。

要通过将 SAML 属性传递给会话主体来完成 [ABAC 教程 \(p. 42\)](#)，请完成 [IAM 教程：根据标签定义访问 Amazon 资源的权限 \(p. 42\)](#) 中的任务以及本主题中介绍的更改。

先决条件

要执行将 SAML 会话标签用于 ABAC 的步骤，您必须满足以下条件：

- 对基于 SAML 的 IdP 的访问权限，您在其中可以创建具有特定属性的测试用户。
- 可使用 IAM 用户身份登录的具有管理权限的 Amazon 账户。如果您有一个新账户并作为 Amazon Web Services 账户根用户进行登录，则[创建 IAM 管理员用户 \(p. 19\)](#)。
- 体验在 Amazon Web Services Management Console 中创建和编辑 IAM 用户、角色和策略。但是，如果您需要获得帮助以便记住 IAM 管理过程，请访问 ABAC 教程提供的链接来查看分步说明。
- 在 IAM 中设置基于 SAML 的 IdP 的经验。要查看详细信息以及指向详细 IAM 文档的链接，请参阅[使用 AssumeRoleWithSAML 传递会话标签 \(p. 297\)](#)。

步骤 1：创建测试 IAM 用户

跳过[步骤 1：创建测试用户 \(p. 44\)](#)中的说明。由于您在提供商中定义身份，因此无需为员工添加 IAM 用户。

步骤 2：创建 ABAC 策略

按照[步骤 2：创建 ABAC 策略 \(p. 45\)](#)中的说明在 IAM 中创建指定的托管策略。

步骤 3：创建和配置 SAML 角色

为 SAML 使用 ABAC 教程时，您必须执行额外的步骤来创建角色、配置 SAML IdP 和启用 Amazon Web Services Management Console 访问权限。有关更多信息，请参阅[步骤 3：创建角色 \(p. 47\)](#)。

步骤 3A：创建 SAML 角色

创建一个信任 SAML 身份提供程序和步骤 1 中创建的 test-session-tags 用户的角色。ABAC 教程使用具有不同角色标签的单独角色。由于您从 SAML IdP 传递会话标签，因此只需要一个角色。要了解如何创建基于 SAML 的角色，请参阅[创建用于 SAML 2.0 联合身份验证的角色（控制台） \(p. 231\)](#)。

将角色命名为 access-session-tags。将 access-same-project-team 权限策略附加到角色。编辑角色信任策略来使用以下策略。有关如何编辑角色的信任关系的详细说明，请参阅[修改角色（控制台） \(p. 261\)](#)。

以下角色信任策略允许您的 SAML 身份提供程序和 test-session-tags 用户代入该角色。在代入角色时，他们必须传递三个指定的会话标签。sts:TagSession 操作是允许传递会话标签所必需的。

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

```
]
}
```

AllowSamlIdentityAssumeRole 语句允许工程和质量保证团队的成员在从示例公司 IdP 将身份联合到 Amazon 中时代入此角色。ExampleCorpProvider SAML 提供商在 IAM 中定义。管理员已经设置 SAML 断言以传递三个必需的会话标签。断言可以传递额外的标签，但必须存在这三个标签。身份的属性可以具有 cost-center 和 access-project 标签的任何值。但是，access-team 属性值必须匹配 eng 或 gas，以指示位于工程或质量保证团队中的身份。

步骤 3B：配置 SAML IdP

配置您的 SAML IdP 以将 cost-center、access-project 和 access-team 属性作为会话标签传递。有关更多信息，请参阅[使用 AssumeRoleWithSAML 传递会话标签 \(p. 297\)](#)。

要将这些属性作为会话标签传递，请在 SAML 断言中包含以下元素。

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

步骤 3C：启用控制台访问

为联合身份 SAML 用户启用控制台访问权限。有关更多信息，请参阅[使 SAML 2.0 联合身份用户能够访问 Amazon Web Services Management Console \(p. 199\)](#)。

步骤 4：测试创建密钥

使用 access-session-tags 角色联合身份到 Amazon Web Services Management Console 中。有关更多信息，请参阅[使 SAML 2.0 联合身份用户能够访问 Amazon Web Services Management Console \(p. 199\)](#)。然后按照说明 [步骤 4：测试创建密钥 \(p. 48\)](#) 创建密钥。使用不同的 SAML 身份，该身份的属性与 ABAC 教程中指示的标签匹配。有关更多信息，请参阅[步骤 4：测试创建密钥 \(p. 48\)](#)。

步骤 5：测试查看密钥

按照[步骤 5：测试查看密钥 \(p. 49\)](#)中的说明查看在上一步中创建的密钥。使用不同的 SAML 身份，该身份的属性与 ABAC 教程中指示的标签匹配。

步骤 6：测试可扩展性

按照[步骤 6：测试可扩展性 \(p. 51\)](#)中的说明测试可扩展性。要执行此操作，您可在基于 SAML 的 IdP 中添加具有以下属性的新身份：

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

步骤 7：测试更新和删除密钥

按照[步骤 7：测试更新和删除密钥 \(p. 52\)](#)中的说明更新和删除密钥。使用不同的 SAML 身份，该身份的属性与 ABAC 教程中指示的标签匹配。

Important

删除您创建的所有密钥以避免产生费用。有关 Secrets Manager 中定价的详细信息，请参阅 [Amazon Secrets Manager 定价](#)。

摘要

您现已成功完成使用 SAML 会话标签和资源标签进行权限管理所需的所有步骤。

Note

您添加的策略仅允许在特定条件下执行操作。如果您将不同的策略应用于具有更广泛权限的用户或角色，则可能不会将操作限制为需要标记。例如，如果您使用 AdministratorAccessAmazon 托管策略向用户授予完全管理权限，则这些策略不会限制该访问。有关在涉及多个策略时如何确定权限的更多信息，请参阅[确定是允许还是拒绝账户内的请求 \(p. 974\)](#)。

IAM 教程：允许用户管理其凭证和 MFA 设置

您可以允许您的用户在 My Security Credentials (我的安全凭证) 页面上自行管理他们自己的多重身份验证 (MFA) 设备和凭证。您可以使用 Amazon Web Services Management Console 为用户配置凭证 (访问密钥、密码、签名证书和 SSH 公有密钥) 和 MFA 设备。这对少数用户非常有用。但随着用户数增加，这个任务很快会变得非常耗时。安全最佳实践指定用户应定期更改其密码并轮换其访问密钥。他们还应该删除或停用不再需要的凭证。我们还强烈建议他们使用 MFA 进行敏感操作。本教程旨在介绍如何实现这些最佳实践，而不给您的管理员带来负担。

本教程介绍如何允许用户访问 Amazon 服务，不过此操作仅限于用户使用 MFA 登录的情况。如果未使用 MFA 设备登录，则用户无法访问其他服务。

此工作流程具有三个基本步骤。

步骤 1：创建策略以强制 MFA 登录 (p. 57)

创建客户管理型策略以禁止除少量 IAM 操作以外的所有操作。这些例外允许用户在 My Security Credentials (我的安全凭证) 页面更改他们自己 MFA 凭证并管理他们的 MFA 设备。有关访问该页面的更多信息，请参阅[IAM 用户如何更改自己的密码 \(控制台 \) \(p. 95\)](#)。

步骤 2：将策略附加到您的测试用户组 (p. 57)

创建一个用户组，在使用 MFA 登录时，该组的成员具有所有 Amazon EC2 操作的完全访问权限。要创建此类用户组，请附加名为 AmazonEC2FullAccess 的 Amazon 托管策略和在步骤 1 中创建的客户托管策略。

步骤 3：测试您的用户的访问权限 (p. 58)

以测试用户身份登录，验证对 Amazon EC2 的访问是否被阻止，直至该用户创建 MFA 设备为止。之后，用户可以使用该设备进行登录。

先决条件

要执行本教程中的步骤，您必须已具备以下内容：

- 可使用 IAM 用户身份登录的具有管理权限的 Amazon 账户。
- 您在步骤 1 的策略中键入的账户 ID 号。

要查找您的账户 ID 号，请在页面顶部的导航栏上，选择 Support，然后选择 Support Center。您可以在该页面的 Support 菜单下找到您的账户 ID。

- [虚拟 \(基于软件的 \) MFA 设备 \(p. 106\)](#)、[FIDO 安全密钥 \(p. 109\)](#)或[基于硬件的 MFA 设备 \(p. 114\)](#)。

- 一个作为组成员的测试 IAM 用户，如下所示：

创建用户账户		创建和配置用户组账户		
MFAUser	仅选择 Amazon Web Services Management Console access 选项，然后分配密码。	EC2MFA	MFAUser	请勿向此用户组附加任何策略或授予任何权限。

步骤 1：创建策略以强制 MFA 登录

首先创建一个 IAM 客户托管策略，除 IAM 用户管理其自有凭证和 MFA 设备所需的权限外，该策略拒绝其他所有权限。

1. 以具有管理员凭证的用户身份登录 Amazon 管理控制台。根据 IAM 最佳实践，请勿使用 Amazon Web Services 账户根用户凭证登录。有关更多信息，请参阅[创建单独的 IAM 用户](#)。
2. 访问：<https://console.aws.amazon.com/iam/>，打开 IAM 控制台。
3. 在导航窗格中，选择 Policies (策略)，然后选择 Create policy (创建策略)。
4. 选择 JSON 选项卡，然后复制以下 JSON 策略文档中的文本：[Amazon：允许经过 MFA 身份验证的 IAM 用户在“My Security Credentials”（我的安全凭证）页面上管理自己的凭证 \(p. 395\)](#)。
5. 将该策略粘贴到 JSON 文本框中。解决策略验证过程中生成的任何安全警告、错误或常规警告，然后选择 Next: Tags (下一步：标签)。

Note

您可以随时在 Visual editor (可视化编辑器) 和 JSON 选项卡之间切换。但是，以上策略包含 NotAction 元素，该元素在可视化编辑器中不受支持。对于此策略，您在 Visual editor 选项卡上将看到一个通知。返回 JSON 选项卡以继续处理此策略。

此示例策略不允许用户在首次登录 Amazon Web Services Management Console 的同时重置密码。我们建议您在新用户登录并重置密码之前不要向他们授予权限。

6. 在存储库的 Tags (标签) 页面您可以将标签键值对添加到客户管理型策略 (可选)，然后选择 Next: Review (下一步：审核)。
7. 在 Review (查看) 页面上，键入 **Force_MFA** 作为策略名称。对于策略说明，键入 **This policy allows users to manage their own passwords and MFA devices but nothing else unless they authenticate with MFA.**。查看策略 Summary (摘要) 以查看您的策略授予的权限，然后选择 Create policy (创建策略) 以保存您的工作。

将在托管策略列表中显示新策略，并已准备好附加该策略。

步骤 2：将策略附加到您的测试用户组

接下来，您将两个策略附加到测试 IAM 用户组，将使用该组授予受 MFA 保护的权限。

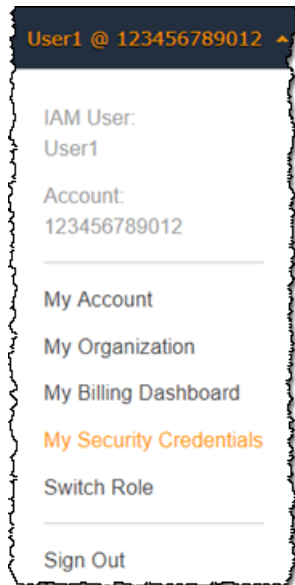
1. 在导航窗格中，选择 User groups (用户组)。
2. 在搜索框中键入 **EC2MFA**，然后在列表中选择组名称 (而不是选中复选框)。
3. 请选择 Permissions (权限) 选项卡，然后选择 Add permissions (添加权限)，接着选择 Attach policy (附加策略)。
4. 在 Attach permission policies to EC2MFA group (将权限策略附加到 EC2MFA 组) 页面上，在搜索框中键入 **EC2Full**。然后，选中列表中 AmazonEC2FullAccess 旁的复选框。暂不保存您的更改。
5. 在搜索框中键入 **Force**，然后选中列表中 Force_MFA 旁边的复选框。

6. 选择 Attach policies (附上策略)。

步骤 3：测试您的用户的访问权限

在本教程的这一部分中，您以测试用户身份登录并验证策略是否正常工作。

1. 使用上一部分中分配给您的密码，以 **MFAUser** 身份登录您的 Amazon 账户。使用 URL：`https://<alias or account ID number>.signin.aws.amazon.com/console`
2. 选择 EC2 打开 Amazon EC2 控制台，验证此用户没有执行任何操作的权限。
3. 在右上角的导航栏中，选择 MFAUser 用户名，然后选择我的安全凭证。



4. 现在添加一个 MFA 设备。在多重身份验证 (MFA) 部分中，选择分配 MFA 设备。

Note

可能会出现未授权您执行 `iam:DeleteVirtualMFADevice` 的错误。如果某个人以前开始将虚拟 MFA 设备分配给该用户并取消了该过程，则可能会发生这种情况。要继续，您或其他管理员必须删除用户的现有 MFA 设备。有关更多信息，请参阅[我没有权限执行：iam:DeleteVirtualMFADevice \(p. 866\)](#)。

5. 对于本教程，我们使用一个虚拟 (基于软件的) MFA 设备，例如，手机上的 Google Authenticator 应用程序。选择 Virtual MFA device (虚拟 MFA 设备)，然后单击 Continue (继续)。

IAM 将生成并显示虚拟 MFA 设备的配置信息，包括 QR 代码图形。此图形是秘密配置密钥的表示形式，适用于不支持 QR 代码的设备上的手动输入。

6. 打开您的虚拟 MFA 应用程序。(有关可以用作托管虚拟 MFA 设备的应用程序的列表，请参阅[虚拟 MFA 应用程序](#)。)如果虚拟 MFA 应用程序支持多个账户 (多个虚拟 MFA 设备)，请选择相应的选项以创建新账户 (新的虚拟 MFA 设备)。
7. 确定 MFA 应用程序是否支持 QR 代码，然后执行以下操作之一：
 - 在向导中，选择 Show QR code (显示 QR 代码)。使用此应用程序扫描 QR 代码。例如，您可选择摄像头图标或选择类似于 Scan code 的选项，然后使用设备的摄像头扫描此代码。
 - 在 Manage MFA Device (管理 MFA 设备) 向导中，选择 Show secret key (显示私有密钥)，然后在您的 MFA 应用程序中键入私有密钥。

完成操作后，虚拟 MFA 设备会开始生成一次性密码。

8. 在 Manage MFA Device (管理 MFA 设备) 向导的 MFA Code 1 (MFA 代码 1) 框中，键入虚拟 MFA 设备上当前显示的一次性密码。请等候 30 秒，以便设备生成新的一次性密码。然后在 MFA Code 2 (MFA 代码 2) 框中键入第二个一次性密码。选择 Assign MFA (分配 MFA)。

Important

生成代码之后立即提交您的请求。如果在生成代码后等待很长时间才提交请求，MFA 设备将成功与用户关联，但 MFA 设备不同步。这是因为基于时间的一次性密码 (TOTP) 很快会过期。这种情况下，您可以[重新同步设备](#) (p. 122)。

虚拟 MFA 设备现已准备好与 Amazon 一起使用。

9. 从控制台注销，然后重新以 **MFAUser** 身份登录。此时 Amazon 会提示输入您手机中的 MFA 代码。收到该代码后，请将代码键入框中，然后选择 Submit。
10. 选择 EC2 以再次打开 Amazon EC2 控制台。请注意，这次您可以看到所有信息，并且可以执行所需的任何操作。如果您以该用户身份转至任何其他控制台，则会看到指示访问被拒绝的消息。导致出现此问题的原因是，本教程中的策略仅向 Amazon EC2 授予访问权限。

相关资源

有关 IAM 用户指南 中的相关信息，请参阅以下资源：

- [在 Amazon 中使用多重身份验证 \(MFA\)](#) (p. 104)
- [为 Amazon 中的用户启用 MFA 设备](#) (p. 105)
- [使用 MFA 设备访问您的 IAM 登录页面](#) (p. 78)

以 IAM 用户或根用户的身份登录到 Amazon Web Services Management Console

Amazon Web Services Management Console 提供了一个基于 Web 的用户界面，您可以使用该界面来创建和管理 Amazon 资源。例如，您可以启动和停止 Amazon EC2 实例、创建 Amazon DynamoDB 表、创建 Amazon S3 存储桶等。

您必须先登录 Amazon 账户，然后才能使用 Amazon Web Services Management Console。您将用于登录 Amazon 账户的过程取决于您是哪种类型的 Amazon 用户。Amazon 中有两种不同类型的用户。您是账户所有者（根用户），或者是 IAM 用户。当使用用于创建 Amazon 账户的电子邮件地址和密码创建该账户时，将创建根用户。IAM 用户由根用户或 Amazon 账户中的 IAM 管理员创建。

如果您不记得凭证或在使用凭证进行登录时遇到问题，请参阅[Amazon 登录问题 \(p. 65\)](#)。

目录

- [以根用户身份登录 \(p. 60\)](#)
- [作为 IAM 用户登录 \(p. 61\)](#)
- [Amazon 账户 ID 及其别名 \(p. 63\)](#)
- [Amazon 登录或账户问题排查 \(p. 65\)](#)

以根用户身份登录

在以根用户身份登录 Amazon 账户之前，请确保您具有以下必需信息。

要求

- 用于创建 Amazon 账户的电子邮件地址。
- 根用户的密码。

以根用户身份登录 Amazon 账户

1. 打开：<https://console.aws.amazon.com/>。
2. 如果您之前没有使用此浏览器登录过，主登录页面将显示如下。选择 Root user（根用户），键入与您的账户关联的电子邮件地址，然后选择 Next（下一步）。

Sign in

☒ **Root user**
Account owner that performs tasks requiring unrestricted access. [Learn more](#)

☐ **IAM user**
User within an account that performs daily tasks. [Learn more](#)

Root user email address

username@example.com

Next

如果您之前使用此浏览器以根用户身份登录过，则浏览器可能会记住 Amazon 账户的电子邮件地址。如果是这样，您将看到下一步中显示的屏幕。

如果您之前使用此浏览器以 IAM 用户身份登录过，则浏览器可能会显示 IAM 用户登录页面。要返回主登录页面，请选择 Sign in using root user email (使用根用户电子邮件登录)。

3. 输入密码并选择 Sign in (登录)。

Root user sign in ⓘ

Email: user@example.com

Password [Forgot password?](#)

Sign in

[Sign in to a different account](#)

[Create a new](#) [account](#)

作为 IAM 用户登录

在以 IAM 用户身份登录 Amazon 账户之前，请确保您具有以下必需信息。如果您没有此信息，请与 Amazon 账户的管理员联系。

要求

- 下列情况之一：
 - 账户别名。
 - 12 位 Amazon 账户 ID。
- IAM 用户的用户名。
- IAM 用户的密码。

如果您是根用户或 IAM 管理员，并需要向 IAM 用户提供 Amazon 账户 ID 或 Amazon 账户别名，请参阅 [Amazon 账户 ID 及其别名 \(p. 63\)](#)。

如果您是 IAM 用户，则可以使用登录 URL 或主登录页面进行登录。

要使用 IAM 用户登录 URL 以 IAM 用户的身份登录到 Amazon 账户

1. 打开浏览器并输入以下登录 URL，用管理员提供的账户别名或账户 ID 替换 `account_alias_or_id`。

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

2. 输入您的 IAM 用户名和密码，然后选择 Sign in (登录)。

Sign in as IAM user

Account ID (12 digits) or account alias

IAM user name

Password

Sign in

[Sign in using root user email](#)

[Forgot password?](#)

使用主登录页面以 IAM 用户身份登录 Amazon 账户

1. 打开：<https://console.aws.amazon.com/>。
2. 如果您之前没有使用此浏览器登录过，则会显示主登录页面。选择 IAM user (IAM 用户)，输入账户别名或账户 ID，然后选择 Next (下一步)。

Sign in

☐ Root user

Account owner that performs tasks requiring unrestricted access. [Learn more](#)

☒ IAM user

User within an account that performs daily tasks. [Learn more](#)

Account ID (12 digits) or account alias

Next

如果您之前使用此浏览器以 IAM 用户身份登录过，则浏览器可能会记住 Amazon 账户的账户别名或账户 ID。如果是这样，您将看到下一步中显示的屏幕。

3. 输入您的 IAM 用户名和密码，然后选择 Sign in (登录)。

Sign in as IAM user

Account ID (12 digits) or account alias

IAM user name

Password

Sign in

[Sign in using root user email](#)

[Forgot password?](#)

如果您之前使用此浏览器以其他 Amazon 账户的 IAM 用户身份登录过，或者您需要以根用户身份登录，请选择使用 Sign in using root user email（根用户登录）以返回主登录页面。

Amazon 账户 ID 及其别名

要以 IAM 用户身份登录 Amazon 账户，您必须具有 Amazon 账户的账户别名或账户 ID。如果您已登录到 Amazon Web Services Management Console 或已使用账户凭证配置 Amazon CLI 或 Amazon SDK，则可以找到 Amazon 账户的账户别名或账户 ID。如果您无法登录，请向管理员询问登录所需的信息。

Note

账户别名不是密码，它们将显示在面向公众的登录页面 URL 中。请勿在账户别名中包含任何敏感信息。

主题

- [查找您的 Amazon 账户 ID \(p. 63\)](#)
- [关于账户别名 \(p. 64\)](#)
- [创建、删除和列出 Amazon 账户别名 \(p. 64\)](#)

查找您的 Amazon 账户 ID

可使用以下方法找到您的 Amazon 账户的账户 ID。

使用控制台查找您的账户 ID

在导航栏中，依次选择 Support（支持）和 Support Center（支持中心）。您当前登录的 12 位账户号 (ID) 将显示在 Support Center（支持中心）导航窗格中。

使用 Amazon CLI 查找您的账户 ID

使用以下命令查看您的用户 ID、账户 ID 和用户 ARN：

- `aws sts get-caller-identity`

使用 API 找到您的账户 ID

使用以下 API 查看您的用户 ID、账户 ID 和用户 ARN：

- [GetCallerIdentity](#)

关于账户别名

如果您希望在登录页面的 URL 用贵公司名称（或其他友好标识）取代您的 Amazon 账户 ID，可以创建一个账户别名。这一节介绍关于 Amazon 账户别名的信息，并列出于创建别名的 API 操作。

您的登录页面 URL 地址默认格式如下：

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

如果您的 Amazon 账户 ID 创建一个 Amazon 账户别名，您的登录页面 URL 地址格式如以下示例。

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

创建 Amazon 账户别名后，包含 Amazon 账户 ID 的原始 URL 地址依然有效，可以使用。

Tip

要在 Web 浏览器中为您的账户登录页面创建书签，应在书签条目中手动键入登录 URL。请勿使用 Web 浏览器的“将此页标记为书签”功能。

创建、删除和列出 Amazon 账户别名

您可以使用 Amazon Web Services Management Console、IAM API 或命令行接口创建或删除您的 Amazon 账户别名。

注意事项

- 您的 Amazon 账户只能有一个别名。如果您的 Amazon 账户创建新的别名，新别名将覆盖原有别名，包含原有别名的 URL 将失效。
- 账户别名必须在所有 Amazon Web Services 产品中是唯一的名称。它必须仅包含数字、小写字母和连字符。有关 Amazon 账户实体的限制条件的更多信息，请参阅 [IAM 与 Amazon STS 配额、名称要求和字符限制](#) (p. 902)。

创建、编辑和删除别名（控制台）

您可以在 Amazon Web Services Management Console 中创建、编辑和删除账户别名。

创建、编辑或删除账户别名（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Dashboard (控制面板)。
3. 在 Amazon Account (亚马逊云科技账户) 部分，找到 Account Alias (账户别名)，然后选择 Create (创建)。如果别名已存在，则选择 Edit (编辑)。
4. 键入要用于别名的名称，然后选择 Save changes (保存更改)。
5. 要删除别名，在 Account Alias (账户别名) 旁请选择 Delete (删除别名)，然后选择 Delete (删除)。登录 URL 会恢复使用 Amazon 账户 ID。

创建、删除和列出别名 (Amazon CLI)

要为 Amazon Web Services Management Console 登录页面 URL 创建别名，请运行以下命令：

- `aws iam create-account-alias`

要删除 Amazon 账户 ID 别名，请运行以下命令：

- `aws iam delete-account-alias`

要显示您的 Amazon 账户 ID 别名，请运行以下命令：

- `aws iam list-account-aliases`

创建、删除和列出别名 (Amazon API)

要为 Amazon Web Services Management Console 登录页面 URL 创建别名，请调用以下操作：

- `CreateAccountAlias`

要删除 Amazon 账户 ID 别名，请调用以下操作：

- `DeleteAccountAlias`

要显示您的 Amazon 账户 ID 别名，请调用以下操作：

- `ListAccountAliases`

Amazon 登录或账户问题排查

使用此处的信息可帮助您排查登录及其他 Amazon 账户问题。有关登录 Amazon 账户的分步说明，请参阅 [以 IAM 用户或根用户的身份登录到 Amazon Web Services Management Console \(p. 60\)](#)。

如果您在登录 Amazon.com 时遇到问题，请参阅 [Amazon 客户服务](#)。

问题

- [我的凭证不起作用 \(p. 65\)](#)
- [我需要我的 Amazon 账户 ID 或 Amazon 账户别名 \(p. 66\)](#)
- [我忘记了我的 IAM 用户名或密码 \(p. 66\)](#)
- [我忘记了我的 Amazon 账户的根用户密码 \(p. 67\)](#)
- [我无权访问我的 Amazon 账户的电子邮件 \(p. 67\)](#)
- [我需要变更我的 Amazon 账户的信用卡 \(p. 67\)](#)
- [我需要举报 Amazon 账户欺诈活动 \(p. 67\)](#)
- [我需要关闭我的 Amazon 账户 \(p. 67\)](#)

我的凭证不起作用

如果您无法登录到 Amazon Web Services Management Console，请尝试回忆您之前是如何成功访问 Amazon 的。

如果您根本不记得使用过密码

您之前可能是在不使用 Amazon 凭证的情况下访问 Amazon 的。这对于通过 IAM Identity Center 进行企业单点登录很常见。如果通过这种方式访问 Amazon，意味着您使用公司凭证访问 Amazon 账户或应用程序，而没有输入您的凭证。

- Amazon 访问门户 – 如果管理员允许您使用来自 Amazon 之外的凭证访问 Amazon，则需要有您的门户的 URL。检查您的电子邮件、浏览器收藏夹或浏览器历史记录，查找包含 `awsapps.com/start` 或 `signin.aws/platform/login` 的 URL。

例如，您的自定义 URL 可能包含某个 ID 或域，例如 `https://d-1234567890.awsapps.com/start`。如果您找不到门户的链接，请联系您的管理员。Amazon Web Services Support 无法帮助您找回这些信息。

如果您记得您的用户名和密码

您访问的页面可能不正确。尝试通过其他页面登录：

- 根用户登录页面 – 如果您创建了或拥有一个 Amazon Web Services 账户，并且需要执行受限制的操作，请在 [Amazon Web Services Management Console](#) 中输入您的账户电子邮件地址。要了解如何访问根用户，请参阅《Amazon 登录用户指南》中的 [以根用户身份登录](#)。如果您忘记了根用户密码，则无法重置密码。有关更多信息，请参阅《Amazon 登录用户指南》中的 [我忘记了我 Amazon 账户的根用户密码](#)。如果您忘记了您的 Amazon Web Services 账户电子邮件地址，请检查您的电子邮件收件箱中是否有来自 Amazon 的电子邮件。
- IAM 用户登录页面 – 如果您或其他人在单个 Amazon 账户创建了 IAM 用户，您必须知道该账户 ID 或别名。在 [Amazon Web Services Management Console](#) 中输入您的账户 ID 或别名、用户名和密码。要了解如何访问 IAM 用户登录页面，请参阅《Amazon 登录用户指南》中的 [以 IAM 用户身份登录](#)。如果您忘记了 IAM 用户密码，请参阅《Amazon 登录用户指南》中的 [我忘记了我 Amazon 账户的 IAM 用户密码](#)，了解有关重置 IAM 用户密码的信息。如果您忘记了账户编号，请搜索您的电子邮件、浏览器收藏夹或浏览器历史记录，查找包含 `signin.aws.amazon.com/` 的 URL。您的账户 ID 或别名将位于 URL 中的 "account=" 文本之后。如果您找不到您的账户 ID 或别名，请联系您的管理员。Amazon Web Services Support 无法帮助您找回这些信息。您只有在登录后才能看到您的账户 ID 或别名。
- Amazon 访问门户 – 如果管理员为 Amazon 设置了 Amazon IAM Identity Center (successor to Amazon Single Sign-On) 身份源，您必须使用您的用户名和密码进行登录。在此情况下，您需要有您的门户的 URL。检查您的电子邮件、安全密码存储、浏览器收藏夹或浏览器历史记录，查找包含 `awsapps.com/start` 或 `signin.aws/platform/login` 的 URL。例如，您的自定义 URL 可能包含 ID 或域（例如 `https://d-1234567890.awsapps.com/start`）。如果您找不到门户的链接，请联系您的管理员。Amazon Web Services Support 无法帮助您找回这些信息。

如果需要有关登录问题排查的更多帮助，请参阅 [如果我在登录或访问我的 Amazon Web Services 账户时遇到问题，该怎么办？](#)

下面的视频更详细介绍了在凭证失效时如何登录：

由于我的凭证不起作用而无法登录到我的 Amazon Web Services 账户 时该怎么办？

我需要我的 Amazon 账户 ID 或 Amazon 账户别名

如果您是 IAM 用户并且尚未登录，则必须向管理员询问 Amazon 账户 ID 或 Amazon 账户别名。您需要此信息以及您的 IAM 用户名和密码才能登录 Amazon 账户。

我忘记了我的 IAM 用户名或密码

如果您是 IAM 用户，管理员会提供您的凭证。如果您忘记了密码，则必须要求管理员重置密码。

出于安全目的，Amazon 无权查看、提供或更改您的凭证。

我忘记了我的 Amazon 账户的根用户密码

如果您是根用户并且丢失或忘记了 Amazon 账户的密码，可以重置您的密码。您必须知道用于创建 Amazon 账户的电子邮件地址，并且必须有权访问该电子邮件账户。有关更多信息，请参阅[重置您丢失或遗忘的 Amazon 密码或访问密钥](#) (p. 103)。

我无权访问我的 Amazon 账户的电子邮件

创建 Amazon 账户时，您会提供电子邮件地址和密码。这些是 Amazon Web Services 账户 根用户的凭证。如果您不确定与您的 Amazon 账户关联的电子邮件地址，请检查从 no-reply@amazon.com 到您企业的任何电子邮件地址的已保存通信，它们可能已用于开设 Amazon 账户。

如果您知道电子邮件地址，但无法再访问电子邮件，请首先尝试使用以下选项之一恢复对电子邮件的访问权限：

- 如果您拥有电子邮件地址的域，则可以恢复已删除的电子邮件地址。或者，您可以为电子邮件账户设置“全部捕获”，该功能将捕获发送到邮件服务器中不再存在的电子邮件地址的所有邮件，并将其重定向到另一个电子邮件地址。
- 如果账户上的电子邮件地址属于您的公司电子邮件系统，我们建议您联系 IT 系统管理员。它们也许能够帮助您重新获得电子邮件的访问权限。

如果您仍然无法登录 Amazon 账户，可以在 [Contact us](#) (联系我们) 中找到备选支持选项。选择 Still unable to log into your Amazon account (仍然无法登录 Amazon 账户)，然后选择一个可用的支持选项。

我需要变更我的 Amazon 账户的信用卡

若要更改您 Amazon 账户的信用卡，您必须能够登录。Amazon 设有保护，要求您证明自己是账户所有者。有关指导，请参阅 Amazon Billing 用户指南中的[管理您的信用卡付款方式](#)。

我需要举报 Amazon 账户欺诈活动

如果您怀疑有欺诈活动正在使用您的 Amazon 账户，并进行举报，请参阅 [How do I report abuse of Amazon resources](#) (如何报告亚马逊云科技资源滥用)。

如果您在 Amazon.com 上购买商品时遇到问题，请参阅 [Amazon 客户服务](#)。

我需要关闭我的 Amazon 账户

如果您有 Amazon 账户，您可以使用以下说明将其关闭：账单和成本管理用户指南中的[关闭账户](#)。

IAM 身份 (用户、用户组和角色)

登录 Amazon 时遇到问题？请确保使用的是适合您用户类型的 [Amazon 登录页面 \(p. 60\)](#)。如果您是 Amazon Web Services 账户根用户 (账户所有者)，则可以使用您在创建 Amazon 账户时设置的凭证登录 Amazon。如果您是 IAM 用户，则您的账户管理员可以向您提供用于登录 Amazon 的凭证。

Note

如果您需要请求支持，请不要使用此页面上的反馈链接，因为该表由 Amazon 文档团队 (而非 Amazon Support) 接收。相反，在 [Contact Us \(联系我们 \)](#) 页面上，展开 I cannot login to my Amazon Account (我无法登录我的亚马逊云科技账户)，然后选择 Request Support for Amazon Account Credentials (请求亚马逊云科技账户凭证的支持)。

Amazon Web Services 账户根用户或账户的 IAM 管理员可以创建 IAM 身份。IAM 身份提供对 Amazon 账户的访问权限。用户组是作为一个单元管理的 IAM 用户的集合。IAM 身份代表用户，可对身份进行验证，然后向其授予在 Amazon 中执行操作的权限。每个 IAM 身份都可以与一个或多个策略相关联。策略确定用户、角色或用户组成员可以在什么样的条件下对哪些 Amazon 资源执行哪些操作。

Amazon 账户根用户 (p. 337)

当您首次创建 Amazon Web Services (Amazon) 账户时，最初使用的是一个对账户中所有 Amazon 服务和资源具有完全访问权限的单点登录身份。此身份称为 Amazon 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。

Important

强烈建议您不使用根用户执行日常任务，即使是管理任务。相反，请遵循[仅使用根用户创建您的第一个 IAM 用户的最佳实践](#)。然后请妥善保存根用户凭证，仅用它们执行少数账户和服务管理任务。要查看需要您以根用户身份登录的任务，请参阅[需要根用户的 Amazon 任务](#)。

IAM 用户 (p. 70)

[IAM 用户 \(p. 70\)](#)是您在 Amazon 中创建的实体。IAM 用户表示使用 IAM 用户与 Amazon 互动的人员或服务。IAM 用户的主要用途是使人们能够登录到 Amazon Web Services Management Console 以执行交互式任务，并向使用 API 或 CLI 的 Amazon 服务发出编程请求。Amazon 中的用户由用于登录 Amazon Web Services Management Console 的用户名和密码组成，最多可包含用于 API 或 CLI 的两个访问密钥。当您创建 IAM 用户时，通过使该用户成为附加了适当的权限策略的用户组的成员或者通过直接将策略附加到该用户，从而授予用户权限。您也可以克隆现有 IAM 用户的权限，这将使得新用户自动成为相同用户组的成员并附加所有相同的策略。要将 IAM 用户添加到您的 IAM 账户，请参阅[在您的 Amazon 账户中创建 IAM 用户 \(p. 72\)](#)。

IAM 用户组 (p. 151)

[IAM 用户组 \(p. 151\)](#)是 IAM 用户的集合。您可以使用用户组为一组用户指定权限，以便更轻松的管理这些用户的权限。例如，您可能有一个名为 Admins 的用户组，并向该用户组授予管理员通常需要的权限类型。该用户组中的任何用户均自动具有分配给该用户组的权限。如果有新用户加入您的企业，并且应具有管理员权限，则可通过将该用户添加到该用户组，分配相应的权限。同样，如果您的企业中有人更换工作，则不

必编辑该用户的权限，只需从旧用户组中将其删除，然后将其添加到合适的新用户组即可。在基于资源的策略中，不能将用户组识别为一个 Principal。用户组是一种一次性将策略附加到多个用户的方法。将基于身份的策略附加到用户组时，该用户组中的所有用户都将获得该用户组的权限。有关这些策略类型的更多信息，请参阅[基于身份的策略](#)和[基于资源的策略](#) (p. 379)。

IAM 角色 (p. 158)

IAM 角色 (p. 158) 非常类似于用户，因为它是一个实体，该实体具有确定其在 Amazon 中可执行和不可执行的操作的权限策略。但是，角色没有任何关联的凭证（密码或访问密钥）。角色旨在让需要它的任何人代入，而不是唯一地与某个人关联。IAM 用户可担任角色来暂时获得针对特定任务的不同权限。可将角色分配给使用外部身份提供程序而非 IAM 登录的[联合身份用户](#) (p. 170)。Amazon 使用由身份提供程序传递的详细信息来确定映射到联合身份用户的角色。

IAM 中的临时凭证 (p. 300)

临时凭证主要用于 IAM 角色，但也有其他用途。您可以请求权限集比标准 IAM 用户限制更严格的临时凭证。这可以防止您意外执行限制更严格的凭证不允许执行的任务。临时凭证的一个好处是会在设定的时间段后自动过期。您可以控制这些凭证的有效期。

何时创建 IAM 用户（而不是角色）

由于 IAM 用户只是您的账户中一个具有特定权限的身份，因此可能不需要为每个需要凭证的场合都创建 IAM 用户。在很多情况下，可利用 IAM 角色及其临时安全凭证来代替使用与 IAM 用户关联的长期凭证。

- 您创建了 Amazon 账户，您是唯一使用您的账户的人员。

可通过您的 Amazon 账户的根用户凭证使用 Amazon，但我们建议不要这样做。相反，强烈建议您为自己创建一个 IAM 用户，并通过该用户的凭证使用 Amazon。有关更多信息，请参阅[IAM 中的安全最佳实践](#) (p. 752)。

- 您所在用户组中的其他人需要在您的 Amazon 账户中工作，而您的用户组不使用其他身份机制。

为需要访问您的 Amazon 资源的个人创建 IAM 用户，向每个用户分配相应的权限，然后为每个用户提供他们的凭证。强烈建议您不要在多个用户间共享凭证。

何时创建 IAM 角色（而不是用户）

在以下情况下创建 IAM 角色：

您要创建一个在 Amazon Elastic Compute Cloud (Amazon EC2) 实例上运行的应用程序，该应用程序向 Amazon 提出请求。

请勿创建 IAM 用户并将该用户的凭证传递给该应用程序，也不要该应用程序中嵌入凭证。相反，创建附加到 EC2 实例的 IAM 角色来向实例上运行的应用程序提供临时安全凭证。当应用程序在 Amazon 中使用这些凭证时，它可以执行附加到角色的策略允许的所有操作。有关详细信息，请参阅[使用 IAM 角色向 Amazon EC2 实例上运行的应用程序授予权限](#) (p. 252)。

您正在创建一个应用程序，该应用程序在手机上运行，并向 Amazon 提出请求。

请勿创建 IAM 用户并用该应用程序分发该用户的访问密钥。相反，请使用身份提供程序（如 Login with Amazon、Amazon Cognito、Facebook 或 Google）验证用户的身份并将该用户映射到 IAM 角色。应用

程序可使用角色获取临时安全凭证，这些凭证拥有附加到该角色的策略中指定的权限。有关更多信息，请参阅下列内容：

- Amazon Mobile SDK for Android 开发人员指南中的 [Amazon Cognito 概述](#)
- Amazon Mobile SDK for iOS 开发人员指南中的 [Amazon Cognito 概述](#)
- [关于 Web 联合身份验证 \(p. 171\)](#)

您公司中的用户在企业网络中进行身份验证，需要能够使用 Amazon 而不必重新登录 - 即，您需要允许用户向 Amazon 进行联合身份验证。

不要创建 IAM 用户。在您的企业身份系统和 Amazon 之间配置联合关系。您可以通过两种方式执行此操作：

- 如果您公司的身份系统与 SAML 2.0 兼容，则可在您公司的身份系统与 Amazon 之间建立信任。有关更多信息，请参阅[关于基于 SAML 2.0 的联合身份验证 \(p. 176\)](#)。
- 创建并使用将企业中的用户身份转换为 IAM 角色以及提供临时 Amazon 安全凭证的自定义代理服务。有关更多信息，请参阅[使自定义身份代理能够访问 Amazon 控制台 \(p. 201\)](#)。

IAM 用户

Amazon Identity and Access Management (IAM) 用户 是您在 Amazon 中创建的一个实体，该实体代表使用它与 Amazon 进行交互的人员或应用程序。Amazon 中的用户包括名称和凭证。

具备管理员权限的 IAM 用户与 Amazon Web Services 账户 根用户并不是一回事。有关根用户的更多信息，请参阅 [Amazon 账户根用户 \(p. 337\)](#)。

Important

如果您发现此页面是因为您正在寻找有关 Product Advertising API 的信息以在您的网站上销售亚马逊产品，请参阅 [Product Advertising API 5.0 文档](#)。

Amazon 如何标识 IAM 用户

当您创建用户时，IAM 提供以下这些方法来识别该用户：

- 该用户的“易记名称”，这是您在创建该用户时指定的名称，如 Richard 或 Anaya。您将在 Amazon Web Services Management Console 中看到这些名称。
- 用户的 Amazon Resource Name (ARN)。当您需要跨所有 Amazon 唯一标识用户时，可以使用 ARN。例如，您可以使用 ARN 在 Amazon S3 存储桶的 IAM policy 中将用户指定为 PrincipalIAM 用户的 ARN 可能类似于以下内容：

```
arn:aws:iam::account-ID-without-hyphens:user/Richard
```

- 用户的唯一标识符。仅在您使用 API、Tools for Windows PowerShell 或 Amazon CLI 创建用户时返回此 ID；控制台中不会显示此 ID。

有关这些标识符的更多信息，请参阅[IAM 标识符 \(p. 896\)](#)。

用户和凭证

您可以通过不同方式访问 Amazon，具体取决于用户凭证：

- [控制台密码 \(p. 87\)](#)：密码，用户可键入密码以登录交互式会话，例如 Amazon Web Services Management Console。禁用用户的密码（控制台访问）可防止用户使用其用户名和密码登录 Amazon

Web Services Management Console。这不会更改他们的权限，也不会阻止他们使用担任的角色访问控制台。

- [访问密钥 \(p. 96\)](#)：访问密钥 ID 和秘密访问密钥的组合。您可以一次向一个用户分配两个访问密钥。这些可用于对 Amazon 进行编程调用。如果该用户具有有效的访问密钥，它们将继续起作用并允许通过 Amazon CLI、Tools for Windows PowerShell、Amazon API，或 Amazon Console Mobile Application 进行访问。
- [与 CodeCommit 结合使用的 SSH 密钥 \(p. 144\)](#)：可用于向 CodeCommit 进行身份验证的采用 OpenSSH 格式的 SSH 公有密钥。
- [服务器证书 \(p. 147\)](#)：您可用于向某些 Amazon 服务进行身份验证的 SSL/TLS 证书。我们建议您使用 Amazon Certificate Manager (ACM) 来预置、管理和部署您的服务器证书。只有当您必须在 ACM 不支持的区域中支持 HTTPS 连接时，才应使用 IAM。要了解哪些区域支持 ACM，请参阅 Amazon 一般参考中的 [Amazon Certificate Manager 端点和配额](#)。

您可以选择最适合您的 IAM 用户的凭证。当您使用 Amazon Web Services Management Console 来创建用户时，必须选择至少包含一个控制台密码或访问密钥。默认情况下，使用 Amazon CLI 或 Amazon API 创建的全新 IAM 用户没有任何类型的凭证。您必须根据 IAM 用户需求为该用户创建凭证类型。

可使用以下选项管理密码、访问密钥和 MFA 设备：

- [管理 IAM 用户的密码 \(p. 87\)](#)。创建和更改允许访问 Amazon Web Services Management Console 的密码。设置密码策略以强制实施最小密码复杂性。允许用户更改其密码。
- [管理 IAM 用户的访问密钥 \(p. 96\)](#)。创建和更新用于通过编程方式访问账户中的资源的访问密钥。
- 您可以通过为用户启用 [多重身份验证 \(MFA\) \(p. 104\)](#)，增强用户凭证的安全性。使用 MFA，用户必须提供两种形式的身份证明：首先，他们提供属于用户身份一部分的凭证（密码或访问密钥）。此外，他们还提供在硬件设备上或由智能手机或平板电脑上的应用程序生成的临时数字代码。
- [查找未使用的密码和访问密钥 \(p. 138\)](#)。拥有您账户或您账户中的 IAM 用户的密码或访问密钥的任何人都可以访问您的 Amazon 资源。安全 [最佳实践](#) 是，在用户不再需要密码和访问密钥时将其删除。
- [下载您账户的凭证报告 \(p. 141\)](#)。您可以生成和下载列出您账户中所有 IAM 用户及其各个凭证状态（包括密码、访问密钥和 MFA 设备）的凭证报告。对于密码和访问密钥，凭证报告将显示多久前使用了密码或访问密钥。

用户和权限

默认情况下，全新的 IAM 用户没有执行任何操作的 [权限 \(p. 353\)](#)。该用户无权执行任何 Amazon 操作或访问任何 Amazon 资源。采用单独 IAM 用户的优势在于可单独向每个用户分配权限。您可以向几个用户分配管理权限，而这些用户随后可管理您的 Amazon 资源，甚至创建和管理其他 IAM 用户。但在大多数情况下，您希望限制用户的权限，使其只能访问工作所需的任务（Amazon 操作）和资源。

设想一个名为 Diego 的用户。在创建 IAM 用户 Diego 时，您可以为该用户创建密码。您还可以向 IAM 用户附加权限，以使其能够启动特定 Amazon EC2 实例以及从 Amazon RDS 数据库中的表读取 (GET) 信息。有关如何创建用户并授予用户初始凭证和权限的过程，请参阅 [在您的 Amazon 账户中创建 IAM 用户 \(p. 72\)](#)。有关如何更改现有用户的权限的过程，请参阅 [更改 IAM 用户的权限 \(p. 82\)](#)。有关如何更改用户的密码或访问密钥的过程，请参阅 [在 Amazon 中管理用户密码 \(p. 87\)](#) 和 [管理 IAM 用户的访问密钥 \(p. 96\)](#)。

您还可以向您的用户添加权限边界。权限边界是一项高级功能，可让您使用 Amazon 托管策略来限制基于身份的策略可向用户或角色授予的最大权限。有关策略类型和用法的更多信息，请参阅 [IAM 中的策略和权限 \(p. 355\)](#)。

用户和账户

每个 IAM 用户均与一个且仅一个 Amazon 账户关联。由于用户是在您的 Amazon 账户中定义的，因此不需要向 Amazon 报备付款方式。用户在您的账户中执行的任何 Amazon 活动产生的费用均计入您的账户。

Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅[IAM 与 Amazon STS 配额、名称要求和字符限制 \(p. 902\)](#)。

作为账户服务的用户

IAM 用户是 IAM 中具有相关凭证和权限的资源。IAM 用户可以表示一个人或使用此人的凭证向 Amazon 提出请求的应用程序。这通常被称为服务账户。如果您选择在应用程序中使用 IAM 用户的长期凭证，请勿直接将访问密钥嵌入您的应用程序代码。使用 Amazon 开发工具包和 Amazon Command Line Interface，可以在已知位置放置访问密钥，这样就不必将其保留在代码中。有关更多信息，请参阅 Amazon 一般参考中的[正确管理 IAM 用户访问密钥](#)。另外，作为最佳实践，您可以[使用临时安全凭证 \(IAM 角色\)](#)代替长期访问密钥。

在您的 Amazon 账户中创建 IAM 用户

 [Follow us on Twitter](#)

可以在您的 Amazon 账户中创建一个或多个 IAM 用户。当有人加入您的团队时，或创建需要对 Amazon 进行 API 调用的新应用程序时，您可能会创建 IAM 用户。

Important

如果您发现此页面是因为您正在寻找有关 Product Advertising API 的信息以在您的网站上销售亚马逊产品，请参阅[Product Advertising API 5.0 文档](#)。

如果您是从 IAM 控制台到达该页面的，您的账户可能不包含 IAM 用户（即使您已登录）。能够使用角色以 Amazon Web Services 账户根用户身份登录，也可以使用临时凭证登录。要了解有关这些 IAM 身份的更多信息，请参阅[IAM 身份 \(用户、用户组和角色\) \(p. 68\)](#)。

主题

- [创建 IAM 用户 \(控制台\) \(p. 73\)](#)
- [创建 IAM 用户 \(Amazon CLI\) \(p. 74\)](#)
- [创建 IAM 用户 \(Amazon API\) \(p. 75\)](#)

创建用户并使该用户能够执行工作任务的过程包含以下步骤：

1. 在 Amazon Web Services Management Console、Amazon CLI、Tools for Windows PowerShell 中或使用 Amazon API 操作创建用户。如果您在 Amazon Web Services Management Console 中创建用户，则将根据您的选择自动处理步骤 1 到步骤 4。如果您以编程方式创建用户，则必须分别执行上述每个步骤。
2. 根据用户所需的访问类型为用户创建凭证：
 - 编程访问：IAM 用户可能需要进行 API 调用、使用 Amazon CLI 或使用 Tools for Windows PowerShell。对于这种情况，请为该用户创建访问密钥（访问密钥 ID 和秘密访问密钥）。
 - Amazon Web Services Management Console 访问：如果用户需要访问 Amazon Web Services Management Console，请[为用户创建密码 \(p. 90\)](#)。禁用用户的控制台访问可防止用户使用其用户名和密码登录 Amazon Web Services Management Console。这不会更改他们的权限，也不会阻止他们使用担任的角色访问控制台。
3. 通过将用户添加到一个或多个组，向用户提供执行所需任务的权限。您还可以通过将权限策略直接附加到用户来授予权限。但是，我们建议您将用户放入组内并通过附加到这些组的策略来管理权限。您还可以使用[权限边界 \(p. 370\)](#)来限制用户可以具有的权限，但这不常用。
4. （可选）通过附加标签来向用户添加元数据。有关在 IAM 中使用标签的更多信息，请参阅[标记 IAM 资源 \(p. 276\)](#)。
5. 向用户提供必要的登录信息。信息包括密码以及用户在其中提供这些凭证的账户登录页面的控制台 URL。有关更多信息，请参阅[IAM 用户如何登录 Amazon \(p. 77\)](#)。

6. (可选) 为用户配置[多重验证 \(MFA\)](#) (p. 104)。MFA 要求用户在每次登录 Amazon Web Services Management Console 时都提供一次性的代码。
7. (可选) 向用户授予管理其自己的安全凭证所需的权限。(默认状态下, 用户没有权限管理自己的凭证。) 有关更多信息, 请参阅[允许 IAM 用户更改自己的密码](#) (p. 94)。

有关创建用户时需要的权限的信息, 请参阅[访问 IAM 资源所需的权限](#) (p. 508)。

创建 IAM 用户 (控制台)

您可以使用 Amazon Web Services Management Console 创建 IAM 用户。

创建一个或多个 IAM 用户 (控制台)

1. 登录 Amazon Web Services Management Console, 然后通过以下网址打开 IAM 控制台: <https://console.aws.amazon.com/iam/>。
2. 在导航窗格中, 选择 Users (用户), 然后选择 Add users (添加用户)。
3. 为新用户键入用户名。这是 Amazon 的登录名。如果您要添加多个用户, 请为每个其他用户选择 Add another user (添加另一个用户) 并键入其用户名。您一次最多可以添加 10 个用户。

Note

Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息, 请参阅[IAM 与 Amazon STS 配额、名称要求和字符限制](#) (p. 902)。用户名可以是一个最多由 64 个字母、数字和以下字符构成的组合: 加号 (+)、等号 (=)、逗号 (,)、句点 (.)、at 符号 (@)、下划线 (_) 和连字符 (-)。账户中的名称必须唯一。名称不区分大小写。例如, 您不能创建名为 TESTUSER 和 testuser 的两个用户。

4. 选择此组用户将拥有的访问权限类型。您可以选择以编程方式访问、访问 Amazon Web Services Management Console, 或者同时选择这二者。
 - 如果用户需要访问 API、Amazon CLI 或 Tools for Windows PowerShell, 请选择 Programmatic access (编程访问)。这会为每个新用户创建访问密钥。您可以在转到 Final 页面后查看或下载访问密钥。
 - 如果用户需要访问 Amazon Web Services Management Console, 请选择 Amazon Web Services Management Console access (控制台访问)。这会为每个新用户创建密码。
 - a. 对于 Console password (控制台密码), 选择下列选项之一:
 - 自动生成的密码。每个用户将获得一个随机生成的密码, 该密码符合[账户密码策略](#) (p. 88)。在转到完成页面后, 您可以查看或下载密码。
 - 自定义密码。向每个用户分配您在框内键入的密码。
 - b. (可选) 我们建议您选择 Require password reset (需要密码重置) 以确保用户在首次登录时必须更改其密码。

Note

如果管理员启用了[Allow users to change their own password \(允许用户更改其密码\)](#) 账户密码策略设置, 则此复选框不执行任何操作。否则, 它会自动将名为 `IAMUserChangePassword` 的 Amazon 托管策略附加到新用户。该策略授予他们更改其密码的权限。

5. 选择 Next: Permissions (下一步: 权限)。
6. 在 Set permissions 页面上, 指定您要向这组新用户分配权限的方式。选择下列三个选项之一:
 - 将用户添加到组。如果您希望将用户分配到已具有权限策略的一个或多个组, 请选择此选项。IAM 将显示您账户中的组及其附加的策略的列表。您可以选择一个或多个现有组, 或者选择 Create group 来创建新组。有关更多信息, 请参阅[更改 IAM 用户的权限](#) (p. 82)。

- 从现有用户复制权限。选择此选项可将现有用户的所有组成员资格、附加的托管策略、嵌入式内联策略以及任何现有的[权限边界 \(p. 370\)](#)都复制给新用户。IAM 将显示您账户中的用户的列表。选择其权限与新用户的需求最为匹配的一个用户。
 - 直接附加现有策略。选择此选项可查看您的账户中的 Amazon 托管策略和客户托管策略的列表。选择您要附加到新用户的策略或选择创建策略，以打开新的浏览器选项卡并从头开始创建新策略。有关更多信息，请参阅过程[创建 IAM 策略 \(p. 438\)](#)中的步骤 4。在您创建策略后，关闭该选项卡并返回到您的原始选项卡，以将策略添加到新用户。作为最佳实践，我们建议您改为将策略附加到组，然后使用户成为相应的组的成员。
7. (可选) 设置[权限边界 \(p. 370\)](#)。这是一项高级功能。
- 打开设置权限边界部分，然后选择使用权限边界控制最大用户权限。IAM 将显示您的账户中的 Amazon 托管策略和客户托管策略的列表。选择要用于权限边界的策略，或选择创建策略以打开新的浏览器选项卡并从头开始创建新策略。有关更多信息，请参阅过程[创建 IAM 策略 \(p. 438\)](#)中的步骤 4。在您创建策略后，关闭该选项卡并返回到您的原始选项卡，以选择要用于权限边界的策略。
8. 选择下一步: 标签。
9. (可选) 通过以键值对的形式附加标签来向用户添加元数据。有关在 IAM 中使用标签的更多信息，请参阅[标记 IAM 资源 \(p. 276\)](#)。
10. 选择 Next: Review 以查看您此时已做出的所有选择。如果您已准备好继续，请选择 Create user (创建用户)。
11. 要查看用户的访问密钥 (访问密钥 ID 和秘密访问密钥)，请选择您要查看的每个密码和访问密钥旁边的显示。要保存访问密钥，请选择下载 .csv，然后将文件保存到安全位置。

Important

这是您查看或下载秘密访问密钥的唯一机会，您必须向用户提供此信息，他们才能使用 Amazon API。将用户的新访问密钥 ID 和秘密访问密钥保存在安全的地方。完成此步骤后，您再也无法访问这些秘密访问密钥。

12. 为每个用户提供各自的凭证。在最后的页面上，您可以选择每个用户旁边的 Send email。您的本地邮件客户端将打开并显示一份草稿，您可以对草稿进行自定义并发送。电子邮件模板包括每个用户的以下详细信息：
- 用户名称
 - 账户登录页面的 URL。使用以下示例，换入正确的账户 ID 号或账户别名：

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

有关更多信息，请参阅[IAM 用户如何登录 Amazon \(p. 77\)](#)。

Important

用户的密码未包括在生成的电子邮件中。您必须以符合您组织的安全准则的方式向客户提供密码。

创建 IAM 用户 (Amazon CLI)

您可以使用 Amazon CLI 创建 IAM 用户。

创建 IAM 用户 (Amazon CLI)

1. 创建用户。
 - `aws iam create-user`
2. (可选) 向用户提供对 Amazon Web Services Management Console 的访问权限。这需要密码。您还必须向用户提供您的[账户登录页的 URL \(p. 77\)](#)

- [aws iam create-login-profile](#)
3. (可选) 向用户提供编程访问。这需要访问密钥。
 - [aws iam create-access-key](#)
 - Tools for Windows PowerShell : [New-IAMAccessKey](#)
 - IAM API : [CreateAccessKey](#)

Important

这是您查看或下载秘密访问密钥的唯一机会，您必须向用户提供此信息，他们才能使用 Amazon API。将用户的新访问密钥 ID 和秘密访问密钥保存在安全的地方。完成此步骤后，您再也无法访问这些秘密访问密钥。

4. 将该用户添加到一个或多个组。您指定的组应具有用于向用户授予适当的权限的附加策略。
 - [aws iam add-user-to-group](#)
5. (可选) 向用户附加策略，此策略用于定义该用户的权限。注意：建议您通过将用户添加到一个组并向该组附加策略 (而不是直接向用户附加策略) 来管理用户权限。
 - [aws iam attach-user-policy](#)
6. (可选) 通过附加标签来向用户添加自定义属性。有关更多信息，请参阅[管理 IAM 用户 \(Amazon CLI 或 Amazon API \) 的标签 \(p. 280\)](#)。
7. (可选) 为用户授予权限以管理自己的安全凭证。有关更多信息，请参阅[Amazon：允许经过 MFA 身份验证的 IAM 用户在“My Security Credentials” \(我的安全凭证 \) 页面上管理自己的凭证 \(p. 395\)](#)。

创建 IAM 用户 (Amazon API)

您可以使用 Amazon API 创建 IAM 用户。

从 (Amazon API) 创建 IAM 用户

1. 创建用户。
 - [CreateUser](#)
2. (可选) 向用户提供对 Amazon Web Services Management Console 的访问权限。这需要密码。您还必须向用户提供[您的账户登录页的 URL \(p. 77\)](#)
 - [CreateLoginProfile](#)
3. (可选) 向用户提供编程访问。这需要访问密钥。
 - [CreateAccessKey](#)

Important

这是您查看或下载秘密访问密钥的唯一机会，您必须向用户提供此信息，他们才能使用 Amazon API。将用户的新访问密钥 ID 和秘密访问密钥保存在安全的地方。完成此步骤后，您再也无法访问这些秘密访问密钥。

4. 将该用户添加到一个或多个组。您指定的组应具有用于向用户授予适当的权限的附加策略。
 - [AddUserToGroup](#)
5. (可选) 向用户附加策略，此策略用于定义该用户的权限。注意：建议您通过将用户添加到一个组并向该组附加策略 (而不是直接向用户附加策略) 来管理用户权限。
 - [AttachUserPolicy](#)
6. (可选) 通过附加标签来向用户添加自定义属性。有关更多信息，请参阅[管理 IAM 用户 \(Amazon CLI 或 Amazon API \) 的标签 \(p. 280\)](#)。

7. (可选) 为用户授予权限以管理自己的安全凭证。有关更多信息, 请参阅[Amazon : 允许经过 MFA 身份验证的 IAM 用户在“My Security Credentials” \(我的安全凭证\) 页面上管理自己的凭证 \(p. 395\)](#)。

控制用户访问 Amazon Web Services Management Console

具有权限的用户通过 Amazon Web Services Management Console 登录到 Amazon 账户可以访问您的 Amazon 资源。以下列表显示了您授权用户通过 Amazon Web Services Management Console 访问您的 Amazon 账户资源的几种方式。同时, 还列出了用户通过 Amazon 网站访问其他 Amazon 账户功能的方式。

Note

使用 IAM 不会产生任何费用。

这些区域有: Amazon Web Services Management Console

您需要访问 Amazon Web Services Management Console 的每位用户创建一个密码。用户通过启用了 IAM 的 Amazon 账户登录页面访问控制台。有关访问登录页面的信息, 请参阅 [以 IAM 用户或根用户的身份登录到 Amazon Web Services Management Console \(p. 60\)](#)。有关创建密码的信息, 请参阅 [在 Amazon 中管理用户密码 \(p. 87\)](#)。

您可以通过删除其密码来禁止用户访问 Amazon Web Services Management Console。这会阻止他们使用其用户名和密码登录 Amazon Web Services Management Console。这不会更改他们的权限, 也不会阻止他们使用担任的角色访问控制台。如果该用户具有有效的访问密钥, 它们将继续起作用并允许通过 Amazon CLI、Tools for Windows PowerShell、Amazon API, 或 Amazon Console Mobile Application 进行访问。

您的 Amazon 资源 (例如 Amazon EC2 实例、Amazon S3 存储桶等)

即使您的用户持有密码, 也仍需拥有访问您的 Amazon 资源的许可。当您创建用户时, 该用户预设情况下没有任何权限。若要为授予用户所需的许可, 您需要为其附加策略。如果您有许多使用相同资源来执行相同任务的用户, 则可将这些用户分配至该组。然后将权限分配给该组。有关创建用户和组的信息, 请参阅 [IAM 身份 \(用户、用户组和角色\) \(p. 68\)](#)。有关使用策略来设置许可的信息, 请参阅 [适用于 Amazon 资源的 Access Management \(p. 353\)](#)。

Amazon 开发论坛

任何人都可阅读 [Amazon 开发论坛](#) 上的帖子。希望在 Amazon 开发论坛上发布问题或评论的用户可使用他们的用户名进行操作。当用户初次在 Amazon 论坛上发帖时, 将提示该用户输入昵称和电子邮件地址。只有该用户可以在 Amazon 论坛中使用该昵称

您的 Amazon 账户账单和使用率信息

您可授权用户访问您的 Amazon 账户账单和使用率信息。有关更多信息, 请参阅 [Amazon Billing 用户指南](#) 中的控制对账单信息的访问权限。

您的 Amazon 账户资料信息

用户无法访问您的 Amazon 账户资料信息。

您的 Amazon 账户安全凭证

用户无法访问您的 Amazon 账户安全凭证。

Note

IAM 策略控制访问不受接口限制。例如, 您可以为用户提供密码以访问 Amazon Web Services Management Console。该用户 (或该用户所属的任何组) 的策略将控制用户可在 Amazon Web Services Management Console 中执行的操作。或者, 您可以为用户提供 Amazon 访问密钥以对 Amazon 进行 API 调用。这些策略将控制用户可以通过使用这些访问密钥进行身份验证的数据库或客户端可以调用哪些操作。

IAM 用户如何登录 Amazon

要以 IAM 用户的身份登录到 Amazon Web Services Management Console，除了您的用户名和密码外，您还须提供您的账户 ID 或账户别名。当您的管理员在[控制台中创建您的 IAM 用户 \(p. 73\)](#)时，他们应该已经向您发送了登录凭证，其中包括您的用户名和账户登录页面的 URL，内含您的账户 ID 或账户别名。

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

提示

要在 Web 浏览器中为您的账户登录页面创建书签，您应在标签条目中手动键入您的账户的登录 URL。不要使用 Web 浏览器的书签功能，因为重定向会掩盖登录 URL。

您也可以在下述常规登录端点登录，然后手动键入您的账户 ID 或账户别名：

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

为方便起见，Amazon 登录页面使用浏览器 Cookie 记住 IAM 用户名和账户信息。下次用户转到 Amazon Web Services Management Console 中的任何页面时，控制台会使用 cookie 将用户重定向到账户登录页面。

您只能访问您的管理员在附加到您的 IAM 用户身份的策略中指定的 Amazon 资源。要在控制台开展工作，您必须有权执行控制台执行的操作（例如列出和创建 Amazon 资源）。有关更多信息，请参阅[适用于 Amazon 资源的 Access Management \(p. 353\)](#)和[IAM 基于身份的策略示例 \(p. 391\)](#)。

Note

如果贵企业现在有一个身份系统，您可能需要创建单点登录 (SSO) 选项。SSO 向用户提供对您账户 Amazon Web Services Management Console 的访问权限，而不要求他们具有 IAM 用户身份。SSO 也无需用户单独登录您的组织的网站和 Amazon。有关更多信息，请参阅[使自定义身份代理能够访问 Amazon 控制台 \(p. 201\)](#)。

在 CloudTrail 中记录登录详细信息

如果您允许 CloudTrail 将登录事件记录到您的日志中，您需要了解 CloudTrail 如何选择在何处记录事件。

- 如果您的用户直接登录到控制台，则系统会根据所选服务控制台是否支持区域，将他们重定向到全局或区域登录终端节点。例如，主控制台主页支持区域，因此，如果您登录以下 URL：

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

您会被重定向到例如 `https://us-east-2.signin.aws.amazon.com` 这样的区域登录终端节点，从而导致该区域日志中产生一个区域 CloudTrail 日志条目：

另一方面，Amazon S3 控制台不支持区域，因此，如果您登录到以下 URL

```
https://alias.signin.aws.amazon.com/console/s3
```

Amazon 会将您重定向到全局登录终端节点 `https://signin.aws.amazon.com`，从而产生一个全局 CloudTrail 日志条目。

- 您可以通过使用类似如下的 URL 语法登录到启用区域的主控制台主页，来手动请求特定区域网站终端节点：

```
https://alias.signin.aws.amazon.com/console?region=ap-southeast-1
```


Amazon 将您重定向到 ap-southeast-1 区域登录终端节点并使该区域中产生一个 CloudTrail 日志事件。

有关 CloudTrail 和 IAM 的更多信息，请参阅[使用 CloudTrail 记录 IAM 事件](#)。

如[管理访问密钥（控制台）](#) (p. 98) 中所述，如果用户需要编程访问来使用您的账户，则可以为所有用户创建访问密钥对（访问密钥 ID 和私有密钥）。

使用 MFA 设备访问您的 IAM 登录页面

[配置了多重身份验证 \(MFA\)](#) (p. 104) 设备的 IAM 用户必须使用其 MFA 设备登录 Amazon Web Services Management Console。在用户输入用户名和密码后，Amazon 将检查用户的账户以查看该用户是否需要 MFA。以下各节提供了有关必须使用 MFA 时用户如何完成登录的信息。

主题

- [使用虚拟 MFA 设备登录](#) (p. 78)
- [使用 FIDO 安全密钥登录](#) (p. 78)
- [使用硬件 MFA 设备登录](#) (p. 78)

使用虚拟 MFA 设备登录

如果 MFA 是用户必须使用的，则会显示另一个登录页面。在 MFA code (MFA 代码) 框中，用户必须输入 MFA 应用程序提供的数字代码。

如果 MFA 代码正确，则用户可以访问 Amazon Web Services Management Console。如果代码不正确，则用户可以使用其他代码重试。

虚拟 MFA 设备可能会不同步。如果用户尝试多次后都无法登录 Amazon Web Services Management Console，系统将提示用户同步虚拟 MFA 设备。用户可以按照屏幕上的提示同步虚拟 MFA 设备。有关如何在您的 Amazon 账户中同步代表用户的设备的信息，请参阅[重新同步虚拟和硬件 MFA 设备](#) (p. 122)。

使用 FIDO 安全密钥登录

如果 MFA 是用户必须使用的，则会显示另一个登录页面。用户需要点击 FIDO 安全密钥。

Note

Google Chrome 用户不应选择弹出窗口中的任何要求 Verify your identity with amazon.com (通过 amazon.com 验证您的身份) 的可用选项。您只需要点击安全密钥即可。

与其他 MFA 设备不同，FIDO 安全密钥不同步。如果 FIDO 安全密钥丢失或损坏，管理员可以停用它。有关更多信息，请参阅[停用 MFA 设备（控制台）](#) (p. 126)。

有关支持 Amazon 所支持的 WebAuthn 和 FIDO 合规设备的浏览器的信息，请参阅[使用 FIDO 安全密钥的支持配置](#) (p. 113)。

使用硬件 MFA 设备登录

如果 MFA 是用户必须使用的，则会显示另一个登录页面。在 MFA code (MFA 代码) 框中，用户必须输入硬件 MFA 设备提供的数字代码。

如果 MFA 代码正确，则用户可以访问 Amazon Web Services Management Console。如果代码不正确，则用户可以使用其他代码重试。

硬件 MFA 设备可能会不同步。如果用户尝试多次后都无法登录 Amazon Web Services Management Console，系统将提示用户同步 MFA 令牌设备。用户可以根据屏幕上的提示同步 MFA 令牌设备。有关如何在您的 Amazon 账户中同步代表用户的设备的信息，请参阅[重新同步虚拟和硬件 MFA 设备](#) (p. 122)。

管理 IAM 用户

Amazon Web Services 提供了多种工具来管理 Amazon 账户中的 IAM 用户。您可以列出您的账户中或用户组中的 IAM 用户，也可以列出用户所属的所有用户组。您可以重命名或更改 IAM 用户的路径。您可以从您的 Amazon 账户中删除 IAM 用户，或暂时停用该用户。

有关添加、更改或删除 IAM 用户的托管策略的更多信息，请参阅 [更改 IAM 用户的权限 \(p. 82\)](#)。有关为 IAM 用户管理内联策略的信息，请参阅 [添加和删除 IAM 身份权限 \(p. 451\)](#)、[编辑 IAM 策略 \(p. 461\)](#) 和 [删除 IAM 策略 \(p. 465\)](#)。作为最佳实践，请使用托管策略而不是内联策略。要了解有关验证 IAM policy 的更多信息，请参阅 [验证 IAM 策略 \(p. 443\)](#)

有关管理 IAM 用户密码的信息，请参阅 [管理 IAM 用户的密码 \(p. 90\)](#)。

主题

- [查看用户访问 \(p. 79\)](#)
- [列出 IAM 用户 \(p. 79\)](#)
- [重命名 IAM 用户 \(p. 80\)](#)
- [删除 IAM 用户 \(p. 80\)](#)
- [停用 IAM 用户 \(p. 82\)](#)

查看用户访问

在删除用户之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅[使用上次访问的信息优化 Amazon 中的权限 \(p. 467\)](#)。

列出 IAM 用户

您可以将 IAM 用户列在您的 Amazon 账户或特定 IAM 用户组中，并列出了用户所属的所有用户组。有关为列出用户而需要的权限的信息，请参阅[访问 IAM 资源所需的权限 \(p. 508\)](#)。

列出账户中的所有用户

- [Amazon Web Services Management Console](#)：在导航窗格中，选择 Users (用户)。控制台显示您 Amazon 账户中的用户。
- Amazon CLI：[aws iam list-users](#)
- Amazon API：[ListUsers](#)

列出特定用户组中的用户

- [Amazon Web Services Management Console](#)：在导航窗格中，请选择 Groups (组)，选择用户组的名称，然后选择 Users (用户) 选项卡。
- Amazon CLI：[aws iam get-group](#)
- Amazon API：[GetGroup](#)

列出用户所属的所有用户组

- [Amazon Web Services Management Console](#)：在导航窗格中，选择 Users，选择用户名称，然后选择 Groups 选项卡。
- Amazon CLI：[aws iam list-groups-for-user](#)

- Amazon API : [ListGroupForUser](#)

重命名 IAM 用户

要更改用户的名称或路径，必须使用 Amazon CLI、Tools for Windows PowerShell 或 Amazon API。控制台没有用于重命名用户的选项。有关为将用户重命名而需要的权限的信息，请参阅[访问 IAM 资源所需的权限](#) (p. 508)。

当您更改用户名或路径时，发生以下情况：

- 应用于用户的所有策略采用新用户名继续生效。
- 采用新用户名用户保留在原来的用户组。
- 用户的唯一 ID 保持不变。有关唯一 ID 的更多信息，请参阅[唯一标识符](#) (p. 900)。
- 任何将该用户视为主体（向该用户授予访问权限）的资源或角色策略均自动更新以使用新用户名或路径。例如，Amazon SQS 中任何基于队列的策略或 Amazon S3 中任何基于资源的策略都会自动更新，以使用新名称和路径。

IAM 不自动更新将该用户视为资源的策略以使用新用户名或路径；必须由您手动更新。例如，假设向用户 Richard 附加了一个策略，该策略使用该用户可管理其自己的安全凭证。如果管理员将 Richard 重命名为 Rich，则管理员还需要更新该策略以将资源从此：

```
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/Richard
```

改为：

```
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/Rich
```

如果管理员更改路径，则也会发生这种情况；管理员需要更新策略以反映该用户使用新路径。

重命名用户

- Amazon CLI : [aws iam update-user](#)
- Amazon API : [UpdateUser](#)

删除 IAM 用户

如果 IAM 用户退出贵公司，则可从您的 Amazon Web Services 账户中删除该用户。如果该用户暂时退出，则可停用其访问权限，而不是如[停用 IAM 用户](#) (p. 82) 中所述将其从账户中删除。

主题

- [删除 IAM 用户（控制台）](#) (p. 80)
- [删除 IAM 用户 \(Amazon CLI\)](#) (p. 81)

删除 IAM 用户（控制台）

使用 Amazon Web Services Management Console 删除 IAM 用户时，IAM 将自动删除以下信息：

- 用户
- 任何用户组成员资格 — 也就是说，该用户将从所属的任何 IAM 用户组中删除
- 任何与该用户关联的密码
- 属于该用户的任何访问密钥

- 嵌入到该用户中的所有内联策略（通过用户组权限应用于用户的策略不受影响）

Note

当您删除用户时，IAM 会移除附加到用户的任何托管策略，但不会删除托管策略。

- 任何关联的 MFA 设备

删除 IAM 用户（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Users（用户），然后选中要删除的用户旁的复选框。
3. 在页面的顶部，选择删除用户。
4. 在确认对话框中，在文本输入字段中输入用户名以确认删除用户。选择 Delete（删除）。

删除 IAM 用户 (Amazon CLI)

与 Amazon Web Services Management Console 不同，在使用 Amazon CLI 删除用户时，您必须手动删除附加到该用户的项目。此过程演示了这个流程。

从您的账户删除用户 (Amazon CLI)

1. 请删除用户的密码（如果用户具有）。

```
aws iam delete-login-profile
```

2. 请删除用户的访问密钥（如果用户具有）。

```
aws iam list-access-keys
```

（用于列出用户的访问密钥）和

```
aws iam delete-access-key
```

3. 请删除用户的签名证书。注意，当您删除安全凭证时，凭证永远消失，无法恢复。

```
aws iam list-signing-certificates
```

（用于列出用户的签名证书）和

```
aws iam delete-signing-certificate
```

4. 请删除用户的 SSH 公有密钥（如果用户具有）。

```
aws iam list-ssh-public-keys
```

（用于列出用户的 SSH 公有密钥）和

```
aws iam delete-ssh-public-key
```

5. 请删除用户的 Git 凭证。

```
aws iam list-service-specific-credentials
```

（用于列出用户的 Git 凭证）和

```
aws iam delete-service-specific-credential
```

6. 停用用户的多重身份验证 (MFA) 设备（如果用户具有）。

```
aws iam list-mfa-devices
```

（用于列出用户的 MFA 设备）、

```
aws iam deactivate-mfa-device
```

（用于停用设备）和

```
aws iam delete-virtual-mfa-device
```

（用于永久删除虚拟 MFA 设备）

7. 请删除用户的内联策略。

```
aws iam list-user-policies
```

（用于列出用户的内联策略）和

```
aws iam delete-user-policy
```

（用于删除策略）

8. 分离附加到用户的任何托管策略。

```
aws iam list-attached-user-policies
```

（用于列出附加到用户的托管策略）和

```
aws iam detach-user-policy
```

（用于分离策略）

9. 请从任何用户组中删除用户。

```
aws iam list-groups-for-user (用于列出用户所属的用户组) 和 aws iam remove-user-from-group
```

10. 请删除用户。

```
aws iam delete-user
```

停用 IAM 用户

如果 IAM 用户暂时退出贵公司，您可能需要将其停用。您可以保留其 IAM 用户凭证，但仍然阻止其进行 Amazon 访问。

要停用用户，请创建并附加策略以拒绝该用户访问 Amazon。您可以稍后恢复该用户的访问权限。

以下是您可以附加到用户以拒绝其访问的拒绝策略的两个示例。

以下策略不包含时间限制。您必须删除该策略才能恢复用户的访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

以下策略包含一个条件，即策略自 2024 年 12 月 24 日晚上 11:59 (UTC) 起生效，并于 2025 年 2 月 28 日晚上 11:59 (UTC) 终止。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "DateGreaterThan": {"aws:CurrentTime": "2024-12-24T23:59:59Z"},
        "DateLessThan": {"aws:CurrentTime": "2025-02-28T23:59:59Z"}
      }
    }
  ]
}
```

更改 IAM 用户的权限

您可以更改您的 Amazon 账户中的 IAM 用户的权限，方法是更改用户的组成员资格、复制现有用户的权限、直接为用户附加策略或设置[权限边界](#) (p. 370)。权限边界控制用户可以具有的最大权限。权限边界是一项高级 Amazon 功能。

有关您修改用户的权限所需的权限的信息，请参阅[访问 IAM 资源所需的权限](#) (p. 508)。

主题

- [查看用户访问](#) (p. 83)

- [基于用户的访问活动生成策略 \(p. 83\)](#)
- [向用户添加权限 \(控制台\) \(p. 83\)](#)
- [更改用户的权限 \(控制台\) \(p. 85\)](#)
- [从用户删除权限策略 \(控制台\) \(p. 86\)](#)
- [从用户删除权限边界 \(控制台\) \(p. 86\)](#)
- [添加和删除用户的权限 \(Amazon CLI 或 Amazon API\) \(p. 86\)](#)

查看用户访问

在更改用户的权限之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅[使用上次访问的信息优化 Amazon 中的权限 \(p. 467\)](#)。

基于用户的访问活动生成策略

有时，您可能会向 IAM 实体（用户或角色）授予超出其需要的权限。为帮助您优化授予的权限，您可以根据实体的访问活动生成 IAM 策略。IAM 访问分析器会查看您的 Amazon CloudTrail 日志并生成一个策略模板，其中包含实体在指定日期范围内使用的权限。您可以使用模板创建具有精细权限的托管策略，然后将其附加到 IAM 实体。这样，您仅需授予用户或角色与特定使用案例中的 Amazon 资源进行交互所需的权限。要了解更多信息，请参阅[“基于访问活动生成策略 \(p. 443\)”](#)。

向用户添加权限 (控制台)

IAM 提供了三种方法来向用户添加权限策略：

- 将用户添加到组 - 使用户成为一个组的成员。来自该组的策略将附加到用户。
- 复制现有用户的权限 - 复制所有组成员资格、附加的托管策略、内联策略以及源用户的任何现有权限边界。
- 直接为用户附加策略 - 直接为用户附加托管策略。作为最佳实践，我们建议您改为将策略附加到组，然后使用户成为相应的组的成员。

Important

如果用户具有权限边界，则您为用户添加的权限不能超过权限边界所允许的权限。

通过将用户添加到组来添加权限

将用户添加到组会立即影响用户。

通过将用户添加到组来向用户添加权限

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users（用户）。
3. 在控制台的 Groups 列中查看用户的当前组成员资格。如有必要，请通过完成以下步骤来将该列添加到用户表中：
 1. 在最右侧的表上方，选择设置符号 (⚙️)。
 2. 在 Manage Columns 对话框中，选择 Groups 列。(可选) 您还可以清除不希望在用户表中显示的任何列标题的复选框。
 3. 选择 Close 返回到用户列表。

Groups 列指示用户所属的组。该列最多包含两个组的组名称。如果用户是三个或更多个组的成员，则只显示前两个组（按字母顺序排序），并且包含其他组成员资格的数量。例如，如果用户属于组 A、组 B、组 C 和组 D，则该字段包含值 Group A, Group B + 2 more。要查看用户所属组的总数，可以向用户表添加 Group count 列。

4. 请选择要修改其权限的用户的名称。
5. 请选择 Permissions 选项卡，然后选择 Add permissions。选择 Add user to group。
6. 选择您希望用户加入的每个组所对应的复选框。列表显示了每个组的名称以及用户在成为该组成员后获得的策略。
7. （可选）除了从现有组中进行选择之外，您还可以选择 Create group 来定义新组。
 - a. 在新选项卡中，对于组名，键入您的新组的名称。

Note

Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅[IAM 与 Amazon STS 配额、名称要求和字符限制 \(p. 902\)](#)。组名称可以是最多由 128 个字母、数字和以下字符构成的组合：加号 (+)、等号 (=)、逗号 (,)、句点 (.)、at 符号 (@) 和连字符 (-)。账户中的名称必须唯一。名称不区分大小写。例如，您不能创建名为 TESTGROUP 和 testgroup 的两个组。

- b. 选中您要附加到组的托管策略所对应的一个或多个复选框。您还可以通过选择 Create policy 来创建新的托管策略。如果您这样做，请在创建新的策略之后返回到此浏览器选项卡或窗口；选择 Refresh，然后选择新策略以将其附加到您的组。有关更多信息，请参阅[创建 IAM policy \(p. 437\)](#)。
 - c. 选择创建组。
 - d. 返回到原始选项卡，刷新您的组列表。然后选中您的新组所对应的复选框。
8. 选择 Next: Review 以查看要添加到用户的组成员资格的列表。然后选择 Add permissions。

通过从其他用户复制来添加权限

复制权限会立即影响用户。

通过从其他用户复制权限来向用户添加权限

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中选择 Users，选择您要修改其权限的用户的名称，然后选择 Permissions 选项卡。
3. 选择添加权限，然后选择从现有用户复制权限。列表显示了可用用户以及其组成员资格和附加的策略。如果组或策略的完整列表不能显示在一行中，您可以选择 and **n** more 链接。执行此操作将打开新的浏览器选项卡，可以查看策略 (Permissions 选项卡) 和组 (Groups 选项卡) 的完整列表。
4. 选择您想要复制其权限的用户旁边的单选按钮。
5. 选择 Next: Review 以查看要对用户所做的更改的列表。然后选择 Add permissions。

通过直接将策略附加到用户来添加权限

附加策略会立即影响用户。

通过直接附加托管策略来向用户添加权限

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中选择 Users，选择您要修改其权限的用户的名称，然后选择 Permissions 选项卡。

3. 选择添加权限，然后选择直接将现有策略附加到用户。
4. 选中您要附加到用户的托管策略所对应的一个或多个复选框。您还可以通过选择 Create policy 来创建新的托管策略。如果您这样做，请在创建新的策略之后返回到此浏览器选项卡或窗口。选择 Refresh (刷新)，然后选中新策略对应的复选框以将该策略附加到您的用户。有关更多信息，请参阅[创建 IAM policy \(p. 437\)](#)。
5. 选择 Next: Review 以查看要附加到用户的策略的列表。然后选择 Add permissions。

为用户设置权限边界

设置权限边界会立即影响用户。

为用户设置权限边界

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 请选择要更改其权限边界的用户的名称。
4. 选择 Permissions 选项卡。如有必要，打开权限边界部分，然后选择设置边界。
5. 选择要用于权限边界的策略。
6. 选择设置边界。

更改用户的权限 (控制台)

IAM 允许您通过以下方式更改与用户关联的权限：

- 编辑权限策略 - 编辑用户的内联策略、用户的组的内联策略，或编辑直接附加到用户或从组附加到用户的托管策略。如果用户具有权限边界，则您提供的权限不能超过用作用户的权限边界的策略所允许的权限。
- 更改权限边界 - 更改用作用户的权限边界的策略。这可以扩大或限制用户可以具有的最大权限。

编辑附加到用户的权限策略

更改权限会立即影响用户。

编辑用户的已附加托管策略

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 请选择要更改其权限策略的用户的名称。
4. 请选择 Permissions 选项卡。如有必要，打开权限策略部分。
5. 请选择要编辑的策略的名称以查看有关该策略的详细信息。请选择用作选项卡以查看您编辑策略可能会影响到的其他实体。
6. 请选择权限选项卡并查看策略授予的权限。然后选择编辑策略。
7. 使用 Visual editor (可视化编辑器) 选项卡或 JSON 选项卡并解决任何[策略验证 \(p. 443\)](#)建议。有关更多信息，请参阅[编辑 IAM 策略 \(p. 461\)](#)。
8. 选择查看策略，查看策略摘要，然后选择保存更改。

更改用户的权限边界

更改权限边界会立即影响用户。

更改用于设置用户的权限边界的策略

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users（用户）。
3. 请选择要更改其权限边界的用户的名称。
4. 选择 Permissions 选项卡。如有必要，打开权限边界部分，然后选择更改边界。
5. 选择要用于权限边界的策略。
6. 选择更改边界。

从用户删除权限策略（控制台）

删除策略会立即影响用户。

撤销 IAM 用户的权限

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users（用户）。
3. 请选择要删除其权限边界的用户的名称。
4. 请选择 Permissions 选项卡。
5. 如果要通过移除现有策略来撤销权限，请查看 Policy type（策略类型）来了解用户如何获取该策略，然后再选择 X 以移除策略：
 - 如果该策略因组成员资格而适用，则选择 X 可从组中删除该用户。请记住，可向单个组中附加多个策略。如果您从组中删除某个用户，该用户将失去对其通过组成员资格接收的所有策略的访问权限。
 - 如果策略是直接附加到该用户的托管策略，则选择 X 可分离该用户和该策略。这不会影响该策略本身或该策略可能附加到的任何其他实体。
 - 如果该策略是内联嵌入式策略，则选择 X 可从 IAM 中删除该策略。直接附加到用户的内联策略只能在该用户中存在。

从用户删除权限边界（控制台）

删除权限边界会立即影响用户。

从用户删除权限边界

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users（用户）。
3. 请选择要删除其权限边界的用户的名称。
4. 请选择 Permissions 选项卡。如有必要，打开权限边界部分，然后选择删除边界。
5. 选择删除以确认您要删除权限边界。

添加和删除用户的权限（Amazon CLI 或 Amazon API）

要以编程方式添加或删除权限，您必须添加组成员资格、附加或分离托管策略或者添加或删除内联策略。有关更多信息，请参阅以下主题：

- [在 IAM 用户组中添加和删除用户 \(p. 154\)](#)

- [添加和删除 IAM 身份权限 \(p. 451\)](#)

在 Amazon 中管理用户密码

可以为 Amazon Web Services 账户根用户和您的账户中的 IAM 用户管理密码。IAM 用户需要密码才能访问 Amazon Web Services Management Console。利用 Amazon CLI、Tools for Windows PowerShell、Amazon SDK 或 API，用户无需密码即可以编程方式访问 Amazon 资源。对于这些环境，用户需要改用[访问密钥 \(p. 96\)](#)。

目录

- [更改 Amazon 账户根用户密码 \(p. 87\)](#)
- [为 IAM 用户设置账户密码策略 \(p. 88\)](#)
- [管理 IAM 用户的密码 \(p. 90\)](#)
- [允许 IAM 用户更改自己的密码 \(p. 94\)](#)
- [IAM 用户如何更改自己的密码 \(p. 95\)](#)

更改 Amazon 账户根用户密码

您必须以根用户身份而不是 IAM 用户登录 Amazon Web Services 账户 才能更改根用户密码。要了解如何重置已经忘记的根用户密码，请参阅 [重置您丢失或遗忘的 Amazon 密码或访问密钥 \(p. 103\)](#)。

为根用户更改密码

1. 使用您的 Amazon 账户电子邮件地址和密码以 Amazon Web Services 账户根用户身份登录 [Amazon Web Services Management Console](#)。

Note

如果显示了三个文本框，则您之前已使用 [IAM 用户](#) 凭证登录控制台。您的浏览器可能会记住此首选项，并在您每次尝试登录时打开此账户特定的登录页面。您无法使用 IAM 用户登录页面以账户所有者身份进行登录。如果您看到 [IAM 用户登录页面](#)，请选择该页面底部附近的 Sign in using root user email (使用根用户电子邮件登录)。这将返回主登录页面。在该页面中，您可以使用 Amazon 账户电子邮件地址和密码以根用户登录。

2. 在控制台的右上角，选择您的账户名称或账号，然后选择 My Security Credentials (我的安全凭证)。
3. 展开 Password (密码) 部分，然后选择 Click here (点击此处) 文本以更改密码。
4. 选择一个强密码。虽然您可以为 [IAM 用户设置账户密码策略 \(p. 88\)](#)，但该策略不适用于您的 Amazon Web Services 账户根用户。

Amazon 要求您的密码满足以下条件：

- 长度最少为 8 个字符，最多为 128 个字符
- 包括至少以下三种字符类型的组合：大写、小写、数字，以及 ! @ # \$ % ^ & * () < > [] { } | _ + = 符号
- 不同于您的 Amazon 账户名称或电子邮件地址

Note

Amazon 将会推出登录过程的改进功能。其中一项改进是为您的账户实施更加安全的密码策略。如果您的账户已升级，则需要满足以上密码策略。如果您的账户尚未升级，Amazon 不会实施该策略，但强烈建议您遵守指南，创建更加安全的密码。

为保护您的密码，请务必遵循以下最佳实践：

- 定期更改密码并且不要泄露您的密码，因为任何知道您密码的人都可以访问您的账户。

- 不要为 Amazon 使用您在其他网站上使用的密码。
- 不要使用容易猜到的密码。此类密码包括 `secret`、`password`、`amazon`、`123456` 等。它们还包括字典中的单词、您的姓名、电子邮件地址或可以轻松获取的其他个人信息。

为 IAM 用户设置账户密码策略

您可以在 Amazon 账户上设置自定义密码策略，以便指定您的 IAM 用户密码的复杂性要求和强制轮换期。如果未设置自定义密码策略，则 IAM 用户密码必须符合默认 Amazon 密码策略。有关更多信息，请参阅[自定义密码策略选项](#) (p. 89)。

主题

- [设置密码策略的规则](#) (p. 88)
- [设置密码策略时所需的权限](#) (p. 88)
- [默认密码策略](#) (p. 89)
- [自定义密码策略选项](#) (p. 89)
- [设置密码策略 \(控制台\)](#) (p. 89)
- [设置密码策略 \(Amazon CLI\)](#) (p. 90)
- [设置密码策略 \(Amazon API\)](#) (p. 90)

设置密码策略的规则

IAM 密码策略不适用于 Amazon Web Services 账户根用户密码或 IAM 用户访问密钥。如果密码过期，IAM 用户将无法登录 Amazon Web Services Management Console，但可以继续使用其访问密钥。

创建或更改密码策略时，大多数密码策略设置会在用户下次更改其密码时实施。但是，一些设置将立即实施。例如：

- 在最小长度和字符类型要求变更后，系统会在您的用户下次更改其密码时强制实施该设置。不强制用户更改其现有密码，即使这些密码不符合更新后的密码策略。
- 设置密码有效期时，有效期立即生效。例如，假定您将密码有效期设置为 90 天。在这种情况下，对于现有密码使用期限超过 90 天的所有 IAM 用户，其密码将过期。这些用户必须在下次登录时更改其密码。

在尝试指定的登录失败次数后，您将无法创建“锁定策略”来锁定账户用户。为了增强安全性，我们建议您将强密码策略与 Multi-Factor Authentication (MFA) 相结合。有关 MFA 的更多信息，请参阅[在 Amazon 中使用多重身份验证 \(MFA\)](#) (p. 104)。

设置密码策略时所需的权限

您必须配置权限以允许 IAM 实体（用户或角色）查看或编辑其账户密码策略。您可以在 IAM policy 中包含以下密码策略操作：

- `iam:GetAccountPasswordPolicy` - 允许实体查看其账户的密码策略
- `iam:DeleteAccountPasswordPolicy` - 允许实体删除其账户的自定义密码策略并恢复到默认密码策略
- `iam:UpdateAccountPasswordPolicy` - 允许实体为其账户创建或更改自定义密码策略

以下策略允许查看和编辑账户密码策略的完全访问权限。要了解如何使用该示例 JSON 策略文档创建 IAM policy，请参阅[the section called “在“JSON”选项卡上创建策略”](#) (p. 438)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FullAccessPasswordPolicy",
```

```

        "Effect": "Allow",
        "Action": [
            "iam:GetAccountPasswordPolicy",
            "iam:DeleteAccountPasswordPolicy",
            "iam:UpdateAccountPasswordPolicy"
        ],
        "Resource": "*"
    }
}

```

有关 IAM 用户更改自己的密码所需权限的信息，请参阅 [允许 IAM 用户更改自己的密码 \(p. 94\)](#)。

默认密码策略

如果管理员未设置自定义密码策略，则 IAM 用户密码必须符合默认 Amazon 密码策略。默认密码策略强制执行以下条件：

- 密码长度最短为 8 个字符，最长为 128 个字符
- 至少包含以下三种字符类型的组合：大写、小写、数字，以及 ! @ # \$ % ^ & * () _ + - = [] { } | ' 符号
- 与您的 Amazon 账户名称或电子邮件地址不同

自定义密码策略选项

为账户配置自定义密码策略时，可以指定以下条件：

- 密码最小长度 - 您可以指定最少 6 个字符和最多 128 个字符。
- Password strength (密码强度) - 您可以选中以下任一复选框来定义 IAM 用户密码的强度：
 - 至少需要一个大写拉丁字母 (A-Z)
 - 至少需要一个小写拉丁字母 (a-z)
 - 至少需要一个数字
 - 至少需要一个非字母数字字符 ! @ # \$ % ^ & * () _ + - = [] { } | '
- Enable password expiration (启用密码过期) - 您可以选择并指定 IAM 用户密码设置后的有效期：至少 1 天和最多 1095 天。例如，如果您指定的过期时间为 90 天，则会立即影响所有用户。对于密码超过 90 天的用户，更改后登录控制台时，必须设置新密码。密码为 75-89 天的用户会收到有关密码过期的 Amazon Web Services Management Console 警告。IAM 用户可以随时更改其密码（如有权限）。密码的有效期从用户设置新密码时起计算。IAM 用户同时只能有一个有效密码。
- Password expiration requires administrator reset (密码过期需要管理员重置) - 选择此选项可防止 IAM 用户在密码过期后使用 Amazon Web Services Management Console 更新自己的密码。选择此选项之前，请确认您的 Amazon 账户具有多个具备管理权限的用户，以重置 IAM 用户密码。具有 iam:UpdateLoginProfile 权限的管理员可重置 IAM 用户密码。具有 iam:ChangePassword 权限和活动访问密钥的 IAM 用户可以编程方式重置其 IAM 用户控制台密码。如果清除此复选框，密码已过期的 IAM 用户仍必须先设置新密码，然后才能访问 Amazon Web Services Management Console。
- Allow users to change their own password (允许用户更改自己的密码) - 您可以允许账户中的所有 IAM 用户更改自己的密码。这样用户就可以仅访问其用户的 iam:ChangePassword 操作和 iam:GetAccountPasswordPolicy 操作。此选项不会将权限策略附加到每个用户，而是通过 IAM 将权限应用于所有用户的账户级别。或者，您可以只允许部分用户管理自己的密码。为此需要清除此复选框。有关使用策略来限制哪些人可以管理密码的更多信息，请参阅 [允许 IAM 用户更改自己的密码 \(p. 94\)](#)。
- Prevent password reuse (防止密码重复使用) - 您可以阻止 IAM 用户重复使用指定数量的前密码。您可以指定最少 1 个和最多 24 个不能重复的前密码。

设置密码策略 (控制台)

您可使用 Amazon Web Services Management Console 创建、更改或删除自定义密码策略。

创建自定义密码策略 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择账户设置。
3. 在 Password policy (密码策略) 部分，选择 Change password policy (更改密码策略)。
4. 选择您要应用于密码策略的选项，然后选择 Save changes (保存更改)。

更改自定义密码策略 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择账户设置。
3. 在 Password policy (密码策略) 部分，选择 Change (更改)。
4. 选择您要应用于密码策略的选项，然后选择 Save changes (保存更改)。

删除密码策略 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择账户设置。
3. 在 Password policy (密码策略) 部分，选择 Delete (删除)。
4. 通过选择 Delete custom (删除自定义) 来确认您要删除自定义密码策略。

设置密码策略 (Amazon CLI)

您可以使用 Amazon Command Line Interface 设置密码策略。

通过 Amazon CLI 管理自定义账户密码策略

运行以下命令：

- 创建或更改自定义密码策略：`aws iam update-account-password-policy`
- 查看密码策略：`aws iam get-account-password-policy`
- 删除自定义密码策略：`aws iam delete-account-password-policy`

设置密码策略 (Amazon API)

您可以使用 Amazon API 操作来设置密码策略。

通过 Amazon API 管理自定义账户密码策略

调用以下操作：

- 创建或更改自定义密码策略：`UpdateAccountPasswordPolicy`
- 查看密码策略：`GetAccountPasswordPolicy`
- 删除自定义密码策略：`DeleteAccountPasswordPolicy`

管理 IAM 用户的密码

通过 Amazon Web Services Management Console 使用 Amazon 资源的 IAM 用户必须具有密码才能登录。您可以创建、更改或删除您的 Amazon 账户中 IAM 用户的密码。

向用户分配密码后，该用户可使用您的账户的登录 URL 登录 Amazon Web Services Management Console，如下所示：

```
https://12-digit-Amazon-account-ID or alias.signin.aws.amazon.com/console
```

有关 IAM 用户如何登录 Amazon Web Services Management Console 的更多信息，请参阅 [以 IAM 用户或根用户的身份登录到 Amazon Web Services Management Console \(p. 60\)](#)。

即使用户有自己的密码，还是需要权限才能访问您的 Amazon 资源。默认情况下，用户没有权限。为授予用户所需的许可，您可向用户或用户所属的群组分配策略。有关创建用户和组的信息，请参阅 [IAM 身份 \(用户、用户组和角色\) \(p. 68\)](#)。有关使用策略来设置许可的信息，请参阅 [更改 IAM 用户的权限 \(p. 82\)](#)。

可向用户授予相应的权限，用于更改其自身的密码。有关更多信息，请参阅 [允许 IAM 用户更改自己的密码 \(p. 94\)](#)。有关用户如何访问账户登录页面的信息，请参阅 [以 IAM 用户或根用户的身份登录到 Amazon Web Services Management Console \(p. 60\)](#)。

主题

- [创建、更改或删除 IAM 用户密码 \(控制台\) \(p. 91\)](#)
- [创建、更改或删除 IAM 用户密码 \(Amazon CLI\) \(p. 93\)](#)
- [创建、更改或删除 IAM 用户密码 \(Amazon API\) \(p. 93\)](#)

创建、更改或删除 IAM 用户密码 (控制台)

您可以使用 Amazon Web Services Management Console 管理 IAM 用户的密码。

当用户离开您的组织或不再需要访问 Amazon 时，请务必找到用户所使用的凭证并确保这些凭证不再被使用。理想情况下，如果不再需要凭证，可将其删除。您始终可在将来需要这些凭证时创建它们。您至少应更改这些凭证，以便之前的用户不再拥有访问权。

为 IAM 用户添加密码 (控制台)

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 请选择要为其创建密码的用户的名称。
4. 选择 Security Credentials 选项卡，然后在 Sign-in Credentials 下选择 Console password 旁边的 Manage password。
5. 在 Manage console access 中，对于 Console access，选中 Enable (如果尚未选中)。如果禁用控制台访问，则无需密码。
6. 对于 Set password (设置密码)，选择是让 IAM 生成密码还是创建自定义密码：
 - 要让 IAM 生成密码，请选择 Autogenerated password (自动生成的密码)。
 - 要创建自定义密码，请选择 Custom password，然后键入密码。

Note

您创建的密码必须符合账户的 [密码策略 \(p. 88\)](#)。

7. 若要求用户在登录时创建新密码，请选择 Require password reset。然后选择 Apply。

Important

如果选择 Require password reset 选项，请确保用户具有密码更改权限。有关更多信息，请参阅 [允许 IAM 用户更改自己的密码 \(p. 94\)](#)。

8. 如果选择此选项来生成密码，请在 New password (新密码) 对话框中选择 Show (显示)。这使您能够查看密码，以便与用户进行共享。

Important

出于安全原因，在完成该步骤后您无法访问该密码，但您可以随时创建新密码。

更改 IAM 用户的密码 (控制台)

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 请选择要更改其密码的用户的名称。
4. 选择 Security Credentials 选项卡，然后在 Sign-in Credentials 下选择 Console password 旁边的 Manage password。
5. 在 Manage console access 中，对于 Console access，选中 Enable (如果尚未选中)。如果禁用控制台访问，则无需密码。
6. 对于 Set password (设置密码)，选择是让 IAM 生成密码还是创建自定义密码：
 - 要让 IAM 生成密码，请选择 Autogenerated password (自动生成的密码)。
 - 要创建自定义密码，请选择 Custom password，然后键入密码。

Note

如果当前设置了账户的[密码策略 \(p. 88\)](#)，则您创建的密码必须符合该策略。

7. 若要求用户在登录时创建新密码，请选择 Require password reset。然后选择 Apply。

Important

如果选择 Require password reset 选项，请确保用户具有密码更改权限。有关更多信息，请参阅[允许 IAM 用户更改自己的密码 \(p. 94\)](#)。

8. 如果选择此选项来生成密码，请在 New password (新密码) 对话框中选择 Show (显示)。这使您能够查看密码，以便与用户进行共享。

Important

出于安全原因，在完成该步骤后您无法访问该密码，但您可以随时创建新密码。

删除 (禁用) IAM 用户的密码 (控制台)

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 请选择要删除其密码的用户的名称。
4. 选择 Security Credentials 选项卡，然后在 Sign-in Credentials 下选择 Console password 旁边的 Manage password。
5. 对于 Console access，选择 Disable，然后选择 Apply。

Important

您可以通过删除其密码来禁止用户访问 Amazon Web Services Management Console。这可以阻止他们使用其用户名和密码登录到 Amazon Web Services Management Console。这不会更改他们的权限，也不会阻止他们使用担任的角色访问控制台。如果该用户具有有效的访问密钥，它们将继续起作用并允许通过 Amazon CLI、Tools for Windows PowerShell、Amazon API，或 Amazon Console Mobile Application 进行访问。

创建、更改或删除 IAM 用户密码 (Amazon CLI)

您可以使用 Amazon CLI API 管理 IAM 用户的密码。

创建密码 (Amazon CLI)

1. (可选) 要确定用户是否有密码, 请运行此命令: [aws iam get-login-profile](#)
2. 要创建密码, 请运行此命令: [aws iam create-login-profile](#)

更改用户的密码 (Amazon CLI)

1. (可选) 要确定用户是否有密码, 请运行此命令: [aws iam get-login-profile](#)
2. 要更改密码, 请运行此命令: [aws iam update-login-profile](#)

删除 (禁用) 用户的密码 (Amazon CLI)

1. (可选) 要确定用户是否有密码, 请运行此命令: [aws iam get-login-profile](#)
2. (可选) 要确定上次使用密码的时间, 请运行此命令: [aws iam get-user](#)
3. 要删除密码, 请运行此命令: [aws iam delete-login-profile](#)

Important

在删除用户的密码后, 用户无法再登录到 Amazon Web Services Management Console。如果该用户具有有效的访问密钥, 它们将继续起作用并允许通过 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 函数调用进行访问。在使用 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 从您的 Amazon 账户中删除用户时, 您必须先使用此操作删除密码。有关更多信息, 请参阅[删除 IAM 用户 \(Amazon CLI\)](#) (p. 81)。

创建、更改或删除 IAM 用户密码 (Amazon API)

您可以使用 Amazon API 管理 IAM 用户的密码。

创建密码 (Amazon API)

1. (可选) 要确定用户是否有密码, 请调用此操作: [GetLoginProfile](#)
2. 要创建密码, 请调用此操作: [CreateLoginProfile](#)

更改用户的密码 (Amazon API)

1. (可选) 要确定用户是否有密码, 请调用此操作: [GetLoginProfile](#)
2. 要更改密码, 请调用此操作: [UpdateLoginProfile](#)

删除 (禁用) 用户的密码 (Amazon API)

1. (可选) 要确定用户是否有密码, 请运行此命令: [GetLoginProfile](#)
2. (可选) 要确定上次使用密码的时间, 请运行此命令: [GetUser](#)
3. 要删除密码, 请运行此命令: [DeleteLoginProfile](#)

Important

在删除用户的密码后, 用户无法再登录到 Amazon Web Services Management Console。如果该用户具有有效的访问密钥, 它们将继续起作用并允许通过 Amazon CLI、Tools for Windows

PowerShell 或 Amazon API 函数调用进行访问。在使用 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 从您的 Amazon 账户中删除用户时，您必须先使用此操作删除密码。有关更多信息，请参阅[删除 IAM 用户 \(Amazon CLI\) \(p. 81\)](#)。

允许 IAM 用户更改自己的密码

您可以向 IAM 用户授予更改其用于登录 Amazon Web Services Management Console 的密码的权限。您可以通过两种方式之一来执行此操作：

- [允许账户中的所有 IAM 用户更改自己的密码 \(p. 94\)](#)。
- [仅允许选定的 IAM 用户更改自己的密码 \(p. 94\)](#)。在此情况下，您禁用可供所有用户用来更改其密码的选项，并使用 IAM policy 仅向某些用户授予权限。这种方法允许这些用户更改自己的密码和可选的其他凭证，例如他们自己的访问密钥。

Important

我们建议您[设置自定义密码策略 \(p. 88\)](#)，要求 IAM 用户创建强密码。

允许所有 IAM 用户更改自己的密码

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，单击 Account settings (账户设置)。
3. 如果您的账户使用默认密码策略，请在 Password policy (密码策略) 部分选择 Change password policy (更改密码策略)。如果您的账户使用自定义密码策略，请选择 Change (更改)。
4. 选择 Allow users to change their own password (允许用户更改其密码)，然后选择 Save changes (保存更改)。这样账户中的所有用户就可以仅访问其用户的 iam:ChangePassword 操作和 iam:GetAccountPasswordPolicy 操作。
5. 向用户提供有关更改密码的以下说明：[IAM 用户如何更改自己的密码 \(p. 95\)](#)。

有关可用来更改账户的密码策略 (包括允许所有用户更改自己的密码) 的 Amazon CLI、Tools for Windows PowerShell 和 API 命令的信息，请参阅[设置密码策略 \(Amazon CLI\) \(p. 90\)](#)。

允许选定的 IAM 用户更改自己的密码

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，单击 Account settings (账户设置)。
3. 在 Password policy (密码策略) 部分中，确保未选择 Allow users to change their own password (允许用户更改其密码)。如果已选中此复选框，所有用户均可更改其密码。(请参阅上一个过程。)
4. 创建允许更改其密码的用户 (如果之前没有这样的用户)。有关详细信息，请参阅[在您的 Amazon 账户中创建 IAM 用户 \(p. 72\)](#)。
5. 为可以更改自己密码的用户创建 IAM 组，然后将上一步骤中的用户添加到该组。有关详细信息，请参阅[创建您的第一个 IAM 管理员用户和用户组 \(p. 19\)](#) 和 [管理 IAM 用户组 \(p. 153\)](#)。

此为可选步骤，但是最佳做法是使用组管理权限。这样，您就可以添加和删除用户以及更改整个组的权限。

6. 将下列策略分配到该组。有关更多信息，请参阅[管理 IAM 策略 \(p. 437\)](#)。

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

```
    "Action": "iam:GetAccountPasswordPolicy",
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": "iam:ChangePassword",
    "Resource": "arn:aws:iam::account-id:user/${aws:username}"
  }
]
```

此策略授予对 [ChangePassword](#) 操作的访问权限，这让用户可从控制台、Amazon CLI、Tools for Windows PowerShell 或 API 仅更改他们自己的密码。它还授予对 [GetAccountPasswordPolicy](#) 操作的访问权限，这可让用户查看当前的密码策略。用户需要此权限才能在 Change password (更改密码) 页面上查看账户密码策略。用户必须能阅读当前密码策略以确保更改后的密码符合策略的要求。

7. 向用户提供有关更改密码的以下说明：[IAM 用户如何更改自己的密码 \(p. 95\)](#)。

有关更多信息

有关管理凭证的更多信息，请参阅以下主题：

- [允许 IAM 用户更改自己的密码 \(p. 94\)](#)
- [在 Amazon 中管理用户密码 \(p. 87\)](#)
- [为 IAM 用户设置账户密码策略 \(p. 88\)](#)
- [管理 IAM 策略 \(p. 437\)](#)
- [IAM 用户如何更改自己的密码 \(p. 95\)](#)

IAM 用户如何更改自己的密码

如果您已获得更改自己 IAM 用户密码的权限，则可以使用 Amazon Web Services Management Console 中的特定页面完成此操作。您还可以使用 Amazon CLI 或 Amazon API。

主题

- [所需权限 \(p. 95\)](#)
- [IAM 用户如何更改自己的密码 \(控制台\) \(p. 95\)](#)
- [IAM 用户如何更改自己的密码 \(Amazon CLI 或 Amazon API\) \(p. 96\)](#)

所需权限

要为您自己的 IAM 用户更改密码，您必须具有以下策略中的权限：[Amazon：允许 IAM 用户在的“My Security Credentials” \(我的安全凭证\) 页面上更改自己的控制台密码 \(p. 402\)](#)。

IAM 用户如何更改自己的密码 (控制台)

以下过程介绍了 IAM 用户如何使用 Amazon Web Services Management Console 更改自己的密码。

更改您自己的 IAM 用户密码 (控制台)

1. 使用 Amazon 账户 ID 或账户别名、您的 IAM 用户名和密码登录到 [IAM 控制台](#)。

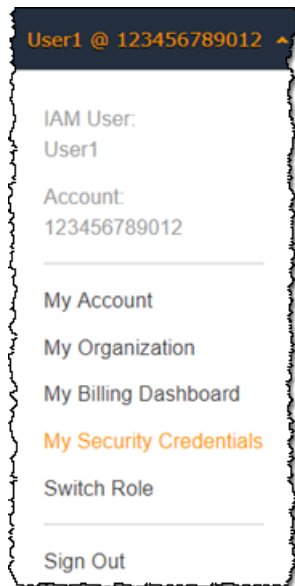
Note

为方便起见，Amazon 登录页面使用浏览器 Cookie 记住您的 IAM 用户名和账户信息。如果您之前以其他用户身份登录，请选择页面底部的 Sign-in to a different account (登录到其他账户)

以返回主登录页面。在此处，您可以输入要重新导向到您账户 IAM 用户登录页面的 Amazon 账户 ID 或账户别名。

要获取 Amazon 账户 ID，请联系管理员。

2. 在右上角的导航栏中，选择您的用户名，然后选择 My Security Credentials (我的安全凭证)。



3. 在 Amazon IAM Credentials (亚马逊云科技 IAM 凭证) 选项卡上，选择 Change password (更改密码)。
4. 对于 Current password (当前密码)，请键入您的当前密码。在 New password (新密码) 和 Confirm new password (确认新密码) 框中键入新密码。然后选择 Change Password (更改密码)。

Note

新密码必须符合账户密码策略的要求。有关更多信息，请参阅[IAM 用户设置账户密码策略 \(p. 88\)](#)。

IAM 用户如何更改自己的密码 (Amazon CLI 或 Amazon API)

以下过程介绍了 IAM 用户如何使用 Amazon CLI 或 Amazon API 更改自己的密码。

要更改自己的 IAM 密码，请使用以下命令：

- Amazon CLI: `aws iam change-password`
- Amazon API : `ChangePassword`

管理 IAM 用户的访问密钥

 Follow us on Twitter

Note

如果您发现此页面是因为您正在寻找有关 Product Advertising API 的信息以在您的网站上销售亚马逊产品，请参阅 [Product Advertising API 5.0 文档](#)。

访问密钥是 IAM 用户或 Amazon Web Services 账户 根用户的长期凭证。您可以使用访问密钥签署对 Amazon CLI 或 Amazon API 的编程请求（直接或使用 Amazon 开发工具包）。有关更多信息，请参阅 Amazon Web Services 一般参考中的[签名 Amazon API 请求](#)。

访问密钥包含两部分：访问密钥 ID（例如 AKIAIOSFODNN7EXAMPLE）和秘密访问密钥（例如 wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY）。与用户名和密码一样，您必须同时使用访问密钥 ID 和秘密访问密钥对请求执行身份验证。像对用户名和密码一样，安全地管理访问密钥。

Important

请不要向第三方提供访问密钥，即便是为了帮助[找到您的规范用户 ID](#)也不行。如果您这样做，可能会向某人提供对您的账户的永久访问权限。

作为最佳实践，请使用临时安全凭证（IAM 角色）代替访问密钥，并禁用 Amazon Web Services 账户根用户访问密钥。有关更多信息，请参阅 Amazon Web Services 一般参考中的[管理 Amazon 访问密钥的最佳实践](#)。

如果您仍需要使用长期访问密钥，则可创建、修改、查看或轮换您的访问密钥（访问密钥 ID 和秘密访问密钥）。您最多可拥有两个访问密钥。这使您能够根据最佳实践轮换有效的密钥。

当您创建访问密钥对时，将访问密钥 ID 和秘密访问密钥保存在一个安全位置。秘密访问密钥仅在您创建它时可用。如果您丢失了秘密访问密钥，则必须删除访问密钥并创建新的访问密钥。有关更多信息，请参阅[重置您丢失或遗忘的 Amazon 密码或访问密钥 \(p. 103\)](#)。

主题

- [所需权限 \(p. 97\)](#)
- [管理访问密钥（控制台）\(p. 98\)](#)
- [管理访问密钥 \(Amazon CLI\) \(p. 100\)](#)
- [管理访问密钥 \(Amazon API\) \(p. 100\)](#)
- [轮换访问密钥 \(p. 100\)](#)
- [审计访问密钥 \(p. 103\)](#)

所需权限

要为您自己的 IAM 用户创建访问密钥，您必须具有以下策略中的权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateOwnAccessKeys",
      "Effect": "Allow",
      "Action": [
        "iam:CreateAccessKey",
        "iam:GetUser",
        "iam:ListAccessKeys"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    }
  ]
}
```

要为您自己的 IAM 用户轮换访问密钥，您必须具有以下策略中的权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ManageOwnAccessKeys",
      "Effect": "Allow",
      "Action": [
        "iam:CreateAccessKey",
        "iam>DeleteAccessKey",

```

```
        "iam:GetAccessKeyLastUsed",  
        "iam:GetUser",  
        "iam:ListAccessKeys",  
        "iam:UpdateAccessKey"  
    ],  
    "Resource": "arn:aws:iam::*:user/${aws:username}"  
}  
]  
}
```

管理访问密钥（控制台）

您可以使用 Amazon Web Services Management Console 管理 IAM 用户的访问密钥。

创建、修改或删除您自己的 IAM 用户访问密钥（控制台）

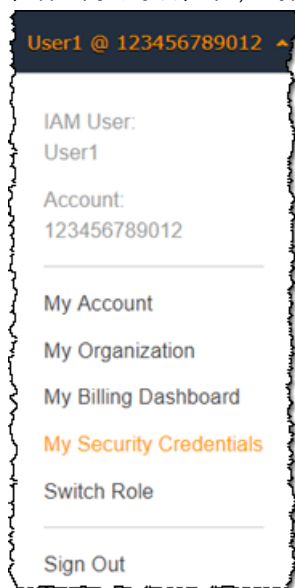
1. 使用 Amazon 账户 ID 或账户别名、您的 IAM 用户名和密码登录到 [IAM 控制台](#)。

Note

为方便起见，Amazon 登录页面使用浏览器 Cookie 记住您的 IAM 用户名和账户信息。如果您之前以其他用户身份登录，请选择页面底部的 Sign-in to a different account（登录到其他账户）以返回主登录页面。在此处，您可以输入要重新导向到您账户 IAM 用户登录页面的 Amazon 账户 ID 或账户别名。

要获取 Amazon 账户 ID，请联系管理员。

2. 在右上角的导航栏中，选择您的用户名，然后选择 My Security Credentials (我的安全凭证)。



3. 展开 Access keys (access key ID and secret access key) 部分。
4. 执行以下任一操作：
 - 要创建访问密钥，请选择 Create New Access Key (创建新的访问密钥)。如果此功能已禁用，则必须先删除现有密钥中的一个，然后才能创建新密钥。一个警告说明，您只有这一次机会可以查看或下载秘密访问密钥。要复制密钥以将其粘贴到其他位置进行保管，请选择 Show Access Key (显示访问密钥)。要将访问密钥 ID 和秘密访问密钥以 .csv 文件形式保存计算机上的安全位置，请选择 Download Key File (下载密钥文件)。
 - 要禁用活动的访问密钥，请选择 Make Inactive (设为非活动)。
 - 要重新启用非活动访问密钥，请选择 Make Active。

- 要删除访问密钥，请选择 Delete (删除)。Amazon 建议在执行此操作之前，先停用该密钥并测试它是否已不再使用。使用 Amazon Web Services Management Console 时，必须先停用密钥，然后才能删除它。

创建、修改或删除其他 IAM 用户的访问密钥 (控制台)

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 选择要管理其访问密钥的用户的名称，然后选择 Security credentials (安全凭证) 选项卡。
4. 在 Access keys (访问密钥) 部分中，执行以下任意操作：
 - 要创建访问密钥，请选择创建访问密钥。然后，选择 Download .csv file (下载 .csv 文件)，将访问密钥 ID 和秘密访问密钥保存至计算机上的 CSV 文件中。将文件存储在安全位置。关闭此对话框后，您将无法再次访问该秘密访问密钥。下载 CSV 文件之后，选择 Close (关闭)。在创建访问密钥时，预设情况下，密钥对处于活动状态，并且您可以立即使用此密钥对。
 - 要禁用活动访问密钥，请选择转为非活跃。
 - 要重新启用非活动访问密钥，请选择转为活跃。
 - 要删除访问密钥，请选择 Delete (删除)。Amazon 建议在执行此操作之前，先停用该密钥并测试它是否已不再使用。使用 Amazon Web Services Management Console 时，必须先停用密钥，然后才能删除它。

列出 IAM 用户的访问密钥 (控制台)

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 选择预期用户的名称，然后选择安全凭证选项卡。将显示用户的访问密钥和每个密钥的状态。

Note

只有用户的访问密钥 ID 是可见的。秘密访问密钥只能在创建密钥时检索到。

列出多个 IAM 用户的访问密钥 ID (控制台)

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 如有必要，可通过完成以下步骤来将 Access key ID 列添加到用户表中：
 - a. 在最右侧的表上方，选择设置图标 (⚙️)。
 - b. 在 Manage columns (管理列) 中，选择访问密钥 ID。
 - c. 选择 Close 返回到用户列表。
4. 访问密钥 ID 列显示各个访问密钥 ID，后跟其状态；例如，23478207027842073230762374023 (活动) 或 22093740239670237024843420327 (不活动)。

您可以利用该信息查看和复制具有一个或两个访问密钥的用户的访问密钥。对于没有访问密钥的用户，该列显示 None。

Note

只有用户的访问密钥 ID 和状态是可见的。秘密访问密钥只能在创建密钥时检索到。

查找哪个 IAM 用户拥有特定的访问密钥 (控制台)

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 在搜索框中，键入或粘贴要查找的用户的访问密钥 ID。
4. 如有必要，可通过完成以下步骤来将 Access key ID 列添加到用户表中：
 - a. 在最右侧的表上方，选择设置图标 ()。
 - b. 在 Manage columns (管理列) 中，选择访问密钥 ID。
 - c. 选择 Close 返回用户列表，并确认筛选出的用户拥有指定的访问密钥。

管理访问密钥 (Amazon CLI)

要从 Amazon CLI 管理 IAM 用户的访问密钥，请运行以下命令。

- 创建访问密钥: `aws iam create-access-key`
- 禁用或重新启用访问密钥: `aws iam update-access-key`
- 列出用户访问密钥的步骤: `aws iam list-access-keys`
- 确定最近使用访问密钥的时间: `aws iam get-access-key-last-used`
- 删除访问密钥: `aws iam delete-access-key`

管理访问密钥 (Amazon API)

要从 Amazon API 管理 IAM 用户的访问密钥，请调用以下操作。

- 创建访问密钥: `CreateAccessKey`
- 禁用或重新启用访问密钥: `UpdateAccessKey`
- 列出用户访问密钥的步骤: `ListAccessKeys`
- 确定最近使用访问密钥的时间: `GetAccessKeyLastUsed`
- 删除访问密钥: `DeleteAccessKey`

轮换访问密钥

建议您定期轮换 (更改) IAM 用户访问密钥，这是最佳安全做法。如果您的管理员向您授予了必要的权限，则您可轮换自己的访问密钥。

有关如何向用户授予权限以轮换他们自己的访问密钥的详细信息，请参阅[Amazon：允许 IAM 用户在“My Security Credentials” \(我的安全凭证 \) 页面上管理自己的密码、访问密钥和 SSH 公有密钥 \(p. 402\)](#)。您还可以将密码策略应用于您的账户，以要求所有 IAM 用户定期轮换其密码。您可以选择这些用户必须执行此操作的频率。有关更多信息，请参阅[IAM 用户设置账户密码策略 \(p. 88\)](#)。

Important

作为最佳实践，请勿使用 Amazon Web Services 账户 根用户。如果您使用 Amazon Web Services 账户 根用户凭证，建议您还要定期轮换这些凭证。此账户密码策略不适用于根用户凭证。IAM 用户无法管理 Amazon Web Services 账户 根用户的凭证，因此您必须使用根用户凭证 (而不是用户凭证) 来更改根用户凭证。请注意，我们建议进行 Amazon 日常工作时不要使用根用户。

主题

- [轮换 IAM 用户访问密钥 \(控制台\)](#) (p. 101)
- [轮换访问密钥 \(Amazon CLI\)](#) (p. 102)
- [轮换访问密钥 \(Amazon API\)](#) (p. 102)

轮换 IAM 用户访问密钥 (控制台)

您可以从 Amazon Web Services Management Console 轮换访问密钥。

在不中断应用程序的情况下轮换 IAM 用户的访问密钥 (控制台)

1. 当第一个访问密钥仍处于活动状态时，创建第二个访问密钥。
 - a. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
 - b. 在导航窗格中，选择 Users (用户)。
 - c. 选择预期用户的名称，然后选择安全凭证选项卡。
 - d. 选择创建访问密钥，然后选择 Download .csv file (下载 .csv 文件)，将访问密钥 ID 和秘密访问密钥保存至计算机上的 .csv 文件。将文件存储在安全位置。完成此步骤后，您将无法再访问这些秘密访问密钥。下载 .csv 文件之后选择 Close。

默认情况下，新的访问密钥处于活动状态。此时，用户拥有两个访问密钥。
2. 更新所有应用程序和工具以使用新的访问密钥。
3. 通过查看最早的访问密钥的 Last used 列来确定第一个访问密钥是否仍在使用。一种方法是等待几天，然后检查旧访问密钥是否被使用，然后再继续。
4. 即使 Last used 列值指示旧密钥从未使用过，我们还是建议您不要立即删除第一个访问密钥。您可以选择 Make inactive 来停用第一个访问密钥。
5. 仅使用新的访问密钥，以确认您的应用程序可以正常工作。此时，任何仍在使用初始访问密钥的应用程序和工具将停止工作，因为它们不再具有对 Amazon 资源的访问权限。如果您发现此类应用程序或工具，可以选择 Make active 来重新启用第一个访问密钥。然后返回到 [Step 3 \(p. 101\)](#) 并更新此应用程序以使用新的密钥。
6. 在等待一段时间以确保所有应用程序和工具均已更新后，可以删除第一个访问密钥：
 - a. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
 - b. 在导航窗格中，选择 Users (用户)。
 - c. 选择预期用户的名称，然后选择安全凭证选项卡。
 - d. 找到要删除的访问密钥，然后选择行最右侧的 X 按钮。输入访问密钥 ID 以确认删除，然后选择 Delete (删除)。

确定访问密钥何时需要轮换 (控制台)

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 如有必要，可通过完成以下步骤来将 Access key age 列添加到用户表中：
 - a. 在最右侧的表上方，选择设置图标 (⚙️)。
 - b. 在 Manage columns (管理列) 中，选择 Access key age (访问密钥使用期限)。
 - c. 选择 Close 返回到用户列表。
4. Access key age 列显示最早的活动访问密钥自创建至今经过的天数。您可以利用该信息查找需要轮换访问密钥的用户。对于没有访问密钥的用户，该列显示 None。

轮换访问密钥 (Amazon CLI)

您可以从 Amazon Command Line Interface 轮换访问密钥。

在不中断应用程序的情况下轮换访问密钥 (Amazon CLI)

1. 当第一个访问密钥仍处于活动状态时，创建第二个访问密钥，后者在默认情况下将处于活动状态。运行以下命令：

- `aws iam create-access-key`

此时，用户拥有两个访问密钥。

2. 更新所有应用程序和工具以使用新的访问密钥。
3. 通过使用以下命令，确定第一个访问密钥是否仍在使用：

- `aws iam get-access-key-last-used`

一种方法是等待几天，然后检查旧访问密钥是否被使用，然后再继续。

4. 即使步骤 [Step 3](#) 指示旧密钥未被使用，我们也建议您不要立即删除第一个访问密钥。而是通过以下命令，将第一个访问密钥的状态更改为 `Inactive`：

- `aws iam update-access-key`

5. 仅使用新的访问密钥，以确认您的应用程序可以正常工作。此时，任何仍在使用初始访问密钥的应用程序和工具将停止工作，因为它们不再具有对 Amazon 资源的访问权限。如果您发现此类应用程序或工具，可将其状态切换回 `Active` 以重新启用第一个访问密钥。然后返回到步骤 [Step 2](#) 并更新此应用程序以使用新的密钥。
6. 在等待一段时间以确保所有应用程序和工具均已更新后，您可以通过以下命令删除第一个访问密钥：

- `aws iam delete-access-key`

有关更多信息，请参阅下列内容：

- [如何轮换 IAM 用户的访问密钥](#)。Amazon 安全博客上的这篇文章介绍了更多有关密钥轮换的信息。
- [IAM 中的安全最佳实践 \(p. 752\)](#)。本页介绍有助于保护您的 Amazon 资源的一般建议。

轮换访问密钥 (Amazon API)

您可以使用 Amazon API 轮换访问密钥。

在不中断应用程序的情况下轮换访问密钥 (Amazon API)

1. 当第一个访问密钥仍处于活动状态时，创建第二个访问密钥，后者在默认情况下将处于活动状态。调用以下操作：

- `CreateAccessKey`

此时，用户拥有两个访问密钥。

2. 更新所有应用程序和工具以使用新的访问密钥。
3. 通过调用以下操作，确定第一个访问密钥是否仍在使用：

- `GetAccessKeyLastUsed`

一种方法是等待几天，然后检查旧访问密钥是否被使用，然后再继续。

4. 即使步骤 [Step 3](#) 指示旧密钥未被使用，我们也建议您不要立即删除第一个访问密钥。而是调用以下操作，将第一个访问密钥的状态更改为 `Inactive`：
 - [UpdateAccessKey](#)
5. 仅使用新的访问密钥，以确认您的应用程序可以正常工作。此时，任何仍在使用初始访问密钥的应用程序和工具将停止工作，因为它们不再具有对 Amazon 资源的访问权限。如果您发现此类应用程序或工具，可将其状态切换回 `Active` 以重新启用第一个访问密钥。然后返回到步骤 [Step 2](#) 并更新此应用程序以使用新的密钥。
6. 在等待一段时间以确保所有应用程序和工具均已更新后，您可以通过调用以下操作，删除第一个访问密钥：
 - [DeleteAccessKey](#)

有关更多信息，请参阅以下内容：

- [如何轮换 IAM 用户的访问密钥](#)。Amazon 安全博客上的这篇文章介绍了更多有关密钥轮换的信息。
- [IAM 中的安全最佳实践 \(p. 752\)](#)。本页介绍有助于保护您的 Amazon 资源的一般建议。

审计访问密钥

您可以在代码中查看 Amazon 访问密钥，以确定密钥是否来自于您拥有的账户。您可以使用 [aws sts get-access-key-info](#) Amazon CLI 命令或 [GetAccessKeyInfo](#) Amazon API 操作传递访问密钥 ID。

Amazon CLI 和 Amazon API 操作返回访问密钥所属的 Amazon 账户的 ID。以 `AKIA` 开头的访问密钥 ID 是 IAM 用户或 Amazon Web Services 账户 根用户的长期凭证。以 `ASIA` 开头的访问密钥 ID 是使用 Amazon STS 操作创建的临时凭证。如果响应中的账户属于您，则您可以根用户的身份登录并查看您的根用户访问密钥。然后，您可以提取[凭证报告 \(p. 141\)](#)以了解哪个 IAM 用户拥有这些密钥。要了解谁请求了 `ASIA` 访问密钥的临时凭证，请查看 CloudTrail 日志中的 Amazon STS 事件。

为了安全起见，您可以[查看 Amazon CloudTrail 日志 \(p. 343\)](#)以了解已在 Amazon 中执行操作的人员。您可以在角色信任策略中使用 `sts:SourceIdentity` 条件键，以要求用户在代入角色时指定身份。例如，您可以要求 IAM 用户指定自己的用户名作为其源身份。这可以帮助您确定哪个用户在 Amazon 中执行了具体的操作。有关更多信息，请参阅[sts:SourceIdentity \(p. 1024\)](#)。

该操作不指示访问密钥的状态。密钥可能处于活动状态、非活动状态或已删除状态。非活动密钥可能没有执行操作的权限。提供删除的访问密钥可能会返回“密钥不存在”错误。

重置您丢失或遗忘的 Amazon 密码或访问密钥

登录 Amazon 时遇到问题？请确保使用的是适合您用户类型的 [Amazon 登录页面 \(p. 60\)](#)。如果您是 Amazon Web Services 账户 根用户（账户所有者），则可以使用您在创建 Amazon 账户时设置的凭证登录 Amazon。如果您是 IAM 用户，则您的账户管理员可以向您提供用于登录 Amazon 的凭证。如果您需要请求支持，请不要使用此页面上的反馈链接，因为该表由 Amazon 文档团队（而非 Amazon Web Services Support）接收。相反，在 [Contact Us](#)（联系我们）页面上选择 `Still unable to log into your Amazon account`（仍然无法登录 Amazon 账户），然后选择一个可用的支持选项。

在主登录页面上，您必须输入电子邮件地址来以根用户的身份登录，或输入您的账户 ID 来以 IAM 用户身份登录。您只能在与您用户类型相符的登录页面上提供密码。有关更多信息，请参阅[IAM 用户或根用户的身份登录到 Amazon Web Services Management Console \(p. 60\)](#)。

如果您在正确的登录页面上但丢失或忘记了密码或访问密钥，您无法从 IAM 检索它们。不过，您可以使用以下方法重置它们：

- Amazon Web Services 账户 根用户密码 - 如果您忘记根用户密码，可以从 Amazon Web Services Management Console 重置密码。有关详细信息，请参阅本主题下文中的[the section called “重置丢失或遗忘的根用户密码” \(p. 104\)](#)。

- Amazon 账户访问密钥 - 如果您忘记账户访问密钥，可以创建新的访问密钥，无需禁用现有访问密钥。如果您未使用现有密钥，可以删除这些密钥。有关详细信息，请参阅 [创建根用户的访问密钥 \(p. 338\)](#) 和 [删除根用户的访问密钥 \(p. 339\)](#)。
- IAM 用户密码 - 如果您是 IAM 用户，并且忘记了密码，您必须要求管理员重置您的密码。要了解管理员如何管理您的密码，请参阅 [管理 IAM 用户的密码 \(p. 90\)](#)。
- IAM 用户访问密钥 - 如果您是 IAM 用户，并且忘记了访问密钥，您将需要新的访问密钥。如果您有权创建自己的访问密钥，则可以在 [管理访问密钥 \(控制台\) \(p. 98\)](#) 处找到有关创建新的访问密钥的说明。如果您没有所需的权限，则必须要求管理员创建新访问密钥。如果您仍使用旧密钥，请要求您的管理员不删除旧密钥。要了解管理员如何管理您的访问密钥，请参阅 [管理 IAM 用户的访问密钥 \(p. 96\)](#)。

您应遵循 Amazon [最佳实践 \(p. 753\)](#)，定期更改您的密码和 Amazon 访问密钥。在 Amazon 中，您通过轮换访问密钥来对其进行更改。这意味着，您将创建一个新的密钥，将您的应用程序配置为使用新密钥，然后删除旧密钥。出于此原因，您可以同时拥有两个有效的访问密钥对。有关更多信息，请参阅 [轮换访问密钥 \(p. 100\)](#)。

重置丢失或遗忘的根用户密码

当您首次创建 Amazon 账户时，您提供了电子邮件地址和密码。它们是您的 Amazon Web Services 账户根用户凭证。如果您忘记根用户密码，可以从 Amazon Web Services Management Console 重置密码。

重置您的根用户密码：

1. 使用您的 Amazon 账户电子邮件地址以开始以根用户的身份登录至 [Amazon Web Services Management Console](#)，然后选择 Next (下一步)。

Note

如果您已使用 IAM 用户凭证登录到 [Amazon Web Services Management Console](#)，则必须注销，然后才能重置根用户密码。如果您看到特定于账户的 IAM 用户登录页面，请选择页面底部附近的 Sign in using root account credentials (使用根账户凭证登录)。如有必要，请提供您的账户电子邮件地址并选择 Next (下一步) 来访问 Root user sign in (根用户登录) 页面。

2. 选择 Forgot your password?。
3. 提供与该账户关联的电子邮件地址。然后，提供 CAPTCHA 文本并选择 Continue。
4. 检查与您的 Amazon 账户相关联的电子邮件邮箱中是否有来自 Amazon Web Services 的邮件。该电子邮件来自以 @amazon.com 或 @aws.amazon.com 结尾的地址。按照电子邮件中的指导进行操作。如果您在账户中未看到电子邮件，请检查垃圾邮件文件夹。如果您不再拥有该电子邮件的访问权限，请参阅 [我无权访问我的 Amazon 账户的电子邮件 \(p. 67\)](#)。

在 Amazon 中使用多重身份验证 (MFA)

 Follow us on Twitter

为增强安全性，我们建议您配置 Multi-Factor Authentication (MFA) 以帮助保护 Amazon 资源。您可以为 IAM 用户或 Amazon Web Services 账户根用户启用 MFA。当您为根用户启用 MFA 时，这只会影响根用户凭证。账户中的 IAM 用户是具有自己的凭证的不同身份，并且每个身份具有自己的 MFA 配置。

Note

您是否考虑过使用 Amazon IAM Identity Center (successor to Amazon Single Sign-On) (IAM Identity Center)？您可以使用 IAM Identity Center 集中管理对多个 Amazon Web Services 账户的访问权限，并为用户提供受 MFA 保护的单点登录访问权限，可从一个位置访问其分配的所有账户。借助 IAM Identity Center，您可以在 IAM Identity Center 中创建和管理用户身份，或者轻松连接到现有的 SAML 2.0 兼容身份提供者。有关更多信息，请参阅《Amazon IAM Identity Center (successor to Amazon Single Sign-On) 用户指南》中的 [什么是 IAM Identity Center？](#)

主题

- [什么是 MFA ? \(p. 105\)](#)
- [为 Amazon 中的用户启用 MFA 设备 \(p. 105\)](#)
- [检查 MFA 的状态 \(p. 120\)](#)
- [重新同步虚拟和硬件 MFA 设备 \(p. 122\)](#)
- [停用 MFA 设备 \(p. 126\)](#)
- [若 MFA 设备遗失或停止工作，该怎么办？ \(p. 127\)](#)
- [配置受 MFA 保护的 API 访问 \(p. 129\)](#)
- [示例代码：使用多重验证请求凭证 \(p. 134\)](#)

什么是 MFA ?

MFA 增强了安全性，因为它要求用户在访问 Amazon 网站或服务时，除了其常规登录凭证之外，还要提供来自 Amazon 支持的 MFA 机制的唯一身份验证。

- **虚拟 MFA 设备。**一种在手机或其他设备上运行并模拟物理设备的软件应用程序。该设备将基于进行了时间同步的一次性密码算法生成一个六位数字代码。在登录时，用户必须在另一个网页上键入来自该设备的有效代码。分配给用户的每个虚拟 MFA 设备必须是唯一的。用户无法从另一个用户的虚拟 MFA 设备代码键入代码来进行身份验证。由于虚拟 MFA 可能在不安全的移动设备上运行，因此，它们可能无法提供与 FIDO2 设备或硬件 MFA 设备相同的安全级别。我们建议您在等待硬件购买批准或等待硬件到达时使用虚拟 MFA 设备。有关可用作虚拟 MFA 设备的一些受支持应用程序的列表，请参阅 [Multi-Factor Authentication](#)。有关使用 Amazon 设置虚拟 MFA 设备的说明，请参阅[启用虚拟 Multi-Factor Authentication \(MFA\) 设备 \(控制台\)](#) (p. 106)。
- **FIDO 安全密钥。**一台设备，您将其插入计算机上的 USB 端口。FIDO2 是由 [FIDO 联盟](#) 制定的开放身份验证标准。启用 FIDO2 安全密钥时，您将通过输入凭证然后点击设备进行登录，而不是手动输入代码。有关受支持 Amazon FIDO2 安全密钥的信息，请参阅[多重身份验证](#)。有关使用 Amazon 设置 FIDO2 安全密钥的说明，请参阅[启用 FIDO 安全密钥 \(控制台\)](#) (p. 109)。
- **硬件 MFA 设备。**一台硬件设备，基于进行了时间同步的一次性密码算法生成一个六位数字代码。在登录时，用户必须在另一个网页上键入来自该设备的有效代码。分配给用户的每台 MFA 设备必须是唯一的。用户无法从另一个用户的设备键入代码来进行身份验证。有关受支持硬件 MFA 设备的信息，请参阅[多重身份验证](#)。有关使用 Amazon 设置硬件 MFA 设备的说明，请参阅[启用硬件 MFA 设备 \(控制台\)](#) (p. 114)。

Note

基于短信的 MFA。Amazon 已终止对短信多重身份验证 (MFA) 的支持。我们建议拥有使用基于短信的 MFA 的 IAM 用户的客户切换到以下替代方法之一：[虚拟 \(基于软件\) MFA 设备 \(p. 106\)](#)、[FIDO 安全密钥 \(p. 109\)](#) 或 [硬件 MFA 设备 \(p. 114\)](#)。您可以确定账户中拥有已分配短信 MFA 设备的用户。为此，请转到 IAM 控制台，从导航窗格中选择用户，然后在表的 MFA 列中查找具有 SMS 的用户。

为 Amazon 中的用户启用 MFA 设备

用于配置 MFA 的步骤取决于您使用的 MFA 设备类型。

主题

- [启用 MFA 设备的一般步骤 \(p. 106\)](#)
- [启用虚拟 Multi-Factor Authentication \(MFA\) 设备 \(控制台\) \(p. 106\)](#)
- [启用 FIDO 安全密钥 \(控制台\) \(p. 109\)](#)
- [启用硬件 MFA 设备 \(控制台\) \(p. 114\)](#)
- [启用和管理虚拟 MFA 设备 \(Amazon CLI 或 Amazon API\) \(p. 119\)](#)

启用 MFA 设备的一般步骤

以下概述流程介绍了如何设置和使用 MFA，并提供了相关信息的链接。

注意

您也可以观看[如何设置 Amazon 多重身份验证 \(MFA\)](#) 和 [Amazon 预算提醒](#) 这一英语视频，了解更多信息。

1. 获取 MFA 设备，如下列设备之一。您只能为每个 Amazon Web Services 账户 根用户或 IAM 用户启用一台 MFA 设备。
 - 虚拟 MFA 设备是符合 [RFC 6238](#)，一种基于标准的 TOTP（基于时间的一次性密码）算法的软件应用程序。您可以在手机或其他设备上安装该应用程序。有关可用作虚拟 MFA 设备的一些受支持应用程序的列表，请参阅 [Multi-Factor Authentication](#)。
 - 具有 [Amazon 支持的配置 \(p. 113\)](#) 的 FIDO 安全密钥，如 [Multi-Factor Authentication \(多重身份验证\)](#) 页面中讨论的 FIDO2 设备之一。
 - 基于硬件的 MFA 设备，如 [多重验证](#) 页面中讨论的受 Amazon 支持的硬件令牌设备之一。
2. 启用 MFA 设备。
 - 使用虚拟或硬件 MFA 设备的 IAM 用户：从 Amazon Web Services Management Console、Amazon CLI 或 IAM API 启用。
 - 具有 FIDO 安全密钥的 IAM 用户：仅从 Amazon Web Services Management Console 启用。
 - 具有任何类型 MFA 设备的 Amazon 账户根用户：仅从 Amazon Web Services Management Console 启用。

有关启用每种类型的 MFA 设备的信息，请参阅以下页面：

- 虚拟 MFA 设备：[启用虚拟 Multi-Factor Authentication \(MFA\) 设备（控制台）\(p. 106\)](#)
 - FIDO 安全密钥：[启用 FIDO 安全密钥（控制台）\(p. 109\)](#)
 - 硬件 MFA 设备：[启用硬件 MFA 设备（控制台）\(p. 114\)](#)
3. 在您登录或访问 Amazon 资源时使用 MFA 设备。请注意以下几点：
 - FIDO 安全密钥：要访问 Amazon 网站，请输入您的凭证，然后在出现提示时点击 FIDO 安全密钥。
 - 虚拟 MFA 设备和硬件 MFA 设备：要访问 Amazon 网站，您需要设备中的 MFA 代码以及您的用户名和密码。

要访问受 MFA 保护的 API 操作，您需要以下信息：

- MFA 代码
- MFA 设备的标识符（物理设备的设备序列号或 Amazon 中定义的虚拟设备的 ARN）
- 常规访问密钥 ID 和私有访问密钥

注意

- 您无法将 FIDO 安全密钥的 MFA 信息传递给 Amazon STS API 操作来请求临时凭证。
- 您无法使用 Amazon CLI 命令或 Amazon API 操作来启用 [FIDO 安全密钥 \(p. 109\)](#)。

有关更多信息，请参阅[使用 MFA 设备访问您的 IAM 登录页面 \(p. 78\)](#)。

启用虚拟 Multi-Factor Authentication (MFA) 设备（控制台）

您可以将手机或其他设备作为虚拟 Multi-Factor Authentication (MFA) 设备。为此，请安装符合 [RFC 6238](#) 的移动应用程序，这是一种基于标准的 TOTP（基于时间的一次性密码）算法。这些应用程序生成六位数的身份验证代码。由于虚拟 MFA 可能在不安全的移动设备上运行，因此，它们可能无法提供与 U2F 设备或硬件 MFA 设备相同的安全级别。我们建议您在等待硬件购买批准或等待硬件到达时使用虚拟 MFA 设备。

大多数虚拟 MFA 应用程序支持创建多个虚拟设备，从而允许您在多个 Amazon 账户或用户中使用相同的应用程序。但是，您只能为每个用户启用一个 MFA 设备。

有关您可以使用的虚拟 MFA 应用程序的列表，请参阅 [Multi-Factor Authentication](#)。请注意，Amazon 需要可产生六位数 OTP 的虚拟 MFA 应用程序。

主题

- [所需权限 \(p. 107\)](#)
- [为 IAM 用户启用虚拟 MFA 设备 \(控制台\) \(p. 107\)](#)
- [为您的 Amazon 账户根用户启用虚拟 MFA 设备 \(控制台\) \(p. 108\)](#)
- [替换或“轮换”虚拟 MFA 设备 \(p. 109\)](#)

所需权限

要管理您的 IAM 用户的虚拟 MFA 设备，您必须具有以下策略中的权限：[Amazon：允许经过 MFA 身份验证的 IAM 用户在“My Security Credentials” \(我的安全凭证\) 页面上管理自己的 MFA 设备。](#) (p. 400)。

为 IAM 用户启用虚拟 MFA 设备 (控制台)

您可在 Amazon Web Services Management Console 中使用 IAM 为您账户中的 IAM 用户启用和管理虚拟 MFA 设备。您可以将标签附加到 IAM 资源 (包括虚拟 MFA 设备)，以识别、组织和控制对这些资源的访问。只有在使用 Amazon CLI 或 Amazon API 时，才能标记虚拟 MFA 设备。要使用 Amazon CLI 或 Amazon API 启用和管理 MFA 设备，请参阅 [启用和管理虚拟 MFA 设备 \(Amazon CLI 或 Amazon API\)](#) (p. 119)。要了解有关标记 IAM 资源的更多信息，请参阅 [标记 IAM 资源](#) (p. 276)。

Note

您必须拥有对将托管用户的虚拟 MFA 设备的硬件的物理访问权限以便配置 MFA。例如，您可能为使用在智能手机上运行的虚拟 MFA 设备的用户配置 MFA。在这种情况下，您必须具有智能手机才能完成该向导。因此，您可能想让用户配置和管理他们自己的虚拟 MFA 设备。在此情况下，您必须授予用户执行必要的 IAM 操作所需的权限。有关更多信息以及授予这些权限的 IAM policy 示例，请参阅 [Amazon：允许经过 MFA 身份验证的 IAM 用户在“My Security Credentials” \(我的安全凭证\) 页面上管理自己的 MFA 设备。](#) (p. 400)。

为 IAM 用户启用虚拟 MFA 设备 (控制台)

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 在 User Name 列表中，选择目标 MFA 用户的名称。
4. 选择 Security Credentials (安全凭证) 选项卡。在 Assigned MFA device (已分配 MFA 设备) 旁边，选择 Manage (管理)。
5. 在 Manage MFA Device (管理 MFA 设备) 向导中，选择 Virtual MFA device (虚拟 MFA 设备)，然后选择 Continue (继续)。

IAM 将生成并显示虚拟 MFA 设备的配置信息，包括 QR 代码图形。此图形是秘密配置密钥的表示形式，适用于不支持 QR 代码的设备上的手动输入。

6. 打开您的虚拟 MFA 应用程序。有关可用于托管虚拟 MFA 设备的应用程序的列表，请参阅 [多重身份验证](#)。

如果虚拟 MFA 应用程序支持多个虚拟 MFA 设备或账户，请选择相应的选项以创建新的虚拟 MFA 设备或账户。

7. 确定 MFA 应用程序是否支持 QR 代码，然后执行以下操作之一：
 - 在向导中，选择 Show QR code (显示 QR 代码)，然后使用该应用程序扫描 QR 代码。例如，您可选择摄像头图标或选择类似于 Scan code 的选项，然后使用设备的摄像头扫描此代码。
 - 在 Manage MFA Device (管理 MFA 设备) 向导中，选择 Show secret key (显示私有密钥)，然后在您的 MFA 应用程序中键入私有密钥。

完成操作后，虚拟 MFA 设备会开始生成一次性密码。

- 在 Manage MFA Device (管理 MFA 设备) 向导的 MFA code 1 (MFA 代码 1) 框中，键入虚拟 MFA 设备上当前显示的一次性密码。请等候 30 秒，以便设备生成新的一次性密码。然后在 MFA code 2 (MFA 代码 2) 框中键入第二个一次性密码。选择 Assign MFA (分配 MFA)。

Important

生成代码之后立即提交您的请求。如果生成代码后等待很长时间才提交请求，MFA 设备会成功与用户关联，但 MFA 设备无法同步。这是因为基于时间的一次性密码 (TOTP) 很快会过期。这种情况下，您可以[重新同步设备](#) (p. 122)。

虚拟 MFA 设备现在已准备好与 Amazon 一起使用了。有关在 Amazon Web Services Management Console 上使用 MFA 的信息，请参阅 [使用 MFA 设备访问您的 IAM 登录页面](#) (p. 78)。

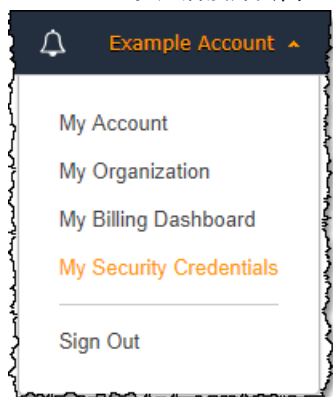
为您的 Amazon 账户根用户启用虚拟 MFA 设备 (控制台)

您可以使用 Amazon Web Services Management Console 为根用户配置和启用虚拟 MFA 设备。要为 Amazon 账户启用 MFA 设备，您必须使用根用户凭证登录 Amazon。

为您的根用户启用 MFA 之前，请检查您的账户设置和联系信息，以确保您可以访问电子邮件地址和电话号码。如果您的 MFA 设备丢失、被盗或无法工作，您仍可以通过使用相应的电子邮件和电话号码验证您的身份，以作为根用户登录。要了解如何使用这些替代的身份验证因素登录，请参阅 [若 MFA 设备遗失或停止工作，该怎么办？](#) (p. 127)。

配置和启用虚拟 MFA 设备以用于您的根用户 (控制台)

- 登录到 Amazon Web Services Management Console。
- 在导航栏右侧，选择您的账户名称，然后选择我的安全凭证。如有必要，选择 Continue to Security Credentials。然后展开页面上的 Multi-Factor Authentication (MFA) 部分。



- 选择 Activate MFA (激活 MFA)。
- 在向导中，选择 Virtual MFA device (虚拟 MFA 设备)，然后选择 Continue (继续)。

IAM 将生成并显示虚拟 MFA 设备的配置信息，包括 QR 代码图形。此图形是秘密配置密钥的表示形式，适用于不支持 QR 代码的设备上的手动输入。

- 在设备上打开虚拟 MFA 应用程序。

如果虚拟 MFA 应用程序支持多个虚拟 MFA 设备或账户，请选择相应的选项以创建新的虚拟 MFA 设备或账户。

- 要配置应用程序，最简单的方法是使用应用程序扫描 QR 代码。如果您无法扫描代码，则可手动键入配置信息。IAM 生成的 QR 代码和私有配置密钥与您的 Amazon 账户关联，不能用于其他账户。但是，如果您失去对原始 MFA 设备的访问权，可以重新使用它们为您的账户配置新的 MFA 设备。

- 要使用 QR 代码配置虚拟 MFA 设备，请在向导中，选择 Show QR code (显示 QR 代码)。然后，按照应用程序说明扫描代码。例如，您可能需要选择摄像头图标或选择扫描账户条形码等命令，然后使用设备的摄像头扫描 QR 代码。
- 在 Manage MFA Device (管理 MFA 设备) 向导中，选择 Show secret key (显示私有密钥)，然后在您的 MFA 应用程序中键入私有密钥。

Important

对 QR 代码或私有配置密钥进行安全备份，或确保启用为您的账户启用多个虚拟 MFA 设备。虚拟 MFA 设备可能变为不可用 (例如，如果丢失了承载虚拟 MFA 设备的智能手机)。如果您无法使用 [恢复根用户用户 MFA 设备 \(p. 127\)](#) 登录您的账户，您将无法登录该账户，必须与客户服务部门联系才能取消账户的 MFA 保护。

设备开始生成六位数编码。

7. 在管理 MFA 设备向导的 MFA 代码 1 框中，输入 MFA 设备当前显示的六位数字。请等候 30 秒，以便设备生成新的编码，然后在 MFA Code 2 (MFA 代码 2) 框中输入新的六位数编码。

Important

生成代码之后立即提交您的请求。如果生成代码后等待很长时间才提交请求，MFA 设备会成功与用户关联，但 MFA 设备无法同步。这是因为基于时间的一次性密码 (TOTP) 很快会过期。这种情况下，您可以 [重新同步设备 \(p. 122\)](#)。

8. 选择 Assign MFA (分配 MFA)，然后选择 Finish (完成)。

设备已准备就绪，可在 Amazon 上使用。有关在 Amazon Web Services Management Console 上使用 MFA 的信息，请参阅 [使用 MFA 设备访问您的 IAM 登录页面 \(p. 78\)](#)。

替换或“轮换”虚拟 MFA 设备

您一次只能将一台 MFA 设备分配给用户。如果用户丢失设备或出于任何原因需要替换它，您必须先停用旧设备。然后，您可以为用户添加新设备。

- 要停用当前与其他 IAM 用户关联的设备，请参阅 [停用 MFA 设备 \(p. 126\)](#)。
- 要为其他 IAM 用户添加替换虚拟 MFA 设备，请按照上述 [为 IAM 用户启用虚拟 MFA 设备 \(控制台\) \(p. 107\)](#) 过程中的步骤进行操作。
- 要为 Amazon Web Services 账户根用户添加替代用虚拟 MFA 设备，请按照本主题前面部分的过程 [为您的 Amazon 账户根用户启用虚拟 MFA 设备 \(控制台\) \(p. 108\)](#) 中的步骤操作。

启用 FIDO 安全密钥 (控制台)

FIDO 安全密钥是一种 [MFA 设备 \(p. 104\)](#)，可用于保护您的 Amazon 资源。您将 FIDO 安全密钥插入到您计算机上的 USB 端口，并使用随后显示的说明启用该密钥。启用后，在出现提示时点击它即可安全完成登录过程。如果您已将 FIDO 安全密钥用于其他服务，它将有一个 [Amazon 支持的配置 \(p. 113\)](#) (例如，来自 Yubico 的 Yubikey 5 Series)，您也可以将其用于 Amazon。否则，如果要在 Amazon 中使用面向 MFA 的 Webauthn，您需要购买 FIDO2 安全密钥。有关规格和购买信息，请参阅 [多重身份验证](#)。

FIDO2 是开放身份验证标准，作为 FIDO U2F 的扩展，基于公有密钥加密提供同样高级别的安全性。FIDO2 由 W3C Web 身份验证规范 (WebAuthn API) 和客户端到身份验证协议 (CTAP，一种应用程序层协议) 组成。CTAP 支持客户端或平台 (如浏览器或操作系统) 与外部身份验证器之间进行通信。您可以继续使用符合 FIDO 标准的设备，例如 FIDO U2F 安全密钥。当您在 Amazon 中启用符合 FIDO 标准的身份验证器时，FIDO 安全密钥会创建仅适用于 Amazon 的新密钥对。首先，输入您的凭证。出现提示时，点击 FIDO 安全密钥，这会响应 Amazon 发出的身份验证质询。要了解有关 FIDO2 标准的更多信息，请参阅 [FIDO2 项目](#)。

您可以为每个根用户或 IAM 用户启用一台 MFA 设备 (任何类型)。

主题

- [所需权限 \(p. 110\)](#)
- [为您自己的 IAM 用户启用 FIDO 安全密钥 \(控制台\) \(p. 110\)](#)
- [为其他 IAM 用户启用 FIDO 安全密钥 \(控制台\) \(p. 111\)](#)
- [为 Amazon 账户根用户启用 FIDO 安全密钥 \(控制台\) \(p. 112\)](#)
- [替换 FIDO 安全密钥 \(p. 113\)](#)
- [使用 FIDO 安全密钥的受支持配置 \(p. 113\)](#)

所需权限

要为您自己的 IAM 用户管理 FIDO 安全密钥，同时保护与 MFA 相关的敏感操作，您必须具有以下策略中的权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowManageOwnUserMFA",
      "Effect": "Allow",
      "Action": [
        "iam:DeactivateMFADevice",
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:ListMFADevices",
        "iam:ResyncMFADevice"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "DenyAllExceptListedIfNoMFA",
      "Effect": "Deny",
      "NotAction": [
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:ListMFADevices",
        "iam:ResyncMFADevice"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}",
      "Condition": {
        "BoolIfExists": {
          "aws:MultiFactorAuthPresent": "false"
        }
      }
    }
  ]
}
```

为您自己的 IAM 用户启用 FIDO 安全密钥 (控制台)

您只能从 Amazon Web Services Management Console 为您自己的 IAM 用户启用 FIDO 安全密钥，而不能从 Amazon CLI 或 Amazon API 启用。

Note

在启用 FIDO 安全密钥之前，您必须拥有对设备的物理访问权限。

Note

您不应选择 Google Chrome 弹出窗口中的任何要求 Verify your identity with amazon.com (通过 amazon.com 验证您的身份) 的可用选项。您只需要点击安全密钥即可。

为您自己的 IAM 用户启用 FIDO 安全密钥 (控制台)

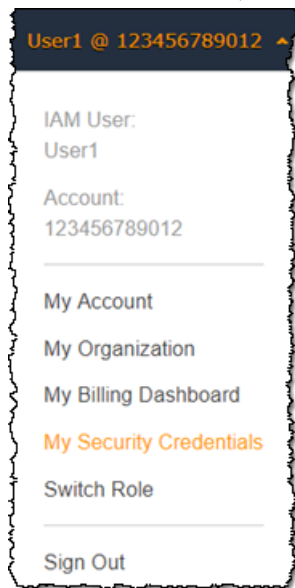
1. 使用 Amazon 账户 ID 或账户别名、您的 IAM 用户名和密码登录到 [IAM 控制台](#)。

Note

为方便起见，Amazon 登录页面使用浏览器 Cookie 记住您的 IAM 用户名和账户信息。如果您之前以其他用户身份登录，请选择页面底部的 Sign-in to a different account (登录到其他账户) 以返回主登录页面。在此处，您可以输入要重新导向到您账户 IAM 用户登录页面的 Amazon 账户 ID 或账户别名。

要获取 Amazon 账户 ID，请联系管理员。

2. 在右上角的导航栏中，选择您的用户名，然后选择 My Security Credentials (我的安全凭证)。



3. 在 Amazon IAM 凭证选项卡的 Multi-Factor Authentication (多重验证) 部分中，选择 Manage MFA device (管理 MFA 设备)。
4. 在 Manage MFA device (管理 MFA 设备) 向导中，选择 FIDO security key (FIDO 安全密钥)，然后选择 Continue (继续)。
5. 将 FIDO 安全密钥插入计算机的 USB 端口。



6. 点击 FIDO2 安全密钥，然后在安装完成时选择 Close (关闭)。

FIDO2 安全密钥现已就绪，可用于 Amazon。有关在 Amazon Web Services Management Console 上使用 MFA 的信息，请参阅 [使用 MFA 设备访问您的 IAM 登录页面 \(p. 78\)](#)。

为其他 IAM 用户启用 FIDO 安全密钥 (控制台)

您只能从 Amazon Web Services Management Console 为其他 IAM 用户启用 FIDO 安全密钥，而不能从 Amazon CLI 或 Amazon API 启用。

为其他 IAM 用户启用 FIDO 安全密钥 (控制台)

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 选择要为其启用 MFA 的用户的名称，然后选择 Security credentials (安全凭证) 选项卡。
4. 在 Assigned MFA device (已分配 MFA 设备) 旁边，选择 Manage (管理)。
5. 在 Manage MFA device (管理 MFA 设备) 向导中，选择 FIDO security key (FIDO 安全密钥)，然后选择 Continue (继续)。
6. 将 FIDO 安全密钥插入计算机的 USB 端口。



7. 单击 FIDO 安全密钥，然后在安装完成时选择 Close (关闭)。

FIDO 安全密钥现已就绪，可用于 Amazon。有关在 Amazon Web Services Management Console 上使用 MFA 的信息，请参阅 [使用 MFA 设备访问您的 IAM 登录页面 \(p. 78\)](#)。

为 Amazon 账户根用户启用 FIDO 安全密钥 (控制台)

您只能从 Amazon Web Services Management Console 为根用户配置和启用虚拟 MFA 设备，而不能从 Amazon CLI 或 Amazon API 配置和启用。

如果 FIDO 安全密钥丢失、被盗或无法正常工作，您仍可使用替代的身份验证因素登录。要了解如何使用替代的身份验证因素登录，请参阅 [若 MFA 设备遗失或停止工作，该怎么办？ \(p. 127\)](#)。要禁用此功能，请联系 [Amazon Web Services Support](#)。

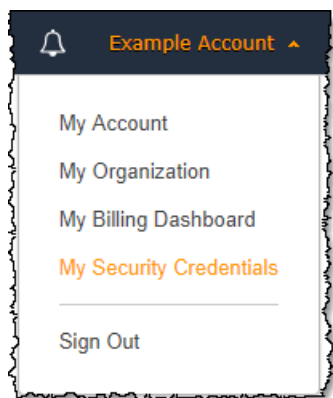
为根用户启用 FIDO 密钥 (控制台)

1. 选择 Root user (根用户) 并输入您的 Amazon Web Services 账户 电子邮件地址，以账户拥有者身份登录 [IAM 控制台](#)。在下一页上，输入您的密码。

Note

如果显示了三个文本框，则您之前已使用 [IAM 用户](#) 凭证登录控制台。您的浏览器可能会记住此首选项，并在您每次尝试登录时打开此账户特定的登录页面。您无法使用 IAM 用户登录页面以账户拥有者身份进行登录。如果您看到 [IAM 用户登录页面](#)，请选择该页面底部附近的 Sign in using root user email (使用根用户电子邮件登录)。这将返回主登录页面。在该页面中，您可以使用 Amazon Web Services 账户 电子邮件地址和密码以根用户登录。

2. 在导航栏的右侧选择您的账户名称，然后选择 My Security Credentials (我的安全凭证)。如有必要，选择 Continue to Security Credentials。



3. 展开 Multi-factor authentication (MFA) (多重验证) 部分。
4. 根据在上一步选择的选项，选择 Manage MFA 或 Activate MFA。
5. 在向导中，选择 FIDO security key (FIDO 安全密钥)，然后选择 Continue (继续)。
6. 将 FIDO 安全密钥插入计算机的 USB 端口。



7. 点击 FIDO 安全密钥，然后在安装完成时选择 Close (关闭)。

FIDO 安全密钥现已就绪，可用于 Amazon。下次使用根用户凭证登录时，您必须点击您的 FIDO 安全密钥以完成登录过程。

替换 FIDO 安全密钥

您一次只能向一个用户分配一台 MFA 设备 (虚拟、FIDO 安全密钥或硬件)。如果用户丢失符合 FIDO 的身份验证器或者出于某种原因需要进行替换，必须先停用符合 FIDO 的旧身份验证器。然后，您可以为用户添加新 MFA 设备。

- 要停用当前与用户关联的设备，请参阅[停用 MFA 设备 \(p. 126\)](#)。
- 要为 IAM 用户添加新的 FIDO 安全密钥，请参阅[启用 FIDO 安全密钥 \(控制台\) \(p. 109\)](#)。

如果您无权访问新的 FIDO 安全密钥，则可以启用新的虚拟 MFA 设备或硬件 MFA 设备。有关说明，请参阅以下章节之一：

- [启用虚拟 Multi-Factor Authentication \(MFA\) 设备 \(控制台\) \(p. 106\)](#)
- [启用硬件 MFA 设备 \(控制台\) \(p. 114\)](#)

使用 FIDO 安全密钥的受支持配置

您可以通过使用当前支持的配置，在 IAM 中将 WebAuthn 用作多重身份验证 (MFA) 方法。其中包括 IAM 支持的经过 FIDO 认证的设备及支持 WebAuthn 的浏览器。

Amazon 支持的 FIDO 设备

IAM 目前支持可插入计算机上的 USB 端口且经过 FIDO 认证的安全设备。我们目前不支持平台身份验证器，如 TouchID、FaceID 或 Windows Hello。

Note

Amazon 需要访问您的计算机上的物理 USB 端口以验证您的经过 FIDO 认证的设备。WebAuthn MFA 不适用于虚拟机、远程连接或浏览器的无痕模式。

有关购买受支持设备的信息，请参阅 [多重身份验证](#)。

支持 WebAuthn 的浏览器

以下浏览器当前支持经过 FIDO 认证的安全密钥：

- Google Chrome 版本 67 及更高版本。
- Apple Safari 版本 14 及更高版本。
- Microsoft Edge 1809 版及更高版本。
- Mozilla Firefox 版本 60 及更高版本。

Note

当前支持 WebAuthn 的大多数 Firefox 版本默认情况下不启用支持。有关在 Firefox 中启用 WebAuthn 支持的说明，请参阅 [FIDO 安全密钥故障排除 \(p. 882\)](#)。

浏览器插件

Amazon 当前仅支持本机支持 FIDO2/WebAuthn 标准的浏览器。Amazon 不支持使用插件添加 FIDO2/WebAuthn 浏览器支持。另请注意，某些浏览器插件与 U2F 标准不兼容，这可能会导致 FIDO2 安全密钥产生意外结果。

有关禁用浏览器插件和其他问题排查提示的信息，请参阅 [我无法启用我的 FIDO 安全密钥 \(p. 882\)](#)。

移动环境

以下浏览器当前支持经过 FIDO 认证的安全密钥：

- Android 7+，配备 Google Chrome。
- iOS 14.5+，配备 Apple Safari、Google Chrome 和 Mozilla Firefox。

Note

Amazon 控制台移动应用程序当前不支持对 MFA 使用经过 FIDO 认证的安全密钥。

Amazon CLI 和 Amazon API

Amazon 当前支持仅在 Amazon Web Services Management Console 中使用经过 FIDO 认证的安全密钥。[Amazon CLI](#) 和 [Amazon API](#) 当前不支持对 MFA 使用经过 FIDO 认证的安全密钥，也不支持使用这些密钥访问 [MFA 保护的 API 操作 \(p. 129\)](#)。

其他资源

- 有关在 Amazon 中使用经过 FIDO 认证的安全密钥的更多信息，请参阅 [启用 FIDO 安全密钥 \(控制台\) \(p. 109\)](#)。
- 有关在 Amazon 中对 FIDO 进行问题排查的帮助信息，请参阅 [FIDO 安全密钥故障排除 \(p. 882\)](#)。
- 有关 FIDO 支持的一般行业信息，请参阅 [FIDO2 项目](#)。

启用硬件 MFA 设备 (控制台)

硬件 MFA 设备将基于进行了时间同步的一次性密码算法生成一个六位数字代码。在登录过程中，用户必须在出现提示时从该设备键入有效代码。分配给用户的每个 MFA 设备都必须是唯一的；进行身份验证时，某个用户无法从另一个用户的设备键入代码。

硬件 MFA 设备和 [FIDO 安全密钥](#) (p. 109) 都是您购买的物理设备。不同之处在于：硬件 MFA 设备生成代码，您可以进行查看，然后在登录 Amazon 期间出现提示时输入代码。使用 FIDO 安全密钥时，您不会看到或键入身份验证代码。相反，FIDO 安全密钥会生成响应而不将其呈现给用户，随后服务会对响应进行验证。有关这两种设备类型的规格和购买信息，请参阅[多重身份验证](#)。

您可以从 Amazon Web Services Management Console、命令行或 IAM API 为 IAM 用户启用硬件 MFA 设备。要为您的 Amazon Web Services 账户根用户启用 MFA 设备，请参阅[为 Amazon 账户根用户启用硬件 MFA 设备（控制台）](#) (p. 117)。

您可以为每个根用户或 IAM 用户启用一台 MFA 设备（任何类型）。

Note

如果想从命令行启用设备，请使用 `iam-userenablemfadevice aws iam enable-mfa-device`。如要使用 IAM API 启用 MFA 设备，请使用 `EnableMFADevice` 操作。

主题

- [所需权限](#) (p. 115)
- [为您自己的 IAM 用户启用硬件 MFA 设备（控制台）](#) (p. 116)
- [为其他 IAM 用户启用硬件 MFA 设备（控制台）](#) (p. 117)
- [为 Amazon 账户根用户启用硬件 MFA 设备（控制台）](#) (p. 117)
- [替换或“轮换”物理 MFA 设备](#) (p. 119)

所需权限

要为您自己的 IAM 用户管理硬件 MFA 设备，同时保护与 MFA 相关的敏感操作，您必须具有以下策略中的权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowManageOwnUserMFA",
      "Effect": "Allow",
      "Action": [
        "iam:DeactivateMFADevice",
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:ListMFADevices",
        "iam:ResyncMFADevice"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "DenyAllExceptListedIfNoMFA",
      "Effect": "Deny",
      "NotAction": [
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:ListMFADevices",
        "iam:ResyncMFADevice"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}",
      "Condition": {
        "BoolIfExists": {
          "aws:MultiFactorAuthPresent": "false"
        }
      }
    }
  ]
}
```

}

为您自己的 IAM 用户启用硬件 MFA 设备 (控制台)

您可以从 Amazon Web Services Management Console 启用您自己的硬件 MFA 设备。

Note

在启用硬件 MFA 设备之前，您必须拥有对设备的物理访问权限。

为您自己的 IAM 用户启用硬件 MFA 设备 (控制台)

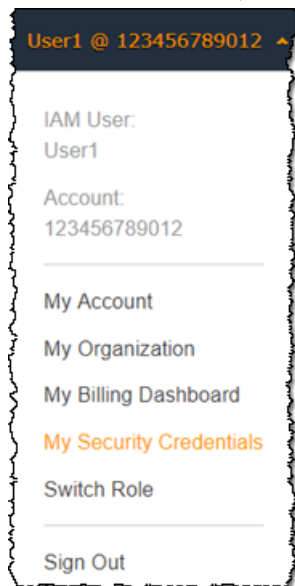
1. 使用 Amazon 账户 ID 或账户别名、您的 IAM 用户名和密码登录到 [IAM 控制台](#)。

Note

为方便起见，Amazon 登录页面使用浏览器 Cookie 记住您的 IAM 用户名和账户信息。如果您之前以其他用户身份登录，请选择页面底部的 Sign-in to a different account (登录到其他账户) 以返回主登录页面。在此处，您可以输入要重新导向到您账户 IAM 用户登录页面的 Amazon 账户 ID 或账户别名。

要获取 Amazon 账户 ID，请联系管理员。

2. 在右上角的导航栏中，选择您的用户名，然后选择 My Security Credentials (我的安全凭证)。



3. 在 Amazon IAM 凭证选项卡的 Multi-Factor Authentication (多重验证) 部分中，选择 Manage MFA device (管理 MFA 设备)。
4. 在 Manage MFA device (管理 MFA 设备) 向导中，选择 Hardware MFA device (硬件 MFA 设备)，然后选择 Continue (继续)。
5. 键入设备序列号。序列号通常位于设备的背面。
6. 在 MFA code 1 (MFA 代码 1) 框中，输入 MFA 设备显示的六位数编码。您需要按设备正面的按钮来显示编码。



7. 在设备刷新期间等候 30 秒，然后在 MFA code 2 (MFA 代码 2) 框中键入第二个六位数编码。您需要再次按设备正面的按钮来显示第二个编码。

8. 选择 Assign MFA (分配 MFA)。

Important

在生成身份验证代码后立即提交您的请求。如果生成代码后等待很长时间才提交请求，MFA 设备会成功与用户关联，但 MFA 设备无法同步。这是因为基于时间的一次性密码 (TOTP) 很快会过期。这种情况下，您可以[重新同步设备 \(p. 122\)](#)。

设备已准备就绪，可在 Amazon 上使用。有关在 Amazon Web Services Management Console 上使用 MFA 的信息，请参阅 [使用 MFA 设备访问您的 IAM 登录页面 \(p. 78\)](#)。

为其他 IAM 用户启用硬件 MFA 设备 (控制台)

您可以从 Amazon Web Services Management Console 为其他 IAM 用户启用硬件 MFA 设备。

为其他 IAM 用户启用硬件 MFA 设备 (控制台)

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 选择要为其启用 MFA 的用户的名称，然后选择 Security credentials (安全凭证) 选项卡。
4. 在 Assigned MFA device (已分配 MFA 设备) 旁边，选择 Manage (管理)。
5. 在 Manage MFA device (管理 MFA 设备) 向导中，选择 Hardware MFA device (硬件 MFA 设备)，然后选择 Continue (继续)。
6. 键入设备序列号。序列号通常位于设备的背面。
7. 在 MFA code 1 (MFA 代码 1) 框中，输入 MFA 设备显示的六位数编码。您需要按设备正面的按钮来显示编码。



8. 在设备刷新期间等候 30 秒，然后在 MFA code 2 (MFA 代码 2) 框中键入第二个六位数编码。您需要再次按设备正面的按钮来显示第二个编码。
9. 选择 Assign MFA (分配 MFA)。

Important

在生成身份验证代码后立即提交您的请求。如果生成代码后等待很长时间才提交请求，MFA 设备会成功与用户关联，但 MFA 设备无法同步。这是因为基于时间的一次性密码 (TOTP) 很快会过期。这种情况下，您可以[重新同步设备 \(p. 122\)](#)。

设备已准备就绪，可在 Amazon 上使用。有关在 Amazon Web Services Management Console 上使用 MFA 的信息，请参阅 [使用 MFA 设备访问您的 IAM 登录页面 \(p. 78\)](#)。

为 Amazon 账户根用户启用硬件 MFA 设备 (控制台)

您只能从 Amazon Web Services Management Console 为根用户配置和启用虚拟 MFA 设备，而不能从 Amazon CLI 或 Amazon API 配置和启用。

如果您的 MFA 设备丢失、被盗或无法正常运行，您仍然可以使用替代的身份验证因素登录。如果您不能使用 MFA 设备登录，则可以使用和您的账户一起注册的电子邮件和电话验证您的身份来进行登录。为您的根用户启用 MFA 之前，请检查您的账户设置和联系信息，以确保您可以访问电子邮件地址和电话号码。要了解如何使用替代的身份验证因素登录，请参阅[若 MFA 设备遗失或停止工作，该怎么办？ \(p. 127\)](#)。要禁用此功能，请联系[Amazon Web Services Support](#)。

Note

您可能会看到不同的文本，例如使用 MFA 登录和排除您的身份验证设备故障。不过，它们提供了相同的功能。在任一情况下，如果您无法使用替代身份验证因素验证您的账户电子邮件地址和电话号码，请与 [Amazon Web Services Support](#) 联系以停用您的 MFA 设置。

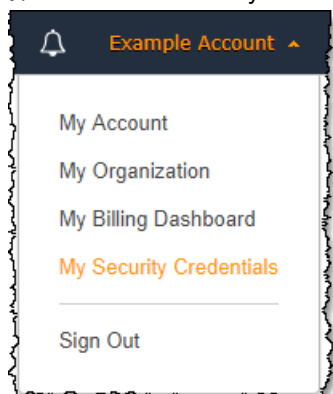
为根用户启用 MFA 设备 (控制台)

1. 选择 Root user (根用户) 并输入您的 Amazon Web Services 账户 电子邮件地址，以账户拥有者身份登录 [IAM 控制台](#)。在下一页上，输入您的密码。

Note

如果显示了三个文本框，则您之前已使用 [IAM 用户](#) 凭证登录控制台。您的浏览器可能会记住此首选项，并在您每次尝试登录时打开此账户特定的登录页面。您无法使用 IAM 用户登录页面以账户拥有者身份进行登录。如果您看到 [IAM 用户登录页面](#)，请选择该页面底部附近的 Sign in using root user email (使用根用户电子邮件登录)。这将返回主登录页面。在该页面中，您可以使用 Amazon Web Services 账户 电子邮件地址和密码以根用户登录。

2. 在导航栏的右侧选择您的账户名称，然后选择 My Security Credentials (我的安全凭证)。如有必要，选择 Continue to Security Credentials。



3. 展开 Multi-factor authentication (MFA) (多重验证) 部分。
4. 根据在上一步选择的选项，选择 Manage MFA 或 Activate MFA。
5. 在向导中，选择 Hardware MFA device (硬件 MFA 设备)，然后选择 Continue (继续)。
6. 在 Serial Number (序列号) 框中，键入在 MFA 设备背面找到的序列号。
7. 在 MFA code 1 (MFA 代码 1) 框中，输入 MFA 设备显示的六位数编码。您需要按设备正面的按钮来显示编码。



8. 在设备刷新期间等候 30 秒，然后在 MFA code 2 (MFA 代码 2) 框中键入第二个六位数编码。您需要再次按设备正面的按钮来显示第二个编码。
9. 选择 Assign MFA (分配 MFA)。MFA 设备现已与 Amazon 账户相关联。

Important

在生成身份验证代码后立即提交您的请求。如果生成代码后等待很长时间才提交请求，MFA 设备会成功与用户关联，但 MFA 设备无法同步。这是因为基于时间的一次性密码 (TOTP) 很快就会过期。这种情况下，您可以 [重新同步设备](#) (p. 122)。

下次使用根用户凭证登录时，您必须键入 MFA 设备生成的代码。

替换或“轮换”物理 MFA 设备

您一次只能将一台 MFA 设备分配给用户。如果用户丢失设备或出于任何原因需要替换它，您必须先停用旧设备。然后，您可以为用户添加新设备。

- 要停用当前与用户关联的设备，请参阅[停用 MFA 设备 \(p. 126\)](#)。
- 要为 IAM 用户添加替代用硬件 MFA 设备，请按照本主题前面部分的过程 [为其他 IAM 用户启用硬件 MFA 设备 \(控制台\) \(p. 117\)](#) 中的步骤进行操作。
- 要为 Amazon Web Services 账户根用户添加替代用虚拟 MFA 设备，请按照本主题前面部分的过程 [为 Amazon 账户根用户启用硬件 MFA 设备 \(控制台\) \(p. 117\)](#) 中的步骤操作。

启用和管理虚拟 MFA 设备 (Amazon CLI 或 Amazon API)

您可以使用 Amazon CLI 命令或 Amazon API 操作为 IAM 用户启用虚拟 MFA 设备。您无法使用 Amazon CLI、Amazon API、Tools for Windows PowerShell 或任何其他命令行工具为 Amazon Web Services 账户根用户启用 MFA 设备。不过，可以使用 Amazon Web Services Management Console 为根用户启用 MFA 设备。

当您从 Amazon Web Services Management Console 启用 MFA 设备时，控制台会为您执行多个步骤。如果您改用 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 创建虚拟设备，则必须按正确的顺序手动执行这些步骤。例如，要创建虚拟 MFA 设备，您必须创建 IAM 对象，将代码提取为字符串或 QR 代码图形。然后，您必须同步该设备并将其与 IAM 用户关联。有关更多详细信息，请参阅 [New-IAMVirtualMFADevice](#) 的 Examples (示例) 部分。对于物理设备，您可以跳过创建步骤，直接同步该设备并将其与用户关联。

您可以将标签附加到 IAM 资源 (包括虚拟 MFA 设备)，以识别、组织和控制对这些资源的访问。只有在使用 Amazon CLI 或 Amazon API 时，才能标记虚拟 MFA 设备。

在 IAM 中创建虚拟设备实体来代表虚拟 MFA 设备

这些命令提供在以下很多命令中代替序列号的设备 ARN。

- Amazon CLI: `aws iam create-virtual-mfa-device`
- Amazon API : `CreateVirtualMFADevice`

启用 MFA 设备，以便在 Amazon 上使用

这些命令将设备与 Amazon 同步，并将其与用户或根用户关联。如果设备是虚拟设备，则将虚拟设备的 ARN 用作序列号。

Important

在生成身份验证代码后立即提交您的请求。如果生成代码后等待很长时间才提交请求，MFA 设备会成功与用户关联，但 MFA 设备无法同步。这是因为基于时间的一次性密码 (TOTP) 很快会过期。如果发生这种情况，可以使用下面介绍的命令重新同步设备。

- Amazon CLI: `aws iam enable-mfa-device`
- Amazon API : `EnableMFADevice`

停用设备

使用这些命令可取消设备与用户的关联并停用设备。如果设备是虚拟设备，则将虚拟设备的 ARN 用作序列号。您还必须单独删除虚拟设备实体。

- Amazon CLI: `aws iam deactivate-mfa-device`

- Amazon API : [DeactivateMFADevice](#)

列出虚拟 MFA 设备实体

使用以下命令列出虚拟 MFA 设备实体。

- Amazon CLI: `aws iam list-virtual-mfa-devices`
- Amazon API : [ListVirtualMFADevices](#)

要标记虚拟 MFA 设备

使用这些命令来标记虚拟 MFA 设备。

- Amazon CLI: `aws iam tag-mfa-device`
- Amazon API : [TagMFADevice](#)

要为虚拟 MFA 设备列出标签

使用这些命令列出附加到虚拟 MFA 设备的标签。

- Amazon CLI: `aws iam list-mfa-device-tags`
- Amazon API : [ListMFADeviceTags](#)

要取消虚拟 MFA 设备的标签

使用这些命令删除附加到虚拟 MFA 设备的标签。

- Amazon CLI: `aws iam untag-mfa-device`
- Amazon API : [UntagMFADevice](#)

重新同步 MFA 设备

如果设备生成的代码不被 Amazon 接受，则使用这些命令。如果设备是虚拟设备，则将虚拟设备的 ARN 用作序列号。

- Amazon CLI: `aws iam resync-mfa-device`
- Amazon API : [ResyncMFADevice](#)

删除 IAM 中的虚拟 MFA 设备实体

在取消设备与用户的关联后，可以删除设备实体。

- Amazon CLI: `aws iam delete-virtual-mfa-device`
- Amazon API : [DeleteVirtualMFADevice](#)

恢复丢失或无法正常工作的虚拟 MFA 设备

有时，托管虚拟 MFA 应用程序的 IAM 用户设备丢失、更换或无法正常工作。在这种情况下，用户无法自行将其恢复。IAM 用户必须与管理员联系，才能将设备停用。有关更多信息，请参阅[若 MFA 设备遗失或停止工作，该怎么办？](#) (p. 127)。

检查 MFA 的状态

使用 IAM 控制台检查 Amazon Web Services 账户 根用户或 IAM 用户是否启用了有效的 MFA 设备。

检查根用户的 MFA 状态

1. 使用您的根用户凭证登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.amazonaws.cn/iam/>。
2. 在右上角的导航栏中，选择您的用户名，然后选择 My Security Credentials (我的安全凭证)。
3. 在 Multi-Factor Authentication (MFA) 下查看 MFA 是已启用还是已禁用。如果尚未激活 MFA，系统会显示一个提醒符号 ()。

如果要为账户启用 MFA，请参阅以下章节之一：

- 为您的 Amazon 账户根用户启用虚拟 MFA 设备 (控制台) (p. 108)
- 为 Amazon 账户根用户启用 FIDO 安全密钥 (控制台) (p. 112)
- 为 Amazon 账户根用户启用硬件 MFA 设备 (控制台) (p. 117)

检查 IAM 用户的 MFA 状态

1. 使用 <https://console.amazonaws.cn/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 如有必要，请通过完成以下步骤来将 MFA 列添加到用户表中：
 - a. 在最右侧的表上方，选择设置图标 ()。
 - b. 在 Manage Columns 中，选择 MFA。
 - c. (可选) 清除不希望在用户表中显示的任何列标题的复选框。
 - d. 选择 Close 返回到用户列表。
4. MFA 列告知您 MFA 设备的启用情况。如果用户没有处于活动状态的 MFA 设备，控制台将显示 None (无)。如果用户启用了 MFA 设备，则 MFA 列将显示启用的设备类型，并附上值 Virtual (虚拟)、FIDO Security Key (FIDO 安全密钥)、Hardware (硬件) 或 SMS (短信)。

Note

Amazon 已终止对短信多重身份验证 (MFA) 的支持。我们建议拥有使用基于短信的 MFA 的 IAM 用户的客户切换到以下替代方法之一：[虚拟 \(基于软件\) MFA 设备 \(p. 106\)](#)、[FIDO 安全密钥 \(p. 109\)](#) 或 [硬件 MFA 设备 \(p. 114\)](#)。您可以确定账户中拥有已分配短信 MFA 设备的用户。为此，请转到 IAM 控制台，从导航窗格中选择用户，然后在表的 MFA 列中查找具有 SMS 的用户。

5. 要查看有关用户的 MFA 设备的其他信息，请选择要检查其 MFA 状态的用户名称，然后选择 Security credentials 选项卡。
6. 如果用户没有处于活动状态的 MFA 设备，控制台的 Assigned MFA device (已分配 MFA 设备) 旁将显示 Not assigned (未分配)。如果用户启用了 MFA 设备，Assigned MFA Device 项目将显示该设备的值：
 - 硬件设备的设备序列号 (通常为设备背面的号码)，例如 GAHT12345678
 - SMS 设备的 Amazon 中的 ARN，例如 `arn:aws:iam::123456789012:sms-mfa/username`
 - 虚拟设备的 Amazon 中的 ARN，例如 `arn:aws:iam::123456789012:mfa/username`

如果要更改当前设置，请选择 Assigned MFA Device (已分配 MFA 设备) 旁的 Manage (管理)。

有关启用 MFA 的更多信息，请参阅以下章节：

- [启用虚拟 Multi-Factor Authentication \(MFA\) 设备 \(控制台\) \(p. 106\)](#)
- [启用 FIDO 安全密钥 \(控制台\) \(p. 109\)](#)

- [启用硬件 MFA 设备 \(控制台\) \(p. 114\)](#)

重新同步虚拟和硬件 MFA 设备

您可以使用 Amazon 重新同步虚拟和硬件 Multi-Factor Authentication (MFA) 设备。如果您尝试使用的设备并未同步，则将导致用户的登录尝试失败，IAM 会提示您重新同步设备。

Note

FIDO 安全密钥不同步。如果 FIDO 安全密钥丢失或损坏，您可以将其停用。有关停用任何 MFA 设备类型的说明，请参阅[停用其他 IAM 用户的 MFA 设备 \(控制台\) \(p. 126\)](#)。

作为 Amazon 管理员，您可以重新同步您的 IAM 用户无法同步的虚拟和硬件 MFA 设备。

如果您的 Amazon Web Services 账户 根用户 MFA 设备不工作，可以使用 IAM 控制台重新同步该设备（完成登录过程与否均可实现）。

主题

- [所需权限 \(p. 122\)](#)
- [重新同步虚拟和硬件 MFA 设备 \(IAM 控制台\) \(p. 123\)](#)
- [重新同步虚拟和硬件 MFA 设备 \(Amazon CLI\) \(p. 125\)](#)
- [重新同步虚拟和硬件 MFA 设备 \(Amazon API\) \(p. 125\)](#)

所需权限

要为您自己的 IAM 用户重新同步虚拟或硬件 MFA 设备，您必须具有以下策略中的权限：此策略不允许您创建或停用设备。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListActions",
      "Effect": "Allow",
      "Action": [
        "iam:ListVirtualMFADevices"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowUserToViewAndManageTheirOwnUserMFA",
      "Effect": "Allow",
      "Action": [
        "iam:ListMFADevices",
        "iam:ResyncMFADevice"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "BlockAllExceptListedIfNoMFA",
      "Effect": "Deny",
      "NotAction": [
        "iam:ListMFADevices",
        "iam:ListVirtualMFADevices",
        "iam:ResyncMFADevice"
      ],
      "Resource": "*",
      "Condition": {
```

```
        "BoolIfExists": {  
            "aws:MultiFactorAuthPresent": "false"  
        }  
    }  
}  
]  
}
```

重新同步虚拟和硬件 MFA 设备 (IAM 控制台)

您可以使用 IAM 控制台重新同步虚拟和硬件 MFA 设备。

重新同步您自己 IAM 用户的虚拟或硬件 MFA 设备 (控制台)

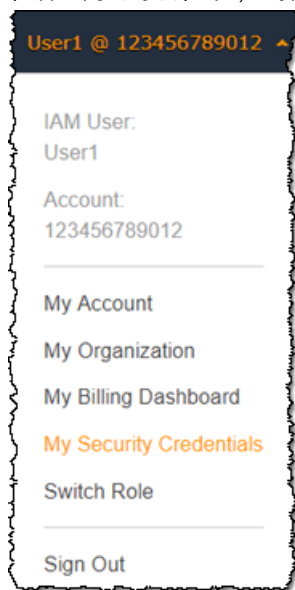
1. 使用 Amazon 账户 ID 或账户别名、您的 IAM 用户名和密码登录到 [IAM 控制台](#)。

Note

为方便起见，Amazon 登录页面使用浏览器 Cookie 记住您的 IAM 用户名和账户信息。如果您之前以其他用户身份登录，请选择页面底部的 Sign-in to a different account (登录到其他账户) 以返回主登录页面。在此处，您可以输入要重新导向到您账户 IAM 用户登录页面的 Amazon 账户 ID 或账户别名。

要获取 Amazon 账户 ID，请联系管理员。

2. 在右上角的导航栏中，选择您的用户名，然后选择 My Security Credentials (我的安全凭证)。



3. 在 Amazon IAM 凭证选项卡的 Multi-Factor Authentication (多重验证) 部分中，选择 Manage MFA device (管理 MFA 设备)。
4. 在 Manage MFA device (管理 MFA 设备) 向导中，选择 Resync (重新同步)，然后选择 Continue (继续)。
5. 依次将设备上按顺序生成的两个代码键入 MFA code 1 (MFA 代码 1) 和 MFA code 2 (MFA 代码 2) 中。然后选择 Continue (继续)。

Important

生成代码之后立即提交您的请求。如果您生成代码，然后等待了很长时间才提交请求，则请求看起来有效，但实际上设备仍然不同步。这是因为基于时间的一次性密码 (TOTP) 很快会过期。

重新同步其他 IAM 用户的虚拟或硬件 MFA 设备 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Users，然后选择其 MFA 设备需要重新同步的用户名称。
3. 选择 Security Credentials (安全凭证) 选项卡。在 Assigned MFA device (已分配 MFA 设备) 旁边，选择 Manage (管理)。
4. 在 Manage MFA device (管理 MFA 设备) 向导中，选择 Resync (重新同步)，然后选择 Continue (继续)。
5. 依次将设备上按顺序生成的两个代码键入 MFA code 1 (MFA 代码 1) 和 MFA code 2 (MFA 代码 2) 中。然后选择 Continue (继续)。

Important

生成代码之后立即提交您的请求。如果您生成代码，然后等待了很长时间才提交请求，则请求看起来有效，但实际上设备仍然不同步。这是因为基于时间的一次性密码 (TOTP) 很快会过期。

在登录前重新同步您的根用户 MFA (控制台)

1. 在 Amazon Web Services Sign In With Authentication Device (亚马逊云科技使用身份验证设备登录) 页面上，选择 Having problems with your authentication device? (身份验证设备出现问题?) Click here (点击此处)。

Note

您可能会看到不同的文本，例如使用 MFA 登录和排除您的身份验证设备故障。不过，它们提供了相同的功能。

2. 在 Re-Sync With Our Servers (与服务器重新同步) 部分中，依次将设备上按顺序生成的两个代码键入 MFA code 1 (MFA 代码 1) 和 MFA code 2 (MFA 代码 2) 中。然后选择 Re-sync authentication device。
3. 如有必要，请再次键入您的密码，然后选择登录。然后使用您的 MFA 设备完成登录。

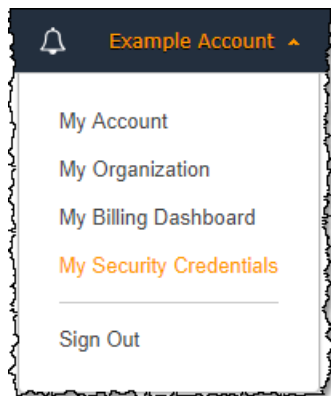
在登录后重新同步您的根用户 MFA 设备 (控制台)

1. 选择 Root user (根用户) 并输入您的 Amazon Web Services 账户电子邮件地址，以账户所有者身份登录 [IAM 控制台](#)。在下一页上，输入您的密码。

Note

如果显示了三个文本框，则您之前已使用 [IAM 用户](#) 凭证登录控制台。您的浏览器可能会记住此首选项，并在您每次尝试登录时打开此账户特定的登录页面。您无法使用 IAM 用户登录页面以账户所有者身份进行登录。如果您看到 [IAM 用户登录页面](#)，请选择该页面底部附近的 Sign in using root user email (使用根用户电子邮件登录)。这将返回主登录页面。在该页面中，您可以使用 Amazon Web Services 账户电子邮件地址和密码以根用户登录。

2. 在导航栏的右侧选择您的账户名称，然后选择 My Security Credentials (我的安全凭证)。如有必要，选择 Continue to Security Credentials。



3. 展开页面上的 Multi-factor authentication (MFA) (多重身份验证) 部分。
4. 在活动 MFA 设备的旁边，选择 Resync (重新同步)。
5. 在 Manage MFA device (管理 MFA 设备) 对话框中，依次将设备上按顺序生成的两个代码键入 MFA code 1 (MFA 代码 1) 和 MFA code 2 (MFA 代码 2) 中。然后选择 Continue (继续)。

Important

生成代码之后立即提交您的请求。如果生成代码后等待很长时间才提交请求，MFA 设备会成功与用户关联，但 MFA 设备无法同步。这是因为基于时间的一次性密码 (TOTP) 很快会过期。

重新同步虚拟和硬件 MFA 设备 (Amazon CLI)

您可以从 Amazon CLI 重新同步虚拟和硬件 MFA 设备。

重新同步 IAM 用户的虚拟或硬件 MFA 设备 (Amazon CLI)

在命令提示符处，发布 `aws iam resync-mfa-device` 命令：

- 虚拟 MFA 设备：将设备的 Amazon Resource Name (ARN) 指定为序列号。

```
aws iam resync-mfa-device --user-name Richard --serial-number  
arn:aws:iam::123456789012:mfa/RichardsMFA --authentication-code1 123456 --  
authentication-code2 987654
```

- 硬件 MFA 设备：将硬件设备的序列号指定为序列号。格式因供应商而异。例如，您可以从 Amazon 购买 gemalto 令牌。它的序列号通常是四个字母，后跟四个数字。

```
aws iam resync-mfa-device --user-name Richard --serial-number ABCD12345678 --  
authentication-code1 123456 --authentication-code2 987654
```

Important

生成代码之后立即提交您的请求。如果您生成代码，然后等待了很长时间才提交请求，则请求将失败，因为代码很快就会过期。

重新同步虚拟和硬件 MFA 设备 (Amazon API)

IAM 有一个执行同步的 API 调用。在这种情况下，建议您授予虚拟和硬件 MFA 设备用户访问此 API 调用的权限。然后，基于此 API 调用构建工具，这样您的用户即可随时根据需要重新同步其设备。

重新同步 IAM 用户的虚拟或硬件 MFA 设备 (Amazon API)

- 发送 `ResyncMFADevice` 请求。

停用 MFA 设备

如果您在使用多重身份验证 (MFA) 设备作为 IAM 用户登录时遇到问题，请与管理员联系以获取帮助。

作为管理员，您可以停用其他 IAM 用户的设备。此操作允许用户不使用 MFA 登录。如果 MFA 设备已更换或设备暂时不可用，您可以将此作为临时解决方案。但是，我们建议您尽快为用户启用新设备。要了解如何启用新 MFA 设备，请参阅 [the section called “启用 MFA 设备” \(p. 105\)](#)。

Note

如果您使用 API 或 Amazon CLI 从 Amazon 账户中删除用户，您必须停用或删除用户的 MFA 设备。您在删除用户的过程中执行此更改。有关删除用户的更多信息，请参阅 [管理 IAM 用户 \(p. 79\)](#)。

主题

- [停用 MFA 设备 \(控制台\) \(p. 126\)](#)
- [停用 MFA 设备 \(Amazon CLI\) \(p. 127\)](#)
- [停用 MFA 设备 \(Amazon API\) \(p. 127\)](#)

停用 MFA 设备 (控制台)

停用其他 IAM 用户的 MFA 设备 (控制台)

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 要停用用户的 MFA 设备，请选择要删除其 MFA 的用户的名称。
4. 选择 Security Credentials (安全凭证) 选项卡。在 Assigned MFA device (已分配 MFA 设备) 旁边，选择 Manage (管理)。
5. 在 Manage MFA device (管理 MFA 设备) 向导中，选择 Remove (删除)，然后再选择 Remove (删除)。

随即从 Amazon 中删除该设备。该设备将不能用于登录或验证请求，直到它重新激活并与 Amazon 用户或 Amazon Web Services 账户根目录相关联。

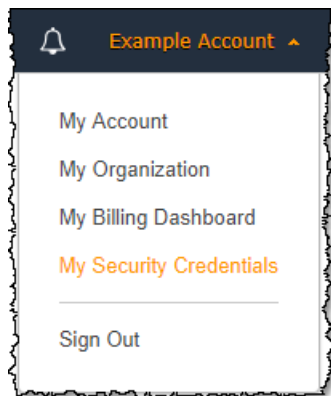
取消激活 Amazon Web Services 账户根用户的 MFA 设备 (控制台)

1. 选择 Root user (根用户) 并输入您的 Amazon Web Services 账户电子邮件地址，以账户所有者身份登录 [IAM 控制台](#)。在下一页上，输入您的密码。

Note

如果显示了三个文本框，则您之前已使用 [IAM 用户](#) 凭证登录控制台。您的浏览器可能会记住此首选项，并在您每次尝试登录时打开此账户特定的登录页面。您无法使用 IAM 用户登录页面以账户所有者身份进行登录。如果您看到 [IAM 用户登录页面](#)，请选择该页面底部附近的 Sign in using root user email (使用根用户电子邮件登录)。这将返回主登录页面。在该页面中，您可以使用 Amazon Web Services 账户电子邮件地址和密码以根用户登录。

2. 在导航栏的右侧选择您的账户名称，然后选择 My Security Credentials (我的安全凭证)。如有必要，选择 Continue to Security Credentials。



3. 展开 Multi-factor authentication (MFA) (多重验证) 部分。
4. 在要停用的 MFA 设备所在的行中，选择 Deactivate。

Amazon 账户的 MFA 设备已停用。检查与您的 Amazon 账户相关联的电子邮件邮箱中是否有来自 Amazon Web Services 的确认邮件。该电子邮件向您通知 Amazon Web Services 多重验证 (MFA) 已停用。该邮件来自以 @amazon.com 或 @aws.amazon.com 结尾的地址。

停用 MFA 设备 (Amazon CLI)

停用 IAM 用户的 MFA 设备 (Amazon CLI)

- 运行以下命令：`aws iam deactivate-mfa-device`

停用 MFA 设备 (Amazon API)

停用 IAM 用户的 MFA 设备 (Amazon API)

- 调用此操作：`DeactivateMFADevice`

若 MFA 设备遗失或停止工作，该怎么办？

若[虚拟 MFA 设备 \(p. 106\)](#)或[硬件 MFA 设备 \(p. 114\)](#)似乎运行正常，但您却无法使用它访问 Amazon 资源，则可能是与 Amazon 不同步所导致的。有关同步虚拟 MFA 设备或硬件 MFA 设备的信息，请参阅[重新同步虚拟和硬件 MFA 设备 \(p. 122\)](#)。[FIDO 安全密钥 \(p. 109\)](#)不同步。

如果您的 Amazon Web Services 账户根用户[多重身份验证 \(MFA\) 设备 \(p. 104\)](#)丢失、损坏或无法工作，您可以恢复对账户的访问权限。IAM 用户必须与管理员联系，才能将设备停用。

恢复根用户用户 MFA 设备

如果您的 Amazon Web Services 账户根用户[多重身份验证 \(MFA\) 设备 \(p. 104\)](#)丢失、损坏或无法正常工作，您可以使用替代的身份验证方法进行登录。这意味着，如果您无法使用 MFA 设备登录，您可以使用和账户一起注册的电子邮件和主要联系人电话号码验证您的身份来进行登录。

作为根用户使用替代的身份验证因素登录之前，您必须能够访问与您的账户关联的电子邮件地址和主要联系人电话号码。如果您无法访问此电子邮件或电话号码，则必须联系 [Amazon Web Services Support](#)。

使用替代的身份验证因素作为 Amazon Web Services 账户根用户登录

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

2. 在 Amazon Web Services Sign In With Authentication Device (亚马逊云科技使用身份验证设备登录) 页面上, 选择 Having problems with your authentication device? (身份验证设备出现问题?) Click here (点击此处)。

Note

可能会显示替代文本, 例如 Sign in using MFA (使用 MFA 登录)、Troubleshoot your authentication device (对身份验证设备进行故障排除) 或 Troubleshoot MFA (对 MFA 进行故障排除), 但其功能均相同。如果您无法使用替代身份验证因素验证您的账户电子邮件地址和主要联系人电话号码, 请与 [Amazon Web Services Support](#) 联系以停用您的 MFA 设备。

3. 如果需要, 请再次键入您的密码, 然后选择 Sign in。
4. 在 Sign In Using Alternative Factors of Authentication 部分中, 选择 Sign in using alternative factors。
5. 要通过验证您的电子邮件地址来验证账户的身份, 请选择 Send。
6. 检查与您的 Amazon 账户关联的电子邮件中是否有来自 Amazon Web Services (no-reply-aws@amazon.com) 的邮件。按照电子邮件中的指导进行操作。

如果您没有在账户中看到该电子邮件, 请检查垃圾邮件文件夹, 或者返回到浏览器并选择 Resend the email。

7. 在验证您的电子邮件地址后, 您可以继续验证您的账户的身份。要验证您的主要联系人电话号码, 请选择 Call me now (立即呼叫我)。
8. 接听 Amazon 打来的电话, 在听到提示时, 在手机键盘上输入从 Amazon 网站获得的 6 位数号码。

如果您没有接到 Amazon 打来的电话, 请选择 Sign in (登录) 以再次登录到控制台并重新开始。或者, 请参阅 [多重身份验证 \(MFA\) 设备丢失或无法使用](#) 联系支持人员以获取帮助。

9. 在验证您的电话号码后, 您可以通过选择 Sign in to the console 登录到您的账户。
10. 下一步取决于您使用的 MFA 的类型:
 - 对于虚拟 MFA 设备, 请从您的设备中删除账户。然后, 转至 [Amazon Security Credentials](#) 页面并删除旧的 MFA 虚拟设备实体, 然后再创建一个新的。
 - 对于 FIDO 安全密钥, 请转至 [Amazon Security Credentials](#) (Amazon 安全凭证) 页面并停用旧 FIDO 密钥然后再启用一个新密钥。
 - 对于硬件 MFA 设备, 请联系第三方提供商以帮助您修复或更换设备。您可以继续使用替代的身份验证因素登录, 直到您收到新设备为止。在您拥有新的硬件 MFA 设备后, 请转至 [Amazon 安全凭证](#) 页面并删除旧的 MFA 硬件设备实体然后再新建一个。

Note

不必将丢失或被盗的 MFA 设备替换为相同类型的设备。例如, 如果 FIDO 安全密钥损坏, 您订购了一个新的密钥, 则可以使用虚拟 MFA 或硬件 MFA 设备, 直到收到新 FIDO 安全密钥。

11. 如果 MFA 设备丢失或被盗, 还可[更改 Amazon 密码 \(p. 87\)](#), 以防攻击者盗走您的身份验证设备以及同时可能拥有您的当前密码。

恢复 IAM 用户 MFA 设备

如果您是 IAM 用户, 且您的设备丢失或停止工作, 则无法自行恢复。您必须与管理员联系, 才能将设备停用。然后, 您可以启用新设备。

作为 IAM 用户获取有关 MFA 设备的帮助

1. 请联系 Amazon 管理员或其他为您提供 IAM 用户的用户名和密码的相关人员。管理员必须停用 MFA 设备 (如[停用 MFA 设备 \(p. 126\)](#)中所述), 以便您能够登录。
2. 下一步取决于您使用的 MFA 的类型:

- 对于虚拟 MFA 设备，请从您的设备中删除账户。然后，启用虚拟设备，如[启用虚拟 Multi-Factor Authentication \(MFA\) 设备 \(控制台\)](#) (p. 106) 中所述。
- 对于 FIDO 安全密钥，请联系第三方提供商帮助您更换设备。当您收到新的 FIDO 安全密钥时，请将其启用，如[启用 FIDO 安全密钥 \(控制台\)](#) (p. 109) 中所述。
- 对于硬件 MFA 设备，请联系第三方提供商以帮助您修复或更换设备。在您拥有新的物理 MFA 设备后，请启用设备，如[启用硬件 MFA 设备 \(控制台\)](#) (p. 114) 中所述。

Note

不必将丢失或被盗的 MFA 设备替换为相同类型的设备。例如，如果 FIDO 安全密钥损坏，您订购了一个新的密钥，则可以使用虚拟 MFA 或硬件 MFA 设备，直到收到新 FIDO 安全密钥。

3. 如果您的 MFA 设备丢失或被盗，还请[更改密码](#) (p. 95)，以防攻击者盗走您的身份验证设备及还可能拥有您当前的密码。

配置受 MFA 保护的 API 访问

利用 IAM policy，可以指定用户可调用哪些 API 操作。在一些情况下，您可能希望先要求用户通过 Amazon 多重身份验证 (MFA) 进行身份验证，然后才允许他们执行特别敏感的操作，从而提高安全性。

例如，您可能拥有允许用户执行 Amazon EC2 RunInstances、DescribeInstances 和 StopInstances 操作的策略。但您可能希望限制破坏性操作（如 TerminateInstances），并确保用户只能在使用 Amazon MFA 设备进行身份验证后执行该操作。

主题

- [概览](#) (p. 129)
- [方案：跨账户委派的 MFA 保护](#) (p. 131)
- [方案：当前账户中 API 操作访问的 MFA 保护](#) (p. 132)
- [方案：拥有基于资源的策略的资源的 MFA 保护](#) (p. 133)

概览

向 API 操作添加 MFA 保护包括以下任务：

1. 管理员为每个用户配置 Amazon MFA 设备，这些用户需要发出要求 MFA 身份验证的 API 请求。为[Amazon 中的用户启用 MFA 设备](#) (p. 105) 中介绍了此过程。
2. 管理员为用户创建包含 Condition 元素的策略，以检查用户是否已使用 Amazon MFA 设备进行身份验证。
3. 用户调用支持 MFA 参数 Amazon STSAssumeRole 或 GetSessionToken 的 API 操作之一，具体取决于 MFA 保护的方案，如稍后所述。调用中包含与用户关联的设备的设备标识符，以及该设备生成的基于时间的一次性密码 (TOTP)。在任一情况下，用户都会取回稍后用来向 Amazon 发出其他请求的临时安全凭证。

Note

仅在服务支持临时安全凭证时，该服务的 API 操作的 MFA 保护才可用。有关这些服务的列表，请参阅[使用临时安全凭证访问 Amazon](#)。

如果授权失败，Amazon 将返回“访问被拒绝”错误消息（与任何未授权的访问一样）。由于采用受 MFA 保护的 API 策略，因此，如果用户在未获得有效的 MFA 身份验证的情况下试图调用 API 操作，则 Amazon 将拒绝用户访问策略中指定的 API 操作。如果 API 操作请求的时间戳在策略中指定的允许范围之外，也会拒绝操作。用户必须使用 MFA 代码和设备序列号请求新的临时安全凭证，通过 MFA 重新进行身份验证。

带 MFA 条件的 IAM policy

带 MFA 条件的策略可附加到以下项：

- 对于 IAM 用户或组：
- Amazon S3 存储桶、Amazon SQS 队列或 Amazon SNS 主题等资源
- 可由用户担任的 IAM 角色的信任策略

可使用策略中的 MFA 条件检查以下属性：

- Existence (存在性) - 如果只是验证用户是否已使用 MFA 进行身份验证，请检查 `aws:MultiFactorAuthPresent` 密钥在 Bool 条件中是否为 True。仅当用户使用短期凭证验证时，才会有该键。长期凭证，例如访问密钥，不包括此键。
- Duration (持续时间) - 如果您只希望在 MFA 身份验证后的指定时间内授予访问权限，请使用数值条件类型将 `aws:MultiFactorAuthAge` 密钥的有效期与某个值（如 3600 秒）进行比较。请注意，如果未使用 MFA，则 `aws:MultiFactorAuthAge` 键不会显示。

以下示例说明了 IAM 角色的信任策略，该策略包含一个 MFA 条件，用于测试是否存在 MFA 身份验证。利用此策略，Principal 元素中指定的 Amazon 账户中的用户（将 `ACCOUNT-B-ID` 替换为有效的 Amazon 账户 ID）能够担任此策略附加到的角色。但是，此类用户仅在已使用 MFA 进行身份验证的情况下才能够担任角色。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "ACCOUNT-B-ID"},
    "Action": "sts:AssumeRole",
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }
}
```

有关 MFA 的条件类型的更多信息，请参阅[Amazon 全局条件上下文密钥 \(p. 998\)](#)、[数字条件运算符 \(p. 953\)](#)和[用于检查条件键是否存在的条件运算符 \(p. 958\)](#)。

在 GetSessionToken 和 AssumeRole 之间选择

Amazon STS 提供了两个 API 操作，可供用户传递 MFA 信息：GetSessionToken 和 AssumeRole。用户调用哪个 API 操作来获取临时安全凭证，取决于采用以下哪个方案。

将 **GetSessionToken** 用于以下方案：

- 调用访问 Amazon 账户中的资源的 API 操作，该账户与发出请求的 IAM 用户相同。请注意，来自 GetSessionToken 请求的临时凭证仅在您将 MFA 信息包含在凭证请求中时才能访问 IAM 和 Amazon STS API 操作。由于 GetSessionToken 返回的临时凭证包含 MFA 信息，因此您可以检查由该凭证发出的单个 API 操作中的 MFA。
- 访问受基于资源且包含 MFA 条件的策略所保护的资源。

GetSessionToken 操作旨在使用 MFA 验证用户身份。您不能使用策略来控制身份验证操作。

将 **AssumeRole** 用于以下方案：

- 调用访问相同或不同 Amazon 账户中的资源的 API 操作。API 调用可包含任何 IAM 或 Amazon STS API。请注意，要保护访问，您可以在用户担任角色时强制执行 MFA。由 AssumeRole 返回的临时凭证未将 MFA 信息包含在上下文中，因此您无法检查单个 API 操作的 MFA。这就是您必须使用 GetSessionToken 限制对受基于资源的策略保护的资源的访问的原因。

本文档稍后将提供有关如何实现这些方案的详细信息。

有关受 MFA 保护的 API 访问的要点

了解 API 操作的 MFA 保护的以下几个方面非常重要：

- MFA 保护仅通过使用临时安全凭证提供，而该凭证必须使用 `AssumeRole` 或 `GetSessionToken` 获取。
- 无法将受 MFA 保护的 API 访问与 Amazon Web Services 账户 根用户凭证配合使用。
- 无法将受 MFA 保护的 API 访问与 U2F 安全密钥配合使用。
- 将 MFA 设备与 Amazon 服务配合使用时，无法向联合身份用户分配这些设备；因此，这些用户无法访问受 MFA 控制的 Amazon 资源。(请参阅下一个要点。)
- 返回临时凭证的其他 Amazon STS API 操作不支持 MFA。对于 `AssumeRoleWithWebIdentity` 和 `AssumeRoleWithSAML`，用户通过外部提供程序进行身份验证，并且 Amazon 无法确定该提供程序是否需要 MFA。对于 `GetFederationToken`，MFA 不一定要与特定用户相关联。
- 同样，长期凭证 (IAM 用户访问密钥和根用户访问密钥) 无法用于受 MFA 保护的 API 访问，因为它们不会过期。
- 此外，可以在没有 MFA 信息的情况下调用 `AssumeRole` 和 `GetSessionToken`。在此情况下，发起人将取回临时安全凭证，但这些临时凭证的会话信息不指示使用 MFA 进行身份验证的用户。
- 要建立 API 操作的 MFA 保护，可将 MFA 条件添加到策略。策略必须包含 `aws:MultiFactorAuthPresent` 条件键以强制使用 MFA。对于跨账户委派，角色的信任策略必须包含条件键。
- 如果您允许其他 Amazon 账户访问您账户中的资源，则您的资源安全性取决于受信任账户 (另一个账户，而不是您的账户) 的配置。即使在您强制实施多重身份验证时，也是如此。有权创建虚拟 MFA 设备的受信任账户中的任何标识都可以构建 MFA 索赔，以满足您角色的信任策略的该部分。在允许其他账户的成员访问必须进行多重身份验证的 Amazon 资源之前，您应确保受信任账户的所有者遵循安全最佳实践。例如，受信任账户应仅允许特定的受信任身份访问敏感 API 操作，例如 MFA 设备管理 API 操作。
- 如果策略包含 MFA 条件，则在以下情况下，将拒绝请求：用户未进行 MFA 身份验证或用户提供了无效的 MFA 设备标识符或无效的 TOTP。

方案：跨账户委派的 MFA 保护

在此方案中，您希望将访问权委托给另一账户中的 IAM 用户，但前提是该用户已使用 Amazon MFA 设备进行身份验证。有关跨账户委派的更多信息，请参阅[角色术语和概念 \(p. 158\)](#)。

假设您有一个账户 A (拥有待访问资源的信任账户) 以及 IAM 用户 Anaya，她拥有管理员权限。她希望向账户 B (受信任账户) 中的用户 Richard 授予访问权，但希望确保 Richard 在担任该角色之前已使用 MFA 进行身份验证。

1. 在信任账户 A 中，Anaya 创建一个名为 `CrossAccountRole` 的 IAM 角色，并将该角色的信任策略中的主体设置为账户 B 的账户 ID。此信任策略授予对 Amazon STS `AssumeRole` 操作的权限。Anaya 还向信任策略添加 MFA 条件，如以下示例中所示。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "ACCOUNT-B-ID"},
    "Action": "sts:AssumeRole",
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }
}
```

2. Anaya 在该角色中添加一个权限策略，以指定允许该角色执行的操作。具有 MFA 保护的角色权限策略与任何其他角色权限策略没有差别。以下示例说明了 Anaya 添加到角色的策略；该策略允许担任该角色的用

户对账户 A 中的表 `Books` 执行任何 Amazon DynamoDB 操作。此策略还允许 `dynamodb:ListTables` 操作，此操作是在控制台中执行操作所必需的。

Note

该权限策略不包含 MFA 条件。了解 MFA 身份验证仅用于确定用户是否可以担任此角色很重要。在用户担任此角色后，将不会进行进一步的 MFA 检查。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "TableActions",
      "Effect": "Allow",
      "Action": "dynamodb:*",
      "Resource": "arn:aws:dynamodb:*:ACCOUNT-A-ID:table/Books"
    },
    {
      "Sid": "ListTables",
      "Effect": "Allow",
      "Action": "dynamodb:ListTables",
      "Resource": "*"
    }
  ]
}
```

3. 在受信任账户 B 中，管理员确保已使用 Amazon MFA 设备配置 IAM 用户 Richard，并且该用户知道设备的 ID。设备 ID 是序列号（如果是硬件 MFA 设备）或设备的 ARN（如果是虚拟 MFA 设备）。
4. 在账户 B 中，管理员将以下策略附加到用户 Richard（或该用户所在的组），该策略允许该用户调用 `AssumeRole` 操作。资源被设置到 Anaya 在第 1 步中创建的角色 ARN。注意，该策略不包含 MFA 条件。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sts:AssumeRole"],
    "Resource": ["arn:aws:iam:*:ACCOUNT-A-ID:role/CrossAccountRole"]
  }]
}
```

5. 在账户 B 中，Richard（或 Richard 正在运行的应用程序）调用 `AssumeRole`。API 调用包含要担任角色的 ARN (`arn:aws:iam:*:ACCOUNT-A-ID:role/CrossAccountRole`)、MFA 设备的 ID 和 Richard 从其设备中获取的当前 TOTP。

当 Richard 调用 `AssumeRole` 时，Amazon 将确定他是否拥有有效的凭证，包括 MFA 要求。如果是这样的话，Richard 将成功担任角色，并且可在使用角色的临时凭证时对账户 A 中名为 `Books` 的表执行任何 DynamoDB 操作。

有关调用 `AssumeRole` 的程序的示例，请参阅[使用 MFA 身份验证调用 `AssumeRole` \(p. 135\)](#)。

方案：当前账户中 API 操作访问的 MFA 保护

在此方案中，您应确保仅当您 Amazon 账户中的用户已使用 Amazon MFA 设备进行身份验证后才能访问敏感 API 操作。

假设您拥有账户 A，其中包含一组需要使用 EC2 实例的开发人员。普通开发人员可以使用实例，但他们未获得 `ec2:StopInstances` 或 `ec2:TerminateInstances` 操作的权限。您希望仅允许几个受信任用户执行这些“破坏性”特权操作，因此您将 MFA 保护添加到允许这些敏感 Amazon EC2 操作的策略中。

在此方案中，用户 Sofia 是受信任用户之一。用户 Anaya 是账户 A 中的管理员。

1. Anaya 确保已使用 Amazon MFA 设备配置 Sofia，并且 Sofia 知道该设备的 ID。设备 ID 是序列号（如果是硬件 MFA 设备）或设备的 ARN（如果是虚拟 MFA 设备）。
2. Anaya 创建一个名为 EC2-Admins 的组并将用户 Sofia 添加到该组中。
3. Anaya 将以下策略附加到 EC2-Admins 组。此策略授予用户调用 Amazon EC2 StopInstances 和 TerminateInstances 操作的权限，但前提是该用户已使用 MFA 进行身份验证。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "ec2:StopInstances",
      "ec2:TerminateInstances"
    ],
    "Resource": ["*"],
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }]
}
```

4. Note

要使此策略生效，用户必须先注销，然后重新登录。

如果用户 Sofia 需要停止或终止 Amazon EC2 实例，则她（或她正在运行的应用程序）可调用 GetSessionToken。此 API 操作传递 MFA 设备的 ID 和 Sofia 从其设备获取的当前 TOTP。

5. 用户 Sofia（或 Sofia 正在使用的应用程序）使用由 GetSessionToken 提供的临时凭证来调用 Amazon EC2 StopInstances 或 TerminateInstances 操作。

有关调用 GetSessionToken 的程序的示例，请参阅本文档后面的 [使用 MFA 身份验证调用 GetSessionToken \(p. 134\)](#)。

方案：拥有基于资源的策略的资源的 MFA 保护

在此方案中，您是 S3 存储桶、SQS 队列或 SNS 主题的所有者。您希望确保访问资源的任何 Amazon 账户中的任何用户都已使用 Amazon MFA 设备进行身份验证。

此方案介绍了一种提供跨账户 MFA 保护的方法，无需用户先担任角色。在此情况下，用户可在满足以下三个条件时访问资源：用户已使用 MFA 进行身份验证、能够从 GetSessionToken 获取临时安全凭证且在受资源策略信任的账户中。

假设您在账户 A 中并创建一个 S3 存储桶。您希望向几个不同 Amazon 账户中的用户授予对此存储桶的访问权，但前提是这些用户已使用 MFA 进行身份验证。

在此方案中，用户 Anaya 是账户 A 中的管理员。用户 Nikhil 是账户 C 中的 IAM 用户。

1. 在账户 A 中，Anaya 创建一个名为 Account-A-bucket 的存储桶。
2. Anaya 向该存储桶添加存储桶策略。该策略允许账户 A、账户 B 或账户 C 中的所有用户执行存储桶中的 Amazon S3 PutObject 和 DeleteObject 操作。该策略包含 MFA 条件。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"AWS": [
      "ACCOUNT-A-ID",
      "ACCOUNT-B-ID",
      "ACCOUNT-C-ID"
    ]},
    "Action": [
```



```
"s3:PutObject",
"s3:DeleteObject"
],
"Resource": ["arn:aws:s3:::ACCOUNT-A-BUCKET-NAME/*"],
"Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
}]
}
```

Note

Amazon S3 (仅) 针对根账户访问提供“MFA 删除”功能。在您设置存储桶的版本化状态时，可启用 Amazon S3 MFA Delete 功能。Amazon S3 MFA Delete 功能不适用于 IAM 用户，在管理时独立于 MFA 保护的 API 访问。即使 IAM 用户具有删除存储桶的权限，但在启用 Amazon S3 MFA Delete 功能时，也无法执行删除。有关 Amazon S3 MFA Delete 功能的更多信息，请参阅 [MFA Delete](#)。

3. 在账户 C 中，管理员确保已使用 Amazon MFA 设备配置用户 Nikhil，并且该用户知道设备的 ID。设备 ID 是序列号（如果是硬件 MFA 设备）或设备的 ARN（如果是虚拟 MFA 设备）。
4. 在账户 C 中，Nikhil（或他正在运行的应用程序）将调用 `GetSessionToken`。此调用包括 MFA 设备的 ID 或 ARN 以及 Nikhil 从其设备中获取的当前 TOTP。
5. Nikhil（或他正在使用的应用程序）使用 `GetSessionToken` 返回的临时凭证调用 Amazon S3 `PutObject` 操作以将文件上传到 `Account-A-bucket`。

有关调用 `GetSessionToken` 的程序的示例，请参阅本文档后面的 [使用 MFA 身份验证调用 `GetSessionToken`](#) (p. 134)。

Note

在此情况下，`AssumeRole` 返回的临时凭证将不可用。尽管用户可以提供 MFA 信息来担任角色，但 `AssumeRole` 返回的临时凭证不包含 MFA 信息。需要此信息才能满足策略中的 MFA 条件。

示例代码：使用多重验证请求凭证

以下示例说明如何调用 `GetSessionToken` 和 `AssumeRole` 操作并传递 MFA 身份验证参数。无需任何权限即可调用 `GetSessionToken`，但您必须拥有允许您调用 `AssumeRole` 的策略。随后，返回的凭证将用于列出账户中的所有 S3 存储桶。

使用 MFA 身份验证调用 `GetSessionToken`

以下示例说明了如何调用 `GetSessionToken` 并传递 MFA 身份验证信息。然后使用 `GetSessionToken` 操作返回的临时安全凭证来列出账户中的所有 S3 存储桶。

附加到运行此代码的用户（或用户所在的组）的策略提供了返回的临时凭证的权限。对于此示例代码，该策略必须向用户授予请求 Amazon S3 `ListBuckets` 操作的权限。

以下代码示例演示了如何获取具有 Amazon STS 的会话令牌并将其用于执行需要 MFA 令牌的服务操作。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

通过传递 MFA 令牌获取会话令牌，然后使用令牌列出账户的 Amazon S3 存储桶。

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
      sts_client):
```

```
"""
Gets a session token with MFA credentials and uses the temporary session
credentials to list Amazon S3 buckets.

Requires an MFA device serial number and token.

:param mfa_serial_number: The serial number of the MFA device. For a virtual
MFA device, this is an Amazon Resource Name (ARN).
:param mfa_totp: A time-based, one-time password issued by the MFA device.
:param sts_client: A Boto3 STS instance that has permission to assume the role.
"""
if mfa_serial_number is not None:
    response = sts_client.get_session_token(
        SerialNumber=mfa_serial_number, TokenCode=mfa_totp)
else:
    response = sts_client.get_session_token()
temp_credentials = response['Credentials']

s3_resource = boto3.resource(
    's3',
    aws_access_key_id=temp_credentials['AccessKeyId'],
    aws_secret_access_key=temp_credentials['SecretAccessKey'],
    aws_session_token=temp_credentials['SessionToken'])

print(f"Buckets for the account:")
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [GetSessionToken](#)。

使用 MFA 身份验证调用 AssumeRole

下面的示例说明了如何调用 AssumeRole 并传递 MFA 身份验证信息。然后使用 AssumeRole 返回的临时安全证书列出账户中的所有 Amazon S3 存储桶。

有关此方案的更多信息，请参阅[方案：跨账户委派的 MFA 保护 \(p. 131\)](#)。

以下代码示例演示了如何代入具有 Amazon STS 的角色。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon STS service client object.
const stsClient = new STSClient({ region: REGION });
export { stsClient };
```

代入 IAM 角色。

```
// Import required AWS SDK clients and commands for Node.js
import { stsClient } from "../libs/stsClient.js";
import {
  AssumeRoleCommand,
  GetCallerIdentityCommand,
} from "@aws-sdk/client-sts";

// Set the parameters
export const params = {
  RoleArn: "ARN_OF_ROLE_TO_ASSUME", //ARN_OF_ROLE_TO_ASSUME
  RoleSessionName: "session1",
  DurationSeconds: 900,
};

export const run = async () => {
  try {
    //Assume Role
    const data = await stsClient.send(new AssumeRoleCommand(params));
    return data;
    const rolecreds = {
      accessKeyId: data.Credentials.AccessKeyId,
      secretAccessKey: data.Credentials.SecretAccessKey,
      sessionToken: data.Credentials.SessionToken,
    };
    //Get Amazon Resource Name (ARN) of current identity
    try {
      const stsParams = { credentials: rolecreds };
      const stsClient = new STSClient(stsParams);
      const results = await stsClient.send(
        new GetCallerIdentityCommand(rolecreds)
      );
      console.log("Success", results);
    } catch (err) {
      console.log(err, err.stack);
    }
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [AssumeRole](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
const AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

var roleToAssume = {RoleArn: 'arn:aws:iam::123456789012:role/RoleName',
  RoleSessionName: 'session1',
  DurationSeconds: 900,};

var roleCreds;

// Create the STS service object
var sts = new AWS.STS({apiVersion: '2011-06-15'});

//Assume Role
```

```
sts.assumeRole(roleToAssume, function(err, data) {
  if (err) console.log(err, err.stack);
  else{
    roleCreds = {accessKeyId: data.Credentials.AccessKeyId,
                  secretAccessKey: data.Credentials.SecretAccessKey,
                  sessionToken: data.Credentials.SessionToken};
    stsGetCallerIdentity(roleCreds);
  }
});

//Get Arn of current identity
function stsGetCallerIdentity(creds) {
  var stsParams = {credentials: creds };
  // Create STS service object
  var sts = new AWS.STS(stsParams);

  sts.getCallerIdentity({}, function(err, data) {
    if (err) {
      console.log(err, err.stack);
    }
    else {
      console.log(data.Arn);
    }
  });
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [AssumeRole](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

代入需要 MFA 令牌的 IAM 角色并使用临时凭证列出该账户的 Amazon S3 存储桶。

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
    account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
    MFA
                            device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp)
    temp_credentials = response['Credentials']
```

```
print(f"Assumed role {assume_role_arn} and got temporary credentials.")

s3_resource = boto3.resource(
    's3',
    aws_access_key_id=temp_credentials['AccessKeyId'],
    aws_secret_access_key=temp_credentials['SecretAccessKey'],
    aws_session_token=temp_credentials['SessionToken'])

print(f"Listing buckets for the assumed role's account:")
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [AssumeRole](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
# This is separated into a factory function so that it can be mocked for unit
testing.
#
# @param key_id [String] The ID of the access key used by the STS client.
# @param key_secret [String] The secret part of the access key used by the STS
client.
def create_sts_client(key_id, key_secret)
  Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
end

# Gets temporary credentials that can be used to assume a role.
#
# @param role_arn [String] The ARN of the role that is assumed when these
credentials
#
# are used.
# @param sts_client [Aws::STS::Client] An AWS STS client.
# @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
def assume_role(role_arn, sts_client)
  credentials = Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: "create-use-assume-role-scenario"
  )
  puts("Assumed role '#{role_arn}', got temporary credentials.")
  credentials
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [AssumeRole](#)。

查找未使用的凭证

为提高您的 Amazon 账户的安全性，请删除不需要的 IAM 用户凭证（即密码和访问密钥）。例如，当用户离开您的组织或不再需要访问 Amazon 时，请找到用户所使用的凭证并确保这些凭证不再有效。理想情况下，

如果不再需要凭证，可将其删除。您始终可在将来需要这些凭证时创建它们。您至少应该更改密码或停用访问密钥，使以前的用户不再具有访问权限。

当然，未使用 这个词可以有多种理解，但通常指在指定时段内没用过的凭证。

查找未使用的密码

您可以使用 Amazon Web Services Management Console 查看用户的密码使用信息。如果您有大量的用户，则您可以使用控制台下载包含有关每个用户上次使用其控制台密码的时间信息的凭证报告。您还可以从 Amazon CLI 或 IAM API 访问信息。

查找未使用的密码 (控制台)

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 如有必要，请将 Console last sign-in 列添加到用户表中：
 - a. 在最右侧的表上方，选择设置图标 (⚙️)。
 - b. 在 Manage Columns 中，选择 Console last sign-in。
 - c. 选择 Close 返回到用户列表。
4. Console last sign-in (控制台上次登录时间) 列显示用户上次通过控制台登录 Amazon 的日期。您可以利用该信息查找超过指定时间段未使用密码登录的用户。对于从未使用密码登录的用户，该列显示 Never。None 表示没有密码的用户。最近未使用的密码可能适合做删除处理。

Important

由于服务问题，上次使用密码的数据不包括从 2018 年 5 月 3 日 22:50 (太平洋时间) 到 2018 年 5 月 23 日 14:08 (太平洋时间) 之间的密码使用。这会影响 IAM 控制台中显示的 [上次登录日期](#) 和 [IAM 凭证报告](#) 中以及 [GetUser API 操作](#) 返回的上次使用密码的日期。如果用户在受影响的时间内登录，则返回的上次使用密码的日期是用户在 2018 年 5 月 3 日之前最后一次登录的日期。对于在 2018 年 5 月 23 日 14:08 (太平洋时间) 之后登录的用户，所返回的上次使用密码的日期准确无误。

如果您使用上次使用密码的信息来识别未使用的凭证以将其删除 (如删除在过去 90 天内未登录过 Amazon 的用户)，建议您调整评估时段以纳入 2018 年 5 月 23 日之后的日期。或者，如果您的用户使用访问密钥以编程方式访问 Amazon，则您可以参阅上次使用访问密钥的信息，因为其中的所有日期都准确无误。

通过下载凭证报告查找未使用的密码 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择凭证报告。
3. 选择 Download Report 以下载名为 status_reports_<date>T<time>.csv 的逗号分隔值 (CSV) 文件。第五列包含具有日期或下列项之一的 password_last_used 列：
 - N/A - 根本未获得密码的用户。
 - no_information - 自 2014 年 10 月 20 日 IAM 开始跟踪密码使用期限以来未使用其密码的用户。

查找未使用的密码 (Amazon CLI)

运行以下命令查找未使用的密码：

- `aws iam list-users` 返回用户列表，每个用户均有一个 PasswordLastUsed 值。如果缺少此值，则用户未获得密码或自 IAM 在 2014 年 10 月 20 日开始跟踪密码使用期限以来未使用密码。

查找未使用的密码 (Amazon API)

调用以下操作查找未使用的密码：

- `ListUsers` 返回用户集合，每个用户均有一个 `<PasswordLastUsed>` 值。如果缺少此值，则用户未获得密码或自 IAM 在 2014 年 10 月 20 日开始跟踪密码使用期限以来未使用密码。

有关用于下载凭证报告的命令的信息，请参阅[获取凭证报告 \(Amazon CLI\)](#) (p. 144)。

查找未使用的访问密钥

您可以使用 Amazon Web Services Management Console 查看用户的访问密钥使用信息。如果您有大量的用户，则您可以使用控制台下载凭证报告，以查找每个用户上次使用其访问密钥的时间。您还可以从 Amazon CLI 或 IAM API 访问信息。

查找未使用的访问密钥 (控制台)

1. 登录 Amazon Web Services Management Console，单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中，选择 Users (用户)。
3. 如有必要，将 Access key last used 列添加到用户表中：
 - a. 在最右侧的表上方，选择设置图标 (⚙️)。
 - b. 在 Manage Columns 中，选择 Access key last used。
 - c. 选择 Close 返回到用户列表。
4. Access key last used (访问密钥上次使用) 列显示用户上次以编程方式访问 Amazon 以来经过的天数。您可以利用该信息查找超过指定时间段未使用访问密钥的用户。对于没有访问密钥的用户，该列显示 None。最近未使用的访问密钥可能适合做删除处理。

通过下载凭证报告来查找未使用的访问密钥 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择凭证报告。
3. 选择 Download Report 以下载名为 `status_reports_<date>T<time>.csv` 的逗号分隔值 (CSV) 文件。列 11 至 13 包含访问密钥 1 的上次使用日期、区域和服务信息。列 16 至列 18 包含访问密钥 2 的相同信息。如果用户没有访问密钥或用户自 IAM 在 2015 年 4 月 22 日开始跟踪访问密钥使用期限以来未使用访问密钥，则值为 N/A。

查找未使用的访问密钥 (Amazon CLI)

运行以下命令查找未使用的访问密钥：

- `aws iam list-access-keys` 返回有关用户的访问密钥的信息，包括 AccessKeyID。
- `aws iam get-access-key-last-used` 采用访问密钥 ID 并返回输出，包括 LastUsedDate、Region (上次在其中使用访问密钥) 和上一次请求的服务的 ServiceName。如果缺少 LastUsedDate，则自 IAM 在 2015 年 4 月 22 日开始跟踪访问密钥使用期限以来未使用访问密钥。

查找未使用的访问密钥 (Amazon API)

调用以下操作查找未使用的访问密钥：

- `ListAccessKeys` 返回与指定用户关联的访问密钥的 AccessKeyID 值的列表。

- `GetAccessKeyLastUsed` 采用访问密钥 ID 并返回值的集合。包括 `LastUsedDate`、`Region` (上次在其中使用访问密钥) 和上次请求的服务的 `ServiceName`。如果缺少此值，则用户未获得访问密钥或自 IAM 在 2015 年 4 月 22 日开始跟踪访问密钥使用期间以来未使用访问密钥。

有关用于下载凭证报告的命令的信息，请参阅[获取凭证报告 \(Amazon CLI\)](#) (p. 144)。

获取您 Amazon 账户的凭证报告

您可以生成和下载列出您账户中所有用户及其各个凭证状态 (包括密码、访问密钥和 MFA 设备) 的凭证报告。您可以从 Amazon Web Services Management Console、[Amazon 开发工具包](#)和[命令行工具](#)或 IAM API 获取凭证报告。

您可以使用凭证报告帮助您进行审核和合规性工作。您可以使用报告审核凭证生命周期要求的效果，如密码和访问密钥轮换。您可以向外部审核员提供报告，或向审核员授予权限，以便他或她可以直接下载报告。

您可以按每四小时一次的频率生成凭证报告。请求报告时，IAM 先检查在过去四小时内是否生成了 Amazon 账户的报告。如果是，则下载最新的报告。如果账户的最新报告是四小时前生成的，或者账户以前没有报告，则 IAM 将生成并下载新报告。

主题

- [所需权限](#) (p. 141)
- [了解报告格式](#) (p. 141)
- [获取凭证报告 \(控制台\)](#) (p. 144)
- [获取凭证报告 \(Amazon CLI\)](#) (p. 144)
- [获取凭证报告 \(Amazon API\)](#) (p. 144)

所需权限

创建和下载报告需要以下权限：

- 创建凭证报告：`iam:GenerateCredentialReport`
- 下载报告：`iam:GetCredentialReport`

了解报告格式

凭证报告采用逗号分隔值 (CSV) 文件格式。您可以使用常用电子表格软件打开 CSV 文件以执行分析，也可以构建应用程序以编程方式使用 CSV 文件并执行自定义分析。

CSV 文件包含以下列：

`user`

用户的友好名称。

`arn`

用户的 Amazon Resource Name (ARN)。有关 ARN 的更多信息，请参阅[IAM ARN](#) (p. 896)。

`user_creation_time`

创建用户的日期和时间，[ISO 8601 日期时间格式](#)。

`password_enabled`

如果用户有密码，则该值为 `TRUE`。否则为 `FALSE`。Amazon Web Services 账户根用户的值始终为 `not_supported`。

password_last_used

Amazon Web Services 账户根用户或 IAM 用户的密码上次用于登录 Amazon 网站的日期和时间，采用 [ISO 8601 date-time format](#) (ISO 8601 日期时间格式)。捕获用户上次登录时间的 Amazon 网站是 Amazon Web Services Management Console、Amazon 论坛和 Amazon Marketplace。如果密码在 5 分钟的时间范围内多次使用，则仅在此字段中记录第一次使用。

- 此字段的值在以下情况下为 no_information：
 - 用户的密码从未使用过。
 - 没有与密码关联的登录数据，例如，当用户的密码在 IAM 于 2014 年 10 月 20 日开始跟踪此信息后尚未使用时。
- 如果用户没有密码，此字段中的值为 N/A (不适用)。

Important

由于服务问题，上次使用密码的数据不包括从 2018 年 5 月 3 日 22:50 (太平洋时间) 到 2018 年 5 月 23 日 14:08 (太平洋时间) 之间的密码使用。这会影响 IAM 控制台中显示的 [上次登录](#) 日期和 [IAM 凭证报告](#) 中以及 [GetUser API 操作](#) 返回的上次使用密码的日期。如果用户在受影响的时间内登录，则返回的上次使用密码的日期是用户在 2018 年 5 月 3 日之前最后一次登录的日期。对于在 2018 年 5 月 23 日 14:08 (太平洋时间) 之后登录的用户，所返回的上次使用密码的日期准确无误。

如果您使用上次使用密码的信息来识别未使用的凭证以将其删除 (如删除在过去 90 天内未登录过 Amazon 的用户)，建议您调整评估时段以纳入 2018 年 5 月 23 日之后的日期。或者，如果您的用户使用访问密钥以编程方式访问 Amazon，则您可以参阅上次使用访问密钥的信息，因为其中的所有日期都准确无误。

password_last_changed

上次设置用户密码的日期和时间，[ISO 8601 日期时间格式](#)。如果用户没有密码，则此字段中的值为 N/A (不适用)。Amazon 账户 (根) 的值始终是 not_supported。

password_next_rotation

如果账户的 [密码策略](#) 要求密码轮换，则此字段包含用户需要设置新密码的日期和时间，[ISO 8601 日期时间格式](#)。Amazon 账户 (根) 的值始终是 not_supported。

mfa_active

如果对用户启用了 [多重验证 \(p. 104\)](#) (MFA) 设备，则此值为 TRUE。否则为 FALSE。

access_key_1_active

如果用户有访问密钥且访问密钥状态为 Active，则该值为 TRUE。否则为 FALSE。

access_key_1_last_rotated

创建或上次更改用户访问密钥的日期和时间，[ISO 8601 日期时间格式](#)。如果用户没有活动的访问密钥，则此字段中的值为 N/A (不适用)。

access_key_1_last_used_date

最近使用用户的访问密钥签署 Amazon API 请求的日期和时间 (采用 [ISO 8601 日期时间格式](#))。如果访问密钥在 15 分钟的时间范围内多次使用，则仅在此字段中记录第一次使用。

此字段的值在这些情况下为 N/A (不适用)：

- 用户没有访问密钥。
- 访问密钥从未使用过。
- 访问密钥在 IAM 于 2015 年 4 月 22 日开始跟踪此信息后尚未使用。

access_key_1_last_used_region

最近在其中使用访问密钥的 [Amazon 区域](#)。如果访问密钥在 15 分钟的时间范围内多次使用，则仅在此字段中记录第一次使用。

此字段的值在这些情况下为 N/A (不适用)：

- 用户没有访问密钥。
- 访问密钥从未使用过。
- 上次使用访问密钥是在 IAM 于 2015 年 4 月 22 日开始跟踪此信息之前。
- 上次使用的服务不是区域特定的服务，如 Amazon S3。

`access_key_1_last_used_service`

最近使用访问密钥访问的 Amazon 服务。此字段的值使用服务的命名空间，例如 `s3` 表示 Amazon S3，`ec2` 表示 Amazon EC2。如果访问密钥在 15 分钟的时间范围内多次使用，则仅在此字段中记录第一次使用。

此字段的值在这些情况下为 N/A (不适用)：

- 用户没有访问密钥。
- 访问密钥从未使用过。
- 上次使用访问密钥是在 IAM 于 2015 年 4 月 22 日开始跟踪此信息之前。

`access_key_2_active`

如果用户有第二个访问密钥且第二个键的状态为 `Active`，则该值为 `TRUE`。否则为 `FALSE`。

Note

用户最多可以有两个访问密钥，以便于进行轮换。有关轮换访问密钥的更多信息，请参阅[轮换访问密钥 \(p. 100\)](#)。

`access_key_2_last_rotated`

创建或上次更改用户第二个访问密钥的日期和时间，[ISO 8601 日期时间格式](#)。如果用户没有第二个活动的访问密钥，则此字段中的值为 N/A (不适用)。

`access_key_2_last_used_date`

最近使用用户的第二个访问密钥签署 Amazon API 请求的日期和时间（采用 [ISO 8601 日期时间格式](#)）。如果访问密钥在 15 分钟的时间范围内多次使用，则仅在此字段中记录第一次使用。

此字段的值在这些情况下为 N/A (不适用)：

- 用户没有第二个访问密钥。
- 用户的第二个访问密钥从未使用过。
- 上次使用用户的第二个访问密钥是在 IAM 于 2015 年 4 月 22 日开始跟踪此信息之前。

`access_key_2_last_used_region`

最近在其中使用用户的第二个访问密钥的 [Amazon 区域](#)。如果访问密钥在 15 分钟的时间范围内多次使用，则仅在此字段中记录第一次使用。此字段的值在这些情况下为 N/A (不适用)：

- 用户没有第二个访问密钥。
- 用户的第二个访问密钥从未使用过。
- 上次使用用户的第二个访问密钥是在 IAM 于 2015 年 4 月 22 日开始跟踪此信息之前。
- 上次使用的服务不是区域特定的服务，如 Amazon S3。

`access_key_2_last_used_service`

最近使用用户的第二个访问密钥访问的 Amazon 服务。此字段的值使用服务的命名空间，例如 `s3` 表示 Amazon S3，`ec2` 表示 Amazon EC2。如果访问密钥在 15 分钟的时间范围内多次使用，则仅在此字段中记录第一次使用。此字段的值在这些情况下为 N/A (不适用)：

- 用户没有第二个访问密钥。
- 用户的第二个访问密钥从未使用过。
- 上次使用用户的第二个访问密钥是在 IAM 于 2015 年 4 月 22 日开始跟踪此信息之前。

cert_1_active

如果用户有 X.509 签名证书且证书状态为 Active，则该值为 TRUE。否则为 FALSE。

cert_1_last_rotated

创建或上次更改用户签名证书的日期和时间，ISO 8601 日期时间格式。如果用户没有活动的签名证书，则此字段中的值为 N/A (不适用)。

cert_2_active

如果用户有第二个 X.509 签名证书且证书状态为 Active，则该值为 TRUE。否则为 FALSE。

Note

用户最多可以有两个 X.509 签名证书，以便于进行证书轮换。

cert_2_last_rotated

创建或上次更改用户第二个签名证书的日期和时间，ISO 8601 日期时间格式。如果用户没有第二个活动的签名证书，则此字段中的值为 N/A (不适用)。

获取凭证报告 (控制台)

您可以使用 Amazon Web Services Management Console 下载逗号分隔值 (CSV) 文件形式的凭证报告。

下载凭证报告 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择凭证报告。
3. 选择下载报告。

获取凭证报告 (Amazon CLI)

下载凭证报告 (Amazon CLI)

1. 生成凭证报告。Amazon 存储单个报告。如果存在报告，生成凭证报告将覆盖以前的报告。`aws iam generate-credential-report`
2. 查看上次生成的报告：`aws iam get-credential-report`

获取凭证报告 (Amazon API)

下载凭证报告 (Amazon API)

1. 生成凭证报告。Amazon 存储单个报告。如果存在报告，生成凭证报告将覆盖以前的报告。`GenerateCredentialReport`
2. 查看上次生成的报告：`GetCredentialReport`

将 IAM 与 CodeCommit 结合使用：Git 凭证、SSH 密钥和 Amazon 访问密钥

CodeCommit 是一种在 Amazon 云中托管私有 Git 存储库的托管版本控制服务。要使用 CodeCommit，您需要配置 Git 客户端，使其与 CodeCommit 存储库通信。作为此配置的一部分，您需要提供 IAM 凭证，CodeCommit 将用该凭证来对您进行身份验证。IAM 支持具有三种凭证的 CodeCommit：

- Git 凭证 - 一种由 IAM 生成的用户名和密码对，供您通过 HTTPS 与 CodeCommit 存储库进行通信。
- SSH 密钥 - 一种本地生成的公有-私有密钥对，供您关联至 IAM 用户，以通过 SSH 与 CodeCommit 存储库进行通信。
- [Amazon 访问密钥 \(p. 96\)](#) - 供您配合 Amazon CLI 中包含的凭证辅助程序使用，以通过 HTTPS 与 CodeCommit 存储库通信。

Note

您无法使用 SSH 密钥或 Git 凭证来访问另一个 Amazon 账户中的存储库。要了解如何在另一个 Amazon 账户中为 IAM 用户和组配置对 CodeCommit 存储库的访问权限，请参阅 Amazon CodeCommit 用户指南中的[配置对使用角色的 Amazon CodeCommit 存储库的跨账户访问](#)。

请参阅下面几节了解有关每个选项的更多信息。

将 Git 凭证和 HTTPS 与 CodeCommit 配合使用（推荐）

借助 Git 凭证，您可以为 IAM 用户生成静态用户名和密码对，然后使用这些凭证进行 HTTPS 连接。您还可以将这些凭证与支持静态 Git 凭证的任何第三方工具或集成开发环境 (IDE) 配合使用。

这些凭证对所有支持的操作系统来说是通用的，并且兼容大多数凭证管理系统、开发环境及其他软件开发工具，因此，这是我们推荐使用的方法。您可以随时重置 Git 凭证的密码。如果不再需要凭证，您还可以停用或删除它们。

Note

您无法为 Git 凭证选择自己的用户名或密码。IAM 会为您生成这些凭证，以帮助确保它们符合 Amazon 的安全标准和 CodeCommit 中的安全存储库。凭证生成之后，您只能下载它们一次。因此，请务必将凭证保存在安全的位置。如有必要，您可以随时重置密码，但这会使通过旧密码配置的所有连接失效。您必须使用新密码重新配置连接，然后才能再次进行连接。

请参阅以下主题了解更多信息：

- 要创建 IAM 用户，请参阅 [在您的 Amazon 账户中创建 IAM 用户 \(p. 72\)](#)。
- 要借助 CodeCommit 生成并使用 Git 凭证，请参阅 Amazon CodeCommit 用户指南中的[适用于使用 Git 凭证的 HTTPS 用户](#)。

Note

在生成 Git 凭证后更改 IAM 用户的名称不会更改 Git 凭证的用户名。用户名和密码保持不变且仍然有效。

轮换特定于服务的凭证

1. 除了当前正在使用的一组凭证外，创建第二组特定于服务的凭证。
2. 更新所有应用程序以使用这组新凭证，然后验证应用程序能否正常工作。
3. 将原始凭证的状态更改为“停用”。
4. 确保您的所有应用程序仍能正常工作。
5. 删除已停用的特定于服务的凭证。

借助 CodeCommit 使用 SSH 密钥和 SSH

借助 SSH 连接，您可以在本地计算机上创建公有和私有密钥文件，以供 Git 和 CodeCommit 进行 SSH 身份验证。将公有密钥关联至 IAM 用户，并将私有密钥存储在本地计算机上。请参阅以下主题了解更多信息：

- 要创建 IAM 用户，请参阅 [在您的 Amazon 账户中创建 IAM 用户 \(p. 72\)](#)。
- 要创建 SSH 公有密钥并将其与 IAM 用户相关联，请参阅 Amazon CodeCommit 用户指南中的[适用于 Linux、macOS 或 Unix 上的 SSH 连接](#)或请参阅[适用于 Windows 上的 SSH 连接](#)。

Note

公有密钥必须采用 ssh-rsa 格式或 PEM 格式进行编码。公有密钥的最小位长度为 2048 位，最大长度为 16384 位。这独立于您上传的文件大小。例如，您可以生成 2048 位密钥，但生成的 PEM 文件大小为 1679 字节。如果您采用其他格式或大小提供公有密钥，则系统会显示一条错误消息，指出该密钥格式无效。

将 HTTPS 与 Amazon CLI 凭证辅助程序和 CodeCommit 配合使用

作为使用 Git 凭证进行 HTTPS 连接的替代方法，您可以允许 Git 在需要进行 Amazon 身份验证以与 CodeCommit 存储库交互时使用 IAM 用户凭证的加密签名版本或 Amazon EC2 实例角色。这是唯一一种不需要 IAM 用户就能连接 CodeCommit 存储库的方法。这也是唯一一种适用于联合访问和临时凭证的方法。除非您的业务要求进行联合访问或使用临时凭证，否则，我们强烈建议您创建和使用 IAM 用户进行访问。请参阅以下主题了解更多信息：

- 有关联合访问的更多信息，请参阅[身份提供程序和联合身份验证 \(p. 170\)](#)和[向经过外部身份验证的用户 \(联合身份验证\) 提供访问权限 \(p. 169\)](#)。
- 有关临时凭证的更多信息，请参阅[IAM 临时安全凭证 \(p. 300\)](#)和[临时访问 CodeCommit 存储库](#)。

Amazon CLI 凭证辅助程序不兼容 Keychain Access、Windows Credential Management 等其他凭证辅助系统。在使用凭证辅助程序配置 HTTPS 连接时，还有一些其他的配置注意事项。有关更多信息，请参阅 Amazon CodeCommit 用户指南中的[使用 Amazon CLI 凭证辅助程序在 Linux、macOS 或 Unix 上进行 HTTPS 连接](#)或使用[Amazon CLI 凭证辅助程序在 Windows 上进行 HTTPS 连接](#)。

将 IAM 与 Amazon Keyspaces 结合使用 (针对 Apache Cassandra)

Amazon Keyspaces (用于 Apache Cassandra) 是一种可扩展、高可用、托管的 Apache Cassandra 兼容数据库服务。您可以通过 Amazon Web Services Management Console 或以编程方式访问 Amazon Keyspaces。要使用服务特定的凭证以编程方式访问 Amazon Keyspaces，您可以使用 `cqlsh` 或开源 Cassandra 驱动程序。服务特定的凭证包括用户名和密码，例如 Cassandra 用于身份验证和访问管理的用户名和密码。要使用 Amazon 访问密钥以编程方式访问 Amazon Keyspaces，您可以使用 Amazon 开发工具包、Amazon Command Line Interface (Amazon CLI) 或带有 SiGv4 插件的开源 Cassandra 驱动程序。要了解更多信息，请参阅《Amazon Keyspaces (Apache Cassandra 兼容) 开发人员指南》中的[以编程方式连接到 Amazon Keyspaces](#)。

Note

如果您计划仅通过控制台与 Amazon Keyspaces 交互，则无需生成服务特定凭证。有关更多信息，请参阅《Amazon Keyspaces (Apache Cassandra 兼容) 开发人员指南》中的[使用控制台访问 Amazon Keyspaces](#)。

有关访问 Amazon Keyspaces 所需权限的更多信息，请参阅 Amazon Keyspaces (for Apache Cassandra) 开发人员指南中的[Amazon Keyspaces \(for Apache Cassra\) 基于身份的策略示例](#)。

生成 Amazon Keyspaces 凭证 (控制台)

您可以使用 Amazon Web Services Management Console 为 IAM 用户生成 Amazon Keyspaces 用户 (针对 Apache Cassandra) 凭证。

生成 Amazon Keyspaces 服务特定凭证 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Users (用户)，然后选择需要凭证的用户名称。
3. 在 Credentials for Amazon Keyspaces (for Apache Cassandra) (Amazon Keyspaces (for Apache Cassandra) 凭证) 下方的 Security Credentials (安全凭证) 选项卡中，选择 Generate credentials (生成凭证)。
4. 您的服务特定凭证现在可用。这是唯一一次查看或下载密码的机会。以后您无法恢复它。不过，您可以随时重置密码。将用户和密码保存在安全的位置，因为随后您需要使用它们。

生成 Amazon Keyspaces 凭证 (Amazon CLI)

您可以使用 Amazon CLI 为 IAM 用户生成 Amazon Keyspaces 用户 (针对 Apache Cassandra) 凭证。

生成 Amazon Keyspaces 服务特定凭证 (Amazon CLI)

- 使用以下命令：
 - `aws iam create-service-specific-credential`

生成 Amazon Keyspaces 凭证 (Amazon API)

您可以使用 Amazon API 为 IAM 用户生成 Amazon Keyspaces 用户 (针对 Apache Cassandra) 凭证。

生成 Amazon Keyspaces 服务特定凭证 (Amazon API)

- 完成以下操作：
 - `CreateServiceSpecificCredential`

在 IAM 中管理服务器证书

要在 Amazon 中启用与您的网站或应用程序的 HTTPS 连接，您需要 SSL/TLS 服务器证书。对于 Amazon Certificate Manager (ACM) 支持的区域中的证书，我们建议您使用 ACM 预置、管理和部署您的服务器证书。在不支持的区域中，您必须将 IAM 作为证书管理器。要了解 ACM 支持的区域，请参阅 Amazon 一般参考中的 [Amazon Certificate Manager 端点和配额](#)。

只有当您必须在 [ACM 不支持](#) 的区域中支持 HTTPS 连接时，才应使用 IAM 作为证书管理器。IAM 安全地加密您的私有密钥并将加密的版本存储在 IAM SSL 证书存储中。IAM 支持在所有区域部署服务器证书，但您必须从外部提供商获取证书，以便与 Amazon 搭配使用。您无法将 ACM 证书上传到 IAM。此外，您还无法从 IAM 控制台管理证书，

有关将第三方证书上传到 IAM 的更多信息，请参阅以下主题。

目录

- [上传服务器证书 \(Amazon API\) \(p. 148\)](#)
- [检索服务器证书 \(Amazon API\) \(p. 148\)](#)
- [列出服务器证书 \(Amazon API\) \(p. 149\)](#)
- [标记和取消标记服务器证书 \(Amazon API\) \(p. 149\)](#)
- [重命名服务器证书或更新其路径 \(Amazon API\) \(p. 149\)](#)
- [删除服务器证书 \(Amazon API\) \(p. 149\)](#)
- [问题排查 \(p. 150\)](#)

上传服务器证书 (Amazon API)

要将服务器证书上传到 IAM，您必须提供证书及其匹配的私有密钥。如果证书不是自签名的，则您还必须提供证书链。(上传自签名证书时无需证书链。) 在上传证书前，请确保您已具有所有这些项目而且它们满足以下条件：

- 证书在上传时必须是有有效的。您不能在证书有效期开始 (证书的 NotBefore 日期) 之前或证书有效期到期 (证书的 NotAfter 日期) 之后上传证书。
- 私有密钥必须是未加密的。您不能上传受密码或口令保护的私有密钥。有关解密已加密的私有密钥的帮助信息，请参阅[问题排查](#) (p. 150)。
- 证书、私有密钥和证书链必须均采用 PEM 编码。有关将这些项目转换为 PEM 格式的 helping 信息，请参阅[问题排查](#) (p. 150)。

要使用 IAM API 上传证书，请发送 [UploadServerCertificate](#) 请求。以下示例说明如何使用 [Amazon Command Line Interface \(Amazon CLI\)](#) 执行该操作。示例假定以下各项：

- PEM 编码的证书存储在名为 Certificate.pem 的文件中。
- PEM 编码的证书链存储在名为 CertificateChain.pem 的文件中。
- PEM 编码的未加密私有密钥存储在名为 PrivateKey.pem 的文件中。
- (可选) 您希望用键-值对标记服务器证书。例如，您可以添加标签键 Department 和标签值 Engineering 来帮助识别和组织证书。

要使用以下示例命令，请用自己的文件名替换这些文件名。将 *ExampleCertificate* 替换为已上传证书的名称。如果要标记证书，请将 *ExampleKey* 和 *ExampleValue* 标签键值对替换为您自己的值。在一个连续行上键入命令。为更便于阅读，以下示例包含了换行符和多余的空格。

```
aws iam upload-server-certificate --server-certificate-name ExampleCertificate
                                --certificate-body file://Certificate.pem
                                --certificate-chain file://CertificateChain.pem
                                --private-key file://PrivateKey.pem
                                --tags '{"Key": "ExampleKey", "Value": "ExampleValue"}'
```

如果上述命令执行成功，则它将返回有关上传的证书的元数据，包括其 [Amazon Resource Name \(ARN\)](#) (p. 896)、友好名称、标识符 (ID)、到期日期、标签等。

Note

如果您要上传服务器证书以用于 Amazon CloudFront，则必须使用 --path 选项指定路径。路径必须以 /cloudfront 开头且必须包含尾部反斜杠 (例如，/cloudfront/test/)。

要使用 Amazon Tools for Windows PowerShell 上传证书，请使用 [Publish-IAMServerCertificate](#)。

检索服务器证书 (Amazon API)

要使用 IAM API 检索证书，请发送 [GetServerCertificate](#) 请求。以下示例说明如何使用 Amazon CLI 执行该操作。将 *ExampleCertificate* 替换为要检索的证书的名称。

```
aws iam get-server-certificate --server-certificate-name ExampleCertificate
```

如果上述命令执行成功，则它将返回证书、证书链 (如果已上传一个) 和有关证书的元数据。

Note

在您上传后，无法从 IAM 下载或检索私有密钥。

要使用 Amazon Tools for Windows PowerShell 检索证书，请使用 [Get-IAMServerCertificate](#)。

列出服务器证书 (Amazon API)

要使用 IAM API 列出您已上传的服务器证书，请发送 [ListServerCertificates](#) 请求。以下示例说明如何使用 Amazon CLI 执行该操作。

```
aws iam list-server-certificates
```

如果上述命令执行成功，则它将返回包含有关每个证书的元数据的列表。

要使用 Amazon Tools for Windows PowerShell 列出您已上传的服务器证书，请使用 [Get-IAMServerCertificates](#)。

标记和取消标记服务器证书 (Amazon API)

您可以将标签附加到 IAM 资源以组织和控制对这些资源的访问。要使用 IAM API 为现有的服务器证书添加证书，请发送 [TagServerCertificate](#) 请求。以下示例说明如何使用 Amazon CLI 执行该操作。

```
aws iam tag-server-certificate --server-certificate-name ExampleCertificate
                                --tags '{"Key": "ExampleKey", "Value": "ExampleValue"}'
```

上述命令成功后，不会返回任何输出。

要使用 IAM API 取消标记服务器证书，请发送 [UntagServerCertificate](#) 请求。以下示例说明如何使用 Amazon CLI 执行该操作。

```
aws iam untag-server-certificate --server-certificate-name ExampleCertificate
                                --tag-keys ExampleKeyName
```

上述命令成功后，不会返回任何输出。

重命名服务器证书或更新其路径 (Amazon API)

要使用 IAM API 重命名服务器证书或更新其路径，请发送 [UpdateServerCertificate](#) 请求。以下示例说明如何使用 Amazon CLI 执行该操作。

要使用以下示例命令，请将旧的证书名称和证书路径替换为新的，然后在一个连续行上键入命令。为更便于阅读，以下示例包含了换行符和多余的空格。

```
aws iam update-server-certificate --server-certificate-name ExampleCertificate
                                  --new-server-certificate-name CloudFrontCertificate
                                  --new-path /cloudfront/
```

如果上述命令执行成功，则它不会返回任何输出

要使用 Amazon Tools for Windows PowerShell 重命名服务器证书或更新其路径，请使用 [Update-IAMServerCertificate](#)。

删除服务器证书 (Amazon API)

要使用 IAM API 删除服务器证书，请发送 [DeleteServerCertificate](#) 请求。以下示例说明如何使用 Amazon CLI 执行该操作。

要使用以下示例命令，请将 *ExampleCertificate* 替换为要删除的证书的名称。

```
aws iam delete-server-certificate --server-certificate-name ExampleCertificate
```

如果上述命令执行成功，则它不会返回任何输出

要使用 Amazon Tools for Windows PowerShell 删除服务器证书，请使用 [Remove-IAMServerCertificate](#)。

问题排查

您必须先确保证书、私有密钥和证书链均是 PEM 编码的，然后才能将证书上传到 IAM。您还必须确保私有密钥是未加密的。请见以下示例。

Example PEM 编码证书示例

```
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----
```

Example PEM 编码的未加密的私有密钥示例

```
-----BEGIN RSA PRIVATE KEY-----  
Base64-encoded private key  
-----END RSA PRIVATE KEY-----
```

Example PEM 编码的证书链示例

一个证书链包含一个或多个证书。您可以使用文本编辑器、Windows 中的复制命令或 Linux cat 命令，以将证书文件串联到一个链中。如果包含多个证书，则每个证书必须证明上一个证书。您可以串联证书以完成该操作，包括最后的根 CA 证书。

以下示例包含三个证书，但证书链可能包含更多或更少的证书。

```
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
Base64-encoded certificate  
-----END CERTIFICATE-----
```

如果这些项目在上传到 IAM 时未采用正确的格式，您可以使用 [OpenSSL](#) 将其转换为正确的格式。

将证书或证书链从 DER 转换为 PEM

使用 [OpenSSL x509 命令](#)，如下示例所示。在以下示例命令中，将 *Certificate.der* 替换为包含您的 DER 编码的证书的文件的名称。将 *Certificate.pem* 替换为要包含 PEM 编码的证书的输出文件的首选名称。

```
openssl x509 -inform DER -in Certificate.der -outform PEM -out Certificate.pem
```

将私有密钥从 DER 转换为 PEM

使用 [OpenSSL rsa](#) 命令，如以下示例所示。在以下示例命令中，将 *PrivateKey.der* 替换为包含您的 DER 编码的私有密钥的文件的名称。将 *PrivateKey.pem* 替换为要包含 PEM 编码的私有密钥的输出文件的首选名称。

```
openssl rsa -inform DER -in PrivateKey.der -outform PEM -out PrivateKey.pem
```

解密已加密的私有密钥 (删除密码或口令)

使用 [OpenSSL rsa](#) 命令，如以下示例所示。要使用以下示例命令，请将 *EncryptedPrivateKey.pem* 替换为包含您的已加密的私有密钥的文件的名称。将 *PrivateKey.pem* 替换为要包含 PEM 编码的未加密私有密钥的输出文件的首选名称。

```
openssl rsa -in EncryptedPrivateKey.pem -out PrivateKey.pem
```

将证书包从 PKCS#12 (PFX) 转换为 PEM

使用 [OpenSSL pkcs12](#) 命令，如以下示例所示。在以下示例命令中，将 *CertificateBundle.p12* 替换为包含您的 PKCS#12 编码的证书包的文件的名称。将 *CertificateBundle.pem* 替换为要包含 PEM 编码的证书捆绑包的输出文件的首选名称。

```
openssl pkcs12 -in CertificateBundle.p12 -out CertificateBundle.pem -nodes
```

将证书包从 PKCS#7 转换为 PEM

使用 [OpenSSL pkcs7](#) 命令，如以下示例所示。在以下示例命令中，将 *CertificateBundle.p7b* 替换为包含您的 PKCS#7 编码的证书包的文件的名称。将 *CertificateBundle.pem* 替换为要包含 PEM 编码的证书捆绑包的输出文件的首选名称。

```
openssl pkcs7 -in CertificateBundle.p7b -print_certs -out CertificateBundle.pem
```

IAM 用户组

[IAM 用户组 \(p. 151\)](#) 是 IAM 用户的集合。利用用户组，可为多个用户指定权限，以便更轻松地管理这些用户的权限。例如，您可能有一个名为 Admins 的用户组，并向该用户组授予管理员通常需要的权限类型。该用户组中的所有用户均自动拥有 Admins 组权限。如果有新用户加入您的组织，并且需要管理员权限，则可通过将此用户添加到 Admins 用户组来分配相应的权限。如果您的组织中有成员更换工作，则不必编辑该用户的权限，只需从旧用户组中将其删除，然后将其添加到相应的新用户组即可。

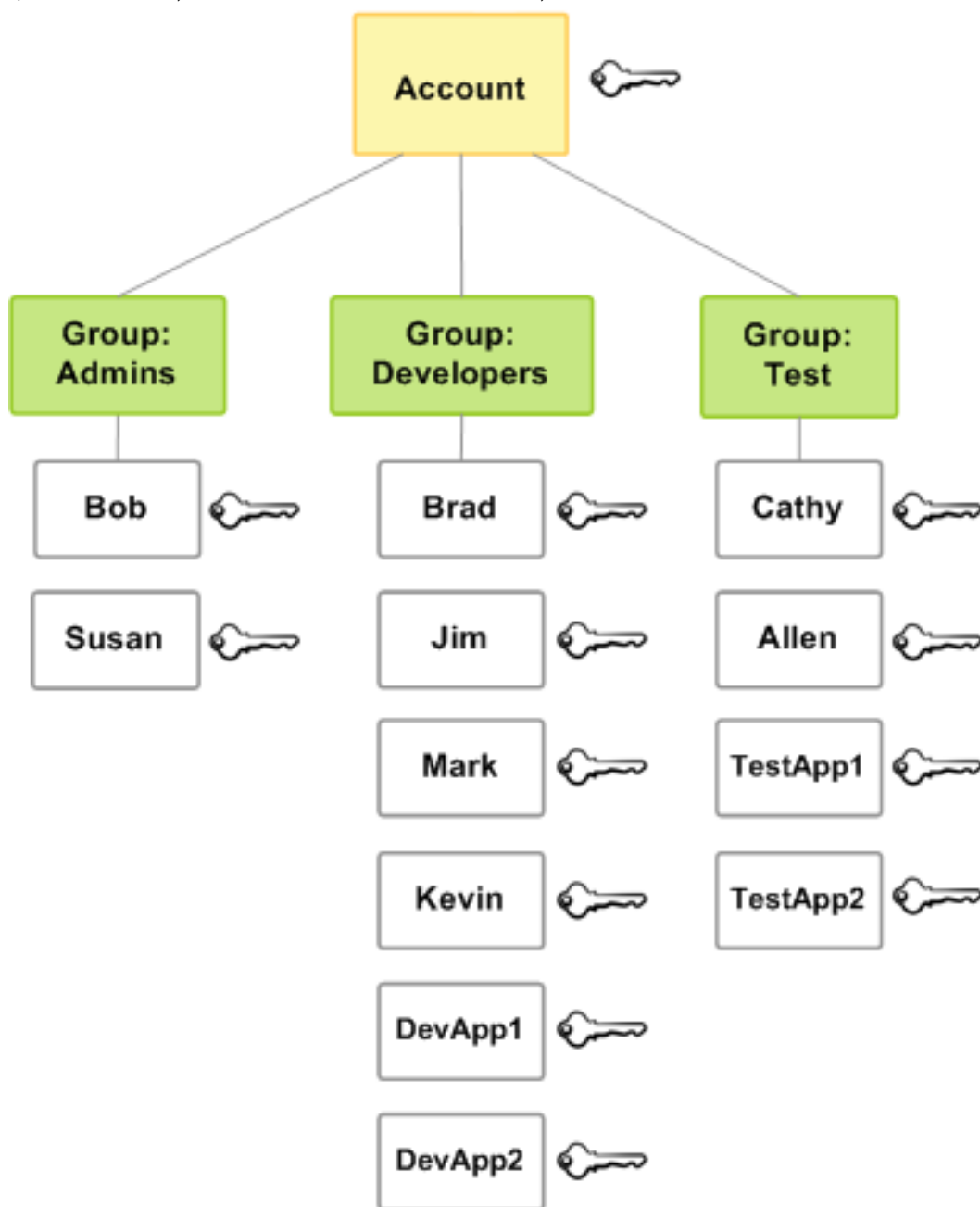
您可以将基于身份的策略附加到用户组，以便该用户组中的所有用户都能获得该策略的权限。您无法在策略（例如基于资源的策略）中将用户组标识为 Principal，因为组与权限相关，与身份验证无关，并且主体是经过身份验证的 IAM 实体。有关策略类型的更多信息，请参阅[基于身份的策略和基于资源的策略 \(p. 379\)](#)。

以下为用户组具有的一些重要特点：

- 一个用户组可包含多个用户，而一个用户又可归属于多个用户组。
- 用户组不能嵌套；其中只能包含用户，而不能包含其他用户组。

- 默认情况下，没有可自动包含 Amazon Web Services 账户内所有用户的用户组。如果希望有这样的用户组，则需要创建该组，然后将每个新用户分配给它。
- Amazon Web Services 账户中 IAM 资源的数量和大小（例如组的数量和用户可加入的组的数量）是有限的。有关更多信息，请参阅[IAM 与 Amazon STS 配额、名称要求和字符限制](#) (p. 902)。

下图以一家小型公司作为简单示例。随着公司的发展，公司所有者为用户创建了一个 Admins 用户组来创建和管理其他用户。Admins 用户组会创建一个 Developers 用户组和 Test 用户组。这两个用户组都包含可与 Amazon 交互的用户（人员和应用程序：Jim、Brad、DevApp1 等）。每个用户均拥有一组单独的安全证书。在此示例中，每个用户均属于单一用户组。不过，用户可归属于多个用户组。



创建 IAM 用户组

要设置用户组，您需要先创建组。然后根据您希望组中用户执行的工作类型向组授予权限。最后，将用户添加到组中。

有关创建用户组时需要的权限的信息，请参阅 [访问 IAM 资源所需的权限 \(p. 508\)](#)。

创建 IAM 用户组并附加策略（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 User groups（用户组），然后选择 Create group（创建组）。
3. 对于 User group name（用户组名称），键入组的名称。

Note

Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅[IAM 与 Amazon STS 配额、名称要求和字符限制 \(p. 902\)](#)。组名称可以是一个最多由 128 个字母、数字和以下字符构成的组合：加号 (+)、等号 (=)、逗号 (,)、句点 (.)、at 符号 (@)、下划线 (_) 和连字符 (-)。账户中的名称必须唯一。名称不区分大小写。例如，您无法同时创建名为 **ADMINS** 和 **admins** 的组。

4. 在用户列表中，选中要添加到组中的每个用户对应的复选框。
5. 在策略列表中，选中要应用于组中的所有成员的每个策略的复选框。
6. 选择创建组。

有关如何设置 Administrators 用户组的示例，请参阅 [创建您的第一个 IAM 管理员用户和用户组 \(p. 19\)](#)。

要创建 IAM 用户组（Amazon CLI 或者 Amazon API）

使用以下值之一：

- Amazon CLI：[aws iam create-group](#)
- Amazon API：[CreateGroup](#)

管理 IAM 用户组

Amazon Web Services 提供了用于管理 IAM 用户组的多种工具。有关在用户组中添加和删除用户所需的权限的信息，请参阅 [访问 IAM 资源所需的权限 \(p. 508\)](#)。

主题

- [列出 IAM 用户组 \(p. 153\)](#)
- [在 IAM 用户组中添加和删除用户 \(p. 154\)](#)
- [将策略附加到 IAM 用户组 \(p. 155\)](#)
- [重命名 IAM 用户组 \(p. 156\)](#)
- [删除 IAM 用户组 \(p. 156\)](#)

列出 IAM 用户组

您可以列出您账户中的所有用户组、列出某个用户组中的用户和列出用户所属的用户组。如果您使用 Amazon CLI 或 Amazon API，则可以列出带特定路径前缀的所有用户组。

列出您账户中的所有用户组

执行以下任一操作：

- [Amazon Web Services Management Console](#)：在导航窗格中，选择 User groups (用户组)。
- Amazon CLI：[aws iam list-groups](#)
- Amazon API：[ListGroup](#)s

列出特定用户组中的用户

执行以下任一操作：

- [Amazon Web Services Management Console](#)：在导航窗格中，请选择 Groups (组)，选择组的名称，然后选择 Users (用户) 选项卡。
- Amazon CLI：[aws iam get-group](#)
- Amazon API：[GetGroup](#)

列出用户所属的所有用户组

执行以下任一操作：

- [Amazon Web Services Management Console](#)：在导航窗格中，选择 Users，选择用户名称，然后选择 Groups 选项卡。
- Amazon CLI：[aws iam list-groups-for-user](#)
- Amazon API：[ListGroupForUser](#)

在 IAM 用户组中添加和删除用户

使用用户组可以跨多个有用户同时应用相同的权限策略。然后，您可以在 IAM 组中添加或删除用户。由于您组织的人员是流动的，因此这样做会很有用。

查看策略访问

在更改策略的权限之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅[使用上次访问的信息优化 Amazon 中的权限](#) (p. 467)。

在用户组中添加或删除用户（控制台）

您可以使用 Amazon Web Services Management Console 在用户组中添加或删除用户。

要将用户添加到 IAM 用户组（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，请选择 User groups (用户组)，然后选择组的名称。
3. 请选择 Users (用户) 选项卡，然后选择 Add users (添加用户)。选中要添加的用户旁边的复选框。
4. 选择 Add Users (添加用户)。

从 IAM 组中删除用户（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。

2. 在导航窗格中，请选择 User groups (用户组) ，然后选择组的名称。
3. 选择 Users (用户) 选项卡。选中要移除的用户旁边的复选框，然后选择 Remove users (移除用户) 。

在用户组中添加或删除用户 (Amazon CLI)

您可以使用 Amazon CLI 在用户组中添加或删除用户。

将用户添加到 IAM 用户组 (Amazon CLI)

- 使用以下命令：
- [aws iam add-user-to-group](#)

从 IAM 用户组中删除用户 (Amazon CLI)

- 使用以下命令：
- [aws iam remove-user-from-group](#)

在用户组中添加或删除用户 (Amazon API)

您可以使用 Amazon API 在用户组中添加或删除用户。

将用户添加到 IAM 组 (Amazon API)

- 完成以下操作：
- [AddUserToGroup](#)

从 IAM 用户组中删除用户 (Amazon API)

- 完成以下操作：
- [RemoveUserFromGroup](#)

将策略附加到 IAM 用户组

您可以将 [Amazon 托管策略 \(p. 364\)](#) - 由 Amazon 提供的预先写入的策略 - 附加到用户组，如下列步骤中所述。要附加客户托管策略 (即具有您创建的自定义权限的策略) ，必须首先创建策略。有关创建客户托管策略的信息，请参阅 [创建 IAM policy \(p. 437\)](#) 。

有关权限和策略的更多信息，请参阅[适用于 Amazon 资源的 Access Management \(p. 353\)](#)。

要将策略附加到用户组 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，请选择 User groups (用户组) ，然后选择组的名称。
3. 请选择 Permissions 选项卡。
4. 选择 Add Permissions (添加权限) ，然后选择 Attach Policy (附上策略) 。
5. 附加到用户组的当前策略显示在 Current permissions policies (当前权限策略) 列表。在 Other permissions policies (其他权限策略) 列表中，选中要附加的策略名称旁边的复选框。您可以使用搜索框通过输入类型和策略名称来筛选策略列表。

6. 选择 Attach policies (附上策略) 。

要将策略附加到用户组 (Amazon CLI 或者 Amazon API)

请执行以下任一操作：

- Amazon CLI : [aws iam attach-group-policy](#)
- Amazon API : [AttachGroupPolicy](#)

重命名 IAM 用户组

当您更改用户组的名称或路径时，发生以下情况：

- 应用于用户组的所有策略采用新群组名称继续生效。
- 该用户组采用新名称保留所有用户。
- 该用户组的唯一 ID 保持不变。有关唯一 ID 的更多信息，请参阅 [唯一标识符 \(p. 900\)](#)。

IAM 不自动更新将该用户组作为资源引用的策略以使用新名称。因此，重命名用户组时，请务必保持谨慎。在重命名用户组之前，您必须手动检查所有策略以按名称查找提到该用户组的任何策略。例如，假设 Bob 是公司测试部门的经理。Bob 向他的 IAM 用户实体中挂载了一个策略，该策略允许他在“测试”用户组中添加和删除用户。如果管理员更改用户组的名称（或更改组的路径），则管理员还必须更新附加到 Bob 的策略以使用新名称或新路径。否则，Bob 将无法在该用户组中添加或删除用户。

查找将用户组作为资源引用的策略：

1. 从 IAM 控制台的导航窗格中，选择 Policies (策略) 。
2. 按 Type (类型) 列排序，以查找您的 Customer managed (客户托管) 自定义策略。
3. 请选择策略的名称进行编辑。
4. 请选择 Permissions (权限) 选项卡，然后选择 Bucket Policy (存储桶策略) 。
5. 从服务列表中选择 IAM (如果存在)
6. 在资源列中查找您的用户组名称。
7. 选择编辑策略以在策略中更改您的用户组名称。

更改 IAM 用户组的名称

执行以下任一操作：

- [Amazon Web Services Management Console](#)：在导航窗格中，选择 User groups (用户组) ，然后选择用户组的名称。选择 Edit (编辑) 。键入新的用户组名称，然后选择 Save changes (保存更改) 。
- Amazon CLI : [aws iam update-group](#)
- Amazon API : [UpdateGroup](#)

删除 IAM 用户组

您在 Amazon Web Services Management Console 中删除用户组时，控制台将自动删除所有组成员，分离所有已附加的托管策略并删除所有内联策略。不过，由于 IAM 不会自动删除将用户组作为资源引用的策略，因此，在删除用户组时必须格外小心。在删除用户组之前，您必须手动检查所有策略以按名称查找提到该组的任何策略。例如，测试团队负责人 John 向其 IAM 用户实体附加了一个策略，该策略允许他在“测试”用户组中添加和删除用户。如果管理员删除该组，管理员还必须删除挂载到 John 的策略。否则，如果管理员重新创建已删除组并以相同名称命名，那么即使 John 已经离开测试团队，其权限仍然保持不变。

查找将用户组作为资源引用的策略

1. 从 IAM 控制台的导航窗格中，选择 Policies (策略)。
2. 按 Type (类型) 列排序，以查找您的 Customer managed (客户托管) 自定义策略。
3. 请选择要删除策略的策略名称。
4. 请选择 Permissions (权限) 选项卡，然后选择 Bucket Policy (存储桶策略)。
5. 从服务列表中选择 IAM (如果存在)
6. 在资源列中查找您的用户组名称。
7. 选择删除策略以删除该策略。

相反，在使用 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 删除用户组时，必须先删除组中的用户。然后删除嵌入到用户组中的所有内联策略。接下来，分离所有已附加到组的托管策略。只有这样才能删除用户组本身。

删除 IAM 用户组 (控制台)

可从 Amazon Web Services Management Console 中删除 IAM 用户组。

删除 IAM 用户组 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 User groups (用户组)。
3. 在用户组列表中，选中要删除的用户组的名称旁边的复选框。您可以使用搜索框按类型、权限和用户组名称筛选用户组列表。
4. 选择 Delete。
5. 在确认框中，如果要删除单个用户组，请键入用户组名称，然后选择 Delete (删除)。如果要删除多个用户组，请键入要删除的用户组数，然后键入 **user groups**，并选择 Delete (删除)。例如，如果您要删除三个用户组，请键入 **3 user groups**。

删除 IAM 用户组 (Amazon CLI)

可从 Amazon CLI 中删除 IAM 用户组。

删除 IAM 用户组 (Amazon CLI)

1. 从群组移除所有用户组。
 - [aws iam get-group](#) (用于获取用户组中的用户列表) 和 [aws iam remove-user-from-group](#) (用于从用户组中删除用户)
2. 删除嵌入到用户组中的所有内联策略。
 - [aws iam list-group-policies](#) (用于获取用户组的内联策略的列表) 和 [aws iam delete-group-policy](#) (用于删除用户组的内联策略)
3. 分离所有已附加到用户组的托管策略。
 - [aws iam list-attached-group-policies](#) (用于获取已附加到用户组的托管策略的列表) 和 [aws iam detach-group-policy](#) (用于将托管策略从用户组中分离)
4. 删除该用户组。
 - [aws iam delete-group](#)

删除 IAM 用户组 (Amazon API)

您可以使用 Amazon API 删除 IAM 用户组。

删除 IAM 用户组 (Amazon API)

1. 从群组移除所有用户组。
 - [GetGroup](#) (用于获取用户组中的用户的列表) 和 [RemoveUserFromGroup](#) (用于从用户组中删除用户)
2. 删除嵌入到用户组中的所有内联策略。
 - [ListGroupPolicies](#) (用于获取用户组的内联策略的列表) 和 [DeleteGroupPolicy](#) (用于删除用户组的内联策略)
3. 分离所有已附加到用户组的托管策略。
 - [ListAttachedGroupPolicies](#) (用于获取已附加到用户组的托管策略的列表) 和 [DetachGroupPolicy](#) (用于将托管策略从用户组中分离)
4. 删除该用户组。
 - [DeleteGroup](#)

IAM 角色

IAM 角色是可在账户中创建的一种具有特定权限的 IAM 身份。IAM 角色类似于 IAM 用户，因为它是一个 Amazon 身份，具有确定其在 Amazon 中可执行和不可执行的操作的权限策略。但是，角色旨在让需要它的任何人代入，而不是唯一地与某个人员关联。此外，角色没有关联的标准长期凭证（如密码或访问密钥）。相反，当您代入角色时，它会为您提供角色会话的临时安全凭证。

您可以使用角色向通常无权访问您的 Amazon 资源的用户、应用程序或服务提供访问权限。例如，您可能需要向您 Amazon 账户中的用户授予对他们通常不拥有的资源的访问权限，或是向一个 Amazon 账户的用户授予对另一个账户中的资源的访问权限。或者，您可能需要允许移动应用程序使用 Amazon 资源，但是不希望将 Amazon 密钥嵌入在应用程序中（在其中难以轮换密钥，而且用户可能提取它们）。有时，您需要向已经具有在 Amazon 外部（例如，在您的公司目录中）定义的身份的用户提供对 Amazon 的访问权限。或者，您可能需要向第三方授予对您账户的访问权限，使他们可以对您的资源执行审核。

对于这些情况，您可以使用 IAM 角色委派对 Amazon 资源的访问权限。本节介绍各种角色和它们的不同使用方式，何时及如何选择方法，如何创建、管理、切换到（或担任）和删除角色。

主题

- [角色术语和概念 \(p. 158\)](#)
- [针对角色的常见情形：用户、应用程序和服务 \(p. 161\)](#)
- [身份提供程序和联合身份验证 \(p. 170\)](#)
- [使用服务相关角色 \(p. 208\)](#)
- [创建 IAM 角色 \(p. 215\)](#)
- [使用 IAM 角色 \(p. 236\)](#)
- [管理 IAM 角色 \(p. 260\)](#)
- [IAM 角色与基于资源的策略有何不同 \(p. 274\)](#)

角色术语和概念

以下是可帮助您开始使用角色的一些基本术语。

角色

可在账户中创建的具有特定权限的 IAM 身份。IAM 角色与 IAM 用户有一些相似之处。角色和用户都是具有权限策略的 Amazon 身份，该策略可确定身份在 Amazon 中可执行和不可执行的操作。但是，角色旨在让需要它的任何人代入，而不是唯一地与某个人员关联。此外，角色没有关联的标准长期凭证（如密码或访问密钥）。相反，当您代入角色时，它会为您提供角色会话的临时安全凭证。

角色可由以下用户使用：

- 与该角色在相同 Amazon 账户中的 IAM 用户
- 位于与该角色不同的 Amazon 账户中的 IAM 用户
- 由 Amazon 提供的 Web 服务，例如 Amazon Elastic Compute Cloud (Amazon EC2)
- 由与 SAML 2.0 或 OpenID Connect 兼容的外部身份提供程序 (IdP) 服务或定制的身份代理进行身份验证的外部用户。

Amazon 服务 角色

服务代表您在您的账户中执行操作而担任的角色。在设置一些 Amazon 服务环境时，您必须为服务定义要代入的角色。这个服务角色必须包含该服务访问所需的 Amazon 资源会用到的所有必要权限。服务角色因服务而异，但只要您满足服务记录在案的要求，许多服务都允许您选择权限。您可以从 IAM 中创建、修改和删除服务角色。

EC2 实例的 Amazon 服务角色

在 Amazon EC2 实例上运行的应用程序可以代入以在您的账户中执行操作的特殊类型的服务角色。角色会在启动时分配给 EC2 实例。在该实例上运行的应用程序可以检索临时安全凭证并执行角色允许的操作。有关使用 EC2 实例的服务角色的详细信息，请参阅[使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限 \(p. 252\)](#)。

Amazon 服务相关角色

与 Amazon 服务直接关联的一种独特类型的服务角色。服务相关角色由服务预定义，具有服务代表您调用其他 Amazon 服务所需的所有权限。相关的服务还定义了创建、修改和删除服务相关角色的方式。服务可以自动创建或删除角色。它可能允许您在服务的向导或流程中创建、修改或删除角色。或者，它可能需要您使用 IAM 创建或删除角色。无论使用哪种方法，服务相关角色都可让您更轻松地设置服务，因为您不必手动添加必要的权限。

Note

如果在某项服务开始支持服务相关角色时您已使用该服务，您可能会收到一封电子邮件，告知您账户中的新角色。在这种情况下，该服务已自动在您的账户中创建服务相关角色。您无需执行任何操作来支持该角色，并且您不应手动删除它。有关更多信息，请参阅[我的 Amazon 账户中出现新角色 \(p. 885\)](#)。

有关哪些服务支持使用服务相关角色的信息，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)并查找其在服务相关角色列中为是的服务。请选择 Yes 与查看该服务的服务相关角色文档的链接。如果该服务不包括有关创建、修改或删除服务相关角色的文档，则可以使用 IAM 控制台、Amazon CLI 或 API。有关更多信息，请参阅[使用服务相关角色 \(p. 208\)](#)。

角色链

角色链是指通过 Amazon CLI 或 API 使用一个角色代入另一个角色。例如，RoleA 有权代入 RoleB。您可以在 AssumeRole API 操作中使用其长期用户凭证以允许 User1 代入 RoleA。该操作会返回 RoleA 短期凭证。借助角色链，您可以使用 RoleA 的短期凭证允许 User1 代入 RoleB。

当您在代入某个角色的同时，您可以传递会话标签并将键设置为可传递。可传递会话标签将传递给角色链中的所有后续会话。要了解会话标签的更多信息，请参阅[在 Amazon STS 中传递会话标签 \(p. 293\)](#)。

角色链将您的 Amazon CLI 或 Amazon API 角色会话限制为最长 1 小时。在使用 AssumeRole API 操作代入角色时，您可以使用 DurationSeconds 参数指定角色会话的持续时间。您可以指定最多为 43200

秒 (12 小时) 的参数值, 具体取决于您的角色的[最大会话持续时间设置 \(p. 238\)](#)。不过, 如果您使用角色链代入角色并提供大于 1 小时的 DurationSeconds 参数值, 则操作将失败。

Amazon 不会将使用角色向 [EC2 实例上运行的应用程序授予权限 \(p. 252\)](#) 视作角色链。

委托

为某人授予权限以允许访问您控制的资源。委派涉及在两个账户之间建立信任。第一个是拥有资源的账户 (信任账户)。第二个是包含需要访问资源的用户的账户 (可信账户)。可信账户和信托账户可为以下任一账户:

- 同一账户。
- 受您的组织控制的两个单独账户。
- 由不同组织拥有的两个账户。

要委派权限以访问资源, 请在附加两个[策略 \(p. 215\)](#)的信任账户中[创建 IAM 角色 \(p. 160\)](#)。权限策略授予角色的用户所需权限对资源执行预期任务。信任策略指定允许哪些受信任账户成员担任角色。

当您创建信任策略时, 您不能将通配符 (*) 指定为主体。信任策略附加到信任账户中的角色, 并且是权限的一半。另一半是附加到[允许用户切换到或担任角色 \(p. 239\)](#)的受信任账户中的用户的权限策略。临时担任一个角色的用户将放弃自己的权限而接过该角色的权限。用户退出或停止使用角色时, 恢复原始用户权限。另一个名为[外部 ID \(p. 164\)](#)的参数有助于确保在不受同一企业控制的账户之间安全使用角色。

联合身份验证

创建外部身份提供程序与 Amazon 之间的信任关系。用户可以登录 Web 身份提供程序, 例如 Login with Amazon、Facebook、Google, 或任何与 OpenID Connect (OIDC) 兼容的 IdP。用户还可登录与安全断言标记语言 (SAML) 2.0 兼容的企业身份系统, 如 Microsoft Active Directory 联合身份验证服务。当您使用 OIDC 和 SAML 2.0 在这些外部身份提供程序与 Amazon 之间配置信任关系时, 会将用户分配到 IAM 角色。用户还会获得允许该用户访问您的 Amazon 资源的临时凭证。

联合身份用户

您也可以不创建 IAM 用户, 而是使用来自 Amazon Directory Service、您的企业用户目录或 Web 身份提供程序的现有身份。这些用户称为联合身份用户。在通过[身份提供程序 \(p. 170\)](#)请求访问权限时, Amazon 将为联合身份用户分配角色。有关联合身份用户的更多信息, 请参阅 IAM 用户指南中的[联合身份用户和角色 \(p. 11\)](#)。

信任策略

在 [JSON 策略文档 \(p. 985\)](#) 中, 您可以定义您信任代入该角色的主体。角色信任策略是必需的[基于资源的策略 \(p. 356\)](#) (将附加到 IAM 中的角色)。您可以在信任策略中指定的[主体 \(p. 936\)](#)包括用户、角色、账户和服务。

权限策略

一个 [JSON 格式](#) 的权限文档, 您可在其中定义角色可以访问的操作和资源。该文档是根据 [IAM policy 语言 \(p. 933\)](#) 的规则编写的。

权限边界

一项高级功能, 借助该功能, 使用基于身份的策略来限制可以授予角色的最大权限。无法将权限边界应用于服务相关角色。有关更多信息, 请参阅[IAM 实体的权限边界 \(p. 370\)](#)。

主体

Amazon 中可执行操作并访问资源的实体。主体可以是 Amazon Web Services 账户根用户、IAM 用户或角色。您可以通过以下两种方式之一授予访问资源的权限:

- 您可以将权限策略附加到用户 (直接附加或通过组间接附加) 或角色。
- 对于那些支持[基于资源的策略 \(p. 11\)](#)的服务, 您可以在附加到资源的策略的 Principal 元素中识别主体。

如果您将一个 Amazon 账户称为主体，则它通常表示该账户中定义的任何主体。

Note

您不能在用户信任策略的 Principal 元素中使用通配符 (*)。
用于跨账户访问的角色

将一个账户中的资源的访问权限授予另一个账户中的受信任主体的账户。角色是授予跨账户访问权限的主要方式。但是，某些 Amazon 服务允许您将策略直接附加到资源（而不是使用角色作为代理）。这些策略称为基于资源的策略，您可以使用它们向其他 Amazon 账户中的主体授予对资源的访问权限。其中一些资源包括 Amazon Simple Storage Service (S3) 存储桶、S3 Glacier 文件库、Amazon Simple Notification Service (SNS) 主题和 Amazon Simple Queue Service (SQS) 队列。要了解哪些服务支持基于资源的策略，请参阅 [使用 IAM 的 Amazon 服务 \(p. 908\)](#)。有关基于资源的策略的更多信息，请参阅 [IAM 角色与基于资源的策略有何不同 \(p. 274\)](#)。

针对角色的常见情形：用户、应用程序和服务

与大多数 Amazon 功能一样，您通常可按照两种方式来使用角色：在 IAM 控制台中以交互方式使用角色，或通过 Amazon CLI、Tools for Windows PowerShell 或 API 以编程方式使用角色。

- 您账户中的使用 IAM 控制台的 IAM 用户可切换为某个角色，以临时使用该控制台中的角色的权限。该用户放弃自己的原始权限，采用分配给该角色的权限。用户退出角色时，将恢复其原始权限。
- 应用程序或 Amazon 提供的服务（如 Amazon EC2）可以通过请求供角色用来向 Amazon 进行编程请求的临时安全凭证来代入角色。您可以按这种方式使用角色，以便不必为需要访问资源的每个实体共享或维护长期安全凭证（例如，通过创建 IAM 用户）。

Note

本指南以可互换方式使用了切换到角色和代入角色这两个短语。

最简单的角色使用方式是向 IAM 用户授予权限以便切换为您在自己或其他 Amazon 账户中创建的角色。他们可以使用 IAM 控制台轻松地切换角色以使用通常您不希望他们拥有的权限，然后退出角色以交出这些权限。这样有助于防止对敏感资源进行意外访问或修改。

若要对角色进行更复杂的使用（如向应用程序和服务或联合身份外部用户授予访问权限），可以调用 AssumeRole API。该 API 调用返回应用程序可在后续 API 调用中使用的一组临时凭证。使用临时凭证执行的操作只拥有相关联的角色所授予的权限。应用程序不必像控制台中的用户那样“退出”角色；而是应用程序直接停止使用临时凭证并使用原始凭证继续调用。

联合身份用户使用身份提供程序 (IdP) 提供的凭证登录。Amazon 随后向可信 IdP 提供临时证书以传递给用户，以便包含在后续 Amazon 资源请求中。这些证书提供向分配的角色授予的权限。

本部分提供了以下方案的概述：

- 为您拥有的一个 Amazon 账户中的 IAM 用户提供对您拥有的另一个账户中的资源的访问权限 (p. 161)
- 提供对非 Amazon 工作负载的访问权限 (p. 163)
- 为第三方拥有的 Amazon 账户中的 IAM 用户提供访问权限 (p. 164)
- 为 Amazon 提供的服务提供对 Amazon 资源的访问权限 (p. 166)
- 为经过外部身份验证的用户（联合身份验证）提供访问权限 (p. 169)

在您拥有的另一个 Amazon 账户中向 IAM 用户提供访问权限

您可以向 IAM 用户授予权限，以便切换至您 Amazon 账户中的角色，或切换至您拥有的其他 Amazon 账户中定义的角色。

Note

如果要授予对您未拥有或无法控制的账户的访问权限，请参阅本主题后面的[向第三方拥有的 Amazon 账户提供访问权限 \(p. 164\)](#)。

假设您拥有一个企业关键型 Amazon EC2 实例。您可以创建具有这些权限的角色，而不是直接向用户授予终止实例的权限。然后，允许管理员在需要终止实例时切换为该角色。这样，可向实例添加以下几层保护：

- 您必须向用户显式授予担任该角色的权限。
- 用户必须使用 Amazon Web Services Management Console 主动切换为该角色，或使用 Amazon CLI 或 Amazon API 担任角色。
- 您可以向该角色添加多重身份验证 (MFA) 保护，以便只有登录 MFA 设备的用户才能担任该角色。要了解如何配置角色以使担任角色的用户必须先使用多重身份验证 (MFA) 进行身份验证，请参阅[配置受 MFA 保护的 API 访问 \(p. 129\)](#)。

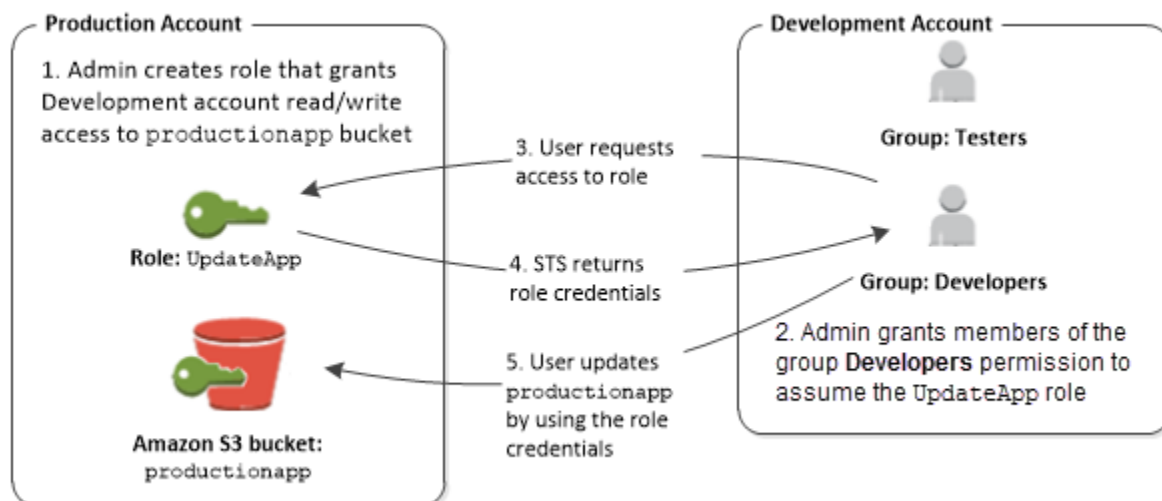
我们建议使用该方法以实施最小权限原则。这意味着，仅限特定任务需要时，才能使用提升的权限。借助角色，您可以帮助防止对敏感环境进行意外更改，如果您将它们与[审核 \(p. 340\)](#)合并以帮助确保仅在需要时才使用角色，这尤其适用。

在您出于此目的创建角色时，可在该角色的信任策略的 Principal 元素中按 ID 指定其用户需要访问权限的账户。随后可以向这些其他账户中的特定用户授予切换到角色的权限。要了解您信任区域之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅[什么是 IAM Access Analyzer ?](#)。

一个账户中的用户可以切换为相同或不同账户中的角色。使用角色过程中，用户只能执行角色允许的操作并且只能访问角色允许的资源；其原始用户权限处于暂停状态。用户退出角色时，恢复原始用户权限。

使用不同的开发和生产账户的示例方案

假设您的组织拥有多个 Amazon 账户以将开发环境与生产环境隔离。开发账户中的用户有时可能需要访问生产账户中的资源。例如，在促进从开发环境到生产环境的更新时，您可能需要进行跨账户访问。尽管您可以在两个账户中工作的用户创建单独的身份（和密码），多个账户的凭证管理还是会为身份管理带来难题。在以下图表中，所有用户都通过开发账户进行管理，但一些开发人员需要对生产账户进行有限访问。开发账户有两个组：测试人员和开发人员，每个组有其自身的策略。



1. 在生产账户中，管理员使用 IAM 在该账户中创建 UpdateApp 角色。在角色中，管理员定义信任策略，该策略将开发账户指定为 Principal，这意味着开发账户中的授权用户可以使用 UpdateApp 角色。管理员还为角色定义权限策略，以指定对名为 productionapp 的 Amazon S3 存储桶的读取和写入权限。

然后，管理员相应信息与需要担任角色的所有人共享。该信息是角色的账号和名称（对于 Amazon 控制台用户）或 Amazon Resource Name (ARN)（用于 Amazon CLI 或 Amazon API 访问）。角色 ARN 类似于 `arn:aws:iam::123456789012:role/UpdateApp`，其中角色名为 `UpdateApp`，角色采用账号 `123456789012` 进行创建。

Note

管理员可以选择配置角色，以便担任角色的用户必须先使用多重身份验证 (MFA) 进行身份验证。有关更多信息，请参阅[配置受 MFA 保护的 API 访问](#) (p. 129)。

- 在开发账户中，管理员向开发人员组的成员授予切换为角色的权限。执行此操作的方法是向开发人员组授予针对 `UpdateApp` 角色调用 Amazon Security Token Service (Amazon STS) `AssumeRole` API 的权限。开发账户中开发人员组的所有 IAM 用户现在都可以切换为生产账户中的 `UpdateApp` 角色。不在开发人员组中的其他用户无权切换为该角色，因此无法访问生产账户中的 S3 存储桶。
- 用户请求切换为角色：
 - Amazon 控制台：用户选择导航栏上的账户名并选择 `Switch Role` (切换角色)。用户指定账户 ID (或别名) 和角色名称。或者，用户可以单击管理员在电子邮件中发送的链接。通过该链接，用户可以转到已填写详细信息的 `Switch Role` 页面。
 - Amazon API/Amazon CLI：开发账户中开发人员组的用户调用 `AssumeRole` 函数以获取 `UpdateApp` 角色的凭证。用户将 `UpdateApp` 角色的 ARN 指定为调用的一部分。如果测试人员组中的用户发出相同请求，请求将失败，因为测试人员没有针对 `UpdateApp` 角色 ARN 调用 `AssumeRole` 的权限。
- Amazon STS 返回临时凭证：
 - Amazon 控制台：Amazon STS 使用角色的信任策略来验证请求，以确保请求来自受信任实体（即开发账户）。验证完成后，Amazon STS 向 Amazon 控制台返回[临时安全凭证](#)。
 - API/CLI：Amazon STS 根据角色的信任策略来验证请求，以确保请求来自受信任实体（即开发账户）。验证完成后，Amazon STS 向应用程序返回[临时安全凭证](#)。
- 临时凭证允许访问 Amazon 资源：
 - Amazon 控制台：Amazon 控制台在所有后续控制台操作中代表用户使用临时凭证，在本例中是用于读取和写入 `productionapp` 存储桶。该控制台无法访问生产账户中的任何其他资源。用户退出角色时，用户的权限恢复为切换为角色之前所拥有的原始权限。
 - API/CLI：应用程序使用临时安全凭证更新 `productionapp` 存储桶。应用程序只能使用临时安全凭证读取和写入 `productionapp` 存储桶，无法访问生产账户的任何其他资源。应用程序不必退出角色，只需在后续 API 调用中停止使用临时凭证并使用原始凭证。

更多信息

有关更多信息，请参阅下列内容：

- [IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限](#) (p. 31)

为非 Amazon 工作负载提供访问权限

[IAM 角色](#) (p. 158) 是 Amazon Identity and Access Management (IAM) 中分配[权限](#) (p. 355) 的对象。当您担任的角色 (p. 236) 使用 IAM 身份或来自 Amazon 外部的身份时，它会为您提供角色会话的临时安全凭证。您的工作负载可能在数据中心或其他 Amazon 外部需要访问您 Amazon 资源的基础设施中运行。除了创建、分发和管理长期访问密钥之外，您可以使用 Amazon Identity and Access Management Roles Anywhere (IAM Roles Anywhere) 来验证您的非 Amazon 工作负载。IAM Roles Anywhere 使用您的证书颁发机构 (CA) 颁发的 X.509 证书对身份进行验证，并安全地使用 IAM 角色提供的临时证书提供对 Amazon Web Services 的访问权限。

要使用 IAM Roles Anywhere，请设置使用 [Amazon Private Certificate Authority](#) 的 CA 或者使用来自您自己的 PKI 基础设施的 CA。设置 CA 后，您可以在 IAM Roles Anywhere 中创建一个名为信任锚点的对象，以在 IAM 角色 Anywhere 和您的 CA 之间建立信任以进行身份验证。然后，您可以配置现有的 IAM 角色，或

者创建信任 IAM Roles Anywhere 服务的新角色。当您的非 Amazon 工作负载使用信任锚点通过 IAM Roles Anywhere 进行身份验证时，它们可以获取您的 IAM 角色的临时凭证来访问您的 Amazon 资源。

向第三方拥有的 Amazon 账户提供访问权限

当第三方请求访问您的组织的 Amazon 资源时，您可以使用角色向他们委派访问权限。例如，第三方可能某种服务来管理您的 Amazon 资源。您可以通过 IAM 角色授予第三方访问您的 Amazon 资源的权限，而无需共享您的 Amazon 安全凭证。而第三方则可以通过担任您在您的 Amazon 账户中创建的角色来访问您的 Amazon 资源。要了解您信任区域之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅[什么是 IAM Access Analyzer？](#)。

为了创建他们可以代入的角色，第三方必须为您提供以下信息：

- 第三方的 Amazon 账户 ID。为角色定义信任策略时，可将其 Amazon 账户 ID 指定为主体。
- 与角色唯一关联的外部 ID。外部 ID 可以是您和第三方已知的任何秘密标识符。例如，您可以使用您与该第三方之间的发票 ID，但不要使用能被猜到的内容，如第三方的名称或电话号码。为角色定义信任策略时，必须指定该 ID。第三方在代入角色时必须提供该 ID。有关外部 ID 的更多信息，请参阅[如何在向第三方授予对 Amazon 资源的访问权时使用外部 ID \(p. 164\)](#)。
- 第三方为使用您的 Amazon 资源而请求获得的权限。在定义角色的权限策略时，您必须指定这些权限。这个策略定义了他们可以执行哪些操作以及可以访问哪些资源。

创建完角色后，您必须向第三方提供该角色的 Amazon Resource Name (ARN)。他们需要使用您的角色的 ARN 来代入该角色。

有关通过创建角色向第三方委派访问权限的详细信息，请参阅[如何在向第三方授予对 Amazon 资源的访问权时使用外部 ID \(p. 164\)](#)。

Important

当您授予第三方访问 Amazon 资源的权限时，他们可以访问您在该策略中指定的任何资源。系统会为您发送他们使用资源的记录。请确保适当限制他们对资源的使用。

如何在向第三方授予对 Amazon 资源的访问权时使用外部 ID

有时，您需要向第三方提供对您的 Amazon 资源的访问权（提供访问权）。此方案的一个重要方面是外部 ID，外部 ID 是一条可选信息，您可在 IAM 角色信任策略中使用该信息来指定谁能代入该角色。

Important

Amazon 不会将该外部 ID 作为秘密信息。您在 Amazon 中创建访问密钥对或密码这类秘密信息后，您无法再次查看它们。有权查看角色的任何人都可以看到该角色的外部 ID。

在您支持多个客户拥有不同 Amazon 账户的多租户环境中，我们建议每个 Amazon 账户使用一个外部 ID。此 ID 应该是第三方生成的随机字符串。

要在代入角色时需要提供外部 ID 的第三方，请使用您选择的外部 ID 来更新角色的信任策略。

要在代入角色时提供外部 ID，请使用 Amazon CLI 或 Amazon API 来代入该角色。有关更多信息，请参阅 STS [AssumeRole](#) API 操作或 STS [assume-role](#) CLI 操作。

例如，假设您决定聘请一家名为 Example Corp 的第三方公司来监控您的 Amazon 账户并帮助优化成本。为跟踪您的日常开支，Example Corp 需要访问您的 Amazon 资源。Example Corp 也可监控其他客户的很多其他 Amazon 账户。

请勿向 Example Corp 提供对您的 Amazon 账户中的 IAM 用户及其长期凭证的访问权限。请使用 IAM 角色及其临时安全凭证。IAM 角色提供了一种机制，可允许第三方访问您的 Amazon 资源而无需共享长期凭证（例如，IAM 用户的访问密钥）。

您可使用 IAM 角色在您的 Amazon 账户和 Example Corp 账户之间建立信任关系。建立这种关系后，Example Corp 账户的成员可以调用 Amazon Security Token Service [AssumeRole](#) API 以获取临时安全凭证。然后，Example Corp 成员可以使用这些凭证访问您的账户中的 Amazon 资源。

Note

有关可调用来获取临时安全凭证的 AssumeRole 和其他 Amazon API 操作的更多信息，请参阅[请求临时安全凭证 \(p. 302\)](#)。

以下是此方案的更多详细信息：

1. 您聘请了 Example Corp，此公司将为您创建唯一客户标识符。他们为您提供此唯一客户 ID 及其 Amazon 账号。您需要此信息来在下一步中创建 IAM 角色。

Note

Example Corp 可使用其想用于 ExternalId 的任何字符串值，只要该值对于每个客户来说都是唯一的。该值可以是客户账号，甚至可以是一个随机字符串，只要没有两个客户拥有相同的值即可。该值不是“机密的”。Example Corp 必须向每个客户提供 ExternalId 值。关键的一点是，该值必须由 Example Corp 而不是其客户生成。

2. 您登录到 Amazon 并创建 IAM 角色，该角色可向 Example Corp 提供对您的资源的访问权。与任何 IAM 角色类似，该角色具有两个策略：权限策略和信任策略。该角色的信任策略规定谁可担任该角色。在我们的示例方案中，该策略将 Example Corp 的 Amazon 账号指定为 Principal。这允许来自此账户的身份担任该角色。此外，将 [Condition](#) 元素添加到信任策略。此 Condition 测试 ExternalId 上下文密钥，以确保它与 Example Corp 的唯一客户 ID 一致。例如：

```
"Principal": {"AWS": "Example Corp's Amazon Account ID"},
"Condition": {"StringEquals": {"sts:ExternalId": "Unique ID Assigned by Example Corp"}}
```

3. 该角色的权限策略指定该角色允许某个人执行哪些操作。例如，您可以规定该角色允许某人仅管理您的 Amazon EC2 和 Amazon RDS 资源，但不能管理您的 IAM 用户或组。在我们的示例方案中，您使用权限策略为 Example Corp 授予您的账户中的所有资源的只读访问权限。
4. 创建完角色后，您向 Example Corp 提供该角色的 Amazon Resource Name (ARN)。
5. 当 Example Corp 需要访问您的 Amazon 资源时，该公司的某人可调用 Amazon `sts:AssumeRole` API。该调用包括要担任的角色的 ARN 以及与其客户 ID 对应的 ExternalId 参数。

如果发出请求的人使用的是 Example Corp 的 Amazon 账户，并且角色 ARN 和外部 ID 是正确的，则请求成功。然后，该请求提供临时安全凭证，Example Corp 可使用这些凭证访问您的角色允许访问的 Amazon 资源。

换句话说，当角色策略包括外部 ID 时，任何需要担任该角色的人都必须是该角色中的主体，还必须包括正确的外部 ID。

为什么要使用外部 ID？

抽象地说，外部 ID 允许正担任该角色的用户断言其运行于的环境。它还账户所有者提供一种方法来允许仅在特定情况下担任该角色。外部 ID 的主要功能是解决并防止 [混淆代理人问题 \(p. 166\)](#)。

我何时应使用外部 ID？

在以下情况下使用外部 ID：

- 您是 Amazon 账户所有者并且已为将访问其他 Amazon 账户以及您的账户的第三方配置角色。您应要求第三方提供其在担任您的角色时包含的外部 ID。然后，在您角色的信任策略中检查该外部 ID。这样做可确保外部方仅在代表您执行操作时才能担任您的角色。
- 在前述情况下，您代表不同客户 (如 Example Corp) 担任角色。您应该为每个客户分配一个唯一的外部 ID 并指导他们将该外部 ID 添加到其角色的信任策略。然后，您必须确保在担任角色的请求中始终包含正确的外部 ID。

您可能已为您的每个客户提供一个唯一标识符，而且此唯一 ID 足以用作外部 ID。该外部 ID 不是您要明确创建或分别跟踪所需的特殊值 (仅用于此目的)。

您应始终在您的 AssumeRole API 调用中指定外部 ID。此外，在客户为您提供角色 ARN 时，请测试是否能在带有/不带正确外部 ID 的情况下担任该角色。如果可在没有正确外部 ID 的情况下担任角色，则不要在你的系统中存储该客户的角色 ARN。等待该客户将角色信任策略更新为要求提供正确的外部 ID。这样一来，您帮助您的客户执行了正确的操作，并帮助您和客户避免了混淆代理人问题。

向 Amazon 服务提供访问权限

很多 Amazon 服务要求您利用角色控制该服务可以访问的内容。由一项服务担任、代表您执行操作的角色称为 [服务角色](#) (p. 159)。如果按服务角色适用的特定目的进行分类，可分为 [EC2 实例的服务角色](#) (p. 159)，或 [服务相关角色](#) (p. 159)。请参阅每种服务的 [Amazon 文档](#)，以了解它是否使用角色以及如何分配角色以供服务使用。

有关创建角色以向 Amazon 提供的服务委派访问权限的详细信息，请参阅 [创建向 Amazon 服务委派权限的角色](#) (p. 219)。

混淆代理人问题

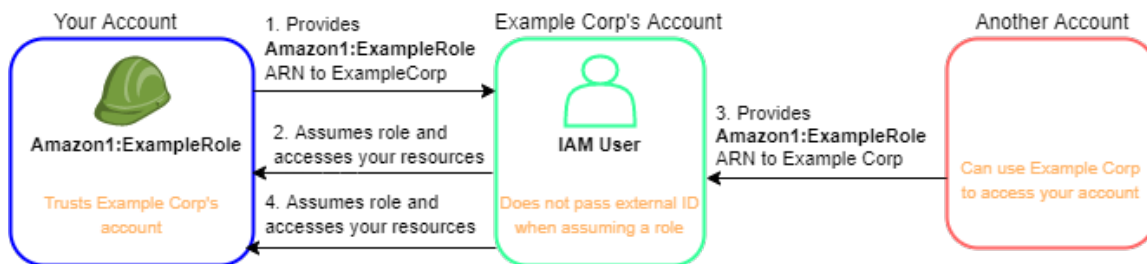
混淆代理问题是一个安全问题，即没有执行操作权限的实体可能会迫使更具权限的实体执行该操作。为了防止这种情况，如果您为账户中的资源提供第三方（称为跨账户）或其他 Amazon 服务（称为跨服务）的访问权限，则 Amazon 会提供用于保护您账户的工具。

有时，您需要向第三方提供对您的 Amazon 资源的访问权 (提供访问权)。例如，假设您决定聘请一家名为 Example Corp 的第三方公司来监控您的 Amazon 账户并帮助优化成本。为跟踪您的日常开支，Example Corp 需要访问您的 Amazon 资源。Example Corp 也可监控其他客户的很多其他 Amazon 账户。您可使用 IAM 角色在您的 Amazon 账户和 Example Corp 账户之间建立信任关系。此方案的一个重要方面是外部 ID，外部 ID 是一条可选信息，您可在 IAM 角色信任策略中使用该信息来指定谁能代入该角色。外部 ID 的主要功能是解决并防止混淆代理人问题。

在 Amazon 中，跨服务模拟可能会导致混淆代理问题。一个服务（呼叫服务）调用另一项服务（所谓的模拟服务）时，可能会发生跨服务模拟。可以操纵调用服务以使用其权限对另一个客户的资源进行操作，否则该服务不应有访问权限。

跨账户混淆代理人预防

下图阐明了跨账户混淆代理人问题。



此场景假定：

- Amazon1 是您的 Amazon 账户 ID。
- Amazon1:ExampleRole 是您账户中的一个角色。该角色的信任策略通过将 Example Corp 的 Amazon 账户指定为可担任该角色的账户来信任 Example Corp。

将发生以下情况：

1. 在您开始使用 Example Corp 的服务时，您将向 Example Corp 提供 Amazon1:ExampleRole 的 ARN。
2. Example Corp 使用该角色 ARN 获取临时安全凭证以访问您的 Amazon 账户中的资源。这样一来，您将信任 Example Corp 作为可代表您执行操作的“代理人”。
3. 另一个 Amazon 客户也开始使用 Example Corp 的服务，而且此客户还为 Example Corp 提供了 Amazon1:ExampleRole 的 ARN 以供其使用。另一个客户可能已了解或猜到已不是机密信息的 Amazon1:ExampleRole。
4. 当另一个客户要求 Example Corp 访问（它声称的）其账户中的 Amazon 资源时，Example Corp 使用 Amazon1:ExampleRole 访问您的账户中的资源。

这就是其他客户可对您的资源进行未授权访问的方式。由于此客户能够诱使 Example Corp 无意中操作您的资源，因此 Example Corp 现在是一个“混淆代理人”。

Example Corp 可以通过在角色信任策略中加入 ExternalId 条件检查来解决混淆的代理问题。Example Corp 为每个客户生成唯一的 ExternalId 值，并在其请求中使用该值来承担该角色。在 Example Corp 的客户中，ExternalId 值必须是独一无二的，并由 Example Corp 而不是其客户控制。这就是您从 Example Corp 获取该 ID 且不能自行提供该 ID 的原因。这可以防止 Example Corp 成为一个混淆代理人，以及授予对另一个账户的 Amazon 资源的访问权。

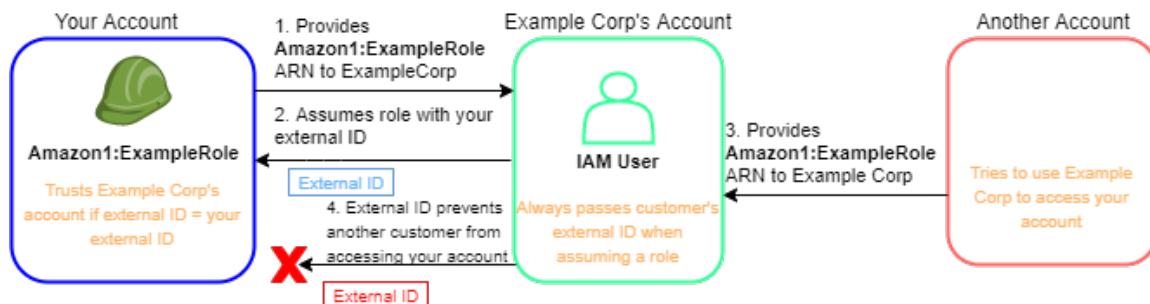
在我们的方案中，假设 Example Corp 为您提供的唯一标识符是 12345，而为另一个客户提供的标识符是 67890。这些标识符已针对此方案进行简化。通常，这些标识符为 GUID。假定这些标识符在 Example Corp 的客户之间是唯一的，它们将是用于外部 ID 的有意义的值。

Example Corp 将向您提供外部 ID 值 12345。然后，您必须将一个 Condition 元素添加到角色的信任策略，该策略要求 sts:ExternalId 值为 12345，如下所示：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "AWS": "Example Corp's Amazon Account ID"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "sts:ExternalId": "12345"
      }
    }
  }
}
```

此策略中的 Condition 元素允许 Example Corp 仅在 AssumeRole API 调用包括外部 ID 值 12345 时代入该角色。Example Corp 确保，只要它代表客户担任角色，就会始终在 AssumeRole 调用中包括客户的外部 ID。即使另一个客户向 Example Corp 提供您的 ARN，也无法控制 Example Corp 包括在其发送给 Amazon 的请求中的外部 ID。这有助于防止未经授权的客户获取对您的资源的访问权限。

下图阐明了此过程。



1. 和之前一样，在您开始使用 Example Corp 的服务时，您将向 Example Corp 提供 Amazon1:ExampleRole 的 ARN。
2. 在 Example Corp 使用该角色 ARN 来代入角色 Amazon1:ExampleRole 时，Example Corp 将在 AssumeRole API 调用中包含您的外部 ID (12345)。该外部 ID 与角色的信任策略匹配，因此 AssumeRole API 调用将成功，并且 Example Corp 将获取用于访问您的 Amazon 账户中的资源的临时安全凭证。
3. 另一个 Amazon 客户也开始使用 Example Corp 的服务，而且如之前一样，此客户还为 Example Corp 提供了 Amazon1:ExampleRole 的 ARN 以供其使用。
4. 但这一次，在 Example Corp 尝试代入角色 Amazon1:ExampleRole 时，它提供了与另一个客户关联的外部 ID (67890)。另一个客户无法更改此外部 ID。Example Corp 这样做是因为另一个客户请求使用该角色，因此 67890 表示 Example Corp 正在其中操作的环境。由于您已将具有您自己的外部 ID (12345) 的条件添加到 Amazon1:ExampleRole 的信任策略，因此 AssumeRole API 调用将失败。而且将阻止另一个客户对您账户中的资源进行未经授权的访问 (由图中的红色“X”表示)。

外部 ID 有助于防止任何其他客户欺骗 Example Corp 不知不觉地访问您的资源。

防止跨服务混淆代理

我们建议在基于资源的策略中使用 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文密钥，以限制对特定资源的权限。如果您只希望将一个资源与跨服务访问相关联，请使用 `aws:SourceArn`。如果您想允许该账户中的任何资源与跨服务使用操作相关联，请使用 `aws:SourceAccount`。

防止代理问题的最有效方法是在基于资源的策略中使用 `aws:SourceArn` 全局条件上下文密钥和资源的完整 ARN。如果您不知道资源的完整 ARN，或者您正在指定多个资源，请针对 ARN 未知部分使用带有通配符 (*) 的 `aws:SourceArn` 全局条件上下文密钥。例如：`arn:aws:service:*:123456789012:*`。

如果 `aws:SourceArn` 值不包含账户 ID，例如 Amazon S3 存储桶 ARN，您必须使用两个全局条件上下文密钥来限制权限。

针对 Amazon Security Token Service 的防止跨服务混淆代理

许多 Amazon 服务要求您使用角色来允许该服务代表您访问其他服务的资源。由一项服务担任、代表您执行操作的角色称为 **服务角色** (p. 159)。角色需要两个策略：一个角色信任策略，用于指定允许代入角色的主体，另一个权限策略用于指定可以对角色执行的操作。角色信任策略是 IAM 中唯一基于资源的策略类型。其他 Amazon 服务采用基于资源的策略，例如 Amazon S3 存储桶策略。

当服务代表您担任角色时，必须允许服务主体在角色信任策略中执行 `sts:AssumeRole` 操作。当服务调用 `sts:AssumeRole` 时，Amazon STS 返回一组临时安全凭证，服务主体使用该凭证访问角色的权限策略所允许的资源。当服务在您的账户中担任角色时，您的角色信任策略中可以包含 `aws:SourceAccount` 和 `aws:SourceArn` 全局条件上下文密钥，以将对角色的访问限制为仅由预期资源生成的请求。

例如，在 Amazon Systems Manager Incident Manager 中，您必须选择一个角色，以允许事件管理器代表您运行 Systems Manager 自动化文档。自动化文档可以包括由 CloudWatch 警报或 EventBridge 事件发起的事件的自动响应计划。在以下角色信任策略示例中，您可以使用 `aws:SourceArn` 条件密钥，用于根据事件记录 ARN 限制对服务角色的访问。只有从响应计划资源 `myresponseplan` 创建的事件记录才能使用此角色。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "Service": "ssm-incidents.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:ssm-incidents:*:111122223333:incident-record/myresponseplan/*"
      }
    }
  }
}
```



```
}  
}  
}
```

向经过外部身份验证的用户（联合身份验证）提供访问权限

您的用户可能已具有 Amazon 外部的身份，如在公司目录中。如果这些用户需要使用 Amazon 资源（或使用访问这些资源的应用程序），则这些用户也需要 Amazon 安全凭证。您可以使用 IAM 角色为身份通过您的企业或第三方身份提供程序 (IdP) 进行联合证明的用户指定权限。

通过 Amazon Cognito 联合移动或基于 Web 的应用程序的用户

如果创建访问 Amazon 资源的移动或基于 Web 的应用程序，则应用程序需要安全凭证才能向 Amazon 进行编程请求。对于大多数移动应用程序情况，我们建议您使用 [Amazon Cognito](#)。您可以将此服务与 [Amazon Mobile SDK for iOS](#) 和 [Amazon Mobile SDK for Android and Fire OS](#) 搭配使用，为用户创建唯一的身份，并对其进行身份验证，以使其安全地访问您的 Amazon 资源。Amazon Cognito 支持与下一个部分中所列的那些提供商相同的身份提供程序，而且还支持已经过[开发人员验证的身份](#)和未经身份验证的（来宾）访问。Amazon Cognito 还提供用于同步用户数据的 API 操作，因此，无论用户在设备间怎样转移，其数据总能得以保留。有关更多信息，请参阅[将 Amazon Cognito 用于移动应用程序 \(p. 171\)](#)。

使用公共身份服务提供商或 OpenID Connect 联合身份用户

在移动和基于 Web 的场景下尽可能使用 Amazon Cognito。Amazon Cognito 为您提供公共身份提供程序服务的大部分幕后工作。它与相同的第三方服务协作，也支持匿名登录。不过，对于更高级的方案，您可以直接使用第三方服务，例如 Login with Amazon、Facebook、Google 或任何与 OpenID Connect (OIDC) 兼容的 IdP。有关通过这些服务之一使用 Web 联合身份验证的更多信息，请参阅[关于 Web 联合身份验证 \(p. 171\)](#)。

通过 SAML 2.0 联合身份用户

如果您的组织已使用支持 SAML 2.0 (安全断言标记语言 2.0) 的身份提供程序软件包，您可以在作为身份提供程序 (IdP) 的您的组织和作为服务提供商的 Amazon 之间建立信任。随后可以使用 SAML 为您的用户提供对 Amazon Web Services Management Console 的联合单一登录 (SSO)，或是用于调用 Amazon API 操作的联合访问。例如，如果您的公司使用 Microsoft Active Directory 和 Active Directory 联合身份验证服务，则您可以使用 SAML 2.0 进行联合。有关使用 SAML 2.0 对用户进行联合身份验证的更多信息，请参阅[关于基于 SAML 2.0 的联合身份验证 \(p. 176\)](#)。

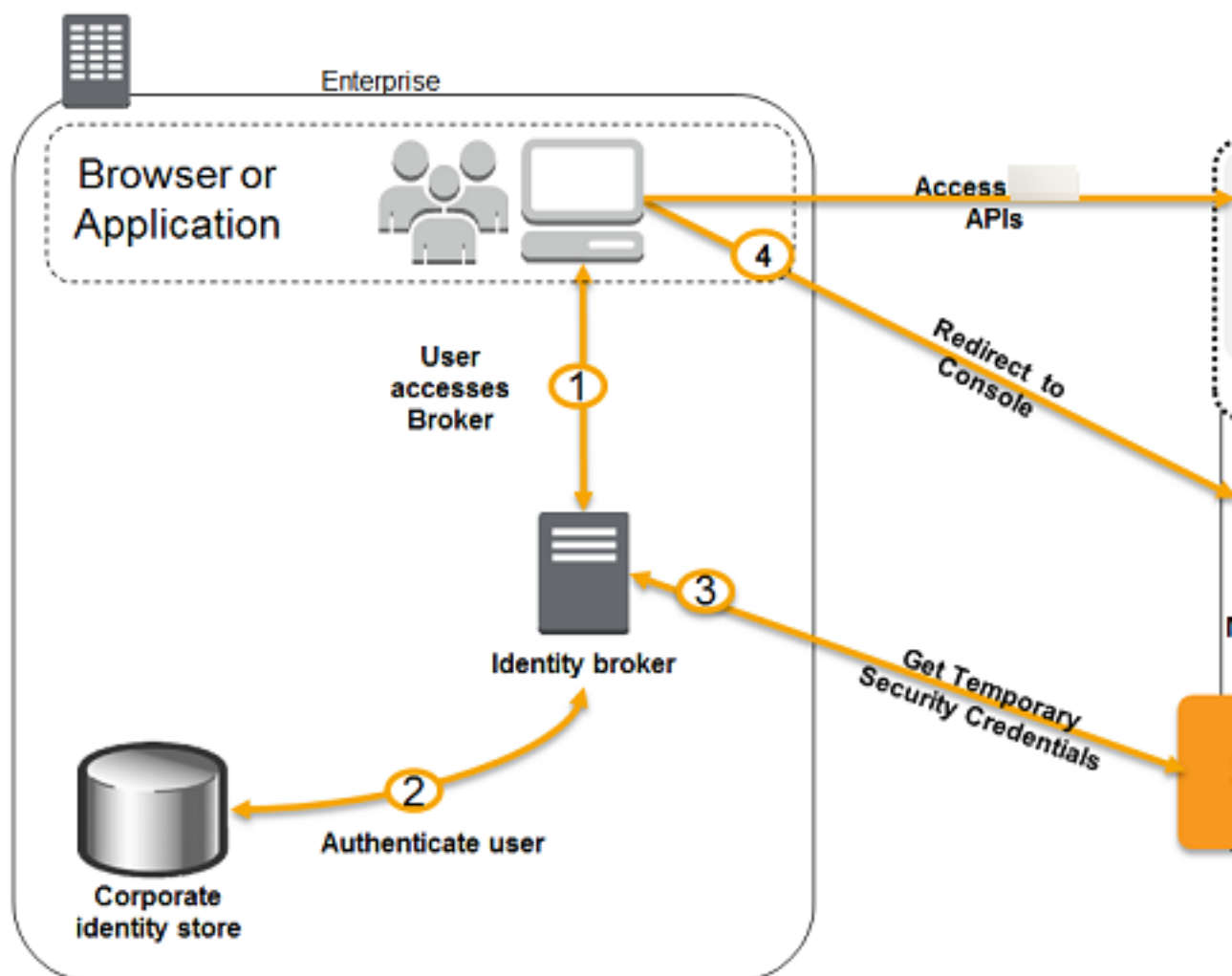
通过创建自定义身份代理应用程序来联合身份用户

如果您的身份存储与 SAML 2.0 不兼容，则可构建自定义身份代理应用程序来执行类似功能。代理应用程序将对用户进行身份验证，从 Amazon 为用户请求临时安全凭证，然后将凭证提供给用户以便访问 Amazon 资源。

例如，Example Corp. 有很多员工需要运行访问公司 Amazon 资源的内部应用程序。这些员工在公司的身份与验证系统中已有身份，并且 Example Corp. 不希望为每位公司员工都创建一个单独的 IAM 用户。

Bob 是 Example Corp. 的一名开发人员。为使 Example Corp. 的内部应用程序可访问公司的 Amazon 资源，Bob 开发了一个自定义身份代理应用程序。该应用程序确认员工登录到现有的 Example Corp. 身份与验证系统，而该系统可能使用 LDAP、Active Directory 或其他系统。然后，身份代理应用程序获取员工的临时安全凭证。此场景与以前的某个场景（一个使用自定义身份验证系统的移动应用程序）类似，但需要访问 Amazon 资源的应用程序全都运行在企业网络中，并且公司现有一个身份验证系统。

为了获取临时安全凭证，该身份代理应用程序调用 `AssumeRole` 或 `GetFederationToken` 以获取临时安全凭证，具体取决于 Bob 要如何管理用户的策略以及临时凭证应何时到期。（有关这些 API 操作之间的差异的更多信息，请参阅[IAM 临时安全凭证 \(p. 300\)](#)和[控制临时安全凭证的权限 \(p. 313\)](#)。）该调用返回由 Amazon 访问密钥 ID、秘密访问密钥和会话令牌牌组成的临时安全凭证。该身份代理应用程序向内部公司应用程序提供这些临时安全凭证。然后，该应用程序可使用这些临时凭证直接调用 Amazon。该应用程序将证书缓存至其到期，然后请求新的一组临时证书。下图举例说明此场景。



此方案有以下属性：

- 该身份代理应用程序具有访问 IAM 的令牌服务 (STS) API 以创建临时安全凭证的权限。
- 该身份代理应用程序可确认员工在现有的身份验证系统中经过身份验证。
- 用户可获得一个临时 URL，通过它可访问 Amazon 管理控制台 (称为单点登录)。

若要查看类似于此场景中描述的身份代理应用程序的示例应用程序，请转到 Amazon 示例代码和库中的 [Active Directory 联合身份验证示例应用程序的使用案例](#)。有关创建临时安全凭证的信息，请参阅 [请求临时安全凭证 \(p. 302\)](#)。有关获取对 Amazon 管理控制台的访问权限的联合身份用户的更多信息，请参阅 [使用 SAML 2.0 联合身份用户能够访问 Amazon Web Services Management Console \(p. 199\)](#)。

身份提供程序和联合身份验证

如果您已管理 Amazon 外部的用户身份，则可以使用 IAM 身份提供程序，而不必在 Amazon 账户中创建 IAM 用户。利用身份提供程序 (IdP)，您可以管理 Amazon 外部的用户身份，并向这些外部用户身份授予使用您账户中的 Amazon 资源的权限。如果您的组织已有自己的身份系统（如企业用户目录），这将十分有用。如果要创建需要访问 Amazon 资源的移动应用程序或 Web 应用程序，这也十分有用。

使用 IAM 身份提供程序时，您不必创建自定义登录代码或管理自己的用户身份。IdP 将向您提供它们。您的外部用户通过已知的 IdP（例如 Login with Amazon、Facebook 或 Google）进行登录。您可以向这些外部身份授予使用您的账户中的 Amazon 资源的权限。IAM 身份提供程序可帮助您确保 Amazon 账户的安全，因为您不必在应用程序中分配或嵌入长期安全凭证（如访问密钥）。

要使用 IdP，您需要创建 IAM 身份提供程序实体以在您的 Amazon 账户和 IdP 之间建立信任关系。IAM 支持与 [OpenID Connect \(OIDC\)](#) 或者 [SAML 2.0 \(Security Assertion Markup Language 2.0\)](#) 兼容的 IdPs。有关通过 Amazon 使用这些 IdP 之一的更多信息，请参阅以下部分：

- [关于 Web 联合身份验证 \(p. 171\)](#)
- [关于基于 SAML 2.0 的联合身份验证 \(p. 176\)](#)

有关创建 IAM 身份提供程序实体以在兼容的 IdP 和 Amazon 之间建立信任关系的详细信息，请参阅 [创建 IAM 身份提供程序 \(p. 180\)](#)。

关于 Web 联合身份验证

假设您要创建一个访问 Amazon 资源的移动应用程序，如在移动设备上运行并使用 Amazon S3 和 DynamoDB 存储玩家和分数信息的游戏。

当您编写此类应用程序时，将向必须使用 Amazon 访问密钥进行签名的 Amazon 服务提出请求。但是，我们强烈建议您不要对用户下载到设备的应用程序嵌入或分配长期 Amazon 凭证，即使在加密存储中也是如此。相反，应构建您的应用程序，让其在需要使用 AmazonWeb 联合身份验证时动态请求临时安全凭证。提供的临时凭证映射到的 Amazon 角色只拥有执行移动应用程序所需任务的必要权限。

利用 Web 联合身份验证，您不需要创建自定义登录代码或管理自己的用户身份。相反，应用程序的用户可以使用知名的外部身份提供程序（例如，Login with Amazon、Facebook、Google 或任何其他 [OpenID Connect \(OIDC\)](#) 兼容的 IdP）登录。他们可以接收身份验证令牌，然后用该令牌交换 Amazon 中的临时安全凭证，这些凭证映射到有权使用您 Amazon 账户中的资源的 IAM 角色。使用 IdP 可帮助您确保 Amazon 账户的安全，因为您不必随应用程序嵌入和分配长期安全凭证。

对于大多数方案，我们建议您使用 [Amazon Cognito](#)，因为它可充当身份代理并为您完成许多联合工作。有关详细信息，请参阅以下部分：[将 Amazon Cognito 用于移动应用程序 \(p. 171\)](#)。

如果您未使用 Amazon Cognito，则必须编写与 Web IdP（例如 Facebook）交互的代码，然后调用 `AssumeRoleWithWebIdentity` API 以使用您从这些 IdP 处获得的身份验证令牌交换 Amazon 临时安全凭证。如果您现有的应用程序已采用这种方法，则您可以继续使用它。

主题

- [将 Amazon Cognito 用于移动应用程序 \(p. 171\)](#)
- [对移动应用程序使用 Web 联合身份验证 API 操作 \(p. 173\)](#)
- [使用 Web 联合身份验证识别用户的身份 \(p. 174\)](#)
- [有关 Web 联合身份验证的其他资源 \(p. 176\)](#)

将 Amazon Cognito 用于移动应用程序

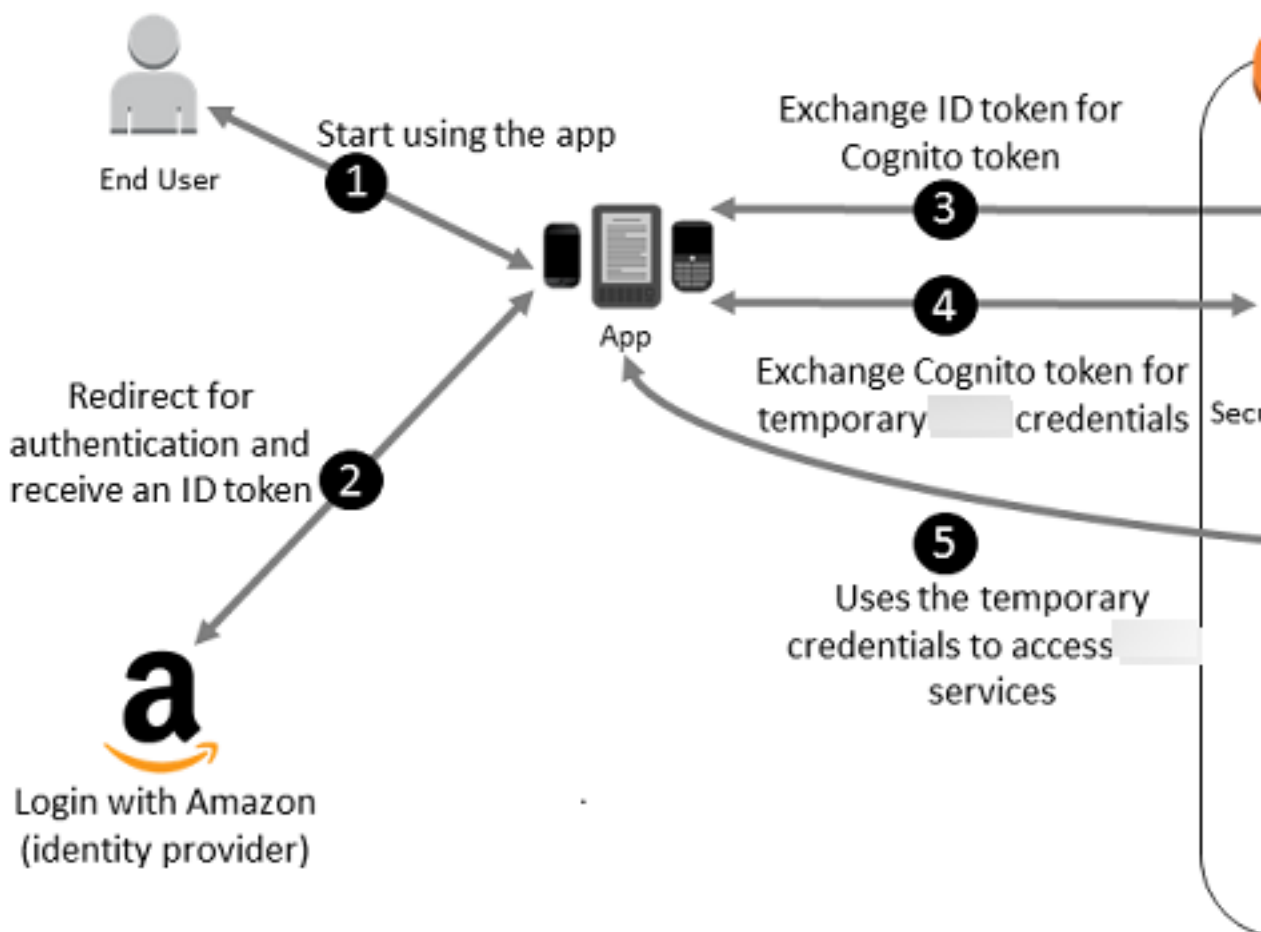
使用 Web 联合身份验证的首选方式是使用 [Amazon Cognito](#)。例如，开发人员 Adele 正在制作一款用于移动设备的游戏，其中将分数和个人资料等用户数据存储在 Amazon S3 和 Amazon DynamoDB 中。Adele 还可将此数据存储在本地设备上，并使用 Amazon Cognito 来跨设备保持数据的同步。她知道，出于安全性和维护的原因，不应随游戏分配长期 Amazon 安全凭证。她还知道，这个游戏可能有大量用户。出于所有这些原因，她不想在 IAM 中为每个玩家都新建用户身份，而是将游戏制作成用户可使用其已通过知名的外部身份提供程序 (IdP)（如 Login with Amazon、Facebook、Google 或任何 OpenID Connect (OIDC) 兼容的 IdP）建立的身份进行登录。她的游戏可利用其中某个提供商的身份验证机制验证用户的身份。

为使该移动应用程序可访问她的 Amazon 资源，Adele 首先向她选择的 IdP 注册一个开发人员 ID。她还按其中某个提供商的要求配置该应用程序。在她的包含该游戏的 Amazon S3 存储桶和 DynamoDB 表的 Amazon

账户中，Adele 使用 Amazon Cognito 创建了精确定义该游戏所需权限的 IAM 角色。如果她使用的是 OIDC IdP，她还可创建 IAM OIDC 身份提供商实体以在其 Amazon Web Services 账户中的 [Amazon Cognito 身份池](#) 和该 IdP 之间建立信任关系。

在应用程序的代码中，Adele 调用其先前配置的 IdP 的登录接口。IdP 处理让用户登录的所有详细信息，然后应用程序从提供商那里获得 OAuth 访问令牌或 OIDC ID 令牌。Adele 的应用程序可使用此身份验证信息换取一组临时安全凭证（包括 Amazon 访问密钥 ID、私有访问密钥和会话令牌）。然后，应用程序可以使用这些凭证访问 Amazon 提供的 Web 服务。该应用程序仅获得在其担任的角色中定义的权限。

下图以 Login with Amazon 作为 IdP，展示此过程运行方式的简化流程。对于步骤 2，该应用程序还可以使用 Facebook、Google 或任何与 OIDC 兼容的 IdP，但此处不进行说明。



1. 客户在移动设备上启动您的应用程序。应用程序要求用户登录。
2. 应用程序使用 Login with Amazon 资源接受用户凭证。
3. 应用程序使用 Amazon Cognito API 操作 `GetId` 和 `GetCredentialsForIdentity` 将 Login with Amazon ID 令牌交换成 Amazon Cognito 令牌。Amazon Cognito 已配置为信任 Login with Amazon 项目，它会生成一个令牌，用于与 Amazon STS 交换临时会话凭证。
4. 该应用程序从 Amazon Cognito 接收临时安全凭证。您的应用程序还可使用 Amazon Cognito 中的基本（经典）工作流程，通过 `AssumeRoleWithWebIdentity` 从 Amazon STS 中检索令牌。有关更多信息，请参阅《Amazon Cognito 开发人员指南》中的 [身份池（联合身份）身份验证流程](#)。

5. 应用程序可使用临时安全凭证访问应用程序运行所需的任何 Amazon 资源。与临时安全凭证关联的角色及其分配的策略将决定可访问的资源。

使用以下过程将您的应用程序配置为使用 Amazon Cognito 对用户进行身份验证，并向您的应用程序授予对 Amazon 资源的访问权限。有关完成此操作的具体步骤，请参阅 Amazon Cognito 的相关文档。

1. (可选) 使用 Login with Amazon、Facebook、Google 或任何其他与 OpenID Connect (OIDC) 兼容的 IdP 注册为开发人员，并使用这些提供商配置一个或多个应用程序。该步骤是可选的，因为 Amazon Cognito 还支持您的用户进行未经身份验证的 (来宾) 访问。
2. 转到 [Amazon Cognito 控制台](#) [Amazon Web Services Management Console](#)。使用 Amazon Cognito 向导创建一个身份池，身份池是一种供 Amazon Cognito 保留为您的应用程序组织的终端用户身份的容器。您可以在应用程序间共享身份池。在设置身份池时，Amazon Cognito 将创建一个或两个定义 Amazon Cognito 用户的权限的 IAM 角色 (一个角色针对经过身份验证的身份，另一个角色针对未经身份验证的“来宾”身份)。
3. 将 [Amazon Amplify](#) 与您的应用程序集成，然后导入使用 Amazon Cognito 所需的文件。
4. 创建一个 Amazon Cognito 证书提供程序实例，并传递身份池 ID、您的 Amazon 账号，以及与身份池关联的角色的 Amazon Resource Name (ARN)。Amazon Web Services Management Console 中的 Amazon Cognito 向导提供了可帮助您入门的示例代码。
5. 当您的应用程序访问 Amazon 资源时，可将该凭证提供程序实例传递给客户端对象，从而将临时安全凭证传递给该客户端。证书的权限基于您之前定义的角色。

有关更多信息，请参阅下列内容：

- [Amazon Amplify 框架文档中的登录 \(Android \)](#)。
- [Amazon Amplify 框架文档中的登录 \(iOS \)](#)。

对移动应用程序使用 Web 联合身份验证 API 操作

为获得最佳效果，请在几乎所有 Web 身份联合身份验证场景中将 Amazon Cognito 用作身份代理。Amazon Cognito 易于使用，并提供了额外功能，如匿名 (未经身份验证的) 访问，以及跨设备和提供商同步用户数据。但是，如果您已通过手动调用 `AssumeRoleWithWebIdentity` API 创建了使用 Web 联合身份验证的应用程序，也可继续使用它，您的应用程序仍能正常工作。

Note

为帮助您了解 Web 联合身份验证的工作方式，可以使用 [Web Identity Federation Playground](#)。该交互式网站演练如下身份验证过程：通过 Login with Amazon、Facebook 或 Google 获得临时安全凭证，然后使用这些凭证向 Amazon 发出请求。

在不使用 Amazon Cognito 的情况下使用 Web 联合身份验证的过程遵循此概要：

1. 以开发人员身份注册到外部身份提供程序 (IdP)，然后使用向您提供应用程序的唯一 ID 的 IdP 来配置您的应用程序。(不同的 IdP 使用不同的术语表示此过程。此概要使用配置一词表示通过 IdP 标识应用程序的过程。) 每个 IdP 均提供一个对于该 IdP 独一无二的应用程序 ID，因此如果按多个 IdP 的要求配置同一应用程序，则该应用程序将具有多个应用程序 ID。可按每个提供商的要求配置多个应用程序。

以下外部链接提供有关使用一些常用身份提供程序 (IdP) 的信息：

- [Login with Amazon 开发人员中心](#)
- 在 Facebook 开发人员网站上，将 [Facebook 登录名](#) 添加到您的应用程序或网站。
- 在 Google 开发人员网站上，使用 [OAuth 2.0 进行登录 \(OpenID Connect\)](#)。

Important

如果您使用 Google、Facebook 或 Amazon Cognito 提供的 OIDC 身份提供程序，请勿在 Amazon Web Services Management Console 中创建单独的 IAM 身份提供程序，Amazon 内置了这些 OIDC 身份提供程序，可供您使用。跳过以下步骤，直接使用身份提供程序创建新角色。

2. 如果您使用与 OIDC 兼容、除 Google、Facebook 或 Amazon Cognito 以外的 IdP，请为其创建 IAM 身份提供程序实体。
3. 在 IAM 中，您可以[创建一个或多个角色 \(p. 224\)](#)。对于每个角色，定义谁可代入该角色（信任策略）和应用程序的用户将具有什么权限（权限策略）。通常，您为应用程序支持的每个 IdP 创建一个角色。例如，可创建一个在用户通过 Login with Amazon 登录时应用程序代入的角色，为同一应用程序再创建第二个角色，其中用户通过 Facebook 登录，然后为该应用程序创建第三个角色，其中用户通过 Google 登录。对于信任关系，指定 IdP (如 Amazon.com) 作为 Principal (可信实体)，并加入一个与 IdP 分配的应用程序 ID 匹配的 Condition。[针对第三方身份提供程序创建角色（联合） \(p. 224\)](#) 将介绍不同提供商的角色示例。
4. 在应用程序中，通过 IdP 验证用户身份。执行此操作的方式的详情因您所使用的 IdP (Login with Amazon、Facebook 或 Google) 和运行应用程序的平台而异。例如，Android 应用程序的身份验证方式与 iOS 应用程序或基于 JavaScript 的 Web 应用程序的不同。

通常，如果用户尚未登录，则 IdP 负责显示登录页面。IdP 在对用户进行身份验证后，会将身份验证令牌与用户相关信息一起返回到您的应用程序。包含的信息取决于 IdP 公开的内容和用户愿意共享的信息。可在应用程序中使用这些信息。

5. 在应用程序中，对操作进行未签名 AssumeRoleWithWebIdentity 调用以请求临时安全凭证。在该请求中，传递 IdP 的身份验证令牌，然后指定为该 IdP 创建的 IAM 角色的 Amazon Resource Name (ARN)。Amazon 将验证令牌是否可信和有效，如果是这样，则会将临时安全凭证返回到具有从您在请求中命名的角色的权限的应用程序。响应中还包括来自 IdP 的用户相关元数据，例如，IdP 将其与用户关联的唯一用户 ID。
6. 通过使用来自 AssumeRoleWithWebIdentity 响应的临时安全凭证，应用程序向 Amazon API 操作发出已签名的请求。IdP 中的用户 ID 信息可以区分应用程序中的用户，例如，您可以将对象放入 Amazon S3 文件夹中，这些文件夹包含用户 ID 作为前缀或后缀。这样可创建锁定该文件夹的访问控制策略，以使仅具有该 ID 的用户能够访问该文件夹。有关更多信息，请参阅本主题后面的[使用 Web 联合身份验证识别用户的身份 \(p. 174\)](#)。
7. 您的应用程序应缓存临时安全凭证，这样您就不需要每次在应用程序需要对 Amazon 发出请求时获取新凭证。默认情况下，证书的有效期为 1 小时。当凭证到期时 (或在此之前)，再次调用 AssumeRoleWithWebIdentity 以获取新的一组临时安全凭证。根据 IdP 及其管理其令牌的方式，可能必须先刷新 IdP 的令牌，然后再对 AssumeRoleWithWebIdentity 进行新的调用，因为 IdP 的令牌通常也在固定的一定时间后到期。如果使用的是 Amazon SDK for iOS 或 Amazon SDK for Android，则可使用 [AmazonSTSCredentialsProvider](#) 操作，该操作管理 IAM 临时凭证，包括按需刷新凭证。

使用 Web 联合身份验证识别用户的身份

在 IAM 中创建访问策略时，可根据所配置的应用程序和根据已使用外部身份提供程序 (IdP) 进行身份验证的用户的 ID 指定权限，这一点通常很有用。举例来说，使用 Web 联合身份验证的移动应用程序可能按类似于如下形式的结构将信息保留在 Amazon S3 中：

```
myBucket/app1/user1
myBucket/app1/user2
myBucket/app1/user3
...
myBucket/app2/user1
myBucket/app2/user2
myBucket/app2/user3
...
```

可能还要按提供商区分这些路径。在这种情况下，结构可能类似于如下形式 (为节省空间，仅列出了两个提供商)：


```
myBucket/Amazon/app1/user1
myBucket/Amazon/app1/user2
myBucket/Amazon/app1/user3
...
myBucket/Amazon/app2/user1
myBucket/Amazon/app2/user2
myBucket/Amazon/app2/user3

myBucket/Facebook/app1/user1
myBucket/Facebook/app1/user2
myBucket/Facebook/app1/user3
...
myBucket/Facebook/app2/user1
myBucket/Facebook/app2/user2
myBucket/Facebook/app2/user3
...
```

在这些结构中，app1 和 app2 表示不同的应用程序 (如不同的游戏)，并且每个应用程序用户都有一个不同的文件夹。app1 和 app2 的值可能是您分配的易记名称 (例如 mynumbersgame)，也可能是在配置应用程序时提供商分配的应用程序 ID。如果决定在路径中加入提供商名称，则这些名称也可以是易记的名称，如 Cognito、Amazon、Facebook 和 Google。

由于应用程序名称为静态值，因此您可以通过 Amazon Web Services Management Console 创建 app1 和 app2 的文件夹。如果在路径中加入提供商名称，则同样如此，因为提供商名称也是静态值。相比之下，用户特定的文件夹 (*user1*、*user2*、*user3* 等) 必须使用通过请求 AssumeRoleWithWebIdentity 而返回的 SubjectFromWebIdentityToken 值中提供的用户 ID，在运行时从应用程序中创建。

要编写策略以允许独占访问个别用户的资源，可搭配文件夹全名，其中包括应用程序名称和提供商名称 (如果使用这两项)。然后，可包含以下提供商特定的上下文密钥，其中引用提供商返回的用户 ID：

- cognito-identity.amazonaws.com:sub
- www.amazon.com:user_id
- graph.facebook.com:id
- accounts.google.com:sub

对于 OIDC 提供商，请将 OIDC 提供商的完全限定 URL 与子上下文密钥一起使用，如以下示例所示：

- *server.example.com*:sub

以下示例说明了一个权限策略，只有在 Amazon S3 中的存储桶前缀与以下字符串匹配时，该策略才会为该存储桶授予访问权限：

```
myBucket/Amazon/mynumbersgame/user1
```

该示例假定用户使用 Login with Amazon 登录并正在使用一个名为 mynumbersgame 的应用程序。用户的唯一 ID 以名为 user_id 的属性表示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::myBucket"],
      "Condition": {"StringLike": {"s3:prefix": ["Amazon/mynumbersgame/
${www.amazon.com:user_id}/*"]}}
    },
  ],
}
```

```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:PutObject",
    "s3:DeleteObject"
  ],
  "Resource": [
    "arn:aws:s3:::myBucket/amazon/mynumbersgame/${www.amazon.com:user_id}",
    "arn:aws:s3:::myBucket/amazon/mynumbersgame/${www.amazon.com:user_id}/*"
  ]
}
```

您可以为使用 Amazon Cognito、Facebook、Google 或其他 OpenID Connect 兼容的 IdP 登录的用户创建类似的策略。这些策略将使用不同的提供商名称作为路径的一部分，并将使用不同的应用程序 ID。

有关可用于策略中的条件检查的 Web 联合身份验证密钥的更多信息，请参阅[Amazon Web 联合身份验证的可用键](#) (p. 1016)。

有关 Web 联合身份验证的其他资源

以下资源可帮助您详细了解 Web 联合身份验证：

- Amazon Mobile SDK for Android 开发人员指南中的 [Amazon Cognito 身份](#) 和 Amazon Mobile SDK for iOS 开发人员指南中的 [Amazon Cognito 身份](#)。
- [Web Identity Federation Playground](#) 是一个交互式网站，可在上面演练通过 Login with Amazon、Facebook 或 Google 进行身份验证、获得临时安全凭证，然后使用这些凭证向 Amazon 提出请求的过程。
- [Amazon .NET 开发博客上的博文 Amazon 使用 SDK for .NET 进行 Web 联合身份验证](#) 详细介绍如何将 Web 联合身份验证与 Facebook 配合使用，并加入了一段 C# 代码，其中展示如何调用 `AssumeRoleWithWebIdentity` 以及如何从该 API 调用中使用临时安全凭证访问 S3 存储桶。
- [Amazon SDK for iOS](#) 和 [Amazon SDK for Android](#) 含有示例应用程序。这些应用程序中包括的代码展示如何调用身份提供程序，然后如何使用这些提供商提供的信息获得和使用临时安全凭证。
- [Web Identity Federation with Mobile Applications](#) 一文讨论 Web 联合身份验证，并通过一个示例，介绍如何使用 Web 联合身份验证获取 Amazon S3 中内容的访问权限。

关于基于 SAML 2.0 的联合身份验证

Amazon 使用 [SAML 2.0 \(安全断言标记语言 2.0\)](#) 支持联合身份验证，SAML 2.0 是许多身份验证提供商 (IdP) 使用的一种开放标准。此功能可实现联合单点登录 (SSO)，因此用户可以登录 Amazon Web Services Management Console 或调用 IAM API 操作，而不必为企业中的每个人都创建一个 Amazon 用户。通过使用 SAML，您可以使用 Amazon 简化配置联合身份验证流程，因为您可以使用 IdP 的服务而不是[编写自定义身份代理代码](#)。

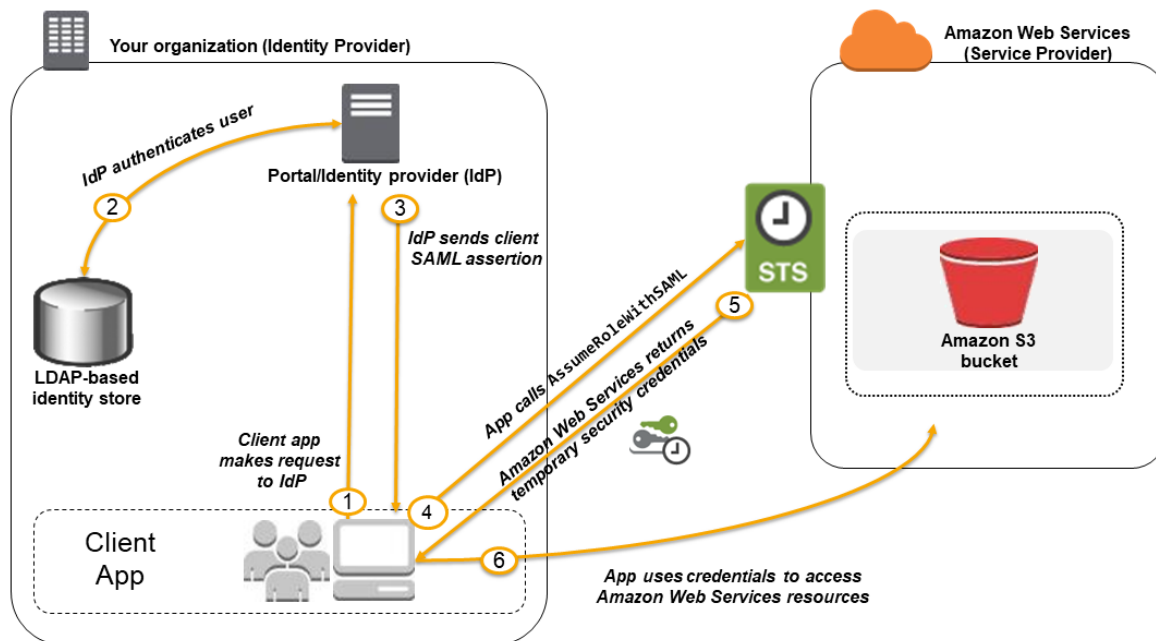
IAM 联合支持这些使用案例：

- 允许组织中的用户或应用程序调用 [Amazon API 操作的联合访问权限](#) (p. 177)。您可以使用组织内生成的 SAML 断言 (身份验证响应的一部分) 获得临时安全凭证。此方案类似于 IAM 支持的其他联合方案，如 [请求临时安全凭证](#) (p. 302) 和 [关于 Web 联合身份验证](#) (p. 171) 中介绍的方案。但是，企业中基于 SAML 2.0 的 IdP 可以在运行时处理很多细节功能，以用于执行身份验证和授权检查。这是本主题中讨论的方案。
- 从组织向 [Amazon Web Services Management Console](#) 进行基于 Web 的单一登录 (SSO) (p. 199)。用户可以登录您企业中由与 SAML 2.0 兼容的 IdP 托管的门户，选择转向 Amazon，并将其重新导向到控制台，而无需提供其他登录信息。您可以使用第三方 SAML IdP 建立对控制台的 SSO 访问，或者可以创建

自定义 IdP 来支持外部用户的控制台访问。有关构建自定义 IdP 的更多信息，请参阅[使自定义身份代理能够访问 Amazon 控制台 \(p. 201\)](#)。

使用基于 SAML 的联合身份验证来对 Amazon 进行 API 访问

假设您想要为员工提供一种将数据从他们的计算机中复制到备份文件夹的方法。您可以构建一个可在用户的计算机上运行的应用程序。在后端，该应用程序可在 S3 存储桶中读写对象。用户没有直接访问 Amazon 的权限。而应使用以下过程：



1. 您组织中的用户使用客户端应用程序来请求您组织的 IdP 进行身份验证。
2. IdP 根据组织的身份存储对用户进行身份验证。
3. IdP 构建一个具有用户相关信息的 SAML 断言，并将此断言发送到客户端应用程序。
4. 客户端应用程序调用 Amazon STS [AssumeRoleWithSAML](#) API，并传递 SAML 提供商的 ARN、要代入的角色的 ARN 以及来自 IdP 的 SAML 断言。
5. API 对客户端应用程序的响应包括临时安全凭证。
6. 客户端应用程序使用临时安全凭证来调用 Amazon S3 API 操作。

配置基于 SAML 2.0 的联合身份验证的概述

在使用前面方案和图表中所述的基于 SAML 2.0 的联合身份验证之前，您必须先配置组织的 IdP 和您的 Amazon 账户，使之相互信任。以下步骤介绍了用于配置此信任的一般过程。组织内部必须有[支持 SAML 2.0 的 IdP \(p. 192\)](#)，例如 Microsoft Active Directory 联合身份验证服务 (AD FS，Windows Server 的一部分)、Shibboleth 或其他兼容的 SAML 2.0 提供商。

配置组织的 IdP 和 Amazon 使之相互信任

1. 将 Amazon 注册为您组织的 IdP 的服务提供商 (SP)。通过 <https://region-code.signin.amazonaws.cn/static/saml-metadata.xml> 使用 SAML 元数据文档

有关可能的 [region-code](#) 值的列表，请参阅 [Amazon 登录端点](#) 中的 Region (区域) 列。

您可以选择通过 <https://signin.amazonaws.cn/static/saml-metadata.xml> 使用 SAML 元数据文档。

2. 通过使用您企业的 IdP，生成一个同等元数据 XML 文件，该文件可将您的 IdP 描述为 Amazon 中的 IAM 身份提供程序。它必须包括发布者名称、创建日期、过期日期以及 Amazon 可用来验证来自您组织的身份验证响应（断言）的密钥。
3. 在 IAM 控制台中，创建一个 SAML 身份提供程序实体。在此过程中，您将上传您组织的 IdP 在 [Step 2](#) 中生成的 SAML 元数据文档。有关更多信息，请参阅 [创建 IAM SAML 身份提供程序 \(p. 188\)](#)。
4. 在 IAM 中创建一个或多个 IAM 角色。在角色的信任策略中，您可将 SAML 提供商设置为可在您的组织与 Amazon 之间建立信任关系的主体。该角色的权限策略确定了允许您组织的用户在 Amazon 中执行的操作。有关更多信息，请参阅 [针对第三方身份提供程序创建角色（联合） \(p. 224\)](#)。
5. 在您企业的 IdP 中，定义可将您企业中的用户或组映射到 IAM 角色的断言。请注意，您的组织中不同的用户和组可能映射到不同的 IAM 角色。执行映射的确切步骤取决于您使用的 IdP。在用户 Amazon S3 文件夹中的 [较早场景 \(p. 177\)](#) 中，则所有用户都可能映射到提供 Amazon S3 权限的同一角色。有关更多信息，请参阅 [为身份验证响应配置 SAML 断言 \(p. 194\)](#)。

如果您的 IdP 支持对 Amazon 控制台的 SSO，则可配置控制台会话的最大持续时间。有关更多信息，请参阅 [使 SAML 2.0 联合身份用户能够访问 Amazon Web Services Management Console \(p. 199\)](#)。

Note

SAML 2.0 联合的 Amazon 实施不支持 IAM 身份提供程序与 Amazon 之间的加密 SAML 断言。但是，客户系统与 Amazon 之间的通信在加密的 (TLS) 通道上进行传输。

6. 在您正在创建的应用程序中，您可以调用 Amazon Security Token Service AssumeRoleWithSAML API，将其传递给您在 [Step 3](#) 中创建的 SAML 提供商的 ARN、您在 [Step 4](#) 中创建的要代入的角色的 ARN 以及从您的 IdP 处获取的有关当前用户的 SAML 断言。Amazon 确保代入角色的请求来自 SAML 提供商所引用的 IdP。

有关更多信息，请参阅 https://docs.amazonaws.cn/STS/latest/APIReference/API_AssumeRoleWithSAML.html API 参考中的 Amazon Security Token Service AssumeRoleWithSAML。

7. 如果请求成功，API 会返回一组临时安全凭证，您的应用程序即可用其向 Amazon 发出已签名的请求。您的应用程序具有有关当前用户的信息并可访问 Amazon S3 中用户特定的文件夹，如上一方案中所述。

用于允许对 Amazon 资源进行 SAML 联合访问的角色的概述

您在 IAM 中创建的角色将确定允许您企业中的联合身份用户在 Amazon 中执行的操作。当您为角色创建信任策略时，您可以将先前创建的 SAML 提供商指定为 Principal。此外，您还可以使用 Condition 设置信任策略的范围，以便仅允许与特定 SAML 属性匹配的用户访问角色。例如，您可以指定仅允许 SAML 从属关系为 staff（在 <https://openidp.feide.no> 中断言）的用户访问角色，如下示例策略所示：

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"Federated": "arn:aws-cn:iam::account-id:saml-provider/ExampleOrgSSOProvider"},
    "Action": "sts:AssumeRoleWithSAML",
    "Condition": {
      "StringEquals": {
        "saml:aud": "https://signin.amazonaws.cn/saml",
        "saml:iss": "https://openidp.feide.no"
      },
      "ForAllValues:StringLike": {"saml:edupersonaffiliation": ["staff"]}
    }
  }]
}
```

有关您可以签入策略的 SAML 密钥的更多信息，请参阅 [基于 SAML 的 Amazon STS 联合身份验证的可用键 \(p. 1019\)](#)。

您可以在 `https://region-code.signin.aws.amazon.com/static/saml-metadata.xml` 中包含 `saml:aud` 属性的区域端点。有关可能的 [region-code](#) 值的列表，请参阅 [Amazon 登录端点](#) 中的 Region (区域) 列。

对于该角色中的权限策略，您可以像任何角色一样指定权限。例如，如果允许您企业的用户管理 Amazon Elastic Compute Cloud 实例，您必须在权限策略中明确允许 Amazon EC2 操作，如 `AmazonEC2FullAccess` 托管策略中的操作。

唯一标识基于 SAML 的联合中的用户

在 IAM 中创建访问策略时，可根据用户的身份指定权限，这一点通常很有用。举例来说，对于已使用 SAML 联合的用户，应用程序可能希望使用如下的结构保留 Amazon S3 中的信息：

```
myBucket/app1/user1
myBucket/app1/user2
myBucket/app1/user3
```

您可以通过 Amazon S3 控制台或 Amazon CLI 创建存储桶 (myBucket) 和文件夹 (app1)，因为这些都是静态值。但是，用户特定文件夹 ([user1](#)、[user2](#)、[user3](#) 等) 必须在运行时使用代码创建，因为在用户首次通过联合流程登录之前，用来标识用户的值是未知的。

要编写在资源名称中引用特定于用户的详细信息的策略，必须在可以用于策略条件的 SAML 密钥中提供用户身份。以下密钥可用于基于 SAML 2.0 的联合身份验证，以便在 IAM policy 中使用。您可以使用以下键返回的值为资源 (如 Amazon S3 文件夹) 创建唯一的用户标识符。

- `saml:namequalifier`. 哈希值，基于 Issuer 响应值 (`saml:iss`)、包含 AWS 账户 ID 的字符串和 IAM 中 SAML 提供商的友好名称 (ARN 的最后一部分) 的串联。账户 ID 与 SAML 提供商的易记名称的串联可作为键 `saml:doc` 供 IAM policy 使用。账户 ID 与提供商名称必须使用 "/" 分隔，例如在 "123456789012/provider_name" 中。有关更多信息，请参阅 `saml:doc` 上的 [基于 SAML 的 Amazon STS 联合身份验证的可用键 \(p. 1019\)](#) 键。

`NameQualifier` 与 `Subject` 的组合可用于唯一识别联合身份用户。以下伪代码显示如何计算此值。在此伪代码中，+ 表示串联，SHA1 代表使用 SHA-1 生成消息摘要的功能，Base64 代表生成哈希输出的 Base-64 编码版本的功能。

```
Base64 ( SHA1 ( "https://example.com/saml" + "123456789012" + "/"
MySAMLIdP" ) )
```

有关可用于基于 SAML 的联合的策略键的更多信息，请参阅 [基于 SAML 的 Amazon STS 联合身份验证的可用键 \(p. 1019\)](#)。

- `saml:sub` (字符串)。这是该陈述的主题，其中包含唯一标识组织中某个用户的值 (例如 `_cbb88bf52c2510eabe00c1642d4643f41430fe25e3`)。
- `saml:sub_type` (字符串)。此键可以是 `persistent`、`transient` 或在您的 SAML 断言中使用的 `Format` 和 `Subject` 元素的完整 `NameID` URI。`persistent` 的值表示在所有会话中用户的 `saml:sub` 值是相同的。如果值为 `transient`，则用户在每个会话中拥有不同的 `saml:sub` 值。有关 `NameID` 元素的 `Format` 属性的信息，请参阅 [为身份验证响应配置 SAML 断言 \(p. 194\)](#)。

以下示例说明了一个权限策略，该策略使用上述密钥为 Amazon S3 中的用户特定文件夹授予权限。该策略假设 Amazon S3 对象使用同时包含 `saml:namequalifier` 和 `saml:sub` 的前缀进行标识。请注意，`Condition` 元素包括一个测试，用于确保 `saml:sub_type` 设置为 `persistent`。如果已设置为 `transient`，每个会话用户的 `saml:sub` 值可以不同，且不应使用值的组合来标识用户特定的文件夹。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
```



```
        "s3:PutObject",
        "s3:DeleteObject"
    ],
    "Resource": [
        "arn:aws-cn:s3:::exampleorgBucket/backup/${saml:namequalifier}/${saml:sub}",
        "arn:aws-cn:s3:::exampleorgBucket/backup/${saml:namequalifier}/${saml:sub}/*"
    ],
    "Condition": { "StringEquals": { "saml:sub_type": "persistent" } }
  }
}
```

有关将断言从 IdP 映射到策略的更多信息，请参阅[为身份验证响应配置 SAML 断言](#) (p. 194)。

创建 IAM 身份提供程序

如果要使用外部身份提供程序服务 (IdP) 服务配置身份联合，可以创建 IAM 身份提供程序，以将 IdP 及其配置告知 Amazon。这将在您的 Amazon 账户和 IdP 之间建立“信任”。以下主题包括有关如何为每种 IdP 类型创建 IAM 身份提供程序的详细信息。

主题

- [创建 OpenID Connect \(OIDC\) 身份提供程序](#) (p. 180)
- [创建 IAM SAML 身份提供程序](#) (p. 188)

创建 OpenID Connect (OIDC) 身份提供程序

IAM OIDC 身份提供程序是 IAM 中的实体，这些实体描述支持 [OpenID Connect](#) (OIDC) 标准的身份提供程序 (IdP) 服务，如 Google 或 Salesforce)。当您要在与 OIDC 兼容的 IdP 和您的 Amazon 账户之间建立信任时，请使用 IAM OIDC 身份提供程序。如果您正在创建需要访问 Amazon 资源的移动应用或 Web 应用程序，但又不想创建自定义登录代码或管理您自己的用户身份，这会很有用。有关此方案的更多信息，请参阅[the section called “关于 Web 联合身份验证”](#) (p. 171)。

您可以使用 Amazon Web Services Management Console、Amazon Command Line Interface、Tools for Windows PowerShell 或 IAM API 创建和管理 IAM OIDC 身份提供程序。

创建 IAM OIDC 身份提供程序后，必须创建一个或多个 IAM 角色。角色是 Amazon 中的一个实体，它没有自己的凭证（与用户一样）。但在此上下文中，角色将动态分配给由组织的 IdP 验证的联合身份用户。该角色允许组织的 IdP 请求临时安全凭证以便访问 Amazon。分配给该角色的策略决定了联合身份用户可在 Amazon 中执行的操作。要为第三方身份提供程序创建角色，请参阅[针对第三方身份提供程序创建角色（联合）](#) (p. 224)。

主题

- [创建和管理 OIDC 提供商（控制台）](#) (p. 180)
- [创建和管理 IAM OIDC 身份提供程序 \(Amazon CLI\)](#) (p. 182)
- [创建和管理 OIDC 身份提供程序 \(Amazon API\)](#) (p. 184)
- [获取 OpenID Connect 身份提供程序的指纹](#) (p. 185)

创建和管理 OIDC 提供商（控制台）

按照以下说明在 Amazon Web Services Management Console 中创建和管理 IAM OIDC 身份提供程序。

Important

如果您使用的是 Google、Facebook 或 Amazon Cognito 的 OIDC 身份提供程序，请勿使用此过程创建单独的 IAM 身份提供程序。这些 OIDC 身份提供程序已经内置到 Amazon，并可供您使用。相反，请按照以下步骤为身份提供程序创建新角色，请参阅[创建用于 Web 联合身份验证或 OpenID Connect 联合身份验证的角色（控制台）](#) (p. 226)。

创建 IAM OIDC 身份提供程序 (控制台)

1. 在创建 IAM OIDC 身份提供程序之前，您必须向 IdP 注册您的应用程序以便接收客户端 ID。客户端 ID (也称为受众) 是您的应用的唯一标识符，在您向 IdP 注册您的应用时颁发给您。有关如何获取客户端 ID 的更多信息，请参阅您的 IdP 的文档。

Note

Amazon 通过我们的受信任证书颁发机构 (CA) 库保护与某些 OIDC 身份提供程序 (IdP) 的通信，而不是使用证书指纹验证您的 IdP 服务器证书。这些 OIDC IdPs 包括 Google，以及那些使用 Amazon S3 存储桶托管 JSON Web 密钥集 (JWKS) 端点的 IdP。在这些情况下，您的旧式指纹将保留在您的配置中，但不再用于验证。

2. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
3. 在导航窗格中，选择 Identity providers (身份提供程序)，然后选择 Add provider (添加提供商)。
4. 对于 Configure provider (配置提供商)，选择 OpenID Connect。
5. 对于 Provider URL，键入 IdP 的 URL。该 URL 必须遵从以下限制：
 - 该 URL 区分大小写。
 - URL 必须以 **https://** 开头。
 - URL 不应包含端口号。
 - 在您的 Amazon 账户中，每个 IAM OIDC 身份提供程序都必须有唯一的 URL。
6. 选择 Get thumbprint (获取指纹) 验证您的 IdP 的服务器证书。要了解如何操作，请参阅[获取 OpenID Connect 身份提供程序的指纹 \(p. 185\)](#)。
7. 对于 Audience，键入您向 IdP 注册的、在[Step 1](#) 中接收的并且将向 Amazon 发出请求的应用程序的客户端 ID。如果您对于此 IdP 有其他客户端 ID (也称为受众)，稍后可以在提供商详细信息页面上添加它们。
8. (可选) 对于 Add tags (添加标签)，您可以添加键值对来帮助识别和组织您的 IdP。您还可以使用标签来控制对 Amazon 资源的访问。要了解有关标记 IAM OIDC 身份提供程序的更多信息，请参阅[标记 OpenID Connect \(OIDC\) 身份提供程序 \(p. 285\)](#)。选择 Add tag (添加标签)。为每个标签键值对输入值。
9. 验证您提供的信息。完成此操作后，选择 Add provider (添加提供商)。
10. 将 IAM 角色分配至身份提供程序，以向身份提供程序管理的外部用户身份授予访问账户中的 Amazon 资源的权限。要了解有关为联合身份创建角色的更多信息，请参阅[针对第三方身份提供程序创建角色 \(联合\) \(p. 224\)](#)。

为 IAM OIDC 身份提供程序添加或删除指纹 (控制台)

1. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Identity providers (身份提供程序)。然后选择要更新的 IAM 身份提供程序的名称。
3. 在 Thumbprints (指纹) 部分，选择 Manage (管理)。要输入新的指纹值，请选择 Add thumbprint (添加指纹)。要删除指纹，请选择您要删除的指纹旁边的 Remove (删除)。

Note

IAM OIDC 身份提供程序必须具有至少 1 个、最多 5 个指纹。

完成后，选择 Save changes (保存更改)。

为 IAM OIDC 身份提供程序添加受众 (控制台)

1. 在导航窗格中，选择 Identity providers (身份提供程序)，然后选择要更新的 IAM 身份提供程序的名称。

2. 在 Audiences (受众) 部分, 选择 Actions (操作), 然后选择 Add audience (添加受众)。
3. 键入您向 IdP 注册的、在 [Step 1](#) 中接收的并且将向 Amazon 发出请求的应用程序的客户端 ID。然后选择 Add audiences (添加受众)。

Note

IAM OIDC 身份提供程序必须具有至少 1 个且最多 100 个受众。

为 IAM OIDC 身份提供程序删除受众 (控制台)

1. 在导航窗格中, 选择 Identity providers (身份提供程序), 然后选择要更新的 IAM 身份提供程序的名称。
2. 在 Audiences (受众) 部分, 选择要删除的受众旁边的单选按钮, 然后选择 Actions (操作)。
3. 选择 Remove audience (删除受众)。此时会打开一个新窗口。
4. 如果删除受众, 受众的联合身份将无法代入与受众关联的角色。在窗口中, 阅读警告并通过在字段中键入 remove 一词以确认删除受众。
5. 选择 Remove (删除) 以删除受众。

删除 IAM OIDC 身份提供程序 (控制台)

1. 通过以下网址打开 IAM 控制台: <https://console.aws.amazon.com/iam/>。
2. 在导航窗格中, 选择 Identity providers (身份提供程序)。
3. 选中要删除的 IAM 身份提供程序旁边的复选框。此时会打开一个新窗口。
4. 通过在字段中键入 delete 一词以确认您要删除此提供商。然后选择 Delete (删除)。

创建和管理 IAM OIDC 身份提供程序 (Amazon CLI)

可以使用以下 Amazon CLI 命令来创建和管理 IAM OIDC 身份提供程序。

创建 IAM OIDC 身份提供程序 (Amazon CLI)

1. (可选) 要获取您 Amazon 账户中所有 IAM OIDC 身份提供程序的列表, 请运行以下命令:

- `aws iam list-open-id-connect-providers`

2. 要创建新的 IAM OIDC 身份提供程序, 请运行以下命令:

- `aws iam create-open-id-connect-provider`

更新现有 IAM OIDC 身份提供程序的服务器证书指纹列表 (Amazon CLI)

- 要更新 IAM OIDC 身份提供程序的服务器证书指纹列表, 请运行以下命令:

- `aws iam update-open-id-connect-provider-thumbprint`

要标记现有 IAM OIDC 身份提供程序 (Amazon CLI)

- 要标记现有 IAM OIDC 身份提供程序, 请运行以下命令:

- `aws iam tag-open-id-connect-provider`

要列出现有 IAM OIDC 身份提供程序 (Amazon CLI) 的标签

- 要列出现有 IAM OIDC 身份提供程序的标签, 请运行以下命令:

- `aws iam list-open-id-connect-provider-tags`

要删除 IAM OIDC 身份提供程序 (Amazon CLI) 的标签

- 要删除现有 IAM OIDC 身份提供程序的标签，请运行以下命令：

- `aws iam untag-open-id-connect-provider`

要标记现有 IAM OIDC 身份提供程序 (Amazon CLI)

- 要标记现有 IAM OIDC 身份提供程序，请运行以下命令：

- `aws iam tag-open-id-connect-provider`

要列出现有 IAM OIDC 身份提供程序 (Amazon CLI) 的标签

- 要列出现有 IAM OIDC 身份提供程序的标签，请运行以下命令：

- `aws iam list-open-id-connect-provider-tags`

要删除 IAM OIDC 身份提供程序 (Amazon CLI) 的标签

- 要删除现有 IAM OIDC 身份提供程序的标签，请运行以下命令：

- `aws iam untag-open-id-connect-provider`

在现有 IAM OIDC 提供商中添加或删除客户端 ID (Amazon CLI)

1. (可选) 要获取您 Amazon 账户中所有 IAM OIDC 身份提供程序的列表，请运行以下命令：

- `aws iam list-open-id-connect-providers`

2. (可选) 要获取有关 IAM OIDC 身份提供程序的详细信息，请运行以下命令：

- `aws iam get-open-id-connect-provider`

3. 要向现有 IAM OIDC 身份提供程序中添加新的客户端 ID，请运行以下命令：

- `aws iam add-client-id-to-open-id-connect-provider`

4. 要从现有 IAM OIDC 身份提供程序中删除客户端，请运行以下命令：

- `aws iam remove-client-id-from-open-id-connect-provider`

删除 IAM OIDC 身份提供程序 (Amazon CLI)

1. (可选) 要获取您 Amazon 账户中所有 IAM OIDC 身份提供程序的列表，请运行以下命令：

- `aws iam list-open-id-connect-providers`

2. (可选) 要获取有关 IAM OIDC 身份提供程序的详细信息，请运行以下命令：

- `aws iam get-open-id-connect-provider`

3. 要删除 IAM OIDC 身份提供程序，请运行以下命令：

- `aws iam delete-open-id-connect-provider`

创建和管理 OIDC 身份提供程序 (Amazon API)

可以使用以下 IAM API 命令来创建和管理 OIDC 提供商。

创建 IAM OIDC 身份提供程序 (Amazon API)

1. (可选) 要获取您 Amazon 账户中所有 IAM OIDC 身份提供程序的列表，请调用以下操作：

- [ListOpenIDConnectProviders](#)

2. 要创建新的 IAM OIDC 身份提供程序，请调用以下操作：

- [CreateOpenIDConnectProvider](#)

更新现有 IAM OIDC 身份提供程序的服务器证书指纹列表 (Amazon API)

• 要更新 IAM OIDC 身份提供程序的服务器证书指纹列表，请调用以下操作：

- [UpdateOpenIDConnectProviderThumbprint](#)

要标记现有 IAM OIDC 身份提供程序 (Amazon API)

• 要标记现有 IAM OIDC 身份提供程序，请调用以下操作：

- [TagOpenIDConnectProvider](#)

要列出现有 IAM OIDC 身份提供程序 (Amazon API) 的标签

• 要列出现有 IAM OIDC 身份提供程序的标签，请调用以下操作：

- [ListOpenIDConnectProviderTags](#)

要删除现有 IAM OIDC 身份提供程序 (Amazon API) 的标签

• 要删除现有 IAM OIDC 身份提供程序的标签，请调用以下操作：

- [UntagOpenIDConnectProvider](#)

要标记现有 IAM OIDC 身份提供程序 (Amazon API)

• 要标记现有 IAM OIDC 身份提供程序，请调用以下操作：

- [TagOpenIDConnectProvider](#)

要列出现有 IAM OIDC 身份提供程序 (Amazon API) 的标签

• 要列出现有 IAM OIDC 身份提供程序的标签，请调用以下操作：

- [ListOpenIDConnectProviderTags](#)

要删除现有 IAM OIDC 身份提供程序 (Amazon API) 的标签

• 要删除现有 IAM OIDC 身份提供程序的标签，请调用以下操作：

- [UntagOpenIDConnectProvider](#)

在现有 IAM OIDC 提供商中添加或删除客户端 ID (Amazon API)

1. (可选) 要获取您 Amazon 账户中所有 IAM OIDC 身份提供程序的列表, 请调用以下操作:
 - [ListOpenIDConnectProviders](#)
2. (可选) 要获取有关 IAM OIDC 身份提供程序的详细信息, 请调用以下操作:
 - [GetOpenIDConnectProvider](#)
3. 要向现有 IAM OIDC 身份提供程序中添加新的客户端 ID, 请调用以下操作:
 - [AddClientIDToOpenIDConnectProvider](#)
4. 要从现有 IAM OIDC 身份提供程序中删除客户端 ID, 请调用以下操作:
 - [RemoveClientIDFromOpenIDConnectProvider](#)

删除 IAM OIDC 身份提供程序 (Amazon API)

1. (可选) 要获取您 Amazon 账户中所有 IAM OIDC 身份提供程序的列表, 请调用以下操作:
 - [ListOpenIDConnectProviders](#)
2. (可选) 要获取有关 IAM OIDC 身份提供程序的详细信息, 请调用以下操作:
 - [GetOpenIDConnectProvider](#)
3. 要删除 IAM OIDC 身份提供程序, 请调用以下操作:
 - [DeleteOpenIDConnectProvider](#)

获取 OpenID Connect 身份提供程序的指纹

当您在 IAM 中[创建 OpenID Connect \(OIDC\) 身份提供程序 \(p. 180\)](#)时, 则必须提供指纹。IAM 需要该指纹, 用于对外部身份提供程序 (IdP) 使用的证书进行签名的顶部中间证书颁发机构 (CA)。该指纹是用于为 OIDC 兼容 IdP 颁发证书的 CA 证书的签名。在创建 IAM OIDC 身份提供程序时, 您信任该 IdP 验证的身份有权访问您的 Amazon 账户。通过提供 CA 的证书指纹, 您信任该 CA 颁发的任何证书, 并且其 DNS 名称与注册的名称相同。这样, 在续订 IdP 的签名证书时, 无需在每个账户中更新信任关系。

Important

在大多数情况下, 联合身份验证服务器使用两个不同的证书:

- 第一个证书将在 Amazon 与您的 IdP 之间建立 HTTPS 连接。这应由已知公有根 CA (如 Amazon Certificate Manager) 发布。这使客户端能够检查证书的可靠性和状态。
- 第二个证书将用于加密令牌, 且应由私有或公有根 CA 签署。

您可以使用 [Amazon Command Line Interface](#)、[Tools for Windows PowerShell](#) 或 [IAM API \(p. 182\)](#) 创建一个 IAM OIDC 身份提供程序。当您使用这些方法时, 必须手动获取指纹并将它提供给 Amazon。当您使用 [IAM 控制台 \(p. 180\)](#) 创建 OIDC 身份提供程序时, 此控制台会尝试为您获取指纹。此外, 我们建议您手动获取 OIDC IdP 的指纹, 并验证控制台提取了正确的指纹。

您可以使用 Web 浏览器和 OpenSSL 命令行工具获取 OIDC 提供商的指纹。有关更多信息, 请参阅以下部分。

获取 OIDC IdP 的指纹

1. 您需要先获取 OpenSSL 命令行工具, 然后才能获取 OIDC IdP 的指纹。您可使用此工具下载 OIDC IdP 的证书链并生成证书链中最终证书的指纹。如果需要安装和配置 OpenSSL, 请遵循[安装 OpenSSL \(p. 187\)](#) 和[配置 OpenSSL \(p. 187\)](#) 中的说明。

您的命令窗口将显示类似于以下示例的证书指纹：

```
SHA1 Fingerprint=99:0F:41:93:97:2F:2B:EC:F1:2D:DE:DA:52:37:F9:C9:52:F2:0D:9E
```

请从此字符串中去掉冒号 (:) 字符以生成最终指纹，如：

```
990F4193972F2BECF12DDEDA5237F9C952F20D9E
```

7. 如果您要使用 Amazon CLI、Tools for Windows PowerShell 或 IAM API 创建 IAM OIDC 身份提供程序，请在创建该提供商时提供此指纹。

如果您要在 IAM 控制台中创建 IAM OIDC 身份提供程序，请在创建 OIDC 提供商时将此指纹与控制台的 Verify Provider Information (确认提供商信息) 页面中显示的指纹进行对比。

Important

如果您获得的指纹与您在控制台中看到的指纹不匹配，则不应该在控制台中创建该 OIDC 提供商。您应等待一段时间，在创建提供商之前确保指纹匹配，然后再次尝试创建 OIDC 商。如果第二次尝试后指纹仍然不匹配，请使用 [IAM 论坛](#) 联系 Amazon。

安装 OpenSSL

如果您没有安装 OpenSSL，请按照本部分中的说明执行操作。

在 Linux 或 Unix 系统上安装 OpenSSL

1. 转到 [OpenSSL : 源、Tarball](https://openssl.org/source/) (https://openssl.org/source/)。
2. 下载最新源并构建包。

如要在 Windows 系统上安装 OpenSSL

1. 转到 [OpenSSL : 二进制分布](https://wiki.openssl.org/index.php/Binaries) (https://wiki.openssl.org/index.php/Binaries)，了解您可以从中安装 Windows 版本的站点列表。
2. 按照所选站点上的说明开始安装。
3. 如果系统要求您安装 Microsoft Visual C++ 2008 Redistributables 并且该程序尚未安装在您的系统上，请选择适合您环境的下载链接。按照 Microsoft Visual C++ 2008 Redistributable 安装向导中的说明操作。

Note

如果您不确定您的系统上是否已安装 Microsoft Visual C++ 2008 Redistributables，则可以尝试先安装 OpenSSL。如果尚未安装 Microsoft Visual C++ 2008 Redistributables，OpenSSL 安装程序将显示提示。请确保安装与您安装的 OpenSSL 版本匹配的体系架构（32 位或 64 位）。

4. 在安装 Microsoft Visual C++ 2008 Redistributables 后，为您的环境选择适当的 OpenSSL 二进制版本，然后在本地保存该文件。启动 OpenSSL 设置向导。
5. 按照 OpenSSL 设置向导中的说明进行操作。

配置 OpenSSL

在使用 OpenSSL 命令之前，您必须配置操作系统，使其具有有关 OpenSSL 安装位置的信息。

要在 Linux 或 Unix 上配置 OpenSSL

1. 在命令行中，将 OPENSSL_HOME 变量设置为 OpenSSL 安装的位置：

```
$ export OpenSSL_HOME=path_to_your_OpenSSL_installation
```

2. 设置包含 OpenSSL 安装的路径：

```
$ export PATH=$PATH:$OpenSSL_HOME/bin
```

Note

通过使用 `export` 命令行对环境变量所做的任何更改只对当前的会话有效。通过在 shell 配置文件中设置环境变量，可对环境变量进行持续更改。有关更多信息，请参阅您的操作系统文档。

要在 Windows 上配置 OpenSSL

1. 打开 Command Prompt (命令提示符窗口)。
2. 将 `OpenSSL_HOME` 变量设置为 OpenSSL 安装的位置：

```
C:\> set OpenSSL_HOME=path_to_your_OpenSSL_installation
```

3. 将 `OpenSSL_CONF` 变量设置为 OpenSSL 安装中配置文件的位置：

```
C:\> set OpenSSL_CONF=path_to_your_OpenSSL_installation\bin\openssl.cfg
```

4. 设置包含 OpenSSL 安装的路径：

```
C:\> set Path=%Path%;%OpenSSL_HOME%\bin
```

Note

通过 Command Prompt (命令提示符) 对 Windows 环境变量所做的任何更改只对当前的命令行会话有效。您可以通过将环境变量设置为系统属性来对其进行持久性更改。确切的流程取决于您使用的 Windows 版本。(例如，在 Windows 7 中，打开 Control Panel (控制面板)、System and Security (系统和安全)、System (系统)。然后选择 Advanced system settings (高级系统设置)、Advanced (高级) 选项卡，Environment Variables (环境变量)。) 有关更多信息，请参阅 Windows 文档。

创建 IAM SAML 身份提供程序

IAM SAML 2.0 身份提供程序是 IAM 中的一个实体，该实体描述支持 [SAML 2.0 \(安全断言标记语言 2.0\)](#) 标准的外部身份提供程序 (IdP) 服务。如果您希望在与 SAML 兼容的 IdP (例如，Shibboleth 或 Active Directory 联合身份验证服务) 和 Amazon 之间建立信任，以便企业中的用户能够访问 Amazon 资源，则需要使用 IAM 身份提供程序。IAM SAML 身份提供程序用作 IAM 信任策略中的主体。

有关此方案的更多信息，请参阅[关于基于 SAML 2.0 的联合身份验证 \(p. 176\)](#)。

您可以在 Amazon Web Services Management Console 中或通过使用 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 调用，创建和管理 IAM 身份提供程序。

创建 SAML 提供商后，必须创建一个或多个 IAM 角色。角色是 Amazon 中的一个实体，它没有自己的凭证 (与用户一样)。但在此上下文中，角色将动态分配给由组织的 IdP 验证的联合身份用户。该角色允许组织的 IdP 请求临时安全凭证以便访问 Amazon。分配给该角色的策略决定了联合身份用户可在 Amazon 中执行的操作。要创建用于 SAML 联合的角色，请参阅[针对第三方身份提供程序创建角色 \(联合 \) \(p. 224\)](#)。

最后，在创建角色后，您可以通过配置包含有关 Amazon 的信息的 IdP 以及希望联合身份用户使用的角色来完成 SAML 信任。这称为在 IdP 和 Amazon 之间配置信赖方信任。要配置信赖方信任，请参阅[配置具有信赖方信任的 SAML 2.0 IdP 并添加陈述](#) (p. 192)。

主题

- [创建和管理 IAM 身份提供程序 \(控制台\)](#) (p. 189)
- [创建和管理 IAM SAML 身份提供程序 \(Amazon CLI\)](#) (p. 190)
- [创建和管理 IAM SAML 身份提供程序 \(Amazon API\)](#) (p. 191)
- [配置具有信赖方信任的 SAML 2.0 IdP 并添加陈述](#) (p. 192)
- [将第三方 SAML 解决方案提供商与 Amazon 集成](#) (p. 192)
- [为身份验证响应配置 SAML 断言](#) (p. 194)

创建和管理 IAM 身份提供程序 (控制台)

您可以使用 Amazon Web Services Management Console 创建和删除 IAM SAML 身份提供程序。

创建 IAM SAML 身份提供程序 (控制台)

1. 在创建 IAM SAML 身份提供程序之前，您需要从 IdP 处获得的 SAML 元数据文档。此文档包括发布者名称、过期信息以及可用于验证从 IdP 处收到的 SAML 身份验证响应 (断言) 的密钥。要生成元数据文档，可使用您的组织用作其 IdP 的身份管理软件。有关如何配置许多可用 IdP 以使用 Amazon (包括如何生成所需的 SAML 元数据文档) 的说明，请参阅[将第三方 SAML 解决方案提供商与 Amazon 集成](#) (p. 192)。

Important

此元数据文件包括颁发者名称、过期信息以及可用于验证从 IdP 处收到的 SAML 身份验证响应 (断言) 的密钥。元数据文件必须采用不含字节顺序标记 (BOM) 的 UTF-8 格式编码。要删除 BOM，您可以使用 Notepad++ 等文本编辑工具以 UTF-8 格式对文件进行编码。

作为 SAML 元数据文档的一部分，X.509 证书必须使用长度至少为 1024 位的密钥。此外，x.509 证书也不能有任何重复的扩展名。您可以使用扩展程序，但扩展程序只能在证书中显示一次。如果 x.509 证书不符合任一条件，则 IdP 将创建失败，并返回 "Unable to parse metadata" 这一错误消息。

2. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
3. 在导航窗格中，选择 Identity providers (身份提供程序)，然后选择 Add provider (添加提供商)。
4. 对于 Configure provider (配置提供商)，选择 SAML。
5. 键入身份提供程序的名称。
6. 对于 Metadata document (元数据文档)，选择 Choose file (选择文件)，指定您在 [Step 1](#) 中下载的 SAML 元数据文档。
7. (可选) 对于 Add tags (添加标签)，您可以添加键值对来帮助识别和组织您的 IdP。您还可以使用标签来控制对 Amazon 资源的访问。要了解有关标记 SAML 身份提供程序的更多信息，请参阅[标记 IAM SAML 身份提供程序](#) (p. 286)。

选择 Add tag (添加标签)。为每个标签键值对输入值。

8. 验证您提供的信息。完成后，选择 Add provider (添加提供商)。
9. 将 IAM 角色分配至身份提供程序，以向身份提供程序管理的外部用户身份授予访问账户中的 Amazon 资源的权限。要了解有关为联合身份创建角色的更多信息，请参阅[针对第三方身份提供程序创建角色 \(联合\)](#) (p. 224)。

删除 SAML 提供商 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。

2. 在导航窗格中，选择 Identity providers (身份提供程序)。
3. 选中要删除的身份提供程序旁边的单选按钮。
4. 选择 Delete。此时会打开一个新窗口。
5. 通过在字段中键入 delete 一词以确认您要删除此提供商。然后选择 Delete (删除)。

创建和管理 IAM SAML 身份提供程序 (Amazon CLI)

您可使用 Amazon CLI 创建和管理 SAML 提供商。

在创建 IAM 身份提供程序之前，您需要从 IdP 处获得的 SAML 元数据文档。此文档包括发布者名称、过期信息以及可用于验证从 IdP 处收到的 SAML 身份验证响应 (断言) 的密钥。要生成元数据文档，可使用您的组织用作其 IdP 的身份管理软件。有关如何配置许多可用 IdP 以使用 Amazon (包括如何生成所需的 SAML 元数据文档) 的说明，请参阅[将第三方 SAML 解决方案提供商与 Amazon 集成 \(p. 192\)](#)。

Important

此元数据文件包括颁发者名称、过期信息以及可用于验证从 IdP 处收到的 SAML 身份验证响应 (断言) 的密钥。元数据文件必须采用不含字节顺序标记 (BOM) 的 UTF-8 格式编码。要删除 BOM，您可以使用 Notepad++ 等文本编辑工具以 UTF-8 格式对文件进行编码。

作为 SAML 元数据文档的一部分，X.509 证书必须使用长度至少为 1024 位的密钥。此外，x.509 证书也不能有任何重复的扩展名。您可以使用扩展程序，但扩展程序只能在证书中显示一次。如果 x.509 证书不符合任一条件，则 IdP 将创建失败，并返回 "Unable to parse metadata" 这一错误消息。

创建 IAM 身份提供程序并上传元数据文档 (Amazon CLI)

- 运行以下命令：`aws iam create-saml-provider`

为 IAM 身份提供程序上传新的元数据文档 (Amazon CLI)

- 运行以下命令：`aws iam update-saml-provider`

要标记现有 IAM 身份提供程序 (Amazon CLI)

- 运行以下命令：`aws iam tag-saml-provider`

要列出现有 IAM 身份提供程序 (Amazon CLI) 的标签

- 运行以下命令：`aws iam list-saml-provider-tags`

要删除现有 IAM 身份提供程序 (Amazon CLI) 的标签

- 运行以下命令：`aws iam untag-saml-provider`

要标记现有 IAM 身份提供程序 (Amazon CLI)

- 运行以下命令：`aws iam tag-saml-provider`

要列出现有 IAM 身份提供程序 (Amazon CLI) 的标签

- 运行以下命令：`aws iam list-saml-provider-tags`

要删除现有 IAM 身份提供程序 (Amazon CLI) 的标签

- 运行以下命令：`aws iam untag-saml-provider`

删除 IAM SAML 身份提供程序 (Amazon CLI)

1. (可选) 要列出所有提供商的信息 (例如 ARN、创建日期和过期时间)，请运行以下命令：

- `aws iam list-saml-providers`

2. (可选) 要获取有关特定提供商的信息 (例如 ARN、创建日期和过期时间)，请运行以下命令：

- `aws iam get-saml-provider`

3. 要删除 IAM 身份提供程序，请运行以下命令：

- `aws iam delete-saml-provider`

创建和管理 IAM SAML 身份提供程序 (Amazon API)

您可使用 Amazon API 创建和管理 SAML 提供商。

在创建 IAM 身份提供程序之前，您需要从 IdP 处获得的 SAML 元数据文档。此文档包括发布者名称、过期信息以及可用于验证从 IdP 处收到的 SAML 身份验证响应 (断言) 的密钥。要生成元数据文档，可使用您的组织用作其 IdP 的身份管理软件。有关如何配置许多可用 IdP 以使用 Amazon (包括如何生成所需的 SAML 元数据文档) 的说明，请参阅[将第三方 SAML 解决方案提供商与 Amazon 集成 \(p. 192\)](#)。

Important

元数据文件必须采用不含字节顺序标记 (BOM) 的 UTF-8 格式编码。此外，作为 SAML 元数据文档的一部分，X.509 证书必须使用长度至少为 1024 位的密钥。如果密钥过短，则 IdP 将创建失败，并显示“无法解析元数据”这一错误消息。要删除 BOM，您可以使用 Notepad++ 等文本编辑工具以 UTF-8 格式对文件进行编码。

创建 IAM 身份提供程序并上传元数据文档 (Amazon API)

- 调用此操作：[CreateSAMLProvider](#)

为 IAM 身份提供程序上传新的元数据文档 (Amazon API)

- 调用此操作：[UpdateSAMLProvider](#)

要标记现有 IAM 身份提供程序 (Amazon API)

- 调用此操作：[TagSAMLProvider](#)

要列出现有 IAM 身份提供程序 (Amazon API) 的标签

- 调用此操作：[ListSAMLProviderTags](#)

要删除现有 IAM 身份提供程序的标签 (Amazon API)

- 调用此操作：[UntagSAMLProvider](#)

要标记现有 IAM 身份提供程序 (Amazon API)

- 调用此操作：[TagSAMLProvider](#)

要列出现有 IAM 身份提供程序 (Amazon API) 的标签

- 调用此操作：[ListSAMLProviderTags](#)

要删除现有 IAM 身份提供程序的标签 (Amazon API)

- 调用此操作：[UntagSAMLProvider](#)

删除 IAM 身份提供程序 (Amazon API)

1. (可选) 要列出所有 IdP 的信息 (例如 ARN、创建日期和过期时间)，请调用以下操作：

- [ListSAMLProviders](#)

2. (可选) 要获取有关特定提供商的信息 (例如 ARN、创建日期和过期时间)，请调用以下操作：

- [GetSAMLProvider](#)

3. 要删除 IdP，请调用以下操作：

- [DeleteSAMLProvider](#)

配置具有信赖方信任的 SAML 2.0 IdP 并添加陈述

当您创建 IAM 身份提供程序和用于 SAML 访问的角色时，您实际上将告知 Amazon 关于外部身份提供程序 (IdP) 的信息以及允许其用户执行的操作。下一步是让 IdP 知道 Amazon 作为服务提供商。这称为在 IdP 和 Amazon 之间添加信赖方信托。添加信赖方信任的具体步骤取决于您使用的 IdP。有关详细信息，请参阅身份管理软件对应的文档。

许多 IdP 允许指定一个 URL，他们可从中读取包含信赖方信息和证书的 XML 文档。对于 Amazon，使用 <https://region-code.signin.aws.amazon.com/static/saml-metadata.xml> 或 <https://signin.aws.amazon.com/static/saml-metadata.xml>。有关可能的 **region-code** 值的列表，请参阅 [Amazon 登录端点](#) 中的 Region (区域) 列。

如果您无法直接指定 URL，请从之前的 URL 下载 XML 文档，然后将其导入您的 IdP 软件。

您还需要在指定 Amazon 作为信赖方的 IdP 中，创建相应的声明规则。当 IdP 向 Amazon 终端节点发送 SAML 响应时，它包括具有一个或多个索赔的 SAML 断言。断言是有关用户及其组的信息。断言规则将该信息映射到 SAML 属性中。这可确保来自 IdP 的 SAML 身份验证响应包含 IAM policy 中 Amazon 用于检查联合身份用户权限的必要属性。有关更多信息，请参阅以下主题：

- [用于允许对 Amazon 资源进行 SAML 联合访问的角色的概述 \(p. 178\)](#) 本主题将讨论如何在 IAM policy 中使用特定于 SAML 的键以及如何使用它们限制 SAML 联合身份用户的权限。
- [为身份验证响应配置 SAML 断言 \(p. 194\)](#) 本主题将讨论如何配置包括用户相关信息的 SAML 陈述。将声明捆绑到 SAML 断言中并包括在发送到 Amazon 的 SAML 响应中。您必须确保 Amazon 策略所需的信息以 Amazon 可识别和使用的形式包括在 SAML 断言中。
- [将第三方 SAML 解决方案提供商与 Amazon 集成 \(p. 192\)](#) 本主题提供了由第三方组织提供的关于如何与 Amazon 集成身份解决方案的文档链接。

将第三方 SAML 解决方案提供商与 Amazon 集成

以下链接将帮助您配置第三方 SAML 2.0 身份提供程序 (IdP) 解决方案以利用 Amazon 联合身份验证。

Note

Amazon Support 工程师可以帮助有商业和企业支持计划的客户完成涉及第三方软件的一些集成任务。有关受支持的平台和应用程序的最新列表，请参阅《Amazon Support 常见问题》中的 [支持哪些第三方软件？](#)。

解决方案	更多信息
Auth0	与 Amazon Web Services 集成 – Auth0 文档网站上的此页面包含了介绍如何使用 Amazon Web Services Management Console 来设置单点登录 (SSO) 的资源的链接，并提供了一个 JavaScript 示例。您可以配置 Auth0 来传递 会话标签 (p. 293) 。有关更多信息，请参阅 Auth0 宣布与 Amazon 合作开发 IAM 会话标签 。
Azure Active Directory (Azure AD)	教程：Azure Active Directory 与 Amazon Web Services 集成 – Microsoft 网站上的此教程介绍了如何使用 SAML 联合身份验证将 Azure AD 设置为身份提供者 (IdP) 。
Centrify	Configure Centrify and Use SAML for SSO to Amazon (配置 Centrify 并对亚马逊云科技使用 SAML SSO) - 此 Centrify 网站页面说明如何配置 Centrify 以对 Amazon 使用 SAML SSO。
CyberArk	配置 CyberArk ，以便为从 CyberArk 用户门户通过 SAML 单点登录(SSO)登入的用户提供 Amazon Web Services (Amazon) 访问。
ForgeRock	ForgeRock Identity Platform 与 Amazon 集成。您可以配置 ForgeRock 来传递 会话标签 (p. 293) 。有关更多信息，请参阅 Amazon Web Services 的基于属性的访问控制 。
Google Workspace	Amazon Web Services 云应用程序 - 这篇有关 Google Workspace Admin Help 站点的文章描述了如何将 Google Workspace 配置为 SAML 2.0 IdP，并使用 Amazon 作为服务提供程序。
IBM	您可以配置 IBM 来传递 会话标签 (p. 293) 。有关更多信息，请参阅 IBM Cloud Identity IDaaS (最早支持 Amazon 会话标签的解决方案之一) 。
Matrix42	MyWorkspace 入门指南 - 本指南介绍如何将 Amazon 身份服务与 Matrix42 MyWorkspace 集成。
Microsoft Active Directory 联合身份验证服务 (AD FS)	现场记录：将 Active Directory 联合身份验证服务与 Amazon IAM Identity Center (successor to Amazon Single Sign-On) 集成 – Amazon 架构博客上的这篇文章介绍了 AD FS 与 Amazon IAM Identity Center (successor to Amazon Single Sign-On) (IAM Identity Center) 之间的身份验证流。IAM Identity Center 通过 SAML 2.0 支持身份联合验证，从而得以与 AD FS 解决方案集成。用户可使用其企业凭证登录 IAM Identity Center 门户，从而减少在 IAM Identity Center 上维护单独凭证的管理开销。您也可以配置 AD FS 来传递会话标签 (p. 293)。有关更多信息，请参阅将基于属性的访问控制与 AD FS 结合使用来简化 IAM 权限管理。 PowerShell Automation to Give Amazon Console Access - Sivaprasad Padisetty 博客上的这篇文章介绍如何使用 Windows PowerShell 自动设置 Active Directory 和 AD FS。它还包括通过 Amazon 启用 SAML 联合身份。
miniOrange	SSO for Amazon - miniOrange 网站上的此页面介绍如何为企业建立对 Amazon 的安全访问以及对 Amazon 应用程序访问的完全控制。

解决方案	更多信息
Okta	使用 Okta 集成 Amazon Web Services Command Line Interface - 通过 Okta 支持网站上的此页面可以了解如何配置 Okta 才能与 Amazon 配合使用。您可以配置 Okta 来传递 会话标签 (p. 293) 。有关更多信息，请参阅 Okta 与 Amazon 合作，通过会话标签简化访问 。
Okta	如何为 Amazon IAM Identity Center (successor to Amazon Single Sign-On) 配置 SAML 2.0 - Okta 网站上的这篇文章介绍了如何为 Amazon 设置和启用 IAM Identity Center。
OneLogin	从 OneLogin 知识库 中，搜索 SAML Amazon 以获取文章列表，这些文章介绍了如何在 OneLogin 与 Amazon 之间设置 IAM Identity Center 功能，以实现单角色和多角色方案。您可以配置 OneLogin 来传递 会话标签 (p. 293) 。有关更多信息，请参阅 OneLogin 和会话标签：Amazon 资源的基于属性的访问控制 。
Ping Identity	PingFederate Amazon Connector - 查看有关 PingFederate Amazon Connector 的详细信息，快速连接模板来轻松设置单点登录 (SSO) 和预配置连接。阅读文档，并下载最新的 PingFederate Amazon Connector，以便与 Amazon 集成。您可以配置 Ping Identity 来传递 会话标签 (p. 293) 。有关更多信息，请参阅 宣布推出针对 Amazon 中的基于属性的访问控制的 Ping 身份支持 。
RadiantLogic	Radiant Logic 技术合作伙伴 - Radiant Logic 的 RadiantOne Federated Identity Service 与 Amazon 进行了集成，可以为基于 SAML 的 SSO 提供身份中心。
RSA	RSA Link 是一个在线社区，促进信息共享和讨论。您可以配置 RSA 来传递 会话标签 (p. 293) 。有关更多信息，请参阅 使用 RSA SecurID 和会话标签简化 Amazon 上的身份访问和保障决策 。
Salesforce.com	How to configure SSO from Salesforce to Amazon - Salesforce.com 开发人员站点上的这篇方法文章介绍如何在 Salesforce 中设置身份提供程序 (IdP) 以及如何将 Amazon 配置为服务提供商。
SecureAuth	Amazon - SecureAuth SAML SSO - SecureAuth 网站上的该文章介绍了如何设置 SAML 以与 SecureAuth 设备的 Amazon 集成在一起。
Shibboleth	如何使用 Shibboleth 对 Amazon Web Services Management Console 进行 SSO - Amazon 安全博客中的这篇文章提供分步教程，介绍如何设置 Shibboleth 并将它配置为 Amazon 的身份提供程序。您可以配置 Shibboleth 来传递 会话标签 (p. 293) 。

有关更多详细信息，请参阅 Amazon 网站上的 [IAM 合作伙伴](#) 页面。

为身份验证响应配置 SAML 断言

在您验证组织中的用户身份后，外部身份提供者 (IdP) 将向 Amazon SAML 端点 (<https://region-code.signin.aws.amazon.com/saml>) 发送身份验证响应。有关潜在 [region-code](#) 替换的列表，请参阅 [Amazon 登录端点](#) 中的 Region (区域) 列。此响应是一个包含 SAML 令牌的 POST 请求，该令牌遵循[适用于 SAML 2.0 的 HTTP POST 绑定标准](#)，其中包含以下元素或断言。您可在与 SAML 兼容的 IdP 中配置这些断言。请参考 IdP 文档，以了解有关如何输入这些声明的说明。

当 IdP 将包含断言的响应发送到 Amazon 时，许多传入断言将映射到 Amazon 上下文密钥。可以在 IAM policy 中使用 Condition 元素检查这些上下文密钥。可用映射的列表如[将 SAML 属性映射到 Amazon 信任策略上下文密钥 \(p. 198\)](#)部分中所示。

Subject-和-NameID

以下摘录显示了一个示例。用您自己的值替代标记值。同时包含 SubjectConfirmation 和 SubjectConfirmationData 属性的 NotOnOrAfter 元素必须恰好具有一个 Recipient 元素。这些属性包含的值必须与 Amazon 端点 (<https://region-code.signin.aws.amazon.com/saml>) 匹配。有关可能的 [region-code](#) 值的列表，请参阅 [Amazon 登录端点](#) 中的 Region (区域) 列。对于 Amazon 值，您也可以使用 <https://signin.aws.amazon.com/static/saml>，如以下示例所示。有关支持用于单一登录交互的名称标识符格式的信息，请参阅 [Oracle Sun OpenSSO Enterprise 管理参考](#)。

```
<Subject>
  <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">_cbb88bf52c2510eabe00c1642d4643f41430fe25e3</NameID>
  <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
    <SubjectConfirmationData NotOnOrAfter="2013-11-05T02:06:42.876Z" Recipient="https://signin.aws.amazon.com/saml"/>
  </SubjectConfirmation>
</Subject>
```

AudienceRestriction-和-Audience

出于安全原因，Amazon 应作为受众包含在您的 IdP 发送到 Amazon 的 SAML 断言中。对于 Audience 元素的值，请指定 <https://region-code.signin.aws.amazon.com/saml> 或 urn:amazon:webservices。有关可能的 [region-code](#) 值的列表，请参阅 [Amazon 登录端点](#) 中的 Region (区域) 列。对于 Audience，您也可以使用值：<https://signin.aws.amazon.com/static/saml>。以下 SAML 断言中的示例 XML 代码段展示了 IdP 如何指定此密钥。包括适用于您的使用案例的示例。

```
<Conditions>
  <AudienceRestriction>
    <Audience>https://signin.aws.amazon.com/saml</Audience>
  </AudienceRestriction>
</Conditions>
```

```
<Conditions>
  <AudienceRestriction>
    <Audience>urn:amazon:webservices</Audience>
  </AudienceRestriction>
</Conditions>
```

Important

来自 IdP 的 SAML 断言中的 SAML AudienceRestriction 值不会映射到您可在 IAM policy 中测试的 saml:aud 上下文密钥。相反，saml:aud 上下文密钥来自 SAML recipient 属性，因为等同于 OIDC 受众字段 (例如 accounts.google.com:aud) 的 SAML。

SAML PrincipalTagAttribute

(可选) 您可使用将 Name 属性设置为 <https://aws.amazon.com/SAML/Attributes/PrincipalTag:{TagKey}> 的 Attribute 元素。此元素允许您将属性作为 SAML 断言中的会话标签传递。有关会话标签的更多信息，请参阅 [在 Amazon STS 中传递会话标签 \(p. 293\)](#)。

要将属性作为会话标签传递，请包含指定标签值的 AttributeValue 元素。例如，要传递标签键/值对 Project = Marketing 和 CostCenter = 12345，请使用以下属性。为每个标签包含一个单独的 Attribute 元素。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Project">
  <AttributeValue>Marketing</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:CostCenter">
  <AttributeValue>12345</AttributeValue>
</Attribute>
```

要将上述标签设置为可传递，请包含另一个 Attribute 元素并将 Name 属性设置为 `https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys`。这是一个可选的多值属性，可将您的会话标签设置为可传递。当您使用 SAML 会话代入 Amazon 中的另一个角色时，可传递标签将保留。这称为[角色链 \(p. 159\)](#)。例如，要将 Principal 和 CostCenter 标签均设置为可传递，请使用以下属性指定键。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys">
  <AttributeValue>Project</AttributeValue>
  <AttributeValue>CostCenter</AttributeValue>
</Attribute>
```

SAML RoleAttribute

您可使用将 Name 属性设置为 `https://aws.amazon.com/SAML/Attributes/Role` 的 Attribute 元素。此元素包含一个或多个 AttributeValue 元素，这些元素可列出用户通过 IdP 映射到的 IAM 身份提供程序和角色。IAM 角色和 IAM 身份提供程序指定为逗号分隔的 ARN 对，格式与传递到 [AssumeRoleWithSAML](#) 的 RoleArn 和 PrincipalArn 参数相同。此元素必须至少包含一个角色/提供商对 (AttributeValue 元素)，也可以包含多个对。如果该元素包含多个角色/提供商对，则用户将需要选择其使用 WebSSO 登录 Amazon Web Services Management Console 时要代入的角色。

Important

Name 标签中 Attribute 属性的值区分大小写。必须将其一字不差地设置为 `https://aws.amazon.com/SAML/Attributes/Role`。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/Role">
  <AttributeValue>arn:aws:iam::account-number:role/role-name1,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
  <AttributeValue>arn:aws:iam::account-number:role/role-name2,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
  <AttributeValue>arn:aws:iam::account-number:role/role-name3,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
</Attribute>
```

SAML RoleSessionNameAttribute

您可使用将 Name 属性设置为 `https://aws.amazon.com/SAML/Attributes/RoleSessionName` 的 Attribute 元素。该元素包含一个为临时凭证提供标识符的 AttributeValue 元素，这些凭证是针对 SSO 颁发的。您可以使用此选项将临时凭证与正在使用您应用程序的用户相关联。该元素用于在 Amazon Web Services Management Console 中显示用户信息。AttributeValue 元素中的值长度必须介于 2 到 64 个字符之间，只能包含字母数字字符、下划线和以下字符：`.,+=@-` (连字符)。它不能含有空格。该值通常是用户 ID (johndoe) 或电子邮件地址 (johndoe@example.com)。该值不应包含空格，如用户的显示名称 (John Doe)。

Important

Name 标签中 Attribute 属性的值区分大小写。必须将其一字不差地设置为 `https://aws.amazon.com/SAML/Attributes/RoleSessionName`。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/RoleSessionName">
  <AttributeValue>user-id-name</AttributeValue>
</Attribute>
```

SAML SessionDurationAttribute

(可选) 您可使用将 Name 属性设置为 `https://aws.amazon.com/SAML/Attributes/SessionDuration` 的 Attribute 元素。该元素包含一个 AttributeValue 元素, 它指定用户在必须请求新的临时凭证之前可访问 Amazon Web Services Management Console 的时间长度。该值是一个表示会话秒数的整数。该值的范围是 900 秒 (15 分钟) 到 43200 秒 (12 小时)。如果该属性不存在, 则凭证持续时间为 1 小时 (DurationSeconds API 的 AssumeRoleWithSAML 参数的默认值)。

要使用该属性, 您必须配置 SAML 提供商, 以通过控制台登录 Web 终端节点 (`https://region-code.signin.aws.amazon.com/saml`) 提供对 Amazon Web Services Management Console 的单一登录访问。有关可能的 **region-code** 值的列表, 请参阅 [Amazon 登录端点](#) 中的 Region (区域) 列。您可以选择使用以下 URL: `https://signin.aws.amazon.com/static/saml`。请注意, 该属性仅将会话扩展到 Amazon Web Services Management Console。它不能延长其他凭证的有效期。但是, 如果它存在于 AssumeRoleWithSAML API 调用中, 则可以用来缩短会话的持续时间。调用返回的凭证的默认生命周期为 60 分钟。

另请注意, 如果还定义了 SessionNotOnOrAfter 属性, 则两个属性的 lesser (较小值) (SessionDuration 或 SessionNotOnOrAfter) 将建立控制台会话的最大持续时间。

在启用具有更长持续时间的控制台会话时, 可能产生凭证外泄的风险。为了帮助缓解这种风险, 您可以通过在 IAM 控制台页面的 Role Summary (角色摘要) 上选择 Revoke Sessions (撤销会话) 立即禁用所有角色的有效控制台会话。有关更多信息, 请参阅 [撤销 IAM 角色临时安全凭证 \(p. 259\)](#)。

Important

Name 标签中 Attribute 属性的值区分大小写。必须将其一字不差地设置为 `https://aws.amazon.com/SAML/Attributes/SessionDuration`。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SessionDuration">
  <AttributeValue>1800</AttributeValue>
</Attribute>
```

SAML SourceIdentityAttribute

(可选) 您可使用将 Name 属性设置为 `https://aws.amazon.com/SAML/Attributes/SourceIdentity` 的 Attribute 元素。该元素包含一个 AttributeValue 元素, 为使用 IAM 角色的人员或应用程序提供标识符。当您使用 SAML 会话代入 Amazon 中的另一个角色时, 源身份的值将保留, 称为 [角色链 \(p. 159\)](#)。源身份的值存在于对角色会话期间执行的每个操作的请求中。在角色会话期间无法更改已设置的值。然后, 管理员可以使用 Amazon CloudTrail 日志来监控和审计源身份信息, 以确定谁在使用共享角色执行操作。

AttributeValue 元素中的值长度必须介于 2 到 64 个字符之间, 只能包含字母数字字符、下划线和以下字符: ., + = @ - (连字符)。它不能含有空格。该值通常是与用户关联的属性, 例如用户 ID (johndoe) 或电子邮件地址 (johndoe@example.com)。该值不应包含空格, 如用户的显示名称 (John Doe)。有关使用基于身份的策略的更多信息, 请参阅 [监控和控制使用所担任角色执行的操作 \(p. 316\)](#)。

Important

如果您的 SAML 断言配置为使用 [SourceIdentity \(p. 197\)](#) 属性, 则您的角色信任策略还必须包含 sts:SetSourceIdentity 操作, 否则代入角色操作会失败。有关使用基于身份的策略的更多信息, 请参阅 [监控和控制使用所担任角色执行的操作 \(p. 316\)](#)。

要传递源身份属性, 请包含 AttributeValue 元素, 指定源身份的值。例如, 要传递源身份 DiegoRamirez 使用以下属性。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SourceIdentity">
  <AttributeValue>DiegoRamirez</AttributeValue>
```


将 SAML 属性映射到 Amazon 信任策略上下文密钥

本部分中的表列出了常用的 SAML 属性以及它们在 Amazon 中映射到信任策略条件上下文密钥的方式。您可以使用这些键来控制对角色的访问。为此，请将键与 SAML 访问请求中所附带断言包含的值进行比较。

Important

这些键仅在 IAM 信任策略（可决定谁能担任角色的策略）中可用，并且不适用于权限策略。

在 eduPerson 和 eduOrg 属性表中，值是以字符串或字符串列表的形式键入的。对于字符串值，您可以使用 StringEquals 或 StringLike 条件测试 IAM 信任策略中的这些值。对于包含字符串列表的值，您可以使用 ForAnyValue 和 ForAllValues [策略集运算符 \(p. 958\)](#) 测试信任策略中的这些值。

Note

每个 Amazon 上下文密钥只能包含一个断言。如果包含多个声明，将仅映射其中一个。

eduPerson 和 eduOrg 属性

eduPerson 或 eduOrg 属性 (Name 密钥)	映射至此 AWS 上下文密钥 (FriendlyName 密钥)	类型
urn:oid:1.3.6.1.4.1.5923.1.1.1.1	eduPersonAffiliation	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.1.1.2	eduPersonNickname	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.1.1.3	eduPersonOrgDN	字符串
urn:oid:1.3.6.1.4.1.5923.1.1.1.4	eduPersonOrgUnitDN	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.1.1.5	eduPersonPrimaryAffiliation	字符串
urn:oid:1.3.6.1.4.1.5923.1.1.1.6	eduPersonPrincipalName	字符串
urn:oid:1.3.6.1.4.1.5923.1.1.1.7	eduPersonEntitlement	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.1.1.8	eduPersonPrimaryOrgUnit	字符串
urn:oid:1.3.6.1.4.1.5923.1.1.1.9	eduPersonScopedAffiliation	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.1.1.10	eduPersonTargetedID	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.1.1.11	eduPersonAssurance	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.2.1.2	eduOrgHomePageURI	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.2.1.3	eduOrgIdentityAuthNPOL	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.2.1.4	eduOrgLegalName	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.2.1.5	eduOrgSuperiorURI	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.2.1.6	eduOrgWhitePagesURI	字符串列表
urn:oid:2.5.4.3	cn	字符串列表

Active Directory 属性

AD 属性	映射到此 Amazon 上下文密钥	类型
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name	name	字符串

AD 属性	映射到此 Amazon 上下文密钥	类型
http://schemas.xmlsoap.org/claims/CommonName	commonName	字符串
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname	givenName	字符串
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname	surname	字符串
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress	mail	字符串
http://schemas.microsoft.com/ws/2008/06/identity/claims/primarygroupsid	uid	字符串

X.500 属性

X.500 属性	映射到此 Amazon 上下文密钥	类型
2.5.4.3	commonName	字符串
2.5.4.4	surname	字符串
2.4.5.42	givenName	字符串
2.5.4.45	x500UniqueIdentifier	字符串
0.9.2342.19200300100.1.1	uid	字符串
0.9.2342.19200300100.1.3	mail	字符串
0.9.2342.19200300.100.1.45	organizationStatus	字符串

使 SAML 2.0 联合身份用户能够访问 Amazon Web Services Management Console

您可使用一个角色配置符合 SAML 2.0 标准的身份提供程序 (IdP) 和 Amazon 以允许您的联合身份用户访问 Amazon Web Services Management Console。该角色为用户授予了在控制台中执行任务的权限。如果您希望为 SAML 联合身份用户提供访问 Amazon 的其他方式，请参阅以下其中一项主题：

- Amazon CLI: [切换到 IAM 角色 \(Amazon CLI\) \(p. 245\)](#)
- Tools for Windows PowerShell: [切换到 IAM 角色 \(Tools for Windows PowerShell\) \(p. 249\)](#)
- Amazon API : [切换到 IAM 角色 \(Amazon API\) \(p. 250\)](#)

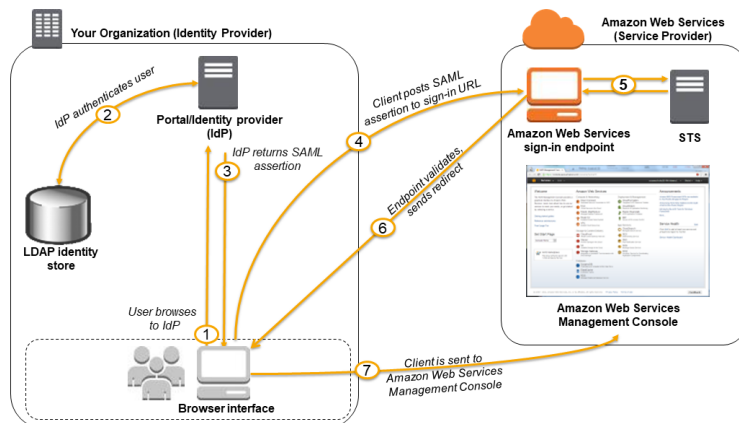
概览

下图说明了启用了 SAML 的单一登录的流程。

Note

SAML 的这种特定用途与[关于基于 SAML 2.0 的联合身份验证 \(p. 176\)](#)中所述的更常规的用途不同，因为此工作流程将代表用户打开 Amazon Web Services Management Console。这需要使用

Amazon 登录端点，而不是直接调用 AssumeRoleWithSAML API。该终端节点将为用户调用 API 并返回将用户的浏览器自动重定向到 Amazon Web Services Management Console 的 URL。



下图说明了以下步骤：

1. 用户浏览到您的组织的门户网站，并选择转到 Amazon Web Services Management Console 的选项。在企业中，门户网站通常充当处理您企业与 Amazon 之间的信任交换的 IdP。例如，在 Active Directory 联合身份验证服务中，门户网站 URL 为：`https://ADFSServiceName/adfs/ls/IdpInitiatedSignOn.aspx`
2. 该门户网站可验证您的组织用户的身份。
3. 该门户网站生成一个 SAML 身份验证响应，其中包括识别用户身份的断言以及用户的相关属性。您也可以配置 IdP 以包含一个名为 `SessionDuration` 的 SAML 断言属性，该属性指定控制台会话的有效时间长度。您还可以配置 IdP，将属性作为 [会话标签](#) (p. 293) 传递。该门户网站将此响应发送到客户端浏览器。
4. 该客户端浏览器将被重定向到 Amazon 单一登录终端节点并发布 SAML 断言。
5. 终端节点将代表用户请求临时安全凭证，并创建一个使用这些凭证的控制台登录 URL。
6. Amazon 将登录 URL 作为重定向发回客户端。
7. 该客户端浏览器将重定向到 Amazon Web Services Management Console。如果 SAML 身份验证响应包含映射到多个 IAM 角色的属性，则系统将首先提示用户选择角色以访问控制台。

从用户的角度来看，整个流程以透明的方式进行：用户在您的企业的内部门户网站开始操作，在 Amazon Web Services Management Console 结束操作，无需提供任何 Amazon 凭证。

有关如何配置此行为的概述以及指向详细步骤的链接，请参阅以下章节。

将网络配置为适用于 Amazon 的 SAML 提供商

在组织的网络中，配置身份存储（例如 Windows Active Directory）以使用基于 SAML 的 IdP，例如 Windows Active Directory 联合身份验证服务、Shibboleth 等。通过使用 IdP，可以生成一个元数据文档，此文档将您的组织描述为 IdP 并且包含身份验证密钥。另外，还要将企业的门户网站配置为将访问 Amazon Web Services Management Console 的用户请求路由至 Amazon SAML 终端节点，以便使用 SAML 断言进行身份验证。如何配置您的 IdP 来生成 `metadata.xml` 文件取决于您的 IdP。请参阅您的 IdP 文档以获得指示，或参阅 [将第三方 SAML 解决方案提供商与 Amazon 集成](#) (p. 192) 以获得指向很多支持的 SAML 提供商 Web 文档。

在 IAM 中创建 SAML 提供商

然后，登录 Amazon Web Services Management Console 并转至 IAM 控制台。在此，您创建一个新的 SAML 提供商，这是在 IAM 中包含您的企业的 IdP 的相关信息的实体。在此过程中，您可以上传在上一节中由贵组织中的 IdP 软件生成的元数据文档。有关详细信息，请参阅 [创建 IAM SAML 身份提供程序](#) (p. 188)。

在 Amazon 中为您的联合身份用户配置权限

下一步是创建一个 IAM 角色，以在 IAM 与您企业的 IdP 之间建立信任关系。该角色必须将您的 IdP 标识为主体（可信实体）以实现联合身份验证目的。该角色还定义了由您的组织的 IdP 进行身份验证的用户可以在 Amazon 中执行的操作。您可使用 IAM 控制台创建该角色。在创建指示谁可以代入角色的信任策略时，您可以将先前创建的 SAML 提供商指定为 IAM。您还可以指定用户要代入角色必须匹配的一个或多个 SAML 属性。例如，您可以指定只允许其 SAML `eduPersonOrgDN` 值为 `ExampleOrg` 的用户登录。角色向导会自动添加一个测试 `saml:aud` 属性的条件，以确保仅出于登录 Amazon Web Services Management Console 的目的担任该角色。该角色的信任策略可能如下所示：

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": { "Federated": "arn:aws:iam::account-id:saml-provider/ExampleOrgSSOProvider" },
    "Action": "sts:AssumeRoleWithSAML",
    "Condition": { "StringEquals": {
      "saml:edupersonorgdn": "ExampleOrg",
      "saml:aud": "https://signin.aws.amazon.com/saml"
    } }
  ]
}
```

您可以在 `saml:aud` 中包含 `https://region-code.signin.aws.amazon.com/static/saml-metadata.xml` 属性的区域端点。有关可能的 `region-code` 值的列表，请参阅 [Amazon 登录端点](#) 中的 Region（区域）列。

对于该角色中的 [权限策略](#) (p. 355)，您可以像任何角色、用户或组一样指定权限。例如，如果允许您的企业中的用户管理 Amazon EC2 实例，您可以在权限策略中显式允许 Amazon EC2 操作。您可通过分配一个 [托管策略](#) (p. 451) (如 Amazon EC2 完全访问托管策略) 执行此操作。

有关创建用于 SAML IdP 的角色的详细信息，请参阅 [创建用于 SAML 2.0 联合身份验证的角色（控制台）](#) (p. 231)。

完成配置并创建 SAML 断言

通过安装位于 `https://region-code.signin.aws.amazon.com/static/saml-metadata.xml` 或 `https://signin.aws.amazon.com/static/saml-metadata.xml` 的 `saml-metadata.xml` 文件，通知 SAML IdP Amazon 是您的服务提供商。有关可能的 `region-code` 值的列表，请参阅 [Amazon 登录端点](#) 中的 Region（区域）列。

安装该文件的方式取决于您的 IdP。一些提供商为您提供了键入该 URL 的选项，此时，IdP 将为您获取并安装该文件。另一些提供商则要求您从该 URL 处下载该文件，然后将其作为本地文件提供。请参阅您的 IdP 文档以获得详细信息或参阅 [将第三方 SAML 解决方案提供商与 Amazon 集成](#) (p. 192) 以获得指向很多支持的 SAML 提供商的 Web 文档。

您还可配置一条信息，您希望 IdP 在身份验证响应期间将此信息作为 SAML 属性传递到 Amazon。这些信息的大部分在 Amazon 中显示为条件上下文密钥，您可以在策略中进行评估。这些条件键确保仅向正确上下文中的授权用户授予权限来访问您的 Amazon 资源。您可以指定限制何时可使用控制台的时段。您还可以指定在必须刷新用户的凭证之前用户可访问控制台的最长时间（最多 12 个小时）。有关详细信息，请参阅 [身份验证响应配置 SAML 断言](#) (p. 194)。

使自定义身份代理能够访问 Amazon 控制台

您可以通过编写和运行代码来创建 URL 以使登录到您组织网络的用户能够安全访问 Amazon Web Services Management Console。该 URL 包含您从 Amazon 获得的登录令牌，而令牌则用于对访问 Amazon 的用户进行身份验证。

Note

如果您的组织使用与 SAML 兼容的身份提供程序 (IdP)，则无需编写代码即可设置对控制台的访问权限。这适用于 Microsoft 的 Active Directory 联合身份验证服务或开源 Shibboleth 等提供商。有关详细信息，请参阅[使用 SAML 2.0 联合身份用户能够访问 Amazon Web Services Management Console \(p. 199\)](#)。

要使您的组织的用户能够访问 Amazon Web Services Management Console，可以创建执行以下步骤的自定义身份代理：

1. 确认您的本地身份系统已对用户进行身份验证。
2. 调用 Amazon Security Token Service (Amazon STS) [AssumeRole](#) (推荐) 或 [GetFederationToken](#) API 操作为该用户获取临时安全凭证。要了解在担任角色时使用的各种方法，请参阅[使用 IAM 角色 \(p. 236\)](#)。要了解在获取安全凭证时如何传递可选的会话标签，请参阅[在 Amazon STS 中传递会话标签 \(p. 293\)](#)。
 - 如果使用某个 `AssumeRole*` API 操作获取角色的临时安全凭证，您可以在调用中包含 `DurationSeconds` 参数。该参数指定 900 秒 (15 分钟) 到角色的最大会话持续时间设置之间的角色会话持续时间。在 `AssumeRole*` 操作中使用 `DurationSeconds` 时，必须以具有长期凭证的 IAM 用户身份调用该操作。否则，在步骤 3 中对联合终端节点的调用将失败。要了解如何查看或更改角色的最大值，请参阅[查看角色的最大会话持续时间设置 \(p. 238\)](#)。
 - 如果使用 `GetFederationToken` API 操作获取凭证，您可以在调用中包含 `DurationSeconds` 参数。该参数指定您的角色会话的持续时间。该值的范围是 900 秒 (15 分钟) 到 129,600 秒 (36 小时)。您只能使用 IAM 用户的长期 Amazon 安全凭证进行该 API 调用。您还可以使用 Amazon Web Services 账户根用户凭证进行这些调用，但我们不建议使用这种方法。如果您以根用户身份进行该调用，则会话默认持续 1 小时。或者，您可以指定 900 秒 (15 分钟) 到 3,600 秒 (1 小时) 之间的会话。
3. 调用 Amazon 联合终端节点并提供临时安全凭证来请求登录令牌。
4. 构造包含该令牌的控制台 URL：
 - 如果在 URL 中使用某个 `AssumeRole*` API 操作，您可以包含 `SessionDuration` HTTP 参数。该参数指定控制台会话持续时间，范围是 900 秒 (15 分钟) 到 43200 秒 (12 小时)。
 - 如果在 URL 中使用 `GetFederationToken` API 操作，您可以包含 `DurationSeconds` 参数。该参数指定联合控制台会话的持续时间。该值的范围是 900 秒 (15 分钟) 到 129,600 秒 (36 小时)。

Note

如果使用 `GetFederationToken` 获取临时凭证，请不要使用 `SessionDuration` HTTP 参数。这会导致操作失败。

5. 将 URL 分配给用户或代表用户调用 URL。

联合终端节点提供的 URL 在创建后的 15 分钟内有效。这不同于与 URL 关联的临时安全凭证会话的持续时间 (以秒为单位)。这些凭证在创建时指定的持续时间内有效，从创建时算起。

Important

如果您在关联的临时安全凭证中启用了权限，则该 URL 授予通过 Amazon Web Services Management Console 访问您的 Amazon 资源的权限。因此，您应该视 URL 为机密。我们建议您通过安全的重定向返回 URL，例如，在 SSL 连接上使用 302 HTTP 响应状态代码。有关 302 HTTP 响应状态代码的详细信息，请参阅 [RFC 2616](#)，第 10.3.3 节。

要查看说明了如何实现单一登录解决方案的示例应用程序，请参阅 Amazon 示例代码和库中的 [Amazon Web Services Management Console 联合代理示例使用案例](#)。

要完成这些任务，您可以使用适用于 Amazon Identity and Access Management 的 [HTTPS 查询 API \(IAM\)](#) 和 [Amazon Security Token Service \(Amazon STS\)](#)。或者，您可以将编程语言 (例如 Java、Ruby 或 C#) 与相应的 [Amazon 开发工具包](#) 结合使用。下节一一叙述了这些方式。

主题

- [使用 IAM 查询 API 操作的示例代码 \(p. 203\)](#)

- [使用 Python 的示例代码 \(p. 205\)](#)
- [使用 Java 的示例代码 \(p. 205\)](#)
- [演示如何构造 URL 的示例 \(Ruby\) \(p. 207\)](#)

使用 IAM 查询 API 操作的示例代码

您可以构造一个 URL，此 URL 向您的联合身份用户授予对 Amazon Web Services Management Console 的直接访问权限。此任务使用 IAM 和 Amazon STS HTTPS 查询 API。有关提出查询请求的详细信息，请参阅[提出查询请求](#)。

Note

以下过程包括了文本字符串的例子。为增加可读性，一些较长的例子中添加了换行符。如果您要创建并使用这些字符串，必须省略所有换行符。

向联合身份用户授予从 Amazon Web Services Management Console 访问您的资源的权限

1. 在您的身份验证系统里验证该用户
2. 为用户获取临时安全凭证。临时凭证由访问密钥 ID、秘密访问密钥和会话令牌组成。有关创建临时凭证的详细信息，请参阅[IAM 临时安全凭证 \(p. 300\)](#)。

要获取临时凭证，您可以调用 Amazon STS [AssumeRole](#) API（推荐）或 [GetFederationToken](#) API。有关这些 API 操作之间区别的更多信息，请参阅 Amazon 安全博客中的[了解安全地委派您的 Amazon 账户访问权限的 API 选项](#)。

Important

在使用 [GetFederationToken](#) API 创建临时安全凭证时，您必须指定这些凭证为担任角色的用户授予的权限。对于任何以 `AssumeRole*` 开头的 API 操作，可使用 IAM 角色来分配权限。对于其他 API 操作，机制因 API 而异。有关更多信息，请参阅[控制临时安全凭证的权限 \(p. 313\)](#)。此外，如果使用 `AssumeRole*` API 操作，您还必须以具有长期凭证的 IAM 用户身份调用这些操作。否则，在步骤 3 中对联合终端节点的调用将失败。

3. 在获取临时安全凭证后，将其构建到 JSON 会话字符串以将其交换为登录令牌。下例展示如何将凭证编码。请将占位符文本替换为上一步骤中接收的凭证中的相应值。

```
{"sessionId": "*** temporary access key ID ***",  
"sessionKey": "*** temporary secret access key ***",  
"sessionToken": "*** session token ***"}
```

4. [URL 编码](#)上一步骤中的会话字符串。由于您编码的信息是敏感信息，因此建议您避免对此编码使用 Web 服务。请改用开发工具包中在本地安装的函数或功能来安全地编码这些信息。您可以使用 Python 的 `urllib.quote_plus` 函数、Java 的 `URLEncoder.encode` 函数或 Ruby 的 `CGI.escape` 函数。请参阅本主题后面的示例。
5. Note

Amazon 在此处支持 POST 请求。

将您的请求发送到 Amazon 联合身份验证端点：

`https://region-codesignin.amazonaws.cn/federation`

有关可能的 **region-code** 值的列表，请参阅 [Amazon 登录端点](#) 中的 Region（区域）列。您可以选择使用默认 Amazon 登录联合身份验证端点：

`https://signin.amazonaws.cn/federation`

该请求必须包含 Action 和 Session 参数；(可选) 如果使用了 [AssumeRole*](#) API 操作，还必须包含 SessionDuration HTTP 参数，如下示例中所示。


```
Action = getSignInToken
SessionDuration = time in seconds
Session = *** the URL encoded JSON string created in steps 3 & 4 ***
```

Note

此步骤中的以下说明仅适用于 GET 请求。

SessionDuration HTTP 参数指定控制台会话的持续时间。它不同于使用 DurationSeconds 参数指定的临时凭证的持续时间。可以将 SessionDuration 的最大值指定为 43200 (12 小时)。如果缺少 SessionDuration 参数,则会话默认为在步骤 2 中从 Amazon STS 检索的凭证的持续时间(默认为 1 小时)。有关如何使用 DurationSeconds 参数指定持续时间的详细信息,请参阅 [AssumeRole API 的文档](#)。创建超过 1 小时的控制台会话的功能是联合终端节点的 getSignInToken 操作所固有的。

Note

如果使用 GetFederationToken 获取临时凭证,请不要使用 SessionDuration HTTP 参数。这会导致操作失败。

在启用具有更长持续时间的控制台会话时,您增加了凭证外泄的风险。为了帮助缓解这种风险,您可以通过在 IAM 控制台页面的 Role Summary (角色摘要)上选择 Revoke Sessions (撤销会话)立即禁用所有角色的有效控制台会话。有关更多信息,请参阅[撤销 IAM 角色临时安全凭证 \(p. 259\)](#)。

以下为请求具体形式的示例。在此处换行以便阅读,但您应将其作为一行字符串提交。

```
https://signin.amazonaws.cn/federation
?Action=getSignInToken
&SessionDuration=1800
&Session=%7B%22sessionId%22%3A+%22ASIAJUMHIZPTOKTBMK5A%22%2C+%22sessionKey%22%3A+%22LSD7LWI%2FL%2FN%2BgYpan5QFz0XUpC8s7HYjRsgcsrsm%22%2C+%22sessionToken%22%3A+%22FQoDYXdzEBQaDLbj3VWv2u50NN%2F3yyLSASwYtWhPnGPMNmzZFfZsL0Qd3vtYHw5A5dWAjOsrkdPkgHomIe3mJip5%2F0djDBbo7SmO%2FENDEiCdpsQKodTpleKA8xQq0CwFg6a69xdEBQT8FipATnLbKoyS4b%2FebhnsTUjZZQWp0wXXqFF7gSm%2FMe2tXe0jzsdP0012obez9lijPSdF1k2b5PfGhiuyAR9aD5%2BubM0pY86fKex1qsytjvyTbZ9nXe6DvxVDcnCohOGETJ7XFkSFdH0v%2FYR25CUAhJ3nXIkIbG7Ucv9cOEpCf%2Fg23ijRgILIBQ%3D%3D%22%7D
```

来自联合终端节点的响应是一个具有 SignInToken 值的 JSON 文档。类似下例。

```
{"SignInToken": "*** the SignInToken string ***"}
```

6. Note

Amazon 在此处支持 POST 请求。

最后,创建联合身份用户可用于访问 Amazon Web Services Management Console 的 URL。此 URL 与您在 [Step 5 \(p. 203\)](#) 中使用的联合 URL 终端节点相同,外加以下参数:

```
?Action = login
&Issuer = *** the form-urlencoded URL for your internal sign-in page ***
&Destination = *** the form-urlencoded URL to the desired Amazon console page ***
&SignInToken = *** the value of SignInToken received in the previous step ***
```

Note

此步骤中的以下说明仅适用于 GET API。

下列 URL 为 URL 最终形式的示例。URL 的有效期为 15 分钟，自创建时算起。在 URL 中嵌入的临时安全凭证和控制台会话的有效期为在最初请求它们时在 SessionDuration HTTP 参数中指定的持续时间。

```
https://signin.amazonaws.cn/federation
?Action=login
&Issuer=http%3A%2F%2Fexample.com
&Destination=http%3A%2F%2Fconsole.aws.amazon.com%2F
&SigninToken=VCQgs5qZZt3Q6fn8Tr5EXAMPLEmLnwB7JjUc-SHwnUUWabcRdnWsi4DBn-dvC
CZ85wrDOnmldUcZEXAMPLE-vXYH4Q__mleuF_W2BE5HYexbe9y4Of-kje53SsjNNecATfjIzpW1
WibbnH6YcYRiBoffZBGExbEXAMPLE5aiKX4THWjQKC6gg6alHu6JFrnOJoK3dtP6I9a6hi6yPgm
iOkPZMmNGmhsVxetKzr8mx3pxhHbMEXAMPLETv1pij0rok3IyCR2YVcIjqwfWv32HU2Xl1j471u
3fU6uOfUComeKiqTGX974xzJOZbdmX_t_lLrhEXAMPLEDDIisSnyHGw2xaZZqudm4mo2uTDk9Pv
9l5K0ZCqIgEXAMPLEEcA6tgLPyKEWGUyH6BdSC6166n4M4JkXIQgac7_7821YqixsNxZ6rsrpzwf
nQoS1407R0eJCCJ684EXAMPLEZRdBNnuLbUYpz2Iw3vIN0tQgOujwnwydPscM9F7foaEK3jwMkg
Apeb1-6L_OB12MzhuFxx5555EXAMPLEhyETEd4ZulKPdXHkg16T9ZkIlHz2Uy1RUTUhhUxNtSQ
nWc5xkbBoEcXqpoSIeK7yhje9Vzhd61AEXAMPLElbWeouACEMG6-Vd3dAgFYd6i5FYoyFrZLWvm
OLSG7RyYKeYN5VIzUk3YWQpyjP0RiT5KursUi-NEXAMPLExMOMdoODBEGKQsk-iu2ozh6r8bxwC
RNhujg
```

使用 Python 的示例代码

以下示例说明了如何使用 Python 以编程方式构造授予联合用户 Amazon Web Services Management Console 直接访问权限的 URL。以下是两个示例：

- 通过 GET 请求联合到 Amazon
- 通过 POST 请求联合到 Amazon

这两个示例都使用 [Amazon SDK for Python \(Boto3\)](#) 和 [AssumeRole](#) API 获取临时安全凭证。

使用 GET 请求

使用 POST 请求

使用 Java 的示例代码

以下示例说明了如何使用 Java 以编程方式构造授予联合身份用户 Amazon Web Services Management Console 直接访问权限的 URL。下列代码段使用 [Amazon SDK for Java](#)。

```
import java.net.URLEncoder;
import java.net.URL;
import java.net.URLConnection;
import java.io.BufferedReader;
import java.io.InputStreamReader;
// Available at http://www.json.org/java/index.html
import org.json.JSONObject;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;

/* Calls to Amazon STS API operations must be signed using the access key ID
and secret access key of an IAM user or using existing temporary
credentials. The credentials should not be embedded in code. For
this example, the code looks for the credentials in a
```

```

    standard configuration file.
*/
AWSCredentials credentials =
    new PropertiesCredentials(
        AwsConsoleApp.class.getResourceAsStream("AwsCredentials.properties"));

AWSSecurityTokenServiceClient stsClient =
    new AWSSecurityTokenServiceClient(credentials);

GetFederationTokenRequest getFederationTokenRequest =
    new GetFederationTokenRequest();
getFederationTokenRequest.setDurationSeconds(1800);
getFederationTokenRequest.setName("UserName");

// A sample policy for accessing Amazon Simple Notification Service (Amazon SNS) in the
// console.

String policy = "{\"Version\":\"2012-10-17\",\"Statement\": [{\"Action\": \"sns:*\", \" +
    \"Effect\": \"Allow\", \"Resource\": \"*\"}]}\"";

getFederationTokenRequest.setPolicy(policy);

GetFederationTokenResult federationTokenResult =
    stsClient.getFederationToken(getFederationTokenRequest);

Credentials federatedCredentials = federationTokenResult.getCredentials();

// The issuer parameter specifies your internal sign-in
// page, for example http://mysignin.internal.mycompany.com/.
// The console parameter specifies the URL to the destination console of the
// Amazon Management Console. This example goes to Amazon SNS.
// The signin parameter is the URL to send the request to.

String issuerURL = "http://mysignin.internal.mycompany.com/";
String consoleURL = "http://console.aws.amazon.com/sns";
String signInURL = "https://signin.amazonaws.cn/federation";

// Create the sign-in token using temporary credentials,
// including the access key ID, secret access key, and session token.
String sessionJson = String.format(
    "{\"%1$s\": \"%2$s\", \"%3$s\": \"%4$s\", \"%5$s\": \"%6$s\"}\",
    "sessionId", federatedCredentials.getAccessKeyId(),
    "sessionKey", federatedCredentials.getSecretAccessKey(),
    "sessionToken", federatedCredentials.getSessionToken());

// Construct the sign-in request with the request sign-in token action, a
// 12-hour console session duration, and the JSON document with temporary
// credentials as parameters.

String getSignInTokenURL = signInURL +
    "?Action=getSignInToken" +
    "&DurationSeconds=43200" +
    "&SessionType=json&Session=" +
    URLEncoder.encode(sessionJson, "UTF-8");

URL url = new URL(getSignInTokenURL);

// Send the request to the Amazon federation endpoint to get the sign-in token
URLConnection conn = url.openConnection();

BufferedReader bufferReader = new BufferedReader(new
    InputStreamReader(conn.getInputStream()));
String returnContent = bufferReader.readLine();

String signinToken = new JSONObject(returnContent).getString("SignInToken");

```

```
String signinTokenParameter = "&SigninToken=" + URLEncoder.encode(signinToken, "UTF-8");

// The issuer parameter is optional, but recommended. Use it to direct users
// to your sign-in page when their session expires.

String issuerParameter = "&Issuer=" + URLEncoder.encode(issuerURL, "UTF-8");

// Finally, present the completed URL for the Amazon console session to the user

String destinationParameter = "&Destination=" + URLEncoder.encode(consoleURL, "UTF-8");
String loginURL = signInURL + "?Action=login" +
    signinTokenParameter + issuerParameter + destinationParameter;
```

演示如何构造 URL 的示例 (Ruby)

以下示例说明了如何使用 Ruby 以编程方式构造授予联合身份用户 Amazon Web Services Management Console 直接访问权限的 URL。下列代码段使用 [Amazon SDK for Ruby](#)。

```
require 'rubygems'
require 'json'
require 'open-uri'
require 'cgi'
require 'aws-sdk'

# Create a new STS instance
#
# Note: Calls to Amazon STS API operations must be signed using an access key ID
# and secret access key. The credentials can be in EC2 instance metadata
# or in environment variables and will be automatically discovered by
# the default credentials provider in the Amazon Ruby SDK.
sts = Aws::STS::Client.new()

# The following call creates a temporary session that returns
# temporary security credentials and a session token.
# The policy grants permissions to work
# in the Amazon SNS console.

session = sts.get_federation_token({
  duration_seconds: 1800,
  name: "UserName",
  policy: "{\"Version\":\"2012-10-17\",\"Statement\": [{\"Effect\":\"Allow\", \"Action\": \"sns:*\", \"Resource\": \"*\"}]}",
})

# The issuer value is the URL where users are directed (such as
# to your internal sign-in page) when their session expires.
#
# The console value specifies the URL to the destination console.
# This example goes to the Amazon SNS console.
#
# The sign-in value is the URL of the Amazon STS federation endpoint.
issuer_url = "http://mysignin.internal.mycompany.com/"
console_url = "http://console.aws.amazon.com/sns"
signin_url = "https://signin.amazonaws.cn/federation"

# Create a block of JSON that contains the temporary credentials
# (including the access key ID, secret access key, and session token).
session_json = {
  :sessionId => session.credentials[:access_key_id],
  :sessionKey => session.credentials[:secret_access_key],
  :sessionToken => session.credentials[:session_token]
}.to_json

# Call the federation endpoint, passing the parameters
```

```
# created earlier and the session information as a JSON block.
# The request returns a sign-in token that's valid for 15 minutes.
# Signing in to the console with the token creates a session
# that is valid for 12 hours.
get_signin_token_url = signin_url +
    "?Action=getSignInToken" +
    "&SessionType=json&Session=" +
    CGI.escape(session_json)

returned_content = URI.parse(get_signin_token_url).read

# Extract the sign-in token from the information returned
# by the federation endpoint.
signin_token = JSON.parse(returned_content)['SignInToken']
signin_token_param = "&SignInToken=" + CGI.escape(signin_token)

# Create the URL to give to the user, which includes the
# sign-in token and the URL of the console to open.
# The "issuer" parameter is optional but recommended.
issuer_param = "&Issuer=" + CGI.escape(issuer_url)
destination_param = "&Destination=" + CGI.escape(console_url)
login_url = signin_url + "?Action=login" + signin_token_param +
    issuer_param + destination_param
```

使用服务相关角色

服务相关角色是一种独特类型的 IAM 角色，它与 Amazon 服务直接相关。服务相关角色由服务预定义，具有服务代表您调用其他 Amazon 服务所需的所有权限。相关的服务还定义了创建、修改和删除服务相关角色的方式。服务可以自动创建或删除角色。它可能允许您在服务的向导或流程中创建、修改或删除角色。或者，它可能需要您使用 IAM 创建或删除角色。无论使用哪种方法，服务相关角色都可让您简化服务的设置流程，因为您不必手动添加服务代表您完成操作所需的权限。

链接服务会定义其服务相关角色的权限，除非另外定义，否则仅该服务可以担任角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其它 IAM 实体的权限策略。

您必须先删除角色的相关资源，之后才能删除角色。这将保护您的资源，因为您不会无意中删除对资源的访问权限。

Tip

有关哪些服务支持使用服务相关角色的信息，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)并查找其在服务相关角色列中为是的服务。请选择 Yes 与查看该服务的[服务相关角色文档](#)的链接。

服务相关角色权限

您必须为 IAM 实体（用户、组或角色）配置权限，以允许用户或角色创建或编辑服务相关角色。

Note

服务相关角色的 ARN 包括服务主体，它在以下策略中显示为 **SERVICE-NAME**.amazonaws.com。请勿尝试猜测服务主体，因为它区分大小写，并且格式会因 Amazon 服务而异。要查看服务的主体，请参阅其[服务相关角色文档](#)。

允许 IAM 实体创建特定服务相关角色

将以下策略添加到需要创建服务相关角色的 IAM 实体中。

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

```
{
  "Effect": "Allow",
  "Action": "iam:CreateServiceLinkedRole",
  "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX",
  "Condition": {"StringLike": {"iam:AWSServiceName": "SERVICE-NAME.amazonaws.com"}}
},
{
  "Effect": "Allow",
  "Action": [
    "iam:AttachRolePolicy",
    "iam:PutRolePolicy"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX"
}
]
```

允许 IAM 实体创建任何服务相关角色

将以下语句添加到 IAM 实体的权限策略，该实体需要创建服务相关角色或任何包含所需策略的服务角色。此策略语句不允许 IAM 实体将策略附加到该角色。

```
{
  "Effect": "Allow",
  "Action": "iam:CreateServiceLinkedRole",
  "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

允许 IAM 实体编辑任何服务角色的描述

将以下语句添加到 IAM 实体的权限策略，该实体需要编辑服务相关角色或任何服务角色的描述。

```
{
  "Effect": "Allow",
  "Action": "iam:UpdateRoleDescription",
  "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

允许 IAM 实体删除特定服务相关角色

将以下语句添加到需要删除服务相关角色的 IAM 实体的权限策略。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:DeleteServiceLinkedRole",
    "iam:GetServiceLinkedRoleDeletionStatus"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX"
}
```

允许 IAM 实体删除任何服务相关角色

将以下语句添加到 IAM 实体的权限策略，该实体需要删除服务相关角色，而不需要删除服务角色。

```
{
```

```
{
  "Effect": "Allow",
  "Action": [
    "iam:DeleteServiceLinkedRole",
    "iam:GetServiceLinkedRoleDeletionStatus"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

允许 IAM 实体将现有角色传递给服务

有些 Amazon 服务允许您将现有角色传递给服务，而不是创建新的服务相关角色。为此，用户必须具有将角色传递给服务的权限。将以下语句添加到需要传递角色的 IAM 实体的权限策略。此策略语句还允许实体查看可从中选择要传递的角色的角色列表。有关更多信息，请参阅[向用户授予权限以将角色传递给 Amazon 服务](#) (p. 240)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListRoles",
    "iam:PassRole"
  ],
  "Resource": "arn:aws:iam::123456789012:role/my-role-for-XYZ"
}
```

转让服务相关角色权限

服务相关角色授予的权限可以间接转让给其他用户和角色。在允许一个服务在其他服务中执行操作时，发端服务将来可以使用这些权限。如果另一个用户或角色有权在发端服务中执行操作，则该发端服务可以随后担任此角色并在其他服务中访问资源。这意味着，另一个用户或角色可以间接访问其他服务。

例如，在创建 Amazon RDS 数据库实例时，[RDS 为您创建服务相关角色](#)。每次编辑数据库实例时，该角色允许 RDS 代表您调用 Amazon EC2、Amazon SNS、Amazon CloudWatch Logs 和 Amazon Kinesis。如果创建一个策略以允许您的账户或其他具有相同权限账户中的用户和角色访问该 Amazon RDS 实例，则 RDS 仍然可以使用该角色代表他们对 EC2、SNS、CloudWatch Logs 和 Kinesis 进行更改。新用户或角色可以间接编辑这些其他服务中的资源。

Important

在转移服务相关角色权限时，请注意“混淆代理人”权限提升风险。请在[防止跨服务混淆代理人](#)中了解有关迁移的更多信息。

创建服务相关角色

您用来创建服务相关角色的方法取决于服务。在某些情况下，无需手动创建服务相关角色。例如，在服务中完成特定操作 (如创建资源) 时，服务可能为您创建服务相关角色。或者，如果您在某项服务开始支持服务相关角色之前已在使用该服务，则该服务可能自动在您的账户中创建角色。要了解更多信息，请参阅[我的 Amazon 账户中出现新角色](#) (p. 885)。

在其他情况下，服务可能支持使用服务控制台、API 或 CLI 手动创建服务相关角色。有关哪些服务支持使用服务相关角色的信息，请参阅[使用 IAM 的 Amazon 服务](#) (p. 908) 并查找其在服务相关角色列表中为是的服务。要了解服务是否支持创建服务相关角色，请选择 Yes 链接以查看该服务的服务相关角色文档。

如果服务不支持创建角色，则可以使用 IAM 创建服务相关角色。

Important

服务相关角色将计入您的 [Amazon 账户中的 IAM 角色限制](#)，但如果您已达到限制，仍可以在账户中创建服务相关角色。只有服务相关角色可以超过此限制。

创建服务相关角色 (控制台)

在 IAM 中创建服务相关角色之前，请查明链接服务是否已自动创建服务相关角色。此外，需了解您是否能从该服务的控制台、API 或 CLI 创建角色。

创建服务相关角色 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在 IAM 控制台的导航窗格中，选择角色。然后，选择 Create role (创建角色)。
3. 选择 Amazon Service (Amazon 服务) 角色类型。
4. 选择用于您的服务的使用案例。使用案例由服务定义以包含服务所需的信任策略。然后，选择 Next (下一步)。
5. 选择一个或多个要附加到角色的权限策略。根据您的选择的使用案例，服务可能执行以下任意操作：
 - 定义角色所使用的权限。
 - 允许您从一组有限的权限中进行选择。
 - 允许您从任意权限中进行选择。
 - 允许您此时不选择策略，稍后创建策略，然后将这些策略附加到角色。

选中分配您希望角色具有的权限的策略旁边的复选框，然后选择 Next (下一步)。

Note

使用角色的任何实体都可以使用您指定的权限。默认情况下，角色没有权限。

6. 对于 Role name (角色名称)，角色名称自定义的程度由服务定义。如果服务定义角色的名称，则此选项不可编辑。在其他情况下，服务可能定义角色的前缀并让您输入可选的后缀。

如果可能，请输入要添加到默认名称的角色名称后缀。该后缀可帮助您确定角色的用途。角色名称在您的 Amazon 账户内必须是唯一的。名称不区分大小写。例如，您无法同时创建名为 **<service-linked-role-name>_SAMPLE** 和 **<service-linked-role-name>_sample** 的角色。由于多个单位可能引用该角色，角色创建完毕后无法编辑角色名称。

7. (可选) 对于 Description (描述)，编辑新服务相关角色的描述。
8. 您无法在创建过程中将标签附加到服务相关角色。有关在 IAM 中使用标签的更多信息，请参阅 [标记 IAM 资源 \(p. 276\)](#)。
9. 检查角色，然后选择 Create role。

创建服务相关角色(Amazon CLI)

在 IAM 中创建服务相关角色之前，请查明链接服务是否已自动创建服务相关角色，以及您是否能从该服务的 CLI 创建角色。如果服务 CLI 不受支持，您可以使用 IAM 命令创建具有服务担任角色时所需的信任策略和内联策略的服务相关角色。

创建服务相关角色 (Amazon CLI)

运行以下 命令：

```
aws iam create-service-linked-role --aws-service-name SERVICE-NAME.amazonaws.com
```

创建服务相关角色 (Amazon API)

在 IAM 中创建服务相关角色之前，请查明链接服务是否已自动创建服务相关角色，以及您是否能从该服务的 API 创建角色。如果服务 API 不受支持，您可以使用 Amazon API 创建具有服务代入角色时所需的信任策略和内联策略的服务相关角色。

创建服务相关角色 (Amazon API)

使用 `CreateServiceLinkedRole` API 调用。在请求中，指定 `SERVICE_NAME_URL.amazonaws.com` 的服务名称。

例如，要创建 Lex Bots 服务相关角色，请使用 `lex.amazonaws.com`。

编辑服务相关角色

您用来编辑服务相关角色的方法取决于服务。某些服务可能允许您从服务控制台、API 或 CLI 编辑服务相关角色的权限。但是，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。您可以从 IAM 控制台、API 或 CLI 编辑任何角色的描述。

有关哪些服务支持使用服务相关角色的信息，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)并查找其在服务相关角色列表中为是的服务。要了解服务是否支持编辑服务相关角色，请选择 Yes 链接以查看该服务的服务相关角色文档。

编辑服务相关角色描述 (控制台)

您可以使用 IAM 控制台编辑服务相关角色的描述。

编辑服务相关角色的描述 (控制台)

1. 在 IAM 控制台的导航窗格中，选择角色。
2. 以下代码示例显示如何将 IAM policy 附加到用户。
3. 在 Role description 的最右侧，选择 Edit。
4. 在框中输入新描述，然后选择 Save (保存)。

编辑服务相关角色描述 (Amazon CLI)

您可以从 Amazon CLI 使用 IAM 命令编辑服务相关角色的描述。

更改服务相关角色描述 (Amazon CLI)

1. (可选) 要查看角色的当前描述，请运行以下命令：

```
aws iam get-role --role-name ROLE-NAME
```

通过 CLI 命令使用角色名称 (并非 ARN) 指向角色。例如，如果某个角色的 ARN 为 `arn:aws:iam::123456789012:role/myrole`，则将该角色称为 **myrole**。

2. 要更新服务相关角色的描述，请运行以下命令：

```
aws iam update-role --role-name ROLE-NAME --description OPTIONAL-DESCRIPTION
```

编辑服务相关角色描述 (Amazon API)

您可以使用 Amazon API 编辑服务相关角色的描述。

更改服务相关角色的描述 (Amazon API)

1. (可选) 要查看角色的当前描述，请调用以下操作，并指定角色的名称：

Amazon API：[GetRole](#)

2. 要更新角色的描述，请调用以下操作，并指定角色的名称 (和可选描述)：

Amazon API：[UpdateRole](#)

删除服务相关角色

您用来创建服务相关角色的方法取决于服务。在某些情况下，无需手动删除服务相关角色。例如，在服务中完成特定操作（如删除资源）时，服务可能为您删除服务相关角色。

在其他情况下，服务可能支持从服务控制台、API 或 Amazon CLI 中手动删除服务相关角色。

有关哪些服务支持使用服务相关角色的信息，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)并查找其在服务相关角色列表中为是的服务。要了解服务是否支持删除服务相关角色，请选择 Yes 链接以查看该服务的服务相关角色文档。

如果服务不支持删除角色，则可以从 IAM 控制台、API 或 Amazon CLI 中删除服务相关角色。如果您不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。但是，您必须先清除您的服务相关角色，然后才能将其删除。

清除服务相关角色

必须先确认服务相关角色没有活动会话并删除该角色使用的任何资源，然后才能使用 IAM 删除服务相关角色。

在 IAM 控制台中检查服务相关角色是否具有活动会话

1. 登录 Amazon Web Services Management Console，打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在 IAM 控制台的导航窗格中，选择角色。然后选择服务相关角色的名称（而不是复选框）。
3. 在所选角色的摘要页面上，选择访问顾问选项卡。
4. 在访问顾问选项卡查看服务相关角色的近期活动。

Note

如果您不确定服务是否使用服务相关角色，您可以尝试删除该角色。如果服务正在使用该角色，则删除操作会失败，并且您可以查看正在使用该角色的区域。如果该角色已被使用，则您必须等待会话结束，然后才能删除该角色。您无法撤销服务相关角色对会话的权限。

删除服务相关角色所使用的资源

有关哪些服务支持使用服务相关角色的信息，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)并查找其在服务相关角色列表中为是的服务。要了解服务是否支持删除服务相关角色，请选择 Yes 链接以查看该服务的服务相关角色文档。请参阅该服务的文档，以了解如何删除服务相关角色所使用的资源。

删除服务相关角色（控制台）

您可以使用 IAM 控制台删除服务相关角色。

删除服务相关角色（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在 IAM 控制台的导航窗格中，选择角色。然后，选中要删除的角色名称旁边的复选框，而不是名称或行本身。
3. 对于页面顶部的 Role actions（角色操作），请选择 Delete（删除）。
4. 在确认对话框中，查看上次访问的信息，该信息显示每个选定角色上次访问 Amazon 服务的时间。这样可帮助您确认角色当前是否处于活动状态。如果要继续，请选择 Yes, Delete 以提交服务相关角色进行删除。
5. 监视 IAM 控制台通知，以监控服务相关角色的删除进度。由于 IAM 服务相关角色删除是异步的，因此，在您提交角色进行删除后，删除任务可能成功，也可能失败。
 - 如果任务成功，则角色将从列表中删除，并会在页面顶部显示成功通知。

- 如果任务失败，您可以从通知中选择 View details 或 View Resources 以了解删除失败的原因。如果因为角色正在使用服务的资源而使删除失败，则通知包含一个资源列表 (如果服务返回该信息)。然后您可以[清除资源 \(p. 213\)](#)并再次提交删除。

Note

您可能需要多次重复执行此过程，这取决于服务返回的信息。例如，您的服务相关角色可能使用六个资源，而您的服务可能返回有关其中五个资源的信息。如果您清除这五个资源并再次提交该角色以进行删除，则删除会失败，并且服务会报告一个剩余资源。服务可能会返回所有资源、其中一些资源，也可能不报告任何资源。

- 如果任务失败，并且通知不包含资源列表，则服务可能不会返回该信息。要了解如何清除该服务的资源，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)。在表中查找您的服务，然后选择 Yes 链接以查看该服务的服务相关角色文档。

删除服务相关角色 (Amazon CLI)

您可以从 Amazon CLI 使用 IAM 命令删除服务相关角色。

删除服务相关角色 (Amazon CLI)

- 如果知道要删除的服务相关角色的名称，请输入以下命令列出账户中的角色：

```
aws iam get-role --role-name role-name
```

通过 CLI 命令使用角色名称 (并非 ARN) 指向角色。例如，如果某个角色的 ARN 为 `arn:aws:iam::123456789012:role/myrole`，则将该角色称为 **myrole**。

- 如果服务相关角色正被使用或具有关联的资源，则无法删除它，因此您必须提交删除请求。如果不满足这些条件，该请求可能会被拒绝。您必须从响应中捕获 `deletion-task-id` 以检查删除任务的状态。输入以下命令以提交服务相关角色的删除请求：

```
aws iam delete-service-linked-role --role-name role-name
```

- 输入以下命令以检查删除任务的状态：

```
aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

删除任务的状态可能是 NOT_STARTED、IN_PROGRESS、SUCCEEDED 或 FAILED。如果删除失败，则调用会返回失败的原因，以便您进行问题排查。如果因为角色正在使用服务的资源而使删除失败，则通知包含一个资源列表 (如果服务返回该信息)。然后您可以[清除资源 \(p. 213\)](#)并再次提交删除。

Note

您可能需要多次重复执行此过程，这取决于服务返回的信息。例如，您的服务相关角色可能使用六个资源，而您的服务可能返回有关其中五个资源的信息。如果您清除这五个资源并再次提交该角色以进行删除，则删除会失败，并且服务会报告一个剩余资源。服务可能会返回所有资源、其中一些资源，也可能不报告任何资源。要了解如何为不报告任何资源的服务清除资源，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)。在表中查找您的服务，然后选择 Yes 链接以查看该服务的服务相关角色文档。

删除服务相关角色 (Amazon API)

您可以使用 Amazon API 删除服务相关角色。

删除服务相关角色 (Amazon API)

- 要提交服务相关角色的删除请求，请调用 `DeleteServiceLinkedRole`。在请求中，指定角色名称。

如果服务相关角色正被使用或具有关联的资源，则无法删除它，因此您必须提交删除请求。如果不满足这些条件，该请求可能会被拒绝。您必须从响应中捕获 `DeletionTaskId` 以检查删除任务的状态。

2. 要检查删除的状态，请调用 [GetServiceLinkedRoleDeletionStatus](#)。在请求中，指定 `DeletionTaskId`。

删除任务的状态可能是 `NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED` 或 `FAILED`。如果删除失败，则调用会返回失败的原因，以便您进行问题排查。如果因为角色正在使用服务的资源而使删除失败，则通知包含一个资源列表 (如果服务返回该信息)。然后您可以[清除资源 \(p. 213\)](#)并再次提交删除。

Note

您可能需要多次重复执行此过程，这取决于服务返回的信息。例如，您的服务相关角色可能使用六个资源，而您的服务可能返回有关其中五个资源的信息。如果您清除这五个资源并再次提交该角色以进行删除，则删除会失败，并且服务会报告一个剩余资源。服务可能会返回所有资源、其中一些资源，也可能不报告任何资源。要了解如何为不报告任何资源的服务清除资源，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)。在表中查找您的服务，然后选择 Yes 链接以查看该服务的服务相关角色文档。

创建 IAM 角色

要创建角色，可以使用 Amazon Web Services Management Console、Amazon CLI、Tools for Windows PowerShell 或 IAM API。

如果使用的是 Amazon Web Services Management Console，则可使用向导来完成创建角色的步骤。向导的步骤可能稍有不同，具体取决于您是 Amazon 服务、Amazon 账户还是联合身份用户创建角色。

主题

- [创建向 IAM 用户委派权限的角色 \(p. 215\)](#)
- [创建向 Amazon 服务委派权限的角色 \(p. 219\)](#)
- [针对第三方身份提供程序创建角色 \(联合\) \(p. 224\)](#)
- [使用自定义信任策略创建角色 \(控制台\) \(p. 232\)](#)
- [委派访问权限的策略示例 \(p. 233\)](#)

创建向 IAM 用户委派权限的角色

您可以使用 IAM 角色委派对 Amazon 资源的访问权限。利用 IAM 角色，您可以在您的信任账户和其他 Amazon 受信任账户之间建立信任关系。信任账户拥有要访问的资源，受信任账户包含需要资源访问权限的用户。但是，其他账户可以拥有您的账户中的资源。例如，信任账户可能允许受信任账户创建新资源，例如在 Amazon S3 存储桶中创建新对象。在这种情况下，创建资源的账户拥有资源并控制谁可以访问该资源。

创建信任关系后，来自受信任账户的 IAM 用户或应用程序可以使用 Amazon Security Token Service (Amazon STS) [AssumeRole](#) API 操作。此操作提供临时安全凭证，可用于访问您账户中的 Amazon 资源。

账户可由您进行控制，或者具有用户的账户可由第三方进行控制。如果另一个具有用户的账户是您无法控制的 Amazon 账户，则可使用 `externalId` 属性。外部 ID 可以是您和第三方账户管理员之间达成一致的任意字词或数字。此选项会在信任策略中自动添加一个条件，即仅当请求包括正确的 `sts:ExternalID` 时，该用户才担任该角色。有关更多信息，请参阅[如何在向第三方授予对 Amazon 资源的访问权时使用外部 ID \(p. 164\)](#)。

有关如何使用角色委派权限的信息，请参阅[角色术语和概念 \(p. 158\)](#)。有关使用服务角色以允许服务访问账户中的资源的信息，请参阅[创建向 Amazon 服务委派权限的角色 \(p. 219\)](#)。

创建 IAM 角色 (控制台)

您可以使用 Amazon Web Services Management Console 创建 IAM 用户可担任的角色。例如，假设贵组织拥有多个 Amazon 账户以便将开发环境与生产环境隔离。有关创建角色 (该角色允许开发账户中的用户访问生产账户中的资源) 的概述信息，请参阅 [使用不同的开发和生产账户的示例方案 \(p. 162\)](#)。

创建角色 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在控制台的导航窗格中，选择 Roles，然后选择 Create role。
3. 选择 Amazon account (Amazon 账户) 角色类型。
4. 要为您的账户创建角色，请选择 This account (此账户)。要为另一个账户创建角色，请选择 Another Amazon account (另一个 Amazon 账户) 然后输入您希望向其授予对您的资源的访问权限的账户的 Account ID (账户 ID)。

指定账户的管理员可向该账户中的任何 IAM 用户授予担任该角色的权限。为此，管理员需要将策略附加到用户或组来授予 `sts:AssumeRole` 操作的权限。该策略必须指定角色的 ARN 作为 Resource。

5. 如果您向不受您控制的账户的用户授予权限，并且用户将以编程方式代入此角色，请选择 Require external ID (需要外部 ID)。外部 ID 可以是您和第三方账户管理员之间达成一致的任何字词或数字。此选项会在信任策略中自动添加一个条件，即仅当请求包括正确的 `sts:ExternalID` 时，该用户才担任该角色。有关更多信息，请参阅[如何在向第三方授予对 Amazon 资源的访问权时使用外部 ID \(p. 164\)](#)。

Important

选择此选项将仅允许通过 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 访问该角色。这是因为，您不能使用 Amazon 控制台切换到其信任策略中有 `externalId` 条件的角色。但您可以通过使用相关开发工具包编写脚本或应用程序，以编程方式创建此类访问。

有关更多信息和示例脚本，请参阅 Amazon 安全博客中的[如何启用对 Amazon Web Services Management Console 的跨账户存取](#)。

6. 如果需要将该角色限制为使用多重身份验证 (MFA) 进行登录的用户，请选择 Require MFA。这将向角色的信任策略中添加用于检查 MFA 登录的条件。需要担任角色的用户必须使用配置的 MFA 设备中的临时一次性密码进行登录。没有经过 MFA 身份验证的用户无法担任该角色。有关 MFA 的更多信息，请参阅[在 Amazon 中使用多重身份验证 \(MFA\) \(p. 104\)](#)。
7. 选择 Next (下一步)。
8. IAM 包括您的账户中的 Amazon 托管策略和客户托管策略的列表。选择要用于权限策略的策略，或选择创建策略以打开新的浏览器选项卡并从头开始创建新策略。有关更多信息，请参阅[创建 IAM 策略 \(p. 438\)](#)。在您创建策略后，关闭该选项卡并返回到您的原始选项卡。选中您希望担任角色的任何人具有的权限策略旁边的复选框。如果您愿意，此时可以不选择任何策略，稍后将策略附加到角色。默认情况下，角色没有权限。
9. (可选) 设置[权限边界 \(p. 370\)](#)。这是一项高级功能。

打开设置权限边界部分，然后选择使用权限边界控制最大角色权限。选择要用于权限边界的策略。

10. 选择 Next (下一步)。
11. 对于 Role name (角色名称)，为您的角色输入一个名称。角色名称在您的 Amazon 账户内必须是唯一的。名称不区分大小写。例如，您无法同时创建名为 **PRODROLE** 和 **prodrole** 的角色。由于其他 Amazon 资源可能引用该角色，角色创建完毕后无法编辑角色名称。
12. (可选) 对于 Description (描述)，输入新角色的描述。
13. 在 Step 1: Select trusted entities (步骤 1：选择可信实体) 或 Step 2: Add permissions (步骤 2：添加权限) 部分中的 Edit (编辑)，以编辑角色的用户案例和权限。
14. (可选) 通过以键值对的形式附加标签来向角色添加元数据。有关在 IAM 中使用标签的更多信息，请参阅[标记 IAM 资源 \(p. 276\)](#)。
15. 检查角色，然后选择 Create role。

Important

请注意，这只是所需配置的前半部分。您还必须向受信任账户中的各个用户授予在控制台中切换到此角色或以编程方式担任此角色的权限。有关这一步骤的更多信息，请参阅[向用户授予切换角色的权限 \(p. 239\)](#)。

创建 IAM 角色 (Amazon CLI)

从 Amazon CLI 创建角色涉及几个步骤。当您使用控制台创建角色时，很多步骤会自动完成，但是使用 Amazon CLI 时，您必须自行完成每一个步骤。您必须创建角色，然后为角色分配权限策略。（可选）您还可以为您的角色设置[权限边界 \(p. 370\)](#)。

创建用于跨账户存取的角色 (Amazon CLI)

1. 创建角色：[aws iam create-role](#)
2. 将托管权限策略附加到角色：[aws iam attach-role-policy](#)

或

为角色创建内联权限策略：[aws iam put-role-policy](#)

3. （可选）通过附加标签来向角色添加自定义属性：[aws iam tag-role](#)

有关更多信息，请参阅[管理 IAM 角色 \(Amazon CLI 或 Amazon API \) 的标签 \(p. 282\)](#)。

4. （可选）为角色设置[权限边界 \(p. 370\)](#)：[aws iam put-role-permissions-boundary](#)

权限边界控制角色可以具有的最大权限。权限边界是一项高级 Amazon 功能。

以下示例演示了在简单环境中创建跨账户角色的前两个最常见的步骤。此示例允许 123456789012 账户中的任何用户担任角色并查看 example_bucket Amazon S3 存储桶。该示例还假设您使用运行 Windows 的客户端计算机，并且已使用您的账户凭证和区域配置了命令行界面。有关更多信息，请参阅[配置 Amazon 命令行界面](#)。

在此示例中，在您创建角色时，在第一个命令中包括以下信任策略。此信任策略允许 123456789012 账户中的用户使用 AssumeRole 操作担任角色，但前提是用户使用 SerialNumber 和 TokenCode 参数提供 MFA 身份验证。有关 MFA 的更多信息，请参阅[在 Amazon 中使用多重身份验证 \(MFA\) \(p. 104\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
      "Action": "sts:AssumeRole",
      "Condition": { "Bool": { "aws:MultiFactorAuthPresent": "true" } }
    }
  ]
}
```

Important

如果 Principal 元素中包含特定 IAM 角色或用户的 ARN，在保存策略时该 ARN 将转换为唯一的主体 ID。如果有人希望通过删除并重新创建角色或用户来提升权限，这样有助于减轻此类风险。您通常不会在控制台中看到这个 ID，因为显示信任策略时它还会反向转换为 ARN。但是，如果您删除角色或用户，则主体 ID 会显示在控制台中，因为 Amazon 无法再将其映射回 ARN。因此，如果您删除并重新创建了信任策略的 Principal 元素所引用的用户或角色，您必须编辑角色以替换 ARN。

当您使用第二个命令时，您必须将现有托管策略附加到角色。下面的权限策略仅允许担任角色的任何人对 `example_bucket` Amazon S3 存储桶执行 `ListBucket` 操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::example_bucket"
    }
  ]
}
```

要创建此 `Test-UserAccess-Role` 角色，您必须先将名为 `trustpolicyforacct123456789012.json` 的上述信任策略保存到本地 `policies` 驱动器中的 `C:` 文件夹。然后，使用名称 `PolicyForRole` 将上述权限策略保存为您的 Amazon 账户中的客户托管策略。然后您可以使用以下命令来创建角色并附加托管策略。

```
# Create the role and attach the trust policy file that allows users in the specified
account to assume the role.
$ aws iam create-role --role-name Test-UserAccess-Role --assume-role-policy-document
file://C:\policies\trustpolicyforacct123456789012.json

# Attach the permissions policy (in this example a managed policy) to the role to specify
what it is allowed to do.
$ aws iam attach-role-policy --role-name Test-UserAccess-Role --policy-arn
arn:aws:iam::123456789012:role/PolicyForRole
```

Important

请注意，这只是所需配置的前半部分。您还必须对受信任账户中的各个用户授予切换角色的权限。有关这一步骤的更多信息，请参阅[向用户授予切换角色的权限](#) (p. 239)。

在创建角色并向该角色授予执行 Amazon 任务或访问 Amazon 资源的权限后，`123456789012` 账户中的任何用户都可以担任该角色。有关更多信息，请参阅[切换到 IAM 角色 \(Amazon CLI\)](#) (p. 245)。

创建 IAM 角色 (Amazon API)

从 Amazon API 创建角色涉及几个步骤。当您使用控制台创建角色时，很多步骤会自动完成，但是使用 API 时，您必须自行完成每一个步骤。您必须创建角色，然后为角色分配权限策略。（可选）您还可以为您的角色设置[权限边界](#) (p. 370)。

在代码中创建角色 (Amazon API)

1. 创建角色：[CreateRole](#)

您可以指定一个文件位置作为角色的信任策略。

2. 将托管权限策略附加到角色：[AttachRolePolicy](#)

或

为角色创建内联权限策略：[PutRolePolicy](#)

Important

请注意，这只是所需配置的前半部分。您还必须对受信任账户中的各个用户授予切换角色的权限。有关这一步骤的更多信息，请参阅[向用户授予切换角色的权限](#) (p. 239)。

3. （可选）通过附加标签来向用户添加自定义属性：[TagRole](#)

有关更多信息，请参阅[管理 IAM 用户 \(Amazon CLI 或 Amazon API\) 的标签](#) (p. 280)。

4. (可选) 为角色设置权限边界 (p. 370) : [PutRolePermissionsBoundary](#)

权限边界控制角色可以具有的最大权限。权限边界是一项高级 Amazon 功能。

在创建角色并向该角色授予执行 Amazon 任务或访问 Amazon 资源的权限后，您必须向账户中的用户授予允许他们担任该角色的权限。有关担任角色的更多信息，请参阅 [切换到 IAM 角色 \(Amazon API\)](#) (p. 250)。

创建 IAM 角色 (Amazon CloudFormation)

有关在 Amazon CloudFormation 中创建 IAM 角色的信息，请参阅 Amazon CloudFormation 用户指南中的 [资源和属性参考](#) 和 [示例](#)。

有关 Amazon CloudFormation 中的 IAM 模板的更多信息，请参阅 Amazon CloudFormation 用户指南中的 [Amazon Identity and Access Management 模板代码段](#)。

创建向 Amazon 服务委派权限的角色

许多 Amazon 服务要求您使用角色来允许该服务代表您访问其他服务中的资源。由一项服务担任、代表您执行操作的角色称为 [服务角色](#) (p. 159)。如果按服务角色适用的特定目的进行分类，可分为 [EC2 实例的服务角色](#) (p. 159) (举例)，或 [服务相关角色](#) (p. 159)。要查看哪些服务支持使用服务相关角色，或服务是否支持任何形式的临时凭证，请参阅 [使用 IAM 的 Amazon 服务](#) (p. 908)。要了解单个服务如何使用角色，请选择表格中的服务名称以查看该服务对应的文档。

有关如何通过角色委派权限的信息，请参阅 [角色术语和概念](#) (p. 158)。

服务角色权限

您必须配置权限以允许 IAM 实体 (用户或角色) 创建或编辑服务角色。

Note

服务相关角色的 ARN 包括服务主体，它在以下策略中显示为 `SERVICE-NAME.amazonaws.com`。请勿尝试猜测服务主体，因为它区分大小写，并且格式会因 Amazon 服务而异。要查看服务的主体，请参阅其服务相关角色文档。

允许 IAM 实体创建特定服务角色

将以下策略添加到需要创建服务角色的 IAM 实体中。此策略允许您为指定服务创建具有特定名称的服务角色。然后，可将托管或内联策略附加到该角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:AttachRolePolicy",
        "iam:CreateRole",
        "iam:PutRolePolicy"
      ],
      "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
    }
  ]
}
```

允许 IAM 实体创建任何服务角色

Amazon 建议您只允许 IAM 管理员创建任何服务角色。拥有创建角色和附加任何策略权限的人员可以升级自己的权限。相反，应创建一个策略，允许他们仅创建他们所需的角色，或者让管理员代表他们创建服务角色。

要附加允许管理员访问您整个 Amazon Web Services 账户 的策略，请使用 [AdministratorAccess](#) Amazon 托管策略。

允许 IAM 实体编辑服务角色

将以下策略添加到需要编辑服务角色的 IAM 实体中。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EditSpecificServiceRole",
      "Effect": "Allow",
      "Action": [
        "iam:AttachRolePolicy",
        "iam:DeleteRolePolicy",
        "iam:DetachRolePolicy",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListRolePolicies",
        "iam:PutRolePolicy",
        "iam:UpdateRole",
        "iam:UpdateRoleDescription"
      ],
      "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
    },
    {
      "Sid": "ViewRolesAndPolicies",
      "Effect": "Allow",
      "Action": [
        "iam:GetPolicy",
        "iam:ListRoles"
      ],
      "Resource": "*"
    }
  ]
}
```

允许 IAM 实体删除特定服务角色

将以下语句添加到需要删除指定服务角色的 IAM 实体的权限策略。

```
{
  "Effect": "Allow",
  "Action": "iam:DeleteRole",
  "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
}
```

允许 IAM 实体删除任何服务相关角色

Amazon 建议您只允许 IAM 管理员删除任何服务角色。相反，应创建一个策略，允许他们仅删除他们所需的角色，或者让管理员代表他们删除服务角色。

要附加到允许管理员访问您整个 Amazon Web Services 账户 的策略，请使用 [AdministratorAccess](#) Amazon 托管策略。

创建用于 Amazon 服务的角色（控制台）

您可使用 Amazon Web Services Management Console 创建用于服务的角色。由于某些服务支持多个服务角色，请参阅您的服务的 [Amazon 文档](#) 以查看要选择的使用案例。您可以了解如何为角色分配必要的信任策略和权限策略，以便服务能够代表您担任角色。可用于控制您的角色的权限的步骤可能会有所不同，具体取决于服务如何定义使用案例，以及您是否创建了服务相关角色。

创建用于 Amazon 服务的角色 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在 IAM 控制台的导航窗格中，选择 Roles，然后选择 Create role。
3. 对于 Select trusted entity (选择可信实体)，选择 Amazon service (Amazon 服务)。
4. 选择用于您的服务的使用案例。使用案例由服务定义以包含服务所需的信任策略。然后选择下一步。
5. 如果可能，选择要用作权限策略的策略，或选择 Create policy (创建策略) 以打开新的浏览器选项卡并从头开始创建新策略。有关更多信息，请参阅[创建 IAM 策略 \(p. 438\)](#)。在您创建策略后，关闭该选项卡并返回到您的原始选项卡。选中您希望服务具有的权限策略旁边的复选框。

根据您选择的使用案例，该服务可能允许您执行以下任意操作：

- 不执行任何操作，因为该服务为角色定义了权限。
 - 从一组有限的权限中进行选择。
 - 从任何权限中进行选择。
 - 此时可以不选择任何策略，稍后创建策略，然后将其附加到角色。
6. (可选) 设置[权限边界 \(p. 370\)](#)。这是一项高级功能，可用于服务角色，但不可用于服务相关角色。

展开 Permissions boundary (权限边界) 部分，然后选择 Use a permissions boundary to control the maximum role permissions (使用权限边界控制最大角色权限)。IAM 包括您的账户中的 Amazon 托管策略和客户托管策略的列表。选择要用于权限边界的策略，或选择 Create policy (创建策略) 以打开新的浏览器选项卡并从头开始创建新策略。有关更多信息，请参阅[创建 IAM 策略 \(p. 438\)](#)。在您创建策略后，关闭该选项卡并返回到您的原始选项卡，以选择要用于权限边界的策略。

7. 选择 Next (下一步)。
8. 对于 Role name (角色名称)，角色名称自定义的程度由服务定义。如果服务定义角色的名称，则此选项不可编辑。在其他情况下，服务可能定义角色的前缀并允许您键入可选的后缀。某些服务允许您指定角色的整个名称。

如果可能，输入有助于识别该角色的作用的角色名称或角色名称后缀。角色名称在您的 Amazon 账户内必须是唯一的。名称不区分大小写。例如，您无法同时创建名为 **PRODROLE** 和 **prodrole** 的角色。由于其他 Amazon 资源可能引用该角色，角色创建完毕后无法编辑角色名称。

9. (可选) 对于 Description (描述)，输入新角色的描述。
10. 在 Step 1: Select trusted entities (步骤 1：选择可信实体) 或 Step 2: Add permissions (步骤 2：添加权限) 部分中的 Edit (编辑)，以编辑角色的用户案例和权限。
11. (可选) 通过以键值对的形式附加标签来向角色添加元数据。有关在 IAM 中使用标签的更多信息，请参阅[标记 IAM 资源 \(p. 276\)](#)。
12. 检查角色，然后选择 Create role。

创建用于服务的角色 (Amazon CLI)

从 Amazon CLI 创建角色涉及几个步骤。当您使用控制台创建角色时，很多步骤会自动完成，但是使用 Amazon CLI 时，您必须自行完成每一个步骤。您必须创建角色，然后为角色分配权限策略。如果您使用的服务是 Amazon EC2，则还必须创建实例配置文件并向其添加角色。(可选) 您还可以为您的角色设置[权限边界 \(p. 370\)](#)。

从 Amazon CLI 为 Amazon 服务创建角色

1. 以下 `create-role` 命令创建了一个名为 Test-Role 的角色并向其附加了信任策略：

```
aws iam create-role --role-name Test-Role --assume-role-policy-document
file://Test-Role-Trust-Policy.json
```

2. 将托管权限策略附加到角色：[aws iam attach-role-policy](#)。

例如，以下内容 `attach-role-policy` 命令将名为 `ReadOnlyAccess` 的附加 Amazon 托管策略附加到了名为 `ReadOnlyRole` 的 IAM 角色：

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess --role-name ReadOnlyRole
```

或

为角色创建内联权限策略：[aws iam put-role-policy](#)

要添加内联权限策略，请参阅以下示例：

```
aws iam put-role-policy --role-name Test-Role --policy-name ExamplePolicy --policy-document file://AdminPolicy.json
```

3. （可选）通过附加标签来向角色添加自定义属性：[aws iam tag-role](#)

有关更多信息，请参阅[管理 IAM 角色 \(Amazon CLI 或 Amazon API\)](#) 的标签 (p. 282)。

4. （可选）为角色设置权限边界 (p. 370)：[aws iam put-role-permissions-boundary](#)

权限边界控制角色可以具有的最大权限。权限边界是一项高级 Amazon 功能。

如果要将角色与 Amazon EC2 或使用 Amazon EC2 的其他 Amazon 服务结合使用，则必须将角色存储在实例配置文件中。实例配置文件是一个角色容器，可在启动时附加到 Amazon EC2 实例。一个实例配置文件只能包含一个角色，不能提高该限制。如果使用 Amazon Web Services Management Console 创建角色，则会为您创建具有与角色相同的名称的实例配置文件。有关实例配置文件的更多信息，请参阅[使用实例配置文件 \(p. 258\)](#)。有关如何通过角色启动 EC2 实例的信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的[控制对 Amazon EC2 资源的访问](#)。

创建实例配置文件并在其中存储角色 (Amazon CLI)

1. 创建实例配置文件：[aws iam create-instance-profile](#)
2. 向实例配置文件添加角色：[aws iam add-role-to-instance-profile](#)

以下 Amazon CLI 示例命令集演示了创建角色并附加权限的前两个步骤。它还演示了创建实例配置文件并将角色添加到该配置文件中的两个步骤。此示例信任策略允许 Amazon EC2 服务担任角色并查看 `example_bucket` Amazon S3 存储桶。该示例还假设您使用运行 Windows 的客户端计算机，并且已使用您的账户凭证和区域配置了命令行界面。有关更多信息，请参阅[配置 Amazon 命令行界面](#)。

在此示例中，在您创建角色时，在第一个命令中包括以下信任策略。此信任策略允许 Amazon EC2 服务代入角色。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"Service": "ec2.amazonaws.com"},
    "Action": "sts:AssumeRole"
  }
}
```

当您使用第二个命令时，您必须将权限策略附加到角色。下面的示例权限策略仅允许角色对 `example_bucket` Amazon S3 存储桶执行 `ListBucket` 操作。

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



```
"Action": "s3:ListBucket",
"Resource": "arn:aws:s3:::example_bucket"
}
}
```

要创建此 `Test-Role-for-EC2` 角色，您必须先将上述名为 `trustpolicyforec2.json` 的信任策略和上述名为 `permissionspolicyforec2.json` 的权限策略保存到您的本地 C: 驱动器中的 `policies` 目录。然后，您可以使用以下命令来创建角色，附加策略，创建实例配置文件，并将角色添加到实例配置文件中。

```
# Create the role and attach the trust policy that allows EC2 to assume this role.
$ aws iam create-role --role-name Test-Role-for-EC2 --assume-role-policy-document file://C:\policies\trustpolicyforec2.json

# Embed the permissions policy (in this example an inline policy) to the role to specify what it is allowed to do.
$ aws iam put-role-policy --role-name Test-Role-for-EC2 --policy-name Permissions-Policy-For-Ec2 --policy-document file://C:\policies\permissionspolicyforec2.json

# Create the instance profile required by EC2 to contain the role
$ aws iam create-instance-profile --instance-profile-name EC2-ListBucket-S3

# Finally, add the role to the instance profile
$ aws iam add-role-to-instance-profile --instance-profile-name EC2-ListBucket-S3 --role-name Test-Role-for-EC2
```

启动 EC2 实例时，如果您使用 Amazon 控制台，请在 `Configure Instance Details` (配置实例详细信息) 页面中指定实例配置文件名称。如果使用 `aws ec2 run-instances` CLI 命令，请指定 `--iam-instance-profile` 参数。

创建用于服务的角色 (Amazon API)

从 Amazon API 创建角色涉及几个步骤。当您使用控制台创建角色时，很多步骤会自动完成，但是使用 API 时，您必须自行完成每一个步骤。您必须创建角色，然后为角色分配权限策略。如果您使用的服务是 Amazon EC2，则还必须创建实例配置文件并向其添加角色。（可选）您还可以为您的角色设置[权限边界](#) (p. 370)。

为 Amazon 服务创建角色 (Amazon API)

1. 创建角色：[CreateRole](#)

您可以指定一个文件位置作为角色的信任策略。

2. 将托管权限策略附加到角色：[AttachRolePolicy](#)

或

为角色创建内联权限策略：[PutRolePolicy](#)

3. （可选）通过附加标签来向用户添加自定义属性：[TagRole](#)

有关更多信息，请参阅[管理 IAM 用户 \(Amazon CLI 或 Amazon API\) 的标签](#) (p. 280)。

4. （可选）为角色设置[权限边界](#) (p. 370)：[PutRolePermissionsBoundary](#)

权限边界控制角色可以具有的最大权限。权限边界是一项高级 Amazon 功能。

如果要将角色与 Amazon EC2 或使用 Amazon EC2 的其他 Amazon 服务结合使用，则必须将角色存储在实例配置文件中。实例配置文件是角色的容器。每个实例配置文件只能包含一个角色，不能提高该限制。如果在 Amazon Web Services Management Console 中创建角色，则为您创建具有与角色相同的名称的实例配置文件。有关实例配置文件的更多信息，请参阅[使用实例配置文件](#) (p. 258)。有关如何通过角色启动 Amazon EC2 实例的信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的[控制对 Amazon EC2 资源的访问](#)。

创建实例配置文件并在其中存储角色 (Amazon API)

1. 创建实例配置文件：[CreateInstanceProfile](#)
2. 向实例配置文件添加角色：[AddRoleToInstanceProfile](#)

针对第三方身份提供程序创建角色 (联合)

您可以使用身份提供程序而不必在 Amazon 账户中创建 IAM 用户。利用身份提供程序 (IdP)，您可以管理 Amazon 外部的用户身份，并向这些外部用户身份授予访问您账户中的 Amazon 资源的权限。有关联合和身份提供程序的更多信息，请参阅[身份提供程序和联合身份验证](#) (p. 170)。

为联合身份用户创建角色 (控制台)

为联合身份用户创建角色的过程取决于您选择的第三方提供商：

- 有关 Web 身份或 OpenID Connect (OIDC)，请参阅[创建用于 Web 联合身份验证或 OpenID Connect 联合身份验证的角色 \(控制台\)](#) (p. 226)。
- 有关 SAML 2.0，请参阅[创建用于 SAML 2.0 联合身份验证的角色 \(控制台\)](#) (p. 231)。

为联合访问创建角色 (Amazon CLI)

从 Amazon CLI 中为支持的身份提供程序 (OIDC 或 SAML) 创建角色的步骤是相同的。区别在于，您在先决条件步骤中创建的信任策略的内容不同。首先，按照先决条件部分中针对您正在使用的提供商类型的步骤操作：

- 有关 OIDC 提供商，请参阅[创建用于 Web 身份或 OIDC 的角色的先决条件](#) (p. 226)。
- 有关 SAML 提供商，请参阅[创建用于 SAML 的角色的先决条件](#) (p. 231)。

从 Amazon CLI 创建角色涉及几个步骤。当您使用控制台创建角色时，很多步骤会自动完成，但是使用 Amazon CLI 时，您必须自行完成每一个步骤。您必须创建角色，然后为角色分配权限策略。(可选) 您还可以为您的角色设置[权限边界](#) (p. 370)。

为联合身份创建角色 (Amazon CLI)

1. 创建角色：[aws iam create-role](#)
2. 将权限策略附加到角色：[aws iam attach-role-policy](#)

或

为角色创建内联权限策略：[aws iam put-role-policy](#)

3. (可选) 通过附加标签来向角色添加自定义属性：[aws iam tag-role](#)

有关更多信息，请参阅[管理 IAM 角色 \(Amazon CLI 或 Amazon API\) 的标签](#) (p. 282)。

4. (可选) 为角色设置[权限边界](#) (p. 370)：[aws iam put-role-permissions-boundary](#)

权限边界控制角色可以具有的最大权限。权限边界是一项高级 Amazon 功能。

以下示例演示了在简单环境中创建身份提供程序角色的前两个最常见的步骤。此示例允许 123456789012 账户中的任何用户担任角色并查看 example_bucket Amazon S3 存储桶。该示例还假设您在运行 Windows 的计算机上运行 Amazon CLI，并且已使用您的凭证配置了 Amazon CLI。有关更多信息，请参阅[配置 Amazon Command Line Interface](#)。

以下示例信任策略是为用户使用 Amazon Cognito 登录时的移动应用程序设计的。在此示例中，**us-east:12345678-ffff-ffff-ffff-123456** 表示由 Amazon Cognito 分配的身份池 ID。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "RoleForCognito",
    "Effect": "Allow",
    "Principal": {"Federated": "cognito-identity.amazonaws.com"},
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud": "us-east:12345678-ffff-ffff-ffff-123456"}}
  }
}
```

下面的权限策略仅允许担任角色的任何人对 example_bucket Amazon S3 存储桶执行 ListBucket 操作。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
  }
}
```

要创建此 Test-Cognito-Role 角色，您必须首先将上述名为 trustpolicyforcognitofederation.json 的信任策略和上述名为 permspolicyforcognitofederation.json 的权限策略保存到您的本地 policies 驱动器中的 C: 文件夹。然后您可以使用以下命令来创建角色并附加内联策略。

```
# Create the role and attach the trust policy that enables users in an account to assume
the role.
$ aws iam create-role --role-name Test-Cognito-Role --assume-role-policy-document file://C:\policies\trustpolicyforcognitofederation.json

# Attach the permissions policy to the role to specify what it is allowed to do.
aws iam put-role-policy --role-name Test-Cognito-Role --policy-name Perms-Policy-For-CognitoFederation --policy-document file://C:\policies\permspolicyforcognitofederation.json
```

为联合访问创建角色 (Amazon API)

从 Amazon CLI 中为支持的身份提供程序 (OIDC 或 SAML) 创建角色的步骤是相同的。区别在于，您在先决条件步骤中创建的信任策略的内容不同。首先，按照先决条件部分中针对您正在使用的提供商类型的步骤操作：

- 有关 OIDC 提供商，请参阅[创建用于 Web 身份或 OIDC 的角色的先决条件](#) (p. 226)。
- 有关 SAML 提供商，请参阅[创建用于 SAML 的角色的先决条件](#) (p. 231)。

为联合身份创建角色 (Amazon API)

1. 创建角色：[CreateRole](#)
 2. 将权限策略附加到角色：[AttachRolePolicy](#)
- 或
- 为角色创建内联权限策略：[PutRolePolicy](#)
3. (可选) 通过附加标签来向用户添加自定义属性：[TagRole](#)

有关更多信息，请参阅[管理 IAM 用户 \(Amazon CLI 或 Amazon API\) 的标签](#) (p. 280)。

4. (可选) 为角色设置权限边界 (p. 370) : [PutRolePermissionsBoundary](#)

权限边界控制角色可以具有的最大权限。权限边界是一项高级 Amazon 功能。

创建用于 Web 联合身份验证或 OpenID Connect 联合身份验证的角色 (控制台)

您可以使用 Web 联合身份验证或 OpenID Connect 联合身份验证 (OIDC) 身份提供程序, 而不必在 Amazon Web Services 账户中创建 Amazon Identity and Access Management 用户。利用身份提供程序 (IdP), 您可以管理 Amazon 外部的用户身份, 并向这些外部用户身份授予访问您账户中的 Amazon 资源的权限。有关联合身份验证和 IdP 的更多信息, 请参阅 [身份提供程序和联合身份验证](#) (p. 170)。

创建用于 Web 身份或 OIDC 的角色的先决条件

您必须先完成以下先决条件步骤, 然后才能创建用于 Web 联合身份验证的角色。

准备创建用于 Web 联合身份验证的角色

1. 注册一项或多项提供 OIDC 联合身份的服务。如果创建需要访问 Amazon 资源的应用程序, 则还需要使用提供商信息来配置应用程序。当您执行此操作时, 提供程序会为您提供一个应用程序 ID 或受众 ID, 该 ID 在您的应用程序中具有唯一性。(不同的提供程序使用不同的术语表示此过程。此指南使用配置一词表示通过提供程序标识应用程序的过程。) 可以对每个提供商配置多个应用程序, 也可以对单个应用程序配置多个提供商。查看有关使用身份提供程序的信息, 如下所示:

- [Login with Amazon 开发人员中心](#)
- 在 Facebook 开发人员网站上, 将 [Facebook 登录名](#) 添加到您的应用程序或网站。
- 在 Google 开发人员网站上, 使用 [OAuth 2.0 进行登录 \(OpenID Connect\)](#)。

2. 从 IdP 接收必要信息后, 在 IAM 中创建 IdP。有关更多信息, 请参阅 [创建 OpenID Connect \(OIDC\) 身份提供程序](#) (p. 180)。

Important

如果您使用的是 Google、Facebook 或 Amazon Cognito 的 OIDC IdP, 请勿在 Amazon Web Services Management Console 中创建单独的 IAM IdP。这些 OIDC 身份提供程序已内置到 Amazon, 并可供您使用。跳过此步骤并在以下步骤中使用 IdP 创建新角色。

3. 为已进行 IdP 身份验证的用户要担任的角色准备策略。正如任何角色一样, 移动应用程序的角色包含两个策略。一个是指定谁可以担任角色的信任策略。另一个是指定允许或拒绝移动应用程序访问的 Amazon 操作和资源的权限策略。

对于 Web IdP, 我们建议您使用 [Amazon Cognito](#) 来管理身份。在这种情况下, 请使用类似于以下示例的信任策略。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": { "Federated": "cognito-identity.amazonaws.com" },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
      "StringEquals": { "cognito-identity.amazonaws.com:aud": "us-east-2:12345678-  
abcd-abcd-abcd-123456" },
      "ForAnyValue:StringLike": { "cognito-identity.amazonaws.com:amr":  
"unauthenticated" }
    }
  }
}
```

将 us-east-2:12345678-abcd-abcd-abcd-123456 替换为由 Amazon Cognito 分配给您的身份池 ID。

如果您手动配置 Web 身份 IdP，则在创建信任策略时，必须使用三个值来确保只有您的应用程序可以担任此角色：

- 对于 Action 元素，使用 `sts:AssumeRoleWithWebIdentity` 操作。
- 对于 Principal 元素，使用字符串 `{"Federated":providerUrl/providerArn}`。
 - 对于一些常见的 OIDC IdP，*providerUrl* 是 URL。以下示例包括为一些常见 IdP 指定主体的方法：

```
"Principal":{"Federated":"cognito-identity.amazonaws.com"}
```

```
"Principal":{"Federated":"www.amazon.com"}
```

```
"Principal":{"Federated":"graph.facebook.com"}
```

```
"Principal":{"Federated":"accounts.google.com"}
```

- 对于其他的 OIDC 提供程序，请使用您在 [Step 2](#) 中创建的 OIDC IdP 的 Amazon 资源名称 (ARN)，如下例所示：

```
"Principal":{"Federated":"arn:aws:iam::123456789012:oidc-provider/
server.example.com"}
```

- 对于 Condition 元素，使用 `StringEquals` 条件来限制权限。测试身份池 ID (对于 Amazon Cognito) 或应用程序 ID (对于其他提供商)。身份池 ID 应与您通过 IdP 配置应用程序时获得的应用程序 ID 一致。ID 一致可确保请求来自您的应用程序。根据您所使用的 IdP 创建类似于以下示例之一的条件元素：

```
"Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud": "us-
east:12345678-ffff-ffff-ffff-123456"}}
```

```
"Condition": {"StringEquals": {"www.amazon.com:app_id":
"amzn1.application-oa2-123456"}}
```

```
"Condition": {"StringEquals": {"graph.facebook.com:app_id":
"111222333444555"}}
```

```
"Condition": {"StringEquals": {"accounts.google.com:aud":
"66677788899900pro0"}}
```

对于 OIDC 提供商，请将 OIDC IdP 的完全限定 URL 与 `aud` 上下文密钥一起使用，如下示例所示：

```
"Condition": {"StringEquals": {"server.example.com:aud":
"appid_from_oidc_idp"}}
```

Note

角色的信任策略中的主体的值将因 IdP 而异。用于 Web 联合身份验证或 OIDC 的角色只能指定一个主体。因此，如果移动应用程序允许用户从多个 IdP 登录，请为您想要支持的每个 IdP 创建单独角色。为每个 IdP 创建单独的信任策略。

如果用户使用移动应用程序从 Login with Amazon 登录，则将应用以下示例信任策略。在示例中，*amzn1.application-oa2-123456* 表示使用 Login with Amazon 配置应用程序时由 Amazon 分配的应用程序 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "RoleForLoginWithAmazon",
```

```
    "Effect": "Allow",
    "Principal": {"Federated": "www.amazon.com"},
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {"StringEquals": {"www.amazon.com:app_id": "amzn1.application-  
oa2-123456"}}
  }
}
```

如果用户使用移动应用程序从 Facebook 登录，则将应用以下示例信任策略。在此示例中，**111222333444555** 表示由 Facebook 分配的应用程序 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "RoleForFacebook",
    "Effect": "Allow",
    "Principal": {"Federated": "graph.facebook.com"},
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {"StringEquals": {"graph.facebook.com:app_id":  
"111222333444555"}}
  ]
}
```

如果用户使用移动应用程序从 Google 登录，则将应用以下示例信任策略。在此示例中，**666777888999000** 表示由 Google 分配的应用程序 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "RoleForGoogle",
    "Effect": "Allow",
    "Principal": {"Federated": "accounts.google.com"},
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {"StringEquals": {"accounts.google.com:aud": "666777888999000"}}
  ]
}
```

如果用户使用移动应用程序从 Amazon Cognito 登录，则将应用以下示例信任策略。在此示例中，**us-east:12345678-ffff-ffff-ffff-123456** 表示由 Amazon Cognito 分配的身份池 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "RoleForCognito",
    "Effect": "Allow",
    "Principal": {"Federated": "cognito-identity.amazonaws.com"},
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud": "us-  
east:12345678-ffff-ffff-ffff-123456"}}
  ]
}
```


创建用于 Web 身份或 OIDC 的角色

完成先决条件后，您可在 IAM 中创建角色。以下过程介绍了如何在 Amazon Web Services Management Console 中创建用于 Web 联合身份验证或 OIDC 联合身份验证的角色。要从 Amazon CLI 或 Amazon API 创建角色，请参阅[针对第三方身份提供程序创建角色（联合）](#) (p. 224) 中的过程。

Important

如果您使用的是 Amazon Cognito，则使用 Amazon Cognito 控制台来设置角色。否则，使用 IAM 控制台创建用于 Web 联合身份验证的角色。

创建适用于 Web 联合身份验证的 IAM 角色

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Roles，然后选择 Create role。
3. 选择 Web Identity (Web 联合身份验证) 角色类型。
4. 对于 Identity provider (身份提供程序)，请为您的角色选择 IdP：
 - 如果要为单个 Web IdP 创建角色，请选择 Login with Amazon、Facebook 或 Google。

Note

您必须为想要支持的每个 IdP 创建单独的角色。

- 如果要为 Amazon Cognito 创建高级方案角色，请选择 Amazon Cognito。

Note

您仅在处理高级方案时需要手动创建与 Amazon Cognito 一起使用的角色。否则，Amazon Cognito 可以为您创建角色。有关 Amazon Cognito 的更多信息，请参阅 Amazon 移动 SDK for iOS 开发人员指南中的[Amazon Cognito 身份](#)，以及 Amazon 移动 SDK for Android 开发人员指南中的[Amazon Cognito 身份](#)。

5. 输入您的应用程序标识符。标识符的标签会根据所选提供程序发生变化：
 - 如果要为 Login with Amazon 创建角色，请在 Application ID (应用程序 ID) 框中输入应用程序 ID。
 - 如果要为 Facebook 创建角色，请在 Application ID (应用程序 ID) 框中输入应用程序 ID。
 - 如果要为 Google 创建角色，请在 Audience (受众) 框中输入受众名称。
 - 如果要为 Amazon Cognito 创建角色，请在 Identity Pool ID (身份池 ID) 框中输入您为 Amazon Cognito 应用程序创建的身份池的 ID。
6. (可选) 选择 Condition (optional) [条件 (可选)] 以创建在应用程序的用户可以使用角色授予的权限之前必须满足的其他条件。例如，您可以添加一个条件，仅授权特定 IAM 用户 ID 访问 Amazon 资源。
7. 检查 Web 身份信息，然后选择 Next (下一步)。
8. IAM 包括您的账户中的 Amazon 托管策略和客户托管策略的列表。选择要用于权限策略的策略，或者选择 Create policy (创建策略) 以打开新的浏览器选项卡并从头开始创建新策略。有关更多信息，请参阅[创建 IAM 策略](#) (p. 438)。在您创建策略后，关闭该选项卡并返回到您的原始选项卡。选中您希望 Web 身份用户具有的权限策略旁边的复选框。如果您愿意，此时可以不选择任何策略，稍后将策略附加到角色。默认情况下，角色没有权限。
9. (可选) 设置[权限边界](#) (p. 370)。这是一项高级功能。

打开 Permissions boundary (权限边界) 部分，然后选择 Use a permissions boundary to control the maximum role permissions (使用权限边界控制最大角色权限)。选择要用于权限边界的策略。

10. 选择 Next (下一步)。
11. 对于 Role name (角色名称)，输入一个角色名称。角色名称在您的 Amazon Web Services 账户内必须是唯一的。不区分大小写。例如，您无法同时创建名为 **PRODRole** 和 **prodrrole** 的角色。由于其他 Amazon 资源可能会引用此角色，因此您无法在创建角色后编辑角色名称。

12. (可选) 对于 Description (描述), 输入新角色的描述。
13. 要编辑角色的使用案例和权限, 在 Step 1: Select trusted entities (步骤 1: 选择可信实体) 或 Step 2: Add permissions (步骤 2: 添加权限) 部分中选择 Edit (编辑)。
14. (可选) 要向角色添加元数据, 请以键值对的形式附加标签。有关在 IAM 中使用标签的更多信息, 请参阅 [标记 IAM 资源 \(p. 276\)](#)。
15. 检查角色, 然后选择 Create role。

为 GitHub OIDC 身份提供程序配置角色

如果您将 GitHub 用作 OIDC IdP, 则最佳实践是限制可以代入与 IAM IdP 关联的角色的实体。如果信任策略中包含条件语句, 则可将角色限制为特定的 GitHub 组织、存储库或分支。您可以使用条件键 `token.actions.githubusercontent.com:sub` 限制访问。我们建议您将条件限制为一组特定的存储库或分支。如果您未包含此条件, 则来自您控制范围之外的组织或存储库的 GitHub 操作可代入与您 Amazon Web Services 账户中 GitHub IAM IdP 关联的角色。有关如何将 Amazon 配置为联合身份以信任 GitHub 的 OIDC 的信息, 请参阅 [GitHub 文档 – 在 Amazon Web Services 中配置 OpenID Connect](#)。

以下示例信任策略限制对定义的 GitHub 组织、存储库和分支的访问。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::012345678910:oidc-provider/
token.actions.githubusercontent.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
          "token.actions.githubusercontent.com:sub": "repo:GitHubOrg/GitHubRepo:ref:refs/
heads/GitHubBranch"
        }
      }
    }
  ]
}
```

以下示例条件限制对定义的 GitHub 组织和存储库的访问, 但授予对存储库内任何分支的访问权限。

```
"Condition": {
  "StringLike": {
    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
    "token.actions.githubusercontent.com:sub": "repo:GitHubOrg/GitHubRepo:*"
  }
}
```

以下示例条件限制对定义的 GitHub 组织内任何存储库或分支的访问。我们建议将此作为将 GitHub 用作 OIDC IdP 的角色的最低信任策略条件。

```
"Condition": {
  "StringLike": {
    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
    "token.actions.githubusercontent.com:sub": "repo:GitHubOrg/*"
  }
}
```

有关可用于策略中的条件检查的 Web 联合身份验证密钥的更多信息，请参阅[Amazon Web 联合身份验证的可用键](#) (p. 1016)。

创建用于 SAML 2.0 联合身份验证的角色 (控制台)

您可以使用 SAML 2.0 联合身份验证而不必在 Amazon 账户中创建 IAM 用户。利用身份提供程序 (IdP)，您可以管理 Amazon 外部的用户身份，并向这些外部用户身份授予访问您账户中的 Amazon 资源的权限。有关联合和身份提供程序的更多信息，请参阅[身份提供程序和联合身份验证](#) (p. 170)。

创建用于 SAML 的角色的先决条件

您必须先完成以下先决条件步骤，然后才能创建用于 SAML 2.0 联合身份验证的角色。

准备创建用于 SAML 2.0 联合的角色

1. 在创建用于 SAML 联合的角色之前，必须在 IAM 中创建 SAML 提供商。有关更多信息，请参阅[创建 IAM SAML 身份提供程序](#) (p. 188)。
2. 为已进行 SAML 2.0 身份验证的用户要担任的角色准备策略。正如任何角色一样，用于 SAML 联合的角色包含两个策略。一个是指定谁可以代入角色的角色信任策略。另一个是指定允许或拒绝联合身份用户访问的 Amazon 操作和资源的 IAM 权限策略。

在为角色创建信任策略时，必须使用三个值来确保只有您的应用程序可以代入此角色：

- 对于 Action 元素，使用 sts:AssumeRoleWithSAML 操作。
- 对于 Principal 元素，使用字符串 {"Federated": "**ARNofIdentityProvider**"}。将 **ARNofIdentityProvider** 替换为您在[Step 1](#) 中创建的 [SAML 身份提供程序](#) (p. 176) 的 ARN。
- 对于 Condition 元素，使用 StringEquals 条件测试 SAML 响应中的 saml:aud 属性是否匹配 Amazon 的 SAML 联合终端节点。

以下示例信任策略是为 SAML 联合身份用户设计的：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRoleWithSAML",
    "Principal": {"Federated": "arn:aws:iam::account-id:saml-provider/PROVIDER-NAME"},
    "Condition": {"StringEquals": {"SAML:aud": "https://signin.aws.amazon.com/saml"}}
  }
}
```

将主体 ARN 替换为您在 IAM 中创建的 SAML 提供商的实际 ARN。它会具备您自己的账户 ID 和提供商名称。

创建用于 SAML 的角色

在完成先决条件步骤后，您可以创建用于基于 SAML 的联合身份验证的角色。

要创建用于基于 SAML 的联合的角色，请执行以下操作

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在 IAM 控制台的导航窗格中，依次选择 Roles (角色) 和 Create role (创建角色)。
3. 选择 SAML 2.0 federation 角色类型。
4. 对于 Select a SAML provider (选择 SAML 提供商)，请为您的角色选择提供商。
5. 选择 SAML 2.0 访问级别方法。

- 选择 Allow programmatic access only (只允许编程访问) 以创建可从 Amazon API 或 Amazon CLI 以编程方式担任的角色。
- 选择允许编程访问和 Amazon Web Services Management Console 访问以创建可以编程方式和从 Amazon Web Services Management Console 中担任的角色。

通过这两种方法创建的角色类似，但也可从控制台担任的角色包括包含带特定条件的信任策略。该条件显式确保将 SAML 受众 (SAML:aud 属性) 设置为 SAML 的 Amazon 登录终端节点 (<https://signin.aws.amazon.com/saml>)。

6. 如果创建用于编程访问的角色，请从属性列表选择一个属性。然后在 Value (值) 框中，键入一个将包含在角色中的值。这样可仅限来自其 SAML 身份验证响应 (断言) 包括您指定的属性的身份提供程序的用户可访问该角色。必须指定至少一个属性，以确保您的角色限于您所在组织中的一部分用户。

如果要创建用于编程访问和控制台访问的角色，则将自动添加 SAML:aud 属性，并将其设置为 Amazon SAML 终端节点的 URL (<https://signin.aws.amazon.com/saml>)。

7. 要将更多与属性相关的条件添加到信任策略，请选择 Condition (optional) [条件 (可选)]，选择其他条件，然后指定值。

Note

列表包括最常用的 SAML 属性。IAM 支持其他可用于创建条件的属性。有关支持的属性的列表，请参阅 [SAML 联合身份验证的可用键](#)。如果需要不在列表中的支持的 SAML 属性的条件，可以手动添加此条件。为此，请在创建角色后编辑信任策略。

8. 检查 SAML 2.0 信任信息，然后选择 Next (下一步) 。
9. IAM 包括您的账户中的 Amazon 托管策略和客户托管策略的列表。选择要用于权限策略的策略，或者选择 Create policy (创建策略) 以打开新的浏览器选项卡并从头开始创建新策略。有关更多信息，请参阅 [创建 IAM 策略 \(p. 438\)](#)。在您创建策略后，关闭该选项卡并返回到您的原始选项卡。选中您希望 Web 身份用户具有的权限策略旁边的复选框。如果您愿意，此时可以不选择任何策略，稍后将策略附加到角色。默认情况下，角色没有权限。
10. (可选) 设置 [权限边界 \(p. 370\)](#)。这是一项高级功能。

打开 Permissions boundary (权限边界) 部分，然后选择 Use a permissions boundary to control the maximum role permissions (使用权限边界控制最大角色权限)。选择要用于权限边界的策略。

11. 选择 Next (下一步)。
12. 请选择 Next: Review (下一步 : 审核) 。
13. 对于 Role name (角色名称)，输入一个角色名称。角色名称在您的 Amazon 账户内必须是唯一的。名称不区分大小写。例如，您无法同时创建名为 **PRODROLE** 和 **prodrole** 的角色。由于其他 Amazon 资源可能引用该角色，角色创建完毕后无法编辑角色名称。
14. (可选) 对于 Description (描述)，输入新角色的描述。
15. 在 Step 1: Select trusted entities (步骤 1 : 选择可信实体) 或 Step 2: Add permissions (步骤 2 : 添加权限) 部分中的 Edit (编辑)，以编辑角色的用户案例和权限。
16. (可选) 通过以键值对的形式附加标签来向角色添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 [标记 IAM 资源 \(p. 276\)](#)。
17. 检查角色，然后选择 Create role。

创建角色后，通过使用有关 Amazon 的信息来配置您的身份提供程序软件，以完成 SAML 信任。此类信息包括您希望联合身份用户使用的角色。这称为在 IdP 和 Amazon 之间配置信赖方信任。有关更多信息，请参阅 [配置具有信赖方信任的 SAML 2.0 IdP 并添加陈述 \(p. 192\)](#)。

使用自定义信任策略创建角色 (控制台)

您可以创建自定义信任策略来委托访问权限并允许其他人在您的 Amazon 账户中执行操作。有关更多信息，请参阅 [创建 IAM 策略 \(p. 438\)](#)。

有关如何使用角色委派权限的信息，请参阅[角色术语和概念](#) (p. 158)。

使用自定义信任策略创建 IAM 角色 (控制台)

您可以使用 Amazon Web Services Management Console 创建 IAM 用户可担任的角色。例如，假设贵组织拥有多个 Amazon 账户以便将开发环境与生产环境隔离。有关创建角色 (该角色允许开发账户中的用户访问生产账户中的资源) 的概述信息，请参阅[使用不同的开发和生产账户的示例方案](#) (p. 162)。

使用自定义信任策略创建角色 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在控制台的导航窗格中，选择 Roles，然后选择 Create role。
3. 选择 Custom trust policy (自定义信任策略) 角色类型。
4. 在 Custom trust policy (自定义信任策略) 部分，输入或粘贴角色的自定义信任策略。有关更多信息，请参阅[创建 IAM 策略](#) (p. 438)。
5. 选择 Next (下一步)。
6. 选中您创建的自定义信任策略旁的复选框。
7. (可选) 设置[权限边界](#) (p. 370)。这是一项高级功能，可用于服务角色，但不可用于服务相关角色。

打开 Permissions boundary (权限边界) 部分，然后选择 Use a permissions boundary to control the maximum role permissions (使用权限边界控制最大角色权限)。IAM 包括您的账户中的 Amazon 托管策略和客户托管策略的列表。选择要用于权限边界的策略。

8. 选择 Next (下一步)。
9. 对于 Role name (角色名称)，角色名称自定义的程度由服务定义。如果服务定义角色的名称，则此选项不可编辑。在其他情况下，服务可能定义角色的前缀并允许您键入可选的后缀。某些服务允许您指定角色的整个名称。

如果可能，输入角色名称或角色名称后缀。角色名称在您的 Amazon 账户内必须是唯一的。名称不区分大小写。例如，您无法同时创建名为 **PRODROLE** 和 **prodrole** 的角色。由于其他 Amazon 资源可能引用该角色，角色创建完毕后无法编辑角色名称。

10. (可选) 对于 Description (描述)，输入新角色的描述。
11. 在 Step 1: Select trusted entities (步骤 1：选择可信实体) 或 Step 2: Add permissions (步骤 2：添加权限) 部分中选择 Edit (编辑)，以编辑角色的自定义策略和权限。
12. (可选) 通过以键值对的形式附加标签来向角色添加元数据。有关在 IAM 中使用标签的更多信息，请参阅[标记 IAM 资源](#) (p. 276)。
13. 检查角色，然后选择 Create role。

委派访问权限的策略示例

以下示例演示了如何允许或授予 Amazon 账户访问另一个 Amazon 账户中的资源。要了解如何使用这些示例 JSON 策略文档创建 IAM 策略，请参阅[the section called “在“JSON”选项卡上创建策略”](#) (p. 438)。

主题

- [使用角色委托针对另一个 Amazon 账户的资源的访问权限](#) (p. 234)
- [使用策略将访问权限委托给服务](#) (p. 234)
- [使用基于资源的策略委托访问另一个账户的 Amazon S3 存储桶](#) (p. 234)
- [使用基于资源的策略委托针对另一个账户中的 Amazon SQS 队列的访问权限](#) (p. 235)
- [当拒绝访问账户时，不得委托访问](#) (p. 236)

使用角色委托针对另一个 Amazon 账户的资源访问权限

有关介绍如何使用 IAM 角色对一个账户中的用户授权以访问另一个账户中 Amazon 资源的教程，请参阅 [IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限 \(p. 31\)](#)。

Important

您可以在角色信任策略的 Principal 元素中包含特定角色或用户的 ARN。保存策略时，Amazon 将该 ARN 转换为唯一主体 ID。如果有人希望通过删除并重新创建角色或用户来提升特权，这样有助于减轻此类风险。您通常不会在控制台中看到这个 ID，因为显示信任策略时它还会反向转换为 ARN。但是，如果您删除角色或用户，这种关系即被打破。即使您重新创建用户或角色，策略也不再适用。因为与信任策略中存储的 ID 不匹配。在这种情况下，主体 ID 会显示在控制台中，因为 Amazon 无法将其映射回 ARN。结果是，如果您删除并重新创建了信任策略的 Principal 元素所引用的用户或角色，您必须编辑角色，替换 ARN。当您保存策略时，它会转换为新的主体 ID。

使用策略将访问权限委托给服务

以下示例显示了一个可以附加到角色的策略。该策略可使两个服务 Amazon EMR 和 Amazon Data Pipeline 担任此角色。然后，这些服务可执行由分配给该角色（未显示）的权限策略授权执行的任何任务。要指定多个服务主体，不用指定两个 Service 元素；您可以只使用一个该元素。实际上，您可以将一组多个服务主体作为单个 Service 元素的值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "elasticmapreduce.amazonaws.com",
          "datapipeline.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

使用基于资源的策略委托访问另一个账户的 Amazon S3 存储桶

在此示例中，账户 A 使用基于资源的策略（一个 Amazon S3 [存储桶策略](#)）授权账户 B 针对账户 A 的 S3 存储桶的完全访问权限。然后，账户 B 创建一个 IAM 用户策略，向账户 B 中的一个用户授予针对账户 A 的存储桶的访问权限。

账户 A 中的 S3 存储桶策略可能与以下策略类似。在此示例中，账户 A 的 S3 存储桶被命名为 mybucket，账户 B 的账号为 111122223333。它在账户 B 中未指定任何单个用户或组，仅指定账户本身。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "AccountBAccess1",
    "Effect": "Allow",
    "Principal": {"AWS": "111122223333"},
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3::mybucket",
      "arn:aws:s3::mybucket/*"
    ]
  }
}
```



```
}
```

或者，账户 A 可使用 Amazon S3 [访问控制列表 \(ACL\)](#) 来授权账户 B 访问 S3 存储桶或某个存储桶内的单个对象。在这种情况下，唯一改变的是账户 A 授权访问账户 B 的方式。如此示例的下一个部分所述，账户 B 仍使用一个策略委托针对账户 B 中的 IAM 组的访问权限。有关控制对 S3 存储桶和对象访问的更多信息，请转到 Amazon Simple Storage Service 用户指南中的[访问控制](#)。

账户 B 的管理员可能创建以下策略示例。该策略向账户 B 中的组或用户授予读取访问权限。前一策略向账户 B 授予访问权限。但是，除非组或用户策略显式授予资源访问权限，否则账户 B 中的单个组 and 用户不能访问资源。此策略中的权限只能是上述跨账户策略中的权限的一个子集。相比于账户 A 在第一个策略中授予账户 B 的权限，账户 B 无法向其组和用户授予更多权限。在该策略中，将显式定义 Action 元素以仅允许 List 操作，而且该策略的 Resource 元素与由账户 A 执行的存储桶策略的 Resource 匹配。

为执行该策略，账户 B 使用 IAM 将它附加到账户 B 中的相应用户（或组）。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:List*",
    "Resource": [
      "arn:aws:s3::mybucket",
      "arn:aws:s3::mybucket/*"
    ]
  }
}
```

使用基于资源的策略委托针对另一个账户中的 Amazon SQS 队列的访问权限

在以下示例中，账户 A 有一个 Amazon SQS 队列，该队列使用附加到该队列的基于资源的策略向账户 B 授权队列访问权限。然后，账户 B 使用 IAM 组策略委托针对账户 B 中的组的访问权限。

以下示例队列策略授予账户 B 对名为 执行 queue1 的账户 A 的队列执行 SendMessage 和 ReceiveMessage 操作的权限，但只在 2014 年 11 月 30 日中午至下午 3:00 之间可行。Account B 的账号为 1111-2222-3333。账户 A 使用 Amazon SQS 执行该策略。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "111122223333"},
    "Action": [
      "sqs:SendMessage",
      "sqs:ReceiveMessage"
    ],
    "Resource": ["arn:aws:sqs*:123456789012:queue1"],
    "Condition": {
      "DateGreaterThan": {"aws:CurrentTime": "2014-11-30T12:00Z"},
      "DateLessThan": {"aws:CurrentTime": "2014-11-30T15:00Z"}
    }
  }
}
```

账户 B 向账户 B 中的组委托访问权限的策略可能类似于以下示例。账户 B 使用 IAM 将此策略附加到组（或用户）。

```
{
  "Version": "2012-10-17",
```

```
"Statement": {
  "Effect": "Allow",
  "Action": "sqs:*",
  "Resource": "arn:aws:sqs:*:123456789012:queue1"
}
```

在前述 IAM 用户策略示例中，账户 B 使用通配符授权其用户访问针对账户 A 的队列的所有 Amazon SQS 操作。但是账户 B 可委托的范围仅限于账户 B 被授权访问的范围。拥有第二个策略的账户 B 组只能在 2014 年 11 月 30 日中午至下午 3:00 之间访问该队列。根据账户 A 的 Amazon SQS 队列策略的定义，用户只能执行 `SendMessage` 和 `ReceiveMessage` 操作。

当拒绝访问账户时，不得委托访问

如果其他账户已显式拒绝访问用户的父账户，则 Amazon 账户不得委托针对该账户的资源的访问权限。此“拒绝”将传播到该账户内的所有用户，无论用户的现有策略是否授予这些用户访问权限。

例如，账户 A 编写了一个针对其账户中 S3 存储桶的存储桶策略，其中显式拒绝了账户 B 访问账户 A 的存储桶。但账户 B 编写了一个 IAM 用户策略，其中对账户 B 中的一个用户授予了对账户 A 的存储桶的访问权限。应用于账户 A 的 S3 存储桶的“显式拒绝”将传播到账户 B 中的用户。它会覆盖用于对账户 B 中用户授予访问权限的 IAM 用户策略。（有关如何计算权限的详细信息，请参阅 [策略评估逻辑 \(p. 972\)](#)。）

Account A 的存储段策略可能与下列策略类似。在此示例中，账户 A 的 S3 存储桶被命名为 `mybucket`，账户 B 的账号为 1111-2222-3333。账户 A 使用 Amazon S3 执行该策略。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "AccountBDeny",
    "Effect": "Deny",
    "Principal": {"AWS": "111122223333"},
    "Action": "s3:*",
    "Resource": "arn:aws:s3::mybucket/*"
  }
}
```

此“显式拒绝”将覆盖账户 B 中所有提供账户 A 中 S3 存储桶访问权限的策略。

使用 IAM 角色

您必须先对用户授予切换到您创建的角色权限，然后 IAM 用户、应用程序或服务才能使用该角色。您可使用附加到 IAM 用户组之一或用户本身的任何策略来授予所需权限。本部分描述如何授予用户使用角色的权限。它还解释了用户如何从 Amazon Web Services Management Console、Tools for Windows PowerShell、Amazon Command Line Interface (Amazon CLI) 和 [AssumeRole](#) API 切换到角色。

Important

当您以编程方式而不是在 IAM 控制台中创建角色，则除最长可达 64 个字符的 `RoleName` 外，您还可以选择添加最长 512 个字符的 `Path`。不过，如果您打算通过 Amazon Web Services Management Console 中的 Switch Role（切换角色）功能使用角色，则组合的 `Path` 和 `RoleName` 不能超过 64 个字符。

您可以从 Amazon Web Services Management Console 中切换角色。您可以调用 Amazon CLI 或 API 操作或使用自定义 URL 以担任角色。您使用的方法决定了谁可以担任该角色，以及角色会话可以持续多长时间。使用 `AssumeRole*` API 操作时，您所担任的 IAM 角色为资源。调用 `AssumeRole*` API 操作的 IAM 用户或角色是主体。

比较使用角色的方法

担任角色的方法	谁可以担任角色	用于指定凭证生命周期的方法	凭证生命周期 (最小值 最大值 默认值)
Amazon Web Services Management Console	IAM 用户 (通过 切换角色 (p. 243))	Role (角色) “Summary (摘要)”页面上的 Maximum session duration (最长会话持续时间)	1 小时 最大会话持续时间设置 ² 1 小时
assume-role CLI 或 AssumeRole API 操作	IAM 用户或角色 ¹	duration-seconds CLI 或 DurationSeconds API 参数	15 分 最大会话持续时间设置 ² 1 小时
assume-role-with-saml CLI 或 AssumeRoleWithSAML API 操作	使用 SAML 验证的任何用户	duration-seconds CLI 或 DurationSeconds API 参数	15 分 最大会话持续时间设置 ² 1 小时
assume-role-with-web-identity CLI 或 AssumeRoleWithWebIdentity API 操作	使用 Web 身份提供程序验证的任何用户	duration-seconds CLI 或 DurationSeconds API 参数	15 分 最大会话持续时间设置 ² 1 小时
使用 构造的 (p. 201) 控制台 URLAssumeRole	IAM 用户或角色	URL 中的 SessionDuration HTML 参数	15 分钟 12 小时 1 小时
使用 构造的 (p. 201) 控制台 URLAssumeRoleWithSAML	使用 SAML 验证的任何用户	URL 中的 SessionDuration HTML 参数	15 分钟 12 小时 1 小时
使用 构造的 (p. 201) 控制台 URLAssumeRoleWithWebIdentity	使用 Web 身份提供程序验证的任何用户	URL 中的 SessionDuration HTML 参数	15 分钟 12 小时 1 小时

¹ 使用一个角色的凭证担任其他角色称为 [role chaining \(p. 159\)](#) (角色链)。在使用角色链时，新凭证的最大持续时间限制为 1 小时。当您使用角色向 [EC2 实例上运行的应用程序授予权限 \(p. 252\)](#)时，则这些应用程序不受制于此限制。

² 该设置可以具有 1 小时到 12 小时之间的值。有关修改最大会话持续时间设置的详细信息，请参阅 [修改角色 \(p. 261\)](#)。该设置确定在获取角色凭证时可请求的最大会话持续时间。例如，在使用 [AssumeRole*](#) API 操作担任角色时，您可以使用 DurationSeconds 参数指定会话长度。可以使用该参数指定 900 秒 (15 分钟) 到角色的最大会话持续时间设置之间的角色会话长度。在控制台内切换角色的 IAM 用户被授予最大会话持续时间或 IAM 用户会话中的剩余时间 (以较少者为准)。假定您在角色上设置 5 小时的最大持续时间。已登录到控制台 10 小时 (默认最多 12 小时) 的 IAM 用户切换到该角色。可用角色会话持续时间为 2 小时。要了解如何查看您的角色的最大值，请参阅本页后面的[查看角色的最大会话持续时间设置 \(p. 238\)](#)。

注意

- 最大会话持续时间设置不限制 Amazon 服务建立的会话。
- 要允许用户在角色会话中重新代入当前角色，请将角色 ARN 或 Amazon Web Services 账户 ARN 指定为角色信任策略中的主体。提供计算资源的 Amazon Web Services (例如 Amazon

EC2、Amazon ECS、Amazon EKS 和 Lambda) 会提供临时凭证并自动轮换这些凭证。这样可以确保您始终拥有一组有效的凭证。对于这些服务, 无需重新代入当前角色即可获得临时凭证。但是, 如果您需要传递 [会话标签 \(p. 293\)](#) 或者 [会话策略 \(p. 357\)](#), 则需要重新代入当前角色。要了解如何修改角色信任策略以添加主体角色 ARN 或 Amazon Web Services 账户 ARN, 请参阅 [修改角色信任策略 \(控制台\) \(p. 261\)](#)。

主题

- [查看角色的最大会话持续时间设置 \(p. 238\)](#)
- [向用户授予切换角色的权限 \(p. 239\)](#)
- [向用户授予权限以将角色传递给 Amazon 服务 \(p. 240\)](#)
- [切换到角色 \(控制台\) \(p. 243\)](#)
- [切换到 IAM 角色 \(Amazon CLI\) \(p. 245\)](#)
- [切换到 IAM 角色 \(Tools for Windows PowerShell\) \(p. 249\)](#)
- [切换到 IAM 角色 \(Amazon API\) \(p. 250\)](#)
- [使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限 \(p. 252\)](#)
- [撤销 IAM 角色临时安全凭证 \(p. 259\)](#)

查看角色的最大会话持续时间设置

您可以使用 Amazon Web Services Management Console 或通过使用 Amazon CLI 或者 Amazon API 来指定角色的最大会话持续时长。在使用 Amazon CLI 或 API 操作代入角色时, 您可以为 `DurationSeconds` 参数指定一个值。您可以使用该参数指定 900 秒 (15 分钟) 到角色的最大会话持续时间设置之间的角色会话持续时间。在指定该参数之前, 应查看您的角色的该设置。如果指定的 `DurationSeconds` 参数值高于最大设置, 操作将失败。

查看角色的最大会话持续时间 (控制台)

1. 在 IAM 控制台的导航窗格中, 选择 Roles。
2. 选择要查看的角色的名称。
3. 在 Maximum session duration (最长会话持续时间) 旁, 查看为角色授予的最大会话长度。这是您可以在您的 Amazon CLI 或 API 操作中指定的最长会话持续时间。

查看角色的最大会话持续时间设置 (Amazon CLI)

1. 如果您不知道要承担的角色名称, 请运行以下命令列出账户中的角色:
 - [aws iam list-roles](#)
2. 要查看角色的最大会话持续时间, 请运行以下命令。然后, 查看最大会话持续时间参数。
 - [aws iam get-role](#)

查看角色的最大会话持续时间设置 (Amazon API)

1. 如果您不知道要承担的角色名称, 请调用以下操作以列出账户中的角色:
 - [ListRoles](#)
2. 要查看角色的最大会话持续时间, 请运行以下操作。然后, 查看最大会话持续时间参数。
 - [GetRole](#)

向用户授予切换角色的权限

当您[创建用于跨账户访问的角色 \(p. 215\)](#)时，您在拥有角色和资源的账户（信任账户）和包含用户的账户（可信账户）之间建立了信任。为此，信任账户的管理员指定可信账号为角色的信任策略中的 `Principal`。这可能会允许可信账户中的任何用户代入该角色。要完成配置，可信账户的管理员必须为该账户中的特定组或用户提供切换到该角色的权限。

要向用户授予切换到角色的权限，可信账户的管理员将为用户创建新策略。或者，管理员将编辑现有策略以添加所需元素。管理员随后可向用户发送链接，以使用户进入已填写所有详细信息的 `Switch Role`（切换角色）页面。或者，管理员可以为用户提供包含角色的账户 ID 号或账户别名以及角色名称。用户随后转到 `Switch Role` 页面，然后手动添加详细信息。有关用户如何切换角色的详细信息，请参阅[切换到角色（控制台） \(p. 243\)](#)。

请注意，仅当您以 IAM 用户、SAML 联合角色或 web 身份联合角色登录时才能切换角色。如果您以 Amazon Web Services 账户根用户身份登录，则无法切换角色。

Important

您无法将 Amazon Web Services Management Console 中的角色切换到需要 [ExternalId \(p. 164\)](#) 值的角色。您只能通过调用支持 `ExternalId` 参数的 `AssumeRole` API 来切换到此类角色。

注意

- 本主题讨论了用户的策略，因为我们最终会向用户授予完成任务的权限。但是，最好不要直接向单个用户授予权限。为方便管理，我们建议向 IAM 组分配策略并授予权限，然后使得用户成为相应组的成员。
- 当您在 Amazon Web Services Management Console 中切换角色时，控制台总是使用您的原始凭证对切换操作进行授权。无论您作为 IAM 用户、SAML 联合角色还是 Web 联合身份角色登录，上述情形均适用。例如，如果您切换到 `RoleA`，它会使用您的原始用户或联合角色凭证确定是否允许您担任 `RoleA`。如果您在使用 `RoleA` 时尝试切换到 `RoleB`，则会使用您的原始用户或联合角色凭证对您的尝试进行授权。`RoleA` 凭证不用于此操作。

主题

- [创建或编辑策略 \(p. 239\)](#)
- [向用户提供信息 \(p. 240\)](#)

创建或编辑策略

向用户授予担任角色权限的策略必须包括一个语句，该语句对以下项具有 `Allow` 影响：

- `sts:AssumeRole` 操作
- `Resource` 元素中该角色的 Amazon Resource Name (ARN)

如以下示例所示。获得了该策略 (通过组成员资格或直接附加) 的用户可以切换到指定角色。

Note

如果将 `Resource` 设置为 `*`，则用户可在信任用户账户的任何账户中担任任何角色。（换言之，角色的信任策略将用户的账户指定为 `Principal`）。作为最佳实践，我们建议您遵循[最低权限原则](#)并仅为用户所需的角色指定完整的 ARN。

以下示例所显示的策略允许用户仅在一个账户中担任角色。此外，该策略使用通配符 (`*`) 来指定用户仅在角色名称以字母 `Test` 开头时才能切换到该角色。

```
{
```

```
"Version": "2012-10-17",
"Statement": {
  "Effect": "Allow",
  "Action": "sts:AssumeRole",
  "Resource": "arn:aws:iam::account-id:role/Test*"
}
}
```

Note

角色向用户授予的权限不会添加到用户已获得的权限。当用户切换到某个角色时，用户可临时放弃其原始权限以换取由该角色授予的权限。用户退出该角色时，将自动恢复原始用户权限。例如，假如用户的权限允许使用 Amazon EC2 实例，但是角色的权限策略未授予这些权限。在这种情况下，使用角色时，用户无法在控制台使用 Amazon EC2 实例。此外，通过 AssumeRole 获取的临时凭证无法以编程方式使用 Amazon EC2 实例。

向用户提供信息

创建角色并向用户授予切换为该角色的权限后，您必须为用户提供以下信息：

- 角色的名称。
- 包含该角色的 ID 或账户别名

通过向用户发送使用账户 ID 和角色名称预配置的链接，可使用户更轻松地执行操作。在 Create Role 向导的最终页面上，或在启用了跨账户功能的任何角色的 Role Summary 页面上，都可以看到角色链接。

您还可使用以下格式手动构建链接。请用您的账户 ID 或别名及角色名称替换以下示例中的两个参数：

```
https://signin.aws.amazon.com/switchrole?
account=your_account_ID_or_alias&roleName=optional_path/role_name
```

我们建议您将用户定向到 [切换到角色（控制台）](#) (p. 243) 以指导他们完成该过程。要排除您在担任角色时可能遇到的常见问题，请参阅 [我无法代入角色](#) (p. 884)。

注意事项

- 如果您以编程方式创建角色，则可使用路径以及名称创建角色。如果执行此操作，则必须向用户提供完整的路径和角色名称，以便他们可以在 Amazon Web Services Management Console 的 Switch Role（切换角色）页面输入该角色。例如：division_abc/subdivision_efg/role_XYZ。
- 如果您以编程方式创建角色，则除 RoleName 外，您还可以添加最长 512 个字符的 Path。角色名称最多可以有 64 个字符。但是，要通过 Amazon Web Services Management Console 中的切换角色功能使用角色，则组合的 Path 和 RoleName 不能超过 64 个字符。
- 为了安全起见，您可以[查看 Amazon CloudTrail 日志](#) (p. 343) 以了解已在 Amazon 中执行操作的人员。您可以在角色信任策略中使用 sts:SourceIdentity 条件键，以要求用户在代入角色时指定身份。例如，您可以要求 IAM 用户指定自己的用户名作为其源身份。这可以帮助您确定哪个用户在 Amazon 中执行了具体的操作。有关更多信息，请参阅[sts:SourceIdentity](#) (p. 1024)。您可以使用 [sts:RoleSessionName](#) (p. 1023)，以要求用户在代入角色时指定会话名称。这可以帮助您在不同主体使用角色时区分角色会话。

向用户授予权限以将角色传递给 Amazon 服务

要配置多项 Amazon 服务，您必须将 IAM 角色传递给相应服务。这允许该服务稍后代入该角色并代表您执行操作。对于大多数服务，您只需在设置期间（而不是服务每次代入角色时）将角色传递给服务。例如，假定您的一款应用程序在 Amazon EC2 实例上运行。该应用程序需要临时凭证来进行身份验证，以及授权应用程

序在 Amazon 中执行操作的权限。在设置该应用程序时，您必须将角色传递给 Amazon EC2，以便与提供这些凭证的实例一起使用。通过向角色附加 IAM policy，您在实例上运行的应用程序定义权限。该应用程序每次需要执行角色允许的操作时都会担任该角色。

要将角色（及其权限）传递至 Amazon 服务，用户必须具有传递角色至服务的权限。这有助于管理员确保仅批准的用户可配置具有能够授予权限的角色的服务。要允许用户将角色传递至 Amazon 服务，您必须向用户的 IAM 用户、角色或组授予 PassRole 权限。

注意

- 您不能使用 PassRole 权限传递跨账户角色。
- 您无法使用 ResourceTag/*key-name* 条件键限制根据附加的标签传递角色的权限。有关更多信息，请参阅[控制对 Amazon 资源的访问 \(p. 390\)](#)。

在创建服务相关角色时，您还必须拥有将该角色传递给服务的权限。在某些服务中执行操作时，该服务自动在您的账户中创建一个服务相关角色。例如，在首次创建自动扩缩组时，Amazon EC2 Auto Scaling 会为您创建 AWSServiceRoleForAutoScaling 服务相关角色。如果您在不具备 PassRole 权限的情况下尝试创建 Auto Scaling 组，则会收到错误。如果选择默认角色，则 iam:PassRole 权限可能不是必需的。要了解哪些服务支持服务相关角色，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)。要了解在哪些服务中执行操作时自动创建服务相关角色，请选择是链接并查看该服务的服务相关角色文档。

在任何使用角色分配服务权限的 API 操作中，用户都可以将角色 ARN 作为参数进行传递。该服务随后检查该用户是否拥有 iam:PassRole 权限。要限制用户只传递批准的角色，您可以使用 IAM policy 语句的 iam:PassRole 元素筛选 Resources 权限。

您可以在 JSON 策略中使用 Condition 元素来测试所有 Amazon 请求的请求上下文中所包含键的值。要了解有关在策略中使用条件键的更多信息，请参阅[IAM JSON 策略元素：Condition \(p. 948\)](#)。iam:PassedToService 条件键可用于指定可将角色传递到的服务的服务主体。要了解有关在策略中使用 iam:PassedToService 条件键的更多信息，请参阅[iam:PassedToService \(p. 1015\)](#)。

示例 1

假设您要授予用户在启动实例时能够将任意批准角色组传递至 Amazon EC2 服务的能力。您需要三个部分：

- IAM 权限策略附加到确定角色可执行哪些任务的角色。将权限范围限定为仅角色必须执行的操作，以及角色进行这些操作所需的资源。您可以使用 Amazon 托管的或客户创建的 IAM 权限策略。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [ "A list of the permissions the role is allowed to use" ],
    "Resource": [ "A list of the resources the role is allowed to access" ]
  }
}
```

- 允许服务担任角色的信任策略。例如，您可以将以下信任策略与具有 UpdateAssumeRolePolicy 操作的角色进行附加。该信任策略允许 Amazon EC2 使用角色和附加在角色上的权限。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "TrustPolicyStatementThatAllowsEC2ServiceToAssumeTheAttachedRole",
    "Effect": "Allow",
    "Principal": { "Service": "ec2.amazonaws.com" },
    "Action": "sts:AssumeRole"
  }
}
```

- 附加至 IAM 用户的 IAM permissions policy (权限策略)，允许该用户仅传递已获批准的那些策略。您通常将 iam:GetRole 添加至 iam:PassRole，使用户能够获取准备进行传递的角色的详细信息。在此示例中，用户只能传递位于指定账户中并且名称以 EC2-roles-for-XYZ- 开头的角色：

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "iam:GetRole",
      "iam:PassRole"
    ],
    "Resource": "arn:aws:iam::account-id:role/EC2-roles-for-XYZ-*"
  }]
}
```

现在用户可以启动具有所分配角色的 Amazon EC2 实例。实例上运行的应用程序可以通过实例配置文件元数据访问角色的临时许可。附加到角色的许可策略确定实例可以执行的任务。

示例 2

Amazon Relational Database Service (Amazon RDS) 支持名为 Enhanced Monitoring (增强监控) 的功能。此功能使 Amazon RDS 能够使用代理监控数据库实例。它还允许 Amazon RDS 将指标记录到 Amazon CloudWatch Logs 中。要启用此功能，您必须创建一个服务角色，以便为 Amazon RDS 提供监控指标和将指标写入日志的权限。

为 Amazon RDS 增强监控创建角色

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 选择 Role，然后选择 Create role。
3. 选择 Amazon Service (Amazon 服务) 角色类型，然后对于 Use cases for other Amazon Web Services (适用于其他 Amazon Web Services 的使用案例)，选择 RDS 服务。选择 RDS - Enhanced Monitoring (RDS - 增强监控)，然后选择 Next (下一步)。
4. 选择 AmazonRDSEnhancedMonitoringRole 权限策略。
5. 选择 Next (下一步)。
6. 对于 Role name (角色名称)，键入有助于标识此角色作用的角色名称。角色名称在您的 Amazon 账户内必须是唯一的。名称不区分大小写。例如，您无法同时创建名为 **PRODROLE** 和 **prodrole** 的角色。由于多个单位可能引用该角色，角色创建完毕后无法编辑角色名称。
7. (可选) 对于 Description (描述)，输入新角色的描述。
8. (可选) 通过以键值对的形式附加标签来向用户添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 [标记 IAM 资源 \(p. 276\)](#)。
9. 检查角色，然后选择 Create role。

角色自动获得授予 monitoring.rds.amazonaws.com 服务担任角色权限的信任策略。在此之后，Amazon RDS 可执行 AmazonRDSEnhancedMonitoringRole 策略允许的所有操作。

您希望使用增强监控的用户需要包括允许用户列出 RDS 角色的语句和允许用户传递角色的语句的策略，如下所示。使用您的账号，然后使用您在步骤 6 中提供的名称替换角色名称。

```
{
  "Sid": "PolicyStatementToAllowUserToListRoles",
  "Effect": "Allow",
  "Action": ["iam:ListRoles"],
  "Resource": ""
}
```

```
    },  
    {  
      "Sid": "PolicyStatementToAllowUserToPassOneSpecificRole",  
      "Effect": "Allow",  
      "Action": [ "iam:PassRole" ],  
      "Resource": "arn:aws:iam::account-id:role/RDS-Monitoring-Role"  
    }  
  ]  
}
```

您可以将该语句与另一策略中的语句进行合并，或将此语句放在其自身策略中。如要指定用户可传递以 RDS- 开始的任何角色，您可以在资源 ARN 中使用通配符替换角色名称，如下所示。

```
"Resource": "arn:aws:iam::account-id:role/RDS-*"
```

切换到角色（控制台）

角色 指定可用于访问所需的 Amazon 资源的一组权限。在这种意义上，它类似于 [Amazon Identity and Access Management 中的用户](#) (IAM)。作为用户登录时，您会获取一组特定权限。但是，您没有登录到角色，不过一旦登录，您就可以切换为角色。这会临时搁置原始用户权限，而向您提供分配给角色的权限。角色可以在您自己的账户中或任何其他 Amazon 账户中。有关角色、其权益以及如何创建角色的更多信息，请参阅 [IAM 角色 \(p. 158\)](#) 和 [创建 IAM 角色 \(p. 215\)](#)。

Important

您的 IAM 用户权限和切换为的角色的权限不会累积。一次只有一组权限处于活动状态。切换到一个角色后，您将临时放弃用户权限并使用分配给该角色的权限。退出该角色后，您的用户权限将自动恢复。

当您在 Amazon Web Services Management Console 中切换角色时，控制台总是使用您的原始凭证对切换操作进行授权。无论您作为 IAM 用户、SAML 联合角色还是 Web 联合身份角色登录，上述情形均适用。例如，如果您切换到角色 A，则 IAM 使用您的原始用户或联合角色凭证确定是否允许您担任角色 A。如果随后在使用角色 A 时尝试切换到角色 B，Amazon 仍会使用您的原始用户或联合角色凭证对切换进行授权，而不是使用角色 A 的凭证。

有关在控制台中切换角色的需知信息

此部分提供有关如何使用 IAM 控制台切换到某个角色的更多信息。

备注:

- 如果您以 Amazon Web Services 账户 根用户身份登录，则无法切换角色。以 IAM 用户、SAML 联合角色或 web 身份联合角色登录时，您可以切换角色。
- 您无法将 Amazon Web Services Management Console 中的角色切换到需要 [ExternalId \(p. 164\)](#) 值的角色。您只能通过调用支持 ExternalId 参数的 [AssumeRole](#) API 来切换到此类角色。
- 如果管理员为您提供了链接，请选择该链接，然后跳到以下过程中的步骤 [Step 5](#)。您可以通过该链接转到相应的网页，并为您填写账户 ID（或别名）和角色名称。
- 您可以手动构造链接，然后跳到以下过程中的步骤 [Step 5](#)。要构造您的链接，请使用以下格式：

```
https://signin.amazonaws.cn/switchrole?  
account=account_id_number&roleName=role_name&displayName=text_to_display
```

在其中替换以下文本：

- account_id_number** - 管理员向您提供的 12 位的账户标识符。或者，您的管理员可能创建账户别名，使得 URL 包含您的账户名称而不是账户 ID。有关更多信息，请参阅 [Amazon 账户 ID 及其别名 \(p. 63\)](#)。

- **role_name** - 要代入的角色的名称。您可以从角色 ARN 的结尾获取此名称。例如，从以下角色 ARN 提供 TestRole 角色名称：namearn:aws:iam::403299380220:role/TestRole。
- (可选) **text_to_display** - 您希望此角色处于活动时显示在导航栏中以替代用户名的文本。
- 您可以通过以下过程使用管理员提供的信息来手动切换角色。

预设情况下，切换角色时，Amazon Web Services Management Console 会话将持续 1 小时。预设情况下，IAM 用户会话为 12 小时。在控制台内切换角色的 IAM 用户被授予最大会话持续时间或 IAM 用户会话中的剩余时间（以较少者为准）。例如，假定角色的最长会话持续时间为 10 小时。当 IAM 用户决定切换到该角色时，已登录控制台 8 小时。IAM 用户会话还剩 4 小时，因此允许的角色会话持续时间为 4 小时。下表显示了如何在控制台中切换角色时确定 IAM 用户的会话持续时间。

IAM 用户控制台角色会话持续时间

IAM 用户会话剩余时间...	角色会话持续时间...		
小于角色最长会话持续时间	等于 IAM 用户会话中的剩余时间		
大于角色最长会话持续时间	等于最长会话持续时间值		
等于角色最长会话持续时间	等于最长会话持续时间值（近似值）		

Note

某些 Amazon 服务控制台可以在角色会话过期时自动续订角色会话，而无需您执行任何操作。有些可能会提示您重新加载浏览器页面以重新验证您的会话。

要排除您在担任角色时可能遇到的常见问题，请参阅 [我无法代入角色 \(p. 884\)](#)。

切换为角色（控制台）

1. 以 IAM 用户登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在 IAM 控制台中，在右上角的导航栏上选择您的用户名。它通常类似于：**username@account_ID_number_or_alias**。
3. 选择 Switch Role。如果这是您首次选择该选项，则会显示一个包含更多信息的页面。在阅读该信息后，请选择切换角色。如果清除您的浏览器 Cookie，则此页面会重新再出现。
4. 在 Switch Role 页面上，键入账户 ID 号或账户别名以及管理员提供的角色的名称。

Note

如果您的管理员创建了包含路径的角色（如 division_abc/subdivision_efg/roleToDoX），则必须在 Role 框中键入完整路径和名称。如果您仅键入角色名称，或组合的 Path 和 RoleName 超过 64 个字符，角色切换将失败。这是存储角色名称的浏览器 Cookie 的限制。如果发生这种情况，请与您的管理员联系，并要求他们减小路径和角色名称大小。

5. (可选) 选择 Display name（显示名称）。键入您希望此角色处于活动时显示在导航栏上以替代用户名的文本。系统会基于账户和角色信息提供建议名称，但是您可以将它更改为对您有意义的任何名称。您还可以选择用于突出显示名称的样式。名称和颜色可帮助提醒您此角色处于活动状态的时间，这将更改您的权限。例如，对于向您提供对测试环境的访问权限的角色，您可以将 Display name（显示名称）指定为 **Test**，并选择绿色的 Color（颜色）。对于向您授予对生产环境的访问权限的角色，您可以将 Display name（显示名称）指定为 **Production**，并选择红色作为 Color（颜色）。
6. 选择 Switch Role。显示名称和颜色会在导航栏上替换您的用户名，您可以开始使用角色向您授予的权限。

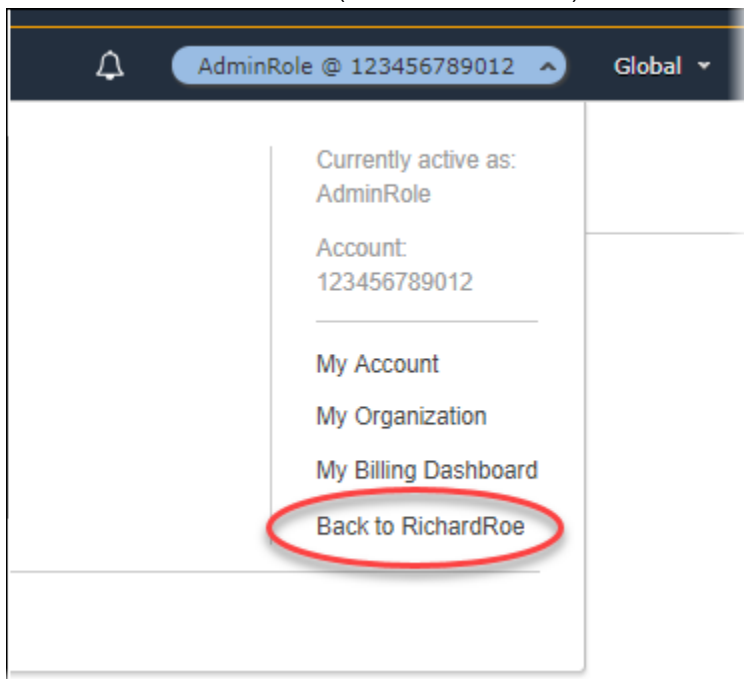
提示

您使用的最后几个角色将显示在菜单上。下次需要切换到其中的一个角色时，您可以直接选择所需的角色。只有在菜单上未显示角色时，您才需要手动键入账户和角色信息。

停止使用角色（控制台）

1. 在 IAM 控制台中，在右上角的导航栏中选择角色的 Display Name（显示名称）。它通常类似于：**rolename@account_ID_number_or_alias**。
2. 选择 Back to **username**（返回 username）。角色和其权限会停用，与您的 IAM 用户和组关联的权限会自动恢复。

例如，假设您使用用户名 123456789012 登录账号 RichardRoe。在使用 AdminRole 角色后，您要停止使用该角色并返回到您的原始权限。要停止使用某个角色，请选择 AdminRole @ 123456789012，然后选择 Back to RichardRoe（返回至 RichardRoe）。



切换到 IAM 角色 (Amazon CLI)

角色 指定可用于访问所需的 Amazon 资源的一组权限。在这种意义上，它类似于 [Amazon Identity and Access Management 中的用户 \(IAM\)](#)。作为用户登录时，您会获取一组特定权限。不过，您不会登录到角色，但在以用户身份登录后，您可以切换到角色。这会临时搁置原始用户权限，而向您提供分配给角色的权限。角色可以在您自己的账户中或任何其他 Amazon 账户中。有关角色、其权益以及如何创建和配置角色的更多信息，请参阅 [IAM 角色 \(p. 158\)](#) 和 [创建 IAM 角色 \(p. 215\)](#)。要了解在担任角色时使用的各种方法，请参阅 [使用 IAM 角色 \(p. 236\)](#)。

Important

不会累积您的 IAM 用户和担任的任何角色的权限。一次只有一组权限处于活动状态。在担任某个角色时，您将临时放弃以前的用户或角色权限并使用为该角色分配的权限。退出该角色后，您的用户权限将自动恢复。

在以 IAM 用户身份登录后，您可以使用角色来运行 Amazon CLI 命令。在以已使用角色的 [externally authenticated user \(p. 170\)](#)（外部验证的用户）（[SAML \(p. 176\)](#) 或 [OIDC \(p. 171\)](#)）身份登录后，

您也可以使用角色来运行 Amazon CLI 命令。此外，您还可以使用角色从通过实例配置文件附加到角色的 Amazon EC2 实例中运行 Amazon CLI 命令。在以 Amazon Web Services 账户 根用户身份登录时，您无法担任角色。

角色链 (p. 159) — 您还可以使用角色链，这将使用来自一个角色的权限访问另一个角色。在控制台中切换角色时，您没有使用角色链。您正在使用的是原始用户的权限。

默认情况下，您的角色会话持续 1 小时。在使用 `assume-role*` CLI 操作担任该角色时，您可以为 `duration-seconds` 参数指定一个值。该值的范围在 900 秒 (15 分钟) 到角色的最大会话持续时间设置之间。要了解如何查看您的角色的最大值，请参阅[查看角色的最大会话持续时间设置](#) (p. 238)。

如果使用角色链，您的会话持续时间限制为最多 1 小时。如果您随后使用 `duration-seconds` 参数提供大于 1 小时的值，操作将失败。

示例方案：切换到生产角色

假设您是一名在开发环境中工作的 IAM 用户。在此场景中，您可能需要使用 [Amazon CLI](#) 通过命令行来使用生产环境。您已经有一组可用的访问密钥凭证。这可能是分配给您的标准 IAM 用户的访问密钥对。或者，如果您以联合身份用户身份登录，则它可能是最初为您分配的角色访问密钥对。如果您的当前权限授予您担任特定 IAM 角色的能力，则可以在 Amazon CLI 配置文件的“配置文件”中标识该角色。然后，将使用指定 IAM 角色而非原始身份的权限运行该命令。请注意，通过 Amazon CLI 命令指定该配置文件时，您使用的是新角色。在这种情况下，您无法同时使用开发账户中的原始权限。原因是，一次仅一组权限能够生效。

Note

为了安全起见，管理员可以[查看 Amazon CloudTrail 日志](#) (p. 343) 以了解已在 Amazon 中执行操作的人员。您的管理员可能会要求您在代入角色时指定源身份或角色会话名称。有关更多信息，请参阅 [sts:SourceIdentity](#) (p. 1024) 和 [sts:RoleSessionName](#) (p. 1023)。

切换到生产角色 (Amazon CLI)

1. 如果您从未使用过 Amazon CLI，则您必须先配置默认的 CLI 配置文件。打开命令提示符并将您的 Amazon CLI 安装设置为使用来自 IAM 用户或联合角色的访问密钥。有关更多信息，请参阅《Amazon Command Line Interface 用户指南》中的[配置 Amazon Command Line Interface](#)。

运行 `aws configure` 命令，如下所述：

```
aws configure
```

当系统提示时，请提供以下信息：

```
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
Default region name [None]: us-east-2
Default output format [None]: json
```

2. 在 Unix 或 Linux 上的 `.aws/config` 文件或 Windows 上的 `C:\Users\USERNAME\.aws\config` 文件中，为角色创建新的配置文件。以下示例创建一个名为 `prodaccess` 的配置文件，它切换到 123456789012 账户中的 `ProductionAccessRole` 角色。您从创建该角色的账户管理员处获取角色 ARN。在调用此配置文件时，Amazon CLI 使用 `source_profile` 的凭证请求该角色的凭证。因此，引用为 `source_profile` 的身份必须具有 `role_arn` 中指定的角色的 `sts:AssumeRole` 权限。

```
[profile prodaccess]
role_arn = arn:aws:iam::123456789012:role/ProductionAccessRole
source_profile = default
```

3. 在创建新的配置文件后，将使用附加到 IAM 角色 `ProductionAccessRole` (而不是默认用户) 的权限运行指定 `--profile prodaccess` 参数的任何 Amazon CLI 命令。


```
aws iam list-users --profile prodaccess
```

如果分配给 `ProductionAccessRole` 的权限允许列出当前 Amazon 账户中的用户，则此命令有效。

4. 要返回到原始凭证授予的权限，请运行不带 `--profile` 参数的命令。Amazon CLI 将恢复使用您在 [Step 1](#) 中配置的默认配置文件中的凭证。

有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的 [代入角色](#)。

示例场景：允许实例配置文件角色切换到另一个账户中的角色

假设您使用两个 Amazon 账户，并希望允许 Amazon EC2 实例上运行的应用程序在两个账户中运行 [Amazon CLI](#) 命令。假设 EC2 实例位于 111111111111 账户中。该实例包含 `abcd` 实例配置文件角色，以允许应用程序对同一 111111111111 账户中的 `my-bucket-1` 存储桶执行只读 Amazon S3 任务。不过，还必须允许应用程序担任 `efgh` 跨账户角色以在账户 222222222222 中执行任务。为此，`abcd` EC2 实例配置文件角色必须具有以下权限策略：

账户 111111111111 **abcd** 角色权限策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccountLevelS3Actions",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetAccountPublicAccessBlock",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Sid": "AllowListAndReadS3ActionOnMyBucket",
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket-1/*",
        "arn:aws:s3:::my-bucket-1"
      ]
    },
    {
      "Sid": "AllowIPToAssumeCrossAccountRole",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::222222222222:role/efgh"
    }
  ]
}
```

假设 `efgh` 跨账户角色允许对同一 222222222222 账户中的 `my-bucket-2` 存储桶执行只读 Amazon S3 任务。为此，`efgh` 跨账户角色必须具有以下权限策略：

账户 222222222222 **efgh** 角色权限策略

```
{
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccountLevelS3Actions",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetAccountPublicAccessBlock",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Sid": "AllowListAndReadS3ActionOnMyBucket",
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket-2/*",
        "arn:aws:s3:::my-bucket-2"
      ]
    }
  ]
}
```

efgh 角色必须允许 abcd 实例配置文件角色担任该角色。为此，efgh 角色必须具有以下信任策略：

账户 222222222222 **efgh** 角色信任策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "efghTrustPolicy",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": { "AWS": "arn:aws:iam::111111111111:role/abcd" }
    }
  ]
}
```

要在 222222222222 账户中运行 Amazon CLI 命令，您必须更新 CLI 配置文件。在 Amazon CLI 配置文件中将 efgh 角色指定为“配置文件”，并将 abcd EC2 实例配置文件角色指定为“凭证源”。然后，将使用 efgh 角色的权限运行 CLI 命令，而不是使用原始 abcd 角色。

Note

出于安全原因，您可以使用 Amazon CloudTrail 审核角色在账户中的使用。要在 CloudTrail 日志中由不同主体使用角色时区分角色会话，可以借助角色会话名称。在 Amazon CLI 按照本主题所述代表用户担任某个角色时，将自动创建一个 Amazon-CLI-session-**nnnnnnnn** 形式的角色会话名称。其中，**nnnnnnnn** 是一个表示 [Unix 纪元时间](#) (自 1970 年 1 月 1 日午夜 UTC 算起的秒数) 的整数。有关更多信息，请参阅 Amazon CloudTrail 用户指南中的 [CloudTrail 事件引用](#)。

允许 EC2 实例配置文件角色切换到跨账户角色 (Amazon CLI)

1. 您不必配置默认 CLI 配置文件。相反，您可以从 EC2 实例配置文件元数据中加载凭证。在 .aws/config 文件中为角色创建新的配置文件。以下示例创建一个 instancecrossaccount 配置文件，它切换到 222222222222 账户中的 **efgh** 角色。在调用该配置文件时，Amazon CLI 使用 EC2 实例配置

文件元数据的凭证请求该角色的凭证。因此，EC2 实例配置文件角色必须具有 `role_arn` 中指定的角色的 `sts:AssumeRole` 权限。

```
[profile instancecrossaccount]
role_arn = arn:aws:iam::222222222222:role/efgh
credential_source = Ec2InstanceMetadata
```

2. 在创建新的配置文件后，将使用附加到 222222222222 账户中的 `efgh` 角色的权限运行指定 `--profile instancecrossaccount` 参数的任何 Amazon CLI 命令。

```
aws s3 ls my-bucket-2 --profile instancecrossaccount
```

如果分配给 `efgh` 角色的权限允许列出当前 Amazon 账户中的用户，则该命令有效。

3. 要恢复为 111111111111 账户中的原始 EC2 实例配置文件权限，请不要使用 `--profile` 参数运行 CLI 命令。

有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的[代入角色](#)。

切换到 IAM 角色 (Tools for Windows PowerShell)

角色指定可用于访问所需的 Amazon 资源的一组权限。在这种意义上，它类似于 [Amazon Identity and Access Management 中的用户](#) (IAM)。作为用户登录时，您会获取一组特定权限。但是，您没有登录到角色，不过一旦登录，您就可以切换为角色。这会临时搁置原始用户权限，而向您提供分配给角色的权限。角色可以在您自己的账户中或任何其他 Amazon 账户中。有关角色、其权益以及如何创建和配置角色的更多信息，请参阅[IAM 角色 \(p. 158\)](#)和[创建 IAM 角色 \(p. 215\)](#)。

Important

您的 IAM 用户权限和切换为的角色的权限不会累积。一次只有一组权限处于活动状态。切换到一个角色后，您将临时放弃用户权限并使用分配给该角色的权限。退出该角色后，您的用户权限将自动恢复。

本部分介绍如何在通过 Amazon Tools for Windows PowerShell 在命令行上工作时切换角色。

假设您在开发环境中有一个账户，并且您有时需要使用 [Tools for Windows PowerShell](#) 在命令行上操作生产环境。您已经有一组可用的访问密钥凭证。这些凭证可以是分配给您的标准 IAM 用户的访问密钥对。或者，如果您以联合身份用户身份登录，则其可以是最初分配给您的角色的访问密钥对。您可以使用这些凭证运行 `Use-STSRole` cmdlet，这会将新角色的 ARN 作为参数传递。该命令将返回请求的角色的临时安全凭证。然后可利用该角色的权限在后续 PowerShell 命令中使用那些凭证来使用访问生产中的资源。在使用角色时，您不能使用开发账户中的用户权限，因为一次只有一组权限有效。

Note

为了安全起见，管理员可以[查看 Amazon CloudTrail 日志 \(p. 343\)](#)以了解已在 Amazon 中执行操作的人员。您的管理员可能会要求您在代入角色时指定源身份或角色会话名称。有关更多信息，请参阅 `sts:SourceIdentity` (p. 1024) 和 `sts:RoleSessionName` (p. 1023)。

请注意，所有访问密钥和令牌都只是示例，不能原样照用。请用您的实际环境的适当值替换。

要切换到某个角色 (Tools for Windows PowerShell)

1. 打开 PowerShell 命令提示符并将您的默认配置文件配置为使用来自您的当前 IAM 用户或联合角色的访问密钥。如果您以前使用过 Tools for Windows PowerShell，则可能已完成此设置。请注意，您只能在以 IAM 用户身份（而不是 Amazon Web Services 账户根用户身份）登录时切换角色。

```
PS C:\> Set-AWSCredentials -AccessKey AKIAIOSFODNN7EXAMPLE -SecretKey wJalrXUtnFEMI/
K7MDENG/bPxrFiCYEXAMPLEKEY -StoreAs MyMainUserProfile
```

```
PS C:\> Initialize-AWSDefaults -ProfileName MyMainUserProfile -Region us-east-2
```

有关更多信息，请参阅 Amazon Tools for Windows PowerShell 用户指南中的[使用 Amazon 凭证](#)。

- 要检索新角色的凭证，请运行以下命令切换到 123456789012 账户中的 **RoleName** 角色。您从创建该角色的账户管理员处获取角色 ARN。该命令还需要您提供会话名称。您可以选择该名称的任何文本。以下命令请求凭证，然后从返回的结果对象中捕获 Credentials 属性对象并将其存储在 \$Creds 变量中。

```
PS C:\> $Creds = (Use-STSRole -RoleArn "arn:aws:iam::123456789012:role/RoleName" -  
RoleSessionName "MyRoleSessionName").Credentials
```

\$Creds 是一个对象，现在包含您在后续步骤中所需的 AccessKeyId、SecretAccessKey 和 SessionToken 元素。以下示例命令说明典型值：

```
PS C:\> $Creds.AccessKeyId  
AKIAIOSFODNN7EXAMPLE  
  
PS C:\> $Creds.SecretAccessKey  
wJalrXUtnFEMI/K7MDENG/bPxrFcIYEXAMPLEKEY  
  
PS C:\> $Creds.SessionToken  
AQoDYXdzEGcaEXAMPLE2gsYULo+Im5ZEXAMPLEeYjs1M2FUIgIJx9tQqNMBEXAMPLECvSryh0FW7jEXAMPLEW  
+vE/7s1HRp  
XviG7b+qYf4nD00EXAMPLEmJ4wxS04L/uZEXAMPLECihzFB5lTYLto9dyBgSDyEXAMPLE9/  
g7QRUhZp4bqbEXAMPLENwGPY  
Oj59pFA4lNKCikVgkREXAMPLEjlxQ7y52gekeVEXAMPLEDiB9ST3UuysgsKdEXAMPLE1TVastU1A0SKFEXAMPLEIywCC/  
C  
s8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP+4eZScEXAMPLEsnf87eNhyDHq6ikBQ==  
  
PS C:\> $Creds.Expiration  
Thursday, June 18, 2018 2:28:31 PM
```

- 要在任何后续命令中使用这些凭证，应使用 -Credentials 参数来包含它们。例如，以下命令使用来自角色的凭证，并只有在角色被授予 iam:ListRoles 权限时，命令才会起作用并可以因此运行 Get-IAMRoles cmdlet：

```
PS C:\> get-iamroles -Credential $Creds
```

- 要返回到您的原始凭证，只需停止使用 -Credentials \$Creds 参数并允许 PowerShell 恢复为在默认配置文件中存储的凭证。

切换到 IAM 角色 (Amazon API)

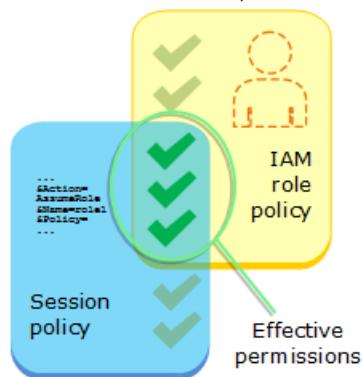
角色 指定可用于访问 Amazon 资源的一组权限。在这种意义上，它类似于 [IAM 用户](#)。主体（用户或应用程序）将担任角色以获得临时权限，从而执行所需任务并与 Amazon 资源交互。角色可以在您自己的账户中或任何其他 Amazon 账户中。有关角色、其权益以及如何创建和配置角色的更多信息，请参阅[IAM 角色 \(p. 158\)](#)和[创建 IAM 角色 \(p. 215\)](#)。要了解在担任角色时使用的各种方法，请参阅[使用 IAM 角色 \(p. 236\)](#)。

Important

不会累积您的 IAM 用户和担任的任何角色的权限。一次只有一组权限处于活动状态。在担任某个角色时，您将临时放弃以前的用户或角色权限并使用为该角色分配的权限。退出该角色后，您的原始权限将自动恢复。

要担任角色，应用程序需调用 Amazon STS [AssumeRole](#) API 操作并传递角色的 ARN 以供使用。该操作将使用临时凭证创建一个新会话。此会话与用于该角色的基于身份的策略具有相同的权限。

在调用 `AssumeRole` 时，您可以选择传递内联或托管会话策略 (p. 357)。会话策略是高级策略，在以编程方式为用户或联合身份用户创建临时凭证会话时，这些策略将作为参数进行传递。您可以使用 `PolicyArns` 参数指定最多 10 个托管会话策略。生成的会话的权限是实体的基于身份的策略与会话策略的交集。如果需要为其他人提供角色的临时凭证，会话策略是非常有用的。他们可以在后续的 Amazon API 调用中使用角色的临时凭证来访问拥有该角色的账户中的资源。您使用会话策略授予的权限不能超过基于身份的策略允许的权限。要了解有关 Amazon 如何确定角色的有效权限的更多信息，请参阅策略评估逻辑 (p. 972)。



在以 IAM 用户或已使用角色的 [externally authenticated user](#) (p. 170) (外部验证的用户 (SAML (p. 176) 或 OIDC (p. 171)) 身份登录后，您可以调用 `AssumeRole`。您还可以使用角色链 (p. 159)，它使用一个角色担任另一个角色。在以 Amazon Web Services 账户根用户身份登录时，您无法担任角色。

默认情况下，您的角色会话持续 1 小时。在使用 Amazon STS `AssumeRole*` API 操作担任该角色时，您可以为 `DurationSeconds` 参数指定一个值。该值的范围在 900 秒 (15 分钟) 到角色的最大会话持续时间设置之间。要了解如何查看您的角色的最大值，请参阅查看角色的最大会话持续时间设置 (p. 238)。

如果使用角色链，您的会话限制为最多 1 小时。如果您随后使用 `DurationSeconds` 参数提供大于 1 小时的值，操作将失败。

Note

为了安全起见，管理员可以查看 [Amazon CloudTrail 日志](#) (p. 343) 以了解已在 Amazon 中执行操作的人员。您的管理员可能会要求您在代入角色时指定源身份或角色会话名称。有关更多信息，请参阅 [sts:SourceIdentity](#) (p. 1024) 和 [sts:RoleSessionName](#) (p. 1023)。

以下 Python 示例使用 Amazon 的 Boto3 接口 ([Amazon SDK for Python \(Boto\) V3](#))，它介绍了如何调用 `AssumeRole`。它还介绍了如何使用 `AssumeRole` 返回的临时安全凭证列出拥有该角色的账户中的所有 Amazon S3 存储桶。

```
import boto3

# The calls to Amazon STS AssumeRole must be signed with the access key ID
# and secret access key of an existing IAM user or by using existing temporary
# credentials such as those from another role. (You cannot call AssumeRole
# with the access key for the root account.) The credentials can be in
# environment variables or in a configuration file and will be discovered
# automatically by the boto3.client() function. For more information, see the
# Python SDK documentation:
# http://boto3.readthedocs.io/en/latest/reference/services/sts.html#client

# create an STS client object that represents a live connection to the
# STS service
sts_client = boto3.client('sts')

# Call the assume_role method of the STSConnection object and pass the role
# ARN and a role session name.
assumed_role_object=sts_client.assume_role(
```

```
RoleArn="arn:aws:iam::account-of-role-to-assume:role/name-of-role",
RoleSessionName="AssumeRoleSession1"
)

# From the response that contains the assumed role, get the temporary
# credentials that can be used to make subsequent API calls
credentials=assumed_role_object['Credentials']

# Use the temporary credentials that AssumeRole returns to make a
# connection to Amazon S3
s3_resource=boto3.resource(
    's3',
    aws_access_key_id=credentials['AccessKeyId'],
    aws_secret_access_key=credentials['SecretAccessKey'],
    aws_session_token=credentials['SessionToken'],
)

# Use the Amazon S3 resource object that is now configured with the
# credentials to access your S3 buckets.
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限

在 Amazon EC2 实例上运行的应用程序必须将 Amazon 凭证包含在其 Amazon API 请求中。您可以让开发人员将 Amazon 凭证直接存储在 Amazon EC2 实例中，并允许该实例中的应用程序使用这些凭证。但开发人员随后必须管理凭证，确保他们能安全地将凭证传递给每个实例，并在需要轮换凭证时更新每个 Amazon EC2 实例。这需要进行大量的额外工作。

相反，您可以且应使用 IAM 角色管理在 Amazon EC2 实例上运行的应用程序的临时凭证。在使用角色时，您不需要将长期凭证（如用户名和密码或访问密钥）分配给 Amazon EC2 实例。相反，角色可提供临时权限供应用程序在调用其他 Amazon 资源时使用。当您启动 Amazon EC2 实例时，可指定要与实例关联的 IAM 角色。然后，实例上运行的应用程序可使用角色提供的临时凭证对 API 请求进行签名。

若要使用角色向 Amazon EC2 实例上运行的应用程序授予权限，需要进行一点额外配置。Amazon EC2 实例上运行的应用程序由虚拟化操作系统从 Amazon 中提取。因为存在这一额外分离操作，所以需要执行一个附加步骤将 Amazon 角色及其关联权限分配给 Amazon EC2 实例，并使这些权限对其应用程序可用。此额外步骤是创建要附加到实例的[实例配置文件](#)。实例配置文件包含角色，并且可以为实例上运行的应用程序提供角色的临时凭证。然后，可以在应用程序的 API 调用中使用这些临时凭证访问资源，以及将访问限制为仅角色指定的那些资源。请注意，一次只能将一个角色分配给 Amazon EC2 实例，实例上的所有应用程序都共享相同的角色和权限。

以这种方式使用该角色拥有多种优势。因为角色凭证是临时的，并且会自动轮换，所以您不必管理凭证，也不必担心长期安全风险。此外，如果您对多个实例使用单个角色，则可以对角色进行更改，而此更改会自动传播到所有实例。

Note

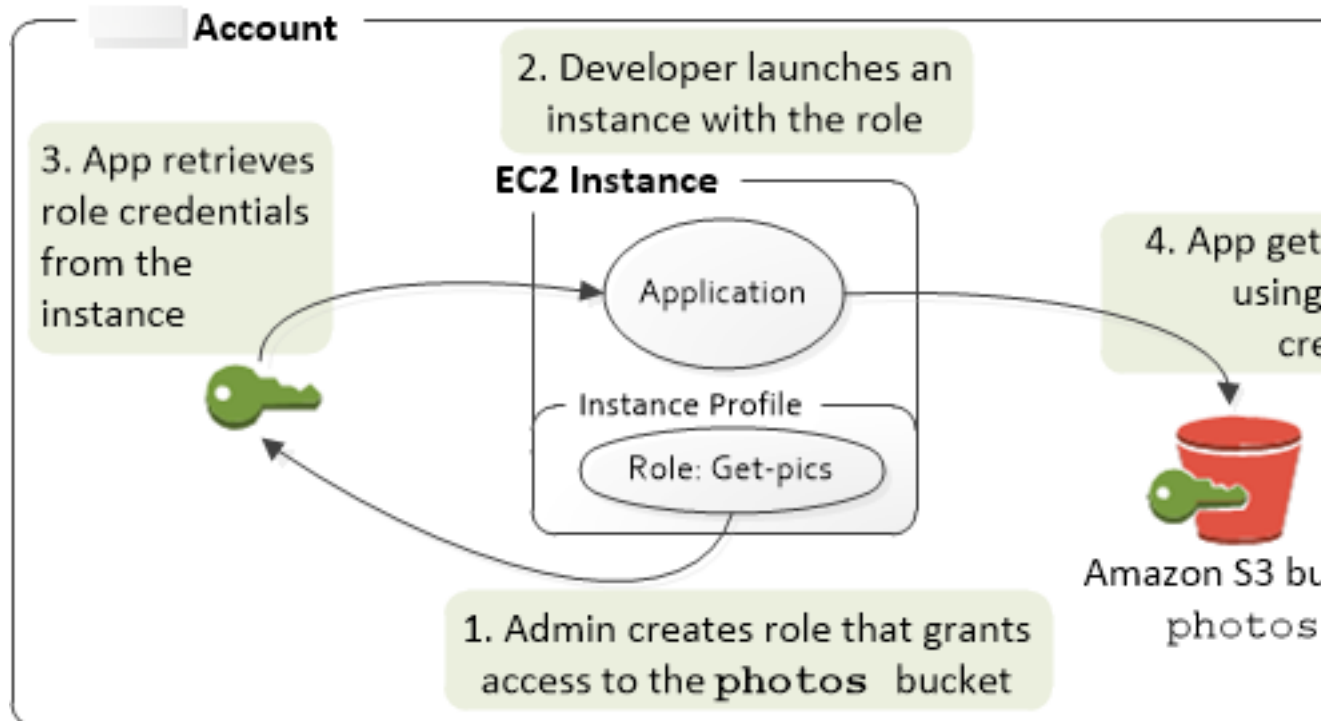
通常会在您启动 Amazon EC2 实例时向它分配角色，不过也可以向已在运行的 Amazon EC2 实例附加角色。要了解如何将角色附加到运行的实例，请参阅[适用于 Amazon EC2 的 IAM 角色](#)。

主题

- [Amazon EC2 实例的角色的工作方式？](#) (p. 253)
- [在 Amazon EC2 中使用角色所需的权限](#) (p. 254)
- [怎样入门？](#) (p. 257)
- [相关信息](#) (p. 257)
- [使用实例配置文件](#) (p. 258)

Amazon EC2 实例的角色的工作方式？

在下图中，一名开发人员在 Amazon EC2 实例上运行一个应用程序，该应用程序要求访问名为 `photos` 的 S3 存储桶。管理员创建 `Get-pics` 服务角色并将该角色附加到 Amazon EC2 实例。该角色包括一个权限策略，该策略授予对指定 S3 存储桶的只读访问权限。它还包括一个信任策略，该策略允许 Amazon EC2 实例担任该角色并检索临时凭证。在该实例上运行应用程序时，应用程序可以使用该角色的临时凭证访问照片存储桶。管理员不必向开发人员授予访问照片存储桶的权限，开发人员完全不必共享或管理凭证。



1. 管理员使用 IAM 创建 **Get-pics** 角色。在角色的信任策略中，管理员指定仅 Amazon EC2 实例能够担任该角色。在角色的权限策略中，管理员为 `photos` 存储桶指定只读权限。
2. 开发人员启动 Amazon EC2 实例，并向该实例分配 `Get-pics` 角色。

Note

如果您使用 IAM 控制台，则会为您管理实例配置文件，该配置文件大部分对您透明的。但是，如果您使用 Amazon CLI 或 API 创建和管理角色和 Amazon EC2 实例，则必须创建实例配置文件并采用单独步骤将角色分配给它。随后在启动实例时，您必须指定实例配置文件名称而不是角色名称。

3. 应用程序在运行时从 Amazon EC2 [实例元数据](#) 获取临时安全凭证，如 [从实例元数据检索安全凭证](#) 中所述。这些是 [临时安全凭证](#) (p. 300)，用于表示角色，在有限时间段内有效。

使用某些 [Amazon 开发工具包](#)，开发人员可以使用提供程序以透明方式管理临时安全凭证。（各 Amazon 开发工具包的文档介绍该软件开发工具包支持的证书管理功能。）

应用程序也可以直接从 Amazon EC2 实例的实例元数据中获取临时凭证。凭证和相关值可从元数据的 `iam/security-credentials/role-name` 类别（在本例中为 `iam/security-credentials/Get-pics`）获得。如果应用程序从实例元数据获取凭证，则它可对凭证进行缓存。

4. 通过使用检索到的临时凭证，应用程序可以访问照片存储桶。由于附加到 **Get-pics** 角色的策略，应用程序具有只读权限。

实例上提供的临时安全凭证会在过期之前自动轮换，因此始终具有有效的凭证集。应用程序只需要确保在当前凭证过期之前从实例元数据获取新的凭证集。可以使用 Amazon 开发工具包管理凭证，这样应用程序

就不需要包含额外的逻辑来刷新凭证。例如，使用实例配置文件凭证提供程序实例化客户端。但是，如果应用程序从实例元数据获取临时安全凭证并对它们进行缓存，则它应在当前凭证集过期之前，每隔一小时(或至少每隔 15 分钟) 获取刷新的凭证集。在 `iam/security-credentials/role-name` 类别中返回的信息中包含过期时间。

在 Amazon EC2 中使用角色所需的权限

要启动具有角色的实例，开发人员必须有启动 Amazon EC2 实例的权限和传递 IAM 角色的权限。

以下示例策略可以实现使用 Amazon Web Services Management Console 发布引用角色的实例。策略包含通配符 (*) 以允许用户传递任意角色并执行所有列出的 Amazon EC2 操作。用户可以通过 `ListInstanceProfiles` 操作浏览 Amazon 账户中的所有角色。

Example 授权用户使用 Amazon EC2 控制台启动具有任何角色的实例的示例策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IamPassRole",
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "ec2.amazonaws.com"
        }
      }
    },
    {
      "Sid": "ListEc2AndListInstanceProfiles",
      "Effect": "Allow",
      "Action": [
        "iam:ListInstanceProfiles",
        "ec2:Describe*",
        "ec2:Search*",
        "ec2:Get*"
      ],
      "Resource": "*"
    }
  ]
}
```

限制哪些角色能够传递到 Amazon EC2 实例 (使用 PassRole)

您可以使用 `PassRole` 权限来限制用户启动实例时可以传递给 Amazon EC2 实例的角色。这有助于防止用户运行具有超出用户获授权限的应用程序 即防止用户获取提升的权限。例如，假设用户 Alice 仅有权启动 Amazon EC2 实例并使用 Amazon S3 存储桶，但是她传递给 Amazon EC2 实例的角色有权使用 IAM 和 Amazon DynamoDB。在这种情况下，Alice 或许能够启动实例，登录它，获取临时安全凭证，然后执行未向她授权的 IAM 或 DynamoDB 操作。

要限制用户可以传递给 Amazon EC2 实例的角色，您可创建一个允许执行 `PassRole` 操作的策略。然后，将策略附加到将启动 Amazon EC2 实例的用户 (或该用户所属的 IAM 组)。在策略的 `Resource` 元素中，列出允许用户传递给 Amazon EC2 实例的角色。用户启动一个实例并将角色与该实例关联时，Amazon EC2 会检查是否允许用户传递该角色。当然，您还应确保用户可传递的角色不包含用户不应拥有的权限。

Note

就像 `PassRole` 或 `RunInstances` 那样，`ListInstanceProfiles` 不是 API 操作。相反，它是只要有角色 ARN 作为参数传递给 API 时，Amazon 便会检查的权限 (或者控制台代表用户这么

做)。它帮助管理员控制哪些角色可以由哪些用户传递。在这种情况下，它可确保用户可以将特定角色附加到 Amazon EC2 实例。

Example 通过特定角色向用户授予启动 Amazon EC2 实例的权限的示例策略

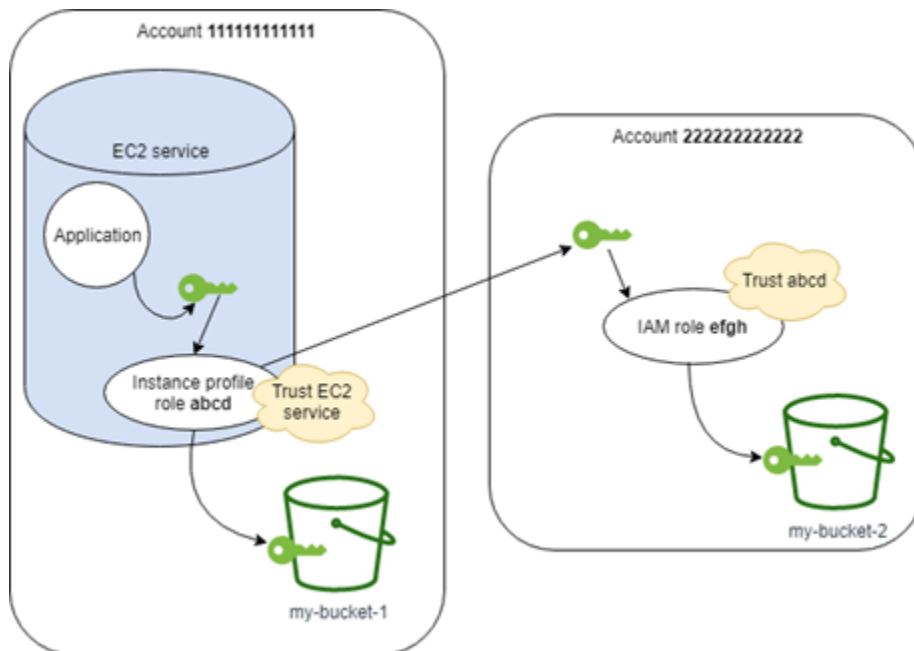
通过以下示例策略，用户可以使用 Amazon EC2 API，通过角色来启动实例。Resource 元素指定角色的 Amazon Resource Name (ARN)。通过指定 ARN，该策略授予用户仅传递 Get-pics 角色的权限。如果在启动实例时用户尝试指定其他角色，则该操作会失败。用户拥有运行任何实例的权限，无论他们是否传递角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:RunInstances",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::account-id:role/Get-pics"
    }
  ]
}
```

允许实例配置文件角色切换到另一个账户中的角色

您可以允许 Amazon EC2 实例上运行的应用程序在另一个账户中运行命令。为此，您必须允许第一个账户中的 Amazon EC2 实例角色切换到第二个账户中的角色。

假设您使用两个 Amazon 账户，并希望允许 Amazon EC2 实例上运行的应用程序在两个账户中运行 [Amazon CLI](#) 命令。假设 Amazon EC2 实例位于 111111111111 账户中。该实例包含 abcd 实例配置文件角色，以允许应用程序对同一 111111111111 账户中的 my-bucket-1 存储桶执行只读 Amazon S3 任务。不过，还必须允许应用程序担任 efgh 跨账户角色以访问 222222222222 账户中的 my-bucket-2 Amazon S3 存储桶。



abcd Amazon EC2 实例配置文件角色必须具有以下权限策略，以允许应用程序访问 my-bucket-1 Amazon S3 存储桶：

账户 111111111111 **abcd** 角色权限策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccountLevelS3Actions",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetAccountPublicAccessBlock",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "arn:aws:s3::*"
    },
    {
      "Sid": "AllowListAndReadS3ActionOnMyBucket",
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": [
        "arn:aws:s3::my-bucket-1/*",
        "arn:aws:s3::my-bucket-1"
      ]
    },
    {
      "Sid": "AllowIPToAssumeCrossAccountRole",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::222222222222:role/efgh"
    }
  ]
}
```

abcd 角色必须信任 Amazon EC2 服务以担任该角色。为此，abcd 角色必须具有以下信任策略：

账户 111111111111 **abcd** 角色信任策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "abcdTrustPolicy",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {"Service": "ec2.amazonaws.com"}
    }
  ]
}
```

假设 efgh 跨账户角色允许对同一 222222222222 账户中的 my-bucket-2 存储桶执行只读 Amazon S3 任务。为此，efgh 跨账户角色必须具有以下权限策略：

账户 222222222222 **efgh** 角色权限策略

```
{
  "Version": "2012-10-17",
```

```
"Statement": [
  {
    "Sid": "AllowAccountLevelS3Actions",
    "Effect": "Allow",
    "Action": [
      "s3:GetBucketLocation",
      "s3:GetAccountPublicAccessBlock",
      "s3:ListAccessPoints",
      "s3:ListAllMyBuckets"
    ],
    "Resource": "arn:aws:s3:::*"
  },
  {
    "Sid": "AllowListAndReadS3ActionOnMyBucket",
    "Effect": "Allow",
    "Action": [
      "s3:Get*",
      "s3:List*"
    ],
    "Resource": [
      "arn:aws:s3:::my-bucket-2/*",
      "arn:aws:s3:::my-bucket-2"
    ]
  }
]
```

efgh 角色必须信任 abcd 实例配置文件角色以担任该角色。为此，efgh 角色必须具有以下信任策略：

账户 222222222222 efgh 角色信任策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "efghTrustPolicy",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {"AWS": "arn:aws:iam::111111111111:role/abcd"}
    }
  ]
}
```

怎样入门？

为了理解角色如何在 Amazon EC2 实例中起作用，您需要使用 IAM 控制台创建一个角色，启动使用该角色的 Amazon EC2 实例，然后检查正在运行的实例。您可以检查[实例元数据](#)，以了解如何为实例提供角色的临时凭证。也可以了解实例上运行的应用程序如何使用角色。使用以下资源了解更多信息。

-
- SDK 演示。Amazon 开发工具包文档包含一些演示，这些演示说明了在 Amazon EC2 实例上运行的应用程序使用角色的临时凭证读取 Amazon S3 存储桶。以下每个演示都采用了不同的编程语言来呈现类似的步骤：
 - Amazon SDK for Java 开发人员指南中的[使用 SDK for Java 为 Amazon EC2 配置 IAM 角色](#)
 - 《Amazon SDK for .NET 开发人员指南》中的[使用 SDK for .NET 启动 Amazon EC2 实例](#)
 - Amazon SDK for Ruby 开发人员指南中的[使用 SDK for Ruby 创建 Amazon EC2 实例](#)

相关信息

有关创建角色或适用于 Amazon EC2 实例的角色的更多信息，请参阅以下信息：

- 有关 [将 IAM 角色与 Amazon EC2 实例结合使用](#) 的更多信息，请转至适用于 Linux 实例的 Amazon EC2 用户指南。
- 如需创建角色，请参阅 [创建 IAM 角色 \(p. 215\)](#)
- 有关使用临时安全凭证的更多信息，请参阅 [IAM 临时安全凭证 \(p. 300\)](#)。
- 如果您操作 IAM API 或 CLI，必须创建并管理 IAM 实例配置文件。有关实例配置文件的更多信息，请参阅 [使用实例配置文件 \(p. 258\)](#)。
- 有关实例元数据中的角色临时安全凭证的更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的 [从实例元数据中检索安全凭证](#)。

使用实例配置文件

使用实例配置文件将 IAM 角色传递给 EC2 实例。有关更多信息，请参见适用于 Linux 实例的 Amazon EC2 用户指南中的 [Amazon EC2 的 IAM 角色](#)。

管理实例配置文件 (控制台)

如果使用 Amazon Web Services Management Console 创建 Amazon EC2 的角色，则控制台自动创建实例配置文件，将其命名为与角色相同的名称。当您随后通过 IAM 角色使用 Amazon EC2 控制台启动实例时，可以选择一个角色与该实例关联。在控制台中，显示的列表实际上是实例配置文件名称的列表。控制台不会为不与 Amazon EC2 关联的角色创建实例配置文件。

如果该角色和实例配置文件的名称相同，则您可以使用 Amazon Web Services Management Console 删除 Amazon EC2 的 IAM 角色和实例配置文件。要了解删除实例配置文件的更多信息，请参阅 [删除角色或实例配置文件 \(p. 271\)](#)。

管理实例配置文件 (Amazon CLI 或 Amazon API)

如果您从 Amazon CLI 或 Amazon API 管理角色，则需要使用单独的操作创建角色和实例配置文件。由于角色和实例配置文件可能具有不同名称，则您必须知道实例配置文件的名称及其包含的角色名称。这样，您才能在启动 EC2 实例时选择正确的实例配置文件。

您可以将标签附加到 IAM 资源（包括实例配置文件），以识别、组织和控制对这些资源的访问。只有在使用 Amazon CLI 或 Amazon API 时，才能标记实例配置文件。

Note

一个实例配置文件只能包含一个 IAM 角色，不过一个角色可以包含在多个实例配置文件中。不能提高每个实例配置文件一个角色这一限制。您可以删除实例配置文件中的现有角色，然后添加另一角色。由于 [最终一致性](#)，您必须等待此更改在整个 Amazon 中出现。要强制进行更改，您必须 [取消关联实例配置文件](#)，然后再次 [关联实例配置文件](#)；您也可以停止并重新启动实例。

管理实例配置文件 (Amazon CLI)

您可以使用以下 Amazon CLI 命令在 Amazon 账户中处理实例配置文件。

- 创建实例配置文件: `aws iam create-instance-profile`
- 标记实例配置文件: `aws iam tag-instance-profile`
- 列出实例配置文件的标签: `aws iam list-instance-profile-tags`
- 取消标记实例配置文件: `aws iam untag-instance-profile`
- 向实例配置文件添加角色: `aws iam add-role-to-instance-profile`
- 列出实例配置文件: `aws iam list-instance-profiles`, `aws iam list-instance-profiles-for-role`
- 获取有关实例配置文件的信息: `aws iam get-instance-profile`
- 从实例配置文件中删除角色: `aws iam remove-role-from-instance-profile`
- 删除实例配置文件: `aws iam delete-instance-profile`

您还可以使用以下命令，将角色附加到已在运行的 EC2 实例。有关更多信息，请参阅 [Amazon EC2 的 IAM 角色](#)。

- 将实例配置文件随角色附加到已停止或正在运行的 EC2 实例: `aws ec2 associate-iam-instance-profile`
- 获取 EC2 实例附加的实例配置文件信息: `aws ec2 describe-iam-instance-profile-associations`
- 将角色的实例配置文件与已停止或正在运行的 EC2 实例分离: `aws ec2 disassociate-iam-instance-profile`

管理实例配置文件 (Amazon API)

您可以调用以下 Amazon API 操作在 Amazon 账户中处理实例配置文件。

- 创建实例配置文件: `CreateInstanceProfile`
- 标记实例配置文件: `TagInstanceProfile`
- 列出实例配置文件上的标签: `ListInstanceProfileTags`
- 取消标记实例配置文件: `UntagInstanceProfile`
- 向实例配置文件添加角色: `AddRoleToInstanceProfile`
- 列出实例配置文件: `ListInstanceProfiles`, `ListInstanceProfilesForRole`
- 获取有关实例配置文件的信息: `GetInstanceProfile`
- 从实例配置文件中删除角色: `RemoveRoleFromInstanceProfile`
- 删除实例配置文件: `DeleteInstanceProfile`

您还可以调用以下操作，将角色附加到已在运行的 EC2 实例。有关更多信息，请参阅 [Amazon EC2 的 IAM 角色](#)。

- 将实例配置文件随角色附加到已停止或正在运行的 EC2 实例: `AssociateIamInstanceProfile`
- 获取 EC2 实例附加的实例配置文件信息: `DescribeIamInstanceProfileAssociations`
- 将角色的实例配置文件与已停止或正在运行的 EC2 实例分离: `DisassociateIamInstanceProfile`

撤消 IAM 角色临时安全凭证

Warning

如果您执行此页面上的步骤，则通过担任角色创建的具有当前会话的所有用户对所有 Amazon 操作和资源的访问将被拒绝。这会导致用户丢失未保存的工作。

在允许用户访问具有较长的会话持续时间（例如 12 小时）的 Amazon Web Services Management Console 时，用户的临时凭证不会很快过期。如果用户无意中向未授权的第三方公开其凭证，则第三方在会话的持续时间内将具有访问权限。不过，如果需要，可以撤销对某个特定时间点之前发布的角色凭证的所有权限。指定时间之前发布的该角色的所有临时凭证将变得无效。这将迫使所有用户重新进行身份验证并请求新的凭证。

Note

您无法撤销 [服务相关角色](#) (p. 159) 对会话的权限。

在使用本主题中的过程撤消角色的权限时，Amazon 会向角色附加新的内联策略来拒绝对所有操作的所有权限。它包括仅当用户在撤销权限时的某个时间点之前 代入角色的情况下应用限制的情况。如果用户在您撤销权限之后 代入角色，则拒绝策略不适用于该用户。

Important

此拒绝策略适用于指定角色的所有用户，而不只是适用于具有持续时间更长的控制台会话的用户。

从角色撤消会话权限所需的最低权限

要从角色成功撤消会话权限，您必须具有该角色的 `PutRolePolicy` 权限。这允许您将 `AWSRevokeOlderSessions` 内联策略附加到该角色。

撤消会话权限

您可以从角色撤消会话权限。

立即拒绝对角色凭证的任何当前用户的所有权限

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，请选择 Roles（角色），然后选择要撤消其权限的角色的名称（而不是复选框）。
3. 在所选角色的 Summary 页面上，选择 Revoke sessions 选项卡。
4. 在 Revoke sessions 选项卡上，选择 Revoke active sessions。
5. Amazon 要求您确认此操作。选择 I acknowledge that I am revoking all active sessions for this role.（我确认我正在撤消此角色的所有活动会话。）复选框，然后在对话框中选择 Revoke active sessions（撤销活动会话）。

IAM 立即将名为 `AWSRevokeOlderSessions` 的策略附加到角色。此策略拒绝对在您选择 Revoke active sessions（撤消活动会话）之前代入角色的用户的所有访问。任何在您选择 Revoke active sessions（撤消活动会话）之后代入角色的用户均不受影响。

在将新策略应用于用户或资源时，可能需要几分钟时间，策略更新才能生效。要了解为什么更改并非始终立即可见，请参阅 [我所做的更改可能不会立即可见 \(p. 865\)](#)。

Note

不必担心要记住删除策略。任何在您撤消会话之后代入角色的用户均不受该策略影响。如果您稍后再次选择 Revoke Sessions（撤消会话），则将刷新策略中的日期和时间戳，并将再次拒绝对在新的指定时间之前代入角色的任何用户的所有权限。

以这种方式调用会话的有效用户必须获得临时凭证，新会话才能继续工作。在凭证过期前，Amazon CLI 会缓存凭证。要强制 CLI 删除并刷新已失效的缓存证书，请运行以下命令之一：

Linux、macOS 或 Unix

```
$ rm -r ~/.aws/cli/cache
```

Windows

```
C:\> del /s /q %UserProfile%\aws\cli\cache
```

有关更多信息，请参阅 [禁用临时安全凭证的权限 \(p. 328\)](#)。

管理 IAM 角色

有时，您需要修改或删除已创建的角色。要更改角色，您可以执行任何以下操作：

- 修改与角色关联的策略
- 更改可访问角色的人员
- 编辑角色为用户授予的权限
- 更改使用 Amazon Web Services Management Console、Amazon CLI 或 API 担任的角色的最大会话持续时间设置

您还可以删除不再需要的角色。您可通过 Amazon Web Services Management Console、Amazon CLI 和 API 管理您的角色。

主题

- [修改角色 \(p. 261\)](#)
- [删除角色或实例配置文件 \(p. 271\)](#)

修改角色

可以使用 Amazon Web Services Management Console、Amazon CLI 或 IAM API 对角色进行更改。

主题

- [查看角色访问 \(p. 261\)](#)
- [基于访问信息生成策略 \(p. 261\)](#)
- [修改角色 \(控制台\) \(p. 261\)](#)
- [修改角色 \(Amazon CLI\) \(p. 264\)](#)
- [修改角色 \(Amazon API\) \(p. 268\)](#)

查看角色访问

在更改角色的权限之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅[使用上次访问的信息优化 Amazon 中的权限 \(p. 467\)](#)。

基于访问信息生成策略

有时，您可能会向 IAM 实体（用户或角色）授予超出其需要的权限。为帮助您优化授予的权限，您可以根据实体的访问活动生成 IAM 策略。IAM 访问分析器会查看您的 Amazon CloudTrail 日志并生成一个策略模板，其中包含实体在指定日期范围内使用的权限。您可以使用模板创建具有精细权限的托管策略，然后将其附加到 IAM 实体。这样，您仅需授予用户或角色与特定使用案例中的 Amazon 资源进行交互所需的权限。要了解更多信息，请参阅[“基于访问活动生成策略 \(p. 443\)”](#)。

修改角色 (控制台)

可以使用 Amazon Web Services Management Console 修改角色。要更改角色的标签集，请参阅[管理 IAM 角色的标签 \(控制台\) \(p. 282\)](#)。

主题

- [修改角色信任策略 \(控制台\) \(p. 261\)](#)
- [修改角色权限策略 \(控制台\) \(p. 263\)](#)
- [修改角色描述 \(控制台\) \(p. 263\)](#)
- [修改角色最大会话持续时间 \(控制台\) \(p. 264\)](#)
- [修改角色权限边界 \(控制台\) \(p. 264\)](#)

修改角色信任策略 (控制台)

要更改可担任角色的人员，您必须修改角色的信任策略。您无法修改[服务相关角色 \(p. 159\)](#)的信任策略。

注意

- 如果用户被列为角色的信任策略中的主体，但无法担任该角色，请检查用户的[权限边界 \(p. 370\)](#)。如果为用户设置了权限边界，则它必须允许该 `sts:AssumeRole` 操作。
- 要允许用户在角色会话中重新代入当前角色，请将角色 ARN 或 Amazon Web Services 账户 ARN 指定为角色信任策略中的主体。提供计算资源的 Amazon Web Services（例如 Amazon

EC2、Amazon ECS、Amazon EKS 和 Lambda) 会提供临时凭证并自动轮换这些凭证。这样可以确保您始终拥有一组有效的凭证。对于这些服务, 无需重新代入当前角色即可获得临时凭证。但是, 如果您需要传递 [会话标签 \(p. 293\)](#) 或者 [会话策略 \(p. 357\)](#), 则需要重新代入当前角色。

修改角色信任策略 (控制台)

1. 登录 Amazon Web Services Management Console, 打开 IAM 控制台: <https://console.aws.amazon.com/iam/>。
2. 在 IAM 控制台的导航窗格中, 选择角色。
3. 在您的账户的角色列表中, 选择要修改的角色的名称。
4. 选择 Trust relationships (信任关系) 选项卡, 然后选择 Edit trust policy (编辑信任策略)。
5. 根据需要编辑信任策略。要添加其他可担任角色的主体, 请在 Principal 元素中指定他们。例如, 以下策略代码段演示如何在 Principal 元素中引用两个 Amazon 账户:

```
"Principal": {
  "AWS": [
    "arn:aws:iam::111122223333:root",
    "arn:aws:iam::444455556666:root"
  ]
},
```

如果您指定其他账户中的主体, 将账户添加到角色的信任策略只是建立跨账户信任关系工作的一半而已。默认情况下, 受信任账户中的任何用户均无法担任角色。新的受信任账户的管理员必须授予用户担任角色的权限。为此, 管理员必须创建或编辑附加到用户以允许该用户访问 `sts:AssumeRole` 操作的策略。有关更多信息, 请参阅以下过程或[向用户授予切换角色的权限 \(p. 239\)](#)。

以下策略代码段演示如何在 Principal 元素中引用两个 Amazon 服务:

```
"Principal": {
  "Service": [
    "opsworks.amazonaws.com",
    "ec2.amazonaws.com"
  ]
},
```

6. 在编辑完信任策略后, 请选择 Update policy (更新策略) 以保存所做更改。
- 有关策略结构和语法的更多信息, 请参阅[IAM 中的策略和权限 \(p. 355\)](#)和[IAM JSON 策略元素参考 \(p. 934\)](#)。

允许可信外部账户中的用户使用角色 (控制台)

有关此过程的更多详细信息, 请参阅[向用户授予切换角色的权限 \(p. 239\)](#)。

1. 登录受信任外部 Amazon 账户。
 2. 确定将权限附加到用户还是附加到组。在 IAM 控制台的导航窗格中, 相应选择 Users (用户) 或 User Groups (用户组)。
 3. 选择您要向其授予访问权限的用户或组的名称, 然后选择 Permissions 选项卡。
 4. 请执行下列操作之一:
- 要编辑某个客户托管策略, 请选择该策略的名称, 选择编辑策略, 然后选择 JSON 选项卡。您不能编

辑 Amazon 托管策略。Amazon 托管策略随 Amazon 图标 () 一起显示。有关 Amazon 托管策略与客户托管策略之间的差别的更多信息, 请参阅[托管策略与内联策略 \(p. 364\)](#)。

- 要编辑某个内联策略，请选择该策略名称旁边的箭头，然后选择 Edit policy。
5. 在策略编辑器中，添加一个新的 Statement 元素，指定以下内容：

```
{
  "Effect": "Allow",
  "Action": "sts:AssumeRole",
  "Resource": "arn:aws:iam::ACCOUNT-ID:role/ROLE-NAME"
}
```

将语句中的 ARN 替换为用户可担任的角色的 ARN。

6. 按照屏幕上的提示完成策略的编辑。

修改角色权限策略 (控制台)

要更改该角色允许的权限，请修改该角色的权限策略。您无法修改 IAM 中的 [服务相关角色 \(p. 159\)](#) 的权限策略。您可能能够修改依赖角色的服务中的权限策略。要检查服务是否支持此功能，请参阅 [使用 IAM 的 Amazon 服务 \(p. 908\)](#) 并查找服务相关角色中列为是的服务。请选择 Yes 与查看该服务的 [服务相关角色文档](#) 的链接。

更改角色允许的权限 (控制台)

1. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在 IAM 控制台的导航窗格中，选择 Roles。
3. 选择要修改的角色的名称，然后选择权限选项卡。
4. 请执行下列操作之一：
 - 要编辑某个现有客户托管策略，请选择该策略的名称，然后选择 Edit policy。

Note

您无法编辑 Amazon 托管策略。Amazon 托管策略随 Amazon 图标 () 一起显示。有关 Amazon 托管策略与客户托管策略之间的差别的更多信息，请参阅 [托管策略与内联策略 \(p. 364\)](#)。

- 要将现有的托管策略附加到角色，请选择 Add permissions (添加权限)，然后选择 Attach policies (附加策略)。
- 要编辑现有内联策略，请展开策略并选择 Edit (编辑)。
- 要嵌入新的内联策略，请选择 Add permissions (添加权限)，然后选择 Create inline policy (创建内联策略)。

修改角色描述 (控制台)

要更改角色的描述，请修改描述文本。

更改角色的描述 (控制台)

1. 登录 Amazon Web Services Management Console，打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在 IAM 控制台的导航窗格中，选择角色。
3. 以下代码示例显示如何将 IAM policy 附加到用户。
4. 在 Summary (摘要) 部分中，选择 Edit (编辑)。
5. 在框中键入新描述，然后选择 Save changes (保存更改)。

修改角色最大会话持续时间 (控制台)

要为使用控制台、Amazon CLI 或 Amazon API 代入的角色指定最大会话持续时间设置，请修改最大会话持续时间设置值。该设置可以具有 1 小时到 12 小时之间的值。如果未指定值，则应用默认最大值 (1 小时)。该设置不限制 Amazon 服务建立的会话。

更改使用控制台、Amazon CLI 或 Amazon API 担任的角色的最大会话持续时间设置 (控制台)

1. 登录 Amazon Web Services Management Console，打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在 IAM 控制台的导航窗格中，选择角色。
3. 以下代码示例显示如何将 IAM policy 附加到用户。
4. 在 Summary (摘要) 部分中，选择 Edit (编辑)。
5. 对于 Maximum session duration (最大会话持续时间)，请选择一个值。您还可以选择 Custom duration (自定义持续时间) 并输入一个值 (以秒为单位)。
6. 选择保存更改。

在某个人下次担任该角色后，您所做的更改才会生效。要了解如何撤销该角色的现有会话，请参阅[撤销 IAM 角色临时安全凭证 \(p. 259\)](#)。

在预设情况下，在 Amazon Web Services Management Console 中的 IAM 用户会话为 12 小时。在控制台内切换角色的 IAM 用户被授予最大会话持续时间或 IAM 用户会话中的剩余时间 (以较少者为准)。

从 Amazon CLI 或 Amazon API 代入角色的任何人都可以请求更长的会话，最多达到这个最大值。MaxSessionDuration 设置确定可请求的最大角色会话的持续时间。

- 要使用 Amazon CLI 指定会话持续时间，请使用 duration-seconds 参数。要了解更多信息，请参阅[“切换到 IAM 角色 \(Amazon CLI\) \(p. 245\)”](#)。
- 要使用 Amazon API 指定会话持续时间，请使用 DurationSeconds 参数。要了解更多信息，请参阅[“切换到 IAM 角色 \(Amazon API\) \(p. 250\)”](#)。

修改角色权限边界 (控制台)

要更改对某角色允许的最大权限，请修改角色的[权限边界 \(p. 370\)](#)。

更改用于设置角色的权限边界的策略

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择角色。
3. 选择具有您要更改的 [权限边界 \(p. 370\)](#) 的角色名称。
4. 请选择 Permissions 选项卡。如有必要，打开权限边界部分，然后选择更改边界。
5. 选择要用于权限边界的策略。
6. 选择更改边界。

在某个人下次担任该角色后，您所做的更改才会生效。

修改角色 (Amazon CLI)

可以使用 Amazon Command Line Interface 修改角色。要更改角色的标签集，请参阅[管理 IAM 角色 \(Amazon CLI 或 Amazon API \) 的标签 \(p. 282\)](#)。

主题

- [修改角色信任策略 \(Amazon CLI\) \(p. 265\)](#)
- [修改角色权限策略 \(Amazon CLI\) \(p. 266\)](#)
- [修改角色描述 \(Amazon CLI\) \(p. 267\)](#)
- [修改角色最大会话持续时间 \(Amazon CLI\) \(p. 267\)](#)
- [修改角色权限边界 \(Amazon CLI\) \(p. 267\)](#)

修改角色信任策略 (Amazon CLI)

要更改可担任角色的人员，您必须修改角色的信任策略。您无法修改[服务相关角色 \(p. 159\)](#)的信任策略。

注意

- 如果用户被列为角色的信任策略中的主体，但无法担任该角色，请检查用户的[权限边界 \(p. 370\)](#)。如果为用户设置了权限边界，则它必须允许该 `sts:AssumeRole` 操作。
- 要允许用户在角色会话中重新代入当前角色，请将角色 ARN 或 Amazon Web Services 账户 ARN 指定为角色信任策略中的主体。提供计算资源的 Amazon Web Services (例如 Amazon EC2、Amazon ECS、Amazon EKS 和 Lambda) 会提供临时凭证并自动轮换这些凭证。这样可以确保您始终拥有一组有效的凭证。对于这些服务，无需重新代入当前角色即可获得临时凭证。但是，如果您需要传递 [会话标签 \(p. 293\)](#) 或者 [会话策略 \(p. 357\)](#)，则需要重新代入当前角色。要了解如何修改角色信任策略以添加主体角色 ARN 或 Amazon Web Services 账户 ARN，请参阅 [修改角色信任策略 \(控制台\) \(p. 261\)](#)。

修改角色信任策略 (Amazon CLI)

1. (可选) 如果不知道要修改的角色的名称，请运行以下命令以列出账户中的角色：
 - `aws iam list-roles`
2. (可选) 要查看角色当前的信任策略，请运行以下命令：
 - `aws iam get-role`
3. 要修改可访问角色的受信任主体，请创建带有已更新信任策略的文本文件。您可以使用任何文本编辑器构件策略。

例如，以下信任策略说明了如何在 Principal 元素中引用两个 Amazon 账户。这允许两个单独的 Amazon 账户中的用户担任此角色。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": { "AWS": [
      "arn:aws:iam::111122223333:root",
      "arn:aws:iam::444455556666:root"
    ] },
    "Action": "sts:AssumeRole"
  }
}
```

如果您指定其他账户中的主体，将账户添加到角色的信任策略只是建立跨账户信任关系工作的一半而已。默认情况下，受信任账户中的任何用户均无法担任角色。新的受信任账户的管理员必须授予用户担任角色的权限。为此，管理员必须创建或编辑附加到用户以允许该用户访问 `sts:AssumeRole` 操作的策略。有关更多信息，请参阅以下过程或[向用户授予切换角色的权限 \(p. 239\)](#)。

4. 要使用您刚刚创建的文件来更新信任策略，请运行以下命令：

- [aws iam update-assume-role-policy](#)

允许受信任外部账户中的用户使用角色 (Amazon CLI)

有关此过程的更多详细信息，请参阅[向用户授予切换角色的权限](#) (p. 239)。

1. 创建一个 JSON 文件，其中包含授予担任角色的权限的权限策略。例如，下面的策略包含最低必需权限：

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

将语句中的 ARN 替换为用户可担任的角色的 ARN。

2. 运行以下命令来将包含信任策略的 JSON 文件上传到 IAM：

- [aws iam create-policy](#)

此命令的输出包含策略的 ARN。请记录此 ARN，因为您在后面的步骤中需要它。

3. 决定要将策略附加到哪个用户或组。如果不知道目标用户或组的名称，请使用下列命令之一列出账户中的用户或组：

- [aws iam list-users](#)
- [aws iam list-groups](#)

4. 使用以下命令之一，将您在上一步中创建的策略附加到用户或组：

- [aws iam attach-user-policy](#)
- [aws iam attach-group-policy](#)

修改角色权限策略 (Amazon CLI)

要更改该角色允许的权限，请修改该角色的权限策略。您无法修改 IAM 中的[服务相关角色](#) (p. 159)的权限策略。您可能能够修改依赖角色的服务中的权限策略。要检查服务是否支持此功能，请参阅[使用 IAM 的 Amazon 服务](#) (p. 908)并查找服务相关角色中列为是的服务。请选择 Yes 与查看该服务的[服务相关角色](#)文档的链接。

更改角色允许的权限 (Amazon CLI)

1. (可选) 如需查看当前与角色关联的权限，请运行以下命令：

1. [aws iam list-role-policies](#) (用于列出内联策略)
2. [aws iam list-attached-role-policies](#) (用于列出托管策略)

2. 对于更新角色的权限所使用的命令，根据您是在更新托管策略还是内联策略而有所不同。

要更新托管策略，请运行以下命令以创建托管策略的新版本：

- [aws iam create-policy-version](#)

要更新内联策略，请运行以下命令：

- [aws iam put-role-policy](#)

修改角色描述 (Amazon CLI)

要更改角色的描述，请修改描述文本。

更改角色的描述 (Amazon CLI)

1. (可选) 要查看角色的当前描述，请运行以下命令：

- [aws iam get-role](#)

2. 要更新角色的描述，请带描述参数运行以下命令：

- [aws iam update-role](#)

修改角色最大会话持续时间 (Amazon CLI)

要为使用 Amazon CLI 或 API 担任的角色指定最大会话持续时间设置，请修改最大会话持续时间设置的值。该设置可以具有 1 小时到 12 小时之间的值。如果未指定值，则应用默认最大值 (1 小时)。该设置不限制 Amazon 服务建立的会话。

Note

从 Amazon CLI 或 API 中担任角色的任何人都可以使用 `duration-seconds` CLI 参数或 `DurationSeconds` API 参数请求更长的会话。`MaxSessionDuration` 设置确定可使用 `DurationSeconds` 参数请求的最大角色会话持续时间。如果用户未指定 `DurationSeconds` 参数值，其安全凭证的有效期为 1 小时。

使用 Amazon CLI (Amazon CLI) 更改担任的角色的最大会话持续时间设置

1. (可选) 要查看角色的当前最大会话持续时间设置，请运行以下命令：

- [aws iam get-role](#)

2. 要更新角色的最大会话持续时间设置，请带 `max-session-duration` CLI 参数或 `MaxSessionDuration` API 参数运行以下命令：

- [aws iam update-role](#)

在某个人下次担任该角色后，您所做的更改才会生效。要了解如何撤销该角色的现有会话，请参阅[撤销 IAM 角色临时安全凭证](#) (p. 259)。

修改角色权限边界 (Amazon CLI)

要更改对某角色允许的最大权限，请修改角色的[权限边界](#) (p. 370)。

更改用于设置角色的权限边界的托管策略 (Amazon CLI)

1. (可选) 要查看角色的当前[权限边界](#) (p. 370)，请运行以下命令：

- [aws iam get-role](#)

2. 要使用不同的托管策略来更新角色的权限边界，请运行以下命令：

- [aws iam put-role-permissions-boundary](#)

角色只能具有一个设置为权限边界的托管策略。如果您更改权限边界，则会更改允许的角色最大权限。

修改角色 (Amazon API)

可以使用 Amazon API 修改角色。要更改角色的标签集，请参阅[管理 IAM 角色 \(Amazon CLI 或 Amazon API \) 的标签 \(p. 282\)](#)。

主题

- [修改角色信任策略 \(Amazon API\) \(p. 268\)](#)
- [修改角色权限策略 \(Amazon API\) \(p. 269\)](#)
- [修改角色描述 \(Amazon API\) \(p. 270\)](#)
- [修改角色最长会话持续时间 \(Amazon API\) \(p. 270\)](#)
- [修改角色权限边界 \(Amazon API\) \(p. 270\)](#)

修改角色信任策略 (Amazon API)

要更改可担任角色的人员，您必须修改角色的信任策略。您无法修改[服务相关角色 \(p. 159\)](#)的信任策略。

注意

- 如果用户被列为角色的信任策略中的主体，但无法担任该角色，请检查用户的[权限边界 \(p. 370\)](#)。如果为用户设置了权限边界，则它必须允许该 `sts:AssumeRole` 操作。
- 要允许用户在角色会话中重新代入当前角色，请将角色 ARN 或 Amazon Web Services 账户 ARN 指定为角色信任策略中的主体。提供计算资源的 Amazon Web Services (例如 Amazon EC2、Amazon ECS、Amazon EKS 和 Lambda) 会提供临时凭证并自动轮换这些凭证。这样可以确保您始终拥有一组有效的凭证。对于这些服务，无需重新代入当前角色即可获得临时凭证。但是，如果您需要传递 [会话标签 \(p. 293\)](#) 或者 [会话策略 \(p. 357\)](#)，则需要重新代入当前角色。要了解如何修改角色信任策略以添加主体角色 ARN 或 Amazon Web Services 账户 ARN，请参阅 [修改角色信任策略 \(控制台\) \(p. 261\)](#)。

修改角色信任策略 (Amazon API)

1. (可选) 如果不知道要修改的角色的名称，请调用以下操作以列出账户中的角色：

- [ListRoles](#)

2. (可选) 要查看角色当前的信任策略，请调用以下操作：

- [GetRole](#)

3. 要修改可访问角色的受信任主体，请创建带有已更新信任策略的文本文件。您可以使用任何文本编辑器构件策略。

例如，以下信任策略说明了如何在 Principal 元素中引用两个 Amazon 账户。这允许两个单独的 Amazon 账户中的用户担任此角色。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": { "AWS": [
      "arn:aws:iam::111122223333:root",
      "arn:aws:iam::444455556666:root"
    ] },
    "Action": "sts:AssumeRole"
  }
}
```

如果您指定其他账户中的主体，将账户添加到角色的信任策略只是建立跨账户信任关系工作的一半而已。默认情况下，受信任账户中的任何用户均无法担任角色。新的受信任账户的管理员必须授予用户担

任角色的权限。为此，管理员必须创建或编辑附加到用户以允许该用户访问 `sts:AssumeRole` 操作的策略。有关更多信息，请参阅以下过程或[向用户授予切换角色的权限 \(p. 239\)](#)。

4. 要使用您刚刚创建的文件来更新信任策略，请调用以下操作：

- [UpdateAssumeRolePolicy](#)

允许可信外部账户中的用户使用角色 (Amazon API)

有关此过程的更多详细信息，请参阅[向用户授予切换角色的权限 \(p. 239\)](#)。

1. 创建一个 JSON 文件，其中包含授予担任角色的权限的权限策略。例如，下面的策略包含最低必需权限：

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

将语句中的 ARN 替换为用户可担任的角色的 ARN。

2. 调用以下操作来将包含信任策略的 JSON 文件上传到 IAM：

- [CreatePolicy](#)

此操作的输出包含策略的 ARN。请记录此 ARN，因为您在后面的步骤中需要它。

3. 决定要将策略附加到哪个用户或组。如果不知道目标用户或组的名称，请调用下列操作之一列出账户中的用户或组：

- [ListUsers](#)
- [ListGroups](#)

4. 调用以下操作之一，将您在上一步中创建的策略附加到用户或组：

- API：[AttachUserPolicy](#)
- [AttachGroupPolicy](#)

修改角色权限策略 (Amazon API)

要更改该角色允许的权限，请修改该角色的权限策略。您无法修改 IAM 中的[服务相关角色 \(p. 159\)](#)的权限策略。您可能能够修改依赖角色的服务中的权限策略。要检查服务是否支持此功能，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)并查找服务相关角色中列为是的服务。请选择 Yes 与查看该服务的[服务相关角色](#)文档的链接。

更改角色允许的权限 (Amazon API)

1. (可选) 如需查看当前与角色关联的权限，请调用以下操作：

1. [ListRolePolicies](#) (用于列出内联策略)
2. [ListAttachedRolePolicies](#) (用于列出托管策略)

2. 对于更新角色的权限所使用的操作，根据您是在更新托管策略还是内联策略而有所不同。

要更新托管策略，请调用以下操作以创建托管策略的新版本：

- [CreatePolicyVersion](#)

要更新内联策略，请调用以下操作：

- [PutRolePolicy](#)

修改角色描述 (Amazon API)

要更改角色的描述，请修改描述文本。

更改角色的描述 (Amazon API)

1. （可选）要查看角色当前的描述，请调用以下操作：

- [GetRole](#)

2. 要更新角色的描述，请带描述参数调用以下操作：

- [UpdateRole](#)

修改角色最长会话持续时间 (Amazon API)

要为使用 Amazon CLI 或 API 担任的角色指定最大会话持续时间设置，请修改最大会话持续时间设置的值。该设置可以具有 1 小时到 12 小时之间的值。如果未指定值，则应用默认最大值 (1 小时)。该设置不限制 Amazon 服务建立的会话。

Note

从 Amazon CLI 或 API 中担任角色的任何人都可以使用 `duration-seconds` CLI 参数或 `DurationSeconds` API 参数请求更长的会话。`MaxSessionDuration` 设置确定可使用 `DurationSeconds` 参数请求的最大角色会话持续时间。如果用户未指定 `DurationSeconds` 参数值，其安全凭证的有效期为 1 小时。

使用 API 更改担任的角色的最大会话持续时间设置 (Amazon API)

1. （可选）要查看角色的当前最大会话持续时间设置，请调用以下操作：

- [GetRole](#)

2. 要更新角色的最大会话持续时间设置，请带 `max-sessionduration` CLI 参数或 `MaxSessionDuration` API 参数调用以下操作：

- [UpdateRole](#)

在某个人下次担任该角色后，您所做的更改才会生效。要了解如何撤销该角色的现有会话，请参阅[撤销 IAM 角色临时安全凭证](#) (p. 259)。

修改角色权限边界 (Amazon API)

要更改对某角色允许的最大权限，请修改角色的[权限边界](#) (p. 370)。

更改用于设置角色的权限边界的托管策略 (Amazon API)

1. （可选）要查看角色的当前[权限边界](#) (p. 370)，请调用以下操作：

- [GetRole](#)

2. 要使用不同的托管策略来更新角色的权限边界，请调用以下操作：

- [PutRolePermissionsBoundary](#)

角色只能具有一个设置为权限边界的托管策略。如果您更改权限边界，则会更改允许的角色最大权限。

删除角色或实例配置文件

如果您不再需要某个角色，我们建议您删除该角色及其关联的权限。这样您就没有未被主动监控或维护的未使用实体。

如果该角色与 EC2 实例关联，您还可以从实例配置文件中删除角色，然后删除实例配置文件。

Warning

确保您没有使用要删除的角色或实例配置文件运行任何 Amazon EC2 实例。删除与正在运行的实例关联的角色或实例配置文件将中断该实例上正在运行的所有应用程序。

如果您不希望永久删除角色，则可以禁用角色。为此，请更改角色的策略，然后撤销所有当前会话。例如，您可以将策略添加到拒绝访问所有 Amazon 的角色。您还可以编辑信任策略，以拒绝任何试图代入此角色的用户的访问。有关撤销会话的更多信息，请参阅 [撤销 IAM 角色临时安全凭证 \(p. 259\)](#)。

主题

- [查看角色访问 \(p. 271\)](#)
- [删除服务相关角色 \(p. 272\)](#)
- [删除 IAM 角色 \(控制台\) \(p. 272\)](#)
- [创建 IAM 角色 \(Amazon CLI\) \(p. 272\)](#)
- [删除 IAM 角色 \(Amazon API\) \(p. 273\)](#)
- [相关信息 \(p. 274\)](#)

查看角色访问

在删除角色之前，建议您查看上次使用角色的时间。可使用 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 完成此操作。如果您不希望从使用该角色的用户处删除访问权限，则应查看此信息。

角色的上次活动日期可能与访问顾问选项卡中报告的上次日期不相符。[Access Advisor \(p. 472\)](#) (访问顾问) 选项卡仅报告角色的权限策略所允许服务的活动。角色的上次活动日期包括访问最后一次尝试 Amazon 中的任何服务。

Note

角色上次活动和访问顾问数据的跟踪周期为 400 天。如果您的地区在一年内开始支持这些功能，则此时间段可能会缩短。角色可能在 400 天之前使用过。有关跟踪周期的更多信息，请参阅 [Amazon 跟踪上次访问信息的位置 \(p. 471\)](#)。

查看上次使用角色的时间 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择角色。
3. 查找要查看其活动的角色的行。您可以使用搜索字段缩小结果范围。查看上一个活动列以查看距离上次使用角色的天数。如果在跟踪周期内未使用该角色，则表将显示无。
4. 选择角色的名称可查看更多信息。角色的摘要页面还包括上一个活动，显示上次使用角色的日期。如果在过去 400 天内未使用该角色，则上一个活动中显示在跟踪期间未被访问。

查看上次使用角色的时间 (Amazon CLI)

`aws iam get-role` – 运行此命令返回有关角色的信息，包括 `RoleLastUsed` 对象。此对象包含上次使用角色的 `Region` 和 `LastUsedDate`。如果存在 `RoleLastUsed` 但不包含值，则在跟踪周期内未使用该角色。

查看上次使用角色的时间 (Amazon API)

`GetRole` – 调用此操作以返回有关角色的信息，包括 `RoleLastUsed` 对象。此对象包含上次使用角色的 `Region` 和 `LastUsedDate`。如果存在 `RoleLastUsed` 但不包含值，则在跟踪周期内未使用该角色。

删除服务相关角色

如果角色是 [服务相关角色](#) (p. 159)，请查看链接服务的文档以了解如何删除该角色。您可以转至控制台中的 IAM Roles (角色) 页面来查看您的账户中的服务相关角色。服务相关角色在表的 Trusted entities (可信实体) 列中随 (Service-linked role) [(服务相关角色)] 一起显示。角色的 Summary (摘要) 页面上的横幅也指示角色是服务相关角色。

如果服务不包括有关删除服务相关角色的文档，您可以使用 IAM 控制台、Amazon CLI 或 API 来删除角色。有关更多信息，请参阅 [删除服务相关角色](#) (p. 213)。

删除 IAM 角色 (控制台)

当您使用 Amazon Web Services Management Console 删除角色时，IAM 还会自动删除与角色关联的任何内联策略。此外，它还会删除包含该角色的任何 Amazon EC2 实例配置文件。

Important

在某些情况下，某个角色可能与 Amazon EC2 实例配置文件关联，并且该角色和实例配置文件可能同名。在这种情况下，您可以使用 Amazon Web Services Management Console 删除角色和实例配置文件。对于您在控制台中创建的角色和实例配置文件，会自动建立这种关联。如果您从 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 创建角色，则角色和实例配置文件可能具有不同的名称。在这种情况下，您无法使用控制台删除它们。而是必须使用 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 首先从实例配置文件中删除该角色。然后必须采取单独的步骤删除该角色。

删除角色 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择角色，然后选中要删除的角色旁的复选框。
3. 在页面的顶部，选择删除用户。
4. 在确认对话框中，查看上次访问的信息，该信息显示每个选定角色上次访问 Amazon 服务的时间。这样可帮助您确认角色当前是否处于活动状态。如果要继续操作，请在文本输入字段中输入角色的名称，然后选择 Delete (删除)。如果确定，您就可以继续执行删除操作，即使仍在加载上次访问的信息。

Note

您不能使用控制台删除实例配置文件，除非它与角色同名。实例配置文件在删除角色的过程中被删除，如上述步骤所述。要在不删除角色的情况下删除实例配置文件，必须使用 Amazon CLI 或 Amazon API。有关更多信息，请参阅以下部分。

创建 IAM 角色 (Amazon CLI)

当您使用 Amazon CLI 删除角色时，必须先删除与角色关联的内联策略。此外，如果您想要删除包含角色的关联实例配置文件，必须分别删除。

删除角色 (Amazon CLI)

1. 如果您不知道要删除的角色的名称，请输入以下命令列出账户中的角色：

```
aws iam list-roles
```

该列表包括各个角色的 Amazon Resource Name (ARN)。通过 CLI 命令使用角色名称（并非 ARN）指向角色。例如，如果某个角色的 ARN 为 `arn:aws:iam::123456789012:role/myrole`，则将该角色称为 **myrole**。

2. 从角色所属的全部实例配置文件移除该角色。
 - a. 要列出与角色关联的所有实例配置文件，请输入以下命令：

```
aws iam list-instance-profiles-for-role --role-name role-name
```

- b. 要从某个实例配置文件中删除该角色，请为每个实例配置文件输入以下命令：

```
aws iam remove-role-from-instance-profile --instance-profile-name instance-profile-name --role-name role-name
```

3. 删除与角色相关的全部策略。
 - a. 要列出角色中的所有策略，请输入以下命令：

```
aws iam list-role-policies --role-name role-name
```

- b. 要删除角色中的每个策略，请为每个策略输入以下命令：

```
aws iam delete-role-policy --role-name role-name --policy-name policy-name
```

4. 输入以下命令可删除角色：

```
aws iam delete-role --role-name role-name
```

5. 如果您没有计划重新使用与角色相关联的实例配置文件，则可以输入以下命令来删除它们：

```
aws iam delete-instance-profile --instance-profile-name instance-profile-name
```

删除 IAM 角色 (Amazon API)

当您使用 IAM API 删除角色时，必须先删除与角色关联的内联策略。此外，如果您想要删除包含角色的关联实例配置文件，必须分别删除。

删除角色 (Amazon API)

1. 要列出角色所属的所有实例配置文件，请调用 [ListInstanceProfilesForRole](#)。

要从角色所属的所有实例配置文件删除该角色，请调用 [RemoveRoleFromInstanceProfile](#)。必须提交角色名称和实例配置文件名称。

如果您不会重用与角色相关联的实例配置文件，请调用 [DeleteInstanceProfile](#) 来删除它。

2. 要列出一个角色的所有策略，请调用 [ListRolePolicies](#)。

要删除与该角色相关联的所有策略，请调用 [DeleteRolePolicy](#)。必须提交角色名称和策略名称。

3. 调用 [DeleteRole](#) 删除角色。

相关信息

有关实例配置文件的基本信息，请参阅 [使用实例配置文件 \(p. 258\)](#)。

有关服务相关角色的一般信息，请参阅 [使用服务相关角色 \(p. 208\)](#)。

IAM 角色与基于资源的策略有何不同

对于一些 Amazon 服务，您可以授予对资源的跨账户访问权限。为此，您可将策略直接附加到要共享的资源，而不是将角色用作代理。要共享的资源必须支持[基于资源的策略 \(p. 379\)](#)。与基于身份的策略不同，基于资源的策略指定谁（哪个主体）可以访问该资源。

Note

IAM 角色和基于资源的策略仅在单个分区内跨账户委派访问权限。例如，假定您在标准 aws 分区的美国西部（加利福尼亚北部）中有一个账户。您在 aws-cn 分区的中国（北京）中也有一个账户。您不能使用中国（北京）的账户中 Amazon S3 基于资源的策略，来允许标准 aws 账户中用户的访问权限。

与使用角色进行跨账户访问相比，使用基于资源的策略进行跨账户访问具有一些优势。利用通过基于资源的策略访问的资源，主体仍在可信账户中工作，并且无需放弃其权限来接收角色权限。换言之，主体既能够继续访问可信账户中的资源，又能继续访问信任账户中的资源。这对于向另一个账户中的共享资源复制信息或从中复制信息之类的任务非常有用。要了解您信任区域之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅[什么是 IAM Access Analyzer？](#)。

可在基于资源的策略中指定的主体包括账户、IAM 用户、联合身份用户、IAM 角色、代入角色会话或 Amazon 服务。有关更多信息，请参阅[指定主体 \(p. 936\)](#)。

以下列表包括一些支持基于资源的策略的 Amazon 服务。有关支持向资源而非主体附加权限策略的更多 Amazon 服务的完整列表，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#) 并寻找具有基于资源列中为是的服务。

- Amazon S3 存储桶 - 策略将附加到存储桶，但该策略可控制对存储桶和存储桶中的对象的访问。有关更多信息，请转至 Amazon Simple Storage Service 用户指南中的[访问控制](#)。

在某些情况下，可能最好使用角色对 Amazon S3 进行跨账户访问。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的[示例演练](#)。

- Amazon Simple Notification Service (Amazon SNS) 主题 — 有关更多信息，请转至 Amazon Simple Notification Service 开发人员指南中的[管理对 Amazon SNS 主题的访问](#)。
- Amazon Simple Queue Service (Amazon SQS) 队列 — 有关更多信息，请转至 Amazon Simple Queue Service 开发人员指南中的[附录：访问策略语言](#)

关于在基于资源的策略中委派 Amazon 权限

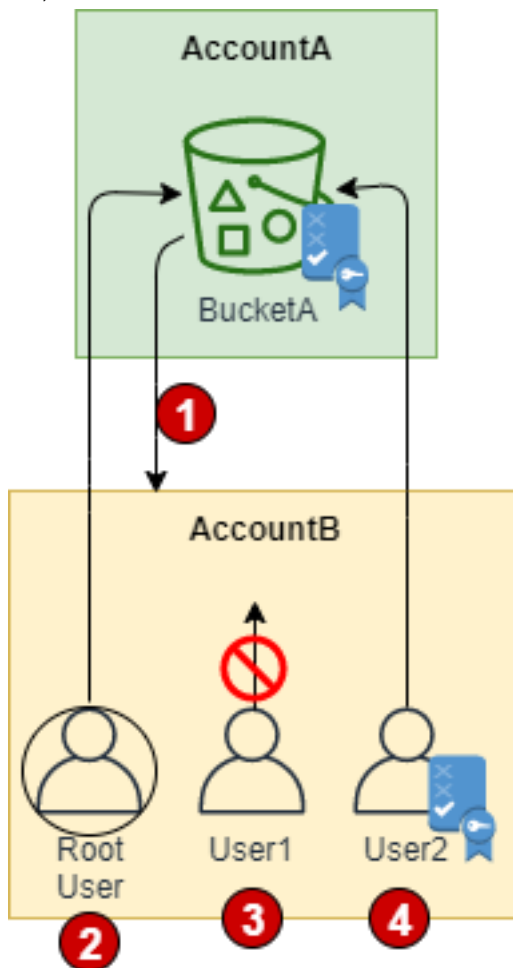
如果资源向账户中的主体授予权限，则您随后可将这些权限委派给特定的 IAM 身份。身份是您账户中的用户、用户组或角色。可以通过将策略附加到身份来委派权限。您可以授予拥有资源的账户所允许的最多权限。

假定基于资源的策略允许您账户中的所有主体对资源进行完全管理访问。随后，您可以将完全访问权限、只读访问权限或任何其他部分访问权限委派给 Amazon 账户中的主体。或者，如果基于资源的策略仅允许列表权限，则您只能委派列表访问权限。如果您尝试委派的权限超出您的账户所拥有的权限，您的主体将只有列表访问权限。有关将策略附加到 IAM 身份的信息，请参阅[管理 IAM 策略 \(p. 437\)](#)。

Note

IAM 角色和基于资源的策略仅在单个分区内跨账户委派访问权限。例如，您不能在标准 aws 分区的账户与 aws-cn 分区中的账户之间添加跨账户访问权限。

例如，假定您管理 AccountA 和 AccountB。在 AccountA 中，您将有名为 BucketA 的 Amazon S3 存储桶。您将基于资源的策略附加到 BucketA，这将允许 AccountB 中的所有主体对存储桶中的对象进行完全访问。这些主体可以创建、读取或删除该存储桶中的任何对象。在 AccountB 中，您将策略附加到名为 User2 的 IAM 用户。该策略允许用户对 BucketA 中的对象进行只读访问。这意味着，User2 可以查看对象，但不能创建、编辑或删除对象。



1. AccountA 通过在基于资源的策略中将 AccountB 指定为主体来向 BucketA 提供对 AccountB 的完全访问权限。因此，AccountB 有权对 BucketA 执行任何操作，并且 AccountB 管理员可以向其在 AccountB 中的用户委派访问权限。
2. AccountB 根用户具有授予给该账户的所有权限。因此，根用户具有对 BucketA 的完全访问权限。
3. AccountB 管理员不向 User1 授予访问权限。默认情况下，除了明确授予的权限之外，用户不具有任何权限。因此，User1 无权访问 BucketA。
4. AccountB 管理员向 User2 授予对 BucketA 的只读访问权限。User2 可以查看存储桶中的对象。AccountB 可委派的最大访问级别是授予给账户的访问级别。在此情况下，基于资源的策略授予了对 AccountB 的完全访问权限，但仅向 User2 授予了只读访问权限。

IAM 将在主体发出请求时对主体的权限进行评估。因此，如果您使用通配符 (*) 向用户授予对资源的完全访问权限，则主体可以访问您的 Amazon 账户有权访问的任何资源。甚至对于您在创建用户策略后添加或获得其访问权的资源也是如此。

在上述示例中，如果 AccountB 已将允许对所有账户中的所有资源进行完全访问的策略附加到 User2，则 User2 将自动具有 AccountB 有权访问的任何资源的访问权限。这包括 BucketA 访问权限以及由 AccountA 中的基于资源的策略授予的对任何其他资源的访问权限。

Important

仅向信任实体授予访问权限，并提供最低级别的访问权限。只要可信实体是其他 Amazon 账户，该账户就能够将访问权限委派给其任何 IAM 用户。可信 Amazon 账户可委派的访问权限范围仅限于它所获得的访问权限；它委派的权限不能超出账户本身所获得的访问权限。

有关权限、策略以及用于编写策略的权限策略语言的信息，请参阅[适用于 Amazon 资源的 Access Management \(p. 353\)](#)。

标记 IAM 资源

标签是您分配给 Amazon 资源的自定义属性标签。每个标签具有两个部分：

- 标签键（例如，CostCenter、Environment、Project 或 Purpose）。标签键区分大小写。
- 一个称为标签值的可选字段（例如，111122223333、Production 或团队名称）。省略标签值与使用空字符串相同。与标签键一样，标签值区分大小写。

这些被统称为键-值对。有关 IAM 资源上可以拥有的标签数量的限制，请参阅[IAM 与 Amazon STS 配额、名称要求和字符限制 \(p. 902\)](#)。

标签有助于您标识和组织 Amazon 资源。许多 Amazon 服务支持标记，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。例如，您可以将相同的标签分配给为 Amazon S3 存储桶分配的 IAM 角色。有关标记策略的更多信息，请参阅 Amazon 一般参考指南中的[标记 Amazon 资源](#)。

除了通过标签标识、组织和跟踪 IAM 资源之外，您还可以在 IAM policy 中使用标签，帮助控制哪些人可以查看并与您的资源交互。要了解有关使用标签控制访问的更多信息，请参阅[使用标签控制对 IAM 用户和角色的访问以及他们进行的访问 \(p. 388\)](#)。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅[在 Amazon STS 中传递会话标签 \(p. 293\)](#)。

选择 Amazon 标签命名约定

当您开始将标签附加到您的 IAM 资源时，请谨慎选择标签命名约定。对您的所有 Amazon 标签应用同一约定。如果您在策略中使用标签来控制对 Amazon 资源的访问，这一点尤为重要。如果您已在 Amazon 中使用标签，请检查您的命名约定并相应地进行调整。

Note

如果您的账户是 Amazon Organizations 的成员，请参阅《Organizations 用户指南》中的[标签策略](#)，以了解有关在 Organizations 中使用标签的更多信息。

标签命名的最佳实践

以下是标签的一些最佳实践和命名约定。

Amazon 标签的名称区分大小写，因此请确保一致地使用它们。例如，标签 CostCenter 与 costcenter 是不同的，因此一个标签可能会被配置为用于财务分析和报告的成本分配标签，而另一个标签可能不会这样配置。同样，Name 标签将出现在 Amazon 控制台中，用于很多资源，但 name 标签不会。

许多标签由 Amazon 预定义，或由各种 Amazon 服务自动创建。很多 Amazon 定义的标签名称全部使用小写字母，名称中的单词之间用连字符分隔，前缀用于标识标签的源服务。例如：

- aws:ec2spot:fleet-request-id 标识启动实例的 Amazon EC2 竞价型实例请求。
- aws:cloudformation:stack-name 标识创建资源的 Amazon CloudFormation 堆栈。

- `elasticbeanstalk:environment-name` 标识创建资源的应用程序。

考虑使用全部小写字母命名标签，用连字符分隔单词，并使用前缀标识组织名称或缩写名称。例如，可以虚构一家名为 AnyCompany 的公司，可以定义如下标签：

- `anycompany:cost-center` 标识内部成本中心代码
- `anycompany:environment-type` 确定环境是开发、测试还是生产环境
- `anycompany:application-id` 标识为其创建资源的应用程序

前缀可以确保将标签明确标识为由贵组织定义，而不是由 Amazon 或您可能正在使用的第三方工具定义。使用所有小写字母和连字符作为分隔符，可以避免对如何大写标签名称产生混淆。例如，`anycompany:project-id` 比 `ANYCOMPANY:ProjectID`、`anycompany:projectID` 或 `Anycompany:ProjectId` 更易记。

在 IAM 和 Amazon STS 中进行标记的规则

大量约定管理 IAM 和 Amazon STS 中标签的创建和应用。

命名标签

为 IAM 资源、Amazon STS 代入角色会话和 Amazon STS 联合身份用户会话制定标签命名约定时，请遵守以下约定：

字符要求 - 标签键和值可以包含字母、数字、空格以及 `_ . : / = + - @` 符号的任意组合。

区分大小写 - 标签键是否区分大小写根据标记的 IAM 资源的类型而不同。IAM 用户和角色的标签键值不区分大小写，但会保留大小写。这意味着您不能拥有单独的 **Department** 和 **department** 标签键。如果您已使用 **Department=finance** 标签标记用户并添加 **department=hr** 标签，则它会替换第一个标签。不会添加第二个标签。

对于其他 IAM 资源类型，标签键值区分大小写。这意味着您可以有单独的 **Costcenter** 和 **costcenter** 标签键。例如，如果您使用 **Costcenter = 1234** 标签为客户托管的策略添加了标签，并且添加了 **costcenter = 5678** 标签，策略将同时拥有 **Costcenter** 和 **costcenter** 标签键。

作为最佳实践，我们建议您避免使用大小写处理不一致的类似标签。我们建议您确定利用标签的策略，并在所有资源类型中一致地实施该策略。要了解有关标记的最佳实践的更多信息，请参阅 Amazon 一般参考中的 [标记 Amazon 资源](#)。

以下列表显示了附加到 IAM 资源的标签键在区分大小写方面的差异。

标签键值不区分大小写：

- IAM 角色
- IAM 用户

标签键值区分大小写。

- 客户托管策略
- 实例配置文件
- OpenID Connect 身份提供程序
- SAML 身份提供程序
- 服务器证书
- 虚拟 MFA 设备

此外，以下规则适用：

- 您不能创建以文本 **aws:** 开头的标签键或值。此标签前缀是专为 Amazon 内部使用预留的。
- 您可以使用空值创建标签（如 **phoneNumber =** ）。不能创建空标签键。
- 您不能在单个标签中指定多个值，但可以在单个值中创建自定义多值结构。例如，假设用户 Zhang 在工程团队和 QA 团队工作。如果附加 **team = Engineering** 标签，然后附加 **team = QA** 标签，则会将标签的值从 **Engineering** 更改为 **QA**。相反，您可以使用自定义分隔符在单个标签中包含多个值。在此示例中，您可以将 **team = Engineering:QA** 标签附加到 Zhang。

Note

在此示例中，要使用 **team** 标签控制对工程师的访问，您必须创建一个允许每个可能包含 **Engineering**（包括 **Engineering:QA**）的配置的策略。要了解有关在策略中使用标签的更多信息，请参阅[使用标签控制对 IAM 用户和角色的访问以及他们进行的访问 \(p. 388\)](#)。

应用和编辑标签

在将标签附加到 IAM 资源时遵循以下约定：

- 您可以标记大多数 IAM 资源，但不能标记组、代入的角色、访问报告或基于硬件的 MFA 设备。
- 不能使用标签编辑器标记 IAM 资源。标签编辑器不支持 IAM 标签。有关将标签编辑器与其他服务结合使用的信息，请参阅《Amazon Resource Groups 用户指南》中的使用标签编辑器。
- 要标记 IAM 资源，您必须拥有特定权限。要标记或取消标记资源，您还必须有权列出标签。有关详细信息，请参阅本页末尾的每个 IAM 资源的主题列表。
- Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅[IAM 与 Amazon STS 配额、名称要求和字符限制 \(p. 902\)](#)。
- 您可以将同一标签应用于多个 IAM 资源。例如，假设您有一个名为 **AWS_Development** 的部门，有 12 位成员。您可以让 12 位用户和一个角色具有标签键 **department** 以及值 **awsDevelopment (department = awsDevelopment)**。您还可以在其他[支持标记的服务 \(p. 908\)](#)中的资源上使用相同标签。
- IAM 实体（用户或角色）不能具有同一标签键的多个实例。例如，如果您有一个标签键值对为 **costCenter = 1234** 的用户，则可以附加标签键值对 **costCenter = 5678**。IAM 将 **costCenter** 标签的值更新为 **5678**。
- 要编辑附加到 IAM 实体（用户或角色）的标签，请附加具有新值的标签以覆盖现有标签。例如，假设您有一个标签键值对为 **department = Engineering** 的用户。如果需要将该用户移动到 QA 部门，则可将 **department = QA** 标签键值对附加到该用户。这将导致 **department** 标签键的 **Engineering** 值被替换为 **QA** 值。

主题

- [标记 IAM 用户 \(p. 278\)](#)
- [标记 IAM 角色 \(p. 280\)](#)
- [标记客户托管策略 \(p. 283\)](#)
- [标记 IAM 身份提供程序 \(p. 284\)](#)
- [为 Amazon EC2 角色标记实例配置文件 \(p. 288\)](#)
- [标记服务器证书 \(p. 290\)](#)
- [标记虚拟 MFA 设备 \(p. 291\)](#)
- [在 Amazon STS 中传递会话标签 \(p. 293\)](#)

标记 IAM 用户

您可以使用 IAM 标签键值对向 IAM 用户添加自定义属性。例如，要向用户添加位置信息，您可以添加标签键 **location** 和标签值 **us_wa_seattle**。或者，也可以使用三种单独的位置标签键值对：**loc-country**

`= us`、`loc-state = wa` 和 `loc-city = seattle`。您可以使用标签控制用户对资源的访问权限或控制可附加到用户的标签。要了解有关使用标签控制访问的更多信息，请参阅[使用标签控制对 IAM 用户和角色的访问以及他们进行的访问](#) (p. 388)。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅[在 Amazon STS 中传递会话标签](#) (p. 293)。

标记 IAM 用户所需的权限

您必须配置权限以允许 IAM 用户标记其他用户。您可以在 IAM 策略中指定以下一项或所有 IAM 标签操作：

- `iam:ListUserTags`
- `iam:TagUser`
- `iam:UntagUser`

要允许 IAM 用户为特定用户添加、列出或删除标签

将以下语句添加到需要管理标签的 IAM 用户的权限策略。使用您的账号并将 `<username>` 替换为需要管理其标签的用户的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略”](#) (p. 438)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListUserTags",
    "iam:TagUser",
    "iam:UntagUser"
  ],
  "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

要允许 IAM 用户自行管理标签

将以下语句添加到用户的权限策略以允许用户管理其自己的标签。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略”](#) (p. 438)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListUserTags",
    "iam:TagUser",
    "iam:UntagUser"
  ],
  "Resource": "arn:aws:iam::user/${aws:username}"
}
```

要允许 IAM 用户将标签添加到特定用户

将以下语句添加到需要为特定用户添加而不是删除标签的 IAM 用户的权限策略。

Note

`iam:TagUser` 操作要求您也包含 `iam:ListUserTags` 操作。

要使用此策略，请将 `<username>` 替换为需要管理其标签的用户的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略”](#) (p. 438)。

```
{
```

```
"Effect": "Allow",
"Action": [
    "iam:ListUserTags",
    "iam:TagUser"
],
"Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

或者，也可以使用 Amazon 托管策略（如 [IAMFullAccess](#)）来提供对 IAM 的完全访问权限。

管理 IAM 用户的标签（控制台）

您可以从 Amazon Web Services Management Console 中管理 IAM 用户的标签。

要管理用户的标签（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在控制台的导航窗格中，选择 Users（用户），然后选择要编辑的用户名称。
3. 选择 Tags（标签）选项卡，然后完成下列操作之一：
 - 如果用户还没有标签，请选择 Add tags（添加标签）。
 - 选择 Edit tags（编辑标签）来管理一组现有的标签。
4. 添加或删除标签以完成标签集。然后选择 Save changes（保存更改）。

管理 IAM 用户（Amazon CLI 或 Amazon API）的标签

您可以为 IAM 用户列出、附加或删除标签。您可以使用 Amazon CLI 或 Amazon API 管理 IAM 用户的标签。

要列出当前附加到 IAM 用户的标签（Amazon CLI 或 Amazon API）

- Amazon CLI：[aws iam list-user-tags](#)
- Amazon API：[ListUserTags](#)

将标签附加到 IAM 用户（Amazon CLI 或 Amazon API）

- Amazon CLI：[aws iam tag-user](#)
- Amazon API：[TagUser](#)

从 IAM 用户中删除标签（Amazon CLI 或 Amazon API）

- Amazon CLI：[aws iam untag-user](#)
- Amazon API：[UntagUser](#)

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 [IAM policy 元素：变量和标签](#) (p. 964)。

标记 IAM 角色

您可以使用 IAM 标签键值对向 IAM 角色添加自定义属性。例如，要向角色添加位置信息，您可以添加标签键 `location` 和标签值 `us_wa_seattle`。或者，也可以使用三种单独的位置标签键值对：`loc-country`

= **us**, **loc-state** = **wa** 和 **loc-city** = **seattle**。您可以使用标签控制角色对资源的访问权限或控制可附加到角色的标签。要了解有关使用标签控制访问的更多信息，请参阅[使用标签控制对 IAM 用户和角色的访问以及他们进行的访问](#) (p. 388)。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅[在 Amazon STS 中传递会话标签](#) (p. 293)。

标记 IAM 角色所需的权限

您必须配置权限以允许 IAM 角色标记其他实体（用户或角色）。您可以在 IAM policy 中指定以下一项或所有 IAM 标签操作：

- iam:ListRoleTags
- iam:TagRole
- iam:UntagRole
- iam:ListUserTags
- iam:TagUser
- iam:UntagUser

要允许 IAM 角色为特定用户添加、列出或删除标签

将以下语句添加到需要管理标签的 IAM 角色的权限策略。使用您的账号并将 **<username>** 替换为需要管理其标签的用户的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略”](#) (p. 438)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListUserTags",
    "iam:TagUser",
    "iam:UntagUser"
  ],
  "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

要允许 IAM 角色将标签添加到特定用户

将以下语句添加到需要为特定用户添加而不是删除标签的 IAM 角色的权限策略。

Note

iam:TagRole 操作要求您也包含 iam:ListRoleTags 操作。

要使用此策略，请将 **<username>** 替换为需要管理其标签的用户的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略”](#) (p. 438)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListUserTags",
    "iam:TagUser"
  ],
  "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

要运行 IAM 角色为特定角色添加、列出或删除标签

将以下语句添加到需要管理标签的 IAM 角色的权限策略。将 `<rolename>` 替换为需要管理其标签的角色的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略” \(p. 438\)](#)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListRoleTags",
    "iam:TagRole",
    "iam:UntagRole"
  ],
  "Resource": "arn:aws:iam::<account-number>:role/<rolename>"
}
```

或者，也可以使用 Amazon 托管策略（如 [IAMFullAccess](#)）来提供对 IAM 的完全访问权限。

管理 IAM 角色的标签（控制台）

您可以从 Amazon Web Services Management Console 中管理 IAM 角色的标签。

要管理角色的标签（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在控制台的导航窗格中，选择 Roles（角色），然后选择要编辑的角色的名称。
3. 选择 Tags (标签) 选项卡，然后完成下列操作之一：
 - 如果用户还没有标签，请选择 Add new tag（添加新标签）。
 - 选择 Manage tags（管理标签）来管理一组现有的标签。
4. 添加或删除标签以完成标签集。然后选择 Save changes（保存更改）。

管理 IAM 角色（Amazon CLI 或 Amazon API）的标签

您可以为 IAM 角色列出、附加或删除标签。您可以使用 Amazon CLI 或 Amazon API 管理 IAM 角色的标签。

列出当前附加到 IAM 角色的标签（Amazon CLI 或 Amazon API）

- Amazon CLI：[aws iam list-role-tags](#)
- Amazon API：[ListRoleTags](#)

将标签附加到 IAM 角色（Amazon CLI 或 Amazon API）

- Amazon CLI：[aws iam tag-role](#)
- Amazon API：[TagRole](#)

从 IAM 角色中删除标签（Amazon CLI 或 Amazon API）

- Amazon CLI：[aws iam untag-role](#)
- Amazon API：[UntagRole](#)

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 [IAM policy 元素：变量和标签 \(p. 964\)](#)。

标记客户托管策略

您可以使用 IAM 标签键值对将自定义属性添加到客户托管策略。例如，要使用部门信息标记策略，您可以添加标签键 **Department** 和标签值 **eng**。或者，您可能希望标签策略表明它们适用于特定环境，例如 **Environment = lab**。您可以使用标签控制对资源的访问权限或控制可附加到资源的标签。要了解有关使用标签控制访问的更多信息，请参阅[使用标签控制对 IAM 用户和角色的访问以及他们进行的访问](#) (p. 388)。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅[在 Amazon STS 中传递会话标签](#) (p. 293)。

标记客户托管策略所需的权限

您必须配置权限以允许 IAM 实体（用户或角色）标记客户托管策略。您可以在 IAM 策略中指定以下一项或所有 IAM 标签操作：

- iam:ListPolicyTags
- iam:TagPolicy
- iam:UntagPolicy

要允许 IAM 实体（用户或角色）添加、列出或删除客户托管策略的标签

将以下语句添加到需要管理标签的 IAM 实体的权限策略。使用您的账号并将 **<policyname>** 替换为需要管理其标签的策略的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略”](#) (p. 438)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListPolicyTags",
    "iam:TagPolicy",
    "iam:UntagPolicy"
  ],
  "Resource": "arn:aws:iam::<account-number>:policy/<policyname>"
}
```

要允许 IAM 实体（用户或角色）向特定客户托管策略添加标签

将以下语句添加到需要为特定策略添加而不是删除标签的 IAM 实体的权限策略。

Note

iam:TagPolicy 操作要求您也包含 iam:ListPolicyTags 操作。

要使用此策略，请将 **<policyname>** 替换为需要管理其标签的策略的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略”](#) (p. 438)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListPolicyTags",
    "iam:TagPolicy"
  ],
  "Resource": "arn:aws:iam::<account-number>:policy/<policyname>"
}
```

或者，也可以使用 Amazon 托管策略（如 [IAMFullAccess](#)）来提供对 IAM 的完全访问权限。

管理 IAM 客户托管策略的标签 (控制台)

您可以从 Amazon Web Services Management Console 中管理 IAM 客户托管策略的标签。

要管理客户托管策略的标签 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在控制台的导航窗格中，选择 Policies (策略)，然后选择要编辑的客户托管策略的名称。
3. 选择 Tags (标签) 选项卡，然后完成下列操作之一：
 - 如果策略还没有标签，请选择 Add tags (添加标签)。
 - 选择 Edit tags (编辑标签) 来管理一组现有的标签。
4. 添加或删除标签以完成标签集。然后选择 Save changes (保存更改)。

管理 IAM 客户托管策略 (Amazon CLI 或 Amazon API) 的标签

您可以列出、附加或删除 IAM 客户托管策略的标签。您可以使用 Amazon CLI 或 Amazon API 管理 IAM 客户托管策略的标签。

要列出当前附加到 IAM 客户托管策略 (Amazon CLI 或 Amazon API) 的标签

- Amazon CLI：[aws iam list-policy-tags](#)
- Amazon API：[ListPolicyTags](#)

要将标签附加到 IAM 客户托管策略 (Amazon CLI 或 Amazon API)

- Amazon CLI：[aws iam tag-policy](#)
- Amazon API：[TagPolicy](#)

要从 IAM 客户托管策略 (Amazon CLI 或 Amazon API) 中删除标签

- Amazon CLI：[aws iam untag-policy](#)
- Amazon API：[UntagPolicy](#)

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 [IAM policy 元素：变量和标签 \(p. 964\)](#)。

标记 IAM 身份提供程序

您可以使用 IAM 标签键值对向 IAM 身份提供程序 (IdP) 添加自定义属性。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅在 [Amazon STS 中传递会话标签 \(p. 293\)](#)。

要了解有关标记 IAM 中的 IdP 的信息，请参阅以下部分：

主题

- [标记 OpenID Connect \(OIDC\) 身份提供程序 \(p. 285\)](#)
- [标记 IAM SAML 身份提供程序 \(p. 286\)](#)

标记 OpenID Connect (OIDC) 身份提供程序

您可以使用 IAM 标签键值向 IAM OpenID Connect (OIDC) 身份提供程序添加自定义属性。例如，要识别 OIDC 身份提供程序，您可以添加标签键 `google` 和标签值 `oidc`。您可以使用标签控制对资源的访问权限或控制可附加到对象的标签。要了解有关使用标签控制访问的更多信息，请参阅[使用标签控制对 IAM 用户和角色的访问以及他们进行的访问](#) (p. 388)。

标记 IAM OIDC 身份提供程序所需的权限

您必须配置权限以允许 IAM OIDC 实体（用户或角色）标记 IAM OIDC 身份提供程序。您可以在 IAM 策略中指定以下一项或所有 IAM 标签操作：

- `iam:ListOpenIDConnectProviderTags`
- `iam:TagOpenIDConnectProvider`
- `iam:UntagOpenIDConnectProvider`

要允许 IAM 实体（用户或角色）添加、列出或删除 IAM OIDC 身份提供程序的标签

将以下语句添加到需要管理标签的 IAM 实体的权限策略。使用您的账号并将 `<OIDCProviderName>` 替换为需要管理其标签的 OIDC 提供商的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略”](#) (p. 438)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListOpenIDConnectProviderTags",
    "iam:TagOpenIDConnectProvider",
    "iam:UntagOpenIDConnectProvider"
  ],
  "Resource": "arn:aws:iam::<account-number>:oidc-provider/<OIDCProviderName>"
}
```

允许 IAM 实体（用户或角色）向特定 IAM OIDC 身份提供程序添加标签

将以下语句添加到需要为特定身份提供程序添加而不是删除标签的 IAM 实体的权限策略。

Note

`iam:TagOpenIDConnectProvider` 操作要求您也包含
`iam:ListOpenIDConnectProviderTags` 操作。

要使用此策略，请将 `<OIDCProviderName>` 替换为需要管理其标签的 OIDC 提供商的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略”](#) (p. 438)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListOpenIDConnectProviderTags",
    "iam:TagOpenIDConnectProvider"
  ],
  "Resource": "arn:aws:iam::<account-number>:oidc-provider/<OIDCProviderName>"
}
```

或者，也可以使用 Amazon 托管策略（如 [IAMFullAccess](#)）来提供对 IAM 的完全访问权限。

管理 IAM OIDC 身份提供程序的标签（控制台）

您可以从 Amazon Web Services Management Console 中管理 IAM OIDC 身份提供程序的标签。

Note

您只能使用新的身份提供程序控制台体验来管理标签。

要管理 OIDC 身份提供程序的标签 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在控制台的导航窗格中，选择 Identity providers (身份提供程序)，然后选择要编辑的身份提供程序的名称。
3. 在 Tags (标签) 部分中，选择 Manage tags (管理标签)，然后完整以下操作之一：
 - 如果 OIDC 身份提供程序还没有标签，请选择 Add tag (添加标签) 或添加新标签。
 - 编辑现有的标签键和值。
 - 选择 Remove tag (删除标签) 可删除标签。
4. 然后选择 Save changes (保存更改)。

管理 IAM OIDC 身份提供程序 (Amazon CLI 或 Amazon API) 的标签

您可以列出、附加或删除 IAM OIDC 身份提供程序的标签。您可以使用 Amazon CLI 或 Amazon API 管理 IAM OIDC 身份提供程序的标签。

要列出当前附加到 IAM OIDC 身份提供程序 (Amazon CLI 或 Amazon API) 的标签

- Amazon CLI：[aws iam list-open-id-connect-provider-tags](#)
- Amazon API：[ListOpenIDConnectProviderTags](#)

要将标签附加到 IAM OIDC 身份提供程序 (Amazon CLI 或 Amazon API)

- Amazon CLI：[aws iam tag-open-id-connect-provider](#)
- Amazon API：[TagOpenIDConnectProvider](#)

要从 IAM OIDC 身份提供程序 (Amazon CLI 或 Amazon API) 中删除标签

- Amazon CLI：[aws iam untag-open-id-connect-provider](#)
- Amazon API：[UntagOpenIDConnectProvider](#)

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 [IAM policy 元素：变量和标签 \(p. 964\)](#)。

标记 IAM SAML 身份提供程序

您可以使用 IAM 标签键值对将自定义属性添加到 SAML 身份提供程序。例如，要识别提供商，您可以添加标签键 `okta` 和标签值 `saml`。您可以使用标签控制对资源的访问权限或控制可附加到对象的标签。要了解有关使用标签控制访问的更多信息，请参阅[使用标签控制对 IAM 用户和角色的访问以及他们进行的访问 \(p. 388\)](#)。

标记 SAML 身份提供程序所需的权限

您必须配置权限以允许 IAM 实体 (用户或角色) 标记基于 SAML 2.0 的身份提供程序 (IdP)。您可以在 IAM 策略中指定以下一项或所有 IAM 标签操作：

- `iam:ListSAMLProviderTags`

- iam:TagSAMLProvider
- iam:UntagSAMLProvider

要允许 IAM 实体（用户或角色）添加、列出或删除 SAML 身份提供程序的标签

将以下语句添加到需要管理标签的 IAM 实体的权限策略。使用您的帐号并将 `<SAMLProviderName>` 替换为需要管理其标签的 SAML 提供商的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 [the section called “在“JSON”选项卡上创建策略” \(p. 438\)](#)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListSAMLProviderTags",
    "iam:TagSAMLProvider",
    "iam:UntagSAMLProvider"
  ],
  "Resource": "arn:aws:iam::<account-number>:saml-provider/<SAMLProviderName>"
}
```

要允许 IAM 实体（用户或角色）向特定的 SAML 身份提供程序添加标签

将以下语句添加到需要为特定 SAML 提供商添加而不是删除标签的 IAM 实体的权限策略。

Note

iam:TagSAMLProvider 操作要求您也包含 iam:ListSAMLProviderTags 操作。

要使用此策略，请将 `<SAMLProviderName>` 替换为需要管理其标签的 SAML 提供商的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 [the section called “在“JSON”选项卡上创建策略” \(p. 438\)](#)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListSAMLProviderTags",
    "iam:TagSAMLProvider"
  ],
  "Resource": "arn:aws:iam::<account-number>:saml-provider/<SAMLProviderName>"
}
```

或者，也可以使用 Amazon 托管策略（如 [IAMFullAccess](#)）来提供对 IAM 的完全访问权限。

管理 IAM SAML 身份提供程序的标签（控制台）

您可以从 Amazon Web Services Management Console 中管理 IAM SAML 身份提供程序的标签。

Note

您只能使用新的身份提供程序控制台体验来管理标签。

要管理 SAML 身份提供程序的标签（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在控制台的导航窗格中，选择 Identity providers（身份提供程序），然后选择要编辑的 SAML 身份提供程序的名称。
3. 在 Tags（标签）部分中，选择 Manage tags（管理标签），然后完整以下操作之一：
 - 如果 SAML 身份提供程序还没有标签，请选择 Add tag（添加标签），或者添加新标签。
 - 编辑现有的标签键和值。

- 选择 Remove tag (删除标签) 可删除标签。
4. 添加或删除标签以完成标签集。然后选择 Save changes (保存更改)。

管理 IAM SAML 身份提供程序 (Amazon CLI 或 Amazon API) 的标签

您可以列出、附加或删除 IAM SAML 身份提供程序的标签。您可以使用 Amazon CLI 或 Amazon API 管理 IAM SAML 身份提供程序的标签。

要列出当前附加到 SAML 身份提供程序 (Amazon CLI 或 Amazon API) 的标签

- Amazon CLI : [aws iam list-saml-provider-tags](#)
- Amazon API : [ListSAMLProviderTags](#)

要将标签附加到 SAML 身份提供程序 (Amazon CLI 或 Amazon API)

- Amazon CLI : [aws iam tag-saml-provider](#)
- Amazon API : [TagSAMLProvider](#)

要从 SAML 身份提供程序 (Amazon CLI 或 Amazon API) 中删除标签

- Amazon CLI : [aws iam untag-saml-provider](#)
- Amazon API : [UntagSAMLProvider](#)

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 [IAM policy 元素：变量和标签 \(p. 964\)](#)。

为 Amazon EC2 角色标记实例配置文件

当您启动 Amazon EC2 实例时，可指定要与实例关联的 IAM 角色。实例配置文件是 IAM 角色的容器，可用在实例启动时将角色信息传递给 Amazon EC2 实例。您可以在使用 Amazon CLI 或 Amazon API 时标记实例配置文件。

您可以使用 IAM 标签键值对将自定义属性添加到实例配置文件。例如，要将部门信息添加到实例配置文件，您可以添加标签键 **access-team** 和标签值 **eng**。这样做可以让具有匹配标签的主体访问具有相同标签的实例配置文件。您可以使用多个标签键值对来指定团队和项目：**access-team = eng** 和 **project = peg**。您可以使用标签控制用户对资源的访问权限或控制可附加到用户的标签。要了解有关使用标签控制访问的更多信息，请参阅[使用标签控制对 IAM 用户和角色的访问以及他们进行的访问 \(p. 388\)](#)。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅[在 Amazon STS 中传递会话标签 \(p. 293\)](#)。

标记实例配置文件所需的权限

您必须配置权限以允许 IAM 实体 (用户或角色) 标记实例配置文件。您可以在 IAM 策略中指定以下一项或所有 IAM 标签操作：

- iam:ListInstanceProfileTags
- iam:TagInstanceProfile
- iam:UntagInstanceProfile

要允许 IAM 实体 (用户或角色) 添加、列出或删除实例配置文件的标签

将以下语句添加到需要管理标签的 IAM 实体的权限策略。使用您的账号并将 `<InstanceProfileName>` 替换为需要管理其标签的实例配置文件的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略” \(p. 438\)](#)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListInstanceProfileTags",
    "iam:TagInstanceProfile",
    "iam:UntagInstanceProfile"
  ],
  "Resource": "arn:aws:iam::<account-number>:instance-profile/<InstanceProfileName>"
}
```

允许 IAM 实体（用户或角色）向特定实例配置文件添加标签

将以下语句添加到需要为特定实例配置文件添加而不是删除标签的 IAM 实体的权限策略。

Note

`iam:TagInstanceProfile` 操作要求您也包含 `iam:ListInstanceProfileTags` 操作。

要使用此策略，请将 `<InstanceProfileName>` 替换为需要管理其标签的实例配置文件的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略” \(p. 438\)](#)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListInstanceProfileTags",
    "iam:TagInstanceProfile"
  ],
  "Resource": "arn:aws:iam::<account-number>:instance-profile/<InstanceProfileName>"
}
```

或者，也可以使用 Amazon 托管策略（如 [IAMFullAccess](#)）来提供对 IAM 的完全访问权限。

管理实例配置文件（Amazon CLI 或 Amazon API）的标签

您可以列出、附加或删除实例配置文件的标签。您可以使用 Amazon CLI 或 Amazon API 管理实例配置文件的标签。

要列出当前附加到实例配置文件的标签（Amazon CLI 或 Amazon API）

- Amazon CLI : [aws iam list-instance-profile-tags](#)
- Amazon API : [ListInstanceProfileTags](#)

要将标签附加到实例配置文件（Amazon CLI 或 Amazon API）

- Amazon CLI : [aws iam tag-instance-profile](#)
- Amazon API : [TagInstanceProfile](#)

要从实例配置文件（Amazon CLI 或 Amazon API）中删除标签

- Amazon CLI : [aws iam untag-instance-profile](#)
- Amazon API : [UntagInstanceProfile](#)

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 [IAM policy 元素：变量和标签 \(p. 964\)](#)。

标记服务器证书

如果您使用 IAM 管理 SSL/TLS 证书，则可以使用 Amazon CLI 或 Amazon API 标记 IAM 中的服务器证书。对于 Amazon Certificate Manager (ACM) 支持的区域中的证书，我们建议您使用 ACM 而不是 IAM 预置、管理和部署您的服务器证书。在不支持的区域中，您必须将 IAM 作为证书管理器。要了解 ACM 支持的区域，请参阅 Amazon 一般参考中的 [Amazon Certificate Manager 端点和配额](#)。

您可以使用 IAM 标签键值对向服务器证书添加自定义属性。例如，要添加有关服务器证书所有者或管理员的信息，请添加标签键 **owner** 和标签值 **net-eng**。或者，您可以通过添加标签键 **CostCenter** 和标签值 **1234** 来指定成本中心。您可以使用标签控制对资源的访问权限或控制可附加到资源的标签。要了解有关使用标签控制访问的更多信息，请参阅[使用标签控制对 IAM 用户和角色的访问以及他们进行的访问 \(p. 388\)](#)。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅[在 Amazon STS 中传递会话标签 \(p. 293\)](#)。

标记服务器证书所需的权限

您必须配置权限以允许 IAM 实体（用户或角色）标记服务器证书。您可以在 IAM 策略中指定以下一项或所有 IAM 标签操作：

- iam:ListServerCertificateTags
- iam:TagServerCertificate
- iam:UntagServerCertificate

要允许 IAM 实体（用户或角色）添加、列出或删除服务器证书的标签

将以下语句添加到需要管理标签的 IAM 实体的权限策略。使用您的账户并将 **<CertificateName>** 替换为需要管理其标签的服务器证书的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略” \(p. 438\)](#)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListServerCertificateTags",
    "iam:TagServerCertificate",
    "iam:UntagServerCertificate"
  ],
  "Resource": "arn:aws:iam::<account-number>:server-certificate/<CertificateName>"
}
```

要允许 IAM 实体（用户或角色）向特定服务器证书添加标签

将以下语句添加到需要为特定服务器证书添加而不是删除标签的 IAM 实体的权限策略。

Note

iam:TagServerCertificate 操作要求您也包含 iam:ListServerCertificateTags 操作。

要使用此策略，请将 **<CertificateName>** 替换为需要管理其标签的服务器证书的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略” \(p. 438\)](#)。

```
{
  "Effect": "Allow",
```

```
"Action": [
    "iam:ListServerCertificateTags",
    "iam:TagServerCertificate"
],
"Resource": "arn:aws:iam::<account-number>:server-certificate/<CertificateName>"
}
```

或者，也可以使用 Amazon 托管策略（如 [IAMFullAccess](#)）来提供对 IAM 的完全访问权限。

管理服务器证书（Amazon CLI 或 Amazon API）的标签

您可以列出、附加或删除服务器证书的标签。您可以使用 Amazon CLI 或 Amazon API 管理服务器证书的标签。

要列出当前附加到服务器证书（Amazon CLI 或 Amazon API）的标签

- Amazon CLI : [aws iam list-server-certificate-tags](#)
- Amazon API : [ListServerCertificateTags](#)

要将标签附加到服务器证书（Amazon CLI 或 Amazon API）

- Amazon CLI : [aws iam tag-server-certificate](#)
- Amazon API : [TagServerCertificate](#)

要从服务器证书（Amazon CLI 或 Amazon API）中删除标签

- Amazon CLI : [aws iam untag-server-certificate](#)
- Amazon API : [UntagServerCertificate](#)

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 [IAM policy 元素：变量和标签](#) (p. 964)。

标记虚拟 MFA 设备

您可以使用 IAM 标签键值对向虚拟 MFA 设备添加自定义属性。例如，要为用户的虚拟 MFA 设备添加成本中心信息，可以添加标签键 **CostCenter** 和标签值 **1234**。您可以使用标签控制对资源的访问权限或控制可附加到对象的标签。要了解有关使用标签控制访问的更多信息，请参阅[使用标签控制对 IAM 用户和角色的访问以及他们进行的访问](#) (p. 388)。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅[在 Amazon STS 中传递会话标签](#) (p. 293)。

标记虚拟 MFA 设备所需的权限

您必须配置权限以允许 IAM 实体（用户或角色）标记虚拟 MFA 设备。您可以在 IAM 策略中指定以下一项或所有 IAM 标签操作：

- iam:ListMFADeviceTags
- iam:TagMFADevice
- iam:UntagMFADevice

要允许 IAM 实体（用户或角色）添加、列出或删除虚拟 MFA 设备的标签

将以下语句添加到需要管理标签的 IAM 实体的权限策略。使用您的账号并将 *<MFATokenID>* 替换为需要管理其标签的虚拟 MFA 设备的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略”](#) (p. 438)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListMFADeviceTags",
    "iam:TagMFADevice",
    "iam:UntagMFADevice"
  ],
  "Resource": "arn:aws:iam::<account-number>:mfa/<MFATokenID>"
}
```

要允许 IAM 实体（用户或角色）向特定虚拟 MFA 设备添加标签

将以下语句添加到需要为特定 MFA 设备添加而不是删除标签的 IAM 实体的权限策略。

Note

iam:TagMFADevice 操作要求您也包含 iam:ListMFADeviceTags 操作。

要使用此策略，请将 *<MFATokenID>* 替换为需要管理其标签的虚拟 MFA 设备的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略”](#) (p. 438)。

```
{
  "Effect": "Allow",
  "Action": [
    "iam:ListMFADeviceTags",
    "iam:TagMFADevice"
  ],
  "Resource": "arn:aws:iam::<account-number>:mfa/<MFATokenID>"
}
```

或者，也可以使用 Amazon 托管策略（如 [IAMFullAccess](#)）来提供对 IAM 的完全访问权限。

管理虚拟 MFA 设备（Amazon CLI 或 Amazon API）的标签

您可以列出、附加或删除虚拟 MFA 设备的标签。您可以使用 Amazon CLI 或 Amazon API 管理虚拟 MFA 设备的标签。

要列出当前附加到虚拟 MFA 设备（Amazon CLI 或 Amazon API）的标签

- Amazon CLI : [aws iam list-mfa-device-tags](#)
- Amazon API : [ListMFADeviceTags](#)

要将标签附加到虚拟 MFA 设备（Amazon CLI 或 Amazon API）

- Amazon CLI : [aws iam tag-mfa-device](#)
- Amazon API : [TagMFADevice](#)

要从虚拟 MFA 设备（Amazon CLI 或 Amazon API）中删除标签

- Amazon CLI : [aws iam untag-mfa-device](#)
- Amazon API : [UntagMFADevice](#)

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 [IAM policy 元素：变量和标签 \(p. 964\)](#)。

在 Amazon STS 中传递会话标签

会话标签是您在代入 IAM 角色或联合 Amazon STS 用户身份时传递的键/值对属性。您可以通过 Amazon STS 或您的身份提供程序 (IdP) 发出 Amazon CLI 或 Amazon API 请求来完成此操作。当您使用 Amazon STS 请求临时安全凭证时，会生成会话。会话过期并具有[凭证](#)，例如访问密钥对和会话令牌。当您使用会话凭证发出后续请求时，[请求上下文 \(p. 949\)](#)包含 `aws:PrincipalTag` ([p. 1005](#)) 上下文密钥。您可以使用策略的 `Condition` 元素中的 `aws:PrincipalTag` 键，基于这些标签来允许或拒绝访问。

当您使用临时凭证发出请求时，您的主体可能会包含一组标签。这些标签来自以下来源：

1. 会话标签 - 这些标签在您使用 Amazon CLI 或 Amazon API 代入角色或联合身份用户身份时传递。有关这些操作的更多信息，请参阅 [会话标记操作 \(p. 293\)](#)。
2. 传入的可传递会话标签 - 这些标签继承自角色链中的上一个会话。有关更多信息，请参阅本主题后面的 [使用会话标签链接角色 \(p. 299\)](#)。
3. IAM 标签 — 附加到您的 IAM 所担任角色的标签。

主题

- [会话标记操作 \(p. 293\)](#)
- [关于会话标签的需知信息 \(p. 294\)](#)
- [添加会话标签所需的权限 \(p. 295\)](#)
- [使用 AssumeRole 传递会话标签 \(p. 296\)](#)
- [使用 AssumeRoleWithSAML 传递会话标签 \(p. 297\)](#)
- [使用 AssumeRoleWithWebIdentity 传递会话标签 \(p. 298\)](#)
- [使用 GetFederationToken 传递会话标签 \(p. 298\)](#)
- [使用会话标签链接角色 \(p. 299\)](#)
- [将会话标签用于 ABAC \(p. 300\)](#)
- [在 CloudTrail 中查看会话标签 \(p. 300\)](#)

会话标记操作

您可以在 Amazon STS 中使用以下 Amazon CLI 或 Amazon API 操作来传递会话标签。Amazon Web Services Management Console [切换角色 \(p. 243\)](#) 功能不支持传递会话标签。

您还可以将会话标签设置为可传递。在角色链的串联过程中，可传递标签持续存在。有关更多信息，请参阅[使用会话标签链接角色 \(p. 299\)](#)。

传递会话标签的方法对比

运算	谁可以担任角色	传递标签的方法	设置可传递标签的方法
assume-role CLI 或 AssumeRole API 操作	IAM 用户或会话	<code>Tags</code> API 参数或 <code>--tags</code> CLI 选项	<code>TransitiveTagKeys</code> API 参数或 <code>--transitive-tag-keys</code> CLI 选项
assume-role-with-saml CLI 或 AssumeRoleWithSAML API 操作	使用 SAML 身份提供程序验证身份的任何用户	<code>PrincipalTag</code> SAML 属性	<code>TransitiveTagKeys</code> SAML 属性

运算	谁可以担任角色	传递标签的方法	设置可传递标签的方法
assume-role-with-web-identity CLI 或 AssumeRoleWithWebIdentity API 操作	使用 Web 身份提供程序验证的任何用户	PrincipalTag Web 身份令牌	TransitiveTagKeys Web 身份令牌
get-federation-token CLI 或 GetFederationToken API 操作	IAM 用户或根用户	Tags API 参数或 --tags CLI 选项	不支持

如果满足以下条件，则支持会话标记的操作会失败：

- 您传递的会话标签数超过 50。
- 会话标签键的纯文本超过 128 个字符。
- 会话标签值的纯文本超过 256 个字符。
- 会话策略纯文本的总大小超过 2048 个字符。
- 会话策略和会话标签的组合包总大小太大。如果操作失败，则错误消息以百分比形式指示策略和标签的组合接近大小上限的程度。

关于会话标签的需知信息

在您使用会话标签之前，请查看以下有关会话和标签的详细信息。

- 使用会话标签时，连接到传递标签的身份提供程序 (IdP) 的所有角色的信任策略都必须具有 [sts:TagSession](#) (p. 295) 权限。对于信任策略中没有此权限的角色，`AssumeRole` 操作会失败。
- 请求会话时，可以将主体标记指定为会话标记。标签适用于您使用会话凭证发出的请求。
- 会话标签使用键/值对。例如，要向会话添加联系信息，您可以添加会话标签键 `email` 和标签值 `johndoe@example.com`。
- 会话标签必须遵循在 [IAM 和 Amazon STS](#) (p. 277) 中命名标签的规则。本主题包括有关适用于会话标签的区分大小写和受限制前缀的信息。
- 新的会话标签覆盖具有相同标签键的现有已代入角色或联合身份用户身份标签，无论大小写如何。
- 您无法使用 Amazon Web Services Management Console 传递会话标签。
- 会话标签仅对当前会话有效。
- 会话标签支持 [角色链](#) (p. 159)。默认情况下，Amazon STS 不传递到后续角色会话。不过，您可以将会话标签设置为可传递。在角色链期间，可传递标签持续存在，在角色信任策略评估期间，将替换匹配 `ResourceTag` 的值。有关更多信息，请参阅 [使用会话标签链接角色](#) (p. 299)。
- 您可以使用会话标签来控制对资源的访问，或者控制哪些标签可以传递到后续会话。有关更多信息，请参阅 [IAM 教程：将 SAML 会话标签用于 ABAC](#) (p. 53)。
- 您可以在 Amazon CloudTrail 日志中查看会话的主体标签，包括其会话标签。有关更多信息，请参阅 [在 CloudTrail 中查看会话标签](#) (p. 300)。
- 您必须为每个会话标签传递一个值。Amazon STS 不支持多值会话标签。
- 您最多可以传递 50 个会话标签。Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 [IAM 与 Amazon STS 配额、名称要求和字符限制](#) (p. 902)。
- Amazon 转换会将传递的会话策略和会话标签合并压缩为具有单独限制的打包二进制格式。如果超过此限制，Amazon CLI 或 Amazon API 错误消息以百分比形式指示策略和标签的组合接近大小上限的程度。

添加会话标签所需的权限

除了与 API 操作匹配的操作之外，您的策略中还必须具有以下仅限授权执行的操作：

```
sts:TagSession
```

Important

使用会话标签时，连接到身份提供程序 (IdP) 的所有角色的角色信任策略都必须具有 `sts:TagSession` 权限。对于连接到 IdP 的任何角色，如果在没有此权限的情况下传递会话标签，`AssumeRole` 操作将失败。如果您不想更新每个角色的角色信任策略，则可以使用单独的 IdP 实例传递会话标签。然后仅将 `sts:TagSession` 权限添加到连接至单独 IdP 的角色。

您可以将 `sts:TagSession` 操作与以下条件键搭配使用。

- [aws:PrincipalTag \(p. 1005\)](#) – 使用此键可将附加到发出请求的主体的标签与您在策略中指定的标签进行比较。例如，您可以仅在发出请求的主体具有指定的标签时，才允许主体传递会话标签。
- [aws:RequestTag \(p. 1007\)](#) – 使用此键可将请求中传递的标签键/值对与您在策略中指定的标签对进行比较。例如，您可以允许主体传递指定的会话标签，但只能使用指定的值。
- [aws:ResourceTag \(p. 1008\)](#) – 使用此键可将您在策略中指定的标签键/值对与附加到资源的键/值对进行比较。例如，您可以仅在当主体代入的角色包含指定标签时，才允许主体传递会话标签。
- [aws:TagKeys \(p. 1011\)](#) – 将请求中的标签键与您在策略中指定的键进行比较。例如，您可以只允许主体传递具有指定标签键的会话标签。此条件键限制可传递的最大一组会话标签。
- [sts:TransitiveTagKeys \(p. 1025\)](#) – 将请求中的可传递会话标签键与在策略中指定的键进行比较。例如，您可以编写一个策略，仅允许主体将特定标签设置为可传递标签。在角色链的串联过程中，可传递标签持续存在。有关更多信息，请参阅[使用会话标签链接角色 \(p. 299\)](#)。

例如，以下[角色信任策略 \(p. 160\)](#)允许 `test-session-tags` 用户代入策略附加到的角色。用户在代入该角色时，必须使用 Amazon CLI 或 Amazon API 传递三个所需的会话标签和所需的外部 ID ([p. 164](#))。此外，用户可以选择将 `Project` 和 `Department` 标签设置为可传递。

Example 会话标签的角色信任策略示例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowIamUserAssumeRole",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {"AWS": "arn:aws:iam::123456789012:user/test-session-tags"},
      "Condition": {
        "StringLike": {
          "aws:RequestTag/Project": "*",
          "aws:RequestTag/CostCenter": "*",
          "aws:RequestTag/Department": "*"
        },
        "StringEquals": {"sts:ExternalId": "Example987"}
      }
    },
    {
      "Sid": "AllowPassSessionTagsAndTransitive",
      "Effect": "Allow",
      "Action": "sts:TagSession",
      "Principal": {"AWS": "arn:aws:iam::123456789012:user/test-session-tags"},
      "Condition": {
        "StringLike": {
          "aws:RequestTag/Project": "*",
          "aws:RequestTag/CostCenter": "*"
        }
      }
    }
  ]
}
```

```
    },
    "StringEquals": {
      "aws:RequestTag/Department": [
        "Engineering",
        "Marketing"
      ]
    },
    "ForAllValues:StringEquals": {
      "sts:TransitiveTagKeys": [
        "Project",
        "Department"
      ]
    }
  }
}
```

此策略有何作用？

- AllowIamUserAssumeRole 语句允许 test-session-tags 用户代入策略所附加到的角色。用户在代入该角色时，必须传递所需的会话标签和外部 ID (p. 164)。
- 此语句的第一个条件块要求用户传递 Project、CostCenter 和 Department 会话标签。标签值在此语句中无关紧要，因此我们为标签值使用通配符 (*)。此块确保用户至少传递这三个会话标签。否则，该操作将失败。用户可以传递其他标签。
- 第二个条件块要求用户传递值为 Example987 的外部 ID (p. 164)。
- 该 AllowPassSessionTagsAndTransitive 语句允许 sts:TagSession 仅限授权执行的操作。必须允许此操作，然后用户才能传递会话标签。如果您的策略包含第一个语句而没有第二个语句，则用户无法代入角色。
- 此语句的第一个条件块允许用户为 CostCenter 和 Project 会话标签传递任何值。您可以通过在策略中为标签值使用通配符 (*) 来执行此操作，这需要使用 StringLike (p. 951) 条件运算符。
- 第二个条件块仅允许用户为 Department 会话标签传递 Engineering 或 Marketing 值。
- 第三个条件块列出可以设置为可传递的最大一组标签。用户可以选择将一部分标签设置为可传递，也可以不将任何标签设置为可传递。他们不能将其他标签设置为可传递。您可以通过添加包含 "Null": {"sts:TransitiveTagKeys": "false"} 的其他条件块，将至少一个标签设置为可传递。

使用 AssumeRole 传递会话标签

AssumeRole 操作返回一组可用于访问 Amazon 资源的临时凭证。您可以使用 IAM 用户或角色凭证来调用 AssumeRole。要在代入角色时传递会话标签，请使用 --tags Amazon CLI 选项或 Tags Amazon API 参数。

要将标签设置为可传递，请使用 --transitive-tag-keys Amazon CLI 选项或 TransitiveTagKeys Amazon API 参数。在角色链的串联过程中，可传递标签持续存在。有关更多信息，请参阅[使用会话标签链接角色 \(p. 299\)](#)。

以下示例显示使用 AssumeRole 的示例请求。在此示例中，在代入 my-role-example 角色时，您将创建一个名为 my-session 的会话。添加会话标签键/值对 Project = Automation、CostCenter = 12345 和 Department = Engineering。您还可以通过指定 Project 和 Department 标签的键，将其设置为可传递标签。

Example 示例 AssumeRole CLI 请求

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/my-role-example \
--role-session-name my-session \
--tags Key=Project,Value=Automation Key=CostCenter,Value=12345
Key=Department,Value=Engineering \
```

```
--transitive-tag-keys Project Department \  
--external-id Example987
```

使用 AssumeRoleWithSAML 传递会话标签

AssumeRoleWithSAML 操作使用基于 SAML 的联合身份进行身份验证。此操作返回一组可用于访问 Amazon 资源的临时凭证。有关将基于 SAML 的联合身份用于 Amazon Web Services Management Console 访问的更多信息，请参阅[使用 SAML 2.0 联合身份用户能够访问 Amazon Web Services Management Console](#) (p. 199)。有关 Amazon CLI 或 Amazon API 访问的详细信息，请参阅[关于基于 SAML 2.0 的联合身份验证](#) (p. 176)。有关为 Active Directory 用户设置 SAML 联合身份的教程，请参阅 Amazon 安全博客中的[使用 Active Directory 联合身份验证服务 \(ADFS\) 的 Amazon 联合身份验证](#)。

作为管理员，您可以使用 Amazon STS AssumeRoleWithSAML 操作，允许公司目录的成员联合身份到 Amazon 中。要执行此操作，您必须完成以下任务：

1. 将网络配置为适用于 Amazon 的 SAML 提供商 (p. 192)。
2. 在 IAM 中创建 SAML 提供商 (p. 188)
3. 在 Amazon 中为联合身份用户配置角色及其权限 (p. 231)
4. 完成配置 SAML IdP 并为 SAML 身份验证响应创建断言 (p. 194)

Amazon 包括身份提供程序，通过其身份解决方案提供经认证的端到端会话标签体验。要了解如何使用这些身份提供程序配置会话标签，请参阅[将第三方 SAML 解决方案提供商与 Amazon 集成](#) (p. 192)。

要将 SAML 属性作为会话标签传递，请包含 Attribute 元素并将 Name 属性设置为 `https://aws.amazon.com/SAML/Attributes/PrincipalTag:{TagKey}`。使用 AttributeValue 元素指定标签的值。为每个会话标签包含一个单独的 Attribute 元素。

例如，假设您要将以下身份属性作为会话标记传递：

- Project:Automation
- CostCenter:12345
- Department:Engineering

要传递这些属性，请在 SAML 断言中包含以下元素。

Example SAML 断言的示例片段

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Project">  
  <AttributeValue>Automation</AttributeValue>  
</Attribute>  
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:CostCenter">  
  <AttributeValue>12345</AttributeValue>  
</Attribute>  
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Department">  
  <AttributeValue>Engineering</AttributeValue>  
</Attribute>
```

要将上述标签设置为可传递，请包含另一个 Attribute 元素并将 Name 属性设置为 `https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys`。在角色链的串联过程中，可传递标签持续存在。有关更多信息，请参阅[使用会话标签链接角色](#) (p. 299)。

要将 Project 和 Department 标签设置为可传递，请使用以下多值属性。

Example SAML 断言的示例片段

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys">
```

```
<AttributeValue>Project</AttributeValue>
<AttributeValue>Department</AttributeValue>
</Attribute>
```

使用 AssumeRoleWithWebIdentity 传递会话标签

使用 OpenID Connect (OIDC) 的 Web 联合身份进行 AssumeRoleWithWebIdentity 操作的身份验证。此操作返回一组可用于访问 Amazon 资源的临时凭证。有关将基于 Web 联合身份用于 Amazon Web Services Management Console 访问的更多信息，请参阅[关于 Web 联合身份验证 \(p. 171\)](#)。

要从 OpenID Connect (OIDC) 传递会话标签，您必须在 JSON Web 令牌 (JWT) 中包含会话标签。当您提交 AssumeRoleWithWebIdentity 请求时，请在令牌的 <http://www.amazonaws.cn/tags> 命名空间中包含会话标签。要了解有关 OIDC 令牌和声明的更多信息，请参阅《Amazon Cognito 开发人员指南》中的[将令牌与用户群体结合使用](#)。

例如，以下解码的 JWT 是一个令牌，用于通过 Project、CostCenter 和 Department 会话标签调用 AssumeRoleWithWebIdentity。该令牌还将 Project 和 CostCenter 标签设置为可传递。在角色链的串联过程中，可传递标签持续存在。有关更多信息，请参阅[使用会话标签链接角色 \(p. 299\)](#)。

Example 解码的 JSON Web 令牌示例

```
{
  "sub": "johndoe",
  "aud": "ac_oic_client",
  "jti": "ZYUCeRMQVtqHypVPWAN3VB",
  "iss": "https://xyz.com",
  "iat": 1566583294,
  "exp": 1566583354,
  "auth_time": 1566583292,
  "https://aws.amazon.com/tags": {
    "principal_tags": {
      "Project": ["Automation"],
      "CostCenter": ["987654"],
      "Department": ["Engineering"]
    },
    "transitive_tag_keys": [
      "Project",
      "CostCenter"
    ]
  }
}
```

使用 GetFederationToken 传递会话标签

GetFederationToken 允许您对用户进行联合身份验证。此操作返回一组可用于访问 Amazon 资源的临时凭证。要将标签添加到联合身份用户会话，请使用 `--tags` Amazon CLI 选项或 `Tags` Amazon API 参数。在使用 GetFederationToken 时，您不能将会话标签设置为可传递，因为您无法使用临时凭证担任角色。在这种情况下，您不能使用角色链。

下面的示例演示使用 GetFederationToken 的示例请求。在此示例中，在请求令牌时，您创建一个名为 `my-fed-user` 的会话。添加会话标签键/值对 `Project = Automation` 和 `Department = Engineering`。

Example GetFederationToken CLI 请求示例

```
aws sts get-federation-token \
--name my-fed-user \
--tags key=Project,value=Automation key=Department,value=Engineering
```

当您使用 `GetFederationToken` 操作返回的临时凭证时，会话的主体标签包括用户的标签以及传递的会话标签。

使用会话标签链接角色

您可以代入一个角色，然后使用临时凭证代入另一个角色。您可以从一个会话继续到另一个会话来使用角色。这就是所说的[角色链](#) (p. 159)。当您在代入某个角色的同时传递会话标签，您可以将键设置为可传递。这可确保将这些会话标签传递到角色链中的后续会话。您无法将角色标签设置为可传递。要将这些标签传递给后续会话，请将其指定为会话标签。

以下示例帮助您了解 Amazon STS 如何在角色链中，将会话标签、可传递标签和角色标签传递到后续会话。

在以下示例角色链场景中，您在 Amazon CLI 中使用 IAM 用户的访问密钥代入名为 `Role1` 的角色。然后，您可以使用生成的会话凭证代入名为 `Role2` 的第二个角色。接下来，您可以使用第二个会话凭证代入名为 `Role3` 的第三个角色。这些请求作为三个单独的操作发生。每个角色均已在 IAM 中标记。在每个请求过程中，您都会传递额外的会话标签。

链接角色时，您可以确保来自较早会话的标签保留到后续会话。要使用 `assume-role` CLI 命令执行此操作，您必须将标签作为会话标签传递，并将标签设置为可传递。您传递标签 `Star = 1` 作为会话标签。此命令还将标签 `Heart = 1` 附加到角色，并在您使用会话时作为主体标签应用。但是，您还希望自动将 `Heart = 1` 标签传递给第二个或第三个会话。为此，您可以手动将其包含为会话标签。生成的会话主体标记包括这两个标记，并将它们设置为可传递。

您可以使用以下 Amazon CLI 命令执行此请求：

Example 示例 AssumeRole CLI 请求

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Role1 \
--role-session-name Session1 \
--tags Key=Star,Value=1 Key=Heart,Value=1 \
--transitive-tag-keys Star Heart
```

然后，您可以使用该会话的凭证代入 `Role2`。此命令将标签 `Sun = 2` 附加到第二个角色，并在您使用第二个会话时作为主体标签应用。`Heart` 和 `Star` 标签继承第一个会话中的可传递会话标签。第二个会话产生的主体标签为 `Heart = 1`、`Star = 1` 和 `Sun = 2`。`Heart` 和 `Star` 将继续为可传递标签。附加到 `Role2` 的 `Sun` 标签不会标记为可传递，因为它不是会话标签。以后的会话不继承此标签。

您可以使用以下 Amazon CLI 命令执行第二个请求：

Example 示例 AssumeRole CLI 请求

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Role2 \
--role-session-name Session2
```

然后，您使用第二个会话凭证来代入 `Role3`。第三个会话的主体标签来自任意新的会话标签、继承的可传递会话标签和角色标签。第二个会话上的 `Heart = 1` 和 `Star = 1` 标签继承自第一个会话中的可传递会话标签。如果您尝试传递 `Heart = 3` 会话标记，操作会失败。继承的 `Star = 1` 会话标签覆盖角色的 `Star = 3` 标签。在角色链中，可传递标记的值将覆盖与 `ResourceTag` 值匹配的角色，然后再评估角色信任策略。在此示例中，如果 `Role3` 在角色信任策略中将 `Star` 用作 `ResourceTag`，并将 `ResourceTag` 值设置为调用角色会话的可传递标签值。角色的 `Lightning` 标签也应用到第三个会话，并且不设置为可传递。

您使用以下 Amazon CLI 命令执行第三个请求：

Example 示例 AssumeRole CLI 请求

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Role3 \
```

```
--role-session-name Session3
```

将会话标签用于 ABAC

基于属性的访问控制 (ABAC) 是一种授权策略，该策略基于标签属性来定义权限。

如果您的公司使用 OIDC 基于 SAML 的身份提供程序 (IdP) 管理用户实体，则可以配置 SAML 断言，将会话标记传递给 Amazon。例如，对于公司用户身份，当您的员工联合到 Amazon，Amazon 会将其属性应用于其生成的主体。然后，您可以使用 ABAC 来允许或拒绝基于这些属性的权限。有关详细信息，请参阅 [IAM 教程：将 SAML 会话标签用于 ABAC \(p. 53\)](#)。

有关将 IAM Identity Center 与 ABAC 配合使用的更多信息，请参阅《Amazon IAM Identity Center (successor to Amazon Single Sign-On) 用户指南》中的 [访问控制的属性](#)。

在 CloudTrail 中查看会话标签

您可以使用 Amazon CloudTrail 查看为代入角色或联合身份用户而发出的请求。CloudTrail 日志文件包括有关所代入角色或联合身份用户会话的主体标签的信息。有关更多信息，请参阅 [使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用 \(p. 340\)](#)。

例如，假定您发出 Amazon STS AssumeRoleWithSAML 请求，传递会话标签，然后将这些标签设置为可传递标签。您可以在 CloudTrail 日志中找到以下信息。

Example AssumeRoleWithSAML CloudTrail 日志示例

```
"requestParameters": {
  "SAMLAssertionID": "_c0046cEXAMPLEb9d4b8eEXAMPLE2619aEXAMPLE",
  "roleSessionName": "MyRoleSessionName",
  "principalTags": {
    "CostCenter": "987654",
    "Project": "Unicorn"
  },
  "transitiveTagKeys": [
    "CostCenter",
    "Project"
  ],
  "durationSeconds": 3600,
  "roleArn": "arn:aws:iam::123456789012:role/SAMLEstTestRoleShibboleth",
  "principalArn": "arn:aws:iam::123456789012:saml-provider/Shibboleth"
},
```

您可以查看以下示例 CloudTrail 日志以查看使用会话标签的事件。

- [CloudTrail 日志文件中 Amazon STS 角色链接 API 事件示例 \(p. 346\)](#)
- [CloudTrail 日志文件中 SAML Amazon STS API 事件的示例 \(p. 349\)](#)
- [CloudTrail 日志文件中的 Web 身份 Amazon STS API 事件示例 \(p. 350\)](#)

IAM 临时安全凭证

您可以使用 Amazon Security Token Service (Amazon STS) 创建可控制对您的 Amazon 资源的访问的临时安全凭证，并将这些凭证提供给受信任用户。临时安全凭证的工作方式与您的 IAM 用户可使用的长期访问密钥凭证的工作方式几乎相同，仅存在以下差异：

- 顾名思义，临时安全凭证是短期凭证。可将这些凭证的有效时间配置几分钟到几小时。一旦这些凭证到期，Amazon 将不再识别这些凭证或不再允许来自使用这些凭证发出的 API 请求的任何类型的访问。
- 临时安全凭证不随用户一起存储，而是动态生成并在用户提出请求时提供给用户。临时安全凭证到期时（甚至之前），用户可以请求新的凭证，只要请求凭证的用户仍有权这样做。

这些差异使得可利用临时凭证获得以下优势：

- 您不必随应用程序分配或嵌入长期 Amazon 安全凭证。
- 可允许用户访问您的 Amazon 资源，而不必为这些用户定义 Amazon 身份。临时凭证是[角色和联合身份验证 \(p. 158\)](#)的基础。
- 临时安全凭证的使用期限有限，因此，在不需要这些凭证时不必轮换或显式撤销这些凭证。临时安全凭证到期后无法重复使用。您可指定凭证的有效期，但有最长期限。

Amazon STS 和 Amazon 区域

临时安全凭证由 Amazon STS 生成。默认情况下，Amazon STS 是一种全球服务，在 <https://sts.amazonaws.com> 上具有单个终端节点。不过，您也可以选择对任何其他支持的区域中的终端节点进行 Amazon STS API 调用。这可能会将请求发送到地理位置离您较近的区域中的服务器以减少延迟（服务器延迟）。无论您的凭证来自于哪个区域，它们都会在全球范围内起作用。有关更多信息，请参阅[在 Amazon 区域中管理 Amazon STS \(p. 331\)](#)。

临时凭证的常见情形

临时凭证在涉及联合身份验证、委派、跨账户访问和 IAM 角色的情形中很有用。

联合身份

您可以在 Amazon 之外的外部系统中管理用户身份，并为从这些系统登录的用户授予访问权限以执行 Amazon 任务和访问您的 Amazon 资源。IAM 支持两种联合身份验证。在两种情况下，身份都存储在 Amazon 外部。区别在于外部系统所处的位置 在您的数据中心内或 Web 上的外部第三方。有关外部身份提供程序的更多信息，请参阅[身份提供程序和联合身份验证 \(p. 170\)](#)。

- Enterprise identity federation (企业联合身份认证) - 您可以对您的企业网络中的用户进行身份验证，然后为这些用户提供对 Amazon 的访问权限，而无需为其创建新的 Amazon 身份并要求他们使用单独的用户名和密码进行登录。这称为用于临时访问的单点登录方法。Amazon STS 支持开放标准 [如安全断言标记语言 (SAML) 2.0]，这使您可以使用 Microsoft AD FS 来利用您的 Microsoft Active Directory。您还可以使用 SAML 2.0 管理自己的联合身份用户身份解决方案。有关更多信息，请参阅[关于基于 SAML 2.0 的联合身份验证 \(p. 176\)](#)。
- 自定义联合身份代理 - 您可以使用您企业的身份验证系统来授予对 Amazon 资源的访问权限。有关示例方案，请参阅[使自定义身份代理能够访问 Amazon 控制台 \(p. 201\)](#)。
- 使用 SAML 2.0 的联合 - 您可以使用您企业的身份验证系统和 SAML 来授予对 Amazon 资源的访问权限。有关更多信息以及示例方案，请参阅[关于基于 SAML 2.0 的联合身份验证 \(p. 176\)](#)。
- Web 联合身份验证 - 您可以让用户使用已知的第三方身份提供程序（例如，Login with Amazon、Facebook、Google 或任何 OpenID Connect (OIDC) 2.0 兼容的提供商）登录。您可以交换来自这些提供商的凭证，以获取对您的 Amazon 账户中的资源的临时使用权限。这称为用于临时访问的 Web 联合身份验证方法。当您移动应用程序或 Web 应用程序使用 Web 联合身份验证时，您不需要创建自定义登录代码或管理您自己的用户身份。使用 Web 联合身份验证可帮助您确保 Amazon 账户的安全，因为您不必随应用程序分配长期安全凭证（如 IAM 用户访问密钥）。有关更多信息，请参阅[关于 Web 联合身份验证 \(p. 171\)](#)。

Amazon STS Web 联合身份验证支持 Login with Amazon、Facebook、Google 和任何 OpenID Connect (OIDC) 兼容的身份提供程序。

Note

有关移动应用程序，我们建议使用 Amazon Cognito。您可以将此服务与 [Amazon Mobile SDK for iOS](#) 和 [Amazon Mobile SDK for Android](#) 搭配使用，为用户创建唯一的身份，并对其进行身份验证，以使其安全地访问您的 Amazon 资源。Amazon Cognito 支持与 Amazon STS 相同的身份提供程序，还支持未经身份验证的（来宾）访问，并允许您在用户登录时迁移用户数据。Amazon

Cognito 还提供用于同步用户数据的 API 操作，因此，无论用户在设备间怎样转移，其数据总得以保留。有关更多信息，请参阅下列内容：

- 《Amazon Mobile SDK for iOS 开发人员指南》中的[使用现有 Amazon Cognito 资源](#)
- 《Amazon Mobile SDK for Android 开发人员指南》中的[使用现有 Amazon Cognito 资源](#)

用于跨账户访问的角色

很多组织维护多个 Amazon 账户。利用角色和跨账户访问，您可以在一个账户中定义用户身份，并使用这些身份访问属于您组织的其他账户中的 Amazon 资源。这称为用于临时访问的委派方法。有关创建跨账户角色的更多信息，请参阅[创建向 IAM 用户委派权限的角色 \(p. 215\)](#)。要了解您信任区域之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅[什么是 IAM Access Analyzer？](#)

适用于 Amazon EC2 的角色

如果您在 Amazon EC2 实例上运行一些应用程序，并且这些应用程序需要访问 Amazon 资源，则您可在启动您的实例时为其提供临时安全凭证。这些临时安全凭证对该实例上运行的所有应用程序都可用，因此您无需在该实例上存储任何长期凭证。有关更多信息，请参阅[使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限 \(p. 252\)](#)。

其他 Amazon 服务

可使用临时安全凭证访问大多数 Amazon 服务。有关接受临时安全凭证的服务的列表，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)。

请求临时安全凭证

要请求临时安全凭证，您可以在 Amazon API 中使用 Amazon Security Token Service (Amazon STS) 操作。这些操作包括创建受信任用户，并为其提供可以控制 Amazon 资源访问的临时安全凭证。有关 Amazon STS 的更多信息，请参阅[IAM 临时安全凭证 \(p. 300\)](#)。要了解在担任角色以请求临时安全凭证时使用的各种方法，请参阅[使用 IAM 角色 \(p. 236\)](#)。

要调用 API 操作，您可以使用其中的一个 [Amazon SDK](#)。这些开发工具包适用于各种不同的编程语言和环境，包括 Java、.NET、Python、Ruby、Android 和 iOS。这些开发工具包负责处理各种任务，如以加密方式对您的请求进行签名、在必要时重试请求以及处理错误响应。还可使用 Amazon STS 查询 API（在[Amazon Security Token Service API 参考](#)中介绍）。最后，两个命令行工具支持 Amazon STS 命令：[Amazon Command Line Interface](#)和[Amazon Tools for Windows PowerShell](#)。

Amazon STS API 操作使用临时安全凭证（包括访问密钥对和会话令牌）创建新会话。访问密钥对由访问密钥 ID 和私有密钥组成。用户（或用户所运行的应用程序）可使用这些凭证访问您的资源。您可以使用 Amazon STS API 操作以编程方式创建角色会话和传递会话策略及会话标签。生成的会话的权限是角色的基于身份的策略与会话策略的交集。有关会话策略的更多信息，请参阅[会话策略 \(p. 357\)](#)。有关会话标签的更多信息，请参阅[在 Amazon STS 中传递会话标签 \(p. 293\)](#)。

Note

Amazon STS API 操作返回的会话令牌大小不固定。我们强烈建议不要假设最大大小。典型的令牌大小小于 4096 字节，但可能会发生变化。

通过 Amazon 区域使用 Amazon STS

您可以将 Amazon STS API 调用发送到全球终端节点或某个区域终端节点。如果您选择更靠近您的终端节点，则可减少延迟并改善 API 调用的性能。如果您不再能与原始终端节点进行通信，也可选择将调用定向到替代的区域终端节点。如果您使用某种 Amazon SDK，请先使用该开发工具包的方法指定一个区域，然后再进行 API 调用。如果您正在手动构建 HTTP API 请求，则必须自行将请求定向到正确的终端节点。有关更多信息，请参阅[区域和终端节点](#)和在[Amazon 区域中管理 Amazon STS \(p. 331\)](#)的 Amazon STS 部分。

下面是让您可以用来获取临时凭证以用于您的 Amazon 环境和应用程序的 API 操作。

AssumeRole – 通过自定义身份凭证代理程序进行跨账户委托和联合身份验证

AssumeRole API 操作对于允许现有 IAM 用户访问其没有访问权限的 Amazon 资源很有用。例如，用户可能需要对其他 Amazon 账户中资源的访问权限。此外，可使用它来暂时获得特权访问权限 - 例如，提供多重身份验证 (MFA)。您必须使用活动凭证调用该 API。要了解谁可以调用此操作，请参阅 [比较 Amazon STS API 操作 \(p. 310\)](#)。有关更多信息，请参阅 [创建向 IAM 用户委派权限的角色 \(p. 215\)](#) 和 [配置受 MFA 保护的 API 访问 \(p. 129\)](#)。

必须使用有效的 Amazon 安全凭证来进行此调用。进行此调用时，您传递以下信息：

- 应用程序应承担的角色的 Amazon Resource Name (ARN)。
- (可选) 持续时间，它指定临时安全凭证的持续时间。可以使用 DurationSeconds 参数指定 900 秒 (15 分钟) 到角色的最大会话持续时间设置之间的角色会话持续时间。要了解如何查看您的角色的最大值，请参阅 [查看角色的最大会话持续时间设置 \(p. 238\)](#)。如果未传递该参数，临时凭证将在 1 小时后过期。该 API 中的 DurationSeconds 参数与用于指定控制台会话持续时间的 SessionDuration HTTP 参数不同。可以在发送到联合终端节点的控制台登录令牌请求中使用 SessionDuration HTTP 参数。有关更多信息，请参阅 [使自定义身份代理能够访问 Amazon 控制台 \(p. 201\)](#)。
- 角色会话名称 使用此字符串值可在不同主体使用角色时标识会话。为了安全起见，管理员可以在 [Amazon CloudTrail 日志 \(p. 343\)](#) 中查看此字段，以帮助识别已在 Amazon 中执行操作的人员。您的管理员可能会要求您在代入角色时指定 IAM 用户名作为会话名称。有关更多信息，请参阅 [sts:RoleSessionName \(p. 1023\)](#)。
- (可选) 源身份。您可以要求用户在担任角色时指定源身份。设置源身份后，无法更改该值。它在角色会话期间执行的所有操作的请求中都会出现。源身份值在 [链接角色 \(p. 159\)](#) 会话中持续存在。您可以将源身份信息用于 Amazon CloudTrail 日志来确定谁使用角色采取了操作。有关使用基于身份的策略的更多信息，请参阅 [监控和控制使用所担任角色执行的操作 \(p. 316\)](#)。
- (可选) 内联或托管会话策略。这些策略限制角色的基于身份的策略中分配给角色会话的权限。生成的会话的权限是角色的基于身份的策略与会话策略的交集。使用会话策略授予的权限不能超过担任的角色的基于身份的策略允许的权限。有关角色会话权限的更多信息，请参阅 [会话策略 \(p. 357\)](#)。
- (可选) 会话标签。您可以代入角色，然后使用临时凭证发出请求。执行此操作时，会话的主体标签包括角色的标签和传递的会话标签。如果您使用临时凭证发出此调用，则新会话还会从调用会话继承可传递会话标签。有关会话标签的更多信息，请参阅 [在 Amazon STS 中传递会话标签 \(p. 293\)](#)。
- (可选) MFA 信息。如果配置为使用 Multi-Factor Authentication (MFA)，则可以包含 MFA 设备的标识符和该设备提供的一次性代码。
- (可选) 一个可在将对您账户的访问权限委派给第三方时使用的可选 ExternalId 值。该值有助于确保仅指定的第三方可以访问该角色。有关更多信息，请参阅 [如何在向第三方授予对 Amazon 资源的访问权时使用外部 ID \(p. 164\)](#)。

以下示例显示了使用 AssumeRole 的示例请求和响应。此示例请求将在指定的持续时间内担任 demo 角色，其中包含 [会话策略 \(p. 357\)](#)、[会话标签 \(p. 293\)](#)、[外部 ID \(p. 164\)](#) 和 [源身份 \(p. 316\)](#)。生成的会话名为 John-session。

Example 示例请求

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=AssumeRole
&RoleSessionName=John-session
&RoleArn=arn:aws::iam::123456789012:role/demo
&Policy=%7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A%5B%7B%22Sid%22%3A%20%22Stmnt1%22%2C%22Effect%22%3A%20%22Allow%22%2C%22Action%22%3A%20%22s3%3A%22%2C%22Resource%22%3A%20%22%2A%22%7D%5D%7D
&DurationSeconds=1800
&Tags.member.1.Key=Project
&Tags.member.1.Value=Pegasus
```

```
&Tags.member.2.Key=Cost-Center
&Tags.member.2.Value=12345
&ExternalId=123ABC
&SourceIdentity=DevUser123
&AUTHPARAMS
```

在前面的示例中显示的策略值是以下策略的 URL 编码版本：

```
{ "Version": "2012-10-17", "Statement":
[ { "Sid": "Stmt1", "Effect": "Allow", "Action": "s3:*", "Resource": "*" } ] }
```

示例中的 AUTHPARAMS 参数是您的签名的占位符。签名是您必须在 Amazon HTTP API 请求中包含的身份验证信息。建议使用 [Amazon 开发工具包](#) 创建 API 请求，这样做的一个好处是开发工具包将为您处理请求签名。如果您必须手动创建并签署 API 请求，请参阅 Amazon Web Services 一般参考中的 [使用签名版本 4 签署 Amazon 的请求](#) 以了解如何签署请求。

除了临时安全凭证之外，该响应还包括联合身份用户的 Amazon Resource Name (ARN) 和凭证的到期时间。

Example 响应示例

```
<AssumeRoleResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
  <AssumeRoleResult>
    <SourceIdentity>DevUser123</SourceIdentity>
    <Credentials>
      <SessionToken>
        AQoDYXdzEPT////////wEXAMPLEtc764bNrc9SAPBSM22wDok4x4HIZ8j4FZTwDQW
        LWsKWHGBuFqwAeMicRXmxfpSPfIeoIYRqTflfKD8YUuwthAx7mSEI/qkPpKPi/kMcGd
        QrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDy0KPkyQDYwT7WZ0wq5VSXDvp75YU
        9HFv1Rd8Tx6q6fE8YQcHNvXakiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
        +scqKmlzm8FDrypNC9Yjc8fPOLn9FX9KSYvKTr4rvx3iS1lTJabIQwj2ICCR/oLxBA==
      </SessionToken>
      <SecretAccessKey>
        wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY
      </SecretAccessKey>
      <Expiration>2019-07-15T23:28:33.359Z</Expiration>
      <AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
    </Credentials>
    <AssumedRoleUser>
      <Arn>arn:aws:sts::123456789012:assumed-role/demo/John</Arn>
      <AssumedRoleId>ARO123EXAMPLE123:John</AssumedRoleId>
    </AssumedRoleUser>
    <PackedPolicySize>8</PackedPolicySize>
  </AssumeRoleResult>
  <ResponseMetadata>
    <RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
  </ResponseMetadata>
</AssumeRoleResponse>
```

Note

Amazon 转换会将传递的会话策略和会话标签压缩为具有单独限制的打包二进制格式。即使您的纯文本符合其他要求，您的请求也可能会由于此限制而失败。PackedPolicySize 响应元素指示您请求的策略和标签接近大小上限的程度，以百分比来表示。

AssumeRoleWithWebIdentity – 通过基于 Web 的身份提供程序进行联合身份验证

AssumeRoleWithWebIdentity API 操作为通过公共身份提供程序验证的联合身份用户返回一组临时安全凭证。公共身份提供程序示例包括 Login with Amazon、Facebook、Google 或任何 OpenID Connect (OIDC) 兼容身份提供程序。要创建需要访问 Amazon 的移动应用程序或基于客户端的 Web 应用程序，该操

作是非常有用的。使用该操作意味着，您的用户不需要使用自己的 Amazon 或 IAM 身份。有关更多信息，请参阅[关于 Web 联合身份验证 \(p. 171\)](#)。

我们建议您使用 Amazon Cognito 和适用于移动开发的 Amazon SDK 附带的 Amazon Cognito 凭证提供程序，而不是直接调用 `AssumeRoleWithWebIdentity`。有关更多信息，请参阅下列内容：

- Amazon Mobile SDK for Android 开发人员指南中的 [Amazon Cognito 身份](#)
- Amazon Mobile SDK for iOS 开发人员指南中的 [Amazon Cognito 身份](#)

如果不使用 Amazon Cognito，则可调用 Amazon STS 的 `AssumeRoleWithWebIdentity` 操作。这是未签名的调用，这意味着应用程序无需使用任何 Amazon 安全凭证即可进行调用。进行此调用时，您传递以下信息：

- 应用程序应承担的角色的 Amazon Resource Name (ARN)。如果应用程序支持多种用户登录方式，则您必须定义多个角色，每个身份提供程序对应一个角色。调用 `AssumeRoleWithWebIdentity` 时应加入用户通过其登录的提供商所特有的角色的 ARN。
- 应用程序验证用户身份后从 IdP 获得的令牌。
- 您可以配置 IdP，将属性作为[会话标签 \(p. 293\)](#)传递到令牌。
- (可选) 持续时间，它指定临时安全凭证的持续时间。可以使用 `DurationSeconds` 参数指定 900 秒 (15 分钟) 到角色的最大会话持续时间设置之间的角色会话持续时间。要了解如何查看您的角色的最大值，请参阅[查看角色的最大会话持续时间设置 \(p. 238\)](#)。如果未传递该参数，临时凭证将在 1 小时后过期。该 API 中的 `DurationSeconds` 参数与用于指定控制台会话持续时间的 `SessionDuration` HTTP 参数不同。可以在发送到联合终端节点的控制台登录令牌请求中使用 `SessionDuration` HTTP 参数。有关更多信息，请参阅[使自定义身份代理能够访问 Amazon 控制台 \(p. 201\)](#)。
- 角色会话名称 使用此字符串值可在不同主体使用角色时标识会话。为了安全起见，管理员可以在 [Amazon CloudTrail 日志 \(p. 343\)](#) 中查看此字段，以了解已在 Amazon 中执行操作的人员。您的管理员可能会要求您在代入角色时为会话名称提供特定的值。有关更多信息，请参阅[sts:RoleSessionName \(p. 1023\)](#)。
- (可选) 源身份。您可以要求联合身份用户在担任角色时指定源身份。设置源身份后，无法更改该值。它在角色会话期间执行的所有操作的请求中都会出现。源身份值在[链接角色 \(p. 159\)](#)会话中持续存在。您可以将源身份信息用于 Amazon CloudTrail 日志来确定谁使用角色采取了操作。有关使用基于身份的策略的更多信息，请参阅[监控和控制使用所担任角色执行的操作 \(p. 316\)](#)。
- (可选) 内联或托管会话策略。这些策略限制角色的基于身份的策略中分配给角色会话的权限。生成的会话的权限是角色的基于身份的策略与会话策略的交集。使用会话策略授予的权限不能超过担任的角色的基于身份的策略允许的权限。有关角色会话权限的更多信息，请参阅[会话策略 \(p. 357\)](#)。

Note

对 `AssumeRoleWithWebIdentity` 的调用未签名 (加密)。因此，只有在通过受信任的中介传输请求时，才应包含可选的会话策略。在这种情况下，有人可能会修改策略以删除限制。

调用 `AssumeRoleWithWebIdentity` 时，Amazon 将验证令牌的真实性。例如，根据提供商的不同，Amazon 可能调用提供商并加入应用程序已传递的令牌。假定身份提供程序证实令牌有效，则 Amazon 返回以下信息：

- 一组临时安全凭证。这些证书由访问密钥 ID、秘密访问密钥和会话令牌组成。
- 所担任角色的角色 ID 和 ARN。
- 一个 `SubjectFromWebIdentityToken` 值，其中包含独一无二的用户 ID。

具有临时安全凭证时，可使用这些凭证调用 Amazon API。该过程与使用长期安全凭证进行 Amazon API 调用相同。不同之处在于，您必须包含会话令牌，以便 Amazon 验证临时安全凭证是否有效。

您的应用程序应缓存凭证。如上所述，默认情况下，凭证在 1 小时后到期。如果未使用 [软件开发工具包中的 AmazonSTSCredentialsProviderAmazon](#) 操作，则由您和您的应用程序负责再次调用

`AssumeRoleWithWebIdentity`。在旧安全凭证到期之前，可以调用该操作以获取一组新的临时安全凭证。

AssumeRoleWithSAML – 通过与 SAML 2.0 兼容的企业身份提供程序进行联合身份验证

`AssumeRoleWithSAML` API 操作为通过您的组织的现有身份系统验证的联合身份用户返回一组临时安全凭证。用户还必须使用 [SAML 2.0](#)（安全断言标记语言）将身份验证和授权信息传递给 Amazon。对于将身份系统（如 Windows Active Directory 或 OpenLDAP）与可生成 SAML 断言的软件集成在一起的组织，该 API 操作是非常有用的。此类集成提供有关用户身份和权限（如 Active Directory 联合身份验证服务或 Shibboleth）的信息。有关更多信息，请参阅[关于基于 SAML 2.0 的联合身份验证](#) (p. 176)。

Note

对 `AssumeRoleWithSAML` 的调用未签名（加密）。因此，只有在通过受信任的中介传输请求时，才应包含可选的会话策略。在这种情况下，有人可能会修改策略以删除限制。

这是未签名的调用，表示应用程序无需有权访问任何 Amazon 安全凭证即可进行调用。进行此调用时，您传递以下信息：

- 应用程序应承担的角色的 Amazon Resource Name (ARN)。
- IAM 中创建的 SAML 提供商的 ARN，用于描述身份提供程序。
- 采用 Base64 编码的 SAML 断言，这是 SAML 身份提供程序在针对您的应用程序的登录请求做出的身份验证响应中提供的。
- 您可以配置 IdP，将属性作为[会话标签](#) (p. 293) 传递到 SAML 断言。
- （可选）持续时间，它指定临时安全凭证的持续时间。可以使用 `DurationSeconds` 参数指定 900 秒（15 分钟）到角色的最大会话持续时间设置之间的角色会话持续时间。要了解如何查看您的角色的最大值，请参阅[查看角色的最大会话持续时间设置](#) (p. 238)。如果未传递该参数，临时凭证将在 1 小时后过期。该 API 中的 `DurationSeconds` 参数与用于指定控制台会话持续时间的 `SessionDuration` HTTP 参数不同。可以在发送到联合终端节点的控制台登录令牌请求中使用 `SessionDuration` HTTP 参数。有关更多信息，请参阅[使自定义身份代理能够访问 Amazon 控制台](#) (p. 201)。
- （可选）内联或托管会话策略。这些策略限制角色的基于身份的策略中分配给角色会话的权限。生成的会话的权限是角色的基于身份的策略与会话策略的交集。使用会话策略授予的权限不能超过担任的角色的基于身份的策略允许的权限。有关角色会话权限的更多信息，请参阅[会话策略](#) (p. 357)。
- 角色会话名称 使用此字符串值可在不同主体使用角色时标识会话。为了安全起见，管理员可以在 [Amazon CloudTrail 日志](#) (p. 343) 中查看此字段，以了解已在 Amazon 中执行操作的人员。您的管理员可能会要求您在代入角色时为会话名称提供特定的值。有关更多信息，请参阅[sts:RoleSessionName](#) (p. 1023)。
- （可选）源身份。您可以要求联合身份用户在担任角色时指定源身份。设置源身份后，无法更改该值。它在角色会话期间执行的所有操作的请求中都会出现。源身份值在[链接角色](#) (p. 159) 会话中持续存在。您可以将源身份信息用于 Amazon CloudTrail 日志来确定谁使用角色采取了操作。有关使用基于身份的策略的更多信息，请参阅[监控和控制使用所担任角色执行的操作](#) (p. 316)。

调用 `AssumeRoleWithSAML` 时，Amazon 将验证 SAML 断言的真实性。假定身份提供程序证实断言有效，则 Amazon 向您返回以下信息：

- 一组临时安全凭证。这些证书由访问密钥 ID、秘密访问密钥和会话令牌组成。
- 所担任角色的角色 ID 和 ARN。
- Audience 值，包含 SAML 断言的 Recipient 元素的 SubjectConfirmationData 属性值。
- Issuer 值，包含 SAML 断言的 Issuer 元素值。
- NameQualifier 元素，包含通过 Issuer 值、Amazon 账户 ID 以及 SAML 提供商的友好名称生成的哈希值。与 Subject 元素相结合时，它们可以唯一地标识联合身份用户。
- Subject 元素，包含 SAML 断言的 NameID 元素的 Subject 元素值。

- `SubjectType` 元素，指示 `Subject` 元素的格式。值可以是 `persistent`、`transient` 或在您的 SAML 断言中使用的 `Format` 和 `Subject` 元素的完整 `NameID URI`。有关 `NameID` 元素的 `Format` 属性的信息，请参阅[为身份验证响应配置 SAML 断言 \(p. 194\)](#)。

具有临时安全凭证时，可使用这些凭证调用 Amazon API。该过程与使用长期安全凭证进行 Amazon API 调用相同。不同之处在于，您必须包含会话令牌，以便 Amazon 验证临时安全凭证是否有效。

您的应用程序应缓存凭证。默认情况下，凭证在 1 小时后到期。如果未使用 [软件开发工具包](#)中的 `AmazonSTSCredentialsProviderAmazon` 操作，则由您和您的应用程序负责再次调用 `AssumeRoleWithSAML`。在旧安全凭证到期之前，可以调用该操作以获取一组新的临时安全凭证。

GetFederationToken – 通过自定义身份凭证代理程序进行联合身份验证

`GetFederationToken` API 操作为联合身份用户返回一组临时安全凭证。该 API 不同于 `AssumeRole`，其默认有效期大大增加 (12 小时而不是 1 小时)。此外，您还可以使用 `DurationSeconds` 参数指定临时安全凭证保持有效的持续时间。生成的凭证在指定的持续时间内有效，即，900 秒 (15 分钟) 到 129,600 秒 (36 小时) 之间。较长的有效期有助于减少对 Amazon 的调用次数，因为您不需要频繁获取新凭证。有关更多信息，请参阅[请求临时安全凭证 \(p. 302\)](#)。

您在发出此请求时使用特定 IAM 用户的凭证。临时安全凭证的权限是由调用 `GetFederationToken` 时传递的会话策略决定的。生成的会话权限是 IAM 用户策略与您传递的会话策略的交集。使用会话策略授予的权限不能超过请求联合的 IAM 用户的基于身份的策略允许的权限。有关角色会话权限的更多信息，请参阅[会话策略 \(p. 357\)](#)。

当您使用 `GetFederationToken` 操作返回的临时凭证时，会话的主体标签包括用户的标签以及传递的会话标签。有关会话标签的更多信息，请参阅[在 Amazon STS 中传递会话标签 \(p. 293\)](#)。

`GetFederationToken` 调用将返回临时安全凭证，其中包括会话令牌、访问密钥、私有密钥和到期时间。如果要在组织内管理权限 (例如，使用代理应用程序分配权限)，则可使用 `GetFederationToken`。要查看使用 `GetFederationToken` 的示例应用程序，请参阅[Amazon Sample Code & Libraries](#) (亚马逊云科技示例代码和库) 中的 Identity Federation Sample Application for an Active Directory Use Case (Active Directory 的联合身份验证示例应用程序使用案例)。

以下示例显示了使用 `GetFederationToken` 的示例请求和响应。此示例请求使用[会话策略 \(p. 357\)](#) ARN 和[会话标签 \(p. 293\)](#)，在指定的持续时间内联合调用用户的身份。生成的会话命名为 `Jane-session`。

Example 示例请求

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=GetFederationToken
&Name=Jane-session
&PolicyArns.member.1.arn==arn%3Aaws%3Aiam%3A%3A123456789012%3Apolicy%2FRole1policy
&DurationSeconds=1800
&Tags.member.1.Key=Project
&Tags.member.1.Value=Pegasus
&Tags.member.2.Key=Cost-Center
&Tags.member.2.Value=12345
&AUTHPARAMS
```

上述示例中显示的策略 ARN 包含以下 URL 编码的 ARN：

```
arn:aws:iam::123456789012:policy/Role1policy
```

还要注意，该示例中的 `&AUTHPARAMS` 参数用作身份验证信息的占位符。这是签名，您必须将其包括在 Amazon HTTP API 请求中。建议使用 [Amazon 开发工具包](#)创建 API 请求，这样做的一个好处是开发工具

包将为您处理请求签名。如果您必须手动创建并签署 API 请求，请前往 Amazon Web Services 一般参考中的[使用签名版本 4 签署 Amazon 的请求](#)以了解如何签署请求。

除了临时安全凭证之外，该响应还包括联合身份用户的 Amazon Resource Name (ARN) 和凭证的到期时间。

Example 响应示例

```
<GetFederationTokenResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
  <GetFederationTokenResult>
    <Credentials>
      <SessionToken>
        AQoDYXdzEPT/////////wEXAMPLEtc764bNrC9SAPBSM22wD0k4x4HIZ8j4FZTWdQW
        LWSKWHGBuFqWAeMicRXmxfpSPfIeoIYRqTflfKD8YUuwthAx7mSEI/qkPpKPi/kMcGd
        QrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDy0KPkYQDYwT7WZ0wq5VSXDvp75YU
        9HFvlRd8Tx6q6fE8YQcHNvXakiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
        +scqKmlzm8FDrypNC9Yjc8fPOLn9FX9KSYvKTr4rvx3iS1lTJabIQwj2ICCEXAMPLE==
      </SessionToken>
      <SecretAccessKey>
        wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY
      </SecretAccessKey>
      <Expiration>2019-04-15T23:28:33.359Z</Expiration>
      <AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
    </Credentials>
    <FederatedUser>
      <Arn>arn:aws:sts::123456789012:federated-user/Jean</Arn>
      <FederatedUserId>123456789012:Jean</FederatedUserId>
    </FederatedUser>
    <PackedPolicySize>4</PackedPolicySize>
  </GetFederationTokenResult>
  <ResponseMetadata>
    <RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
  </ResponseMetadata>
</GetFederationTokenResponse>
```

Note

Amazon 转换会将传递的会话策略和会话标签压缩为具有单独限制的打包二进制格式。即使您的纯文本符合其他要求，您的请求也可能会由于此限制而失败。PackedPolicySize 响应元素指示您请求的策略和标签接近大小上限的程度，以百分比来表示。

Amazon 建议您在资源级别授予权限（例如，将基于资源的策略附加到 Amazon S3 存储桶），则可省略 Policy 参数。但是，如果不加入针对联合身份用户的策略，则临时安全凭证将不授予任何权限。在这种情况下，您必须使用资源策略为联合身份用户授予您的 Amazon 资源的访问权限。

例如，假设您的 Amazon 账号为 111122223333，并具有希望允许 Susan 访问的 Amazon S3 存储桶。Susan 的临时安全凭证不包括该存储桶的策略。在这种情况下，您需要确保该存储桶的策略具有与 Susan 的 ARN (如 arn:aws:sts::111122223333:federated-user/Susan) 匹配的 ARN。

GetSessionToken – 不受信任环境中用户的临时凭证

GetSessionToken API 操作向现有 IAM 用户返回一组临时安全凭证。此 API 对提高安全性非常有用，例如在已针对 IAM 用户启用 MFA 的情况下提出 Amazon 请求。由于凭证是临时的，因此，在 IAM 用户通过不太安全的环境访问您的资源时，它们提供了增强的安全性。不太安全的环境示例包括移动设备或 Web 浏览器。有关更多信息，请参阅[请求临时安全凭证 \(p. 302\)](#)或 Amazon Security Token Service API 参考中的[GetSessionToken](#)。

默认情况下，IAM 用户的临时安全凭证的有效期最长为 12 小时。但是，您可以使用 DurationSeconds 参数请求短至 15 分钟或长达 36 小时的持续时间。出于安全考虑，Amazon Web Services 账户根用户的令牌有效期仅为一小时。

GetSessionToken 将返回临时安全凭证，其中包括会话令牌、访问密钥 ID 和秘密访问密钥。以下示例显示了使用 GetSessionToken 的示例请求和响应。响应也包括了临时安全凭证的过期时间。

Example 示例请求

```
https://sts.amazonaws.com/  
?Version=2011-06-15  
&Action=GetSessionToken  
&DurationSeconds=1800  
&AUTHPARAMS
```

示例中的 AUTHPARAMS 参数是您的签名的占位符。签名是您必须在 Amazon HTTP API 请求中包含的身份验证信息。建议使用 [Amazon 开发工具包](#) 创建 API 请求，这样做的一个好处是开发工具包将为您处理请求签名。如果您必须手动创建并签署 API 请求，请前往 Amazon Web Services 一般参考中的 [使用签名版本 4 签署 Amazon 的请求](#) 以了解如何签署请求。

Example 响应示例

```
<GetSessionTokenResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">  
<GetSessionTokenResult>  
<Credentials>  
<SessionToken>  
AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwnKwRcOIfrRh3c/L  
To6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/IvU1dYUg2RVAJBanLiHb4IgRmpRV3z  
rkuWJOgQs8IZZaIv2BXIa2R4OlGkBN9bkUDNCJiBeb/AXlZBBko7b15fjrBs2+cTQtp  
Z3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE  
</SessionToken>  
<SecretAccessKey>  
wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY  
</SecretAccessKey>  
<Expiration>2011-07-11T19:55:29.611Z</Expiration>  
<AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>  
</Credentials>  
</GetSessionTokenResult>  
<ResponseMetadata>  
<RequestId>58c5dbae-abef-11e0-8cfe-09039844ac7d</RequestId>  
</ResponseMetadata>  
</GetSessionTokenResponse>
```

(可选) GetSessionToken 请求可能包括 SerialNumber 和 TokenCode 值以进行 Amazon Multi-Factor Authentication (MFA) 验证。如果提供的值有效，Amazon STS 将提供包含 MFA 身份验证状态的临时安全凭证。然后，可以使用临时安全凭证访问受 MFA 保护的 API 操作或 Amazon 网站，但前提是 MFA 身份验证有效。

下例展示一个 GetSessionToken 请求，其中包括 MFA 验证代码和设备序列号。

```
https://sts.amazonaws.com/  
?Version=2011-06-15  
&Action=GetSessionToken  
&DurationSeconds=7200  
&SerialNumber=YourMFADeviceSerialNumber  
&TokenCode=123456  
&AUTHPARAMS
```

Note

对 Amazon STS 的调用可能针对的是全球终端节点，也可能针对的是激活您的 Amazon 账户的任何区域终端节点。有关更多信息，请参阅 [区域和终端节点的 Amazon STS 部分](#)。

示例中的 `AUTHPARAMS` 参数是您的签名的占位符。签名是您必须在 Amazon HTTP API 请求中包含的身份验证信息。建议使用 [Amazon 开发工具包](#) 创建 API 请求，这样做的一个好处是开发工具包将为您处理请求签名。如果您必须手动创建并签署 API 请求，请参阅 [Amazon Web Services 一般参考](#) 中的 [使用签名版本 4 签署 Amazon 的请求](#) 以了解如何签署请求。

比较 Amazon STS API 操作

下表比较了 Amazon STS 中返回临时安全凭证的 API 操作的功能。要了解在担任角色以请求临时安全凭证时使用的各种方法，请参阅 [使用 IAM 角色 \(p. 236\)](#)。要了解允许您传递会话标签的不同 Amazon STS API 操作，请参阅 [在 Amazon STS 中传递会话标签 \(p. 293\)](#)。

比较 API 选项

Amazon STS API	谁能调用	凭证生命周期 (最小值 最大值 默认值)	MFA 支持 ¹	会话策略支持 ²	对生成的临时凭证的限制
AssumeRole	具有现有临时安全凭证的 IAM 用户或 IAM 角色	15 分钟 最大会话持续时间设置 ³ 1 小时	是	是	无法调用 <code>GetFederationToken</code> 或 <code>GetSessionToken</code> 。
AssumeRoleWithSAML	任何用户，发起人必须传递 SAML 身份验证响应，指示身份验证来自自己知的身份提供程序	15 分钟 最大会话持续时间设置 ³ 1 小时	否	是	无法调用 <code>GetFederationToken</code> 或 <code>GetSessionToken</code> 。
AssumeRoleWithWebIdentity	任何用户，发起人必须传递 Web 身份令牌，指示身份验证来自自己知的身份提供程序	15 分钟 最大会话持续时间设置 ³ 1 小时	否	是	无法调用 <code>GetFederationToken</code> 或 <code>GetSessionToken</code> 。
GetFederationToken	IAM 用户或 Amazon Web Services 账户根用户	IAM 用户：15 分钟 36 小时 12 小时 根用户：15 分钟 1 小时 1 小时	否	是	无法使用 Amazon CLI 或 Amazon API 调用 IAM 操作。 无法调用除 <code>GetCallerIdentity</code> 之外的 Amazon STS 操作。 ⁴ 允许通过 SSO 登录到控制台。 ⁵
GetSessionToken	IAM 用户或 Amazon Web Services 账户根用户	IAM 用户：15 分钟 36 小时 12 小时 根用户：15 分钟 1 小时 1 小时	是	否	除非请求附带了 MFA 信息，否则无法调用 IAM API 操作。 无法调用除 <code>AssumeRole</code> 或 <code>GetCallerIdentity</code> 之外的 Amazon STS API 操作。 不允许通过 SSO 登录到控制台。 ⁶

¹ MFA 支持。在调用 AssumeRole 和 GetSessionToken API 操作时，可以包含有关 Multi-Factor Authentication (MFA) 设备的信息。这可确保通过 API 调用生成的临时安全凭证只能由已使用 MFA 设备进行身份验证的用户使用。有关更多信息，请参阅[配置受 MFA 保护的 API 访问](#) (p. 129)。

² 会话策略支持。会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的策略。此策略限制来自角色或用户的基于身份的策略且分配给该会话的权限。生成的会话的权限是实体的基于身份的策略与会话策略的交集。使用会话策略授予的权限不能超过担任的角色的基于身份的策略允许的权限。有关角色会话权限的更多信息，请参阅[会话策略](#) (p. 357)。

³ 最大会话持续时间设置。可以使用 DurationSeconds 参数指定 900 秒 (15 分钟) 到角色的最大会话持续时间设置之间的角色会话持续时间。要了解如何查看您的角色的最大值，请参阅[查看角色的最大会话持续时间设置](#) (p. 238)。

⁴ GetCallerIdentity。无需权限即可执行该操作。如果管理员为您的 IAM 用户或角色添加明确拒绝访问 sts:GetCallerIdentity 操作的策略，您仍然可以执行该操作。不需要权限，因为在拒绝 IAM 用户或角色访问时返回相同的信息。要查看示例响应，请参阅[我没有权限执行：iam:DeleteVirtualMFADevice](#) (p. 866)。

⁵ 通过单一登录 (SSO) 登录到控制台。为了支持 SSO，Amazon 可让您调用联合终端节点 (<https://signin.aws.amazon.com/federation>) 并传递临时安全凭证。终端节点将返回一个令牌，可使用该令牌构建一个让用户直接登录到控制台而无需使用密码的 URL。有关更多信息，请参阅[使 SAML 2.0 联合身份用户能够访问 Amazon Web Services Management Console](#) (p. 199) 安全性博客中的[和 Amazon 如何启用对管理控制台的跨账户访问 Amazon](#)。

⁶ 在检索您的临时凭证后，无法将该凭证传递到联合单一登录终端节点以访问 Amazon Web Services Management Console。有关更多信息，请参阅[使自定义身份代理能够访问 Amazon 控制台](#) (p. 201)。

将临时凭证用于 Amazon 资源

借助 Amazon CLI 或 Amazon API (使用 [Amazon SDK](#))，您可以使用临时安全凭证提出对 Amazon 资源的编程请求。临时凭证提供的权限与长期安全凭证 (例如 IAM 用户凭证) 相同。但还是有几个区别：

- 使用临时安全凭证进行调用时，必须在调用中包含随这些临时凭证一同返回的会话令牌。Amazon 使用该会话令牌来验证临时安全凭证的有效性。
- 临时凭证在指定间隔后到期。临时凭证到期后，任何使用这些凭证进行的调用都将失败，因此您必须生成一组新的临时凭证。临时凭证的延期或刷新时间不得超过最初指定的时间间隔。
- 当您使用临时凭证发出请求时，您的主体可能会包含一组标签。这些标签来自附加到您代入的角色的会话标签和标签。有关会话标签的更多信息，请参阅[在 Amazon STS 中传递会话标签](#) (p. 293)。

如果您使用的是 [Amazon SDK](#)、[Amazon Command Line Interface](#) (Amazon CLI) 或 [Tools for Windows PowerShell](#)，则获取和使用临时安全凭证的方式随上下文而不同。如果您在 Amazon CLI 内部运行代码、或 Tools for Windows PowerShell 命令，则可使用 Amazon EC2 的角色。否则，您可调用 [Amazon STS API](#) 来获取临时凭证，然后使用这些凭证显式地调用 Amazon 服务。

Note

您可以使用 Amazon Security Token Service (Amazon STS) 创建可控制对您的 Amazon 资源的访问的临时安全凭证，并将这些凭证提供给可信用户。有关 Amazon STS 的更多信息，请参阅[IAM 临时安全凭证](#) (p. 300)。Amazon STS 是一种全球服务，在 <https://sts.amazonaws.com> 上具有默认终端节点。此端点处于美国东部 (弗吉尼亚州北部) 区域，不过，您从此端点和其他端点获取的凭证在全球范围都有效。这些凭证可用于任何区域中的服务和资源。您也可以选择对任何支持的区域中的终端节点进行 Amazon STS API 调用。这可能会从地理位置离您较近的区域的服务端点发出请求以减少延迟。无论您的凭证来自于哪个区域，它们都会在全球范围内起作用。有关更多信息，请参阅[在 Amazon 区域中管理 Amazon STS](#) (p. 331)。

目录

- [在 Amazon EC2 实例中使用临时凭证](#) (p. 312)

- [将临时安全凭证用于 Amazon SDK。](#) (p. 312)
- [将临时安全凭证用于 Amazon CLI。](#) (p. 312)
- [将临时安全凭证用于 API 操作](#) (p. 313)
- [更多信息](#) (p. 313)

在 Amazon EC2 实例中使用临时凭证

如果要在 EC2 实例内部运行 Amazon CLI 命令或代码，建议的获取凭证的方法是使用 [Amazon EC2](#) 的角色。您可创建一个 IAM 角色，通过该角色指定要授予在 EC2 实例中运行的应用程序的权限。启动实例时，将该角色关联至实例。

之后，在该实例上运行的应用程序、Amazon CLI 和 Tools for Windows PowerShell 命令将能够从该实例的元数据获取自动临时安全凭证。您无需明确获取临时安全凭证。Amazon SDK、Amazon CLI 和 Tools for Windows PowerShell 自动从 EC2 实例元数据服务获取凭证并使用它们。临时凭证具有您为该实例关联的角色定义的权限。

有关更多信息及示例，请参阅：

- [使用 IAM 角色授予对 Amazon Elastic Compute Cloud 上 Amazon 资源的访问权](#) — Amazon SDK for Java
- [使用 IAM 角色授予访问权限 IAM](#) - Amazon SDK for .NET
- [创建角色](#) - Amazon SDK for Ruby

将临时安全凭证用于 Amazon SDK。

要在代码中使用临时安全凭证，可以编程方式调用类似于 Amazon STS 的 AssumeRole API，提取生成的凭证和会话令牌。然后，您可以使用这些值作为对 Amazon 的后续调用的凭证。以下示例说明了有关使用 Amazon 开发工具包时如何使用临时安全凭证的伪代码：

```
assumeRoleResult = AssumeRole(role-arn);
tempCredentials = new SessionAWSCredentials(
    assumeRoleResult.AccessKeyId,
    assumeRoleResult.SecretAccessKey,
    assumeRoleResult.SessionToken);
s3Request = CreateAmazonS3Client(tempCredentials);
```

有关用 Python 编写的示例（使用 [Amazon SDK for Python \(Boto\)](#)），请参阅[切换到 IAM 角色 \(Amazon API\)](#) (p. 250)。此示例说明如何调用 AssumeRole 以获取临时安全凭证，然后使用这些凭证调用 Amazon S3。

有关如何调用 AssumeRole、GetFederationToken 和其他 API 操作的详细信息，请参阅 [Amazon Security Token Service API 参考](#)。有关从结果中获取临时安全凭证和会话令牌的信息，请参阅所用开发工具包的文档。可以在主 [Amazon 文档页](#) 上的 SDKs and Toolkits（开发工具包和工具箱）部分中找到所有 Amazon SDK 的文档。

您必须确保在旧凭证到期之前，获得一组新的凭证。在某些开发工具包中，可让提供商为您管理刷新凭证的过程；并可检查所用开发工具包的文档。

将临时安全凭证用于 Amazon CLI。

可将临时安全凭证用于 Amazon CLI。这对于测试策略来说很有用。

借助 [Amazon CLI](#)，您可以调用 AssumeRole 或 GetFederationToken 之类的 [Amazon STS API](#)，然后捕获输出结果。下面的示例说明了一个将输出发送至文件的 AssumeRole 调用。在示例中，假定 profile 参数是配置文件中的 Amazon CLI 配置文件。它还假定为有权代入角色的 IAM 用户引用凭证。


```
aws sts assume-role --role-arn arn:aws:iam::123456789012:role/role-name --role-session-name  
"RoleSession1" --profile IAM-user-name > assume-role-output.txt
```

当命令完成后，您可以从路由到的任意位置中提取访问密钥 ID、秘密访问密钥和会话令牌。您可以手动或使用脚本执行此操作。之后，您可以将这些值分配给环境变量。

在运行 Amazon CLI 命令时，Amazon CLI 依特定顺序查找凭证 - 首先查找环境变量，然后是配置文件。因此，在将临时凭证放入环境变量后，Amazon CLI 会默认使用这些凭证。（如果在该命令中指定了 `profile` 参数，则 Amazon CLI 将跳过环境变量，而 Amazon CLI 在配置文件中查找，这使您能够根据需要覆盖环境变量中的凭证。）

下面的示例说明了如何为临时安全凭证设置环境变量然后调用 Amazon CLI 命令。由于未在 Amazon CLI 命令中包含 `profile` 参数，Amazon CLI 首先在环境变量中查找凭证，因而将使用该临时凭证。

Linux

```
$ export AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE  
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY  
$ export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of session token>  
$ aws ec2 describe-instances --region us-west-1
```

Windows

```
C:\> SET AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE  
C:\> SET AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY  
C:\> SET AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of token>  
C:\> aws ec2 describe-instances --region us-west-1
```

将临时安全凭证用于 API 操作

如果是直接向 Amazon 发出 HTTPS API 请求，则可以使用从 Amazon Security Token Service (Amazon STS) 获取的临时安全凭证对这些请求进行签名。为此，您可以使用接收自 Amazon STS 的访问密钥 ID 和秘密访问密钥。您使用访问密钥 ID 和秘密访问密钥的方式，与使用长期凭证对请求签名是一样的。此外，还要将您从 Amazon STS 获得的会话令牌添加到 API 请求中。将会话令牌添加到 HTTP 标头或名为 `X-Amz-Security-Token` 的查询字符串参数中。将会话令牌添加到 HTTP 标头或 查询字符串参数，但不是同时添加到这两者。有关签署 HTTPS API 请求的更多信息，请参阅 Amazon 一般参考中的[签署 Amazon API 请求](#)。

更多信息

有关将 Amazon STS 与其他 Amazon 服务结合使用的更多信息，请参阅以下链接：

- Amazon S3。请参阅 Amazon Simple Storage Service 用户指南中的[使用 IAM 用户临时凭证创建请求或者使用联合身份用户临时凭证创建请求](#)。
- Amazon SNS。请参阅 Amazon Simple Notification Service 开发人员指南中的[使用临时安全凭证](#)。
- Amazon SQS。请参阅 Amazon Simple Queue Service 开发人员指南中的[使用临时安全凭证](#)。
- Amazon SimpleDB。请参阅 Amazon SimpleDB 开发人员指南中的[使用临时安全凭证](#)。

控制临时安全凭证的权限

您可以使用 Amazon Security Token Service (Amazon STS) 创建可控制对您的 Amazon 资源的访问的临时安全凭证，并将这些凭证提供给可信用户。有关 Amazon STS 的更多信息，请参阅[IAM 临时安全凭证 \(p. 300\)](#)。Amazon STS 颁发临时安全凭证后，这些凭证在到期时间之前有效，并且无法撤销。但是，

由于每次使用证书发出请求时都会评估分配给临时安全凭证的权限，因此，即使证书已经签发，您仍可通过更改证书的访问权限来实现撤消证书的效果。

以下主题假定您具备一些 Amazon 权限和策略方面的知识。有关这些主题的更多信息，请参阅[适用于 Amazon 资源的 Access Management \(p. 353\)](#)。

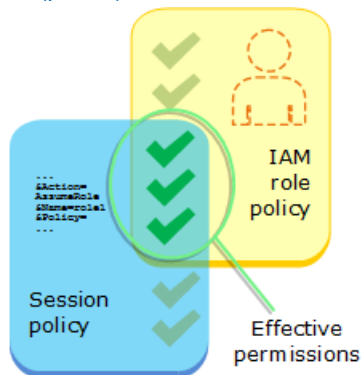
主题

- [AssumeRole、AssumeRoleWithSAML 和 AssumeRoleWithWebIdentity 的权限 \(p. 314\)](#)
- [监控和控制使用所担任角色执行的操作 \(p. 316\)](#)
- [GetFederationToken 的权限 \(p. 324\)](#)
- [GetSessionToken 的权限 \(p. 327\)](#)
- [禁用临时安全凭证的权限 \(p. 328\)](#)
- [授予创建临时安全凭证的权限 \(p. 330\)](#)

AssumeRole、AssumeRoleWithSAML 和 AssumeRoleWithWebIdentity 的权限

所担任的角色的权限策略决定由 AssumeRole、AssumeRoleWithSAML 和 AssumeRoleWithWebIdentity 返回的临时安全凭证的权限。您可在创建或更新该角色时定义这些权限。

(可选) 您可以将内联或托管[会话策略 \(p. 357\)](#)作为 AssumeRole、AssumeRoleWithSAML 或 AssumeRoleWithWebIdentity API 操作的参数传递。会话策略限制角色的临时凭证会话的权限。生成的会话的权限是角色的基于身份的策略与会话策略的交集。您可以在后续的 Amazon API 调用中使用角色的临时凭证来访问拥有该角色的账户中的资源。您使用会话策略授予的权限不能超过担任的角色的基于身份的策略允许的权限。要了解有关 Amazon 如何确定角色的有效权限的更多信息，请参阅[策略评估逻辑 \(p. 972\)](#)。



在做出“允许”或“拒绝”授权决定时，Amazon 不会对附加至发出 AssumeRole 原始调用的凭证的策略进行评估。用户将临时放弃其原始权限，以支持担任的角色所分配的权限。对于 AssumeRoleWithSAML 和 AssumeRoleWithWebIdentity API 操作，不会有任何策略受到评估，因为这些 API 的发起人不是 Amazon 身份。

示例：使用 AssumeRole 分配权限

您可以将 AssumeRole API 操作与不同类型的策略结合使用。以下是一些示例。

角色权限策略

在本示例中，您调用 AssumeRole API 操作，而无需在可选 Policy 参数中指定会话策略。分配给临时凭证的权限取决于所担任角色的权限策略。以下示例权限策略向角色授予权限，允许该角色列出名为 productionapp 的 S3 存储桶中包含的所有对象。它还允许该角色获取、放置和删除该存储桶中的对象。

Example 角色权限策略示例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::productionapp"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::productionapp/*"
    }
  ]
}
```

作为参数传递的会话策略

假设您希望允许某用户担任上例中的相同角色。但在这种情况下，您希望角色会话只具有在 productionapp S3 存储桶中获取和放入对象的权限。您不希望允许他们删除对象。完成该操作的一种方法是创建一个新角色，并在该角色的权限策略中指定所需的权限。完成该操作的另一种方法是，调用 AssumeRole API 并在可选的 Policy 参数中包含会话策略以作为 API 操作的一部分。生成的会话的权限是角色的基于身份的策略与会话策略的交集。使用会话策略授予的权限不能超过担任的角色的基于身份的策略允许的权限。有关角色会话权限的更多信息，请参阅[会话策略 \(p. 357\)](#)。

在检索新会话的临时凭证后，您可以将其传递给希望具有这些权限的用户。

例如，假设将以下策略作为该 API 调用的参数传递。使用会话的用户只具备执行以下操作的权限：

- 列出 productionapp 存储桶中的所有对象。
- 获取 productionapp 存储桶中的对象或向其中上传对象。

在以下会话策略中，s3:DeleteObject 权限已被筛选掉，因此，未向担任的会话授予 s3:DeleteObject 权限。该策略为角色会话设置最大权限，以便它覆盖角色上的任何现有权限策略。

Example 通过 AssumeRole API 调用传递的示例会话策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::productionapp"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::productionapp/*"
    }
  ]
}
```

```
}
```

基于资源的策略

某些 Amazon 资源支持基于资源的策略，这些策略提供另一种机制用于定义影响临时安全凭证的权限。仅几种资源（如 Amazon S3 存储桶、Amazon SNS 主题和 Amazon SQS 队列）支持基于资源的策略。下面的示例使用名为 `productionapp` 的 S3 存储桶进一步阐述上述示例。以下策略被附加到该存储桶。

在将以下基于资源的策略附加到 `productionapp` 存储桶时，将会拒绝所有用户从该存储桶中删除对象的权限。（请参阅策略中的 `Principal` 元素。）这包括所有担任角色的用户（即使角色权限策略授予了 `DeleteObject` 权限）。显式 `Deny` 语句总是优先于 `Allow` 语句。

Example 存储桶策略的示例

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Principal": {"AWS": "*"},
    "Effect": "Deny",
    "Action": "s3:DeleteObject",
    "Resource": "arn:aws:s3:::productionapp/*"
  }
}
```

有关 Amazon 如何对多个策略类型进行合并和评估的更多信息，请参阅[策略评估逻辑](#) (p. 972)。

监控和控制使用所担任角色执行的操作

[IAM 角色](#) (p. 158)是 IAM 中分配[权限](#) (p. 355)的对象。当您使用 IAM 身份或来自 Amazon 以外的身份担任该角色 (p. 236)时，将会收到具有分配给该角色的权限的会话。

当您在 Amazon 中执行操作时，则有关您会话的信息可以记录到 Amazon CloudTrail 以便您的[账户管理员](#) (p. 19)进行监控。管理员可以配置角色以要求身份来传递自定义字符串，该字符串标识在 Amazon 中执行操作的个人或应用程序，称为源身份。此身份信息在 Amazon CloudTrail 中存储为源身份。当管理员查看 CloudTrail 中的活动时，他们可以查看源身份信息，以确定谁通过担任角色会话执行了操作或执行了哪些操作。

设置源身份后，它会出现出现在任何在角色会话期间执行的 Amazon 操作的请求中。当角色用于通过 Amazon CLI 或者 Amazon API 担任其他角色时，设置的值仍然存在，这称为[角色链](#) (p. 159)。在角色会话期间无法更改已设置的值。管理员可以根据源身份的存在或值配置精细权限，以进一步控制使用共享角色执行的 Amazon 操作。您可以决定是否可以使用源身份属性、是否需要该属性以及可以使用哪些值。

使用源身份的方式与使用角色会话名称和会话标签的方式大有不同。设置后，源身份值无法更改，并且对角色会话执行的任何其他操作将持续存在。以下是如何使用会话标签和角色会话名称的说明：

- Session tags (会话标签) - 您也可以在代入角色或联合身份用户时传递会话标签。担任角色时会出现会话标签。您可以定义策略，这些策略使用标签条件键来根据主体的标签向其授予权限。然后您可以使用 CloudTrail 查看为代入角色或联合身份用户而发出的请求。要了解会话标签的更多信息，请参阅[在 Amazon STS 中传递会话标签](#) (p. 293)。
- Role session name (角色会话名称) - 您可以在角色信任策略中使用 `sts:RoleSessionName` 条件键，以要求您的用户在代入角色时提供特定的会话名称。角色会话名称可用于区分角色由不同主体使用时的角色会话。要了解角色会话名称的更多信息，请参阅[sts:RoleSessionName](#) (p. 1023)。

我们建议您在要控制担任角色的身份时使用源身份。源身份在挖掘 CloudTrail 日志以确定是何人在使用该角色执行操作时也很有用。

主题

- [设置以使用源身份 \(p. 317\)](#)
- [有关源身份的需知信息 \(p. 317\)](#)
- [设置源身份时所需的权限 \(p. 318\)](#)
- [在担任角色时指定源身份 \(p. 319\)](#)
- [使用具有 AssumeRole 的源身份 \(p. 319\)](#)
- [使用具有 AssumeRoleWithSAML 的源身份 \(p. 319\)](#)
- [使用具有 AssumeRoleWithWebIdentity 源身份 \(p. 320\)](#)
- [使用源身份信息控制访问 \(p. 320\)](#)
- [在 CloudTrail 中查看源身份 \(p. 323\)](#)

设置以使用源身份

设置以使用源身份的方式取决于担任角色时使用的方法。例如，您的 IAM 用户可以直接使用 AssumeRole 操作。如果您具有企业身份（也称为人力身份），则他们可能会使用 AssumeRoleWithSAML 访问您的 Amazon 资源。如果终端用户访问您的移动应用程序或 Web 应用程序，他们可能会使用 AssumeRoleWithWebIdentity。以下是一个高级别的工作流概览，可帮助您了解如何进行设置以在现有环境中利用源身份信息。

1. 配置测试用户和角色 — 使用预生产环境，配置测试用户和角色，并配置其策略以允许设置源身份。

如果您对联合身份使用身份提供程序 (IdP)，请将 IdP 配置为在断言或令牌中传递您选择的源身份的用户属性。

2. 担任角色 — 测试所担任角色并将源身份传递给您为测试而设置的用户和角色。
3. 查看 CloudTrail — 查看 CloudTrail 日志中测试角色的源身份信息。
4. 培训您的用户 — 在预生产环境中进行测试后，请确保用户知道如何传递源身份信息（如有必要）。设置要求用户在生产环境中提供源身份的最后期限。
5. 配置生产策略 — 为生产环境配置策略，然后将其添加到生产用户和角色中。
6. 监控活动 — 使用 CloudTrail 日志监控您的生产角色活动。

有关源身份的需知信息

使用源身份时请记住以下事项。

- 连接到身份提供程序 (IdP) 的所有角色的角色信任策略都必须具有 sts:SetSourceIdentity 权限。对于信任策略中没有此权限的角色，AssumeRole* 操作将失败。如果您不想更新每个角色的角色信任策略，则可以使用单独的 IdP 实例传递源身份。然后仅将 sts:SetSourceIdentity 权限添加到连接至单独 IdP 的角色。
- 当身份设置源身份时，sts:SourceIdentity 密钥将出现在请求中。对于角色会话期间执行的后续操作，aws:SourceIdentity 密钥将出现在请求中。Amazon 不控制 sts:SourceIdentity 或 aws:SourceIdentity 密钥中源身份的值。如果选择要求源身份，则必须选择希望用户或 IdP 提供的属性。出于安全考虑，您必须确保可以控制如何来提供这些值。
- 源身份值的长度必须介于 2 到 64 个字符之间，只能包含字母数字字符、下划线和以下字符：., += @ - (连字符)。您不能创建以文本 **aws:** 开头的值。此前缀是专为 Amazon 内部使用预留的。
- 当 Amazon 服务或服务关联角色代表联合身份或工作人员身份执行操作时，源身份信息不会被 CloudTrail 捕获。

Important

您无法切换为 Amazon Web Services Management Console 中在担任角色时需要设置源身份的角色。要担任这样的角色，您可以使用 Amazon CLI 或者 Amazon API 来调用 AssumeRole 操作并指定源身份参数。

设置源身份时所需的权限

除了与 API 操作匹配的操作之外，您的策略中还必须具有以下仅限授权执行的操作：

```
sts:SetSourceIdentity
```

- 要指定源身份，主体（IAM 用户和角色）必须具有对 `sts:SetSourceIdentity` 的权限。作为管理员，您可以在角色信任策略和主体的权限策略中对此进行配置。
- 当您使用另一个角色担任角色时，该操作称为 [角色链 \(p. 159\)](#)，在担任角色的主体的权限策略和目标角色的角色信任策略中都需要对 `sts:SetSourceIdentity` 的权限。否则，担任角色的操作将失败。
- 使用源身份时，连接到 IdP 的所有角色的角色信任策略都必须具有 `sts:SetSourceIdentity` 权限。对于连接到 IdP 的任何角色，如果在没有此权限的情况下，`AssumeRole*` 操作将失败。如果您不想更新每个角色的角色信任策略，则可以使用单独的 IdP 实例传递源身份，然后将 `sts:SetSourceIdentity` 权限仅添加到与单独的 IdP 连接的角色。
- 要跨账户边界设置源身份，您必须在两个位置包含 `sts:SetSourceIdentity` 权限。该权限必须位于原始账户中主体的权限策略和目标账户中角色的角色信任策略中。例如，当某个角色用于在另一个账户中使用 [角色链 \(p. 159\)](#) 担任某个角色时，您可能需要执行此操作。

作为账户管理员，假设您希望允许 IAM 用户 `DevUser` 在您的账户中担任在同一账户中的 `Developer_Role`。但是，您希望只有当用户已将源身份设置为其 IAM 用户名时才允许此操作。那么，您可以将以下策略附加到 IAM 用户。

Example 附加到 `DevUser` 的基于身份的策略示例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeRole",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::123456789012:role/Developer_Role"
    },
    {
      "Sid": "SetAwsUserNameAsSourceIdentity",
      "Effect": "Allow",
      "Action": "sts:SetSourceIdentity",
      "Resource": "arn:aws:iam::123456789012:role/Developer_Role",
      "Condition": {
        "StringLike": {
          "sts:SourceIdentity": "${aws:username}"
        }
      }
    }
  ]
}
```

要强制实施可接受的源身份值，可以配置以下角色信任策略。该策略为 IAM 用户提供 `DevUser` 权限以担任该角色并设置源身份。`sts:SourceIdentity` 条件键定义可接受的源身份值。

Example 源身份的角色信任策略示例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDevUserAssumeRole",
```



```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::123456789012:user/DevUser"
  },
  "Action": [
    "sts:AssumeRole",
    "sts:SetSourceIdentity"
  ],
  "Condition": {
    "StringEquals": {
      "sts:SourceIdentity": "DevUser"
    }
  }
}
```

借助 IAM 用户 DevUser 的凭证，用户将尝试担任使用以下 Amazon CLI 请求的 DeveloperRole。

Example 示例 AssumeRole CLI 请求

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Developer_Role \
--role-session-name Dev-project \
--source-identity DevUser \
```

当 Amazon 评估请求时，请求上下文将包含 DevUser 的 sts:SourceIdentity。

在担任角色时指定源身份

当您使用 Amazon STS AssumeRole*API 操作来获取角色的临时安全凭证时，可以指定源身份。您使用的 API 操作因您的使用案例而异。例如，如果您使用 IAM 角色授予 IAM 用户访问 Amazon 资源的权限（而其通常并不具有此权限），则可以使用 AssumeRole 操作。如果您使用企业联合身份验证来管理工作人员用户，则可以使用 AssumeRoleWithSAML 操作。如果您使用 Web 联合身份验证来允许终端用户访问您的移动或 Web 应用程序，则可以使用 AssumeRoleWithWebIdentity 操作。以下部分介绍如何在每个操作中使用源身份。要了解有关临时凭证常见场景的更多信息，请参阅 [临时凭证的常见情形 \(p. 301\)](#)。

使用具有 AssumeRole 的源身份

AssumeRole 操作返回一组可用于访问 Amazon 资源的临时凭证。您可以使用 IAM 用户或角色凭证来调用 AssumeRole。要在代入角色时传递源身份，请使用 --source-identity Amazon CLI 选项或 SourceIdentity Amazon API 参数。以下示例说明如何使用 Amazon CLI 指定源身份。

Example 示例 AssumeRole CLI 请求

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/developer \
--role-session-name Audit \
--source-identity Admin \
```

使用具有 AssumeRoleWithSAML 的源身份

主体调用 AssumeRoleWithSAML 操作使用基于 SAML 的联合身份进行身份验证。此操作返回一组可用于访问 Amazon 资源的临时凭证。有关将基于 SAML 的联合身份用于 Amazon Web Services Management Console 访问的更多信息，请参阅[使 SAML 2.0 联合身份用户能够访问 Amazon Web Services Management Console \(p. 199\)](#)。有关 Amazon CLI 或 Amazon API 访问的详细信息，请参阅[关于基于 SAML 2.0 的联合](#)

[身份验证 \(p. 176\)](#)。有关为 Active Directory 用户设置 SAML 联合身份的教程，请参阅 Amazon 安全博客中的[使用 Active Directory Amazon 联合身份验证服务 \(ADFS\) 的亚马逊云科技联合身份验证](#)。

作为管理员，您可以使用 Amazon STS `AssumeRoleWithSAML` 操作，允许公司目录的成员联合身份到 Amazon 中。要执行此操作，您必须完成以下任务：

1. [在企业中配置 SAML 提供程序 \(p. 192\)](#)。
2. [在 IAM 中创建 SAML 提供商 \(p. 188\)](#)
3. [在 Amazon 中为联合身份用户配置角色及其权限 \(p. 231\)](#)。
4. [完成配置 SAML IdP 并为 SAML 身份验证响应创建断言 \(p. 194\)](#)。

要为源身份设置 SAML 属性，请包含 `Attribute` 元素，并将 `Name` 属性设置为 `https://aws.amazon.com/SAML/Attributes/SourceIdentity`。使用 `AttributeValue` 元素指定源身份的值。例如，假设您要以下身份属性作为源身份传递。

```
SourceIdentity:DiegoRamirez
```

要传递这些属性，请在 SAML 断言中包含以下元素。

Example SAML 断言的示例片段

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SourceIdentity">
<AttributeValue>DiegoRamirez</AttributeValue>
</Attribute>
```

使用具有 `AssumeRoleWithWebIdentity` 源身份

主体调用 `AssumeRoleWithWebIdentity` 操作使用兼容 OpenID Connect (OIDC) 的 Web 联合身份进行身份验证。此操作返回一组可用于访问 Amazon 资源的临时凭证。有关将基于 Web 联合身份用于 Amazon Web Services Management Console 访问的更多信息，请参阅[关于 Web 联合身份验证 \(p. 171\)](#)。

要从 OpenID Connect (OIDC) 传递源身份，您必须在 JSON Web Token (JWT) 中包含源身份。当您提交 `AssumeRoleWithWebIdentity` 请求时，请在令牌的 http://www.amazonaws.cn/source_identity 命名空间中包含源身份。要了解有关 OIDC 令牌和声明的更多信息，请参阅《Amazon Cognito 开发人员指南》中的[将令牌与用户群体结合使用](#)。

例如，以下解码的 JWT 是一个令牌，用于通过 Admin 源身份调用 `AssumeRoleWithWebIdentity`。

Example 解码的 JSON Web 令牌示例

```
{
  "sub": "johndoe",
  "aud": "ac_oic_client",
  "jti": "ZYUCeRMQVtqHypVPWAN3VB",
  "iss": "https://xyz.com",
  "iat": 1566583294,
  "exp": 1566583354,
  "auth_time": 1566583292,
  "https://aws.amazon.com/source_identity": "Admin"
}
```

使用源身份信息控制访问

当初始设置源身份时，`sts:SourceIdentity` (p. 1024) 密钥将出现在请求中。设置源身份后，`aws:SourceIdentity` (p. 1009) 密钥存在于角色会话期间提出的所有后续请求中。作为管理员，您可以编写策略，授予条件授权以根据源身份属性的现状或值执行 Amazon 操作。

假设您希望要求开发人员设置源身份，以承担一个关键角色，该角色有权写入生产关键 Amazon 资源。再假设您使用 `AssumeRoleWithSAML` 向您的工作人员身份授予了 Amazon 访问权限。您只希望高级开发人员 Saanvi 和 Diego 具有该角色的访问权限，因此您可以为角色创建以下信任策略。

Example 源身份的角色信任策略示例 (SAML)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SAMLProviderAssumeRoleWithSAML",
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:saml-provider/name-of-identity-provider"
      },
      "Action": [
        "sts:AssumeRoleWithSAML"
      ],
      "Condition": {
        "StringEquals": {
          "SAML:aud": "https://signin.aws.amazon.com/saml"
        }
      }
    },
    {
      "Sid": "SetSourceIdentitySrEngs",
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:saml-provider/name-of-identity-provider"
      },
      "Action": [
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringLike": {
          "sts:SourceIdentity": [
            "Saanvi",
            "Diego"
          ]
        }
      }
    }
  ]
}
```

信任策略包含 `sts:SourceIdentity` 的条件，需要将源身份设置为能够使 Saanvi 或 Diego 担任关键角色。

或者，如果您使用 OIDC 提供程序进行 web 身份联合，并用 `AssumeRoleWithWebIdentity` 对用户进行身份验证，您的角色信任策略可能如下所示。

Example 源身份的角色信任策略示例 (OIDC 提供商)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/server.example.com"
      },
      "Action": [
```

```
    "sts:AssumeRoleWithWebIdentity",
    "sts:SetSourceIdentity"
  ],
  "Condition": {
    "StringEquals": {
      "server.example.com:aud": "oidc-audience-id"
    },
    "StringLike": {
      "sts:SourceIdentity": [
        "Saanvi",
        "Diego"
      ]
    }
  }
}
```

角色链和跨账户要求

假设您想允许已经担任 CriticalRole 的用户在另一个账户中担任 CriticalRole_2。获得以用户担任 CriticalRole 的角色会话凭证用于[角色链](#) (p. 159)到不同账户中的第二个角色，CriticalRole_2。该角色正在以跨账户边界的形式担任。因此，sts:SetSourceIdentity 权限必须在权限策略中授予 CriticalRole，在角色信任策略中授予 CriticalRole_2。

Example CriticalRole 上的权限策略示例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeRoleAndSetSourceIdentity",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Resource": "arn:aws:iam::222222222222:role/CriticalRole_2"
    }
  ]
}
```

为确保跨账户边界设置源身份，以下角色信任策略仅信任 CriticalRole 的角色主体来设置源身份。

Example CriticalRole_2 上的角色信任策略示例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111111111111:role/CriticalRole"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringLike": {
          "aws:SourceIdentity": ["Saanvi", "Diego"]
        }
      }
    }
  ]
}
```

```
}  
}  
]  
}
```

用户使用从担任 CriticalRole 获得的角色会话凭证进行以下调用。源身份是在担任 CriticalRole 期间设置的，因此无需再次显式设置。如果用户尝试设置与担任 CriticalRole 所设置值不同的源身份，则担任角色请求将被拒绝。

Example 示例 AssumeRole CLI 请求

```
aws sts assume-role \  
--role-arn arn:aws:iam::222222222222:role/CriticalRole_2 \  
--role-session-name Audit \  

```

当调用主体担任角色时，请求中的源身份将从第一个担任的角色会话保留。因此，aws:SourceIdentity 和 sts:SourceIdentity 密钥将会出现在请求上下文中。

在 CloudTrail 中查看源身份

您可以使用 CloudTrail 查看为代入角色或联合身份用户而发出的请求。您还可以查看角色或用户请求，以便在 Amazon 中执行操作。CloudTrail 日志文件包括有关所代入角色或联合身份用户会话的源身份集的信息。有关更多信息，请参阅

例如，假设用户提出 Amazon STS AssumeRole 请求，并设置了源身份。您可以在 CloudTrail 日志的 requestParameters 键中找到 sourceIdentity 信息。

Example Amazon CloudTrail 日志中的示例 requestParameters 部分

```
"eventVersion": "1.05",  
  "userIdentity": {  
    "type": "AWSAccount",  
    "principalId": "AIDAJ45Q7YFFAREXAMPLE",  
    "accountId": "123456789012"  
  },  
  "eventTime": "2020-04-02T18:20:53Z",  
  "eventSource": "sts.amazonaws.com",  
  "eventName": "AssumeRole",  
  "awsRegion": "us-east-1",  
  "sourceIPAddress": "203.0.113.64",  
  "userAgent": "aws-cli/1.16.96 Python/3.6.0 Windows/10 botocore/1.12.86",  
  "requestParameters": {  
    "roleArn": "arn:aws:iam::123456789012:role/Assumed_Role",  
    "roleSessionName": "Test1",  
    "sourceIdentity": "source-identity-value-set",  
  },  
}
```

如果用户使用所担任的角色会话执行操作，则源身份信息将出现在 CloudTrail 日志的 userIdentity 密钥中。

Example Amazon CloudTrail 日志中的 userIdentity 密钥示例

```
{  
  "eventVersion": "1.08",  
  "userIdentity": {  
    "type": "AssumedRole",  
    "principalId": "AROAJ45Q7YFFAREXAMPLE:Dev1",  
  },  
}
```

```
{
  "arn": "arn:aws:sts::123456789012:assumed-role/DevRole/Dev1",
  "accountId": "123456789012",
  "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
  "sessionContext": {
    "sessionIssuer": {
      "type": "Role",
      "principalId": "AROAJ45Q7YFFAREXAMPLE",
      "arn": "arn:aws:iam::123456789012:role/DevRole",
      "accountId": "123456789012",
      "userName": "DevRole"
    },
    "webIdFederationData": {},
    "attributes": {
      "mfaAuthenticated": "false",
      "creationDate": "2021-02-21T23:46:28Z"
    },
    "sourceIdentity": "source-identity-value-present"
  }
}
```

要查看 CloudTrail 日志中的示例 Amazon STS API 事件，请参阅 [CloudTrail 日志中的 IAM API 事件示例 \(p. 344\)](#)。有关 CloudTrail 日志文件中所包含信息的更多详细信息，请参阅 Amazon CloudTrail 用户指南中的 [CloudTrail 事件参考](#)。

GetFederationToken 的权限

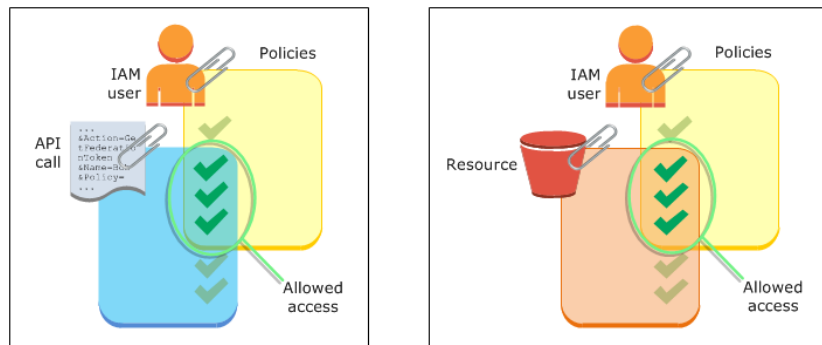
GetFederationToken 操作是由 IAM 用户调用的，并返回该用户的临时凭证。该操作联合 用户。分配给联合身份用户的权限是在以下两个位置之一中定义的：

- 作为 GetFederationToken API 调用参数传递的会话策略。(最常见。)
- 一项基于资源的策略，在该策略的 Principal 元素中对联合身份用户进行显式命名。(不太常见。)

会话策略是高级策略，在以编程方式创建临时会话时，这些策略将作为参数进行传递。在创建联合身份用户会话并传递会话策略时，生成的会话的权限是 IAM 用户的基于身份的策略与会话策略的交集。您使用会话策略授予的权限不能超过联合的用户的基于身份的策略允许的权限。

在大多数情况下，如果您未通过 GetFederationToken API 调用传递策略，则生成的临时安全凭证没有任何权限。不过，基于资源的策略可以为会话提供额外的权限。您可以使用将会话指定为允许的主体的基于资源的策略访问资源。

下图形象地展示了各策略之间如何相互影响，以确定调用 GetFederationToken 所返回的临时安全凭证的权限。



示例：使用 GetFederationToken 分配权限

您可以将 GetFederationToken API 操作与不同类型的策略结合使用。以下是一些示例。

策略已附加到 IAM 用户

在此示例中，您拥有一个基于浏览器的客户端应用程序，该程序依赖于两个后端 Web 服务。一个后端服务是您自己的身份验证服务器，该服务器使用您自己的身份系统对客户端应用程序进行验证。另一个后端服务是一项 Amazon 服务，该服务用于提供此客户端应用程序的部分功能。您的服务器对该客户端应用程序进行身份验证，并创建或检索相应的权限策略。之后，您的服务器调用 `GetFederationToken` API 来获取临时安全凭证，并将这些凭证返回给客户端应用程序。接下来，客户端应用程序就可以借助该临时安全凭证向 Amazon 服务直接发出请求。这一架构允许客户端应用程序在不嵌入长期 Amazon 凭证的情况下发出 Amazon 请求。

您的身份验证服务器使用名为 `token-app` 的 IAM 用户的长期安全凭证调用 `GetFederationToken` API。但是，长期 IAM 用户凭证保留在您的服务器上，从不分发给客户端。下面的示例策略将被附加至 `token-app` IAM 用户，其中定义了您的联合身份用户（客户端）需要的一组最广泛的权限。请注意，您的身份验证服务器需要 `sts:GetFederationToken` 权限才能获取联合身份用户的临时安全凭证。

Note

Amazon 提供了一个实现这一目的的 Java 示例应用程序，您可从此处下载：[身份注册令牌售卖机 - Java Web 示例应用程序](#)。

Example 附加到 IAM 用户 `token-app` 并调用 `GetFederationToken` 的示例策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:GetFederationToken",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "dynamodb:ListTables",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "sqs:ReceiveMessage",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "sns:ListSubscriptions",
      "Resource": "*"
    }
  ]
}
```

上述策略为 IAM 用户授予多个权限。不过，该策略本身不会为联合身份用户授予任何权限。如果该 IAM 用户调用 `GetFederationToken`，并且未将策略作为该 API 调用的参数传递，则生成的联合身份用户没有任何有效的权限。

作为参数传递的会话策略

要确保为联合身份用户分配相应的权限，最常用的方法是在 `GetFederationToken` API 调用中传递会话策略。让我们进一步阐述上述示例，并假设使用 IAM 用户 `token-app` 的凭证调用了 `GetFederationToken`。然后，假设将以下会话策略作为该 API 调用的参数进行传递。生成的联合身份用

户有权列出名为 `productionapp` 的 Amazon S3 存储桶的内容。用户无法对 `productionapp` 存储桶中的项目执行 Amazon S3 `GetObject`、`PutObject`、和 `DeleteObject` 操作。

将为联合身份用户分配这些权限，因为这些权限是 IAM 用户策略与您传递的会话策略的交集。

联合身份用户无法在 Amazon SNS、Amazon SQS、Amazon DynamoDB，或任何 S3 存储桶（`productionapp` 除外）中执行操作。这些操作会被拒绝，即使这些权限被授给了与 `GetFederationToken` 调用关联的 IAM 用户。

Example 作为 **`GetFederationToken`** API 调用参数传递的示例会话策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::productionapp"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3>DeleteObject"
      ],
      "Resource": ["arn:aws:s3:::productionapp/*"]
    }
  ]
}
```

基于资源的策略

某些 Amazon 资源支持基于资源的策略，这些策略提供了另一种直接向联合身份用户授权的机制。只有一部分 Amazon 服务支持基于资源的策略。例如，Amazon S3 具有存储桶，Amazon SNS 有主题，Amazon SQS 具有队列，您可以向这些内容附加策略。有关支持基于资源的策略的所有服务的列表，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)并回顾表的“基于资源的策略”列。您可以使用基于资源的策略将权限直接分配给联合身份用户。为此，请在基于资源的策略的 `Principal` 元素中指定联合身份用户的 Amazon Resource Name (ARN)。以下示例说明了这一点，并使用名为 `productionapp` 的 S3 存储桶进一步阐述上述示例。

以下基于资源的策略附加到存储桶。此存储桶策略允许名为 Carol 的联合身份用户访问该存储桶。当之前介绍的示例策略被附加至 `token-app` IAM 用户时，名为 Carol 的联合身份用户将具有对名为 `productionapp` 的存储桶执行 `s3:GetObject`、`s3:PutObject` 和 `s3>DeleteObject` 操作的权限。即使未将任何会话策略作为 `GetFederationToken` API 调用的参数传递，Carol 也能够执行此操作。这是因为，在此示例中，以下基于资源的策略已向名为 Carol 的联合身份用户明确授予权限。

请记住，只有向 IAM 用户和联合身份用户都进行显式授权时，该联合身份用户才具有这些权限。另外，也可通过在策略的 `Principal` 元素中对联合用户进行显式命名的基于资源的策略授予这些权限（在账户内），如下例所示。

Example 允许访问联合身份用户的示例存储桶策略

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Principal": {"AWS": "arn:aws:sts::account-id:federated-user/Carol"},
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:PutObject",
      "s3>DeleteObject"
    ]
  }
}
```

```
    ],  
    "Resource": ["arn:aws:s3:::productionapp/*"]  
  }  
}
```

有关如何评估策略的更多信息，请参阅[策略评估逻辑](#) (p. 972)。

GetSessionToken 的权限

调用 GetSessionToken API 操作或 get-session-token CLI 命令主要发生在用户必须使用 Multi-Factor Authentication (MFA) 进行身份验证时。可以编写一条策略，只允许特定操作，且仅当这些操作是由经过 MFA 身份验证的用户请求时，才予以放行。为成功通过 MFA 身份验证检查，用户必须先调用 GetSessionToken，并包含可选的 SerialNumber 和 TokenCode 参数。如果用户成功通过 MFA 设备的身份验证，则 GetSessionToken API 调用返回的凭证将包含 MFA 上下文。此上下文指示用户已使用 MFA 进行了身份验证，并已获得了需要 MFA 身份验证的 API 操作的授权。

GetSessionToken 所需的权限

用户无需任何权限即可获得会话令牌。GetSessionToken 操作旨在使用 MFA 验证用户身份。您不能使用策略来控制身份验证操作。

要授予执行大多数 Amazon 操作的权限，您可以将具有相同名称的操作添加到策略。例如，要创建用户，您必须使用 CreateUser API 操作、create-user CLI 命令或 Amazon Web Services Management Console。要执行这些操作，您必须具有一个策略，该策略允许您访问 CreateUser 操作。

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

虽然您可以在策略中包含 GetSessionToken 操作，但不会影响用户执行 GetSessionToken 操作的能力。

由 GetSessionToken 授予的权限

如果调用 GetSessionToken 时使用的是 IAM 用户的凭证，则临时安全凭证将具有与该 IAM 用户相同的权限。同样，如果使用 Amazon Web Services 账户根用户凭证调用 GetSessionToken，临时安全凭证将拥有根用户权限。

Note

我们建议您不要使用根用户凭证来调用 GetSessionToken。相反，请遵循我们的[最佳实践](#) (p. 751)，并创建具有所需权限的 IAM 用户。然后使用这些 IAM 用户执行与 Amazon 的日常交互工作。

您在调用 GetSessionToken 时获得的临时凭证具有以下功能和限制：

- 您可以通过将凭证传递到 <https://signin.aws.amazon.com/federation> 上的联合身份验证单一登录终端节点来访问 Amazon Web Services Management Console。有关更多信息，请参阅[使自定义身份代理能够访问 Amazon 控制台](#) (p. 201)。
- 您无法使用凭证调用 IAM 或 Amazon STS API 操作。您可以使用它们来调用其他 Amazon 服务的 API 操作。

请参阅 [比较 Amazon STS API 操作 \(p. 310\)](#)，将此 API 操作及其限制和功能与创建临时安全凭证的其他 API 操作比较。

有关使用 `GetSessionToken` 进行受 MFA 保护的 API 访问的更多信息，请参阅[配置受 MFA 保护的 API 访问 \(p. 129\)](#)。

禁用临时安全凭证的权限

临时安全凭证在过期之前一直有效，并且无法撤消。但是，由于每次使用凭证发出 Amazon 请求时都会评估权限，因此，即使凭证已经签发，您仍可通过更改凭证的权限来实现撤消凭证的效果。如果您删除了临时安全凭证的所有权限，则使用这些凭证的后续 Amazon 请求都将失败。以下章节将讲解更改或撤消分配给临时安全凭证的权限的机制。

Note

在更新现有策略权限时或在向用户或资源应用新策略时，可能需要几分钟时间，策略更新才能生效。

主题

- [拒绝访问临时安全凭证的创建者 \(p. 328\)](#)
- [根据名称拒绝访问临时安全证书 \(p. 329\)](#)
- [拒绝访问在特定时间之前签发的临时安全凭证 \(p. 329\)](#)

拒绝访问临时安全凭证的创建者

要更改或删除分配给临时安全凭证的权限，您可更改或删除与这些凭证创建者关联的权限。凭证的创建者由用于获取该凭证的 Amazon STS API 决定。以下章节将讲解更改或删除与此创建者关联的权限的机制。

拒绝访问由 `AssumeRole`、`AssumeRoleWithSAML` 或 `AssumeRoleWithWebIdentity` 创建的证书

要更改或删除为调用 `AssumeRole`、`AssumeRoleWithSAML` 或 `AssumeRoleWithWebIdentity` API 操作而获得的临时安全凭证分配的权限，请编辑或删除为所代入角色定义权限的角色权限策略。通过代入角色获得的临时安全凭证所拥有的权限绝不会多于所代入角色的权限策略中定义的权限，且每次使用临时安全凭证发出 Amazon 请求时都会评估分配给其的权限。在编辑或删除角色的权限策略时，所做的更改影响与该角色关联的所有临时安全凭证（包括在更改该角色的权限策略之前颁发的凭证）的权限。通过执行[撤消 IAM 角色临时安全凭证 \(p. 259\)](#)中的步骤可以立即撤消对会话的所有权限。

有关编辑角色权限策略的更多信息，请参阅[修改角色 \(p. 261\)](#)。

拒绝访问由 `GetFederationToken` 或 `GetSessionToken` 创建的证书

要更改或删除分配给通过调用 `GetFederationToken` 或 `GetSessionToken` API 操作获得的临时安全凭证的权限，可编辑或删除其凭证用于调用 `GetFederationToken` 或 `GetSessionToken` 的 IAM 用户的关联策略。通过调用 `GetFederationToken` 或 `GetSessionToken` 获得的临时安全凭证所拥有的权限绝不会多于其凭证用于获取这些权限的 IAM 用户的权限。此外，每次使用临时安全凭证发出 Amazon 请求时都会评估分配给该凭证的权限。请务必注意，编辑或删除 IAM 用户的权限时，所做出的更改会影响该 IAM 用户及由该用户创建的所有临时安全凭证。

Important

您无法更改 Amazon Web Services 账户 根用户的权限。同样，以根用户身份登录时，您无法更改调用 `GetFederationToken` 或 `GetSessionToken` 创建的临时安全凭证的权限。因此，我们建议您不要以根用户身份调用 `GetFederationToken` 或 `GetSessionToken`。

有关如何更改或删除其凭证用于调用 `GetFederationToken` 或 `GetSessionToken` 的 IAM 用户的关联策略，请参阅[管理 IAM 策略 \(p. 437\)](#)。

根据名称拒绝访问临时安全证书

您可以拒绝访问临时安全凭证而不影响创建该凭证的 IAM 用户或角色的权限。方法是：在基于资源的策略的 `Principal` 元素中指定临时安全凭证的 Amazon Resource Name (ARN)。(只有一部分 Amazon 服务支持基于资源的策略。)

拒绝访问联合身份用户

例如，假设您有一个名为 `token-app` 的 IAM 用户，且其凭证用于调用 `GetFederationToken`。`GetFederationToken` API 调用使得临时安全凭证关联至名为 Bob 的联合身份用户 (该联合身份用户的名称来自该 API 调用的 `Name` 参数)。要拒绝联合身份用户 Bob 访问名为 `EXAMPLE-BUCKET` 的 S3 存储桶，您可将以下示例存储桶策略附加到 `EXAMPLE-BUCKET`。请务必注意，这只影响该联合身份用户的 Amazon S3 权限，授予该联合身份用户的所有其他权限均保持不变。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Principal": {"AWS": "arn:aws:sts::account-id:federated-user/Bob"},
    "Effect": "Deny",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::EXAMPLE-BUCKET"
  }
}
```

您可在该存储桶策略的 `GetFederationToken` 元素中指定拥有 `Principal` 调用凭证的 IAM 用户的 ARN，而非指定该联合身份用户。在此示例中，上述策略的 `Principal` 元素如下所示：

```
"Principal": {"AWS": "arn:aws:iam::account-id:user/token-app"}
```

请务必注意，在该策略中指定 IAM 用户 `token-app` 的 ARN 会导致对由 `token-app` 创建的所有联合身份用户 (而不仅仅是名为 Bob 的联合身份用户) 的访问遭拒。

拒绝访问担任角色的用户

也可指定通过担任角色创建的临时安全凭证的 ARN。区别在于在基于资源的策略的 `Principal` 元素中使用的语法。例如，某用户担任名为 `Accounting-Role` 的角色并指定 `RoleSessionName` 为 `Mary` (`RoleSessionName` 是 `AssumeRole` API 调用的一个参数)。要拒绝访问由该 API 调用产生的临时安全凭证，基于资源的策略的 `Principal` 元素如下所示：

```
"Principal": {"AWS": "arn:aws:sts::account-id:assumed-role/Accounting-Role/Mary"}
```

此外，您还可以在基于资源的策略的 `Principal` 元素中指定该 IAM 角色的 ARN。在这种情况下，策略将导致拒绝对与名为 `Accounting-Role` 的角色关联的所有临时安全凭证的访问。

```
"Principal": {"AWS": "arn:aws:iam::account-id:role/Accounting-Role"}
```

拒绝访问在特定时间之前签发的临时安全凭证

可以只拒绝对在特定时间和日期前创建的临时安全凭证的访问。方法是：在某策略的 `aws:TokenIssueTime` 元素的 `Condition` 键中指定值。以下策略显示了一个示例。您可将类似以下示例的策略关联至创建该临时安全凭证的 IAM 用户。仅当 `aws:TokenIssueTime` 的值早于指定的日期和时间时，该策略才会拒绝所有权限。`aws:TokenIssueTime` 的值对应于临时安全凭证的确切创建时间。`aws:TokenIssueTime` 值仅存在于使用临时安全凭证签署的 Amazon 请求的上下文中，因此，该策略中的 `Deny` 语句不会影响使用该 IAM 用户的长期凭证签署的请求。

以下策略还可以附加到角色。在这种情况下，该策略只会影响由该角色在指定日期和时间之前创建的临时安全凭证。如果凭证是该角色在指定日期和时间之后创建的，则该策略中的 `Condition` 元素将评估为 `false`，因此，`Deny` 语句将不起作用。

Example 根据签发时间拒绝临时凭证所有权限的示例策略

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {"DateLessThan": {"aws:TokenIssueTime": "2014-05-07T23:47:00Z"}}
  }
}
```

以这种方式调用会话的有效用户必须获得临时凭证，新会话才能继续工作。请注意，在凭证过期前，Amazon CLI 会缓存凭证。要强制 CLI 删除并刷新已失效的缓存证书，请运行以下命令之一：

Linux、MacOS 或 Unix

```
$ rm -r ~/.aws/cli/cache
```

Windows

```
C:\> del /s /q %UserProfile%\aws\cli\cache
```

授予创建临时安全凭证的权限

默认情况下，IAM 用户无权为联合身份用户和角色创建临时安全凭证。您必须使用策略来向用户提供这些权限。虽然您可以直接向用户授予权限，但我们强烈建议您向组授予权限。这样可以使权限管理轻松得多。如果某个用户不再需要执行与权限关联的任务时，您只需从组中将其删除。如果其他用户需要执行这些任务，请将这些用户添加到组以授予权限。

要向 IAM 组授予为联合身份用户或角色创建临时安全凭证的权限，应附加一个策略，该策略授予以下一项或两项权限：

- 对于要访问 IAM 角色的联合身份用户，请授予对 Amazon STS AssumeRole 的访问权限。
- 对于无需角色的联合身份用户，请授予对 Amazon STS GetFederationToken 的访问权限。

有关 AssumeRole 和 GetFederationToken API 操作之间的差异的更多信息，请参阅[请求临时安全凭证 \(p. 302\)](#)。

IAM 用户也可以调用 [GetSessionToken](#) 以创建临时安全凭证。用户无需任何权限即可调用 GetSessionToken。此操作旨在使用 MFA 验证用户身份。您不能使用策略来控制身份验证。这意味着，您不能阻止 IAM 用户调用 GetSessionToken 来创建临时凭证。

Example 为授予担任角色的权限的示例策略

以下示例策略为 Amazon 账户 123123123123 中的 UpdateApp 角色授予调用 AssumeRole 的权限。在使用 AssumeRole 时，代表联合身份用户创建安全凭证的用户（或应用程序）无法委派尚未在角色权限策略中指定的任何权限。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::123123123123:role/UpdateAPP"
  }]
}
```


Example 示例策略，该策略授予为联合身份用户创建临时安全凭证的权限

以下示例策略授予访问 `GetFederationToken` 的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sts:GetFederationToken",
    "Resource": "*"
  }]
}
```

Important

使用 `GetFederationToken` 向 IAM 用户授予为联合身份用户创建临时安全凭证的权限时应注意，这将允许这些用户委派自己的权限。有关跨 IAM 用户和 Amazon 账户委派权限的更多信息，请参阅[委派访问权限的策略示例 \(p. 233\)](#)。有关控制临时安全凭证权限的详细信息，请参阅[控制临时安全凭证的权限 \(p. 313\)](#)。

Example 向用户授予为联合身份用户创建临时安全凭证的有限权限的示例策略

在让 IAM 用户调用 `GetFederationToken` 时，最佳实践是限制 IAM 用户可以委派的权限。举例来说，以下策略展示如何让 IAM 用户仅为其名称以 `Manager` 开头的联合身份用户创建临时安全凭证。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sts:GetFederationToken",
    "Resource": ["arn:aws:sts::123456789012:federated-user/Manager*"]
  }]
}
```

在 Amazon 区域中管理 Amazon STS

默认情况下，Amazon Security Token Service (Amazon STS) 作为全球服务提供，并且所有 Amazon STS 请求发送到单个终端节点 (<https://sts.amazonaws.com>)。Amazon 建议使用区域 Amazon STS 终端节点而不是全球终端节点以减少延迟，建立冗余以及提高会话令牌有效性。

- 减少延迟 - 通过向在地理位置离您较近的服务和应用程序的终端节点发出 Amazon STS 调用，可以缩短访问 Amazon STS 服务时的响应时间并减少延迟。
- 建立冗余 - 可以在您的应用程序中添加代码，以将 Amazon STS API 调用切换到不同的区域。这会确保在第一个区域停止响应时，您的应用程序继续运行。此冗余不是自动的；您必须在您的代码中构建此功能。
- 提高会话令牌有效性 - 来自区域 Amazon STS 终端节点的会话令牌在所有 Amazon 区域中都有效。来自全球 STS 终端节点的会话令牌仅在默认启用的 Amazon 区域中有效。如果打算为您的账户启用新的区域，您可以使用来自区域 STS 终端节点的会话令牌。如果选择使用全球终端节点，您必须更改全球终端节点的 STS 会话令牌的区域兼容性。这样做可以确保令牌在所有 Amazon 区域中都有效。

管理全球终端节点会话令牌

默认情况下，为所有 Amazon 服务中的操作启用了大多数 Amazon 区域。将自动激活这些区域以用于 Amazon STS。必须手动启用某些区域，例如，亚太区域（香港）。要了解启用和禁用 Amazon 区域的更多信息，请参阅 Amazon 一般参考中的[管理 Amazon 区域](#)。在启用这些 Amazon 区域时，将自动激活这些区域以用于 Amazon STS。您无法为禁用的区域激活 STS 终端节点。在所有 Amazon 区域中都有效的令牌包含的字符数比在默认启用的区域中有效的令牌多。更改该设置可能会影响临时存储令牌的现有系统。

您可以使用 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 更改该设置。

更改全球终端节点的会话令牌的区域兼容性 (控制台)

1. 以有权执行 IAM 管理任务的根用户或 IAM 用户身份登录。要更改会话令牌的兼容性，您必须具有允许 `iam:SetSecurityTokenServicePreferences` 操作的策略。
2. 打开 [IAM 控制台](#)。在导航窗格中，选择账户设置。
3. 如有必要，请展开 Security Token Service (STS) 部分。在 Global endpoint (全球终端节点) 旁边的第一个表中，Region compatibility of session tokens (会话令牌的区域兼容性) 列指示 `Valid only in AWS Regions enabled by default`。选择 `Change` (更改)。
4. 在 `Change region compatibility of session tokens for global endpoint` (更改全球终端节点的会话令牌的区域兼容性) 对话框中，选择 `Valid in all Amazon Regions` (在所有亚马逊云科技区域中都有效)。然后选择 `Save changes` (保存更改)。

Note

在所有 Amazon 区域中都有效的令牌包含的字符数比在默认启用的区域中有效的令牌多。更改该设置可能会影响临时存储令牌的现有系统。

更改全球终端节点的会话令牌的区域兼容性 (Amazon CLI)

设置会话令牌版本。版本 1 令牌仅在默认启用的 Amazon 区域中有效。这些令牌不适用于手动启用的区域，例如，亚太地区 (香港)。版本 2 令牌在所有区域中都有效。不过，版本 2 令牌包含更多字符，可能会影响临时存储令牌的系统。

- `aws iam set-security-token-service-preferences`

更改全球终端节点的会话令牌的区域兼容性 (Amazon API)

设置会话令牌版本。版本 1 令牌仅在默认启用的 Amazon 区域中有效。这些令牌不适用于手动启用的区域，例如，亚太地区 (香港)。版本 2 令牌在所有区域中都有效。不过，版本 2 令牌包含更多字符，可能会影响临时存储令牌的系统。

- `SetSecurityTokenServicePreferences`

在 Amazon 区域中激活和停用 Amazon STS

在为区域激活 STS 终端节点时，Amazon STS 可能会向您的账户中发出 Amazon STS 请求的用户和角色颁发临时凭证。然后，可以在默认启用或手动启用的任何区域中使用这些凭证。您必须在生成临时凭证的账户中激活该区域。在发出请求时，用户是登录到同一账户还是不同账户并不重要。

例如，假设账户 A 中的用户要向 STS 区域终端节点 `https://sts.us-west-2.amazonaws.com` 发送 `sts:AssumeRole` API 请求。该请求旨在为账户 B 中名为 `Developer` 的角色获取临时凭证。由于该请求旨在为账户 B 中的实体创建凭证，因此，账户 B 必须激活 `us-west-2` 区域。账户 A (或任何其他账户) 中的用户可以调用 `us-west-2` 终端节点，以便为账户 B 请求凭证，而无论是否在其账户中激活了该区域。

Note

活动区域适用于该账户中使用临时凭证的每个用户。要控制哪些 IAM 用户或角色可以访问区域，请在权限策略中使用 `aws:RequestedRegion` (p. 1006) 条件键。

在默认启用的区域中激活或停用 Amazon STS (控制台)

1. 以有权执行 IAM 管理任务的根用户或 IAM 用户身份登录。
2. 打开 [IAM 控制台](#)，然后在导航窗格中选择 [Account settings](#) (账户设置)。

3. 如有必要，请展开 Security Token Service (STS)，找到要激活的区域，然后选择激活或停用。对于必须启用的区域，当您启用相关区域时，我们会自动激活 STS。启用区域后，Amazon STS 对于该区域将始终处于活动状态，您无法将其停用。要了解如何启用区域，请参阅Amazon指南参考中的[管理 Amazon 区域](#)。

写入用于使用 Amazon STS 区域的代码

在激活区域后，您可以将 Amazon STS API 调用定向到该区域。以下 Java 代码段说明了如何配置 AWSSecurityTokenService 对象，以向欧洲（爱尔兰）(eu-west-1) 区域发出请求。

```
EndpointConfiguration regionEndpointConfig = new EndpointConfiguration("https://sts.eu-west-1.amazonaws.com", "eu-west-1");
AWSSecurityTokenService stsRegionalClient = AWSSecurityTokenServiceClientBuilder.standard()
    .withCredentials(credentials)
    .withEndpointConfiguration(regionEndpointConfig)
    .build();
```

Amazon STS 建议您调用区域终端节点。要了解如何手动启用区域，请参阅 Amazon 一般参考中的[管理 Amazon 区域](#)。

在该示例中，第一行将名为 regionEndpointConfig 的 EndpointConfiguration 对象实例化，并将终端节点的 URL 和区域作为参数传递。

要了解如何使用 Amazon SDK 的环境变量设置 Amazon STS 区域端点，请参阅 Amazon SDK 和工具参考指南中的[Amazon STS 区域化端点](#)。

有关所有其他的语言和编程环境组合，请参阅[相关开发工具包的文档](#)。

区域和终端节点

下表列出了区域及其终端节点。其中指示哪些是默认激活的，哪些是可激活或停用的。

区域名称	端点	默认情况下激活	手动激活/停用
--全球--	sts.amazonaws.com	✔ 是	✘ 不支持
美国东部（俄亥俄州）	sts.us-east-2.amazonaws.com	✔ 是	✔ 是
美国东部（弗吉尼亚州北部）	sts.us-east-1.amazonaws.com	✔ 是	✘ 不支持
美国西部（北加利福尼亚）	sts.us-west-1.amazonaws.com	✔ 是	✔ 是
美国西部（俄勒冈州）	sts.us-west-2.amazonaws.com	✔ 是	✔ 是
非洲（开普敦）	sts.af-south-1.amazonaws.com	✘ 否 ¹	✘ 不支持
亚太地区（香港）	sts.ap-east-1.amazonaws.com	✘ 否 ¹	✘ 不支持
亚太地区（雅加达）	sts.ap-southeast-3.amazonaws.com	✘ 否 ¹	✘ 不支持
亚太地区（孟买）	sts.ap-south-1.amazonaws.com	✔ 是	✔ 是
亚太地区（大阪）	sts.ap-northeast-3.amazonaws.com	✔ 是	✔ 是

区域名称	端点	默认情况下激活	手动激活/停用
亚太地区 (首尔)	sts.ap-northeast-2.amazonaws.com	✔ 是	✔ 是
亚太地区 (新加坡)	sts.ap-southeast-1.amazonaws.com	✔ 是	✔ 是
亚太地区 (悉尼)	sts.ap-southeast-2.amazonaws.com	✔ 是	✔ 是
亚太地区 (东京)	sts.ap-northeast-1.amazonaws.com	✔ 是	✔ 是
加拿大 (中部)	sts.ca-central-1.amazonaws.com	✔ 是	✔ 是
中国 (北京)	sts.cn-north-1.amazonaws.com.cn	✔ 是 ²	✘ 不支持
中国 (宁夏)	sts.cn-northwest-1.amazonaws.com.cn	✔ 是 ²	✔ 是
欧洲 (法兰克福)	sts.eu-central-1.amazonaws.com	✔ 是	✔ 是
欧洲 (爱尔兰)	sts.eu-west-1.amazonaws.com	✔ 是	✔ 是
欧洲 (伦敦)	sts.eu-west-2.amazonaws.com	✔ 是	✔ 是
欧洲 (米兰)	sts.eu-south-1.amazonaws.com	✘ 否 ¹	✘ 不支持
欧洲 (巴黎)	sts.eu-west-3.amazonaws.com	✔ 是	✔ 是
欧洲 (斯德哥尔摩)	sts.eu-north-1.amazonaws.com	✔ 是	✔ 是
中东 (巴林)	sts.me-south-1.amazonaws.com	✘ 否 ¹	✘ 不支持
中东 (阿联酋)	sts.me-central-1.amazonaws.com	✘ 否 ¹	✘ 不支持
南美洲 (圣保罗)	sts.sa-east-1.amazonaws.com	✔ 是	✔ 是

您必须 [启用区域](#) 才能使用它。这会激活 STS。您无法在这些区域中手动激活或停用 STS。

²要在中国使用 Amazon，您需要中国 Amazon 的特定账户和凭证。

Amazon CloudTrail 和区域终端节点

与对区域服务的任何调用一样，将在 Amazon CloudTrail 中记录对区域终端节点 (如 `us-east-2.amazonaws.com`) 的调用。对全球终端节点 (如 `sts.amazonaws.com`) 的调用会记录为对全球服务的调用。有关更多信息，请参阅 [使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用 \(p. 340\)](#)。

使用 Amazon STS 接口 VPC 终端节点

如果您使用 Amazon Virtual Private Cloud (Amazon VPC) 托管 Amazon 资源，则可以在您的 VPC 和 Amazon STS 之间建立私有连接。您可以使用此连接实现 Amazon STS 与您的 VPC 上的资源的通信而不用访问公共 Internet。

Amazon VPC 是一项 Amazon 服务，可用来启动在虚拟网络中定义的 Amazon 资源。借助 VPC，您可以控制您的网络设置，如 IP 地址范围、子网、路由表和网络网关。要将 VPC 连接到 Amazon STS，请为

Amazon STS 定义一个接口 VPC 终端节点。该终端节点提供了到 Amazon STS 的可靠、可扩展的连接，无需 Internet 网关、网络地址转换 (NAT) 实例或 VPN 连接。有关更多信息，请参阅 [Amazon VPC 用户指南](#) 中的 [什么是 Amazon VPC ?](#)。

接口 VPC 终端节点由 Amazon PrivateLink 提供支持，后者是一种 Amazon 技术，可将弹性网络接口与私有 IP 地址结合使用来支持 Amazon 服务之间的私有通信。有关更多信息，请参阅最新 - 适用于 Amazon 服务的 [Amazon PrivateLink](#)。

以下步骤适用于 Amazon VPC 的用户。有关更多信息，请参阅 Amazon VPC 用户指南中的 [Amazon VPC 入门](#)。

可用性

Amazon STS 当前在以下区域中支持 VPC 终端节点：

- 美国东部 (俄亥俄)
- 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 (俄勒冈)
- Africa (Cape Town)
- Asia Pacific (Hong Kong)
- Asia Pacific (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seoul)
- 亚太地区 (新加坡)
- 亚太地区 (悉尼)
- 亚太区域 (东京)
- 加拿大 (中部)
- 中国 (北京)
- China (Ningxia)
- 欧洲 (法兰克福)
- 欧洲 (爱尔兰)
- 欧洲 (伦敦)
- 欧洲 (米兰)
- Europe (Paris)
- Europe (Stockholm)
- Middle East (Bahrain)
- 南美洲 (圣保罗)
- Amazon GovCloud (美国东部)
- Amazon GovCloud (美国西部)

为 Amazon STS 创建 VPC 终端节点

要开始将您的 Amazon STS 与 VPC 一起使用，请为 Amazon STS 创建接口 VPC 终端节点。有关更多信息，请参阅 Amazon VPC 用户指南中的 [创建接口端点](#)。

在创建 VPC 终端节点后，您必须使用匹配的区域终端节点发送 Amazon STS 请求。Amazon STS 建议您同时使用 `setRegion` 和 `setEndpoint` 方法以对区域终端节点进行调用。您可以仅将 `setRegion` 方法用于

手动启用的区域，例如，亚太区域（香港）。在这种情况下，调用将定向到 STS 区域终端节点。要了解如何手动启用区域，请参阅 Amazon 一般参考中的[管理 Amazon 区域](#)。如果仅将 `setRegion` 方法用于默认启用的区域，调用将定向到全球终端节点 <https://sts.amazonaws.com>。

在您使用区域终端节点时，Amazon STS 使用公有终端节点或私有接口 VPC 终端节点（二者中正在使用的那个）调用其他 Amazon 服务。例如，假设您已为 Amazon STS 创建了接口 VPC 终端节点，并且已经从位于 VPC 中的资源的 Amazon STS 请求了临时凭证。在这种情况下，这些凭证默认开始流经接口 VPC 终端节点。有关使用 Amazon STS 发出区域请求的更多信息，请参阅在[Amazon 区域中管理 Amazon STS](#) (p. 331)。

使用持有者令牌

某些 Amazon 服务需要您有权获取 Amazon STS 服务持有者令牌，然后您才能以编程方式访问它们的资源。这些服务支持一种协议，该协议要求您使用持有者令牌，而不是使用传统的[签名版本 4 签名请求](#)。当您执行需要持有者令牌的 Amazon CLI 或 Amazon API 操作时，Amazon 服务将代表您请求持有者令牌。该服务将为您提供令牌，然后您可以使用该令牌在该服务中执行后续操作。

Amazon STS 服务持有者令牌包括来自您的原始主体身份验证的信息，这些信息可能会影响您的权限。这些信息可以包括主体标签、会话标签和会话策略。令牌的访问密钥 ID 以 `ABIA` 前缀开头。这可帮助您在 CloudTrail 日志中识别使用服务持有者令牌执行的操作。

Important

持有者令牌只能用于对生成它的服务的调用以及生成它的区域。您不能使用持有者令牌在其他服务或区域中执行操作。

支持持有者令牌的服务示例是 Amazon CodeArtifact。在使用 NPM、Maven 或 PIP 等程序包管理器与 Amazon CodeArtifact 进行交互之前，您必须调用 `aws codeartifact get-authorization-token` 操作。此操作返回可用于执行 Amazon CodeArtifact 操作的持有者令牌。或者，您可以使用 `aws codeartifact login` 命令完成相同的操作，然后自动配置您的客户端。

如果您在为您生成持有者令牌的 Amazon 服务中执行操作，则您的 IAM policy 中必须具有以下权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowServiceBearerToken",
      "Effect": "Allow",
      "Action": "sts:GetServiceBearerToken",
      "Resource": "*"
    }
  ]
}
```

有关服务持有者令牌的示例，请参阅《Amazon CodeArtifact 用户指南》中的[为 Amazon CodeArtifact 使用基于身份的策略](#)。

使用临时凭证的示例应用程序

您可以使用 Amazon Security Token Service (Amazon STS) 创建可控制对您的 Amazon 资源的访问的临时安全凭证，并将这些凭证提供给可信用户。有关 Amazon STS 的更多信息，请参阅[IAM 临时安全凭证](#) (p. 300)。要了解如何使用 Amazon STS 管理临时安全凭证，可下载以下示例应用程序，其中实现完整的示例方案：

- [Identity Federation Sample Application for an Active Directory Use Case](#)。说明了如何使用与 Active Directory (.NET/C#) 中定义的用户关联的权限来颁发用于访问 Amazon S3 文件和存储桶的临时安全凭证。

- [Amazon 管理控制台联合身份验证代理示例使用案例](#)。说明了如何创建启用单一登录 (SSO) 的自定义联合代理，以使现有 Active Directory 用户能够登录到 Amazon Web Services Management Console (.NET/C#)。
- [将 Shibboleth 与 Amazon Identity and Access Management 集成](#)。介绍如何使用 Shibboleth 和 SAML (p. 176) 为用户提供对 Amazon Web Services Management Console 的单点登录 (SSO) 访问权限。

Web 联合身份验证示例

下面的示例应用程序说明了如何使用 Login with Amazon、Amazon Cognito、Facebook 或 Google 等提供商的 Web 联合身份验证。您可以将从这些提供商处获得的身份验证信息作为临时 Amazon 安全凭证以访问 Amazon 服务。

- [Amazon Cognito 教程](#) — 建议您将 Amazon Cognito 与适用于移动开发的 Amazon SDK 搭配使用。Amazon Cognito 是管理适用于移动应用程序的身份的最简单方式，它提供了同步和跨设备身份等额外功能。有关 Amazon Cognito 的更多信息，请参阅 Amazon Mobile SDK for Android 开发人员指南中的 [Amazon Cognito 身份](#) 和 Amazon Mobile SDK for iOS 开发人员指南中的 [使用 Amazon Cognito Identity 验证用户身份](#)。
- [Web Identity Federation Playground](#)。该网站提供 [Web 联合身份验证 \(p. 171\)](#) 和 AssumeRoleWithWebIdentity API 的交互式演示。

有关临时安全凭证的其他资源

以下场景和应用可指导您使用临时安全凭证：

- [关于 Web 联合身份验证 \(p. 171\)](#)。本节讨论如何在使用 Web 联合身份验证和 AssumeRoleWithWebIdentity API 时配置 IAM 角色。
- [配置受 MFA 保护的 API 访问 \(p. 129\)](#)。本主题介绍如何使用角色要求多重身份验证 (MFA)，以保护您账户中的敏感 API 操作。
- [用于身份注册的令牌售卖机](#)。该示例 Java Web 应用程序使用 GetFederationToken API 向远程客户端提供临时安全凭证。

有关 Amazon 中的策略和权限的更多信息，请参阅以下主题：

- [适用于 Amazon 资源的 Access Management \(p. 353\)](#)
- [策略评估逻辑 \(p. 972\)](#)。
- Amazon Simple Storage Service 用户指南中的 [管理对 Amazon S3 资源的访问权限](#)。
- 要了解您信任区域之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅 [什么是 IAM Access Analyzer ?](#)。

Amazon 账户根用户

当您首次创建 Amazon Web Services (Amazon) 账户时，最初使用的是一个对账户中所有 Amazon 服务和资源具有完全访问权限的身份。此身份称作 Amazon 账户根用户。您可以使用在创建账户所用的电子邮件地址和密码以根用户身份登录。

Important

在北京和宁夏区域，没有根用户的概念。所有用户都是 IAM 用户，包括创建 Amazon 账户的用户。

您可以创建、轮换、禁用或删除 Amazon Web Services 账户 账户根用户的访问密钥 (访问密钥 ID 和秘密访问密钥)。还可以更改根用户密码。拥有您的根用户账户凭证的任何人都可以无限制地访问您 Amazon 账户中的所有资源，包括账单信息。

当您创建访问密钥时，您会将访问密钥 ID 和秘密访问密钥创建一个集。在创建访问密钥的过程中，Amazon 仅允许您查看和下载一次访问密钥的秘密访问密钥部分。如果您未下载或丢失了访问密钥，则可删除访问密钥，然后创建新的访问密钥。您可以使用 [IAM 控制台](#)、Amazon CLI 或 Amazon API 创建根用户访问密钥。

新创建的访问密钥的状态为已激活，这意味着，您可以使用访问密钥进行 CLI 和 API 调用。每个 IAM 用户 [限于两个访问密钥](#)，这在您需要 [轮换访问密钥](#) 时很有用。您还可以为根用户分配多达两个访问密钥。若您禁用了访问密钥，您就不能将其用于 API 调用。非活动的密钥仍将计入您的限制。您随时可以创建或删除访问密钥。不过，当您删除访问密钥时，意味着永久删除且无法恢复。

您可以前往 [Security Credentials](#) 页面更改电子邮件地址和密码。您还可以选择 Amazon 登录页面上的 [Forgot password?](#) 来重置您的密码。

任务

- [创建或删除 Amazon 账户 \(p. 338\)](#)
- [在 Amazon 账户根用户上启用 MFA \(p. 338\)](#)
- [创建根用户的访问密钥 \(p. 338\)](#)
- [删除根用户的访问密钥 \(p. 339\)](#)
- [更改根用户的密码 \(p. 340\)](#)
- [保护根用户的凭证 \(p. 340\)](#)
- [转移根用户拥有者 \(p. 340\)](#)

创建或删除 Amazon 账户

有关更多信息，请参阅 Amazon 知识中心中的以下文章：

- [如何创建和激活 Amazon 账户？](#)
- [如何关闭 Amazon 账户？](#)

在 Amazon 账户根用户上启用 MFA

我们建议您遵循为您的账户启用多重身份验证 (MFA) 的安全最佳实践。由于您的根用户可以在您的账户中执行敏感性操作，因此添加额外一层身份验证可帮助您更好地保护您的账户。有多个 MFA 类型可用。有关启用 MFA 的更多信息，请参阅以下内容：

- [为您的 Amazon 账户根用户启用虚拟 MFA 设备 \(控制台\) \(p. 108\)](#)
- [为 Amazon 账户根用户启用硬件 MFA 设备 \(控制台\) \(p. 117\)](#)

创建根用户的访问密钥

您可以使用 Amazon Web Services Management Console 或 Amazon 编程工具来创建根用户的访问密钥。

创建 Amazon Web Services 账户 根用户的访问密钥 (控制台)

1. 选择 Root user (根用户) 并输入您的 Amazon Web Services 账户 电子邮件地址，以账户拥有者身份登录 [IAM 控制台](#)。在下一页上，输入您的密码。

Note

如果显示了三个文本框，则您之前已使用 [IAM 用户](#) 凭证登录控制台。您的浏览器可能会记住此首选项，并在您每次尝试登录时打开此账户特定的登录页面。您无法使用 IAM 用户登录页面以账户所有者身份进行登录。如果您看到 [IAM 用户登录页面](#)，请选择该页面底部附近的 Sign in using root user email (使用根用户电子邮件登录)。这将返回主登录页面。在该页面中，您可以使用 Amazon Web Services 账户电子邮件地址和密码以根用户登录。

2. 在导航栏上选择您的账户名称，然后选择我的安全凭证。
3. 如果看到有关访问您的 Amazon 账户的安全凭证的警告，选择 Continue to Security Credentials。
4. 展开 Access keys (access key ID and secret access key) 部分。
5. 选择 Create New Access Key。如果此功能已禁用，则必须先删除现有访问密钥中的一个，然后才能创建新密钥。有关更多信息，请参阅 IAM 用户指南中的 [IAM 实体对象限制](#)

一个警告说明，您只有这一次机会可以查看或下载秘密访问密钥。之后无法检索。

- 如果您选择 Show Access Key，您可以从浏览器窗口复制访问密钥 ID 和私有密钥，将其粘贴到其他位置。
 - 如果您选择 Download Key File，则会接收一个包含访问密钥 ID 和私有密钥、名为 rootkey.csv 的文件。将该文件安全保存在某个位置。
6. 当不再需要使用访问密钥时，[我们建议您删除它 \(p. 754\)](#)，或者至少通过选择 Make Inactive (转为非活跃) 将其标记为非活动，以免被误用。

创建根用户的访问密钥 (Amazon CLI 或 Amazon API)

使用以下值之一：

- Amazon CLI : [aws iam create-access-key](#)
- Amazon API : [CreateAccessKey](#)

删除根用户的访问密钥

您可以使用 Amazon Web Services Management Console 删除根用户的访问密钥。

1. 使用您的 Amazon 账户电子邮件地址和密码以 Amazon Web Services 账户根用户身份登录 [Amazon Web Services Management Console](#)。

Note

如果显示了三个文本框，则您之前已使用 [IAM 用户](#) 凭证登录控制台。您的浏览器可能会记住此首选项，并在您每次尝试登录时打开此账户特定的登录页面。您无法使用 IAM 用户登录页面以账户所有者身份进行登录。如果您看到 [IAM 用户登录页面](#)，请选择该页面底部附近的 Sign in using root user email (使用根用户电子邮件登录)。这将返回主登录页面。在该页面中，您可以使用 Amazon 账户电子邮件地址和密码以根用户登录。

2. 在导航栏上选择您的账户名称，然后选择我的安全凭证。
3. 如果看到有关访问您的 Amazon 账户的安全凭证的警告，选择 Continue to Security Credentials。
4. 展开 Access keys (access key ID and secret access key) 部分。
5. 查找要删除的访问密钥，然后在 Actions 列下，选择 Delete。

Note

您可以将访问密钥标记为非活动而不是删除它。这样，您将来可以继续使用它，无需更改密钥 ID 或私有密钥。当它为非活动状态时，任何在 Amazon API 请求中使用它的尝试都会以访问被拒绝的状态失败。

更改根用户的密码

有关如何更改根用户密码的信息，请参阅 [更改 Amazon 账户根用户密码 \(p. 87\)](#)。要更改根用户，您必须使用根用户凭证登录。要查看需要您以根用户身份登录的任务，请参阅 [需要根用户的 Amazon 任务](#)。

保护根用户的凭证

有关为 Amazon Web Services 账户 根用户确保凭证的更多信息，请参阅 [保护您的根用户凭证，不要将其用于日常任务 \(p. 753\)](#)。

转移根用户拥有者

要转移根用户的所有权，请参阅 [如何将我的 Amazon 账户转移到另一个人或企业？](#)。

使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用

IAM 和 Amazon STS 与 Amazon CloudTrail 集成，后者是记录 IAM 用户或角色所执行操作的服务。CloudTrail 将对 IAM 和 Amazon STS 的所有 API 调用作为事件捕获，包括来自控制台的调用和 API 调用。如果您创建了跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶。如果您不配置跟踪记录，则仍可在 CloudTrail 控制台中的 Event history（事件历史记录）中查看最新事件。可使用 CloudTrail 获取有关对 IAM 或 Amazon STS 发出的请求的信息。例如，您可以查看发出请求的源 IP 地址、用户、时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅《[Amazon CloudTrail 用户指南](#)》。

主题

- [CloudTrail 中的 IAM 和 Amazon STS 信息 \(p. 340\)](#)
- [记录 IAM 和 Amazon STS API 请求 \(p. 341\)](#)
- [将 API 请求记录到其他 Amazon 服务 \(p. 341\)](#)
- [记录区域登录事件 \(p. 341\)](#)
- [记录用户登录事件 \(p. 343\)](#)
- [记录临时凭证的登录事件 \(p. 343\)](#)
- [CloudTrail 日志中的 IAM API 事件示例 \(p. 344\)](#)
- [CloudTrail 日志中的 Amazon STS API 事件示例 \(p. 345\)](#)
- [CloudTrail 日志中的登录事件示例 \(p. 351\)](#)

CloudTrail 中的 IAM 和 Amazon STS 信息

在您创建 Amazon 账户时，将在该账户上启用 CloudTrail。当 IAM 或 Amazon STS 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 Amazon 服务事件一同保存在 Event history（事件历史记录）中。您可以在 Amazon 账户中查看、搜索和下载最新事件。有关更多信息，请参阅 [使用 CloudTrail 事件历史记录查看事件](#)。

要持续记录 Amazon 账户中的事件（包括 IAM 和 Amazon STS 的事件），请创建跟踪。通过跟踪，CloudTrail 可将日志文件传送到 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪时，此跟踪应用于所有区域。此跟踪记录在 Amazon 分区中记录所有区域中的事件，并将日志文件传送到您指定的 Simple Storage Service（Amazon S3）存储桶。此外，您可以配置其他 Amazon 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅：

- [创建跟踪概览](#)
- [CloudTrail 支持的服务和集成](#)
- [为 CloudTrail 配置 Amazon SNS 通知](#)
- [从多个区域接收 CloudTrail 日志文件](#)和[从多个账户接收 CloudTrail 日志文件](#)

所有 IAM 和 Amazon STS 操作均由 CloudTrail 记录，并记录在了 [IAM API 参考](#)和 [Amazon Security Token Service API 参考](#)中。

记录 IAM 和 Amazon STS API 请求

CloudTrail 将所有经过身份验证的 API 请求（使用凭证发出）记录到 IAM 和 Amazon STS API 操作。CloudTrail 还会将未经身份验证的请求记录到 Amazon STS 操作、AssumeRoleWithSAML 和 AssumeRoleWithWebIdentity，并记录身份提供程序提供的信息。您可以使用此信息将由一个联合身份用户通过担任角色发出的调用映射回发出请求的外部联合发起人。如果使用 AssumeRole，您可以将调用映射回到发出请求的 Amazon 服务或发出请求的用户的账户。CloudTrail 日志条目中 JSON 数据的 userIdentity 部分包含您将 AssumeRole* 请求与特定联合身份用户进行映射时所需的信息。有关更多信息，请参阅 Amazon CloudTrail 用户指南中的 [CloudTrail userIdentity 元素](#)。

例如，IAM CreateUser、DeleteRole、ListGroups 调用和其他 API 操作全部都由 CloudTrail 记录。

本主题后面将会介绍有关此类日志条目的示例。

Important

如果在区域中激活 Amazon STS 终端节点而不是默认全球终端节点，您还必须在这些区域中启用 CloudTrail 日志记录。这是记录在这些区域中进行的任何 Amazon STS API 调用所必需的。有关更多信息，请参阅 Amazon CloudTrail 用户指南中的[在其他区域中启用 CloudTrail](#)。

将 API 请求记录到其他 Amazon 服务

对其他 Amazon 服务 API 操作的经过身份验证的请求由 CloudTrail 记录，这些日志条目包含有关生成请求的人员的信息。

例如，假设您发出一个请求来列出 Amazon EC2 实例或创建 Amazon CodeDeploy 部署组。有关发出该请求的人员或服务的详细信息将包含在该请求的日志条目中。此信息可帮助您确定发出该请求的是 Amazon Web Services 账户根用户、IAM 用户、角色还是其他 Amazon 服务。

有关 CloudTrail 日志条目中用户身份信息的详细信息，请参阅 Amazon CloudTrail 用户指南中的 [userIdentity 元素](#)。

记录区域登录事件

如果您允许 CloudTrail 将登录事件记录到您的日志中，您需要了解 CloudTrail 如何选择在何处记录事件。

- 如果您的用户直接登录到控制台，他们将重定向到全球或区域登录终端节点。终端节点依赖于所选服务控制台是否支持区域。例如，主控制台主页支持区域。如果您登录到 <https://alias.signin.aws.amazon.com/console>，则会重定向到区域登录终端节点，例如，<https://us-east-2.signin.aws.amazon.com>。这种重定向在用户的区域日志中创建一个区域 CloudTrail 日志条目：

另一方面，Amazon S3 控制台主页不支持区域，因此，如果您登录 <https://alias.signin.aws.amazon.com/console/s3>，Amazon 会将您重定向到全局登录终端节点 <https://signin.aws.amazon.com>。这种重定向创建一个全球 CloudTrail 日志条目。

- 您可以通过使用类似 <https://alias.signin.aws.amazon.com/console?region=ap-southeast-1> 这样的 URL 登录到启用区域的主控制台主页，手动请求特定区域登录终端节点。在这种情况下，Amazon 将您重定向到 ap-southeast-1 区域登录终端节点，并生成一个区域 CloudTrail 日志事件。

将登录事件视为区域还是全球事件取决于用户登录到的控制台，以及用户构建登录 URL 的方式。

- 服务控制台是否区域化？如果是，登录请求将自动重定向到区域登录终端节点，并在该区域的 CloudTrail 日志中记录事件。例如，如果您登录 <https://alias.signin.aws.amazon.com/console>，将会自动重定向到您区域的登录终端节点，如 <https://us-east-2.signin.aws.amazon.com>。将在该区域的日志中记录事件。

不过，某些服务尚未区域化。例如，Amazon S3 服务目前无法区域化。如果您登录到 <https://alias.signin.aws.amazon.com/console/s3>，系统会将您重定向到位于 <https://signin.aws.amazon.com> 的全球登录终端节点。这种重定向在全球日志中创建一个事件。

- 您还可以使用类似于 <https://alias.signin.aws.amazon.com/console?region=ap-southeast-1> 的 URL 来手动请求特定区域登录终端节点。此 URL 将您重定向到 ap-southeast-1 区域登录终端节点。这种重定向在区域日志中生成一个事件。

防止重复的区域日志条目

CloudTrail 在每个区域创建单独的跟踪。这些跟踪包括这些区域所发生的事件的信息，以及全球事件和非区域特定的事件。示例包括 IAM API 调用、对全球终端节点的 Amazon STS 调用以及 Amazon 登录事件。例如，假设您具有两个跟踪，分别位于不同的区域中。如果您随后创建新的 IAM 用户，则会在两个区域的日志文件中添加 CreateUser 事件，从而创建重复的日志条目。

Amazon Security Token Service (STS) 是一项全球服务，在 <https://sts.amazonaws.com> 上具有单个全球终端节点。对此终端节点的调用会记录为对全球服务的调用。但是，因为此端点在物理上位于美国东部（弗吉尼亚北部）区域，因此您的日志会将 us-east-1 列为事件区域。CloudTrail 不会将这些日志写入美国东部（俄亥俄）区域，除非您选择包括该区域的全球服务日志。Amazon STS 还允许调用区域端点，例如 sts.eu-central-1.amazonaws.com。CloudTrail 将对所有区域端点的调用写入到各自的区域。例如，对 sts.us-east-2.amazonaws.com 的调用发布到美国东部（俄亥俄）区域。对 sts.eu-central-1.amazonaws.com 的调用发布到欧洲（法兰克福）区域日志中。

有关多个区域和 Amazon STS 的更多信息，请参阅[在 Amazon 区域中管理 Amazon STS \(p. 331\)](#)。

下表列出了区域以及 CloudTrail 在每个区域中记录 Amazon STS 请求的方式。“位置”列指示 CloudTrail 写入的日志。“全球”表示，在您选择包含全球服务日志的任何区域中记录事件。“区域”表示，仅在终端节点所在的区域中记录事件。最后一列指示如何在日志条目标识请求的区域。

区域名称	CloudTrail 日志中的区域身份	端点	CloudTrail 日志的位置
无 - 全球	us-east-1	sts.amazonaws.com	服务全球
US East (Ohio)	us-east-2	sts.us-east-2.amazonaws.com	Region
US East (N. Virginia)	us-east-1	sts.us-east-1.amazonaws.com	Region
US West (N. California)	us-west-1	sts.us-west-1.amazonaws.com	Region
US West (Oregon)	us-west-2	sts.us-west-2.amazonaws.com	Region
Canada (Central)	ca-central-1	sts.ca-central-1.amazonaws.com	Region
欧洲（法兰克福）	eu-central-1	sts.eu-central-1.amazonaws.com	Region
Europe (Ireland)	eu-west-1	sts.eu-west-1.amazonaws.com	Region
欧洲（伦敦）	eu-west-2	sts.eu-west-2.amazonaws.com	Region
Asia Pacific (Tokyo)	ap-northeast-1	sts.ap-northeast-1.amazonaws.com	Region

区域名称	CloudTrail 日志中的区域身份	端点	CloudTrail 日志的位置
Asia Pacific (Seoul)	ap-northeast-2	sts.ap-northeast-2.amazonaws.com	Region
Asia Pacific (Mumbai)	ap-south-1	sts.ap-south-1.amazonaws.com	Region
亚太地区 (新加坡)	ap-southeast-1	sts.ap-southeast-1.amazonaws.com	Region
Asia Pacific (Sydney)	ap-southeast-2	sts.ap-southeast-2.amazonaws.com	Region
South America (São Paulo)	sa-east-1	sts.sa-east-1.amazonaws.com	Region

如果配置 CloudTrail 以将您的账户中的多个区域的跟踪信息聚合到单个 Amazon S3 存储桶中，将在日志中具有重复的 IAM 事件。换句话说，每个区域的跟踪将相同的 IAM 事件写入到聚合的日志中。为了防止出现这种重复，您可以选择性地包括全球事件。典型的方法是在一个跟踪中启用全球事件。然后，禁用写入同一 Amazon S3 存储桶中的所有其他跟踪的全球事件。这样便仅写入一组全局事件。

有关更多信息，请参阅 Amazon CloudTrail 用户指南中的[聚合日志](#)。

记录用户登录事件

CloudTrail 会将登录事件记录到 Amazon Web Services Management Console、Amazon 论坛 和 Amazon Web Services Marketplace 中。CloudTrail 将为 IAM 用户和联合身份用户记录成功和失败的登录尝试。

要查看成功和不成功的根用户登录的 CloudTrail 事件示例，请参阅 Amazon CloudTrail 用户指南中的[根用户的事件记录示例](#)。

作为最佳安全做法，当用户名不正确导致登录故障时，Amazon 不记录输入的 IAM 用户名文本。用户名文本将用 `HIDDEN_DUE_TO_SECURITY_REASONS` 值代替。有关示例，请参阅本主题后面的[用户名不正确导致的登录失败事件示例 \(p. 352\)](#)。用户名文本隐藏，因为此类错误可能是用户错误所导致。记录这些错误可能会泄露潜在的敏感信息。例如：

- 您不小心在用户名框中键入了密码。
- 您选择一个 Amazon 账户的登录页面链接，但键入了另一账户的账号。
- 您忘记了所登录的是哪个账户，不小心键入了个人电子邮件账户的账户名、银行登录标识符或某些其他私有 ID。

记录临时凭证的登录事件

当主体请求临时凭证时，主体类型将决定 CloudTrail 记录事件的方式。当主体代入其他账户中的角色时，这会很复杂。有多个 API 调用执行与角色跨账户操作相关的操作。首先，主体调用 Amazon STS API 来检索临时凭证。该操作记录到调用账户中以及执行 Amazon STS 操作的账户中。然后，主体使用角色在代入角色的账户中执行其他 API 调用。

您可以在角色信任策略中使用 `sts:SourceIdentity` 条件键，以要求用户在代入角色时指定身份。例如，您可以要求 IAM 用户指定自己的用户名作为其源身份。这可以帮助您确定哪个用户在 Amazon 中执行了具体的操作。有关更多信息，请参阅[sts:SourceIdentity \(p. 1024\)](#)。您可以使用 [sts:RoleSessionName \(p. 1023\)](#)，以要求用户在代入角色时指定会话名称。这可以在您查看 Amazon CloudTrail 日志时帮助您区分不同主体对角色使用的角色会话。

下表说明 CloudTrail 如何为每个生成临时凭证的 API 调用记录不同的信息。

主体类型	STS API	发起人账户的 CloudTrail 日志中的用户身份	所代入角色的账户的 CloudTrail 日志中的用户身份	角色后续 API 调用的 CloudTrail 日志中的用户身份
Amazon Web Services 账户 根用户凭证	GetSessionToken	根用户身份	角色所有者账户与调用账户相同	根用户身份
IAM 用户	GetSessionToken	IAM 用户身份。	角色所有者账户与调用账户相同	IAM 用户身份。
IAM 用户	GetFederationToken	IAM 用户身份。	角色所有者账户与调用账户相同	IAM 用户身份。
IAM 用户	AssumeRole	IAM 用户身份。	账号和主体 ID (如果是用户) 或 Amazon 服务主体	仅角色身份 (无用户)
外部验证的用户	AssumeRoleWithSAML	不适用	SAML 用户身份	仅角色身份 (无用户)
外部验证的用户	AssumeRoleWithWebIdentity	不适用	OIDC/Web 用户身份	仅角色身份 (无用户)

CloudTrail 日志中的 IAM API 事件示例

CloudTrail 日志文件包含的事件采用 JSON 格式。一个 API 事件表示一个 API 请求，并包括有关主体、所请求的操作、任意参数以及该操作日期和时间的信息。

CloudTrail 日志文件中的 IAM API 事件示例

以下示例说明了为进行 IAM GetUserPolicy 操作而发出的请求的 CloudTrail 日志条目。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/JaneDoe",
    "accountId": "444455556666",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "userName": "JaneDoe",
    "sessionContext": {
      "attributes": {
        "creationDate": "2014-07-15T21:39:40Z"
      }
    }
  },
  "invokedBy": "signin.amazonaws.com"
},
{
  "eventTime": "2014-07-15T21:40:14Z",
  "eventSource": "iam.amazonaws.com",
  "eventName": "GetUserPolicy",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "signin.amazonaws.com",
  "userAgent": "signin.amazonaws.com",
  "requestParameters": {
```

```
    "userName": "JaneDoe",
    "policyName": "ReadOnlyAccess-JaneDoe-201407151307"
  },
  "responseElements": null,
  "requestID": "9EXAMPLE-0c68-11e4-a24e-d5e16EXAMPLE",
  "eventID": "cEXAMPLE-127e-4632-980d-505a4EXAMPLE"
}
```

通过此事件信息，您可以确定发出该请求的目的是为了获取用户 `ReadOnlyAccess-JaneDoe-201407151307` 的名为 `JaneDoe` 的用户策略，如 `requestParameters` 元素中所述。您还可以看到，该请求是由名为 `JaneDoe` 的 IAM 用户在 2014 年 7 月 15 日下午 9:40 (UTC) 发出的。在本示例中，请求源自 Amazon Web Services Management Console，您可以从 `userAgent` 元素中判断出来。

CloudTrail 日志中的 Amazon STS API 事件示例

CloudTrail 日志文件包含的事件采用 JSON 格式。一个 API 事件表示一个 API 请求，并包括有关主体、所请求的操作、任意参数以及该操作日期和时间的信息。

CloudTrail 日志文件中跨账户 Amazon STS API 事件的示例

账户 777788889999 中名为 `JohnDoe` 的 IAM 用户调用 Amazon STS `AssumeRole` 操作来在账户 111122223333 中担任角色 `EC2-dev`。账户管理员要求用户在担任角色时设置与其用户名相同的源身份。用户传入 `JohnDoe` 的源身份值。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAQRSTUVWXYZEXAMPLE",
    "arn": "arn:aws:iam::777788889999:user/JohnDoe",
    "accountId": "777788889999",
    "accessKeyId": "AKIAQRSTUVWXYZEXAMPLE",
    "userName": "JohnDoe"
  },
  "eventTime": "2014-07-18T15:07:39Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRole",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/1.11.10 Python/2.7.8 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64
botocore/1.4.67",
  "requestParameters": {
    "roleArn": "arn:aws:iam::111122223333:role/EC2-dev",
    "roleSessionName": "JohnDoe-EC2-dev",
    "sourceIdentity": "JohnDoe",
  },
  "responseElements": {
    "credentials": {
      "sessionToken": "<encoded session token blob>",
      "accessKeyId": "AKIAQRSTUVWXYZEXAMPLE",
      "expiration": "Jul 18, 2014 4:07:39 PM"
    },
    "assumedRoleUser": {
      "assumedRoleId": "AIDAQRSTUVWXYZEXAMPLE:JohnDoe-EC2-dev",
      "arn": "arn:aws:sts::111122223333:assumed-role/EC2-dev/JohnDoe-EC2-dev"
    }
  },
  "sourceIdentity": "JohnDoe"
},
{
  "resources": [
    {
      "ARN": "arn:aws:iam::111122223333:role/EC2-dev",
```

```

    "accountId": "111122223333",
    "type": "AWS::IAM::Role"
  }
],
"requestID": "4EXAMPLE-0e8d-11e4-96e4-e55c0EXAMPLE",
"sharedEventID": "bEXAMPLE-efea-4a70-b951-19a88EXAMPLE",
"eventID": "dEXAMPLE-ac7f-466c-a608-4ac8dEXAMPLE",
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
}

```

第二个示例显示同一请求的代入角色账户 (111122223333) 的 CloudTrail 日志条目。

```

{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AWSAccount",
    "principalId": "AIDAQRSTUVWXYZEXAMPLE",
    "accountId": "777788889999"
  },
  "eventTime": "2014-07-18T15:07:39Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRole",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/1.11.10 Python/2.7.8 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64
botocore/1.4.67",
  "requestParameters": {
    "roleArn": "arn:aws:iam::111122223333:role/EC2-dev",
    "roleSessionName": "JohnDoe-EC2-dev",
    "sourceIdentity": "JohnDoe",
  },
  "responseElements": {
    "credentials": {
      "sessionToken": "<encoded session token blob>",
      "accessKeyId": "AKIAQRSTUVWXYZEXAMPLE",
      "expiration": "Jul 18, 2014 4:07:39 PM"
    },
    "assumedRoleUser": {
      "assumedRoleId": "AIDAQRSTUVWXYZEXAMPLE:JohnDoe-EC2-dev",
      "arn": "arn:aws:sts::111122223333:assumed-role/EC2-dev/JohnDoe-EC2-dev"
    },
    "sourceIdentity": "JohnDoe"
  },
  "requestID": "4EXAMPLE-0e8d-11e4-96e4-e55c0EXAMPLE",
  "sharedEventID": "bEXAMPLE-efea-4a70-b951-19a88EXAMPLE",
  "eventID": "dEXAMPLE-ac7f-466c-a608-4ac8dEXAMPLE"
}

```

CloudTrail 日志文件中 Amazon STS 角色链接 API 事件示例

以下示例演示账户 111111111111 中 John Doe 发出的请求的 CloudTrail 日志条目。John 之前使用自己的 JohnDoe 用户代入 JohnRole1 角色。对于此请求，他使用来自该角色的凭证代入 JohnRole2 角色。这称为 [角色链](#) (p. 159)。当他担任 JohnDoe1 角色时设置的源身份将在担任 JohnRole2 的请求中继续存在。如果 John 尝试在担任角色时设置不同的源身份，请求将被拒绝。John 将两个 [会话标签](#) (p. 293) 传递到请求中。他将这两个标签设置为可传递。请求继承 Department 标签作为可传递标签，因为 John 在代入 JohnRole1 时将其设置为可传递。更多有关源身份的信息，请参阅 [监控和控制使用所担任角色执行的操作](#) (p. 316)。有关角色链中可传递键的更多信息，请参阅 [使用会话标签链接角色](#) (p. 299)。

```

{
  "eventVersion": "1.05",
  "userIdentity": {

```

```

    "type": "AssumedRole",
    "principalId": "AROAIN5ATK5U7KEXAMPLE:JohnRole1",
    "arn": "arn:aws:sts::111111111111:assumed-role/JohnDoe/JohnRole1",
    "accountId": "111111111111",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2019-10-02T21:50:54Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAIN5ATK5U7KEXAMPLE",
        "arn": "arn:aws:iam::111111111111:role/JohnRole1",
        "accountId": "111111111111",
        "userName": "JohnDoe"
      },
      "sourceIdentity": "JohnDoe"
    },
    "eventTime": "2019-10-02T22:12:29Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRole",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "123.145.67.89",
    "userAgent": "aws-cli/1.16.248 Python/3.4.7
Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 botocore/1.12.239",
    "requestParameters": {
      "incomingTransitiveTags": {
        "Department": "Engineering"
      },
      "tags": [
        {
          "value": "johndoe@example.com",
          "key": "Email"
        },
        {
          "value": "12345",
          "key": "CostCenter"
        }
      ],
      "roleArn": "arn:aws:iam::111111111111:role/JohnRole2",
      "roleSessionName": "Role2WithTags",
      "sourceIdentity": "JohnDoe",
      "transitiveTagKeys": [
        "Email",
        "CostCenter"
      ],
      "durationSeconds": 3600
    },
    "responseElements": {
      "credentials": {
        "accessKeyId": "ASIAWHOJDLGPOEXAMPLE",
        "expiration": "Oct 2, 2019 11:12:29 PM",
        "sessionToken": "AgoJb3JpZ2luX2VjEB4aCXVzLXdlc3QtMSJHMEEXAMPLETOKEN
+rJb8Lo30mFc5MlhFCEbubZvEj0wHB/mDMwIgSEe9gk/Zjr09tZV7F1HDTMhmEXAMPLETOKEN/iEJ/
rkqngII9/////////
ARABGgwOMjgzMDc4NjM5NjYiDLZjZFKwP4qxQG5sFCryASO4UPz5qE97wPPH1eLMvs7CgSDBSwfonmRTCfokm2FN1+hWUdQQH6adjbb
+C+WKFZb701eiv9J5La2EXAMPLETOKEN/c7S5Iro1WUJ0q3Cxuo/8HUoSxVhQHM7zF7mWWLhXLEQ52ivL
+F6q5dpXu4aTFedpMfnJa8JtkWwG9x1Axj0Ypy2ok8v5unpQGwYchlvwdvj6ez1Dm8Xg1+qIzXILiEXAMPLETOKEN/
vQGqu8H+nxp3kabcrtOvTFTvxX6vsc8OGwUfHhzAfYGEXAMPLETOKEN/
L6vlyMM3B10wFOrQBno1HEjf1oNI8RnQiMNFdUOtWYj7HUZIOCMzmjFN8PPHq77N7GJl9lZvIZKQA00wcjg
+mc78zHCj8y0siY8C96paEXAMPLETOKEN/
E3cpksxWdgs91HRzJWScjN2+r2LTGjYhyPqcmFzZo2mCE7mBNEXAMPLETOKEN/oJy
+2o83YNW5tOiDmczgDzJZ4UKR84yGYOMfSnF4XcEjRdGAJ30JFwmTcTQICAlSwLEXAMPLETOKEN"
      },

```

```
    "assumedRoleUser": {
      "assumedRoleId": "AROAIFR7WHDTSOYQYHFUE:Role2WithTags",
      "arn": "arn:aws:sts::111111111111:assumed-role/test-role/Role2WithTags"
    },
    "sourceIdentity": "JohnDoe"
  },
  "requestID": "b96b0e4e-e561-11e9-8b3f-7b396EXAMPLE",
  "eventID": "1917948f-3042-46ec-98e2-62865EXAMPLE",
  "resources": [
    {
      "ARN": "arn:aws:iam::111111111111:role/JohnRole2",
      "accountId": "111111111111",
      "type": "AWS::IAM::Role"
    }
  ],
  "eventType": "AwsApiCall",
  "recipientAccountId": "111111111111"
}
```

CloudTrail 日志文件中 Amazon 服务 Amazon STS API 事件的示例

以下示例显示由 Amazon 服务使用服务角色中的权限调用其他服务 API 发出的请求的 CloudTrail 日志条目。它显示了在 777788889999 账户中发出的请求的 CloudTrail 日志条目。

```
{
  "eventVersion": "1.04",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDAQRSTUVWXYZEXAMPLE:devdsk",
    "arn": "arn:aws:sts::777788889999:assumed-role/AssumeNothing/devdsk",
    "accountId": "777788889999",
    "accessKeyId": "AKIAQRSTUVWXYZEXAMPLE",
    "sessionContext": {
      "attributes": {
        "creationDate": "2016-11-14T17:25:26Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDAQRSTUVWXYZEXAMPLE",
        "arn": "arn:aws:iam::777788889999:role/AssumeNothing",
        "accountId": "777788889999",
        "userName": "AssumeNothing"
      }
    }
  },
  "eventTime": "2016-11-14T17:25:45Z",
  "eventSource": "s3.amazonaws.com",
  "eventName": "DeleteBucket",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.1",
  "userAgent": "[aws-cli/1.11.10 Python/2.7.8 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 botocore/1.4.67]",
  "requestParameters": {
    "bucketName": "my-test-bucket-cross-account"
  },
  "responseElements": null,
  "requestID": "EXAMPLE463D56D4C",
  "eventID": "dEXAMPLE-265a-41e0-9352-4401bEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "777788889999"
}
```


CloudTrail 日志文件中 SAML Amazon STS API 事件的示例

以下示例说明了为进行 Amazon STS AssumeRoleWithSAML 操作而发出的请求的 CloudTrail 日志条目。请求包括 SAML 属性 CostCenter 和 Project，这些属性作为会话标签 (p. 293) 通过 SAML 断言传递。这些标签被设置为可传递标签，因此在角色链场景中持续存在 (p. 299)。请求包含 SAML 属性 sourceIdentity，该属性在 SAML 断言中传递。如果某人使用生成的角色会话凭证担任另一个角色，则此源身份将仍然存在。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "SAMLUser",
    "principalId": "SampleUkhl14+ExampLexL/jEvs=:SamlExample",
    "userName": "SamlExample",
    "identityProvider": "bdGOnTesti4+ExampLexL/jEvs="
  },
  "eventTime": "2019-11-01T19:14:36Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRoleWithSAML",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/1.16.263 Python/3.4.7
Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 botocore/1.12.253",
  "requestParameters": {
    "SAMLAssertionID": "_c0046cEXAMPLEb9d4b8eEXAMPLE2619aEXAMPLE",
    "roleSessionName": "MyAssignedRoleSessionName",
    "sourceIdentity": "MySAMLUser",
    "principalTags": {
      "CostCenter": "987654",
      "Project": "Unicorn",
      "Department": "Engineering"
    },
    "transitiveTagKeys": [
      "CostCenter",
      "Project"
    ],
    "durationSeconds": 3600,
    "roleArn": "arn:aws:iam::444455556666:role/SAMLTestRoleShibboleth",
    "principalArn": "arn:aws:iam::444455556666:saml-provider/Shibboleth"
  },
  "responseElements": {
    "subjectType": "transient",
    "issuer": "https://server.example.com/idp/shibboleth",
    "sourceIdentity": "MySAMLUser",
    "credentials": {
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
      "expiration": "Mar 23, 2016 2:39:57 AM",
      "sessionToken": "<encoded session token blob>"
    },
    "nameQualifier": "bdGOnTesti4+ExampLexL/jEvs=",
    "assumedRoleUser": {
      "assumedRoleId": "AROAD35QRSTUVWEXAMPLE:MyAssignedRoleSessionName",
      "arn": "arn:aws:sts::444455556666:assumed-role/SAMLTestRoleShibboleth/MyAssignedRoleSessionName"
    },
    "subject": "SamlExample",
    "audience": "https://signin.aws.amazon.com/saml"
  },
  "resources": [
    {
      "ARN": "arn:aws:iam::444455556666:role/SAMLTestRoleShibboleth",
      "accountId": "444455556666",
      "type": "AWS::IAM::Role"
    }
  ],
}
```

```
{
  "ARN": "arn:aws:iam::444455556666:saml-provider/test-saml-provider",
  "accountId": "444455556666",
  "type": "AWS::IAM::SAMLProvider"
},
{
  "requestID": "6EXAMPLE-e595-11e5-b2c7-c974fEXAMPLE",
  "eventID": "dEXAMPLE-265a-41e0-9352-4401bEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "444455556666"
}
```

CloudTrail 日志文件中的 Web 身份 Amazon STS API 事件示例

以下示例说明了为进行 Amazon STS AssumeRoleWithWebIdentity 操作而发出的请求的 CloudTrail 日志条目。请求包括属性 CostCenter 和 Project，这些属性作为[会话标签](#) (p. 293) 通过身份提供程序令牌传递。这些标签被设置为可传递标签，因此在[角色链中持续存在](#) (p. 299)。请求包含身份提供程序令牌中的 sourceIdentity 属性。如果某人使用生成的角色会话凭证担任另一个角色，则此源身份将仍然存在。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "WebIdentityUser",
    "principalId": "accounts.google.com:<id-of-application>.apps.googleusercontent.com:<id-of-user>",
    "userName": "<id of user>",
    "identityProvider": "accounts.google.com"
  },
  "eventTime": "2016-03-23T01:39:51Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRoleWithWebIdentity",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/1.3.23 Python/2.7.6 Linux/2.6.18-164.el5",
  "requestParameters": {
    "sourceIdentity": "MyWebIdentityUser",
    "durationSeconds": 3600,
    "roleArn": "arn:aws:iam::444455556666:role/FederatedWebIdentityRole",
    "roleSessionName": "MyAssignedRoleSessionName"
  },
  "principalTags": {
    "CostCenter": "24680",
    "Project": "Pegasus"
  },
  "transitiveTagKeys": [
    "CostCenter",
    "Project"
  ],
  "responseElements": {
    "provider": "accounts.google.com",
    "subjectFromWebIdentityToken": "<id of user>",
    "sourceIdentity": "MyWebIdentityUser",
    "audience": "<id of application>.apps.googleusercontent.com",
    "credentials": {
      "accessKeyId": "ASIAQQRSTUVWRAOEXAMPLE",
      "expiration": "Mar 23, 2016 2:39:51 AM",
      "sessionToken": "<encoded session token blob>"
    },
    "assumedRoleUser": {
      "assumedRoleId": "AROACQQRSTUVWRAOEXAMPLE:MyAssignedRoleSessionName",
      "arn": "arn:aws:sts::444455556666:assumed-role/FederatedWebIdentityRole/MyAssignedRoleSessionName"
    }
  }
}
```

```
{,
  "resources": [
    {
      "ARN": "arn:aws:iam::444455556666:role/FederatedWebIdentityRole",
      "accountId": "444455556666",
      "type": "AWS::IAM::Role"
    }
  ],
  "requestID": "6EXAMPLE-e595-11e5-b2c7-c974fEXAMPLE",
  "eventID": "bEXAMPLE-0b30-4246-b28c-e3da3EXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "444455556666"
}
```

CloudTrail 日志中的登录事件示例

CloudTrail 日志文件包含的事件采用 JSON 格式。登录事件表示单个登录请求，并包括有关登录主体、区域以及操作的日期和事件的信息。

CloudTrail 日志文件中的登录成功事件示例

以下示例显示了一个成功的登录事件的 CloudTrail 日志条目。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/JohnDoe",
    "accountId": "111122223333",
    "userName": "JohnDoe"
  },
  "eventTime": "2014-07-16T15:49:27Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.110",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0",
  "requestParameters": null,
  "responseElements": {
    "ConsoleLogin": "Success"
  },
  "additionalEventData": {
    "MobileVersion": "No",
    "LoginTo": "https://console.aws.amazon.com/s3/"
  },
  "eventID": "3fcfb182-98f8-4744-bd45-10a395ab61cb"
}
```

有关 CloudTrail 日志文件中所包含信息的更多详细信息，请参阅 Amazon CloudTrail 用户指南中的 [CloudTrail 事件参考](#)。

CloudTrail 日志文件中的登录失败事件示例

以下示例显示了一个失败的登录事件的 CloudTrail 日志条目。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
```

```
{
  "principalId": "AIDACKCEVSQ6C2EXAMPLE",
  "arn": "arn:aws:iam::111122223333:user/JaneDoe",
  "accountId": "111122223333",
  "userName": "JaneDoe"
},
"eventTime": "2014-07-08T17:35:27Z",
"eventSource": "signin.amazonaws.com",
"eventName": "ConsoleLogin",
"awsRegion": "us-east-2",
"sourceIPAddress": "192.0.2.100",
"userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0",
"errorMessage": "Failed authentication",
"requestParameters": null,
"responseElements": {
  "ConsoleLogin": "Failure"
},
"additionalEventData": {
  "MobileVersion": "No",
  "LoginTo": "https://console.aws.amazon.com/sns"
},
"eventID": "11ea990b-4678-4bcd-8fbe-62509088b7cf"
}
```

从此信息，您可以确定此次登录尝试源自名为 JaneDoe 的 IAM 用户，如 `userIdentity` 元素中所示。您还可以发现此次登录尝试失败，如 `responseElements` 元素中所示。可以看出 JaneDoe 是在 2014 年 7 月 8 日下午 5:35 (UTC) 尝试登录 Amazon SNS 控制台的。

用户名称不正确导致的登录失败事件示例

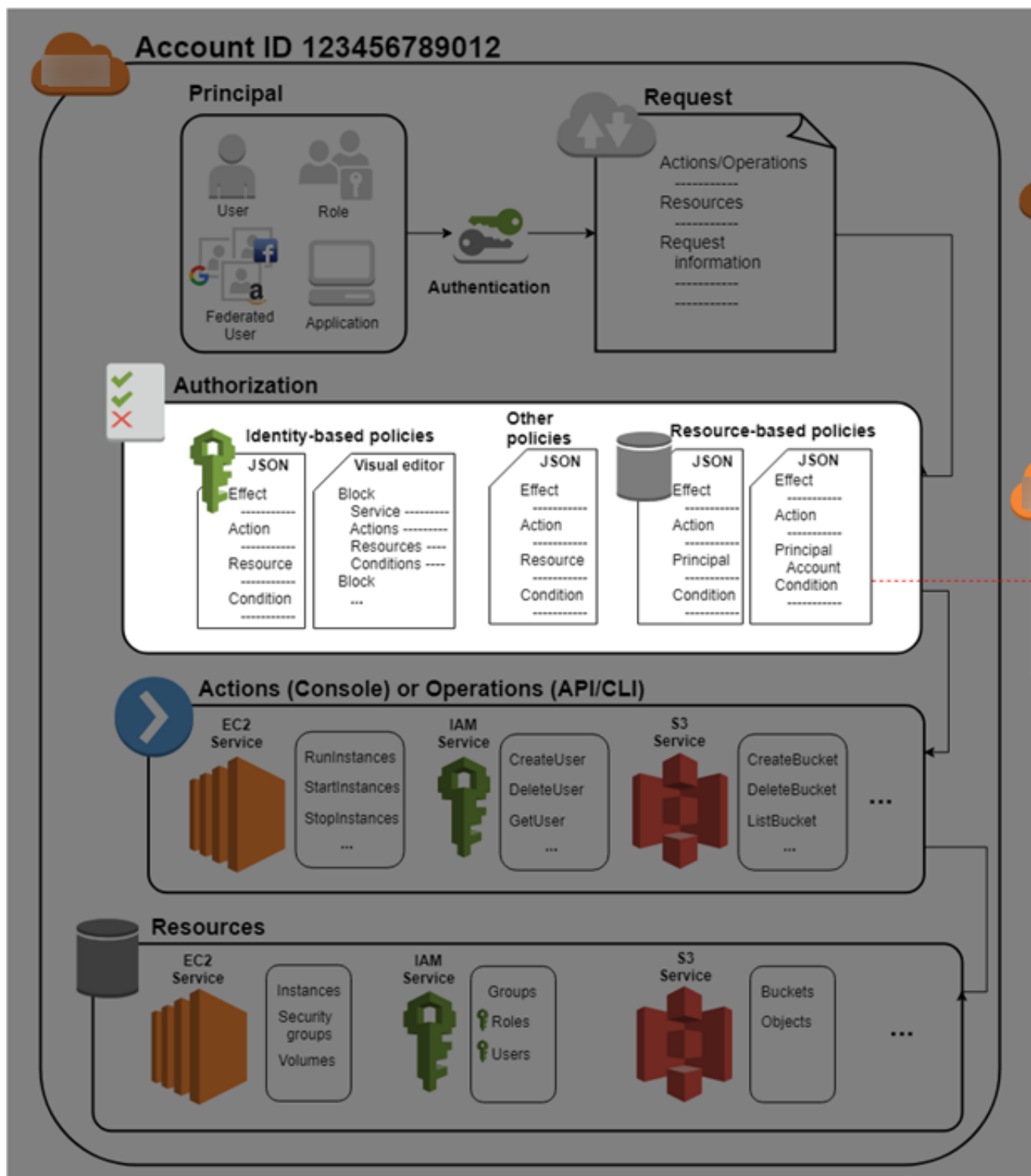
以下示例显示了一个失败登录事件的 CloudTrail 日志条目，该事件是由于用户输入错误用户名导致的。Amazon 用 `HIDDEN_DUE_TO_SECURITY_REASONS` 代替了 `userName` 文本以帮助防止泄露可能的敏感信息。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "accountId": "123456789012",
    "accessKeyId": "",
    "userName": "HIDDEN_DUE_TO_SECURITY_REASONS"
  },
  "eventTime": "2015-03-31T22:20:42Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0",
  "errorMessage": "No username found in supplied account",
  "requestParameters": null,
  "responseElements": {
    "ConsoleLogin": "Failure"
  },
  "additionalEventData": {
    "LoginTo": "https://console.aws.amazon.com/console/home?state=hashArgs%23&isauthcode=true",
    "MobileVersion": "No"
  },
  "eventID": "a7654656-0417-45c6-9386-ea8231385051",
  "eventType": "AwsConsoleSignin",
  "recipientAccountId": "123456789012"
}
```

适用于 Amazon 资源的 Access Management

Amazon Identity and Access Management (IAM) 是一种 Web 服务，可以帮助您安全地控制对 Amazon 资源的访问。当[主体 \(p. 5\)](#)在 Amazon 中发出请求时，Amazon 执行代码会检查是否对主体进行身份验证（登录）和授权（具有权限）。您将创建策略并将其附加到 IAM 身份或 Amazon 资源，以便管理 Amazon 中的访问。策略是 Amazon 中的 JSON 文档，在附加到身份或资源时，策略定义它们的权限。有关策略类型和用法的更多信息，请参阅[IAM 中的策略和权限。\(p. 355\)](#)。

有关身份验证和授权过程的其余部分的详细信息，请参阅[了解 IAM 的工作方式 \(p. 3\)](#)。



在授权期间，Amazon 执行代码使用[请求上下文 \(p. 5\)](#)中的值检查匹配的策略并确定是允许还是拒绝请求。

Amazon 检查应用于请求上下文的每个策略。如果一个策略拒绝请求，Amazon 将拒绝整个请求并停止评估策略。这称为显式拒绝。由于请求是默认拒绝的，因此，只有在适用的策略允许请求的每个部分时，IAM 才会授权请求。单个账户中对于请求的[评估逻辑 \(p. 972\)](#)遵循以下规则：

- 默认情况下，所有请求都被隐式拒绝。（或者，默认情况下，Amazon Web Services 账户根用户拥有完全访问权限。）
- 基于身份或基于资源的策略中的显式允许将覆盖此默认值。
- 如果存在权限边界、Organizations SCP 或会话策略，它可能会使用隐式拒绝覆盖允许。
- 任何策略中的显式拒绝将覆盖任何允许。

在对您的请求进行身份验证和授权后，Amazon 将批准该请求。如果您需要在另一个账户中发出请求，其他账户中的策略必须允许访问资源。此外，您用于发出请求的 IAM 实体必须具有基于身份的策略，该策略允许该请求。

访问管理资源

有关权限和创建策略的更多信息，请参阅以下资源：

Amazon 安全博客中的以下文章介绍 Amazon S3 存储桶和对象访问策略的常用编写方法。

- [编写 IAM policy：如何授予对 Amazon S3 存储桶的访问权限](#)
- [编写 IAM policy：授予对 Amazon S3 存储桶中用户特定文件夹的访问权限](#)
- [IAM policy、存储桶策略和 ACL！天哪！（Controlling Access to S3 Resources）](#)（控制对 S3 资源的访问）。
- [RDS 资源级别权限读本](#)
- [阐明 EC2 资源级权限](#)

IAM 中的策略和权限。

您在 Amazon 中通过创建策略并将其附加到 IAM 身份（用户、用户组或角色）或 Amazon 资源来管理访问权限。策略是 Amazon 中的对象；在与身份或资源相关时，策略定义它们的权限。在某个 IAM 主体（用户或角色）发出请求时，Amazon 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略作为 JSON 文档存储在 Amazon 中。Amazon 支持六种类型的策略：基于身份的策略、基于资源的策略、权限边界、Organizations SCP、ACL 和会话策略。

IAM policy 定义操作的权限，无关乎您使用哪种方法执行操作。例如，如果一个策略允许 [GetUser](#) 操作，则具有该策略的用户可以从 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 获取用户信息。在创建 IAM 用户时，您可以选择允许控制台或编程访问。如果允许控制台访问，则 IAM 用户可以使用用户名和密码登录到控制台。或者，如果允许编程访问，则用户可以通过访问密钥来使用 CLI 或 API。

策略类型

以下策略类型按从最常用到不常用的顺序列出，可在 Amazon 中使用。有关更多详细信息，请参阅下面有关各种策略类型的各部分。

- [基于身份的策略 \(p. 356\)](#) – 将[托管 \(p. 356\)](#)策略和[内联 \(p. 356\)](#)策略附加到 IAM 身份（用户、用户所属组或角色）。基于身份的策略向身份授予权限。
- [基于资源的策略 \(p. 356\)](#) – 将内联策略附加到资源。基于资源的策略的最常见示例是 Amazon S3 存储桶策略和 IAM 角色信任策略。基于资源的策略向在策略中指定的主体授予权限。主体可以与资源位于同一个账户中，也可以位于其他账户中。
- [权限边界 \(p. 356\)](#) – 使用托管策略作为 IAM 实体（用户或角色）的权限边界。该策略定义基于身份的策略可以授予实体的最大权限，但不授予权限。权限边界不定义基于资源的策略可以授予实体的最大权限。

- [企业 SCP \(p. 357\)](#) - 使用 Amazon Organizations 服务控制策略 (SCP) 为企业或企业单元 (OU) 的账户成员定义最大权限。SCP 限制基于身份的策略或基于资源的策略授予账户中实体 (用户或角色) 的权限, 但不授予权限。
- [访问控制列表 \(ACL\) \(p. 357\)](#) - 使用 ACL 来控制其他账户中的哪些主体可以访问 ACL 附加到的资源。ACL 类似于基于资源的策略, 但它们是唯一不使用 JSON 策略文档结构的策略类型。ACL 是跨账户的权限策略, 向指定的主体授予权限。ACL 不能向同一账户内的实体授予权限。
- [会话策略 \(p. 357\)](#) - 在您使用 Amazon CLI 或 Amazon API 担任某个角色或联合身份用户时, 传递高级会话策略。会话策略限制角色或用户的基于身份的策略授予会话的权限。会话策略限制所创建会话的权限, 但不授予权限。有关更多信息, 请参阅[会话策略](#)。

基于身份的策略

基于身份的策略是 JSON 权限的策略文档, 用于控制身份 (用户、用户组和角色) 可在什么样的条件下对哪些资源执行哪些操作。基于身份的策略可以进一步分类:

- 托管策略 - 基于身份的独立策略, 可附加到您的 Amazon 账户中的多个用户、组和角色。有两种托管策略:
 - Amazon 托管策略 - 由 Amazon 创建和管理的托管策略。
 - 客户托管策略 - 您在 Amazon 账户中创建和管理的托管策略。与 Amazon 托管策略相比, 客户托管策略可以更精确地控制策略。
- 内联策略 — 直接添加到单个用户、组或角色的策略。内联策略维持策略与身份之间严格的一对一关系。当您删除身份时, 它们将会被删除。

要了解如何在托管策略或内联策略之间选择, 请参阅 [在托管策略与内联策略之间进行选择 \(p. 368\)](#)。

基于资源的策略

基于资源的策略是附加到资源 (如 Amazon S3 存储桶) 的 JSON 策略文档。这些策略授予指定的主体对该资源执行特定操作的权限, 并定义这在哪些条件下适用。基于资源的策略是内联策略。没有基于托管资源的策略。

要启用跨账户存取, 您可以将整个账户或其它账户中的 IAM 实体指定为基于资源的策略中的主体。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当主体和资源位于单独的 Amazon 账户中时, 还必须使用基于身份的策略授予对资源的主体访问权限。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。有关授予跨服务访问权限的分步说明, 请参阅 [IAM 教程: 使用 IAM 角色委托跨 Amazon 账户的访问权限 \(p. 31\)](#)。

IAM 服务仅支持一种类型的基于资源的策略 (称为角色信任策略), 这种策略附加到 IAM 角色。IAM 角色既是身份, 又是支持基于资源的策略的资源。因此, 您必须将信任策略和基于身份的策略同时附加到 IAM 角色。信任策略定义哪些主体实体 (账户、用户、角色和联合身份用户) 可以代入该角色。要了解 IAM 角色如何与其他基于资源的策略不同, 请参阅 [IAM 角色与基于资源的策略有何不同 \(p. 274\)](#)。

要了解哪些其他服务支持基于资源的策略, 请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)。要了解基于资源的策略的更多信息, 请参阅[基于身份的策略和基于资源的策略 \(p. 379\)](#)。要了解您信任区域之外的账户 (受信的企业或账户) 中的主体是否有权承担您的角色, 请参阅[什么是 IAM Access Analyzer?](#)。

IAM 权限边界

权限边界是一项高级功能, 借助该功能, 您可以设置基于身份的策略可以授予 IAM 实体的最大权限。当您设置实体的权限边界时, 该实体只能执行其基于身份的策略和其权限边界同时允许的操作。指定用户或角色作为主体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 实体的权限边界 \(p. 370\)](#)。

服务控制策略 (SCP)

Amazon Organizations 是用于分组和集中管理企业拥有的 Amazon 账户的服务。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 是指定组织或组织单元 (OU) 的最大权限的 JSON 策略。SCP 限制成员账户中实体的权限，包括每个 Amazon Web Services 账户 根用户。任一项策略中的显式拒绝将覆盖允许。

有关 Organization 和 SCP 的更多信息，请参阅 Amazon Organizations 用户指南中的 [SCP 的工作原理](#)。

访问控制列表 (ACL)

访问控制列表 (ACL) 是一种服务策略，允许您控制另一个账户中的哪些主体可以访问资源。不能使用 ACL 控制同一账户中的主体的访问权限。ACL 类似于基于资源的策略，但它们是唯一不使用 JSON 策略文档格式的策略类型。Simple Storage Service (Amazon S3)、Amazon WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅 Amazon Simple Storage Service 开发人员指南中的 [访问控制列表 \(ACL\) 概述](#)。

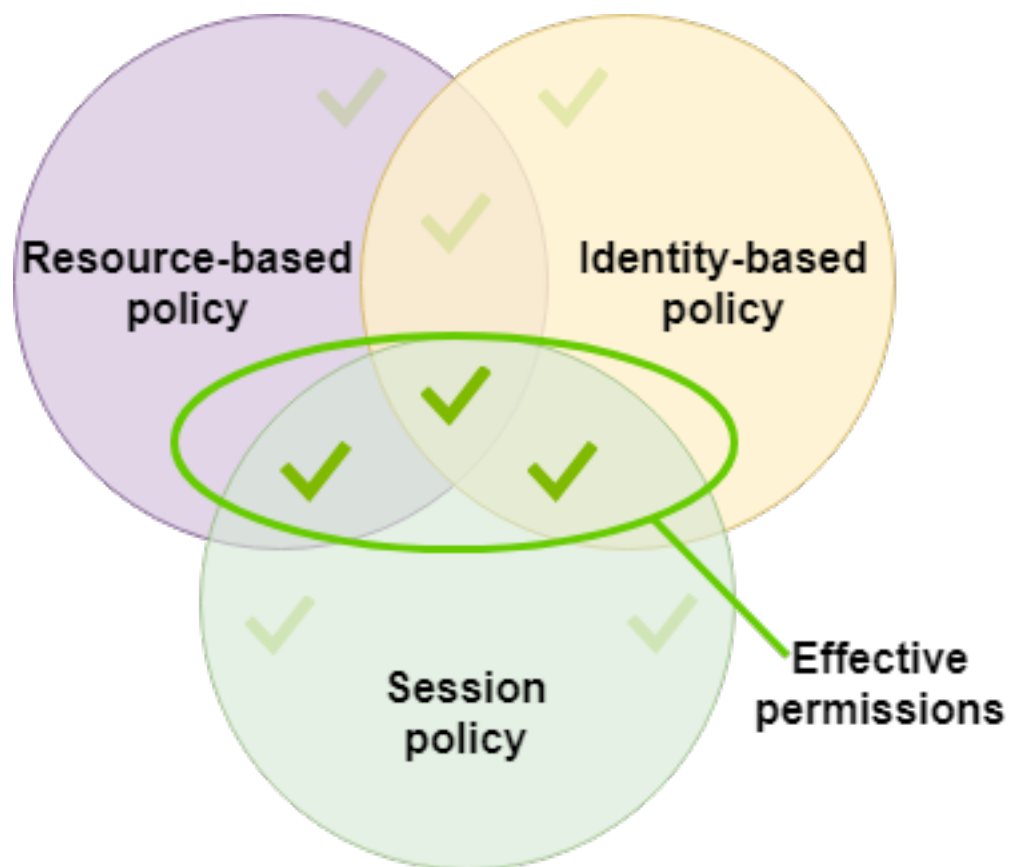
会话策略

会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。会话的权限是用于创建会话的 IAM 实体 (用户或角色) 的基于身份的策略与会话策略的交集。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。

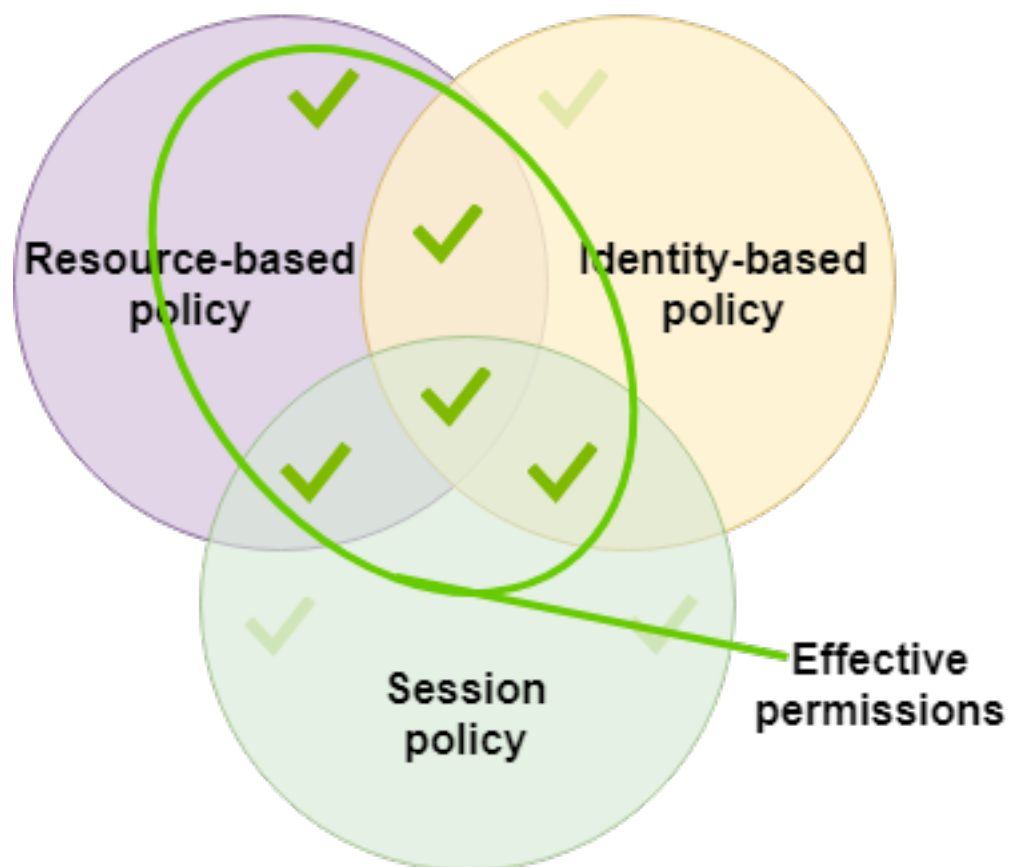
您可以使用 AssumeRole、AssumeRoleWithSAML 或 AssumeRoleWithWebIdentity API 操作以编程方式创建角色会话和传递会话策略。您可以使用 Policy 参数传递单个 JSON 内联会话策略文档。您可以使用 PolicyArns 参数指定最多 10 个托管会话策略。有关创建角色会话的更多信息，请参阅 [请求临时安全凭证 \(p. 302\)](#)。

当您创建联合身份用户会话时，您使用 IAM 用户的访问密钥以编程方式调用 GetFederationToken API 操作。您还必须传递会话策略。生成的会话的权限是 IAM 用户的基于身份的策略与会话策略的交集。有关创建联合身份用户的更多信息，请参阅 [GetFederationToken – 通过自定义身份凭证代理程序进行联合身份验证 \(p. 307\)](#)。

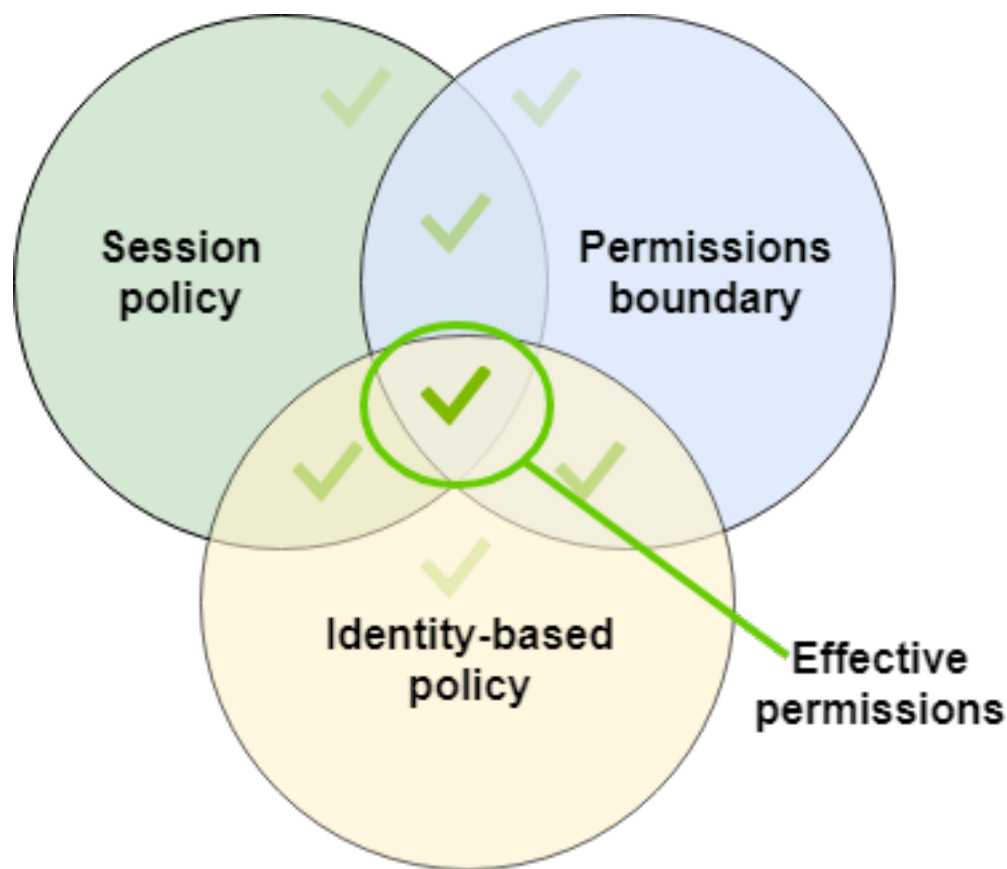
基于资源的策略可以将用户或角色的 ARN 指定为主体。在这种情况下，在创建会话之前，将在角色或用户的基于身份的策略中添加基于资源的策略中的权限。会话策略限制由基于资源的策略和基于身份的策略授予的总权限。生成的会话权限是会话策略与基于资源的策略的交集以及会话策略与基于身份的策略的交集。



基于资源的策略可以将会话的 ARN 指定为主体。在这种情况下，在创建会话后，将添加基于资源的策略中的权限。基于资源的策略权限不受会话策略限制。生成的会话具有基于资源的策略中的所有权限以及 基于身份的策略与会话策略的交集。



权限边界可以设置用于创建会话的用户或角色的最大权限。在这种情况下，生成的会话的权限是会话策略、权限边界和基于身份的策略的交集。不过，权限边界不会限制指定生成的会话 ARN 基于资源的策略授予权限。



策略和根用户

Amazon Web Services 账户 根用户会受到一些策略类型的影响，但不会受其他策略类型的影响。您不能将基于身份的策略附加到根用户，也不能为根用户设置权限边界。不过，您可以在基于资源的策略或 ACL 中将根用户指定为主体。根用户仍然是账户的成员。如果该账户是 Amazon Organizations 中的企业成员，则根用户受账户的任何 SCP 的影响。

JSON 策略概述

大多数策略在 Amazon 中存储为 JSON 文档。基于身份的策略和用于设置权限边界的策略是您附加到用户或角色的 JSON 策略文档。基于资源的策略是附加到资源的 JSON 策略文档。SCP 是附加到 Amazon Organizations 组织单元 (OU) 的使用限制语法的 JSON 策略文档。ACL 也可附加到资源，但必须使用不同的语法。会话策略是您在创建角色或联合身份用户会话时提供的 JSON 策略。

您无需了解 JSON 语法。您可以使用 Amazon Web Services Management Console 中的可视化编辑器创建和编辑客户托管策略，而无需使用 JSON。不过，如果在组中使用内联策略或复杂的策略，您还必须使用控制台在 JSON 编辑器中创建和编辑这些策略。有关使用可视化编辑器的更多信息，请参阅 [创建 IAM policy \(p. 437\)](#) 和 [编辑 IAM 策略 \(p. 461\)](#)。

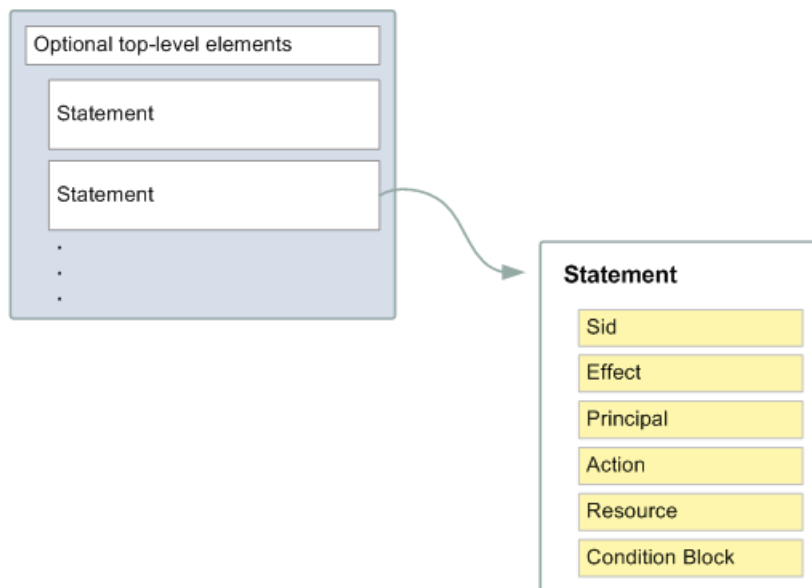
当您创建或编辑 JSON 策略时，IAM 可以执行策略验证以帮助您创建有效的策略。IAM 可识别 JSON 语法错误，而 IAM Access Analyzer 将提供额外的策略检查和建议，以帮助您进一步优化策略。要了解策略验证的更多信息，请参阅 [验证 IAM 策略 \(p. 443\)](#)。要了解有关 IAM Access Analyzer 策略检查和可执行建议的更多信息，请参阅 [IAM Access Analyzer 策略验证](#)。

JSON 策略文档结构

如下图所示，JSON 策略文档包含以下元素：

- 文档顶部的可选策略范围信息
- 一个或多个单独语句

每个语句都包含有关单个权限的信息。如果一个策略包含多个语句，则 Amazon 会在评估它们时跨这些语句应用逻辑 OR。如果有多个策略应用于请求，则 Amazon 会在评估它们时跨所有这些策略应用逻辑 OR。



语句中的信息均含在一系列的元素内。

- Version - 指定要使用的策略语言版本。作为最佳实践，请使用最新的 2012-10-17 版本。
- Statement - 将该主要策略元素作为以下元素的容器。可以在一个策略中包含多个语句。
- Sid (可选) - 包括可选的语句 ID 以区分不同的语句。
- Effect - 使用 Allow 或 Deny 指示策略是允许还是拒绝访问。
- Principal (仅在某些情况下需要) - 如果创建基于资源的策略，您必须指示要允许或拒绝访问的账户、用户、角色或联合身份用户。如果要创建 IAM 权限策略以附加到用户或角色，则不能包含该元素。主体暗示为该用户或角色。
- Action - 包括策略允许或拒绝的操作列表。
- Resource (仅在某些情况下需要) - 如果创建 IAM 权限策略，您必须指定操作适用的资源列表。如果创建基于资源的策略，则该元素是可选的。如果不包含该元素，则该操作适用的资源是策略附加到的资源。
- Condition (可选) - 指定策略在哪些情况下授予权限。

要了解上述及其他更高级的策略元素，请参阅[IAM JSON 策略元素参考 \(p. 934\)](#)。

多个声明和多个策略

如果要为实体（用户或角色）定义多个权限，您可以在单个策略中使用多个语句。您也可以附加多个策略。如果您尝试在单个语句中定义多个权限，则策略可能没有授予预期访问权限。作为最佳实践，请按资源类型分解策略。

由于[策略的大小有限 \(p. 902\)](#)，可能需要对更复杂的权限使用多个策略。在单独的客户托管策略中创建权限的功能分组也是个好主意。例如，为 IAM 用户管理创建一个策略，为自我管理创建一个策略，并为 S3 存储桶管理创建另一个策略。无论多个语句和多个策略如何组合，Amazon 都会以相同方式[评估 \(p. 972\)](#)您的策略。

例如，以下策略具有三个语句，其中每个语句在单个账户中定义一组单独的权限。这些语句定义以下权限：

- Sid (语句 ID) 为 FirstStatement 的第一个语句让具有附加策略的用户更改自己的密码。该语句中的 Resource 元素是 "*" (这表示“所有资源”)。但实际上，ChangePassword API 操作 (或等效的 change-password CLI 命令) 仅影响发出请求的用户的密码。
- 第二个语句使用户可以列出其 Amazon 账户中的所有 Amazon S3 存储桶。该语句中的 Resource 元素是 "*" (这表示“所有资源”)。但由于策略没有为其他账户中的资源授予访问权限，因此，用户只能列出自己的 Amazon 账户中的存储桶。
- 第三个语句允许用户列出和检索名为 confidential-data 的存储桶中的任何对象，但是仅当使用 Multi-Factor Authentication (MFA) 对用户进行了身份验证时才能如此。策略中的 Condition 元素将强制实施 MFA 身份验证。

如果策略语句包含 Condition 元素，则仅当 Condition 元素计算为 true 时，语句才有效。在此示例中，Condition 在用户进行了 MFA 身份验证时计算为 true。如果用户没有进行 MFA 身份验证，则此 Condition 计算为 false。在这种情况下，此策略中的第三个语句不会应用，因此用户将无法访问 confidential-data 存储桶。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FirstStatement",
      "Effect": "Allow",
      "Action": ["iam:ChangePassword"],
      "Resource": "*"
    },
    {
      "Sid": "SecondStatement",
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
    {
      "Sid": "ThirdStatement",
      "Effect": "Allow",
      "Action": [
        "s3:List*",
        "s3:Get*"
      ],
      "Resource": [
        "arn:aws:s3:::confidential-data",
        "arn:aws:s3:::confidential-data/*"
      ],
      "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
    }
  ]
}
```

JSON 策略语法示例

以下基于身份的策略允许暗示的主体列出名为 example_bucket 的单个 Amazon S3 存储桶：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
  }
}
```

```
}
```

以下基于资源的策略可附加到 Amazon S3 存储桶。此策略允许特定 Amazon 账户的成员在名为 mybucket 的存储桶中执行任何 Amazon S3 操作。它允许可对存储桶或其中的对象执行的任何操作。(因为该策略仅向该账户授予信任, 所以仍必须向该账户中的各个用户授予执行指定 Amazon S3 操作的权限。)

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "1",
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::account-id:root"]},
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3::mybucket",
      "arn:aws:s3::mybucket/*"
    ]
  }]
}
```

要查看常见方案的示例策略, 请参阅[IAM 基于身份的策略示例 \(p. 391\)](#)。

授予最低权限

创建 IAM policy 时, 请遵循授予最小权限这一标准安全建议, 或仅授予执行任务所需的权限。确定用户和角色需要执行的操作, 然后制订允许他们仅执行这些任务的策略。

最开始只授予最低权限, 然后根据需要授予其它权限。这样做比起一开始就授予过于宽松的权限而后再尝试收紧权限来说更为安全。

作为最低权限的替代方案, 您可以使用 [Amazon 托管策略 \(p. 364\)](#) 或带通配符 * 权限的策略开始使用策略。考虑授予您的主体超出其完成工作所需的更多权限所带来的安全风险。监控这些主体以了解他们正在使用哪些权限。然后写入最低权限策略。

IAM 提供了多个选项来帮助您优化授予的权限。

- 了解访问级别分组 - 您可以使用访问级别分组来了解策略授予的访问级别。[策略操作 \(p. 944\)](#)被归类为 List、Read、Write、Permissions management 或 Tagging。例如, 您可以从 List 和 Read 访问级别中选择操作, 以向您的用户授予只读访问权限。要了解如何使用策略摘要来了解访问级别权限, 请参阅 [了解策略摘要内的访问级别摘要 \(p. 491\)](#)。
- 验证您的策略 — 您可以在创建和编辑 JSON 策略时使用 IAM Access Analyzer 执行策略验证。我们建议您查看和验证所有现有策略。IAM Access Analyzer 提供 100 多项策略检查来验证您的策略。当您策略中的语句允许我们认为过于宽容的访问时, 它会生成安全警告。在授予最小权限时, 您可以使用通过安全警告提供的可操作建议。要了解有关 IAM Access Analyzer 提供的策略检查的更多信息, 请参阅 [IAM Access Analyzer 策略验证](#)。
- 基于访问活动生成策略 — 为了帮助您优化授予的权限, 您可以根据 IAM 实体 (用户或角色) 的访问活动生成 IAM policy。IAM 访问分析器会查看您的 Amazon CloudTrail 日志并生成一个策略模板, 其中包含实体在指定时间框架内使用的权限。您可以使用模板创建具有精细权限的托管策略, 然后将其附加到 IAM 实体。这样, 您仅需授予用户或角色与特定使用案例中的 Amazon 资源进行交互所需的权限。要了解更多信息, 请参阅[“基于访问活动生成策略 \(p. 443\)”](#)。
- 使用上次访问的信息 — 另一项可帮助提供最低权限的功能是上次访问的信息。可以在 IAM 控制台详细信息页面上的访问顾问选项卡中查看此信息, 以了解 IAM 用户、组、角色或策略。上次访问的信息还包括有关上次访问某些服务 (如 Amazon EC2、IAM、Lambda 和 Amazon S3) 的操作的信息。如果您使用 Amazon Organizations 管理账户凭证登录, 则可以在 IAM 控制台的 Amazon Organizations 部分查看上次访问的服务信息。也可以使用 Amazon CLI 或 Amazon API 为 IAM 或 Organizations 中的实体或策略检索上次访问的信息报告。您可以使用该信息确定不需要的权限, 以便优化 IAM 或 Organizations 策略以更好地遵循最小权限原则。有关更多信息, 请参阅[使用上次访问的信息优化 Amazon 中的权限 \(p. 467\)](#)。

- 查看 Amazon CloudTrail 中的账户事件 — 要进一步减少权限，您可以在 Amazon CloudTrail Event history (事件历史记录) 中查看您的账户事件。CloudTrail 事件日志包含详细的事件信息，您可以用来减少策略的权限。这些日志仅包含 IAM 实体所需的操作和资源。有关更多信息，请参阅 Amazon CloudTrail 用户指南中的[在 CloudTrail 控制台中查看 CloudTrail 事件](#)。

有关更多信息，请参阅以下单个服务的策略主题，其中提供了如何针对特定产品的资源编写策略的示例。

- Amazon DynamoDB 开发人员指南中的[Amazon DynamoDB 的身份验证和访问控制](#)
- Amazon Simple Storage Service 用户指南中的[使用存储桶策略和用户策略](#)
- Amazon Simple Storage Service 用户指南中的[访问控制列表 \(ACL\) 概述](#)。

托管策略与内联策略

如果您需要在 IAM 中为身份设置权限，您必须决定是使用 Amazon 托管策略、客户托管策略还是内联策略。以下几节提供了有关每个类型的基于身份的策略以及何时使用它们的更多信息。

主题

- [Amazon 托管策略 \(p. 364\)](#)
- [客户托管策略 \(p. 365\)](#)
- [通过 Amazon 托管策略开始使用权限 \(p. 367\)](#)
- [内联策略 \(p. 367\)](#)
- [在托管策略与内联策略之间进行选择 \(p. 368\)](#)
- [将内联策略转换为托管策略 \(p. 369\)](#)
- [已弃用的 Amazon 托管策略 \(p. 370\)](#)

Amazon 托管策略

Amazon 托管策略 是由 Amazon 创建和管理的独立策略。Standalone policy (独立策略) 意味着策略有自己的 Amazon Resource Name (ARN)，其中包含策略名称。例如，arn:aws:iam::aws:policy/IAMReadOnlyAccess 是一个 Amazon 托管策略。有关 ARN 的更多信息，请参阅 [IAM ARN \(p. 896\)](#)。

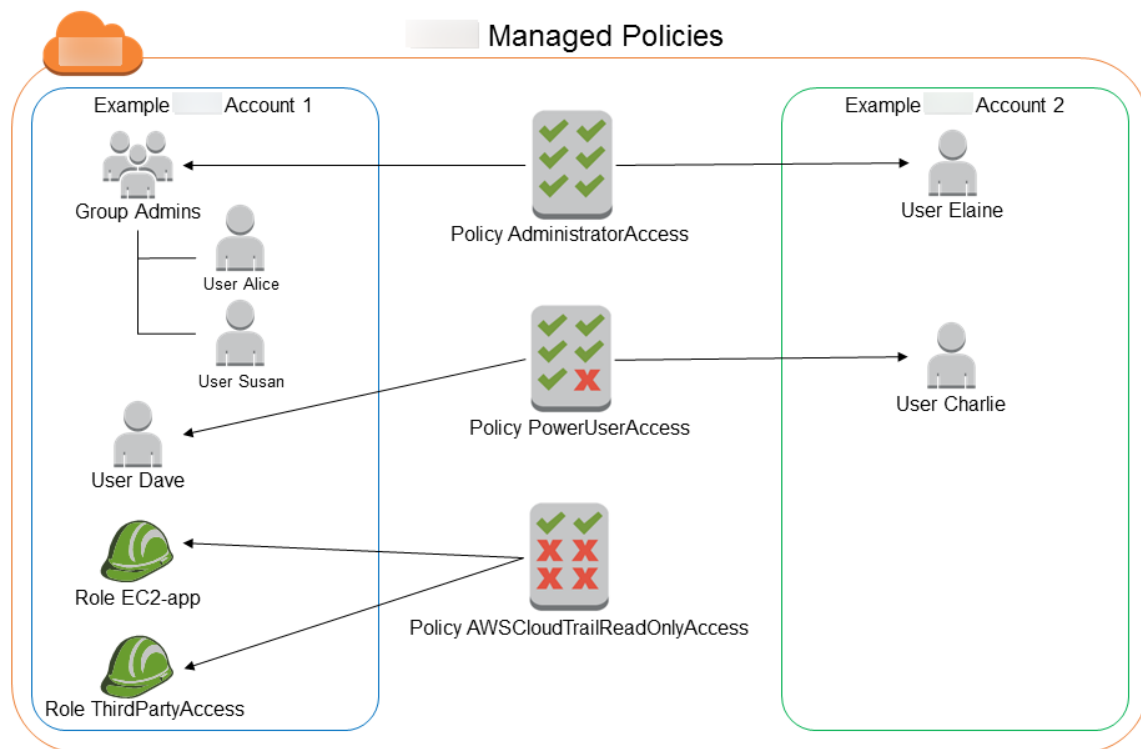
Amazon 托管策略可用于为很多常用案例提供权限。完全访问 Amazon 托管策略 (如 [AmazonDynamoDBFullAccess](#) 和 [IAMFullAccess](#)) 通过授予对服务的完全访问权限来定义服务管理员的权限。高级用户 Amazon 托管策略 (如 [AWSCodeCommitPowerUser](#) 和 [AWSKeyManagementServicePowerUser](#)) 是为高级用户设计的。部分访问 Amazon 托管策略 (如 [AmazonMobileAnalyticsWriteOnlyAccess](#) 和 [AmazonEC2ReadOnlyAccess](#)) 提供对 Amazon 服务的特定级别的访问权限，但未授予 [permissions management \(p. 492\)](#) (权限管理) 访问级别权限。利用 Amazon 托管策略，您可更轻松地向用户、组和角色分配相应权限，而不必自己编写策略。

那些专为工作职能设计的策略是一个特别有用的 Amazon 托管策略类别。这些策略与 IT 行业的常用工作职能紧密贴合。目的是让这些常见工作职能授予权限变得简单。使用工作职能策略的一个重要优势是，在推出新的服务和 API 操作时，Amazon 对这些策略进行维护和更新。例如，[AdministratorAccess](#) 工作职能提供对 Amazon 中的每个服务和资源的完全访问权限和权限委派。我们建议此策略仅用于账户管理员。对于需要对 IAM 和 Organizations 的有限访问权限以及对其他每个服务的完全访问权限的高级用户，请使用 [PowerUserAccess](#) 工作职能。有关工作职能策略的列表和说明，请参阅[工作职能的 Amazon 托管策略 \(p. 990\)](#)。

您不能更改 Amazon 托管策略中定义的权限。Amazon 有时会更新 Amazon 托管策略中定义的权限。在 Amazon 执行此操作时，更新会影响策略附加到的所有主体实体 (用户、组和角色)。在推出新的 Amazon 服务或为现有服务提供新的 API 调用时，Amazon 很可能会更新 Amazon 托管策略。例如，名为 ReadOnlyAccess 的 Amazon 托管策略提供针对所有 Amazon 服务和资源的只读访问权限。在 Amazon 推出

新的服务时，Amazon 将更新 ReadOnlyAccess 策略，以便为新服务添加只读权限。更新的权限会应用于策略附加到的所有主体实体。

下图对 Amazon 托管策略进行说明。该图显示三个 Amazon 托管策略：AdministratorAccess、PowerUserAccess 和 AmazonCloudTrailReadOnlyAccess。请注意，单个 Amazon 托管策略可以附加到不同 Amazon 账户中的主体实体，可以附加到单个 Amazon 账户中的不同主体实体。



客户托管策略

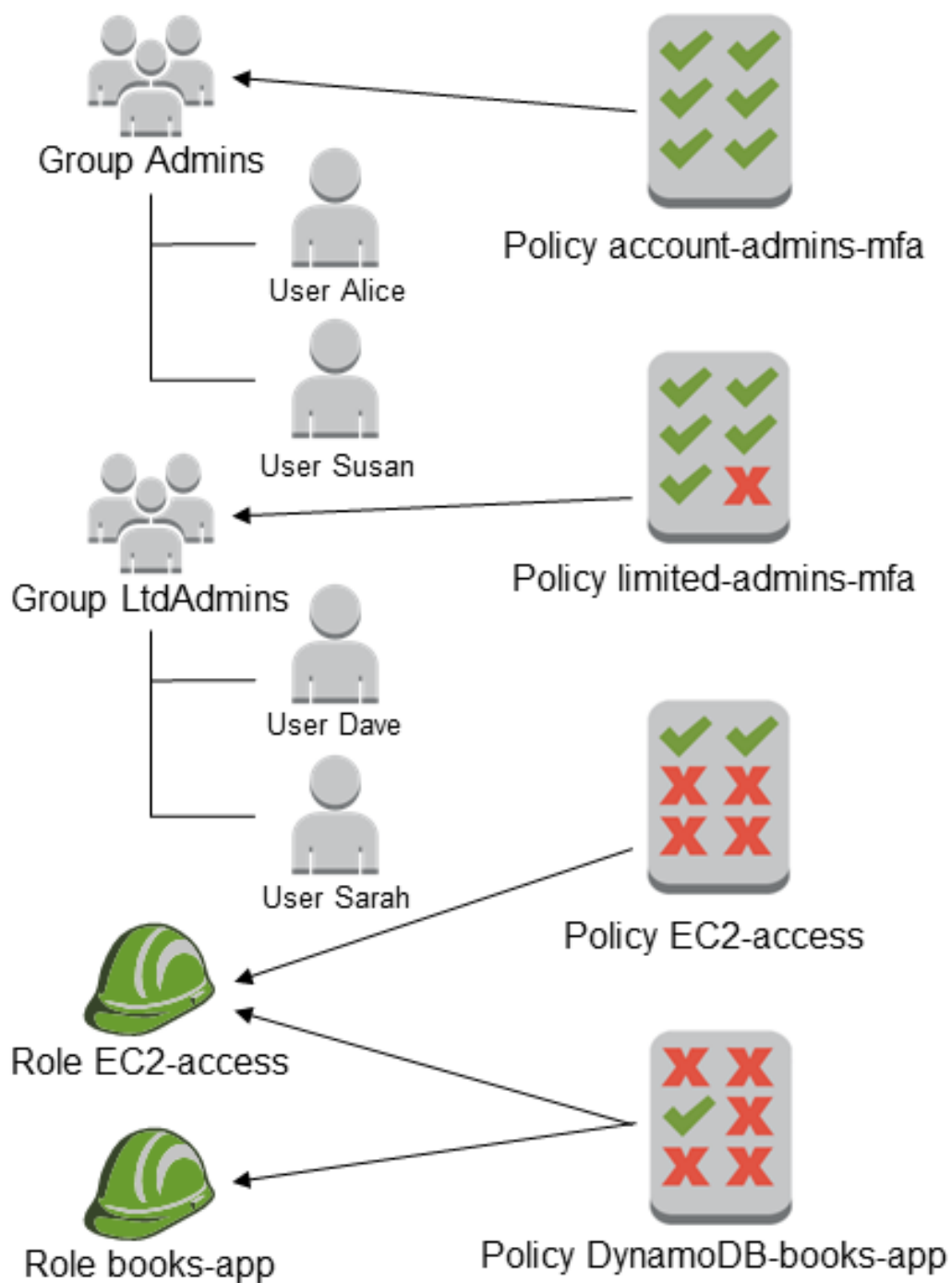
您可以创建您在自己的 Amazon 账户中管理的独立策略，我们称其为客户托管策略。随后可以将这些策略附加到您 Amazon 账户中的多个主体实体。将策略附加到主体实体时，便向实体授予了策略中定义的权限。

创建客户托管策略的理想方式是：首先复制一个现有 Amazon 托管策略。这样从一开始您就可以确信策略是正确的，只需根据您的环境进行自定义即可。

下面的示意图对客户托管策略进行说明。每个策略都是 IAM 中的一个实体，有自己的 [Amazon Resource Name \(ARN\)](#) (p. 896)，其中包含策略名称。请注意，同一策略可以附加到多个主体实体；例如，同一 DynamoDB-books-app 策略附加到两个不同的 IAM 角色。

Customer Managed Policies

Example Account



通过 Amazon 托管策略开始使用权限

我们建议使用[授予最低权限 \(p. 363\)](#)的策略，或仅授予执行任务所需的许可。授予最低权限的最安全方式是编写一个仅具有团队所需权限的客户管理型策略。您必须创建一个流程，以允许您的团队在必要时请求更多权限。创建仅为团队提供所需权限的[IAM 客户托管策略 \(p. 438\)](#)需要时间和专业知识。

要开始向您的 IAM 身份（用户、用户组和角色）添加权限，您可以使用[Amazon 托管策略 \(p. 364\)](#)。Amazon 托管策略不会授予最低权限。您必须考虑授予您的主体超出其完成工作所需的更多权限所带来的安全风险。

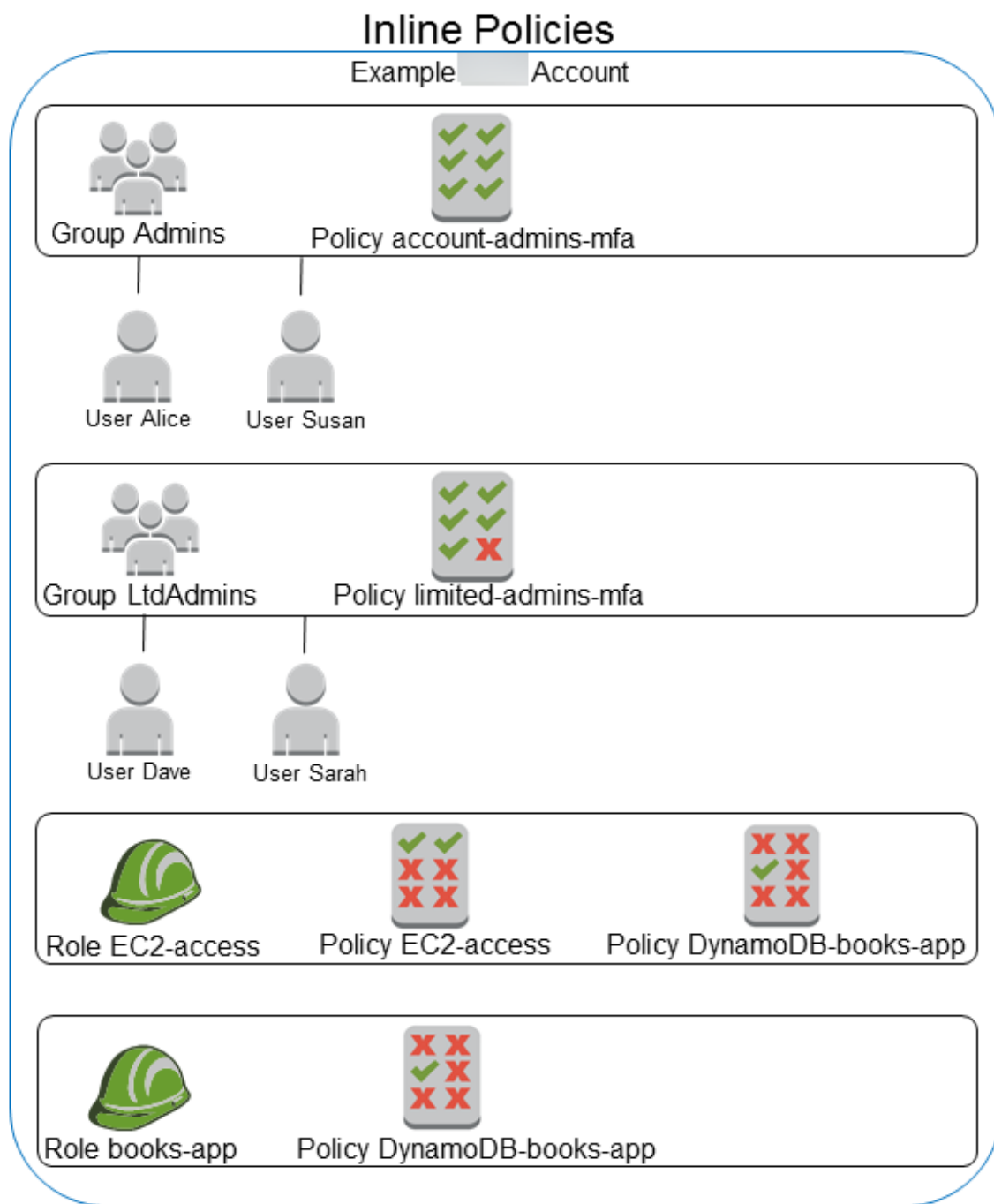
您可以将 Amazon 托管策略（包括任务函数）附加到任何 IAM 身份。要切换到最小权限权限，您可以运行 Amazon Identity and Access Management Access Analyzer 以使用 Amazon 托管策略监控主体。了解他们使用的权限后，您可以编写自定义策略或生成仅包含团队所需权限的策略。这中方法不太安全，但您能够以更灵活的方式了解您的团队如何使用 Amazon。

Amazon 托管策略可用于为很多常用案例提供权限。有关专为特定任务函数制定的 Amazon 托管策略的更多信息，请参阅[工作职能的 Amazon 托管策略 \(p. 990\)](#)。

内联策略

内联策略是嵌入在 IAM 身份（用户、组或角色）中的策略。也就是说，策略是身份的固有组成部分。您可以创建策略并将其嵌入到身份中，无论是在创建身份时还是之后。

下面的示意图对内联策略进行说明。每个策略都是用户、组或角色的固有部分。请注意，两个角色包含同一策略（DynamoDB-books-app 策略），但是它们不共享单个策略；每个角色都有自己的策略副本。



在托管策略与内联策略之间进行选择

不同类型的策略适用于不同的使用案例。在大多数情况下，我们建议使用托管策略而不是内联策略。

托管策略具备以下功能：

可重复使用性

单个托管策略可以附加到多个主体实体 (用户、组和角色)。实际上,您可以创建策略库,在库中定义对您 Amazon 账户有用的权限,然后根据需要将这些策略附加到主体实体。

集中变更管理

更改托管策略时,更改会应用于策略附加到的所有主体实体。例如,如果要针对新 Amazon API 添加权限,则可以更新托管策略以添加权限。(如果使用 Amazon 托管策略,Amazon 会更新为该策略。)更新策略时,更改会应用于策略附加到的所有主体实体。相比较而言,要更改内联策略,您必须分别编辑包含该策略的每个身份。例如,如果一个组和一个角色都包含同一内联策略,则您必须分别编辑这两个主体实体才能更改该策略。

版本控制和回滚

在更改客户托管策略时,更改的策略不会覆盖现有的策略。而是由 IAM 创建新的托管策略版本。IAM 最多可以存储五个版本的客户管理策略。如果需要,可以使用策略版本将策略还原为较早版本。

Note

策略版本与 Version 策略元素不同。Version 策略元素用在策略之中,用于定义策略语言的版本。要了解策略版本的更多信息,请参阅[the section called "IAM 策略版本控制" \(p. 458\)](#)。

要了解 Version 策略元素的更多信息,请参阅[IAM JSON 策略元素: Version \(p. 934\)](#)。

委派权限管理

您可以允许您 Amazon 账户中的用户附加和分离策略,同时保持对这些策略中定义的权限的控制。实际上,您可以将一些用户指定为完全管理员,即可以创建、更新和删除策略的管理员。然后,您可以将其其他用户指定为受限管理员。即,可以将策略(限于您允许附加的策略)附加到其他主体实体的管理员。

有关委派权限管理的更多信息,请参阅[控制对策略的访问 \(p. 384\)](#)。

Amazon 托管策略的自动更新

Amazon 维护 Amazon 托管策略并在需要时更新它们(例如,针对新 Amazon 服务添加权限),您不必进行更改。更新会自动应用于已附加了 Amazon 托管策略的主体实体。

使用内联策略

如果您要在策略与应用它的身份之间维持严格的一对一关系,则内联策略十分有用。例如,您要确保策略中的权限不会无意中分配给预期身份之外的身份。使用内联策略时,策略中的权限不可能意外分配给错误的身份。此外,使用 Amazon Web Services Management Console 删除身份时,也会删除嵌入在身份中的策略。因为它们是主体实体的一部分。

将内联策略转换为托管策略

如果您的账户中具有内联策略,则将其转换为托管策略。为此,请将策略复制到新的托管策略。接下来,将新策略附加到具有内联策略的身份。然后,请删除内联策略。您可使用以下说明执行此操作。

将内联策略转换为托管策略

1. 登录 Amazon Web Services Management Console,然后通过以下网址打开 IAM 控制台:<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中,选择 Groups (组)、Users (用户) 或 Roles (角色)。
3. 在列表中,请选择具有要修改的策略的用户组、用户或角色的名称。
4. 请选择 Permissions 选项卡。
5. 对于用户组,请选择要移除的内联策略的名称。对于用户和角色,选择 Show **n** more (再显示 n 个)(如有必要),然后选择要删除的内联策略旁边的箭头。

6. 复制策略的 JSON 策略文档。
 7. 在导航窗格中，选择 Policies (策略)。
 8. 选择 Create policy (创建策略)，然后选择 JSON 选项卡。
 9. 将现有文本替换为您的 JSON 策略文本，然后选择 Review policy (查看策略)。
 10. 为您的策略输入名称，然后选择 Create policy (创建策略)。
 11. 在导航窗格中，请选择 Groups (组)、Users (用户) 或 Roles (角色)，然后再次选择具有要删除的策略的用户组、用户或角色的名称。
 12. 对于用户组，请选择 Permissions (权限) 选项卡。对于用户和角色，选择 Add permissions (添加权限)。
 13. 对于用户组，选中新策略名称旁边的复选框，选择 Add permissions (附加权限) 然后选择 Attach Policy (附加策略)。对于用户或角色，选择 Add permissions (添加权限)。在下一页上，选择 Attach existing policies directly (直接附加现有策略)，选中您的新策略名称旁边的复选框，选择 Next: Review (下一步: 查看)，然后选择 Add permissions (添加权限)。
- 您将返回您的用户组、用户或角色的 Summary (摘要) 页面。
14. 对于用户组，选中要移除的内联策略旁边的复选框，然后选择 Remove (移除)。对于用户或角色，选择要删除的内联策略旁边的 X。

已弃用的 Amazon 托管策略

为了简化权限的分配，Amazon 提供[托管策略 \(p. 364\)](#) — 这是一项预定义策略，可随时附加到您的 IAM 用户、组和角色。

有时 Amazon 需要向现有策略添加新权限，例如在引入新服务时。向现有策略添加新权限不会中断或移除任何功能或能力。

但是，如果所需的更改在应用于现有策略时可能影响客户，则 Amazon 可能会选择创建新策略。例如，从现有策略中移除权限可能中断依赖该策略的任何 IAM 实体或应用程序的权限，从而可能中断关键操作。

因此，当需要此类更改时，Amazon 会根据所需更改创建一种全新的策略，并将其提供给客户。然后，旧策略被标记为已弃用。已弃用的托管策略将显示在 IAM 控制台中的 Policies (策略) 列表中，旁边带有一个警告图标。

已弃用的策略具有以下特性：

- 该策略继续为所有当前附加的用户、组和角色工作。什么都不会中断。
- 该策略不能附加到任何新的用户、组或角色。该策略若与当前实体分离则不能重新附加。
- 在与所有当前实体分离以后，该策略将不再显示，并且再不能以任何方式使用。

如果任何用户、组或角色需要该策略，则必须附加新的策略。当您收到某策略已弃用的通知时，我们建议您立即计划将所有用户、组和角色附加到替换策略，并将它们与已弃用的策略分离。继续使用已弃用的策略可能会带来风险，只能通过切换到替换策略来降低这些风险。

IAM 实体的权限边界

Amazon 对于 IAM 实体（用户或角色）支持权限边界。权限边界是一个高级功能，它使用托管策略设置基于身份的策略可以为 IAM 实体授予的最大权限。实体的权限边界仅允许实体执行其基于身份的策略和权限边界同时允许的操作。

有关策略类型的更多信息，请参阅[策略类型 \(p. 355\)](#)。

您可以使用 Amazon 托管策略或客户托管策略为 IAM 实体（用户或角色）设置边界。该策略限制用户或角色的最大权限。

例如，假设名为 ShirleyRodriguez 的 IAM 用户应仅允许管理 Amazon S3、Amazon CloudWatch 和 Amazon EC2。要执行此规则，您可以使用以下策略为 ShirleyRodriguez 用户设置权限边界：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "cloudwatch:*",
        "ec2:*"
      ],
      "Resource": "*"
    }
  ]
}
```

当使用策略为用户设置权限边界时，它会限制用户的权限，但自己不提供权限。在本示例中，策略设置 ShirleyRodriguez 的最大权限作为 Amazon S3、CloudWatch 和 Amazon EC2 中的所有操作。Shirley 无法在任何其他服务（包括 IAM）中执行操作，即使她有一个允许这样做的权限策略也是如此。例如，您可以将以下策略添加给 ShirleyRodriguez 用户：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:CreateUser",
    "Resource": "*"
  }
}
```

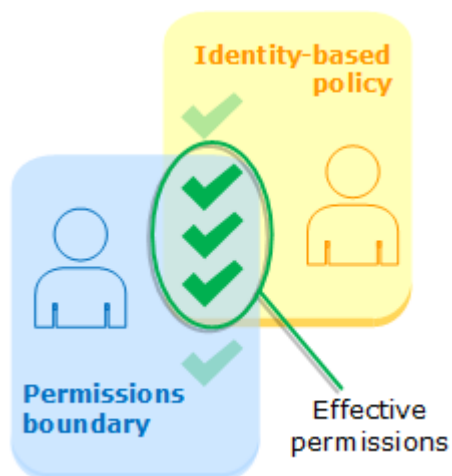
此策略允许在 IAM 中创建用户。如果您将此权限策略附加到 ShirleyRodriguez 用户，然后 Shirley 尝试创建用户，该操作会失败。它由于权限边界不允许执行 iam:CreateUser 操作而失败。鉴于这两个策略，Shirley 无权在 Amazon 中执行任何操作。您必须添加不同的权限策略以允许其他服务中的操作，例如 Amazon S3。或者，您也可以更新权限边界，以允许她在 IAM 中创建用户。

评估具有边界的有效权限

IAM 实体（用户或角色）的权限边界设置实体可以具有的最大权限。这可以更改该用户或角色的有效权限。实体的有效权限是影响用户或角色的所有策略所授予的权限。在账户中，基于身份的策略、基于资源的策略、权限边界、Organizations SCP 或会话策略可以影响实体的权限。有关不同类型的策略的更多信息，请参阅 [IAM 中的策略和权限](#)。（p. 355）。

如果以下任一策略类型显式拒绝操作的访问权限，则请求会被拒绝。多个权限类型向实体授予的权限更复杂。有关 Amazon 如何评估策略的更多信息，请参阅 [策略评估逻辑](#)（p. 972）。

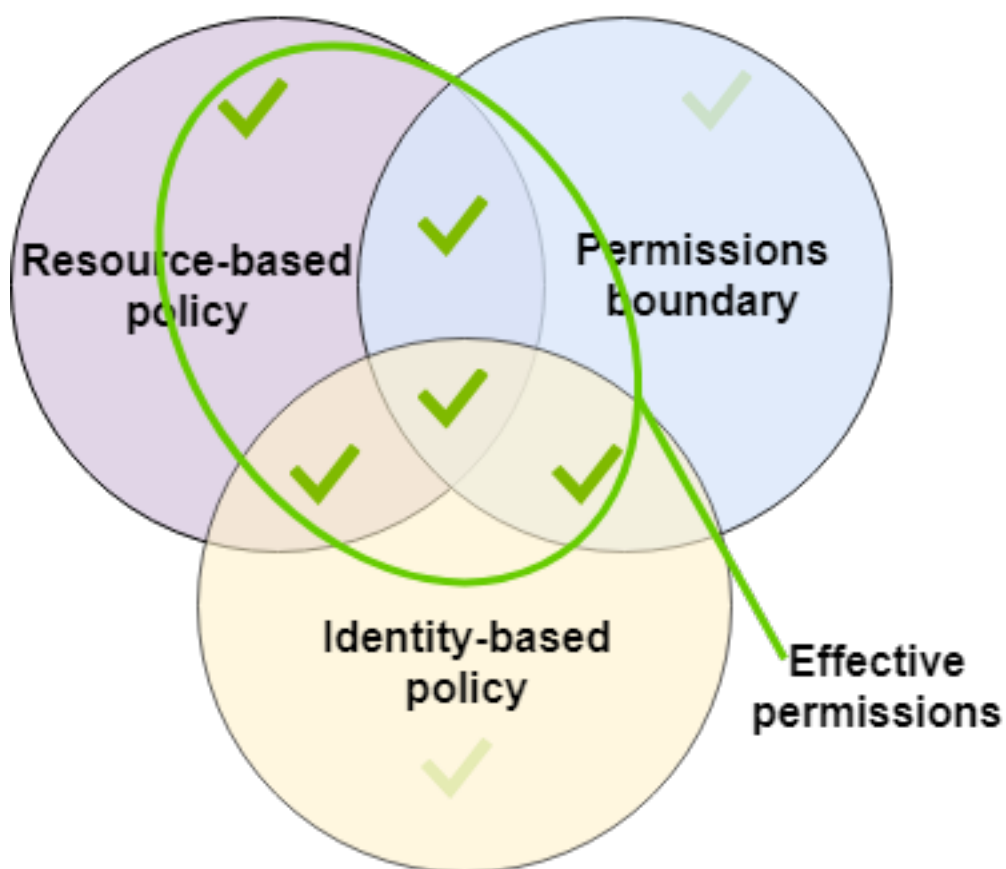
基于身份的策略及边界 - 基于身份的策略是附加到用户、用户组或角色的内联或托管策略。基于身份的策略向实体授予权限，而权限边界限制这些权限。有效的权限是两种策略类型的交集。其中任一项策略中的显式拒绝将覆盖允许。



基于资源的策略 - 基于资源的策略控制指定的主体可以访问策略附加到的资源。

针对 IAM 用户的基于资源的策略

在同一账户中，向 IAM 用户 ARN（即非联合身份用户会话）授予权限的基于资源的策略不受基于身份的策略或权限边界中的隐式拒绝限制。



针对 IAM 用户的基于资源的策略

IAM 角色 — 授予 IAM 角色 ARN 权限的基于资源的策略受权限边界或会话策略中隐式拒绝的限制。

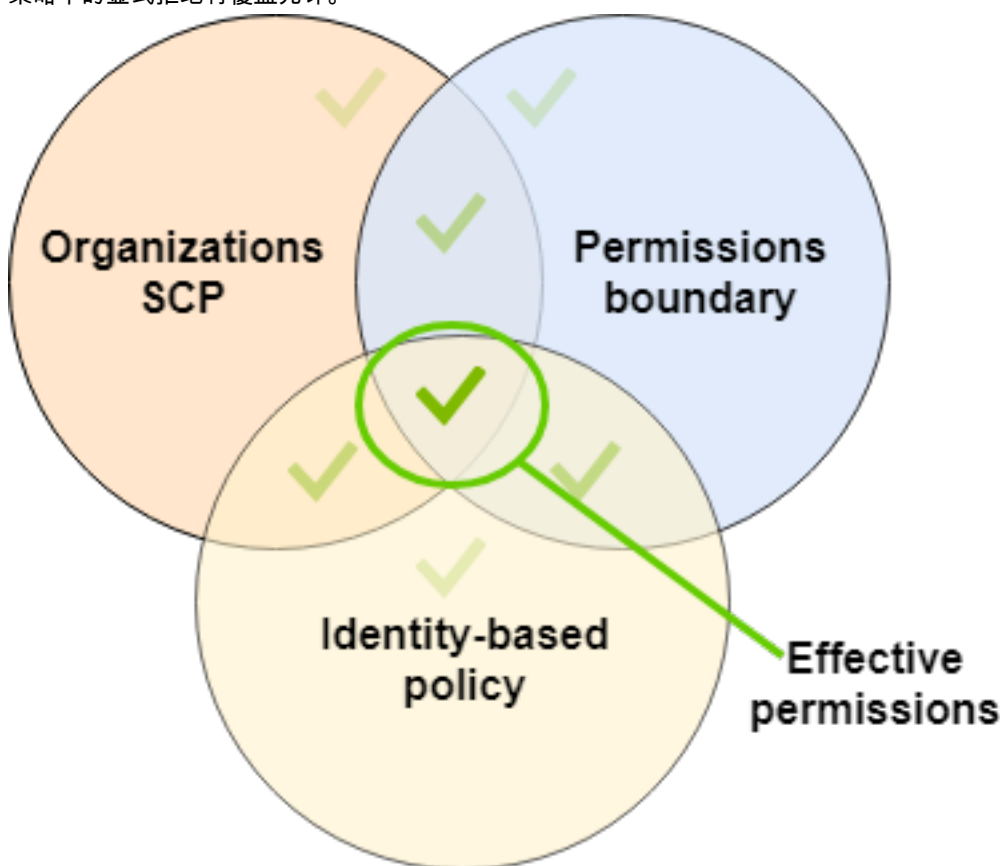
IAM 角色会话 — 在同一账户中，向 IAM 角色会话 ARN 授予权限的策略直接向所担任的角色会话授予权限。直接授予会话的权限不受基于身份的策略、权限边界或会话策略中的隐式拒绝的限制。当您担任角色并提出请求时，发出请求的主体是 IAM 角色会话 ARN，而不是角色本身的 ARN。

针对 IAM 联合身份用户的基于资源的策略

IAM 联合身份用户会话 — IAM 联合身份用户会话是通过调用 [GetFederationToken \(p. 307\)](#) 创建的会话。当联合身份用户发出请求时，发出请求的主体是联合身份用户 ARN，而不是联合身份的 IAM 用户的 ARN。在同一个账户中，将权限授予联合身份用户 ARN 的基于资源的策略直接将权限授予会话。直接授予会话的权限不受基于身份的策略、权限边界或会话策略中的隐式拒绝的限制。

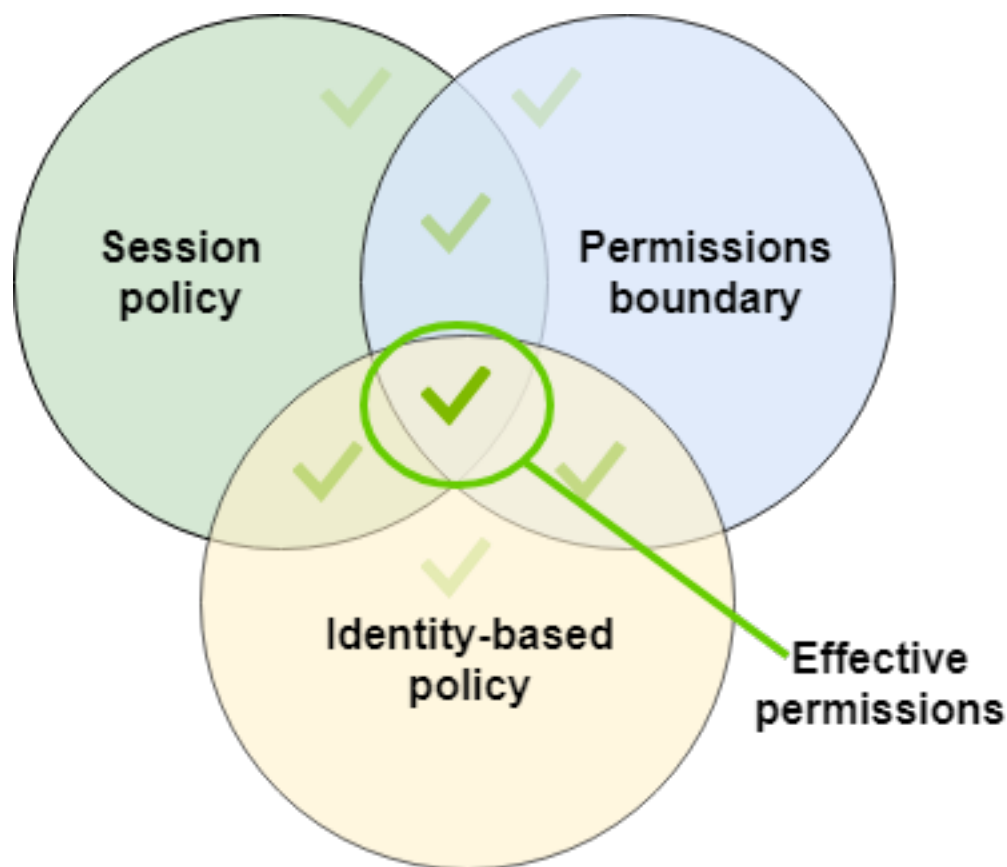
但是，如果基于资源的策略向联合身份的 IAM 用户的 ARN 授予权限，则联合身份用户在会话期间发出的请求将受权限边界或会话策略中隐式拒绝的限制。

Organizations SCP - SCP 应用于整个 Amazon 账户。它们限制账户中的主体所提出的每个请求的权限。IAM 实体（用户或角色）可能会发出受 SCP、权限边界和基于身份的策略影响的请求。在这种情况下，只有在所有三种策略类型都允许时，才允许发出该请求。有效的权限是所有三种策略类型的交集。任一项策略中的显式拒绝将覆盖允许。



您可以在 Amazon Organizations 中了解[您的账户是否为某个组织的成员](#)。组织成员可能会受 SCP 的影响。要使用 Amazon CLI 命令或 Amazon API 操作查看该数据，您必须具有 Organizations 实体的 `organizations:DescribeOrganization` 操作的权限。您必须具有额外的权限才能在 Organizations 控制台中执行该操作。要了解 SCP 是否拒绝访问特定的请求或更改您的有效权限，请与您的 Amazon Organizations 管理员联系。

会话策略 – 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。会话的权限来自用于创建会话的 IAM 实体（用户或角色）和会话策略。该实体的基于身份的策略权限受会话策略和权限边界所限制。这组策略类型的有效权限是所有三种策略类型的交集。任一项策略中的显式拒绝将覆盖允许。有关会话策略的更多信息，请参阅[会话策略](#)。



使用权限边界将责任委派给其他人

您可以使用权限边界将权限管理任务（如用户创建）委派给您账户中的 IAM 用户。这允许其他人在特定权限边界内代表您执行任务。

例如，假定 Maria 是 X 公司 Amazon 账户的管理员。她希望将用户创建责任委派给 Zhang。但是，她必须确保 Zhang 创建的用户符合以下公司规则：

- 用户不能使用 IAM 来创建或管理用户、组、角色或策略。
- 拒绝用户访问 Amazon S3 logs 存储桶，并且用户无法访问 i-1234567890abcdef0 Amazon EC2 实例。
- 用户无法删除自己的边界策略。

为执行这些规则，Maria 完成以下任务，下面包括其详细信息：

1. Maria 创建 xCompanyBoundaries 托管策略以用作账户中的所有新用户的权限边界。
2. Maria 创建 DelegatedUserBoundary 托管策略并将其指定为 Zhang 的权限边界。Maria 记录了其管理员 IAM 用户的 ARN，并在策略中使用它来阻止 Zhang 访问它。
3. Maria 创建 DelegatedUserPermissions 托管策略并将其作为 Zhang 的权限策略进行附加。
4. Maria 告诉 Zhang 他的新责任和限制。

任务 1：Maria 必须首先创建托管策略来为新用户定义边界。Maria 将允许 Zhang 向用户授予他们所需的权限策略，但她希望限制这些用户。为此，她创建以下名为 xCompanyBoundaries 的客户托管策略。该策略执行以下操作：

- 允许用户完全访问一些服务
- 允许在 IAM 控制台中进行有限的自我管理访问。这意味着，他们可以在登录控制台后更改密码。他们不能设置初始密码。要允许此操作，请将 "*LoginProfile" 操作添加到 AllowManageOwnPasswordAndAccessKeys 语句。
- 拒绝用户访问 Amazon S3 日志存储桶或 i-1234567890abcdef0 Amazon EC2 实例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ServiceBoundaries",
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "cloudwatch:*",
        "ec2:*",
        "dynamodb:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowIAMConsoleForCredentials",
      "Effect": "Allow",
      "Action": [
        "iam:ListUsers",
        "iam:GetAccountPasswordPolicy"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowManageOwnPasswordAndAccessKeys",
      "Effect": "Allow",
      "Action": [
        "iam:*AccessKey*",
        "iam:ChangePassword",
        "iam:GetUser",
        "iam:*ServiceSpecificCredential*",
        "iam:*SigningCertificate*"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "DenyS3Logs",
      "Effect": "Deny",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::logs",
        "arn:aws:s3:::logs/*"
      ]
    },
    {
      "Sid": "DenyEC2Production",
      "Effect": "Deny",
      "Action": "ec2:*",
      "Resource": "arn:aws:ec2:*:*:instance/i-1234567890abcdef0"
    }
  ]
}
```

每个语句用于不同目的：

1. 此策略的 `ServiceBoundaries` 语句允许完全访问指定的 Amazon 服务。这意味着，新用户在这些服务中的操作仅受附加到该用户的权限策略的限制。
2. `AllowIAMConsoleForCredentials` 语句允许访问以列出所有 IAM 用户。要在 Amazon Web Services Management Console 中导航用户页面，此访问权限是必需的。它还允许查看账户的密码要求，这在更改自己的密码时是必需的。
3. `AllowManageOwnPasswordAndAccessKeys` 语句允许用户仅管理其自己的控制台密码和编程访问密钥。如果 Zhang 或其他管理员为新用户提供具有完全 IAM 访问权限的权限策略，该语句是非常重要的。在这种情况下，该用户可能会更改自己或其他用户的权限。此语句可防止这种情况的发生。
4. `DenyS3Logs` 语句显式拒绝对 logs 存储桶的访问。
5. `DenyEC2Production` 语句显式拒绝对 `i-1234567890abcdef0` 实例的访问。

任务 2：Maria 希望允许 Zhang 创建所有 X 公司用户，但仅具有 `XCompanyBoundaries` 权限边界。她创建以下名为 `DelegatedUserBoundary` 的客户托管策略。此策略定义 Zhang 可以具有的最大权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateOrChangeOnlyWithBoundary",
      "Effect": "Allow",
      "Action": [
        "iam:CreateUser",
        "iam:DeleteUserPolicy",
        "iam:AttachUserPolicy",
        "iam:DetachUserPolicy",
        "iam:PutUserPermissionsBoundary",
        "iam:PutUserPolicy"
      ],
      "Resource": "*",
      "Condition": { "StringEquals": {
        "iam:PermissionsBoundary": "arn:aws:iam::123456789012:policy/XCompanyBoundaries"
      } }
    },
    {
      "Sid": "CloudWatchAndOtherIAMTasks",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:*",
        "iam:GetUser",
        "iam:ListUsers",
        "iam:DeleteUser",
        "iam:UpdateUser",
        "iam:CreateAccessKey",
        "iam:CreateLoginProfile",
        "iam:DeleteLoginProfile",
        "iam:GetAccountPasswordPolicy",
        "iam:GetLoginProfile",
        "iam:ListGroups",
        "iam:ListGroupsForUser",
        "iam:CreateGroup",
        "iam:GetGroup",
        "iam:DeleteGroup",
        "iam:UpdateGroup",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:DeletePolicyVersion",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:GetUserPolicy",
        "iam:GetRolePolicy",
        "iam:ListAccessKeys",

```

```

        "iam:ListSigningCertificates",
        "iam:ListSSHPublicKeys",
        "iam:ListServiceSpecificCredentials",
        "iam:ListMFADevices",
        "iam:ListPolicies",
        "iam:ListPolicyVersions",
        "iam:ListEntitiesForPolicy",
        "iam:ListUserPolicies",
        "iam:ListAttachedUserPolicies",
        "iam:ListRolePolicies",
        "iam:ListAttachedRolePolicies",
        "iam:SetDefaultPolicyVersion",
        "iam:SimulatePrincipalPolicy",
        "iam:SimulateCustomPolicy"
    ],
    "NotResource": "arn:aws:iam::123456789012:user/Maria"
  },
  {
    "Sid": "NoBoundaryPolicyEdit",
    "Effect": "Deny",
    "Action": [
      "iam:CreatePolicyVersion",
      "iam:DeletePolicy",
      "iam:DeletePolicyVersion",
      "iam:SetDefaultPolicyVersion"
    ],
    "Resource": [
      "arn:aws:iam::123456789012:policy/XCompanyBoundaries",
      "arn:aws:iam::123456789012:policy/DelegatedUserBoundary"
    ]
  },
  {
    "Sid": "NoBoundaryUserDelete",
    "Effect": "Deny",
    "Action": "iam:DeleteUserPermissionsBoundary",
    "Resource": "*"
  }
]
}

```

每个语句用于不同目的：

1. `CreateOrChangeOnlyWithBoundary` 语句允许 Zhang 创建 IAM 用户，但仅当他使用 `XCompanyBoundaries` 策略设置权限边界时。此语句还允许他为现有用户设置权限边界，但仅使用该相同策略。最后，此语句允许 Zhang 管理设置了此权限边界的用户的权限策略。
2. `CloudWatchAndOtherIAMTasks` 语句允许 Zhang 完成其他用户、组和策略管理任务。他有权重置密码并为条件键中未列出的任何 IAM 用户创建访问密钥。这使他能够帮助用户解决登录问题。
3. `NoBoundaryPolicyEdit` 语句拒绝 Zhang 访问以更新 `XCompanyBoundaries` 策略。不允许他更改用于为自己或其他用户设置权限边界的任何策略。
4. `NoBoundaryUserDelete` 语句拒绝 Zhang 访问以便为他或其他用户删除权限边界。

Maria 然后将 `DelegatedUserBoundary` 策略指定为 Zhang 用户的[权限边界 \(p. 85\)](#)。

任务 3：由于权限边界限制最大权限，但自己不授予访问权限，Maria 必须为 Zhang 创建权限策略。她创建以下名为 `DelegatedUserPermissions` 的策略。此策略定义 Zhang 在定义的边界内可以执行的操作。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAM",

```

```
        "Effect": "Allow",
        "Action": "iam:*",
        "Resource": "*"
    },
    {
        "Sid": "CloudWatchLimited",
        "Effect": "Allow",
        "Action": [
            "cloudwatch:GetDashboard",
            "cloudwatch:GetMetricData",
            "cloudwatch:ListDashboards",
            "cloudwatch:GetMetricStatistics",
            "cloudwatch:ListMetrics"
        ],
        "Resource": "*"
    },
    {
        "Sid": "S3BucketContents",
        "Effect": "Allow",
        "Action": "s3:ListBucket",
        "Resource": "arn:aws:s3:::ZhangBucket"
    }
]
}
```

每个语句用于不同目的：

1. 此策略的 IAM 语句允许 Zhang 完全访问 IAM。但是，由于他的权限边界只允许某些 IAM 操作，他的有效 IAM 权限仅受他的权限边界限制。
2. CloudWatchLimited 语句允许 Zhang 在 CloudWatch 中执行五种操作。他的权限边界允许 CloudWatch 中的所有操作，因此他的有效 CloudWatch 权限仅受他的权限策略限制。
3. S3BucketContents 语句允许 Zhang 列出 ZhangBucket Amazon S3 存储桶。但是，他的权限边界不允许任何 Amazon S3 操作，因此无论他的权限策略如何，他都无法执行任何 S3 操作。

Note

Zhang 的策略允许他创建一个用户，然后可以访问自己无法访问的 Amazon S3 资源。通过委托这些管理操作，Maria 实际上相信张能够访问 Amazon S3。

Maria 然后将 DelegatedUserPermissions 策略作为 Zhang 用户的权限策略进行附加。

任务 4：她向 Zhang 提供创建新用户的说明。她告诉他，他可以创建具有所需任何权限的新用户，但他必须为他们分配 xCompanyBoundaries 策略作为权限边界。

Zhang 完成以下任务：

1. Zhang 使用 Amazon Web Services Management Console [创建用户 \(p. 73\)](#)。他为该用户键入用户名 Nikhil 并启用控制台访问。他清除 Requires password reset (需要重置密码) 旁边的复选框，因为上述策略仅允许用户在登录到 IAM 控制台后才能更改密码。
2. 在 Set permissions (设置权限) 页面上，Zhang 选择允许 Nikhil 执行其工作的 IAMFullAccess 和 AmazonS3ReadOnlyAccess 权限策略。
3. Zhang 跳过设置权限边界部分，忘记 Maria 的说明。
4. Zhang 查看用户详细信息并选择创建用户。

操作失败，并且拒绝访问。Zhang 的 DelegatedUserBoundary 权限边界要求他创建的任何用户都将 xCompanyBoundaries 策略用作权限边界。

5. Zhang 返回到上一页。在设置权限边界部分中，他选择了 xCompanyBoundaries 策略。
6. Zhang 查看用户详细信息并选择创建用户。

将创建用户。

当 Nikhil 登录时，他有权访问 IAM 和 Amazon S3，但权限边界拒绝的操作除外。例如，他可以在 IAM 中更改自己的密码，但无法创建另一个用户或编辑他的策略。Nikhil 具有对 Amazon S3 的只读访问权限。

如果某个人在 logs 存储桶中添加基于资源的策略以允许 Nikhil 将对象放入该存储桶中，他仍然无法访问该存储桶。原因是，其权限边界显式拒绝到 logs 存储桶执行的任何操作。任何策略类型中的显式拒绝都会导致请求被拒绝。不过，如果附加到 Secrets Manager 密钥的基于资源的策略允许 Nikhil 执行 `secretsmanager:GetSecretValue` 操作，则 Nikhil 可以检索和解密该密钥。原因是，其权限边界未显式拒绝 Secrets Manager 操作，并且权限边界中的隐式拒绝不限制基于资源的策略。

基于身份的策略和基于资源的策略

策略是 Amazon 中的对象；在与标识或资源相关联时，策略定义它们的权限。在创建权限策略以限制对资源的访问时，您可以选择基于身份的策略或基于资源的策略。

基于身份的策略附加到 IAM 用户、组或角色。这些策略可让您指定该身份可执行哪些操作（其权限）。例如，您可以将策略附加到名为 John 的 IAM 用户，以声明允许他执行 Amazon EC2 `RunInstances` 操作。该策略可能会进一步声明，John 可以从名为 MyCompany 的 Amazon DynamoDB 表中获取项目。您也可以允许 John 管理自己的 IAM 安全凭证。基于标识的策略可以是[托管或内联](#) (p. 364) 的。

基于资源的策略附加到某个资源。例如，您可以将基于资源的策略附加到 Amazon S3 存储桶、Amazon SQS 队列、VPC 终端节点和 Amazon Key Management Service 加密密钥。有关支持基于资源的策略的服务列表，请参阅[使用 IAM 的 Amazon 服务](#) (p. 908)。

通过使用基于资源的策略，您可以指定哪些用户有权访问资源，以及他们可以对资源执行哪些操作。要了解您信任区域之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅[什么是 IAM Access Analyzer ?](#)。基于资源的策略只有内联的，没有托管的。

Note

基于资源的策略与资源级权限不同。如本主题所述，您可以直接将基于资源的策略附加到某一资源。资源级权限是指使用 [ARN](#) (p. 896) 在策略中指定各项资源的能力。只有部分 Amazon 服务支持基于资源的策略。有关支持基于资源的策略和资源级权限的服务列表，请参阅[使用 IAM 的 Amazon 服务](#) (p. 908)。

要了解基于身份的策略和基于资源的策略如何在同一个账户中进行交互，请参阅[评估单个账户中的策略](#) (p. 973)。

要了解这些策略如何在账户之间进行交互，请参阅[跨账户策略评估逻辑](#) (p. 981)。

要更好地了解这些概念，请查看下图。123456789012 账户的管理员已将基于身份的策略附加到 JohnSmith、CarlosSalazar 和 MaryMajor 用户。可对特定资源执行这些策略中的一些操作。例如，用户 JohnSmith 可以对 Resource X 执行一些操作。这是基于身份的策略中的资源级权限。管理员还将基于资源的策略添加到 Resource X、Resource Y 和 Resource Z。利用基于资源的策略，您可以指定谁可以访问资源。例如，Resource X 上的基于资源的策略允许用户 JohnSmith 和 MaryMajor 列出和读取资源。

Account ID: 123456789012

Identity-based policies

JohnSmith

Can List, Read
On Resource X

CarlosSalazar

Can List, Read
On Resource Y,Z

MaryMajor

Can List, Read, Write
On Resource X,Y,Z

ZhangWei

No policy

Resource-based policies

Resource X

JohnSmith: Can List, Read
MaryMajor: Can List, Read

Resource Y

CarlosSalazar: Can List, Write
ZhangWei: Can List, Read

Resource Z

CarlosSalazar: Denied access
ZhangWei: Allowed full access

123456789012 账户示例允许以下用户执行列出的操作：

- JohnSmith - 可以对 Resource X 执行列出和读取操作。已通过基于身份的策略（针对其用户）和基于资源的策略（针对 Resource X）向他授予此权限。
- CarlosSalazar - Carlos 可对 Resource Y 执行列出、读取和写入操作，但被拒绝访问 Resource Z。Carlos 的基于身份的策略允许他对 Resource Y 执行列出和读取操作。基于 Resource Y 资源的策略还向他授予写入权限。不过，尽管其基于身份的策略允许他访问 Resource Z，但基于 Resource Z 资源的策略拒绝该访问。显式 Deny 会覆盖 Allow，并拒绝他对 Resource Z 进行的访问。有关更多信息，请参阅[策略评估逻辑](#) (p. 972)。
- MaryMajor - Mary 可对 Resource X、Resource Y 和 Resource Z 执行列出、读取和写入操作。与基于资源的策略相比，她的基于身份的策略允许她对更多资源执行更多操作，但都不拒绝访问。
- ZhangWei - 具有 Resource Z 的完全访问权限。Zhang 没有基于身份的策略，但基于 Resource Z 资源的策略向他授予对资源的完全访问权限。Zhang 还可以对 Resource Y 执行列出和读取操作。

基于身份的策略和基于资源的策略都属于权限策略，并且将一起评估。对于仅将权限策略应用于的请求，Amazon 将先检查 Deny 的所有策略。如果存在一个此类策略，则该请求将被拒绝。然后，Amazon 检查每个 Allow。如果至少一个策略语句允许请求中的操作，则将允许此请求。无论 Allow 是在基于身份的策略中还是基于资源的策略中，都无关紧要。

Important

此逻辑仅当请求在单个 Amazon 账户中发出时适用。对于从一个账户向另一个账户发出的请求，Account A 中的请求者必须具有基于身份的策略，该策略允许其向 Account B 中的资源发出请求。此外，Account B 中的基于资源的策略必须允许 Account A 中的请求者访问资源。这两个账户中都必须有允许操作的策略，否则请求失败。有关将基于资源的策略用于跨账户访问的更多信息，请参阅 [IAM 角色与基于资源的策略有何不同 \(p. 274\)](#)。

具有特定权限的用户可能会请求也附加了权限策略的资源。在这种情况下，在确定是否授予资源的访问权限时，Amazon 将评估这两组权限。有关如何评估策略的信息，请参阅 [策略评估逻辑 \(p. 972\)](#)。

Note

Amazon S3 支持基于身份的策略和基于资源的策略（称为存储桶策略）。此外，Amazon S3 还支持称为访问控制列表 (ACL) 的权限机制，该机制独立于 IAM 策略和权限。您可以将 IAM 策略与 Amazon S3 ACL 结合使用。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的 [访问控制](#)。

使用策略控制对 Amazon 资源的访问。

您可以使用策略控制对 IAM 或所有 Amazon 中的资源的访问。

要使用 [标签 \(p. 355\)](#) 在 Amazon 中控制访问，您必须了解 Amazon 如何授予访问权限。Amazon 由资源集合组成。IAM 用户是一种资源。Amazon S3 存储桶是一种资源。在使用 Amazon API、Amazon CLI 或 Amazon Web Services Management Console 执行操作（如创建用户）时，您将为该操作发送一个请求。您的请求指定操作、资源、主体实体（用户或角色）、主体账户以及所需的任何请求信息。所有这些信息提供了上下文。

Amazon 然后，检查是否对您（主体）进行身份验证（登录）和授权（具有权限），以便对指定的资源执行指定的操作。在授权期间，Amazon 检查应用于请求上下文的所有策略。大多数策略作为 [JSON 文档 \(p. 360\)](#) 存储在 Amazon 中，并指定主体实体的权限。有关策略类型和用法的更多信息，请参阅 [IAM 中的策略和权限 \(p. 355\)](#)。

只有在策略允许请求的每个部分时，Amazon 才会授权该请求。要查看该过程的图表，请参阅 [了解 IAM 的工作方式 \(p. 3\)](#)。有关 Amazon 如何确定是否允许请求的详细信息，请参阅 [策略评估逻辑 \(p. 972\)](#)。

在创建 IAM policy 时，您可以控制对以下内容的访问：

- [主体 \(p. 382\)](#) - 控制允许发出请求的人员（[主体 \(p. 5\)](#)）执行哪些操作。
- [IAM 身份 \(p. 383\)](#) - 控制可访问哪些 IAM 身份（用户组、用户和角色）以及如何进行访问。
- [IAM policy \(p. 384\)](#) - 控制哪些用户可以创建、编辑和删除客户管理型策略，以及哪些用户可以附加和分离所有托管策略。
- [Amazon 资源 \(p. 387\)](#) - 控制哪些用户有权使用基于身份的策略或基于资源的策略访问资源。
- [Amazon 账户 \(p. 387\)](#) - 控制是否仅允许特定账户的成员发出请求。

通过使用策略，您可以指定哪些用户有权访问 Amazon 资源，以及他们可以对这些资源执行哪些操作。每个 IAM 用户在一开始都没有许可。换言之，在默认状态下，用户什么都不能做，甚至不能查看自己的访问密钥。要为用户授予执行某些操作的权限，您可以为用户添加权限（即，将策略附加到用户）。或者，您可以将用户添加到具有所需权限的用户组中。

例如，您可能授予用户列出自己的访问密钥的许可。您可能扩展该许可，并让每位用户创建、更新和删除自己的密钥。

当您授予一个用户组许可时，用户组内的全部用户都会获得那些许可。例如，您可以向管理员用户组授予对任意 Amazon 账户资源执行任意 IAM 操作的权限。另一个示例：您可以授予 Managers 用户组描述 Amazon 账户的 Amazon EC2 实例的权限。

有关如何向用户、用户组和角色委派基本权限的信息，请参阅 [访问 IAM 资源所需的权限 \(p. 508\)](#)。有关说明基本权限的更多策略示例，请参阅 [管理 IAM 资源的策略示例 \(p. 510\)](#)。

控制 主体进行的访问

您可以使用策略控制允许发出请求的人员（主体）执行哪些操作。为此，您必须将基于身份的策略附加到此人的身份（用户、用户组或角色）。您还可以使用[权限边界 \(p. 370\)](#)以设置实体（用户或角色）可以拥有的最大权限。

例如，假定您希望用户 Zhang Wei 可以完全访问 CloudWatch、Amazon DynamoDB、Amazon EC2 和 Amazon S3。您可以创建两个不同的策略，以便以后其他用户需要一组权限时为其分配其中的一个策略。或者，您可以将这两个权限放在单个策略中，然后将该策略附加到名为 Zhang Wei 的 IAM 用户。您还可以将一个策略附加到 Zhang 所属的用户组，或者附加到 Zhang 可以担任的角色。因此，在 Zhang 查看 S3 存储桶内容时，将允许他的请求。如果他尝试创建新的 IAM 用户，则会拒绝他的请求，因为他没有权限。

您可以对 Zhang 使用权限边界，以确保他无法访问 **DOC-EXAMPLE-BUCKET1** S3 存储桶。为此，请确定您希望 Zhang 具有的最大权限。在这种情况下，您使用权限策略控制他执行的操作。在这里，您只希望他无法访问机密存储桶。因此，您使用以下策略来定义 Zhang 的边界，以允许对 Amazon S3 和一些其他服务执行所有 Amazon 操作，但拒绝访问 **DOC-EXAMPLE-BUCKET1** S3 存储桶。由于权限边界不允许任何 IAM 操作，它可防止 Zhang 删除他的（或任何人的）边界。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PermissionsBoundarySomeServices",
      "Effect": "Allow",
      "Action": [
        "cloudwatch:*",
        "dynamodb:*",
        "ec2:*",
        "s3:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "PermissionsBoundaryNoConfidentialBucket",
      "Effect": "Deny",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3::DOC-EXAMPLE-BUCKET1",
        "arn:aws:s3::DOC-EXAMPLE-BUCKET1/*"
      ]
    }
  ]
}
```

当您分配此类策略作为用户的权限边界时，请记住，它不授予任何权限。它设置基于身份的策略可以授予 IAM 实体的最大权限。有关权限边界的更多信息，请参阅[IAM 实体的权限边界 \(p. 370\)](#)。

有关上述过程的详细信息，请参阅以下资源：

- 要了解创建可附加到主体的 IAM policy 的更多信息，请参阅 [创建 IAM policy \(p. 437\)](#)。
- 要了解如何将 IAM policy 附加到主体，请参阅 [添加和删除 IAM 身份权限 \(p. 451\)](#)。
- 要查看授予 EC2 完全访问权限的示例策略，请参阅[Amazon EC2：允许以编程方式和使用控制台在特定区域进行不受限制的 EC2 访问 \(p. 412\)](#)。
- 要允许对 S3 存储桶进行只读访问，请使用以下示例策略的前两个语句：[Amazon S3：允许以编程方式和在控制台对 S3 存储桶中的对象进行读写访问 \(p. 436\)](#)。
- 要查看允许用户设置或轮换其凭证（如控制台密码、编程访问密钥和 MFA 设备）的示例策略，请参阅[Amazon：允许经过 MFA 身份验证的 IAM 用户在“My Security Credentials”（我的安全凭证）页面上管理自己的凭证 \(p. 395\)](#)。

控制对身份的访问

您可以通过用户组创建附加到所有用户的 IAM policy，以使用该策略控制您的用户可对某个身份执行哪些操作。为此，请创建一个策略以限制可对某个身份执行哪些操作，以及哪些用户可以访问该身份。

例如，您可以创建一个名为 AllUsers 的组，然后将该用户组附加到所有用户。在创建该用户组时，您可以为所有用户授予访问权限以轮换其凭证，如前一节中所述。然后，您可以创建一个策略，以便拒绝访问以禁止更改该用户组，除非在策略条件中包含用户名。但是，该策略部分仅拒绝列出的用户以外的任何人进行访问。您还必须包含相应的权限，以允许该用户组中的所有用户执行所有用户组管理操作。最后，您将该策略附加到该用户组，以便将其应用于所有用户。因此，在策略中未指定的用户尝试更改该用户组时，将拒绝请求。

使用可视化编辑器创建该策略

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
 2. 在左侧的导航窗格中，选择策略。
- 如果这是您首次选择 Policies，则会显示 Welcome to Managed Policies 页面。选择开始使用。
3. 选择 Create policy (创建策略)。
 4. 在可视化编辑器选项卡上，选择选择服务以开始使用。然后，选择 IAM。
 5. 选择 Select actions (选择操作)，然后在搜索框中键入 **group**。可视化编辑器将显示包含 group 一词的所有 IAM 操作。选中所有复选框。
 6. 选择资源以指定您的策略的资源。根据选择的操作，将会看到 group、group-path 和 user 资源类型。

- group - 选择 Add ARN (添加 ARN)。对于资源，选中任意旁边的复选框。对于 Group Name With Path (具有路径的组名称)，键入用户组名称 **AllUsers**。然后，选择 Add (添加)。
- group-path - 选中 Any (任意) 旁边的复选框。
- user - 选中 Any (任意) 旁边的复选框。

您选择的操作之一 (ListGroups) 不支持使用特定的资源。您不需要为该操作选择所有资源。在 JSON 选项卡上保存或查看您的策略时，您可以看到 IAM 自动创建新的权限块，以便为所有资源授予该操作权限。

7. 要添加其他权限块，请选择添加额外的权限。
8. 选择 Choose a service (选择服务)，然后选择 IAM。
9. 选择选择操作，然后选择切换到拒绝权限。在执行该操作时，将使用整个块拒绝权限。
10. 在搜索框中，键入 **group**。可视化编辑器将显示包含 group 一词的所有 IAM 操作。选中以下操作旁边的复选框：
 - CreateGroup
 - DeleteGroup
 - RemoveUserFromGroup
 - AttachGroupPolicy
 - DeleteGroupPolicy
 - DetachGroupPolicy
 - PutGroupPolicy
 - UpdateGroup
11. 选择资源以指定您的策略的资源。根据选择的操作，将会看到 group 资源类型。选择 Add ARN (添加 ARN)。对于资源，选中任意旁边的复选框。对于 Group Name With Path (具有路径的组名称)，键入用户组名称 **AllUsers**。然后，选择 Add (添加)。
12. 选择指定请求条件 (可选)，然后选择添加条件。使用以下值填写表单：

- Key - 选择 `aws:username`
- Qualifier (限定符) - 选择 Default (默认值)
- Operator (运算符) - 选择 StringNotEquals
- Value (值) - 键入 `srodriguez`，然后选择 Add another condition value (添加另一个条件值)。键入 `mjackson`，然后选择 Add another condition value (添加另一个条件值)。键入 `adesai`，然后选择 Add (添加)。

在列表中不包含进行调用的用户时，该条件确保拒绝访问指定的用户组管理操作。由于这会显式拒绝权限，它将覆盖前面允许这些用户调用这些操作的块。不会拒绝列表中的用户进行访问，并在第一个权限块中为他们授予权限，因此，他们可以完全管理该用户组。

13. 完成后，选择 Review policy (查看策略)。

Note

您可以随时在可视化编辑器和 JSON 选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy (查看策略)，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅[调整策略结构 \(p. 871\)](#)。

14. 在 Review policy (查看策略)页面上，对于 Name (名称)，键入 `LimitAllUserGroupManagement`。对于描述，请键入 `Allows all users read-only access to a specific user group, and allows only specific users access to make changes to the user group`。查看策略摘要以确保您授予了所需的权限。然后，选择创建策略以保存新策略。
15. 将策略附加到您的用户组。有关更多信息，请参阅[添加和删除 IAM 身份权限 \(p. 451\)](#)。

或者，您也可以使用该示例 JSON 策略文档创建相同的策略。要查看该 JSON 策略，请参阅[IAM：允许特定的 IAM 用户以编程方式和在控制台中管理组 \(p. 423\)](#)。有关使用 JSON 文档创建策略的详细说明，请参阅[the section called “在“JSON”选项卡上创建策略” \(p. 438\)](#)。

控制对策略的访问

您可以控制您的用户如何应用 Amazon 托管策略。为此，请将该策略附加到您的所有用户。理想情况下，您可以使用用户组执行该操作。

例如，您可能会创建一个策略允许用户仅将 `IAMUserChangePassword` 和 `PowerUserAccess` Amazon 托管策略附加到新的 IAM 用户、用户组或角色。

对于客户托管策略，您可以控制哪些用户可以创建、更新和删除这些策略。您可以控制哪些用户可以将策略附加到主体实体（用户组、用户和角色）以及将策略从主体实体中分离。还可以控制用户可以对哪些实体附加或分离哪些策略。

例如，您可以为账户管理员授予权限以创建、更新和删除策略。然后，您可以为小组负责人或其他有限管理员授予权限，以便将这些策略附加到该有限管理员管理的主体实体以及将这些策略从这些主体实体中分离。

有关更多信息，请参阅以下资源：

- 要了解创建可附加到主体的 IAM policy 的更多信息，请参阅[创建 IAM policy \(p. 437\)](#)。
- 要了解如何将 IAM policy 附加到主体，请参阅[添加和删除 IAM 身份权限 \(p. 451\)](#)。
- 要查看限制使用托管策略的示例策略，请参阅[IAM：对可应用于 IAM 用户、组或角色的托管策略加以限制 \(p. 427\)](#)。

控制创建、更新和删除客户托管策略的权限

您可以使用 [IAM policy \(p. 355\)](#) 控制谁有权在您的 Amazon 账户中创建、更新和删除客户托管策略。以下列表包含与创建、更新和删除策略或策略版本直接相关的 API 操作：

- [CreatePolicy](#)
- [CreatePolicyVersion](#)
- [DeletePolicy](#)
- [DeletePolicyVersion](#)
- [SetDefaultPolicyVersion](#)

上述列表中的 API 操作对应于可使用 IAM policy 允许或拒绝的操作，即，您可以授予的权限。

考虑以下示例策略。它允许用户创建、更新（即创建新的策略版本）、删除 Amazon 账户中的所有客户托管策略以及设置这些策略的默认版本。该示例策略还允许该用户列出策略并获取策略。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略” \(p. 438\)](#)。

Example 允许创建、更新、删除、列出、获取所有策略以及设置这些策略的默认版本的示例策略

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:CreatePolicy",
      "iam:CreatePolicyVersion",
      "iam:DeletePolicy",
      "iam:DeletePolicyVersion",
      "iam:GetPolicy",
      "iam:GetPolicyVersion",
      "iam:ListPolicies",
      "iam:ListPolicyVersions",
      "iam:SetDefaultPolicyVersion"
    ],
    "Resource": "*"
  }
}
```

您可以创建限制使用这些 API 操作的策略以仅影响指定的托管策略。例如，您可能希望允许用户设置默认版本和删除策略版本，但是仅允许针对特定客户托管策略执行这些操作。通过在授予这些权限的策略的 `Resource` 元素中指定策略 ARN，可以实现此目的。

下例所示的策略允许用户删除策略版本和设置默认版本。但是，仅包含路径 `/TEAM-A/` 的客户托管策略允许这些操作。客户托管策略 ARN 在策略的 `Resource` 元素中指定。（在此示例中，ARN 包含一个路径和一个通配符，因而与包含路径 `/TEAM-A/` 的所有客户托管策略匹配）。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略” \(p. 438\)](#)。

有关在客户托管策略名称中使用路径的更多信息，请参阅[易记名称和路径 \(p. 896\)](#)。

Example 仅允许针对特定策略删除策略版本和设置默认版本的示例策略

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:DeletePolicyVersion",
      "iam:SetDefaultPolicyVersion"
    ],
    "Resource": "arn:aws:iam::account-id:policy/TEAM-A/*"
  }
}
```

控制附加和分离托管策略的权限

您还可以使用 IAM policy 仅允许用户使用特定的托管策略。实际上，就是可以控制允许用户向其他主体实体授予的权限。

以下列表显示与将托管策略附加到主体实体以及从中分离直接相关的 API 操作：

- [AttachGroupPolicy](#)
- [AttachRolePolicy](#)
- [AttachUserPolicy](#)
- [DetachGroupPolicy](#)
- [DetachRolePolicy](#)
- [DetachUserPolicy](#)

您可以创建限制使用这些 API 操作的策略以仅影响指定的特定托管策略和/或主体实体。例如，您可能希望允许用户附加托管策略，但是只能附加您指定的托管策略。或者，您可能希望允许用户附加托管策略，但是只能附加到您指定的主体实体。

以下示例策略仅允许用户将托管策略附加到包含路径 /TEAM-A/ 的用户组和角色。用户组和角色 ARN 是在策略的 `Resource` 元素中指定的。(在此示例中，ARN 包含一个路径和一个通配符，因而与包含路径 /TEAM-A/ 的所有用户组和角色匹配。) 要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略” \(p. 438\)](#)。

Example 仅允许将托管策略附加到特定用户组或角色的示例策略

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:AttachGroupPolicy",
      "iam:AttachRolePolicy"
    ],
    "Resource": [
      "arn:aws:iam:::group/TEAM-A/*",
      "arn:aws:iam:::role/TEAM-A/*"
    ]
  }
}
```

您可以进一步限制前面示例中的操作，使之仅影响特定的策略。即，通过向策略添加条件，可以控制允许用户附加到其他主体实体的权限。

以下示例中的条件可确保仅当附加的策略与指定策略之一匹配时才允许 `AttachGroupPolicy` 和 `AttachRolePolicy` 权限。该条件使用 `iam:PolicyARN` [条件键 \(p. 948\)](#) 确定允许附加的策略。以下示例策略扩展前一示例。它仅允许用户将包含路径 /TEAM-A/ 的托管策略附加到包含路径 /TEAM-A/ 的用户组和角色。要了解如何使用该示例 JSON 策略文档创建策略，请参阅[the section called “在“JSON”选项卡上创建策略” \(p. 438\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:AttachGroupPolicy",
      "iam:AttachRolePolicy"
    ],
    "Resource": [
```

```
    "arn:aws:iam:::group/TEAM-A/*",
    "arn:aws:iam:::role/TEAM-A/*"
  ],
  "Condition": { "ArnLike":
    { "iam:PolicyARN": "arn:aws:iam:::policy/TEAM-A/*" }
  }
}
```

该策略使用 ArnLike 条件运算符，因为 ARN 包含通配符。对于特定的 ARN，请使用 ArnEquals 条件运算符。有关 ArnLike 和 ArnEquals 的更多信息，请参阅[Amazon Resource Name \(ARN\) 条件运算符 \(p. 956\)](#)策略元素参考 的条件类型 部分中的。

例如，您可以限制使用操作以仅涉及指定的托管策略。通过在授予这些权限的策略的 Condition 元素中指定策略 ARN，可以实现此目的。例如，指定客户托管策略的 ARN：

```
"Condition": { "ArnEquals":
  { "iam:PolicyARN": "arn:aws:iam::123456789012:policy/POLICY-NAME" }
}
```

您还可以在策略的 Condition 元素中指定 Amazon 托管策略的 ARN。Amazon 托管策略的 ARN 在策略 ARN 中使用特殊别名 aws 而不是账户 ID，如此示例所示：

```
"Condition": { "ArnEquals":
  { "iam:PolicyARN": "arn:aws:iam::aws:policy/AmazonEC2FullAccess" }
}
```

控制对资源的访问

您可以使用基于身份的策略或基于资源的策略控制对资源的访问。在基于身份的策略中，您将策略附加到一个身份并指定该身份可以访问哪些资源。在基于资源的策略中，您将策略附加到要控制的资源。在该策略中，您指定哪些主体可以访问该资源。有关这两种类型的策略的更多信息，请参阅[基于身份的策略和基于资源的策略 \(p. 379\)](#)。

有关更多信息，请参阅以下资源：

- 要了解创建可附加到主体的 IAM policy 的更多信息，请参阅 [创建 IAM policy \(p. 437\)](#)。
- 要了解如何将 IAM policy 附加到主体，请参阅 [添加和删除 IAM 身份权限 \(p. 451\)](#)。
- Amazon S3 支持在存储桶上使用基于资源的策略。有关更多信息，请参阅[存储桶策略示例](#)。

资源创建者不会自动拥有许可

如果使用 Amazon Web Services 账户根用户凭证登录，您有权对属于该账户的资源执行任何操作。但是，对 IAM 用户来说并非如此。IAM 用户可以获得创建资源的权限，但即使对于该资源，该用户权限也仅限于明确授予的内容。这意味着，您不会仅仅因为创建资源（如 IAM 角色）而自动具有编辑或删除该角色的权限。此外，账户所有者或有权管理您的权限的其他用户可以随时撤消您的权限。

控制对特定账户中的主体的访问

您可以直接向自己账户中的 IAM 用户授予对您的资源的访问权限。如果其他账户中的用户需要访问您的资源，您可以创建 IAM 角色。角色是包含权限但不与特定用户关联的实体。然后，其他账户中的用户可以担任该角色，并根据您为该角色分配的权限访问资源。有关更多信息，请参阅[在您拥有的另一个 Amazon 账户中向 IAM 用户提供访问权限 \(p. 161\)](#)。

Note

有些服务支持基于资源的策略，如[基于身份的策略和基于资源的策略 \(p. 379\)](#)中所述（例如 Amazon S3、Amazon SNS 和 Amazon SQS）。对于这些服务，使用角色的替代方法是将策略

附加到要共享的资源（存储桶、主题或队列）。基于资源的策略可以指定具有访问资源权限的 Amazon 账户。

使用标签控制对 IAM 用户和角色的访问以及他们进行的访问

可以使用以下部分中的信息控制谁可以访问您的 IAM 用户和角色以及您的用户和角色可以访问哪些资源。有关控制对其他 Amazon 资源（包括其他 IAM 资源）的访问的更多一般信息和示例，请参阅 [使用标签控制对 Amazon 资源的访问 \(p. 389\)](#)。

可以将标签附加到 IAM 资源，在请求中传递标签，或者将标签附加到发出请求的主体。IAM 用户或角色可以同时是资源和主体。例如，您可以编写一个策略以允许用户列出用户的组。只有在发出请求的用户（主体）具有与他们尝试查看的用户相同的 `project=blue` 标签时，才允许执行该操作。在该示例中，用户可以查看任何用户（包括他们自己）的组成员资格，只要他们处理的是同一个项目。

要基于标签控制访问，您需要在策略的 [条件元素 \(p. 948\)](#) 中提供标签信息。在创建 IAM policy 时，您可以使用 IAM 标签和关联的标签条件键控制对任何以下内容的访问：

- [Resource \(p. 390\)](#)（资源）- 根据标签控制对用户或角色资源的访问。为此，请使用 `aws:ResourceTag/key-name` 条件键指定必须将哪个标签键值对附加到资源。有关更多信息，请参阅 [控制对 Amazon 资源的访问 \(p. 390\)](#)。
- [请求 \(p. 390\)](#) - 控制可以在 IAM 请求中传递什么标签。为此，请使用 `aws:RequestTag/key-name` 条件键来指定可在 IAM 用户或角色中添加、更改或删除的标签。对于 IAM 资源和其他 Amazon 资源，将按相同的方式使用该键。有关更多信息，请参阅 [在 Amazon 请求期间控制访问 \(p. 390\)](#)。
- [主体 \(p. 388\)](#) - 根据附加到发出请求的人员（主体）的 IAM 用户或角色的标签，控制允许该人员执行哪些操作。为此，请使用 `aws:PrincipalTag/key-name` 条件键指定在允许请求之前必须将哪些标签附加到 IAM 用户或角色。
- [授权过程的任何部分 \(p. 389\)](#) - 使用 `aws:TagKeys` 条件键来控制是否可以对资源、请求或主体使用特定的标签键。在这种情况下，键值无关紧要。对于 IAM 资源和其他 Amazon 资源，该键的行为是类似的。不过，在 IAM 中标记用户时，这还会控制主体是否可以向任何服务发出请求。有关更多信息，请参阅 [根据标签键控制访问 \(p. 391\)](#)。

您可以使用可视化编辑器或 JSON 创建 IAM policy，也可以导入现有的托管策略以创建该策略。有关详细信息，请参阅 [创建 IAM policy \(p. 437\)](#)。

控制 IAM 主体进行的访问

您可以根据附加到主体身份的标签来控制允许该人员执行哪些操作。

此示例显示了您可以如何创建基于身份的策略，以允许此账户中的任何用户查看任何用户（包括他们自己）的组成员资格，只要他们处理的是同一个项目。只有在用户的资源标签和主体的标签具有相同的标签键 `project` 值时，才允许执行该操作。要使用此策略，请将示例策略中的 `#####` 替换为您自己的信息。然后，按照 [创建策略 \(p. 437\)](#) 或 [编辑策略 \(p. 461\)](#) 中的说明操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "iam:ListGroupsForUser",
      "Resource": "arn:aws:iam::111222333444:user/*",
      "Condition": {
        "StringEquals": {"aws:ResourceTag/project": "${aws:PrincipalTag/project"}
      }
    }
  ]
}
```

```
}
```

根据标签键控制访问

您可以在 IAM policy 中使用标签来控制特定的标签键是否可用于资源、请求中或由主体使用。

此示例说明如何创建基于身份的策略，以允许仅从用户删除带有 `temporary` 密钥的标签。要使用此策略，请将示例策略中的 `#####` 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:UntagUser",
    "Resource": "*",
    "Condition": {"ForAllValues:StringEquals": {"aws:TagKeys": ["temporary"]}}
  ]
}
```

使用标签控制对 Amazon 资源的访问

您可以使用标签以控制对支持标记的 Amazon 资源（包括 IAM 资源）的访问。您可以标记 IAM 用户和角色以控制他们可以访问哪些内容。要了解如何标记 IAM 用户和角色，请参阅[标记 IAM 资源 \(p. 276\)](#)。此外，您还可以控制对以下 IAM 资源的访问：客户托管策略、IAM 身份提供程序、实例配置文件、服务器证书和虚拟 MFA 设备。对于创建策略，以允许具有主体标签的 IAM 角色访问具有匹配标签的资源并测试该策略，有关教程，请参阅[IAM 教程：根据标签定义访问 Amazon 资源的权限 \(p. 42\)](#)。可以使用以下部分中的信息以控制对其他 Amazon 资源（包括 IAM 资源）的访问，而无需标记 IAM 用户或角色。

在使用标签控制对 Amazon 资源的访问之前，您必须了解 Amazon 如何授予访问权限。Amazon 由一组资源组成。Amazon EC2 实例是一种资源。Amazon S3 存储桶是一种资源。您可以使用 Amazon API、Amazon CLI 或 Amazon Web Services Management Console 执行操作，例如，在 Amazon S3 中创建存储桶。在执行操作时，您将发送该操作的请求。您的请求指定操作、资源、主体实体（用户或角色）、主体账户以及所需的任何请求信息。所有这些信息提供了上下文。

Amazon 然后，检查是否对您（主体实体）进行身份验证（登录）和授权（具有权限），以便对指定的资源执行指定的操作。在授权期间，Amazon 检查应用于请求上下文的所有策略。大多数策略作为 [JSON 文档 \(p. 360\)](#) 存储在 Amazon 中，并指定主体实体的权限。有关策略类型和用法的更多信息，请参阅[IAM 中的策略和权限 \(p. 355\)](#)。

只有在策略允许请求的每个部分时，Amazon 才会授权该请求。要查看示意图和详细了解 IAM 基础设施，请参阅[了解 IAM 的工作方式 \(p. 3\)](#)。有关 IAM 如何确定是否允许请求的详细信息，请参阅[策略评估逻辑 \(p. 972\)](#)。

标签可能使此过程变得复杂，因为标签可以附加到资源，也可以从请求传入支持标签的服务。要基于标签控制访问，您需要在策略的[条件元素 \(p. 948\)](#)中提供标签信息。要了解 Amazon 服务是否支持使用标签控制访问，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)并查找在 Authorization based on tags（根据标签进行授权）列中具有 Yes（是）的服务。选择服务名称以查看该服务的授权和访问控制文档。

然后，您可以创建一个 IAM policy，以根据资源的标签允许或拒绝访问资源。在该策略中，您可以使用标签条件键以控制对任何以下内容的访问：

- [Resource \(p. 390\)](#)（资源）- 基于 Amazon 服务资源上的标签控制对这些资源的访问。为此，请使用 `ResourceTag/key-name` 条件键根据附加到资源的标签确定是否允许访问资源。
- [Request \(p. 390\)](#)（请求）- 控制可以在请求中传递哪些标签。为此，请使用 `aws:RequestTag/key-name` 条件键指定可以在请求中传递哪些标签键值对以标记或取消标记 Amazon 资源。
- [授权过程的任何部分 \(p. 391\)](#) - 使用 `aws:TagKeys` 条件键以控制是否可以在资源或请求中使用特定的标签键。

您可以通过可视化方式创建 IAM policy，也可以使用 JSON，或通过导入现有托管策略来创建。有关详细信息，请参阅[创建 IAM policy \(p. 437\)](#)。

Note

如果用户有权使用创建资源的操作，则某些服务允许用户在创建资源时指定标签。

控制对 Amazon 资源的访问

您可以使用 IAM policy 中的条件根据 Amazon 资源上的标签控制对该资源的访问。您可以使用全局 `aws:ResourceTag/tag-key` 条件键或服务特定的键执行该操作。某些服务仅支持此键的服务特定版本，而不支持全局版本。

Note

不要在策略中将 `ResourceTag` 条件键与 `iam:PassRole` 操作一起使用。您无法在 IAM 角色上使用标签以控制可以传递该角色的用户的访问权限。有关将角色传递给服务所需的权限的更多信息，请参阅[向用户授予权限以将角色传递给 Amazon 服务 \(p. 240\)](#)。

此示例说明如何创建允许启动或停止 Amazon EC2 实例的基于身份的策略。只有在实例标签 `Owner` 具有该用户的用户名值时，才允许执行这些操作。此策略定义了程序访问和控制台访问的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:DescribeInstances",
      "Resource": "*"
    }
  ]
}
```

您可以将该策略附加到您账户中的 IAM 用户。如果名为 `richard-roe` 的用户尝试启动 Amazon EC2 实例，则实例必须标记为 `Owner=richard-roe` 或 `owner=richard-roe`。否则，他将被拒绝访问。标签键 `Owner` 同时匹配 `owner` 和 `Owner`，因为条件键名称不区分大小写。有关更多信息，请参阅[IAM JSON 策略元素：Condition \(p. 948\)](#)。

在 Amazon 请求期间控制访问

您可以在 IAM policy 中使用条件以控制可以在标记 Amazon 资源的请求中传递哪些标签键值对。

此示例说明如何创建允许使用 Amazon EC2 `CreateTags` 操作将标签附加到实例的基于身份的策略。只有在标签包含 `environment` 键和 `preprod` 或 `production` 值时，才能附加标签。如果需要，您可以将 `ForAllValues` 修饰符与 `aws:TagKeys` 条件键一起使用，以指示仅允许在请求中使用 `environment` 键。这将阻止用户包括其他键，例如意外使用 `Environment` 而不是 `environment`。

```
{
  "Version": "2012-10-17",
  "Statement": {
```



```
{
  "Effect": "Allow",
  "Action": "ec2:CreateTags",
  "Resource": "arn:aws:ec2:*:*:instance/*",
  "Condition": {
    "StringEquals": {
      "aws:RequestTag/environment": [
        "preprod",
        "production"
      ]
    },
    "ForAllValues:StringEquals": {"aws:TagKeys": "environment"}
  }
}
```

根据标签键控制访问

您可以在 IAM policy 中使用条件来控制是否可以在资源或请求中使用特定标签键。

作为最佳实践，当您使用策略来通过标签控制访问时，应该使用 [aws:TagKeys 条件键 \(p. 1011\)](#)。支持标签的 Amazon 服务可能允许您创建多个只有大小写不同的标签键名称，例如使用 `stack=production` 和 `Stack=test` 标记 Amazon EC2 实例。在策略条件中，键名称不区分大小写。这意味着，如果您在策略的条件元素中指定 `"aws:ResourceTag/TagKey1": "Value1"`，则条件将匹配名为 `TagKey1` 或 `tagkey1` 的资源标签键，但不会同时匹配两者。要防止只有键的大小写形式不同的重复标签，请使用 `aws:TagKeys` 条件来定义用户可以应用的标签键，或使用带有 Amazon Organizations 的标签策略。有关更多信息，请参阅《Organizations 用户指南》中的[标签策略](#)。

此示例说明了如何创建允许创建 Secrets Manager 密钥及将其贴标签的基于身份的策略，但仅限于标签密钥 `environment` 或 `cost-center`。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:CreateSecret",
      "secretsmanager:TagResource"
    ],
    "Resource": "*",
    "Condition": {
      "ForAllValues:StringEquals": {
        "aws:TagKeys": [
          "environment",
          "cost-center"
        ]
      }
    }
  }
}
```

IAM 基于身份的策略示例

[策略 \(p. 355\)](#) 是 Amazon 中的对象；在与身份或资源相关联时，策略定义它们的权限。在某个 IAM 主体（用户或角色）发出请求时，Amazon 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略作为附加到 IAM 身份（用户、用户组或角色）的 JSON 文档存储在 Amazon 中。基于身份的策略包括 Amazon 托管策略、客户托管策略和内联策略。要了解如何使用这些示例 JSON 策略文档创建 IAM policy，请参阅 [the section called “在“JSON”选项卡上创建策略” \(p. 438\)](#)

默认情况下，所有请求都会被拒绝，因此必须提供相关权限供身份访问服务、操作和资源。如果还需要允许访问以在 IAM 控制台中完成指定的操作，则需要提供额外的权限。

下面的策略库可以帮助您为自己的 IAM 身份定义权限。在找到所需的策略后，请选择查看该策略以查看该策略的 JSON 版本。您可以将该 JSON 策略文档用作自己策略的模板。

Note

如果您愿意提交策略供本参考指南采用，请使用该页面底部的 Feedback 按钮。

示例策略：Amazon

- 允许在特定日期范围内进行访问。([查看该策略 \(p. 394\)](#)。)
- 允许启用和禁用 Amazon 区域。([查看该策略 \(p. 394\)](#)。)
- 允许经过 MFA 身份验证的用户在 My Security Credentials (我的安全凭证) 页面上管理自己的凭证。([查看该策略 \(p. 395\)](#)。)
- 允许在特定日期范围内使用 MFA 进行特定访问。([查看该策略 \(p. 398\)](#)。)
- 允许用户在 My Security Credentials (我的安全凭证) 页面上管理自己的凭证。([查看该策略 \(p. 398\)](#)。)
- 允许用户在 My Security Credentials (我的安全凭证) 页面上管理自己的 MFA 设备。([查看该策略 \(p. 400\)](#)。)
- 允许用户在 My Security Credentials (我的安全凭证) 页面上管理自己的密码。([查看该策略 \(p. 402\)](#)。)
- 允许用户在 My Security Credentials (我的安全凭证) 页面上管理自己的密码、访问密钥和 SSH 公有密钥。([查看该策略 \(p. 402\)](#)。)
- 根据请求的区域拒绝对 Amazon 的访问。([查看该策略 \(p. 404\)](#)。)
- 基于源 IP 拒绝对 Amazon 的访问。([查看该策略 \(p. 405\)](#)。)

示例策略：Amazon Data Pipeline

- 拒绝访问不是用户创建的管道 ([查看该策略 \(p. 405\)](#)。)

示例策略：Amazon DynamoDB

- 允许访问特定的 Amazon DynamoDB 表 ([查看该策略 \(p. 406\)](#)。)
- 允许访问特定的 Amazon DynamoDB 属性 ([查看该策略 \(p. 407\)](#)。)
- 允许基于 Amazon Cognito ID 对 Amazon DynamoDB 进行项目级访问 ([查看该策略 \(p. 407\)](#)。)

示例策略：Amazon EC2

- 允许 Amazon EC2 实例附加或分离卷 ([查看该策略 \(p. 408\)](#)。)
- 允许根据标签为 Amazon EC2 实例附加或分离 Amazon EBS 卷 ([查看该策略 \(p. 409\)](#)。)
- 允许以编程方式和使用控制台在特定子网中启动 Amazon EC2 实例 ([查看该策略 \(p. 409\)](#)。)
- 允许以编程方式和在控制台中管理与特定 VPC 关联的 Amazon EC2 安全组 ([查看该策略 \(p. 410\)](#)。)
- 允许以编程方式和在控制台中启动或停止用户标记的 Amazon EC2 实例 ([查看该策略 \(p. 411\)](#)。)
- 允许以编程方式和在控制台中根据资源和主体标签启动或停止 Amazon EC2 实例 ([查看该策略 \(p. 411\)](#)。)
- 在资源和主体标签匹配时，允许启动或停止 Amazon EC2 实例 ([查看该策略 \(p. 412\)](#)。)
- 允许以编程方式和使用控制台在特定区域进行不受限制的 Amazon EC2 访问 ([查看该策略 \(p. 412\)](#)。)
- 允许以编程方式和在控制台中启动或停止特定的 Amazon EC2 实例和修改特定的安全组 ([查看该策略 \(p. 413\)](#))
- 拒绝在没有 MFA 的情况下访问特定的 Amazon EC2 操作 ([查看该策略 \(p. 413\)](#)。)
- 将要终止的 Amazon EC2 实例限定为特定的 IP 地址范围 ([查看该策略 \(p. 414\)](#))

示例策略：Amazon Identity and Access Management (IAM)

- 允许访问策略模拟器 API ([查看该策略 \(p. 415\)](#)。)
- 允许访问策略模拟器控制台 ([查看该策略 \(p. 415\)](#)。)
- 允许以编程方式和在控制台中担任具有特定标签的任何角色 ([查看该策略 \(p. 416\)](#)。)
- 允许和拒绝以编程方式和在控制台中访问多个服务 ([查看该策略 \(p. 416\)](#)。)
- 允许以编程方式和在控制台中为具有不同的特定标签的 IAM 用户添加特定的标签 ([查看该策略 \(p. 417\)](#)。)
- 允许以编程方式和在控制台中为任何 IAM 用户或角色添加特定的标签 ([查看该策略 \(p. 418\)](#)。)
- 允许创建仅具有特定标签的新用户 ([查看该策略 \(p. 419\)](#)。)
- 允许生成和检索 IAM 凭证报告 ([查看该策略 \(p. 420\)](#)。)
- 允许以编程方式和在控制台中管理组的成员资格 ([查看该策略 \(p. 420\)](#)。)
- 允许管理特定的标签 ([查看该策略 \(p. 421\)](#)。)
- 允许将 IAM 角色传递到特定的服务 ([查看该策略 \(p. 421\)](#)。)
- 允许对 IAM 控制台进行只读访问而不生成报告 ([查看该策略 \(p. 422\)](#)。)
- 允许对 IAM 控制台进行只读访问 ([查看该策略 \(p. 422\)](#)。)
- 允许特定用户以编程方式和在控制台中管理组 ([查看该策略 \(p. 423\)](#)。)
- 允许以编程方式和在控制台中设置账户密码要求 ([查看该策略 \(p. 424\)](#)。)
- 允许具有特定路径的用户使用策略模拟器 API ([查看该策略 \(p. 424\)](#)。)
- 允许具有特定路径的用户使用策略模拟器控制台 ([查看该策略 \(p. 424\)](#)。)
- 允许 IAM 用户自行管理 MFA 设备 ([查看该策略 \(p. 425\)](#)。)
- 允许 IAM 用户以编程方式和在控制台中轮换其凭证。 ([查看该策略 \(p. 426\)](#)。)
- 允许在 IAM 控制台中查看 Amazon Organizations 策略的上次访问的服务信息。 ([查看该策略 \(p. 427\)](#)。)
- 对可应用于 IAM 用户、组或角色的托管策略加以限制 ([查看该策略 \(p. 427\)](#)。)

示例策略：Amazon Lambda

- 允许 Amazon Lambda 函数访问 Amazon DynamoDB 表 ([查看该策略 \(p. 428\)](#)。)

示例策略：Amazon RDS

- Amazon RDS 允许在特定区域中进行完全 RDS 数据库访问。 ([查看该策略 \(p. 429\)](#)。)
- 允许以编程方式和在控制台中还原 Amazon RDS 数据库 ([查看该策略 \(p. 429\)](#)。)
- 允许标签所有者不受限制地访问其标记的 Amazon RDS 资源 ([查看该策略 \(p. 429\)](#)。)

示例策略：Amazon S3

- 允许 Amazon Cognito 用户访问自己的 Amazon S3 存储桶中的对象 ([查看该策略 \(p. 431\)](#)。)
- 允许联合身份用户以编程方式和在控制台中访问他们位于 Amazon S3 中的主目录 ([查看该策略 \(p. 432\)](#)。)
- 允许完全 S3 访问，但如果管理员在过去的 30 分钟内未使用 MFA 登录，则显式拒绝访问 Production 存储桶 ([查看该策略 \(p. 433\)](#)。)
- 允许 IAM 用户以编程方式和在控制台中访问自己位于 Amazon S3 中的主目录 ([查看该策略 \(p. 434\)](#)。)
- 允许用户管理单个 Amazon S3 存储桶并拒绝所有其他 Amazon 操作和资源 ([查看该策略 \(p. 435\)](#)。)
- 允许对特定的 Amazon S3 存储桶进行 Read 和 Write 访问 ([查看该策略 \(p. 435\)](#))

- 允许以编程方式和在控制台中对特定的 Amazon S3 存储桶进行 Read 和 Write 访问 ([查看该策略 \(p. 436\)](#)。)

Amazon : 允许基于日期和时间进行访问

此示例说明如何创建允许访问基于日期和时间的操作的基于身份的策略。此策略限制访问 2020 年 4 月 1 日和 2020 年 6 月 30 日 (含这两个日期) 之间发生的操作。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略, 请将示例策略中的 ##### 替换为您自己的信息。然后, 按照 [创建策略 \(p. 437\)](#) 或 [编辑策略 \(p. 461\)](#) 中的说明操作。

要了解在 IAM policy 的 Condition 块中使用多个条件的信息, 请参阅 [一个条件中包含多个值 \(p. 950\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "service-prefix:action-name",
      "Resource": "*",
      "Condition": {
        "DateGreaterThan": {"aws:CurrentTime": "2020-04-01T00:00:00Z"},
        "DateLessThan": {"aws:CurrentTime": "2020-06-30T23:59:59Z"}
      }
    }
  ]
}
```

Note

您不能对策略变量使用 Date 条件运算符。要了解更多信息, 请参阅 [条件元素 \(p. 968\)](#)。

Amazon : 允许启用和禁用 Amazon 区域

此示例现在可以创建基于身份的策略, 以允许管理员启用和禁用亚太地区 (香港) 区域 (ap-east-1)。此策略定义了程序访问和控制台访问的权限。此设置显示在 Amazon Web Services Management Console 中的 Account settings (账户设置) 页面。该页面包含只应由账户管理员查看和管理的敏感账户级信息。要使用此策略, 请将示例策略中的 ##### 替换为您自己的信息。然后, 按照 [创建策略 \(p. 437\)](#) 或 [编辑策略 \(p. 461\)](#) 中的说明操作。

Important

您无法启用或禁用默认启用的区域。您只能包含默认禁用的区域。有关更多信息, 请参阅 Amazon 一般参考中的 [管理 Amazon 区域](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnableDisableHongKong",
      "Effect": "Allow",
      "Action": [
        "account:EnableRegion",
        "account:DisableRegion"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {"account:TargetRegion": "ap-east-1" }
      }
    },
    {
      "Sid": "ViewConsole",
```

```
        "Effect": "Allow",
        "Action": [
            "aws-portal:ViewAccount",
            "account:ListRegions"
        ],
        "Resource": "*"
    }
}
```

Amazon：允许经过 MFA 身份验证的 IAM 用户在“My Security Credentials”（我的安全凭证）页面上管理自己的凭证

此示例显示您可以创建基于身份的策略，允许通过[多重身份验证（MFA）](#)（p. 104）进行身份验证的 IAM 用户在 My Security Credentials（我的安全凭证）页面上管理他们自己的凭证。此 Amazon Web Services Management Console 页面显示账户信息，例如账户 ID 和规范用户 ID。用户还可以查看和编辑自己的密码、访问密钥、MFA 设备、X.509 证书、SSH 密钥和 Git 凭证。此示例策略包括查看和编辑页面上所有信息所需的权限。它还要求用户在执行 Amazon 中的任何其他操作之前使用 MFA 进行设置和身份验证。要允许用户在不使用 MFA 的情况下管理自己的凭证，请参阅[Amazon：允许 IAM 用户在“My Security Credentials”（我的安全凭证）页面上管理其凭证](#)（p. 398）。

要了解用户如何访问 My Security Credentials（我的安全凭证）页面，请参阅[IAM 用户如何更改自己的密码（控制台）](#)（p. 95）。

Note

- 此示例策略不允许用户在首次登录 Amazon Web Services Management Console 的同时重置密码。我们建议您在新用户登录之前不要向他们授予权限。有关更多信息，请参阅[如何安全地创建 IAM 用户？](#)（p. 866）。这还可以防止密码过期的用户在登录期间重置其密码。您可以通过向语句 `DenyAllExceptListedIfNoMFA` 中添加 `iam:ChangePassword` 和 `iam:GetAccountPasswordPolicy` 来允许此操作。但是，我们不建议您这样做，因为允许用户在进行 MFA 验证的情况下更改密码可能存在安全风险。
- 如果您打算将此策略用于编程访问，则必须调用 `GetSessionToken` 以使用 MFA 进行身份验证。有关更多信息，请参阅[配置受 MFA 保护的 API 访问](#)（p. 129）。

此策略有何作用？

- `AllowViewAccountInfo` 语句允许用户查看账户级信息。这些权限必须位于自己的语句中，因为它们不支持或不需要指定资源 ARN。相反，权限指定 `"Resource" : "*"`。此语句包括允许用户查看特定信息的以下操作：
 - `GetAccountPasswordPolicy` - 查看账户密码要求，同时更改他们自己的 IAM 用户密码。
 - `ListVirtualMFADevices` - 查看有关为用户启用的虚拟 MFA 设备的详细信息。
- `AllowManageOwnPasswords` 语句还允许用户更改他们自己的密码。此语句还包括 `GetUser` 操作，查看 My Security Credentials（我的安全凭证）页面上的大多数信息都需要此操作。
- `AllowManageOwnAccessKeys` 语句允许用户创建、更新和删除他们自己的访问密钥。
- `AllowManageOwnSigningCertificates` 语句允许用户上传、更新和删除他们自己的签名证书。
- `AllowManageOwnSSHPublicKeys` 语句允许用户上传、更新和删除他们自己的 CodeCommit 的 SSH 公有密钥。
- `AllowManageOwnGitCredentials` 语句允许用户创建、更新和删除他们自己的 CodeCommit 的 Git 凭证。
- `AllowManageOwnVirtualMFADevice` 语句允许用户创建和删除他们自己的虚拟 MFA 设备。此语句中的资源 ARN 仅允许访问与当前登录用户具有相同名称的 MFA 设备。用户不能创建或删除除自己设备外的任何虚拟 MFA 设备。
- `AllowManageOwnUserMFA` 语句允许用户查看或管理他们自己用户的虚拟、U2F 或硬件 MFA 设备。此语句中的资源 ARN 仅允许访问用户自己的 IAM 用户。用户无法查看或管理其他用户的 MFA 设备。

- `DenyAllExceptListedIfNoMFA` 语句拒绝访问所有 Amazon 服务中的每个操作（除了一些列出的操作），但前提是用户未使用 MFA 登录。该语句使用 `"Deny"` 和 `"NotAction"` 的组合来显式拒绝对未列出的每个操作的访问。此语句不会拒绝或允许列出的项目。但是，策略中的其他语句允许这些操作。有关此语句的逻辑的更多信息，请参阅 [NotAction](#) 以及 [Deny \(p. 945\)](#)。如果用户已使用 MFA 登录，则 `Condition` 测试将失败，并且此语句不会拒绝任何操作。在这种情况下，用户的其他策略或语句确定用户的权限。

此语句确保当用户未使用 MFA 登录时，他们只能执行列出的操作。此外，只有在另一个语句或策略允许访问这些操作时，它们才能执行列出的操作。这不允许用户在登录时创建密码，因为如果没有 MFA 授权，则不允许 `iam:ChangePassword` 操作。

`...IfExists` 运算符的 `Bool` 版本可确保：如果 `aws:MultiFactorAuthPresent` ([p. 1002](#)) 键缺失，条件将返回 `true`。这意味着使用长期凭证（例如访问密钥）访问 API 的用户被拒绝访问非 IAM API 操作。

此策略不允许用户在 IAM 控制台中查看 Users（用户）页面，或使用该页面访问自己的用户信息。要允许此操作，请将 `iam:ListUsers` 操作添加到 `AllowViewAccountInfo` 和 `DenyAllExceptListedIfNoMFA` 语句。它也不允许用户在自己的用户页面上更改密码。要允许此操作，请将 `iam:GetLoginProfile` 和 `iam:UpdateLoginProfile` 操作添加到 `AllowManageOwnPasswords` 语句。要允许用户在不使用 MFA 登录的情况下从自己的用户页面更改密码，请将 `iam:UpdateLoginProfile` 操作添加到 `DenyAllExceptListedIfNoMFA` 语句。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowViewAccountInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetAccountPasswordPolicy",
        "iam:ListVirtualMFADevices"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowManageOwnPasswords",
      "Effect": "Allow",
      "Action": [
        "iam:ChangePassword",
        "iam:GetUser"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "AllowManageOwnAccessKeys",
      "Effect": "Allow",
      "Action": [
        "iam:CreateAccessKey",
        "iam>DeleteAccessKey",
        "iam:ListAccessKeys",
        "iam:UpdateAccessKey"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "AllowManageOwnSigningCertificates",
      "Effect": "Allow",
      "Action": [
        "iam>DeleteSigningCertificate",
        "iam:ListSigningCertificates",
        "iam:UpdateSigningCertificate",
        "iam:UploadSigningCertificate"
      ]
    }
  ]
}
```



```

    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
  },
  {
    "Sid": "AllowManageOwnSSHPublicKeys",
    "Effect": "Allow",
    "Action": [
      "iam:DeleteSSHPublicKey",
      "iam:GetSSHPublicKey",
      "iam:ListSSHPublicKeys",
      "iam:UpdateSSHPublicKey",
      "iam:UploadSSHPublicKey"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
  },
  {
    "Sid": "AllowManageOwnGitCredentials",
    "Effect": "Allow",
    "Action": [
      "iam:CreateServiceSpecificCredential",
      "iam:DeleteServiceSpecificCredential",
      "iam:ListServiceSpecificCredentials",
      "iam:ResetServiceSpecificCredential",
      "iam:UpdateServiceSpecificCredential"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
  },
  {
    "Sid": "AllowManageOwnVirtualMFADevice",
    "Effect": "Allow",
    "Action": [
      "iam:CreateVirtualMFADevice",
      "iam:DeleteVirtualMFADevice"
    ],
    "Resource": "arn:aws:iam::*:mfa/${aws:username}"
  },
  {
    "Sid": "AllowManageOwnUserMFA",
    "Effect": "Allow",
    "Action": [
      "iam:DeactivateMFADevice",
      "iam:EnableMFADevice",
      "iam:ListMFADevices",
      "iam:ResyncMFADevice"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
  },
  {
    "Sid": "DenyAllExceptListedIfNoMFA",
    "Effect": "Deny",
    "NotAction": [
      "iam:CreateVirtualMFADevice",
      "iam:EnableMFADevice",
      "iam:GetUser",
      "iam:ListMFADevices",
      "iam:ListVirtualMFADevices",
      "iam:ResyncMFADevice",
      "sts:GetSessionToken"
    ],
    "Resource": "*",
    "Condition": {
      "BoolIfExists": {
        "aws:MultiFactorAuthPresent": "false"
      }
    }
  }
}

```

```
}
}
```

Amazon：允许在特定日期内使用 MFA 进行特定访问

此示例说明了如何创建一个基于身份的策略，该策略将使用多个条件，借助 AND 逻辑对这些条件进行评估。它允许对名为 SERVICE-NAME-1 的服务进行完全访问，并且允许对名为 ACTION-NAME-A 的服务中的 ACTION-NAME-B 和 SERVICE-NAME-2 操作进行访问。只有在用户使用[多重身份验证 \(MFA\)](#) 进行身份验证时，才允许执行这些操作。只能对在 2017 年 7 月 1 日至 2017 年 12 月 31 日 (UTC 时间，包含这两个时间) 期间发生的操作进行访问。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

要了解在 IAM policy 的 Condition 块中使用多个条件的信息，请参阅[一个条件中包含多个值 \(p. 950\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "service-prefix-1:*",
      "service-prefix-2:action-name-a",
      "service-prefix-2:action-name-b"
    ],
    "Resource": "*",
    "Condition": {
      "Bool": {"aws:MultiFactorAuthPresent": true},
      "DateGreaterThan": {"aws:CurrentTime": "2017-07-01T00:00:00Z"},
      "DateLessThan": {"aws:CurrentTime": "2017-12-31T23:59:59Z"}
    }
  }
}
```

Amazon：允许 IAM 用户在“My Security Credentials”（我的安全凭证）页面上管理其凭证

此示例说明了如何创建基于身份的策略，以允许 IAM 用户在 My Security Credentials（我的安全凭证）页管理其所拥有的凭证。此 Amazon Web Services Management Console 页面显示账户信息，例如账户 ID 和规范用户 ID。用户还可以查看和编辑自己的密码、访问密钥、X.509 证书、SSH 密钥和 Git 凭证。此示例策略包括查看和编辑页面上所有信息（用户的 MFA 设备除外）所需的权限。要允许用户使用 MFA 管理他们自己的凭证，请参阅[Amazon：允许经过 MFA 身份验证的 IAM 用户在“My Security Credentials”（我的安全凭证）页面上管理自己的凭证 \(p. 395\)](#)。

要了解用户如何访问 My Security Credentials（我的安全凭证）页面，请参阅[IAM 用户如何更改自己的密码（控制台） \(p. 95\)](#)。

此策略有何作用？

- AllowViewAccountInfo 语句允许用户查看账户级信息。这些权限必须位于自己的语句中，因为它们不支持或不需要指定资源 ARN。相反，权限指定 "Resource" : "*"。此语句包括允许用户查看特定信息的以下操作：
 - GetAccountPasswordPolicy - 查看账户密码要求，同时更改他们自己的 IAM 用户密码。
 - GetAccountSummary - 查看账户 ID 和账户的[规范用户 ID](#)。
- AllowManageOwnPasswords 语句还允许用户更改他们自己的密码。此语句还包括 GetUser 操作，查看 My Security Credentials（我的安全凭证）页面上的大多数信息都需要此操作。
- AllowManageOwnAccessKeys 语句允许用户创建、更新和删除他们自己的访问密钥。

- AllowManageOwnSigningCertificates 语句允许用户上传、更新和删除他们自己的签名证书。
- AllowManageOwnSSHPublicKeys 语句允许用户上传、更新和删除他们自己的 CodeCommit 的 SSH 公有密钥。
- AllowManageOwnGitCredentials 语句允许用户创建、更新和删除他们自己的 CodeCommit 的 Git 凭证。

该策略不允许用户查看或管理他们自己的 MFA 设备。他们也不能在 IAM 控制台中查看 Users (用户) 页面，或使用该页面访问自己的用户信息。要允许此操作，请将 iam:ListUsers 操作添加到 AllowViewAccountInfo 语句。它也不允许用户在自己的用户页面上更改密码。要允许此操作，请将 iam:CreateLoginProfile、iam:DeleteLoginProfile、iam:GetLoginProfile 和 iam:UpdateLoginProfile 操作添加到 AllowManageOwnPasswords 语句。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowViewAccountInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetAccountPasswordPolicy",
        "iam:GetAccountSummary"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowManageOwnPasswords",
      "Effect": "Allow",
      "Action": [
        "iam:ChangePassword",
        "iam:GetUser"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "AllowManageOwnAccessKeys",
      "Effect": "Allow",
      "Action": [
        "iam:CreateAccessKey",
        "iam:DeleteAccessKey",
        "iam:ListAccessKeys",
        "iam:UpdateAccessKey"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "AllowManageOwnSigningCertificates",
      "Effect": "Allow",
      "Action": [
        "iam:DeleteSigningCertificate",
        "iam:ListSigningCertificates",
        "iam:UpdateSigningCertificate",
        "iam:UploadSigningCertificate"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "AllowManageOwnSSHPublicKeys",
      "Effect": "Allow",
      "Action": [
        "iam:DeleteSSHPublicKey",
        "iam:GetSSHPublicKey",
        "iam:ListSSHPublicKeys",
```

```
        "iam:UpdateSSHPublicKey",
        "iam:UploadSSHPublicKey"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
},
{
    "Sid": "AllowManageOwnGitCredentials",
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceSpecificCredential",
        "iam:DeleteServiceSpecificCredential",
        "iam:ListServiceSpecificCredentials",
        "iam:ResetServiceSpecificCredential",
        "iam:UpdateServiceSpecificCredential"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}"
}
]
```

Amazon：允许经过 MFA 身份验证的 IAM 用户在“My Security Credentials”（我的安全凭证）页面上管理自己的 MFA 设备。

此示例显示您可以创建基于身份的策略，允许通过[多重身份验证 \(MFA\)](#) (p. 104)进行身份验证的 IAM 用户在 My Security Credentials（我的安全凭证）页面上管理他们自己的 MFA 设备。此 Amazon Web Services Management Console 页面显示账户和用户信息，但是用户只能查看和编辑他们自己的 MFA 设备。要允许用户使用 MFA 管理他们自己的所有凭证，请参阅[Amazon：允许经过 MFA 身份验证的 IAM 用户在“My Security Credentials”（我的安全凭证）页面上管理自己的凭证](#) (p. 395)。

Note

如果具有此策略的 IAM 用户未通过 MFA 进行身份验证，则此策略会拒绝对所有 Amazon 操作（使用 MFA 进行身份验证所需的操作除外）的访问。要使用 Amazon CLI 和 Amazon API，IAM 用户必须先使用 Amazon STS [GetSessionToken](#) 操作检索其 MFA 令牌，然后使用该令牌验证所需的操作。其他策略（如基于资源的策略或其他基于身份的策略）可以允许其他服务中的操作。如果 IAM 用户没有经过 MFA 身份验证，则此策略将拒绝该访问。

要了解用户如何访问 My Security Credentials（我的安全凭证）页面，请参阅[IAM 用户如何更改自己的密码（控制台）](#) (p. 95)。

此策略有何作用？

- `AllowViewAccountInfo` 语句允许用户查看有关为用户启用的虚拟 MFA 设备的详细信息。此权限必须在它自己的语句中，因为它不支持指定资源 ARN。相反，您必须指定 `"Resource" : "*"` 。
- `AllowManageOwnVirtualMFADevice` 语句允许用户创建和删除他们自己的虚拟 MFA 设备。此语句中的资源 ARN 仅允许访问与当前登录用户具有相同名称的 MFA 设备。用户不能创建或删除除自己设备外的任何虚拟 MFA 设备。
- `AllowManageOwnUserMFA` 语句允许用户查看或管理他们自己的虚拟、U2F 或硬件 MFA 设备。此语句中的资源 ARN 仅允许访问用户自己的 IAM 用户。用户无法查看或管理其他用户的 MFA 设备。
- `DenyAllExceptListedIfNoMFA` 语句拒绝访问所有 Amazon 服务中的每个操作（除了一些列出的操作），但前提是用户未使用 MFA 登录。该语句使用 `"Deny"` 和 `"NotAction"` 的组合来显式拒绝对未列出的每个操作的访问。此语句不会拒绝或允许列出的项目。但是，策略中的其他语句允许这些操作。有关此语句的逻辑的更多信息，请参阅 [NotAction 以及 Deny](#) (p. 945)。如果用户已使用 MFA 登录，则 `Condition` 测试将失败，并且此语句不会拒绝任何操作。在这种情况下，用户的其他策略或语句确定用户的权限。

此语句确保当用户未使用 MFA 登录时，他们只能执行列出的操作。此外，只有在另一个语句或策略允许访问这些操作时，它们才能执行列出的操作。

...IfExists 运算符的 Bool 版本可确保：如果 aws:MultiFactorAuthPresent 键缺失，条件将返回 true。这意味着使用长期凭证（例如访问密钥）访问 API 操作的用户被拒绝访问非 IAM API 操作。

此策略不允许用户在 IAM 控制台中查看 Users（用户）页面，或使用该页面访问自己的用户信息。要允许此操作，请将 iam:ListUsers 操作添加到 AllowViewAccountInfo 和 DenyAllExceptListedIfNoMFA 语句。

Warning

在未通过 MFA 进行身份验证的情况下，请不要添加删除 MFA 设备的权限。具有该策略的用户可能会尝试为他们自己分配一个 MFA 设备，并出现错误以指示未授权他们执行 iam:DeleteVirtualMFADevice。如果发生这种情况，请不要将该权限添加到 DenyAllExceptListedIfNoMFA 语句中。切勿允许未使用 MFA 进行身份验证的用户删除 MFA 设备。如果用户以前开始将虚拟 MFA 设备分配给用户并取消了该过程，则可能会看到该错误。要解决该问题，您或其他管理员必须使用 Amazon CLI 或 Amazon API 删除用户的现有 MFA 设备。有关更多信息，请参阅[我没有权限执行：iam:DeleteVirtualMFADevice \(p. 866\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowViewAccountInfo",
      "Effect": "Allow",
      "Action": "iam:ListVirtualMFADevices",
      "Resource": "*"
    },
    {
      "Sid": "AllowManageOwnVirtualMFADevice",
      "Effect": "Allow",
      "Action": [
        "iam:CreateVirtualMFADevice",
        "iam:DeleteVirtualMFADevice"
      ],
      "Resource": "arn:aws:iam::*:mfa/${aws:username}"
    },
    {
      "Sid": "AllowManageOwnUserMFA",
      "Effect": "Allow",
      "Action": [
        "iam:DeactivateMFADevice",
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:ListMFADevices",
        "iam:ResyncMFADevice"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "DenyAllExceptListedIfNoMFA",
      "Effect": "Deny",
      "NotAction": [
        "iam:CreateVirtualMFADevice",
        "iam:EnableMFADevice",
        "iam:GetUser",
        "iam:ListMFADevices",
        "iam:ListVirtualMFADevices",
        "iam:ResyncMFADevice",
        "sts:GetSessionToken"
      ],
      "Resource": "*",
      "Condition": {
        "BoolIfExists": {"aws:MultiFactorAuthPresent": "false"}
      }
    }
  ]
}
```

```
}  
  }  
]  
}
```

Amazon：允许 IAM 用户在“My Security Credentials”（我的安全凭证）页面上更改自己的控制台密码

此示例说明如何创建基于身份的策略，以允许 IAM 用户更改其在 My Security Credentials（我的安全凭证）页面上 Amazon Web Services Management Console 密码。此 Amazon Web Services Management Console 页面显示账户和用户信息，但是用户只能访问他们自己的密码。要允许用户使用 MFA 管理他们自己的所有凭证，请参阅[Amazon：允许经过 MFA 身份验证的 IAM 用户在“My Security Credentials”（我的安全凭证）页面上管理自己的凭证](#)（p. 395）。要允许用户在不使用 MFA 的情况下管理自己的凭证，请参阅[Amazon：允许 IAM 用户在“My Security Credentials”（我的安全凭证）页面上管理其凭证](#)（p. 398）。

要了解用户如何访问 My Security Credentials（我的安全凭证）页面，请参阅[IAM 用户如何更改自己的密码（控制台）](#)（p. 95）。

此策略有何作用？

- ViewAccountPasswordRequirements 语句允许用户查看账户密码要求，同时更改他们自己的 IAM 用户密码。
- ChangeOwnPassword 语句还允许用户更改他们自己的密码。此语句还包括 GetUser 操作，查看 My Security Credentials（我的安全凭证）页面上的大多数信息都需要此操作。

此策略不允许用户在 IAM 控制台中查看 Users（用户）页面，或使用该页面访问自己的用户信息。要允许此操作，请将 iam:ListUsers 操作添加到 ViewAccountPasswordRequirements 语句。它也不允许用户在自己的用户页面上更改密码。要允许此操作，请将 iam:GetLoginProfile 和 iam:UpdateLoginProfile 操作添加到 ChangeOwnPasswords 语句。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "ViewAccountPasswordRequirements",  
      "Effect": "Allow",  
      "Action": "iam:GetAccountPasswordPolicy",  
      "Resource": "*"  
    },  
    {  
      "Sid": "ChangeOwnPassword",  
      "Effect": "Allow",  
      "Action": [  
        "iam:GetUser",  
        "iam:ChangePassword"  
      ],  
      "Resource": "arn:aws:iam::*:user/${aws:username}"  
    }  
  ]  
}
```

Amazon：允许 IAM 用户在“My Security Credentials”（我的安全凭证）页面上管理自己的密码、访问密钥和 SSH 公有密钥

此示例说明了如何创建基于身份的策略，以允许 IAM 用户管理其在 My Security Credentials（我的安全凭证）页面上的密码、访问密钥和 X.509 证书。此 Amazon Web Services Management Console 页面显示账户信息，例如账户 ID 和规范用户 ID。用户还可以查看和编辑自己的密码、访问密钥、MFA 设备、X.509 证

书、SSH 密钥和 Git 凭证。此示例策略包括仅查看和编辑其密码、访问密钥和 X.509 证书所需的权限。要允许用户使用 MFA 管理他们自己的所有凭证，请参阅[Amazon：允许经过 MFA 身份验证的 IAM 用户在“My Security Credentials”（我的安全凭证）页面上管理自己的凭证](#)（p. 395）。要允许用户在不使用 MFA 的情况下管理自己的凭证，请参阅[Amazon：允许 IAM 用户在“My Security Credentials”（我的安全凭证）页面上管理其凭证](#)（p. 398）。

要了解用户如何访问 My Security Credentials（我的安全凭证）页面，请参阅[IAM 用户如何更改自己的密码（控制台）](#)（p. 95）。

此策略有何作用？

- AllowViewAccountInfo 语句允许用户查看账户级信息。这些权限必须位于自己的语句中，因为它们不支持或不需要指定资源 ARN。相反，权限指定 "Resource"："*"。此语句包括允许用户查看特定信息的以下操作：
 - GetAccountPasswordPolicy – 查看账户密码要求，同时更改他们自己的 IAM 用户密码。
 - GetAccountSummary – 查看账户 ID 和账户的[规范用户 ID](#)。
- AllowManageOwnPasswords 语句还允许用户更改他们自己的密码。此语句还包括 GetUser 操作，查看 My Security Credentials（我的安全凭证）页面上的大多数信息都需要此操作。
- AllowManageOwnAccessKeys 语句允许用户创建、更新和删除他们自己的访问密钥。
- AllowManageOwnSSHPublicKeys 语句允许用户上传、更新和删除他们自己的 CodeCommit 的 SSH 公有密钥。

该策略不允许用户查看或管理他们自己的 MFA 设备。他们也不能在 IAM 控制台中查看 Users（用户）页面，或使用该页面访问自己的用户信息。要允许此操作，请将 iam:ListUsers 操作添加到 AllowViewAccountInfo 语句。它也不允许用户在自己的用户页面上更改密码。要允许此操作，请将 iam:GetLoginProfile 和 iam:UpdateLoginProfile 操作添加到 AllowManageOwnPasswords 语句。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowViewAccountInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetAccountPasswordPolicy",
        "iam:GetAccountSummary"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowManageOwnPasswords",
      "Effect": "Allow",
      "Action": [
        "iam:ChangePassword",
        "iam:GetUser"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
      "Sid": "AllowManageOwnAccessKeys",
      "Effect": "Allow",
      "Action": [
        "iam:CreateAccessKey",
        "iam:DeleteAccessKey",
        "iam:ListAccessKeys",
        "iam:UpdateAccessKey"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    }
  ],
}
```

```
{
  "Sid": "AllowManageOwnSSHPublicKeys",
  "Effect": "Allow",
  "Action": [
    "iam:DeleteSSHPublicKey",
    "iam:GetSSHPublicKey",
    "iam:ListSSHPublicKeys",
    "iam:UpdateSSHPublicKey",
    "iam:UploadSSHPublicKey"
  ],
  "Resource": "arn:aws:iam::*:user/${aws:username}"
}
```

Amazon：根据请求的区域拒绝访问 Amazon

此示例说明了如何创建一个基于身份的策略，以拒绝对使用 `aws:RequestedRegion` 条件键指定的区域以外的任何操作的访问，但服务中使用 `NotAction` 指定的操作除外。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的 `#####` 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

该策略使用具有 `Deny` 效果的 `NotAction` 元素，这会显式拒绝访问语句中未列出的所有操作。不应拒绝 CloudFront、IAM、Route 53 和 Amazon Web Services Support 服务中的操作，因为这些服务是常用的 Amazon 全球服务，并具有一个实际位于 `us-east-1` 区域的终端节点。由于针对这些服务的所有请求都是向 `us-east-1` 区域发出的，因此，将在没有 `NotAction` 元素的情况下拒绝这些请求。请编辑该元素以包含您使用的其他 Amazon 全球服务（如 `budgets`、`globalaccelerator`、`importexport`、`organizations` 或 `waf`）的操作。一些其他全球服务（例如 Amazon Chatbot 和 Amazon Device Farm）是具有物理位置位于 `us-west-2` 区域的终端节点的全球服务。要了解具有单个全球终端节点的所有服务，请参阅 Amazon 一般参考中的[Amazon 区域和终端节点](#)。有关使用具有 `Deny` 效果的 `NotAction` 元素的更多信息，请参阅[IAM JSON 策略元素：NotAction \(p. 945\)](#)。

Important

该策略不允许进行任何操作。可将此策略与允许特定操作的其他策略结合使用。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAllOutsideRequestedRegions",
      "Effect": "Deny",
      "NotAction": [
        "cloudfront:*",
        "iam:*",
        "route53:*",
        "support:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:RequestedRegion": [
            "eu-central-1",
            "eu-west-1",
            "eu-west-2",
            "eu-west-3"
          ]
        }
      }
    }
  ]
}
```

```
}
```

Amazon：基于源 IP 拒绝对 Amazon 的访问

此示例说明了如何创建基于身份的策略，以当请求来自指定 IP 范围以外的主体时拒绝对账户中所有 Amazon 操作的访问。当您公司的 IP 地址位于指定范围内时，该策略很有用。该策略不拒绝 Amazon 服务使用主体的凭证发出的请求。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

在与 "Effect": "Deny" 相同的策略语句中谨慎使用否定条件。使用否定条件时，在所有条件下（指定的条件除外），在策略语句中指定的操作将被明确拒绝。

此外，此策略还包括导致逻辑 AND 的[多条件键 \(p. 958\)](#)。在此策略中，当源 IP 地址不在指定范围内且当 Amazon 服务未进行调用时，将会拒绝所有 Amazon 操作。notANDnot

Important

该策略不允许进行任何操作。可将此策略与允许特定操作的其他策略结合使用。

当其他策略允许操作时，主体可以从 IP 地址范围内发出请求。Amazon 服务还可以使用主体的凭证发出请求。当主体从 IP 范围之外发出请求时，请求将被拒绝。

有关使用 aws:SourceIp 和 aws:ViaAWSService 条件键的更多信息，包括有关 aws:SourceIp 密钥可能无法在您的策略中起作用的信息，请参阅[Amazon 全局条件上下文密钥 \(p. 998\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "NotIpAddress": {
        "aws:SourceIp": [
          "192.0.2.0/24",
          "203.0.113.0/24"
        ]
      },
      "Bool": { "aws:ViaAWSService": "false" }
    }
  }
}
```

Amazon Data Pipeline：拒绝用户访问他人创建的 DataPipeline 管道

此示例说明如何创建基于身份的策略以拒绝对用户未创建的管道的访问。如果 PipelineCreator 字段的值与 IAM 用户名匹配，则指定的操作不会被拒绝。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。

Important

该策略不允许进行任何操作。可将此策略与允许特定操作的其他策略结合使用。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExplicitDenyIfNotTheOwner",
```

```

    "Effect": "Deny",
    "Action": [
        "datapipeline:ActivatePipeline",
        "datapipeline:AddTags",
        "datapipeline:DeactivatePipeline",
        "datapipeline>DeletePipeline",
        "datapipeline:DescribeObjects",
        "datapipeline:EvaluateExpression",
        "datapipeline:GetPipelineDefinition",
        "datapipeline:PollForTask",
        "datapipeline:PutPipelineDefinition",
        "datapipeline:QueryObjects",
        "datapipeline:RemoveTags",
        "datapipeline:ReportTaskProgress",
        "datapipeline:ReportTaskRunnerHeartbeat",
        "datapipeline:SetStatus",
        "datapipeline:SetTaskStatus",
        "datapipeline:ValidatePipelineDefinition"
    ],
    "Resource": ["*"],
    "Condition": {
        "StringNotEquals": {"datapipeline:PipelineCreator": "${aws:userid}"}
    }
}
]
}

```

Amazon DynamoDB：允许访问特定的表

此示例说明了如何创建基于身份的策略以允许对 MyTable DynamoDB 表的完全访问。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

Important

该策略允许可在 DynamoDB 表上执行的所有操作。要了解这些操作的信息，请参阅 Amazon DynamoDB 开发人员指南中的 [DynamoDB API 权限：操作、资源和条件参考](#)。您可以通过列出每个单独的操作来提供同样的权限。不过，如果您在 Action 元素中使用通配符 (*) (例如，"dynamodb:List*")，则在 DynamoDB 添加新的列表操作时无需更新策略。

该策略只允许在具有指定名称的 DynamoDB 表上执行操作。要允许您的用户对 DynamoDB 中的所有内容进行 Read 访问，也可以附加 [AmazonDynamoDBReadOnlyAccess](#) Amazon 托管策略。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListAndDescribe",
      "Effect": "Allow",
      "Action": [
        "dynamodb:List*",
        "dynamodb:DescribeReservedCapacity*",
        "dynamodb:DescribeLimits",
        "dynamodb:DescribeTimeToLive"
      ],
      "Resource": "*"
    },
    {
      "Sid": "SpecificTable",
      "Effect": "Allow",
      "Action": [
        "dynamodb:BatchGet*",
        "dynamodb:DescribeStream",

```

```
        "dynamodb:DescribeTable",
        "dynamodb:Get*",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:BatchWrite*",
        "dynamodb:CreateTable",
        "dynamodb>Delete*",
        "dynamodb:Update*",
        "dynamodb:PutItem"
    ],
    "Resource": "arn:aws:dynamodb:*:*:table/MyTable"
}
]
```

Amazon DynamoDB：允许访问特定的属性

此示例说明如何创建基于身份的策略以允许访问特定 DynamoDB 属性。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 **#####** 替换为您自己的信息。然后，按照[创建策略](#) (p. 437) 或 [编辑策略](#) (p. 461) 中的说明操作。

dynamodb:Select 要求会防止 API 操作返回未授权的属性，例如来自索引投影的属性。要了解有关 DynamoDB 条件键的更多信息，请参阅 Amazon DynamoDB 开发人员指南中的[指定条件：使用条件键](#)。要了解在 IAM policy 的 Condition 块中使用多个条件或多个条件键的信息，请参阅[一个条件中包含多个值](#) (p. 950)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb>DeleteItem",
        "dynamodb:BatchWriteItem"
      ],
      "Resource": ["arn:aws:dynamodb:*:*:table/table-name"],
      "Condition": {
        "ForAllValues:StringEquals": {
          "dynamodb:Attributes": [
            "column-name-1",
            "column-name-2",
            "column-name-3"
          ]
        },
        "StringEqualsIfExists": {"dynamodb:Select": "SPECIFIC_ATTRIBUTES"}
      }
    }
  ]
}
```

Amazon DynamoDB：允许基于 Amazon Cognito ID 对 DynamoDB 进行项目级别的访问

此示例说明了如何创建基于身份的策略以允许基于 Amazon Cognito ID 对 MyTable DynamoDB 表的项目级访问。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将

示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

要使用该策略，您必须将 DynamoDB 表构建为使用 Amazon Cognito 用户 ID 作为分区键。有关更多信息，请参阅 Amazon DynamoDB 开发人员指南中的[创建表](#)。

要了解有关 DynamoDB 条件键的更多信息，请参阅 Amazon DynamoDB 开发人员指南中的[指定条件：使用条件键](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Query",
        "dynamodb:UpdateItem"
      ],
      "Resource": ["arn:aws:dynamodb:*:*:table/MyTable"],
      "Condition": {
        "ForAllValues:StringEquals": {
          "dynamodb:LeadingKeys": ["${cognito-identity.amazonaws.com:sub}"]
        }
      }
    }
  ]
}
```

Amazon EC2：在 EC2 实例上附加或分离卷

此示例说明了如何创建基于身份的策略以允许 EBS 卷所有者向指定 EC2 实例附加或从中分离卷。实例由 Condition 元素中的 ARN 指定。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

Amazon EC2 实例可以使用附加到实例配置文件的 [EC2 实例 Amazon 服务角色 \(p. 159\)](#)授予的权限运行 Amazon 命令。您可以将该策略挂载到角色，或将该语句添加到现有策略中。仅由 *instance-id* 标识的实例可以针对账户中的实例附加或分离卷（包括其自身）。更大的策略中可能存在的其他语句元素不受这个语句的限制的影响。有关创建 IAM policy 以控制对 Amazon EC2 资源访问的更多信息，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的[控制对 Amazon EC2 资源的访问](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AttachVolume",
        "ec2:DetachVolume"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:volume/*",
        "arn:aws:ec2:*:*:instance/*"
      ],
      "Condition": {
        "ArnEquals": {"ec2:SourceInstanceARN": "arn:aws:ec2:*:*:instance/instance-id"}
      }
    }
  ]
}
```



```
    ]
  }
}
```

Amazon EC2：基于标签为 EC2 实例附加或分离 Amazon EBS 卷

此示例显示您可以创建基于身份的策略，允许 EBS 卷所有者为标记为开发实例 (VolumeUser) 的 EC2 实例附加或分离使用标签 Department=Dev 定义的 EBS 卷。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AttachVolume",
        "ec2:DetachVolume"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "StringEquals": {"aws:ResourceTag/Department": "Development"}
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AttachVolume",
        "ec2:DetachVolume"
      ],
      "Resource": "arn:aws:ec2:*:*:volume/*",
      "Condition": {
        "StringEquals": {"aws:ResourceTag/VolumeUser": "${aws:username}"}
      }
    }
  ]
}
```

Amazon EC2：允许以编程方式和使用控制台在特定子网中启动 EC2 实例

此示例说明您如何创建基于身份的策略以允许列出所有 EC2 对象的信息并在特定的子网中启动 EC2 实例。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:Describe*",
        "ec2:GetConsole*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "ec2:RunInstances",
      "Resource": [
```

```
        "arn:aws:ec2:*:*:subnet/subnet-subnet-id",
        "arn:aws:ec2:*:*:network-interface/*",
        "arn:aws:ec2:*:*:instance/*",
        "arn:aws:ec2:*:*:volume/*",
        "arn:aws:ec2:*:*:image/ami-*",
        "arn:aws:ec2:*:*:key-pair/*",
        "arn:aws:ec2:*:*:security-group/*"
    ]
  }
}
```

Amazon EC2：允许以编程方式和在控制台中管理关联至特定标签密钥值对的 EC2 安全组

此示例说明如何创建基于身份的策略，该策略授予用户对具有相同标签的安全组采取某些操作的权限。该策略授予相应的权限，以便在 Amazon EC2 控制台中查看安全组、添加和删除入站和出站规则，以及为具有标签 Department=Test 的现有安全组列出并修改规则描述。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照 [创建策略 \(p. 437\)](#) 或 [编辑策略 \(p. 461\)](#) 中的说明操作。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeSecurityGroups",
      "ec2:DescribeSecurityGroupRules",
      "ec2:DescribeTags"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:AuthorizeSecurityGroupIngress",
      "ec2:RevokeSecurityGroupIngress",
      "ec2:AuthorizeSecurityGroupEgress",
      "ec2:RevokeSecurityGroupEgress",
      "ec2:ModifySecurityGroupRules",
      "ec2:UpdateSecurityGroupRuleDescriptionsIngress",
      "ec2:UpdateSecurityGroupRuleDescriptionsEgress"
    ],
    "Resource": [
      "arn:aws:ec2:region:111122223333:security-group/*"
    ],
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/Department": "Test"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:ModifySecurityGroupRules"
    ],
    "Resource": [
      "arn:aws:ec2:region:111122223333:security-group-rule/*"
    ]
  }
]
```

}

Amazon EC2：允许以编程方式和在控制台中启动或停止用户已标记的 EC2 实例

此示例说明了如何创建基于身份的策略以允许 IAM 用户启动或停止 EC2 实例，但仅限实例标签 Owner 具有该用户的用户名值时。此策略定义了程序访问和控制台访问的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Owner": "${aws:username}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:DescribeInstances",
      "Resource": "*"
    }
  ]
}
```

EC2：基于标签来启动或停止实例

此示例说明了如何创建基于身份的策略以允许使用标签键值对 Project = DataAnalytics 启动或停止实例，但仅限具有标签键值对 Department = Data 的主体执行。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的#####替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

如果策略中的条件的两个部分都为 true，则此条件返回 true。实例必须具有 Project=DataAnalytics 标签。此外，发起请求的 IAM 主体（用户或角色）必须具有 Department=Data 标签。

Note

作为最佳实践，请将具有 aws:PrincipalTag 条件键的策略附加到 IAM 组，以应对一些用户可能具有指定标签而另一些用户可能没有这些标签的情况。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "StartStopIfTags",
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances"
      ],
      "Resource": "arn:aws:ec2:region:account-id:instance/*",
      "Condition": {
        "StringEquals": {
```

```
        "aws:ResourceTag/Project": "DataAnalytics",
        "aws:PrincipalTag/Department": "Data"
      }
    }
  ]
}
```

EC2：基于匹配的主体和资源标签来启动或停止实例

此示例说明如何创建基于身份的策略以允许主体在实例的资源标签和主体标签具有相同的标签键值 CostCenter 时启动或停止 Amazon EC2 实例。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

Note

作为最佳实践，请将具有 aws:PrincipalTag 条件键的策略附加到 IAM 组，以应对一些用户可能具有指定标签而另一些用户可能没有这些标签的情况。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "ec2:startInstances",
      "ec2:stopInstances"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/CostCenter": "${aws:PrincipalTag/CostCenter}"
      }
    }
  }
}
```

Amazon EC2：允许以编程方式和使用控制台在特定区域进行不受限制的 EC2 访问

此示例说明了如何创建基于身份的策略以允许在特定区域中进行完全 EC2 访问。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。有关区域代码的列表，请参阅 Amazon EC2 用户指南中的[可用区域](#)。

或者，您也可以使用全局条件键 [aws:RequestedRegion](#)，所有 Amazon EC2 API 操作均支持该条件键。有关更多信息，请参阅 Amazon EC2 用户指南中的[示例：限制对特定区域的访问](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "ec2:*",
      "Resource": "*",
      "Effect": "Allow",
      "Condition": {
        "StringEquals": {
          "ec2:Region": "us-east-2"
        }
      }
    }
  ]
}
```

Amazon EC2：允许以编程方式和在控制台中启动或停止 EC2 实例并修改安全组

此示例说明如何创建基于身份的策略以允许启动或停止特定的 EC2 实例并修改特定的安全组。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的 `#####` 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSecurityGroupReferences",
        "ec2:DescribeStaleSecurityGroups"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:StartInstances",
        "ec2:StopInstances"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:instance/i-instance-id",
        "arn:aws:ec2:*:*:security-group/sg-security-group-id"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Amazon EC2：要求进行 MFA (GetSessionToken) 以执行特定的 EC2 操作

此示例说明如何创建基于身份的策略以允许完全访问 Amazon EC2 中的 Amazon API 操作。不过，如果未使用[多重身份验证 \(MFA\) \(p. 104\)](#)验证用户身份，则它显式拒绝访问 `StopInstances` 和 `TerminateInstances` API 操作。要以编程方式执行该操作，用户必须在调用 `GetSessionToken` 操作时包含可选的 `SerialNumber` 和 `TokenCode` 值。该操作返回已使用 MFA 验证的临时凭证。要了解 `GetSessionToken` 的更多信息，请参阅[GetSessionToken – 不受信任环境中用户的临时凭证 \(p. 308\)](#)。

此策略有何作用？

- `AllowAllActionsForEC2` 语句允许所有 Amazon EC2 操作。
- 在缺少 MFA 上下文时，`DenyStopAndTerminateWhenMFAIsNotPresent` 语句拒绝 `StopInstances` 和 `TerminateInstances` 操作。这意味着，在缺少多重身份验证上下文（表示不使用 MFA）时，将拒绝这些操作。拒绝将覆盖允许。

Note

`Deny` 语句中的 `MultiFactorAuthPresent` 的条件检查不应为 `{"Bool": {"aws:MultiFactorAuthPresent": false}}`，因为在未使用 MFA 时，该键不存在并且无法

进行评估。因此，在检查值之前，请使用 `BoolIfExists` 检查以确定该键是否存在。有关更多信息，请参阅[...IfExists 条件运算符 \(p. 957\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllActionsForEC2",
      "Effect": "Allow",
      "Action": "ec2:*",
      "Resource": "*"
    },
    {
      "Sid": "DenyStopAndTerminateWhenMFAIsNotPresent",
      "Effect": "Deny",
      "Action": [
        "ec2:StopInstances",
        "ec2:TerminateInstances"
      ],
      "Resource": "*",
      "Condition": {
        "BoolIfExists": {"aws:MultiFactorAuthPresent": false}
      }
    }
  ]
}
```

Amazon EC2：将要终止的 EC2 实例限制为某个 IP 地址范围

此示例显示了如果请求来自指定 IP 范围以外，您可以创建基于身份的策略，通过允许操作但显式拒绝访问来限制 EC2 实例。当您公司的 IP 地址位于指定范围内时，该策略很有用。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 `#####` 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

如果将该策略与允许 `ec2:TerminateInstances` 操作的其他策略（例如 [AmazonEC2FullAccess](#) Amazon 托管策略）一起使用，则访问会被拒绝。这是因为“显式拒绝”声明优先于“允许”声明。有关更多信息，请参阅[the section called “确定是允许还是拒绝账户内的请求” \(p. 974\)](#)。

Important

`aws:SourceIp` 条件键拒绝对 Amazon 服务（例如 Amazon CloudFormation）的访问（代表您进行的调用）。有关使用 `aws:SourceIp` 条件键的更多信息，请参阅[Amazon 全局条件上下文密钥 \(p. 998\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["ec2:TerminateInstances"],
      "Resource": ["*"]
    },
    {
      "Effect": "Deny",
      "Action": ["ec2:TerminateInstances"],
      "Condition": {
        "NotIpAddress": {
          "aws:SourceIp": [
            "192.0.2.0/24",
            "203.0.113.0/24"
          ]
        }
      }
    }
  ]
}
```



```
        "Resource": ["*"]
    }
}
}
```

IAM：访问策略模拟器 API

此示例说明您如何创建基于身份的策略，允许对附加到当前 Amazon 账户中的用户、组或角色的策略使用策略模拟器 API。该策略还允许模拟以字符串形式传递给该 API 的不太敏感的策略。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:GetContextKeysForCustomPolicy",
        "iam:GetContextKeysForPrincipalPolicy",
        "iam:SimulateCustomPolicy",
        "iam:SimulatePrincipalPolicy"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Note

要允许用户访问策略模拟器控制台以模拟挂载附加到当前 Amazon 账户中用户、组或角色的策略，请参阅 [IAM：访问策略模拟器控制台 \(p. 415\)](#)。

IAM：访问策略模拟器控制台

此示例说明您如何创建基于身份的策略，以允许对附加到当前 Amazon 账户中的用户、组或角色的策略使用策略模拟器控制台。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。

您可以在以下位置访问 IAM policy simulator 控制台：<https://policysim.aws.amazon.com/>

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:GetGroup",
        "iam:GetGroupPolicy",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam:GetUser",
        "iam:GetUserPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListAttachedRolePolicies",
        "iam:ListAttachedUserPolicies",
        "iam:ListGroups",
        "iam:ListGroupPolicies",
        "iam:ListGroupsForUser",
        "iam:ListRolePolicies",
        "iam:ListRoles",
        "iam:ListUserPolicies",
        "iam:ListUsers"
      ]
    }
  ]
}
```

```
    ],  
    "Effect": "Allow",  
    "Resource": "*"    
  }  
]  
}
```

IAM：代入具有特定标签的角色

此示例说明了如何创建基于身份的策略以允许 IAM 用户代入具有标签键值对 `Project = ExampleCorpABC` 的角色。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 `#####` 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

如果具有该标签的角色在与该用户相同的账户中存在，则用户可以担任该角色。如果具有该标签的角色在与该用户不同的账户中存在，则需要使用额外的权限。跨账户角色的信任策略还必须允许该用户或其账户的所有成员担任该角色。有关使用角色进行跨账户访问的信息，请参阅[在您拥有的另一个 Amazon 账户中向 IAM 用户提供访问权限 \(p. 161\)](#)。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AssumeTaggedRole",  
      "Effect": "Allow",  
      "Action": "sts:AssumeRole",  
      "Resource": "*",  
      "Condition": {  
        "StringEquals": {"iam:ResourceTag/Project": "ExampleCorpABC"}  
      }  
    }  
  ]  
}
```

IAM：允许或拒绝以编程方式和在控制台中访问多个服务

此示例说明了如何创建基于身份的策略，以允许完全访问多个服务并在 IAM 中进行有限的自我管理访问。该策略还会拒绝对 Amazon S3 logs 存储桶或 Amazon EC2 `i-1234567890abcdef0` 实例的访问。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的 `#####` 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

Warning

该策略允许完全访问多个服务中的每个操作和资源。只应将该策略应用于受信任的管理员。

您可以将该策略作为权限边界，以定义基于身份的策略可以为 IAM 用户授予的最大权限。有关更多信息，请参阅[使用权限边界将责任委派给其他人 \(p. 374\)](#)。在将策略作为用户的权限边界时，这些语句定义以下边界：

- `AllowServices` 语句允许完全访问指定的 Amazon 服务。这意味着，用户在这些服务中的操作仅受附加到用户的权限策略的限制。
- `AllowIAMConsoleForCredentials` 语句允许访问以列出所有 IAM 用户。要在 Amazon Web Services Management Console 中导航用户页面，此访问权限是必需的。它还允许查看账户的密码要求，这是用户更改自己的密码所必需的。
- `AllowManageOwnPasswordAndAccessKeys` 语句允许用户仅管理其自己的控制台密码和编程访问密钥。这是非常重要的，因为如果另一个策略为用户授予完全 IAM 访问权限，该用户可能会更改自己或其他用户的权限。此语句可防止这种情况的发生。
- `DenyS3Logs` 语句显式拒绝对 logs 存储桶的访问。该策略对用户实施公司限制。
- `DenyEC2Production` 语句显式拒绝对 `i-1234567890abcdef0` 实例的访问。

该策略不允许访问其他服务或操作。在将策略作为用户的权限边界时，即使附加到用户的其他策略允许这些操作，Amazon 也会拒绝请求。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowServices",
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "cloudwatch:*",
        "ec2:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowIAMConsoleForCredentials",
      "Effect": "Allow",
      "Action": [
        "iam:ListUsers",
        "iam:GetAccountPasswordPolicy"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowManageOwnPasswordAndAccessKeys",
      "Effect": "Allow",
      "Action": [
        "iam:*AccessKey*",
        "iam:ChangePassword",
        "iam:GetUser",
        "iam:*LoginProfile*"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "DenyS3Logs",
      "Effect": "Deny",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::logs",
        "arn:aws:s3:::logs/*"
      ]
    },
    {
      "Sid": "DenyEC2Production",
      "Effect": "Deny",
      "Action": "ec2:*",
      "Resource": "arn:aws:ec2:*:*:instance/i-1234567890abcdef0"
    }
  ]
}
```

IAM：将特定标签添加到具有特定标签的用户

此示例说明了如何创建基于身份的策略以允许将带有标签值 Marketing、Development 或 QualityAssurance 的标签键 Department 添加到 IAM 用户。该用户必须已包含标签键值对 JobFunction = manager。您可以使用此策略来要求一个经理仅属于三个部门之一。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的#####替换为您自己的信息。然后，按照[创建策略](#) (p. 437)或[编辑策略](#) (p. 461)中的说明操作。

ListTagsForAllUsers 语句允许对您账户中的所有用户查看标签。

TagManagerWithSpecificDepartment 语句中的第一个条件使用 StringEquals 条件运算符。如果条件的两个部分都为 true，则此条件返回 true。要标记的用户必须已具有 JobFunction=Manager 标签。请求必须包括 Department 标签键以及其中一个列出的标签值。

第二个条件使用 ForAllValues:StringEquals 条件运算符。如果请求中的标签键与策略中的键匹配，则条件返回 true。这意味着，请求中的唯一标签键必须为 Department。有关使用 ForAllValues 的更多信息，请参阅 [创建具有多个键或值的条件 \(p. 958\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListTagsForAllUsers",
      "Effect": "Allow",
      "Action": [
        "iam:ListUserTags",
        "iam:ListUsers"
      ],
      "Resource": "*"
    },
    {
      "Sid": "TagManagerWithSpecificDepartment",
      "Effect": "Allow",
      "Action": "iam:TagUser",
      "Resource": "*",
      "Condition": { "StringEquals": {
        "iam:ResourceTag/JobFunction": "Manager",
        "aws:RequestTag/Department": [
          "Marketing",
          "Development",
          "QualityAssurance"
        ]
      } },
      "ForAllValues:StringEquals": { "aws:TagKeys": "Department" }
    }
  ]
}
```

IAM：添加具有特定值的特定标签

此示例说明了如何创建基于身份的策略以允许仅向任何 IAM 用户或角色添加标签键 CostCenter 和标签值 A-123 或标签值 B-456。您可以使用此策略将标记限制到一个特定的标签键或一组标签值。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照 [创建策略 \(p. 437\)](#) 或 [编辑策略 \(p. 461\)](#) 中的说明操作。

ConsoleDisplay 语句允许对您账户中的所有用户和角色查看标签。

AddTag 语句中的第一个条件使用 StringEquals 条件运算符。如果请求包含 CostCenter 标签键以及其中一个列出的标签值，则条件返回 true。

第二个条件使用 ForAllValues:StringEquals 条件运算符。如果请求中的标签键与策略中的键匹配，则条件返回 true。这意味着，请求中的唯一标签键必须为 CostCenter。有关使用 ForAllValues 的更多信息，请参阅 [创建具有多个键或值的条件 \(p. 958\)](#)。

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

```

        "iam:GetRole",
        "iam:GetUser",
        "iam:ListRoles",
        "iam:ListRoleTags",
        "iam:ListUsers",
        "iam:ListUserTags"
    ],
    "Resource": "*"
  },
  {
    "Sid": "AddTag",
    "Effect": "Allow",
    "Action": [
      "iam:TagUser",
      "iam:TagRole"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/CostCenter": [
          "A-123",
          "B-456"
        ]
      },
      "ForAllValues:StringEquals": {"aws:TagKeys": "CostCenter"}
    }
  }
]
}

```

IAM：仅创建具有特定标签的新用户

此示例说明了如何创建基于身份的策略以允许创建 IAM 用户，但仅限于包含 Department 和 JobFunction 标签键之一或同时包含两者。Department 标签键必须具有 Development 或 QualityAssurance 标签值。JobFunction 标签键必须具有 Employee 标签值。您可以使用此策略要求新用户具有特定工作职能和部门。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照 [创建策略 \(p. 437\)](#) 或 [编辑策略 \(p. 461\)](#) 中的说明操作。

语句中的第一个条件使用 StringEqualsIfExists 条件运算符。如果请求中存在 Department 或 JobFunction 键，则标签必须包含指定的值。如果任一键都不存在，则此条件的计算结果将为 true。该条件的计算结果为 false 的唯一方法是：其中一个指定条件键在请求中存在，但具有的值不同于允许的值。有关使用 IfExists 的更多信息，请参阅 [...IfExists 条件运算符 \(p. 957\)](#)。

第二个条件使用 ForAllValues:StringEquals 条件运算符。如果请求中指定的每个标签键都与策略中至少一个值匹配，则该条件将返回 true。这意味着，请求中的所有标签必须在此列表中。但是，请求只能包含列表中的一个标签。例如，您可以创建一个仅具有 Department=QualityAssurance 标签的 IAM 用户。但是，您不能创建一个同时具有 JobFunction=employee 标签和 Project=core 标签的 IAM 用户。有关使用 ForAllValues 的更多信息，请参阅 [创建具有多个键或值的条件 \(p. 958\)](#)。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "TagUsersWithOnlyTheseTags",
      "Effect": "Allow",
      "Action": [
        "iam:CreateUser",
        "iam:TagUser"
      ],
      "Resource": "*",
      "Condition": {

```

```
    "StringEqualsIfExists": {
      "aws:RequestTag/Department": [
        "Development",
        "QualityAssurance"
      ],
      "aws:RequestTag/JobFunction": "Employee"
    },
    "ForAllValues:StringEquals": {
      "aws:TagKeys": [
        "Department",
        "JobFunction"
      ]
    }
  }
}
```

IAM：生成和检索 IAM 凭证报告

此示例说明了如何创建基于身份的策略，以允许用户生成和下载报告，其中将列出其 Amazon 账户中的所有 IAM 用户。该报告包含用户凭证的状态，包括密码、访问密钥、MFA 设备和签名证书。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。

有关凭证报告的更多信息，请参阅[获取您 Amazon 账户的凭证报告 \(p. 141\)](#)。

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

IAM：允许以编程方式和在控制台中管理组的成员资格

此示例说明如何创建基于身份的策略以允许更新名为 MarketingTeam 组的成员资格。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的 `#####` 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

此策略有何作用？

- ViewGroups 语句允许用户列出 Amazon Web Services Management Console 中的所有用户和组。它还允许用户查看有关账户中的用户的基本信息。这些权限必须位于自己的语句中，因为它们不支持或不需要指定资源 ARN。相反，权限指定 "Resource" : "*"。
- ViewEditThisGroup 语句允许用户查看有关 MarketingTeam 组的信息，以及在该组中添加和删除用户。

该策略不允许用户查看或编辑用户或 MarketingTeam 组的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewGroups",
```



```
    "Effect": "Allow",
    "Action": [
        "iam:ListGroups",
        "iam:ListUsers",
        "iam:GetUser",
        "iam:ListGroupsForUser"
    ],
    "Resource": "*"
  },
  {
    "Sid": "ViewEditThisGroup",
    "Effect": "Allow",
    "Action": [
        "iam:AddUserToGroup",
        "iam:RemoveUserFromGroup",
        "iam:GetGroup"
    ],
    "Resource": "arn:aws:iam::*:group/MarketingTeam"
  }
]
```

IAM：管理特定标签

此示例说明如何创建基于身份的策略以允许添加和删除来自 IAM 实体（用户和角色）且带有标签键 Department 的 IAM 标签。此策略不会限制 Department 标签的值。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
        "iam:TagUser",
        "iam:TagRole",
        "iam:UntagUser",
        "iam:UntagRole"
    ],
    "Resource": "*",
    "Condition": {"ForAllValues:StringEquals": {"aws:TagKeys": "Department"}}
  }
}
```

IAM：将 IAM 角色传递给特定 Amazon 服务

此示例说明了如何创建基于身份的策略以允许将任何 IAM 服务角色传递给 Amazon CloudWatch 服务。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

服务角色是一个 IAM 角色，它将 Amazon 服务指定为可担任该角色的主体。这允许服务担任该角色，并代表您访问其他服务中的资源。要允许 Amazon CloudWatch 担任您传递的角色，必须在您的角色的信任策略中将 cloudwatch.amazonaws.com 服务主体指定为主体。服务主体由服务定义。要了解某项服务的主体，请参阅该服务的文档。对于某些服务，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)并查找服务相关角色列为是的服务。请选择 Yes 与查看该服务的主体相关角色文档的链接。请搜索 amazonaws.com 以查看服务主体。

要了解将服务角色传递给服务的更多信息，请参阅[向用户授予权限以将角色传递给 Amazon 服务 \(p. 240\)](#)。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "*",
    "Condition": {
      "StringEquals": {"iam:PassedToService": "cloudwatch.amazonaws.com"}
    }
  }
]
```

IAM：允许对 IAM 控制台进行只读访问而不生成报告

此示例说明了如何创建基于身份的策略以允许 IAM 用户执行任何以字符串 `Get` 或 `List` 开头的 IAM 操作。在用户使用控制台时，控制台向 IAM 发出请求以列出组、用户、角色和策略，并生成有关这些资源的报告。

星号将作为通配符。在策略中使用 `iam:Get*` 时，生成的权限包括以 `Get` 开头的所有 IAM 操作，例如，`GetUser` 和 `GetRole`。如果将来在 IAM 中添加新类型的实体，通配符是非常有用的。在这种情况下，策略授予的权限自动允许用户列出和获取有关这些新实体的详细信息。

此策略不能用于生成报告或服务上次访问详细信息。有关允许此操作的其他策略，请参阅 [IAM：允许对 IAM 控制台进行只读访问 \(p. 422\)](#)。

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

IAM：允许对 IAM 控制台进行只读访问

此示例说明了如何创建基于身份的策略以允许 IAM 用户执行任何以字符串 `Get`、`List` 或 `Generate` 开头的 IAM 操作。在用户使用 IAM 控制台时，控制台发出请求以列出组、用户、角色和策略，并生成有关这些资源的报告。

星号将作为通配符。在策略中使用 `iam:Get*` 时，生成的权限包括以 `Get` 开头的所有 IAM 操作，例如，`GetUser` 和 `GetRole`。使用通配符是非常有用的，特别是将来在 IAM 中添加新类型的实体。在这种情况下，策略授予的权限自动允许用户列出和获取有关这些新实体的详细信息。

将此策略用于控制台访问，其中包括生成报告或服务上次访问详细信息的权限。有关不允许生成操作的其他策略，请参阅 [IAM：允许对 IAM 控制台进行只读访问而不生成报告 \(p. 422\)](#)。

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

}

IAM：允许特定的 IAM 用户以编程方式和在控制台中管理组

此示例说明如何创建基于身份的策略以允许特定 IAM 用户管理 AllUsers 组。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

此策略有何作用？

- AllowAllUsersToListAllGroups 语句允许列出所有组。这对于控制台访问是必需的。该权限必须位于自己的语句中，因为它不支持资源 ARN。相反，权限指定 "Resource" : "*"。
- AllowAllUsersToViewAndManageThisGroup 语句允许可以对组资源类型执行的所有组操作。它不允许执行 ListGroupsForUser 操作，可以对用户资源类型执行该操作，但不能对组资源类型执行该操作。有关可以为 IAM 操作指定的资源类型的更多信息，请参阅 [Amazon Identity and Access Management 的操作、资源和条件键](#)。
- LimitGroupManagementAccessToSpecificUsers 语句拒绝具有指定名称的用户访问以执行写入和权限管理组操作。如果策略中指定的用户尝试对组进行更改，该语句不会拒绝请求。AllowAllUsersToViewAndManageThisGroup 语句允许该请求。如果其他用户尝试执行这些操作，则会拒绝该请求。在 IAM 控制台中创建该策略时，您可以查看使用写入或权限管理访问级别定义的 IAM 操作。为此，请从 JSON 选项卡切换到可视化编辑器选项卡。有关访问级别的更多信息，请参阅 [Amazon Identity and Access Management 的操作、资源和条件键](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAllUsersToListAllGroups",
      "Effect": "Allow",
      "Action": "iam:ListGroups",
      "Resource": "*"
    },
    {
      "Sid": "AllowAllUsersToViewAndManageThisGroup",
      "Effect": "Allow",
      "Action": "iam:*Group*",
      "Resource": "arn:aws:iam::*:group/AllUsers"
    },
    {
      "Sid": "LimitGroupManagementAccessToSpecificUsers",
      "Effect": "Deny",
      "Action": [
        "iam:AddUserToGroup",
        "iam:CreateGroup",
        "iam:RemoveUserFromGroup",
        "iam>DeleteGroup",
        "iam:AttachGroupPolicy",
        "iam:UpdateGroup",
        "iam:DetachGroupPolicy",
        "iam>DeleteGroupPolicy",
        "iam:PutGroupPolicy"
      ],
      "Resource": "arn:aws:iam::*:group/AllUsers",
      "Condition": {
        "StringNotEquals": {
          "aws:username": [
            "srodriguez",
            "mjackson",
            "adesai"
          ]
        }
      }
    }
  ]
}
```

```
}
    }
  }
]
```

IAM：允许以编程方式和在控制台中设置账户密码要求

此示例说明了如何创建基于身份的策略以允许用户查看和更新其账户密码要求。密码要求指定账户成员密码的复杂性要求和强制轮换期。此策略定义了程序访问和控制台访问的权限。

要了解如何为您的账户设置账户密码要求策略，请参阅 [IAM 用户设置账户密码策略 \(p. 88\)](#)。

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

IAM：基于用户路径访问策略模拟器 API

此示例说明了如何创建基于身份的策略以仅允许具有路径 `Department/Development` 的用户使用策略模拟器 API。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 `#####` 替换为您自己的信息。然后，按照 [创建策略 \(p. 437\)](#) 或 [编辑策略 \(p. 461\)](#) 中的说明操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:GetContextKeysForPrincipalPolicy",
        "iam:SimulatePrincipalPolicy"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:user/Department/Development/*"
    }
  ]
}
```

Note

要创建允许具有路径 `Department/Development` 的用户使用策略模拟器控制台的策略，请参阅 [IAM：基于用户路径访问策略模拟器控制台 \(p. 424\)](#)。

IAM：基于用户路径访问策略模拟器控制台

此示例说明了如何创建基于身份的策略以仅允许具有路径 `Department/Development` 的用户使用策略模拟器控制台。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 `#####` 替换为您自己的信息。然后，按照 [创建策略 \(p. 437\)](#) 或 [编辑策略 \(p. 461\)](#) 中的说明操作。

您可以在以下位置访问 IAM policy simulator：<https://policysim.aws.amazon.com/>

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:GetPolicy",
        "iam:GetUserPolicy"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": [
        "iam:GetUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListGroupsForUser",
        "iam:ListUserPolicies",
        "iam:ListUsers"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:user/Department/Development/*"
    }
  ]
}
```

IAM : 允许 IAM 用户自行管理 MFA 设备

此示例说明如何创建 IAM policy 以允许基于身份的用户自行管理其[多重身份验证 \(MFA\)](#) (p. 104)设备。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。

Note

如果具有此策略的 IAM 用户未通过 MFA 进行身份验证，则此策略会拒绝对所有 Amazon 操作（使用 MFA 进行身份验证所需的操作除外）的访问。如果为已登录 Amazon 的用户添加这些权限，他们需要注销并重新登录才能看到这些更改。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListActions",
      "Effect": "Allow",
      "Action": [
        "iam:ListUsers",
        "iam:ListVirtualMFADevices"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowIndividualUserToManageTheirOwnMFA",
      "Effect": "Allow",
      "Action": [
        "iam:CreateVirtualMFADevice",
        "iam:DeleteVirtualMFADevice",
        "iam:ListMFADevices",
        "iam:EnableMFADevice",
        "iam:ResyncMFADevice"
      ],
      "Resource": [
        "arn:aws:iam::*:mfa/${aws:username}",
        "arn:aws:iam::*:user/${aws:username}"
      ]
    }
  ],
}
```

```
{
  "Sid": "AllowIndividualUserToDeactivateOnlyTheirOwnMFAOnlyWhenUsingMFA",
  "Effect": "Allow",
  "Action": [
    "iam:DeactivateMFADevice"
  ],
  "Resource": [
    "arn:aws:iam::*:mfa/${aws:username}",
    "arn:aws:iam::*:user/${aws:username}"
  ],
  "Condition": {
    "Bool": {
      "aws:MultiFactorAuthPresent": "true"
    }
  }
},
{
  "Sid": "BlockMostAccessUnlessSignedInWithMFA",
  "Effect": "Deny",
  "NotAction": [
    "iam:CreateVirtualMFADevice",
    "iam:EnableMFADevice",
    "iam:ListMFADevices",
    "iam:ListUsers",
    "iam:ListVirtualMFADevices",
    "iam:ResyncMFADevice"
  ],
  "Resource": "*",
  "Condition": {
    "BoolIfExists": {
      "aws:MultiFactorAuthPresent": "false"
    }
  }
}
]
```

IAM：允许 IAM 用户以编程方式和在控制台中轮换其凭证

此示例说明您如何创建基于身份的策略以允许 IAM 用户轮换自己的访问密钥、签名证书、特定于服务的凭证和密码。此策略定义了程序访问和控制台访问的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:ListUsers",
        "iam:GetAccountPasswordPolicy"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:*AccessKey*",
        "iam:ChangePassword",
        "iam:GetUser",
        "iam:*ServiceSpecificCredential*",
        "iam:*SigningCertificate*"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    }
  ]
}
```



```
}  
]
```

要了解用户如何在控制台中更改自己的密码，请参阅[the section called “IAM 用户如何更改自己的密码” \(p. 95\)](#)。

IAM：查看 Organizations 策略的上次访问的服务信息

此示例说明了如何创建基于身份的策略以允许查看特定 Organizations 策略的服务上次访问信息。该策略允许检索具有 p-policy123 ID 的服务控制策略 (SCP) 的数据。必须使用 Amazon Organizations 管理账户凭证对生成和查看报告的人员进行身份验证。该策略允许请求者检索其企业中的任何 Organizations 实体的数据。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

有关上次访问的信息的重要信息（包括所需的权限、故障排除和支持的区域），请参阅[使用上次访问的信息优化 Amazon 中的权限 \(p. 467\)](#)。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowOrgsReadOnlyAndIamGetReport",  
      "Effect": "Allow",  
      "Action": [  
        "iam:GetOrganizationsAccessReport",  
        "organizations:Describe*",  
        "organizations:List*"  
      ],  
      "Resource": "*"  
    },  
    {  
      "Sid": "AllowGenerateReportOnlyForThePolicy",  
      "Effect": "Allow",  
      "Action": "iam:GenerateOrganizationsAccessReport",  
      "Resource": "*",  
      "Condition": {  
        "StringEquals": {"iam:OrganizationsPolicyId": "p-policy123"}  
      }  
    }  
  ]  
}
```

IAM：对可应用于 IAM 用户、组或角色的托管策略加以限制

此示例说明了如何创建基于身份的策略以限制客户托管和 Amazon 托管的策略，这些策略将应用于 IAM 用户、组或角色。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

```
{  
  "Version": "2012-10-17",  
  "Statement": {  
    "Effect": "Allow",  
    "Action": [  
      "iam:AttachUserPolicy",  
      "iam:DetachUserPolicy"  
    ],  
    "Resource": "*",  
    "Condition": {  
      "ArnEquals": {
```

```

        "iam:PolicyARN": [
            "arn:aws:iam::*:policy/policy-name-1",
            "arn:aws:iam::*:policy/policy-name-2"
        ]
    }
}

```

Amazon Lambda：允许 Lambda 函数访问 Amazon DynamoDB 表

此示例说明了如何创建基于身份的策略以允许对特定 Amazon DynamoDB 表进行读写访问。该策略还允许将日志文件写入到 CloudWatch Logs 中。要使用此策略，请将示例策略中的 **#####** 替换为您自己的信息。然后，按照 [创建策略 \(p. 437\)](#) 或 [编辑策略 \(p. 461\)](#) 中的说明操作。

要使用该策略，请将策略附加到 Lambda [服务角色 \(p. 219\)](#)。服务角色是在您的账户中创建的一个角色，以允许服务代表您执行操作。该服务角色必须在信任策略中包含 Amazon Lambda 以作为主体。有关如何使用此策略的详细信息，请参阅 Amazon 安全博客中的 [如何创建 Amazon IAM policy 以授予 Amazon Lambda 访问 Amazon DynamoDB 表的权限](#)。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadWriteTable",
      "Effect": "Allow",
      "Action": [
        "dynamodb:BatchGetItem",
        "dynamodb:GetItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:BatchWriteItem",
        "dynamodb:PutItem",
        "dynamodb:UpdateItem"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/SampleTable"
    },
    {
      "Sid": "GetStreamRecords",
      "Effect": "Allow",
      "Action": "dynamodb:GetRecords",
      "Resource": "arn:aws:dynamodb:*:*:table/SampleTable/stream/* "
    },
    {
      "Sid": "WriteLogStreamsAndGroups",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CreateLogGroup",
      "Effect": "Allow",
      "Action": "logs:CreateLogGroup",
      "Resource": "*"
    }
  ]
}

```

Amazon RDS：允许在特定区域中进行完全 RDS 数据库访问。

此示例说明了如何创建基于身份的策略以允许在特定区域中进行完全 RDS 数据库访问。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的#####替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "rds:*",
      "Resource": ["arn:aws:rds:region:*:*"]
    },
    {
      "Effect": "Allow",
      "Action": ["rds:Describe*"],
      "Resource": ["*"]
    }
  ]
}
```

Amazon RDS：允许以编程方式和在控制台中还原 RDS 数据库

此示例说明了如何创建允许还原 RDS 数据库的基于身份的策略。此策略定义了程序访问和控制台访问的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:Describe*",
        "rds:CreateDBParameterGroup",
        "rds:CreateDBSnapshot",
        "rds>DeleteDBSnapshot",
        "rds:Describe*",
        "rds:DownloadDBLogFilePortion",
        "rds:List*",
        "rds:ModifyDBInstance",
        "rds:ModifyDBParameterGroup",
        "rds:ModifyOptionGroup",
        "rds:RebootDBInstance",
        "rds:RestoreDBInstanceFromDBSnapshot",
        "rds:RestoreDBInstanceToPointInTime"
      ],
      "Resource": "*"
    }
  ]
}
```

Amazon RDS：允许标签所有者不受限制地访问其标记的 RDS 资源

此示例显示您可以创建一个基于身份的策略，对标签所有者授予对已标记的 RDS 资源的完全访问权限。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Action": [
      "rds:Describe*",
      "rds:List*"
    ],
    "Effect": "Allow",
    "Resource": "*"
  },
  {
    "Action": [
      "rds>DeleteDBInstance",
      "rds:RebootDBInstance",
      "rds:ModifyDBInstance"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
      "StringEqualsIgnoreCase": {"rds:db-tag/Owner": "${aws:username}"}
    }
  },
  {
    "Action": [
      "rds:ModifyOptionGroup",
      "rds>DeleteOptionGroup"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
      "StringEqualsIgnoreCase": {"rds:og-tag/Owner": "${aws:username}"}
    }
  },
  {
    "Action": [
      "rds:ModifyDBParameterGroup",
      "rds:ResetDBParameterGroup"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
      "StringEqualsIgnoreCase": {"rds:pg-tag/Owner": "${aws:username}"}
    }
  },
  {
    "Action": [
      "rds:AuthorizeDBSecurityGroupIngress",
      "rds:RevokeDBSecurityGroupIngress",
      "rds>DeleteDBSecurityGroup"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
      "StringEqualsIgnoreCase": {"rds:secgrp-tag/Owner": "${aws:username}"}
    }
  },
  {
    "Action": [
      "rds>DeleteDBSnapshot",
      "rds:RestoreDBInstanceFromDBSnapshot"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
      "StringEqualsIgnoreCase": {"rds:snapshot-tag/Owner": "${aws:username}"}
    }
  }
]

```

```

    },
    {
      "Action": [
        "rds:ModifyDBSubnetGroup",
        "rds>DeleteDBSubnetGroup"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringEqualsIgnoreCase": {"rds:subgrp-tag/Owner": "${aws:username}"}
      }
    },
    {
      "Action": [
        "rds:ModifyEventSubscription",
        "rds:AddSourceIdentifierToSubscription",
        "rds:RemoveSourceIdentifierFromSubscription",
        "rds>DeleteEventSubscription"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringEqualsIgnoreCase": {"rds:es-tag/Owner": "${aws:username}"}
      }
    }
  ]
}

```

Amazon S3：允许 Amazon Cognito 用户访问其存储桶中的对象

此示例说明了如何创建基于身份的策略以允许 Amazon Cognito 用户访问特定 S3 存储桶中的对象。该策略仅允许访问名称包含 `cognito`、应用程序名称以及联合身份用户 ID 的对象，由 `${cognito-identity.amazonaws.com:sub}` 变量表示。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 `#####` 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

Note

对象键中使用的“sub”值不是用户池中的 `sub` 值，而是与身份池中的用户关联的身份 ID。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListYourObjects",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": [
        "arn:aws:s3:::bucket-name"
      ],
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "cognito/application-name/${cognito-identity.amazonaws.com:sub}/*"
          ]
        }
      }
    },
    {
      "Sid": "ReadWriteDeleteYourObjects",
      "Effect": "Allow",
      "Action": [
        "s3>DeleteObject",
        "s3:GetObject",

```

```
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name/cognito/application-name/${cognito-identity.amazonaws.com:sub}/*"
      ]
    }
  ]
}
```

Amazon Cognito 为您的 Web 和移动应用程序提供身份验证、授权和用户管理。您的用户可使用用户名和密码直接登录，也可以通过第三方 (如 Facebook、Amazon 或 Google) 登录。

Amazon Cognito 的两个主要组件是用户池和身份池。用户池是为您的应用程序提供注册和登录选项的用户目录。使用身份池，您可以授予用户访问其他 Amazon 服务的权限。您可以单独或配合使用身份池和用户池。

有关 Amazon Cognito 身份池的更多信息，请参阅以下内容：

- Amazon Mobile SDK for Android 开发人员指南中的 [Amazon Cognito 身份](#)
- Amazon Mobile SDK for iOS 开发人员指南中的 [Amazon Cognito 身份](#)

Amazon S3：允许联合身份用户以编程方式和在控制台中访问其 S3 主目录

此示例说明了如何创建基于身份的策略以允许联合用户访问其位于 S3 中的主目录存储桶对象。主目录是一个包含 home 文件夹和个人联合用户文件夹的存储桶。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的 **#####** 替换为您自己的信息。然后，按照 [创建策略 \(p. 437\)](#) 或 [编辑策略 \(p. 461\)](#) 中的说明操作。

此策略中的 `${aws:userid}` 变量解析为 `role-id:specified-name`。联合身份用户 ID 的 `role-id` 部分是创建过程中分配给联合身份用户的唯一标识符。有关更多信息，请参阅 [唯一标识符 \(p. 900\)](#)。specified-name 是当联合身份用户担任其角色时传递给 AssumeRoleWithWebIdentity 请求的 [RoleSessionName](#) 参数。

您可使用 Amazon CLI 命令 `aws iam get-role --role-name specified-name` 查看此角色 ID。例如，假设您指定友好名称 John 并且 CLI 返回角色 ID `AROAXXT2NJT7D3SIQN7Z6`。在此示例中，联合身份用户 ID 为 `AROAXXT2NJT7D3SIQN7Z6:John`。此策略之后允许联合身份用户 John 访问具有前缀 `AROAXXT2NJT7D3SIQN7Z6:John` 的 Amazon S3 存储桶。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3ConsoleAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetAccountPublicAccessBlock",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation",
        "s3:GetBucketPolicyStatus",
        "s3:GetBucketPublicAccessBlock",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ListObjectsInBucket",
      "Effect": "Allow",

```



```

    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::bucket-name",
    "Condition": {
      "StringLike": {
        "s3:prefix": [
          "",
          "home/",
          "home/${aws:userid}/*"
        ]
      }
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::bucket-name/home/${aws:userid}",
        "arn:aws:s3:::bucket-name/home/${aws:userid}/*"
      ]
    }
  ]
}

```

Amazon S3 : S3 存储桶访问，但在最近未进行 MFA 的情况下拒绝生产存储桶

此示例说明如何创建基于身份的策略以允许 Amazon S3 管理员访问任何存储桶（包括更新、添加和删除对象）。不过，如果用户在过去 30 分钟内未使用多重身份验证 (MFA)??? (p. 104) 登录，则显式拒绝访问 Production 存储桶。该策略授予所需的权限以在控制台中执行该操作，或者以编程方式使用 Amazon CLI 或 Amazon API 执行该操作。要使用此策略，请将示例策略中的#####替换为您自己的信息。然后，按照[创建策略](#) (p. 437)或[编辑策略](#) (p. 461)中的说明操作。

该策略从不允许使用长期用户访问密钥以编程方式访问 Production 存储桶。这是将 `aws:MultiFactorAuthAge` 条件键与 `NumericGreaterThanIfExists` 条件运算符结合使用来实现的。如果 MFA 不存在或 MFA 的使用期限超过 30 分钟，则此策略条件将返回 `true`。在这些情况下，访问将被拒绝。要以编程方式访问 Production 存储桶，S3 管理员必须使用通过 [GetSessionToken](#) (p. 308) API 操作在过去 30 分钟内生成的临时凭证。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListAllS3Buckets",
      "Effect": "Allow",
      "Action": ["s3:ListAllMyBuckets"],
      "Resource": "arn:aws:s3:::"
    },
    {
      "Sid": "AllowBucketLevelActions",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::"
    },
    {
      "Sid": "AllowBucketObjectActions",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl",

```

```

        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:DeleteObject"
    ],
    "Resource": "arn:aws:s3:::*/*"
  },
  {
    "Sid": "RequireMFAForProductionBucket",
    "Effect": "Deny",
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::Production/*",
      "arn:aws:s3:::Production"
    ],
    "Condition": {
      "NumericGreaterThanIfExists": {"aws:MultiFactorAuthAge": "1800"}
    }
  }
]
}

```

Amazon S3：允许 IAM 用户以编程方式和在控制台中访问其 S3 主目录

此示例说明了如何创建基于身份的策略以允许 IAM 用户访问其位于 S3 中的主目录存储桶对象。主目录是一个包含 home 文件夹和个人用户文件夹的存储桶。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的 **#####** 替换为您自己的信息。然后，按照[创建策略 \(p. 437\)](#)或[编辑策略 \(p. 461\)](#)中的说明操作。

使用 IAM 角色时，此策略将不起作用，因为使用角色 IAM 时 `aws:username` 变量不可用。有关主体键值的详细信息，请参阅[主体键值 \(p. 969\)](#)。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3ConsoleAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetAccountPublicAccessBlock",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation",
        "s3:GetBucketPolicyStatus",
        "s3:GetBucketPublicAccessBlock",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ListObjectsInBucket",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::bucket-name",
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "",
            "home/",
            "home/${aws:username}/*"
          ]
        }
      }
    }
  ]
}

```

```

    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::bucket-name/home/${aws:username}",
        "arn:aws:s3:::bucket-name/home/${aws:username}/*"
      ]
    }
  ]
}

```

Amazon S3：将管理限制为特定 S3 存储桶

此示例说明了如何创建基于身份的策略以限制 Amazon S3 存储桶对该特定存储桶的管理。此策略授予执行所有 Amazon S3 操作的权限，但拒绝访问除 Amazon S3 外的所有 Amazon Web Service。请参阅以下示例。根据此策略，您只能访问可以对 S3 存储桶或 S3 对象资源执行的 Amazon S3 操作。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略](#) (p. 437) 或[编辑策略](#) (p. 461) 中的说明操作。

如果将该策略与允许该策略拒绝的操作的其他策略（例如 [AmazonS3FullAccess](#) 或 [AmazonEC2FullAccess](#) Amazon 托管策略）一起使用，则访问会被拒绝。这是因为“显式拒绝”声明优先于“允许”声明。有关更多信息，请参阅[the section called “确定是允许还是拒绝账户内的请求”](#) (p. 974)。

Warning

[NotAction](#) (p. 945) 和 [NotResource](#) (p. 948) 是必须谨慎使用的高级策略元素。该策略将拒绝对除 Amazon S3 以外所有 Amazon 服务的访问。如果将该策略挂载到用户，则授予其他服务访问权限的任何其他策略都会被忽略（访问会被拒绝）。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::bucket-name",
        "arn:aws:s3:::bucket-name/*"
      ]
    },
    {
      "Effect": "Deny",
      "NotAction": "s3:*",
      "NotResource": [
        "arn:aws:s3:::bucket-name",
        "arn:aws:s3:::bucket-name/*"
      ]
    }
  ]
}

```

Amazon S3：允许对 S3 存储桶中的对象进行读写访问

此示例说明如何创建基于身份的策略以允许 Read 和 Write 对特定 S3 存储桶中的对象进行访问。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照[创建策略](#) (p. 437) 或[编辑策略](#) (p. 461) 中的说明操作。

s3:*Object 操作使用通配符作为操作名称的一部分。AllObjectActions 语句允许以单词“Object”结尾的 GetObject、DeleteObject、PutObject 和任何其他 Amazon S3 操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListObjectsInBucket",
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::bucket-name"]
    },
    {
      "Sid": "AllObjectActions",
      "Effect": "Allow",
      "Action": "s3:*Object",
      "Resource": ["arn:aws:s3:::bucket-name/*"]
    }
  ]
}
```

Note

要允许对 Amazon S3 存储桶中的对象进行 Read 和 Write 访问并包含用于控制台访问的其他权限，请参阅 [Amazon S3：允许以编程方式和在控制台中对 S3 存储桶中的对象进行读写访问](#) (p. 436)。

Amazon S3：允许以编程方式和在控制台中对 S3 存储桶中的对象进行读写访问

此示例说明如何创建基于身份的策略以允许 Read 和 Write 对特定 S3 存储桶中的对象进行访问。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的 ##### 替换为您自己的信息。然后，按照 [创建策略](#) (p. 437) 或 [编辑策略](#) (p. 461) 中的说明操作。

s3:*Object 操作使用通配符作为操作名称的一部分。AllObjectActions 语句允许以单词“Object”结尾的 GetObject、DeleteObject、PutObject 和任何其他 Amazon S3 操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3ConsoleAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetAccountPublicAccessBlock",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation",
        "s3:GetBucketPolicyStatus",
        "s3:GetBucketPublicAccessBlock",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ListObjectsInBucket",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": ["arn:aws:s3:::bucket-name"]
    },
    {
      "Sid": "AllObjectActions",
      "Effect": "Allow",
      "Action": "s3:*Object",
    }
  ]
}
```

```
        "Resource": [ "arn:aws:s3:::bucket-name/*" ]
    }
  ]
}
```

管理 IAM 策略

IAM 为您提供了相应的工具以创建和管理所有类型的 IAM 策略（托管策略和内联策略）。要向 IAM 身份（IAM 用户、组或角色）添加权限，请创建策略、验证策略，然后将该策略附加到身份。您可以将多个策略附加到一个身份，每个策略可以包含多个权限。

有关详细信息，请参阅以下资源：

- 有关不同类型的 IAM 策略的更多信息，请参阅 [IAM 中的策略和权限](#) (p. 355)。
- 有关使用 IAM 中的策略的一般信息，请参阅 [适用于 Amazon 资源的 Access Management](#) (p. 353)。
- 有关在多个策略对给定 IAM 身份有效时如何评估权限的信息，请参阅 [策略评估逻辑](#) (p. 972)。
- Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 [IAM 与 Amazon STS 配额、名称要求和字符限制](#) (p. 902)。

主题

- [创建 IAM policy](#) (p. 437)
- [验证 IAM 策略](#) (p. 443)
- [基于访问活动生成策略](#) (p. 443)
- [使用 IAM policy simulator 测试 IAM policy](#) (p. 443)
- [添加和删除 IAM 身份权限](#) (p. 451)
- [IAM 策略版本控制](#) (p. 458)
- [编辑 IAM 策略](#) (p. 461)
- [删除 IAM 策略](#) (p. 465)
- [使用上次访问的信息优化 Amazon 中的权限](#) (p. 467)

创建 IAM policy

[策略](#) (p. 355) 是一个实体；在附加到身份或资源时，策略定义了它们的权限。您可以使用 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 在 IAM 中创建客户托管策略。客户托管策略是您在自己的 Amazon 账户中管理的独立策略。然后，您可以将策略附加到 Amazon 账户中的身份（用户、组和角色）。

附加到 IAM 中的身份的策略称为基于身份的策略。基于身份的策略可以包括 Amazon 托管策略、客户托管策略和内联策略。Amazon 托管策略由 Amazon 创建和管理。您可以使用它们，但无法管理它们。内联策略是您创建并直接嵌入到 IAM 组、用户或角色的策略。内联策略不能在其他身份上重复使用，也不能在它们所存在的身份之外托管。有关更多信息，请参阅 [添加和删除 IAM 身份权限](#) (p. 451)。

使用客户管理型策略而不是内联策略。此外，最好使用客户托管策略而非 Amazon 托管策略。Amazon 托管策略通常提供广泛的管理权限或只读权限。为了获得最高安全性，[请授予最低特权](#) (p. 754)，即仅授予执行特定作业任务所需的权限。

创建或编辑 IAM policy 时，Amazon 可以自动执行策略验证，以帮助您创建具有最低权限的有效策略。在 Amazon Web Services Management Console 中，IAM 可识别 JSON 语法错误，而 IAM Access Analyzer 提供额外的策略检查和建议，以帮助您进一步优化策略。要了解策略验证的更多信息，请参阅 [验证 IAM 策略](#) (p. 443)。要了解有关 IAM Access Analyzer 策略检查和可执行建议的更多信息，请参阅 [IAM Access Analyzer 策略验证](#)。

您可以使用 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 在 IAM 中创建客户托管策略。

创建 IAM 策略 (控制台)

[策略 \(p. 355\)](#) 是一个实体；在附加到身份或资源时，策略定义了它们的权限。您可以使用 Amazon Web Services Management Console 在 IAM 中创建客户托管策略。客户托管策略是您在自己的 Amazon 账户中管理的独立策略。然后，您可以将策略附加到 Amazon 账户中的身份（用户、组和角色）。

主题

- [创建 IAM 策略 \(p. 438\)](#)
- [在“JSON”选项卡上创建策略 \(p. 438\)](#)
- [使用可视化编辑器创建策略 \(p. 439\)](#)
- [导入现有托管策略 \(p. 440\)](#)

创建 IAM 策略

您可以使用下列方法之一在 Amazon Web Services Management Console 中创建客户托管策略：

- [JSON \(p. 438\)](#) - 粘贴和自定义已发布的[基于身份的示例策略 \(p. 391\)](#)。
- [Visual editor \(p. 439\)](#) (可视化编辑器) - 在可视化编辑器中从头开始构建新的策略。如果使用可视化编辑器，您不需要了解 JSON 语法。
- [Import \(p. 440\)](#) (导入) - 从您的账户中导入并自定义托管策略。您可以导入之前创建的 Amazon 托管策略或客户托管策略。

Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅[IAM 与 Amazon STS 配额、名称要求和字符限制 \(p. 902\)](#)。

在“JSON”选项卡上创建策略

您可以选择 JSON 选项卡以在 JSON 中键入或粘贴策略。这种方法适用于复制[示例策略 \(p. 391\)](#)以在您的账户中使用。或者，您可以在 JSON 编辑器中键入自己的 JSON 策略文档。也可以使用 JSON 选项卡在可视化编辑器和 JSON 之间切换以比较这些视图。

当您在 JSON 编辑器中创建或编辑策略时，IAM 会执行策略验证以帮助您创建有效的策略。IAM 可识别 JSON 语法错误，而 IAM Access Analyzer 提供额外的策略检查和可执行的建议，以帮助您进一步优化策略。

[JSON 策略 \(p. 355\)](#) 文档包含一个或多个语句。每个语句应包含具有相同效果 (Allow 或 Deny) 并支持相同资源和条件的所有操作。如果一个操作要求指定所有资源 ("*")，而另一个操作支持特定资源的 Amazon Resource Name (ARN)，则它们必须位于两个单独的 JSON 语句中。有关 ARN 格式的详细信息，请参阅 Amazon 一般参考指南中的 [Amazon Resource Name \(ARN\)](#)。有关 IAM 策略的一般信息，请参阅 [IAM 中的策略和权限 \(p. 355\)](#)。有关 IAM 策略语言的信息，请参阅 [IAM JSON 策略参考 \(p. 933\)](#)。

使用 JSON 策略编辑器创建策略

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在左侧的导航窗格中，选择策略。
3. 选择 Create policy (创建策略)。
4. 请选择 JSON 选项卡。
5. 键入或粘贴一个 JSON 策略文档。有关 IAM 策略语言的详细信息，请参阅 [IAM JSON 策略参考 \(p. 933\)](#)。

6. 解决[策略验证 \(p. 443\)](#)过程中生成的任何安全警告、错误或常规警告，然后选择 Review policy (查看策略)。

Note

您可以随时在 Visual editor (可视化编辑器) 和 JSON 选项卡之间切换。不过，如果您进行更改或在 Visual editor (可视化编辑器) 选项卡中选择 Next: Tags (下一步：标签)，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅[调整策略结构 \(p. 871\)](#)。

7. 完成后，选择 Next: Tags (下一步：标签)。
(可选) 通过以键值对的形式附加标签来向策略添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 [标记 IAM 资源 \(p. 276\)](#)。
8. 在查看策略页上，为创建的策略键入名称和说明 (可选)。查看策略 Summary (摘要) 以查看您的策略授予的权限。然后，选择创建策略以保存您的工作。

在创建策略后，可以将其附加到您的组、用户或角色。有关更多信息，请参阅[添加和删除 IAM 身份权限 \(p. 451\)](#)。

使用可视化编辑器创建策略

IAM 控制台中的可视化编辑器指导您完成创建策略的过程，而无需编写 JSON 语法。要查看使用可视化编辑器创建策略的示例，请参阅[the section called “控制对身份的访问” \(p. 383\)](#)。

使用可视化编辑器创建策略

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在左侧的导航窗格中，选择策略。
3. 选择 Create policy (创建策略)。
4. 在可视化编辑器选项卡上，选择选择服务，然后选择 Amazon 服务。您可以使用顶部的搜索框限制服务列表中的结果。您只能在一个可视化编辑器权限块中选择一个服务。要为多个服务授予访问权限，请选择添加额外的权限以添加多个权限块。
5. 对于操作，选择要添加到策略的操作。您可以使用以下方法选择操作：
 - 选中所有操作的复选框。
 - 选择添加操作以键入特定操作的名称。您可以使用通配符 (*) 指定多个操作。
 - 选择访问级别组之一以选择访问级别的所有操作 (例如，读取、写入或列出)。
 - 展开每个访问级别组以选择单独的操作。

预设情况下，您创建的策略允许执行选择的操作。要拒绝选择的操作，请选择切换到拒绝权限。由于 [IAM 默认拒绝 \(p. 972\)](#)，作为安全最佳实践，我们建议您仅允许用户所需的操作和资源的权限。只有在要覆盖由其他语句或策略单独允许的权限时，您才应创建 JSON 语句以拒绝权限。我们建议您将拒绝权限数限制为最低，因为它们可能会增加解决权限问题的难度。

6. 对于资源，如果您在前面步骤中选择的服务和操作不支持选择[特定资源 \(p. 387\)](#)，则允许使用所有资源，并且您无法编辑此部分。

如果您选择了一个或多个操作支持[资源级权限 \(p. 387\)](#)，可视化编辑器会列出这些资源。然后，展开资源以指定您的策略的资源。

您可以通过以下方式指定资源：

- 选择添加 ARN，通过 Amazon 资源名称 (ARN) 指定资源。您可以使用可视化 ARN 编辑器或手动列出 ARN。有关 ARN 语法的更多信息，请参阅 Amazon 一般参考指南中的 [Amazon Resource](#)

Name (ARN)。有关在策略的 Resource 元素中使用 ARN 的信息，请参阅[IAM JSON 策略元素：Resource \(p. 946\)](#)。

- 选择资源旁边的任意以授予对该类型的任何资源的权限。
 - 选择所有资源以选择该服务的所有资源。
7. (可选) 选择指定请求条件 (可选) 以在创建的策略中添加条件。条件限制 JSON 策略语句的效果。例如，您可以指定仅在以下情况下允许用户对资源执行操作：该用户的请求发生在特定的时间范围内。您还可以使用常用的条件来限制是否必须使用多重验证 (MFA) 设备来验证用户身份。或者，您可以要求请求来自特定范围的 IP 地址。有关可以在策略条件中使用的所有上下文密钥的列表，请参阅[Amazon 服务的操作、资源和条件键](#)。

您可以使用以下方法选择条件：

- 使用复选框选择常用的条件。
- 选择添加条件以指定其他条件。选择条件的条件键、限定词和运算符，然后键入一个值。要添加多个值，请选择添加新的值。您可以将这些值视为通过逻辑“OR”运算符连接在一起。完成后，选择添加。

要添加多个条件，请再次选择添加条件。根据需要重复上述步骤。每个条件仅适用于该可视化编辑器权限块。所有条件都必须为 true，才会将权限块视为匹配项。也就是说，将条件视为通过逻辑“AND”运算符连接在一起。

有关 Condition 元素的更多信息，请参阅[IAM JSON 策略元素：Condition \(p. 948\)](#)中的 [IAM JSON 策略参考 \(p. 933\)](#)。

8. 要添加更多权限块，请选择添加额外的权限。对于每个块，重复步骤 2 到步骤 5。

Note

您可以随时在 Visual editor (可视化编辑器) 和 JSON 选项卡之间切换。不过，如果您进行更改或在 Visual editor (可视化编辑器) 选项卡中选择 Next: Tags (下一步：标签)，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅[调整策略结构 \(p. 871\)](#)。

9. 完成后，选择 Next: Tags (下一步：标签)。

(可选) 通过以键值对的形式附加标签来向策略添加元数据。有关在 IAM 中使用标签的更多信息，请参阅[标记 IAM 资源 \(p. 276\)](#)。

10. 完成后，选择 Next: Review。

11. 在查看策略页上，为创建的策略键入名称和说明 (可选)。查看策略摘要以确保您授予了所需的权限，然后选择创建策略以保存新策略。

在创建策略后，可以将其附加到您的组、用户或角色。有关更多信息，请参阅[添加和删除 IAM 身份权限 \(p. 451\)](#)。

导入现有托管策略

要创建新的策略，一种简单的方法是在您的账户中导入至少具有一部分所需权限的现有托管策略。您随后可以自定义该策略，使其符合您的新要求。

您无法导入内联策略。要了解托管策略和内联策略之间的差别，请参阅[托管策略与内联策略 \(p. 364\)](#)。

在可视化编辑器中导入现有的托管策略

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在左侧的导航窗格中，选择策略。
3. 选择 Create policy (创建策略)。
4. 选择可视化编辑器选项卡，然后在页面右侧选择导入托管策略。

5. 在导入托管策略窗口中，选择与要在新策略中包含的策略最匹配的托管策略。您可以使用筛选器菜单或在顶部搜索框中键入内容以限制策略列表中的结果。
6. 选择 Import (导入)。

将在策略底部的新权限块中添加导入的策略。

7. 使用可视化编辑器或选择 JSON 以自定义您的策略。然后，选择查看策略。

Note

您可以随时在 Visual editor (可视化编辑器) 和 JSON 选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy (查看策略)，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅[调整策略结构 \(p. 871\)](#)。

8. 在审核页上，为创建的策略键入名称和说明 (可选)。您以后无法编辑这些设置。查看策略摘要，然后选择创建策略以保存您的工作。

在 JSON 选项卡中导入现有的托管策略

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在左侧的导航窗格中，选择策略。
3. 选择 Create policy (创建策略)。
4. 选择 JSON 选项卡，然后在页面右侧选择导入托管策略。
5. 在导入托管策略窗口中，选择与要在新策略中包含的策略最匹配的托管策略。您可以使用筛选器菜单或在顶部搜索框中键入内容以限制策略列表中的结果。
6. 选择 Import (导入)。

导入的策略中的语句将添加到 JSON 策略底部。

7. 在 JSON 中自定义您的策略。解决[策略验证 \(p. 443\)](#)过程中生成的任何安全警告、错误或常规警告，然后选择 Review policy (查看策略)。或者，在 Visual editor (可视化编辑器) 中自定义您的策略。然后，选择查看策略。

Note

您可以随时在 Visual editor (可视化编辑器) 和 JSON 选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy (查看策略)，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅[调整策略结构 \(p. 871\)](#)。

8. 在查看策略页上，为创建的策略键入名称和说明 (可选)。您以后无法编辑这些字段。查看策略摘要，然后选择创建策略以保存您的工作。

在创建策略后，可以将其附加到您的组、用户或角色。有关更多信息，请参阅[添加和删除 IAM 身份权限 \(p. 451\)](#)。

创建 IAM 策略 (Amazon CLI)

[策略 \(p. 355\)](#)是一个实体；在附加到身份或资源时，策略定义了它们的权限。您可以使用 Amazon CLI 在 IAM 中创建客户托管策略。客户托管策略是您在自己的 Amazon 账户中管理的独立策略。作为最佳实践，我们建议您采取额外步骤以[验证您的策略 \(p. 443\)](#)并查看和纠正任何错误或建议。然后，您可以将策略附加到 Amazon 账户中的身份 (用户、组和角色)。

Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅[IAM 与 Amazon STS 配额、名称要求和字符限制 \(p. 902\)](#)。

创建 IAM 策略 (Amazon CLI)

您可以使用 Amazon Command Line Interface (Amazon CLI) 创建 IAM 客户托管策略或内联策略。

创建客户托管策略 (Amazon CLI)

使用以下命令：

- [create-policy](#)

为 IAM 身份（组、用户或角色）创建内联策略 (Amazon CLI)

使用以下命令之一：

- [put-group-policy](#)
- [put-role-policy](#)
- [put-user-policy](#)

Note

您不能使用 IAM 嵌入[服务相关 \(p. 159\)](#)角色的内联策略。

验证客户托管策略 (Amazon CLI)

使用以下 IAM Access Analyzer 命令：

- [validate-policy](#)

创建 IAM 策略 (Amazon API)

[策略 \(p. 355\)](#)是一个实体；在附加到身份或资源时，策略定义了它们的权限。您可以使用 Amazon API 在 IAM 中创建客户托管策略。客户托管策略是您在自己的 Amazon 账户中管理的独立策略。作为最佳实践，我们建议您采取额外步骤以[验证您的策略 \(p. 443\)](#)并查看和纠正任何错误或建议。然后，您可以将策略附加到 Amazon 账户中的身份（用户、组和角色）。

Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅[IAM 与 Amazon STS 配额、名称要求和字符限制 \(p. 902\)](#)。

创建 IAM 策略 (Amazon API)

您可以使用 Amazon API 创建 IAM 客户托管策略或内联策略。

创建客户托管策略 (Amazon API)

调用以下操作：

- [CreatePolicy](#)

为 IAM 身份（组、用户或角色）创建内联策略 (Amazon API)

调用下列一项操作：

- [PutGroupPolicy](#)
- [PutRolePolicy](#)
- [PutUserPolicy](#)

Note

您不能使用 IAM 嵌入[服务相关 \(p. 159\)](#)角色的内联策略。

验证客户托管策略 (Amazon API)

调用以下 IAM Access Analyzer 操作：

- [ValidatePolicy](#)

验证 IAM 策略

策略是使用 [IAM 策略语法](#)编写的 JSON 文档。当您将策略附加到 IAM 实体（例如用户、组或角色）时，策略将向该实体授予权限。

当您使用 Amazon Web Services Management Console 创建或编辑 IAM 访问控制策略时，Amazon 会自动检查它们，以确保它们符合 IAM 策略语法。如果 Amazon 确定策略不符合语法，则会提示您修复策略。

IAM Access Analyzer 提供额外的策略检查和建议，以帮助您进一步优化策略。要了解有关 IAM Access Analyzer 策略检查和可执行建议的更多信息，请参阅 [IAM Access Analyzer 策略验证](#)。要查看 IAM Access Analyzer 返回的警告、错误和建议的列表，请参阅 [IAM Access Analyzer 策略检查参考](#)。

验证范围

Amazon 检查 JSON 策略语法。它还验证您的 ARN 格式是否正确，操作名称和条件键是否正确。

访问策略验证

当您创建 JSON 策略或在 Amazon Web Services Management Console 中编辑现有策略时，将自动对策略进行验证。如果策略语法无效，您会收到通知，必须先解决问题，然后才能继续。如果您有适用于 `access-analyzer:ValidatePolicy` 的权限，则 IAM Access Analyzer 策略验证的结果会自动返回到 Amazon Web Services Management Console。您还可以使用 Amazon API 或 Amazon CLI 验证策略。

现有策略

您的现有策略可能无效，因为它们是在策略引擎的最新更新之前创建或保存的。作为最佳实践，我们建议您打开现有策略并查看生成的策略验证结果。如果未修复任何策略语法错误，则无法编辑和保存现有策略。

基于访问活动生成策略

作为管理员或开发人员，您可以向 IAM 实体（用户或角色）授予超出其需要的权限。IAM 提供了多个选项来帮助您优化授予的权限。其中一种选择是根据实体的访问活动生成 IAM policy。IAM 访问分析器会查看您的 Amazon CloudTrail 日志并生成一个策略模板，其中包含角色在指定日期范围内使用的权限。您可以使用模板创建具有精细权限的策略，该策略仅授予支持特定使用案例所需的权限。

例如，假设您是一名开发人员，您的工程团队正在开发一个项目来创建新应用程序。为了鼓励实验并加快团队进度，您在应用程序开发过程中配置了具有广泛权限的角色。现在该应用程序已准备就绪，可投入生产。在应用程序可以在生产账户中启动之前，您希望仅确定角色需要为应用程序运行提供的权限。这可帮助您遵循 [授予最低权限 \(p. 754\)](#) 的最佳实践。您可以根据开发账户中应用程序使用的角色的访问活动生成策略。您可以进一步优化生成的策略，然后将该策略附加到生产账户中的实体。

要了解有关 IAM Access Analyzer 策略生成的更多信息，请参阅 [IAM Access Analyzer 策略生成](#)。

使用 IAM policy simulator 测试 IAM policy

有关如何以及为何使用 IAM policy 的更多信息，请参阅 [IAM 中的策略和权限 \(p. 355\)](#)

您可以在以下位置访问 IAM policy simulator 控制台：<https://policysim.aws.amazon.com/>

[IAM policy simulator 入门](#)

使用 IAM policy simulator，您可以测试基于身份的策略、IAM 权限边界、组织服务控制策略 (SCPs) 和基于资源的策略并对它们进行故障排除。以下是您可以使用策略模拟器执行的一些常见操作：

- 测试附加到您 Amazon 账户中的 IAM 用户、用户组或角色的策略。如果多个策略附加到用户、用户组或角色，您可以测试所有策略，也可以选择单个策略进行测试。您可以测试为特定资源所选的策略允许或拒绝哪些操作。
- 测试[权限边界 \(p. 370\)](#)对 IAM 实体的影响并进行故障排除。注意：一次只能模拟一个权限边界。
- 测试附加到 Amazon 资源（例如 Amazon S3 存储桶、Amazon SQS 队列、Amazon SNS 主题或 Amazon S3 Glacier 文件库）的策略。
- 如果您的 Amazon 账户是 [Amazon Organizations](#) 中的企业的成员，则您可以测试服务控制策略 (SCP) 对您的 IAM policy 和资源策略的影响。
- 通过将尚未附加到用户、用户组或角色的新策略键入或复制到模拟器中，对这些新策略进行测试。这些仅在模拟中使用，不会保存。请注意：您不能将基于资源的策略键入或者复制到模拟器中。要在模拟器中使用基于资源的策略，必须在模拟中包含该资源。还必须选中该复选框以将该资源的策略包括在模拟中。
- 使用所选服务、操作和资源测试策略。例如，您可以通过测试确保策略允许某个实体在 Amazon S3 服务中对特定存储桶执行 `ListAllMyBuckets`、`CreateBucket` 和 `DeleteBucket` 操作。
- 通过提供上下文密钥（如 IP 地址或日期，包含在被测试策略的 `Condition` 元素中）模拟真实方案。
- 识别策略中导致允许或拒绝访问特定资源或操作的特定语句。

主题

- [IAM policy simulator 的工作方式 \(p. 444\)](#)
- [使用 IAM policy simulator 所需的权限 \(p. 444\)](#)
- [使用 IAM policy simulator \(控制台\) \(p. 446\)](#)
- [使用 IAM policy simulator \(Amazon CLI 和 Amazon API\) \(p. 450\)](#)

IAM policy simulator 的工作方式

模拟器评估您选择的策略，为您指定的每项操作确定有效权限。模拟器使用真实 Amazon 服务请求期间所用的同一策略评估引擎。只是在以下几方面与真实 Amazon 环境不同：

- 模拟器不发出真实 Amazon 服务请求，因此，您可以安全地测试在真实 Amazon 环境中会进行不需要的更改的请求。
- 因为模拟器不模拟运行所选操作，所以它不能报告对模拟请求的任何响应。返回的唯一结果是请求的操作是被允许还是被拒绝。
- 如果您在模拟器内编辑策略，所做的更改只影响模拟器。您的 Amazon 账户中的相应策略保持不变。
- 您不能测试带有[全局条件键 \(p. 998\)](#)的 Amazon Organizations 服务控制策略 (SCP)。

使用 IAM policy simulator 所需的权限

您可以使用策略模拟器控制台或策略模拟器 API 来测试策略。默认情况下，控制台用户可以通过将尚未附加到用户、用户组或角色的策略键入或复制到模拟器中来对其进行测试。这些策略仅在模拟中使用，不会揭露敏感信息。API 用户必须具有测试未附加策略的权限。您可以允许控制台或 API 用户测试附加到 Amazon 账户中的 IAM 用户、用户组或角色的策略。为此，您必须提供检索这些策略的权限。要测试基于资源的策略，用户必须有权检索相关资源的策略。

有关允许用户模拟策略的控制台和 API 策略示例，请参阅[the section called “示例策略：Amazon Identity and Access Management \(IAM\)” \(p. 393\)](#)。

使用策略模拟器控制台所需的权限

您可以允许用户测试附加到 Amazon 账户中的 IAM 用户、用户组或角色的策略。为此，您必须向用户提供检索这些策略的权限。要测试基于资源的策略，用户必须有权检索相关资源的策略。

要查看允许为附加到用户、用户组或角色的策略使用策略模拟器控制台的示例策略，请参阅 [IAM：访问策略模拟器控制台 \(p. 415\)](#)。

要查看只允许具有特定路径的用户使用策略模拟器控制台的示例策略，请参阅 [IAM：基于用户路径访问策略模拟器控制台 \(p. 424\)](#)。

要创建只允许为一种实体类型使用策略模拟器控制台的策略，请按照以下步骤操作。

允许控制台用户模拟用户的策略

在策略中包含以下操作：

- iam:GetGroupPolicy
- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetUser
- iam:GetUserPolicy
- iam:ListAttachedUserPolicies
- iam:ListGroupsForUser
- iam:ListGroupPolicies
- iam:ListUserPolicies
- iam:ListUsers

允许控制台用户模拟用户组的策略

在策略中包含以下操作：

- iam:GetGroup
- iam:GetGroupPolicy
- iam:GetPolicy
- iam:GetPolicyVersion
- iam:ListAttachedGroupPolicies
- iam:ListGroupPolicies
- iam:ListGroups

允许控制台用户模拟角色的策略

在策略中包含以下操作：

- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetRole
- iam:GetRolePolicy
- iam:ListAttachedRolePolicies
- iam:ListRolePolicies
- iam:ListRoles

要测试基于资源的策略，用户必须有权检索相关资源的策略。

允许控制台用户在 Amazon S3 存储桶中测试基于资源的策略

在您的策略中包含以下操作：

- `s3:GetBucketPolicy`

例如，以下策略使用该操作，以允许控制台用户在特定的 Amazon S3 存储桶中模拟基于资源的策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetBucketPolicy",
      "Resource": "arn:aws:s3:::bucket-name/*"
    }
  ]
}
```

使用策略模拟器 API 所需的权限

策略模拟器 API 操作 [GetContextKeyForCustomPolicy](#) 和 [SimulateCustomPolicy](#) 允许用户测试尚未附加到用户、用户组或角色的策略。要测试此类策略，请将策略以字符串形式传递给 API。这些策略仅在模拟中使用，不会揭露敏感信息。您还可以使用此 API 测试附加到 Amazon 账户中的 IAM 用户、用户组或角色的策略。要执行此操作，您必须为用户提供调用 [GetContextKeyForPrincipalPolicy](#) 和 [SimulatePrincipalPolicy](#) 的权限。

要查看允许为当前 Amazon 账户中已附加策略和未附加的策略使用策略模拟器 API 的示例策略，请参阅 [IAM：访问策略模拟器 API \(p. 415\)](#)。

要创建只允许为一种策略类型使用策略模拟器 API 的策略，请按照以下步骤操作。

允许 API 用户模拟作为字符串直接传递到 API 的策略

在策略中包含以下操作：

- `iam:GetContextKeysForCustomPolicy`
- `iam:SimulateCustomPolicy`

允许 API 用户模拟附加到 IAM 用户、用户组、角色或资源的策略

在策略中包含以下操作：

- `iam:GetContextKeysForPrincipalPolicy`
- `iam:SimulatePrincipalPolicy`

例如，要向名为“Bob”的用户提供模拟已分配到名为“Alice”的用户的策略的权限，请向 Bob 提供访问以下资源的权限：`arn:aws:iam::777788889999:user/alice`。

要查看只允许具有特定路径的用户使用策略模拟器 API 的示例策略，请参阅 [IAM：基于用户路径访问策略模拟器 API \(p. 424\)](#)。

使用 IAM policy simulator (控制台)

默认情况下，用户可以通过将尚未附加到用户、用户组或角色的策略键入或复制到策略模拟器控制台上来对其进行测试。这些策略仅在模拟中使用，不会揭露敏感信息。

测试未附加到用户、用户组或角色的策略 (控制台)

1. 在以下位置打开 IAM policy simulator 控制台：<https://policysim.aws.amazon.com/>。
2. 在页面顶部的 Mode: 菜单中，选择 New Policy。

3. 在 Policy Sandbox 中，选择 Create New Policy。
4. 将策略键入或复制到模拟器中，按照以下步骤中所述使用模拟器。

获得使用 IAM policy simulator 控制台的权限后，可以使用该模拟器测试 IAM 用户、用户组、角色或资源策略。

测试附加到用户、用户组或角色的策略（控制台）

1. 在以下位置打开 IAM policy simulator 控制台：<https://policysim.aws.amazon.com/>。

Note

要作为 IAM 用户登录策略模拟器，请使用您的唯一登录 URL 登录 Amazon Web Services Management Console。然后转到<https://policysim.aws.amazon.com/>。有关以 IAM 用户身份登录的更多信息，请参阅 [IAM 用户如何登录 Amazon \(p. 77\)](#)。

模拟器将以 Existing Policies（现有策略）模式打开，并在 Users, Groups, and Roles（用户、组和角色）下列出您的账户中的 IAM 用户。

2. 选择适合于您的任务的选项：

测试目的：	请执行此操作：
附加到用户的策略	在 Users, Groups, and Roles 列表中，选择 Users。然后选择用户。
附加到用户组的策略	在 Users, Groups, and Roles 列表中，选择 Groups。然后选择用户组。
附加到角色的策略	在 Users, Groups, and Roles 列表中，选择 Roles。然后选择角色。
附加到资源的策略	请参阅 Step 9 。
用户、用户组或角色的自定义策略	选择 Create New Policy（创建新策略）。在新的 Policies（策略）窗格中，键入或粘贴策略，然后选择应用。

提示

要测试附加到组的策略，您可以直接从 [IAM 控制台](#) 启动 IAM policy simulator：在导航窗格中，选择 User groups（用户组）。选择要对其测试策略的组的名称，然后选择 Permissions 选项卡。选择 Simulate（模拟）。

要测试附加到用户的客户托管策略，请执行以下操作：在导航窗格中选择 Users（用户）。选择要对其测试策略的用户名称。然后选择 Permissions 选项卡，并展开您要测试的策略。在最右端，选择 Simulate policy。IAM policy simulator 在新窗口中打开，并在 Policies（策略）窗格中显示所选策略。

3. （可选）如果您的账户是 [Amazon Organizations](#) 中某个企业的成员，则请选中 Amazon Organizations SCP 旁边的复选框以将 SCP 包含在模拟评估中。SCP 是指定组织或组织单元 (OU) 的最大权限的 JSON 策略。SCP 限制成员账户中实体的权限。如果 SCP 阻止某个服务或操作，则该账户中的所有实体均无法访问该服务或执行该操作。即使管理员使用 IAM 或资源策略明确地为该服务或操作授予权限，也依然如此。

如果您的账户不是企业的成员，则不会显示该复选框。

4. （可选）您可以测试为 IAM 实体（用户或角色，但不包括用户组）设置为 [权限边界 \(p. 370\)](#) 的策略。如果当前为实体设置了权限边界策略，则该策略将显示在 Policies（策略）窗格中。您仅可为实体设置一个权限边界。要测试不同的权限边界，您可以创建自定义权限边界。为此，选择 Create New Policy（创建新策略）。将打开新的 Policies（策略）窗格。在菜单中，选择 Custom IAM Permissions Boundary Policy（自定义 IAM 权限边界策略）。输入新策略的名称，然后在下面的空间中键入或复制策略。选择应用以保存策略。接下来，选择 Back（返回）以返回原始 Policies（策略）窗格。然后选中要用于模拟的权限边界旁边的复选框。

5. (可选) 您可以只测试附加到用户、用户组或角色的策略的子集。为此, 请在 Policies (策略) 窗格中清除每个要排除的策略旁边的复选框。
6. 在 Policy Simulator 下, 选择 Select service, 然后选择要测试的服务。然后选择 Select actions, 选择一个或多个要测试的操作。尽管菜单一次只为一项服务显示可用选择, 不过 Action Settings and Results 中会显示所选择的全部服务和操作。
7. (可选) 如果在 Step 2 和 Step 5 中选择的任一策略包含具有 [Amazon 全局条件键 \(p. 998\)](#) 的条件, 请为这些键提供值。为此, 可以展开 Global Settings (全局设置) 部分, 为其中显示的键名称键入值。

Warning

如果将某个条件键的值留空, 则模拟过程会忽略该键。在某些情况下, 这会导致错误, 使模拟无法运行。在其他情况下, 模拟运行, 但结果可能不可靠。在这些情况下, 模拟过程与条件键或变量包含值的真实条件不相符。

8. (可选) 所选的每个操作均显示在 Action Settings and Results (操作设置和结果) 列表中, 在实际运行模拟前, Permission (权限) 列中显示 Not simulated。在运行模拟前, 您可以使用资源配置每个操作。要为特定方案配置单独的操作, 请选择箭头展开相应操作的行。如果该操作支持资源级权限, 您可以键入要测试其访问权限的特定资源的 [Amazon Resource Name \(ARN\) \(p. 896\)](#)。默认情况下, 每个资源都设置为一个通配符 (*)。您还可以为任意条件上下文密钥指定值。如前所述, 系统会忽略值为空的密钥, 从而导致模拟失败或结果不可靠。
 - a. 单击操作名称旁的箭头展开每一行, 可以配置准确模拟方案中的操作所需的任何其他信息。如果操作需要任何资源级权限, 您可以键入要模拟访问的特定资源的 [Amazon Resource Name \(ARN\) \(p. 896\)](#)。默认情况下, 每个资源都设置为一个通配符 (*)。
 - b. 如果操作支持但未设置资源级权限, 可以选择 Add Resource, 从而选择要添加到模拟中的资源类型。
 - c. 如果所选任一策略包含的 Condition 元素引用该操作的服务的上下文密钥, 则该操作下会显示相应键名称。您可以指定在对指定的资源模拟该操作的过程中将使用的值。

需要不同资源类型组的操作

某些操作在不同情况下需要不同的资源类型。每个资源类型组都与一个方案相关联。如果其中一个适用于您的模拟, 则选择它, 模拟器需要适合该方案的资源类型。下表列出了每个受支持的方案选项, 以及为运行模拟您必须定义的资源。

以下每个 Amazon EC2 场景都需要指定 instance、image 和 security-group 资源。如果方案包含某个 EBS 卷, 您必须指定该 volume 作为资源。如果 Amazon EC2 场景包含 Virtual Private Cloud (VPC), 则必须提供 network-interface 资源。如果它包含 IP 子网, 您必须指定 subnet 资源。有关 Amazon EC2 场景选项的更多信息, 请参阅 Amazon EC2 用户指南中的 [支持的平台](#)。

- EC2-VPC-InstanceStore

实例、映像、安全组、网络接口

- EC2-VPC-InstanceStore-Subnet

实例、映像、安全组、网络接口、子网

- EC2-VPC-EBS

实例、映像、安全组、网络接口、卷

- EC2-VPC-EBS-Subnet

实例、映像、安全组、网络接口、子网、卷

9. (可选) 如果要在模拟中包含基于资源的策略, 必须首先在 Step 6 中选择要对资源模拟的操作。展开所选操作的行, 对要模拟的策略键入该资源的 ARN。然后选择 ARN 文本框旁边的 Include Resource Policy (包括资源策略)。IAM policy simulator 当前仅支持来自以下服务的基于资源的策略: Amazon S3 (仅限基于资源的策略; 当前不支持 ACL)、Amazon SQS、Amazon SNS 和未锁定的 S3 Glacier 文件库 (当前不支持锁定的文件库)。

10. 选择右上角的 Run Simulation。

Action Settings and Results 每行中的 Permission 列显示对指定资源模拟该操作的结果。

11. 要查看策略中的哪个语句显式允许或拒绝操作，请在 Permission (权限) 列中，选择 **N** matching statement(s) 链接，展开该行。然后，选择 Show statement (显示语句) 链接。Policies 窗格显示其语句会影响突出显示的模拟结果的相关策略。

Note

如果操作为隐式拒绝，意即，如果操作仅因未明确允许而被拒绝，则 List (列表) 和 Show statement (显示语句) 选项不会显示。

IAM policy simulator 控制台消息故障排除

下表列出了在使用 IAM policy simulator 时可能遇到的告知性和警告性消息。表中还提供了解决消息所示问题的操作步骤。

消息	解决步骤
This policy has been edited. Changes will not be saved to your account.	无需操作。 此为告知性消息。如果在 IAM policy simulator 中编辑现有策略，所做的更改不会影响您的 Amazon 账户。通过模拟器，您可以仅出于测试目的对策略进行更改。
无法获取资源策略。原因：#####	模拟器无法访问所请求的基于资源的策略。确保指定的资源 ARN 正确，并且运行模拟的用户有权读取该资源的策略。
One or more policies require values in the simulation settings. The simulation might fail without these values.	如果您测试的策略中包含条件键或变量，但未在 Simulation Settings (模拟设置) 中为这些键或变量提供任何值，则会显示此消息。 要取消显示此消息，请选择 Simulation Settings (模拟设置)，然后为每个条件键或变量输入值。
You have changed policies. These results are no longer valid.	若您在 Results 窗格中显示结果的同时更改了所选策略，则会显示此消息。Results 窗格中显示的结果不会动态更新。 要取消显示此消息，请再次选择 Run Simulation，以根据 Policies 窗格中所做的更改显示新的模拟结果。
The resource you typed for this simulation does not match this service.	如果您在 Simulation Settings (模拟设置) 窗格中键入的 Amazon Resource Name (ARN) 与您为当前模拟选择的服务不相符，则会显示此消息。例如，如果为 Amazon DynamoDB 资源指定一个 ARN，但选择 Amazon Redshift 作为要模拟的服务，则会显示此消息。 要取消显示此消息，请执行下列操作之一： • 请从 Simulation Settings 窗格中的框中删除该 ARN。 • 选择与在 Simulation Settings 中指定的 ARN 相符的服务。

消息	解决步骤
此操作所属的服务除支持基于资源的策略之外，还支持特殊访问控制机制，如 Amazon S3 ACL 或 Glacier 文件库锁定策略。策略模拟器不支持这些机制，因此结果可能与您的生产环境不同。	无需操作。 此为告知性消息。在当前版本中，模拟器评估附加到用户和用户组的策略，可评估针对 Amazon S3、Amazon SQS、Amazon SNS 和 S3 Glacier 的基于资源的策略。策略模拟器只支持其他 Amazon 服务支持的部分访问控制机制。
DynamoDB FGAC is currently not supported.	无需操作。 该信息性消息指的是精细访问控制。精细的访问控制是指使用 IAM policy 条件来确定谁可以访问 DynamoDB 表和索引中的各个数据项和属性的功能。它还指可以对这些表和索引执行的操作。IAM policy simulator 的当前版本不支持这类策略条件。有关 DynamoDB 访问权限精细控制的更多信息，请参阅 DynamoDB 访问权限的精细控制 。
You have policies that do not comply with the policy syntax. 您可以使用策略模拟器查看您策略的推荐更新。	如果策略不符合 IAM policy 语法，则会在策略列表的顶部显示此消息。要模拟这些策略，请查看位于 验证 IAM 策略 (p. 443) 的策略验证选项来识别和修复这些策略。
This policy must be updated to comply with the latest policy syntax rules.	如果您的策略不符合 IAM policy 语法，则会显示此消息。要模拟这些策略，请查看位于 验证 IAM 策略 (p. 443) 的策略验证选项来识别和修复这些策略。

使用 IAM policy simulator (Amazon CLI 和 Amazon API)

策略模拟器命令通常需要调用 API 操作以执行两项操作：

1. 评估策略并返回策略引用的上下文密钥的列表。您需要了解引用了哪些上下文密钥，以便在下一个步骤中提供它们的值。
2. 模拟策略，在模拟过程中提供操作、资源和上下文密钥的列表。

出于安全原因，API 操作已分为两个组：

- 只对以字符串形式直接传递给 API 的策略进行模拟的 API 操作。此组包括 [GetContextKeysForCustomPolicy](#) 和 [SimulateCustomPolicy](#)。
- 对挂载到指定 IAM 用户、用户组、角色或资源的策略进行模拟的 API 操作。因为这些 API 操作可以获得分配给其他 IAM 实体的权限的详细信息，所以您应考虑限制对这些 API 操作的访问。此组包括 [GetContextKeysForPrincipalPolicy](#) 和 [SimulatePrincipalPolicy](#)。有关限制对 API 操作的访问的更多信息，请参阅 [示例策略：Amazon Identity and Access Management \(IAM\) \(p. 393\)](#)。

在这两种情况下，API 操作都将模拟一个或多个策略对操作和资源的列表的影响。每个操作都与每个资源配对，模拟将确定策略对该资源是允许还是拒绝该操作。您还可以为您的策略引用的任何上下文密钥提供值。通过首先调用 [GetContextKeysForCustomPolicy](#) 或 [GetContextKeysForPrincipalPolicy](#)，可以获取策略引用的上下文密钥的列表。如果不提供上下文密钥值，模拟仍会运行。但结果可能不可靠，这是因为模拟器无法在评估中包含该上下文密钥。

获取上下文密钥的列表 (Amazon CLI、Amazon API)

使用以下命令评估策略列表，返回策略中使用的上下文密钥的列表。

- Amazon CLI `aws iam get-context-keys-for-custom-policy` 和 `aws iam get-context-keys-for-principal-policy`
- Amazon API : `GetContextKeysForCustomPolicy` 和 `GetContextKeysForPrincipalPolicy`

模拟 IAM policy (Amazon CLI、Amazon API)

使用以下命令模拟 IAM policy , 以确定用户的有效权限。

- Amazon CLI `aws iam simulate-custom-policy` 和 `aws iam simulate-principal-policy`
- Amazon API : `SimulateCustomPolicy` 和 `SimulatePrincipalPolicy`

添加和删除 IAM 身份权限

可使用策略定义身份 (用户、用户组或角色) 的权限。您可以通过使用 Amazon Web Services Management Console、Amazon Command Line Interface (Amazon CLI) 或 IAM API 附加和分离 Amazon 策略来添加和删除权限。您还可使用策略为使用相同方法的唯一实体 (用户或角色) 设置 [permissions boundaries](#) (p. 370) (权限边界)。权限边界是一项控制实体可拥有的最大权限的高级 Amazon 功能。

主题

- [术语](#) (p. 451)
- [查看身份活动](#) (p. 452)
- [添加 IAM 身份权限 \(控制台 \)](#) (p. 452)
- [删除 IAM 身份权限 \(控制台 \)](#) (p. 453)
- [添加 IAM policy \(Amazon CLI\)](#) (p. 454)
- [移除 IAM policy \(Amazon CLI\)](#) (p. 455)
- [添加 IAM policy \(Amazon API\)](#) (p. 456)
- [删除 IAM policy \(Amazon API\)](#) (p. 457)

术语

将权限策略与身份 (用户、用户组和角色) 相关联时, 术语和过程因您使用托管策略还是内联策略而异 :

- 附加 - 与托管策略一起使用。您将托管策略附加到身份 (用户、用户组或角色)。如果附加一个策略, 则会将该策略中的权限应用于身份。
- 分离 - 与托管策略一起使用。将托管策略与 IAM 身份 (用户、用户组或角色) 分离。如果分离一个策略, 则会将其权限从身份中删除。
- 嵌入 - 与内联策略一起使用。您在身份 (用户、用户组或角色) 中嵌入内联策略。如果嵌入一个策略, 则会将该策略中的权限应用于身份。由于内联策略存储在身份中, 因此, 将嵌入而不是附加该策略, 尽管结果类似。

Note

您只能将 [服务相关角色](#) (p. 159) 的内联策略嵌入依赖该角色的服务。请参阅服务的 [Amazon 文档](#), 以了解它是否支持该功能。

- 删除 - 与内联策略一起使用。从 IAM 身份 (用户、用户组或角色) 删除内联策略。如果删除一个策略, 则会将其权限从身份中删除。

Note

您只能将 [服务相关角色](#) (p. 159) 的内联策略从依赖该角色的服务中删除。请参阅服务的 [Amazon 文档](#), 以了解它是否支持该功能。

您可以使用控制台、Amazon CLI 或 Amazon API 来执行这些操作中的任一操作。

更多信息

- 有关托管策略和内联策略之间的差别的更多信息，请参阅[托管策略与内联策略](#) (p. 364)。
- 有关权限边界的更多信息，请参阅[IAM 实体的权限边界](#) (p. 370)。
- 有关 IAM policy 的一般信息，请参阅[IAM 中的策略和权限](#)。(p. 355)
- 有关验证 IAM 用户策略的信息，请参阅[验证 IAM 策略](#) (p. 443)。
- Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅[IAM 与 Amazon STS 配额、名称要求和字符限制](#) (p. 902)。

查看身份活动

在更改身份（用户、用户组或角色）的权限之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅[使用上次访问的信息优化 Amazon 中的权限](#) (p. 467)。

添加 IAM 身份权限（控制台）

您可以使用 Amazon Web Services Management Console 向身份（用户、用户组或角色）添加权限。为此，请附加控制权限的托管策略，或指定充当[权限边界](#) (p. 370) 的策略。您还可以嵌入内联策略。

将托管策略用作身份的权限策略（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Policies (策略)。
3. 在策略列表中，选中要附加的策略的名称旁边的复选框。您可以使用搜索框筛选策略列表。
4. 选择 Actions (操作)，然后选择 Attach (附加)。
5. 选择一个或多个要将策略附加到的身份。您可以使用搜索框来筛选主体实体列表。在选择身份后，选择附加策略。

使用托管策略设置权限边界（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Policies (策略)。
3. 在策略列表中，选择要设置的策略的名称。您可以使用搜索框筛选策略列表。
4. 在策略摘要页面上，选择策略用法选项卡，然后（如有必要）打开权限边界部分并选择设置边界。
5. 选择要将策略用于其权限边界的一个或多个用户或角色。您可以使用搜索框来筛选主体实体列表。选择主体后，选择设置边界。

为用户或角色嵌入内联策略（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择用户或角色。
3. 在列表中，选择要在其中嵌入策略的用户或角色的名称。

4. 请选择 Permissions 选项卡。
5. 选择 Add permissions (添加权限) , 然后选择 Add inline policy (添加内联策略) 。

Note

您不能在 IAM 中的 [服务相关角色 \(p. 159\)](#) 中嵌入内联策略。由于链接服务定义了您是否能修改角色的权限, 因此, 您可能能够从服务控制台、API 或 Amazon CLI 添加其他策略。要查看某个服务的 [服务相关角色文档](#), 请参阅 [使用 IAM 的 Amazon 服务 \(p. 908\)](#) 并在您的服务的 Service-Linked Role 列中选择 Yes。

6. 选择以下方法之一来查看创建策略所需的步骤：
 - [导入现有托管策略 \(p. 440\)](#) - 您可以在您的账户中导入一个托管策略, 然后编辑该策略以根据您的特定要求进行自定义。托管策略可以是 Amazon 托管策略, 也可以是您以前创建的客户托管策略。
 - [使用可视化编辑器创建策略 \(p. 439\)](#) - 您可以在可视化编辑器中从头开始构建新的策略。如果使用可视化编辑器, 您不需要了解 JSON 语法。
 - [在“JSON”选项卡上创建策略 \(p. 438\)](#) - 在 JSON 选项卡上, 您可以使用 JSON 语法创建策略。您可以输入新的 JSON 策略文档或粘贴 [example policy \(p. 391\)](#) (示例策略) 。
7. 创建内联策略后, 它会自动嵌入您的用户和角色。

为用户组嵌入内联策略 (控制台)

1. 登录 Amazon Web Services Management Console, 然后通过以下网址打开 IAM 控制台: <https://console.aws.amazon.com/iam/>。
2. 在导航窗格中, 选择 User groups (用户组) 。
3. 在列表中, 请选择要在其中嵌入策略的用户组的名称。
4. 请选择 Permissions (权限) 选项卡, 然后选择 Add permissions (添加权限) , 接着选择 Create inline policy (创建内联策略) 。
5. 请执行下列操作之一：
 - 在创建策略页面上, 请选择 Visual editor (可视化编辑器) 选项卡。有关更多信息, 请参阅 [使用可视化编辑器创建策略 \(p. 439\)](#)。
 - 请选择以在 JSON 选项卡上创建策略。有关更多信息, 请参阅 [在“JSON”选项卡上创建策略 \(p. 438\)](#)。
6. 若您满意所创建的策略, 请选择 Create policy (创建策略) 。

更改一个或多个实体的权限边界 (控制台)

1. 登录 Amazon Web Services Management Console, 然后通过以下网址打开 IAM 控制台: <https://console.aws.amazon.com/iam/>。
2. 在导航窗格中, 选择 Policies (策略)。
3. 在策略列表中, 选择要设置的策略的名称。您可以使用搜索框筛选策略列表。
4. 在策略摘要页面上, 选择策略用法选项卡, 然后 (如有必要) 打开权限边界部分。选中要更改其边界的用户或角色旁边的复选框, 然后选择更改边界。
5. 选择要用于权限边界的新策略。您可以使用搜索框筛选策略列表。在选择策略后, 选择更改边界。

删除 IAM 身份权限 (控制台)

您可以使用 Amazon Web Services Management Console 删除身份 (用户、用户组或角色) 的权限。为此, 请分离控制权限的托管策略, 或删除充当 [权限边界 \(p. 370\)](#) 的策略。您还可以删除内联策略。

分离用作权限策略的托管策略（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Policies (策略)。
3. 在策略列表中，选中要分离的策略的名称旁边的复选框。您可以使用搜索框筛选策略列表。
4. 选择 Actions (操作)，然后选择 Detach (删除)。
5. 选择要从中分离策略的身份。您可以使用搜索框筛选身份列表。在选择身份后，选择分离策略。

删除权限边界（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Policies (策略)。
3. 在策略列表中，选择要设置的策略的名称。您可以使用搜索框筛选策略列表。
4. 在策略摘要页面上，选择策略用法选项卡，然后（如有必要）打开权限边界部分并选择删除边界。
5. 确认您要删除边界并选择删除。

删除内联策略（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Groups (组)、Users (用户) 或 Roles (角色)。
3. 在列表中，请选择具有要修改的策略的用户组、用户或角色的名称。
4. 请选择 Permissions 选项卡。
5. 请执行下列操作之一：
 - 在 User groups (用户组) 或 Roles (角色) 下，选中策略旁边的复选框，然后选择 Remove (移除)。
 - 在 Users (用户) 下，选择 X。
6. 请执行下列操作之一：
 - 在 User groups (用户组) 或 Roles (角色) 下，在确认框中选择 Delete (删除)。
 - 在 Users (用户) 下，针对直接附加的策略，在策略的确认框中选择 Detach (分离)，或者针对从组附加的策略选择 Remove from group (从组中移除)。

添加 IAM policy (Amazon CLI)

您可以使用 Amazon CLI 向身份（用户、用户组或角色）添加权限。为此，请附加控制权限的托管策略，或指定充当**权限边界** (p. 370) 的策略。您还可以嵌入内联策略。

将托管策略用作实体的权限策略 (Amazon CLI)

1. （可选）要查看有关托管策略的信息，请运行以下命令：
 - 列出托管策略：[aws iam list-policies](#)
 - 检索有关托管策略的详细信息：[get-policy](#)
2. 要将托管策略附加到身份（用户、用户组或角色），请使用以下命令之一：
 - [aws iam attach-user-policy](#)

- [aws iam attach-group-policy](#)
- [aws iam attach-role-policy](#)

使用托管策略设置权限边界 (Amazon CLI)

1. (可选) 要查看有关托管策略的信息, 请运行以下命令:
 - 列出托管策略: [aws iam list-policies](#)
 - 检索有关托管策略的详细信息: [aws iam get-policy](#)
2. 要使用托管策略为实体 (用户或角色) 设置权限边界, 请使用下列命令之一:
 - [aws iam put-user-permissions-boundary](#)
 - [aws iam put-role-permissions-boundary](#)

嵌入内联策略 (Amazon CLI)

要将内联策略嵌入身份 (用户、用户组或不是[服务相关角色 \(p. 159\)](#)的角色), 请使用以下命令之一:

- [aws iam put-user-policy](#)
- [aws iam put-group-policy](#)
- [aws iam put-role-policy](#)

移除 IAM policy (Amazon CLI)

您可以使用 Amazon CLI 分离控制权限的托管策略, 或删除充当[权限边界 \(p. 370\)](#)的策略。您还可以删除内联策略。

分离用作权限策略的托管策略 (Amazon CLI)

1. (可选) 要查看有关策略的信息, 请运行以下命令:
 - 列出托管策略: [aws iam list-policies](#)
 - 检索有关托管策略的详细信息: [aws iam get-policy](#)
2. (可选) 要了解策略与身份之间的关系, 请运行以下命令:
 - 列出托管策略附加到的身份 (用户、用户组和角色):
 - [aws iam list-entities-for-policy](#)
 - 要列出附加到身份 (用户、用户组或角色) 的托管策略, 请使用以下一个命令:
 - [aws iam list-attached-user-policies](#)
 - [aws iam list-attached-group-policies](#)
 - [aws iam list-attached-role-policies](#)
3. 要从身份 (用户、用户组或角色) 分离托管策略, 请使用以下命令之一:
 - [aws iam detach-user-policy](#)
 - [aws iam detach-group-policy](#)
 - [aws iam detach-role-policy](#)

删除权限边界 (Amazon CLI)

1. (可选) 要查看当前用于为用户或角色设置权限边界的托管策略, 请运行以下命令:
 - [aws iam get-user](#)

- [aws iam get-role](#)
- 2. (可选) 要查看将托管策略用于其权限边界的用户或角色, 请运行以下命令:
 - [aws iam list-entities-for-policy](#)
- 3. (可选) 要查看有关托管策略的信息, 请运行以下命令:
 - 列出托管策略: [aws iam list-policies](#)
 - 检索有关托管策略的详细信息: [aws iam get-policy](#)
- 4. 要从用户或角色删除权限边界, 请使用以下命令之一:
 - [aws iam delete-user-permissions-boundary](#)
 - [aws iam delete-role-permissions-boundary](#)

删除内联策略 (Amazon CLI)

1. (可选) 要列出附加到身份 (用户、用户组和角色) 的所有内联策略, 请使用以下命令之一:
 - [aws iam list-user-policies](#)
 - [aws iam list-group-policies](#)
 - [aws iam list-role-policies](#)
2. (可选) 要检索嵌入到身份 (用户、用户组或角色) 中的内联策略文档, 请使用以下命令之一:
 - [aws iam get-user-policy](#)
 - [aws iam get-group-policy](#)
 - [aws iam get-role-policy](#)
3. 要从身份 (用户、用户组或不是[服务相关角色 \(p. 159\)](#)的角色) 中删除内联策略, 请使用以下命令之一:
 - [aws iam delete-user-policy](#)
 - [aws iam delete-group-policy](#)
 - [aws iam delete-role-policy](#)

添加 IAM policy (Amazon API)

您可以使用 Amazon API 附加控制权限的托管策略, 或指定充当[权限边界 \(p. 370\)](#)的策略。您还可以嵌入内联策略。

将托管策略用作实体的权限策略 (Amazon API)

1. (可选) 要查看有关策略的信息, 请调用以下操作:
 - 列出托管策略: [ListPolicies](#)
 - 检索有关托管策略的详细信息: [GetPolicy](#)
2. 要将托管策略附加到身份 (用户、用户组或角色), 请调用以下操作之一:
 - [AttachUserPolicy](#)
 - [AttachGroupPolicy](#)
 - [AttachRolePolicy](#)

使用托管策略设置权限边界 (Amazon API)

1. (可选) 要查看有关托管策略的信息, 请调用以下操作:

- 列出托管策略：[ListPolicies](#)
 - 检索有关托管策略的详细信息：[GetPolicy](#)
2. 要使用托管策略为实体（用户或角色）设置权限边界，请调用下列操作之一：
- [PutUserPermissionsBoundary](#)
 - [PutRolePermissionsBoundary](#)

嵌入内联策略 (Amazon API)

要将内联策略嵌入身份（用户、用户组或不是[服务相关角色](#) (p. 159)的角色），请调用以下操作之一：

- [PutUserPolicy](#)
- [PutGroupPolicy](#)
- [PutRolePolicy](#)

删除 IAM policy (Amazon API)

您可以使用 Amazon API 分离控制权限的托管策略，或删除充当[权限边界](#) (p. 370)的策略。您还可以删除内联策略。

分离用作权限策略的托管策略 (Amazon API)

1. （可选）要查看有关策略的信息，请调用以下操作：
 - 列出托管策略：[ListPolicies](#)
 - 检索有关托管策略的详细信息：[GetPolicy](#)
2. （可选）要了解策略与身份之间的关系，请调用以下操作：
 - 列出托管策略附加到的身份（用户、用户组和角色）：
 - [ListEntitiesForPolicy](#)
 - 要列出附加到身份（用户、用户组或角色）的托管策略，请调用以下一项操作：
 - [ListAttachedUserPolicies](#)
 - [ListAttachedGroupPolicies](#)
 - [ListAttachedRolePolicies](#)
3. 要将托管策略与身份（用户、用户组或角色）分离，请调用以下操作之一：
 - [DetachUserPolicy](#)
 - [DetachGroupPolicy](#)
 - [DetachRolePolicy](#)

删除权限边界 (Amazon API)

1. （可选）要查看当前用于为用户或角色设置权限边界的托管策略，请调用以下操作：
 - [GetUser](#)
 - [GetRole](#)
2. （可选）要查看将托管策略用于其权限边界的用户或角色，请调用以下操作：
 - [ListEntitiesForPolicy](#)
3. （可选）要查看有关托管策略的信息，请调用以下操作：

- 列出托管策略：[ListPolicies](#)
 - 检索有关托管策略的详细信息：[GetPolicy](#)
4. 要从用户或角色删除权限边界，请调用以下操作之一：
- [DeleteUserPermissionsBoundary](#)
 - [DeleteRolePermissionsBoundary](#)

删除内联策略 (Amazon API)

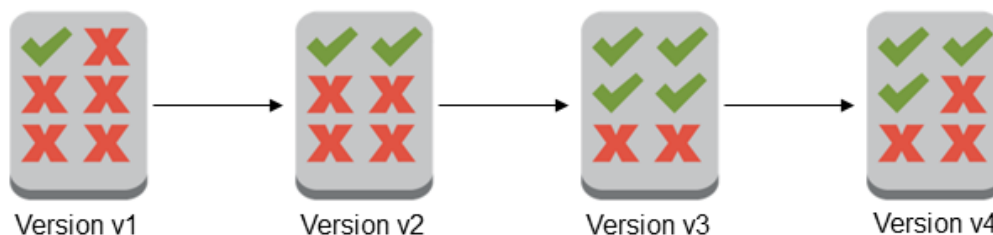
1. (可选) 要列出附加到身份 (用户、用户组和角色) 的所有内联策略，请调用以下操作之一：
- [ListUserPolicies](#)
 - [ListGroupPolicies](#)
 - [ListRolePolicies](#)
2. (可选) 要检索嵌入到身份 (用户、用户组或角色) 中的内联策略文档，请调用以下操作之一：
- [GetUserPolicy](#)
 - [GetGroupPolicy](#)
 - [GetRolePolicy](#)
3. 要将内联策略从身份 (用户、用户组或不是[服务相关角色](#) (p. 159) 的角色) 中删除，请调用以下操作之一：
- [DeleteUserPolicy](#)
 - [DeleteGroupPolicy](#)
 - [DeleteRolePolicy](#)

IAM 策略版本控制

在对 IAM 客户托管策略进行更改时，以及 Amazon 对 Amazon 托管策略进行更改时，更改的策略不会覆盖现有的策略。而是由 IAM 创建新的托管策略版本。IAM 最多可以存储五个版本的客户管理策略。IAM 不支持内联策略版本控制。

下图说明了客户托管策略版本控制。在此示例中，已保存版本 1-4。您最多可以将五个托管策略版本保存到 IAM。在您编辑将创建第六个已保存版本的策略时，可以选择不再保存哪个较旧版本。您可以随时恢复到其他四个已保存版本中的任何一个。

Multiple versions of a single managed policy



策略版本与 `Version` 策略元素不同。`Version` 策略元素用在策略之中，用于定义策略语言的版本。要了解 `Version` 策略元素的更多信息，请参阅[IAM JSON 策略元素：Version](#) (p. 934)。

您可以使用不同版本跟踪对托管策略的更改。例如，您对托管策略进行更改，然后发现更改有意外效果。在这种情况下，通过将以前版本设置为默认版本，可以回滚到以前版本的托管策略。

下面几节说明如何对托管策略进行版本控制。

主题

- [用于设置默认策略版本的权限 \(p. 459\)](#)
- [设置客户托管策略的默认版本 \(p. 459\)](#)
- [使用不同版本回滚更改 \(p. 461\)](#)
- [版本限制 \(p. 461\)](#)

用于设置默认策略版本的权限

设置默认策略版本所需的权限对应于任务的 Amazon API 操作。您可以使用 `CreatePolicyVersion` 或 `SetDefaultPolicyVersion` API 操作来设置默认策略版本。要允许某人设置现有策略的默认策略版本，您可以允许访问 `iam:CreatePolicyVersion` 操作或 `iam:SetDefaultPolicyVersion` 操作。`iam:CreatePolicyVersion` 操作允许他们创建新版本的策略并将该版本设置为默认版本。`iam:SetDefaultPolicyVersion` 操作允许他们将任何现有策略版本设置为默认版本。

Important

在用户策略中拒绝 `iam:SetDefaultPolicyVersion` 操作不会阻止用户创建新策略版本并将它设置为默认版本。

您可以使用以下策略来拒绝用户访问以更改现有客户托管策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "iam:CreatePolicyVersion",
        "iam:SetDefaultPolicyVersion"
      ],
      "Resource": "arn:aws:iam::*:policy/POLICY-NAME"
    }
  ]
}
```

设置客户托管策略的默认版本

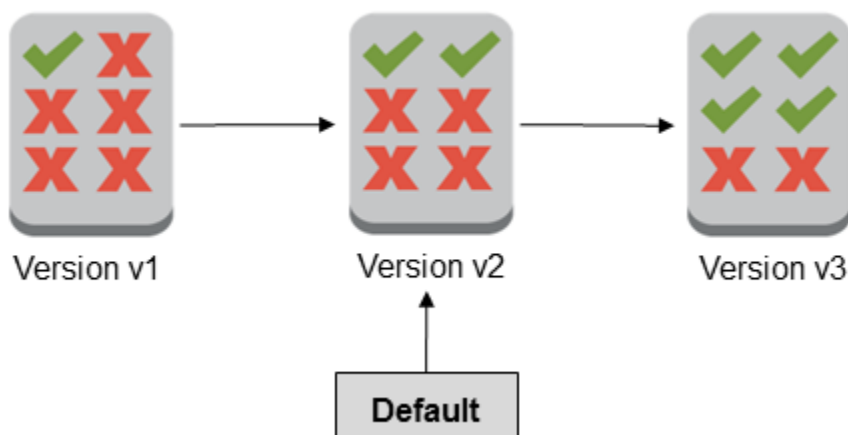
托管策略的一个版本会设置为默认版本。策略的默认版本是有效版本，即，它是对托管策略附加到的所有主体实体（用户、用户组和角色）都生效的版本。

创建客户托管策略时，策略从标识为 v1 的单个版本开始。对于只有单个版本的托管策略，该版本会自动设置为默认版本。对于有多个版本的客户托管策略，由您选择要设置为默认版本的版本。对于 Amazon 托管策略，默认版本由 Amazon 设置。下面的示意图对此概念进行说明。

Managed policy with one version



Managed policy with multiple versions



您可以设置客户托管策略的默认版本，以将该版本应用于该策略附加到的每个 IAM 身份（用户、用户组和角色）。您无法为 Amazon 托管策略或内联策略设置默认版本。

设置客户托管策略的默认版本（控制台）

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Policies (策略)。
3. 在策略列表中，选择要设置其默认版本的策略的名称。您可以使用搜索框筛选策略列表。
4. 选择 Policy Versions (策略版本) 选项卡。选中要设置为默认版本的版本旁的复选框，然后选择 Set as default。

要了解如何从 Amazon Command Line Interface 或 Amazon API 中设置客户托管策略的默认版本，请参阅[编辑客户托管策略 \(Amazon CLI\) \(p. 463\)](#)。

使用不同版本回滚更改

您可以设置客户托管策略的默认版本以回滚您的更改。例如，考虑以下情形：

您创建一个客户托管策略，以允许用户使用 Amazon Web Services Management Console 管理特定的 Amazon S3 存储桶。创建时，您的客户托管策略只有一个版本（标识为 v1），因此该版本自动设置为默认版本。该策略可正常工作。

随后，您将更新策略，以添加权限对第二个 Amazon S3 存储桶进行管理。IAM 会创建新的策略版本（标识为 v2），该版本将包含您所做的更改。您将版本 v2 设置为默认版本，不久之后，您的用户报告，他们缺少使用 Amazon S3 控制台的权限。在此情况下，您可以回滚到策略版本 v1，您知道该版本可正常工作。为此，您可以将版本 v1 设置为默认版本。您的用户现在可以使用 Amazon S3 控制台来管理原始存储桶。

后来，在确定策略版本 v2 中的错误之后，您再次更新策略，添加了对第二个 Amazon S3 存储桶的管理权限。IAM 会创建另一个新版本的策略，标识为 v3。您将版本 v3 设置为默认版本，此版本可正常工作。此时，您删除策略版本 v2。

版本限制

一个托管策略最多可以有五个版本。如果需要从 Amazon Command Line Interface 或 Amazon API 在五个版本之外更改托管策略更改，必须先删除一个或多个现有版本。如果使用的是 Amazon Web Services Management Console，则您在编辑策略之前不必删除版本。当您保存第六个版本时，会出现一个对话框，提示您删除策略的一个或多个非默认版本。您可以查看每个版本的 JSON 策略文档，以方便做出决定。有关该对话框的详细信息，请参阅[the section called “编辑 IAM 策略” \(p. 461\)](#)。

您可以根据需要删除任何托管策略版本（默认版本除外）。删除某个版本时，其余版本的版本标识符不会更改。因此，版本标识符可能是不连续的。例如，如果您删除托管策略版本 v2 和 v4，然后添加两个新版本，则其余版本标识符可能是 v1、v3、v5、v6 和 v7。

编辑 IAM 策略

[策略 \(p. 355\)](#) 是一个实体；在附加到身份或资源时，策略定义了它们的权限。策略作为 JSON 文档存储在 Amazon 中，并在 IAM 中作为基于身份的策略附加到主体。您可以将基于身份的策略附加到主体（或身份），例如，IAM 用户组、用户或角色。基于身份的策略包括 Amazon 托管策略、客户托管策略和[内联策略 \(p. 364\)](#)。您可以在 IAM 中编辑客户托管策略和内联策略。Amazon 托管策略无法编辑。Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅[IAM 与 Amazon STS 配额、名称要求和字符限制 \(p. 902\)](#)。

主题

- [查看策略访问 \(p. 461\)](#)
- [编辑客户托管策略 \(控制台\) \(p. 461\)](#)
- [编辑内联策略 \(控制台\) \(p. 463\)](#)
- [编辑客户托管策略 \(Amazon CLI\) \(p. 463\)](#)
- [编辑客户托管策略 \(Amazon API\) \(p. 464\)](#)

查看策略访问

在更改策略的权限之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅[使用上次访问的信息优化 Amazon 中的权限 \(p. 467\)](#)。

编辑客户托管策略 (控制台)

您可编辑客户托管策略以更改在策略中定义的权限。客户托管策略最多可以具有五个版本。这是非常重要的，因为如果在五个版本以外更改托管策略，Amazon Web Services Management Console 将提示您决定删

除哪个版本。也可以更改策略的默认版本，或者在编辑之前删除一个版本以避免出现提示。要了解版本的更多信息，请参阅[IAM 策略版本控制 \(p. 458\)](#)。

编辑客户托管策略 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Policies (策略)。
3. 在策略列表中，选择要编辑的策略的名称。您可以使用搜索框筛选策略列表。
4. 选择权限选项卡，然后选择编辑策略。
5. 请执行下列操作之一：
 - 选择可视化编辑器选项卡以更改您的策略，而无需了解 JSON 语法。您可以更改策略中的每个权限块的服务、操作、资源或可选条件。也可以导入一个策略以在您的策略底部添加其他权限。在完成更改后，选择查看策略以继续。
 - 选择 JSON 选项卡，然后在 JSON 文本框中键入或粘贴文本以修改您的策略。也可以导入一个策略以在您的策略底部添加其他权限。解决[策略验证 \(p. 443\)](#)过程中生成的任何安全警告、错误或常规警告，然后选择 Review policy (查看策略)。

Note

您可以随时在可视化编辑器和 JSON 选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy (查看策略)，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅[调整策略结构 \(p. 871\)](#)。

6. 在审核页面上，检查策略摘要，然后选择保存更改进行保存。
7. 如果托管策略已达到最大版本数 (五个)，选择保存将显示一个对话框。要保存新版本，您必须删除至少一个旧版本。无法删除默认版本。从以下选项中进行选择：
 - Remove oldest non-default policy version (version v# - created # days ago) (删除最早的非默认策略版本 (版本 v# - # 天前创建)) - 使用此选项可以查看将被删除的版本及其创建时间。您可以通过选择第二个选项 Select versions to remove 来查看所有非默认版本的 JSON 策略文档。
 - Select versions to remove (选择要删除的版本) - 使用此选项可以查看 JSON 策略文档，并选择一个或多个要删除的版本。

选择要删除的版本后，选择 Delete version and save 来保存新策略版本。

设置客户托管策略的默认版本 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Policies (策略)。
3. 在策略列表中，选择要设置其默认版本的策略的名称。您可以使用搜索框筛选策略列表。
4. 选择 Policy Versions (策略版本) 选项卡。选中要设置为默认版本的版本旁的复选框，然后选择 Set as default。

删除某个版本的客户托管策略 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Policies (策略)。
3. 选择要删除其版本的客户托管策略的名称。您可以使用搜索框筛选策略列表。
4. 选择 Policy Versions (策略版本) 选项卡。选中要删除的版本旁边的复选框，然后选择 Delete (删除)。
5. 确认您要删除该版本，然后选择 Delete。

编辑内联策略 (控制台)

您可以从 Amazon Web Services Management Console 编辑内联策略。

编辑用户、用户组或角色的内联策略 (控制台)

1. 在导航窗格中，选择 Users (用户)、User groups (用户组) 或 Roles (角色)。
2. 请选择具有要修改的策略的用户组、用户或角色的名称。然后选择权限选项卡并展开此策略。
3. 要编辑内联策略，请选择 Edit Policy。
4. 请执行下列操作之一：
 - 选择可视化编辑器选项卡以更改您的策略，而无需了解 JSON 语法。您可以更改策略中的每个权限块的服务、操作、资源或可选条件。也可以导入一个策略以在您的策略底部添加其他权限。在完成更改后，选择查看策略以继续。
 - 选择 JSON 选项卡，然后在 JSON 文本框中键入或粘贴文本以修改您的策略。也可以导入一个策略以在您的策略底部添加其他权限。解决策略验证 (p. 443) 过程中生成的任何安全警告、错误或常规警告，然后选择 Review policy (查看策略)。要在不影响当前附加的实体的情况下保存更改，请清除 Save as default version (以默认版本保存) 的复选框。

Note

您可以随时在可视化编辑器和 JSON 选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy (查看策略)，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅调整策略结构 (p. 871)。

5. 在审核页面上，检查策略摘要，然后选择保存更改进行保存。

编辑客户托管策略 (Amazon CLI)

您可以从 Amazon Command Line Interface (Amazon CLI) 编辑客户托管策略。

Note

一个托管策略最多可以有五个版本。如果您需要在五个版本之外继续对客户托管策略进行更改，则必须首先删除一个或多个现有版本。

编辑客户托管策略 (Amazon CLI)

1. (可选) 要查看有关策略的信息，请运行以下命令：
 - 列出托管策略：[list-policies](#)
 - 检索有关托管策略的详细信息：[get-policy](#)
2. (可选) 要了解策略与身份之间的关系，请运行以下命令：
 - 列出托管策略附加到的身份 (用户、用户组和角色)：
 - [list-entities-for-policy](#)
 - 列出附加到身份 (用户、用户组或角色) 的托管策略：
 - [list-attached-user-policies](#)
 - [list-attached-group-policies](#)
 - [list-attached-role-policies](#)
3. 要编辑客户托管策略，请运行以下命令：
 - [create-policy-version](#)
4. (可选) 要验证客户托管策略，请运行以下 IAM Access Analyzer 命令：

- [validate-policy](#)

设置客户托管策略的默认版本 (Amazon CLI)

1. (可选) 要列出托管策略，请运行以下命令：
 - [list-policies](#)
2. 要设置客户托管策略的默认版本，请运行以下命令：
 - [set-default-policy-version](#)

删除客户托管策略的某个版本 (Amazon CLI)

1. (可选) 要列出托管策略，请运行以下命令：
 - [list-policies](#)
2. 要删除客户托管策略，请运行以下命令：
 - [delete-policy-version](#)

编辑客户托管策略 (Amazon API)

您可以使用 Amazon API 编辑客户托管策略。

Note

一个托管策略最多可以有五个版本。如果您需要在五个版本之外继续对客户托管策略进行更改，则必须首先删除一个或多个现有版本。

编辑客户托管策略 (Amazon API)

1. (可选) 要查看有关策略的信息，请调用以下操作：
 - 列出托管策略：[ListPolicies](#)
 - 检索有关托管策略的详细信息：[GetPolicy](#)
2. (可选) 要了解策略与身份之间的关系，请调用以下操作：
 - 列出托管策略附加到的身份（用户、用户组和角色）：
 - [ListEntitiesForPolicy](#)
 - 列出附加到身份（用户、用户组或角色）的托管策略：
 - [ListAttachedUserPolicies](#)
 - [ListAttachedGroupPolicies](#)
 - [ListAttachedRolePolicies](#)
3. 要编辑客户托管策略，请调用以下操作：
 - [CreatePolicyVersion](#)
4. (可选) 要验证客户托管策略，请调用以下 IAM Access Analyzer 操作：
 - [ValidatePolicy](#)

设置客户托管策略的默认版本 (Amazon API)

1. (可选) 要列出托管策略，请调用以下操作：

- [ListPolicies](#)
2. 要设置客户托管策略的默认版本，请调用以下操作：
 - [SetDefaultPolicyVersion](#)

删除客户托管策略的某个版本 (Amazon API)

1. (可选) 要列出托管策略，请调用以下操作：
 - [ListPolicies](#)
2. 要删除客户托管策略，请调用以下操作：
 - [DeletePolicyVersion](#)

删除 IAM 策略

您可以使用 Amazon Web Services Management Console、Amazon Command Line Interface (Amazon CLI) 或 IAM API 删除 IAM 策略。

有关托管策略和内联策略之间的差别的更多信息，请参阅[托管策略与内联策略](#) (p. 364)。

有关 IAM 策略的一般信息，请参阅 [IAM 中的策略和权限](#)。 (p. 355)。

Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅[IAM 与 Amazon STS 配额、名称要求和字符限制](#) (p. 902)。

主题

- [查看策略访问](#) (p. 465)
- [删除 IAM 策略 \(控制台 \)](#) (p. 465)
- [删除 IAM 策略 \(Amazon CLI\)](#) (p. 466)
- [删除 IAM 策略 \(Amazon API\)](#) (p. 467)

查看策略访问

在删除策略之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体 (个人或应用程序) 的访问权限。有关查看上次访问的信息的更多信息，请参阅[使用上次访问的信息优化 Amazon 中的权限](#) (p. 467)。

删除 IAM 策略 (控制台)

您可删除客户托管策略以便从您的 Amazon 账户中删除它。您无法删除 Amazon 托管策略。

删除客户托管策略 (控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Policies (策略)。
3. 选中您要删除的客户托管策略旁边的复选框。您可以使用搜索框筛选策略列表。
4. 选择 Actions，然后选择 Delete。
5. 确认您要删除该策略，然后选择 Delete。

删除用户组、用户或角色的内联策略 (控制台)

1. 在导航窗格中, 选择 Groups (组)、Users (用户) 或 Roles (角色)。
2. 请选择具有要删除的策略的用户组、用户或角色的名称。然后选择 Permissions 选项卡。如果已选择 Users 或 Roles, 请展开策略。
3. 要删除 User groups (用户组) 中的内联策略, 请选择 Delete (删除)。要删除 Users 或 Roles 中的内联策略, 请选择 X。
4. 如果您要删除 User groups (用户组) 中的单个内联策略, 请键入策略的名称, 然后选择 Delete (删除)。如果要删除 User groups (用户组) 中的多个内联策略, 请键入要删除的策略数, 接着键入 **inline policies**, 然后选择 Delete (删除)。例如, 如果要删除三个内联策略, 请键入 **3 inline policies**。

删除 IAM 策略 (Amazon CLI)

您可以从 Amazon Command Line Interface 中删除客户托管策略。

删除客户托管策略 (Amazon CLI)

1. (可选) 要查看有关策略的信息, 请运行以下命令:
 - 列出托管策略: [list-policies](#)
 - 检索有关托管策略的详细信息: [get-policy](#)
2. (可选) 要了解策略与身份之间的关系, 请运行以下命令:
 - 要列出托管策略附加到的身份 (用户、用户组和角色), 请运行以下命令:
 - [list-entities-for-policy](#)
 - 要列出附加到身份 (用户、用户组或角色) 的托管策略, 请运行下列命令之一:
 - [list-attached-user-policies](#)
 - [list-attached-group-policies](#)
 - [list-attached-role-policies](#)
3. 要删除客户托管策略, 请运行以下命令:
 - [delete-policy](#)

删除内联策略 (Amazon CLI)

1. (可选) 要列出附加到身份 (用户、用户组和角色) 的所有内联策略, 请使用以下命令之一:
 - [aws iam list-user-policies](#)
 - [aws iam list-group-policies](#)
 - [aws iam list-role-policies](#)
2. (可选) 要检索嵌入到身份 (用户、用户组或角色) 中的内联策略文档, 请使用以下命令之一:
 - [aws iam get-user-policy](#)
 - [aws iam get-group-policy](#)
 - [aws iam get-role-policy](#)
3. 要从身份 (用户、用户组或不是[服务相关角色 \(p. 159\)](#)的角色) 中删除内联策略, 请使用以下命令之一:
 - [aws iam delete-user-policy](#)
 - [aws iam delete-group-policy](#)
 - [aws iam delete-role-policy](#)

删除 IAM 策略 (Amazon API)

您可以使用 Amazon API 删除客户托管策略。

删除客户托管策略 (Amazon API)

1. (可选) 要查看有关策略的信息，请调用以下操作：
 - 列出托管策略：[ListPolicies](#)
 - 检索有关托管策略的详细信息：[GetPolicy](#)
2. (可选) 要了解策略与身份之间的关系，请调用以下操作：
 - 要列出托管策略附加到的身份（用户、用户组和角色），请调用以下操作：
 - [ListEntitiesForPolicy](#)
 - 要列出附加到身份（用户、用户组或角色）的托管策略，请调用以下一项操作：
 - [ListAttachedUserPolicies](#)
 - [ListAttachedGroupPolicies](#)
 - [ListAttachedRolePolicies](#)
3. 要删除客户托管策略，请调用以下操作：
 - [DeletePolicy](#)

删除内联策略 (Amazon API)

1. (可选) 要列出附加到身份（用户、用户组和角色）的所有内联策略，请调用以下操作之一：
 - [ListUserPolicies](#)
 - [ListGroupPolicies](#)
 - [ListRolePolicies](#)
2. (可选) 要检索嵌入到身份（用户、用户组或角色）中的内联策略文档，请调用以下操作之一：
 - [GetUserPolicy](#)
 - [GetGroupPolicy](#)
 - [GetRolePolicy](#)
3. 要将内联策略从身份（用户、用户组或不是[服务相关角色](#) (p. 159)的角色）中删除，请调用以下操作之一：
 - [DeleteUserPolicy](#)
 - [DeleteGroupPolicy](#)
 - [DeleteRolePolicy](#)

使用上次访问的信息优化 Amazon 中的权限

作为管理员，您可能会向实体（用户或角色）授予超出其需要的权限。IAM 提供上次访问的信息以帮助您识别未使用的权限，以便您可以将其移除。您可以使用上次访问的信息来优化策略，并仅允许访问您的实体使用的服务和操作。这有助于更好地遵循[最小权限的最佳实践](#) (p. 754)。您可以查看 IAM 或 Amazon Organizations 中存在的实体或策略的上次访问信息。

主题

- [IAM 的上次访问的信息类型](#) (p. 468)
- [Amazon Organizations 的上次访问信息](#) (p. 468)

- [关于上次访问的信息的知识 \(p. 468\)](#)
- [所需权限 \(p. 469\)](#)
- [IAM 和 Organizations 实体的故障排除活动 \(p. 471\)](#)
- [Amazon 跟踪上次访问信息的位置 \(p. 471\)](#)
- [查看 IAM 的上次访问信息 \(p. 472\)](#)
- [查看 Organizations 的上次访问信息 \(p. 475\)](#)
- [使用上次访问信息的示例应用场景 \(p. 479\)](#)

IAM 的上次访问的信息类型

您可以查看 IAM 实体的两种类型的上次访问信息：允许的 Amazon 服务信息和允许的操作信息。此信息包括进行尝试的日期和时间。上次访问的操作信息可用于 Amazon EC2、IAM、Lambda 和 Amazon S3 管理操作。管理操作包括创建、删除和修改操作。要了解有关如何查看 IAM 的上次访问信息的更多信息，请参阅 [查看 IAM 的上次访问信息 \(p. 472\)](#)。

有关使用上次访问信息做出有关向 IAM 实体授予的权限的决策的示例场景，请参阅 [使用上次访问信息的示例应用场景 \(p. 479\)](#)。

要了解有关如何提供管理操作信息的更多信息，请参阅[关于上次访问的信息的知识 \(p. 468\)](#)。

Amazon Organizations 的上次访问信息

如果使用管理账户凭证登录，则可以查看企业中的 Amazon Organizations 实体或策略的上次访问的服务信息。Amazon Organizations 实体包括企业根、企业部门 (OU) 或账户。Amazon Organizations 的上次访问信息包括有关服务控制策略 (SCP) 允许的服务的信息。该信息指明组织或账户中的哪些主体上次尝试访问该服务以及尝试访问的时间。要了解有关报告以及如何查看 Amazon Organizations 的上次访问信息的更多信息，请参阅[查看 Organizations 的上次访问信息 \(p. 475\)](#)。

有关使用上次访问信息做出有关向 Organizations 实体授予的权限的决策的示例场景，请参阅 [使用上次访问信息的示例应用场景 \(p. 479\)](#)。

关于上次访问的信息的知识

在使用报告中的上次访问信息更改 IAM 或 Organizations 实体的权限之前，请查看有关该信息的以下细节。

- **跟踪周期** - 最近的活动通常会在四小时内显示在 IAM 控制台中。服务信息的跟踪周期为最近 400 天。Amazon S3 操作信息的跟踪周期从 2020 年 4 月 12 日开始。Amazon EC2、IAM 和 Lambda 操作的跟踪周期从 2021 年 4 月 7 日开始。有关更多信息，请参阅[Amazon 跟踪上次访问信息的位置 \(p. 471\)](#)。
- **Attempts reported (报告的尝试)** - 服务上次访问的数据包含访问 Amazon API 的所有尝试，而不仅仅是成功的尝试。这包括使用 Amazon Web Services Management Console、Amazon API (通过任何开发工具包) 或任何命令行工具进行的所有尝试。在上次访问的服务相关数据中看到意外的条目并不意味着您的账户信息泄露，因为请求可能已被拒。请参阅您的 CloudTrail 日志并将其作为有关所有 API 调用以及它们是成功还是被拒绝的访问的信息的权威来源。
- **PassRole - iam:PassRole** 操作不会被跟踪，也不包括在 IAM 上次访问的服务信息中。
- **上次访问的操作信息** — 上次访问的操作信息可用于 IAM 实体访问的 Amazon EC2、IAM、Lambda 和 Amazon S3 管理操作。IAM 为 CloudTrail 记录的 Amazon EC2、IAM、Lambda 和 Amazon S3 管理事件提供操作信息。有时，CloudTrail 管理事件也称为控制层面操作或控制层面事件。通过管理事件，可以了解对在您 Amazon 账户内的资源执行的管理操作。要了解有关 CloudTrail 中的管理事件的更多信息，请参阅[使用 CloudTrail 记录管理事件](#)。
- **Report owner (报告所有者)** - 只有生成报告的主体才能查看报告详细信息。这意味着，当您查看 Amazon Web Services Management Console 中的信息时，您可能需要等待它生成和加载。如果您使用 Amazon CLI 或 Amazon API 获取报告详细信息，则您的凭证必须与生成报告的主体的凭证相匹配。如果对角色或

联合身份用户使用临时凭证，则必须在同一会话期间生成和检索报告。有关代入角色会话主体的更多信息，请参阅[Amazon JSON 策略元素：Principal \(p. 936\)](#)。

- IAM 实体 — IAM 的信息包括账户中的 IAM 实体（用户或角色）。Organizations 的信息包括指定 Organizations 实体中的主体（IAM 用户、IAM 角色或 Amazon Web Services 账户根用户）。这些信息不包括未经身份验证的尝试。
- IAM policy 类型 - IAM 的信息包括 IAM 实体的策略允许的服务。这些是附加到角色或直接或通过组附加到用户的策略。其他策略类型允许的访问权限未包含在您的报告中。排除的策略类型包括基于资源的策略、访问控制列表、Amazon Organizations SCP、IAM 权限边界以及会话策略。由服务相关角色提供的权限由它们所关联的服务定义，不能在 IAM 中进行修改。要了解有关服务相关角色的更多信息，请参阅[使用服务相关角色 \(p. 208\)](#)。要了解如何评估不同的策略类型以允许或拒绝访问，请参阅[策略评估逻辑 \(p. 972\)](#)。
- Organizations 策略类型 - Amazon Organizations 的信息仅包含 Organizations 实体继承的服务控制策略 (SCP) 允许的服务。SCP 是附加到根、OU 或账户的策略。其他策略类型允许的访问权限未包含在您的报告中。排除的策略类型包括基于身份的策略、基于资源的策略、访问控制列表、IAM 权限边界以及会话策略。要了解如何评估不同的策略类型以允许或拒绝访问，请参阅[策略评估逻辑 \(p. 972\)](#)。
- 指定策略 ID - 在 Organizations 中使用 Amazon CLI 或 Amazon API 为上次访问的信息生成报告时，您可以选择指定策略 ID。生成的报告包含仅该策略允许的服务的信息。该信息包括指定的 Organizations 实体或其子实体中的最新账户活动。有关更多信息，请参阅[aws iam generate-organizations-access-report](#) 或 [GenerateOrganizationsAccessReport](#)。
- 企业管理账户 - 您必须登录到企业的管理账户才能查看上次访问的服务信息。您可以选择使用 IAM 控制台、Amazon CLI 或 Amazon API 查看管理账户的信息。生成的报告列出所有 Amazon 服务，因为管理账户不受 SCP 限制。如果在 CLI 或 API 中指定策略 ID，则会忽略该策略。对于每个服务，报告仅包含管理账户的信息。不过，其他 Organizations 实体的报告不会返回管理账户中的活动的信息。
- Organizations 设置 – 管理员必须先[在企业根中启用 SCP](#)，然后才能生成 Organizations 的数据。

所需权限

要在 Amazon Web Services Management Console 中查看上次访问的信息，您必须具有授予所需权限的策略。

IAM 信息的权限

要使用 IAM 控制台查看 IAM 用户、角色或策略的上次访问信息，您必须具有包含以下操作的策略：

- iam:GenerateServiceLastAccessedDetails
- iam:Get*
- iam:List*

这些权限允许用户查看以下内容：

- 哪些用户、组或角色将附加到[托管策略](#)
- 用户或角色可访问哪些服务
- 他们上次访问服务的时间
- 他们上次尝试使用特定 Amazon EC2、IAM、Lambda 或 Amazon S3 操作的时间

要使用 Amazon CLI 或 Amazon API 查看 IAM 的上次访问信息，您必须具有与要使用的操作匹配的权限：

- iam:GenerateServiceLastAccessedDetails
- iam:GetServiceLastAccessedDetails
- iam:GetServiceLastAccessedDetailsWithEntities
- iam:ListPoliciesGrantingServiceAccess

此示例说明如何创建基于身份的策略以允许查看 IAM 上次访问的信息。此外，它还允许对所有 IAM 进行只读访问。此策略定义了程序访问和控制台访问的权限。

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

Amazon Organizations 信息的权限

要使用 IAM 控制台查看 Organizations 中的根、OU 或账户实体的报告，您必须具有包含以下操作的策略：

- iam:GenerateOrganizationsAccessReport
- iam:GetOrganizationsAccessReport
- organizations:DescribeAccount
- organizations:DescribeOrganization
- organizations:DescribeOrganizationalUnit
- organizations:DescribePolicy
- organizations:ListChildren
- organizations:ListParents
- organizations:ListPoliciesForTarget
- organizations:ListRoots
- organizations:ListTargetsForPolicy

要使用 Amazon CLI 或 Amazon API 查看 Organizations，的上次访问的服务信息，您必须具有包含以下操作的策略：

- iam:GenerateOrganizationsAccessReport
- iam:GetOrganizationsAccessReport
- organizations:DescribePolicy
- organizations:ListChildren
- organizations:ListParents
- organizations:ListPoliciesForTarget
- organizations:ListRoots
- organizations:ListTargetsForPolicy

此示例说明如何创建基于身份的策略以允许查看 Organizations 的服务上次访问信息。此外，它还允许对所有 Organizations 进行只读访问。此策略定义了程序访问和控制台访问的权限。

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

```

```
        "iam:GetOrganizationsAccessReport",
        "organizations:Describe*",
        "organizations:List*"
    ],
    "Resource": "*"
}
```

您也可以使用 [iam:OrganizationsPolicyId \(p. 1015\)](#) 条件键以仅允许为特定 Organizations 策略生成报告。有关策略示例，请参阅 [IAM：查看 Organizations 策略的上次访问的服务信息 \(p. 427\)](#)。

IAM 和 Organizations 实体的故障排除活动

在某些情况下，Amazon Web Services Management Console 上次访问的信息表可能为空。或者，Amazon CLI 或 Amazon API 请求可能会返回空信息集或空字段。在这些情况下，请检查以下问题：

- 对于上次访问的操作信息，您期望看到的操作可能不会在列表中返回。这可能是由于 IAM 实体没有操作权限，或者 Amazon 尚未跟踪操作以获取上次访问的信息。
- 对于 IAM 用户，请确保该用户直接或通过组成员资格附加了至少一个内联或托管策略。
- 对于 IAM 组，请确认该组附加了至少一个内联或托管策略。
- 对于 IAM 组，该报告仅返回使用组的策略访问服务的成员的上次访问的服务信息。要了解成员是否使用了其他策略，请查看该用户的上次访问的信息。
- 对于 IAM 角色，请确认该角色附加了至少一个内联或托管策略。
- 对于 IAM 实体（用户或角色），请检查可能影响该实体的权限的其他策略类型。其中包含基于资源的策略、访问控制列表、Amazon Organizations 策略、IAM 权限边界或会话策略。有关更多信息，请参阅 [策略类型 \(p. 355\)](#) 或 [评估单个账户中的策略 \(p. 973\)](#)。
- 对于 IAM policy，请确保指定的托管策略附加到至少一个用户、具有成员的组或角色。
- 对于 Organizations 实体（根、OU 或账户），请确保您使用 Organizations 管理账户凭证登录。
- 确认在组织根中启用了 SCP。
- 操作上次访问的信息仅适用于某些 Amazon EC2、IAM、Lambda 和 Amazon S3 操作。

在进行更改时，请等待至少四小时，以便在 IAM 控制台报告中显示活动。如果您使用 Amazon CLI 或 Amazon API，则必须生成新的报告以查看更新后的信息。

Amazon 跟踪上次访问信息的位置

Amazon 收集标准 Amazon 区域的上次访问信息。在 Amazon 添加额外的区域时，这些区域将添加到下表中，包括 Amazon 开始在每个区域中跟踪信息的日期：

- Service information（服务信息）- 服务的跟踪周期为过去 400 天，如果您的区域在上一年内开始支持此功能，则不超过 400 天。
- 操作信息 - Amazon S3 管理操作的跟踪周期从 2020 年 4 月 12 日开始。Amazon EC2、IAM 和 Lambda 管理操作的跟踪周期从 2021 年 4 月 7 日开始。如果区域在稍后某个日期开始支持某项活动，则该日期也是该区域的操作跟踪开始日期。

区域名称	区域	跟踪开始日期
美国东部（俄亥俄州）	us-east-2	2017 年 10 月 27 日
美国东部（弗吉尼亚州北部）	us-east-1	2015 年 10 月 1 日
美国西部（北加利福尼亚）	us-west-1	2015 年 10 月 1 日

区域名称	区域	跟踪开始日期
美国西部 (俄勒冈州)	us-west-2	2015 年 10 月 1 日
亚太地区 (香港)	ap-east-1	2019 年 4 月 24 日
亚太地区 (雅加达)	ap-southeast-3	2021 年 12 月 13 日
亚太地区 (孟买)	ap-south-1	2016 年 6 月 27 日
亚太地区 (首尔)	ap-northeast-2	2016 年 1 月 6 日
亚太地区 (新加坡)	ap-southeast-1	2015 年 10 月 1 日
Asia Pacific (Sydney)	ap-southeast-2	2015 年 10 月 1 日
亚太地区 (东京)	ap-northeast-1	2015 年 10 月 1 日
Canada (Central)	ca-central-1	2017 年 10 月 28 日
欧洲 (法兰克福)	eu-central-1	2015 年 10 月 1 日
欧洲 (斯德哥尔摩)	eu-north-1	2018 年 12 月 12 日
Europe (Ireland)	eu-west-1	2015 年 10 月 1 日
欧洲 (伦敦)	eu-west-2	2017 年 10 月 28 日
欧洲 (米兰)	eu-south-1	2020 年 4 月 28 日
欧洲 (巴黎)	eu-west-3	2017 年 12 月 18 日
中东 (巴林)	me-south-1	2019 年 7 月 29 日
中东 (阿联酋)	me-central-1	2022 年 8 月 30 日
非洲 (开普敦)	af-south-1	2020 年 4 月 22 日
South America (São Paulo)	sa-east-1	2015 年 12 月 11 日

如果某个区域未在上表中列出，则表明此区域尚不提供上次访问的信息。

Amazon 区域是地理区域中的 Amazon 资源集合。区域分组为分区。标准区域是属于 aws 分区的区域。有关不同分区的更多信息，请参阅 Amazon 一般参考中的 [Amazon Resource Name \(ARN\) 格式](#)。有关区域的更多信息，也可参阅 Amazon 一般参考中的[关于 Amazon 区域](#)。

查看 IAM 的上次访问信息

您可以使用 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 查看 IAM 的上次访问信息。上次访问信息包括有关 Amazon EC2、IAM、Lambda 和 Amazon S3 上次访问的某些操作的信息。有关上次访问信息的更多信息，请参阅[使用上次访问的信息优化 Amazon 中的权限 \(p. 467\)](#)。

您可以在 IAM 中查看每种类型的资源的信息。在每种情况下，该信息包括给定报告周期允许的服务：

- User (用户) - 查看有关用户上次尝试访问每个允许的服务的信息。
- User group (用户组) - 查看有关用户组成员上次尝试访问每个允许的服务的信息。此报告还包括已尝试访问的成员的总数。
- Role (角色) - 查看有关某个人上次使用角色尝试访问每个允许的服务的信息。
- Policy (策略) - 查看有关用户或角色上次尝试访问每个允许的服务的信息。此报告还包括已尝试访问的实体的总数。

Note

在 IAM 中查看资源的访问信息之前，请确保您了解信息的报告周期、报告的实体和已评估的策略类型。有关更多信息，请参阅 [the section called “关于上次访问的信息的知识” \(p. 468\)](#)。

查看 IAM (控制台) 的信息

您可以在 IAM 控制台的 Access Advisor (访问顾问) 选项卡上查看 IAM 的上次访问信息。

查看 IAM (控制台) 的信息

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Groups (组)、Users (用户)、Roles (角色) 或 Policies (策略)。
3. 请选择任意用户、用户组、角色或策略名称以打开其 Summary (摘要) 页面并选择 Access Advisor (访问顾问) 选项卡。根据您选择的资源查看以下信息：
 - User group (用户组) - 查看用户组成员 (用户) 可以访问的服务列表。您还可以查看成员上次访问服务的时间、成员使用的用户组策略以及发出请求的用户组成员。请选择策略的名称以了解它是托管策略还是内联用户组策略。请选择用户组成员的名称以查看该用户组的所有成员以及他们上次访问服务的时间。
 - User (用户) - 查看用户可以访问的服务列表。您还可以查看他们上次访问服务的时间以及他们使用的策略。请选择策略的名称以了解它是托管策略、内联用户策略还是用户组的内联策略。
 - Role (角色) - 查看角色可以访问的服务的列表、角色上次访问服务的时间以及所使用的策略。选择策略的名称以了解它是托管策略还是内联角色策略。
 - Policy (策略) - 查看策略中允许的操作的服务列表。还可以查看上次使用策略访问服务的时间以及使用该策略的实体 (用户或角色)。选择实体的名称以了解哪些实体已附加此策略以及实体上次访问服务的时间。
4. (可选) 在表的 Service (服务) 列中，选择 Amazon EC2、Amazon Identity and Access Management、Amazon Lambda 或 Amazon S3 以查看 IAM 实体尝试访问的管理操作的列表。您可以查看 Amazon 区域以及显示某人上次尝试执行操作的时间戳。
5. Last accessed (上次访问) 列显示服务以及 Amazon EC2、IAM、Lambda 和 Amazon S3 管理操作。查看此列中返回的以下可能结果。这些结果会有所不同，具体取决于是否允许某个服务或操作、是否访问了此服务或操作，以及 Amazon 是否跟踪此服务或操作以获取上次访问的信息。

<number of> 天前

自跟踪周期内使用服务或操作以来的天数。服务的跟踪周期为过去 400 天。Amazon S3 操作的跟踪周期从 2020 年 4 月 12 日开始。Amazon EC2、IAM Lambda 和操作的跟踪周期从 2021 年 4 月 7 日开始。要了解有关每个 Amazon 区域的跟踪开始日期的更多信息，请参阅[Amazon 跟踪上次访问信息的位置 \(p. 471\)](#)。

在跟踪周期内未访问

所跟踪的服务或操作在跟踪周期内未被实体使用。

您可能对未出现在列表中的操作拥有权限。如果操作的跟踪信息当前不受 Amazon 支持，则可能会发生这种情况。您不应仅出于缺少跟踪信息来做出权限决定。相反，我们建议您使用此信息来告知和支持授予最小权限的总体策略。检查您的策略以确认访问级别是否适当。

查看 IAM 的信息 (Amazon CLI)

您可以使用 Amazon CLI 检索有关上次使用 IAM 资源尝试访问 Amazon 服务和 Amazon S3、Amazon EC2、IAM 以及 Lambda 操作的信息。IAM 资源可能是用户、用户组、角色或策略。

查看 IAM 的信息 (Amazon CLI)

1. 生成报告。请求必须包括要报告的 IAM 资源（用户、用户组、角色或策略）的 ARN。您可以指定要在报告中生成的粒度级别，以查看服务或服务 and 操作的访问详细信息。该请求返回一个 job-id，您之后可在 `get-service-last-accessed-details` 和 `get-service-last-accessed-details-with-entities` 操作中使用它来监控 job-status，直到作业完成。
 - [aws iam generate-service-last-accessed-details](#)
2. 检索有关使用上一步中的 job-id 参数的报告的详细信息。
 - [aws iam get-service-last-accessed-details](#)

此操作根据您在 `generate-service-last-accessed-details` 操作中请求的资源类型和粒度级别返回以下信息：

- User（用户）- 返回指定用户可访问的服务的列表。对于每个服务，此操作返回用户上次尝试的日期和时间以及用户的 ARN。
 - 用户组 — 返回指定用户组的成员可使用附加到用户组的策略访问的服务列表。对于每个服务，此操作返回任何用户组成员（用户）上次尝试的日期和时间。它还返回该用户的 ARN 以及已尝试访问服务的用户组成员的总数。使用 [GetServiceLastAccessedDetailsWithEntities](#) 操作可检索所有成员的列表。
 - Role（角色）- 返回指定角色可访问的服务的列表。对于每个服务，此操作返回角色上次尝试的日期和时间以及角色的 ARN。
 - Policy（策略）- 返回指定策略允许访问的服务的列表。对于每个服务，此操作返回实体（用户或角色）上次尝试使用策略访问服务的日期和时间。它还返回实体的 ARN 以及已尝试访问的实体的总数。
3. 了解有关在尝试访问特定服务时使用用户组或策略权限的实体的更多信息。此操作返回具有每个实体的 ARN、ID、名称、路径、类型（用户或角色）的实体列表以及实体上次尝试访问服务的时间。您还可以对用户和角色使用此操作，但它仅返回有关该实体的信息。
 - [aws iam get-service-last-accessed-details-with-entities](#)
 4. 了解有关在尝试访问特定服务时身份（用户、用户组或角色）使用的基于身份的策略的更多信息。在指定身份和服务时，此操作返回身份可用于访问指定服务的权限策略的列表。此操作提供策略的当前状态，而不依赖于生成的报告。它也不返回其他策略类型，例如基于资源的策略、访问控制列表、Amazon Organizations 策略、IAM 权限边界或会话策略。有关更多信息，请参阅 [策略类型](#) (p. 355) 或 [评估单个账户中的策略](#) (p. 973)。
 - [aws iam list-policies-granting-service-access](#)

查看 IAM 的信息 (Amazon API)

您可以使用 Amazon API 检索有关上次使用 IAM 资源尝试访问 Amazon 服务和 Amazon S3、Amazon EC2、IAM 以及 Lambda 操作的信息。IAM 资源可能是用户、用户组、角色或策略。您可以指定要在报告中生成的粒度级别，以查看服务或服务 and 操作的详细信息。

查看 IAM 的信息 (Amazon API)

1. 生成报告。请求必须包括要报告的 IAM 资源（用户、用户组、角色或策略）的 ARN。它返回一个 JobId，您之后可在 `GetServiceLastAccessedDetails` 和 `GetServiceLastAccessedDetailsWithEntities` 操作中使用它来监控 JobStatus，直到作业完成。
 - [GenerateServiceLastAccessedDetails](#)
2. 检索有关使用上一步中的 JobId 参数的报告的详细信息。

- [GetServiceLastAccessedDetails](#)

此操作根据您在 `GenerateServiceLastAccessedDetails` 操作中请求的资源类型和粒度级别返回以下信息：

- User (用户) - 返回指定用户可访问的服务的列表。对于每个服务，此操作返回用户上次尝试的日期和时间以及用户的 ARN。
 - 用户组 — 返回指定用户组的成员可使用附加到用户组的策略访问的服务列表。对于每个服务，此操作返回任何用户组成员 (用户) 上次尝试的日期和时间。它还返回该用户的 ARN 以及已尝试访问服务的用户组成员的总数。使用 [GetServiceLastAccessedDetailsWithEntities](#) 操作可检索所有成员的列表。
 - Role (角色) - 返回指定角色可访问的服务的列表。对于每个服务，此操作返回角色上次尝试的日期和时间以及角色的 ARN。
 - Policy (策略) - 返回指定策略允许访问的服务的列表。对于每个服务，此操作返回实体 (用户或角色) 上次尝试使用策略访问服务的日期和时间。它还返回实体的 ARN 以及已尝试访问的实体的总数。
3. 了解有关在尝试访问特定服务时使用用户组或策略权限的实体的更多信息。此操作返回具有每个实体的 ARN、ID、名称、路径、类型 (用户或角色) 的实体列表以及实体上次尝试访问服务的时间。您还可以对用户和角色使用此操作，但它仅返回有关该实体的信息。

- [GetServiceLastAccessedDetailsWithEntities](#)

4. 了解有关在尝试访问特定服务时身份 (用户、用户组或角色) 使用的基于身份的策略的更多信息。在指定身份和服务时，此操作返回身份可用于访问指定服务的权限策略的列表。此操作提供策略的当前状态，而不依赖于生成的报告。它也不返回其他策略类型，例如基于资源的策略、访问控制列表、Amazon Organizations 策略、IAM 权限边界或会话策略。有关更多信息，请参阅 [策略类型 \(p. 355\)](#) 或 [评估单个账户中的策略 \(p. 973\)](#)。

- [ListPoliciesGrantingServiceAccess](#)

查看 Organizations 的上次访问信息

您可以使用 IAM 控制台、Amazon CLI 或 Amazon API 查看 Amazon Organizations 的上次访问的服务信息。有关数据、所需的权限、故障排除和支持的区域的重要信息，请参阅[使用上次访问的信息优化 Amazon 中的权限 \(p. 467\)](#)。

使用 Amazon Organizations 管理账户凭证登录到 IAM 控制台时，您可以查看企业中任何实体的信息。企业实体包括企业根、企业单元 (OU) 和账户。您还可以使用 IAM 控制台查看企业中的任何服务控制策略 (SCP) 的信息。IAM 显示适用于实体的任何 SCP 允许的服务列表。对于每个服务，您可以查看所选的 Organizations 实体或其子实体的最新账户活动信息。

在将 Amazon CLI 或 Amazon API 与管理账户凭证一起使用时，您可以为企业中的任何实体或策略生成报告。实体的编程报告包括适用于该实体的任何 SCP 允许的服务列表。对于每个服务，该报告包括指定的 Organizations 实体或其子树中的账户的最新活动。

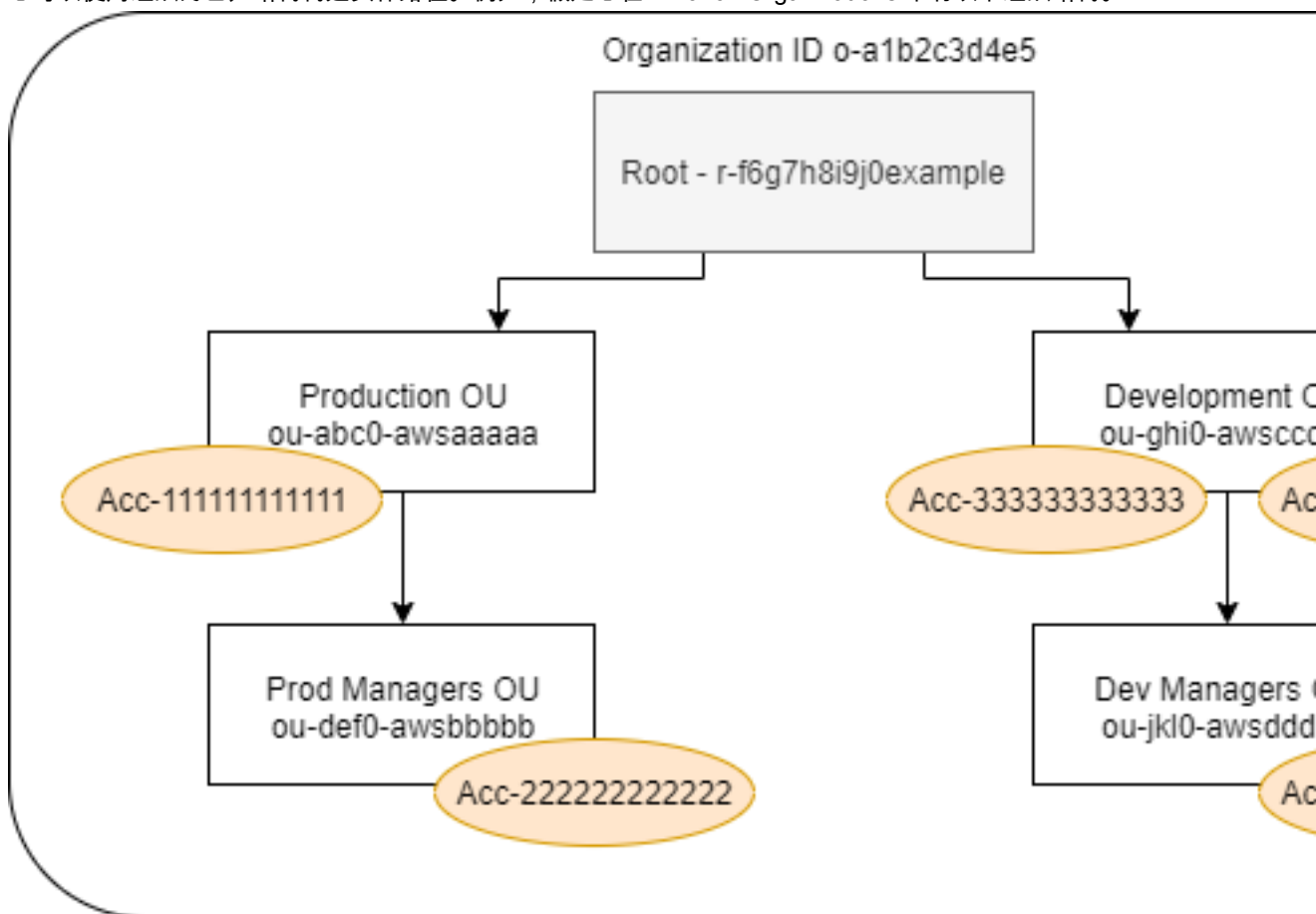
在为策略生成编程报告时，您必须指定一个 Organizations 实体。该报告包括指定的 SCP 允许的服务列表。对于每个服务，它包括通过该策略授予权限的实体或其子实体中的最新账户活动。有关更多信息，请参阅 [aws iam generate-organizations-access-report](#) 或 [GenerateOrganizationsAccessReport](#)。

在查看报告之前，请确保您了解管理账户要求和信息、报告周期、报告的实体和评估的策略类型。有关更多信息，请参阅 [the section called “关于上次访问的信息的知识” \(p. 468\)](#)。

了解 Amazon Organizations 实体路径

使用 Amazon CLI 或 Amazon API 生成 Amazon Organizations 访问报告时，必须指定实体路径。路径是 Organizations 实体结构的文本表示形式。

您可以使用组织的已知结构构建实体路径。例如，假定您在 Amazon Organizations 中有以下组织结构。



Dev Managers OU 的路径是使用组织 ID、根目录及该路径下的所有 OU（包括 Dev Managers OU）构建的。

```
o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-ghi0-awscccc/ou-jkl0-awsdddd
```

Production OU 中账户的路径是使用组织 ID、根目录、该 OU 和账号构建的。

```
o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-abc0-awsaaaa/111111111111
```

Note

组织 ID 是全局唯一的，但 OU ID 和根目录 ID 仅在组织内是唯一的。这意味着没有两个组织具有相同的组织 ID。但是，另一个组织可能具有与您的组织相同的 OU 或根目录 ID。我们建议您在指定 OU 或根目录时始终包含组织 ID。

查看 Organizations 的信息（控制台）

您可以使用 IAM 控制台查看根、OU、账户或策略的上次访问的服务信息。

查看根目录的信息（控制台）

1. 使用 Organizations 管理账户凭证登录到 Amazon Web Services Management Console，然后打开 IAM 控制台，地址：<https://console.aws.amazon.com/iam/>。

2. 在导航窗格中的 Access reports (访问报告) 部分下, 选择 Organization activity (组织活动)。
3. 在 Organization activity (组织活动) 页面上, 选择 Root (根)。
4. 在 Details and activity (详细信息和活动) 选项卡上, 查看 Service access report (服务访问报告) 部分。该信息包括直接附加到根的策略允许的服务列表。该信息显示您上次从中访问服务的账户以及访问时间。有关访问服务的主体的更多详细信息, 请以该账户中的管理员身份登录并[查看 IAM 服务上次访问的服务信息 \(p. 472\)](#)。
5. 请选择 Attached SCPs (附加的 SCP) 选项卡以查看附加到根的服务控制策略 (SCP) 列表。IAM 显示您可以查看每个策略附加到的目标实体数量。您可以使用该信息确定要检查的 SCP。
6. 请选择一个 SCP 名称以查看该策略允许的所有服务。对于每个服务, 查看上次从中访问该服务的账户以及访问时间。
7. 选择 Edit in Amazon Organizations (在 Amazon Organizations 中编辑) 以查看其他详细信息并在 Organizations 控制台中编辑 SCP。有关更多信息, 请参阅 Amazon Organizations 用户指南中的[更新 SCP](#)。

查看 OU 或账户的信息 (控制台)

1. 使用 Organizations 管理账户凭证登录到 Amazon Web Services Management Console, 然后打开 IAM 控制台, 地址: <https://console.aws.amazon.com/iam/>。
2. 在导航窗格中的 Access reports (访问报告) 部分下, 选择 Organization activity (组织活动)。
3. 在 Organization activity (组织活动) 页面上, 展开您的组织的结构。接下来, 请选择要查看的 OU 或任何账户的名称, 但管理账户除外。
4. 在 Details and activity (详细信息和活动) 选项卡上, 查看 Service access report (服务访问报告) 部分。该信息包括附加到 OU 或账户及其所有父项的 SCP 允许的服务列表。该信息显示您上次从中访问服务的账户以及访问时间。有关访问服务的主体的更多详细信息, 请以该账户中的管理员身份登录并[查看 IAM 服务上次访问的服务信息 \(p. 472\)](#)。
5. 请选择 Attached SCPs (附加的 SCP) 选项卡以查看直接附加到 OU 或账户的服务控制策略 (SCP) 列表。IAM 显示您可以查看每个策略附加到的目标实体数量。您可以使用该信息确定要检查的 SCP。
6. 请选择一个 SCP 名称以查看该策略允许的所有服务。对于每个服务, 查看上次从中访问该服务的账户以及访问时间。
7. 选择 Edit in Amazon Organizations (在 Amazon Organizations 中编辑) 以查看其他详细信息并在 Organizations 控制台中编辑 SCP。有关更多信息, 请参阅 Amazon Organizations 用户指南中的[更新 SCP](#)。

查看管理账户的信息 (控制台)

1. 使用 Organizations 管理账户凭证登录到 Amazon Web Services Management Console, 然后打开 IAM 控制台, 地址: <https://console.aws.amazon.com/iam/>。
2. 在导航窗格中的 Access reports (访问报告) 部分下, 选择 Organization activity (组织活动)。
3. 在 Organization activity (企业活动) 页面上, 展开您的企业的结构, 然后选择管理账户的名称。
4. 在 Details and activity (详细信息和活动) 选项卡上, 查看 Service access report (服务访问报告) 部分。该信息包括所有 Amazon 服务的列表。管理账户不受 SCP 限制。该信息显示账户上次是否访问服务以及访问时间。有关访问服务的主体的更多详细信息, 请以该账户中的管理员身份登录并[查看 IAM 服务上次访问的服务信息 \(p. 472\)](#)。
5. 请选择 Attached SCPs (附加的 SCP) 选项卡以确认没有附加的 SCP, 因为该账户是管理账户。

查看策略的信息 (控制台)

1. 使用 Organizations 管理账户凭证登录到 Amazon Web Services Management Console, 然后打开 IAM 控制台, 地址: <https://console.aws.amazon.com/iam/>。
2. 在导航窗格中的 Access reports (访问报告) 部分下, 选择 Service control policies (SCPs) (服务控制策略(SCP))。

3. 在 Service control policies (SCPs) (服务控制策略 (SCP)) 页面上，查看您组织中的策略列表。您可以查看每个策略附加到的目标实体数量。
4. 请选择一个 SCP 名称以查看该策略允许的所有服务。对于每个服务，查看上次从中访问该服务的账户以及访问时间。
5. 选择 Edit in Amazon Organizations (在 Amazon Organizations 中编辑) 以查看其他详细信息并在 Organizations 控制台中编辑 SCP。有关更多信息，请参阅 Amazon Organizations 用户指南中的[更新 SCP](#)。

查看 Organizations 的信息 (Amazon CLI)

您可以使用 Amazon CLI 检索 Organizations 根、OU、账户或策略的上次访问的服务信息。

查看 Organizations 服务的上次访问信息 (Amazon CLI)

1. 使用具有所需的 IAM 和 Organizations 权限的 Organizations 管理账户凭证，并确认为根启用了 SCP。有关更多信息，请参阅[关于上次访问的信息的知识 \(p. 468\)](#)。
2. 生成报告。请求必须包含要生成报告的 Organizations 实体（根、OU 或账户）的路径。您可以选择包含 organization-policy-id 参数以查看特定策略的报告。该命令返回 job-id，您可以随后在 get-organizations-access-report 命令中使用该内容以监控 job-status，直到作业完成。
 - [aws iam generate-organizations-access-report](#)
3. 检索有关使用上一步中的 job-id 参数的报告的详细信息。
 - [aws iam get-organizations-access-report](#)

该命令返回实体成员可以访问的服务列表。对于每个服务，该命令返回账户成员的上次尝试日期和时间以及账户的实体路径。它还返回可访问的服务总数以及未访问的服务数。如果指定了可选的 organizations-policy-id 参数，则可访问的服务是指定策略允许的那些服务。

查看 Organizations 的信息 (Amazon API)

您可以使用 Amazon API 检索 Organizations 根、OU、账户或策略的上次访问的服务信息。

查看 Organizations 服务的上次访问信息 (Amazon API)

1. 使用具有所需的 IAM 和 Organizations 权限的 Organizations 管理账户凭证，并确认为根启用了 SCP。有关更多信息，请参阅[关于上次访问的信息的知识 \(p. 468\)](#)。
2. 生成报告。请求必须包含要生成报告的 Organizations 实体（根、OU 或账户）的路径。您可以选择包含 OrganizationsPolicyId 参数以查看特定策略的报告。该操作返回 JobId，您可以随后在 GetOrganizationsAccessReport 操作中使用该内容以监控 JobStatus，直到作业完成。
 - [GenerateOrganizationsAccessReport](#)
3. 检索有关使用上一步中的 JobId 参数的报告的详细信息。
 - [GetOrganizationsAccessReport](#)

该操作返回实体成员可以访问的服务列表。对于每个服务，该操作返回账户成员的上次尝试日期和时间以及账户的实体路径。它还返回可访问的服务总数以及未访问的服务数。如果指定了可选的 OrganizationsPolicyId 参数，则可访问的服务是指定策略允许的那些服务。

使用上次访问信息的示例应用场景

您可以使用上次访问的信息确定您为 IAM 实体或 Amazon Organizations 实体授予的权限。有关更多信息，请参阅[使用上次访问的信息优化 Amazon 中的权限 \(p. 467\)](#)。

Note

在 IAM 或 Amazon Organizations 中查看实体或策略的访问信息之前，请确保您了解信息的报告周期、报告的实体和评估的策略类型。有关更多信息，请参阅 [the section called “关于上次访问的信息的知识” \(p. 468\)](#)。

作为管理员，您可以平衡适合您的公司的可访问性和最小权限。

使用信息减少 IAM 组的权限

您可以使用上次访问的信息减少 IAM 组权限，以仅包含用户所需的服务。此方法是在服务级别[授予最小权限 \(p. 754\)](#)的重要步骤。

例如，Paulo Santos 是负责为 Example Corp 定义 Amazon 用户权限的管理员。这家公司刚刚开始使用 Amazon，而软件开发团队尚未定义他们将使用何种 Amazon 服务。Paulo 希望仅向团队授予访问他们所需的服务的权限，但由于尚未定义，因此他暂时向团队授予高级用户权限。然后，他使用上次访问的信息来减少组的权限。

Paulo 使用以下 JSON 文本创建名为 ExampleDevelopment 的托管策略。然后，他将该策略附加到名为 Development 的组，并将所有开发人员添加到该组。

Note

Paulo 的高级用户可能需要 iam:CreateServiceLinkedRole 权限才能使用某些服务和功能。他知道添加此权限将允许用户创建任何服务相关角色。他为高级用户接受这种风险。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FullAccessToAllServicesExceptPeopleManagement",
      "Effect": "Allow",
      "NotAction": [
        "iam:*",
        "organizations:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "RequiredIamAndOrgsActions",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:ListRoles",
        "organizations:DescribeOrganization"
      ],
      "Resource": "*"
    }
  ]
}
```

Paulo 决定等待 90 天，之后再使用 Amazon Web Services Management Console[查看上次访问的信息 \(p. 473\)](#) (适用于 Development 组)。他查看组成员访问过的服务的列表。他了解到，用户上周访问了 5 项服务：Amazon CloudTrail、Amazon CloudWatch Logs、Amazon EC2、Amazon KMS 和 Amazon S3。他们在首次评估 Amazon 时访问过其他几项服务，但之后便未访问。

Paulo 决定减少策略权限以仅包含上述 5 项服务以及所需的 IAM 和 Organizations 操作。他使用以下 JSON 文本编辑 ExampleDevelopment 策略。

Note

Paulo 的高级用户可能需要 `iam:CreateServiceLinkedRole` 权限才能使用某些服务和功能。他知道添加此权限将允许用户创建任何服务相关角色。他为高级用户接受这种风险。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FullAccessToListedServices",
      "Effect": "Allow",
      "Action": [
        "s3:*",
        "kms:*",
        "cloudtrail:*",
        "logs:*",
        "ec2:*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "RequiredIamAndOrgsActions",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:ListRoles",
        "organizations:DescribeOrganization"
      ],
      "Resource": "*"
    }
  ]
}
```

为了进一步减少权限，Paulo 可在 Amazon CloudTrail Event history (事件历史记录) 中查看账户的事件。在那里，他可以查看可用于减少策略权限的详细事件信息，以仅包括开发人员所需的操作和资源。有关更多信息，请参阅 Amazon CloudTrail 用户指南中的[在 CloudTrail 控制台中查看 CloudTrail 事件](#)。

使用信息减少 IAM 用户的权限

您可以使用上次访问的信息来减少单个 IAM 用户的权限。

例如，Martha Rivera 是 IT 管理员，她负责确保公司的人员没有多余的 Amazon 权限。在定期安检中，她检查所有 IAM 用户的权限。这些用户中，有一个是名叫 Nikhil Jayashankar 的应用程序开发人员，以前担任过安全工程师。由于工作要求发生变化，Nikhil 成为了 app-dev 组和 security-team 组的成员。适用于他的新工作的 app-dev 组授予了对多项服务（包括 Amazon EC2、Amazon EBS、Auto Scaling、Amazon S3、Route 53 和 Elastic Transcoder）的权限。他的旧工作的 security-team 组授予了对 IAM 和 CloudTrail 的权限。

Martha 以管理员身份登录 IAM 控制台，并依次选择 Users（用户）、姓名 nikhilj 和 Access Advisor（访问顾问）选项卡。

Martha 查看 Last Accessed（上次访问时间）列，发现 Nikhil 最近未访问过 IAM、CloudTrail、Route 53、Amazon Elastic Transcoder 以及很多其他 Amazon 服务。Nikhil 已访问 Amazon S3。Martha 从服务列表中选择 S3，并了解到 Nikhil 在过去两周内执行了一些 S3 List 操作。在其公司内部，Martha 确认 Nikhil 不再有访问 IAM 和 CloudTrail 的业务需要，因为他不再是内部安全团队的成员。

Martha 管理员现在准备好处理上次访问的服务和操作信息。但是，与上一个示例中的组不同，像 nikhilj 这样的 IAM 用户可能受多个策略的约束并且是多个组的成员。Martha 必须谨慎行事，以避免无意中破坏

nikhilj 或其他组成员的访问权限。除了了解 Nikhil 应拥有的访问权限之外，她还必须确定他如何接收这些权限。

Martha 选择 Permissions (权限) 选项卡，可在其中查看直接附加到 nikhilj 的策略以及从组附加的策略。她扩展每个策略并查看策略摘要，以了解哪个策略允许访问 Nikhil 未使用的服务：

- IAM – IAMFullAccess Amazon 托管策略直接附加到 nikhilj 并附加到 security-team 组。
- CloudTrail - AmazonCloudTrailReadOnlyAccess Amazon 托管策略附加到 security-team 组。
- Route 53 – App-Dev-Route53 客户托管策略附加到 app-dev 组。
- Elastic Transcoder - App-Dev-ElasticTranscoder 客户托管策略附加到 app-dev 组。

Martha 决定删除直接附加到 nikhilj 的 IAMFullAccess Amazon 托管策略。她还删除 Nikhil 的 security-team 组成员资格。这两项操作删除了对 IAM 和 CloudTrail 的不必要的访问权限。

Nikhil 对 Route 53 和 Elastic Transcoder 的访问权限由 app-dev 组授予。虽然 Nikhil 未使用这些服务，但组的其他成员可能使用了这些服务。Martha 查看 app-dev 组的上次访问信息，并了解到有几个成员最近访问了 Route 53 和 Amazon S3。但在过去一年中，没有任何组成员访问过 Elastic Transcoder。她从组中删除 App-Dev-ElasticTranscoder 客户托管策略。

然后，Martha 查看 App-Dev-ElasticTranscoder 客户托管策略的上次访问的信息。她了解到该策略未附加到任何其他 IAM 身份。她在公司内部进行调查，以确保将来不再需要该策略，然后将其删除。

在删除 IAM 资源之前使用信息

在删除 IAM 资源之前，您可以使用上次访问的信息，以确保自上次使用该资源以来已经过了一定的时间。这适用于用户、组、角色和策略。要了解这些操作的更多信息，请参阅以下主题：

- Users (用户) - [Deleting a user \(p. 80\)](#) (删除用户)
- Groups (组) - [Deleting a group \(p. 156\)](#) (删除组)
- Roles (角色) - [Deleting a role \(p. 271\)](#) (删除角色)
- Policies (策略) - [Deleting a managed policy \(this also detaches the policy from identities \) \(p. 465\)](#) (删除托管策略 (这还会将策略与身份分离))

在编辑 IAM 策略之前使用信息

您可以先查看 IAM 身份 (用户、组或角色) 或 IAM 策略的上次访问信息，然后再编辑影响该资源的策略。这很重要，因为您不想删除正在使用它的人的访问权限。

例如，Arnav Desai 是 Example Corp 的开发人员和 Amazon 管理员。当他的团队开始使用 Amazon，他们为所有开发人员提供了高级用户访问权限，允许他们完全访问除 IAM 和 Organizations 之外的所有服务。作为 [授予最小权限 \(p. 754\)](#) 操作的第一步，Arnav 希望使用 Amazon CLI 来查看其账户中的托管策略。

为此，Arnav 首先使用以下命令在其账户中列出附加到身份的客户托管权限策略：

```
aws iam list-policies --scope Local --only-attached --policy-usage-filter PermissionsPolicy
```

从响应中，他捕获了每个策略的 ARN。然后，Arnav 使用以下命令为每个策略生成上次访问的信息报告。

```
aws iam generate-service-last-accessed-details --arn arn:aws:iam::123456789012:policy/ExamplePolicy1
```

在该响应中，他从 JobId 字段中捕获了生成的报告的 ID。然后，Arnav 轮询以下命令，直到 JobStatus 字段返回 COMPLETED 或 FAILED 值。如果作业失败，他将捕获错误。

```
aws iam get-service-last-accessed-details --job-id 98a765b4-3cde-2101-2345-example678f9
```

当作业的状态为 COMPLETED 时，Arnav 将解析 JSON 格式的 ServicesLastAccessed 数组的内容。

```
"ServicesLastAccessed": [
  {
    "TotalAuthenticatedEntities": 1,
    "LastAuthenticated": 2018-11-01T21:24:33.222Z,
    "ServiceNamespace": "dynamodb",
    "LastAuthenticatedEntity": "arn:aws:iam::123456789012:user/IAMExampleUser",
    "ServiceName": "Amazon DynamoDB"
  },
  {
    "TotalAuthenticatedEntities": 0,
    "ServiceNamespace": "ec2",
    "ServiceName": "Amazon EC2"
  },
  {
    "TotalAuthenticatedEntities": 3,
    "LastAuthenticated": 2018-08-25T15:29:51.156Z,
    "ServiceNamespace": "s3",
    "LastAuthenticatedEntity": "arn:aws:iam::123456789012:role/IAMExampleRole",
    "ServiceName": "Amazon S3"
  }
]
```

通过此信息，Arnav 了解到 ExamplePolicy1 策略允许访问三项服务，即 Amazon DynamoDB、Amazon S3 和 Amazon EC2。名为 IAMExampleUser 的 IAM 用户上次于 11 月 1 日尝试访问过 DynamoDB，并且有人已于 8 月 25 日使用 IAMExampleRole 角色尝试访问 Amazon S3。此外，还有另外两个实体在去年尝试访问过 Amazon S3。但是，去年没有人尝试访问 Amazon EC2。

这意味着，Arnav 可从策略中安全地删除 Amazon EC2 操作。Arnav 想查看策略的当前 JSON 文档。首先，他必须使用以下命令确定策略的版本号。

```
aws iam list-policy-versions --policy-arn arn:aws:iam::123456789012:policy/ExamplePolicy1
```

从响应中，Arnav 收集 Versions 数组中的当前默认版本号。然后，他使用该版本号 (v2) 通过以下命令请求 JSON 策略文档。

```
aws iam get-policy-version --policy-arn arn:aws:iam::123456789012:policy/ExamplePolicy1 --version-id v2
```

Arnav 存储 PolicyVersion 数组的 Document 字段中返回的 JSON 策略文档。在该策略文档中，Arnav 在 ec2 命名空间中搜索操作。如果在该策略中没有其他命名空间中的操作，则他将该策略与受影响的身份（用户、组和角色）分离。然后，他删除该策略。在这种情况下，策略确实包含 Amazon DynamoDB 和 Amazon S3 服务。因此，Arnav 从文档中删除 Amazon EC2 操作并保存更改。然后，他使用以下命令通过文档的新版本更新策略并将该版本设置为默认策略版本。

```
aws iam create-policy-version --policy-arn arn:aws:iam::123456789012:policy/ExamplePolicy1 --policy-document file://UpdatedPolicy.json --set-as-default
```

ExamplePolicy1 策略现已更新以删除对不必要的 Amazon EC2 服务的访问权限。

其他 IAM 应用场景

有关 IAM 资源（用户、组、角色或策略）上次尝试访问服务的时间的信息可在您完成以下任何任务时为您提供帮助：

- Policies（策略）- [编辑现有的客户托管策略或内联策略以删除权限 \(p. 461\)](#)

- Policies (策略) - 将内联策略转换为托管策略, 然后将其删除 (p. 369)
- Policies (策略) - 向现有策略添加显式拒绝 (p. 980)
- Policies (策略) - 将托管策略与身份 (用户、组或角色) 分离 (p. 453)
- Entities (实体) - 设置权限边界以控制实体 (用户或角色) 可以拥有的最大权限 (p. 451)
- Groups (组) - 从组中删除用户 (p. 154)

使用信息来优化组织部门的权限

您可以在 Amazon Organizations 中使用上次访问的信息优化组织部门 (OU) 的权限。

例如, John Stiles 是一个 Amazon Organizations 管理员。他负责确保具有公司 Amazon 账户的人员没有多余的权限。作为定期安全审核的一部分, 他检查组织的权限。他的 Development OU 包含经常用于测试新 Amazon 服务的账户。John 决定定期查看超过 180 天未访问的服务的报告。然后, 他删除 OU 成员访问这些服务的权限。

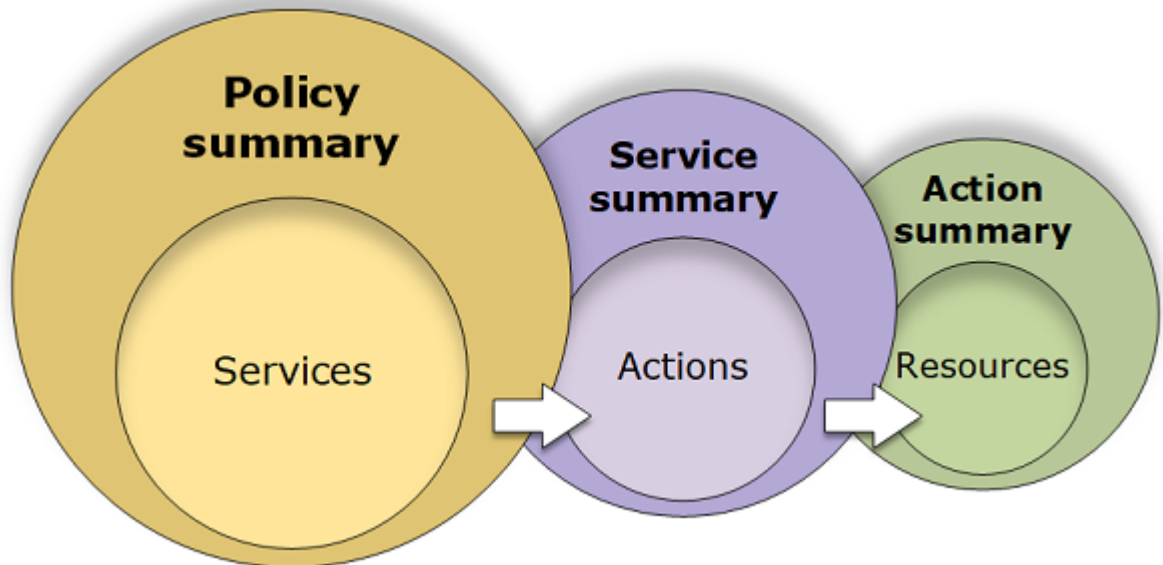
John 使用管理账户凭证登录到 IAM 控制台。在 IAM 控制台中, 他找到 Development OU 的 Organizations 数据。他查看 Service access report (服务访问报告) 表, 并发现两个超过 180 天 (首选的时段) 未访问的 Amazon 服务。他记得为开发团队添加了权限以访问 Amazon Lex 和 Amazon Database Migration Service。John 联系开发团队, 并确认他们不再有测试这些服务的业务需求。

John 现在已准备好处理上次访问的信息。他选择 Edit in Amazon Organizations (在 Amazon Organizations 中编辑), 系统提醒他 SCP 将附加到多个实体。他选择 Continue (继续)。在 Amazon Organizations 中, 他检查目标以了解 SCP 附加到哪些 Organizations 实体。所有实体均位于 Development OU 中。

John 决定在 NewServiceTest SCP 中拒绝访问 Amazon Lex 和 Amazon Database Migration Service 操作。该操作将删除这些对这些服务的非必要访问。

了解策略授予的权限

IAM 控制台中提供了策略摘要表, 这些表总结了策略中对每个服务允许或拒绝的访问级别、资源和条件。策略在三个表中概括: 策略摘要 (p. 484)、服务摘要 (p. 493) 和操作摘要 (p. 496)。策略摘要表包含服务列表。选择其中的服务可查看服务摘要。该摘要表包含所选服务的操作和关联权限的列表。您可以选择该表中的操作以查看操作摘要。该表包含所选操作的资源和条件列表。



您可以在用户或角色页上查看附加到该用户的所有策略 (托管和内联) 的策略摘要。可在 Policies 页面上查看所有托管策略的摘要。托管策略包括 Amazon 托管策略、Amazon 托管工作职能策略和客户托管策略。您可以在 Policies (策略) 页面上查看这些策略的摘要，无论它们是附加到用户还是其他 IAM 身份。

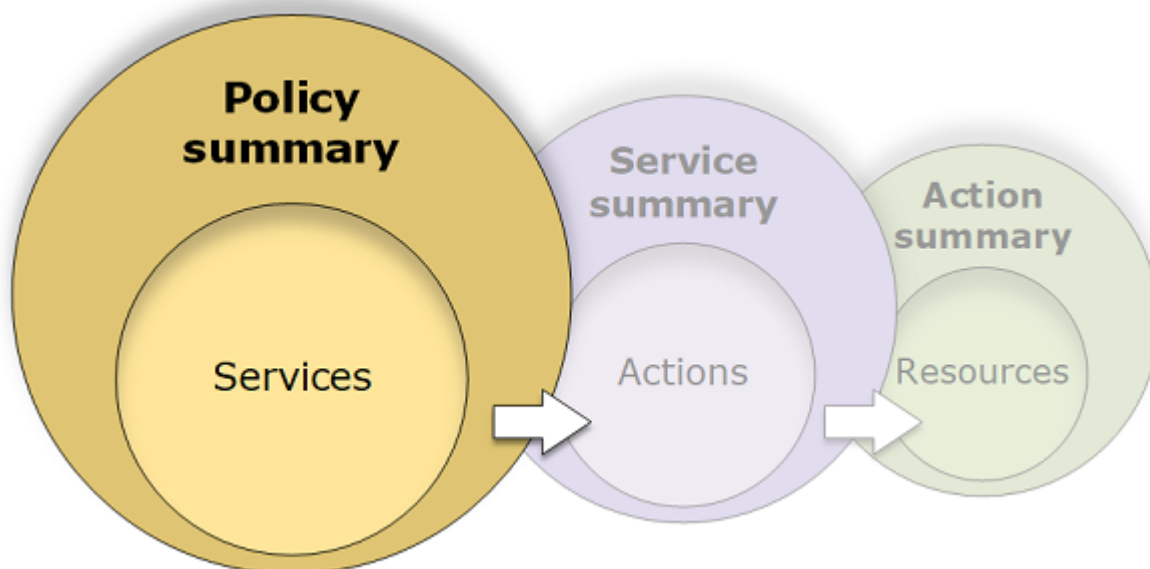
您可以使用策略摘要中的信息了解您的策略允许或拒绝的权限。策略摘要可帮助您[排查故障 \(p. 870\)](#)并修复未提供您期望的权限的策略。

主题

- [策略摘要 \(服务列表 \) \(p. 484\)](#)
- [服务摘要 \(操作列表 \) \(p. 493\)](#)
- [操作摘要 \(资源列表 \) \(p. 496\)](#)
- [策略摘要示例 \(p. 499\)](#)

策略摘要 (服务列表)

策略在三个表中概括：策略摘要、[服务摘要 \(p. 493\)](#)和[操作摘要 \(p. 496\)](#)。策略摘要表包括由所选策略定义的服务列表和权限摘要。



策略摘要表分为 Uncategorized services、Explicit deny 和 Allow 几部分。如果策略包含 IAM 无法识别的服务，则该服务将包含在表的 Uncategorized services (未分类服务) 部分中。如果 IAM 能够识别服务，则该服务将包含在表的 Explicit deny (显式拒绝) 或 Allow (允许) 部分下，具体取决于策略作用的结果 (Deny 或 Allow)。

查看策略摘要

您可以在 Users 页面上查看附加到用户的任何策略的摘要。您可以在 Roles 页面上查看附加到角色的任何策略的摘要。可以在 Policies 页面上查看托管策略的策略摘要。如果您的策略不包含策略摘要，请参阅[缺少策略摘要 \(p. 873\)](#)了解原因。

从 Policies 页面查看策略摘要

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。

2. 在导航窗格中，选择 Policies (策略)。
3. 在策略列表中，选择要查看的策略的名称。
4. 在策略的 Summary 页面上，查看 Permissions 选项卡以查看策略摘要。

查看附加到用户的策略的摘要

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 从导航窗格中选择 Users。
3. 在用户列表中，选择要查看其策略的用户的名称。
4. 在用户的 Summary 页面上，查看 Permissions 选项卡，以查看直接附加到用户或从组附加到用户的策略列表。
5. 在用户的策略表中，展开要查看的策略的行。

查看附加到角色的策略的摘要

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择角色。
3. 在角色列表中，选择要查看其策略的角色的名称。
4. 在角色的 Summary 页面上，查看 Permissions 选项卡，以查看附加到角色的策略列表。
5. 在角色的策略表中，展开要查看的策略的行。

编辑策略以纠正警告

在查看策略摘要时，您可能会发现拼写错误，或者注意到策略未提供所需的权限。您无法直接编辑策略摘要。不过，您可以使用可视化策略编辑器编辑客户管理的策略，该编辑器捕获策略摘要报告的很多相同的错误和警告。然后，您可以在策略摘要中查看更改以确认纠正了所有问题。要了解如何编辑内联策略，请参阅 [the section called “编辑 IAM 策略” \(p. 461\)](#)。您无法编辑 Amazon 托管策略。

使用可视化编辑器选项卡为您的策略摘要编辑策略

1. 按照上述步骤中的说明打开策略摘要。
2. 选择编辑策略。

如果您已打开用户页面，请选择编辑附加到该用户的客户托管策略，之后您会被重定向到策略页面。您只能在 Policies 页面上编辑客户托管策略。
3. 请选择 Visual editor (可视化编辑器) 选项卡以查看您的策略的可编辑可视化形式。IAM 可能会调整您的策略结构以针对可视化编辑器进行优化，并使您更轻松地查找和解决任何问题。页面上的警告和错误消息可以指导您解决策略中的任何问题。有关 IAM 如何调整策略结构的更多信息，请参阅 [调整策略结构 \(p. 871\)](#)。
4. 编辑您的策略，然后选择查看策略以查看是否在策略摘要中反映了更改。如果仍出现问题，请选择上一步以返回到编辑屏幕。
5. 选择 Save (保存) 以保存您的更改。

使用 JSON 选项卡为您的策略摘要编辑策略

1. 按照上述步骤中的说明打开策略摘要。
2. 选择 {} JSON 和 Policy summary 将策略摘要与 JSON 策略文档进行比较。您可以使用该信息来确定要更改策略文档中的哪些行。
3. 选择编辑策略，然后选择 JSON 选项卡以编辑 JSON 策略文档。

Note

您可以随时在可视化编辑器和 JSON 选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy (查看策略)，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅[调整策略结构](#) (p. 871)。

如果您已打开用户页面，请选择编辑附加到该用户的客户托管策略，之后您会被重定向到策略页面。您只能在 Policies 页面上编辑客户托管策略。

4. 编辑策略。解决[策略验证](#) (p. 443)过程中生成的任何安全警告、错误或常规警告，然后选择 Review policy (查看策略)。如果仍出现问题，请选择上一步以返回到编辑屏幕。
5. 选择 Save (保存) 以保存您的更改。

了解策略摘要的元素

在下面的用户详细信息页面示例中，PolSumUser 用户附加有八个策略。SummaryAllElements 策略是直接附加到用户的托管策略 (用户管理策略)。此策略已展开，显示了策略摘要。要查看该策略的 JSON 策略文档，请参阅[the section called “SummaryAllElements JSON 策略文档”](#) (p. 490)。

The screenshot displays the AWS IAM console interface for a user named PolSumUser. The 'Permissions' tab is selected, showing a list of attached policies. The 'SummaryAllElements' policy is expanded, revealing its details. A red box highlights the policy's summary and the table of permissions. The table lists various services and their associated permissions, including 'S3', 'Billing', 'EC2', and 'S3' again, with details on access levels, resources, and request conditions.

Service	Access level	Resource	Request condition
Unrecognized services			
codedploy			
Explicit deny (1 of 103 services)			
S3	Full: Read, Write, Permissions management Limited: List	Multiple	None
Allow (3 of 103 services) Show remaining 100			
Billing	Full: Read Limited: Write	All resources	Multiple
EC2	None	All resources	None
S3	Limited: Write, Permissions management	BucketName = developer_bucket, ObjectPath = All	s3:x-amz-acl = public-read

在上图中，策略摘要可在用户详细信息页面内可见：

1. 用户的 Permissions 选项卡包含附加到 PoISumUser 用户的策略。
2. SummaryAllElements 策略是附加到用户的几个策略之一。此策略已展开，显示了策略摘要。
3. 如果策略未向策略中定义的所有操作、资源和条件授予权限，则将在页面顶部显示警告或错误横幅。策略摘要中包含有关问题的详细信息。要了解策略摘要如何帮助您了解策略授予的权限并进行相关问题排查，请参阅[the section called “我的策略未授予预期权限” \(p. 875\)](#)。
4. 使用 Policy summary 和 { } JSON 按钮可在策略摘要和 JSON 策略文档之间切换。
5. Simulate policy 可打开用于测试策略的策略模拟器。
6. 使用搜索框可减少服务列表，轻松查找特定服务。
7. 展开的视图显示了 SummaryAllElements 策略的更多详细信息。

下面的策略摘要表图像显示在 PoISumUser 用户详细信息页面上的已展开 SummaryAllElements 策略。

A Service	G Access level	H Resource	I Request condition
B Unrecognized services			
codedploy			
C Explicit deny (1 of 103 services)			
S3	Full: Read, Write, Permissions management Limited: List	Multiple	None
Allow (3 of 103 services) Show remaining 100			
Billing	Full: Read Limited: Write	All resources	Multiple
E EC2	None	All resources	None
F S3	Limited: Write, Permissions management	BucketName = developer_bucket, ObjectPath = All	s3:x-amz-acl = public-read

在上图中，策略摘要可在用户详细信息页面内可见：

- A. Service (服务) - 此列将列出在策略内定义的服务并提供每项服务的详细信息。策略摘要表中的每个服务名称都是指向服务摘要表的一个链接，[服务摘要 \(操作列表 \) \(p. 493\)](#)中对其进行了说明。在此示例中，为 Amazon S3、账单和 Amazon EC2 服务定义了权限。该策略还为 IAM 无法识别的 (拼写错误的) codedploy 服务定义了权限。
- B. Unrecognized services (无法识别的服务) - 此策略包括一个无法识别的服务 (在此示例中为 codedploy)。您可以使用此警告来检查服务名称是否可能包含拼写错误。如果服务名称正确，则该服务可能不支持策略摘要，可能处于预览状态，或者可能是自定义服务。要为公开提供 (GA) 服务请求策略摘要支持，请参阅[服务不支持 IAM policy 摘要 \(p. 874\)](#)。在此示例中，策略包括一个无法识别的 codedploy 服务，其名称中缺少一个 e。由于这种拼写错误，该策略不提供预期的 Amazon CodeDeploy 权限。您可以[编辑策略 \(p. 485\)](#)以包括准确的 codedeploy 服务名称；随后该服务将出现在策略摘要中。
- C. 对于 IAM 识别的那些服务，它根据策略是允许还是显式拒绝使用该服务来安排服务。在此示例中，策略包含用于 Amazon S3 服务的 Allow 和 Deny 语句。因此，策略摘要在 Explicit deny 和 Allow 部分中都包含 S3。
- D. Show remaining 100 (显示剩余 100 个) - 选择此链接可展开表以包含策略未定义的服务。这些服务在该策略中被隐式拒绝 (或默认拒绝)。但是，另一策略中的语句可能仍然允许或显式拒绝使用该服务。策略摘要汇总了单个策略的权限。要了解 Amazon 服务如何决定是允许还是拒绝给定的请求，请参阅[策略评估逻辑 \(p. 972\)](#)。
- E. EC2 — 此服务包括未识别的操作。IAM 将识别支持策略摘要的服务的服务名称、操作和资源类型。如果已识别某个服务，但无法识别该服务包含的操作，IAM 将在该服务旁边显示一条警告。在此示例中，IAM 至少无法识别一个 Amazon EC2 操作。要了解无法识别的操作的更多信息，并查看 S3 服务摘要中无法识别的操作，请参阅[服务摘要 \(操作列表 \) \(p. 493\)](#)。

Note

IAM 将检查支持策略摘要的服务的服务名称、操作和资源类型。但是，您的策略摘要可能包含不存在的资源值或条件。始终通过[策略模拟器 \(p. 443\)](#)测试您的策略。

F.  S3 — 此服务包括未识别的资源。IAM 将识别支持策略摘要的服务的服务名称、操作和资源类型。如果已识别某个服务，但无法识别该服务包含的资源类型，IAM 将在该服务旁边显示一条警告。在此示例中，IAM 至少无法识别一个 Amazon S3 操作。要了解无法识别的资源的更多信息，并查看 S3 服务摘要中无法识别的资源类型，请参阅[服务摘要 \(操作列表\) \(p. 493\)](#)。

G. Access level (访问级别) - 此列指出每个访问级别 (List、Read、Write 和 Permissions management) 中的操作是具有策略中定义的 Full 还是 Limited 权限。有关访问权限级别摘要的更多详细信息和示例，请参阅[了解策略摘要内的访问级别摘要 \(p. 491\)](#)。

- Full access (完全访问权限) - 此条目指示服务对该服务可用的全部四个访问权限级别中的所有操作都拥有访问权限。在此示例中，因为此行在表的 Explicit deny (显式拒绝) 部分中，所以将对策略中包含的资源拒绝所有 Amazon S3 操作。
- 如果该项不包含 Full access，则服务可以访问部分但不是全部用于该服务的操作。然后，通过对四个访问级别分类 (List、Read、Write 和 Permissions management) 中的每个分类的说明来定义访问权限：

Full：策略提供对列出的每个访问级别分类中的所有操作的访问权限。在此示例中，策略提供对所有账单 Read 操作的访问权限。

Limited：策略提供对所列每个访问级别分类内的一个或多个但不是全部操作的访问权限。在此示例中，策略提供对部分账单 Write 操作的访问权限。

H. Resource (资源) - 此列显示策略为每项服务指定的资源。

- Multiple (多个) - 策略包含服务内多个但不是全部资源。在此示例中，显式拒绝对多个 Amazon S3 资源的访问权限。
- All resources (所有资源) - 策略是为服务内的所有资源定义的。在此示例中，策略允许对所有账单资源执行列出的操作。
- Resource text - 该策略包含服务内的一个资源。在此示例中，只允许对 developer_bucket Amazon S3 存储桶资源执行列出的操作。根据服务提供给 IAM 的信息，您可能会看到一个 ARN (如 arn:aws:s3:::developer_bucket/*)，或者您可能会看到定义的资源类型 (如 BucketName = developer_bucket)。

Note

此列可以包括来自不同服务的资源。如果包含资源的策略语句不包括来自同一服务的操作和资源，则您的策略将包括不匹配的资源。在创建策略或在策略摘要中查看策略时，IAM 不会就不匹配的资源向您发出警告。如果此列包含不匹配的资源，那么您应该查看策略中是否有错误。为了更好地了解您的策略，请始终使用[策略模拟器 \(p. 443\)](#)进行测试。

I. Request condition (请求操作) - 此列指示与资源关联的服务或操作是否受条件约束。

- None (无) - 策略对服务不包含任何条件。在此示例中，没有条件适用于 Amazon S3 服务中拒绝的操作。
- Condition text - 策略对服务包含一个条件。在此示例中，仅当源的 IP 地址与 203.0.113.0/24 匹配时，才允许列出的账单操作。
- Multiple (多个) - 策略对服务包含多个条件。在此示例中，基于多个条件允许对列出的 Amazon S3 操作的访问权限。要查看策略多个条件中的每一个条件，请选择 {} JSON 查看策略文档。

当策略或策略中的元素未授予权限时，IAM 在策略摘要中提供额外的警告和信息。下面的策略摘要表显示了在 PoIsumUser 用户详细信息页面上展开的 Show remaining 100 (显示剩余 100 条) 服务，并附带可能的警告。

Service	Access level	Resource	Request condition
Unrecognized services			
codedeploy			
Explicit deny (1 of 103 services)			
S3	Full: Read, Write, Permissions management Limited: List	Multiple One or more actions do not have an applicable resource.	None
Allow (3 of 103 services) Hide remaining 100			
...	None		
Billing	Full: Read Limited: Write	All resources	Multiple
CodeBuild	None - No actions are defined.	arn:aws:codebuild:us-east-1:123456789012:project/my-demo-project	None
CodeCommit	None	No resources are defined.	None
CodeDeploy	None - No actions are defined.	arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup:*	None
EC2	None	All resources	None
S3	Limited: Write, Permissions management	BucketName = developer_bucket, ObjectPath = All One or more resources do not have an applicable action.	One or more conditions do not have an applicable action.

在上图中，您可以看到包含没有权限的既定操作、资源或条件的所有服务：

a. Resource warnings (资源警告) - 对于没有为所有包含的操作或资源提供权限的服务，您将在表的 Resource (资源) 列中看到以下警告之一：

- ⚠ No resources are defined. (未定义任何资源。) - 这意味着服务具有定义的操作，但策略不包含支持的资源。
- ⚠ One or more actions do not have an applicable resource. (一个或多个操作没有适用的资源。) - 这意味着该服务具有定义的操作，但其中的一些操作没有支持的资源。
- ⚠ One or more resources do not have an applicable action. (一个或多个资源没有适用的操作。) - 这意味着该服务具有定义的资源，但其中的一些资源没有支持的操作。

如果服务同时包含没有适用资源的操作和没有适用操作的资源，则显示 One or more resources do not have an applicable action. (一个或多个资源没有适用的操作。) 警告。这是因为：当您查看服务的策略摘要时，不会显示不适用于任何操作的资源。对于 ListAllMyBuckets 操作，该策略包含最后一条警告，因为该操作不支持资源级权限，也不支持 s3:x-amz-acl 条件键。如果修复了资源问题或条件问题，详细警告中将显示剩余问题。

b. Request condition warnings (请求条件警告) - 对于没有为所有包含的条件提供权限的服务，您将在表的 Request condition (请求条件) 列中看到以下警告之一：

- ⚠ One or more actions do not have an applicable condition. (一个或多个操作没有适用的条件。) - 这意味着服务具有定义的操作，但其中的一些操作没有支持的条件。
- ⚠ One or more conditions do not have an applicable action. (一个或多个条件没有适用的操作。) - 这意味着服务定义了一些条件，但其中的一些条件没有支持的操作。

c. Multiple | ⚠ One or more actions do not have an applicable resource. (一个或多个操作没有适用的资源。) - Amazon S3 的 Deny 语句包含多个资源。它还包含多个操作，但其中一些操作支持资源，一些不支持。要查看该策略，请参阅 [the section called "SummaryAllElements JSON 策略文档" \(p. 490\)](#)。在这种情况下，策略包含所有 Amazon S3 操作，只拒绝可在存储桶或存储桶对象上执行的操作。

- d. 省略号 (...) 表示在页面中包含所有服务，但我们只显示具有此策略相关信息的行。在 Amazon Web Services Management Console 中查看该页面时，您将看到所有 Amazon 服务。
- e. 表中行的背景颜色表示不授予任何权限的服务。您无法在策略摘要中获取有关这些服务的任何其他信息。对于以白色行显示的服务，您可以选择服务名称查看服务摘要 (操作列表) 页面。在该页面中，您可以了解有关为该服务授予的权限的更多信息。
- f.  None - No actions are defined. (无 - 未定义任何操作。) - 这意味着服务定义为资源或条件，但服务不包含任何操作，因此服务不提供任何权限。在这种情况下，策略包含 CodeBuild 资源但不包含 CodeBuild 操作。
- g. No resources are defined  服务具有定义的操作，但策略中不包含支持的资源，因此服务不提供任何权限。在这种情况下，策略包含 CodeCommit 操作但不包含 CodeCommit 资源。
- h. BucketName = developer_bucket, ObjectPath = All |  One or more resources do not have an applicable action. (一个或多个资源没有适用的操作。) - 服务具有定义的存储桶对象资源和至少一个没有支持操作的资源。
- i. s3:x-amz-acl = public-read |  One or more conditions do not have an applicable action. (一个或多个条件没有适用的操作。) - 服务具有定义的 s3:x-amz-acl 条件键和至少一个没有支持操作的条件键。

SummaryAllElements JSON 策略文档

SummaryAllElements 策略不适用于在账户中定义权限。包含它的目的在于演示您在查看策略摘要时可能会遇到的错误和警告。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "aws-portal:ViewBilling",
        "aws-portal:ViewPaymentMethods",
        "aws-portal:ModifyPaymentMethods",
        "aws-portal:ViewAccount",
        "aws-portal:ModifyAccount",
        "aws-portal:ViewUsage"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "203.0.113.0/24"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::customer",
        "arn:aws:s3:::customer/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "ec2:GetConsoleScreenshots"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "codedploy:*",
        "codecommit:*"
      ],
      "Resource": [
        "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup:*",
        "arn:aws:codebuild:us-east-1:123456789012:project/my-demo-project"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": [
        "arn:aws:s3:::developer_bucket",
        "arn:aws:s3:::developer_bucket/*",
        "arn:aws:autoscaling:us-east-2:123456789012:autoscalgrp"
      ],
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": [
            "public-read"
          ],
          "s3:prefix": [
            "custom",
            "other"
          ]
        }
      }
    }
  ]
}
```

了解策略摘要内的访问级别摘要

Amazon 访问级别摘要

策略摘要中包括一个访问级别摘要，用于描述为策略中提及的每项服务定义的操作权限。要了解策略摘要，请参阅[了解策略授予的权限 \(p. 483\)](#)。访问级别摘要指出在策略中为每个访问级别中的操作（List、Read、Tagging、Write 和 Permissions management）定义的是 Full 还是 Limited 权限。要查看分配给服务中每个操作的访问级别分类，请参阅[Amazon 服务的操作、资源和条件键](#)。

下面的示例介绍策略为给定服务提供的访问权限。有关完整 JSON 策略文档及其相关摘要的示例，请参阅[策略摘要示例 \(p. 499\)](#)。

服务	访问级别	此策略提供以下访问权限
IAM	完全访问权限	对 IAM 服务内所有操作的访问权限。

服务	访问级别	此策略提供以下访问权限
CloudWatch	Full: List	对 List 访问级别中所有 CloudWatch 操作的访问权限，但无权访问 Read、Write 或 Permissions management 访问级别分类的操作。
Data Pipeline	Limited: List, Read	对于 List 和 Read 访问级别中至少一项但不是全部 Amazon Data Pipeline 操作 (但不是 Write 或 Permissions management 操作) 的访问权限。
EC2	Full: List, Read Limited: Write	对所有 Amazon EC2 List 和 Read 操作的访问权限以及对至少一项但不是全部 Amazon EC2 Write 操作的访问权限，但不具有对于 Permissions management 访问级别分类中的操作的访问权限。
S3	Limited : Read、Write、Permissions management	访问至少一个但并非全部 Amazon S3 Read、Write 和 Permissions management 操作。
CodeDeploy	(空)	未知访问权限，因为 IAM 无法识别此服务。
API Gateway	None (无)	策略中未定义任何访问权限。
CodeBuild	 未定义任何操作。	没有为服务定义任何操作，因而无法访问。要了解该问题和进行问题排查，请参阅 the section called “我的策略未授予预期权限” (p. 875) 。

如前所述 (p. 488)，完全访问权限表示策略提供对服务内所有操作的访问权限。提供对服务内的部分但不是全部操作的访问权限的策略将根据访问级别分类进一步分组。这由下面的其中一个访问级别分组来指示：

- Full：策略提供对指定访问级别分类中所有操作的访问权限。
- Limited：策略提供对指定访问级别分类内的一个或多个但不是全部操作的访问权限。
- None：策略未提供任何访问权限。
- (空)：IAM 无法识别该服务。如果服务名称包含拼写错误，则该策略不允许访问该服务。如果服务名称正确，则服务可能不支持策略摘要或可能正处于预览状态。在这种情况下，策略可能会提供访问权限，但访问权限可能不会显示在策略摘要中。要为公开提供 (GA) 服务请求策略摘要支持，请参阅 [服务不支持 IAM policy 摘要 \(p. 874\)](#)。

包括对操作的有限 (部分) 访问权限的访问级别摘要使用 Amazon 服务级别分类 List、Read、Tagging、Write 或 Permissions management 进行分组。

Amazon 访问级别

Amazon 为服务中的操作定义以下访问级别分类：

- List (列出)：列出服务内的资源以确定某个对象是否存在的权限。此访问权限级别的操作可以列出对象，但是看不到资源的内容。例如，Amazon S3 操作 ListBucket 具有 List (列出) 访问级别。
- Read (读取)：读取服务中资源的内容和属性但不对其进行编辑的权限。例如，Amazon S3 操作 GetObject 和 GetBucketLocation 具有 Read (读取) 访问权限级别。
- 标记：执行仅更改资源标签状态的操作的权限。例如，IAM 操作 TagRole 和 UntagRole 具有标记访问级别，因为它们仅允许标记或取消标记角色。不过，CreateRole 操作允许在创建角色时标记该角色资源。由于该操作并非仅添加标签，因此，它具有 Write 访问级别。
- Write (写入)：在服务中创建、删除或修改资源的权限。例如，Amazon S3 操作 CreateBucket、DeleteBucket 和 PutObject 具有写入访问级别。Write 操作可能还允许修改资源标签。不过，仅允许更改标签的操作具有 Tagging 访问级别。

- Permissions management (权限管理)：在服务中授予或修改资源权限的权限。例如，大多数 IAM 和 Amazon Organizations 操作以及 PutBucketPolicy 和 DeleteBucketPolicy 之类的 Amazon S3 操作具有 Permissions (权限管理) 访问级别。

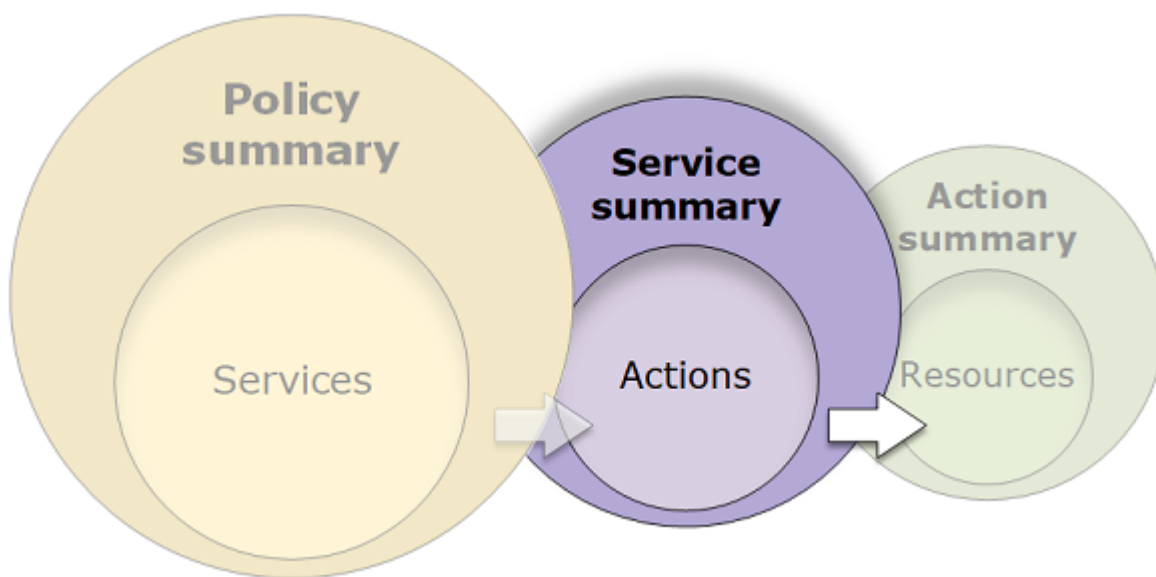
提示

要提高您的 Amazon 账户的安全性，请限制或定期监控具有 Permissions management (权限管理) 访问级别分类的策略。

要查看分配给服务中每个操作的访问级别分类，请参阅 [Amazon 服务的操作、资源和条件键](#)。

服务摘要 (操作列表)

策略在三个表中概括：策略摘要、[服务摘要 \(p. 484\)](#)和[操作摘要 \(p. 496\)](#)。服务摘要表包括由所选服务的策略定义的操作列表和权限摘要。



对于策略摘要中列出的每个授予权限的服务，您都可以查看服务摘要。该表分组为 Uncategorized actions、Uncategorized resource types 和访问级别部分。如果策略包含 IAM 无法识别的操作，则该操作将包含在表的 Uncategorized actions (无法识别的操作) 部分中。如果 IAM 能够识别该操作，则它将包含在表的某个访问级别 (List (列出)、Read (读取)、Write (写入) 和 Permissions management (权限管理)) 部分下。要查看分配给服务中每个操作的访问级别分类，请参阅 [Amazon 服务的操作、资源和条件键](#)。

查看服务摘要

您可以在 Policies 页面中查看托管策略的服务摘要，或通过 Users 和 Roles 页面查看附加到用户或角色的内联和托管策略的服务摘要。但是，如果您从托管策略在 Users 页面和 Roles 页面上选择服务名称，那么您将被重定向到 Policies 页面。必须在 Policies 页面上查看托管策略的服务摘要。

查看托管策略的服务摘要

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Policies (策略)。
3. 在策略列表中，选择要查看的策略的名称。

4. 在策略的 Summary 页面上, 查看 Permissions 选项卡以查看策略摘要。
5. 在服务的策略摘要列表中, 选择您想要查看的服务的名称。

查看附加到用户的策略的服务摘要

1. 登录 Amazon Web Services Management Console, 单击 <https://console.aws.amazon.com/iam/> 打开 IAM 控制台。
2. 在导航窗格中, 选择 Users (用户)。
3. 在用户列表中, 选择要查看其策略的用户的名称。
4. 在用户的 Summary 页面上, 查看 Permissions 选项卡, 以查看直接附加到用户或从组附加到用户的策略列表。
5. 在用户的策略表中, 展开要查看的策略的行。
6. 在服务的策略摘要列表中, 选择您想要查看的服务的名称。

Note

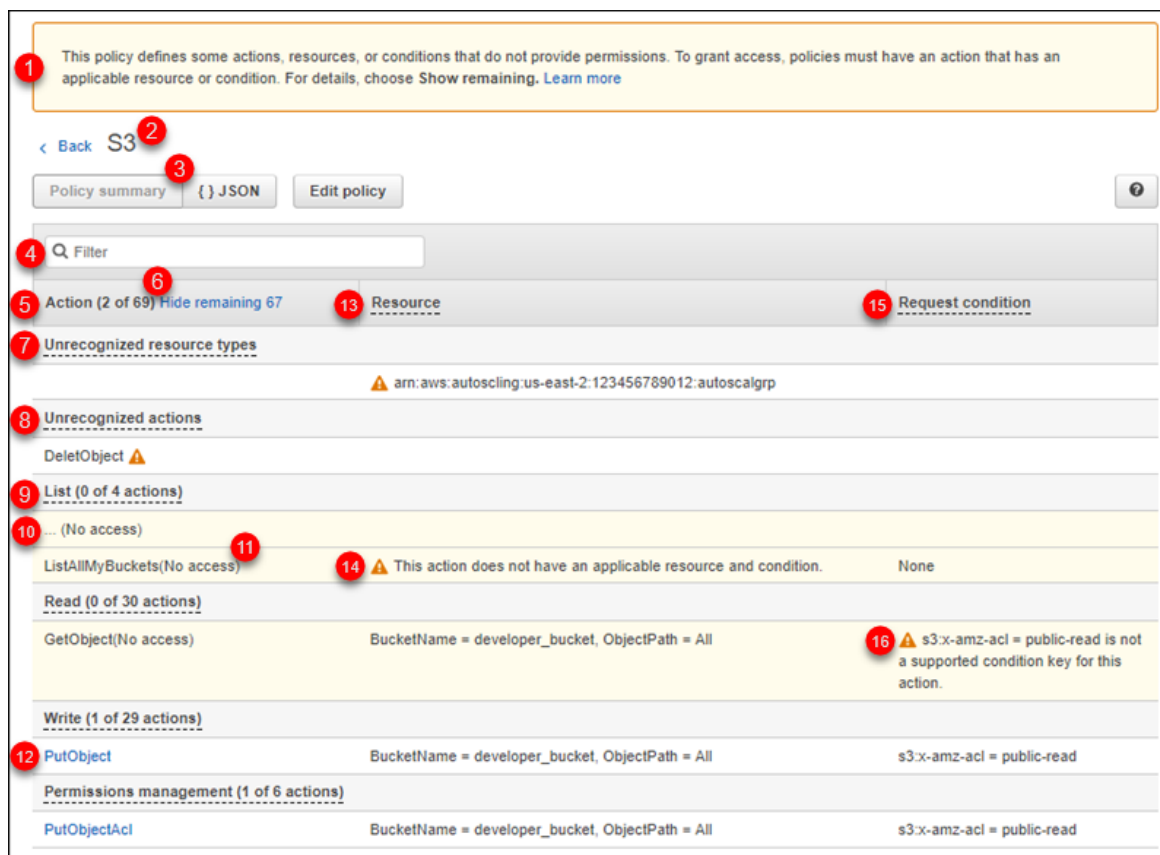
如果您选择的策略是直接附加到用户的内联策略, 则将显示服务摘要表。如果策略是附加到组的内联策略, 则您将看到该组的 JSON 策略文档。如果策略是托管策略, 则您将在 Policies 页面中看到该策略的服务摘要。

查看附加到角色的策略的服务摘要

1. 登录 Amazon Web Services Management Console, 然后通过以下网址打开 IAM 控制台: <https://console.aws.amazon.com/iam/>。
2. 从导航窗格中, 选择 Roles。
3. 在角色列表中, 选择要查看其策略的角色的名称。
4. 在角色的 Summary 页面上, 查看 Permissions 选项卡, 以查看附加到角色的策略列表。
5. 在角色的策略表中, 展开要查看的策略的行。
6. 在服务的策略摘要列表中, 选择您想要查看的服务的名称。

了解服务摘要的元素

下面的示例是 Amazon S3 操作的服务摘要, 这些操作由 SummaryAllElements 策略摘要许可 (请参阅 [the section called “SummaryAllElements JSON 策略文档” \(p. 490\)](#))。此服务的操作按 Uncategorized actions、Uncategorized resource types 和访问级别分组。例如, 两个 Write (写入) 操作定义为可用于服务的总计 29 个 Write (写入) 操作之中。



托管策略的服务摘要页面上包含以下信息：

1. 如果策略未向策略中服务定义的所有操作、资源和条件授予权限，则将在页面顶部显示警告或错误横幅。服务摘要中包含有关该问题的详细信息。要了解策略摘要如何帮助您了解策略授予的权限并进行相关问题排查，请参阅[the section called “我的策略未授予预期权限” \(p. 875\)](#)。
2. 在 Back 链接旁将显示服务的名称 (在此例中为 S3)。此服务的服务摘要包括在策略中定义的允许的操作列表。如果在服务名称旁显示的是 (Explicitly denied)，则服务摘要表中列出的操作将被显式拒绝。
3. 选择 {} JSON 可查看策略的其他详细信息。您可以执行此操作以查看应用于操作的所有条件。(如果您要查看直接附加到用户的内联策略的服务摘要，则必须关闭服务摘要对话框并返回到策略摘要以访问 JSON 策略文档。)
4. 要查看特定操作的摘要，请在搜索框中键入关键字以缩短可用操作列表。
5. Action (2 of 69 actions) (操作 (2 个，共 69 个操作)) - 此列将列出在策略中定义的操作，并提供每个操作的资源和条件。如果策略向操作授予权限，则操作名称链接到[操作摘要 \(p. 496\)](#) 表。计数表示提供权限的已识别操作的数量。总计是服务的已知操作数量。在此示例中，总共 69 个已知 S3 操作中有 2 个操作提供权限。
6. Show/Hide remaining 67 (显示/隐藏剩余 67 个) - 选择此链接可展开或隐藏该表，以包含已知但未向此服务提供权限的操作。展开链接还将显示未提供权限的任何元素的警告。
7. Unrecognized resource types (无法识别的资源类型) - 此策略在服务策略中包含至少一个无法识别的资源类型。您可以使用该警告来检查资源类型是否可能包含拼写错误。如果资源类型正确，则该服务可能不完全支持策略摘要，可能处于预览状态，或者可能是自定义服务。要在公开提供 (GA) 服务中请求对特定资源类型的策略摘要支持，请参阅[服务不支持 IAM policy 摘要 \(p. 874\)](#)。在本示例中，autoscaling 服务名称缺少一个 a。
8. Unrecognized actions (无法识别的操作) - 此策略在服务策略中包含至少一个无法识别的操作。您可以使用此警告来检查操作是否可能包含拼写错误。如果操作名称正确，则该服务可能不完全支持策略摘要，可

能处于预览状态，或者可能是自定义服务。要在公开提供 (GA) 服务中请求对特定操作的策略摘要支持，请参阅[服务不支持 IAM policy 摘要 \(p. 874\)](#)。在此示例中，DeleteObject  操作缺少 e。

Note

IAM 将检查支持策略摘要的服务的服务名称、操作和资源类型。但是，您的策略摘要可能包含不存在的资源值或条件。始终通过[策略模拟器 \(p. 443\)](#)测试您的策略。

9. 对于 IAM 能够识别的那些操作，该表将操作分组为至少 1 个、最多 4 个部分，具体取决于策略允许或拒绝的访问权限级别。这些部分是 List、Read、Write 和 Permissions management。您还可以查看每个访问权限级别内可用的总操作数以及定义的操作数。要查看分配给服务中每个操作的访问级别分类，请参阅[Amazon 服务的操作、资源和条件键](#)。
10. 省略号 (...) 表示在页面中包含所有操作，但我们只显示包含该策略相关信息的行。在 Amazon Web Services Management Console 中查看此页面时，您将看到服务的所有操作。
11. (No access) (无权访问) - 此策略包含未提供权限的操作。
12. 提供权限的操作包含指向操作摘要的链接。
13. Resource (资源) - 此列显示策略为服务定义的资源。IAM 不检查资源是否适用于每个操作。在此示例中，只允许对 developer_bucket Amazon S3 存储桶资源执行 S3 服务中的操作。根据服务提供给 IAM 的信息，您可能会看到一个 ARN (如 arn:aws:s3:::developer_bucket/*)，或者您可能会看到定义的资源类型 (如 BucketName = developer_bucket)。

Note

此列可以包括来自不同服务的资源。如果包含资源的策略语句不包括来自同一服务的操作和资源，则您的策略将包括不匹配的资源。在创建策略或在服务摘要中查看策略时，IAM 不会就不匹配的资源向您发出警告。IAM 也不会指示操作是否适用于资源，仅会指示服务是否匹配。如果此列包含不匹配的资源，那么您应该查看策略中是否有错误。为了更好地了解您的策略，请始终使用[策略模拟器 \(p. 443\)](#)进行测试。

14. Resource warning (资源警告) 对于资源不提供完全权限的操作，您将看到以下警告之一：
 - 此操作不支持资源级权限。这需要为资源提供通配符 (*)。 - 这意味着策略包含资源级权限，但必须包含 "Resource": ["*"] 来提供此操作的权限。
 - 此操作没有适用的资源。 - 这意味着策略中包含该操作，但没有支持资源。
 - This action does not have an applicable resource and condition. (此操作没有适用的资源和条件。) - 这意味着策略中包含该操作，但没有支持资源和支持条件。在这种情况下，策略中还包含用于此服务的条件，但没有适用于此操作的条件。

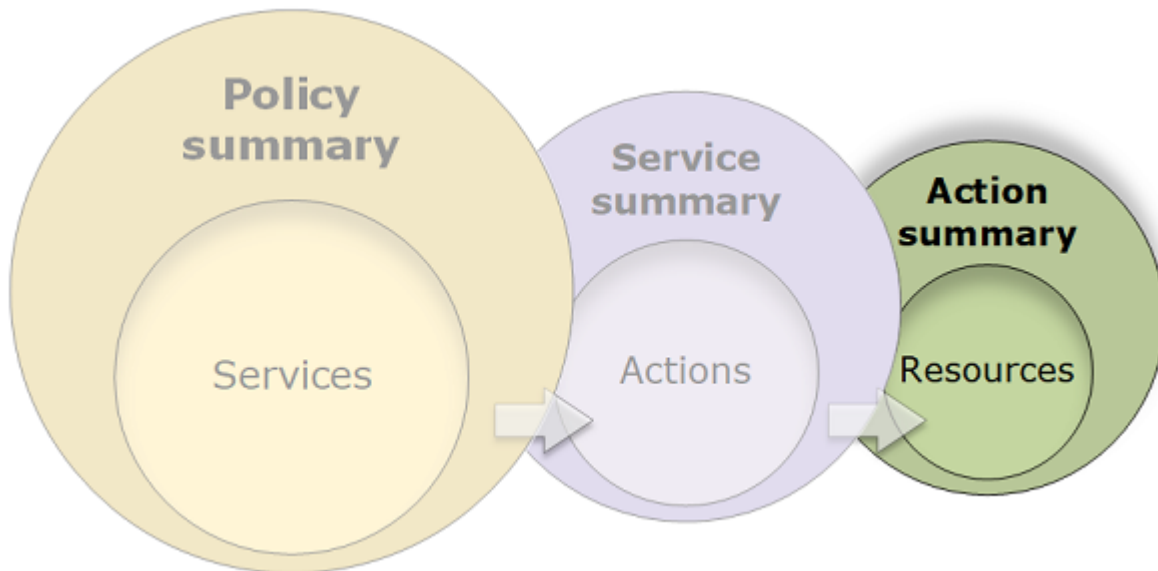
对于 ListAllMyBuckets 操作，该策略包含最后一条警告，因为该操作不支持资源级权限，也不支持 s3:x-amz-acl 条件键。如果修复了资源问题或条件问题，详细警告中将显示剩余问题。

15. Request condition (请求条件) - 此列指出与资源关联的操作是否受条件约束。要了解有关这些条件的更多信息，请选择 {} JSON 以查看 JSON 策略文档。
16. Condition warning (条件警告) - 对于条件不提供完全权限的操作，您将看到以下警告之一：
 - <CONDITION_KEY> is not a supported condition key for this action. (<CONDITION_KEY> 不是此操作的支持条件键。) - 这意味着策略包含此操作不支持的服务条件键。
 - Multiple condition keys are not supported for this action. (此操作不支持多个条件键。) - 这意味着策略包含此操作不支持的多个服务条件键。

对于 GetObject，该策略包含 s3:x-amz-acl 条件键，这不适用于此操作。虽然该操作支持资源，但由于条件对于该操作永远为 false，因而策略不为该操作授予任何权限。

操作摘要 (资源列表)

策略在三个表中概括：策略摘要、[服务摘要 \(p. 484\)](#)和[操作摘要 \(p. 493\)](#)。操作摘要 表包含应用至所选操作的资源和相关条件的列表。



要查看授予权限的每个操作的操作摘要，请在服务摘要中选择链接。操作摘要表包含资源的详细信息，其中包括资源的 Region 和 Account。您还可以查看适用于每个资源的条件。这将显示适用于某些资源而不是其他资源的条件。

查看操作摘要

您可以在 Users 页面上查看附加到用户的任何策略的操作摘要，在 Roles 页面上查看附加到角色的任何策略的操作摘要，以及在 Policies 页面上查看托管策略的操作摘要。但是，如果您尝试从 Users 页面或 Roles 页面查看托管策略的操作摘要，则将被重定向到 Policies 页面。

查看托管策略的操作摘要

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Policies (策略)。
3. 在策略列表中，选择要查看的策略的名称。
4. 在策略的 Summary 页面上，查看 Permissions 选项卡以查看策略摘要。
5. 在服务的策略摘要列表中，选择您想要查看的服务的名称。
6. 在操作的服务摘要列表中，选择您想要查看的操作的名称。

查看附加到用户的策略的操作摘要

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 从导航窗格中选择 Users。
3. 在用户列表中，选择要查看其策略的用户的名称。
4. 在用户的 Summary 页面上，查看 Permissions 选项卡，以查看直接附加到用户或从组附加到用户的策略列表。
5. 在用户的策略表中，展开要查看的策略的行。
6. 在服务的策略摘要列表中，选择您想要查看的服务的名称。

Note

如果您选择的策略是直接附加到用户的内联策略，则将显示服务摘要表。如果策略是附加到组的内联策略，则您将看到该组的 JSON 策略文档。如果策略是托管策略，则您将在 Policies 页面中看到该策略的服务摘要。

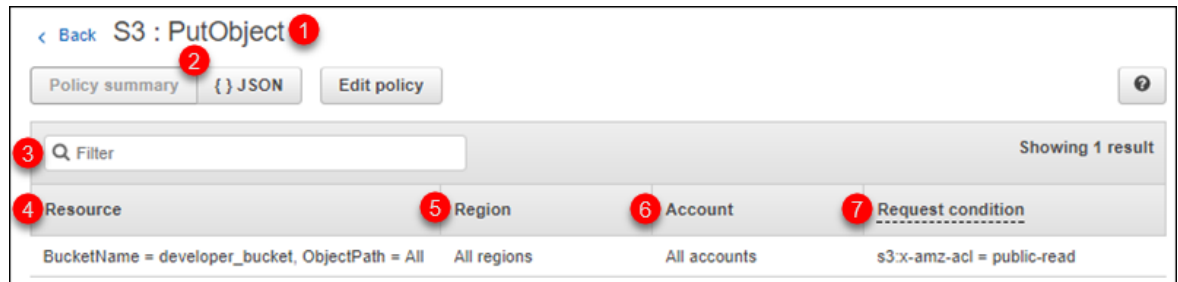
7. 在操作的服务摘要列表中，选择您想要查看的操作的名称。

查看附加到角色的策略的操作摘要

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择角色。
3. 在角色列表中，选择要查看其策略的角色的名称。
4. 在角色的 Summary 页面上，查看 Permissions 选项卡，以查看附加到角色的策略列表。
5. 在角色的策略表中，展开要查看的策略的行。
6. 在服务的策略摘要列表中，选择您想要查看的服务的名称。
7. 在操作的服务摘要列表中，选择您想要查看的操作的名称。

了解操作摘要的元素

以下示例是 Amazon S3 服务摘要中 PutObject (写入) 操作的操作摘要 (请参阅 [服务摘要 \(操作列表 \)](#) (p. 493))。对于此操作，策略在单个资源上定义多个条件。



操作摘要页面包含以下信息：

1. Back 链接旁边显示的是 service: action 格式的服务和操作名称 (在本例中为 S3: PutObject)。此服务的操作摘要包括在策略中定义的资源列表。
2. 选择 { } JSON 可查看有关策略的其他详细信息，如查看适用于这些操作的多个条件。(如果您要查看直接附加到用户的内联策略的操作摘要，步骤则不同。这种情况下要访问 JSON 策略文档，则必须关闭服务摘要对话框并返回到策略摘要。)
3. 要查看特定资源的摘要，请在搜索框中键入关键字以缩短可用资源列表。
4. Resource (资源) - 此列将列出策略为所选服务定义的资源。在此示例中，PutObject 操作被允许在所有对象路径上执行，但仅在 developer_bucket Amazon S3 存储桶资源上使用。根据服务提供给 IAM 的信息，您可能会看到一个 ARN (如 arn:aws:s3:::developer_bucket/*)，或者您可能会看到定义的资源类型 (如 BucketName = developer_bucket, ObjectPath = All)。
5. Region (区域) - 该列显示在其中定义资源的区域。可以为所有区域或单个区域定义资源。它们不能位于多个特定的区域中。
 - All Regions (所有区域) - 与资源关联的操作适用于所有区域。在此示例中，该操作属于一项全球服务 Amazon S3。属于全球服务的操作适用于所有区域。
 - Region text - 与资源关联的操作适用于一个区域。例如，策略可以为资源指定 us-east-2 区域。

6. Account (账户) - 此列指示与资源相关联的服务或操作是否适用于特定账户。资源可以存在于所有账户中，也可以存在于单个账户中。它们不能存在于多个特定账户中。
 - All accounts (所有账户) - 与资源相关联的操作适用于所有账户。在此示例中，该操作属于一项全球服务 Amazon S3。属于全球服务的操作适用于所有账户。
 - This account (此账户) - 与资源相关联的操作仅适用于您当前登录的账户。
 - Account number - 与该资源相关联的操作适用于一个账户 (不是您当前登录的账户)。例如，如果策略为资源指定 123456789012 的账户，则账号将显示在策略摘要中。
7. Request condition (请求条件) - 此列显示与资源关联的操作是否受条件约束。此示例包含 `s3:x-amz-acl = public-read` 条件。要了解有关这些条件的更多信息，请选择 {} JSON 以查看 JSON 策略文档。

策略摘要示例

以下示例包括 JSON 策略及其关联的**策略摘要** (p. 484)、**服务摘要** (p. 493)和**操作摘要** (p. 496)，可帮助您了解通过策略授予的权限。

策略 1 : DenyCustomerBucket

此策略展示对同一项服务的允许和拒绝。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "FullAccess",
      "Effect": "Allow",
      "Action": ["s3:*"],
      "Resource": ["*"]
    },
    {
      "Sid": "DenyCustomerBucket",
      "Action": ["s3:*"],
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::customer", "arn:aws:s3:::customer/*" ]
    }
  ]
}
```

DenyCustomerBucket 策略摘要：

This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an applicable resource or condition. For details, choose Show remaining . Learn more			
Policy summary	{ } JSON	Edit policy	Simulate policy
Q Filter			
Service	Access level	Resource	Request condition
Explicit deny (1 of 103 services)			
S3	Full: Read, Write, Permissions management Limited: List	Multiple	None
Allow (1 of 103 services) Show remaining 102			
S3	Full access	All resources	None

DenyCustomerBucket S3 (Explicit deny) 服务摘要：

Action (66 of 69) Hide remaining 3	Resource	Request condition
List (1 of 4 actions)		
HeadBucket (No access)	⚠ This action does not support resource-level permissions. This requires a wildcard (*) for the resource.	None
ListAllMyBuckets(No access)	⚠ This action does not support resource-level permissions. This requires a wildcard (*) for the resource.	None
ListBucket	BucketName = customer	None
ListObjects (No access)	⚠ This action does not support resource-level permissions. This requires a wildcard (*) for the resource.	None
Read (30 of 30 actions)		
GetAccelerateConfiguration	BucketName = customer	None
GetAnalyticsConfiguration	BucketName = customer	None
GetBucketAcl	BucketName = customer	None
GetBucketCORS	BucketName = customer	None
GetBucketLocation	BucketName = customer	None
GetBucketLogging	BucketName = customer	None
GetBucketNotification	BucketName = customer	None
GetBucketPolicy	BucketName = customer	None
GetBucketRequestPayment	BucketName = customer	None
GetBucketTagging	BucketName = customer	None
GetBucketVersioning	BucketName = customer	None
GetBucketWebsite	BucketName = customer	None
GetInventoryConfiguration	BucketName = customer	None
GetIpConfiguration	BucketName = customer	None
GetLifecycleConfiguration	BucketName = customer	None
GetMetricsConfiguration	BucketName = customer	None
GetObject	BucketName = customer, ObjectPath = All	None
GetObjectAcl	BucketName = customer, ObjectPath = All	None
GetObjectTagging	BucketName = customer, ObjectPath = All	None
GetObjectTorrent	BucketName = customer, ObjectPath = All	None
GetObjectVersion	BucketName = customer, ObjectPath = All	None

GetObject (Read) 操作摘要：

Resource	Region	Account	Request condition
BucketName = customer, ObjectPath = All	All regions	All accounts	None

策略2：DynamoDbRowCognitoID

该策略基于用户的 Amazon Cognito ID 提供对 Amazon DynamoDB 的行级别访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:UpdateItem"
      ],
      "Resource": [
        "arn:aws:dynamodb:us-west-1:123456789012:table/myDynamoTable"
      ],
      "Condition": {
        "ForAllValues:StringEquals": {
          "dynamodb:LeadingKeys": [
            "${cognito-identity.amazonaws.com:sub}"
          ]
        }
      }
    }
  ]
}
```

DynamoDbRowCognitoID 策略摘要：

Service	Access level	Resource	Request condition
Allow (1 of 102 services) Show remaining 101			
DynamoDB	Limited: Read, Write	TableName = myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}

DynamoDbRowCognitoID DynamoDB (允许) 服务摘要：

Action (4 of 25) Show remaining 21	Resource	Request condition
Read (1 of 14 actions)		
GetItem	TableName = myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}
Write (3 of 10 actions)		
DeleteItem	TableName = myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}
PutItem	TableName = myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}
UpdateItem	TableName = myDynamoTable	dynamodb:LeadingKeys = \${cognito-identity.amazonaws.com:sub}

GetItem (List) 操作摘要：

Resource	Region	Account	Request
TableName = myDynamoTable	us-west-1	123456789012	dynamodb: \${cognito-identity.ar

策略 3 : MultipleResourceCondition

此策略包括多个资源和条件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": ["arn:aws:s3:::Apple_bucket/*"],
      "Condition": {"StringEquals": {"s3:x-amz-acl": ["public-read"]}}
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": ["arn:aws:s3:::Orange_bucket/*"],
      "Condition": {"StringEquals": {
        "s3:x-amz-acl": ["custom"],
        "s3:x-amz-grant-full-control": ["1234"]
      }}
    }
  ]
}
```

MultipleResourceCondition 策略摘要：

Service	Access level	Resource	Request
Allow (1 of 100 services) Show remaining 99			
S3	Limited: Write, Permissions management	Multiple	Multiple

MultipleResourceCondition S3 (允许) 服务摘要：

Action (2 of 52 actions) Show remaining 50	Resource	Request
Write (1 of 21 actions)		
PutObject	Multiple	Multiple
Permissions management (1 of 5 actions)		
PutObjectAcl	Multiple	Multiple

PutObject (Write) 操作摘要：

Resource	Region	Account	Request condition
BucketName = Orange_bucket, ObjectPath = All	All regions	All accounts	Multiple
BucketName = Apple_bucket, ObjectPath = All	All regions	All accounts	s3:x-amz-acl = public-read

策略 4 : EC2_troubleshoot

以下策略允许用户获取正在运行的 Amazon EC2 实例的截图，这可以帮助排查 EC2 故障。该策略还允许查看有关 Amazon S3 开发人员存储桶中的项目的信息。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:GetConsoleScreenshot"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::developer"
      ]
    }
  ]
}
```

EC2_Troubleshoot 策略摘要：

Service	Access level	Resource	Request condition
Allow (2 of 102 services) Show remaining 100			
EC2	Limited: Read	All resources	None
S3	Limited: List	BucketName = developer	None

EC2_Troubleshoot S3 (允许) 服务摘要：

Action (1 of 52) Show remaining 51	Resource	Request condition
List (1 of 4 actions)		
ListBucket	BucketName = developer	None

ListBucket (List) 操作摘要：

Filter			
Resource	Region	Account	Request
BucketName = developer	All regions	All accounts	None

策略 5 : Unrecognized_Service_Action

以下策略旨在提供对 DynamoDB 的完全访问，但由于 dynamodb 错误地拼写为 dynamobd，导致访问失败。该策略旨在允许在 us-east-2 区域中访问某些 Amazon EC2 操作，但在 ap-northeast-2 区域中拒绝该访问。不过，由于在 RebootInstances 操作期间出现无法识别的 o，未显式拒绝在 ap-

northeast-2 区域中重新引导实例的访问权限。此示例显示如何使用策略摘要来定位策略中的错误。要了解如何根据策略摘要中的信息编辑策略，请参阅[编辑策略以纠正警告 \(p. 485\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "ec2:RunInstances",
        "ec2:StartInstances",
        "ec2:StopInstances",
        "ec2:RebootInstances"
      ],
      "Resource": "*",
      "Effect": "Deny",
      "Condition": {
        "StringEquals": {
          "ec2:Region": "ap-northeast-2"
        }
      }
    },
    {
      "Action": [
        "ec2:RunInstances",
        "ec2:StartInstances",
        "ec2:StopInstances",
        "ec2:RebootInstances"
      ],
      "Resource": "*",
      "Effect": "Allow",
      "Condition": {
        "StringEquals": {
          "ec2:Region": "us-east-2"
        }
      }
    }
  ]
}
```

Unrecognized_Service_Action 策略摘要：

Service ▼	Access level	Resource	Request condition
Unrecognized services			
dynamodb ⚠			
Explicit deny (1 of 103 services)			
EC2 ⚠	Limited: Write	All resources	ec2:Region = ap-northeast-2
Allow (1 of 103 services) Show remaining 102			
EC2	Limited: Write	All resources	ec2:Region = us-east-2

Unrecognized_Service_Action EC2 (Explicit deny) 服务摘要：

Action (3 of 229) Show remaining 226	Resource	Request condition
Unrecognized actions		
RebootInstances 		
Write (3 of 157 actions)		
RunInstances	All resources	ec2:Region = ap-northeast-2
StartInstances	All resources	ec2:Region = ap-northeast-2
StopInstances	All resources	ec2:Region = ap-northeast-2

Unrecognized_Service_Action StartInstances (Write) 操作摘要：

Resource	Region	Account	Request condition
All resources	All regions	All accounts	ec2:Region = ap-northeast-2

策略 6：CodeBuild_CodeCommit_CodeDeploy

此策略提供对特定 CodeBuild、CodeCommit 和 CodeDeploy 资源的访问。由于这些资源特定于每个服务，因此它们只与匹配的服务一起出现。如果您包含的资源与 Action 元素中的任何服务均不匹配，则该资源将出现在所有操作摘要中。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1487980617000",
      "Effect": "Allow",
      "Action": [
        "codebuild:*",
        "codecommit:*",
        "codedeploy:*"
      ],
      "Resource": [
        "arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project",
        "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo",
        "arn:aws:codedeploy:us-east-2:123456789012:application:WordPress_App",
        "arn:aws:codedeploy:us-east-2:123456789012:instance/AssetTag*"
      ]
    }
  ]
}
```

CodeBuild_CodeCommit_CodeDeploy 策略摘要：

Service ▼	Access level	Resource
Allow (3 of 103 services) Show remaining 100		
CodeBuild	Limited: List, Read, Write	arn:aws:codebuild: 2:12345678901: project
CodeCommit	Full: Read, Write Limited: List	arn:aws:codecommit: 2:12345678901:
CodeDeploy	Limited: List, Read, Write	Multiple

CodeBuild_CodeCommit_CodeDeploy CodeBuild (Allow) 服务摘要：

Action (9 of 15) Show remaining 6	Resource
List (1 of 3 actions)	
ListBuildsForProject	arn:aws:codebuild:us-east-2:12345678901:project
Read (2 of 5 actions)	
BatchGetBuilds	arn:aws:codebuild:us-east-2:12345678901:project
BatchGetProjects	arn:aws:codebuild:us-east-2:12345678901:project
Write (6 of 7 actions)	
BatchDeleteBuilds	arn:aws:codebuild:us-east-2:12345678901:project
CreateProject	arn:aws:codebuild:us-east-2:12345678901:project
DeleteProject	arn:aws:codebuild:us-east-2:12345678901:project
StartBuild	arn:aws:codebuild:us-east-2:12345678901:project
StopBuild	arn:aws:codebuild:us-east-2:12345678901:project
UpdateProject	arn:aws:codebuild:us-east-2:12345678901:project

CodeBuild_CodeCommit_CodeDeploy StartBuild (Write) 操作摘要：

Resource	Region	Account
codebuild:us-east-2:123456789012:project/my-demo-project	us-east-2	1234

访问 IAM 资源所需的权限

Resources (资源) 是服务中的对象。IAM 资源包括组、用户、角色和策略。如果您使用 Amazon Web Services 账户根用户凭证进行登录，则在管理 IAM 凭证或 IAM 资源方面没有任何限制。不过，必须显式为 IAM 用户授予权限以管理凭证或 IAM 资源。您可以将基于身份的策略附加到用户以执行此操作。

Note

在整个 Amazon 文档中，在提及 IAM 策略而未提到任何特定类别时，我们指的是基于身份的客户托管策略。有关策略类别的详细信息，请参阅[the section called “策略和权限” \(p. 355\)](#)。

用于管理 IAM 身份的权限

管理 IAM 组、用户、角色和凭证所需的权限通常与该任务的 API 操作相对应。例如，要创建 IAM 用户，您必须具有执行相应 API 命令的 `iam:CreateUser` 权限：[CreateUser](#)。要允许 IAM 用户创建其他 IAM 用户，您可以将类似下面的 IAM 策略附加到该用户：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:CreateUser",
    "Resource": "*"
  }
}
```

在策略中，Resource 元素的值取决于操作以及操作能够影响的资源。在上述示例中，策略允许用户创建任何用户 (* 是与所有字符串匹配的通配符)。相反，仅允许用户更改自己的访问密钥的策略 (API 操作 [CreateAccessKey](#) 和 [UpdateAccessKey](#)) 通常具有一个 Resource 元素。在此情况下，ARN 包含一个解析为当前用户名的变量 (`${aws:username}`)，如下示例中所示：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListUsersForConsole",
      "Effect": "Allow",
      "Action": "iam:ListUsers",
      "Resource": "arn:aws:iam::*:user/*"
    },
    {
      "Sid": "ViewAndUpdateAccessKeys",
      "Effect": "Allow",
      "Action": [
        "iam:UpdateAccessKey",
        "iam:CreateAccessKey",
        "iam:ListAccessKeys"
      ],
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    }
  ]
}
```

```
}  
}
```

在上述示例中，`${aws:username}` 是解析为当前用户的用户名的变量。有关策略变量的更多信息，请参阅 [IAM policy 元素：变量和标签 \(p. 964\)](#)。

一般来说，在操作名称中使用通配字符 (*) 可更轻松地为与特定任务相关的所有操作授予权限。例如，要允许用户执行任意 IAM 操作，您可以为操作使用 `iam:*`。要允许用户执行仅与访问密钥相关的操作，您可以在策略语句的 `iam:*AccessKey*` 元素中使用 Action。这可以授予用户执行 [CreateAccessKey](#)、[DeleteAccessKey](#)、[GetAccessKeyLastUsed](#)、[ListAccessKeys](#) 和 [UpdateAccessKey](#) 操作的权限。（如果未来在 IAM 中添加了名称包含“AccessKey”的操作，为 Action 元素使用 `iam:*AccessKey*` 还会授予用户执行该新操作的权限。）以下示例演示允许用户执行与自己的访问密钥有关的所有操作（将 `account-id` 替换为您的 Amazon 账户 ID）的策略：

```
{  
  "Version": "2012-10-17",  
  "Statement": {  
    "Effect": "Allow",  
    "Action": "iam:*AccessKey*",  
    "Resource": "arn:aws:iam::account-id:user/${aws:username}"  
  }  
}
```

有些任务（例如删除组）涉及多个操作：必须先从组中删除用户，再分离或删除组的策略，然后才能真正删除组。如果您希望一个用户能够删除群组，必须确保授予该用户执行所有相关操作的权限。

在 Amazon Web Services Management Console中工作的权限

上述示例演示允许用户使用 [Amazon CLI](#) 或 [Amazon 开发工具包](#) 执行操作的策略。

在用户使用控制台时，控制台向 IAM 发出请求以列出组、用户、角色和策略并获取与组、用户或角色关联的策略。控制台还会发出请求以获取 Amazon 账户信息以及有关主体的信息。主体是在控制台中发出请求的用户。

通常，要执行某个操作，您必须仅在策略中包含匹配的操作。要创建用户，您需要具有调用 `CreateUser` 操作的权限。通常，在使用控制台执行操作时，您必须具有相应的权限才能在控制台中显示、列出、获取或以其他方式查看资源。您必须具有该权限，以便能够在控制台中导航以执行指定的操作。例如，如果用户 Jorge 要使用控制台更改自己的访问密钥，他可以访问 IAM 控制台并选择 Users（用户）。这种操作会导致控制台发出 `ListUsers` 请求。如果 Jorge 没有执行 `iam:ListUsers` 操作的权限，在控制台尝试列出用户时，将会被拒绝访问。因此，Jorge 无法获取自己的名称和访问密钥，即使他具有执行 [CreateAccessKey](#) 和 [UpdateAccessKey](#) 操作的权限。

如果要为用户授予权限以使用 Amazon Web Services Management Console 管理组、用户、角色、策略和凭证，您需要包含控制台执行的操作的权限。有关可用于向用户授予这些权限的部分策略示例，请参阅 [管理 IAM 资源的策略示例 \(p. 510\)](#)。

授予跨 Amazon 账户权限

您可以直接向自己账户中的 IAM 用户授予对您的资源的访问权限。如果其他账户中的用户需要访问您的资源，您可以创建 IAM 角色，该角色是一个实体，其包含权限，但不与特定用户相关联。随后，其他账户中的用户可以使用角色，并根据您分配给角色的权限来访问资源。有关更多信息，请参阅 [在您拥有的另一个 Amazon 账户中向 IAM 用户提供访问权限 \(p. 161\)](#)。

Note

有些服务支持基于资源的策略，如 [基于身份的策略和基于资源的策略 \(p. 379\)](#) 中所述（例如 Amazon S3、Amazon SNS 和 Amazon SQS）。对于这些服务，使用角色的替代方法是将策略

附加到要共享的资源（存储桶、主题或队列）。基于资源的策略可以指定具有访问资源权限的 Amazon 账户。

某种服务访问其他服务的权限

许多 Amazon 服务都要访问其他 Amazon 服务。例如，几个 Amazon 服务（包括 Amazon EMR、Elastic Load Balancing 和 Amazon EC2 Auto Scaling）负责管理 Amazon EC2 实例。其他 Amazon 服务利用 Amazon S3 存储桶、Amazon SNS 主题、Amazon SQS 队列等。

在这些情况下管理许可的方案根据服务的不同而各异。下面是一些如何针对不同服务处理许可的示例：

- 在 Amazon EC2 Auto Scaling 中，用户必须拥有使用 Auto Scaling 的权限，但无需明确获得管理 Amazon EC2 实例的权限。
- 在 Amazon Data Pipeline 中，IAM 角色决定了管道可以执行哪些操作；用户需要权限来担任角色。（有关详细信息，请参阅 Amazon Data Pipeline 开发人员指南中的[使用 IAM 向管道授予权限](#)。）

有关如何正确配置权限以便 Amazon 服务能够完成您打算执行的任务的详细信息，请参阅您调用服务的文档。要了解如何为服务创建角色，请参阅[创建向 Amazon 服务委派权限的角色](#) (p. 219)。

使用 IAM 角色配置服务以代表您工作

当要配置 Amazon 服务代表您工作时，您通常要为 IAM 角色提供 ARN，用于定义允许该服务执行哪些操作。Amazon 会检查并确保您有权将角色传递给该服务。有关更多信息，请参阅[向用户授予权限以将角色传递给 Amazon 服务](#) (p. 240)。

所需的操作

操作是可以对资源执行的操作，例如，查看、创建、编辑和删除该资源。操作是由每个 Amazon 服务定义的。

要允许某个人执行操作，您必须在应用于进行调用的身份或受影响的资源的策略中包含所需的操作。通常，要提供执行某个操作所需的权限，您必须在策略中包含该操作。例如，要创建用户，您需要在策略中添加 CreateUser 操作。

在某些情况下，操作可能要求在策略中包含其他相关操作。例如，要为某个人提供在 Amazon Directory Service 中使用 ds:CreateDirectory 操作创建目录的权限，您必须在策略中包含以下操作：

- ds:CreateDirectory
- ec2:DescribeSubnets
- ec2:DescribeVpcs
- ec2:CreateSecurityGroup
- ec2:CreateNetworkInterface
- ec2:DescribeNetworkInterfaces
- ec2:AuthorizeSecurityGroupIngress
- ec2:AuthorizeSecurityGroupEgress

在使用可视化编辑器创建或编辑策略时，将显示警告和提示以帮助您选择您的策略所需的所有操作。

有关在 Amazon Directory Service 中创建目录所需的权限的更多信息，请参阅[示例 2：允许用户创建目录](#)。

管理 IAM 资源的策略示例

以下 IAM 策略示例允许用户执行与管理 IAM 用户、组和凭证相关的任务。这包括允许用户管理自己的密码、访问密钥和多重验证 (MFA) 设备的策略。

有关允许用户执行对其他 Amazon 服务 (如 Amazon S3、Amazon EC2 和 DynamoDB) 的任务的策略的示例, 请参阅 [IAM 基于身份的策略示例 \(p. 391\)](#)。

主题

- 允许用户列出账户的组、用户、策略等, 以供报告之用 (p. 511)
- 允许用户管理组的成员资格 (p. 511)
- 允许用户管理 IAM 用户 (p. 511)
- 允许用户设置账户密码策略 (p. 512)
- 允许用户生成和检索 IAM 凭证报告 (p. 512)
- 允许所有 IAM 操作 (管理员访问) (p. 512)

允许用户列出账户的组、用户、策略等, 以供报告之用

以下策略允许用户调用以字符串 Get 或 List 开头的任何 IAM 操作, 并生成报告。要查看示例策略, 请参阅 [IAM : 允许对 IAM 控制台进行只读访问 \(p. 422\)](#)。

允许用户管理组的成员资格

以下策略允许用户更新名为 MarketingGroup 的组的成员资格。要查看示例策略, 请参阅 [IAM : 允许以编程方式和在控制台中管理组的成员资格 \(p. 420\)](#)。

允许用户管理 IAM 用户

以下策略允许用户执行所有与管理 IAM 用户相关的任务, 但是不允许对其他实体执行操作, 如创建组或策略。允许的操作包括这些:

- 创建用户 ([CreateUser](#) 操作)。
- 删除用户。此任务需要授予执行以下所有操作的权限: [DeleteSigningCertificate](#)、[DeleteLoginProfile](#)、[RemoveUserFromGroup](#) 和 [DeleteUser](#)。
- 列出账户和组中的用户 ([GetUser](#)、[ListUsers](#) 和 [ListGroupsForUser](#) 操作)。
- 列出和删除用户的策略 ([ListUserPolicies](#)、[ListAttachedUserPolicies](#)、[DetachUserPolicy](#)、[DeleteUserPolicy](#) 操作)
- 重命名或更改用户的路径 ([UpdateUser](#) 操作)。Resource 元素必须包括涉及源路径和目标路径的 ARN。有关路径的更多信息, 请参阅 [易记名称和路径 \(p. 896\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUsersToPerformUserActions",
      "Effect": "Allow",
      "Action": [
        "iam:ListPolicies",
        "iam:GetPolicy",
        "iam:UpdateUser",
        "iam:AttachUserPolicy",
        "iam:ListEntitiesForPolicy",
        "iam:DeleteUserPolicy",
        "iam:DeleteUser",
        "iam:ListUserPolicies",
        "iam:CreateUser",
        "iam:RemoveUserFromGroup",
```

```

        "iam:AddUserToGroup",
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:PutUserPolicy",
        "iam:ListAttachedUserPolicies",
        "iam:ListUsers",
        "iam:GetUser",
        "iam:DetachUserPolicy"
    ],
    "Resource": "*"
  },
  {
    "Sid": "AllowUsersToSeeStatsOnIAMConsoleDashboard",
    "Effect": "Allow",
    "Action": [
      "iam:GetAccount*",
      "iam:ListAccount*"
    ],
    "Resource": "*"
  }
]
}

```

上述策略中包含的很多权限允许用户在 Amazon Web Services Management Console 中执行任务。仅从 [Amazon CLI](#)、[Amazon SDK](#) 或 IAM HTTP 查询 API 执行与用户相关的任务的用户可能不需要特定权限。例如，如果用户已知道从用户分离的策略的 ARN，则不需要 `iam:ListAttachedUserPolicies` 权限。用户所需权限的确切列表取决于用户管理其他用户时必须执行的任务。

以下策略中的权限允许通过 Amazon Web Services Management Console 访问用户任务：

- `iam:GetAccount*`
- `iam:ListAccount*`

允许用户设置账户密码策略

您可以授予某些用户获取和更新您的 Amazon 账户的 [密码策略](#) (p. 88) 的权限。要查看示例策略，请参阅 [IAM：允许以编程方式和在控制台中设置账户密码要求](#) (p. 424)。

允许用户生成和检索 IAM 凭证报告

您可以授予用户生成和下载报告的权限，该报告列出了您 Amazon 账户中的所有用户。该报告还列出了各种用户凭证的状态，包括密码、访问密钥、MFA 设备和签名证书。有关凭证报告的更多信息，请参阅 [获取您 Amazon 账户的凭证报告](#) (p. 141)。要查看示例策略，请参阅 [IAM：生成和检索 IAM 凭证报告](#) (p. 420)。

允许所有 IAM 操作（管理员访问）

您可以授予某些用户在 IAM 中执行所有操作的管理员权限，包括管理密码、访问密钥、MFA 设备和用户凭证。以下示例策略授予这些权限。

Warning

当您向用户授予对 IAM 的完全访问权时，对用户可以向自己或他人授予的权限没有限制。用户可以创建新的 IAM 实体（用户或角色）并授予这些实体对您 Amazon 账户中所有资源的完全访问权限。您向用户授予对 IAM 的完全访问权限时，实际上是向用户授予对您 Amazon 账户中所有资源的完全访问权限。其中包括删除所有资源的权限。您应该仅将这些权限授予信任的管理员，还应对这些管理员强制采用多重身份验证 (MFA)。

```

{
  "Version": "2012-10-17",

```

```
"Statement": {  
  "Effect": "Allow",  
  "Action": "iam:*",  
  "Resource": "*"   
}
```

使用 Amazon 开发工具包的 IAM 代码示例

以下代码示例显示如何将 IAM 与 Amazon 软件开发工具包 (SDK) 一起使用。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。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 Amazon 开发工具包的 IAM 代码示例 \(p. 515\)](#)
 - [使用 Amazon 开发工具包的 IAM 操作 \(p. 517\)](#)
 - [使用 Amazon 开发工具包将 IAM policy 附加到角色 \(p. 518\)](#)
 - [使用 Amazon 开发工具包将 IAM policy 附加到用户 \(p. 526\)](#)
 - [使用 Amazon 开发工具包创建 IAM policy \(p. 528\)](#)
 - [使用 Amazon 开发工具包创建 IAM policy 版本 \(p. 535\)](#)
 - [使用 Amazon 开发工具包创建 IAM 角色 \(p. 536\)](#)
 - [使用 Amazon 开发工具包创建 IAM 服务相关角色 \(p. 542\)](#)
 - [使用 Amazon 开发工具包创建 IAM 用户 \(p. 545\)](#)
 - [使用 Amazon 开发工具包创建 IAM 访问密钥 \(p. 550\)](#)
 - [使用 Amazon 开发工具包为 IAM 账户创建别名 \(p. 556\)](#)
 - [使用 Amazon 开发工具包创建用户的内联 IAM policy \(p. 560\)](#)
 - [使用 Amazon 开发工具包删除 IAM policy \(p. 561\)](#)
 - [使用 Amazon 开发工具包删除 IAM 角色 \(p. 564\)](#)
 - [使用 Amazon 开发工具包删除 IAM 角色策略 \(p. 566\)](#)
 - [使用 Amazon 开发工具包删除 IAM 服务器证书 \(p. 567\)](#)
 - [使用 Amazon 开发工具包删除 IAM 服务相关角色 \(p. 568\)](#)
 - [使用 Amazon 开发工具包删除 IAM 用户 \(p. 569\)](#)
 - [使用 Amazon 开发工具包删除 IAM 访问密钥 \(p. 575\)](#)
 - [使用 Amazon 开发工具包删除 IAM 账户别名 \(p. 581\)](#)
 - [使用 Amazon 开发工具包删除用户的内联 IAM policy \(p. 584\)](#)
 - [使用 Amazon 开发工具包从角色分离 IAM policy \(p. 585\)](#)
 - [使用 Amazon 开发工具包从用户分离 IAM policy \(p. 590\)](#)
 - [使用 Amazon 开发工具包从 IAM 生成凭证报告 \(p. 593\)](#)
 - [使用 Amazon 开发工具包从 IAM 获取凭证报告 \(p. 593\)](#)
 - [使用 Amazon 开发工具包获取您的账户的详细 IAM 授权报告 \(p. 594\)](#)
 - [使用 Amazon 开发工具包获取 IAM policy \(p. 595\)](#)
 - [使用 Amazon 开发工具包获取 IAM policy 版本 \(p. 599\)](#)
 - [使用 Amazon 开发工具包获取 IAM 角色 \(p. 599\)](#)
 - [使用 Amazon 开发工具包获取 IAM 服务器证书 \(p. 603\)](#)
 - [使用 Amazon 开发工具包从 IAM 获取账户使用情况摘要 \(p. 604\)](#)
 - [使用 Amazon 开发工具包获取有关上次使用 IAM 访问密钥的数据 \(p. 605\)](#)
 - [使用 Amazon 开发工具包获取 IAM 账户密码策略 \(p. 608\)](#)

- [使用 Amazon 开发工具包列出 IAM 的 SAML 提供商 \(p. 612\)](#)
- [使用 Amazon 开发工具包列出用户的 IAM 访问密钥 \(p. 615\)](#)
- [使用 Amazon 开发工具包列出 IAM 账户别名 \(p. 620\)](#)
- [使用 Amazon 开发工具包列出 IAM 组 \(p. 624\)](#)
- [使用 Amazon 开发工具包列出 IAM 角色的内联策略 \(p. 628\)](#)
- [使用 Amazon 开发工具包列出 IAM policy \(p. 631\)](#)
- [使用 Amazon 开发工具包列出附加到 IAM 角色的策略 \(p. 635\)](#)
- [使用 Amazon 开发工具包列出 IAM 角色 \(p. 639\)](#)
- [使用 Amazon 开发工具包列出 IAM 服务器证书 \(p. 643\)](#)
- [使用 Amazon 开发工具包列出 IAM 用户 \(p. 645\)](#)
- [使用 Amazon 开发工具包更新 IAM 服务器证书 \(p. 650\)](#)
- [使用 Amazon 开发工具包更新 IAM 用户 \(p. 652\)](#)
- [使用 Amazon 开发工具包更新 IAM 访问密钥 \(p. 656\)](#)
- [使用 Amazon 开发工具包的 IAM 场景 \(p. 660\)](#)
 - [使用 Amazon STS 开发工具包创建 IAM 用户并使用 Amazon 代入角色 \(p. 660\)](#)
 - [使用 Amazon 开发工具包创建只读和读写 IAM 用户 \(p. 710\)](#)
 - [使用 Amazon 开发工具包管理 IAM 访问密钥 \(p. 716\)](#)
 - [使用 Amazon 开发工具包管理 IAM policy \(p. 719\)](#)
 - [使用 Amazon 开发工具包管理 IAM 角色 \(p. 723\)](#)
 - [使用 Amazon 开发工具包管理您的 IAM 账户 \(p. 725\)](#)
 - [使用 Amazon SDK 回滚 IAM policy 版本 \(p. 729\)](#)
- [适用于使用 Amazon SDK 的 IAM 的跨服务示例 \(p. 730\)](#)
 - [创建一个长期 Amazon EMR 集群并使用 Amazon SDK 运行多个步骤 \(p. 730\)](#)
 - [创建一个短期 Amazon EMR 集群并使用 Amazon SDK 运行一个步骤 \(p. 731\)](#)
- [使用 Amazon 开发工具包的 Amazon STS 代码示例 \(p. 732\)](#)
 - [使用 Amazon 开发工具包的 Amazon STS 操作 \(p. 732\)](#)
 - [使用 Amazon 开发工具包代入具有 Amazon STS 的角色 \(p. 732\)](#)
 - [使用 Amazon 开发工具包获取具有 Amazon STS 的会话令牌 \(p. 736\)](#)
 - [使用 Amazon 开发工具包的 Amazon STS 场景 \(p. 737\)](#)
 - [使用 Amazon 开发工具包代入需要具有 Amazon STS 的 MFA 令牌的 IAM 角色 \(p. 737\)](#)
 - [使用 Amazon 开发工具包为联合用户构建具有 Amazon STS 的 URL \(p. 741\)](#)
 - [使用 Amazon 开发工具包获取需要具有 Amazon STS 的 MFA 令牌的会话令牌 \(p. 743\)](#)

使用 Amazon 开发工具包的 IAM 代码示例

以下代码示例显示如何将 IAM 与 Amazon 软件开发工具包 (SDK) 一起使用。

这些示例可分为以下几类：

操作

展示如何调用具体服务函数的代码节选。

场景

展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

跨服务示例

跨多个 Amazon Web Services 工作的示例应用程序。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。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 Amazon 开发工具包的 IAM 操作 \(p. 517\)](#)
 - [使用 Amazon 开发工具包将 IAM policy 附加到角色 \(p. 518\)](#)
 - [使用 Amazon 开发工具包将 IAM policy 附加到用户 \(p. 526\)](#)
 - [使用 Amazon 开发工具包创建 IAM policy \(p. 528\)](#)
 - [使用 Amazon 开发工具包创建 IAM policy 版本 \(p. 535\)](#)
 - [使用 Amazon 开发工具包创建 IAM 角色 \(p. 536\)](#)
 - [使用 Amazon 开发工具包创建 IAM 服务相关角色 \(p. 542\)](#)
 - [使用 Amazon 开发工具包创建 IAM 用户 \(p. 545\)](#)
 - [使用 Amazon 开发工具包创建 IAM 访问密钥 \(p. 550\)](#)
 - [使用 Amazon 开发工具包为 IAM 账户创建别名 \(p. 556\)](#)
 - [使用 Amazon 开发工具包创建用户的内联 IAM policy \(p. 560\)](#)
 - [使用 Amazon 开发工具包删除 IAM policy \(p. 561\)](#)
 - [使用 Amazon 开发工具包删除 IAM 角色 \(p. 564\)](#)
 - [使用 Amazon 开发工具包删除 IAM 角色策略 \(p. 566\)](#)
 - [使用 Amazon 开发工具包删除 IAM 服务器证书 \(p. 567\)](#)
 - [使用 Amazon 开发工具包删除 IAM 服务相关角色 \(p. 568\)](#)
 - [使用 Amazon 开发工具包删除 IAM 用户 \(p. 569\)](#)
 - [使用 Amazon 开发工具包删除 IAM 访问密钥 \(p. 575\)](#)
 - [使用 Amazon 开发工具包删除 IAM 账户别名 \(p. 581\)](#)
 - [使用 Amazon 开发工具包删除用户的内联 IAM policy \(p. 584\)](#)
 - [使用 Amazon 开发工具包从角色分离 IAM policy \(p. 585\)](#)
 - [使用 Amazon 开发工具包从用户分离 IAM policy \(p. 590\)](#)
 - [使用 Amazon 开发工具包从 IAM 生成凭证报告 \(p. 593\)](#)
 - [使用 Amazon 开发工具包从 IAM 获取凭证报告 \(p. 593\)](#)
 - [使用 Amazon 开发工具包获取您的账户的详细 IAM 授权报告 \(p. 594\)](#)
 - [使用 Amazon 开发工具包获取 IAM policy \(p. 595\)](#)
 - [使用 Amazon 开发工具包获取 IAM policy 版本 \(p. 599\)](#)
 - [使用 Amazon 开发工具包获取 IAM 角色 \(p. 599\)](#)
 - [使用 Amazon 开发工具包获取 IAM 服务器证书 \(p. 603\)](#)
 - [使用 Amazon 开发工具包从 IAM 获取账户使用情况摘要 \(p. 604\)](#)
 - [使用 Amazon 开发工具包获取有关上次使用 IAM 访问密钥的数据 \(p. 605\)](#)
 - [使用 Amazon 开发工具包获取 IAM 账户密码策略 \(p. 608\)](#)
 - [使用 Amazon 开发工具包列出 IAM 的 SAML 提供商 \(p. 612\)](#)
 - [使用 Amazon 开发工具包列出用户的 IAM 访问密钥 \(p. 615\)](#)
 - [使用 Amazon 开发工具包列出 IAM 账户别名 \(p. 620\)](#)
 - [使用 Amazon 开发工具包列出 IAM 组 \(p. 624\)](#)
 - [使用 Amazon 开发工具包列出 IAM 角色的内联策略 \(p. 628\)](#)
 - [使用 Amazon 开发工具包列出 IAM policy \(p. 631\)](#)
 - [使用 Amazon 开发工具包列出附加到 IAM 角色的策略 \(p. 635\)](#)

- 使用 Amazon 开发工具包列出 IAM 角色 (p. 639)
- 使用 Amazon 开发工具包列出 IAM 服务器证书 (p. 643)
- 使用 Amazon 开发工具包列出 IAM 用户 (p. 645)
- 使用 Amazon 开发工具包更新 IAM 服务器证书 (p. 650)
- 使用 Amazon 开发工具包更新 IAM 用户 (p. 652)
- 使用 Amazon 开发工具包更新 IAM 访问密钥 (p. 656)
- 使用 Amazon 开发工具包的 IAM 场景 (p. 660)
 - 使用 Amazon STS 开发工具包创建 IAM 用户并使用 Amazon 代入角色 (p. 660)
 - 使用 Amazon 开发工具包创建只读和读写 IAM 用户 (p. 710)
 - 使用 Amazon 开发工具包管理 IAM 访问密钥 (p. 716)
 - 使用 Amazon 开发工具包管理 IAM policy (p. 719)
 - 使用 Amazon 开发工具包管理 IAM 角色 (p. 723)
 - 使用 Amazon 开发工具包管理您的 IAM 账户 (p. 725)
 - 使用 Amazon SDK 回滚 IAM policy 版本 (p. 729)
- 适用于使用 Amazon SDK 的 IAM 的跨服务示例 (p. 730)
 - 创建一个长期 Amazon EMR 集群并使用 Amazon SDK 运行多个步骤 (p. 730)
 - 创建一个短期 Amazon EMR 集群并使用 Amazon SDK 运行一个步骤 (p. 731)

使用 Amazon 开发工具包的 IAM 操作

以下代码示例演示了如何使用 Amazon 开发工具包来执行各个 IAM 操作。这些代码节选调用了 IAM API，但不旨在孤立运行。每个示例都包含一个指向 GitHub 的链接，其中包含了有关如何在上下文中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 Identity and Access Management API 参考](#)。

示例

- 使用 Amazon 开发工具包将 IAM policy 附加到角色 (p. 518)
- 使用 Amazon 开发工具包将 IAM policy 附加到用户 (p. 526)
- 使用 Amazon 开发工具包创建 IAM policy (p. 528)
- 使用 Amazon 开发工具包创建 IAM policy 版本 (p. 535)
- 使用 Amazon 开发工具包创建 IAM 角色 (p. 536)
- 使用 Amazon 开发工具包创建 IAM 服务相关角色 (p. 542)
- 使用 Amazon 开发工具包创建 IAM 用户 (p. 545)
- 使用 Amazon 开发工具包创建 IAM 访问密钥 (p. 550)
- 使用 Amazon 开发工具包为 IAM 账户创建别名 (p. 556)
- 使用 Amazon 开发工具包创建用户的内联 IAM policy (p. 560)
- 使用 Amazon 开发工具包删除 IAM policy (p. 561)
- 使用 Amazon 开发工具包删除 IAM 角色 (p. 564)
- 使用 Amazon 开发工具包删除 IAM 角色策略 (p. 566)
- 使用 Amazon 开发工具包删除 IAM 服务器证书 (p. 567)
- 使用 Amazon 开发工具包删除 IAM 服务相关角色 (p. 568)
- 使用 Amazon 开发工具包删除 IAM 用户 (p. 569)
- 使用 Amazon 开发工具包删除 IAM 访问密钥 (p. 575)
- 使用 Amazon 开发工具包删除 IAM 账户别名 (p. 581)
- 使用 Amazon 开发工具包删除用户的内联 IAM policy (p. 584)

- 使用 Amazon 开发工具包从角色分离 IAM policy (p. 585)
- 使用 Amazon 开发工具包从用户分离 IAM policy (p. 590)
- 使用 Amazon 开发工具包从 IAM 生成凭证报告 (p. 593)
- 使用 Amazon 开发工具包从 IAM 获取凭证报告 (p. 593)
- 使用 Amazon 开发工具包获取您的账户的详细 IAM 授权报告 (p. 594)
- 使用 Amazon 开发工具包获取 IAM policy (p. 595)
- 使用 Amazon 开发工具包获取 IAM policy 版本 (p. 599)
- 使用 Amazon 开发工具包获取 IAM 角色 (p. 599)
- 使用 Amazon 开发工具包获取 IAM 服务器证书 (p. 603)
- 使用 Amazon 开发工具包从 IAM 获取账户使用情况摘要 (p. 604)
- 使用 Amazon 开发工具包获取有关上次使用 IAM 访问密钥的数据 (p. 605)
- 使用 Amazon 开发工具包获取 IAM 账户密码策略 (p. 608)
- 使用 Amazon 开发工具包列出 IAM 的 SAML 提供商 (p. 612)
- 使用 Amazon 开发工具包列出用户的 IAM 访问密钥 (p. 615)
- 使用 Amazon 开发工具包列出 IAM 账户别名 (p. 620)
- 使用 Amazon 开发工具包列出 IAM 组 (p. 624)
- 使用 Amazon 开发工具包列出 IAM 角色的内联策略 (p. 628)
- 使用 Amazon 开发工具包列出 IAM policy (p. 631)
- 使用 Amazon 开发工具包列出附加到 IAM 角色的策略 (p. 635)
- 使用 Amazon 开发工具包列出 IAM 角色 (p. 639)
- 使用 Amazon 开发工具包列出 IAM 服务器证书 (p. 643)
- 使用 Amazon 开发工具包列出 IAM 用户 (p. 645)
- 使用 Amazon 开发工具包更新 IAM 服务器证书 (p. 650)
- 使用 Amazon 开发工具包更新 IAM 用户 (p. 652)
- 使用 Amazon 开发工具包更新 IAM 访问密钥 (p. 656)

使用 Amazon 开发工具包将 IAM policy 附加到角色

以下代码示例显示如何将 IAM policy 添加到角色。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
/// <summary>
/// Attach the policy to the role so that the user can assume it.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="policyArn">The ARN of the policy to attach.</param>
/// <param name="roleName">The name of the role to attach the policy to.</
param>
public static async Task AttachRoleAsync(
    AmazonIdentityManagementServiceClient client,
    string policyArn,
    string roleName)
{
    var request = new AttachRolePolicyRequest
```

```
{
    PolicyArn = policyArn,
    RoleName = roleName,
};

var response = await client.AttachRolePolicyAsync(request);

if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
{
    Console.WriteLine("Successfully attached the policy to the role.");
}
else
{
    Console.WriteLine("Could not attach the policy.");
}
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [AttachRolePolicy](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// AttachRolePolicy

_, err = service.AttachRolePolicy(context.Background(),
&iam.AttachRolePolicyInput{
    PolicyArn: aws.String(ExamplePolicyARN),
    RoleName:  aws.String(ExampleRoleName),
})

if err != nil {
    panic("Couldn't apply a policy to the role!")
}

fmt.Println("## Attached policy " + ExamplePolicyARN + " to role " +
ExampleRoleName)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [AttachRolePolicy](#)。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

    try {
```

```

        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
    .roleName(roleName)
    .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        // Ensure that the policy is not attached to this role
String polArn = "";
for (AttachedPolicy policy: attachedPolicies) {
    polArn = policy.policyArn();
    if (polArn.compareTo(policyArn)==0) {
        System.out.println(roleName + " policy is already attached to
this role.");
        return;
    }
}

        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
    .roleName(roleName)
    .policyArn(policyArn)
    .build();

        iam.attachRolePolicy(attachRequest);

        System.out.println("Successfully attached policy " + policyArn +
" to role " + roleName);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}

```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [AttachRolePolicy](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```

import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };

```

附加策略。

```

// Import required AWS SDK clients and commands for Node.js.

```

```
import { iamClient } from "../libs/iamClient.js";
import {
  ListAttachedRolePoliciesCommand,
  AttachRolePolicyCommand,
} from "@aws-sdk/client-iam";

// Set the parameters.
const ROLENAME = "ROLE_NAME";
const paramsRoleList = { RoleName: ROLENAME }; //ROLE_NAME
export const params = {
  PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
  RoleName: ROLENAME,
};
export const run = async () => {
  try {
    const data = await iamClient.send(
      new ListAttachedRolePoliciesCommand(paramsRoleList)
    );
    const myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (_val, index) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        console.log(
          "AmazonDynamoDBFullAccess is already attached to this role."
        );
        process.exit();
      }
    });
  } catch (err) {
    console.log("Error", err);
  }
  try {
    const data = await iamClient.send(new AttachRolePolicyCommand(params));
    console.log("Role attached successfully");
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [AttachRolePolicy](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var paramsRoleList = {
  RoleName: process.argv[2]
};

iam.listAttachedRolePolicies(paramsRoleList, function(err, data) {
  if (err) {
```

```
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === 'AmazonDynamoDBFullAccess') {
        console.log("AmazonDynamoDBFullAccess is already attached to this role.")
        process.exit();
      }
    });
    var params = {
      PolicyArn: 'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess',
      RoleName: process.argv[2]
    };
    iam.attachRolePolicy(params, function(err, data) {
      if (err) {
        console.log("Unable to attach policy to role", err);
      } else {
        console.log("Role attached successfully");
      }
    });
  }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [AttachRolePolicy](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun attachIAMRolePolicy(roleNameVal: String, policyArnVal: String) {

    val request = ListAttachedRolePoliciesRequest {
        roleName = roleNameVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkList(attachedPolicies, policyArnVal)
            if (checkStatus == -1)
                return
        }

        val policyRequest = AttachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }
        iamClient.attachRolePolicy(policyRequest)
    }
}
```



```

        println("Successfully attached policy $policyArnVal to role $roleNameVal")
    }
}

fun checkList(attachedPolicies: List<AttachedPolicy>, policyArnVal: String): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [AttachRolePolicy](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```

$uuid = uniqid();
$service = new IamService();

$assumeRolePolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Principal\": {\"AWS\": \"{user['Arn']}\"},
        \"Action\": \"sts:AssumeRole\"
    }]
}";

$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
    $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Action\": \"s3:ListAllMyBuckets\",
        \"Resource\": \"arn:aws:s3:::*\"}]
}";

$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
    $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
    $listAllBucketsPolicy['Arn']);

public function attachRolePolicy($roleName, $policyArn)
{
    return $this->customWaiter(function () use ($roleName, $policyArn) {
        $this->iamClient->attachRolePolicy([
            'PolicyArn' => $policyArn,
            'RoleName' => $roleName,

```

```
    });  
  });  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [AttachRolePolicy](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

使用 Boto3 策略对象将策略附加到角色。

```
def attach_to_role(role_name, policy_arn):  
    """  
    Attaches a policy to a role.  
  
    :param role_name: The name of the role. **Note** this is the name, not the ARN.  
    :param policy_arn: The ARN of the policy.  
    """  
    try:  
        iam.Policy(policy_arn).attach_role(RoleName=role_name)  
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)  
    except ClientError:  
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,  
            role_name)  
        raise
```

使用 Boto3 角色对象将策略附加到角色。

```
def attach_policy(role_name, policy_arn):  
    """  
    Attaches a policy to a role.  
  
    :param role_name: The name of the role. **Note** this is the name, not the ARN.  
    :param policy_arn: The ARN of the policy.  
    """  
    try:  
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)  
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)  
    except ClientError:  
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,  
            role_name)  
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [AttachRolePolicy](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Creates a policy that grants permission to list S3 buckets in the account, and
# then attaches the policy to a role.
#
# @param policy_name [String] The name to give the policy.
# @param role [Aws::IAM::Role] The role that the policy is attached to.
# @return [Aws::IAM::Policy] The newly created policy.
def create_and_attach_role_policy(policy_name, role)
  policy = @iam_resource.create_policy(
    policy_name: policy_name,
    policy_document: {
      Version: "2012-10-17",
      Statement: [{
        Effect: "Allow",
        Action: "s3:ListAllMyBuckets",
        Resource: "arn:aws:s3:::*"
      }]
    }.to_json)
  role.attach_policy(policy_arn: policy.arn)
  puts("Created policy #{policy.policy_name} and attached it to role
#{role.name}.")
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't create a policy and attach it to role #{role.name}. Here's why:
")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    policy
  end
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [AttachRolePolicy](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn attach_role_policy(
  client: &iamClient,
  role: &Role,
  policy: &Policy,
) -> Result<AttachRolePolicyOutput, SdkError<AttachRolePolicyError>> {
  client
    .attach_role_policy()
    .role_name(role.role_name.as_ref().unwrap())
    .policy_arn(policy.arn.as_ref().unwrap())
    .send()
    .await
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [AttachRolePolicy](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包将 IAM policy 附加到用户

以下代码示例显示如何将 IAM policy 附加到用户。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
package main

import (
    "context"
    "flag"
    "fmt"

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

// IAMAttachRolePolicyAPI defines the interface for the AttachRolePolicy function.
// We use this interface to test the function using a mocked service.
type IAMAttachRolePolicyAPI interface {
    AttachRolePolicy(ctx context.Context,
        params *iam.AttachRolePolicyInput,
        optFns ...func(*iam.Options)) (*iam.AttachRolePolicyOutput, error)
}

// AttachDynamoFullPolicy attaches an Amazon DynamoDB full-access policy to an AWS
// Identity and Access Management (IAM) role.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, an AttachRolePolicyOutput object containing the result of the
//     service call and nil.
//     Otherwise, nil and an error from the call to AttachRolePolicy.
func AttachDynamoFullPolicy(c context.Context, api IAMAttachRolePolicyAPI, input
    *iam.AttachRolePolicyInput) (*iam.AttachRolePolicyOutput, error) {
    return api.AttachRolePolicy(c, input)
}

func main() {
    roleName := flag.String("r", "", "The name of the IAM role")
    policyName := flag.String("p", "", "The name of the policy to attach to the role")
    flag.Parse()

    if *roleName == "" || *policyName == "" {
        fmt.Println("You must supply a role and policy name (-r ROLE -p POLICY)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }
}
```

```
client := iam.NewFromConfig(cfg)

policyArn := "arn:aws:iam::aws:policy/" + *policyName

input := &iam.AttachRolePolicyInput{
    PolicyArn: &policyArn,
    RoleName:  roleName,
}

_, err = AttachDynamoFullPolicy(context.TODO(), client, input)
if err != nil {
    fmt.Println("Unable to attach policy " + *policyName + " to role " + *roleName)
    return
}

fmt.Println("Policy " + *policyName + " attached to role " + *roleName)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [AttachUserPolicy](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
            user_name)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [AttachUserPolicy](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn attach_user_policy(  
    client: &iamClient,  
    user_name: &str,  
    policy_arn: &str,  
) -> Result<(), iamError> {  
    client  
        .attach_user_policy()  
        .user_name(user_name)  
        .policy_arn(policy_arn)  
        .send()  
        .await?;  
  
    Ok(())  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [AttachUserPolicy](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包创建 IAM policy

以下代码示例显示如何创建 IAM policy。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
/// <summary>  
/// Create a policy to allow a user to list the buckets in an account.  
/// </summary>  
/// <param name="client">The initialized IAM client object.</param>  
/// <param name="policyName">The name of the poicy to create.</param>  
/// <param name="policyDocument">The permissions policy document.</param>  
/// <returns>The newly created ManagedPolicy object.</returns>  
public static async Task<ManagedPolicy> CreatePolicyAsync(  
    AmazonIdentityManagementServiceClient client,  
    string policyName,  
    string policyDocument)  
{  
    var request = new CreatePolicyRequest  
    {  
        PolicyName = policyName,  
        PolicyDocument = policyDocument,  
    };  
  
    var response = await client.CreatePolicyAsync(request);  
  
    return response.Policy;  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [CreatePolicy](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// CreatePolicy

fmt.Println("## CreatePolicy")

policyDocument := `{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Query",
        "dynamodb:Scan",
        "dynamodb:UpdateItem"
      ],
      "Resource": [
        "arn:aws:dynamodb:us-west-2:123456789012:table/mytable",
        "arn:aws:dynamodb:us-west-2:123456789012:table/mytable/*"
      ]
    }
  ]
}`

createPolicyResult, err := service.CreatePolicy(context.Background(),
&iam.CreatePolicyInput{
  PolicyDocument: &policyDocument,
  PolicyName:     aws.String(ExamplePolicyName),
})

if err != nil {
  panic("Couldn't create policy!" + err.Error())
}

fmt.Print("Created a new policy: " + *createPolicyResult.Policy.Arn)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [CreatePolicy](#)。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static String createIAMPolicy(IamClient iam, String policyName ) {

    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();
```

```
CreatePolicyRequest request = CreatePolicyRequest.builder()
    .policyName(policyName)
    .policyDocument(PolicyDocument)
    .build();

CreatePolicyResponse response = iam.createPolicy(request);

// Wait until the policy is created.
GetPolicyRequest polRequest = GetPolicyRequest.builder()
    .policyArn(response.policy().arn())
    .build();

WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
return response.policy().arn();

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

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [CreatePolicy](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

创建策略。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { CreatePolicyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const myManagedPolicy = {
    Version: "2012-10-17",
    Statement: [
        {
            Effect: "Allow",
            Action: "logs:CreateLogGroup",
            Resource: "RESOURCE_ARN", // RESOURCE_ARN
        },
    ],
};
```

```
{
  Effect: "Allow",
  Action: [
    "dynamodb:DeleteItem",
    "dynamodb:GetItem",
    "dynamodb:PutItem",
    "dynamodb:Scan",
    "dynamodb:UpdateItem",
  ],
  Resource: "DYNAMODB_POLICY_NAME", // DYNAMODB_POLICY_NAME; For example,
  "myDynamoDBName".
},
],
};
export const params = {
  PolicyDocument: JSON.stringify(myManagedPolicy),
  PolicyName: "IAM_POLICY_NAME",
};

export const run = async () => {
  try {
    const data = await iamClient.send(new CreatePolicyCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [CreatePolicy](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var myManagedPolicy = {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:CreateLogGroup",
      "Resource": "RESOURCE_ARN"
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Scan",
        "dynamodb:UpdateItem"
      ]
    }
  ]
};
```

```

        ],
        "Resource": "RESOURCE_ARN"
    }
  ]
};

var params = {
  PolicyDocument: JSON.stringify(myManagedPolicy),
  PolicyName: 'myDynamoDBPolicy',
};

iam.createPolicy(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});

```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [CreatePolicy](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```

suspend fun createIAMPolicy(policyNameVal: String?): String {

    val policyDocumentVal = "{" +
        "  \"Version\": \"2012-10-17\", " +
        "  \"Statement\": [ " +
        "    { " +
        "      \"Effect\": \"Allow\", " +
        "      \"Action\": [ " +
        "        \"dynamodb:DeleteItem\", " +
        "        \"dynamodb:GetItem\", " +
        "        \"dynamodb:PutItem\", " +
        "        \"dynamodb:Scan\", " +
        "        \"dynamodb:UpdateItem\" " +
        "      ], " +
        "      \"Resource\": \"*\" " +
        "    } " +
        "  ] " +
        "}"

    val request = CreatePolicyRequest {
      policyName = policyNameVal
      policyDocument = policyDocumentVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
      val response = iamClient.createPolicy(request)
      return response.policy?.arn.toString()
    }
}

```

```
}  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [CreatePolicy](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
$uuid = uniqid();  
$service = new IamService();  
  
$listAllBucketsPolicyDocument = "{  
    \"Version\": \"2012-10-17\",  
    \"Statement\": [{  
        \"Effect\": \"Allow\",  
        \"Action\": \"s3:ListAllMyBuckets\",  
        \"Resource\": \"arn:aws:s3:::*\"}]  
}";  
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",  
    $listAllBucketsPolicyDocument);  
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";  
  
public function createPolicy(string $policyName, string $policyDocument)  
{  
    $result = $this->customWaiter(function () use ($policyName,  
    $policyDocument) {  
        return $this->iamClient->createPolicy([  
            'PolicyName' => $policyName,  
            'PolicyDocument' => $policyDocument,  
        ]);  
    });  
    return $result['Policy'];  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [CreatePolicy](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def create_policy(name, description, actions, resource_arn):  
    """  
    Creates a policy that contains a single statement.  
  
    :param name: The name of the policy to create.  
    :param description: The description of the policy.  
    :param actions: The actions allowed by the policy. These typically take the  
        form of service:action, such as s3:PutObject.  
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy
```

```

        applies to. This ARN can contain wildcards, such as
        'arn:aws:s3:::my-bucket/*' to allow actions on all objects
        in the bucket named 'my-bucket'.
:return: The newly created policy.
"""
policy_doc = {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": actions,
            "Resource": resource_arn
        }
    ]
}
try:
    policy = iam.create_policy(
        PolicyName=name, Description=description,
        PolicyDocument=json.dumps(policy_doc))
    logger.info("Created policy %s.", policy.arn)
except ClientError:
    logger.exception("Couldn't create policy %s.", name)
    raise
else:
    return policy

```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [CreatePolicy](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```

# Creates a policy that grants permission to list S3 buckets in the account, and
# then attaches the policy to a role.
#
# @param policy_name [String] The name to give the policy.
# @param role [Aws::IAM::Role] The role that the policy is attached to.
# @return [Aws::IAM::Policy] The newly created policy.
def create_and_attach_role_policy(policy_name, role)
  policy = @iam_resource.create_policy(
    policy_name: policy_name,
    policy_document: {
      Version: "2012-10-17",
      Statement: [{
        Effect: "Allow",
        Action: "s3:ListAllMyBuckets",
        Resource: "arn:aws:s3:::*"
      }]
    }.to_json)
  role.attach_policy(policy_arn: policy.arn)
  puts("Created policy #{policy.policy_name} and attached it to role
#{role.name}.")
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't create a policy and attach it to role #{role.name}. Here's why:
")
    puts("\t#{e.code}: #{e.message}")
    raise
  else

```



```
policy
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [CreatePolicy](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn create_policy(
    client: &iamClient,
    policy_name: &str,
    policy_document: &str,
) -> Result<Policy, iamError> {
    let policy = client
        .create_policy()
        .policy_name(policy_name)
        .policy_document(policy_document)
        .send()
        .await?;
    Ok(policy.policy.unwrap())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [CreatePolicy](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包创建 IAM policy 版本

以下代码示例显示如何创建 IAM policy 版本。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
```

```
:param resource_arn: The ARN of the resource this policy version applies to.
:param set_as_default: When True, this policy version is set as the default
                        version for the policy. Otherwise, the default
                        is not changed.
:return: The newly created policy version.
"""
policy_doc = {
    'Version': '2012-10-17',
    'Statement': [
        {
            'Effect': 'Allow',
            'Action': actions,
            'Resource': resource_arn
        }
    ]
}
try:
    policy = iam.Policy(policy_arn)
    policy_version = policy.create_version(
        PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default)
    logger.info(
        "Created policy version %s for policy %s.",
        policy_version.version_id, policy_version.arn)
except ClientError:
    logger.exception("Couldn't create a policy version for %s.", policy_arn)
    raise
else:
    return policy_version
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [CreatePolicyVersion](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包创建 IAM 角色

以下代码示例显示如何创建 IAM 角色。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
/// <summary>
/// Create a new IAM role which we can attach to a user.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="roleName">The name of the IAM role to create.</param>
/// <param name="rolePermissions">The permissions which the role will
have.</param>
/// <returns>A Role object representing the newly created role.</returns>
public static async Task<Role> CreateRoleAsync(
    AmazonIdentityManagementServiceClient client,
    string roleName,
    string rolePermissions)
```

```
{
    var request = new CreateRoleRequest
    {
        RoleName = roleName,
        AssumeRolePolicyDocument = rolePermissions,
    };

    var response = await client.CreateRoleAsync(request);

    return response.Role;
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [CreateRole](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// CreateRole
myRole, err := service.CreateRole(context.Background(), &iam.CreateRoleInput{
    RoleName:    aws.String(ExampleRoleName),
    Description: aws.String("My super awesome example role"),
    AssumeRolePolicyDocument: aws.String(`{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "ec2.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }`),
})

if err != nil {
    panic("Couldn't create role: " + err.Error())
}

fmt.Println("## Create Role")
fmt.Printf("The new role's ARN is %s \n", *myRole.Role.Arn)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [CreateRole](#)。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static String createIAMRole(IamClient iam, String rolename, String
fileLocation ) throws Exception {

    try {
        JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(jsonObject.toJSONString())
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is "+response.role().arn());

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

public static Object readJsonSimpleDemo(String filename) throws Exception {
    FileReader reader = new FileReader(filename);
    JSONParser jsonParser = new JSONParser();
    return jsonParser.parse(reader);
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [CreateRole](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

创建角色。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { CreateRoleCommand } from "@aws-sdk/client-iam";

// Sample assume role policy JSON.
const role_json = {
    Version: "2012-10-17",
    Statement: [
        {
            Effect: "Allow",
            Principal: {
                AWS: "USER_ARN", // The ARN of the user.
```

```

        },
        Action: "sts:AssumeRole",
    },
],
};
// Stringify the assume role policy JSON.
const myJson = JSON.stringify(role_json);

// Set the parameters.
const params = {
    AssumeRolePolicyDocument: myJson,
    Path: "/",
    RoleName: "ROLE_NAME"
};

const run = async () => {
    try {
        const data = await iamClient.send(new CreateRoleCommand(params));
        console.log("Success. Role created. Role Arn: ", data.Role.RoleName);
    } catch (err) {
        console.log("Error", err);
    }
};
run();

```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [CreateRole](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```

$uuid = uniqid();
$service = new IamService();

$assumeRolePolicyDocument = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Principal\": {\"AWS\": \"${user['Arn']}\"},
        \"Action\": \"sts:AssumeRole\"
    }]
}";

$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
    $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

/**
 * @param string $roleName
 * @param string $rolePolicyDocument
 * @return array
 * @throws AwsException
 */
public function createRole(string $roleName, string $rolePolicyDocument)
{
    $result = $this->customWaiter(function () use ($roleName,
        $rolePolicyDocument) {
        return $this->iamClient->createRole([
            'AssumeRolePolicyDocument' => $rolePolicyDocument,

```

```
        'RoleName' => $roleName,  
    ]});  
});  
return $result['Role'];  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [CreateRole](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def create_role(role_name, allowed_services):  
    """  
    Creates a role that lets a list of specified services assume the role.  
  
    :param role_name: The name of the role.  
    :param allowed_services: The services that can assume the role.  
    :return: The newly created role.  
    """  
    trust_policy = {  
        'Version': '2012-10-17',  
        'Statement': [{  
            'Effect': 'Allow',  
            'Principal': {'Service': service},  
            'Action': 'sts:AssumeRole'  
        } for service in allowed_services  
    ]  
}  
  
    try:  
        role = iam.create_role(  
            RoleName=role_name,  
            AssumeRolePolicyDocument=json.dumps(trust_policy))  
        logger.info("Created role %s.", role.name)  
    except ClientError:  
        logger.exception("Couldn't create role %s.", role_name)  
        raise  
    else:  
        return role
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [CreateRole](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Creates a role that can be assumed by a user.  
#
```



```
# @param role_name [String] The name to give the role.
# @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
# @return [Aws::IAM::Role] The newly created role.
def create_role(role_name, user)
  role = @iam_resource.create_role(
    role_name: role_name,
    assume_role_policy_document: {
      Version: "2012-10-17",
      Statement: [{
        Effect: "Allow",
        Principal: {'AWS': user.arn},
        Action: "sts:AssumeRole"
      }]
    }.to_json)
  puts("Created role #{role.name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't create a role for the demo. Here's why: ")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  role
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [CreateRole](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn create_role(
  client: &iamClient,
  role_name: &str,
  role_policy_document: &str,
) -> Result<Role, iamError> {
  let response: CreateRoleOutput = loop {
    if let Ok(response) = client
      .create_role()
      .role_name(role_name)
      .assume_role_policy_document(role_policy_document)
      .send()
      .await
    {
      break response;
    }
  };
  Ok(response.role.unwrap())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [CreateRole](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包创建 IAM 服务相关角色

以下代码示例显示如何创建 IAM 服务相关角色。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// CreateServiceLinkedRole

fmt.Println("## Create SLR for " + ExampleSLRService)
createSlrResult, err := service.CreateServiceLinkedRole(context.Background(),
&iam.CreateServiceLinkedRoleInput{
    AWSServiceName: aws.String(ExampleSLRService),
    Description:    aws.String(ExampleSLRDescription),
})

// NOTE: We don't consider this an error as running this example multiple times
// will cause an error.
if err != nil {
    fmt.Printf("Couldn't create service-linked role: %v\n", err.Error())
} else {

    fmt.Printf("Created service-linked role with ARN: %s\n",
        *createSlrResult.Role.Arn)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [CreateServiceLinkedRole](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

创建服务相关角色。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
```

```
import { CreateServiceLinkedRoleCommand } from "@aws-sdk/client-iam";
// Set the parameters.
const params = {
  AWSServiceName: "AWS_SERVICE_NAME" /* required */,
};

const run = async () => {
  try {
    const data = await iamClient.send(
      new CreateServiceLinkedRoleCommand(params)
    );
    console.log("Success", data);
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [CreateServiceLinkedRole](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
$uuid = uniqid();
$service = new IamService();

public function createServiceLinkedRole($awsServiceName, $customSuffix = "",
    $description = "")
{
    $createServiceLinkedRoleArguments = ['AWSServiceName' => $awsServiceName];
    if ($customSuffix) {
        $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
    }
    if ($description) {
        $createServiceLinkedRoleArguments['Description'] = $description;
    }
    return $this->iamClient-
>createServiceLinkedRole($createServiceLinkedRoleArguments);
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [CreateServiceLinkedRole](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def create_service_linked_role(service_name, description):
```

```
"""
Creates a service-linked role.

:param service_name: The name of the service that owns the role.
:param description: A description to give the role.
:return: The newly created role.
"""
try:
    response = iam.meta.client.create_service_linked_role(
        AWSServiceName=service_name, Description=description)
    role = iam.Role(response['Role']['RoleName'])
    logger.info("Created service-linked role %s.", role.name)
except ClientError:
    logger.exception("Couldn't create service-linked role for %s.",
service_name)
    raise
else:
    return role
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [CreateServiceLinkedRole](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Creates a service-linked role.
#
# @param service_name [String] The name of the service that owns the role.
# @param description [String] A description to give the role.
# @return [Aws::IAM::Role] The newly created role.
def create_service_linked_role(service_name, description)
    response = @iam_resource.client.create_service_linked_role(
        aws_service_name: service_name, description: description)
    role = @iam_resource.role(response.role.role_name)
    puts("Created service-linked role #{role.name}.")
rescue Aws::Errors::ServiceError => e
    puts("Couldn't create service-linked role for #{service_name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
else
    role
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [CreateServiceLinkedRole](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn create_service_linked_role(  
    client: &iamClient,  
    aws_service_name: String,  
    custom_suffix: Option<String>,  
    description: Option<String>,  
) -> Result<CreateServiceLinkedRoleOutput, SdkError<CreateServiceLinkedRoleError>>  
{  
    let response = client  
        .create_service_linked_role()  
        .aws_service_name(aws_service_name)  
        .set_custom_suffix(custom_suffix)  
        .set_description(description)  
        .send()  
        .await?;  
  
    Ok(response)  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [CreateServiceLinkedRole](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包创建 IAM 用户

以下代码示例显示如何创建 IAM 用户。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
/// <summary>  
/// Create a new IAM user.  
/// </summary>  
/// <param name="client">The initialized IAM client object.</param>  
/// <param name="userName">A string representing the user name of the  
/// new user.</param>  
/// <returns>The newly created user.</returns>  
public static async Task<User> CreateUserAsync(  
    AmazonIdentityManagementServiceClient client,  
    string userName)  
{  
    var request = new CreateUserRequest  
    {  
        UserName = userName,  
    };  
  
    var response = await client.CreateUserAsync(request);
```

```
        return response.User;
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [CreateUser](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// CreateUser

fmt.Println("## Create user " + ExampleUserName)

createUserResult, err := service.CreateUser(context.Background(),
&iam.CreateUserInput{
    UserName: aws.String(ExampleUserName),
})

if err != nil {
    panic("Couldn't create user: " + err.Error())
}

fmt.Printf("Created user %s\n", *createUserResult.User.Arn)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [CreateUser](#)。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static String createIAMUser(IamClient iam, String username ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        CreateUserResponse response = iam.createUser(request);

        // Wait until the user is created
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);
```



```
waitUntilUserExists.matched().response().ifPresent(System.out::println);
    return response.user().userName();

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

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [CreateUser](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

创建用户。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { GetUserCommand, CreateUserCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { UserName: "USER_NAME" }; //USER_NAME

export const run = async () => {
    try {
        const data = await iamClient.send(new GetUserCommand(params));
        console.log(
            "User " + "USER_NAME" + " already exists",
            data.User.UserId
        );
        return data;
    } catch (err) {
        try {
            const results = await iamClient.send(new CreateUserCommand(params));
            console.log("Success", results);
            return results;
        } catch (err) {
            console.log("Error", err);
        }
    }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [CreateUser](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  UserName: process.argv[2]
};

iam.getUser(params, function(err, data) {
  if (err && err.code === 'NoSuchEntity') {
    iam.createUser(params, function(err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  } else {
    console.log("User " + process.argv[2] + " already exists", data.User.UserId);
  }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [CreateUser](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request = CreateUserRequest {
        userName = usernameVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [CreateUser](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
$uuid = uniqid();
$service = new IamService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

/**
 * @param string $name
 * @return array
 * @throws AwsException
 */
public function createUser(string $name): array
{
    $result = $this->iamClient->createUser([
        'UserName' => $name,
    ]);

    return $result['User'];
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [CreateUser](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [CreateUser](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Creates a user.
#
# @param user_name [String] The name to give the user.
# @return [Aws::IAM::User] The newly created user.
def create_user(user_name)
  user = @iam_resource.create_user(user_name: user_name)
  puts("Created demo user named #{user.name}.")
rescue Aws::Errors::ServiceError => e
  puts("Tried and failed to create demo user.")
  puts("\t#{e.code}: #{e.message}")
  puts("\nCan't continue the demo without a user!")
  raise
else
  user
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [CreateUser](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn create_user(client: &iamClient, user_name: &str) -> Result<User,
iamError> {
    let response = client.create_user().user_name(user_name).send().await?;

    Ok(response.user.unwrap())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [CreateUser](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包创建 IAM 访问密钥

以下代码示例显示如何创建 IAM 访问密钥。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
/// <summary>
/// Create a new AccessKey for the user.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="userName">The name of the user for whom to create the
key.</param>
/// <returns>A new IAM access key for the user.</returns>
public static async Task<AccessKey> CreateAccessKeyAsync(
    AmazonIdentityManagementServiceClient client,
    string userName)
{
    var request = new CreateAccessKeyRequest
    {
        UserName = userName,
    };

    var response = await client.CreateAccessKeyAsync(request);

    if (response.AccessKey is not null)
    {
        Console.WriteLine($"Successfully created Access Key for
{userName}.");
    }

    return response.AccessKey;
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [CreateAccessKey](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
package main

import (
    "context"
    "flag"
    "fmt"

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

// IMACreateAccessKeyAPI defines the interface for the CreateAccessKey function.
// We use this interface to test the function using a mocked service.
```

```

type IAMCreateAccessKeyAPI interface {
    CreateAccessKey(ctx context.Context,
        params *iam.CreateAccessKeyInput,
        optFns ...func(*iam.Options)) (*iam.CreateAccessKeyOutput, error)
}

// MakeAccessKey creates a new AWS Identity and Access Management (IAM) access key
// for a user.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, a CreateAccessKeyOutput object containing the result of the
//     service call and nil.
//     Otherwise, nil and an error from the call to CreateAccessKey.
func MakeAccessKey(c context.Context, api IAMCreateAccessKeyAPI, input
    *iam.CreateAccessKeyInput) (*iam.CreateAccessKeyOutput, error) {
    return api.CreateAccessKey(c, input)
}

func main() {
    userName := flag.String("u", "", "The name of the user")
    flag.Parse()

    if *userName == "" {
        fmt.Println("You must supply a user name (-u USER)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.CreateAccessKeyInput{
        UserName: userName,
    }

    result, err := MakeAccessKey(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error creating a new access key")
        fmt.Println(err)
        return
    }

    fmt.Println("Created new access key with ID: " + *result.AccessKey.AccessKeyId + "
    and secret key: " + *result.AccessKey.SecretAccessKey)
}

```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [CreateAccessKey](#)。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。


```
public static String createIAMAccessKey(IamClient iam,String user) {  
    try {  
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()  
            .userName(user)  
            .build();  
  
        CreateAccessKeyResponse response = iam.createAccessKey(request);  
        return response.accessKey().accessKeyId();  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    return "";  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [CreateAccessKey](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";  
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

创建访问密钥。

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "../libs/iamClient.js";  
import { CreateAccessKeyCommand } from "@aws-sdk/client-iam";  
  
// Set the parameters.  
export const params = {UserName: "IAM_USER_NAME"}; //IAM_USER_NAME  
  
export const run = async () => {  
    try {  
        const data = await iamClient.send(new CreateAccessKeyCommand(params));  
        console.log("Success", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};  
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [CreateAccessKey](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.createAccessKey({UserName: 'IAM_USER_NAME'}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.AccessKey);
  }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [CreateAccessKey](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun createIAMAccessKey(user: String?): String {
    val request = CreateAccessKeyRequest {
        userName = user
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [CreateAccessKey](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name, key_pair.id)
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [CreateAccessKey](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Creates an access key for a user.
#
# @param user [Aws::IAM::User] The user that owns the key.
# @return [Aws::IAM::AccessKeyPair] The newly created access key.
def create_access_key_pair(user)
  user_key = user.create_access_key_pair
  puts("Created access key pair for user.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't create access keys for user #{user.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  user_key
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [CreateAccessKey](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn create_access_key(client: &iamClient, user_name: &str) ->
Result<AccessKey, iamError> {
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<CreateAccessKeyOutput, SdkError<CreateAccessKeyError>> =
loop {
    match client.create_access_key().user_name(user_name).send().await {
        Ok(inner_response) => {
            break Ok(inner_response);
        }
        Err(e) => {
            tries += 1;
            if tries > max_tries {
                break Err(e);
            }
            sleep(Duration::from_secs(2)).await;
        }
    }
}

Ok(response.unwrap().access_key.unwrap())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [CreateAccessKey](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包为 IAM 账户创建别名

以下代码示例显示如何为 IAM 账户创建别名。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
package main

import (
    "context"
    "flag"
    "fmt"

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

// IAMCreateAccountAliasAPI defines the interface for the CreateAccountAlias
function.
// We use this interface to test the function using a mocked service.
type IAMCreateAccountAliasAPI interface {
    CreateAccountAlias(ctx context.Context,
        params *iam.CreateAccountAliasInput,
        optFns ...func(*iam.Options)) (*iam.CreateAccountAliasOutput, error)
```

```

}

// MakeAccountAlias creates an alias for your AWS Identity and Access Management
// (IAM) account.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If successful, a CreateAccountAliasOutput object containing the result of
//   the service call and nil.
//   Otherwise, nil and an error from the call to CreateAccountAlias.
func MakeAccountAlias(c context.Context, api IAMCreateAccountAliasAPI, input
    *iam.CreateAccountAliasInput) (*iam.CreateAccountAliasOutput, error) {
    return api.CreateAccountAlias(c, input)
}

func main() {
    alias := flag.String("a", "", "The account alias")
    flag.Parse()

    if *alias == "" {
        fmt.Println("You must supply an account alias (-a ALIAS)")
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.CreateAccountAliasInput{
        AccountAlias: alias,
    }

    _, err = MakeAccountAlias(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error creating an account alias")
        fmt.Println(err)
        return
    }

    fmt.Printf("Created account alias " + *alias)
}

```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [CreateAccountAlias](#)。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```

public static void createIAMAccountAlias(IamClient iam, String alias) {
    try {
        CreateAccountAliasRequest request = CreateAccountAliasRequest.builder()
            .accountAlias(alias)

```

```
        .build();

        iam.createAccountAlias(request);
        System.out.println("Successfully created account alias: " + alias);

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

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [CreateAccountAlias](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

创建账户别名。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { CreateAccountAliasCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { AccountAlias: "ACCOUNT_ALIAS" }; //ACCOUNT_ALIAS

export const run = async () => {
    try {
        const data = await iamClient.send(new CreateAccountAliasCommand(params));
        console.log("Success", data);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [CreateAccountAlias](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。


```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.createAccountAlias({AccountAlias: process.argv[2]}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [CreateAccountAlias](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun createIAMAccountAlias(alias: String) {

    val request = CreateAccountAliasRequest {
        accountAlias = alias
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [CreateAccountAlias](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of the
```

account ID in the sign-in URL. An account can have only one alias. When a new alias is created, it replaces any existing alias.

```
:param alias: The alias to assign to the account.
"""

try:
    iam.create_account_alias(AccountAlias=alias)
    logger.info("Created an alias '%s' for your account.", alias)
except ClientError:
    logger.exception("Couldn't create alias '%s' for your account.", alias)
    raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [CreateAccountAlias](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包创建用户的内联 IAM policy

以下代码示例演示如何为用户创建内联 IAM policy。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Creates an inline policy for a user that lets the user assume a role.
#
# @param policy_name [String] The name to give the policy.
# @param user [Aws::IAM::User] The user that owns the policy.
# @param role [Aws::IAM::Role] The role that can be assumed.
# @return [Aws::IAM::UserPolicy] The newly created policy.
def create_user_policy(policy_name, user, role)
  policy = user.create_policy(
    policy_name: policy_name,
    policy_document: {
      Version: "2012-10-17",
      Statement: [{
        Effect: "Allow",
        Action: "sts:AssumeRole",
        Resource: role.arn
      }]
    }.to_json)
  puts("Created an inline policy for #{user.name} that lets the user assume role
#{role.name}.")
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't create an inline policy for user #{user.name}. Here's why: ")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    policy
  end
end
```

- 有关 API 的详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [PutUserPolicy](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包删除 IAM policy

以下代码示例显示如何删除 IAM policy。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void deleteIAMPolicy(IamClient iam,String policyARN) {  
    try {  
        DeletePolicyRequest request = DeletePolicyRequest.builder()  
            .policyArn(policyARN)  
            .build();  
  
        iam.deletePolicy(request);  
        System.out.println("Successfully deleted the policy");  
  
    } catch (IamException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
    System.out.println("Done");  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [DeletePolicy](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";  
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

删除策略。

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "../libs/iamClient.js";  
import { DeletePolicyCommand } from "@aws-sdk/client-iam";  
  
// Set the parameters.
```

```
const params = { PolicyArn: "POLICY_ARN"};

const run = async () => {
  try {
    const data = await iamClient.send(new DeletePolicyCommand(params));
    console.log("Success. Policy deleted.", data);

  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [DeletePolicy](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {

    val request = DeletePolicyRequest {
        policyArn = policyARNVal
    }

    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [DeletePolicy](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
```

```
logger.exception("Couldn't delete policy %s.", policy_arn)
raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [DeletePolicy](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Deletes a role. If the role has policies attached, they are detached and
# deleted before the role is deleted.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_role(role)
  role.attached_policies.each do |policy|
    name = policy.policy_name
    policy.detach_role(role_name: role.name)
    policy.delete
    puts("Deleted policy #{name}.")
  end
  name = role.name
  role.delete
  puts("Deleted role #{name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't detach policies and delete role #{role.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [DeletePolicy](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn delete_policy(client: &iamClient, policy: Policy) -> Result<(),
iamError> {
  client
    .delete_policy()
    .policy_arn(policy.arn.unwrap())
    .send()
    .await?;
  Ok(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [DeletePolicy](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包删除 IAM 角色

以下代码示例显示如何删除 IAM 角色。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

删除角色。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { DeleteRoleCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const params = {
  RoleName: "ROLE_NAME"
}

const run = async () => {
  try {
    const data = await iamClient.send(new DeleteRoleCommand(params));
    console.log("Success. Role deleted.", data);
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [DeleteRole](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def delete_role(role_name):
```



```
"""
Deletes a role.

:param role_name: The name of the role to delete.
"""
try:
    iam.Role(role_name).delete()
    logger.info("Deleted role %s.", role_name)
except ClientError:
    logger.exception("Couldn't delete role %s.", role_name)
    raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [DeleteRole](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Deletes a role. If the role has policies attached, they are detached and
# deleted before the role is deleted.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_role(role)
  role.attached_policies.each do |policy|
    name = policy.policy_name
    policy.detach_role(role_name: role.name)
    policy.delete
    puts("Deleted policy #{name}.")
  end
  name = role.name
  role.delete
  puts("Deleted role #{name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't detach policies and delete role #{role.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [DeleteRole](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn delete_role(client: &iamClient, role: &Role) -> Result<(), iamError> {
```

```
let role = role.clone();
loop {
  match client
    .delete_role()
    .role_name(role.role_name.as_ref().unwrap())
    .send()
    .await
  {
    Ok(_) => {
      break;
    }
    Err(_) => {
      sleep(Duration::from_secs(2)).await;
    }
  }
}
Ok(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [DeleteRole](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包删除 IAM 角色策略

以下代码示例显示如何删除 IAM 角色策略。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using System.Threading.Tasks;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

public class DeleteRolePolicy
{
    /// <summary>
    /// Initializes the IAM client object and then calls DeleteRolePolicyAsync
    /// to delete the Policy attached to the Role.
    /// </summary>
    public static async Task Main()
    {
        var client = new AmazonIdentityManagementServiceClient();
        var response = await client.DeleteRolePolicyAsync(new
DeleteRolePolicyRequest
        {
            PolicyName = "ExamplePolicy",
            RoleName = "Test-Role",
        });

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("Policy successfully deleted.");
        }
    }
}
```

```
        else
        {
            Console.WriteLine("Could not delete pollicy.");
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [DeleteRolePolicy](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包删除 IAM 服务器证书

以下代码示例显示如何删除 IAM 服务器证书。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

删除服务器证书。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { DeleteServerCertificateCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { ServerCertificateName: "CERTIFICATE_NAME" }; //
CERTIFICATE_NAME

export const run = async () => {
    try {
        const data = await iamClient.send(
            new DeleteServerCertificateCommand(params)
        );
        console.log("Success", data);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [DeleteServerCertificate](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.deleteServerCertificate({ServerCertificateName: 'CERTIFICATE_NAME'},
function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [DeleteServerCertificate](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包删除 IAM 服务相关角色

以下代码示例显示如何删除 IAM 服务相关角色。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Deletes a service-linked role from the account.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_service_linked_role(role)
  response = @iam_resource.client.delete_service_linked_role(role_name:
role.name)
  task_id = response.deletion_task_id
  while true
    response = @iam_resource.client.get_service_linked_role_deletion_status(
deletion_task_id: task_id)
    status = response.status
    puts("Deletion of #{role.name} #{status}.")
    if %w(SUCCEEDED FAILED).include?(status)
      break
    else

```

```
        sleep(3)
      end
    end
  rescue Aws::Errors::ServiceError => e
    # If AWS has not yet fully propagated the role, it deletes the role but
    # returns NoSuchEntity.
    if e.code != "NoSuchEntity"
      puts("Couldn't delete #{role.name}. Here's why:")
      puts("\t#{e.code}: #{e.message}")
      raise
    end
  end
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [DeleteServiceLinkedRole](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn delete_service_linked_role(
  client: &iamClient,
  role_name: &str,
) -> Result<(), iamError> {
  client
    .delete_service_linked_role()
    .role_name(role_name)
    .send()
    .await?;

  Ok(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [DeleteServiceLinkedRole](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包删除 IAM 用户

以下代码示例显示如何删除 IAM 用户。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```

    /// <summary>
    /// Delete the user, and other resources created for this example.
    /// </summary>
    /// <param name="client">The initialized client object.</param>
    /// <param name="accessKeyId">The Id of the user's access key.</param>
    /// <param name="userName">The user name of the user to delete.</param>
    /// <param name="policyName">The name of the policy to delete.</param>
    /// <param name="policyArn">The Amazon Resource Name ARN of the Policy to
delete.</param>
    /// <param name="roleName">The name of the role that will be deleted.</
param>
    public static async Task DeleteResourcesAsync(
        AmazonIdentityManagementServiceClient client,
        string accessKeyId,
        string userName,
        string policyArn,
        string roleName)
    {
        var detachPolicyResponse = await client.DetachRolePolicyAsync(new
DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        var delPolicyResponse = await client.DeletePolicyAsync(new
DeletePolicyRequest
        {
            PolicyArn = policyArn,
        });

        var delRoleResponse = await client.DeleteRoleAsync(new
DeleteRoleRequest
        {
            RoleName = roleName,
        });

        var delAccessKey = await client.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        var delUserResponse = await client.DeleteUserAsync(new
DeleteUserRequest
        {
            UserName = userName,
        });
    }

```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [DeleteUser](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。


```
// DeleteUser

_, err = service.DeleteUser(context.Background(), &iam.DeleteUserInput{
    UserName: aws.String(ExampleUserName),
})

if err != nil {
    panic("Couldn't delete user: " + err.Error())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [DeleteUser](#)。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void deleteIAMUser(IamClient iam, String userName) {

    try {
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("Successfully deleted IAM user " + userName);

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

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [DeleteUser](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

请删除用户。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { DeleteUserCommand, GetUserCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { UserName: "USER_NAME" }; //USER_NAME

export const run = async () => {
  try {
    const data = await iamClient.send(new GetUserCommand(params));
    return data;
  } catch (err) {
    console.log("Error", err);
  }
  try {
    const results = await iamClient.send(new DeleteUserCommand(params));
    console.log("Success", results);
    return results;
  } catch (err) {
    console.log("Error", err);
  }
  try {
    console.log("User " + "USER_NAME" + " does not exist.");
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [DeleteUser](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  UserName: process.argv[2]
};

iam.getUser(params, function(err, data) {
  if (err && err.code === 'NoSuchEntity') {
    console.log("User " + process.argv[2] + " does not exist.");
  } else {
    iam.deleteUser(params, function(err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [DeleteUser](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun deleteIAMUser(userNameVal: String) {  
  
    val request = DeleteUserRequest {  
        userName = userNameVal  
    }  
  
    // To delete a user, ensure that the user's access keys are deleted first.  
    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        iamClient.deleteUser(request)  
        println("Successfully deleted user $userNameVal")  
    }  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [DeleteUser](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def delete_user(user_name):  
    """  
    Deletes a user. Before a user can be deleted, all associated resources,  
    such as access keys and policies, must be deleted or detached.  
  
    :param user_name: The name of the user.  
    """  
    try:  
        iam.User(user_name).delete()  
        logger.info("Deleted user %s.", user_name)  
    except ClientError:  
        logger.exception("Couldn't delete user %s.", user_name)  
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [DeleteUser](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Deletes a user. If the user has inline policies or access keys, they are
deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
  user.policies.each do |policy|
    name = policy.name
    policy.delete
    puts("Deleted user policy #{name}.")
  end
  user.access_keys.each do |key|
    key.delete
    puts("Deleted access key for user #{user.name}.")
  end
  name = user.name
  user.delete
  puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [DeleteUser](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn delete_user(client: &iamClient, user: &User) -> Result<(),
SdkError<DeleteUserError>> {
  let user = user.clone();
  let mut tries: i32 = 0;
  let max_tries: i32 = 10;

  let response: Result<(), SdkError<DeleteUserError>> = loop {
    match client
      .delete_user()
      .user_name(user.user_name.as_ref().unwrap())
      .send()
      .await
    {
      {
        Ok(_) => {
          break Ok(());
        }
        Err(e) => {
          tries += 1;
          if tries > max_tries {
            break Err(e);
          }
          sleep(Duration::from_secs(2)).await;
        }
      }
    }
  }
}
```

```

    }
  }
};

response
}

```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [DeleteUser](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包删除 IAM 访问密钥

以下代码示例显示如何删除 IAM 访问密钥。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```

/// <summary>
/// Delete the user, and other resources created for this example.
/// </summary>
/// <param name="client">The initialized client object.</param>
/// <param name="accessKeyId">The Id of the user's access key.</param>
/// <param name="userName">The user name of the user to delete.</param>
/// <param name="policyName">The name of the policy to delete.</param>
/// <param name="policyArn">The Amazon Resource Name ARN of the Policy to
delete.</param>
/// <param name="roleName">The name of the role that will be deleted.</
param>
public static async Task DeleteResourcesAsync(
    AmazonIdentityManagementServiceClient client,
    string accessKeyId,
    string userName,
    string policyArn,
    string roleName)
{
    var detachPolicyResponse = await client.DetachRolePolicyAsync(new
DetachRolePolicyRequest
    {
        PolicyArn = policyArn,
        RoleName = roleName,
    });

    var delPolicyResponse = await client.DeletePolicyAsync(new
DeletePolicyRequest
    {
        PolicyArn = policyArn,
    });

    var delRoleResponse = await client.DeleteRoleAsync(new
DeleteRoleRequest
    {
        RoleName = roleName,
    });
}

```

```
        var delAccessKey = await client.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
    {
        AccessKeyId = accessKeyId,
        UserName = userName,
    });

    var delUserResponse = await client.DeleteUserAsync(new
DeleteUserRequest
    {
        UserName = userName,
    });

}
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [DeleteAccessKey](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
package main

import (
    "context"
    "flag"
    "fmt"

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

// IAMDeleteAccessKeyAPI defines the interface for the DeleteAccessKey function.
// We use this interface to test the function using a mocked service.
type IAMDeleteAccessKeyAPI interface {
    DeleteAccessKey(ctx context.Context,
        params *iam.DeleteAccessKeyInput,
        optFns ...func(*iam.Options)) (*iam.DeleteAccessKeyOutput, error)
}

// RemoveAccessKey deletes an AWS Identity and Access Management (IAM) access key.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If successful, a DeleteAccessKeyOutput object containing the result of the
//   service call and nil.
//   Otherwise, nil and an error from the call to DeleteAccessKey.
func RemoveAccessKey(c context.Context, api IAMDeleteAccessKeyAPI, input
    *iam.DeleteAccessKeyInput) (*iam.DeleteAccessKeyOutput, error) {
    return api.DeleteAccessKey(c, input)
}

func main() {
    keyID := flag.String("k", "", "The ID of the access key")
```



```

userName := flag.String("u", "", "The name of the user")
flag.Parse()

if *keyID == "" || *userName == "" {
    fmt.Println("You must supply the key ID and user name (-k KEY-ID -u USER-NAME)")
    return
}

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic("configuration error, " + err.Error())
}

client := iam.NewFromConfig(cfg)

input := &iam.DeleteAccessKeyInput{
    AccessKeyId: keyID,
    UserName:    userName,
}

_, err = RemoveAccessKey(context.TODO(), client, input)
if err != nil {
    fmt.Println("Error", err)
    return
}

fmt.Println("Deleted key with ID " + *keyID + " from user " + *userName)
}

```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [DeleteAccessKey](#)。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```

public static void deleteKey(IamClient iam, String username, String accessKey) {
    try {
        DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
            .accessKeyId(accessKey)
            .userName(username)
            .build();

        iam.deleteAccessKey(request);
        System.out.println("Successfully deleted access key " + accessKey +
            " from user " + username);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [DeleteAccessKey](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

删除访问密钥。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { DeleteAccessKeyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  AccessKeyId: "ACCESS_KEY_ID", // ACCESS_KEY_ID
  UserName: "USER_NAME", // USER_NAME
};

export const run = async () => {
  try {
    const data = await iamClient.send(new DeleteAccessKeyCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [DeleteAccessKey](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  AccessKeyId: 'ACCESS_KEY_ID',
  UserName: 'USER_NAME'
}
```

```
};  
  
iam.deleteAccessKey(params, function(err, data) {  
  if (err) {  
    console.log("Error", err);  
  } else {  
    console.log("Success", data);  
  }  
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [DeleteAccessKey](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun deleteKey(userNameVal: String, accessKey: String) {  
  
    val request = DeleteAccessKeyRequest {  
        accessKeyId = accessKey  
        userName = userNameVal  
    }  
  
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        iamClient.deleteAccessKey(request)  
        println("Successfully deleted access key $accessKey from $userNameVal")  
    }  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [DeleteAccessKey](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def delete_key(user_name, key_id):  
    """  
    Deletes a user's access key.  
  
    :param user_name: The user that owns the key.  
    :param key_id: The ID of the key to delete.  
    """  
  
    try:
```

```
key = iam.AccessKey(user_name, key_id)
key.delete()
logger.info(
    "Deleted access key %s for %s.", key.id, key.user_name)
except ClientError:
    logger.exception("Couldn't delete key %s for %s", key_id, user_name)
    raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [DeleteAccessKey](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Deletes a user. If the user has inline policies or access keys, they are
deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
  user.policies.each do |policy|
    name = policy.name
    policy.delete
    puts("Deleted user policy #{name}.")
  end
  user.access_keys.each do |key|
    key.delete
    puts("Deleted access key for user #{user.name}.")
  end
  name = user.name
  user.delete
  puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [DeleteAccessKey](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn delete_access_key(
```

```

    client: &iamClient,
    user: &User,
    key: &AccessKey,
) -> Result<(), iamError> {
    loop {
        match client
        .delete_access_key()
        .user_name(user.user_name.as_ref().unwrap())
        .access_key_id(key.access_key_id.as_ref().unwrap())
        .send()
        .await
        {
            Ok(_) => {
                break;
            }
            Err(e) => {
                println!("Can't delete the access key: {:?}", e);
                sleep(Duration::from_secs(2)).await;
            }
        }
    }
    Ok(())
}

```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [DeleteAccessKey](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包删除 IAM 账户别名

以下代码示例显示如何删除 IAM 账户别名。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```

public static void deleteIAMAccountAlias(IamClient iam, String alias ) {

    try {
        DeleteAccountAliasRequest request = DeleteAccountAliasRequest.builder()
            .accountAlias(alias)
            .build();

        iam.deleteAccountAlias(request);
        System.out.println("Successfully deleted account alias " + alias);

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    System.out.println("Done");
}

```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [DeleteAccountAlias](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

删除账户别名。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { DeleteAccountAliasCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { AccountAlias: "ALIAS" }; // ALIAS

export const run = async () => {
  try {
    const data = await iamClient.send(new DeleteAccountAliasCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [DeleteAccountAlias](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.deleteAccountAlias({AccountAlias: process.argv[2]}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

```
}  
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [DeleteAccountAlias](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun deleteIAMAccountAlias(alias: String) {  
  
    val request = DeleteAccountAliasRequest {  
        accountAlias = alias  
    }  
  
    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        iamClient.deleteAccountAlias(request)  
        println("Successfully deleted account alias $alias")  
    }  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [DeleteAccountAlias](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def delete_alias(alias):  
    """  
    Removes the alias from the current account.  
  
    :param alias: The alias to remove.  
    """  
    try:  
        iam.meta.client.delete_account_alias(AccountAlias=alias)  
        logger.info("Removed alias '%s' from your account.", alias)  
    except ClientError:  
        logger.exception("Couldn't remove alias '%s' from your account.", alias)  
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [DeleteAccountAlias](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包删除用户的内联 IAM policy

以下代码示例显示如何删除用户的内联 IAM policy。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Deletes a user. If the user has inline policies or access keys, they are
deleted
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
  user.policies.each do |policy|
    name = policy.name
    policy.delete
    puts("Deleted user policy #{name}.")
  end
  user.access_keys.each do |key|
    key.delete
    puts("Deleted access key for user #{user.name}.")
  end
  name = user.name
  user.delete
  puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [DeleteUserPolicy](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn delete_user_policy(
    client: &iamClient,
    user: &User,
    policy_name: &str,
) -> Result<(), SdkError<DeleteUserPolicyError>> {
```

```
client
    .delete_user_policy()
    .user_name(user.user_name.as_ref().unwrap())
    .policy_name(policy_name)
    .send()
    .await?;

Ok(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [DeleteUserPolicy](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包从角色分离 IAM policy

以下代码示例显示如何从角色分离 IAM policy。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
/// <summary>
/// Delete the user, and other resources created for this example.
/// </summary>
/// <param name="client">The initialized client object.</param>
/// <param name="accessKeyId">The Id of the user's access key.</param>
/// <param name="userName">The user name of the user to delete.</param>
/// <param name="policyName">The name of the policy to delete.</param>
/// <param name="policyArn">The Amazon Resource Name ARN of the Policy to
delete.</param>
/// <param name="roleName">The name of the role that will be deleted.</
param>
public static async Task DeleteResourcesAsync(
    AmazonIdentityManagementServiceClient client,
    string accessKeyId,
    string userName,
    string policyArn,
    string roleName)
{
    var detachPolicyResponse = await client.DetachRolePolicyAsync(new
DetachRolePolicyRequest
    {
        PolicyArn = policyArn,
        RoleName = roleName,
    });

    var delPolicyResponse = await client.DeletePolicyAsync(new
DeletePolicyRequest
    {
        PolicyArn = policyArn,
    });

    var delRoleResponse = await client.DeleteRoleAsync(new
DeleteRoleRequest
```

```
        {
            RoleName = roleName,
        });

        var delAccessKey = await client.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        var delUserResponse = await client.DeleteUserAsync(new
DeleteUserRequest
        {
            UserName = userName,
        });
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [DetachRolePolicy](#)。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void detachPolicy(IamClient iam, String roleName, String
policyArn ) {

    try {
        DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.detachRolePolicy(request);
        System.out.println("Successfully detached policy " + policyArn +
            " from role " + roleName);

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

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [DetachRolePolicy](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

分离策略。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import {
  ListAttachedRolePoliciesCommand,
  DetachRolePolicyCommand,
} from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { RoleName: "ROLE_NAME" }; //ROLE_NAME

export const run = async () => {
  try {
    const data = await iamClient.send(
      new ListAttachedRolePoliciesCommand(params)
    );
    const myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (_val, index) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        try {
          await iamClient.send(
            new DetachRolePolicyCommand(paramsRoleList)
          );
          console.log("Policy detached from role successfully");
          process.exit();
        } catch (err) {
          console.log("Unable to detach policy from role", err);
        }
      } else {
      }
    });
    return data;
  } catch (err) {
    console.log("User " + "USER_NAME" + " does not exist.");
  }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [DetachRolePolicy](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
```

```
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var paramsRoleList = {
  RoleName: process.argv[2]
};

iam.listAttachedRolePolicies(paramsRoleList, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === 'AmazonDynamoDBFullAccess') {
        var params = {
          PolicyArn: 'arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess',
          RoleName: process.argv[2]
        };
        iam.detachRolePolicy(params, function(err, data) {
          if (err) {
            console.log("Unable to detach policy from role", err);
          } else {
            console.log("Policy detached from role successfully");
            process.exit();
          }
        });
      }
    });
  }
});
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [DetachRolePolicy](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun detachPolicy(roleNameVal: String, policyArnVal: String) {

    val request = DetachRolePolicyRequest {
        roleName = roleNameVal
        policyArn = policyArnVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role $roleNameVal")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [DetachRolePolicy](#)

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

使用 Boto3 策略对象从角色分离策略。

```
def detach_from_role(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).detach_role(RoleName=role_name)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name)
        raise
```

使用 Boto3 角色对象从角色分离策略。

```
def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [DetachRolePolicy](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Deletes a role. If the role has policies attached, they are detached and
```

```
# deleted before the role is deleted.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_role(role)
  role.attached_policies.each do |policy|
    name = policy.policy_name
    policy.detach_role(role_name: role.name)
    policy.delete
    puts("Deleted policy #{name}.")
  end
  name = role.name
  role.delete
  puts("Deleted role #{name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't detach policies and delete role #{role.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [DetachRolePolicy](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn detach_role_policy(
  client: &iamClient,
  role_name: &str,
  policy_arn: &str,
) -> Result<(), iamError> {
  client
    .detach_role_policy()
    .role_name(role_name)
    .policy_arn(policy_arn)
    .send()
    .await?;

  Ok(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [DetachRolePolicy](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包从用户分离 IAM policy

以下代码示例显示如何从用户分离 IAM policy。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
package main

import (
    "context"
    "flag"
    "fmt"

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

// IAMDetachRolePolicyAPI defines the interface for the DetachRolePolicy function.
// We use this interface to test the function using a mocked service.
type IAMDetachRolePolicyAPI interface {
    DetachRolePolicy(ctx context.Context,
        params *iam.DetachRolePolicyInput,
        optFns ...func(*iam.Options)) (*iam.DetachRolePolicyOutput, error)
}

// DetachDynamoFullPolicy detaches an Amazon DynamoDB full-access policy from an
// AWS Identity and Access Management (IAM) role.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If successful, a DetachRolePolicyOutput object containing the result of the
//   service call and nil.
//   Otherwise, nil and an error from the call to DetachRolePolicy.
func DetachDynamoFullPolicy(c context.Context, api IAMDetachRolePolicyAPI, input
    *iam.DetachRolePolicyInput) (*iam.DetachRolePolicyOutput, error) {
    return api.DetachRolePolicy(c, input)
}

func main() {
    roleName := flag.String("r", "", "The name of the IAM role")
    flag.Parse()

    if *roleName == "" {
        fmt.Println("You must supply a role name (-r ROLE)")
        return
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    policyArn := "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess"
    input := &iam.DetachRolePolicyInput{
        PolicyArn: &policyArn,
        RoleName:  roleName,
    }
}
```

```
_, err = DetachDynamoFullPolicy(context.TODO(), client, input)
if err != nil {
    fmt.Println("Unable to detach DynamoDB full-access role policy from role")
    return
}
fmt.Println("Role detached successfully")
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [DetachUserPolicy](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [DetachUserPolicy](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn detach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
```

```
client
    .detach_user_policy()
    .user_name(user_name)
    .policy_arn(policy_arn)
    .send()
    .await?;

Ok(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 DetachUserPolicy。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包从 IAM 生成凭证报告

以下代码示例说明如何从 IAM 生成当前账户的凭证报告。生成报告后，使用 GetCredentialReport 操作获取报告。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info("Generating credentials report for your account. "
                    "Current state is %s.", response['State'])
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your
account.")
        raise
    else:
        return response
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [GenerateCredentialReport](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包从 IAM 获取凭证报告

以下代码示例显示如何获取 IAM 生成的最新凭证报告。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def get_credential_report():
    """
    Gets the most recently generated credentials report about the current account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response['Content'])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response['Content']
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [GetCredentialReport](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包获取您的账户的详细 IAM 授权报告

以下代码示例显示如何获取您账户的详细 IAM 授权报告。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report, such
                           as users or roles. When not specified, all resources
                           are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [GetAccountAuthorizationDetails](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包获取 IAM policy

以下代码示例显示如何获取 IAM policy。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();
var request = new GetPolicyRequest
{
    PolicyArn = "POLICY_ARN",
};

var response = await client.GetPolicyAsync(request);

Console.WriteLine($"{response.Policy.PolicyName} was created on ");
Console.WriteLine($"{response.Policy.CreateDate}");
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [GetPolicy](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// GetPolicy

getPolicyResponse, err := service.GetPolicy(context.Background(),
&iam.GetPolicyInput{
    PolicyArn: policyArn,
})

if err != nil {
    panic("Couldn't get policy from ARN: " + err.Error())
}
```

```
fmt.Printf("policy: %s, name %s\n",
    *getPolicyResponse.Policy.Arn,
    *getPolicyResponse.Policy.PolicyName)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [GetPolicy](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

获取策略。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { GetPolicyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const params = {
  PolicyArn: "POLICY_ARN" /* required */,
};

const run = async () => {
  try {
    const data = await iamClient.send(new GetPolicyCommand(params));
    console.log("Success", data.Policy);
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [GetPolicy](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});
```

```
// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  PolicyArn: 'arn:aws:iam::aws:policy/AWSLambdaExecute'
};

iam.getPolicy(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Policy.Description);
  }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [GetPolicy](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun getIAMPolicy(policyArnVal: String?) {

    val request = GetPolicyRequest {
        policyArn = policyArnVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [GetPolicy](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
$uuid = uniqid();
$service = new IamService();

public function getPolicy($policyArn)
{
```



```
        return $this->customWaiter(function () use ($policyArn) {
            return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
        });
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [GetPolicy](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get('Statement', None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
            policy_arn)
        raise
    else:
        return policy_statement
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [GetPolicy](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Gets data about a policy.
#
# @param policy_arn [String] The ARN of the policy to look up.
# @return [Aws::IAM::Policy] The retrieved policy.
def get_policy(policy_arn)
    policy = @iam_resource.policy(policy_arn)
    puts("Got policy '#{policy.policy_name}'. Its ID is: #{policy.policy_id}.")
    rescue Aws::Errors::ServiceError => e
        puts("Couldn't get policy '#{policy_arn}'. Here's why:")
        puts("\t#{e.code}: #{e.message}")
        raise
    else
    end
end
```

```
    policy
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [GetPolicy](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包获取 IAM policy 版本

以下代码示例显示如何获取 IAM policy 版本。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get('Statement', None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
            policy_arn)
        raise
    else:
        return policy_statement
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [GetPolicyVersion](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包获取 IAM 角色

以下代码示例显示如何获取 IAM 角色。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

var response = await client.GetRoleAsync(new GetRoleRequest
{
    RoleName = "LambdaS3Role",
});

if (response.Role is not null)
{
    Console.WriteLine($"{response.Role.RoleName} with ARN: {response.Role.Arn}");
    Console.WriteLine($"{response.Role.Description}");
    Console.WriteLine($"Created: {response.Role.CreateDate} Last used on:
    { response.Role.RoleLastUsed}");
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [GetRole](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// GetRole

getRoleResult, err := service.GetRole(context.Background(), &iam.GetRoleInput{
    RoleName: aws.String(ExampleRoleName),
})

if err != nil {
    panic("Couldn't get role! " + err.Error())
}

fmt.Println("## GetRole results: ")
fmt.Println("ARN: ", *getRoleResult.Role.Arn)
fmt.Println("Name: ", *getRoleResult.Role.RoleName)
fmt.Println("Created On: ", *getRoleResult.Role.CreateDate)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [GetRole](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

获取角色。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { GetRoleCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const params = {
  RoleName: "ROLE_NAME" /* required */
};

const run = async () => {
  try {
    const data = await iamClient.send(new GetRoleCommand(params));
    console.log("Success", data.Role);
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [GetRole](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
$uuid = uniqid();
$service = new IamService();

public function getRole($roleName)
{
    return $this->customWaiter(function () use ($roleName) {
        return $this->iamClient->getRole(['RoleName' => $roleName]);
    });
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [GetRole](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def get_role(role_name):
    """
    Gets a role by name.

    :param role_name: The name of the role to retrieve.
    :return: The specified role.
    """
    try:
        role = iam.Role(role_name)
        role.load() # calls GetRole to load attributes
        logger.info("Got role with arn %s.", role.arn)
    except ClientError:
        logger.exception("Couldn't get role named %s.", role_name)
        raise
    else:
        return role
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [GetRole](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Gets data about a role.
#
# @param name [String] The name of the role to look up.
# @return [Aws::IAM::Role] The retrieved role.
def get_role(name)
    role = @iam_resource.role(name)
    puts("Got data for role '#{role.name}'. Its ARN is '#{role.arn}'.")
    rescue Aws::Errors::ServiceError => e
        puts("Couldn't get data for role '#{name}' Here's why:")
        puts("\t#{e.code}: #{e.message}")
        raise
    else
        role
    end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [GetRole](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn get_role(  
    client: &iamClient,  
    role_name: String,  
) -> Result<GetRoleOutput, SdkError<GetRoleError>> {  
    let response = client.get_role().role_name(role_name).send().await?;  
    Ok(response)  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [GetRole](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用](#) (p. 15)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包获取 IAM 服务器证书

以下代码示例显示如何获取 IAM 服务器证书。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";  
// Set the AWS Region.  
const REGION = "REGION"; // For example, "us-east-1".  
// Create an IAM service client object.  
const iamClient = new IAMClient({ region: REGION });  
export { iamClient };
```

获取服务器证书。

```
// Import required AWS SDK clients and commands for Node.js.  
import { iamClient } from "../libs/iamClient.js";  
import { GetServerCertificateCommand } from "@aws-sdk/client-iam";  
  
// Set the parameters.  
export const params = { ServerCertificateName: "CERTIFICATE_NAME" }; //  
CERTIFICATE_NAME  
  
export const run = async () => {  
    try {  
        const data = await iamClient.send(new GetServerCertificateCommand(params));  
        console.log("Success", data);  
        return data;  
    } catch (err) {  
        console.log("Error", err);  
    }  
};
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 `GetServerCertificate`。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.getServerCertificate({ServerCertificateName: 'CERTIFICATE_NAME'}, function(err,
data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 `GetServerCertificate`。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包从 IAM 获取账户使用情况摘要

以下代码示例显示如何从 IAM 获取账户使用情况摘要。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
```



```
return summary.summary_map
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [GetAccountSummary](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包获取有关上次使用 IAM 访问密钥的数据

以下代码示例显示如何获取关于上次使用 IAM 访问密钥的数据。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
package main

import (
    "context"
    "flag"
    "fmt"

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

// IAMGetAccessKeyLastUsedAPI defines the interface for the GetAccessKeyLastUsed
// function.
// We use this interface to test the function using a mocked service.
type IAMGetAccessKeyLastUsedAPI interface {
    GetAccessKeyLastUsed(ctx context.Context,
        params *iam.GetAccessKeyLastUsedInput,
        optFns ...func(*iam.Options)) (*iam.GetAccessKeyLastUsedOutput, error)
}

// WhenWasKeyUsed retrieves when an AWS Identity and Access Management (IAM) access
// key was last used, including the AWS Region and with which service.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If successful, a GetAccessKeyLastUsedOutput object containing the result of
//   the service call and nil.
//   Otherwise, nil and an error from the call to GetAccessKeyLastUsed.
func WhenWasKeyUsed(c context.Context, api IAMGetAccessKeyLastUsedAPI, input
    *iam.GetAccessKeyLastUsedInput) (*iam.GetAccessKeyLastUsedOutput, error) {
    return api.GetAccessKeyLastUsed(c, input)
}

func main() {
    keyID := flag.String("k", "", "The ID of the access key")
    flag.Parse()

    if *keyID == "" {
```

```
    fmt.Println("You must supply the ID of an access key (-k KEY-ID)")
    return
}

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic("configuration error, " + err.Error())
}

client := iam.NewFromConfig(cfg)

input := &iam.GetAccessKeyLastUsedInput{
    AccessKeyId: keyID,
}

result, err := WhenWasKeyUsed(context.TODO(), client, input)
if err != nil {
    fmt.Println("Got an error retrieving when access key was last used:")
    fmt.Println(err)
    return
}

fmt.Println("The key was last used:", *result.AccessKeyLastUsed)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [GetAccessKeyLastUsed](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

获取访问密钥。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { GetAccessKeyLastUsedCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { AccessKeyId: "ACCESS_KEY_ID" }; //ACCESS_KEY_ID

export const run = async () => {
    try {
        const data = await iamClient.send(new GetAccessKeyLastUsedCommand(params));
        console.log("Success", data);
        return data;
    } catch (err) {
```

```
        console.log("Error", err);
    }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [GetAccessKeyLastUsed](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.getAccessKeyLastUsed({AccessKeyId: 'ACCESS_KEY_ID'}, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        console.log("Success", data.AccessKeyLastUsed);
    }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [GetAccessKeyLastUsed](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response['AccessKeyLastUsed'].get('LastUsedDate', None)
        last_service = response['AccessKeyLastUsed'].get('ServiceName', None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.", key_id,
            response['UserName'], last_used_date, last_service)
```

```
except ClientError:
    logger.exception("Couldn't get last use of key %s.", key_id)
    raise
else:
    return response
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [GetAccessKeyLastUsed](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包获取 IAM 账户密码策略

以下代码示例显示如何获取 IAM 账户密码策略。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

try
{
    var request = new GetAccountPasswordPolicyRequest();
    var response = await client.GetAccountPasswordPolicyAsync(request);

    Console.WriteLine($"{response.PasswordPolicy}");
}
catch (NoSuchEntityException ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [GetAccountPasswordPolicy](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// GetAccountPasswordPolicy
```

```
fmt.Println("## GetAccountPasswordPolicy")

accountPasswordPolicy, err :=
service.GetAccountPasswordPolicy(context.Background(),
&iam.GetAccountPasswordPolicyInput{})

if err != nil {
    var notexists *types.NoSuchEntityException
    if errors.As(err, &notexists) {
        fmt.Println("No password policy")
    } else {
        panic("Couldn't get account password policy! " + err.Error())
    }
} else {
    fmt.Println("Users can change password: ",
accountPasswordPolicy.PasswordPolicy.AllowUsersToChangePassword)
    fmt.Println("Passwords expire: ",
accountPasswordPolicy.PasswordPolicy.ExpirePasswords)
    fmt.Println("Minimum password length: ",
accountPasswordPolicy.PasswordPolicy.MinimumPasswordLength)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [GetAccountPasswordPolicy](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

获取账户密码策略。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { GetAccountPasswordPolicyCommand } from "@aws-sdk/client-iam";

const run = async () => {
    try {
        const data = await iamClient.send(new GetAccountPasswordPolicyCommand({}));
        console.log("Success", data.PasswordPolicy);
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [GetAccountPasswordPolicy](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
$uuid = uniqid();
$service = new IamService();

public function getAccountPasswordPolicy()
{
    return $this->iamClient->getAccountPasswordPolicy();
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [GetAccountPasswordPolicy](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def print_password_policy():
    """
    Prints the password policy for the account.
    """
    try:
        pw_policy = iam.AccountPasswordPolicy()
        print("Current account password policy:")
        print(f"\tallow_users_to_change_password:
{pw_policy.allow_users_to_change_password}")
        print(f"\texpire_passwords: {pw_policy.expire_passwords}")
        print(f"\thard_expiry: {pw_policy.hard_expiry}")
        print(f"\tmax_password_age: {pw_policy.max_password_age}")
        print(f"\tminimum_password_length: {pw_policy.minimum_password_length}")
        print(f"\tpassword_reuse_prevention:
{pw_policy.password_reuse_prevention}")
        print(f"\trequire_lowercase_characters:
{pw_policy.require_lowercase_characters}")
        print(f"\trequire_numbers: {pw_policy.require_numbers}")
        print(f"\trequire_symbols: {pw_policy.require_symbols}")
        print(f"\trequire_uppercase_characters:
{pw_policy.require_uppercase_characters}")
        printed = True
    except ClientError as error:
        if error.response['Error']['Code'] == 'NoSuchEntity':
            print("The account does not have a password policy set.")
        else:
            logger.exception("Couldn't get account password policy.")
            raise
```

```
else:
    return printed
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [GetAccountPasswordPolicy](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Prints the password policy for the account.
def print_account_password_policy
  policy = @iam_resource.account_password_policy
  policy.load
  puts("The account password policy is:")
  puts(policy.data.to_h)
rescue Aws::Errors::ServiceError => e
  if e.code == "NoSuchEntity"
    puts("The account does not have a password policy.")
  else
    puts("Couldn't print the account password policy. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [GetAccountPasswordPolicy](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn get_account_password_policy(
    client: &iamClient,
) -> Result<GetAccountPasswordPolicyOutput,
    SdkError<GetAccountPasswordPolicyError>> {
    let response = client.get_account_password_policy().send().await?;

    Ok(response)
}
```


- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [GetAccountPasswordPolicy](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用](#) (p. 15)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包列出 IAM 的 SAML 提供商

以下代码示例展示如何列出 IAM 的 SAML 提供商。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

var response = await client.ListSAMLProvidersAsync(new ListSAMLProvidersRequest());

response.SAMLProviderList.ForEach(samlProvider =>
{
    Console.WriteLine($"{samlProvider.Arn} created on: {samlProvider.CreateDate}");
});
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [ListSAMLProviders](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// ListSAMLProviders

samlProviderList, err := service.ListSAMLProviders(context.Background(),
&iam.ListSAMLProvidersInput{})

if err != nil {
    panic("Couldn't list saml providers: " + err.Error())
}

for _, provider := range samlProviderList.SAMLProviderList {
    fmt.Printf("%s %s -> %s", *provider.Arn, *provider.CreateDate,
*provider.ValidUntil)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [ListSAMLProviders](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

列出 SAML 提供商。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { ListSAMLProvidersCommand } from "@aws-sdk/client-iam";

export const run = async () => {
  try {
    const results = await iamClient.send(new ListSAMLProvidersCommand({}));
    console.log("Success", results);
    return results;
  } catch (err) {
    console.log("Error", err);
  }
}
run();
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [ListSAMLProviders](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
$uuid = uniqid();
$service = new IamService();

public function listSAMLProviders()
{
    return $this->iamClient->listSAMLProviders();
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [ListSAMLProviders](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def list_saml_providers(count):
    """
    Lists the SAML providers for the account.

    :param count: The maximum number of providers to list.
    """
    try:
        found = 0
        for provider in iam.saml_providers.limit(count):
            logger.info('Got SAML provider %s.', provider.arn)
            found += 1
        if found == 0:
            logger.info("Your account has no SAML providers.")
    except ClientError:
        logger.exception("Couldn't list SAML providers.")
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [ListSAMLProviders](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Lists up to a specified number of SAML providers for the account.
#
# @param count [Integer] The maximum number of providers to list.
def list_saml_providers(count)
  @iam_resource.saml_providers.limit(count).each do |provider|
    puts("\t#{provider.arn}")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't list SAML providers. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [ListSAMLProviders](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn list_saml_providers(  
    client: &Client,  
) -> Result<ListSamlProvidersOutput, SdkError<ListSAMLProvidersError>> {  
    let response = client.list_saml_providers().send().await?;  
  
    Ok(response)  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [ListSAMLProviders](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包列出用户的 IAM 访问密钥

以下代码示例显示如何列出用户的 IAM 访问密钥。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
package main  
  
import (  
    "context"  
    "flag"  
    "fmt"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/config"  
    "github.com/aws/aws-sdk-go-v2/service/iam"  
)  
  
// IAMListAccessKeysAPI defines the interface for the ListAccessKeys function.  
// We use this interface to test the function using a mocked service.  
type IAMListAccessKeysAPI interface {  
    ListAccessKeys(ctx context.Context,  
        params *iam.ListAccessKeysInput,  
        optFns ...func(*iam.Options)) (*iam.ListAccessKeysOutput, error)  
}  
  
// GetAccessKeys retrieves up to the AWS Identity and Access Management (IAM)  
// access keys for a user.  
// Inputs:  
//     c is the context of the method call, which includes the AWS Region.  
//     api is the interface that defines the method call.  
//     input defines the input arguments to the service call.  
// Output:  
//     If successful, a ListAccessKeysOutput object containing the result of the  
//     service call and nil.
```

```
// Otherwise, nil and an error from the call to ListAccessKeys.
func GetAccessKeys(c context.Context, api IAMListAccessKeysAPI, input
    *iam.ListAccessKeysInput) (*iam.ListAccessKeysOutput, error) {
    return api.ListAccessKeys(c, input)
}

func main() {
    maxItems := flag.Int("m", 10, "The maximum number of access keys to show")
    userName := flag.String("u", "", "The name of the user")
    flag.Parse()

    if *userName == "" {
        fmt.Println("You must supply the name of a user (-u USER)")
        return
    }

    if *maxItems < 0 {
        *maxItems = 10
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.ListAccessKeysInput{
        MaxItems: aws.Int32(int32(*maxItems)),
        UserName: userName,
    }

    result, err := GetAccessKeys(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error retrieving user access keys:")
        fmt.Println(err)
        return
    }

    for _, key := range result.AccessKeyMetadata {
        fmt.Println("Status for access key " + *key.AccessKeyId + ": " +
            string(key.Status))
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [ListAccessKeys](#)。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void listKeys( IamClient iam,String userName ){

    try {
        boolean done = false;
        String newMarker = null;
```

```
while (!done) {
    ListAccessKeysResponse response;

    if(newMarker == null) {
        ListAccessKeysRequest request = ListAccessKeysRequest.builder()
            .userName(userName)
            .build();

        response = iam.listAccessKeys(request);
    } else {
        ListAccessKeysRequest request = ListAccessKeysRequest.builder()
            .userName(userName)
            .marker(newMarker)
            .build();

        response = iam.listAccessKeys(request);
    }

    for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
        System.out.format("Retrieved access key %s",
            metadata.accessKeyId());
    }

    if (!response.isTruncated()) {
        done = true;
    } else {
        newMarker = response.marker();
    }
}

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

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [ListAccessKeys](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

列出访问密钥。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
```

```
import { ListAccessKeysCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  MaxItems: 5,
  UserName: "IAM_USER_NAME", //IAM_USER_NAME
};

export const run = async () => {
  try {
    const data = await iamClient.send(new ListAccessKeysCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [ListAccessKeys](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  MaxItems: 5,
  UserName: 'IAM_USER_NAME'
};

iam.listAccessKeys(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [ListAccessKeys](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun listKeys(userNameVal: String?) {  
  
    val request = ListAccessKeysRequest {  
        userName = userNameVal  
    }  
    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        val response = iamClient.listAccessKeys(request)  
        response.accessKeyMetadata?.forEach { md ->  
            println("Retrieved access key ${md.accessKeyId}")  
        }  
    }  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [ListAccessKeys](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def list_keys(user_name):  
    """  
    Lists the keys owned by the specified user.  
  
    :param user_name: The name of the user.  
    :return: The list of keys owned by the user.  
    """  
    try:  
        keys = list(iam.User(user_name).access_keys.all())  
        logger.info("Got %s access keys for %s.", len(keys), user_name)  
    except ClientError:  
        logger.exception("Couldn't get access keys for %s.", user_name)  
        raise  
    else:  
        return keys
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [ListAccessKeys](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Deletes a user. If the user has inline policies or access keys, they are  
deleted
```

```
# before the user is deleted.
#
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
  user.policies.each do |policy|
    name = policy.name
    policy.delete
    puts("Deleted user policy #{name}.")
  end
  user.access_keys.each do |key|
    key.delete
    puts("Deleted access key for user #{user.name}.")
  end
  name = user.name
  user.delete
  puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [ListAccessKeys](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包列出 IAM 账户别名

以下代码示例展示如何列出 IAM 账户别名。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
package main

import (
    "context"
    "flag"
    "fmt"

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

// IAMListAccountAliasesAPI defines the interface for the ListAccountAliases
function.
// We use this interface to test the function using a mocked service.
type IAMListAccountAliasesAPI interface {
    ListAccountAliases(ctx context.Context,
        params *iam.ListAccountAliasesInput,
        optFns ...func(*iam.Options)) (*iam.ListAccountAliasesOutput, error)
}

// GetAccountAliases retrieves the aliases for your AWS Identity and Access
Management (IAM) account.
```

```
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If successful, a ListAccountAliasesOutput object containing the result of
//   the service call and nil.
//   Otherwise, nil and an error from the call to ListAccountAliases.
func GetAccountAliases(c context.Context, api IAMListAccountAliasesAPI, input
    *iam.ListAccountAliasesInput) (*iam.ListAccountAliasesOutput, error) {
    return api.ListAccountAliases(c, input)
}

func main() {
    maxItems := flag.Int("m", 10, "Maximum number of aliases to list")
    flag.Parse()

    if *maxItems < 0 {
        *maxItems = 10
    }

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error, " + err.Error())
    }

    client := iam.NewFromConfig(cfg)

    input := &iam.ListAccountAliasesInput{
        MaxItems: aws.Int32(int32(*maxItems)),
    }

    result, err := GetAccountAliases(context.TODO(), client, input)
    if err != nil {
        fmt.Println("Got an error retrieving account aliases")
        fmt.Println(err)
        return
    }

    for i, alias := range result.AccountAliases {
        fmt.Printf("Alias %d: %s\n", i, alias)
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [ListAccountAliases](#)。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void listAliases(IamClient iam) {
    try {
        ListAccountAliasesResponse response = iam.listAccountAliases();
        for (String alias : response.accountAliases()) {
            System.out.printf("Retrieved account alias %s", alias);
        }
    }
}
```

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

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [ListAccountAliases](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

列出账户别名。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { ListAccountAliasesCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { MaxItems: 5 };

export const run = async () => {
    try {
        const data = await iamClient.send(new ListAccountAliasesCommand(params));
        console.log("Success", data);
        return data;
    } catch (err) {
        console.log("Error", err);
    }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [ListAccountAliases](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
```

```
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.listAccountAliases({MaxItems: 10}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [ListAccountAliases](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun listAliases() {

    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [ListAccountAliases](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
    alias.

    :return: The list of aliases for the account.
    """
    try:
```

```
response = iam.meta.client.list_account_aliases()
aliases = response['AccountAliases']
if len(aliases) > 0:
    logger.info("Got aliases for your account: %s.", ','.join(aliases))
else:
    logger.info("Got no aliases for your account.")
except ClientError:
    logger.exception("Couldn't list aliases for your account.")
    raise
else:
    return response['AccountAliases']
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [ListAccountAliases](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包列出 IAM 组

以下代码示例显示如何列出 IAM 组。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

var request = new ListGroupsRequest
{
    MaxItems = 10,
};

var response = await client.ListGroupsAsync(request);

do
{
    response.Groups.ForEach(group =>
    {
        Console.WriteLine($"{group.GroupName} created on: {group.CreateDate}");
    });

    if (response.IsTruncated)
    {
        request.Marker = response.Marker;
        response = await client.ListGroupsAsync(request);
    }
} while (response.IsTruncated);
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [ListGroups](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// ListGroups

listGroupsResult, err := service.ListGroups(context.Background(),
&iam.ListGroupsInput{})

if err != nil {
    panic("Couldn't list groups! " + err.Error())
}

for _, group := range listGroupsResult.Groups {
    fmt.Printf("group %s - %s\n", *group.GroupId, *group.Arn)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [ListGroups](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

列出组。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { ListGroupsCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
    RoleName: 'ROLE_NAME', /* This is a number value. Required */
    Marker: 'MARKER', /* This is a string value. Optional */
    MaxItems: 'MAX_ITEMS' /* This is a number value. Optional */
};

export const run = async () => {
    try {
        const data = await iamClient.send(new ListGroupsCommand({}));
        console.log("Success", data.Groups);
    }
}
```



```
    } catch (err) {  
        console.log("Error", err);  
    }  
}  
run();
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [ListGroups](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
$uuid = uniqid();  
$service = new IamService();  
  
public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)  
{  
    $listGroupsArguments = [];  
    if ($pathPrefix) {  
        $listGroupsArguments["PathPrefix"] = $pathPrefix;  
    }  
    if ($marker) {  
        $listGroupsArguments["Marker"] = $marker;  
    }  
    if ($maxItems) {  
        $listGroupsArguments["MaxItems"] = $maxItems;  
    }  
  
    return $this->iamClient->listGroups($listGroupsArguments);  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [ListGroups](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def list_groups(count):  
    """  
    Lists the specified number of groups for the account.  
  
    :param count: The number of groups to list.  
    """  
    try:  
        for group in iam.groups.limit(count):  
            logger.info("Group: %s", group.name)  
    except ClientError:  
        logger.exception("Couldn't list groups for the account.")  
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [ListGroups](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Lists up to a specified number of groups for the account.
#
# @param count [Integer] The maximum number of groups to list.
def list_groups(count)
  @iam_resource.groups.limit(count).each do |group|
    puts("\t#{group.name}")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't list groups for the account. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [ListGroups](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn list_groups(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListGroupsOutput, SdkError<ListGroupsError>> {
    let response = client
        .list_groups()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [ListGroups](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用](#) (p. 15)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包列出 IAM 角色的内联策略

以下代码示例展示如何列出 IAM 角色的内联策略。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;
using System;

var client = new AmazonIdentityManagementServiceClient();
var request = new ListRolePoliciesRequest
{
    RoleName = "LambdaS3Role",
};

var response = new ListRolePoliciesResponse();

do
{
    response = await client.ListRolePoliciesAsync(request);

    if (response.PolicyNames.Count > 0)
    {
        response.PolicyNames.ForEach(policyName =>
        {
            Console.WriteLine($"{policyName}");
        });
    }

    // As long as response.IsTruncated is true, set request.Marker equal
    // to response.Marker and call ListRolesAsync again.
    if (response.IsTruncated)
    {
        request.Marker = response.Marker;
    }
} while (response.IsTruncated);
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [ListRolePolicies](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// ListRolePolicies

rolePoliciesList, err := service.ListRolePolicies(context.Background(),
&iam.ListRolePoliciesInput{
    RoleName: aws.String(ExampleRoleName),
})

if err != nil {
    panic("Couldn't list policies for role: " + err.Error())
}

for _, rolePolicy := range rolePoliciesList.PolicyNames {
    fmt.Printf("Policy ARN: %v", rolePolicy)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [ListRolePolicies](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

列出策略。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { ListRolePoliciesCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
    RoleName: 'ROLE_NAME', /* This is a number value. Required */
    Marker: 'MARKER', /* This is a string value. Optional */
    MaxItems: 'MAX_ITEMS' /* This is a number value. Optional */
};

export const run = async () => {
    try {
        const results = await iamClient.send(new ListRolePoliciesCommand(params));
        console.log("Success", results);
        return results;
    } catch (err) {
        console.log("Error", err);
    }
}

run();
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [ListRolePolicies](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
$uuid = uniqid();
$service = new IamService();

public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
{
    $listRolePoliciesArguments = ['RoleName' => $roleName];
    if ($marker) {
        $listRolePoliciesArguments['Marker'] = $marker;
    }
    if ($maxItems) {
        $listRolePoliciesArguments['MaxItems'] = $maxItems;
    }
    return $this->customWaiter(function () use ($listRolePoliciesArguments) {
        return $this->iamClient->listRolePolicies($listRolePoliciesArguments);
    });
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [ListRolePolicies](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def list_policies(role_name):
    """
    Lists inline policies for a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.policies.all():
            logger.info("Got inline policy %s.", policy.name)
    except ClientError:
        logger.exception("Couldn't list inline policies for %s.", role_name)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [ListRolePolicies](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn list_role_policies(
    client: &iamClient,
    role_name: &str,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolePoliciesOutput, SdkError<ListRolePoliciesError>> {
    let response = client
        .list_role_policies()
        .role_name(role_name)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [ListRolePolicies](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包列出 IAM policy

以下代码示例显示如何列出 IAM policy。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;
using System;

var client = new AmazonIdentityManagementServiceClient();

var request = new ListPoliciesRequest
{
    MaxItems = 10,
};

var response = new ListPoliciesResponse();
```

```
do
{
    response = await client.ListPoliciesAsync(request);
    response.Policies.ForEach(policy =>
    {
        Console.WriteLine($"{policy.PolicyName} ");
        Console.WriteLine($"with ID: {policy.PolicyId} ");
        Console.WriteLine($"and ARN: {policy.Arn}. ");
        Console.WriteLine($"It was created on {policy.CreateDate}.");
    });

    if (response.IsTruncated)
    {
        request.Marker = response.Marker;
    }
} while (response.IsTruncated);
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [ListPolicies](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// ListPolicies

policyListResponse, err := service.ListPolicies(context.Background(),
&iam.ListPoliciesInput{})

if err != nil {
    panic("Couldn't get list of policies! " + err.Error())
}

fmt.Print("PolicyName\tARN")
for _, policy := range policyListResponse.Policies {
    fmt.Printf("%s\t%s\n", *policy.PolicyName, *policy.Arn)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [ListPolicies](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
```



```
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

列出策略。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { ListPoliciesCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  Marker: 'MARKER',
  MaxItems: 'MAX_ITEMS',
  OnlyAttached: "ONLY_ATTACHED", /* Options are "true" or "false"*/
  PathPrefix: 'PATH_PREFIX',
  PolicyUsageFilter: "POLICY_USAGE_FILTER", /* Options are "PermissionsPolicy" or
  "PermissionsBoundary"*/
  Scope: "SCOPE" /* Options are "All", "AWS", "Local"*/
};

export const run = async () => {
  try {
    const results = await iamClient.send(new ListPoliciesCommand(params));
    console.log("Success", results);
    return results;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [ListPolicies](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
$uuid = uniqid();
$service = new IamService();

public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
{
    $listPoliciesArguments = [];
    if ($pathPrefix) {
        $listPoliciesArguments["PathPrefix"] = $pathPrefix;
    }
    if ($marker) {
        $listPoliciesArguments["Marker"] = $marker;
    }
    if ($maxItems) {
        $listPoliciesArguments["MaxItems"] = $maxItems;
    }

    return $this->iamClient->listPolicies($listPoliciesArguments);
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [ListPolicies](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are
    returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [ListPolicies](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Lists up to a specified number of policies in the account.
#
# @param count [Integer] The maximum number of policies to list.
# @return [Array] The Amazon Resource Names (ARNs) of the policies listed.
def list_policies(count)
  policy_arns = []
  @iam_resource.policies.limit(count).each_with_index do |policy, index|
    puts("\t#{index + 1}: #{policy.policy_name}: #{policy.arn}")
    policy_arns.append(policy.arn)
  end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list policies for the account. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    policy_arns
  end
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [ListPolicies](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn list_policies(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListPoliciesOutput, SdkError<ListPoliciesError>> {
    let response = client
        .list_policies()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [ListPolicies](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包列出附加到 IAM 角色的策略

以下代码示例显示如何列出附加到 IAM 角色的策略。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();
var request = new ListAttachedRolePoliciesRequest
{
    MaxItems = 10,
    RoleName = "testAssumeRole",
}
```

```
};

var response = await client.ListAttachedRolePoliciesAsync(request);

do
{
    response.AttachedPolicies.ForEach(policy =>
    {
        Console.WriteLine($"{policy.PolicyName} with ARN: {policy.PolicyArn}");
    });

    if (response.IsTruncated)
    {
        request.Marker = response.Marker;
        response = await client.ListAttachedRolePoliciesAsync(request);
    }
} while (response.IsTruncated);
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [ListAttachedRolePolicies](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// ListAttachedRolePolicies

attachedPoliciesList, err :=
service.ListAttachedRolePolicies(context.Background(),
&iam.ListAttachedRolePoliciesInput{
    RoleName: aws.String(ExampleRoleName),
})

if err != nil {
    panic("Couldn't call ListAttachedRolePolicies: " + err.Error())
}

fmt.Println("## List attached role policies for " + ExampleRoleName)

for _, attachedPolicy := range attachedPoliciesList.AttachedPolicies {
    fmt.Printf("attached policy: %v\n (%v) \n", attachedPolicy.PolicyArn,
attachedPolicy.PolicyName)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [ListAttachedRolePolicies](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

列出附加到角色的策略。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { ListAttachedRolePoliciesCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  RoleName: 'ROLE_NAME' /* required */
};

export const run = async () => {
  try {
    const data = await iamClient.send(new
      ListAttachedRolePoliciesCommand(params));
    console.log("Success", data.AttachedPolicies);
  } catch (err) {
    console.log("Error", err);
  }
}
run();
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [ListAttachedRolePolicies](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
$uuid = uniqid();
$service = new IamService();

public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker =
"", $maxItems = 0)
{
    $listAttachRolePoliciesArguments = ['RoleName' => $roleName];
    if ($pathPrefix) {
        $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;
    }
    if ($marker) {
        $listAttachRolePoliciesArguments['Marker'] = $marker;
    }
    if ($maxItems) {
        $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;
    }
}
```

```
        return $this->iamClient->listAttachedRolePolicies($listAttachRolePoliciesArguments);
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [ListAttachedRolePolicies](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def list_attached_policies(role_name):
    """
    Lists policies attached to a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.attached_policies.all():
            logger.info("Got policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't list attached policies for %s.", role_name)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [ListAttachedRolePolicies](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Deletes a role. If the role has policies attached, they are detached and
# deleted before the role is deleted.
#
# @param role [Aws::IAM::Role] The role to delete.
def delete_role(role)
  role.attached_policies.each do |policy|
    name = policy.policy_name
    policy.detach_role(role_name: role.name)
    policy.delete
    puts("Deleted policy #{name}.")
  end
  name = role.name
  role.delete
  puts("Deleted role #{name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't detach policies and delete role #{role.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
```

```
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [ListAttachedRolePolicies](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn list_attached_role_policies(
    client: &iamClient,
    role_name: String,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListAttachedRolePoliciesOutput,
SdkError<ListAttachedRolePoliciesError>> {
    let response = client
        .list_attached_role_policies()
        .role_name(role_name)
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [ListAttachedRolePolicies](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包列出 IAM 角色

以下代码示例显示如何列出 IAM 角色。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using Amazon.IdentityManagement;
```



```
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();

// Without the MaxItems value, the ListRolesAsync method will
// return information for up to 100 roles. If there are more
// than the MaxItems value or more than 100 roles, the response
// value IsTruncated will be true.
var request = new ListRolesRequest
{
    MaxItems = 10,
};

var response = new ListRolesResponse();

do
{
    response = await client.ListRolesAsync(request);
    response.Roles.ForEach(role =>
    {
        Console.WriteLine($"{role.RoleName} - ARN {role.Arn}");
    });

    // As long as response.IsTruncated is true, set request.Marker equal
    // to response.Marker and call ListRolesAsync again.
    if (response.IsTruncated)
    {
        request.Marker = response.Marker;
    }
} while (response.IsTruncated);
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [ListRoles](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// ListRoles

roles, err := service.ListRoles(context.Background(), &iam.ListRolesInput{})

if err != nil {
    panic("Could not list roles: " + err.Error())
}

fmt.Println("## list roles")
for _, idxRole := range roles.Roles {

    fmt.Printf("%s\t%s\t%s\t",
        *idxRole.RoleId,
        *idxRole.RoleName,
        *idxRole.Arn)
    if idxRole.Description != nil {
        fmt.Print(*idxRole.Description)
    }
    fmt.Print("\n")
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [ListRoles](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

列出角色。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { ListRolesCommand } from "@aws-sdk/client-iam";

// Set the parameters.
const params = {
  Marker: 'MARKER', // This is a string value.
  MaxItems: 'MAX_ITEMS' // This is a number value.
};

const run = async () => {
  try {
    const results = await iamClient.send(new ListRolesCommand(params));
    console.log("Success", results);
    return results;
  } catch (err) {
    console.log("Error", err);
  }
};

run();
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [ListRoles](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
$uuid = uniqid();
$service = new IamService();
```

```
/**
 * @param string $pathPrefix
 * @param string $marker
 * @param int $maxItems
 * @return Result
 * $roles = $service->listRoles();
 */
public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
{
    $listRolesArguments = [];
    if ($pathPrefix) {
        $listRolesArguments["PathPrefix"] = $pathPrefix;
    }
    if ($marker) {
        $listRolesArguments["Marker"] = $marker;
    }
    if ($maxItems) {
        $listRolesArguments["MaxItems"] = $maxItems;
    }
    return $this->iamClient->listRoles($listRolesArguments);
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [ListRoles](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def list_roles(count):
    """
    Lists the specified number of roles for the account.

    :param count: The number of roles to list.
    """
    try:
        roles = list(iam.roles.limit(count=count))
        for role in roles:
            logger.info("Role: %s", role.name)
    except ClientError:
        logger.exception("Couldn't list roles for the account.")
        raise
    else:
        return roles
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [ListRoles](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Lists up to a specified number of roles for the account.
#
# @param count [Integer] The maximum number of roles to list.
# @return [Array] The names of the listed roles.
def list_roles(count)
  role_names = []
  @iam_resource.roles.limit(count).each_with_index do |role, index|
    puts("\t#{index + 1}: #{role.name}")
    role_names.append(role.name)
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't list roles for the account. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  role_names
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [ListRoles](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn list_roles(
  client: &iamClient,
  path_prefix: Option<String>,
  marker: Option<String>,
  max_items: Option<i32>,
) -> Result<ListRolesOutput, SdkError<ListRolesError>> {
  let response = client
    .list_roles()
    .set_path_prefix(path_prefix)
    .set_marker(marker)
    .set_max_items(max_items)
    .send()
    .await?;
  Ok(response)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [ListRoles](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包列出 IAM 服务器证书

以下代码示例显示如何列出 IAM 服务器证书。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

列出证书。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { ListServerCertificatesCommand } from "@aws-sdk/client-iam";

export const run = async () => {
  try {
    const data = await iamClient.send(new ListServerCertificatesCommand({}));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [ListServerCertificates](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

iam.listServerCertificates({}, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [ListServerCertificates](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包列出 IAM 用户

以下代码示例显示如何列出 IAM 用户。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;

var client = new AmazonIdentityManagementServiceClient();
var request = new ListUsersRequest
{
    MaxItems = 10,
};
var response = await client.ListUsersAsync(request);

do
{
    response.Users.ForEach(user =>
    {
        Console.WriteLine($"{user.UserName} created on {user.CreateDate}.");
        Console.WriteLine($"ARN: {user.Arn}\n");
    });

    request.Marker = response.Marker;
    response = await client.ListUsersAsync(request);
} while (response.IsTruncated);
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的 [ListUsers](#)。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// ListUsers

fmt.Println("## List users")
```

```
userListResult, err := service.ListUsers(context.Background(),
&iam.ListUsersInput{})
if err != nil {
    panic("Couldn't list users: " + err.Error())
}
for _, userResult := range userListResult.Users {
    fmt.Printf("%s\t%s\n", *userResult.UserName, *userResult.Arn)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [ListUsers](#)。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void listAllUsers(IamClient iam ) {

    try {

        boolean done = false;
        String newMarker = null;

        while(!done) {
            ListUsersResponse response;
            if (newMarker == null) {
                ListUsersRequest request = ListUsersRequest.builder().build();
                response = iam.listUsers(request);
            } else {
                ListUsersRequest request = ListUsersRequest.builder()
                    .marker(newMarker)
                    .build();

                response = iam.listUsers(request);
            }

            for(User user : response.users()) {
                System.out.format("\n Retrieved user %s", user.userName());
                AttachedPermissionsBoundary permissionsBoundary =
                user.permissionsBoundary();
                if (permissionsBoundary != null)
                    System.out.format("\n Permissions boundary details %s",
                permissionsBoundary.permissionsBoundaryTypeAsString());
            }

            if(!response.isTruncated()) {
                done = true;
            } else {
                newMarker = response.marker();
            }
        }

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


- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [ListUsers](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

列出用户。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { ListUsersCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = { MaxItems: 10 };

export const run = async () => {
  try {
    const data = await iamClient.send(new ListUsersCommand(params));
    return data;
    const users = data.Users || [];
    users.forEach(function (user) {
      console.log("User " + user.UserName + " created", user.CreateDate);
    });
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [ListUsers](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});
```

```
var params = {
    MaxItems: 10
};

iam.listUsers(params, function(err, data) {
    if (err) {
        console.log("Error", err);
    } else {
        var users = data.Users || [];
        users.forEach(function(user) {
            console.log("User " + user.UserName + " created", user.CreateDate);
        });
    }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [ListUsers](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun listAllUsers() {

    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listUsers(ListUsersRequest { })
        response.users?.forEach { user ->
            println("Retrieved user ${user.userName}")
            val permissionsBoundary = user.permissionsBoundary
            if (permissionsBoundary != null)
                println("Permissions boundary details\n${permissionsBoundary.permissionsBoundaryType}")
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [ListUsers](#)。

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
$uuuid = uniqid();
$service = new IamService();

public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)
```

```
{
    $listUsersArguments = [];
    if ($pathPrefix) {
        $listUsersArguments["PathPrefix"] = $pathPrefix;
    }
    if ($marker) {
        $listUsersArguments["Marker"] = $marker;
    }
    if ($maxItems) {
        $listUsersArguments["MaxItems"] = $maxItems;
    }

    return $this->iamClient->listUsers($listUsersArguments);
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的 [ListUsers](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [ListUsers](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
# Lists up to a specified number of users in the account.
#
# @param count [Integer] The maximum number of users to list.
def list_users(count)
    @iam_resource.users.limit(count).each do |user|
        puts "\t#{user.name}"
    end
rescue Aws::Errors::ServiceError => e
```

```
puts("Couldn't list users for the account. Here's why:")
puts("\t#{e.code}: #{e.message}")
raise
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [ListUsers](#)。

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
pub async fn list_users(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListUsersOutput, SdkError<ListUsersError>> {
    let response = client
        .list_users()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 [ListUsers](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包更新 IAM 服务器证书

以下代码示例演示了如何更新 IAM 服务器证书。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
```

```
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

更新服务器证书。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { UpdateServerCertificateCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  ServerCertificateName: "CERTIFICATE_NAME", //CERTIFICATE_NAME
  NewServerCertificateName: "NEW_CERTIFICATE_NAME", //NEW_CERTIFICATE_NAME
};

export const run = async () => {
  try {
    const data = await iamClient.send(
      new UpdateServerCertificateCommand(params)
    );
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [UpdateServerCertificate](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  ServerCertificateName: 'CERTIFICATE_NAME',
  NewServerCertificateName: 'NEW_CERTIFICATE_NAME'
};

iam.updateServerCertificate(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [UpdateServerCertificate](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包更新 IAM 用户

以下代码示例演示了如何更新 IAM 用户。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
package main

import (
    "context"
    "flag"
    "fmt"

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

// IAMUpdateUserAPI defines the interface for the UpdateUser function.
// We use this interface to test the function using a mocked service.
type IAMUpdateUserAPI interface {
    UpdateUser(ctx context.Context,
        params *iam.UpdateUserInput,
        optFns ...func(*iam.Options)) (*iam.UpdateUserOutput, error)
}

// RenameUser changes the name for an AWS Identity and Access Management (IAM)
// user.
// Inputs:
//     c is the context of the method call, which includes the AWS Region.
//     api is the interface that defines the method call.
//     input defines the input arguments to the service call.
// Output:
//     If successful, a UpdateUserOutput object containing the result of the
//     service call and nil.
//     Otherwise, nil and an error from the call to UpdateUser.
func RenameUser(c context.Context, api IAMUpdateUserAPI, input
    *iam.UpdateUserInput) (*iam.UpdateUserOutput, error) {
    return api.UpdateUser(c, input)
}

func main() {
    userName := flag.String("u", "", "The name of the user")
    newName := flag.String("n", "", "The new name of the user")
    flag.Parse()

    if *userName == "" || *newName == "" {
        fmt.Println("You must supply a user name and new name (-u USERNAME -n NEW-NAME)")
        return
    }
}
```

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic("configuration error, " + err.Error())
}

client := iam.NewFromConfig(cfg)

input := &iam.UpdateUserInput{
    UserName:    userName,
    NewUserName: newName,
}

_, err = RenameUser(context.TODO(), client, input)
if err != nil {
    fmt.Println("Got an error updating user " + *userName)
}

fmt.Println("Updated user name from: " + *userName + " to: " + *newName)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [UpdateUser](#)。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
public static void updateIAMUser(IamClient iam, String curName,String newName )
{
    try {
        UpdateUserRequest request = UpdateUserRequest.builder()
            .userName(curName)
            .newUserName(newName)
            .build();

        iam.updateUser(request);
        System.out.printf("Successfully updated user to username %s", newName);
    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [UpdateUser](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

更新用户。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { UpdateUserCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  UserName: "ORIGINAL_USER_NAME", //ORIGINAL_USER_NAME
  NewUserName: "NEW_USER_NAME", //NEW_USER_NAME
};

export const run = async () => {
  try {
    const data = await iamClient.send(new UpdateUserCommand(params));
    console.log("Success, username updated");
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [UpdateUser](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  UserName: process.argv[2],
  NewUserName: process.argv[3]
};

iam.updateUser(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

```
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [UpdateUser](#)。

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
suspend fun updateIAMUser(curName: String?, newName: String?) {  
  
    val request = UpdateUserRequest {  
        userName = curName  
        newUserName = newName  
    }  
  
    iamClient { region = "AWS_GLOBAL" }.use { iamClient ->  
        iamClient.updateUser(request)  
        println("Successfully updated user to $newName")  
    }  
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的 [UpdateUser](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def update_user(user_name, new_user_name):  
    """  
    Updates a user's name.  
  
    :param user_name: The current name of the user to update.  
    :param new_user_name: The new name to assign to the user.  
    :return: The updated user.  
    """  
    try:  
        user = iam.User(user_name)  
        user.update(NewUserName=new_user_name)  
        logger.info("Renamed %s to %s.", user_name, new_user_name)  
    except ClientError:  
        logger.exception("Couldn't update name for user %s.", user_name)  
        raise  
    return user
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [UpdateUser](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包更新 IAM 访问密钥

以下代码示例演示了如何更新 IAM 访问密钥。

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
package main

import (
    "context"
    "flag"
    "fmt"

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

// IAMUpdateAccessKeyAPI defines the interface for the UpdateAccessKey function.
// We use this interface to test the function using a mocked service.
type IAMUpdateAccessKeyAPI interface {
    UpdateAccessKey(ctx context.Context,
        params *iam.UpdateAccessKeyInput,
        optFns ...func(*iam.Options)) (*iam.UpdateAccessKeyOutput, error)
}

// ActivateKey sets the status of an AWS Identity and Access Management (IAM)
// access key to active.
// Inputs:
//   c is the context of the method call, which includes the AWS Region.
//   api is the interface that defines the method call.
//   input defines the input arguments to the service call.
// Output:
//   If successful, a UpdateAccessKeyOutput object containing the result of the
//   service call and nil.
//   Otherwise, nil and an error from the call to UpdateAccessKey.
func ActivateKey(c context.Context, api IAMUpdateAccessKeyAPI, input
    *iam.UpdateAccessKeyInput) (*iam.UpdateAccessKeyOutput, error) {
    return api.UpdateAccessKey(c, input)
}

func main() {
    keyID := flag.String("k", "", "The ID of the access key")
    userName := flag.String("u", "", "The name of the user")
    flag.Parse()

    if *keyID == "" || *userName == "" {
        fmt.Println("You must supply an access key ID and user name (-k KEY-ID -u USER-NAME)")
        return
    }
}
```

```

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic("configuration error, " + err.Error())
}

client := iam.NewFromConfig(cfg)

input := &iam.UpdateAccessKeyInput{
    AccessKeyId: keyID,
    Status:      types.StatusTypeActive,
    UserName:    userName,
}

_, err = ActivateKey(context.TODO(), client, input)
if err != nil {
    fmt.Println("Error", err)
    return
}

fmt.Println("Access Key activated")
}

```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的 [UpdateAccessKey](#)。

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```

public static void updateKey(IamClient iam, String username, String accessId,
String status ) {

    try {
        if (status.toLowerCase().equalsIgnoreCase("active")) {
            statusType = StatusType.ACTIVE;
        } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
            statusType = StatusType.INACTIVE;
        } else {
            statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
        }

        UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
            .accessKeyId(accessId)
            .userName(username)
            .status(statusType)
            .build();

        iam.updateAccessKey(request);
        System.out.printf("Successfully updated the status of access key %s to"
+
            "status %s for user %s", accessId, status, username);

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

```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的 [UpdateAccessKey](#)。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
const REGION = "REGION"; // For example, "us-east-1".
// Create an IAM service client object.
const iamClient = new IAMClient({ region: REGION });
export { iamClient };
```

更新访问密钥。

```
// Import required AWS SDK clients and commands for Node.js.
import { iamClient } from "../libs/iamClient.js";
import { UpdateAccessKeyCommand } from "@aws-sdk/client-iam";

// Set the parameters.
export const params = {
  AccessKeyId: "ACCESS_KEY_ID", //ACCESS_KEY_ID
  Status: "Active",
  UserName: "USER_NAME", //USER_NAME
};

export const run = async () => {
  try {
    const data = await iamClient.send(new UpdateAccessKeyCommand(params));
    console.log("Success", data);
    return data;
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [UpdateAccessKey](#)。

SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
var AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});
```

```
// Create the IAM service object
var iam = new AWS.IAM({apiVersion: '2010-05-08'});

var params = {
  AccessKeyId: 'ACCESS_KEY_ID',
  Status: 'Active',
  UserName: 'USER_NAME'
};

iam.updateAccessKey(params, function(err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅 [Amazon SDK for JavaScript 开发人员指南](#)。
- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [UpdateAccessKey](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
    deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", 'Activated' if activate else 'Deactivated',
            key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", 'Activate' if activate else 'Deactivate',
            key_id)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [UpdateAccessKey](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包的 IAM 场景

以下代码示例显示如何通过 Amazon 开发工具包实施 IAM 中的常见场景。这些场景显示了如何通过 IAM 中调用多个函数来完成特定任务。每个场景都包含一个指向 GitHub 的链接，其中包含了有关如何设置和运行代码的说明。

示例

- [使用 Amazon STS 开发工具包创建 IAM 用户并使用 Amazon 代入角色 \(p. 660\)](#)
- [使用 Amazon 开发工具包创建只读和读写 IAM 用户 \(p. 710\)](#)
- [使用 Amazon 开发工具包管理 IAM 访问密钥 \(p. 716\)](#)
- [使用 Amazon 开发工具包管理 IAM policy \(p. 719\)](#)
- [使用 Amazon 开发工具包管理 IAM 角色 \(p. 723\)](#)
- [使用 Amazon 开发工具包管理您的 IAM 账户 \(p. 725\)](#)
- [使用 Amazon SDK 回滚 IAM policy 版本 \(p. 729\)](#)

使用 Amazon STS 开发工具包创建 IAM 用户并使用 Amazon 代入角色

以下代码示例显示了如何：

- 创建一个没有权限的用户。
- 创建授予列出账户的 Amazon S3 存储桶的权限的角色。
- 添加策略以允许用户代入该角色。
- 代入角色并使用临时凭证列出 Amazon S3 存储桶。
- 删除策略、角色和用户。

.NET

Amazon SDK for .NET

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon;
using Amazon.IdentityManagement;
using Amazon.IdentityManagement.Model;
using Amazon.S3;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;

public class IAM_Basics
{
    // Values needed for user, role, and policies.
    private const string Username = "example-user";
    private const string S3PolicyName = "s3-list-buckets-policy";
    private const string RoleName = "temporary-role";
    private const string AssumePolicyName = "sts-trust-user";

    private static readonly RegionEndpoint Region = RegionEndpoint.USEast2;
```



```
public static async Task Main()
{
    DisplayInstructions();

    // Create the IAM client object.
    var client = new AmazonIdentityManagementServiceClient(Region);

    // First create a user. By default, the new user has
    // no permissions.
    Console.WriteLine($"Creating a new user with user name: {UserName}.");
    var user = await CreateUserAsync(client, UserName);
    var userArn = user.Arn;
    Console.WriteLine($"Successfully created user: {UserName} with ARN:
{userArn}.");

    // Create an AccessKey for the user.
    var accessKey = await CreateAccessKeyAsync(client, UserName);

    var accessKeyId = accessKey.AccessKeyId;
    var secretAccessKey = accessKey.SecretAccessKey;

    // Try listing the Amazon Simple Storage Service (Amazon S3)
    // buckets. This should fail at this point because the user doesn't
    // have permissions to perform this task.
    var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
    await ListMyBucketsAsync(s3Client1);

    // Define a role policy document that allows the new user
    // to assume the role.
    // string assumeRolePolicyDocument =
    File.ReadAllText("assumePolicy.json");
    string assumeRolePolicyDocument = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Effect\": \"Allow\"," +
        "\"Principal\": {" +
        "$\" \"AWS\": \"{userArn}\"" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "}]" +
        "}";

    // Permissions to list all buckets.
    string policyDocument = "{" +
        "\"Version\": \"2012-10-17\"," +
        "\"Statement\": [{" +
        "\"Action\" : [\"s3:ListAllMyBuckets\"]," +
        "\"Effect\" : \"Allow\"," +
        "\"Resource\" : \"*\"]" +
        "}";

    // Create the role to allow listing the S3 buckets. Role names are
    // not case sensitive and must be unique to the account for which it
    // is created.
    var role = await CreateRoleAsync(client, RoleName,
    assumeRolePolicyDocument);
    var roleArn = role.Arn;

    // Create a policy with permissions to list S3 buckets
    var policy = await CreatePolicyAsync(client, S3PolicyName,
    policyDocument);

    // Wait 15 seconds for the policy to be created.
    WaitABit(15, "Waiting for the policy to be available.");
```

```
// Attach the policy to the role you created earlier.
await AttachRoleAsync(client, policy.Arn, RoleName);

// Wait 15 seconds for the role to be updated.
Console.WriteLine();
WaitABit(15, "Waiting to time for the policy to be attached.");

// Use the AWS Security Token Service (AWS STS) to have the user
// assume the role we created.
var stsClient = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

// Wait for the new credentials to become valid.
WaitABit(10, "Waiting for the credentials to be valid.");

var assumedRoleCredentials = await AssumeS3RoleAsync(stsClient,
"temporary-session", roleArn);

// Try again to list the buckets using the client created with
// the new user's credentials. This time, it should work.
var s3Client2 = new AmazonS3Client(assumedRoleCredentials);

await ListMyBucketsAsync(s3Client2);

// Now clean up all the resources used in the example.
await DeleteResourcesAsync(client, accessKeyId, UserName, policy.Arn,
RoleName);

Console.WriteLine("IAM Demo completed.");
}

/// <summary>
/// Create a new IAM user.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="userName">A string representing the user name of the
/// new user.</param>
/// <returns>The newly created user.</returns>
public static async Task<User> CreateUserAsync(
    AmazonIdentityManagementServiceClient client,
    string userName)
{
    var request = new CreateUserRequest
    {
        UserName = userName,
    };

    var response = await client.CreateUserAsync(request);

    return response.User;
}

/// <summary>
/// Create a new AccessKey for the user.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="userName">The name of the user for whom to create the
key.</param>
/// <returns>A new IAM access key for the user.</returns>
public static async Task<AccessKey> CreateAccessKeyAsync(
    AmazonIdentityManagementServiceClient client,
    string userName)
{

```

```

        var request = new CreateAccessKeyRequest
        {
            UserName = userName,
        };

        var response = await client.CreateAccessKeyAsync(request);

        if (response.AccessKey is not null)
        {
            Console.WriteLine($"Successfully created Access Key for
{userName}.");
        }

        return response.AccessKey;
    }

    /// <summary>
    /// Create a policy to allow a user to list the buckets in an account.
    /// </summary>
    /// <param name="client">The initialized IAM client object.</param>
    /// <param name="policyName">The name of the poicy to create.</param>
    /// <param name="policyDocument">The permissions policy document.</param>
    /// <returns>The newly created ManagedPolicy object.</returns>
    public static async Task<ManagedPolicy> CreatePolicyAsync(
        AmazonIdentityManagementServiceClient client,
        string policyName,
        string policyDocument)
    {
        var request = new CreatePolicyRequest
        {
            PolicyName = policyName,
            PolicyDocument = policyDocument,
        };

        var response = await client.CreatePolicyAsync(request);

        return response.Policy;
    }

    /// <summary>
    /// Attach the policy to the role so that the user can assume it.
    /// </summary>
    /// <param name="client">The initialized IAM client object.</param>
    /// <param name="policyArn">The ARN of the policy to attach.</param>
    /// <param name="roleName">The name of the role to attach the policy to.</
param>
    public static async Task AttachRoleAsync(
        AmazonIdentityManagementServiceClient client,
        string policyArn,
        string roleName)
    {
        var request = new AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        };

        var response = await client.AttachRolePolicyAsync(request);

        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("Successfully attached the policy to the role.");
        }
    }

```

```

    }
    else
    {
        Console.WriteLine("Could not attach the policy.");
    }
}

/// <summary>
/// Create a new IAM role which we can attach to a user.
/// </summary>
/// <param name="client">The initialized IAM client object.</param>
/// <param name="roleName">The name of the IAM role to create.</param>
/// <param name="rolePermissions">The permissions which the role will
have.</param>
/// <returns>A Role object representing the newly created role.</returns>
public static async Task<Role> CreateRoleAsync(
    AmazonIdentityManagementServiceClient client,
    string roleName,
    string rolePermissions)
{
    var request = new CreateRoleRequest
    {
        RoleName = roleName,
        AssumeRolePolicyDocument = rolePermissions,
    };

    var response = await client.CreateRoleAsync(request);

    return response.Role;
}

/// <summary>
/// List the Amazon S3 buckets owned by the user.
/// </summary>
/// <param name="accessKeyId">The access key Id for the user.</param>
/// <param name="secretAccessKey">The Secret access key for the user.</
param>
public static async Task ListMyBucketsAsync(AmazonS3Client client)
{
    Console.WriteLine("\nPress <Enter> to list the S3 buckets using the new
user.\n");
    Console.ReadLine();

    try
    {
        // Get the list of buckets accessible by the new user.
        var response = await client.ListBucketsAsync();

        // Loop through the list and print each bucket's name
        // and creation date.
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Listing S3 buckets:\n");
        response.Buckets
            .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
created on: {b.CreationDate}"));
    }
    catch (AmazonS3Exception ex)
    {
        // Something else went wrong. Display the error message.
        Console.WriteLine($"Error: {ex.Message}");
    }
}

```

```

        Console.WriteLine("Press <Enter> to continue.");
        Console.ReadLine();
    }

    /// <summary>
    /// Have the user assume the role that allows the role to be used to
    /// list all S3 buckets.
    /// </summary>
    /// <param name="client">An initialized AWS STS client object.</param>
    /// <param name="roleSession">The name of the session where the role
    /// assumption will be active.</param>
    /// <param name="roleToAssume">The Amazon Resource Name (ARN) of the
    /// role to assume.</param>
    /// <returns>The AssumedRoleUser object needed to perform the list
    /// buckets procedure.</returns>
    public static async Task<Credentials> AssumeS3RoleAsync(
        AmazonSecurityTokenServiceClient client,
        string roleSession,
        string roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };

        var response = await client.AssumeRoleAsync(request);

        return response.Credentials;
    }

    /// <summary>
    /// Delete the user, and other resources created for this example.
    /// </summary>
    /// <param name="client">The initialized client object.</param>
    /// <param name="accessKeyId">The Id of the user's access key.</param>
    /// <param name="userName">The user name of the user to delete.</param>
    /// <param name="policyName">The name of the policy to delete.</param>
    /// <param name="policyArn">The Amazon Resource Name ARN of the Policy to
    delete.</param>
    /// <param name="roleName">The name of the role that will be deleted.</
    param>
    public static async Task DeleteResourcesAsync(
        AmazonIdentityManagementServiceClient client,
        string accessKeyId,
        string userName,
        string policyArn,
        string roleName)
    {
        var detachPolicyResponse = await client.DetachRolePolicyAsync(new
        DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        var delPolicyResponse = await client.DeletePolicyAsync(new
        DeletePolicyRequest
        {
            PolicyArn = policyArn,
        });
    }

```

```

        var delRoleResponse = await client.DeleteRoleAsync(new
DeleteRoleRequest
        {
            RoleName = roleName,
        });

        var delAccessKey = await client.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        var delUserResponse = await client.DeleteUserAsync(new
DeleteUserRequest
        {
            UserName = userName,
        });
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public static void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        Console.WriteLine("\n\nPress <Enter> to continue.");
        Console.ReadLine();
    }

    /// <summary>
    /// Shows the a description of the features of the program.
    /// </summary>
    public static void DisplayInstructions()
    {
        var separator = new string('-', 80);

        Console.WriteLine(separator);
        Console.WriteLine("IAM Basics");
        Console.WriteLine("This application uses the basic features of the AWS
Identity and Access");
        Console.WriteLine("Management (IAM) creating, managing, and controlling
access to resources for");
        Console.WriteLine("users. The application was created using the AWS SDK
for .NET version 3.7 and");
        Console.WriteLine(".NET Core 5. The application performs the following
actions:");
        Console.WriteLine();
        Console.WriteLine("1. Creates a user with no permissions");
        Console.WriteLine("2. Creates a rolw and policy that grants
s3:ListAllMyBuckets permission");
        Console.WriteLine("3. Grants the user permission to assume the role");
        Console.WriteLine("4. Creates an Amazon Simple Storage Service (Amazon
S3) client and tries");
    }

```

```
        Console.WriteLine("    to list buckets. (This should fail.)");
        Console.WriteLine("5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("6. Creates an Amazon S3 client object with the
temporary credentials and");
        Console.WriteLine("    lists the buckets. (This time it should work.)");
        Console.WriteLine("7. Deletes all of the resources created.");
        Console.WriteLine(separator);
        Console.WriteLine("Press <Enter> to continue.");
        Console.ReadLine();
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for .NET API 参考》中的以下主题。

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

C++

适用于 C++ 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
namespace AwsDoc {
    namespace IAM {
        //! Cleanup by deleting created entities.
        /*!
        \sa DeleteCreatedEntities
        \param client IAM client.
        \param role IAM role.
        \param user IAM user.
        \param policy IAM policy.
        \param logProgress enables verbose logging.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                         const Aws::IAM::Model::Role &role,
                                         const Aws::IAM::Model::User &user,
                                         const Aws::IAM::Model::Policy &policy,
                                         bool logProgress);
    }
}

//! Scenario to create an IAM user, create an IAM role, and apply the role to the
use.
```



```
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are need to run this code; (note, it may be
// necessary to
// create a custom policy).
/*!
\sa IAMCreateUserAssumeRoleScenario
\param clientConfig Aws client configuration.
\param logProgress enables verbose logging.
*/
bool AwsDoc::IAM::IAMCreateUserAssumeRoleScenario(const
    Aws::Client::ClientConfiguration &clientConfig,
    bool logProgress) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
            Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);

        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else if (logProgress) {
            std::cout << "Successfully created IAM user " << userName << std::endl;
        }

        user = outcome.GetResult().GetUser();
    }

    // 2. Create a role.
    {
        // Get the IAM user for the current client in order to access its ARN.
        Aws::String iamUserArn;
        {
            Aws::IAM::Model::GetUserRequest request;
            Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error getting Iam user. " <<
                    outcome.GetError().GetMessage() << std::endl;

                DeleteCreatedEntities(client, role, user, policy, logProgress);
                return false;
            }
            else if (logProgress) {
                std::cout << "Successfully retrieved Iam user " <<
                    outcome.GetResult().GetUser().GetUserName()
                    << std::endl;
            }

            iamUserArn = outcome.GetResult().GetUser().GetArn();
        }

        Aws::IAM::Model::CreateRoleRequest request;

        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleName = "iam-demo-role-" +
```

```

        Aws::Utils::StringUtils::ToLower(uuid.c_str());
request.SetRoleName(roleName);

// Build policy document for role.
Aws::Utils::Document jsonStatement;
jsonStatement.WithString("Effect", "Allow");

Aws::Utils::Document jsonPrincipal;
jsonPrincipal.WithString("AWS", iamUserArn);
jsonStatement.WithObject("Principal", jsonPrincipal);
jsonStatement.WithString("Action", "sts:AssumeRole");
jsonStatement.WithObject("Condition", Aws::Utils::Document());

Aws::Utils::Document policyDocument;
policyDocument.WithString("Version", "2012-10-17");

Aws::Utils::Array<Aws::Utils::Document> statements(1);
statements[0] = jsonStatement;
policyDocument.WithArray("Statement", statements);

if (logProgress) {
    std::cout << "Setting policy for role\n" <<
policyDocument.View().WriteCompact() << std::endl;
}

// Set role policy document as JSON string.
request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error creating role. " <<
        outcome.GetError().GetMessage() << std::endl;

    DeleteCreatedEntities(client, role, user, policy, logProgress);
    return false;
}
else if (logProgress) {
    std::cout << "Successfully created a role with name " << roleName <<
std::endl;
}

role = outcome.GetResult().GetRole();
}

// 3. Create an IAM policy.
{
    Aws::IAM::Model::CreatePolicyRequest request;
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String policyName = "iam-demo-policy-" +
        Aws::Utils::StringUtils::ToLower(uuid.c_str());
    request.SetPolicyName(policyName);

    // Build IAM policy document.
    Aws::Utils::Document jsonStatement;
    jsonStatement.WithString("Effect", "Allow");
    jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
    jsonStatement.WithString("Resource", "arn:aws:s3:*");

    Aws::Utils::Document policyDocument;
    policyDocument.WithString("Version", "2012-10-17");

    Aws::Utils::Array<Aws::Utils::Document> statements(1);
    statements[0] = jsonStatement;
    policyDocument.WithArray("Statement", statements);

    if (logProgress) {

```

```

        std::cout << "Creating a policy.\n" <<
policyDocument.View().WriteCompact() << std::endl;
    }

    // Set IAM policy document as JSON string.
    request.SetPolicyDocument(policyDocument.View().WriteCompact());

    Aws::IAM::Model::CreatePolicyOutcome outcome =
client.CreatePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy. " <<
            outcome.GetError().GetMessage() << std::endl;

        DeleteCreatedEntities(client, role, user, policy, logProgress);
        return false;
    }
    else if (logProgress) {
        std::cout << "Successfully created a policy with name, " << policyName
<<
            "." << std::endl;
    }

    policy = outcome.GetResult().GetPolicy();
}

// 4. Assume the new role using the AWS Security Token Service (STS).
Aws::STS::Model::Credentials credentials;
{
    Aws::STS::STSCClient stsClient(clientConfig);

    Aws::STS::Model::AssumeRoleRequest request;
    request.SetRoleArn(role.GetArn());
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String roleSessionName = "iam-demo-role-session-" +

Aws::Utils::StringUtils::ToLower(uuid.c_str());
    request.SetRoleSessionName(roleSessionName);

    Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

    // Repeatedly call AssumeRole, because there is often a delay
    // before the role is available to be assumed.
    // Repeat at most 20 times when access is denied.
    int count = 0;
    while (true) {
        assumeRoleOutcome = stsClient.AssumeRole(request);
        if (!assumeRoleOutcome.IsSuccess()) {
            if (count > 20 ||
                assumeRoleOutcome.GetError().GetErrorType() !=
Aws::STS::STSErrors::ACCESS_DENIED) {
                std::cerr << "Error assuming role after 20 tries. " <<
                    assumeRoleOutcome.GetError().GetMessage() <<
std::endl;

                DeleteCreatedEntities(client, role, user, policy, logProgress);
                return false;
            }
            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {
            if (logProgress) {
                std::cout << "Successfully assumed the role after " << count <<
" seconds." << std::endl;
            }
            break;
        }
    }
}

```

```

        count++;
    }

    credentials = assumeRoleOutcome.GetResult().GetCredentials();
}

// 5. List objects in the bucket (This should fail).
{
    Aws::S3::S3Client
s3Client(Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
credentials.GetSecretAccessKey(),
credentials.GetSessionToken()),
        clientConfig);
    Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
    if (!listBucketsOutcome.IsSuccess()) {
        if (listBucketsOutcome.GetError().GetErrorType() !=
Aws::S3::S3Errors::ACCESS_DENIED) {
            std::cerr << "Could not lists buckets. " <<
                listBucketsOutcome.GetError().GetMessage() << std::endl;
        }
        else if (logProgress) {
            std::cout << "Access to list buckets denied because privileges have
not been applied."
                << std::endl;
        }
    }
    else {
        std::cerr << "Successfully retrieved bucket lists when this should not
happen." << std::endl;
    }
}

// 6. Attach the policy to the role.
{
    Aws::IAM::Model::AttachRolePolicyRequest request;
    request.SetRoleName(role.GetRoleName());
    request.WithPolicyArn(policy.GetArn());

    Aws::IAM::Model::AttachRolePolicyOutcome outcome =
client.AttachRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy. " <<
            outcome.GetError().GetMessage() << std::endl;

        DeleteCreatedEntities(client, role, user, policy, logProgress);
        return false;
    }
    else if (logProgress) {
        std::cout << "Successfully attached the policy with name, " <<
policy.GetPolicyName() <<
            ", to the role, " << role.GetRoleName() << "." << std::endl;
    }
}

int count = 0;
// 7. List objects in the bucket (this should succeed).
// Repeatedly call ListBuckets, because there is often a delay
// before the policy with ListBucket permissions has been applied to the role.
// Repeat at most 20 times when access is denied.
while (true) {
    Aws::S3::S3Client
s3Client(Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),

```

```

credentials.GetSecretAccessKey(),

credentials.GetSessionToken()),
    clientConfig);
    Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
    if (!listBucketsOutcome.IsSuccess()) {
        if ((count > 20) ||
            listBucketsOutcome.GetError().GetErrorType() !=
Aws::S3::S3Errors::ACCESS_DENIED) {
            std::cerr << "Could not lists buckets after 20 seconds. " <<
                listBucketsOutcome.GetError().GetMessage() << std::endl;
            DeleteCreatedEntities(client, role, user, policy, logProgress);
            return false;
        }

        std::this_thread::sleep_for(std::chrono::seconds(1));
    }
    else {
        if (logProgress) {
            std::cout << "Successfully retrieved bucket lists after " << count
                << " seconds." << std::endl;
        }
        break;
    }
    count++;
}

// 8. Delete all the created resources.
return DeleteCreatedEntities(client, role, user, policy, logProgress);
}

bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                        const Aws::IAM::Model::Role &role,
                                        const Aws::IAM::Model::User &user,
                                        const Aws::IAM::Model::Policy &policy,
                                        bool logProgress) {

    bool result = true;
    if (policy.ArnHasBeenSet()) {
        // Detach the policy from the role.
        {
            Aws::IAM::Model::DetachRolePolicyRequest request;
            request.SetPolicyArn(policy.GetArn());
            request.SetRoleName(role.GetRoleName());

            Aws::IAM::Model::DetachRolePolicyOutcome outcome =
client.DetachRolePolicy(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error Detaching policy from roles. " <<
                    outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else if (logProgress) {
                std::cout << "Successfully detached the policy with arn " <<
policy.GetArn()
                    << " from role " << role.GetRoleName() << "." <<
std::endl;
            }
        }

        // Delete the policy.
        {
            Aws::IAM::Model::DeletePolicyRequest request;
            request.WithPolicyArn(policy.GetArn());

```

```

        Aws::IAM::Model::DeletePolicyOutcome outcome =
client.DeletePolicy(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting policy. " <<
                outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else if (logProgress) {
            std::cout << "Successfully deleted the policy with arn " <<
policy.GetArn() << std::endl;
        }
    }

    }

    if (role.RoleIdHasBeenSet()) {
        // Delete the role.
        Aws::IAM::Model::DeleteRoleRequest request;
        request.SetRoleName(role.GetRoleName());

        Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting role. " <<
                outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else if (logProgress) {
            std::cout << "Successfully deleted the role with name " <<
role.GetRoleName() << std::endl;
        }
    }

    if (user.ArnHasBeenSet()) {
        // Delete the user.
        Aws::IAM::Model::DeleteUserRequest request;
        request.WithUserName(user.GetUserName());

        Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting user. " <<
                outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else if (logProgress) {
            std::cout << "Successfully deleted the user with name " <<
user.GetUserName() << std::endl;
        }
    }

    return result;
}

```

- 有关 API 详细信息，请参阅《Amazon SDK for C++ API 参考》中的以下主题。

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)

- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

Go

SDK for Go V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
package main

import (
    "context"
    "errors"
    "fmt"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/credentials"
    "github.com/aws/aws-sdk-go-v2/credentials/stscreds"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/sts"
    "github.com/aws/smithy-go"
)

/*
This is a basic scenario for using AWS Identity and Access Management (IAM) and AWS
Security Token Service (STS).

This example will do the following:

* Create a user that has no permissions.
* Create a role and policy that grant s3:ListAllMyBuckets permission.
* Grant the user permission to assume the role.
* Create an S3 client object as the user and try to list buckets (this should
fail).
* Get temporary credentials by assuming the role.
* Create an S3 client object with the temporary credentials and list the buckets
(this should succeed).
* Delete all the resources.

*/

const (
    scenario_UserName           = "ExampleUser123"
    scenario_BucketListerRoleName = "BucketListerRole"
    scenario_BucketListerPolicyName = "BucketListerListMyBucketsPolicy"
)

func scenario() {

    cfg, err := config.LoadDefaultConfig(context.Background())

    if err != nil {
        panic("Couldn't load a configuration")
    }
}
```



```

}

iamSvc := iam.NewFromConfig(cfg)

fmt.Println("## Create user")

var userInfo types.User

user, err := iamSvc.CreateUser(context.Background(), &iam.CreateUserInput{
    UserName: aws.String(scenario_UserName),
})

if err != nil {

    var existsAlready *types.EntityAlreadyExistsException
    if errors.As(err, &existsAlready) {
        // The user possibly exists.
        // Check if the user actually exists within IAM.
        uuser, err := iamSvc.GetUser(context.Background(), &iam.GetUserInput{UserName:
aws.String(scenario_UserName)})
        if err != nil {
            panic("Can't get user: " + err.Error())
        } else {
            // Make sure the user info is set for later.
            userInfo = *uuser.User
            fmt.Println("User already existed...")
        }
    } else {
        fmt.Println("Couldn't create user! " + err.Error())
    }
} else {
    userInfo = *user.User
}

fmt.Printf("User %s has id %s\n", scenario_UserName, *userInfo.Arn)

fmt.Println("## Create access key")

creds, err := iamSvc.CreateAccessKey(context.Background(),
&iam.CreateAccessKeyInput{
    UserName: aws.String(scenario_UserName),
})

if err != nil {
    panic("Couldn't create credentials for a user! " + err.Error())
}

akId := *creds.AccessKey.AccessKeyId
sakId := *creds.AccessKey.SecretAccessKey

fmt.Printf("### CREDs: accessKeyId(%s) Secretkey(%s)\n", akId, sakId)

// Grant the user the ability to assume the role.

fmt.Println("## waiting for a few moments for keys to become available")

time.Sleep(10 * time.Second)

fmt.Println("## Create the bucket lister role")
bucketListerRole, err := iamSvc.CreateRole(context.Background(),
&iam.CreateRoleInput{
    RoleName: aws.String(scenario_BucketListerRoleName),
    AssumeRolePolicyDocument: aws.String(`{
        "Version": "2012-10-17",
        "Statement": [
            {

```

```

        "Effect": "Allow",
        "Principal": {
            "AWS": "` + (*userInfo.Arn) + `"
        },
        "Action": "sts:AssumeRole"
    }
}
    }`),
    Description: aws.String("Role to let users list their buckets."),
})
var bucketListerRoleArn string
if err != nil {
    // Check to see if the role exists.
    var existsException *types.EntityAlreadyExistsException
    if errors.As(err, &existsException) {
        // Check if we can look up the role as it stands already
        tRole, err := iamSvc.GetRole(context.Background(), &iam.GetRoleInput{
            RoleName: aws.String(scenario_BucketListerRoleName),
        })
        if err != nil {
            // Told it already exists, but now it's gone.
            panic("Couldn't find seemingly extant role: " + err.Error())
        } else {
            bucketListerRoleArn = *tRole.Role.Arn
        }
    } else {
        panic("Couldn't create role! " + err.Error())
    }
} else {
    bucketListerRoleArn = *bucketListerRole.Role.Arn
}

fmt.Printf("## The ARN for the bucket lister role is %s", bucketListerRoleArn)

fmt.Println("## Create policy to allow bucket listing")
bucketAllowPolicy := aws.String(`{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1646154730759",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Effect": "Allow",
            "Resource": "*" } ] }`)

var bucketListerPolicyArn string
bucketListerPolicy, err := iamSvc.CreatePolicy(context.Background(),
&iam.CreatePolicyInput{
    PolicyName:      aws.String(scenario_BucketListerPolicyName),
    PolicyDocument: bucketAllowPolicy,
    Description:     aws.String("Allow user to list their own buckets"),
})

if err != nil {
    var existsException *types.EntityAlreadyExistsException
    if errors.As(err, &existsException) {

        stsClient := sts.NewFromConfig(cfg)
        mCallerId, _ := stsClient.GetCallerIdentity(context.Background(),
&sts.GetCallerIdentityInput{})

        mpolicyArn := fmt.Sprintf("arn:aws:iam::%s:policy/%s", *mCallerId.Account,
scenario_BucketListerPolicyName)

        _, err := iamSvc.GetPolicy(context.Background(), &iam.GetPolicyInput{

```

```

    PolicyArn: &mpolicyArn,
  })
  if err != nil {
    panic("Failed to find policy by arn(" + mpolicyArn + ") -> " + err.Error())
  }
  bucketListerPolicyArn = mpolicyArn

} else {
  panic("Couldn't create policy! " + err.Error())
}
} else {
  bucketListerPolicyArn = *bucketListerPolicy.Policy.Arn
}

fmt.Println("## Attach role policy")

_, err = iamSvc.AttachRolePolicy(context.Background(), &iam.AttachRolePolicyInput{
  PolicyArn: &bucketListerPolicyArn,
  RoleName:  aws.String(scenario_BucketListerRoleName),
})

if err != nil {
  fmt.Println("Couldn't attach policy to role! " + err.Error())
}

fmt.Printf("# attached role policy ")

fmt.Println("## waiting for a few moments for keys to become available")

time.Sleep(10 * time.Second)

// Create an S3 client that acts as the user.
userConfig, err := config.LoadDefaultConfig(context.TODO(),
  // Use credentials created earlier.
  config.WithCredentialsProvider(credentials.StaticCredentialsProvider{
    Value: aws.Credentials{
      AccessKeyID: akId, SecretAccessKey: sakId,
      Source: "Example user creds",
    },
  }))
if err != nil {
  panic("Couldn't create config for new credentials! " + err.Error())
} else {
  creds, err := userConfig.Credentials.Retrieve(context.Background())
  if err != nil {
    panic("Couldn't get credentials for our new config, something is wrong: " +
      err.Error())
  }
  fmt.Printf("-> config creds are %s %s\n", creds.AccessKeyID,
    creds.SecretAccessKey)
}

s3Client := s3.NewFromConfig(userConfig)
// Attempt to list buckets.
_, err = s3Client.ListBuckets(context.Background(), &s3.ListBucketsInput{})

if err == nil {
  fmt.Println("Call to s3:ListBuckets was not denied (unexpected!)")
} else {
  var oe smithy.APIError
  if errors.As(err, &oe) && (oe.ErrorCode() == "AccessDenied") {
    fmt.Println("Couldn't list buckets (expected!)")
  } else {
    panic("unexpected error: " + err.Error())
  }
}
}

```

```

stsClient := sts.NewFromConfig(userConfig)
roleCreds := stscreds.NewAssumeRoleProvider(stsClient, bucketListerRoleArn)

tmpCreds, err := roleCreds.Retrieve(context.Background())
if err != nil {
    fmt.Println("couldn't get role creds: " + err.Error())
} else {
    fmt.Printf("role creds: %s %v\n\n", tmpCreds.AccessKeyID, tmpCreds.Expires)
}

roleConfig, err := config.LoadDefaultConfig(context.Background(),
    config.WithCredentialsProvider(roleCreds),
)
if err != nil {
    panic("Couldn't create config with assumed role! " + err.Error())
}

roleS3client := s3.NewFromConfig(roleConfig)

mybuckets, err := roleS3client.ListBuckets(context.Background(),
    &s3.ListBucketsInput{})

if err != nil {
    panic("Couldn't list buckets while assuming role that allows this: " +
err.Error())
} else {
    fmt.Println("Buckets owned by the user....")
    for _, bucket := range mybuckets.Buckets {
        fmt.Printf("%s -> %s\n", *bucket.Name, bucket.CreationDate)
    }
}

// ---- Clean up ----

// Delete the user's access keys.

fmt.Println("cleanup: Delete created access key")
_, _ = iamSvc.DeleteAccessKey(context.Background(), &iam.DeleteAccessKeyInput{
    AccessKeyId: &akId,
    UserName:    aws.String(scenario_UserName),
})

// Delete the user.
fmt.Println("cleanup: delete the user we created")
_, err = iamSvc.DeleteUser(context.Background(), &iam.DeleteUserInput{UserName:
aws.String(scenario_UserName)})
if err != nil {
    fmt.Println("Couldn't delete user! " + err.Error())
}

// Detach the role policy.

fmt.Println("cleanup: Detach the policy from the role")
_, err = iamSvc.DetachRolePolicy(context.Background(), &iam.DetachRolePolicyInput{
    RoleName:    aws.String(scenario_BucketListerRoleName),
    PolicyArn:   &bucketListerPolicyArn,
})

if err != nil {
    fmt.Println("Couldn't detach role policy from role " + err.Error())
}

// Delete the role.
fmt.Println("cleanup: Remove the role")
_, err = iamSvc.DeleteRole(context.Background(), &iam.DeleteRoleInput{

```

```

    RoleName: aws.String(scenario_BucketListerRoleName),
  })
  if err != nil {
    fmt.Println("Couldn't delete role! " + err.Error())
  }

  // Delete the policy.
  fmt.Println("cleanup: delete the policy")
  _, err = iamSvc.DeletePolicy(context.Background(), &iam.DeletePolicyInput{
    PolicyArn: &bucketListerPolicyArn,
  })
  if err != nil {
    fmt.Println("couldn't delete policy!")
  }

  fmt.Println("done!")
}

```

- 有关 API 详细信息，请参阅《Amazon SDK for Go API 参考》中的以下主题。

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

Java

SDK for Java 2.x

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建包装 IAM 用户操作的函数。

```

public class IAMScenario {

    public static final String PolicyDocument =
        "{" +
            "  \"Version\": \"2012-10-17\", " +
            "  \"Statement\": [ " +
            "    { " +
            "      \"Effect\": \"Allow\", " +
            "      \"Action\": [ " +
            "        \"s3:*\" " +
            "      ], " +
            "      \"Resource\": \"*\" " +
            "    } " +
            "  ] " +
            "}"

```

```

public static void main(String[] args) throws Exception {

    final String usage = "\n" +
        "Usage:\n" +
        "    <username> <policyName> <roleName> <roleSessionName>
<fileLocation> <bucketName> \n\n" +
        "Where:\n" +
        "    username - The name of the IAM user to create. \n\n" +
        "    policyName - The name of the policy to create. \n\n" +
        "    roleName - The name of the role to create. \n\n" +
        "    roleSessionName - The name of the session required for the
assumeRole operation. \n\n" +
        "    fileLocation - The file location to the JSON required to create
the role (see Readme). \n\n" +
        "    bucketName - The name of the Amazon S3 bucket from which objects
are read. \n\n" ;

    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }

    String userName = args[0];
    String policyName = args[1];
    String roleName = args[2];
    String roleSessionName = args[3];
    String fileLocation = args[4];
    String bucketName = args[5];

    Region region = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    // Create the IAM user.
    Boolean createUser = createIAMUser(iam, userName);

    if (createUser) {
        System.out.println(userName + " was successfully created.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully
created.");
        String roleArn = createIAMRole(iam, roleName, fileLocation);
        System.out.println(roleArn + " was successfully created.");
        attachIAMRolePolicy(iam, roleName, polArn);

        System.out.println("*** Wait for 1 MIN so the resource is available");
        TimeUnit.MINUTES.sleep(1);
        assumeGivenRole(roleArn, roleSessionName, bucketName);

        System.out.println("*** Getting ready to delete the AWS resources");
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println("This IAM Scenario has successfully completed");
    } else {
        System.out.println(userName + " was not successfully created.");
    }
}

public static Boolean createIAMUser(IamClient iam, String username ) {

    try {
        // Create an IamWaiter object
        IamWaiter iamWaiter = iam.waiter();
    }
}

```

```

        CreateUserRequest request = CreateUserRequest.builder()
            .userName(username)
            .build();

        // Wait until the user is created.
        CreateUserResponse response = iam.createUser(request);
        GetUserRequest userRequest = GetUserRequest.builder()
            .userName(response.user().userName())
            .build();

        WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);

waitUntilUserExists.matched().response().ifPresent(System.out::println);
        return true;

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return false;
}

public static String createIAMRole(IamClient iam, String rolename, String
fileLocation ) throws Exception {

    try {
        JSONObject jsonObject = (JSONObject) readJsonSimpleDemo(fileLocation);
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(jsonObject.toJSONString())
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        System.out.println("The ARN of the role is "+response.role().arn());
        return response.role().arn();

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

public static String createIAMPolicy(IamClient iam, String policyName ) {

    try {
        // Create an IamWaiter object.
        IamWaiter iamWaiter = iam.waiter();
        CreatePolicyRequest request = CreatePolicyRequest.builder()
            .policyName(policyName)
            .policyDocument(PolicyDocument).build();

        CreatePolicyResponse response = iam.createPolicy(request);

        // Wait until the policy is created.
        GetPolicyRequest polRequest = GetPolicyRequest.builder()
            .policyArn(response.policy().arn())
            .build();

        WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
        return response.policy().arn();
    }
}

```



```

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

public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn ) {

    try {
        ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
            .roleName(roleName)
            .build();

        ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
        List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

        String polArn;
        for (AttachedPolicy policy: attachedPolicies) {
            polArn = policy.policyArn();
            if (polArn.compareTo(policyArn)==0) {
                System.out.println(roleName + " policy is already attached to
this role.");
                return;
            }
        }

        AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
            .roleName(roleName)
            .policyArn(policyArn)
            .build();

        iam.attachRolePolicy(attachRequest);
        System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

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

// Invoke an Amazon S3 operation using the Assumed Role.
public static void assumeGivenRole(String roleArn, String roleSessionName,
String bucketName) {

    StsClient stsClient = StsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
            .roleArn(roleArn)
            .roleSessionName(roleSessionName)
            .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials myCreds = roleResponse.credentials();
        String key = myCreds.accessKeyId();
        String secKey = myCreds.secretAccessKey();
        String secToken = myCreds.sessionToken();
    }
}

```

```

        // List all objects in an Amazon S3 bucket using the temp creds.
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()

.credentialsProvider(StaticCredentialsProvider.create(AwsSessionCredentials.create(key,
secKey, secToken)))
        .region(region)
        .build();

        System.out.println("Created a S3Client using temp credentials.");
        System.out.println("Listing objects in "+bucketName);
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.println("The name of the key is " + myValue.key());
            System.out.println("The owner is " + myValue.owner());
        }

    } catch (StsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteRole(IamClient iam, String roleName, String polArn) {

    try {
        // First the policy needs to be detached.
        DetachRolePolicyRequest rolePolicyRequest =
        DetachRolePolicyRequest.builder()
            .policyArn(polArn)
            .roleName(roleName)
            .build();

        iam.detachRolePolicy(rolePolicyRequest);

        // Delete the policy.
        DeletePolicyRequest request = DeletePolicyRequest.builder()
            .policyArn(polArn)
            .build();

        iam.deletePolicy(request);
        System.out.println("*** Successfully deleted "+polArn);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
            .roleName(roleName)
            .build();

        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " +roleName);

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

public static void deleteIAMUser(IamClient iam, String userName) {

    try {

```

```
        DeleteUserRequest request = DeleteUserRequest.builder()
            .userName(userName)
            .build();

        iam.deleteUser(request);
        System.out.println("*** Successfully deleted " + userName);

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

public static Object readJsonSimpleDemo(String filename) throws Exception {
    FileReader reader = new FileReader(filename);
    JSONParser jsonParser = new JSONParser();
    return jsonParser.parse(reader);
}
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的以下主题。

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
// Create service client module using ES6 syntax.
import { IAMClient } from "@aws-sdk/client-iam";
// Set the AWS Region.
export const REGION = "REGION"; // For example, "us-east-1".
// Create an Amazon S3 service client object.
export const iamClient = new IAMClient({ region: REGION });
```

创建 IAM 用户和授予列出 Amazon S3 存储桶的权限的角色。用户仅具有代入该角色的权限。代入该角色后，使用临时凭证列出该账户的存储桶。

```
// Import required AWS SDK clients and commands for Node.js.
```

```
import { iamClient, REGION } from "../libs/iamClient.js"; // Helper function that
creates an IAM service client module.
import {
  CreateUserCommand,
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  AttachUserPolicyCommand,
  DeleteAccessKeyCommand,
  DeleteUserCommand,
  DeleteRoleCommand,
  DeletePolicyCommand,
  DetachUserPolicyCommand,
  DetachRolePolicyCommand,
} from "@aws-sdk/client-iam";
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";

if (process.argv.length < 6) {
  console.log(
    "Usage: node iam_basics.js <user name> <s3 policy name> <role name> <assume"
    "policy name>\n" +
    "Example: node iam_basics.js test-user my-s3-policy my-iam-role my-assume-"
    "role"
  );
}
// Set the parameters.
const region = REGION;
const userName = process.argv[2];
const s3_policy_name = process.argv[3];
const role_name = process.argv[4];
const assume_policy_name = process.argv[5];

// Helper function to delay running the code while the AWS service calls wait for
responses.
function wait(ms) {
  var start = Date.now();
  var end = start
  while (end < start + ms){
    end = Date.now()
  }
}

export const run = async (
  userName,
  s3_policy_name,
  role_name,
  assume_policy_name
) => {
  try {
    // Create a new user.
    const user_params = { UserName: userName };
    console.log("\nCreating a user name " + user_params.UserName + "...\\n");
    const data = await iamClient.send(
      new CreateUserCommand({ UserName: userName })
    );
    const user_arn = data.User.Arn;
    const user_name = data.User.UserName;
    console.log(
      "User with name" + user_name + " and ARN " + user_arn + " created."
    );
  }
  try {
    // Create access keys for the new user.
    console.log(
      "\nCreating access keys for " + user_params.UserName + "...\\n"
    );
  }
}
```

```

);
const access_key_params = { UserName: user_name };
const data = await iamClient.send(
    new CreateAccessKeyCommand(access_key_params)
);
console.log("Success. Access key created: ", data.AccessKey.AccessKeyId);
var myAccessKey = data.AccessKey.AccessKeyId;
var mySecretAccessKey = data.AccessKey.SecretAccessKey;

try {
    // Attempt to list S3 buckets.
    console.log(
        "\nWaiting 10 seconds for user and access keys to be created...\n"
    );
    wait(10000);
    console.log(
        "\nAttempt to list S3 buckets with the new user (without permissions)...\n"
    );
    // Use the credentials for the new user that you created.
    var user_creds = {
        accessKeyId: myAccessKey,
        secretAccessKey: mySecretAccessKey,
    };
    const s3Client = new S3Client({
        credentials: user_creds,
        region: region,
    });
    await s3Client.send(new ListBucketsCommand({}));
} catch (err) {
    console.log(
        "Error. As expected the new user has no permissions to list buckets. ",
        err.stack
    );
    console.log(
        "\nCreating policy to allow the new user to list all buckets, and to\nassume an STS role...\n"
    );
    const myManagedPolicy = {
        Version: "2012-10-17",
        Statement: [
            {
                Effect: "Allow",
                Action: ["s3:ListAllMyBuckets", "sts:AssumeRole"],
                Resource: "*",
            },
        ],
    };
    const policy_params = {
        PolicyDocument: JSON.stringify(myManagedPolicy),
        PolicyName: s3_policy_name, // Name of the new policy.
    };
    const data = await iamClient.send(
        new CreatePolicyCommand(policy_params)
    );
    console.log(
        "Success. Policy created that allows listing of all S3 buckets.\n" +
        "Policy ARN: " +
        data.Policy.Arn +
        "\n" +
        "Policy name: " +
        data.Policy.PolicyName +
        "\n"
    );

    var s3_policy_arn = data.Policy.Arn;

```

```

    try {
      console.log(
        "\nCreating a role with a trust policy that lets the user assume the
role....\n"
      );

      const role_json = {
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: {
              AWS: user_arn, // The ARN of the user.
            },
            Action: "sts:AssumeRole",
          },
        ],
      };
      const myJson = JSON.stringify(role_json);

      const role_params = {
        AssumeRolePolicyDocument: myJson, // Trust relationship policy
        Path: "/",
        RoleName: role_name // The name of the new role.
      };
      const data = await iamClient.send(new CreateRoleCommand(role_params));
      console.log("Success. Role created. Role Arn: ", data.Role.Arn);
      const role_arn = data.Role.Arn;
      try {
        console.log(
          "\nAttaching to the role the policy with permissions to list all
buckets....\n"
        );
        const params = {
          PolicyArn: s3_policy_arn,
          RoleName: role_name,
        };
        await iamClient.send(new AttachRolePolicyCommand(params));
        console.log("Success. Policy attached successfully to role.");
        try {
          console.log(
            "\nCreate a policy that enables the user to assume the role ....
\n"
          );
          const myNewPolicy = {
            Version: "2012-10-17",
            Statement: [
              {
                Effect: "Allow",
                Action: ["sts:AssumeRole"],
                Resource: role_arn,
              },
            ],
          };
          const policy_params = {
            PolicyDocument: JSON.stringify(myNewPolicy),
            PolicyName: assume_policy_name,
          };
          const data = await iamClient.send(
            new CreatePolicyCommand(policy_params)
          );
          console.log(
            "Success. Policy created. Policy ARN: " + data.Policy.Arn
          );
        }
      }
    }
  }
}

```

```

const assume_policy_arn = data.Policy.Arn;
try {
  console.log("\nAttaching the policy to the user....\n");
  const attach_policy_to_user_params = {
    PolicyArn: assume_policy_arn,
    UserName: user_name,
  };
  await iamClient.send(
    new AttachUserPolicyCommand(attach_policy_to_user_params)
  );
  console.log(
    "\nWaiting 10 seconds for policy to be attached...\n"
  );
  wait(10000);
  console.log(
    "Success. Policy attached to user " + user_name + "."
  );
  try {
    console.log(
      "\nAssume for the user the role with permission to list all
buckets....\n"
    );
    const assume_role_params = {
      RoleArn: role_arn, //ARN_OF_ROLE_TO_ASSUME
      RoleSessionName: "session1",
      DurationSeconds: 900,
    };
    // Create an AWS STS client with the credentials for the user.
    Remember, the user has permissions to assume roles using AWS STS.
    const stsClientWithUsersCreds = new STSClient({
      credentials: user_creds,
      region: REGION,
    });

    const data = await stsClientWithUsersCreds.send(
      new AssumeRoleCommand(assume_role_params)
    );
    console.log(
      "Success assuming role. Access key id is " +
      data.Credentials.AccessKeyId +
      "\n" +
      "Secret access key is " +
      data.Credentials.SecretAccessKey
    );

    const newAccessKey = data.Credentials.AccessKeyId;
    const newSecretAccessKey = data.Credentials.SecretAccessKey;

    console.log(
      "\nWaiting 10 seconds for the user to assume the role with
permission to list all buckets...\n"
    );
    wait(10000);
    // Set the parameters for the temporary credentials. This grants
    permission to list S3 buckets.
    var new_role_creds = {
      accessKeyId: newAccessKey,
      secretAccessKey: newSecretAccessKey,
      sessionToken: data.Credentials.SessionToken,
    };
    try {
      console.log(
        "Listing the S3 buckets using the credentials of the
assumed role... \n"
      );
    }

```



```
// Create an S3 client with the temporary credentials.
const s3ClientWithNewCreds = new S3Client({
  credentials: new_role_creds,
  region: REGION,
});
const data = await s3ClientWithNewCreds.send(
  new ListBucketsCommand({})
);
console.log("Success. Your S3 buckets are:", data.Buckets);
try {
  console.log(
    "Detaching s3 policy from user " + userName + " ... \n"
  );
  await iamClient.send(
    new DetachUserPolicyCommand({
      PolicyArn: assume_policy_arn,
      UserName: userName,
    })
  );
  console.log("Success, S3 policy detached from user.");
  try {
    console.log(
      "Detaching role policy from " + role_name + " ... \n"
    );
    await iamClient.send(
      new DetachRolePolicyCommand({
        PolicyArn: s3_policy_arn,
        RoleName: role_name,
      })
    );
    console.log(
      "Success, assume policy detached from role."
    );
  }
  try {
    console.log("Deleting s3 policy ... \n");
    await iamClient.send(
      new DeletePolicyCommand({
        PolicyArn: s3_policy_arn,
      })
    );
    console.log("Success, S3 policy deleted.");
    try {
      console.log("Deleting assume role policy ... \n");
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: assume_policy_arn,
        })
      );
    }
    try {
      console.log("Deleting access keys ... \n");
      await iamClient.send(
        new DeleteAccessKeyCommand({
          UserName: userName,
          AccessKeyId: myAccessKey,
        })
      );
    }
    try {
      console.log(
        "Deleting user " + user_name + " ... \n"
      );
      await iamClient.send(
        new DeleteUserCommand({ UserName: userName })
      );
      console.log("Success, user deleted.");
      try {
        console.log(
```

```

        "Deleting role " + role_name + " ... \n"
    );
    await iamClient.send(
        new DeleteRoleCommand({
            RoleName: role_name,
        })
    );
    console.log("Success, role deleted.");
    return "Run successfully"; // For unit tests.
} catch (err) {
    console.log("Error deleting role .", err);
}
} catch (err) {
    console.log("Error deleting user.", err);
}
} catch (err) {
    console.log("Error deleting access keys.", err);
}
} catch (err) {
    console.log(
        "Error detaching assume role policy from user.",
        err
    );
}
} catch (err) {
    console.log("Error deleting role.", err);
}
} catch (err) {
    console.log("Error deleting user.", err);
}
} catch (err) {
    console.log("Error detaching S3 policy from role.", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error listing S3 buckets.", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error assuming role.", err);
    process.exit(1);
}
} catch (err) {
    console.log(
        "Error adding permissions to user to assume role.",
        err
    );
    process.exit(1);
}
} catch (err) {
    console.log("Error assuming role.", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error creating policy. ", err);
    process.exit(1);
}
} catch (err) {
    console.log("Error attaching policy to role.", err);
    process.exit(1);
}
}
} catch (err) {
    console.log("Error creating access keys. ", err);
    process.exit(1);
}
}

```

```
    } catch (err) {  
        console.log("Error creating user. ", err);  
    }  
};  
run(userName, s3_policy_name, role_name, assume_policy_name);
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的以下主题。

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

Kotlin

SDK for Kotlin

Note

这是适用于预览版中功能的预发行文档。本文档随时可能更改。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建包装 IAM 用户操作的函数。

```
suspend fun main(args: Array<String>) {  
  
    val usage = ""  
    Usage:  
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>  
        <bucketName>  
  
    Where:  
        username - The name of the IAM user to create.  
        policyName - The name of the policy to create.  
        roleName - The name of the role to create.  
        roleSessionName - The name of the session required for the assumeRole  
operation.  
        fileLocation - The file location to the JSON required to create the role  
(see Readme).  
        bucketName - The name of the Amazon S3 bucket from which objects are read.  
    ""  
  
    if (args.size != 6) {  
        println(usage)  
        exitProcess(1)  
    }  
  
    val userName = args[0]
```

```

val policyName = args[1]
val roleName = args[2]
val roleSessionName = args[3]
val fileLocation = args[4]
val bucketName = args[5]

createUser(userName)
println("$userName was successfully created.")

val polArn = createPolicy(policyName)
println("The policy $polArn was successfully created.")

val roleArn = createRole(roleName, fileLocation)
println("$roleArn was successfully created.")
attachRolePolicy(roleName, polArn)

println("*** Wait for 1 MIN so the resource is available.")
delay(60000)
assumeGivenRole(roleArn, roleSessionName, bucketName)

println("*** Getting ready to delete the AWS resources.")
deleteRole(roleName, polArn)
deleteUser(userName)
println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {

    val request = CreateUserRequest {
        userName = usernameVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}

suspend fun createPolicy(policyNameVal: String?): String {

    val policyDocumentValue: String = "{" +
        "  \"Version\": \"2012-10-17\", " +
        "  \"Statement\": [ " +
        "    { " +
        "      \"Effect\": \"Allow\", " +
        "      \"Action\": [ " +
        "        \"s3:*\" " +
        "      ], " +
        "      \"Resource\": \"*\" " +
        "    } " +
        "  ] " +
        "}"

    val request = CreatePolicyRequest {
        policyName = policyNameVal
        policyDocument = policyDocumentValue
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(rolenameVal: String?, fileLocation: String?): String? {

```

```

val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

val request = CreateRoleRequest {
    roleName = rolenameVal
    assumeRolePolicyDocument = jsonObject.toJSONString()
    description = "Created using the AWS SDK for Kotlin"
}

IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
    val response = iamClient.createRole(request)
    return response.role?.arn
}
}

suspend fun attachRolePolicy(roleNameVal: String, policyArnVal: String) {

    val request = ListAttachedRolePoliciesRequest {
        roleName = roleNameVal
    }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1)
                return
        }

        val policyRequest = AttachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role $roleNameVal")
    }
}

fun checkMyList(attachedPolicies: List<AttachedPolicy>, policyArnVal: String): Int
{
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(roleArnVal: String?, roleSessionNameVal: String?,
    bucketName: String) {

    val stsClient = StsClient {
        region = "us-east-1"
    }

    val roleRequest = AssumeRoleRequest {
        roleArn = roleArnVal
        roleSessionName = roleSessionNameVal
    }
}

```

```

val roleResponse = stsClient.assumeRole(roleRequest)
val myCreds = roleResponse.credentials
val key = myCreds?.accessKeyId
val secKey = myCreds?.secretAccessKey
val secToken = myCreds?.sessionToken

val staticCredentials = StaticCredentialsProvider {
    accessKeyId = key
    secretAccessKey = secKey
    sessionToken = secToken
}

// List all objects in an Amazon S3 bucket using the temp creds.
val s3 = S3Client {
    credentialsProvider = staticCredentials
    region = "us-east-1"
}

println("Created a S3Client using temp credentials.")
println("Listing objects in $bucketName")

val listObjects = ListObjectsRequest {
    bucket = bucketName
}

val response = s3.listObjects(listObjects)
response.contents?.forEach { myObject ->
    println("The name of the key is ${myObject.key}")
    println("The owner is ${myObject.owner}")
}
}

suspend fun deleteRole(roleNameVal: String, polArn: String) {

    val iam = IamClient { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest = DetachRolePolicyRequest {
        policyArn = polArn
        roleName = roleNameVal
    }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request = DeletePolicyRequest {
        policyArn = polArn
    }

    iam.deletePolicy(request)
    println("**** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest = DeleteRoleRequest {
        roleName = roleNameVal
    }

    iam.deleteRole(roleRequest)
    println("**** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient { region = "AWS_GLOBAL" }
    val request = DeleteUserRequest {
        userName = userNameVal
    }
}

```

```
    }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

• 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的以下主题。

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

PHP

SDK for PHP

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IamService;

echo("-----\n");
print("Welcome to the Amazon IAM getting started demo using PHP!\n");
echo("-----\n");
$uuid = uniqid();
$service = new IamService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
    \"Version\": \"2012-10-17\",
```



```

        \Statement\": [{
            \Effect\": \"Allow\",
            \Principal\": {\"AWS\": \"${user['Arn']}\"},
            \Action\": \"sts:AssumeRole\"
        }]
    }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
    $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
    \Version\": \"2012-10-17\",
    \Statement\": [{
        \Effect\": \"Allow\",
        \Action\": \"s3:ListAllMyBuckets\",
        \Resource\": \"arn:aws:s3:*:*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
    $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
    $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
    \Version\": \"2012-10-17\",
    \Statement\": [{
        \Effect\": \"Allow\",
        \Action\": \"sts:AssumeRole\",
        \Resource\": \"${assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
    $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
    'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
    'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
    'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\n";
}

```

```
$service->detachRolePolicy($assumeRoleRole['RoleName'],
    $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

• 有关 API 详细信息，请参阅《Amazon SDK for PHP API 参考》中的以下主题。

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建 IAM 用户和授予列出 Amazon S3 存储桶的权限的角色。用户仅具有代入该角色的权限。代入该角色后，使用临时凭证列出该账户的存储桶。

```
import json
import sys
import time
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError

def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
        print('.', end='')
        sys.stdout.flush()
    print()

def setup(iam_resource):
    """
```

```

Creates a new user with no permissions.
Creates an access key pair for the user.
Creates a role with a policy that lets the user assume the role.
Creates a policy that allows listing Amazon S3 buckets.
Attaches the policy to the role.
Creates an inline policy for the user that lets the user assume the role.

:param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
                      that has permissions to create users, roles, and policies
                      in the account.
:return: The newly created user, user key, and role.
"""
try:
    user = iam_resource.create_user(UserName=f'demo-user-{uuid4()}')
    print(f"Created user {user.name}.")
except ClientError as error:
    print(f"Couldn't create a user for the demo. Here's why: "
          f"{error.response['Error']['Message']}")
    raise

try:
    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")
except ClientError as error:
    print(f"Couldn't create access keys for user {user.name}. Here's why: "
          f"{error.response['Error']['Message']}")
    raise

print(f"Wait for user to be ready.", end='')
progress_bar(10)

try:
    role = iam_resource.create_role(
        RoleName=f'demo-role-{uuid4()}',
        AssumeRolePolicyDocument=json.dumps({
            'Version': '2012-10-17',
            'Statement': [{
                'Effect': 'Allow',
                'Principal': {'AWS': user.arn},
                'Action': 'sts:AssumeRole'}]))
    print(f"Created role {role.name}.")
except ClientError as error:
    print(f"Couldn't create a role for the demo. Here's why: "
          f"{error.response['Error']['Message']}")
    raise

try:
    policy = iam_resource.create_policy(
        PolicyName=f'demo-policy-{uuid4()}',
        PolicyDocument=json.dumps({
            'Version': '2012-10-17',
            'Statement': [{
                'Effect': 'Allow',
                'Action': 's3:ListAllMyBuckets',
                'Resource': 'arn:aws:s3:::*'}]))
    role.attach_policy(PolicyArn=policy.arn)
    print(f"Created policy {policy.policy_name} and attached it to the role.")
except ClientError as error:
    print(f"Couldn't create a policy and attach it to role {role.name}. Here's
why: "
          f"{error.response['Error']['Message']}")
    raise

try:
    user.create_policy(
        PolicyName=f'demo-user-policy-{uuid4()}',

```

```

        PolicyDocument=json.dumps({
            'Version': '2012-10-17',
            'Statement': [{
                'Effect': 'Allow',
                'Action': 'sts:AssumeRole',
                'Resource': role.arn}}])
        print(f"Created an inline policy for {user.name} that lets the user assume
"
            f"the role.")
    except ClientError as error:
        print(f"Couldn't create an inline policy for user {user.name}. Here's why:
"
            f"{error.response['Error']['Message']}")
        raise

    print("Give AWS time to propagate these new resources and connections.",
end='')
    progress_bar(10)

    return user, user_key, role

def show_access_denied_without_role(user_key):
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
        have permission to list buckets in the account.
    """
    print(f"Try to list buckets without first assuming the role.")
    s3_denied_resource = boto3.resource(
        's3', aws_access_key_id=user_key.id, aws_secret_access_key=user_key.secret)
    try:
        for bucket in s3_denied_resource.buckets.all():
            print(bucket.name)
        raise RuntimeError("Expected to get AccessDenied error when listing
buckets!")
    except ClientError as error:
        if error.response['Error']['Code'] == 'AccessDenied':
            print("Attempt to list buckets with no permissions: AccessDenied.")
        else:
            raise

def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
    Assumes a role that grants permission to list the Amazon S3 buckets in the
    account.
    Uses the temporary credentials from the role to list the buckets that are owned
    by the assumed role's account.

    :param user_key: The access key of a user that has permission to assume the
    role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
        grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
        'sts', aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret)
    try:
        response = sts_client.assume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name)
        temp_credentials = response['Credentials']
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:

```

```

        print(f"Couldn't assume role {assume_role_arn}. Here's why: "
              f"{error.response['Error']['Message']}")
        raise

    # Create an S3 resource that can access the account with the temporary
    credentials.
    s3_resource = boto3.resource(
        's3',
        aws_access_key_id=temp_credentials['AccessKeyId'],
        aws_secret_access_key=temp_credentials['SecretAccessKey'],
        aws_session_token=temp_credentials['SessionToken'])
    print(f"Listing buckets for the assumed role's account:")
    try:
        for bucket in s3_resource.buckets.all():
            print(bucket.name)
    except ClientError as error:
        print(f"Couldn't list buckets for the account. Here's why: "
              f"{error.response['Error']['Message']}")
        raise

def teardown(user, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print("Couldn't detach policy, delete policy, or delete role. Here's why: "
              f"{error.response['Error']['Message']}")
        raise

    try:
        for user_pol in user.policies.all():
            user_pol.delete()
            print("Deleted inline user policy.")
        for key in user.access_keys.all():
            key.delete()
            print("Deleted user's access key.")
        user.delete()
        print(f"Deleted {user.name}.")
    except ClientError as error:
        print("Couldn't delete user policy or delete user. Here's why: "
              f"{error.response['Error']['Message']}")

def usage_demo():
    """Drives the demonstration."""
    print('-'*88)
    print(f"Welcome to the IAM create user and assume role demo.")
    print('-'*88)
    iam_resource = boto3.resource('iam')
    user = None
    role = None
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")

```

```
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn, 'AssumeRoleDemoSession')
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
            teardown(user, role)
        print("Thanks for watching!")

if __name__ == '__main__':
    usage_demo()
```

• 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的以下主。

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
require "aws-sdk-iam"
require "aws-sdk-s3"

# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_resource

  # @param iam_resource [Aws::IAM::Resource] An AWS IAM resource.
  def initialize(iam_resource)
    @iam_resource = iam_resource
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
    puts("Give AWS time to propagate resources...")
    sleep(duration)
  end

  # Creates a user.
  #
```

```
# @param user_name [String] The name to give the user.
# @return [Aws::IAM::User] The newly created user.
def create_user(user_name)
  user = @iam_resource.create_user(user_name: user_name)
  puts("Created demo user named #{user.name}.")
rescue Aws::Errors::ServiceError => e
  puts("Tried and failed to create demo user.")
  puts("\t#{e.code}: #{e.message}")
  puts("\nCan't continue the demo without a user!")
  raise
else
  user
end

# Creates an access key for a user.
#
# @param user [Aws::IAM::User] The user that owns the key.
# @return [Aws::IAM::AccessKeyPair] The newly created access key.
def create_access_key_pair(user)
  user_key = user.create_access_key_pair
  puts("Created access key pair for user.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't create access keys for user #{user.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  user_key
end

# Creates a role that can be assumed by a user.
#
# @param role_name [String] The name to give the role.
# @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
# @return [Aws::IAM::Role] The newly created role.
def create_role(role_name, user)
  role = @iam_resource.create_role(
    role_name: role_name,
    assume_role_policy_document: {
      Version: "2012-10-17",
      Statement: [{
        Effect: "Allow",
        Principal: {'AWS': user.arn},
        Action: "sts:AssumeRole"
      }]
    }.to_json)
  puts("Created role #{role.name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't create a role for the demo. Here's why: ")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  role
end

# Creates a policy that grants permission to list S3 buckets in the account, and
# then attaches the policy to a role.
#
# @param policy_name [String] The name to give the policy.
# @param role [Aws::IAM::Role] The role that the policy is attached to.
# @return [Aws::IAM::Policy] The newly created policy.
def create_and_attach_role_policy(policy_name, role)
  policy = @iam_resource.create_policy(
    policy_name: policy_name,
    policy_document: {
      Version: "2012-10-17",
```



```

        Statement: [{
          Effect: "Allow",
          Action: "s3:ListAllMyBuckets",
          Resource: "arn:aws:s3:::*"
        }]
      }.to_json)
      role.attach_policy(policy_arn: policy.arn)
      puts("Created policy #{policy.policy_name} and attached it to role
#{role.name}.")
      rescue Aws::Errors::ServiceError => e
        puts("Couldn't create a policy and attach it to role #{role.name}. Here's why:
")
        puts("\t#{e.code}: #{e.message}")
        raise
      else
        policy
      end

# Creates an inline policy for a user that lets the user assume a role.
#
# @param policy_name [String] The name to give the policy.
# @param user [Aws::IAM::User] The user that owns the policy.
# @param role [Aws::IAM::Role] The role that can be assumed.
# @return [Aws::IAM::UserPolicy] The newly created policy.
def create_user_policy(policy_name, user, role)
  policy = user.create_policy(
    policy_name: policy_name,
    policy_document: {
      Version: "2012-10-17",
      Statement: [{
        Effect: "Allow",
        Action: "sts:AssumeRole",
        Resource: role.arn
      }]
    }.to_json)
  puts("Created an inline policy for #{user.name} that lets the user assume role
#{role.name}.")
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't create an inline policy for user #{user.name}. Here's why: ")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    policy
  end

# Creates an Amazon S3 resource with specified credentials. This is separated
into a
# factory function so that it can be mocked for unit testing.
#
# @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
def create_s3_resource(credentials)
  Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
end

# Lists the S3 buckets for the account, using the specified Amazon S3 resource.
# Because the resource uses credentials with limited access, it may not be able
to
# list the S3 buckets.
#
# @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
def list_buckets(s3_resource)
  count = 10
  s3_resource.buckets.each do |bucket|
    puts "\t#{bucket.name}"
    count -= 1
  end
end

```

```

        break if count.zero?
      end
    rescue Aws::Errors::ServiceError => e
      if e.code == "AccessDenied"
        puts("Attempt to list buckets with no permissions: AccessDenied.")
      else
        puts("Couldn't list buckets for the account. Here's why: ")
        puts("\t#{e.code}: #{e.message}")
        raise
      end
    end
  end

  # Creates an AWS Security Token Service (AWS STS) client with specified
  # credentials.
  # This is separated into a factory function so that it can be mocked for unit
  # testing.
  #
  # @param key_id [String] The ID of the access key used by the STS client.
  # @param key_secret [String] The secret part of the access key used by the STS
  # client.
  def create_sts_client(key_id, key_secret)
    Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
  end

  # Gets temporary credentials that can be used to assume a role.
  #
  # @param role_arn [String] The ARN of the role that is assumed when these
  # credentials
  #
  # are used.
  # @param sts_client [Aws::STS::Client] An AWS STS client.
  # @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
  # the role.
  def assume_role(role_arn, sts_client)
    credentials = Aws::AssumeRoleCredentials.new(
      client: sts_client,
      role_arn: role_arn,
      role_session_name: "create-use-assume-role-scenario"
    )
    puts("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
  end

  # Deletes a role. If the role has policies attached, they are detached and
  # deleted before the role is deleted.
  #
  # @param role [Aws::IAM::Role] The role to delete.
  def delete_role(role)
    role.attached_policies.each do |policy|
      name = policy.policy_name
      policy.detach_role(role_name: role.name)
      policy.delete
      puts("Deleted policy #{name}.")
    end
    name = role.name
    role.delete
    puts("Deleted role #{name}.")
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't detach policies and delete role #{role.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes a user. If the user has inline policies or access keys, they are
  # deleted
  # before the user is deleted.
  #

```

```
# @param user [Aws::IAM::User] The user to delete.
def delete_user(user)
  user.policies.each do |policy|
    name = policy.name
    policy.delete
    puts("Deleted user policy #{name}.")
  end
  user.access_keys.each do |key|
    key.delete
    puts("Deleted access key for user #{user.name}.")
  end
  name = user.name
  user.delete
  puts("Deleted user #{name}.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't detach policies and delete user #{user.name}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the IAM create a user and assume a role demo!")
  puts("-" * 88)

  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-#{Random.uuid}",
  role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user, role)
  scenario.wait(10)
  puts("Try to list buckets with credentials for a user who has no permissions.")
  puts("Expect AccessDenied from this call.")
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.id,
  user_key.secret)))
  puts("Now, assume the role that grants permission.")
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.id, user_key.secret))
  puts("Here are your buckets:")
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.name}' and attached policies.")
  scenario.delete_role(role)
  puts("Deleting user '#{user.name}', policies, and keys.")
  scenario.delete_user(user)

  puts("Thanks for watching!")
  puts("-" * 88)
rescue Aws::Errors::ServiceError => e
  puts("Something went wrong with the demo.")
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Resource.new)) if
$PROGRAM_NAME == __FILE__
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的以下主题。
 - [AttachRolePolicy](#)
 - [CreateAccessKey](#)
 - [CreatePolicy](#)

- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

Rust

SDK for Rust

Note

本文档适用于预览版中的软件开发工具包。软件开发工具包可能随时发生变化，不应在生产环境中使用。

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{Client as iamClient, Credentials as iamCredentials};
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use aws_types::region::Region;
use std::borrow::Borrow;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document) =
        initialize_variables().await;

    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{:?}", e);
    };

    Ok(())
}

async fn initialize_variables() -> (iamClient, String, String, String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-west-2"));

    let shared_config =
        aws_config::from_env().region(region_provider).load().await;
    let client = iamClient::new(&shared_config);
```

```

let uuid = Uuid::new_v4().to_string();

let list_all_buckets_policy_document = "{
  \"Version\": \"2012-10-17\",
  \"Statement\": [{
    \"Effect\": \"Allow\",
    \"Action\": \"s3:ListAllMyBuckets\",
    \"Resource\": \"arn:aws:s3:*:*\"}]
}"
.to_string();
let inline_policy_document = "{
  \"Version\": \"2012-10-17\",
  \"Statement\": [{
    \"Effect\": \"Allow\",
    \"Action\": \"sts:AssumeRole\",
    \"Resource\": \"{}\"}]
}"
.to_string();

(
  client,
  uuid,
  list_all_buckets_policy_document,
  inline_policy_document,
)

}

async fn run_iam_operations(
  client: iamClient,
  uuid: String,
  list_all_buckets_policy_document: String,
  inline_policy_document: String,
) -> Result<(), iamError> {
  let user = iam_service::create_user(&client, &format!("{}", "iam_demo_user_",
  uuid)).await?;
  println!(
    "Created the user with the name: {}",
    user.user_name.as_ref().unwrap()
  );
  let key = iam_service::create_access_key(&client,
  user.user_name.as_ref().unwrap()).await?;

  let assume_role_policy_document = "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
      \"Effect\": \"Allow\",
      \"Principal\": {\"AWS\": \"{}\"},
      \"Action\": \"sts:AssumeRole\"
    }]
  }"
  .to_string()
  .replace!("{}", user.arn.as_ref().unwrap());

  let assume_role_role = iam_service::create_role(
    &client,
    &format!("{}", "iam_demo_role_", uuid),
    &assume_role_policy_document,
  )
  .await?;
  println!(
    "Created the role with the ARN: {}",
    assume_role_role.arn.as_ref().unwrap()
  );

  let list_all_buckets_policy = iam_service::create_policy(
    &client,

```

```

        &format!("{}", "iam_demo_policy_", uuid),
        &list_all_buckets_policy_document,
    )
    .await?;
println!(
    "Created policy: {}",
    list_all_buckets_policy.policy_name.as_ref().unwrap()
);

let attach_role_policy_result =
    iam_service::attach_role_policy(&client, &assume_role_role,
    &list_all_buckets_policy)
    .await?;
println!(
    "Attached the policy to the role: {:?}",
    attach_role_policy_result
);

let inline_policy_name = format!("{}", "iam_demo_inline_policy_", uuid);
let inline_policy_document =
    inline_policy_document.replace("{}",
    assume_role_role.arn.as_ref().unwrap());
iam_service::create_user_policy(&client, &user, &inline_policy_name,
    &inline_policy_document)
    .await?;
println!("Created inline policy.");

//First, fail to list the buckets with the user.
let creds = iamCredentials::from_keys(
    key.access_key_id.as_ref().unwrap(),
    key.secret_access_key.as_ref().unwrap(),
    None,
);
let fail_config = aws_config::from_env()
    .credentials_provider(creds.clone())
    .load()
    .await;
println!("Fail config: {:?}", fail_config);
let fail_client: s3Client = s3Client::new(&fail_config);
match fail_client.list_buckets().send().await {
    Ok(e) => {
        println!("This should not run. {:?}", e);
    }
    Err(e) => {
        println!("Successfully failed with error: {:?}", e)
    }
}

let sts_config = aws_config::from_env()
    .credentials_provider(creds.clone())
    .load()
    .await;
let sts_client: stsClient = stsClient::new(&sts_config);
sleep(Duration::from_secs(10)).await;
let assumed_role = sts_client
    .assume_role()
    .role_arn(assume_role_role.arn.as_ref().unwrap())
    .role_session_name(&format!("{}", "iam_demo_assumerole_session_", uuid))
    .send()
    .await;
println!("Assumed role: {:?}", assumed_role);
sleep(Duration::from_secs(10)).await;

let assumed_credentials = iamCredentials::from_keys(
    assumed_role
    .as_ref()

```

```

        .unwrap()
        .credentials
        .as_ref()
        .unwrap()
        .access_key_id
        .as_ref()
        .unwrap(),
    assumed_role
        .as_ref()
        .unwrap()
        .credentials
        .as_ref()
        .unwrap()
        .secret_access_key
        .as_ref()
        .unwrap(),
    assumed_role
        .as_ref()
        .unwrap()
        .credentials
        .as_ref()
        .unwrap()
        .session_token
        .borrow()
        .clone(),
    );

let succeed_config = aws_config::from_env()
    .credentials_provider(assumed_credentials)
    .load()
    .await;
println!("succeed config: {:?}", succeed_config);
let succeed_client: s3Client = s3Client::new(&succeed_config);
sleep(Duration::from_secs(10)).await;
match succeed_client.list_buckets().send().await {
    Ok(_) => {
        println!("This should now run successfully.")
    }
    Err(e) => {
        println!("This should not run. {:?}", e);
        panic!()
    }
}

//Clean up.
iam_service::detach_role_policy(
    &client,
    assume_role_role.role_name.as_ref().unwrap(),
    list_all_buckets_policy.arn.as_ref().unwrap(),
)
.await?;
iam_service::delete_policy(&client, list_all_buckets_policy).await?;
iam_service::delete_role(&client, &assume_role_role).await?;
println!(
    "Deleted role {}",
    assume_role_role.role_name.as_ref().unwrap()
);
iam_service::delete_access_key(&client, &user, &key).await?;
println!("Deleted key for {}", key.user_name.as_ref().unwrap());
iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
println!("Deleted inline user policy: {}", inline_policy_name);
iam_service::delete_user(&client, &user).await?;
println!("Deleted user {}", user.user_name.as_ref().unwrap());

Ok(())
}

```


- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的以下主题。
 - [AttachRolePolicy](#)
 - [CreateAccessKey](#)
 - [CreatePolicy](#)
 - [CreateRole](#)
 - [CreateUser](#)
 - [DeleteAccessKey](#)
 - [DeletePolicy](#)
 - [DeleteRole](#)
 - [DeleteUser](#)
 - [DeleteUserPolicy](#)
 - [DetachRolePolicy](#)
 - [PutUserPolicy](#)

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包创建只读和读写 IAM 用户

以下代码示例显示了如何：

- 创建两个 IAM 用户。
- 附加一项策略，该策略允许其中一个用户获取对象并将其放入 Amazon Simple Storage Service (Amazon S3) 存储桶。
- 附加一项策略，该策略允许另一个用户从该存储桶获取对象。
- 根据用户凭证获取对该存储桶的不同权限。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建包装 IAM 用户操作的函数。

```
import logging
import time

import boto3
from botocore.exceptions import ClientError

import access_key_wrapper
import policy_wrapper

logger = logging.getLogger(__name__)
iam = boto3.resource('iam')

def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.
```

```

:param user_name: The name of the user.
:return: The newly created user.
"""
try:
    user = iam.create_user(UserName=user_name)
    logger.info("Created user %s.", user.name)
except ClientError:
    logger.exception("Couldn't create user %s.", user_name)
    raise
else:
    return user

def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user

def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users

def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
        raise

def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """

```

```
try:
    iam.User(user_name).attach_policy(PolicyArn=policy_arn)
    logger.info("Attached policy %s to user %s.", policy_arn, user_name)
except ClientError:
    logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
user_name)
    raise

def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name)
        raise
```

创建包装 IAM policy 操作的函数。

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource('iam')

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
        form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy
        applies to. This ARN can contain wildcards, such as
        'arn:aws:s3:::my-bucket/*' to allow actions on all objects
        in the bucket named 'my-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": actions,
                "Resource": resource_arn
            }
        ]
    }
    try:
        policy = iam.create_policy(
            PolicyName=name, Description=description,
```

```
        PolicyDocument=json.dumps(policy_doc))
    logger.info("Created policy %s.", policy.arn)
except ClientError:
    logger.exception("Couldn't create policy %s.", name)
    raise
else:
    return policy

def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

创建包装 IAM 访问密钥操作的函数。

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource('iam')

def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name, key_pair.id)
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair

def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """
    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info(
            "Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
```

```
logger.exception("Couldn't delete key %s for %s", key_id, user_name)
raise
```

使用包装器函数创建具有不同策略的用户，然后使用其凭证访问 Amazon S3 存储桶。

```
def usage_demo():
    """
    Shows how to manage users, keys, and policies.
    This demonstration creates two users: one user who can put and get objects in
    an
    Amazon S3 bucket, and another user who can only get objects from the bucket.
    The demo then shows how the users can perform only the actions they are
    permitted
    to perform.
    """
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    print('-'*88)
    print("Welcome to the AWS Identity and Account Management user demo.")
    print('-'*88)
    print("Users can have policies and roles attached to grant them specific "
          "permissions.")
    s3 = boto3.resource('s3')
    bucket = s3.create_bucket(
        Bucket=f'demo-iam-bucket-{time.time_ns()}',
        CreateBucketConfiguration={
            'LocationConstraint': s3.meta.client.meta.region_name
        }
    )
    print(f"Created an Amazon S3 bucket named {bucket.name}.")
    user_read_writer = create_user('demo-iam-read-writer')
    user_reader = create_user('demo-iam-reader')
    print(f"Created two IAM users: {user_read_writer.name} and {user_reader.name}")
    update_user(user_read_writer.name, 'demo-iam-creator')
    update_user(user_reader.name, 'demo-iam-getter')
    users = list_users()
    user_read_writer = next(user for user in users if user.user_id ==
user_read_writer.user_id)
    user_reader = next(user for user in users if user.user_id ==
user_reader.user_id)
    print(f"Changed the names of the users to {user_read_writer.name} "
          f"and {user_reader.name}.")

    read_write_policy = policy_wrapper.create_policy(
        'demo-iam-read-write-policy',
        'Grants rights to create and get an object in the demo bucket.',
        ['s3:PutObject', 's3:GetObject'],
        f'arn:aws:s3::{bucket.name}/*')
    print(f"Created policy {read_write_policy.policy_name} with ARN:
{read_write_policy.arn}")
    print(read_write_policy.description)
    read_policy = policy_wrapper.create_policy(
        'demo-iam-read-policy',
        'Grants rights to get an object from the demo bucket.',
        's3:GetObject',
        f'arn:aws:s3::{bucket.name}/*')
    print(f"Created policy {read_policy.policy_name} with ARN: {read_policy.arn}")
    print(read_policy.description)
    attach_policy(user_read_writer.name, read_write_policy.arn)
    print(f"Attached {read_write_policy.policy_name} to {user_read_writer.name}.")
    attach_policy(user_reader.name, read_policy.arn)
    print(f"Attached {read_policy.policy_name} to {user_reader.name}.")

    user_read_writer_key = access_key_wrapper.create_key(user_read_writer.name)
    print(f"Created access key pair for {user_read_writer.name}")
```

```

user_reader_key = access_key_wrapper.create_key(user_reader.name)
print(f"Created access key pair for {user_reader.name}.")

s3_read_writer_resource = boto3.resource(
    's3',
    aws_access_key_id=user_read_writer_key.id,
    aws_secret_access_key=user_read_writer_key.secret)
demo_object_key = f'object-{time.time_ns()}'
demo_object = None
while demo_object is None:
    try:
        demo_object = s3_read_writer_resource.Bucket(bucket.name).put_object(
            Key=demo_object_key, Body=b'AWS IAM demo object content!')
    except ClientError as error:
        if error.response['Error']['Code'] == 'InvalidAccessKeyId':
            print("Access key not yet available. Waiting...")
            time.sleep(1)
        else:
            raise
print(f"Put {demo_object_key} into {bucket.name} using "
      f"{user_read_writer.name}'s credentials.")

read_writer_object = s3_read_writer_resource.Bucket(
    bucket.name).Object(demo_object_key)
read_writer_content = read_writer_object.get()['Body'].read()
print(f"Got object {read_writer_object.key} using read-writer user's
credentials.")
print(f"Object content: {read_writer_content}")

s3_reader_resource = boto3.resource(
    's3',
    aws_access_key_id=user_reader_key.id,
    aws_secret_access_key=user_reader_key.secret)
demo_content = None
while demo_content is None:
    try:
        demo_object =
s3_reader_resource.Bucket(bucket.name).Object(demo_object_key)
        demo_content = demo_object.get()['Body'].read()
        print(f"Got object {demo_object.key} using reader user's credentials.")
        print(f"Object content: {demo_content}")
    except ClientError as error:
        if error.response['Error']['Code'] == 'InvalidAccessKeyId':
            print("Access key not yet available. Waiting...")
            time.sleep(1)
        else:
            raise

    try:
        demo_object.delete()
    except ClientError as error:
        if error.response['Error']['Code'] == 'AccessDenied':
            print('-'*88)
            print("Tried to delete the object using the reader user's credentials.
"
                  "Got expected AccessDenied error because the reader is not "
                  "allowed to delete objects.")
            print('-'*88)

access_key_wrapper.delete_key(user_reader.name, user_reader_key.id)
detach_policy(user_reader.name, read_policy.arn)
policy_wrapper.delete_policy(read_policy.arn)
delete_user(user_reader.name)
print(f"Deleted keys, detached and deleted policy, and deleted
{user_reader.name}.")

```

```
access_key_wrapper.delete_key(user_read_writer.name, user_read_writer_key.id)
detach_policy(user_read_writer.name, read_write_policy.arn)
policy_wrapper.delete_policy(read_write_policy.arn)
delete_user(user_read_writer.name)
print(f"Deleted keys, detached and deleted policy, and deleted
{user_read_writer.name}.")

bucket.objects.delete()
bucket.delete()
print(f"Emptied and deleted {bucket.name}.")
print("Thanks for watching!")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的以下主。
 - [AttachUserPolicy](#)
 - [CreateAccessKey](#)
 - [CreatePolicy](#)
 - [CreateUser](#)
 - [DeleteAccessKey](#)
 - [DeletePolicy](#)
 - [DeleteUser](#)
 - [DetachUserPolicy](#)
 - [ListUsers](#)
 - [UpdateUser](#)

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包管理 IAM 访问密钥

以下代码示例显示了如何：

- 创建并列出访问密钥。
- 了解上次使用访问密钥的时间和方式。
- 更新和删除访问密钥。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建包装 IAM 访问密钥操作的函数。

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource('iam')

def list_keys(user_name):
    """
```



```

Lists the keys owned by the specified user.

:param user_name: The name of the user.
:return: The list of keys owned by the user.
"""
try:
    keys = list(iam.User(user_name).access_keys.all())
    logger.info("Got %s access keys for %s.", len(keys), user_name)
except ClientError:
    logger.exception("Couldn't get access keys for %s.", user_name)
    raise
else:
    return keys

def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name, key_pair.id)
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair

def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response['AccessKeyLastUsed'].get('LastUsedDate', None)
        last_service = response['AccessKeyLastUsed'].get('ServiceName', None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.", key_id,
            response['UserName'], last_used_date, last_service)
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response

def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
    deactivated.
    """
    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:

```

```

        key.activate()
    else:
        key.deactivate()
    logger.info("%s key %s.", 'Activated' if activate else 'Deactivated',
key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", 'Activate' if activate else 'Deactivate',
key_id)
        raise

def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info(
            "Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise

```

使用包装器函数为当前用户执行访问密钥操作。

```

def usage_demo():
    """Shows how to create and manage access keys."""
    def print_keys():
        """Gets and prints the current keys for a user."""
        current_keys = list_keys(current_user_name)
        print("The current user's keys are now:")
        print(*[f"{key.id}: {key.status}" for key in current_keys], sep='\n')

    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    print('-'*88)
    print("Welcome to the AWS Identity and Account Management access key demo.")
    print('-'*88)
    current_user_name = iam.CurrentUser().user_name
    print(f"This demo creates an access key for the current user "
          f"{current_user_name}, manipulates the key in a few ways, and then "
          f"deletes it.")
    all_keys = list_keys(current_user_name)
    if len(all_keys) == 2:
        print("The current user already has the maximum of 2 access keys. To run "
              "this demo, either delete one of the access keys or use a user "
              "that has only 1 access key.")
    else:
        new_key = create_key(current_user_name)
        print(f"Created a new key with id {new_key.id} and secret "
              f"{new_key.secret}.")
        print_keys()
        existing_key = next(key for key in all_keys if key != new_key)
        last_use = get_last_use(existing_key.id)['AccessKeyLastUsed']
        print(f"Key {all_keys[0].id} was last used to access "
              f"{last_use['ServiceName']} "
              f"on {last_use['LastUsedDate']}")
        update_key(current_user_name, new_key.id, False)
        print(f"Key {new_key.id} is now deactivated.")
        print_keys()

```

```
delete_key(current_user_name, new_key.id)
print_keys()
print("Thanks for watching!")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的以下主。
 - [CreateAccessKey](#)
 - [DeleteAccessKey](#)
 - [GetAccessKeyLastUsed](#)
 - [ListAccessKeys](#)
 - [UpdateAccessKey](#)

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包管理 IAM policy

以下代码示例显示了如何：

- 创建和列出策略。
- 创建和获取策略版本。
- 将策略回滚到以前的版本。
- 删除策略。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建包装 IAM policy 操作的函数。

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource('iam')

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
        form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this policy
        applies to. This ARN can contain wildcards, such as
        'arn:aws:s3:::my-bucket/*' to allow actions on all objects
        in the bucket named 'my-bucket'.
```

```

:return: The newly created policy.
"""
policy_doc = {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": actions,
            "Resource": resource_arn
        }
    ]
}
try:
    policy = iam.create_policy(
        PolicyName=name, Description=description,
        PolicyDocument=json.dumps(policy_doc))
    logger.info("Created policy %s.", policy.arn)
except ClientError:
    logger.exception("Couldn't create policy %s.", name)
    raise
else:
    return policy

def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
        'Local' specifies that only locally managed policies are
    returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies

def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
        version for the policy. Otherwise, the default
        is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Action': actions,
                'Resource': resource_arn
            }
        ]
    }
    try:
        policy = iam.Policy(policy_arn)

```

```

        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default)
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id, policy_version.arn)
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version

def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get('Statement', None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
            policy_arn)
        raise
    else:
        return policy_statement

def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter('create_date'))
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
        logger.info("Set %s as the default version.", rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
            default_version.version_id)
    
```

```

except IndexError:
    if default_version is None:
        logger.warning("No default version found for %s.", policy_arn)
    elif rollback_version is None:
        logger.warning(
            "Default version %s found for %s, but no previous version exists,
so "
            "nothing to roll back to.", default_version.version_id, policy_arn)
except ClientError:
    logger.exception("Couldn't roll back version for %s.", policy_arn)
    raise
else:
    return rollback_version

def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise

```

使用包装器函数来创建策略、更新版本并获取相关信息。

```

def usage_demo():
    """Shows how to use the policy functions."""
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    print('-'*88)
    print("Welcome to the AWS Identity and Account Management policy demo.")
    print('-'*88)
    print("Policies let you define sets of permissions that can be attached to "
          "other IAM resources, like users and roles.")
    bucket_arn = f'arn:aws:s3::made-up-bucket-name'
    policy = create_policy(
        'demo-iam-policy', 'Policy for IAM demonstration.',
        ['s3:ListObjects'], bucket_arn)
    print(f"Created policy {policy.policy_name}.")
    policies = list_policies('Local')
    print(f"Your account has {len(policies)} managed policies:")
    print(*[pol.policy_name for pol in policies], sep=', ')
    time.sleep(1)
    policy_version = create_policy_version(
        policy.arn, ['s3:PutObject'], bucket_arn, True)
    print(f"Added policy version {policy_version.version_id} to policy "
          f"{policy.policy_name}.")
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is:")
    pprint.pprint(default_statement)
    rollback_version = rollback_policy_version(policy.arn)
    print(f"Rolled back to version {rollback_version.version_id} for "
          f"{policy.policy_name}.")
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is now:")
    pprint.pprint(default_statement)
    delete_policy(policy.arn)
    print(f"Deleted policy {policy.policy_name}.")
    print("Thanks for watching!")

```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的以下主。
 - [CreatePolicy](#)
 - [CreatePolicyVersion](#)
 - [DeletePolicy](#)
 - [DeletePolicyVersion](#)
 - [GetPolicyVersion](#)
 - [ListPolicies](#)
 - [ListPolicyVersions](#)
 - [SetDefaultPolicyVersion](#)

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包管理 IAM 角色

以下代码示例显示了如何：

- 创建一个 IAM 角色。
- 为角色附加和分离策略。
- 删除角色。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建包装 IAM 角色操作的函数。

```
import json
import logging
import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource('iam')

def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        'Version': '2012-10-17',
        'Statement': [{
            'Effect': 'Allow',
            'Principal': {'Service': service},
            'Action': 'sts:AssumeRole'
        } for service in allowed_services
    ]
}
```



```

try:
    role = iam.create_role(
        RoleName=role_name,
        AssumeRolePolicyDocument=json.dumps(trust_policy))
    logger.info("Created role %s.", role.name)
except ClientError:
    logger.exception("Couldn't create role %s.", role_name)
    raise
else:
    return role

def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
            role_name)
        raise

def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name)
        raise

def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise

```

使用包装器函数创建角色，然后附加和分离策略。

```

def usage_demo():
    """Shows how to use the role functions."""
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    print('-'*88)
    print("Welcome to the AWS Identity and Account Management role demo.")
    print('-'*88)
    print("Roles let you define sets of permissions and can be assumed by "

```

```
"other entities, like users and services.")
print("The first 10 roles currently in your account are:")
roles = list_roles(10)
print(f"The inline policies for role {roles[0].name} are:")
list_policies(roles[0].name)
role = create_role(
    'demo-iam-role',
    ['lambda.amazonaws.com', 'batchoperations.s3.amazonaws.com'])
print(f"Created role {role.name}, with trust policy:")
pprint.pprint(role.assume_role_policy_document)
policy_arn = 'arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess'
attach_policy(role.name, policy_arn)
print(f"Attached policy {policy_arn} to {role.name}.")
print(f"Policies attached to role {role.name} are:")
list_attached_policies(role.name)
detach_policy(role.name, policy_arn)
print(f"Detached policy {policy_arn} from {role.name}.")
delete_role(role.name)
print(f"Deleted {role.name}.")
print("Thanks for watching!")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的以下主。
 - [AttachRolePolicy](#)
 - [CreateRole](#)
 - [DeleteRole](#)
 - [DetachRolePolicy](#)

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包管理您的 IAM 账户

以下代码示例显示了如何：

- 获取并更新账户的别名。
- 生成用户及其凭证的报告。
- 获取账户使用情况的摘要。
- 获取有关您的账户中的所有用户、组、角色和策略的详细信息，包括它们之间的相互关系。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建包装 IAM 账户操作的函数。

```
import logging
import pprint
import sys
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource('iam')
```

```
def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
    alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response['AccountAliases']
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ','.join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response['AccountAliases']

def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """
    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise

def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise

def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info("Generating credentials report for your account. "
                    "Current state is %s.", response['State'])
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your
account.")
        raise
    else:
        return response
```

```

        return response

def get_credential_report():
    """
    Gets the most recently generated credentials report about the current account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response['Content'])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response['Content']

def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map

def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report, such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details

```

调用 wrapper 函数来更改账户别名并获取有关账户的报告。

```

def usage_demo():
    """Shows how to use the account functions."""
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
    print('-'*88)
    print("Welcome to the AWS Identity and Account Management account demo.")
    print('-'*88)
    print("Setting an account alias lets you use the alias in your sign-in URL "
          "instead of your account number.")
    old_aliases = list_aliases()
    if len(old_aliases) > 0:

```

```

        print(f"Your account currently uses '{old_aliases[0]}' as its alias.")
    else:
        print("Your account currently has no alias.")
    for index in range(1, 3):
        new_alias = f'alias-{index}-{time.time_ns()}'
        print(f"Setting your account alias to {new_alias}")
        create_alias(new_alias)
    current_aliases = list_aliases()
    print(f"Your account alias is now {current_aliases}.")
    delete_alias(current_aliases[0])
    print(f"Your account now has no alias.")
    if len(old_aliases) > 0:
        print(f"Restoring your original alias back to {old_aliases[0]}...")
        create_alias(old_aliases[0])

print('-'*88)
print("You can get various reports about your account.")
print("Let's generate a credentials report...")
report_state = None
while report_state != 'COMPLETE':
    cred_report_response = generate_credential_report()
    old_report_state = report_state
    report_state = cred_report_response['State']
    if report_state != old_report_state:
        print(report_state, sep='')
    else:
        print('.', sep='')
    sys.stdout.flush()
    time.sleep(1)
print()
cred_report = get_credential_report()
col_count = 3
print(f"Got credentials report. Showing only the first {col_count} columns.")
cred_lines = [line.split(',')[0:col_count] for line
               in cred_report.decode('utf-8').split('\n')]
col_width = max([len(item) for line in cred_lines for item in line]) + 2
for line in cred_report.decode('utf-8').split('\n'):
    print(''.join(element.ljust(col_width)
                    for element in line.split(',')[0:col_count]))

print('-'*88)
print("Let's get an account summary.")
summary = get_summary()
print("Here's your summary:")
pprint.pprint(summary)

print('-'*88)
print("Let's get authorization details!")
details = get_authorization_details([])
see_details = input("These are pretty long, do you want to see them (y/n)? ")
if see_details.lower() == 'y':
    pprint.pprint(details)

print('-'*88)
pw_policy_created = None
see_pw_policy = input("Want to see the password policy for the account (y/n)? ")
if see_pw_policy.lower() == 'y':
    while True:
        if print_password_policy():
            break
        else:
            answer = input("Do you want to create a default password policy (y/n)? ")
            if answer.lower() == 'y':
                pw_policy_created = iam.create_account_password_policy()

```

```
        else:
            break
    if pw_policy_created is not None:
        answer = input("Do you want to delete the password policy (y/n)? ")
        if answer.lower() == 'y':
            pw_policy_created.delete()
            print("Password policy deleted.")

    print("The SAML providers for your account are:")
    list_saml_providers(10)

    print('-'*88)
    print("Thanks for watching.")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的以下主。
 - [CreateAccountAlias](#)
 - [DeleteAccountAlias](#)
 - [GenerateCredentialReport](#)
 - [GetAccountAuthorizationDetails](#)
 - [GetAccountSummary](#)
 - [GetCredentialReport](#)
 - [ListAccountAliases](#)

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon SDK 回滚 IAM policy 版本

以下代码示例显示了如何：

- 获取按日期顺序排序的策略版本列表。
- 查找默认策略版本。
- 将以前的策略版本设为默认版本。
- 删除旧的默认版本。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
```

```
try:
    policy_versions = sorted(
        iam.Policy(policy_arn).versions.all(),
        key=operator.attrgetter('create_date'))
    logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
except ClientError:
    logger.exception("Couldn't get versions for %s.", policy_arn)
    raise

default_version = None
rollback_version = None
try:
    while default_version is None:
        ver = policy_versions.pop()
        if ver.is_default_version:
            default_version = ver
    rollback_version = policy_versions.pop()
    rollback_version.set_as_default()
    logger.info("Set %s as the default version.", rollback_version.version_id)
    default_version.delete()
    logger.info("Deleted original default version %s.",
default_version.version_id)
except IndexError:
    if default_version is None:
        logger.warning("No default version found for %s.", policy_arn)
    elif rollback_version is None:
        logger.warning(
            "Default version %s found for %s, but no previous version exists,
so "
            "nothing to roll back to.", default_version.version_id, policy_arn)
except ClientError:
    logger.exception("Couldn't roll back version for %s.", policy_arn)
    raise
else:
    return rollback_version
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的以下主。
 - [DeletePolicyVersion](#)
 - [ListPolicyVersions](#)
 - [SetDefaultPolicyVersion](#)

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。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。

适用于使用 Amazon SDK 的 IAM 的跨服务示例

以下示例应用程序使用 Amazon SDK 将 IAM 与其他 Amazon Web Services 相组合。每个示例都包含一个指向 GitHub 的链接，其中包含了有关如何设置和运行应用程序的说明。

示例

- [创建一个长期 Amazon EMR 集群并使用 Amazon SDK 运行多个步骤 \(p. 730\)](#)
- [创建一个短期 Amazon EMR 集群并使用 Amazon SDK 运行一个步骤 \(p. 731\)](#)

创建一个长期 Amazon EMR 集群并使用 Amazon SDK 运行多个步骤

以下代码示例展示如何创建一个长期 Amazon EMR 集群并运行多个步骤。

Python

适用于 Python (Boto3) 的 SDK

创建一个长期 Amazon EMR 集群，该集群使用 Apache Spark 从 [Amazon Customer Reviews Dataset](#) (Amazon 客户评论数据集) 中查询历史 Amazon 评论数据。运行一项作业，该作业将获取特定类别中评价最高的产品的数据，这些产品的产品名称中包含关键字。作业结果将写入 Amazon Simple Storage Service (Amazon S3) 存储桶中。

- 创建 Amazon S3 存储桶并上传作业脚本。
- 创建 Amazon Identity and Access Management (IAM) 角色。
- 创建 Amazon Elastic Compute Cloud (Amazon EC2) 安全组。
- 创建一个长期集群并运行多个作业步骤。

此示例最好在 GitHub 上查看。有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon EC2
- Amazon EMR
- IAM
- Amazon S3

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

创建一个短期 Amazon EMR 集群并使用 Amazon SDK 运行一个步骤

以下代码示例展示如何创建一个短期 Amazon EMR 集群，该集群将运行一个步骤，并在该步骤完成后自动终止。

Python

适用于 Python (Boto3) 的 SDK

创建一个短期 Amazon EMR 集群，该集群将使用 Apache Spark 估计 pi 的值，以并行处理大量计算。该作业会将输出写入 Amazon EMR 日志中，以及 Amazon Simple Storage Service (Amazon S3) 存储桶中。该集群将在完成该作业后自行终止。

- 创建 Amazon S3 存储桶并上传作业脚本。
- 创建 Amazon Identity and Access Management (IAM) 角色。
- 创建 Amazon Elastic Compute Cloud (Amazon EC2) 安全组。
- 创建一个短期集群并运行单个作业步骤。

此示例最好在 GitHub 上查看。有关完整的源代码以及如何设置和运行的说明，请参阅 [GitHub](#) 上的完整示例。

本示例中使用的服务

- Amazon EC2
- Amazon EMR
- IAM
- Amazon S3

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包的 Amazon STS 代码示例

以下代码示例显示如何将 Amazon STS 与 Amazon 软件开发工具包 (SDK) 一起使用。

这些示例可分为以下几类：

操作

展示如何调用具体服务函数的代码节选。

场景

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 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。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 Amazon 开发工具包的 Amazon STS 操作 \(p. 732\)](#)
 - [使用 Amazon 开发工具包代入具有 Amazon STS 的角色 \(p. 732\)](#)
 - [使用 Amazon 开发工具包获取具有 Amazon STS 的会话令牌 \(p. 736\)](#)
- [使用 Amazon 开发工具包的 Amazon STS 场景 \(p. 737\)](#)
 - [使用 Amazon 开发工具包代入需要具有 Amazon STS 的 MFA 令牌的 IAM 角色 \(p. 737\)](#)
 - [使用 Amazon 开发工具包为联合用户构建具有 Amazon STS 的 URL \(p. 741\)](#)
 - [使用 Amazon 开发工具包获取需要具有 Amazon STS 的 MFA 令牌的会话令牌 \(p. 743\)](#)

使用 Amazon 开发工具包的 Amazon STS 操作

以下代码示例演示了如何使用 Amazon 开发工具包来执行各个 Amazon STS 操作。这些代码节选调用了 Amazon STS API，但不旨在孤立运行。每个示例都包含一个指向 GitHub 的链接，其中包含了有关如何在上下文中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 Security Token Service API 参考](#)。

示例

- [使用 Amazon 开发工具包代入具有 Amazon STS 的角色 \(p. 732\)](#)
- [使用 Amazon 开发工具包获取具有 Amazon STS 的会话令牌 \(p. 736\)](#)

使用 Amazon 开发工具包代入具有 Amazon STS 的角色

以下代码示例演示了如何代入具有 Amazon STS 的角色。

JavaScript

SDK for JavaScript V3

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建客户端。

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "REGION"; //e.g. "us-east-1"
// Create an Amazon STS service client object.
const stsClient = new STSClient({ region: REGION });
export { stsClient };
```

代入 IAM 角色。

```
// Import required AWS SDK clients and commands for Node.js
import { stsClient } from "../libs/stsClient.js";
import {
  AssumeRoleCommand,
  GetCallerIdentityCommand,
} from "@aws-sdk/client-sts";

// Set the parameters
export const params = {
  RoleArn: "ARN_OF_ROLE_TO_ASSUME", //ARN_OF_ROLE_TO_ASSUME
  RoleSessionName: "session1",
  DurationSeconds: 900,
};

export const run = async () => {
  try {
    //Assume Role
    const data = await stsClient.send(new AssumeRoleCommand(params));
    return data;
    const rolecreds = {
      accessKeyId: data.Credentials.AccessKeyId,
      secretAccessKey: data.Credentials.SecretAccessKey,
      sessionToken: data.Credentials.SessionToken,
    };
    //Get Amazon Resource Name (ARN) of current identity
    try {
      const stsParams = { credentials: rolecreds };
      const stsClient = new STSClient(stsParams);
      const results = await stsClient.send(
        new GetCallerIdentityCommand(rolecreds)
      );
      console.log("Success", results);
    } catch (err) {
      console.log(err, err.stack);
    }
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

• 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [AssumeRole](#)。
SDK for JavaScript V2

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```
// Load the AWS SDK for Node.js
```

```
const AWS = require('aws-sdk');
// Set the region
AWS.config.update({region: 'REGION'});

var roleToAssume = {RoleArn: 'arn:aws:iam::123456789012:role/RoleName',
                    RoleSessionName: 'session1',
                    DurationSeconds: 900,};

var roleCreds;

// Create the STS service object
var sts = new AWS.STS({apiVersion: '2011-06-15'});

//Assume Role
sts.assumeRole(roleToAssume, function(err, data) {
    if (err) console.log(err, err.stack);
    else{
        roleCreds = {accessKeyId: data.Credentials.AccessKeyId,
                     secretAccessKey: data.Credentials.SecretAccessKey,
                     sessionToken: data.Credentials.SessionToken};
        stsGetCallerIdentity(roleCreds);
    }
});

//Get Arn of current identity
function stsGetCallerIdentity(creds) {
    var stsParams = {credentials: creds };
    // Create STS service object
    var sts = new AWS.STS(stsParams);

    sts.getCallerIdentity({}, function(err, data) {
        if (err) {
            console.log(err, err.stack);
        }
        else {
            console.log(data.Arn);
        }
    });
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for JavaScript API 参考》中的 [AssumeRole](#)。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

代入需要 MFA 令牌的 IAM 角色并使用临时凭证列出该账户的 Amazon S3 存储桶。

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
    account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                           grants access to list the other account's buckets.
```

```

:param session_name: The name of the STS session.
:param mfa_serial_number: The serial number of the MFA device. For a virtual
MFA
                        device, this is an ARN.
:param mfa_totp: A time-based, one-time password issued by the MFA device.
:param sts_client: A Boto3 STS instance that has permission to assume the role.
"""
response = sts_client.assume_role(
    RoleArn=assume_role_arn,
    RoleSessionName=session_name,
    SerialNumber=mfa_serial_number,
    TokenCode=mfa_totp)
temp_credentials = response['Credentials']
print(f"Assumed role {assume_role_arn} and got temporary credentials.")

s3_resource = boto3.resource(
    's3',
    aws_access_key_id=temp_credentials['AccessKeyId'],
    aws_secret_access_key=temp_credentials['SecretAccessKey'],
    aws_session_token=temp_credentials['SessionToken'])

print(f"Listing buckets for the assumed role's account:")
for bucket in s3_resource.buckets.all():
    print(bucket.name)

```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [AssumeRole](#)。

Ruby

SDK for Ruby

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

```

# Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
# This is separated into a factory function so that it can be mocked for unit
testing.
#
# @param key_id [String] The ID of the access key used by the STS client.
# @param key_secret [String] The secret part of the access key used by the STS
client.
def create_sts_client(key_id, key_secret)
  Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
end

# Gets temporary credentials that can be used to assume a role.
#
# @param role_arn [String] The ARN of the role that is assumed when these
credentials
#
                        are used.
# @param sts_client [Aws::STS::Client] An AWS STS client.
# @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
def assume_role(role_arn, sts_client)
  credentials = Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: "create-use-assume-role-scenario"
  )
  puts("Assumed role '#{role_arn}', got temporary credentials.")

```

```
credentials
end
```

- 有关 API 详细信息，请参阅《Amazon SDK for Ruby API 参考》中的 [AssumeRole](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包获取具有 Amazon STS 的会话令牌

以下代码示例演示了如何获取具有 Amazon STS 的会话令牌并将其用于执行需要 MFA 令牌的服务操作。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

通过传递 MFA 令牌获取会话令牌，然后使用令牌列出账户的 Amazon S3 存储桶。

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
      sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
    MFA
                           device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp)
    else:
        response = sts_client.get_session_token()
    temp_credentials = response['Credentials']

    s3_resource = boto3.resource(
        's3',
        aws_access_key_id=temp_credentials['AccessKeyId'],
        aws_secret_access_key=temp_credentials['SecretAccessKey'],
        aws_session_token=temp_credentials['SessionToken'])

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [GetSessionToken](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包的 Amazon STS 场景

以下代码示例显示如何通过 Amazon 开发工具包实施 Amazon STS 中的常见场景。这些场景显示了如何通过 Amazon STS 中调用多个函数来完成特定任务。每个场景都包含一个指向 GitHub 的链接，其中包含了有关如何设置和运行代码的说明。

示例

- [使用 Amazon 开发工具包代入需要具有 Amazon STS 的 MFA 令牌的 IAM 角色 \(p. 737\)](#)
- [使用 Amazon 开发工具包为联合用户构建具有 Amazon STS 的 URL \(p. 741\)](#)
- [使用 Amazon 开发工具包获取需要具有 Amazon STS 的 MFA 令牌的会话令牌 \(p. 743\)](#)

使用 Amazon 开发工具包代入需要具有 Amazon STS 的 MFA 令牌的 IAM 角色

以下代码示例显示了如何：

- 创建一个 IAM 角色，该角色授予列出 Amazon S3 存储桶的权限。
- 创建一个 IAM 用户，该用户仅在提供 MFA 凭证时才有权代入角色。
- 为该用户注册一个 MFA 设备。
- 代入该角色并使用临时凭证列出 S3 存储桶。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建一个 IAM 用户，注册一个 MFA 设备，然后创建一个角色，该角色授予列出 S3 存储桶的权限。用户仅具有代入该角色的权限。

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual MFA device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role and requires
    MFA.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    For demonstration purposes, the user is created in the same account as the
    role,
    but in practice the user would likely be from another account.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
        that has permissions to create users, roles, and policies
    """
```



```

        in the account.
:return: The newly created user, user key, virtual MFA device, and role.
"""
user = iam_resource.create_user(UserName=unique_name('user'))
print(f"Created user {user.name}.")

virtual_mfa_device = iam_resource.create_virtual_mfa_device(
    VirtualMFADeviceName=unique_name('mfa'))
print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

print(f"Showing the QR code for the device. Scan this in the MFA app of your "
      f"choice.")
with open('qr.png', 'wb') as qr_file:
    qr_file.write(virtual_mfa_device.qr_code_png)
webbrowser.open(qr_file.name)

print(f"Enter two consecutive code from your MFA device.")
mfa_code_1 = input("Enter the first code: ")
mfa_code_2 = input("Enter the second code: ")
user.enable_mfa(
    SerialNumber=virtual_mfa_device.serial_number,
    AuthenticationCode1=mfa_code_1,
    AuthenticationCode2=mfa_code_2)
os.remove(qr_file.name)
print(f"MFA device is registered with the user.")

user_key = user.create_access_key_pair()
print(f"Created access key pair for user.")

print(f"Wait for user to be ready.", end='')
progress_bar(10)

role = iam_resource.create_role(
    RoleName=unique_name('role'),
    AssumeRolePolicyDocument=json.dumps({
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Principal': {'AWS': user.arn},
                'Action': 'sts:AssumeRole',
                'Condition': {'Bool': {'aws:MultiFactorAuthPresent': True}}
            }
        ]
    })
)
print(f"Created role {role.name} that requires MFA.")

policy = iam_resource.create_policy(
    PolicyName=unique_name('policy'),
    PolicyDocument=json.dumps({
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Action': 's3:ListAllMyBuckets',
                'Resource': 'arn:aws:s3:::*'
            }
        ]
    })
)
role.attach_policy(PolicyArn=policy.arn)
print(f"Created policy {policy.policy_name} and attached it to the role.")

user.create_policy(
    PolicyName=unique_name('user-policy'),

```

```

PolicyDocument=json.dumps({
    'Version': '2012-10-17',
    'Statement': [
        {
            'Effect': 'Allow',
            'Action': 'sts:AssumeRole',
            'Resource': role.arn
        }
    ]
})
)
print(f"Created an inline policy for {user.name} that lets the user assume "
      f"the role.")

print("Give AWS time to propagate these new resources and connections.",
end='')
progress_bar(10)

return user, user_key, virtual_mfa_device, role

```

显示不允许代入没有 MFA 令牌的角色。

```

def try_to_assume_role_without_mfa(assume_role_arn, session_name, sts_client):
    """
    Shows that attempting to assume the role without sending MFA credentials
    results
    in an AccessDenied error.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role to assume.
    :param session_name: The name of the STS session.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    print(f"Trying to assume the role without sending MFA credentials...")
    try:
        sts_client.assume_role(RoleArn=assume_role_arn,
                               RoleSessionName=session_name)
        raise RuntimeError("Expected AccessDenied error.")
    except ClientError as error:
        if error.response['Error']['Code'] == 'AccessDenied':
            print("Got AccessDenied.")
        else:
            raise

```

代入授予列出 S3 存储桶权限的角色，传递所需的 MFA 令牌，并显示可以列出的存储桶。

```

def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
    account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                           grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
MFA
                           device, this is an ARN.

```

```
:param mfa_totp: A time-based, one-time password issued by the MFA device.
:param sts_client: A Boto3 STS instance that has permission to assume the role.
"""
response = sts_client.assume_role(
    RoleArn=assume_role_arn,
    RoleSessionName=session_name,
    SerialNumber=mfa_serial_number,
    TokenCode=mfa_totp)
temp_credentials = response['Credentials']
print(f"Assumed role {assume_role_arn} and got temporary credentials.")

s3_resource = boto3.resource(
    's3',
    aws_access_key_id=temp_credentials['AccessKeyId'],
    aws_secret_access_key=temp_credentials['SecretAccessKey'],
    aws_session_token=temp_credentials['SessionToken'])

print(f"Listing buckets for the assumed role's account:")
for bucket in s3_resource.buckets.all():
    print(bucket.name)
```

销毁为演示创建的资源。

```
def teardown(user, virtual_mfa_device, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        policy_name = attached.policy_name
        role.detach_policy(PolicyArn=attached.arn)
        attached.delete()
        print(f"Detached and deleted {policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

使用之前定义的函数运行此方案。

```
def usage_demo():
    """Drives the demonstration."""
    print('-'*88)
    print(f"Welcome to the AWS Security Token Service assume role demo, "
          f"starring multi-factor authentication (MFA)!")
    print('-'*88)
    iam_resource = boto3.resource('iam')
    user, user_key, virtual_mfa_device, role = setup(iam_resource)
    print(f"Created {user.name} and {role.name}.")
    try:
```

```
sts_client = boto3.client(
    'sts', aws_access_key_id=user_key.id,
    aws_secret_access_key=user_key.secret)
try_to_assume_role_without_mfa(role.arn, 'demo-sts-session', sts_client)
mfa_totp = input('Enter the code from your registered MFA device: ')
list_buckets_from_assumed_role_with_mfa(
    role.arn, 'demo-sts-session', virtual_mfa_device.serial_number,
    mfa_totp, sts_client)
finally:
    teardown(user, virtual_mfa_device, role)
    print("Thanks for watching!")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [AssumeRole](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包为联合用户构建具有 Amazon STS 的 URL

以下代码示例显示了如何：

- 创建一个 IAM 角色，该角色授予对当前账户的 Amazon S3 资源的只读访问权限。
- 从 Amazon 联合身份验证端点获取安全令牌。
- 构建一个可以用于通过联合凭证访问控制台的 URL。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建一个角色，该角色授予对当前账户的 S3 资源的只读访问权限。

```
def setup(iam_resource):
    """
    Creates a role that can be assumed by the current user.
    Attaches a policy that allows only Amazon S3 read-only access.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) instance
                        that has the permission to create a role.
    :return: The newly created role.
    """
    role = iam_resource.create_role(
        RoleName=unique_name('role'),
        AssumeRolePolicyDocument=json.dumps({
            'Version': '2012-10-17',
            'Statement': [
                {
                    'Effect': 'Allow',
                    'Principal': {'AWS': iam_resource.CurrentUser().arn},
                    'Action': 'sts:AssumeRole'
                }
            ]
        })
    )
```

```
role.attach_policy(PolicyArn='arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess')
print(f"Created role {role.name}.")

print("Give AWS time to propagate these new resources and connections.",
end='')
progress_bar(10)

return role
```

从 Amazon 联合身份验证端点获取安全令牌并构建一个可用于通过联合凭证访问控制台的 URL。

```
def construct_federated_url(assume_role_arn, session_name, issuer, sts_client):
    """
    Constructs a URL that gives federated users direct access to the AWS Management
    Console.

    1. Acquires temporary credentials from AWS Security Token Service (AWS STS)
    that
        can be used to assume a role with limited permissions.
    2. Uses the temporary credentials to request a sign-in token from the
        AWS federation endpoint.
    3. Builds a URL that can be used in a browser to navigate to the AWS federation
        endpoint, includes the sign-in token for authentication, and redirects to
        the AWS Management Console with permissions defined by the role that was
        specified in step 1.

    :param assume_role_arn: The role that specifies the permissions that are
    granted.
                                The current user must have permission to assume the
    role.
    :param session_name: The name for the STS session.
    :param issuer: The organization that issues the URL.
    :param sts_client: A Boto3 STS instance that can assume the role.
    :return: The federated URL.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn, RoleSessionName=session_name)
    temp_credentials = response['Credentials']
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    session_data = {
        'sessionId': temp_credentials['AccessKeyId'],
        'sessionKey': temp_credentials['SecretAccessKey'],
        'sessionToken': temp_credentials['SessionToken']
    }
    aws_federated_signin_endpoint = 'https://signin.aws.amazon.com/federation'

    # Make a request to the AWS federation endpoint to get a sign-in token.
    # The requests.get function URL-encodes the parameters and builds the query
    string
    # before making the request.
    response = requests.get(
        aws_federated_signin_endpoint,
        params={
            'Action': 'getSigninToken',
            'SessionDuration': str(datetime.timedelta(hours=12).seconds),
            'Session': json.dumps(session_data)
        })
    signin_token = json.loads(response.text)
    print(f"Got a sign-in token from the AWS sign-in federation endpoint.")

    # Make a federated URL that can be used to sign into the AWS Management
    Console.
    query_string = urllib.parse.urlencode({
```

```
'Action': 'login',
'Issuer': issuer,
'Destination': 'https://console.aws.amazon.com/',
'SigninToken': signin_token['SigninToken']
})
federated_url = f'{aws_federated_signin_endpoint}?{query_string}'
return federated_url
```

销毁为演示创建的资源。

```
def teardown(role):
    """
    Removes all resources created during setup.

    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        role.detach_policy(PolicyArn=attached.arn)
        print(f"Detached {attached.policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
```

使用之前定义的函数运行此方案。

```
def usage_demo():
    """Drives the demonstration."""
    print('-'*88)
    print(f"Welcome to the AWS Security Token Service federated URL demo.")
    print('-'*88)
    iam_resource = boto3.resource('iam')
    role = setup(iam_resource)
    sts_client = boto3.client('sts')
    try:
        federated_url = construct_federated_url(
            role.arn, 'AssumeRoleDemoSession', 'example.org', sts_client)
        print("Constructed a federated URL that can be used to connect to the "
              "AWS Management Console with role-defined permissions:")
        print('-'*88)
        print(federated_url)
        print('-'*88)
        _ = input("Copy and paste the above URL into a browser to open the AWS "
                  "Management Console with limited permissions. When done, press "
                  "Enter to clean up and complete this demo.")
    finally:
        teardown(role)
    print("Thanks for watching!")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [AssumeRole](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。本主题还包括有关入门的信息以及有关先前的软件开发工具包版本的详细信息。

使用 Amazon 开发工具包获取需要具有 Amazon STS 的 MFA 令牌的会话令牌

以下代码示例显示了如何：

- 创建一个 IAM 角色，该角色授予列出 Amazon S3 存储桶的权限。
- 创建一个 IAM 用户，该用户仅在提供 MFA 凭证时才有权代入角色。
- 为该用户注册一个 MFA 设备。
- 提供 MFA 凭证以获取会话令牌，并使用临时凭证列出 S3 存储桶。

Python

适用于 Python (Boto3) 的 SDK

Tip

要了解如何设置和运行此示例，请参阅 [GitHub](#)。

创建一个 IAM 用户，注册一个 MFA 设备，然后创建一个角色，该角色授予仅在使用 MFA 凭证时允许用户列出 S3 存储桶的权限。

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual multi-factor authentication (MFA) device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates an inline policy for the user that lets the user list Amazon S3
    buckets,
    but only when MFA credentials are used.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM) resource
                        that has permissions to create users, MFA devices, and
                        policies in the account.
    :return: The newly created user, user key, and virtual MFA device.
    """
    user = iam_resource.create_user(Username=unique_name('user'))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name('mfa'))
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(f"Showing the QR code for the device. Scan this in the MFA app of your "
          f"choice.")
    with open('qr.png', 'wb') as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
        AuthenticationCode2=mfa_code_2)
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")

    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")
```



```
print(f"Wait for user to be ready.", end='')
progress_bar(10)

user.create_policy(
    PolicyName=unique_name('user-policy'),
    PolicyDocument=json.dumps({
        'Version': '2012-10-17',
        'Statement': [
            {
                'Effect': 'Allow',
                'Action': 's3:ListAllMyBuckets',
                'Resource': 'arn:aws:s3:::*',
                'Condition': {'Bool': {'aws:MultiFactorAuthPresent': True}}
            }
        ]
    })
)
print(f"Created an inline policy for {user.name} that lets the user list
buckets, "
      f"but only when MFA credentials are present.")

print("Give AWS time to propagate these new resources and connections.",
end='')
progress_bar(10)

return user, user_key, virtual_mfa_device
```

通过传递 MFA 令牌获取临时会话凭证，并使用凭证列出账户的 S3 存储桶。

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
MFA
                           device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp)
    else:
        response = sts_client.get_session_token()
    temp_credentials = response['Credentials']

    s3_resource = boto3.resource(
        's3',
        aws_access_key_id=temp_credentials['AccessKeyId'],
        aws_secret_access_key=temp_credentials['SecretAccessKey'],
        aws_session_token=temp_credentials['SessionToken'])

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

销毁为演示创建的资源。

```
def teardown(user, virtual_mfa_device):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo MFA device.
    """
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

使用之前定义的函数运行此方案。

```
def usage_demo():
    """Drives the demonstration."""
    print('-'*88)
    print(f"Welcome to the AWS Security Token Service assume role demo, "
          f"starring multi-factor authentication (MFA)!")
    print('-'*88)
    iam_resource = boto3.resource('iam')
    user, user_key, virtual_mfa_device = setup(iam_resource)
    try:
        sts_client = boto3.client(
            'sts', aws_access_key_id=user_key.id,
            aws_secret_access_key=user_key.secret)
        try:
            print("Listing buckets without specifying MFA credentials.")
            list_buckets_with_session_token_with_mfa(None, None, sts_client)
        except ClientError as error:
            if error.response['Error']['Code'] == 'AccessDenied':
                print("Got expected AccessDenied error.")
            mfa_totp = input('Enter the code from your registered MFA device: ')
            list_buckets_with_session_token_with_mfa(
                virtual_mfa_device.serial_number, mfa_totp, sts_client)
    finally:
        teardown(user, virtual_mfa_device)
    print("Thanks for watching!")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API 参考》中的 [GetSessionToken](#)。

有关 Amazon 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 IAM 与 Amazon 开发工具包配合使用 \(p. 15\)](#)。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。

IAM 和 Amazon STS 中的安全性

Amazon 十分重视云安全性。为了满足对安全性最敏感的组织的需求，我们打造了具有超高安全性的数据中心和网络架构。作为 Amazon 客户，您也将从这些数据中心和网络架构受益。

安全性是 Amazon 和您的共同责任。[责任共担模型](#)将其描述为云的安全性和云中的安全性：

- 云的安全性 – Amazon 负责保护在 Amazon 云中运行 Amazon 服务的基础设施。Amazon 还向您提供可安全使用的服务。第三方审核员定期测试和验证我们的安全性的有效性，作为 [Amazon Compliance Programs](#) 的一部分。要了解适用于 Amazon Identity and Access Management (IAM) 的合规性计划，请参阅 [合规性计划范围内的 Amazon 服务](#)。
- 云中的安全性 - 您的责任由您使用的 Amazon 服务决定。您还需要对其它因素负责，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。

此文档将帮助您了解如何在使用 Amazon Identity and Access Management (IAM) 和 Amazon Security Token Service (Amazon STS) 时应用责任共担模式。以下主题说明如何配置 IAM 和 Amazon STS 以实现您的安全性和合规性目标。您还会了解如何使用其他 Amazon 服务以帮助您监控和保护 IAM 资源。

目录

- [中的数据保护 Amazon Identity and Access Management](#) (p. 747)
- [Amazon Identity and Access Management 中的日志记录和监控](#) (p. 748)
- [Amazon Identity and Access Management 的合规性验证](#) (p. 749)
- [Amazon Identity and Access Management 中的故障恢复能力](#) (p. 749)
- [Amazon Identity and Access Management 中的基础设施安全性](#) (p. 751)
- [Amazon Identity and Access Management 中的配置和漏洞分析](#) (p. 751)
- [Amazon Identity and Access Management 中的安全最佳实践和使用案例](#) (p. 751)
- [适用于 Amazon Identity and Access Management Access Analyzer 的 Amazon 托管策略](#) (p. 757)

中的数据保护 Amazon Identity and Access Management

Amazon [责任共担模式](#)适用于 Amazon Identity and Access Management 中的数据保护。如该模式所述，Amazon 负责保护运行所有 Amazon Web Services 云的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。此内容包括您所使用的 Amazon Web Services 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 Amazon Web Services 账户凭证并使用 Amazon Identity and Access Management (IAM) 设置单独的用户账户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用 Multi-Factor Authentication (MFA)。
- 使用 SSL/TLS 与 Amazon 资源进行通信。建议使用 TLS 1.2 或更高版本。
- 使用 Amazon CloudTrail 设置 API 和用户活动日志记录。
- 使用 Amazon 加密解决方案以及 Amazon 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Simple Storage Service (Amazon S3) 中的个人数据。
- 如果在通过命令行界面或 API 访问 Amazon 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 终端节点。有关可用的 FIPS 终端节点的更多信息，请参阅 [《美国联邦信息处理标准 \(FIPS\) 第 140-2 版》](#)。

我们强烈建议您切勿将机密信息或敏感信息（例如您客户的电子邮件地址）放入标签或自由格式字段（例如名称字段）。这包括使用控制台、API、Amazon CLI 或 Amazon SDK 处理 IAM 或其他 Amazon 服务时。您在用于名称的标签或自由格式字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，我们强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

IAM 和 Amazon STS 中的数据加密

数据加密通常分为两类：静态加密和传输中加密。

静态加密

将对由 IAM 收集和存储的数据进行静态加密。

- IAM – 在 IAM 中收集和存储的数据包括 IP 地址、客户账户元数据和客户识别数据（包括密码）。客户账户元数据和客户识别数据使用 AES 256 进行静态加密或使用 SHA 256 进行哈希处理。
- Amazon STS - Amazon STS 不收集客户内容（用于记录针对服务的成功请求和有错误的请求的服务日志除外）。

传输中加密

客户识别数据（包括密码）将通过 TLS 1.1 和 1.2 进行传输中加密。所有 Amazon STS 终端节点都支持 HTTPS 来对传输中的数据进行加密。有关 Amazon STS 终端节点的列表，请参阅 [区域和终端节点 \(p. 333\)](#)。

IAM 和 Amazon STS 中的密钥管理

您无法使用 IAM 或 Amazon STS 管理加密密钥。有关加密密钥的更多信息，请参阅 Amazon Key Management Service 开发人员指南中的 [什么是 Amazon KMS？](#)。

IAM 和 Amazon STS 中的互联网络流量保密性

必须使用传输层安全协议 (TLS) 发出对 IAM 的请求。可以使用 VPC 终端节点保护与 Amazon STS 服务的连接。要了解更多信息，请参阅 [使用 Amazon STS 接口 VPC 终端节点 \(p. 334\)](#)。

Amazon Identity and Access Management 中的日志记录和监控

监控是使您的 Amazon Identity and Access Management (IAM)、Amazon Security Token Service (Amazon STS) 和您的其他 Amazon 解决方案保持可靠性、可用性和性能的重要环节。Amazon 提供了多种工具，用于监控您 Amazon 资源和对潜在的事件作出响应：

- Amazon CloudTrail 将对 IAM 和 Amazon STS 的所有 API 调用作为事件捕获，包括来自控制台的调用和 API 调用。要了解有关将 CloudTrail 与 IAM 和 Amazon STS 搭配使用的更多信息，请参阅 [使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用 \(p. 340\)](#)。有关 CloudTrail 的更多信息，请参阅《[Amazon CloudTrail 用户指南](#)》。
- Amazon Identity and Access Management Access Analyzer 帮助您标识企业和账户中与外部实体共享的资源，例如 Amazon S3 存储桶或 IAM 角色。这可以帮助您识别对资源和数据的意外访问，此类访问会带来安全风险。要了解更多信息，请参阅 [什么是 IAM 访问分析器？](#)
- Amazon CloudWatch 实时监控您的 Amazon 资源以及在 Amazon 上运行的应用程序。您可以收集和跟踪指标，创建自定义的控制面板，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。例如，您可以具有 Amazon EC2 实例的 CloudWatch 跟踪 CPU 使用率或其他指标并且在需要时自动启动新实例。有关更多信息，请参阅 [Amazon CloudWatch 用户指南](#)。

- Amazon CloudWatch Logs 帮助您监控、存储和访问来自 Amazon EC2 实例、CloudTrail 和其他来源的日志文件。CloudWatch Logs 可以监控日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 [Amazon CloudWatch Logs 用户指南](#)。

有关 IAM 的其他资源和安全最佳实践，请参阅 [Amazon Identity and Access Management 中的安全最佳实践和使用案例 \(p. 751\)](#)。

Amazon Identity and Access Management 的合规性验证

作为多项 Amazon 合规性计划的一部分，第三方审计员将评估 Amazon Identity and Access Management (IAM) 的安全性和合规性。这些合规性计划包括 SOC、PCI、FedRAMP、ISO 等。

要了解其他 Amazon Web Services 是否在特定合规性计划范围内，请参阅[合规性计划范围内的 Amazon Web Services](#)。有关常规信息，请参阅[Amazon Web Services 合规性计划](#)。

您可以使用 Amazon Artifact 下载第三方审计报告。有关更多信息，请参阅、[在 Amazon Artifact 中下载报告](#)。

您使用 Amazon Web Services 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。Amazon 提供以下资源来帮助满足合规性：

- [安全性与合规性快速入门指南](#) – 这些部署指南讨论了架构注意事项，并提供了在 Amazon 上部署注重安全性和合规性的基准环境的步骤。
- [亚马逊云科技上的 HIPAA 安全性和合规性架构设计](#) – 此白皮书介绍了企业如何使用 Amazon Web Services 创建符合 HIPAA 要求的应用程序。

Note

并非所有 Amazon Web Services 都符合 HIPAA 要求。有关更多信息，请参阅[符合 HIPAA 要求的服务参考](#)。

- [Amazon 合规性资源](#) – 此业务手册和指南集合可能适用于您的行业和位置。
- Amazon Config 开发人员指南中的[使用规则评估资源](#) – 此 Amazon Config 服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- [Amazon Security Hub](#)：此 Amazon Web Service 提供了 Amazon 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践规范。

Amazon Identity and Access Management 中的故障恢复能力

Amazon 全球基础设施围绕 Amazon 区域和可用区构建。Amazon 区域有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。有关 Amazon 区域和可用区的更多信息，请参阅 [Amazon 全球基础设施](#)。

Amazon Identity and Access Management (IAM) 和 Amazon Security Token Service (Amazon STS) 是可自我维持、基于区域的服务，可在全球范围内使用。

IAM 是一种非常关键的 Amazon Web Service。在 Amazon 中执行的每项操作必须由 IAM 进行身份验证和授权。IAM 根据存储在 IAM 中的身份和策略检查每个请求，以确定是允许还是拒绝请求。IAM 的设计采用了单独的控制面板和数据面板，这样，即使在意外故障期间，服务也能进行身份验证。授权中使用的 IAM 资源（如角色和策略）存储在控制面板中。IAM 客户可以通过使用 `DeletePolicy` 和 `AttachRolePolicy` 等 IAM 操作来更改这些资源的配置。这些配置更改请求将发送到控制面板。所有商业 Amazon Web Services

区域 都有一个 IAM 控制面板，位于美国东部（弗吉尼亚州北部）区域。随后，IAM 系统会在每个已启用的 [Amazon Web Services 区域](#) 中将配置更改传播到 IAM 数据面板。IAM 数据面板本质上是 IAM 控制面板配置数据的只读副本。每个 Amazon Web Services 区域 都有一个完全独立的 IAM 数据面板实例，该实例对来自同一区域的请求执行身份验证和授权。在每个区域中，IAM 数据面板至少分布在三个可用区中，并且具有足够的容量来容忍可用区丢失而不会对客户造成任何损害。IAM 控制面板和数据面板都是为零停机时间而构建，所有软件更新和扩展操作均以对客户不可见的方式执行。

默认情况下，总是将 Amazon STS 请求将转到单个全局端点。但是，您可以选择使用区域 Amazon STS 端点来减少延迟或为应用程序提供额外冗余。要了解更多信息，请参阅[“在 Amazon 区域中管理 Amazon STS \(p. 331\)”](#)。

某些事件可能会中断 Amazon Web Services 区域 之间的网络通信。但是，即使您无法与全局 IAM 端点进行通信，Amazon STS 仍然可以对 IAM 主体进行身份验证，并且 IAM 可以授权您的请求。中断通信的事件的具体详细信息将决定您访问 Amazon 服务的能力。在大多数情况下，您可继续在 Amazon 环境中使用 IAM 凭证。以下情况可能适用于中断通信的事件。

IAM 用户的访问密钥

您可以在具有长期 [IAM 用户的访问密钥](#) 的区域中无限期地进行身份验证。当您使用 Amazon Command Line Interface 和 API 时，您需要提供您的 Amazon 访问密钥，以便 Amazon 可以在编程请求中验证您的身份。

Important

作为最佳实践，我们建议您的用户使用[临时凭证](#)而不是长期访问密钥来登录。

临时凭证

您可以使用 Amazon STS 区域性[服务端点申请新的临时凭证](#)，凭证至少在 24 小时内有效。以下 API 操作会生成临时凭证。

- AssumeRole
- AssumeRoleWithWebIdentity
- AssumeRoleWithSAML
- GetFederationToken
- GetSessionToken

主体和权限

- 您可能无法在 IAM 中添加、修改或删除主体或权限。
- 您的凭证可能无法反映您最近在 IAM 中应用的权限的更改。有关更多信息，请参阅[我所做的更改可能不会立即可见 \(p. 865\)](#)。

Amazon Web Services Management Console

- 您可能可以作为 IAM 用户使用区域性登录端点登录 Amazon Web Services Management Console。区域性登录端点具有以下 URL 格式。

`https://{Account ID}.signin.aws.amazon.com/console?region={Region}`

示例：https://111122223333.signin.aws.amazon.com/console?region=us-west-2

- 您可能无法完成 [Universal 2nd Factor \(U2F \)](#) 多重身份验证 (MFA) 。

IAM 恢复能力的最佳实践

Amazon 已将恢复能力内置于 Amazon Web Services 区域 和可用区。若您在与您的环境交互的系统中观察到有以下 IAM 最佳实践，您就可以利用这种恢复能力。

1. 使用 Amazon STS 区域性[服务端点](#)而不是默认的全局端点。
2. 检查您的环境中是否存在例行创建或修改 IAM 资源的重要资源的配置，并准备使用现有 IAM 资源的回退解决方案。

Amazon Identity and Access Management 中的基础设施安全性

作为一项托管式服务，Amazon Identity and Access Management (IAM) 和 Amazon Security Token Service (Amazon STS) 由 [Amazon Web Services：安全流程概览白皮书](#) 中所述的 Amazon 全球网络安全程序提供保护。

您可以使用 Amazon 发布的 API 调用通过网络访问 IAM。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 Amazon STS 生成临时安全凭证来对请求进行签名。

可使用 IAM HTTPS API（它可让您直接向服务发布 HTTPS 请求）以编程方式访问 IAM。查询 API 将返回敏感信息，包括安全凭证。因此，您必须将 HTTPS 用于所有 API 请求。使用 HTTPS API 时，必须添加代码，才能使用您的凭证对请求进行数字化签名。

您可以从任何网络位置调用这些 API 操作，但 IAM 不支持基于资源的访问策略，其中可以包含基于源 IP 地址的限制。您还可以使用 IAM 策略来控制来自特定 Amazon Virtual Private Cloud (Amazon VPC) 终端节点或特定 VPC 的访问。事实上，这隔离了在 Amazon 网络中仅从特定 VPC 到给定 IAM 资源的网络访问。

Amazon Identity and Access Management 中的配置和漏洞分析

Amazon 负责处理基本安全任务，如来宾操作系统 (OS) 和数据库补丁、防火墙配置和灾难恢复等。这些流程已通过相应第三方审核和认证。有关更多详细信息，请参阅以下资源：

- [责任共担模式](#)
- [Amazon Web Services：安全过程概述](#)（白皮书）

以下资源还处理 Amazon Identity and Access Management (IAM) 中的配置和漏洞分析：

- [Amazon Identity and Access Management 的合规性验证](#) (p. 749)
- [Amazon Identity and Access Management 中的安全最佳实践和使用案例](#) (p. 751)

Amazon Identity and Access Management 中的安全最佳实践和使用案例

Amazon Identity and Access Management (IAM) 提供了在您开发和实施自己的安全策略时需要考虑的大量安全功能。以下最佳实践是一般准则，并不代表完整的安全解决方案。这些最佳实践可能不适合您的环境或不满足您的环境要求，请将其视为有用的考虑因素而不是惯例。

要从 IAM 获取最大收益，需要花一些时间了解建议的最佳实践。完成这项工作的方法之一是了解如何在实际方案中结合使用 IAM 和其他 Amazon 服务。

主题

- [IAM 中的安全最佳实践](#) (p. 752)

- [IAM 的商用案例 \(p. 755\)](#)

IAM 中的安全最佳实践

 [Follow us on Twitter](#)

Amazon Identity and Access Management 最佳实践已于 2022 年 7 月 14 日更新。

为了帮助您确保 Amazon 资源的安全，请遵循 Amazon Identity and Access Management (IAM) 的最佳实践。

主题

- [要求人类用户使用带有身份提供商的联合身份验证才能使用临时凭证访问 Amazon \(p. 752\)](#)
- [要求工作负载使用具有 IAM 角色的临时证书访问 Amazon \(p. 752\)](#)
- [需要多重身份验证 \(MFA \) \(p. 753\)](#)
- [对于需要长期凭证的使用场景，请定期轮换访问密钥 \(p. 753\)](#)
- [保护您的根用户凭证，不要将其用于日常任务 \(p. 753\)](#)
- [应用最低权限许可 \(p. 754\)](#)
- [Amazon 托管策略及转向最低权限许可入门 \(p. 754\)](#)
- [使用 IAM Access Analyzer 根据访问活动生成最低权限策略 \(p. 754\)](#)
- [定期查看并移除未使用的用户、角色、权限、策略和凭证 \(p. 754\)](#)
- [使用 IAM policy 中的条件进一步限制访问权限 \(p. 754\)](#)
- [使用 IAM Access Analyzer 验证对资源的公共和跨账户存取 \(p. 754\)](#)
- [使用 IAM Access Analyzer 验证您的 IAM policy，以确保许可的安全性和功能性 \(p. 754\)](#)
- [跨多个账户建立许可防护机制 \(p. 755\)](#)
- [使用许可边界在账户内委派权限管理 \(p. 755\)](#)

要求人类用户使用带有身份提供商的联合身份验证才能使用临时凭证访问 Amazon

人类用户，也称为人类身份，是应用程序的人员、管理员、开发人员、操作员和使用者。他们必须有身份才能访问您的 Amazon 环境和应用程序。作为组织成员的人类用户也称为员工身份。人类用户也可以是您与之协作以及与您 Amazon 资源交互的外部用户。他们可以通过 Web 浏览器、客户端应用程序、移动应用程序或交互式命令行工具执行此操作。

请要求您的人类用户在访问 Amazon 时使用临时证书。您可以使用身份提供商来以担任角色的形式提供为人类用户对 Amazon 账户的联合访问权限，这将提供临时证书。对于集中式访问权限管理，我们建议使用 [Amazon IAM Identity Center \(successor to Amazon Single Sign-On\) \(IAM Identity Center \)](#) 来管理对您账户的访问权限以及这些账户中的其他权限。您可以使用 IAM Identity Center 管理您的用户身份，或者从外部身份提供者管理 IAM Identity Center 中用户身份的访问权限。有关更多信息，请参阅《[Amazon IAM Identity Center \(successor to Amazon Single Sign-On\) 用户指南](#)》中的什么是 Amazon IAM Identity Center (successor to Amazon Single Sign-On)。

有关角色的更多信息，请参阅[角色术语和概念 \(p. 158\)](#)。

要求工作负载使用具有 IAM 角色的临时证书访问 Amazon

工作负载是一系列资源和代码，它们可提供商业价值，如应用程序或后端过程。您的工作负载可能包含应用程序、操作工具和组件，它们需要身份才能向 Amazon Web Services 发送请求，例如请求读取数据。这些身份包括运行在您的 Amazon 环境（例如 Amazon EC2 实例或 Amazon Lambda 函数）中运行的计算机。

您还可以为需要访问权限的外部团体管理计算机身份。要为计算机身份授予访问权限，您可以使用 IAM 角色。IAM 角色具有特定的权限并可通过使用带有角色会话的临时安全凭证提供访问 Amazon 的方法。此外，您还可以拥有位于 Amazon 以外且需要访问您的 Amazon 环境的计算机。对于在 Amazon 外部运行的计算机，您可以使用 [Amazon Identity and Access Management Roles Anywhere](#)。有关角色的更多信息，请参阅 [IAM 角色 \(p. 158\)](#)。有关如何使用角色跨 Amazon Web Services 账户委派访问权限的详情，请参阅 [IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限 \(p. 31\)](#)。

需要多重身份验证 (MFA)

我们建议对访问您 Amazon 资源的人类用户和工作负载使用 IAM 角色，以便他们使用临时证书。但是，如果您的账户中需要 IAM 或根用户，则需要使用 MFA 来提高安全性。启用 MFA 后，用户便拥有了一部可生成身份验证质询响应的设备。各个用户的凭证和设备生成的响应是完成登录过程所必需的。有关更多信息，请参阅 [在 Amazon 中使用多重身份验证 \(MFA\) \(p. 104\)](#)。

如果您使用 IAM Identity Center 来对人类用户进行集中访问管理，则当您的身份源配置了 IAM Identity Center 身份存储、Amazon Managed Microsoft AD 或 AD Connector 时，您可以使用 IAM Identity Center MFA 功能。有关 IAM Identity Center 中的 MFA 的更多信息，请参阅《Amazon IAM Identity Center (successor to Amazon Single Sign-On) 用户指南》中的[多重身份验证](#)。

对于需要长期凭证的使用场景，请定期轮换访问密钥

在可能的情况下，我们建议使用临时凭证，而不是创建访问密钥等长期凭证。但是，对于需要具有编程访问权限和长期凭证的 IAM 用户情形，我们建议您轮换访问密钥。定期轮换长期凭证有助于您熟悉该流程。这在您遇到必须轮换凭证的情况（例如员工离开公司时）下非常有用。我们建议您使用 IAM 访问上次使用的信息来安全地轮换和移除访问密钥。有关更多信息，请参阅[轮换访问密钥 \(p. 100\)](#)。

在 Amazon 中，有一些特定的使用场景需要带有 IAM 用户的长期凭证。部分使用场景包括：

- 无法使用 IAM 角色的编程用例 – 您可以从需要访问 Amazon 的位置运行代码。在某些应用场景中，您无法使用 IAM 角色提供临时凭证，例如 WordPress 插件。在这些情况下，请使用该代码的 IAM 用户长期访问密钥对 Amazon 进行身份验证。
- Third-party Amazon clients (第三方 Amazon 客户端) – 如果您正在使用的工具不支持访问 IAM Identity Center，如不在 Amazon 上托管的第三方 Amazon 客户端或供应商，请使用 IAM 用户长期访问密钥。
- Amazon CodeCommit 访问 – 如果您正在使用 CodeCommit 来存储您的代码，则可以使用具有 SSH 密钥或服务特定凭证的 IAM 用户对 CodeCommit 进行存储库身份验证。我们建议您除了使用 IAM Identity Center 中的用户进行普通身份验证之外，再执行此操作。IAM Identity Center 中的用户是您的员工队伍中需要访问您的 Amazon 账户或您的云应用程序的人员。要在不配置 IAM 用户的情况下授予用户访问您 CodeCommit 存储库的权限，您可以配置 git-remote-codecommit 实用工具。有关 IAM 和 CodeCommit 的更多信息，请参阅[将 IAM 与 CodeCommit 结合使用：Git 凭证、SSH 密钥和 Amazon 访问密钥 \(p. 144\)](#)。有关配置 git-remote-codecommit 实用工具的更多信息，请参阅《Amazon CodeCommit 用户指南》中的[连接到具有轮换凭证的 Amazon CodeCommit 存储库](#)。
- Amazon Keyspaces (for Apache Cassandra) access (Amazon Keyspaces (Apache Cassandra 兼容) 访问) – 在您无法使用 IAM Identity Center 中的用户的情况下，如出于测试 Cassandra 兼容性的目的，您可以使用具有服务特定凭证的 IAM 用户向 Amazon Keyspaces 进行身份验证。IAM Identity Center 中的用户是您的员工队伍中需要访问您的 Amazon 账户或您的云应用程序的人员。您还可以使用临时凭证连接到 Amazon Keyspaces。有关更多信息，请参阅《Amazon Keyspaces (Apache Cassandra 兼容) 开发人员指南》中的[使用临时凭证连接到使用 IAM 角色和 SigV4 插件的 Amazon Keyspaces](#)。

保护您的根用户凭证，不要将其用于日常任务

在创建 Amazon Web Services 账户时，您将创建根用户名和密码，以登录 Amazon Web Services Management Console。请像保护其他敏感个人信息一样保护您的根用户凭证。您可以通过为根用户凭证配置 MFA 来完成此操作。我们不建议您为根用户生成访问密钥，因为它们允许完全访问您的所有 Amazon Web Services 资源，包括您的账单信息。请不要将根用户用于日常任务。仅将根用户用于完成仅限根用户开

展的任务。有关这些任务的完整列表，请参阅《Amazon 一般参考》中的[需要根用户凭证的任务](#)。有关更多信息，请参阅《Amazon Account Management 用户指南》中的[保护账户根用户的最佳实践](#)。

应用最低权限许可

在使用 IAM policy 设置权限时，请仅授予执行任务所需的许可。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。在探索工作负载或使用场景所需的权限时，您可以从较广泛的权限开始。随着使用场景逐渐成熟，您可以努力减少授予许可，直至达到最低权限的标准。有关使用 IAM 应用权限的更多信息，请参阅 [IAM 中的策略和权限](#)。(p. 355)。

Amazon 托管策略及转向最低权限许可入门

要开始向用户和工作负载授予权限，请使用 Amazon 托管策略来为许多常见使用场景授予权限。您可以在 Amazon Web Services 账户中找到这些策略。请记住，Amazon 托管策略可能不会为您的特定使用场景授予最低权限许可，因为它们可供所有 Amazon 客户使用。因此，我们建议通过定义特定于您的使用场景的[客户管理型策略](#)来减少许可。有关更多信息，请参阅[Amazon 托管策略](#) (p. 364)。有关专为特定任务函数制定的 Amazon 托管策略的更多信息，请参阅 [工作职能的 Amazon 托管策略](#) (p. 990)。

使用 IAM Access Analyzer 根据访问活动生成最低权限策略

如果仅授予执行任务所需的许可，您可以根据记录在 Amazon CloudTrail 中的访问活动生成策略。[IAM Access Analyzer](#) 会分析您的 IAM 角色使用的服务和操作，然后生成您可以使用的精细策略。测试每个生成的策略后，可以将该策略部署到生产环境中。这可确保您仅向工作负载授予所需的权限。有关策略生成的更多信息，请参阅 [IAM Access Analyzer 策略生成](#)。

定期查看并移除未使用的用户、角色、权限、策略和凭证

您的 Amazon Web Services 账户中可能存在不再需要的 IAM 用户、角色、许可、策略或凭证。IAM 提供了上次访问的信息来帮助您识别不再需要的用户、角色、许可、策略和凭证，以便您将其删除。这有助于减少必须监控的用户、角色、许可、策略和凭证的数量。您可以使用此信息优化 IAM policy 以更好地遵循最低权限许可。有关更多信息，请参阅[使用上次访问的信息优化 Amazon 中的权限](#) (p. 467)。

使用 IAM policy 中的条件进一步限制访问权限

您可以指定策略语句在何种条件下生效。这样，您就可以授予对操作和资源的访问权限，但前提是访问请求满足特定条件。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。您也可以使用条件来授予对服务操作的访问权限，但前提是它们是通过特定的 Amazon Web Service 使用的，例如 Amazon CloudFormation。有关更多信息，请参阅[IAM JSON 策略元素：Condition](#) (p. 948)。

使用 IAM Access Analyzer 验证对资源的公共和跨账户存取

在 Amazon 中授予公共或跨账户存取许可之前，我们建议您验证是否需要此类访问权限。您可以使用 IAM Access Analyzer 来帮助您预览和分析对受支持资源类型的公共和跨账户存取。您可以通过查看 IAM Access Analyzer 生成的[结果](#)来完成此操作。这些结果可帮助您验证资源访问控制是否授予了期望的访问权限。此外，在更新公共和跨账户权限时，您可以在对资源部署新的访问控制之前验证更改的效果。IAM Access Analyzer 还会持续监控受支持的资源类型，并为允许公共或跨账户存取的资源生成结果。有关更多信息，请参阅[使用 IAM Access Analyzer API 预览访问权限](#)。

使用 IAM Access Analyzer 验证您的 IAM policy，以确保许可的安全性和功能性

验证您创建的策略以确保它们符合 [IAM policy 语言](#) (p. 360) (JSON) 和 IAM 最佳实践。您可以使用 IAM Access Analyzer 策略验证来验证您的策略。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。在控制台中创作新策略或编辑现有策略时，IAM Access Analyzer 会

提供一些建议，帮助您在保存策略之前对其进行优化和验证。此外，我们还建议您查看和验证所有现有策略。有关更多信息，请参阅 [IAM Access Analyzer 策略验证](#)。有关 IAM Access Analyzer 提供的策略检查的更多信息，请参阅 [IAM Access Analyzer 策略检查参考](#)。

跨多个账户建立许可防护机制

在扩展工作负载时，请使用由 Amazon Organizations 托管的多个账户将工作负载分离。建议您使用 Organizations [服务控制策略](#) (SCP) 建立权限防护机制，以控制账户中所有 IAM 用户和角色的访问许可。SCP 是一种组织策略，可用于在 Amazon 组织、OU 或账户级别管理组织中的权限。您建立的许可防护机制适用于账户中涵盖的所有用户和角色。然而，单凭 SCP 不足以授予您组织中的账户许可。要实现此目标，管理员仍然必须将[基于身份或基于资源的策略](#) (p. 379)附加到 IAM 用户、IAM 角色或者您账户中的资源。有关更多信息，请参阅 [Amazon Organizations、账户和 IAM 防护机制](#)。

使用许可边界在账户内委派权限管理

在某些场景中，您可能需要将账户中的许可管理委派给其他用户。例如，您可以允许开发人员为其工作负载创建和管理角色。将许可委派给其他人时，请使用许可边界以设置您委派的许可最大数量。许可边界是一个高级功能，它使用托管策略设置基于身份的策略以为 IAM 角色授予的许可。许可边界自己不授予访问权限。有关更多信息，请参阅[IAM 实体的权限边界](#) (p. 370)。

IAM 的商用案例

IAM 的简单商用案例可帮助您了解使用该服务来控制您的用户所拥有的 Amazon 访问权限的基本方法。此使用案例只是粗略介绍，并未涵盖您使用 IAM API 来实现所需结果的技术性细节。

此使用案例将探讨一家名为 Example Corp 的虚构公司可能使用 IAM 的两种典型方式。第一种场景考虑 Amazon Elastic Compute Cloud (Amazon EC2)。第二种场景考虑 Amazon Simple Storage Service (Amazon S3)。

有关通过 Amazon 的其他服务使用 IAM 的更多信息，请参阅 [使用 IAM 的 Amazon 服务](#) (p. 908)。

主题

- [Example Corp 的初始设置](#) (p. 755)
- [将 IAM 与 Amazon EC2 结合使用的使用案例](#) (p. 756)
- [将 IAM 与 Amazon S3 结合使用的使用案例](#) (p. 756)

Example Corp 的初始设置

John 是 Example Corp 创始人。在公司成立之初，他创建了自己的 Amazon 账户，他本人使用 Amazon 产品。之后，他雇佣了员工，担任开发人员、管理员、测试人员、管理人员及系统管理员。

John 通过 Amazon Web Services 账户根用户凭证使用 Amazon Web Services Management Console，为自己创建了名为 John 的用户以及名为 Admins 的用户组。他使用 Amazon 托管策略 [AdministratorAccess](#) 向 Admins 组授予了对所有 Amazon 账户的资源执行所有操作的权限。然后，他将 John 用户添加到了 Admins 用户组中。有关为您自己创建管理员用户组和 IAM 用户，然后将用户添加至管理员用户组的分步指南，请参阅 [创建您的第一个 IAM 管理员用户和用户组](#) (p. 19)。

此时，John 可以停止使用根用户的凭证与 Amazon 交互，他开始只使用自己的用户凭证。

John 还创建了一个名为 AllUsers 的用户组，这样他就可以将任何账户范围内的权限轻松应用于 Amazon 账户内的所有用户。他将本人添加至该用户组。随后，他又创建了名为 Developers、Testers、Managers 及 SysAdmins 的用户组。他为每位员工创建了用户，并将这些用户归入各自的组。他还将所有用户添加至 AllUsers 用户组。有关创建用户和用户组的信息，请参阅 [创建 IAM 用户组](#) (p. 153)。有关创建用户的信息，请参阅[在您的 Amazon 账户中创建 IAM 用户](#) (p. 72)。有关向用户组中添加用户的信息，请参阅 [管理 IAM 用户组](#) (p. 153)。

将 IAM 与 Amazon EC2 结合使用的使用案例

类似 Example Corp 的公司通常使用 IAM 与类似 Amazon EC2 的服务互动。如要理解使用案例的这一部分，您需要对 Amazon EC2 有基本的了解。有关 Amazon EC2 的更多信息，请转至[适用于 Linux 实例的 Amazon EC2 用户指南](#)。

用户组的 Amazon EC2 权限

为提供“周边”控制，John 对 AllUsers 用户组附加了一个策略。如果来源 IP 地址位于 Example Corp 企业网络外部，则该策略拒绝用户的任何 Amazon 请求。

在 Example Corp.，不同的用户组需要不同的权限：

- System administrators - 需要创建和管理 AMI、实例、快照、卷、安全组等的权限。John 向 SysAdmins 组附加了一个 AmazonEC2FullAccess Amazon 托管策略，以为该用户组成员授予使用所有 Amazon EC2 操作的权限。
- Developers (开发人员) - 只需能够使用实例即可。因此，John 创建并向 Developers 用户组附加了策略，以允许开发人员调用 DescribeInstances、RunInstances、StopInstances、StartInstances 及 TerminateInstances。

Note

Amazon EC2 使用 SSH 密钥、Windows 密码及安全组来控制哪些人能够访问特定 Amazon EC2 实例的操作系统。在 IAM 系统中，无法允许或拒绝访问特定实例的操作系统。

- Managers - 无法执行任何 Amazon EC2 操作，但可列出当前可用的 Amazon EC2 资源。因此，John 创建并向 Managers 用户组附加了一个策略，以便仅允许该组成员调用 Amazon EC2 "Describe" API 操作。

如需有关上述策略具体形式的示例，请参阅 [IAM 基于身份的策略示例 \(p. 391\)](#) 和 [适用于 Linux 实例的 Amazon EC2 用户指南](#) 的 [使用 Amazon Identity and Access Management](#)。

用户的工作职能更改

此时，其中一位开发人员 Paulo 的工作职能发生转变，成为一名管理人员。John 将 Paulo 从 Developers 用户组移至 Managers 用户组。现在，Paulo 位于 Managers 用户组，因此他与 Amazon EC2 实例交互的能力受到限制。他无法启动或启用实例。即使他是启动或启用实例的用户，也无法停止或终止现有实例。他只能列出 Example Corp 用户已启动的实例。

将 IAM 与 Amazon S3 结合使用的使用案例

类似 Example Corp 的公司通常还通过 Amazon S3 使用 IAM。John 已为公司创建一个名为 aws-s3-bucket 的 Amazon S3 存储桶。

创建其他用户和用户组

作为员工，Zhang 和 Mary 都需要能够在公司的存储桶中创建自己的数据。他们还需要读取和写入所有开发人员都要处理的共享数据。为做到这一点，John 采用 Amazon S3 键前缀方案，在 aws-s3-bucket 中按照逻辑方式排列数据，如下图所示。

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

John 针对每位员工将 `/aws-s3-bucket` 分隔成一系列主目录，并为开发人员和管理人员组留出一个共享区域。

现在，John 创建一组策略，以便向用户和用户组分配权限：

- Zhang 的主目录访问 - John 向 Zhang 附加的策略允许后者读取、写入和列出任何带 Amazon S3 键前缀 `/aws-s3-bucket/home/zhang/` 的对象
- Mary 的主目录访问 - John 向 Mary 附加的策略允许后者读取、写入和列出任何带 Amazon S3 键前缀 `/aws-s3-bucket/home/mary/` 的对象
- Developers 用户组的共享目录访问 - John 向该用户组附加的策略允许开发人员读取、写入和列出 `/aws-s3-bucket/share/developers/` 中的任何对象
- Managers 用户组的共享目录访问 - John 向该用户组附加的策略允许管理人员读取、写入和列出 `/aws-s3-bucket/share/managers/` 中的对象

Note

对于创建存储桶或对象的用户，Amazon S3 不会自动授予其对存储桶或对象执行其他操作的权限。因此，在您的 IAM 策略中，您必须显式授予用户使用他们所创建的 Amazon S3 资源的权限。

有关这些策略具体形式的示例，请参阅 Amazon Simple Storage Service 用户指南中的[访问控制](#)。有关如何在运行时对策略进行评估的信息，请参阅[策略评估逻辑](#) (p. 972)。

用户的工作职能更改

此时，其中一位开发人员 Zhang 的工作职能发生转变，成为一名管理人员。我们假设他不再需要访问 `share/developers` 目录中的文档。作为管理员，John 将 Zhang 从 Managers 用户组移至 Developers 用户组。通过简单的重新分配，Zhang 将自动获得所有授予给 Managers 用户组的权限，但将无法再访问 `share/developers` 目录中的数据。

与第三方企业集成

组织经常与合作公司、顾问及承包商合作。Example Corp 是 Widget Company 的合作伙伴，而 Widget Company 的员工 Shirley 需要将数据放入存储桶中，以供 Example Corp 使用。John 创建了一个名为 WidgetCo 的用户组和名为 shirley 的用户，并将 Shirley 添加至 WidgetCo 用户组。John 还创建了一个名为 `aws-s3-bucket1` 的专用存储桶以供 Shirley 使用。

John 更新现有策略或添加新的策略来满足合作伙伴 Widget Company 的需求。例如，John 可新建用于拒绝 WidgetCo 用户组成员使用任何操作（写入操作除外）的策略。除非有一个广泛的策略，授予所有用户访问大量 Amazon S3 操作的许可，否则此策略将非常必要。

适用于 Amazon Identity and Access Management Access Analyzer 的 Amazon 托管策略

要向用户、组 and 角色添加权限，与自己编写策略相比，使用 Amazon 托管策略更简单。创建仅为团队提供所需权限的 [IAM 客户托管策略](#) 需要时间和专业知识。要快速入门，您可以使用我们的 Amazon 托管策略。这些策略涵盖常见使用案例，可在您的 Amazon 账户中使用。有关 Amazon 托管策略的更多信息，请参阅 IAM 用户指南中的 [Amazon 托管策略](#)。

Amazon 服务负责维护和更新 Amazon 托管策略。您无法更改 Amazon 托管策略中的权限。服务偶尔会向 Amazon 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，服务最有可能更新 Amazon 托管策略。服务不会从 Amazon 托管策略中删除权限，因此策略更新不会破坏您的现有权限。

此外，Amazon 还支持跨多种服务的工作职能的托管策略。例如，ReadOnlyAccess Amazon 托管策略提供对所有 Amazon 服务和资源的只读访问权限。当服务启动新功能时，Amazon 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 Amazon 托管策略](#)。

IAMReadOnlyAccess

使用 IAMReadOnlyAccess 托管策略来允许对 IAM 资源的只读权限。此策略授予权限，使其能够获取和列出所有 IAM 资源。它允许查看用户、组、角色、策略、身份提供程序和 MFA 设备的详细信息和活动报告。它不包括创建或删除资源或访问 IAM Access Analyzer 资源的能力。有关此策略支持的服务和操作的完整列表，请查看此 [policy \(策略\)](#)。

IAMUserChangePassword

使用 IAMUserChangePassword 托管策略可允许 IAM 用户更改他们的密码。

通过配置 IAM Account 设置和 Password 策略，IAM 用户可以修改自己的 IAM 账户密码。当您允许此操作时，IAM 会向每个用户附加以下策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:ChangePassword"
      ],
      "Resource": [
        "arn:aws:iam::*:user/${aws:username}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetAccountPasswordPolicy"
      ],
      "Resource": "*"
    }
  ]
}
```

IAMAccessAnalyzerFullAccess

使用 IAMAccessAnalyzerFullAccess Amazon 托管策略，以允许您的管理员访问 IAM Access Analyzer。

权限分组

此策略根据提供的权限集分为多个语句。

- IAM Access Analyzer — 允许对 IAM Access Analyzer 中所有资源的完全管理权限。
- 创建服务链接角色 — 允许管理员创建[服务关联角色](#)，它允许 IAM Access Analyzer 代表您分析其他服务中的资源。此权限允许创建仅供 IAM Access Analyzer 使用的服务关联角色。

- Amazon Organizations — 允许管理员将 IAM Access Analyzer 用于 Amazon Organizations 中的企业。在 Amazon Organizations 为 IAM Access Analyzer [启用可信访问权限](#)后，管理账户的成员可以查看整个企业的结果。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "access-analyzer:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": "access-analyzer.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization",
        "organizations:DescribeOrganizationalUnit",
        "organizations:ListAccounts",
        "organizations:ListAccountsForParent",
        "organizations:ListAWSServiceAccessForOrganization",
        "organizations:ListChildren",
        "organizations:ListDelegatedAdministrators",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:ListParents",
        "organizations:ListRoots"
      ],
      "Resource": "*"
    }
  ]
}
```

IAMAccessAnalyzerReadOnlyAccess

要允许对 IAM Access Analyzer 进行只读访问，请使用 IAMAccessAnalyzerReadOnlyAccess Amazon 托管策略。

要同时允许对用于 Amazon Organizations 的 IAM Access Analyzer 进行只读访问中，请创建一个客户托管策略，以允许来自 [IAMAccessAnalyzerFullAccess](#) (p. 758) Amazon 托管策略的“Describe”（描述）和“List”（列示）操作。

服务级别权限

此策略提供对 IAM Access Analyzer 的只读访问权限。此策略中不包含其他服务权限。

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

```
{
  "Effect": "Allow",
  "Action": [
    "access-analyzer:Get*",
    "access-analyzer:List*",
    "access-analyzer:ValidatePolicy"
  ],
  "Resource": "*"
}
```

AccessAnalyzerServiceRolePolicy

您无法将 AccessAnalyzerServiceRolePolicy 附加到您的 IAM 实体。此附加到服务相关角色的策略允许 IAM Access Analyzer 代表您执行操作。有关更多信息，请参阅[将服务相关角色用于 Amazon Identity and Access Management Access Analyzer](#)。

服务级别权限

该策略允许访问 IAM Access Analyzer，以分析来自多个 Amazon Web Services 的资源元数据。该策略还允许对 Amazon Organizations 的权限，并允许在作为信任区域的 Amazon 企业中创建分析程序。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeAddresses",
        "ec2:DescribeByoipCidrs",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcs",
        "iam:GetRole",
        "iam:ListRoles",
        "kms:DescribeKey",
        "kms:GetKeyPolicy",
        "kms:ListGrants",
        "kms:ListKeyPolicies",
        "kms:ListKeys",
        "lambda:GetFunctionUrlConfig",
        "lambda:GetLayerVersionPolicy",
        "lambda:GetPolicy",
        "lambda:ListAliases",
        "lambda:ListFunctions",
        "lambda:ListLayers",
        "lambda:ListLayerVersions",
        "lambda:ListVersionsByFunction",
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization",
        "organizations:DescribeOrganizationalUnit",
        "organizations:ListAccounts",
        "organizations:ListAccountsForParent",
        "organizations:ListAWSServiceAccessForOrganization",
        "organizations:ListChildren",
        "organizations:ListDelegatedAdministrators",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:ListParents",
        "organizations:ListRoots",
        "s3:DescribeMultiRegionAccessPointOperation",
        "s3:GetAccessPoint",
        "s3:GetAccessPointPolicy",

```

```

    "s3:GetAccessPointPolicyStatus",
    "s3:GetAccountPublicAccessBlock",
    "s3:GetBucketAcl",
    "s3:GetBucketLocation",
    "s3:GetBucketPolicyStatus",
    "s3:GetBucketPolicy",
    "s3:GetBucketPublicAccessBlock",
    "s3:GetMultiRegionAccessPoint",
    "s3:GetMultiRegionAccessPointPolicy",
    "s3:GetMultiRegionAccessPointPolicyStatus",
    "s3:ListAccessPoints",
    "s3:ListAllMyBuckets",
    "s3:ListMultiRegionAccessPoints",
    "sns:GetTopicAttributes",
    "sns:ListTopics",
    "secretsmanager:DescribeSecret",
    "secretsmanager:GetResourcePolicy",
    "secretsmanager:ListSecrets",
    "sqs:GetQueueAttributes",
    "sqs:ListQueues"
  ],
  "Resource": "*"
}
]
}

```

IAM 和 IAM Access Analyzer 更新 Amazon 托管策略

查看有关 IAM 和 Amazon 托管策略更新的详细信息（从这些服务开始跟踪这些更改开始）。有关此页面更改的自动警报，请订阅 IAM 和 IAM Access Analyzer 文档历史记录页面上的 RSS 源。

更改	说明	日期
AccessAnalyzerServiceRolePolicy – 添加权限	IAM Access Analyzer 添加了对 <code>AccessAnalyzerServiceRolePolicy</code> 的服务级别权限的 <code>lambda:GetFunctionUrlConfig</code> 操作。	2022 年 4 月 6 日
AccessAnalyzerServiceRolePolicy – 添加权限	IAM Access Analyzer 添加了新的 Amazon S3 操作来分析与多区域访问点关联的元数据。	2021 年 9 月 2 日
IAMAccessAnalyzerReadOnlyAccessPolicy – 添加权限	IAM Access Analyzer 添加了一个新操作以授予 <code>ValidatePolicy</code> 权限，以允许您使用策略检查进行验证。 IAM Access Analyzer 需要此权限才能对您的策略执行策略检查。	2021 年 3 月 16 日
IAM Access Analyzer 开始跟踪更改	IAM Access Analyzer 开始为其 Amazon 托管策略跟踪更改。	2021 年 3 月 1 日

使用 Amazon Identity and Access Management Access Analyzer

Amazon IAM Access Analyzer 提供以下功能：

- IAM Access Analyzer 帮助您[标识企业和账户中与外部实体共享的资源 \(p. 762\)](#)。
- IAM Access Analyzer 将根据策略语法和最佳实践验证 IAM policy (p. 763)。
- IAM Access Analyzer 将根据您的 Amazon CloudTrail 日志中的访问活动[生成 IAM policy \(p. 763\)](#)。

识别与外部实体共享的资源

IAM Access Analyzer 帮助您标识企业和账户中与外部实体共享的资源，例如 Amazon S3 存储桶或 IAM 角色。这可以帮助您识别对资源和数据的意外访问，此类访问会带来安全风险。IAM Access Analyzer 使用基于逻辑的推理来分析 Amazon 环境中基于资源的策略，确定与外部主体共享的资源。对于在您的账户外共享的资源的每个实例，IAM Access Analyzer 都会生成一个调查结果。调查结果包括有关访问权限以及该访问权限授予到的外部主体的信息。您可以查看调查结果，以确定该访问权限是否按计划授予且安全，或者该访问权限是否是计划外的访问权限并存在安全风险。除了帮助您识别与外部实体共享的资源外，您还可以使用 IAM Access Analyzer 结果预览策略如何影响对资源的公共和跨账户访问，然后再部署资源权限。

Note

外部实体可以是另一个 Amazon 账户、根用户、IAM 用户或角色、联合身份用户、Amazon 服务、匿名用户或可用于创建筛选器的其他实体。有关更多信息，请参阅 [Amazon JSON 策略元素：主体](#)。

启用 IAM Access Analyzer 后，可以为整个企业或账户创建分析器。您选择的组织或账户称为分析器的信任区。该分析器将监控信任区域内所有[受支持的资源 \(p. 770\)](#)。信任区域内的主体对资源的任何访问都被视为受信任的。启用后，IAM Access Analyzer 将分析应用于您的信任区中所有受支持的资源的策略。在进行第一次分析后，IAM Access Analyzer 将定期分析这些策略。如果添加了新策略或更改了现有策略，IAM Access Analyzer 会在约 30 分钟内分析新策略或更新后的策略。

在分析策略时，如果 IAM Access Analyzer 发现一个向不在信任区域内的外部主体授予访问权限的策略，则会生成一个结果。每个结果均包含有关资源、有权访问该资源的外部实体以及授予的权限的详细信息，以便您能采取适当的操作。您可以查看查找结果中包含的详细信息，以确定资源访问是有意的还是应解决的潜在风险。在向资源添加策略或更新现有策略时，IAM Access Analyzer 将对策略进行分析。IAM Access Analyzer 还会定期分析所有基于资源的策略。

在某些情况下，IAM Access Analyzer 很少会收到添加或更新了策略的通知。如果您创建或删除与 S3 存储桶关联的多区域访问点，或者更新多区域访问点的策略，IAM Access Analyzer 可能需要长达 6 小时才能生成或解析结果。此外，如果 Amazon CloudTrail 日志传输存在传输问题，则策略更改不会触发对结果中报告的资源进行重新扫描。在发生此情况时，IAM Access Analyzer 会在下一个定期扫描期间（24 小时内）分析新策略或更新后的策略。如果要确认对策略所做的更改是否可以解决查找结果中报告的访问问题，您可以通过使用 Finding（结果）详情页面中的 Rescan（重新扫描）链接，或通过使用 IAM Access Analyzer API 的 [StartResourceScan](#) 操作，以重新扫描查找结果中报告的资源。要了解更多信息，请参阅[“解决结果 \(p. 770\)”](#)。

Important

IAM Access Analyzer 仅分析应用于与启用该功能所处同一 Amazon 区域中资源的策略。要监控 Amazon 环境中的所有资源，您必须创建一个分析器，以便在您使用受支持的 Amazon 资源的每个区域中启用 IAM Access Analyzer。

IAM Access Analyzer 分析以下资源类型：

- [Amazon Simple Storage Service 存储桶 \(p. 771\)](#)
- [Amazon Identity and Access Management 角色 \(p. 771\)](#)
- [Amazon Key Management Service 密钥 \(p. 771\)](#)
- [Amazon Lambda 函数和层 \(p. 772\)](#)
- [Amazon Simple Queue Service 队列 \(p. 772\)](#)
- [Amazon Secrets Manager 密钥 \(p. 772\)](#)

验证策略

您可以使用 IAM Access Analyzer 策略检查来验证您的策略。您可以在 IAM 控制台中使用 Amazon CLI、Amazon API 或 JSON 策略编辑器创建或编辑策略。IAM Access Analyzer 将根据 IAM [policy 语法 \(p. 985\)](#) 和 [最佳实践 \(p. 752\)](#) 验证您的策略。您可以查看策略验证检查结果，其中包括策略的安全警告、错误、常规警告和策略建议。这些结果提供了可操作的建议，可帮助您编写可操作且符合安全最佳实践的策略。要了解有关使用 IAM Access Analyzer 验证策略的详细信息，请参阅 [IAM Access Analyzer 策略验证 \(p. 791\)](#)。

生成策略

IAM Access Analyzer 分析 Amazon CloudTrail 日志以识别指定日期范围内 IAM 实体（用户或角色）已使用的操作和服务。然后，它会生成基于该访问活动的 IAM policy。您可以使用生成的策略通过将实体的权限附加到 IAM 用户或角色来优化实体权限。要了解有关使用访问 IAM Access Analyzer 生成策略的详细信息，请参阅 [IAM Access Analyzer 策略生成 \(p. 852\)](#)。

有关公共和跨账户存取的结果

IAM Access Analyzer 为基于资源的策略的每个实例生成一个结果，该策略向不在信任区域内的主体授予对信任区域内的资源的访问权限。在创建分析器时，您可以选择要分析的企业或 Amazon 账户。您为分析器选择的组织或账户中的任何主体都被视为受信任。由于同一组织或账户中的主体是受信任的，因此组织或账户中的资源和主体构成了分析器的信任区域。信任区域内的任何共享都被视为是安全的，因此 IAM Access Analyzer 不会生成结果。例如，如果选择某个组织作为分析器的信任区域，则该组织中的所有资源和主体都在信任区域内。如果您将某个企业成员账户中的 S3 存储桶的权限授予另一个企业成员账户中的主体，则 IAM Access Analyzer 不会生成结果。但是，如果您向非企业成员账户中的主体授予权限，则 IAM Access Analyzer 会生成结果。

主题

- [IAM Access Analyzer 结果的工作原理 \(p. 764\)](#)
- [开始使用 Amazon Identity and Access Management Access Analyzer 结果 \(p. 764\)](#)
- [处理结果 \(p. 766\)](#)
- [查看结果 \(p. 766\)](#)
- [筛选结果 \(p. 768\)](#)
- [对结果进行存档 \(p. 770\)](#)
- [解决结果 \(p. 770\)](#)
- [IAM Access Analyzer 资源类型 \(p. 770\)](#)
- [IAM Access Analyzer 的设置 \(p. 773\)](#)

- [存档规则 \(p. 773\)](#)
- [使用 Amazon EventBridge 监控 Amazon Identity and Access Management Access Analyzer \(p. 775\)](#)
- [与 Amazon Security Hub 集成 \(p. 779\)](#)
- [使用 Amazon CloudTrail 记录 IAM Access Analyzer API 调用 \(p. 782\)](#)
- [IAM Access Analyzer 筛选条件键 \(p. 784\)](#)
- [将服务相关角色用于 Amazon Identity and Access Management Access Analyzer \(p. 786\)](#)

IAM Access Analyzer 结果的工作原理

本主题介绍了 IAM Access Analyzer 中用于帮助您熟悉访问分析器如何监控对 Amazon 资源的访问的概念和术语。

Amazon Identity and Access Management Access Analyzer 基于 [Zelkova](#) 构建，它将 IAM Access Analyzer 转换为等效逻辑语句，并针对问题运行一套通用和专门逻辑求解器（可满足性理论）。IAM Access Analyzer 反复将 Zelkova 应用于具有越来越具体查询的策略，以根据策略的内容表征策略允许的行为类别。要了解有关可满足性理论的更多信息，请参阅[可满足性理论](#)。

IAM Access Analyzer 不会检查访问日志来确定外部实体是否已访问信任区域内的资源。当基于资源的策略允许访问资源时，会生成一个结果，即使外部实体未访问资源也是如此。IAM Access Analyzer 在做出决定时也不考虑任何外部账户的状态。也就是说，它指示账户 11112222333 可以访问您的 S3 存储桶，但它完全不知道该账户中的用户、角色、服务控制策略 (SCP) 和其他相关配置的状态。这是为了保护客户隐私 – IAM Access Analyzer 不考虑其他账户的拥有者。这也是出于安全考虑——如果该账户不由 IAM Access Analyzer 客户拥有，则即使该账户中当前没有主体可以访问外部资源，外部实体也可以获取对其资源的访问权限，了解这一点仍然很重要。

IAM Access Analyzer 仅考虑外部用户不能直接影响的或对授权有影响的某些 IAM 条件键。有关 IAM Access Analyzer 考虑的条件键的示例，请参阅 [IAM Access Analyzer 筛选条件键 \(p. 784\)](#)。

IAM Access Analyzer 当前不会报告来自 Amazon 服务主体或内部服务账户的结果。在极少数情况下，IAM Access Analyzer 无法完全确定策略语句是否授予对外部实体的访问权限，它会错误地声明一个假阳性结果。IAM Access Analyzer 旨在提供您账户中资源共享的全面视图，并努力减少假阴性结果。

开始使用 Amazon Identity and Access Management Access Analyzer 结果

通过本主题中的信息了解使用和管理 Amazon Identity and Access Management Access Analyzer 的要求，以及如何启用 IAM Access Analyzer。要了解有关 IAM Access Analyzer 的服务相关角色的更多信息，请参阅 [将服务相关角色用于 Amazon Identity and Access Management Access Analyzer \(p. 786\)](#)。

使用 IAM Access Analyzer 所需的权限

要成功配置和使用 IAM Access Analyzer，您使用的账户必须获得所需的权限。

IAM Access Analyzer 的 Amazon 托管策略

Amazon Identity and Access Management Access Analyzer 提供 Amazon 托管策略，可帮助您快速入门。

- `IAMAccessAnalyzerFullAccess` — 允许管理员完全访问 IAM Access Analyzer 的权限。此策略还允许创建与服务关联的角色，以便允许 IAM Access Analyzer 分析您的账户或 Amazon 企业。
- `IAMAccessAnalyzerReadOnlyAccess` — 允许对 IAM Access Analyzer 的只读访问权限。您必须将其他策略添加到 IAM 身份（用户、用户组或角色），以允许他们查看他们的发现结果。

Amazon Identity and Access Management Access Analyzer 定义的资源

若要查看 IAM Access Analyzer 定义的资源，请参阅服务授权参考中的 [Amazon Identity and Access Management Access Analyzer 定义的资源类型](#)。

所需的 IAM Access Analyzer 服务权限

IAM Access Analyzer 使用名为 `AWSServiceRoleForAccessAnalyzer` 的服务相关角色授予服务只读访问权限，以代表您使用基于资源的策略分析 Amazon 资源。在创建将账户作为信任区域的分析器时，此服务会为您的账户创建角色。有关更多信息，请参阅[将服务相关角色用于 Amazon Identity and Access Management Access Analyzer \(p. 786\)](#)。

Note

IAM Access Analyzer 是区域性的。您必须在每个区域内单独启用 IAM Access Analyzer。

在某些情况下，启用 IAM Access Analyzer 后，Findings (结果) 页将加载，但不显示任何结果。此情况可能是因用于填充结果的控制台中出现延迟导致的。您需要手动刷新浏览器才能查看您的结果。如果您仍未看到任何结果，则原因是您的账户中没有可供外部实体访问的受支持的资源。如果将向外部实体授予访问权限的策略应用于资源，则 IAM Access Analyzer 会生成结果。

Note

在修改策略后，IAM Access Analyzer 可能最多需要 30 分钟来分析资源，然后生成新的结果或更新现有结果以便访问资源。

启用 IAM Access Analyzer

要在一个区域中启用 IAM Access Analyzer，您必须在该区域中创建分析器。您必须在要监控对资源的访问的每个区域中创建一个分析器。

创建将账户作为可信区域的分析器

1. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 选择 Access analyzer (访问分析器)。
3. 选择 Create analyzer (创建分析器)。
4. 在 Create analyzer (创建分析器) 页面上，确认显示的区域是要启用 IAM Access Analyzer 的区域。
5. 输入分析器的名称。
6. 请选择该账户作为分析器的信任区域。

Note

如果您的账户不是 Amazon Organizations 管理账户或[委托管理员 \(p. 773\)](#)账户，则您只能创建一个将您的账户作为信任区域的分析器。

7. 可选。添加要应用于分析器的所有标签。
8. 选择 Create Analyzer (创建分析器)。

在创建分析器来启用 IAM Access Analyzer 时，会在您的账户中创建一个名为 `AWSServiceRoleForAccessAnalyzer` 的服务相关角色。

创建将组织作为信任区域的分析器

1. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 选择 Access analyzer (访问分析器)。

3. 选择 Create analyzer (创建分析器)。
4. 在 Create analyzer (创建分析器) 页面上，确认显示的区域是要启用 IAM Access Analyzer 的区域。
5. 输入分析器的名称。
6. 选择您的组织作为分析器的信任区域。
7. 可选。添加要应用于分析器的所有标签。
8. 选择 Create Analyzer (创建分析器)。

在创建将组织作为信任区域的分析器时，系统会在组织的每个账户中创建一个名为 `AWSServiceRoleForAccessAnalyzer` 的服务相关角色。

IAM Access Analyzer 状态

要查看分析器的状态，请选择 Analyzers (分析器)。为组织或账户创建的分析器可具有以下状态：

状态	描述
处于活动状态	分析器将主动监控其信任区域内的资源。分析器会主动生成新的结果并更新现有结果。
创建	分析器的创建过程仍在进行中。在创建过程完成后，分析器将变为活动状态。
已禁用	分析器因 Amazon Organizations 管理员执行的操作而被禁用。例如，删除作为 IAM Access Analyzer 的委派管理员的分析器的账户。当分析器处于禁用状态时，它不会生成新的结果或更新现有结果。
失败	由于配置问题，分析器创建失败。分析器将不会生成任何结果。删除该分析器并创建新的分析器。

处理结果

仅为在信任区域之外共享的资源的每个实例生成一次结果。每当修改基于资源的策略时，IAM Access Analyzer 都会分析策略。如果更新的策略共享结果中已标识的资源，但具有不同的权限或条件，则会为该资源共享实例生成新结果。如果删除第一个结果中的访问，则该结果将更新为“Resolved (已解决)”状态。

所有结果的状态都将保持为“Active (活动)”状态，直到您对这些结果进行存档或删除生成结果的访问。在删除访问时，结果状态将更新为“Resolved (已解决)”。

Note

在修改策略后，IAM Access Analyzer 可能最多需要 30 分钟来分析资源并更新结果。

您应查看账户中的所有结果，以确定是否需要并批准共享。如果需要结果中已标识的共享，您可以对结果进行存档。在对结果进行存档后，其状态将变为“Archived (已存档)”，并且该结果将从活动结果列表中移除。该结果不会被删除。您随时可以查看已存档的结果。处理您账户中的所有结果，直到不存在活动结果。在不存在活动结果后，您便能知道生成的任何新活动结果都来自环境中的最近更改。

查看结果

启用 [IAM Access Analyzer \(p. 765\)](#) 后，下一步是查看所有结果以确定结果中标识的访问是有意的还是无意的。您还可以查看结果以确定预期访问的常见结果，然后 [创建存档规则 \(p. 773\)](#) 以自动对这些结果进行存档。此外，您还可以查看已存档和已解决的结果。

查看结果

1. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 选择 Access analyzer (访问分析器)。

Note

仅在您有权查看分析器的结果时，才会显示结果。

将为分析器显示所有活动结果。要查看由分析器生成的其他结果，请选择相应的选项卡：

- 选择 Active (活动) 可查看分析器已生成的所有活动结果。
- 选择 Archived (已存档) 可仅查看已存档的由分析器生成的结果。要了解更多信息，请参阅“[对结果进行存档 \(p. 770\)](#)”。
- 选择 Resolved (已解决) 可仅查看已解决的由分析器生成的结果。修复生成结果的问题后，结果状态会变为“Resolved (已解决)”。

Important

已解决的结果将在上次更新后 90 天被删除。除非您删除生成活动结果和已存档结果的分析器，否则不会删除这些结果。

- 选择 All (全部) 可查看分析器生成的具有任何状态的所有结果。

Findings (结果) 页显示有关生成结果的共享资源和策略语句的以下详细信息：

结果 ID

分配给结果的唯一 ID。选择结果 ID 可显示有关生成结果的资源和策略语句的其他详细信息。

资源

已应用策略的资源的类型和部分名称，该策略向不在信任区域内的外部实体授予访问权限。

Resource owner account (资源拥有者账户)

仅当您使用组织作为信任区域时，才会显示此列。组织中拥有结果中报告的资源的账户。

External principal (外部主体)

分析的策略向其授予访问权限的主体（不在您的信任区域内）。有效值包括：

- Amazon 账户 - 列出的 Amazon 账户中拥有该账户管理员权限的所有主体都可以访问资源。
- Any principal (任何主体) - 任何 Amazon 账户中符合 Conditions (条件) 列中包含的条件的所有主体都有权访问资源。例如，如果列出了一个 VPC，则意味着任何账户中有权访问该 VPC 的任何主体都能访问资源。
- Canonical user (规范用户) - Amazon 账户中具有列出的规范用户 ID 的所有主体都有权访问资源。
- IAM role (IAM 角色) - 列出的 IAM 角色有权访问资源。
- IAM user (IAM 用户) - 列出的 IAM 用户有权访问资源。

Condition

策略语句中用于授予访问权限的条件。例如，如果 Condition (条件) 字段包含 Source VPC (源 VPC)，则意味着将与有权访问列出的 VPC 的主体共享资源。条件可以是全局性的，也可以是服务特定的。[全局条件键](#)具有 aws: 前缀。

Shared through (共享方式)

Shared through (共享方式) 字段指定如何授予生成结果的访问权限。有效值包括：

- 存储桶策略 — 附加到 Amazon S3 存储桶的存储桶策略。

- Access control list (访问控制列表) - 附加到 Amazon S3 存储桶的访问控制列表 (ACL) 。
- 访问点 — 与 Amazon S3 存储桶关联的访问点或多区域访问点。访问点的 ARN 显示在 Findings (结果) 详细信息中。

访问级别

基于资源的策略中的操作向外部实体授予的访问权限的级别。查看结果的详细信息以了解更多信息。访问级别值包括：

- List (列出) - 列出服务内的资源以确定某个对象是否存在的权限。此访问权限级别的操作可以列出对象，但是看不到资源的内容。
- Read (读取) - 读取服务中资源的内容和属性但不对其进行编辑的权限。
- Write (写入) - 在服务中创建、删除或修改资源的权限。
- Permissions (权限) - 在服务中授予或修改资源权限的权限。
- Tagging (标记) - 执行仅更改资源标签状态的操作的权限。

Updated

结果状态的最近更新的时间戳，或生成结果的日期和时间（如果未进行更新）。

Note

在修改策略后，IAM Access Analyzer 可能最多需要 30 分钟来再次分析资源并更新结果。

状态

结果的状态为 Active (活动)、Archived (已存档) 或 Resolved (已解决)。

筛选结果

该页面的默认筛选是显示所有活动结果。要查看已存档结果，请选择 Archived (已存档) 选项卡。首次开始使用 IAM Access Analyzer 时，没有已存档的结果。

可以使用筛选器来仅显示特定资源、账户、主体或其他值的结果。要创建筛选器，请选择要作为筛选条件的属性，然后选择要作为筛选条件的属性值。例如，要创建仅显示特定 Amazon 账户的结果的筛选器，请选择 AmazonAccount (亚马逊云科技账户) 作为属性，然后输入要查看其结果的 Amazon 账户的账号。要创建一个筛选条件以仅显示允许公有访问的资源的结果，可以选择公有访问属性，然后选择公共访问: true。

有关可用于创建或更新存档规则的筛选条件键的列表，请参阅 [IAM Access Analyzer 筛选条件键 \(p. 784\)](#)。

筛选显示的结果

1. 选择 Filter active findings (筛选活动结果) 字段。
2. 请选择要用于筛选所显示结果的属性。
3. 为该属性选择要匹配的值。仅显示具有该结果中的值的结果。

例如，如果您选择 Resource (资源) 作为属性，请键入存储桶的部分或完整名称，然后按 Enter 键。仅显示与筛选条件匹配的存储桶的结果。

您可以添加其他属性来进一步筛选显示的结果。在添加其他属性时，仅显示与筛选器中的所有条件匹配的结果。无法定义筛选器来显示与一个属性或另一个属性匹配的结果。

仅当您查看将组织作为信任区域的分析器的结果时，某些字段才会显示。

可使用以下属性定义筛选器：

- Public access (公有访问) - 要按结果筛选允许公有访问的资源，请按公有访问进行筛选，然后选择 Public access: true (公有访问: true) 。

- Resource (资源) - 要按资源进行筛选, 请键入资源的完整或部分名称。
- Resource Type (资源类型) - 要按资源类型进行筛选, 请从显示的列表中选择类型。
- Amazon Account (亚马逊云科技账户) - 使用此属性可按策略声明的主体部分中授予访问权限的 Amazon 账户进行筛选。要按 Amazon 账户进行筛选, 请键入完整的 12 位 Amazon 账户 ID 或其一部分, 或者键入有权访问当前账户中资源的外部 Amazon 用户或角色的完整账户 ARN 或其一部分。
- Canonical User (规范用户) - 要按规范用户进行筛选, 请键入为 S3 存储桶定义的规范用户 ID。要了解更多信息, 请参阅 [Amazon 账户标识符](#)。
- Federated User (联合身份用户) - 要按联合身份用户进行筛选, 请键入联合身份身份的完整或部分 ARN。要了解更多信息, 请参阅 [身份提供商和联合身份验证](#)。
- Principal ARN (主体 ARN) - 使用此属性可筛选 aws:PrincipalArn 条件键中使用的主体 (IAM 用户、角色或组) 的 ARN。要按主体 ARN 进行筛选, 请键入结果中报告的外部 Amazon 账户中的 IAM 用户、角色或组的完整或部分 ARN。
- Principal OrgID (主体 OrgID) - 要按主体 OrgID 进行筛选, 请键入与属于 Amazon 企业 (指定为结果中的条件) 的外部主体关联的企业 ID 的完整或部分 ID。要了解更多信息, 请参阅 [Amazon 全局条件上下文键](#)。
- Principal Org Paths (主体企业路径) - 要按主体企业路径进行筛选, 请键入 Amazon 企业或企业单位 (OU) 的完整或部分 ID, 该 ID 允许访问属于指定为策略中的条件的企业或 OU 的账户成员的所有外部主体。要了解更多信息, 请参阅 [Amazon 全局条件上下文键](#)。
- Source Account (源账户) - 要按源账户进行筛选, 请键入与资源关联的 Amazon 账户 ID 的完整或部分 ID, 如同 Amazon 中的某些跨服务权限中使用的那样。
- Source ARN (源 ARN) - 要按源 ARN 进行筛选, 请键入指定为结果中的条件的完整或部分 ARN。要了解更多信息, 请参阅“按主体企业路径进行筛选”, 键入 Amazon 企业或企业单位 (OU) 的完整或部分 ID, 该 ID 允许访问属于指定为策略中的条件的企业或 OU 的账户成员的所有外部主体。要了解更多信息, 请参阅 [Amazon 全局条件上下文键](#)。
- Source IP (源 IP) - 要按源 IP 进行筛选, 请键入允许外部实体在使用指定 IP 地址时访问当前账户中的资源的完整或部分 IP 地址。要了解更多信息, 请参阅 [Amazon 全局条件上下文键](#)。
- Source VPC (源 VPC) - 要按源 VPC 进行筛选, 请键入允许外部实体在使用指定 VPC 时访问当前账户中的资源的全部或部分 VPC ID。要了解更多信息, 请参阅 [Amazon 全局条件上下文键](#)。
- Source VPCE (源 VPCE) - 要按源 VPCE 进行筛选, 请键入完整或部分 VPC 终端节点 ID, 此 ID 允许外部实体在使用指定 VPC 终端节点时访问当前账户中的资源。要了解更多信息, 请参阅 [Amazon 全局条件上下文键](#)。
- User ID (用户 ID) - 要按用户 ID 进行筛选, 请键入外部 Amazon 账户中有权访问当前账户中的资源的 IAM 用户的完整或部分用户 ID。要了解更多信息, 请参阅 [Amazon 全局条件上下文键](#)。
- KMS Key ID (KMS 密钥 ID) - 要按 KMS 密钥 ID 进行筛选, 请键入 KMS 密钥的完整或部分密钥 ID, 该密钥指定为您当前账户中 KMS 加密的 S3 对象访问的条件。
- Google Audience (Google 受众) - 要按 Google 受众进行筛选, 请键入指定为当前账户中的 IAM 角色访问条件的 Google 应用程序 ID 的完整或部分 ID。要了解更多信息, 请参阅 [IAM 和 Amazon STS 条件上下文键](#)。
- Cognito Audience (Cognito 受众) - 要按 Cognito 受众进行筛选, 请键入指定为您当前账户中的 IAM 角色访问条件的完整或部分 Amazon Cognito 身份池 ID。要了解更多信息, 请参阅 [IAM 和 Amazon STS 条件上下文键](#)。
- Caller Account (调用方账户) - 拥有或包含调用实体 (如 IAM 角色、用户或账户根用户) 的账户的 Amazon 账户 ID。这将由调用 KMS 的服务使用。要按调用方账户进行筛选, 请键入完整或部分 Amazon 账户 ID。
- Facebook App ID (Facebook 应用程序 ID) - 要按 Facebook 应用程序 ID 进行筛选, 请键入完整或部分 Facebook 应用程序 ID (或站点 ID), 此 ID 指定为条件以允许使用 Login with Facebook 联合身份验证访问您当前账户中的 IAM 角色。要了解更多信息, 请参阅 [IAM 和 Amazon STS 条件上下文键](#) 中的 id 部分。
- Amazon App ID (Amazon 应用程序 ID) - 要按 Amazon 应用程序 ID 进行筛选, 请键入完整或部分 Amazon 应用程序 ID (或站点 ID), 此 ID 指定为条件以允许使用“以 Amazon 登录”联合身份验证访问您当前账户中的 IAM 角色。要了解更多信息, 请参阅 [IAM 和 Amazon STS 条件上下文键](#) 中的 id 部分。
- Lambda Event Source Token (Lambda 事件源令牌) - 要按随 Alexa 集成传入的 Lambda 事件源令牌进行筛选, 请键入完整或部分令牌字符串。

对结果进行存档

在获得有意访问资源的结果（例如，多个用户用于批准的工作流程的 IAM 角色）时，您可以对该结果进行存档。在对一个结果进行存档时，将从活动结果列表中清除该结果，从而让您专注于需要解决的结果。存档的结果不会被删除。您可以对“Findings (结果)”页进行筛选，以显示已存档的结果，并随时取消对结果的存档。

从 Findings (结果) 页对结果进行存档

1. 选中要存档的一个或多个结果旁边的复选框。
2. 选择 Archive (存档)。

屏幕顶部将显示一条确认消息。

从 Findings Details (结果详细信息) 页对结果进行存档。

1. 为要存档的结果选择 Finding ID (结果 ID)。
2. 选择 Archive (存档)。

屏幕顶部将显示一条确认消息。

要取消对结果的存档，请重复之前的步骤，但选择 Unarchive (取消存档) 而非 Archive (存档)。在取消对结果的存档时，状态将设置为“Active (活动)”。

解决结果

要解决通过您不打算允许的访问生成的结果，请修改策略语句以删除允许访问已标识资源的权限。例如，对于 S3 存储桶上的结果，请使用 Amazon S3 控制台配置该存储桶上的权限。对于 IAM 角色，请使用 IAM 控制台为列出的 IAM 角色 [修改信任策略](#)。使用其他受支持资源的控制台修改导致生成结果的策略语句。

在进行更改来解决结果（例如，修改应用于 IAM 角色的策略）后，IAM Access Analyzer 会重新扫描资源。如果资源不再在信任区域之外共享，则结果的状态将变为“Resolved (已解决)”。结果不再显示在 Active findings (活动结果) 表中，而是显示在 Resolved findings (已解决结果) 表中。

Note

这不适用于错误结果。当 IAM Access Analyzer 无法访问资源时，它会生成错误结果。如果您已解决导致 IAM Access Analyzer 无法访问资源的问题，则错误结果将被完全删除而不是变为已解决的结果。

如果您所做的更改导致资源在信任区域之外通过其他方式进行共享（例如，通过不同的主体或其他权限），则 IAM Access Analyzer 会生成新的活动结果。

Note

在修改策略后，IAM Access Analyzer 可能最多需要 30 分钟来再次分析资源并更新结果。已解决的结果将在上次更新到查找状态后 90 天被删除。

IAM Access Analyzer 资源类型

IAM Access Analyzer 分析应用于您已启用 IAM Access Analyzer 的区域中 Amazon 资源的基于资源的策略。仅分析基于资源的策略。查看有关每个资源的信息以了解有关 IAM Access Analyzer 如何为每个资源类型生成结果的详细信息。

支持的资源类型：

- [Amazon Simple Storage Service 存储桶 \(p. 771\)](#)

- [Amazon Identity and Access Management 角色 \(p. 771\)](#)
- [Amazon Key Management Service 密钥 \(p. 771\)](#)
- [Amazon Lambda 函数和层 \(p. 772\)](#)
- [Amazon Simple Queue Service 队列 \(p. 772\)](#)
- [Amazon Secrets Manager 密钥 \(p. 772\)](#)

Amazon Simple Storage Service 存储桶

当 IAM Access Analyzer 分析 Amazon S3 存储桶时，它会在应用于存储桶的 Amazon S3 存储桶策略、ACL 或访问点（包括多区域访问点）向对外部实体授予访问权限时生成结果。外部实体是可用于[创建筛选器 \(p. 768\)](#)（该筛选器不在信任区域内）的主体或其他实体。例如，如果存储桶策略向其他账户授予访问权限或允许公共访问，则 IAM Access Analyzer 会生成结果。不过，如果您在存储桶上启用[阻止公共访问](#)设置，则可以阻止账户级别或存储桶级别的访问。

Amazon S3 块公共访问设置将覆盖应用于存储桶的存储桶策略。这些设置还将覆盖应用于存储桶访问点的访问点策略。IAM Access Analyzer 会在策略发生更改时在存储桶级别分析数据块公共访问设置。但是，它仅评估账户级别的阻止公共访问设置（每 6 小时一次）。这意味着，IAM Access Analyzer 可能在最多 6 小时内无法为对存储桶的公共访问生成或解决结果。例如，如果您有允许公共访问的存储桶策略，则 IAM Access Analyzer 会为该访问生成结果。如果您随后启用阻止公共访问来阻止在账户级别对存储桶进行的所有公共访问，则 IAM Access Analyzer 在最多 6 小时内不会解决存储桶策略的结果，即使已阻止对存储桶的所有公共访问。

对于多区域访问点，IAM Access Analyzer 会使用已建立的策略来生成查找结果。IAM Access Analyzer 每 6 小时评估一次对多区域访问点的更改。这意味着 IAM Access Analyzer 在 6 小时内不会生成或解析查找结果，即使您在此期间创建或删除了多区域访问点，或是更新了该访问点的策略。

Amazon Identity and Access Management 角色

对于 IAM 角色，IAM Access Analyzer 将分析[信任策略](#)。在角色信任策略中，您可以定义您信任代入该角色的主体。角色信任策略是附加到 IAM 中角色所必需的基于资源的策略。IAM Access Analyzer 为信任区域内可由信任区域外的外部实体访问的角色生成查询结果。

Note

IAM 角色是一类全局资源。如果角色信任策略向外部实体授予访问权限，则 IAM Access Analyzer 会在每个已启用的区域中生成结果。

Amazon Key Management Service 密钥

对于 Amazon KMS keys，IAM Access Analyzer 将分析应用于密钥的密钥策略和授权。如果密钥策略或授权允许外部实体访问密钥，则 IAM Access Analyzer 会生成查询结果。例如，如果您在策略语句中使用 `kms:CallerAccount` 条件键以允许访问特定 Amazon 账户中的所有用户，并且您指定了当前账户（当前分析器的信任区域）以外的账户，则 IAM Access Analyzer 会生成结果。要了解有关 IAM policy 语句中的 Amazon KMS 条件键的更多信息，请参阅[Amazon KMS 条件键](#)。

在 IAM Access Analyzer 分析 KMS 密钥时，它会读取密钥元数据，例如密钥策略和授权列表。如果密钥策略不允许 IAM Access Analyzer 角色读取密钥元数据，则会生成“Access Denied (访问被拒绝)”错误结果。例如，如果以下示例策略语句是应用于密钥的唯一策略，则会导致 IAM Access Analyzer 中出现“Access Denied (访问被拒绝)”错误结果：

```
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/Admin"
```

```
    },  
    "Action": "kms:*",  
    "Resource": "*" }  
}
```

由于此语句仅允许 Amazon 账户 111122223333 中名为 Admin 的角色访问密钥，因此会生成“Access Denied (访问被拒绝)”错误结果，原因是 IAM Access Analyzer 无法完全分析密钥。Findings (结果) 表中会显示一个错误结果 (红色文本)。该结果看起来类似于以下内容：

```
{  
  "error": "ACCESS_DENIED",  
  "id": "12345678-1234-abcd-dcba-111122223333",  
  "analyzedAt": "2019-09-16T14:24:33.352Z",  
  "resource": "arn:aws:kms:us-west-2:1234567890:key/1a2b3c4d-5e6f-7a8b-9c0d-1a2b3c4d5e6f7g8a",  
  "resourceType": "AWS::KMS::Key",  
  "status": "ACTIVE",  
  "updatedAt": "2019-09-16T14:24:33.352Z"  
}
```

在创建 KMS 密钥时，为访问密钥而授予的权限取决于您创建密钥的方式。如果您收到密钥资源的“Access Denied (访问被拒绝)”错误结果，请将以下策略语句应用于密钥资源以向 IAM Access Analyzer 授予对密钥的访问权限。

```
{  
  "Sid": "Allow IAM Access Analyzer access to key metadata",  
  "Effect": "Allow",  
  "Principal": {  
    "AWS": "arn:aws:iam::111122223333:role/aws-service-role/access-analyzer.amazonaws.com/AWSServiceRoleForAccessAnalyzer"  
  },  
  "Action": [  
    "kms:DescribeKey",  
    "kms:GetKeyPolicy",  
    "kms:List*" ]  
  "Resource": "*" }  
},
```

在收到 KMS 密钥资源的“Access Denied (访问被拒绝)”结果，然后通过更新密钥策略来解决此结果后，此结果的状态将更新为“Resolved (已解决)”。如果存在向外部实体授予密钥权限的策略语句或密钥授权，您可能会看到密钥资源的其他结果。

Amazon Lambda 函数和层

对于 Amazon Lambda 函数，IAM Access Analyzer 会分析策略，包括策略中的条件语句，该策略将向外部实体授予对函数的访问权限。IAM Access Analyzer 还会分析借助 EventSourceToken 使用 Amazon Lambda API 的 [AddPermission](#) 的操作时授予的权限。

Amazon Simple Queue Service 队列

对于 Amazon SQS 队列，IAM Access Analyzer 会分析允许外部实体访问队列的策略 (包括策略中的条件语句)。

Amazon Secrets Manager 密钥

对于 Amazon Secrets Manager 密钥，IAM Access Analyzer 会分析允许外部实体访问密钥的策略 (包括策略中的条件语句)。

IAM Access Analyzer 的设置

如果您在 Amazon Organizations 管理账户中配置 Amazon Identity and Access Management Access Analyzer，则可以在企业中添加成员账户作为委托管理员来管理企业的 IAM Access Analyzer。委托管理员有权创建和管理将组织作为信任区域的分析器。只有管理账户才能添加委派管理员。

IAM Access Analyzer 的委托管理员。

IAM Access Analyzer 的委托管理员是企业中的一个成员账户，该账户有权创建和管理将企业作为信任区域的分析器。只有管理账户才能添加、删除或更改委托管理员。

如果您添加了委派管理员，则可稍后更改为委派管理员的其他账户。在执行此操作时，以前的委托管理员账户将丢失对该账户创建的所有分析器（这些分析器将组织作为信任区域）的权限。这些分析器将变为禁用状态，并且不再生成新的结果或更新现有结果。这些分析器的现有结果也不再可供访问。您可在以后通过将账户配置为委派管理员来再次访问它们。如果您知道您不会使用与委派管理员相同的账户，请考虑在更改委派管理员之前删除分析器。这将删除生成的所有结果。当新的委派管理员创建新分析器时，会生成相同结果的新实例。您不会丢失任何结果，只是会在其他账户中为新分析器生成结果。您可以继续使用企业管理账户（也具有管理员权限）访问企业的结果。新的委托管理员必须为 IAM Access Analyzer 创建新的分析器，以便它开始监控企业中的资源。

如果委托管理员离开 Amazon 组织，则将从账户中删除委托管理权限。账户中所有将组织作为信任区域的分析器都将变为禁用状态。这些分析器的现有结果也不再可供访问。

首次和管理账户中配置分析器时，可以选择 Add delegated administrator（添加委托管理员）选项，该选项显示在 IAM 控制台中的 IAM Access Analyzer 主页上。

使用控制台添加委派管理员

1. 使用企业的管理账户登录 Amazon 控制台。
2. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
3. 在 Access Analyzer（访问分析器）下，选择 Settings（设置）。
4. 选择 Add delegated administrator（添加委派管理员）。
5. 输入组织成员账户的账号以生成委派管理员。

该账户必须是您组织的成员。

6. 选择保存更改。

使用 Amazon CLI 或 Amazon 开发工具包添加委托管理员

当您使用 Amazon CLI、Amazon API（使用 Amazon 开发工具包）或 Amazon CloudFormation 在委托管理员账户中创建将企业作为信任区域的分析器时，您必须使用 Amazon Organizations API 为 IAM Access Analyzer 启用服务访问，并将成员账户注册为委托管理员。

1. 在 Amazon Organizations 中为 IAM Access Analyzer 启用受信任的服务访问。请参阅《Amazon Organizations 用户指南》中的[如何启用或禁用可信访问](#)。
2. 使用 Amazon Organizations `RegisterDelegatedAdministrator` API 操作或 `register-delegated-administrator` Amazon CLI 命令将 Amazon 企业的有效成员账户注册为委托管理员。

更改委派管理员后，新管理员必须创建分析器才能开始监控对组织中资源的访问。

存档规则

存档规则会自动存档符合您在创建规则时所定义条件的新结果。还可以追溯应用存档规则，以存档符合存档规则条件的现有结果。例如，您可以创建一个存档规则，来自动存档您定期向其授予访问权限的特定 S3 存

存储桶的任何结果。或者，如果您向特定主体授予对多个资源的访问权限，则可以创建一个规则，来自动存档针对向该主体授予的访问权限而生成的任何新结果。这可让您仅关注可能指示安全风险的活动结果。

使用结果详细信息中提供的信息来确定在创建或编辑规则时要使用的特定资源和外部实体。在创建存档规则时，仅自动存档与规则条件匹配的新结果。不会自动存档现有结果。在创建规则时，可以在规则中为每个标识包含最多 20 个值。有关可用于创建或更新存档规则的筛选条件键的列表，请参阅 [IAM Access Analyzer 筛选条件键 \(p. 784\)](#)。

Note

在创建或编辑存档规则时，IAM Access Analyzer 不会验证规则筛选器中包含的值。例如，如果您添加一个规则来匹配 Amazon 账户，则 IAM Access Analyzer 将接受字段中的任何值，即使该值不是有效的 Amazon 账号。

创建存档规则

1. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 选择 Access analyzer (访问分析器)，然后选择 Archive rules (存档规则)。
3. 选择 Create archive rule (创建存档规则)。
4. 如果要更改默认名称，请为规则输入一个名称。
5. 在 Rule (规则) 部分中的 Criteria (条件) 下，为该规则选择要匹配的属性。
6. 选择属性值的运算符，例如 contains。

可用的运算符取决于您选择的属性。

7. (可选) 为属性添加其他值，或为规则添加其他条件。为确保规则不会存档新的结果以供公有访问，还可以包含公有访问条件并将其设置为 false。

要为标准添加另一个值，请选择 Add another value (添加另一个值)。要为规则添加另一个标准，请选择 Add (添加) 按钮。

8. 添加完条件和值后，选择创建规则以仅将规则应用于新结果。选择创建和存档活动的结果，以根据规则条件存档新的和现有的结果。在结果部分中，您可以查看存档规则应用于的活动结果的列表。

例如，要创建自动存档 S3 存储桶的任何结果的规则，请执行以下操作：选择 Resource type (资源类型)，然后为运算符选择 is。接下来，从 Select resource type (选择资源类型) 列表中选择 S3 bucket (S3 存储桶)，然后选择 Add (添加)。

继续定义条件以自定义适用于您环境的规则，然后选择 Create archive rule (创建存档规则)。

如果您创建一个新规则并添加多个标准，则可以通过选择 Remove this criterion (删除此标准) 来从规则中删除单个标准。您可以选择 Remove value (删除值) 来删除为标准添加的值。

编辑存档规则

1. 在 Name (名称) 框中选择要编辑的规则的名称。

您一次只能编辑一个存档规则。

2. 为每个标准添加新条件和值或删除现有条件和值。为确保规则不会存档新的结果以供公有访问，还可以包含公有访问条件并将其设置为 false。
3. 选择保存更改以将规则仅应用于新结果。选择保存和存档活动的结果，以根据规则条件存档新的和现有的结果。

删除存档规则

1. 选中要删除的规则对应的复选框。

您可以同时删除一个、多个或所有规则。

2. 选择 Delete (删除)。
3. 在 Delete archive rule (删除存档规则) 确认对话框中键入 **delete**，然后选择 Delete (删除)。

这仅会从当前区域的分析器中删除规则。您必须为在其他区域中创建的每个分析器单独删除存档规则。

使用 Amazon EventBridge 监控 Amazon Identity and Access Management Access Analyzer

通过本主题中的信息了解如何使用 Amazon EventBridge 监控 IAM Access Analyzer 结果以及如何访问预览。EventBridge 是 Amazon CloudWatch Events 的新版本。

结果事件

IAM Access Analyzer 针对每个生成的结果、对现有结果状态的更改以及删除结果的时间向 EventBridge 发送一个事件。要接收结果以及有关结果的通知，您必须在 Amazon EventBridge 中创建事件规则。在创建事件规则时，您还可以根据规则指定要触发的目标操作。例如，您可以创建一个事件规则，该规则会在从 IAM Access Analyzer 接收新结果的事件时触发 Amazon SNS 主题。

访问预览事件

IAM Access Analyzer 会向 EventBridge 发送事件，其中包含每个访问预览并更改为其状态。这包括首次创建访问预览（状态为“Creating”）、访问预览完成（状态为“Completed”）或访问预览创建失败（状态为“Failed”）时的事件。要接收有关访问预览的通知，您必须在 EventBridge 中创建事件规则。在创建事件规则时，您还可以根据规则指定要触发的目标操作。例如，您可以创建一个事件规则，该规则会在从 IAM Access Analyzer 接收完成的访问预览的事件时触发 Amazon SNS 主题。

事件通知频率

自您的账户中发生事件后约 1 小时内，IAM Access Analyzer 会将新结果的事件以及带状态更新的结果发送到 EventBridge。IAM Access Analyzer 还会在因保留期已过而删除已解析的结果时向 EventBridge 发送事件。对于因生成它们的分析器被删除而被删除的结果，事件将在分析器被删除后大约 24 小时发送到 EventBridge。删除结果时，不会更改其状态。相反，会将 `isDeleted` 属性设置为 `true`。IAM Access Analyzer 还会将新创建的访问预览和访问预览状态更改的事件发送到 EventBridge。

示例结果事件

以下是发送到 EventBridge 的示例 IAM Access Analyzer 事件。列出的 `id` 是 EventBridge 中事件的 ID。要了解更多信息，请参阅 [EventBridge 中的事件和事件模式](#)。

在 `detail` 对象中，`accountId` 和 `region` 属性的值是指结果中报告的账户和区域。`isDeleted` 属性指示事件是否来自要删除的结果。`id` 是结果 ID。`resources` 数组是一个单例，其中包含生成结果的分析器的 ARN。

```
{
  "account": "111122223333",
  "detail": {
    "accountId": "111122223333",
    "action": [
      "s3:GetObject"
    ],
    "analyzedAt": "2019-11-21T01:22:22Z",
    "condition": {},
    "createdAt": "2019-11-20T04:58:50Z",
```

```
{
  "id": "22222222-dcba-4444-dcba-333333333333",
  "isDeleted": false,
  "isPublic": false,
  "principal": {
    "AWS": "999988887777"
  },
  "region": "us-west-2",
  "resource": "arn:aws:s3::my-bucket",
  "resourceType": "AWS::S3::Bucket",
  "status": "ACTIVE",
  "updatedAt": "2019-11-21T01:14:07Z",
  "version": "1.0"
},
"detail-type": "Access Analyzer Finding",
"id": "11111111-2222-4444-aaaa-333333333333",
"region": "us-west-2",
"resources": [
  "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
],
"source": "aws.access-analyzer",
"time": "2019-11-21T01:22:33Z",
"version": "0"
}
```

IAM Access Analyzer 还会将错误结果的事件发送到 EventBridge。错误结果是在 IAM Access Analyzer 无法分析的资源时生成的结果。错误结果的事件包括一个 `error` 属性，如以下示例所示。

```
{
  "account": "111122223333",
  "detail": {
    "accountId": "111122223333",
    "analyzedAt": "2019-11-21T01:22:22Z",
    "createdAt": "2019-11-20T04:58:50Z",
    "error": "ACCESS_DENIED",
    "id": "22222222-dcba-4444-dcba-333333333333",
    "isDeleted": false,
    "region": "us-west-2",
    "resource": "arn:aws:s3::my-bucket",
    "resourceType": "AWS::S3::Bucket",
    "status": "ACTIVE",
    "updatedAt": "2019-11-21T01:14:07Z",
    "version": "1.0"
  },
  "detail-type": "Access Analyzer Finding",
  "id": "11111111-2222-4444-aaaa-333333333333",
  "region": "us-west-2",
  "resources": [
    "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
  ],
  "source": "aws.access-analyzer",
  "time": "2019-11-21T01:22:33Z",
  "version": "0"
}
```

访问预览事件示例

以下示例显示了在您创建访问预览时发送到 EventBridge 的第一个事件的数据。`resources` 数组是一个单例，其中包含了访问预览所关联的分析器的 ARN。在 `detail` 对象中，`id` 是指访问预览 ID，`configuredResources` 是指为其创建访问预览的资源。`status` 为 `Creating`，指访问预览状态。`previousStatus` 未指定，因为访问预览刚刚创建。

```
{
```

```
{
  "account": "111122223333",
  "detail": {
    "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
    "configuredResources": [
      "arn:aws:s3:::example-bucket"
    ],
    "createdAt": "2020-02-20T00:00:00.00Z",
    "region": "us-west-2",
    "status": "CREATING",
    "version": "1.0"
  },
  "detail-type": "Access Preview State Change",
  "id": "aaaabbbb-2222-3333-4444-555566667777",
  "region": "us-west-2",
  "resources": [
    "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
  ],
  "source": "aws.access-analyzer",
  "time": "2020-02-20T00:00:00.00Z",
  "version": "0"
}
```

以下示例显示了发送到 EventBridge 以进行访问预览的事件数据，其状态从 `Creating` 更改为 `Completed`。在详细信息对象中，`id` 是指访问预览 ID。`status` 和 `previousStatus` 是指访问预览状态，其中之前的状态为 `Creating`，当前状态为 `Completed`。

```
{
  "account": "111122223333",
  "detail": {
    "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
    "configuredResources": [
      "arn:aws:s3:::example-bucket"
    ],
    "createdAt": "2020-02-20T00:00:00.000Z",
    "previousStatus": "CREATING",
    "region": "us-west-2",
    "status": "COMPLETED",
    "version": "1.0"
  },
  "detail-type": "Access Preview State Change",
  "id": "11112222-3333-4444-5555-666677778888",
  "region": "us-west-2",
  "resources": [
    "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
  ],
  "source": "aws.access-analyzer",
  "time": "2020-02-20T00:00:00.00Z",
  "version": "0"
}
```

以下示例显示了发送到 EventBridge 以进行访问预览的事件数据，其状态从 `Creating` 更改为 `Failed`。在 `detail` 对象中，`id` 是指访问预览 ID。`status` 和 `previousStatus` 是指访问预览状态，其中之前的状态为 `Creating`，当前状态为 `Failed`。`statusReason` 字段提供了指示访问预览由于资源配置无效而失败的原因代码。

```
{
  "account": "111122223333",
  "detail": {
    "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
    "configuredResources": [
      "arn:aws:s3:::example-bucket"
    ],
    "createdAt": "2020-02-20T00:00:00.00Z",
```

```
{
  "previousStatus": "CREATING",
  "region": "us-west-2",
  "status": "FAILED",
  "statusReason": {
    "code": "INVALID_CONFIGURATION"
  },
  "version": "1.0"
},
{
  "detail-type": "Access Preview State Change",
  "id": "99998888-7777-6666-5555-444433332222",
  "region": "us-west-2",
  "resources": [
    "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
  ],
  "source": "aws.access-analyzer",
  "time": "2020-02-20T00:00:00.00Z",
  "version": "0"
}
```

使用控制台创建事件规则

以下过程介绍了如何使用控制台创建事件规则。

1. 访问 <https://console.aws.amazon.com/events/>，打开 Amazon EventBridge 控制台。
2. 使用以下值创建监控查找事件或访问预览事件的 EventBridge 规则：
 - 对于 Rule type (规则类型)，选择 Rule with an event pattern (具有事件模式的规则)。
 - 对于 Event source (事件源)，选择 Other (其他)。
 - 对于 Event pattern (事件模式)，选择 Custom patterns (JSON editor) [自定义模式 (JSON 编辑器)]，并将以下事件模式之一粘贴到文本区域：
 - 要基于发现事件创建规则，请使用以下模式示例：

```
{
  "source": [
    "aws.access-analyzer"
  ],
  "detail-type": [
    "Access Analyzer Finding"
  ]
}
```

- 要基于访问预览事件创建规则，请使用以下模式示例：

```
{
  "source": [
    "aws.access-analyzer"
  ],
  "detail-type": [
    "Access Preview State Change"
  ]
}
```

- 对于 Target types (目标类型)，选择 Amazon service (Amazon 服务)；对于 Select a target (选择一个目标)，选择一个目标，如 Amazon SNS 主题或 Amazon Lambda 函数。在收到与规则中定义的事件模式匹配的事件时将触发目标。

要详细了解关于创建规则的信息，请参阅《Amazon EventBridge 用户指南》中的[创建对事件作出反应的 Amazon EventBridge 规则](#)。

使用 CLI 创建事件规则

1. 通过以下命令为使用 Amazon CLI 的 Amazon EventBridge 创建规则。将规则名称 **TestRule** 替换为您的规则名称。

```
aws events put-rule --name TestRule --event-pattern '{"source":["aws.access-analyzer\n"]}'
```

2. 您可以自定义规则以便仅针对一小部分生成的结果（例如，具有特定属性的结果）触发目标操作。以下示例演示了如何创建仅针对状态为“活动”的结果触发目标操作的规则。

```
aws events put-rule --name TestRule --event-pattern '{"source":["aws.access-analyzer\n"],"detail-type":["Access Analyzer Finding"],"detail":{"status":["ACTIVE\n"]}}'
```

以下示例演示了如何创建仅针对状态从 Creating 到 Completed 的访问预览触发目标操作的规则。

```
aws events put-rule --name TestRule --event-pattern '{"source":["aws.access-analyzer\n"],"detail-type":["Access Preview State Change"],"detail":{"status":["COMPLETED\n"]}}'
```

3. 要将 Lambda 函数定义为您所创建规则的目标，请使用以下示例命令。根据您的环境，替换 ARN 中的区域和函数名称。

```
aws events put-targets --rule TestRule --targets Id=1,Arn=arn:aws:lambda:us-east-1:111122223333:function:MyFunction
```

4. 添加调用规则目标所需的权限。以下示例演示了如何按照前面的示例向 Lambda 函数授予权限。

```
aws lambda add-permission --function-name MyFunction --statement-id 1 --action 'lambda:InvokeFunction' --principal events.amazonaws.com
```

与 Amazon Security Hub 集成

[Amazon Security Hub](#) 提供了您在 Amazon 中的安全状态的全面视图，可帮助您检查环境是否符合安全行业标准和最佳实践。Security Hub 跨各 Amazon 账户、服务和受支持的第三方合作伙伴产品收集安全数据，帮助您分析安全趋势并确定最高优先级的安全问题。

IAM Access Analyzer 与 Security Hub 的集成使您能够将结果从 Access Analyzer 发送到 Security Hub。Security Hub 可以在其对您的安保状况分析中包含这些结果。

目录

- [Access Analyzer 如何向 Security Hub 发送结果 \(p. 780\)](#)
 - [Access Analyzer 发送的结果类型 \(p. 780\)](#)
 - [发送结果的延迟 \(p. 780\)](#)
 - [Security Hub 不可用时重试 \(p. 780\)](#)
 - [更新 Security Hub 中的现有结果 \(p. 780\)](#)
- [查看 Security Hub 中 Access Analyzer 的结果 \(p. 780\)](#)
 - [解释 Security Hub 中的 Access Analyzer 结果名称 \(p. 781\)](#)
- [来自 Access Analyzer 典型结果 \(p. 781\)](#)
- [启用和配置集成 \(p. 782\)](#)
- [如何停止发送结果 \(p. 782\)](#)

Access Analyzer 如何向 Security Hub 发送结果

在 Security Hub 中，安全问题按结果进行跟踪。一些发现结果来自其他 Amazon 服务或第三方合作伙伴检测到的问题。Security Hub 还有一套用于检测安全问题和生成结果的规则。

Security Hub 提供了管理来自所有这些来源的结果的工具。您可以查看和筛选结果列表，并查看结果的详细信息。请参阅 Amazon Security Hub 用户指南中的[查看结果](#)。您还可以跟踪结果的调查状态。请参阅 Amazon Security Hub 用户指南中的[对结果采取措施](#)。

Security Hub 中的所有结果都使用标准的 JSON 格式，称为 Amazon Security Finding 格式 (ASFF)。ASFF 包括有关问题根源、受影响资源以及结果当前状态的详细信息。请参阅 Amazon Security Hub 用户指南中的[Amazon Security Finding 格式 \(ASFF\)](#)。

IAM Access Analyzer 是一项 Amazon 服务，会将结果发送到 Security Hub。Access Analyzer 在检测到允许外部主体访问您的企业或账户中[受支持资源 \(p. 770\)](#)的策略语句时生成一个结果。Access Analyzer 会对资源的所有结果进行分组，并将单个结果发送到 Security Hub。

Access Analyzer 发送的结果类型

Access Analyzer 使用 [Amazon Security Finding 格式 \(ASFF\)](#) 将结果发送到 Security Hub。在 ASFF 中，Types 字段提供结果类型。来自 Access Analyzer 的结果可能具有 Types 的以下值。

- 效果/数据泄漏/外部访问已授权
- 软件和配置检查/Amazon 安全最佳实践/外部访问已授权

发送结果的延迟

Access Analyzer 创建新结果时，通常会在 30 分钟内将结果发送到 Security Hub。在某些情况下，很少会通知 Access Analyzer 添加或更新了策略。例如，更改 Amazon S3 账户级块公共访问设置可能需要长达 12 小时。此外，如果 Amazon CloudTrail 日志传输存在传输问题，则策略更改不会触发对结果中报告的资源进行重新扫描。在发生此情况时，Access Analyzer 会在下一个定期扫描期间（24 小时内）分析新策略或更新后的策略。

Security Hub 不可用时重试

如果 Security Hub 不可用，Access Analyzer 会重试定期发送结果。

更新 Security Hub 中的现有结果

将结果发送到 Security Hub 之后，IAM Access Analyzer 会发送反映对查找活动的其他观测结果到 Security Hub 中。这些更新反映在同一结果中。

如果 Security Hub 中的资源至少有一个结果处于活动状态，则 Access Analyzer 中的资源的结果将处于活动状态。如果某个资源在 Access Analyzer 中的所有结果均已存档和解决，则 Security Hub 结果也会存档。

当您在公共和跨账户访问之间更改策略访问权限时，Security Hub 结果将更新。此更新可能包括对结果的类型、标题、描述和严重性的更改。

查看 Security Hub 中 Access Analyzer 的结果

要在 Security Hub 中查看 Access Analyzer 结果，请在汇总页面的 Amazon: IAM Access Analyzer (亚马逊云科技：IAM 访问分析器) 部分中选择 See findings (查看结果)。或者，您可以从导航面板中选择 Findings (结果)。然后，您可以通过选择值为 **IAM Access Analyzer** 的 Product name: (产品名称) 字段来筛选结果，以便仅显示 IAM Access Analyzer 结果。

解释 Security Hub 中的 Access Analyzer 结果名称

IAM Access Analyzer 使用 Amazon Security Finding 格式 (ASFF) 将结果发送到 Security Hub。在 ASFF 中，Types (类型) 字段提供结果类型。ASFF 使用与 IAM Access Analyzer 不同的命名方案。下表包含与 Security Hub 中显示的 IAM Access Analyzer 结果关联的所有 ASFF 类型的详细信息。

ASFF 结果类型	Security Hub 结果标题	描述
效果/数据泄漏/外部访问已授权	<resource ARN> 允许公开访问	附加到资源的基于资源的策略允许所有外部主体对资源的公开访问。
软件和配置检查/Amazon 安全最佳实践/外部访问已授权	<resource ARN> 允许跨账户访问	附加到资源的基于资源的策略允许跨账户访问分析器信任区之外的外部主体。

来自 Access Analyzer 典型结果

Access Analyzer 使用 [AmazonSecurity Finding 格式 \(ASFF\)](#) 将结果发送到 Security Hub。

下面是 Access Analyzer 典型结果的示例。

```
{
  "SchemaVersion": "2018-10-08",
  "Id": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/my-analyzer/",
  "arn:aws:s3::my-bucket",
  "ProductArn": "arn:aws:securityhub:us-west-2::product/aws/access-analyzer",
  "GeneratorId": "aws/access-analyzer",
  "AwsAccountId": "111122223333",
  "Types": ["Software and Configuration Checks/Amazon Security Best Practices/External Access Granted"],
  "CreatedAt": "2020-11-10T16:17:47Z",
  "UpdatedAt": "2020-11-10T16:43:49Z",
  "Severity": {
    "Product": 1,
    "Label": "LOW",
    "Normalized": 1
  },
  "Title": "AwsS3Bucket/arn:aws:s3::my-bucket/ allows cross-account access",
  "Description": "AWS::S3::Bucket/arn:aws:s3::my-bucket/ allows cross-account access from Amazon 444455556666",
  "Remediation": {
    "Recommendation": {"Text": "If the access isn't intended, it indicates a potential security risk. Use the console for the resource to modify or remove the policy that grants the unintended access. You can use the Rescan button on the Finding details page in the Access Analyzer console to confirm whether the change removed the access. If the access is removed, the status changes to Resolved."}
  },
  "SourceUrl": "https://console.aws.amazon.com/access-analyzer/home?region=us-west-2/findings/details/dad90d5d-63b4-6575-b0fa-ef9c5566ge798",
  "Resources": [
    {
      "Type": "AwsS3Bucket",
      "Id": "arn:aws:s3::my-bucket",
      "Details": {
        "Other": {
          "External Principal Type": "AWS",
          "Condition": "none",
          "Action Granted": "s3:GetObject,s3:GetObjectVersion",
          "External Principal": "444455556666"
        }
      }
    }
  ]
}
```

```
    }  
  },  
  ],  
  "WorkflowState": "NEW",  
  "Workflow": {"Status": "NEW"},  
  "RecordState": "ACTIVE"  
}
```

启用和配置集成

若要使用与 Security Hub 的集成，您必须启用 Security Hub。有关如何启用 Security Hub 的信息，请参阅 Amazon Security Hub 用户指南中的[设置 Security Hub](#)。

当您同时启用 Access Analyzer 和 Security Hub 时，集成将自动启用。Access Analyzer 立即开始将结果发送到 Security Hub。

如何停止发送结果

要停止向 Security Hub 发送结果，您可以使用 Security Hub 控制台或 API。

请参阅 Amazon Security Hub 用户指南中的[禁用和启用来自集成的结果流（控制台）](#)或[禁用来自集成的结果流（Security Hub API、Amazon CLI）](#)。

使用 Amazon CloudTrail 记录 IAM Access Analyzer API 调用

IAM Access Analyzer 已与 Amazon CloudTrail 集成，该服务提供 IAM Access Analyzer 中的用户、角色或 Amazon 服务采取的操作记录。CloudTrail 将 IAM Access Analyzer 的所有 API 调用作为事件捕获。捕获的调用包含来自 IAM Access Analyzer 控制台的调用和对 IAM Access Analyzer API 操作的代码调用。

如果您创建跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 IAM Access Analyzer 的事件）。如果您不配置跟踪，则仍可在 CloudTrail 控制台中的 Event history（事件历史记录）中查看最新事件。

使用 CloudTrail 收集的信息，您可以确定向 IAM Access Analyzer 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅《[Amazon CloudTrail 用户指南](#)》。

CloudTrail 中的 IAM Access Analyzer 信息

在您创建 Amazon 账户时，将在该账户上启用 CloudTrail。当 IAM Access Analyzer 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 Amazon 服务事件一同保存在 Event history（事件历史记录）中。您可以在 Amazon 账户中查看、搜索和下载最新事件。有关更多信息，请参阅[使用 CloudTrail 事件历史记录查看事件](#)。

要持续记录 Amazon 账户中的事件（包括 IAM Access Analyzer 的事件），请创建跟踪。通过跟踪记录，CloudTrail 可将日志文件传送至 Simple Storage Service（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 Amazon 区域。此跟踪记录在 Amazon 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Simple Storage Service（Amazon S3）存储桶。此外，您可以配置其他 Amazon 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅下列内容：

- [创建跟踪概览](#)
- [CloudTrail 支持的服务和集成](#)
- [为 CloudTrail 配置 Amazon SNS 通知](#)
- [从多个区域接收 CloudTrail 日志文件](#)和[从多个账户接收 CloudTrail 日志文件](#)

CloudTrail 录入了所有 IAM Access Analyzer 操作，[IAM Access Analyzer API 引用](#)中记录了这些操作。例如，对 `CreateAnalyzer`、`CreateArchiveRule` 和 `ListFindings` 操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 Amazon Identity and Access Management (IAM) 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其它 Amazon 服务发出。

有关更多信息，请参阅 [CloudTrail userIdentity 元素](#)。

了解 IAM Access Analyzer 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了名为 `Alice-tempcreds` 的假定角色会话在“2021 年 6 月 14 日”执行的 `CreateAnalyzer` 操作。角色会话由名为 `admin-tempcreds` 的角色发出。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAIKBKVSQ6C2EXAMPLE:Alice-tempcreds",
    "arn": "arn:aws:sts::111122223333:assumed-role/admin-tempcreds/Alice-tempcreds",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "true",
        "creationDate": "2021-06-14T22:54:20Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/admin-tempcreds",
        "accountId": "111122223333",
        "userName": "admin-tempcreds"
      },
      "webIdFederationData": {},
    },
  },
  "eventTime": "2021-06-14T22:57:36Z",
  "eventSource": "access-analyzer.amazonaws.com",
  "eventName": "CreateAnalyzer",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "198.51.100.179",
  "userAgent": "aws-sdk-java/1.12.79 Linux/5.4.141-78.230 OpenJDK_64-Bit_Server_VM/25.302-b08 java/1.8.0_302 vendor/Oracle_Corporation cfg/retry-mode/standard",
  "requestParameters": {
    "analyzerName": "test",
    "type": "ACCOUNT",
    "clientToken": "11111111-abcd-2222-abcd-222222222222",
    "tags": {
      "tagkey1": "tagvalue1"
    }
  },
  "responseElements": {
    "arn": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/test"
  }
}
```

```

},
"requestID": "22222222-dcba-4444-dcba-333333333333",
"eventID": "33333333-bcde-5555-bcde-444444444444",
"readOnly": false,
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333",
"eventCategory": "Management"
}

```

IAM Access Analyzer 筛选条件键

您可以使用下面的筛选键定义存档规则 [CreateArchiveRule](#)、更新存档规则 [UpdateArchiveRule](#)、检索结果列表 [ListFindings](#)，或检索资源 [ListAccessPreviewFindings](#) 的访问预览结果列表。使用 IAM API 或是 Amazon CloudFormation 来配置归档规则的操作没有区别。

Criterion	描述	Type	存档规则	列出结果	列出访问预览结果
资源	ARN 唯一标识外部主体有权访问的资源。要了解更多信息，请参阅 Amazon 资源名称 (ARN) 。	字符串	✔ 是	✔ 是	✔ 是
resourceType	外部主体有权访问的资源类型。 AWS::IAM::Role AWS::KMS::Key AWS::Lambda::Function AWS::Lambda::LayerVersion AWS::S3::Bucket AWS::SQS::Queue AWS::SecretsManager::Secret	字符串	✔ 是	✔ 是	✔ 是
resourceOwnerAccountId	拥有资源的 12 位数 Amazon 账户 ID。要了解更多信息，请参阅 Amazon 账户标识符 。	字符串	✔ 是	✔ 是	✔ 是
isPublic	指示结果是否报告具有允许公共访问的策略的资源。	布尔值	✔ 是	✔ 是	✔ 是
状态	结果的当前状态。 ACTIVE ARCHIVED RESOLVED	字符串	✘ 否	✔ 是	✔ 是
error	指示为结果报告的错误。	字符串	✔ 是	✔ 是	✔ 是
principal.AWS	在结果的 Principal 字段中授予对资源访问权限的账	字符串	✔ 是	✔ 是	✔ 是

Criterion	描述	Type	存档规则	列出结果	列出访问预览结果
	户的 ARN。要了解更多信息，请参阅 Amazon 账户标识符 。				
principal.Federated	有权访问结果中资源的联合身份的 ARN。要了解更多信息，请参阅 身份提供商和联合身份验证 。	字符串	✔ 是	✔ 是	✔ 是
condition.aws:PrincipalArn	指定作为资源访问条件的主体（IAM 用户、角色或组）的 ARN。要了解更多信息，请参阅 Amazon 全局条件上下文键 。	字符串	✔ 是	✔ 是	✔ 是
condition.aws:PrincipalOrgID	指定作为资源访问条件的主体的组织标识符。要了解更多信息，请参阅 Amazon 全局条件上下文键 。	字符串	✔ 是	✔ 是	✔ 是
condition.aws:PrincipalOrgUnit	指定作为资源访问条件的组织或组织单元 (OU) 的 ID。要了解更多信息，请参阅 Amazon 全局条件上下文键 。	字符串	✔ 是	✔ 是	✔ 是
condition.aws:SourceIP	允许主体在使用指定 IP 地址时访问资源的 IP 地址。要了解更多信息，请参阅 Amazon 全局条件上下文键 。	IP 地址	✔ 是	✔ 是	✔ 是
condition.aws:SourceVpc	允许主体在使用指定 VPC 时访问资源的 VPC ID。要了解更多信息，请参阅 Amazon 全局条件上下文键 。	字符串	✔ 是	✔ 是	✔ 是
condition.aws:UserSub	外部账户中 IAM 用户的用户 ID，该用户指定作为访问资源的条件。要了解更多信息，请参阅 Amazon 全局条件上下文键 。	字符串	✔ 是	✔ 是	✔ 是
condition.cognito-identity.amazonaws.com:PrincipalType	Amazon Cognito 身份池 ID，指定作为结果中 IAM 角色访问的条件。要了解更多信息，请参阅 IAM 和 Amazon STS 条件上下文键 。	字符串	✔ 是	✔ 是	✔ 是

Criterion	描述	Type	存档规则	列出结果	列出访问预览结果
condition.graphiclogin	将 Facebook 应用程序 ID (或网站 ID) 指定作为条件, 用于允许对结果中 IAM 角色的“用 Facebook 登录”联合身份访问。要了解更多信息, 请参阅 IAM 和 Amazon STS 条件上下文键 。	字符串	✔ 是	✔ 是	✔ 是
condition.account.google	指定作为访问 IAM 角色的条件的 Google 应用程序 ID。要了解更多信息, 请参阅 IAM 和 Amazon STS 条件上下文键 。	字符串	✔ 是	✔ 是	✔ 是
condition.kms:ChannelArn	拥有调用实体的 Amazon 账户 ID (IAM 用户、角色或根用户), 调用 Amazon KMS 的服务使用该账户。要了解更多信息, 请参阅 Amazon Key Management Service 的条件键 。	字符串	✔ 是	✔ 是	✔ 是
condition.www.amazon	将 Amazon 应用程序 ID (或网站 ID) 指定作为条件, 用于允许对角色的“Login with Amazon”联合身份访问。要了解更多信息, 请参阅	字符串	✔ 是	✔ 是	✔ 是
id	结果的 ID。	字符串	✘ 否	✔ 是	✔ 是
changeType	提供有关访问预览查找结果如何与 IAM Access Analyzer 中标识的现有访问进行比较的上下文。	字符串	✘ 否	✘ 否	✔ 是
existingFindingId	IAM Access Analyzer 中查找结果的现有 ID, 仅为访问预览中的现有查找结果提供。	字符串	✘ 否	✘ 否	✔ 是
existingFindingStatus	查找结果的现有状态, 仅为访问预览中的现有调查结果提供。	字符串	✘ 否	✘ 否	✔ 是

将服务相关角色用于 Amazon Identity and Access Management Access Analyzer

Amazon Identity and Access Management Access Analyzer 使用 IAM [服务相关角色](#)。服务相关角色是一种与 Access Analyzer 直接关联的独特类型的 IAM 角色。服务相关角色由 Access Analyzer 预定义, 并包含该服务代表您调用其他 Amazon 服务所需的一切权限。

服务相关角色可让您更轻松地了解设置 Access Analyzer，因为您不必手动添加必要的权限。Access Analyzer 定义其服务相关角色的权限，除非另外定义，否则只有 Access Analyzer 可以代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其它 IAM 实体的权限策略。

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 Amazon 服务](#)并查找 Service-Linked Role（服务相关角色）列中显示为 Yes（是）的服务。请选择 Yes 与查看该服务的服务相关角色文档的链接。

Amazon Identity and Access Management Access Analyzer 的服务相关角色权限

Amazon Identity and Access Management Access Analyzer 使用名为 AWSServiceRoleForAccessAnalyzer 的服务相关角色，允许 Access Analyzer 分析资源元数据。

AWSServiceRoleForAccessAnalyzer 服务相关角色信任以下服务来代入该角色：

- `access-analyzer.amazonaws.com`

名为 [AccessAnalyzerServiceRolePolicy \(p. 760\)](#) 的角色权限策略允许 Access Analyzer 完成对特定资源的操作。

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](#)。

为 Access Analyzer 创建服务关联角色

无需手动创建服务相关角色。当您在 Amazon Web Services Management Console 或 Amazon API 中启用 Access Analyzer 时，Access Analyzer 将为您创建服务关联角色。在您启用 IAM Access Analyzer 的所有区域中使用相同的角色。

Note

IAM Access Analyzer 是区域性的。您必须在每个区域内单独启用 IAM Access Analyzer。

如果您删除此服务相关角色，则在您下次创建分析器时，IAM Access Analyzer 会重新创建该角色。

您也可以使用 IAM 控制台为 Access Analyzer 使用案例创建服务相关角色。在 Amazon CLI 或 Amazon API 中，使用 `access-analyzer.amazonaws.com` 服务名称创建服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的[创建服务相关角色](#)。如果您删除了此服务相关角色，则可以使用此相同过程再次创建角色。

编辑 Access Analyzer 的服务相关角色

Access Analyzer 不允许您编辑 AWSServiceRoleForAWSLicenseManagerMasterAccountRole 服务相关角色。创建服务相关角色后，将无法更改角色名称，因为可能有多个实体引用该角色。但是可以使用 IAM 编辑角色说明。有关更多信息，请参见 IAM 用户指南中的[编辑服务相关角色](#)。

删除 Access Analyzer 的服务关联角色

如果您不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。但是，您必须先清除服务相关角色的资源，然后才能手动删除它。

Note

在您尝试删除资源时，如果 Access Analyzer 正在使用该角色，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。

要删除 `AWSServiceRoleForAccessAnalyzer` 服务相关角色所使用的 Access Analyzer 资源

1. 通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在 Access reports (访问报告) 部分中的 Access analyzer (访问分析器) 下，选择 Analyzers (分析器)。
3. 选中 Analyzer (分析器) 表中分析器列表左上方的复选框以选择所有分析器。
4. 选择 Delete (删除)。
5. 要确认您要删除分析器，请输入 `delete`，然后选择 Delete (删除)。

使用 IAM 手动删除服务相关角色

使用 IAM 控制台、Amazon CLI 或 Amazon API 删除 `AWSServiceRoleForAccessAnalyzer` 服务相关角色。有关更多信息，请参见 IAM 用户指南中的[删除服务相关角色](#)。

Access Analyzer 服务相关角色的受支持区域

Access Analyzer 支持在服务可用的所有区域中使用服务相关角色。有关更多信息，请参阅[Amazon 区域和终端节点](#)。

预览访问

除了帮助您识别与外部实体共享的资源外，Amazon IAM Access Analyzer 还允许您在部署资源权限之前预览 IAM Access Analyzer 的结果，以便您可以验证策略更改是否仅授予对资源的预期公共和跨账户访问权限。这可以帮助您从预期的外部访问资源开始操作。

您可以预览和验证对 [Amazon S3 控制台](#) 中 Amazon S3 存储桶的公共和跨账户访问权限。您还可以通过为您的资源提供建议的权限，以使用 IAM Access Analyzer API 预览 Amazon S3 存储桶、Amazon KMS 密钥、IAM 角色、Amazon SQS 队列和 Secrets Manager 密钥的公共和跨账户访问权限。

主题

- [在 Amazon S3 控制台中预览访问权限 \(p. 788\)](#)
- [使用 IAM Access Analyzer API 预览访问 \(p. 789\)](#)

在 Amazon S3 控制台中预览访问权限

在 Amazon S3 控制台中完成存储桶策略后，您可以选择预览 Amazon S3 存储桶的公共和跨账户访问权限。您可以验证策略更改是否仅授予预期的外部访问权限，然后再选择 Save changes (保存更改)。利用此可选步骤，您可以预览存储桶的 Amazon Identity and Access Management Access Analyzer 结果。您可以验证策略更改是否引入了新的结果，或是解析现有的结果以供外部访问。您可以跳过此验证步骤，并随时保存您的 Amazon S3 存储桶策略。

要预览对存储桶的外部访问权限，您必须在存储桶的区域中拥有一个活动账户分析器，并且该账户需作为信任区域。您还必须具有使用 IAM Access Analyzer 和预览访问所需的权限。有关启用 IAM Access Analyzer 以及所需权限的更多信息，请参阅[启用 IAM Access Analyzer \(p. 765\)](#)。

要在创建或编辑存储桶策略时预览 Amazon S3 存储桶的访问

1. 完成创建或编辑存储桶策略后，请确保您的策略是有效的 Amazon S3 存储桶策略。策略 ARN 必须与存储桶 ARN 相匹配，且策略元素必须有效。
2. 在政策下方的 Preview external access (预览外部访问) 项下，选择活动的账户分析器，然后选择 Preview (预览)。系统将为您的存储桶生成 IAM Access Analyzer 结果的预览。预览会分析显示的

Amazon S3 存储桶策略以及现有存储桶权限。这包括存储桶和账户 BPA 设置、存储桶 ACL、Amazon S3 访问点和附加到存储桶的多区域访问点，以及它们的策略和 BPA 设置。

3. 访问预览完成后，将显示 IAM Access Analyzer 结果的预览。在您保存策略后，每个结果都会报告账户外的主体实例，该实例可访问您的存储桶。您可以通过查看每个结果来验证对存储桶的访问权限。结果标题提供访问的摘要，您可以展开结果以查看[结果详细信息](#)。结果标记可提供有关保存存储桶策略如何更改对存储桶的访问权限的上下文。例如，它们可帮助您验证策略更改是否引入了新的结果，或是解决了外部访问的现有结果：
 - a. 新结果 – 表示策略将引入的新外部访问结果。
 - b. 已解决 – 表示策略将删除的现有外部访问结果。
 - c. 已归档 — 表示将根据分析器的归档规则自动归档的新外部访问结果，这些规则定义了何时应将结果标记为预期结果。
 - d. 现有 – 表示将保持不变的外部访问的现有结果。
 - e. 公开 — 如果一个结果是用于公共访问资源，它将带有一个 Public (公开) 徽章，此外还将带有上述徽章之一。
4. 如果您识别不打算引入或删除的外部访问，则可以修改策略，然后选择 Preview (预览)，直到已经达到您想要的外部访问。如果你有一个标记标注为 Public (公开) 的结果，我们建议您修改策略以删除公共访问权限，然后再选择 Save changes (保存更改)。预览访问是一个可选步骤，您随时都可以选择 Save changes (保存更改)。

使用 IAM Access Analyzer API 预览访问

您可以使用 [IAM Access Analyzer API](#) 预览 Amazon S3 存储桶的公共和跨账户访问、Amazon KMS 密钥、IAM 角色、Amazon SQS 队列和 Secrets Manager 密钥。您可以通过为您拥有的现有资源或您想要部署的新资源提供建议的权限来预览访问。

要预览对资源的外部访问，您必须具有适用于资源账户和区域的活动账户分析器。您还必须具有使用 IAM Access Analyzer 和预览访问所需的权限。有关启用 IAM Access Analyzer 以及所需权限的更多信息，请参阅 [启用 IAM Access Analyzer \(p. 765\)](#)。

要预览资源的访问，可以使用 CreateAccessPreview 操作，并提供分析器 ARN 和资源的访问控制配置。该服务返回访问预览的唯一 ID，您可以使用该 ID 通过 GetAccessPreview 操作查看访问预览的状态。当状态为 Completed，则可以使用 ListAccessPreviewFindings 操作来检索为访问预览生成的结果。GetAccessPreview 和 ListAccessPreviewFindings 操作将检索大约 24 小时内创建的访问预览和结果。

检索的每个结果都包含描述访问的[结果详细信息](#)。结果的预览状态，描述权限部署之后结果是否为 Active、Archived 或者 Resolved，以及 changeType。changeType 提供了有关访问预览结果如何与 IAM Access Analyzer 中标识的现有访问进行比较的上下文：

- 新结果 – 结果适用于新引入的访问。
- 不变 – 预览结果是将保持不变的现有结果。
- 已更改 – 预览结果是状态发生变化的现有查找结果。

status 和 changeType 帮助您了解资源配置将如何更改现有资源访问。如果 changeType 为 Unchanged 或“已更改”，则结果还将包含 IAM Access Analyzer 中结果的现有 ID 和状态。例如，Changed 结果具有预览状态 Resolved 和现有状态 Active，指示资源的现有 Active 结果将由于建议权限更改而成为 Resolved。

您可以使用 ListAccessPreviews 操作来检索指定分析器的访问预览列表。此操作将检索约一小时内创建的访问预览信息。

通常，如果访问预览适用于现有资源，并且未指定配置选项，则预设情况下，访问预览将使用现有资源配置。如果访问预览适用于新资源，并且未指定配置选项，则访问预览将使用默认值，具体取决于资源类型。有关每个资源类型的配置案例，请参阅下文。

预览对 Amazon S3 存储桶的访问

要为新 Amazon S3 存储桶或您拥有的现有 Amazon S3 存储桶创建访问预览，您可以通过指定附加到存储桶的 Amazon S3 存储桶策略、存储桶 ACL、存储桶 BPA 设置和 Amazon S3 访问点（包括多区域访问点）来建议存储桶配置。

Note

在尝试为新存储桶创建访问预览之前，我们建议您调用 Amazon S3 [HeadBucket](#) 操作来检查指定的存储桶是否存在。此操作可用于确定存储桶是否存在以及您是否有权访问该存储桶。

存储桶策略 — 如果配置适用于现有 Amazon S3 存储桶，并且您未指定 Amazon S3 存储桶策略，则访问预览将使用附加到存储桶的现有策略。如果访问预览适用于新资源，并且您没有指定 Amazon S3 存储桶策略，则访问预览会假定存储桶没有策略。要建议删除现有存储桶策略，您可以指定一个空字符串。有关支持的存储桶策略限制的更多信息，请参阅[存储桶策略示例](#)。

存储桶 ACL 授权 — 每个存储桶最多可提议 100 个 ACL 授权。如果建议的授权配置适用于现有存储桶，则访问预览会使用建议的授权配置列表来代替现有授权。否则，访问预览将使用存储桶的现有授权。

存储桶访问点 — 分析支持每个存储桶多达 100 个访问点，包括多区域访问点，包括每个存储桶最多可建议的 10 个新访问点。如果建议的 Amazon S3 访问点配置适用于现有存储桶，则访问预览会使用建议的访问点配置代替现有访问点。要建议没有策略的访问点，可以提供空字符串作为访问点策略。有关访问点策略限制的更多信息，请参阅[访问点限制和局限性](#)。

阻止公有访问配置 — 如果建议的配置适用于现有 Amazon S3 存储桶，并且您未指定配置，则访问预览将使用现有设置。如果建议的配置适用于新存储桶，并且您未指定存储桶 BPA 配置，则访问预览会使用 `false`。如果建议的配置适用于新的访问点或多区域访问点，并且您未指定访问点 BPA 配置，则访问预览会使用 `true`。

预览对您的 Amazon KMS 密钥的访问

要新的 Amazon KMS 密钥或您所拥有的现有 Amazon KMS 密钥创建访问预览，可以通过指定免邮策略和 Amazon KMS 授权配置来建议 Amazon KMS 密钥。

Amazon KMS 密钥策略 — 如果配置适用于现有密钥，并且您未指定密钥策略，则访问预览将使用该密钥的现有策略。如果访问预览适用于新资源，并且您未指定密钥策略，则访问预览将使用默认密钥策略。建议的密钥策略不能是空字符串。

Amazon KMS 授权 — 分析最多支持每个配置 100 个 KMS 授权 *。* 如果建议的授权配置适用于现有密钥，则访问预览会在现有授权中使用建议的授权配置列表。否则，访问预览将使用该密钥的现有授权。

预览对 IAM 角色的访问

要为新 IAM 角色或您拥有的现有 IAM 角色创建访问预览，您可以通过指定信任策略来建议 IAM 角色配置。

角色信任策略 — 如果配置适用于新 IAM 角色，则必须指定信任策略。如果配置适用于您拥有的现有 IAM 角色，并且您没有建议信任策略，则访问预览将使用角色的现有信任策略。建议的信任策略不能是空字符串。

预览对您的 Amazon SQS 队列的访问

要为新 Amazon SQS 队列或您拥有的现有 Amazon SQS 队列创建访问预览，您可以通过为队列指定 Amazon SQS 策略来建议 Amazon SQS 队列配置。

Amazon SQS 队列策略 — 如果配置适用于现有 Amazon SQS 队列，并且您未指定 Amazon SQS 策略，则访问预览将为队列使用现有的 Amazon SQS 策略。如果访问预览适用于新资源，并且您没有指定策略，则访问预览会假定 Amazon SQS 队列没有策略。要建议删除现有 Amazon SQS 队列策略，您可以为 Amazon SQS 策略指定一个空字符串。

预览对您的 Secrets Manager 密钥的访问

要为新的 Secrets Manager 密钥或您拥有的现有 Secrets Manager 密钥创建访问预览，您可以通过指定密钥策略和可选的 Amazon KMS 加密密钥。

密钥策略 — 如果配置适用于现有密钥，并且您未指定密钥策略，则访问预览将使用现有的密钥策略。如果访问预览适用于新资源，并且您未指定策略，则访问预览会假定密钥没有策略。要建议删除现有策略，可以指定一个空字符串。

Amazon KMS 加密密钥 — 如果建议的配置适用于新密钥，并且您没有指定 Amazon KMS 密钥 ID，则访问预览将使用 Amazon 账户的默认 KMS 密钥。如果您为 Amazon KMS 密钥 ID 指定空字符串，则访问预览将使用 Amazon 账户的默认 KMS 密钥。

IAM Access Analyzer 策略验证

您可以使用 Amazon Identity and Access Management Access Analyzer 策略检查验证策略。您可以在 IAM 控制台或使用 Amazon CLI、Amazon API 或 JSON 策略编辑器创建或编辑策略。IAM Access Analyzer 将根据 IAM [policy 语法 \(p. 985\)](#) 和 [最佳实践 \(p. 752\)](#) 验证您的策略。您可以查看策略验证检查结果，其中包括策略的安全警告、错误、常规警告和策略建议。这些结果提供了可操作的建议，可帮助您编写可操作且符合安全最佳实践的策略。若要查看 Access Analyzer 返回的警告、错误和建议列表，请参阅 [Access Analyzer 策略检查参考 \(p. 792\)](#)。

在 IAM (控制台) 中验证策略

在 IAM 控制台中创建或编辑托管策略时，您可以查看策略检查生成的结果。您还可以查看这些内联用户或角色策略的结果。Access Analyzer 不会为内联组策略生成这些结果。

查看 IAM JSON 策略的策略检查生成的结果

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 使用以下方法之一开始创建或编辑策略：
 - a. 要创建新的托管策略，请转到 Policies (策略) 页面并创建新策略。有关更多信息，请参阅 [在“JSON”选项卡上创建策略 \(p. 438\)](#)。
 - b. 要查看现有托管策略的策略检查，请转到 Policies (策略) 页面，选择策略的名称，然后选择 Edit policy (编辑策略)。有关更多信息，请参阅 [编辑客户托管策略 \(控制台 \) \(p. 461\)](#)。
 - c. 要查看针对用户或角色的内联策略的策略检查，请转到 Users (用户) 或者 Roles (角色) 页面，选择用户或角色的名称，然后选择 Permissions (权限) 选项卡上的 Edit policy (编辑策略)。有关更多信息，请参阅 [编辑客户托管策略 \(控制台 \) \(p. 461\)](#)。
3. 在策略编辑器中，选择 JSON 选项卡。
4. 在策略下方的策略验证窗格中，选择以下一个或多个选项卡。选项卡名称还指示策略的每种查找类型的数量。
 - Security (安全) — 如果您的策略允许访问，但由于访问权限过于宽松而致使 Amazon 认为存在安全风险，则可查看相关警告。
 - Errors (错误) — 如果策略包含阻止策略运行的行，则可查看错误。
 - General (常规) — 如果您的策略不符合最佳实践，但问题不属于安全风险，则可查看警告。
 - Suggestions (建议) — 如果 Amazon 的建议不影响策略权限的改进，则可查看建议。
5. 查看 IAM Access Analyzer 策略检查提供的结果详细信息。每个结果都会指示所报告问题的位置。要了解有关导致问题的原因以及如何解决问题的详细信息，请选择结果旁的 Learn more (了解更多) 链接。您还可以在 [Access Analyzer policy checks \(p. 792\)](#) (Access Analyzer 策略检查) 参考页面搜索与各个结果关联的策略检查。

6. 请更新策略以解决结果。

Important

在生产工作流中实施新策略之前，测试新策略或彻底编辑这些策略。

7. 完成后，选择 Review policy (查看策略)。策略验证程序 (p. 443) 会报告 Access Analyzer 未报告的任何语法错误。

Note

您可以随时在 Visual editor (可视化编辑器) 和 JSON 选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy (查看策略)，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅 [调整策略结构 \(p. 871\)](#)。

8. 在查看策略页上，为创建的策略键入名称和说明 (可选)。查看策略 Summary (摘要) 以查看您的策略授予的权限。然后，选择创建策略以保存您的工作。

使用 Access Analyzer 验证策略 (Amazon CLI 或者 Amazon API)

您可以从 Amazon Command Line Interface (Amazon CLI) 查看 IAM Access Analyzer 策略检查生成的结果。

查看 IAM Access Analyzer 策略检查生成的结果 (Amazon CLI 或 Amazon API)

使用以下值之一：

- Amazon CLI: [aws accessanalyzer validate-policy](#)
- Amazon API: [ValidatePolicy](#)

Access Analyzer 策略检查参考

您可以使用 Amazon IAM Access Analyzer 策略检查来验证您的策略。您可以在 IAM 控制台中使用 Amazon CLI、Amazon API 或 JSON 策略编辑器创建或编辑策略。Access Analyzer 将根据 IAM [policy 语法 \(p. 985\)](#) 和 [最佳实践 \(p. 752\)](#) 验证您的策略。您可以查看策略验证检查结果，其中包括策略的安全警告、错误、常规警告和策略建议。这些结果提供了可操作的建议，可帮助您编写可操作且符合安全最佳实践的策略。要了解有关使用 Access Analyzer 验证策略的详细信息，请参阅 [IAM Access Analyzer 策略验证 \(p. 791\)](#)。

错误 — 不允许 ARN 账户

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
ARN account not allowed: The service does not support specifying an account ID in the resource ARN.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The service does not support specifying an account ID in the resource ARN."
```

解决错误

请从资源 ARN 中删除账户 ID。某些 Amazon 服务的资源 ARN 不支持指定账户 ID。

例如，Amazon S3 不支持将账户 ID 作为存储桶 ARN 中的命名空间。Amazon S3 存储桶名称是全局唯一的，并且命名空间由所有 Amazon 账户共享。要查看 Amazon S3 中所有可用的资源类型，请参阅服务授权参考中的 [Amazon S3 定义的资源类型](#)。

相关术语

- [策略资源 \(p. 946\)](#)
- [账户标识符](#)
- [资源 ARN \(p. 896\)](#)
- [采用 ARN 格式的 Amazon 服务资源](#)

错误 — 不允许 ARN 区域

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
ARN Region not allowed: The service does not support specifying a Region in the resource ARN.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The service does not support specifying a Region in the resource ARN."
```

解决错误

请从资源 ARN 中删除区域。某些 Amazon 服务的资源 ARN 不支持指定区域。

例如，IAM 是一项全球性服务。IAM 资源 ARN 的区域部分始终保留为空。IAM 资源是全球性的，如同现在的 Amazon 账户。例如，以 IAM 用户身份登录后，您可以在任何地理区域访问 Amazon 服务。

- [策略资源 \(p. 946\)](#)
- [资源 ARN \(p. 896\)](#)
- [采用 ARN 格式的 Amazon 服务资源](#)

错误 — 数据类型不匹配

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Data type mismatch: The text does not match the expected JSON data type {{data_type}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The text does not match the expected JSON data type {{data_type}}."
```

解决错误

请更新文本以使用支持的数据类型。

例如，Version 全局条件键需要 String 数据类型。如果您提供日期或整数，则数据类型将不匹配。

相关术语

- [全局条件键 \(p. 998\)](#)
- [IAM JSON 策略元素：条件运算符 \(p. 951\)](#)

错误 — 使用不同的大小写重复密钥

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Duplicate keys with different case: The condition key appears more than once with different capitalization in the same condition block. Remove the duplicate condition keys.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key appears more than once with different capitalization in the same condition block. Remove the duplicate condition keys."
```

解决错误

查看同一条件数据块中的类似条件键，并对所有实例使用相同的大写字母。

条件数据块是策略语句 `Condition` 元素中的文本。条件键名称不区分大小写。条件键值是否区分大小写取决于您使用的条件运算符。有关条件键中区分大小写的更多信息，请参阅 [IAM JSON 策略元素：Condition \(p. 948\)](#)。

相关术语

- [条件 \(p. 948\)](#)
- [条件数据块 \(p. 950\)](#)
- [全局条件键 \(p. 998\)](#)
- [Amazon 服务条件键](#)

错误 — 无效的操作

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid action: The action does not exist. Did you mean ?
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The action does not exist. Did you mean ?"
```

解决错误

指定的操作无效。如果您键入了错误的服务前缀或操作名称，则可能会发生这种情况。对于一些常见问题，策略检查将返回建议的操作。

相关术语

- [策略操作 \(p. 944\)](#)
- [Amazon 服务操作](#)

出现此错误的 Amazon 托管策略

[Amazon 托管策略 \(p. 364\)](#)将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

以下 Amazon 托管策略在其策略语句中包含无效操作。无效操作不会影响策略所授予的权限。使用 Amazon 托管策略作为创建托管策略的参考时，Amazon 会建议您从策略中删除无效操作。

- [AmazonEMRFullAccessPolicy_v2](#)
- [CloudWatchSyntheticsFullAccess](#)

错误 — 无效的 ARN 账户

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid ARN account: The resource ARN account ID is not valid. Provide a 12-digit account ID.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The resource ARN account ID is not valid. Provide a 12-digit account ID."
```

解决错误

请更新资源 ARN 中的账户 ID。账户 ID 是 12 位整数。要了解如何查看您的账户 ID，请参阅[查找您的 Amazon 账户 ID](#)。

相关术语

- [策略资源 \(p. 946\)](#)
- [账户标识符](#)
- [资源 ARN \(p. 896\)](#)
- [采用 ARN 格式的 Amazon 服务资源](#)

错误 — 无效的 ARN 前缀

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid ARN prefix: Add the required prefix (arn) to the resource ARN.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add the required prefix (arn) to the resource ARN."
```

解决错误

Amazon 资源 ARN 必须包含所需 `arn:` 前缀。

相关术语

- [策略资源 \(p. 946\)](#)
- [资源 ARN \(p. 896\)](#)
- [采用 ARN 格式的 Amazon 服务资源](#)

错误 — 无效的 ARN 区域

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid ARN Region: The Region is not valid for this resource. Update the resource ARN to include a supported Region.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The Region is not valid for this resource. Update the resource ARN to include a supported Region."
```

解决错误

资源类型在指定区域不支持。有关每个区域（不带终端节点）支持的 Amazon 服务表，请参阅[区域列表](#)。

相关术语

- [策略资源 \(p. 946\)](#)
- [资源 ARN \(p. 896\)](#)
- [区域名称和代码](#)

错误 — 无效的 ARN 资源

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid ARN resource: Resource ARN does not match the expected ARN format. Update the resource portion of the ARN.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Resource ARN does not match the expected ARN format. Update the resource portion of the ARN."
```

解决错误

资源 ARN 必须与已知资源类型的规范相匹配。要查看服务的预期 ARN 格式，请参阅[Amazon 服务的操作、资源和条件键](#)。请选择服务的名称以查看其资源类型和 ARN 格式。

相关术语

- [策略资源 \(p. 946\)](#)
- [资源 ARN \(p. 896\)](#)
- [采用 ARN 格式的 Amazon 服务资源](#)

错误 — 无效的 ARN 服务案例

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid ARN service case: Update the service name in the resource ARN to use all lowercase letters.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Update the service name in the resource ARN to use all lowercase letters."
```

解决错误

资源 ARN 中的服务必须与服务前缀的规范（包括大写）匹配。要查看服务的前缀，请参阅[Amazon 服务的操作、资源和条件键](#)。请选择服务的名称并在第一句中找到服务的前缀。

相关术语

- [策略资源 \(p. 946\)](#)
- [资源 ARN \(p. 896\)](#)
- [采用 ARN 格式的 Amazon 服务资源](#)

错误 — 无效的条件数据类型

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid condition data type: The condition value data types do not match. Use condition values of the same JSON data type.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition value data types do not match. Use condition values of the same JSON data type."
```

解决错误

条件键值对中的值必须与条件键和条件运算符的数据类型匹配。要查看服务的条件键数据类型，请参阅[Amazon 服务的操作、资源和条件键](#)。请选择服务名称以查看该服务的条件键。

例如，[currentTime \(p. 1001\)](#) 全局条件键支持 Date 条件运算符。如果为条件数据块中的值提供字符串或整数，则数据类型将不匹配。

相关术语

- [条件 \(p. 948\)](#)
- [条件数据块 \(p. 950\)](#)
- [IAM JSON 策略元素：条件运算符 \(p. 951\)](#)
- [全局条件键 \(p. 998\)](#)
- [Amazon 服务条件键](#)

错误 — 无效的条件键格式

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid condition key format: The condition key format is not valid. Use the format service:keyname.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key format is not valid. Use the format service:keyname."
```

解决错误

条件键值对中的键必须与服务的规格匹配。要查看每个服务定义的条件键，请参阅 [Amazon 服务的操作、资源和条件键](#)。请选择服务名称以查看该服务的条件键。

相关术语

- [条件](#) (p. 948)
- [全局条件键](#) (p. 998)
- [Amazon 服务条件键](#)

错误 — 无效条件多个布尔值

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid condition multiple Boolean: The condition key does not support multiple Boolean values. Use a single Boolean value.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key does not support multiple Boolean values. Use a single Boolean value."
```

解决错误

条件键值对中的键需要一个布尔值。当您提供多个布尔值时，条件匹配可能不会返回您期望的结果。

要查看每个服务定义的条件键，请参阅 [Amazon 服务的操作、资源和条件键](#)。请选择服务名称以查看该服务的条件键。

- [条件](#) (p. 948)
- [全局条件键](#) (p. 998)
- [Amazon 服务条件键](#)

错误 — 无效的条件运算符

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid condition operator: The condition operator is not valid. Use a valid condition operator.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition operator is not valid. Use a valid condition operator."
```

解决错误

请更新条件以使用受支持的条件运算符。

相关术语

- [IAM JSON 策略元素：条件运算符](#) (p. 951)
- [条件元素](#) (p. 948)
- [JSON 策略概述](#) (p. 360)

错误 — 无效的效果

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid effect: The effect is not valid. Use Allow or Deny.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The effect is not valid. Use Allow or Deny."
```

解决错误

请更新 Effect 元素以使用有效的效果。Effect 的有效值为 **Allow** 和 **Deny**。

相关术语

- [效果元素 \(p. 936\)](#)
- [JSON 策略概述 \(p. 360\)](#)

错误 — 无效的全局条件键

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid global condition key: The condition key does not exist. Use a valid condition key.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key does not exist. Use a valid condition key."
```

解决错误

请更新条件键值对中的条件键以使用受支持的全局条件键。

全局条件键是带有 `aws:` 前缀的条件键。Amazon 服务既可支持全局条件键，也可提供包含服务前缀的服务特定键。例如，IAM 条件键包含 `iam:` 前缀。有关更多信息，请参阅 [Amazon 服务的操作、资源和条件键](#)，然后选择要查看其键的服务。

相关术语

- [全局条件键 \(p. 998\)](#)

错误 — 无效的分区

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid partition: The resource ARN for the service does not support the partition. Use the supported values:
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The resource ARN for the service does not support the partition. Use the supported values:"
```

解决错误

请更新资源 ARN 以包含支持的分区。如果您已包含支持的分区，则服务或资源可能不支持您包含的分区。

分区是一组 Amazon 区域。每个 Amazon 账户的作用域为一个分区。在 Classic 区域中，使用 aws 分区。在中国区域中，使用 aws-cn。

相关术语

- [Amazon Resource Names \(ARN\) - 分区](#)

错误 — 无效的策略元素

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid policy element: The policy element is not valid.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The policy element is not valid."
```

解决错误

请更新策略以仅包含受支持的 JSON 策略元素。

相关术语

- [JSON 策略元素 \(p. 934\)](#)

错误 — 无效的主体格式

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid principal format: The Principal element contents are not valid. Specify a key-value pair in the Principal element.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The Principal element contents are not valid. Specify a key-value pair in the Principal element."
```

解决错误

更新主体以使用受支持的键值对格式。

您可以在基于资源的策略中指定主体，但不能采用基于身份的策略。

例如，要为 Amazon 账户中的每个人定义访问权限，请在策略中使用以下主体：

```
"Principal": { "AWS": "123456789012" }
```

相关术语

- [JSON 策略元素 : Principal \(p. 936\)](#)
- [基于身份的策略和基于资源的策略 \(p. 379\)](#)

错误 — 无效的主体密钥

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid principal key: The principal key is not valid.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The principal key is not valid."
```

解决错误

更新主体键值对中的键以使用受支持的主体密钥。以下是受支持的主体密钥：

- Amazon
- CanonicalUser
- 联合身份
- 服务

相关术语

- [主体要素 \(p. 936\)](#)

错误 — 无效的区域

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid Region: The Region is not valid. Update the condition value to a suported Region.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The Region is not valid. Update the condition value to a suported Region."
```

解决错误

请更新条件键值对的值以包含受支持的区域。有关每个区域（不带终端节点）支持的 Amazon 服务表，请参阅[区域列表](#)。

相关术语

- [策略资源 \(p. 946\)](#)
- [资源 ARN \(p. 896\)](#)
- [区域名称和代码](#)

错误 — 无效的服务

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid service: The service does not exist. Use a valid service name.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The service does not exist. Use a valid service name."
```

解决错误

操作或条件键中的服务前缀必须与服务前缀的规范（包括大写）匹配。要查看服务的前缀，请参阅 [Amazon 服务的操作、资源和条件键](#)。请选择服务的名称并在第一句中找到服务的前缀。

相关术语

- [已知服务及其操作、资源和条件键](#)

错误 — 无效的服务条件键

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid service condition key: The condition key does not exist in the service. Use a valid condition key.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key does not exist in the service. Use a valid condition key."
```

解决错误

请更新条件键值对中的键以使用服务的已知条件键。全局条件键名称以 aws 前缀开头。Amazon 服务可提供包含服务前缀的服务特定密钥。要查看服务的前缀，请参阅 [Amazon 服务的操作、资源和条件键](#)。

相关术语

- [全局条件键 \(p. 998\)](#)
- [已知服务及其操作、资源和条件键](#)

错误 — 操作中的无效服务

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid service in action: The service specified in the action does not exist. Did you mean ?
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The service specified in the action does not exist. Did you mean ?"
```

解决错误

操作中的服务前缀必须与服务前缀的规范（包括大写）匹配。要查看服务的前缀，请参阅 [Amazon 服务的操作、资源和条件键](#)。请选择服务的名称并在第一句中找到服务的前缀。

相关术语

- [操作元素 \(p. 944\)](#)

- [已知服务及其行动](#)

错误 — 无效的运算符的变量

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid variable for operator: Policy variables can only be used with String and ARN operators.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Policy variables can only be used with String and ARN operators."
```

解决错误

您可以在 Resource 元素中和 Condition 元素的字符串比较中使用策略变量。当您使用字符串运算符或 ARN 运算符时，条件支持变量。字符串运算符包括 StringEquals、StringLike 和 StringNotLike。ARN 运算符包括 ArnEquals 和 ArnLike。不能将策略变量与其他运算符（如数字、日期、布尔值、二进制、IP 地址或 Null 运算符）一起使用。

相关术语

- [在条件元素中使用策略变量 \(p. 968\)](#)
- [条件元素 \(p. 948\)](#)

错误 — 无效的版本

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid version: The version is not valid. Use one of the following versions:
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The version is not valid. Use one of the following versions:"
```

解决错误

Version 策略元素指定 Amazon 用于处理策略的语言语法规则。要使用所有可用策略功能，将以下 Version 元素包含在所有策略中的 Statement 元素之前。

```
"Version": "2012-10-17"
```

相关术语

- [版本元素 \(p. 934\)](#)

错误 — Json 语法错误

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Json syntax error: Fix the JSON syntax error at index line column.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Fix the JSON syntax error at index line column."
```

解决错误

您的策略包含语法错误。请检查您的 JSON 语法。

相关术语

- [JSON 验证程序](#)
- [IAM JSON 策略元素参考 \(p. 934\)](#)
- [JSON 策略概述 \(p. 360\)](#)

错误 — Json 语法错误

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Json syntax error: Fix the JSON syntax error.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Fix the JSON syntax error."
```

解决错误

您的策略包含语法错误。请检查您的 JSON 语法。

相关术语

- [JSON 验证程序](#)
- [IAM JSON 策略元素参考 \(p. 934\)](#)
- [JSON 策略概述 \(p. 360\)](#)

错误 — 缺少操作

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Missing action: Add an Action or NotAction element to the policy statement.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add an Action or NotAction element to the policy statement."
```

解决错误

Amazon JSON 策略必须包含 Action 或者 NotAction 元素。

相关术语

- [操作元素 \(p. 944\)](#)
- [NotAction 元素 \(p. 945\)](#)
- [JSON 策略概述 \(p. 360\)](#)

错误 — 缺少 ARN 字段

在 Amazon Web Services Management Console 中，则此检查的结果包括以下消息：

```
Missing ARN field: Resource ARNs must include at least fields in the following structure:
arn:partition:service:region:account:resource
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Resource ARNs must include at least fields in the following structure:
arn:partition:service:region:account:resource"
```

解决错误

资源 ARN 中的所有字段都必须与已知资源类型的规范相匹配。要查看服务的预期 ARN 格式，请参阅[Amazon 服务的操作、资源和条件键](#)。请选择服务的名称以查看其资源类型和 ARN 格式。

相关术语

- [策略资源 \(p. 946\)](#)
- [资源 ARN \(p. 896\)](#)
- [采用 ARN 格式的 Amazon 服务资源](#)

错误 — 缺少 ARN 区域

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Missing ARN Region: Add a Region to the resource ARN.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a Region to the resource ARN."
```

解决错误

大多数 Amazon 服务的资源 ARN 都需要您指定一个区域。有关每个区域（不带终端节点）支持的 Amazon 服务表，请参阅[区域列表](#)。

相关术语

- [策略资源 \(p. 946\)](#)
- [资源 ARN \(p. 896\)](#)
- [区域名称和代码](#)

错误 — 缺少效果

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Missing effect: Add an Effect element to the policy statement with a value of Allow or
Deny.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add an Effect element to the policy statement with a value of Allow or Deny."
```

解决错误

Amazon JSON 策略必须包含 Effect 元素，其值为 **Allow** 和 **Deny**。

相关术语

- [效果元素 \(p. 936\)](#)
- [JSON 策略概述 \(p. 360\)](#)

错误 — 缺少主体

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Missing principal: Add a Principal element to the policy statement.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a Principal element to the policy statement."
```

解决错误

基于资源的策略必须包含 Principal 元素。

例如，要为 Amazon 账户中的每个人定义访问权限，请在策略中使用以下主体：

```
"Principal": { "AWS": "123456789012" }
```

相关术语

- [主体要素 \(p. 936\)](#)
- [基于身份的策略和基于资源的策略 \(p. 379\)](#)

错误 — 缺少限定符

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Missing qualifier: The request context key has multiple values. Use the ForAllValues or ForAnyValue condition key qualifiers in your policy.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The request context key has multiple values. Use the ForAllValues or ForAnyValue condition key qualifiers in your policy."
```

解决错误

在 Condition 元素中，您可构建表达式并使用条件运算符（等于、小于等）将策略中的条件键和值与请求上下文中的键和值进行匹配。对于包含单个条件键的多个值的请求，必须将条件括在方括号内，例如数组 ("Key2":["Value2A", "Value2B"])。您还必须将 ForAllValues 或 ForAnyValue 集合运算符与 StringLike 条件运算符搭配使用。这些限定词为条件运算符添加了集合运算功能，以便您针对多个条件值测试多个请求值。

相关术语

- [使用多种条件键和值 \(p. 960\)](#)
- [条件元素 \(p. 948\)](#)

出现此错误的 Amazon 托管策略

[Amazon 托管策略 \(p. 364\)](#)将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

以下 Amazon 托管策略在其策略语句中包含缺少的条件键限定符。使用 Amazon 托管策略作为创建客户托管策略的参考时，Amazon 会建议您将 `ForAllValues` 或 `ForAnyValue` 条件键限定符添加到 `Condition` 元素。

- [AWSGlueConsoleSageMakerNotebookFullAccess](#)

错误 — 缺少资源

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Missing resource: Add a Resource or NotResource element to the policy statement.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a Resource or NotResource element to the policy statement."
```

解决错误

基于身份的策略必须包含 `Resource` 或者 `NotResource` 元素。

相关术语

- [资源元素 \(p. 946\)](#)
- [NotResource 元素 \(p. 948\)](#)
- [基于身份的策略和基于资源的策略 \(p. 379\)](#)
- [JSON 策略概述 \(p. 360\)](#)

错误 — 缺少语句

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Missing statement: Add a statement to the policy
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a statement to the policy"
```

解决错误

JSON 策略必须包含语句。

相关术语

- [JSON 策略元素 \(p. 934\)](#)

错误 — 如果存在则为空

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Null with if exists: The Null condition operator cannot be used with the IfExists suffix.  
Update the operator or the suffix.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The Null condition operator cannot be used with the IfExists suffix.  
Update the operator or the suffix."
```

解决错误

除 Null 条件运算符外，您可在任何条件运算符名称的末尾添加 `IfExists`。使用 Null 条件运算符检查授权时是否有条件键。使用 `...ifExists` “如果请求的内容中存在策略键，则依照策略所述来处理键。如果该键不存在，则条件元素的计算结果将为 `true`。”

相关术语

- [...IfExists 条件运算符 \(p. 957\)](#)
- [Null 条件运算符 \(p. 958\)](#)
- [条件元素 \(p. 948\)](#)

错误 — SCP 语法错误操作通配符

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
SCP syntax error action wildcard: SCP actions can include wildcards (*) only at the end of  
a string. Update.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "SCP actions can include wildcards (*) only at the end of a string.  
Update."
```

解决错误

Amazon Organizations 服务控制策略 (SCP) 支持在 `Action` 或 `NotAction` 元素中指定值。但是，这些值只能在字符串末尾包含通配符 (*)。这意味着您可以指定 `iam:Get*` 但不能指定 `iam:*role`。

要指定多个操作，Amazon 建议您单独列出它们。

相关术语

- [SCP 操作与 NotAction 元素](#)
- [有关使用 SCP 的策略](#)
- [Amazon Organizations 服务控制策略](#)
- [IAM JSON 策略元素 : Action \(p. 944\)](#)

错误 — SCP 语法错误允许条件

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

SCP syntax error allow condition: SCPs do not support the Condition element with effect Allow. Update the element Condition or the effect.

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

"findingDetails": "SCPs do not support the Condition element with effect Allow. Update the element Condition or the effect."

解决错误

Amazon Organizations 服务控制策略 (SCP) 仅支持当您在使用 "Effect": "Deny" 时指定 Condition 元素。

要仅允许执行单个操作，您可以拒绝除对使用 ...NotEquals 版本条件运算符指定的条件以外的所有访问。这否定了运算符所做的比较。

相关术语

- [SCP 条件元素](#)
- [有关使用 SCP 的策略](#)
- [Amazon Organizations 服务控制策略](#)
- [示例策略：根据请求的区域拒绝对 Amazon 的访问 \(p. 404\)](#)
- [IAM JSON 策略元素：条件运算符 \(p. 951\)](#)
- [IAM JSON 策略元素：Condition \(p. 948\)](#)

错误 — SCP 语法错误允许 NotAction

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

SCP syntax error allow NotAction: SCPs do not support NotAction with effect Allow. Update the element NotAction or the effect.

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

"findingDetails": "SCPs do not support NotAction with effect Allow. Update the element NotAction or the effect."

解决错误

Amazon Organizations 服务控制策略 (SCP) 不支持使用带有元素 "Effect": "Allow" 的 NotAction。

您必须重写逻辑以允许操作列表或拒绝所有未列出的操作。

相关术语

- [SCP 操作与 NotAction 元素](#)
- [有关使用 SCP 的策略](#)
- [Amazon Organizations 服务控制策略](#)
- [IAM JSON 策略元素：Action \(p. 944\)](#)

错误 — SCP 语法错误允许资源

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

SCP syntax error allow resource: SCPs do not support Resource with effect Allow. Update the element Resource or the effect.

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

"findingDetails": "SCPs do not support Resource with effect Allow. Update the element Resource or the effect."

解决错误

Amazon Organizations 服务控制策略 (SCP) 仅支持当您在使用 "Effect": "Deny" 时指定 Resource 元素。

您必须重写逻辑以允许所有资源或拒绝列出的每个资源。

相关术语

- [SCP Resource 元素](#)
- [有关使用 SCP 的策略](#)
- [Amazon Organizations 服务控制策略](#)
- [IAM JSON 策略元素：Resource \(p. 946\)](#)

错误 — SCP 语法错误 NotResource

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

SCP syntax error NotResource: SCPs do not support the NotResource element. Update the policy to use Resource instead.

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

"findingDetails": "SCPs do not support the NotResource element. Update the policy to use Resource instead."

解决错误

Amazon Organizations 服务控制策略 (SCP) 不支持 NotResource 元素。

您必须重写逻辑以允许所有资源或拒绝列出的每个资源。

相关术语

- [SCP Resource 元素](#)
- [有关使用 SCP 的策略](#)
- [Amazon Organizations 服务控制策略](#)
- [IAM JSON 策略元素：Resource \(p. 946\)](#)

错误 — SCP 语法错误主体

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

SCP syntax error principal: SCPs do not support specifying principals. Remove the Principal or NotPrincipal element.

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "SCPs do not support specifying principals. Remove the Principal or NotPrincipal element."
```

解决错误

Amazon Organizations 服务控制策略 (SCP) 不支持 Principal 或 NotPrincipal 元素。

您可以使用 `aws:PrincipalArn` 全局条件键在 Condition 元素中指定 Amazon Resource Name (ARN)。

相关术语

- [SCP 语法](#)
- [主体的全局条件键 \(p. 1004\)](#)

错误 — 需要唯一的 Sid

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Unique Sids required: Duplicate statement IDs are not supported for this policy type.  
Update the Sid value.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Duplicate statement IDs are not supported for this policy type. Update  
the Sid value."
```

解决错误

对于某些策略类型，语句 ID 必须是唯一的。sid (声明 ID) 元素允许您针对策略语句输入可选标识符。您可以在使用 sid 元素的语句数组中为每个语句指定语句 ID 的值。在允许您指定 ID 元素的产品中，例如 SQS 和 SNS，sid 值正是策略文件 ID 的子 ID。例如，在 IAM 中，sid 值必须在 JSON 策略中是唯一的。

相关术语

- [IAM JSON 策略元素：Sid \(p. 936\)](#)

错误 — 策略中不支持的操作

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Unsupported action in policy: The action is not supported for the resource-based policy  
attached to the resource type.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The action is not supported for the resource-based policy attached to  
the resource type."
```

解决错误

附加到不同资源类型的基于资源策略中的 Action 元素不支持某些操作。例如，Amazon S3 存储桶策略中不支持 Amazon Key Management Service 操作。指定附加到基于资源的策略的资源类型支持的操作。

相关术语

- [JSON 策略元素 : Action \(p. 944\)](#)

错误 — 不支持的元素组合

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Unsupported element combination: The policy elements and can not be used in the same statement. Remove one of these elements.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The policy elements and can not be used in the same statement. Remove one of these elements."
```

解决错误

某些 JSON 策略元素的组合不能一起使用。例如，您不能在同一策略语句中同时使用 Action 和 NotAction。相互排斥的其他对包括 Principal/NotPrincipal 和 Resource/NotResource。

相关术语

- [IAM JSON 策略元素参考 \(p. 934\)](#)
- [JSON 策略概述 \(p. 360\)](#)

错误 — 不支持的全局条件键

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Unsupported global condition key: The condition key aws:ARN is not supported. Use aws:PrincipalArn or aws:SourceArn instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key aws:ARN is not supported. Use aws:PrincipalArn or aws:SourceArn instead."
```

解决错误

Amazon 不支持使用指定的全局条件键。根据您的使用案例，您可以使用 aws:PrincipalArn 或 aws:SourceArn 全局条件键。例如，与 aws:ARN 不同，使用 aws:PrincipalArn 将发出请求的主体的 Amazon Resource Name (ARN) 与您在策略中指定的 ARN 进行比较。或者，使用 aws:SourceArn 全局条件键可将发出服务到服务请求的资源的 Amazon Resource Name (ARN) 与您在策略中指定的 ARN 进行比较。

相关术语

- [Amazon 全局条件上下文密钥 \(p. 998\)](#)

错误 — 不支持的主体

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Unsupported principal: The policy type does not support the Principal element. Remove the Principal element.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The policy type does not support the Principal element. Remove the Principal element."
```

解决错误

Principal 元素指定允许或拒绝访问资源的主体。无法在基于 IAM 身份的策略中使用 Principal 元素。可以在 IAM 角色的信任策略和基于资源的策略中使用它。基于资源的策略是直接嵌入资源中的策略。例如，您可以在 Amazon S3 存储桶或 Amazon KMS 键中嵌入策略。

相关术语

- [Amazon JSON 策略元素：Principal \(p. 936\)](#)
- [IAM 角色与基于资源的策略有何不同 \(p. 274\)](#)

错误 — 策略中不支持的资源 ARN

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Unsupported resource ARN in policy: The resource ARN is not supported for the resource-based policy attached to the resource type.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The resource ARN is not supported for the resource-based policy attached to the resource type."
```

解决错误

策略附加到不同资源类型时，基于资源策略中的 Resource 元素不支持某些操作。例如，Amazon S3 存储桶策略的 Resource 元素中不支持 Amazon KMS ARN。指定附加到基于资源的策略的资源类型支持的资源 ARN。

相关术语

- [JSON 策略元素：Action \(p. 944\)](#)

错误 — 不支持的 Sid

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Unsupported Sid: Update the characters in the Sid element to use one of the following character types: [a-z, A-Z, 0-9]
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Update the characters in the Sid element to use one of the following character types: [a-z, A-Z, 0-9]"
```

解决错误

Sid 元素支持大写字母、小写字母和数字。

相关术语

- [IAM JSON 策略元素 : Sid \(p. 936\)](#)

错误 — 主体中不支持的通配符

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Unsupported wildcard in principal: Wildcards (*, ?) are not supported with the principal key. Replace the wildcard with a valid principal value.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Wildcards (*, ?) are not supported with the principal key. Replace the wildcard with a valid principal value."
```

解决错误

Principal 元素结构支持使用键值对。策略中指定的主体值包含通配符 (*)。您不能在指定的主体密钥中包含通配符。当在 Principal 元素中指定用户时，不能使用通配符表示“所有用户”。您必须对一个特定用户或多个用户命名。同样地，当您指定代入角色会话时，不能使用通配符表示“所有会话”。您必须指明特定会话。同时，您不能使用通配符匹配一部分名称或 ARN。

要解决此结果，请移除通配符并提供更具体的主体。

相关术语

- [Amazon JSON 策略元素 : Principal \(p. 936\)](#)

错误 — 变量中缺少大括号

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Missing brace in variable: The policy variable is missing a closing curly brace. Add } after the variable text.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The policy variable is missing a closing curly brace. Add } after the variable text."
```

解决错误

策略变量结构支持使用 \$ 前缀，后跟一对大括号 ({ })。在 \${ } 字符内，包含想要在策略中使用的请求中的值名称。

若要解决此结果，请添加缺少的大括号，以确保存在完整的大括号开闭集。

相关术语

- [IAM policy 元素 : 变量 \(p. 964\)](#)

错误 — 变量中不支持的符号

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Unsupported symbol in variable: The symbol is not supported within the policy variable text. Remove the symbol.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The symbol is not supported within the policy variable text. Remove the symbol."
```

解决错误

策略变量结构支持使用 \$ 前缀，后跟一对大括号 ({ })。在 \${ } 字符内，包含想要在策略中使用的请求中的值名称。

要解决此结果，请移除不支持的符号。

相关术语

- [IAM policy 元素：变量 \(p. 964\)](#)

错误 — 变量中不支持的符号

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Unsupported symbol in variable: The symbol is not supported within the policy variable. Use instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The symbol is not supported within the policy variable. Use instead."
```

解决错误

策略变量结构支持使用 \$ 前缀，后跟一对大括号 ({ })。在 \${ } 字符内，包含想要在策略中使用的请求中的值名称。

要解决此结果，请移除不支持的符号。

相关术语

- [IAM policy 元素：变量 \(p. 964\)](#)

错误 — 变量中缺少引号

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Missing quote in variable: The policy variable default value must begin and end with a single quote. Add the missing quote.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The policy variable default value must begin and end with a single quote. Add the missing quote."
```

解决错误

向策略添加变量时，您可以为变量指定默认值。如果不存在变量，则 Amazon 使用您提供的原定设置文本。

要向变量添加默认值，请用单引号 (' ') 括起默认值，并用逗号和空格 (,) 分隔变量文本和默认值。

例如，如果主体使用 `team=yellow` 标记，他们可以访问名为 `DOC-EXAMPLE-BUCKET-yellow` 的 `DOC-EXAMPLE-BUCKET` Amazon S3 存储桶。使用此资源的策略可能允许团队成员访问自己的资源，但不能访问其他团队的资源。对于没有团队标签的用户，您可以将原定设置值设为 `company-wide`。这些用户只能访问 `DOC-EXAMPLE-BUCKET-company-wide` 存储桶，他们可以在其中查看广泛的信息，例如加入团队的说明。

```
"Resource": "arn:aws:s3::DOC-EXAMPLE-BUCKET-${aws:PrincipalTag/team, 'company-wide'}"
```

相关术语

- [IAM policy 元素：变量 \(p. 964\)](#)

错误 — 变量中不支持的空间

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Unsupported space in variable: A space is not supported within the policy variable text.  
Remove the space.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "A space is not supported within the policy variable text. Remove the  
space."
```

解决错误

策略变量结构支持使用 `$` 前缀，后跟一对大括号 ({ })。在 `${ }` 字符内，包含想要在策略中使用的请求中的值名称。虽然在指定默认变量时可以包含空格，但不能在变量名称中包含空格。

相关术语

- [IAM policy 元素：变量 \(p. 964\)](#)

错误 — 空变量

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Empty variable: Empty policy variable. Remove the variable structure or provide a variable  
within the structure.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Empty policy variable. Remove the variable structure or provide a  
variable within the structure."
```

解决错误

策略变量结构支持使用 `$` 前缀，后跟一对大括号 (`{ }`)。在 `${ }` 字符内，包含想要在策略中使用的请求中的值名称。

相关术语

- [IAM policy 元素：变量 \(p. 964\)](#)

错误 — 元素中不支持变量

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Variable unsupported in element: Policy variables are supported in the Resource and Condition elements. Remove the policy variable from this element.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Policy variables are supported in the Resource and Condition elements. Remove the policy variable from this element."
```

解决错误

您可以在 `Resource` 元素中和 `Condition` 元素的字符串比较中使用策略变量。

相关术语

- [IAM policy 元素：变量 \(p. 964\)](#)

错误 — 版本中不支持变量

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Variable unsupported in version: To include variables in your policy, use the policy version 2012-10-17 or later.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "To include variables in your policy, use the policy version 2012-10-17 or later."
```

解决错误

要使用策略变量，您必须在语句中包含 `Version` 元素，而且版本必须设置为支持策略变量的版本。变量是在 2012-10-17 版本中引入的。较早版本的策略语言不支持策略变量。如果您未将 `Version` 设置为 2012-10-17 或更高版本，则 `${aws:username}` 等变量在策略中被视为文字字符串。

`Version` 策略元素与策略版本不同。`Version` 策略元素用在策略之中，用于定义策略语言的版本。当您更改 IAM 中的客户托管策略时，将创建一个策略版本。已更改的策略不会覆盖现有策略。而是由 IAM 创建新的托管策略版本。

相关术语

- [IAM policy 元素：变量 \(p. 964\)](#)
- [IAM JSON 策略元素：Version \(p. 934\)](#)

错误 — 服务主体的条件密钥不受支持

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Unsupported condition key for service principal: The following condition keys are not supported when used with the service principal:.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The following condition keys are not supported when used with the service principal:."
```

解决错误

您可以使用服务主体指定基于资源策略的 Principal 元素中的 Amazon Web Services，这是服务的标识符。您不能对某些服务主体使用某些条件密钥。例如，您不能使用带有服务主体 `cloudfront.amazonaws.com` 的 `aws:PrincipalOrgID` 条件密钥。您应该移除 Principal 元素中不适用于服务主体的条件密钥。

相关术语

- [服务主体](#)
- [JSON 策略元素：Principal \(p. 936\)](#)

错误 — 私有 IP 地址

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Private IP address: aws:SourceIp works only for public IP address ranges. The values for condition key aws:SourceIp include only private IP addresses and will not have the desired effect. Update the value to include only public IP addresses.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "aws:SourceIp works only for public IP address ranges. The values for condition key aws:SourceIp include only private IP addresses and will not have the desired effect. Update the value to include only public IP addresses."
```

解决错误

全局条件键 `aws:SourceIp` 仅适用于公有 IP 地址范围。当您的策略仅允许私有 IP 地址时，您会收到此错误。在这种情况下，条件将永远不匹配。

- [aws:SourceIp 全局条件键 \(p. 1010\)](#)
- [IAM JSON 策略元素：Condition \(p. 948\)](#)

错误 — 私有 NotIpAddress

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Private NotIpAddress: The values for condition key aws:SourceIp include only private IP addresses and has no effect. aws:SourceIp works only for public IP address ranges. Update the value to include only public IP addresses.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The values for condition key aws:SourceIp include only private IP addresses and has no effect. aws:SourceIp works only for public IP address ranges. Update the value to include only public IP addresses."
```

解决错误

全局条件键 `aws:SourceIp` 仅适用于公有 IP 地址范围。当您使用 `NotIpAddress` 条件运算符并仅列出私有 IP 地址时，就会收到此错误。在这种情况下，条件将始终匹配并且无效。

- [aws:SourceIp 全局条件键 \(p. 1010\)](#)
- [IAM JSON 策略元素：Condition \(p. 948\)](#)

错误 — 策略大小超过 SCP 配额

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Policy size exceeds SCP quota: The characters in the service control policy (SCP) exceed the character maximum for SCPs. We recommend that you use multiple granular policies.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The characters in the service control policy (SCP) exceed the character maximum for SCPs. We recommend that you use multiple granular policies."
```

解决错误

Amazon Organizations 服务控制策略 (SCP) 支持在 `Action` 或 `NotAction` 元素中指定值。但是，这些值只能在字符串末尾包含通配符 (*)。这意味着您可以指定 `iam:Get*` 但不能指定 `iam:*role`。

要指定多个操作，Amazon 建议您单独列出它们。

相关术语

- [Amazon Organizations 的配额](#)
- [Amazon Organizations 服务控制策略](#)

错误 — 无效的服务主体格式

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid service principal format: The service principal does not match the expected format. Use the format with all lowercase letters.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The service principal does not match the expected format. Use the format with all lowercase letters."
```

解决错误

条件键值对中的值必须与定义的服务主体格式匹配。

服务主体是一个标识符，用于向服务授予权限。您可以在 `Principal` 元素中指定服务主体，也可以将其作为某些全局条件键和特定于服务的键的值。服务主体由每项服务定义。

服务主体的标识符包括服务名称，通常采用以下格式（都是小写字母）：

`service-name.amazonaws.com`

某些特定于服务的键可能会对服务主体使用不同的格式。例如，`kms:ViaService` 条件密钥要求所有小写字母的服务主体采用以下格式：

`service-name.AWS_region.amazonaws.com`

相关术语

- [服务主体](#)
- [Amazon 全局条件键 \(p. 998\)](#)
- [kms:ViaService 条件键](#)

错误 — 条件中缺少标签密钥

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Missing tag key in condition: The condition key must include a tag key to control access based on tags. Use the format tag-key and specify a key name for tag-key.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key must include a tag key to control access based on tags. Use the format tag-key and specify a key name for tag-key."
```

解决错误

要基于标签控制访问，您需要在策略的[条件元素 \(p. 948\)](#)中提供标签信息。

例如，要[控制对 Amazon 资源的访问权限](#)，您可以包含 `aws:ResourceTag` 条件键。此键需要格式 `aws:ResourceTag/tag-key`。要指定条件中的标签密钥 `owner` 和标签值 `JaneDoe`，请使用以下格式。

```
"Condition": {
  "StringEquals": {"aws:ResourceTag/owner": "JaneDoe"}
}
```

相关术语

- [使用标签控制访问 \(p. 388\)](#)
- [条件 \(p. 948\)](#)
- [全局条件键 \(p. 998\)](#)
- [Amazon 服务条件键](#)

错误 — VPC 格式无效

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid vpc format: The VPC identifier in the condition key value is not valid. Use the prefix 'vpc-' followed by 8 or 17 alphanumeric characters.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The VPC identifier in the condition key value is not valid. Use the prefix 'vpc-' followed by 8 or 17 alphanumeric characters."
```

解决错误

aws:SourceVpc 条件密钥必须使用前缀 vpc-，并且后面跟 8 或 17 个字母数字字符，例如 vpc-11223344556677889 或 vpc-12345678。

相关术语

- [Amazon 全局条件密钥](#)：aws:SourceVpc

错误 — vpce 格式无效

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid vpce format: The VPCE identifier in the condition key value is not valid. Use the prefix 'vpce-' followed by 8 or 17 alphanumeric characters.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The VPCE identifier in the condition key value is not valid. Use the prefix 'vpce-' followed by 8 or 17 alphanumeric characters."
```

解决错误

aws:SourceVpce 条件密钥必须使用前缀 vpce-，并且后面跟 8 或 17 个字母数字字符，例如 vpce-11223344556677889 或 vpce-12345678。

相关术语

- [Amazon 全局条件密钥](#)：aws:SourceVpce

错误 — 不支持联合主体

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Federated principal not supported: The policy type does not support a federated identity provider in the principal element. Use a supported principal.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The policy type does not support a federated identity provider in the principal element. Use a supported principal."
```

解决错误

这些区域有：Principal 元素将联合主体用于附加到 IAM 角色的信任策略，以便通过联合身份验证提供访问权限。标识策略和其他基于资源的策略不支持 Principal 元素中的联合标识提供程序。例如，您不能在 Amazon S3 存储桶策略中使用 SAML 主体。将 Principal 元素更改为受支持的主体类型。

相关术语

- [创建适用于身份联合验证的角色 \(p. 224\)](#)
- [JSON 策略元素 : Principal \(p. 936\)](#)

错误 — 条件密钥的不支持的操作

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Unsupported action for condition key: The following actions: are not supported by the
condition key. The condition will not be evaluated for these actions. We recommend that
you move these actions to a different statement without this condition key.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The following actions: are not supported by the condition key. The
condition will not be evaluated for these actions. We recommend that you move these
actions to a different statement without this condition key."
```

解决错误

确保策略声明 Condition 元素中的条件密钥适用于 Action 元素中的每个操作。为确保您的策略有效允许或拒绝指定的操作，应将不受支持的操作移动到不带条件密钥的其他语句中。

Note

如果 Action 元素具有带通配符的操作，IAM Access Analyzer 不会对此错误的这些操作进行评估。

相关术语

- [JSON 策略元素 : Action \(p. 944\)](#)

常规警告 — 使用 NotResource 创建 SLR

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Create SLR with NotResource: Using the iam:CreateServiceLinkedRole action with NotResource
can allow creation of unintended service-linked roles for multiple resources. We recommend
that you specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the iam:CreateServiceLinkedRole action with NotResource can allow
creation of unintended service-linked roles for multiple resources. We recommend that you
specify resource ARNs instead."
```

解决一般警告

操作 iam:CreateServiceLinkedRole 授予权限以创建 IAM 角色，它允许 Amazon 服务代表您执行操作。在带有 NotResource 元素的策略中使用 iam:CreateServiceLinkedRole 可以允许为多个资源创建意外的服务关联角色。Amazon 会建议您在 Resource 元素中指定允许的 ARN。

- [CreateServiceLinkedRole 操作](#)
- [IAM JSON 策略元素 : NotResource \(p. 948\)](#)

- [IAM JSON 策略元素 : Resource \(p. 946\)](#)

常规警告 — 使用带有星号的操作和 NotResource 创建 SLR

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Create SLR with star in action and NotResource: Using an action with a wildcard(*) and NotResource can allow creation of unintended service-linked roles because it can allow iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using an action with a wildcard(*) and NotResource can allow creation of unintended service-linked roles because it can allow iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you specify resource ARNs instead."
```

解决一般警告

操作 `iam:CreateServiceLinkedRole` 授予权限以创建 IAM 角色，它允许 Amazon 服务代表您执行操作。在 Action 中具有通配符 (*) 且 NotResource 元素的策略可以允许为多个资源创建意外的服务关联角色。Amazon 会建议您在 Resource 元素中指定允许的 ARN。

- [CreateServiceLinkedRole 操作](#)
- [IAM JSON 策略元素 : NotResource \(p. 948\)](#)
- [IAM JSON 策略元素 : Resource \(p. 946\)](#)

常规警告 — 使用 NotAction 和 NotResource 创建 SLR

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Create SLR with NotAction and NotResource: Using NotAction with NotResource can allow creation of unintended service-linked roles because it allows iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using NotAction with NotResource can allow creation of unintended service-linked roles because it allows iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you specify resource ARNs instead."
```

解决一般警告

操作 `iam:CreateServiceLinkedRole` 授予权限以创建 IAM 角色，它允许 Amazon 服务代表您执行操作。将 NotAction 元素与 NotResource 元素搭配使用可以允许为多个资源创建意外的服务链接角色。Amazon 会建议您重写策略以允许 Resource 元素中 ARN 有限列表上的 `iam:CreateServiceLinkedRole`。您还可以将 `iam:CreateServiceLinkedRole` 添加到 NotAction 元素。

- [CreateServiceLinkedRole 操作](#)
- [IAM JSON 策略元素 : NotAction \(p. 945\)](#)
- [IAM JSON 策略元素 : Action \(p. 944\)](#)
- [IAM JSON 策略元素 : NotResource \(p. 948\)](#)
- [IAM JSON 策略元素 : Resource \(p. 946\)](#)

常规警告 — 使用带星号的资源创建 SLR

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Create SLR with star in resource: Using the iam:CreateServiceLinkedRole action with wildcards (*) in the resource can allow creation of unintended service-linked roles. We recommend that you specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the iam:CreateServiceLinkedRole action with wildcards (*) in the resource can allow creation of unintended service-linked roles. We recommend that you specify resource ARNs instead."
```

解决一般警告

操作 `iam:CreateServiceLinkedRole` 授予权限以创建 IAM 角色，它允许 Amazon 服务代表您执行操作。在带有包含通配符 (*) 的 Resource 元素的策略中使用 `iam:CreateServiceLinkedRole` 可以允许为多个资源创建意外的服务关联角色。Amazon 会建议您在 Resource 元素中指定允许的 ARN。

- [CreateServiceLinkedRole 操作](#)
- [IAM JSON 策略元素：Resource \(p. 946\)](#)

带有此常规警告的 Amazon 托管策略

[Amazon 托管策略 \(p. 364\)](#)将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

其中一些使用案例适用于您的账户中的高级用户。以下 Amazon 托管策略提供高级用户访问权限并授予为任何 Amazon 服务创建[服务关联角色 \(p. 208\)](#)的权限。Amazon 建议您仅将以下 Amazon 托管策略附加到您认为是高级用户的 IAM 身份。

- [PowerUserAccess](#)
- [AlexaForBusinessFullAccess](#)
- [AWSOrganizationsServiceTrustPolicy](#) — 此 Amazon 托管策略提供了权限以供 Amazon Organizations 服务关联角色使用。此角色允许 Organizations 为 Amazon 企业中的其他服务创建额外的服务关联角色。

常规警告 — 使用带星号的行动和资源创建 SLR

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Create SLR with star in action and resource: Using wildcards (*) in the action and the resource can allow creation of unintended service-linked roles because it allows iam:CreateServiceLinkedRole permissions on all resources. We recommend that you specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using wildcards (*) in the action and the resource can allow creation of unintended service-linked roles because it allows iam:CreateServiceLinkedRole permissions on all resources. We recommend that you specify resource ARNs instead."
```

解决一般警告

操作 `iam:CreateServiceLinkedRole` 授予权限以创建 IAM 角色，它允许 Amazon 服务代表您执行操作。在 Action 和 Resource 元素中带有通配符 (*) 的策略可以允许为多个资源创建意外的服务关联角色。

这允许您在指定 "Action": "*"、"Action": "iam:*" 或 "Action": "iam:Create*" 时创建服务关联角色。Amazon 会建议您在 Resource 元素中指定允许的 ARN。

- [CreateServiceLinkedRole 操作](#)
- [IAM JSON 策略元素 : Action \(p. 944\)](#)
- [IAM JSON 策略元素 : Resource \(p. 946\)](#)

带有此常规警告的 Amazon 托管策略

[Amazon 托管策略 \(p. 364\)](#)将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

其中一些使用案例适用于您账户中的管理员。以下 Amazon 托管策略提供管理员访问权限并授予为任何 Amazon 服务创建[服务关联角色 \(p. 208\)](#)的权限。Amazon 建议您仅将以下 Amazon 托管策略附加到您认为是管理员的 IAM 身份。

- [AdministratorAccess](#)
- [IAMFullAccess](#)

常规警告 — 使用带有星号的资源和 NotAction 创建 SLR

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Create SLR with star in resource and NotAction: Using a resource with wildcards (*) and NotAction can allow creation of unintended service-linked roles because it allows iam:CreateServiceLinkedRole permissions on all resources. We recommend that you specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using a resource with wildcards (*) and NotAction can allow creation of unintended service-linked roles because it allows iam:CreateServiceLinkedRole permissions on all resources. We recommend that you specify resource ARNs instead."
```

解决一般警告

操作 `iam:CreateServiceLinkedRole` 授予权限以创建 IAM 角色，它允许 Amazon 服务代表您执行操作。在带有包含通配符 (*) 的 Resource 元素的策略中使用 NotAction 元素可以允许为多个资源创建意外的服务关联角色。Amazon 会建议您在 Resource 元素中指定允许的 ARN。您还可以将 `iam:CreateServiceLinkedRole` 添加到 NotAction 元素。

- [CreateServiceLinkedRole 操作](#)
- [IAM JSON 策略元素 : NotAction \(p. 945\)](#)
- [IAM JSON 策略元素 : Action \(p. 944\)](#)
- [IAM JSON 策略元素 : Resource \(p. 946\)](#)

常规警告 — 已弃用的全局条件键

在 Amazon Web Services Management Console 中，则此检查的结果包括以下消息：

```
Deprecated global condition key: We recommend that you update aws:ARN to use the newer condition keys aws:PrincipalArn or aws:SourceArn.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "We recommend that you update aws:ARN to use the newer condition keys  
aws:PrincipalArn or aws:SourceArn."
```

解决一般警告

该策略包含已弃用的全局条件键。请更新条件键值对中的条件键以使用受支持的全局条件键。

- [全局条件键 \(p. 998\)](#)

常规警告 — 无效的日期值

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid date value: The date might not resolve as expected. We recommend that you use the  
YYYY-MM-DD format.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The date might not resolve as expected. We recommend that you use the  
YYYY-MM-DD format."
```

解决一般警告

Unix Epoch 时间描述自 1970 年 1 月 1 日以来已经过去的时间点，减去闰秒。Epoch 时间可能无法解析到您期望的精确时间。Amazon 建议您对日期和时间格式使用 W3C 标准。例如，您可以指定一个完整的日期，如 YYYY-MM-DD (1997-07-16)，也可以将时间附加到秒，例如 YYYY-MM-DDThh:mm:ssTZD (1997-07-16T19:20:30+01:00)。

- [W3C 日期和时间格式](#)
- [IAM JSON 策略元素：Version \(p. 934\)](#)
- [aws:CurrentTime 全局条件键 \(p. 1001\)](#)

常规警告 — 无效的角色引用

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid role reference: The Principal element includes the IAM role ID. We recommend that  
you use a role ARN instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The Principal element includes the IAM role ID. We recommend that you  
use a role ARN instead."
```

解决一般警告

Amazon 建议您为 IAM 角色指定 Amazon Resource Name (ARN)，而不是其主体 ID。当 IAM 保存策略时，它会将 ARN 转换为现有角色的主体 ID。Amazon 包括安全预防措施。如果某人删除并重新创建该角色，则该角色将有一个新 ID，并且策略不会与新角色的 ID 匹配。

- [指定主体：IAM 角色 \(p. 938\)](#)
- [IAM ARN \(p. 896\)](#)
- [IAM 唯一 ID \(p. 900\)](#)

常规警告 — 无效的用户引用

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Invalid user reference: The Principal element includes the IAM user ID. We recommend that you use a user ARN instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The Principal element includes the IAM user ID. We recommend that you use a user ARN instead."
```

解决一般警告

Amazon 建议您为 IAM 用户指定 Amazon Resource Name (ARN)，而不是其主体 ID。当 IAM 保存策略时，它会将 ARN 转换为现有用户的主体 ID。Amazon 包括安全预防措施。如果某人删除并重新创建该用户，则该用户将有一个新 ID，并且策略不会与新用户的 ID 匹配。

- [指定主体：IAM 用户 \(p. 939\)](#)
- [IAM ARN \(p. 896\)](#)
- [IAM 唯一 ID \(p. 900\)](#)

常规警告 — 缺少版本

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Missing version: We recommend that you specify the Version element to help you with debugging permission issues.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "We recommend that you specify the Version element to help you with debugging permission issues."
```

解决一般警告

Amazon 建议在您的策略中包括可选的 Version 参数。如果未包含 Version 元素，则此值默认为 2012-10-17，但策略变量等较新的功能将不适用于您的策略。例如，系统不会将 `${aws:username}` 等变量识别为变量，而是将其视为策略中的文本字符串。

- [IAM JSON 策略元素：Version \(p. 934\)](#)

一般警告 — 建议的使用唯一的 Sid

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Unique Sids recommended: We recommend that you use statement IDs that are unique to your policy. Update the Sid value.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "We recommend that you use statement IDs that are unique to your policy. Update the Sid value."
```

解决一般警告

Amazon 建议您使用唯一语句 ID。Sid (声明 ID) 元素允许您针对策略语句输入可选标识符。您可以在使用 SID 元素的语句数组中为每个语句指定语句 ID 的值。

相关术语

- [IAM JSON 策略元素 : Sid \(p. 936\)](#)

常规警告 — 没有类似运算符的通配符

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Wildcard without like operator: Your condition value includes a * or ? character. If you meant to use a wildcard (*, ?), update the condition operator to include Like.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Your condition value includes a * or ? character. If you meant to use a wildcard (*, ?), update the condition operator to include Like."
```

解决一般警告

Condition 元素结构要求您使用条件运算符和键值对。当您指定使用通配符 (*, ?) 的条件值时，必须使用 Like 版本的条件运算符。例如，请使用 StringLike，而不是 StringEquals 字符串条件运算符。

```
"Condition": {"StringLike": {"aws:PrincipalTag/job-category": "admin-*"}}
```

- [IAM JSON 策略元素 : 条件运算符 \(p. 951\)](#)
- [IAM JSON 策略元素 : Condition \(p. 948\)](#)

带有此常规警告的 Amazon 托管策略

[Amazon 托管策略 \(p. 364\)](#)将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

以下 Amazon 托管策略在其条件值中包含通配符，而不带有包含用于模式匹配的 Like 的条件运算符。使用 Amazon 托管策略作为创建客户管理策略的参考时，Amazon 会建议您使用支持模式匹配且带有通配符 (*, ?) 的条件运算符，例如 StringLike。

- [AWSGlueConsoleSageMakerNotebookFullAccess](#)

常规警告 — 策略大小超出身份策略配额

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Policy size exceeds identity policy quota: The characters in the identity policy, excluding whitespace, exceed the character maximum for inline and managed policies. We recommend that you use multiple granular policies.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The characters in the identity policy, excluding whitespace, exceed the character maximum for inline and managed policies. We recommend that you use multiple granular policies."
```


解决一般警告

您可以将最多 10 个托管策略附加到 IAM 身份（用户、用户组或角色）。但是，每个托管策略的大小不能超过默认配额的 6144 个字符。在计算策略大小时，IAM 不会将空格计入该配额。配额，也称为 Amazon 中的限制，是 Amazon 账户中资源、操作和项目的最大值。

此外，您还可以向 IAM 身份添加所需数量的内联策略。但是，每个身份的所有内联策略的总大小不能超过指定的配额。

如果您的策略大于配额，您可以将策略组织为多个语句，并将这些语句分组为多个策略。

相关术语

- [IAM 和 Amazon STS STS 字符配额 \(p. 902\)](#)
- [多个声明和多个策略 \(p. 361\)](#)
- [IAM 客户托管策略 \(p. 365\)](#)
- [JSON 策略概述 \(p. 360\)](#)
- [IAM JSON 策略语法 \(p. 985\)](#)

带有此常规警告的 Amazon 托管策略

[Amazon 托管策略 \(p. 364\)](#)将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

以下 Amazon 托管策略向跨多个 Amazon 服务的操作授予权限，且超过了最大策略大小。使用 Amazon 托管策略作为参考来创建托管策略时，您必须将策略拆分为多个策略。

- [ReadOnlyAccess](#)
- [AWSSupportServiceRolePolicy](#)

常规警告 — 策略大小超出资源策略配额

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Policy size exceeds resource policy quota: The characters in the resource policy exceed the character maximum for resource policies. We recommend that you use multiple granular policies.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The characters in the resource policy exceed the character maximum for resource policies. We recommend that you use multiple granular policies."
```

解决一般警告

基于资源的策略是附加到资源（如 Amazon S3 存储桶）的 JSON 策略文档。这些策略授予指定的主体对该资源执行特定操作的权限，并定义这在哪些条件下适用。基于资源的策略的大小不能超过为该资源设置的配额。配额，也称为 Amazon 中的限制，是 Amazon 账户中资源、操作和项目的最大值。

如果您的策略大于配额，您可以将策略组织为多个语句，并将这些语句分组为多个策略。

相关术语

- [基于资源的策略 \(p. 356\)](#)
- [Amazon S3 存储桶策略](#)
- [多个声明和多个策略 \(p. 361\)](#)
- [JSON 策略概述 \(p. 360\)](#)

- [IAM JSON 策略语法 \(p. 985\)](#)

常规警告 — 类型不匹配

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Type mismatch: Use the operator type instead of operator for the condition key.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Use the operator type instead of operator for the condition key."
```

解决一般警告

请更新文本以使用支持的条件运算符数据类型。

例如，`aws:MultiFactorAuthPresent` 全局条件键需要带有 Boolean 数据类型的条件运算符。如果您提供日期或整数，则数据类型将不匹配。

相关术语

- [全局条件键 \(p. 998\)](#)
- [IAM JSON 策略元素：条件运算符 \(p. 951\)](#)

常规警告 — 类型不匹配布尔值

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Type mismatch Boolean: Add a valid Boolean value (true or false) for the condition operator.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a valid Boolean value (true or false) for the condition operator."
```

解决一般警告

请更新文本以使用布尔条件运算符数据类型，例如 `true` 或 `false`。

例如，`aws:MultiFactorAuthPresent` 全局条件键需要带有 Boolean 数据类型的条件运算符。如果您提供日期或整数，则数据类型将不匹配。

相关术语

- [布尔值条件运算符 \(p. 954\)](#)
- [IAM JSON 策略元素：条件运算符 \(p. 951\)](#)

常规警告 — 类型不匹配日期

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Type mismatch date: The date condition operator is used with an invalid value. Specify a valid date using YYYY-MM-DD or other ISO 8601 date/time format.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The date condition operator is used with an invalid value. Specify a valid date using YYYY-MM-DD or other ISO 8601 date/time format."
```

解决一般警告

请更新文本以使用日期条件运算符数据类型，采用 YYYY-MM-DD 或其他 ISO 8601 日期时间格式。

相关术语

- [日期条件运算符 \(p. 953\)](#)
- [IAM JSON 策略元素：条件运算符 \(p. 951\)](#)

常规警告 — 类型不匹配 IP 范围

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Type mismatch IP range: The condition operator is used with an invalid IP range value. Specify the IP range in standard CIDR format.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition operator is used with an invalid IP range value. Specify the IP range in standard CIDR format."
```

解决一般警告

请更新文本以使用 CIDR 格式的 IP 地址条件运算符数据类型。

相关术语

- [IP 地址条件运算符 \(p. 955\)](#)
- [IAM JSON 策略元素：条件运算符 \(p. 951\)](#)

常规警告 — 类型不匹配的数字

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Type mismatch number: Add a valid numeric value for the condition operator.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a valid numeric value for the condition operator."
```

解决一般警告

请更新文本以使用数字条件运算符数据类型。

相关术语

- [数字条件运算符 \(p. 953\)](#)
- [IAM JSON 策略元素：条件运算符 \(p. 951\)](#)

常规警告 — 类型不匹配的字符串

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Type mismatch string: Add a valid base64-encoded string value for the condition operator.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a valid base64-encoded string value for the condition operator."
```

解决一般警告

请更新文本以使用字符串条件运算符数据类型。

相关术语

- [字符串条件运算符 \(p. 951\)](#)
- [IAM JSON 策略元素：条件运算符 \(p. 951\)](#)

安全警告 — 允许使用 NotPrincipal

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Allow with NotPrincipal: Using Allow with NotPrincipal can be overly permissive. We recommend that you use Principal instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using Allow with NotPrincipal can be overly permissive. We recommend that you use Principal instead."
```

解决安全警告

使用 "Effect": "Allow" 与 NotPrincipal 可能会过于宽松。例如，这可以向匿名主体授予权限。Amazon 会建议您使用 Principal 元素指定需要访问权限的主体。或者，您可以允许广泛访问，然后添加另一个使用带有 "Effect": "Deny" 的 NotPrincipal 元素的语句。

- [Amazon JSON 策略元素：Principal \(p. 936\)](#)
- [Amazon JSON 策略元素：NotPrincipal \(p. 942\)](#)

安全警告 — 带有单值密钥的 ForAllValues

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
ForAllValues with single valued key: Using ForAllValues qualifier with the single-valued condition key can be overly permissive. We recommend that you remove ForAllValues:.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using ForAllValues qualifier with the single-valued condition key can be overly permissive. We recommend that you remove ForAllValues:."
```

解决安全警告

Amazon 会建议您仅将 `ForAllValues` 用于多值条件。`ForAllValues` 集合运算符测试请求集的每个成员的值是否为条件键集的子集。如果请求中的每个键值均与策略中的至少一个值匹配，则条件返回 `true`。如果请求中没有键或者键值解析为空数据集（如空字符串），则也会返回 `true`。

要了解条件是支持单个值还是多个值，请查看该服务的[操作](#)、[资源](#)和[条件密钥](#)页面。带有 `ArrayOf` 数据类型前缀的条件密钥是多值条件密钥。例如，Amazon SES 支持具有单个值 (`String`) 的密钥和 `ArrayOfString` 多值数据类型。

- [使用多个键和值 \(p. 960\)](#)

安全警告 — 使用 `NotResource` 传递角色

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Pass role with NotResource: Using the iam:PassRole action with NotResource can be overly permissive because it can allow iam:PassRole permissions on multiple resources. We recommend that you specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the iam:PassRole action with NotResource can be overly permissive because it can allow iam:PassRole permissions on multiple resources. We recommend that you specify resource ARNs instead."
```

解决安全警告

要配置多项 Amazon 服务，您必须将 IAM 角色传递给相应服务。要允许这样做，您必须对身份（用户、用户组或角色）授予 `iam:PassRole` 的权限。在带有 `NotResource` 元素的策略中使用 `iam:PassRole` 可以允许您的主体访问比您预期更多的服务或功能。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。此外，您可以通过使用 `iam:PassedToService` 条件键减少对单个服务的权限。

- [将角色传递给服务 \(p. 240\)](#)
- [iam:PassedToService \(p. 1015\)](#)
- [IAM JSON 策略元素：NotResource \(p. 948\)](#)
- [IAM JSON 策略元素：Resource \(p. 946\)](#)

安全警告 — 使用带有星号的操作和 `NotResource` 传递角色

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Pass role with star in action and NotResource: Using an action with a wildcard (*) and NotResource can be overly permissive because it can allow iam:PassRole permissions on multiple resources. We recommend that you specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using an action with a wildcard (*) and NotResource can be overly permissive because it can allow iam:PassRole permissions on multiple resources. We recommend that you specify resource ARNs instead."
```

解决安全警告

要配置多项 Amazon 服务，您必须将 IAM 角色传递给相应服务。要允许这样做，您必须对身份（用户、用户组或角色）授予 `iam:PassRole` 的权限。在 `Action` 中带有通配符 (*) 且包含 `NotResource` 元素的策

略可以允许您的主体访问比您预期更多的服务或功能。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。此外，您可以通过使用 `iam:PassedToService` 条件键减少对单个服务的权限。

- [将角色传递给服务 \(p. 240\)](#)
- [iam:PassedToService \(p. 1015\)](#)
- [IAM JSON 策略元素 : NotResource \(p. 948\)](#)
- [IAM JSON 策略元素 : Resource \(p. 946\)](#)

安全警告 — 使用 NotAction 和 NotResource 传递角色

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Pass role with NotAction and NotResource: Using NotAction with NotResource can be overly permissive because it can allow iam:PassRole permissions on multiple resources.. We recommend that you specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using NotAction with NotResource can be overly permissive because it can allow iam:PassRole permissions on multiple resources.. We recommend that you specify resource ARNs instead."
```

解决安全警告

要配置多项 Amazon 服务，您必须将 IAM 角色传递给相应服务。要允许这样做，您必须对身份（用户、用户组或角色）授予 `iam:PassRole` 的权限。使用 `NotAction` 元素并列出 `NotResource` 元素中的一些资源可以允许您的主体访问比您预期更多的服务或功能。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。此外，您可以通过使用 `iam:PassedToService` 条件键减少对单个服务的权限。

- [将角色传递给服务 \(p. 240\)](#)
- [iam:PassedToService \(p. 1015\)](#)
- [IAM JSON 策略元素 : NotAction \(p. 945\)](#)
- [IAM JSON 策略元素 : Action \(p. 944\)](#)
- [IAM JSON 策略元素 : NotResource \(p. 948\)](#)
- [IAM JSON 策略元素 : Resource \(p. 946\)](#)

安全警告 — 使用带有星号的资源传递角色

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Pass role with star in resource: Using the iam:PassRole action with wildcards (*) in the resource can be overly permissive because it allows iam:PassRole permissions on multiple resources. We recommend that you specify resource ARNs or add the iam:PassedToService condition key to your statement.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the iam:PassRole action with wildcards (*) in the resource can be overly permissive because it allows iam:PassRole permissions on multiple resources. We recommend that you specify resource ARNs or add the iam:PassedToService condition key to your statement."
```

解决安全警告

要配置多项 Amazon 服务，您必须将 IAM 角色传递给相应服务。要允许这样做，您必须对身份（用户、用户组或角色）授予 `iam:PassRole` 的权限。允许 `iam:PassRole` 且在 `Resource` 元素中包含通配符 (*) 的策略可以允许您的主体访问比您预期更多的服务或功能。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。此外，您可以通过使用 `iam:PassedToService` 条件键减少对单个服务的权限。

某些 Amazon 服务在其角色名称中包含服务命名空间。此策略检查会在分析策略以生成结果时考虑到这些约定。例如，以下资源 ARN 可能不会生成结果：

```
arn:aws:iam::*:role/Service*
```

- [将角色传递给服务 \(p. 240\)](#)
- [iam:PassedToService \(p. 1015\)](#)
- [IAM JSON 策略元素：Resource \(p. 946\)](#)

带有此安全警告的 Amazon 托管策略

[Amazon 托管策略 \(p. 364\)](#)将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

其中一个使用案例适用于您账户中的管理员。以下 Amazon 托管策略提供管理员访问权限，并授予将任何 IAM 角色传递给任何服务的权限。Amazon 会建议您仅将以下 Amazon 托管策略附加到您认为是管理员的 IAM 身份。

- [AdministratorAccess-Amplify](#)

以下 Amazon 托管策略包括对 `iam:PassRole` 的权限，并在资源包含通配符 (*)，且位于[弃用路径 \(p. 370\)](#)。对于这每一个策略，我们都更新了权限指南，例如推荐一个新的 Amazon 托管策略，以支持使用案例。要查看这些策略的替代方案，请参阅[每项服务 \(p. 908\)](#)的指南。

- `AWS ElasticBeanstalkFullAccess`
- `AWS ElasticBeanstalkService`
- `AWS LambdaFullAccess`
- `AWS LambdaReadOnlyAccess`
- `AWS OpsWorksFullAccess`
- `AWS OpsWorksRole`
- `AWS DataPipelineRole`
- `Amazon DynamoDBFullAccesswithDataPipeline`
- `Amazon ElasticMapReduceFullAccess`
- `Amazon DynamoDBFullAccesswithDataPipeline`
- `Amazon EC2 ContainerServiceFullAccess`

以下 Amazon 托管策略仅向[服务关联角色 \(p. 208\)](#)提供权限，它允许 Amazon 服务代表您执行操作。您可以将策略附加得到 IAM 身份。

- [AWSServiceRoleForAmazonEKSNodegroup](#)

安全警告 — 使用带有星号的行动和资源传递角色

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Pass role with star in action and resource: Using wildcards (*) in the action and the resource can be overly permissive because it allows iam:PassRole permissions on all
```



```
resources. We recommend that you specify resource ARNs or add the iam:PassedToService condition key to your statement.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using wildcards (*) in the action and the resource can be overly permissive because it allows iam:PassRole permissions on all resources. We recommend that you specify resource ARNs or add the iam:PassedToService condition key to your statement."
```

解决安全警告

要配置多项 Amazon 服务，您必须将 IAM 角色传递给相应服务。要允许这样做，您必须对身份（用户、用户组或角色）授予 iam:PassRole 的权限。在 Action 中带有通配符 (*) 且包含 Resource 元素的策略可以允许您的主体访问比您预期更多的服务或功能。Amazon 会建议您在 Resource 元素中指定允许的 ARN。此外，您可以通过使用 iam:PassedToService 条件键减少对单个服务的权限。

- [将角色传递给服务 \(p. 240\)](#)
- [iam:PassedToService \(p. 1015\)](#)
- [IAM JSON 策略元素：Action \(p. 944\)](#)
- [IAM JSON 策略元素：Resource \(p. 946\)](#)

带有此安全警告的 Amazon 托管策略

[Amazon 托管策略 \(p. 364\)](#)将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

其中一些使用案例适用于您账户中的管理员。以下 Amazon 托管策略提供管理员访问权限，并授予将任何 IAM 角色传递给任何 Amazon 服务的权限。Amazon 会建议您仅将以下 Amazon 托管策略附加到您认为是管理员的 IAM 身份。

- [AdministratorAccess](#)
- [IAMFullAccess](#)

安全警告 — 使用带有星号的资源和 NotAction 传递角色

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Pass role with star in resource and NotAction: Using a resource with wildcards (*) and NotAction can be overly permissive because it allows iam:PassRole permissions on all resources. We recommend that you specify resource ARNs or add the iam:PassedToService condition key to your statement.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using a resource with wildcards (*) and NotAction can be overly permissive because it allows iam:PassRole permissions on all resources. We recommend that you specify resource ARNs or add the iam:PassedToService condition key to your statement."
```

解决安全警告

要配置多项 Amazon 服务，您必须将 IAM 角色传递给相应服务。要允许这样做，您必须对身份（用户、用户组或角色）授予 iam:PassRole 的权限。在策略中使用 NotAction 元素且在 Resource 元素中包含通配符 (*) 可以允许您的主体访问比您预期更多的服务或功能。Amazon 会建议您在 Resource 元素中指定允许的 ARN。此外，您可以通过使用 iam:PassedToService 条件键减少对单个服务的权限。

- [将角色传递给服务 \(p. 240\)](#)

- [iam:PassedToService](#) (p. 1015)
- IAM JSON 策略元素 : [NotAction](#) (p. 945)
- IAM JSON 策略元素 : [Action](#) (p. 944)
- IAM JSON 策略元素 : [Resource](#) (p. 946)

安全警告 — 缺少配对条件键

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Missing paired condition keys: Using the condition key can be overly permissive without also using the following condition keys:. Condition keys like this one are more secure when paired with a related key. We recommend that you add the related condition keys to the same condition block.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the condition key can be overly permissive without also using the following condition keys:. Condition keys like this one are more secure when paired with a related key. We recommend that you add the related condition keys to the same condition block."
```

解决安全警告

当与其他相关条件键配对时，某些条件键会更加安全。Amazon 建议您将相关条件键包含在与现有条件键相同的条件数据块中。这会使得通过策略授予的权限更加安全。

例如，您可以使用 `aws:VpcSourceIp` 条件键将发出请求的 IP 地址与您在策略中指定的 IP 地址进行比较。Amazon 会建议您添加相关 `aws:SourceVPC` 条件键。这将检查请求是否来自您在策略中指定的 VPC 以及您指定的 IP 地址。

相关术语

- [aws:VpcSourceIp](#) 全局条件键 (p. 1013)
- [aws:SourceVPC](#) 全局条件键 (p. 1011)
- [全局条件键](#) (p. 998)
- [条件元素](#) (p. 948)
- [JSON 策略概述](#) (p. 360)

安全警告 — 使用不支持的标签条件键拒绝服务

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Deny with unsupported tag condition key for service: Using the effect Deny with the tag condition key and actions for services with the following prefixes can be overly permissive:. Actions for the listed services are not denied by this statement. We recommend that you move these actions to a different statement without this condition key.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the effect Deny with the tag condition key and actions for services with the following prefixes can be overly permissive:. Actions for the listed services are not denied by this statement. We recommend that you move these actions to a different statement without this condition key."
```

解决安全警告

在带有 "Effect": "Deny" 的政策的 Condition 元素中使用不受支持的标签条件键可能会过于宽松，因为该服务的条件将被忽略。Amazon 会建议您删除不支持条件键的服务操作，并创建另一个语句来拒绝对这些操作的特定资源的访问。

如果您使用 `aws:ResourceTag` 条件键，并且服务操作不支持该键，则该键不会包含在请求上下文中。在本例中，Deny 语句中的条件总是返回 `false`，并且该行动从来没有被拒绝。即使资源被正确标记，也会发生这种情况。

当服务支持 `aws:ResourceTag` 条件键时，您可以使用标签来控制对该服务资源的访问。这称为[基于属性的访问控制 \(ABAC\)](#) (p. 12)。不支持这些密钥的服务需要您使用[基于资源的访问控制 \(RBAC\)](#) (p. 12) 来控制对资源的访问。

Note

某些服务允许支持其资源和操作子集的 `aws:ResourceTag` 条件键。IAM Access Analyzer 将返回不受支持的服务操作结果。例如，Amazon S3 支持其资源子集的 `aws:ResourceTag`。要查看 Amazon S3 中支持 `aws:ResourceTag` 条件键的所有资源类型，请参阅服务授权参考中的[Amazon S3 定义的资源类型](#)。

例如，假设您想拒绝访问，以将被删除但使用了键值对 `status=Confidential` 进行标记的特定资源解除标记。同时假设 Amazon Lambda 允许您标记和取消标记资源，但不支持 `aws:ResourceTag` 条件键。要在此标签存在的情况下拒绝 Amazon App Mesh 和 Amazon Backup 的删除操作，请使用 `aws:ResourceTag` 条件键。对于 Lambda，请使用包含 "Confidential" 前缀的资源命名约定。然后包含一个单独的语句，以防止使用该命名约定删除资源。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyDeleteSupported",
      "Effect": "Deny",
      "Action": [
        "appmesh:DeleteMesh",
        "backup:DeleteBackupPlan"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/status": "Confidential"
        }
      }
    },
    {
      "Sid": "DenyDeleteUnsupported",
      "Effect": "Deny",
      "Action": "lambda:DeleteFunction",
      "Resource": "arn:aws:lambda:*:123456789012:function:status-Confidential*"
    }
  ]
}
```

Warning

请勿将[...IfExists](#) (p. 957) 版本的条件运算符用作此结果的解决方法。这表示“当键存在于请求上下文中且值匹配时，拒绝操作。否则，拒绝操作。”在前面的示例中，在带有 `StringEqualsIfExists` 运算符的 `DenyDeleteSupported` 语句中包含 `lambda:DeleteFunction` 操作将始终拒绝该操作。对于该操作，密钥不存在于上下文中，并且每次尝试删除该资源类型时都会被拒绝，无论资源是否被标记。

相关术语

- [全局条件键 \(p. 998\)](#)
- [比较 ABAC 与 RBAC \(p. 12\)](#)
- [IAM JSON 策略元素：条件运算符 \(p. 951\)](#)
- [条件元素 \(p. 948\)](#)
- [JSON 策略概述 \(p. 360\)](#)

安全警告 — 使用不支持的标签条件键拒绝服务的 NotAction

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Deny NotAction with unsupported tag condition key for service: Using the effect Deny with NotAction and the tag condition key can be overly permissive because some service actions are not denied by this statement. This is because the condition key doesn't apply to some service actions. We recommend that you use Action instead of NotAction.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the effect Deny with NotAction and the tag condition key can be overly permissive because some service actions are not denied by this statement. This is because the condition key doesn't apply to some service actions. We recommend that you use Action instead of NotAction."
```

解决安全警告

在带有元素 NotAction 和 "Effect": "Deny" 的策略的 Condition 元素中使用标签条件键可能会过于宽松。对于不支持条件键的服务操作，该条件将被忽略。Amazon 会建议您重写逻辑以拒绝操作列表。

如果您将 aws:ResourceTag 条件键与 NotAction 搭配使用，则不会拒绝任何不支持密钥的新或现有服务操作。Amazon 会建议您明确列出要拒绝的操作。IAM Access Analyzer 为列出的不支持 aws:ResourceTag 条件键的操作返回单独的结果。有关更多信息，请参阅[安全警告 — 使用不支持的标签条件键拒绝服务 \(p. 837\)](#)。

当服务支持 aws:ResourceTag 条件键时，您可以使用标签来控制对该服务资源的访问。这称为[基于属性的访问控制 \(ABAC\) \(p. 12\)](#)。不支持这些密钥的服务需要您使用[基于资源的访问控制 \(RBAC\) \(p. 12\)](#)来控制对资源的访问。

相关术语

- [全局条件键 \(p. 998\)](#)
- [比较 ABAC 与 RBAC \(p. 12\)](#)
- [IAM JSON 策略元素：条件运算符 \(p. 951\)](#)
- [条件元素 \(p. 948\)](#)
- [JSON 策略概述 \(p. 360\)](#)

安全警告 — 限制对服务主体的访问

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Restrict access to service principal: Granting access to a service principal without specifying a source is overly permissive. Use aws:SourceArn or aws:SourceAccount condition key and scope access to a specific source to grant fine-grained access.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Granting access to a service principal without specifying a source is overly permissive. Use aws:SourceArn or aws:SourceAccount condition key and scope access to a specific source to grant fine-grained access."
```

解决安全警告

您可以使用服务主体指定基于资源策略的 Principal 元素中的 Amazon Web Services，这是服务的标识符。授予服务主体的访问权限以代表您行事时，请限制访问权限。您可以通过 `aws:SourceAccount` 或 `aws:SourceArn` 密钥避免过于宽松的策略，例如特定资源 ARN 或 Amazon Web Services 账户。限制访问可以帮助您防止称为混淆代理问题的安全问题。

相关术语

- [服务主体](#)
- [Amazon 全局条件密钥](#)：`aws:SourceAccount`
- [Amazon 全局条件密钥](#)：`aws:SourceArn`
- [混淆代理人问题](#)

建议 — 空数组操作

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Empty array action: This statement includes no actions and does not affect the policy. Specify actions.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "This statement includes no actions and does not affect the policy. Specify actions."
```

解决建议

语句必须包含 Action 或者 NotAction 元素，其中包含一组操作。当元素为空时，策略语句不提供任何权限。在 Action 元素中指定操作。

- [IAM JSON 策略元素](#)：[Action \(p. 944\)](#)

建议 — 空数组条件

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Empty array condition: There are no values for the condition key and it does not affect the policy. Specify conditions.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "There are no values for the condition key and it does not affect the policy. Specify conditions."
```

解决建议

可选 Condition 元素结构要求您使用条件运算符和键值对。当条件值为空时，条件返回 true 并且策略语句不再提供任何权限。指定条件值。

- [IAM JSON 策略元素：Condition \(p. 948\)](#)

建议 — 空数组条件 ForAllValues

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Empty array condition ForAllValues: The ForAllValues prefix with an empty condition key matches only if the key is missing from the request context. To determine if the request context is empty, we recommend that you use the Null condition operator with the value of true instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The ForAllValues prefix with an empty condition key matches only if the key is missing from the request context. To determine if the request context is empty, we recommend that you use the Null condition operator with the value of true instead."
```

解决建议

Condition 元素结构要求您使用条件运算符和键值对。ForAllValues 集合运算符测试请求集的每个成员的值是否为条件键集的子集。

当您将 ForAllValues 用于空数组条件键时，则只有在请求中没有密钥时，条件才匹配。Amazon 会建议如果要测试请求上下文是否为空，请使用 Null 条件运算符。

- [使用多个键和值 \(p. 960\)](#)
- [Null 条件运算符 \(p. 958\)](#)
- [IAM JSON 策略元素：Condition \(p. 948\)](#)

建议 — 空数组条件 ForAnyValue

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Empty array condition ForAnyValue: The ForAnyValue prefix with an empty condition key never matches the request context and it does not affect the policy. Specify conditions.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The ForAnyValue prefix with an empty condition key never matches the request context and it does not affect the policy. Specify conditions."
```

解决建议

Condition 元素结构要求您使用条件运算符和键值对。ForAnyValues 集合运算符测试请求值集的至少一个成员是否与条件键值集的至少一个成员匹配。

当您使用带有空条件键的 ForAnyValues 时，条件永远不会匹配。这表示语句对策略没有影响。Amazon 会建议您重写条件。

- [使用多个键和值 \(p. 960\)](#)

- [IAM JSON 策略元素：Condition \(p. 948\)](#)

建议 — 空数组条件 IfExists

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Empty array condition IfExists: The IfExists suffix with an empty condition key matches only if the key is missing from the request context. To determine if the request context is empty, we recommend that you use the Null condition operator with the value of true instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The IfExists suffix with an empty condition key matches only if the key is missing from the request context. To determine if the request context is empty, we recommend that you use the Null condition operator with the value of true instead."
```

解决建议

...IfExists 后缀编辑条件运算符。这意味着如果请求的内容中存在策略键，则依照策略所述来处理键。如果该键不存在，则条件元素的计算结果将为 true。

当您将在 ...IfExists 用于空数组条件键时，则只有在请求中没有密钥时，条件才匹配。Amazon 会建议如果要测试请求上下文是否为空，请使用 Null 条件运算符。

- [...IfExists 条件运算符 \(p. 957\)](#)
- [IAM JSON 策略元素：Condition \(p. 948\)](#)

建议 — 空数组主体

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Empty array principal: This statement includes no principals and does not affect the policy. Specify principals.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "This statement includes no principals and does not affect the policy. Specify principals."
```

解决建议

您必在 IAM 角色的信任策略和基于资源的策略中使用 Principal 或 NotPrincipal 元素。基于资源的策略是直接嵌入资源中的策略。

当您在语句的 Principal 元素中提供空数组时，语句对策略没有影响。Amazon 会建议您指定应有权访问资源的主体。

- [IAM JSON 策略元素：Principal \(p. 936\)](#)
- [IAM JSON 策略元素：NotPrincipal \(p. 942\)](#)

建议 — 空数组资源

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

Empty array resource: This statement includes no resources and does not affect the policy. Specify resources.

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "This statement includes no resources and does not affect the policy. Specify resources."
```

解决建议

语句必须包含 Resource 或 NotResource 元素。

当您在语句的资源元素中提供空数组时，该语句对策略没有影响。Amazon 会建议您为资源指定 Amazon Resource Names (ARN)。

- [IAM JSON 策略元素：Resource \(p. 946\)](#)
- [IAM JSON 策略元素：NotResource \(p. 948\)](#)

建议 — 空对象条件

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

Empty object condition: This condition block is empty and it does not affect the policy. Specify conditions.

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "This condition block is empty and it does not affect the policy. Specify conditions."
```

解决建议

Condition 元素结构要求您使用条件运算符和键值对。

当您在语句的条件元素中提供空对象时，该语句对策略没有影响。请删除可选元素或指定条件。

- [IAM JSON 策略元素：Condition \(p. 948\)](#)

建议 — 空对象主体

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

Empty object principal: This statement includes no principals and does not affect the policy. Specify principals.

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "This statement includes no principals and does not affect the policy. Specify principals."
```

解决建议

您必在 IAM 角色的信任策略和基于资源的策略中使用 Principal 或 NotPrincipal 元素。基于资源的策略是直接嵌入资源中的策略。

当您在语句的 `Principal` 元素中提供空对象时，语句对策略没有影响。Amazon 会建议您指定应有权访问资源的主体。

- [IAM JSON 策略元素：Principal \(p. 936\)](#)
- [IAM JSON 策略元素：NotPrincipal \(p. 942\)](#)

建议 — 空 Sid 值

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Empty Sid value: Add a value to the empty string in the Sid element.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a value to the empty string in the Sid element."
```

解决建议

`Sid` (声明 ID) 元素允许您针对策略语句输入标识符。您可以为语句数组中的每个语句指定 `Sid` 值。如果您选择使用 `Sid` 元素，则您必须提供字符串值。

相关术语

- [IAM JSON 策略元素：Sid \(p. 936\)](#)

建议 — 改善 IP 范围

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Improve IP range: The non-zero bits in the IP address after the masked bits are ignored.  
Replace address with.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The non-zero bits in the IP address after the masked bits are ignored.  
Replace address with."
```

解决建议

IP 地址条件必须采用标准的 CIDR 格式，例如 203.0.113.0/24 或 2001:DB8:1234:5678::/64。如果在屏蔽位之后包含非零位，则不考虑这些位作为条件。Amazon 会建议您使用消息中包含的新地址。

- [IP 地址条件运算符 \(p. 955\)](#)
- [IAM JSON 策略元素：Condition \(p. 948\)](#)

建议 — 带限定符的空值

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Null with qualifier: Avoid using the Null condition operator with the ForAllValues or  
ForAnyValue qualifiers because they always return a true or false respectively.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Avoid using the Null condition operator with the ForAllValues or ForAnyValue qualifiers because they always return a true or false respectively."
```

解决建议

在 Condition 元素中，您可构建表达式并使用条件运算符（等于、小于等）将策略中的条件键和值与请求上下文中的键和值进行匹配。对单个条件键包含多个值的请求而言，必须使用 ForAllValues 或 ForAnyValue 集合运算符。

当您使用带有 ForAllValues 的 Null 条件运算符时，则该语句将始终返回 true。当您使用带有 ForAnyValue 的 Null 条件运算符时，则该语句始终返回 false。Amazon 会建议您将 StringLike 条件运算符与这些集合运算符搭配使用。

相关术语

- [使用多种条件键和值 \(p. 960\)](#)
- [Null 条件运算符 \(p. 958\)](#)
- [条件元素 \(p. 948\)](#)

建议 — 私有 IP 地址子集

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Private IP address subset: The values for condition key aws:SourceIp include a mix of private and public IP addresses. The private addresses will not have the desired effect. aws:SourceIp works only for public IP address ranges. To define permissions for private IP ranges, use aws:VpcSourceIp.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The values for condition key aws:SourceIp include a mix of private and public IP addresses. The private addresses will not have the desired effect. aws:SourceIp works only for public IP address ranges. To define permissions for private IP ranges, use aws:VpcSourceIp."
```

解决建议

全局条件键 aws:SourceIp 仅适用于公有 IP 地址范围。

当您的 Condition 元素同时包含私有 IP 地址和公有 IP 地址时，语句可能不会产生所需的效果。您不能使用 aws:VpcSourceIP 指定私有 IP 地址。

Note

全局条件键 aws:VpcSourceIP 仅在请求来自指定的 IP 地址并经过 VPC 终端节点时进行匹配。

- [aws:SourceIp 全局条件键 \(p. 1010\)](#)
- [aws:VpcSourceIp 全局条件键 \(p. 1013\)](#)
- [IP 地址条件运算符 \(p. 955\)](#)
- [IAM JSON 策略元素：Condition \(p. 948\)](#)

建议 — 私有 NotIpAddress 子集

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

Private NotIpAddress subset: The values for condition key aws:SourceIp include a mix of private and public IP addresses. The private addresses have no effect. aws:SourceIp works only for public IP address ranges. To define permissions for private IP ranges, use aws:VpcSourceIp.

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The values for condition key aws:SourceIp include a mix of private and public IP addresses. The private addresses have no effect. aws:SourceIp works only for public IP address ranges. To define permissions for private IP ranges, use aws:VpcSourceIp."
```

解决建议

全局条件键 aws:SourceIp 仅适用于公有 IP 地址范围。

当您的 Condition 元素包含 NotIpAddress 条件运算符以及同时包括私有 IP 地址和公有 IP 地址的时，语句可能不会产生所需的效果。策略中未指定的每个公有 IP 地址都将匹配。任何私有 IP 地址都不会匹配。为了达到这个效果，您可以使用带有 aws:VpcSourceIp 的 NotIpAddress 并指定不应匹配的私有 IP 地址。

- [aws:SourceIp 全局条件键 \(p. 1010\)](#)
- [aws:VpcSourceIp 全局条件键 \(p. 1013\)](#)
- [IP 地址条件运算符 \(p. 955\)](#)
- [IAM JSON 策略元素：Condition \(p. 948\)](#)

建议 — 冗余操作

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Redundant action: The action(s) are redundant because they provide similar permissions. Update the policy to remove the redundant action such as:.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The action(s) are redundant because they provide similar permissions. Update the policy to remove the redundant action such as:."
```

解决建议

当在 Action 元素中使用通配符 (*) 时，您可以包含冗余权限。Amazon 会建议您查看策略并仅包含所需权限。这可以帮助您删除冗余操作。

例如，以下操作包括两次 iam:GetCredentialReport 操作。

```
"Action": [  
    "iam:Get*",  
    "iam:List*",  
    "iam:GetCredentialReport"  
],
```

在此示例中，为以 Get 或 List 开头的每个 IAM 操作均定义了权限。当 IAM 添加额外的获取或列示操作时，此策略将允许这些操作。您可能希望允许所有这些只读操作。iam:GetCredentialReport 操作已作为 iam:Get* 的一部分纳入。要删除重复的权限，您可以删除 iam:GetCredentialReport。

当操作的所有内容都是冗余时，您会收到此策略检查的结果。在此示例中，如果元素包含 `iam:*CredentialReport`，则不被认为是冗余的。这包括冗余的 `iam:GetCredentialReport` 和非冗余的 `iam:GenerateCredentialReport`。删除 `iam:Get*` 或 `iam:*CredentialReport` 将更改策略的权限。

- [IAM JSON 策略元素：Action \(p. 944\)](#)

带有此建议的 Amazon 托管策略

[Amazon 托管策略 \(p. 364\)](#)将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

冗余操作不影响策略所授予的权限。使用 Amazon 托管策略作为创建客户托管策略的参考时，Amazon 会建议您从策略中删除冗余操作。

建议 — 冗余条件值编号

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Redundant condition value num: Multiple values in are redundant. Replace with the greatest/least single value for.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Multiple values in are redundant. Replace with the greatest/least single value for."
```

解决建议

对条件键中的类似值使用数字条件运算符时，您可以创建重叠，从而会导致冗余权限。

例如，以下 Condition 元素包含多个具有 1200 秒年龄重叠的 `aws:MultiFactorAuthAge` 条件。

```
"Condition": {
  "NumericLessThan": {
    "aws:MultiFactorAuthAge": [
      "2700",
      "3600"
    ]
  }
}
```

在此示例中，如果在不到 3600 秒（1 小时）之前完成多重身份验证 (MFA)，则定义权限。您可以删除冗余 2700 值。

- [数字条件运算符 \(p. 953\)](#)
- [IAM JSON 策略元素：Condition \(p. 948\)](#)

建议 — 冗余资源

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Redundant resource: The resource ARN(s) are redundant because they reference the same resource. Review the use of wildcards (*)
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The resource ARN(s) are redundant because they reference the same resource. Review the use of wildcards (*)"
```

解决建议

当您在 Amazon Resource Names (ARN) 中使用通配符 (*) 时，您可以创建冗余资源权限。

例如，以下 Resource 元素包含多个具有冗余权限的 ARN。

```
"Resource": [  
  "arn:aws:iam::111122223333:role/jane-admin",  
  "arn:aws:iam::111122223333:role/jane-s3only",  
  "arn:aws:iam::111122223333:role/jane*",  
],
```

在此示例中，为名称以 jane 开头的任何角色定义了权限。您可以删除冗余 jane-admin 和 jane-s3only ARN，而不更改生成的权限。这确实会使策略变得动态化。它将为未来的任何以 jane 开头的角色定义权限。如果策略的目的是允许访问静态数量的角色，则删除最后一个 ARN 并仅列出应定义的 ARN。

- [IAM JSON 策略元素：Resource \(p. 946\)](#)

带有此建议的 Amazon 托管策略

[Amazon 托管策略 \(p. 364\)](#)将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

冗余资源不影响策略所授予的权限。使用 Amazon 托管策略作为创建客户托管策略的参考时，Amazon 会建议您从策略中删除冗余资源。

建议 — 冗余语句

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Redundant statement: The statements are redundant because they provide identical permissions. Update the policy to remove the redundant statement.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The statements are redundant because they provide identical permissions. Update the policy to remove the redundant statement."
```

解决建议

Statement 元素为策略的主要元素。该元素为必填项。Statement 元素可以包含一条语句或由单独语句组成的数组。

如果在长策略中多次包含相同的语句，则为冗余语句。您可以删除其中一个语句，不会影响策略授予的权限。当某人编辑策略时，他们可能会更改其中一个语句而不更新副本。这可能会导致超出预期的权限。

- [IAM JSON 策略元素：Statement \(p. 935\)](#)

建议 — 服务名称中的通配符

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

Wildcard in service name: Avoid using wildcards (*, ?) in the service name because it might grant unintended access to other AWS services with similar names.

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Avoid using wildcards (*, ?) in the service name because it might grant unintended access to other AWS services with similar names."
```

解决建议

当您在策略中包含 Amazon 服务的名称时，Amazon 建议您不要包括通配符 (*, ?)。这可能会为您意料之外的未来服务添加权限。例如，有数十个 Amazon 服务在其名称中包含 *code*。

```
"Resource": "arn:aws:*code*::111122223333:*"
```

- [IAM JSON 策略元素：Resource \(p. 946\)](#)

建议 — 使用不支持的标签条件键允许服务

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

Allow with unsupported tag condition key for service: Using the effect Allow with the tag condition key and actions for services with the following prefixes does not affect the policy:. Actions for the listed service are not allowed by this statement. We recommend that you move these actions to a different statement without this condition key.

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the effect Allow with the tag condition key and actions for services with the following prefixes does not affect the policy:. Actions for the listed service are not allowed by this statement. We recommend that you move these actions to a different statement without this condition key."
```

解决建议

在带有 "Effect": "Allow" 的策略的 Condition 元素中使用不受支持的标签条件键不会影响策略授予的权限，因为该服务操作将忽略该条件。Amazon 会建议您删除将不支持条件键的服务的操作删除，并创建另一个语句以允许访问该服务中的特定资源。

如果您使用 `aws:ResourceTag` 条件键，并且服务操作不支持该键，则该键不会包含在请求上下文中。在本例中，Allow 语句中的条件总是返回 false 并且该操作将永远不被允许。即使资源被正确标记，也会发生这种情况。

当服务支持 `aws:ResourceTag` 条件键时，您可以使用标签来控制对该服务资源的访问。这称为[基于属性的访问控制 \(ABAC\) \(p. 12\)](#)。不支持这些密钥的服务需要您使用[基于资源的访问控制 \(RBAC\) \(p. 12\)](#)来控制对资源的访问。

Note

某些服务允许支持其资源和操作子集的 `aws:ResourceTag` 条件键。IAM Access Analyzer 将返回不受支持的服务操作结果。例如，Amazon S3 支持其资源子集的 `aws:ResourceTag`。要查看 Amazon S3 中支持 `aws:ResourceTag` 条件键的所有资源类型，请参阅服务授权参考中的[Amazon S3 定义的资源类型](#)。

例如，假设您希望允许团队成员查看使用键值对 `team=BumbleBee` 标记的特定资源的详细信息。同时假设 Amazon Lambda 允许您标记资源，但不支持 `aws:ResourceTag` 条件键。要在此标签存在的情况下允许查

看 Amazon App Mesh 和 Amazon Backup 的操作，请使用 `aws:ResourceTag` 条件键。对于 Lambda，请使用资源命名约定，其中包含团队名称作为前缀。然后包含一个单独的语句，以允许查看具有该命名约定的资源。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowViewSupported",
      "Effect": "Allow",
      "Action": [
        "appmesh:DescribeMesh",
        "backup:GetBackupPlan"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/team": "BumbleBee"
        }
      }
    },
    {
      "Sid": "AllowViewUnsupported",
      "Effect": "Allow",
      "Action": "lambda:GetFunction",
      "Resource": "arn:aws:lambda:*:123456789012:function:team-BumbleBee*"
    }
  ]
}
```

Warning

请勿将带有 "Effect": "Allow" 的 Not [版本条件运算符 \(p. 951\)](#) 用为此结果的解决方法。这些条件运算符提供否定匹配。这意味着在评估条件后，结果将被否定。在前面的示例中，在带有 StringNotEquals 运算符的 AllowViewSupported 语句中包含 lambda:GetFunction 操作将始终允许操作，无论资源是否被标记。

请勿将...[IfExists \(p. 957\)](#) 版本的条件运算符用作此结果的解决方法。这表示“当键存在于请求上下文中且值匹配时，允许操作。否则，允许操作。”在前面的示例中，在带有 StringEqualsIfExists 运算符的 AllowViewSupported 语句中包含 lambda:GetFunction 操作将始终允许该操作。对于该操作，密钥不存在于上下文中，并且每次尝试查看该资源类型时都会被允许，无论资源是否被标记。

相关术语

- [全局条件键 \(p. 998\)](#)
- [IAM JSON 策略元素：条件运算符 \(p. 951\)](#)
- [条件元素 \(p. 948\)](#)
- [JSON 策略概述 \(p. 360\)](#)

建议 — 使用不支持的标签条件键允许服务的 NotAction

在 Amazon Web Services Management Console 中，则此检查的结果包括以下消息：

```
Allow NotAction with unsupported tag condition key for service: Using the effect Allow with NotAction and the tag condition key allows only service actions that support the condition key. The condition key doesn't apply to some service actions. We recommend that you use Action instead of NotAction.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the effect Allow with NotAction and the tag condition key allows only service actions that support the condition key. The condition key doesn't apply to some service actions. We recommend that you use Action instead of NotAction."
```

解决建议

在带有 NotAction 和 "Effect": "Allow" 元素的策略的 Condition 元素使用不受支持的标签条件键不会影响策略授予的权限。对于不支持条件键的服务操作，该条件将被忽略。Amazon 会建议您重写逻辑以允许操作列表。

如果您将 aws:ResourceTag 条件键与 NotAction 搭配使用，则不会允许任何不支持密钥的新或现有服务操作。Amazon 会建议您明确列出要允许的操作。IAM Access Analyzer 为列出的不支持 aws:ResourceTag 条件键的操作返回单独的结果。有关更多信息，请参阅[建议 — 使用不支持的标签条件键允许服务](#) (p. 849)。

当服务支持 aws:ResourceTag 条件键时，您可以使用标签来控制对该服务资源的访问。这称为[基于属性的访问控制 \(ABAC\)](#) (p. 12)。不支持这些密钥的服务需要您使用[基于资源的访问控制 \(RBAC\)](#) (p. 12) 来控制对资源的访问。

相关术语

- [全局条件键](#) (p. 998)
- [比较 ABAC 与 RBAC](#) (p. 12)
- [IAM JSON 策略元素：条件运算符](#) (p. 951)
- [条件元素](#) (p. 948)
- [JSON 策略概述](#) (p. 360)

建议 — 策略中的条件密钥无关

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

```
Irrelevant condition key in policy: The condition key is not relevant for the policy. Use this key in an identity-based policy to govern access to this resource.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key is not relevant for the policy. Use this key in an identity-based policy to govern access to this resource."
```

解决建议

与基于资源的策略不相关的条件密钥。例如，s3:ResourceAccount 条件密钥与附加到 Amazon S3 存储桶或 Amazon S3 访问点资源类型的基于资源的策略无关。

您可以在基于身份的策略中使用条件，以便控制对资源的访问。

相关术语

- [基于身份的策略和基于资源的策略](#) (p. 379)

建议 — 服务主体的推荐条件密钥

在 Amazon Web Services Management Console 中，此检查的结果包括以下消息：

Recommended condition key for service principal: To restrict access to the service principal operating on your behalf, we recommend `aws:SourceArn` or `aws:SourceAccount` instead of.

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "To restrict access to the service principal operating on your behalf, we recommend aws:SourceArn or aws:SourceAccount instead of."
```

解决建议

您可以使用服务主体指定基于资源策略的 `Principal` 元素中的 Amazon Web Services，这是服务的标识符。授予服务主体访问权限时，您应该使用 `aws:SourceAccount` 或 `aws:SourceArn` 条件密钥，而不是其他条件密钥，如 `aws:Referer`。这样可以帮助您防止称为混淆代理问题的安全问题。

相关术语

- [服务主体](#)
- [Amazon 全局条件密钥：aws:SourceAccount](#)
- [Amazon 全局条件密钥：aws:SourceArn](#)
- [混淆代理人问题](#)

IAM Access Analyzer 策略生成

作为管理员或开发人员，您可以向 IAM 实体（用户或角色）授予超出其需要的权限。IAM 提供了多个选项来帮助您优化授予的权限。其中一种选择是根据实体的访问活动生成 IAM policy。IAM 访问分析器会查看您的 Amazon CloudTrail 日志并生成一个策略模板，其中包含角色在指定日期范围内使用的权限。您可以使用模板创建具有精细权限的策略，该策略仅授予支持特定使用案例所需的权限。

主题

- [策略生成的工作原理 \(p. 852\)](#)
- [服务和操作级别的信息 \(p. 853\)](#)
- [生成策略需知信息 \(p. 854\)](#)
- [生成策略所需的权限 \(p. 855\)](#)
- [基于 CloudTrail 活动生成策略（控制台）\(p. 857\)](#)
- [使用其他账户中的 Amazon CloudTrail 数据生成策略 \(p. 859\)](#)
- [基于 CloudTrail 活动生成策略（Amazon CLI）\(p. 861\)](#)
- [基于 CloudTrail 活动生成策略（Amazon API）\(p. 861\)](#)

策略生成的工作原理

IAM Access Analyzer 分析 CloudTrail 事件以识别 IAM 实体（用户或角色）已使用的操作和服务。然后，它会生成基于该活动的 IAM policy。将附加到实体的广泛权限策略替换为生成的策略时，来可以优化实体的权限。以下是策略生成过程的高度概述。

- **设置策略模板生成** – 您可以为 IAM Access Analyzer 指定一个最多 90 天的时间段来分析历史 Amazon CloudTrail CloudTrail 事件。您必须指定一个现有服务角色或创建一个新的服务角色。服务角色允许 IAM Access Analyzer 访问您的 CloudTrail 跟踪和服务上次访问的信息，以识别所使用的服务和操作。在生成策略之前，必须指定记录账户事件的 CloudTrail 跟踪。有关 CloudTrail 数据 IAM Access Analyzer 配额的信息，请参阅 [IAM Access Analyzer 配额](#)。

- 生成策略 – IAM Access Analyzer 会根据 CloudTrail 事件中的访问活动生成策略。
- 查看和自定义策略 – 生成策略后，您可以查看实体在指定日期范围内使用的服务和操作。您可以通过添加或删除权限、指定资源以及向策略模板添加条件来进一步自定义策略。
- 创建和附加策略 – 您可以通过创建托管策略来保存生成的策略。您可以将创建的策略附加到其活动用于生成策略的用户或角色。

服务和操作级别的信息

IAM Access Analyzer 生成 IAM policy 时，会返回信息来帮助您进一步自定义策略。生成策略时可以返回两类信息：

- 包含操作级别信息的策略 – 对于某些 Amazon 服务（例如 Amazon EC2），IAM Access Analyzer 可以识别在 CloudTrail 事件中发现的操作，并列出生成的策略中所使用的操作。对于某些服务，我们会提示您将服务的操作添加到生成的策略中。
- 包含服务级别信息的策略 – IAM Access Analyzer 使用[上次访问](#)的信息创建策略模板，其中包含最近使用过的所有服务。使用 Amazon Web Services Management Console 时，我们会提示您查看服务并添加操作以完成策略。

IAM Access Analyzer 为以下 Amazon 服务生成包含操作级别信息的策略：有关每项服务中的操作列表，请参阅《服务授权参考》中的[Amazon 服务的操作、资源和条件键](#)。

操作级别支持服务的信息

- IAM Access Analyzer
- Amazon Amplify
- Amazon AppIntegrations
- Amazon AppFlow
- Amazon Application Cost Profiler
- Amazon Athena
- Amazon Batch
- Amazon Braket
- Amazon CloudTrail
- Amazon CloudWatch
- Amazon CloudWatch Logs
- Amazon CloudWatch Synthetics
- Amazon CodeGuru Profiler
- Amazon Cognito Identity
- Amazon Cognito Sync
- Amazon Cognito 用户池
- Amazon Cost and Usage Report
- Amazon DevOps Guru
- Device Advisor
- Amazon Directory Service
- Amazon Elastic Block Store
- Amazon EC2
- Amazon Elastic Container Registry
- Amazon ECS
- 弹性负载均衡

- Amazon EventBridge Schemas
- Fleet Hub for Amazon IoT Device Management
- Amazon Ground Station
- Amazon GuardDuty
- IAM
- Amazon Interactive Video Service
- Amazon KMS
- Amazon Kinesis Data Firehose
- Amazon Lambda
- Amazon MQ
- Amazon Managed Blockchain
- Amazon Managed Streaming for Apache Kafka
- Amazon Web Services Marketplace
- Amazon Nimble Studio
- Amazon OpsWorks
- Amazon Outposts
- Amazon Performance Insights
- Amazon RDS
- Amazon RAM
- Amazon Resource Groups
- Resource Groups 标记 API
- Savings Plans
- Amazon Secrets Manager
- Amazon Security Hub
- Amazon Security Token Service
- Amazon Server Migration Service
- Service Quotas
- Amazon Signer
- Amazon Simple Email Service
- Amazon Simple Storage Service
- Amazon Systems Manager
- Amazon Textract
- Amazon Translate
- Amazon Well-Architected Tool
- Amazon WorkLink

生成策略需知信息

在生成策略之前，请查看以下重要详细信息。

- 启用 CloudTrail 跟踪 – 您必须为账户启用 CloudTrail 跟踪，才能基于访问活动生成策略。创建 CloudTrail 跟踪时，CloudTrail 会将与跟踪相关的事件发送到您指定的 Amazon S3 存储桶。要了解如何创建 CloudTrail 跟踪，请参阅《Amazon CloudTrail 用户指南》中的[为您的 Amazon 账户创建跟踪](#)。
- 数据事件不可用 — IAM Access Analyzer 不会在生成的策略中识别数据事件（例如 Amazon S3 数据事件）的操作级别活动。
- PassRole – CloudTrail 不会跟踪 `iam:PassRole` 操作，生成的策略中也不包含该操作。

- 缩短策略生成时间 – 要加快生成策略的速度，请缩短在设置策略生成时指定的日期范围。
- 使用 CloudTrail 进行审计 – 请勿将策略生成用于审计目的；而是使用 CloudTrail 进行审计。有关使用 CloudTrail 的更多信息，请参阅[使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用](#)。
- 一个策略 IAM 控制台 – 您可以在 IAM 控制台中一次生成一个策略。
- 所生成策略的可用性 IAM 控制台 – 您可以在生成策略后的 7 天内 IAM 控制台中查看生成的策略。7 天后，您必须生成新的策略。
- 策略生成配额 – 有关 IAM Access Analyzer 策略生成配额的更多信息，请参阅[IAM Access Analyzer 配额](#)。
- 适用 Amazon S3 Standard 费率 - 当您使用策略生成功能时，IAM 访问分析器会审核 S3 存储桶中的 CloudTrail 日志。访问您的 CloudTrail 日志以生成策略不会产生额外的存储费用。Amazon 对存储在 S3 存储桶中的 CloudTrail 日志的请求和数据传输收取标准 Amazon S3 费率。

生成策略所需的权限

首次生成策略所需的权限与为后续使用生成策略所需的权限不同。

首次设置

首次生成策略时，必须在账户中选择一个合适的现有[服务角色](#)或创建一个新的服务角色。服务角色允许 IAM Access Analyzer 访问账户中的 CloudTrail 以及服务上次访问的信息。仅管理员有权创建和配置角色。因此，我们建议管理员在首次设置时创建服务角色。要了解有关创建服务角色所需权限的更多信息，请参阅[创建角色以向 Amazon 服务委派权限](#)。

创建服务角色所需的权限

创建服务角色时，您需要为该角色配置两个策略。您可以将 IAM 权限策略附加到角色，它指定角色可以执行的操作。您还可以将角色信任策略附加到角色，它指定可以使用该角色的主体。

第一个示例策略显示了生成策略所需的服务角色的权限策略。第二个示例策略显示了服务角色所需的角色信任策略。在使用 Amazon API 或 Amazon CLI 生成策略时，可以使用这些策略帮助创建服务角色。在策略生成过程中使用 IAM 控制台创建服务角色时，我们会为您生成这些策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloudtrail:GetTrail",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:GenerateServiceLastAccessedDetails",
        "iam:GetServiceLastAccessedDetails"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ]
    }
  ]
}
```

```
    }  
  ]  
}
```

以下示例策略显示的角色信任策略具有允许 IAM Access Analyzer 担任角色的权限。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "Service": "access-analyzer.amazonaws.com"  
      },  
      "Action": "sts:AssumeRole"  
    }  
  ]  
}
```

后续使用

要在 Amazon Web Services Management Console 中生成策略，IAM 用户必须拥有允许他们将用于生成策略的服务角色传递给 IAM Access Analyzer 的权限策略。iam:PassRole 通常与 iam:GetRole 共同使用，使用户能够获取准备进行传递的角色的详细信息。在此示例中，用户只能传递位于指定账户中并且名称以 AccessAnalyzerMonitorServiceRole* 开头的角色。要了解将 IAM 角色传递给 Amazon 服务的更多信息，请参阅[向用户授予权限以将角色传递给 Amazon 服务](#)。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowUserToPassRole",  
      "Effect": "Allow",  
      "Action": [  
        "iam:GetRole",  
        "iam:PassRole"  
      ],  
      "Resource": "arn:aws:iam::123456789012:role/service-role/  
AccessAnalyzerMonitorServiceRole*"  
    }  
  ]  
}
```

您还必须具有以下 IAM Access Analyzer 权限才能在 Amazon Web Services Management Console、Amazon API 中或 Amazon CLI 中生成策略，如下策略语句所示。

```
{  
  "Sid": "AllowUserToGeneratePolicy",  
  "Effect": "Allow",  
  "Action": [  
    "access-analyzer:ListPolicyGenerations",  
    "access-analyzer:StartPolicyGeneration",  
    "access-analyzer:GetGeneratedPolicy",  
    "access-analyzer:CancelPolicyGeneration"  
  ],  
  "Resource": "*"   
}
```

首次使用和后续使用

使用 Amazon Web Services Management Console 生成策略时，必须授予 `cloudtrail:ListTrails` 在账户中列出 CloudTrail 跟踪的权限，如以下策略语句所示。

```
{
  "Sid": "AllowUserToListTrails",
  "Effect": "Allow",
  "Action": [
    "CloudTrail:ListTrails"
  ],
  "Resource": "*"
}
```

基于 CloudTrail 活动生成策略 (控制台)

您可以为 IAM 用户或角色生成策略。

步骤 1：基于 CloudTrail 活动生成策略

以下过程介绍了如何使用 Amazon Web Services Management Console 为角色生成策略。

为 IAM 角色生成策略

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在左侧的导航窗格中，选择角色。

Note

基于 IAM 用户的活动生成策略的步骤几乎完全相同。要执行此操作，请选择 Users (用户)，不要选择 Roles (角色)。

3. 在账户的角色列表中，选择要将其活动用于生成策略的角色的名称。
4. 在 Permissions (权限) 选项卡的基于 CloudTrail 事件生成策略部分，选择 Generate policy (生成策略)。
5. 在 Generate policy (生成策略) 页面，指定希望 IAM Access Analyzer 分析 CloudTrail 事件 (以角色采取操作) 的时间段。最多可以选择 90 天。我们建议您选择尽可能短的时间段以缩短策略生成时间。
6. 在 CloudTrail 访问部分中，选择合适的现有角色或创建新角色 (如果不存在合适角色)。此角色授予 IAM Access Analyzer 代表您访问 CloudTrail 数据的权限，以查看访问活动并识别已使用的服务和操作。要了解该角色所需的权限，请参阅 [生成策略所需的权限 \(p. 855\)](#)。
7. 在要分析的 CloudTrail 跟踪部分，指定记录账户事件的 CloudTrail 跟踪。

如果您选择将日志存储在其他账户中的 CloudTrail 跟踪记录，则会显示一个有关跨账户访问的信息框。跨账户访问需要额外的设置。要了解更多信息，请稍后参阅本主题中的 [Choose a role for cross-account access](#)。

8. 选择 Generate Policy (生成策略)。
9. 在策略生成过程中，您将返回到 Permissions (权限) 选项卡上的 Roles (角色) Summary (摘要) 页面。等到 Policy request details (策略请求详细信息) 部分中的状态显示为 Success (成功) 后，选择 View generated policy (查看生成的策略)。您可以在策略生成后的七天内进行查看。如果生成另一个策略，则会将现有策略替换为生成的新策略。

步骤 2：查看权限并为所使用的服务添加操作

查看 IAM Access Analyzer 识别到的角色所使用的服务和操作。您可以为生成的策略模板添加用于任何服务的操作。

1. 查看以下部分：

- 在 Review permissions (查看权限) 页面，查看 Actions included in the generated policy (生成的策略中包含的操作) 列表。该列表显示了角色在指定日期范围内使用的 IAM Access Analyzer 已识别到的服务和操作。
 - Services used (使用的服务) 部分显示了角色在指定日期范围内使用的 IAM Access Analyzer 已识别到的其他服务。有关使用了哪些操作的信息可能不适用于本部分中列出的服务。使用列出的各项服务的菜单手动选择您希望在策略中包含的操作。
- 添加完操作后，选择 Next (下一步)。

步骤 3：进一步自定义生成的策略

您可以通过添加或删除权限或指定资源来进一步自定义策略。

自定义生成的策略

- 请更新策略模板。策略模板包含支持资源级权限的操作的资源 ARN 占位符，如下图所示。资源级权限指的是能够指定允许用户对哪些资源执行操作的能力。我们建议您使用 [ARN](#) 在策略中为支持资源级权限的操作指定个人资源。您可以将占位符资源 ARN 替换为使用案例的有效资源 ARN。

如果操作不支持资源级权限，则必须使用通配符 (*) 来指定所有资源都可能受到操作的影响。要了解哪些 Amazon 服务支持资源级权限，请参阅 [Amazon services that work with IAM](#) (能够与 IAM 搭配使用的亚马逊云科技服务)。有关各项服务中的操作列表，以及要了解哪些操作支持资源级权限，请参阅 [Amazon 服务的操作、资源和条件键](#)。

Generated policy

1 2 3

Customize permissions

Review the following policy template. You must specify resources for actions that support resource-level permissions to continue creating the policy.

The screenshot shows the 'Generated policy' section of the IAM console. It displays a JSON policy template with several statements. The 'Resource' field for each statement is highlighted with a blue box, showing the placeholder 'arn:aws:iam::\${Account}:role/\${RoleNameWithPath}'. On the right side, there is an 'Edit statement' panel with a 'Select a statement' dropdown and an 'Add new statement' button.

- (可选) 在模板中添加、修改或删除 JSON 策略语句。要了解有关编写 JSON 策略的更多信息，请参阅 [创建 IAM policy \(控制台 \)](#)。
- 自定义策略模板后，可以选择以下选项：
 - (可选) 您可以复制模板中的 JSON，以便在 Generated policy (生成的策略) 页面之外单独使用。例如，您想要使用 JSON 在其他账户中创建策略时。如果模板中策略的字符超过 JSON 策略的字符上限 6144，则会将该策略拆分为多个策略。
 - 选择 Next (下一步)，以在同一账户中查看并创建托管策略。

步骤 4：查看并创建托管策略

如果您有权创建和附加 IAM policy，则可以通过生成的策略创建托管策略。然后，您可以将策略附加到账户中的用户或角色。

查看并创建策略

1. 在 Review and create managed policy (查看并创建托管策略) 页面，为创建的策略输入 Name (名称) 和 Description (描述) (可选)。
2. (可选) 在 Summary (摘要) 部分，您可以查看策略中将包含的权限。
3. (可选) 通过以键值对的形式附加标签来向策略添加元数据。有关将在 IAM 中使用标签的更多信息，请参阅 [Tagging IAM resources](#) (标记 IAM 资源)。
4. 完成后，请执行以下操作之一：
 - 您可以将新策略直接附加到用于生成策略的角色。为此，请在页面底部附近，选中 Attach policy to **YourRoleName** (将策略附加到 YourRoleName) 旁边的复选框。然后选择 Create and attach policy (创建并附加策略)。
 - 否则，请选择 Create policy (创建策略)。您创建的策略位于 IAM 控制台上，可以在 Policies (策略) 导航窗格的策略列表中找到它。
5. 您可以将创建的策略附加到账户中的实体。附加策略后，您可以删除可以附加到该实体的任何其他过于宽泛的策略。要了解如何附加托管策略，请参阅 [添加 IAM 身份权限 \(控制台 \)](#)。

使用其他账户中的 Amazon CloudTrail 数据生成策略

您可以创建将数据存储存储在中央账户中的 CloudTrail 跟踪记录，以简化管理活动。例如，您可以使用 Amazon Organizations 创建跟踪记录，以为该企业中的所有 Amazon Web Services 账户 记录所有事件。跟踪记录属于一个中央账户。如果要为账户中的用户或角色生成策略，而该账户与您存储 CloudTrail 日志数据所在的账户不同，则必须授予跨账户访问权限。为此，您需要一个角色和一个存储桶策略来授予 IAM Access Analyzer 对您的 CloudTrail 日志的权限。有关创建 Organizations 跟踪记录的更多信息，请参阅[为企业创建跟踪记录](#)。

在此示例中，假定您要为账户 A 中的用户或角色生成策略。账户 A 中的 CloudTrail 跟踪记录将 CloudTrail 日志存储在账户 B 的存储桶中。在生成策略之前，您必须进行以下更新：

1. 选择一个现有角色，或创建一个新的服务角色，以授予 IAM Access Analyzer 对账户 B (存储您的 CloudTrail 日志的位置) 中存储桶的访问权限。
2. 验证账户 B 中的 Amazon S3 存储桶对象所有权和存储桶权限策略，以便 IAM 访问分析器可以访问存储桶中的对象。

步骤 1：选择或创建用于跨账户访问的角色

- 在 Generate policy (生成策略) 屏幕上，如果您的账户中存在具有所需权限的角色，则会为您预先选择 Use an existing role (使用现有角色) 的选项。否则，请选择 Create and use a new service role (创建和使用新的服务角色)。新角色用于向 IAM Access Analyzer 授予对账户 B 中 CloudTrail 日志的访问权限。

步骤 2：验证或更新账户 B 中的 Amazon S3 存储桶配置

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Amazon S3 控制台：<https://console.aws.amazon.com/s3/>。
2. 在 Buckets (存储桶) 列表中，请选择存储 CloudTrail 日志的存储桶名称。
3. 请选择 Permissions (权限) 选项卡，然后转到 Object Ownership (对象所有权) 部分。

使用 Amazon S3 对象所有权存储桶设置来控制上传到存储桶的对象的对象的所有权。默认情况下，当其他 Amazon Web Services 账户 将对象上传到存储桶时，这些对象仍为上传账户所有。要生成策略，存储桶中的所有对象都必须归存储桶所有者所有。您可能需要更改您存储桶的 Object Ownership (对象所有权) 设置，具体视您的 ACL 使用案例而定。将 Object Ownership (对象所有权) 设置为以下选项之一。

- 强制存储桶所有者（推荐）
- 首选存储桶所有者

Important

要成功生成策略，存储桶中的对象必须归存储桶所有者所有。如果您选择使用 Bucket owner preferred（首选存储桶所有者），您只能在对象所有权更改后的时间段生成策略。

如需了解关于 Amazon S3 中的对象所有权的更多信息，请参阅《Amazon S3 用户指南》中的[控制存储桶的对象所有权和禁用 ACL](#)。

4. 向账户 B 中的 Amazon S3 存储桶策略添加权限，以允许对账户 A 中的角色进行访问。

以下示例策略针对名为 **DOC-EXAMPLE-BUCKET** 的存储桶允许 ListBucket 和 GetObject。如果访问存储桶的角色属于您企业中的账户并且名称以 AccessAnalyzerMonitorServiceRole 开头，则其会允许访问权限。使用 **aws:PrincipalArn** 作为 Resource 元素中的 Condition 可确保角色只能访问账户活动（如果其属于账户 A）。您可以将 **DOC-EXAMPLE-BUCKET** 替换为您的存储桶名称，将 organization-id 替换为企业 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PolicyGenerationBucketPolicy",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3::DOC-EXAMPLE-BUCKET",
        "arn:aws:s3::DOC-EXAMPLE-BUCKET/AWSLogs/organization-id/${aws:PrincipalAccount}/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": "organization-id"
        },
        "StringLike": {
          "aws:PrincipalArn": "arn:aws:iam::${aws:PrincipalAccount}:role/service-role/AccessAnalyzerMonitorServiceRole*"
        }
      }
    }
  ]
}
```

5. 如果您使用 Amazon KMS 加密日志，请更新您的 Amazon KMS 密钥策略以授予 IAM Access Analyzer 访问权限来使用密钥，如以下策略示例所示。将 CROSS_ACCOUNT_ORG_TRAIL_FULL_ARN 替换为您跟踪记录的 ARN，将 organization-id 替换为您的企业 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:GenerateDataKey",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:RevokeGrant",
        "kms:ScheduleKeyDeletion",
        "kms:UnscheduleKeyDeletion",
        "kms:Verify"
      ],
      "Resource": "*"
    }
  ]
}
```

```
    },
    "Action": "kms:Decrypt",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "kms:EncryptionContext:aws:cloudtrail:arn":
        "CROSS_ACCOUNT_ORG_TRAIL_FULL_ARN",
        "aws:PrincipalOrgID": "organization-id"
      },
      "StringLike": {
        "kms:ViaService": "s3.*.amazonaws.com",
        "aws:PrincipalArn": "arn:aws:iam::${aws:PrincipalAccount}:role/service-role/
AccessAnalyzerMonitorServiceRole*"
      }
    }
  }
}
```

基于 CloudTrail 活动生成策略 (Amazon CLI)

您可以通过 Amazon CLI 使用以下命令生成策略。

生成策略

- [aws accessanalyzer start-policy-generation](#)

查看生成的策略

- [aws accessanalyzer get-generated-policy](#)

取消策略生成请求

- [aws accessanalyzer cancel-policy-generation](#)

查看策略生成请求列表

- [aws accessanalyzer list-policy-generations](#)

基于 CloudTrail 活动生成策略 (Amazon API)

您可以通过 Amazon API 使用以下操作生成策略。

生成策略

- [StartPolicyGeneration](#)

查看生成的策略

- [GetGeneratedPolicy](#)

取消策略生成请求

- [CancelPolicyGeneration](#)

查看策略生成请求列表

- [ListPolicyGenerations](#)

IAM Access Analyzer 配额

IAM Access Analyzer 具有以下配额：

资源	默认配额	最大配额
具有信任账户区域的最大分析器数	1	1
具有组织信任区域的分析器的最大数目	5	20 ¹
每个分析器的最大存档规则数	100 对于每个标准，每个存档规则可具有最多 20 个值。	1,000 ¹
每个分析器每小时的最大访问预览数	1000	1000
每个策略生成处理的 Amazon CloudTrail 日志文件	100000	100000
并发策略生成器	1	1
策略生成 Amazon CloudTrail 数据大小	25 GB	25 GB
策略生成 Amazon CloudTrail 时间范围	90 天	90 天
每天生成策略	非洲（开普敦）：5 亚太地区（香港）：5 欧洲（米兰）：5 中东（巴林）：5 所有其他支持的区域：50 Note 已取消的策略生成请求适用于每日配额。	非洲（开普敦）：5 亚太地区（香港）：5 欧洲（米兰）：5 中东（巴林）：5 所有其他支持的区域：50

¹某些配额可由客户通过 [Service Quotas](#) 进行配置。

IAM 故障排除

如果您在使用 Amazon Identity and Access Management (IAM) 时遇到拒绝访问或类似的问题，请参阅本部分的主题。

主题

- [排查一般 IAM 问题 \(p. 863\)](#)
- [排查访问被拒绝错误消息 \(p. 867\)](#)
- [排查 IAM policy 问题 \(p. 870\)](#)
- [FIDO 安全密钥故障排除 \(p. 882\)](#)
- [IAM 角色问题排查 \(p. 883\)](#)
- [IAM 和 Amazon EC2 故障排除 \(p. 888\)](#)
- [排查 IAM 和 Amazon S3 的问题 \(p. 891\)](#)
- [SAML 2.0 与 Amazon 联合的问题排查 \(p. 891\)](#)

排查一般 IAM 问题

使用此处的信息可帮助您诊断和修复在使用 Amazon Identity and Access Management (IAM) 时出现的访问被拒绝或其他常见问题。

问题

- [我无法登录我的 Amazon 账户 \(p. 863\)](#)
- [我丢失了访问密钥 \(p. 863\)](#)
- [当我向某个 Amazon 服务发送请求时，收到了“访问被拒绝” \(p. 864\)](#)
- [当我使用临时安全凭证发送请求时，收到了“访问被拒绝” \(p. 864\)](#)
- [策略变量不起作用 \(p. 865\)](#)
- [我所做的更改可能不会立即可见 \(p. 865\)](#)
- [我没有权限执行 : iam:DeleteVirtualMFADevice \(p. 866\)](#)
- [如何安全地创建 IAM 用户？ \(p. 866\)](#)
- [其他资源 \(p. 867\)](#)

我无法登录我的 Amazon 账户

确认您具有正确的凭证，并且使用正确的方法进行登录。有关更多信息，请参阅[Amazon 登录或账户问题排查 \(p. 65\)](#)。

我丢失了访问密钥

访问密钥由两个部分组成：

- 访问密钥标识符。标识符是公开的，您可以在列出访问密钥的任意 IAM 控制台中进行查看，例如用户摘要页面。

- 秘密访问密钥。该密钥会在您最初创建访问密钥时提供。它与密码一样，之后无法检索。如果您丢失了私有访问密钥，则必须创建新的访问密钥对。如果您已达到[访问密钥的数量上限 \(p. 903\)](#)，必须删除一个现有的密钥对，然后才可以创建另一个。

有关更多信息，请参阅[重置您丢失或遗忘的 Amazon 密码或访问密钥 \(p. 103\)](#)。

当我向某个 Amazon 服务发送请求时，收到了“访问被拒绝”

- 检查错误消息是否包含与拒绝访问有关的策略类型。例如，如果错误提到由于服务控制策略 (SCP) 而拒绝访问，则可以专注于对 SCP 问题进行故障排除。知晓策略类型后，您还可以检查该策略类型策略中的特定操作是否存在拒绝语句或缺少允许。有关更多信息，请参阅[排查访问被拒绝错误消息 \(p. 867\)](#)。如果错误消息没有提到与拒绝访问有关的策略类型，请使用本部分中的其余指南进一步排除故障。
- 验证您是否具有调用您请求的操作和资源的基于身份的策略权限。如果设置了任何条件，您还必须在发送请求时满足这些条件。有关查看或修改用于 IAM 用户、组或角色的策略的信息，请参阅[管理 IAM 策略 \(p. 437\)](#)。
- 如果 Amazon Web Services Management Console 返回消息称您无权执行某个操作，则必须联系您的管理员寻求帮助。您的管理员为您提供了登录凭证或登录链接。

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，但不具有虚构 widgets:`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:
widgets:GetWidget on resource: my-example-widget
```

在这种情况下，Mateo 必须请求他的管理员更新其策略，以允许他使用 widgets:`GetWidget` 操作访问 `my-example-widget` 资源。

- 您是否尝试访问支持[基于资源的策略 \(p. 379\)](#) (如 Amazon S3、Amazon SNS 或 Amazon SQS) 的服务？如果是，请确认策略将您指定为主体并为您授予访问权限。如果您在您的账户中向服务发出请求，基于身份的策略或基于资源的策略将向您授予相应的权限。如果您在不同的账户中向服务发出请求，则基于身份的策略和基于资源的策略这两者都必须向您授予权限。要查看哪些服务支持基于资源的策略，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)。
- 如果您的策略包含的条件具有键值对，请仔细检查。示例包括 `aws:RequestTag/tag-key` (p. 998) 全局条件密钥、Amazon KMS `kms:EncryptionContext:encryption_context_key` 和多个服务支持的 `ResourceTag/tag-key` 条件键。确保键名称不与多个结果匹配。由于条件键名称不区分大小写，因为检查名为 `foo` 的键的条件将与 `foo`、`Foo` 或 `FOO` 匹配。如果您的请求包含多个键值对，其中的键名称只是大小写形式不同，则您的访问可能会被意外拒绝。有关更多信息，请参阅[IAM JSON 策略元素：Condition \(p. 948\)](#)。
- 如果您具有[权限边界 \(p. 370\)](#)，请验证用于权限边界的策略是否允许您的请求。如果基于身份的策略允许请求，但权限边界不允许，则会拒绝请求。权限边界控制 IAM 主体（用户或角色）可以拥有的最大权限。基于资源的策略不受权限边界限制。权限边界不常用。有关 Amazon 如何评估这些策略的更多信息，请参阅[策略评估逻辑 \(p. 972\)](#)。
- 如果您手动签署请求（不使用 [Amazon SDK](#)），请验证您已正确[签署请求](#)。

当我使用临时安全凭证发送请求时，收到了“访问被拒绝”

- 首先，请确保您未因与您的临时凭证无关的原因而被拒绝访问。有关更多信息，请参阅[当我向某个 Amazon 服务发送请求时，收到了“访问被拒绝” \(p. 864\)](#)。
- 要验证服务是否接受临时安全凭证，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)。

- 验证您的请求是否采用了正确的签名和适当的格式。有关详细信息，请参阅[工具包文档](#)或[将临时凭证用于 Amazon 资源](#) (p. 311)。
- 验证您的临时安全凭证没有过期。有关更多信息，请参阅[IAM 临时安全凭证](#) (p. 300)。
- 验证 IAM 用户或角色拥有正确的许可。临时安全证书的权限派生自 IAM 用户或角色。因此，权限限于向您已担任其临时凭证的角色授予的权限。有关如何确定临时安全凭证的权限的更多信息，请参阅[控制临时安全凭证的权限](#) (p. 313)。
- 如果您担任一个角色，您的角色会话可能受会话策略的限制。以编程方式使用 Amazon STS [请求临时安全凭证](#) (p. 302)时，您可以选择传递内联或托管[会话策略](#) (p. 357)。会话策略是高级策略，在以编程方式为角色创建临时凭证会话时，这些策略将作为参数进行传递。您可以使用 Policy 参数传递单个 JSON 内联会话策略文档。您可以使用 PolicyArns 参数指定最多 10 个托管会话策略。生成的会话的权限是角色的基于身份的策略与会话策略的交集。或者，如果您的管理员或自定义程序为您提供临时凭证，它们可能已包含会话策略以限制您的访问。
- 如果您是联合身份用户，您的会话可能受会话策略的限制。您以 IAM 用户身份登录到 Amazon，然后请求联合令牌以成为联合身份用户。有关联合身份用户的更多信息，请参阅[GetFederationToken – 通过自定义身份凭证代理程序进行联合身份验证](#) (p. 307)。在请求联合令牌时，如果您或您的身份代理传递了会话策略，则您的会话受这些策略的限制。生成的会话的权限是您的 IAM 用户的基于身份的策略与会话策略的交集。有关会话策略的更多信息，请参阅[会话策略](#) (p. 357)。
- 如果您使用角色访问具有基于资源的策略的资源，则验证策略已授予该角色权限。例如，以下策略允许 MyRole 从账户 111122223333 访问 MyBucket。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "S3BucketPolicy",
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::111122223333:role/MyRole"]},
    "Action": ["s3:PutObject"],
    "Resource": ["arn:aws:s3:::MyBucket/*"]
  }]
}
```

策略变量不起作用

- 验证包含变量的所有策略（包括以下策略版本编号）："Version": "2012-10-17"。如果没有正确的版本编号，则在评估过程中不会更换变量。只会从字面上评估这些变量。如果您包含最新的版本号，则不包含变量的任何策略仍将起作用。

Version 策略元素与策略版本不同。Version 策略元素用在策略之中，用于定义策略语言的版本。另一方面，当您 IAM 中的客户托管策略进行更改时，将创建一个策略版本。已更改的策略不会覆盖现有策略。而是由 IAM 创建新的托管策略版本。要了解 Version 策略元素的更多信息，请参阅[IAM JSON 策略元素：Version](#) (p. 934)。要了解策略版本的更多信息，请参阅[the section called “IAM 策略版本控制”](#) (p. 458)。

- 验证您的策略变量为正确的大小写。有关详细信息，请参阅[IAM policy 元素：变量和标签](#) (p. 964)。

我所做的更改可能不会立即可见

作为全球数据中心的计算机要访问的服务，IAM 使用称为[最终一致性的](#)分布式计算模型。您在 IAM（或其他 Amazon 服务）中所做的任何更改，包括在[基于属性的访问控制 \(ABAC\)](#) 中使用的标记，都需要一段时间才能在所有可能的端点中可见。它在服务器与服务器之间、复制区域与复制区域之间，以及全球的区域与区域之间发送数据需要时间，这会造成一定的延迟。IAM 也使用缓存来提高性能，但在某些情况下，这可能会增加时间。在之前缓存的数据超时之前，更改可能不可见。

您在设计全球应用程序时，必须考虑到这些可能的延迟。确保应用程序可以按预期工作，即使在一个位置进行的更改不能立即在其他位置可见。此类更改包括创建或更新用户、组、角色或策略。在应用程序的关键、

高可用性代码路径中，我们不建议进行此类 IAM 更改。而应在不常运行的、单独的初始化或设置例程中进行 IAM 更改。另外，在生产工作流程依赖这些更改之前，请务必验证更改已传播。

有关其他某些 Amazon 服务如何受此影响的更多信息，请参阅以下资源：

- Amazon DynamoDB：DynamoDB 常见问题中的[什么是 Amazon DynamoDB 的一致性模型？](#)和 Amazon DynamoDB 开发人员指南中的[读取一致性](#)。
- Amazon EC2：Amazon EC2 API 参考中的[EC2 最终一致性](#)。
- Amazon EMR：Amazon 大数据博客中的[Ensuring Consistency When Using Amazon S3 and Amazon Elastic MapReduce for ETL Workflows](#)
- Amazon Redshift：Amazon Redshift 数据库开发人员指南中的[管理数据一致性](#)
- Amazon S3：Amazon Simple Storage Service 用户指南中的[Amazon S3 数据一致性模型](#)

我没有权限执行：iam:DeleteVirtualMFADevice

当您尝试为自己或其他用户分配或删除虚拟 MFA 设备时，可能会收到以下错误：

```
User: arn:aws:iam::123456789012:user/Diego is not authorized to perform:
iam:DeleteVirtualMFADevice on resource: arn:aws:iam::123456789012:mfa/Diego with an
explicit deny
```

如果某个人以前在 IAM 控制台中开始将虚拟 MFA 设备分配给用户后取消该过程，则可能会发生这种情况。这会在 IAM 中为用户创建一个 MFA 设备，但绝不激活该设备。您必须先删除现有 MFA 设备，然后才能将新设备与用户关联。

Amazon 推荐了一种策略，该策略仅允许用户在通过 MFA 进行身份验证后删除其自己的虚拟 MFA 设备。有关更多信息，请参阅[Amazon：允许经过 MFA 身份验证的 IAM 用户在“My Security Credentials”（我的安全凭证）页面上管理自己的凭证](#)（p. 395）。

要修复此问题，管理员不应编辑策略权限。相反，管理员必须使用 Amazon CLI 或 Amazon API 来删除现有但已停用的设备。

删除现有但已停用的 MFA 设备

1. 查看您账户中的虚拟 MFA 设备。
 - Amazon CLI: `aws iam list-virtual-mfa-devices`
 - Amazon API : `ListVirtualMFADevices`
2. 在响应中，找到您要修复的用户的虚拟设备的 ARN。
3. 请删除该设备。
 - Amazon CLI: `aws iam delete-virtual-mfa-device`
 - Amazon API : `DeleteVirtualMFADevice`

如何安全地创建 IAM 用户？

如果您的员工需要访问 Amazon，则可以选择创建 IAM 用户或使用 [IAM Identity Center 进行身份验证](#)。如果使用 IAM，Amazon 建议您创建 IAM 用户并将凭证安全地传达给员工。如果您不在员工身旁，请使用安全的工作流程向员工传达凭证。

使用以下工作流程在 IAM 中安全地创建新用户：

1. 使用 Amazon Web Services Management Console [创建新用户](#)。选择使用自动生成的密码授予 Amazon Web Services Management Console 访问权限。如有必要，请选中 Users must create a new password at next sign-in (用户必须在下次登录时创建新密码) 复选框。在用户更改密码之前，不要向用户添加权限策略。
2. 添加用户后，复制新用户的登录 URL、用户名和密码。要查看密码，请选择 Show (显示)。
3. 使用贵公司中的安全通信方法 (如电子邮件、聊天或支持工单系统) 将密码发送给员工。单独为您的用户提供 IAM 用户控制台链接及其用户名。在授予员工权限之前，告诉员工确认他们可以成功登录。
4. 员工确认后，添加他们所需的权限。作为安全最佳实践，添加一个策略，要求用户使用 MFA 进行身份验证以管理其凭证。有关策略示例，请参阅 [Amazon：允许经过 MFA 身份验证的 IAM 用户在“My Security Credentials” \(我的安全凭证 \) 页面上管理自己的凭证 \(p. 395\)](#)。

其他资源

下列相关资源有助于您在使用 Amazon 期间排查问题。

- [Amazon CloudTrail 用户指南](#) - 使用 Amazon CloudTrail 可跟踪对 Amazon 进行的 API 调用的历史记录并将这些信息存储在日志文件中。这有助于确定访问了您账户中的资源的用户和账户、进行调用的时间、请求的操作等。有关更多信息，请参阅[使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用 \(p. 340\)](#)。
- [Amazon 知识中心](#) - 查找常见问题解答和其他资源的链接，帮助您排查问题。
- [Amazon 支持中心](#) - 获取技术支持。
- [Amazon Premium Support 中心](#) - 获取高级技术支持。

排查访问被拒绝错误消息

当 Amazon 显式或隐式拒绝授权请求时，将显示拒绝访问错误消息。当策略包含特定的 Amazon 操作的 Deny 语句时，将发生显式拒绝。当没有适用的 Deny 语句且没有适用的 Allow 语句时，会发生隐式拒绝。由于 IAM policy 默认拒绝 IAM 主体，因此该策略必须显式允许主体执行操作。否则，该策略会隐式拒绝访问。有关更多信息，请参阅[显式拒绝和隐式拒绝之间的区别 \(p. 980\)](#)。

大多数拒绝访问的错误消息都以 User **user** is not authorized to perform **action** on **resource** because **context** 格式显示。在此示例中，**user** (用户) 是无法获得访问权限的 [Amazon 资源名称 \(ARN \)](#)，**action** (操作) 是策略拒绝的服务操作，**resource** (资源) 是策略所执行操作的资源的 ARN。**context** (上下文) 字段表示有关策略类型的其他上下文，用于解释策略拒绝访问的原因。

当策略因包含 Deny 语句而显式拒绝访问时，Amazon 将在拒绝访问错误消息中包含短语 with an explicit deny in a **type** policy。当策略隐式拒绝访问时，Amazon 将在拒绝访问错误消息中包含短语 because no **type** policy allows the **action** action。

如果同一策略类型的多个策略拒绝授权请求，则 Amazon 在访问被拒绝错误消息中没有指定策略编号。如果授权请求被多种策略类型拒绝，则 Amazon 仅在错误消息中包含其中一种策略类型。

拒绝访问错误消息示例

下面的示例展示了不同类型的拒绝访问错误消息的格式。

由于服务控制策略而拒绝访问

对于以下错误，请在服务控制策略 (SCP) 中检查 `codecommit:ListRepositories` 的 Deny 语句或缺失的 Allow 语句。

Note

当 SCP 拒绝访问时，错误消息始终包含短语 `due to an explicit deny in a Service Control Policy`，即使拒绝是隐式的。

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories with an explicit deny in a service control policy
```

由于 VPC 端点策略而拒绝访问

- 隐式拒绝：对于以下错误，请在虚拟私有云 (VPC) 端点策略中检查 `codecommit:ListRepositories` 的缺失 Allow 语句。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories because no VPC endpoint policy allows the
codecommit:ListRepositories action
```

- 显式拒绝：对于以下错误，请在 VPC 端点策略中检查 `codecommit:ListDeployments` 的显式 Deny 语句。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* with an explicit deny in a VPC endpoint policy
```

由于权限边界而拒绝访问

- 隐式拒绝：对于以下错误，请在权限边界中检查 `codecommit:ListDeployments` 的缺失 Allow 语句。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* because no permissions boundary allows the
codedeploy:ListDeployments action
```

- 显式拒绝：对于以下错误，请在权限边界中检查 `sagemaker:ListModels` 的显式 Deny 语句。

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
sagemaker:ListModels with an explicit deny in a permissions boundary
```

由于会话策略而拒绝访问

- 隐式拒绝：对于以下错误，请在会话策略中检查 `codecommit:ListRepositories` 的缺失 Allow 语句。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories because no session policy allows the
codecommit:ListRepositories action
```

- 显式拒绝：对于以下错误，请在会话策略中检查 `codecommit:ListDeployments` 的显式 Deny 语句。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* with an explicit deny in a sessions policy
```


由于基于资源的策略而拒绝访问

- 隐式拒绝：对于以下错误，请在基于资源的策略中检查 `secretsmanager:GetSecretValue` 的缺失 Allow 语句。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
secretsmanager:GetSecretValue because no resource-based policy allows the
secretsmanager:GetSecretValue action
```

- 显式拒绝：对于以下错误，请在基于资源的策略中检查 `secretsmanager:GetSecretValue` 的显式 Deny 语句。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
secretsmanager:GetSecretValue on resource: arn:aws:secretsmanager:us-
east-1:123456789012:secret:* with an explicit deny in a resource-based policy
```

由于角色信任策略而拒绝访问

- 隐式拒绝：对于以下错误，请在角色信任策略中检查 `sts:AssumeRole` 的缺失 Allow 语句。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
sts:AssumeRole because the role trust policy allows the sts:AssumeRole action
```

- 显式拒绝：对于以下错误，请在角色信任策略中检查 `sts:AssumeRole` 的缺失 Allow 语句。

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
sts:AssumeRole with an explicit deny in the role trust policy
```

由于基于身份的策略而拒绝访问

- 隐式拒绝：对于以下错误，请在附加到用户 JohnDoe 的基于身份的策略中检查 `codecommit:ListRepositories` 的缺失 Allow 语句。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories because no identity-based policy allows the
codecommit:ListRepositories action
```

- 显式拒绝：对于以下错误，请在附加到用户 JohnDoe 的基于身份的策略中检查 `codedeploy:ListDeployments` 的显式 Deny 语句。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* with an explicit deny in an identity-based policy
```

由于其他策略，在 VPC 请求失败时拒绝访问

对于以下错误，请在 SCP 中检查 `SNS:Publish` 的显式 Deny 语句。

```
User: arn:aws:sts::111122223333:assumed-role/role-name/role-session-name is not authorized
to perform:
SNS:Publish on resource: arn:aws:sns:us-east-1:444455556666:role-name-2
```

with an explicit deny in a VPC endpoint policy transitively through a service control policy

排查 IAM policy 问题

[策略 \(p. 355\)](#) 是 Amazon 中的一个实体；在附加到身份或资源时，策略定义它们的权限。在主体（如用户）发出请求时，Amazon 将评估这些策略。策略中的权限确定是允许还是拒绝请求。策略作为 JSON 文档存储在 Amazon 中，它们作为基于身份的策略附加到主体或作为基于资源的策略附加到资源。您可以将基于身份的策略附加到主体（或身份），例如，IAM 组、用户或角色。基于身份的策略包括 Amazon 托管策略、客户托管策略和内联策略。您可以在 Amazon Web Services Management Console 中使用 Visual editor（可视化编辑器）选项卡或 JSON 选项卡创建和编辑客户托管策略。在 Amazon Web Services Management Console 中查看策略时，您可以查看该策略授予的权限的摘要。您可以使用可视化编辑器和策略摘要帮助诊断和修复在管理 IAM policy 时遇到的常见错误。

请记住，所有 IAM policy 是使用以 [JavaScript 对象表示法 \(JSON\)](#) 规则开头的语法存储的。您无需了解该语法，即可创建或管理您的策略。您可以在 Amazon Web Services Management Console 中使用可视化编辑器创建和编辑策略。要了解 IAM policy 中的 JSON 语法的更多信息，请参阅 [IAM JSON 策略语言的语法 \(p. 985\)](#)

IAM policy 故障排除主题

- [使用可视化编辑器排除故障 \(p. 870\)](#)
 - [调整策略结构 \(p. 871\)](#)
 - [在可视化编辑器中选择资源 ARN \(p. 871\)](#)
 - [在可视化编辑器中拒绝权限 \(p. 871\)](#)
 - [在可视化编辑器中指定多个服务 \(p. 872\)](#)
 - [在可视化编辑器中减小策略大小 \(p. 872\)](#)
 - [在可视化编辑器中修复无法识别的服务、操作或资源类型 \(p. 872\)](#)
- [使用策略摘要排除故障 \(p. 873\)](#)
 - [缺少策略摘要 \(p. 873\)](#)
 - [摘要策略包含无法识别的服务、操作或资源类型 \(p. 873\)](#)
 - [服务不支持 IAM policy 摘要 \(p. 874\)](#)
 - [我的策略未授予预期权限 \(p. 875\)](#)
- [策略管理故障排查 \(p. 878\)](#)
 - [在 IAM 账户中附加或分离策略 \(p. 878\)](#)
 - [根据您的 IAM 身份活动更改这些身份的策略 \(p. 878\)](#)
- [JSON 策略文档故障排查 \(p. 878\)](#)
 - [验证您的策略 \(p. 879\)](#)
 - [我在 JSON 编辑器中没有用于策略验证的权限 \(p. 879\)](#)
 - [多个 JSON 策略对象 \(p. 879\)](#)
 - [多个 JSON Statement 元素 \(p. 880\)](#)
 - [在 JSON Statement 元素中具有多个 Effect、Action 或 Resource 元素 \(p. 880\)](#)
 - [缺少 JSON Version 元素 \(p. 881\)](#)

使用可视化编辑器排除故障

在创建或编辑客户托管策略时，您可以使用可视化编辑器选项卡中的信息帮助纠正您的策略中的错误。要查看使用可视化编辑器创建策略的示例，请参阅 [the section called “控制对身份的访问” \(p. 383\)](#)。

调整策略结构

在创建策略时，Amazon 将在存储之前验证、处理和转换该策略。如果 Amazon 返回策略以响应用户查询或在控制台中显示策略，Amazon 将该策略转换回用户可读格式，而不更改策略授予的权限。这可能会导致与策略可视化编辑器或 JSON 选项卡中显示的内容有所不同：可以添加或删除可视化编辑器权限块或将这些块重新排序，并且可以优化块中的内容。在 JSON 选项卡中，可以删除不太重要的空格，并且可以将 JSON 映射中的元素重新排序。此外，可以将主体元素中的 Amazon 账户 ID 替换为 Amazon Web Services 账户根用户的 ARN。由于可能发生这些更改，不应以字符串形式来比较 JSON 策略文档。

在 Amazon Web Services Management Console 中创建客户托管策略时，您可以选择完全在 JSON 选项卡中工作。如果从未在 Visual editor（可视化编辑器）选项卡中进行任何更改并从 JSON 选项卡中选择 Review policy（查看策略），则不太可能会调整策略结构。不过，如果您创建一个策略并使用可视化编辑器选项卡进行任何修改，或者从可视化编辑器选项卡中选择查看策略，则 IAM 可能会调整策略结构以在可视化编辑器中优化其外观。

该结构调整仅在您的编辑会话中进行，不会自动进行保存。

如果在您的编辑会话中调整您的策略结构，则 IAM 根据以下情况确定是否保存结构调整：

在以下选项卡中	如果编辑您的策略	然后从以下选项卡中选择查看策略	在选择保存更改时
可视化编辑器	已编辑	可视化编辑器	将调整策略结构
可视化编辑器	已编辑	JSON	将调整策略结构
可视化编辑器	未编辑	可视化编辑器	将调整策略结构
JSON	已编辑	可视化编辑器	将调整策略结构
JSON	已编辑	JSON	不会更改策略结构
JSON	未编辑	JSON	不会更改策略结构

IAM 可能会调整复杂策略或具有权限块或语句以允许多个服务、资源类型或条件键的策略的结构。

在可视化编辑器中选择资源 ARN

在使用可视化编辑器创建或编辑策略时，您必须先选择一个服务，然后从该服务中选择操作。如果选择的服务和操作支持选择[特定的资源](#) (p. 387)，可视化编辑器将列出支持的资源类型。然后，您可以选择添加 ARN 以提供有关您的资源的详细信息。您可以从以下选项中进行选择以添加资源类型的 ARN。

- 使用 ARN 生成器 - 根据资源类型，您可能会看到用于生成 ARN 的不同字段。您也可以选择任意，以便为指定的设置提供任何值提供权限。例如，如果选择 Amazon EC2 Read（读取）访问级别组，则您的策略中的操作支持 instance 资源类型。必须为您的资源提供区域、账户和实例 ID 值。如果提供您的账户 ID，但为区域和实例 ID 选择任意，则策略为您的账户中的任何实例授予权限。
- 键入或粘贴 ARN - 您可以按 [Amazon Resource Name \(ARN\)](#) (p. 896) 指定资源。您可以在 ARN 的任何字段中包含通配符 (*)（每对冒号之间）。有关更多信息，请参阅 [IAM JSON 策略元素：Resource](#) (p. 946)。

在可视化编辑器中拒绝权限

默认情况下，使用可视化编辑器创建的策略允许执行选择的操作。要拒绝选择的操作，请选择切换到拒绝权限。由于默认拒绝请求，作为安全最佳实践，我们建议您仅允许用户所需的操作和资源的权限。只有在要覆盖其他语句或策略单独允许的权限时，您才应创建语句来拒绝权限。我们建议您将拒绝权限数限制为最

低，因为它们可能会增加解决权限问题的难度。有关 IAM 如何评估策略逻辑的更多信息，请参阅 [策略评估逻辑 \(p. 972\)](#)。

Note

默认情况下，仅 Amazon Web Services 账户 根用户有权访问该账户中的所有资源。因此，如果未以根用户身份登录，您必须具有策略授予的权限。

在可视化编辑器中指定多个服务

在使用可视化编辑器构建策略时，您每次只能选择一个服务。这是最佳实践，因为可视化编辑器允许您从该服务的操作中进行选择。然后，您可以从该服务和选定的操作支持的资源中进行选择。这样，就可以更轻松创建策略和进行故障排除。

如果您熟悉 JSON 语法，还可以使用通配符 (*) 手动指定多个服务。例如，键入 **Code***，以便为以 Code 开头的服务（如 CodeBuild 和 CodeCommit）提供权限。不过，您必须随后键入操作和资源 ARN 以完成您的策略。此外，在保存您的策略时，可能会 [调整 \(p. 871\)](#) 策略结构以在单独的权限块中包含每个服务。

或者，要在服务中使用 JSON 语法（如通配符），请使用 JSON 选项卡创建、编辑和保存策略。

在可视化编辑器中减小策略大小

在使用可视化编辑器创建策略时，IAM 将创建一个 JSON 文档以存储您的策略。您可以切换到 JSON 选项卡以查看该文档。如果该 JSON 文档超过策略的大小限制，可视化编辑器将显示一条错误消息，并禁止查看和保存您的策略。要查看 IAM 的托管策略大小限制，请参阅 [IAM 和 STS 字符限制 \(p. 905\)](#)。

要在可视化编辑器中减小您的策略大小，请编辑您的策略或将权限块移到另一个策略。错误消息包括策略文档中包含的字符数，您可以使用该信息帮助减小您的策略大小。

在可视化编辑器中修复无法识别的服务、操作或资源类型

在可视化编辑器中创建或编辑策略时，您可能会看到一条警告，指出您的策略包含无法识别的服务、操作或资源类型。

Note

IAM 将检查支持策略摘要的服务的服务名称、操作和资源类型。但是，您的策略摘要可能包含不存在的资源值或条件。始终通过 [策略模拟器 \(p. 443\)](#) 测试您的策略。

如果您的策略包含无法识别的服务、操作或资源类型，则存在以下错误之一：

- 预览服务 - 处于预览状态的服务不支持可视化编辑器。如果您参与预览，您可以忽略该警告并继续，但必须手动键入操作和资源 ARN 以完成您的策略。或者，您也可以选择 JSON 选项卡以键入或粘贴 JSON 策略文档。
- 自定义服务 - 自定义服务不支持可视化编辑器。如果您使用自定义服务，您可以忽略该警告并继续，但必须手动键入操作和资源 ARN 以完成您的策略。或者，您也可以选择 JSON 选项卡以键入或粘贴 JSON 策略文档。
- 服务不支持可视化编辑器 - 如果您的策略包含不支持可视化编辑器的已公开发布 (GA) 的服务，您可以忽略该警告并继续，但必须手动键入操作和资源 ARN 以完成您的策略。或者，您也可以选择 JSON 选项卡以键入或粘贴 JSON 策略文档。

公开提供服务是公开发布的服务，不是预览或自定义服务。如果无法识别的服务是公开提供的，并且名称拼写正确，则该服务不支持可视化编辑器。要了解如何请求 GA 服务的可视化编辑器或策略摘要支持，请参阅 [服务不支持 IAM policy 摘要 \(p. 874\)](#)。

- 操作不支持可视化编辑器 - 如果您的策略包含的受支持服务具有不支持的操作，您可以忽略该警告并继续，但必须手动键入操作和资源 ARN 以完成您的策略。或者，您也可以选择 JSON 选项卡以键入或粘贴 JSON 策略文档。

如果您的策略包含的受支持服务具有不支持的操作，则该服务不完全支持可视化编辑器。要了解如何请求 GA 服务的可视化编辑器或策略摘要支持，请参阅[服务不支持 IAM policy 摘要 \(p. 874\)](#)。

- 资源类型不支持可视化编辑器 - 如果您的策略包含的受支持操作具有不支持的资源类型，您可以忽略该警告并继续。不过，IAM 无法确认是否包含所有选定操作的资源，并且您可能会看到额外的警告。
- 拼写错误 - 在可视化编辑器中手动键入服务、操作或资源时，您创建的策略可能会包含拼写错误。作为最佳实践，请使用可视化编辑器从服务和操作列表中进行选择，然后根据提示完成资源部分。不过，如果服务不完全支持可视化编辑器，您可能需要手动键入某些策略部分。

如果您确定自己的策略不包含上述任何错误，那么您的策略可能包含拼写错误。检查服务、操作和资源类型名称有无拼写错误。例如，您可能误用了 `s2` 而不是 `s3`，或误用了 `ListMyBuckets` 而不是 `ListAllMyBuckets`。另一个常见的操作拼写错误是 ARN 中包含不必要的文字 (如 `arn:aws:s3: : :*`) 或操作中缺少冒号 (如 `iam.CreateUser`)。您可以选择查看策略以查看策略摘要并确认策略是否提供所需的权限，从而对可能包含拼写错误的策略进行评估。

使用策略摘要排除故障

您可以诊断并解决与策略摘要相关的问题。

缺少策略摘要

IAM 控制台中提供了策略摘要表，这些表总结了策略中对每个服务允许或拒绝的访问级别、资源和条件。策略在三个表中概括：[策略摘要 \(p. 484\)](#)、[服务摘要 \(p. 493\)](#)和[操作摘要 \(p. 496\)](#)。策略摘要表包括由所选策略定义的服务列表和权限摘要。您可以在 Users (用户) 页面上查看附加到用户的任何策略的[策略摘要 \(p. 483\)](#)。可以在 Policies 页面上查看托管策略的策略摘要。如果 Amazon 无法呈现策略摘要，您将看到 JSON 策略文档而不是摘要，还会收到以下错误：

无法为此策略生成摘要。您仍然可以查看或编辑 JSON 策略文档。

如果您的策略不包含摘要，则已发生下列错误之一：

- 不支持的策略元素 - IAM 不支持为包含以下[策略元素 \(p. 934\)](#)之一的策略生成策略摘要：
 - Principal
 - NotPrincipal
 - NotResource
- 无策略权限 - 如果策略不提供任何有效权限，则无法生成策略摘要。例如，如果策略包含元素 `"NotAction": "*" 的单个语句`，则它将授予对除“所有操作”(*) 之外的所有操作的访问权限。这意味着，它未授予对任何内容的 Deny 或 Allow 访问权限。

Note

在使用这些策略元素 (如 NotPrincipal、NotAction 和 NotResource) 时要谨慎。有关使用策略元素的信息，请参阅[IAM JSON 策略元素参考 \(p. 934\)](#)。

如果您提供不匹配的服务和资源，则可能会创建一个不提供有效权限的策略。如果您指定一个服务中的操作和另一个服务中的资源，则可能会发生这种情况。在这种情况下，仍然会出现策略摘要。表示有问题的唯一迹象是，摘要中的资源列可能包含来自不同服务的资源。如果此列包含不匹配的资源，那么您应该查看策略中是否有错误。为了更好地了解您的策略，请始终使用[策略模拟器 \(p. 443\)](#)进行测试。

摘要策略包含无法识别的服务、操作或资源类型

在 IAM 控制台中，如果[策略摘要 \(p. 483\)](#)包含警告符号 ()，则策略可能包含无法识别的服务、操作或资源类型。要了解策略摘要内的警告，请参阅[策略摘要 \(服务列表\) \(p. 484\)](#)。

Note

IAM 将检查支持策略摘要的服务的服务名称、操作和资源类型。但是，您的策略摘要可能包含不存在的资源值或条件。始终通过[策略模拟器 \(p. 443\)](#)测试您的策略。

如果您的策略包含无法识别的服务、操作或资源类型，则存在以下错误之一：

- 预览服务 - 处于预览状态的服务不支持策略摘要。
- 自定义服务 - 自定义服务不支持策略摘要。
- 服务不支持摘要 - 如果您的策略包括不支持策略摘要的公开发布 (GA) 服务，则该服务将包含在策略摘要表的 Unrecognized services (无法识别的服务) 部分。公开提供服务是公开发布的服务，不是预览或自定义服务。如果无法识别的服务是公开提供的，并且名称拼写正确，则该服务不支持 IAM policy 摘要。要了解如何请求 GA 服务的策略摘要支持，请参阅[服务不支持 IAM policy 摘要 \(p. 874\)](#)。
- 操作不支持摘要 - 如果您的策略包含的受支持服务具有不支持的操作，则该操作将包含在服务摘要表的 Unrecognized actions (无法识别的操作) 部分。要了解服务摘要内的警告，请参阅[服务摘要 \(操作列表\) \(p. 493\)](#)。
- 资源类型不支持摘要 - 如果您的策略包含的受支持操作具有不支持的资源类型，则该资源将包含在服务摘要表的 Unrecognized resource types (无法识别的资源类型) 部分。要了解服务摘要内的警告，请参阅[服务摘要 \(操作列表\) \(p. 493\)](#)。
- 错字 — Amazon 检查 JSON 语法是否正确，并且策略不在[策略验证 \(p. 443\)](#)中包含错字或其他错误。作为最佳实践，我们建议您打开现有策略，查看并解决任何策略验证建议。

服务不支持 IAM policy 摘要

如果 IAM policy 摘要或可视化编辑器无法识别公开提供 (GA) 服务或操作，则该服务可能不支持这些功能。公开提供服务是公开发布的服务，不是预览或自定义服务。如果无法识别的服务是公开提供的，并且名称拼写正确，则该服务不支持这些功能。如果您的策略包含的受支持服务具有不受支持的操作，则表示该服务不完全支持 IAM policy 摘要。

请求服务添加 IAM policy 摘要或可视化编辑器支持

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 找到包含不支持的服务的策略：
 - 如果该策略是托管策略，请在导航窗格中选择 Policies。在策略列表中，选择要查看的策略的名称。
 - 如果该策略是附加到用户的内联策略，请在导航窗格中选择 Users。在用户列表中，选择要查看其策略的用户名称。在用户的策略表中，展开您要查看的策略摘要的标题。
3. 在 Amazon Web Services Management Console 页脚左侧，选择 Feedback (反馈)。在 Tell us about your experience: (告诉我们有关您体验的信息) 框中，键入 **I request that the <ServiceName> service add support for IAM policy summaries and the visual editor**。如果希望多个服务支持摘要，请键入 **I request that the <ServiceName1>, <ServiceName2>, and <ServiceName3> services add support for IAM policy summaries and the visual editor**。

请求服务为缺失的操作增加 IAM policy 摘要支持

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 找到包含不支持的服务的策略：
 - 如果该策略是托管策略，请在导航窗格中选择 Policies。在策略列表中，选择要查看的策略的名称。
 - 如果该策略是附加到用户的内联策略，请在导航窗格中选择 Users。在用户列表中，选择要查看其策略的用户名称。在用户的策略表中，选择要查看的策略的名称以展开策略摘要。

3. 在策略摘要中，选择包含不支持操作的服务的名称。
4. 在 Amazon Web Services Management Console 页脚左侧，选择 Feedback (反馈)。在 Tell us about your experience: (告诉我们有关您体验的信息:) 框中，键入 **I request that the <ServiceName> service add IAM policy summary and the visual editor support for the <ActionName> action.** 如果要报告多个不支持的操作，请键入 **I request that the <ServiceName> service add IAM policy summary and the visual editor support for the <ActionName1>, <ActionName2>, and <ActionName3> actions.**

要请求包含缺少操作的不同服务，请重复最后三个步骤。

我的策略未授予预期权限

要给用户、组、角色或资源指定权限，您必须创建一个策略，它是一个定义权限的文档。策略文档包含以下元素：

- 效果 - 策略允许还是拒绝访问
- 操作 - 策略允许或拒绝的操作的列表
- 资源 - 作为操作目标的资源的列表
- 条件 (可选) - 策略在哪些情况下授予权限

要了解上述及其他策略元素，请参阅 [IAM JSON 策略元素参考 \(p. 934\)](#)。

要授予访问权限，策略必须定义具有支持资源的操作。如果策略还包含条件，该条件必须包含 [全局条件键 \(p. 998\)](#) 或必须应用至操作。要了解操作支持哪些资源，请参阅服务的 [Amazon 文档](#)。要了解操作支持哪些条件，请参阅 [Amazon 服务的操作、资源或条件键](#)。

要了解策略是否定义了不授予权限的操作、资源或条件，可使用 IAM 控制台在 <https://console.aws.amazon.com/iam/> 查看策略的 [策略摘要 \(p. 484\)](#)。您可以使用策略摘要识别和纠正策略中的问题。

对于元素不按 IAM policy 中的定义授予权限的情况，原因可能有：

- 定义了操作，但没有适用的资源 (p. 875)
- 定义了资源，但没有适用的操作 (p. 876)
- 定义了条件，但没有适用的操作 (p. 877)

要查看包含警告的策略摘要的示例，请参阅 [the section called “策略摘要 \(服务列表\)” \(p. 484\)](#)。

定义了操作，但没有适用资源

下面的策略定义了所有 `ec2:Describe*` 操作和一个特定资源。这些操作都不支持资源级权限，因而未授予任何 `ec2:Describe` 操作。资源级权限意味着操作支持使用策略的 [Resource \(p. 946\)](#) 元素中的 [ARN \(p. 896\)](#) 的资源。如果操作不支持资源级权限，则策略中的语句必须在 `*` 元素中使用通配符 (Resource)。要了解哪些服务支持资源级权限，请参阅 [使用 IAM 的 Amazon 服务 \(p. 908\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "ec2:Describe*",
    "Resource": "arn:aws:ec2:us-east-2:ACCOUNT-ID:instance/*"
  }]
}
```

此策略未提供任何权限，并且策略摘要包含以下错误：

This policy does not grant any permissions. To grant access, policies must have an action that has an applicable resource or condition.

要修复该策略，必须在 * 元素中使用 Resource。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "ec2:Describe*",
    "Resource": "*"
  }]
}
```

定义了资源，但没有适用操作

下面的策略定义了 Amazon S3 存储桶资源，但不包含可在该资源上执行的 S3 操作。该策略还向所有 Amazon CloudFront 操作授予完全访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "cloudfront:*",
    "Resource": [
      "arn:aws:cloudfront:*",
      "arn:aws:s3:::examplebucket"
    ]
  }]
}
```

该策略为所有 CloudFront 操作提供权限。但由于策略定义了 S3 examplebucket 资源而未定义任何 S3 操作，策略摘要将包含以下警告：

This policy defines some actions, resources, or conditions that do not provide permissions. To grant access, policies must have an action that has an applicable resource or condition.

要修复该策略以提供 S3 存储桶权限，必须定义可在存储桶资源上执行的 S3 操作。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "cloudfront:*",
      "s3:CreateBucket",
      "s3:ListBucket*",
      "s3:PutBucket*",
      "s3:GetBucket*"
    ],
    "Resource": [
      "arn:aws:cloudfront:*",
      "arn:aws:s3:::examplebucket"
    ]
  }]
}
```

此外，要修复该策略以只提供 CloudFront 权限，应删除 S3 资源。

定义了条件，但没有适用的操作

如果 S3 前缀等于 custom 且版本 ID 等于 1234，下面的策略为所有 S3 资源定义两个 Amazon S3 操作。但是，s3:VersionId 条件键用于对象版本标记，不受所定义的存储桶操作的支持。要了解操作支持哪些条件，请参阅 [Amazon 服务的操作、资源和条件键](#)，然后通过链接访问条件键的服务文档。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucketVersions",
        "s3:ListBucket"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "s3:prefix": [
            "custom"
          ],
          "s3:VersionId": [
            "1234"
          ]
        }
      }
    }
  ]
}
```

如果存储桶名称包含 s3:ListBucketVersions 前缀，该策略为 s3:ListBucket 操作和 custom 操作提供权限。但是，由于任何定义的操作都不支持 s3:VersionId 条件，策略摘要将包含以下错误：

This policy does not grant any permissions. To grant access, policies must have an action that has an applicable resource or condition.

要修复该策略以使用 S3 对象版本标记，就必须定义支持 s3:VersionId 条件键的 S3 操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucketVersions",
        "s3:ListBucket",
        "s3:GetObjectVersion"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "s3:prefix": [
            "custom"
          ],
          "s3:VersionId": [
            "1234"
          ]
        }
      }
    }
  ]
}
```


此策略为策略中的每个操作和条件提供权限。但由于不存在单一操作匹配两个条件的情况，策略不提供任何权限。为解决这一问题，您必须创建两个单独的语句，每个语句只包含具有要应用条件的操作。

要修复此策略，请创建两个语句。第一个语句包含支持 `s3:prefix` 条件的操作，第二个语句包含支持 `s3:VersionId` 条件的操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucketVersions",
        "s3:ListBucket"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "s3:prefix": "custom"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "s3:GetObjectVersion",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "s3:VersionId": "1234"
        }
      }
    }
  ]
}
```

策略管理故障排查

您可以诊断并解决与策略管理相关的问题。

在 IAM 账户中附加或分离策略

有些 Amazon 托管策略与服务关联。这些策略仅用于该服务的某个[服务相关角色](#) (p. 159)。在 IAM 控制台中，当您查看策略的 Summary (摘要) 页面时，该页面包含一个横幅，指示该策略与服务关联。您不能将该策略附加到 IAM 内的用户、组或角色。当您为服务创建服务相关角色时，该策略会自动附加到您的新角色。由于策略是必需的，因此您无法将策略与服务相关角色分离。

根据您的 IAM 身份活动更改这些身份的策略

您可以根据 IAM 身份 (用户、组和角色) 的活动更新其策略。要执行此操作，请在 CloudTrail 事件历史记录中查看您账户的事件。CloudTrail 事件日志包含详细的事件信息，您可以用来更改策略的权限。您可能会发现用户或角色尝试在 Amazon 中执行操作，并且该请求被拒绝。在这种情况下，您可以考虑用户或角色是否应具有执行该操作的权限。如果应该具有，您可以添加操作，甚至添加它们试图根据其策略访问的资源 ARN。或者，如果用户或角色具有他们不使用的权限，则可以考虑从策略中删除这些权限。请确保您的策略仅授予执行必需的操作所需要的[最低权限](#) (p. 754)。有关使用 CloudTrail 的更多信息，请参阅 Amazon CloudTrail 用户指南中的[在 CloudTrail 控制台中查看 CloudTrail 事件](#)。

JSON 策略文档故障排查

您可以诊断并解决与 JSON 策略文档相关的问题。

验证您的策略

当您创建或编辑 JSON 策略时，IAM 可以执行策略验证以帮助您创建有效的策略。IAM 可识别 JSON 语法错误，而 IAM Access Analyzer 将提供额外的策略检查和建议，以帮助您进一步优化策略。要了解策略验证的更多信息，请参阅 [验证 IAM 策略 \(p. 443\)](#)。要了解有关 IAM Access Analyzer 策略检查和可操作建议的更多信息，请参阅 [IAM Access Analyzer 策略验证](#)。

我在 JSON 编辑器中没有用于策略验证的权限

在 Amazon Web Services Management Console 中，如果您没有查看 IAM Access Analyzer 策略验证结果的权限，则可能会收到以下错误：

You need permissions. You do not have the permissions required to perform this operation. Ask your administrator to add permissions.

要纠正该错误，请要求管理员为您添加 `access-analyzer:ValidatePolicy` 权限。

多个 JSON 策略对象

一个 IAM policy 必须包含一个并且只能包含一个 JSON 对象。可通过在两旁放置 {} 括号来表示对象。虽然您可以通过在外部对中嵌入额外 {} 括号在 JSON 对象中嵌套其他对象，但是一个策略只能包含一个最外层的 {} 括号对。以下示例不正确，因为它在顶层包含两个对象 (以##标示)：

```
{
  "Version": "2012-10-17",
  "Statement": {
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    }
  }
}
{
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": "arn:aws:s3::my-bucket/*"
  }
}
}
```

不过，您可以使用正确的策略语法来实现上面示例的意图。可以将两个数据块合并到单个 Statement 元素中，而不是包含两个完整的策略对象（每个都有自己的 Statement 元素）。Statement 元素将两个对象组成的数组作为其值，如下示例所示 (以粗体标示)：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "arn:aws:s3::my-bucket/*"
    }
  ]
}
```

多个 JSON Statement 元素

此错误乍一看似乎是由上一部分变化而来的。但是，它在语法上是不同类型的错误。在以下示例中，顶层只有一个策略对象，由单个 {} 括号对表示。但是，该对象包含两个 Statement 元素。

一个 IAM policy 只能包含一个 Statement 元素，名称 (Statement) 在冒号左侧，它的值在冒号右侧。Statement 元素的值必须是对象，以 {} 括号表示，其中包含一个 Effect 元素、一个 Action 元素和一个 Resource 元素。以下示例不正确，因为策略对象包含两个 Statement 元素 (以##标示)：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "ec2:Describe*",
    "Resource": "*"
  },
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::my-bucket/*"
  }
}
```

值对象可以是多个值对象组成的数组。为解决此问题，可使用对象数组将两个 Statement 元素组合为一个元素，如下例所示 (以粗体标示)：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::my-bucket/*"
    }
  ]
}
```

Statement 元素的值是对象数组。此示例中的数组包含两个对象，每个对象自身是 Statement 元素的正确值。数组中的每个对象之间用逗号隔开。

在 JSON Statement 元素中具有多个 Effect、Action 或 Resource 元素

在 Statement 名称/值对的值侧，对象只能包含一个 Effect 元素、一个 Action 元素和一个 Resource 元素。以下策略不正确，因为在 Effect 的值对象中有两个 Statement 元素：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Effect": "Allow",
    "Action": "ec2:* ",
    "Resource": "*"
  }
}
```

Note

策略引擎不允许新的或已编辑的策略中出现此类错误。但是，策略引擎仍然允许在引擎更新之前保存的策略。现有策略的错误行为如下所示：

- 多个 Effect 元素：仅遵循最后一个 Effect 元素。忽略其他元素。
- 多个 Action 元素：所有 Action 元素进行内部合并，被视为单个列表。
- 多个 Resource 元素：所有 Resource 元素进行内部合并，被视为单个列表。

策略引擎不允许您保存任何有语法错误的策略。您必须先更正策略中的错误，然后才能保存策略。我们建议您查看适用于策略的任何正确[策略验证 \(p. 443\)](#)建议。

在每种情况下，解决方案都是删除不正确的多余元素。对于 Effect 元素，这十分简单：如果您希望前面的示例拒绝针对 Amazon EC2 实例的权限，则必须从策略中删除行 "Effect": "Allow"，如下所示：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "ec2:* ",
    "Resource": "*"
  }
}
```

但是，如果重复元素是 Action 或 Resource，则解决方法可能会更加复杂。您可能需要向多个操作允许(或拒绝) 权限，或者需要控制对多个资源的访问。例如，以下示例不正确，因为它有多个 Resource 元素(以##标示)：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:* ",
    "Resource": "arn:aws:s3::my-bucket",
    "Resource": "arn:aws:s3::my-bucket/*"
  }
}
```

Statement 元素的值对象中的每个必需元素都只能出现一次。解决方案是将每个值置于数组中。以下示例通过将两个单独的 Resource 元素合并为一个以数组为值对象的 Resource 元素来对此进行说明(以粗体标示)：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:* ",
    "Resource": [
      "arn:aws:s3::my-bucket",
      "arn:aws:s3::my-bucket/*"
    ]
  }
}
```

缺少 JSON Version 元素

Version 策略元素与策略版本不同。Version 策略元素用在策略之中，用于定义策略语言的版本。另一方面，当您 IAM 中的客户托管策略进行更改时，将创建一个策略版本。已更改的策略不会覆盖现有策略。

而是由 IAM 创建新的托管策略版本。要了解 Version 策略元素的更多信息，请参阅[IAM JSON 策略元素：Version \(p. 934\)](#)。要了解策略版本的更多信息，请参阅[the section called “IAM 策略版本控制” \(p. 458\)](#)。

随着 Amazon 功能的发展，为支持这些功能，IAM policy 增加了新的功能。有时，策略语法更新包括新版本号。如果您在策略中使用策略语法的较新功能，则必须向策略分析引擎告知所使用的版本。默认策略版本是“2008-10-17”。如果要使用之后引入的任何策略功能，则必须指定支持所需功能的版本号。我们建议始终包含最新的策略语法版本号（目前为 "Version": "2012-10-17"）。例如，以下策略是不正确的，因为它在资源的 ARN 中使用 `${...}` 策略变量。但是，它无法指定支持该策略变量的策略语法版本（以 **##** 标示）：

```
{
  "Statement":
  {
    "Action": "iam:*AccessKey*",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::123456789012:user/${aws:username}"
  }
}
```

通过在策略顶层添加值为 2012-10-17（支持策略变量的第一个 IAM API 版本）的 Version 元素，可解决此问题（以**粗体**标示）：

```
{
  "Version": "2012-10-17",
  "Statement":
  {
    "Action": "iam:*AccessKey*",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::123456789012:user/${aws:username}"
  }
}
```

FIDO 安全密钥故障排除

使用此处的信息可帮助您诊断和修复在使用 FIDO2 安全密钥时可能遇到的常见问题。

主题

- [我无法启用我的 FIDO 安全密钥 \(p. 882\)](#)
- [我无法使用 FIDO 安全密钥登录 \(p. 883\)](#)
- [我的 FIDO 安全密钥已丢失或损坏 \(p. 883\)](#)
- [其它问题 \(p. 883\)](#)

我无法启用我的 FIDO 安全密钥

请参阅以下解决方案，具体取决于您的身份是 IAM 用户还是系统管理员

IAM 用户

如果您无法启用 FIDO 安全密钥，请检查以下各项：

- 您使用的配置是否受支持？

有关可与 WebAuthn 和 Amazon 结合使用的设备和浏览器的信息，请参阅 [使用 FIDO 安全密钥的受支持配置 \(p. 113\)](#)。

- 您是否在使用 Mozilla Firefox？

当前的 Firefox 版本默认支持 WebAuthn。要在 Firefox 中启用对 WebAuthn 的支持，请执行以下操作：

1. 从 Firefox 地址栏中，键入 **about:config**。
2. 在所打开屏幕的“搜索”栏中键入 **webauthn**。
3. 选择 **security.webauth.webauthn**，然后将其值更改为 **true**。

- 您是否使用了任何浏览器插件？

Amazon 不支持使用插件来添加 WebAuthn 浏览器支持。请改用可对 WebAuthn 标准提供本机支持的浏览器。

即使您使用的是支持的浏览器，也可能有插件与 WebAuthn 不兼容。不兼容的插件可能会使您无法启用和使用符合 FIDO 的安全密钥。您应禁用任何可能不兼容的插件，然后重新启动浏览器。然后重新尝试启用 FIDO 安全密钥。

- 您是否具有适当的权限？

如果没有上述任何兼容性问题，可能您不具有适当的权限。请联系您的系统管理员。

系统管理员

如果您是管理员，您的 IAM 用户尽管使用支持的配置但仍不能启用其 FIDO 安全密钥，请确保他们具有适当的权限。有关详细示例，请参阅 [IAM 教程：允许用户管理其凭证和 MFA 设置 \(p. 56\)](#)。

我无法使用 FIDO 安全密钥登录

如果您是 IAM 用户，并且无法使用 WebAuthn 登录 Amazon Web Services Management Console，请先参阅 [使用 FIDO 安全密钥的受支持配置 \(p. 113\)](#)。如果您使用的是受支持的配置，但无法登录，请联系您的系统管理员以获得帮助。

我的 FIDO 安全密钥已丢失或损坏

一次只向一个用户分配一台 MFA 设备（虚拟、FIDO 安全密钥或硬件）。更换 FIDO 安全密钥与更换硬件 MFA 设备相似。有关在任何类型的 MFA 设备丢失或损坏时如何操作的更多信息，请参阅 [若 MFA 设备遗失或停止工作，该怎么办？ \(p. 127\)](#)。

其它问题

如果您遇到此处未涉及的 FIDO 安全密钥问题，请执行下列操作之一：

- IAM 用户：请联系您的系统管理员。
- Amazon Web Services 账户根用户：请联系 [Amazon Support](#)。

IAM 角色问题排查

使用此处的信息可帮助您诊断和修复在使用 IAM 角色时可能遇到的常见问题。

主题

- [我无法代入角色 \(p. 884\)](#)
- [我的 Amazon 账户中出现新角色 \(p. 885\)](#)
- [我无法编辑或删除我的 Amazon 账户中的角色 \(p. 885\)](#)

- 我无权执行 : [iam:PassRole](#) (p. 885)
- 为什么我无法在 12 小时会话中担任角色 ? (Amazon CLI、Amazon API) (p. 886)
- 当我尝试在 IAM 控制台中切换角色时收到错误 (p. 886)
- 我的角色具有一个允许我执行操作的策略，但出现“访问被拒绝”错误 (p. 886)
- 服务未创建角色的默认策略版本 (p. 886)
- 控制台中没有服务角色的使用案例 (p. 887)

我无法代入角色

请检查以下事项：

- 要允许用户在角色会话中重新代入当前角色，请将角色 ARN 或 Amazon Web Services 账户 ARN 指定为角色信任策略中的主体。提供计算资源的 Amazon Web Services (例如 Amazon EC2、Amazon ECS、Amazon EKS 和 Lambda) 会提供临时凭证并自动轮换这些凭证。这样可以确保您始终拥有一组有效的凭证。对于这些服务，无需重新代入当前角色即可获得临时凭证。但是，如果您需要传递 [会话标签](#) (p. 293) 或者 [会话策略](#) (p. 357)，则需要重新代入当前角色。要了解如何修改角色信任策略以添加主体角色 ARN 或 Amazon Web Services 账户 ARN，请参阅 [修改角色信任策略 \(控制台\)](#) (p. 261)。
- 当您使用 Amazon Web Services Management Console 担任角色时，确保使用角色的确切名称。担任角色时，角色名称区分大小写。
- 使用 Amazon STS API 或 Amazon CLI 担任角色时，请确保使用 ARN 中角色的确切名称。担任角色时，角色名称区分大小写。
- 验证您的 IAM policy 是否为您授予为要担任的角色调用 `sts:AssumeRole` 的权限。您的 IAM policy 的 `Action` 元素必须允许您调用 `AssumeRole` 操作。此外，您的 IAM policy 的 `Resource` 元素必须指定您要带入的角色。例如，`Resource` 元素可以按其 Amazon Resource Name (ARN) 或按通配符 (*) 指定角色。例如，至少一个适用于您的策略必须授予与以下权限类似的权限：

```
"Effect": "Allow",
"Action": "sts:AssumeRole",
"Resource": "arn:aws:iam::account_id_number:role/role-name-you-want-to-assume"
```

- 验证您的 IAM 身份是否标记有 IAM policy 需要的任何标签。例如，在以下策略权限中，`Condition` 元素要求您 (因为主体请求担任该角色) 必须具有特定标签。您必须被标记有 `department = HR` 或 `department = CS`。否则，您无法担任该角色。要了解如何标记 IAM 用户和角色，请参阅 [the section called “标记 IAM 资源”](#) (p. 276)。

```
"Effect": "Allow",
"Action": "sts:AssumeRole",
"Resource": "*",
"Condition": {"StringEquals": {"aws:PrincipalTag/department": [
    "HR",
    "CS"
]}}
```

- 确保您满足角色的信任策略中指定的所有条件。`Condition` 可以指定有效期、外部 ID 或请求必须来自于特定 IP 地址。请考虑以下示例：如果当前日期是指定日期以后的任意时间，则策略根本不会匹配，并且无法为您授予担任该角色的权限。

```
"Effect": "Allow",
"Action": "sts:AssumeRole",
"Resource": "arn:aws:iam::account_id_number:role/role-name-you-want-to-assume"
"Condition": {
    "DateLessThan": {
        "aws:CurrentTime": "2016-05-01T12:00:00Z"
    }
}
```


- 验证您用于调用 AssumeRole 的 Amazon 账户是否为您要担任的角色的受信任实体。在角色的信任策略中将受信任实体定义为 Principal。以下示例是一个信任策略，它附加到您要担任的角色。在该示例中，您登录时使用的 IAM 用户的账户 ID 必须是 123456789012。如果您的账号未在角色信任策略的 Principal 元素中列出，则无法担任该角色。在访问策略中为您授予何种权限无关紧要。注意，示例策略将权限限制为在 2017 年 7 月 1 日至 2017 年 12 月 31 日 (UTC 时间，这两个日期也包含在内) 之间发生的操作。如果您在上述日期范围之前或之后登录，则策略不匹配，您无法担任该角色。

```
"Effect": "Allow",
"Principal": { "AWS": "arn:aws:iam::123456789012:root" },
"Action": "sts:AssumeRole",
"Condition": {
  "DateGreaterThan": { "aws:CurrentTime": "2017-07-01T00:00:00Z" },
  "DateLessThan": { "aws:CurrentTime": "2017-12-31T23:59:59Z" }
}
```

- 来源身份 — 管理员可以配置角色以要求身份来传递自定义字符串，该字符串标识在 Amazon 中执行操作的个人或应用程序，称为源身份。验证所担任的角色是否需要设置源身份。更多有关源身份的信息，请参阅 [监控和控制使用所担任角色执行的操作 \(p. 316\)](#)。

我的 Amazon 账户中出现新角色

某些 Amazon 服务要求您使用直接与该服务相链接的唯一服务角色类型。该[服务相关角色 \(p. 159\)](#)由服务预定义，具有服务所需的所有权限。这样您就不必手动添加必要的权限，从而能够更轻松地进行设置。有关服务相关角色的一般信息，请参阅[使用服务相关角色 \(p. 208\)](#)。

在开始支持服务相关角色时，您可能已在使用服务。如果事实如此，您可能会收到一封电子邮件，告知您账户中将有新角色。该角色包括此项服务代表您执行操作所需的所有权限。您不需要执行任何操作支持该角色。但是，您不能从账户中删除该角色。这样会删除此项服务访问 Amazon 资源所需的权限。您可以转至 IAM 控制台的 IAM Roles (角色) 页面来查看您的账户中的服务相关角色。服务相关角色在表的 Trusted entities (可信实体) 列中随 (Service-linked role) [(服务相关角色)] 一起显示。

有关哪些服务支持服务相关角色的信息，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)并查找服务相关角色列为是的服务。有关使用某个服务的服务相关角色的信息，请选择 Yes 链接。

我无法编辑或删除我的 Amazon 账户中的角色

您不能删除或编辑 IAM 中[服务相关角色 \(p. 159\)](#)的权限。这些角色包括服务代表您执行操作所需的预定义信任和权限。您可以使用 IAM 控制台、Amazon CLI 或 API 仅编辑服务相关角色的描述。您可以转至控制台中的 IAM Roles (角色) 页面来查看您的账户中的服务相关角色。服务相关角色在表的 Trusted entities (可信实体) 列中随 (Service-linked role) [(服务相关角色)] 一起显示。角色的 Summary 页面上的横幅也指示角色是服务相关角色。您只能通过链接的服务管理和删除这些角色 (如果该服务支持此操作)。修改或删除服务相关角色时要小心，因为这样做可能会删除服务访问 Amazon 资源所需的权限。

有关哪些服务支持服务相关角色的信息，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)并查找服务相关角色列为是的服务。

我无权执行 : iam:PassRole

在创建服务相关角色时，您必须有权将该角色传递给服务。在某些服务中执行操作时，该服务自动在您的账户中创建一个服务相关角色。例如，在首次创建 Auto Scaling 组时，Amazon EC2 Auto Scaling 为您创建 AWSServiceRoleForAutoScaling 服务相关角色。如果您尝试创建 Auto Scaling 组而没有 PassRole 权限，则会出现以下错误：

```
ClientError: An error occurred (AccessDenied) when calling the
PutLifecycleHook operation: User: arn:aws:sts::111122223333:assumed-role/
```

```
Testrole/Diego is not authorized to perform: iam:PassRole on resource:
arn:aws:iam::111122223333:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling
```

要纠正该错误，请要求管理员为您添加 `iam:PassRole` 权限。

要了解哪些服务支持服务相关角色，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)。要了解服务是否自动为您创建服务相关角色，请选择是链接以查看服务的角色相关文档。

为什么我无法在 12 小时会话中担任角色？(Amazon CLI、Amazon API)

在使用 Amazon STS `AssumeRole*` API 或 `assume-role*` CLI 操作担任角色时，您可以为 `DurationSeconds` 参数指定一个值。您可以指定 900 秒（15 分钟）到角色的最大会话持续时间设置之间的值。如果指定的值高于该设置，操作将失败。该设置的最大值为 12 小时。例如，如果您指定的会话持续时间为 12 小时，但管理员设置的最大会话持续时间为 6 小时，您的操作将失败。要了解如何查看您的角色的最大值，请参阅[查看角色的最大会话持续时间设置 \(p. 238\)](#)。

如果您使用[角色链 \(p. 159\)](#)（使用一个角色担任另一个角色），您的会话限制为最多 1 小时。如果您随后使用 `DurationSeconds` 参数提供大于 1 小时的值，操作将失败。

当我尝试在 IAM 控制台中切换角色时收到错误

您在切换角色页上输入的信息必须与角色的信息匹配。否则，操作失败，您会收到以下错误：

```
Invalid information in one or more fields. Check your information or contact
your administrator.
```

如果您收到此错误，请确认以下信息是否正确：

- 账户 ID 或别名 - Amazon 账户 ID 是 12 位数字。您的账户可能有一个别名，这是一个友好的标识符，例如您的公司名称，可用来代替您的 Amazon 账户 ID。您可以在此字段中使用账户 ID 或别名。
- 角色名称 - 角色名称区分大小写。账户 ID 和角色名称必须与为角色配置的内容匹配。

如果您继续收到错误消息，请与您的管理员联系以验证以前的信息。角色信任策略或 IAM 用户策略可能会限制您的访问权限。您的管理员可以验证这些策略的权限。

我的角色具有一个允许我执行操作的策略，但出现“访问被拒绝”错误

您的角色会话可能受会话策略的限制。以编程方式使用 Amazon STS [请求临时安全凭证 \(p. 302\)](#) 时，您可以选择传递内联或托管[会话策略 \(p. 357\)](#)。会话策略是高级策略，在以编程方式为角色创建临时凭证会话时，这些策略将作为参数进行传递。您可以使用 `PolicyArns` 参数传递单个 JSON 内联会话策略文档。您可以使用 `PolicyArns` 参数指定最多 10 个托管会话策略。生成的会话的权限是角色的基于身份的策略与会话策略的交集。或者，如果您的管理员或自定义程序为您提供临时凭证，它们可能已包含会话策略以限制您的访问。

服务未创建角色的默认策略版本

服务角色是服务代表您在您的账户中执行操作而担任的角色。在设置一些 Amazon 服务环境时，您必须为服务定义要代入的角色。在某些情况下，服务会在 IAM 中为您创建服务角色及其策略。尽管您可以从 IAM 内部修改或删除服务角色及其策略，但 Amazon 不建议这样做。角色和策略仅供该服务使用。如果编辑策略并设置另一个环境，则当服务尝试使用相同的角色和策略时，操作可能会失败。

例如，首次使用 Amazon CodeBuild 时，服务会创建一个名为 `codebuild-RWBCore-service-role` 的角色。该服务角色使用名为 `codebuild-RWBCore-managed-policy` 的策略。如果编辑策略，它会创建一个新版本并将该版本保存为默认版本。如果您在 Amazon CodeBuild 中执行后续操作，服务可能会尝试更新策略。如果是这样，您会收到以下错误：

```
codebuild.amazonaws.com did not create the default version (V2) of the codebuild-RWBCore-managed-policy policy that is attached to the codebuild-RWBCore-service-role role. To continue, detach the policy from any other identities and then delete the policy and the role.
```

如果您收到此错误，则必须在 IAM 中进行更改，然后才能继续服务操作。首先，将默认策略版本设置为 V1，然后重试该操作。如果先前删除了 V1，或者如果选择 V1 不起作用，则清理并删除现有策略和角色。

有关编辑托管策略的更多信息，请参阅[编辑客户托管策略（控制台）](#) (p. 461)。有关策略版本的更多信息，请参阅[IAM 策略版本控制](#) (p. 458)。

删除服务角色及其策略

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在导航窗格中，选择 Policies (策略)。
3. 在策略列表中，选择要删除的策略的名称。
4. 请选择策略使用情况选项卡，以查看哪些 IAM 用户、组或角色使用此策略。如果这些身份中的任何一个使用此策略，请完成以下任务：
 - a. 创建具有必要权限的新托管策略。要确保这些身份在操作之前和之后具有相同的权限，请从现有策略中复制 JSON 策略文档。然后创建新的托管策略并粘贴 JSON 文档，如[JSON 选项卡上的创建策略](#) (p. 438)中所述。
 - b. 对于每个受影响的身份，附加新策略，然后分离旧策略。有关更多信息，请参阅[添加和删除 IAM 身份权限](#) (p. 451)。
5. 在导航窗格中，选择 Roles (角色)。
6. 在角色列表中，选择要删除的角色的名称。
7. 选择信任关系选项卡以查看哪些实体可以代入该角色。如果列出了服务以外的任何实体，请完成以下任务：
 - a. [创建信任这些实体的新角色](#) (p. 216)。
 - b. 您在上一步中创建的策略。如果跳过了该步骤，请立即创建新的托管策略。
 - c. 通知代入该角色的任何人，他们不能再这样做。向他们提供有关如何代入新角色并具有相同权限的信息。
8. [删除策略](#) (p. 465)。
9. [删除角色](#) (p. 272)。

控制台中没有服务角色的使用案例

某些服务要求您手动创建服务角色才能向服务授予权限，以代表您执行操作。如果该服务未在 IAM 控制台中列出，您必须手动将该服务列为受信任的主体。如果您正在使用的服务或功能的文档中不包含将服务列为受信任主体的说明，请为该页面提供反馈。

要手动创建服务角色，您必须知道将担任该角色的服务的[服务主体](#) (p. 940)。服务主体是一个标识符，用于向服务授予权限。服务主体由服务定义。

您可以通过检查以下内容来查找某些服务的服务主体：

1. 打开 [使用 IAM 的 Amazon 服务 \(p. 908\)](#)。
2. 检查 Service-linked roles (服务相关角色) 列中是否具有 Yes (是) 的服务。
3. 选择 Yes 链接以查看该服务的服务相关角色文档。
4. 查找该服务的“服务相关角色权限”部分，查看[服务主体 \(p. 940\)](#)。

您可以使用 [Amazon CLI 命令 \(p. 221\)](#) 或 [Amazon API 操作 \(p. 223\)](#) 手动创建服务角色。要使用 IAM 控制台手动创建服务角色，请完成以下任务：

1. 使用您的账户 ID 创建 IAM 角色。请勿附加策略或授予任何权限。有关详细信息，请参阅[创建向 IAM 用户委派权限的角色 \(p. 215\)](#)。
2. 打开角色并编辑信任关系。角色必须信任该服务，而不是信任该账户。例如，更新以下 Principal 元素：

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

将主体更改为服务的值，例如 IAM。

```
"Principal": { "Service": "iam.amazonaws.com" }
```

3. 通过将权限策略附加到角色来添加服务所需的权限。
4. 返回到需要权限的服务，然后使用记录的方法将新的服务角色通知服务。

IAM 和 Amazon EC2 故障排除

使用此处的信息可帮助您排查和修复在使用 Amazon EC2 和 IAM 时可能遇到的拒绝访问或其他问题。

主题

- [当尝试启动实例时，我没有在 Amazon EC2 控制台的 IAM 角色列表中看到本应看到的角色 \(p. 888\)](#)
- [我的实例的凭证适用于错误的角色 \(p. 889\)](#)
- [当时尝试调用 AddRoleToInstanceProfile 时，发生 AccessDenied 错误 \(p. 889\)](#)
- [Amazon EC2：在我尝试使用角色启动实例时，出现 AccessDenied 错误 \(p. 889\)](#)
- [我无法访问我的 EC2 实例的临时安全凭证 \(p. 889\)](#)
- [IAM 树形子目录中 info 文档的错误意味着什么？ \(p. 890\)](#)

当尝试启动实例时，我没有在 Amazon EC2 控制台的 IAM 角色列表中看到本应看到的角色

请检查以下事项：

- 如果您以 IAM 用户身份登录，请验证您是否有调用 ListInstanceProfiles 的权限。有关使用角色所需的权限的信息，请参阅中“在 [使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限 \(p. 252\)](#) 中通过 Amazon EC2 使用角色所需的权限”。有关为用户添加权限的信息，请参阅 [管理 IAM 策略 \(p. 437\)](#)。

如果您无法修改自己的权限，您必须联系可以使用 IAM 的管理员来更新您的权限。

- 如果您使用 IAM CLI 或 API 创建角色，则验证您已创建了实例配置文件，并将角色添加到了该实例配置文件。另外，如果角色和实例配置文件的名称不同，则无法在 Amazon EC2 控制台的 IAM 角色列表中看到正确的角色名称。Amazon EC2 控制台的 IAM 角色列表列明了实例配置文件的名称，而不是角色名称。

您必须选择包含所需角色的实例配置文件的名称。有关实例配置文件的详细信息，请参阅 [使用实例配置文件 \(p. 258\)](#)。

Note

如果您使用 IAM 控制台创建角色，则不需要使用实例配置文件。对于您在 IAM 控制台中创建的每个角色，使用与角色相同的名称创建实例配置文件，而且角色将自动添加到实例配置文件。一个实例配置文件只能包含一个 IAM 角色，不能提高该限制。

我的实例的凭证适用于错误的角色

最近可能替换了实例配置文件中的角色。如果是，您的应用程序需要等待下一次自动计划的凭证轮换，然后才能使用您的角色的凭证。

当时尝试调用 `AddRoleToInstanceProfile` 时，发生 `AccessDenied` 错误

如果您以 IAM 用户身份发出请求，请验证您拥有以下权限：

- `iam:AddRoleToInstanceProfile` 对匹配实例配置文件 ARN (例如 `arn:aws:iam::999999999999:instance-profile/ExampleInstanceProfile`) 的资源执行操作的权限。

有关使用角色所需权限的更多信息，请参阅 [使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限 \(p. 252\)](#) 中的“如何入门？”有关为用户添加权限的信息，请参阅 [管理 IAM 策略 \(p. 437\)](#)。

Amazon EC2：在我尝试使用角色启动实例时，出现 `AccessDenied` 错误

请检查以下事项：

- 发布不包含实例配置文件的实例。这样做可以帮助您确认问题的范围在 Amazon EC2 实例的 IAM 角色中。
- 如果您以 IAM 用户身份发出请求，请验证您拥有以下权限：
 - 对通配符资源 ("*") 执行 `ec2:RunInstances` 操作的许可
 - `iam:PassRole` 对匹配角色 ARN (例如 `arn:aws:iam::999999999999:role/ExampleRoleName`) 的资源执行操作的权限。
- 调用 IAM `GetInstanceProfile` 操作，确认您正使用有效的实例配置文件名称或有效的实例配置文件 ARN。有关更多信息，请参见[通过 Amazon EC2 实例使用 IAM 角色](#)。
- 调用 IAM `GetInstanceProfile` 操作，确认实例配置文件中包含角色。空的实例配置文件将发生 `AccessDenied` 错误，运行失败。有关创建角色的更多信息，请参阅 [创建 IAM 角色 \(p. 215\)](#)。

有关使用角色所需权限的更多信息，请参阅 [使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限 \(p. 252\)](#) 中的“如何入门？”有关为用户添加权限的信息，请参阅 [管理 IAM 策略 \(p. 437\)](#)。

我无法访问我的 EC2 实例的临时安全凭证

要访问 EC2 实例上的临时安全凭证，您必须首先使用 IAM 控制台创建角色。然后，您使用该角色启动 EC2 实例并检查运行中的实例。有关更多信息，请参阅 [使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限 \(p. 252\)](#) 中的怎样入门？。

如果您仍然无法访问 EC2 实例上的临时安全凭证，请检查以下内容：

- 您是否可以访问实例元数据服务 (IMDS) 的其他部分？如果不能，请检查确认没有阻止向 IMDS 发送请求的防火墙规则。

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/  
hostname; echo
```

- 是否存在 IMDS 的 iam 子树？如果不存在，请调用 EC2 DescribeInstances API 操作或者使用 aws ec2 aws ec2 describe-instances CLI 命令，验证您的实例是否具有与其关联的 IAM 实例配置文件。

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam;  
echo
```

- 检查 IAM 子树中的 info 文档是否指示发生错误。如果发生错误，请参阅 [IAM 树形子目录中 info 文档的错误意味着什么？](#) (p. 890) 来了解更多信息。

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam/  
info; echo
```

IAM 树形子目录中 info 文档的错误意味着什么？

iam/info 文档指示 "Code": "InstanceProfileNotFound"

您的 IAM 实例配置文件已经被删除，Amazon EC2 无法再为您的实例继续提供凭证。您必须将有效的实例配置文件附加到您的 Amazon EC2 实例。

如果存在具有该名称的实例配置文件，请检查是该实例配置文件没有被删除还是以相同名称创建了另一个配置文件：

1. 调用 IAM GetInstanceProfile 操作，获取 InstanceProfileId。
2. 调用 Amazon EC2 DescribeInstances 操作，获取实例的 IamInstanceProfileId。
3. 验证 IAM 操作的 InstanceProfileId 与 Amazon EC2 操作的 IamInstanceProfileId 是否匹配。

如果两个 ID 不同，那么附加至实例的实例配置文件将失去效力。您必须将有效的实例配置文件附加到该实例。

iam/info 文档指示成功，但指示 "Message": "Instance Profile does not contain a role..."。

这意味着已经通过 IAM RemoveRoleFromInstanceProfile 操作从实例配置文件删除了该角色。您可以使用 IAM AddRoleToInstanceProfile 操作为实例配置文件附加一个角色。您的应用程序需要等到下一次更新才能获得适用于该角色的凭证。

iam/security-credentials/[role-name] 文档指示 "Code": "AssumeRoleUnauthorizedAccess"

Amazon EC2 没有担任角色的许可。担任该角色的权限受制于该角色附属的信任策略，如下例所示。请使用 IAM UpdateAssumeRolePolicy API 更新信任策略。

```
{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "ec2.amazonaws.com" }, "Action": [ "sts:AssumeRole" ] } ] }
```

您的应用程序需要等到下一次自动更新才能获得适用于该角色的凭证。

排查 IAM 和 Amazon S3 的问题

使用此处的信息可帮助您对使用 Amazon S3 和 IAM 过程中可能遇到的问题进行排查和修复。

如何授权匿名访问 Amazon S3 存储桶？

若您在 `principal` 元素中使用指定通配符的 (*) 的 Amazon S3 存储桶策略，则意味着任何人都可以访问该存储桶。通过匿名访问，任何人 (包括没有 Amazon 账户的用户) 都能够访问该存储桶。有关策略示例，请参阅 Amazon Simple Storage Service 用户指南中的 [Amazon S3 存储桶策略示例](#)。

我以 Amazon 账户根用户的身份登录后，为什么无法在自己的账户中访问 Amazon S3 存储桶？

在某些情况下，您可能拥有能够完全访问 IAM 和 Amazon S3 的 IAM 用户。如果 IAM 用户将存储桶策略指定给 Amazon S3 存储桶，而且未指定 Amazon Web Services 账户根用户作为主体，则根用户访问该存储桶会被拒绝。不过，作为根用户，您仍然可以访问该存储桶。为此，请修改存储桶策略以允许根用户从 Amazon S3 控制台或 Amazon CLI 中进行访问。使用以下主体，同时将 `123456789012` 替换为 Amazon 账户的 ID。

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

SAML 2.0 与 Amazon 联合的问题排查

使用此处的信息可帮助您诊断和修复在将 SAML 2.0 与 IAM 联合使用时可能遇到的问题。

主题

- 错误：您的请求包含无效的 SAML 响应。要注销，请单击此处。(p. 892)
- 错误：AuthnResponse 中需要 RoleSessionName (服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken) (p. 892)
- 错误：未授权执行 sts:AssumeRoleWithSAML (服务：AWSSecurityTokenService；状态代码：403；错误代码：AccessDenied) (p. 892)
- 错误：AuthnResponse 中的 RoleSessionName 必须匹配 [a-zA-Z_0-9+=[.~@-]{2,64} (服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken) (p. 893)
- 错误：源身份必须匹配 [a-zA-Z_0-9+=[.~@-]{2,64} 且不能以 "aws:" 开始 (服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken) (p. 893)
- 错误：响应签名无效 (服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken) (p. 893)
- 错误：无法担任角色：指定提供商中没有发布者 (服务：AWSOpenIdDiscoveryService；状态代码：400；错误代码：AuthSamlInvalidSamlResponseException) (p. 893)
- 错误：无法解析元数据。(p. 893)
- 错误：指定的提供商不存在。(p. 894)
- 错误：请求的 DurationSeconds 超过为该角色设置的 MaxSessionDuration。(p. 894)

- [如何在浏览器中查看 SAML 响应以排查问题 \(p. 894\)](#)

错误：您的请求包含无效的 SAML 响应。要注销，请单击此处。

当来自身份提供商的 SAML 响应没有包含将 Name 设置为 `https://aws.amazon.com/SAML/Attributes/Role` 的属性时，会出现此错误。属性必须包含一个或多个 `AttributeValue` 元素，每个元素中包含以逗号分隔的一对字符串：

- 用户可以映射到的角色的 ARN
- SAML 提供商的 ARN

有关更多信息，请参阅[为身份验证响应配置 SAML 断言 \(p. 194\)](#)。要在浏览器中查看 SAML 响应，请按照[如何在浏览器中查看 SAML 响应以排查问题 \(p. 894\)](#)中列出的步骤操作。

错误：AuthnResponse 中需要 RoleSessionName (服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken)

当来自身份提供商的 SAML 响应没有包含将 Name 设置为 `https://aws.amazon.com/SAML/Attributes/RoleSessionName` 的属性时，会出现此错误。属性值是用户的标识符，通常是用户 ID 或电子邮件地址。

有关更多信息，请参阅[为身份验证响应配置 SAML 断言 \(p. 194\)](#)。要在浏览器中查看 SAML 响应，请按照[如何在浏览器中查看 SAML 响应以排查问题 \(p. 894\)](#)中列出的步骤操作。

错误：未授权执行 sts:AssumeRoleWithSAML (服务：AWSSecurityTokenService；状态代码：403；错误代码：AccessDenied)

当在 SAML 响应中指定的 IAM 角色有拼写错误或者不存在时，会出现此错误。确保使用角色的确切名称，因为角色名称区分大小写。在 SAML 服务提供商配置中更正角色的名称。

只有当角色信任策略包含 `sts:AssumeRoleWithSAML` 操作时，才允许您访问。如果您的 SAML 断言配置为使用 [PrincipalTag 属性 \(p. 195\)](#)，则信任策略还必须包含 `sts:TagSession` 操作。有关会话标签的更多信息，请参阅[在 Amazon STS 中传递会话标签 \(p. 293\)](#)。

如果角色信任策略中没有 `sts:SetSourceIdentity` 权限，则可能发生此错误。如果您的 SAML 断言配置为使用 [SourceIdentity \(p. 197\)](#) 属性，则信任策略还必须包含 `sts:SetSourceIdentity` 操作。更多有关源身份的信息，请参阅[监控和控制使用所担任角色执行的操作 \(p. 316\)](#)。

如果联合身份用户没有担任该角色的权限，则会出现此错误。该角色必须具有将 IAM SAML 身份提供商的 ARN 指定为 `Principal` 的信任策略。角色还包含控制哪些用户可以担任该角色的条件。确保您的用户满足条件要求。

如果 SAML 响应中没有包含 `Subject` 的 `NameID`，也会出现此错误。

有关更多信息，请参阅[在 Amazon 中为联合身份用户设置权限和为身份验证响应配置 SAML 断言 \(p. 194\)](#)。要在浏览器中查看 SAML 响应，请按照[如何在浏览器中查看 SAML 响应以排查问题 \(p. 894\)](#)中列出的步骤操作。

错误：AuthnResponse 中的 RoleSessionName 必须匹配 [a-zA-Z_0-9+=,.,@-]{2,64} (服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken)

如果 RoleSessionName 属性值太长或者包含无效字符，则会出现此错误。最大有效长度为 64 个字符。

有关更多信息，请参阅 [为身份验证响应配置 SAML 断言 \(p. 194\)](#)。要在浏览器中查看 SAML 响应，请按照 [如何在浏览器中查看 SAML 响应以排查问题 \(p. 894\)](#) 中列出的步骤操作。

错误：源身份必须匹配 [a-zA-Z_0-9+=,.,@-]{2,64} 且不能以 "aws:" 开始 (服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken)

如果 sourceIdentity 属性值太长或者包含无效字符，则会出现此错误。最大有效长度为 64 个字符。更多有关源身份的信息，请参阅 [监控和控制使用所担任角色执行的操作 \(p. 316\)](#)。

有关创建 SAML 断言的更多信息，请参阅 [为身份验证响应配置 SAML 断言 \(p. 194\)](#)。要在浏览器中查看 SAML 响应，请按照 [如何在浏览器中查看 SAML 响应以排查问题 \(p. 894\)](#) 中列出的步骤操作。

错误：响应签名无效 (服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken)

在身份提供商的联合身份元数据与 IAM 身份提供商的元数据不匹配时，会出现此错误。例如，身份服务提供商的元数据文件可能已更改，以便更新过期的证书。从身份服务提供商下载更新后的 SAML 元数据文件。然后在 Amazon 身份提供商实体中更新它，该实体在 IAM 中使用 `aws iam update-saml-provider` 跨平台 CLI 命令或 `Update-IAMSAMLProvider` PowerShell cmdlet 定义。

错误：无法担任角色：指定提供商中没有发布者 (服务：AWSOpenIdDiscoveryService；状态代码：400；错误代码：AuthSamlInvalidSamlResponseException)

如果 SAML 响应中的颁发者与联合元数据文件中声明的颁发者不匹配，则可能会出现该错误。在 IAM 中创建身份提供商时，元数据文件将上传到 Amazon 中。

错误：无法解析元数据。

如果您的元数据文件没有正确格式化，则可能发生此错误。

在 Amazon Web Services Management Console 中 [创建或管理 SAML 身份提供商 \(p. 189\)](#) 时，您必须从您的身份提供商处检索 SAML 元数据文档。

此元数据文件包括颁发者名称、过期信息以及可用来验证从 IdP 处收到的 SAML 身份验证响应 (断言) 的密钥。元数据文件必须采用不含字节顺序标记 (BOM) 的 UTF-8 格式编码。要删除 BOM，您可以使用 Notepad ++ 等文本编辑工具以 UTF-8 格式对文件进行编码。

作为 SAML 元数据文档的一部分，X.509 证书必须使用长度至少为 1024 位的密钥。此外，x.509 证书也不能有任何重复的扩展名。您可以使用扩展程序，但扩展程序只能在证书中显示一次。如果 x.509 证书不符合任一条件，则 IdP 将创建失败，并返回 "Unable to parse metadata" 这一错误消息。

错误：指定的提供商不存在。

如果您在 SAML 断言中指定的提供商名称与在 IAM 中配置的提供商名称不匹配，则可能会出现该错误。有关查看提供商名称的更多信息，请参阅[创建 IAM SAML 身份提供程序 \(p. 188\)](#)。

错误：请求的 DurationSeconds 超过为该角色设置的 MaxSessionDuration。

如果从 Amazon CLI 或 API 中担任角色，则可能会出现该错误。

在使用 [assume-role-with-saml](#) CLI 或 [AssumeRoleWithSAML](#) API 操作担任角色时，您可以为 DurationSeconds 参数指定一个值。您可以指定 900 秒 (15 分钟) 到角色的最大会话持续时间设置之间的值。如果指定的值高于该设置，操作将失败。例如，如果您指定的会话持续时间为 12 小时，但管理员设置的最大会话持续时间为 6 小时，您的操作将失败。要了解如何查看您的角色的最大值，请参阅[查看角色的最大会话持续时间设置 \(p. 238\)](#)。

如何在浏览器中查看 SAML 响应以排查问题

以下过程介绍了在排除 SAML 2.0 相关问题时，如何在浏览器中查看服务提供商的 SAML 响应。

对于所有浏览器，请转到您可以重现问题的页面。然后，针对相应的浏览器执行以下步骤：

主题

- [Google Chrome \(p. 894\)](#)
- [Mozilla Firefox \(p. 894\)](#)
- [Apple Safari \(p. 895\)](#)
- [如何处理 Base64 编码的 SAML 响应 \(p. 895\)](#)

Google Chrome

在 Chrome 中查看 SAML 响应

已使用版本 54.0.2840.87m 测试了这些步骤。如果您使用其他版本，则可能需要相应地调整步骤。

1. 按 F12 以启动开发人员控制台。
2. 选择 Network 选项卡，然后选择 Preserve log。
3. 重现问题。
4. 在开发人员控制台窗格中，查找 SAML Post。选择该行，然后查看底部的 Headers 选项卡。查找包含编码请求的 SAMLResponse 属性。

Mozilla Firefox

在 Firefox 中查看 SAML 响应

此过程已在 Mozilla Firefox 版本 37.0.2 上进行了测试。如果您使用其他版本，则可能需要相应地调整步骤。

1. 按 F12 以启动开发人员控制台。
2. 在开发人员工具窗口的右上角，单击选项（小齿轮图标）。在 Common Preferences 下，选择 Enable persistent logs。
3. 选择 Network 选项卡。
4. 重现问题。
5. 在表中查找 POST SAML。选择该行。在右侧的 Form Data 窗口中，选择 Params 选项卡并查找 SAMLResponse 元素。

Apple Safari

在 Safari 中查看 SAML 响应

这些步骤已使用版本 8.0.6 (10600.6.3) 进行了测试。如果您使用其他版本，则可能需要相应地调整步骤。

1. 在 Safari 中启用 Web Inspector。打开 Preferences 窗口，选择 Advanced 选项卡，然后选择 Show Develop menu in the menu bar。
2. 现在您可以打开 Web Inspector。单击 Develop，然后选择 Show Web Inspector。
3. 选择 Resources 选项卡。
4. 重现问题。
5. 查找 saml-signin.aws.amazon.com 请求。
6. 向下滚动，查找名为 Request Data 的 SAMLResponse。关联值为 Base64 编码的响应。

如何处理 Base64 编码的 SAML 响应

在浏览器中找到 Base64 编码的 SAML 响应元素之后，复制这些元素并使用您偏好的 Base-64 解码工具来提取带有 XML 标签的响应。

安全提示

由于您查看的 SAML 响应数据可能包含敏感安全数据，我们建议您不要使用在线 base64 解码程序。而是使用安装在本地计算机上、不会通过网络发送 SAML 数据的工具。

适用于 Windows 系统的内置选项 (PowerShell)：

```
PS C:\> [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("base64encodedtext"))
```

适用于 MacOS 和 Linux 系统的内置选项：

```
$ echo "base64encodedtext" | base64 --decode
```

Amazon Identity and Access Management 的参考信息

使用本节中的主题查找 IAM 和 Amazon STS 各方面的详细参考资料。

主题

- [IAM 标识符 \(p. 896\)](#)
- [IAM 与 Amazon STS 配额、名称要求和字符限制 \(p. 902\)](#)
- [使用 IAM 的 Amazon 服务 \(p. 908\)](#)
- [IAM JSON 策略参考 \(p. 933\)](#)

IAM 标识符

IAM 针对用户、用户组、策略、角色及服务器证书使用几个不同的标识符。本部分主要介绍标识符及您何时使用每个标识符。

主题

- [易记名称和路径 \(p. 896\)](#)
- [IAM ARN \(p. 896\)](#)
- [唯一标识符 \(p. 900\)](#)

易记名称和路径

在您创建用户、角色、用户组或策略时，或在您上传服务器证书时，您可为其取一个易于识别的名称。例如 Bob、TestApp1、Developers、ManageCredentialsPermissions 或 ProdServerCert。

如果您使用 IAM API 或 Amazon Command Line Interface (Amazon CLI) 创建 IAM 资源，则可以添加可选路径。您可以使用单个路径，或嵌套多个路径，就像它们是文件夹结构一样。例如，可使用嵌套路径 `/division_abc/subdivision_xyz/product_1234/engineering/` 以匹配贵公司的企业结构。您可以随后创建策略以允许该路径的所有用户访问策略模拟器 API。要查看该策略，请参阅 [IAM：基于用户路径访问策略模拟器 API \(p. 424\)](#)。有关如何指定友好名称的信息，请参阅 [用户 API 文档](#)。有关如何使用路径的其他示例，请参阅 [IAM ARN \(p. 896\)](#)。

使用 Amazon CloudFormation 创建资源时，您可以为用户、用户组和角色以及客户托管策略指定路径。

如果您的用户和用户组位于同一路径中，IAM 不会自动将该用户置于该用户组中。例如，您可创建一个 Developers 组，并指定其路径为 `/division_abc/subdivision_xyz/product_1234/engineering/`。如果您创建了一个名为 Bob 的用户并向其添加了相同的路径，则这不会自动将 Bob 归入开发人员用户组。IAM 不会基于路径而强行在用户或用户组之间划分边界。具有不同路径的用户可以使用相同的资源（假设他们已获权使用这些资源）。Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 [IAM 与 Amazon STS 配额、名称要求和字符限制 \(p. 902\)](#)。

IAM ARN

大多数资源都有易记名称（例如，名为 Bob 的用户或名为 Developers 的用户组）。不过，权限策略语言要求您使用以下 Amazon Resource Name (ARN) 格式指定资源。

```
arn:partition:service:region:account:resource
```

其中：

- `partition` 用于标识资源所在的分区。对于标准 Amazon 区域，分区是 `aws`。如果资源位于其他分区，则分区是 `aws-partitionname`。例如，中国（北京）区域中的资源的分区为 `aws-cn`。您不能在不同分区的账户之间委派访问权限 (p. 274)。
- `service` 标识 Amazon 产品。IAM 资源始终使用 `iam`。
- `region` 标识资源的区域。对于 IAM 资源，它始终保持空白。
- `account` 指定没有连字符的 Amazon 账户 ID。
- `resource` 按名称标识特定资源。

您可以使用以下语法指定 IAM 和 Amazon STS ARN。由于 IAM 资源是全球资源，因此，ARN 的区域部分是空的。

语法：

```
arn:aws:iam::account:root
arn:aws:iam::account:user/user-name-with-path
arn:aws:iam::account:group/group-name-with-path
arn:aws:iam::account:role/role-name-with-path
arn:aws:iam::account:policy/policy-name-with-path
arn:aws:iam::account:instance-profile/instance-profile-name-with-path
arn:aws:sts::account:federated-user/user-name
arn:aws:sts::account:assumed-role/role-name/role-session-name
arn:aws:iam::account:mfa/virtual-device-name-with-path
arn:aws:iam::account:u2f/u2f-token-id
arn:aws:iam::account:server-certificate/certificate-name-with-path
arn:aws:iam::account:saml-provider/provider-name
arn:aws:iam::account:oidc-provider/provider-name
```

以下示例中有很多在 ARN 的资源部分中包含路径。不能在 Amazon Web Services Management Console 中创建或操作路径。要使用路径，必须通过使用 Amazon API、Amazon CLI 或 Tools for Windows PowerShell 与资源相结合。

示例：

```
arn:aws:iam::123456789012:root
arn:aws:iam::123456789012:user/JohnDoe
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/JaneDoe
arn:aws:iam::123456789012:group/Developers
arn:aws:iam::123456789012:group/division_abc/subdivision_xyz/product_A/Developers
arn:aws:iam::123456789012:role/S3Access
arn:aws:iam::123456789012:role/application_abc/component_xyz/RDSAccess
arn:aws:iam::123456789012:role/aws-service-role/access-analyzer.amazonaws.com/
AWSServiceRoleForAccessAnalyzer
arn:aws:iam::123456789012:role/service-role/QuickSightAction
arn:aws:iam::123456789012:policy/UsersManageOwnCredentials
arn:aws:iam::123456789012:policy/division_abc/subdivision_xyz/UsersManageOwnCredentials
arn:aws:iam::123456789012:instance-profile/Webserver
arn:aws:sts::123456789012:federated-user/JohnDoe
arn:aws:sts::123456789012:assumed-role/Accounting-Role/JaneDoe
arn:aws:iam::123456789012:mfa/JaneDoeMFA
arn:aws:iam::123456789012:u2f/user/JohnDoe/default (U2F security key)
arn:aws:iam::123456789012:server-certificate/ProdServerCert
arn:aws:iam::123456789012:server-certificate/division_abc/subdivision_xyz/ProdServerCert
arn:aws:iam::123456789012:saml-provider/ADFSPProvider
arn:aws:iam::123456789012:oidc-provider/GoogleProvider
```

以下示例提供了更详细的信息，帮助您了解不同类型 IAM 和 Amazon STS 资源的 ARN 格式。

- 账户中的 IAM 用户：

Note

每个 IAM 用户名都是唯一的，不区分大小写。

```
arn:aws:iam::123456789012:user/JohnDoe
```

- 具有反映组织结构图的路径的其他用户：

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/JaneDoe
```

- 对于 IAM 用户组：

```
arn:aws:iam::123456789012:group/Developers
```

- 具有路径的 IAM 用户组：

```
arn:aws:iam::123456789012:group/division_abc/subdivision_xyz/product_A/Developers
```

- IAM 角色：

```
arn:aws:iam::123456789012:role/S3Access
```

- [服务相关角色 \(p. 159\)](#)：

```
arn:aws:iam::123456789012:role/aws-service-role/access-analyzer.amazonaws.com/  
AWSServiceRoleForAccessAnalyzer
```

- [服务角色 \(p. 159\)](#)：

```
arn:aws:iam::123456789012:role/service-role/QuickSightAction
```

- 托管策略：

```
arn:aws:iam::123456789012:policy/ManageCredentialsPermissions
```

- 可与 EC2 实例关联的实例配置文件：

```
arn:aws:iam::123456789012:instance-profile/Webserver
```

- 在 IAM 中识别为“Paulo”的联合身份用户：

```
arn:aws:sts::123456789012:federated-user/Paulo
```

- 担任“Accounting-Role”且角色会话名称为“Mary”的人的活动会话：

```
arn:aws:sts::123456789012:assumed-role/Accounting-Role/Mary
```

- 分配给名为 Jorge 的用户的多重身份验证设备：

```
arn:aws:iam::123456789012:mfa/Jorge
```

- 服务器证书：

```
arn:aws:iam::123456789012:server-certificate/ProdServerCert
```

- 具有反映组织结构图的路径的服务器证书：


```
arn:aws:iam::123456789012:server-certificate/division_abc/subdivision_xyz/ProdServerCert
```

- 身份提供程序 (SAML 和 OIDC) :

```
arn:aws:iam::123456789012:saml-provider/ADFSPProvider
arn:aws:iam::123456789012:oidc-provider/GoogleProvider
```

另一个重要的 ARN 是根用户 ARN。虽然这不是一个 IAM 资源，但您应熟悉此 ARN 的格式。它通常用于基于资源的策略的 [Principal 元素 \(p. 936\)](#)。

- Amazon 账户会显示以下内容：

```
arn:aws:iam::123456789012:root
```

以下示例说明了您可向 Richard 分配的策略，该策略允许其管理访问密钥。请注意，这里的资源即是 IAM 用户 Richard。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ManageRichardAccessKeys",
      "Effect": "Allow",
      "Action": [
        "iam:*AccessKey*",
        "iam:GetUser"
      ],
      "Resource": "arn:aws:iam::*:user/division_abc/subdivision_xyz/Richard"
    },
    {
      "Sid": "ListForConsole",
      "Effect": "Allow",
      "Action": "iam:ListUsers",
      "Resource": "*"
    }
  ]
}
```

Note

使用 ARN 在 IAM policy 中标识资源时，您可以包含策略变量。策略变量可以包含表示运行时信息（如用户名称）的占位符，作为 ARN 的一部分。有关更多信息，请参阅 [IAM policy 元素：变量和标签 \(p. 964\)](#)

在 ARN 中使用通配符和路径

您可以在 ARN 的 *resource* 部分中使用通配符，以指定多个用户、用户组或策略。例如，如要指定使用 product_1234 的所有用户，您应使用：

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/product_1234/*
```

如果您的用户名称以字符串 app_ 开头，可以通过以下 ARN 引用所有这些用户。

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/product_1234/app_*
```

要指定您 Amazon 账户中的所有用户、用户组或策略，请分别在 ARN 的 `user/`、`group/` 或 `policy/` 部分之后使用通配符。

```
arn:aws:iam::123456789012:user/*
arn:aws:iam::123456789012:group/*
arn:aws:iam::123456789012:policy/*
```

如果为用户 `arn:aws:iam::111122223333:user/*` 指定以下 ARN，则它与以下两个示例相匹配。

```
arn:aws:iam::111122223333:user/JohnDoe
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/JaneDoe
```

但是，如果您为用户 `arn:aws:iam::111122223333:user/division_abc*` 指定以下 ARN，则它匹配第二个示例，但不匹配第一个示例。

```
arn:aws:iam::111122223333:user/JohnDoe
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/JaneDoe
```

请勿在 ARN 的 `user/`、`group/` 或 `policy/` 部分中使用通配符。例如，IAM 不允许执行以下操作：

```
arn:aws:iam::123456789012:u*
```

Example 针对基于项目的用户组使用路径和 ARN 的示例

不能在 Amazon Web Services Management Console 中创建或操作路径。要使用路径，必须通过使用 Amazon API、Amazon CLI 或 Tools for Windows PowerShell 与资源相结合。

在此示例中，Marketing_Admin 用户组中的 Jules 在 `/marketing/` 路径内创建一个基于项目的用户组。Jules 并将公司内不同部门的用户分配至该用户组。此示例说明，用户的路径与用户所在的用户组并无关系。

市场部组负责一款即将上市的新产品，因此 Jules 在 `/marketing/` 路径内创建了一个新的用户组，并命名为 `Widget_Launch`。随后，Jules 将下列策略分配至此用户组，而该策略将允许此用户组访问 `example_bucket`（专用于此款新产品）部分中的对象。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::example_bucket/marketing/newproductlaunch/widget/*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket*",
      "Resource": "arn:aws:s3:::example_bucket",
      "Condition": {"StringLike": {"s3:prefix": "marketing/newproductlaunch/widget/*"}}
    }
  ]
}
```

最后，Jules 将负责此款新产品的用户分配至该用户组。其中包括 `/marketing/` 路径中的 Patricia 和 Eli。另外，还包括 `/sales/` 路径中的 Chris 和 Chloe，及 `/legal/` 路径中的 Alice 和 Jim。

唯一标识符

当 IAM 创建用户、用户组、角色、策略、实例配置文件或服务器证书时，它会为每个资源分配一个唯一 ID。该唯一 ID 如下所示：

AIDAJQABLZS4A3QDU576Q

大多数情况下，在使用 IAM 资源时，您使用易于识别的名称和 [ARN \(p. 896\)](#)。这样，您就不需要知道特定资源的唯一 ID。不过有时，当实际情况不允许使用好友名称时，唯一 ID 可能非常有用。

有一个示例涉及到对您的 Amazon 账户中的易记名称的重复使用。在您的账户中，用户、用户组、角色或策略的易记名称必须是唯一的。例如，您可以创建一个名为 John 的 IAM 用户。公司使用 Amazon S3 并有一个存储桶，其中包含每位员工的文件夹。IAM 用户 John 是名为 User-S3-Access 的 IAM 用户组的成员，该用户组的权限仅允许用户访问存储桶中其自己的文件夹。有关可以通过何种方式创建基于身份的策略以允许 IAM 用户使用其易记名称访问 S3 中其自己的存储桶对象的示例，请参阅 [Amazon S3：允许 IAM 用户以编程方式和在控制台中访问其 S3 主目录 \(p. 434\)](#)。

假设名为 John 的员工从贵公司离职，并且您删除了名为 John 的相应 IAM 用户。但后来另一位名为 John 的员工入职贵公司，并且您创建了一个名为 John 的新 IAM 用户。您将这位名为 John 的新 IAM 用户添加到了现有 IAM 用户组 User-S3-Access 中。如果与该用户组关联的策略指定易记的 IAM 用户名 John，则该策略将允许新 John 访问前一个 John 留下的信息。

一般情况下，建议您为策略中的资源指定 ARN，而不是其唯一 ID。但是，每位 IAM 用户都具有一个唯一 ID，即使您创建的新 IAM 用户重复使用之前已删除的易记名称，也是如此。在该示例中，旧 IAM 用户 John 和新 IAM 用户 John 具有不同的唯一 ID。您可以创建基于资源的策略，这些策略按唯一 ID 而不仅仅是用户名授予访问权限。这样做可以减少您无意中向员工授予了不应具备的信息访问权限的机会。

以下示例说明了如何在基于资源策略的 [Principal 元素 \(p. 936\)](#) 中指定唯一 ID。

```
"Principal": {
  "AWS": [
    "arn:aws:iam::111122223333:role/role-name",
    "AIDACKCEVSQ6C2EXAMPLE",
    "AROADBQP57FF2AEXAMPLE"
  ]
}
```

以下示例说明了如何在策略的 [Condition 元素 \(p. 948\)](#) 中使用全局条件键 [aws:userid \(p. 1012\)](#) 指定唯一 ID。

```
"Condition": {
  "StringLike": {
    "aws:userId": [
      "AIDACKCEVSQ6C2EXAMPLE",
      "AROADBQP57FF2AEXAMPLE:role-session-name",
      "AROAI234567890EXAMPLE:*",
      "111122223333"
    ]
  }
}
```

用户 ID 发挥作用的另一个示例是您维护自己的 IAM 用户或角色信息数据库（或其他存储）。唯一 ID 可为您创建的每个 IAM 用户或角色提供唯一标识符。当您有 IAM 用户或角色重复使用某个名称时也是如此，如前面的示例所示。

了解唯一 ID 前缀

IAM 使用以下前缀来指示每个唯一 ID 适用于的资源类型。

Prefix	资源类型
ABIA	Amazon STS 服务持有者令牌 (p. 336)
ACCA	上下文特定凭证

Prefix	资源类型
AGPA	用户组
AIDA	IAM 用户
AIPA	Amazon EC2 实例配置文件
AKIA	访问密钥
ANPA	托管策略
ANVA	托管策略中的版本
APKA	公有密钥
AROA	角色
ASCA	证书
ASIA	临时 (Amazon STS) 访问密钥 ID 使用此前缀，但仅在与秘密访问密钥和会话令牌结合使用时才具有唯一性。

获取唯一标识符

无法通过 IAM 控制台获得 IAM 资源的唯一 ID。要获得唯一 ID，您可以使用下面的 Amazon CLI 命令或 IAM API 调用。

Amazon CLI:

- [get-caller-identity](#)
- [get-group](#)
- [get-role](#)
- [get-user](#)
- [get-policy](#)
- [get-instance-profile](#)
- [get-server-certificate](#)

IAM API :

- [GetCallerIdentity](#)
- [GetGroup](#)
- [GetRole](#)
- [GetUser](#)
- [GetPolicy](#)
- [GetInstanceProfile](#)
- [GetServerCertificate](#)

IAM 与 Amazon STS 配额、名称要求和字符限制

Amazon Identity and Access Management (IAM) 和 Amazon Security Token Service (STS) 具有限制对象大小的配额。这会影响您命名对象的方式、可以创建的对象数量以及传递对象时可以使用的字符数。

Note

要获取有关 IAM 使用量和配额的账户级别信息，请使用 [GetAccountSummary](#) API 操作或 [get-account-summary](#) Amazon CLI 命令。

IAM 名称要求

IAM 名称具有以下要求和限制：

- 策略文档只能包含以下 Unicode 字符：水平制表符 (U+0009)、换行符 (U+000A)、回车符 (U+000D) 以及 U+0020 到 U+00FF 范围内的字符。
- 用户、组、角色、策略、实例配置文件以及服务器证书的名称必须是字母数字，包括以下常用字符：加号 (+)、等号 (=)、逗号 (,)、句号 (.)、at 符 (@)、下划线 (_) 和连字符 (-)。
- 账户中的用户、组、角色和实例配置文件的名称必须唯一。上述名称不区分大小写，例如，您无法同时创建名为 **ADMINS** 和 **admins** 的组。
- 第三方用于担任角色的外部 ID 值必须至少包含 2 个字符，最多包含 1224 个字符。该值必须是字母数字，没有空格。它还可以包含以下符号：加号 (+)、等号 (=)、逗号 (,)、句点 (.)、@ 符号、冒号 (:)、正斜杠 (/) 和连字符 (-)。有关外部 ID 的更多信息，请参阅 [如何在向第三方授予对 Amazon 资源的访问权时使用外部 ID](#) (p. 164)。
- 路径名称必须以正斜杠 (/) 开始和结束。
- 对于[内联策略](#) (p. 364)嵌入到的用户、组或角色，这些策略的名称必须是唯一的。这些名称可以包含任何基本拉丁语 (ASCII) 字符，但以下保留字符除外：反斜杠 (\)、正斜杠 (/)、星号 (*)、问号 (?) 和空格。根据 [RFC 3986 第 2.2 部分](#)，这些字符是保留字符。
- 用户密码（登录配置文件）可以包含任何基本拉丁语 (ASCII) 字符。
- 所有 Amazon 产品的 Amazon 账户 ID 别名都必须唯一的，必须是遵循 DNS 命名约定的字母数字。别名必须是小写字母，不能以连字符开头或结尾，不能连续使用两个连字符，也不能是 12 个纯数字。

要获取基本拉丁语 (ASCII) 字符列表，请转到 [Library of Congress Basic Latin \(ASCII\) Code Table](#)。

IAM 对象配额

配额，也称为 Amazon 中的限制，是 Amazon 账户中资源、操作和项目的最大值。使用 Service Quotas 管理 IAM 配额。您可以请求提高可调整 IAM 配额的默认配额。最大 [maximum quota](#) 的请求将自动得到批准，并在几分钟内完成。

要请求提高配额，请登录到 Amazon Web Services Management Console 并在以下位置打开 Service Quotas 控制台：<https://console.aws.amazon.com/servicequotas/>。在导航窗格中，选择 Amazon services (Amazon 服务)。在导航栏中，选择 US East (N. Virginia) 区域。然后搜索 **IAM**。选择 Amazon Identity and Access Management (IAM)，选择配额，然后按照说明请求增加配额。有关更多信息，请参阅《服务限额用户指南》中的[请求增加配额](#)。

要查看有关如何使用 Service Quotas 控制台请求增加 IAM 配额的示例，请观看以下视频。

[使用 Service Quotas 控制台请求增加 IAM 配额。](#)

以下配额是可调的。

IAM 实体的默认配额

资源	默认配额	最大配额
一个 Amazon 账户中的客户托管策略数	1500	5000

资源	默认配额	最大配额
一个 Amazon 账户中的组数	300	500
一个 Amazon 账户中的实例配置文件数	1000	5000
附加到一个 IAM 角色的托管策略数	10	20
附加到一个 IAM 用户的托管策略数	10	20
角色信任策略长度	2048 个字符	4096 个字符
一个 Amazon 账户中的角色数	1000	5000
存储在一个 Amazon 账户中的服务器证书数	20	1000
一个 Amazon 账户中的虚拟 MFA 设备数 (已分配或未分配)	等于账户的用户配额	不适用

您不能请求增加以下配额。

IAM 实体的配额

资源	配额
分配给一个 IAM 用户的访问密钥数	2
分配给 Amazon Web Services 账户 根用户的访问密钥数	2
一个 Amazon 账户的别名数	1
IAM 用户可加入的组数	10
组中的 IAM 用户	等于账户的用户配额
与一个 IAM SAML 提供商对象关联的身份提供程序 (IdP) 数	10
每个 SAML 提供商的键数	10
一个 IAM 用户的登录配置文件数	1
附加到一个 IAM 组的托管策略数	10
每个 Amazon 账户的 OpenId Connect 身份提供程序	100
IAM 用户的权限边界	1
IAM 角色的权限边界	1
一个 IAM 用户使用的 MFA 设备数	1
Amazon Web Services 账户 根用户使用的 MFA 设备数	1

资源	配额
一个实例配置文件中的角色数	1
一个 Amazon 账户中的 SAML 提供商数	100
分配给一个 IAM 用户的签名证书数	2
分配给一个 IAM 用户的 SSH 公有密钥数	5
可以附加到客户托管策略的标签	50
可以附加到 SAML 身份提供程序的标签	50
可以附加到服务器证书的标签	50
可以附加到虚拟 MFA 设备的标签	50
可以附加到实例配置文件的标签	50
可附加到 IAM 角色的标签	50
可附加到 IAM 用户的标签	50
可以附加到 Open ID Connect (OIDC) 身份提供程序的标签	50
一个 Amazon 账户中的用户数	5000 (如果您需要添加大量用户，建议您使用 临时安全凭证 (p. 300))。
可存储的一个托管策略的版本数	5

IAM Access Analyzer 配额

有关 IAM Access Analyzer 配额，请参阅 [IAM Access Analyzer Quotas](#)。

IAM 和 STS 字符限制

以下是 IAM 和 Amazon STS 的最大字符数和大小限制。您不能请求提高以下限制。

描述	限制
Amazon 账户 ID 的别名	3-63 个字符
对于 内联策略 (p. 364)	您可以向 IAM 用户、角色或组添加所需数量的内联策略。但是，每个实体的总聚合策略大小（所有内联策略的总大小）不能超过以下限制： - 用户策略大小不得超过 2048 个字符。 - 角色策略大小不得超过 10240 个字符。 - 组策略大小不得超过 5120 个字符。 Note 在计算策略大小时，IAM 不会将空格计入这些限制。

描述	限制
对于 托管策略 (p. 364)	每个托管策略的大小不能超过 6,144 个字符。 Note 在计算策略大小时，IAM 不会将空格计入这一限制。
组名	128 个字符
实例配置文件名称	128 个字符
登录配置文件的密码	1-128 个字符
路径	512 个字符
策略名称	128 个字符
角色名称	64 个字符 Important 如果您通过 Amazon Web Services Management Console 中的切换角色功能使用角色，则组合的 Path 和 RoleName 不能超过 64 个字符。
角色会话持续时间	12 小时 从 Amazon CLI 或 API 中担任角色时，您可以使用 duration-seconds CLI 参数或 DurationSeconds API 参数请求更长的角色会话。您可以指定 900 秒（15 分钟）到角色的最大会话持续时间设置之间的值，该设置的范围是 1 - 12 小时。如果未指定 DurationSeconds 参数值，您的安全凭证的有效期为 1 小时。在控制台内切换角色的 IAM 用户被授予最大会话持续时间或 IAM 用户会话中的剩余时间（以较少者为准）。最大会话持续时间设置不限制 Amazon 服务建立的会话。要了解如何查看您的角色的最大值，请参阅查看角色的最大会话持续时间设置 (p. 238)。
角色会话名称	64 个字符

描述	限制
角色会话策略 (p. 357)	- 传递的 JSON 策略文档和所有传递的托管策略 ARN 字符加在一起的大小不能超过 2,048 个字符。 - 创建会话时，最多可传递 10 个托管策略 ARN。 - 在以编程方式为角色或联合身份用户创建临时会话时，您只能传递一个 JSON 策略文档。 - 此外，Amazon 转换会将传递的会话策略和会话标签压缩为具有单独限制的打包二进制文件格式。PackedPolicySize 响应元素指示您请求的策略和标签接近大小上限的程度，以百分比来表示。 - 建议您使用 Amazon CLI 或 Amazon API 传递会话策略。Amazon Web Services Management Console 可能会将其他控制台会话信息添加到打包策略中。
角色会话标签 (p. 293)	- 会话标签必须满足 128 个字符的标签键限制和 256 个字符的标签值限制。 - 您最多可以传递 50 个会话标签。 - Amazon 转换会将传递的会话策略和会话标签压缩为具有单独限制的打包二进制格式。您可以使用 Amazon CLI 或 Amazon API 传递会话标签。PackedPolicySize 响应元素指示您请求的策略和标签接近大小上限的程度，以百分比来表示。
经过 base64 编码的 SAML 身份验证	100000 个字符 此字符限制适用于 <code>assume-role-with-saml</code> CLI 或 <code>AssumeRoleWithSAML</code> API 操作。
标记密钥	128 个字符 此字符限制适用于 IAM 资源的标签和会话标签 (p. 293)。
标记值	256 个字符 此字符限制适用于 IAM 资源的标签和会话标签 (p. 293)。 标签值可以为空，这意味着标签值的长度可以为 0 个字符。

描述	限制
IAM 创建的唯一 ID	128 个字符。例如： • 以 AIDA 开头的用户 ID • 以 AGPA 开头的组 ID • 以 AROA 开头的角色 ID • 以 ANPA 开头的托管策略 ID • 以 ASCA 开头的服务器证书 ID Note 这不是一个详尽的列表，也不保证某种类型的 ID 仅以指定的字母组合开始。
用户名	64 个字符

使用 IAM 的 Amazon 服务

下面列出的 Amazon 服务根据其 [Amazon 产品类别分组](#)，包含所支持的 IAM 功能的信息：

- 服务 - 您可以选择一项服务的名称，以查看有关该服务的 IAM 授权和访问权限的 Amazon 文档。
- 操作 - 您可以指定策略中的各项操作。如果服务不支持此功能，则 [visual editor \(p. 439\)](#) (可视化编辑器) 中将选中 All actions (所有操作)。在 JSON 策略文档中，您必须在 * 元素中使用 Action。有关每个服务中的操作列表，请参阅 [Amazon 服务的操作、资源和条件键](#)。
- 资源级权限 - 您可以使用 [ARN \(p. 896\)](#) 在策略中指定各个资源。如果服务不支持此功能，则 [policy visual editor \(p. 439\)](#) (策略可视化编辑器) 中将选中 All resources (所有资源)。在 JSON 策略文档中，您必须在 * 元素中使用 Resource。某些操作 (如 List* 操作) 不支持指定 ARN，因为它们被设计为返回多个资源。如果某项服务只针对部分资源支持此功能，将在表格中用 Partial (部分) 标明。有关更多信息，请参阅该服务的文档。
- 基于资源的策略 - 您可以将基于资源的策略附加到服务中的某项资源。基于资源的策略包含 Principal 元素，用于指定可以访问此资源的 IAM 身份。有关更多信息，请参阅[基于身份的策略和基于资源的策略 \(p. 379\)](#)。
- ABAC (基于标签的授权) - 要利用标签来控制访问权限，您需要在策略的 [条件元素](#) 中使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键提供标签信息。如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为 Yes (是)。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为 Partial (部分)。有关基于标签等属性定义权限的更多信息，请参阅[什么是适用于 Amazon 的 ABAC ? \(p. 12\)](#)。要查看有关 ABAC 设置步骤的教程，请参阅[使用基于属性的访问权限控制 \(ABAC\)](#)。
- Temporary credentials (临时凭证) - 您可以使用在通过 IAM Identity Center 登录时获得的短期凭证、在控制台中切换角色或在 Amazon CLI 或 Amazon API 中使用 Amazon STS 生成的临时凭证。只有在使用长期 IAM 用户凭证时，您才能访问带有否值的服务。这包括用户名和密码或用户访问密钥。有关更多信息，请参阅[IAM 临时安全凭证 \(p. 300\)](#)。
- 服务相关角色 - [服务相关角色 \(p. 159\)](#) 是一种特殊类型的服务角色，可授予服务代表您访问其他服务中的资源的权限。选择 Yes 链接可查看文档，了解支持这些角色的服务。此列不指示服务是否使用标准服务角色。有关更多信息，请参阅[使用服务相关角色 \(p. 208\)](#)。
- 更多信息 - 如果服务不能完全支持某项功能，您可以检查该条目的脚注，查看限制以及相关信息的链接。

计算服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon App Runner	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Batch	✔ 是	⚠ 部分	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Elastic Compute Cloud (Amazon EC2)	✔ 是	⚠ 部分	✘ 不支持	✔ 是	✔ 是	⚠ 部分 ¹
Amazon EC2 Auto Scaling	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
EC2 Image Builder	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon EC2 Instance Connect	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Elastic Beanstalk	✔ 是	⚠ 部分	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Elastic Inference	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Elastic Load Balancing	✔ 是	⚠ 部分	✘ 不支持	⚠ 部分	✔ 是	✔ 是
Amazon Lambda	✔ 是	✔ 是	✔ 是	⚠ 部分 ²	✔ 是	⚠ 部分 ³
Amazon Lightsail	✔ 是	⚠ 部分 ⁴	✘ 不支持	⚠ 部分 ⁴	✔ 是	✔ 是
Amazon Outposts	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon 回收站	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Serverless Application Repository	✔ 是	✔ 是	✔ 是	✘ 不支持	✔ 是	✘ 不支持

¹ Amazon EC2 仅针对以下功能支持服务相关角色：[竞价型实例请求](#)、[竞价型实例集请求](#)、[Amazon EC2 Fleet](#) 和 [Windows 实例的快速启动](#)。

² Amazon Lambda 支持对使用 Lambda 函数作为所需资源的 API 操作进行基于属性的访问权限控制 (ABAC)。不支持层、事件源映射和代码签名配置资源。

³ Amazon Lambda 不具有服务相关角色，但 Lambda@Edge 具有。有关更多信息，请参阅《Amazon CloudFront 开发人员指南》中的 [Lambda@Edge 的服务相关角色](#)。

⁴ Amazon Lightsail 部分支持资源级权限和 ABAC。有关更多信息，请参阅 [Amazon Lightsail 的操作、资源和条件键](#)。

容器服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon App Runner	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Elastic Container Registry (Amazon ECR)	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是
Amazon Elastic Container Registry 公有 (Amazon ECR 公有)	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Elastic Container Service (Amazon ECS)	✔ 是	⚠ 部分 ¹	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Elastic Kubernetes Service (Amazon EKS)	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是

¹ 仅某些 Amazon ECS 操作支持资源级权限。

存储服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Backup	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是
Amazon Backup 网关	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Backup 存储	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Elastic Block Store (Amazon EBS)	✔ 是	⚠ 部分	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon 弹性灾难恢复	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Elastic File System (Amazon EFS)	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon FSx	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Import/Export	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon S3 Glacier	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是	✘ 不支持
Amazon Simple Storage Service (Amazon S3)	✔ 是	✔ 是	✔ 是	⚠ 部分 ¹	✔ 是	⚠ 部分 ²
Amazon Outposts 上的 Amazon Simple Storage Service (Amazon S3)	✔ 是	✔ 是	✔ 是	✘ 不支持	✔ 是	✘ 不支持
Amazon Simple Storage Service (Amazon S3) 对象 Lambda	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Snow Device Management	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Snowball	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Snowball 边缘	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Storage Gateway	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

¹ Amazon S3 仅为对象资源支持基于标签的授权。

² Amazon S3 支持针对 Amazon S3 Storage Lens 的服务关联角色。

数据库服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Database Query Metadata Service	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon DynamoDB	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon DynamoDB Accelerator (DAX)	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon ElastiCache	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Keyspaces (for Apache Cassandra)	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon MediaImport	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✘ 不支持	✘ 不支持
Amazon MemoryDB for Redis	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Neptune	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon 性能详情	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Quantum Ledger Database (Amazon QLDB)	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Redshift	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Redshift 数据 API	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Redshift Serverless	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Relational Database Service (Amazon RDS)	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon RDS Data API	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon RDS IAM 身份验证	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon SimpleDB	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon SQL Workbench	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Timestream	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

开发人员工具服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Cloud9	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是
Amazon Web Services 云 Control API	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon CloudShell	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon CodeArtifact	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✗ 不支持
Amazon CodeBuild	✓ 是	✓ 是	✓ 是 ¹	⚠ 部分 ²	✓ 是	✗ 不支持
Amazon CodeCommit	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CodeDeploy	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CodeDeploy 安全主机命令服务	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon CodePipeline	✓ 是	⚠ 部分	✗ 不支持	✓ 是	✓ 是	✗ 不支持
AWS CodeStar	✓ 是	⚠ 部分	✗ 不支持	✓ 是	✓ 是	✗ 不支持
AWS CodeStar 连接	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
AWS CodeStar 通知	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Fault Injection Simulator	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Microservice Extractor for .NET	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon X-Ray	✓ 是	⚠ 部分 ³	✗ 不支持	⚠ 部分 ⁴	✓ 是	✗ 不支持

¹ CodeBuild 支持使用 Amazon RAM 进行跨账户资源共享。

² CodeBuild 支持将 ABAC 用于基于项目的操作。

³ X-Ray 并非对所有操作均支持资源级权限。

⁴ X-Ray 支持针对组和采样规则实现基于标签的访问控制。

安全性、身份与合规性服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Artifact	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon 审计管理器	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Cloud Directory	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Cognito	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Cognito Sync	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon Cognito 用户池	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Detective	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Directory Service	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Firewall Manager	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	⚠ 部分
Amazon GuardDuty	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon IAM Identity Center (successor to Amazon Single Sign-On)	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
IAM Identity Center 目录	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
IAM Identity Center Identity Store	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Identity and Access Management (IAM)	✔ 是	✔ 是	⚠ 部分 ¹	⚠ 部分 ² (p. 388)	⚠ 部分 ³	✘ 不支持
Amazon Identity and Access Management 访问分析器	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	⚠ 部分

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Identity and Access Management Roles Anywhere	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Identity Sync	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Inspector Classic	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon Inspector	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Macie	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon 网络防火墙	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Resource Access Manager (Amazon RAM)	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Secrets Manager	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是	✘ 不支持
Amazon Security Hub	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Security Token Service (Amazon STS)	✔ 是	⚠ 部分 ⁴	✘ 不支持	✔ 是	⚠ 部分 ⁵	✘ 不支持
Amazon Shield	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon WAF	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon WAF Classic	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon WAF 区域性	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是

¹ IAM 仅支持一种类型的基于资源的策略（称为角色信任策略），这种策略附加到 IAM 角色。有关更多信息，请参阅[向用户授予切换角色的权限](#) (p. 239)。

² IAM 为大多数 IAM 资源支持基于标签的访问控制。有关更多信息，请参阅[标记 IAM 资源](#) (p. 276)。

³ 只能使用临时凭证调用 IAM 的一部分 API 操作。有关更多信息，请参阅[比较您的 API 选项](#)。

⁴ Amazon STS 没有“资源”，但允许以类似的方式限制用户访问。有关更多信息，请参阅[根据名称拒绝访问临时安全凭证](#)。

⁵ 仅 Amazon STS 的一部分 API 操作支持使用临时凭证进行调用。有关更多信息，请参阅[比较您的 API 选项](#)。

加密和 PKI 服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Certificate Manager (ACM)	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Private Certificate Authority (ACM PCA)	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是	✘ 不支持
Amazon CloudHSM	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Key Management Service (Amazon KMS)	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是
Amazon 签署人	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

机器学习服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon BugBust	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon CodeGuru Profiler	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon CodeGuru Reviewer	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Comprehend	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Comprehend Medical	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon DeepComposer	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon DeepLens	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon DeepRacer	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon DevOps Guru	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon Forecast	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Fraud Detector	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Ground Truth 标记	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon HealthLake	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Kendra	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Lex	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Lex V2	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是
Amazon Lookout for Equipment	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Lookout for Metrics	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Lookout for Vision	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Machine Learning	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Monitron	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Panorama	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Personalize	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Polly	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Rekognition	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon SageMaker	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	⚠ 部分 ¹
Amazon SageMaker Ground Truth Synthetic	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Textract	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Transcribe	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Translate	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持

¹ 服务相关角色目前可用于 SageMaker Studio 和 SageMaker 培训职位。

管理和治理服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Account Management	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Application Auto Scaling	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon AppConfig	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Auto Scaling	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon Chatbot	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon CloudFormation	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon CloudTrail	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon CloudWatch	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	⚠ 部分 ¹
Amazon CloudWatch Application Insights	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon CloudWatch Evidently	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon CloudWatch Logs	✔ 是	✔ 是	✔ 是	⚠ 部分	✔ 是	✔ 是
Amazon CloudWatch RUM	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon CloudWatch Synthetics	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Compute Optimizer	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon Config	✔ 是	⚠ 部分 ²	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Control Tower	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Data Lifecycle Manager	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Health API 和通知	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Launch Wizard	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon License Manager	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon License Manager 用户订阅	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Managed Grafana	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Managed Service for Prometheus	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Message Delivery Service	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon OpsWorks	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon OpsWorks 配置管理	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Organizations	✔ 是	✔ 是	✘ 不支持	✔ 是	✘ 不支持	✔ 是
Amazon Proton	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Resilience Hub	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Resource Groups	✔ 是	✔ 是	✘ 不支持	✔ 是	⚠ 部分 ³	✘ 不支持
Amazon Resource Groups Tagging API	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon RHEL 知识库门户	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Service Catalog	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Session Manager Message Gateway Service	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Systems Manager	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Systems Manager GUI Connect	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Systems Manager Incident Manager	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是
Amazon Systems Manager Incident Manager Contacts	✔ 是	✔ 是	✔ 是	✘ 不支持	✔ 是	✘ 不支持
Amazon 标签编辑器	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Trusted Advisor	⚠ 部分 ⁴	✔ 是	✘ 不支持	✘ 不支持	⚠ 部分	✔ 是
Amazon Well-Architected Tool	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
服务限额	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

¹ 无法使用 Amazon Web Services Management Console 创建 Amazon CloudWatch 服务相关角色，这些角色仅支持警报操作功能。

² 对于多账户多区域数据聚合和 Amazon Config 规则，Amazon Config 支持资源级权限。有关受支持的资源列表，请参阅 Amazon Config API 指南中的多账户多区域数据聚合部分与 [Amazon Config 规则](#) 部分。

³ 用户可以通过允许 Amazon Resource Groups 操作的策略代入角色。

⁴ 针对 Trusted Advisor 的 API 访问通过 Amazon Web Services Support API 执行，并受 Amazon Web Services Support IAM policy 的控制。

迁移和传输服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Application Discovery Service	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon Application Discovery Arsenal	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Application Migration Service	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Connector Service	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Transfer Family	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Database Migration Service	✔ 是	✔ 是	✘ 否 ¹	✔ 是	✔ 是	✘ 不支持
Amazon DataSync	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Mainframe Modernization	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Migration Hub	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon Migration Hub Orchestrator	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Migration Hub Refactor Spaces	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Migration Hub Strategy Recommendations	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon Server Migration Service	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✔ 是

¹ 您可以创建和修改附加到您创建的 Amazon KMS 加密密钥的策略，以加密迁移到支持的目标终端节点的数据。支持的目标终端节点包括 Amazon Redshift 和 Amazon S3。有关更多信息，请参阅 Amazon Database

Migration Service 用户指南中的[创建和使用 Amazon KMS 密钥以加密 Amazon Redshift 目标数据](#)和[创建 Amazon KMS 密钥以加密 Amazon S3 目标对象](#)。

移动服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Amplify	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Amplify 管理员	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Amplify UI Builder	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon AppSync	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Device Farm	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Location Service	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

网络和内容分发服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon API Gateway	✔ 是	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是
Amazon API Gateway 管理	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon API Gateway 管理第 2 版	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon App Mesh	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon App Mesh 预览	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon CloudFront	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	⚠ 部分 ¹

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Cloud Map	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Direct Connect	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Global Accelerator	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Network Manager	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是 ²
Amazon 私有 5G	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Route 53	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Route 53 域	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✘ 不支持	✘ 不支持
Amazon Route 53 Recovery 集群	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Route 53 恢复控制配置	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Route 53 Recovery 就绪性	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Route 53 Resolver	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Tiros API (用于 VPC Reachability Analyzer)	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✘ 不支持	✘ 不支持
Amazon Virtual Private Cloud (Amazon VPC)	✔ 是	⚠ 部分 ³	⚠ 部分 ⁴	✔ 是	✔ 是	⚠ 部分 ⁵

¹ Amazon CloudFront 不具有服务相关角色，但 Lambda@Edge 具有。有关更多信息，请参阅《Amazon CloudFront 开发人员指南》中的 [Lambda@Edge 的服务相关角色](#)。

² Amazon Cloud WAN 还支持服务相关角色。有关更多信息，请参阅《Amazon VPC Amazon Cloud WAN 指南》中的 [Amazon Cloud WAN 相关角色](#)。

³ 在 IAM 用户策略中，您不能限定为仅向某个 Amazon VPC 端点授予权限。包含 `ec2:*VpcEndpoint*` 或 `ec2:DescribePrefixLists` API 操作的任何 Action 元素都必须指定 `"Resource": "*"` 。有关更多信息，请参阅《Amazon PrivateLink 指南》中的 [VPC 终端节点和 VPC 终端节点服务的身份和访问管理](#)。

⁴ Amazon VPC 支持通过将单个资源策略附加到 VPC 端点来限制可通过该端点访问的内容。有关使用基于资源的策略控制特定 Amazon VPC 端点访问资源的更多信息，请参阅 Amazon PrivateLink 用户指南中的 [使用端点策略控制服务访问](#)。

⁵ Amazon VPC 不具有服务相关角色，但 Amazon Transit Gateway 具有。有关更多信息，请参阅《Amazon VPC Amazon Transit Gateway 用户指南》中的[使用中转网关的服务相关角色](#)。

媒体服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Elastic Transcoder	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Elemental Appliances and Software	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Elemental Appliances and Software 激活服务	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
AWS Elemental MediaConnect	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
AWS Elemental MediaConvert	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
AWS Elemental MediaLive	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
AWS Elemental MediaPackage	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	⚠ 部分 ¹
AWS Elemental MediaPackage VOD	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	⚠ 部分 ¹
AWS Elemental MediaStore	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是	✘ 不支持
AWS Elemental MediaTailor	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Elemental Support 案例	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Elemental Support 内容	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Interactive Video Service	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Interactive Video Service Chat	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Kinesis Video Streams	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Nimble Studio	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

¹ MediaPackage 支持将客户访问日志发布到 CloudWatch 的服务相关角色，但不支持其他 API 操作。

分析服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Athena	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon CloudSearch	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Data Exchange	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Data Pipeline	✔ 是	✘ 不支持	✘ 不支持	⚠ 部分	✔ 是	✘ 不支持
Amazon OpenSearch Service	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是
Amazon EMR	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon EMR on EKS	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon EMR Serverless	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon FinSpace	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Glue	✔ 是	✔ 是	✔ 是	⚠ 部分	✔ 是	✘ 不支持
Amazon Glue DataBrew	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Kinesis Data Analytics	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Kinesis Data Analytics V2	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Kinesis Data Firehose	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Kinesis Data Streams	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Lake Formation	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon Managed Streaming for Apache Kafka (MSK)	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Managed Streaming for Kafka Connect	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✔ 是
适用于 Amazon MSK 集群的 Apache Kafka API	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Managed Workflows for Apache Airflow	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon QuickSight	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

应用程序集成服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon AppFlow	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon EventBridge	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是	✘ 不支持
Amazon EventBridge Schemas	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是	✘ 不支持
Amazon MQ	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Simple Notification Service (Amazon SNS)	✔ 是	✔ 是	✔ 是	✔ 是	✔ 是	✘ 不支持
Amazon Simple Queue Service (Amazon SQS)	✔ 是	✔ 是	✔ 是	✘ 不支持	✔ 是	✘ 不支持
Amazon Step Functions	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Simple Workflow Service (Amazon SWF)	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

业务应用程序服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Alexa for Business	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Chime	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Honeycode	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon WorkMail	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon WorkMail Message Flow	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持

卫星服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Ground Station	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

物联网服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
FreeRTOS	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon IoT	✔ 是	✔ 是	⚠ 部分 ¹	✔ 是	✔ 是	✘ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon IoT 1-Click	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon IoT Analytics	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon IoT Core Device Advisor	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon IoT Core 对于 LoRaWAN	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon IoT Device Tester	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon IoT Events	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Fleet Hub for Amazon IoT Device Management	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon IoT FleetWise	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon IoT Greengrass	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon IoT GreengrassV2	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon IoT Jobs DataPlane	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon IoT RoboRunner	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon IoT SiteWise	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon IoT Things Graph	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon IoT TwinMaker	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

¹ 连接到 Amazon IoT 的设备通过 X.509 证书或 Amazon Cognito 身份进行身份验证。您可以将 Amazon IoT 策略附加到 X.509 证书或 Amazon Cognito 身份以控制设备有权执行哪些操作。有关更多信息，请参阅 [Amazon IoT 开发人员指南](#) 中的 Amazon IoT 的安全和身份。

机器人服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon RoboMaker	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是

量子计算服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Braket	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是

区块链服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Managed Blockchain	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

游戏开发服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon GameLift	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon GameSparks	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

AR 和 VR 服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Sumerian	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持

客户支持服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon IQ	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon IQ 权限	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Web Services Support	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✔ 是

客户参与服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon AppIntegrations	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Connect	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是
Amazon Connect Cases	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Connect Customer Profiles	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Connect 大容量出站通信	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Connect Voice ID	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Connect Wisdom	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Pinpoint	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Pinpoint 电子邮件服务	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Pinpoint 短信和语音服务	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Pinpoint SMS 和 Voice Service v2	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Simple Email Service (Amazon SES) 第 2 版	✔ 是	⚠ 部分 ¹	✔ 是	✔ 是	⚠ 部分 ²	✘ 不支持

¹ 您只能在引用与发送电子邮件相关的操作（如 `ses:SendEmail` 或 `ses:SendRawEmail`）的策略语句中使用资源级权限。对于引用任何其他操作的策略语句，Resource 元素只能包含 `*`。

² 仅 Amazon SES API 支持临时安全凭证。Amazon SES SMTP 接口不支持从临时安全凭证派生的 SMTP 凭证。

最终用户计算服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon AppStream 2.0	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon WAM	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon WorkDocs	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon WorkSpaces	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon WorkSpaces Application Manager	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon WorkSpaces Web	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✔ 是

账单和成本管理服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Application Cost Profiler 服务	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Billing and Cost Management	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Billing Conductor	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon 成本和使用率报告	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Cost Explorer	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon 发票管理	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon 价目表	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon 采购订单控制台	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Savings Plans	✔ 是	✔ 是	✘ 不支持	✔ 是	✔ 是	✘ 不支持
Amazon Sustainability	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon 税务设置	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持

其他资源

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Activate	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Budget 服务	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✘ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务相关角色
Amazon Web Services Marketplace	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✔ 是
Amazon Web Services Marketplace 目录	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Marketplace Commerce Analytics	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✘ 不支持	✘ 不支持
Amazon Web Services Marketplace 管理门户	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Marketplace Metering Service	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Web Services Marketplace Private Marketplace	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Web Services Marketplace Vendor Insights	✔ 是	✔ 是	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持
Amazon Mechanical Turk	✔ 是	✘ 不支持	✘ 不支持	✘ 不支持	✔ 是	✘ 不支持

IAM JSON 策略参考

本节介绍了 IAM 中的 JSON 策略的元素、变量和评估逻辑的详细语法、描述和示例。有关更多一般信息，请参阅 [JSON 策略概述](#) (p. 360)。

此参考包括以下部分。

- [IAM JSON 策略元素参考](#) (p. 934) - 详细了解您在创建策略时可以使用的元素。查看其他策略示例，并了解条件、支持的数据类型以及在各种服务中的使用方式。
- [策略评估逻辑](#) (p. 972) - 本部分介绍 Amazon 请求、如何对其进行身份验证，以及 Amazon 如何使用策略来确定对资源的访问权限。
- [IAM JSON 策略语言的语法](#) (p. 985) - 本部分介绍用于在 IAM 中创建策略的语言的正式语法。
- [工作职能的 Amazon 托管策略](#) (p. 990) - 本部分列出直接映射到 IT 行业常见工作职能的所有 Amazon 托管策略。使用这些策略可以授予执行特定工作职能中的预期任务所需要的权限。这些策略将许多服务的权限整合到一个策略中。
- [Amazon 全局条件上下文密钥](#) (p. 998) - 本部分包含可用于限制 IAM 策略中的权限的所有 Amazon 全局条件键的列表。
- [IAM 和 Amazon STS 条件上下文密钥](#) (p. 1014) - 本部分包含可用于限制 IAM 策略中的权限的所有 IAM 和 Amazon STS 全局条件键的列表。
- [Amazon 服务的操作、资源和条件键](#) - 本部分提供了您可以在 IAM 策略中用作权限的所有 Amazon API 操作。它还包括可用于进一步优化请求的特定于服务的条件键。

IAM JSON 策略元素参考

JSON 策略文档由元素组成。这些元素按照在策略中使用的大致顺序列出。元素的顺序并不重要，例如，Resource 元素可以在 Action 元素之前出现。您无需在策略中指定任何 Condition 元素。要详细了解 JSON 策略文档的一般结构和目的，请参阅 [JSON 策略概述 \(p. 360\)](#)。

一些 JSON 策略元素相互排斥。这意味着，您不能创建同时使用两个元素的策略。例如，您不能在同一策略语句中同时使用 Action 和 NotAction。相互排斥的其他对包括 Principal/NotPrincipal 和 Resource/NotResource。

策略中所包含的具体内容因产品而异，其取决于产品所提供的操作、包含的资源类型等。当您针对特定产品编写策略时，参考该产品的示例将为您提供帮助。如需获取支持 IAM 的所有服务的列表，以及阐述这些服务的 IAM 和策略的文档链接，请参阅 [使用 IAM 的 Amazon 服务 \(p. 908\)](#)。

当您创建或编辑 JSON 策略时，IAM 可以执行策略验证以帮助您创建有效的策略。IAM 可识别 JSON 语法错误，而 IAM Access Analyzer 将提供额外的策略检查和建议，以帮助您进一步优化策略。要了解策略验证的更多信息，请参阅 [验证 IAM 策略 \(p. 443\)](#)。要了解有关 IAM Access Analyzer 策略检查和可执行建议的更多信息，请参阅 [IAM Access Analyzer 策略验证](#)。

主题

- [IAM JSON 策略元素：Version \(p. 934\)](#)
- [IAM JSON 策略元素：Id \(p. 935\)](#)
- [IAM JSON 策略元素：Statement \(p. 935\)](#)
- [IAM JSON 策略元素：Sid \(p. 936\)](#)
- [IAM JSON 策略元素：Effect \(p. 936\)](#)
- [Amazon JSON 策略元素：Principal \(p. 936\)](#)
- [Amazon JSON 策略元素：NotPrincipal \(p. 942\)](#)
- [IAM JSON 策略元素：Action \(p. 944\)](#)
- [IAM JSON 策略元素：NotAction \(p. 945\)](#)
- [IAM JSON 策略元素：Resource \(p. 946\)](#)
- [IAM JSON 策略元素：NotResource \(p. 948\)](#)
- [IAM JSON 策略元素：Condition \(p. 948\)](#)
- [IAM policy 元素：变量和标签 \(p. 964\)](#)
- [IAM JSON 策略元素：支持的数据类型 \(p. 971\)](#)

IAM JSON 策略元素：Version

消除歧义的说明

此 Version JSON 策略元素与策略版本不同。Version 策略元素用在策略之中，用于定义策略语言的版本。另一方面，当您 IAM 中的客户托管策略进行更改时，将创建一个策略版本。已更改的策略不会覆盖现有策略。而是由 IAM 创建新的托管策略版本。如果您在寻找与托管策略适用的多版本支持相关的信息，请参阅 [the section called “IAM 策略版本控制” \(p. 458\)](#)。

Version 策略元素指定用于处理策略的语言语法规则。要使用所有可用策略功能，请将以下 Version 元素包含在所有策略中的 Statement 元素之外。

```
"Version": "2012-10-17"
```

IAM 支持以下 Version 元素值：

- 2012-10-17. 这是策略语言的当前版本，您应始终包含一个 Version 元素并将其设置为 2012-10-17。否则，您将无法使用此版本中引入的 [策略变量 \(p. 964\)](#) 等功能。

- 2008-10-17. 这是策略语言的早期版本。您可能会在较旧的现有策略中看到此版本。请勿将此版本用于任何新策略或正在更新的任何现有策略。较新的功能（如策略变量）将不适用于您的策略。例如，系统不会将 `${aws:username}` 等变量识别为变量，而是将其视为策略中的文本字符串。

IAM JSON 策略元素：Id

Id 元素指定策略的可选标识符。不同产品使用的 ID 也不尽相同。

对于允许您设置 ID 元素的产品，我们建议您使用 UUID (GUID) 值，或将 UUID 与 ID 合并，以确保唯一性。

```
"Id": "cd3ad3d9-2776-4ef1-a904-4c229d1642ee"
```

Note

某些 Amazon 产品（例如，Amazon SQS 或 Amazon SNS）可能需要该元素并要求其具有唯一性。有关编写策略的特定于服务的信息，请参阅您使用的服务的文档。

IAM JSON 策略元素：Statement

Statement 元素为策略的主要元素。该元素为必填项。Statement 元素可以包含一条语句或由单独语句组成的数组。每条单独的语句块必须用大括号 `{}` 括起来。对于多条语句，数组必须用方括号 `[]` 括起来。

```
"Statement": [{...},{...},{...}]
```

在下列示例所列举的策略中，在单一 Statement 元素内包含一个有三份声明的数组。（该策略允许您在 Amazon S3 控制台中访问您自己的“主文件夹”。）该策略包含 `aws:username` 变量，在策略评估期间它将被替换为请求中的用户名称。有关更多信息，请参阅[介绍 \(p. 965\)](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::BUCKET-NAME",
      "Condition": {"StringLike": {"s3:prefix": [
        "",
        "home/",
        "home/${aws:username}/"
      ]}}
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}",
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}/*"
      ]
    }
  ]
}
```

```
}
```

IAM JSON 策略元素 : Sid

您可以针对策略语句提供的可选标识符 `Sid` (声明 ID)。您可以为语句数组中的每个语句指定 `Sid` 值。在允许您指定 ID 元素的产品中, 例如 SQS 和 SNS, `Sid` 值正是策略文件 ID 的子 ID。在 IAM 中, `Sid` 值必须在 JSON 策略中是唯一的。

```
"Sid": "1"
```

`Sid` 元素支持 ASCII 大写字母 (A-Z)、小写字母 (a-z) 和数字 (0-9)。

IAM 不会公开在 IAM API 中的 `Sid`。您无法基于此 ID 检索特定声明。

Note

某些 Amazon 产品 (例如, Amazon SQS 或 Amazon SNS) 可能需要该元素并要求其具有唯一性。有关编写策略的特定于服务的信息, 请参阅您使用的服务的文档。

IAM JSON 策略元素 : Effect

`Effect` 元素是必需具备的元素, 用于指定声明所产生的结果是“允许”还是“显式拒绝”。`Effect` 的有效值为 `Allow` 和 `Deny`。

```
"Effect": "Allow"
```

在默认情况下, 将拒绝访问资源。如要允许访问资源, 您必须将 `Effect` 元素设置为 `Allow`。如要置换“允许”结果 (例如, 置换否则会生效的“允许”结果), 您需要将 `Effect` 元素设置为 `Deny`。有关更多信息, 请参阅[策略评估逻辑 \(p. 972\)](#)。

Amazon JSON 策略元素 : Principal

在基于资源的 JSON 策略中使用 `Principal` 元素指定允许或拒绝访问资源的主体。

您必须使用[基于资源的策略 \(p. 379\)](#)中的 `Principal` 元素。包括 IAM 在内的多项服务支持基于资源的策略。IAM 中唯一基于资源的策略类型是角色信任策略。在 IAM 角色中, 在角色的信任策略中使用 `Principal` 元素来指定可担任该角色的对象。对于跨账户存取, 您必须指定受信任账户的 12 位标识符。要了解您信任区域之外的账户 (受信任的企业或账户) 中的主体是否有权承担您的角色, 请参阅[什么是 IAM Access Analyzer ?](#)。

Note

创建角色后, 您可以将账户更改为 `***`, 允许所有人担任该角色。如果执行此操作, 我们强烈建议您通过其他方法 (如将访问只限定为特定 IP 地址的 `Condition` 元素) 限制能够访问该角色的用户。不要将角色的访问权限开放给所有人 !

支持基于资源的策略的其他资源示例包括 Amazon S3 存储桶或 Amazon KMS key。

无法在基于身份的策略中使用 `Principal` 元素。基于身份的策略是附加到 IAM 身份 (用户、群体或角色) 的权限策略。在这些策略中, 附加了策略的身份即是主体的身份。

指定主体

您可以在基于资源的策略的 `Principal` 元素中或在支持主体的条件密钥中指定主体。

您可以在策略中指定以下任意主体 :

- Amazon 账户和根用户
- IAM 角色
- 角色会话
- IAM 用户
- 联合身份用户会话
- Amazon 服务
- 所有主体

您无法在策略（例如基于资源的策略）中将用户组标识为主体，因为组与权限相关，与身份验证无关，并且主体是经过身份验证的 IAM 实体。

您可以使用数组在以下部分中为每个主体类型指定多个主体。数组可以采用一个或多个值。如果您在该元素中指定多个主体，则表示您向每个主体授予权限。这是逻辑 OR 而不是逻辑 AND，因为您一次被认证为一个主体。如果包含多个值，请使用方括号（[和]）并用逗号分隔数组的每个条目。以下示例策略定义了 123456789012 账户或 555555555555 账户的权限。

```
"Principal": {
  "AWS": [
    "123456789012",
    "555555555555"
  ]
}
```

Note

您不能使用通配符匹配一部分主体名称或 ARN。

Amazon 账户主体

您可以在基于资源策略的 Principal 元素中或支持主体的条件密钥中指定 Amazon 账户标识符。这将权限委派给账户。当您允许访问其他账户时，该账户中的管理员必须为该账户中身份（IAM 用户或角色）授予访问权限。在指定 Amazon 账户时，您可以使用账户 ARN (arn:aws:iam::[account-ID](#):root) 或 "AWS": 前缀后加账户 ID 构成的简略格式。

例如，给定账户 ID 为 123456789012 的情况下，您可以使用以下任一方法来在 Principal 元素中指定账户：

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

```
"Principal": { "AWS": "123456789012" }
```

账户 ARN 和缩短的账户 ID 的行为方式相同。两者都向账户委派权限。在 Principal 元素中使用账户 ARN 并不会将权限限制为该账户的根用户。

Note

当您保存包含缩短账户 ID 的基于资源的策略时，服务可能会将其转换为主体 ARN。这不会更改策略的功能。

某些 Amazon 服务支持其他用于指定账户主体的选项。例如，Amazon S3 允许您使用以下格式指定[规范用户 ID](#)：

```
"Principal": { "CanonicalUser":
  "79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be" }
```

您还可以指定多个 Amazon 账户（或规范用户 ID）作为使用数组的主体。例如，您可以使用所有三种方法在存储桶策略中指定主体。

```
"Principal": {
  "AWS": [
    "arn:aws:iam::123456789012:root",
    "999999999999"
  ],
  "CanonicalUser": "79a59df900b949e55d96a1e698fbacedfd6e09d98eacbf8f8d5218e7cd47ef2be"
}
```

IAM 角色主体

您可以在支持主体的基于资源的策略或条件密钥的 `Principal` 元素中指定 IAM 角色主体 ARN。IAM 角色是身份。在 IAM 中，身份是您可以向其分配权限的资源。角色信任另一个身份验证来担任该角色。这包含 Amazon 中的主体或来自外部身份提供程序 (IdP) 的用户。当主体或身份代入角色时，他们会收到具有代入角色权限的临时安全证书。当他们使用这些凭证在 Amazon 中执行操作时，将变为角色会话主体。

IAM 角色是 IAM 中存在的身份。角色信任另一个经过身份验证的身份，如 Amazon 中的主体或来自外部身份提供程序的用户。当主体或身份代入角色时，他们会收到临时安全凭证。然后，他们可以使用这些凭证作为角色会话主体在 Amazon 中执行操作。

当您在基于资源的策略中指定角色主体时，主体的有效权限受限制该角色权限的任何策略类型的限制。这包括会话策略和权限边界。有关如何评估角色会话的有效权限的更多信息，请参阅 [策略评估逻辑 \(p. 972\)](#)。

要指定 `Principal` 元素中的角色 ARN，请采用以下格式：

```
"Principal": { "AWS": "arn:aws:iam::Amazon-account-ID:role/role-name" }
```

Important

如果角色信任策略的 `Principal` 元素中包含指向特定 IAM 角色的 ARN，在保存策略时该 ARN 将转换为该角色的唯一主体 ID。如果有人希望通过删除并重新创建角色来提升特权，这样有助于减轻此类风险。您通常不会在控制台中看到这个 ID，因为 IAM 在显示信任策略时使用反向转换回角色 ARN。但是，如果您删除角色，这种关系即被打破。即使您重新创建角色，策略也不再适用。因为新角色拥有新的主体 ID，与信任策略中存储的 ID 不匹配。发生这种情况时，主体 ID 会出现在基于资源的策略中，因为 Amazon 无法将其映射回有效的 ARN。最终结果是，如果您删除并重新创建了信任策略的 `Principal` 元素所引用的角色，您必须编辑角色，用正确的 ARN 替换目前不正确的主体 ID。当您保存策略时，ARN 会再次转换为该角色新的主体 ID。

IAM 联合身份用户 (p. 941) — IAM 用户使用 `aws:PrincipalArn` 操作联合身份，为 IAM 用户生成联合身份用户会话主体。当您使用此密钥时，角色会话主体将根据所担任的角色的 ARN 而不是所生成的会话的 ARN 授予权限。由于 Amazon 不将条件密钥 ARN 转换为 ID，如果您删除该角色，然后创建一个具有相同名称的新角色，则授予该角色 ARN 的权限将继续存在。在基于资源的策略的 `Principal` 元素中使用带有通配符 (*) 的 `aws:PrincipalArn` 条件键授予的许可不受基于身份的策略类型的限制，例如许可边界或会话策略。任一项策略中的显式拒绝将覆盖允许。

角色会话主体

您可以在支持主体的基于资源的策略或条件密钥的 `Principal` 元素中指定角色会话。当主体或身份代入角色时，他们会收到具有代入角色权限的临时安全证书。当他们使用这些凭证，在 Amazon 中执行操作时，将变为角色会话主体。

您用于角色会话主体的格式取决于用来代入角色的 Amazon STS 操作。

此外，管理员可以设计一个流程来控制角色会话的发放方式。例如，他们可以为用户提供一键式解决方案，以创建可预测的会话名称。如果管理员执行此操作，则您可以在策略或条件密钥中使用角色会话主体。否

则，您可以在 `aws:PrincipalArn` 条件键中将角色 ARN 指定为主体。如何将角色指定为主体可以更改所生成的会话的有效权限。有关更多信息，请参阅[IAM 角色主体 \(p. 938\)](#)。

代入角色会话主体

代入的角色会话主体是使用 Amazon STS `AssumeRole` 操作所产生的会话主体。有关哪些主体可以使用此操作代入角色的更多信息，请参阅[比较 Amazon STS API 操作 \(p. 310\)](#)。

要指定 `Principal` 元素中代入角色会话的 ARN，请采用以下格式：

```
"Principal": { "AWS": "arn:aws:sts::AWS-account-ID:assumed-role/role-name/role-session-name" }
```

当您在 `Principal` 元素中指定代入角色会话时，不能使用通配符 "*" 表示所有会话。主体必须始终指明特定会话。

Web 身份会话主体

Web 身份会话主体是使用 Amazon STS `AssumeRoleWithWebIdentity` 操作产生的会话主体。您可以使用外部 Web 身份提供程序 (IdP) 登录，然后使用此操作代入 IAM 角色。这利用了联合身份并发出角色会话。有关哪些主体可以使用此操作代入角色的更多信息，请参阅[比较 Amazon STS API 操作 \(p. 310\)](#)。

当通过 Web 身份提供程序发出角色时，您将获得这种特殊类型的会话主体，其中包括 Web 身份提供程序相关信息。

在策略中使用此主体类型可基于受信任的 Web 身份提供程序来允许或拒绝访问。要指定角色信任策略 `Principal` 元素中 Web 身份角色会话的 ARN，请采用以下格式：

```
"Principal": { "Federated": "cognito-identity.amazonaws.com" }
```

```
"Principal": { "Federated": "www.amazon.com" }
```

```
"Principal": { "Federated": "graph.facebook.com" }
```

```
"Principal": { "Federated": "accounts.google.com" }
```

SAML 会话主体

SAML 会话主体是使用 Amazon STS `AssumeRoleWithSAML` 操作产生的会话主体。您可以使用外部 SAML 身份提供程序 (IdP) 登录，然后使用此操作代入 IAM 角色。这利用了身份联合身份验证并发出角色会话。有关哪些主体可以使用此操作代入角色的更多信息，请参阅[比较 Amazon STS API 操作 \(p. 310\)](#)。

从 SAML 身份提供程序发出角色时，您将获得这种特殊类型的会话主体，其中包括有关 SAML 身份提供程序的信息。

在策略中使用此主体类型可基于受信任的 SAML 身份提供程序来允许或拒绝访问。要指定角色信任策略 `Principal` 元素中 Web 身份角色会话的 ARN，请采用以下格式：

```
"Principal": { "Federated": "arn:aws:iam::Amazon-account-ID:saml-provider/provider-name" }
```

IAM 用户主体

您可以在支持主体的基于资源的策略或条件键的 `Principal` 元素中指定 IAM 用户。

Note

在 Principal 元素中，[Amazon Resource Name \(ARN\)](#) (p. 896) 的用户名部分区分大小写。

```
"Principal": { "AWS": "arn:aws:iam::Amazon-account-ID:user/user-name" }
```

```
"Principal": {  
  "AWS": [  
    "arn:aws:iam::Amazon-account-ID:user/user-name-1",  
    "arn:aws:iam::Amazon-account-ID:user/user-name-2"  
  ]  
}
```

当在 Principal 元素中指定用户时，不能使用通配符 (*) 表示“所有用户”。主体必须始终指明特定用户。

Important

如果角色信任策略的 Principal 元素中包含指向特定 IAM 用户的 ARN，则在保存策略时该 IAM 将 ARN 转换为该用户的唯一主体 ID。如果有人希望通过删除并重新创建用户来提升特权，这样有助于减轻此类风险。您通常不会在控制台中看到这个 ID，因为显示信任策略时它还会反向转换为用户的 ARN。但是，如果您删除角色，这种关系即被打破。即使您重新创建用户，策略也不再适用。这是因为：新用户拥有新的主体 ID，该 ID 与信任策略中存储的 ID 不匹配。发生这种情况时，主体 ID 会出现在基于资源的策略中，因为 Amazon 无法将其映射回有效的 ARN。结果是，如果您删除并重新创建了信任策略的 Principal 元素所引用的用户，您必须编辑角色，用正确的 ARN 替换目前不正确的主体 ID。当您保存策略时，IAM 会再次将 ARN 转换为该用户新的主体 ID。

Amazon STS 联合身份用户会话主体

您可以在支持主体的基于资源的策略或条件密钥的 Principal 元素中指定 federated user sessions (联合身份用户会话)。

Important

Amazon 建议只在必要时使用 Amazon STS 联合身份用户会话，如[需要使用根用户访问权限](#)时。反过来，[使用角色来委托权限](#) (p. 31)。

Amazon STS 联合身份用户会话主体是使用 Amazon STS GetFederationToken 操作产生的会话主体。在这种情况下，Amazon STS 将[联合身份](#)而非 IAM 角色作为获取临时访问令牌的方法。

在 Amazon 中，IAM 用户或 Amazon Web Services 账户 根用户可以使用长期访问密钥进行身份验证。有关哪些主体可以使用此操作联合身份的更多信息，请参阅 [比较 Amazon STS API 操作](#) (p. 310)。

- IAM 联合身份用户 — IAM 用户使用 GetFederationToken 操作联合身份，为 IAM 用户生成联合身份用户会话主体。
- 联合身份根用户 — 一个根用户联邦成员使用对该根用户产生联邦用户会话主体的 GetFederationToken 操作。

当 IAM 用户或根用户通过使用此操作的 Amazon STS 请求临时凭证时，他们开启临时的联合身份会话。本会话的 ARN 基于联合身份的原始身份。

要指定 Principal 元素中联合身份用户会话的 ARN，请采用以下格式：

```
"Principal": { "AWS": "arn:aws:sts::AWS-account-ID:federated-user/user-name" }
```

Amazon 服务主体

您可以在支持主体的基于资源的策略或条件密钥的 Principal 元素中指定 Amazon 服务。一个服务主体是服务的标识符。

可以由 Amazon 服务担任的 IAM 角色称为 [服务角色 \(p. 159\)](#)。服务角色必须包括信任策略。信任策略是附加到角色的基于资源的策略，这些策略定义了可担任该角色的主体。某些服务角色具有预定义的信任策略。但有些情况下，您必须在信任策略中指定服务主体。

服务主体的标识符包括服务名称，通常采用以下格式：

```
service-name.amazonaws.com
```

但是，某些服务可能使用以下格式代替常用格式或作为常用格式的补充：

```
service-name.amazonaws.com
```

服务主体由服务定义。可以通过以下方式找到某些服务的主体：打开 [使用 IAM 的 Amazon 服务 \(p. 908\)](#)，检查服务在 Service-linked role（服务相关角色）列中是否具有 Yes，然后打开 Yes（是）链接以查看该服务的服务相关角色文档。查找该服务的服务相关角色权限部分，查看服务主体。

以下示例显示了一个可以附加到服务角色的策略。该策略可启用两个服务（Amazon ECS 和 Elastic Load Balancing）以担任该角色。然后，这些服务可执行由分配给该角色（未显示）的权限策略授权执行的任何任务。要指定多个服务主体，不用指定两个 Service 元素；您可以只使用一个该元素。实际上，您可以将一组多个服务主体作为单个 Service 元素的值。

```
"Principal": {
  "Service": [
    "ecs.amazonaws.com",
    "elasticloadbalancing.amazonaws.com"
  ]
}
```

所有主体

您可以在基于资源的策略的 Principal 元素或在支持主体的条件键中使用通配符（*）指定所有主体。使用 [基于资源的策略 \(p. 356\)](#) 授予权限和 [条件键](#) 来限制策略语句的条件。

Important

我们强烈建议不要在带有 Allow 效果的基于资源的策略的 Principal 元素中使用通配符（*），除非您打算授予公共或匿名访问权限。否则，请在 Principal 元素中指定预期的主体、服务或 Amazon 账户，然后进一步将访问权限限制在 Condition 元素内。对于 IAM 角色信任策略尤其如此，因为它们允许其他主体成为您账户中的主体。

对于基于资源的策略，使用带 Allow 效果的通配符（*）向所有用户（包括匿名用户）授予访问权限（公共访问权限）。对于账户中的 IAM 用户和角色主体，不需要其他权限。对于其他账户中的主体，他们还必须在其账户中拥有基于身份的权限，以允许他们访问您的资源。这称为 [cross-account access](#)（跨账户存取）。

对于匿名用户，以下元素是等效的：

```
"Principal": ""
```

```
"Principal" : { "AWS" : "" }
```

您不能使用通配符匹配一部分主体名称或 ARN。

以下示例说明了基于资源的策略（而非 [NotPrincipal](#) 与 [Deny \(p. 943\)](#)）可用于显式拒绝除 Condition 元素中指定的主体之外的所有主体。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "UsePrincipalArnInsteadOfNotPrincipalWithDeny",
    "Effect": "Deny",
    "Action": "s3:*",
    "Principal": "*",
    "Resource": [
      "arn:aws:s3:::BUCKETNAME/*",
      "arn:aws:s3:::BUCKETNAME"
    ],
    "Condition": {
      "ArnNotEquals": {
        "aws:PrincipalArn": "arn:aws:iam::444455556666:user/user-name"
      }
    }
  }
]
```

更多信息

有关更多信息，请参阅以下内容：

- 《Amazon Simple Storage Service 用户指南》中的 [存储桶策略示例](#)
- 《Amazon Simple Notification Service 开发人员指南》中的 [Amazon SNS 示例策略](#)
- 《Amazon Simple Queue Service 开发商指南》中的 [Amazon SQS 策略示例](#)
- 《Amazon Key Management Service 开发人员指南》中的 [密钥策略](#)
- 《Amazon 一般参考》中的 [账户标识符](#)
- [关于 Web 联合身份验证 \(p. 171\)](#)

Amazon JSON 策略元素：NotPrincipal

使用 NotPrincipal 元素指定 IAM 用户、联合身份用户或 IAM 角色、Amazon 账户、Amazon 服务或其他不被允许或拒绝其访问资源的主体。通过使用 NotPrincipal 元素，可指定主体列表中的例外情况。使用此元素拒绝对除 NotPrincipal 元素中指定的主体之外的所有主体的访问。指定 NotPrincipal 的语法与指定的 [Amazon JSON 策略元素：Principal \(p. 936\)](#) 语法相同。

无法在基于 IAM 身份的策略或 IAM 角色信任策略中使用 NotPrincipal 元素。对于一些 Amazon 服务，可以在基于资源的策略中使用。基于资源的策略是直接嵌入资源中的策略。

Important

极少有需要使用 NotPrincipal 的情况，我们建议在决定使用 NotPrincipal 前了解其他授权选项。

NotPrincipal 与 Allow

我们强烈建议不要在与 NotPrincipal 相同的策略语句中使用 "Effect": "Allow"。这样做将允许除 NotPrincipal 元素中指定的主体之外的所有主体。我们建议不要这样做，因为在策略语句中指定的权限将授予除了所指定主体之外的所有主体。执行此操作后，您可能会将访问权限授予匿名（未经身份验证的）用户。

无法在基于 IAM 身份的策略或 IAM 角色信任策略中使用 NotPrincipal 元素。大多数基于资源的策略类型不允许将 NotPrincipal 和 "Effect": "Allow" 结合使用。随着时间的推移，Amazon 将限制在所有权限策略中使用这些策略语句。

NotPrincipal 与 Deny

Note

当您在与 "Effect": "Deny" 相同的策略语句中使用 NotPrincipal 时，排查多策略类型的影响可能会很难。所以，我们建议使用 aws:PrincipalArn 条件键。有关更多信息，请参阅[所有主体 \(p. 941\)](#)。

当在与 "Effect": "Deny" 相同的策略语句中使用 NotPrincipal 时，策略语句中指定的操作将被明确拒绝授予除指定主体之外的所有主体。当您使用 NotPrincipal 与 Deny 一起使用时，还必须指定未拒绝主体的账户 ARN。否则，策略可能会拒绝包含主体的整个账户的访问。根据您的策略中包含的服务，Amazon 可能会先验证账户，然后验证用户。如果评估一个担任角色的用户（使用角色的某个人），Amazon 会相继验证账户、角色，最后才是担任角色的用户。担任角色的用户是由他们担任角色时指定的角色会话名称标识的。因此，我们强烈建议您明确包括用户账户的 ARN，或者包括角色的 ARN 以及包含角色的账户的 ARN。

Note

作为最佳实践，您应在策略中包含账户的 ARN。一些服务需要账户 ARN，虽然这并非在所有情况下均必需。没有必需 ARN 的任意现有策略将继续工作，但包括这些服务的新策略必须满足此要求。IAM 不会跟踪这些服务，因此建议您始终包括账户 ARN。

以下示例说明了在同一策略语句中有效使用 NotPrincipal 和 "Effect": "Deny" 的方法。

Example 相同或不同账户中的示例 IAM 用户

在下面的示例中，除 Amazon 账户 444455556666 中名为 Bob 的用户外，所有其他主体均被显式拒绝访问某个资源。请注意，作为最佳实践，NotPrincipal 元素包含用户 Bob 的 ARN 以及 Bob 所属的 Amazon 账户 (arn:aws:iam::444455556666:root) 的 ARN。如果 NotPrincipal 元素仅包含 Bob 的 ARN，策略的效果可能是明确拒绝包含用户 Bob 的 Amazon 账户的访问。在某些情况下，用户不能拥有比其父账户更多的权限，因此，如果 Bob 的账户被明确拒绝访问，则 Bob 可能无法访问该资源。

若本示例是基于资源的策略（挂载到相同或不同 Amazon 账户（非 444455556666）中的资源）中的策略语句的一部分，则它可以正常工作。本示例本身不会向 Bob 授予访问权限，而只会将 Bob 从被显式拒绝的主体的列表中省略。要允许 Bob 访问资源，另一个策略语句必须使用 "Effect": "Allow" 明确允许访问。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "NotPrincipal": { "AWS": [
      "arn:aws:iam::444455556666:user/Bob",
      "arn:aws:iam::444455556666:root"
    ] },
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::BUCKETNAME",
      "arn:aws:s3:::BUCKETNAME/*"
    ]
  }]
}
```

Example 相同或不同账户中的示例 IAM 角色

在下面的示例中，除 Amazon 账户 444455556666 中名为 cross-account-audit-app 的担任了角色的用户之外，所有其他主体均被显式拒绝访问资源。作为最佳实践，NotPrincipal 元素包含担任角色的用户 (cross-account-audit-app)、角色 (cross-account-read-only-role) 和角色所属的 Amazon 账户 (444455556666) 的 ARN。如果 NotPrincipal 元素缺少角色的 ARN，策略的效果可能是明确拒绝角色的访问。同样，如果 NotPrincipal 元素缺少角色所属的 Amazon 账户的 ARN，策略的效果可能是明确拒绝 Amazon 账户以及该账户中所有实体的访问。在一些情况下，担任角色的用户不能拥有比其父角色更多

的权限，而且角色不能拥有比其父 Amazon 账户更多的权限，因此，当角色或账户被明确拒绝访问时，担任角色的用户可能无法访问资源。

当本示例是基于资源的策略（附加到不同的 Amazon 账户（非 444455556666）中的资源）中的策略语句的一部分时，它可以正常工作。本示例本身不允许担任角色的用户 cross-account-audit-app 访问，而是仅在被明确拒绝的主体的列表中忽略 cross-account-audit-app。要向 cross-account-audit-app 授予对资源的访问权限，另一个策略语句必须使用 "Effect": "Allow" 显式允许访问。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "NotPrincipal": { "AWS": [
      "arn:aws:sts::444455556666:assumed-role/cross-account-read-only-role/cross-account-audit-app",
      "arn:aws:iam::444455556666:role/cross-account-read-only-role",
      "arn:aws:iam::444455556666:root"
    ]},
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::Bucket_AccountAudit",
      "arn:aws:s3:::Bucket_AccountAudit/*"
    ]
  }]
}
```

当您在 NotPrincipal 元素中指定代入角色会话时，不能使用通配符 (*) 表示“所有会话”。主体必须始终指明特定会话。

IAM JSON 策略元素：Action

Action 元素描述将允许或拒绝的特定操作。声明必须包含 Action 或 NotAction 元素。每款 Amazon 服务各自拥有一套描述任务的操作，您可使用相应服务来执行所描述的任务。例如，Amazon S3 的操作列表可以在 Amazon Simple Storage Service 用户指南中的[在策略中指定权限](#)找到，Amazon EC2 操作列表可以在[Amazon EC2 API 参考](#)中找到，而适用于 Amazon Identity and Access Management 的操作列表可以在[IAM API 参考](#)中找到。要查找其他服务的操作列表，请参阅 API 参考[文档](#)了解相关服务。

通过将服务命名空间用作操作前缀（iam、ec2、sqs、sns、s3 等）并后跟允许或拒绝的操作名称来指定值。该名称必须与产品支持的操作相匹配。前缀和操作名称不区分大小写。例如，iam:ListAccessKeys 与 IAM:listaccesskeys 相同。以下示例显示用于不同服务的 Action 元素。

Amazon SQS 操作

```
"Action": "sqs:SendMessage"
```

Amazon EC2 操作

```
"Action": "ec2:StartInstances"
```

IAM 操作

```
"Action": "iam:ChangePassword"
```

Amazon S3 操作

```
"Action": "s3:GetObject"
```

您可以为 Action 元素指定多个值。

```
"Action": [ "sqs:SendMessage", "sqs:ReceiveMessage", "ec2:StartInstances",  
            "iam:ChangePassword", "s3:GetObject" ]
```

您可使用通配符 (*) 来访问特定 Amazon 产品提供的所有操作。例如，以下 Action 元素适用于所有 S3 操作。

```
"Action": "s3:*"
```

还可以使用通配符 (*) 作为操作名称的一部分。例如，下列 Action 元素适用于所有包含字符串 AccessKey 的 IAM 操作，包括 CreateAccessKey、DeleteAccessKey、ListAccessKeys 及 UpdateAccessKey。

```
"Action": "iam:*AccessKey*"
```

某些产品允许您限制可用的操作。例如，Amazon SQS 允许您只提供所有可能的 Amazon SQS 操作的一部分。在这种情况下，* 通配符不允许完全控制队列；而是仅允许控制您已共享的操作子集。有关更多信息，请参阅 Amazon Simple Storage Service 开发人员指南中的[了解权限](#)。

IAM JSON 策略元素：NotAction

NotAction 是与指定的操作列表之外的所有内容显式匹配的高级策略元素。使用 NotAction 时只列出不应匹配的一些操作，而不是包括将匹配的长操作列表，因此生成的策略较短。使用 NotAction 时应注意，在此元素中指定的操作是受限的唯一操作。这也就意味着，如果使用 Allow 效果，则允许未列出的所有适用操作或服务。此外，如果使用 Deny 效果，则拒绝此类未列出的操作或服务。将 NotAction 与 Resource 元素结合使用时，您需要提供策略的范围。这是 Amazon 确定哪些操作或服务适用的方式。有关更多信息，请参阅以下策略示例。

NotAction 与 Allow

可在包含 "Effect": "Allow" 的语句中使用 NotAction 元素来提供对 Amazon 服务中所有操作（在 NotAction 中指定的操作除外）的访问权限。您可以将它与 Resource 元素结合使用以提供策略的范围，从而将允许的操作限制为那些可在指定资源上执行的操作。

以下示例允许用户访问可在任何 S3 资源上执行的所有 Amazon S3 操作，但删除存储桶的操作除外。这将不允许用户使用 ListAllMyBuckets S3 API 操作，因为该操作需要 "*" 资源。此策略也不允许其他服务中的操作，因为其他服务操作不适用于 S3 资源。

```
"Effect": "Allow",  
"NotAction": "s3:DeleteBucket",  
"Resource": "arn:aws:s3:::*"
```

有时您可能需要允许访问大量操作。使用 NotAction 元素可有效地修改语句，生成更短的操作列表。例如，由于 Amazon 提供了如此多的服务，您可能需要创建一个策略，以允许用户执行除访问 IAM 操作之外的所有操作。

下面的示例允许用户访问除 IAM 之外的 Amazon 服务中的所有操作。

```
"Effect": "Allow",  
"NotAction": "iam:*",  
"Resource": "*" 
```

在同一语句或一个策略内的不同语句中使用 NotAction 元素和 "Effect": "Allow" 时要谨慎。NotAction 匹配未显式列出或适用指定资源的所有服务和操作，并且可能导致向用户授予超出您意图的更多权限。

NotAction 与 Deny

可在一个语句中将 NotAction 元素与 "Effect": "Deny" 结合使用，以拒绝访问列出的所有资源，在 NotAction 元素中指定的操作除外。此组合不允许列出的项目，而是显式拒绝未列出的操作。您仍必须允许您要允许的操作。

下面的条件示例在用户未使用 MFA 登录时拒绝其访问非 IAM 操作。如果用户已使用 MFA 登录，则 "Condition" 测试失败，最后的 "Deny" 语句将无效。但请注意，这不会向用户授予对任何操作的访问权限，只会明确拒绝除 IAM 操作之外的所有其他操作。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "DenyAllUsersNotUsingMFA",
    "Effect": "Deny",
    "NotAction": "iam:*",
    "Resource": "*",
    "Condition": {"BoolIfExists": {"aws:MultiFactorAuthPresent": "false"}}
  }]
}
```

有关拒绝访问特定区域以外的操作（来自特定服务的操作除外）的示例策略，请参阅[Amazon：根据请求的区域拒绝访问 Amazon \(p. 404\)](#)。

IAM JSON 策略元素：Resource

Resource 元素可指定声明涵盖的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。您可使用 ARN 来指定资源。有关 ARN 格式的更多信息，请参见 [IAM ARN \(p. 896\)](#)。

每款产品均拥有其各自的资源。尽管您经常使用 ARN 来指定资源，但资源的 ARN 详情要取决于产品和资源。有关如何指定资源的信息，请参阅您编写的资源声明所对应的产品文档。

Note

某些服务不让您指定用于单个资源的操作，而是在 Action 或 NotAction 元素中列出的所有操作都适用于该服务中的所有资源。在这些情况下，您可以在 * 元素中使用通配符 Resource。

下列示例适用于特定的 Amazon SQS 队列。

```
"Resource": "arn:aws:sqs:us-east-2:account-ID-without-hyphens:queue1"
```

以下示例引用了 Amazon 账户中名为 Bob 的 IAM 用户。

Note

每个 IAM 用户名都是唯一的，不区分大小写。

```
"Resource": "arn:aws:iam::account-ID-without-hyphens:user/Bob"
```

在资源 ARN 中使用通配符

您可将通配符作为资源 ARN 的一部分。您可在任何 ARN 分段（用冒号分隔的部分）中使用通配符（* 和 ?）。星号（*）表示字符的任意组合，问号（?）表示任何单个字符。您可以在每个分段中使用多个 * 或 ? 字符。以下示例引用了其路径为 /accounting 的所有 IAM 用户。

```
"Resource": "arn:aws:iam::account-ID-without-hyphens:user/accounting/*"
```


下列示例适用于特定 Amazon S3 存储桶内的所有项目。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
```

星号 (*) 字符可以展开以替换分段中的所有内容，包括像正斜杠 (/) 这样的字符，否则这些字符可能在给定服务命名空间中显示为分隔符。例如，请考虑以下 Amazon S3 ARN，因为相同的通配符扩展逻辑适用于所有服务。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*/test/*"
```

ARN 中的通配符适用于存储桶中的以下所有对象，而不仅仅是列出的第一个对象。

```
DOC-EXAMPLE-BUCKET/1/test/object.jpg  
DOC-EXAMPLE-BUCKET/1/2/test/object.jpg  
DOC-EXAMPLE-BUCKET/1/2/test/3/object.jpg  
DOC-EXAMPLE-BUCKET/1/2/3/test/4/object.jpg  
DOC-EXAMPLE-BUCKET/1///test///object.jpg  
DOC-EXAMPLE-BUCKET/1/test/.jpg  
DOC-EXAMPLE-BUCKET//test/object.jpg  
DOC-EXAMPLE-BUCKET/1/test/
```

考虑前一个列表中的最后两个对象。Amazon S3 对象名称可以有效地以常规分隔符正斜杠 (/) 字符开头或结尾。虽然 "/" 可作为分隔符，但在资源 ARN 中使用此字符时没有特定意义。它将被视为与任何其他有效字符相同。ARN 将不与以下对象匹配：

```
DOC-EXAMPLE-BUCKET/1-test/object.jpg  
DOC-EXAMPLE-BUCKET/test/object.jpg  
DOC-EXAMPLE-BUCKET/1/2/test.jpg
```

指定多个资源

您可指定多个资源。下列示例适用于两个 DynamoDB 表。

```
"Resource": [  
  "arn:aws:dynamodb:us-east-2:account-ID-without-hyphens:table/books_table",  
  "arn:aws:dynamodb:us-east-2:account-ID-without-hyphens:table/magazines_table"  
]
```

在资源 ARN 中使用策略变量

在 Resource 元素中，您可以在标识特定资源的 ARN 部分 (即，ARN 尾部) 中使用 JSON [策略变量 \(p. 964\)](#)。例如，您可以使用键 {aws:username} 作为资源 ARN 的一部分，以表示应包含当前用户的名称作为资源名称的一部分。下列示例显示如何在 {aws:username} 元素中使用 Resource 键。该策略允许访问匹配当前用户名称的 Amazon DynamoDB 表。

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

有关 JSON 策略变量的更多信息，请参阅 [IAM policy 元素：变量和标签 \(p. 964\)](#)。

IAM JSON 策略元素：NotResource

NotResource 是高级策略元素，可明确匹配除指定资源以外的所有资源。使用 NotResource 时只列出不匹配的一些资源，而不是包括将匹配的长资源列表，因此生成的策略较短。这对于在单个 Amazon 服务中应用的策略特别有用。

例如，假设您有名为 HRPayroll 的组。不应允许 HRPayroll 的成员访问除 HRBucket 存储桶中的 Amazon S3 文件夹之外的任何 Payroll 资源。以下策略显式拒绝访问除所列资源外的所有 Amazon S3 资源。但请注意，该策略不向用户授予对任何资源的访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "s3:*",
    "NotResource": [
      "arn:aws:s3:::HRBucket/Payroll",
      "arn:aws:s3:::HRBucket/Payroll/*"
    ]
  }
}
```

通常，为了显式拒绝对某一资源的访问，可以编写一条策略，其中使用 "Effect": "Deny"，而且包含一个单独列出每个文件夹的 Resource 元素。不过，在此示例中，每当向 HRBucket 添加文件夹或向 Amazon S3 添加不应访问的资源时，都必须将其名称添加到 Resource 中的列表。如果改用 NotResource 元素，除非将文件夹名称添加到 NotResource 元素，否则会自动拒绝用户访问新文件夹。

使用 NotResource 时应注意，在此元素中指定的资源是不受限的唯一资源。反过来，这限制了将应用于该操作的所有资源。在上述示例中，该策略只影响 Amazon S3 操作，因此只影响 Amazon S3 资源。如果操作还包含 Amazon EC2 操作，则该策略不会拒绝访问任何 EC2 资源。要了解服务中的哪些操作允许指定资源的 ARN，请参阅 [Amazon 服务的操作、资源和条件键](#)。

具有其他元素的 NotResource

您应从不将 "Effect": "Allow"、"Action": "*" 和 "NotResource": "arn:aws:s3:::HRBucket" 元素一起使用。此语句非常危险，因为它允许对在 HRBucket S3 存储桶之外的所有资源执行 Amazon 中的所有操作。这甚至允许用户向自己添加策略，从而允许他们访问 HRBucket。请勿执行此操作。

在同一语句或一个策略内的不同语句中使用 NotResource 元素和 "Effect": "Allow" 时要谨慎。NotResource 允许未显式列出的所有服务和资源，并且可能导致向用户授予超出您意图的更多权限。在同一语句中使用 NotResource 元素和 "Effect": "Deny" 将拒绝未显式列出的服务和资源。

IAM JSON 策略元素：Condition

Condition 元素（或 Condition 块）允许您指定策略生效的条件。Condition 元素是可选的。在 Condition 元素中，您可构建表达式并使用 [条件运算符](#) (p. 951)（等于、小于等）将策略中的条件键和值与请求上下文中的键和值进行匹配。要了解有关请求上下文的更多信息，请参阅 [请求](#) (p. 5)。

```
"Condition" : { "{condition-operator}" : { "{condition-key}" : "{condition-value}" }}
```

您指定的条件键可以是 [全局条件键](#) (p. 998) 或特定于服务的条件键。全局条件键具有 aws: 前缀。服务特定的条件键具有服务的前缀。例如，Amazon EC2 允许您使用服务独有的 ec2:InstanceType 键来编写条件。要查看带 iam: 前缀的特定于服务的 IAM 条件键，请参阅 [IAM 和 Amazon STS 条件上下文密钥](#) (p. 1014)。

条件键名称不区分大小写。例如，包含 aws:SourceIP 条件键等效于测试 AWS:SourceIp。条件键值是否区分大小写取决于您使用的 [条件运算符](#) (p. 951)。例如，以下条件包含 StringEquals 运算符，以确保只有 johndoe 提出的请求才符合条件。名为 JohnDoe 的用户会被拒绝访问。

```
"Condition" : { "StringEquals" : { "aws:username" : "johndoe" } }
```

以下条件使用 [StringEqualsIgnoreCase \(p. 951\)](#) 运算符来匹配名为 johndoe 或 JohnDoe 的用户。

```
"Condition" : { "StringEqualsIgnoreCase" : { "aws:username" : "johndoe" } }
```

某些条件键支持允许您指定键名称的一部分的键-值对。示例包括 [aws:RequestTag/tag-key \(p. 998\)](#) 全局条件键、Amazon KMS [kms:EncryptionContext:encryption_context_key](#) 和多个服务支持的 [ResourceTag/tag-key](#) 条件键。

- 如果您对 [Amazon EC2](#) 等服务使用 [ResourceTag/tag-key](#) 条件键，则必须指定 tag-key 的键名称。
- 键名称不区分大小写。这意味着，如果您在策略的条件元素中指定 "aws:ResourceTag/TagKey1": "Value1"，则条件将匹配名为 TagKey1 或 tagkey1 的资源标签键，但不会同时匹配两者。
- 支持这些属性的 Amazon 服务可能允许您创建多个仅大小写不同的键名称。例如，您可能需要通过 ec2=test1 和 EC2=test2 标记 Amazon EC2 实例。在使用 "aws:ResourceTag/EC2": "test1" 等条件以允许访问该资源时，键名称匹配两个标签，但仅一个值匹配。这可能会导致意外的条件失败。

Important

最佳实践是，在命名键-值对属性时，请确保您的账户成员遵循一致的命名约定。示例包括标签或 Amazon KMS 加密上下文。您可以使用 [aws:TagKeys \(p. 1011\)](#) 条件键（对于标记）或 [kms:EncryptionContextKeys](#)（对于 Amazon KMS 加密上下文）强制实现此目的。

- 有关所有条件运算符及其功能描述的列表，请参阅[条件运算符 \(p. 951\)](#)
- 除非另行指定，否则所有键均可有多个值。有关如何处理具有多个值的条件键，请参阅[创建具有多个键或值的条件 \(p. 958\)](#)
- 有关所有全局可用的条件键的列表，请参阅[Amazon 全局条件上下文密钥 \(p. 998\)](#)。
- 有关每个服务定义的条件键，请参阅[Amazon 服务的操作、资源和条件键](#)。

请求上下文

当主体 (p. 5) 向 Amazon 发出请求 (p. 5) 时，Amazon 会将请求信息收集到请求上下文中。该信息用于对请求进行评估和授权。您可以使用 JSON 策略的 Condition 元素来针对请求上下文测试特定条件。例如，您可以创建一个策略，该策略使用 [aws:CurrentTime \(p. 1001\)](#) 条件键以允许用户仅在特定的日期范围内执行操作 (p. 394)。

在提交请求时，Amazon 将评估策略中的每个条件键并返回值 true、false、not present 和（有时）null（空数据字符串）。请求中不存在的键会视为不匹配。例如，以下策略允许删除您自己的多重身份验证 (MFA) 设备，但前提是您在最近 1 小时（3600 秒）内已使用 MFA 进行登录。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "AllowRemoveMfaOnlyIfRecentMfa",
    "Effect": "Allow",
    "Action": [
      "iam:DeactivateMFADevice",
      "iam>DeleteVirtualMFADevice"
    ],
    "Resource": "arn:aws:iam::*:user/${aws:username}",
    "Condition": {
      "NumericLessThanEquals": {"aws:MultiFactorAuthAge": "3600"}
    }
  }
}
```

请求上下文可返回以下值：

- True - 如果请求者在过去 1 小时或更短时间内已使用 MFA 进行登录，则条件返回 true。
- False - 如果请求者在 1 小时前已使用 MFA 进行登录，则条件返回 false。
- Not present (不存在) - 如果请求者已使用其 IAM 用户访问密钥在 Amazon CLI 或 Amazon API 中发出请求，则键不存在。在此情况下，键不存在，并且它不匹配。
- Null - 对于用户定义的条件键（如在请求中传递标签），可以包含一个空字符串。在此情况下，请求上下文中的值为 null。在某些情况下，null 值可能会返回 true。例如，如果您将多值 [ForAllValues](#) (p. 960) 条件运算符与 [aws:TagKeys](#) (p. 1011) 条件键结合使用，则当请求上下文返回 null 时，您将遇到意外结果。有关更多信息，请参阅 [aws:TagKeys](#) (p. 1011) 和 [使用多个键和值](#) (p. 960)。

条件块

以下示例显示 Condition 元素的基本格式：

```
"Condition": {"StringLike": {"s3:prefix": ["janedoe/*"]}}
```

请求中的值由条件键表示，在此示例中为 s3:prefix。将上下文密钥值与您指定为文本值的值进行比较，例如 janedoe/*。要进行的比较类型由[条件运算符](#) (p. 951) 指定（此处为 StringLike）。您可以使用典型的布尔比较（例如等于、大于和小于）来创建比较字符串、日期、数字等的条件。使用[字符串运算符](#) (p. 951) 或 [ARN 运算符](#) (p. 956) 时，还可以在条件值中使用[策略变量](#) (p. 964)。以下示例包括 aws:username 变量。

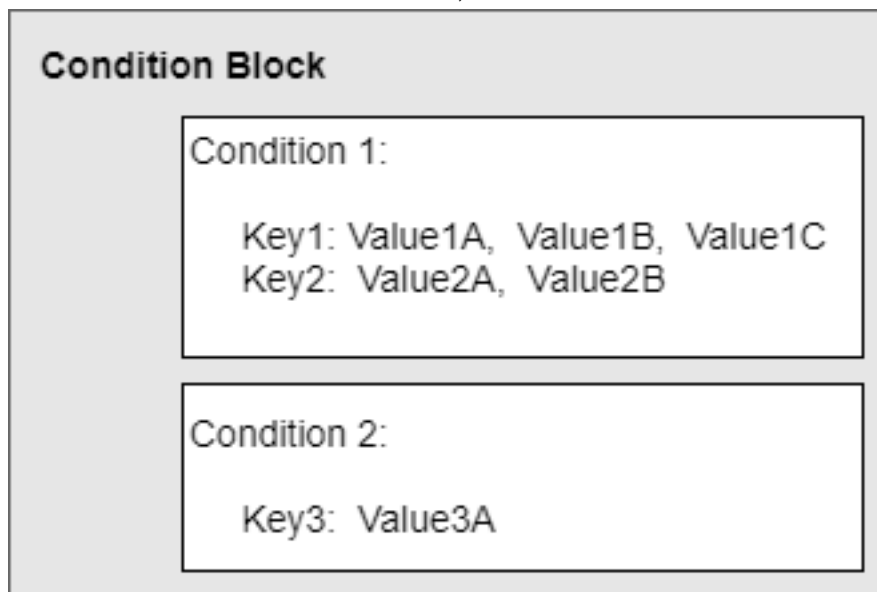
```
"Condition": {"StringLike": {"s3:prefix": ["${aws:username}/*"]}}
```

在某些情况下，键可能包含多个值。例如，对 Amazon DynamoDB 的请求可能需要从表返回或更新多个属性。访问 DynamoDB 表的策略可能包含 dynamodb:Attributes 键，此键包含请求中列出的所有属性。您可以使用 Condition 元素中的集合运算符，根据策略中的允许属性列表测试请求中的多个属性。有关更多信息，请参阅[创建具有多个键或值的条件](#) (p. 958)。

在请求期间评估策略时，Amazon 会将键替换为请求中的相应值。（在此示例中，Amazon 将使用请求的日期和时间。）将对条件进行评估以返回 true 或 false，然后再整体考虑策略是允许还是拒绝请求。

一个条件中包含多个值

一个 Condition 元素可以包含多个条件，而每个条件又可以包含多个键值对。下图对此进行了说明。



有关更多信息，请参阅[创建具有多个键或值的条件 \(p. 958\)](#)。

IAM JSON 策略元素：条件运算符

在 Condition 元素中使用条件运算符来将策略中的条件键和值与请求上下文中的值进行匹配。有关 Condition 元素的更多信息，请参阅[IAM JSON 策略元素：Condition \(p. 948\)](#)。

您可以在策略中使用的条件运算符取决于您选择的条件键。您可以选择全局条件键或特定于服务的条件键。要了解可用于全局条件键的条件运算符，请参阅[Amazon 全局条件上下文密钥 \(p. 998\)](#)。要了解可用于特定于服务的条件键的条件运算符，请参阅[Amazon 服务的操作、资源和条件键](#)并选择要查看的服务。

Important

如果请求上下文中没有您在策略条件中指定的键，则这些值不匹配，并且条件为 false。如果策略条件要求该键为不匹配，如 StringNotLike 或 ArnNotLike，并且右键不存在，则条件为 true。此逻辑适用于所有条件运算符，但 [...IfExists \(p. 957\)](#) 和 [Null check \(p. 958\)](#) 除外。这些运算符测试请求上下文中是否存在 (exists) 键。

条件运算符可分为以下类别：

- [字符串 \(p. 951\)](#)
- [数值 \(p. 953\)](#)
- [日期和时间 \(p. 953\)](#)
- [Boolean \(p. 954\)](#)
- [二进制 \(p. 955\)](#)
- [IP 地址 \(p. 955\)](#)
- [Amazon Resource Name \(ARN\) \(p. 956\)](#) (仅适用于某些服务。)
- [... IfExists \(p. 957\)](#) (检查密钥值是否作为另一检查的一部分存在)
- [空检查 \(p. 958\)](#) (检查密钥值是否作为独立检查存在)

字符串条件运算符

利用字符串条件运算符，您可以构建基于键与字符串值的对比来限制访问的 Condition 元素。

条件运算符	描述
StringEquals	精确匹配，区分大小写
StringNotEquals	否定匹配
StringEqualsIgnoreCase	精确匹配，忽略大小写
StringNotEqualsIgnoreCase	否定匹配，忽略大小写
StringLike	区分大小写的匹配。值可以在字符串中的任何位置包括多字符匹配的通配符 (*) 和单字符匹配的通配符 (?)。您必须指定通配符才能实现部分字符串匹配。 Note 如果一个键包含多个值，则 StringLike 可以用集合运算符 (ForAllValues:StringLike 和 ForAnyValue:StringLike) 来限定。有关更多信息，请参阅创建具有多个键或值的条件 (p. 958)。

条件运算符	描述
StringNotLike	不区分大小写的无效匹配。值可以在字符串中的任何位置包括多字符匹配的通配符 (*) 或单字符匹配的通配符 (?)。

例如，以下语句包含一个 Condition 元素，该元素使用 `aws:PrincipalTag` 键来指定必须将发出请求的主体为 `iamuser-admin` 作业类别。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::account-id:user/*",
    "Condition": {"StringEquals": {"aws:PrincipalTag/job-category": "iamuser-admin"}}
  }
}
```

如果请求上下文中没有您在策略条件中指定的键，则这些值不匹配。在此示例中，当主体使用附加了标签的 IAM 用户时，请求上下文中才会存在 `aws:PrincipalTag/job-category` 键。对于主体，如果使用附加了标签或会话标签的 IAM 角色，则也会包括它。如果没有标签的用户尝试查看或编辑访问键，则该条件返回 `false`，并且此语句将隐式拒绝请求。

您可以使用带有 String 条件运算符的[策略变量](#) (p. 964)。

以下示例使用 StringLike 条件运算符执行与[策略变量](#) (p. 964)的字符串匹配来创建策略，该策略允许 IAM 用户使用 Amazon S3 控制台管理其 Amazon S3 存储桶中的“主目录”。该策略允许对 S3 存储桶执行指定操作，前提是 `s3:prefix` 与任一指定模式相匹配。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::BUCKET-NAME",
      "Condition": {"StringLike": {"s3:prefix": [
        "",
        "home/",
        "home/${aws:username}/"
      ]}}
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}",
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}/*"
      ]
    }
  ]
}
```

有关说明了如何使用 Condition 元素基于 Web 联合身份验证的应用程序 ID 和用户 ID 限制对资源的访问的策略示例，请参阅[Amazon S3：允许 Amazon Cognito 用户访问其存储桶中的对象](#) (p. 431)。

数字条件运算符

利用数字条件运算符，您可以构建基于键与整数或小数值的对比来限制访问的 Condition 元素。

条件运算符	描述
NumericEquals	匹配
NumericNotEquals	否定匹配
NumericLessThan	“小于”匹配
NumericLessThanEquals	“小于或等于”匹配
NumericGreaterThan	“大于”匹配
NumericGreaterThanEquals	“大于或等于”匹配

例如，以下语句包含一个 Condition 元素，该元素使用 NumericLessThanEquals 条件运算符与 s3:max-keys 密钥来指定请求者一次最多可在 example_bucket 内列出 10 个对象。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket",
    "Condition": {"NumericLessThanEquals": {"s3:max-keys": "10"}}
  }
}
```

如果请求上下文中没有您在策略条件中指定的键，则这些值不匹配。在此示例中，当您执行 ListBucket 操作时，s3:max-keys 键始终存在于请求中。如果此策略允许所有 Amazon S3 操作，则只允许包含值小于或等于 10 的 max-keys 上下文密钥的操作。

您不能对[策略变量](#) (p. 964)使用 Numeric 条件运算符。

日期条件运算符

利用日期条件运算符，您可以构建基于键与日期/时间值的对比来限制访问的 Condition 元素。您可以将这些条件运算符与 [aws:CurrentTime](#) 键或 [aws:EpochTime](#) 键配合使用。您必须指定日期/时间值，且其中一个 [W3C 实现](#)要采用 ISO 8601 日期格式或新纪元 (UNIX) 时间格式。

Note

日期条件运算符不允许使用通配符。

条件运算符	描述
DateEquals	匹配特定日期
DateNotEquals	否定匹配
DateLessThan	在特定日期和时间之前匹配。
DateLessThanEquals	在特定日期和时间或之前匹配

条件运算符	描述
DateGreaterThan	在特定日期和时间之后匹配
DateGreaterThanEquals	在特定日期和时间或之后匹配

例如，以下语句包含一个 Condition 元素，该元素使用 DateGreaterThan 条件运算符与 `aws:TokenIssueTime` 键。此条件指定用于发出请求的临时安全凭证在 2020 年签发。此策略可以通过编程方式每天更新，以确保账户成员使用最新的凭证。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::account-id:user/*",
    "Condition": {"DateGreaterThan": {"aws:TokenIssueTime": "2020-01-01T00:00:01Z"}}
  }
}
```

如果请求上下文中没有您在策略条件中指定的键，则这些值不匹配。只有在主体使用临时凭证发出请求时，才会在请求上下文中包含 `aws:TokenIssueTime` 键。使用访问密钥发出的 Amazon CLI、Amazon API 或 Amazon 开发工具包请求中未提供此键。在此示例中，如果 IAM 用户尝试查看或编辑访问键，请求将被拒绝。

您不能对策略变量 (p. 964) 使用 Date 条件运算符。

布尔值条件运算符

利用布尔值条件，您可以构建基于键与“正确”或“错误”的对比来限制访问的 Condition 元素。

条件运算符	描述
Bool	布尔值匹配

例如，此基于身份的策略使用 Bool 条件运算符与 `aws:SecureTransport` 键拒绝对存储桶及其内容的所有 S3 操作（如果请求不是通过 SSL 进行的）。

Important

该策略不允许进行任何操作。可将此策略与允许特定操作的其他策略结合使用。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "BooleanExample",
      "Action": "s3:*",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:s3::DOC-EXAMPLE-BUCKET",
        "arn:aws:s3::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      }
    }
  ]
}
```



```
]
}
```

如果请求上下文中没有您在策略条件中指定的键，则这些值不匹配。`aws:SecureTransport` 键始终存在于请求上下文中。

您可以使用带有 Boolean 条件运算符的[策略变量 \(p. 964\)](#)。

二进制条件运算符

利用 `BinaryEquals` 条件运算符，您可以构建测试二进制格式键值的 `Condition` 元素。它会比较指定密钥字节的值和策略中 [base-64](#) 编码表示的二进制值。

```
"Condition": {
  "BinaryEquals": {
    "key": "QmluYXJ5VmFsdWVJbkJhc2U2NA=="
  }
}
```

如果请求上下文中没有您在策略条件中指定的键，则这些值不匹配。

您不能对[策略变量 \(p. 964\)](#)使用 `Binary` 条件运算符。

IP 地址条件运算符

利用 IP 地址条件运算符，您可以构建 `Condition` 元素，它们会基于键与 IPv4 或 IPv6 地址或 IP 地址范围的对比来限制访问。可以用 `aws:SourceIp` 键使用它们。该值必须采用标准的 CIDR 格式 (例如 203.0.113.0/24 或 2001:DB8:1234:5678::/64)。如果您指定的 IP 地址没有关联的路由前缀，IAM 将使用 `/32` 作为默认前缀值。

某些 Amazon 服务支持 IPv6，使用 `::` 表示一系列 0。要了解某项服务是否支持 IPv6，请参阅该服务的文档。

条件运算符	描述
<code>IpAddress</code>	指定的 IP 地址或范围
<code>NotIpAddress</code>	除指定 IP 地址或范围外的所有 IP 地址

例如，下列语句使用 `IpAddress` 条件运算符与 `aws:SourceIp` 键来指定必须从 IP 范围 203.0.113.0 至 203.0.113.255 发出请求。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::account-id:user/*",
    "Condition": {"IpAddress": {"aws:SourceIp": "203.0.113.0/24"}}
  }
}
```

`aws:SourceIp` 条件键解析为发出请求的 IP 地址。如果请求源自 Amazon EC2 实例，则 `aws:SourceIp` 计算为实例的公有 IP 地址。

如果请求上下文中没有您在策略条件中指定的键，则这些值不匹配。`aws:SourceIp` 键始终存在于请求上下文中，但请求者使用 VPC 终端节点发出请求时除外。在这种情况下，条件返回 `false`，并且此语句隐式拒绝请求。

您不能对[策略变量](#) (p. 964)使用 IP Address 条件运算符。

下面的示例说明如何结合使用 IPv4 和 IPv6 地址来覆盖组织的所有有效 IP 地址。建议您使用您的 IPv6 地址范围以及已拥有的 IPv4 范围来增强组织的策略，以确保策略在您过渡到 IPv6 时继续有效。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "someservice:*",
    "Resource": "*",
    "Condition": {
      "IpAddress": {
        "aws:SourceIp": [
          "203.0.113.0/24",
          "2001:DB8:1234:5678::/64"
        ]
      }
    }
  }
}
```

如果以用户身份直接调用测试的 API，aws:SourceIp 条件键仅在 JSON 策略中有效。如果改为使用服务代表您调用目标服务，则目标服务看到的是进行调用的服务的 IP 地址而不是源用户的 IP 地址。举例来说，如果使用 Amazon CloudFormation 调用 Amazon EC2 来构建实例，则会发生这种情况。目前，无法通过进行调用的服务将源 IP 地址传递给目标服务以在 JSON 策略中进行评估。对于这些服务 API 调用类型，请勿使用 aws:SourceIp 条件键。

Amazon Resource Name (ARN) 条件运算符

利用 Amazon Resource Name (ARN) 条件运算符，您可以构建基于键与 ARN 的对比来限制访问的 Condition 元素。ARN 被视为一个字符串。

条件运算符	描述
ArnEquals, ArnLike	区分大小写的 ARN 匹配。ARN 的六个由冒号分隔开的部分都要单独检查，每一个部分都可包括一个多字符匹配通配符 (*) 或一个单字符匹配通配符 (?)。ArnEquals 和 ArnLike 条件运算符的行为相同。
ArnNotEquals, ArnNotLike	ARN 无效匹配。ArnNotEquals 和 ArnNotLike 条件运算符的行为相同。

在某些情况下，当 ARN 运算符不匹配时，字符串运算符会匹配。在这种情况下，请尝试使用[字符串条件运算符](#) (p. 951)。

例如，如果使用以下模式进行匹配：

```
arn:aws:someservice*:111122223333:finance/*
```

而请求中包含以下值：

```
arn:aws:someservice:us-east-2:999999999999:store/abc:111122223333:finance/document.txt
```

借助 StringLike 条件，匹配成功。第一个星号匹配 us-east-2:999999999999:store/abc:。使用 ArnLike 条件，该条件匹配冒号之间的元素，匹配失败。第一个星号仅匹配 us-east-2，而未能匹配 999999999999:store/abc:。

您可以使用带有 ARN 条件运算符的[策略变量](#) (p. 964)。

以下基于资源的策略示例显示附加到 Amazon SQS 队列（您希望向其发送 SNS 消息的队列）的策略。它授予 Amazon SNS 将消息发送到所选队列的权限，但仅在该服务代表特定 Amazon SNS 主题发送这些消息时才授予此权限。您在 Resource 字段中指定队列，Amazon SNS 主题则作为 SourceArn 密钥的值。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "123456789012"},
    "Action": "SQS:SendMessage",
    "Resource": "arn:aws:sqs:REGION:123456789012:QUEUE-ID",
    "Condition": {"ArnEquals": {"aws:SourceArn": "arn:aws:sns:REGION:123456789012:TOPIC-ID"}}
  }
}
```

如果请求上下文中没有您在策略条件中指定的键，则这些值不匹配。仅在资源触发一项服务以代表资源拥有者调用另一项服务时，`aws:SourceArn` 键存在于请求上下文中。如果 IAM 用户尝试直接执行此操作，则条件返回 false，并且此语句隐式拒绝请求。

...IfExists 条件运算符

除 Null 条件外，您可在任何条件运算符名称的末尾添加 `IfExists`，例如，`StringLikeIfExists`。如果您是指“如果请求的内容中存在策略键，则依照策略所述来处理键。如果该键不存在，则条件元素的计算结果将为 true。”语句中其他条件因素仍然可以导致不匹配，但使用 `...IfExists` 检查时没有缺失键。如果您使用带有否定条件运算符（如 `StringNotEqualsIfExists`）的 `"Effect": "Deny"` 元素，则即使缺少标签，请求仍会被拒绝。

使用 `IfExists` 的示例

许多条件键描述有关特定类型的资源的信息，仅当访问该类型的资源时才存在。这些条件键在其他类型的资源上不存在。当策略语句仅适用于一种类型的资源时，这不会导致问题。但是，有时单个语句可以适用于多种类型的资源，例如当策略语句从多个服务引用操作时，或是当服务中的给定操作访问同一服务中的多种不同资源类型时。在这种情况下，在策略语句中包含仅适用于一种资源的条件键可能会导致策略语句中的 `Condition` 元素失败，从而使语句的 `"Effect"` 不适用。

例如，请考虑以下策略示例：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "THISPOLICYDOESNOTWORK",
    "Effect": "Allow",
    "Action": "ec2:RunInstances",
    "Resource": "*",
    "Condition": {"StringLike": {"ec2:InstanceType": [
      "t1.*",
      "t2.*",
      "m3.*"
    ]}}
  }
}
```

上述策略的意图是使用户可以启动类型为 t1、t2 或 m3 的任何实例。但是，启动实例实际上要求访问除了实例本身之外的许多资源；例如映像、密钥对、安全组等。会针对启动实例所需的每个资源来评估整个语句。这些其他资源没有 `ec2:InstanceType` 条件键，因此 `StringLike` 检查会失败，并且不会向用户授予启动任何实例类型的能力。要解决此问题，请改用 `StringLikeIfExists` 条件运算符。这样，仅当条件键

存在时才会进行测试。您会读到以下内容：“If the resource being checked has an "ec2:InstanceType" condition key, then allow the action only if the key value begins with "t1.*", "t2.*", or "m3.*". If the resource being checked does not have that condition key, then don't worry about it.” DescribeActions 语句包含在控制台中查看实例所需的操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RunInstance",
      "Effect": "Allow",
      "Action": "ec2:RunInstances",
      "Resource": "*",
      "Condition": {
        "StringLikeIfExists": {
          "ec2:InstanceType": [
            "t1.*",
            "t2.*",
            "m3.*"
          ]
        }
      }
    },
    {
      "Sid": "DescribeActions",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeImages",
        "ec2:DescribeInstances",
        "ec2:DescribeVpcs",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups"
      ],
      "Resource": "*"
    }
  ]
}
```

用于检查条件键是否存在的条件运算符

使用 Null 条件运算符检查授权时是否缺少条件键。在策略语句中使用 true (密钥不存在 - 为 null) 或 false (密钥存在且值不为 null)。

您不能对[策略变量](#) (p. 964)使用 Null 条件运算符。

例如，您可以使用此条件运算符确定用户使用的是自己的针对该操作的凭证还是临时凭证。如果用户使用的是临时凭证，则键 aws:TokenIssueTime 存在并具有一个值。以下示例说明了一个条件，该条件说明用户在使用 Amazon EC2 API 时不能使用临时凭证 (键不能存在)。

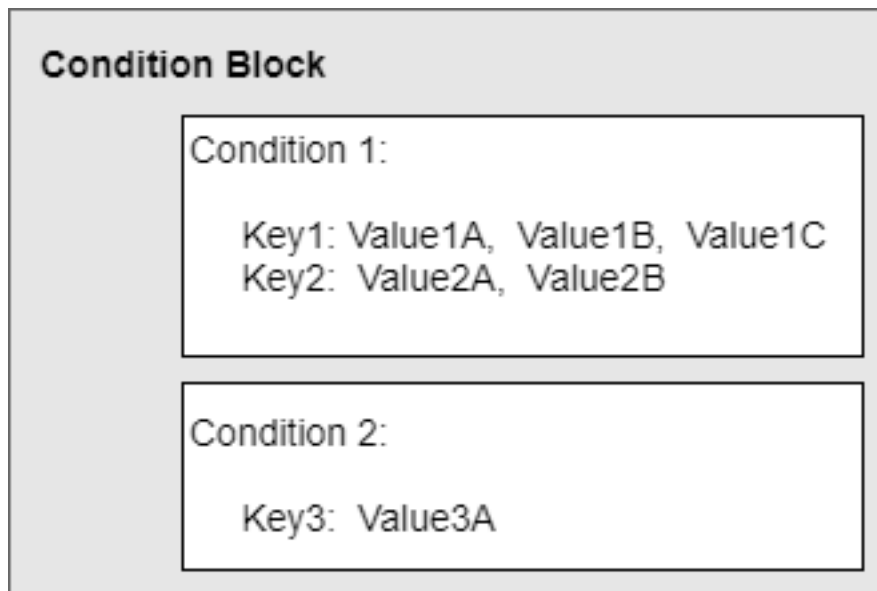
```
{
  "Version": "2012-10-17",
  "Statement": {
    "Action": "ec2:*",
    "Effect": "Allow",
    "Resource": "*",
    "Condition": { "Null": { "aws:TokenIssueTime": "true" } }
  }
}
```

创建具有多个键或值的条件

您可以使用策略的 Condition 元素测试请求中的多个键或单个键的多个值。在以编程方式或通过 Amazon 向 Amazon Web Services Management Console 发出请求时，请求包含有关您的主体、操作和标签等的信

息。要了解请求中包含的信息和数据，请参阅[请求 \(p. 5\)](#)。您可以使用条件键来测试请求中的匹配键的值。例如，您可以使用条件键根据标签控制对 DynamoDB 表的特定属性或 Amazon EC2 实例的访问。

一个 Condition 元素可以包含多个条件，而每个条件又可以包含多个键值对。大多数条件键支持使用多个值。下图对此进行了说明。除非另行指定，否则所有键均可有多个值。

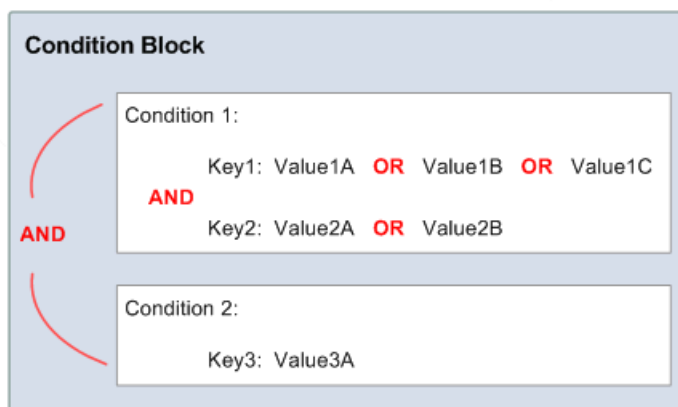


主题

- [具有多个键或值的条件的评估逻辑 \(p. 959\)](#)
- [使用多个键和值 \(p. 960\)](#)
- [将多个值与条件集合运算符一起使用的示例 \(p. 961\)](#)
- [将多个值与条件集合运算符一起使用的评估逻辑 \(p. 962\)](#)

具有多个键或值的条件的评估逻辑

如果您的策略具有多个[条件运算符 \(p. 951\)](#)或将多个键附加到单个条件运算符，则使用逻辑 AND 评估条件。如果单个条件运算符包含一个键的多个值，则采用逻辑 OR 评估该条件运算符。要调用所需的 Allow 或 Deny 效果，所有条件都必须解析为 true。



例如，以下条件块显示了上图在策略中的呈现方式。此条件块使用条件运算符 `StringEqualsIgnoreCase` 和 `StringEquals`，以及条件键 `aws:PrincipalTag` 和 `aws:PrincipalAccount`。要调用所需的

Allow 或 Deny 效果，所有条件都必须解析为 true。提出请求的用户必须同时拥有 department 和 role 这两个主体标签键，包括策略中指定的一个标签键值。此外，发出请求的用户必须属于策略中指定的 Amazon 账户标识符。

```
"Condition": {
  "StringEqualsIgnoreCase": {
    "aws:PrincipalTag/department": [ "finance", "hr", "legal" ],
    "aws:PrincipalTag/role": [ "audit", "security" ]
  },
  "StringEquals": {
    "aws:PrincipalAccount": "123456789012"
  }
}
```

Note

当多个值针对否定匹配条件运算符 (StringNotEquals 和 DateNotEquals) 在策略中列出时，有效权限会像 AND 那样工作。例如，如果在 StringNotEquals 条件运算符中有多个 aws:PrincipalAccount 值，则字符串无法匹配用来将条件转换为 true 所罗列的 aws:PrincipalAccount 数值。

使用多个键和值

要将条件与具有多个键值的[请求上下文 \(p. 5\)](#)进行比较，必须使用 ForAllValues 或者 ForAnyValue 集合运算符。这些集合运算符用于比较两组值。

当策略条件比较两组值（例如请求中的标签集和策略条件中的标签集）时，您需要告诉 Amazon 如何比较集合。IAM 定义了两个集合运算符，ForAnyValue 和 ForAllValues。您必须将集合运算符用于多值条件键。不要将集合运算符用于单值条件键。

要确定 Amazon 条件键为单值还是多值，请查看其数据类型。要查看用于一项服务的条件键数据类型，请参阅[Amazon 服务的操作、资源和条件键](#)，然后选择要查看其条件键的服务。如果条件 Type（类型）字段包含 ArrayOf 前缀，则条件键为多值。例如，[Amazon Key Management Service](#) 支持 String 单值数据类型和 ArrayOfString 多值数据类型的条件键。

- 单值条件键在请求上下文中最多有一个值。例如，您可以在 Amazon 中标记资源。资源标签存储为标签键值对。资源标签键可以具有单个标签值。因此，aws:ResourceTag/tag-key 是一个单值条件键。不要将集合运算符用于单值条件键。
- 多值条件键在请求上下文中可以有多个值。例如，您可以在 Amazon 中标记资源并在请求中包含多个标签键值对。因此，aws:TagKeys 是一个多值条件键。多值条件键需要一个集合运算符。

限定词 ForAllValues 和 ForAnyValue 为条件运算符添加了集合运算功能，以便您针对多个条件值测试多个请求值。此外，如果在策略中包含带有通配符或变量的多值字符串键，则还必须使用 StringLike [条件运算符 \(p. 951\)](#)。对于包含单个键的多个值的请求，必须将条件键值括在方括号内，就像[数组 \(p. 986\)](#)一样。例如："Key2":["Value2A", "Value2B"]。

- ForAllValues – 与多值条件键一起使用。测试请求集的每个成员的值是否为条件键集的子集。如果请求中的每个键值均与策略中的至少一个值匹配，则条件返回 true。如果请求中没有键或者键值解析为空数据集（如空字符串），则也会返回 true。不要使用带有 Allow 效果的 ForAllValues，因为这样可能会过于宽容。
- ForAnyValue – 与多值条件键一起使用。测试请求值集的至少一个成员是否与条件键值集的至少一个成员匹配。如果请求中的任何一个键值与策略中的任何一个条件值匹配，则条件返回 true。对于没有匹配的键或空数据集，条件返回 false。

Note

单值条件键和多值条件键之间的差异取决于请求上下文中的值数量，而不是策略条件中的值数量。

将多个值与条件集合运算符一起使用的示例

您可以创建一个策略，以根据您在策略中指定的一个或多个值测试请求中的多个值。假设有一个名为 Thread 的 Amazon DynamoDB 表用来存储某技术支持论坛中有关话题的信息。该表具有名为 ID、UserName、PostDateTime、Message 和 Tags 的属性。

```
{
  ID=101
  UserName=Bob
  PostDateTime=20130930T231548Z
  Message="A good resource for this question is docs.aws.amazon.com"
  Tags=["AWS", "Database", "Security"]
}
```

有关如何在 DynamoDB 中使用集合运算符来实现对单个数据项和属性的精细访问的信息，请参阅 Amazon DynamoDB 开发人员指南中的[适用于 DynamoDB 的精细访问控制](#)。

您可以创建一个策略以允许用户仅查看 PostDateTime、Message 和 Tags 属性。如果用户的请求包含其中的任何属性，则允许该请求。但如果请求包含任何其他属性（例如 ID），则拒绝该请求。从逻辑上讲，您希望创建允许的属性列表 (PostDateTime, Message, Tags)。您还希望在策略中指示，用户请求的所有属性必须位于该允许的属性列表中。

以下示例策略显示如何将 ForAllValues 限定词与 StringEquals 条件运算符搭配使用。该条件仅允许用户请求名为 Thread 的 DynamoDB 表中的 ID、Message 或 Tags 属性。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "dynamodb:GetItem",
      "Resource": "arn:aws:dynamodb:*:*:table/Thread",
      "Condition": {
        "ForAllValues:StringEquals": {
          "dynamodb:Attributes": [
            "ID",
            "Message",
            "Tags"
          ]
        }
      }
    }
  ]
}
```

假设用户向 DynamoDB 发出请求以从 Thread 表中获取 Message 和 Tags 属性。在这种情况下，允许该请求，因为用户请求的属性均与策略中指定的值匹配。GetItem 操作要求用户将 ID 属性作为数据库表键传递（策略也允许这样做）。不过，如果用户的请求包含 UserName 属性，请求将会失败。原因是 UserName 不在允许的属性列表中，并且 ForAllValues 限定词要求所有请求的值在策略中列出。

Important

如果使用 dynamodb:Attributes，您必须指定表的所有主键和索引键属性的名称。您还必须指定在策略中列出的任何二级索引。否则，DynamoDB 无法使用这些键属性执行所请求的操作。

或者，您可能希望确保明确禁止用户在请求中包含某些属性，如 ID 和 UserName 属性。例如，您可能在用户更新 DynamoDB 表时排除一些属性，因为更新（PUT 操作）不应更改特定属性。在这种情况下，您创建禁止的属性列表 (ID, UserName)。如果用户请求的任何属性与任何禁止的属性匹配，则拒绝该请求。

以下示例说明如何使用 ForAnyValue 限定词以在用户尝试执行 ID 操作时拒绝对 PostDateTime 和 PutItem 属性的访问。也就是说，在用户尝试更新 Thread 表中的任一属性的情况下。请注意，Effect 元素设置为 Deny。


```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "dynamodb:PutItem",
    "Resource": "arn:aws:dynamodb:*:*:table/Thread",
    "Condition": {
      "ForAnyValue:StringEquals": {
        "dynamodb:Attributes": [
          "ID",
          "PostDateTime"
        ]
      }
    }
  }
}
```

假定用户请求更新 `PostDateTime` 表的 `Message` 和 `Thread` 属性。`ForAnyValue` 限定词可确定是否有请求的属性在策略列表中出现。在这种情况下，只有一项匹配 (`PostDateTime`)，因此，条件为 `true`。假设请求中的其他值也可以满足 (例如，资源)，则总策略评估返回 `true`。因为策略的效果为 `Deny`，请求被拒绝。

假设用户改为只请求对 `PutItem` 属性执行 `UserName`。请求中没有任何属性 (仅 `UserName`) 与策略中列出的任何属性 (`ID`、`PostDateTime`) 相匹配。条件将返回 `false`，因此，策略 (`Deny`) 的效果也为 `false`，该策略不会拒绝该请求。(为使请求取得成功，必须由不同的策略显式允许它。此策略没有显式拒绝它，但所有请求都被隐式拒绝。)

Warning

当您使用 `ForAllValues` 条件运算符时，如果请求中没有键或者键值解析为空数据集 (如空字符串)，则返回 `true`。如要求该请求至少包含一个值，则您必须在策略中使用另一个条件。有关示例，请参阅 [在 Amazon 请求期间控制访问 \(p. 390\)](#)。

将多个值与条件集合运算符一起使用的评估逻辑

本部分讨论有关与 `ForAllValues` 和 `ForAnyValue` 运算符结合使用的评估逻辑的详细信息。下表显示请求中可能包含的键 (`PostDateTime` 和 `UserName`)，以及一个包含 `PostDateTime`、`Message` 和 `Tags` 值的策略条件。

键 (在请求中)	条件值 (在策略中)
<code>PostDateTime</code>	<code>PostDateTime</code>
<code>UserName</code>	<code>Message</code>
	<code>Tags</code>

组合的评估如下：

<code>PostDateTime matches PostDateTime?</code>
<code>PostDateTime matches Message?</code>
<code>PostDateTime matches Tags?</code>
<code>UserName matches PostDateTime?</code>
<code>UserName matches Message?</code>
<code>UserName matches Tags?</code>

条件运算符结果取决于策略条件使用了哪个修饰符：

- **ForAllValues**. 如果请求中的每个键 (`PostDateTime` 或 `UserName`) 至少与策略中的一个条件值 (`PostDateTime`、`Message`、`Tags`) 匹配, 则条件运算符将返回 `true`。换言之, 为了使条件返回 `true`, `PostDateTime` 必须等于 `PostDateTime`、`Message` 或 `Tags` 并且 `UserName` 必须等于 `PostDateTime`、`Message` 或 `Tags`。
- **ForAnyValue**. 如果请求值和策略值的任意组合 (示例中的六种组合之一) 返回 `true`, 则条件运算符将返回 `true`。

以下策略包含一个 `ForAllValues` 限定词：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "dynamodb:GetItem",
    "Resource": "arn:aws:dynamodb:*:*:table/Thread",
    "Condition": {
      "ForAllValues:StringEquals": {
        "dynamodb:Attributes": [
          "PostDateTime",
          "Message",
          "Tags"
        ]
      }
    }
  }
}
```

假设用户向 DynamoDB 请求获取 `PostDateTime` 和 `UserName` 属性。组合的评估如下：

PostDateTime matches PostDateTime?	True
PostDateTime matches Message?	False
PostDateTime matches Tags?	False
UserName matches PostDateTime?	False
UserName matches Message?	False
UserName matches Tags?	False

策略包含 `ForAllValues` 条件运算符修饰符, 这意味着 `PostDateTime` 和 `UserName` 必须至少各有一个匹配项。`UserName` 没有匹配项, 因此条件运算符将返回 `false`, 且策略将拒绝请求。

以下策略包含一个 `ForAnyValue` 限定词：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "dynamodb:PutItem",
    "Resource": "arn:aws:dynamodb:*:*:table/Thread",
    "Condition": {
      "ForAnyValue:StringEquals": {
        "dynamodb:Attributes": [
          "ID",
          "PostDateTime"
        ]
      }
    }
  }
}
```

```

    }
  }
}

```

请注意，策略包含 "Effect": "Deny"，操作是 PutItem。假设用户的 PutItem 请求包含属性 UserName、Message 和 PostDateTime。评估如下：

UserName matches ID?	False
UserName matches PostDateTime?	False
Messages matches ID?	False
Message matches PostDateTime?	False
PostDateTime matches ID?	False
PostDateTime matches PostDateTime?	True

使用修饰符 ForAnyValue 后，如果任何一个测试返回 true，则条件将返回 true。最后一个测试返回 true，因此条件也将返回 true；因为 Effect 元素设置为 Deny，因此请求将被拒绝。

Note

如果请求中的键值解析为空数据集（例如，空字符串），ForAllValues 修饰的条件运算符将返回 true。此外，ForAnyValue 修饰的条件运算符返回 false。

单值和多值条件密钥

在一次 API 调用的 [请求上下文](#) (p. 5) 中与条件相关联的潜在值数会使条件成为单值或多值，而不是策略条件中列出的值的数量。单值条件键在 API 调用的请求上下文中最多具有一个值。在 API 调用的请求上下文中，多值条件键可以有多个值。

您可以使用任何可用的单值条件密钥作为策略变量。您不能使用多值条件键作为策略变量。有关策略变量的更多信息，请参阅 [IAM policy 元素：变量和标签](#) (p. 964)。

多值条件键需要条件集合运算符 ForAllValues 或 ForAnyValue。包含键值对的条件键，例如 [the section called "RequestTag" \(p. 1007\)](#) 和 [the section called "ResourceTag" \(p. 1008\)](#) 可能会导致混乱，因为可能有多个 **tag-key** 值。但是因为每一个 **tag-key** 只能有一个值，aws:RequestTag 和 aws:ResourceTag 都是单值条件键。使用带有单值条件密钥的条件集运算符可能会导致过于宽容的策略。

IAM policy 元素：变量和标签

在编写策略时，如果您不知道资源或条件键的精确值，可以使用 Amazon Identity and Access Management (IAM) 策略变量作为占位符。

Note

如果 Amazon 无法解析变量，这可能会导致整个语句无效。例如，如果您使用 aws:TokenIssueTime 变量，只有在请求者（IAM 角色）使用临时凭证进行身份验证时，该变量才会解析为一个值。要防止变量导致无效的语句，请使用 [...IfExists 条件运算符](#) (p. 957)。

主题

- [介绍](#) (p. 965)
- [作为策略变量的标签](#) (p. 966)
- [您可以使用策略变量的位置](#) (p. 967)
- [可以用于策略变量的请求信息](#) (p. 968)

- [指定默认值 \(p. 971\)](#)
- [有关更多信息 \(p. 971\)](#)

介绍

在 IAM policy 中，您可通过很多操作来控制其访问权限的特定资源指定名称。例如，以下策略允许用户在 Amazon S3 存储桶 mybucket 中列出、读取和写入具有前缀 David 的对象。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket"],
      "Condition": {"StringLike": {"s3:prefix": ["David/*"]}}
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/David/*"]
    }
  ]
}
```

在某些情况下，您在编写策略时可能不知道资源的精确名称。您可能需要概括策略，这样无需为每个用户制作策略的唯一副本即可将该策略用于很多用户。例如，您要编写一条允许每位用户都可在 Amazon S3 存储桶中访问自己对象的策略，如上例所示。但不要为每个用户创建一个单独的策略，该策略明确指定用户的名称作为资源的一部分。相反，应创建一个适用于该组中的任何用户的组策略。

您可以通过使用策略变量实现这一点，该功能可让您在策略中指定占位符。当评估策略时，策略变量将被替换为来自请求本身上下文中的值。

Important

您可以使用任何可用的单值条件密钥作为变量。您不能使用多值条件键作为变量。

以下示例显示了使用策略变量的 Amazon S3 存储桶的策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket"],
      "Condition": {"StringLike": {"s3:prefix": ["${aws:username}/*"]}}
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/${aws:username}/*"]
    }
  ]
}
```

评估该策略时，IAM 将 `${aws:username}` 变量替换为当前实际用户 [便于记忆的名称 \(p. 896\)](#)。这意味着适用于一组用户的单个策略可以控制对存储桶的访问。它通过将用户名作为资源名称的一部分来实现此目的。

该变量使用 `$` 前缀标记，后跟一对大括号 (`{ }`)。在 `${ }` 字符内，可以包含想要在策略中使用的请求中的值名称。本页稍后将讨论您可以使用的值。

Note

为了使用策略变量，您必须在语句中包含 `Version` 元素，而且版本必须设置为支持策略变量的版本。变量是在 2012-10-17 版本中引入的。较早版本的策略语言不支持策略变量。如果您未添加 `Version` 元素，且没有将它设为相应的版本日期，则系统会将变量 (如 `${aws:username}`) 视为策略中的文字字符串。

`Version` 策略元素与策略版本不同。`Version` 策略元素用在策略之中，用于定义策略语言的版本。另一方面，当您更改 IAM 中的客户托管策略时，将创建一个策略版本。已更改的策略不会覆盖现有策略。而是由 IAM 创建新的托管策略版本。要了解 `Version` 策略元素的更多信息，请参阅 [the section called “Version” \(p. 934\)](#)。要了解策略版本的更多信息，请参阅 [the section called “IAM 策略版本控制” \(p. 458\)](#)。

您可以通过类似方式使用策略变量来允许每个用户管理自己的访问密钥。允许用户以编程方式更改用户 David 的访问密钥的策略如下所示：

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Action": ["iam:*AccessKey*"],
    "Effect": "Allow",
    "Resource": ["arn:aws:iam::account-id:user/David"]
  }]
}
```

如果该策略附加到用户 David，那么该用户可以更改自己的访问密钥。对于访问特定用户的 Amazon S3 对象的策略，您必须为每位用户创建包含用户名称的单独的策略，然后将每个策略附加到各个用户。

通过使用策略变量，您可以创建如下所示的策略：

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Action": ["iam:*AccessKey*"],
    "Effect": "Allow",
    "Resource": ["arn:aws:iam::account-id:user/${aws:username}"]
  }]
}
```

当您策略变量用于这样的用户名称时，不必为每个单独的用户创建单独的策略。您可以改为将新策略附加到 IAM 组，该组包含应允许管理自己的访问密钥的每个人。当用户发出修改其访问密钥的请求时，IAM 将使用当前请求中的用户名称替代 `${aws:username}` 变量并评估策略。

作为策略变量的标签

在某些 Amazon 服务中，您可以将自己的自定义属性附加到这些服务创建的资源。例如，您可以将标签应用于 Amazon S3 存储桶或者 IAM 用户。这些标签是键值对。您要定义标签键名称以及与该键名称关联的值。例如，您可以创建一个具有 **department** 键和 **Human Resources** 值的标签。有关标记 IAM 实体的更多信息，请参阅 [标记 IAM 资源 \(p. 276\)](#)。有关标记其他 Amazon 服务创建的资源的信息，请参阅该服务的文档。有关使用标签编辑器的信息，请参阅《Amazon Web Services Management Console 用户指南》中的 [使用标签编辑器](#)。

您可以标记 IAM 资源来简化对您的 IAM 资源的发现、组织和跟踪。您也可以标记 IAM 身份以控制对资源或标记本身的访问。要了解有关使用标签控制访问的更多信息，请参阅 [使用标签控制对 IAM 用户和角色的访问以及他们进行的访问 \(p. 388\)](#)。

您可以使用策略变量的位置

您可以在 Resource 元素中和 Condition 元素的字符串比较中使用策略变量。

资源元素

您可以在 Resource 元素中使用策略变量，但只能在 ARN 的资源部分中使用策略变量。ARN 的这一部分出现在第五个冒号 (:) 之后。不能使用变量来替换 ARN 中第五个冒号之前的部分，例如服务或账户。有关 ARN 格式的更多信息，请参见 [IAM ARN \(p. 896\)](#)。

以下策略可以关联到组。它给予 群组中的每位用户以编程方式访问 Amazon S3 中特定用户的数据元（自己的“主目录”）的完全权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3::mybucket"],
      "Condition": {"StringLike": {"s3:prefix": ["${aws:username}/*"]}}
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3::mybucket/${aws:username}/*"]
    }
  ]
}
```

Note

此示例使用 `aws:username` 键，它将返回用户的易记名称（比如“Adele”或“David”）。在某些情况下，您可能想要改用 `aws:userid` 键，它是全局唯一的值。有关更多信息，请参阅 [唯一标识符 \(p. 900\)](#)。

以下策略可用于 IAM 组。它为该组中的用户授予权限，以创建、使用和删除包含其名称以及位于 us-east-2 区域的队列。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListForConsole",
      "Effect": "Allow",
      "Action": "sqs:ListQueues",
      "Resource": "*"
    },
    {
      "Sid": "AllQueueActions",
      "Effect": "Allow",
      "Action": "sqs:*",
      "Resource": "arn:aws:sqs:us-east-2:*:${aws:username}-queue"
    }
  ]
}
```

要将 ARN 的一部分替换为标签值，请用 `${}` 将前缀和键名称括起。例如，以下 Resource 元素仅指与请求用户的 `department` 标签中的值具有相同的命名的存储桶。

```
"Resource": [ "arn:aws:s3:::bucket/${aws:PrincipalTag/department}" ]
```

条件元素

您可以在任何涉及字符串运算符或 ARN 运算符的条件下对 Condition 值使用策略变量。字符串运算符包括 StringEquals、StringLike 和 StringNotLike。ARN 运算符包括 ArnEquals 和 ArnLike。不能将策略变量与其他运算符（如 Numeric、Date、Boolean、Binary、IP Address 或 Null 运算符）一起使用。有关条件运算符的更多信息，请参阅 [IAM JSON 策略元素：条件运算符 \(p. 951\)](#)。

以下 Amazon SNS 主题策略让 Amazon 账户 999999999999 中的用户能够管理该主题（执行所有适用操作）。但是，仅当 URL 与其 Amazon 用户名匹配时，才会授予此权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {
        "AWS": "999999999999"
      },
      "Effect": "Allow",
      "Action": "sns:*",
      "Condition": {
        "StringLike": {
          "sns:endpoint": "https://example.com/${aws:username}/"
        },
        "StringEquals": {
          "sns:Protocol": "https"
        }
      }
    }
  ]
}
```

当引用 Condition 元素表达式中的标签时，请将相关的前缀和标签键用作条件键。然后，使用要在条件值中测试的值。例如，以下策略示例仅在标签 costCenter 附加到用户时才允许完全访问 IAM 用户。该标签还必须具有值 12345 或 67890。如果该标签没有值或具有任何其他值，则请求会失败。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:user*"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iam:ResourceTag/costCenter": [ "12345", "67890" ]
        }
      }
    }
  ]
}
```

可以用于策略变量的请求信息

您可以使用 JSON 策略的 Condition 元素将 [请求上下文 \(p. 972\)](#) 中的键与您在策略中指定的键值进行比较。当您使用策略变量时，Amazon 使用请求上下文密钥中的值替换策略中的变量。

在所有请求中可用的信息

策略包含您可以使用其值作为策略变量的键。（某些情况下，密钥不包含值 - 请参阅此列表后的信息。）

- **aws:CurrentTime** 这可用于检查日期和时间的条件。
- **aws:EpochTime** 这是用纪元或 Unix 时间表示的日期，用于日期/时间条件。
- **aws:TokenIssueTime** 这是颁发临时安全凭证的日期和时间，可用于日期/时间条件。注意：此键仅在使用临时安全凭证签名的请求中可用。有关临时安全凭证的更多信息，请参阅[IAM 临时安全凭证 \(p. 300\)](#)。
- **aws:PrincipalType** 该值表示主体是账户用户、联合身份用户还是担任的角色 - 请参阅后面的解释。
- **aws:SecureTransport** 这是一个布尔值，该值指示是否已使用 SSL 发送请求。
- **aws:SourceIp** 这是请求者的 IP 地址，用于 IP 地址条件。有关 [IP 地址条件运算符 \(p. 955\)](#) 何时有效以及何时应该改用 VPC 特定键的信息，请参阅 [SourceIp](#)。
- **aws:UserAgent** 此值是一个字符串，包含有关请求者的客户端应用程序的信息。此字符串由客户端生成，可能不可靠。您只能从 Amazon CLI 中使用此上下文密钥。
- **aws:userid** 该值是当前用户的唯一 ID - 参见下图。
- **aws:username** 这是一个包含当前用户的 [易记名称 \(p. 896\)](#) 的字符串 - 参见下图。
- **ec2:SourceInstanceARN** 这是从中发起请求的 Amazon EC2 实例的 Amazon Resource Name (ARN)。仅在请求来自使用与 EC2 实例配置文件关联的 IAM 角色的 Amazon EC2 实例时，才提供此密钥。

Important

密钥名称不区分大小写。例如，`aws:CurrentTime` 等同于 `AWS:currenttime`。

主体键值

`aws:username`、`aws:userid` 和 `aws:PrincipalType` 的值取决于发起请求的主体的类型。例如，可能使用 IAM 用户、IAM 角色或 Amazon Web Services 账户 根用户的凭证发出请求。以下列表为不同类型的主体显示这些键的值。

- Amazon Web Services 账户 根用户
 - `aws:username`: (不存在)
 - `aws:userid`: Amazon 账户 ID
 - `aws:PrincipalType`: Account
- IAM 用户
 - `aws:username`: **IAM ####**
 - `aws:userid`: **唯一 ID (p. 900)**
 - `aws:PrincipalType`: User
- 联合身份用户
 - `aws:username`: (不存在)
 - `aws:userid`: **#####**
 - `aws:PrincipalType`: FederatedUser
- Web 联合身份用户和 SAML 联合身份用户

Note

有关在使用 Web 联合身份验证时可用的策略键的信息，请参阅[使用 Web 联合身份验证识别用户的身份 \(p. 174\)](#)。

- `aws:username`: (不存在)
- `aws:userid`: (不存在)
- `aws:PrincipalType`: AssumedRole

- 担任的角色
 - `aws:username`: (不存在)
 - `aws:userid`: `## ID:#####`
 - `aws:PrincipalType`: `Assumed role`
- 分配给 Amazon EC2 实例的角色
 - `aws:username`: (不存在)
 - `aws:userid`: `## ID:EC2 ## ID`
 - `aws:PrincipalType`: `Assumed role`
- 匿名调用者 (仅限 Amazon SQS、Amazon SNS 和 Amazon S3)
 - `aws:username`: (不存在)
 - `aws:userid`: (不存在)
 - `aws:PrincipalType`: `Anonymous`

对于此列表，请注意以下事项：

- 不存在 表示值在当前请求信息中不存在，并且匹配该值的任何尝试都失败并导致语句无效。
- `## ID` 是在创建时分配给每个角色的唯一标识符。可以使用 Amazon CLI 命令显示角色 ID：`aws iam get-role --role-name rolename`
- `#####`和`#####` 是进行调用以获取临时凭证时调用进程 (例如应用程序或服务) 传递的名称。
- `ec2-instance-id` 是在实例启动时分配给它的值，显示在 Amazon EC2 控制台的 Instances (实例) 页面上。您还可以通过运行 Amazon CLI 命令来显示实例 ID：`aws ec2 describe-instances`

联合身份用户的请求中的可用信息

联合身份用户是使用系统而不是 IAM 验证的用户。例如，公司可能拥有调用 Amazon 的供内部使用的应用程序。为使用该应用程序的每位公司用户都提供 IAM 身份可能不现实。相反，该公司可能使用具有单一 IAM 身份的代理 (中间层) 应用程序，或该公司可能使用 SAML 身份提供程序 (IdP)。代理应用程序或 SAML IdP 会使用企业网络对单个用户进行身份验证。然后，代理应用程序可使用其 IAM 身份获取单个用户的临时安全凭证。实际上，SAML IdP 可以将身份信息交换为 Amazon 临时安全凭证。然后，临时凭证即可用于访问 Amazon 资源。

类似地，您可以创建用于移动设备的应用程序，该应用程序需要访问 Amazon 资源。在这种情况下，您可以使用 Web 联合身份验证，其中应用程序使用知名身份提供程序 (如 Login with Amazon、Amazon Cognito、Facebook 或 Google) 对用户进行身份验证。随后，应用程序可以使用这些提供商提供的用户身份验证信息来获取访问 Amazon 资源的临时安全凭证。

使用 Web 联合身份验证的推荐方法是利用 Amazon Cognito 和 Amazon 移动开发工具包。有关更多信息，请参阅下列内容：

- Amazon Mobile SDK for Android 开发人员指南中的 [Amazon Cognito 概述](#)
- Amazon Mobile SDK for iOS 开发人员指南中的 [Amazon Cognito 概述](#)
- [临时凭证的常见情形](#) (p. 301).

特定服务的信息

请求还可以在其请求上下文中包含特定服务的密钥和值。示例包括：

- `s3:prefix`
- `s3:max-keys`
- `s3:x-amz-acl`
- `sns:Endpoint`

- `sns:Protocol`

有关您可以用来获取策略变量值的特定服务键的信息，请参考与单个服务相关的文档。例如，请参阅以下主题：

- Amazon Simple Storage Service 用户指南中的 [Amazon S3 策略中的存储桶密钥](#)
- Amazon Simple Notification Service 开发人员指南中的 [Amazon SNS 密钥](#)。

特殊字符

有几个特殊预定义策略变量具有固定值，可用于表示字符（这些字符本身有特殊的含义）。如果这些特殊字符是您尝试匹配的字符串的一部分，而您原样插入这些字符，则不能正确进行解释。例如，在字符串中插入 * 星号会解释为与任何字符匹配的通配符（而不是解释为文本 *）。这种情况下，可以使用以下预定义策略变量：

- `${*}` - 在需要 *（星号）字符的位置使用。
- `${?}` - 在需要 ?（问号）字符的位置使用。
- `${$}` - 在需要 \$（美元符号）字符的位置使用。

在任何可以使用常规策略变量的字符串中，都可以使用这些预定义策略变量。

指定默认值

如果 Amazon 无法解析变量，这可能会导致整个语句无效。然而，向策略添加变量时，您可以为变量指定默认值。如果未为变量指定值，则 Amazon 会使用您提供的默认文本。

要向变量添加默认值，请用单引号（' '）括起默认值，并用逗号和空格（, ）分隔变量文本和默认值。

例如，如果使用主体标记 `team=yellow`，他们可以访问名为 `DOC-EXAMPLE-BUCKET-yellow` 的 `ExampleCorp`'s Amazon S3 存储桶。使用此资源的策略可能允许团队成员访问自己的团队存储桶，但不能访问其他团队的存储桶。对于没有团队标签的用户，它将原定设置值设为 `company-wide` 存储桶名称。这些用户只能访问 `DOC-EXAMPLE-BUCKET-company-wide` 存储桶，他们可以在其中查看广泛的信息，例如加入团队的说明。

```
"Resource": "arn:aws:s3::DOC-EXAMPLE-BUCKET-${aws:PrincipalTag/team, 'company-wide'}"
```

有关更多信息

欲了解更多有关策略的信息，请参阅：

- [IAM 中的策略和权限](#)。（p. 355）
- [IAM 基于身份的策略示例](#)（p. 391）
- [IAM JSON 策略元素参考](#)（p. 934）
- [策略评估逻辑](#)（p. 972）
- [关于 Web 联合身份验证](#)（p. 171）

IAM JSON 策略元素：支持的数据类型

本节列出了在 JSON 策略中指定值时支持的数据类型。策略语言并不针对每个策略元素支持所有类型；有关各个元素的信息，请参阅前面各部分。

- 字符串
- 数字（整数和浮动值）
- 布尔值

- Null
- 列表
- 映射
- 结构 (仅为嵌套映射)

以下图表将各个数据类型映射至串行化。注意所有策略必须为 UTF-8。有关 JSON 数据类型的信息，请转到 [RFC 4627](#)。

类型	JSON
字符串	字符串
整数	数字
Float	数字
布尔值	真假
Null	null
日期	字符串，符合 W3C Profile of ISO 8601
IpAddress	字符串，符合 RFC 4632
List	数组
对象	对象

策略评估逻辑

在主体尝试使用 Amazon Web Services Management Console、Amazon API 或 Amazon CLI 时，该主体将向 Amazon 发送请求。在 Amazon 服务收到请求时，Amazon 会完成几个步骤来确定是允许还是拒绝该请求。

1. 身份验证 - Amazon 首先对发出请求的主体进行身份验证（如有必要）。有一些服务（如 Amazon S3）不需要此步骤，它们允许来自匿名用户的某些请求。
2. [处理请求上下文 \(p. 972\)](#) - Amazon 处理在请求中收集的信息以确定应用于请求的策略。
3. [评估单个账户中的策略 \(p. 973\)](#) - Amazon 评估所有策略类型，这会影响策略的评估顺序。
4. [确定是允许还是拒绝账户内的请求 \(p. 974\)](#) - Amazon 然后根据请求上下文处理策略以确定是允许还是拒绝请求。

处理请求上下文

Amazon 处理请求以将以下信息收集到请求上下文中：

- Actions (or operations) (操作(或运营)) - 主体希望执行的操作。
- Resources (资源) - 对其执行操作的 Amazon 资源对象。
- Principal (主体) - 发送请求的用户、角色、联合身份用户或应用程序。有关主体的信息包括与该主体关联的策略。
- 环境数据 – 有关 IP 地址、用户代理、SSL 启用状态或当天时间的信息。
- 资源数据 – 与请求的资源相关的数据。这可能包括 DynamoDB 表名称或 Amazon EC2 实例上的标签等信息。

Amazon 随后使用此类信息来查找应用于请求上下文的策略。

评估单个账户中的策略

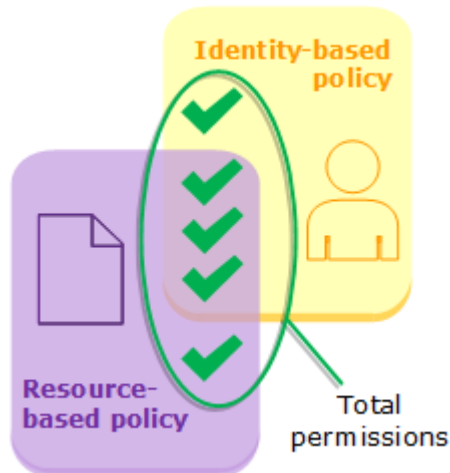
Amazon 如何评估策略取决于适用于请求上下文的策略类型。可在单个 Amazon 账户中使用以下策略类型，它们按使用频率顺序列出。有关这些策略类型的更多信息，请参阅[IAM 中的策略和权限](#) (p. 355)。要了解 Amazon 如何评估跨账户访问策略，请参阅[跨账户策略评估逻辑](#) (p. 981)。

1. 基于身份的策略 — 基于身份的策略附加到 IAM 身份（用户、用户组或角色）并向 IAM 实体（用户和角色）授予权限。如果仅基于身份的策略适用于请求，则 Amazon 检查所有这些策略以找到至少一个 Allow。
2. Resource-based policies（基于资源的策略）- 基于资源的策略将权限授予指定为主体的主体（账户、用户、角色和会话主体，如角色会话和 IAM 联邦用户）。权限定义主体可以对策略附加到的资源执行哪些操作。如果基于资源的策略和基于身份的策略同时适用于请求，则 Amazon 检查所有这些策略以找到至少一个 Allow。评估基于资源的策略时，策略中指定的主体 ARN 决定了其他策略类型中的隐式拒绝是否适用于最终决策。
3. IAM 权限边界 - 权限边界是一项高级功能，借助该功能，您可以设置基于身份的策略可以授予 IAM 实体（用户或角色）的最大权限。当您设置实体的权限边界时，该实体只能执行其基于身份的策略和其权限边界同时允许的操作。某些情况下，权限边界中的隐式拒绝可能会限制由基于资源的策略所授予的权限。要了解更多信息，请稍后参阅本主题中的[确定是允许还是拒绝账户内的请求](#) (p. 974)。
4. Amazon Organizations 服务控制策略 (SCP) - Organizations SCP 指定企业或企业单元 (OU) 的最大权限。SCP 最大权限适用于成员账户中主体的权限，包括每个 Amazon Web Services 账户根用户。如果 SCP 存在，仅当基于身份的策略和基于资源的策略以及 SCP 允许该操作时，才向成员账户中的主体授予这些权限。如果权限边界和 SCP 同时存在，则边界、SCP 以及基于身份的策略必须全部允许操作。
5. Session policies（会话策略）- 会话策略是高级策略，在以编程方式为角色或联合身份用户创建临时会话时，这些策略将作为参数进行传递。要以编程方式创建角色会话，请使用 AssumeRole* API 操作之一。在执行该操作并传递会话策略时，生成的会话的权限是 IAM 实体的基于身份的策略与会话策略的交集。要创建联合身份用户会话，您需要使用 IAM 用户的访问密钥以编程方式调用 GetFederationToken API 操作。基于资源的策略对会话策略权限评估具有不同的影响。这种差异取决于用户或角色的 ARN 还是会话的 ARN 在基于资源的策略中作为主体列出。有关更多信息，请参阅[会话策略](#) (p. 357)。

请记住，任一项策略中的显式拒绝将覆盖允许。

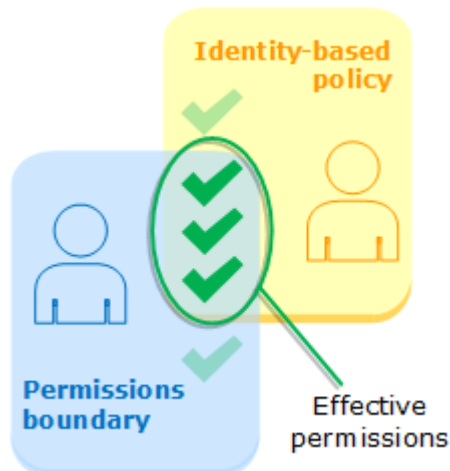
评估基于身份的策略以及基于资源的策略

基于身份的策略和基于资源的策略向策略所附加到的身份或资源授予权限。在 IAM 实体（用户或角色）请求访问同一账户中的资源时，Amazon 评估基于身份的策略和基于资源的策略授予的所有权限。生成的权限是指两种类型的权限的总和。如果基于身份的策略和/或基于资源的策略允许此操作，则 Amazon 允许执行该操作。其中任一项策略中的显式拒绝将覆盖允许。



评估具有权限边界的基于身份的策略

在 Amazon 评估用户的基于身份的策略和权限边界时，生成的权限是这两种类别的交集。这意味着，当您通过现有基于身份的策略向用户添加权限边界时，您可能会减少用户可以执行的操作。或者，当您从用户删除权限边界时，您可能会增加用户可以执行的操作。其中任一项策略中的显式拒绝将覆盖允许。要查看有关如何使用权限边界评估其他策略类型的信息，请参阅[评估具有边界的有效权限](#) (p. 371)。



评估具有 Organizations SCP 的基于身份的策略

在用户属于作为组织成员的账户时，生成的权限是用户的策略与 SCP 的交集。这意味着，操作必须由基于身份的策略和 SCP 同时允许。其中任一项策略中的显式拒绝将覆盖允许。



您可以在 Amazon Organizations 中了解[您的账户是否为某个组织的成员](#)。组织成员可能会受 SCP 的影响。要使用 Amazon CLI 命令或 Amazon API 操作查看该数据，您必须具有 Organizations 实体的 `organizations:DescribeOrganization` 操作的权限。您必须具有额外的权限才能在 Organizations 控制台中执行该操作。要了解 SCP 是否拒绝访问特定的请求或更改您的有效权限，请与您的 Amazon Organizations 管理员联系。

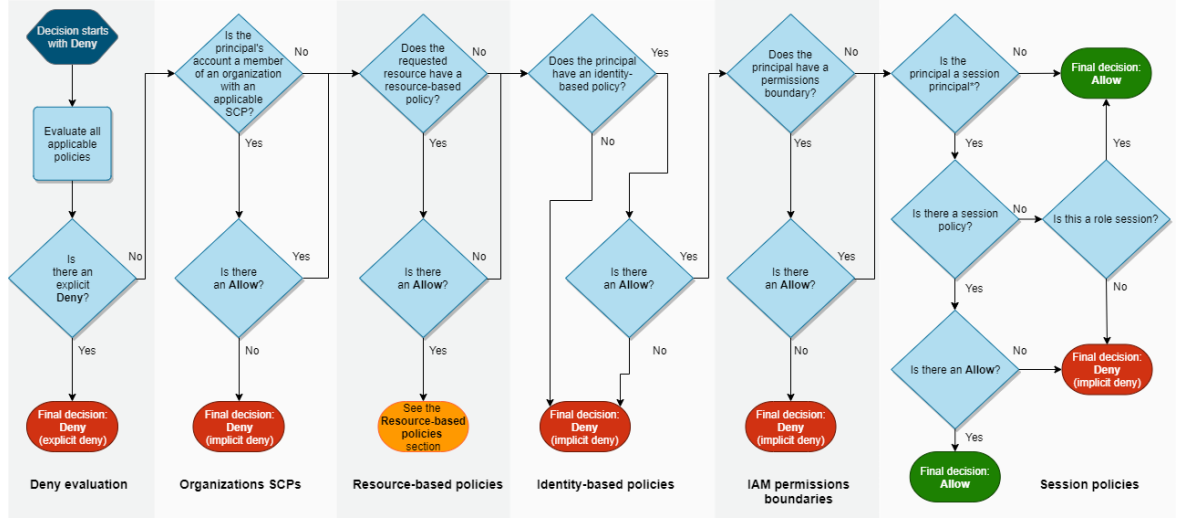
确定是允许还是拒绝账户内的请求

假设主体向 Amazon 发送请求以访问与主体的实体相同的账户中的资源。Amazon 执行代码决定是应该允许还是拒绝请求。Amazon 评估适用于请求上下文的所有策略。下面概括介绍了单个账户中有关这些策略的 Amazon 评估逻辑。

- 默认情况下，除 Amazon Web Services 账户根用户外，所有请求都被隐式否定，具有完全访问权限。

- 基于身份或基于资源的策略中的显式允许将覆盖此默认值。
- 如果存在权限边界、Organizations SCP 或会话策略，它可能会使用隐式拒绝覆盖允许。
- 任何策略中的显式拒绝将覆盖任何允许。

以下流程图提供了如何做出决定的详细信息。此流程图不涵盖基于资源的策略和其他类型策略中隐含拒绝的影响。



1. Deny evaluation (拒绝评估) - 默认情况下，所有请求都被拒绝。这称为 **隐式拒绝** (p. 980)。Amazon 执行代码评估账户中应用于请求的所有策略。其中包含 Amazon Organizations SCP、基于资源的策略、基于身份的策略、IAM 权限边界和会话策略。在所有这些策略中，执行代码查找应用于请求的 Deny 语句。这称为 **显式拒绝** (p. 980)。如果代码找到一个适用的显式拒绝，则代码将返回最终决定拒绝。如果没有显式拒绝，则代码会继续。
2. Organizations SCP - 然后，代码评估适用于请求的 Amazon Organizations 服务控制策略 (SCP)。SCP 适用于附加 SCP 的账户的主体。如果执行代码没有在 SCP 中找到任何适用的 Allow 语句，则隐式拒绝请求。代码将返回拒绝最终决定。如果没有 SCP，或者 SCP 允许所请求的操作，则代码继续。
3. 基于资源的策略 — 在同一账户中，基于资源的策略会对策略评估产生不同的影响，具体取决于访问资源的主体类型以及基于资源的策略中允许的主体。根据主体的类型，Allow 在基于资源的策略中可能会导致最终决定 Allow，即使基于身份的策略、权限边界或会话策略中存在隐式拒绝。

对于大多数资源，您只需要在基于身份的策略或基于资源的策略中明确授予主体访问权限。**IAM 角色信任策略** (p. 274) 和 **KMS 密钥策略** 是此逻辑的例外，因为它们必须明确允许 **主体** (p. 936) 的访问权限。

当指定的主体是 IAM 用户、IAM 角色或会话主体时，基于资源的策略逻辑会与其他策略类型有所不同。会话主体包括 **IAM 角色会话** (p. 938) 或者 **IAM 联合身份用户会话** (p. 940)。如果基于资源的策略直接向提出请求的 IAM 用户或会话主体授予权限，则基于身份的策略、权限边界或会话策略中的隐式拒绝不会影响最终决策。

如果基于身份的策略、权限边界和会话策略中存在隐式拒绝，则下表有助于您了解基于资源的策略对不同主体类型的影响。

基于资源的策略和其他策略类型的隐式拒绝（同一账户）

提出请求的主体	基于资源的策略	基于身份的策略	权限边界	会话策略	结果	Reason
IAM 角色	不适用	不适用	不适用	不适用	不适用	角色本身无法提出请求。在担任

提出请求的主体	基于资源的策略	基于身份的策略	权限边界	会话策略	结果	Reason
						角色之后，向角色会话发出请求。
IAM 角色会话	允许角色 ARN	隐式拒绝	隐式拒绝	隐式拒绝	DENY	权限边界和会话策略将作为最终决策的一部分进行评估。任何一项策略中的隐式拒绝都会导致拒绝决定。
IAM 角色会话	允许角色会话 ARN	隐式拒绝	隐式拒绝	隐式拒绝	允许	权限直接授予会话。其他策略类型不影响决策。
IAM 用户	允许 IAM 用户 ARN	隐式拒绝	隐式拒绝	不适用	允许	权限直接授予会话。其他策略类型不影响决策。
IAM 联合身份用户 (GetFederationToken)	允许 IAM 用户 ARN	隐式拒绝	隐式拒绝	隐式拒绝	DENY	权限边界或会话策略中的隐式拒绝将导致 DENY。
IAM 联合身份用户 (GetFederationToken)	允许 IAM 联合身份用户会话 ARN	隐式拒绝	隐式拒绝	隐式拒绝	允许	权限直接授予会话。其他策略类型不影响决策。

提出请求的主体	基于资源的策略	基于身份的策略	权限边界	会话策略	结果	Reason
根用户	允许根用户 ARN	不适用	不适用	不适用	允许	根账户可以不受限制地访问 Amazon 账户中的所有资源。要了解如何在 Amazon Organizations 中控制对账户的根用户的访问权限，请参阅 Organizations 用户指南中的 服务控制策略 (SCP) 。
Amazon 服务主体	允许 Amazon 服务主体	不适用	不适用	不适用	允许	当基于资源的策略直接向 Amazon 服务主体 (p. 940) ，其他策略类型不会影响决策。

- IAM 角色— 授予 IAM 角色 ARN 权限的基于资源的策略受权限边界或会话策略中隐式拒绝的限制。

角色示例 ARN

```
arn:aws:iam::111122223333:role/examplerole
```

- IAM 角色会话— 在同一账户中，向 IAM 角色会话 ARN 授予权限的基于资源的策略直接向担任的角色会话授予权限。直接授予会话的权限不受基于身份的策略、权限边界或会话策略中的隐式拒绝的限制。当您担任角色并提出请求时，发出请求的主体是 IAM 角色会话 ARN，而不是角色本身的 ARN。

示例角色会话 ARN

```
arn:aws:sts::111122223333:assumed-role/examplerole/examplerolesessionname
```

- IAM 用户 – 在同一账户中，向 IAM 用户 ARN (不是联合身份用户会话) 授予权限的基于资源的策略不受基于身份的策略或权限边界中隐式拒绝的限制。

示例 IAM 用户 ARN

```
arn:aws:iam::111122223333:user/exampleuser
```

- IAM 联合身份用户会话— IAM 联合身份用户会话是通过调用 [GetFederationToken \(p. 307\)](#) 创建的会话。当联合身份用户发出请求时，发出请求的主体是联合身份用户 ARN，而不是联合身份的 IAM 用户的 ARN。在同一个账户中，将权限授予联合身份用户 ARN 的基于资源的策略直接将权限授予会话。直接授予会话的权限不受基于身份的策略、权限边界或会话策略中的隐式拒绝的限制。

但是，如果基于资源的策略向联合身份的 IAM 用户的 ARN 授予权限，则联合身份用户在会话期间发出的请求将受权限边界或会话策略中隐式拒绝的限制。

示例 IAM 联合身份用户会话 ARN

```
arn:aws:sts::111122223333:federated-user/exampleuser
```

4. Identity-based policies (基于身份的策略) - 代码随后会检查主体的基于身份的策略。对于 IAM 用户，这包括用户策略和来自用户所属组的策略。如果没有基于身份的策略或者基于身份的策略中没有允许请求动作的语句，那么请求被隐式否定，代码返回 Deny 的一个最终决定。如果任何适用的基于身份的策略中的任何语句都允许请求的动作，则代码继续适用。
5. IAM 权限边界 - 代码随后会检查主体使用的 IAM 实体是否具有权限边界。如果用于设置权限边界的策略不允许所请求的操作，则请求会被隐式拒绝。代码将返回拒绝最终决定。如果没有权限边界，或者权限边界允许所请求的操作，则代码继续。
6. 会话策略 - 然后代码检查主体是否为会话主体。会话主体包括 IAM 角色会话或者 IAM 联合身份用户会话。如果主体不是会话主体，执行代码将返回 Allow 的最终决定。

对于会话负责人，他的代码检查请求中是否传递了会话策略。您可以传递会话策略，同时使用 Amazon CLI 或 Amazon API 为某个角色或 IAM 联合身份用户获取临时凭证。

- 如果会话策略存在但不允许所请求的操作，则请求会被隐式拒绝。代码将返回拒绝最终决定。
- 如果没有会话策略，则代码将检查主体是否为角色会话。如果主体是角色会话，那么请求是已允许。否则，请求被隐式拒绝，代码返回 Deny 的最终判决。
- 如果会话策略在场并允许请求的动作，那么执行代码返回一个 Allow 的最终决策。

7. Errors (错误) - 如果 Amazon 执行代码在评估过程中的任意点遇到错误，它将生成异常并关闭。

基于身份的策略和基于资源的策略评估示例

策略的最常见类型是基于身份的策略和基于资源的策略。

假定 Carlos 具有用户名 carlossalazar，他尝试将文件保存到 carlossalazar-logs Amazon S3 存储桶。

还假定将以下策略附加到 carlossalazar IAM 用户。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowS3ListRead",
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetAccountPublicAccessBlock",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets"
      ],
      "Resource": "arn:aws:s3::*"
    },
    {
      "Sid": "AllowS3Self",
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::carlossalazar/*",
        "arn:aws:s3:::carlossalazar"
      ]
    }
  ]
}
```

```
    },
    {
      "Sid": "DenyS3Logs",
      "Effect": "Deny",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::*log*"
    }
  ]
}
```

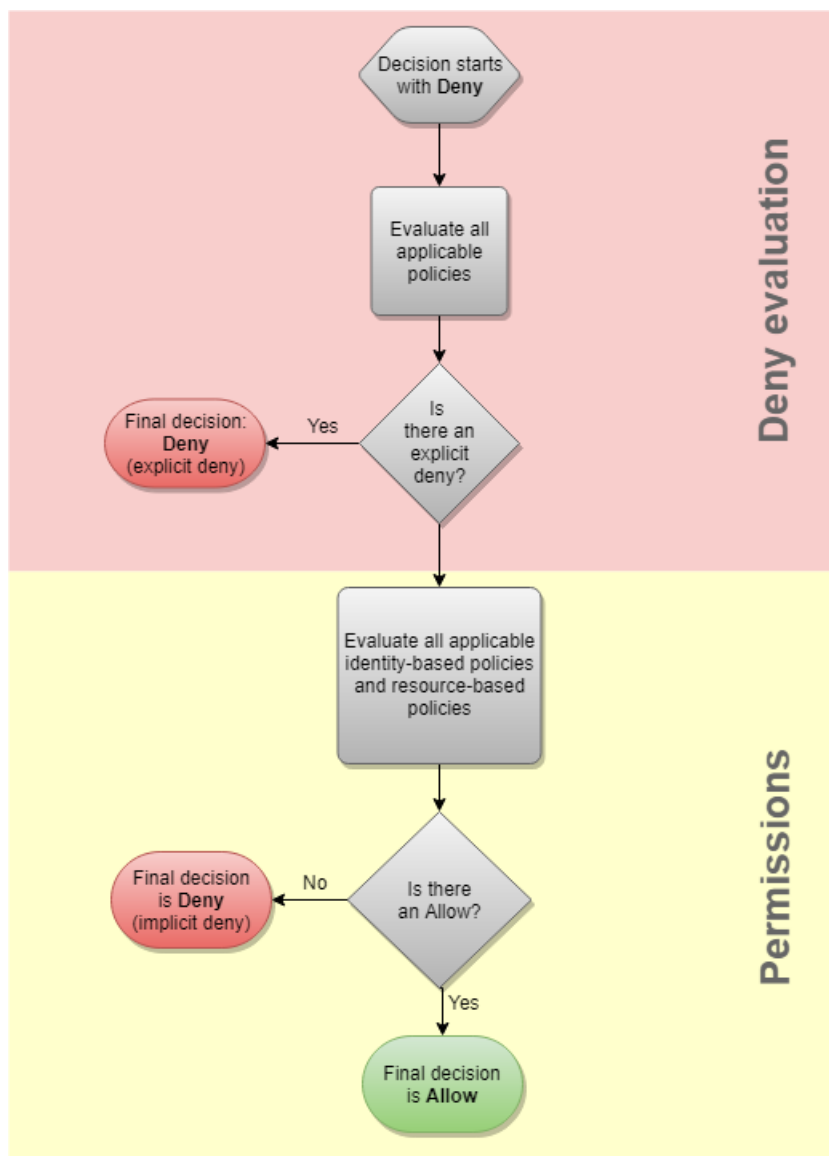
此策略中的 AllowS3ListRead 语句允许 Carlos 查看账户中的所有存储桶的列表。AllowS3Self 语句允许 Carlos 完全访问与其用户名同名的存储桶。DenyS3Logs 语句拒绝 Carlos 访问名称中包含 log 的任何 S3 存储桶。

此外，以下基于资源的策略（称为存储桶策略）附加到 carlossalazar 存储桶。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/carlossalazar"
      },
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::carlossalazar/*",
        "arn:aws:s3:::carlossalazar"
      ]
    }
  ]
}
```

此策略指定只有 carlossalazar 用户可以访问 carlossalazar 存储桶。

当 Carlos 请求将文件保存到 carlossalazar-logs 存储桶时，Amazon 将确定应用于请求的策略。在此示例中，仅基于身份的策略和基于资源的策略适用。这些都是权限策略。由于没有任何权限边界适用，评估逻辑将减少为以下逻辑。



Amazon 首先检查应用于请求上下文的 Deny 语句。它找到一个，因为基于身份的策略显式拒绝 Carlos 访问用于日志记录的任何 S3 存储桶。拒绝 Carlos 访问。

假设他随后意识到自己的错误，尝试将文件保存到 carlossalazar 存储桶。Amazon 检查 Deny 语句，但未找到。然后，它检查权限策略。基于身份的策略和基于资源的策略均允许请求。因此，Amazon 允许请求。如果其中任何一个显式拒绝了语句，则将拒绝请求。如果其中一种策略类型允许请求，而另一种不允许，则仍允许请求。

显式拒绝和隐式拒绝之间的区别

如果适用策略包含 Deny 语句，则请求会导致显式拒绝。如果应用于请求的策略包含一个 Allow 语句和一个 Deny 语句，Deny 语句优先于 Allow 语句。将显式拒绝请求。

当没有适用的 Deny 语句但也没有适用的 Allow 语句时，会发生隐式拒绝。由于原定设置下拒绝访问 IAM 主体，因此必须显式允许他们执行操作。否则，将隐式拒绝他们访问。

在设计授权战略时，您必须创建包含 Allow 语句的策略才能允许主体成功发出请求。但是，您可以选择显式拒绝和隐式拒绝的任意组合。

例如，您可以创建以下策略，其中包括允许的操作、隐式拒绝的操作和显式拒绝的操作。AllowGetList 语句允许对 IAM 操作的只读权限，这些操作以 Get 和 List 为前缀。IAM 中的所有其他操作，例如 iam:CreatePolicy 是隐式拒绝。DenyReports 语句通过拒绝对包含 Report 后缀（例如 iam:GetOrganizationsAccessReport）的操作的访问权限，显式拒绝对 IAM 报告的访问权限。如果有人向此主体添加另一个策略以授予他们访问 IAM 报告的权限（例如 iam:GenerateCredentialReport），由于这种明确拒绝，报告相关请求仍被拒绝。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGetList",
      "Effect": "Allow",
      "Action": [
        "iam:Get*",
        "iam:List*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "DenyReports",
      "Effect": "Deny",
      "Action": "iam:*Report",
      "Resource": "*"
    }
  ]
}
```

跨账户策略评估逻辑

您可以允许一个账户中的主体访问另一个账户中的资源。这称为 cross-account access（跨账户访问）。当您允许跨账户访问时，主体所在的账户称为受信任账户。资源所在的账户是信任账户。

要允许跨账户访问，请将基于资源的策略附加到您要共享的资源。您还必须向在请求中充当主体的身份附加基于身份的策略。信任账户中基于资源的策略必须指定受信任账户中有权访问资源的主体。您可以指定整个账户或其 IAM 用户、联合身份用户、IAM 角色或代入角色会话。您还可以将 Amazon 服务指定为主体。有关更多信息，请参阅[指定主体](#) (p. 936)。

主体的基于身份的策略必须允许对信任服务中的资源进行请求的访问。您可以通过指定资源的 ARN 或允许访问所有资源 (*) 来实现这一点。

在 IAM 中，您可以将基于资源的策略附加到 IAM 角色，以允许其他账户中的主体代入该角色。角色的基于资源的策略称为角色信任策略。代入该角色之后，允许的主体可以使用生成的临时凭证访问您账户中的多个资源。此访问权限在角色的基于身份的权限策略中定义。如需了解使用角色允许跨账户访问与使用其他基于资源的策略允许跨账户访问之间的不同之处，请参阅[IAM 角色与基于资源的策略有何不同](#) (p. 274)。

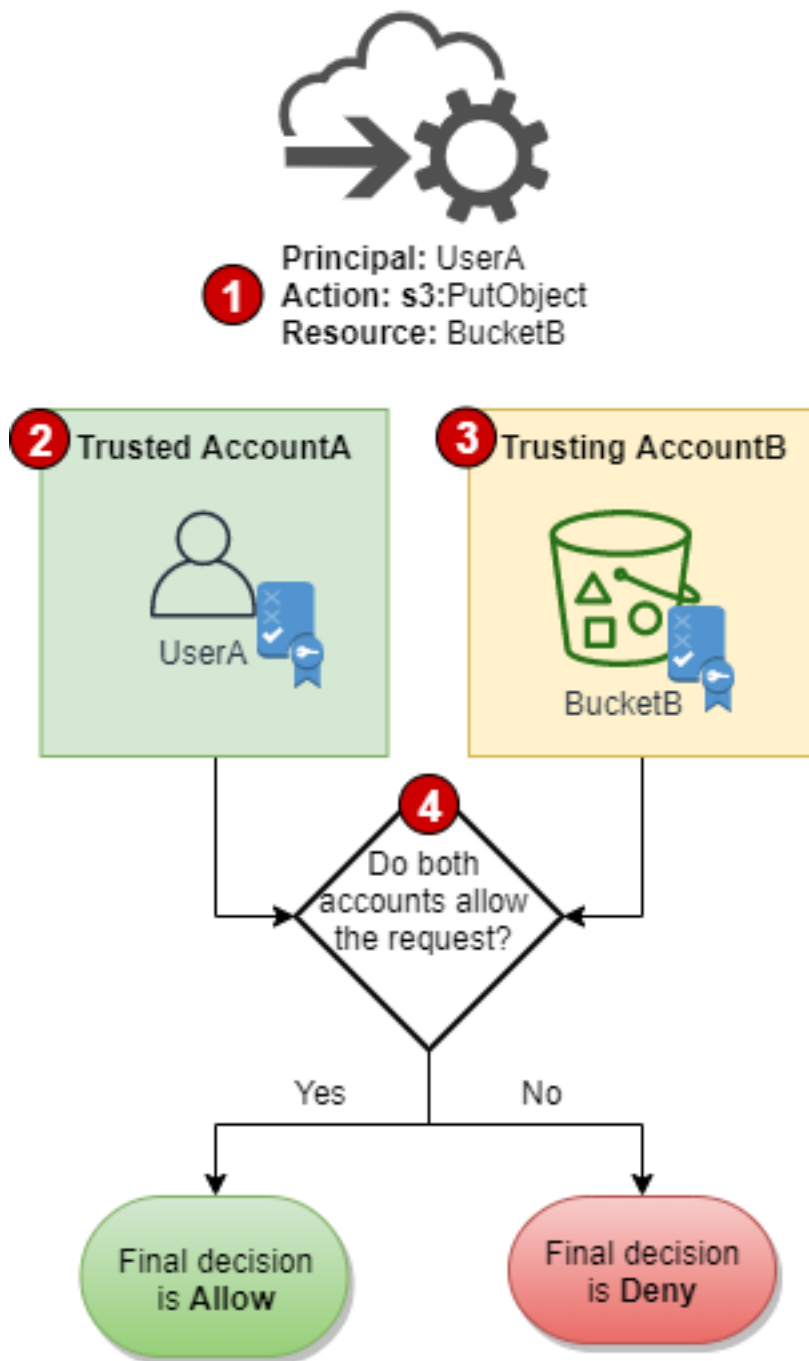
Important

其他服务可能会影响策略评估逻辑。例如，Amazon Organizations 支持可应用于一个或多个账户中的主体的[服务控制策略](#)。Amazon Resource Access Manager 支持[策略片段](#)，用于控制允许主体对其共享的资源执行哪些操作。

确定是否允许跨账户请求

对于跨账户请求，受信任账户 AccountA 中的请求者必须具有基于身份的策略。该策略必须允许他们向信任账户 AccountB 中的资源发出请求。此外，AccountB 中的基于资源的策略必须允许 AccountA 中的请求者访问资源。

当您发出跨账户请求时，Amazon 会执行两个评估。Amazon 评估信任账户和受信任账户中的请求。有关如何在单个账户中评估请求的更多信息，请参阅[确定是允许还是拒绝账户内的请求](#) (p. 974)。仅当两个评估都返回 Allow 决策时，才允许该请求。



1. 当一个账户中的主体发出请求以访问另一个账户中的资源时，这是一个跨账户请求。
2. 请求主体存在于受信任账户 (AccountA) 中。当 Amazon 评估此账户时，它会检查基于身份的策略以及可以限制基于身份的策略的任何策略。有关更多信息，请参阅[评估单个账户中的策略 \(p. 973\)](#)。
3. 请求的资源存在于信任账户 (AccountB) 中。当 Amazon 评估此账户时，它会检查附加到所请求资源的基于资源的策略，以及可以限制基于资源的策略的任何策略。有关更多信息，请参阅[评估单个账户中的策略 \(p. 973\)](#)。
4. 仅当两个账户策略评估均允许该请求时，Amazon 才允许该请求。

跨账户策略评估示例

以下示例演示了一个账户中基于资源的策略向另一个账户中的用户授予权限的情况。

假设 Carlos 是一名开发人员，在账户 111111111111 中具有名为 carlossalazar 的 IAM 用户。他想要将文件保存到账户 222222222222 中的 Production-logs Amazon S3 存储桶。

还假定将以下策略附加到 carlossalazar IAM 用户。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowS3ListRead",
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
    {
      "Sid": "AllowS3ProductionObjectActions",
      "Effect": "Allow",
      "Action": "s3:*Object*",
      "Resource": "arn:aws:s3:::Production/*"
    },
    {
      "Sid": "DenyS3Logs",
      "Effect": "Deny",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::*log*",
        "arn:aws:s3:::*log/*"
      ]
    }
  ]
}
```

此策略中的 AllowS3ListRead 语句允许 Carlos 查看 Amazon S3 中所有存储桶的列表。AllowS3ProductionObjectActions 语句允许 Carlos 对 Production 存储桶中对象的完全访问权限。DenyS3Logs 语句拒绝 Carlos 访问名称中包含 log 的任何 S3 存储桶。它还拒绝对这些存储桶中所有对象的访问。

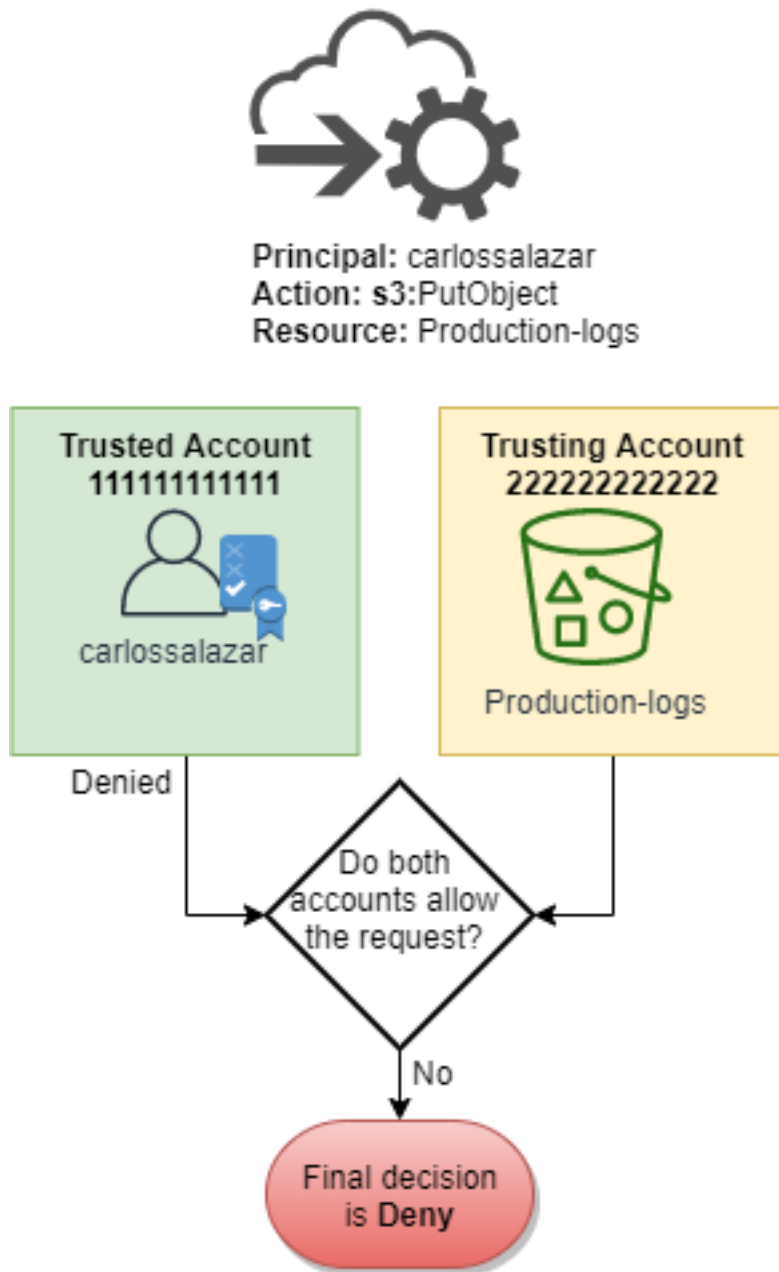
此外，以下基于资源的策略（称为存储桶策略）附加到账户 222222222222 中的 Production 存储桶。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject*",
        "s3:PutObject*",
        "s3:ReplicateObject",
        "s3:RestoreObject"
      ],
      "Principal": { "AWS": "arn:aws:iam::111111111111:user/carlossalazar" },
      "Resource": "arn:aws:s3:::Production/*"
    }
  ]
}
```

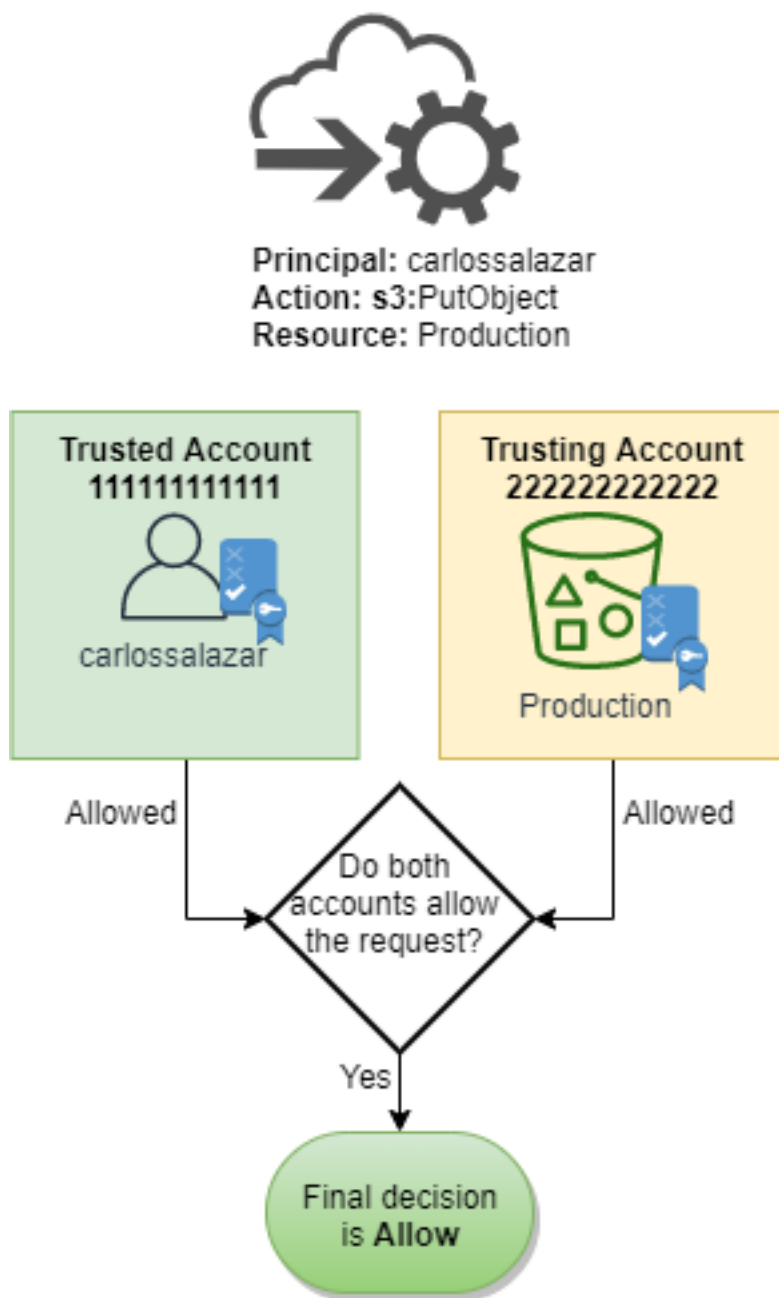
此策略允许 carlossalazar 用户访问 Production 存储桶中的对象。他可以创建和编辑存储桶中的对象，但不能删除。他无法管理存储桶本身。

当 Carlos 请求将文件保存到 Production-logs 存储桶时，Amazon 将确定应用于请求的策略。在这种情况下，附加到 carlossalazar 用户的基于身份的策略是在账户 111111111111 中应用的唯一策略。在账户 222222222222 中，Production-logs 存储桶没有附加基于资源的策略。Amazon 评估账户 111111111111 时返回了决策 Deny。这是因为基于身份的策略中的 DenyS3Logs 语句明确拒绝访问任何日志存储桶。有关如何在单个账户中评估请求的更多信息，请参阅[确定是允许还是拒绝账户内的请求 \(p. 974\)](#)。

由于在一个账户中明确拒绝了请求，因此最终决策是拒绝请求。



假设随后 Carlos 意识到他的错误并尝试将文件保存到 Production 存储桶。Amazon 首先检查账户 111111111111 以确定是否允许请求。仅基于身份的策略适用并允许请求。随后，Amazon 检查账户 222222222222。仅附加到 Production 存储桶的基于资源的策略适用，并允许请求。由于两个账户均允许请求，因此最终决策是允许请求。



IAM JSON 策略语言的语法

该页提供了用于在 IAM 中创建 JSON 策略的语言的正式语法。提供此语法的目的是为了便于您理解如何构造和验证策略。

有关策略示例，请参阅以下主题：

- [IAM 中的策略和权限。](#) (p. 355)
- [IAM 基于身份的策略示例](#) (p. 391)
- [Amazon EC2 User Guide for Linux Instances](#) (适用于 Linux 实例的 Amazon EC2 用户指南) 中[用于 Amazon EC2 控制台的策略示例](#)和使用 [Amazon CLI](#)、[Amazon EC2 CLI](#) 或 [Amazon SDK](#) 的策略示例。

- Amazon Simple Storage Service User Guide (亚马逊云科技 Simple Storage Service 用户指南) 中的 [存储桶策略示例](#)和[用户策略示例](#)。

有关其他 Amazon 产品中使用的策略的示例，请参阅相应服务的文档。

主题

- [策略语言和 JSON \(p. 986\)](#)
- [此语法采用的约定 \(p. 986\)](#)
- [语法 \(p. 987\)](#)
- [策略语法说明 \(p. 988\)](#)

策略语言和 JSON

策略是以 JSON 格式表示的。当您创建或编辑 JSON 策略时，IAM 可以执行策略验证以帮助您创建有效的策略。IAM 可识别 JSON 语法错误，而 IAM Access Analyzer 将提供额外的策略检查和建议，以帮助您进一步优化策略。要了解策略验证的更多信息，请参阅 [验证 IAM 策略 \(p. 443\)](#)。要了解有关 IAM Access Analyzer 策略检查和可执行建议的更多信息，请参阅 [IAM Access Analyzer 策略验证](#)。

在此文档中，我们不提供有效 JSON 组成部分的完整说明。而是提供一些基本的 JSON 规则：

- 各实体间允许使用空格。
- 值用引号括起来。对于数字值和布尔值，引号是可选的。
- 很多元素 (例如 `action_string_list` 和 `resource_string_list`) 都可以采用 JSON 数组作为值。数组可以采用一个或多个值。如果包含多个值，则数组用方括号 ([和]) 括起来并用逗号分隔，如以下示例中所示：

```
"Action" : [ "ec2:Describe*", "ec2:List*" ]
```

- 基本 JSON 数据类型 (布尔型、数字和字符串) 在 [RFC 7159](#) 中定义。

此语法采用的约定

此语法采用以下约定：

- 以下字符是 JSON 令牌，包含 在策略中：

```
{ } [ ] " , :
```

- 以下字符是语法中的特殊字符，不包含 在策略中：

```
= < > ( ) |
```

- 当一个元素允许多个值时，使用重复值、逗号分隔符和省略号 (...) 进行表示。示例：

```
[ <action_string>, <action_string>, ... ]
```

```
<principal_map> = { <principal_map_entry>, <principal_map_entry>, ... }
```

允许多个值时，只包含一个值也是有效的。只有一个值时，必须省略尾部的逗号。如果元素采用数组 (使用 [和] 标记) 但只包含一个值，则括号可选。示例：

```
"Action": [ <action_string> ]
```

```
"Action": <action_string>
```

- 元素后的问号 (?) 表示该元素是可选的。例如：

```
<version_block?>
```

不过，请务必参考语法列表后的说明了解有关可选元素的详细信息。

- 元素之间的竖线 (|) 表示备选项。在语法中，圆括号定义备选项的范围。例如：

```
("Principal" | "NotPrincipal")
```

- 必须为文字字符串的元素括在双引号 (") 中。例如：

```
<version_block> = "Version" : ("2008-10-17" | "2012-10-17")
```

有关更多说明，请参阅语法列表后面的[策略语法说明 \(p. 988\)](#)。

语法

下面的列表描述了策略语言语法。有关此列表中采用的约定，请参阅前面的部分。有关更多信息，请参阅后面的说明。

Note

此语法描述了使用版本号 2008-10-17 和 2012-10-17 标记的策略。Version 策略元素与策略版本不同。Version 策略元素用在策略之中，用于定义策略语言的版本。另一方面，当您 IAM 中的客户托管策略进行更改时，将创建一个策略版本。已更改的策略不会覆盖现有策略。而是由 IAM 创建新的托管策略版本。要了解 Version 策略元素的更多信息，请参阅[IAM JSON 策略元素：Version \(p. 934\)](#)。要了解策略版本的更多信息，请参阅[the section called “IAM 策略版本控制” \(p. 458\)](#)。

```
policy = {
    <version_block?>
    <id_block?>
    <statement_block>
}

<version_block> = "Version" : ("2008-10-17" | "2012-10-17")

<id_block> = "Id" : <policy_id_string>

<statement_block> = "Statement" : [ <statement>, <statement>, ... ]

<statement> = {
    <sid_block?>,
    <principal_block?>,
    <effect_block>,
    <action_block>,
    <resource_block>,
    <condition_block?>
}

<sid_block> = "Sid" : <sid_string>

<effect_block> = "Effect" : ("Allow" | "Deny")

<principal_block> = ("Principal" | "NotPrincipal") : ("*" | <principal_map>)

<principal_map> = { <principal_map_entry>, <principal_map_entry>, ... }

<principal_map_entry> = ("AWS" | "Federated" | "Service" | "CanonicalUser") :
    [<principal_id_string>, <principal_id_string>, ...]

<action_block> = ("Action" | "NotAction") :
    ("*" | [<action_string>, <action_string>, ...])

<resource_block> = ("Resource" | "NotResource") :
    ("*" | [<resource_string>, <resource_string>, ...])
```

```
<condition_block> = "Condition" : { <condition_map> }
<condition_map> = {
  <condition_type_string> : { <condition_key_string> : <condition_value_list> },
  <condition_type_string> : { <condition_key_string> : <condition_value_list> }, ...
}
<condition_value_list> = [<condition_value>, <condition_value>, ...]
<condition_value> = ("string" | "number" | "Boolean")
```

策略语法说明

- 一个策略可以包含一组语句。
- 策略的最大大小介于 2048 个字符和 10,240 个字符之间，具体取决于策略所附加到的实体。有关更多信息，请参阅[IAM 与 Amazon STS 配额、名称要求和字符限制 \(p. 902\)](#)。策略大小计算不包括空格字符。
- 单个元素不能包含同一个键的多个实例。例如，不能在同一语句中包含两次 Effect 块。
- 各个块的显示顺序无限制。例如，在策略中，version_block 可以跟在 id_block 后面。同样地，effect_block、principal_block、action_block 在语句中也可以按任何顺序显示。
- id_block 在基于资源的策略中是可选的。它一定不能包含在基于身份的策略中。
- principal_block 元素在基于资源的策略中（例如，在 Amazon S3 存储桶策略中）和 IAM 角色的信任策略中是必需的。它一定不能包含在基于身份的策略中。
- Amazon S3 存储桶策略中的 principal_map 元素可能包含 CanonicalUser ID。大多数基于资源的策略不支持该映射。要了解有关在存储桶策略中使用规范用户 ID 的更多信息，请参阅 Amazon Simple Storage Service 用户指南中的[在策略中指定主体](#)。
- 每个字符串值
(policy_id_string、sid_string、principal_id_string、action_string、resource_string、condition_string 以及 condition_value 的字符串版本) 都可以有其自己的最小和最大长度限制、具体的允许值或必需的内部格式。

有关字符串值的说明

这一部分提供了有关在策略的不同元素中使用的字符串值的更多信息。

action_string

由服务命名空间、冒号和操作名称组成。操作名称可以包含通配符。示例：

```
"Action": "ec2:StartInstances"

"Action": [
  "ec2:StartInstances",
  "ec2:StopInstances"
]

"Action": "cloudformation:*"

"Action": "*"

"Action": [
  "s3:Get*",
  "s3:List*"
]
```

policy_id_string

提供一种方法将关于策略的信息作为整体包含在其中。某些服务（如 Amazon SQS 和 Amazon SNS）以预留方式使用 Id 元素。除非另受某个服务限制，否则 policy_id_string 可以包含空格。有些服务要求此值在 Amazon 账户内唯一。

Note

`id_block` 可在基于资源的策略中使用，但不能在基于身份的策略中使用。

长度没有限制，但此字符串计入策略总长度，而策略总长度是有限制的。

```
"Id": "Admin_Policy"

"Id": "cd3ad3d9-2776-4ef1-a904-4c229d1642ee"
```

`sid_string`

提供一种方法来包含有关单个语句的信息。对于 IAM policy，`Sid` 值中仅允许使用基本的字母数字 (A-Z、a-z、0-9) 字符。支持资源策略的其他 Amazon 服务对于 `Sid` 值可能有其他要求。例如，一些服务要求此值在 Amazon 账户内唯一，而一些服务允许在 `Sid` 值中使用其他字符，如空格。

```
"Sid": "1"

"Sid": "ThisStatementProvidesPermissionsForConsoleAccess"
```

`principal_id_string`

提供使用 Amazon 账户、IAM 用户、IAM 角色、联合身份用户或担任角色的用户的 [Amazon Resource Name \(ARN\) \(p. 896\)](#) 指定主体的一种方式。对于 Amazon 账户，您还可以使用简单形式 `AWS:accountnumber` 而不是完整的 ARN。有关所有选项（包括 Amazon 服务、担任的角色等），请参阅 [指定主体 \(p. 936\)](#)。

请注意，您只能使用 `*` 指定“everyone/anonymous”。您不能使用它指定一部分名称或 ARN。

`resource_string`

在大多数情况下，由一个 [Amazon Resource Name \(p. 896\)](#) (ARN) 组成。

```
"Resource": "arn:aws:iam::123456789012:user/Bob"

"Resource": "arn:aws:s3:::examplebucket/*"
```

`condition_type_string`

标识所测试条件的类型，例如

`StringEquals`、`StringLike`、`NumericLessThan`、`DateGreaterThanEquals`、`Bool`、`BinaryEquals`、`Ip` 等。有关条件类型的完整列表，请参阅 [IAM JSON 策略元素：条件运算符 \(p. 951\)](#)。

```
"Condition": {
  "NumericLessThanEquals": {
    "s3:max-keys": "10"
  }
}

"Condition": {
  "Bool": {
    "aws:SecureTransport": "true"
  }
}

"Condition": {
  "StringEquals": {
    "s3:x-amz-server-side-encryption": "AES256"
  }
}
```



```
}
```

condition_key_string

标识将对其值进行测试以便确定条件是否满足的条件键。Amazon 定义了一组在所有 Amazon 服务 (包括 `aws:PrincipalType`、`aws:SecureTransport` 和 `aws:userid`) 中可用的条件键。

有关 Amazon 条件键的列表, 请参阅 [Amazon 全局条件上下文密钥 \(p. 998\)](#)。有关服务特定的条件键, 请参阅相应服务的文档, 例如:

- Amazon Simple Storage Service 用户指南中的[在策略中指定条件](#)
- 适用于 Linux 实例的 Amazon EC2 用户指南中的[适用于 Amazon EC2 的 IAM policy](#)

```
"Condition":{
  "Bool": {
    "aws:SecureTransport": "true"
  }
}

"Condition": {
  "StringNotEquals": {
    "s3:x-amz-server-side-encryption": "AES256"
  }
}

"Condition": {
  "StringEquals": {
    "aws:ResourceTag/purpose": "test"
  }
}
```

工作职能的 Amazon 托管策略

我们建议使用[授予最低权限 \(p. 754\)](#)的策略, 或仅授予执行任务所需的许可。授予最小权限的最安全方式是编写一个仅具有团队所需权限的自定义策略。您必须创建一个流程, 以允许您的团队在必要时请求更多权限。创建仅为团队提供所需权限的[IAM 客户托管策略 \(p. 438\)](#)需要时间和专业知识。

要开始向您的 IAM 身份 (用户、用户组和角色) 添加权限, 您可以使用 [Amazon 托管策略 \(p. 364\)](#)。Amazon 托管策略涵盖常见使用案例, 可在您的 Amazon 账户中使用。Amazon 托管策略不会授予最低权限。您必须考虑授予您的主体超出其完成工作所需的更多权限所带来的安全风险。

您可以将 Amazon 托管策略 (包括任务函数) 附加到任何 IAM 身份。要切换到最小权限, 您可以运行 Amazon Identity and Access Management Access Analyzer 以使用 Amazon 托管策略监控主体。了解他们使用的权限后, 您可以编写自定义策略或生成仅包含团队所需权限的策略。这中方法不太安全, 但您能够以更灵活的方式了解您的团队如何使用 Amazon。

Amazon 工作职能托管策略, 旨在贴合 IT 行业的常见工作职能。您可以使用这些策略, 给特定工作职能的某人, 授予执行预期任务所需的权限。这些策略将许多服务的权限整合到单个策略, 比起分散在多个策略之间的权限, 处理起来更加简单。

使用角色来合并服务

某些策略使用 IAM 服务角色帮助您利用在其他 Amazon 服务中发现的功能。这些策略授予对 `iam:passrole` 的访问权限, 允许使用策略的用户将角色传递给 Amazon 服务。此角色向 Amazon 服务委派代表您执行操作的 IAM 权限。

您必须根据需求创建角色。例如, 网络管理员策略允许具有策略的用户向 Amazon CloudWatch 服务传递名为 "flow-logs-vpc" 的角色。CloudWatch 使用该角色记录和捕获用户创建的 VPC 的 IP 流量。

要遵循安全最佳实践，工作职能的策略包括限制可传递的有效角色名称的筛选器。这有助于避免授予不必要的权限。如果您的用户确实需要可选的服务角色，您必须按照策略中指定的命名约定创建一个角色。随后才能给角色授予权限。完成该操作后，用户方可配置服务来使用该角色，为服务授予角色提供的任何权限。

在以下部分中，每个策略的名称都是指向 Amazon Web Services Management Console 中策略详细信息页面的链接。您可以在该页面中查看策略文档并了解它所授予的权限。

管理员任务函数

Amazon 托管策略名称：[AdministratorAccess](#)

使用案例：此用户具有完全访问权限，而且可以为 Amazon 中的每个服务和资源委派权限。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions (策略版本) 选项卡。有关任务函数策略更新的更多信息，请参阅 [工作职能的 Amazon 托管策略 \(p. 995\)](#)。

策略描述：此策略为账户中的所有 Amazon 服务和所有资源授予所有操作权限。

Note

您必须先激活 IAM 用户和角色访问权限，然后 IAM 用户或角色才能使用此策略中的权限访问 Amazon Billing and Cost Management 控制台。为此，请按照[“向账单控制台委派访问权限”教程：第 1 步 \(p. 28\)](#)中的说明进行操作。

账单任务函数

Amazon 托管策略名称：[Billing](#)

使用案例：此用户需要查看账单信息、设置付款和授权付款。用户可以监控整个 Amazon 服务累积的各项成本。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions (策略版本) 选项卡。有关任务函数策略更新的更多信息，请参阅 [工作职能的 Amazon 托管策略 \(p. 995\)](#)。

策略描述：此策略授予管理账单、成本、付款方式、预算和报告的完整权限。

Note

您必须先激活 IAM 用户和角色访问权限，然后 IAM 用户或角色才能使用此策略中的权限访问 Amazon Billing and Cost Management 控制台。为此，请按照[“向账单控制台委派访问权限”教程：第 1 步 \(p. 28\)](#)中的说明进行操作。

数据库管理员任务函数

Amazon 托管策略名称：[DatabaseAdministrator](#)

使用案例：此用户设置、配置和维护 Amazon 云中的数据库。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions (策略版本) 选项卡。有关任务函数策略更新的更多信息，请参阅 [工作职能的 Amazon 托管策略 \(p. 995\)](#)。

策略说明：此策略授予创建、配置和维护数据库的权限。它包括访问 Amazon 数据库服务 (例如 Amazon DynamoDB、Amazon Relational Database Service (RDS) 和 Amazon Redshift)。有关此策略支持的数据库服务的完整列表，请查看此策略。

此作业功能策略支持将角色传递给 Amazon 服务的功能。该策略仅允许对下表中指定的角色执行 `iam:PassRole` 操作。有关更多信息，请参阅本主题后面的 [创建角色并附加策略（控制台）](#) (p. 996)。

适合数据库管理员工作职能的可选 IAM 服务角色

使用案例	角色名称 (* 为通配符)	可选择的服务角色类型	选择此 Amazon 托管策略
允许用户监控 RDS 数据库	rds-monitoring-role	Amazon RDS 角色用于增强监控	AmazonRDSEnhancedMonitoringRole
允许 Amazon Lambda 监控您的数据库和访问外部数据库	rdbms-lambda-access	Amazon EC2	AWSLambda_FullAccess
允许 Lambda 使用 DynamoDB 将文件上传到 Amazon S3 和 Amazon Redshift 集群	lambda_exec_role	Amazon Lambda	按照 Amazon 大数据博客 中的定义，新建一个托管策略
允许 Lambda 函数触发您的 DynamoDB 表	lambda-dynamodb-*	Amazon Lambda	AWSLambdaDynamoDBExecutionRole
允许 Lambda 函数访问 VPC 中的 Amazon RDS	lambda-vpc-execution-role	使用 Amazon Lambda 开发人员指南 中定义的信任策略创建一个角色	AWSLambdaVPCAccessExecutionRole
允许 Amazon Data Pipeline 访问您的 Amazon 资源	DataPipelineDefaultRole	使用 Amazon Data Pipeline 开发人员指南 中定义的信任策略创建一个角色	Amazon Data Pipeline 文档列出了此使用案例所需的权限。请参阅 适用于 Amazon Data Pipeline 的 IAM 角色
允许您在 Amazon EC2 实例上运行的应用程序访问您的 Amazon 资源	DataPipelineDefaultResourceRole	使用 Amazon Data Pipeline 开发人员指南 中定义的信任策略创建一个角色	AmazonEC2RoleforDataPipelineRole

数据科学家任务函数

Amazon 托管策略名称：[DataScientist](#)

使用案例：此用户运行 Hadoop 作业和查询。该用户还访问并分析有关数据分析和商业智能的信息。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions (策略版本) 选项卡。有关任务函数策略更新的更多信息，请参阅 [工作职能的 Amazon 托管策略](#) (p. 995)。

策略描述：此策略授予在 Amazon EMR 集群上创建、管理和运行查询的权限，以及使用 Amazon QuickSight 等工具执行数据分析的权限。该策略包括访问其他数据科学家服务，例如 Amazon Data Pipeline、Amazon EC2、Amazon Kinesis、Amazon Machine Learning 和 SageMaker。有关此策略支持的数据科学家服务的完整列表，请查看此策略。

此作业功能策略支持将角色传递给 Amazon 服务的功能。一个语句允许将任何角色传递给 SageMaker。另一个语句仅允许对下表中指定的角色执行 `iam:PassRole` 操作。有关更多信息，请参阅本主题后面的 [创建角色并附加策略（控制台）](#) (p. 996)。

适合数据科学家工作职能的可选 IAM 服务角色

使用案例	角色名称 (* 为通配符)	可选择的服务角色类型	可选择的 Amazon 托管策略
允许 Amazon EC2 实例访问适合集群的服务和资源	EMR-EC2_DefaultRole	Amazon EMR for EC2	AmazonElasticMapReduceforEC2Role
允许 Amazon EMR 访问适合集群的 Amazon EC2 服务和资源	EMR_DefaultRole	Amazon EMR	AmazonEMRServicePolicy_v2
允许 Kinesis Kinesis Data Analytics 访问串流数据源	kinesis-*	使用 Amazon 大数据博客 中定义的信任策略创建一个角色。	请参阅 Amazon; 大数据博客 ，其中根据您的使用情况概述了四种可能的选项。
允许 Amazon Data Pipeline 访问您的 Amazon 资源	DataPipelineDefaultRole	使用 Amazon Data Pipeline 开发人员指南 中定义的信任策略创建一个角色	Amazon Data Pipeline 文档列出了此使用案例所需的权限。请参阅 适用于 Amazon Data Pipeline 的 IAM 角色
允许您在 Amazon EC2 实例上运行的应用程序访问您的 Amazon 资源	DataPipelineDefaultRole	使用 Amazon Data Pipeline 开发人员指南 中定义的信任策略创建一个角色	AmazonEC2RoleforDataPipelineRole

高级开发人员用户任务函数

Amazon 托管策略名称：[PowerUserAccess](#)

使用案例：此用户执行应用程序开发任务，并可创建和配置特定的资源和服务，它们为能够识别 Amazon 的应用程序开发提供支持。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions (策略版本) 选项卡。有关任务函数策略更新的更多信息，请参阅 [工作职能的 Amazon 托管策略 \(p. 995\)](#)。

策略描述：此策略的第一条语句使用 [NotAction \(p. 945\)](#) 元素以允许对于所有 Amazon 服务和所有资源执行所有操作 (Amazon Identity and Access Management、Amazon Organizations 和 Amazon Account Management 除外)。第二条语句授予创建服务相关角色的 IAM 权限。这是某些服务访问其他服务 (如 Amazon S3 存储桶) 中的资源所必需的。该策略还授予 Organizations 权限以查看有关用户所在企业的信息，包括管理账户电子邮件和企业限制。尽管此策略限制 IAM、Organizations，但它允许用户执行所有 IAM Identity Center 操作 (如果 IAM Identity Center 已启用)。它还授予账户管理权限以查看为账户启用或禁用了哪些 Amazon 区域。

网络管理员任务函数

Amazon 托管策略名称：[NetworkAdministrator](#)

使用案例：此用户分配到的任务是设置和维护 Amazon 网络资源。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions (策略版本) 选项卡。有关任务函数策略更新的更多信息，请参阅 [工作职能的 Amazon 托管策略 \(p. 995\)](#)。

策略说明：此策略授予在 Auto Scaling、Amazon EC2、Amazon Direct Connect、Route 53、Amazon CloudFront、Elastic Load Balancing、Amazon Elastic Beanstalk、Amazon

SNS、CloudWatch、CloudWatch Logs、Amazon S3、IAM 以及 Amazon Virtual Private Cloud 中创建和维护网络资源的权限。

此工作职能需要将角色传递给 Amazon 服务的能力。该策略仅为下表中列出的角色授予 `iam:GetRole` 和 `iam:PassRole` 的权限。有关更多信息，请参阅本主题后面的 [创建角色并附加策略（控制台）](#) (p. 996)。

适合网络管理员工作职能的可选 IAM 服务角色

使用案例	角色名称 (* 为通配符)	可选择的服务角色类型	可选择的 Amazon 托管策略
允许 Amazon VPC 代表用户在 CloudWatch Logs 中创建和管理日志，以监控进出您的 VPC 的 IP 流量	<code>flow-logs-*</code>	使用 Amazon VPC 用户指南 中定义的信任策略创建一个角色	此使用案例没有现成的 Amazon 托管策略，但是，文档列出了所需权限。参阅 Amazon VPC 用户指南 。

只读访问权限

Amazon 托管策略名称：[ReadOnlyAccess](#)

使用案例：此用户需要对 Amazon 账户中所有资源的只读访问权限。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions (策略版本) 选项卡。有关任务函数策略更新的更多信息，请参阅 [工作职能的 Amazon 托管策略](#) (p. 995)。

策略说明：此策略授予列出、获取、描述和查看资源及其属性的权限。它不包括创建或删除等转换功能。此策略包含对与安全相关的 Amazon 服务 (例如 Amazon Identity and Access Management 和 Amazon Billing and Cost Management) 的只读访问权限。有关此策略支持的服务和操作的完整列表，请查看此策略。

安全审计员任务函数

Amazon 托管策略名称：[SecurityAudit](#)

使用案例：此用户监控账户是否符合安全要求。此用户可以访问日志和事件来调查潜在的安全漏洞或潜在的恶意活动。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions (策略版本) 选项卡。有关任务函数策略更新的更多信息，请参阅 [工作职能的 Amazon 托管策略](#) (p. 995)。

策略描述：此策略可授予查看多项 Amazon 服务的配置数据并检查其日志的权限。

支持用户任务函数

Amazon 托管策略名称：[SupportUser](#)

使用案例：此用户与 Amazon Support 联系，创建支持案例并查看现有案例的状态。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions (策略版本) 选项卡。有关任务函数策略更新的更多信息，请参阅 [工作职能的 Amazon 托管策略](#) (p. 995)。

策略描述：此策略授予创建和更新 Amazon Support 案例的权限。

系统管理员任务函数

Amazon 托管策略名称：[SystemAdministrator](#)

使用案例：此用户设置和维护用于开发操作的资源。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions (策略版本) 选项卡。有关任务函数策略更新的更多信息，请参阅 [工作职能的 Amazon 托管策略 \(p. 995\)](#)。

策略说明：此策略授予权限，以跨诸多 Amazon 服务 (包括 Amazon CloudTrail、Amazon CloudWatch、Amazon CodeCommit、Amazon CodeDeploy、Amazon Config、Amazon Directory Service、Amazon EC2、Amazon Identity and Access Management、Amazon Key Management Service、Amazon Lambda、Amazon RDS、Route 53、Amazon S3、Amazon SES、Amazon SQS、Amazon Trusted Advisor 和 Amazon VPC) 创建和维护资源。

此工作职能需要将角色传递给 Amazon 服务的能力。该策略仅为下表中列出的角色授予 iam:GetRole 和 iam:PassRole 的权限。有关更多信息，请参阅本主题后面的 [创建角色并附加策略 \(控制台 \) \(p. 996\)](#)。

适合系统管理员工作职能的可选 IAM 服务角色

使用案例	角色名称 (* 为通配符)	可选择的服务角色类型	可选择的 Amazon 托管策略
允许 Amazon ECS 集群中在 EC2 实例中运行的应用程序访问 Amazon ECS	ecr-sysadmin-*	适合 EC2 Container Service 的 Amazon EC2 角色	AmazonEC2ContainerServiceforEC2
允许用户监控数据库	rds-monitoring-role	Amazon RDS 角色用于增强监控	AmazonRDSEnhancedMonitoringRo
允许在 EC2 实例中运行的应用程序访问 Amazon 资源。	ec2-sysadmin-*	Amazon EC2	角色的示例策略，授予对 S3 存储桶的访问权限，如 适用于 Linux 实例的 Amazon EC2 用户指南 中所示；可根据需要自定义
允许 Lambda 读取 DynamoDB Streams 并写入 CloudWatch Logs	lambda-sysadmin-*	Amazon Lambda	AWSLambdaDynamoDBExecutionR

“仅查看”用户任务函数

Amazon 托管策略名称：[ViewOnlyAccess](#)

使用案例：此用户可以查看账户中所有服务的 Amazon 资源和基本元数据的列表。用户不得读取超出资源配额和列表信息的资源内容和元数据。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions (策略版本) 选项卡。有关任务函数策略更新的更多信息，请参阅 [工作职能的 Amazon 托管策略 \(p. 995\)](#)。

策略描述：此策略授予针对大多数 Amazon 服务的资源的 List*、Describe*、Get*、View* 和 Lookup* 访问权限。要查看此策略包含针对每种服务的哪些操作，请参阅 [ViewOnlyAccess](#)。

工作职能的 Amazon 托管策略

这些策略均由 Amazon 维护并随时保持更新，以便在 Amazon 添加新服务和新功能时，包括对它们的支持。这些策略不能由客户修改。您可以复制策略并对副本进行修改，但在 Amazon 推出新的服务和 API 操作时，不会自动更新该副本。

对于任务函数策略，您可以在 IAM 控制台中查看版本历史记录以及每次更新的时间和日期。为此，请使用此页上的链接查看策略详细信息。然后选择 Policy versions (策略版本) 选项卡以查看版本。此页显示策略的最新 25 个版本。要查看策略的所有版本，请调用 [get-policy-version](#) Amazon CLI 命令或 [GetPolicyVersion](#) API 操作。

Note

您最多可以有五个版本的客户托管策略，但 Amazon 会保留 Amazon 托管策略的完整版本历史记录。

创建角色并附加策略 (控制台)

上文列出的一些策略授予了通过角色配置 Amazon 服务的能力，以便这些服务能够代表您执行操作。工作职能策略或者指定您必须使用的确切角色名称，或者至少包括指定可用名称的开头部分的前缀。要创建这些角色之一，请执行以下程序中的步骤。

创建用于 Amazon Web Service 的角色 (IAM 控制台)

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：<https://console.aws.amazon.com/iam/>。
2. 在 IAM 控制台的导航窗格中，选择 Roles，然后选择 Create role。
3. 选择 Amazon Web Service 角色名称。
4. 选择用于您的服务的使用案例。使用案例由服务定义以包含服务要求的信任策略。
5. 选择 Next (下一步)。
6. 如果可能，选择要用于权限策略的策略。否则，请选择 Create policy (创建策略) 以打开新的浏览器选项卡并从头开始创建新策略。有关更多信息，请参阅 IAM 用户指南 中的 [创建 IAM policy](#)。
7. 在您创建策略后，关闭该选项卡并返回到您的原始选项卡。选中您希望服务具有的权限策略旁边的复选框。

根据您选择的使用案例，服务可能让您执行以下任意操作：

- 不执行任何操作，因为该服务为角色定义了权限。
 - 从一组有限的权限中进行选择。
 - 从任何权限中进行选择。
 - 此时不选择任何策略。但是，您可以稍后创建策略，然后将这些策略附加到角色。
8. (可选) 设置 [权限边界](#)。这是一项高级功能，可用于服务角色，但不可用于服务相关角色。

展开 Permissions boundary (权限边界) 部分，然后选择 Use a permissions boundary to control the maximum role permissions (使用权限边界控制最大角色权限)。IAM 包括您的账户中的 Amazon 托管策略和客户托管策略的列表。选择要用于权限边界的策略，或选择创建策略以打开新的浏览器选项卡并从头开始创建新策略。有关更多信息，请参阅 IAM 用户指南 中的 [创建 IAM policy](#)。在您创建策略后，关闭该选项卡并返回到您的原始选项卡，以选择要用于权限边界的策略。

9. 选择 Next (下一步)。
10. 对于 Role name (角色名称)，角色名称自定义的程度由服务定义。如果服务定义角色的名称，则此选项不可编辑。在其他情况下，服务可能定义角色的前缀并让您输入可选的后缀。某些服务让您指定角色的整个名称。

如果可能，输入有助于识别该角色的作用的角色名称或角色名称后缀。角色名称在您的 Amazon Web Services 账户内必须是唯一的。由于角色名不区分大小写，因此您无法创建名为 **PRODROLE** 和 **prodrole** 两者的角色。各种实体可能会引用该角色。因此，角色创建完毕后无法编辑角色名称。

11. (可选) 对于 Description (描述)，输入新角色的描述。
12. 在 Step 1: Select trusted entities (步骤 1：选择可信实体) 或 Step 2: Select permissions (步骤 2：选择权限) 部分中的 Edit (编辑)，以编辑角色的用户案例和权限。

13. (可选) 通过以键值对的形式附加标签来向用户添加元数据。有关在 IAM 中使用标签的更多信息, 请参阅《IAM 用户指南》中的[标记 IAM 资源](#)。
14. 检查该角色, 然后选择创建角色。

示例 1：将用户配置为数据库管理员（控制台）

此示例显示将 IAM 用户 Alice 配置为 [Database Administrator \(p. 991\)](#) (数据库管理员) 需要执行的步骤。您需要使用此部分中的表中第一行中的信息, 并允许该用户启用 Amazon RDS 监控。您将 [DatabaseAdministrator](#) 策略附加给 Alice 的 IAM 用户, 以便他们可以管理 Amazon 数据库服务。该策略还允许 Alice 将名为 `rds-monitoring-role` 的角色传递给 Amazon RDS 服务, 从而允许该服务代表他们监控 Amazon RDS 数据库。

1. 登录 Amazon Web Services Management Console, 然后通过以下网址打开 IAM 控制台: <https://console.aws.amazon.com/iam/>。
2. 选择 Policies (策略), 然后在搜索框中输入 **database**。
3. 选择 DatabaseAdministrator 策略对应的复选框, 再选择 Policy actions (策略操作), 然后选择 Attach (附上)。
4. 在用户列表中选择 Alice, 然后选择 Attach policy。Alice 现在可以管理 Amazon 数据库了。但是, 要允许 Alice 监控这些数据库, 您必须配置服务角色。
5. 在 IAM 控制台的导航窗格中, 选择 Roles, 然后选择 Create role。
6. 请选择 Amazon Service (亚马逊云科技服务) 角色类型, 然后选择 Amazon RDS。
7. 请选择用于增强监控的 Amazon RDS 角色使用案例。
8. Amazon RDS 将为您角色定义权限。选择下一步: 审核以继续。
9. 角色名称必须是 Alice 当前拥有的 DatabaseAdministrator 策略中指定的一个。其中一个是 **rds-monitoring-role**。针对 Role name (角色名称) 输入该信息。
10. (可选) 对于 Role description (角色描述), 输入新角色的描述。
11. 在检查详细信息后, 选择 Create role。
12. 现在, Alice 即可在 Amazon RDS 控制台的 Monitoring (监控) 部分中启用 RDS Enhanced Monitoring (RDS 增强监控)。例如, 他们可以在创建数据库实例、创建只读副本或修改数据库实例时执行此操作。当他们将 Enable Enhanced Monitoring (启用增强监控) 设置为 Yes (是) 时, 他们必须输入他们在 Monitoring Role (监控角色) 框中创建的角色名称 (`rds-monitoring-role`)。

示例 2：将用户配置为网络管理员（控制台）

此示例显示将 IAM 用户 Jorge 配置为 [Network Administrator \(p. 993\)](#) (网络管理员) 需要执行的步骤。它使用该部分的表中的信息来允许 Jorge 监控往来于 VPC 的 IP 流量。它还允许 Jorge 将该信息记录在 CloudWatch Logs 日志中。您将 [NetworkAdministrator](#) 策略附加给 Jorge 的 IAM 用户, 以便他们可以配置 Amazon 网络资源。该策略还允许 Juan 在您创建流日志时, 将名称以 `flow-logs*` 开头的角色传递给 Amazon EC2。此场景与示例 1 不同, 这里没有预定义的服务角色类型, 因此您必须执行几个不同的步骤。

1. 登录 Amazon Web Services Management Console, 然后通过以下网址打开 IAM 控制台: <https://console.aws.amazon.com/iam/>。
2. 在导航窗格中, 选择 Policies (策略), 然后在搜索框中输入 **network**。
3. 选中 NetworkAdministrator 策略旁的复选框, 然后依次选择 Policy actions (策略操作) 和 Attach (附上)。
4. 在用户列表中选择 Jorge 对应的复选框, 然后选择 Attach policy (附加策略)。Jorge 现在可以管理 Amazon 网络资源了。但是, 为了监控 VPC 中的 IP 流量, 您必须配置服务角色。
5. 由于您需要创建的服务角色没有预定义的托管策略, 您必须先创建一个。在导航窗格中, 选择策略, 然后选择创建策略。
6. 选择 JSON 选项卡, 然后复制以下 JSON 策略文档中的文本。将该文本粘贴到 JSON 文本框中。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

7. 解决[策略验证 \(p. 443\)](#)过程中生成的任何安全警告、错误或常规警告，然后选择 Review policy (查看策略)。

Note

您可以随时在可视化编辑器和 JSON 选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy (查看策略)，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅[调整策略结构 \(p. 871\)](#)。

8. 在 Review (查看) 页面上，输入 **vpc-flow-logs-policy-for-service-role** 作为策略名称。查看策略摘要以查看您的策略授予的权限，然后选择创建策略以保存您的工作。

将在托管策略列表中显示新策略，并已准备好附加该策略。

9. 在 IAM 控制台的导航窗格中，选择 Roles，然后选择 Create role。
10. 请选择 Amazon Service (亚马逊云科技服务) 角色类型，然后选择 Amazon EC2。
11. 请选择 Amazon EC2 使用案例。
12. 在附加权限策略页面上，选择您之前创建的策略：vpc-flow-logs-policy-for-service-role，然后选择下一步：审核。
13. 必须使用 Jorge 当前拥有的 NetworkAdministrator 策略所允许的角色名称。允许以 flow-logs- 开头的任何名称。在此示例中，对于 Role name (角色名称)，请输入 **flow-logs-for-jorge**。
14. (可选) 对于 Role description (角色描述)，输入新角色的描述。
15. 在检查详细信息后，选择 Create role。
16. 现在可以配置本场景需要的信任策略。在 Roles (角色) 页面上，选择 flow-logs-for-jorge 角色 (选择角色名称而不是复选框)。在新角色的详细信息页面上，选择 Trust relationships (信任关系) 选项卡，然后选择 Edit trust relationship (编辑信任关系)。
17. 将条目替换为 ec2.amazonaws.com，“Service”行改为读取，如下所示：

```
"Service": "vpc-flow-logs.amazonaws.com"
```

18. Jorge 现在可以在 Amazon EC2 控制台中为 VPC 或子网创建流日志了。当创建流日志时，请指定 flow-logs-for-jorge 角色。该角色拥有创建日志并向其写入数据的权限。

Amazon 全局条件上下文密钥

当主体 (p. 5) 向 Amazon 发出请求 (p. 5) 时，Amazon 会将请求信息收集到请求上下文 (p. 5) 中。您可以使用 JSON 策略的 Condition 元素将请求上下文中的键与您在策略中指定的键值进行比较。要了解有关在请求上下文中包含全局键的情况的更多信息，请参阅每个全局条件键的可用性信息。有关如何在 JSON 策略中使用 Condition 元素的信息，请参阅[IAM JSON 策略元素：Condition \(p. 948\)](#)。

Note

如果您使用只在某些情况下可用的条件键，则可使用 [IfExists \(p. 957\)](#) 版本的条件运算符。如果请求上下文中缺少条件键，则策略将无法通过评估。例如，将以下条件块与 `...IfExists` 运算符结合使用以在请求来自特定 IP 范围或特定 VPC 时进行匹配。如果请求上下文中未包含这两个键或其中之一，则条件仍将返回 `true`。仅当请求上下文中包含指定的键时，才检查值。

```
"Condition": {
  "IpAddressIfExists": { "aws:SourceIp" : [ "xxx" ] },
  "StringEqualsIfExists" : { "aws:SourceVpc" : [ "yyy" ] }
}
```

全局条件键是带有 `aws:` 前缀的条件键。Amazon 服务既可支持全局条件键，也可提供包含服务前缀的服务特定键。例如，IAM 条件键包含 `iam:` 前缀。有关更多信息，请参阅 [Amazon 服务的操作、资源和条件键](#)，然后选择要查看其键的服务。

Important

要将条件与具有多个键值的请求上下文进行比较，必须使用 `ForAllValues` 或者 `ForAnyValue` 集合运算符。仅将集合运算符用于多值条件键。不要将集合运算符用于单值条件键。有关更多信息，请参阅 [创建具有多个键或值的条件 \(p. 958\)](#)。

aws:CalledVia

与 [字符串运算符 \(p. 951\)](#) 结合使用。

使用此键可以将策略中的服务与代表 IAM 主体（用户或角色）发出请求的服务进行比较。当主体向 Amazon 服务发出请求时，该服务可能会使用主体的凭证向其他服务发出后续请求。`aws:CalledVia` 键包含链中代表主体发出请求的每个服务的有序列表。

例如，您可以使用 Amazon CloudFormation 从 Amazon DynamoDB 表中读取和写入。然后，DynamoDB 使用 Amazon Key Management Service (Amazon KMS) 提供的加密操作。

- 可用性 - 当支持 `aws:CalledVia` 的服务使用 IAM 主体的凭证向其他服务发出请求时，此键会出现在请求中。如果服务使用 [服务角色](#) 或 [服务相关角色](#) 代表主体进行调用，则此键不存在。当主体直接进行调用时，此键也不存在。
- 值类型— 多值

要在策略中使用 `aws:CalledVia` 条件键，您必须提供服务主体以允许或拒绝 Amazon 服务请求。Amazon 支持将以下服务用于 `aws:CalledVia`。

CalledVia 服务

Amazon 服务	服务主体
Amazon Athena	athena.amazonaws.com
Amazon CloudFormation	cloudformation.amazonaws.com
Amazon DynamoDB	dynamodb.amazonaws.com
Amazon Key Management Service (Amazon KMS)	kms.amazonaws.com

要在任何服务使用主体的凭证发出请求时允许或拒绝访问，请使用 [aws:ViaAWSService \(p. 1013\)](#) 条件键。该条件键支持 Amazon 服务。

aws:CalledVia 键是[多值键 \(p. 958\)](#)。但是，您不能在条件下强制使用此键进行排序。使用上面的示例，User 1 (用户 1) 向 Amazon CloudFormation 发出请求，然后依次调用 DynamoDB 和调用 Amazon KMS。这是三个单独的请求。对 Amazon KMS 的最终调用是由用户 1 通过 Amazon CloudFormation，然后通过 DynamoDB 进行的。

在这种情况下，请求上下文中的 aws:CalledVia 键包括 cloudformation.amazonaws.com 和 dynamodb.amazonaws.com (按照该顺序)。如果您只关心调用是在请求链中的某个地方通过 DynamoDB 进行的，则可以在策略中使用此条件键。

例如，以下策略允许管理名为 my-example-key 的 Amazon KMS 键，但前提为 DynamoDB 是发出请求的服务之一。[ForAnyValue:StringEquals \(p. 960\)](#) 条件运算符确保 DynamoDB 是发出调用的服务之一。如果主体直接调用 Amazon KMS，则条件将返回 false，且此策略不允许请求。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "KmsActionsIfCalledViaDynamodb",
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey",
        "kms:DescribeKey"
      ],
      "Resource": "arn:aws:kms:region:111122223333:key/my-example-key",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": ["dynamodb.amazonaws.com"]
        }
      }
    }
  ]
}
```

如果要强制由哪个服务在链中进行第一个或最后一个调用，可以使用 [aws:CalledViaFirst \(p. 1001\)](#) 和 [aws:CalledViaLast \(p. 1001\)](#) 键。例如，以下策略允许管理 Amazon KMS 中名为 my-example-key 的键。仅当链中包含多个请求时，才允许执行这些 Amazon KMS 操作。第一个请求必须通过 Amazon CloudFormation 发出，而最后一个请求必须通过 DynamoDB 发出。如果其他服务在链的中间发出请求，则仍然允许该操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "KmsActionsIfCalledViaChain",
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey",
        "kms:DescribeKey"
      ],
      "Resource": "arn:aws:kms:region:111122223333:key/my-example-key",
      "Condition": {
        "StringEquals": {
          "aws:CalledViaFirst": "cloudformation.amazonaws.com",
          "aws:CalledViaLast": "dynamodb.amazonaws.com"
        }
      }
    }
  ]
}
```

```
}  
]  
}
```

当服务使用 IAM 主体的凭证调用另一个服务时，请求中会存在 [aws:CalledViaFirst \(p. 1001\)](#) 和 [aws:CalledViaLast \(p. 1001\)](#) 键。它们表示在请求链中进行调用的第一个和最后一个服务。例如，假定 Amazon CloudFormation 调用另一个名为 x Service 的服务，然后依次调用 DynamoDB，然后调用 Amazon KMS。对 Amazon KMS 的最终调用是由 User 1 依次通过 Amazon CloudFormation、x Service 和 DynamoDB 进行的。首先通过 DynamoDB 进行调用，最后通过 Amazon CloudFormation 进行调用。

aws:CalledViaFirst

与 [字符串运算符 \(p. 951\)](#) 结合使用。

使用此键可以将策略中的服务与代表 IAM 主体（用户或角色）发出请求的第一个服务进行比较。有关更多信息，请参阅[aws:CalledVia \(p. 999\)](#)。

- Availability（可用性）- 当服务使用 IAM 主体的凭证向其他服务发出至少一个其他请求时，此键会出现在请求中。如果服务使用 [服务角色](#) 或 [服务相关角色](#) 代表主体进行调用，则此键不存在。当主体直接进行调用时，此键也不存在。
- 值类型— 单值

aws:CalledViaLast

与 [字符串运算符 \(p. 951\)](#) 结合使用。

使用此键可以将策略中的服务与代表 IAM 主体（用户或角色）发出请求的最后一个服务进行比较。有关更多信息，请参阅[aws:CalledVia \(p. 999\)](#)。

- Availability（可用性）- 当服务使用 IAM 主体的凭证向其他服务发出至少一个其他请求时，此键会出现在请求中。如果服务使用 [服务角色](#) 或 [服务相关角色](#) 代表主体进行调用，则此键不存在。当主体直接进行调用时，此键也不存在。
- 值类型— 单值

aws:CurrentTime

与 [日期运算符 \(p. 953\)](#) 结合使用。

使用此键可将请求的日期和时间与您在策略中指定的日期和时间进行比较。要查看使用此条件键的示例策略，请参阅[Amazon：允许基于日期和时间进行访问 \(p. 394\)](#)。

- Availability（可用性）- 此键始终包含在请求上下文中。
- 值类型— 单值

aws:EpochTime

与 [日期运算符 \(p. 953\)](#) 或 [数字运算符 \(p. 953\)](#) 结合使用。

使用此键可将请求的日期和时间（以纪元或 Unix 时间表示）与您在策略中指定的值进行比较。此键还接受自 1970 年 1 月 1 日以来的秒数。

- Availability（可用性）- 此键始终包含在请求上下文中。
- 值类型— 单值

aws:FederatedProvider

与[字符串运算符](#) (p. 951)结合使用。

使用此密钥将主体的发布身份提供程序 (IdP) 与您在策略中指定的 IdP 进行比较。这意味着 IAM 角色正在使用 AssumeRoleWithWebIdentity 或 AssumeRoleWithSAML Amazon STS 操作代入。使用生成的角色会话的临时证书发出请求时，请求上下文将标识对原始联合身份进行身份验证的 IdP。

- 可用性 — 当主体是角色会话主体并且该会话是使用第三方身份提供程序发出的时候，此密钥就存在。
- 值类型— 单值

例如，如果用户已通过 Amazon Cognito 进行身份验证，则请求上下文将包含值 `cognito-identity.amazonaws.com`。同样，如果已通过 Login with Amazon 对用户进行身份验证，则请求上下文将包含值 `www.amazon.com`。

您可以使用任何可用的单值条件密钥作为[变量](#) (p. 964)。以下示例基于资源的策略使用 `aws:FederatedProvider` 密钥作为资源的 ARN 中的策略变量。此策略允许使用 IdP 进行身份验证的任何主体，使用特定于发布身份提供程序的路径从 Amazon S3 存储桶获取对象。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/${aws:FederatedProvider}/*"
  }
}
```

aws:MultiFactorAuthAge

与[数字运算符](#) (p. 953)结合使用。

使用此键可将自使用 MFA 向请求主体授权以来的秒数与您在策略中指定的数量进行比较。有关 MFA 的更多信息，请参阅[在 Amazon 中使用多重身份验证 \(MFA\)](#) (p. 104)。

- Availability (可用性) - 仅在使用 MFA 对进行调用的主体进行身份验证后，才将此键包含在请求上下文中。如果未使用 MFA，则此键不存在。
- 值类型— 单值

aws:MultiFactorAuthPresent

与[布尔值运算符](#) (p. 954)结合使用。

使用此键可检查是否已使用多重验证 (MFA) 来验证发出请求的临时安全凭证。

- Availability (可用性) - 仅在主体使用临时凭证发出请求时，才将此键包含在请求上下文中。此键在 Amazon CLI、Amazon API 或使用长期凭证发出的 Amazon 开发工具包请求中不存在。
- 值类型— 单值

临时凭证用于验证 IAM 角色、联合身份用户、具有来自 `sts:GetSessionToken` 的临时令牌的 IAM 用户以及 Amazon Web Services Management Console 用户。IAM 用户访问密钥是长期凭证，但在某些情况下，Amazon 会代表 IAM 用户创建临时凭证以执行操作。在这些情况下，`aws:MultiFactorAuthPresent` 密钥存在于请求中，并设置为值 `false`。有两种常见情况可能会发生这种情形：

- Amazon Web Services Management Console 中的 IAM 用户在不知情的情况下使用了临时凭证。用户使用其作为长期凭证的用户名和密码登录控制台。但在后台，控制台代表用户生成临时凭证。
- 如果 IAM 用户对某个 Amazon 服务进行调用，则该服务将重新使用该用户的凭证向其他服务发出另一个请求。例如，当调用 Athena 以访问 Amazon S3 存储桶或使用 Amazon CloudFormation 创建 Amazon EC2 实例时。对于后续请求，Amazon 使用临时凭证。

要了解哪些服务支持使用临时凭证，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)。

在使用长期凭证（例如，用户访问密钥对）调用 API 或 CLI 命令时，不提供 `aws:MultiFactorAuthPresent` 键。因此我们建议，当您检查此键时使用 [...IfExists \(p. 957\)](#) 版本的条件运算符。

必须理解，以下 Condition 元素不是检查请求是否使用 MFA 进行身份验证的可靠方式。

```
##### WARNING: NOT RECOMMENDED #####
"Effect" : "Deny",
"Condition" : { "Bool" : { "aws:MultiFactorAuthPresent" : "false" } }
```

Deny 效果、Bool 元素和 false 值的这一组合可拒绝可以使用但未使用 MFA 进行身份验证的请求。这仅适用于支持使用 MFA 的临时凭证。此语句不会拒绝对使用长期凭证发出的请求或使用 MFA 进行身份验证的请求的访问权限。使用此示例时务必要谨慎，因为其逻辑复杂，并且未对是否实际使用 MFA 身份验证进行测试。

此外，请勿使用 Deny 效果、Null 元素和 true 的组合，因为该组合行为方式相同，但逻辑更加复杂。

建议的组合

我们建议您使用 [BoolIfExists \(p. 957\)](#) 运算符检查是否使用 MFA 对请求进行身份验证。

```
"Effect" : "Deny",
"Condition" : { "BoolIfExists" : { "aws:MultiFactorAuthPresent" : "false" } }
```

Deny、BoolIfExists 和 false 的组合将拒绝未使用 MFA 进行身份验证的请求。具体来说，它拒绝来自不包含 MFA 的临时凭证的请求。它还拒绝使用长期凭证发出的请求，例如使用访问密钥执行的 Amazon CLI 或 Amazon API 操作。*IfExists 运算符用于检查 `aws:MultiFactorAuthPresent` 键是否存在以及它是否可以存在（由其存在性指示）。当您要拒绝未使用 MFA 进行身份验证的任何请求时，可使用此运算符。这是更安全的，但可能会中断使用访问密钥访问 Amazon CLI 或 Amazon API 的任何代码或脚本。

替代组合

您还可以使用 [BoolIfExists \(p. 957\)](#) 运算符以允许通过 MFA 进行身份验证的请求以及使用长期凭证发出的 Amazon CLI 或 Amazon API 请求。

```
"Effect" : "Allow",
"Condition" : { "BoolIfExists" : { "aws:MultiFactorAuthPresent" : "true" } }
```

此条件将匹配键存在或键不存在的情况。Allow、BoolIfExists 和 true 的组合允许已使用 MFA 进行身份验证的请求或无法使用 MFA 进行身份验证的请求。这意味着，在请求者使用其长期访问密钥时，允许执行 Amazon CLI、Amazon API 和 Amazon 开发工具包操作。该组合不允许来自临时凭证的请求，这些凭证可能包含 MFA，但未包含 MFA。

在使用 IAM 控制台可视化编辑器创建策略并选择需要 MFA 时，将应用该组合。该设置要求使用 MFA 进行控制台访问，但允许在没有 MFA 的情况下进行编程访问。

或者，您也可以使用 Bool 运算符，以便仅在使用 MFA 进行身份验证时允许编程和控制台请求。

```
"Effect" : "Allow",
"Condition" : { "Bool" : { "aws:MultiFactorAuthPresent" : "true" } }
```


Allow、Bool 和 true 的组合仅允许已进行 MFA 身份验证的请求。这仅适用于支持使用 MFA 的临时凭证。该语句不允许访问使用长期访问密钥发出的请求，也不允许访问在没有 MFA 的情况下使用临时凭证发出的请求。

请勿使用策略进行类似以下的构建，以检查 MFA 密钥是否存在：

```
##### WARNING: USE WITH CAUTION #####

"Effect" : "Allow",
"Condition" : { "Null" : { "aws:MultiFactorAuthPresent" : "false" } }
```

Allow 效果、Null 元素和 false 值的组合仅允许可以使用 MFA 进行身份验证的请求，而不考虑该请求是否已实际进行身份验证。这组合允许使用临时凭证发出的所有请求，而对长期凭证拒绝访问权限。使用此示例时务必要谨慎，因为它未对是否实际使用 MFA 身份验证进行测试。

aws:PrincipalAccount

与字符串运算符 (p. 951) 结合使用。

使用此键可将请求主体所属的账户与您在策略中指定的账户标识符进行比较。对于匿名请求，请求上下文将返回 anonymous。

- Availability (可用性) – 此键包含在所有请求 (包括匿名请求) 的请求上下文中。
- 值类型 — 单值

在以下示例中，除了拥有账号 123456789012 的主体外，访问会被拒绝。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessFromPrincipalNotInSpecificAccount",
      "Action": "service:*",
      "Effect": "Deny",
      "Resource": [
        "arn:partition:service:region:accountID:resource"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalAccount": [
            "123456789012"
          ]
        }
      }
    }
  ]
}
```

aws:PrincipalArn

使用 ARN 运算符 (p. 956) 和字符串运算符 (p. 951)。

使用此键可将发出请求的主体的 Amazon Resource Name (ARN) (p. 896) 与您在策略中指定的 ARN 进行比较。对于 IAM 角色，请求上下文将返回角色的 ARN，而不是已代入角色的用户的 ARN。

- Availability (可用性) - 此键包含在所有签名请求的请求上下文中。匿名请求不包括此键。您可以在此条件键中指定以下类型的主体：
 - IAM 角色

- IAM 用户
- Amazon STS 联合用户会话
- Amazon Web Services 账户根用户
- 值类型 — 单值

下表显示了您可以在 `aws:PrincipalArn` 条件键中指定的为不同类型的主体返回的请求上下文值：

- IAM 角色 – 请求上下文包含条件键 `aws:PrincipalArn` 的以下值。请勿将所担任角色会话 ARN 指定为此条件键的值。有关所担任角色会话主体的更多信息，请参阅 [角色会话主体 \(p. 938\)](#)。

```
arn:aws:iam::123456789012:role/role-name
```

- IAM 用户 – 请求上下文包含条件键 `aws:PrincipalArn` 的以下值。

```
arn:aws:iam::123456789012:user/user-name
```

- Amazon STS 联合用户会话 – 请求上下文包含条件键 `aws:PrincipalArn` 的以下值。

```
arn:aws:sts::123456789012:federated-user/user-name
```

- Amazon Web Services 账户 根用户 – 请求上下文包含条件键 `aws:PrincipalArn` 的以下值。当您为根用户 ARN 指定为 `aws:PrincipalArn` 条件键，它仅限制 Amazon Web Services 账户 根用户的权限。这不同于在基于资源策略的主体元素中指定根用户 ARN，后者会将权限委托给 Amazon Web Services 账户。有关在基于资源策略的主体元素中指定根用户 ARN 的更多信息，请参阅 [Amazon 账户主体 \(p. 937\)](#)。

```
arn:aws:iam::123456789012:root
```

您可以将根用户 ARN 指定为 Amazon Organizations 服务控制策略 (SCP) 条件键 `aws:PrincipalArn` 的值。SCP 是一种组织策略，用于管理组织中的权限，仅影响组织中的成员账户。SCP 会限制成员账户中的 IAM 用户和角色的权限，包括成员账户的根用户。有关 SCP 对权限影响的更多信息，请参阅《Organizations 用户指南》中的 [SCP 对权限的影响](#)。

aws:PrincipalTag/tag-key

与 [字符串运算符 \(p. 951\)](#) 结合使用。

使用此键可将附加到发出请求的主体的标签与您在策略中指定的标签进行比较。如果已为主体附加多个标签，则请求上下文会为每个附加的标签键包含一个 `aws:PrincipalTag` 键。

- 可用性 - 仅在主体使用具有附加标签的 IAM 用户时，才将此键包含在请求上下文中。对于主体，如果使用附加了标签或会话标签 (p. 293) 的 IAM 角色，则应包括它。匿名请求不包括此键。
- 值类型 — 单值

您可采用键/值对的形式向用户或角色添加自定义属性。有关 IAM 标签的更多信息，请参阅 [标记 IAM 资源 \(p. 276\)](#)。您可以将 `aws:PrincipalTag` 用于针对 Amazon 主体的 [访问控制 \(p. 388\)](#)。

此示例说明如何创建基于角色的策略，以允许具有 `department=hr` 标签的用户管理 IAM 用户、组或角色。要使用此策略，请将示例策略中的 `#####` 替换为您自己的信息。然后，按照 [创建策略 \(p. 437\)](#) 或 [编辑策略 \(p. 461\)](#) 中的说明操作。

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

```
{
  "Effect": "Allow",
  "Action": "iam:*",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:PrincipalTag/department": "hr"
    }
  }
}
```

aws:PrincipalType

与[字符串运算符](#) (p. 951) 结合使用。

使用此键可将发出请求的主体的类型与您在策略中指定的主体类型进行比较。有关更多信息，请参阅[指定主体](#) (p. 936)。有关 principal 密钥值的具体示例，请参阅[主体键值](#) (p. 969)。

- Availability (可用性) – 此键包含在所有请求 (包括匿名请求) 的请求上下文中。
- 值类型— 单值

aws:referrer

与[字符串运算符](#) (p. 951) 结合使用。

使用此键可将客户端浏览器中引用请求的站点与您在策略中指定的引用站点进行比较。aws:referrer 请求上下文值是由调用方在 HTTP 标头中提供的。当您在网页上选择链接时，Referer 标题将包含在 Web 浏览器请求中。Referer 标题包含选定链接的网页的 URL。

- Availability (可用性) - 仅当通过从浏览器中的网页 URL 链接来调用对 Amazon 资源的请求时，才将此键包含在请求上下文中。编程请求不包括此键，因为它不使用浏览器链接即可访问 Amazon 资源。
- 值类型— 单值

例如，您可以直接使用 URL 或使用直接 API 调用访问 Amazon S3 对象。有关更多信息，请参阅[使用 Web 浏览器直接调用 Amazon S3 API 操作](#)。当您从网页中存在的 URL 访问 Amazon S3 对象时，源网页的 URL 将在 aws:referrer 中使用。当您通过在浏览器中键入 URL 来访问 Amazon S3 对象时，aws:referrer 不存在。直接调用 API 时，aws:referrer 也不存在。您可以使用策略中的 aws:referrer 条件键以允许来自特定引用站点发出的请求，例如公司域中网页上的链接。

Warning

应谨慎使用此键。包含公共已知的引用站点标头值是非常危险的。未经授权方可能会使用修改的浏览器或自定义浏览器提供他们选择的任何 aws:referrer 值。因此，aws:referrer 不应用于阻止未经授权方直接发出 Amazon 请求。提供它只是为了允许客户保护其数字内容 (如存储在 Amazon S3 中的内容)，以免在未经授权的第三方站点上引用。

aws:RequestedRegion

与[字符串运算符](#) (p. 951) 结合使用。

使用此键可将已在请求中调用的 Amazon 区域与您在策略中指定的区域进行比较。您可以使用此全局条件键来控制可请求的区域。要查看每种服务的 Amazon 区域，请参阅 Amazon Web Services 一般参考中的[服务终端节点和配额](#)。

- Availability (可用性) - 此键始终包含在请求上下文中。
- 值类型— 单值

一些全球服务（如 IAM）具有单个终端节点。因为此端点的物理位置在美国东部（弗吉尼亚北部）区域，所以 IAM 调用始终会向 us-east-1 区域发出。例如，如果您创建一个策略以在请求的区域不是 us-west-2 时拒绝访问所有服务，则 IAM 调用将始终失败。要查看如何解决此问题的示例，请参阅 [NotAction 与 Deny \(p. 945\)](#)。

Note

使用 `aws:RequestedRegion` 条件键可以控制服务的哪个终端节点被调用，但不控制该操作产生的影响。一些服务具有跨区域影响。例如，Amazon S3 具有控制跨区域复制的 API 操作。您可以在一个区域中调用 `s3:PutBucketReplication`（受 `aws:RequestedRegion` 条件键的影响），但其他区域将会受到影响，具体取决于复制配置设置。

您可以使用该上下文密钥将访问限制为一组给定区域中的 Amazon 服务。例如，以下策略允许用户查看 Amazon Web Services Management Console 中的所有 Amazon EC2 实例。但是，它仅允许用户更改爱尔兰 (eu-west-1)、伦敦 (eu-west-2) 或巴黎 (eu-west-3) 中的实例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "InstanceConsoleReadOnly",
      "Effect": "Allow",
      "Action": [
        "ec2:Describe*",
        "ec2:Export*",
        "ec2:Get*",
        "ec2:Search*"
      ],
      "Resource": "*"
    },
    {
      "Sid": "InstanceWriteRegionRestricted",
      "Effect": "Allow",
      "Action": [
        "ec2:Associate*",
        "ec2:Import*",
        "ec2:Modify*",
        "ec2:Monitor*",
        "ec2:Reset*",
        "ec2:Run*",
        "ec2:Start*",
        "ec2:Stop*",
        "ec2:Terminate*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": [
            "eu-west-1",
            "eu-west-2",
            "eu-west-3"
          ]
        }
      }
    }
  ]
}
```

`aws:RequestTag/tag-key`

与[字符串运算符 \(p. 951\)](#)结合使用。

使用此键可将请求中传递的标签键/值对与您在策略中指定的标签对进行比较。例如，您可以检查请求是否包含标签键 "Dept" 并具有 "Accounting" 值。有关更多信息，请参阅[Amazon 请求期间控制访问 \(p. 390\)](#)。

- Availability (可用性) – 当在请求中传递标签键值对时，将此键包含在请求上下文中。在请求中传递多个标签时，每个标签键/值对均有一个上下文密钥。
- 值类型— 单值

该上下文密钥的格式为 "aws:RequestTag/*tag-key*": "*tag-value*"，其中 *tag-key* 和 *tag-value* 是标签键值对。标签密钥和值不区分大小写。这意味着，如果您在策略的条件元素中指定 "aws:RequestTag/TagKey1": "Value1"，则条件将匹配名为 TagKey1 或 tagkey1 的请求标签键，但不会同时匹配两者。

这个例子表明，虽然该键是单值的，但如果键不同，您仍然可以在请求中使用多个键值对。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2::instance/*",
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/environment": [
          "preprod",
          "production"
        ],
        "aws:RequestTag/team": [
          "engineering"
        ]
      }
    }
  }
}
```

aws:ResourceTag/tag-key

与[字符串运算符 \(p. 951\)](#)结合使用。

使用此键可将您在策略中指定的标签键/值对与附加到资源的键/值对进行比较。例如，您可能会要求只有在资源具有附加的标签键 "Dept" 和值 "Marketing" 时才允许访问该资源。有关更多信息，请参阅[控制对 Amazon 资源的访问 \(p. 390\)](#)。

- Availability (可用性) – 当请求的资源已附加标签，或者是在利用附加包创建资源的请求中，键会包含在请求上下文中。只有[支持基于标签的授权 \(p. 908\)](#)的资源返回此键。每个标签键/值对均有一个上下文密钥。
- 值类型— 单值

该上下文密钥的格式为 "aws:ResourceTag/*tag-key*": "*tag-value*"，其中 *tag-key* 和 *tag-value* 是标签键值对。标签密钥和值不区分大小写。这意味着，如果您在策略的条件元素中指定 "aws:ResourceTag/TagKey1": "Value1"，则条件将匹配名为 TagKey1 或 tagkey1 的资源标签键，但不会同时匹配两者。

有关使用 aws:ResourceTag 键控制对 IAM 资源的访问的示例，请参阅[控制对 Amazon 资源的访问 \(p. 390\)](#)。

有关使用 aws:ResourceTag 键控制对其他 Amazon 资源的访问的示例，请参阅[使用标签控制对 Amazon 资源的访问 \(p. 389\)](#)。

有关使用 `aws:ResourceTag` 条件键进行基于属性的访问控制 (ABAC) 的教程，请参阅 [IAM 教程：根据标签定义访问 Amazon 资源的权限 \(p. 42\)](#)。

aws:SecureTransport

与 [布尔值运算符 \(p. 954\)](#) 结合使用。

使用此键可检查是否已使用 SSL 发送请求。请求上下文将返回 `true` 或 `false`。在策略中，只能在使用 SSL 发送请求时允许特定操作。

- Availability (可用性) - 此键始终包含在请求上下文中。
- 值类型— 单值

aws:SourceAccount

与 [字符串运算符 \(p. 951\)](#) 结合使用。

使用此键可将发出服务到服务请求的资源的账户 ID 与您在策略中指定的账户 ID 进行比较。

- Availability (可用性) - 仅在访问资源触发一项 Amazon 服务以代表资源所有者调用另一项服务时，将此键包含在请求上下文中。发出调用的服务必须将源的资源账户 ID 传递给被调用的服务。此账户 ID 包括源账户 ID。
- 值类型— 单值

您可以使用此条件键来防止 Amazon 服务在服务之间执行事务时被用作[混淆代理人 \(p. 166\)](#)。将此条件键的值设置为请求中资源的账户。例如，当 Amazon S3 存储桶更新触发 Amazon SNS 主题帖子时，Amazon S3 服务将调用 `sns:Publish` API 操作。在允许 `sns:Publish` 操作的策略中，将条件键的值设置为 Amazon S3 存储桶的账户 ID。有关如何以及建议何时使用这些条件键的信息，请参阅有关您正在使用的 Amazon 服务的文档。

aws:SourceArn

与 [ARN 运算符 \(p. 956\)](#) 和 [字符串运算符 \(p. 951\)](#) 结合使用。Amazon 建议在比较 ARN 时使用 ARN 运算符而不是字符串运算符。

使用此键可将发出服务到服务请求的资源的 [Amazon Resource Name \(ARN\) \(p. 896\)](#) 与您在策略中指定的 ARN 进行比较。

此键不适用于发出请求的主体的 ARN。请改用 [aws:PrincipalArn \(p. 1004\)](#)。源的 ARN 包含账户 ID，因此不必将 `aws:SourceAccount` 与 `aws:SourceArn` 结合使用。

- Availability (可用性) - 仅在访问资源触发一项 Amazon 服务以代表资源所有者调用另一项服务时，将此键包含在请求上下文中。调用服务必须将原始资源的 ARN 传递给被调用的服务。
- 值类型— 单值

您可以使用此条件键来防止 Amazon 服务在服务之间执行事务时被用作[混淆代理人 \(p. 166\)](#)。将此条件键的值设置为请求中资源的 ARN。例如，当 Amazon S3 存储桶更新触发 Amazon SNS 主题帖子时，Amazon S3 服务将调用 `sns:Publish` API 操作。在允许 `sns:Publish` 操作的策略中，将条件键的值设置为 Amazon S3 存储桶的 ARN。有关如何以及建议何时使用这些条件键的信息，请参阅有关您正在使用的 Amazon 服务的文档。

aws:SourceIdentity

与 [字符串运算符 \(p. 951\)](#) 结合使用。

使用此密钥可将主体设置的源身份与您在策略中指定的源身份进行比较。

- 可用性 — 在使用任何 Amazon STS 担任角色 CLI 命令或 Amazon STS AssumeRoleAPI 操作担任角色时设置好源身份后，此密钥将会被包含在请求中。
- 值类型— 单值

您可以在策略中使用此密钥来允许 Amazon 中的主体操作，这些主体在担任角色时设置了源身份。角色指定源身份的活动显示在 [Amazon CloudTrail \(p. 343\)](#)。这使管理员能够更轻松确定什么角色在 Amazon 中执行了操作。

与 [sts:RoleSessionName \(p. 1023\)](#) 不同，在设置源身份后，无法更改该值。它存在于角色执行的所有操作的请求上下文中。当您使用会话凭证担任另一个角色时，该值将保留到后续角色会话中。从一个角色代入另一个角色的过程称为 [角色链 \(p. 159\)](#)。

当主体使用任何 Amazon STS 担任角色 CLI 命令或 Amazon STS AssumeRole API 操作担任角色并初次设置源身份时，[sts:SourceIdentity \(p. 1024\)](#) 密钥将会出现在请求中。aws:SourceIdentity 键存在于对具有源身份设置的角色会话执行的任何操作请求中。

以下账户 111122223333 中适用于 CriticalRole 的角色信任策略包含一个条件适用于 aws:SourceIdentity 的条件，可防止未将源身份设置为 Saanvi 或 Diego 主体担任该角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeRoleIfSourceIdentity",
      "Effect": "Allow",
      "Principal": { "AWS": "arn:aws:iam::123456789012:role/CriticalRole" },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringLike": {
          "aws:SourceIdentity": ["Saanvi", "Diego"]
        }
      }
    }
  ]
}
```

要了解有关使用源身份信息的更多信息，请参阅 [监控和控制使用所担任角色执行的操作 \(p. 316\)](#)。

aws:SourceIp

与 [IP 地址运算符 \(p. 955\)](#) 结合使用。

使用此键可将请求者的 IP 地址与您在策略中指定的 IP 地址进行比较。aws:SourceIp 条件键只能用于公有 IP 地址范围。

- Availability (可用性) - 此键将包含在请求上下文中，但请求者使用 VPC 终端节点发出请求时除外。
- 值类型— 单值

可在策略中使用 aws:SourceIp 条件键以仅允许主体从指定的 IP 范围发出请求。但是，如果 Amazon 服务代表主体进行调用，则此策略拒绝访问。在这种情况下，您可以将 aws:SourceIp 与 [aws:ViaAWSService \(p. 1013\)](#) 键结合使用，以确保源 IP 限制仅适用于主体直接发出的请求。

例如，您可以将以下策略附加到一个 IAM 用户。如果用户从指定的 IP 地址进行调用，则此策略允许用户将对象直接放入 **DOC-EXAMPLE-BUCKET3** Amazon S3 存储桶中。但是，如果用户发

出另一个请求而导致服务调用 Amazon S3，则 IP 地址限制不适用。只有当请求不是由服务发出时，PrincipalPutObjectIfIpAddress 语句才会限制 IP 地址。如果请求是由服务发出的，则 ServicePutObject 语句允许执行此操作而没有 IP 地址限制。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PrincipalPutObjectIfIpAddress",
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET3/*",
      "Condition": {
        "Bool": { "aws:ViaAWSService": "false" },
        "IpAddress": { "aws:SourceIp": "203.0.113.0" }
      }
    },
    {
      "Sid": "ServicePutObject",
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
      "Condition": {
        "Bool": { "aws:ViaAWSService": "true" }
      }
    }
  ]
}
```

如果请求来自使用 Amazon VPC 终端节点的主机，则 `aws:SourceIp` 密钥不可用。您应改用特定于 VPC 的键，例如 `aws:VpcSourceIp` (p. 1010)。有关使用 VPC 端点的更多信息，请参阅《Amazon PrivateLink 指南》中的 [VPC 端点和 VPC 端点服务的身份和访问管理](#)。

aws:SourceVpc

与[字符串运算符](#) (p. 951) 结合使用。

使用此键可检查请求是否来自您在策略中指定的 VPC。在策略中，您可以使用此键以仅允许访问特定的 VPC。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的[限制对特定 VPC 的访问](#)。

- Availability (可用性) - 仅在请求者使用 VPC 终端节点发出请求时，才将此键包含在请求上下文中。
- 值类型— 单值

aws:SourceVpce

与[字符串运算符](#) (p. 951) 结合使用。

使用此键可将请求的 VPC 终端节点标识符与您在策略中指定的终端节点 ID 进行比较。在策略中，您可以使用此键来限制对特定 VPC 的访问。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的[限制对特定 VPC 终端节点的访问](#)。

- Availability (可用性) - 仅在请求者使用 VPC 终端节点发出请求时，才将此键包含在请求上下文中。
- 值类型— 单值

aws:TagKeys

与[字符串运算符](#) (p. 951) 结合使用。

使用此键可将请求中的标签键与您在策略中指定的键进行比较。作为使用策略来通过标签控制访问的最佳实践，请使用 `aws:TagKeys` 条件键来定义允许的标签键。有关示例策略和更多信息，请参阅[the section called “根据标签键控制访问” \(p. 391\)](#)。

- Availability (可用性) – 仅在操作支持将标签传递到请求中时，才将此键包含在请求上下文中。
- 值类型— 多值

该上下文密钥的格式为 `"aws:TagKeys": "tag-key"`，其中 *tag-key* 是没有值的标签键列表 (例如，`["Dept", "Cost-Center"]`)。

由于可在一个请求中包含多个标签键/值对，因此，请求内容可以是[多值 \(p. 958\)](#)请求。在此情况下，您必须使用 `ForAllValues` 或 `ForAnyValue` 集合运算符。有关更多信息，请参阅[使用多个键和值 \(p. 960\)](#)。

某些服务支持同时使用标记和资源操作，例如创建、修改或删除资源。要允许在单次调用中同时使用标记和操作，您必须创建同时包含标记操作和资源修改操作的策略。然后，您可以使用 `aws:TagKeys` 条件键强制在请求中使用特定标签键。例如，要在某人创建 Amazon EC2 快照时限制标签，您必须在策略中包含 `ec2:CreateSnapshot` 创建操作和 `ec2:CreateTags` 标记操作。要查看此使用 `aws:TagKeys` 场景的策略，请参阅适用于 Linux 实例的 Amazon EC2 用户指南中的[创建具有标签的快照](#)。

aws:TokenIssueTime

与[日期运算符 \(p. 953\)](#)结合使用。

使用此键可将临时安全凭证的颁发日期和时间与您在策略中指定的日期和时间进行比较。

- Availability (可用性) - 仅在主体使用临时凭证发出请求时，才将此键包含在请求上下文中。使用访问密钥发出的 Amazon CLI、Amazon API 或 Amazon 开发工具包请求中未提供此键。
- 值类型— 单值

要了解哪些服务支持使用临时凭证，请参阅[使用 IAM 的 Amazon 服务 \(p. 908\)](#)。

aws:UserAgent

与[字符串运算符 \(p. 951\)](#)结合使用。

使用此键可将请求者的客户端应用程序与您在策略中指定的应用程序进行比较。

- Availability (可用性) - 此键始终包含在请求上下文中。
- 值类型— 单值

Warning

应谨慎使用此键。由于 `aws:UserAgent` 值由发起人在 HTTP 标头中提供，因此未经授权方可以修改或自定义浏览器以提供他们选择的任何 `aws:UserAgent` 值。因此，`aws:UserAgent` 不应用于阻止未经授权方直接发出 Amazon 请求。您可以使用它来仅允许特定客户端应用程序，并且仅在对策略进行测试之后。

aws:userid

与[字符串运算符 \(p. 951\)](#)结合使用。

使用此键可将请求者的主体标识符与您在策略中指定的 ID 进行比较。对于 IAM 用户，请求上下文值是用户 ID。对于 IAM 角色，此值的格式可能有所不同。有关如何为不同的主体显示信息的详细信息，请参阅[指定主体 \(p. 936\)](#)。有关 principal 密钥值的具体示例，请参阅[主体键值 \(p. 969\)](#)。

- Availability (可用性) – 此键包含在所有请求 (包括匿名请求) 的请求上下文中。
- 值类型— 单值

aws:username

与字符串运算符 (p. 951) 结合使用。

使用此键可将请求者的用户名与您在策略中指定的用户名进行比较。有关如何为不同的主体显示信息的详细信息，请参阅[指定主体 \(p. 936\)](#)。有关 principal 密钥值的具体示例，请参阅[主体键值 \(p. 969\)](#)。

- 可用性 - 此键始终包含在 IAM 用户的请求上下文中。匿名请求或使用 Amazon Web Services 账户 根用户或 IAM 角色 角色发出的请求不包含此键。使用 IAM Identity Center 凭证发出的请求不会在上下文中包含此键。要了解如何控制对 IAM Identity Center 中的用户的访问权限，请参阅[identitystore:UserId 将 IAM Identity Center 身份存储中的预定义属性用于 Amazon 中的访问控制](#)。IAM Identity Center 中的用户是您的员工队伍中需要访问您的 Amazon 账户或您的云应用程序的人员。
- 值类型 — 单值

aws:ViaAWSService

与布尔值运算符 (p. 954) 结合使用。

使用此键检查 Amazon 服务是否代表您向其他服务发出请求。

当服务使用 IAM 主体的凭证代表主体发出请求时，请求上下文密钥返回 true。如果服务使用[服务角色或服务相关角色](#)代表主体进行调用，则上下文密钥返回 false。当主体直接进行调用时，请求上下文密钥也会返回 false。

- Availability (可用性) - 此键始终包含在请求上下文中。
- 值类型— 单值

您可以根据请求是否由服务发出，使用此条件键允许或拒绝访问。有关示例策略，请参阅[Amazon：基于源 IP 拒绝对 Amazon 的访问 \(p. 405\)](#)。

aws:VpcSourceIp

与 IP 地址运算符 (p. 955) 结合使用。

使用此键可将从中发出请求的 IP 地址与您在策略中指定的 IP 地址进行比较。在策略中，键仅在请求来自指定的 IP 地址并经过 VPC 终端节点时进行匹配。

- Availability (可用性) - 仅在使用 VPC 终端节点发出请求时，才将此键包含在请求上下文中。
- 值类型— 单值

有关更多信息，请参阅《Amazon VPC 用户指南》中的[使用 VPC 终端节点控制对服务的访问](#)。

其他跨服务条件键

全局条件键是带 aws: 前缀的条件键。单个服务可以创建自己的条件键。这些特定于服务的条件键包括与服务名称匹配的前缀，例如 iam: 或 sts:。

服务可以创建服务特定键，这些键在其他服务的请求上下文中可用。这些键可在多个服务中使用，但不是全局条件键。例如，Amazon STS 支持[基于 SAML 的联合条件键 \(p. 1019\)](#)。当使用 SAML 联合

的用户在其他服务中执行 Amazon 操作时，这些键可用。其他示例包括 `identitystore:UserId` 和 `ec2:SourceInstanceArn`。

要查看用于一项服务的特定条件键，请参阅[Amazon 服务的操作、资源和条件键](#)，然后选择要查看其键的服务。

IAM 和 Amazon STS 条件上下文密钥

您可以在 JSON 策略中使用 `Condition` 元素来测试所有 Amazon 请求的请求上下文中所包含键的值。这些键提供有关请求本身或请求所引用资源的信息。在允许用户请求的操作之前，您可以检查这些键是否具有指定值。这样，您就可以精细控制 JSON 策略语句何时与传入的请求匹配或不匹配。有关如何在 JSON 策略中使用 `Condition` 元素的信息，请参阅[IAM JSON 策略元素：Condition \(p. 948\)](#)。

本主题介绍了由 IAM 服务定义和提供的键（带 `iam:` 前缀）以及由 Amazon Security Token Service (Amazon STS) 服务定义和提供的键（带 `sts:` 前缀）。其他几个 Amazon 服务也提供与该服务定义的操作和资源相关的服务特定键。有关更多信息，请参阅[Amazon 服务的操作、资源和条件键](#)。支持条件键的服务的文档通常包含附加信息。例如，有关可在 Amazon S3 资源的策略中使用的密钥信息，请参阅 Amazon Simple Storage Service 用户指南中的[Amazon S3 策略密钥](#)。

主题

- [IAM 的可用键 \(p. 1014\)](#)
- [Amazon Web 联合身份验证的可用键 \(p. 1016\)](#)
- [基于 SAML 的 Amazon STS 联合身份验证的可用键 \(p. 1019\)](#)
- [Amazon STS 的可用键 \(p. 1022\)](#)

IAM 的可用键

可以在控制对 IAM 资源的访问的策略中使用以下条件键：

`iam:AssociatedResourceArn`

与 [ARN 运算符 \(p. 956\)](#) 结合使用。

指定此角色在目标服务上关联的资源的 ARN。资源通常属于主体将角色传递到的服务。有时，资源可能属于第三个服务。例如，您可以将角色传递给他们，在 Amazon EC2 实例上使用的 Amazon EC2 Auto Scaling。在这种情况下，条件将匹配 Amazon EC2 实例的 ARN。

此条件键仅适用于策略中的 [PassRole \(p. 240\)](#) 操作。无法使用它来限制任何其他操作。

在策略中使用此条件键可允许实体传递角色，但前提是角色与指定资源相关联。您可以使用通配符 (*) 允许对特定类型的资源执行操作，而无需限制区域或资源 ID。例如，您可以允许 IAM 用户或角色将任何角色传递给 Amazon EC2 服务，以便用于 `us-east-1` 或 `us-west-1` 区域中的实例。不允许将 IAM 用户或角色传递给其他服务。此外，它不允许 Amazon EC2 将角色用于其他区域中的实例。

```
{
  "Effect": "Allow",
  "Action": "iam:PassRole",
  "Resource": "*",
  "Condition": {
    "StringEquals": {"iam:PassedToService": "ec2.amazonaws.com"},
    "StringLike": {
      "iam:AssociatedResourceARN": [
        "arn:aws:ec2:us-east-1:111122223333:instance/*",
        "arn:aws:ec2:us-west-1:111122223333:instance/*"
      ]
    }
  }
}
```

```
}
```

Note

支持 [iam:PassedToService](#) (p. 1015) 的 Amazon 服务也支持此条件键。

iam:AWSServiceName

与字符串运算符 (p. 951) 结合使用。

指定此角色将附加到的 Amazon 服务。

在此示例中，如果服务名称为 `access-analyzer.amazonaws.com`，您将允许实体创建与服务关联的角色。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "iam:AWSServiceName": "access-analyzer.amazonaws.com"
      }
    }
  }]
}
```

iam:OrganizationsPolicyId

与字符串运算符 (p. 951) 结合使用。

检查具有指定 Amazon Organizations ID 的策略是否与请求中使用的策略匹配。要查看使用此条件键的示例 IAM policy，请参阅 [IAM：查看 Organizations 策略的上次访问的服务信息](#) (p. 427)

iam:PassedToService

与字符串运算符 (p. 951) 结合使用。

指定可将角色传递到的服务的主体。此条件键仅适用于策略中的 [PassRole](#) (p. 240) 操作。无法使用它来限制任何其他操作。

在策略中使用此条件键时，请使用服务主体指定服务。服务主体是可在策略的 `Principal` 元素中指定的服务的名称。常见格式为：`SERVICE_NAME_URL.amazonaws.com`。

您可以使用 `iam:PassedToService` 限制您的用户，使其只能将角色传递到特定服务。例如，用户可能会创建一个 [服务角色](#) (p. 159)，该角色信任 CloudWatch 代表用户将日志数据写入到 Amazon S3 存储桶。之后，用户必须将一个权限策略和一个信任策略附加到新的服务角色。在此情况下，信任策略必须指定 `cloudwatch.amazonaws.com` 元素中的 `Principal`。要查看允许用户将角色传递给 CloudWatch 的策略，请参阅 [IAM：将 IAM 角色传递给特定 Amazon 服务](#) (p. 421)。

通过使用此条件键，您可以确保用户仅为您指定的服务创建服务角色。例如，如果具有之前的策略的用户尝试为 Amazon EC2 创建服务角色，操作将失败。失败的原因是因为用户无权将角色传递到 Amazon EC2。

有时，您会将角色传递给一个服务，随后将角色传递给另一个服务。`iam:PassedToService` 仅包含担任角色的最终服务，而不包括传递角色的中间服务。

Note

某些服务不支持该条件键，例如 Amazon CodeBuild 和 Amazon CodeCommit。

iam:PermissionsBoundary

与字符串运算符 (p. 951) 结合使用。

检查指定的策略附加为 IAM 主体资源上的权限边界。有关更多信息，请参阅 [IAM 实体的权限边界 \(p. 370\)](#)

iam:PolicyARN

与 ARN 运算符 (p. 956) 结合使用。

检查涉及托管策略的请求中的托管策略的 Amazon Resource Name (ARN)。有关更多信息，请参阅 [控制对策略的访问 \(p. 384\)](#)。

iam:ResourceTag/**key-name**

与字符串运算符 (p. 951) 结合使用。

检查附加到身份资源（用户或角色）的标签是否与指定的键名称和键值匹配。

Note

IAM 和 Amazon STS 同时支持 iam:ResourceTag IAM 条件键和 aws:ResourceTag 全局条件键。

您可采用键值对的形式向 IAM 资源添加自定义属性。有关 IAM 资源的标签的更多信息，请参阅 [the section called “标记 IAM 资源” \(p. 276\)](#)。您可以使用 ResourceTag [控制 \(p. 390\)](#) 对 Amazon 资源（包括 IAM 资源）的访问。但是，由于 IAM 不支持组的标签，因此您不能使用标签来控制对组的访问。

此示例说明如何创建基于身份的策略以允许删除具有 **status=terminated** 标签的用户。要使用此策略，请将示例策略中的 **#####** 替换为您自己的信息。然后，按照 [创建策略 \(p. 437\)](#) 或 [编辑策略 \(p. 461\)](#) 中的说明操作。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "iam:DeleteUser",
    "Resource": "*",
    "Condition": {"StringEquals": {"iam:ResourceTag/status": "terminated"}}
  ]
}
```

Amazon Web 联合身份验证的可用键

您可以使用 Web 联合身份验证为通过身份提供程序 (IdP) 进行身份验证的用户提供临时安全凭证。此类提供商的示例包括 Login with Amazon、Amazon Cognito、Google 或 Facebook。在此情况下，当使用临时安全凭证发出请求时，将会有额外条件键可用。您可以使用这些键编写策略，将联合身份用户的访问权限限制为与特定提供商、应用程序或用户关联的资源。这些键通常在角色的信任策略中使用。

amr

与字符串运算符 (p. 951) 结合使用。

示例: cognito-identity.amazonaws.com:amr

如果您正在使用 Amazon Cognito 进行 Web 联合身份验证，则 cognito-identity.amazonaws.com:amr 键（身份验证方法引用）包含有关该用户的登录信息。该键有多个值，这意味着您要在使用 [条件集合运算符 \(p. 958\)](#) 的策略中对它进行测试。该键可包含以下值：

- 如果用户未经过身份验证，则该键仅包含 `unauthenticated`。
- 如果该用户已经过身份验证，则该键包含值 `authenticated` 以及在调用中使用的登录提供商的名称 (`graph.facebook.com`、`accounts.google.com` 或 `www.amazon.com`)。

例如，某 Amazon Cognito 角色信任策略中的以下条件测试用户是否未经过身份验证：

```
"Condition": {
  "StringEquals": {
    { "cognito-identity.amazonaws.com:aud": "us-east-2:identity-pool-id" },
    "ForAnyValue:StringLike": {
      { "cognito-identity.amazonaws.com:amr": "unauthenticated" }
    }
  }
}
```

aud

与字符串运算符 (p. 951) 结合使用。

使用 `aud` 条件键可验证 Google 客户端 ID 或 Amazon Cognito 身份池 ID 是否与您在策略中指定的 ID 匹配。对于同一身份提供程序，可以将 `aud` 键与 `sub` 键结合使用。

示例：

- `accounts.google.com:aud`
- `cognito-identity.amazonaws.com:aud`

`accounts.google.com:aud` 条件键与以下 Google ID 令牌字段匹配。

- 应用程序的 OAuth 2.0 Google 客户端 ID 的 `aud` (如果未设置 `azp` 字段)。如果设置了 `azp` 字段，则 `aud` 字段将与 `accounts.google.com:oauth` (p. 1018) 条件键匹配。
- `azp` (如果设置了 `azp` 字段)。对于 Web 应用程序和 Android 应用程序具有不同的 OAuth 2.0 Google 客户端 ID 但共享相同的 Google API 项目的混合应用程序，可能会出现这种情况。

有关 Google `aud` 和 `azp` 字段的更多信息，请参阅 [Google Identity Platform OpenID Connect](#) 指南。

在使用 `accounts.google.com:aud` 条件键编写策略时，您必须了解该应用程序是否为设置 `azp` 字段的混合应用程序。

azp 字段未设置

以下示例策略适用于未设置 `azp` 字段的非混合应用程序。在此情况下，Google ID 令牌 `aud` 字段值将与 `accounts.google.com:aud` 和 `accounts.google.com:oauth` 条件键值匹配。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "Federated": "accounts.google.com" },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "accounts.google.com:aud": "aud-value",
          "accounts.google.com:oauth": "aud-value",
          "accounts.google.com:sub": "sub-value"
        }
      }
    }
  ]
}
```

azp 字段已设置

以下示例策略适用于设置 azp 字段的混合应用程序。在此情况下，Google ID 令牌 aud 字段值仅与 accounts.google.com:oauth2:aud 条件键值匹配。azp 字段值与 accounts.google.com:aud 条件键值匹配。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {"Federated": "accounts.google.com"},
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "accounts.google.com:aud": "azp-value",
          "accounts.google.com:oauth2:aud": "aud-value",
          "accounts.google.com:sub": "sub-value"
        }
      }
    }
  ]
}
```

id

与字符串运算符 (p. 951) 结合使用。

示例：

- graph.facebook.com:app_id
- graph.facebook.com:id
- www.amazon.com:app_id
- www.amazon.com:user_id

使用这些键可验证应用程序（或站点）ID 或用户 ID 是否与您在策略中指定的 ID 匹配。这适用于 Facebook 或 Login with Amazon。对于同一身份提供程序，可以将 app_id 键与 id 键结合使用。

oauth2:aud

与字符串运算符 (p. 951) 结合使用。

示例: accounts.google.com:oauth2:aud

如果您使用 Google 进行 Web 联合身份验证，则此键将指定该 ID 令牌用于的 Google 受众 (aud)。它必须是应用程序的 OAuth 2.0 客户端 ID 之一。

sub

与字符串运算符 (p. 951) 结合使用。

示例：

- accounts.google.com:sub
- cognito-identity.amazonaws.com:sub

使用这些键可验证用户 ID 是否与您在策略中指定的 ID 匹配。对于同一身份提供程序，可以将 sub 键与 aud 键结合使用。

有关 Web 联合身份验证的更多信息

有关 Web 联合身份验证的更多信息，请参阅以下内容：

- Amazon Mobile SDK for Android 开发人员指南中的 [Amazon Cognito 概述](#)
- Amazon Mobile SDK for iOS 开发人员指南中的 [Amazon Cognito 概述](#)

- [关于 Web 联合身份验证 \(p. 171\)](#)

基于 SAML 的 Amazon STS 联合身份验证的可用键

如果您通过 Amazon Security Token Service (Amazon STS) 使用[基于 SAML 的联合](#)，则可以在策略中包含更多条件键。

SAML 角色信任策略

在角色的信任策略中，您可以包括以下键，以帮助确定发起人是否有权担任角色。除了 `saml:doc`，所有值均源自 SAML 断言。在创建或编辑带条件的策略时，可在 IAM 控制台可视化编辑器中使用列表中的所有项目。标有 [] 的项目可以具有指定类型列表中的值。

`saml:aud`

与[字符串运算符 \(p. 951\)](#)结合使用。

SAML 断言提交到的终端节点 URL。此键的值来自断言中的 `SAML Recipient` 字段，而不是 `Audience` 字段。

`saml:commonName[]`

与[字符串运算符 \(p. 951\)](#)结合使用。

这是 `commonName` 属性。

`saml:cn[]`

与[字符串运算符 \(p. 951\)](#)结合使用。

这是 `eduOrg` 属性。

`saml:doc`

与[字符串运算符 \(p. 951\)](#)结合使用。

这代表担任角色所用的主体。格式为 `account-ID/provider-friendly-name`，例如 `123456789012/SAMLProviderName`。账户 ID 值指拥有 [SAML 提供商 \(p. 188\)](#)的账户。

`saml:edupersonaffiliation[]`

与[字符串运算符 \(p. 951\)](#)结合使用。

这是 `eduPerson` 属性。

`saml:edupersonassurance[]`

与[字符串运算符 \(p. 951\)](#)结合使用。

这是 `eduPerson` 属性。

`saml:edupersonentitlement[]`

与[字符串运算符 \(p. 951\)](#)结合使用。

这是 `eduPerson` 属性。

`saml:edupersonnickname[]`

与[字符串运算符 \(p. 951\)](#)结合使用。

这是 `eduPerson` 属性。

`saml:edupersonorgdn`

与[字符串运算符 \(p. 951\)](#)结合使用。

这是 eduPerson 属性。

saml:edupersonorgunitdn[]

与字符串运算符 (p. 951) 结合使用。

这是 eduPerson 属性。

saml:edupersonprimaryaffiliation

与字符串运算符 (p. 951) 结合使用。

这是 eduPerson 属性。

saml:edupersonprimaryorgunitdn

与字符串运算符 (p. 951) 结合使用。

这是 eduPerson 属性。

saml:edupersonprincipalname

与字符串运算符 (p. 951) 结合使用。

这是 eduPerson 属性。

saml:edupersonscopedaffiliation[]

与字符串运算符 (p. 951) 结合使用。

这是 eduPerson 属性。

saml:edupersontargetedid[]

与字符串运算符 (p. 951) 结合使用。

这是 eduPerson 属性。

saml:eduorghomepageuri[]

与字符串运算符 (p. 951) 结合使用。

这是 eduOrg 属性。

saml:eduorgidentityauthnpolicyuri[]

与字符串运算符 (p. 951) 结合使用。

这是 eduOrg 属性。

saml:eduorglegalname[]

与字符串运算符 (p. 951) 结合使用。

这是 eduOrg 属性。

saml:eduorgsuperioruri[]

与字符串运算符 (p. 951) 结合使用。

这是 eduOrg 属性。

saml:eduorgwhitepagesuri[]

与字符串运算符 (p. 951) 结合使用。

这是 eduOrg 属性。

saml:givenName[]

与字符串运算符 (p. 951) 结合使用。

这是 givenName 属性。

saml:iss

与字符串运算符 (p. 951) 结合使用。

发布者，以 URN 表示。

saml:mail[]

与字符串运算符 (p. 951) 结合使用。

这是 mail 属性。

saml:name[]

与字符串运算符 (p. 951) 结合使用。

这是 name 属性。

saml:namequalifier

与字符串运算符 (p. 951) 结合使用。

基于 SAML 提供商的友好名称的哈希值。该值是以下值按顺序的连接，以“/”字符分隔：

1. Issuer 响应值 (saml:iss)
2. AWS 账户 ID
3. IAM 中 SAML 提供商的友好名称 (ARN 的最后一部分)

账户 ID 与 SAML 提供商的易记名称的串联可作为键 saml:doc 供 IAM policy 使用。有关更多信息，请参阅[唯一标识基于 SAML 的联合中的用户](#) (p. 179)。

saml:organizationStatus[]

与字符串运算符 (p. 951) 结合使用。

这是 organizationStatus 属性。

saml:primaryGroupSID[]

与字符串运算符 (p. 951) 结合使用。

这是 primaryGroupSID 属性。

saml:sub

与字符串运算符 (p. 951) 结合使用。

这是该陈述的主题，其中包含唯一标识组织中某个用户的值 (例如 _cbb88bf52c2510eabe00c1642d4643f41430fe25e3)。

saml:sub_type

与字符串运算符 (p. 951) 结合使用。

此键的值可以是 persistent、transient 或由 SAML 断言中使用的 Format 和 Subject 元素的完整 NameID URI 构成。persistent 值表示在不同会话之间用户的 saml:sub 值是相同的。如果值为 transient，则用户在每个会话中拥有不同的 saml:sub 值。有关 NameID 元素的 Format 属性的信息，请参阅[为身份验证响应配置 SAML 断言](#) (p. 194)。

saml:surname[]

与字符串运算符 (p. 951) 结合使用。

这是 surnameuid 属性。

saml:uid[]

与字符串运算符 (p. 951) 结合使用。

这是 uid 属性。

saml:x500UniqueIdentifier[]

与字符串运算符 (p. 951) 结合使用。

这是 x500UniqueIdentifier 属性。

有关 eduPerson 和 eduOrg 属性的一般信息，请参阅 [REFEDS Wiki 网站](#)。有关 eduPerson 属性的列表，请参阅 [eduPerson 对象类规范 \(201602\)](#)。

列表类型的条件键可以包含多个值。要在策略中创建条件以获取列表值，可以使用 [集合运算符 \(p. 958\)](#) (ForAllValues、ForAnyValue)。例如，要支持从属关系为“教员”或“职员”（而非“学生”）的用户，可以使用下面这样的条件：

```
"Condition": {
  "ForAllValues:StringLike": {
    "saml:edupersonaffiliation": [ "faculty", "staff" ]
  }
}
```

SAML 角色权限策略

在定义允许用户在 Amazon 中访问的内容的 SAML 联合角色权限策略中，您可以包含以下键：

saml:namequalifier

与字符串运算符 (p. 951) 结合使用。

这包含代表 saml:doc 和 saml:iss 值组合的一个哈希值。它用作命名空间限定词；saml:namequalifier 和 saml:sub 组合可唯一标识用户。

saml:sub

与字符串运算符 (p. 951) 结合使用。

这是该陈述的主题，其中包含唯一标识组织中某个用户的值（例如 _cbb88bf52c2510eabe00c1642d4643f41430fe25e3）。

saml:sub_type

与字符串运算符 (p. 951) 结合使用。

此键的值可以是 persistent、transient 或由 SAML 断言中使用的 Format 和 Subject 元素的完整 NameID URI 构成。persistent 值表示在不同会话之间用户的 saml:sub 值是相同的。如果值为 transient，则用户在每个会话中拥有不同的 saml:sub 值。有关 NameID 元素的 Format 属性的信息，请参阅 [为身份验证响应配置 SAML 断言 \(p. 194\)](#)。

有关使用这些密钥的更多信息，请参阅[关于基于 SAML 2.0 的联合身份验证 \(p. 176\)](#)。

Amazon STS 的可用键

您可以对使用 Amazon Security Token Service (Amazon STS) 操作担任的角色使用 IAM 角色信任策略中的以下条件键。

sts:AWSServiceName

与字符串运算符 (p. 951) 结合使用。

使用此键指定可以在其中使用持有者令牌的服务。在策略中使用此条件键时，请使用服务主体指定服务。服务主体是可在策略的 `Principal` 元素中指定的服务的名称。例如，`codeartifact.amazonaws.com` 是 Amazon CodeArtifact 服务主体。

某些 Amazon 服务需要您有权获取 Amazon STS 服务持有者令牌，然后您才能以编程方式访问它们的资源。例如，Amazon CodeArtifact 要求主体使用持有者令牌来执行某些操作。`aws codeartifact get-authorization-token` 命令返回一个持有者令牌。然后，您可以使用持有者令牌来执行 Amazon CodeArtifact 操作。有关持有者令牌的更多信息，请参阅 [使用持有者令牌 \(p. 336\)](#)。

Availability (可用性) - 此键存在于获取持有者令牌的请求中。您不能直接调用 Amazon STS 以获取持有者令牌。当您在其他服务中执行某些操作时，服务代表您请求持有者令牌。

您可以使用此条件键以允许主体获取用于特定服务的持有人令牌。

sts:DurationSeconds

与数字运算符 (p. 953) 结合使用。

使用此键指定主体在获取 Amazon STS 持有者令牌时可以使用的持续时间（以秒为单位）。

某些 Amazon 服务需要您有权获取 Amazon STS 服务持有者令牌，然后您才能以编程方式访问它们的资源。例如，Amazon CodeArtifact 要求主体使用持有者令牌来执行某些操作。`aws codeartifact get-authorization-token` 命令返回一个持有者令牌。然后，您可以使用持有者令牌来执行 Amazon CodeArtifact 操作。有关持有者令牌的更多信息，请参阅 [使用持有者令牌 \(p. 336\)](#)。

Availability (可用性) - 此键存在于获取持有者令牌的请求中。您不能直接调用 Amazon STS 以获取持有者令牌。当您在其他服务中执行某些操作时，服务代表您请求持有者令牌。此键不适用于 Amazon STS `assume-role` 操作。

sts:ExternalId

与字符串运算符 (p. 951) 结合使用。

使用此键可要求主体在代入 IAM 角色时提供特定标识符。

可用性 - 如果主体在使用 Amazon CLI 或 Amazon API 代入角色时提供外部 ID，则请求中存在此键。

在其他账户中担任角色时可能需要的唯一标识符。如果角色所属的账户的管理员为您提供了外部 ID，请在 `ExternalId` 参数中提供该值。该值可以是任意字符串，如密码或账号。外部 ID 的主要功能是解决并防止混淆代理人问题。有关外部 ID 和混淆代理人问题的更多信息，请参阅 [如何在向第三方授予对 Amazon 资源的访问权时使用外部 ID \(p. 164\)](#)。

`ExternalId` 值的长度必须最少为 2 个字符，最多为 1224 个字符。该值必须是字母数字，没有空格。它还可以包含以下符号：加号 (+)、等号 (=)、逗号 (,)、句点 (.)、@ 符号、冒号 (:)、正斜杠 (/) 和连字符 (-)。

sts:RoleSessionName

与字符串运算符 (p. 951) 结合使用。

使用此键可将主体在代入角色时指定的会话名称与策略中指定的值进行比较。

可用性 - 当主体使用 Amazon Web Services Management Console 管理控制台、`assume-role` CLI 命令或 Amazon STS `AssumeRole` API 操作代入角色时，则请求中存在此键。

您可以在角色信任策略中使用此键，以要求您的用户在代入角色时提供特定的会话名称。例如，您可以要求 IAM 用户指定自己的用户名作为其会话名称。在 IAM 用户代入角色后，活动将与匹配用户名的会话名称一起显示在 [Amazon CloudTrail 日志 \(p. 343\)](#) 中。这样，当不同的主体使用角色时，管理员可以更轻松地区分角色会话。

以下角色信任策略要求账户 111122223333 中的 IAM 用户在代入角色时提供其 IAM 用户名作为会话名称。使用条件键中的 `aws:username` [条件变量 \(p. 964\)](#) 来强制执行此要求。此策略允许 IAM 用户代入策略附加到的角色。此策略禁止任何使用临时凭证的人员代入角色，因为 `username` 变量仅适用于 IAM 用户。

Important

您可以使用任何可用的单值条件密钥作为 [变量 \(p. 964\)](#)。您不能使用多值条件键作为变量。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RoleTrustPolicyRequireUsernameForSessionName",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
      "Condition": {
        "StringLike": {"sts:RoleSessionName": "${aws:username}"}
      }
    }
  ]
}
```

当管理员查看操作的 Amazon CloudTrail 日志时，他们可以将会话名称与其账户中的用户名进行比较。在以下示例中，名为 `matjac` 的用户使用名为 `MateoRole` 的角色执行操作。之后，管理员会联系具有名为 `matjac` 用户的 Mateo Jackson。

```
"assumedRoleUser": {
  "assumedRoleId": "AROACQRSTUVWRAOEXAMPLE:matjac",
  "arn": "arn:aws:sts::111122223333:assumed-role/MateoRole/matjac"
}
```

如果您允许 [使用角色进行跨账户访问 \(p. 161\)](#)，则一个账户中的用户可以代入另一个账户中的角色。CloudTrail 中列出的代入角色的 ARN 包括角色所在的账户。它不包括已代入角色的用户的账户。用户仅在账户中是唯一的。因此，我们建议您使用此方法以仅在 CloudTrail 日志中查看您管理的账户中的用户所代入的角色。您的用户可以在多个账户中使用相同的用户名。

sts:SourceIdentity

与 [字符串运算符 \(p. 951\)](#) 结合使用。

使用此键可将主体在代入角色时指定的源身份与策略中指定的值进行比较。

可用性 — 如果主体提供源身份，同时担任使用任何 Amazon STS 担任角色 CLI 命令或 Amazon STS AssumeRole API 操作时，此密钥将会出现在请求中。

您可以在角色信任策略中使用此键，以要求您的用户在代入角色时设置特定的源身份。例如，您可以要求您的工作人员或联合身份为源身份指定值。您可以将身份提供程序 (IdP) 配置为使用与用户关联的属性之一（例如用户名或电子邮件）作为源身份。然后，IdP 会将源身份作为其发送的断言或声明中的一个属性传递给 Amazon。源身份属性的值标识担任角色的用户或应用程序。

在用户担任角色后，活动将在使用已设置源身份值的 [Amazon CloudTrail 日志 \(p. 343\)](#) 中出现。这使管理员能够更轻松地确定什么角色在 Amazon 中执行了操作。您必须授予 `sts:SetSourceIdentity` 操作相应权限以允许身份设置源身份。

与 `sts:RoleSessionName` [\(p. 1023\)](#) 不同，在设置源身份后，无法更改该值。它将存在于源身份对角色执行的所有操作的请求上下文中。当您使用会话凭证担任另一个角色时，该值将保留到后续角色会话中。从一个角色代入另一个角色的过程称为 [角色链 \(p. 159\)](#)。

您可以使用 [aws:SourceIdentity \(p. 1009\)](#) 全局条件键根据后续请求中源身份的值进一步控制对 Amazon 资源的访问权限。

以下角色信任策略允许 IAM 用户 AdminUser 在账户中担任角色 111122223333。它还会向 AdminUser 授予权限来设置源身份，只要源身份设置为 DiegoRamirez 即可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAdminUserAssumeRole",
      "Effect": "Allow",
      "Principal": { "AWS": " arn:aws:iam::111122223333:user/AdminUser" },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": { "sts:SourceIdentity": "DiegoRamirez" }
      }
    }
  ]
}
```

要了解有关使用源身份信息的更多信息，请参阅 [监控和控制使用所担任角色执行的操作 \(p. 316\)](#)。

sts:TransitiveTagKeys

与[字符串运算符 \(p. 951\)](#)结合使用。

使用此键将请求中的可传递会话标签键，与在策略中指定的键进行比较。

Availability (可用性) - 使用临时安全凭证发出请求时，请求中存在此密钥。这些凭证包括使用任何 assume-role 操作或 GetFederationToken 操作创建的凭证。

当您使用临时安全凭证发出请求时，[请求上下文 \(p. 949\)](#)包含 [aws:PrincipalTag \(p. 1005\)](#) 上下文密钥。此键包括[会话标签 \(p. 293\)](#)、[可传递会话标签 \(p. 299\)](#)和角色标签的列表。可传递会话标签是当您使用会话凭证代入另一个角色时，持续传递到所有后续会话中的标记。从一个角色代入另一个角色的过程称为[角色链 \(p. 159\)](#)。

您可以在策略中使用此条件键，以便要求在代入角色或联合身份用户身份时，要求将特定会话标签设置为可传递标签。

Amazon 服务的操作、资源和条件键

每个Amazon服务可以定义操作、资源和条件上下文密钥以在 IAM 策略中使用。有关 Amazon 服务及其操作、资源和条件上下文密钥的列表，请参阅服务授权参考中的[操作、资源和条件键](#)。

了解 IAM 详情的资源

IAM 是一种丰富的产品，您可找到许多资源来帮助了解有关 IAM 如何帮助保护 Amazon 账户和资源的更多信息。

主题

- [用户和组 \(p. 1026\)](#)
- [证书 \(密码、访问密钥和 MFA 设备\) \(p. 1026\)](#)
- [权限与策略 \(p. 1026\)](#)
- [联合和委托 \(p. 1027\)](#)
- [IAM 和其他 Amazon 产品 \(p. 1027\)](#)
- [一般安全实践 \(p. 1027\)](#)
- [一般资源 \(p. 1028\)](#)

用户和组

请参阅这些资源以便创建、管理和使用用户和组。

- [创建您的第一个 IAM 管理员用户和用户组 \(p. 19\)](#) – 演示如何创建 IAM 用户和分配权限的分步过程。
- [IAM 身份 \(用户、用户组和角色\) \(p. 68\)](#) – 有关如何管理 IAM 用户和组的深入讨论。

证书 (密码、访问密钥和 MFA 设备)

查看以下指南为您的 Amazon 账户和 IAM 用户管理密码。您还可以找到有关访问密钥的信息 — 该密钥用于以编程的方式调用 Amazon。

- [Amazon 安全凭证](#) - 介绍您用于访问 Amazon Web Services 的凭证类型，说明如何创建和管理它们，并包含有关安全地管理访问密钥的建议。
- [在 Amazon 中管理用户密码 \(p. 87\)](#) 和 [管理 IAM 用户的访问密钥 \(p. 96\)](#) - 介绍为账户中的 IAM 用户管理凭证的选项。
- [在 Amazon 中使用多重身份验证 \(MFA\) \(p. 104\)](#) - 介绍如何配置账户和 IAM 用户以要求在允许登录之前提供密码和一次性使用代码 (在设备上生成)。(这有时称为双重身份验证)。

权限与策略

了解 IAM policy 的内部工作原理并查找有关授予权限的最佳方式的提示：

- [IAM 中的策略和权限。 \(p. 355\)](#) - 介绍用于定义权限的策略语言。介绍如何将权限附加到用户或组或是 (对于一些 Amazon 产品) 资源本身。
- [IAM JSON 策略元素参考 \(p. 934\)](#) - 提供每个策略语言元素的说明和示例。
- [验证 IAM 策略 \(p. 443\)](#) - 查找用于 JSON 策略验证的资源。
- [IAM 基于身份的策略示例 \(p. 391\)](#) - 说明了用于各种 Amazon 产品中的常见任务的策略示例。
- [Amazon 策略生成器](#) - 通过从列表选择产品和操作来创建自定义策略。
- [IAM policy simulator](#) - 测试策略是允许还是拒绝对 Amazon 的特定请求。

联合和委托

您可以为在其他位置进行了身份验证 (登录) 的用户授予对您 Amazon 账户中的资源的访问权限。这些可以是其他 Amazon 账户中的 IAM 用户 (称为委托)、使用您企业的登录过程进行了身份验证的用户或是来自 Internet 身份提供程序 (如 Login with Amazon、Facebook、Google 或任何其他与 OpenID Connect (OIDC) 兼容的身份提供程序) 的用户。在这些情况下, 用户可获取临时安全凭证以访问 Amazon 资源。

- [IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限 \(p. 31\)](#) - 指导您向其他 Amazon 账户中的 IAM 用户授予跨账户存取权限。
- [临时凭证的常见情形 \(p. 301\)](#) - 介绍在 Amazon 外部进行身份验证之后使用户经过联合身份验证进入 Amazon 的方式。
- [Web Identity Federation Playground](#) - 使您可以体验 Login with Amazon、Google 或 Facebook 以进行身份验证, 然后对 Amazon S3 进行调用。

IAM 和其他 Amazon 产品

大多数 Amazon 产品与 IAM 集成, 以便您可以使用 IAM 功能帮助保护对这些产品中的资源的访问。以下资源讨论 IAM 以及一些最受欢迎的 Amazon 产品的安全性。有关使用 IAM 的产品的完整列表 (包括有关每个产品的更多信息的链接), 请参阅 [使用 IAM 的 Amazon 服务 \(p. 908\)](#)。

将 IAM 与 Amazon EC2 结合使用

- [控制对 Amazon EC2 资源的访问](#) - 介绍如何使用 IAM 功能允许用户管理 Amazon EC2 实例、卷等。
- [使用实例配置文件 \(p. 258\)](#) - 介绍如何使用 IAM 角色为在 Amazon EC2 实例上运行的应用程序以及需要访问其他 Amazon 产品的应用程序安全地提供凭证。

将 IAM 与 Amazon S3 结合使用

- [管理对 Amazon S3 资源的访问权限](#) - 讨论用于存储桶和对象 (包括 IAM policy) 的 Amazon S3 安全模型。
- [编写 IAM policy：授予 Amazon S3 存储桶中用户特定文件夹的访问权限](#) - 讨论如何让用户在 Amazon S3 中保护自己的文件夹。(有关 Amazon S3 和 IAM 的更多文章, 请在博客文章标题下选择 S3 标签。)

将 IAM 与 Amazon RDS 结合使用

- [使用 Amazon Identity and Access Management \(IAM\) 来管理对 Amazon RDS 资源的访问](#) — 介绍如何使用 IAM 控制对数据库实例、数据库快照等的访问。
- [RDS 资源级别权限读本](#) - 介绍如何使用 IAM 控制对特定 Amazon RDS 实例的访问。

将 IAM 与 Amazon DynamoDB 结合使用

- [使用 IAM 控制对 DynamoDB 资源的访问](#) - 介绍如何使用 IAM 允许用户管理 DynamoDB 表和索引。

一般安全实践

查找有关保护 Amazon 账户和资源的最佳方式的专家提示和指南：

- [安全、身份和合规性的最佳实践](#) - 深入介绍如何跨 Amazon 账户和产品管理安全性，包括有关安全架构、IAM 的使用、加密和数据安全等的建议。
- [身份与访问权限管理](#) - 该 Amazon Well-Architected Framework 可帮助您了解在云中设计和运行工作负载的关键概念、设计原则和架构最佳实践。
- [IAM 中的安全最佳实践 \(p. 752\)](#) - 提供有关使用 IAM 帮助保护 Amazon 账户和资源的方式的建议。
- [Amazon CloudTrail 用户指南](#) - 使用 Amazon CloudTrail 可跟踪对 Amazon 进行的 API 调用的历史记录并将这些信息存储在日志文件中。这有助于确定访问了您账户中的资源的用户和账户、进行调用的时间、请求的操作等。

一般资源

浏览以下资源可了解有关 IAM 和 Amazon 的更多信息。

- [IAM 的产品信息](#) - 有关 Amazon Identity and Access Management 产品的一般信息。
- [Amazon Identity and Access Management 开发论坛](#) - 供客户讨论与 IAM 有关的技术问题的社区论坛。
- [入门资源中心](#) - 学习基础知识并开始在 Amazon 上构建。
- [Amazon 动手实践教程](#) - 针对常用 Amazon 使用案例的各种分步教程。
- [课程和研讨会](#) - 指向基于角色的专业课程和自主进度动手实验室的链接，这些课程和实验室旨在帮助您增强 Amazon 技能并获得实践经验。
- [Amazon 开发人员工具](#) - 指向开发人员工具、开发工具包、IDE 工具包和命令行工具的链接，这些资源用于开发和管理 Amazon 应用程序。
- [Amazon 白皮书](#) - 指向 Amazon 技术白皮书的完整列表的链接，这些资料涵盖了架构、安全性、经济性等主题，由 Amazon 解决方案架构师或其他技术专家编写。
- [Amazon Web Services Support 中心](#) - 用于创建和管理 Amazon Web Services Support 案例的中心。还提供指向其他有用资源的链接，如论坛、技术常见问题、服务运行状况以及 Amazon Trusted Advisor。
- [Amazon Web Services Support](#) - 提供有关 Amazon Web Services Support 的信息的主要网页，这是一个一对一的快速响应支持渠道，可以帮助您在云中构建和运行应用程序。
- [联系我们](#) - 用于查询有关 Amazon 账单、账户、事件、滥用和其他问题的中央联系点。
- [Amazon 网站条款](#) - 有关我们的版权和商标、您的账户、许可、网站访问和其他主题的详细信息。

使用 HTTP 查询请求调用 IAM API

目录

- [终端节点 \(p. 1029\)](#)
- [必须使用 HTTPS \(p. 1029\)](#)
- [签署 IAM API 请求 \(p. 1029\)](#)

您可以使用查询 API 以编程方式访问 IAM 和 Amazon STS 服务。查询 API 请求是必须包含参数 `Action` 以指示要执行的操作的 HTTPS 请求。IAM 和 Amazon STS 支持所有操作的 GET 和 POST 请求。也就是说，API 不要求您使用某些操作的 GET 请求和其他操作的 POST 请求。然而，GET 请求受 URL 的大小限制；此限制与浏览器相关，通常为 2048 字节。因此，对于要求更高的查询 API 请求，您必须使用 POST 请求。

响应是 XML 文档。有关响应的详细信息，请参阅 [IAM 参考](#) 或 [Amazon Security Token Service API 参考](#) 中的单个操作页面。

Tip

您可以使用 Amazon SDK 之一，而不是直接调用 IAM 或 Amazon STS API 操作。Amazon SDK 中包含适用于各种编程语言和平台（Java、Ruby、.NET、iOS、Android 等）的库和示例代码。开发工具包提供了可通过编程方式访问 IAM 和 Amazon 的便捷方式。例如，这些开发工具包负责处理各种任务，例如以加密方式签署服务请求（见下文）、管理错误以及自动重试请求。有关 Amazon 开发工具包的信息（包括如何下载及安装），请参阅[适用于 Amazon Web Services 的工具页面](#)。

有关这些 API 操作和错误的详细信息，请参阅 [IAM 参考](#) 或 [Amazon Security Token Service API 参考](#)。

终端节点

IAM 和 Amazon STS 各有一个全球终端节点：

- (IAM) <https://iam.amazonaws.com>
- (Amazon STS) <https://sts.amazonaws.com>

Note

除了全球终端节点，Amazon STS 还支持向区域终端节点发送请求。必须先在一个区域中为您的 Amazon 账户激活 STS，然后才能在该区域中使用 Amazon STS。有关为 Amazon STS 激活其他区域的更多信息，请参阅[在 Amazon 区域中管理 Amazon STS \(p. 331\)](#)。

有关所有服务的 Amazon 终端节点和区域的更多信息，请参阅在 Amazon 一般参考中的[区域和终端节点](#)。

必须使用 HTTPS

由于查询 API 返回安全凭证等敏感信息，必须对所有 API 请求使用 HTTPS。

签署 IAM API 请求

必须使用访问密钥 ID 和秘密访问密钥签署请求。我们强烈建议您不要使用 Amazon Web Services 账户根用户凭证处理 IAM 日常工作。您可以使用 IAM 用户凭证，也可以使用 Amazon STS 生成临时安全凭证。

如需对 API 请求进行签名，建议您使用 Amazon 签名版本 4。有关使用签名版本 4 的信息，请转到 Amazon 常规参考 中的[签名版本 4 签名流程](#)。

如需使用签名版本 2，可以在 [Amazon 常规参考](#) 中获取有关签名版本 2 使用方法的信息。

有关更多信息，请参阅下列内容：

- [Amazon 安全凭证](#)。提供有关用于访问 Amazon 的凭证类型的一般信息。
- [IAM 中的安全最佳实践 \(p. 752\)](#)。提出一系列使用 IAM 服务帮助保护 Amazon 资源的建议。
- [IAM 临时安全凭证 \(p. 300\)](#)。说明如何创建和使用临时安全凭证。

IAM 文件历史记录

下表介绍了 IAM 的主要文档更新。

变更	说明	日期
U2F 弃用和 WebAuthn/FIDO 更新	删除了提及 U2F 作为 MFA 选项的内容，并添加了有关 WebAuthn、FIDO2 和 FIDO 安全密钥的信息。	2022 年 5 月 31 日
IAM 中恢复能力的更新	新增了有关在事件中断 Amazon Web Services 区域之间的通信时保持对 IAM 凭证的访问权限的信息。	2022 年 5 月 16 日
新的资源全局条件键	现在，您可以根据 Amazon Organizations 中的账户、组织单位 (OU) 或组织来控制对包含您的资源的资源的访问权限。您可以使用 IAM policy 中的 <code>aws:ResourceAccount</code> 、 <code>aws:ResourceOrgID</code> 和 <code>aws:ResourceOrgPaths</code> 全局条件键。	2022 年 4 月 27 日
使用 Amazon 开发工具包的 IAM 代码示例	新增的代码示例显示如何将 IAM 与 Amazon 软件开发工具包 (SDK) 一起使用。这些示例被划分成代码摘录，展示如何调用单个服务函数，以及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。	2022 年 4 月 7 日
更新策略评估逻辑流程图	更新 确定账户中是允许还是拒绝请求 部分中的策略评估逻辑流程图和相关文本。	2021 年 11 月 17 日
安全最佳实践的更新	添加了有关创建 IAM 管理员用户而不是使用根用户凭证的信息，移除了使用用户组向 IAM 用户分配权限的最佳做法，并澄清了何时使用托管策略而不是内联策略。	2021 年 10 月 5 日
基于资源的策略的策略评估逻辑主题的更新	添加了有关同一账户中基于资源的策略和不同主体类型的影响的信息。	2021 年 10 月 5 日
对单值和多值条件密钥的更新	现在将更详细地解释单值和多值条件键之间的差异。值类型已添加到每个 Amazon 全局条件上下文密钥 。	2021 年 9 月 30 日
IAM Access Analyzer 支持 Amazon S3 多区域访问点	IAM Access Analyzer 可识别允许公共和跨账户访问的 Amazon S3 存储桶，包括使用 Amazon S3 多区域访问点 的存储桶。	2021 年 9 月 2 日

Amazon 管理策略更新 — 对现有策略的更新	IAM Access Analyzer 更新了现有 Amazon 管理策略。	2021 年 9 月 2 日
支持更多用于生成操作级策略的服务	IAM Access Analyzer 可以为其他 Amazon 服务生成包含操作级访问活动信息的 IAM policy。	2021 年 8 月 24 日
为跨账户跟踪生成 IAM policy	现在，您可以使用 IAM Access Analyzer 根据您在不同账户中使用 Amazon CloudTrail 跟踪的访问活动生成精细策略，例如集中的 Amazon Organizations 跟踪。	2021 年 8 月 18 日
其他 IAM Access Analyzer 策略检查	IAM Access Analyzer 通过添加新的策略检查来验证 IAM policy 中包含的条件，从而扩展策略验证。这些检查会分析策略语句中的条件数据块，并报告安全警告、错误和建议以及可操作建议。 IAM Access Analyzer 添加了以下策略检查： • 错误 — 无效的服务主体格式 • 错误 — 条件中缺少标签密钥 • 安全警告 — 使用不支持的标签条件键拒绝服务的 NotAction • 安全警告 — 使用不支持的标签条件键拒绝服务 • 安全警告 — 缺少配对条件键 • 建议 — 使用不支持的标签条件键允许服务的 NotAction • 建议 — 使用不支持的标签条件键允许服务	2021 年 6 月 29 日
上次访问的操作支持更多服务	现在，您可以在 IAM 控制台中查看上次访问的操作信息，了解 IAM 主体上次对以下服务使用操作的内容：Amazon EC2、IAM、Lambda 和 Amazon S3 管理操作。您也可以使用 Amazon CLI 或 Amazon API 检索数据报告。您可以使用此信息确定不必要的权限，从而优化 IAM policy 以更好地遵循最低权限原则。	2021 年 4 月 19 日
监控和控制使用担任角色执行的操作	管理员可以配置 IAM 角色，以要求身份传递已在 Amazon CloudTrail 中登录的源身份。查看源身份信息可帮助管理员确定谁通过担任角色会话执行了操作或执行了哪些操作。	2021 年 4 月 13 日

基于访问活动生成 IAM policy	现在，您可以使用 IAM Access Analyzer 根据在 Amazon CloudTrail 中找到的访问活动生成精密策略了。	2021 年 4 月 7 日
IAM Access Analyzer 策略检查	IAM Access Analyzer 现在在策略创作过程中提供了 100 多项策略检查，并附带了可操作的建议。	2021 年 3 月 16 日
扩展的策略验证选项	使用 IAM Access Analyzer 中的策略检查扩展 IAM 控制台、Amazon API 以及 Amazon CLI 中可用的策略验证来帮助您编写安全且功能性强的 JSON 策略。	2021 年 3 月 15 日
标记 IAM 资源	您现在可以使用标签键值对标记额外的 IAM 资源。	2021 年 2 月 11 日
IAM 用户的默认密码策略	如果您没有为 Amazon 账户设置自定义密码策略，则 IAM 用户密码现在必须符合默认 Amazon 密码策略。	2020 年 11 月 18 日
Amazon 服务的操作、资源和条件键页面已移动	每个 Amazon 服务可以定义操作、资源和条件上下文密钥以在 IAM policy 中使用。有关 Amazon 服务及其操作、资源和条件上下文密钥的列表，请参阅服务授权参考。	2020 年 11 月 16 日
IAM 用户更长的角色会话持续时间	IAM 用户在 Amazon Web Services Management Console 中切换角色时现在可以拥有更长的角色会话持续时间，从而减少由于会话过期而造成的中断。用户被授予为角色设置的最大会话持续时间或 IAM 用户会话中的剩余时间（以较少者为准）。	2020 年 7 月 24 日
使用 Service Quotas 请求快速增加 IAM 实体	您可以使用 Service Quotas 控制台请求增加可调 IAM 配额的配额。现在，某些增加将在 Service Quotas 中几分钟内自动得到批准并可用于您的账户中。更大的增加请求将提交给 Amazon Web Services Support。	2020 年 6 月 25 日
Amazon S3 中的上次访问信息现在包括 IAM 管理操作	除了服务上次访问的信息之外，您现在可以在 IAM 控制台中查看有关 IAM 主体上次使用 Amazon S3 操作的时间的信息。您也可以使用 Amazon CLI 或 Amazon API 检索数据报告。该报告包括有关主体上次尝试访问的允许的服务和操作以及访问时间的信息。您可以使用此信息确定不必要的权限，从而优化 IAM policy 以更好地遵循最低权限原则。	2020 年 6 月 3 日

添加了安全性章节	安全性章节可帮助您了解如何配置 IAM 和 Amazon STS 以满足您的安全性和合规性目标。您还会了解如何使用其他 Amazon 服务以帮助您监控和保护 IAM 资源。	2020 年 4 月 29 日
sts:RoleSessionName	您现在可以编写一个策略，该策略基于主体在代入角色时指定的会话名称授予权限。	2020 年 4 月 21 日
Amazon 登录页面更新	在主 Amazon 登录页面上登录时，您无法选择以 Amazon Web Services 账户根用户或 IAM 用户身份登录。当您执行此操作时，页面上的标签会指示您是否应提供根用户电子邮件地址或 IAM 用户账户信息。本文档包含更新的屏幕截图，以帮助您了解 Amazon 登录页面。	2020 年 3 月 4 日
aws:ViaAWSService 和 aws:CalledVia 条件键	您现在可以编写策略来限制服务是否可以代表 IAM 主体（用户或角色）发出请求。当主体向 Amazon 服务发出请求时，该服务可能会使用主体的凭证向其他服务发出后续请求。如果任何服务使用主体的凭证发出请求，请使用 <code>aws:ViaAWSService</code> 条件键进行匹配。如果特定服务使用主体的凭证发出请求，请使用 <code>aws:CalledVia</code> 条件键进行匹配。	2020 年 2 月 20 日
策略模拟器添加对权限边界的支持	您现在可以使用 IAM policy simulator 测试权限边界对 IAM 实体的影响。	2020 年 1 月 23 日
跨账户策略评估	现在，您可以了解 Amazon 如何评估跨账户访问策略。当信任账户中的资源包含的基于资源的策略允许另一个账户中的主体访问该资源时，就会发生此情况。这两个账户都必须允许该请求。	2020 年 1 月 2 日
会话标签	现在，您可以在 Amazon STS 中代入角色或联合身份用户身份时包含标签。执行 <code>AssumeRole</code> 或 <code>GetFederationToken</code> 操作时，您可以将会话标签作为属性传递。执行 <code>AssumeRoleWithSAML</code> 或 <code>AssumeRoleWithWebIdentity</code> 操作时，您可以将属性从公司身份传递到 Amazon。	2019 年 11 月 22 日

控制 Amazon Organizations 中 Amazon 账户组的访问权限	现在，您可以在 IAM policy 中从 Amazon Organizations 引用企业部门 (OU)。如果您使用 Organizations 将账户组织到 OU 中，则可以要求主体属于特定 OU，然后再向其授予对您资源的访问权限。主体包括 Amazon Web Services 账户 根用户、IAM 用户和 IAM 角色。为此，请在策略的 <code>aws:PrincipalOrgPaths</code> 条件键中指定 OU 路径。	2019 年 11 月 20 日
上次使用的角色	您现在可以查看上次使用角色的日期、时间和区域。此信息还可帮助您确定账户中未使用的角色。您可以使用 Amazon Web Services Management Console、Amazon CLI 和 Amazon API 查看有关上次使用角色时间的信息。	2019 年 11 月 19 日
对全局条件上下文密钥页面的更新	现在，您可以了解什么时候在请求的上下文中包含各个全局条件键。您还可以使用页面目录 (TOC) 轻松地导航到各个键。页面上的信息可帮助您编写更准确的策略。例如，如果您的员工通过 IAM 角色使用联合身份，则您应使用 <code>aws:userId</code> 密钥而不是 <code>aws:userName</code> 密钥。 <code>aws:userName</code> 密钥仅适用于 IAM 用户，不适用于角色。	2019 年 10 月 6 日
Amazon 中的 ABAC	了解如何通过标签在 Amazon 中使用基于属性的访问控制 (ABAC)，以及它与传统 Amazon 授权模型的比较。使用 ABAC 教程，了解如何创建策略，允许具有主体标签的 IAM 角色访问具有匹配标签的资源，并测试该策略。此策略仅允许用户查看或编辑其作业需要的 Amazon 资源。	2019 年 10 月 3 日
Amazon STS GetAccessKeyInfo 操作	您可以在代码中查看 Amazon 访问密钥，以确定密钥是否来自于您拥有的账户。您可以使用 <code>aws sts get-access-key-info</code> Amazon CLI 命令或 <code>GetAccessKeyInfo</code> Amazon API 操作传递访问密钥 ID。	2019 年 7 月 24 日

在 IAM 中查看 Organizations 服务的上次访问信息	您现在可以在 IAM 控制台的 Amazon Organizations 部分中查看 Amazon Organizations 实体或策略的上次访问的服务信息。您也可以使用 Amazon CLI 或 Amazon API 检索数据报告。该数据包括有关 Organizations 账户中的主体上次尝试访问的允许服务以及访问时间的信息。您可以使用该信息确定不需要的权限，以便优化 Organizations 策略以更好地遵循最小权限原则。	2019 年 6 月 20 日
将托管策略用作会话策略	您现在可以在担任角色时最多传递 10 个托管策略 ARN。这允许您限制角色的临时凭证的权限。	2019 年 5 月 7 日
全局终端节点的会话令牌的 Amazon STS 区域兼容性	您现在可以选择是使用版本 1 还是版本 2 全球终端节点令牌。版本 1 令牌仅在默认启用的 Amazon 区域中有效。这些令牌不适用于手动启用的区域，例如，亚太地区（香港）。版本 2 令牌在所有区域中都有效。不过，版本 2 令牌较长，可能会影响临时存储令牌的系统。	2019 年 4 月 26 日
允许启用和禁用 Amazon 区域	您现在可以创建策略，以允许管理员启用和禁用亚太地区（香港）区域（ap-east-1）。	2019 年 4 月 24 日
IAM 用户的“我的安全凭证”页面	IAM 用户现在可在 My Security Credentials（我的安全凭证）页面上管理自己的所有凭证。此 Amazon Web Services Management Console 页面显示账户信息，例如账户 ID 和规范用户 ID。用户还可以查看和编辑自己的密码、访问密钥、X.509 证书、SSH 密钥和 Git 凭证。	2019 年 1 月 24 日
访问顾问 API	您现在可使用 Amazon CLI 和 Amazon API 查看上次访问的服务相关信息。	2018 年 12 月 7 日
标记 IAM 用户和角色	您现在可使用 IAM 标签利用标签键值对向身份（IAM 用户或角色）添加自定义属性。您还可以使用标签控制身份对资源的访问权限或控制可附加到身份的标签。	2018 年 11 月 14 日
U2F 安全密钥	现在，您可以使用 U2F 安全密钥作为登录 Amazon Web Services Management Console 时的 Multi-Factor Authentication (MFA) 选项。	2018 年 9 月 25 日

对 Amazon VPC 终端节点的支持	在美国西部（俄勒冈）区域中，您现在可以在 VPC 与 Amazon STS 之间建立私有连接。	2018 年 7 月 31 日
权限边界	利用新功能，可以更轻松地向可信员工授予管理 IAM 权限的能力，而无需授予完整的 IAM 管理访问权限。	2018 年 7 月 12 日
aws:PrincipalOrgID	通过指定 IAM 主体的 Amazon 企业，新的条件键提供了控制对 Amazon 资源的访问的更简单方法。	2018 年 5 月 17 日
aws:RequestedRegion	新的条件键提供了使用 IAM policy 来控制对 Amazon 区域的访问的更简单方法。	2018 年 4 月 25 日
增加了 IAM 角色的会话持续时间	IAM 角色 角色现在可以具有 12 小时的会话持续时间。	2018 年 3 月 28 日
更新了角色创建工作流程	新的工作流程改进了创建信任关系并将权限附加到角色的过程。	2017 年 9 月 8 日
Amazon 账户登录过程	更新的 Amazon 登录体验允许根用户和 IAM 用户使用 Amazon Web Services Management Console 主页上的 Sign In to the Console（登录控制台）链接。	2017 年 8 月 25 日
示例 IAM policy	文档更新包含 30 多个示例策略。	2017 年 8 月 2 日
IAM 最佳实践	利用添加到 IAM 控制台的 Users（用户）部分中的信息，可以更轻松地遵循 IAM 最佳实践。	2017 年 5 月 7 日
Auto Scaling 资源	资源级权限可以控制对 Auto Scaling 资源的访问以及 Auto Scaling 资源的权限。	2017 年 5 月 16 日
Amazon RDS for MySQL 和 Amazon Aurora 数据库	数据库管理员可以将数据库用户与 IAM 用户和角色相关联，从而从一个位置管理对所有 Amazon 资源的用户访问。	2017 年 4 月 24 日
服务相关角色	服务相关角色提供了更轻松、更安全的方法来将权限委派给 Amazon 服务。	2017 年 4 月 19 日
策略摘要	新的策略摘要使您可以更轻松地了解 IAM policy 中的权限。	2017 年 3 月 23 日