
AWS Identity and Access Management

Using IAM

API Version 2010-05-08



Amazon Web Services

AWS Identity and Access Management: Using IAM

Amazon Web Services

Copyright © 2013 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

The following are trademarks or registered trademarks of Amazon: Amazon, Amazon.com, Amazon.com Design, Amazon DevPay, Amazon EC2, Amazon Web Services Design, AWS, CloudFront, EC2, Elastic Compute Cloud, Kindle, and Mechanical Turk. In addition, Amazon.com graphics, logos, page headers, button icons, scripts, and service names are trademarks, or trade dress of Amazon in the U.S. and/or other countries. Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon.

All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Welcome	1
Getting Started	4
Creating an Admins Group	5
What Your Users Need to Know	10
Where to Go Next	10
Best Practices	13
Introduction to IAM	18
What Is IAM?	18
IAM Concepts	21
Business Use Cases	27
Permissions and Policies	30
Overview of Permissions	30
Overview of Policies	32
The Access Policy Language	33
Key Concepts	33
Architectural Overview	35
Evaluation Logic	37
How to Write a Policy	43
Basic Policy Structure	43
Element Descriptions	43
Supported Data Types	57
Ways to Access IAM	59
About IAM Entities	60
Identifiers for IAM Entities	60
Limitations on IAM Entities	64
Setting Up an Administrators Group	67
Working with Users and Groups	72
Adding a New User to Your AWS Account	72
Listing Users	79
Deleting a User from Your AWS Account	82
Creating and Listing Groups	85
Adding Users to and Removing Users from a Group	89
Deleting a Group	91
Renaming Users and Groups	93
Managing Passwords	95
Changing Your AWS Account Password	96
Managing an IAM Password Policy	96
Granting an IAM User Permission to Manage the Password Policy	97
Displaying, Creating, Changing, or Deleting a Password Policy (AWS Management Console)	97
Displaying, Creating, Changing, or Deleting a Password Policy (API and CLI)	99
Creating, Changing, or Deleting a Password for an IAM User	100
Creating, Changing, or Deleting an IAM User Password (AWS Management Console)	100
Creating, Changing, or Deleting an IAM User Password (API and CLI)	103
Granting IAM Users Permission to Change Their Own Password	104
How IAM Users Change Their Own Password	105
Using Multi-Factor Authentication (MFA) Devices with AWS	106
Setting Up an MFA Device	107
Checking MFA Status	107
Using a Virtual MFA Device with AWS	108
Configuring and Enabling a Virtual MFA Device for a User	109
Configuring and Enabling a Virtual MFA Device for Your AWS Account	111
Using the IAM CLI or API with Virtual MFA Devices	113
Installing the AWS Virtual MFA Mobile Application	114
Enabling a Hardware MFA Device for Use with AWS	114
Synchronizing an MFA Device	117
Deactivating an MFA Device	118

Configuring MFA-Protected API Access	120
What If an MFA Device Is Lost or Stops Working?	124
Managing User Keys and Certificates	124
Creating, Modifying, and Viewing User Security Credentials	125
Creating and Uploading a User Signing Certificate	129
Adding a Credentials Management Policy to the User	135
Rotating Credentials	136
Delegating API Access by Using Roles	138
Granting Applications that Run on Amazon EC2 Instances Access to AWS Resources	139
Instance Profiles	142
Enabling Cross-Account API Access	142
Enabling Cross-Account API Access Walkthrough	144
Creating a Role	145
Modifying Group Permissions	148
Assuming a Role	149
Delegating API Access to Third Parties	150
Delegating API Access Within an AWS Account	151
Creating a Role	151
Modifying a Role	157
Assuming a Role	160
Deleting Roles or Instance Profiles	160
Troubleshooting Roles	162
Cross-Account Access Using Resource-Based Policies	168
Managing IAM Policies	173
Example Policies for IAM Entities	179
Controlling User Access to Your AWS Account Billing Information	183
Controlling User Access to the AWS Management Console	188
How Users Sign In to the AWS Management Console	190
The AWS Management Console Sign-in Page	190
Using an Alias for Your AWS Account ID	192
MFA Devices and Your IAM-Enabled Sign-in Page	194
Integrating with Other AWS Products	195
Managing Server Certificates	198
Actions on Server Certificates	198
Renaming Server Certificates	199
Creating and Uploading Server Certificates	199
Making Query Requests	207
Document History	209

Welcome

Topics

- [Using IAM to Give Users Access to Your AWS Resources \(p. 1\)](#)
- [Do I Need to Sign Up for IAM? \(p. 2\)](#)
- [How Do I... ? \(p. 3\)](#)

AWS Identity and Access Management (IAM) helps you securely control access to Amazon Web Services and your account resources. IAM can also keep your account credentials private. With IAM, you can create multiple IAM users under the umbrella of your AWS account or enable temporary access through identity federation with your corporate directory. In some cases, you can also enable access to resources across AWS accounts.

Without IAM, however, you must either create multiple AWS accounts—each with its own billing and subscriptions to AWS products—or your employees must share the security credentials of a single AWS account. In addition, without IAM, you cannot control the tasks a particular user or system can do and what AWS resources they might use.

To learn more about IAM and its features, see [What Is IAM? \(p. 18\)](#)

This guide provides a conceptual overview of IAM, describes business use cases, and explains AWS permissions and policies.

Using IAM to Give Users Access to Your AWS Resources

Here are the ways you can use IAM to control access to your AWS resources.

Type of access	Why would I use it?	Where can I get more information?
Access for users under your AWS account	You want to add users under the umbrella of your AWS account, and you want to use IAM to create users and manage their permissions.	To learn how to use the AWS Management Console to create users and to manage their permissions under your AWS account, see Getting Started (p. 4) . To learn about using the IAM application programming interface (API) or command line interface (CLI) to create users under your AWS account, see Setting Up an Administrator's Group . For more information about working with IAM users, see Working with Users and Groups .
Non-AWS user access via identity federation between your authorization system and AWS	You have non-AWS users in your identity and authorization system, and they need access to your AWS resources.	To learn how to use security tokens to give your users access to your AWS account resources through federation with your corporate directory, go to Using Temporary Security Credentials . For information about the AWS Security Token Service API, go to the AWS Security Token Service API Reference .
Cross-account access between AWS accounts	You want to share access to certain AWS resources with users under other AWS accounts.	To learn how to use IAM to grant permissions to other AWS accounts, see Delegating API Access by Using Roles or Cross-Account Access Using Resource-Based Policies .

Do I Need to Sign Up for IAM?

IAM is a feature of your AWS account. If you are already signed up for a product that is integrated with IAM, you don't need to do anything else to sign up for IAM, nor will you be charged extra for using it.

Note

IAM works only with AWS products that are integrated with IAM. For a list of such products, see [Integrating with Other AWS Products](#).

If you don't already have an AWS account, you need to create one to use IAM. You create an AWS account when you sign up to use an AWS product for the first time.

To sign up for AWS

1. Go to <http://aws.amazon.com>, and then click **Sign Up Now**.
2. Follow the on-screen instructions.

Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

AWS product documentation is available at [AWS Documentation](#).

How Do I... ?

Here are some resources to help you get things done with AWS Identity and Access Management.

How Do I...	Relevant Resources
Learn more about the business cases for AWS Identity and Access Management	Business Use Cases (p. 27)
Get started with IAM	Getting Started (p. 4)
Get the release notes	Release Notes
Get the FAQ	AWS Identity and Access Management FAQ
Learn more about how IAM works	What Is IAM? (p. 18)
Get developer tools	Developer Tools
Get the Java library	Sample Code & Libraries
Get technical support	AWS Support Center
Get premium technical support	AWS Premium Support Center
Get community support	IAM Discussion Forums
Contact AWS	Contact Us

For definitions of AWS terms, go to the [Amazon Web Services Glossary](#).

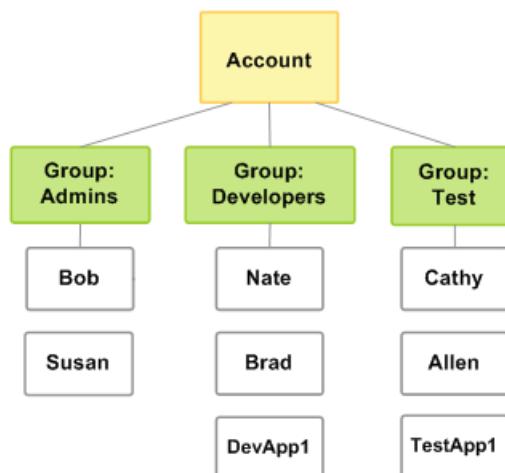
Getting Started

Topics

- [Creating an Admins Group \(p. 5\)](#)
- [What Your Users Need to Know \(p. 10\)](#)
- [Where to Go Next \(p. 10\)](#)

This topic shows you how to give access to your AWS resources by creating users under your AWS account. First, you'll learn concepts you should understand before you create groups and users, and then you'll walk through how to perform the necessary tasks using the AWS Management Console. The first task is to set up an administrators group for your AWS account. Having an administrators group for your AWS account isn't required, but we strongly recommend it.

The following figure shows a simple example of an AWS account with three groups. A group is a collection of users who have similar responsibilities. In this example, one group is for administrators (it's called *Admins*). There's also a *Developers* group and a *Test* group. Each group has multiple users. Each user can be in more than one group, although the figure doesn't illustrate that. You can't put groups inside other groups. You use policies to grant permissions to groups.



In the procedure that follows, you will perform the following tasks:

- Create an Admins group

- Create the policy controlling permissions for the group
- Create or add the users who will be in the Admins group
- Create access keys for users who need them
- Create passwords for users who need them

You will grant the Admins group permission to access all your available AWS account resources. Available resources are any AWS products you use, or that you are signed up for. Users cannot access your AWS account information, including the following:

- Account profile information
- Billing and metering information
- Security credentials

Tip

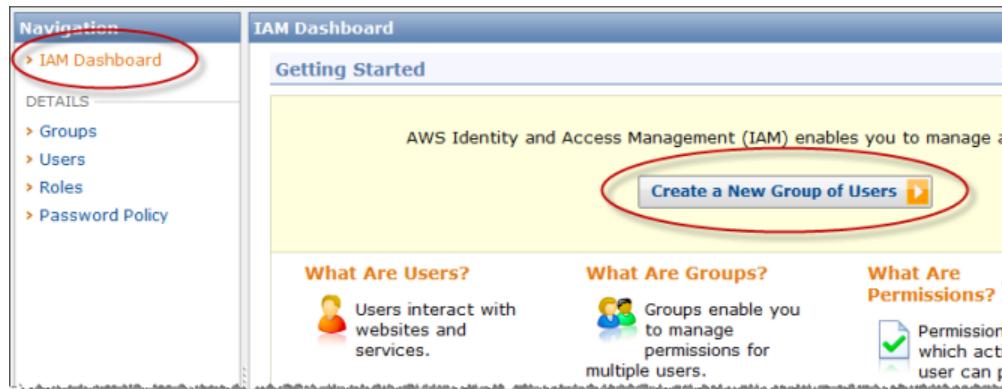
You should create a user for yourself and add it to your Admins group. Then, after you establish the Admins group and yourself as a user in the group, all interaction with your AWS account should be at the user level, not at the AWS account level. Limiting the use of your AWS account credentials will help ensure that when you want to rotate credentials for a user or for the AWS account, potential impact is limited. For more information about the credentials and the security benefits of rotating credentials, go to [Managing User Keys and Certificates](#) in *Using AWS Identity and Access Management*.

Creating an Admins Group

This procedure describes how to create an Admins group and grant the group the necessary permissions to access your AWS resources.

To create the Admins group

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. From the **IAM Dashboard**, click **Create a New Group of Users**.



3. Name the group **Admins**, and then click **Continue**.

AWS Identity and Access Management Using IAM

Creating an Admins Group

The screenshot shows the 'Create a New Group of Users Wizard' with the 'GROUP NAME' tab selected. A progress bar at the top indicates steps 1 through 4. The main area contains a text input field labeled 'Group Name' with 'Admins' typed in. Below it is a note: 'Specify a group name. Group names can be edited any time.' and a help text: 'Example: Developers or ProjectAlpha' and 'Maximum 128 characters'. A 'Continue' button with a yellow arrow icon is at the bottom.

Note

You can use only certain characters to name a group. For information about limitations on group names, see [Limitations on IAM Entities](#) in *Using AWS Identity and Access Management*.

4. Next, you need to use a policy to assign permissions to your Admins group. A policy is a document that formally states one or more permissions. To select a policy template, next to **Administrator Access**, click **Select**.

The screenshot shows the 'Create a New Group of Users Wizard' with the 'PERMISSIONS' tab selected. A progress bar at the top indicates steps 1 through 4. The main area is titled 'Set Permissions' and contains a note: 'Select a policy template, generate a policy, or create your own custom policy to apply permissions to Admins. A policy is a document that formally states one or more permissions. You can edit the policy on the following screen, or at a later time using the User or Group detail pages.' Below this is a list of policy templates:

- Administrator Access**: Provides full access to AWS services and resources. A 'Select' button is highlighted with a red circle.
- Power User Access**: Provides full access to AWS services and resources, but does not allow management of Users and groups. A 'Select' button is shown.
- Read Only Access**: Provides read-only access to AWS services and resources. A 'Select' button is shown.
- AWS CloudFormation Read Only Access**: Provides access to AWS CloudFormation via the AWS Management Console. A 'Select' button is shown.

At the bottom are links for 'Policy Generator', 'Custom Policy', and 'No Permissions', and a 'Back' link.

IAM provides several policy templates you can use to automatically assign permissions to the groups you create. The Administrator Access policy template gives the Admins group permission to access all account resources, *except* your AWS account information as described previously.

5. Next, you can optionally edit the policy and the policy name. If you used a policy template, the default policy name includes the template name and today's date. If you want to make any changes to the policy before you apply it to the group, you can use the **Edit Permissions** screen to edit the policy. If you don't want to edit the policy, or if you are finished making edits, click **Continue**. (Policies are described in detail in [Permissions and Policies](#) in *Using AWS Identity and Access Management*.)

AWS Identity and Access Management Using IAM

Creating an Admins Group

Create a New Group of Users Wizard

Cancel X

GROUP NAME PERMISSIONS USERS REVIEW

Edit Permissions

You can customize permissions by editing the policy document below. For more information about the access policy language, see [Key Concepts](#) in Using AWS Identity and Access Management. You can create a policy document using the [AWS Policy Generator](#).

Policy Name
AdministratorAccess-Admins-201104231049

Policy Document

```
{  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "*",  
      "Resource": "*"  
    }  
  ]  
}
```

< Back Continue >

Your Admins group has permissions; now it needs one or more users.

6. Add yourself as an administrator for your AWS account. Also add any other users you want to be administrators for your AWS account.
 - a. To add yourself as a user to your Admins group, select the **Create New Users** tab and enter the user name you want to assign to yourself.

Create a New Group of Users Wizard

Cancel X

GROUP NAME PERMISSIONS USERS REVIEW

Users below will be added to your **Admins** group.

Create New Users Add Existing Users

Enter User Names:

1. YourName
2. Bob
3. Susan
4.
5.
Maximum 128 characters each

Generate an access key for each User
Users need access keys to make secure REST or Query protocol requests to AWS service APIs.
For Users who need access to the [AWS Management Console](#), create a password in the Users panel after completing this wizard.

< Back Continue >

- b. Use the same process to create new users or add existing users to the Admins group.
- c. If you want to automatically generate security credentials for your new users, on the **Create New Users** tab, make sure that **Generate an access key for each user** is selected. Users

need security credentials if they will be working with an AWS API. For information about creating security credentials for existing users, go to [Managing User Keys and Certificates](#) in *Using AWS Identity and Access Management*.

Note

If you have users who will be working with the AWS Management Console, you will need to create passwords for each of them. Creating passwords is described later in this procedure.

Note

You can edit user properties later, but you cannot use the console to change the user name. For information about editing user properties, go to [Working with Users and Groups](#) in *Using AWS Identity and Access Management*. For information on limitations on user names, see [Limitations on IAM Entities](#) in *Using AWS Identity and Access Management*.

- d. When you are finished adding users to this group, click **Continue**.

7. Review the group details on the confirmation screen, and then click **Finish**.
8. If you chose to create security credentials for the users, click **Download Credentials** to save the Access Key IDs and Secret Access Keys to a .csv file on your computer. You will need to provide each user with their Access Key ID and Secret Access Key before they can begin using an AWS API. For security reasons, you cannot retrieve the Secret Access Key after this step.



9. When you are finished downloading your users' security credentials, click **Close Window** to continue. Now you are now ready to add passwords for your users.
10. Users who will access the AWS Management Console will need a password. To add a password for a user
 - a. On the **Navigation** pane, click **Users**.
 - b. Select the user who you want to create a password for, and then select the user **Security Credentials** tab.
 - c. Click **Manage Password**.

AWS Identity and Access Management Using IAM

Creating an Admins Group

The screenshot shows the AWS IAM 'Users' interface. In the main table, 'Bob' is selected and highlighted with a red circle. Below the table, a message indicates '1 User selected'. Under 'User: Bob', there are tabs for 'Groups', 'Permissions', 'Security Credentials' (which is selected), and 'Summary'. The 'Security Credentials' section contains two main parts: 'Access Credentials' and 'Sign-In Credentials'. In 'Access Credentials', it shows an access key labeled 'AK [REDACTED] BQ - Active' created on 2011-04-29T22:11:41Z, with a 'Manage Access Keys' button. In 'Sign-In Credentials', it shows 'User name: Bob' and 'Password: No', with a 'Manage Password' button circled in red. There is also a 'Multi-Factor Authentication Device: No' section with a 'Manage MFA Device' button.

- d. You can create a custom password, or you can have IAM automatically generate a password.
- To create a custom password, select **Assign a custom password**, and then enter and confirm the password. When you are finished, click **Apply**. The selected user can begin using the password. For information about how users access your sign-in page, see [The AWS Management Console Sign-in Page](#) in *Using AWS Identity and Access Management*.
 - To have IAM generate a password, select **Assign an auto-generated password**, and then click **Apply**. Click **Download Credentials** to save the password as a .csv file to your computer. You will need to provide this password to the user, and you will not be able to access the password after completing this step. Click **Close Window** to continue. For more information, see [What Your Users Need to Know \(p. 10\)](#).

The screenshot shows the 'Manage Password' dialog box. It displays a green checkmark and the message 'Your password has been created successfully.' It states that this is the last time the user security credentials will be available for download. A 'Download Credentials' button is shown, and a note says 'You can manage and recreate these credentials any time.' Below this, a section for 'User Security Credentials' is expanded, showing 'Bob' with a password field containing 'Insecure8'. There are 'Close Window' and 'Cancel' buttons at the bottom.

Congratulations, you have created an Admins group, added permissions for the group, and added users to the group. You can use the same process to create more groups and users, and to give

your users access to your AWS account resources. To learn about using policies to restrict user access to specific AWS resources, go to [Integrating with Other AWS Products](#) in *Using AWS Identity and Access Management*.

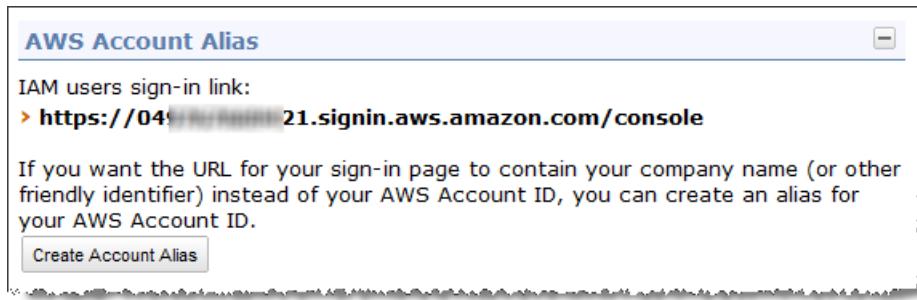
Continue to [What Your Users Need to Know \(p. 10\)](#)

What Your Users Need to Know

When you are finished creating groups, assigning permissions, and adding users, you need to provide each user with the following information:

- The user name
- The password (if you created one for the user)
- The Access Key ID and Secret Access Key (if you created these for the user)
- The AWS account sign-in page URL (if the user will be accessing your account through the AWS Management Console)

Your AWS account sign-in page URL is displayed on the IAM Dashboard under **AWS Account Alias**. For information about creating an account alias for the URL, go to [Using an Alias for Your AWS Account ID](#) in *Using AWS Identity and Access Management*.



Continue to [Where to Go Next \(p. 10\)](#)

Where to Go Next

Topics

- [Learn More About IAM \(p. 11\)](#)
- [Learn About Policies and Permissions \(p. 11\)](#)
- [Other Ways to Access IAM \(p. 11\)](#)
- [IAM Resources \(p. 12\)](#)

The Getting Started shows you how to use AWS Identity and Access Management to create groups and users within your AWS account, but IAM also enables you to do many tasks not covered there. This section provides links to additional resources that will help you deepen your understanding of permissions and how to use IAM with other AWS products to control users' access.

Learn More About IAM

To learn more about IAM, read the following sections:

- [Business Use Cases](#) describes how you might use IAM with other AWS products.
- [Working with Users and Groups](#) has important details you need to know about managing users and groups in your AWS account.
- [Integrating with Other AWS Products](#) links to information about how IAM works with other AWS products (for example, Amazon EC2, Amazon S3, etc.), and what you can do to control your users' access to specific AWS products and resources.
- *Friendly Names and Paths* in [Identifiers for IAM Entities](#) explains how to use *paths* to logically separate groups and users based on organizational structure.

Learn About Policies and Permissions

To learn more about IAM policies and permissions, refer to the following resources:

- [Permissions and Policies](#) describes permissions, policies, and the access policy language you use to write policies.
- [Example Policies for IAM Entities](#) provides examples of policies that control access for your IAM resources.
- The [AWS Policy Generator](#) is a tool you can use to generate AWS policies automatically.

Other Ways to Access IAM

This guide has shown you how to create users and groups using the AWS Management Console. You can continue using IAM through the console, or you can try one of the other interfaces.

Use the Command Line Interface to Issue Commands

If you want to issue commands with the IAM command line interface (CLI), go to the [AWS Identity and Access Management Command Line Interface Reference](#). The guide describes how to set up and use the IAM CLI, describes the commands, and lists available commands by function.

Use an Existing Library

AWS offers SDKs that let you access IAM programmatically to create groups, users, and so on. The following SDKs are available:

- [AWS SDK for Java](#)
- [AWS SDK for PHP](#)
- [AWS SDK for .NET](#)

Code Directly to the Web Service API

If you want to write code directly to the IAM Query API, go to [AWS Identity and Access Management API Reference](#). The guide describes how to create and authenticate API requests. [Making Query Requests](#) describes the basics about using the Query API.

IAM Resources

Use the following resources to learn more about how to work with this service.

Resource	Description
Using AWS Identity and Access Management	Describes how to use the service and all its features through the AWS Management Console, the CLI, or API.
AWS Identity and Access Management Command Line Interface Reference	Gives complete descriptions of the commands in the CLI.
AWS Identity and Access Management API Reference	Gives the WSDL and schema location; complete descriptions of the API actions, parameters, and data types; and a list of errors that the service returns.
AWS Identity and Access Management Quick Reference Card	Gives a concise listing of the commands you use with the CLI.
Product Information for IAM	The primary web page for information about IAM.
IAM Release Notes	The release notes give a high-level overview of the current release. They specifically note any new features, corrections, and known issues.
AWS Developer Resource Center	A central starting point to find documentation, code samples, release notes, and other information to help you build innovative applications with AWS.
IAM Discussion Forum	A community-based forum for developers to discuss technical questions related to IAM.
AWS Support Center	The home page for AWS Technical Support, including access to AWS developer forums, technical FAQs, service health dashboard, and premium support (if you are subscribed to this program).
AWS Premium Support Information	The primary web page for information about AWS Premium Support, a one-on-one, fast-response support channel to help you build and run applications on AWS Infrastructure Services.
Contact Us	A central contact point for inquiries concerning AWS billing, your AWS account, events, abuse, etc.
Conditions of Use	Detailed information about copyright and trademark usage at Amazon.com and other topics.

IAM Best Practices

Topics

- [Lock away your AWS root account credentials \(p. 13\)](#)
- [Create individual users \(p. 14\)](#)
- [Use groups to assign permissions to IAM users \(p. 14\)](#)
- [Grant least privilege \(p. 14\)](#)
- [Configure a strong password policy for your users \(p. 15\)](#)
- [Enable MFA for privileged users \(p. 16\)](#)
- [Use roles for applications that run on EC2 instances \(p. 16\)](#)
- [Delegate by using roles instead of by sharing credentials \(p. 16\)](#)
- [Rotate credentials regularly \(p. 16\)](#)
- [Use policy conditions for extra security \(p. 17\)](#)
- [Video presentation about IAM best practices \(p. 17\)](#)

Any request you make for an AWS resource requires credentials so that AWS can make sure you have permission to access the resource. When you created your AWS account, you got an access key ID and a secret access key. Our recommendation: don't use those keys.

This article presents a list of suggestions for using the AWS Identity and Access Management (IAM) service instead of your root AWS account credentials to help secure access to your AWS account.

Lock away your AWS root account credentials

The access key ID and secret access key for your AWS account give you access to all your resources, including your billing information. Therefore, protect your AWS account credentials like you would your credit card numbers or any other secret in these ways:

- Delete your AWS root account access key ID and secret access key, or at least rotate (change) the credentials regularly. For information about managing your AWS account password, see [Changing Your AWS Account Password \(p. 96\)](#).
- Never share your AWS root account credentials with anyone. The remaining sections of this document discuss various ways to avoid having to share your root credentials with other users and to avoid having to embed them in an application.
- Use a strong password to help protect account-level access to the [AWS Management Console](#).

- Enable AWS multi-factor authentication (MFA) on your AWS root account. For more information, see [Using Multi-Factor Authentication \(MFA\) Devices with AWS \(p. 106\)](#).

Create individual users

Don't use your AWS root account credentials to access AWS, and don't give your credentials to anyone else. Instead, create individual users (using the [IAM console](#), the APIs, or the command line interface) for anyone who needs access to your AWS account. Create an IAM user for yourself as well, give that IAM user administrative privileges, and use that IAM user for all your work.

By creating individual IAM users for people accessing your account, you can give each IAM user a unique set of security credentials. You can also grant different permissions to each IAM user. If necessary, you can change or revoke an IAM user's permissions any time. (If you give out your AWS root credentials, it can be difficult to revoke them.)

Note

Before you set permissions for individual IAM users, though, see the next point about groups.

For more information, see [Working with Users and Groups \(p. 72\)](#).

Use groups to assign permissions to IAM users

Instead of defining permissions for individual IAM users, it's usually more convenient to create groups that relate to job functions (Admins, Developers, Accounting, etc.), define the relevant permissions for each group, and then assign IAM users to those groups. All the users in an IAM group share the same permissions. That way, you can make changes for everyone in a group in just one place. As people move around in your company, you can simply change what IAM group their IAM user belongs to.

For more information, see the following:

- [What Is IAM? \(p. 18\)](#)
- [Setting Up an Administrators Group \(p. 67\)](#)
- [Adding Users to and Removing Users from a Group \(p. 89\)](#)

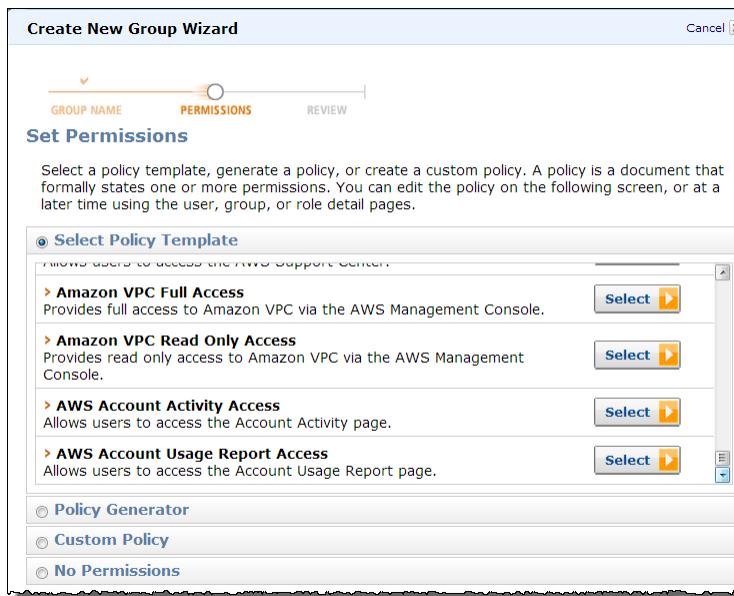
Grant least privilege

When you create IAM policies, follow the standard security advice of granting *least privilege*—that is, granting only the permissions required to perform a task. Determine what users need to do and then craft policies for them that let the users perform *only* those tasks. Similarly, create policies for individual resources (such as S3 buckets) that identify precisely who is allowed to access the resource, and allow only the minimal permissions for those users. For example, perhaps developers should be allowed to write to an S3 bucket, but testers only need to read from it.

It's more secure to start with a minimum set of permissions and grant additional permissions as necessary, rather than starting with permissions that are too lenient and then trying to tighten them later.

Defining the right set of permissions requires some research to determine what is required for the specific task, what actions a particular service supports, and what permissions are required in order to perform those actions.

A good way to start is to use the built-in policy templates in the console. These templates include predefined permissions for common use cases (administrator, power user, etc.) in individual services.



You can also create custom policies that set permissions precisely. When you do, you need to be familiar with how Allow and Deny work in policies, and how policies are evaluated when more than one policy is in force during a request for a resource. (In cases where the right combination of permissions is complex, it can be tempting to loosen up the permissions—even to grant "all access" privileges—to make sure users have access to resources. But we do not recommend this approach; as noted, standard security advice is to grant least privilege.) You'll also need to test thoroughly to make sure that users can do their work and that policies are working as you intend.

For more information, see the following:

- [Key Concepts \(p. 33\)](#)
- [Evaluation Logic \(p. 37\)](#)
- Policy topics for individual services, which provide examples of how to write policies for service-specific resources. Examples:
 - [Controlling Access to Amazon DynamoDB Resources](#) in the *Amazon DynamoDB Developer Guide*
 - [Using IAM Policies](#) in the *Amazon Simple Storage Service Developer Guide*
 - [Access Control List \(ACL\) Overview](#) in the *Amazon Simple Storage Service Developer Guide*

Configure a strong password policy for your users

If you allow users to change their own passwords, make sure that they create strong passwords. In the [Password Policy](#) section of the IAM console, you can choose options for password policy, such as the password's minimum length, whether it requires a non-alphabetic character, and so on.

For more information, see [Managing an IAM Password Policy \(p. 96\)](#).

Enable MFA for privileged users

For extra security, enable multi-factor authentication (MFA) for privileged IAM users (users who are allowed access to sensitive resources). With MFA, users have a device that generates a unique authentication code (a one-time password, or OTP) and users must provide both their normal credentials (like their user name and password) and the OTP. The MFA device can either be a special piece of hardware, or it can be a virtual device, such as an app that runs on a smartphone.

For more information, see [Using Multi-Factor Authentication \(MFA\) Devices with AWS \(p. 106\)](#).

Use roles for applications that run on EC2 instances

Applications that run on an EC2 instance need credentials in order to access other AWS services. To provide credentials to the application in a secure way, use IAM *roles*. A role is an entity that has its own set of permissions, but that isn't a user or group. Roles also don't have their own permanent set of credentials the way IAM users do. Instead, a role is *assumed* by other entities. Credentials are then either associated with the assuming identity, or IAM dynamically provides temporary credentials (in the case of EC2).

When you launch an EC2 instance, you can specify a role for the instance as a launch parameter. Applications that run on the EC2 instance can use the role's credentials when they access AWS resources. The role's permissions determine what the application is allowed to do.

For more information, see [Granting Applications that Run on Amazon EC2 Instances Access to AWS Resources \(p. 139\)](#).

Delegate by using roles instead of by sharing credentials

You might need to allow users from another AWS account to access resources in your AWS account. If so, don't share security credentials, such as access keys, between accounts. Instead, use IAM roles. You can define a role that specifies what permissions the IAM users in the other account are allowed, and from which AWS accounts IAM users are allowed to assume the role.

For more information, see [Cross-Account Access Using Resource-Based Policies \(p. 168\)](#).

Rotate credentials regularly

Change your own passwords and access keys regularly, and make sure that all IAM users do as well. That way, if a password is compromised without your knowledge, you limit how long the password can be used to access your resources.

To make it easier to rotate credentials, let your users manage their own passwords.

For more information, see [Rotating Credentials \(p. 136\)](#).

Use policy conditions for extra security

To the extent that it's practical, define the conditions under which the policy allows access to a resource. For example, you can write conditions to specify a range of allowable IP addresses for a request, or you can specify that a request is allowed only within a specified date range or time range. You can also set conditions that require the use of SSL or MFA. For example, you can require that a user has authenticated with an MFA device in order to be allowed to terminate an EC2 instance. The more explicitly you can define when resources are available (and otherwise unavailable), the safer your resources will be.

For more information, see [Condition \(p. 49\)](#).

Video presentation about IAM best practices

The following video includes a conference presentation that covers these best practices and shows additional details about how to work with the features discussed here.

[AWS re:Invent SEC 303: Top 10 AWS Identity and Access Management \(IAM\) Best Practices](#)

Introduction to IAM

Topics

- [What Is IAM? \(p. 18\)](#)
- [IAM Concepts \(p. 21\)](#)
- [Business Use Cases \(p. 27\)](#)
- [Permissions and Policies \(p. 30\)](#)

This section provides an introduction to IAM.

What Is IAM?

Topics

- [Features of IAM \(p. 19\)](#)
- [Supported AWS Products \(p. 19\)](#)
- [Migration to IAM \(p. 19\)](#)
- [IAM and Consolidated Billing \(p. 20\)](#)

AWS Identity and Access Management is a web service that enables Amazon Web Services (AWS) customers to manage users and user permissions in AWS. The service is targeted at organizations with multiple users or systems that use AWS products such as Amazon EC2, Amazon SimpleDB, and the AWS Management Console. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users can access.

Without IAM, organizations with multiple users and systems must either create multiple AWS accounts, each with its own billing and subscriptions to AWS products, or employees must all share the security credentials of a single AWS account. Also, without IAM, you have no control over the tasks a particular user or system can do and what AWS resources they might use.

IAM addresses this issue by enabling organizations to create multiple *users* (each user is a person, system, or application) who can use AWS products, each with individual security credentials, all controlled by and billed to a single AWS account. With IAM, each user is allowed to do only what they *need* to do as part of the user's job.

Features of IAM

IAM includes the following features:

- **Central control of users and security credentials**—You can control creation, rotation, and revocation of each user's AWS security credentials (such as access keys)
- **Central control of user access**—You can control what data in the AWS system users can access and how they access it
- **Shared AWS resources**—Users can share data for collaborative projects
- **Permissions based on organizational groups**—You can restrict users' AWS access based on their job duties (for example, admin, developer, etc.) or departments. When users move inside the organization, you can easily update their AWS access to reflect the change in their role
- **Central control of AWS resources**—Your organization maintains central control of the AWS data the users create, with no breaks in continuity or lost data as users move around within or leave the organization
- **Control over resource creation**—You can help make sure that users create AWS data only in sanctioned places
- **Networking controls**—You can help make sure that users can access AWS resources only from within the organization's corporate network, using SSL
- **Single AWS bill**—Your organization's AWS account gets a single AWS bill for all your users' AWS activity

Supported AWS Products

IAM is integrated with many AWS products. For information about which products are integrated with IAM, see [Integrating with Other AWS Products \(p. 195\)](#). As AWS products integrate with IAM, they will be added to the list on that page.

Important

If you're an existing AWS account owner, you should know that the APIs for these products do not change because of their integration with IAM. Products that integrate with IAM have no new API actions related to access control.

If the users in your AWS account make API calls to any AWS products other than those that are integrated with IAM, they'll get an error. However, the AWS account itself (the umbrella entity above all the users) can call any AWS product the AWS account is signed up for.

Regarding IAM and Amazon DevPay:

- Users in your AWS account can't create or sign up for products that use Amazon DevPay; only the AWS account can
- If your AWS account purchases an Amazon EC2 paid Amazon Machine Image (AMI), the AWS account's users can launch and use instances of the AMI (assuming they have an IAM policy giving them permission to use the necessary Amazon EC2 actions)
- If your AWS account purchases an Amazon S3 product that uses Amazon DevPay, the AWS account's users can't use the product; only the AWS account can

Migration to IAM

If your organization already uses AWS, migrating to IAM can be easy or potentially more challenging, depending on how your organization currently allocates its AWS resources. Here are the three scenarios.

1. **Your organization has just a single AWS account.** In this case, you can easily migrate to using IAM, because all the organization's AWS resources are already together under a single AWS account.
2. **Your organization has multiple AWS accounts, with each AWS account belonging to a division in the organization.** If these divisions don't need to share resources or users, then migrating is easy. Each division can keep its own AWS account and use IAM separately from the other divisions. You could also use Consolidated Billing, which would allow your organization to get a single bill across the AWS accounts (see [IAM and Consolidated Billing \(p. 20\)](#)).
3. **Your organization has multiple AWS accounts that don't represent logical boundaries between divisions.** If you need the AWS accounts to share their resources and have common users, migrating to IAM will be more of a challenge. You will need to move the resources that need to be shared so they're under the ownership of a single AWS account. However, there's no automatic way to transfer the AWS resources from one AWS account to another. You need to create those resources again under the single AWS account.

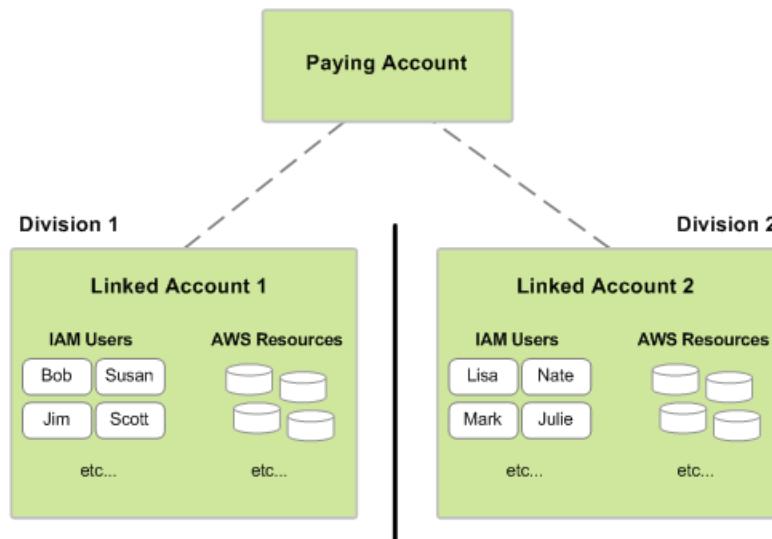
No Change to Basic AWS Account Functions

There's no change to how an AWS account functions in terms of its login/password, security credentials, payment method, AWS account activity page, usage report, and so on. At this time, the AWS account activity page does not show a breakdown by user.

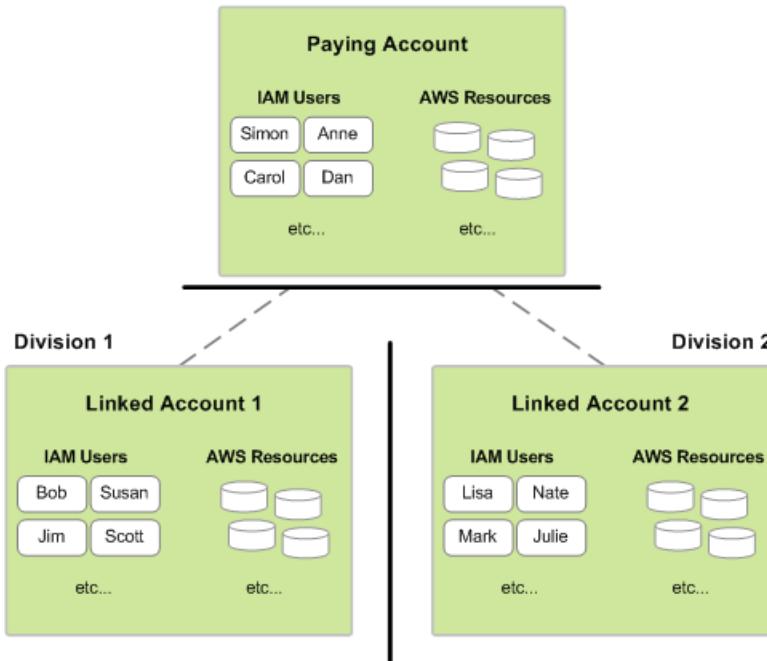
IAM and Consolidated Billing

AWS offers a billing feature called *Consolidated Billing* that is complementary to IAM. Consolidated Billing lets you receive a single bill for multiple AWS accounts. (For more information, go to the [AWS Consolidated Billing Guide](#).) Compare this to IAM, which gives you a single bill across all the *users* in an AWS account.

Your organization could use both Consolidated Billing and IAM together. You might do this if your organization has multiple large divisions, and you want to isolate the users and AWS resources in each division from the other divisions. You could have a separate AWS account for each division, and use IAM in each division to create users and control their access to the division's AWS resources. You could then use Consolidated Billing to get a single bill across all the AWS accounts. The following diagram illustrates the concept.



With Consolidated Billing, one AWS account becomes the *paying account*, and pays for its own charges plus the charges of any *linked AWS accounts*. Each linked AWS account doesn't need to maintain a payment method with AWS, only the paying account does. Each month, AWS charges the paying account only. The paying account still functions like a normal AWS account; it could have its own users and AWS resources. Just as with the other AWS accounts, the users and resources in the paying account are isolated from the users and resources in the divisions' AWS accounts. The following diagram shows the paying account with its own users and AWS resources.



IAM Concepts

Topics

- Concepts Related to AWS Account Entities (p. 21)
- Concepts Related to Permissions (p. 26)

To help you understand IAM, we recommend you think about it in two parts:

- The basic *entities* that comprise your AWS account (such as users and groups)
- The *permissions* you grant to the entities in your AWS account

Concepts Related to AWS Account Entities

This section describes the basic entities in your AWS account and how they fit together. They're presented in a logical order, with the first items you need to know at the start of the list.

AWS Account

Tip

If you're already an AWS customer, you're probably already familiar with AWS accounts and their characteristics. With IAM, an AWS account remains essentially the same, except the account can now have *users* under its umbrella.

An *AWS account* is the first entity you create when initiating a relationship with AWS. The AWS account centrally controls all the resources created under its umbrella and pays for all AWS activity for those resources.

Any permissions created for users or groups within the AWS account don't apply to the AWS account itself. The AWS account has permission to do anything and everything with all the AWS account resources. In this way, the AWS account is similar in concept to the UNIX *root* or *superuser*.

The AWS account has its own set of security credentials that can be used to interact with AWS programmatically. The AWS account also has an email address login and password, giving it access to the AWS Management Console and to the secure pages on the AWS website. The secure pages display the AWS account's security credentials and payment method, among other data.

Note

To help keep that sensitive information secure, we recommend that you use Multi-Factor Authentication (MFA) with your AWS account's email login and password. For detailed information about AWS MFA, see the [AWS Multi-Factor Authentication FAQs](#). For information about using MFA with IAM, see [Using Multi-Factor Authentication \(MFA\) Devices with AWS \(p. 106\)](#).

Ideally, someone in your organization uses the AWS account credentials to set up an administrator group for the AWS account. The group has admin users responsible for all subsequent management of users and permissions. After setting up the admins group, your organization would no longer use the AWS account credentials, and only users (with their own credentials) would interact with AWS from then on.

Note

If your users need to interact with an AWS product that doesn't integrate with IAM, then they must use the AWS account's credentials. For a list of AWS products that integrate with IAM, see [Supported AWS Products \(p. 19\)](#).

Before using IAM, your organization might have had a single AWS account for all the employees, or the organization might have given each division or each employee a separate AWS account. With IAM, your organization will probably have one or only a few AWS accounts, but many users. The organization won't need to have an individual AWS account per employee. If your organization already has an AWS account, you don't need to get a new or different one for your organization to use IAM. In fact, we recommend you keep the existing one for your organization when you begin to use IAM, because any AWS resources that your organization has already created can't be moved to a different AWS account.

User

A *user* is an individual, system, or application that interacts with AWS programmatically. Each user has a unique name within the AWS account, and a set of *security credentials* not shared with other users. These credentials are separate from the AWS account's security credentials. Each user is associated with one and only one AWS account. Users don't need to sign up for AWS products (such as Amazon EC2), nor do they need to have a payment method on file with AWS. The AWS account is the entity that is subscribed to AWS products and pays for the users' activity with AWS.

Users are the primary actors who interact with AWS. They can use the AWS Management Console, an API, or a command line interface (if the AWS product provides one). Authentication happens at the user level by means of the user's login and password (for the AWS Management Console), or the user's security credentials (for the API or command line interface). Users access the AWS Management Console via a link to your AWS account sign-in page that you provide to them. (For more information on your AWS account sign-in page, see [The AWS Management Console Sign-in Page \(p. 190\)](#).)

By default, newly created users have no password, no security credentials, and no permissions to do anything. The AWS account owner (or any user with the authority, such as an administrator) must provide each user with what they need.

There's a limit to the number of users you can have. For more information, see [Limitations on IAM Entities \(p. 64\)](#).

For more information about creating users, see [Adding a New User to Your AWS Account \(p. 72\)](#).

Group

A *group* is a collection of users. Groups don't directly interact with AWS; only users do. The main reason to create groups is to collectively assign permissions to the users so they can do their jobs. For example, you could have a group called *Admins* and give that group the types of permissions admins typically need.

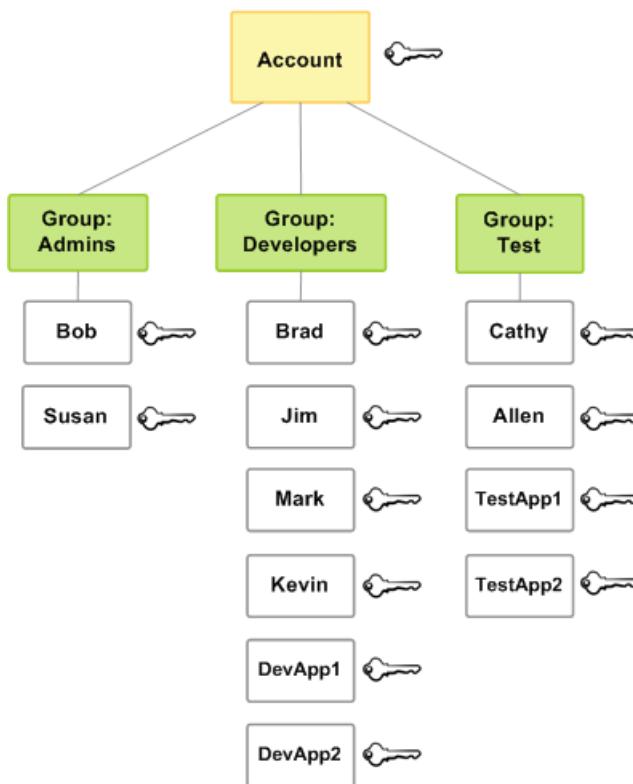
Following are some important characteristics of groups:

- A group can contain many users, and a user can belong to multiple groups.
- Groups can't be nested; they can contain only users.
- There's no default group that automatically includes all users in the AWS account. If you'd like to have a group like that, you need to create it yourself and assign each new user to it.
- There's a limit to the number of groups you can have, and a limit to how many groups a user can be in. For more information, see [Limitations on IAM Entities \(p. 64\)](#).

For more information about creating and managing groups, see [Working with Users and Groups \(p. 72\)](#).

Example

The following diagram shows a simple example of a small company. The company owner uses the AWS account credentials to create an *Admins* group that can create and manage the users as the company grows. The *Admins* group establishes a *Development* group and a *Test* group. Each of these two groups consists of users (humans and applications) that interact with AWS (Jim, Brad, DevApp1, and so on). Each user has an individual set of security credentials. In this example, each user belongs to a single group. However, users can belong to multiple groups.



Role

A *role* is an entity that has a set of permissions, and that another entity assumes to make calls to access your AWS resources. The entity who assumes the role uses temporary security credentials to make calls. For more information about roles and how they differ from users or groups, see [Delegating API Access by Using Roles \(p. 138\)](#).

Security Credentials

Any person or application making API calls to AWS needs *security credentials*. The entity making the call can be the AWS account, a user, a role. AWS uses these credentials to identify who is making the call and whether to allow the requested access.

The basic credentials consist of a *Secret Access Key*, and a corresponding *Access Key ID*. The requester signs each request with the Secret Access Key and includes the Access Key ID in the request. How this works for a given AWS product is covered in the product's technical documentation.

AWS Account Credentials

As mentioned elsewhere, when you create an AWS account, AWS gives the AWS account its own Secret Access Key and Access Key ID by default. The AWS account can make API calls to AWS with them. We expect that you won't use those credentials on a regular basis, but will use them only to initially set up an administrators group for your organization. We recommend that all further API interaction between your AWS account and your AWS resources be at the user level (for example, using users' security credentials).

To get your AWS account's Secret Access Key and Access Key ID, go to <http://aws.amazon.com/security-credentials> and sign in with your AWS account's login and password.

Note

To help control who has access to the AWS account's Secret Access Key, we recommend that you use Multi-Factor Authentication (MFA) with your AWS account's login and password. For detailed information about AWS MFA, see the [AWS Multi-Factor Authentication FAQs](#). For information about using MFA with IAM, see [Using Multi-Factor Authentication \(MFA\) Devices with AWS \(p. 106\)](#).

User Credentials

By default, a user has no security credentials. You create security credentials for your users as needed. The type of credentials a user needs depends on how the user will access AWS.

Secret Access Keys and Access Key IDs

To make API calls or to work with the command line interface, the user needs a Secret Access Key and Access Key ID. The IAM API and command line interface provide actions that create these for a user. You can give your users permission to create and manage their own credentials if you like, or you can have an administrators group in your organization handle this. For more information about creating keys for a user, see [Adding a New User to Your AWS Account \(p. 72\)](#).

X.509 Certificates

Another type of credential a user might have is an *X.509 certificate* (referred to here as a *signing certificate*) and corresponding *certificate ID*. Some AWS products use this instead of a Secret Access Key for access to certain interfaces. For example, Amazon EC2 uses a Secret Access Key for access to its Query interface, but it uses a signing certificate for access to its SOAP interface and command line tool interface.

Although you can use IAM to create an access key, you can't use IAM to create a signing certificate. However, you can use free third-party tools such as OpenSSL to create the certificate. (For information about OpenSSL, go to <http://www.openssl.org/>.) After you have the signing certificate, you must upload it to IAM; the user needs to keep the corresponding private key to use for signing requests. You can use IAM to upload the certificate. For more information about using signing certificates, see [Managing User Keys and Certificates \(p. 124\)](#).

Important

For security purposes, we recommend that you rotate your users' credentials on a regular basis. A user can have multiple access keys or signing certificates at a given time for this purpose. For more information, see [Rotating Credentials \(p. 136\)](#).

Passwords

A password enables a user to sign in to the [AWS Management Console](#). You need to create a password for each user who needs to use the AWS Management Console. For more information about managing passwords, see [Managing Passwords \(p. 95\)](#).

Multi-Factor Authentication for Users

AWS offers Multi-Factor Authentication (MFA). MFA is an optional feature that requires the user to possess an authentication device and provide a randomly generated, six-digit code from the device in addition to the standard security credentials. For detailed information about AWS MFA, see the [AWS Multi-Factor Authentication FAQs](#).

Your users can use MFA in these situations:

- When logging in to the AWS Management Console
- When requesting secure API actions

An example of a secure API action is the Amazon S3 action for deleting versioned objects. You can configure Amazon S3 versioning to require the requester to provide an MFA code in order to delete a versioned object. If you do, and you give a particular user permission to delete versioned objects, the user must provide an MFA code in the API call.

You can use IAM to enable an MFA device for a user. You can also use IAM to resynchronize the device, delete it, and list (discover) a user's MFA device. For information about using MFA devices, see [Using Multi-Factor Authentication \(MFA\) Devices with AWS \(p. 106\)](#).

Concepts Related to Permissions

This section summarizes the basic concepts related to permissions. For a more detailed discussion of these concepts, see [Permissions and Policies \(p. 30\)](#).

Resource

A *resource* is an entity in an AWS service that a user can interact with, such as an Amazon S3 bucket or object, an Amazon SQS queue, and so on.

Resources typically have a friendly name (such as `example_bucket`), and then an *Amazon Resource Name (ARN)*, which uses a standardized format and uniquely identifies the resource in AWS. For more information about ARNs, see [Identifiers for IAM Entities \(p. 60\)](#).

Permission

A *permission* is the concept of allowing (or disallowing) an entity such as a user, group, or role some type of access to one or more resources. For example, Bob has permission to read and write objects to a particular Amazon S3 bucket named `example_bucket`.

There are two general types of permissions you might use within AWS: *user-based* and *resource-based*. The easiest way to understand the difference is to think of the question each type of permission answers:

- **User-based**—What does a particular entity have access to?

For example, Bob might have permission to use the Amazon EC2 `RunInstance` action with any of the AWS account's resources, permission to use any Amazon SimpleDB actions with any of the AWS account's domains that start with the string `bob`, and permission to use any of the IAM actions with his own security credentials.

- **Resource-based**—Who has access to a particular resource?

For example, the `example_bucket` resource might have a permission that states that Bob, Susan, and DevApp1 have read, write, and list permission.

For more information about permissions, see [Overview of Permissions \(p. 30\)](#).

Policy

A *policy* is a document that provides a formal statement of one or more permissions. With IAM, you can assign a policy to an entity, permissions stated in the policy. You can assign multiple policies to an entity. If you want to assign the same policy to multiple users, we recommend you put the users in a group and assign the policy to the group.

For more information about policies, see [Overview of Policies \(p. 32\)](#) and [Managing IAM Policies \(p. 173\)](#).

Other Permissions Systems

IAM has its own permissions system that you use to give entities the ability to do tasks with IAM resources in the AWS account.

IAM also integrates with individual AWS products so you can control an entity's access to those products' resources. Some of the AWS products that IAM integrates with have their own resource-based permissions systems for giving access to specific resources. How IAM works with the different permissions systems is covered elsewhere in this guide. For more information, see the product-specific sections described in [Integrating with Other AWS Products \(p. 195\)](#).

Business Use Cases

Topics

- [Initial Setup of Example Corp. \(p. 27\)](#)
- [Use Case for IAM with Amazon EC2 \(p. 28\)](#)
- [Use Case for IAM with Amazon S3 \(p. 28\)](#)

This section describes a simple business use case for IAM to help you understand basic ways you might implement the service to control the AWS access your users have. The use case is described at a high level, without the mechanics of how you'd use the IAM API to achieve the results you want.

The use case centers on a fictional company called Example Corp. After setting up an AWS account for Example Corp., we show two typical examples of how the company might use IAM—first, with Amazon Elastic Compute Cloud (Amazon EC2), and then with Amazon Simple Storage Service (Amazon S3).

For more information about using IAM with other AWS products, including how to implement individual APIs, see [Integrating with Other AWS Products \(p. 195\)](#).

Initial Setup of Example Corp.

Joe is the founder of Example Corp. Upon starting the company, he creates his own AWS account and he uses AWS products by himself. Then he hires employees to work as developers, admins, testers, managers, and system administrators.

Joe uses the IAM API with the AWS account's security credentials to create a user for himself called Joe, and a group called *Admins*. He gives the Admins group the permissions it needs to administer users, groups, and permissions for the AWS account, and he gives the Admins group permissions to perform all actions on all the AWS account's resources (for example, root privileges).

Joe then assigns himself (as Joe the user) to the Admins group. At this point, he stops using the AWS account's credentials to interact with AWS, and instead he begins using his user credentials.

Joe also creates a group called *AllUsers* so he can easily apply any account-wide permissions to all users in the AWS account. He adds himself to the group. He then creates a group called *Developers*, a group called *Testers*, a group called *Managers*, and a group called *SysAdmins*. He creates users for each of his employees, and puts the users in their respective groups. He also adds them all to the AllUsers group.

For information about how to set up an Admins group, see [Setting Up an Administrators Group \(p. 67\)](#). For information about creating users, see [Adding a New User to Your AWS Account \(p. 72\)](#).

Use Case for IAM with Amazon EC2

This use case illustrates how Example Corp. uses IAM with Amazon EC2. To understand this part of the use case, you need to have a basic understanding of Amazon EC2. For more information about Amazon EC2, go to the [Amazon Elastic Compute Cloud User Guide](#).

Amazon EC2 Permissions for the Groups

To provide "perimeter" control, Joe adds a policy to the AllUsers group that denies any AWS request from a user if the originating IP address is outside the Example Corp.'s corporate network.

At Example Corp., different groups require different permissions:

- **System Administrators** - Need permission to create and manage AMIs, instances, snapshots, volumes, security groups, and so on. Joe adds a policy to the SysAdmins group that gives members of the group permission to use all the EC2 actions.
- **Developers** - Need the ability to work with instances only. Joe therefore adds a policy to the Developers group that allows developers to call `DescribeInstances`, `RunInstances`, `StopInstances`, `StartInstances`, and `TerminateInstances`. Amazon EC2 currently doesn't support IAM policies that restrict access to a particular AMI, volume, instance, etc., so Example Corp. can control only which Amazon EC2 APIs the developers can call.

Note

Amazon EC2 uses SSH keys, Windows passwords, and security groups to control who has access to specific Amazon EC2 instances. There's no method in the IAM system to allow or deny access to a specific instance.

- **Managers** - Should not be able to perform any EC2 actions except listing the Amazon EC2 resources currently available. Therefore, Joe adds a policy to the Managers group that only lets them call EC2 "Describe" APIs.

For examples of what these policies might look like, see [Example Policies for IAM Entities \(p. 179\)](#) and [Using AWS Identity and Access Management](#) in the [Amazon Elastic Compute Cloud User Guide](#).

User's Role Change

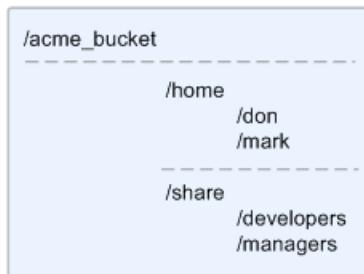
At some point, one of the developers, Don, changes roles and becomes a manager. Joe moves Don from the Developers group to the Managers group. Now that he's in the Managers group, Don's ability to interact with Amazon EC2 instances is limited. He can't launch or start instances. He also can't stop or terminate existing instances, even if he was the user who launched or started the instance. He can list only the instances that Example Corp. users have launched.

Use Case for IAM with Amazon S3

This use case illustrates how Example Corp. uses IAM with Amazon S3. Joe has created an Amazon S3 bucket for the company called `example_bucket`.

Creation of Other Users and Groups

As employees, Don and Mark each need to be able to create their own data in the company's bucket, as well as read and write shared data that all developers will work on. To enable this, Joe logically arranges the data in `example_bucket` using an Amazon S3 key prefix scheme as shown in the following figure.



Joe divides the master `/example_bucket` into a set of home directories for each employee, and a shared area for groups of developers and managers.

Now Joe creates a set of policies to assign permissions to the users and groups:

- **Home directory access for Don:** Joe assigns a policy to Don that lets him read, write, and list any objects with the Amazon S3 key prefix `/example_bucket/home/don/`
- **Home directory access for Mark:** Joe assigns a policy to Mark that lets him read, write, and list any objects with the Amazon S3 key prefix `/example_bucket/home/mark/`
- **Shared directory access for the Developers group:** Joe assigns a policy to the group that lets developers read, write, and list any objects in `/example_bucket/share/developers/`
- **Shared directory access for the Managers group:** Joe assigns a policy to the group that lets managers read, write, and list objects in `/example_bucket/share/managers/`

Note

Amazon S3 doesn't automatically give a user who creates a bucket or object permission to perform other actions on that bucket or object. Therefore, in your IAM policies, you must explicitly give users permission to use the Amazon S3 resources they create.

The preceding set of policies clearly defines the actions and resources available IAM bucket policies or bucket Access Control Lists (ACLs) when anyone in the company attempts to work on data in the corporate space. For examples of what these policies might look like, see [Access Control](#) in the *Amazon Simple Storage Service Developer Guide*. For information on how policies are evaluated at run time, see [Evaluation Logic](#).

User's Role Change

At some point, one of the developers, Don, changes roles and becomes a manager. We'll assume he no longer needs access to the documents in the `share/developers` directory. Joe, as an admin, moves Don to the Managers group and out of the Developers group. With just that simple reassignment, Don automatically gets all permissions granted to the Managers group, but can no longer access data in the `share/developers` directory.

Integration with a Third-Party Business

Organizations often work with partner companies, consultants, and contractors. Example Corp. has a partner called the Widget Company, and a Widget Company employee named Nate needs to put data into a bucket for Example Corp.'s use. Joe creates a group called `WidgetCo` and a user named `Nate` and adds `Nate` to the `WidgetCo` group. Joe also creates a special bucket called `example_partner_bucket` for `Nate` to use.

Joe updates existing policies or adds new ones to accommodate the partner Widget Company. For example, Joe can create a new policy that denies members of the `WidgetCo` group the ability to use any

actions other than write. This policy would be necessary only if there's a broad policy that gives all users access to a wide set of Amazon S3 actions.

Permissions and Policies

Topics

- [Overview of Permissions \(p. 30\)](#)
- [Overview of Policies \(p. 32\)](#)
- [The Access Policy Language \(p. 33\)](#)

This section describes permissions, policies, and the access policy language you use to write policies.

Overview of Permissions

Topics

- [Types of Permissions \(p. 30\)](#)
- [No Default Permissions for Resource Creators \(p. 31\)](#)
- [User Permissions Across AWS Accounts \(p. 32\)](#)
- [Permissions Across Services \(p. 32\)](#)

A *permission* is a general term we use to mean the ability to perform an action against a resource. For example, you might have permission to read but not write data in an Amazon SimpleDB domain.

Types of Permissions

There are two general types of permissions we talk about in this guide: *user-based* and *resource-based*. With IAM, you work entirely with user-based permissions; whereas with Amazon Simple Storage Service (Amazon S3) and other products, you work with resource-based permissions only. The permissions answer different questions, which are outlined in the following table.

Question It Answers	Permission Type	Example
What does a particular entity (user, group, or role) have access to?	User-based	IAM policy for the user Bob, stating that he has permission to use the Amazon Elastic Compute Cloud (Amazon EC2) RunInstances action with any of the AWS account's resources, permission to use any Amazon SimpleDB actions with any of the AWS account's domains that start with the string bob, and permission to use any of the IAM actions with his own security credentials. The policy resides in the IAM system with Bob.
Who has access to a particular resource?	Resource-based	Amazon S3 access control list (ACL) on a specific bucket, stating that Bob, Susan, and DevApp1 have read, write, and list permission to that bucket. The ACL resides in the Amazon S3 system with the bucket.

The following figure illustrates the difference between the two types of permissions. Each user-based permission focuses on a specific entity; each resource-based permission focuses on a specific resource.



You can have both resource-based permissions and user-based permissions that apply to the same resource or user, even though the permissions are located in separate places (with the resource in the AWS service versus with the user in the IAM system). Both types of permissions are taken into account when AWS evaluates whether to grant a particular user access to the resource.

Every user you create in the IAM system starts with no permissions. In other words, by default, users can do nothing, not even view their own access keys. To give a user permission to do something, you can either add the permission to the user (that is, attach a policy to the user), or add the user to a group that has the desired permission. We expect that you'll typically do the latter for most permissions you want to give to a user.

For example, one type of permission you might add to a user is the ability for the user to list his or her own access keys. You could expand on that permission and also let each user create, update, and delete their own keys (for example, if you're letting users rotate their own credentials).

When you give permissions to a group, all users in that group get those permissions. For example, you can give the Admins group permission to perform any of the IAM actions on any of the AWS account resources. Another example: You can give the Managers group permission to list all the Amazon SimpleDB domains, and describe the AWS account's Amazon EC2 instances.

You can grant permissions that cover IAM resources and actions, and permissions that cover other AWS products such as Amazon SimpleDB, Amazon EC2, and so on. For more information, see [Integrating with Other AWS Products \(p. 195\)](#).

No Default Permissions for Resource Creators

The AWS account (that is, someone using the AWS account's security credentials) always has the ability to use any of the API actions for any AWS products the AWS account is signed up to use. This means that when the AWS account creates a resource such as an Amazon SimpleDB domain, the AWS account can then automatically do anything with it (and no one can revoke that access). This isn't true of the users

within the AWS account. Just because the user Bob is the creator of a particular domain, he doesn't automatically get full control over that domain. He can only do whatever he's been given permission to do with the domain. His permission can be revoked at any time by the AWS account or any user in the AWS account who has the permission to revoke Bob's permissions.

User Permissions Across AWS Accounts

In most cases, a user belongs to an AWS account and works within the context of that account. However, in some cases, IAM also enables users to work *across* AWS accounts. You can enable cross-account API accesss by using IAM roles so that IAM users from external AWS accounts can access resources in your organization's AWS account. For more information, see [Enabling Cross-Account API Access \(p. 142\)](#).

Although IAM provides a way to share resources between some AWS accounts and resources, there are also product permissions systems in AWS that provide similar functionality (such as the Amazon S3 ACL system). Those systems still apply even when you're using IAM.

Permissions Across Services

If an AWS product that a user wants to use automatically makes API calls to other AWS products, a user only needs permission to use the first product. For example, Auto Scaling automatically launches Amazon EC2 instances as needed for an AWS account. A user who wants to use Auto Scaling must have permission to use it, but doesn't need permission to use Amazon EC2.

Overview of Policies

You give a permission to a user or group with a *policy*. A policy is a document that formally states one or more permissions. The distinction between a *permission* and a *policy* is important. To give a particular IAM entity a permission, you simply write a policy according to the access policy language IAM uses, then attach the policy to the entity you want it to apply to (a particular user or group in your AWS account). You don't actually specify the entity in the policy itself; the act of attaching the policy to the entity grants it the permission stated in the policy. For information about the access policy language, see [The Access Policy Language \(p. 33\)](#). For an example of how to attach a policy to an entity, see [Adding a Policy to the Group \(p. 69\)](#).

You can attach more than one policy to a particular entity. For information about the maximum number you can attach, see [Limitations on IAM Entities \(p. 64\)](#). If you have multiple permissions you want to grant an entity, you can put them in separate policies, or you can put them all in one policy. For more information about the structure of policies and how to write them, see [The Access Policy Language \(p. 33\)](#).

Each user will probably have multiple policies that apply to them (but aren't necessarily *attached* to them). For example, Bob could have policies attached to him, and other policies attached to the groups he's in. AWS evaluates all of the relevant policies at request time to determine whether Bob can perform a certain action, under what conditions, and on what resources. If there are relevant permissions that an AWS product has in its own system separate from IAM (for example, a policy in the Amazon SQS system specifying access control for the queue of interest), we also include those in the evaluation. Because of how the evaluation process is designed, there's never a conflict or uncertain result. The result is always a Boolean; access is either granted or denied. For more information about the permissions that the AWS products provide separately from IAM, see [Integrating with Other AWS Products \(p. 195\)](#). For more information about the evaluation logic we use, see [Evaluation Logic \(p. 37\)](#).

When you specify a particular resource in an IAM policy (for example, a user, an Amazon SimpleDB domain, and so on), you must use the *Amazon Resource Name (ARN)* format, which is a standardized format that AWS uses. You only need to use this format when specifying a resource *in the policy*, and not when specifying a resource in API calls. The API calls use the resource's friendly name. For example, when creating a user named Bob, you call `CreateUser` with `UserName=Bob`. For more information about the ARN format, see [Identifiers for IAM Entities \(p. 60\)](#).

Note

IAM policies control access regardless of the interface. For example, you could provide a user with a password to access the AWS Management Console, and the policies for that user (or any groups the user belongs to) would control what the user can do in the AWS Management console. Or, you could provide the user with AWS access keys for making API calls to AWS, and the policies would control what actions the user could call through a library or client that uses those access keys for authentication.

For basic example policies that cover common scenarios, see [Example Policies for IAM Entities \(p. 179\)](#), [Integrating with Other AWS Products \(p. 195\)](#), and the [AWS Policy Examples](#) page in the *AWS Sample Code & Libraries* section of the AWS website.

IAM policy templates and the policy generator are available from the IAM console of the AWS Management Console. For more information about creating policies in the console, see [Managing IAM Policies \(p. 173\)](#). Also, you can use the [AWS Policy Generator](#) online to create policies for AWS products without accessing the console.

The Access Policy Language

Topics

- [Key Concepts \(p. 33\)](#)
- [Architectural Overview \(p. 35\)](#)
- [Evaluation Logic \(p. 37\)](#)
- [How to Write a Policy \(p. 43\)](#)

To write policies for IAM, you use the *access policy language*. This section gives complete information about the access policy language, including details specific to IAM's use of the access policy language.

Key Concepts

The following sections describe the concepts you need to understand to use the access policy language. They're presented in a logical order, with the first terms you need to know at the top of the list.

Permission

A *permission* is the concept of allowing or disallowing some kind of access to a particular resource or action. Permissions essentially follow this form: "A is/isn't allowed to do B to C where D applies." For example, Jane (A) has permission to *receive messages* (B) from *Amazon SQS queue named queue12345* (C), as long as *she asks to receive them before midnight on May 30, 2010* (D). Whenever Jane sends a request to Amazon SQS to use queue12345, the service checks to see if she has permission and if the request satisfies the conditions set forth in the permission.

Statement

A *statement* is the formal description of a single permission, written in the access policy language. You always write a statement as part of a broader container document known as a *policy* (see the next concept).

Policy

A *policy* is a document (written in the access policy language) that acts as a container for one or more statements. For example, a policy could have two statements in it: one that states that Jane can use the Amazon SQS *SendMessage* action, and another that states that Jane can use all the Amazon S3 actions. As shown in the following figure, an equivalent scenario would be to have two policies, one containing the statement that Jane can use *SendMessage*, and another containing the statement that Jane can use all Amazon S3 actions.



AWS uses the information in the statements (whether they're contained in a single policy or multiple) to determine if someone requesting access to a resource should be granted that access. We often use the term *policy* interchangeably with *statement*, as they generally represent the same concept (an entity that represents a permission).

Principal

The *principal* is the person or persons who receive the permission in the policy. The principal is A in the statement "A has permission to do B to C where D applies." When you write a policy specifically to grant permission to a user or group, you don't explicitly state that user or group as the principal in the policy. Instead, you attach the policy to a user or group, and the principal is therefore implied.

Action

The *action* is the activity the principal has permission to perform. The action is B in the statement "A has permission to do B to C where D applies." Typically, the action is just the operation in the request to AWS. For example, Jane sends a request to Amazon SQS with *Action*=ReceiveMessage. You can specify one or multiple actions in a policy.

For the names of the actions you can specify in a policy, see [Integrating with Other AWS Products \(p. 195\)](#).

Resource

The *resource* is the object the principal is requesting access to. The resource is C in the statement "A has permission to do B to C where D applies." You can specify one or more resources in a policy. For information about to specify a resource in a policy, see [ARNs \(p. 61\)](#).

Conditions and Keys

The *conditions* are any restrictions or details about the permission. The condition is D in the statement "A has permission to do B to C where D applies." The part of the policy that specifies the conditions can be the most detailed and complex of all the parts. Typical conditions are related to:

- Date and time (e.g., the request must arrive before a specific day)
- IP address (e.g., the requester's IP address must be part of a particular CIDR range)

A *key* is the specific characteristic that is the basis for access restriction. For example, the date and time of request.

You use both *conditions* and *keys* together to express the restriction. The easiest way to understand how you actually implement a restriction is with an example: If you want to restrict access to before May 30, 2010, you use the condition called `DateLessThan`. You use the key called `aws:CurrentTime` and set it to the value `2010-05-30T00:00:00Z`. AWS defines the conditions and keys you can use. The AWS product of interest itself (e.g., Amazon SimpleDB) might also define service-specific keys. For more

information about conditions, see [Condition \(p. 49\)](#). For more information about the available keys, see [Available Keys \(p. 51\)](#).

Requester

The *requester* is the person who sends a request to an AWS service and asks for access to a particular resource. The requester sends a request to AWS that essentially says: "Will you allow me to do B to C where D applies?"

Evaluation

Evaluation is the process the AWS service uses to determine if an incoming request should be denied or allowed based on the applicable policies. For information about the evaluation logic, see [Evaluation Logic \(p. 37\)](#).

Effect

The *effect* is the result that you want a policy statement to return at evaluation time. You specify this value when you write the statements in a policy, and the possible values are *deny* and *allow*.

For example, you could write a policy that has a statement that *denies* all requests that come from Antarctica (*effect=deny* given that the request uses an IP address allocated to Antarctica). Alternately, you could write a policy that has a statement that *allows* all requests that *don't* come from Antarctica (*effect=allow*, given that the request doesn't come from Antarctica). Although the two statements sound like they do the same thing, in the access policy language logic, they are different. For more information, see [Evaluation Logic \(p. 37\)](#).

Although there are only two possible values you can specify for the effect (*allow* or *deny*), there can be three different results at policy evaluation time: *default deny*, *allow*, or *explicit deny*. For more information, see the following concepts and [Evaluation Logic \(p. 37\)](#).

Default Deny

A *default deny* is the default result from a policy in the absence of an *allow* or *explicit deny*. For example, if a user requests to use Amazon SQS, but the only policy that applies to the user states that the user can use Amazon SimpleDB, then that policy results in a *default deny*.

Allow

An *allow* results from a statement that has *effect=allow*, assuming any stated conditions are met. Example: Allow requests if they are received before 1:00 p.m. on April 30, 2010. An *allow* overrides all *default denies*, but never an *explicit deny*.

Explicit Deny

An *explicit deny* results from a statement that has *effect=deny*, assuming any stated conditions are met. Example: Deny all requests if they are from Antarctica. Any request that comes from Antarctica will always be denied no matter what any other policies might allow.

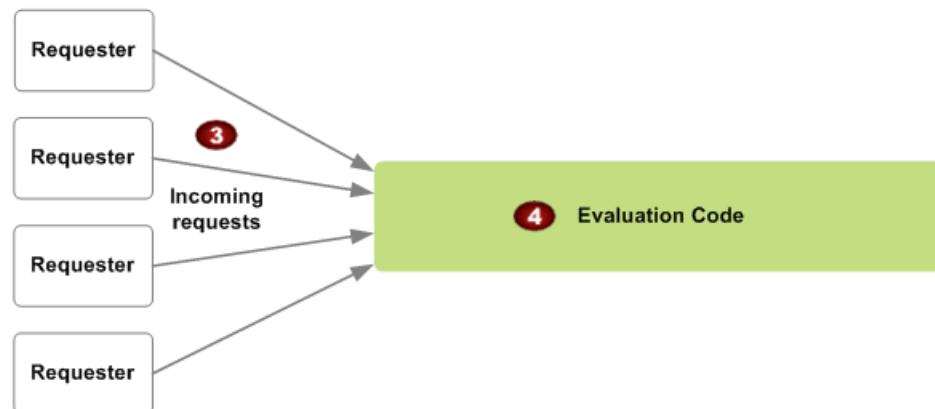
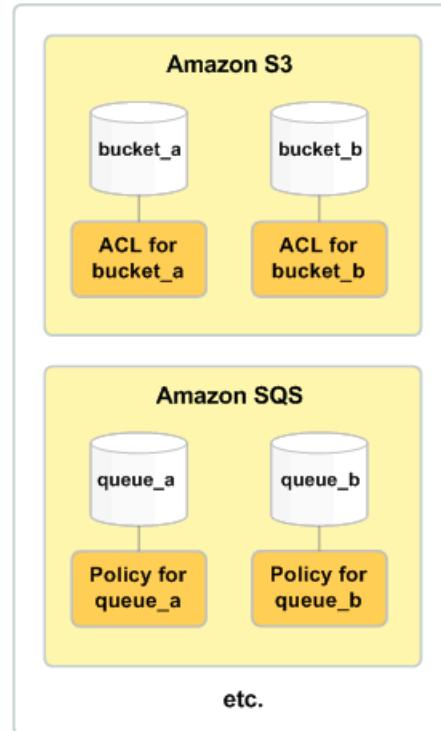
Architectural Overview

The following figure and table describe the main components that interact to provide access control for your resources.

1 User-Based Permissions for Your Account



2 Resource-Based Permissions for Your Account



- | | |
|----------|--|
| 1 | Your user-based permissions (policies attached to users or groups within the IAM system). These are created by someone in your AWS account who has permission to manage policies for your AWS account. In the diagram, each user or group has a single policy attached to it, although in practice, there could be multiple. |
| 2 | Your resource-based permissions (ACLs attached to your Amazon S3 buckets and objects, policies attached to your Amazon SQS queues, etc.). These are created by someone in your AWS account who has permission to manage ACLs, policies, etc., for the AWS resources in your AWS account. |

3	Requesters and their incoming requests to AWS to use your AWS account's resources. They might be from inside or outside your own AWS account.
4	The evaluation code. This is the set of code that evaluates incoming requests against the applicable user-based permissions and resource-based permissions and determines whether the requester is allowed access to the resource. For information about how AWS makes the decision, see Evaluation Logic (p. 37) .

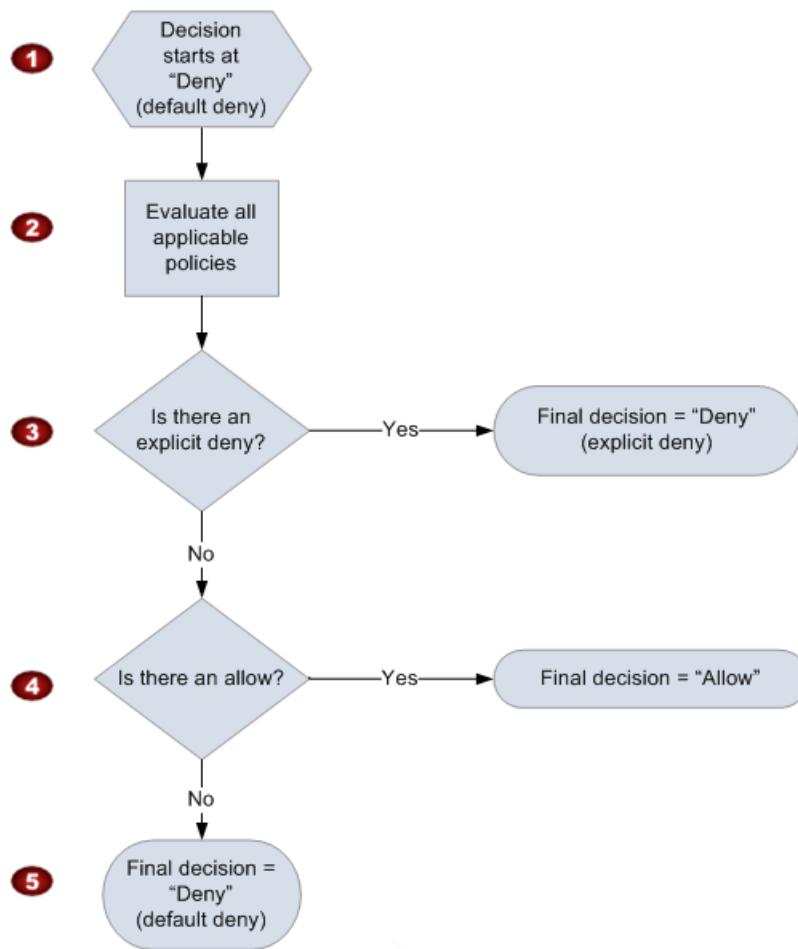
Evaluation Logic

The goal at evaluation time is to decide whether a given request should be allowed or denied. The evaluation logic follows several basic rules:

- By default, all requests receive a default deny, except for requests that use the AWS account's root security credentials
- An allow overrides any default denies
- An explicit deny overrides any allows
- The order in which the policies are evaluated is not important

The evaluation logic never results in a conflict. There is always a true/false result that either allows or denies the requested access.

The following flow chart and discussion describe in more detail how the decision is made.



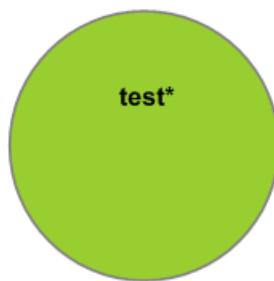
1	The decision starts with a default deny.
2	The enforcement code then evaluates all the applicable user-based policies and resource-based policies/ACLs, etc. that are applicable to the request (based on the resource, principal, action, and conditions). The order in which the enforcement code evaluates the policies is not important.
3	In all those policies, the enforcement code looks for an explicit deny instruction that would apply to the request. If it finds even one, the enforcement code returns a decision of "deny" and the process is finished (this is an explicit deny; for more information, see Explicit Deny (p. 35)).
4	If no explicit deny is found, the enforcement code looks for any "allow" instructions that would apply to the request. If it finds even one, the enforcement code returns a decision of "allow" and the process is done (the service continues to process the request).
5	If no allow is found, then the final decision is "deny" (because there was no explicit deny or allow, this is considered a <i>default deny</i> (for more information, see Default Deny (p. 35))).

IAM policies you write address only users and groups within your AWS account, and the policies provide a default deny for any requests from a different AWS account. A default deny can be overridden by an allow, which means any allow permissions you've given to another AWS account through another permissions system override the default deny. For example, if a different AWS account sends a request to use one of your AWS account's Amazon S3 buckets, the IAM result is a deny by default. But, if you've updated the bucket's ACL and given that AWS account permission to use the bucket, then that AWS account can use the bucket. The IAM policies in this case won't prevent it.

Example of an Explicit Deny Overriding an Allow

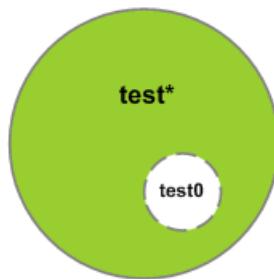
This example illustrates how you might use an explicit deny to override a broad policy that allows access to a wide set of resources. For example, let's say you give a group access to use any Amazon SQS queues in your AWS account whose names begin with the string test.

The following diagram represents that set of queues.

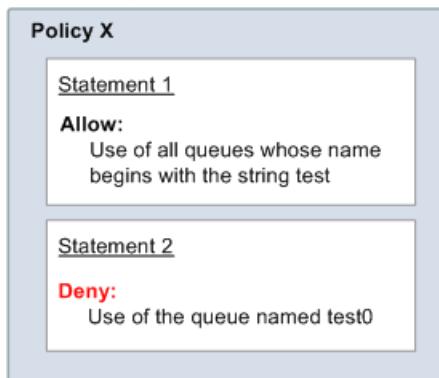


Let's say that you have a particular queue called test0 that you want to remove the group's access to.

The following diagram represents that set of queues.



You could add another policy to the group (or just add a statement to the existing policy) that explicitly denies the group's access to just that queue. The group would still have access to all other queues whose names begin with the string test. The following diagram illustrates those two statements in the policy.



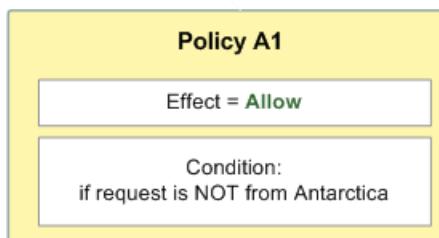
When someone in the group requests to use the queue test0, the explicit deny overrides the allow, and the user is denied access to the queue.

The Interplay of Explicit and Default Denials

A policy results in a default deny if it doesn't directly apply to the request. For example, if a user requests to use Amazon SQS, but the only policy that applies to the user states that the user can use Amazon SimpleDB, then that policy results in a default deny.

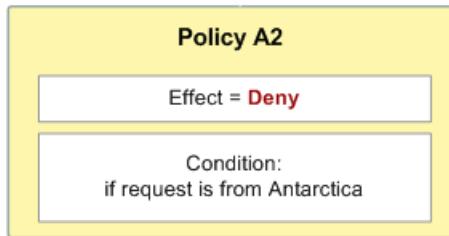
A policy also results in a default deny if a condition in a statement isn't met. If all conditions in the statement are met, then the policy results in either an allow or an explicit deny, based on the value of the Effect element in the policy. Policies don't specify what to do if a condition isn't met, and so the default result in that case is a default deny.

For example, let's say you want to prevent requests coming in from Antarctica. You write a policy (called Policy A1) that allows a request only if it doesn't come from Antarctica. The following diagram illustrates the policy.



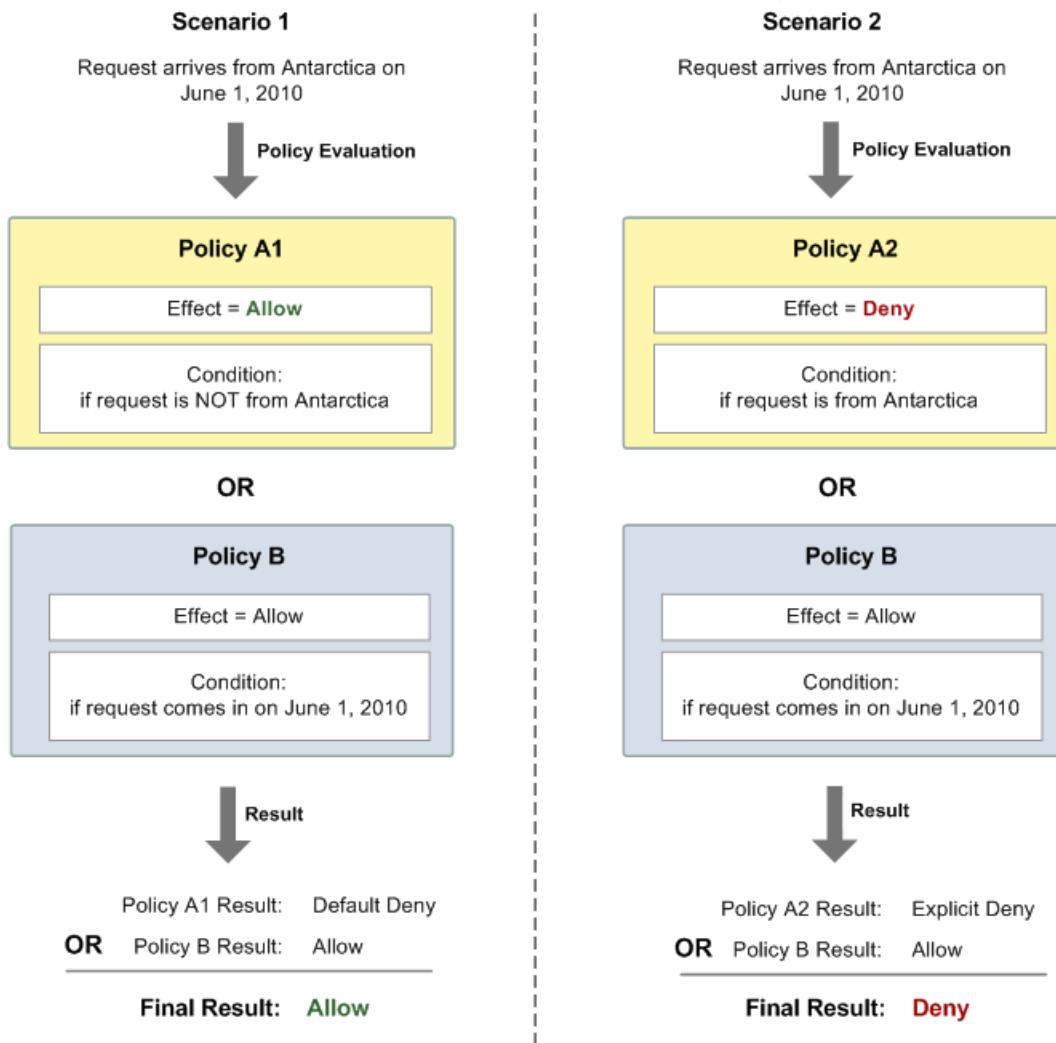
If someone sends a request from the U.S., the condition is met (the request is not from Antarctica). Therefore, the request is allowed. But, if someone sends a request from Antarctica, the condition isn't met, and the policy's result is therefore a default deny.

You could turn the result into an explicit deny by rewriting the policy (named Policy A2) as in the following diagram. Here, the policy explicitly denies a request if it comes from Antarctica.



If someone sends a request from Antarctica, the condition is met, and the policy's result is therefore an explicit deny.

The distinction between a default deny and an explicit deny is important because a default deny can be overridden by an allow, but an explicit deny can't. For example, let's say there's another policy that allows requests if they arrive on June 1, 2010. How does this policy affect the overall outcome when coupled with the policy restricting access from Antarctica? We'll compare the overall outcome when coupling the date-based policy (we'll call Policy B) with the preceding policies A1 and A2. Scenario 1 couples Policy A1 with Policy B, and Scenario 2 couples Policy A2 with Policy B. The following figure and discussion show the results when a request comes in from Antarctica on June 1, 2010.



In Scenario 1, Policy A1 returns a default deny, as described earlier in this section. Policy B returns an allow because the policy (by definition) allows requests that come in on June 1, 2010. The allow from Policy B overrides the default deny from Policy A1, and the request is therefore allowed.

In Scenario 2, Policy B2 returns an explicit deny, as described earlier in this section. Again, Policy B returns an allow. The explicit deny from Policy A2 overrides the allow from Policy B, and the request is therefore denied.

How to Write a Policy

Topics

- [Basic Policy Structure \(p. 43\)](#)
- [Element Descriptions \(p. 43\)](#)
- [Supported Data Types \(p. 57\)](#)

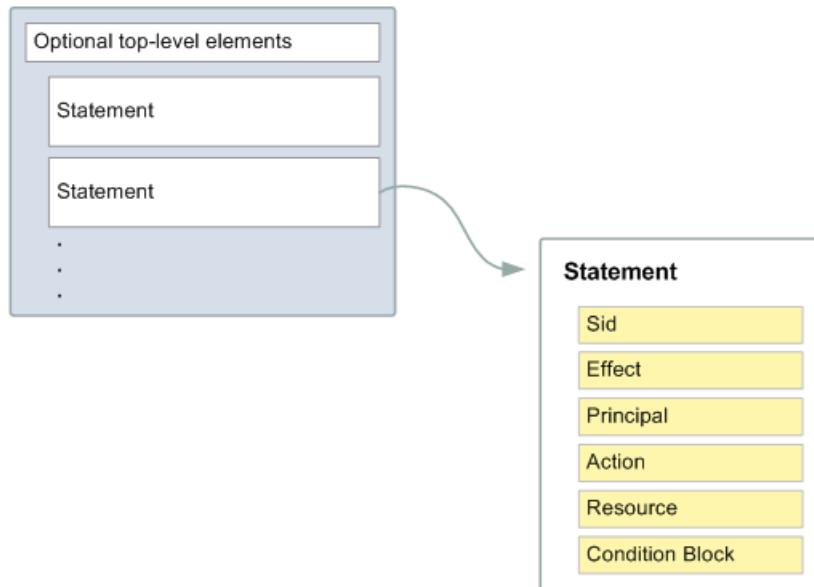
This section describes how to write policies and gives reference information about each policy element.

Basic Policy Structure

Each policy is a JSON document. As illustrated in the following figure, a policy includes:

- Optional policy-wide information (at the top of the document)
- One or more individual *statements*

Each statement includes the core information about a single permission. If a policy includes multiple statements, we apply a logical OR across the statements at evaluation time. If multiple policies are applicable to a request, we apply a logical OR across the policies at evaluation time.



The information in a statement is contained within a series of *elements*. For information about these elements, see [Element Descriptions \(p. 43\)](#).

Element Descriptions

Topics

- [Version \(p. 44\)](#)
- [Id \(p. 44\)](#)
- [Statement \(p. 45\)](#)
- [Sid \(p. 45\)](#)
- [Effect \(p. 45\)](#)

- [Principal \(p. 46\)](#)
- [NotPrincipal \(p. 47\)](#)
- [Action \(p. 47\)](#)
- [NotAction \(p. 48\)](#)
- [Resource \(p. 48\)](#)
- [NotResource \(p. 49\)](#)
- [Condition \(p. 49\)](#)

This section describes the elements that you can use in a policy. The elements are listed here in the general order you use them in a policy. In all services, the `Version` and `Statement` are top-level policy elements and the rest are statement-level elements. In some services, like Amazon SQS and Amazon SNS, the `Id` element is also a top-level element.

All elements are optional for the purposes of parsing the policy document itself. The order of the elements doesn't matter (e.g., the `Resource` element can come before the `Action` element). You're not required to specify any `Condition` elements in the policy.

Note

Not all services support policies, but most do. The details of what goes into a policy vary for each service, depending on what actions the service makes available, what types of resources it contains, and so on. When you're writing policies for a specific service, it's helpful to see examples of policies for that service. For a list of all the services that support IAM, and for links to the documentation in those services that discusses IAM and policies, see [Integrating with Other AWS Products \(p. 195\)](#).

Version

The `Version` element specifies the access policy language version. This is an optional element. Currently the only allowed value is `2008-10-17`.

```
"Version": "2008-10-17"
```

Id

The `Id` element specifies an optional identifier for the policy. The ID is used differently in different services.

For IAM policies, the service automatically sets the policy's `Id` value to the policy name that you create. If you attempt to set the `Id` element, the ID is rejected. The `Id` value for an IAM policy might look like the following example.

```
"Id": "Admin_Policy"
```

For services that let you set an `Id` element, we recommend you use a UUID (GUID) for the value, or incorporate a UUID as part of the ID to ensure uniqueness.

```
"Id": "cd3ad3d9-2776-4ef1-a904-4c229d1642ee"
```

Important

Some AWS services (for example, Amazon SQS or Amazon SNS) might require this element and have uniqueness requirements for it. For service-specific information about writing policies, refer to the documentation for the service you're working with.

Statement

The `Statement` element is the main element for a policy. It can include multiple elements (see the subsequent sections in this page). The `Statement` element contains an array of individual statements. Each individual statement is a JSON block enclosed in braces `{ }`.

```
"Statement": [ { . . . } , { . . . } , { . . . } ]
```

The following example shows a policy that contains an array of two statements inside a single `Statement` element.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "sdb:*" , "s3:*" ],
      "Resource": [ "arn:aws:sdb:us-east-1:123456789012:domain/mySDBDomain",
                    "arn:aws:s3:::my_corporate_bucket",
                    "arn:aws:s3:::my_corporate_bucket/*" ]
    },
    {
      "Effect": "Deny",
      "Action": [ "sdb:*" , "s3:*" ],
      "NotResource": [ "arn:aws:sdb:us-east-1:123456789012:domain/mySDBDomain",
                      "arn:aws:s3:::my_corporate_bucket",
                      "arn:aws:s3:::my_corporate_bucket/*" ]
    }
  ]
}
```

Sid

The `Sid` (statement ID) is an optional identifier that you provide for the policy statement. In services that let you specify an `ID` element, such as SQS and SNS, the `Sid` value is just a sub-ID of the policy document's ID. In IAM, the `Sid` value must be unique within a policy.

```
"Sid": "1"
```

In IAM, the `Sid` is not exposed in the IAM API. You can't retrieve a particular statement based on this ID.

Important

Some AWS services (for example, SQS or SNS) might require this element and have uniqueness requirements for it. For service-specific information about writing policies, refer to the documentation for the service you're working with.

Effect

The `Effect` element is required and specifies whether you want the statement to result in an allow or an explicit deny. Valid values for `Effect` are `Allow` and `Deny`.

```
"Effect": "Allow"
```

By default, access to resources is denied. To allow access to a resource, you must set the `Effect` element to `Allow`. To override an allow (for example, to override an allow that is inherited from another policy), you set the `Effect` element to `Deny`. For more information, see [Evaluation Logic \(p. 37\)](#).

Principal

The `Principal` element specifies the user, account, service, or other entity that is allowed or denied access to a resource.

For IAM users and groups, you do not specify a principal. For those cases, the principal is implicitly the user or group that you are attaching the policy to.

In IAM, principals are used in roles to specify who can assume the role. In other services, principals are used in policies that are attached to a resource, such as an SQS queue or an S3 bucket.

You specify a principal using the *Amazon Resource Name* (ARN) of the account, IAM user, group, or role, using a format like this:

```
arn:aws:<vendor>:<region>:<namespace>:<relative-id>
```

Where:

- `vendor` identifies the AWS product (e.g., Amazon SNS) using a namespace.
- `region` is the region the resource resides in (e.g., `us-east-1`), if any. (IAM users, groups, and roles do not include a region.)
- `namespace` is an AWS account ID with no hyphens (e.g., `111122223333`).
- `relative-id` is the portion that identifies the user, group, role, etc.

For more information about the format of ARNs, see [ARNs \(p. 61\)](#).

You can use wildcards (*) to indicate all possible users. As a shortcut, if you're specifying an account ID, you can use a shortened form that consists of the `AWS`: prefix followed by the ID.

The following examples show various ways in which principals can be specified.

```
<!-- Everyone (anonymous users) -->
"Principal": "AWS": "*.*"

<!-- Specific account or accounts -->
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
"Principal": { "AWS": "123456789012" }
"Princip
al": { "AWS": [ "arn:aws:iam::123456789012:root", "arn:aws:iam::999999999999:root" ] }

<!-- Individual user -->
"Principal": "AWS": "arn:aws:iam::123456789012:user/Username"

<!-- Specific role -->
"Principal": { "AWS": [ "arn:aws:iam::123456789012:role/MyRole" ] }

<!-- Specific service -->
"Principal": { "Service": [ "ec2.amazonaws.com" ] }
```

Some services support additional options for specifying a principal. For example, Amazon S3 lets you use a canonical user in a format like this:

```
<!-- Individual user in S3 -->
"Principal": {"CanonicalUser": "79a59df900b949e55d96ale698fbaced
fd6e09d98eacf8f8d5218e7cd47ef2be"}
```

For more information, see the following:

- [Amazon SQS Policy Examples](#) in the *Amazon Simple Queue Service Developer Guide*
- [Example Cases for Amazon S3 Bucket Policies](#) in the *Amazon Simple Storage Service Developer Guide*
- [Account Identifiers](#) in the AWS Portal (login required)

NotPrincipal

The `NotPrincipal` element lets you specify an exception to a list of principals. For example, you can use this to prevent all AWS accounts except a specific account from accessing a resource. As with `Principal`, you specify the user or account that should be allowed or denied permission; the difference is that `NotPrincipal` translates to everyone *except* that person or account.

In the following example, all users are denied access to a resource except for the user named Bob in the AWS account 123456789012.

```
"Effect": "Deny",
"NotPrincipal": {
    "AWS": "arn:aws:iam::123456789012:user/Bob"
}
```

Action

The `Action` element describes the type of access that should be allowed or denied (for example, read, write, list, delete, and so on). Each AWS service has its own set of actions that describe tasks that you can perform with that service.

You specify a value using a prefix that identifies a service (`iam`, `sqs`, `sns`, `s3`, etc.) followed by the name of the action to allow or deny. The name must match an action that is supported by the service. The prefix and the action name are case insensitive. For example, `iam>ListAccessKeys` is the same as `IAM:listaccesskeys`.

```
<!-- SQS action -->
>Action": "sqs:SendMessage"

<!-- EC2 action -->
>Action": "ec2:StartInstances"

<!-- IAM action -->
>Action": "iam:ChangePassword"

<!-- S3 action -->
>Action": "s3:GetObject"
```

You can specify multiple values for this element.

```
"Action": [ "sqs:SendMessage", "sqs:ReceiveMessage" ]
```

You can use a wildcard (*) to give access to all the actions the specific AWS product offers. For example, the following Action element applies to all IAM actions.

```
"Action": "iam:*"
```

You can also use wildcards (* or ?) as part of the action name. For example, the following Action element applies to all IAM actions that include the string AccessKey, including CreateAccessKey, DeleteAccessKey, ListAccessKeys, and UpdateAccessKey.

```
"Action": "iam:AccessKey*"
```

Some services let you limit the actions that are available. For example, Amazon SQS lets you make available just a subset of all the possible SQS actions. In that case, the wildcard doesn't allow complete control of the queue; it allows only the subset of actions that you've shared. (For more information, see [Understanding Permissions](#) in the *Amazon Simple Queue Services Developer Guide*.)

NotAction

The NotAction element lets you specify an exception to a list of actions. For example, you can use NotAction to let users use only the Amazon SQS SendMessage action.

The following example refers to all actions *other than* the Amazon SQS SendMessage. You would use this in a policy with "Effect": "Deny" to keep users from accessing any other actions.

```
"NotAction": "sns:SendMessage"
```

The following example matches any action except Publish.

```
"NotAction": "sns:Publish"
```

The following example shows how to reference all S3 actions except GetObject.

```
"NotAction": "s3:GetObject"
```

For an example of how to use the NotAction element in a policy that controls access to an Amazon S3 bucket, see Example 5 under [Example Policies for Amazon S3](#) in the *Amazon Simple Storage Service Developer Guide*.

Resource

The Resource element specifies the object or objects that the policy covers. You specify a resource using an ARN. (For more information about the format of ARNs, see [ARNs \(p. 61\)](#).)

Each service has its own set of resources. Although you always use an ARN to specify a resource, the details of the ARN for a resource depend on the service and the resource. For information about how to specify a resource, refer to the documentation for the service whose resources you're writing a policy for.

The following example refers to a specific SQS queue.

```
"Resource": "arn:aws:sqs:us-east-1:111122223333:queue1"
```

The following example refers to the IAM user named Bob whose path is /division_abc/subdivision_xyz/ in an AWS account.

```
"Resource": "arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/Bob"
```

You can use wildcards as part of the resource ARN. The following example refers to all IAM users whose path is /division_abc/subdivision_xyz/.

```
"Resource": "arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/*"
```

The following example refers to all items within a specific S3 bucket.

```
"Resource": "arn:aws:s3:::my_corporate_bucket/*"
```

You can specify multiple resources. The following example refers to a specific Amazon SimpleDB domain, a specific S3 bucket, and all locations within that bucket.

```
"Resource": [ "arn:aws:sdb:us-east-1:123456789012:domain/mySDBDomain",
  "arn:aws:s3:::my_corporate_bucket",
  "arn:aws:s3:::my_corporate_bucket/*" ]
```

NotResource

The `NotResource` element lets you specify an exception to a list of resources. For example, you can use `NotResource` if you want users to be able to access a specific Amazon SQS queue that belongs to the AWS account. If a new queue is added to the AWS account, the administrator doesn't have to update the policy with the new queue's name in order to prevent users from being able to use the queue—by default, users won't be able to use it.

The following example refers to all resources *except* the queue called `my_corporate_queue`. You would use this in a policy with `"Effect": "Deny"` to keep users from accessing any queue except this one.

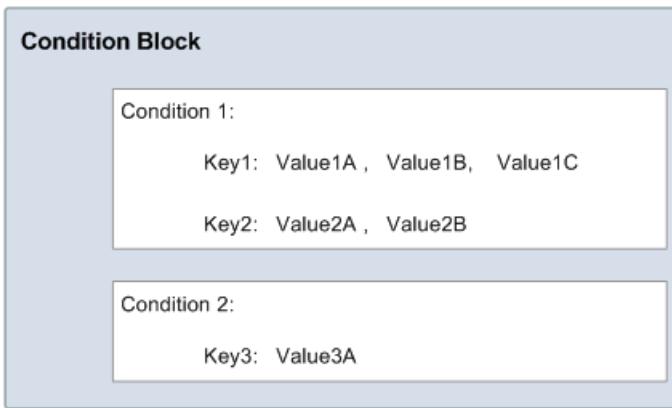
```
"NotResource": "arn:aws:sqs:*:123456789012:my_corporate_queue"
```

Condition

The `Condition` element lets you specify conditions for when a policy is in effect. Condition values can include date, time, the IP address of the requester, the ARN of the request source, and the user agent of the requester. In the `Condition` element, you build expressions in which you use Boolean operators (equal, less than, etc.) to match your condition against values in the request.

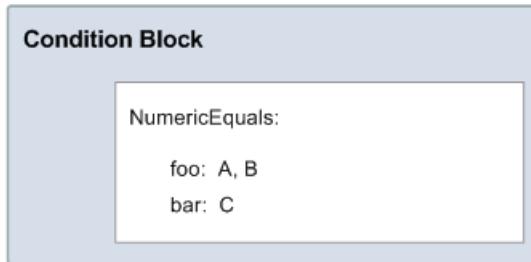
The Condition Block

We refer to the `Condition` element as the *condition block*, because although it has a single `Condition` element, it can contain multiple conditions, and each condition can contain multiple key-value pairs. The following figure illustrates this. Unless otherwise specified, all keys can have multiple values.

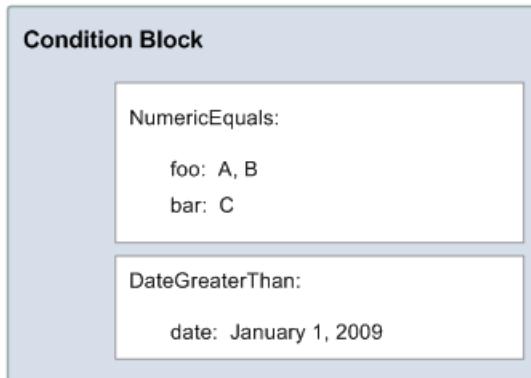


When you create a condition block, you specify the name of each condition and at least one key-value pair. AWS defines the conditions and keys you can use (as listed in the next sections).

An example of a condition is `NumericEquals`. Let's say you want to let John use a resource only if a numeric value `foo` equals either A or B, and another numeric value `bar` equals C. You would create a condition block that looks like the following figure.

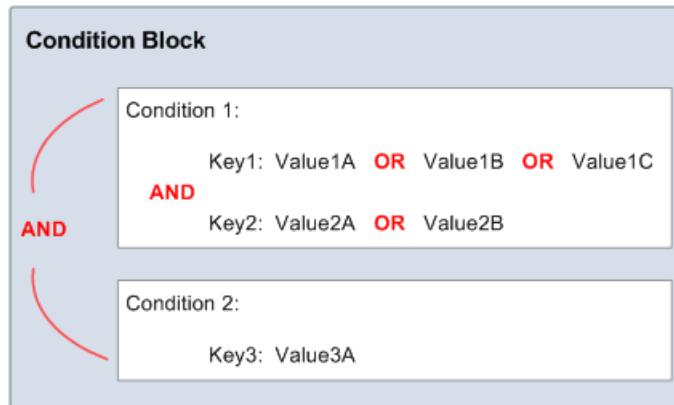


Let's say you also want to restrict John's access to after January 1, 2009. You would add another condition, `DateGreaterThan`, with a date equal to January 1, 2009. The condition block would then look like the following figure.



If there are multiple conditions, or if there are multiple keys in a single condition, we evaluate them using a logical `AND`. If a single condition include multiple values for one key, we evaluate that condition using

a logical OR. All conditions must be met to return an allow or an explicit deny decision. If a condition isn't met, the result is a default deny.



As noted, AWS has predefined conditions and keys (like `aws:CurrentTime`). Individual AWS services can also define service-specific keys. For a list of available keys, see [Available Keys \(p. 51\)](#).

As an example, let's say you want to let user John access your Amazon SQS queue under the following conditions:

- The time is after 12:00 noon on 8/16/2010
- The time is before 3:00 p.m. on 8/16/2010
- The request (IAM or SQS) or message (SNS) comes from an IP address within the range 192.168.176.0 to 192.168.176.24 or the range 192.168.143.0 to 192.168.143.24.

Your condition block has three separate conditions, and all three of them must be met for John to have access to your queue, topic, or resource.

The following shows what the condition block looks like in your policy. The two values for `aws:SourceIp` are evaluated using OR. The three separate conditions are evaluated using AND.

```
"Condition" : {
    "DateGreaterThan" : {
        "aws:CurrentTime" : "2009-04-16T12:00:00Z"
    },
    "DateLessThan" : {
        "aws:CurrentTime" : "2009-04-16T15:00:00Z"
    },
    "IpAddress" : {
        "aws:SourceIp" : [ "192.168.176.0/24" , "192.168.143.0/24" ]
    }
}
```

Available Keys

AWS provides the following predefined keys for all AWS services that support the access policy language for access control:

- `aws:CurrentTime`—To check for date/time conditions (see [Date Conditions \(p. 54\)](#)).

- `aws:EpochTime`—To check for date/time conditions using a date in epoch or UNIX time (see [Date Conditions \(p. 54\)](#)).
- `aws:MultiFactorAuthAge`—To check how long ago (in seconds) the MFA-validated security credentials making the request were issued using Multi-Factor Authentication (MFA). Unlike other keys, if MFA is not used successfully, this key is not present (see [Existence of Condition Keys \(p. 57\)](#), [Numeric Conditions \(p. 53\)](#) and [Using Multi-Factor Authentication \(MFA\) Devices with AWS \(p. 106\)](#)).
- `aws:SecureTransport`—To check whether the request was sent using SSL (see [Boolean Conditions \(p. 55\)](#)).
- `aws:SourceArn`—To check the source of the request, using the Amazon Resource Name (ARN) of the source. (This value is available for only some services. For more information, see [Amazon Resource Name \(ARN\)](#) under "Element Descriptions" in the *Amazon Simple Queue Service Developer Guide*.)
- `aws:SourceIp`—To check the requester's IP address (see [IP Address \(p. 55\)](#)).
- `aws:UserAgent`—To check the requester's client application (see [String Conditions \(p. 52\)](#)).

The key names are case insensitive. For example, `aws:CurrentTime` is equivalent to `AWS:currenttime`.

Note

If you use `aws:SourceIp`, and the request comes from an Amazon EC2 instance, we evaluate the instance's public IP address to determine whether access is allowed.

Each AWS service that uses the access policy language might also provide service-specific keys.

Condition Types

These are the types of conditions you can specify:

- String
- Numeric
- Date and time
- Boolean
- IP address
- Amazon Resource Name (ARN). (This value is available for only some services.)
- ...IfExists
- Existence of condition keys

String Conditions

String conditions let you restrict access based on comparing a key to a string value.

Condition	Description
<code>StringEquals</code>	Exact matching, case sensitive. Short version: <code>streq</code>
<code>StringNotEquals</code>	Negated matching Short version: <code>strneq</code>
<code>StringEqualsIgnoreCase</code>	Exact matching, ignoring case Short version: <code>streqi</code>
<code>StringNotEqualsIgnoreCase</code>	Strict negated matching, ignoring case Short version: <code>strneqi</code>

Condition	Description
StringLike	Loose case-sensitive matching. The values can include a multi-character match wildcard (*) or a single-character match wildcard (?) anywhere in the string. Short version: strl
StringNotLike	Negated loose case-insensitive matching. The values can include a multi-character match wildcard (*) or a single-character match wildcard (?) anywhere in the string. Short version: strnl

For example, the following statement uses the `StringEquals` condition with the `aws:UserAgent` key to specify that the request must include a specific value in its user agent header.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:*AccessKey*",
      "Resource": "arn:aws:iam::123456789012:user/*",
      "Condition": {
        "StringEquals": {
          "aws:UserAgent": "Example Corp Java Client"
        }
      }
    }
  ]
}
```

Numeric Conditions

Numeric conditions let you restrict access based on comparing a key to an integer or decimal value. (Fractional or irrational values are not supported.)

Condition	Description
NumericEquals	Strict matching Short version: numeq
NumericNotEquals	Strict negated matching Short version: numneq
NumericLessThan	"Less than" matching Short version: numlt
NumericLessThanEquals	"Less than or equals" matching Short version: numlteq
NumericGreater Than	"Greater than" matching Short version: numgt
NumericGreater ThanEquals	"Greater than or equals" matching Short version: numgteq

For example, the following statement uses the `NumericLessThanEquals` condition with the `s3:max-keys` key to specify that the requester can list up to 10 objects in `example_bucket` at a time.

```
{  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "s3>ListBucket",  
            "Resource": "arn:aws:s3:::example_bucket",  
            "Condition": {  
                "NumericLessThanEquals": {  
                    "s3:max-keys": "10"  
                }  
            }  
        }  
    ]  
}
```

Date Conditions

Date conditions let you restrict access based on comparing a key to a date/time value. You use these conditions with the `aws:CurrentTime` key or `aws:EpochTime` keys. You must specify date/time values with one of the [W3C implementations of the ISO 8601 date formats](#) or in epoch (UNIX) time.

Note

Wildcards are not permitted for date conditions.

Condition	Description
<code>DateEquals</code>	Strict matching Short version: <code>dateeq</code>
<code>DateNotEquals</code>	Strict negated matching Short version: <code>dateneq</code>
<code>DateLessThan</code>	A point in time at which a key stops taking effect Short version: <code>datelt</code>
<code>DateLessThanEquals</code>	A point in time at which a key stops taking effect Short version: <code>datelteq</code>
<code>DateGreaterThan</code>	A point in time at which a key starts taking effect Short version: <code>dategt</code>
<code>DateGreaterThanOrEquals</code>	A point in time at which a key starts taking effect Short version: <code>dategteq</code>

For example, the following statement uses the `DateLessThan` condition with the `aws:CurrentTime` key to specify that the request must be received before June 30, 2013.

```
{  
    "Statement": [  
        {  
            "Effect": "Allow"-,  
            "Action": "iam:*AccessKey*",  
            "Resource": "arn:aws:iam::123456789012:user/*",  
            "Condition": {  
                "DateLessThan": {  
                    "aws:EpochTime": "1372395600"  
                }  
            }  
        }  
    ]  
}
```

```
        "Condition": {
            "DateLessThan": {
                "aws:CurrentTime": "2013-06-30T00:00:00Z"
            }
        }
    }
}
```

Boolean Conditions

Boolean conditions let you restrict access based on comparing a key to "true" or "false."

Condition	Description
Bool	Strict Boolean matching

For example, the following statement uses the `Bool` condition with the `aws:SecureTransport` key to specify that the request must use SSL.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:*AccessKey*",
            "Resource": "arn:aws:iam::123456789012:user/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
        }
    ]
}
```

IP Address

IP address conditions let you restrict access based on comparing a key to an IP address or range of IP addresses. You use these with the `aws:SourceIp` key. The value must be in the standard CIDR format (for example, 10.52.176.0/24). For more information, see [RFC 4632](#).

Condition	Description
IpAddress	The specified IP address or range
NotIpAddress	All IP addresses except the specified IP address or range

For example, the following statement uses the `IpAddress` condition with the `aws:SourceIp` key to specify that the request must come from the IP range 192.168.176.0 to 192.168.176.24.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:*AccessKey*",
            "Condition": {
                "IpAddress": "192.168.176.0/24"
            }
        }
    ]
}
```

```
        "Resource": "arn:aws:iam::123456789012:user/*",
        "Condition": {
            "IpAddress": {
                "aws:SourceIp": "192.168.176.0/24"
            }
        }
    }
}
```

Amazon Resource Name (ARN)

Amazon Resource Name (ARN) conditions let you restrict access based on comparing a key to an ARN. The ARN is considered a string. (This value is available for only some services.)

Condition	Description
ArnEquals	Strict matching for ARN
ArnNotEquals	Strict negated matching for ARN
ArnLike	Loose case-insensitive matching of the ARN. Each of the six colon-delimited components of the ARN is checked separately and each can include a multi-character match wildcard (*) or a single-character match wildcard (?).
ArnNotLike	Negated loose case-insensitive matching of the ARN. The values can include a multi-character match wildcard (*) or a single-character match wildcard (?) anywhere in the string.

The following example shows a policy you need to attach to any queue that you want Amazon SNS to send messages to. It gives Amazon SNS permission to send messages to the queue (or queues) of your choice, but only if the service is sending the messages on behalf of a particular Amazon SNS topic (or topics). You specify the queue in the `Resource` field, and the Amazon SNS topic as the value for the `SourceArn` key.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": { "AWS": "210987654321" },
            "Action": "sns:Publish",
            "Resource": "arn:aws:sns:us-east-1:123456789012:your_special_topic_1",
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:sqs:us-east-1:01234567891:your_queue_xyz"
                }
            }
        }
    ]
}
```

The following example policy grants permissions to anonymous users for items in an S3 bucket.

```
{
    "Version": "2008-10-17",
    "Statement": [ {
```

```
        "Sid": "AddPerm",
        "Effect": "Allow",
        "Principal": { "AWS": "*" },
        "Action": [ "s3:ReadObject" ],
        "Resource": [ "arn:aws:s3:::bucket/*" ]
    }
}
```

...IfExists Conditions

You can add `IfExists` at the end of any condition except the `Null` condition—for example, `StringEqualsIfExists`. You do this if you mean "If the policy key is present in the context of the request, process the key as specified in the policy. If the key is not present, I don't care."

Existence of Condition Keys

Use a `Null` condition to check if a condition key is present at the time of authorization. In the policy statement, use either `true` (the key doesn't exist) or `false` (the key exists and its value is not null). You can use this condition to determine if a user has authenticated with MFA (multi-factor authentication). For example, the following condition states that MFA authentication must exist (be *not null*) for the user to use the Amazon EC2 API.

```
{
  "Statement": [ {
    "Action": [ "ec2:*" ],
    "Effect": "Allow",
    "Resource": [ "*" ],
    "Condition": {
      "Null": { "aws:MultiFactorAuthAge": "false" }
    }
  }
]
```

Supported Data Types

This section lists the set of data types the access policy language supports. The language doesn't support all types for each policy element (for the supported data types for each element, see [Element Descriptions \(p. 43\)](#)).

The access policy language supports the following data types:

- Strings
- Numbers (Ints and Floats)
- Boolean
- Null
- Lists
- Maps
- Structs (which are just nested Maps)

The following table maps each data type to the serialization. Note that all policies must be in UTF-8. For information about the JSON data types, go to [RFC 4627](#).

Type	JSON
String	String
Integer	Number
Float	Number
Boolean	true false
Null	null
Date	String adhering to the W3C Profile of ISO 8601
IpAddress	String adhering to RFC 4632
List	Array
Object	Object

Ways to Access IAM

AWS Identity and Access Management is available through the following interfaces:

- AWS Management Console
- Command line interface (CLI)
- IAM Query API
- Existing libraries

If you don't want to write code, you can use IAM through the AWS Management Console, or through a Java-based command line interface (CLI). The AWS Management Console enables you to do almost everything you can do with any of the code-based interfaces. You should already have read [Getting Started \(p. 4\)](#), which shows you how to use the AWS Management Console to do the following:

- Create groups and assign permissions to groups
- Add users
- Create security credentials for your users
- Assign passwords to your users

For more information about accessing IAM through the console, see [How Users Sign In to the AWS Management Console \(p. 190\)](#).

The CLI commands are not mirror images of the API actions; instead, the CLI combines API actions into multi-function commands. For example, to set up a new user with the API, you must make separate API calls to create the user, create an access key for the user, and add the user to groups. In the CLI, you can perform all those functions with one command. For information about setting up and using the IAM CLI, go to the [AWS Identity and Access Management Command Line Interface Reference](#).

You can also code directly to the IAM Query API, which is an interface that accepts basic HTTPS requests. The interface is designed to provide low-level building blocks that perform atomic functions, giving you the flexibility to create work flows that match your needs. In most cases, the API provides CRUD actions (create, read, update, delete) for the different IAM entities (users, groups, policies, and so on).

If you prefer to use IAM through an existing programmatic interface, AWS offers SDKs for:

- Java
- .NET

About IAM Entities

Topics

- [Identifiers for IAM Entities \(p. 60\)](#)
- [Limitations on IAM Entities \(p. 64\)](#)

This section describes identifiers and limitations for the basic IAM entities (users, groups, and so on).

Identifiers for IAM Entities

Topics

- [Friendly Names and Paths \(p. 60\)](#)
- [ARNs \(p. 61\)](#)
- [GUIDs \(p. 64\)](#)

IAM uses a few different identifiers for users, groups, roles and server certificates. This section describes the identifiers and when you use each.

Friendly Names and Paths

When you create a user, a role, or a group, or when you upload a server certificate, you give it a friendly name, such as Bob, TestApp1, Developers, or ProdServerCert. Whenever you need to specify a particular entity in an API call to IAM (for example, to delete a user, or update a group with a new user), you use the friendly name.

If you are using the API or command line interface to create users, you can also optionally give the entity a path that you define. You might use the path to identify which division or part of the organization the entity belongs in. For example: /division_abc/subdivision_xyz/product_1234/engineering/. Examples of how you might use paths are shown in the next section (see [ARNs \(p. 61\)](#)).

Just because you give a user and group the same path doesn't automatically put that user in that group. For example, you might create a Developers group and specify its path as /division_abc/subdivision_xyz/product_1234/engineering/. Just because you create a user named Bob and give him that same path doesn't automatically put Bob in the Developers group.

IAM doesn't enforce any boundaries between users or groups based on their paths. Users with different paths can use the same resources (assuming they've been granted permission to). For information about limitations on names, see [Limitations on IAM Entities \(p. 64\)](#).

ARNs

Although most resources have a friendly name (for example, a user named Bob or a group named Developers), the access policy language requires you to specify the resource or resources using the following *Amazon Resource Name (ARN)* format.

```
arn:aws:<service>:<region>:<namespace>:<relative-id>
```

Where:

- `service` identifies the AWS product (for example, iam)
- `region` is the Region the resource resides in (for example, us-east-1), if any
- `namespace` is the AWS account ID with no hyphens (for example, 123456789012)
- `relative-id` is the portion that identifies the specific resource

This format is applicable to all types of AWS resources (IAM users and federated users, groups, roles, server certificates, virtual MFA devices, Amazon SimpleDB domains, Amazon SQS queues, and so on). The specific values to use in the ARN (such as the service and region) depend on the AWS product itself. The following information covers the format to use for IAM resources. For information about the format to use for other AWS products, see [Integrating with Other AWS Products \(p. 195\)](#).

IAM has several types of resources you can specify in a policy: users (IAM and federated), groups, roles, instance profiles, virtual MFA devices, and server certificates. The following table shows the ARN format for each and an example. The region portion of the ARN is blank because IAM resources are global. (For more information about using server certificates, see [Managing Server Certificates \(p. 198\)](#).)

Resource type	Format for ARN
User	<code>arn:aws:iam::{account_ID}:user/{path/to/user/UserName}</code> Example: <code>arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/Bob</code>
Federated user	<code>arn:aws:sts::{account_ID}:federated-user/{UserName}</code> Example: <code>arn:aws:sts::123456789012:federated-user/Bob</code>
Group	<code>arn:aws:iam::{account_ID}:group/{path/to/group/GroupName}</code> Example: <code>arn:aws:iam::123456789012:group/division_abc/subdivision_xyz/product_A/Developers</code>
Role	<code>arn:aws:iam::{account_ID}:role/{path/to/role/RoleName}</code> Example: <code>arn:aws:iam::123456789012:role/application_abc/component_xyz/S3Access</code>
Instance profile	<code>arn:aws:iam::{account_ID}:instance-profile/{path/to/instance-profile/InstanceProfileName}</code> Example: <code>arn:aws:iam::123456789012:instance-profile/application_abc/component_xyz/Webserver</code>
Virtual MFA	<code>arn:aws:iam::{account_ID}:mfa/{path/to/virtual_device/VirtualDeviceName}</code> Example: <code>arn:aws:iam::123456789012:mfa/division_abc/subdivision_xyz/BobJonesMFA</code> For virtual MFA devices, the ARN is the serial number of the device.

Resource type	Format for ARN
Server certificate	arn:aws:iam::{account_ID}:server-certificate/{path/to/server certificate/ServerCertName} Example: arn:aws:iam::123456789012:server-certificate/division_abc/subdivision_xyz/ProdServerCert
AWS account (root)	arn:aws:iam::{account_ID}:root Example: arn:aws:iam::123456789012:root

When you create a user, group, role, or virtual MFA device, or when you upload a server certificate, we return the ARN to you as a convenience. For more information about federated users, see [Using Temporary Security Credentials](#).

The following example shows a policy you could assign to Bob to allow him to manage his own access keys. Notice that the resource is Bob himself.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["iam:*AccessKey*"],
      "Resource": "arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/Bob"
    }
  ]
}
```

You can use wildcards in the `relative-id` portion of the ARN to specify multiple users or groups. For example, to specify all users working on product_1234, you would use:

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/product_1234/*
```

Let's say you have users whose names start with the string `app_`. You could refer to them all with the following ARN.

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/product_1234/app_*
```

To specify all users or groups in the AWS account, use a wildcard after the `user/` or `group/` part of the ARN, respectively.

```
arn:aws:iam::123456789012:user/*
arn:aws:iam::123456789012:group/*
```

Don't use a wildcard in the `user/` or `group/` part of the ARN. In other words, the following is not allowed:

```
arn:aws:iam::123456789012:u*
```

Example 1: Use of Paths and ARNs for Distributed Administrator Groups

Dave is the main administrator in Example Corp., and he decides to use paths to help delineate the users in the company and set up a separate administrator group for each path-based division. Following is a subset of the full list of paths he plans to use:

- /marketing
- /sales
- /legal

Dave creates an administrator group for the marketing part of the company and calls it Marketing_Admin. He assigns it the /marketing path. The group's ARN is arn:aws:iam::123456789012:group/marketing/Marketing_Admin.

Dave assigns the following policy to the Marketing_Admin group that gives the group permission to use all IAM actions with all groups and users in the /marketing path. The policy also gives the Marketing_Admin group permission to perform any Amazon S3 actions on the objects in the portion of the corporate bucket dedicated to the marketing employees in the company.

```
{  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "iam:*",  
            "Resource": [ "arn:aws:iam::123456789012:group/marketing/*",  
                        "arn:aws:iam::123456789012:user/marketing/*" ]  
        },  
        {  
            "Effect": "Allow",  
            "Action": "s3:*",  
            "Resource": "arn:aws:s3:::example_bucket/marketing/*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "s3>ListBucket*",  
            "Resource": "arn:aws:s3:::example_bucket",  
            "Condition": {  
                "StringLike": {  
                    "s3:prefix": "marketing/*"  
                }  
            }  
        }  
    ]  
}
```

The policy has a separate statement that is necessary to let the group list the objects only in the portion of the bucket dedicated to the marketing group. For more information about constructing policies to control user and group access to Amazon S3, go to [Using IAM Policies](#) in the *Amazon Simple Storage Service Developer Guide*.

Dave then creates a user named Jules in the /marketing path, and assigns Jules to the Marketing_Admin group. Jules can now create and manage new users and groups in the /marketing path, and work with the objects in the marketing part of the bucket.

Dave then sets up similar administrator groups for the other paths (for example, /sales, etc.).

Example 2: Use of Paths and ARNs for a Project-Based Group

In this example, Jules in the Marketing_Admin group creates a project-based group within the /marketing path, and assigns users from different parts of the company to the group. This example illustrates that a user's path isn't related to the groups the user is in.

The marketing group has a new product they'll be launching, so Jules creates a new group in the /marketing path called Widget_Launch. Jules then assigns the following policy to the group, which gives the group access to objects in the part of the example_bucket designated to this particular launch.

```
{  
    "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/*"  
                }  
            }  
        }  
    ]  
}
```

Jules then assigns the users who are working on this launch to the group. This includes Pat and Eli from the /marketing path. It also includes Chris and Carl in the /sales path, and Aline and Jim from the /legal path.

GUIDs

We assign each user, group, and server certificate a globally unique identifier (GUID), which we return to you when you use the API or CLI to create it. We recommend you store the GUID in your own database along with the user, group, or certificate name. Internally we use the GUID to identify the user, group, or certificate, and we translate the value into the ARN or friendly name as appropriate when displaying the user, group, or certificate information to you. If you delete a user, group, or server certificate, any residual remote references to that item display the GUID as the friendly name part in the ARN. If you've stored the GUID in your own system, you can then use the displayed GUID to identify the deleted item being referred to.

Limitations on IAM Entities

This section lists restrictions on IAM entities, and describes how to get information about entity usage and quotas.

Note

To retrieve account level information about entity usage and quotas, use the [GetAccountSummary](#) API action or the [iam-accountgetsummary](#) CLI command.

Following are restrictions on names:

- Names of users, groups, roles, instance profiles, and server certificates must be alphanumeric, including the following common characters: plus (+), equal (=), comma (,), period (.), at (@), and dash (-).
- Path names must begin with a forward slash (/).
- Policy names must be unique to the user, group, or role they are attached to, and can contain any Basic Latin (ASCII) characters, minus the following reserved characters: backward slash (\), forward slash (/), asterisk (*), question mark (?), and white space. These characters are reserved according to RFC 3986 (for more information, see <http://www.ietf.org/rfc/rfc3986.txt>).
- User passwords (login profiles) can contain any Basic Latin (ASCII) characters.
- AWS account ID aliases must be unique across AWS products, and must be alphanumeric following DNS naming conventions. An alias must be lowercase, it must not start or end with a hyphen, it cannot contain two consecutive hyphens, and it cannot be a 12 digit number.

For a list of Basic Latin (ASCII) characters, go to the [Library of Congress Basic Latin \(ASCII\) Code Table](#).

Names for entities are case sensitive and must be unique within the scope of your AWS account (regardless of the path you might give the entity).

Following are the default maximums for your entities:

- Groups per AWS account: 100
- Users per AWS account: 5000
If you need to add a large number of users, consider using temporary security credentials. For more information about temporary security credentials, go to [Using Temporary Security Credentials](#).
- Roles per AWS account: 250
- Instance Profiles per AWS account: 100
- Number of groups per user: 10 (that is, the user can be in this many groups)
- Access keys per user: 2
- Signing certificates per user: 2
- MFA devices in use per user: 1
- MFA devices in use per AWS account (at the root account level): 1
- Virtual MFA devices (assigned or unassigned) per AWS account: equal to the user quota for the account
- Server certificates per AWS account: 10
- AWS account aliases per AWS account: 1
- Login profiles per user: 1

You can request to increase these quotas for your AWS account on the [IAM Limit Increase Contact Us Form](#).

Following are the maximum lengths for entities:

- Path: 512 characters
- User name: 64 characters
- Group name: 128 characters
- Role name: 64 characters
- Instance profile name: 128 characters
- GUID (applicable to users, groups, roles, and server certificates): 32 characters
- Policy name: 128 characters
- Certificate ID: 128 characters
- Login profile password: 1 to 128 characters

- AWS account ID alias: 3 to 63 characters.
- You can add as many policies as you want to a user, role, or group, but the total aggregate policy size (the sum size of all policies) per entity cannot exceed the following limits:
 - User policy size cannot exceed 2,048 characters
 - Role policy size cannot exceed 10,240 characters
 - Group policy size cannot exceed 5,120 characters

Setting Up an Administrators Group

The first thing we recommend you do after creating an AWS account is to set up an administrators group. It's not required, but is highly recommended. Going forward, the users in the administrators group should set up the groups, users, and so on, for the AWS account, and all future key-based interaction should be through the AWS account's users and their own keys. You can stop using the AWS account's security credentials to make API calls to AWS.

Tip

If you read through the [Getting Started \(p. 4\)](#), you used the AWS Management Console to set up an administrators group for your AWS account. We've repeated the information here if you're interested in using a different interface than the one presented in the *Getting Started*.

Process for Setting Up an Administrators Group

1	Create a group with a name of your choice (e.g., <i>Admins</i>). For more information, see Creating a Group (p. 67) .
2	Add a policy that gives the group access to all AWS actions and resources. For more information, see Adding a Policy to the Group (p. 69) .
3	Set up at least one user in the group. For more information, see Adding a New User to Your AWS Account (p. 72) .

Creating a Group

This section shows how to create a group in the IAM system. For information about the limitations on the group's name and the maximum number of groups you can have, see [Limitations on IAM Entities \(p. 64\)](#).

Command Line Interface

To create an administrators group

1. Use the `iam-groupcreate` command with the name you've chosen for the group and an optional path you've chosen. For more information about paths, see [Friendly Names and Paths \(p. 60\)](#).

In this example, you create a group called *Admins* with the default path (/).

```
PROMPT> iam-groupcreate -g Admins
```

If the command is successful, there's no response.

2. Use the `iam-grouplistbypath` command to list the groups in your AWS account and confirm the group was created.

```
PROMPT> iam-grouplistbypath
```

Sample response:

```
arn:aws:iam::123456789012:group/Admins
```

The response lists the Amazon Resource Name (ARN) for your *Admins* group. The ARN is a standard format that AWS uses to identify resources. The 12-digit number in the ARN is your AWS account ID. The friendly name you assigned to the group (*Admins*) appears at the end of the group's ARN.

API

To create an administrators group

1. Use the `CreateGroup` action with the name you've chosen for the group and an optional path you've chosen. The following example uses *Admins* as the name of the group.

How you structure the `AUTHPARAMS` depends on how you are signing your API request. For information on `AUTHPARAMS` in Signature Version 4, go to [Examples of Signed Signature Version 4 Requests](#). For information on `AUTHPARAMS` in Signature Version 2, go to [Signature Version 2 Signing Process](#).

For more information about the `CreateGroup` action, refer to the [AWS Identity and Access Management API Reference](#), or refer to your SDK's documentation.

```
https://iam.amazonaws.com/  
?Action=CreateGroup  
&Path=/  
&GroupName=Admins  
&Version=2010-05-08  
&AUTHPARAMS
```

IAM returns output similar to the following:

```
<CreateGroupResponse>  
  <CreateGroupResult>  
    <Group>  
      <Path>/</Path>  
      <GroupName>Admins</GroupName>  
      <GroupId>AGPACKCEVSQ6C2EXAMPLE</GroupId>  
      <Arn>arn:aws:iam::123456789012:group/Admins</Arn>  
    </Group>  
  </CreateGroupResult>  
  <ResponseMetadata>  
    <RequestId>7a62c49f-347e-4fc4-9331-6e8eEXAMPLE</RequestId>
```

```
    </ResponseMetadata>
</CreateGroupResponse>
```

2. Use the `ListGroups` action to list the groups in your AWS account and confirm the group was created.

```
https://iam.amazonaws.com/
?Action=ListGroups
&Version=2010-05-08
&AUTHPARAMS
```

Sample response:

```
<ListGroupsResponse>
  <ListGroupsResult>
    <Groups>
      <member>
        <Path>/</Path>
        <GroupName>Admins</GroupName>
        <GroupId>AGPACKCEVSQ6C2EXAMPLE</GroupId>
        <Arn>arn:aws:iam::123456789012:group/Admins</Arn>
      </member>
    </Groups>
    <IsTruncated>False</IsTruncated>
  </ListGroupsResult>
  <ResponseMetadata>
    <RequestId>7a62c49f-347e-4fc4-9331-6e8eEXAMPLE</RequestId>
  </ResponseMetadata>
</ListGroupsResponse>
```

Adding a Policy to the Group

This section shows how to add a policy that lets any user in the group perform any action on any resource in the AWS account. You write the policy using the *access policy language*, and then attach the policy to the Admins group. For more information about the access policy language, see [The Access Policy Language \(p. 33\)](#).

Command Line Interface

To add a policy giving root privileges

1. Create a policy document by copying and pasting the following JSON block into a text file.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

2. Save the text file as `MyPolicy.txt` in the main CLI installation directory.
3. Type the following command to upload this policy to IAM and attach it to your Admins group. The command assigns the following name to the policy: `AdminRoot`.

```
PROMPT> iam-groupuploadpolicy -g Admins -p AdminRoot -f MyPolicy.txt
```

If the command is successful, there's no response.

4. Type the following command to confirm the policy is attached to the Admins group.

```
PROMPT> iam-grouplistpolicies -g Admins
```

The response lists the names of the policies attached to the Admins group. Sample response:

```
AdminRoot
```

In the preceding policy, you could have used `iam-groupaddpolicy` instead of the `iam-groupuploadpolicy`. The `iam-groupaddpolicy` command lets you specify the contents of the policy as parameters instead of uploading a JSON policy document file. Following is an example of what the `iam-groupaddpolicy` command would look like. The `-o` option causes the output to include the JSON policy document we construct for you based on the parameters you provided.

```
PROMPT> iam-groupaddpolicy -g Admins -p AdminRoot -e Allow -a "*" -r "*" -o
```

Sample response:

```
{"Version": "2008-10-17", "Statement": [ {"Effect": "Allow", "Action": ["*"], "Resource": ["*"]}]}
```

API

To add a policy giving root privileges

1. Create the following JSON policy.

```
{  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "*",  
            "Resource": "*"  
        }  
    ]  
}
```

2. Issue a `PutPolicy` request. The following example names the policy `AdminRoot` and attaches it to the Admins group. The policy document in the request doesn't include white space or line feeds, and it's not URL encoded here so you can more easily see what it looks like (normally you must form URL encode it as part of POST request). For more information about how to do a POST request, go to [Examples of Signed Signature Version 4 Requests](#), or [Signature Version 2 Signing Process](#).

```
POST / HTTP/1.1
Host: iam.amazonaws.com
Content-Type: application/x-www-form-urlencoded

Action=PutPolicy
&GroupName=Admins
&PolicyName=AdminRoot
&PolicyDocument={"Statement":[{"Effect":"Allow","Action":"*","Resource": "*"}]}
&Version=2010-05-08
&AUTHPARAMS
```

Sample response:

```
<PutPolicyResponse>
  <ResponseMetadata>
    <RequestId>7a62c49f-347e-4fc4-9331-6e8eEXAMPLE</RequestId>
  </ResponseMetadata>
</PutPolicyResponse>
```

3. Issue a `ListGroupPolicies` request to confirm the policy is attached to the Admins group.

```
https://iam.amazonaws.com/
?Action=ListGroupPolicies
&GroupName=Admins
&Version=2010-05-08
&AUTHPARAMS
```

Sample response:

```
<ListGroupPoliciesResponse>
  <ListGroupPoliciesResult>
    <PolicyNames>
      <member>AdminRoot</member>
    </PolicyNames>
    <IsTruncated>false</IsTruncated>
  </ListGroupPoliciesResult>
  <ResponseMetadata>
    <RequestId>7a62c49f-347e-4fc4-9331-6e8eEXAMPLE</RequestId>
  </ResponseMetadata>
</ListGroupPoliciesResponse>
```

You can confirm the contents of a particular policy with the `GetUserPolicy` or `GetGroupPolicy` actions. For information about the actions, go to the [AWS Identity and Access Management API Reference](#), or refer to your SDK's documentation.

Important

After you have the administrators group set up, you must add at least one user to it. For more information about adding users to a group, see [Adding a New User to Your AWS Account \(p. 72\)](#).

Working with Users and Groups

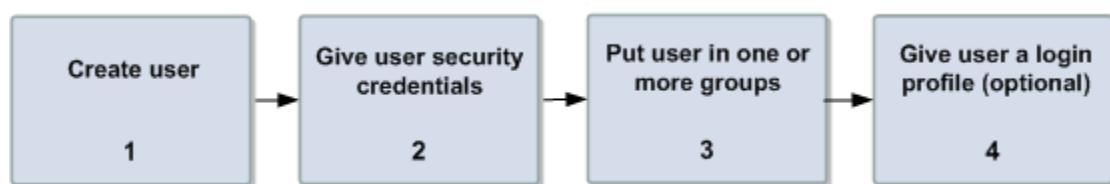
Topics

- [Adding a New User to Your AWS Account \(p. 72\)](#)
- [Listing Users \(p. 79\)](#)
- [Deleting a User from Your AWS Account \(p. 82\)](#)
- [Creating and Listing Groups \(p. 85\)](#)
- [Adding Users to and Removing Users from a Group \(p. 89\)](#)
- [Deleting a Group \(p. 91\)](#)
- [Renaming Users and Groups \(p. 93\)](#)
- [Managing Passwords \(p. 95\)](#)
- [Using Multi-Factor Authentication \(MFA\) Devices with AWS \(p. 106\)](#)
- [Managing User Keys and Certificates \(p. 124\)](#)

This section describes how to create and manage users and groups.

Adding a New User to Your AWS Account

When someone joins your organization, or if you have a new application or system that needs to make API calls to AWS, you need to add a new user to your AWS account. This section describes the process for setting up a user in IAM.



Process for Adding a New User

1	Create a user.
---	----------------

2	Optionally, if the user will be making API calls or using the command line interface, you need to create security credentials for the user and give them to the user.
3	Optionally add the user to one or more groups. If you've decided to create a group that includes all your AWS account's users (for example, an <i>AllUsers</i> group), make sure to add the new user to that group.
4	Optionally give the user a password, which is required if the user needs to use the AWS Management Console. For more information about passwords and optional MFA devices for securing passwords, see Passwords (p. 25) and Multi-Factor Authentication for Users (p. 25) .

How you execute the tasks in the preceding table depends on which interface you're using to access IAM. The interface-specific details are covered in the sections that follow.

Note

There are character limitations for user names. For more information, see [Limitations on IAM Entities \(p. 64\)](#).

AWS-Assigned User Identifiers

When you create a user, in return you receive the following AWS-assigned identifiers:

- Amazon Resource Name (ARN) for the user
- Globally unique identifier (GUID) for the user (returned only when you use the API or CLI to create the user)

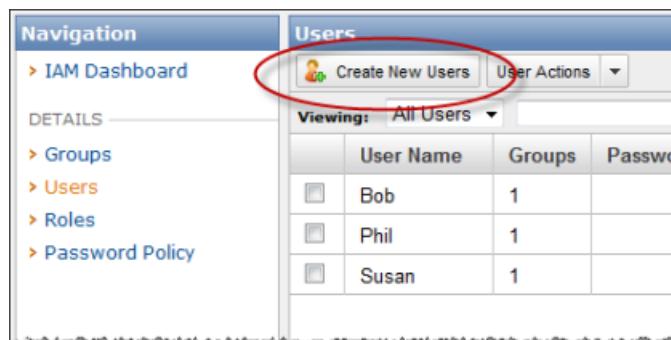
For more information about these identifiers, see [Identifiers for IAM Entities \(p. 60\)](#).

AWS Management Console

If you're accessing IAM through the AWS Management Console, you can create and manage users and their security credentials in a few simple steps.

To add a user

1. On the **Navigation** pane of the console, click **Users**, and then click **Create New Users**.



2. Enter the user names for users you want to create. You can create up to five users at one time.

AWS Identity and Access Management Using IAM AWS Management Console

The screenshot shows the 'Create User' dialog box. At the top, there is a 'Cancel' button. Below it, the heading 'Enter User Names:' is followed by five input fields labeled 1. John, 2. Janet, 3., 4., and 5. A note below the fields states 'Maximum 128 characters each'. Underneath the input fields is a checked checkbox labeled 'Generate an access key for each user'. A note next to it says 'Users need access keys to make secure REST or Query protocol requests to AWS service APIs.' and 'For Users who need access to the AWS Management Console, create a password in the Users panel after completing this wizard.' At the bottom right of the dialog is a 'Create' button.

3. If you want to automatically generate an Access Key ID and Secret Access Key for your new users, select **Generate an access key for each user**. Users need keys if they will be working with an AWS API. For more information about creating user keys for existing users, go to [Managing User Keys and Certificates](#) in *Using AWS Identity and Access Management*.

Note

If you have users who will be working with the AWS Management Console, you will need to create passwords for each of them. Creating passwords is described later in this procedure.

4. Click **Create**.
5. If you chose to create security credentials for the users, click **Download Credentials** to save the Access Key IDs and Secret Access Keys to a .CSV file on your computer. You will not have access to the Secret Access Keys again after this dialog box closes, and you will need to provide this information to your users before they can begin using an AWS API.

The screenshot shows the 'Manage Access Keys' dialog box. At the top, there is a 'Cancel' button. Inside, a green message box says 'Your access key has been created successfully.' Below it, a note says 'This is the last time these User security credentials will be available for download. You can manage and recreate these credentials any time.' There is a link 'Show User Security Credentials' and a 'Download Credentials' button, which is circled in red. At the bottom right is a 'Close Window' button.

6. When you are finished downloading your users' security credentials, click **Close Window** to continue.
7. Users who will access the AWS Management Console will need a password. To add a password for a user, follow these steps:
 - a. In the **Navigation** pane, click **Users**.
 - b. Select the user who you want to create a password for, and then select the user **Security Credentials** tab.
 - c. Click **Manage Password**.

AWS Identity and Access Management Using IAM
AWS Management Console

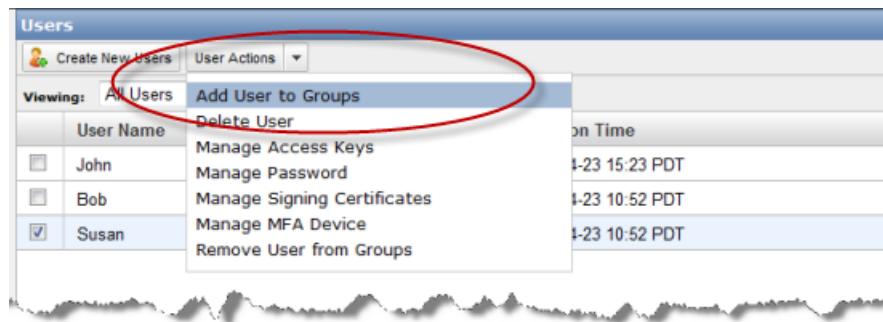
The screenshot shows the AWS IAM 'Users' page. A user named 'Bob' is selected, indicated by a red circle around the checkbox in the 'User Name' column. Below the table, it says '1 User selected'. Under 'User: Bob', there are tabs for 'Groups', 'Permissions', 'Security Credentials' (which is selected), and 'Summary'. In the 'Security Credentials' section, there are two main sections: 'Access Credentials' and 'Sign-In Credentials'. Under 'Access Credentials', it shows 'Access Keys: AK [REDACTED] BQ - Active' created on 2011-04-29T22:11:41Z with a 'Manage Access Keys' button. Under 'Sign-In Credentials', it shows 'User name: Bob' and 'Password: No' (circled in red). There is a 'Manage Password' button next to the password field. Under 'Multi-Factor Authentication Device', it says 'No' with a 'Manage MFA Device' button.

- d. You can create a custom password, or you can have IAM automatically generate a password.
 - To create a custom password, select **Assign a custom password**, and then enter and confirm the password. When you are finished, click **Apply**. The selected user can begin using the password. For information about how users access your sign-in page, see [The AWS Management Console Sign-in Page \(p. 190\)](#).
 - To have IAM generate a password, select **Assign an auto-generated password**, and then click **Apply**. Next, click **Download Credentials** to save the password as a .CSV file to your computer. You will need to provide this password to the user, and you will not be able to access the password after completing this step. Click **Close Window** to continue.

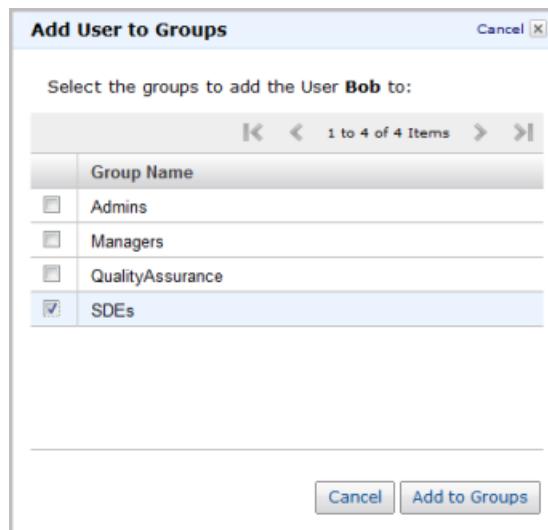
The screenshot shows a 'Manage Password' dialog box. It has a green checkmark icon and the text 'Your password has been created successfully.' Below that, it says 'This is the last time these User security credentials will be available for download. You can manage and recreate these credentials any time.' There is a 'Hide User Security Credentials' link. Under 'Bob', it shows a password field with the value 'ln[REDACTED]8'. At the bottom, there is a 'Download Credentials' button with a green arrow icon and a 'Close Window' button.

8. To add the user to a group, follow these steps:
 - a. Select the user name, and then, from the **User Actions** list, select **Add User to Groups**.

AWS Identity and Access Management Using IAM AWS Management Console



- b. Select the groups you want to add the user to, and then click **Add to Groups**.



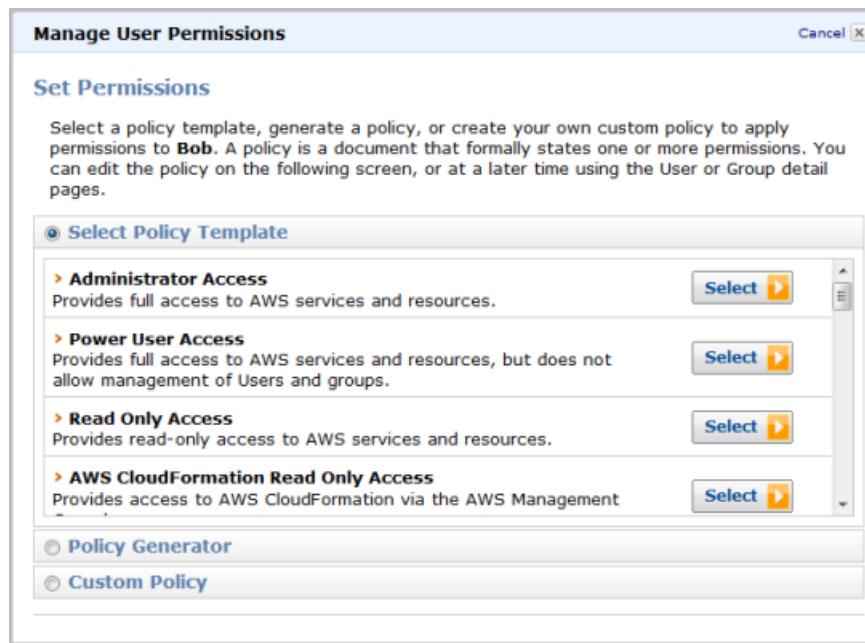
9. To attach a policy to the user, follow these steps:

- a. Select the **Permissions** tab, and then click **Attach User Policy**.



- b. Choose the method for creating the policy document by clicking either **Select Policy Template**, **Policy Generator**, or **Custom Policy**. Then click **Select**.

AWS Identity and Access Management Using IAM AWS Management Console

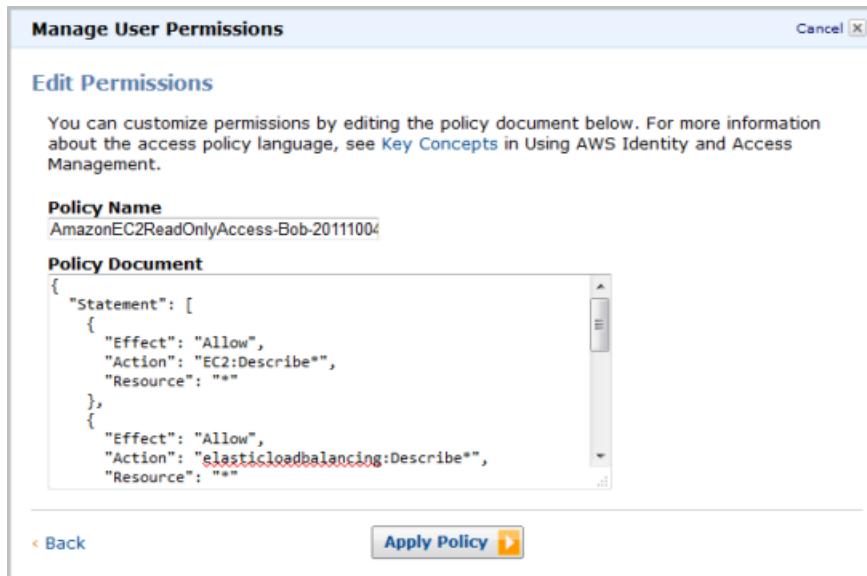


- c. How you complete the next step depends on the method you selected to create the policy.
- If you are using a template to create the policy, review the policy content in the dialog box, then click **Apply Policy**.
 - If you are using the policy generator, select the appropriate **Effect**, **AWS Service**, and **Actions** options, enter the ARN (if applicable), and add any conditions you want to include. Then click **Add Statement**. You can add as many statements as you want to the policy.

The screenshot shows the 'Edit Permissions' dialog box. It includes fields for 'Effect' (set to 'Allow'), 'AWS Service' (set to 'Amazon EC2'), 'Actions' (a dropdown menu), and 'Amazon Resource Name (ARN)' (an input field). Below these are buttons for 'Add Conditions (Optional)' and 'Add Statement'. A table at the bottom lists a single statement: 'Effect' 'Allow', 'Action' 'ec2:MonitorInstances, ec2:RebootInstances', and 'Resource' '*'. At the bottom are 'Back' and 'Continue' buttons.

When you are finished adding statements, click **Continue**. Review the generated output, then click **Apply Policy**.

- If you are using a custom policy, enter a name for the policy under **Policy Name** and write the policy or paste the policy document from your text editor into the **Policy Document** box. When you are finished, click **Apply Policy**.



IAM applies the policy to the user.

Note

There are limitations on policy names and on policy size. For information about policy limitations, see [Limitations on IAM Entities \(p. 64\)](#).

For information about optionally giving users permission to manage their own security credentials, see [Adding a Credentials Management Policy to the User \(p. 135\)](#).

Command Line Interface

If you're using the IAM command line interface, you can do tasks 1–3 in the preceding table with one command: `iam-usercreate`. The command's main purpose is task 1 (creating the user), and it optionally does tasks 2 and 3. You can perform tasks 2 and 3 separately with other commands (`iam-useraddkey` and `iam-groupadduser`).

To give the user a password, use the `iam-useraddloginprofile` command. To give the user permission to manage his or her own security credentials, use the `iam-useraddpolicy` or `iam-useruploadpolicy` commands.

The following example creates a user named Bob, creates an access key for Bob, and assigns Bob to the Developer group.

```
PROMPT> iam-usercreate -u Bob -g Developer -k -v
```

Sample response:

```
arn:aws:iam::123456789012:user/Bob
AIDACKCEV р6C2EXAMPLE
```

The command used verbose mode (`-v`), so the response includes the user's ARN and GUID. If you don't use verbose mode, the response is empty.

We recommend saving the user's new Access Key ID and Secret Access Key to a text file, such as `Bob-credentials.txt`. You can give that file to Bob so he'll have his credentials to use with calls to AWS.

The following example creates the password `Welcome` for the user Bob.

```
PROMPT> iam-useraddloginprofile -u Bob -p Welcome
```

If the command is successful, there is no response.

You can list the users in your AWS account or in a particular group with the `iam-userlistbypath` and `iam-grouplistusers` commands.

For information about optionally giving users permission to manage their own security credentials, see [Adding a Credentials Management Policy to the User \(p. 135\)](#).

API

If you're programmatically accessing IAM, you use a separate API call for each of the tasks involved in setting up a new user. The following table lists the API actions to use.

Process for Adding a New User

1	Create a user: CreateUser
2	Create security credentials for the user: CreateAccessKey
3	Optionally add the user to one or more groups: AddUserToGroup
4	Optionally give the user a password, which is required if the user needs to use the AWS Management Console: CreateLoginProfile

You can also optionally give the user permission to manage his or her own security credentials. For more information, go to [PutUserPolicy](#).

For more information about the actions, go to the [AWS Identity and Access Management API Reference](#), or refer to your SDK's documentation.

Listing Users

This section describes how to list the users in your AWS account or in a particular group.

AWS Management Console

To list all users in your AWS account

- In the **Navigation** pane, click **Users**. The console displays ten users per screen. To see more users, use the page controls on the right side of the console to advance to the next screen.

The screenshot shows the AWS IAM Management Console. In the left navigation pane, under the 'DETAILS' section, 'Users' is selected and highlighted with a red circle. The main area is titled 'Users' and contains a table with five rows of user information. The table has columns for User Name, Groups, Password, Access Keys, and Creation Time. The first four users (Ben, Tom, Colm, Robert) have their 'Groups' column showing a value of '1'. The fifth user (Robert) has a green checkmark in the 'Groups' column. The 'Access Keys' column shows '1 active' for all users. The 'Creation Time' column shows '2012-06-03 16:32 PDT' for all users. At the bottom of the table, there is a pagination control with arrows and the text '1 to 10 of 15 Items'.

User Name	Groups	Password	Access Keys	Creation Time
Ben	1		1 active	2012-06-03 16:32 PDT
Tom	1	✓	1 active	2012-06-03 16:32 PDT
Colm	1		1 active	2012-06-03 16:32 PDT
Robert	1		1 active	2012-06-03 16:32 PDT

To list all users in a group

- In the **Navigation** pane, click **Groups**, and then select the group name. The console displays group details. If necessary, select the **Users** tab to display all users belonging to the group.

The screenshot shows the AWS IAM Management Console. In the left navigation pane, 'Groups' is selected and highlighted with a red circle. Within the 'Groups' section, the 'Ops' group is selected and highlighted with a red circle. The main area is titled 'Groups' and contains a table with five rows of group information. The table has columns for Group Name, Users, Permissions Granted, and Creation Time. The first four groups (QualityAssurance, Ops, Developers, Admins) have their 'Users' column showing values of 3, 4, 11, and 3 respectively. The 'Permissions Granted' column shows green checkmarks for all groups. The 'Creation Time' column shows dates ranging from June 3, 2012, to June 4, 2012. Below the table, a message says '1 Group selected'. Under the 'Ops' group, there is a sub-section titled 'Group: Ops' with tabs for 'Users', 'Permissions', and 'Summary'. The 'Users' tab is selected. It shows a table with two rows: Robert and Swetha. Each row has a 'Remove from group' link under the 'Actions' column.

Group Name	Users	Permissions Granted	Creation Time
QualityAssurance	3	✓	2012-06-03 17:00 PDT
Ops	4	✓	2012-06-03 16:34 PDT
Developers	11	✓	2012-06-03 16:33 PDT
Admins	3	✓	2012-06-03 16:21 PDT

Command Line Interface

The `iam-userlistbypath` command lets you list all the users in the AWS account or list all the users with a particular path prefix. The output lists the ARN for each resulting user.

The `iam-userlistgroups` command lists all the groups a particular user is in. The output lists the ARNs for the groups the user is in. For more information about ARNs and paths, see [Identifiers for IAM Entities \(p. 60\)](#).

For more information about these commands, refer to the [AWS Identity and Access Management Command Line Interface Reference](#).

API

The `ListUsers` action lets you list all the users in the AWS account, or list all the users with a particular path prefix. The response lists the path, ARN, and GUID for each resulting user. For more information about paths, ARNs, and GUIDs, see [Identifiers for IAM Entities \(p. 60\)](#).

If you want to get a list of users in a particular group, use `GetGroup`.

For more information about `ListUsers` and `GetGroup`, go to the [AWS Identity and Access Management API Reference](#), or refer to your SDK's documentation.

Paginating the Results

Some of the API actions such as `ListUsers` and `GetGroup` let you paginate the results. Note that these actions limit the number of users returned in the response, so to get the full list of users, you must call the action multiple times.

To paginate the list results

1. In the first request, use a `MaxItems` parameter specifying how many users you want in the response.

```
https://iam.amazonaws.com/  
?Action=ListUsers  
&MaxItems=10  
&Version=2010-05-08  
&AUTHPARAMS
```

The response includes up to `MaxItems` number of users (assuming there are at least that many total). If there are more users left beyond that maximum, the response includes an element called `IsTruncated` with a `true` value (otherwise, the value is `false`). The response also includes a `Marker` element with a long string as the value.

```
<ListUsersResponse>  
  <ListUsersResult>  
    <Users>  
      <member>  
        ...  
      </member>  
      <member>  
        ...  
      </member>  
      ...  
    </Users>  
    <IsTruncated>true</IsTruncated>  
    <Marker>AAHu6Jk1qUhsapsw5lZ5xq/bjVoYLSbLeLoLhUS4G1RQK2UYzMW40g  
39qbmiJeRLnkYI4WUSXIT45gZEXAMPLE</Marker>  
  </ListUsersResult>  
  <ResponseMetadata>  
    <RequestId>7a62c49f-347e-4fc4-9331-6e8eEXAMPLE</RequestId>
```

```
    </ResponseMetadata>
    </ListUsersResponse>
```

2. In the subsequent request to continue listing the users, again include the `MaxItems` request parameter, and a `Marker` request parameter set to that long `Marker` value from the response.

```
https://iam.amazonaws.com/
?Action=ListUsers
&MaxItems=10
&Marker=AAHu6JklqUhsapsw5lZ5xq/bjVoYLSbLeLoLhUS4G1RQK2UYzMW40g39qbmiJeRLnkYI4
WUSXIT45gZEXAMPLE
&Version=2010-05-08
&AUTHPARAMS
```

If there are still more users to be listed, the `IsTruncated` value in the response is `true`, and there's a new value for `Marker` for you to use in the next request to continue listing.

Deleting a User from Your AWS Account

Topics

- [AWS Management Console \(p. 82\)](#)
- [Command Line Interface or API \(p. 83\)](#)

You can remove a user from your AWS account at any time. For example, you might do this if someone quits your company. The process of removing a user deletes the user's credentials and policies. If the user is only temporarily leaving the company, you could instead just disable the user's credentials instead of deleting the user entirely from the AWS account. That way, you can easily prevent the user from accessing the AWS account's resources during the absence, without removing the user's credentials or policies. For more information about disabling credentials, see [Managing User Keys and Certificates \(p. 124\)](#).

AWS Management Console

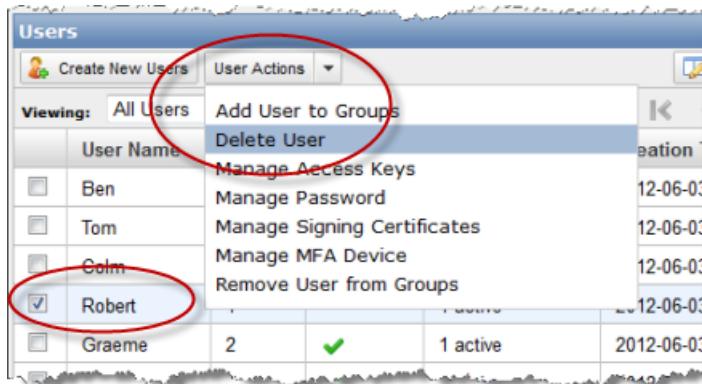
When you use the AWS Management Console to delete an IAM user, IAM also deletes all the information associated with the user. IAM deletes the following information when you delete a user:

- The user
- Any signing certificates belonging to the user
- Any access keys belonging to the user
- Any associated MFA device
- Any password associated with the user
- All policies attached to the user (policies that are applied to a user via group permissions are not affected)

Also, the user is removed from any groups it belongs to.

To use the AWS Management Console to delete a user

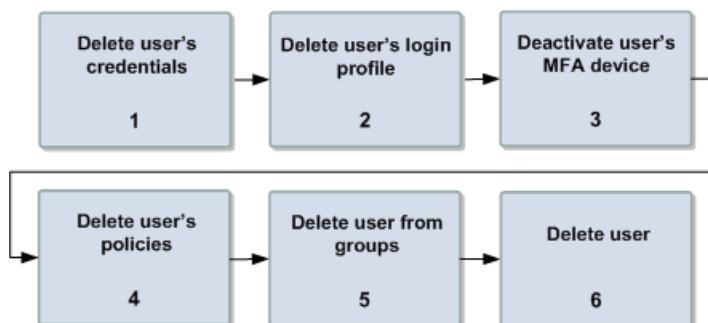
1. In the **Navigation** pane, click **Users**, and then select the user name.
2. From the **User Actions** list, select **Delete User**.



3. Review your changes, and then click **Yes, Delete**.

Command Line Interface or API

The following figure and table describe the process for using the IAM CLI or API to delete a user from your AWS account.



Process for Deleting a User from Your AWS Account

1	Delete the user's keys and certificates. This helps ensure that the user can't access your AWS account's resources anymore. Note that when you delete a security credential, it's gone forever and can't be retrieved.
2	Delete the user's password, if the user has one. For a general description of using IAM to manage passwords, see Passwords (p. 25) .
3	Deactivate the user's MFA device, if the user has one. For a general description of MFA, see Multi-Factor Authentication for Users (p. 25) .
4	Delete any policies that were attached to the user.
5	Get a list of any groups the user was in, and delete the user from those groups.
6	Delete the user.

Note

If you delete a user, any residual remote references to that user (e.g., an Amazon SQS policy) display the GUID in the user's ARN instead of the user's friendly name. If you've stored the GUID in your own system, you can then use the displayed GUID to identify the deleted user being referred to.

How you actually execute the tasks in the preceding table depends on which interface you're using to access IAM. The interface-specific details are covered in the sections that follow.

Command Line Interface

If you're using the command line interface to access IAM, you can use a single command for each of the tasks involved in deleting a user from your AWS account, or you can perform each step individually. The following table lists the commands to use when deleting the user in steps.

To delete the user recursively, use the `iam-userdel` with the `-r` option. Recursively deleting the user automatically deletes it from any associated groups and deletes any attached entities such as keys, signing certificates, and policies. Use the `-r` option with caution. Before performing a recursive delete, to ensure you are not deleting anything you don't want to, use the `-p` option along with the `-r` option to list all the user's associated entities and groups without actually performing the recursive deletion. For more information about the commands, go to the [AWS Identity and Access Management Command Line Interface Reference](#).

Process for Deleting a User from Your AWS Account

1	Delete the user's keys and certificates: <code>iam-userdelkey</code> and <code>iam-userdelcert</code>
2	Delete the user's password, if the user has one: <code>iam-userdelloginprofile</code>
3	Deactivate the user's MFA device, if the user has one: <code>iam-userdeactivatemfadeface</code> or <code>iam-virtualmfadeficedelete</code> (if the user has a virtual MFA device)
4	Delete any policies that were attached to the user: <code>iam-userlistpolicies</code> (to list the policies attached to the user) and <code>iam-userdelpolicy</code>
5	Get a list of any groups the user was in, and delete the user from those groups: <code>iam-userlistgroups</code> and <code>iam-groupremoveuser</code>
6	Delete the user: <code>iam-userdel</code>

API

If you're programmatically accessing IAM, you use a separate API call for each of the tasks involved in setting up a new user. The following table lists the API actions to use.

Process for Deleting a User from Your AWS Account

1	Delete the user's keys and certificates: <code>DeleteAccessKey</code> and <code>DeleteSigningCertificate</code>
2	Delete the user's password, if the user has one: <code>DeleteLoginProfile</code>
3	Deactivate the user's MFA device, if the user has one: <code>DeactivateMFADevice</code> or <code>DeleteVirtualMFADevice</code> (if the user has a virtual MFA device)
4	Delete any policies that were attached to the user: <code>ListUserPolicies</code> (to list the policies attached to the user) and <code>DeleteUserPolicy</code>

5	Get a list of any groups the user was in and delete the user from those groups: ListGroupsForUser and RemoveUserFromGroup
6	Delete the user: DeleteUser

For more information about the actions, go to the [AWS Identity and Access Management API Reference](#), or refer to your SDK's documentation.

Creating and Listing Groups

Topics

- [Attaching a Policy to a Group \(p. 85\)](#)
- [Listing Groups \(p. 87\)](#)

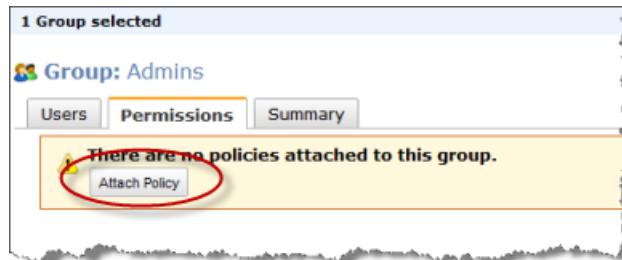
To set up a group, you need to create the group and then give it permissions based on the type of work you expect the users in the group to do. For an example of how to set up a group, go to the [Getting Started \(p. 4\)](#).

Attaching a Policy to a Group

To give a group permissions, you attach a policy to the group as described in the following procedure. For information about permissions and policies, see [Permissions and Policies \(p. 30\)](#).

To attach a policy to a group

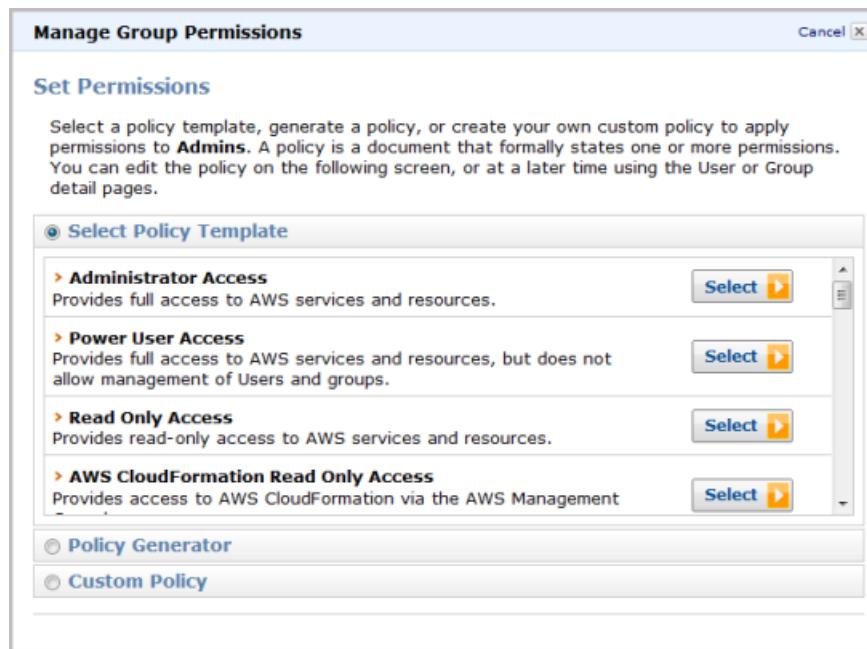
1. In the **Navigation** pane, select **Groups**.
2. Under **Group Name**, select the group that you want to attach a policy to.
3. Select the **Permissions** tab, and then click **Attach Policy**.



4. Choose the method for creating the policy document by clicking either **Select Policy Template**, **Policy Generator**, or **Custom Policy**. Then click **Select**.

AWS Identity and Access Management Using IAM

Attaching a Policy to a Group



5. How you complete the next step depends on the method you selected to create the policy.
 - If you are using a template to create the policy, review the policy content in the dialog box, then click **Apply Policy**.
 - If you are using the policy generator, select the appropriate **Effect**, **AWS Service**, and **Actions** options, enter the ARN (if applicable), and add any conditions you want to include. Then click **Add Statement**. You can add as many statements as you want to the policy.

The screenshot shows the 'Edit Permissions' dialog box. It includes the following fields:

- Effect**: Radio buttons for **Allow** (selected) and **Deny**.
- AWS Service**: A dropdown menu set to **AWS CloudFormation**.
- Actions**: A dropdown menu set to **-- Select Actions --**.
- Amazon Resource Name (ARN)**: An input field containing an empty string.
- Add Conditions (Optional)**: A link to add conditions.
- Add Statement**: A button to add a new statement.

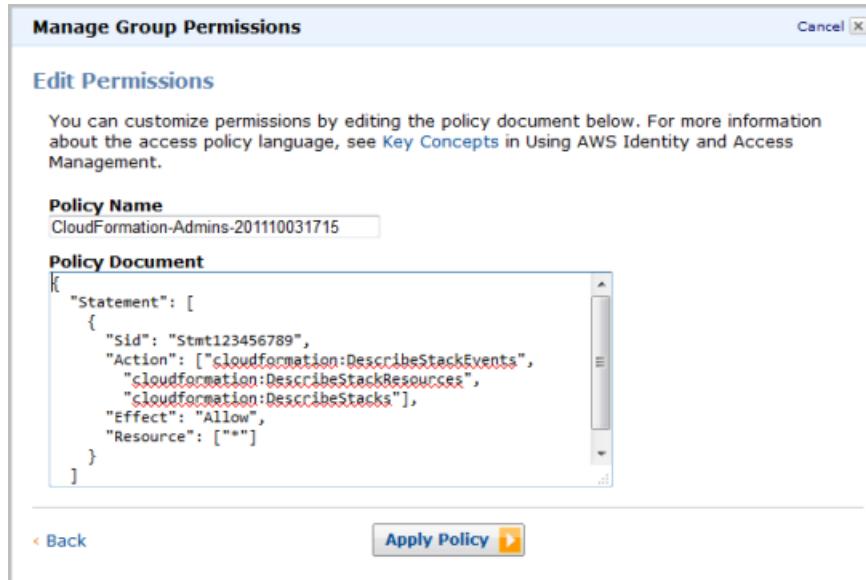
Below these fields is a table showing the current statements:

Effect	Action	Resource	Remove
Allow	cloudformation:DescribeStackEvents cloudformation:DescribeStackResources cloudformation:DescribeStacks	*	Remove

At the bottom of the dialog are buttons for **Back**, **Continue**, and **Remove**.

When you are finished adding statements, click **Continue**. Review the generated output, then click **Apply Policy**.

- If you are using a custom policy, enter a name for the policy under **Policy Name** and write the policy or paste the policy document from your text editor into the **Policy Document** box. When you are finished, click **Apply Policy**.



IAM applies the policy to the group.

Note

There are limitations on policy names and on policy size. For information about policy limitations, see [Limitations on IAM Entities \(p. 64\)](#).

Listing Groups

You can list the groups that a particular user is in, get a list of the groups in the AWS account, or if you are using the IAM API or CLI, you can get a list of the groups with a certain path prefix. How you perform these tasks depends on the interface you use to access IAM.

AWS Management Console

To list all groups in your AWS account

- Click **Groups**. If you have more than 50 groups, you can use the page controls on the right side of the console to advance to the next screen.

AWS Identity and Access Management Using IAM Listing Groups

The screenshot shows the AWS IAM Groups listing interface. On the left, a navigation pane includes links for IAM Dashboard, Groups (which is highlighted with a red oval), Users, Roles, and Password Policy. The main area is titled 'Groups' and shows a table with four items. The table columns are Group Name, Users, Permissions Granted, and Creation Time. The items listed are QualityAssurance (3 users, permissions granted), Ops (4 users, permissions granted), Developers (11 users, permissions granted), and Admins (3 users, permissions granted). At the top right of the table, there is a pagination control showing '1 to 4 of 4 Items' with arrows and a 'Show/Hide' button.

To list all the groups a user belongs to

1. In the Navigation pane, click **Users**, and then select the user name.
2. To view the groups the user belongs to, select the **Groups** tab.

The screenshot shows the AWS IAM User details interface for 'User: John'. In the main table, 'John' is selected (indicated by a checked checkbox) and has 3 groups. Below the table, it says '1 User selected'. Under 'User: John', there are tabs for Groups (highlighted with a red oval), Permissions, Security Credentials, and Summary. The 'Groups' tab shows that John belongs to 3 groups: Managers, Admins, and Developers. Each group entry has a 'Remove from Group' link.

Command Line Interface

The `iam-grouplistbypath` command lets you list all the groups in the AWS account, or list all the groups with a particular path prefix. To get a list of each of the users in a particular group, use the `iam-grouplistusers` command. The output lists the ARN for each resulting group. For more information about ARNs and paths, see [Identifiers for IAM Entities \(p. 60\)](#).

For more information about `iam-grouplistbypath` or `iam-grouplistusers` commands, go to [iam-grouplistbypath](#) or [iam-grouplistusers](#) in the *AWS Identity and Access Management Command Line Interface Reference*.

API

The `ListGroups` action lets you list all the groups in the AWS account, or list all the groups with a particular path prefix. The response lists the path, ARN, and GUID for each resulting group. For more information about paths, ARNs, and GUIDs, see [Identifiers for IAM Entities \(p. 60\)](#).

If you have a large number of groups, you might want to paginate the results. For an example of how to do that, see [Listing Users \(p. 79\)](#).

If you want to get a list of the groups a particular user is in, use `ListGroupsForUser`.

For more information about `ListGroups` and `ListGroupsForUser`, go to [AWS Identity and Access Management API Reference](#), or refer to your SDK's documentation.

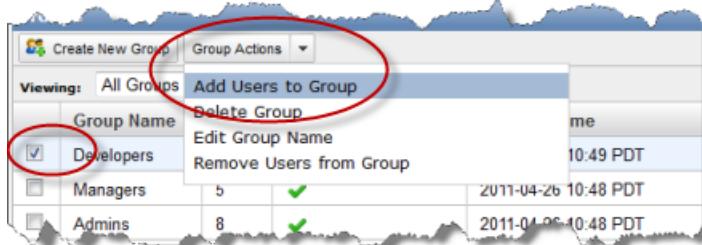
Adding Users to and Removing Users from a Group

You can use the AWS Management Console, the command line interface, or the API to add users to a group, or to remove them.

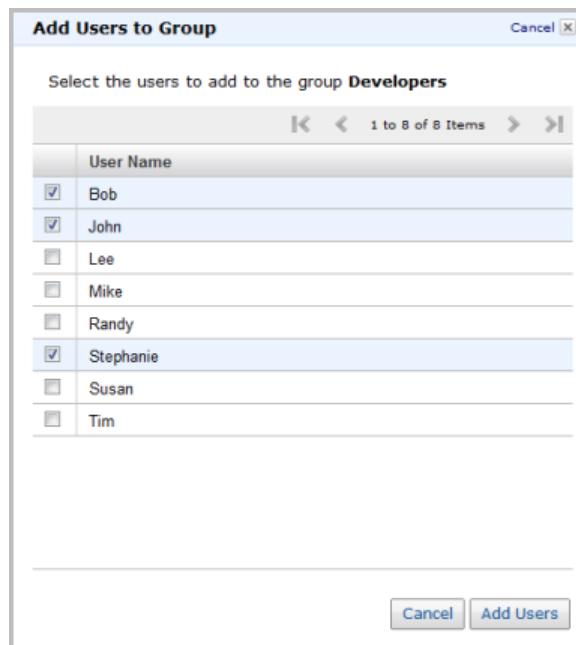
AWS Management Console

To add a user to a group

1. In the **Navigation** pane, click **Groups**, and then select the group name.
2. From the **Group Actions** list, select **Add Users to Group**.

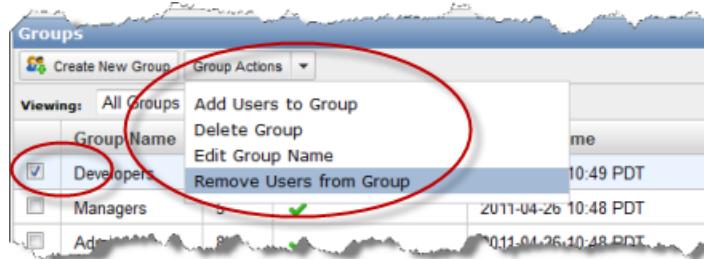


3. Select the users to add to the group, and then click **Add Users**.



To remove a user from a group

1. In the Navigation pane, click **Groups**, and then select the group name.
2. From the **Group Actions** list, select **Remove Users from Group**.



3. Select the users to remove from the group, and then click **Remove Users**.



Command Line Interface

To add a user to a group, use the `iam-groupadduser` command. Use the `iam-groupremoveuser` to remove a user from a group.

API

To add a user to a group, use the `AddUserToGroup` action. To remove a user from a group, use the `RemoveUserFromGroup` action. You can add or remove only a single user to or from a single group with a single API call. For more information about the actions, go to the [AWS Identity and Access Management API Reference](#), or refer to your SDK's documentation.

The response doesn't include an updated list of the users in the group; it just contains the basic information about the group. If you want to get a list of users in the group, use `GetGroup`. For more information about listing users, see [Listing Users \(p. 79\)](#).

Deleting a Group

Topics

- [AWS Management Console \(p. 92\)](#)
- [Command Line Interface or API \(p. 92\)](#)

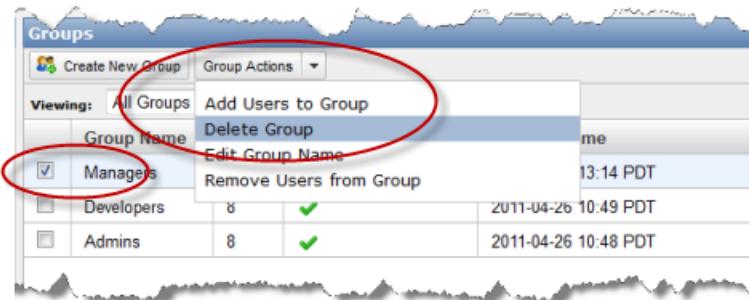
You might want to delete a group that you no longer need. When you use the AWS Management Console to delete a group, IAM deletes the group and any associated policies, but leaves the users intact. If you use the IAM CLI or API to remove the group, you must remove the users and policies before you can remove the group.

AWS Management Console

When you use the console to remove a group, IAM removes any policies associated with the group, and it removes the group. Users are removed from the group, and permissions the users had because they belonged to the group will no longer apply to them.

To use the AWS Management Console to delete a group

1. In the **Navigation** pane, click **Groups**, and then select the group name.
2. From the **Group Actions** list, select **Delete Group**.



3. Review your changes, and then click **Yes, Delete**.

Command Line Interface or API

The following diagram and table describe the general process for deleting a group.



Process for Deleting a Group from Your AWS Account

1	Remove all users from the group.
2	Delete all policies attached to the group.
3	Delete the group.

How you actually execute the tasks in the preceding table depends on which interface you're using to access IAM. The interface-specific details are covered in the sections that follow.

Command Line Interface

If you're using the command line interface to access IAM, you can use a separate command for each of the tasks involved in deleting a group from your AWS account. Or, optionally, you can recursively delete the group and any attached policies by specifying an option with the command. The following table lists

the commands to use. For more information about the commands, go to the [AWS Identity and Access Management Command Line Interface Reference](#).

Process for Deleting a Group from Your AWS Account

1	Individually remove all users from the group: <code>iam-groupremoveuser</code>
2	Delete the policies attached to the group: <code>iam-groupdelpolicy</code>
3	Delete the group: <code>iam-groupdel</code> This function works only when the users have been removed from the group, and when the policies are no longer attached. If you want to delete the group and all attached policies without first removing the users and policies, use the <code>iam-groupdel -r</code> option. For more information about this command, go to the AWS Identity and Access Management Command Line Interface Reference .

API

If you're programmatically accessing IAM, you use a separate API call for each of the tasks involved in setting up a new user. The following table lists the API actions to use. Before deleting the group, you must delete all users from the group and remove any policies attached to it.

Process for Deleting a User from Your AWS Account

1	Remove all users from the group: <code>GetGroup</code> (to get the list of users in the group), and <code>RemoveUserFromGroup</code>
2	Delete all policies attached to the group: <code>ListGroupPolicies</code> (to get a list of the group's policies), and <code>DeleteGroupPolicy</code>
3	Delete the group: <code>DeleteGroup</code>

For more information about the actions, go to the [AWS Identity and Access Management API Reference](#), or refer to your SDK's documentation.

Renaming Users and Groups

Topics

- [Permission to Rename \(p. 93\)](#)
- [Changing a User's Name or Path \(p. 94\)](#)
- [Changing a Group's Name or Path \(p. 94\)](#)

This section shows how to rename or change the path for a user or group.

Permission to Rename

Administrators in your AWS account are probably the only types of users who will have permission to rename users and groups (or change their paths). To understand why, think of changing the name or path of a user or group as a "move" operation. Whoever wants to rename the user or group needs to have permission to do it on both sides of the "move."

For example, let's say a user is changing from one division in the company to another. You need to change the user's path from /division_abc/ to /division_efg/. So, you need permission to remove the user from /division_abc/, and you need permission to put the user into /division_efg/. Effectively this means you need permission to call `UpdateUser` on both `arn:aws:iam::123456789012:user/division_abc/*` and `arn:aws:iam::123456789012:user/division_efg/*`. It's possible that the only people within the organization who have that type of permission are administrators.

Changing a User's Name or Path

You must use the IAM CLI or API to change a user's name. When you change a user's name or path, the following happens:

- Any policies attached to the user stay with the user under the new name
- The user stays in the same groups under the new name
- The GUID for the user remains the same (for more information about GUIDs, see [GUIDs \(p. 64\)](#))

In addition, any policies that refer to the user as *the principal* (the user being granted access) are automatically updated to use the new name or path. For example, any queue-based policies in the Amazon SQS system that give the user access to a particular queue are automatically updated to use the new name and path. Amazon S3 bucket policies are also automatically updated.

However, we do not automatically update policies that refer to the user as *a resource* to use the new name or path; you must manually do that. For example, let's say Bob has a policy attached to him that lets him manage his security credentials. If an administrator renames Bob to Robert, the admin also needs to update that policy to change the resource from

`arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/Bob` to
`arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/Robert`. This is also true if the admin changes the path; the admin needs to update the policy to reflect the new path for the user.

Command Line Interface

The `iam-usermod` command lets you change the user's name or path. For more information about the command, go to the [AWS Identity and Access Management Command Line Interface Reference](#).

API

The `UpdateUser` action lets you change the user's name or path. For more information about the action, go to the [AWS Identity and Access Management API Reference](#), or refer to your SDK's documentation.

Changing a Group's Name or Path

You can use the AWS Management Console to change a group's name, or you can use the IAM CLI or API.

When you change a group's name or path, the following happens:

- Any policies attached to the group stay with the group under the new name
- The group retains all its users under the new name
- The GUID for the group remains the same (for more information about GUIDs, see [GUIDs \(p. 64\)](#))

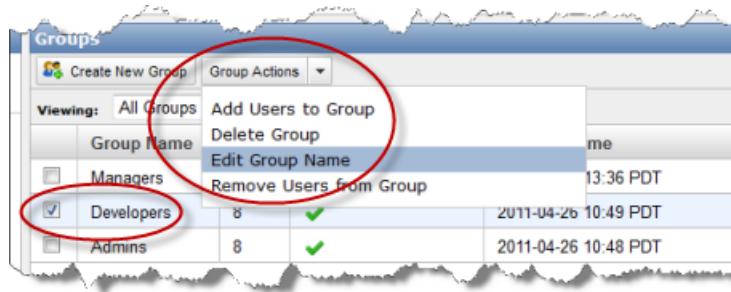
We do not automatically update policies that refer to the group as a resource to use the new name; you must manually do that. For example, let's say Bob is the manager of the testing part of the organization, and he has a policy attached to him that lets him use `UpdateGroup` specifically with the Test group (to add and remove users). Let's say that an admin changes the name of the group to Test_1 (or changes

the path for the group). The admin also needs to update the policy attached to Bob to use the new name (or new path) so that Bob can continue to add and remove users from the group.

AWS Management Console

To change a group's name

1. In the **Navigation** pane, click **Groups**, and then select the group name.
2. From the **Group Actions** list, select **Edit Group Name**.



3. Enter the new group name, and then click **Yes, Edit**.

Command Line Tools

The `iam-groupmod` command lets you change the group's name or path. For more information about the command, go to the [AWS Identity and Access Management Command Line Interface Reference](#).

API

The `UpdateGroup` action lets you change the group's name or path. For more information about the action, go to the [AWS Identity and Access Management API Reference](#), or refer to your SDK's documentation.

Managing Passwords

Topics

- [Changing Your AWS Account Password \(p. 96\)](#)
- [Managing an IAM Password Policy \(p. 96\)](#)
- [Creating, Changing, or Deleting a Password for an IAM User \(p. 100\)](#)
- [Granting IAM Users Permission to Change Their Own Password \(p. 104\)](#)
- [How IAM Users Change Their Own Password \(p. 105\)](#)

To access your AWS account resources from the [AWS Management Console](#), an IAM user must have a password (formerly referred to as a *login profile*). This topic explains how to create, change, and delete a user's password. It also shows how to change the AWS account password, how you can enable users to change their own passwords, and how to set the password policy for users of your AWS account.

Note

Users who access your services only through the API or command line interface do not need a password.

For information about how users access your account sign-in page, see [How Users Sign In to the AWS Management Console \(p. 190\)](#).

Changing Your AWS Account Password

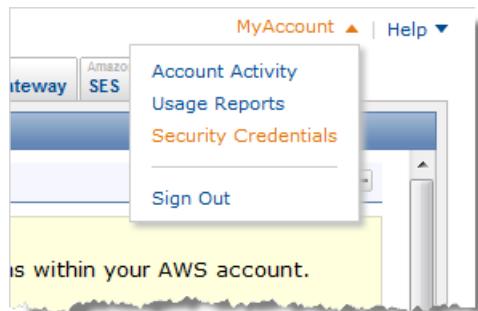
To change the password for your AWS account

1. Use your AWS account email address and password to sign in to the [AWS Management Console](#).

Note

If you previously signed in to the console with your IAM user credentials, your browser may remember this preference and open your account-specific sign-in page by default. You cannot use this page to sign in with your root credentials (your email address and password). In this case, to access the regular AWS sign-in page, click **Sign in using AWS Account credentials** near the bottom of the page.

2. In the upper right corner of the console, click the arrow next to the account name or number and then click **Security Credentials**.



3. On the AWS Security Credentials page, under **Sign-in Credentials**, click the link at: *If you'd like to change your e-mail address or password now, [click here](#).*

click here'. This link is circled in red. At the bottom, there is a note: 'For your protection, do not share your password with anyone. Industry best practice recommends frequent'."/>

Sign-In Credentials

To sign in to AWS web sites and applications, AWS requires your Amazon e-mail address and password. Additionally, it supports the AWS Multi-Factor Authentication option. Each sign-in credential is explained below.

Amazon E-mail Address and Password

To sign in to secure pages on the AWS web site, the AWS Management Console, the AWS Discussion Forums, and the AWS Premium Support site, you need to provide your Amazon e-mail address and password.

Email Address: myaccount@example.com
Password: *****

If you'd like to change your e-mail address or password now, [click here](#).

For your protection, do not share your password with anyone. Industry best practice recommends frequent

4. Follow the on-screen instructions.

Managing an IAM Password Policy

Topics

- [Granting an IAM User Permission to Manage the Password Policy \(p. 97\)](#)
- [Displaying, Creating, Changing, or Deleting a Password Policy \(AWS Management Console\) \(p. 97\)](#)
- [Displaying, Creating, Changing, or Deleting a Password Policy \(API and CLI\) \(p. 99\)](#)

Use password policies to enforce creation of strong passwords by your IAM users. A password policy sets the rules that apply when users change their password. For example, your password policy might require that a password include both uppercase and lowercase letters, or that it include numbers. When you set a password policy, as part of that process, you can also enforce an account-wide policy that grants permission to all users to change their own password.

When you create or change a password policy, the change is enforced immediately when users change their passwords. IAM will not force users to change pre-existing passwords.

The IAM password policy does not apply to your AWS root account password.

Note

If you do not want to allow all users to change their password, you can use an IAM policy to grant only some users permission to change their password. For information about using an IAM policy to grant users permission to change their own password, see [Granting IAM Users Permission to Change Their Own Password \(p. 104\)](#).

For enhanced security, use password policies together with multi-factor authentication (MFA). For more information about MFA, see [Using Multi-Factor Authentication \(MFA\) Devices with AWS \(p. 106\)](#).

Granting an IAM User Permission to Manage the Password Policy

To grant a user permission to manage the password policy on your AWS account, you attach an IAM policy to the user (or to a group of users) that grants the user access to the IAM password policy APIs. This policy allows users to use the AWS Management Console to change the password policy. It also enables users to call the corresponding API actions directly, and to use the corresponding CLI commands.

The following policy is an example of what your policy might look like.

```
{  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "iam:*AccountPasswordPolicy*",  
      "Resource": "*"  
    }  
  ]  
}
```

For information about attaching policies to users or groups, see [Managing IAM Policies \(p. 173\)](#).

Displaying, Creating, Changing, or Deleting a Password Policy (AWS Management Console)

Topics

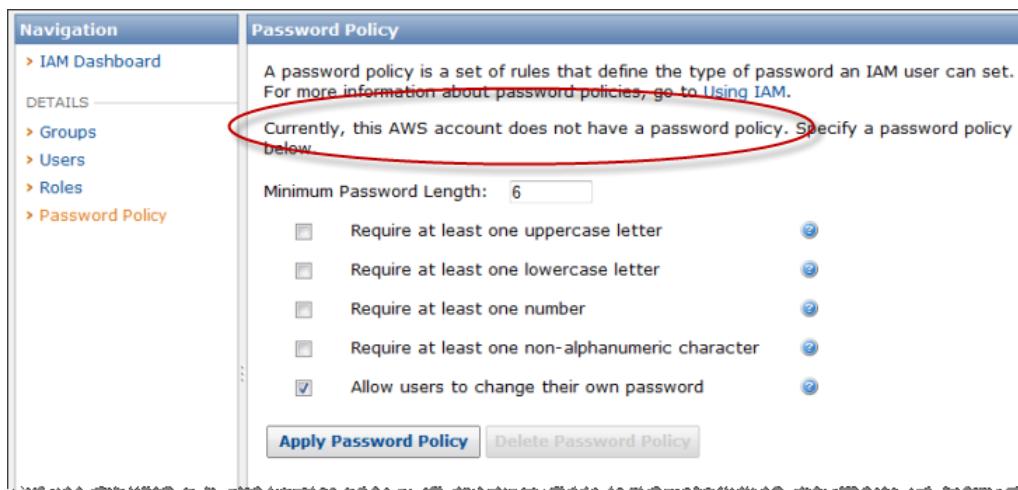
- [Displaying, Creating, or Changing the Password Policy \(p. 98\)](#)
- [Deleting a Password Policy \(p. 98\)](#)

You can use the AWS Management Console to display, create, change, or delete a password policy. For general information about password policies and how they work, see the overview topic [Managing an IAM Password Policy \(p. 96\)](#).

Displaying, Creating, or Changing the Password Policy

To display, create, or change a password policy

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. To display your current password policy, on the **Navigation** pane of the console, click **Password Policy**. If you do not have a password policy, you will see a message indicating that a password policy hasn't been set for your AWS account.

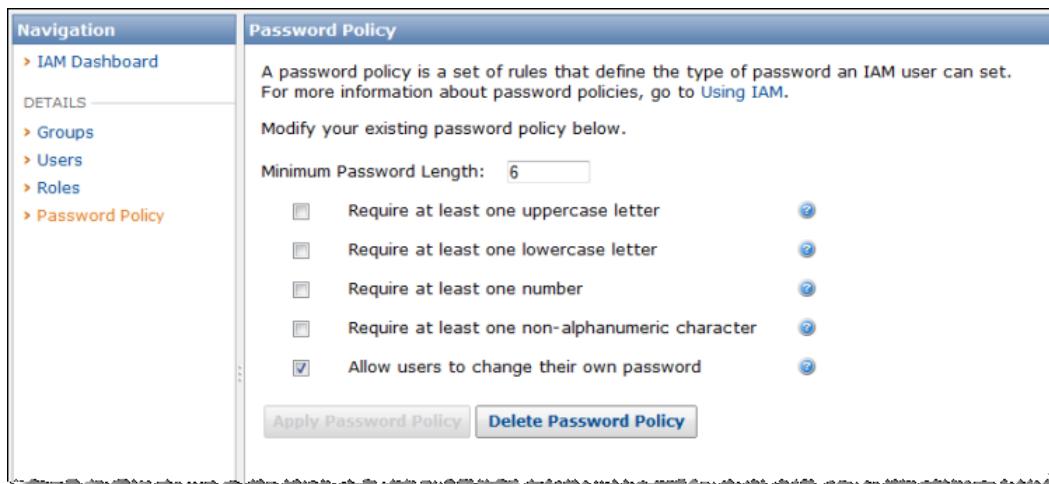


3. To create a new password policy or to change your existing policy, select the check box for the rules you want to apply to your password policy. You can specify the minimum length for the password, and you can require that passwords contain at least one of certain types of characters. If you require that passwords contain non-alphanumeric characters, users must use at least one of the following characters: ! @ # \$ % ^ & * () _ + - = [] { } | '.
4. To enable all your users to change their password, make sure **Allow users to change their own password** is checked. If you prefer to enable only some users to change their password, clear this option. For more information about enabling users to change their own password, see [Granting IAM Users Permission to Change Their Own Password \(p. 104\)](#).
5. To save your changes, click **Apply Password Policy**.

Deleting a Password Policy

To delete a password policy

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the **Navigation** pane, click **Password Policy**, and then click **Delete Password Policy**.



Displaying, Creating, Changing, or Deleting a Password Policy (API and CLI)

Topics

- [Using the API to Create, Change, Display, or Delete a Password Policy \(p. 99\)](#)
- [Using the CLI to Create, Change, Display, or Delete an IAM User Password \(p. 99\)](#)

You can use the IAM API or CLI to manage the password policy for your AWS account. These topics show the API actions and CLI commands you use to complete these tasks.

Using the API to Create, Change, Display, or Delete a Password Policy

Action	Use
UpdateAccountPasswordPolicy	Create or change a password policy.
GetAccountPasswordPolicy	Display the current password policy.
DeleteAccountPasswordPolicy	Delete a password policy.

For more information about the actions, go to the [AWS Identity and Access Management API Reference](#).

Using the CLI to Create, Change, Display, or Delete an IAM User Password

Command	Use
iam-accountmodpasswordpolicy	Create or change a password policy.
iam-accountgetpasswordpolicy	Display a password policy.
iam-accountdelpasswordpolicy	Delete a password policy.

For more information about the commands, go to the [AWS Identity and Access Management Command Line Interface Reference](#).

Creating, Changing, or Deleting a Password for an IAM User

Topics

- [Creating, Changing, or Deleting an IAM User Password \(AWS Management Console\) \(p. 100\)](#)
- [Creating, Changing, or Deleting an IAM User Password \(API and CLI\) \(p. 103\)](#)

IAM users who access your AWS resources from the AWS Management Console must have a password in order to sign in. This topic explains how to create, change, or delete a password for a user under your AWS account. To complete these tasks, you can use the AWS Management Console, the IAM API, or the command line interface.

After you have created passwords for your users, you can grant them permission to change their own passwords. For more information, see [Granting IAM Users Permission to Change Their Own Password \(p. 104\)](#). For information about how users access your account sign-in page, see [How Users Sign In to the AWS Management Console \(p. 190\)](#).

Note

Users who access your services only through the API or command line interface do not need a password.

Creating, Changing, or Deleting an IAM User Password (AWS Management Console)

Topics

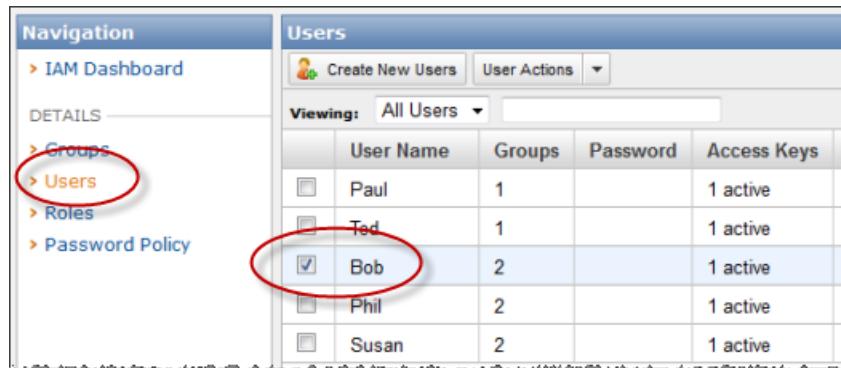
- [Creating or Changing an IAM User Password \(p. 100\)](#)
- [Deleting an IAM User Password \(p. 102\)](#)

You can use the AWS Management Console to create, change, or delete a password for an IAM user.

Creating or Changing an IAM User Password

To create or change a password for a user

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the **Navigation** pane, click **Users**. In the **Users** pane, click the user whose password you want to create or change.



3. In the user details pane, click the **Security Credentials** tab, and then click **Manage Password**.

AWS Identity and Access Management Using IAM Creating, Changing, or Deleting a Password for an IAM User

The screenshot shows the AWS IAM User Details page for a user named Bob. The 'Security Credentials' tab is active. In the 'Access Credentials' section, there is one access key listed with the status 'TXQ - Active' and a creation date of '2011-10-05 12:16 PDT'. There is a 'Manage Access Keys' button. In the 'Sign-In Credentials' section, the 'User name' is 'Bob' and the 'Password' is 'No'. There is a 'Manage Password' button, which is circled in red. Other sections visible include 'Signing Certificates' (None) and 'Multi-Factor Authentication Device' (No).

4. You can create a custom password, or you can have IAM automatically generate a password.

Important

When you create the password, you should save it in a safe place. For security reasons, neither you nor the user will be able to retrieve the password after this step.

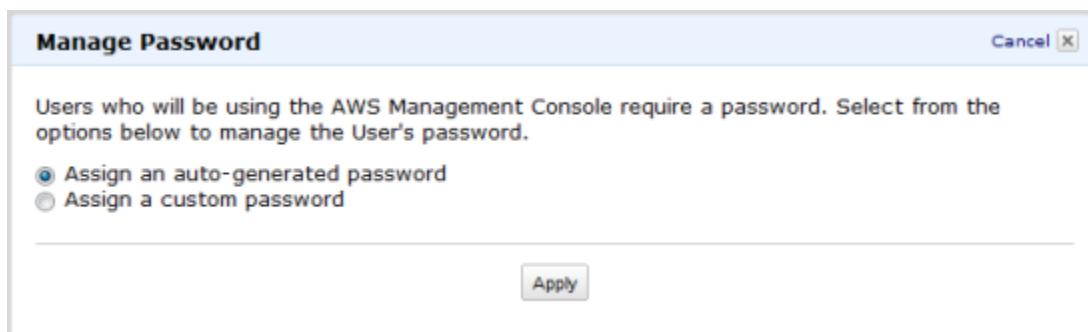
- To create a custom password, follow these steps:
 - a. In the **Manage Password** dialog box, click **Assign a custom password** (or **Replace existing password with a new custom password** if this is a replacement password). In the boxes provided, enter and confirm the password, and then click **Apply**.

The 'Manage Password' dialog box is shown. It contains two radio buttons: 'Assign an auto-generated password' (unchecked) and 'Assign a custom password' (checked). Below the radio buttons are two input fields: 'Password' and 'Confirm password', both containing the placeholder text 'Enter password here...'. At the bottom right is an 'Apply' button.

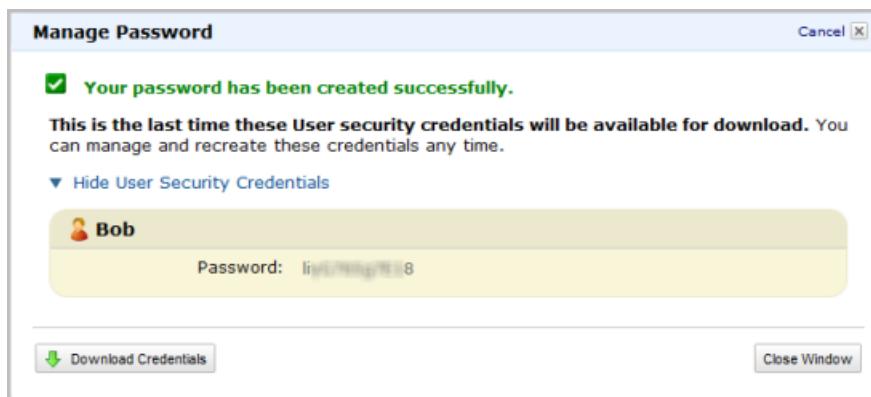
The password must conform to your password policy, if you have one configured. For more information about password policies, see [Managing an IAM Password Policy \(p. 96\)](#).

- To have IAM generate a password, follow these steps:
 - a. In the **Manage Password** dialog box, click **Assign an auto-generated password** (or **Replace existing password with new auto-generated password** if this is a replacement password), and then click **Apply**.

AWS Identity and Access Management Using IAM Creating, Changing, or Deleting a Password for an IAM User



- b. Click **Download Credentials** to save the password as a .CSV file to your computer.



After you create the password, you must provide the user with the following information so the user can sign in from your account sign-in page. IAM does not send this information to the user.

- The user's name and password
- The URL to your account sign-in page

For information on accessing the account sign-in page, see [How Users Sign In to the AWS Management Console \(p. 190\)](#).

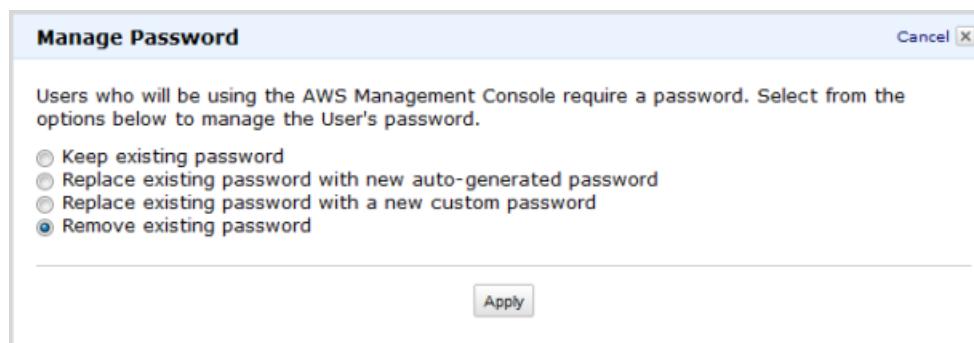
Note

Even if your users have a password, they still need permission to access your AWS resources. By default, a new user has no permissions. To give your users the permissions they need, you assign policies to them or to the groups they belong to. For information about creating users and groups, see [Working with Users and Groups \(p. 72\)](#). For information about using policies to set permissions, see [Permissions and Policies \(p. 30\)](#).

Deleting an IAM User Password

To delete a user's password

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the **Navigation** pane, click **Users**. In the **Users** pane, click the user whose password you want to delete, and then click **Manage Password**.
3. Click **Remove existing password**, and then click **Apply**.



Note

Without a password, a user cannot sign in to AWS website features such as the AWS Management Console or the AWS forums.

Creating, Changing, or Deleting an IAM User Password (API and CLI)

Topics

- [Using the API to Create, Change, or Delete an IAM User Password \(p. 103\)](#)
- [Using the CLI to Create, Change, or Delete an IAM User Password \(p. 103\)](#)

You can use the IAM API or CLI to manage user passwords. These topics show the API actions and CLI commands you use to complete these tasks.

Note

If you use the API to delete a user from your AWS account, you must delete the password as a separate step in the process of removing the user. For more information about deleting a user, see [Deleting a User from Your AWS Account \(p. 82\)](#).

Using the API to Create, Change, or Delete an IAM User Password

Action	Use
CreateLoginProfile	Create a password for a user.
UpdateLoginProfile	Change a user's password.
DeleteLoginProfile	Delete a user's password.

For more information about the actions, go to the [AWS Identity and Access Management API Reference](#).

Using the CLI to Create, Change, or Delete an IAM User Password

Command	Use
iam-useraddloginprofile	Create a password for a user.
iam-usermodloginprofile	Change a user's password.
iam-userdelloginprofile	Delete a user's password.

For more information about the commands, go to the [AWS Identity and Access Management Command Line Interface Reference](#).

Granting IAM Users Permission to Change Their Own Password

When you set a password policy, as part of that process, you can also set an account-wide policy that grants permission to all of your IAM users to change their own password. If you prefer to allow only some users to change their password, you can create a group for your users and then use an IAM policy to grant permission to the group. This topic describes how to use an IAM policy to grant permission to a group of users to change their own password. For information about enabling all of your users to change their own password when setting the account-wide password policy, see [Managing an IAM Password Policy \(p. 96\)](#).

Important

Setting a password policy is recommended because it enforces creation of strong passwords by your users.

To grant a group of users permission to change their password

1. Create a group and add the users whom you want to grant permission to. For more information, see [Creating and Listing Groups \(p. 85\)](#).
2. Create an IAM policy that grants users permission to change their password and apply that policy to the new group.

Your policy might look like the following policy. This policy enables users to call `ChangePassword` directly, to use the corresponding `iam-userchangepassword` CLI command, and to use the AWS Management Console to change their password themselves. This policy also allows users to call the `GetAccountPasswordPolicy` action, which enables users to view the applicable password policy.

```
{  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iam:ChangePassword",  
                "iam:GetAccountPasswordPolicy"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

For information about attaching a policy to a group, see [Managing IAM Policies \(p. 173\)](#).

3. Next, create a new password policy or modify your existing password policy, and make sure that the **Allow users to change their own password** check box is *not* checked. For more information about creating or viewing a password policy, see [Managing an IAM Password Policy \(p. 96\)](#).

Note

You can enable users to change their password without setting a password policy, but it is not recommended. To enforce use of strong passwords by your users, you should use a password policy when you enable users to manage their own password.

Related topics

- [Managing an IAM Password Policy \(p. 96\)](#)
- [Managing IAM Policies \(p. 173\)](#)
- [How IAM Users Change Their Own Password \(p. 105\)](#)
- [Creating, Changing, or Deleting a Password for an IAM User \(p. 100\)](#)

How IAM Users Change Their Own Password

To change their password, users can use the AWS Management Console, the IAM API, or the command line interface. This topic describes how a user uses the AWS Management Console to change a password. The IAM APIs and CLI commands your users need to change their passwords are listed at the end of this topic.

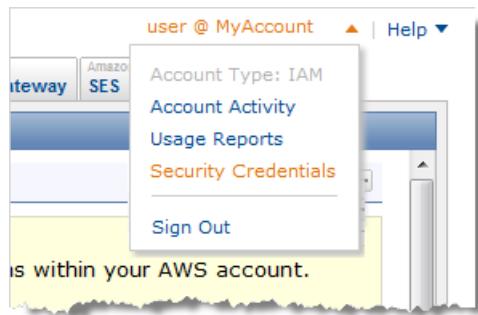
The following procedure shows how users change their own passwords. You should provide this information to your users after you grant them permission to change their passwords. For information about how to grant users permission to change their passwords, see the previous topic, [Granting IAM Users Permission to Change Their Own Password \(p. 104\)](#).

To change your IAM user password

1. Use your user name and password to sign in to your company's AWS Management Console sign-in page.

To obtain the URL for your company's sign-in page, contact your administrator. For more information about using a sign-in page to access the AWS Management Console, see [The AWS Management Console Sign-in Page \(p. 190\)](#).

2. In the upper right corner of the console, click the arrow next to your user name, and then click **Security Credentials**.



3. On the **My Password** pane, in the **Old Password** box, type your password.

The screenshot shows a web-based password change interface titled "My Password". A message at the top states: "Use this form to change your password. Your new password must meet the requirements defined in the AWS accounts password policy." Below this, instructions say: "Your new password: • must not be the same as your old password • must be at least 6 characters long". There are three input fields: "Old Password", "New Password", and "Confirm New Password". A "Change Password" button is located below the "Confirm New Password" field.

4. In the **New Password** box, type your new password. In the **Confirm New Password** box, retype your new password, and then click **Change Password**.

Note

The requirements for your password, such as minimum length and whether your password must contain a number, are defined by the password policy settings selected by your administrator.

Related API Actions and CLI Commands

To change their own password, users can use the `ChangePassword` API action or the `iam-userchangepassword` CLI command. For more information, see [ChangePassword](#) in the *AWS Identity and Access Management API Reference*, or [iam-userchangepassword](#) in the *AWS Identity and Access Management Command Line Interface Reference*.

Using Multi-Factor Authentication (MFA) Devices with AWS

Topics

- [Setting Up an MFA Device \(p. 107\)](#)
- [Checking MFA Status \(p. 107\)](#)
- [Using a Virtual MFA Device with AWS \(p. 108\)](#)
- [Enabling a Hardware MFA Device for Use with AWS \(p. 114\)](#)
- [Synchronizing an MFA Device \(p. 117\)](#)
- [Deactivating an MFA Device \(p. 118\)](#)
- [Configuring MFA-Protected API Access \(p. 120\)](#)
- [What If an MFA Device Is Lost or Stops Working? \(p. 124\)](#)

For increased security, we recommend that you protect your AWS resources by configuring AWS Multi-Factor Authentication (MFA). MFA adds extra security by requiring users to enter a unique authentication code from their authentication device when accessing AWS websites or services.

For MFA to work, you must assign an MFA device (hardware or virtual) to the IAM user or root account. The MFA device must be unique for each user; a user cannot enter a code from another user's device to authenticate.

To get started setting up an MFA device for root account or IAM user access to the console, see [Setting Up an MFA Device \(p. 107\)](#).

To set up MFA-protected API access for IAM users with an enabled MFA device, see [Configuring MFA-Protected API Access \(p. 120\)](#).

Setting Up an MFA Device

This section shows you how to set up and enable a new MFA device, as well as how to synchronize and deactivate existing devices, and what to do when a device is lost or stops working. For answers to commonly asked questions about AWS MFA, go to the [AWS Multi-Factor Authentication FAQs](#).

The following high-level procedure describes how to set up and use an MFA device, and provides links to related information.

1. *Get an MFA device.* The device can be a hardware MFA device or a virtual MFA device. A virtual device can be a smartphone that has an MFA application installed on it.

If you want to use a hardware device, you can find information about where to purchase the devices that AWS supports at <http://aws.amazon.com/mfa>. If you want to use a virtual MFA device, or if you just want to learn more about virtual MFA, see [Using a Virtual MFA Device with AWS \(p. 108\)](#).

2. *Enable the MFA device.* You can enable the MFA device for use with AWS using the AWS Management Console, the IAM command line tools, or the IAM API.

For information about enabling an MFA device, see either [Using a Virtual MFA Device with AWS \(p. 108\)](#) or [Enabling a Hardware MFA Device for Use with AWS \(p. 114\)](#).

3. *Use the MFA device when logging on or accessing AWS resources.* For access to an AWS website, you need a username, password, and MFA code. For access to MFA-protected APIs, you need access keys and an MFA code.

For information about user passwords, see [Managing Passwords \(p. 95\)](#). For information about using MFA with the AWS Management Console, see [MFA Devices and Your IAM-Enabled Sign-in Page \(p. 194\)](#). For information about the AWS service APIs that use MFA, go to [Does AWS MFA affect how I access AWS Service APIs?](#) on the [AWS Multi-Factor Authentication FAQs](#) page.

Checking MFA Status

Use the **IAM** console to verify that an MFA device is enabled and configured for the root account or IAM user. In this section, you'll learn how to check whether a root account or IAM user has a valid MFA device enabled.

To check the MFA status of a root account

1. Open the **IAM** console at <https://console.aws.amazon.com/iam>.
2. In the **IAM Dashboard**, check under **Security Status** to see whether MFA is enabled or disabled.



- Click **Manage MFA Device** to change the current setting.

To check the MFA status of an IAM user

- Open the **IAM** console at <https://console.aws.amazon.com/iam>.
- Select **Users**.
- Select the user.

In the **Security Credentials** tab of the user's details pane, if the user has an MFA device enabled, the **Multi-Factor Authentication Device** item shows a value for the device.



The **Multi-Factor Authentication Device** is either a value for a virtual device, such as `arn:aws:iam::123456789012:mfa/user`, or it is the device serial number for a hardware device (usually the number from the back of the device), such as `GAHT12345678`.

- Click **Manage MFA Device** to change the current setting.

For virtual device information, see [Using a Virtual MFA Device with AWS \(p. 108\)](#).

For hardware device information, see [Enabling a Hardware MFA Device for Use with AWS \(p. 114\)](#).

Using a Virtual MFA Device with AWS

Topics

- Configuring and Enabling a Virtual MFA Device for a User (p. 109)
- Configuring and Enabling a Virtual MFA Device for Your AWS Account (p. 111)
- Using the IAM CLI or API with Virtual MFA Devices (p. 113)

- [Installing the AWS Virtual MFA Mobile Application \(p. 114\)](#)

To use a virtual MFA device with AWS, you must configure it for use with AWS, and then enable it. In this section you'll learn what a virtual MFA device is and what you need to do to configure and enable it.

A virtual MFA device uses a software application that generates six-digit authentication codes that are compatible with the Time-Based One-Time Password (TOTP) standard, as described in [RFC 6238](#). The software application can run on any mobile hardware device, including a smartphone. Most virtual MFA applications allow you to host more than one virtual MFA device, which makes them more convenient than hardware MFA devices. However, you should be aware that because a virtual MFA might be run on a less secure device such as a smartphone, a virtual MFA might not provide the same level of security as a hardware MFA device.

Tip

The AWS Virtual MFA application is one example of an TOTP-compatible virtual MFA device. The AWS Virtual MFA application runs on the Android mobile operating system and is available for download from the [Amazon Appstore for Android](#). For more information about downloading and installing the AWS Virtual MFA, see [Installing the AWS Virtual MFA Mobile Application \(p. 114\)](#).

Configuring and Enabling a Virtual MFA Device for a User

You can use IAM in the AWS Management Console to configure and enable a virtual MFA device for a user under your account. This section shows you how to use the console to complete these tasks. If you prefer to use the CLI or API, see [Using the IAM CLI or API with Virtual MFA Devices \(p. 113\)](#).

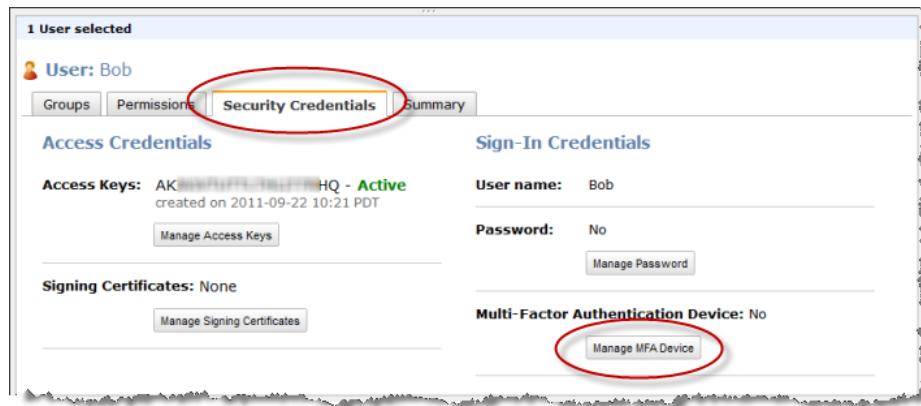
Note

You must have physical access to the user's MFA device in order to configure MFA. For example, if you are configuring MFA for a user who will use a smartphone to generate an OTP, you must have the smartphone available in order to finish the wizard.

The following procedure uses the AWS Virtual MFA mobile application as an example. However, the same general steps apply regardless of the virtual MFA application you use.

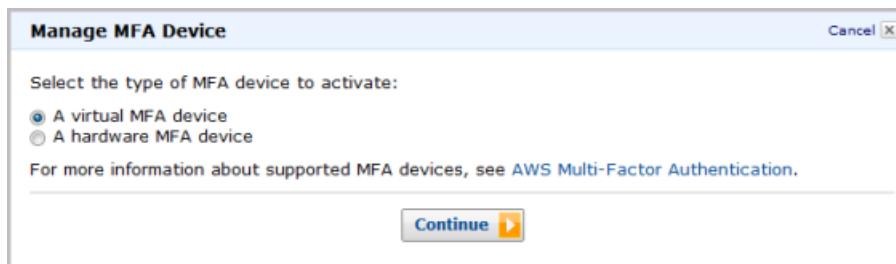
To configure and enable a virtual MFA device for a user

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the **Navigation** pane, click **User** and then select the user you want to enable the virtual MFA for.
3. In the user details pane, select **Security Credentials**, and then click **Manage MFA Device**. This starts the **Manage MFA Device** wizard.



4. In the wizard, select **A virtual MFA device** and then click **Continue**.

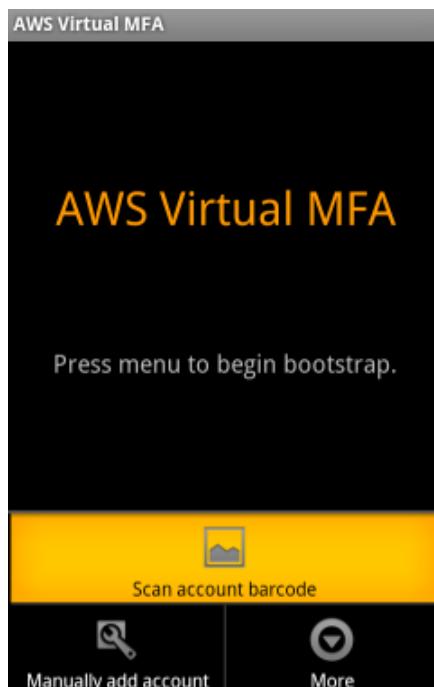
AWS Identity and Access Management Using IAM Using a Virtual MFA Device with AWS



5. Confirm that a virtual MFA application is installed on the user's mobile device and then click **Continue**. IAM generates and displays configuration information for the virtual MFA device, including a QR code similar to the following graphic.



6. With the **Manage MFA Device** wizard still open, open the virtual MFA application on the device. If the device supports QR codes, the easiest way to configure the application is to use the application to scan the QR code. If you cannot scan the code, you can enter the secret configuration key manually.
 - To use the QR code to configure the virtual MFA device, in the AWS Virtual MFA application, tap **Scan account barcode**, and then use the device's camera to scan the code. If you are using a different application on the device, use the equivalent command.



- If you cannot scan the code, enter the configuration information manually by typing the **Secret Configuration Key** value into the application. To do this in the AWS Virtual MFA application, tap **Manually add account**, and then type the secret configuration key and click **Create**.

Note

The QR code and secret configuration key are unique and cannot be reused.

When you are finished configuring the device, the device starts generating six-digit numbers.

7. In the **IAM Manage MFA Device** wizard, in the **Authentication Code 1** box, type the six-digit number that's currently displayed by the MFA device. Wait 30 seconds for the device to generate a new number, and then type the new six-digit number into the **Authentication Code 2** box.



8. Click **Activate Virtual MFA**.

The device is ready for use with AWS. For information about using MFA with the AWS Management Console, see [MFA Devices and Your IAM-Enabled Sign-in Page \(p. 194\)](#).

Configuring and Enabling a Virtual MFA Device for Your AWS Account

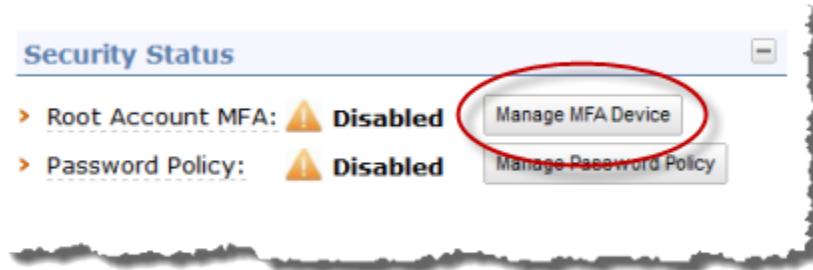
You can use IAM in the AWS Management Console to configure and enable a virtual MFA device for your root account. This section describes how to use the console to complete these tasks. If you prefer to use the CLI or API, see [Using the IAM CLI or API with Virtual MFA Devices \(p. 113\)](#).

To manage MFA devices for the AWS account, you must be signed in to AWS using your root account credentials. You cannot manage MFA devices for the root account using other credentials.

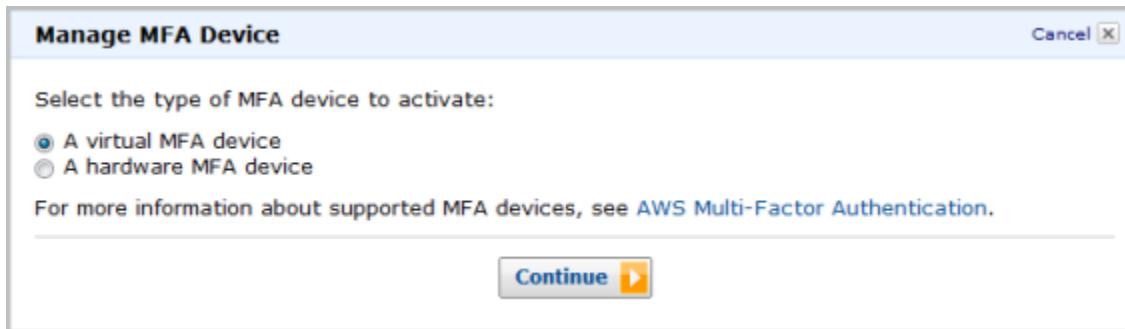
The following procedure uses the AWS Virtual MFA mobile application as an example. However, the same general steps apply regardless of the virtual MFA application you use.

To configure and enable a virtual MFA device for use with your root account

1. Use your root credentials to sign in to the [AWS Management Console](#), and then go to the **IAM** console.
2. On the **IAM Dashboard**, click **Manage MFA Device**. This starts the **Manage MFA Device** wizard.



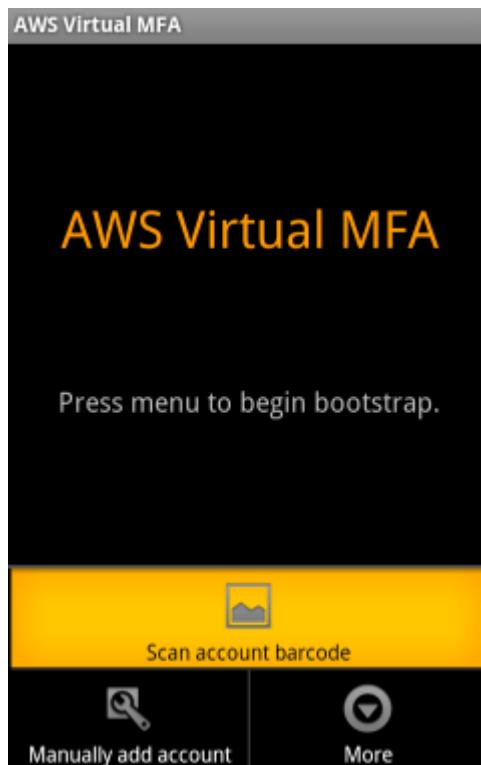
3. In the wizard, select **A virtual MFA device** and then click **Continue**.



4. Confirm that a virtual MFA application is installed on the device, then click **Continue**. IAM generates and displays configuration information for the virtual MFA device, including a QR code similar to the following graphic.



5. With the **Manage MFA Device** wizard still open, open the virtual MFA application on the device. The easiest way to configure the application is to use the application to scan the QR code. If you cannot scan the code, you can enter the configuration information manually.
 - To use the QR code to configure the virtual MFA device, in the AWS Virtual MFA application, tap **Scan account barcode**, and then use the device's camera to scan the code. If you are using a different application on the device, use the equivalent command.



- If you cannot scan the code, you can enter the configuration information manually by typing the **Secret Configuration Key** value into the application. To do this in the AWS Virtual MFA application, tap **Manually add account**, then type the secret configuration key and click **Create**.

Note

The QR code and secret configuration key are unique and cannot be reused.

When you are finished configuring the device, the device starts generating six-digit numbers.

6. In the **IAM Manage MFA Device** wizard, click the link to enable the device.
7. The **Enable AWS Multi-Factor Authentication** page opens on the AWS portal. (You might be prompted to sign in again before you can access this page.) In the **Authentication Code 1** box, type the six-digit number displayed by the MFA device. Wait 30 seconds while the device refreshes, and then type the second number into **Authentication Code 2**.
8. Click **Activate Authentication Device**.

The device is ready for use with AWS. For information about using MFA with the AWS Management Console, see [MFA Devices and Your IAM-Enabled Sign-in Page \(p. 194\)](#).

Using the IAM CLI or API with Virtual MFA Devices

The following list shows the command line commands or API actions to use to configure and enable a virtual MFA device. Use these commands in the following order.

1. Task: Create the configuration information for the virtual MFA device.

CLI command: [iam-virtualmfadevicecreate](#)

API action: [CreateVirtualMFADevice](#)

2. Task: Enable the device for use with AWS.

CLI command: [iam-userenablemfadevice](#)

API action: [EnableMFADevice](#)

The following list shows additional commands and actions related to virtual MFA.

- Task: Deactivate a device.

CLI command: [iam-userdeactivatemfadevice](#)

API action: [DeactivateMFADevice](#)

- Task: List virtual MFA devices.

CLI command: [iam-virtualmfadevicelist](#)

API action: [ListVirtualMFADevices](#)

- Task: Re-sync an MFA device.

CLI command: [iam-userresyncmfadevice](#)

API action: [ResyncMFADevice](#)

- Task: Delete a virtual MFA device.

CLI command: [iam-virtualmfadevicedel](#)

API action: [DeleteVirtualMFADevice](#)

Installing the AWS Virtual MFA Mobile Application

To download and install the AWS Virtual MFA application, go to the [Amazon Appstore for Android](#) and locate the AWS Virtual MFA application. Download the application and follow the on-screen instructions to complete the installation.

For information about configuring and enabling the AWS Virtual MFA device for use with AWS, see [Using a Virtual MFA Device with AWS \(p. 108\)](#).

To open the [Amazon Appstore for Android](#) on a smartphone, scan this code with any QR code reader application.



Enabling a Hardware MFA Device for Use with AWS

You can enable a hardware MFA device using the AWS Management Console, the command line, or the IAM API. The following procedure shows you how to use the AWS Management Console to enable the device for a user under your AWS account. To enable an MFA device for your root account, see [Enabling a Hardware MFA Device for Your AWS Root Account \(p. 116\)](#).

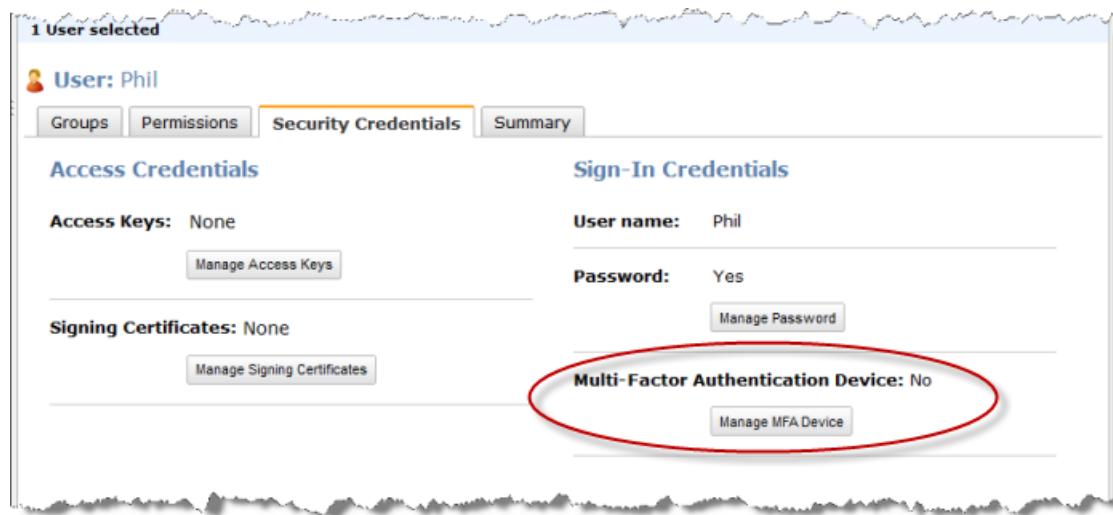
Note

If you want to enable the device from the command line, use `iam-userenablemfadevice`, described in the [AWS Identity and Access Management Command Line Interface Reference](#). To enable the MFA device with the IAM API, use the `EnableMFADevice` action, described in the [AWS Identity and Access Management API Reference](#).

Enabling a User's Hardware MFA Device

To use IAM in the AWS Management Console to enable a hardware MFA device for a user

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. On the Navigation pane, choose **Users**.
3. Select the user you want to enable an MFA device for, and then click **Manage MFA Device**.



4. Enter the device serial number. The serial number is usually on the back of the device.

The dialog box has a title 'Manage Multi-Factor Authentication Device' and a 'Cancel' button. It contains instructions: 'Fill in the fields below to associate a Multi-Factor Authentication (MFA) Device with user Phil.' There is a link 'Show Instructions for associating a MFA device'. Below are three input fields: 'Serial Number:' with an empty text box, 'Authentication Code 1:' with an empty text box, and 'Authentication Code 2:' with an empty text box. At the bottom are 'Cancel' and 'Associate MFA' buttons.

5. In the **Authentication Code 1** box, type the six-digit number displayed by the MFA device. You might need to press the button on the front of the device to display the number.



6. Wait 30 seconds while the device refreshes, and then type the next six-digit number into the **Authentication Code 2** box. You might need to press the button on the front of the device again to display the second number.
7. Click **Associate MFA**.

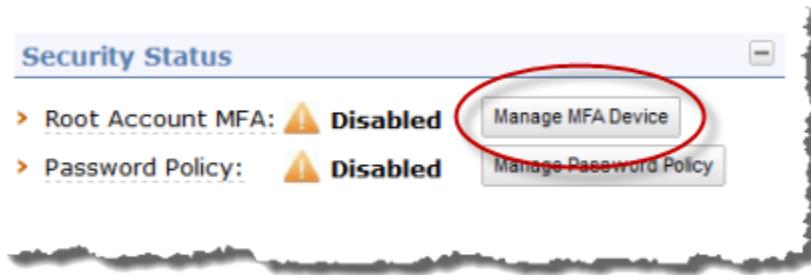
The device is ready for use with AWS. For information about using MFA with the AWS Management Console, see [MFA Devices and Your IAM-Enabled Sign-in Page \(p. 194\)](#).

Enabling a Hardware MFA Device for Your AWS Root Account

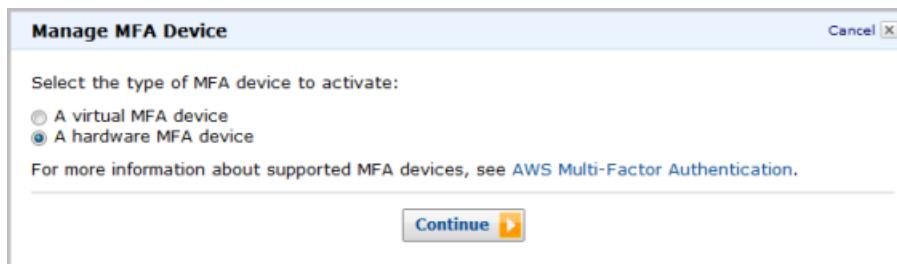
To enable the MFA device for your AWS account

1. Use your root credentials to sign in to the AWS Management Console, then go to the **IAM** console.

Important
To manage MFA devices for the AWS account, you must sign in to AWS using your root account credentials. You cannot manage MFA devices for the root account using other credentials.
2. From the **IAM Dashboard**, click **Manage MFA Device**. This starts the **Manage MFA Device** wizard.



3. In the wizard, select **A hardware MFA device**, and then click **Continue**.



4. To complete the remainder of this process, you need to go to the AWS portal. To go to the portal, click **Click here to enable your device**.
5. In the **Serial Number** box, enter the serial number displayed on the back of the MFA device. Re-enter the serial number in the **Re-Enter Serial Number** box.

Enter Serial Number and Authentication Codes

In order to start using your Gemalto Ezio authentication device it must be activated with Amazon Web Services using the hardware's serial number and two consecutive authentication codes.

Enter the serial number that is displayed on the back of your hardware.

Serial Number:

Re-Enter Serial Number:

Press the button on your authentication device and enter the currently displayed authentication code.

Authentication code #1:

Wait until the display clears press the button on your authentication device again and enter the updated authentication code below.

Authentication code #2:

Activate Authentication Device

6. In the **Authentication Code 1** box, type the six-digit number displayed by the MFA device. You might need to press the button on the front of the device to display the number.



7. Wait 30 seconds while the device refreshes, and then type the next six-digit number into the **Authentication Code 2** box. You might need to press the button on the front of the device again to display the second number.
8. Click **Activate Authentication Device**. The MFA device is now associated with the AWS Account.
9. In the AWS Management Console, click **Close** to close the confirmation dialog box.

The next time anyone signs in using the AWS account credentials, they will need to enter a code from the MFA device.

For information about using MFA with the AWS Management Console, see [MFA Devices and Your IAM-Enabled Sign-in Page \(p. 194\)](#).

Synchronizing an MFA Device

It's possible for an MFA device to get out of synchronization. If the device is not synchronized when the user tries to use it, the user's login will fail. If the user is using the MFA device to sign in to the AWS Management Console, IAM prompts the user to resynchronize the device.

If the user is using the MFA device with the API, IAM has an API call that performs synchronization. In this case, we recommend that you give your users with MFA devices permission to access this API call. You should build a tool based on that API call that lets your users resynchronize their devices whenever they need to. To use the API to synchronize an MFA device for a user, use [ResyncMFADevice](#). To use the command line to synchronize the device, use [iam-userresyncmfadevice](#).

You can also use the AWS Management Console to resynchronize the device for a user under your AWS account.

To resynchronize a user's MFA device

1. On the AWS Management Console **IAM** console, click **Users**, then select the name of the user you want to synchronize a device for.
2. Click the **Security Credentials** tab, then click **Manage MFA Device**.
3. Select **Resynchronize MFA device**.
4. Type the six-digit number the MFA device is displaying into the **Authentication Code 1** box. If you are using a hardware MFA device, you will need to press the button on the front of the device to display the number.



5. Wait 30 seconds while the device refreshes, and then type the next six-digit number into the **Authentication Code 2** box. If you are using a hardware device, you will need to press the device button again to display the second number.
6. Click **Update**.

Deactivating an MFA Device

You can temporarily disable an MFA device by deactivating it. If you are using the IAM API or CLI, then you deactivate an MFA device with the [IAM DeactivateMFADevice API action](#) or the [iam-userdeactivatemfadvice command](#).

Note

If you use the API or CLI to delete a user from your AWS account, you must deactivate or delete the user's MFA device as part of the process of removing the user. For more information about deleting users, see [Deleting a User from Your AWS Account \(p. 82\)](#).

If you are using the AWS Management Console to deactivate the device for a user under your AWS account, the following procedure describes the steps. The process to deactivate an MFA device for the root account is described in [Deactivating Your AWS Account's MFA Device \(p. 119\)](#).

Deactivating a User's MFA Device

To deactivate a user's MFA device

1. On the **IAM** console, click **Users**, and then select the name of the user for whom you want to deactivate a device.
2. Click the **Security Credentials** tab, and then click **Manage MFA Device**.
3. Select **Deactivate MFA device**.



4. Click **Update**.

The user's device is deactivated.

Deactivating Your AWS Account's MFA Device

You can temporarily deactivate our AWS account's MFA device.

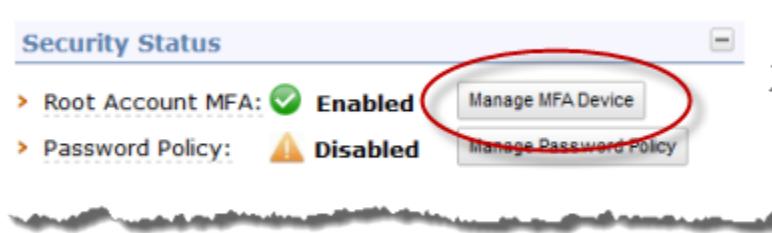
To deactivate the MFA device for your AWS account

1. Use your root credentials to sign in to the AWS Management Console, then go to the **IAM** console.

Note

To manage MFA devices for the AWS account, you must be signed in to AWS using your root account credentials. You cannot manage MFA devices for the root account using other credentials.

2. From the **IAM Dashboard**, click **Manage MFA Device**.



3. Select **Deactivate this device**.



The MFA device is deactivated for the AWS account.

Configuring MFA-Protected API Access

MFA-protected API access is an optional feature that offers an extra layer of security by requiring users to authenticate with an MFA device before they can use APIs you specify in a policy. Users can authenticate through the AWS Management Console or programmatically through the AWS Security Token Service (STS).

MFA-protected API access is available only to services that support temporary security credentials. For a complete list of these services, and for information about using the AWS STS API for requesting temporary security credentials, see [Using Temporary Security Credentials to Access AWS](#).

If the user is denied access to APIs because of an authorization failure, AWS returns an “Access Denied” error message (as it does for any unauthorized access). With MFA-protected API policies in place, AWS denies access to the APIs specified in the policy if the user attempts to use the APIs without valid MFA authentication, or if the time of the request for the APIs is beyond the duration specified in the policy. The user must re-authenticate with MFA, either through the console, or by requesting new temporary security credentials using an MFA code and device serial number.

Note

MFA-protected API access cannot be applied to root accounts accessing their own resources. Federated users cannot be assigned an MFA device for use with AWS services, so they cannot access AWS resources controlled by MFA.

Using MFA-Protected APIs Through the Console

AWS evaluates MFA-protected API policies for actions in the console, such as terminating an Amazon EC2 instance. Set up the IAM user with an MFA device and enable an MFA-protected API policy. The user can then simply log into the console with MFA authentication and is subject to the policies for MFA-protected APIs. For users who already have an assigned MFA device, the console experience doesn't change (except for optional time limits on certain MFA-protected APIs that require more frequent re-authentication). For more information on setting up an IAM user with an MFA device, see [Setting Up an MFA Device \(p. 107\)](#).

Using MFA-Protected APIs Programmatically

MFA-protected API policies also apply to IAM users who access MFA-protected API programmatically, either through their own application or an application provided to them, specifically:

1. An MFA-protected API policy is attached to a user, group, or resource.
2. The IAM user acquires temporary security credentials using an MFA code and device serial number, programmatically, in a request to the AWS Security Token Service (STS). The temporary security credentials include the MFA authentication status.

An application can use the STS API and custom code to prompt the user to provide the MFA code and serial number in the application. For information on implementing MFA authentication using STS, see [Creating Temporary Security Credentials in Using Temporary Security Credentials](#).

3. When the IAM user tries to use an API directly or through an application, AWS uses the condition in the MFA-protected API policy to check whether the IAM user is allowed access to the API.
4. AWS grants the IAM user access to the MFA-protected API (or prevents access) according to the policy.

Policies for MFA-Protected APIs

MFA-protected API policies are attached to a user, group, or resource, such as an Amazon Simple Storage Service (Amazon S3) bucket, Amazon Simple Queue Service (Amazon SQS) queue, or Amazon Simple Notification Service (Amazon SNS) topic.

MFA-protected API policies include a condition statement (or statements) with the `aws:MultiFactorAuthAge` key. The policy can use the key in two ways:

- Existence—To verify that MFA authentication is present, use an existence check with a `Null` statement to evaluate whether the `aws:MultiFactorAuthAge` key exists (is not null, matching the Boolean value "false") or not (is null, matching the Boolean value "true").
- Duration—for situations that require control over the duration of the MFA authentication, independent of the lifetime of the temporary security credentials, use a numeric condition type to compare the key's age to a value (such as 3600 seconds). If a policy checks for existence, only, the access is available for the entire authentication session (the default is 12 hours). A numeric condition enables the policy to restrict access to APIs for a user who does not have recent MFA authentication, even if the user's authorization session is still valid. If necessary, an IAM user can refresh their MFA authentication, either through the console, or by requesting new temporary security credentials using an MFA code and device serial number.

For more information on the condition types for `aws:MultiFactorAuthAge`, see [Existence of Condition Keys \(p. 57\)](#) and [Numeric Conditions \(p. 53\)](#).

The following policies demonstrate several use-cases that require MFA authentication.

Topics

- [Example 1: Granting access after MFA authentication \(p. 121\)](#)
- [Example 2: Granting access after recent MFA authentication \(p. 122\)](#)
- [Example 3: Denying access to specific APIs without valid MFA authentication \(p. 122\)](#)
- [Example 4: Denying access to specific APIs without recent valid MFA authentication \(p. 122\)](#)
- [Example 5: Granting access to a resource after MFA authentication \(p. 123\)](#)

Example 1: Granting access after MFA authentication

The following example shows a policy attached to a user or group that grants Amazon EC2 access only after valid MFA authentication. Specifically, AWS grants access only when the existence check for a `Null` value evaluates to `false`. In other words, the `aws:MultiFactorAuthAge` key value is not null.

```
{  
    "Statement": [  
        {  
            "Action": [ "ec2:*" ],  
            "Effect": "Allow",  
            "Resource": [ "*" ],  
            "Condition": {  
                "Null": { "aws:MultiFactorAuthAge": "false" }  
            }  
        }  
    ]  
}
```

Example 2: Granting access after recent MFA authentication

The following example shows a policy attached to a user or group that grants Amazon EC2 access only after checking for a valid MFA authentication within an hour of the request. Specifically, AWS grants access only when the `aws:MultiFactorAuthAge` key value is present and less than 3600 seconds (1 hour).

```
{  
    "Statement": [ {  
        "Action": [ "ec2:*" ],  
        "Effect": "Allow",  
        "Resource": [ "*" ],  
        "Condition": {  
            "NumericLessThan": { "aws:MultiFactorAuthAge": "3600" }  
        }  
    }  
]
```

Example 3: Denying access to specific APIs without valid MFA authentication

The following example shows a policy attached to a user or group that grants access to the entire Amazon EC2 API, but restricts access to `StopInstances` and `TerminateInstances` until valid MFA authentication is present. Specifically, AWS denies access to `StopInstances` and `TerminateInstances` when the existence check for a `Null` value evaluates to `true`, meaning the `aws:MultiFactorAuthAge` key value is null.

```
{  
    "Statement": [  
        {  
            "Action": [ "ec2:*" ],  
            "Effect": "Allow",  
            "Resource": [ "*" ]  
        },  
        {  
            "Action": [ "ec2:StopInstances", "ec2:TerminateInstances" ],  
            "Effect": "Deny",  
            "Resource": [ "*" ],  
            "Condition": {  
                "Null": { "aws:MultiFactorAuthAge": "true" }  
            }  
        }  
    ]  
}
```

Example 4: Denying access to specific APIs without recent valid MFA authentication

The following example shows a policy attached to a user or group that grants access to the entire Amazon EC2 API, but restricts access to `StopInstances` and `TerminateInstances` unless valid MFA authentication occurred within the last hour. Specifically, AWS denies access to `StopInstances` and `TerminateInstances` when the existence check for a `Null` value evaluates to `true`, meaning the `aws:MultiFactorAuthAge` key value is null. Additionally, AWS denies access to `StopInstances` and

TerminateInstances when aws:MultiFactorAuthAge key value is present and greater than 3600 seconds (1 hour).

Note

MFA-protected API policies using Deny statements that check for the numeric value of aws:MultiFactorAuthAge should include an existence check. AWS evaluates existence and duration independently; evaluating only for duration does not enforce a Deny condition if MFA has not been used at all.

```
{
  "Statement": [
    {
      "Action": ["ec2:*"],
      "Effect": "Allow",
      "Resource": ["*"]
    },
    {
      "Action": ["ec2:StopInstances", "ec2:TerminateInstances"],
      "Effect": "Deny",
      "Resource": ["*"],
      "Condition": {
        "Null": {"aws:MultiFactorAuthAge": "true"}
      }
    },
    {
      "Action": ["ec2:StopInstances", "ec2:TerminateInstances"],
      "Effect": "Deny",
      "Resource": ["*"],
      "Condition": {
        "NumericGreaterThan": {"aws:MultiFactorAuthAge": "3600"}
      }
    }
  ]
}
```

Example 5: Granting access to a resource after MFA authentication

MFA-protected API policies can be attached to a resource, as well, such as Amazon S3 bucket policies. The following policy is attached to an Amazon S3 bucket. This policy grants access to the Amazon S3 actions PutObject and DeleteObject only after valid MFA authentication for all IAM users with access to the bucket. Specifically, AWS grants access only when the existence check for a Null value evaluates to false. In other words, the aws:MultiFactorAuthAge key value is not null.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": ["s3:PutObject", "s3:DeleteObject"],
      "Resource": "arn:aws:s3:::myawsbucket/*",
      "Condition": {
        "Null": {"aws:MultiFactorAuthAge": "false"}
      }
    }
  ]
}
```

]

}

Note

Amazon S3 offers an MFA Delete feature for *root* account (only) access. You can enable Amazon S3 MFA Delete when you set the versioning state of the bucket. Amazon S3 MFA Delete cannot be applied to an IAM user, and is managed independently from MFA-protected API access. An IAM user with permission to delete a bucket cannot delete a bucket with Amazon S3 MFA Delete enabled. For more information on Amazon S3 MFA Delete, see [MFA Delete](#).

What If an MFA Device Is Lost or Stops Working?

If an MFA device stops working, is lost, or is destroyed, and you can't sign in to the AWS portal or the AWS Management Console, then you need to deactivate the device. AWS can help you deactivate the device. The way you get help depends on whether an MFA device is assigned to the root account or to a user under an AWS account.

Note

If the device appears to be functioning properly, but you cannot use it to access your AWS resources, then it simply might be out of synchronization with the AWS system. For information about synchronizing an MFA device, see [Synchronizing an MFA Device \(p. 117\)](#).

To get help for an MFA device associated with an AWS root account

1. Go to the AWS [Contact Us](#) page for help with disabling AWS MFA so that you can temporarily access secure pages on the AWS website and the AWS Management Console using just your user name and password.
2. Change your AWS password in case an attacker has stolen the authentication device and might also have your current password.
3. If you are using a hardware MFA device, contact the third party provider Gemalto [using their website](#) for help fixing or replacing the device. If you the device is a virtual MFA device, delete the old MFA account entry on the device before creating a new one.
4. After you have the new physical MFA device or you have completed deleting the old entry from the mobile device, return to the AWS website and activate the MFA device to re-enable AWS MFA for your AWS account. To manage a hardware MFA for your AWS account, go to **Sign-In Credentials** on the [AWS Security Credentials](#) page.

To get help for an MFA device associated with an IAM user

- Contact the system administrator or other person who gave you the user name and password for the IAM user. They will need to deactivate the MFA device using the procedure described at [Deactivating a User's MFA Device \(p. 118\)](#).

Managing User Keys and Certificates

Topics

- [Creating, Modifying, and Viewing User Security Credentials \(p. 125\)](#)
- [Creating and Uploading a User Signing Certificate \(p. 129\)](#)
- [Adding a Credentials Management Policy to the User \(p. 135\)](#)
- [Rotating Credentials \(p. 136\)](#)

This section describes how to create and manage access keys and signing certificates (also known as X.509 certificates) for IAM users. Each user needs an access key to make programmatic calls to AWS. If the user needs to use the SOAP API or command line tools for Amazon EC2, you must also give the user a signing certificate. For more detailed information about the credentials, see [Security Credentials \(p. 24\)](#) and [Adding a New User to Your AWS Account \(p. 72\)](#).

Each user can have two sets of active keys and two certificates for the purposes of credential rotation. For more information about the number of allowed for IAM entities, see [Limitations on IAM Entities \(p. 64\)](#).

Creating, Modifying, and Viewing User Security Credentials

This section shows how to create, modify, or view a user's security credentials. Security credentials consist of an Access Key ID and a Secret Access Key for a user. If you've performed the steps in [Getting Started \(p. 4\)](#), you've already created security credentials for a user.

By default, when you create a key, its status is `Active`, which means the user can use the key for API calls. At any point, you can disable the key, which means it can't be used for API calls. You might switch the status of a key if you're rotating keys (for more information, see [Rotating Credentials \(p. 136\)](#)) or to revoke API access for a user.

When you create the key, IAM returns the Secret Access Key and an Access Key ID, which is a public identifier for the key. You need to save these keys and give them to the user.

Important

To ensure the security of your AWS account, the Secret Access Key is accessible only during key and user creation. You must save the key (for example, in a text file) if you want to be able to access it again. If a secret key is lost, you can delete the access keys for the associated user and then create new keys.

You can give your users permission to list and manage their own keys. For more information, see [Adding a Credentials Management Policy to the User \(p. 135\)](#).

You can delete an access key at any time. However, when you delete an access key, it's gone forever and cannot be retrieved. (You can create new keys.)

The details of how you manage keys depends on whether you're using the IAM console, the command line interface (CLI), or API calls. The following sections provide instructions for each of these approaches.

AWS Management Console

To list a user's access keys

- On the **Navigation** pane, click **Users**, and then select the user.

AWS Identity and Access Management Using IAM

Creating, Modifying, and Viewing User Security Credentials

The screenshot shows the AWS IAM interface. On the left, a navigation sidebar lists 'IAM Dashboard', 'Groups', 'Users' (which is selected and highlighted in orange), 'Roles', and 'Password Policy'. The main area is titled 'Users' and shows a table of users. A red oval highlights the row for 'Tom'. The table columns are 'User Name', 'Groups', 'Password', 'Access Keys', and 'Creation Time'. The 'Access Keys' column for Tom shows '1 active, 1 inactive'. Below the table, a message says '1 User selected'. Under 'User: Tom', there are tabs for 'Groups', 'Permissions', 'Security Credentials' (which is selected and highlighted in orange), and 'Summary'. The 'Security Credentials' section is titled 'Access Credentials' and shows two sets of access keys: one labeled 'AQ - Inactive' created on 2012-06-03 16:32 PDT, and another labeled 'IPQ - Active' created on 2012-06-03 17:19 PDT. A 'Manage Access Keys' button is present. At the bottom, it says 'Signing Certificates: None'.

User Name	Groups	Password	Access Keys	Creation Time
Ben	1		1 active	2012-06-03
<input checked="" type="checkbox"/> Tom	1	✓	1 active, 1 inactive	2012-06-03
Colm	1		1 active	2012-06-03
Robert	1	✓	None	2012-06-03

The user's access keys and the status of each key is displayed on the **Security Credentials** tab. Users cannot have more than two sets of access keys.

Important

The user's Access Key ID is visible. The Secret Access Key cannot be retrieved after you are finished creating it.

To create, modify, or delete a user's access keys

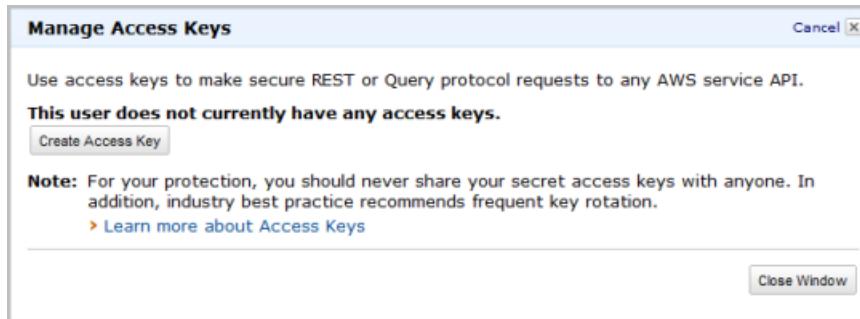
1. Select the user.

AWS Identity and Access Management Using IAM

Creating, Modifying, and Viewing User Security Credentials

The screenshot shows the AWS IAM 'Users' page. In the left navigation pane, 'Users' is selected. The main area displays a table of users with columns: User Name, Groups, Password, Access Keys, and Creation Date. The row for 'Tom' has a checked checkbox and is highlighted with a red oval. Below the table, it says '1 User selected'. Under 'User: Tom', the 'Security Credentials' tab is active. The 'Access Credentials' section shows two access keys: one inactive (AKI...AQ) and one active (AKI...IPQ). A red oval highlights the 'Manage Access Keys' button. At the bottom, it says 'Signing Certificates: None'.

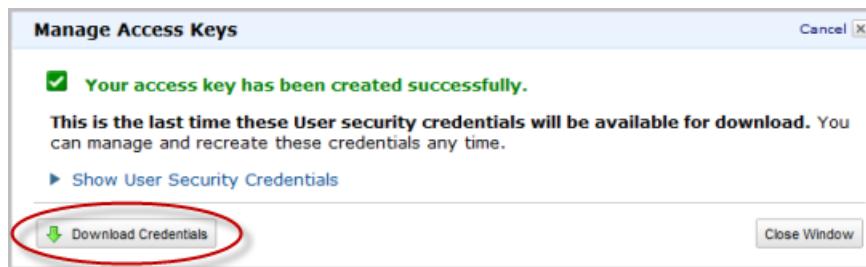
2. Click **Manage Access Keys**.
3. To create an access key, do this:
 - i. Click **Create Access Key**.



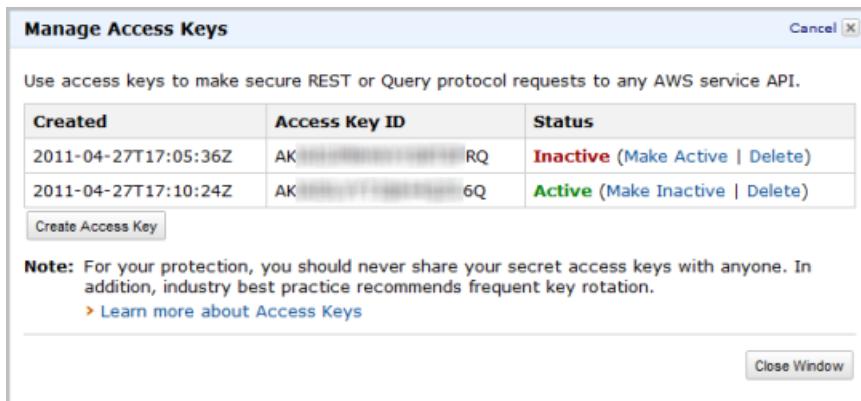
- ii. Click **Download Credentials** to save the Access Key ID and Secret Access Key to a .csv file on your computer. You will not have access to the Secret Access Key again after this dialog box closes, and you will need to provide this information to your users before they can begin using an AWS API.

AWS Identity and Access Management Using IAM

Creating, Modifying, and Viewing User Security Credentials



4. To disable an access key, click **Make Inactive**. To re-enable the key, click **Make Active**.



5. To delete a user's access key, click **Delete**.
6. Click **Close Window**.

Command Line Tools

The following table lists the CLI commands for managing a user's access keys. For more information about the commands, see the [AWS Identity and Access Management Command Line Interface Reference](#).

Task	Command to Use
Create an access key	iam-useraddkey
Disable or re-enable an access key	iam-usermodkey
List a user's access keys	iam-userlistkeys
Delete an access key	iam-userdelkey

API

The following table lists the actions for managing a user's access keys. For more information about the actions, go to the [AWS Identity and Access Management API Reference](#).

Task	Action to Use
Create an access key	CreateAccessKey

Task	Action to Use
Disable or re-enable an access key	UpdateAccessKey
List a user's access keys	ListAccessKeys
Delete an access key	DeleteAccessKey

Creating and Uploading a User Signing Certificate

If a user needs a signing certificate (for example, to use the Amazon EC2 command line tools), you first must obtain a signing certificate and then upload it to the IAM system. IAM doesn't have an API action to create signing certificates, so you must use a third-party tool such as OpenSSL to create the user signing certificate.

To create a user certificate using OpenSSL, you do the following:

1. Install OpenSSL from the <http://www.openssl.org/> site.
2. Configure OpenSSL for your operating system.
3. Create a private key.
4. Generate a certificate using the private key. Because you're creating a user signing certificate and not a server certificate, you don't need to submit a certificate signing request (CSR) to a Certificate Authority (CA). During this process, OpenSSL prompts you for values like your company name. For server certificates, these values are verified and signed by the CA and used by browsers to ensure that the domain is genuine. But for an AWS User certificate, these values are not used and you can leave them blank. The certificate that is generated must be in PEM format so you can copy the certificate contents into the console GUI to upload the user's signing certificate.
5. Upload the certificate.

Note

Although you can use the [security credentials](#) page to create an X.509 certificate, that method is only for the AWS account credentials. You can't upload a certificate generated using the security credentials page for IAM users. Instead, you must use the process described in this section (that is, using a third-party tool like OpenSSL).

As with access keys, each certificate can have a status of either `Active` or `Inactive`. By default, the status is `Active` when you upload the certificate.

When you upload the certificate, it returns a certificate ID that you can save for your records. However, if necessary, you can list the IDs for the user's certificates. You can delete a certificate at any time.

You can give your users permission to list and manage their own certificates. For more information, see [Adding a Credentials Management Policy to the User \(p. 135\)](#).

Install and Configure OpenSSL

Creating and uploading a certificate requires a tool that supports the SSL and TLS protocols. OpenSSL is an open-source tool that provides the basic cryptographic functions necessary to create an RSA token and sign it with your private key. If you don't already have OpenSSL installed, follow these instructions.

To install OpenSSL on Linux and UNIX

1. Go to [OpenSSL: Source, Tarballs](#) (<http://www.openssl.org/source/>).
2. Download the latest source and build the package.

To install OpenSSL on Windows

1. Go to [OpenSSL: Binary Distributions](http://www.openssl.org/related/binaries.html) (<http://www.openssl.org/related/binaries.html>).
2. Click **OpenSSL for Windows**.
A new page displays with links to the Windows downloads.
3. If it is not already installed on your system, select the **Microsoft Visual C++ 2008 Redistributables** link appropriate for your environment and click **Download**. Follow the instructions provided by the **Microsoft Visual C++ 2008 Redistributable Setup Wizard**.
4. After you have installed the Microsoft Visual C++ 2008 Redistributables, select the appropriate version of the OpenSSL binaries for your environment, save the file locally, and run it. The **OpenSSL Setup Wizard** launches.
5. Follow the instructions described in the **OpenSSL Setup Wizard**. Save the OpenSSL binaries to a folder in your working directory.

Before you use OpenSSL commands, you must configure the operating system so that it has information about the location of the OpenSSL install point.

To configure OpenSSL on Linux and UNIX

1. At the command line, set the `OpenSSL_HOME` variable to the location of the OpenSSL installation:

```
export OpenSSL_HOME=path_to_your_OpenSSL_installation
```

2. Set the path to the OpenSSL installation:

```
export PATH=$PATH:$OpenSSL_HOME/bin
```

To configure OpenSSL on Windows

1. Open a **Command Prompt** window.
2. Set the `OpenSSL_HOME` variable to the location of the OpenSSL installation:

```
set Path=OpenSSL_HOME\bin;%Path%
```

3. Set the path to the OpenSSL installation:

```
set Path=OpenSSL_HOME\bin;%Path%
```

Note

Any changes you make to Windows environment variables in a **Command Prompt** window are valid only for the current command-line session. You can make persistent changes to the environment variables by setting them as system properties. The exact procedures depends on what version of Windows you're using. (For example, in Windows 7, open **Control Panel > System and Security > System**. Then choose **Advanced system settings > Advanced tab > Environment Variables**.) For more information, see the Windows documentation.

Create a Private Key

You need a unique private key that you use when generating the user signing certificate.

To create a private key

- At the command line, use the `openssl genrsa` command with the following syntax:

```
openssl genrsa 1024 > private-key.pem
```

For `private-key.pem`, specify your own file name. In the example, 1024 represents 1024-bit encryption. AWS also supports 2048-bit and 4096-bit encryption. We recommend you create a 1024-bit or 2048-bit RSA key.

Create the User Signing Certificate

You can now create a user signing certificate for use with EC2 SOAP requests.

To create a user signing certificate

- Use the `openssl req` command and the following syntax:

```
openssl req -new -x509 -nodes -sha1 -days 365 -key private-key.pem -outform PEM > certificate.pem
```

For `private-key.pem`, use the .pem file that you generated in a previous step. For `certificate.pem`, use the name of a file into which you want the certificate to be generated. The certificate must be in .pem format.

In this example, the `-days 365` switch specifies that the certificate is good for 365 days. For information about the other switches, enter `openssl req -h` at the command line.

OpenSSL displays a message similar to the following:

```
You are about to be asked to enter information that will be incorporated into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank.  
For some fields there will be a default value.  
If you enter '.', the field will be left blank.
```

Because you're creating a user signing certificate (not a server certificate), you can leave all the values blank when you're prompted. These values are used by the Certificate Authority (CA) to help authenticate the server certificate. However, because user signing certificates are uploaded in an authenticated session, AWS does not need any information in the certificate for further validation, and requires only the public-private key pair.

The .pem file contains the certificate value that you can copy and paste during the upload procedure that follows.

Managing the Certificate Using the AWS Management Console

To upload a signing certificate for a user

1. Select the user.

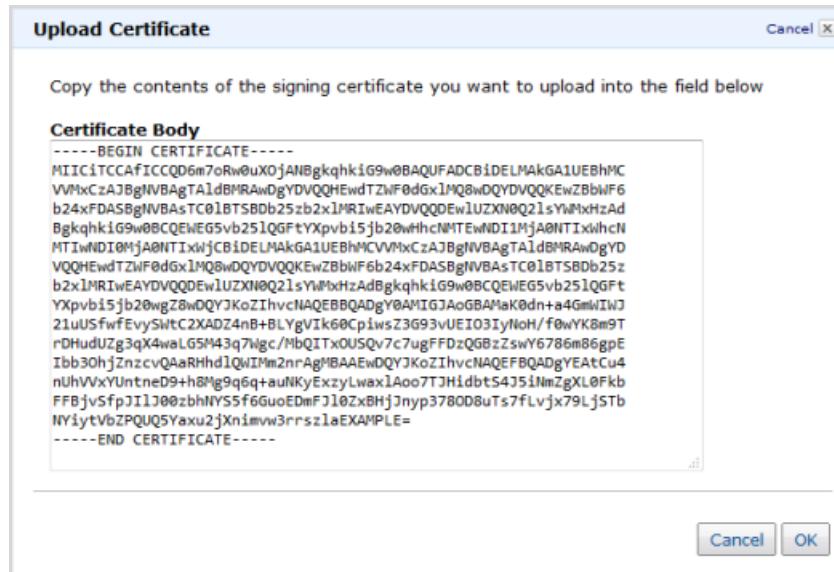
The screenshot shows the AWS IAM Management Console. In the left sidebar, under 'Navigation', 'Users' is selected. The main area displays a table of users with columns: User Name, Groups, Password, Access Keys, and Creation Time. The user 'Ted' is selected, indicated by a checked checkbox and highlighted with a red circle. On the right, a detailed view for 'User: Ted' is shown. It includes tabs for Groups, Permissions, Security Credentials (which is selected), and Summary. Under 'Security Credentials', it shows 'Access Keys: None' and 'Signing Certificates: None'. A red circle highlights the 'Manage Signing Certificates' button. To the right, there are fields for 'User name:' and 'Password:' and a 'Multi-Factor' section.

2. Click **Manage Signing Certificates**.
3. Click **Upload Signing Certificate**.

The screenshot shows the 'Manage Signing Certificates' dialog box. It contains a note: 'This user does not currently have any signing certificates.' Below this is a button labeled 'Upload Signing Certificate' which is highlighted with a red circle. A note below states: 'Note: Currently, IAM doesn't have an API action to create signing certificates, so you must use a third-party tool such as OpenSSL to create the certificate first. We recommend you create an RSA key that is either 1024-bit or 2048-bit in length. The certificate can be self-signed, and the key and certificate must be in PEM format.' There is also a link 'Learn more about Signing Certificates'. At the bottom right is a 'Close' button.

4. Using a text editor, open the .pem file that contains the user signing certificate that you created using OpenSSL, as described earlier in this section, and copy the contents of the .pem file.
5. Paste the contents of the user's signing certificate into the **Certificate Body** field.

AWS Identity and Access Management Using IAM Creating and Uploading a User Signing Certificate



To modify or delete a signing certificate for a user

1. Select the user.

The screenshot shows the AWS IAM "Users" page. On the left, there's a navigation sidebar with links for "IAM Dashboard", "Groups", "Users" (which is selected), "Roles", and "Password Policy". The main area displays a table of users:

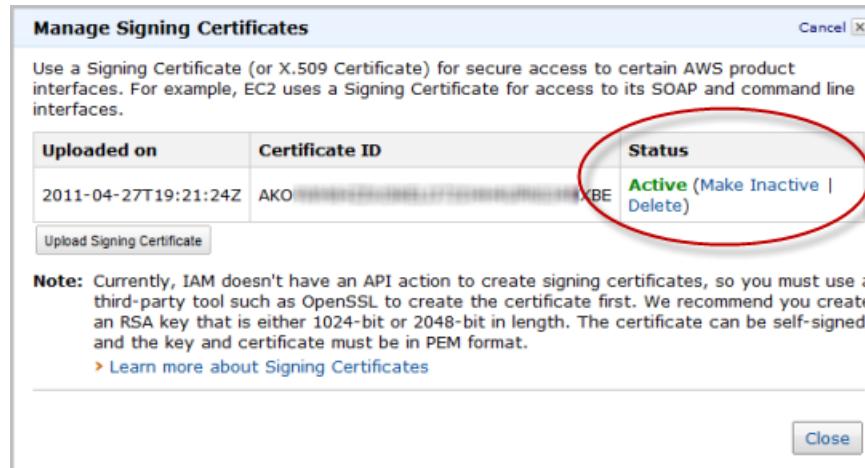
User Name	Groups	Password	Access Keys	Creation Time
Paul	1		1 active	2012-06-03 16:29 P
<input checked="" type="checkbox"/> Ted	1		None	2012-06-03 16:29 P
Bob	2		1 active	2012-06-03 16:18 P
Phil	2		1 active	2012-06-03 16:18 P
Susan	2		1 active	2012-06-03 16:18 P

Below the table, it says "1 User selected" and shows details for "User: Ted". The "Security Credentials" tab is selected. Under "Access Credentials", it says "Access Keys: None" and has a "Manage Access Keys" button. Under "Signing Certificates", it says "Signing Certificates: None" and has a "Manage Signing Certificates" button, which is circled in red.

2. Click **Manage Signing Certificates**.
3. To disable or re-enable a signing certificate, click **Make Inactive** or **Make Active**.

AWS Identity and Access Management Using IAM

Creating and Uploading a User Signing Certificate



4. To delete a user's signing certificate, click **Delete**.
5. Click **Close**.

Note

After the certificate has been uploaded, it cannot be retrieved, its contents cannot be viewed, and it cannot be reused.

To view a list of signing certificates belonging to a user

- On the **Navigation** pane, click **Users**, and then select the name of the user whose signing certificate you want to view.

The screenshot shows the 'Users' page in the AWS IAM console. The left sidebar shows 'Navigation' with 'Users' selected, indicated by a red oval. The main area displays a table of users with columns: User Name, Groups, Password, Access Keys, and Creation Time. The user 'Ted' is selected, indicated by a red oval. Below the table, a section for 'User: Ted' shows tabs for 'Groups', 'Permissions', 'Security Credentials' (which is active), and 'Summary'. Under 'Access Credentials', it says 'Access Keys: None' and has a 'Manage Access Keys' button. Under 'Signing Certificates', it shows 'AKO...XBE - Active' uploaded on 2011-04-27T19:21:24Z, with a 'Manage Signing Certificates' button.

User Name	Groups	Password	Access Keys	Creation Time
Paul	1		1 active	2012-06-03 16:29 PDT
<input checked="" type="checkbox"/> Ted	1		1 active	2012-06-03 16:29 PDT
Bob	2		1 active	2012-06-03 16:18 PDT
Phil	2		1 active	2012-06-03 16:18 PDT
Susan	2		1 active	2012-06-03 16:18 PDT

Signing certificates and the status of certificate is displayed on the **Security Credentials** tab. Users cannot have more than two signing certificates.

Command Line Tools

The following table lists which command to use for each task you want to perform with a signing certificate. For more information about the commands, go to the [AWS Identity and Access Management Command Line Interface Reference](#).

Task	Command to Use
Upload a certificate	iam-useraddcert
Disable or re-enable a certificate	iam-usermodcert
List a user's certificates	iam-userlistcerts
Delete a certificate	iam-userdelcert

API

The following table lists which actions to use for each task you want to perform with a signing certificate. For more information about the actions, go to the [AWS Identity and Access Management API Reference](#), or refer to your SDK's documentation.

Task	Action to Use
Upload a certificate	UploadSigningCertificate
Disable or re-enable a certificate	UpdateSigningCertificate
List a user's certificates	ListSigningCertificates
Delete a certificate	DeleteSigningCertificate

Note

Use a POST request when uploading a signing certificate because of the certificate's size.

Adding a Credentials Management Policy to the User

You might want to give your users the ability to manage their own security credentials. This section shows how to add a policy that lets a specific user create and manage his or her own credentials.

There's a limit to how many credentials a user can have. (Users can have two sets of keys; for more information, see [Limitations on IAM Entities \(p. 64\)](#)). Typically, the only reason a user might have more than one active at a time is for the purposes of rotating credentials. For more information, see [Rotating Credentials \(p. 136\)](#).

You could give the user access to only some of the actions related to the credentials (perhaps just listing them), or to a wider set if you want the user to be able to create and rotate the credentials. The following example policy lets the user named Bob perform any of the credential-related actions on his own credentials.

```
{  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": ["iam:*AccessKey*", "iam:*SigningCertificate*"],  
            "Resource": "arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/Bob"  
        }  
    ]  
}
```

Notice that the resource in the policy is the Amazon Resource Name (ARN) for Bob. If you want to let another user named Jane manage her own credentials, you would replace the ARN for Bob with the ARN for Jane in the policy, and then attach the policy to Jane. You must explicitly name the user in the resource; for purposes of referential integrity, there's no way to use a wildcard that automatically refers to *self* (that is, the user the policy is attached to). In other words, there's no group-level policy you can write that will give all your users credential management permissions only for their own credentials. You need to write an individual policy for each user, and attach the policy to the user. For more information, see Example 3 in [Example Policies for IAM Entities \(p. 179\)](#)

The preceding policy uses wildcards ("Action": "iam:*AccessKey*", where the * matches zero or multiple characters). You could instead explicitly list each of the IAM actions related to access keys and certificates (for example, CreateAccessKey, UploadSigningCertificate, and so on). If you use the wildcards in your policy, be aware that if we were to add any new IAM actions that include the string AccessKey or SigningCertificate in their names, this policy would automatically give Bob access to those new actions to use with his own credentials.

For information about uploading and managing policies, see [Managing IAM Policies \(p. 173\)](#).

Rotating Credentials

For the purposes of security, we recommend that you, an administrator, or your users regularly rotate the users' credentials (we recommend a frequency of every 90 days).

Note

If you're using the AWS account's credentials on a regular basis, we recommend you also regularly rotate those. Users can't manage credentials for the AWS account, so you must use the AWS account's credentials (and not a user's) when making the required API calls to rotate the credentials.

The following steps describe the general process for rotating either a key or a certificate without interrupting your applications.

1. While the first set of credentials is still active, create a second set of credentials (or upload a second one, in the case of a certificate), which will be active by default. At this point, the user has two active sets of credentials.

Related action: [CreateAccessKey](#) or [UploadSigningCertificate](#)

Related command: [iam-useraddkey](#) or [iam-useraddcert](#)

2. Update all applications to use the new credentials.
3. Change the state of the first set of credentials to `Inactive`.

Related action: [UpdateAccessKey](#) or [UpdateSigningCertificate](#)

Related command: [iam-usermodkey](#) or [iam-usermodcert](#)

4. Using only the new credentials, confirm that your applications are working well. If you need to, you can revert to using the original set of credentials by switching its state back to `Active`.

5. Delete the first set of credentials.

Related action: [DeleteAccessKey](#) or [DeleteSigningCertificate](#)

Related command: [iam-userdelkey](#) or [iam-userdelcert](#)

For information about creating and enabling or disabling access keys in the AWS Management Console, see [Creating, Modifying, and Viewing User Security Credentials \(p. 125\)](#). For information about uploading and enabling or disabling signing certificates in the console, see [Creating and Uploading a User Signing Certificate \(p. 129\)](#).

Delegating API Access by Using Roles

IAM roles allow you to delegate access to users or services that normally don't have access to your organization's AWS resources. IAM users or AWS services can assume a role to obtain temporary security credentials that can be used to make AWS API calls. Consequently, you don't have to share long-term credentials or define permissions for each entity that requires access to a resource. The following scenarios highlight some of the challenges that you can address by delegating access:

Granting applications that run on Amazon EC2 instances access to AWS resources

To grant applications on an Amazon EC2 instance access to AWS resources, developers might distribute their credentials to each instance. Applications can then use those credentials to access resources such as Amazon S3 buckets or Amazon DynamoDB data. However, distributing long-term credentials to each instance is challenging to manage and a potential security risk.

Cross-account access

To control or manage access to resources, such as isolating a development environment from a production environment, you might have multiple AWS accounts. However, in some cases, users from one account might need to access resources on the other account. For example, a user from the development environment might require access to the production environment to promote an update. Therefore, users must have credentials for each account, but managing multiple credentials for multiple accounts makes identity management difficult.

Instead of creating and distributing multiple credentials, you can delegate API access by using IAM roles. You create a role in the AWS account where you want to delegate access. When you create the role, you specify two policies: one that defines who can assume the role (the trusted entities) and another that defines what actions can be done on which resources. You can specify account IDs or AWS services as trusted entities.

For IAM users to assume the role, they must be granted permission to call `sts:AssumeRole` for that specific role. After assuming the role, the user can use the role's temporary security credentials to access resources that are defined in the role. With the temporary security credentials, the user acts as the role and has only the permissions that are associated with the role. Callers can further restrict permissions by passing an additional policy when they assume a role.

For details about how you can use roles to delegate API access, see the following scenarios:

- [Granting Applications that Run on Amazon EC2 Instances Access to AWS Resources \(p. 139\)](#)
- [Enabling Cross-Account API Access \(p. 142\)](#)

Granting Applications that Run on Amazon EC2 Instances Access to AWS Resources

Imagine that you are an administrator who manages your organization's AWS resources. Developers in your organization have applications that run on Amazon EC2 instances. These applications require access to other AWS resources--for example, making updates to Amazon S3 buckets.

Applications that run on an Amazon EC2 instance must sign their AWS API requests with AWS credentials. One way to do this is for developers to pass their AWS credentials to the Amazon EC2 instance, allowing applications to use the credentials to sign requests.

However, when AWS credentials are rotated, developers have to update each Amazon EC2 instance that uses their credentials.

IAM Roles Provide a Solution

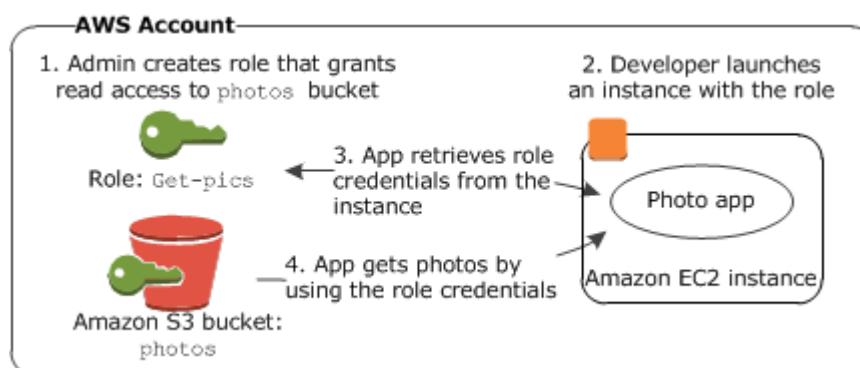
With IAM roles, no one has to distribute AWS credentials to Amazon EC2 instances. Instead, an administrator can create a role with the permissions that applications require when the application makes calls to other AWS resources. When developers launch an Amazon EC2 instance, they can specify a role to associate with the instance. All applications that run on the instance can use the role credentials to sign requests. Because role credentials are temporary and rotated automatically, you don't have to worry about long-term security risks.

Also, if you use a single role for multiple instances, you can make a change to the role and that change is propagated to all the instances, simplifying credential management.

How Do Roles for Amazon EC2 Instances Work?

In the following figure, a developer is running an application on an Amazon EC2 instance that requires access to the Amazon S3 bucket named `photos`. An administrator creates the `Get-pics` role, which grants read permissions for the bucket, and allows the developer to launch it with an Amazon EC2 instance. When the application runs on the instance, it can access the `photos` bucket by using the role's temporary credentials. The administrator doesn't have to grant the developer permission to access the `photos` bucket, and the developer never has to share credentials.

Application on an instance accessing an AWS resource



1. An administrator uses IAM to create the `Get-pics` role. In the role, the administrator defines that only Amazon EC2 instances can assume the role and allows read permissions for the `photos` bucket.

2. A developer launches an Amazon EC2 instance and associates the `Get-pics` role with that instance. Now, any application on the instance can assume the `Get-pics` role to obtain read access to the photos bucket.

Important

Because roles are temporary security credentials, you must ensure that applications that use roles on Amazon EC2 instances support [temporary security credentials](#). Otherwise, the application might be denied access.

3. When the application runs, it uses the role credentials that are available on the Amazon EC2 instance. The application retrieves credentials from the instance metadata.
4. Using the role credentials, the application accesses the photo bucket with read-only permissions.

Permissions Required for Using Roles with Amazon EC2

To launch an instance with a role, the developer must have permission to launch Amazon EC2 instances and permission to pass IAM roles.

The following sample policy allows users to use the AWS Management Console to launch an instance with a role. The policy allows a user to pass any role and to perform all Amazon EC2 actions by specifying an asterisk (*). The `ListInstanceProfiles` action allows users to view all the roles that are available on the AWS account.

Example Policy that grants a user permission to launch an instance with any role by using the Amazon EC2 console

```
{  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "iam:PassRole",  
            "Resource": "*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "iam>ListInstanceProfiles",  
            "Resource": "*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "ec2:*",  
            "Resource": "*"  
        }]  
}
```

Important

Users could indirectly escalate their privileges by launching an Amazon EC2 instance with a role that has more permissions assigned to it than the user. After launching the instance, users can access the role if they have access to the launched instance. You can control which roles users can use when launching Amazon EC2 instances by limiting permissions on the `PassRole` action so that users can launch instances with certain roles and not any role.

The following sample policy allows developers to use the Amazon EC2 API to launch an instance with a role. The `Resource` element specifies an Amazon Resource Name (ARN). By specifying the ARN, the policy grants the user to pass only the `Get-pics` role.

Example Policy that grants a user permission to launch an instance with a role by using the Amazon EC2 API

```
{  
    "Statement": [ {  
        "Effect": "Allow",  
        "Action": "ec2:RunInstances",  
        "Resource": "*"  
    },  
    {  
        "Effect": "Allow",  
        "Action": "iam:PassRole",  
        "Resource": "arn:aws:iam::123456789012:role/Get-pics"  
    } ]  
}
```

How Do I Get Started?

To understand how roles work with EC2 instances, you need to use the IAM console to create a role, launch an EC2 instance that uses that role, and then examine the running instance to see how it's using the role credentials. You cannot directly see the role being used. However, you can see how the role credentials are made available to an instance, and you can see how an application that runs in an instance can use the role. Use the following resources to learn more:

- [Getting Started with IAM Roles for EC2 Instances](#). This three-minute video walks through how to create a role, set permissions for the role, and launch an EC2 instance with the role. The video then shows how to use instance metadata to verify that the instance uses the role's credentials.
- SDK walkthroughs. The AWS SDK documentation includes walkthroughs that show an application running in an Amazon EC2 instance that uses role credentials to read an Amazon S3 bucket. Each of the following walkthroughs presents a similar walkthrough using a different programming language:
 - [Using IAM Roles for EC2 Instances with the SDK for Java](#) in the *AWS SDK for Java Developer Guide*
 - [Using IAM Roles for EC2 Instances with the SDK for .NET](#) in the *AWS SDK for .NET Developer Guide*
 - [Using IAM Roles for EC2 Instances with the SDK for PHP](#) in the *AWS SDK for PHP Developer Guide*
 - [Using IAM Roles for EC2 Instances with the SDK for Ruby](#) in the *AWS SDK for Ruby Developer Guide*

The walkthroughs provide complete step-by-step instructions for creating and compiling the example program, creating the role, launching the instance, connecting to the instance, deploying the example program, and testing it.

Related Information

For more information about creating roles or roles for Amazon EC2 instances, view the following information:

- For more information about [using IAM roles with Amazon EC2 instances](#), go to the *Amazon Elastic Compute Cloud Using Guide*.
- To create a role, view [Creating a Role \(p. 151\)](#)
- For more information about [using temporary security credentials](#), go to the *Using Temporary Security Credentials* guide.
- If you work with the IAM API or CLI, you must create and manage IAM instance profiles. For more information about instance profiles, see [Instance Profiles \(p. 142\)](#).

- For more information about role credentials in the instance metadata, see IAM security credentials in the Metadata Categories section of [Instance Metadata](#).

Instance Profiles

An instance profile is a container for a role. To associate a role to an Amazon EC2 instance, you must use the instance profile name.

If you use the AWS Management Console to create and manage roles, instance profiles are automatically managed for you. If you use the IAM API or CLI to create and manage roles, you must create instance profiles for each role. A role can be associated with many instance profiles, but an instance profile can be associated with only one role.

When you use the Amazon EC2 console to launch an instance with an IAM role, you use the **IAM Role** list. The IAM Role list is populated with instance profile names. If you use the AWS Management Console to create a role, the instance profile is created automatically and its name is the same as the name of the role it corresponds to. However, if you use the API, the CLI, or a third-party tool to create roles and instance profiles, the names of your instance profiles might not correspond to the names of your roles. In this case, you need to know the names of your instance profiles as well as the names of roles they contain so that you can choose the correct instance profile.

You can use the following commands to list all the instance profiles in a AWS account, or all the instance profiles with a particular path prefix. The output lists the Amazon Resource Name (ARN) for each instance profile.

- For the CLI, enter [iam-instanceprofilelistbypath](#).
- For the API, call [ListInstanceProfiles](#).

Use the following commands to list the ARNs of the instance profiles associated with a particular role:

- For the CLI, enter [iam-instanceprofilelistforrole](#).
- For the API, call [ListInstanceProfilesForRole](#).

Enabling Cross-Account API Access

You are an administrator who uses IAM to control access to your organization's AWS resources. Occasionally, your organization might have resources that users must access across multiple AWS accounts. To allow this access, you can establish trust between accounts so that users from one account can access resources in another account. Using IAM roles, you define trusted entities and the actions they are permitted to do.

Suppose your organization has multiple AWS accounts that you use to isolate different environments. For example, you might isolate a production account from a development account. In the Development account, you let developers work freely with all AWS resources. But in the Production account, you limit developers to read and write access to specific resources.

Because the environments are separated by two different AWS accounts, you can tightly control who can access the Production account and what actions they can do. However, with multiple accounts, you must manage multiple users on multiple accounts, increasing overhead for you and your users.

To avoid this overhead, you can create roles for cross-account API access, and you won't need to create multiple IAM users for each person to access resources in another account. For example, if you want IAM users in the Development account to be able to access resources in the Production account, you can create a role in the Production account that IAM users in the Development account can assume.

When you create the role, you specify an AWS account from which IAM users can assume the role. You also define the permissions that are available with the role, such as read and write permissions to a specific Amazon S3 bucket.

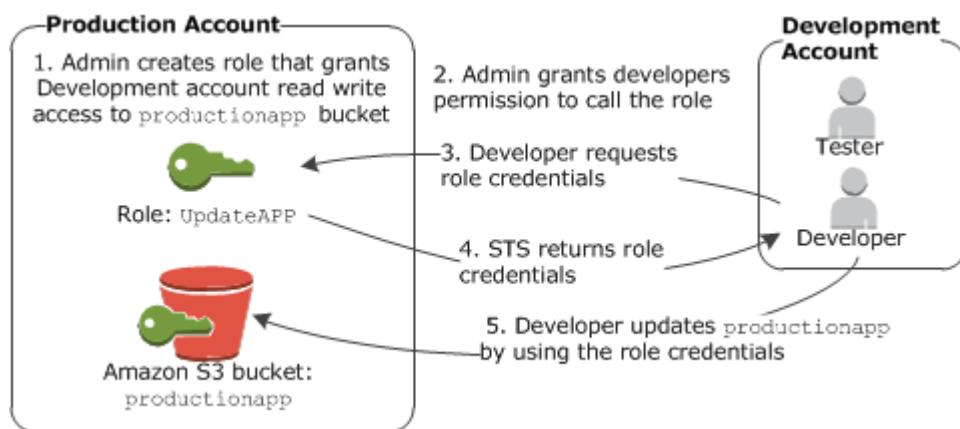
Important

Only IAM users can assume a role. If you use AWS account credentials, access is denied.

How Does Cross-Account Access with Roles Work?

In the following figure, all users are managed in the Development account, but some developers require limited access to the Production account. The Development account has two groups: Testers and Developers, and each group has its own policy.

Cross-account API access



1. An administrator in the Production account uses IAM to create the `UpdateAPP` role. The administrator defines the Development account as a trusted entity, meaning authorized users from the Development account can assume the `UpdateAPP` role. The administrator also defines a policy for the role that specifies that the role has read and write permissions for the `productionapp` bucket.

The administrator then shares the Amazon Resource Name (ARN) of the role with anyone who needs to assume it. An ARN is automatically associated with a role when a role is created.

`arn:aws:iam::123456789012:role/UpdateAPP` is a sample ARN for the `UpdateAPP` role in account number 123456789012.

2. Because only developers are required to update the `productionapp` bucket, the administrator grants the `Developer` group permission to call the role. The administrator of the Development account must grant the `Developer` group permissions to call the AWS Security Token Service (AWS STS) `AssumeRole` API for the `UpdateAPP` role. By default, Developers and Testers don't have permission to call `AssumeRole`.
3. A user in the `Developer` group of the Development account calls `AssumeRole` to obtain the `UpdateAPP` role credentials. The user specifies the `UpdateAPP` ARN as part of the call. If a user in the `Testers` group makes the same request, the request fails because Testers do not have permission to call `AssumeRole` by using the `UpdateAPP` role ARN.
4. AWS STS verifies the request against the role's trust policy to ensure that it's from a trusted entity (Development account). After verification, AWS STS returns temporary security credentials to the developer.
5. The developer uses the temporary security credentials to update the `productionapp` bucket. With the temporary security credentials, the developer can only read and write to the `productionapp` bucket and cannot access any other resource in the Production account.

Related Information

To view a detailed walkthrough of cross-account access, see [Enabling Cross-Account API Access Walkthrough \(p. 144\)](#).

To see a list of services that can use roles, see [Using Temporary Security Credentials to Access AWS](#) in the *Using Temporary Security Credentials* guide.

Enabling Cross-Account API Access Walkthrough

In the following walkthrough, you will learn how to delegate access to AWS accounts that you own so that you don't need to manage multiple credentials for each user on different AWS accounts.

Before You Begin

Suppose that your organization has two AWS accounts: Production and Development. The Production account is where live applications are managed, and the Development account is a sandbox where developers and testers can freely test applications. In each account, application information is stored in Amazon S3 buckets.

The walkthrough assumes that you are an administrator of both accounts. You manage IAM users in the Development account, where you have two IAM groups: Developer and Tester with power-user access (permissions to all actions except IAM). A developer, David, is in the Developer group, and a tester, Teresa, is in the Tester group.

Also, you have an existing Amazon S3 bucket called `productionapp` in the Production account that developers from the Development account will access. The walkthrough shows you how you can use IAM roles to provide cross-account API access to that `productionapp` bucket.

For more information about IAM users and groups or Amazon S3 buckets, see the following:

- [Working with Users and Groups \(p. 72\)](#)
- [Create a Bucket with Amazon Simple Storage Service](#) in the *Amazon Simple Storage Service Getting Started Guide*.

Overview

From time to time, David must update the live applications in the Production account; however, you don't want to create another IAM user for David in the Production account. As your organization grows, creating multiple IAM users for each developer will be costly to manage. Instead, you can delegate API access to developers from the Development account. You establish trust between the two accounts, and allow only developers from the Development account to access the `productionapp` bucket in the Production account.

What You Will Accomplish

By the end of the walkthrough, you will have a role in the Production account that allows users from the Development account to access the `productionapp` bucket in the Production account. After assuming the role, David uses the assumed role credentials to access the `productionapp` bucket in the Production account. A similar call by Teresa to assume the role fails.

The following list outlines the steps that you will complete:

1. [Creating a Role \(p. 145\)](#)

Use the AWS Management Console to establish trust between the Production and Development account by creating an IAM role named *UpdateAPP*. When you create the role, you will define the Development account as a trusted entity and specify a role policy that allows trusted users to only update the `productionapp` bucket.

2. Modifying Group Permissions (p. 148)

Modify the IAM group policy so that testers are denied access to the `UpdateAPP` role. In this walkthrough, the permissions provided by power-user access allows Developers to assume the role. Testers, who have similar permissions, will be denied permission to assume the role.

3. Assuming a Role (p. 149)

As the IAM user David, assume the `UpdateAPP` role. Using the assumed role credentials, you can update the `productionapp` bucket in the Production account.

Creating a Role

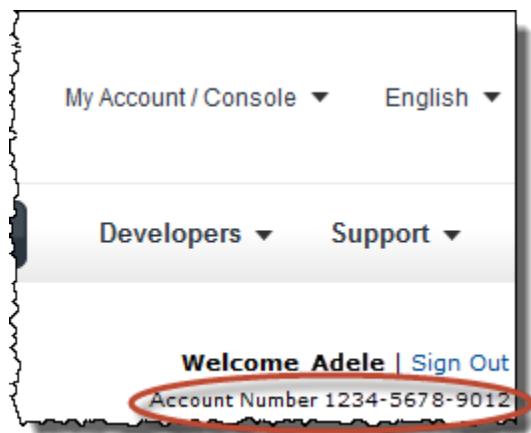
To allow users from one AWS account to access resources on another AWS account, create a role that defines who can access the account and what permissions are available on the role.

You create the role on the Production account and specify the Development account as a trusted entity. You also limit the role's permissions to only read and write access to the `productionapp` bucket. As long as David is granted permission to assume the role, he can read and write to the `productionapp` bucket by using the assumed role's temporary security credentials.

Before you can create a role, you need the account ID of the Development account. The account ID is a unique identifier that is unique to each AWS account.

To obtain the Development account ID

1. From the [AWS website](#), click **My Account/Console** and then select **My Account**.



2. Log in to the AWS Management Console for the Development account.

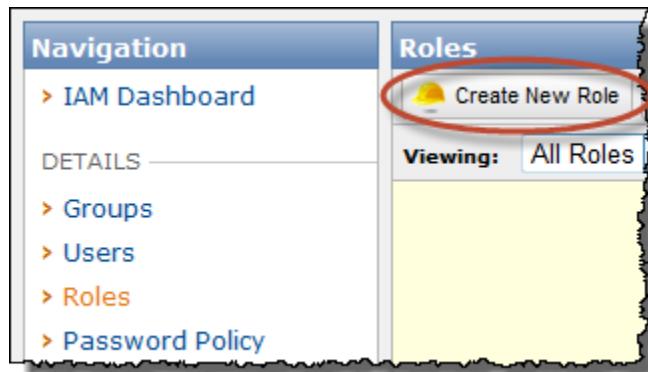
Ensure that you are logging in to the correct account.

3. After you log in, you can find the account number below your user name.

The account ID is a 12-digit number. For this scenario, the Development account ID is 123456789012; however, you should use a valid account ID to avoid any errors.

To create a role

1. Log in to the AWS Management Console as an administrator of the Production account.
2. Navigate to the IAM console.
3. In the **Navigation** pane, click **Roles**.
4. Click **Create New Role**.



5. Enter `updateAPP` for the role name, and then click **Continue**.
6. Click **Roles for Cross-Account Access**, and then select **Provide access between AWS accounts you own**.

The screenshot shows the 'Create Role' wizard, step 1: 'Select Role Type'. It has four tabs at the top: 'CONFIGURE ROLE' (highlighted), 'ESTABLISH TRUST', 'SET PERMISSIONS', and 'REVIEW'. Below the tabs, there are two radio button options: 'AWS Service Roles' and 'Roles for Cross-Account Access'. The 'Roles for Cross-Account Access' option is selected and highlighted with a red circle. Underneath, there are two sections: 'Provide access between AWS accounts you own' (selected) and 'Provide access to a 3rd party AWS account'. Each section has a 'Select' button with a yellow arrow icon, which is also circled in red.

7. Enter the Development account ID `123456789012`.

For this walkthrough, the Development account ID is `123456789012`; however, you should use a valid account ID to complete this walkthrough without any errors.

8. Click **Continue** to set the permissions that will be associated with the role.
9. Select **Custom Policy** to create read and write permissions for the `productionapp` bucket.

You want to set read write access to the `productionapp` bucket. Although IAM provides some Amazon S3 policy templates, one is not provided for read write access to an Amazon S3 bucket, so you can create your own policy instead.

10. Enter `read-write-app-bucket` for the policy name.
11. Add the following permissions to the policy document:

```
{  
    "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/*" ]  
        }  
    ]  
}
```

The `ListBucket` permission allows users to view objects in the `productionapp` bucket. The `GetObject`, `PutObject`, `DeleteObject` permissions allows users to view and update the contents within the `productionapp` bucket.

12. Click **Continue** to review the role before it's created.
13. After reviewing the role, click **Create Role**.

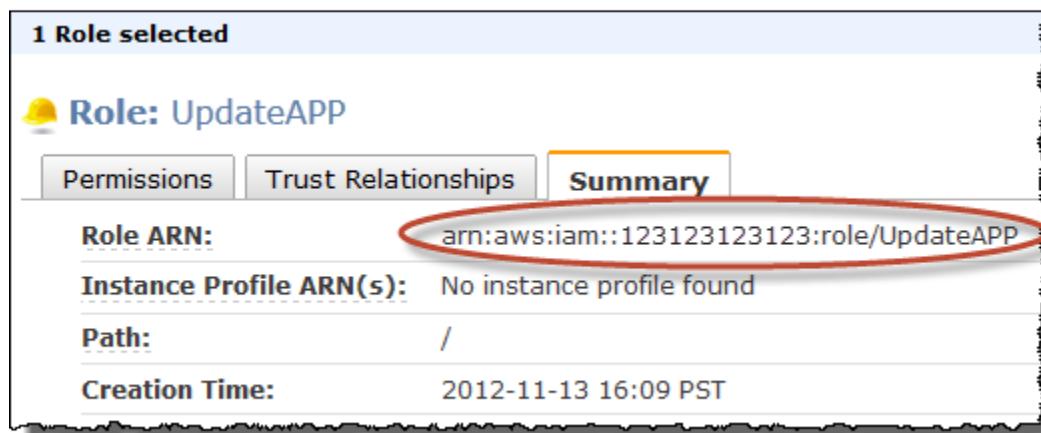
The `UpdateAPP` role is displayed in the list of roles.

Now, you must obtain the role's Amazon Resource Name (ARN), which is a unique identifier for the role. When you modify the Tester group's policy, you will specify the role's ARN to deny testers from assuming this role.

To obtain the ARN for UpdateAPP

1. In the **Navigation** pane of the IAM console, click **Roles**.
2. From the list of roles, click the `UpdateAPP` role.
3. In the **Summary** tab for the `UpdateAPP` role, record the Role ARN value.

The Production account has an account ID of 123123123123, so the role ARN is `arn:aws:iam::123123123123:role/UpdateAPP`.



Summary and What's Next?

At this point, you have established trust between the Production and Development accounts by creating a role in the Production account. You also defined what users from the Development account can do if they are allowed to assume the `UpdateAPP` role.

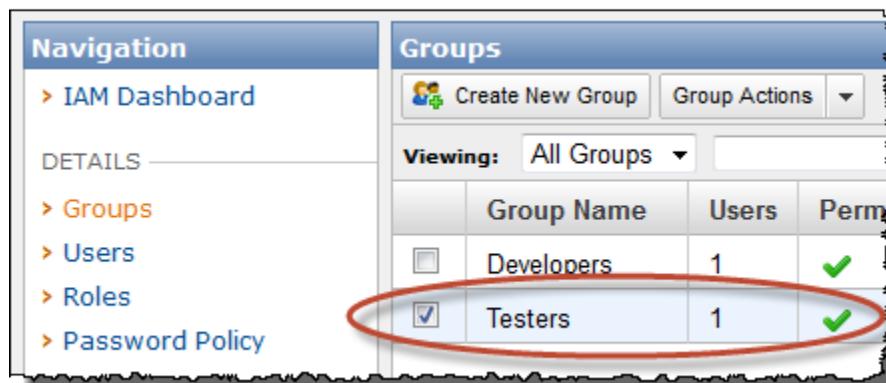
Next, modify the permissions for the Tester group.

Modifying Group Permissions

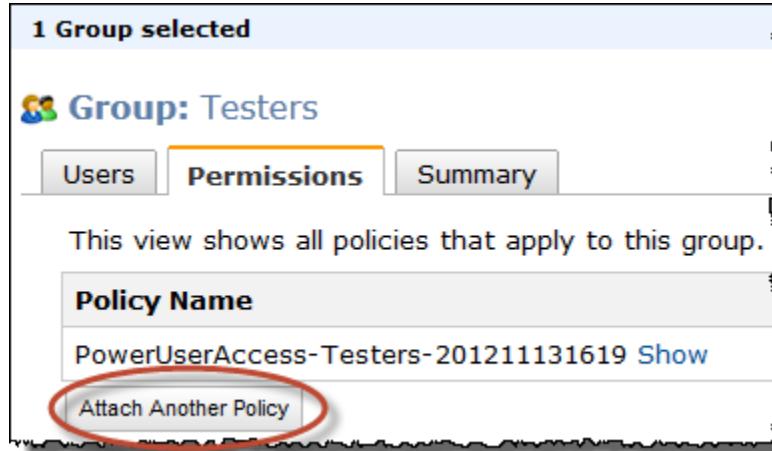
Because testers and developers have power-user access, they can freely test applications in the Development account. They have permission to all actions except for IAM actions so that they cannot create or modify permissions. With power-user access, the Developer group has the necessary permissions to assume the `UpdateAPP` role. However, the Testers group's policy must be modified so that it doesn't have access to the `UpdateAPP` role.

To modify the Testers group

1. Log in as an administrator in the Development account.
2. Navigate to the IAM console.
3. Click **Groups**.
4. From the list of groups, click **Testers**.



5. Click the **Permissions** tab for the Testers group.
6. Click **Attach Another Policy**.



7. Select **Custom Policy**.
8. (Optional) You can rename the policy by editing the **Policy Name** field.
9. Add the following policy statement to deny the assume role action on the `UpdateAPP` role:

```
{  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Action": "sts:AssumeRole",  
      "Resource": "arn:aws:iam::123123123123:role/UpdateAPP"  
    }  
  ]  
}
```

The deny permission explicitly denies the Testers group access to the `UpdateAPP` role. Any tester who tries to access the role will get an access denied message. However, because you specified a specific role ARN, testers can still assume other roles.

10. Click **Apply Policy** to add the policy to the Tester group.

Summary and what's next?

The Developer group has access to assume the `UpdateAPP` role. The Testers group had a similar policy, but you modified the Testers group policy by adding a deny permission that prevents them from accessing the `UpdateAPP` role.

Next, you'll learn how David can access the `productionapp` bucket in the Production account by using the `AssumeRole` API call.

Assuming a Role

When David needs to make an update in the Production account, he makes an `AssumeRole` call to assume the `UpdateAPP` role. AWS Security Token Service (STS) returns temporary credentials that he can use to access the `productionapp` bucket in the Production account. With those credentials, David

can update the `productionapp` bucket but cannot access any other resources in the Production account, even though he has power-user privileges in the Development account.

Important

Only IAM users can assume a role. If you use AWS account credentials, access is denied.

To assume a role

1. David calls `AssumeRole` as part of an application. He must specify the `UpdateAPP` ARN:
`arn:aws:iam::123123123123:role/UpdateAPP`.

The response from the `AssumeRole` call includes the temporary credentials. A similar call by Teresa would fail.

2. With the temporary credentials, David makes an `s3:PutObject` call to update the `productionapp` bucket.

Because the temporary role credentials have only read-write access to the `productionapp` bucket, any other actions in the Production account are denied.

Summary

You have completed the cross-account API access walkthrough. You created a role to establish trust with another account and defined what actions trusted entities can do. Then, you modified a group policy to limit which IAM users can access the role. As a result, developers from the Development account can make updates to the `productionapp` bucket in the Production account by using temporary credentials.

Delegating API Access to Third Parties

When third parties require access to your organization's AWS resources, you can use roles to delegate API access to them. For example, a third party might provide a service for managing your AWS resources. With IAM roles, you can grant these third parties access to your AWS resources without sharing your AWS security credentials. Instead, they can assume a role that you created to access your AWS resources.

Note

When you grant third parties access to your AWS resources, they can access any resource that you give them permissions to. In addition, their use of your resources is billed to you. Make sure that you limit their use of your resources appropriately.

Third parties must provide you the following information for you to create a role that they can assume:

- The AWS account ID that the third party's IAM users use to assume your role. You specify their AWS account ID when you define the trusted entity for the role.
- An external ID that the third party can associate you with your role. You specify the ID that was provided by the third party when you define the trusted entity for the role.
- The permissions that the third party requires in order to work with your AWS resources. You specify these permissions when defining the role's permission policy. This policy defines what action they can do and what resources they can access.

After you create the role, you must share the role's Amazon Resource Name (ARN) with the third party. They require your role's ARN in order to assume the role.

Related Information

- For more information about the external ID, see [About the External ID](#).
- For a detailed walkthrough about creating a role with the AWS Management Console, see [Creating a Role \(p. 145\)](#).

Delegating API Access Within an AWS Account

For mission critical permissions that IAM users might not frequently use, you can separate those permissions from their normal day-to-day permissions by using roles. Users would then have to actively assume a role, which can prevent them from accidentally performing disruptive actions.

For example, you might have Amazon EC2 instances that are critical to your organization. Instead of directly granting administrators permission to terminate the instances, you can create a role with those privileges and allow administrators to assume the role. Administrators won't have permission to terminate those instances; they must first assume a role. By using a role, administrators must take an additional step to assume a role before they can stop an instance that's critical to your organization.

How Do I Get Started?

When you create a role, specify the AWS account ID for the account that you are creating the role in so that IAM users within the account can assume the role. Then, you must grant assume role permissions to IAM users who can assume the role.

Related Information

For more information about creating and assuming roles, see the following topics:

- For a detailed walkthrough about creating a role with the AWS Management Console, see [Creating a Role \(p. 145\)](#).
- To assume a role, see [Assuming a Role \(p. 160\)](#).

Creating a Role

You can create a role by using the AWS Management Console, CLI, or the API. If you use the CLI or API to create roles for Amazon EC2 instances, you must also create an instance profile and add the role to the instance profile.

Topics

- [Create a Role \(AWS Management Console\) \(p. 151\)](#)
- [Create a Role \(CLI\) \(p. 156\)](#)
- [Create a Role \(API\) \(p. 157\)](#)

Note

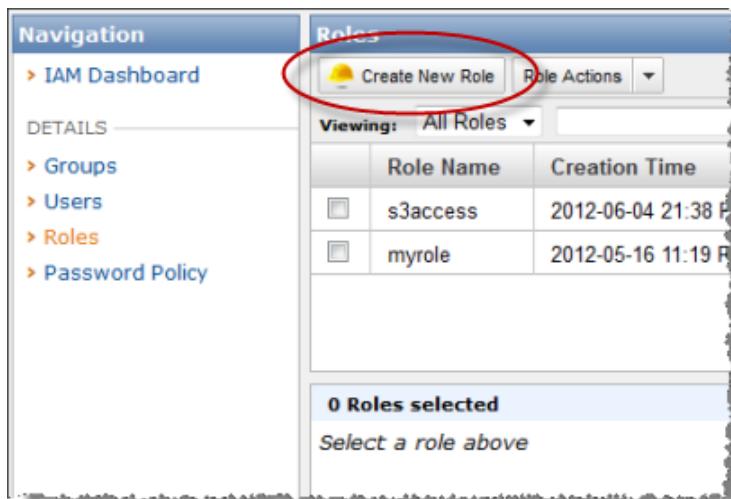
Role names have character limitations. The number of roles and policy size are also limited. For more information, see [Limitations on IAM Entities \(p. 64\)](#). After you create a role, you cannot rename it.

Create a Role (AWS Management Console)

If you use the AWS Management Console to work with IAM, you can create a role for an AWS service or for an AWS account by using the **Create New Role** wizard.

To create a role for an AWS service

1. In the **Navigation** pane, click **Roles**, and then click **Create New Role**.



2. In the **Role name** box, enter a role name that can help you identify the purpose of this role.

Because various entities might reference the role, you cannot edit the name of the role after it has been created.

3. Click **AWS Service Roles**, and then select the service that you will allow to assume this role.
4. Depending on the role that you selected, review the predefined policy or create a policy.
 - a. If the role included a predefined policy, you can modify the policy name or policy document, and then click **Continue** to review the role.
 - b. If the role that you selected doesn't include a predefined policy, select a method for creating the policy document by clicking **Select Policy Template**, **Policy Generator**, or **Custom Policy**.
 - Policy templates are predefined policies that have one or more permissions specified. If you are specifying permissions that match or are related to a template, select the template and then make any modifications on the next screen.
 - The policy generator helps you create permissions for a policy by providing drop-down menus where you can select services, actions, conditions, and keys. The generator creates the policy document for you.
 - If you want to fully write and customize a policy, select **Custom Policy**.
 - c. How you complete the next step depends on the method you selected to create the policy.
 - If you are using a template to create the policy, review the policy content in the dialog box.
 - If you are using the policy generator, select the appropriate **Effect**, **AWS Service**, and **Actions** options, enter the ARN (if applicable), and add any conditions you want to include. Then click **Add Statement**. You can add as many statements as you want to the policy. When you are finished adding statements, click **Continue**.

AWS Identity and Access Management Using IAM

Create a Role (AWS Management Console)

Create Role Cancel

SET PERMISSIONS

Edit Permissions

The policy generator enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see [Key Concepts in Using AWS Identity and Access Management](#).

Effect Allow Deny

AWS Service

Actions

Amazon Resource Name (ARN) [Add Conditions \(Optional\)](#)

Add Statement

[Back](#) [Continue](#)

- If you are using a custom policy, enter a name for the policy under **Policy Name** and write the policy or paste the policy document from your text editor into the **Policy Document** box.

Create Role Cancel

SET PERMISSIONS

Verify Role Permissions

You can customize permissions by editing the following policy document. For more information about the access policy language, see [Key Concepts in Using AWS Identity and Access Management](#).

Policy Name

Policy Document

```
{  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "EC2:Describe*",  
      "Resource": "*"  
    },  
    {  
      "Effect": "Allow",  
      "Action": "elasticloadbalancing:Describe*",  
      "Resource": "*"  
    }  
  ]  
}
```

[Back](#) [Continue](#)

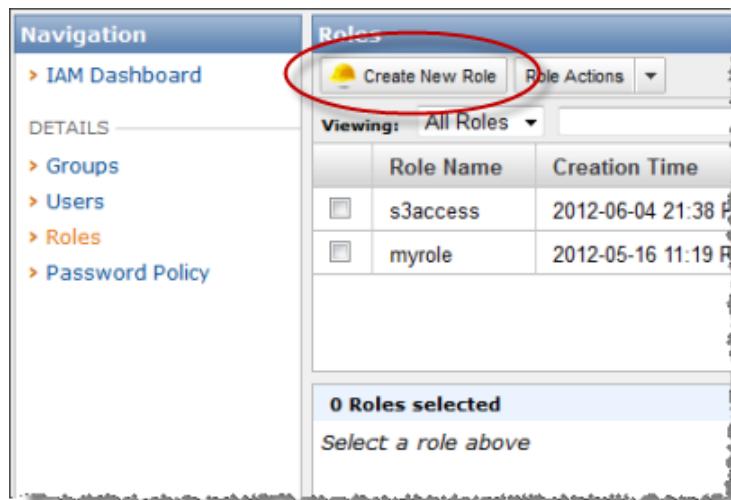
Note

There are limitations on policy names and on policy size. For information about policy limitations, see [Limitations on IAM Entities \(p. 64\)](#).

5. Click **Continue** to review the role.
6. After reviewing the role, click **Create Role** to complete the wizard.

To create a role that IAM users can assume

1. In the **Navigation** pane of the console, click **Roles**, and then click **Create New Role**.



2. In the **Role name** box, enter a role name that can help you identify the purpose of this role.

Because various entities might reference the role, you cannot edit the name of the role after it has been created.

3. Click **Roles for Cross-Account Access**, and then select the type of role that you want to create.
4. Specify an AWS account ID that you want to establish trust with.

Any IAM user from the trusted AWS account can assume this role, as long as they are granted permission to call the role.

5. If you selected a role for a third party, enter the external ID provided by the third party.

For more information about the external ID, see [About the External ID](#).

6. Click **Continue**.
7. Set the permissions for the role to specify what actions can be done on specific resources (similar to setting permissions for IAM groups).

The permissions that you specify are available to entities that assumes the role. By default, roles have no permissions.

- a. Choose the method for creating the policy document by clicking **Select Policy Template**, **Policy Generator**, or **Custom Policy**, and then click **Select**.
 - Policy templates are predefined policies that have one or more permissions specified. If you are specifying permissions that match or are related to a template, select the template and then make any modifications on the next screen.

AWS Identity and Access Management Using IAM

Create a Role (AWS Management Console)

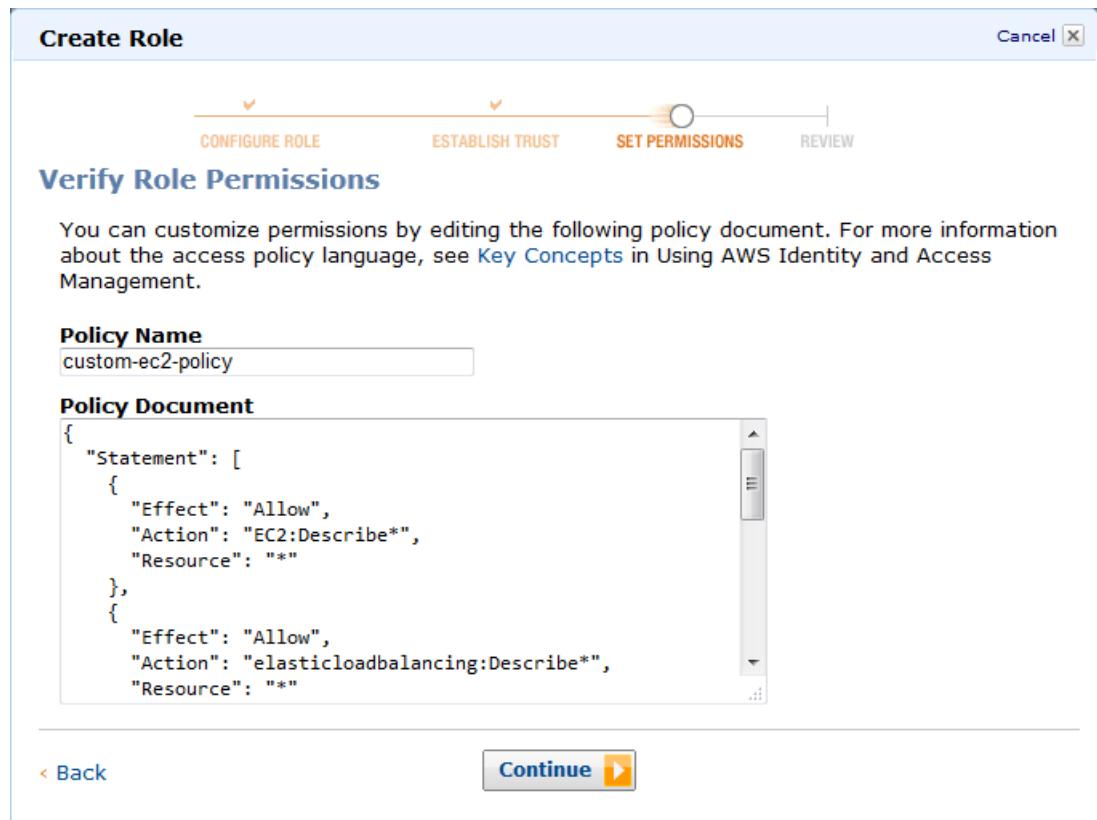
- The policy generator helps you create permissions for a policy by providing drop-down menus where you can select services, actions, conditions, and keys. The generator creates the policy document for you.
 - If you want to fully write and customize a policy, select **Custom Policy**.
- b. How you complete the next step depends on the method you selected to create the policy.
- If you are using a template to create the policy, review the policy content in the dialog box.
 - If you are using the policy generator, select the appropriate **Effect**, **AWS Service**, and **Actions** options, enter the ARN (if applicable), and add any conditions you want to include. Then click **Add Statement**. You can add as many statements as you want to the policy. When you are finished adding statements, click **Continue**.

The screenshot shows the 'Create Role' wizard at the 'Edit Permissions' step. The top navigation bar includes 'Cancel' and tabs for 'CONFIGURE ROLE', 'ESTABLISH TRUST', 'SET PERMISSIONS' (which is active and highlighted in orange), and 'REVIEW'. Below the tabs, a section titled 'Edit Permissions' contains the following fields:

- Effect:** A radio button group showing 'Allow' (selected) and 'Deny'.
- AWS Service:** A dropdown menu set to 'AWS CloudFormation'.
- Actions:** A dropdown menu set to '-- Select Actions --'.
- Amazon Resource Name (ARN):** An input field containing '*'.
- Add Conditions (Optional):** A link to add optional conditions.
- Add Statement:** A blue button to add a new policy statement.

At the bottom of the form are 'Back' and 'Continue' buttons.

- If you are using a custom policy, enter a name for the policy under **Policy Name** and write the policy or paste the policy document from your text editor into the **Policy Document** box.



Note

There are limitations on policy names and on policy size. For information about policy limitations, see [Limitations on IAM Entities \(p. 64\)](#).

- c. Click **Continue** to review the role.
8. After reviewing the role, click **Create Role** to complete the wizard.

Create a Role (CLI)

This topic shows the process for using the IAM CLI to create a role.

To create a role with the CLI

1. Create a role by entering the following command: `iam-rolecreate -r role_name [-f policy_document_file | -s service_endpoint]`.
For the role's trust policy, you can specify a file location or a service endpoint. The policy document file can contain multiple entities that can assume the role and any conditions for assuming the role.
2. Associate a policy to the role by entering the following command: `iam-roleaddpolicy -r role_name -p policy_name -e Allow_or_Deny {-a action ...} {-c ARN ...}`.
3. If you are launching an Amazon EC2 instance with this role, create and add the role to an instance profile by entering the following command: `iam-instanceprofilecreate -r role_name -s instance_profile_name`.

An instance profile is a container for a role. Each instance profile can contain only one role. For more information about instance profiles, see [Instance Profiles \(p. 142\)](#).

Create a Role (API)

This topic shows the process for using the IAM API to create a role.

To create a role with the API

1. Create a role by calling [CreateRole](#) with the role name and role trust policy.
2. Associate a policy to the role by calling [PutRolePolicy](#) with the policy document, policy name, and role name.
3. If you are launching an Amazon EC2 with this role, you must complete the following steps:
 - a. Create an instance profile by calling [CreateInstanceProfile](#) with the instance profile name.
 - b. Add the role to the instance profile by calling [AddRoleToInstanceProfile](#) with the role name and instance profile name.

An instance profile is a container for a role. Each instance profile can contain only one role. For more information about instance profiles, see [Instance Profiles \(p. 142\)](#).

Related Information

For information about how to launch an instance with the role you just created, see [Using IAM roles with Amazon EC2 instances](#) in the *Amazon Elastic Compute Cloud User Guide*.

For information about cross-account access with roles, see [Enabling Cross-Account API Access \(p. 142\)](#).

For information about managing policies, see [Managing IAM Policies \(p. 173\)](#).

For more information about IAM CLI commands or API actions, see [AWS Identity and Access Management Command Line Interface Reference](#) or [AWS Identity and Access Management API Reference](#).

Modifying a Role

You can edit who can access a role and the permissions associated with the role by using the AWS Management Console, CLI, or API. Any changes to the role are propagated in near realtime.

Topics

- [Modify a Role \(AWS Management Console\) \(p. 157\)](#)
- [Modify a Role \(CLI\) \(p. 158\)](#)
- [Modify a Role \(API\) \(p. 159\)](#)

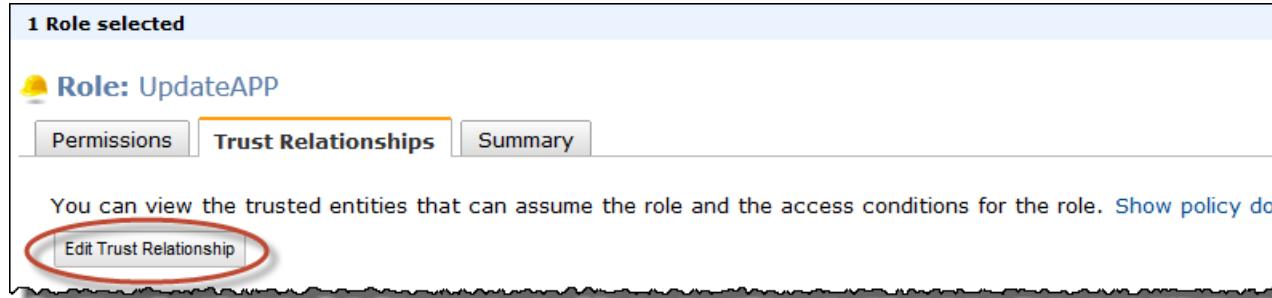
Modify a Role (AWS Management Console)

To edit who can access the role

1. In the **Navigation** pane of the IAM console, click **Roles**.

A list of roles for your account is displayed in the **Roles** pane.

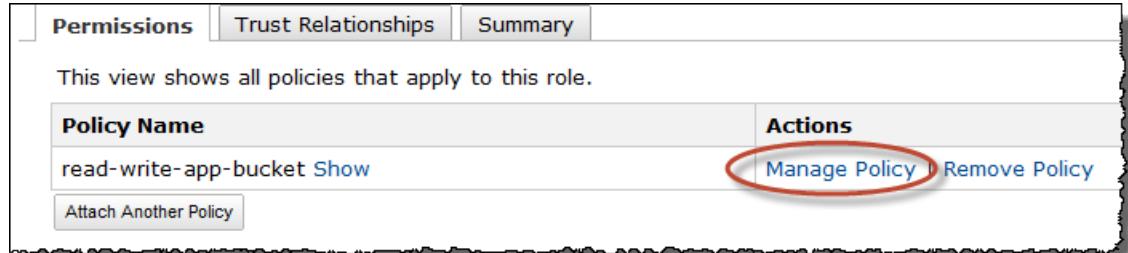
2. Select the role that you want to modify, and then in the properties pane for the role, click the **Trust Relationship** tab.
3. In the **Trust Relationship** tab, click **Edit Trust Relationship**.



4. Edit, add, or remove trusted entities, and then click **Continue**.
5. Edit, add, or remove conditions for assuming the role, and then click **Continue**.
6. Review your changes, and then click **Save Role**.

To edit the permissions associated with a role

1. In the **Navigation** pane of the console, click **Roles**.
2. Select the role that you want to modify, and in the properties pane for the role, click the **Permissions** tab.
3. Find the policy that you want to edit, and in the **Actions** column, click **Manage Policy**.



4. Edit the policy document, and then click **Apply Policy**.

Modify a Role (CLI)

To edit who can access the role

1. If you don't know the name of the role that you want to modify, list the roles in your account by entering the following command: `iam-rolelistbypath`.

A list of roles with their Amazon Resource Name (ARN) is displayed. Use the role name, not the ARN, to refer to roles with the CLI commands. For example, if a role had the following ARN:
`arn:aws:iam::123456789012:role/myrole`, you refer to the role as `myrole`.
2. (Optional) To view the current trust policy for a role, enter the following command:
`iam-rolegetattributes -r myrole`.
3. Create a text file with the updated trust policy.

You can use any text editor to construct the policy. The trust policy must contain the entities that can assume the role. Optionally, you can include any conditions for assuming the role.

4. To update the trust policy, enter the following command: `iam-roleupdateassumepolicy -r myrole -f policy_document_file`.

Changes to the role are propagated in near real time. You can verify that the role has been updated by entering the following command: `iam-rolegetattributes -r myrole`.

To edit the permissions associated with a role

1. If you don't know the name of the role that you want to modify, list the roles in your account by entering the following command: `iam-roledislistbypath`.

A list of roles with their Amazon Resource Name (ARN) is displayed. Use the role name, not the ARN, to refer to roles with the CLI commands. For example, if a role had the following ARN:
arn:aws:iam::123456789012:role/myrole, you refer to the role as `myrole`.

2. (Optional) To view the current permissions associated with a role, enter the following command: `iam-roledislistpolicies -r myrole -v`.
3. Create a text file with the updated permissions for the role.

You can use any text editor to construct the policy. The policy defines the actions that are allowed or denied on a given resource.

4. To update the permissions on the role, enter the following command: `iam-roleuploadpolicy -r myrole -p policy_name -f policy_document_file`.

Changes to the role are propagated in near real time. You can verify that the role has been updated by entering the following command: `iam-roledislistpolicies -r myrole`.

Modify a Role (API)

To edit who can access the role

1. If you don't know the name of the role that you want to modify, list the roles in your account by calling `ListRole`.

A response is returned with information for each role. Use the role name, not the ARN, to refer to roles with the API actions.

2. (Optional) To view the current trust policy for a role, call `GetRole` with the role name.
3. Call `UpdateAssumeRolePolicy` to update the trust policy.

You must include the policy and role name with the API call. The policy document must contain the entities that can assume the role. Optionally, you can include any conditions for assuming the role.

Changes to the role are propagated in near realtime. You can verify that role has been updated by calling `GetRole`.

To edit the permissions associated with a role

1. If you don't know the name of the role that you want to modify, list the roles in your account by calling `ListRole`.

A response is returned with information for each role. Use the role name, not the ARN, to refer to roles with the API actions.

2. (Optional) If you want to view the current permissions associated with a role, call `GetRolePolicy`

3. Call [PutRolePolicy](#) to update the permissions on the role.

You must include the policy, policy name, and role name with the API call. The policy defines the actions that are allowed or denied on a given resource.

Assuming a Role

If your AWS account is a trusted entity for a role, you can assume that role as an IAM user to access AWS resources that you might not have explicit permissions to. When you assume a role, you receive temporary security credentials that you can use to sign requests to access AWS resources. The permissions of the temporary security credentials are defined in the policy associated with the role.

Important

Only IAM users can assume a role. If you use AWS account credentials, access is denied.

Before you can assume a role, you must ensure that role includes your AWS account as a trusted entity and that you have permission to call the AWS Security Token Service (STS) AssumeRole API. You must also have the Amazon Resource Name (ARN) for the role.

To assume a role

- Call [AssumeRole](#) and pass the role ARN.

Depending on the conditions that are set in the role, you might also need to pass additional parameters, such as an external ID.

You can also include a supplemental policy that narrows the permissions of the role's temporary security credentials. By narrowing the permissions, you can maintain the least amount of privileges.

In the response for `AssumeRole`, you receive temporary security credentials that you can use to access AWS resources that are defined by role.

Related Information

- For a detailed scenario about using roles to delegate access, see [Enabling Cross-Account API Access Walkthrough \(p. 144\)](#).
- To see a list of services that can use roles, see [Using Temporary Security Credentials to Access AWS](#) in the *Using Temporary Security Credentials* guide.
- For more information about the external ID, see [About the External ID](#).

Deleting Roles or Instance Profiles

If you are not using a role, delete the role and its associated permissions so that you don't have an unused entity that is not actively monitored or maintained.

You can also remove roles from instance profiles.

Caution

Make sure you do not have any Amazon EC2 instances running with the role or instance profile you are about to delete. Deleting a role or instance profile that is associated with a running instance will break any applications running on the instance.

Topics

- [Delete a Role \(AWS Management Console\) \(p. 161\)](#)

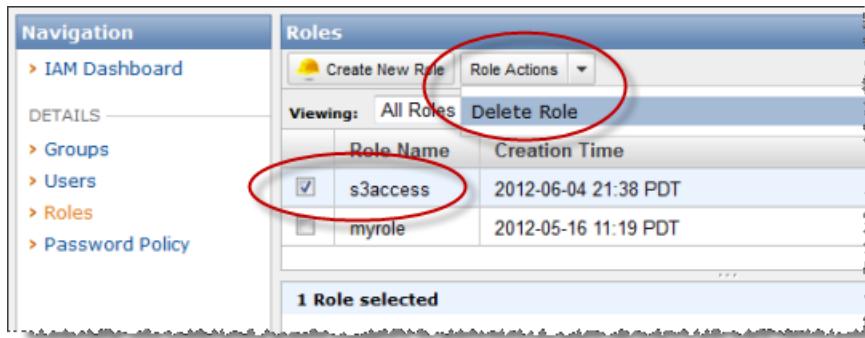
- [Delete a Role \(CLI and API\) \(p. 161\)](#)
- [Related Information \(p. 162\)](#)

Delete a Role (AWS Management Console)

When you use the AWS Management Console to delete a role, IAM also automatically deletes the policies associated with the role, and the instance profile that contains the role.

To delete a role

1. In the **Navigation** pane of the IAM Dashboard, click **Roles**.
2. Select the role you want to delete.
3. From the **Role Actions** list, select **Delete Role**.



4. Review your changes, and then click **Yes, Delete**.

Note

You cannot use the console to delete an instance profile, except when you delete it as part of the process of deleting a role as described in the preceding procedure. To delete an instance profile without also deleting the role, you must use the CLI or API. For information about using the CLI or API to remove a role from an instance profile, see [Delete a Role \(CLI and API\) \(p. 161\)](#).

Delete a Role (CLI and API)

When you use the IAM CLI or API to delete a role, you must first delete the policies associated with the role. Also, if you want to delete the associated instance profile that contains the role, you must delete it separately.

To delete a role

1. If you don't know the name of the role that you want to delete, list the roles in your account by entering the following command: `iam-rolelistbypath`.
A list of roles with their Amazon Resource Name (ARN) is displayed. Use the role name, not the ARN, to refer to roles with the CLI commands. For example, if a role had the following ARN:
`arn:aws:iam::123456789012:role/myrole`, you refer to the role as `myrole`.
2. Remove the role from all instance profiles that the role is in.
 - a. List all instance profiles that the role is associated with by entering the following command:
`iam-instanceprofilelistforrole -r myrole`.

- b. To remove the role from each instance profile, for each instance profile, enter the following command: `iam-instanceprofileremoverole -r myrole -s instance_profile_name`.
3. Delete all policies that are associated with the role.
 - a. List all policies that are in the role by entering the following command: `iam-roledlistpolicies -r myrole`.
 - b. To delete each policy from the role, for each policy, enter the following command: `iam-roledelpolicy -r myrole -p policy_name`.
4. Delete the role by entering the following command: `iam-roledel -r myrole`.
5. If you are not using the instance profiles that were associated with the role, you can delete them by entering the following command: `iam-instanceprofiledel -s instance_profile_name`.

To delete a role

1. Remove the role from all instance profiles that the role is in by calling [RemoveRoleFromInstanceProfile](#).

You must pass the role name and instance profile name. You can list all instance profiles that a role is in by calling [ListInstanceProfilesForRole](#).

2. Delete all policies that are associated with the role by calling [DeleteRolePolicy](#).

You must pass the role name and policy name. You can list all policies for a role by calling [ListRolePolicies](#).

3. Delete the role by calling [DeleteRole](#).
4. If you are not using the instance profiles that were associated with the role, you can delete them by calling [DeleteInstanceProfile](#).

Related Information

For general information about instance profiles, see [Instance Profiles \(p. 142\)](#).

Troubleshooting Roles

Topics

- [AWS Management Console \(p. 163\)](#)
- [AWS Identity and Access Management APIs \(p. 163\)](#)
- [Amazon EC2 API \(p. 164\)](#)
- [Amazon EC2 Instances \(p. 164\)](#)
- [AWS Service API Requests \(p. 166\)](#)
- [AWS Security Token Service AssumeRole API Requests \(p. 166\)](#)

This section offers solutions to common issues you might encounter while working with IAM roles.

AWS Management Console

When attempting to launch an instance, I don't see the role I expected to see in the Amazon EC2 console IAM Role list.

Check the following:

- If you are signed in as an IAM user, check that you have permission to call `ListInstanceProfiles`. For information about the permissions necessary to work with roles, see "How do I get started?" in [Granting Applications that Run on Amazon EC2 Instances Access to AWS Resources \(p. 139\)](#). For information about adding permissions to a user, see [Managing IAM Policies \(p. 173\)](#).
- If the role was created using the IAM CLI or API, the role might not be contained in an instance profile. The IAM console creates an instance profile for every role created. The CLI and API do not. You will need to use the CLI or API to find the instance profile that contains the role. If an instance profile doesn't exist, then you need to create one. Creating instance profiles is described as part of the process of using the CLI or API to create roles in [Creating a Role \(p. 151\)](#).
- If a role was created using the IAM CLI or API, the role might have a different name than the instance profile. In this case, the name you need to select from the list is the instance profile name. Note that the EC2 console lists the instance profiles for the account even though the list is named **IAM Role**.

When I attempt to launch an instance with a role, I get an `AccessDenied` error.

See the following topic, [Amazon EC2 API \(p. 164\)](#).

AWS Identity and Access Management APIs

When attempting to call `CreateRole`, `PutRolePolicy`, or `CreateInstanceProfile`, I get an `AccessDenied` error.

If you are making requests as an IAM user, verify that you have permission to call `CreateRole`, `PutRolePolicy`, and `CreateInstanceProfile`. For more information about the permissions necessary to work with roles, see "How Do I Get Started?" in [Granting Applications that Run on Amazon EC2 Instances Access to AWS Resources \(p. 139\)](#). For information about adding permissions to a user, see [Managing IAM Policies \(p. 173\)](#).

When I attempt to call the `AddRoleToInstanceProfile`, I get an `AccessDenied` error.

If you are making requests as an IAM user, verify that you have the following permissions:

- `iam:AddRoleToInstanceProfile` with the resource matching the instance profile ARN (for example, `arn:aws:iam::1234567890:instance-profile/MyInstanceProfile`).
- `iam:PassRole` with the resource matching the role ARN (for example, `arn:aws:iam::1234567890:role/MyRole`).

For more information about the permissions necessary to work with roles, see "How Do I Get Started?" in [Granting Applications that Run on Amazon EC2 Instances Access to AWS Resources \(p. 139\)](#). For information about adding permissions to a user, see [Managing IAM Policies \(p. 173\)](#).

Amazon EC2 API

When I attempt to launch an instance with a role, I get an `AccessDenied` error.

Check the following:

- Launch an instance without an instance profile. This will help ensure that the problem is limited to IAM roles for EC2 instances.
- Call the IAM `GetInstanceProfile` action to ensure that you are using a valid instance profile name or a valid instance profile ARN. For more information, see [Using IAM roles with Amazon EC2 instances](#).
- Call the IAM `GetInstanceProfile` action to ensure that the instance profile has a role. Empty instance profiles will fail with an `AccessDenied` error. For more information about creating a role, see [Creating a Role \(p. 151\)](#).
- If you are making requests as an IAM user, verify that you have the following permissions:
 - `ec2:RunInstances` with a wildcard resource ("*")
 - `iam:PassRole` with the resource matching the role ARN (for example, `arn:aws:iam::1234567890:role/MyRole`)

For more information about the permissions necessary to work with roles, see "How Do I Get Started?" in [Granting Applications that Run on Amazon EC2 Instances Access to AWS Resources \(p. 139\)](#). For information about adding permissions to a user, see [Managing IAM Policies \(p. 173\)](#).

Amazon EC2 Instances

I can't access the security credentials on my EC2 instance.

Check the following:

- Can you access another part of the instance metadata service (IMDS)? If not, check that you have no firewall rules blocking access to requests to IMDS.

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/hostname; echo
```

- Does the `iam` subtree of IMDS exist? If not, verify that your instance has an IAM instance profile associated with it by calling `ec2:DescribeInstances`.

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam; echo
```

- Does the `info` document in the IAM subtree indicate an error? (The following sections describe what to do if an error is indicated.)

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam/info; echo
```

My Amazon EC2 instance has an IAM instance profile, but I don't see credentials.

Check the `info` document in the IAM subtree for an indication of an error. (The following sections describe what to do if an error is indicated.)

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam/info; echo
```

The credentials on my instance are for the wrong role.

If the role in the instance profile was replaced recently, your application will need to wait for the next automatically scheduled credential rotation before credentials for your role become available.

The `iam/info` document indicates "Code": "InstanceProfileNotFound".

This means that your IAM instance profile has been deleted and Amazon EC2 can no longer provide credentials to your instance. You will need to terminate your instances and restart with a valid instance profile.

If an instance profile with that name exists, check that the instance profile wasn't deleted and another recreated with the same name.

To verify the status of the instance profile:

1. Call the IAM `GetInstanceProfile` action to get the `InstanceProfileId`.
2. Call the Amazon EC2 `DescribeInstances` action to get the `IamInstanceProfileId` for the instance.
3. Verify that the `InstanceProfileId` from the IAM action matches the `IamInstanceProfileId` from the Amazon EC2 action.

If the IDs are different, then the instance profile attached to your instances is no longer valid. You will need to terminate your instances and restart with a valid instance profile.

The `iam/info` document indicates a success but indicates

"Message": "Instance Profile does not contain a role..."

This means that the role has been removed from the instance profile via the IAM `RemoveRoleFromInstanceProfile` action. You can use the IAM `AddRoleToInstanceProfile` action to attach a role to the instance profile. Your application will need to wait until the next scheduled refresh to have access to credentials for the role.

The `iam/security-credentials/[role-name]` document indicates

"Code": "AssumeRoleUnauthorizedAccess".

This means that Amazon EC2 does not have permission to assume the role. Permission to assume the role is controlled by the assume role policy attached to the role. Use the IAM `UpdateAssumeRolePolicy` API to update the assume role policy.

```
{"Version": "2008-10-17", "Statement": [{"Effect": "Allow", "Principal": {"Service": ["ec2.amazonaws.com"]}, "Action": ["sts:AssumeRole"]}]}
```

Your application will need to wait until the next automatically scheduled refresh to have access to credentials for the role.

AWS Service API Requests

I cannot successfully make requests to an AWS service.

This section is dependent on the client toolkit you use to call AWS service APIs. For information about the AWS SDKs, refer to the AWS SDK documentation on the [AWS Documentation page](#).

- Verify that the service accepts temporary security credentials. See [Using Temporary Security Credentials to Access AWS](#).
- Verify that your requests are being signed correctly and that the request is well-formed. Consult your toolkit documentation for more information, or see [Using Temporary Security Credentials to Authenticate an AWS Request](#).
- Verify that your temporary security credentials haven't expired. For more information, see [Using Temporary Security Credentials](#).
- Verify that the permissions on the role are correct. Use the IAM `PutRolePolicy` API to update role permissions. For information about the permissions necessary to work with roles, see "How Do I Get Started?" in [Granting Applications that Run on Amazon EC2 Instances Access to AWS Resources \(p. 139\)](#). For information about adding permissions to a user, see [Managing IAM Policies \(p. 173\)](#).
- If you are using resource policies to manage permissions, verify that permission is granted to the role. For example, the following policy allows a role access to an Amazon S3 bucket.

```
{  
    "Version": "2008-10-17",  
    "Statement": [  
        {  
            "Sid": "BucketPolicy",  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": [ "arn:aws:iam::111122223333:role/MyRole" ]  
            },  
            "Action": [ "s3:PutObject" ],  
            "Resource": [ "arn:aws:s3:::MyBucket/*" ]  
        }]  
    }  
}
```

This policy would apply to any EC2 instances that have assumed the role. Granting access via resource policies for a particular EC2 instance is not supported.

AWS Security Token Service AssumeRole API Requests

I cannot successfully assume a role.

- Verify that the AWS account that you are calling `AssumeRole` from is a trusted entity for the role that you are assuming. Trusted entities are defined in a role's trust policy.
- Verify that you meet all the conditions that are specified in the role. Conditions are defined as part of the role's trust relationship. A condition can specify an expiration date, an external ID, or that request come from specific IP addresses.

- Verify that your IAM policy grants you privilege to call `sts:AssumeRole`. The policy must also specify the role's Amazon Resource Name (ARN) that you are assuming.

Cross-Account Access Using Resource-Based Policies

Topics

- [Access to AWS Resources \(p. 168\)](#)
- [Delegating AWS Account Permissions to IAM Users \(p. 169\)](#)

You can enable trusted entities outside of your AWS account to access certain resources within your AWS account. In addition, when the trusted entity is another AWS account, that account can delegate access to its AWS Identity and Access Management (IAM) users. This process is referred to as *cross-account access*. Cross-account access enables you to share access to your resources with users under other AWS accounts.

Cross-account access is a two-step process. First you give another AWS account access to specific resources, and then that AWS account delegates access to its users. An AWS account can delegate access only to the extent that it has been granted access. This section links to information about enabling access to particular AWS account resources, and then describes how to use IAM to delegate access to users within an account.

Access to AWS Resources

Certain AWS services enable you to grant access to your AWS resources to trusted entities outside of your AWS account. For example, you can use Amazon S3 ACLs and policies to grant other AWS accounts access to your Amazon S3 buckets.

The following table shows the AWS services that use policies to control access to resources, lists the resources available, and provides links to detailed information about how to use policies with each service.

Note

You can also enable cross-account access by using IAM roles. With a single role, you can manage access and permissions to multiple AWS services. IAM roles also supports additional AWS services that are not available with resource-based policies. For more information about using roles for cross-account access, see [Enabling Cross-Account API Access \(p. 142\)](#). For a list of services that support roles, see [Using Temporary Security Credentials to Access AWS](#) in the *Using Temporary Security Credentials* guide.

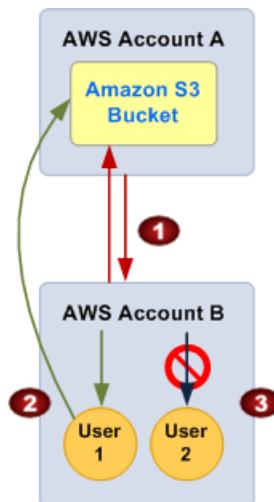
AWS service	Resource	Reference
Amazon S3	buckets	The policy is attached to the bucket, but the policy controls access to both the bucket and the objects in it; you can use ACLs to control access to individual objects. For more information, go to Access Control in the <i>Amazon Simple Storage Service Developer Guide</i> .
Amazon SNS	topics	For more information, go to Appendix: Managing Access to Your Amazon SNS Topics in the <i>Amazon Simple Notification Service Getting Started Guide</i> .
Amazon SQS	queues	For more information, go to Appendix: The Access Policy Language in the <i>Amazon Simple Queue Service Developer Guide</i> .

Important

Give access only to entities you trust, and give the minimum amount of access necessary. Granting access to an AWS account enables it to delegate access to its users. An AWS account can delegate access only to the extent that it has access.

Delegating AWS Account Permissions to IAM Users

An AWS account with access to another AWS account's resources can use an IAM policy to delegate access to the users under its account, as described in the following figure and table. (For information about permissions, policies, and the access policy language you use to write policies, see [Permissions and Policies \(p. 30\)](#).)



①	Account A gives Account B full access to Account A's Amazon S3 bucket. As a result, Account B can perform any action on Account A's bucket, and Account B can grant access to users under Account B.
②	Account B gives User 1 read access to Account A's Amazon S3 bucket. User 1 can view the objects in Account A's bucket. The level of access Account B can delegate is equivalent to, or less than, the access it has.

3

Account B does not give access to User 2. Because User 2 has not been granted access to Account A's Amazon S3 bucket by Account B, by default, User 2 cannot access the bucket or the objects in the bucket.

Important

In the preceding example, be aware that if Account B had used wildcards (*) to give a user full access to all its resources, that user would automatically have access to any resources that Account B has access to, including access granted by another account to its own resources. In this case, the user would have access to Account A's resources even though Account B did not specifically apply the permission.

Policies are late-binding. This means that IAM evaluates a user's permissions at the time the user makes a request. Therefore, if you use wildcards (*) to give users full access to your resources, users are able to access any resources your AWS account has access to, even resources you add or gain access to after creating the user's policy.

How to Delegate AWS Account Permissions

To delegate permissions to users under your AWS account, you attach a policy to the user or group that you want to delegate your permissions to. You can delegate permissions equivalent to, or less than, the permissions you have.

For example, if your account has full access to the resources of another AWS account, you can delegate full access, list access, or any other partial access to users under your AWS account. If you have been granted list access only, you can delegate only list access. This means that where you have list access only, but you grant users under your AWS account full access, your users will have list access only. (For information about attaching a policy to a user or group, see [Managing IAM Policies \(p. 173\)](#).)

Example 1: Use a policy to delegate access to another AWS account's Amazon S3 bucket

In this example, Account A uses a bucket policy to grant Account B full access to Account A's Amazon S3 bucket. Then, Account B creates an IAM user policy to delegate access to Account A's bucket to one of the users under Account B.

Account A's bucket policy might look like the following policy. In this example, Account A's Amazon S3 bucket is named *mybucket*, and Account B's account number is 1111-2222-3333. Account A uses Amazon S3 to implement this policy.

```
{  
    "Statement" : {  
        "Effect": "Allow",  
        "Sid": "AccountBAccess1",  
        "Principal" : {  
            "AWS": "111122223333"  
        },  
        "Action": "s3:*",  
        "Resource": "arn:aws:s3:::mybucket/*"  
    }  
}
```

Note that, alternatively, Account A could use Amazon S3 Access Control Lists (ACLs) to grant Account B access to an Amazon S3 bucket or a single object within a bucket. In this case, the only thing that would change is how Account A grants access to Account B. Account B would still use a policy to delegate access to a user under Account B, as described in the second part of this example. For more information about controlling access on Amazon S3 buckets and objects, go to [Access Control](#) in the *Amazon Simple Storage Service Developer Guide*.

The next example shows the IAM user (or group) policy that Account B might create to delegate read access to a user under Account B. In this policy, the `Action` element is explicitly defined to allow only `List` actions, and the `Resource` element of this policy matches the `Resource` for the bucket policy implemented by Account A.

Account B implements this policy by using IAM to attach it to the appropriate user (or group) under Account B.

```
{  
    "Statement": {  
        "Effect": "Allow",  
        "Action": "s3>List*",  
        "Resource": "arn:aws:s3:::mybucket/*"  
    }  
}
```

Example 2: Delegate access to another AWS account's Amazon SQS queue

In this example, Account A has an Amazon SQS queue and Account A uses a queue policy to grant queue access to Account B. Then, Account B uses an IAM user policy to delegate access to a user under Account B.

The following example queue policy gives Account B `SendMessage` and `ReceiveMessage` permission for Account A's queue named `1234-5678-9012/queue1`, but only between noon and 3:00 p.m. on November 30, 2011. Account B's account number is 1111-2222-3333. Account A uses Amazon SQS to implement this policy.

```
{  
    "Statement": {  
        "Effect": "Allow",  
        "Action": ["sqs SendMessage", "sqs ReceiveMessage"],  
        "Principal": {  
            "AWS": "111122223333"  
        }  
        "Resource": "arn:aws:sqs:*:/123456789012:queue1",  
        "Condition": {  
            "DateGreaterThan": {  
                "aws:CurrentTime": "2011-11-30T12:00Z"  
            }  
            "DateLessThan": {  
                "aws:CurrentTime": "2011-11-30T15:00Z"  
            },  
        },  
    }  
}
```

Account B's policy delegating access to a user under Account B might look like the following example. Account B uses IAM to attach this policy to a user (or group).

```
{  
    "Statement": {  
        "Effect": "Allow",  
        "Action": "sqs:*",  
        "Resource": "arn:aws:sqs:*:/123456789012:queue1"  
    }  
}
```

In the preceding IAM user policy example, Account B uses a wildcard to grant its user access to Account A's queue. But, because Account B can delegate access only to the extent that Account B has been granted access, Account B's user can access the queue only between noon and 3:00 p.m. on November 30, 2011, and the user can only perform SendMessage and ReceiveMessage actions, as defined in Account A's Amazon SQS queue policy.

Example 3: Cannot delegate access when the account is denied access

By default, other AWS accounts and their users cannot access your AWS account resources. But, when you use a policy to explicitly deny an AWS account access to your resources, the deny propagates to the users under that account regardless of whether the users have existing policies granting them access. This means that an AWS account cannot delegate access to another account's resources if the other account has explicitly denied access to the user's parent account.

For example, Account A writes a bucket policy on Account A's Amazon S3 bucket that explicitly denies Account B access to Account A's bucket. But Account B writes an IAM user policy that grants a user under Account B access to Account A's bucket. The explicit deny applied to Account A's Amazon S3 bucket propagates to the users under Account B and overrides the IAM user policy granting access to the user under Account B. (For detailed information how permissions are evaluated, see [Evaluation Logic \(p. 37\)](#).)

Account A's bucket policy might look like the following policy. In this example, Account A's Amazon S3 bucket is named *mybucket*, and Account B's account number is 1111-2222-3333. Account A uses Amazon S3 to implement this policy.

```
{  
    "Statement" : {  
        "Effect": "Deny",  
        "Sid": "AccountBDeny",  
        "Principal" : {  
            "AWS": "111122223333"  
        },  
        "Action": "s3:*",  
        "Resource": "arn:aws:s3:::mybucket/*"  
    }  
}
```

Account B implements the following IAM user policy by using IAM to attach it to the a user under Account B.

```
{  
    "Statement":{  
        "Effect": "Allow",  
        "Action": "s3:*",  
        "Resource": "arn:aws:s3:::mybucket/*"  
    }  
}
```

Because Account A's bucket policy explicitly denies Account B access to *mybucket*, the denial propagates to the user under Account B, and Account B's IAM user policy granting Account B's user access to Account A's bucket has no effect. The user cannot access Account A's bucket.

Managing IAM Policies

Topics

- [Managing Policies \(AWS Management Console\) \(p. 174\)](#)
- [Managing Policies \(CLI\) \(p. 177\)](#)
- [Managing Policies \(API\) \(p. 177\)](#)

This section describes creating, viewing, listing, and deleting policies. For general information about what policies are and how they work, see [Overview of Policies \(p. 32\)](#).

When you want to add a permission to an IAM entity such as a user, group, or role, you create a policy in the access policy language that contains the permission, and then attach the policy to the entity. You can attach multiple policies to an entity. Or, instead of creating a new policy, you can edit an existing policy to add the permission. Policy names must be unique to the entity the policy is attached to. For information about how permissions are evaluated when permissions are applied to an IAM entity, see [Evaluation Logic](#).

Note

There are limits on the cumulative size of the policy and on the policy name. For more information, see [Limitations on IAM Entities \(p. 64\)](#).

You can get a list of the policies attached to a user, group, or role. But, there's no single IAM action that gets a list of all the policies you've created for all entities in the AWS account.

You can delete a policy at any time. If you use the API or CLI to delete a user, group, or role, you must delete the attached policies separately. For more information, see [Deleting a User from Your AWS Account \(p. 82\)](#), [Deleting a Group \(p. 91\)](#), or [Deleting Roles or Instance Profiles \(p. 160\)](#).

How you create, view, list, or delete a policy depends on which interface you're using to access IAM. The interface-specific details are covered in the following sections.

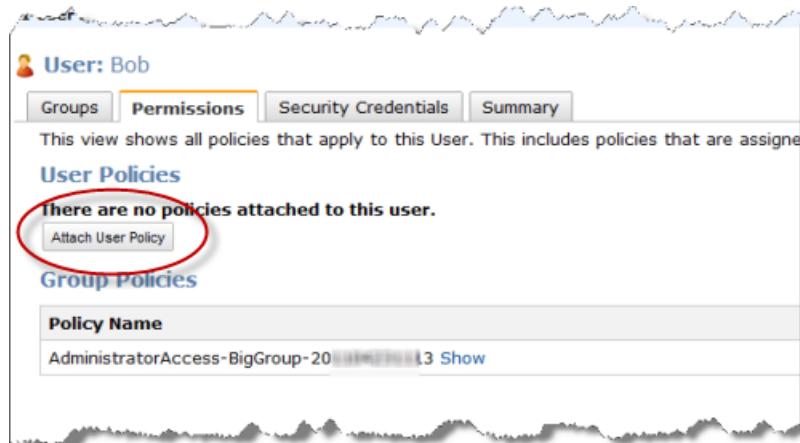
Tip

Policy templates and the policy generator are available in the AWS Management Console. You can use the [AWS Policy Generator online](#) to create policies for AWS products without accessing the console. For more information about the access policy language, see [The Access Policy Language \(p. 33\)](#).

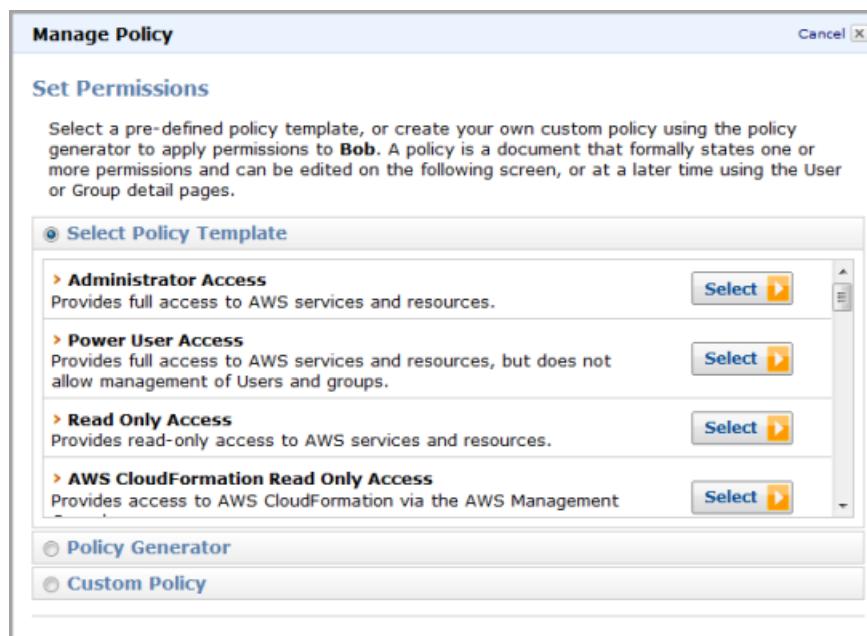
Managing Policies (AWS Management Console)

To create a policy and attach it to a group, user, or role

1. In the Navigation pane, click **Groups, Users, or Roles**.
2. Select the group, user, or role you want to create a policy for, and then select the **Permissions** tab.
3. If you are creating a policy for a user, click **Attach User Policy**. If you are creating a policy for a group or role, click **Attach Policy**.



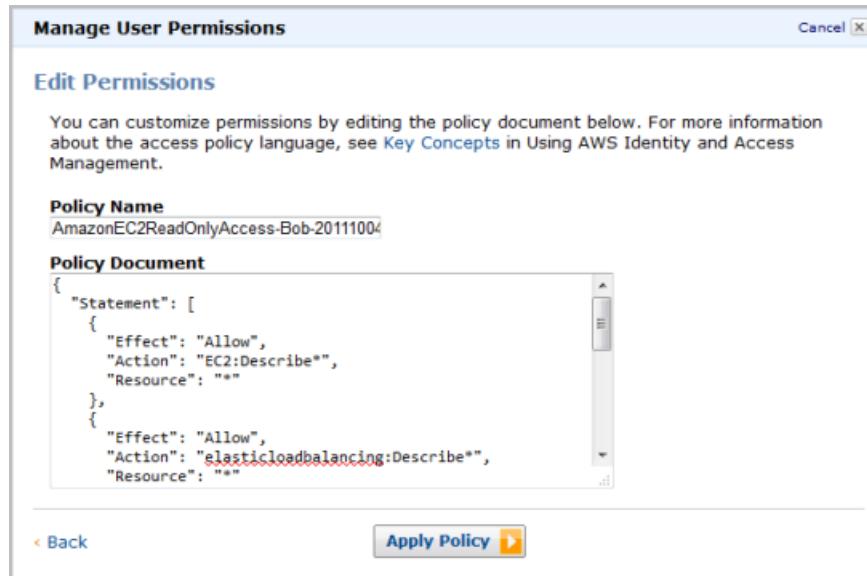
4. Choose the method for creating the policy document by clicking either **Select Policy Template**, **Policy Generator**, or **Custom Policy**. Then click **Select**.



5. In the **Edit Permissions** screen, you can name the policy and create or edit your policy document. Or if you are using the policy generator to create your policy, select the appropriate **Effect**, **AWS Service**, and **Actions** options, enter the ARN (if applicable), and add any conditions you want to

include. Then click **Add Statement**. You can add as many statements as you want to the policy. When you are finished adding statements, click **Continue**.

6. When you are satisfied with the policy, click **Apply Policy**. IAM applies the policy to the selected entity.



To view a policy or a list of policies associated with a user, group, or role

- In the **Navigation** pane, click **Users**, **Groups**, or **Roles**, and then select the name of the entity whose policies you want to view.

User Name	Groups	Password	Access Keys	Creation Time
Ben	1		1 active	2012-06-03 16:32 PDT
Tom	1	✓	1 active, 1 inactive	2012-06-03 16:32 PDT
Colm	1		1 active	2012-06-03 16:32 PDT
Dobrod	1		1 active	2012-06-03 16:32 PDT

User Policies

Policy Name	Actions
Route53ReadOnly-Tom-201206042153 Show	Manage Policy Remove Policy

Group Policies

Policy Name	Group Name
AdministratorAccess-Ops-201206031634 Show	Ops

IAM lists all policies associated with the entity on the **Permissions** tab. If you want to see the contents of a policy, click **Show**.

Note

For users, under **User Policies**, you can see any policies attached to the user directly. IAM lists policies attached to any groups that the user belongs to under **Group Policies**. You cannot modify group policies from a user's **Permissions** tab. To create or modify group permissions, you must go to the **Permissions** tab for the group whose permissions you want to change.

Note

Currently, permission to assume a role is limited only to Amazon EC2 instances in your AWS account. The permissions described in this step control only what the entity who assumes the role can do.

To edit or delete a policy for a group, user, or role

1. In the **Navigation** pane, click **Groups**, **Users**, or **Roles**.
2. Select the group, user, or role you want to modify a policy for, and then select the **Permissions** tab.
3. If you're editing a policy, click **Manage Policy**. You can edit the policy in the **Edit Permissions** screen (shown in the preceding task), or, if you prefer to work from a policy template, click **View Policy Templates** and select a template. You can also use the policy generator to modify a policy.
4. To delete a policy, click **Remove Policy**. The policy is deleted and permissions associated with the policy no longer apply to the selected group, user, or role.

Managing Policies (CLI)

The following table lists which CLI command to use for each task you want to perform with a policy. For more information about the commands, go to the [AWS Identity and Access Management Command Line Interface Reference](#).

For an example of attaching a policy and listing policies, see [Adding a Policy to the Group \(p. 69\)](#).

Task	Command to Use
Attach a simple policy by passing in the contents of the policy as parameters in the command	iam-groupaddpolicy iam-useraddpolicy iam-roleaddpolicy
Attach a JSON policy document you've written	iam-groupuploadpolicy iam-useruploadpolicy iam-roleuploadpolicy
List policies	iam-grouplistpolicies iam-userlistpolicies iam-rolelistpolicies
Delete a policy	iam-groupdelpolicy iam-userdelpolicy iam-roledelpolicy

Managing Policies (API)

The following table lists which API actions to use for each task you want to perform with a policy. For more information about the actions, go to the [AWS Identity and Access Management API Reference](#), or refer to your SDK's documentation.

For an example of attaching a policy and listing policies, see [Adding a Policy to the Group \(p. 69\)](#).

Task	Action to Use
Attach a JSON policy document you've written	PutGroupPolicy PutUserPolicy PutRolePolicy
List policies	ListGroupPolicies ListUserPolicies ListRolePolicies
Get the contents of a policy	GetGroupPolicy GetUserPolicy GetRolePolicy

Task	Action to Use
Delete a policy	DeleteGroupPolicy DeleteUserPolicy DeleteRolePolicy

Example Policies for IAM Entities

This section shows some basic examples of policies that control access for your IAM resources.

Example 1: Allow requests only if they come from a certain IP address or range

In this example we create a group called *AllUsers* for all the users in the AWS account. We attach a policy to the group that covers all AWS products and resources in the AWS account. This policy assumes the valid IP address ranges for the company are 10.1.2.0/24 and 10.1.3.0/24. The policy denies any traffic to AWS from the AWS account that doesn't come from those IP address ranges.

Note

If you're using Amazon VPC, you might expect all your users' requests to originate from a particular IP address, and so you want to deny requests from any other address.

The policy uses the AWS-wide key called `aws:SourceIp`. For information about the AWS-wide policy keys, see [Available Keys \(p. 51\)](#).

```
{  
    "Statement": [ {  
        "Effect": "Deny",  
        "Action": "*",  
        "Resource": "*",  
        "Condition": {  
            "NotIpAddress": {  
                "aws:SourceIp": [ "10.1.2.0/24", "10.1.3.0/24" ]  
            }  
        }  
    }]  
}
```

Example 2: Allow users to work only with a specific set of AWS products and resources

In this example we create a group called *AllUsers* for all the users in the AWS account. We attach a policy to the group that gives users access to only the following:

- The AWS account's Amazon SimpleDB domain called mySDBDomain (which exists only in the us-east-1 Region)
- The AWS account's corporate Amazon S3 bucket called my_corporate_bucket and all the objects it contains

```
{  
    "Statement": [ {  
        "Effect": "Allow",  
        "Action": [ "sdb:*", "s3:*" ],  
        "Resource": [ "arn:aws:sdb:us-east-1:123456789012:domain/mySDBDomain",  
                    "arn:aws:s3:::my_corporate_bucket",  
                    "arn:aws:s3:::my_corporate_bucket/*" ]  
    },  
    {  
        "Effect": "Deny",  
        "Action": [ "sdb:*", "s3:*" ],  
        "NotResource": [ "arn:aws:sdb:us-east-1:123456789012:domain/mySDBDomain",  
                        "arn:aws:s3:::my_corporate_bucket",  
                        "arn:aws:s3:::my_corporate_bucket/*" ]  
    }  
]  
}
```

The explicit deny statement helps to ensure that the users can't use any other AWS products or resources than those specified in the policy.

Example 3: Allow a user to manage his or her own security credentials

In this example we give Bob permission to access any of the IAM actions related to access keys and signing certificates, but only with his own credentials.

We attach a policy to Bob that covers all the actions that include the literal string `AccessKey` or `SigningCertificate`. The policy specifies Bob as the resource.

```
{  
    "Statement": [ {  
        "Effect": "Allow",  
        "Action": [ "iam:*AccessKey*", "iam:*SigningCertificate*" ],  
        "Resource": "arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/Bob"  
    }  
]  
}
```

Note

The policy uses wildcards to specify all the IAM actions related to access keys and certificates (the `*` matches zero or multiple characters). You could instead list each action explicitly. If you use the wildcards instead of explicitly listing each action, be aware that if we were to add any new IAM actions that include the string `AccessKey` or `SigningCertificate` in their names, this policy would automatically give Bob access to those new actions to use with his own credentials.

If you want to let another user named Jane manage her own credentials, then you would simply need to replace the ARN for Bob with the ARN for Jane in the policy, and attach the policy to Jane. You must explicitly name the user in the resource; there's no way to use a wildcard that automatically refers to "self" (i.e., the user the policy is attached to). In other words, there's no group-level policy you can write that will give all your users credential management permissions only for their own credentials. You need to write an individual policy for each user, and assign the policy to the user.

Example 4: Allow a user to manage a group's membership

In this example we create a policy that allows the user to update the membership of the group called `MarketingGroup`.

```
{  
    "Statement": [ {  
        "Effect": "Allow",  
        "Action": [  
            "iam:AddUserToGroup",  
            "iam:RemoveUserFromGroup",  
            "iam:GetGroup"  
        ],  
        "Resource": "arn:aws:iam::123456789012:group/division_abc/subdivision_xyz/MarketingGroup"  
    }  
]
```

Example 5: Allow a user to list the AWS account's groups and users for reporting purposes

In this example we create a policy that allows the user to call any IAM action that starts with the literal string `Get` or `List`. You probably don't want the user reading the Access Key IDs of other users, so the policy includes a statement that denies the user access to the `ListAccessKeys` action.

```
{  
    "Statement": [ {  
        "Effect": "Allow",  
        "Action": [  
            "iam:Get*",  
            "iam>List*"  
        ],  
        "Resource": "*"  
    },  
    {  
        "Effect": "Deny",  
        "Action": "iam>ListAccessKeys",  
        "Resource": "*"  
    }  
]  
}
```

Note

If we were to add new types of security credentials in the future, the access granted in the policy to all `List*` actions would automatically allow the user to list those new credentials.

Controlling User Access to Your AWS Account Billing Information

The AWS website integrates with AWS Identity and Access Management (IAM) so you can grant users access to billing information. You can control access to the [Account Activity](#) page and the [Usage Reports](#) page. The **Account Activity** page displays invoices, and detailed information about charges and account activity, itemized by service and by usage type. The **Usage Reports** page provides detailed usage reports for each service you are subscribed to.

You can control your users' access to your account billing information in three steps:

1. Configure your security challenge questions.
2. Create custom policies for your IAM users.
3. Activate access to the AWS website.

By default, users do not have access to these pages. However, if you used a wildcard policy to grant users administrator-level access to AWS, then, after you activate access to these pages, those users will have access to these pages unless you use a policy to specifically deny them access. The following topics have examples on using policies to control access to the Account Activity and Usage Reports pages.

For information about deactivating access to the AWS website, see [Deactivate Access to the AWS Website \(p. 187\)](#).

Configure Your Security Challenge Questions

If you haven't already configured your security challenge questions, you need to do this first. Amazon uses these questions to identify you as the owner of your AWS account if you contact our customer service.

To configure your security challenge questions

1. In the **Configure Security Challenge Questions** section on the [Personal Information](#) page, configure three security challenge questions.

Configure Security Challenge Questions

Improve the security of your AWS account by adding security challenge questions. We use these to help identify you as the owner of your AWS account if you ever need to contact our customer service for help.

Question:	What was the name of your first pet? <input type="text"/>
Answer:	<input type="text" value="*****"/>
Question:	What was the first live concert you attended? <input type="text"/>
Answer:	<input type="text" value="*****"/>
Question:	In what city were you living at age 16? <input type="text"/>
Answer:	<input type="text" value="*****"/>

Save Changes

2. Click **Save questions**.

After you have saved your security challenge questions, you can create custom IAM policies for your IAM users to control access to the AWS website.

Create Custom Policies for IAM Users

By default, users that you create in IAM do not have access to billing information such as the [Account Activity](#) page and the [Usage Reports](#) page. Access to your account's billing information is controlled with IAM policies that you can assign to either a single user or a group of users. For instructions, see [Managing IAM Policies](#).

The rest of this section provides simple example policies that you can attach to your user or your group to control access to your account's billing information.

Example 1: Allow users to access the Account Activity page

This policy allows the user to access the **Account Activity** page.

```
{  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "aws-portal:ViewBilling",  
            "Resource": "*"  
        }  
    ]  
}
```

Example 2: Allow users to access the Usage Reports page

This policy allows the user to access the **Usage Reports** page.

```
{  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "aws-portal:ViewUsage",  
            "Resource": "*"  
        }  
    ]  
}
```

Example 3: Deny users access to the Account Activity page

This policy denies the user access to the **Account Activity** page.

```
{  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": "aws-portal:ViewBilling",  
            "Resource": "*"  
        }  
    ]  
}
```

Example 4: Deny users access to the Usage Reports page

This policy denies the user access to the **Usage Reports** page.

```
{  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Action": "aws-portal:ViewUsage",  
            "Resource": "*"  
        }  
    ]  
}
```

Example 5: Allow full access to AWS services but deny users access to billing information

This policy enables full access to all AWS services but denies the user access to the **Account Activity** or **Usage Reports** page.

```
{  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "*",  
            "Resource": "*"  
        },  
        {  
            "Effect": "Deny",  
            "Action": "aws-portal:*",  
            "Resource": "*"  
        }  
    ]  
}
```

After you set custom IAM policies for your IAM users, you can activate access to the AWS website.

Activate Access to the AWS Website

To grant your IAM users access to your account's billing information, you need to activate the functionality.

To activate access to the AWS website

- In the **IAM user access to the AWS website** section on the [Manage Your Account](#) page, click **Activate Now**.

IAM user access to the AWS Website

IAM user access to the AWS Website enables IAM users with appropriate permissions configured to access the Account Activity and Usage Reports pages. When activated, if you want to limit access to these pages for IAM users that currently have full access permissions configured, you must update their policies to restrict their access.

Please see [Using IAM](#) for more details.

Activate the following pages:

- Account Activity Page
 Usage Reports Page

Activate Now

Note

Note that by activating IAM user access to the AWS website, all users for whom you have granted full access to AWS APIs will now also have access to the AWS website. As always, you may restrict their access by applying a more constrained set of permissions. See [Example 5: Allow full access to AWS services but deny users access to billing information \(p. 186\)](#).

Deactivate Access to the AWS Website

To deactivate access to the AWS website

- In the **IAM user access to the AWS Website** section on the [Manage Your Account](#) page, click **Deactivate Now**.

IAM user access to the AWS Website

IAM user access to the AWS Website enables IAM users with appropriate permissions configured to access the Account Activity and Usage Reports pages. When activated, if you want to limit access to these pages for IAM users that currently have full access permissions configured, you must update their policies to restrict their access.

Please see [Using IAM](#) for more details.

Activate the following pages:

Account Activity Page
 Usage Reports Page

Deactivate Now

Controlling User Access to the AWS Management Console

Users who sign in to your AWS account through the sign-in page can access your AWS resources through the AWS Management Console to the extent that you grant them permission. The following table shows the ways you can grant users access to your AWS account resources through the AWS Management Console. It also shows how users can access other AWS account features through the AWS website.

AWS feature	User access
The AWS Management Console	You create a password for each user who needs access to the AWS Management Console. Users access the console via your IAM-enabled AWS account sign-in page. For information about accessing the sign-in page, see The AWS Management Console Sign-in Page (p. 190) . For information about creating passwords, see Managing Passwords (p. 95) .
Your AWS resources, such as Amazon EC2, Amazon S3, and so on	Even if your users have passwords, they still need permission to access your AWS resources. When you create a user, that user has no permissions by default. To give your users the permissions they need, you assign policies to them. If you have many users who will be performing the same tasks with the same resources, you can assign those users to a group, then assign that group permissions under a single policy. For information about creating users and groups, see Working with Users and Groups (p. 72) . For information about using policies to set permissions, see Permissions and Policies (p. 30) .
AWS Discussion Forums	Anyone can read the posts on the AWS Discussion Forums . Users who want to post questions or comments to the AWS Discussion Forum can do so using their user name. The first time a user posts to the AWS Discussion Forum, the user is prompted to enter a nickname and email address for use only by that user in the AWS Discussion Forums.
Your AWS account billing and usage information	You can grant users access your AWS account billing and usage information. For more information, see Controlling User Access to Your AWS Account Billing Information (p. 183) .

AWS feature	User access
Your AWS account profile information	Users cannot access your AWS account profile information.
Your AWS account security credentials	Users cannot access your AWS account security credentials.

Note

IAM policies control access regardless of the interface. For example, you could provide a user with a password to access the AWS Management Console, and the policies for that user (or any groups the user belongs to) would control what the user can do in the AWS Management Console. Or, you could provide the user with AWS access keys for making API calls to AWS, and the policies would control which actions the user could call through a library or client that uses those access keys for authentication.

How Users Sign In to the AWS Management Console

Topics

- [The AWS Management Console Sign-in Page \(p. 190\)](#)
- [Using an Alias for Your AWS Account ID \(p. 192\)](#)
- [MFA Devices and Your IAM-Enabled Sign-in Page \(p. 194\)](#)

The AWS Management Console provides a web-based interface for many AWS product APIs (such as the API for Amazon S3, Amazon EC2, and so on). Users can make requests directly to an API, or users can use the AWS Management Console to make requests to a service's API. Users who work with your AWS resources through the AWS Management Console will use your AWS account's IAM-enabled sign-in page and the passwords you create for them to access the console.

This section provides information about the IAM-enabled AWS Management Console sign-in page and explains how to create an AWS account alias for your sign-in page. For information about creating user passwords, see [Managing Passwords \(p. 95\)](#).

Note

Some AWS products are not supported on the AWS Management Console. For a list of AWS products that are supported on the console, go to [AWS Management Console](#).

The AWS Management Console Sign-in Page

Users who use the AWS Management Console must sign in to your AWS account through the IAM-enabled sign-in page. You provide your users with the URL they need to access the sign-in page. You might consider sending the link through email to the users who need it, or you might create a link to the sign-in page from a page in your company intranet.

Your IAM-enabled AWS Management Console sign-in page will look similar to the following page.

AWS Identity and Access Management Using IAM Using AWS Account Credentials to Sign In to the AWS Management Console

Important

In addition to providing users with a URL to your IAM-enabled sign-in page, for users to sign in to your page, you must provide each user with a password and, if appropriate, an MFA device. For detailed information about passwords and MFA devices, see [Managing Passwords \(p. 95\)](#) and [Using Multi-Factor Authentication \(MFA\) Devices with AWS \(p. 106\)](#).

The IAM-enabled sign-in page URL is created automatically when you begin using IAM. It has the following format.

```
https://your_AWS_Account_ID.signin.aws.amazon.com/console/ec2
```

Note

Your AWS account ID is the same as your account number, but without hyphens. To locate your AWS account number, go to the AWS [Manage Your Account](#) page. Your account number is located near the top right corner of the page.

If you want the URL for your sign-in page to contain your company name (or other friendly identifier) instead of your AWS account ID, you can create an alias for your AWS account ID. For more information about AWS account ID aliases, see [Using an Alias for Your AWS Account ID \(p. 192\)](#).

Note

When a user signs in, the console opens to the console referenced by the abbreviation at the end of your URL. In the preceding sample, signing in opens the Amazon EC2 console. If the URL ended with s3, then the AWS Management Console would open to the Amazon S3 console.

Using AWS Account Credentials to Sign In to the AWS Management Console

When users sign in to your AWS account, they sign in via an IAM-enabled user sign-in page. For their convenience, this sign-in page uses a cookie to remember user status so that the next time a user goes

to the AWS Management Console, the AWS Management Console calls the IAM-enabled user sign-in page by default.

If you want to sign in to the AWS Management Console under your AWS account credentials instead of as an AWS account user, from the user sign-in page, click **Sign in using AWS account credentials**. The Amazon Web Services sign-on page appears, from which you can sign in using your AWS account credentials.

Using an Alias for Your AWS Account ID

If you want the URL for your sign-in page to contain your company name (or other friendly identifier) instead of your AWS account ID, you can create an alias for your AWS account ID. This section provides information about AWS account aliases and lists the API actions you use to create an alias.

Your sign-in page URL has the following format, by default.

```
https://your_AWS_Account_ID.signin.aws.amazon.com/console/ec2
```

If you create an AWS account alias for your AWS account ID, your sign-in page URL will look like the following example.

```
https://youralias.signin.aws.amazon.com/console/ec2
```

Note

The original URL containing your AWS account ID remains active after you create your AWS account alias.

Creating, Deleting, and Listing an AWS Account Alias

You can use the AWS Management Console, the IAM API, or the command line interface to create or delete your AWS account alias.

Important

Your AWS account cannot have more than one alias. If you create a new alias for your AWS account, the new alias overwrites the old alias, and the URL containing the old alias will no longer work.

AWS Management Console

To create an account alias

1. On the **Navigation** pane, select **IAM Dashboard**.
2. Under **AWS Account Alias**, click **Create Account Alias**.



3. Enter the name you want to use for your alias, then click **Yes, Create**.

To remove an account alias

1. On the **Navigation** pane, select **IAM Dashboard**.
2. Under **AWS Account Alias**, click **Remove Account Alias**.



3. Review your change, and then click **Yes, Delete**.

API or CLI

The following table lists the API actions or command line interface (CLI) commands to use to create, delete, or list an AWS account ID sign-in page alias.

Task	Command to Use
Create an alias for your AWS Management Console sign-in page URL	CLI: <code>iam-accountaliascreate</code> API: <code>CreateAccountAlias</code>
Delete an AWS account ID alias	CLI: <code>iam-accountaliasdelete</code> API: <code>DeleteAccountAlias</code>
List your AWS account ID alias	CLI: <code>iam-accountaliaslist</code> API: <code>ListAccountAliases</code>

Note

The alias must be unique across all Amazon Web Services products, and the number of characters it can contain is limited. For more information on limitations on AWS account entities, see [Limitations on IAM Entities \(p. 64\)](#).

For more information on the IAM API actions or CLI commands, go to the [AWS Identity and Access Management API Reference](#) or [AWS Identity and Access Management Command Line Interface Reference](#).

MFA Devices and Your IAM-Enabled Sign-in Page

If a user must use an MFA device to sign in to your IAM-enabled sign-in page, the user is prompted to enter the MFA device authentication code after entering a user name and password. In most cases, the user will be able to use the AWS Management Console after entering the required information.

However, it's possible for an MFA device to get out of synchronization. If after several unsuccessful tries a user cannot sign in to the AWS Management Console, the user will be prompted to synchronize the MFA device. The user can follow the on-screen prompts to synchronize the MFA device. For information about how you can synchronize a device for a user under your AWS account, see [Synchronizing an MFA Device \(p. 117\)](#).

Integrating with Other AWS Products

This section links to topics that describe how IAM integrates with the different AWS products, and how to write policies to control access to a particular AWS product and its resources.

The following table summarizes whether you can grant IAM permissions that control access to a service's actions, resources, or both. For example, you can use IAM to control which Amazon EC2 actions users have access to, but you can't use IAM to control users' access to AMIs, volumes, instances, etc.

AWS Product	Actions	Resources	For more information, see...
Auto Scaling	✓		Auto Scaling and AWS Identity and Access Management in the Auto Scaling Developer Guide
Amazon Web Services Account Billing Information	✓ *		Controlling User Access to Your AWS Account Billing Information *You can use IAM policies to control user access to your account's Account Activity page and Usage Reports page.
AWS CloudFormation	✓		Controlling User Access With AWS Identity and Access Management in the AWS CloudFormation User Guide
Amazon CloudFront	✓		Controlling User Access to Your AWS Account in the Amazon CloudFront Developer Guide
Amazon CloudWatch	✓		Controlling User Access to Your AWS Account in the Amazon CloudWatch Developer Guide

AWS Product	Actions	Resources	For more information, see...
Amazon DynamoDB	✓	✓	Controlling Access to Amazon DynamoDB Resources in the Amazon DynamoDB Developer Guide
AWS Elastic Beanstalk	✓	✓	Using AWS Elastic Beanstalk with AWS Identity and Access Management (IAM) in the AWS Elastic Beanstalk Developer Guide
Amazon Elastic Compute Cloud (Amazon EC2)	✓		Using AWS Identity and Access Management in the Amazon Elastic Compute Cloud User Guide
Elastic Load Balancing	✓		Controlling User Access to Your AWS Account in the Elastic Load Balancing Developer Guide
Amazon Elastic MapReduce	✓		Configuring User Permissions in the Amazon Elastic MapReduce Developer Guide
Amazon ElastiCache	✓		Controlling User Access to Your AWS Account in the Amazon ElastiCache User Guide
Amazon Glacier	✓	✓	Access Control Using AWS Identity and Access Management (IAM) in the Amazon Glacier Developer Guide
AWS Identity and Access Management (IAM)	✓	✓	Example Policies for IAM Entities (p. 179)
AWS Marketplace	✓		Controlling Access to AWS Marketplace Subscriptions
Amazon Relational Database Service (Amazon RDS)	✓		Controlling User Access to Your AWS Account in the Amazon Relational Database Service (RDS) User Guide
Amazon Route 53	✓	✓	Controlling User Access with IAM in the Amazon Route 53 Developer Guide
Amazon Simple Email Service (Amazon SES)	✓		Controlling User Access with IAM in the Amazon Simple Email Service Developer Guide
Amazon Simple Notification Service (Amazon SNS)	✓	✓	Controlling User Access to Your AWS Account in the Amazon Simple Notification Service Getting Started Guide

AWS Product	Actions	Resources	For more information, see...
Amazon Simple Queue Service (Amazon SQS)	✓	✓	Controlling User Access to Your AWS Account in the <i>Amazon Simple Queue Service Developer Guide</i>
Amazon Simple Storage Service (Amazon S3)	✓	✓	Using IAM Policies in the <i>Amazon Simple Storage Service Developer Guide</i>
Amazon Simple Workflow Service (Amazon SWF)	✓ *		Managing Access to Your Amazon SWF Workflows in the <i>Amazon Simple Workflow Service Developer Guide</i> *For Amazon SWF, IAM can only grant access to all Amazon SWF functionality within an account.
Amazon SimpleDB	✓	✓	Managing Users of Amazon SimpleDB in the <i>Amazon SimpleDB Developer Guide</i>
AWS Storage Gateway	✓	✓	Access Control Using AWS Identity and Access Management (IAM) in the <i>AWS Storage Gateway User Guide</i>
AWS Support	✓ *		IAM for AWS Support on the AWS Support site *For AWS Support, IAM can only grant access to all Amazon Support functionality within an account.
Amazon Virtual Private Cloud (Amazon VPC)	✓		Controlling VPC Management in the <i>Amazon Virtual Private Cloud User Guide</i>

Managing Server Certificates

Topics

- [Actions on Server Certificates \(p. 198\)](#)
- [Renaming Server Certificates \(p. 199\)](#)
- [Creating and Uploading Server Certificates \(p. 199\)](#)

This section lists the IAM server certificate actions, describes what you need to know about renaming server certificates, and describes how to create and upload server certificates.

Note

If you want to create user signing certificate (not a server certificate) that you can use to sign SOAP requests in EC2, see [Creating and Uploading a User Signing Certificate \(p. 129\)](#)

Currently, Amazon Elastic Load Balancing is the only service to support the use of server certificates with IAM. For more information about using server certificates with Elastic Load Balancing, see the [Elastic Load Balancing Developer Guide](#).

Actions on Server Certificates

The following table describes actions you can use to manage server certificates in IAM.

Action	API	Command Line Interface
Delete a server certificate	DeleteServerCertificate	iam-servercertdel
Get server certificate information	GetServerCertificate	iam-servercertgetattributes
List server certificates	ListServerCertificates	iam-servercertlistbypath
Update server certificates	UpdateServerCertificate	iam-servercertmod
Upload server certificates	Upload ServerCertificate	iam-servercertupload

For more information about these actions, see the [AWS Identity and Access Management API Reference](#) or the [AWS Identity and Access Management Command Line Interface Reference](#).

Renaming Server Certificates

When you rename a server certificate, the GUID for the server certificate remains the same (for more information about GUIDs, see [GUIDs in Identifiers for IAM Entities \(p. 60\)](#)). However, IAM does not automatically update policies that refer to the server certificate as a resource to use the new name. You must manually do that. For example, Bob is a developer in the company ABC and has a policy attached to him that lets him manage the company's build server certificate, `arn:aws:iam::123456789012:server-certificate/abc/certs/build`. If an admin changes the name of the build server certificate to `build_01` or changes the path for the server certificate, the admin also needs to update the policy attached to Bob to use the new name or path so that Bob can continue to manage that server certificate.

Creating and Uploading Server Certificates

This section describes the process of generating a digital server certificate and preparing it to use with AWS products through IAM. To create a signed certificate, you perform a series of tasks as described by the following topics.

Note

If you want to create user signing certificate (not a server certificate) that you can use to sign SOAP requests in EC2, see [Creating and Uploading a User Signing Certificate \(p. 129\)](#)

Topics

- [Install and Configure OpenSSL \(p. 199\)](#)
- [Create a Private Key \(p. 200\)](#)
- [Create a Certificate Signing Request \(p. 201\)](#)
- [Submit the CSR to a Certificate Authority \(p. 202\)](#)
- [Upload the Signed Certificate \(p. 202\)](#)
- [Verify the Certificate Object \(p. 203\)](#)
- [Sample Certificates \(p. 203\)](#)

Install and Configure OpenSSL

Creating and uploading a certificate requires a tool that supports the SSL and TLS protocols. OpenSSL is an open-source tool that provides the basic cryptographic functions necessary to create an RSA token and sign it with your private key. If you don't already have OpenSSL installed, follow the instructions in this section.

To install OpenSSL on Linux and UNIX

1. Go to [OpenSSL: Source, Tarballs](#) (<http://www.openssl.org/source/>).
2. Download the latest source and build the package.

To install OpenSSL on Windows

1. Go to [OpenSSL: Binary Distributions](#) (<http://www.openssl.org/related/binaries.html>).
2. Click **OpenSSL for Windows**.

A new page displays with links to the Windows downloads.

3. If it is not already installed on your system, select the **Microsoft Visual C++ 2008 Redistributables** link appropriate for your environment and click **Download**. Follow the instructions provided by the **Microsoft Visual C++ 2008 Redistributable Setup Wizard**.
4. After you have installed the Microsoft Visual C++ 2008 Redistributables, select the appropriate version of the OpenSSL binaries for your environment and save the file locally. The **OpenSSL Setup Wizard** launches.
5. Follow the instructions described in the **OpenSSL Setup Wizard**. Save the OpenSSL binaries to a folder in your working directory.

Before you use OpenSSL commands, you must configure the operating system so that it has information about the location of the OpenSSL install point.

To configure OpenSSL on Linux and UNIX

1. At the command line, set the `OpenSSL_HOME` variable to the location of the OpenSSL installation:

```
export OpenSSL_HOME=path_to_your_OpenSSL_installation
```

2. Set the path to the OpenSSL installation:

```
export PATH=$PATH:$OpenSSL_HOME/bin
```

To configure OpenSSL on Windows

1. Open a **Command Prompt** window.
2. Set the `OpenSSL_HOME` variable to the location of the OpenSSL installation:

```
set Path=OpenSSL_HOME\bin;%Path%
```

3. Set the path to the OpenSSL installation:

```
set Path=OpenSSL_HOME\bin;%Path%
```

Note

Any changes you make to Windows environment variables in a **Command Prompt** window are valid only for the current command-line session. You can make persistent changes to the environment variables by setting them as system properties. The exact procedures depends on what version of Windows you're using. (For example, in Windows 7, open **Control Panel > System and Security > System**. Then choose **Advanced system settings > Advanced tab > Environment Variables**.) For more information, see the Windows documentation.

Create a Private Key

You need a unique private key to create your Certificate Signing Request (CSR).

To create a private key

- At the command line, use the `openssl genrsa` command and the following syntax:

```
openssl genrsa 1024 > private-key.pem
```

For *private-key.pem*, specify your own file name. In the example, 1024 represents 1024-bit encryption. AWS also supports 2048-bit and 4096-bit encryption. We recommend you create an RSA key that is 1024-bit or 2048-bit in length.

Create a Certificate Signing Request

The next step is to create a Certificate Signing Request (CSR). This is a file that you can send to a Certificate Authority (CA) to apply for a digital server certificate.

To create a CSR

- Use the `openssl req` command to create a CSR and the following syntax:

```
openssl req -new -key private-key.pem -out csr.pem
```

The output will look similar to the following example:

```
You are about to be asked to enter information that will be incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank.  
For some fields there will be a default value.  
If you enter '.', the field will be left blank.
```

The following table can help you create your certificate request.

Name	Description	Example
Country Name	The two-letter ISO abbreviation for your country.	US = United States
State or Province	The name of the state or province where your organization is located. This name cannot be abbreviated.	Washington
Locality Name	The name of the city where your organization is located.	Seattle
Organization Name	The full legal name of your organization. Do not abbreviate your organization name.	Example Corp.
Organizational Unit	Optional, for additional organization information.	Marketing
Common Name	The fully qualified domain name for your CNAME. You will receive a certificate name check warning if this is not an exact match.	www.yourdomain.com
Email address	The server administrator's email address	someone@yourdomain.com

Note

The Common Name field is often misunderstood and is completed incorrectly. The common name is typically your host plus domain name. It will look like "www.company.com" or "company.com". You need to create a CSR using your correct common name.

Submit the CSR to a Certificate Authority

Your CSR contains information identifying you. To apply for a digital server certificate, send your CSR to a Certificate Authority (CA). The CA might require other credentials or proofs of identity.

If the request for a certificate is successful, the CA returns an identity certificate (and possibly a chain certificate) that is digitally signed.

AWS does not recommend any one CA. For a partial listing of available CAs, see [Third-Party Certificate Authorities](#).

Upload the Signed Certificate

When you receive your digital server certificate from the certificate authority (CA), you can upload the certificate along with the private certificate and, optionally, a certificate chain to IAM. After you upload the certificates to IAM, the certificates are available for other AWS services to use.

To upload certificates on IAM, you use the IAM command line interface (IAM CLI). For more information about installing the IAM command line toolkit, refer to [Getting the Command Line Tools](#) in the *AWS IAM Command Line Reference*.

Note

A certificate authority, might return a signed digital certificate in a format that is not supported by IAM. You can convert the certificate to the correct format (X.509 PEM) by using OpenSSL.

The specific command depends on the current format of your certificate.

To upload a signed certificate

Your digital server certificate can include a chain. A chain contains a list of certificates that is used to build a trust path to a trusted Certificate Authority. If your signed certificate does not require a chain, omit the `-c` parameter.

- Use the `iam-servercertupload` command to upload a signed certificate:
 - On Linux and UNIX computers, enter the following command:

```
& ./iam-servercertupload -b public_key_certificate_file -c certificate_chain_file -k privatekey.pem -s certificate_object_name
```

- On Windows computers, enter the following command:

```
c:\ iam-servercertupload -b public_key_certificate_file -c certificate_chain_file -k privatekey.pem -s certificate_object_name
```

When you upload your certificates, IAM validates the certificates with the following criteria:

- Certificates must follow the X.509 PEM format.
- The current date must be between the certificate's start and end date.

- Public and private certificate files must contain only a single certificate.
- The private key must match the public key that is in the digital server certificate.
- The private key must be an RSA private key in PEM format, where the PEM header is BEGIN RSA PRIVATE KEY and the footer is END RSA PRIVATE KEY (as shown in [Sample Certificates \(p. 203\)](#)).
- The private key cannot be encrypted with a password.
- A certificate chain starts with the immediate signing certificate and is then followed by any intermediaries in order. Intermediaries that are not involved in the trust path must not be included. The trusted root certificate can be optionally included as the last certificate.

If you have certificates that result in an error when you upload them, ensure that they meet the criteria, and then try uploading them again.

To see sample certificates that are valid with IAM, see [Sample Certificates \(p. 203\)](#).

Verify the Certificate Object

After the digital server certificate is uploaded, you can verify that the information is stored in IAM. Each certificate object has a unique Amazon Resource Name (ARN) and GUID. You can request these details for a specific certificate object by referencing the name of the certificate object.

To view the certificate object's ARN and GUID

- Use the `iam-servercertgetattributes` command to verify the certificate object:
 - On Linux and UNIX computers, enter the following command:

```
& ./iam-servercertgetattributes -s certificate_object_name
```

- On Windows computers, enter the following command:

```
c:\ iam-servercertgetattributes -s certificate_object_name
```

The output will look similar to the following example.

```
arn:aws:iam::Your_AWS_Account_ID:server-certificate/Your_Certificate_Object_Name_Certificate_Object_GUID
```

You have now completed the process for creating and uploading a signed certificate. For information about setting up a load balancer using Amazon ELB's HTTPS support, see the command line interface (CLI) examples in the [How to Set Up a LoadBalancer with HTTPS Support](#) section of the *Elastic Load Balancing Developer Guide*.

Sample Certificates

The following certificates show the valid format that IAM accepts for digital server certificates and their associated private key and certificate chain.

The digital server certificate associates your public key with your identity. When you submit your Certificate Signing Request (CSR) to a Certificate Authority (CA), a digital server certificate is returned to you by the CA. The following figure is a sample digital server certificate:

Sample digital server certificate

```
-----BEGIN CERTIFICATE-----
MIIE+TCCA+GgAwIBAgIQU306HIX4KsiotW1s2A2krTANBgkqhkiG9w0BAQUFADCB
tTELMAkGA1UEBhMCVVMxFzAVBqNVBAoTD1Z1cm1TaWduLCBjbMmMR8wHQYDVQQL
ExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMTswOQYDVQQLEzJUXJtcyBvZiB1c2Ug
YXQgaHR0cHM6Ly93d3cudmVyaXNpZ24uY29tL3JwYSAoYykwoTEvMC0GA1UEAxMm
VmVyaVNpZ24gQ2xhc3MgMyBTZWN1cmUgU2VydmyIENbIC0gRzIwHhcNMTAxMDA4
MDAwMDAwWhcNMTMxMDA3MjM1OTU5WjBqMQswCQYDVQQGEwJVUzETMBEGA1UECBMK
V2FzaGluZ3RvbjEQMA4GA1UEBxQHU2VhdHRsZTEYMBYGA1UEChQPQW1hem9uLmNv
bSBJbmMuMRowGAYDVQQDFBFpYW0uY1hem9uYXdzLmNvbTCBnzANBgkqhkiG9w0B
AQEFAAOBJQAwgYkCgYE3Xb0EGea2dB8QGEUwLcEpwvGawEkUdLZmGL1rQJZdeeN
3vaF+ZTm8Qw5Adk2Gr/RwYXtpx04xvQXmNm+9YmksHmCZdrucrW1eN/P9wBfqMMZ
X964CjVov3NrF5Aux8jgtw0yu//C3hWnOuIVGdg76626ggOoJSaj48R2n0MnVcC
AwEAAoOCAdEwgHNMAkGA1UDewQCMAAwCwYDVR0PBAQDAgWgMEUGA1UDHwQ+MDww
OqA4oDaGNh0dHA6Ly9TV1JTZWN1cmUtRzItY3JsLnZ1cm1zaWduLmNvbS9TV1JT
ZWN1cmVHMi5jcmwwRAYDVR0gBD0wOzA5BgtghkgBhvfaQcXAzAqMCgGCCsGAQUF
BwIBFhxodHRwczovL3d3dy52ZXJpc2lnbi5jb20vcnBhMB0GA1UDJQQWMBQGCCsG
AQFBwMBBgggBgEFBQcDAjAfBgNVHSMEGDAwgs17wsRzsBBA6NKZZBiShzgVy19
RzB2BgggBgEFBQcBAQRqMGgwJAYIKwYBBQUHMACGGGh0dHA6Ly9vY3NwLnZ1cm1z
aWduLmNvbTBABgggBgEFBQcwAoY0aHR0cDovL1NWU1N1Y3VyzS1HMi1haWEudmVy
aXNpZ24uY29tL1NWU1N1Y3VyzUcyLmN1cjBuBgggBgEFBQcBDARiMGChXqBcMFow
WDBWFglpbWFnZS9naWYwITAfMacGBSSoAwIaBRLa7kolgyMu9BSOJsprEsHiyEF
GDAmFiRodHRwOi8vbG9nby52ZXJpc2lnbi5jb20vdnNsbdvMS5naWYwdQYJKoZI
hvcNAQEFBQADggEBALpFBXeG782QsTtGwEE9zBcVCuKjrs13dWK1dFiq3OP4y/Bi
ZBYEywBt8zNuYFUE25Ub/zmvmppe7p0G76tmQ8bRp/4qkJoiSesHJvFgJ1mksr3IQ
3gaE1aN2BSUIhXGLn9N4F09hYwwbeEZaCxfgBiLDEIodNwzcvGJ+2L1DWGJOGrNI
NM856xjqhJCPxYzk9buuCl1B4Kzu0CTbexz/iEgYY+DiuTxcfA4uhwMDSe0nybn
1qiwrk450mCOnqH4ly4P41Xo02t4A/DI1I8ZNct/Qf169a2Lf6vc9rf7BELT0e5Y
R7CKx7fc5xRaeQdyGj/dJevm9BF/mSdncls5vas=
-----END CERTIFICATE-----
```

The private key allows you to decrypt messages that are encrypted with your public key. The following figure is a sample private key:

Sample private key

```
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQDCvraqjUowfPitkkryxATvssBmq4p/voCwkU2ihjSqe3jw6g9xu
VrJ/B2SAtaY2KEfKCiW9hpOvmnhPBsZKfucP581Pad4CQyU1wX+rumsydcpebvKti
UMg4RhoPiOmR8yqnaUJiVA1Lo4WoyPGaTfdEzc6xRLZCF3RKm+gZXGiOwIDAQAB
AoGAfiZ41hVB6XcnOsYg7w4imEbWyah/A6cc58INyYvxqulDi6emucURO8tnmaM
3aYoIsuB+Qx1Hjq0dnqn9lfQKo8tJcU4VTwTI5acB/AWLuuSVbrp1fGQYPTqw/Ag
07Lfg9V4pWC6CrQUDNtPQLBM6gPVo4FMDk/STXbujjVJkCQQDhE5Eo1I8B7GsG
J1sFzHiUscxb9EEKi5+BeFjxdfl78TKDILRX8doj3S4wtB+5/oUpQBD2mSH+mQU6
E1i3pVhfAkEA3YBXDtssGxc0q/6QhDP PJ7KOvqBRj9CziMuzz/S3GEu3xJUqNLFF
vlie8meapgnOrpmW9ErJNaC/c3mIgfNzpQJBAM2Q4XL+u94130mviCKzrS2hddRG
MWFARF4rXJCr/0CD+m5o4E2yRlmbGSTCXnexTk1uhfU3NyUg/PUD111kWmECQQCg
Hn1QvN4uxSynNGM1ngoeiT3u4TF0gTp0lcRLDDyajImwSp/y7VGok2h85JXvduF4
Z8Cuoa0n3ibng7BB0EqdAkEAvuq6aCH+yYiCYhzWe6IDYCUnDa8PP/ExURzUAyYA
fGLgzft0+yAIkqfgvkqLBvVGtvYItJuzU4LiUiGriX6NNAA==
-----END RSA PRIVATE KEY-----
```

A certificate chain is an ordered list of certificates that starts with the immediate signing certificate and is then followed by any intermediaries in order. Intermediaries that are not involved in the trust path must

not be included. The trusted root certificate can be optionally included as the last certificate. The following figure is a sample certificate chain:

Sample certificate chain

```
-----BEGIN CERTIFICATE-----
MIIGLDCCBZWgAwIBAgIQbk/6s8XmacTRZ8mSq+hYxDANBgkqhkiG9w0BAQUFADCB
wTELMAkGA1UEBhMCVVMxFzAVBqNVBAoTDlZlcmlTaWduLCBjbMmuMTwwOgYDVQQL
EzNDbGFzcyAzIFB1YmxpYyBQcm1tYXJ5IENlcnRpZmljYXRpb24gQXV0aG9yaXR5
IC0gRzIxOjA4BgNVBAsTMShjKSAXOTk4IFZlcmlTaWduLCBjbMmuIC0gRm9yIGF1
dGhvcml6ZWQgdXN1IG9ubHkxHzAdBqNVBAsTF1ZlcmlTaWduIFRydXN0IE5ldHdv
cmswHhcNMDkwMzI1MDAwMDAwWhcNMTkwmZl0Mjm1OTU5wjcBtTELMAkGA1UEBhMC
VVMxFzAVBqNVBAoTDlZlcmlTaWduLCBjbMmuMR8wHQYDVQQLExZWZXJpu2lnbiBU
cnVzdCBOZXr3b3JrMTswOQYDVQQLExZJUZXJtcyBvZiB1c2UgYXQgaHR0cHM6Ly93
d3cudmVyaXnpZ24uY29tL3JwYSAoYykwoTEvMC0GA1UEAxMmVmVyaVNpZ24gQ2xh
c3MgMyBTZWN1cmUgU2VydMvYIENBIC0gRzIwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQDUvo9XOzcokBj0pXBXTatR1qltZxVy/iwDSMoJWzjOE3JPMu
7UNFBY6J1/raSrX4Po1Ox/lJUEU3QJ90qqBRVWHxYISJpZ6AjS+wIapFgsTPtBR/
RxUgKIKwablAr1wH1/ZzzMtiv1xNSf8miKtUJTovStoOmOKJcrn892g8xB85essX
gfMMrQ/cYWibEAsEHikYcV5iy0PevjG6cQIZTiapUdqMZGkD3pz9ff17Ybz8hHyI
XLTDe+1fK0YS8f0AAZqLW+mjBS6PL1ve8xt4+GaRCMBetzWwNsruqHugffkwer/4
3R1RKyC6/qfPoU6wZ/WAqiuDLtKOViM0HikLAGMBAAGjggKpMIICpTA0BggrBgfEF
BQcBAQQoMCYwJAYIKwYBBQUHMAggGh0dHA6Ly9vY3NwLnZlcmlzaWduLmNvbTAS
BgNVHRMBAf8ECDAGAQH/AgEAMHAGA1UdIARpMGcwQYLYIZIAyB4RQEHFwMwVjaC
BggrBgfFBQcCARYcaHR0cHM6Ly93d3cudmVyaXnpZ24uY29tL2NwczaQBggrBgfEF
BQcCAjAeGhxodHRwczovL3d3dy52ZXJpc2lnbi5jb20vcnBhMDQGA1UdHwQtMCsw
KaAnoCWGI2h0dHA6Ly9jcmwudmVyaXnpZ24uY29tL3bjYT MtZzIuY3JsmA4GA1Ud
DwEB/wQEAWIBBjBtBggrBgfEFBQcBDARhMF+hXaBbMFkwVzBFg1pbWFnzs9naWYw
ITAfMACGBSSoAwIaBBS5dMahqyNjmvDz4Bq1EgYLhsZLjAlFiNodHRwOi8vbG9n
by52ZXJpc2lnbi5jb20vdnNsb2dvLmdpZjApBqNVHREEIjAgpB4wHDEaMBgGA1UE
AxMRQ2xhc3MzQ0EyMDQ4LTEtNTIwHQYDVROOBByEFKXvCxH0wEEDo0plkEiyHOBX
LX1HMIhNBgNVHSMEgd8wgdyhgcekgcQwgcExCzAJBgNVBAYTA1VTMRcwFQYDVQK
Ew5WZXJpU2lnbiwgSW5jLiE8MDoGA1UECxMzQ2xhc3MgMyBQdWJsaWMgUHJpbWFy
eSBDZXJ0awZpY2F0aw9uIEF1dGhvcml0eSATIEcyMTowOAYDVQQLExEoYykgMTk5
OCBWZXJpU2lnbiwgSW5jLiAtIEZvcibhdXRob3JpemVkiHVzZSBvbmx5MR8wHQYD
VQQLExZWZXJpU2lnbiUBcnVzdCBOZXr3b3JrghB92f4Hz6getxB5Z/unitTTGMA0G
CSqGSIB3DQEBBQUAA4GBAGN0Lz1Tqi+X7CYRZhr+8d5BjxnSf9jBHPniOFY6H5Cu
OcUgdav4bc1nHynCiDcUiGNLjsnY5H48KMBJLb7j+M9AgtvVP7UzNvWhb981R5e
YhHB2QmcQrmy1KotmDojYMyimvFu6M+00Ro8XhnF15s1sAIjJOUFuNWI4+D6ufRf
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIDAjCCAmCEH3Z/gfPqB63EHln+6eJNMYwDQYJKoZIhvcNAQEFBQAwgcExCzAJ
BgNVBAYTA1VTMRcwFQYDVQKew5WZXJpU2lnbiwgSW5jLjE8MDoGA1UECxMzQ2xh
c3MgMyBQdWJsaWMgUHJpbWFyeSBDZXJ0awZpY2F0aW9uIEF1dGhvcml0eSATIEcy
MTowOAYDVQQLExEoYykgMTk5OCBWZXJpU2lnbiwgSW5jLiAtIEZvcibhdXRob3Jp
emVkiHVzZSBvbmx5MR8wHQYDVQQLExZWZXJpU2lnbiUBcnVzdCBOZXr3b3JrMB4X
DTk4MDUxODAwMDAwMFoXDTI4MDgwMTIzNTk1OVowgcExCzAJBgNVBAYTA1VTMRcw
FQYDVQKew5WZXJpU2lnbiwgSW5jLiE8MDoGA1UECxMzQ2xhc3MgMyBQdWJsaWMg
UHJpbWFyeSBDZXJ0awZpY2F0aW9uIEF1dGhvcml0eSATIEcyMTowOAYDVQQLExEo
YykgMTk5OCBWZXJpU2lnbiwgSW5jLiAtIEZvcibhdXRob3JpemVkiHVzZSBvbmx5
MR8wHQYDVQQLExZWZXJpU2lnbiUBcnVzdCBOZXr3b3JrMIGfMA0GCSqGSIb3DQE
BAAQAA4GNADCBiQKBgQDMxtERXVxp0KvTuWpMmR9ZmDCOfOugRm1HP9SFIIThbbP4
p00M8RcPO/mn+SXXwc+EY/J8Y8+iR/LGwzOOZEAEaMGauWQcRXfh2G711Sk8U0g0
13gfqLptQ5GVj0VXXn7F+8qkB0vqlzdUMG+7AUcyM83cV5tkaWH4mx0ciU9cZwID
AQABMA0GCSqGSIB3DQEBBQUAA4GBAFFNzb5cy5gZnBWyAT14Lk0PZ3BwmcYQWpSk
U01UbSuvDV1Ai2TT1+7eVmGSX6bEHRBhNtMsJzzoKQm5EWR0zLVznxxIqbkhAe7i
F6YM40AIow7n60RzKprxaZLvcRTDOaxxp5EJb+RxBrO6WVcmeQD2+A2iMzAo1KpY
oJ2daZH9
-----END CERTIFICATE-----
```

Making Query Requests

Topics

- [Endpoints \(p. 207\)](#)
- [HTTPS Required \(p. 207\)](#)
- [Signing AWS API Requests \(p. 208\)](#)

This section contains general information about using the Query API. For details about the API actions and errors for the IAM API or the AWS Security Token Service API, go to the [AWS Identity and Access Management API Reference](#) or the [AWS Security Token Service API Reference](#).

IAM and AWS Security Token Service support Query requests for calling service actions. Query requests are simple HTTPS requests, using the GET or POST method. Query requests must contain an *Action* parameter to indicate the action to be performed.

The response is an XML document that conforms to a schema. The schema is included in the WSDL. The IAM WSDL is located at <https://iam.amazonaws.com/doc/2010-05-08/AWSIdentityManagement.wsdl>. The AWS Security Token Service WSDL is located at:
<https://sts.amazonaws.com/doc/2011-06-15/AWSSecurityTokenService.wsdl>.

Endpoints

IAM and AWS Security Token Service each have a single global endpoint. The IAM endpoint is <https://iam.amazonaws.com>. The AWS Security Token Service endpoint is <https://sts.amazonaws.com>.

For more information about AWS product endpoints and regions go to [Regions and Endpoints](#) in the [Amazon Web Services General Reference](#).

HTTPS Required

Because the Query API returns sensitive information such as security credentials, you must use HTTPS with all API requests.

Signing AWS API Requests

To sign your API requests, we recommend using AWS Signature Version 4. For information about using Signature Version 4, go to [Signature Version 4 Signing Process](#) in the *AWS General Reference*.

If you need to use Signature Version 2, information about using Signature Version 2 is also available in the [AWS General Reference](#).

Document History

This Document History is associated with the 2010-05-08 release of AWS Identity and Access Management. This guide was last updated on 19 November 2012.

The following table describes important changes since the last release of *Using AWS Identity and Access Management*.

Change	Description	Release Date
Cross-account API access	This release adds support for cross-account API access with IAM roles. With IAM roles, you can delegate access to resources in your AWS account so that IAM users from another AWS account can access your resources. For details, see Enabling Cross-Account API Access (p. 142) .	This release
MFA-Protected API Access	This release introduces MFA-protected API access, a feature that enables you to add an extra layer of security over AWS APIs using AWS Multi-Factor Authentication (MFA), see Configuring MFA-Protected API Access (p. 27) .	July 8, 2012
Business Use Cases	This section has been rewritten and updated. For more information, see Business Use Cases (p. 27)	June 22, 2012
IAM Roles for Amazon EC2 Instances	This release introduces IAM roles for Amazon EC2 instances. Use roles to enable applications running on your Amazon EC2 instances to securely access your AWS resources. For more information about IAM roles for EC2 instances, see Delegating API Access by Using Roles (p. 138) .	June 07, 2012
AWS Storage Gateway	This release introduces AWS Storage Gateway integration with IAM. For more information about using IAM with AWS Storage Gateway, go to Access Control Using AWS Identity and Access Management (IAM) in the AWS Storage Gateway User Guide . For a general description of AWS Storage Gateway, go to AWS Storage Gateway .	May 14, 2012
Updated Documentation	The IAM Getting Started Guide was merged into Using IAM, and Using IAM was reorganized to enhance usability. The Getting Started is now available at Getting Started (p. 4) .	May 02, 2012

Change	Description	Release Date
Signature Version 4	With this release of IAM, you can use Signature Version 4 to sign your IAM API requests. For more information about Signature Version 4, go to Signature Version 4 Signing Process in the <i>AWS General Reference</i> .	March 15, 2012
User Password Management	With this release of IAM, you can enable your IAM users to change their password. For more information, see Managing Passwords (p. 95) .	March 08, 2012
Account Password Policy	IAM now includes an account-wide password policy you can use to ensure your IAM users create strong passwords. For more information, see Managing an IAM Password Policy (p. 96) .	March 08, 2012
IAM User Access to Your AWS Account Billing Information	With this release of IAM, you can enable your IAM users to access your AWS account billing and usage information. For more information, see Controlling User Access to Your AWS Account Billing Information (p. 183) .	March 08, 2012
Amazon Simple Workflow Service (SWF)	This release introduces Amazon Simple Workflow Service (SWF) integration with IAM. For more information about using IAM with Amazon Simple Workflow Service, go to Managing Access to Your Amazon SWF Workflows in the <i>Amazon Simple Workflow Service Developer Guide</i> . For a general description of Amazon Simple Workflow Service, go to Amazon Simple Workflow Service .	February 22, 2012
Single Sign-on Access to the AWS Management Console for Federated Users	With this release, you can give your federated users single sign-on access to the AWS Management Console through your identity and authorization system, without requiring users to sign in to Amazon Web Services (AWS). For more information, go to Giving Federated Users Direct Access to the AWS Management Console in <i>Using Temporary Security Credentials</i> .	January 19, 2012
New Documentation: Using Temporary Security Credentials	The documentation that describes creating temporary security credentials for federated users and mobile applications has been moved to a new, expanded stand-alone guide named Using Temporary Security Credentials .	January 19, 2012
Amazon DynamoDB	This release introduces Amazon DynamoDB integration with IAM. For more information about using IAM with Amazon DynamoDB, go to Controlling Access to Amazon DynamoDB Resources in the <i>Amazon DynamoDB Developer Guide</i> . For a general description of Amazon DynamoDB, go to Amazon DynamoDB .	January 18, 2012
AWS Elastic Beanstalk	This release introduces AWS Elastic Beanstalk integration with IAM. For more information about using IAM with AWS Elastic Beanstalk, go to Using AWS Elastic Beanstalk with AWS Identity and Access Management (IAM) in the <i>AWS Elastic Beanstalk Developer Guide</i> . For a general description of AWS Elastic Beanstalk, go to AWS Elastic Beanstalk . For IAM use cases, see Business Use Cases (p. 27) .	November 21, 2011

Change	Description	Release Date
AWS Virtual MFA	With this release, you can use IAM to configure and enable a virtual MFA device. A virtual MFA device uses a software application that can generate six-digit authentication codes that are compatible with the Time-Based One-Time Password (TOTP) standard, as described in RFC 6238 . The software application can run on any mobile hardware device, including a smartphone. For more information about virtual MFA and about using IAM to configure and enable a virtual MFA device, see Using a Virtual MFA Device with AWS (p. 108) .	November 02, 2011
Policy Generator Integration with the AWS Identity and Access Management Console	<p>This release introduces the integration of the policy generator with the AWS Identity and Access Management (IAM) console. Integrating the policy generator with the IAM console makes it even easier to set permissions for your IAM users and groups. To use the policy generator in the console, select Policy Generator in the user or group permissions dialogs.</p> <p>For more information about the AWS access policy language, see Key Concepts in <i>Using AWS Identity and Access Management</i>. If you want to use the policy generator online to create policies for AWS products without accessing the console, go to the AWS Policy Generator.</p>	October 06, 2011
Amazon ElastiCache	This release introduces Amazon ElastiCache integration with IAM. For more information about using IAM with ElastiCache, go to Controlling User Access to Your AWS Account in the <i>Amazon ElastiCache User Guide</i> . For a general description of Amazon ElastiCache, go to Amazon ElastiCache . For IAM use cases, see Business Use Cases (p. 27) .	August 22, 2011
Temporary Security Credentials	This release of IAM introduces temporary security credentials that you can use to grant temporary access to non-AWS users (federated users), to IAM users who need temporary access to your AWS resources, and to your mobile and browser-based applications that need to access your AWS resources securely. For more information, go to Using Temporary Security Credentials .	August 03, 2011
Cross-Account Access for IAM Users	This release of IAM introduces cross-account access for IAM users. For more information, see Delegating API Access by Using Roles (p. 138) .	June 06, 2011
The AWS Management Console IAM Tab	This release of IAM introduces AWS Management Console support. The IAM tab of the console is a graphical user interface (GUI) that enables you to do almost everything you can do with the IAM APIs. For more information, see Ways to Access IAM (p. 59) and the AWS Identity and Access Management Getting Started Guide .	May 03, 2011
Amazon CloudFront	This release of IAM includes integration with Amazon CloudFront. For more information, go to Controlling User Access to Your AWS Account in the CloudFront Developer Guide.	March 10, 2011
AWS CloudFormation	This release introduces AWS CloudFormation integration with IAM. For more information, go to Controlling User Access With AWS Identity and Access Management in the <i>AWS CloudFormation User Guide</i> .	24 February 2011

Change	Description	Release Date
Amazon Elastic MapReduce	This release introduces Amazon Elastic MapReduce integration with IAM. For more information, go to Amazon Elastic MapReduce in Business Use Cases in the <i>Using AWS Identity and Access Management</i> guide.	22 February 2011
IAM-Enabled User Access to the AWS Management Console and AWS Developer Forums	IAM now provides an IAM-enabled sign-in page for the AWS Management Console. You provide your users with a login profile and with appropriate permissions so they can access your available AWS resources through the AWS Management Console. For information about accessing the AWS Management Console through IAM, see How Users Sign In to the AWS Management Console (p. 190) . For information about the AWS Management Console, see AWS Management Console .	14 February 2011
Amazon Simple Email Service	This release introduces Amazon Simple Email Service (Amazon SES) integration with IAM. For more information, see Controlling User Access with IAM .	24 January 2011
AWS IAM Integration with Amazon Route 53	Amazon Route 53 DNS service is now integrated with IAM. For information about using Amazon Route 53 with IAM, see Integrating with Other AWS Products (p. 195) . For more information about Amazon Route 53, go to Amazon Route 53 on the AWS website.	05 December 2010
AWS IAM Integration with Amazon CloudWatch	Amazon CloudWatch is now integrated with IAM. For information about using Amazon CloudWatch with IAM, see Controlling User Access to Your AWS Account . For more information about Amazon CloudWatch, see Amazon CloudWatch on the AWS website.	29 November 2010
Server Certificate Support	IAM now provides server certificate APIs for use with Elastic Load Balancing server certificates. For information about using IAM to manage server certificates, see Managing Server Certificates (p. 198) .	14 October 2010
Initial Release	This is the first release of <i>Using AWS Identity and Access Management</i> .	02 September 2010