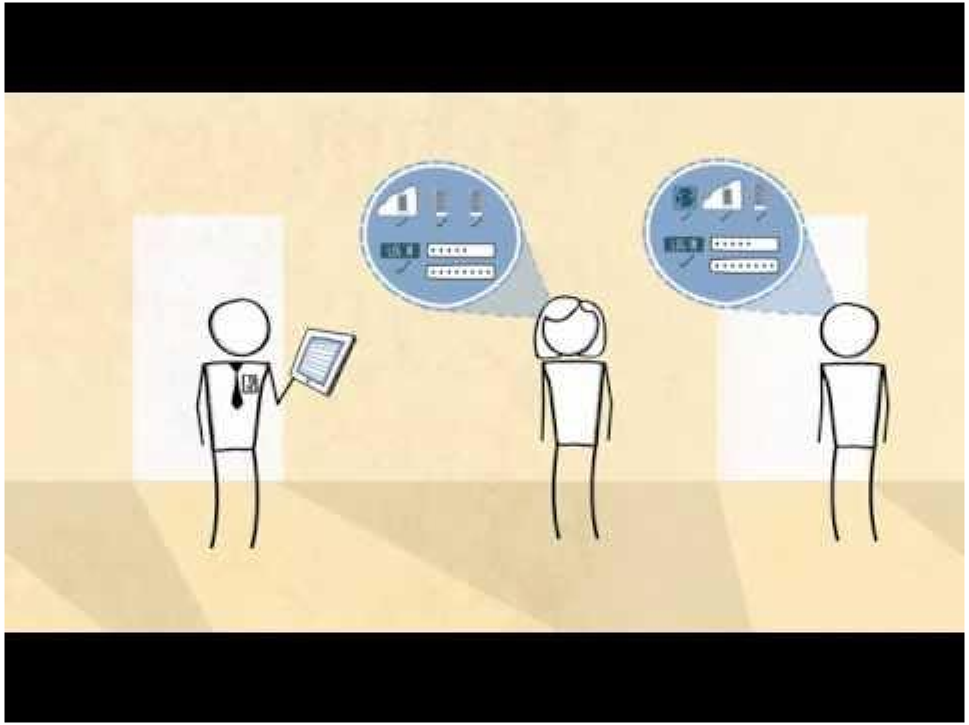
AWS CERTIFIED SOLUTIONS ARCHITECT - ASSOCIATE

Amazon IAM

What is IAM?

AWS Identity and Access Management (IAM) is a web service that **helps you securely control access to AWS resources for your users**. You use IAM to control **who can use** your AWS resources (***authentication***) and **what resources they can use** and in what ways (***authorization***).



IAM Features

- **Shared access to your AWS account**
  - You can grant other people permission to administer and use resources in your AWS account without having to share your password or access key.
- **Granular permissions**
  - You can grant different permissions to different people for different resources. For example, you might allow some users complete access to Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB, Amazon Redshift, and other AWS services. For other users, you can allow read-only access to just some S3 buckets, or permission to administer just some EC2 instances, or to access your billing information but nothing else.
- **Secure access to AWS resources for applications that run on Amazon EC2**
  - You can use IAM features to securely give applications that run on EC2 instances the credentials that they need in order to access other AWS resources, like S3 buckets and RDS or DynamoDB databases.
- **Multi-factor authentication (MFA)**
  - You can add two-factor authentication to your account and to individual users for extra security. With MFA you or your users must provide not only a password or access key to work with your account, but also a code from a specially configured device.

IAM Features

- **Identity federation**
  - You can allow users who already have passwords elsewhere—for example, in your corporate network or with an Internet identity provider—to get temporary access to your AWS account.
- **Identity information for assurance**
  - If you use AWS CloudTrail, you receive log records that include information about those who made requests for resources in your account. That information is based on IAM identities.
- **PCI DSS Compliance**
  - IAM supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS).
- **Integrated with many AWS services**

IAM Features

- **Eventually Consistent**
  - IAM, like many other AWS services, is eventually consistent. IAM achieves high availability by replicating data across multiple servers within Amazon's data centers around the world. If a request to change some data is successful, the change is committed and safely stored. However, the change must be replicated across IAM, which can take some time. Such changes include creating or updating users, groups, roles, or policies. We recommend that you do not include such IAM changes in the critical, high-availability code paths of your application. Instead, make IAM changes in a separate initialization or setup routine that you run less frequently. Also, be sure to verify that the changes have been propagated before production workflows depend on them.
- **Free to use**
  - AWS Identity and Access Management is a feature of your AWS account offered at no additional charge. You will be charged only for use of other AWS products by your IAM users.
  - AWS Security Token Service is an included feature of your AWS account offered at no additional charge. You are charged only for the use of other AWS services that are accessed by your AWS STS temporary security credentials.

Accessing IAM

- AWS Management Console
- AWS Command Line Tools
- AWS SDKs
- IAM HTTPS API
    - Access IAM and AWS programmatically by using the IAM HTTPS API, which lets you issue HTTPS requests directly to the service

Overview of Identity Management: Users

- First-Time Access Only: Your Root User Credentials
    - complete, unrestricted access to all resources in your AWS account, including access to your billing information and the ability to change your password. don't use root user credentials for everyday access.
- IAM Users
- Federating Existing Users

Principals

The first IAM concept to understand is principals. A principal is an IAM entity that is allowed to interact with AWS resources. A principal can be permanent or temporary, and it can represent a human or an application. There are three types of principals:

- Root users
- IAM users
- Roles/temporary security tokens.

Root User

When you first create an AWS account, you begin with only a single sign-in principal that has complete access to all AWS Cloud services and resources in the account. This principal is called the root user. As long as you have an open account with AWS, the root user for that relationship will persist. The root user can be used for both console and programmatic access to AWS resources. The root user is similar in concept to the UNIX root or Windows Administrator account—it has full privileges to do anything in the account, including closing the account.

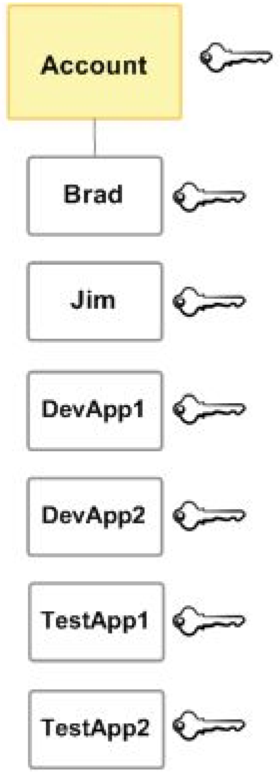
Do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user and then securely locking away the root user credentials.

IAM Users

- The "identity" aspect of AWS Identity and Access Management (IAM) helps you with the question "Who is that user?".
- Create individual IAM users within your account that correspond to users in your organization.
- IAM users are not separate accounts; they are users within your account. Each user can have its own password for access to the AWS Management Console.
- You can also create an individual access key for each user so that the user can make programmatic requests to work with resources in your account.

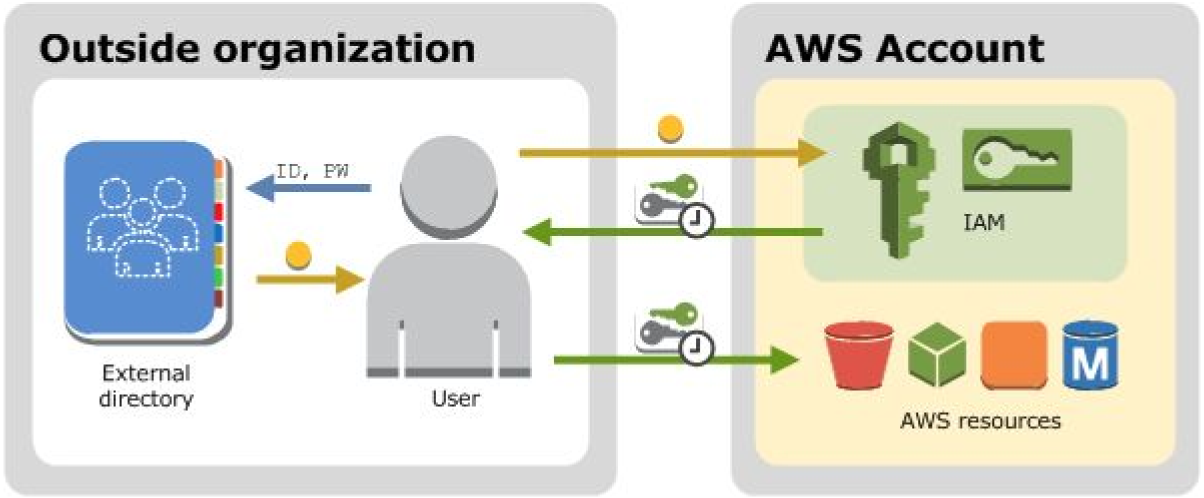
IAM Users

The users Brad, Jim, DevApp1, DevApp2, TestApp1, and TestApp2 have been added to a single AWS account. Each user has its own credentials. Notice that some of the users are actually applications (for example, DevApp1). An **IAM user doesn't have to represent an actual person**; you can create an IAM user in order to generate an access key for an application that runs in your corporate network and needs AWS access.



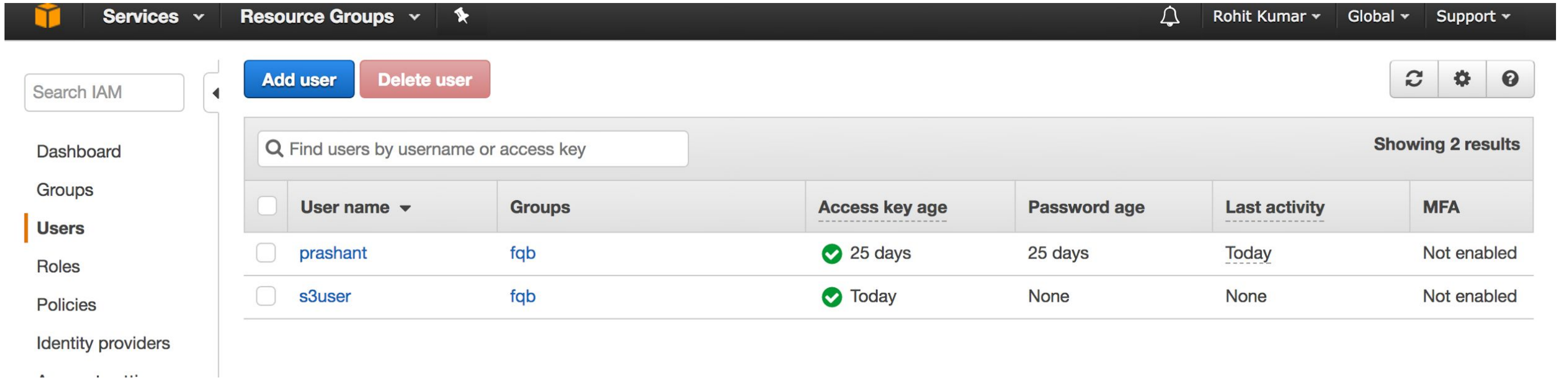
Federating Existing Users

- Your users already have identities in a corporate directory.
- Your users already have Internet identities.



IAM Users

An IAM *user* is an entity that you create in AWS to represent the person or service that uses it to interact with AWS. A user in AWS consists of a name and credentials. An IAM user with administrator permissions is not the same thing as the AWS account root user.



User name	Groups	Access key age	Password age	Last activity	MFA
prashant	fqb	25 days	25 days	Today	Not enabled
s3user	fqb	Today	None	None	Not enabled

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

Managing IAM Users

- Listing IAM Users
- Renaming an IAM User
  - To change a user's name or path, you must use the AWS CLI, Tools for Windows PowerShell, or AWS API. There is no option in the console to rename a user.
- Deleting an IAM User

Changing Permissions for a User

You can change the permissions for an IAM user in your AWS account by changing its group memberships or by attaching and detaching managed policies. A user gets its permissions through one of the following methods:

Group membership

- Add or remove a user from a group.
- Add, remove, or edit a managed policy attached to the group. This policy can be customer-created and managed, or it can be an AWS managed policy.
- Add, remove, or edit a group's inline policies. This kind of policy is always customer-created.

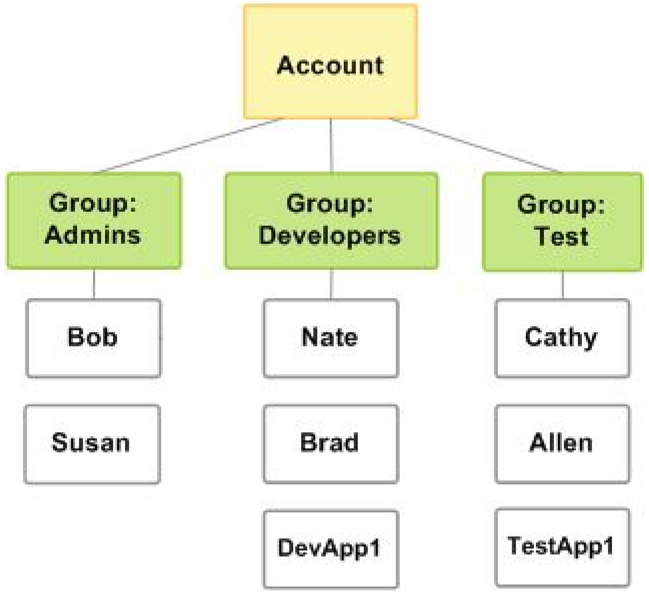
Direct policy attachment

- Add, remove, or edit a managed policy attached directly to a user. This policy can be customer-created and managed or it can be an AWS managed policy.
- Add, remove, or edit a user's inline policies. This kind of policy is always customer-created.

IAM Groups

An IAM *group* is a collection of IAM users. Groups let you specify permissions for multiple users, which can make it easier to manage the permissions for those users. For example, you could have a group called *Admins* and give that group the types of permissions that administrators typically need. Any user in that group automatically has the permissions that are assigned to the group. If a new user joins your organization and needs administrator privileges, you can assign the appropriate permissions by adding the user to that group. Similarly, if a person changes jobs in your organization, instead of editing that user's permissions, you can remove him or her from the old groups and add him or her to the appropriate new groups.

Getting Started



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

1. Create an Administrators group and give the group permission to access all of your AWS account's resources.
2. Create a user for yourself and add that user to the Administrators group.
3. Create a password for your user so you can sign in to the AWS Management Console.
4. You will grant the Administrators 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 in the Administrators group can also access your AWS account information, except for your AWS account's security credentials.

Practical: Creating an Administrator IAM User and Group

Creating an Administrator IAM User and Group

1. In the navigation pane, choose **Users** and then choose **Add user**.
2. For **User name**, type a user name, such as Administrator. The name can consist of letters, digits, and the following characters: plus (+), equal (=), comma (,), period (.), at (@), underscore (_), and hyphen (-). The name is not case sensitive and can be a maximum of 64 characters in length.
3. Select the checkbox next to AWS Management Console access, select Custom password, and then type your new password in the text box. If you're creating the user for someone other than yourself, you can optionally select Require password reset to force the user to create a new password when first signing in.
4. Choose Next: Permissions.
5. On the Set permissions for user page, choose Add user to group.

Creating an Administrator IAM User and Group

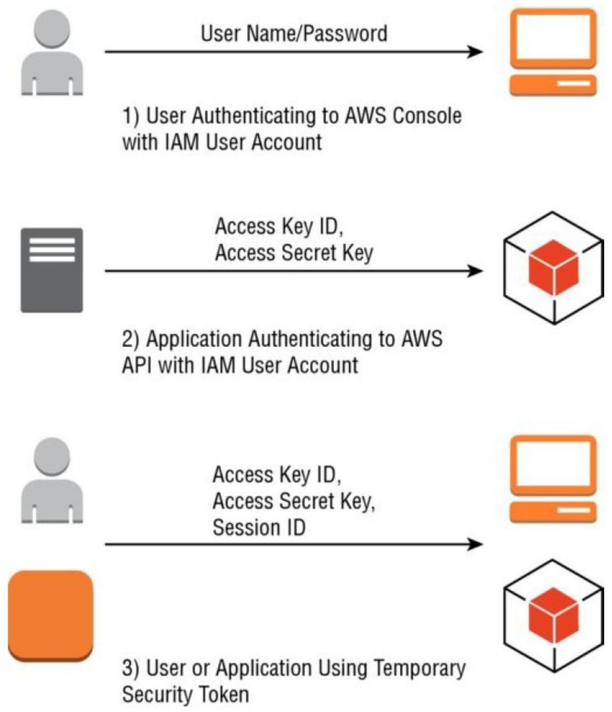
6. Choose Create group.
7. In the Create group dialog box, type the name for the new group. The name can consist of letters, digits, and the following characters: plus (+), equal (=), comma (,), period (.), at (@), underscore (_), and hyphen (-). The name is not case sensitive and can be a maximum of 128 characters in length.
8. In the policy list, select the checkbox next to AdministratorAccess. Then choose Create group.
9. Back in the list of groups, select the check box for your new group. Choose Refresh if necessary to see the group in the list.
10. Choose Next: Review to see the list of group memberships to be added to the new user. When you are ready to proceed, choose Create user.

Deleting an IAM Group

When you delete a group in the AWS Management Console, the console automatically removes all group members, detaches all attached managed policies, and deletes all inline policies.

Authentication

- User Name/Password
  - Human Interaction
- Access Key
  - access key ID (20 characters) and an access secret key (40 characters). When a program is manipulating the AWS infrastructure via the API
- Access Key/Session Token
  - access key (remember that it consists of two parts), the token also includes a session token


User Name/Password
1) User Authenticating to AWS Console with IAM User Account
Access Key ID, Access Secret Key
2) Application Authenticating to AWS API with IAM User Account
Access Key ID, Access Secret Key, Session ID
3) User or Application Using Temporary Security Token

Authorization

After IAM has authenticated a principal, it must then manage the access of that principal to protect your AWS infrastructure. The process of specifying exactly what actions a principal can and cannot perform is called authorization. Authorization is handled in IAM by defining specific privileges in policies and associating those policies with principals.

Authorization

Policies

Understanding how access management works under IAM begins with understanding policies. A policy is a JSON document that fully defines a set of permissions to access and manipulate AWS resources. Policy documents contain one or more permissions, with each
permission defining:

- **Effect**: A single word: Allow or Deny.
- **Service**: For what service does this permission apply? Most AWS Cloud services support granting access through IAM, including IAM itself.
- **Resource**: The resource value specifies the specific AWS infrastructure for which this permission applies. This is specified as an Amazon Resource Name (ARN). The format for an ARN varies slightly between services, but the basic format is:

  "Arn:aws:service:region:account-id:[resourcetype:]resource"

Authorization

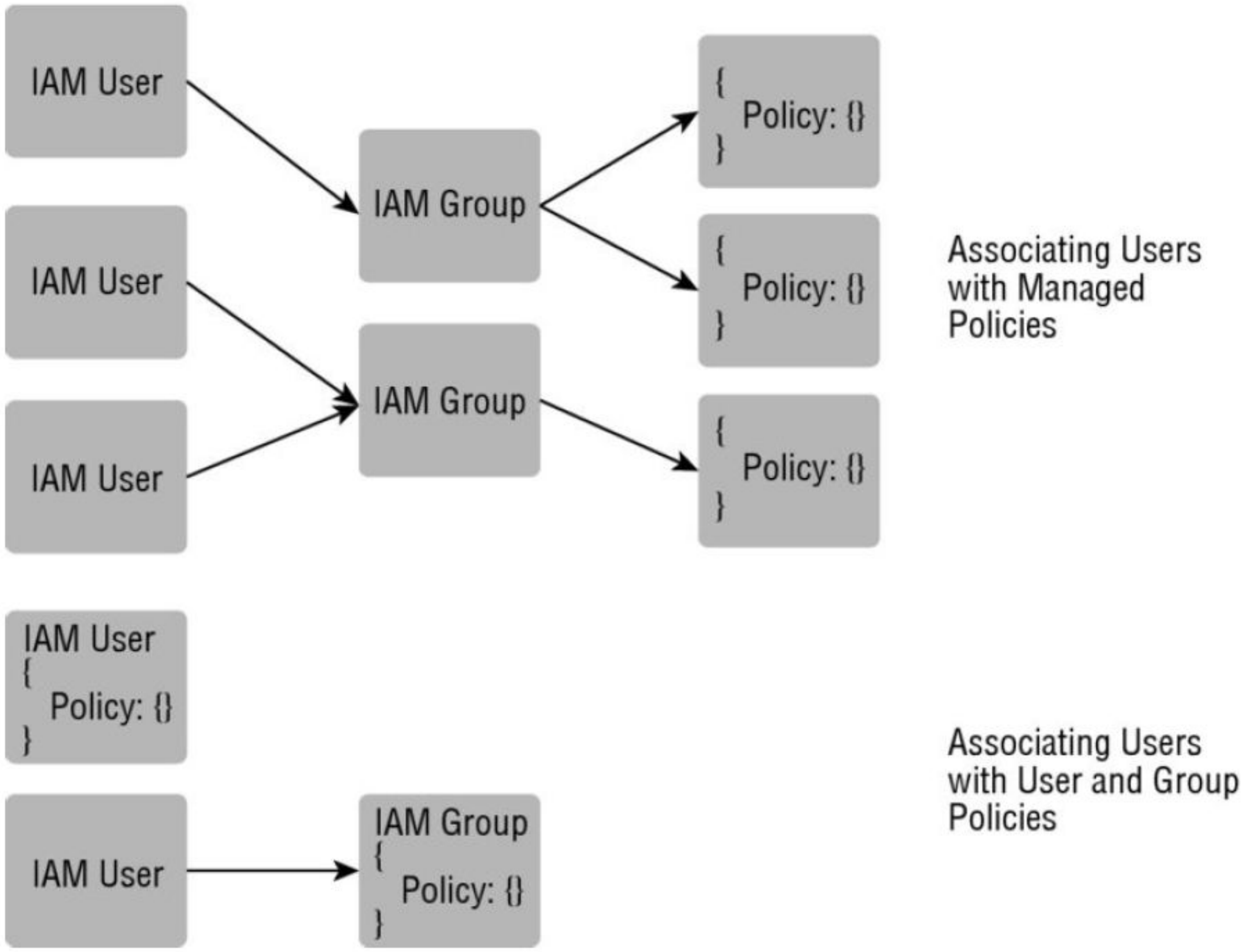
Policies

- **Action**: The action value specifies the subset of actions within a service that the permission allows or denies. For instance, a permission may grant access to any read based action for Amazon S3. A set of actions can be specified with an enumerated list or by using wildcards (Read*).
- **Condition**: The condition value optionally defines one or more additional restrictions that limit the actions allowed by the permission. For instance, the permission might contain a condition that limits the ability to access a resource to calls that come from a specific IP address range. Another condition could restrict the permission only to apply during a specific time interval. There are many types of permissions that allow a rich variety of functionality that varies between services.

Sample Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1441716043000",
            "Effect": "Allow",  <-  This policy grants access
            "Action": [ <-  Allows identities to list
                "s3:GetObject", <-  and get objects in
                "s3:ListBucket" <-  the S3 bucket
            ],
            "Condition": {
                "IpAddress": {                          <-  Only from a specific
                    "aws:SourceIp": "192.168.0.1"       <-  IP Address
                }
            },
            "Resource": [
                "arn:aws:s3:::my_public_bucket/*"       <-  Only this bucket
            ]
        }
    ]
}
```

Allows a principal to list the objects in a specific bucket and to retrieve those objects, but only if the call comes from a specific IP address.

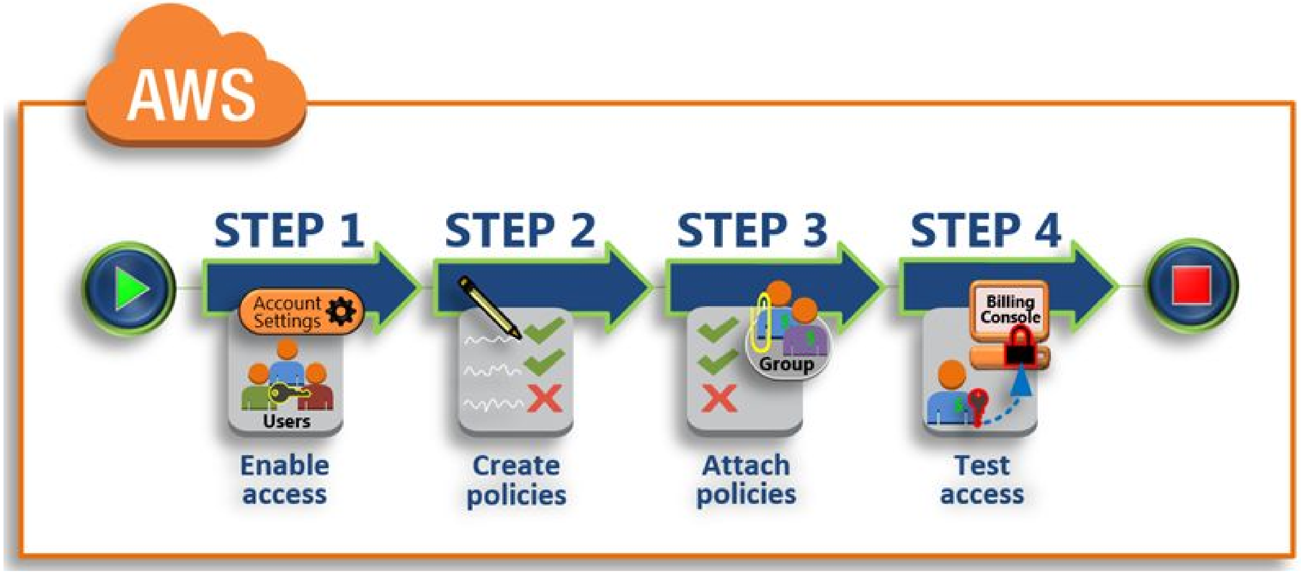


Different ways that policies can be associated with an IAM User

Practical: Delegate Access to the Billing Console

Delegate Access to the Billing Console

- Step 1: Enable Access to Billing Data on Your AWS Test Account
- Step 2: Create IAM Policies That Grant Permissions to Billing Data
- Step 3: Attach Billing Policies to Your Groups
- Step 4: Test Access to the Billing Console



1: Enable Access to Billing Data

- Sign in to the AWS Management Console with your root user credentials (the email and password that you used to create your AWS test account).

  Note

  This requires root user credentials. It cannot be performed by an IAM user. To view additional tasks that require you to sign in as the root user.

- On the navigation bar, choose your account name, and then choose **My Account**.
- Next to **IAM User and Role Access to Billing Information**, choose **Edit**, and then select the checkbox to activate IAM access to the Billing and Cost Management pages.
  - **Important**
  - When you activate IAM user access to the AWS website, you grant full access to the AWS website to all users who already have full access to the AWS APIs. You can restrict their access by applying a more constrained set of permissions.
- Sign out of the console.

2: Create IAM Policies Permissions to Billing Data

1. Sign in to the AWS Management Console as a user with administrator credentials. To adhere to IAM best practices, don't sign in with your root user credentials.
2. Open the IAM console
3. In the navigation pane, choose **Policies**, and then choose **Create Policy**.
4. Next to **Policy Generator**, choose **Select**.
5. On the **Edit Permissions** page, for **Effect** choose **Allow**.
6. For **AWS Service**, choose **AWS Billing**.
7. Follow these steps to create two policies:

Full access

- For **Actions**, choose **All Actions (*)**.
- Choose **Add Statement**, and then choose **Next Step**.
- On the **Review Policy** page, next to **Policy Name**, type `BillingFullAccess`, and then choose **Create Policy** to save it.

View-only access

- Repeat steps 3 through 6.
- For **Actions**, choose only those permissions that begin with **View**.
- Choose **Add Statement**, and then choose **Next Step**.
- On the **Review Policy** page, for **Policy Name**, type `BillingViewAccess`. Then choose **Create Policy** to

3: Attach Billing Policies to Your Groups

1. In the navigation pane, choose **Policies** to display the full list of policies available to your AWS account. To attach each policy to its appropriate group, follow these steps:
2. **Full access**
    - In the search box, type **`BillingFullAccess`**, and then select the check box next to the policy name.
    - Choose **Policy actions**, and then choose **Attach**.
    - In the search box, type **`FullAccess`**, select the check box next to the name of the group, and then choose **Attach policy**.
3. **View-only access**
    - In the search box, type **`BillingViewAccess`**, and then select the check box next to the policy name.
    - Choose **Policy actions**, and then choose **Attach**.
    - For **Filter**, choose **Groups**. In the search box, type **`ViewAccess`**, select the check box next to the name of the group, and then choose **Attach policy**.
4. Sign out of the console

4: Test Access to the Billing Console

1. Go to the sign-in URL for your AWS test account. For example, if your AWS account name is CompanyXYZ, your sign-in URL would look like `https://companyxyz.signin.aws.amazon.com/console`. If you did not assign an alias like CompanyXYZ, then use your account ID number as in this example:`https://123456789012.signin.aws.amazon.com/console.`
2. Sign-in with each account using the steps provided below so you can compare the different user experiences.
3. **Full access**
   - Sign in to your AWS account as the user FinanceManager.
   - On the navigation bar, choose **FinanceManager@*`<account alias or ID number>`*** , and then choose **Billing & Cost Management**.
   - Browse through the pages and choose the various buttons to ensure you have full modify permissions.
4. **View-only access**
   - Sign in to your AWS account as the user FinanceUser.
   - On the navigation bar, choose **FinanceUser@*`<account alias or ID number>`***, and then choose **Billing & Cost Management**.
   - Browse through the pages. Notice that you can display costs, reports, and billing data with no problems. However, if you choose an option to modify a value, you receive an **Access Denied** message. For example, on the **Preferences** page, choose any of the check boxes on the page, and then choose **Save preferences**. The console message informs you that you need **ModifyBilling** permissions to make changes to that page.

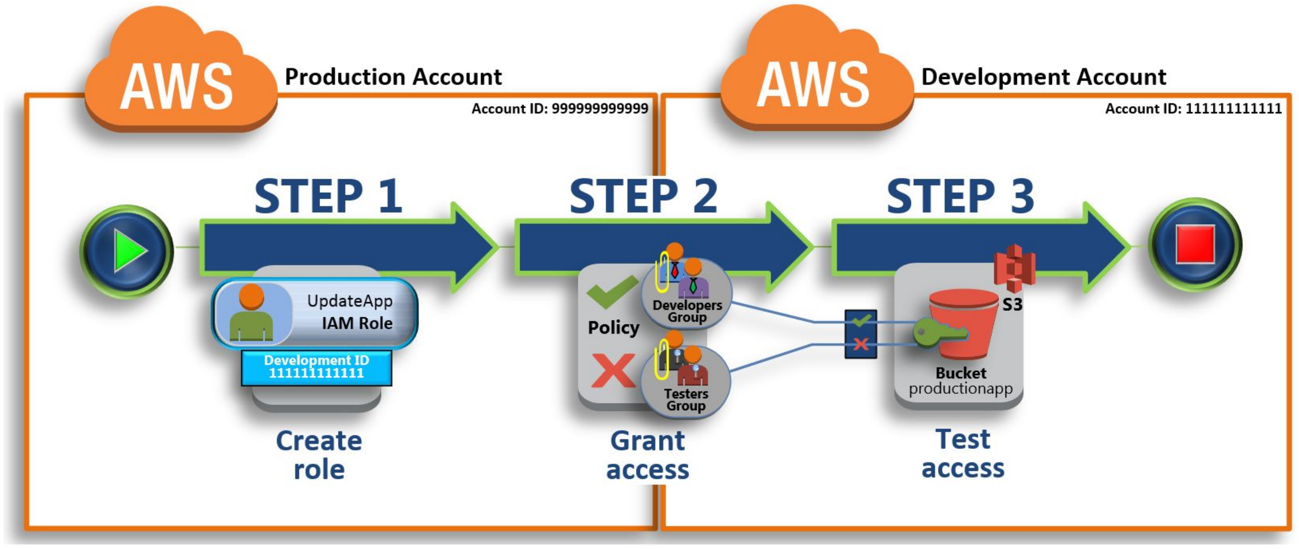
IAM policy simulator

1. Open the IAM policy simulator at https://policysim.aws.amazon.com/. (If you are not already signed in to AWS, you are prompted to sign in).
2. Under **Users, Groups, and Roles**, select one of the users that is a member of the group you recently attached the policy to.
3. Under **Policy Simulator**, choose **Select service**, and then choose **Billing**.
4. Next to **Select actions**, choose **Select All.**
5. Choose **Run Simulation** and compare the user's listed permissions with all possible billing-related permission options to make sure that the correct rights have been applied.

Practical: Delegate Access Across AWS Accounts Using IAM Roles

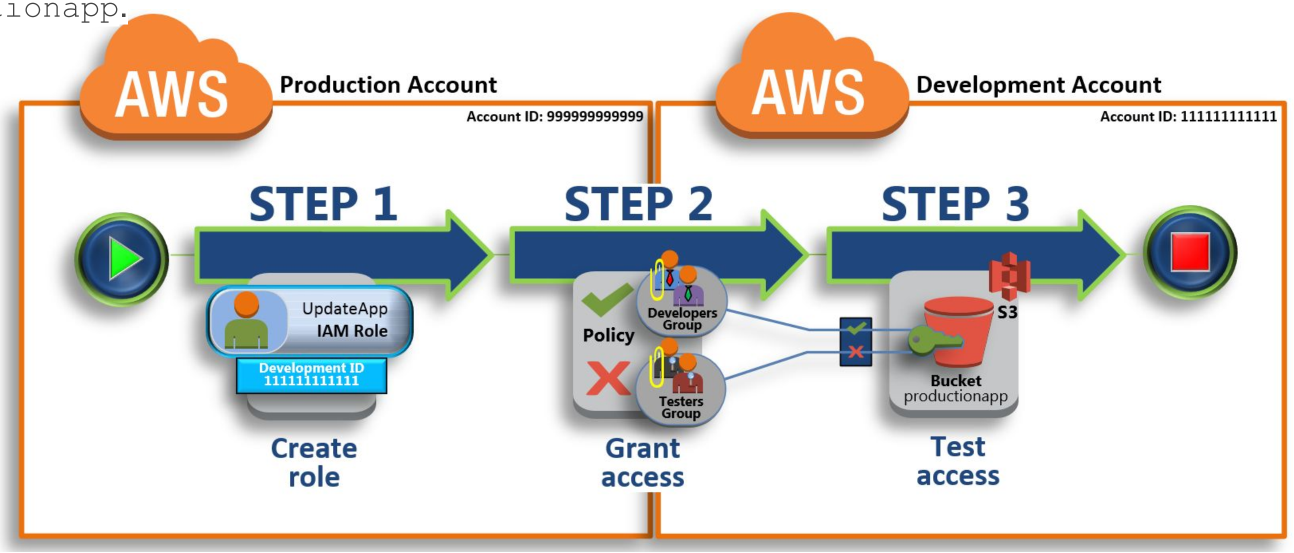
Delegate Access Across AWS Accounts Using IAM Roles

Delegate access to resources that are in different AWS accounts that you own (Production and Development)



Problem Statement

Imagine that 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. You manage IAM users in the Development account, where you have two IAM groups: Developers and Testers. Users in both groups have permissions to work in the Development account and access resources there. From time to time, a developer must update the live applications in the Production account. These applications are stored in an Amazon S3 bucket called `productionapp`.



Steps

- **Step 1 - Create a Role**
  - First, you use the AWS Management Console to establish trust between the Production account (ID number 999999999999) and the Development account (ID number 111111111111). You start by creating an IAM role named *UpdateApp*. When you create the role, you define the Development account as a trusted entity and specify a permissions policy that allows trusted users to update the `productionapp` bucket.
- **Step 2 - Grant Access to the Role**
  - In this step of the tutorial, you modify the IAM group policy so that Testers are denied access to the `UpdateAPP` role. Because Testers have PowerUser access in this scenario, we must explicitly deny the ability to use the role.
- **Step 3 - Test Access by Switching Roles**
  - Finally, as a Developer, you use the `UpdateAPP` role to update the `productionapp` bucket in the Production account. You see how to access the role through the AWS console, the AWS CLI, and the API.

Step 1 - Create a Role

To allow users from one AWS account to access resources in another AWS account, create a role that defines who can access it and what permissions it grants to users that switch to it.

1. Sign in to the AWS Management Console as an administrator of the Production account, and open the IAM console.
2. Before creating the role, prepare the managed policy that defines the permissions that the role requires. You attach this policy to the role in a later step.
   You want to set read and write access to the `productionapp` bucket. Although AWS provides some Amazon S3 managed policies, there isn't one that provides read and write access to a single Amazon S3 bucket. You can create your own policy instead.
   In the navigation pane on the left, choose **Policies** and then choose **Create policy**.
3. Next to **Create Your Own Policy**, choose **Select**.
4. Enter **`read-write-app-bucket`** for the policy name.
5. Add the following permissions to the policy document. Ensure that you replace the resource ARN (`arn:aws:s3:::productionapp`) with the real one appropriate to your S3 bucket.

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

Step 1 - Create a Role

6. Choose **Create policy**.
   The new policy appears in the list of managed policies.
7. In the navigation pane on the left, choose **Roles** and then choose **Create new role**.
8. Under **Select role type**, choose **Role for cross-account access**, and then choose **Select** next to **Provide access between AWS accounts you own**.
9. Enter the Development account ID.
10. Choose **Next Step** to set the permissions that will be associated with the role.
11. Select the box next to the policy that you created previously.
12. Type **`UpdateAPP`** for the role name.
13. (Optional) For **Role description**, type a description for the new role.
14. Confirm the settings for the role before it's created. One very important item to note on this page is the link that you can send to your users who need to use this role. Users who choose the link go straight to the **Switch Role** page with the **Account ID** and **Role name** fields already filled in. You can also see this link later on the **Role Summary** page for any cross-account role.
15. After reviewing the role, choose **Create role**.
16. The `UpdateAPP` role appears in the list of roles.

Step 2 - Grant Access to the Role

To modify the Developers group to allow them to switch to the UpdateApp role

1. Sign in as an administrator in the Development account, and open the IAM console.
2. Choose **Groups**, and then choose **Developers**.
3. Choose the **Permissions** tab, expand the **Inline Policies** section, and then choose **Create Group Policy**. If no inline policy exists yet, then the button does not appear. Instead, choose the link at the end of "To create one, click here."
4. Choose **Custom Policy** and then choose **Select** button.
5. Type a policy name like **`allow-assume-S3-role-in-production`**.
6. Add the following policy statement to allow the `AssumeRole` action on the `UpdateAPP` role in the Production account. Be sure that you change *`PRODUCTION-ACCOUNT-ID`* in the `Resource` element to the actual AWS account ID of the Production account.

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

Step 2 - Grant Access to the Role

7. Choose **Apply Policy** to add the policy to the Developer group.

To modify the Testers group to deny permission to assume the UpdateApp role

1. Choose **Groups**, and then choose **Testers**.
2. Choose the **Permissions** tab, expand the **Inline Policies** section, and then choose **Create Group Policy**.
3. Choose **Custom Policy** and then choose the **Select** button.
4. Type a policy name like **`deny-assume-S3-role-in-production`**.

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

Step 3 - Test Access by Switching Roles

Test

Practical: Create and Attach Your First Customer Managed Policy

Step 1: Create the Policy

In this step, you create a customer managed policy that allows any attached user to sign in to the AWS Management Console with read-only access to IAM data.

To create the policy for your test user

1. Sign in to the IAM console at https://console.aws.amazon.com/iam/ with your user that has administrator permissions.
2. In the navigation pane, choose **Policies**.
3. In the content pane, choose **Create Policy**.
4. Next to **Create Your Own Policy**, choose **Select**.
5. For **Policy Name**, type **`UsersReadOnlyAccessToIAMConsole`**.
6. For **Policy Document**, paste the following policy.
7. Choose **Validate Policy** and ensure that no errors display in a red box at the top of the screen. Correct any that are reported.
8. **Note**
9. If **Use autoformatting for policy editing** is selected, the policy is reformatted whenever you open a policy or choose **Validate Policy**.
10. Choose **Create Policy**.
11. Now Policy Ready to attach

Copy

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

Step 2: Attach the Policy

Next you attach the policy you just created to your test IAM user.
To attach the policy to your test user

1. In the IAM console, in the navigation pane, choose **Policies**.
2. At the top of the policy list, in the search box, start typing `UsersReadOnlyAccesstoIAMConsole` until you can see your policy, and then check the box next to **UsersReadOnlyAccessToIAMConsole** in the list.
3. Choose the **Policy actions** button, and then chose **Attach**.
4. For **Filter**, choose **Users**.
5. In the search box, start typing `PolicyUser` until that user is visible on the list, and then check the box next to that user in the list.
6. Choose **Attach Policy**.

You have attached the policy to your IAM test user, which means that user now has read-only access to the IAM console.

Step 3: Test User Access

To test access by signing in with your test user account

1. Sign in to the IAM console at https://console.aws.amazon.com/iam/ with your `PolicyUser` test user.
2. Browse through the pages of the console and try to create a new user or group. Notice that `PolicyUser` can display data but cannot create or modify existing IAM data.

IAM Best Practices and Use Cases

To get the greatest benefits from IAM, take time to learn the recommended best practices. One way to do this is to see how IAM is used in real-world scenarios to work with other AWS services.

Topics

- IAM Best Practices
- Business Use Cases

IAM Best Practices

- Lock Away Your AWS Account Root User Access Keys
- Create Individual IAM Users
- Use AWS Defined Policies to Assign Permissions Whenever Possible
- Use Groups to Assign Permissions to IAM Users
- Grant Least Privilege
- Use Access Levels to Review IAM Permissions
- Configure a Strong Password Policy for Your Users
- Enable MFA for Privileged Users
- Use Roles for Applications That Run on Amazon EC2 Instances
- Delegate by Using Roles Instead of by Sharing Credentials
- Rotate Credentials Regularly
- Remove Unnecessary Credentials
- Use Policy Conditions for Extra Security
- Monitor Activity in Your AWS Account

Some IAM Use Cases

Companies

- Developers, Admins, Testers, Analytics, PM
- User Role changes
- Directory Access Restriction

```
/example_bucket
    /home
        /don
        /mary
    /share
        /developers
        /managers
```

Other Key Features

- Multi-Factor Authentication (MFA)
  - extra layer of security to your infrastructure by adding a second method of authentication
- Rotating Keys
  - IAM facilitates this process by allowing two active access keys at a time
- Resolving Multiple Permissions
  - Initially the request is denied by default.
  - All the appropriate policies are evaluated; if there is an explicit “deny” found in any policy, the request is denied and evaluation stops.
  - If no explicit “deny” is found and an explicit “allow” is found in any policy, request is allowed.
  - If there are no explicit “allow” or “deny” permissions found, then the default “deny” is maintained and the request is denied.
  - The only exception to this rule is if an AssumeRole call includes a role and a policy, the policy cannot expand the privileges of the role (for example, the policy cannot override any permission that is denied by default in the role)

Summary

IAM is a powerful service that gives you the ability to control which people and applications can access your AWS account at a very granular level. Because the root user in an AWS account cannot be limited, you should set up IAM users and temporary security tokens for your people and processes to interact with AWS.

Policies define what actions can and cannot be taken. Policies are associated with IAM users either directly or through group membership. A temporary security token is associated with a policy by assuming an IAM role. You can write your own policies or use one of the managed policies provided by AWS.

Common use cases for IAM roles include federating identities from external IdPs, assigning privileges to an Amazon EC2 instance where they can be assumed by applications running on the instance, and cross-account access.

IAM user accounts can be further secured by rotating keys, implementing MFA, and adding conditions to policies. MFA ensures that authentication is based on something you have in addition to something you know, and conditions can add further restrictions such as limiting client IP address ranges or setting a particular time interval.

Exam Essentials

- **Know the different principals in IAM.** The three principals that can authenticate and interact with AWS resources are the root user, IAM users, and roles. The root user is associated with the actual AWS account and cannot be restricted in any way. IAM users are persistent identities that can be controlled through IAM. Roles allow people or processes the ability to operate temporarily with a different identity. People or processes assume a role by being granted a temporary security token that will expire after a specified period of time.
- **Know how principals are authenticated in IAM**. When you log in to the AWS Management Console as an IAM user or root user, you use a user name/password combination. A program that accesses the API with an IAM user or root user uses a two-part access key. A temporary security token authenticates with an access key plus an additional session token unique to that temporary security token.
- **Know the parts of a policy.** A policy is a JSON document that defines one or more permissions to interact with AWS resources. Each permission includes the effect, service, action, and resource. It may also include one or more conditions. AWS makes many predefined policies available as managed policies.

Exam Essentials

- **Know how a policy is associated with a principal.** An authenticated principal is associated with zero to many policies. For an IAM user, these policies may be attached directly to the user account or attached to an IAM group of which the user account is a member. A temporary security token is associated with policies by assuming an IAM role.
- **Understand MFA.** MFA increases the security of an AWS account by augmenting the password (something you know) with a rotating OTP from a small device (something you have), ensuring that anyone authenticating the account has both knowledge of the password and possession of the device. AWS supports both Gemalto hardware MFA devices and a number of virtual MFA apps.
- **Understand key rotation.** To protect your AWS infrastructure, access keys should be rotated regularly. AWS allows two access keys to be valid simultaneously to make the rotation process straightforward: Generate a new access key, configure your application to use the new access key, test, disable the original access key, test, delete the original access key, and test again.

Exam Essentials

- **Understand IAM roles and federation.** IAM roles are prepackaged sets of permissions that have no credentials. Principals can assume a role and then use the associated permissions. When a temporary security token is created, it assumes a role that defines the permissions assigned to the token. When an Amazon EC2 instance is associated with an IAM role, SDK calls acquire a temporary security token based on the role associated with the instance and use that token to access AWS resources. Roles are the basis for federating external IdPs with AWS. You configure an IAM IdP to interact with the external IdP, the authenticated identity from the IdP is mapped to a role, and a temporary security token is returned that has assumed that role. AWS supports both SAML and OIDC IdPs.
- **Know how to resolve conflicting permissions.** Resolving multiple permissions is relatively straightforward. If an action on a resource has not been explicitly allowed by a policy, it is denied. If two policies contradict each other; that is, if one policy allows an action on a resource and another policy denies that action, the action is denied. While this sounds improbable, it may occur due to scope differences in a policy. One policy may expose an entire fleet of Amazon EC2 instances, and a second policy may explicitly lock down one particular instance.

Test

Test