**What is Identity Access Management (IAM)?**

AWS Identity Access Management is a service that helps to control access to AWS resources securely.

Identity Access Management (IAM) is used to control who is authenticated (signed in) and authorized (has permissions) to access services/resources.

When we create an AWS account first time it has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user*. It is accessed by signing in with the email address and password that you used to create the account.

AWS strongly recommend that do not use root user for everyday tasks, even for the administrative tasks also. Instead create a user with administrator policy for managing all tasks.

Let’s consider i have 10 employees in my company, 7 of them are developers and others are admins. 4 out of 7 Developers want to use AWS developer services like EC2, Lambda, S3 and remaining want to use Sagemaker, API Gateway.

Creating individual AWS account for each employee is not possible because company needs to pay money for each account. So, we have to create only one AWS account and all employees in my company has to access services using this account. But, it’s not good to share root user credentials to all.

So, AWS has introduced the concept called Users in IAM. we can create an IAM user for each employee in our company and attach policies to each user as per the services he wants to access.

As per our example 4 developers wants to access same services. So, instead of attaching policies to each user separately we will group them together and attach the policies to this group.

**IAM features**

IAM gives you the following features.

**1. 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.

**2. 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 DynamoDB. 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.

**3. Secure access to AWS resources for applications that run on Amazon EC2**

You can use IAM features to securely provide credentials for applications that run on EC2 instances. These credentials provide permissions for your application to access other AWS resources. Examples include S3 buckets and DynamoDB tables.

**4. 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.

**5. 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.

**6. Identity information for assurance**

If you use [AWS CloudTrail](http://www.amazonaws.cn/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.

**7. 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). For more information about PCI DSS, including how to request a copy of the AWS PCI Compliance Package, see [PCI DSS Level 1](http://www.amazonaws.cn/compliance/pci-dss-level-1-faqs/).

**8. Integrated with many AWS services**

For a list of AWS services that work with IAM, see [AWS services that work with IAM](https://docs.amazonaws.cn/en_us/IAM/latest/UserGuide/reference_aws-services-that-work-with-iam.html).

**9. Eventually Consistent**

IAM, like many other AWS services, is [eventually consistent](https://wikipedia.org/wiki/Eventual_consistency). 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. For more information, see [Changes that I make are not always immediately visible](https://docs.amazonaws.cn/en_us/IAM/latest/UserGuide/troubleshoot_general.html#troubleshoot_general_eventual-consistency).

**10. Free to use**

AWS Identity and Access Management (IAM) and AWS Security Token Service (AWS STS) are features of your AWS account offered at no additional charge. You are charged only when you access other AWS services using your IAM users or AWS STS temporary security credentials. For information about the pricing of other AWS products, see the [Amazon Web Services pricing page](http://www.amazonaws.cn/pricing/).

IAM mainly contains 4 blocks.

1. Users
2. Groups
3. Roles
4. Policies

**Users**

Instead of sharing our root user credentials with others, we can create individual IAM users within our account that correspond to users in our organization. So, IAM users are not separate AWS accounts, they are users within our root AWS account.

Each user can have its own password for access to the AWS Management Console. We can also create an individual access key for each user so that the user can make programmatic requests to work with resources in our account.

**Note:** 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.

AWS recommend that we create an IAM user for ourself and then assign ourself administrative permissions for our account. we can then sign in as that user to add more users as needed.

**Federating existing users**

If the users in your organization already have a way to be authenticated, such as by signing in to your corporate network, you don't have to create separate IAM users for them. Instead, you can *federate* those user identities into AWS.

The following diagram shows how a user can use IAM to get temporary AWS security credentials to access resources in your AWS account.


        Users who are already authenticated elsewhere can be federated into AWS without
          requiring an IAM user.
      

Federation is particularly useful in these cases:

1. **Your users already have identities in a corporate directory.**

If your corporate directory is compatible with Security Assertion Markup Language 2.0 (SAML 2.0), you can configure your corporate directory to provide single-sign on (SSO) access to the AWS Management Console for your users.

If your corporate directory is not compatible with SAML 2.0, you can create an identity broker application to provide single-sign on (SSO) access to the AWS Management Console for your users.

If your corporate directory is Microsoft Active Directory, you can use [AWS Directory Service](http://www.amazonaws.cn/directoryservice/) to establish trust between your corporate directory and your AWS account.

1. **Your users already have Internet identities.**

If you are creating a mobile app or web-based app that can let users identify themselves through an Internet identity provider like Login with Amazon, Facebook, Google, or any OpenID Connect (OIDC) compatible identity provider, the app can use federation to access AWS.

**Note:** To use identity federation with Internet identity providers, we recommend you use [Amazon Cognito](https://docs.amazonaws.cn/cognito/latest/developerguide/what-is-amazon-cognito.html).

**Note:** We can add the same user to more than one group.

**Groups**

Groups are nothing but organizing users based on the permissions/policies required.

Let’s say we have 10 users and 5 users out of 10 are required full access to amazon s3 service. So instead of giving permission to all 5 users individually we make them as group and attach required permission policy to the group.

**Policies (permission or authorization)**

The access management portion of AWS Identity and Access Management (IAM) helps you define what a principal entity is allowed to do in an account.

A principal entity is nothing but a person or application that is authenticated.

You manage access in AWS by creating policies and attaching them to IAM identities (users, groups of users, or roles) or AWS resources.

Let’s say we have a user and he want to get the data from AWS S3 bucket. So, the user should have read permission (Authorization) on S3 service to get the data. We will create a policy with read access to S3 and attach it to the user.

**Types of Policies:**

1. AWS Managed policies.
2. Customer managed policies. # Created and managed by us (admin user).
3. In-line policies. # This option can see under EC2 attached IAM role in EC2.

**What is Inline policy?**

So, generally we create managed policies for the purpose of reusing. This means we can attach same policy to different users, groups or roles. But in-line policies are policies that we can’t reuse them. in-line policy is embedded directly into a single user, that policy will applicable for that user only. We can’t attach same policy for other user, if we want, we have to create a new separate in-line policy.

Policy contains an array named as statement which having json objects in it. Each json object having 3 fields.

1. Effect =🡺 Allow or Deny
2. Action =🡺 What actions / permissions granting
3. Resource =🡺 On which resource

Ex:

"Statement": [

{

"Effect": "Allow",

"Action": "ec2:Describe\*",

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": [

"cloudwatch:ListMetrics",

"cloudwatch:GetMetricStatistics",

"cloudwatch:Describe\*"

],

"Resource": "\*"

},

]

**Roles**

An IAM *role* is an IAM identity that you can create in your account that has specific permissions/policies.

An IAM role is similar to an IAM user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS.

However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session.

You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources.

IAM roles are a secure way to grant permissions to entities that you trust. Examples of entities include the following:

* IAM user in another account
* Application code running on an EC2 instance that needs to perform actions on AWS resources
* An AWS service that needs to act on resources in your account to provide its features
* Users from a corporate directory who use identity federation with SAML

IAM roles issue keys that are valid for short durations, making them a more secure way to grant access.

**IAM password policy**

Use a password policy to require your IAM users to create strong passwords and to rotate their passwords regularly.

We have two types of password policies.

1. Password policy
2. Security token service (STS).

We can see these options in **Account Settings** menu in IAM Dashboard.

**Password policy**

A password policy is a set of rules that define the type of password an IAM user can set.

A password policy is a set of rules that define complexity requirements and mandatory rotation periods for your IAM users' passwords.

**Select your account password policy requirements:**

Enforce minimum password length

  characters

Require at least one uppercase letter from Latin alphabet (A-Z)

Require at least one lowercase letter from Latin alphabet (a-z)

Require at least one number

Require at least one non-alphanumeric character (!@#$%^&\*()\_+-=[]{}|')

Enable password expiration

Expire passwords in  day(s) # Default is 90 days.

Password expiration requires administrator reset

Allow users to change their own password

Prevent password reuse

Remember  password(s) # Default is 5

**Security Token Service (STS)**

AWS recommends using regional STS endpoints to reduce latency. Session tokens from regional STS endpoints are valid in all AWS Regions. If you use regional STS endpoints, no action is required.

Session tokens from the global STS endpoint (https://sts.amazonaws.com) are valid only in AWS Regions that are enabled by default.

If you intend to enable a new Region for your account, you can use session tokens from regional STS endpoints or activate the global STS endpoint to issue session tokens that are valid in all AWS Regions.

**Multi Factor Authentication (MFA)**

AWS offer increased security to AWS resources by using multi factor authentication. We can use MFA to IAM user accounts or AWS Root user account. If we enable MFA for root user that will affects to root user only. IAM users in the account are distinct identities with their own credentials, and each identity has its own MFA configuration.

MFA adds extra security because it requires users to provide unique authentication from an AWS supported MFA mechanism in addition to their regular sign-in credentials when they access AWS websites or services.

We can choose MFA code to be delivered in following ways.

1. Virtual MFA devices
2. U2F security key
3. Hardware MFA device
4. SMS text message-based MFA

**Virtual MFA Device**

A software app that runs on a phone or other device and emulates a physical device. The device generates a six-digit numeric code based upon a time-synchronized one-time password algorithm.

The user must type a valid code from the device on a second webpage during sign-in.

Each virtual MFA device assigned to a user must be unique.

A user cannot type a code from another user's virtual MFA device to authenticate. Because they can run on unsecured mobile devices, virtual MFA might not provide the same level of security as U2F devices or hardware MFA devices.

We do recommend that you use a virtual MFA device while waiting for hardware purchase approval or while you wait for your hardware to arrive.

**U2F security key**

A device that you plug into a USB port on your computer.

U2F is an open authentication standard hosted by the [FIDO Alliance](https://fidoalliance.org/). When you enable a U2F security key, you sign in by entering your credentials and then tapping the device instead of manually entering a code.

**Hardware MFA device**

This is same as virtual MFA device. Instead of mobile you will have a dedicated hardware device specially designed for MFA authentication.

A hardware device that generates a six-digit numeric code based upon a time-synchronized one-time password algorithm.

**SMS text message-based MFA**

A type of MFA in which the IAM user settings include the phone number of the user's SMS-compatible mobile device.

When the user signs in, AWS sends a six-digit numeric code by SMS text message to the user's mobile device.

**Note:** Note that SMS-based MFA is available only for IAM users. You cannot use this type of MFA with the AWS account root user.

**Note:** AWS will soon end support for SMS multi-factor authentication (MFA). We are not allowing new customers to preview this feature. We recommend that existing customers switch to one of the other methods.