IAM

- Introduction to AWS Identity and Access Management (IAM)
- IAM Features
- Multi-Factor Authentication & MFA Form Factors
- Virtual MFAS
- **User's Security Credentials**
- Temporary security credentials
- Manage Permissions and Policies
- Manage Federation
- IAM Roles
- Manage Users, Use Cases
- IAM Groups, Accessing IAM

AWS Identity and Access Management (IAM)

- AWS Identity and Access Management (IAM) enables us to securely control access to AWS services and resources for the users.
- Using IAM, we can create and manage AWS users and groups, and assign permissions to allow and deny their access to AWS resources.
- AWS Identity and Access Management (IAM) 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** in the cloud that use AWS products such as **Amazon EC2**, Amazon **RDS**, and the **AWS**

Management Console. With IAM, we can **centrally** manage users, security credentials such as **access keys**, and permissions that control AWS resources for the users.

Free to use

- AWS Identity and Access Management is a feature of our AWS account offered at **no additional charge**. We will be charged only for use of other AWS products by our IAM users

IAM Features

Shared access to our AWS account

- We can **grant other people permission to administer and use resources in our AWS account** without sharing our password or access key.

Granular permissions

- we can grant **different permissions** to different people for different resources.
- **For example**, we might allow some users **for complete access** to Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), Amazon RDS, and other AWS products.
- For other users, we might allow **read-only access** to just some S3 buckets, or permission to administer just some EC2 instances, or to access our billing information but nothing else.

IAM Features

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

We can use IAM features to **securely access applications** that run on EC2 instances**.**

Multi-factor authentication (MFA)

We can add two-factor authentication to our account and to individual users for extra security.

With MFA, our users must provide not only a password or access key to work with the account, **but also a code from a specially configured device** that will be given by vendor (chargeable).

Identity federation

We can allow users who already have passwords elsewhere—for example, in our corporate network or with an Internet identity provider—to get temporary access to our AWS account.

Identity information for assurance

If we use AWS CloudTrail, we receive log records that include information about those who made requests for resources in our account. That information is based on IAM identities.

Enhanced Security

IAM enables security best practices by allowing us to grant unique security credentials to users and groups to specify which AWS service APIs and resources we can access.

Temporary Credentials

In addition to define access permissions directly to users and groups, IAM lets us to create roles. Roles allow us to define a set of permissions and authentication for users or EC2 instances (temporarily).

Flexible security credential management

We can assign a range of security credentials including passwords, key pairs, and X509 certificates. we can also enforce multi-factor authentication (MFA) on users who access the AWS Management Console or use APIs.

Leverage external identity systems

We can use IAM to grant our employees and applications access to the AWS Management Console and to AWS service APIs. AWS supports federation from corporate systems like Microsoft Active Directory, SDS, as well as external Web Identity Providers like Google and Facebook

Seamlessly integrated into AWS services

IAM is integrated into most AWS services through AWS Management Console that will take effect throughout our AWS environment.

Multi-Factor Authentication

- AWS Multi-Factor Authentication (MFA) is a simple best practice that adds an extra layer of protection on top of our user name and password.
- With MFA enabled, when a user signs in to an AWS website or environment, they will be prompted for their user name and password (the first factor—what they know), as well as for an authentication code from their AWS MFA device (the second factor—what they have). Taken together, these multiple factors provide increased security for our AWS account settings and resources.
- we can enable MFA for our AWS account and for individual IAM users we have created under our account.
- After we've obtained a **supported hardware** or **virtual MFA device**, AWS does not charge any additional fees for using MFA.

	Virtual MFA Device	Hardware Key Fob MFA Device	Hardware Display Card MFA Device
Device	See table below.		Purchase device
Physical Form Factor	Use our existing smartphone or tablet running any application that supports the open TOTP standard.	Tamper-evident hardware key fob device provided by Gemalto, a third-party provider.	Tamper-evident hardware display card device provided by Gemalto, a third-party provider.
Price	Free	12.99	19.99
Security	Better	Best	Best
Features	Support for multiple tokens on a single device.	The same type of device used by many financial services and enterprise IT organizations.	Similar to key fob devices, but in a convenient form factor that fits in our wallet like a credit card.

Virtual MFA Applications

- MFA Applications for our smartphone can be installed from the application store that is specific to our phone type.
- The following table lists some applications for different smartphone types.

Android	AWS Virtual MFA; Google Authenticator
iPhone	Google Authenticator
Windows Phone	Authenticator
Blackberry	Google Authenticator

Manage Users' Security Credentials

AWS Identity and Access Management (IAM) lets us manage several types of long-term security credentials for IAM users:

- **Passwords –** Used to sign in to secure AWS pages, such as the AWS Management Console, the AWS Discussion Forums, and the AWS Premium Support site.
- **Access keys and Key Pairs –** Used to make secure Query protocol requests to any AWS service API.
- we can assign AWS security credentials to our IAM users by using the API, CLI, or AWS Management Console. we can rotate or revoke these credentials whenever we want.
- In addition to managing these user credentials, we can further enhance the security of IAM user access to AWS by enforcing the use of **multi-factor authentication (MFA).**

Temporary security credentials

- IAM also lets us grant any user temporary security credentials with a **defined expiration** for access to our AWS resources.

For example, temporary access is useful when:

- Creating a mobile app with third-party sign-in.
- Creating a mobile app with custom authentication.
- Using our organization's authentication system to grant access to AWS resources.
- Using our organization's authentication system and SAML to grant access to AWS resources.
- Security Assertion Markup Language (SAML, pronounced sam-el) is an XML-based, open-standard data format for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider

- Using web-based Single Sign-On (SSO) to the AWS Management Console.
  Delegating API access to third parties to access resources in our account or in another account we own.
- All are for temporary purposes.

Manage Permissions and Policies

- Permissions let us specify, who has access to AWS resources and which actions they can perform on those resources.
- Every AWS Identity and Access Management (IAM) user 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, we can 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**

To assign permissions to a user, group, role, or resource, we create a policy that lets us specify:

- **Actions –** Which AWS actions we allow.

  For example, we might allow a user to call the Amazon S3 ListBucket action.
- **Resources –** Which AWS resources we allow the action on.

  For example, what Amazon S3 buckets will we allow the user to perform the ListBucket action on? **Effect –** Whether to allow or deny access.

  Any actions/resources/effect that we don't explicitly allow are denied.
- **Conditions –** Which conditions must be present for the policy to take effect.

  For example, we might allow access only to the specific S3 buckets if the user is connecting from a specific IP range or has used multi-factor authentication at login.

- Policies are created using JSON by developers. A policy consists of one or more statements, each of which describes one set of permissions.
- If we use the AWS Management Console to manage permissions, we can select a predefined policy. Any policy can be customized.

Manage Federation

- AWS Identity and Access Management (IAM) supports identity federation for **delegated access** to the AWS Management Console or AWS APIs.
- With identity federation, external identities (**federated users**) are granted secure access to resources in our AWS account without having to create IAM users.
- These external identities can come from our corporate identity provider (such as Microsoft Active Directory or from the AWS Directory Service) or from a web identity provider, such as Amazon Cognito, Login with Amazon, Facebook, Google.
- **How**?
- Identity federation is enabled by requesting temporary security credentials from the AWS **Security Token Service** (STS). These credentials can be used to log into the AWS Management Console or to make AWS API requests.
- Temporary security credentials consist of a short-lived access key ID, a secret access key, and a session token. As with any AWS API request, federated users can sign the requests using the access key ID and secret access key; however, federated users must also pass the session token. There is no limit on the number of temporary security credentials that can be issued.

Manage IAM Roles

- IAM roles allow us to delegate access to users or services.
- IAM users or AWS services can have a role to obtain **temporary security credentials** that can be used to make **AWS API calls and/or for permanent purposes**.

Scenarios

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.

Cross-account access

To control or manage access to resources, such as isolating a development environment from a production environment, we might have multiple AWS accounts.

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. **Using an IAM role can simplify this.**

Manage Users

Manage IAM users and their access—we can create users in IAM, assign them individual security credentials (such as access keys, passwords, and multi-factor authentication devices) or request temporary security credentials to provide users access to AWS services and resources.

we can manage permissions in order to control which operations a user can perform.

IAM users can be:

- Privileged administrators who need console access to manage our AWS resources.
- End users who need access to content in AWS.
- Systems/SW that need privileges to programmatically access our data in AWS.

General Use Cases for Creating IAM Users

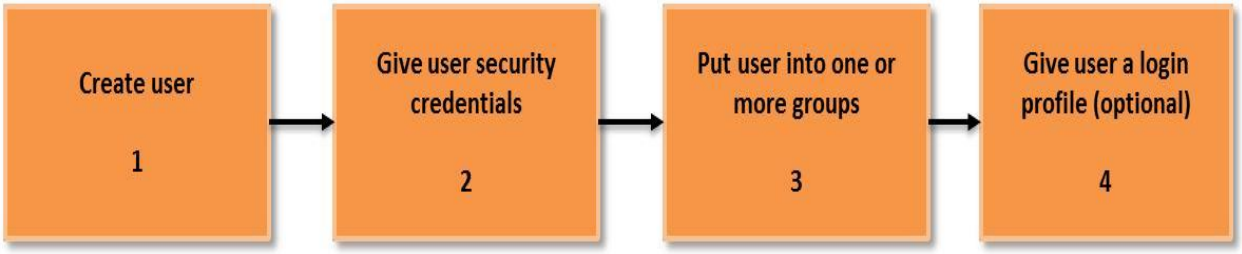
- It is security best practice to *NOT* use root account. The root account grants access to all services and resources, and users can perform any action on those services.
- we have to have other people in our group who have varied access and authorization permissions. IAM users make it easy to assign policies to specific users accessing specific services and associated resources.
- If we want to use the AWS CLI, we will need an IAM user.
- If we want to use a role, we will need an IAM user.
- If we want to federate access to the IAM console or to an AWS service API, we will need an IAM user.
- If we want to leverage web identity federation, we will need an IAM user.

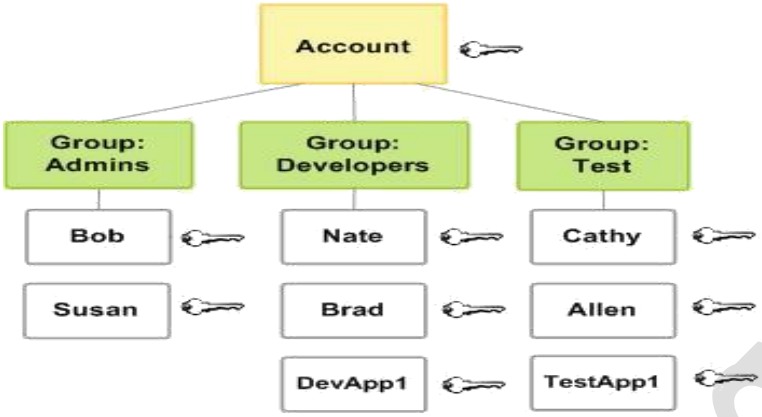
IAM Users

- The **"**identity" aspect of AWS Identity and Access Management (IAM) helps us with the question "Who is that user?", often referred to as *AUTHENTICATION*. Instead of sharing our root account credentials with others, we can create individual IAM users within our account that correspond to users in our organization.
- **IAM users are not separate accounts; they are users within our account**. Each user can have its own **password** for access to the AWS Management Console so that the user can make programmatic requests to work with resources in our account.
- In the following figure, the users Brad, Jim, DevApp1, DevApp2, TestApp1, and TestApp2 have been added to a single AWS account. Each user has its own credentials

Using Groups for Easy Administration

- A group is a collection of IAM users.
- Groups let us assign permissions to a collection of users, which can make it easier to manage the permissions for those users.
- **For example, in almost all the companies, we 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 our organization and should have administrator privileges, we can assign the appropriate permissions by adding the user to that group. Similarly, if a person changes jobs in our organization, instead of editing that user's permissions, we can remove him or her from the old group and add him or her to the new group.**





Accessing IAM

we can work with AWS Identity and Access Management in any of the following ways.

AWS Management Console

The console is a browser-based interface to manage IAM and AWS resources

AWS Command Line Tools

we can use the AWS command line tools to issue commands at our system's command line to perform IAM and AWS tasks;

AWS SDKs

AWS provides SDKs (software development kits) that consist of libraries and sample code for various programming languages and platforms (Java, Python, Ruby, .NET, iOS, Android, etc.). The SDKs provide a convenient way to create programmatic access to IAM and AWS, mostly in development areas.

IAM HTTPS API

we can access IAM and AWS programmatically by using the IAM HTTPS API, which lets us issue HTTPS requests directly to the service, mostly in browser cases.