

AWS IAM (Identity and Access Management) and MFA (Multi-Factor Authentication)

IAM (Identity and Access Management)

What is IAM?

IAM (Identity and Access Management) is a **global AWS service** that allows you to **securely manage access** to AWS services and resources for users, groups, and roles.

With IAM, you can:

- Create users and groups
- Assign permissions through **policies**
- Manage **roles** for service-to-service communication
- Enforce **least privilege access**

Types of Users in AWS

1. **Root User** – The account owner with **full access** to all AWS services.
2. **IAM User** – A user created by the root or admin, with **limited or managed permissions**.

Note: The root user should only be used for initial setup and critical tasks like billing or account closure. Use IAM users for daily operations.

IAM Concepts

Policy

A **policy** is a JSON document that defines **permissions** (allow/deny) for an entity to access AWS resources.

Types of Policies

Type	Description
AWS Managed Policy	Predefined by AWS, maintained by AWS.
Customer Managed	Created and managed by the user (more flexible).
Inline Policy	Embedded directly within a user, group, or role for granular access .

Root User vs IAM Admin User

Capability	Root User	IAM Admin User
Delete AWS Account	✓	✗
Access All Services	✓	✓ (with policy)
Enable/Disable MFA	✓	✓

Hands-On: IAM Configuration

Step 1: Create an IAM User

1. Open the **AWS Management Console**.
2. Go to **IAM → Users → Add users**.
3. Enter **User name**.
4. Select **Access type**:
 - AWS Management Console access
 - Programmatic access (for CLI/API access)

The screenshot shows the AWS IAM console. In the top right, there's a green success message: "User created successfully. You can view and download the user's password and email instructions for signing in to the AWS Management Console." Below this, the "Users (1)" section displays a table with one row for "rohan". The table columns include "User name", "Path", "Group", "Last activity", "MFA", "Password age", "Console last sign-in", and "Access key ID". The "rohan" entry has a green "Now" icon under "Last activity" and a green "C" icon under "Console last sign-in".

Screenshot 1: IAM User Creation – Select Access Type

Step 2: Create a Group

1. Navigate to **IAM → User groups → Create group**.
2. Enter a **group name** (e.g., AWS_Group).
3. Attach a **policy** (e.g., AmazonS3FullAccess).

The screenshot shows the AWS IAM console. In the top right, there's a green success message: "AWS_Group user group created." Below this, the "User groups (1/1)" section displays a table with one row for "AWS_Group". The table columns include "Group name", "Users", "Permissions", and "Creation time". The "AWS_Group" entry has a green "Defined" icon under "Permissions" and a green "Now" icon under "Creation time".

Screenshot 2: Create IAM Group and Attach AdministratorAccess Policy

Step 3: Add User to Group

1. Go to the created **user's settings**.
2. Choose **Add user to group**.
3. Select the previously created group.

The screenshot shows the AWS IAM User Groups page. A green banner at the top indicates '1 user added to this group.' The main section displays the 'AWS_Group' info with a summary table showing the user group name, creation time, and ARN. Below this, there are tabs for 'Users' (1), 'Permissions', and 'Access Advisor'. The 'Users' tab lists 'Users in this group (1)' with 'roshani' as the only entry. There are buttons for 'Remove' and 'Add users'.

Screenshot 3: Add IAM User to Created Group

Step 4: Attach Policy Directly (Optional)

1. Go to IAM → Users → [Your User].
2. Click on the **Permissions** tab.
3. Click **Add permissions** → **Attach policies directly**.
4. Choose a policy like AmazonS3ReadOnlyAccess.

The screenshot shows the AWS IAM Users page. A green banner at the top indicates '1 policy added'. The main section displays the 'rohan' user info with a summary table showing ARN, console access status, and access key details. Below this, there are tabs for 'Permissions', 'Groups', 'Tags', 'Security credentials', and 'Last Accessed'. The 'Permissions' tab shows 'Permissions policies (1)' with a table listing the policy name and type. The policy listed is 'AmazonS3ReadOnlyAccess' (AWS managed).

Screenshot 4: Attach AmazonS3ReadOnlyAccess Policy to User

Step 5: Add Inline Policy for Granular Control

1. Go to IAM → Users → [User] → **Add Inline Policy**.
2. Use the JSON editor to define custom permissions.
Example:
Allow EC2 to view and launch instances but deny deletion.

Policy s3-inline-policy created.

Summary

ARN: arn:aws:iam::396608790002:user/rohan
Created: July 08, 2025, 00:01 (UTC+05:30)
Console access: Enabled without MFA
Last console sign-in: Never
Access key 1: Create access key

Permissions | Groups | Tags | Security credentials | Last Accessed

Permissions policies (2)
Permissions are defined by policies attached to the user directly or through groups.

Policy name	Type	Attached via
EC2-inline-policy	Customer inline	Inline
s3-inline-policy	Customer inline	Inline

Screenshot 5: Add Inline Policy Using JSON Editor

Example Use Case of IAM Role

You want an EC2 instance to access an S3 bucket:

1. Create a **Role** with AmazonS3ReadOnlyAccess.
2. Attach the role to the EC2 instance.
3. The EC2 instance now inherits the role's permissions.

role_for_ec2_to_s3fullaccess

Summary

Creation date: July 08, 2025, 00:43 (UTC+05:30)
Last activity: -
ARN: arn:aws:iam::396608790002:role/role_for_ec2_to_s3fullaccess
Maximum session duration: 1 hour
Instance profile ARN: arn:aws:iam::396608790002:instance-profile/role_for_ec2_to_s3fullaccess

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (1)
You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonS3FullAccess	AWS managed	2

Permissions boundary (not set)

Screenshot 6: Create IAM Role with AmazonS3ReadOnlyAccess

EC2

Instances (1/2)
Last updated: less than a minute ago

Actions | Launch instances

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
instance1	i-0787ba3af0283db57	Running	t2.micro	2/2 checks passed	View alarms +	eu-west-3c	ec2-13-3-
instance2	i-0faf243a87dd1c51	Running	t2.micro	2/2 checks passed	View alarms +	eu-west-3c	ec2-35-1-

Screenshot 7: Attach IAM Role to EC2 Instance

Multi-Factor Authentication (MFA)

What is MFA?

MFA (Multi-Factor Authentication) is a **security enhancement** that requires two forms of verification:

1. Your account **password**
2. A **temporary one-time passcode (OTP)** generated by an authenticator app

It helps protect accounts from **unauthorized access** even if the password is compromised.

How MFA Works

When logging into the AWS Console:

1. Enter **username and password**
2. Enter **OTP from your device** (e.g., Google Authenticator)

Even if someone steals your password, they cannot log in without your second device (MFA code).

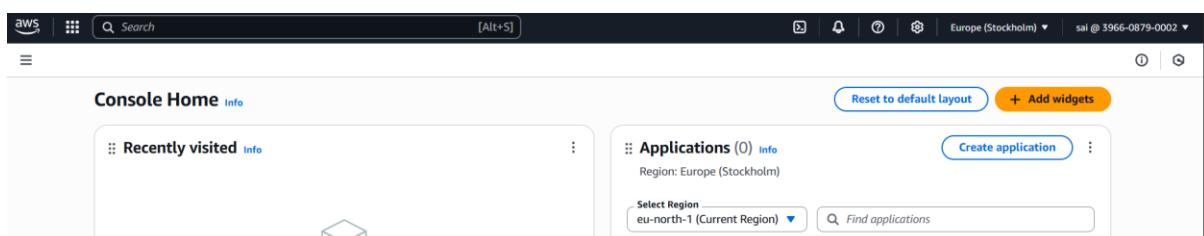
Supported MFA Apps

- Google Authenticator
- Authy
- Duo Mobile
- Microsoft Authenticator

Hands-On: Enable MFA for IAM User

Step 1: Log in as the IAM User

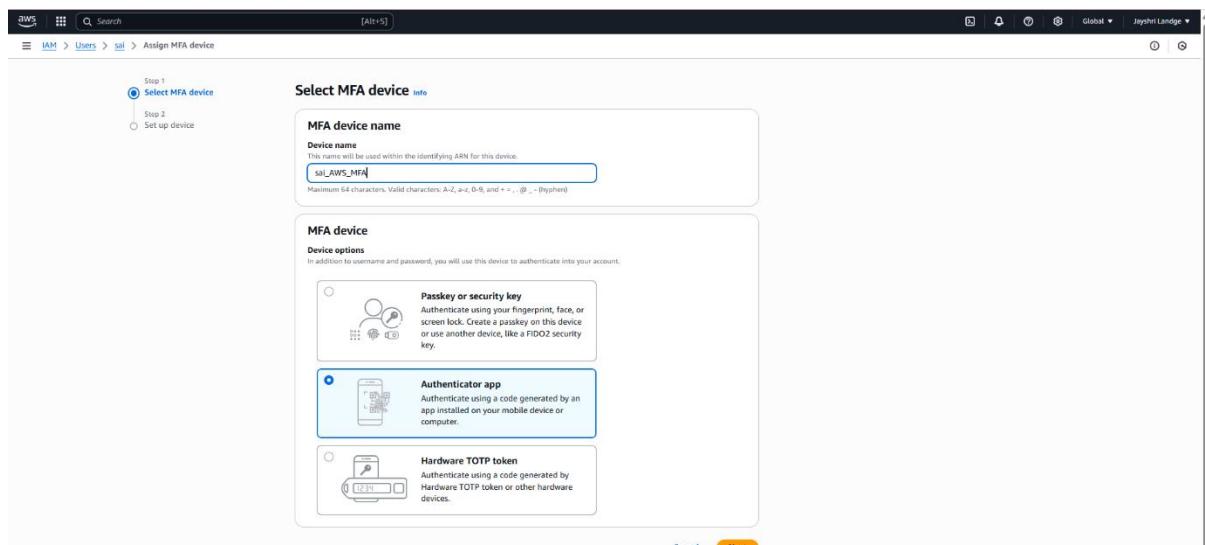
Use the credentials sent during IAM user creation.



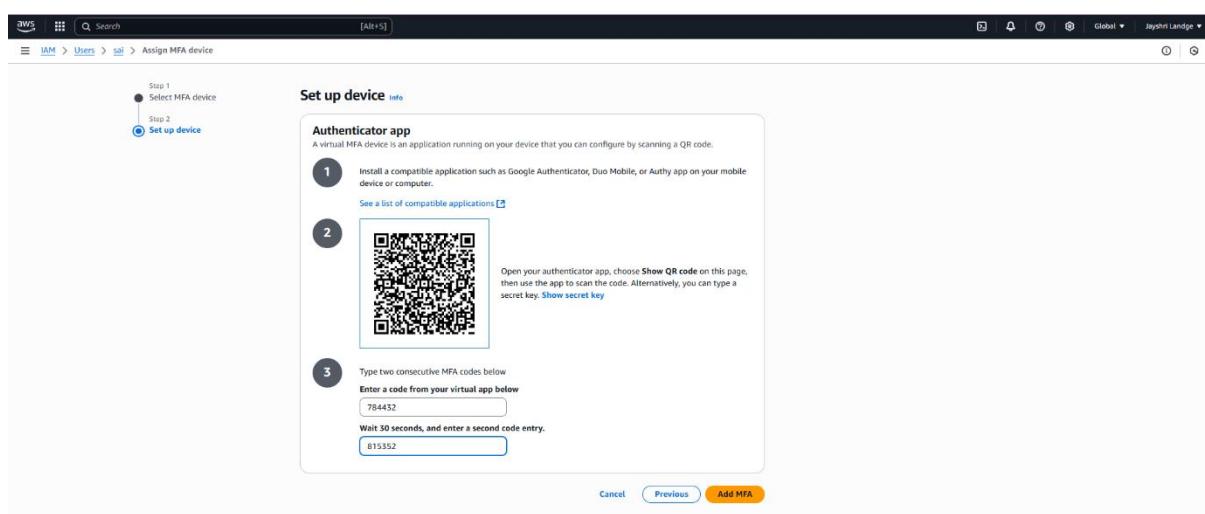
Screenshot 8: IAM User Console Login

Step 2: Set Up MFA

1. Go to **My Security Credentials** in the top right.
2. Choose **Multi-Factor Authentication (MFA)**.
3. Select **Assign MFA device**.
4. Choose **Authenticator App**.
5. Scan the QR code using Google Authenticator or any supported app.
6. Enter the 2 consecutive OTPs shown.



Screenshot 9: Navigate to My Security Credentials for MFA Setup



Screenshot 10: Assign MFA Device and Scan QR Code

The screenshot shows the AWS IAM Security Credentials page for a user named 'sa1'. A green banner at the top indicates that MFA device assignment is available. Below the banner, there is a summary card for the MFA device. The device has the ARN 'arn:aws:iam::396608790002:user/sa1' and was created on July 07, 2025, at 20:33 UTC+05:30. It has 'Console access Enabled with MFA' and 'Last console sign-in Today'. There is one 'Access key' listed. The 'Security credentials' tab is selected, showing a 'Console sign-in' section with a link to the AWS console sign-in page and a 'Multi-factor authentication (MFA) (1)' section where a virtual MFA device is assigned.

Screenshot 11: Enter OTP Codes to Complete MFA Setup

Step 3: Verify and Activate

After entering valid codes, MFA is enabled successfully for that IAM user.

The screenshot shows the AWS sign-in page. A message at the top says, 'You are currently using the improved sign in UI experience. The improved sign in experience will launch soon. During this time, you can still change back to legacy sign in using the dropdown in the upper right corner.' The main sign-in form is displayed, asking for an MFA code. The code '146027' is entered in the field. Below the form is an advertisement for 'Amazon Lightsail' with the tagline 'Lightsail is the easiest way to get started on AWS' and a cartoon character giving a thumbs up.

Screenshot 12: MFA Successfully Enabled for IAM User

Conclusion

- IAM is essential for controlling **who can access** AWS resources and **what actions** they can perform.
- IAM policies (Managed, Inline, and Customer Managed) offer **flexibility** in access control.
- IAM roles are used for **service-to-service access** (e.g., EC2 to S3).
- MFA significantly **enhances account security** by introducing an additional layer of authentication.

Always follow the **principle of least privilege** and **enable MFA** for both root and IAM users.