

# AWS IAM - Identity and Access Management

- Concepts
- Sub-topics
- Hands-on Practical

## What is AWS IAM?

AWS IAM (Identity and Access Management) is a service that allows us to manage who can access AWS resources and what actions they can perform.

### IAM helps in:

- Creating users
- Assigning permissions
- Securing AWS accounts

📍 IAM is a global service (not region-specific).

## Why IAM is Important?

- Prevents unauthorized access
- Controls access to AWS resources
- Ensures security & compliance
- Required for every AWS service

📍 Without IAM, AWS usage is insecure.

## IAM Basics

### ◆ Authentication vs Authorization

Authentication → Who are you? (Login)

Authorization → What can you do? (Permissions)

## ◆ Root User vs IAM User

Root User	IAM User
Full access	Limited access
Used only for setup	Used for daily work
High risk	Safer

📌 Never use root account daily.

## 2 ways to access AWS

### 1.Console access (GUI) ---> Email/Password (root user)

Username /Password (IAM user)

### 2.Programmatical access (CLI) ---> CLI, SDK's

1.we need to install AWSCLI software in Linux and windows (CMD).

2.we need to install SDKs for java, .NET, python etc.).

- Authentication can be done on programmatical access using keys (Access key and secret key).
- “aws configure” is used to configure AWSCLI on windows cmd and on Linux.
- Keys are user specific, individual IAM user have their own keys.
- It is not recommended to share the keys to anyone.
- Every IAM user can have max 2 set of keys.
- Once the keys are lost it is lost you can't get the same keys back. But we can regenerate then you will get the new keys.

## IAM Identities

### 🔒 IAM Users

An IAM user represents a person or application that interacts with AWS.

### Use cases:

- Admin
- Developer
- Tester

## Best Practices:

- One user per person
- Enable MFA
- Avoid access keys unless required

## IAM Groups

A group is a collection of IAM users.

## Why use groups?

- Easier permission management
  - Common access for many users
1. Group under group (nested group) are not possible.
  2. It is possible to attach multiple policies to the IAM user and IAM group.
  3. You can add or remove policies to the user and groups anytime.

## IAM Roles

IAM roles provide temporary permissions.

## Used by:

- EC2
- Lambda
- Other AWS services
- Cross-account access

 Roles do not have passwords.

- If you configure IAM roles, you no need to configure keys on machines.
- 1 EC2 instance can have only 1 role attached at a time. But 1 role can be attached to multiple EC2 instances at the same time.
- 2 AWS services will not communicate with each other by default, if you want them to communicate use IAM role.
- ex: - If EC2 wants to access S3 then we create the role and give permissions. Trusted entity = EC2 (To whom you are attaching role) Permissions = S3 (What kind of permissions the role should have).

## Switch Role

Switch Role allows an IAM user to temporarily access another AWS account or role without creating a separate user in that account.

It uses IAM Roles + Trust Policy to provide temporary permissions.

## Why Switch Role Is Used

- To access multiple AWS accounts securely
- To avoid creating multiple IAM users
- To follow least privilege and security best practices

## How Switch Role Works (Simple Flow)

IAM User → Switch Role → Temporary Permissions → Access Resources

- No passwords are shared
- Temporary credentials are used
- Access expires automatically

## Steps to Use Switch Role

- 1.Create IAM Role in target account.
- 2.Add trust policy to allow source account.
- 3.Login to source account.
- 4.Click Switch Role.
- 5.Enter account ID & role name.

## Key Benefits

- Better security
- No long-term credentials
- Easy multi-account management

Switch Role allows users to temporarily assume permissions of another role or AWS account without creating separate IAM users.

## IAM Policies

### ◆ What is a Policy?

A policy is a JSON document that defines permissions. AWS has policy editor/policy generator these will help to generate JSON code automatically.

### ◆ Policy Elements

- Version – policy language version

- Effect – Allow or Deny
- Action – AWS actions
- Resource – AWS resource
- Condition – optional rules

#### ◆ Policy Structure

```
{  
  "Version": "2012-10-17",  
  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "s3>ListBucket",  
      "Resource": "*"  
    }  
  ]  
}
```

#### ◆ Types of Policies

- AWS Managed Policies
- Customer Managed Policies
- Inline Policies

## IAM Roles for AWS Services

#### ◆ Why Services Need Roles?

Services like EC2 cannot store credentials securely.

#### IAM roles:

- Provide temporary access
- Improve security

#### ◆ EC2 Role Example

- Attach S3 access role

- EC2 can access S3 without access keys

## IAM Permissions & Security

### ◆ Least Privilege Principle

Grant only required permissions, nothing extra.

### ◆ Password Policy

- Minimum length
- Complexity
- Rotation

### ◆ MFA (Multi-Factor Authentication)

Adds an extra layer of security using OTP.

📌 Mandatory for root user.

## IAM Tags

IAM Tags are key-value pairs that help you organize, identify, and manage IAM resources such as users, roles, and policies.

Key: Environment

Value: Dev

### Why IAM Tags Are Important?

- Helps organize IAM resources
- Enables better access control
- Useful for automation and cost tracking
- Simplifies management in large AWS environments

### Where IAM Tags Are Used?

IAM tags can be added to:

- IAM Users
- IAM Roles
- IAM Policies

## IAM Tags in Access Control

Tags can be used in IAM policy conditions to control access.

Example use case: Allow access only to users with Department = Dev

## Identity Provider (IdP)

An Identity Provider allows users to sign in to AWS using external identities instead of creating IAM users.

### Examples:

- Active Directory
- Google
- Azure AD

📌 Used for Single Sign-On (SSO) in enterprises.

Identity providers allow federated access to AWS using external user identities.

## IAM Credential Report

The IAM Credential Report provides details about:

- IAM users
- Password usage
- Access keys status
- MFA status

📌 Used for security auditing.

### Why important:

Helps identify inactive users and unused credentials.

## IAM Access Advisor

IAM Access Advisor shows:

- Which services a user or role has to access
- When those services were last accessed

- 📌 Helps apply least privilege.

## Use case:

Remove unused permissions safely.

## IAM Access Analyzer

IAM Access Analyzer identifies:

- Resources that are accessible from outside your AWS account
- Over-permissive policies

- 📌 Improves security posture.

Commonly used with: S3, IAM roles, KMS

## AWS Identity Center (SSO)

AWS Identity Center (formerly AWS SSO) provides:

- Centralized access management
- Single sign-on to multiple AWS accounts
- Integration with external identity providers

- 📌 Used in multi-account AWS environments.

## Common IAM Mistakes:

- Giving admin access to everyone
- Not enabling MFA
- Hardcoding access keys
- Ignoring policy restrictions

## Real-World IAM Scenarios

### ◆ Scenario 1: EC2 Accessing S3 Securely

**Problem:** EC2 needs access to S3 without credentials

**Solution:** Attach IAM Role with S3 permissions to EC2

## ◆ Scenario 2: Developer Read-Only Access

**Problem:** Developer should only view S3 objects

**Solution:** Attach read-only S3 policy to IAM group

## ◆ Scenario 3: Restrict Console Access

**Problem:** Allow CLI access but block console login

**Solution:** Policy condition to deny console actions

## ◆ Scenario 4: Enforce MFA

**Problem:** User should access AWS only with MFA

**Solution:** Add MFA condition in IAM policy

## ◆ Scenario 5: Multiple Users, Same Permissions

**Problem:** 10 users need same access

**Solution:** Create IAM group and attach policy once

## ◆ Scenario 6: Cross-Team Access

**Problem:** Dev team needs limited Prod access

**Solution:** Use IAM role with trust policy

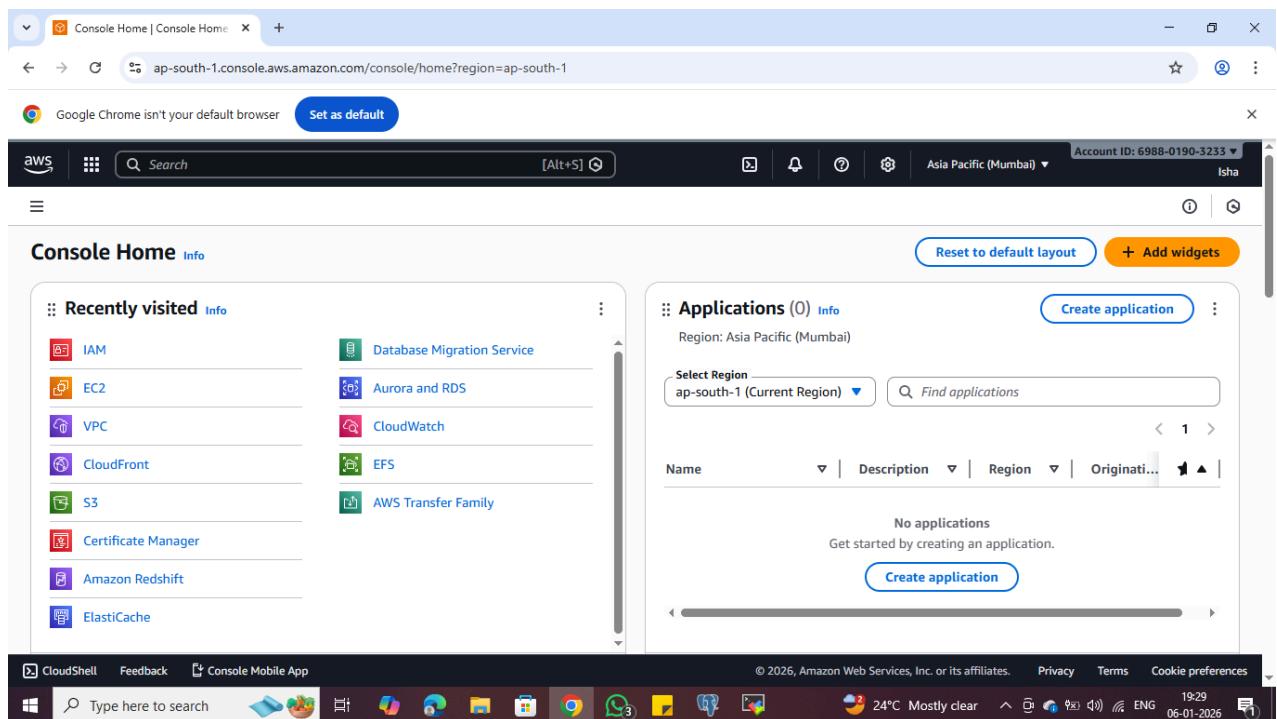
# IAM Practical Work

For AWS console we have two accounts. One is root account and next one is IAM account. Always login with IAM account with username and password and then enter the MFA (I already said that how to enable MFA for accounts in previous posts) of IAM account to enter into console. Use only IAM account for daily use.

## Practical 1: Create IAM User

### Steps:

Go to AWS console. Here is the console...



Now search for the IAM service...

The screenshot shows the AWS IAM Dashboard. On the left, there's a sidebar with 'Identity and Access Management (IAM)' and a search bar. The main area has a 'Security recommendations' section with three items: 'Root user has MFA', 'You have MFA', and 'Your user, Isha, does not have any active access keys that have been unused for more than a year'. To the right is an 'AWS Account' summary with the Account ID (698801903233), Account Alias (Create), and Sign-in URL (https://698801903233.sigin.aws.amazon.com/console). Below that is a 'Quick Links' section with 'My security credentials' and 'Manage your access keys, multi-factor'. At the bottom, there's a navigation bar with CloudShell, Feedback, and Console Mobile App.

Go to users then select create user. Enter username and enable console access

The screenshot shows the 'Specify user details' step of the IAM User creation wizard. It's Step 1 of 4. The user name is set to 'raju'. There are two checked options: 'Provide user access to the AWS Management Console - optional' and 'Autogenerated password'. A note says, 'You can view the password after you create the user.' Below that is a 'Console password' field with 'Custom password' selected. A note says, 'Enter a custom password for the user.' There are also notes about password complexity and a checkbox for users to create their own password. At the bottom, there's a note about generating programmatic access and a 'Learn more' link. The 'Next' button is visible at the bottom right.

Then attach policy (Admin Access).

**Set permissions**

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

**Permissions options**

Add user to group  
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

Copy permissions  
Copy all group memberships, attached managed policies, and inline policies from an existing user.

Attach policies directly  
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

**Permissions policies (1/1440)**

Choose one or more policies to attach to your new user.

Policy name	Type	Attached entities
AdministratorAccess	AWS managed - job function	1
AdministratorAccess-Amplify	AWS managed	0
AdministratorAccess-AWSElasti...	AWS managed	0
AdministratorAccess-AWSIncide...	AWS managed	0
AdministratorAccessPolicy	AWS managed	0

[Create policy](#)

**Review and create**

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

**User details**

User name: raju

Console password type: Autogenerated

Require password reset: No

**Permissions summary**

Name	Type	Used as	Permissions policy
AdministratorAccess	AWS managed - job function		

**Tags - optional**

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#) [Previous](#) [Create user](#)

Then click on create user.

The screenshot shows the AWS IAM 'Users' page. A green success message box at the top states 'User created successfully' with a link to 'View user'. Below it, the 'Users' section shows a table with two entries:

User name	Path	Group	Last activity	MFA	Password age	Console last sign-in
Isha	/	1	1 hour ago	Virtual...	63 days	1 hour ago
raju	/	0	-	-	Now	-

The left sidebar includes sections for 'Access management' (User groups, Roles, Policies, Identity providers, Account settings, Root access management, Temporary delegation requests) and 'Access reports' (CloudShell, Feedback, Console Mobile App). The bottom navigation bar includes links for CloudShell, Feedback, and Console Mobile App, along with system status (21°C, Mostly clear), language (ENG), and date (06-01-2026).

User created successfully.

## Practical 2: Create IAM Group

Go to IAM then select user groups then click create group. After that enter group name and add the user to the group.

The screenshot shows the 'Create user group' page. The 'Name the group' section has 'User group name' set to 'AWS'. The 'Add users to the group - Optional (1/2)' section shows a table with two users:

User name	Group	Last activity	Creation time
Isha	1	2 hours ago	2 months ago
raju	0	-	29 minutes ago

The left sidebar includes sections for 'Access management' (User groups, Roles, Policies, Identity providers, Account settings, Root access management, Temporary delegation requests) and 'Access reports' (CloudShell, Feedback, Console Mobile App). The bottom navigation bar includes links for CloudShell, Feedback, and Console Mobile App, along with system status (Gold +0.63%, 21:28, 06-01-2026).

Then attach the policies. Ex: I enable admin access. U can add any policies what u want.

The screenshot shows the 'Create user group' wizard in the AWS IAM console. The current step is 'Attach permissions policies - Optional (1/1109)'. A table lists available policies, with 'AdministratorAccess' selected. Other policies shown include AccountManagementF..., AdministratorAccess-A..., AdministratorAccess-A..., AIOpsAssistantInciden..., and AIOpsAssistantPolicy.

Policy name	Type	Used as	Description
AccountManagementF...	AWS managed	None	For use with accounts cr...
<b>AdministratorAccess</b>	AWS managed - job function	Permissions policy (2)	Provides full access to A...
AdministratorAccess-A...	AWS managed	None	Grants account adminis...
AdministratorAccess-A...	AWS managed	None	Grants account adminis...
AIOpsAssistantInciden...	AWS managed	None	Provides permissions re...
AIOpsAssistantPolicy	AWS managed	None	Provides ReadOnly perm...

After that click create user group. Now group created successfully.

The screenshot shows the 'User groups' page in the AWS IAM console. A success message 'AWS user group created.' is displayed. The table lists two user groups: 'Administrators' and 'AWS', both created 'Now' with 'Defined' permissions.

Group name	Users	Permissions	Creation time
Administrators	1	Defined	2 months ago
AWS	1	Defined	Now

### Practical 3: Create Policy without Json

Go to IAM then select policies then click create policy.

Step 1  
Specify permissions

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

Select a service

Service

+ Add more permissions

Cancel Next

Now I want to create policy for S3 read access. We can create an S3 Read-Only custom policy using **visual editor**. Make sure visual editor tab is selected (default). Under Service, search S3. Select S3

Step 1  
Specify permissions

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

S3

Set permissions for S3

+ Add more permissions

Cancel Next

Now expand s3 under that

Expand List

- ListAllMyBuckets
- ListBucket

Expand Read

- GetObject

- GetBucketLocation

Do not select Write or Delete actions.

The screenshot shows the AWS IAM 'Create policy' interface. In the 'Actions' section, under the 'List' heading, several actions are listed with checkboxes. The 'ListBuckets' action is checked, while others like 'ListAllMyBuckets' and 'ListAllMyBuckets' are checked. The 'Effect' dropdown is set to 'Allow'. The status bar at the bottom indicates 'CloudShell Feedback Console Mobile App' and a date/time of '06-01-2026'.

The screenshot shows the AWS IAM 'Create policy' interface. In the 'Actions' section, under the 'Read' heading, many actions are listed with checkboxes. The 'ListTagsForResource' action is checked, along with numerous other actions such as 'GetBucketLocation', 'GetObject', and 'GetObjectTorrent'. The status bar at the bottom indicates 'CloudShell Feedback Console Mobile App' and a date/time of '06-01-2026'.

Now choose resources. Under resources select all. Then click next then set the name of the policy then click create policy.

The screenshot shows the 'Review and create' step of the AWS IAM policy creation wizard. It includes fields for 'Policy name' (S3-ReadOnly-Visual), 'Description - optional', and a summary of permissions defined in the policy. The navigation bar at the top shows 'AWS > IAM > Policies > Create policy'. The bottom status bar indicates the URL is us-east-1.console.aws.amazon.com/iam/home?region=ap-south-1#/policies/create.

visual policy created successfully.

Now attach the policy to the user or group. Go to Users. Select your user then open Permissions.

The screenshot shows the 'Permissions' tab for the user 'raju'. It displays a summary of ARN, console access (Enabled without MFA), and access keys. Below this, the 'Permissions policies (1)' section shows a single policy named 'AdministratorAccess' attached directly via AWS managed - job function. The navigation bar at the top shows 'AWS > IAM > Users > raju'. The bottom status bar indicates the URL is https://us-east-1.console.aws.amazon.com/iam/home?region=ap-south-1#/users/details/raju/add-permissions.

Click Add permissions. Select Attach policies.

The screenshot shows the 'Add permissions' step 2 review screen. At the top, there are two tabs: 'Add user to group' (selected) and 'Review'. Below the tabs, the heading 'Permissions options' is displayed. Three options are shown: 'Add user to group' (radio button), 'Copy permissions' (radio button), and 'Attach policies directly' (radio button, selected). The 'Attach policies directly' option is described as attaching a managed policy directly to a user. Below this, the heading 'Permissions policies (1/1440)' is shown, followed by a table with one row. The table has columns for 'Name' (S3-ReadOnly-Visual), 'Type' (Customer managed), and 'Used as' (Permissions policy). A search bar at the top of the table filters results for 's3-Read'. The bottom right of the screen has 'Cancel' and 'Next' buttons.

Search S3-ReadOnly-Visual then click next. Now Select add permissions.

The screenshot shows the 'Review' step screen. At the top, there are two tabs: 'Add permissions' (selected) and 'Review'. Below the tabs, the heading 'Review' is displayed. It states that the following policies will be attached to the user. The 'User details' section shows the user name 'raju'. The 'Permissions summary' section shows one policy: 'S3-ReadOnly-Visual' (Customer managed, Used as Permissions policy). The bottom right of the screen has 'Cancel', 'Previous', and 'Add permissions' buttons.

The screenshot shows the AWS IAM User Details page for a user named 'raju'. The 'Permissions' tab is active, displaying three attached policies: 'AdministratorAccess', 'S3-ReadOnly-Custom', and 'S3-ReadOnly-Visual'. The 'AdministratorAccess' policy is highlighted. The left sidebar shows navigation options like 'Dashboard', 'Access management', and 'Access reports'. The bottom of the screen shows the Windows taskbar with various pinned icons.

policy will be attached successfully

## Practical 4: Create Custom Policy with Json

Go to IAM then select policies then click create policy. select JSON tab. You will see a default JSON template. Delete the existing content

The screenshot shows the 'Create policy' page in the AWS IAM console. The 'JSON' tab is selected in the 'Policy editor'. The current JSON content is a placeholder: "Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor." A button labeled '+ Add new statement' is visible. The left sidebar shows 'Step 1: Specify permissions' and 'Step 2: Review and create'. The bottom of the screen shows the Windows taskbar with various pinned icons.

Now I want to create policy for S3 read access. We can create an S3 Read-Only custom policy using JSON template. For the code, AWS provides an official tool called the IAM Policy Generator, which lets you SELECT options and AUTO-GENERATE JSON.

Open the Policy Generator Link in browser

👉 <https://awspolicygen.s3.amazonaws.com/policygen.html>

## Select Policy Type: IAM Policy

The AWS Policy Generator is a tool that 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](#).

**Step 1: Select policy type**  
A Policy is a container for permissions. The different types of policies you can create are an [IAM Policy](#), an [S3 Bucket Policy](#), an [SNS Topic Policy](#), a [VPC Endpoint Policy](#), and an [SQS Queue Policy](#).

**Type of Policy**  
IAM Policy

**Step 2: Add statement(s)**  
A statement is the formal description of a single permission. See [a description of elements](#) that you can use in statements.

**Effect**  
 Allow  
 Deny

**AWS**  
 All Services ("")  
 Amazon S3

## Select AWS Service: Amazon S3

Select Actions → select:

- ✓ ListAllMyBuckets
- ✓ ListBucket
- ✓ GetObject

Specify Resource → Amazon Resource Name (ARN): For practice → \*

**Effect**  
 Allow  
 Deny

**AWS**  
 All Services ("")  
 Amazon S3

**Actions**  
 All Actions ("")  
--Select Actions--  
ListAllMyBuckets X ListBucket X GetObject X

**Amazon Resource Name (ARN)**  
 All Resources ("")  
\* ARN should follow the following format: arn:aws:s3:::\${BucketName}/\${KeyName}. Use a comma to separate multiple values.

► Add conditions (optional)  
[Add Statement](#)

## Click Add Statement

The screenshot shows the AWS Policy Generator interface. At the top, there are tabs for 'Create policy | IAM | Global' and 'AWS Policy Generator'. The main content area has a heading 'Statements added (1)'. Below it, a table lists one statement:

Effect	Action	Resource(s)	Condition(s)	Remove
Allow	s3>ListAllMyBuckets s3>ListBucket s3GetObject	*	None	Remove

### Step 3: Generate policy

A policy is a document (written in the [Access Policy Language](#)) that acts as a container for one or more statements.

**Generate Policy**

This AWS Policy Generator is provided for informational purposes only, you are still responsible for your use of Amazon Web Services technologies and ensuring that your use is in compliance with all applicable terms and conditions. This AWS Policy Generator is provided **as is** without warranty of any kind, whether express, implied, or statutory. This AWS Policy Generator does not modify the applicable terms and conditions governing your use of Amazon Web Services technologies.

and then click the generate policy

The screenshot shows the AWS Policy Generator interface with a modal window titled 'Policy JSON Document'. The modal contains the generated JSON policy code:

```
1 * [ "Version": "2012-10-17", 2 * "Statement": [ 3 * { 4 *   "Sid": "Statement1", 5 *   "Effect": "Allow", 6 *   "Action": [ 7 *     "s3>ListAllMyBuckets", 8 *     "s3>ListBucket", 9 *     "s3GetObject" 10 *   ], 11 *   "Resource": "*" 12 * } 13 * ] 14 * ] 15 *
```

Below the code, there is a note: 'Click below to edit. To save the policy, copy the text below to a text editor. Changes made below will **not be reflected in the policy generator tool**.', a 'Close' button, and a 'Copy Policy' button. The status bar at the bottom shows system information: 19°C, Mostly clear, ENG, 23:05, 06-01-2026.

copy the policy and paste it in the JSON tab

ListBucket", "s3:GetObject" ], "Resource": "\*" } ] }."/>

Step 1  
Specify permissions

Step 2  
Review and create

Specify permissions Info

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Sid": "Statement1",  
6       "Effect": "Allow",  
7       "Action": [  
8         "s3>ListAllMyBuckets",  
9         "s3>ListBucket",  
10        "s3GetObject"  
11      ],  
12      "Resource": "*"  
13    }  
14  ]  
15 }
```

Visual JSON Actions ▾

Edit statement

Select a statement

Select an existing statement in the policy or add a new statement.

+ Add new statement

CloudShell Feedback Console Mobile App

Type here to search

Earnings upcoming

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

23:14 06-01-2026

then next and set the policy name and click create policy.

Step 1  
Specify permissions

Step 2  
Review and create

Review and create Info

Review the permissions, specify details, and tags.

Policy details

Policy name

Enter a meaningful name to identify this policy.

S3-ReadOnly-Custom

Description - optional

Add a short explanation for this policy.

Permissions defined in this policy Info

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it.

Search

CloudShell Feedback Console Mobile App

Type here to search

Earnings upcoming

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

23:16 06-01-2026

Now custom policy will be created successfully.

same as visual policy we can add this policy to the user or group. Now this time I add this policy to the group.

Go to IAM select your group. Open Permissions tab. Click Add permissions.

Screenshot of the AWS IAM User Groups page. The 'Permissions' tab is selected. A table shows the following policy:

Policy name	Type	Attached entities
AdministratorAccess	AWS managed - job function	3

Choose Attach policies. Search S3-ReadOnly-Custom. Select the policy. Click Attach policies.

Screenshot of the 'Add Permissions' page for the 'AWS' user group. The 'Attach policies' section is displayed, showing a search bar with 's3-Re' and a list of policies:

Policy name	Type	Used as
<input checked="" type="checkbox"/> S3-ReadOnly-Custom	Customer managed	None
<input type="checkbox"/> S3-ReadOnly-Visual	Customer managed	Permissions policy (1)

Buttons at the bottom right include 'Cancel' and 'Attach policies'.

The screenshot shows the AWS IAM User Groups page. The 'Permissions' tab is selected. It displays a table of policies attached to the user group 'AWS'. The table includes columns for Policy name, Type, and Attached entities. Two policies are listed:

Policy name	Type	Attached entities
AdministratorAccess	AWS managed - job function	3
S3-ReadOnly-Custom	Customer managed	1

policy will be attached successfully.

IAM policies can be created using either the JSON editor or the visual policy editor provided by AWS.

## Practical 5: Create IAM Role for EC2

Role is used for communication between two services or two accounts. If we create a role and give permissions, then both can access each other.

Go to IAM then select roles then click create role

The screenshot shows the AWS IAM Roles page. A table lists nine existing roles, each with a 'Delete' button and a 'Create role' button. The table has columns for Role name, Trusted entities, and Last activity. The roles listed are:

Role name	Trusted entities	Last activity
AWSServiceRoleForAmazonElasticFileSystem	AWS Service: elasticfilesystem (Service-Linked Role)	33 days ago
AWSServiceRoleForElastiCache	AWS Service: elasticache (Service-Linked Role)	24 hours ago
AWSServiceRoleForRDS	AWS Service: rds (Service-Linked Role)	25 minutes ago
AWSServiceRoleForResourceExplorer	AWS Service: resource-explorer-2 (Service-Linked Role)	29 minutes ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-
dms-cloudwatch-logs-role	AWS Service: dms	-
dms-vpc-role	AWS Service: dms	2 days ago
rds-monitoring-role	AWS Service: monitoring.rds	-

The screenshot shows the 'Create role' wizard in the AWS IAM console. The current step is 'Step 1: Select trusted entity'. On the left, there's a navigation sidebar with three steps: 'Select trusted entity' (selected), 'Add permissions', and 'Name, review, and create'. The main content area is titled 'Select trusted entity' and contains a 'Trusted entity type' section. It lists five options: 'AWS service' (selected, highlighted in blue), 'AWS account', 'Web identity', 'SAML 2.0 federation', and 'Custom trust policy'. Below this is a 'Use case' section with a sub-section 'Service or use case' containing a dropdown menu with 'Choose a service or use case'.

now we have to decide which one have to select to use

if we have to create a role for two services to communicate or access each other then we go with first one or else if we have to create a role for another account to access our account then we go with second one.

For ex, Ec2 wants to access s3 service now I'm selecting AWS service

then we have to mention trusted entity and permissions

**Trusted entity** - which one have to attach

**Permissions** - to whom we are attach

The screenshot shows the 'Create role' wizard in the AWS IAM console, specifically the 'Add permissions' step. The main content area is titled 'Use case' and contains a 'Service or use case' section with a dropdown menu showing 'EC2'. Below this is a 'Use case' section with a list of options for EC2:

- EC2 - Allows EC2 instances to call AWS services on your behalf.
- EC2 Role for AWS Systems Manager - Allows EC2 Instances to call AWS services like CloudWatch and Systems Manager on your behalf.
- EC2 Spot Fleet Role - Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.
- EC2 - Spot Fleet Auto Scaling - Allows Auto Scaling to access and update EC2 spot fleets on your behalf.
- EC2 - Spot Fleet Tagging - Allows EC2 to launch spot instances and attach tags to the launched instances on your behalf.
- EC2 - Spot Instances - Allows EC2 Spot Instances to launch and manage spot instances on your behalf.
- EC2 - Spot Fleet - Allows EC2 Spot Fleet to launch and manage spot fleet instances on your behalf.

The screenshot shows the 'Add permissions' step of the IAM role creation wizard. A search bar at the top right is set to 's3'. Below it, a table lists 16 AWS managed policies. The 'AmazonS3FullAccess' policy is selected and highlighted with a blue border.

Policy name	Type	Description
AmazonDMSRedshiftS3Role	AWS managed	Provides access to manage S3 settings ...
<b>AmazonS3FullAccess</b>	AWS managed	Provides full access to all buckets via t...
AmazonS3ObjectLambdaExecutionRolePolicy	AWS managed	Provides AWS Lambda functions permis...
AmazonS3OutpostsFullAccess	AWS managed	Provides full access to Amazon S3 on ...
AmazonS3OutpostsReadOnlyAccess	AWS managed	Provides read only access to Amazon S...
AmazonS3ReadOnlyAccess	AWS managed	Provides read only access to all bucket...
AmazonS3TablesFullAccess	AWS managed	Provides full access to all S3 table buc...

Then set the role name and click create role. Then role is created successfully.

The screenshot shows the 'Name, review, and create' step of the IAM role creation wizard. The 'Role name' field is filled with 'firstrole'. The 'Description' field contains the text 'Allows EC2 Instances to call AWS services on your behalf.'

Then we have to attach this role to Ec2 to access the s3 service.

How I attach this role to Ec2 process I explained later after explanation about Ec2 service.

These are the simple basic practical's about IAM. I provide some tasks to do yourself to practice the IAM service.

## TASKS:

Login to the ROOT account --> Do the Tour first on Console Home --> Navigate to IAM service

1. Create Alias, bookmark it IAM user sign-in Link.

2. Create your own IAM user with admin policy attached

3. Setup MFA for the IAM user that you have created(yourownname)

4. Setup MFA for Root account (Security Credentials)

Note: Your Google Authenticator should have only 2 accounts. Don't Delete the APP.

\*\*\*\* Sign-out from root account and login as IAM user that you have just created. From now onwards always login with IAM user with that bookmarked IAM sign link

5. Create another sample IAM user(testuser1)

6. Attach/add and detach/remove policies for sample IAM user testuser1(not your main IAM user)

--> attach 11 managed policies to the IAM user and see max how many you can attach to IAM user and GROUP also

7. Create 2 or 3 groups and attach users to the groups. Remove users from the group, Remove Permissions from Group

8. Try to disable console access to the test IAM user and try to login with that disabled IAM user(see what message you get)

9. Reset the IAM user password and try to login with the rested password

10. Create Access key and Secret Keys for the test IAM user

11. Make the Keys De-active.

12. Delete the KEYS.

13. See Groups Access Advisor and IAM User Last Accessed

14. Delete the sample IAM Users not yours (don't delete your own IAM user)

15. Delete Groups

16. Sign-out IAM user

1: Create a role and assign permissions (3 scenarios - EC2, Lambda, Switch Role )

2. Create a Inline policy

3. Create 2 IAM Users(test1 and test2)

--> Test1 assign IAMFULLPERMISSIONS

--> TEST2 assign IAMREADONLYACCESS

Logout and login with test1 and verify IAM permissions

Logout and login with test2 and verify IAM permissions

4. Create IAM Users (test3)

This user should be able to only create users, list users, get users, Get Groups, create group, list group, update user, update group

--> Attach this Policy test3 user

Logout and login with test3 and verify permissions what test3 can do or cannot do

Switch Roles (account id): Create a test IAM User in your friends account using switch role (while creating role, attach IAM permissions)

Create Analyzer (External) and delete it

The main point for Console is after completing the everyday practical we have to sign-out from the IAM account. Next day if we have any work, then we have to sign in again through

IAM account....