



# Cloud Security with AWS IAM



Anas Oubassou





# Introducing Today's Project!

In this project, I focused on acquiring hands-on cloud computing skills through practical implementation of AWS access management principles.

## Tools and concepts

Services used: EC2 (launching instances), IAM (users/groups/policies)

Key concepts learned: Least-privilege access control, resource tagging for permissions, policy structure (Effect/Action/Resource), secure access management

## Project reflection

This project required approximately 1 hour to complete, divided across several phases:

- EC2 setup (~15 mins)
- IAM policy/user creation (~20 mins)
- Testing access configurations (~10 mins)
- Account alias implementation (~5 mins)



# Tags

Tags are metadata labels attached to AWS resources that serve multiple important functions in cloud resource management. For this project, I created an "Env" tag to distinguish between production and development EC2 instances.

Tags provide several benefits for resource management: they organize resources into logical groups, enable efficient filtering when viewing resources, support detailed cost allocation tracking, and—most relevant to this project—allow for targeted policy application based on environment type. I assigned the "Env" tag to my two EC2 instances, designating one with the value "production" and the other with "development" to establish distinct resource environments with different security requirements.

### Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

▼ Name and tags Info

Key Info

Q Name X

Value Info

Q network-dev-Out X

Resource types Info

Select resource types ▼

Remove

Instances X

Key Info

Q env X

Value Info

Q development X

Resource types Info

Select resource types ▼

Remove

Instances X

Add new tag

You can add up to 48 more tags.



# IAM Policies

IAM policies define permissions that control access to AWS resources. They function as precise rule sets specifying who can perform what actions on which resources, ensuring secure and controlled access. In this project, this capability allowed me to grant an intern access exclusively to the development EC2 instance while restricting access to production environments.

## The policy I set up

For this project, I implemented a policy using JSON format that establishes clear security boundaries. The policy configuration:

- Allows all EC2 actions on instances tagged "Env=development"
- Permits describing (viewing) all EC2 resources for inventory purposes
- Explicitly denies creating or deleting tags on any resource to prevent permission escalation

When creating a JSON policy, three core elements must be defined:

**Effect:** Specifies whether the policy allows or denies the specified actions (either "Allow" or "Deny")

**Action:** Lists the AWS actions that are permitted or restricted (e.g., "ec2:\*" for all EC2 actions)

**Resource:** Defines which AWS resources the policy applies to, often using Amazon Resource Names (ARNs)

## When creating a JSON policy, you have to define its Effect, Action and Resource.

The Effect, Action, and Resource attributes of a JSON policy means...  
Effect: Specifies if the policy allows or denies actions ("Allow" or "Deny"). Action: Lists the AWS actions permitted or restricted (e.g., "ec2:" for all EC2 actions). Resource: Defines the AWS resources the policy applies to.



# My JSON Policy

The screenshot shows the AWS IAM console's 'Create policy' page. The left pane displays a JSON policy document with the following content:

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Action": "ec2:*",  
7       "Resource": "*",  
8       "Condition": {  
9         "StringEquals": {  
10          "ec2:ResourceTag/Env": "development"  
11        }  
12      }  
13    },  
14    {  
15      "Effect": "Allow",  
16      "Action": "ec2:Describe*",  
17      "Resource": "*",  
18    },  
19    {  
20      "Effect": "Deny",  
21      "Action": [  
22        "ec2:DeleteTags",  
23        "ec2:CreateTags"  
24      ],  
25      "Resource": "*"   
26    }  
27  ]  
28 }
```

The right pane, titled 'Edit statement', contains the text 'Select a statement' and a button labeled '+ Add new statement'.




## Account Alias

An AWS account alias is a user-friendly name that replaces the standard 12-digit AWS account ID in the login URL. This customization makes it considerably easier for users like the intern to access the AWS Management Console without needing to remember numeric account identifiers.

Creating an account alias was a straightforward process taking just a few minutes to implement. With this change, the new AWS console sign-in URL became <https://an-asawsprojects.signin.aws.amazon.com/console>, providing a more intuitive access point for team members.

### AWS Account


#### Account ID

 867344474403

#### Account Alias

an-asawsprojects [Edit](#) | [Delete](#)

#### Sign-in URL for IAM users in this account

 <https://an-asawsprojects.signin.aws.amazon.com/console>



# IAM Users and User Groups

## Users

IAM users represent individual accounts created for people or services requiring AWS access. Each IAM user has unique credentials and permission sets, ensuring secure and controlled resource access without sharing account credentials. This approach follows security best practices by maintaining individual accountability for all actions.

## User Groups

IAM groups bundle users to manage permissions collectively. Assign policies once to a group, and all members inherit them—ideal for teams like interns needing same access.

Attaching the policy to the user group grants all interns the same permissions: EC2 access only for "development" instances (not production) while blocking tag modifications.

# Logging in as an IAM User

IAM groups serve as containers that bundle users together to manage permissions collectively. By assigning policies once at the group level, all member users automatically inherit those permissions—an efficient approach for teams like interns requiring identical access profiles.

Attaching the custom policy to the intern user group ensures all interns receive consistent permissions: EC2 access limited to "development" instances only (not production resources) while preventing any tag modifications that could potentially circumvent these restrictions.



Console sign-in details

Email sign-in instructions

Console sign-in URL  
https://an-asawsprojects.signin.aws.amazon.com/console

User name  
Networks-dev-OubassouANas

Console password  
\*\*\*\*\* Show

## Testing IAM Policies

As the intern IAM user:

- Production instance: Failed to stop (received "Access Denied" error)
- Development instance: Successfully stopped (permitted via policy)

These test results validate that the policy correctly restricts access to only development resources, enforcing the intended security boundaries.





## Stopping the production instance

When attempting to stop the production instance while logged in as the intern IAM user, AWS denied the action with an "Access Denied" error message. This confirmation demonstrates that the policy is working as designed—the user only has permissions for the development instance (tagged Env=development), not production resources.





# Testing IAM Policies

## Stopping the development instance

Success! The stop action completed because the policy explicitly allows full EC2 control for resources tagged Env=development. This successful test confirms that our tagged-based permission strategy works correctly, allowing appropriate access while maintaining security boundaries

