

# Cloud Security with AWS IAM

---

## Project Overview

This project demonstrates the implementation of AWS Identity and Access Management (IAM) to control access to cloud resources. I successfully configured EC2 instances with tag-based access control, created custom IAM policies, established user groups, and tested security permissions to ensure proper access restrictions.

**Project Duration:** Approximately 40 minutes

**Difficulty Level:** Easy

**AWS Region Used:** eu-west-3 (Paris)

---

## Table of Contents

- [What I Built](#)
  - [Technologies & Concepts](#)
  - [Step-by-Step Implementation](#)
  - [Key Learnings](#)
  - [Conclusion](#)
- 

## What I Built

A complete IAM security infrastructure demonstrating:

- Two EC2 instances with environment-based tagging (production and development)
- Custom IAM policy with conditional access controls
- IAM user group for managing intern permissions
- IAM user with restricted access based on resource tags
- AWS account alias for simplified login

This project simulates a real-world scenario where an intern needs access to development resources while being restricted from production environments.

---

## Technologies & Concepts

### AWS Services Used

- **Amazon EC2 (Elastic Compute Cloud)** - Virtual servers for hosting applications
- **AWS IAM (Identity and Access Management)** - Authentication and authorization service
- **Resource Tagging** - Metadata labels for organizing and controlling access to resources

### Key Concepts Learned

1. **IAM Policies** - JSON-based permission documents that define what actions are allowed or denied
2. **IAM Users** - Individual identities representing people or applications

3. **IAM User Groups** - Collections of users with shared permissions
4. **Tag-Based Access Control** - Using resource tags to conditionally grant or deny permissions
5. **Account Aliases** - Custom, memorable names for AWS account sign-in URLs
6. **Policy Effects** - Understanding "Allow" and "Deny" statements (Deny always takes precedence)
7. **Conditions** - Adding logic to policies to make permissions context-aware

## Step-by-Step Implementation

### Step 1: Launch EC2 Instances

#### What I did:

I launched two Amazon EC2 instances to simulate production and development environments for the NextWork company.

#### Instance Configuration:

- **Instance Type:** t2.micro (Free tier eligible)
- **AMI:** Amazon Linux 2023 (64-bit x86)
- **Key Pair:** Proceeded without a key pair (console access only)
- **Network Settings:** Default VPC with auto-assign public IP enabled
- **Security Group:** New security group with SSH access from anywhere
- **Storage:** 8 GiB gp3 volume

**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	Value   Info	Resource types   Info
<input type="text" value="Name"/> <span>X</span>	<input type="text" value="nextwork-dev-youhad"/> <span>X</span>	Select resource types <span>▼</span> <span>Remove</span>
<span>Instances X</span>		
Key   Info	Value   Info	Resource types   Info
<input type="text" value="env"/> <span>X</span>	<input type="text" value="production"/> <span>X</span>	Select resource types <span>▼</span> <span>Remove</span>
<span>Instances X</span>		
<span>Add new tag</span>		

You can add up to 48 more tags.

**Application and OS Images (Amazon Machine Image)** Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Recents
Quick Start

Amazon Linux 
Ubuntu 
Windows 
Red Hat 
SUSE Linux 
Debian 

[Browse more AMIs](#)  
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI  
ami-078abd88811000d7e (64-bit (x86), uefi-preferred) / ami-0387f15c965e9e817 (64-bit (Arm), uefi)  
Virtualization: hvm ENA enabled: true Root device type: ebs

**Summary**

Number of instances | [Info](#)

1

**Software Image (AMI)**  
Amazon Linux 2023 AMI 2023.9.2... [read more](#)  
ami-078abd88811000d7e

**Virtual server type (instance type)**  
t2.micro

**Firewall (security group)**  
New security group

**Storage (volumes)**  
1 volume(s) - 8 GiB

ⓘ Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where t2.micro isn't available) when used with free tier AMIs, 750 hours per month of public IPv4 address usage, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GB of bandwidth to the internet. Data transfer charges are not included as part of the free tier allowance.

Cancel Launch instance Preview code

Activ  
Go to !

#### Tags Applied:

##### 1. Production Instance:

2 / 16

**Name and tags**

Key   Info	Value   Info	Resource types   Info
<input type="text" value="Name"/> <span>X</span>	<input type="text" value="nextwork-dev-youhad"/> <span>X</span>	Select resource types <span>▼</span> <span>Remove</span>
<span>Instances X</span>		
Key   Info	Value   Info	Resource types   Info
<input type="text" value="env"/> <span>X</span>	<input type="text" value="production"/> <span>X</span>	Select resource types <span>▼</span> <span>Remove</span>
<span>Instances X</span>		
<span>Add new tag</span>		

You can add up to 48 more tags.

## 1. Development Instance:

**Name and tags**

Key   Info	Value   Info	Resource types   Info
<input type="text" value="Name"/> <span>X</span>	<input type="text" value="nextwork-dev-youhad"/> <span>X</span>	Select resource types <span>▼</span> <span>Remove</span>
<span>Instances X</span>		
Key   Info	Value   Info	Resource types   Info
<input type="text" value="env"/> <span>X</span>	<input type="text" value="development"/> <span>X</span>	Select resource types <span>▼</span> <span>Remove</span>
<span>Instances X</span>		
<span>Add new tag</span>		

You can add up to 48 more tags.

## Why tags matter:

Tags are key-value pairs that help organize AWS resources. More importantly, they can be used in IAM policies to create conditional access controls. In this project, the `env` tag determines whether a user can perform actions on an instance.

**Instances (2) Info**

Last updated <span>less than a minute ago</span>		<span>Connect</span>	<span>Instance state ▾</span>	<span>Actions ▾</span>	<span>Launch instances ▾</span>
<span>Find Instance by attribute or tag (case-sensitive)</span>		All states <span>▼</span>			
<input type="checkbox"/>	Name <span>▼</span>	Instance ID	Instance state <span>▼</span>	Instance type <span>▼</span>	Status check
<input type="checkbox"/>	nextwork-prod-youhad	i-094b7492e81fc4f82	<span>Running</span> <span>Q</span> <span>Q</span>	t2.micro	<span>2/2 checks passed</span> <span>View alarms +</span>
<input type="checkbox"/>	nextwork-dev-youhad	i-031e18d7547f7980e	<span>Running</span> <span>Q</span> <span>Q</span>	t2.micro	<span>Initializing</span> <span>View alarms +</span>

**Result:** Both instances successfully launched and are running in the eu-west-3 region.

## Step 2: Create an IAM Policy

### What I did:

I created a custom IAM policy called `NextWorkDevEnvironmentPolicy` that grants specific permissions based on resource tags.

The screenshot shows the AWS IAM 'Specify permissions' step. It displays a JSON editor with the following policy 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 }
  
```

Below the editor, there's a button '+ Add new statement'.

## Breaking Down the Policy:

### Statement 1 - Conditional Full Access:

```
{
  "Effect": "Allow",
  "Action": "ec2:*",
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "ec2:ResourceTag/Env": "development"
    }
  }
}
```

- Allows **all EC2 actions** (`ec2:*`)
- Only on resources tagged with `Env=development`
- This means users can start, stop, modify, and manage development instances

### Statement 2 - Read-Only Access:

```
{
  "Effect": "Allow",
  "Action": "ec2:Describe*",
  "Resource": "*"
}
```

- Allows all **Describe** actions (read-only operations)
- On **all resources** (no conditions)
- Users can view all instances but can only modify development ones

### Statement 3 - Explicit Deny:

```
{
  "Effect": "Deny",
  "Action": ["ec2:DeleteTags", "ec2:CreateTags"],
  "Resource": "*"
}
```

- **Denies** tag modification actions
- On **all resources**
- Prevents users from changing tags to bypass access controls
- **Remember:** Deny always overrides Allow

## Policy Details:

**Review and create** Info

Review the permissions, specify details, and tags.

**Policy details**

**Policy name**  
Enter a meaningful name to identify this policy.  
**NextWorkDevEnvironmentPolicy**

**Description - optional**  
Add a short explanation for this policy.  
IAM Policy for NextWork's development environment

**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.

Service	Access level	Resource	Request condition
EC2	Full: Tagging	All resources	None

Service	Access level	Resource	Request condition
EC2	Full: List, Permissions management, Read, Write	All resources	ec2:ResourceTag/Env = development

**Add tags - optional** Info

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

**Add new tag**  
You can add up to 50 more tags.

Activate Windows  
Go to Settings to activate Windows.  
Cancel Previous Create policy

## Key Understanding:

This policy demonstrates the **principle of least privilege** - users get exactly the permissions they need, nothing more. Interns can fully manage development resources but only view production resources, and they cannot manipulate tags to escalate their privileges.

## Step 3: Create an AWS Account Alias

### What I did:

I created a custom account alias to make the IAM sign-in URL more user-friendly.

### Account Details:

## AWS Account

### Account ID

 [Redacted]

### Account Alias

youhad-aws [Edit](#) | [Delete](#)

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

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

- **Account ID:** [Redacted for security]
- **Account Alias:** youhad-aws
- **Sign-in URL for IAM users:** <https://youhad-aws.signin.aws.amazon.com/console>

### What is an Account Alias?

An account alias is a custom, memorable name that replaces your 12-digit AWS account ID in the sign-in URL. Instead of sharing a complex URL like:

`https://123456789012.signin.aws.amazon.com/console`

Users can access:

`https://youhad-aws.signin.aws.amazon.com/console`

### Benefits:

- Easier to remember and share
- More professional appearance
- Reduces login errors
- One alias per AWS account (must be globally unique)

---

## Step 4: Create IAM User Group

### What I did:

I created an IAM user group called `nextwork-dev-group` to manage permissions for all NextWork interns collectively.

**Add users to the group - Optional (1) Info**

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

User name	Last activity	Creation time
Ayoub	22 minutes ago	8 months ago

**Attach permissions policies - Optional (1/1113) Info**

You can attach up to 10 policies to this user group. All the users in this group will have permissions that are defined in the selected policies.

Filter by Type: Customer managed | 4 matches

Policy name	Type	Used as	Description
AWSLambdaBasicExecutionRole	Customer managed	Permissions policy (1)	-
MyIAMPolicy	Customer managed	None	-
NextWorkDevEnvironmentPolicy	Customer managed	None	IAM Policy for NextWork's development environment
s3cr_for_youhad-s3-origin-bu...	Customer managed	Permissions policy (1)	-

Activate Windows Cancel Create user group  
Go to Settings to activate Windows.

## User Group Configuration:

- **Group Name:** nextwork-dev-group
- **Attached Policy:** NextWorkDevEnvironmentPolicy
- **Users:** Initially 0 (user will be added in next step)

## What are IAM User Groups?

IAM user groups are collections of IAM users that share the same set of permissions. Instead of attaching policies to individual users, you attach them to groups, making permission management much more efficient.

## Why use groups?

Imagine managing 50 interns individually - if you need to update permissions, you'd have to modify 50 separate users. With groups:

1. Create one group
2. Attach policies to the group
3. Add all interns to the group
4. Any policy changes automatically apply to all members

**Best Practice:** Always use groups for permission management, even for a single user. This makes future scaling easier and follows AWS security best practices.

---

## Step 5: Create IAM User

### What I did:

I created an IAM user for a new NextWork intern with console access and added them to the development user group.

### User Configuration:

## User details

### User name

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = . @ \_ - (hyphen)

### Provide user access to the AWS Management Console - *optional*

In addition to console access, users with SignInLocalDevelopmentAccess permissions can use the same console credentials for programmatic access without the need for access keys.

### Console password

#### Autogenerated password

You can view the password after you create the user.

#### Custom password

Enter a custom password for the user.

- Must be at least 8 characters long
- Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols ! @ # \$ % ^ & \* ( ) \_ + - (hyphen) = [ ] { } | '

Show password

### Users must create a new password at next sign-in - Recommended

Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

 If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

## Permissions Setup:

### 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.

#### User groups (1/2)

User groups (1/2)			
<span>Search</span> <span></span> <span><a href="#">Create group</a></span> <span></span> <span></span> <span></span>			
	Group name	Users	Attached policies
<input type="checkbox"/>	Admin	1	AdministratorAccess 2025-04-12 (8 months ago)
<input checked="" type="checkbox"/>	nextwork-dev-group	0	NextWorkDevEnvironmentPolicy 2025-12-18 (13 minutes ago)

#### ► Set permissions boundary - *optional*

Activate Windows  
[Cancel](#) [Previous](#) [Next](#)  
 Go to Settings to activate Windows

## Why add to a group?

Rather than attaching the policy directly to the user, adding them to a group is a best practice because:

- Easier to manage permissions for multiple users
- Consistent permissions across team members
- Simpler to audit and modify access
- Follows the principle of role-based access control (RBAC)

## Review Summary:

## 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 nextwork-dev-youhad	Console password type Autogenerated	Require password reset No
----------------------------------	--	------------------------------

**Permissions summary**

Name ↗	Type	Used as
<a href="#">nextwork-dev-group</a>	Group	Permissions group

**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.

## Console Sign-in Details:

### Retrieve password

You can view and download the user's password below or email users instructions for signing in to the AWS Management Console. This is the only time you can view and download this password.

**Console sign-in details**

Console sign-in URL <a href="https://youhad-aws.signin.aws.amazon.com/console">https://youhad-aws.signin.aws.amazon.com/console</a>	<a href="#">Email sign-in instructions ↗</a>
User name <a href="#">nextwork-dev-youhad</a>	
Console password <a href="#">***** Show</a>	

[Cancel](#) [Download .csv file](#) [Return to users list](#)

## Important Security Note:

This is the **only time** you can view or download the autogenerated password. In production environments, you should:

- Download the CSV file with credentials
- Send credentials through a secure channel (not email)
- Require password reset on first login
- Enable multi-factor authentication (MFA)

**Result:** The IAM user is now created and ready to test our policy restrictions!

---

## Step 6: Testing IAM Policy - Failed Stop Production Instance

### What I did:

I logged in as the IAM user **nextwork-dev-youhad** and attempted to stop the production instance to verify that the policy correctly denies access.

### Test Scenario:

- **Instance:** **nextwork-prod-youhad** (tagged with **env=production**)
- **Action:** Stop instance

- **Instance ID:** i-094b7492e81fc4f82

**Stop instance** X

Stopping your instance allows you to reduce costs, modify settings, and troubleshoot problems.

Instance ID	Stop protection	Result
<input type="checkbox"/> i-094b7492e81fc4f82 (nextwork-prod-youhad) ↗	Disabled	<input checked="" type="checkbox"/> Can stop

**Associated resources**  
You will continue to incur charges for these resources while the instance is stopped

**⚠ You will be billed for associated resources**  
After you stop the instance, you are no longer charged usage or data transfer fees for it. However, you will still be billed for associated Elastic IP addresses and EBS volumes.

**Skip OS shutdown**  
This option skips the graceful OS shutdown process. Use only when your instance must be stopped immediately, such as during an emergency or failover.  
 Skip OS shutdown

Cancel
Stop

## Result: Access Denied!

**Failed to stop the instance i-094b7492e81fc4f82**  Diagnose with Amazon Q X

You are not authorized to perform this operation. User: arn:aws:iam::056481036163:user/nextwork-dev-youhad is not authorized to perform: ec2:StopInstances on resource: arn:aws:ec2:eu-west-3:056481036163:instance/i-094b7492e81fc4f82 because no identity-based policy allows the ec2:StopInstances action. Encoded authorization failure message:  
 G142lKErnJuSrkvcb21cpk\_rK1pyofsLOFewPOn8CMn59CqJNGBhXDm9K9nwPsiwhRGrU9NT163Tmj9Hq7PYPB4RushPYiz7K7rFEDqc9qtO8HkfZO\_2zx7W3oOvqM4rA8\_UAl0ubGV3xQrs2vmfl\_Q-OoFlobFcBKGn-gwh7stccm4BTgJOBDYgpJRzPLkFQ-VpluFwCGH-1CkLsP0tSqWeSkCqOwskhDyVgGyLoEkz1MyVogCFXIOB6MbDzMcpxXhegnj3p4rpGHyJlxGwPOumXAmDCslGba-IMds\_kzD\_vaUkK4dwPN9hYGOQzBNyAzulxbFcML1qayK45LbSEzzZBPEebryJA\_Sq4n4qDFj9Bsgzh0qOsYzQwgUWiKk5bCVMbJvTDBdLp7ExoU8l2qWxqc9radv5w1vE2\_BNGmfmSFHft3km35v6mlU-djPMbGIOQzQUYorQeBLIF4SAgeyewohHgBM6EK876Zr3PFMyVRFOeqAOJ5E5ww-zJ6MKffvLsOOvA6gN-lpnriJFlSJbpCsAvUgX\_w1BB0Uevi9RnphuaUFTKQhL0d784CQCY48knjNRh5ByBkTyAN7h1wQtJ-fgHzvv2Nkd8Gwo8FrvbGny2jdAtVjAw\_dRsOfxQJK9OMMklVuZ1JFTlvrqjQ5kMajNNM-zCr5ogArVwu5ctx24ptZO3pvPEmNpetHJcA3s2mbKW7r4K3aQU-mum\_Bmd9TzsYbttdKpRxJy-PQX-uoUf80snUeFPehZS2zmcHd5V3WpVy06nN-Rje-IAZHpnFyHeoeNDRFIHKy0CsJSG2D\_CfvXL5BmhTJvqe1e72Ud1tFplg86q5QxcvwWJDVPab1UovlpX4NCzYs9JW62ZLJ\_RNnKnKjt9

## Why did this fail?

Looking back at our IAM policy, the first statement only allows EC2 actions when:

```
"Condition": {
  "StringEquals": {
    "ec2:ResourceTag/Env": "development"
  }
}
```

Since the production instance has **Env=production**, the condition doesn't match, and the action is denied. The policy is working correctly!

**This is exactly what we wanted** - interns cannot modify production resources, preventing accidental or unauthorized changes to critical infrastructure.

---

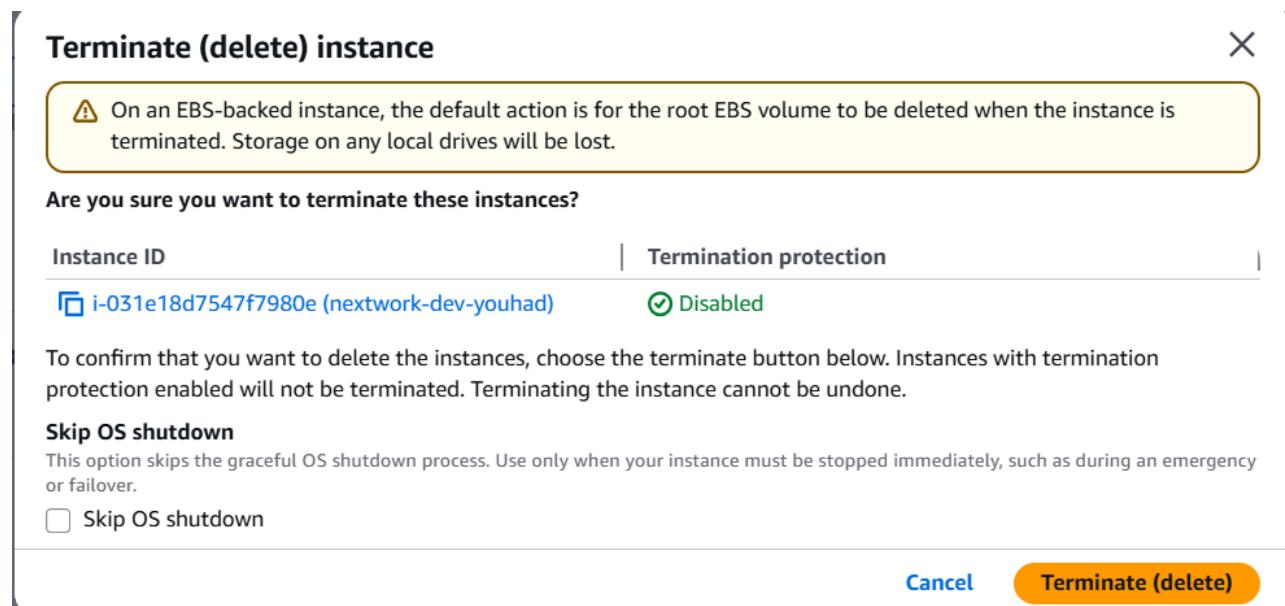
## Step 7: Testing IAM Policy - Successfully Stop Development Instance

## What I did:

Still logged in as the IAM user, I attempted to stop the development instance to verify the policy grants proper access.

## Test Scenario:

- **Instance:** `nextwork-dev-youhad` (tagged with `env=development`)
- **Action:** Stop instance
- **Instance ID:** `i-031e18d7547f7980e`



## Result: Success!

The instance state changed from "Running" to "Stopping", proving that the IAM user has permission to manage development resources.

## Why did this succeed?

The policy's first statement grants full EC2 permissions (`ec2:*`) when the resource tag matches `Env=development`. Since this instance has the correct tag, the user can:

- Stop the instance
- Start the instance
- Reboot the instance
- Modify instance settings
- Perform any other EC2 action

## Key Takeaway:

Tag-based access control allows for flexible, scalable security policies. You can have hundreds of resources, and users automatically get the right permissions based on tags - no need to list specific instance IDs in policies.

## Step 8: Cleanup - Terminate Instances

### What I did:

I switched back to my admin AWS account (with full administrator access) to terminate the production and devolepement instances.

### Why switch accounts?

The IAM user **nextwork-dev-youhad** doesn't have permission to terminate production instances due to our policy restrictions. This demonstrates another layer of security - only administrators with proper credentials can manage production resources.

### Final Instance Status:

Instances (2) <a href="#">Info</a>										Last updated  less than a minute ago	<a href="#">Connect</a>	<a href="#">Instance state ▾</a>	<a href="#">Actions ▾</a>	<a href="#">Launch instances</a>	
										All states					
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4							
<input type="checkbox"/>	nextwork-prod-youhad	i-094b7492e81fc4f82	Terminated	t2.micro	-	<a href="#">View alarms +</a>	eu-west-3a	-							
<input type="checkbox"/>	nextwork-dev-youhad	i-031e18d7547f7980e	Terminated	t2.micro	-	<a href="#">View alarms +</a>	eu-west-3a	-							

Both instances now show "Terminated" status, and AWS will automatically remove them from the console view after a short period.

## Step 10: Cleanup - Delete IAM User

### What I did:

I deleted the IAM user **nextwork-dev-youhad** since the project testing is complete.

Users (1/2) <a href="#">Info</a>											<a href="#">Delete</a>	<a href="#">Create user</a>			
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.															
<input type="checkbox"/>	User name	Path	Group:	Last activity	MFA	Password age	Console last sign-in	Access key ID							
<input type="checkbox"/>	Ayoub	/	1 ..	1 hour ago	Virtual...	250 days	1 hour ago	Active - AKIAQ2JUIG6...							
<input checked="" type="checkbox"/>	nextwork-dev-youhad	/	0	31 minutes ago	-	40 minutes	31 minutes ago	-							

### User Deletion Confirmation:

- User Name:** **nextwork-dev-youhad**
- Groups:** 0 (automatically removed from groups upon deletion)
- Last Activity:** 31 minutes ago
- Password Age:** 40 minutes

The screenshot shows the AWS IAM 'Users' page. At the top, a green banner displays the message: 'User "nextwork-dev-youhad" deleted.' Below this, the heading 'Users (1) [Info](#)' is shown. A note states: 'An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.' A search bar is present. The main table has columns: User name, Path, Group, Last activity, MFA, Password age, Console last sign-in, Access key ID, and Status. One row is listed: 'Ayoub' with a path of '/', last active '1 hour ago', virtual MFA, password age '250 days', signed in '1 hour ago', and status 'Active - AKIAQ2JUIG6...'. Action buttons for 'Edit' (blue), 'Delete' (orange), and 'Create user' (yellow) are at the top right.

## What Happens When You Delete a User?

- All associated credentials (password, access keys) are permanently revoked
- User is automatically removed from all groups
- Any policies directly attached to the user are detached
- The user's sign-in URL becomes invalid
- Action cannot be undone (though you can create a new user with the same name)

## Security Best Practice:

Delete IAM users when:

- Employees leave the company
- Test users are no longer needed
- Credentials may have been compromised
- Following the principle of least privilege and clean account hygiene

## Step 11: Cleanup - Delete IAM Policy

### What I did:

I deleted the custom IAM policy [NextWorkDevEnvironmentPolicy](#) to complete the cleanup process.

The screenshot shows the AWS IAM 'Policies' page. At the top, a green banner displays the message: 'Policy deleted.' Below this, the heading 'Policies (1443) [Info](#)' is shown. A note states: 'A policy is an object in AWS that defines permissions.' A search bar is present. The main table has columns: Policy name, Type, Used as, and Description. The 'Type' dropdown is set to 'Customer managed' and shows '3 matches'. Three rows are listed: 'AWSLambdaBasicExecutionRole-0a976...' (Customer managed, Used as 'Permissions policy (1)'), 'MyIAMPolicy' (Customer managed, Used as 'None'), and 's3crr\_for\_youhad-s3-origin-bucket\_1c8...' (Customer managed, Used as 'Permissions policy (1)'). Action buttons for 'Edit' (blue), 'Actions' (dropdown), 'Delete' (orange), and 'Create policy' (yellow) are at the top right.

**Remaining Policies:** After deletion, only standard AWS-managed policies and other custom policies remain:

- [AWSLambdaBasicExecutionRole](#) (AWS managed)
- [MyIAMPolicy](#) (Custom managed)
- [s3crr\\_for\\_youhad-s3-origin-bucket](#) (Custom managed)

### Important Notes About Deleting Policies:

- You can only delete customer-managed policies (not AWS-managed policies)

- Policy must not be attached to any users, groups, or roles
- Once deleted, the policy cannot be recovered
- All versions of the policy are permanently deleted

## Why Clean Up Policies?

- **Security:** Unused policies can be accidentally attached, granting unintended permissions
  - **Organization:** Keep your IAM console clean and manageable
  - **Compliance:** Audit logs are cleaner when unused policies are removed
  - **Best Practice:** Following the principle of least privilege includes removing unnecessary permission definitions
- 

# Key Learnings

## 1. IAM Policy Structure

IAM policies use JSON format with these key components:

- **Version:** Policy language version (always use "2012-10-17")
- **Statement:** Array of permission statements
- **Effect:** "Allow" or "Deny" (Deny always wins)
- **Action:** AWS service actions (e.g., `ec2:StopInstances`)
- **Resource:** AWS resources the policy applies to
- **Condition:** Optional logic for conditional permissions

## 2. Tag-Based Access Control

Tags are powerful tools for:

- Organizing resources
- Cost allocation and tracking
- Conditional access control in IAM policies
- Automation and scripting

Best practices:

- Use consistent naming conventions (e.g., `Env` vs `Environment`)
- Apply tags consistently across all resources
- Protect tags from unauthorized modification (like we did with the Deny statement)

## 3. Principle of Least Privilege

Grant users only the permissions they need to do their job:

- Development interns need full access to dev resources
- They need read-only access to view production (for learning/context)
- They should not modify production or change tags

## 4. IAM Best Practices Demonstrated

- Use groups instead of attaching policies directly to users
- Use account aliases for user-friendly sign-in URLs
- Test policies thoroughly before production use
- Use explicit deny statements to enforce critical restrictions
- Clean up unused users, groups, and policies
- Use conditions in policies for context-aware permissions

## 5. Security Insights

- **Defense in Depth:** Multiple layers of security (tags + policies + groups)
  - **Fail-Safe Defaults:** Resources are secure by default, permissions must be explicitly granted
  - **Audit Trail:** All IAM actions are logged in CloudTrail
  - **Separation of Duties:** Different users have different permission levels based on roles
- 

## Conclusion

This project successfully demonstrated how to implement secure access control in AWS using IAM. I learned how to:

- Launch and tag EC2 instances for environment-based access control
- Write custom JSON IAM policies with conditions
- Create IAM users and user groups following best practices
- Implement tag-based conditional access control
- Test security policies to verify they work as intended
- Clean up resources properly to maintain account hygiene

### Real-World Applications:

This type of IAM configuration is used by companies of all sizes to:

- Safely onboard interns and contractors with limited access
- Separate development, staging, and production environments
- Prevent accidental deletion or modification of critical resources
- Enable self-service infrastructure management within guardrails
- Maintain security compliance and audit requirements

**Time Investment:** ~40 minutes

**Cost:** \$0 (Free tier eligible)

**Security Value:** Priceless 🔒

---

## Additional Resources

- [AWS IAM Documentation](#)
  - [IAM Best Practices](#)
  - [IAM Policy Simulator](#)
  - [AWS Tagging Strategies](#)
-

**Project Completed:** December 2025

**Author:** YOUHAD AYOUB

**Region:** eu-west-3 (Paris)

**NextWork Challenge:** AWS Beginners Challenge - Project 2

---

*This project was completed as part of the NextWork AWS Beginners Challenge. Special thanks to the NextWork community for their excellent guidance and project structure!*