

# Cloud Security with Amazon IAM using EC2. Also using Terraform as Secret Mission

A

Aiman Haziq





# Introducing today's project!

## What is Amazon IAM ?

Amazon IAM manages access to AWS resources securely by allowing you to create users, groups, and permissions. It ensures granular access control, centralized management, compliance, and cost efficiency.

## How I'm using Amazon IAM in this project

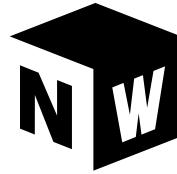
In today's project, Amazon IAM was used to create policies and user groups to manage access to EC2 instances. I set up permissions for different environments and tested user access to ensure correct functionality.

## One thing I didn't expect...

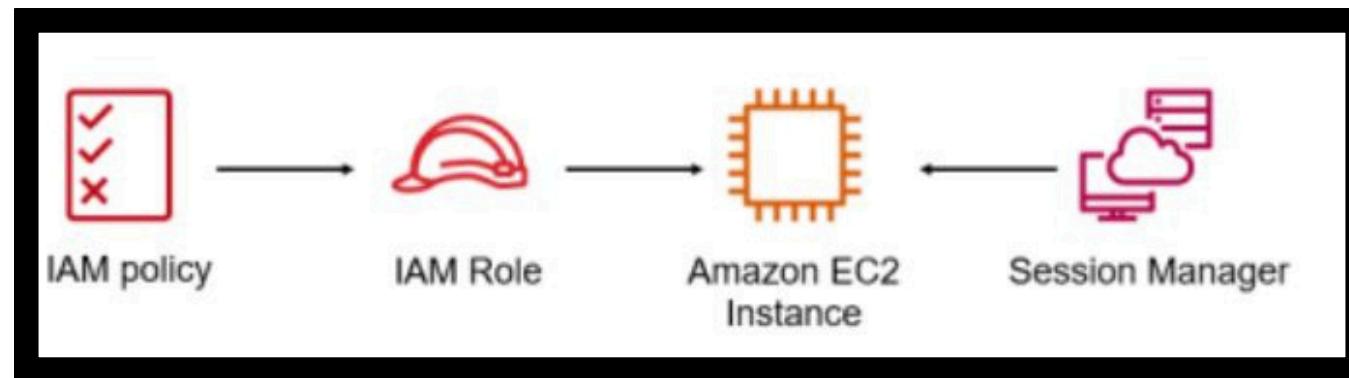
One thing I didn't expect was how useful IAM policies could be in precisely controlling access. The ability to tailor permissions for different environments and test them thoroughly added a layer of security and management I hadn't fully anticipated.

## This project took me...

This project took me around 2-3 hours, including setting up EC2 instances, creating IAM policies, and testing user access. It also included time for taking screenshots and verifying all configurations.



# Cloud Security with Amazon IAM using EC2 using Amazon Management Console





# Tags

Tags are key-value pairs for categorizing AWS resources. Launch EC2 instances with: Name: nextwork-production-aimanhaziq, Tag: Env=production  
Name: nextwork-development-aimanhaziq, Tag: Env=development Create IAM policies and users to manage access.

The tag used on my EC2 instances is called `Env`. The values assigned are: -  
For the production instance: `nextwork-production- aimanhaziq`,  
`Env=production` - For the development instance: `nextwork- development- aimanhaziq`, Env=development'

Instances (1/2) [Info](#)

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
nextwork-pro...	i-0cfb95d8123b24690	Running	t2.micro	2/2 checks passed	View alarms +	ap-southeast-1b	ec2-54-179-147-241.ap...
<b>nextwork-dev...</b>	<b>i-0558abe784cabfea1</b>	<b>Running</b>	<b>t2.micro</b>	-	<b>View alarms +</b>	<b>ap-southeast-1b</b>	<b>ec2-3-0-97-81.ap.south...</b>

**i-0558abe784cabfea1 (nextwork-development-aimanhaziq)**

[Details](#) | [Status and alarms](#) | [Monitoring](#) | [Security](#) | [Networking](#) | [Storage](#) | [Tags](#)

**Tags**

Key	Value
Env	production
Name	nextwork-development-aimanhaziq

[Manage tags](#)



## Step by Step Launch EC2 Instances with Tags

### Objective

Launch EC2 instances for production and development environments and apply appropriate tags.

### Instructions

1. Log in to Amazon Management Console
  - Access the Amazon Management Console using your credentials.
2. Navigate to EC2 Service
  - Select EC2 from the AWS services menu.
3. Switch Region
  - Set the Region to the one closest to your location.
4. Launch EC2 Instances
  - Click Launch instances to start the process.
5. Configure Production Instance
  - Name: nextwork-production-aimanhaziq
  - Tags:
    - Key: Env
    - Value: production
  - AMI: Choose a Free Tier eligible option.
  - Instance Type: Select a Free Tier eligible instance.
  - Key Pair: Choose Proceed without a key pair (not recommended for production).
6. Launch Instance
  - Click Launch instance to complete.
7. Repeat for Development Instance
  - Name: nextwork-development-aimanhaziq
  - Tags: Env = development
8. View Instances
  - Select Instances and refresh to see both instances.
9. Screenshot
  - Capture a screenshot of the Name and tags panel.



## Step by Step Launch EC2 Instances with Tags

Instances (2) <a href="#">Info</a>								
<input type="text"/> Find Instance by attribute or tag (case-sensitive)		All states <a href="#">▼</a>						
<input type="checkbox"/>	Name <a href="#">✎</a>	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
<input type="checkbox"/>	nextwork-pro...	i-0cfb95d8123b24690	<span>Running</span> <a href="#">Q</a> <a href="#">Q</a>	t2.micro	<span>2/2 checks passed</span> <a href="#">View alarms</a> +	ap-southeast-1b	ec2-54-179-147-241.ap...	
<input type="checkbox"/>	nextwork-dev...	i-0558abe784cabfea1	<span>Running</span> <a href="#">Q</a> <a href="#">Q</a>	t2.micro	- <a href="#">View alarms</a> +	ap-southeast-1b	ec2-3-0-97-81.ap-south...	

Instances (1/2) [Info](#)

<input type="checkbox"/>	Name <a href="#">✎</a>	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
<input type="checkbox"/>	nextwork-pro...	i-0cfb95d8123b24690	<span>Running</span> <a href="#">Q</a> <a href="#">Q</a>	t2.micro	<span>2/2 checks passed</span> <a href="#">View alarms</a> +	ap-southeast-1b	ec2-54-179-147-241.ap...	
<input checked="" type="checkbox"/>	nextwork-dev...	i-0558abe784cabfea1	<span>Running</span> <a href="#">Q</a> <a href="#">Q</a>	t2.micro	- <a href="#">View alarms</a> +	ap-southeast-1b	ec2-3-0-97-81.ap-south...	

i-0558abe784cabfea1 (nextwork-development-aimanhaziq)

[Details](#) [Status and alarms](#) [Monitoring](#) [Security](#) [Networking](#) [Storage](#) [Tags](#)

Tags

Key	Value
Env	production
Name	nextwork-development-aimanhaziq



# IAM Policies

IAM Policies are crucial for controlling access. The tag I've used on my EC2 instances is called 'Env'. The values are: 'production' for 'nextwork-production-aimanhaziq' 'development' for 'nextwork-development-aimanhaziq.'

## The policy I set up

For this project, I set up a policy using JSON. It provides detailed permissions control, allowing full access to instances tagged `Env=development`, while denying tag modifications.

I've created a policy that allows full access to EC2 instances tagged with `Env=development`, permits viewing and describing all EC2 resources, and denies any actions related to creating or deleting tags.

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

The Effect attribute specifies whether the policy allows or denies actions. Action defines what operations are permitted (e.g., ec2:StartInstances). Resource identifies which AWS resources the actions apply to.



# My JSON Policy

IAM > Policies > Create policy

Step 1  
**Specify permissions** Info

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

Step 2  
Review and create

**Policy editor**

```
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 }
```

**Visual** **JSON** **Actions ▾**

**Edit statement**

Select a statement

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

**+ Add new statement**



## Step by Step Create an IAM Policy

### Objective

Define permissions for accessing EC2 instances based on tags.

### Instructions

#### 1. Navigate to IAM Console

- Access the IAM console from the AWS services menu.

#### 2. Create a New Policy

- Choose Policies and click Create policy.
- Switch to JSON mode and paste the following policy (look at below)

#### 3. Fill Policy Details

- Name: NextWorkDevEnvironmentPolicy
- Description: IAM Policy for NextWork's development environment
- Create Policy
- Save the policy.

```
json Copy code

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "ec2:ResourceTag/Env": "development"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": [
        "ec2:DeleteTags",
        "ec2>CreateTags"
      ],
      "Resource": "*"
    }
  ]
}
```



## Step by Step Create an IAM Policy

Step 1 Policies > Create policy

Specify permissions [Info](#)

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

Step 2 Review and create

**Policy editor**

```
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   ]
}
```

Visual    **JSON**    Actions ▾

Edit statement

Select a statement

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

+ Add new statement

Review and create [Info](#)

Review the permissions, specify details, and tags.

**Policy details**

**Policy name**  
Enter a meaningful name to identify this policy.  
  
Maximum 128 characters. Use alphanumeric and '+,-,@-\_-' characters.

**Description - optional**  
Add a short explanation for this policy.  
  
Maximum 1,000 characters. Use alphanumeric and '+,-,@-\_-' characters.



## Step by Step Create an IAM Policy

The screenshot shows the AWS IAM Policies page with the policy 'NextWorkDevEnvironmentPolicy' selected. The policy details are as follows:

- Type:** Customer managed
- Creation time:** July 31, 2024, 06:11 (UTC+08:00)
- Edited time:** July 31, 2024, 06:11 (UTC+08:00)
- ARN:** arn:aws:iam::538908854590:policy/NextWorkDevEnvironmentPolicy

The 'Permissions' tab is selected, showing the JSON document:

```
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        }  
18    }]
```

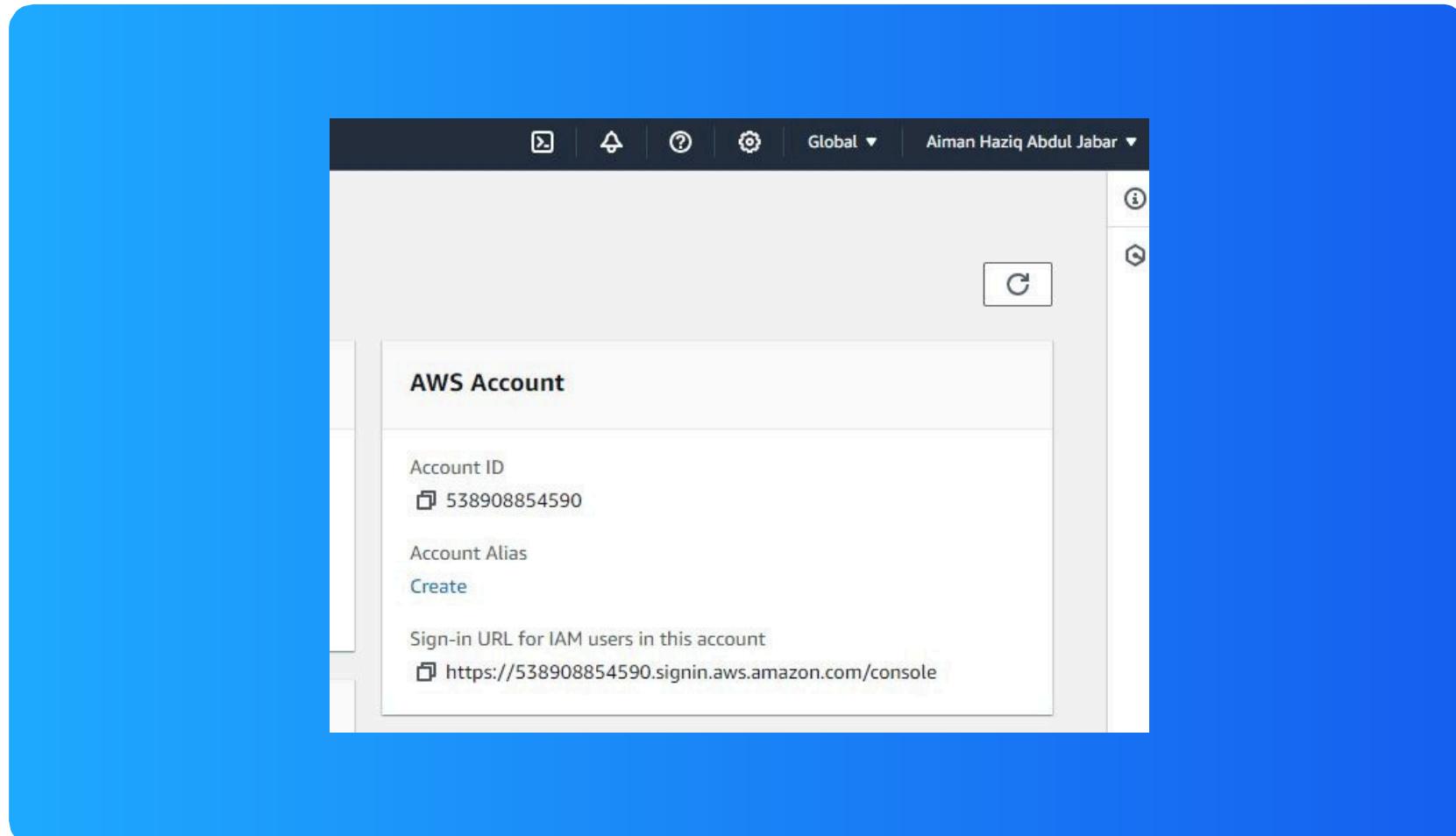


# Account Alias

An account alias is a unique name that you can assign to your AWS account to make it easier to remember and access. Instead of using the default account ID, you can log in using the alias.

Creating an account alias took me just a few minutes. The process involves navigating to the IAM dashboard, entering the desired alias, and confirming the creation.

Now, my new AWS console sign-in URL is <https://nextwork-alias-aimanhaziq.signin.aws.amazon.com/console>.





## Step by Step Create an AWS Account Alias

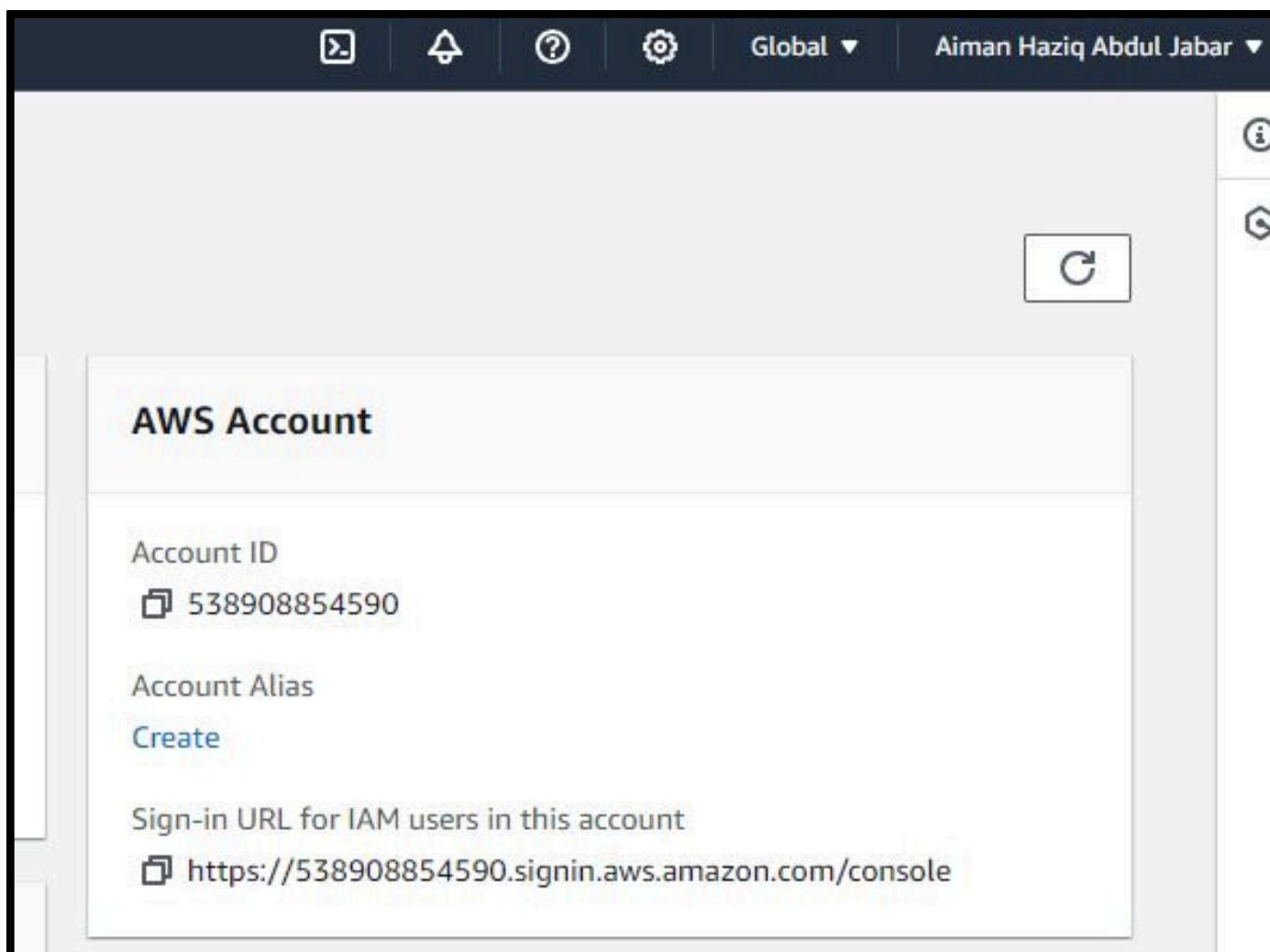
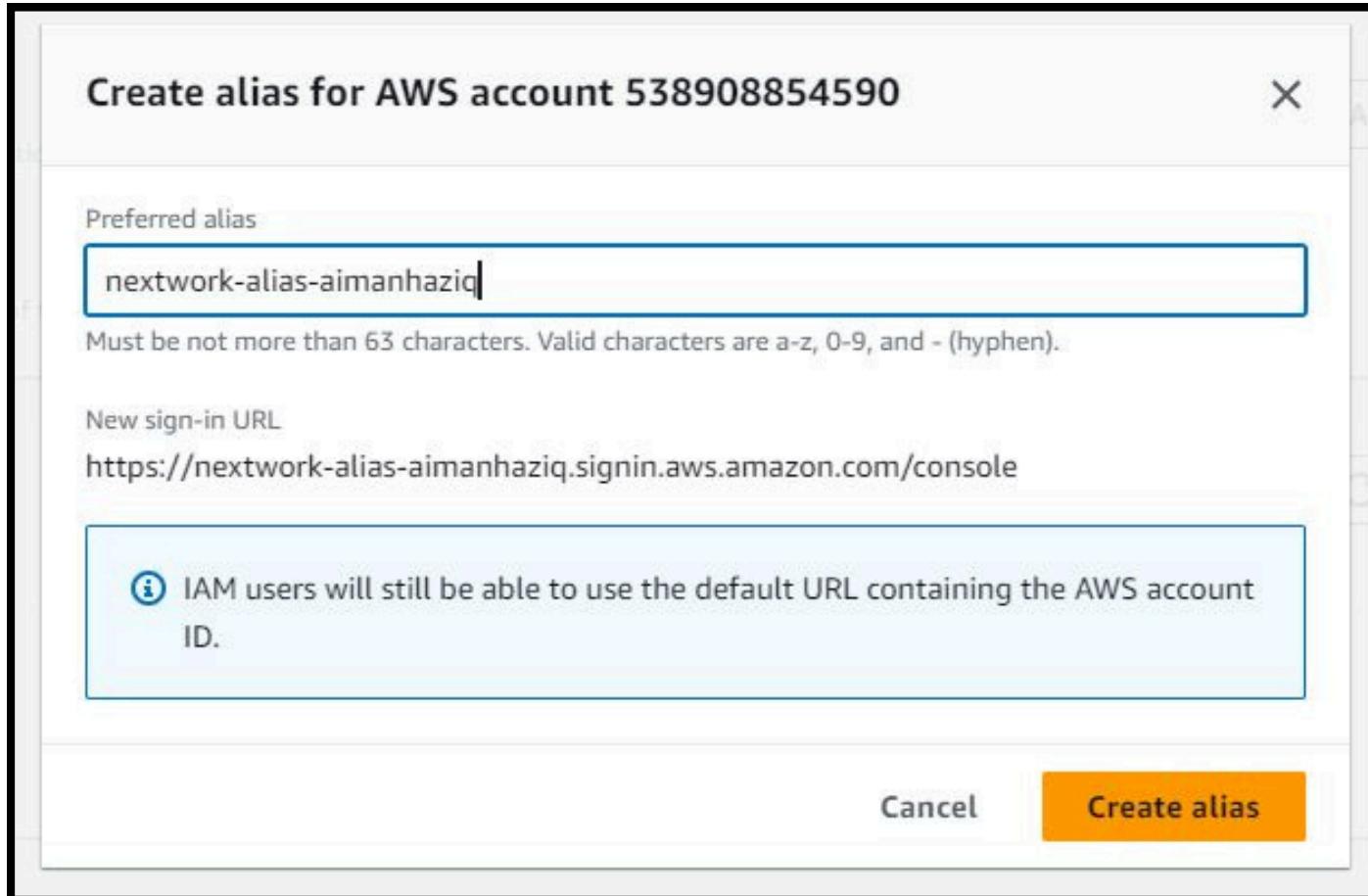
### Objective

Assign a custom alias to your AWS account for easier identification.

### Instructions

1. Navigate to IAM Dashboard
  - Access the IAM dashboard from the AWS services menu.
2. Create Account Alias
  - Choose Create under Account Alias.
  - Enter nextwork-alias-aimanhaziq.
3. Screenshot
  - Capture a screenshot of the Account Alias popup.
4. Create Alias
  - Confirm and create the alias.

## Step by Step Create an AWS Account Alias





# IAM Users and User Groups

## Users

IAM users are entities created within your AWS account that represent individuals or applications. Each IAM user has unique credentials and permissions to access AWS services and resources.

## User Groups

IAM user groups are collections of IAM users. They allow you to manage permissions for multiple users simultaneously by assigning policies to the group, rather than to individual users.

I attached the policy I created to this user group, which means all users in the group inherit the permissions defined in the policy. This simplifies management by applying consistent access controls to multiple users.



## Step by Step Create an AWS Account Alias

### Objective

Set up IAM users and groups with appropriate permissions.

### Instructions

#### 1. Create a User Group

- Choose User groups and click Create group.
- Name: nextwork-dev-group
- Attach Policy: NextWorkDevEnvironmentPolicy
- Create the group.

#### 2. Add Users

- Choose Users and click Add user.
- User name: nextwork-dev-aimanhaziq
- Access Type: Check Provide user access to the Amazon Management Console.
- Uncheck Users must create a new password at next sign-in.
- Add the user to nextwork-dev-group.

#### 3. Screenshot

- Capture the console sign-in details.



## Step by Step Create an AWS Account Alias

[Alt+S]

Name the group

User group name  
Enter a meaningful name to identify this group.  
  
Maximum 128 characters. Use alphanumeric and '+,-,@-' characters.

Add users to the group - *Optional* (0) [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	Type	Used as	Description
<input checked="" type="checkbox"/> <a href="#">NextWorkDevEnvironmentPolicy</a>	Customer managed	None	IAM Policy for NextWorks development environment

No resources to display

Attach permissions policies - *Optional* (1/954) [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  
   1 match  

Policy name	Type	Used as	Description
<input checked="" type="checkbox"/> <a href="#">NextWorkDevEnvironmentPolicy</a>	Customer managed	None	IAM Policy for NextWorks development environment

© 2024, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

IAM > User groups

User groups (1) [Info](#)  
A user group is a collection of IAM users. Use groups to specify permissions for a collection of users.  
  

Group name	Users	Permissions	Creation time
<input checked="" type="checkbox"/> <a href="#">nextwork-dev-group</a>	0	Defined	Now



## Step by Step Create an AWS Account Alias

### Specify user details

**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*  
If you're providing console access to a person, it's a best practice [to manage their access in IAM Identity Center](#).

**Are you providing console access to a person?**

User type  
 Specify a user in Identity Center - Recommended  
We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.  
 I want to create an IAM user  
We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.

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

### 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 (1/1)**

Group name	Users	Attached policies	Created
nextwork-dev-group	0	<a href="#">NextWorkDevEnvironmentPolicy</a>	2024-07-31 (6 minutes ago)

**Set permissions boundary - *optional***

Cancel Previous Next



## Step by Step Create an AWS Account Alias

**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-aimanhaziq	Console password type Autogenerated	Require password reset No

**Permissions summary**

Name	Type	Used as
nextwork-dev-group	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.

Cancel Previous Create user

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

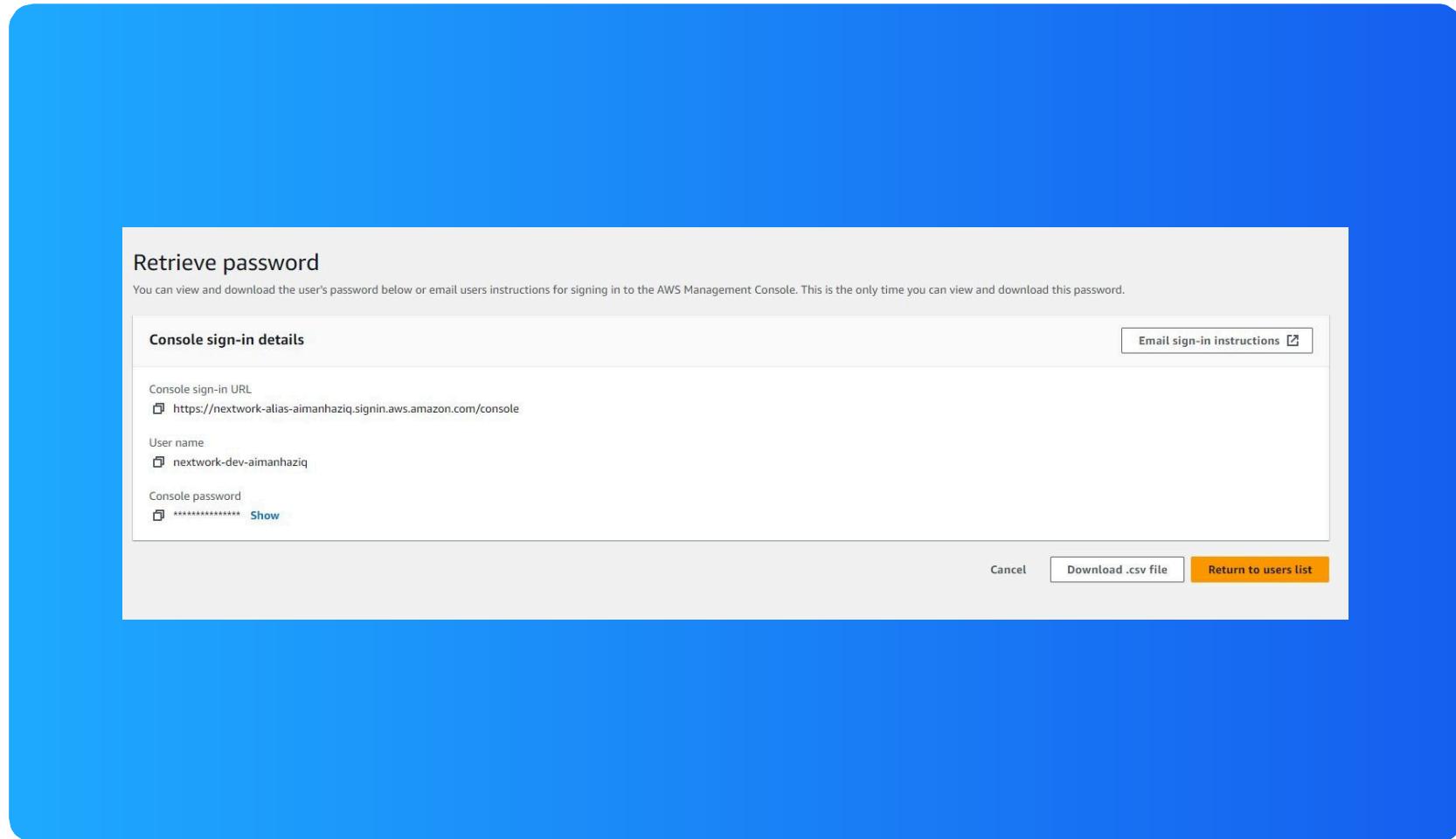
Cancel Download .csv file Return to users list



# Logging in as an IAM User

The first way is to email the sign-in details directly to the new user. The second way is to download the credentials as a CSV file and share it securely with the user.

Once I logged in as my IAM user, I noticed that the dashboard displayed limited access based on the policy. I had permissions for the development EC2 instance but restricted access to the production instance.





## Step by Step Test Your User's Access

### Objective

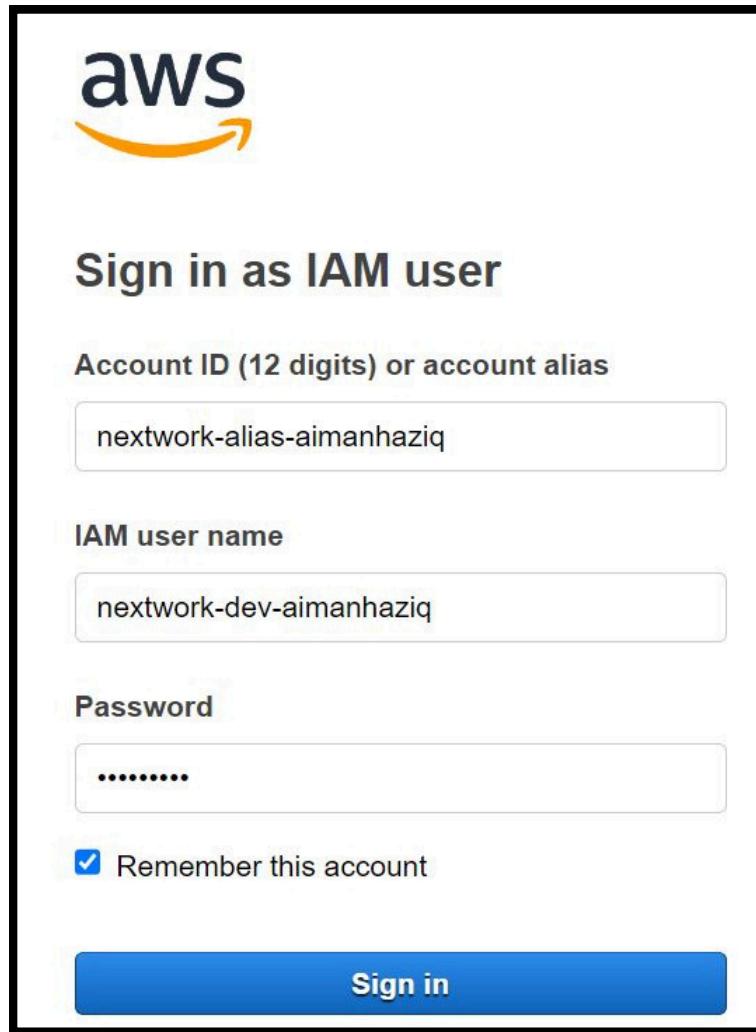
Verify that the IAM policy is working as intended by testing user permissions.

### Instructions

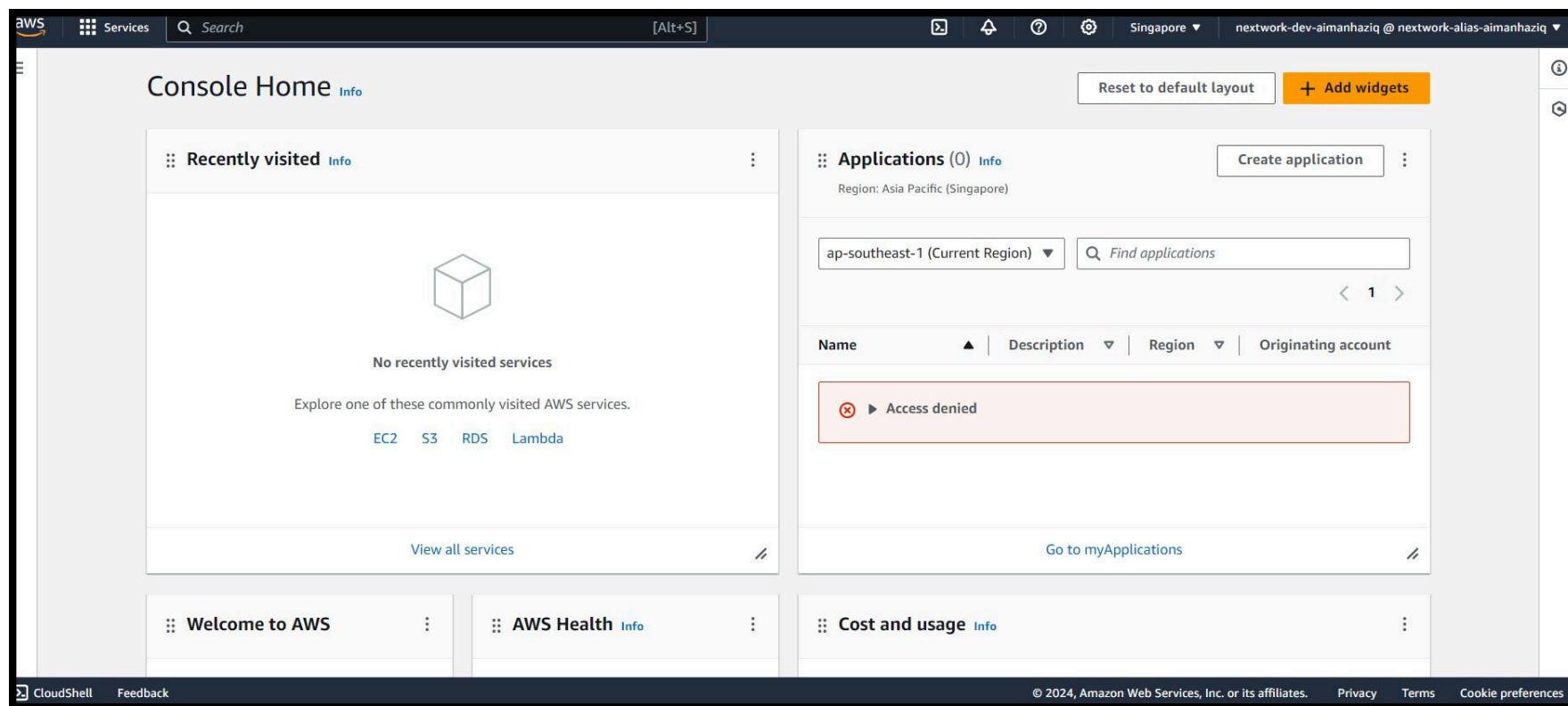
1. Copy Console Sign-in URL
  - Obtain the URL for console access.
2. Log In
  - Open an incognito window, access the URL, and log in using the new user credentials.
3. Test Access
  - Navigate to the EC2 console.
  - Attempt to stop the production instance (expect an authorization error).
  - Attempt to stop the development instance (should succeed).
4. Screenshot
  - Capture screenshots of error and success messages.



## Step by Step Test Your User's Access



The screenshot shows the AWS sign-in page for an IAM user. It features the AWS logo at the top. Below it, the heading "Sign in as IAM user" is displayed. There are three input fields: "Account ID (12 digits) or account alias" containing "nextwork-alias-aimanhaziq", "IAM user name" containing "nextwork-dev-aimanhaziq", and "Password" which is masked with dots. A checkbox labeled "Remember this account" is checked. At the bottom is a large blue "Sign in" button.



The screenshot shows the AWS Console Home page. The top navigation bar includes the AWS logo, a "Services" dropdown, a search bar, and account information ("nextwork-dev-aimanhaziq @ nextwork-alias-aimanhaziq"). The main area has a dark header with "Console Home" and a "Reset to default layout" button. Below the header, there are two main sections: "Recently visited" (empty) and "Applications". The "Applications" section shows a table with one row: "ap-southeast-1 (Current Region)" with a status of "Access denied". Other visible links include "View all services", "Go to myApplications", "Welcome to AWS", "AWS Health", and "Cost and usage". The footer contains standard links like "CloudShell", "Feedback", and copyright information ("© 2024, Amazon Web Services, Inc. or its affiliates.").

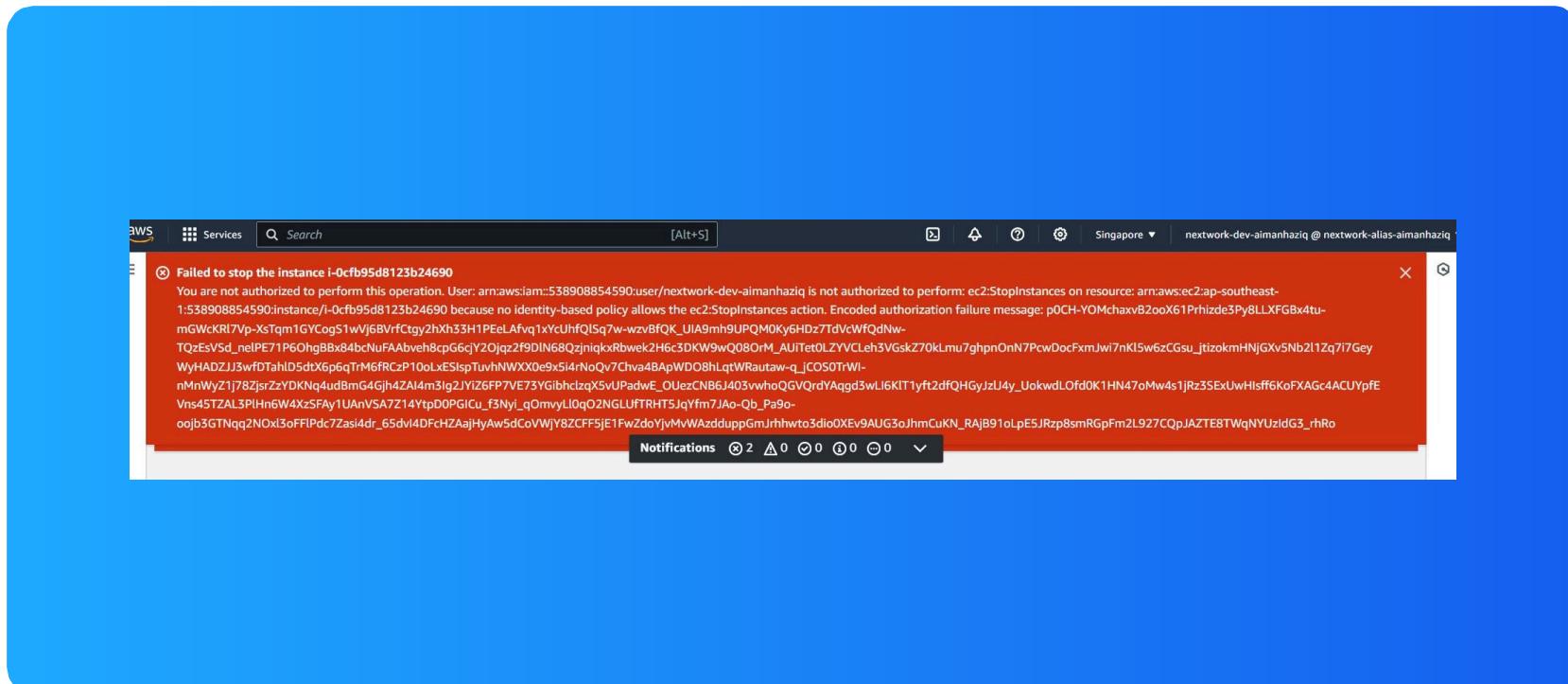


# Testing IAM Policies

I tested my JSON IAM policy by attempting to stop both EC2 instances. I was able to stop the `nextwork-development-aimanhaziq` instance but received an authorization error for the `nextwork-production-aimanhaziq` instance.

## Stopping the production instance

When I tried to stop the production instance, I received an authorization error, indicating that my IAM policy correctly restricted access to this action.





## Testing IAM Policies

Instances (1/2) [Info](#)

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Actions
nextwork-pro...	i-0cfb95d8123b24690	Running	t2.micro	2/2 checks passed	User: arn:aws:iam::538908854590:root	<a href="#">Connect</a>
nextwork-dev...	i-0558abe784cabfea1	Running	t2.micro	2/2 checks passed	User: arn:aws:iam::538908854590:root	<a href="#">View details</a>

WS Services Search [Alt+S]

Failed to stop the instance i-0cfb95d8123b24690

You are not authorized to perform this operation. User: arn:aws:iam::538908854590:user/nextwork-dev-aimanhaziq is not authorized to perform: ec2:StopInstances on resource: arn:aws:ec2:ap-southeast-1:538908854590:instance/i-0cfb95d8123b24690 because no identity-based policy allows the ec2:StopInstances action. Encoded authorization failure message: p0CH-YOMchavxB2ooX61Prhizde3Py8LLXFGBx4tu-mGwckRl7Vp-XsTqm1GYCogS1wVj6BVrfCtgy2hXh33H1PEelAfVq1YcUhfQISq7w-wzvBfQK\_UIA9mh9UPQMOKy6HDz7TdVcWfQdNw-TQzEsVsdlneIPE71P6OhgBBxB4bcNuFAAbveh8cpG6cJY2Ojqz2f9DIN68QzjnqkxRbwk2H6c3DKW9wQ08OrM\_AUItetOLZYVCLeh3VGskZ70kLmu7ghpnOnN7PcwDocFxmJwi7nKl5w6zCGsu\_jtzokmHNjGXv5Nb2l1Zq7i7Gey-WyHADZJJ3wfDTahlD5dtX6p6qTrM6fRCzP10oLxEStspTuvhNWXX0e9x5i4rNoQv7Chva4BApWDO8hLqtWRautaw-q\_jCOS0TrWi-nMnWyZ1j78ZjsrZzYDKNq4udBmG4Gjh4ZAl4m3lg2JYIZ6FP7VE73YGibhclzqX5vUPadwE\_OUezCNB61403vwhoQGVQrdYAqgd3wLi6KIT1yft2dfQHGyJzU4y\_UokwdLOfd0K1HN47oMw4s1jRz3SExUwHsff6KoFXAGc4ACUYpfEVns45TZA3Plhn6W4XzSFay1UAnVSA7Z14YtpDOPGIcu\_f3NyI\_qOmnyL0qQ2NGLUFTRHT5JqYfm7JAo-Qb\_Pa9o-oojb3GTNqq2NOxl3oFFIPdc7ZasI4dr\_65dvl4DFcHZAajHyAw5dCoVWjY8ZCF5jE1fwZdoYjvMvwAzdduppGmJrhwto3dio0XEv9AUG3oJhmCuKN\_RAjB91oLpE5JRzp8smRGpFm2L927CQpJAZTE8TwqNYUzldG3\_rhRo

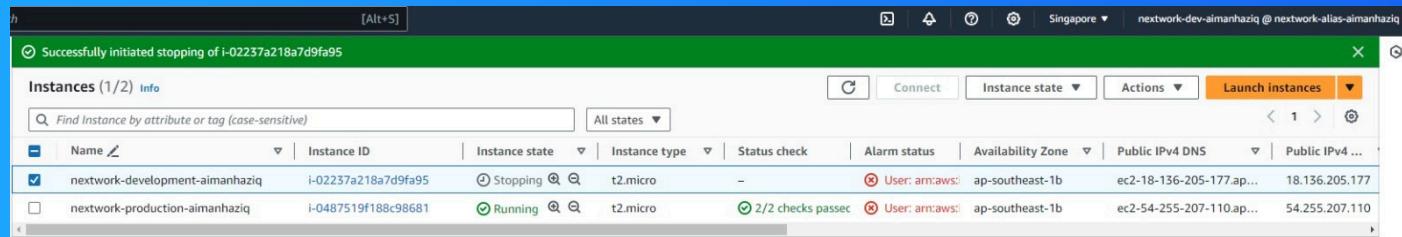
Notifications 2 ▲ 0 ○ 0 ⓘ 0 ⏺ 0

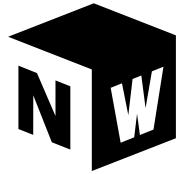


# Testing IAM Policies

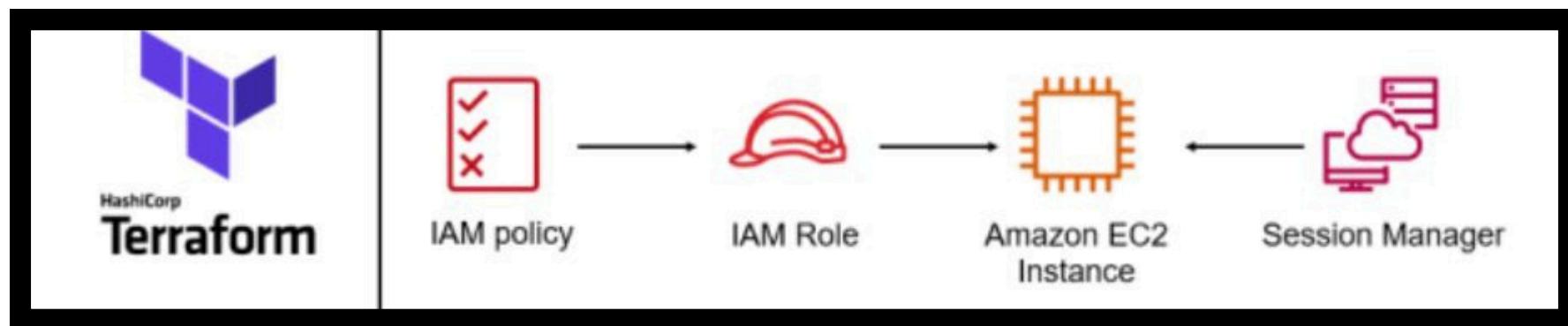
## Stopping the development instance

Next, when I tried to stop the development instance, the operation succeeded as expected, since the IAM policy allowed actions on instances with the `Env` tag set to `development`.





# Secret Mission: Cloud Security with Amazon IAM using EC2 using Terraform





## Secret Mission: Setting Up IAM Policy and EC2 with Terraform on Linux

### Prerequisites

1. Linux System: Ensure you have a Linux distribution installed.
2. AWS Account: Create an AWS account if you don't already have one.
3. Amazon IAM User: Create an IAM user with administrative access in your AWS account and get the access key and secret key for the user.

### Step 1: Install Required Software

#### 1.1 Install Terraform

1. Open your terminal.
2. Run the following commands to install Terraform:

```
cisco@labvm:~$ curl -O https://releases.hashicorp.com/terraform/1.5.3/terraform_1.5.3_linux_amd64.zip
```

```
cisco@labvm:~$ unzip terraform_1.5.3_linux_amd64.zip
```

```
cisco@labvm:~$ sudo mv terraform /usr/local/bin/
```



## 1.2 Install AWS CLI

1. Open your terminal.
2. Run the following commands to install the AWS CLI:

```
cisco@labvm:~$ sudo apt-get install awscli -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docutils-common groff libimagequant0 libnetpbm10 libpaper-utils libraqm0 mailcap mime-support netpbm psutils python3-botocore python3-certifi
  python3-docutils python3-jmespath python3-olefile python3-pil python3-pygments python3-requests python3-roman python3-rsa python3-s3transfer
  python3-urllib3 sgml-base xml-core
Suggested packages:
  docutils-doc fonts-linuxlibertine | ttf-linux-libertine texlive-lang-french texlive-latex-base texlive-latex-recommended python-pil-doc
  python-pygments-doc ttf-bitstream-vera python3-socks python-requests-doc sgml-base-doc debhelper
The following NEW packages will be installed:
  awscli docutils-common groff libimagequant0 libnetpbm10 libpaper-utils libraqm0 mailcap mime-support netpbm psutils python3-botocore
  python3-certifi python3-docutils python3-jmespath python3-olefile python3-pil python3-pygments python3-requests python3-roman python3-rsa
  python3-s3transfer python3-urllib3 sgml-base xml-core
0 upgraded, 25 newly installed, 0 to remove and 380 not upgraded.
Need to get 13.1 MB of archives.
After this operation, 102 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 sgml-base all 1.30 [12.5 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 groff amd64 1.22.4-8build1 [4,104 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 python3-jmespath all 0.10.0-1 [21.7 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 python3-urllib3 all 1.26.5-1~exp1ubuntu0.1 [98.2 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 python3-certifi all 2020.6.20-1 [150 kB]
```



## Setting Up AWS Infrastructure with Terraform on Linux

### 1.3 Install Visual Studio Code

1. Open your terminal.
2. Run the following commands to install Visual Studio Code:

```
cisco@labvm:~$ sudo apt-get install -y software-properties-common apt-transport-https wget
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-software-properties software-properties-gtk
Recommended packages:
  unattended-upgrades gnome-keyring
The following NEW packages will be installed:
  apt-transport-https
The following packages will be upgraded:
  python3-software-properties software-properties-common
  software-properties-gtk wget
4 upgraded, 1 newly installed, 0 to remove and 376 not upgraded.
Need to get 454 kB of archives.
After this operation, 121 kB of additional disk space will be used.
```

```
cisco@labvm:~$ wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
cisco@labvm:~$ sudo add-apt-repository "deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable main"
Repository: 'deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable main'
Description:
Archive for codename: stable components: main
More info: https://packages.microsoft.com/repos/vscode
Adding repository.
Press [ENTER] to continue or Ctrl-c to cancel.
```

```
cisco@labvm:~$ sudo apt update
Hit:1 https://packages.microsoft.com/repos/vscode stable InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:4 https://ppa.launchpadcontent.net/mozillateam/ppa/ubuntu jammy InRelease
Hit:5 https://ppa.launchpadcontent.net/wireshark-dev/stable/ubuntu jammy InRelease
Hit:6 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:7 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Reading package lists... Done
```



## Setting Up AWS Infrastructure with Terraform on Linux

```
cisco@labvm:~$ sudo apt install code
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libvulkani mesa-vulkan-drivers
```



## Setting Up AWS Infrastructure with Terraform on Linux

### Step 2: Configure AWS CLI

1. Open your terminal and run:
2. Provide your AWS credentials (access key, secret access key, region, and output format):

```
cisco@labvm:~$ aws configure
AWS Access Key ID [None]: AKIAJ26LYRE7HI6FNWVW
AWS Secret Access Key [None]: lvN/gDmWdt2vEbybhzzMnBJPk/4BZjcWQTj36NTp
Default region name [None]: ap-southeast-1
Default output format [None]: json
```

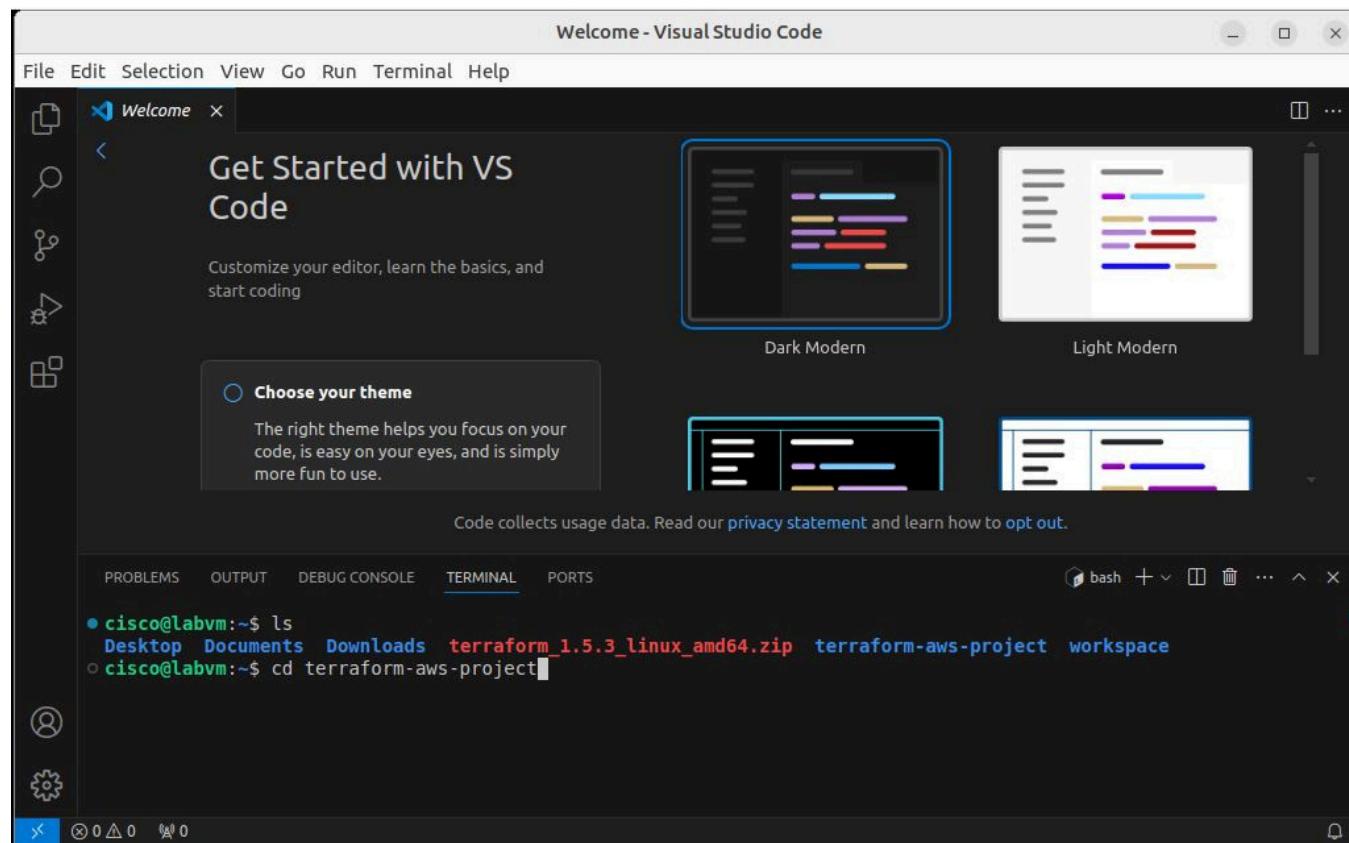


# Setting Up AWS Infrastructure with Terraform on Linux

## Step 3: Set Up Your Terraform Project in Visual Studio Code

1. Open Visual Studio Code and create a new directory for your project.

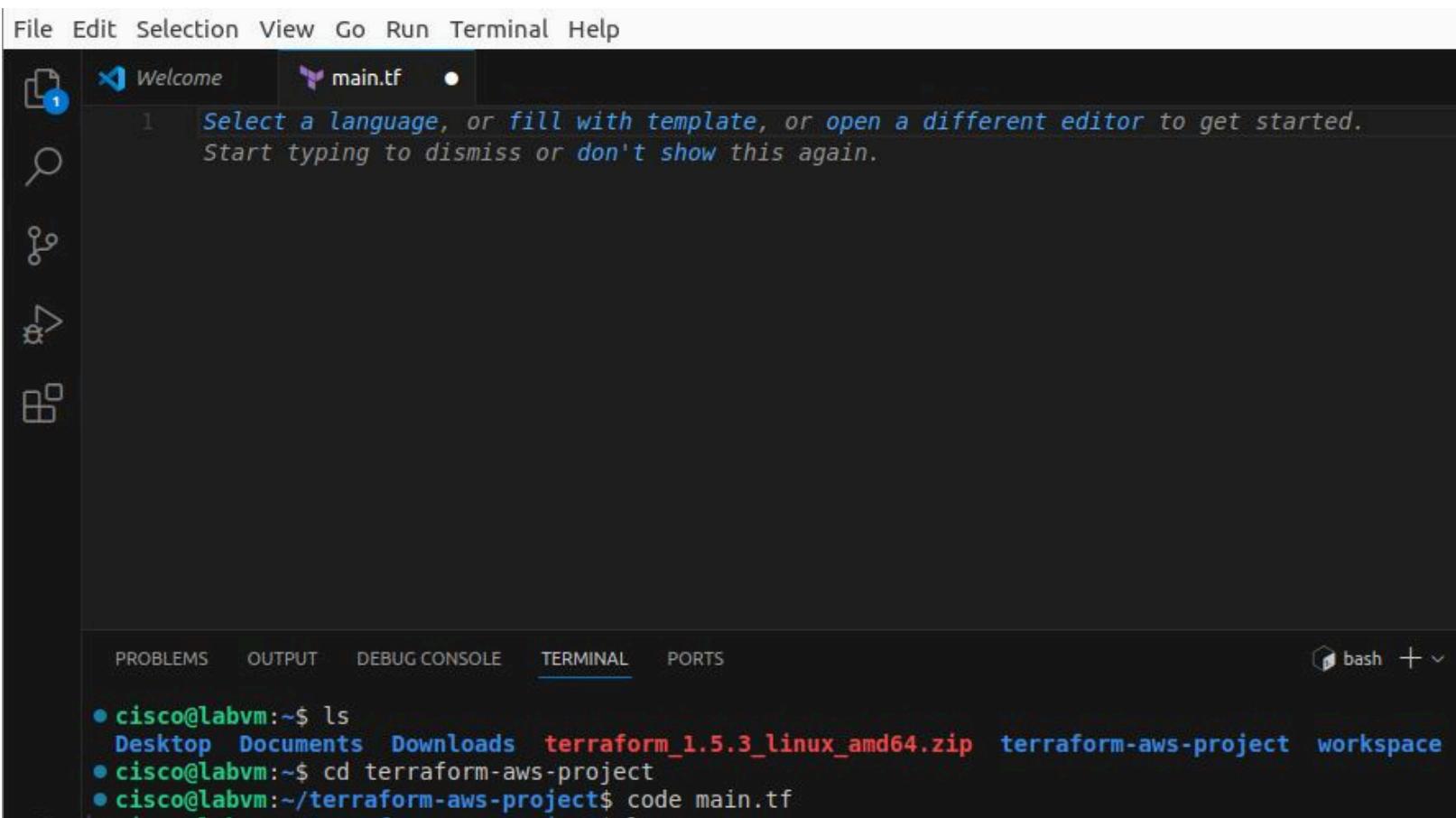
Open the terminal within VS Code (Ctrl + or Terminal > New Terminal). Use **mkdir terraform-aws-project** and **cd terraform-aws-project**



## Setting Up AWS Infrastructure with Terraform on Linux

2. Create main.tf file by using **touch main.tf** and **code main.tf**

```
● cisco@labvm:~/terraform-aws-project$ touch main.tf
● cisco@labvm:~/terraform-aws-project$ ls
main.tf
● cisco@labvm:~/terraform-aws-project$ code main.tf
```





# Setting Up AWS Infrastructure with Terraform on Linux

## Step 4: Write Terraform Configuration

In the main.tf file, write the following configuration:

```
main.tf  x
home > cisco > terraform-aws-project > main.tf
1 provider "aws" {
2   region = "ap-southeast-1"
3 }
4
5 resource "aws_instance" "production_instance" {
6   ami           = "ami-012c2e8e24e2ae21d"
7   instance_type = "t2.micro"
8   tags = {
9     Name = "nextwork-production-aimanhaziq"
10    Env  = "production"
11  }
12 }
13
14 resource "aws_instance" "development_instance" {
15   ami           = "ami-012c2e8e24e2ae21d"
16   instance_type = "t2.micro"
17   tags = {
18     Name = "nextwork-development-aimanhaziq"
19     Env  = "development"
20   }
21 }
22
23 resource "aws_iam_policy" "nextwork_dev_environment_policy" {
24   name          = "NextWorkDevEnvironmentPolicy"
25   description   = "IAM Policy for NextWork's development environment"
26   policy        = <<EOF
27 {
28   "Version": "2012-10-17",
29   "Statement": [
30     {
31       "Effect": "Allow",
32       "Action": [
33         "ec2:StartInstances",
34         "ec2:StopInstances",
35         "ec2:RebootInstances"
36       ],
37       "Resource": "*",
38       "Condition": {
39         "StringEquals": {
40           "ec2:ResourceTag/Env": "development"
41         }
42       }
43     },
44     {
45       "Effect": "Allow",
46       "Action": [
47         "ec2:DescribeInstances",
48         "ec2:DescribeTags"
49       ],
50       "Resource": "*"
51     },
52     {
53       "Effect": "Deny",
54       "Action": [
55         "ec2:TerminateInstances",
56         "ec2:DeleteTags",
57         "ec2>CreateTags"
58       ],
59       "Resource": "*"
60     }
61   ]
62 }
63 EOF
64 }
```

```
main.tf  x
home > cisco > terraform-aws-project > main.tf
27 {
28   "Statement": [
29     {
30       "Effect": "Allow",
31       "Action": [
32         "ec2:StartInstances",
33         "ec2:StopInstances",
34         "ec2:RebootInstances"
35       ],
36       "Resource": "*",
37       "Condition": {
38         "StringEquals": [
39           "ec2:ResourceTag/Env": "development"
40         ]
41       }
42     },
43     {
44       "Effect": "Allow",
45       "Action": [
46         "ec2:DescribeInstances",
47         "ec2:DescribeTags"
48       ],
49       "Resource": "*"
50     },
51     {
52       "Effect": "Deny",
53       "Action": [
54         "ec2:TerminateInstances",
55         "ec2:DeleteTags",
56         "ec2>CreateTags"
57       ],
58       "Resource": "*"
59     }
60   ]
61 }
62 EOF
63 }
```



# Setting Up AWS Infrastructure with Terraform on Linux

## Step 4: Write Terraform Configuration

```
main.tf
home > cisco > terraform-aws-project > main.tf
64 }
65
66 resource "aws_iam_group" "nextwork_dev_group" {
67   name = "nextwork-dev-group"
68 }
69
70 resource "aws_iam_group_policy_attachment" "attach_policy_to_group" {
71   group      = aws_iam_group.nextwork_dev_group.name
72   policy_arn = aws_iam_policy.nextwork_dev_environment_policy.arn
73 }
74
75 resource "aws_iam_user" "nextwork_dev_user" {
76   name = "nextwork-dev-aimanhaziq"
77 }
78
79 resource "aws_iam_user_group_membership" "add_user_to_group" {
80   user    = aws_iam_user.nextwork_dev_user.name
81   groups = [
82     aws_iam_group.nextwork_dev_group.name
83   ]
84 }
85
86 resource "aws_iam_user_login_profile" "login_profile" {
87   user = aws_iam_user.nextwork_dev_user.name
88   password_reset_required = false
89 }
90
```



# Setting Up AWS Infrastructure with Terraform on Linux

## Step 5: Initialize and Apply Terraform Configuration

### 1. Initialize Terraform:

```
cisco@labvm:~/terraform-aws-project$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.61.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

### 2. Validate the Configuration. This mean you can know which line is error:

```
cisco@labvm:~/terraform-aws-project$ terraform validate
Error: "policy" contains an invalid JSON policy: invalid character '#' looking for beginning of object key string, at byte offset 59
with aws_iam_policy.nextwork_dev_environment_policy,
on main.tf line 26, in resource "aws_iam_policy" "nextwork_dev_environment_policy":
26:   policy      = <EOF
27: {
28:   "Version": "2012-10-17",
29:   "Statement": [
30:     {
31:       # This statement allows specific actions (Start, Stop, Reboot) on EC2 instances
32:       # but only for those instances that are tagged with "Env=development".
33:       "Effect": "Allow",
34:       "Action": [
35:         "ec2:StartInstances",
36:         "ec2:StopInstances",
37:         "ec2:RebootInstances"
38:       ],
39:       "Resource": "*",
40:       "Condition": {
41:         "StringEquals": {
42:           "ec2:ResourceTag/Env": "development"
43:         }
44:       },
45:     },
46:     {
47:       # This statement allows read-only actions on all EC2 instances.
48:       # Actions like DescribeInstances and DescribeTags are allowed for all instances,
49:       # regardless of their tags.
50:       "Effect": "Allow",
51:       "Action": [
52:         "ec2:DescribeInstances",
53:         "ec2:DescribeTags"
54:       ],
55:       "Resource": "*"
56:     },
57:   }
```



# Setting Up AWS Infrastructure with Terraform on Linux

## Step 5: Initialize and Apply Terraform Configuration

### 3. Apply the Configuration:

```
cisco@labvm:~/terraform-aws-project$ terraform apply
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
+ create

Terraform will perform the following actions:

# aws_iam_group.nextwork_dev_group will be created
+ resource "aws_iam_group" "nextwork_dev_group" {
  + arn      = (known after apply)
  + id       = (known after apply)
  + name     = "nextwork-dev-group"
  + path     = "/"
  + unique_id = (known after apply)
}

# aws_iam_group_policy_attachment.attach_policy_to_group will be created
+ resource "aws_iam_group_attachment" "attach_policy_to_group" {
  + group      = "nextwork-dev-group"
  + id         = (known after apply)
  + policy_arn = (known after apply)
}

# aws_iam_policy.nextwork_dev_environment_policy will be created
+ resource "aws_iam_policy" "nextwork_dev_environment_policy" {
  + arn      = (known after apply)
  + attachment_count = (known after apply)
  + description      = "IAM Policy for NextWork's development environment"
  + id       = (known after apply)
  + name     = "NextWorkDevEnvironmentPolicy"
  + name_prefix = (known after apply)
  + path     = "/"
  + policy   = jsonencode(
    {
      + Statement = [
        + {
          + Action   = [
            + ...
          ]
        }
      ]
    }
  )
}
```

Terraform will show a plan of the actions it will take. Type yes to approve and apply the changes.



# Setting Up AWS Infrastructure with Terraform on Linux

## Step 6: Verify the Configuration

1. Log in to the Amazon Management Console using your web browser.
2. Navigate to the EC2 console to see the instances created.
3. Navigate to the IAM console to see the IAM policies, groups, and users created.

The screenshot shows the IAM Users page with two entries:

User name	Path	Groups	Last activity	MFA	Password age	Console last sign-in	Access key ID	Active key age	Access key last used
aimanhaziq	/	0	18 minutes ago	-	-	-	Active - AKIAZ6LYRE...	22 hours	18 minutes
nextwork-dev-aimanhaziq	/	1	-	-	-	18 minutes	-	-	-

The screenshot shows the IAM User groups page with one entry:

Group name	Users	Permissions	Creation time
nextwork-dev-group	1	Defined	19 minutes ago

The screenshot shows the EC2 Instances page with two running instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
nextwork-development-aimanhaziq	i-0410df0f632244ce	Running	t2.micro	2/2 checks passed	View alarms	ap-southeast-1b	ec2-18-140-66-118
nextwork-production-aimanhaziq	i-044ef1c3a79446f71	Running	t2.micro	2/2 checks passed	View alarms	ap-southeast-1b	ec2-18-141-174-36



# Setting Up AWS Infrastructure with Terraform on Linux

## Step 7: Delete all resources

### .1. Use **terraform destroy** for remove/delete all resources

```
cisco@labvm:~/terraform-aws-project$ terraform destroy
aws_instance.production_instance: Refreshing state... [id=i-044ef1c3a79446f71]
aws_iam_policy.nextwork_dev_environment_policy: Refreshing state... [id=arn:aws:iam::538908854590:policy/NextWorkDevEnvironmentPolicy]
aws_iam_user.nextwork_dev_user: Refreshing state... [id=nextwork-dev-aimanhaziq]
aws_iam_group.nextwork_dev_group: Refreshing state... [id=nextwork-dev-group]
aws_instance.development_instance: Refreshing state... [id=i-0410df0f632244ce]
aws_iam_user_login_profile.login_profile: Refreshing state... [id=nextwork-dev-aimanhaziq]
aws_iam_user_group_membership.add_user_to_group: Refreshing state... [id=terraform-20240802032412117600000003]
aws_iam_group_policy_attachment.attach_policy_to_group: Refreshing state... [id=nextwork-dev-group-20240802032412527800000004]

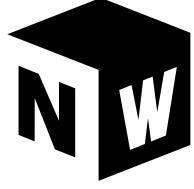
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# aws_iam_group.nextwork_dev_group will be destroyed
- resource "aws_iam_group" "nextwork_dev_group" {
    - arn      = "arn:aws:iam::538908854590:group/nextwork-dev-group" -> null
    - id       = "nextwork-dev-group" -> null
    - name     = "nextwork-dev-group" -> null
    - path     = "/" -> null
    - unique_id = "AGPAX26LYRE7K47JJ3PKB" -> null
}

# aws_iam_group_policy_attachment.attach_policy_to_group will be destroyed
- resource "aws_iam_group_policy_attachment" "attach_policy_to_group" {
    - group      = "nextwork-dev-group" -> null
    - id         = "nextwork-dev-group-20240802032412527800000004" -> null
    - policy_arn = "arn:aws:iam::538908854590:policy/NextWorkDevEnvironmentPolicy" -> null
}

# aws_iam_policy.nextwork_dev_environment_policy will be destroyed
- resource "aws_iam_policy" "nextwork_dev_environment_policy" {
    - arn      = "arn:aws:iam::538908854590:policy/NextWorkDevEnvironmentPolicy" -> null
    - attachment_count = 1 -> null
    - description      = "IAM Policy for NextWork's development environment" -> null
    - id       = "arn:aws:iam::538908854590:policy/NextWorkDevEnvironmentPolicy" -> null
}
```



NextWork.org

# Everyone should be in a job they love.

Check out [nextwork.org](http://nextwork.org) for  
more projects. Thank you  
Nextwork for providing this  
project.

---

