NextWork.org

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

Aiman Haziq

Aiman Haziq
NextWork Student

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...

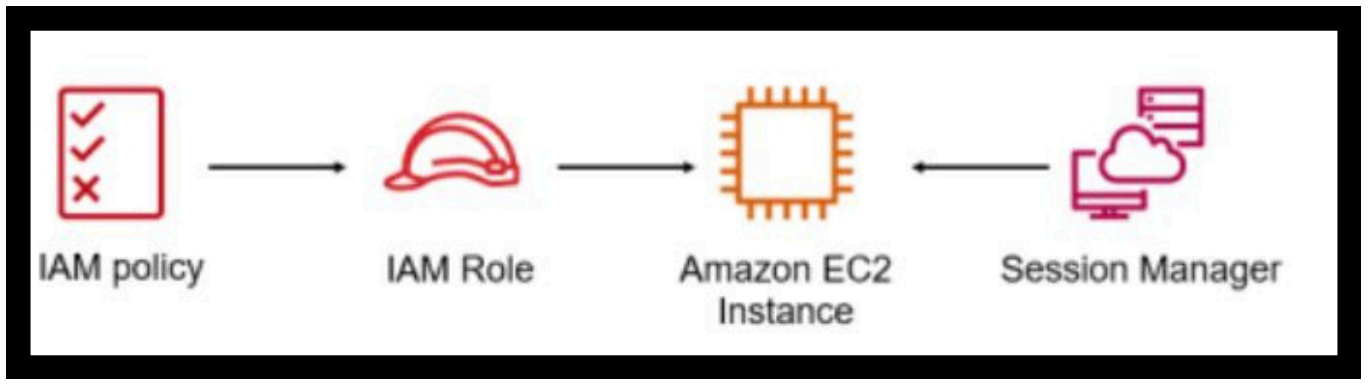
One thing I didn't expect was how useful IAM policies could be in precisely controlling access. The ability to tailor permissions for different environments and test them thoroughly added a layer of security and management I hadn't fully anticipated.

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.

NextWork.org

Cloud Security with Amazon IAM using EC2 using Amazon Management Console

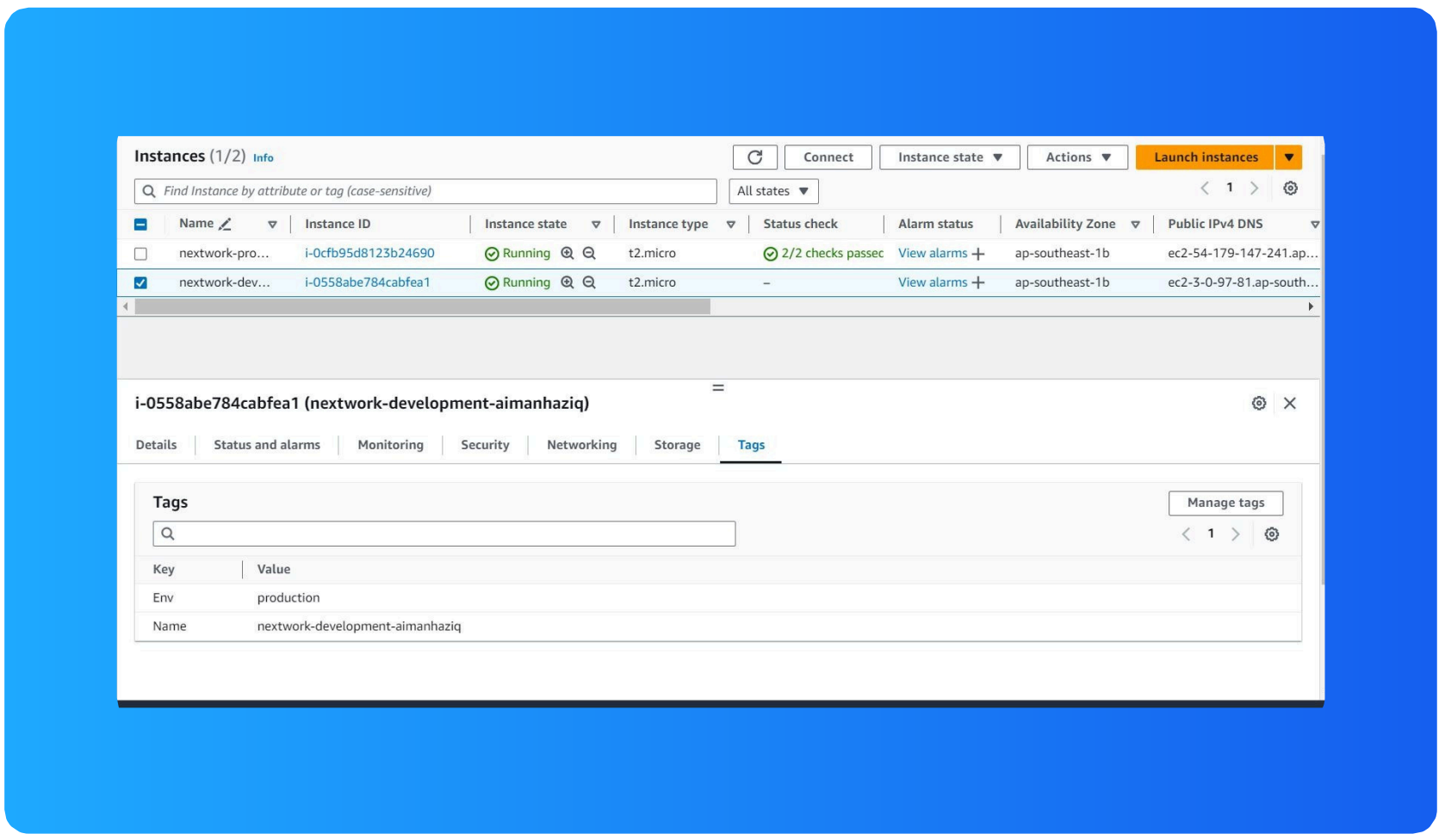


Aiman Haziq
NextWork Student

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 valuesassigned are: - For the production instance: `nextwork-production- aimanhaziq`, `Env=production`- For the development instance: `nextwork- development-aimanhaziq`, Env=development'



Aiman Haziq
NextWork Student
NextWork.org

Step by Step Launch EC2 Instances with Tags

Objective

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

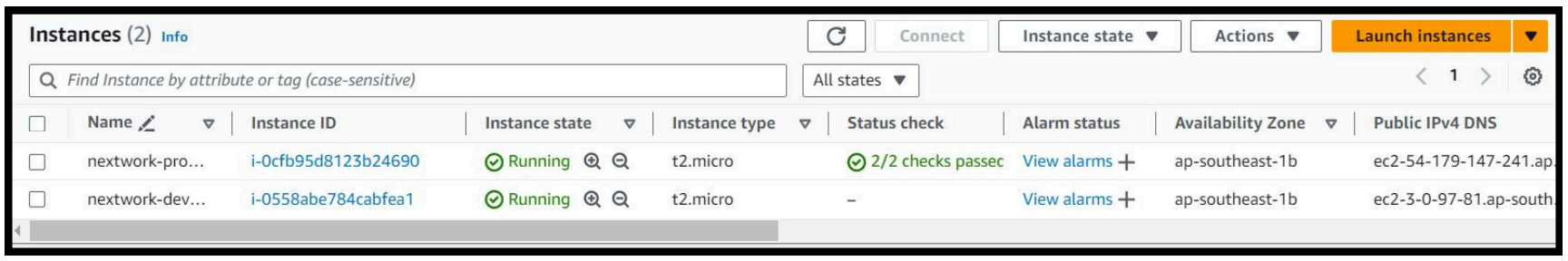
Instructions

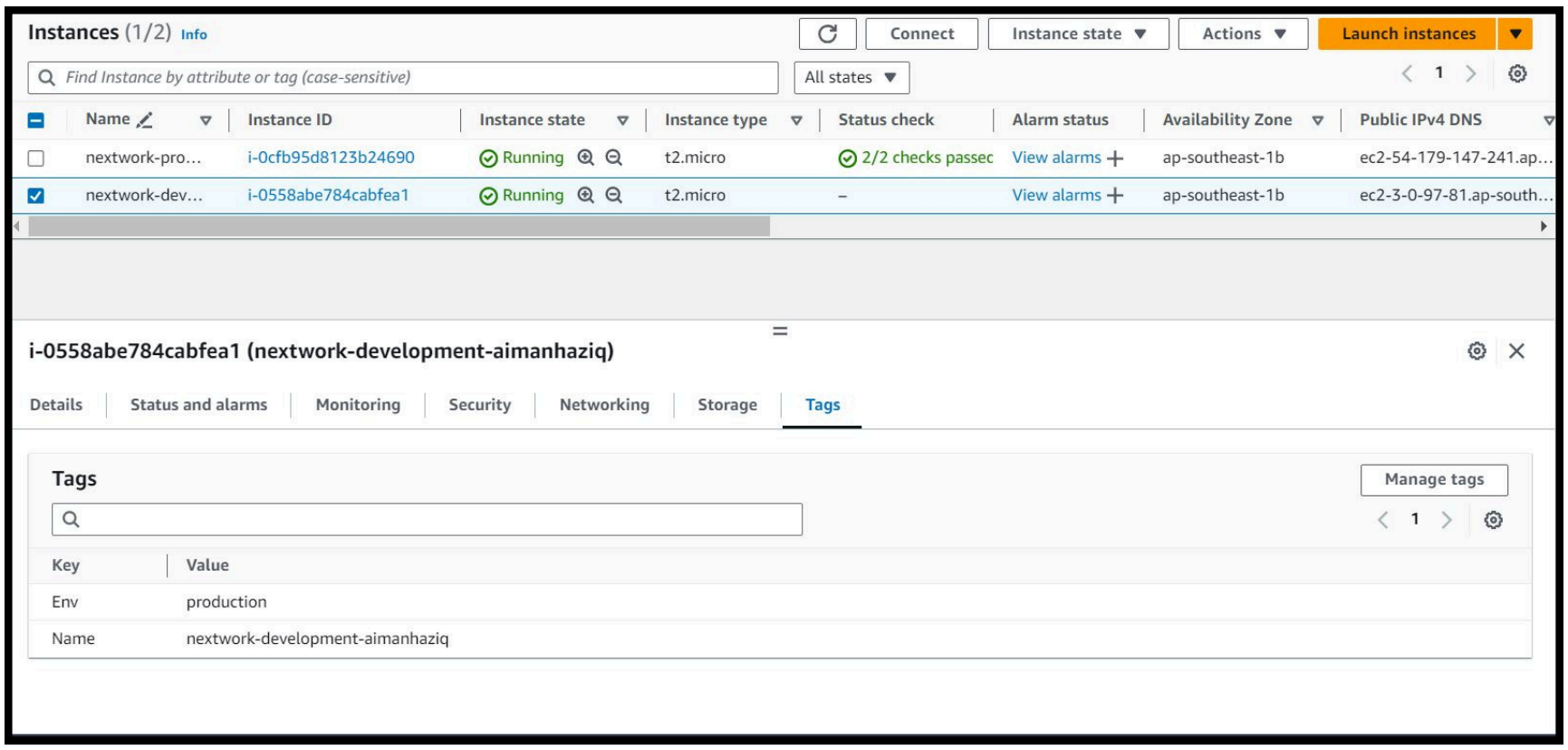
1. Log in to Amazon Management Console
    - Access the Amazon Management Console using your credentials.
2. Navigate to EC2 Service
    - Select EC2 from the AWS services menu.
3. Switch Region
    - Set the Region to the one closest to your location.
4. Launch EC2 Instances
    - Click Launch instances 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.

Aiman Haziq
NextWork Student

NextWork.org

Step by Step Launch EC2 Instances with Tags





Aiman Haziq
NextWork Student

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.

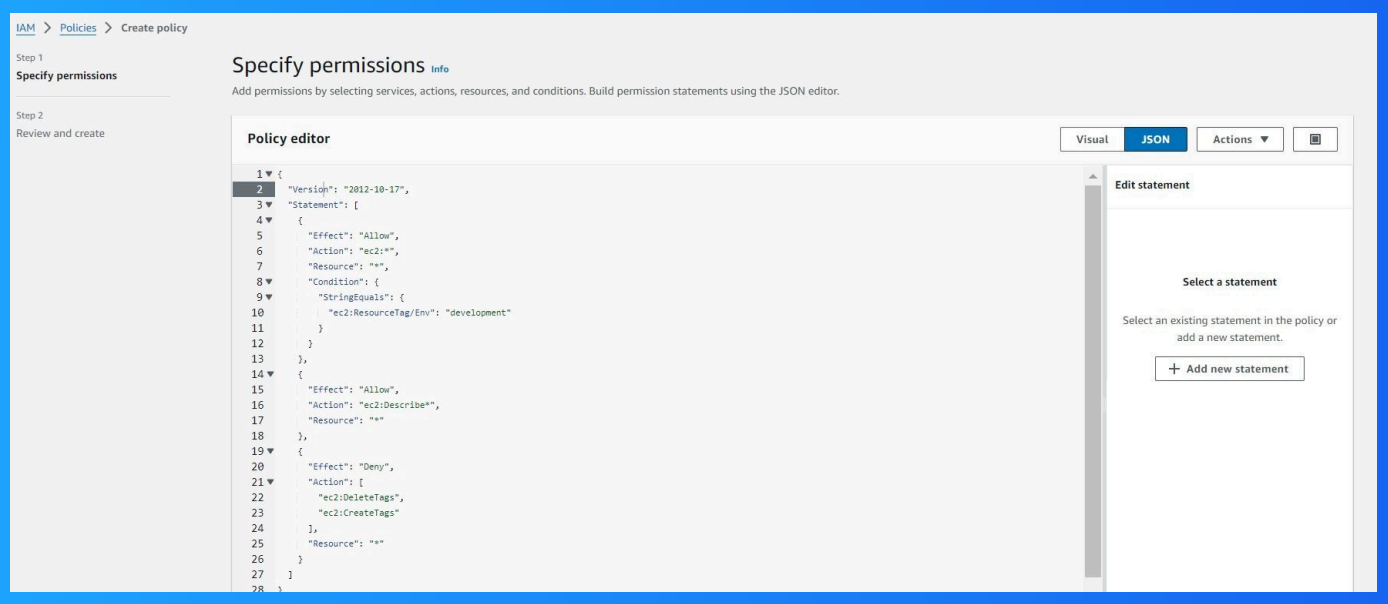
When creating a JSON policy, you have to define its Effect, Action and Resource.

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.

Aiman Haziq
NextWork Student

NextWork.org

My JSON Policy



Aiman Haziq
NextWork Student

Step by Step Create an IAM Policy

Objective

Define permissions for accessing EC2 instances based on tags.

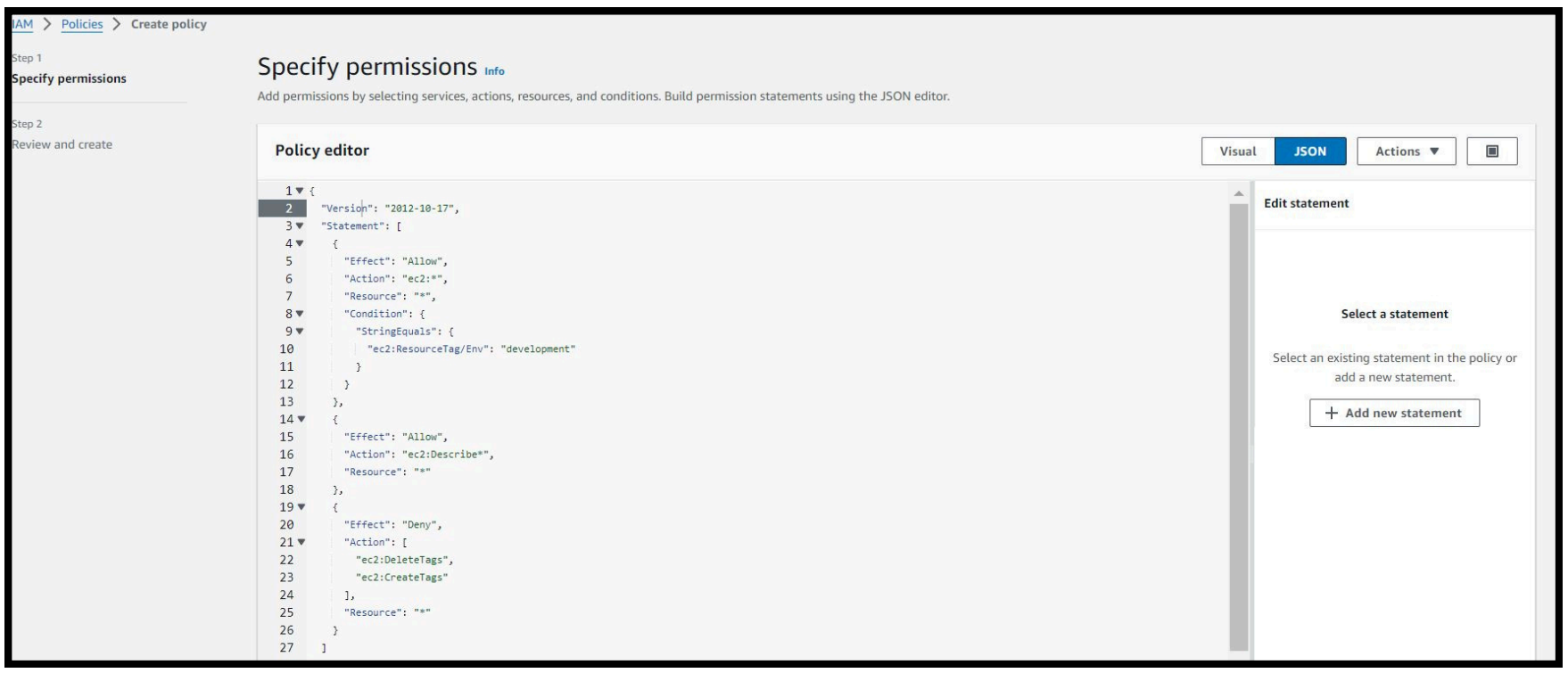
Instructions

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.

Aiman Haziq
NextWork Student

NextWork.org

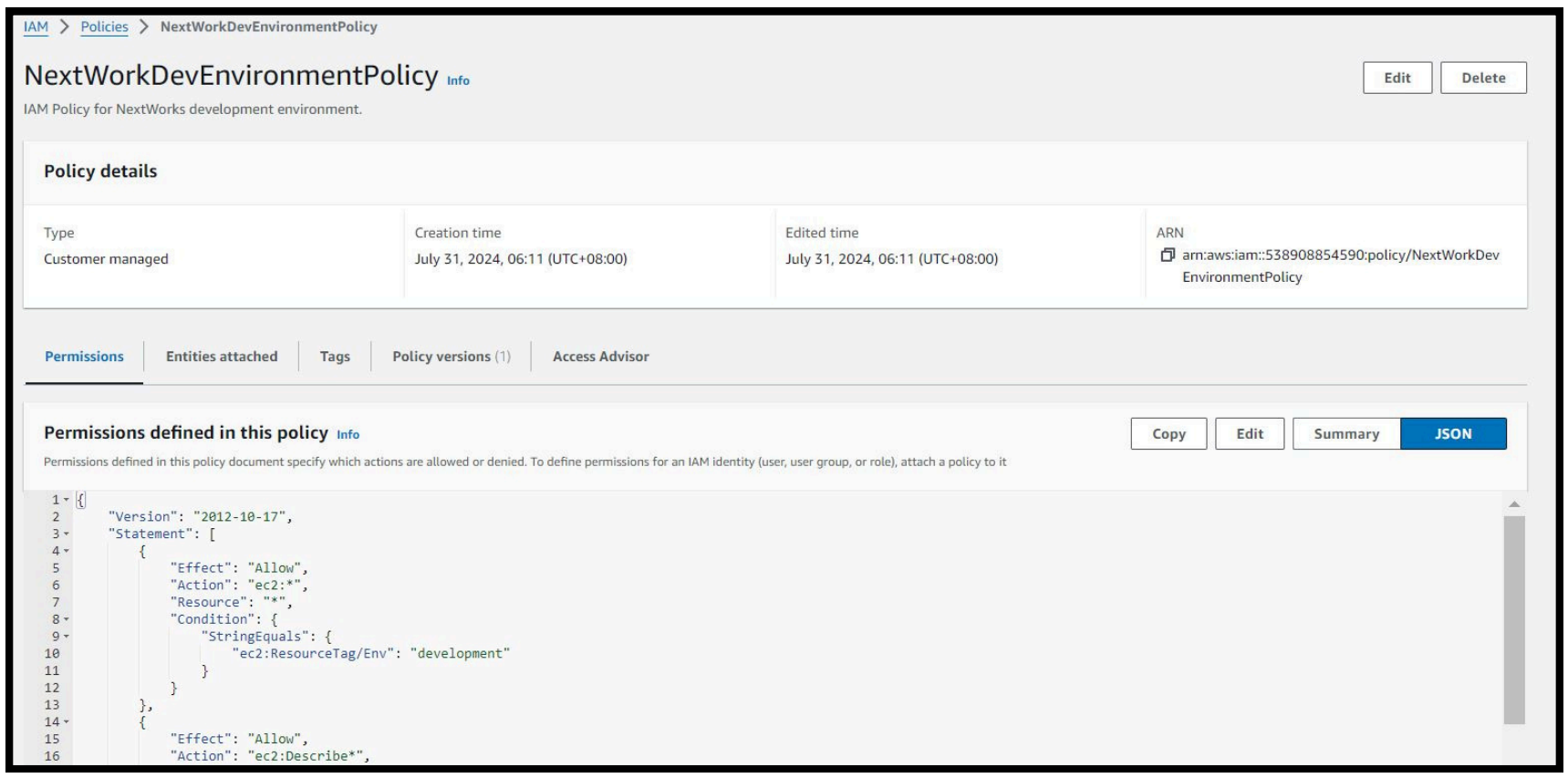
Step by Step Create an IAM Policy



Aiman Haziq
NextWork Student

NextWork.org

Step by Step Create an IAM Policy



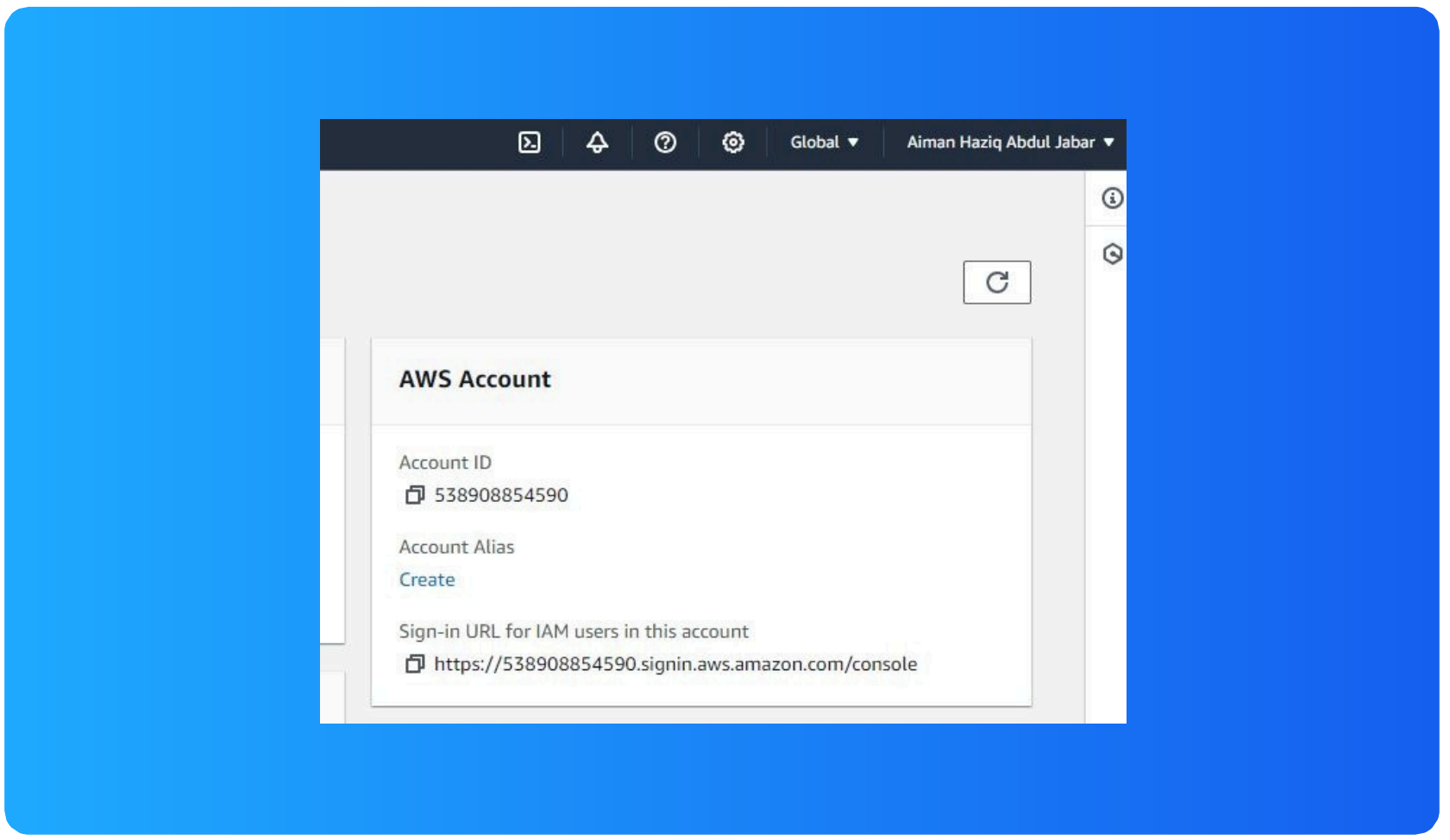
Aiman Haziq
NextWork Student

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.



Aiman Haziq
NextWork Student

Step by Step Create an AWS Account Alias

Objective

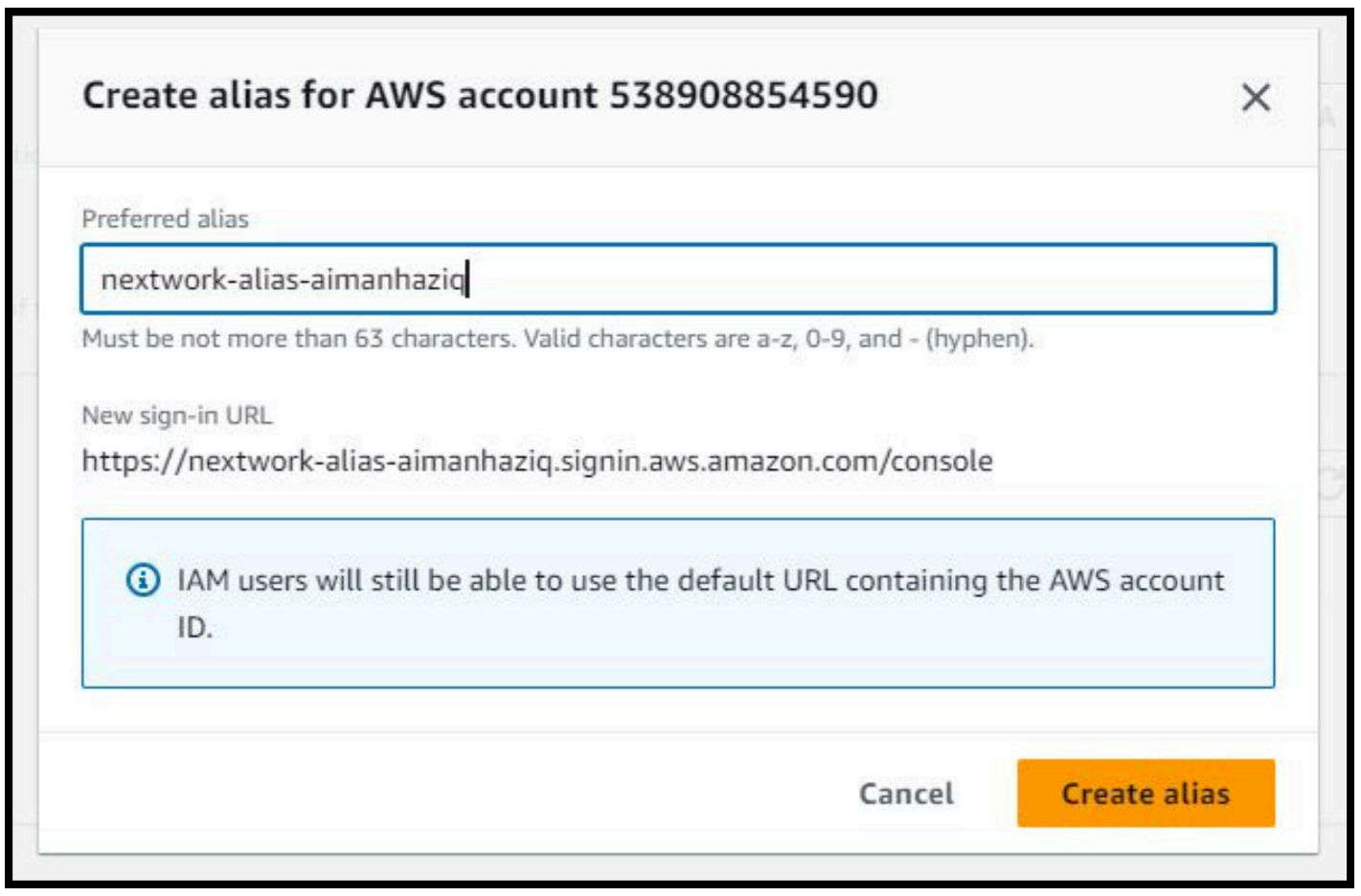
Assign a custom alias to your AWS account for easier identification.

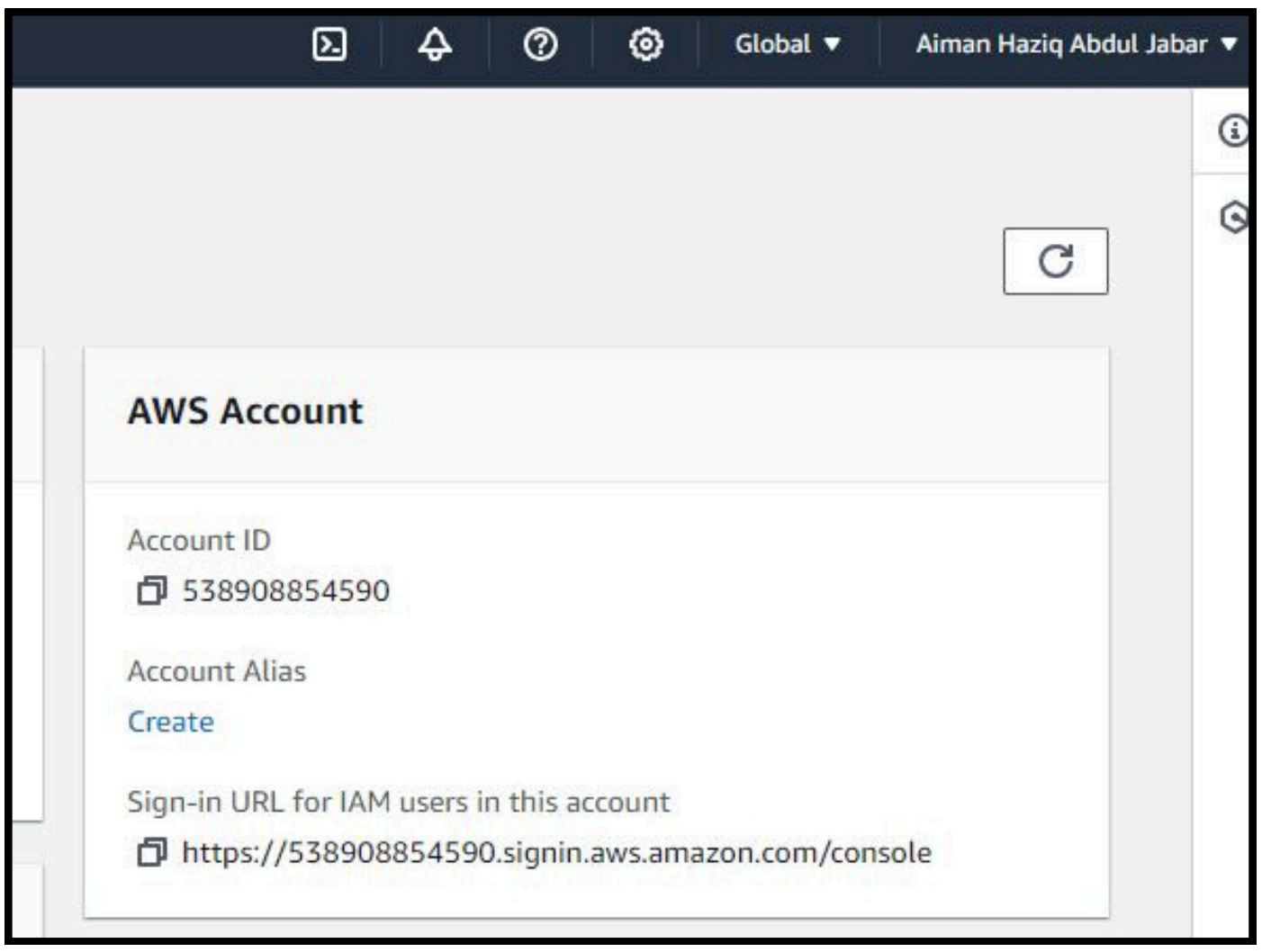
Instructions

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.

Aiman Haziq
NextWork Student

Step by Step Create an AWS Account Alias





Aiman Haziq
NextWork Student

NextWork.org

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.

Aiman Haziq
NextWork Student
NextWork.org

Step by Step Create an AWS Account Alias

Objective

Set up IAM users and groups with appropriate permissions.

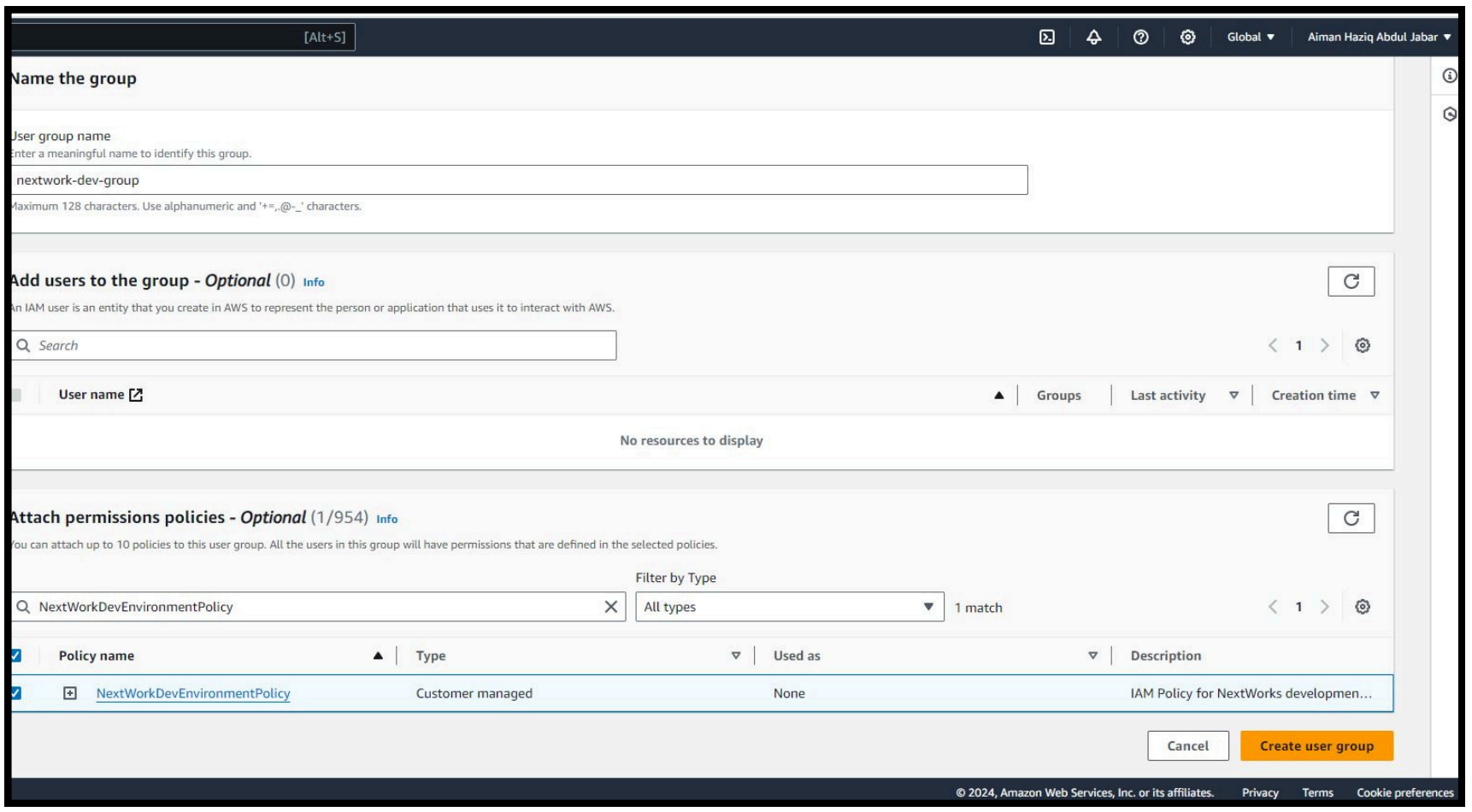
Instructions

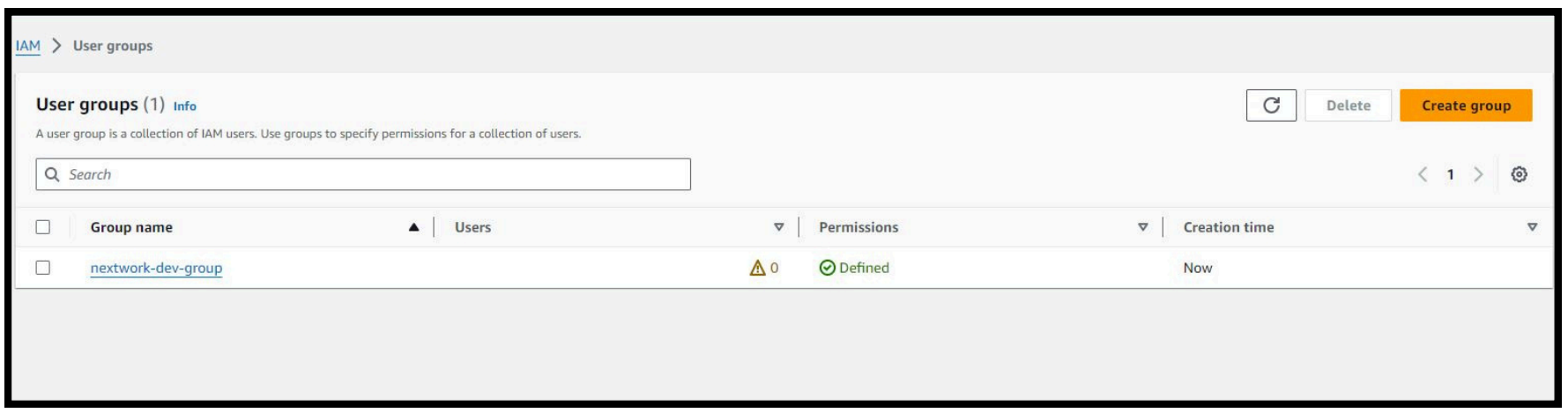
1. Create a User Group
    - Choose User groups and click Create group.
    - Name: nextwork-dev-group
    - Attach Policy: NextWorkDevEnvironmentPolicy
    - Create the group.
2. Add Users
    - Choose Users and click Add user.
    - User name: nextwork-dev-aimanhaziq
    - Access Type: Check Provide 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.

Aiman Haziq
NextWork Student

NextWork.org

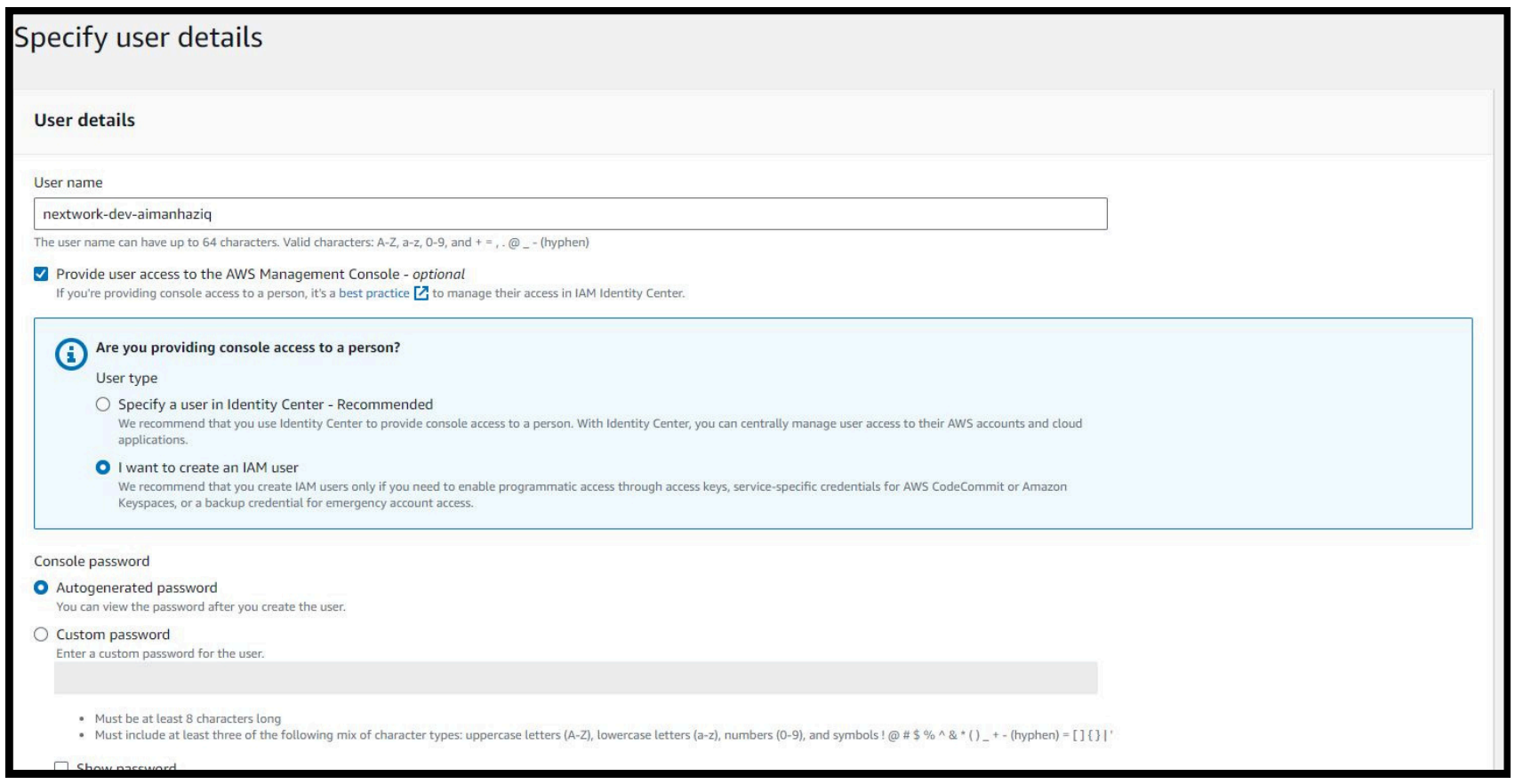
Step by Step Create an AWS Account Alias

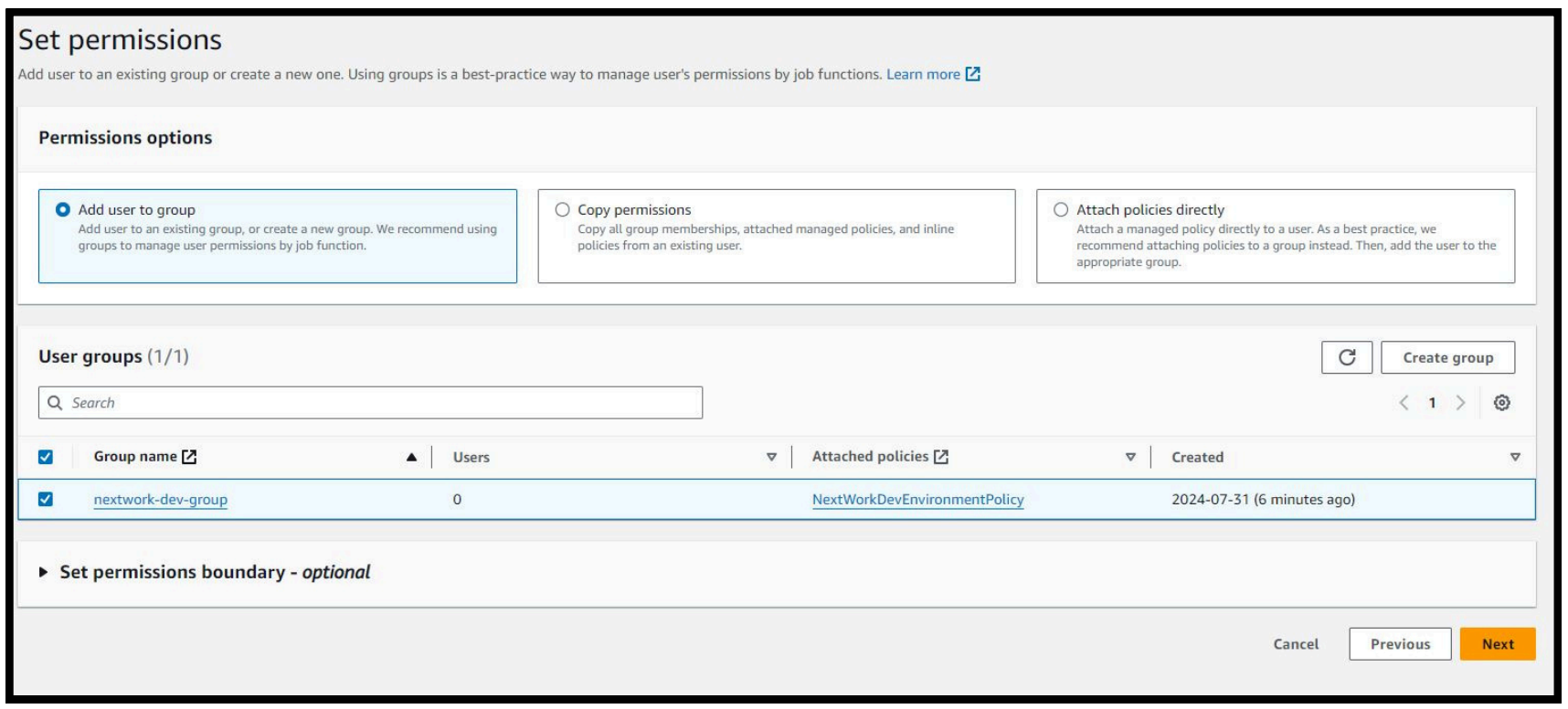




Aiman Haziq
NextWork Student

Step by Step Create an AWS Account Alias

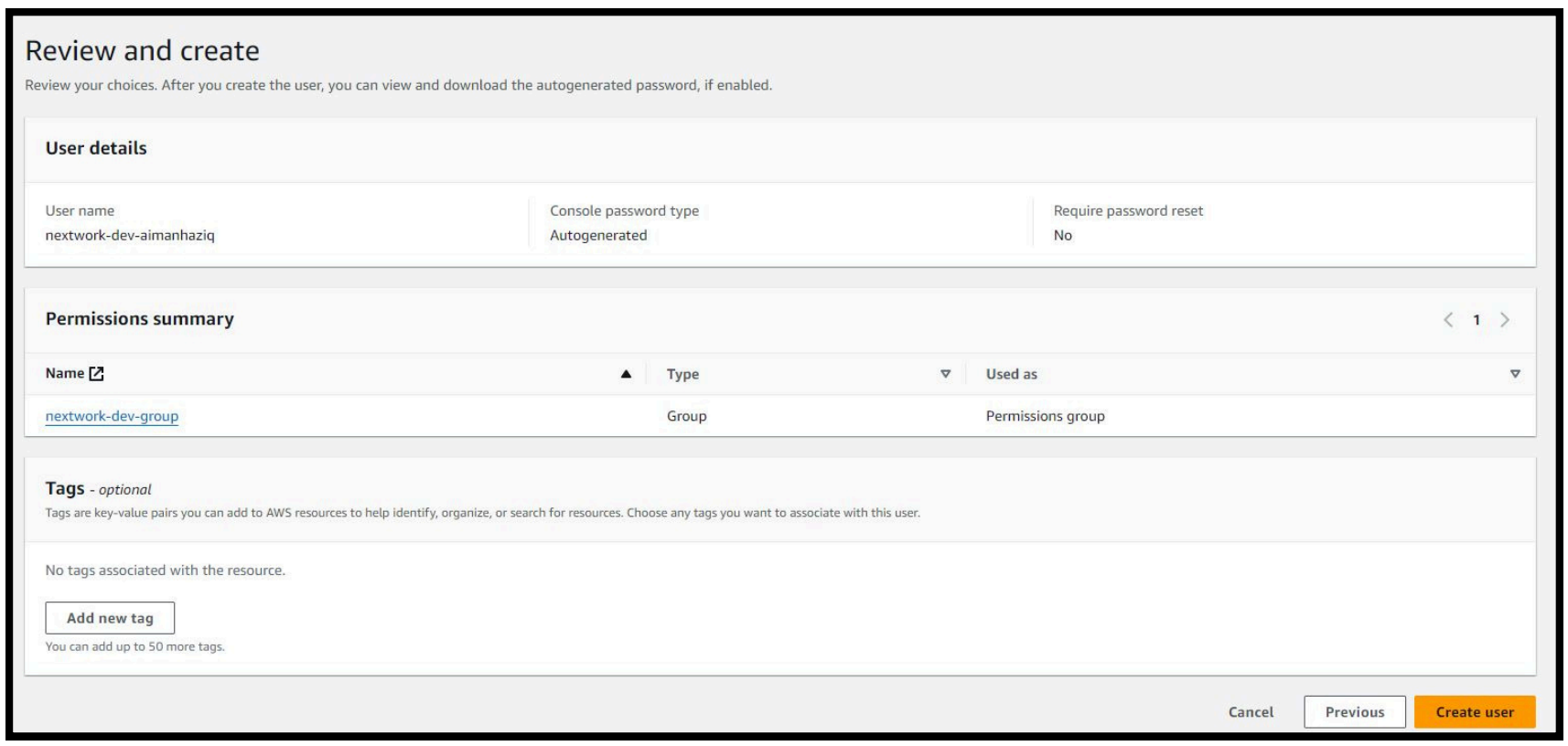


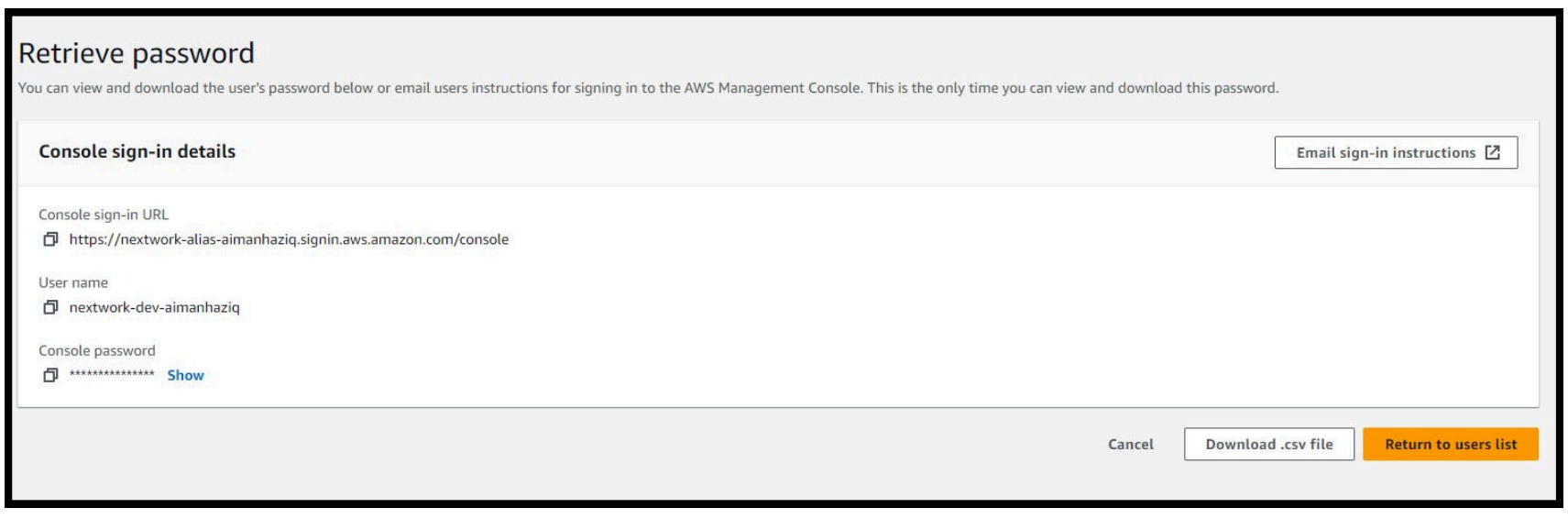


Aiman Haziq
NextWork Student

NextWork.org

Step by Step Create an AWS Account Alias



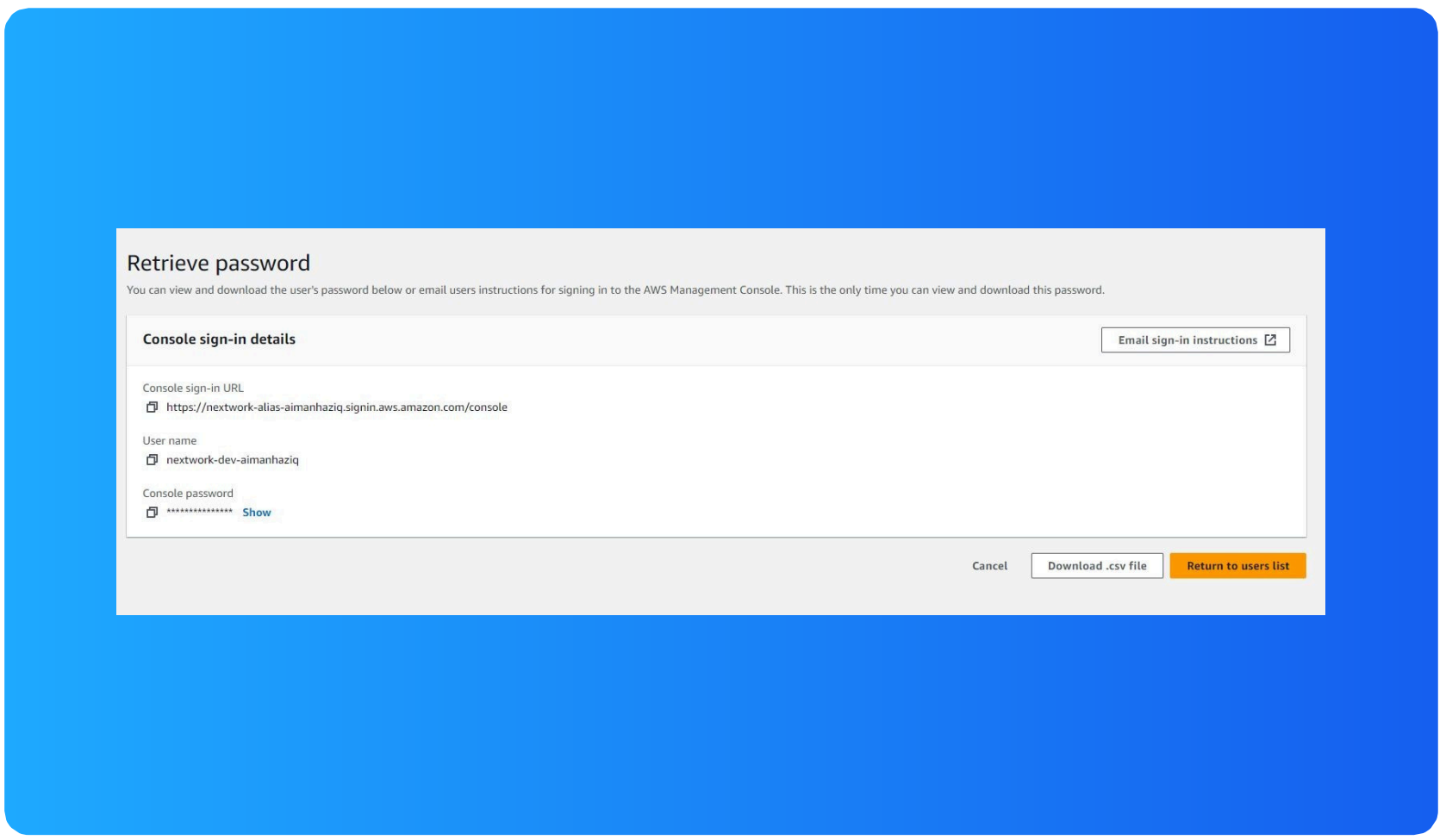


Aiman Haziq
NextWork Student

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.



Aiman Haziq
NextWork Student
NextWork.org

Step by Step Test Your User's Access

Objective

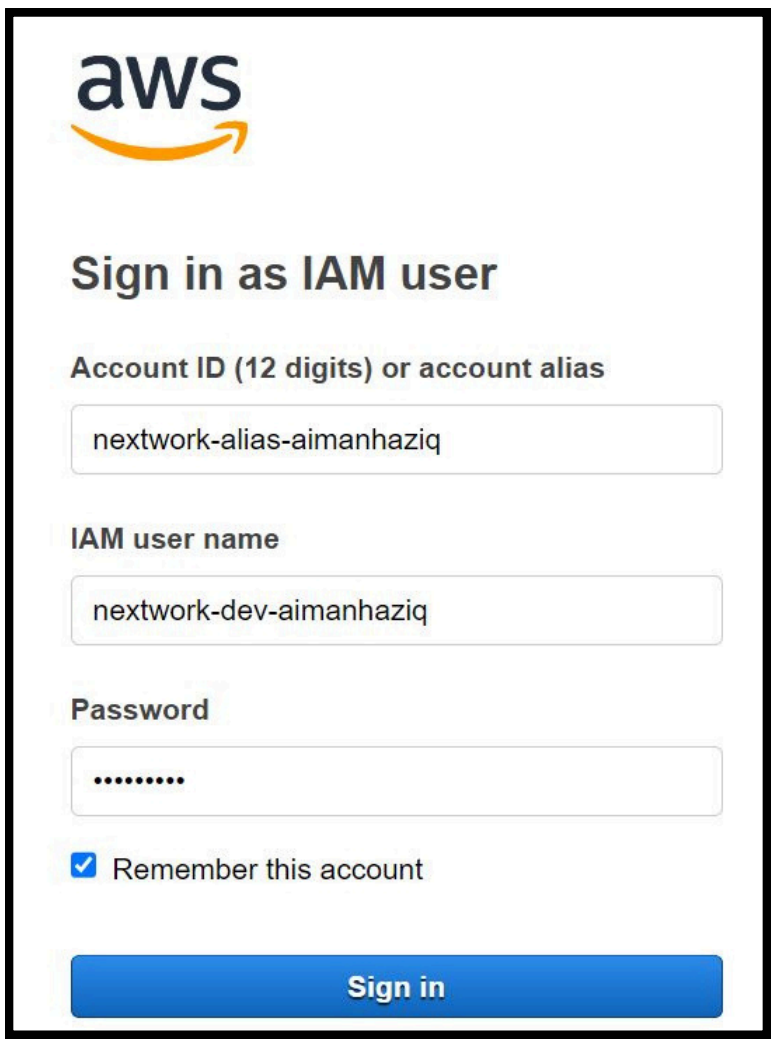
Verify that the IAM policy is working as intended by testing user permissions.

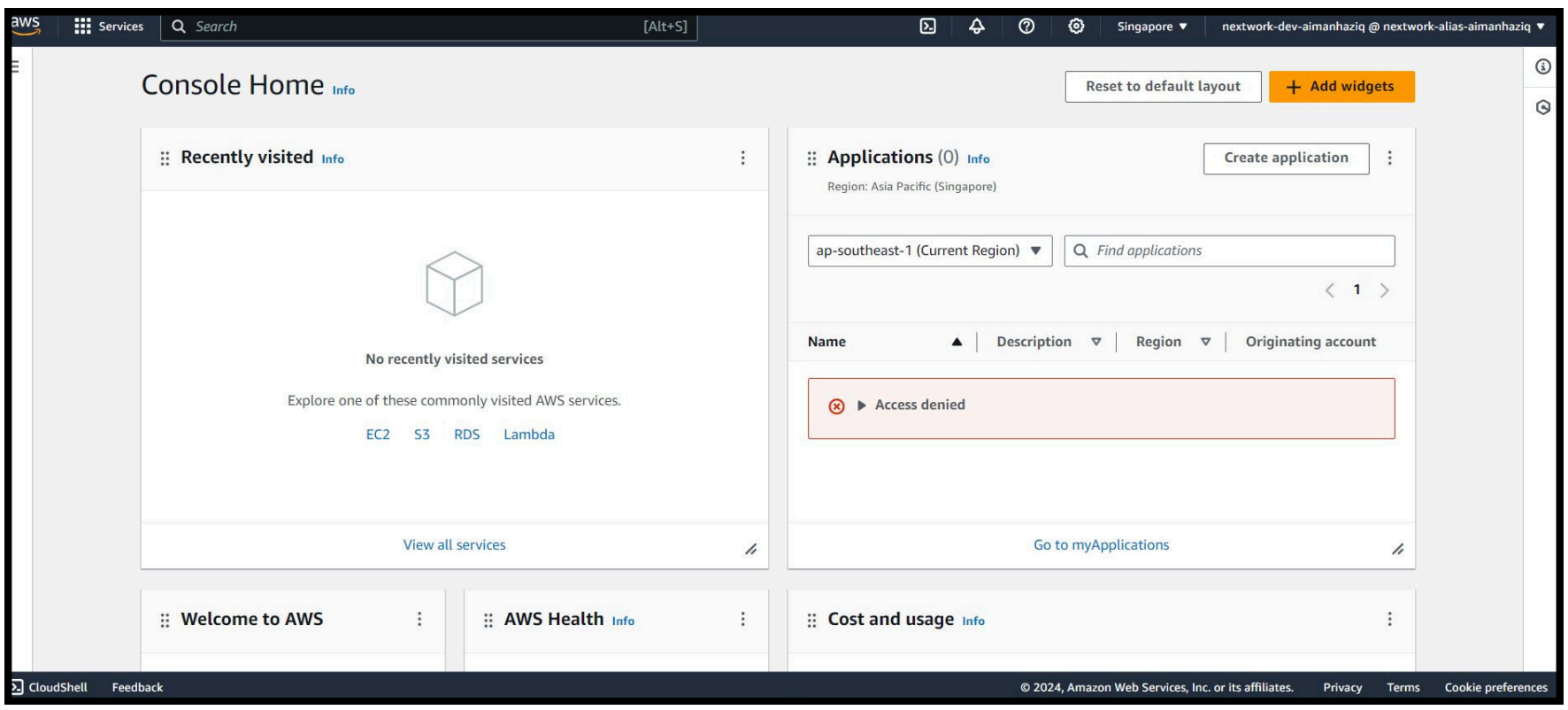
Instructions

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.

Aiman Haziq
NextWork Student

Step by Step Test Your User's Access





Aiman Haziq
NextWork Student

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.

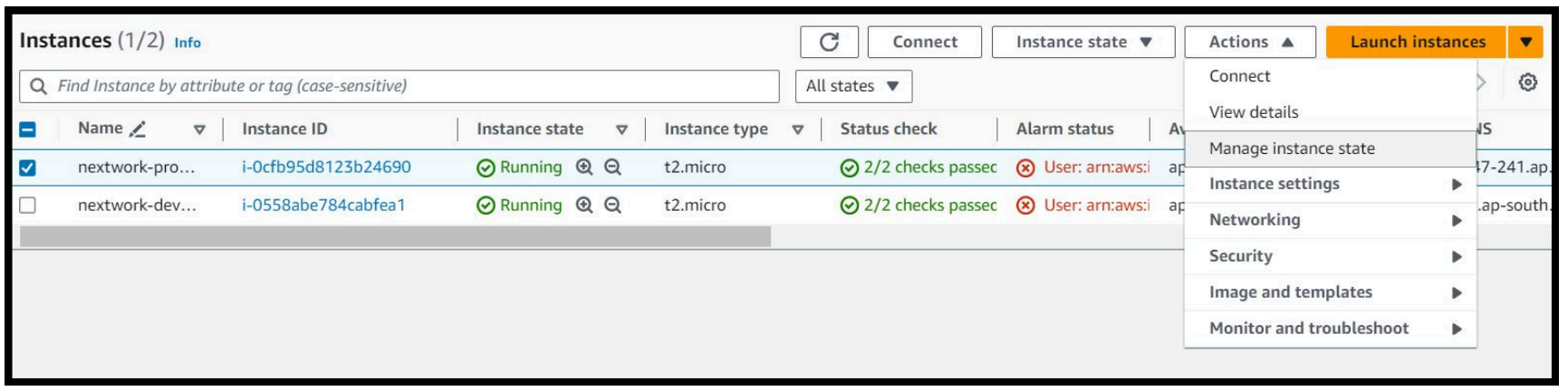
Stopping the production instance

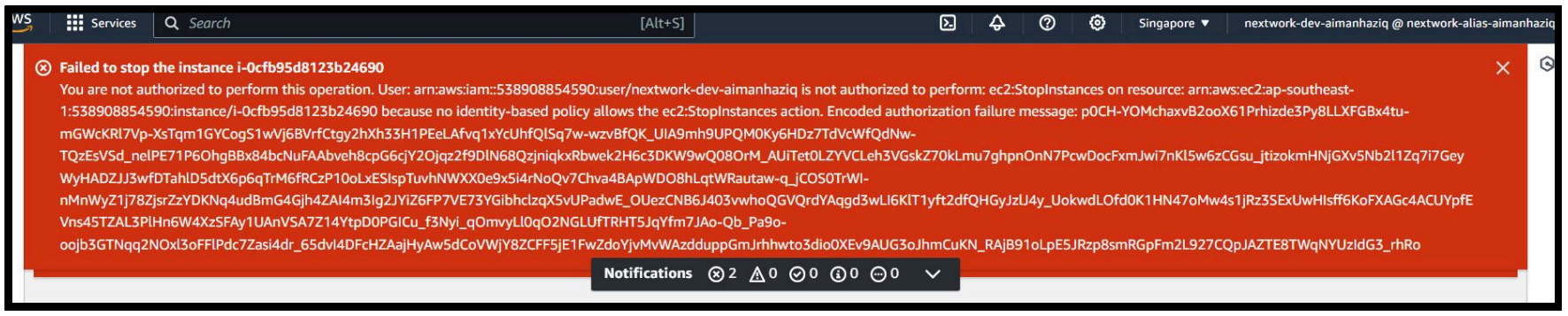
When I tried to stop the production instance, I received an authorization error, indicating that my IAM policy correctly restricted access to this action.

Aiman Haziq
NextWork Student

NextWork.org

Testing IAM Policies



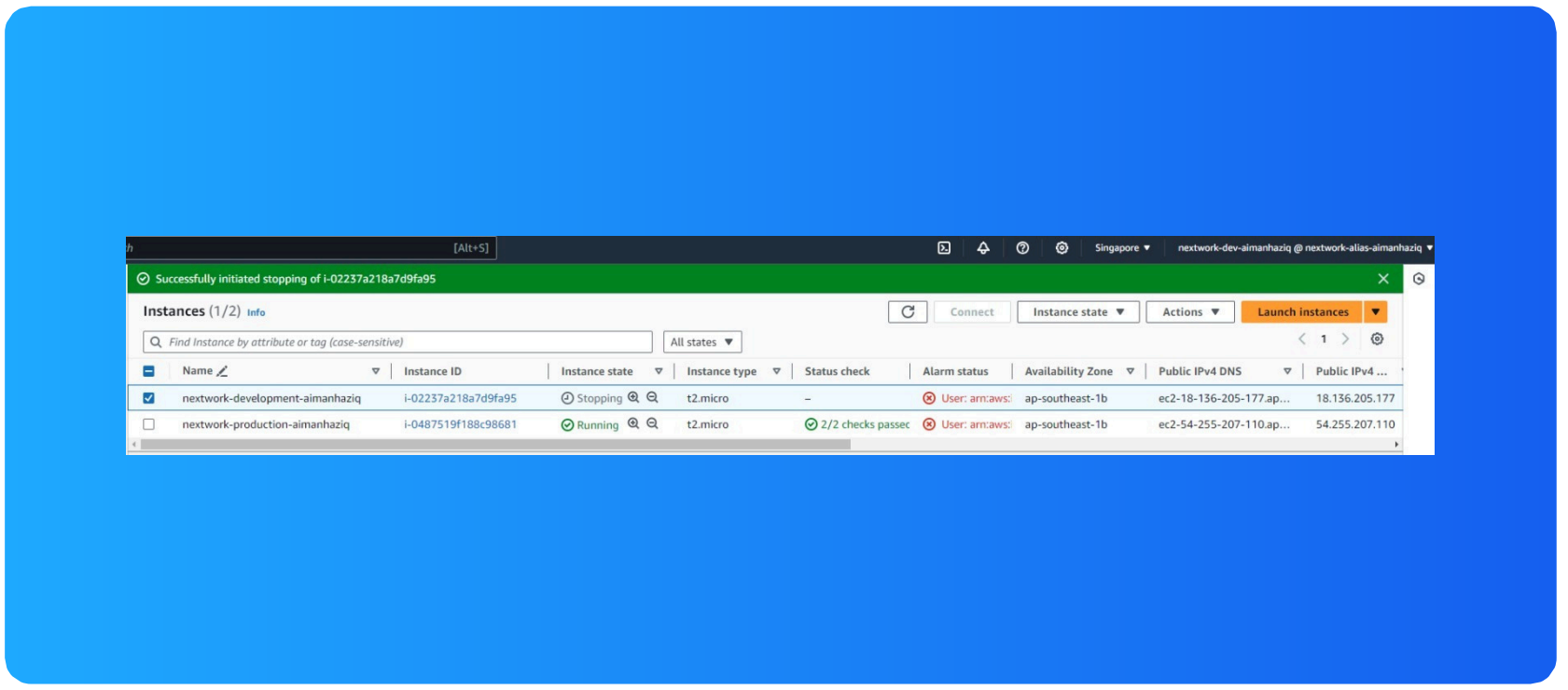


Aiman Haziq
NextWork Student

Testing IAM Policies

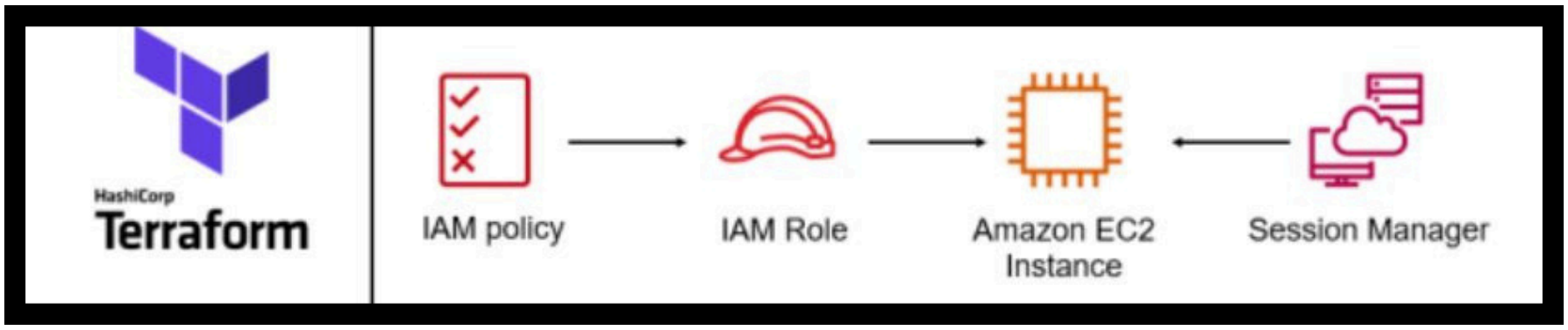
Stopping the development instance

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



NextWork.org

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



Aiman Haziq
NextWork Student

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
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 20.0M  100 20.0M    0     0  5144k      0  0:00:03  0:00:03 --:--:-- 5144k
```

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

```
cisco@labvm:~$ sudo mv terraform /usr/local/bin/
[sudo] password for cisco:
```

Aiman Haziq
NextWork Student

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]
```

Aiman Haziq
NextWork Student

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:

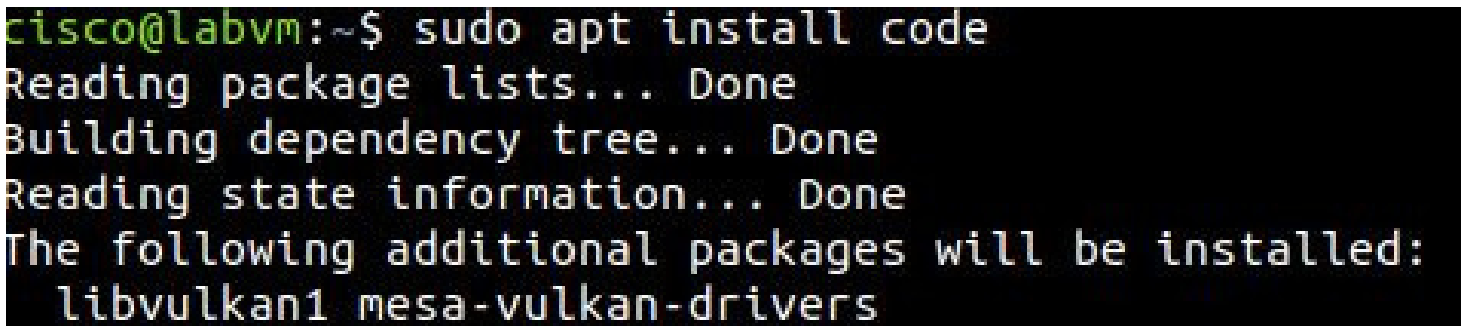
```
cisco@labvm:~$ sudo apt-get install -y software-properties-common apt-transport-https wget
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-software-properties software-properties-gtk
Recommended 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
```

Aiman Haziq
NextWork Student

Setting Up AWS Infrastructure with Terraform on Linux



Aiman Haziq
NextWork Student

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):

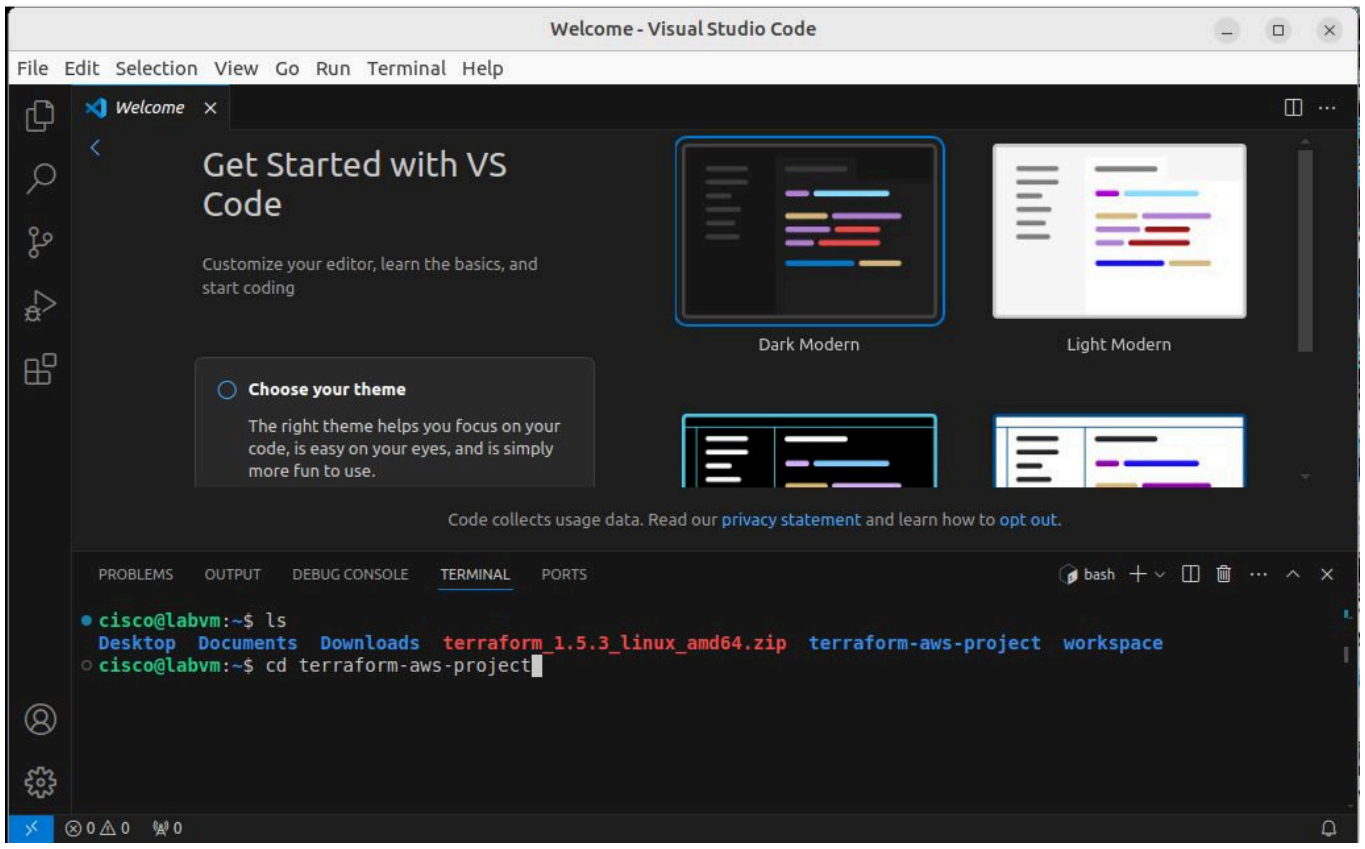
```
cisco@labvm:~$ aws configure
AWS Access Key ID [None]: AKIAX26LYRE7HI6FNWVW
AWS Secret Access Key [None]: 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
Default region name [None]: ap-southeast-1
Default output format [None]: json
```

Aiman Haziq
NextWork Student

Setting Up AWS Infrastructure with Terraform on Linux

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

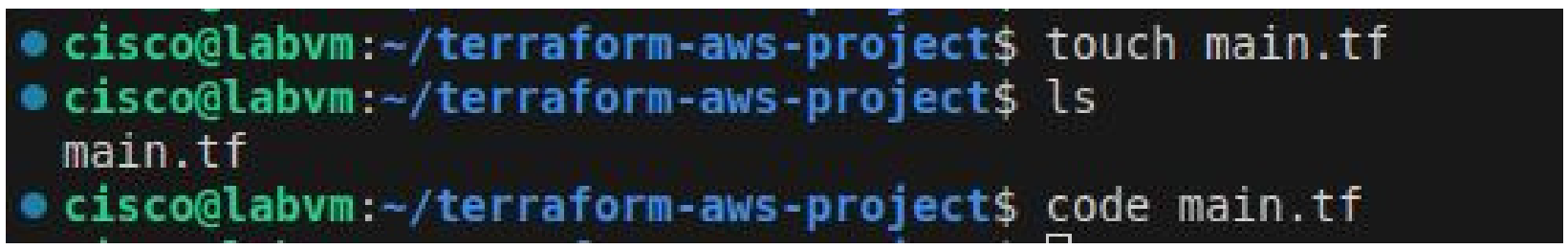
1. Open Visual Studio Code and create a new directory for your project. Open the terminal within VS Code (Ctrl + or Terminal > New Terminal). Use **mkdir terraform-aws-project** and **cd terraform-aws-project**

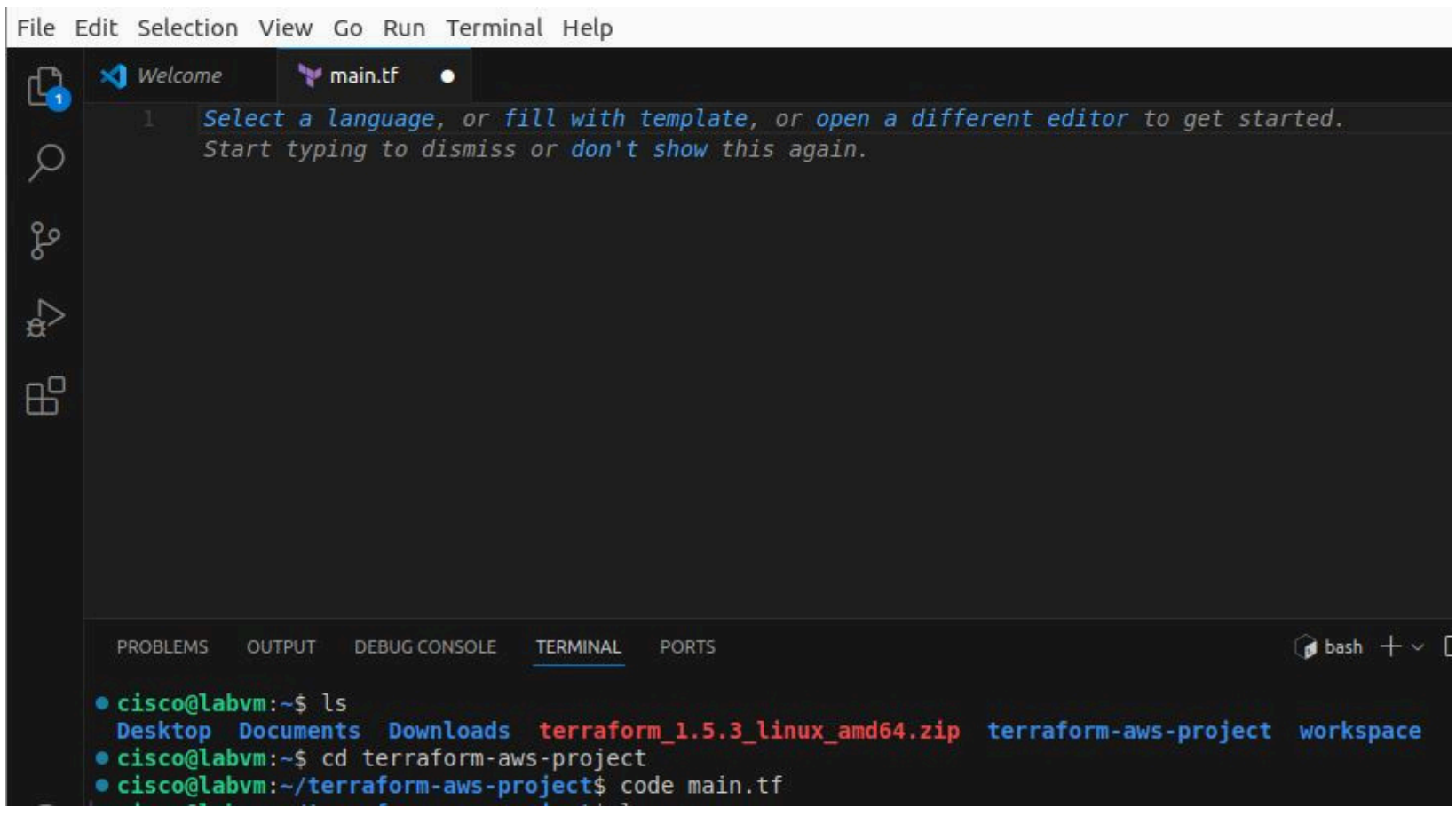


Aiman Haziq
NextWork Student

Setting Up AWS Infrastructure with Terraform on Linux

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





Aiman Haziq
NextWork Student

Setting Up AWS Infrastructure with Terraform on Linux

Step 4: Write Terraform Configuration

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

main.tf

home > cisco > terraform-aws-project > main.tf

```
provider "aws" {
  region = "ap-southeast-1"
}

resource "aws_instance" "production_instance" {
  ami           = "ami-012c2e8e24e2ae21d"
  instance_type = "t2.micro"
  tags = {
    Name = "nextwork-production-aimanhaziq"
    Env  = "production"
  }
}

resource "aws_instance" "development_instance" {
  ami           = "ami-012c2e8e24e2ae21d"
  instance_type = "t2.micro"
  tags = {
    Name = "nextwork-development-aimanhaziq"
    Env  = "development"
  }
}

resource "aws_iam_policy" "nextwork_dev_environment_policy" {
  name        = "NextWorkDevEnvironmentPolicy"
  description = "IAM Policy for NextWork's development environment"
  policy      = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances",
        "ec2:RebootInstances"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
```

main.tf

home > cisco > terraform-aws-project > main.tf

```
{
  "Statement": [
    {
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "ec2:ResourceTag/Env": "development"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeTags"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": [
        "ec2:TerminateInstances",
        "ec2:DeleteTags",
        "ec2:CreateTags"
      ],
      "Resource": "*"
    }
  ]
}
EOF
}
```

Aiman Haziq
NextWork Student

Setting Up AWS Infrastructure with Terraform on Linux

Step 4: Write Terraform Configuration

e Edit Selection View Go Run Terminal Help

main.tf

home > cisco > terraform-aws-project > main.tf

```
}

resource "aws_iam_group" "nextwork_dev_group" {
  name = "nextwork-dev-group"
}

resource "aws_iam_group_policy_attachment" "attach_policy_to_group" {
  group      = aws_iam_group.nextwork_dev_group.name
  policy_arn = aws_iam_policy.nextwork_dev_environment_policy.arn
}

resource "aws_iam_user" "nextwork_dev_user" {
  name = "nextwork-dev-aimanhaziq"
}

resource "aws_iam_user_group_membership" "add_user_to_group" {
  user   = aws_iam_user.nextwork_dev_user.name
  groups = [
    aws_iam_group.nextwork_dev_group.name
  ]
}

resource "aws_iam_user_login_profile" "login_profile" {
  user = aws_iam_user.nextwork_dev_user.name
  password_reset_required = false
}

```

Aiman Haziq
NextWork Student

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:

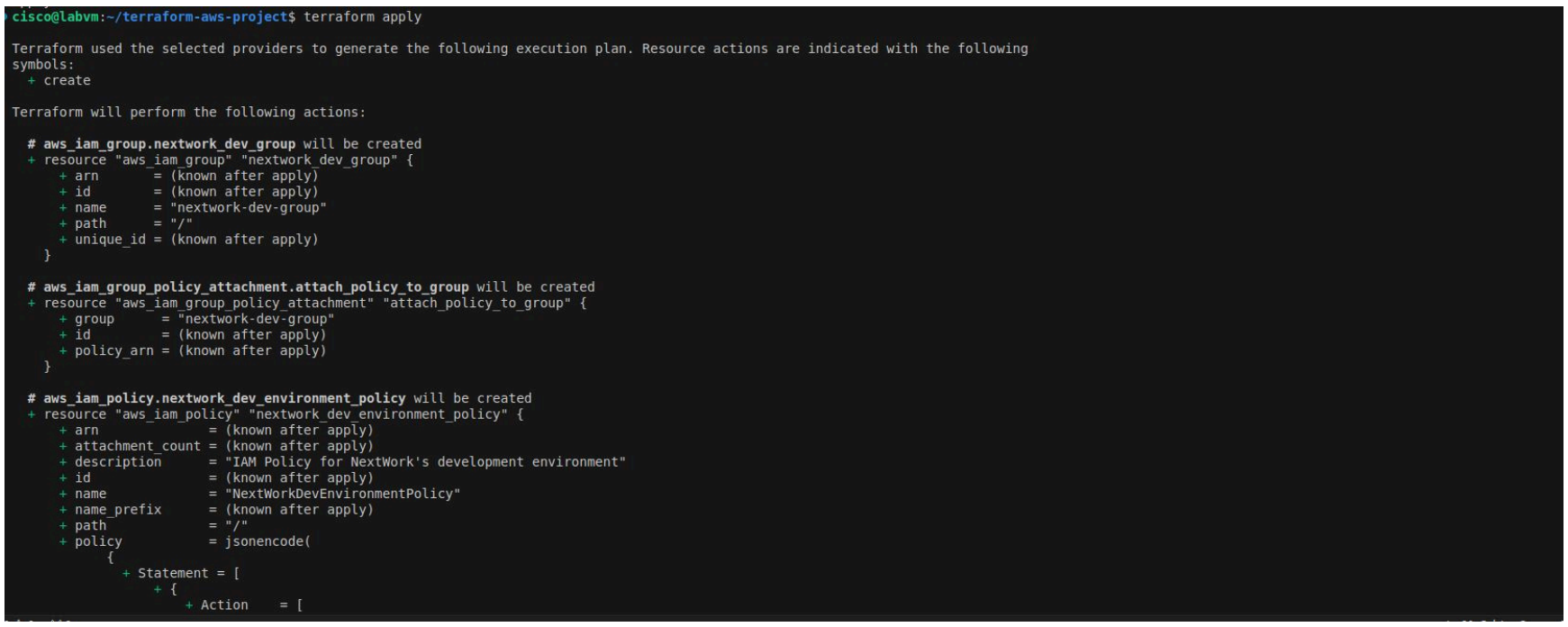
```
cisco@labvm:~/terraform-aws-project$ terraform validate

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:     {
```

Aiman Haziq
NextWork Student

Setting Up AWS Infrastructure with Terraform on Linux

Step 5: Initialize and Apply Terraform Configuration

3.Apply the Configuration:



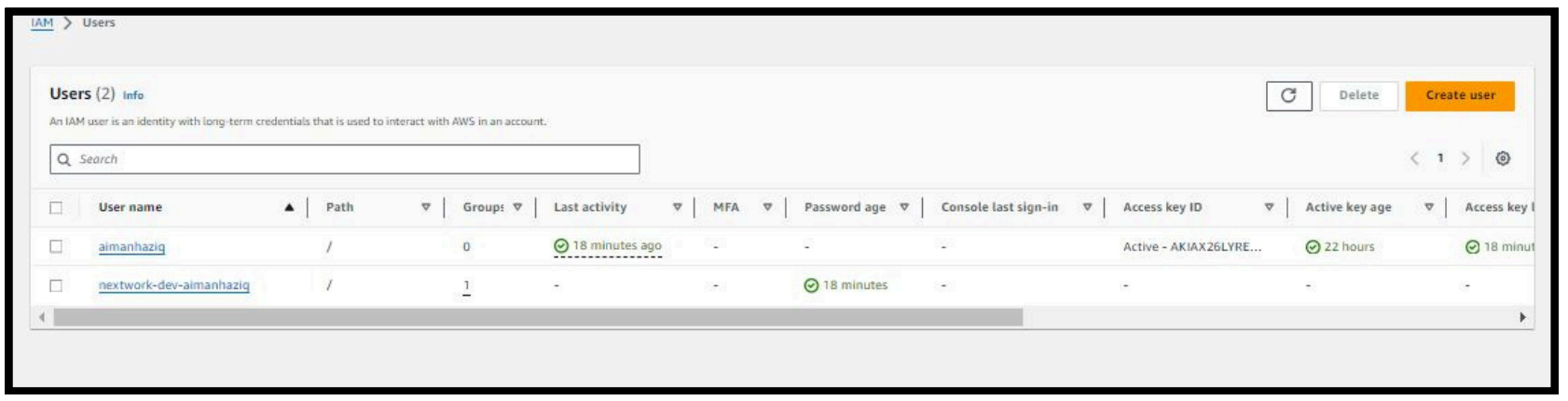
Terraform will show a plan of the actions it will take. Type yes to approve and apply the changes.

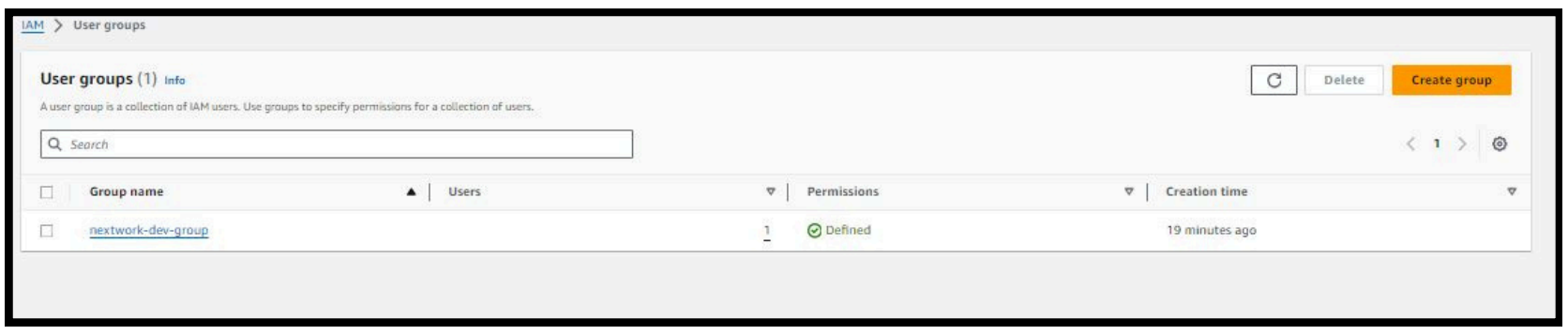
Aiman Haziq
NextWork Student

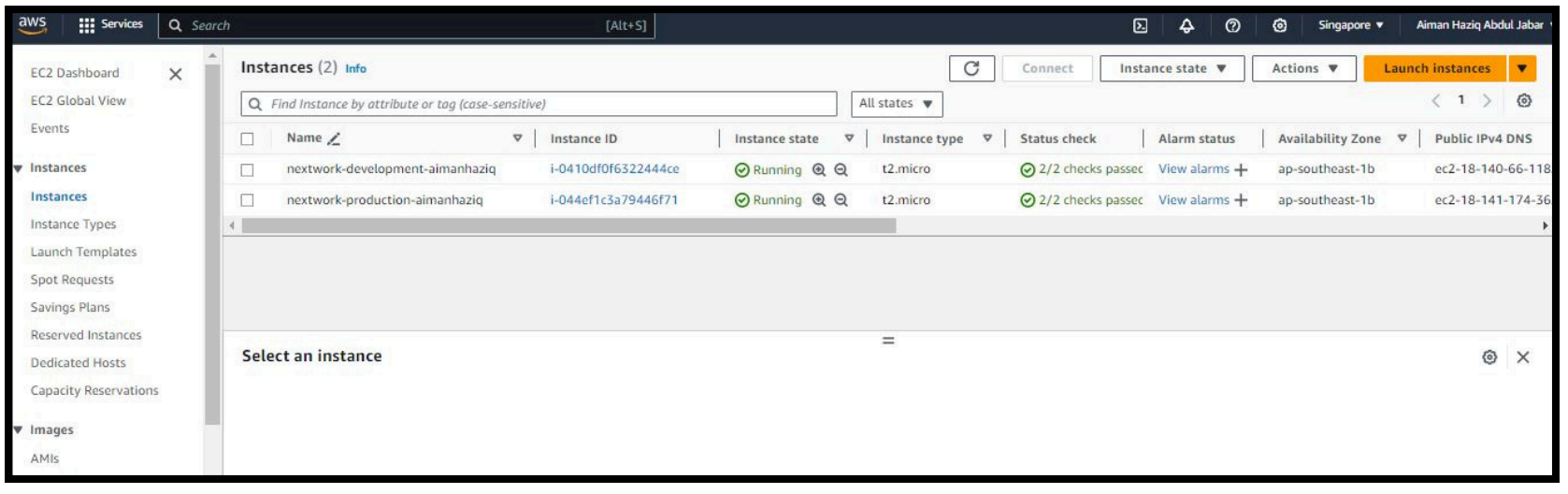
Setting Up AWS Infrastructure with Terraform on Linux

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.







Aiman Haziq
NextWork Student

Setting Up AWS Infrastructure with Terraform on Linux

Step 7: Delete all resources

.1. Use **terraform destory** 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-0410df0f6322444ce]
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 for more projects. Thank you Nextwork for providing this project.