Lesson 11 Demo 01

Implementing Workflow for Managing Configurations in Terraform

Objective: To implement a workflow for managing AWS infrastructure configurations in Terraform efficiently and effectively

Tools required: Terraform, AWS, and Visual Studio Code

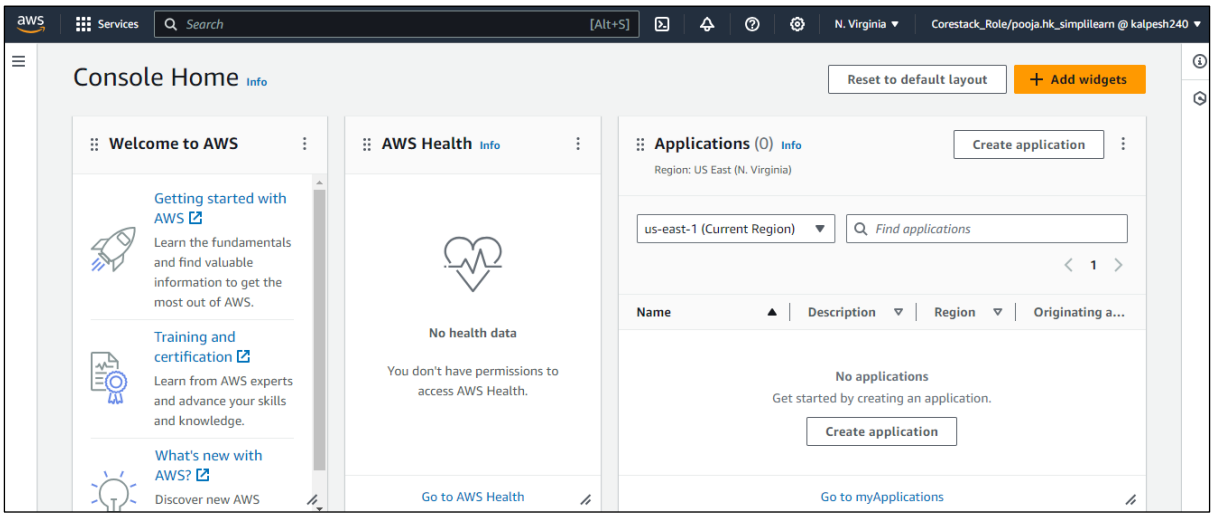
Prerequisites: None

Steps to be followed:

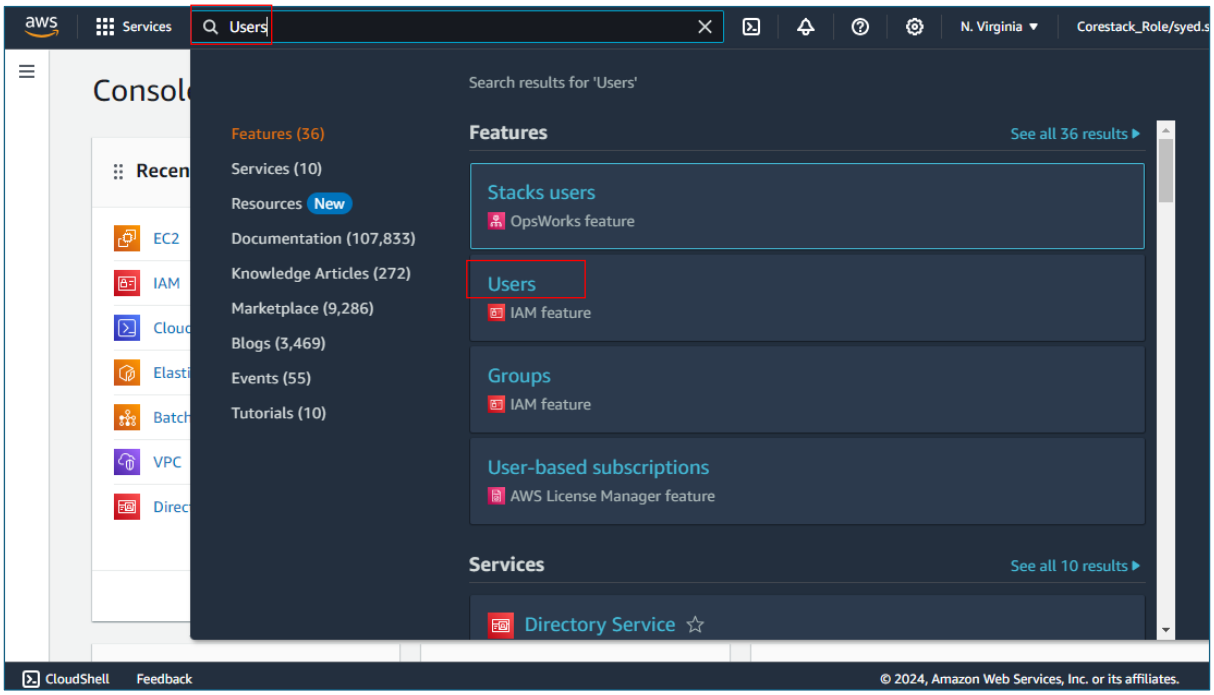
1. Create an IAM user
2. Create access and secret key
3. Implement a workflow for managing configurations in Terraform

Step 1: Create an IAM user

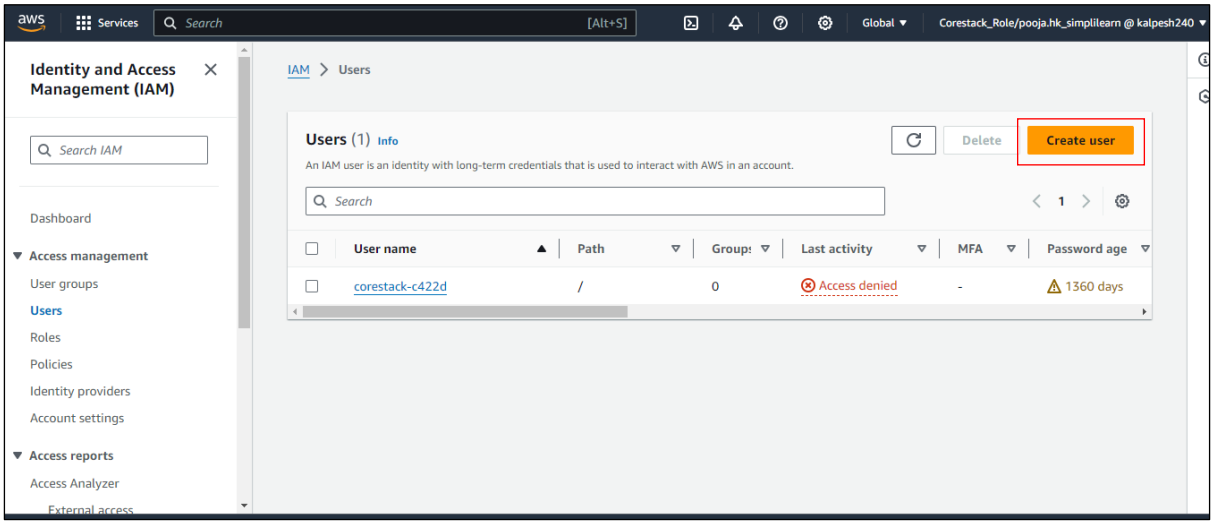
1.1 Navigate to the AWS console as shown in the screenshot below:



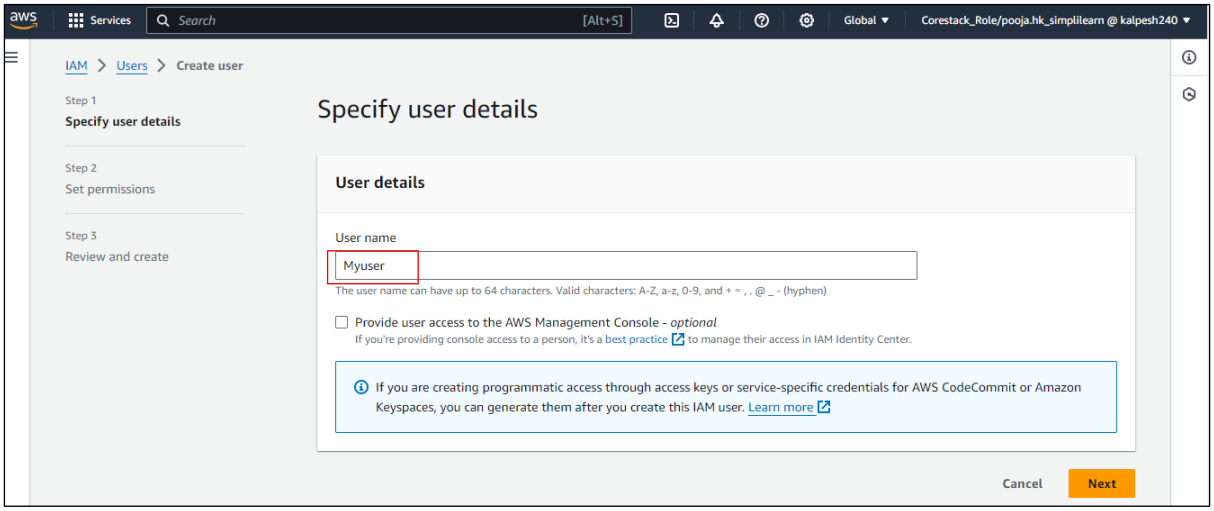
1.2 Search for and click on **Users** as shown in the screenshot below:



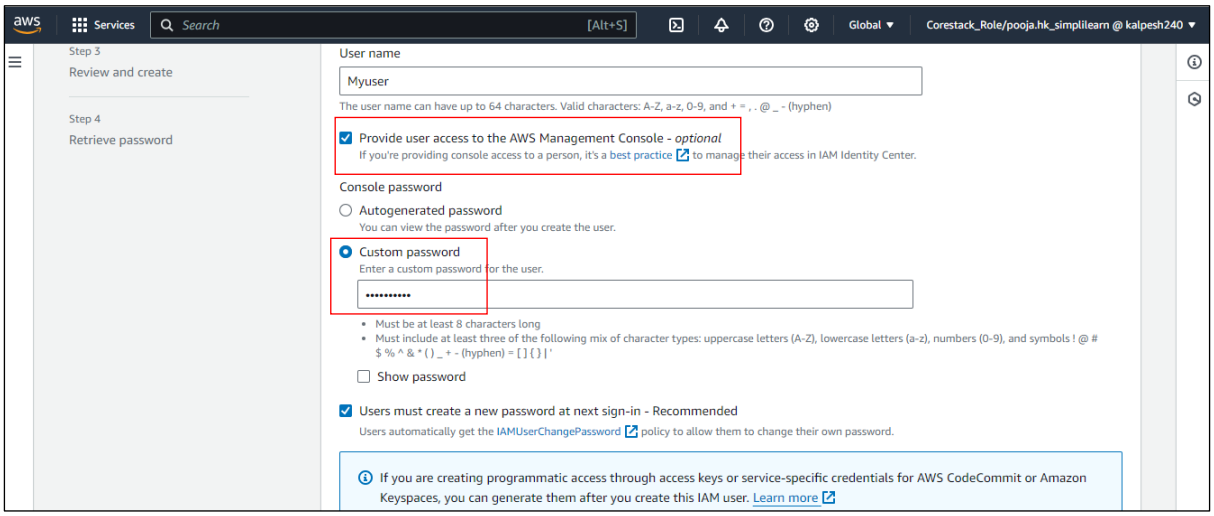
1.3 Click on **Create user** as shown in the screenshot below:



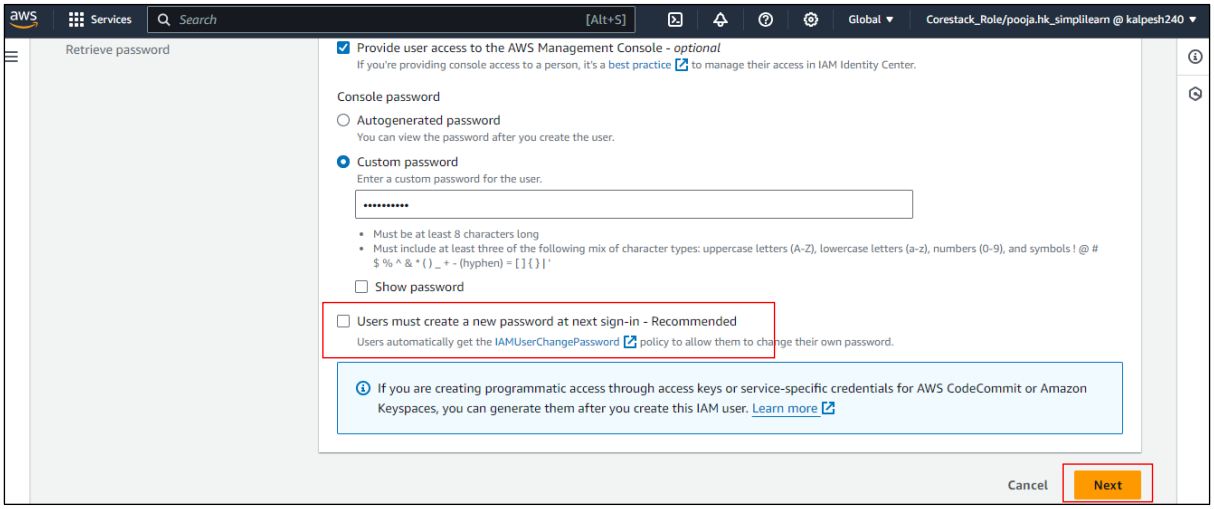
1.4 Enter a desired name in the **User name** field as shown in the screenshot below:



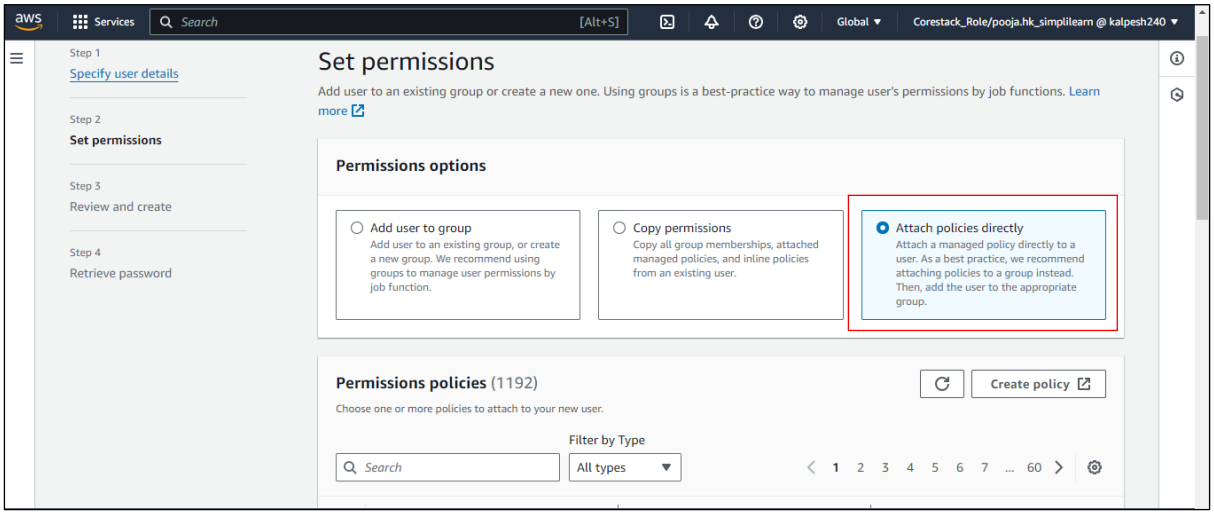
1.5 Click on the check box of the user access, select **Custom password**, and enter the password as shown in the screenshot below:



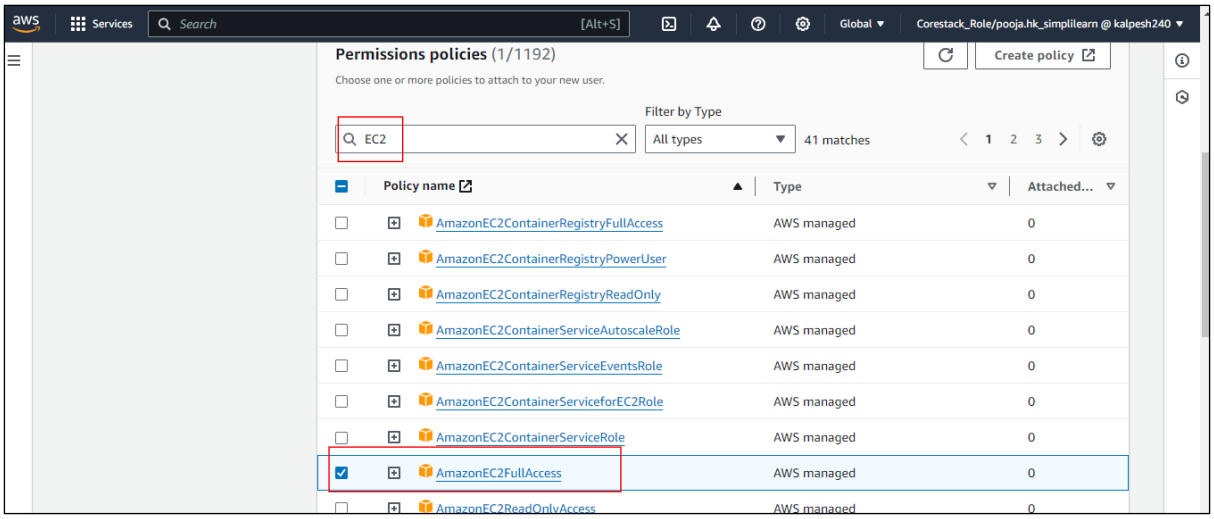
1.6 Untick the check box and then click on **Next** as shown in the screenshot below:



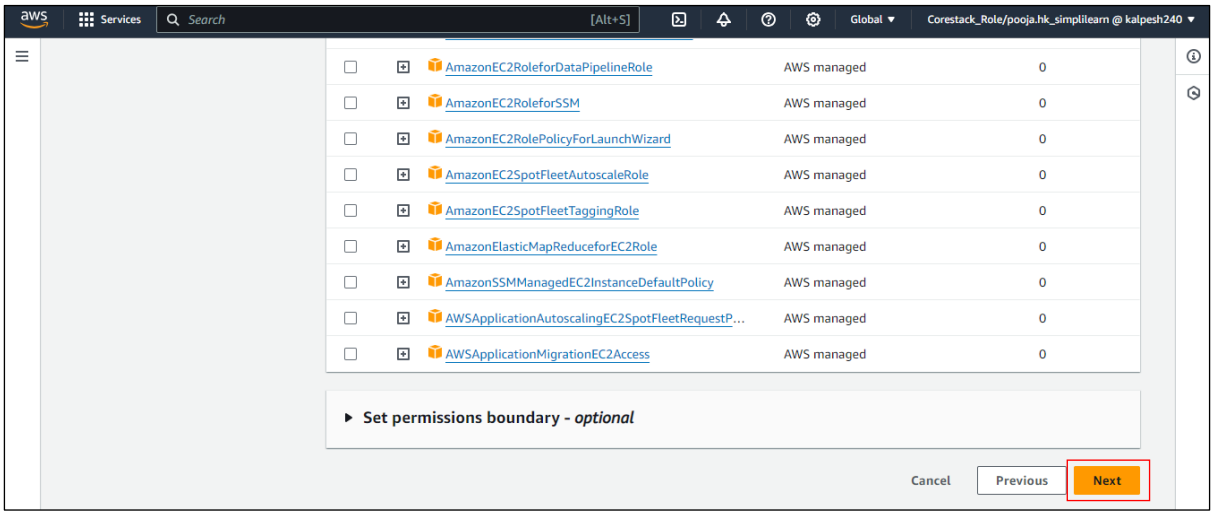
1.7 Click on **Attach policies directly** as shown in the screenshot below:



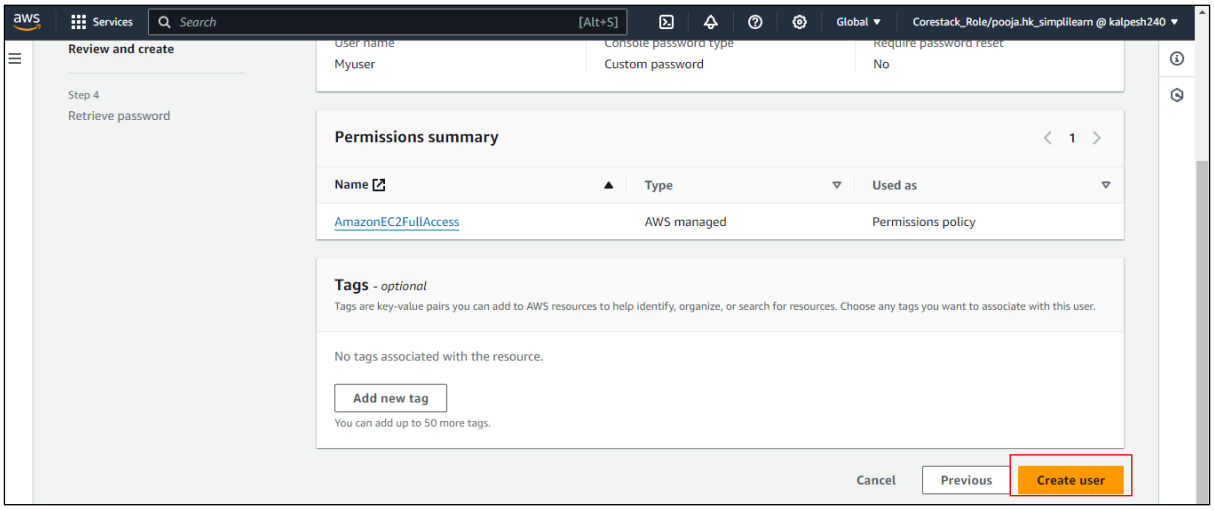
1.8 Search for EC2 in the **Permissions policies** section and select **AmazonEC2FullAccess** as shown in the screenshot below:



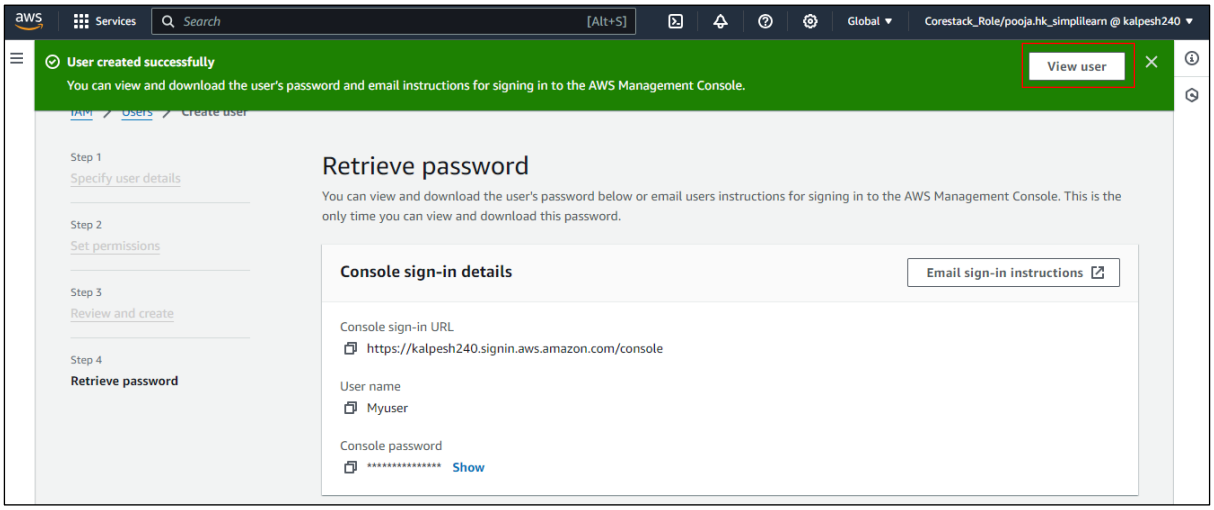
1.9 Scroll down and click on the **Next** button as shown in the screenshot below:



1.10 Click on **Create user** as shown in the screenshot below:

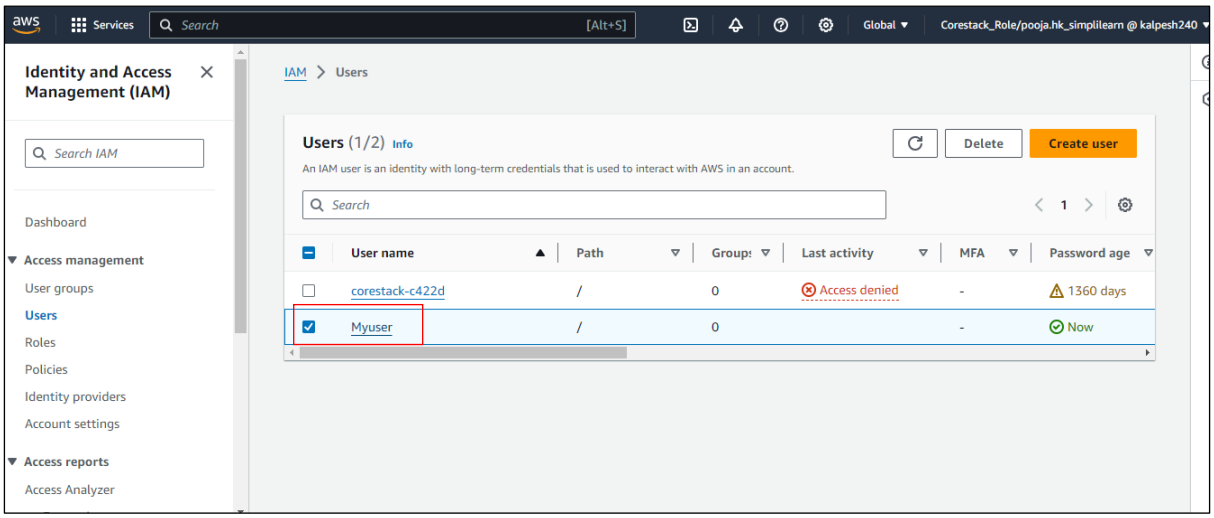


1.11 Click on **View user** as shown in the screenshot below:

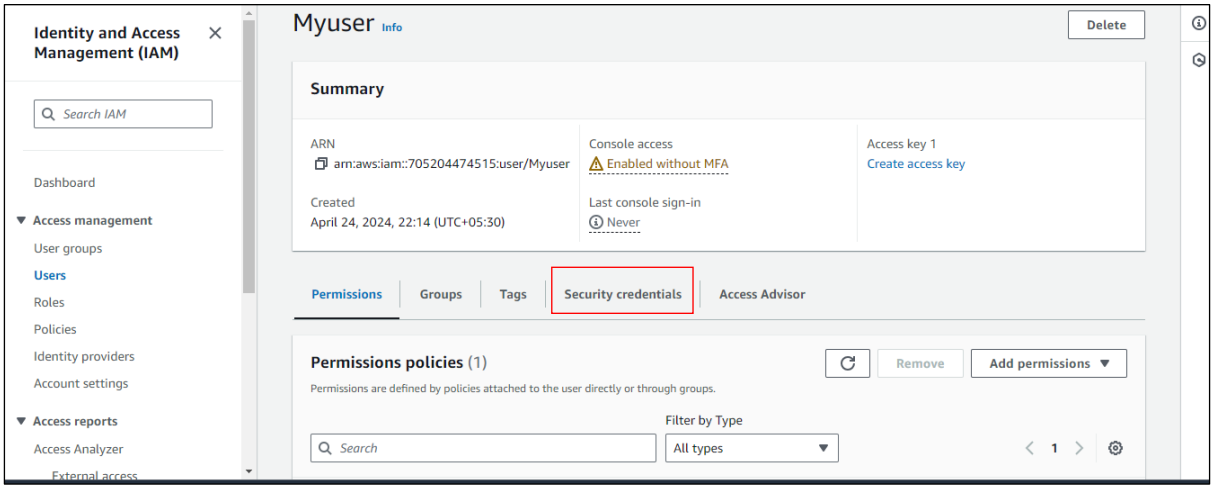


Step 2: Create access and secret key

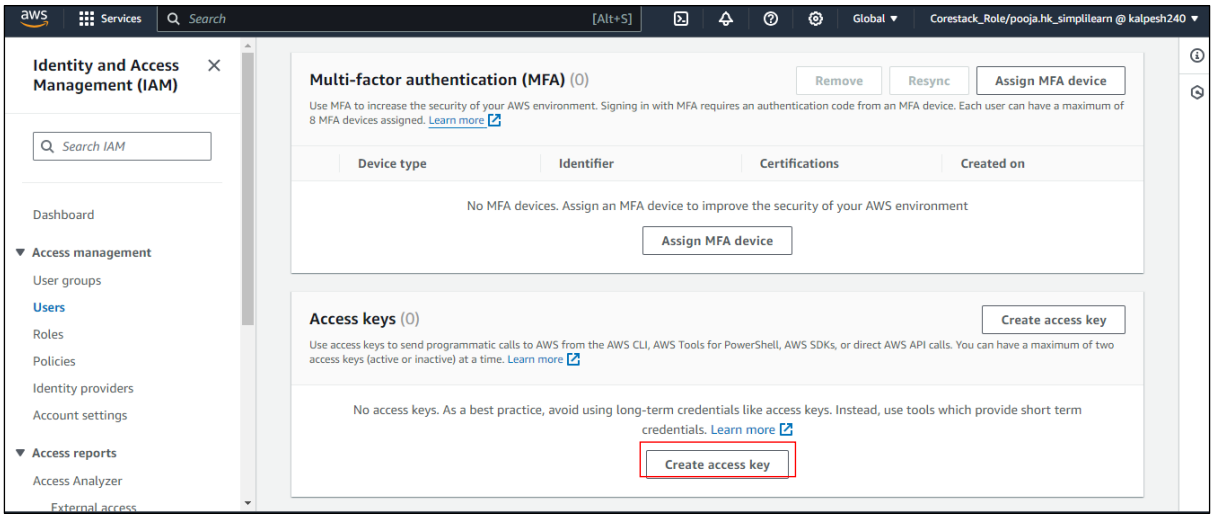
2.1 Click on the username that you created as shown in the screenshot below:



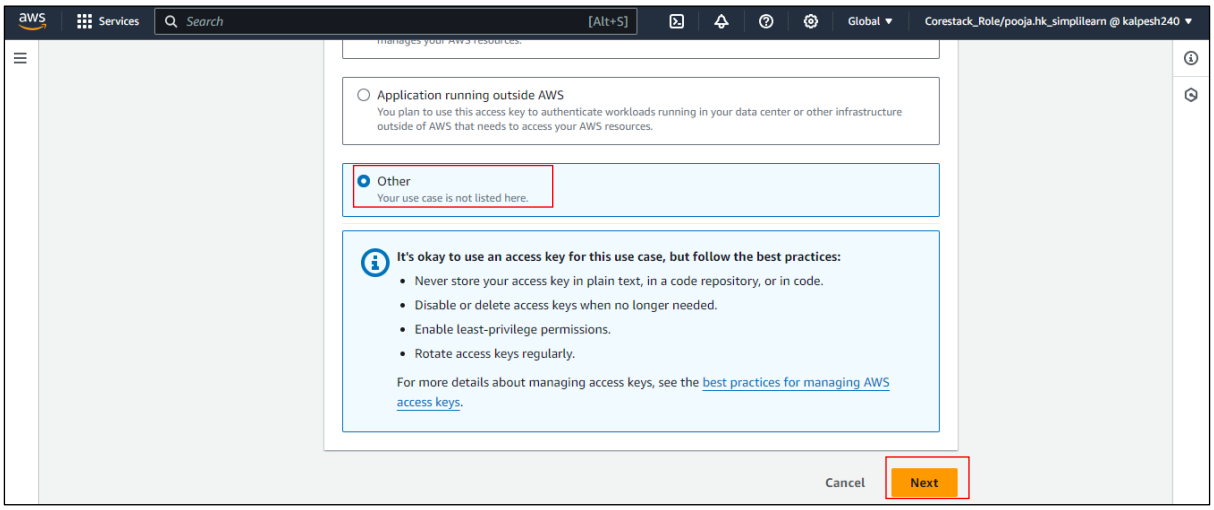
2.2 Click on **Security credentials** as shown in the screenshot below:



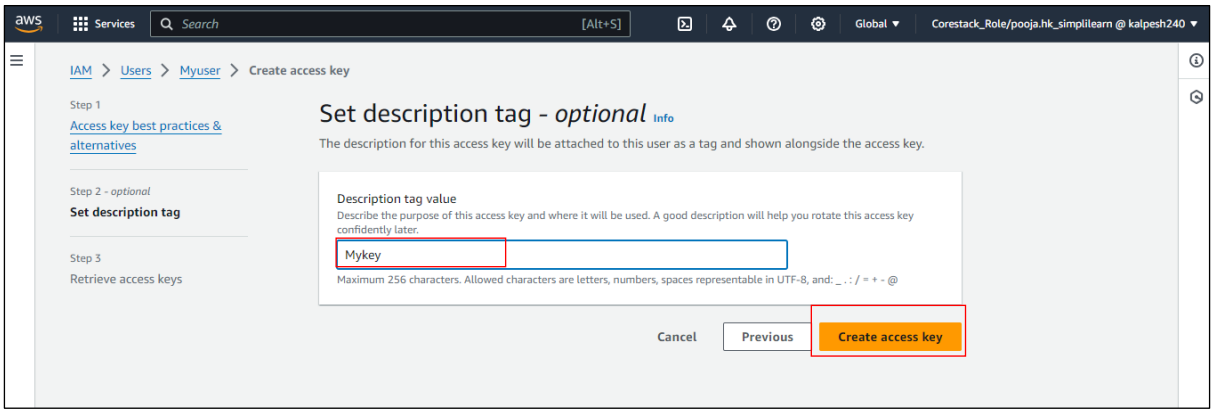
2.3 Scroll down to **Access keys** section and then click on **Create access key** as shown in the screenshot below:



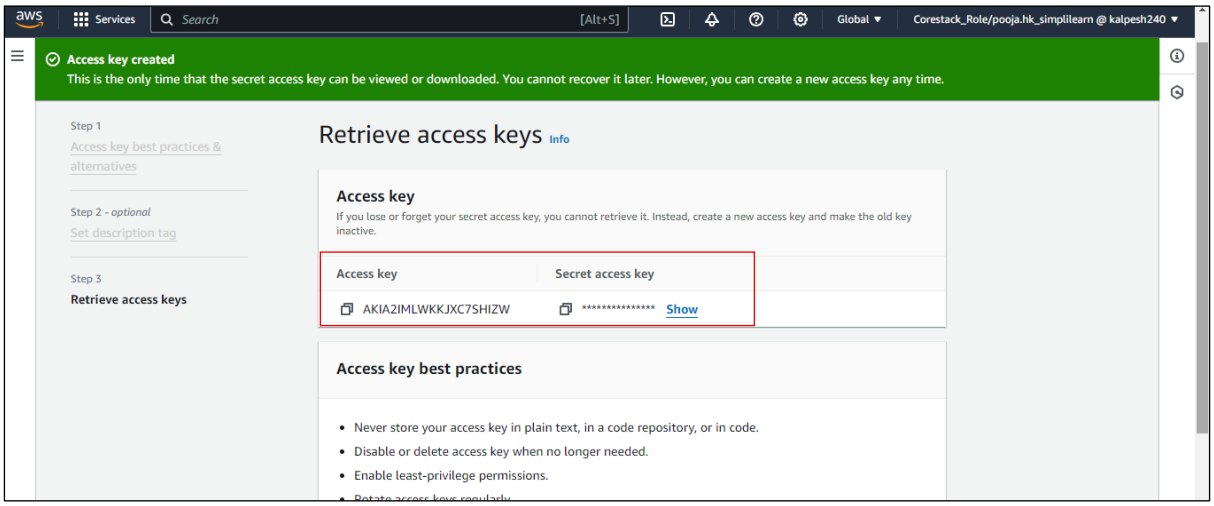
2.4 Select **Other** from the listed options and the click on **Next** as shown in the screenshot below:



2.5 Enter a desired name in the **Description tag value** section and then click on **Create access key** as shown in the screenshot below:



2.6 Copy the **Access key** and **Secret access key** and save it in Notepad for the next steps as shown in the screenshot below:

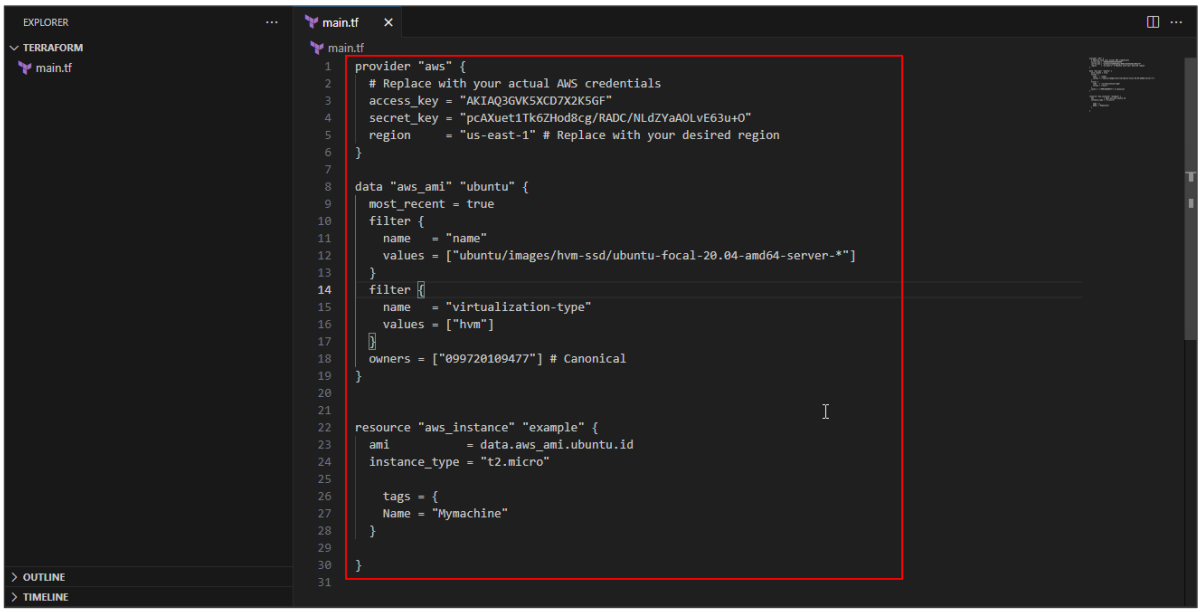


Note: Save the Access key and Secret access key in notepad to use in next steps

Step 3: Implement a workflow for managing configurations in Terraform

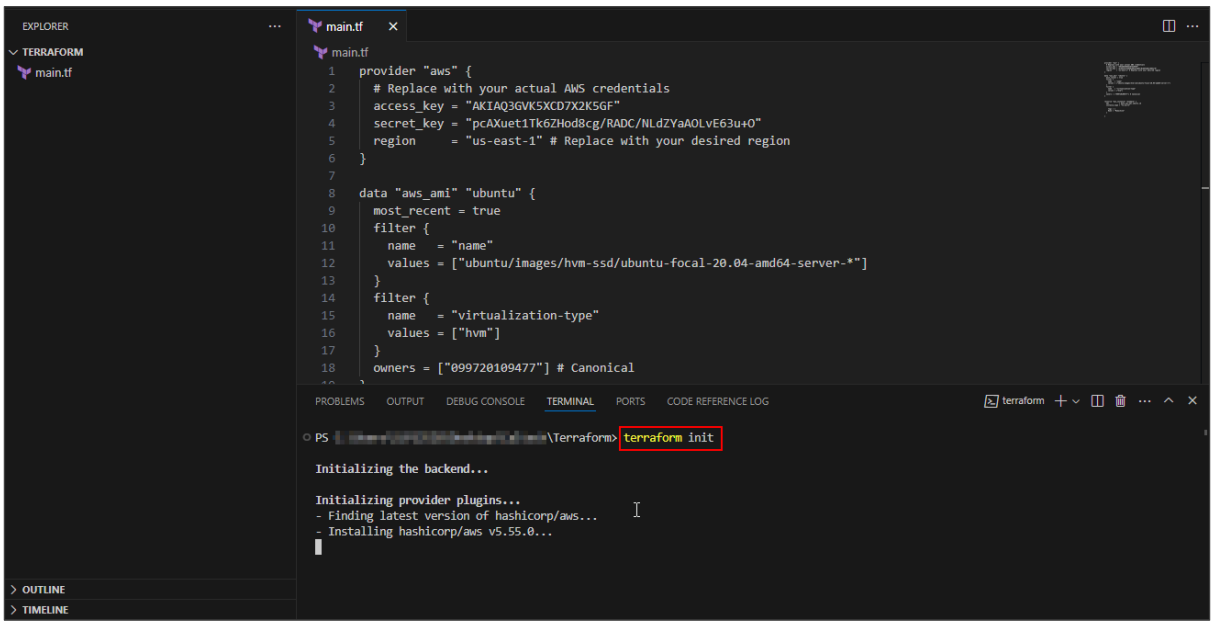
3.1 Open Visual Studio Code, create a file named **main.tf**, and add the following configuration code as shown in the screenshot below:

```
provider "aws" {
  # Replace with your actual AWS credentials
  access_key = “YOUR_ACCESS_KEY"
  secret_key = “YOUR_SECRET_KEY"
  region     = "us-east-1" # Replace with your desired region
}
data "aws_ami" "ubuntu" {
  most_recent = true
  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"]
  }
  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
  owners = ["099720109477"] # Canonical
}
resource "aws_instance" "example" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t2.micro"
    tags = {
    Name = "Mymachine"
  }
}
```

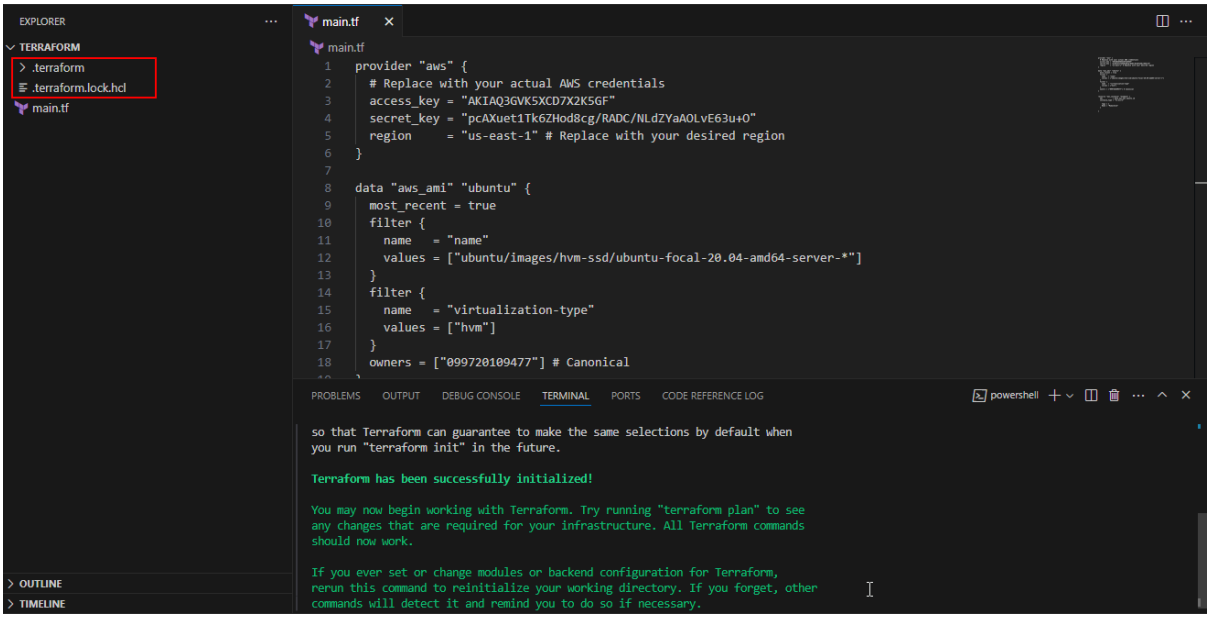


3.2 Initialize the configuration using the following command as shown in the screenshot below:

terraform init

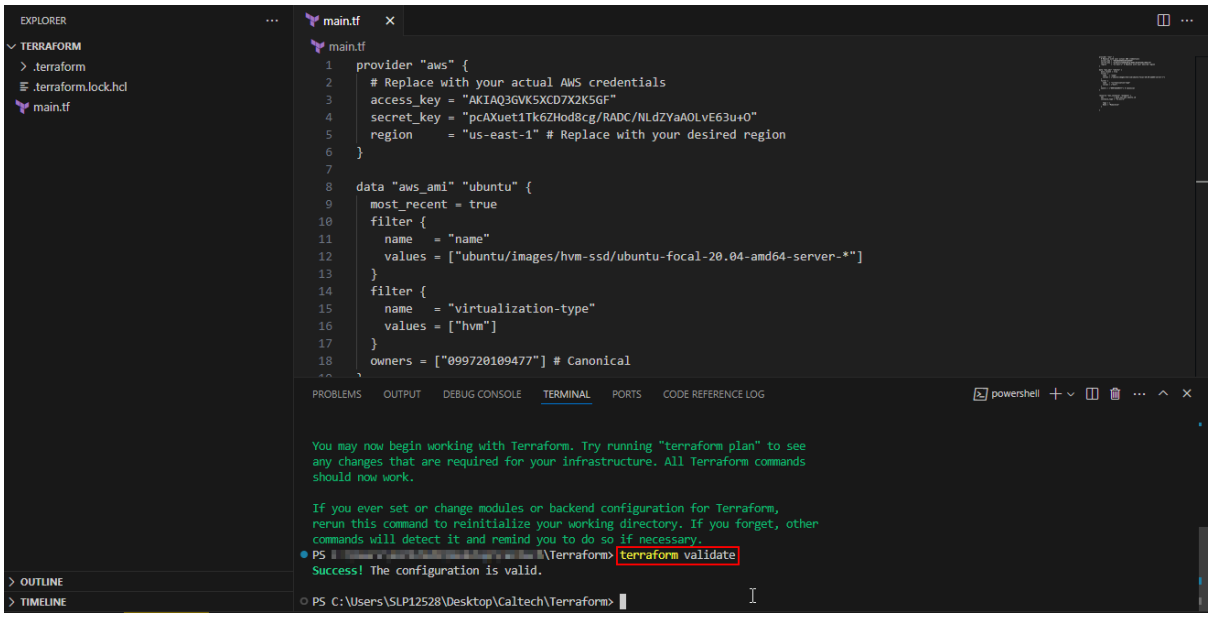


Once the Terraform has been initialized successfully, providers are downloaded as shown in the screenshot below:



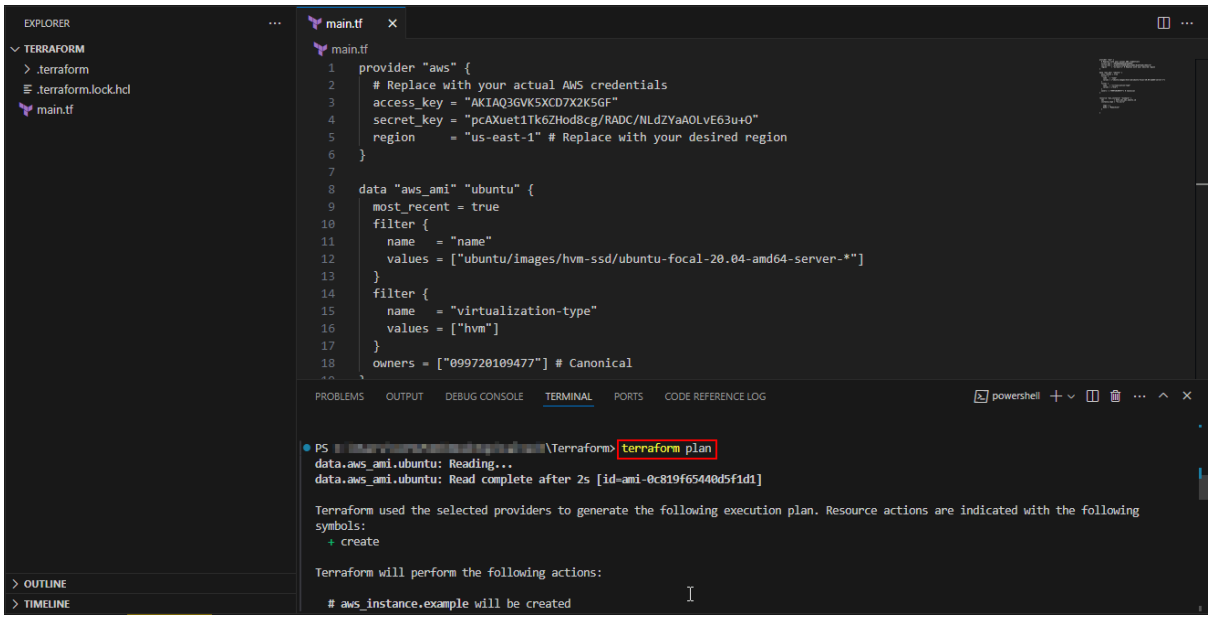
3.3 Validate the configuration using the following command as shown in the screenshot below:

terraform validate



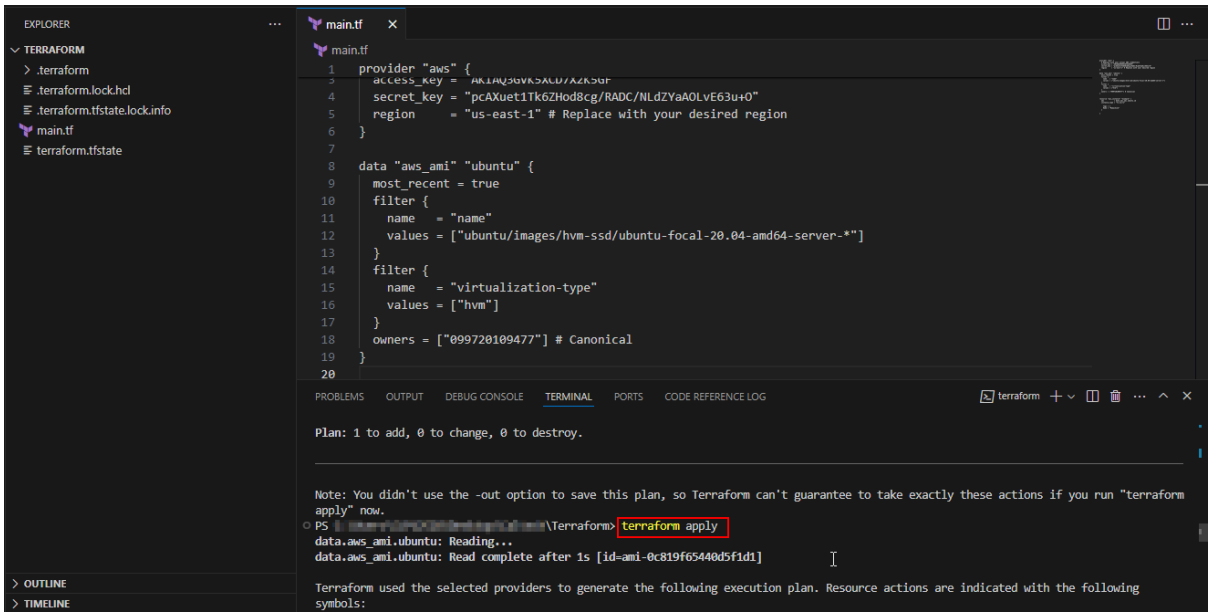
3.4 Plan the changes using the following command as shown in the screenshot below:

terraform plan

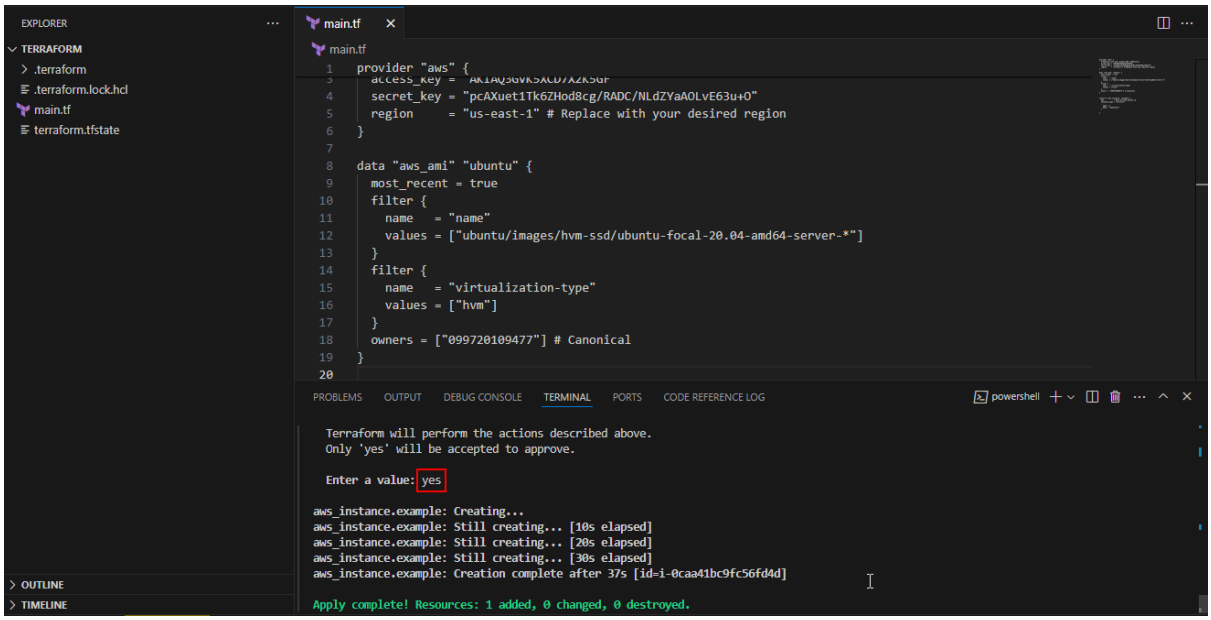


3.5 Apply the configuration using the following command as shown in the screenshot below:

terraform apply

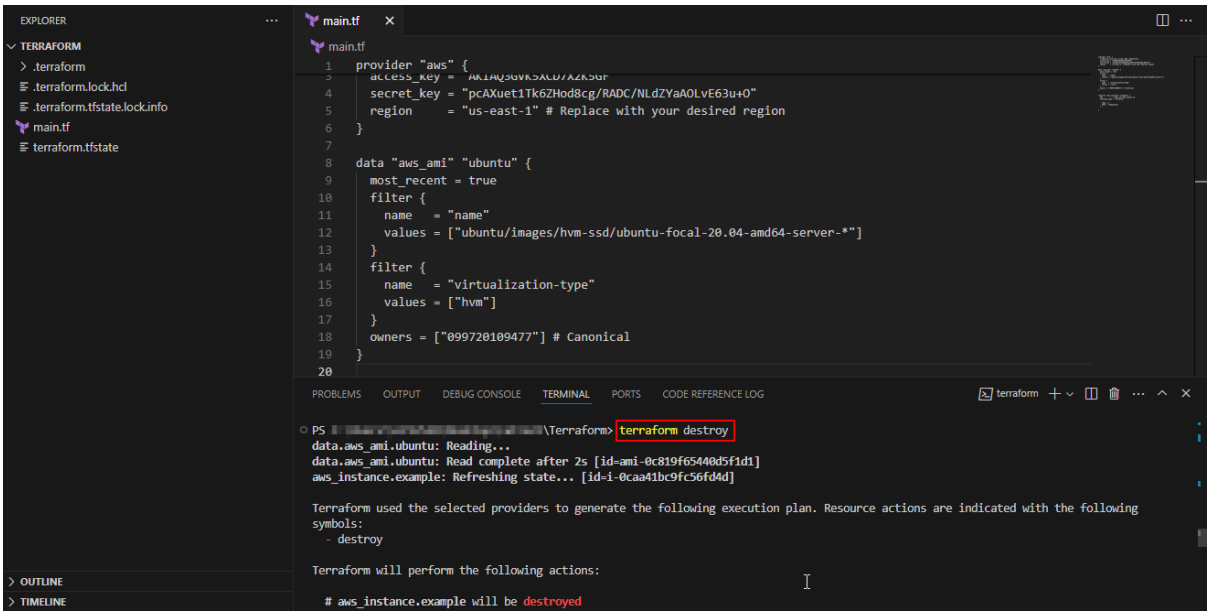


3.6 Type **yes** and click on the **enter** key to proceed further as shown in the screenshot below:

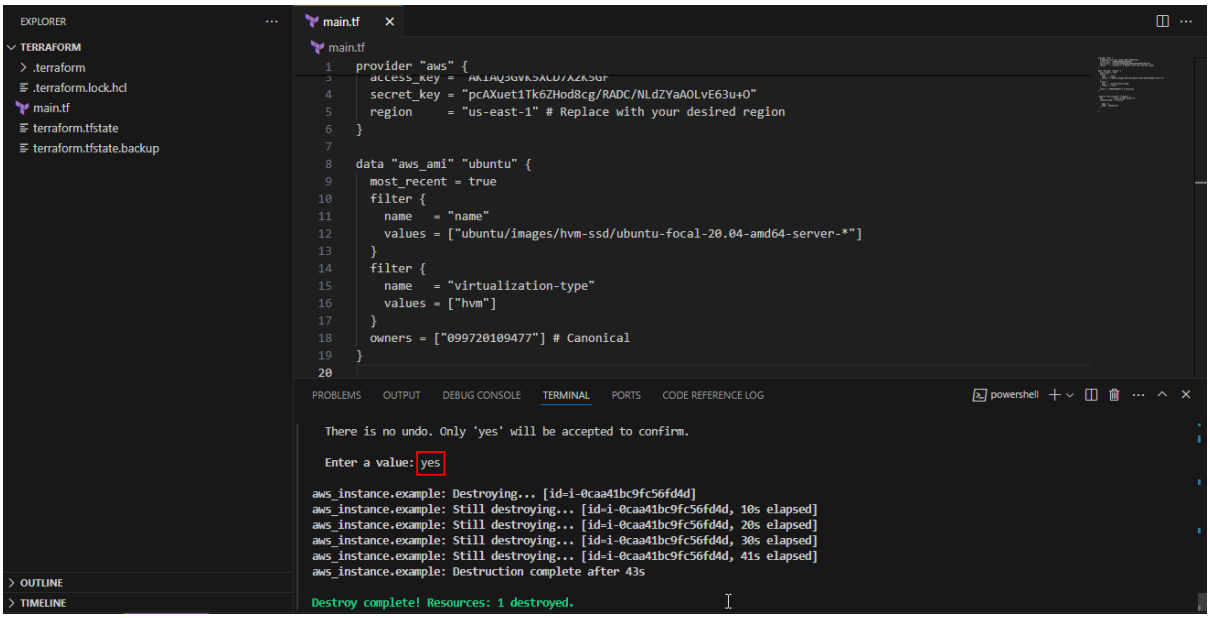


3.7 Execute the following command to remove all the resources defined in the Terraform configuration as shown in the screenshot below:

terraform destroy



3.8 Type **yes** and click on the **Enter** key to proceed further as shown in the screenshot below:



By following these steps, you have successfully implemented a workflow for efficiently and effectively managing AWS infrastructure configurations in Terraform.