Lesson 08 Demo 02

Deploying AWS Infrastructure with Terraform

Objective: To prepare files and set credentials necessary for using Terraform to deploy and manage AWS infrastructure efficiently

Tools required: AWS Account, Terraform, VS code

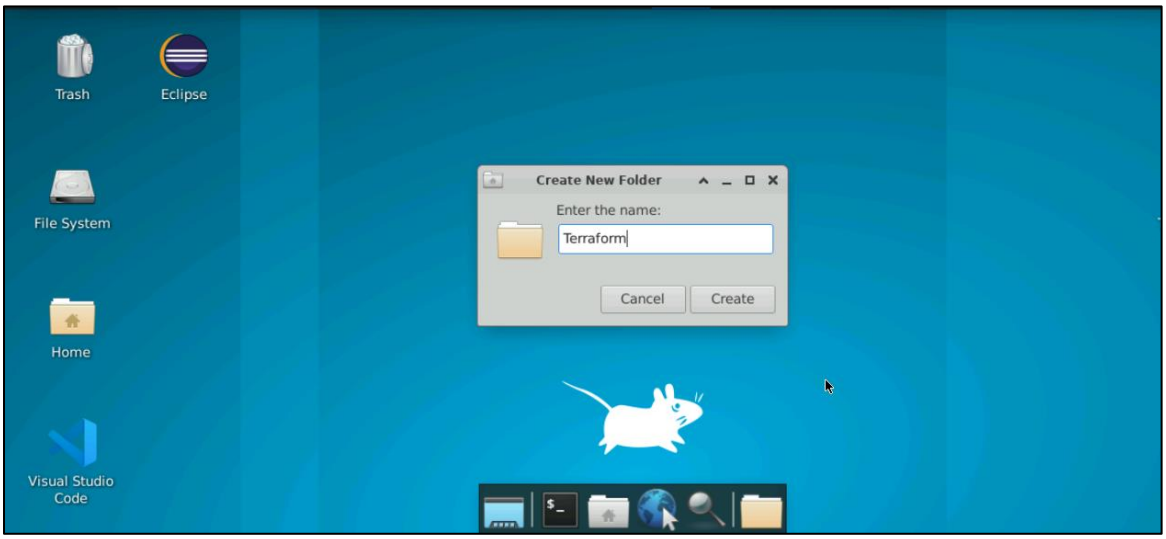
Prerequisites: None

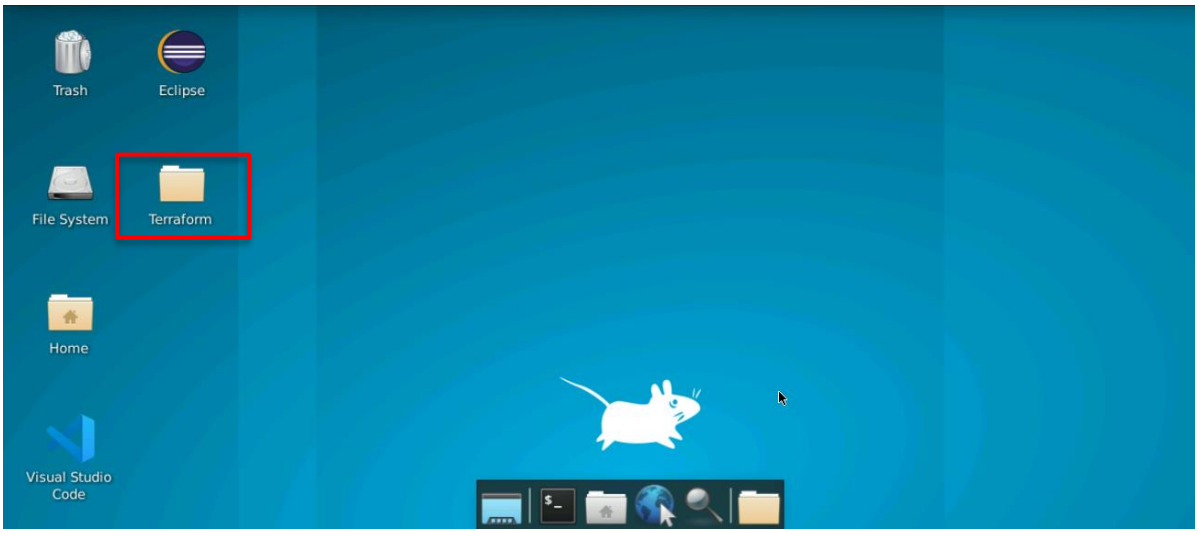
Steps to be followed:

1. Prepare files and credentials for using Terraform to deploy cloud resources
2. Set credentials for Terraform deployment
3. Deploy the AWS infrastructure using Terraform
4. Delete the AWS resources using Terraform to clean up your AWS environment

Step 1: Prepare files and credentials for using Terraform to deploy cloud resources

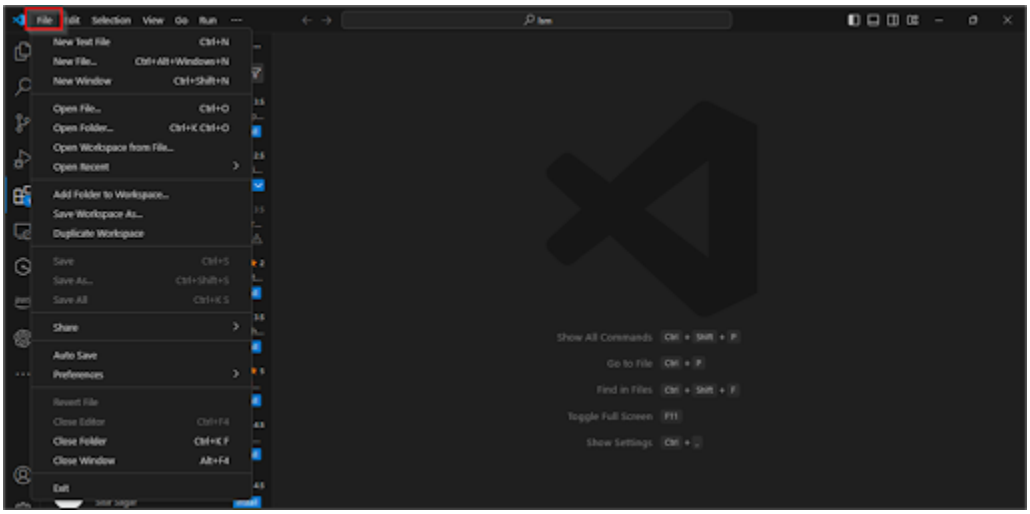
1.1 Create a folder named **Terraform** on the desktop



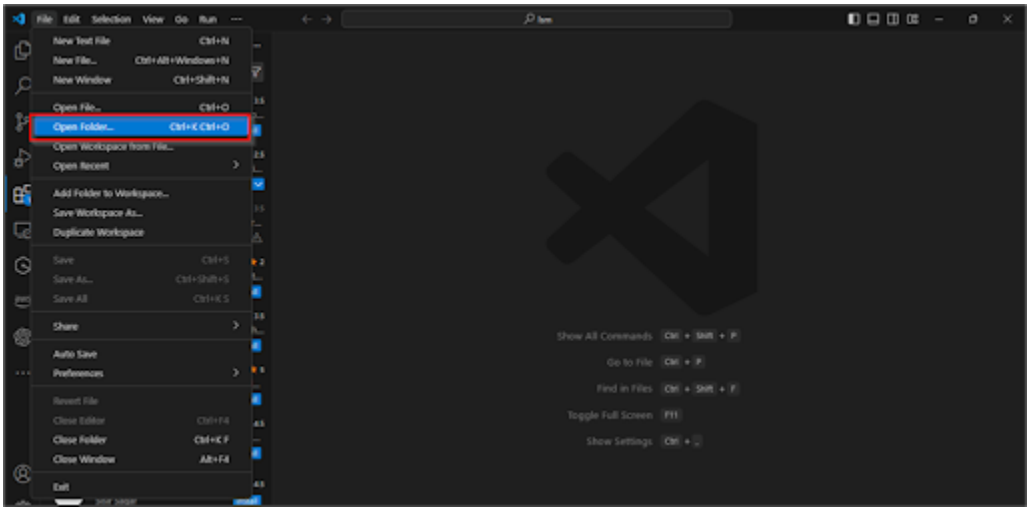


1.2 Double-click on the **VS Code editor** icon present on the desktop to open it

1.3 Open the **File** option present on the top console



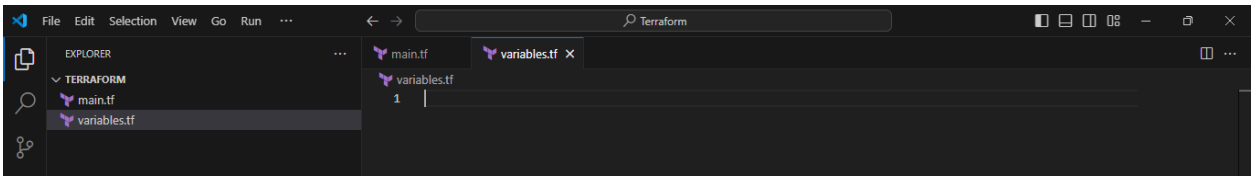
1.4 Click on the **Open Folder** from the drop-down menu to open the Terraform folder



The **Terraform** folder will be opened in the VS Code editor.

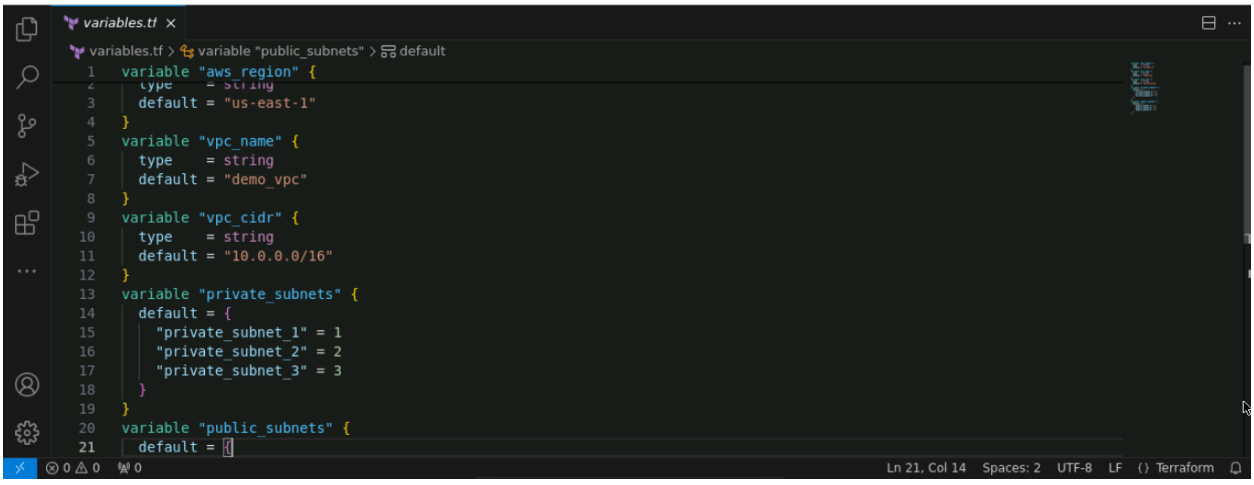
1.5 Navigate to the **Terraform** folder on the editor and click on the **New File** option to create the main file

1.6 Create two files named **main.tf** and **validate.tf**



1.7 In the **variables.tf**, copy the following variable definitions and save the file

```
variable "aws_region"
{
type = string default = "us-east-1"
}
variable "vpc_name"
{
type = string default = "demo_vpc"
}
 variable "vpc_cidr"
{
 type = string default = "10.0.0.0/16"
} variable "private_subnets"
{ default = { "private_subnet_1" = 1 "private_subnet_2" = 2 "private_subnet_3" = 3 }
}
 variable "public_subnets" {
default = { "public_subnet_1" = 1 "public_subnet_2" = 2 "public_subnet_3" = 3 }
}
```

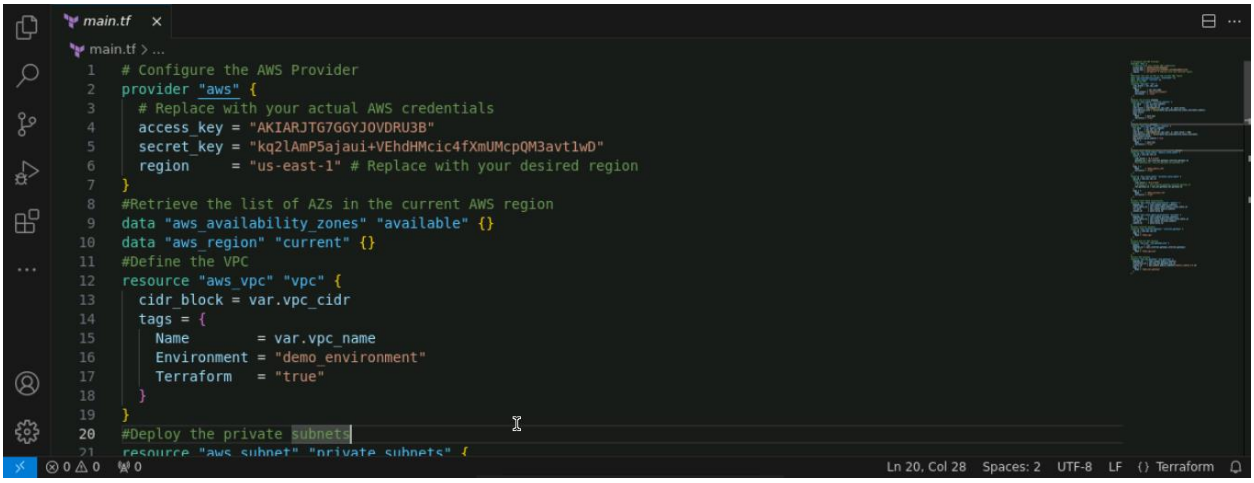


1.8 In the **main.tf** file, copy the following Terraform configuration, and save the file

```
# Configure the AWS Provider
provider "aws" {
  # Replace with your actual AWS credentials
  access_key = "AKIARJTG7GGYJOVDRU3B"
  secret_key = "kq2lAmP5ajaui+VEhdHMcic4fXmUMcpQM3avt1wD"
  region     = "us-east-1" # Replace with your desired region
}
#Retrieve the list of AZs in the current AWS region
data "aws_availability_zones" "available" {}
data "aws_region" "current" {}
#Define the VPC
resource "aws_vpc" "vpc" {
  cidr_block = var.vpc_cidr
  tags = {
    Name        = var.vpc_name
    Environment = "demo_environment"
    Terraform   = "true"
  }
}
#Deploy the private subnets

resource "aws_subnet" "private_subnets" {
  for_each   = var.private_subnets
  vpc_id     = aws_vpc.vpc.id
  cidr_block = cidrsubnet(var.vpc_cidr, 8, each.value)
  availability_zone = tolist(data.aws_availability_zones.available.names)[
  each.value]
  tags = {
    Name      = each.key
    Terraform = "true"
  }
}
#Deploy the public subnets
resource "aws_subnet" "public_subnets" {
  for_each   = var.public_subnets
  vpc_id     = aws_vpc.vpc.id
```

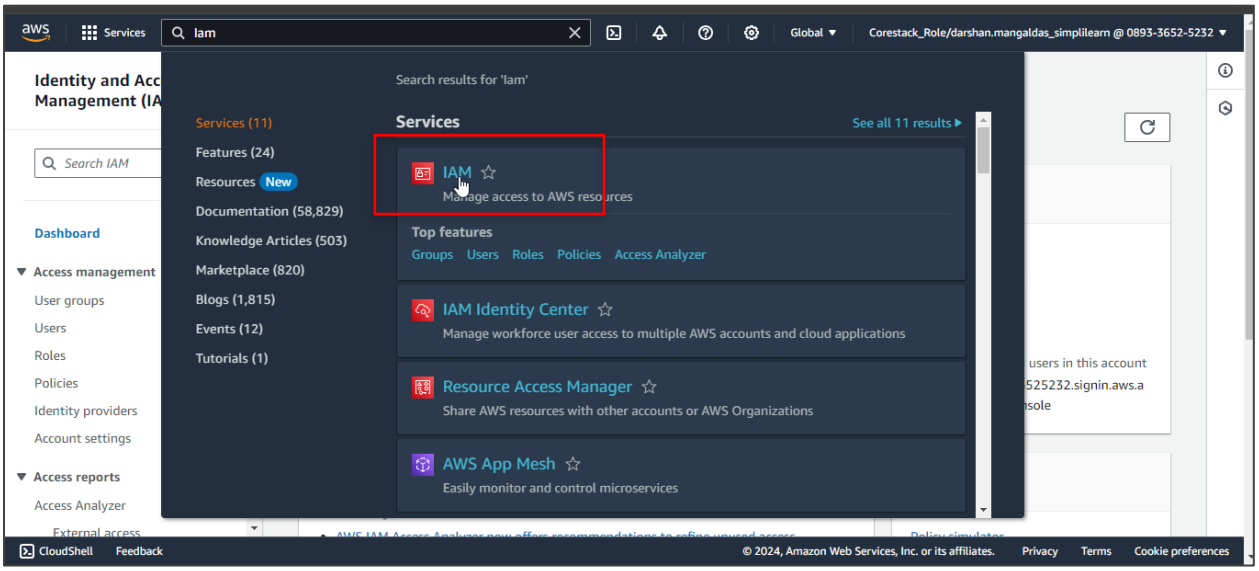
```
  cidr_block = cidrsubnet(var.vpc_cidr, 8, each.value + 100)
  availability_zone = tolist(data.aws_availability_zones.available.
  names)[each.value]
  map_public_ip_on_launch = true
  tags = {
    Name      = each.key
    Terraform = "true"
  }
}
#Create route tables for public and private subnets
resource "aws_route_table" "public_route_table" {
  vpc_id = aws_vpc.vpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.internet_gateway.id
    #nat_gateway_id = aws_nat_gateway.nat_gateway.id
  }
  tags = {
    Name      = "demo_public_rtb"
    Terraform = "true"
  }
}
resource "aws_route_table" "private_route_table" {
  vpc_id = aws_vpc.vpc.id
  route {
    cidr_block = "0.0.0.0/0"
    # gateway_id = aws_internet_gateway.internet_gateway.id
    nat_gateway_id = aws_nat_gateway.nat_gateway.id

  }
  tags = {
    Name      = "demo_private_rtb"
    Terraform = "true"
  }
}
#Create route table associations
resource "aws_route_table_association" "public" {
  depends_on     = [aws_subnet.public_subnets]
```

```
  route_table_id = aws_route_table.public_route_table.id
  for_each       = aws_subnet.public_subnets
  subnet_id      = each.value.id
}
resource "aws_route_table_association" "private" {
  depends_on     = [aws_subnet.private_subnets]
  route_table_id = aws_route_table.private_route_table.id
  for_each       = aws_subnet.private_subnets
  subnet_id      = each.value.id
}
#Create Internet Gateway
resource "aws_internet_gateway" "internet_gateway" {
  vpc_id = aws_vpc.vpc.id
  tags = {
    Name = "demo_igw"
  }
}
#Create EIP for NAT Gateway
resource "aws_eip" "nat_gateway_eip" {
  domain     = "vpc"
  depends_on = [aws_internet_gateway.internet_gateway]
  tags = {
    Name = "demo_igw_eip"
  }
}
#Create NAT Gateway
resource "aws_nat_gateway" "nat_gateway" {
  depends_on    = [aws_subnet.public_subnets]
  allocation_id = aws_eip.nat_gateway_eip.id
  subnet_id     = aws_subnet.public_subnets["public_subnet_1"].id
  tags = {
    Name = "demo_nat_gateway"
  }
}
```

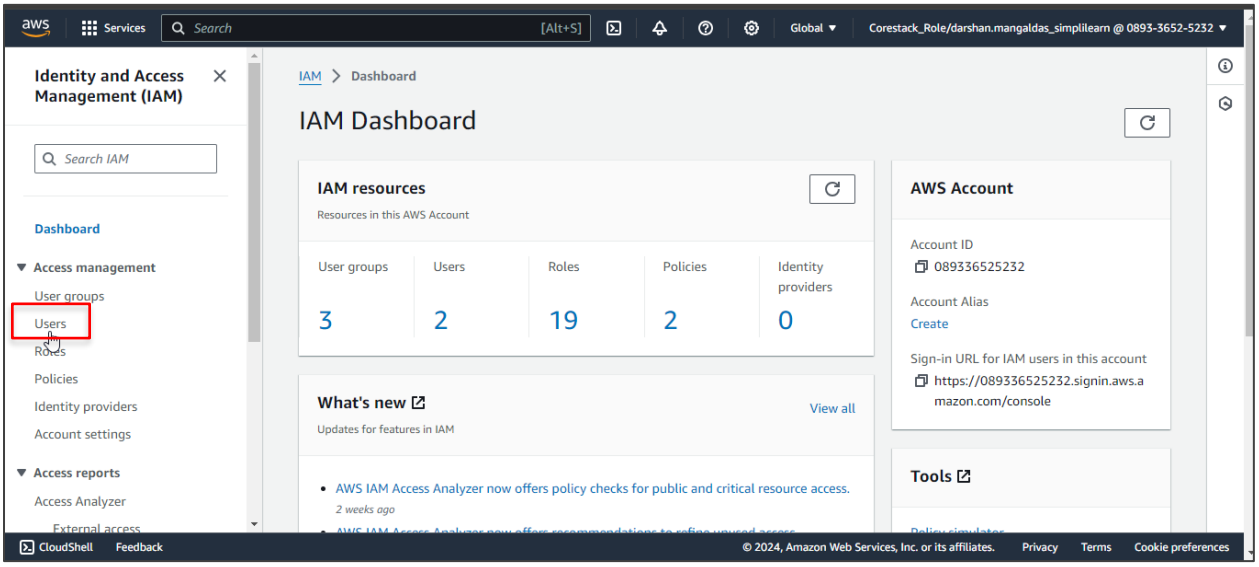


Step 2: Set credentials for Terraform deployment

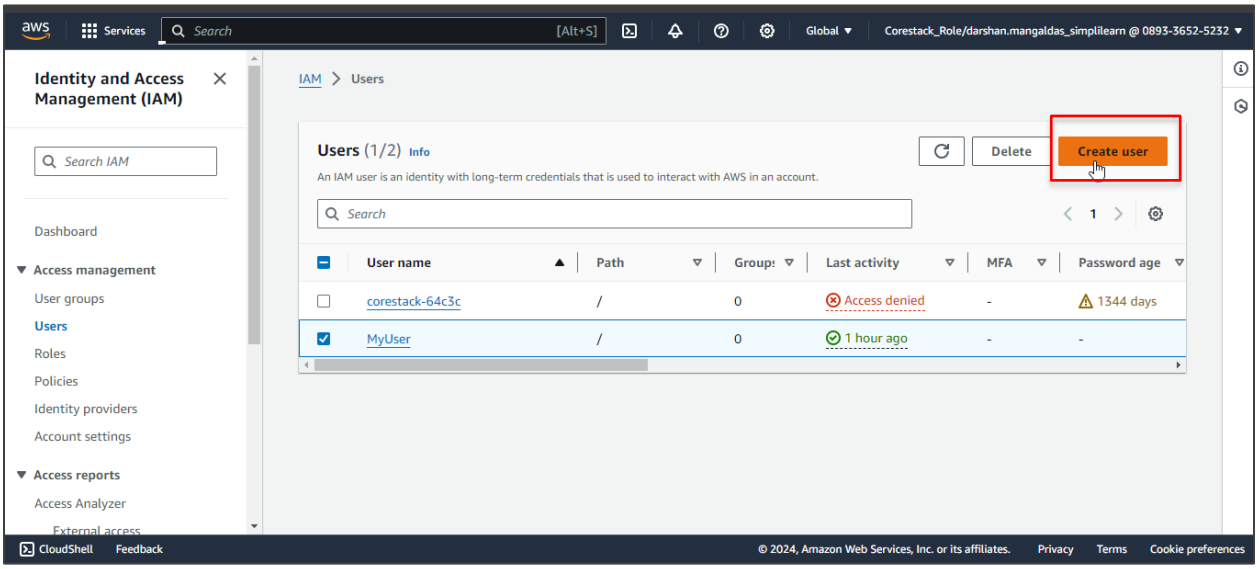
2.1 To set the access key and secret key, search for **IAM** on the AWS console



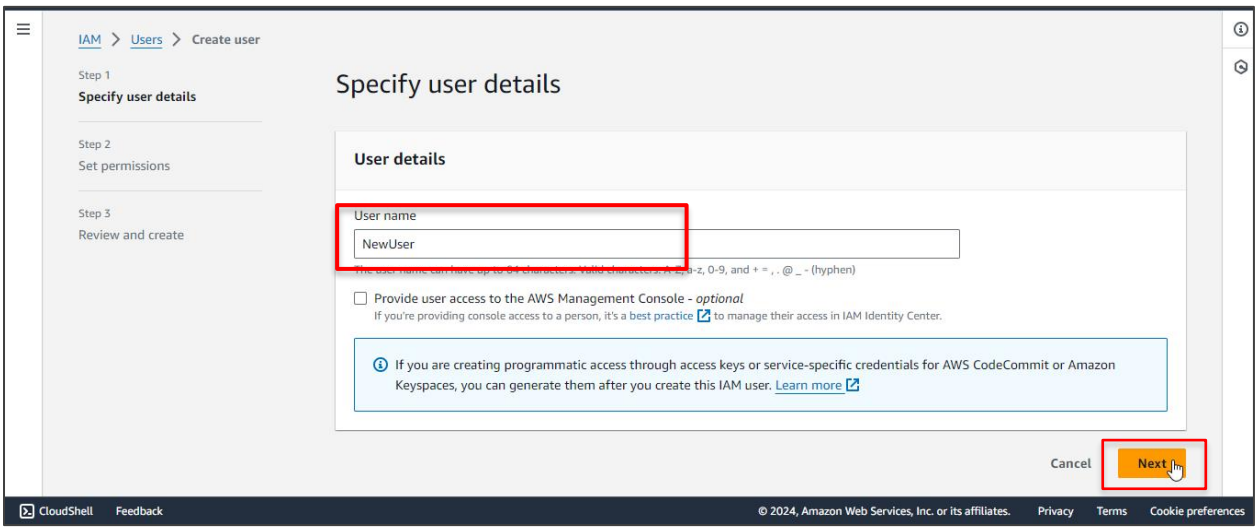
2.2 Click on the **Users** option under the dashboard



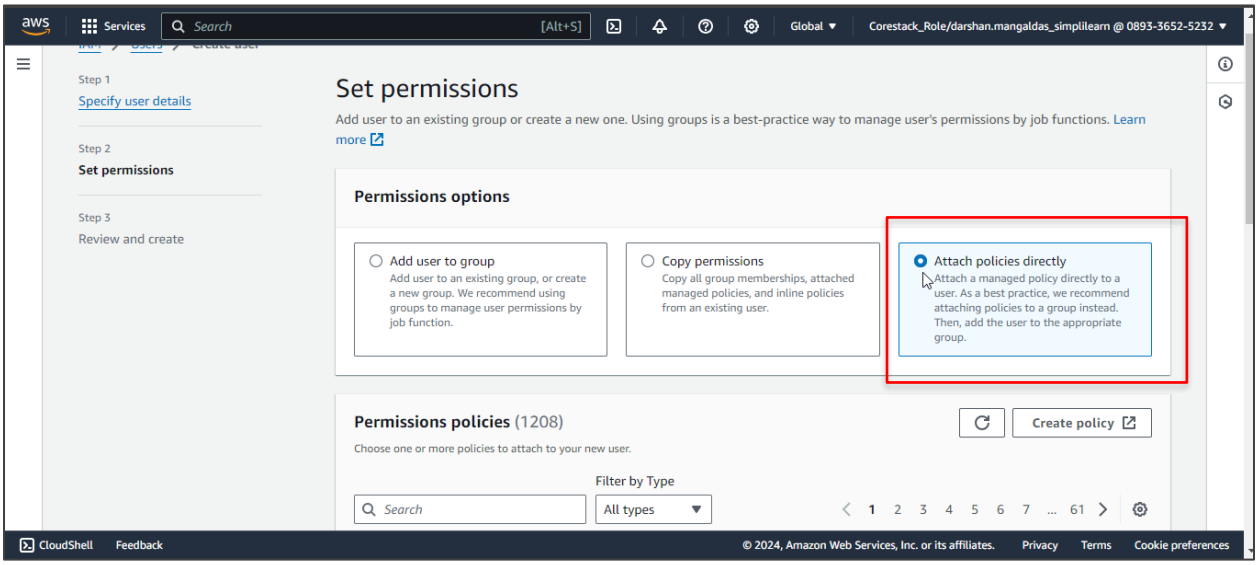
2.3 Click on the **Create user** button



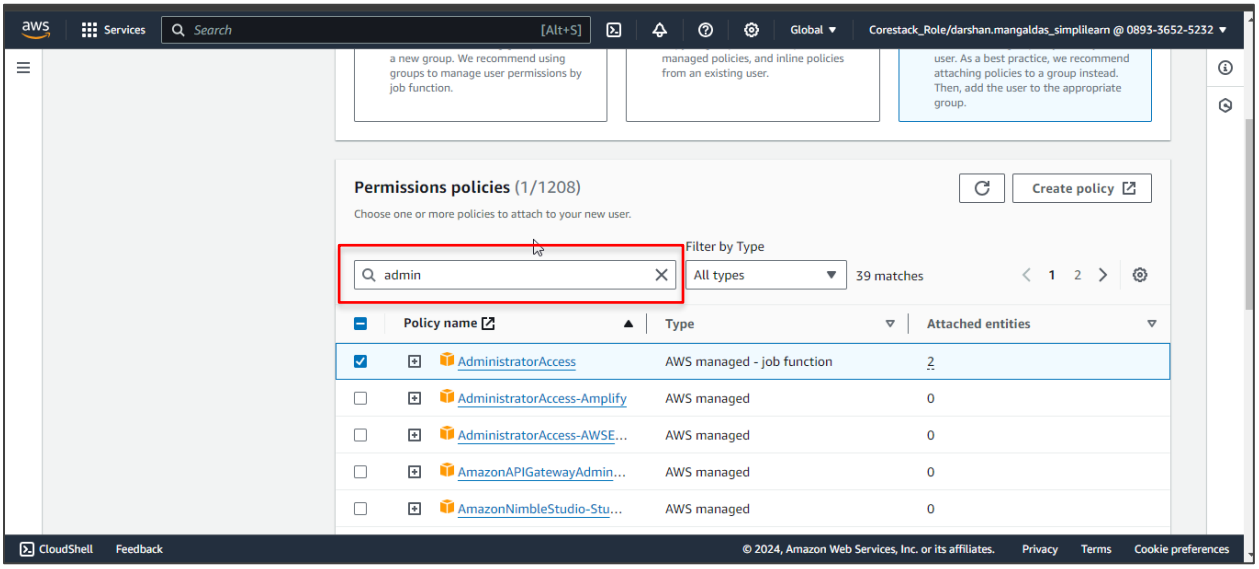
2.4 Enter **NewUser** under the **User name** field in the **User details** page and click on **Next**



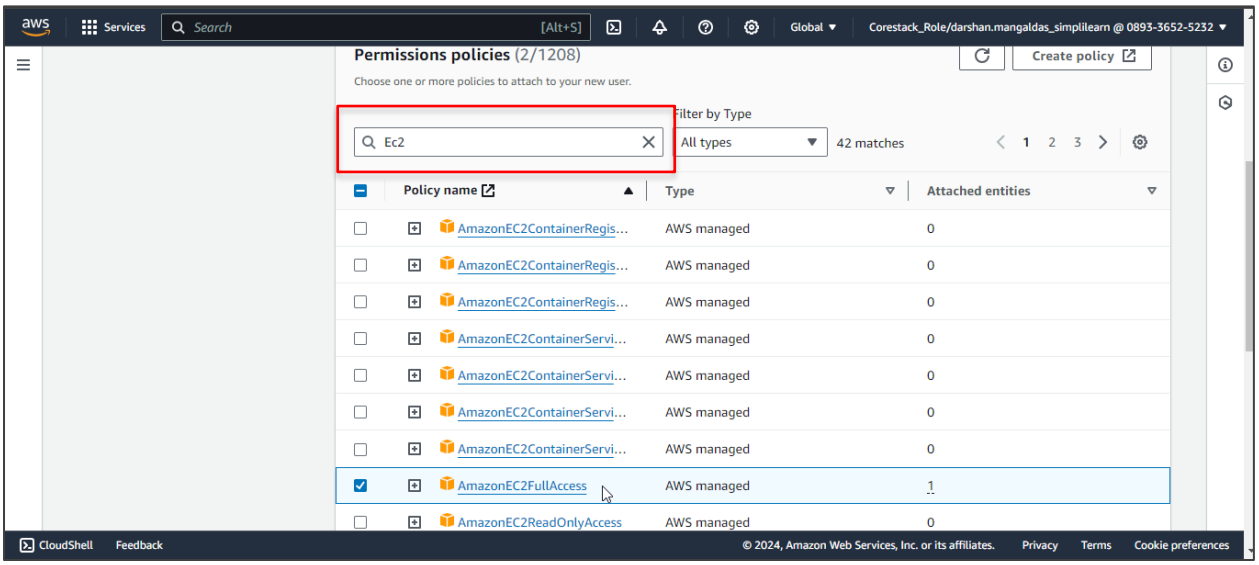
2.5 Select the **Attach policies directly** in the **Set permissions** page



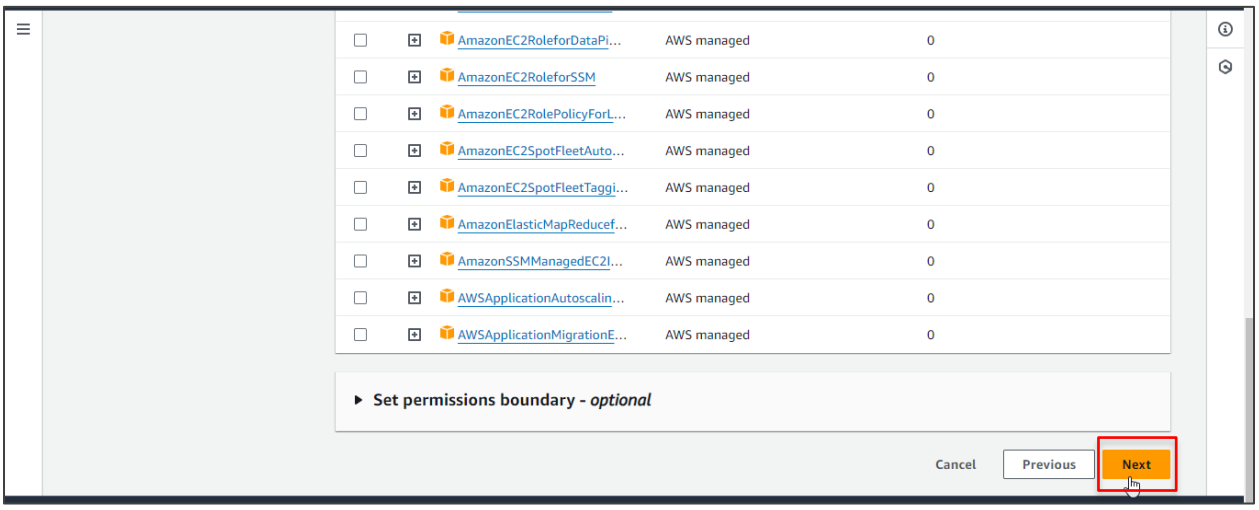
2.6 Scroll down to the **Permission policies** page, search for **admin** in the search box, and select **AdminstratorAccess** under **Policy name**



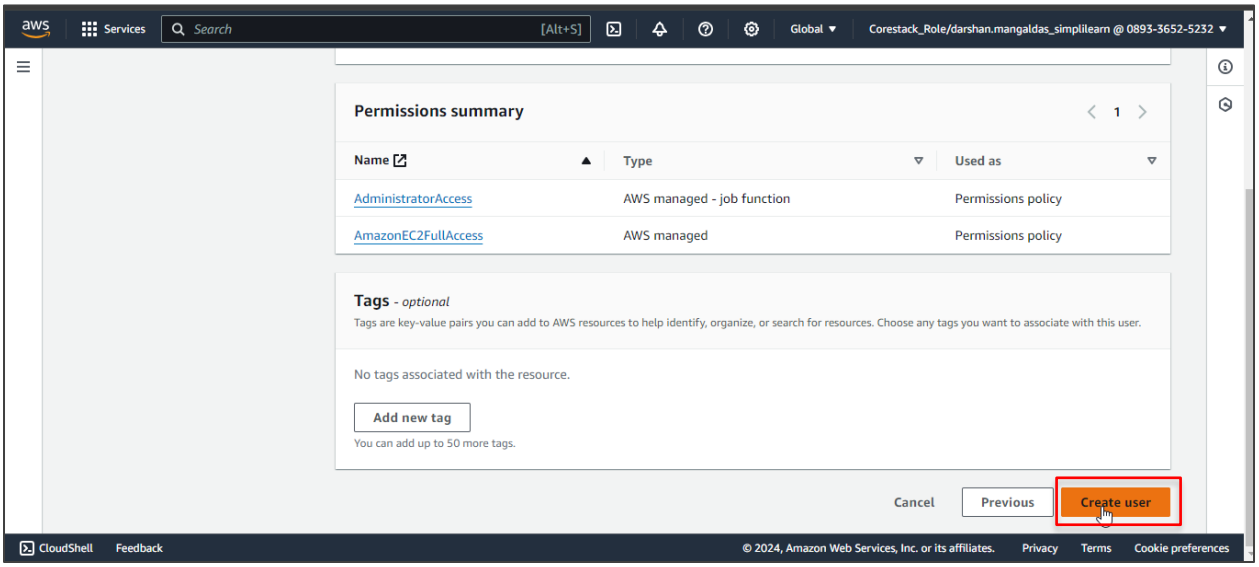
2.7 Search for **Ec2** in the search bar and select **AmazonEC2FullAccess** under **Policy name**



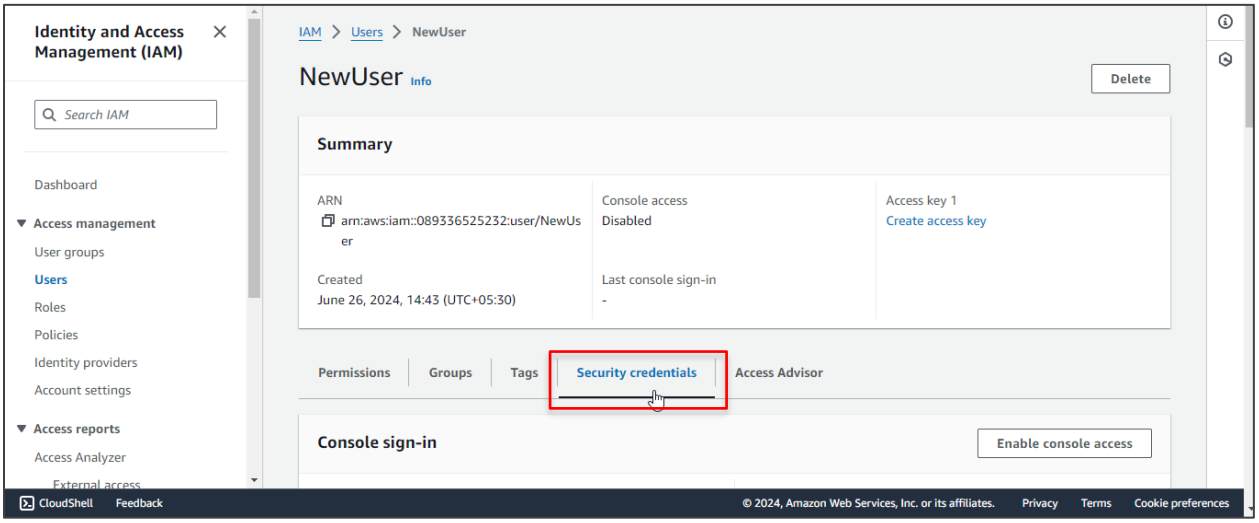
2.8 Scroll down and click on **Next**



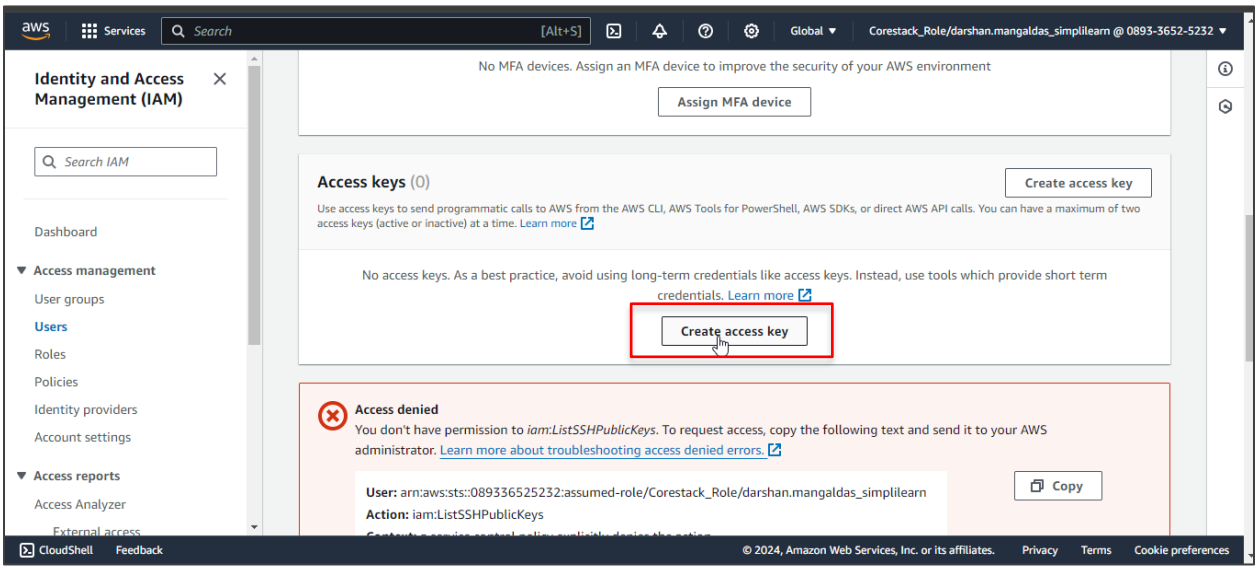
2.9 Click on **Create user** in the **Permissions summary** tab



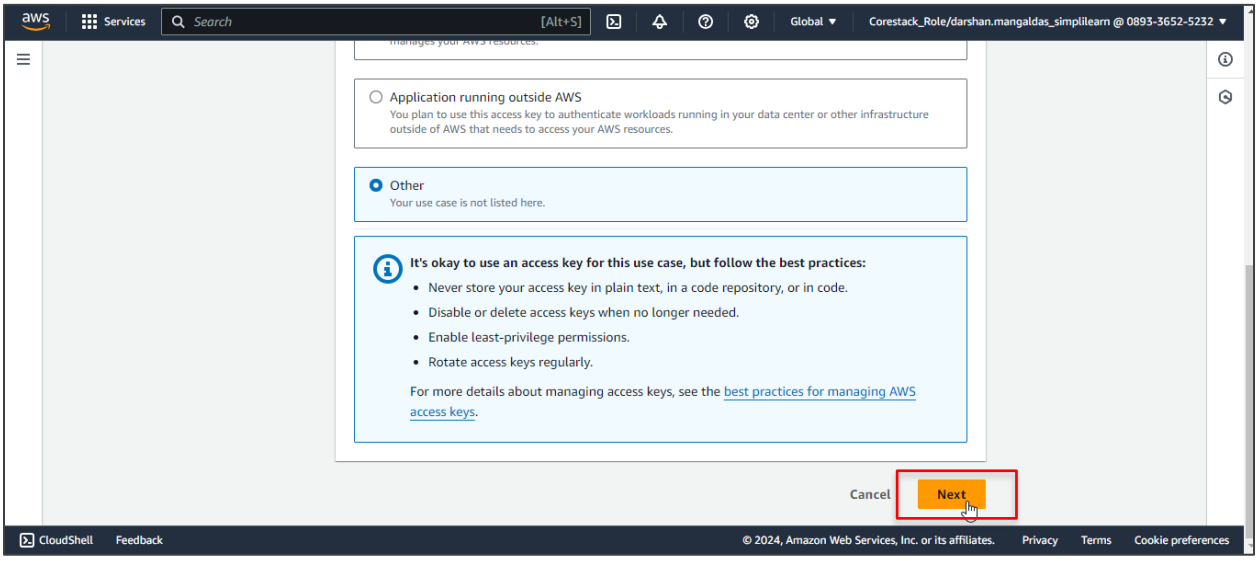
2.10 Click on **Security credentials** as shown in the screenshot below:



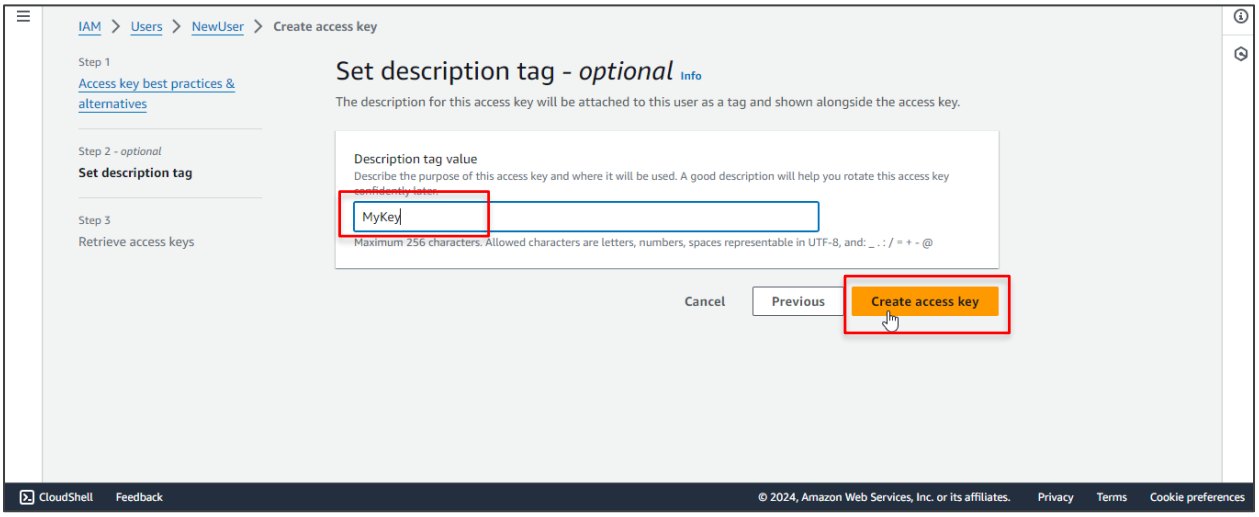
2.11 Click on **Create access key** as shown in the screenshot below:



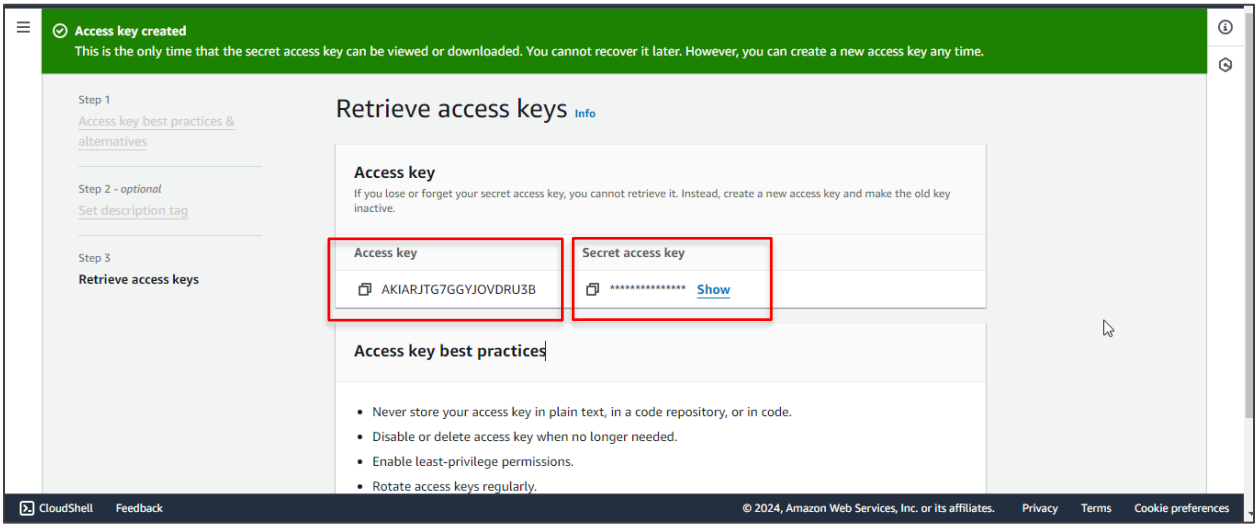
2.12 Scroll down, choose **Other,** and click on Next as shown in the screenshot below:



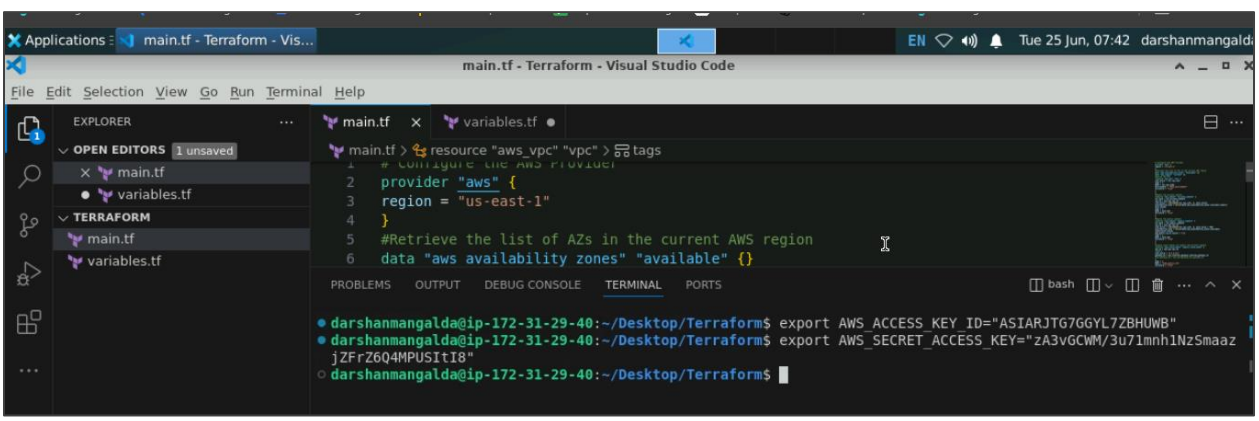
2.13 Enter a desired name in the **Description tag value** text area and then click on **Create access key**



2.14 Copy the **Access key** and **Secret access key**

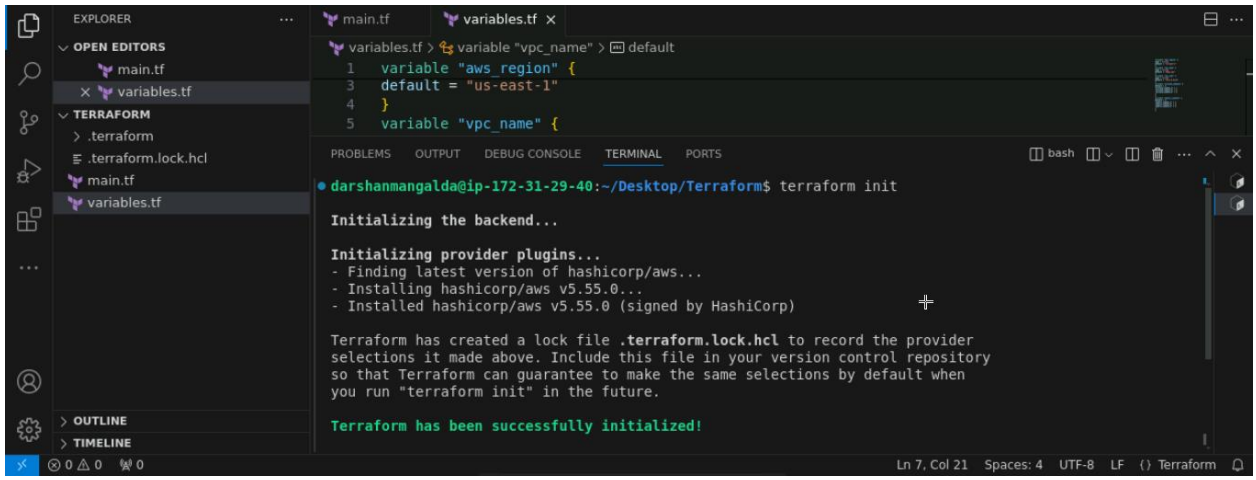


2.15 Once you have the credentials, go to the VS Code terminal and provide the given credentials **export AWS_ACCESS_KEY_ID= " "**
export AWS_SECRET_ACCESS_KEY=" " as shown in the screenshot below and click on Enter



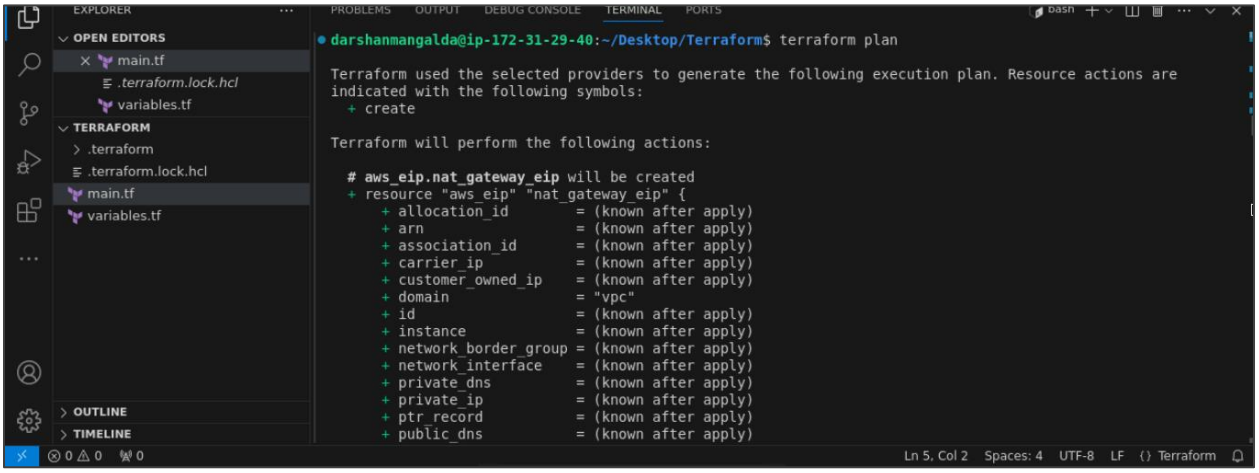
Step 3: Deploy the AWS infrastructure using Terraform

3.1 Open the terminal and execute the given command to initialize the terraform file:
terraform init



The Terraform configuration file is successfully initialized.

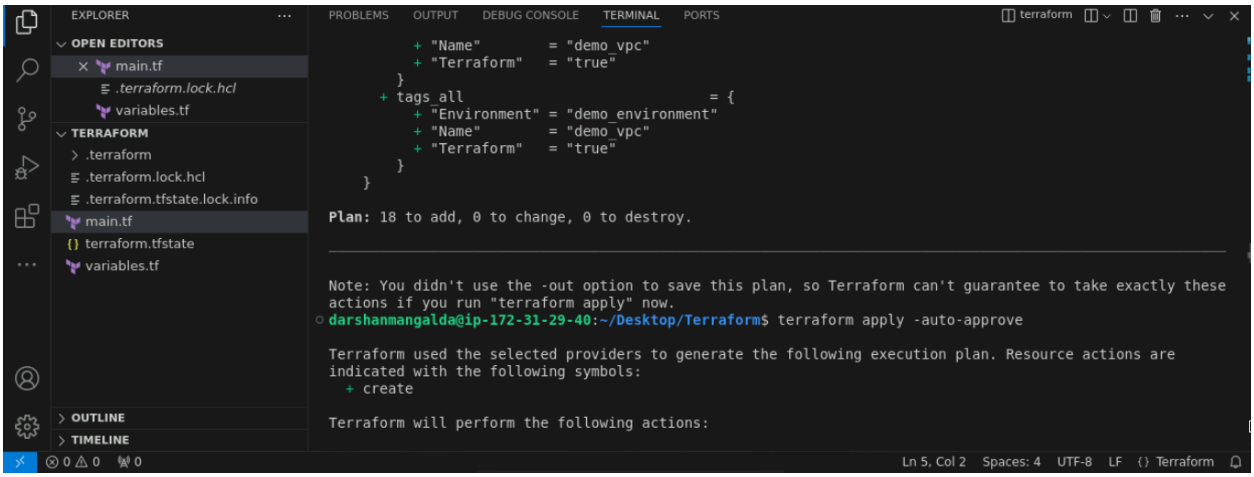
3.2 Execute the given command to create an execution plan:
terraform plan

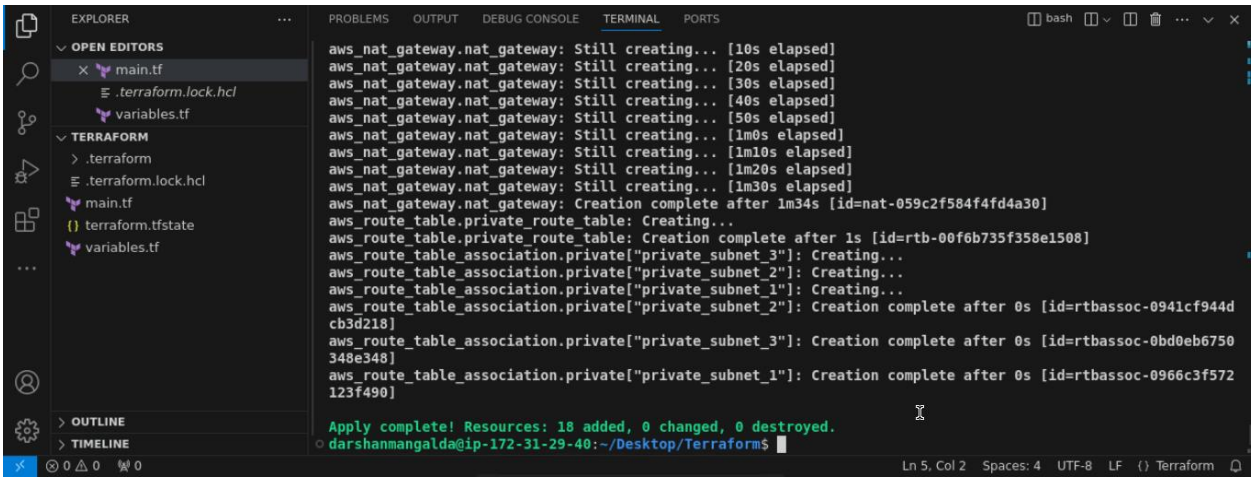


The execution plan is successfully created.

3.3 Enter the following command to execute the changes:

terraform apply -auto-approve

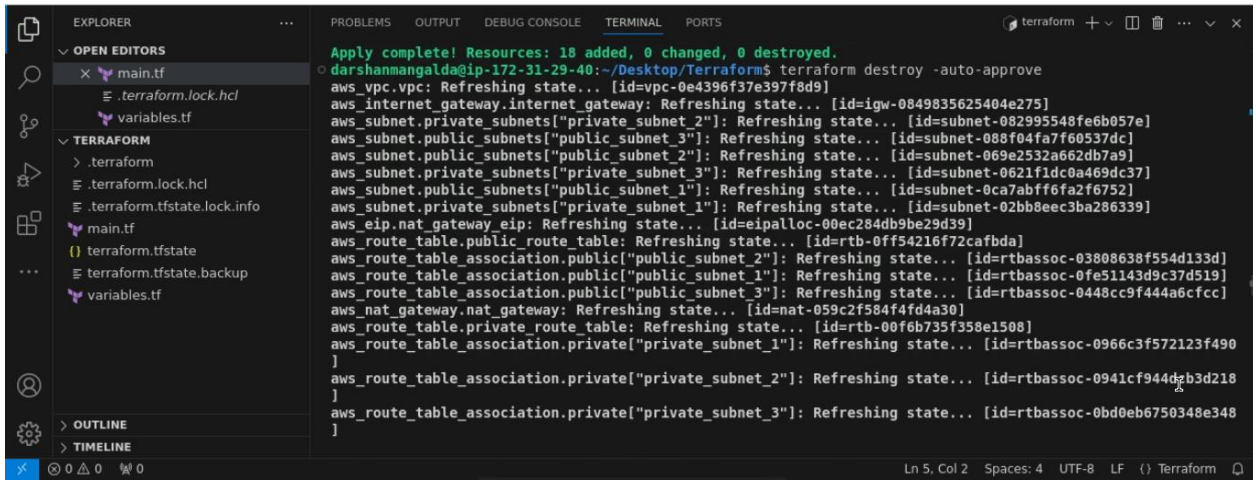


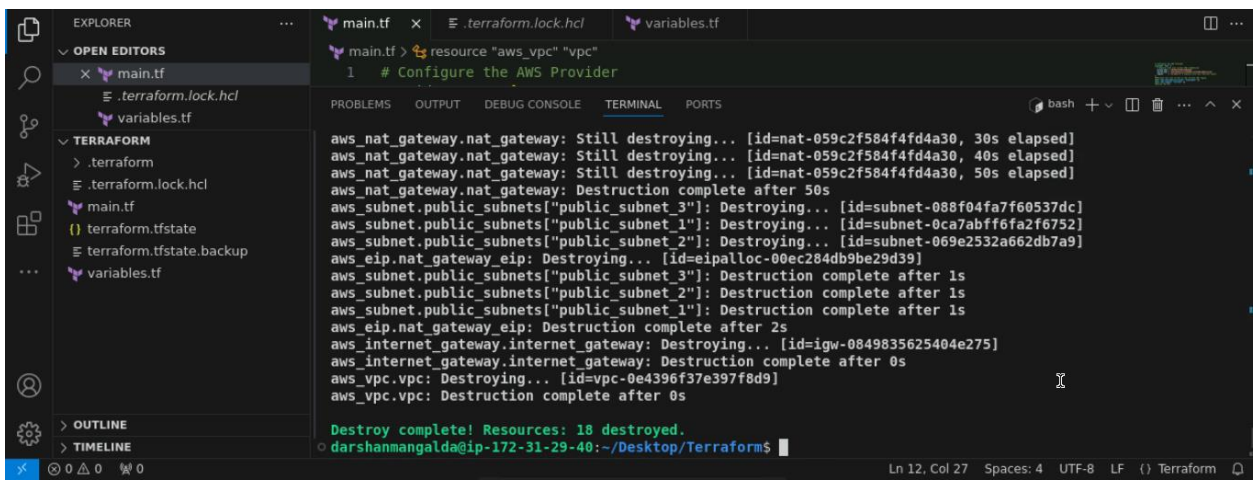


The changes are successfully executed.

Step 4: Delete the AWS resources using Terraform to clean up your AWS environment

4.1 To destroy your resources, execute the following command in the terminal:
terraform destroy -auto-approve





By following these steps, you have effectively streamlined the process of preparing files and setting the necessary credentials, allowing you to use Terraform to deploy and manage AWS infrastructure with greater efficiency.