

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

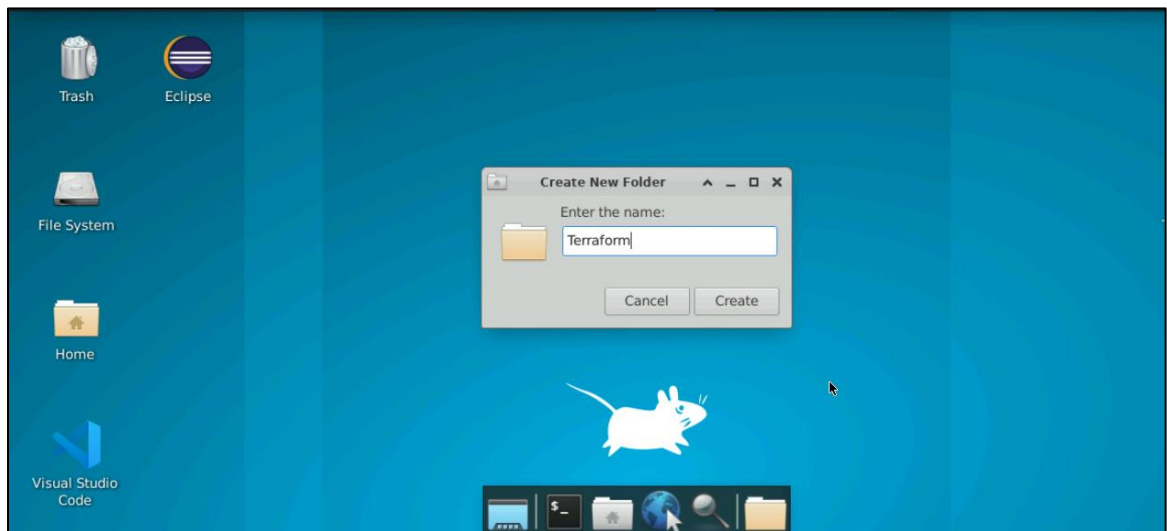
Prerequisites: None

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

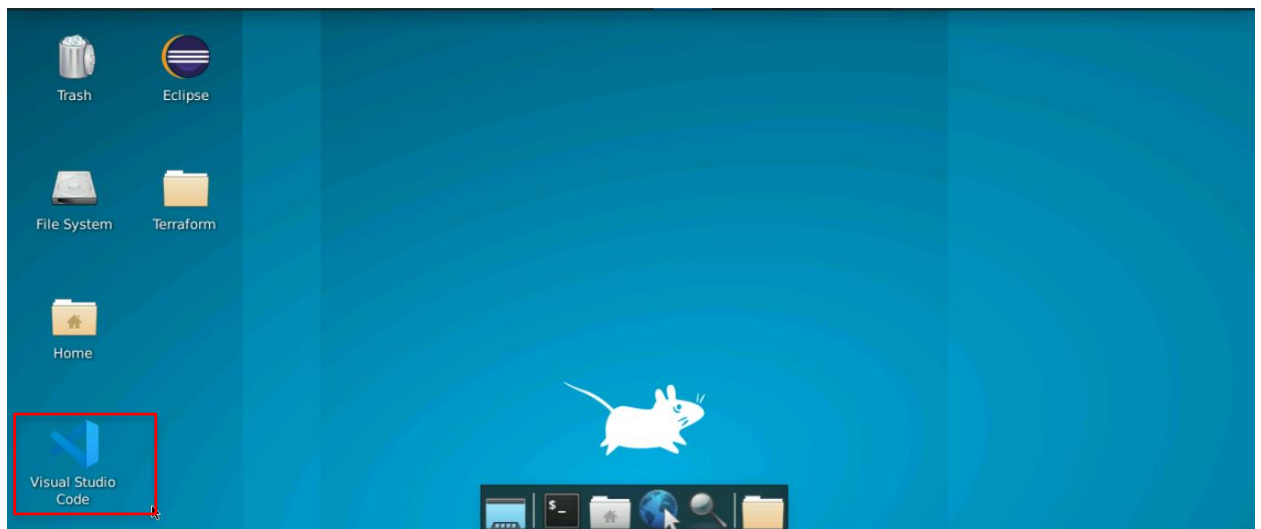
Step 1: Prepare files and credentials for using Terraform to deploy cloud resources

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

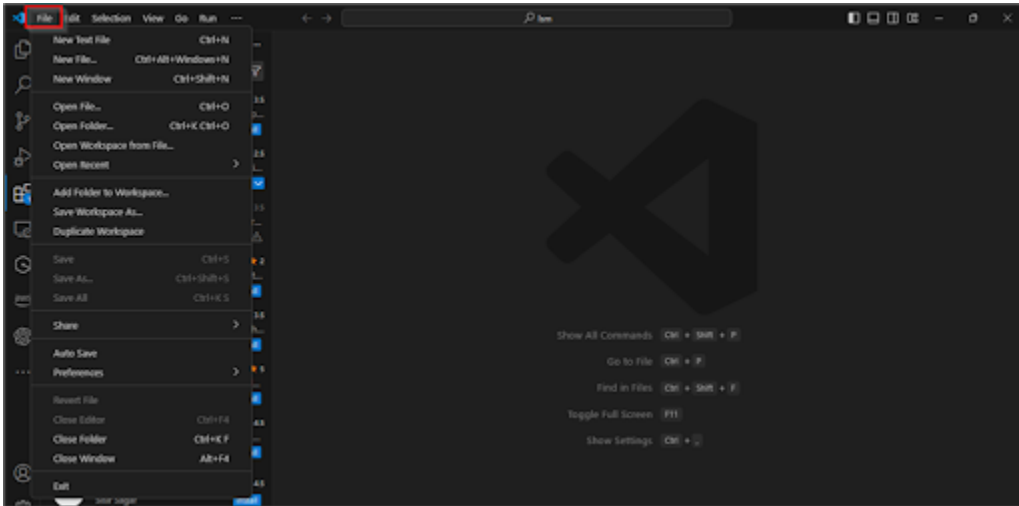




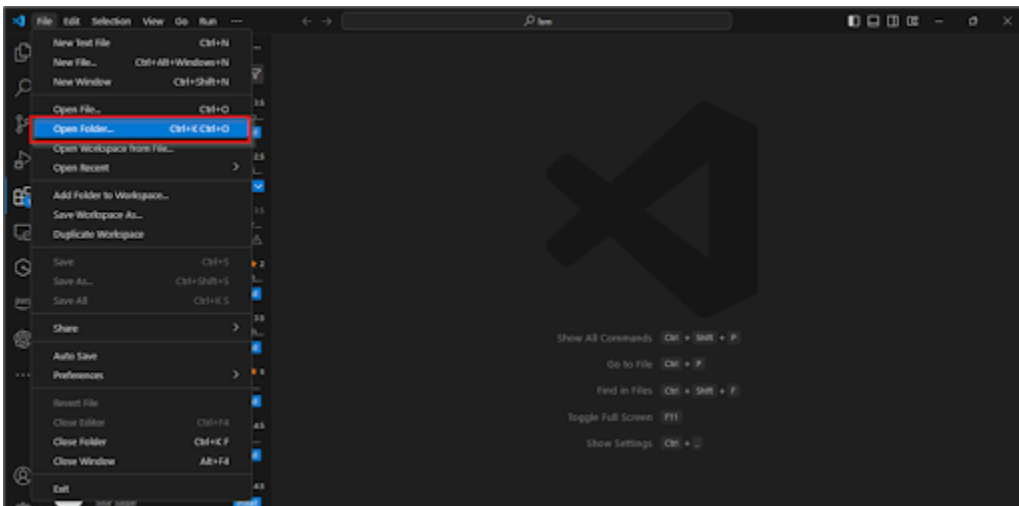
1.2 Double-click on the **VS Code editor** icon present on the desktop to open it



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

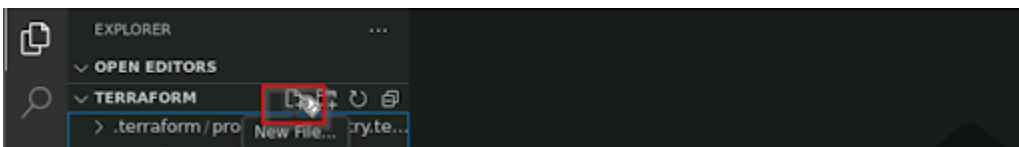


1.4 Click on the **Open Folder** from the drop-down menu to open the Terraform folder

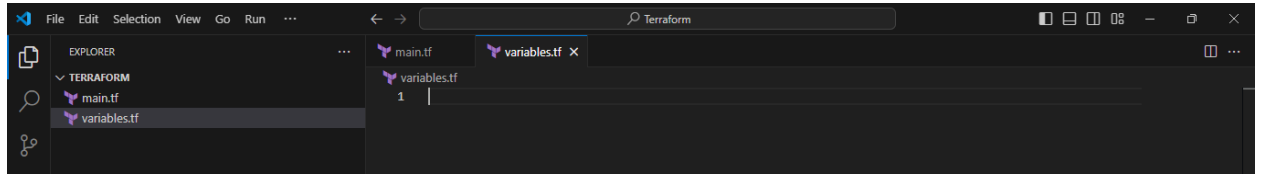


The **Terraform** folder will be opened in the VS Code editor.

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

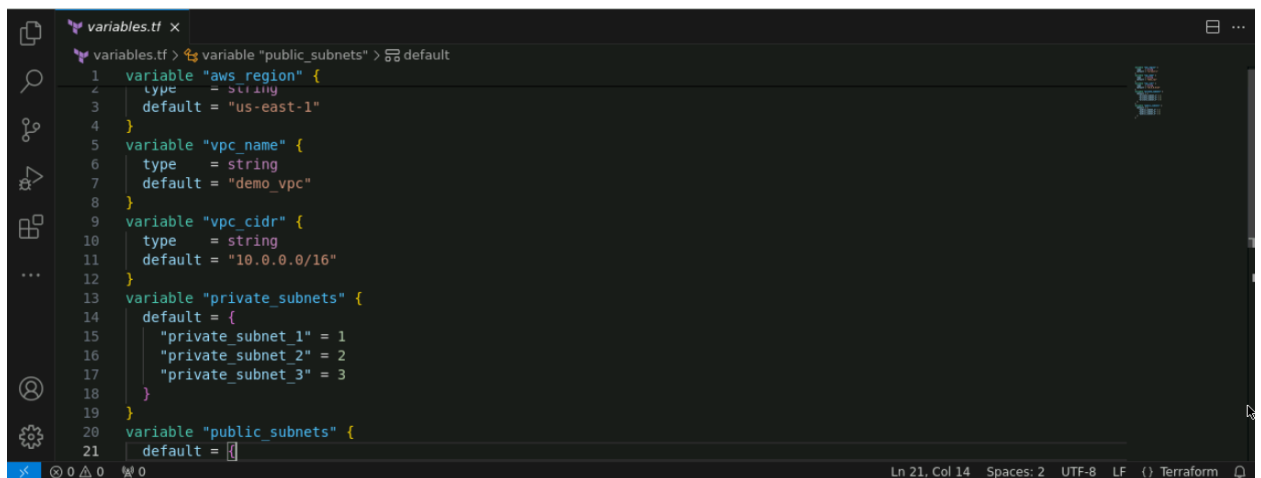


1.6 Create two files named **main.tf** and **validate.tf**



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 = "kq2lAmP5ajai+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_rt"
        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_rt"
        Terraform = "true"
    }
}
#Create route table associations
resource "aws_route_table_association" "public" {
    depends_on = [aws_subnet.public_subnets]

```

```

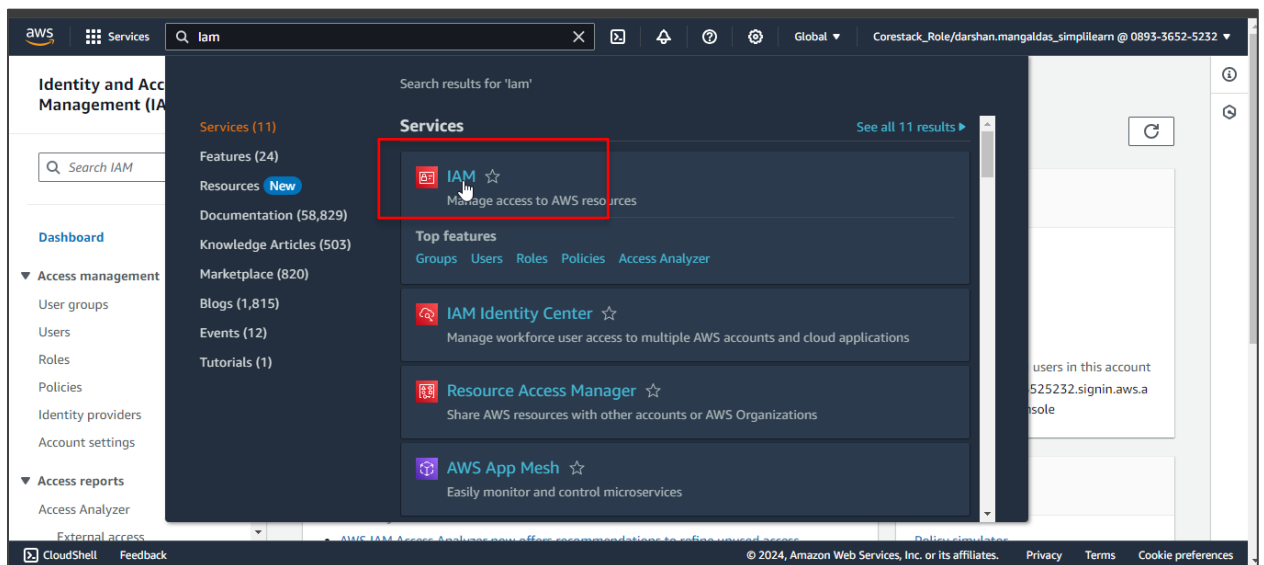
    route_table_id = aws_route_table.public_route_table.id
    for_each      = aws_subnet.public_subnets
    subnet_id     = each.value.id
  }
  resource "aws_route_table_association" "private" {
    depends_on    = [aws_subnet.private_subnets]
    route_table_id = aws_route_table.private_route_table.id
    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"
  }
}

```

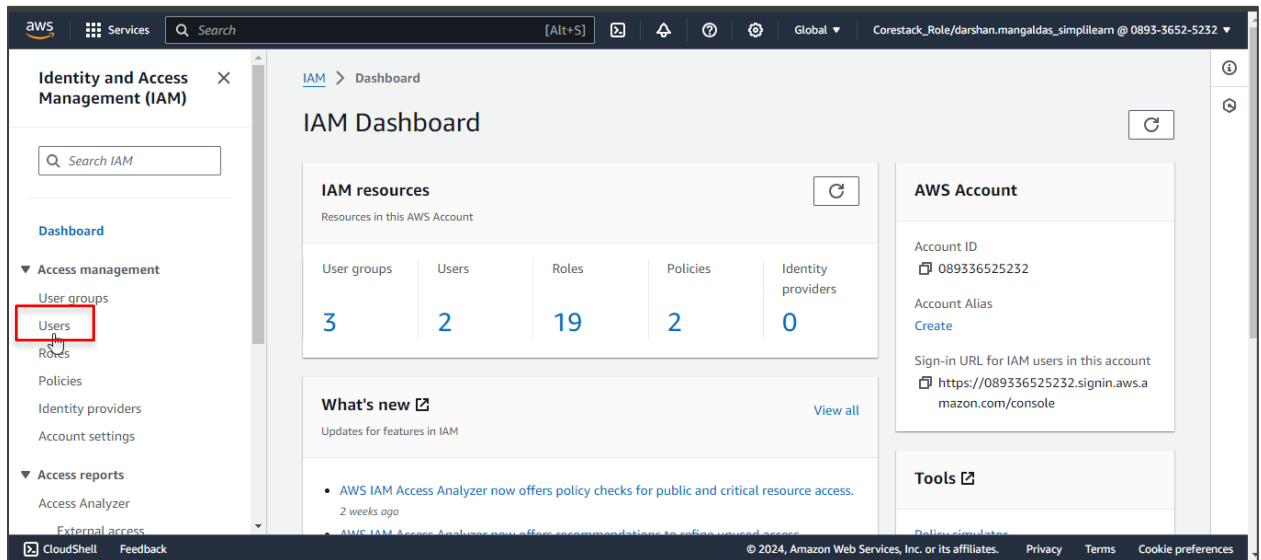
```
main.tf x
main.tf > ...
1 # Configure the AWS Provider
2 provider "aws" {
3   # Replace with your actual AWS credentials
4   access_key = "AKIARJTG6GYJOVDRU3B"
5   secret_key = "kq2lAmP5ajauI+VEhdHMcic4fXmUMcpQM3avt1wD"
6   region     = "us-east-1" # Replace with your desired region
7 }
8 #Retrieve the list of AZs in the current AWS region
9 data "aws_availability_zones" "available" {}
10 data "aws_region" "current" {}
11 #Define the VPC
12 resource "aws_vpc" "vpc" {
13   cidr_block = var.vpc_cidr
14   tags = {
15     Name       = var.vpc_name
16     Environment = "demo_environment"
17     Terraform  = "true"
18   }
19 }
20 #Deploy the private subnet
21 resource "aws_subnet" "private_subnet" {
```

Step 2: Set credentials for Terraform deployment

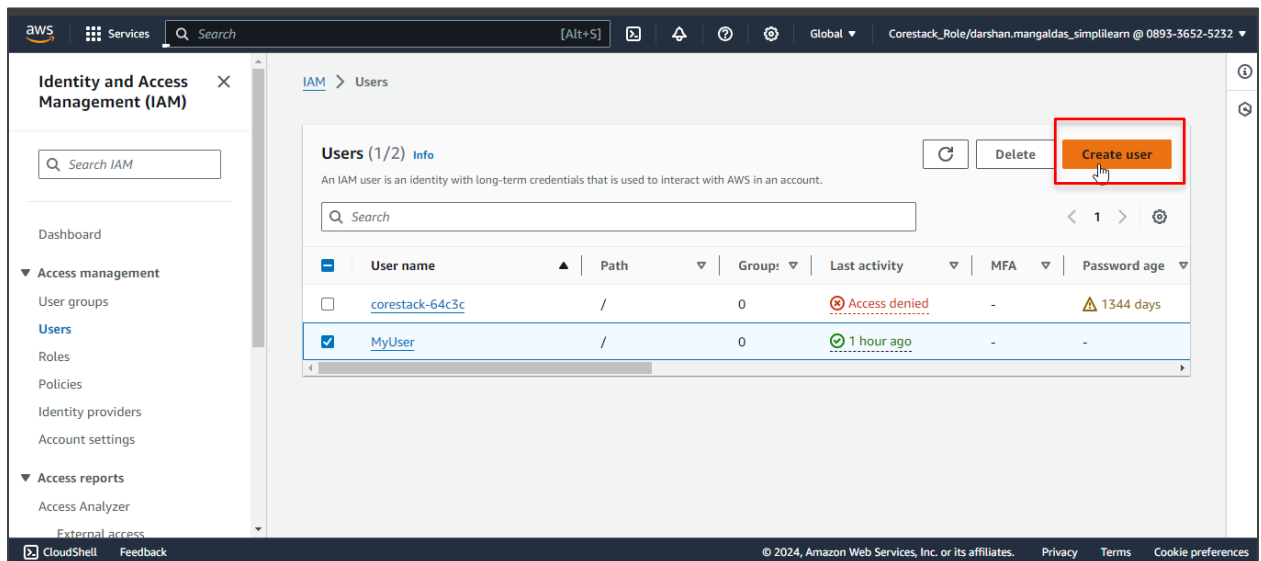
2.1 To set the access key and secret key, search for **IAM** on the AWS console



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



2.3 Click on the **Create user** button



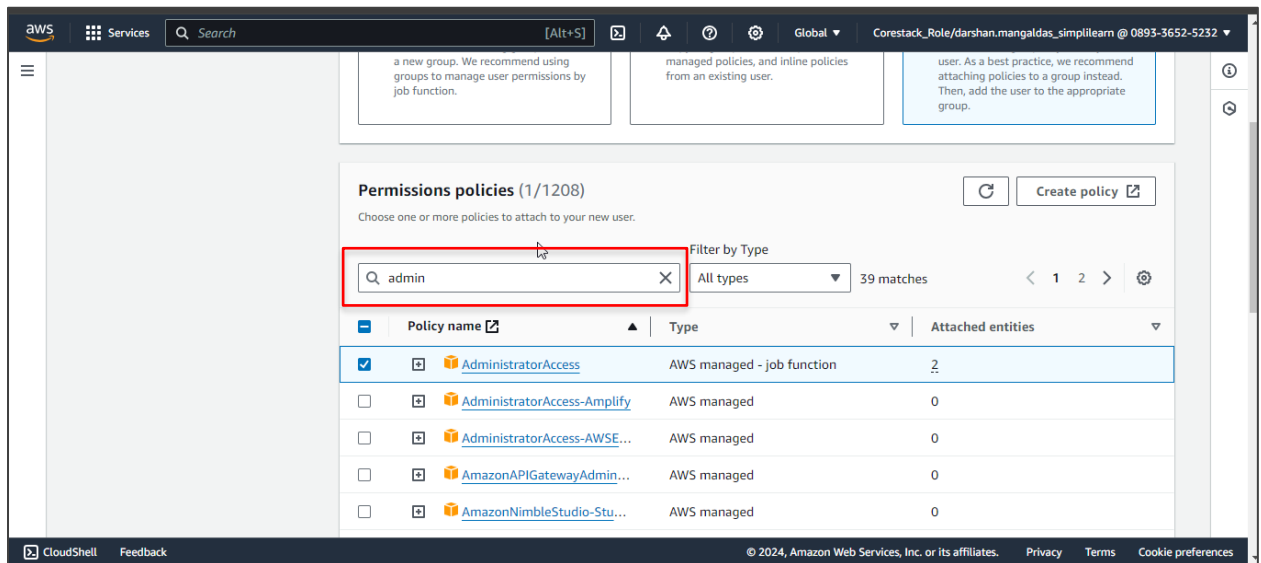
2.4 Enter **NewUser** under the **User name** field in the **User details** page and click on **Next**

The screenshot shows the 'Specify user details' page in the AWS IAM console. The 'User name' field is highlighted with a red box and contains the text 'NewUser'. Below the field, there is a note: 'The user name can have up to 64 characters. Valid characters include a-z, 0-9, and + = , . @ _ - (hyphen)'. There is also a checkbox for 'Provide user access to the AWS Management Console - optional' with a note: 'If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.' A blue box contains a tip: 'If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more'. The 'Next' button is highlighted with a red box.

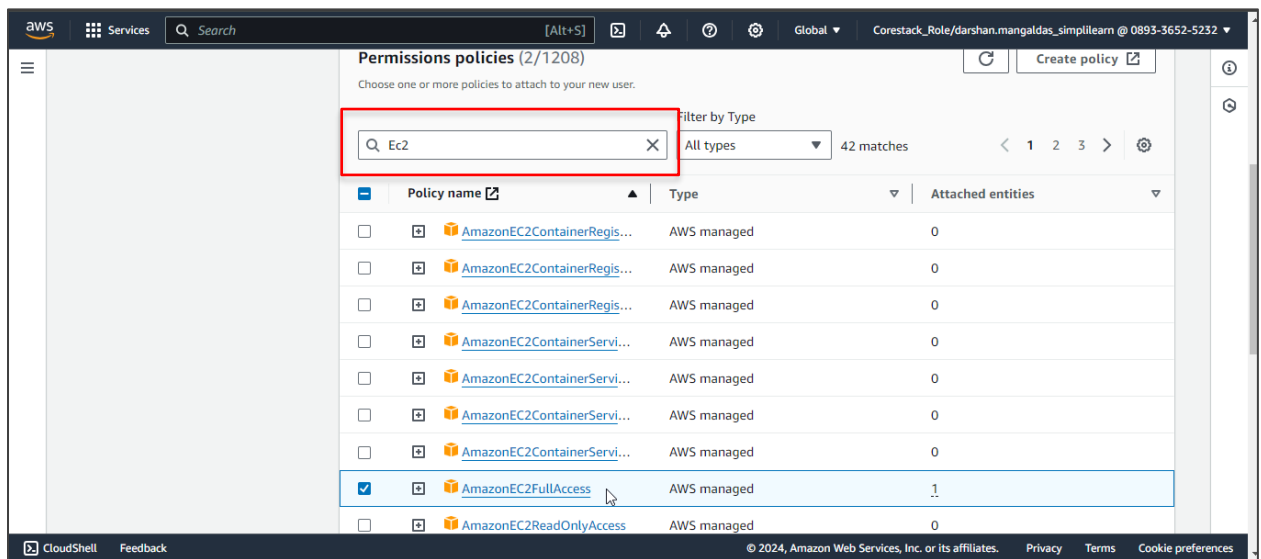
2.5 Select the **Attach policies directly** in the **Set permissions** page

The screenshot shows the 'Set permissions' page in the AWS IAM console. The 'Attach policies directly' option is highlighted with a red box. The page shows a sidebar with steps: Step 1 (Specify user details), Step 2 (Set permissions), and Step 3 (Review and create). The 'Permissions options' section has three radio buttons: 'Add user to group', 'Copy permissions', and 'Attach policies directly'. The 'Attach policies directly' option is selected. Below this, there is a section for 'Permissions policies (1208)' with a search bar and a filter by type dropdown. The 'Next' button is highlighted with a red box.

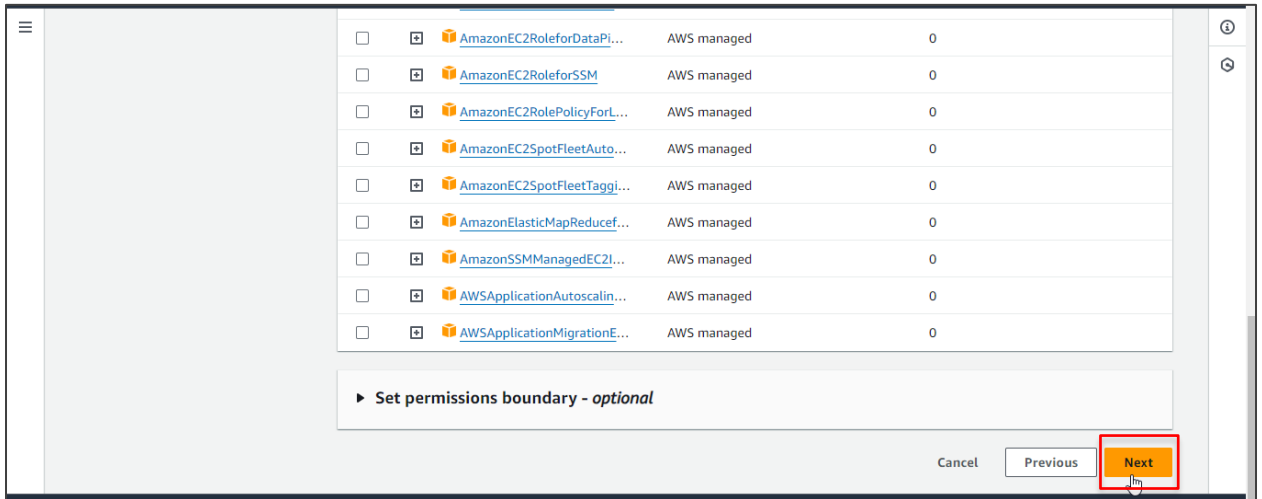
2.6 Scroll down to the **Permission policies** page, search for **admin** in the search box, and select **AdministratorAccess** under **Policy name**



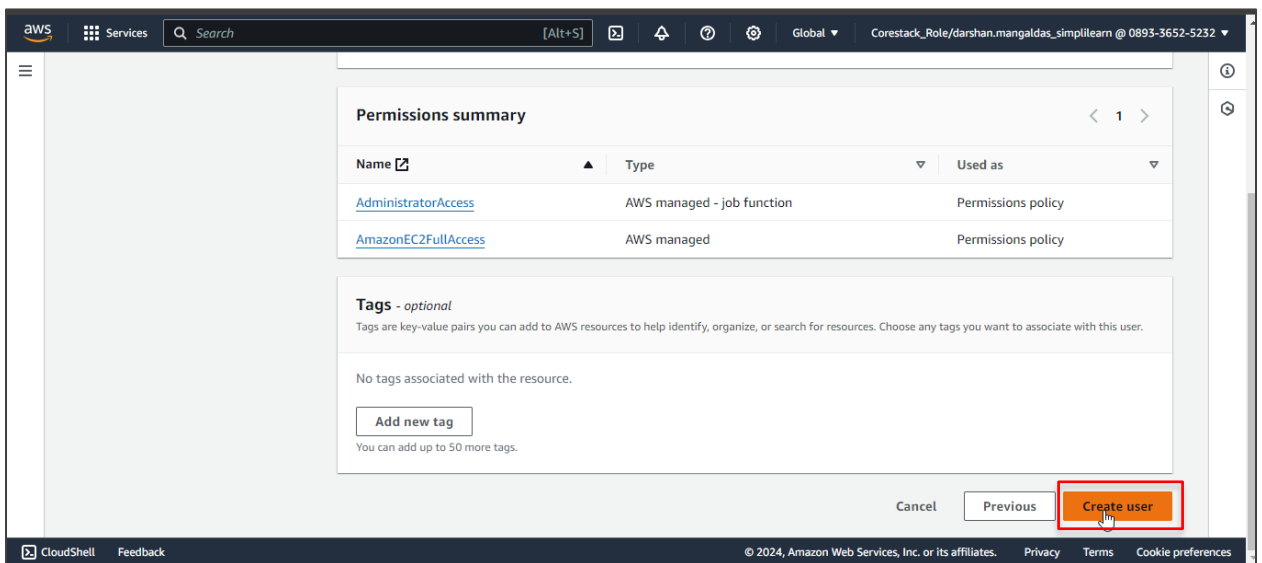
2.7 Search for **Ec2** in the search bar and select **AmazonEC2FullAccess** under **Policy name**



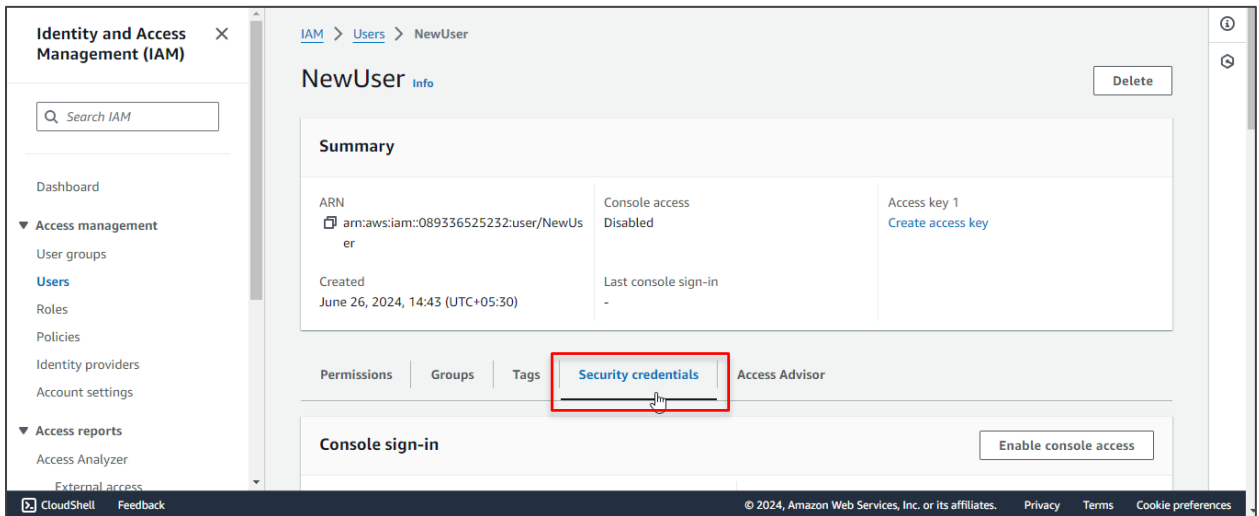
2.8 Scroll down and click on **Next**



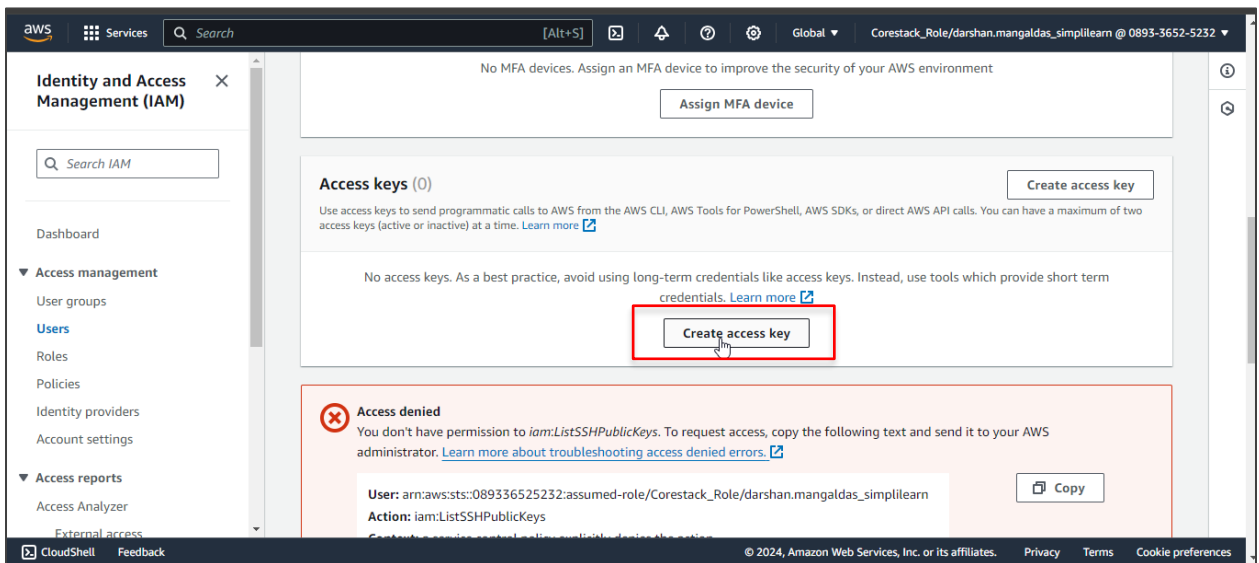
2.9 Click on **Create user** in the **Permissions summary** tab



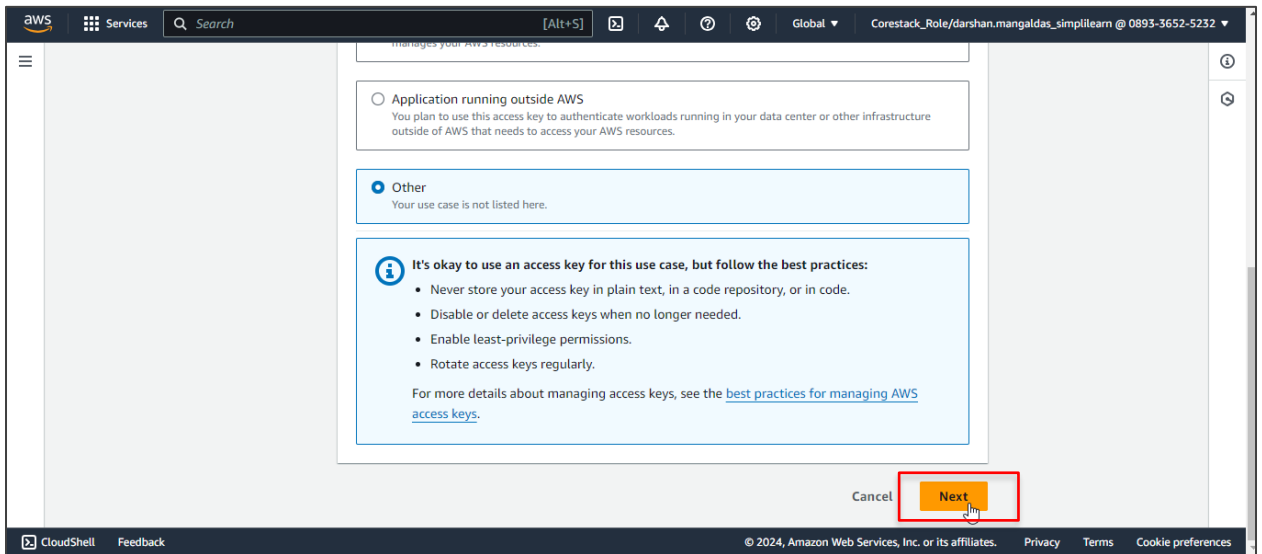
2.10 Click on **Security credentials** as shown in the screenshot below:



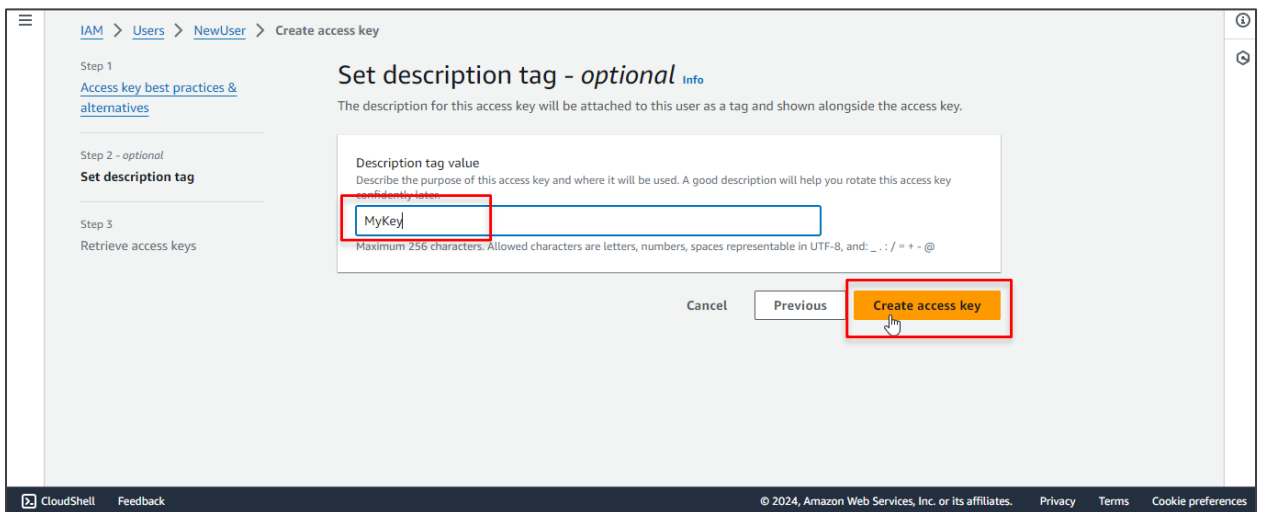
2.11 Click on **Create access key** as shown in the screenshot below:



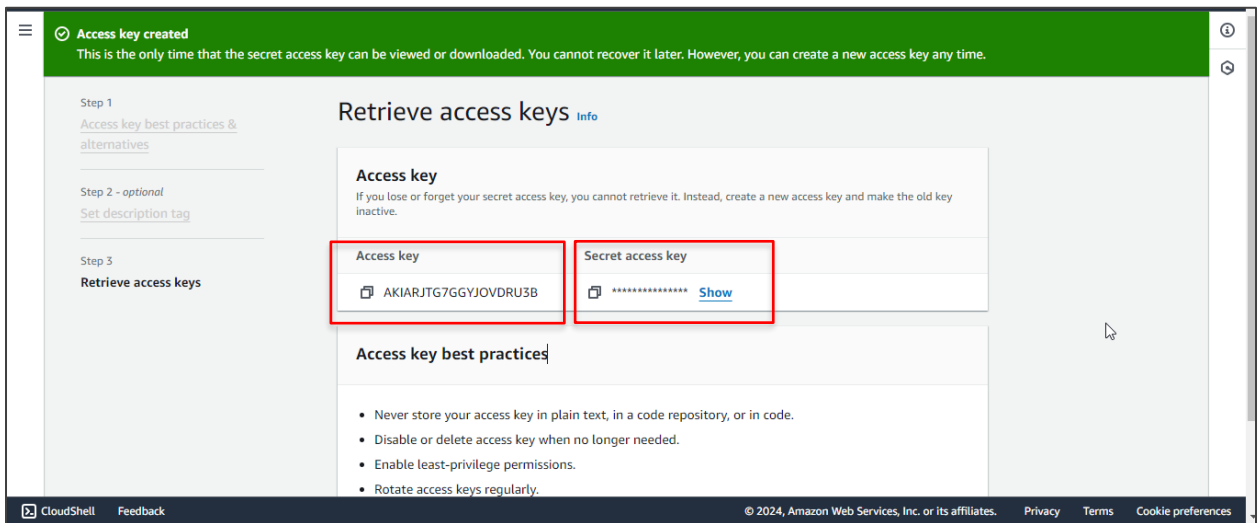
2.12 Scroll down, choose **Other**, and click on Next as shown in the screenshot below:



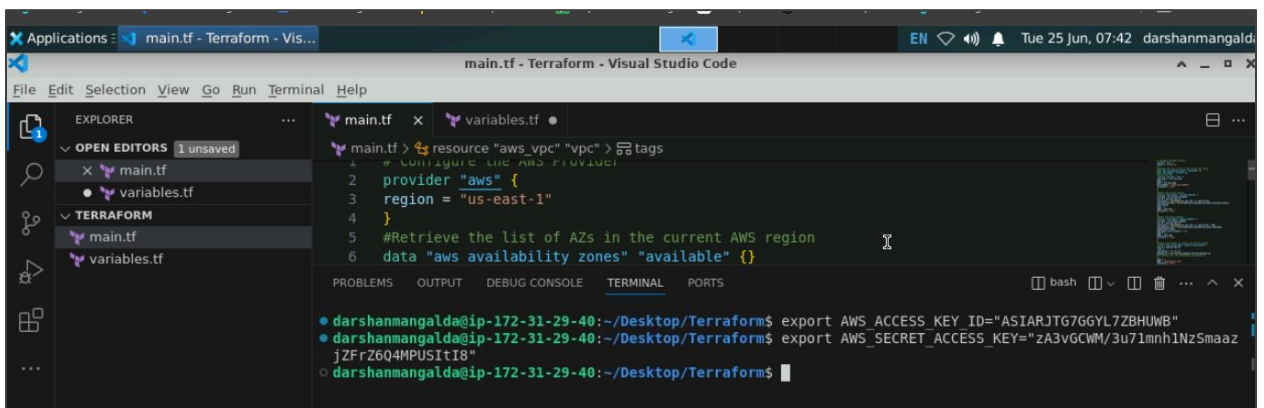
2.13 Enter a desired name in the **Description tag value** text area and then click on **Create access key**



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



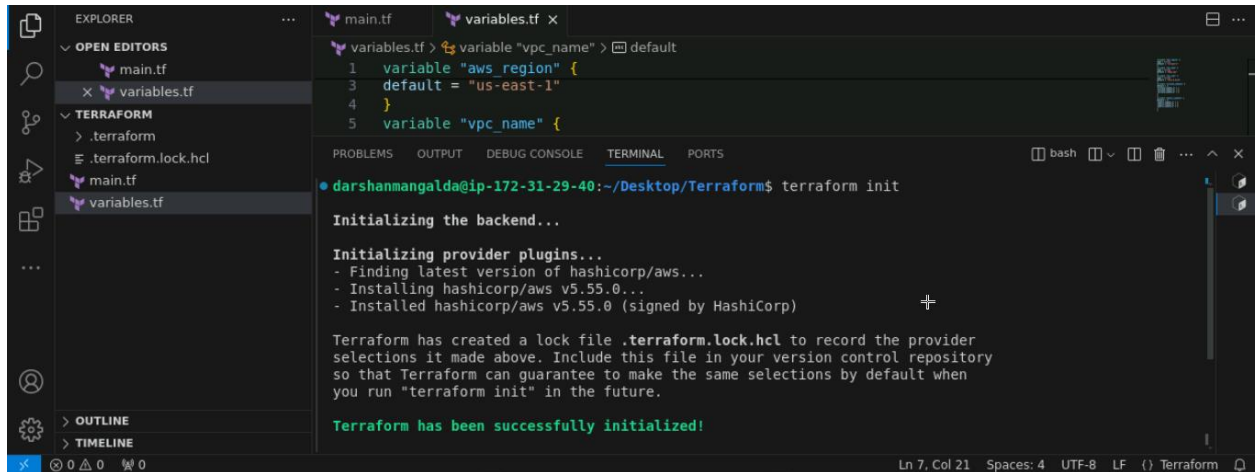
2.15 Once you have the credentials, go to the VS Code terminal and provide the given credentials **export AWS_ACCESS_KEY_ID= " "**
export AWS_SECRET_ACCESS_KEY=" " as shown in the screenshot below and click on Enter



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 screenshot shows a VS Code editor with two files open: `main.tf` and `variables.tf`. The `variables.tf` file contains the following content:

```
1 variable "vpc_name" {
2   default = "vpc-1"
3 }
4
5 variable "aws_region" {
6   default = "us-east-1"
7 }
```

The terminal window shows the output of the `terraform init` command:

```
darshanmangalada@ip-172-31-29-40:~/Desktop/Terraform$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.55.0...
- Installed hashicorp/aws v5.55.0 (signed by HashiCorp)

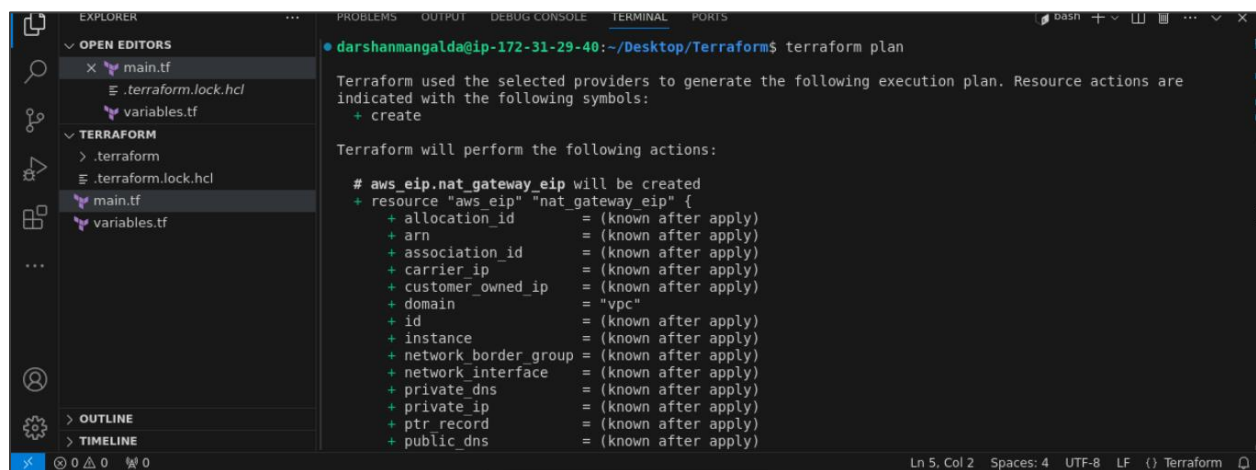
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!
```

The Terraform configuration file is successfully initialized.

3.2 Execute the given command to create an execution plan:

terraform plan



The screenshot shows the VS Code editor with the `terraform plan` command executed in the terminal. The output is as follows:

```
darshanmangalada@ip-172-31-29-40:~/Desktop/Terraform$ terraform plan

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

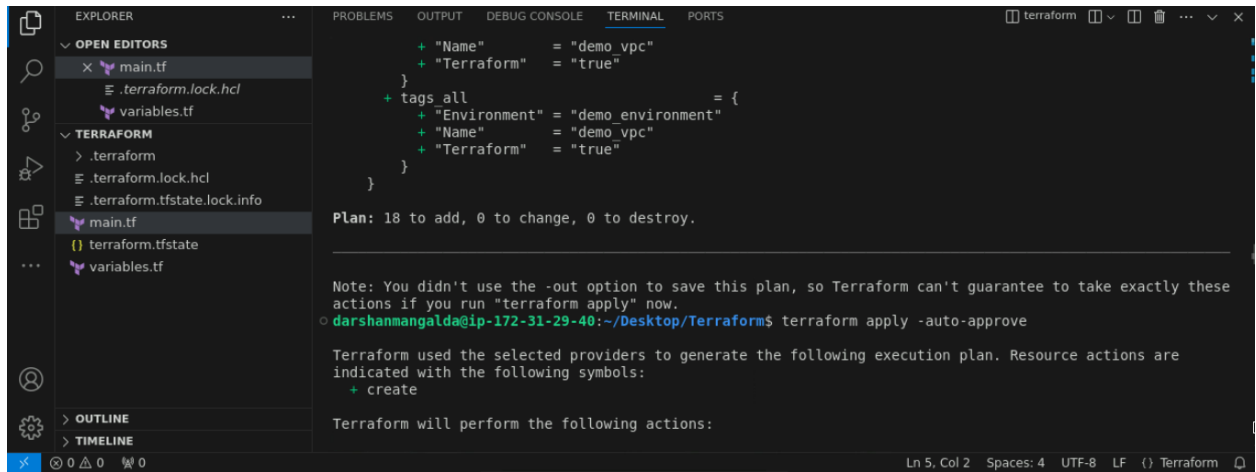
Terraform will perform the following actions:

# aws_eip.nat_gateway_eip will be created
+ resource "aws_eip" "nat_gateway_eip" {
  + allocation_id = (known after apply)
  + arn            = (known after apply)
  + association_id = (known after apply)
  + carrier_ip     = (known after apply)
  + customer_owned_ip = (known after apply)
  + domain         = "vpc"
  + id            = (known after apply)
  + instance      = (known after apply)
  + network_border_group = (known after apply)
  + network_interface = (known after apply)
  + private_dns     = (known after apply)
  + private_ip      = (known after apply)
  + ptr_record      = (known after apply)
  + public_dns      = (known after apply)
}
```

The execution plan is successfully created.

3.3 Enter the following command to execute the changes:

terraform apply -auto-approve



The screenshot shows the VS Code interface with the Explorer panel on the left displaying the project structure. The main editor shows the Terraform plan output in the terminal. The plan indicates that 18 resources will be added, 0 will be changed, and 0 will be destroyed. The plan is for a VPC named 'demo_vpc' with tags for 'Environment' and 'Terraform'.

```
+ "Name"      = "demo_vpc"
+ "Terraform" = "true"
}
+ tags_all    = {
+   "Environment" = "demo_environment"
+   "Name"        = "demo_vpc"
+   "Terraform"   = "true"
+ }
}
```

Plan: 18 to add, 0 to change, 0 to destroy.

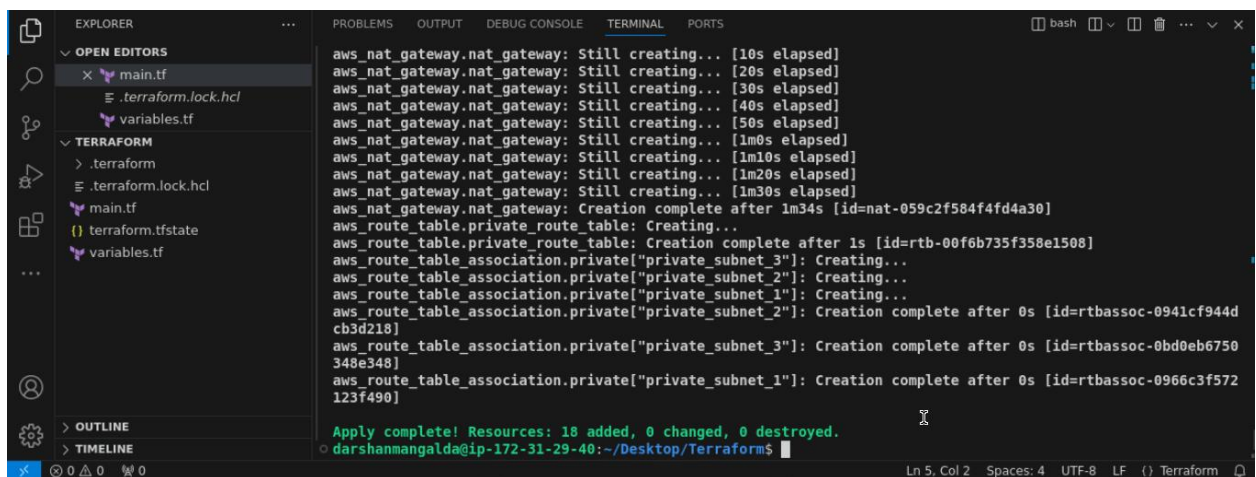
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

darshanmangalda@ip-172-31-29-40:~/Desktop/Terraform\$ terraform apply -auto-approve

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:



The screenshot shows the VS Code interface with the Explorer panel on the left displaying the project structure. The main editor shows the Terraform apply output in the terminal. The output indicates that the resources were successfully created, including the VPC, route table, and route table associations.

```
aws_nat_gateway.nat_gateway: Still creating... [10s elapsed]
aws_nat_gateway.nat_gateway: Still creating... [20s elapsed]
aws_nat_gateway.nat_gateway: Still creating... [30s elapsed]
aws_nat_gateway.nat_gateway: Still creating... [40s elapsed]
aws_nat_gateway.nat_gateway: Still creating... [50s elapsed]
aws_nat_gateway.nat_gateway: Still creating... [1m0s elapsed]
aws_nat_gateway.nat_gateway: Still creating... [1m10s elapsed]
aws_nat_gateway.nat_gateway: Still creating... [1m20s elapsed]
aws_nat_gateway.nat_gateway: Still creating... [1m30s elapsed]
aws_nat_gateway.nat_gateway: Creation complete after 1m34s [id=nat-059c2f584f4fd4a30]
aws_route_table.private_route_table: Creating...
aws_route_table.private_route_table: Creation complete after 1s [id=rtb-00f6b735f358e1508]
aws_route_table_association.private["private_subnet_3"]: Creating...
aws_route_table_association.private["private_subnet_2"]: Creating...
aws_route_table_association.private["private_subnet_1"]: Creating...
aws_route_table_association.private["private_subnet_2"]: Creation complete after 0s [id=rtbassoc-0941cf944dcb3d218]
aws_route_table_association.private["private_subnet_3"]: Creation complete after 0s [id=rtbassoc-0bd0eb6750348e348]
aws_route_table_association.private["private_subnet_1"]: Creation complete after 0s [id=rtbassoc-0966c3f572123f490]

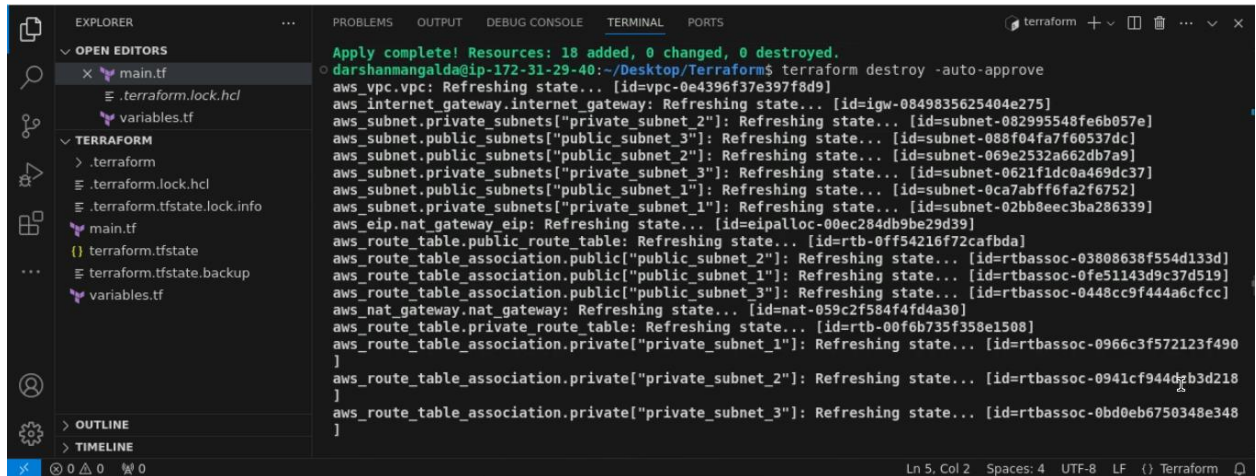
Apply complete! Resources: 18 added, 0 changed, 0 destroyed.
darshanmangalda@ip-172-31-29-40:~/Desktop/Terraform$
```

The changes are successfully executed.

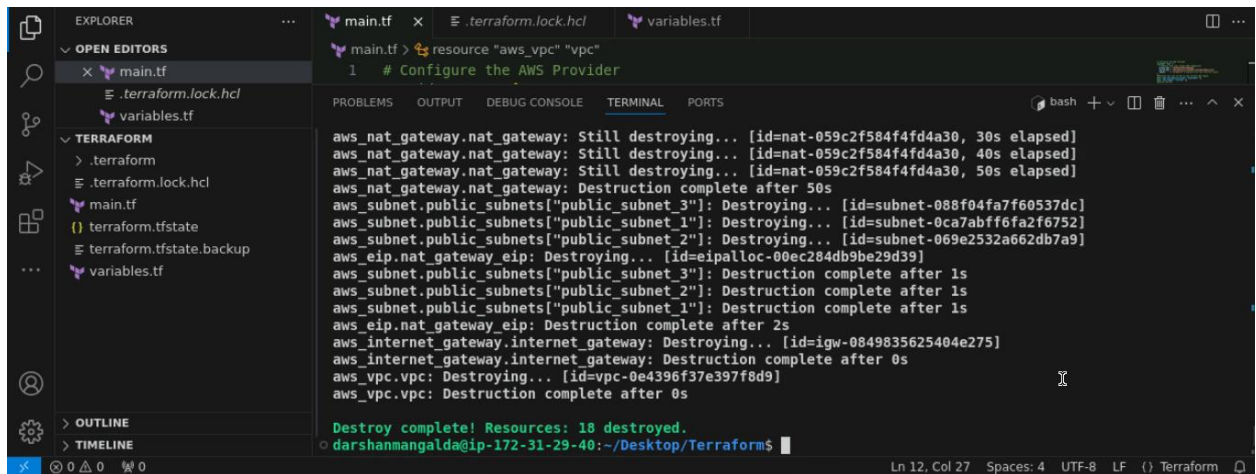
Step 4: Delete the AWS resources using Terraform to clean up your AWS environment

4.1 To destroy your resources, execute the following command in the terminal:

terraform destroy -auto-approve



```
Apply complete! Resources: 18 added, 0 changed, 0 destroyed.
darshanmangalada@ip-172-31-29-40: ~/Desktop/Terraform$ terraform destroy -auto-approve
aws_vpc.vpc: Refreshing state... [id=vpc-0e4396f37e397f8d9]
aws_internet_gateway.internet_gateway: Refreshing state... [id=igw-0849835625404e275]
aws_subnet.private_subnets["private_subnet_2"]: Refreshing state... [id=subnet-082995548fe6b057e]
aws_subnet.public_subnets["public_subnet_3"]: Refreshing state... [id=subnet-088f04fa7f60537dc]
aws_subnet.public_subnets["public_subnet_2"]: Refreshing state... [id=subnet-069e2532a662db7a9]
aws_subnet.private_subnets["private_subnet_3"]: Refreshing state... [id=subnet-0621f1dc0a469dc37]
aws_subnet.private_subnets["private_subnet_1"]: Refreshing state... [id=subnet-0ca7abff6a2f6752]
aws_subnet.private_subnets["private_subnet_1"]: Refreshing state... [id=subnet-02bb8eec3ba286339]
aws_eip.nat_gateway_eip: Refreshing state... [id=eipalloc-00ec284db9be29d39]
aws_route_table.public_route_table: Refreshing state... [id=rtb-0ff54216f72cafbda]
aws_route_table_association.public["public_subnet_2"]: Refreshing state... [id=rtbassoc-03808638f554d133d]
aws_route_table_association.public["public_subnet_1"]: Refreshing state... [id=rtbassoc-0fe51143d9c37d519]
aws_route_table_association.public["public_subnet_3"]: Refreshing state... [id=rtbassoc-0448cc9f444a6cfc]
aws_nat_gateway.nat_gateway: Refreshing state... [id=nat-059c2f584f4fd4a30]
aws_route_table.private_route_table: Refreshing state... [id=rtb-00f6b735f358e1508]
aws_route_table_association.private["private_subnet_1"]: Refreshing state... [id=rtbassoc-0966c3f572123f490]
aws_route_table_association.private["private_subnet_2"]: Refreshing state... [id=rtbassoc-0941cf944db3d218]
aws_route_table_association.private["private_subnet_3"]: Refreshing state... [id=rtbassoc-0bd0eb6750348e348]
```



```
aws_nat_gateway.nat_gateway: Still destroying... [id=nat-059c2f584f4fd4a30, 30s elapsed]
aws_nat_gateway.nat_gateway: Still destroying... [id=nat-059c2f584f4fd4a30, 40s elapsed]
aws_nat_gateway.nat_gateway: Still destroying... [id=nat-059c2f584f4fd4a30, 50s elapsed]
aws_nat_gateway.nat_gateway: Destruction complete after 50s
aws_subnet.public_subnets["public_subnet_3"]: Destroying... [id=subnet-088f04fa7f60537dc]
aws_subnet.public_subnets["public_subnet_1"]: Destroying... [id=subnet-0ca7abff6a2f6752]
aws_subnet.public_subnets["public_subnet_2"]: Destroying... [id=subnet-069e2532a662db7a9]
aws_eip.nat_gateway_eip: Destroying... [id=eipalloc-00ec284db9be29d39]
aws_subnet.public_subnets["public_subnet_3"]: Destruction complete after 1s
aws_subnet.public_subnets["public_subnet_2"]: Destruction complete after 1s
aws_subnet.public_subnets["public_subnet_1"]: Destruction complete after 1s
aws_eip.nat_gateway_eip: Destruction complete after 2s
aws_internet_gateway.internet_gateway: Destroying... [id=igw-0849835625404e275]
aws_internet_gateway.internet_gateway: Destruction complete after 0s
aws_vpc.vpc: Destroying... [id=vpc-0e4396f37e397f8d9]
aws_vpc.vpc: Destruction complete after 0s

Destroy complete! Resources: 18 destroyed.
darshanmangalada@ip-172-31-29-40: ~/Desktop/Terraform$
```

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.