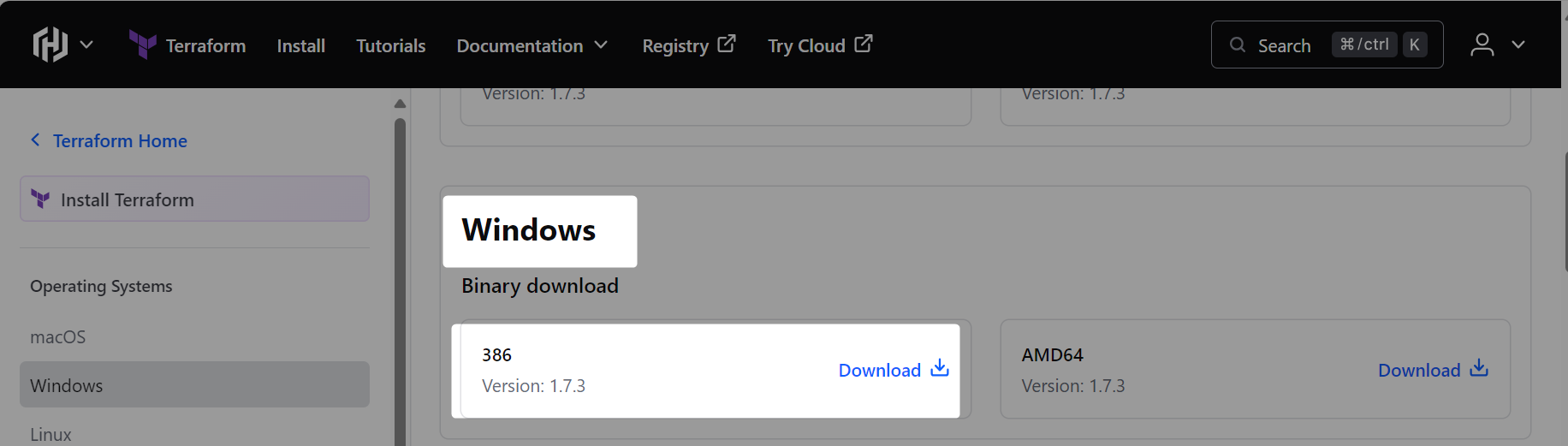
Installation Terraform

DAY1

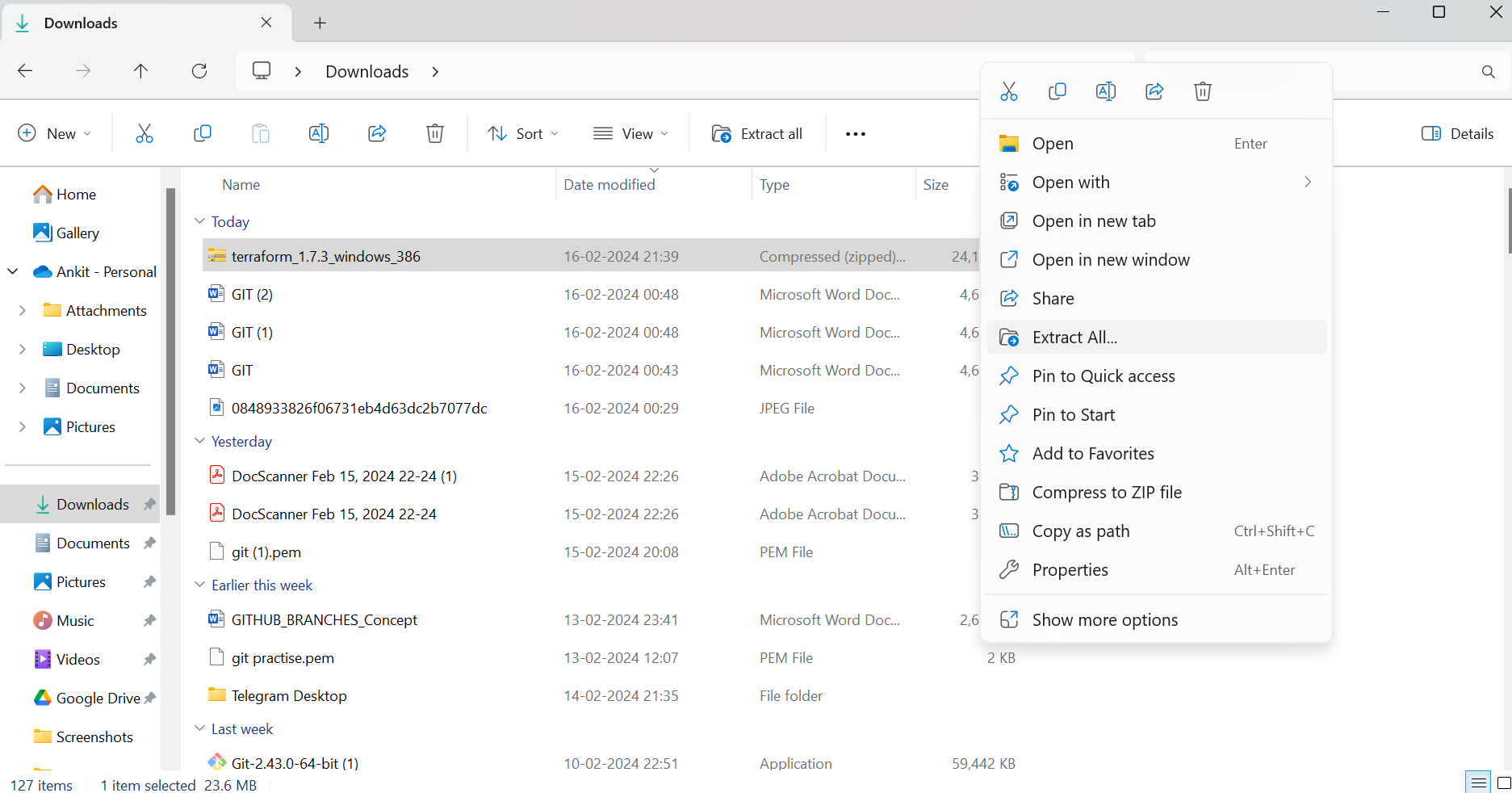
Install Terraform on your local system

Step 1: Click the link – <https://developer.hashicorp.com/terraform/install>

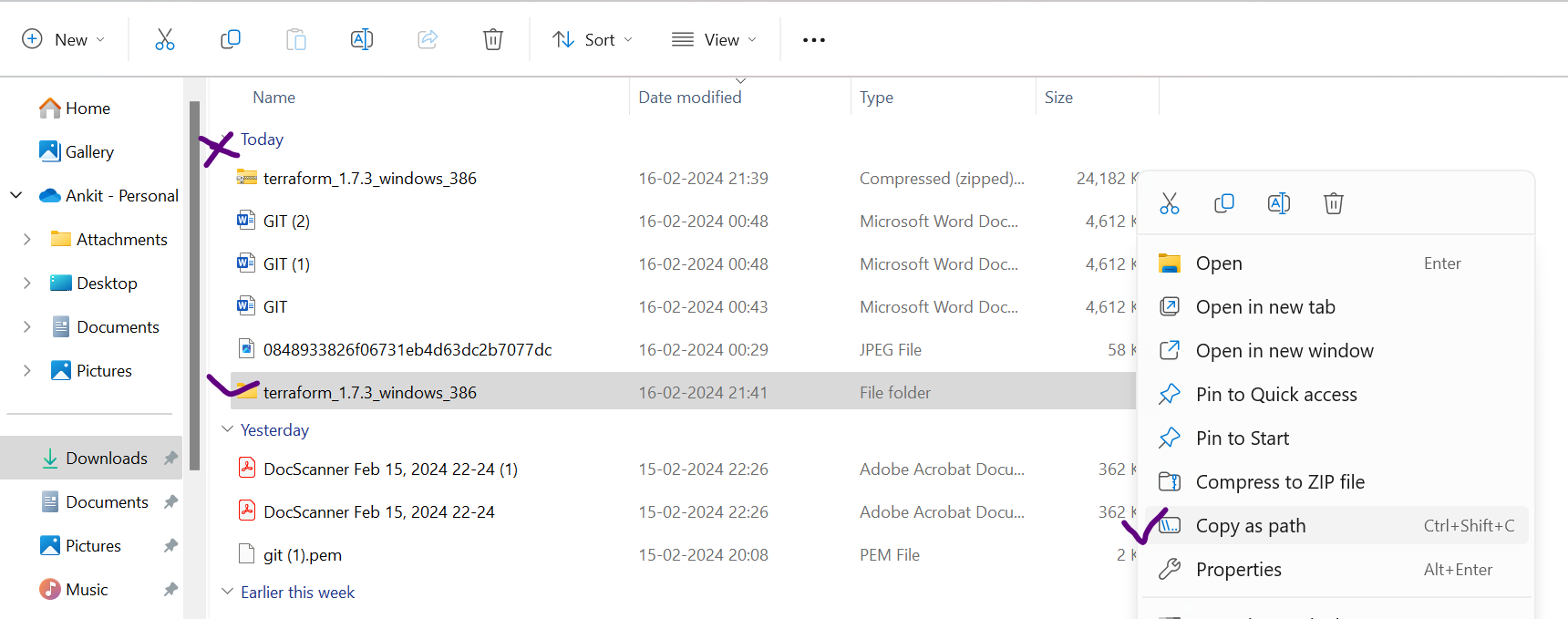
Step 2: select window >386 >download



Step3: Extract all from download

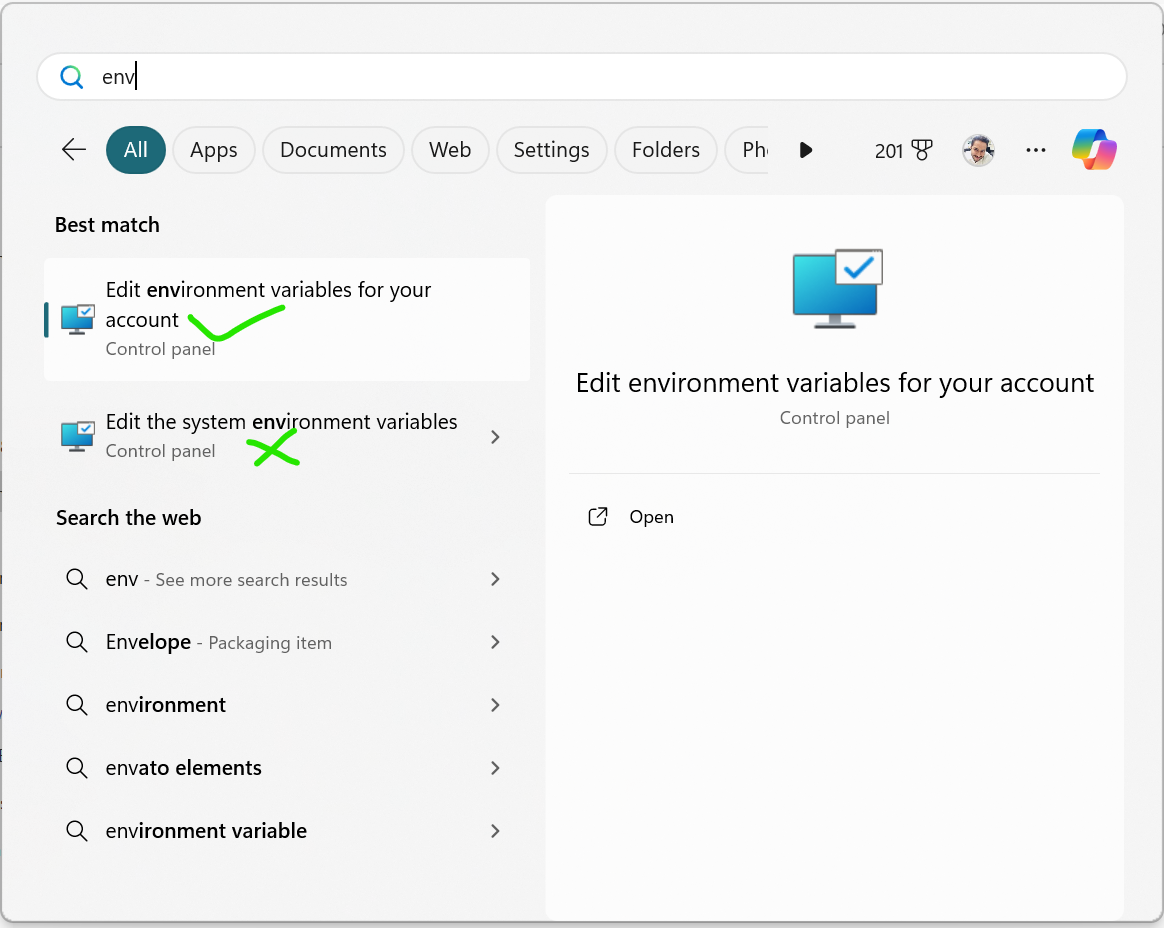


Step4: after extract copy the full path

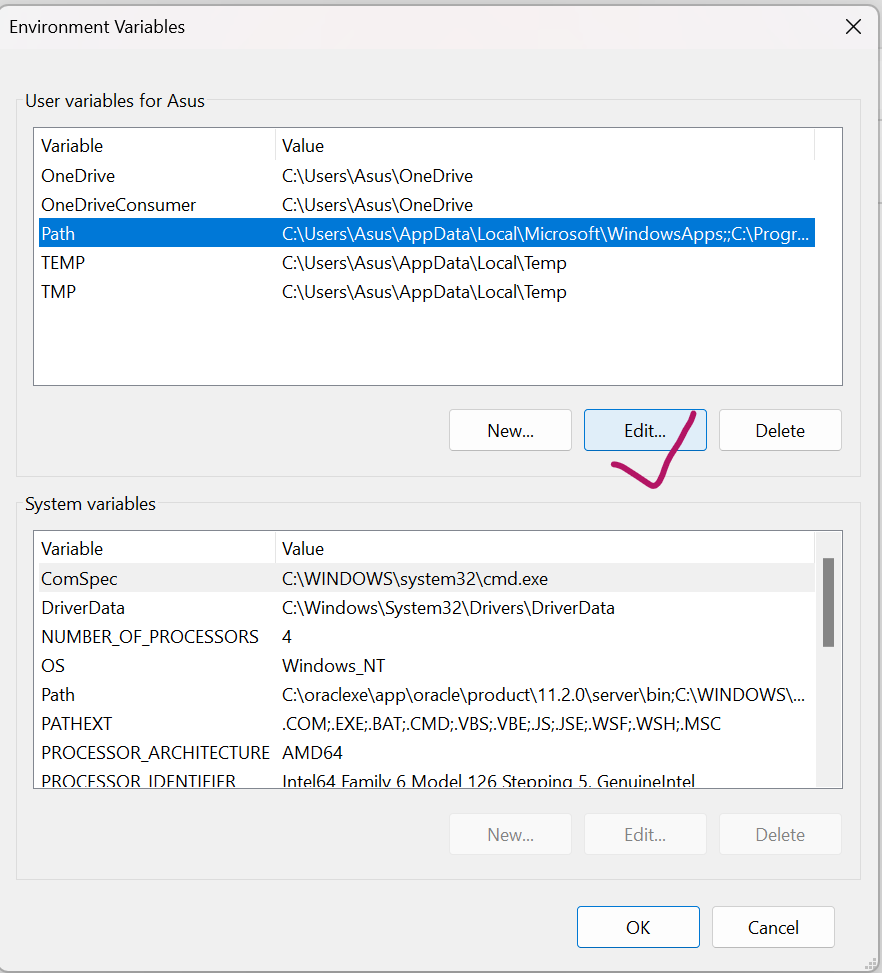


C:\Users\Asus\Downloads\terraform\_1.7.3\_windows\_386

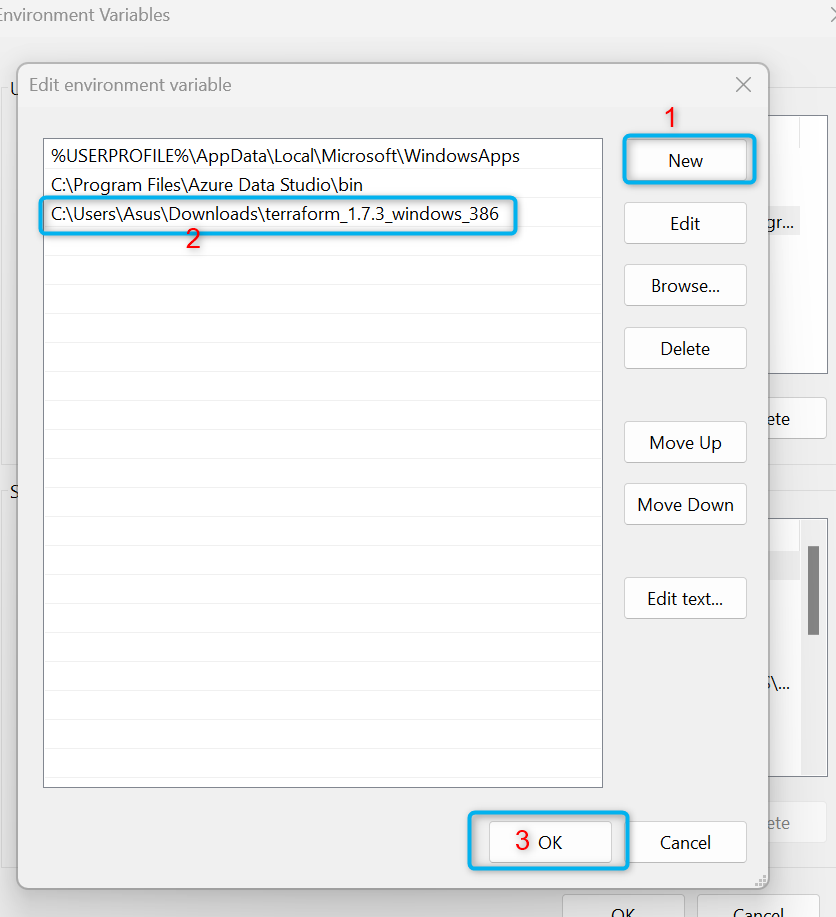
Step 5: click on Edit environment variables for your account



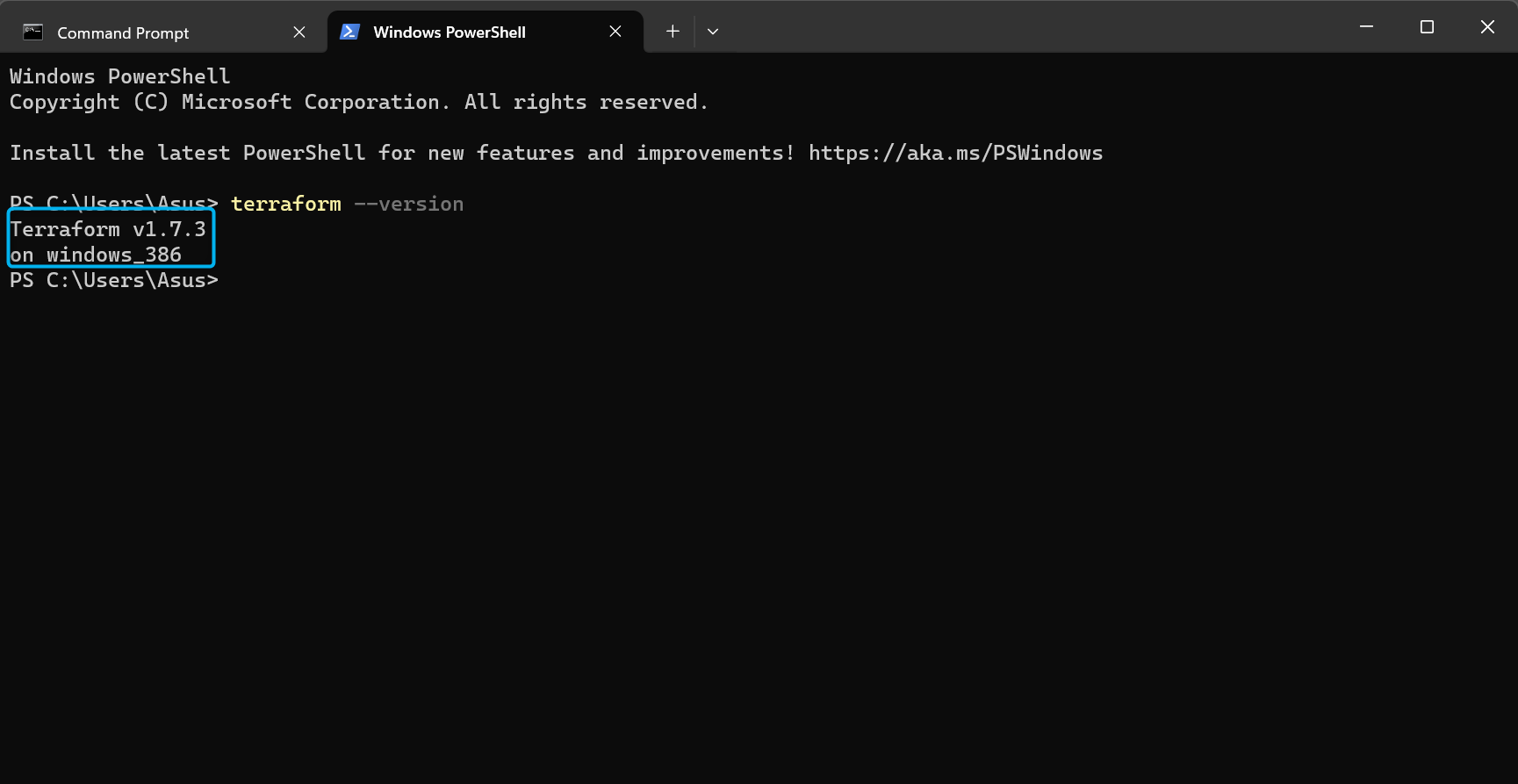
Step 6: click on path and edit



Step 7 : click on new > paste the path > ok



Step 8: open cmd & check version



Terraform Codes

DAY 2

**#---create custom network and custom ec2 instance-----------**

* Custom Network

1st block : provider.tf

provider "aws" {

    access\_key = "AKIA4HJWDM3GLTF7HTUH"

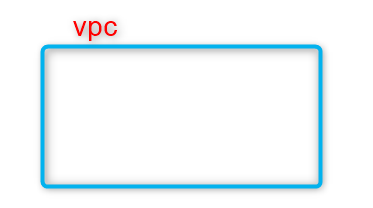
    secret\_key = "SW0nDd9au1JGcC4z+FbSUCXngyyTEqy9AjP6NXWI"

    region = "us-east-1"

}

2nd block : Main.tf

#create vpc

****

resource "aws\_vpc" "custnw" {

cidr\_block = "10.0.0.0/16"

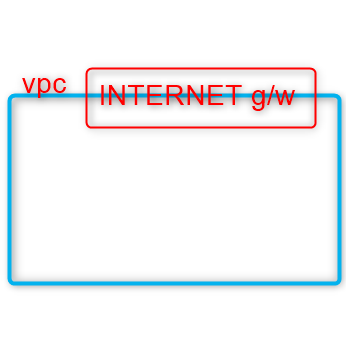
tags = {

Name = "ankit\_vpc"

}

}

#Create Internet Gateway and attach to VPC



resource "aws\_internet\_gateway" "custnw" {

vpc\_id = aws\_vpc.custnw.id

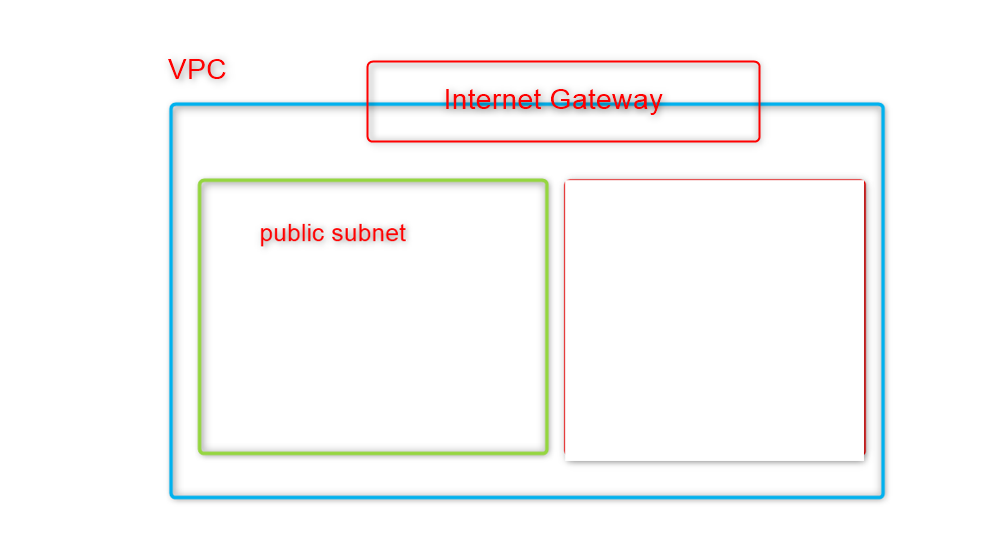
tags = {

Name = "Ankit Internet Gateway"

}

}

#Create subnet & attach to vpc



resource "aws\_subnet" "custnw" {

vpc\_id = aws\_vpc.custnw.id

cidr\_block = "10.0.0.0/24"

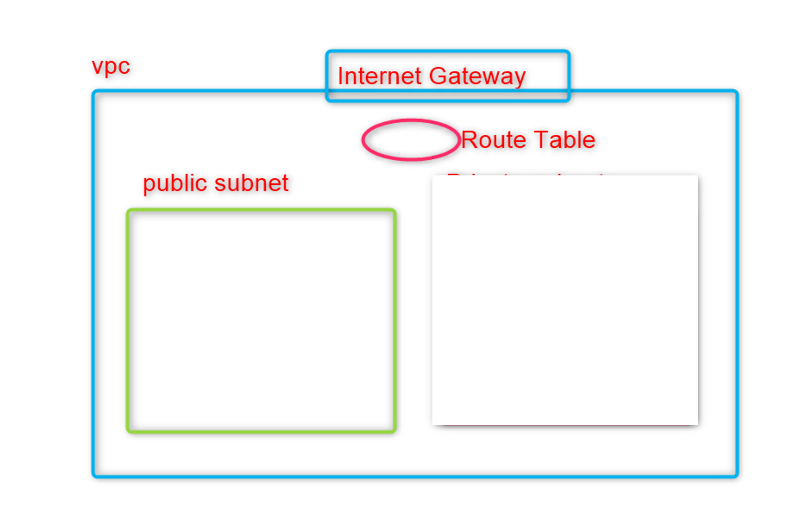
tags = {

Name = "Ankit subnet"

}

}

#Create RT and attach to vpc



resource "aws\_route\_table" "custnw" {

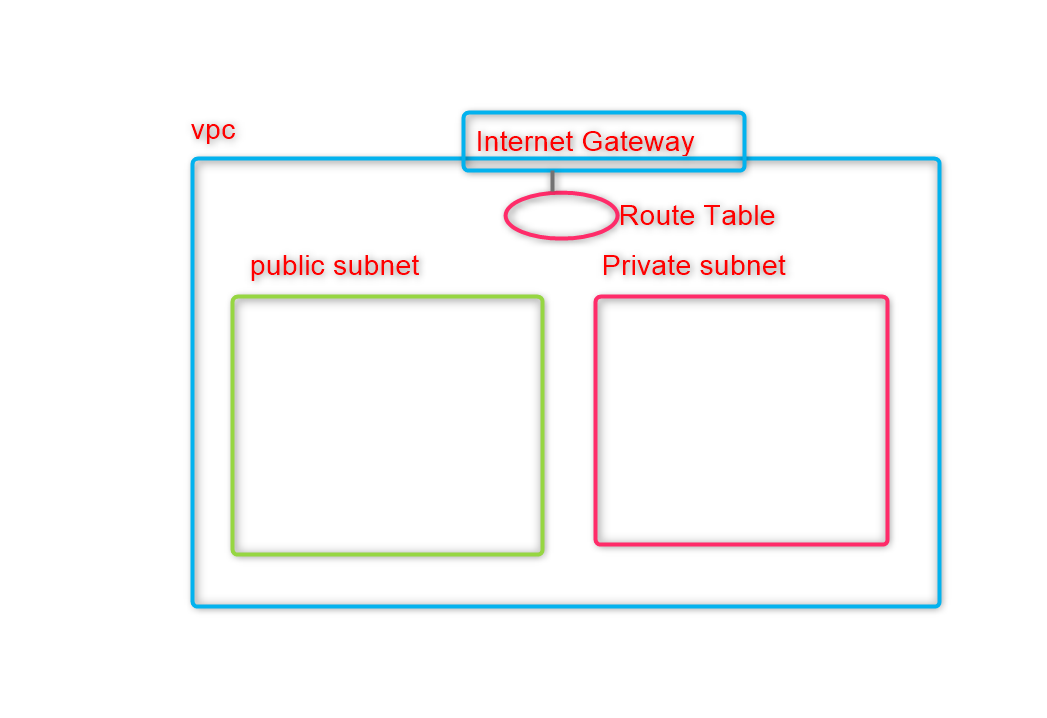
vpc\_id = aws\_vpc.custnw.id

tags = {

Name = "Ankit Rt"

}

#associate route table with internetgateway

****

route {

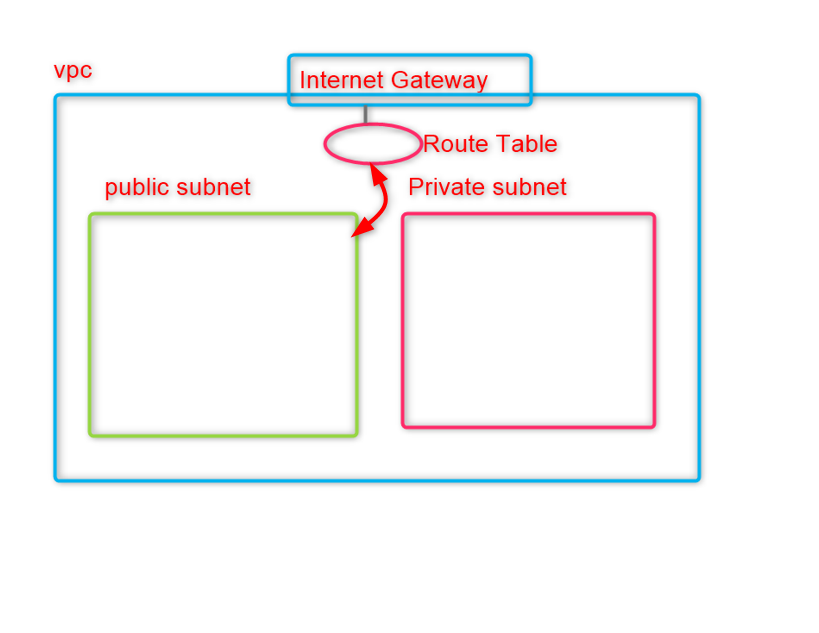
cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.custnw.id

}

}

#associate route table with subnet



resource "aws\_route\_table\_association" "custnw" {

route\_table\_id = aws\_route\_table.custnw.id

subnet\_id = aws\_subnet.custnw.id

}

#cust security group

resource "aws\_security\_group" "custnw\_sg" {

name = "custnw\_sg"

description = "Allow TLS inbound traffics"

vpc\_id = aws\_vpc.custnw.id

ingress {

description = "TLS from VPC"

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

description = "TLS from VPC"

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

description = "TLS from VPC"

from\_port = 443

to\_port = 443

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

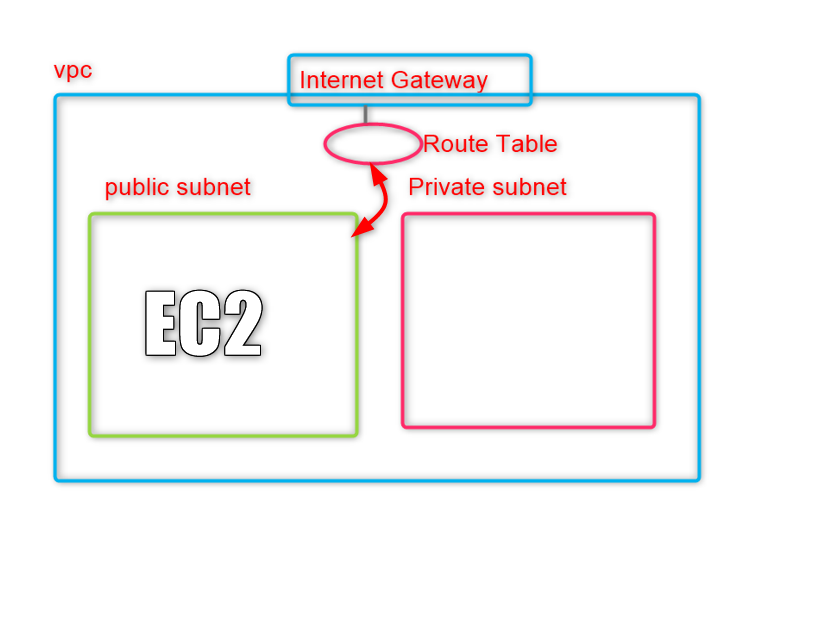
cidr\_blocks = ["0.0.0.0/0"]

}

}

* Custom EC2 Instance

#Create custom ec2 instance



resource "aws\_instance" "custnw" {

ami = var.ami

instance\_type = var.instance\_type

key\_name = var.key\_name

subnet\_id = aws\_subnet.custnw.id

associate\_public\_ip\_address = true

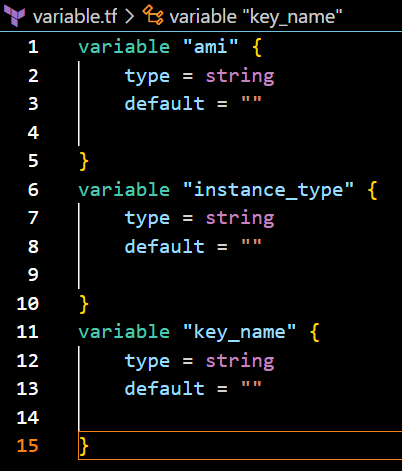
tags = {

Name = "CustANKITec2"

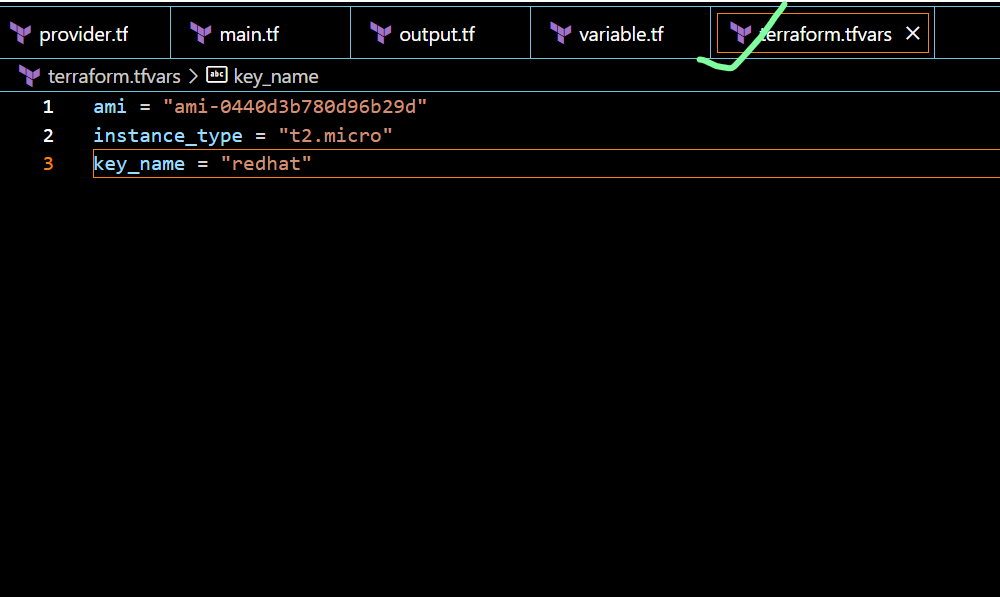
}

}

3rd block : Variable.tf



4th block : Terraform.tfvars



DAY 3

* S3 BUCKET CREATION WITH VERSIONING

2nd block : Main.tf

#Create S3 Bucket

resource "aws\_s3\_bucket" "devankit" {

bucket = "terrabucketcreate"

}

#Get version enabled of created s3 bucket

resource "aws\_s3\_bucket\_versioning" "versioning\_adhvikanand" {

bucket = aws\_s3\_bucket.devankit.id

versioning\_configuration {

status = "Enabled"

}

}

* OUTPUT BLOCK CODES AND SENSATIVE CONCEPT

2nd block : Main.tf

#Create the fresh EC2 instance and print the output of public ip, public dns and private ip dns

++Don't print output of privateip\_by using sensative.

resource "aws\_instance" "MrSingh" {

ami = var.ami

instance\_type = var.instance\_type

key\_name = var.key\_name

tags = {

Name = "MrSinghec2"

}

}

#to print output, we have written code in output.tf

5th block : Output.tf

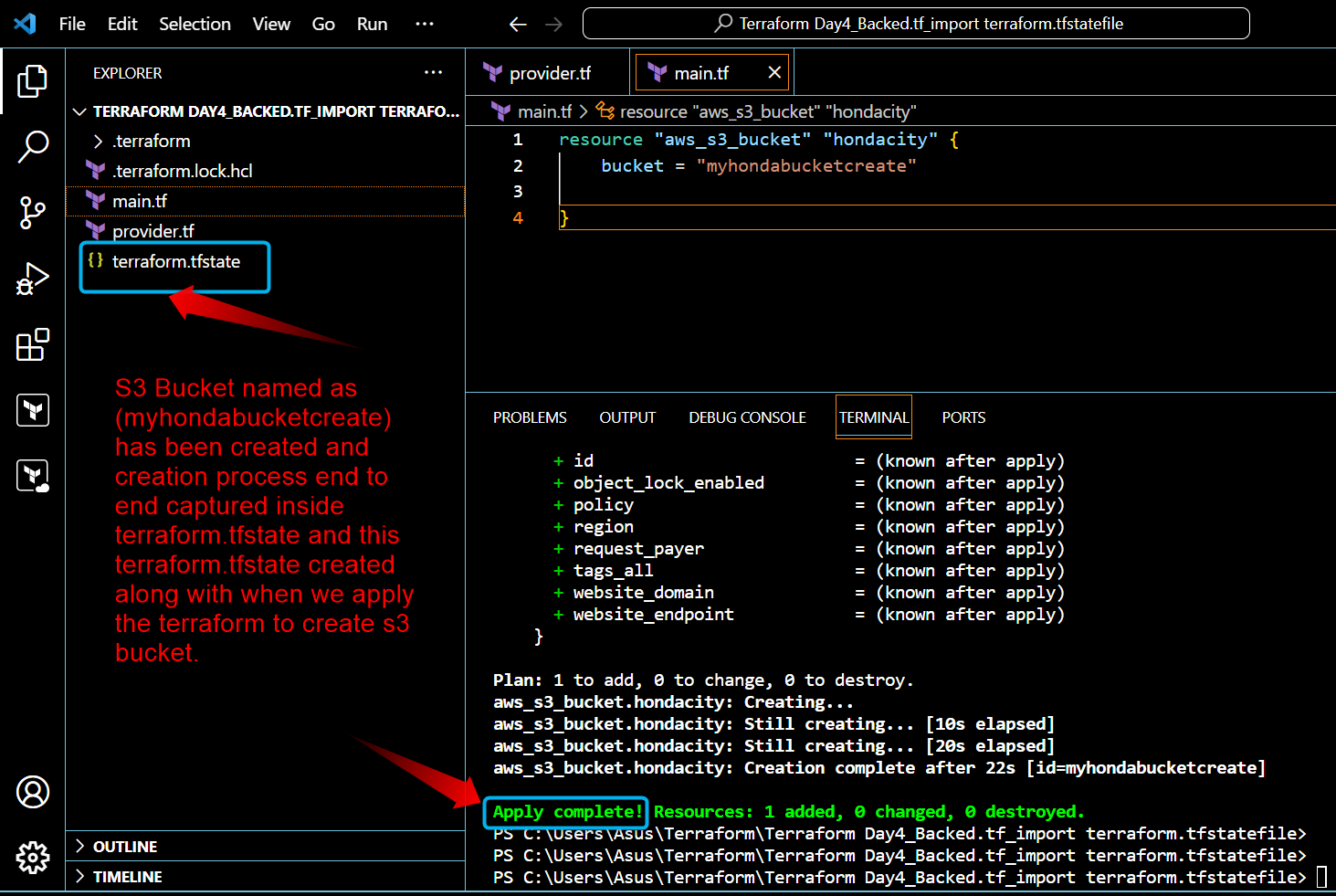


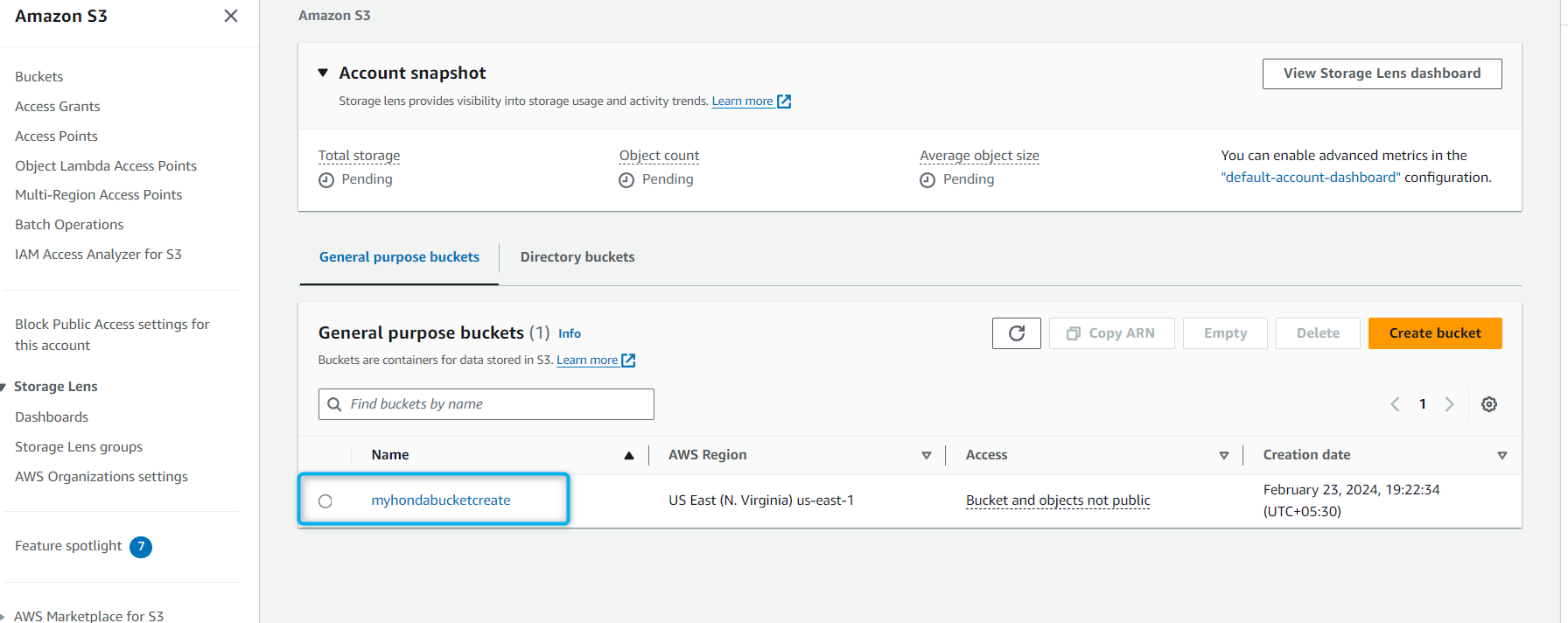


DAY 4

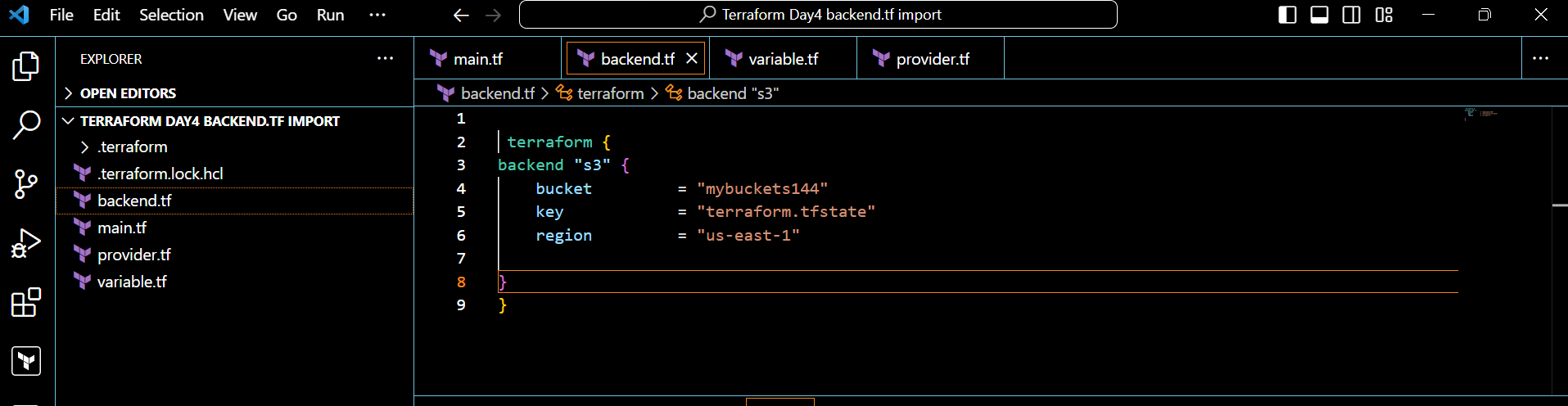
* Backend.tf script

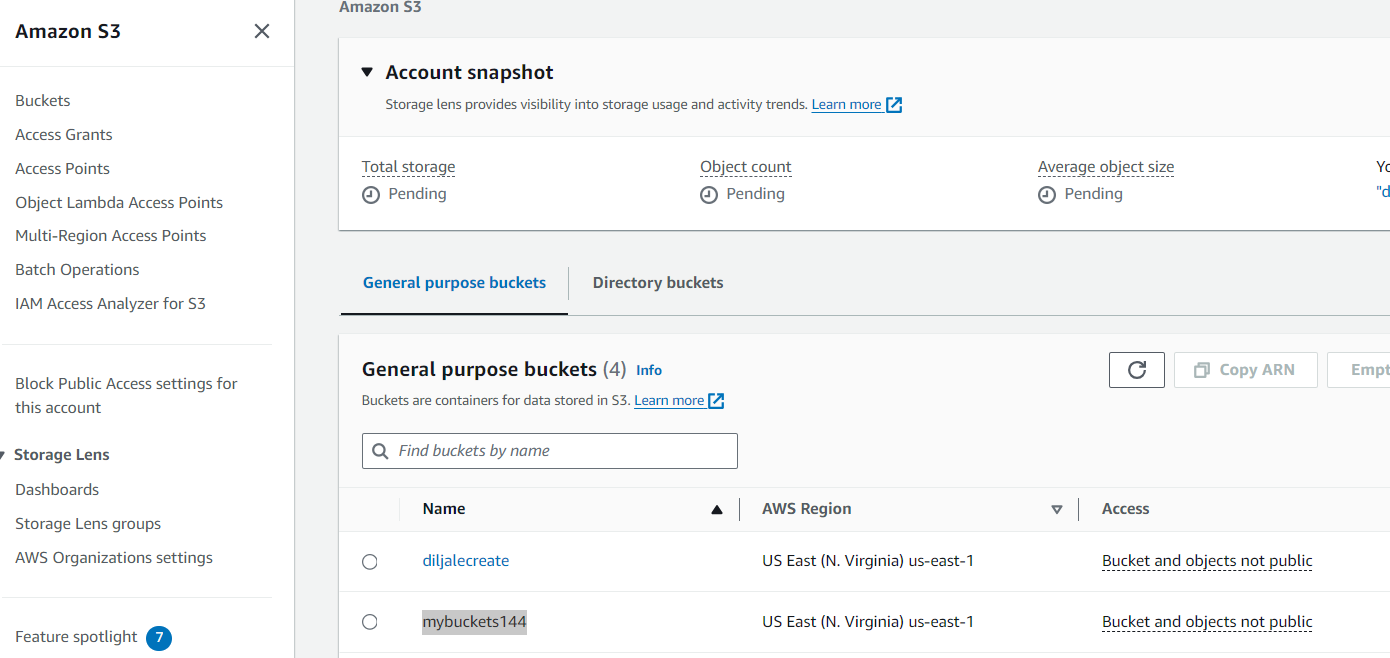
#we are creating one S3 Bucket and try to see the whole creation process inside terraform.tfstate.





#terraform.tfstate can be vanished or it will not get seen into the local as above by using configuring backend.tf block. Means after we do terraform apply terraform.tfstate will get created and it will capture also the running process whatever any creation deletion any ongoing process it will able to capture but it will not located into local as above it will get located into backend.tf



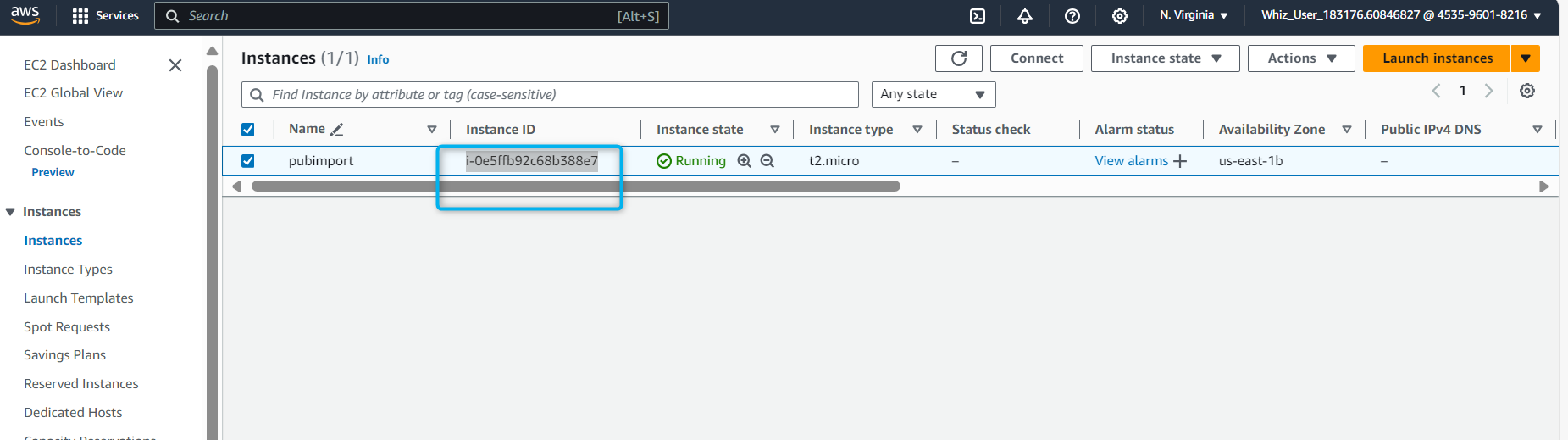


DAY 5

* IMPORT : import resource into terraform

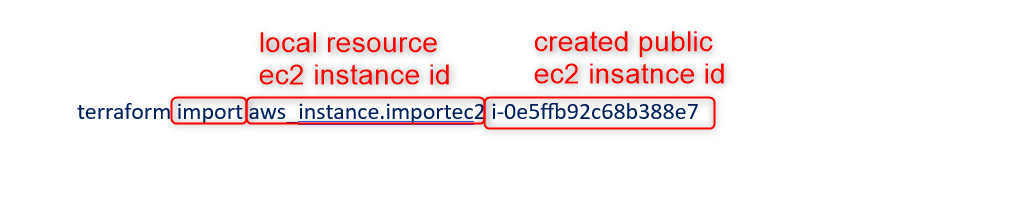
To do any further changes in created ec2 instance we import or clone to our local system and control the main.tf for further changes for ec2 instance.

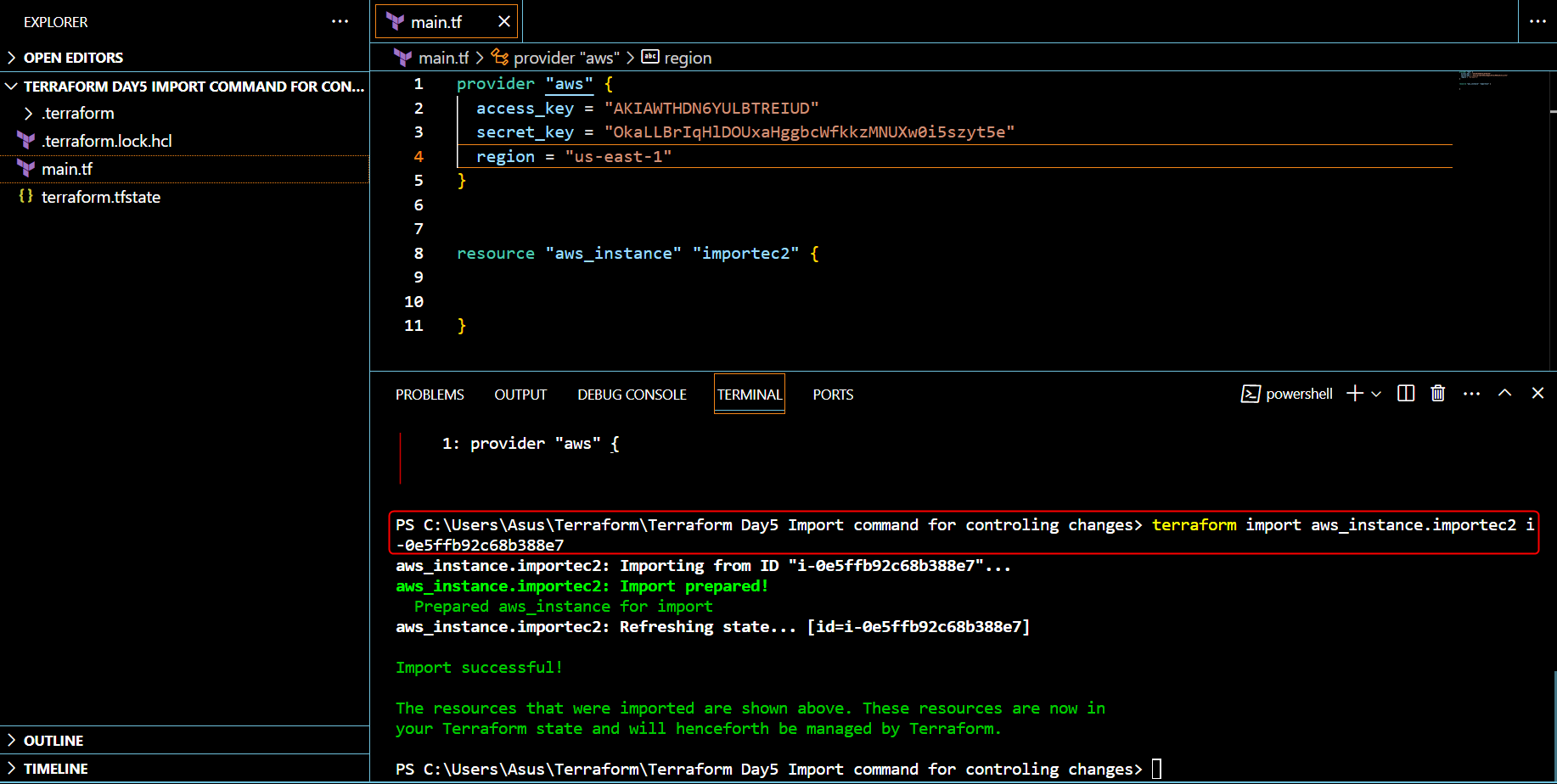
First we create a resource block before that we will create a ec2 instance



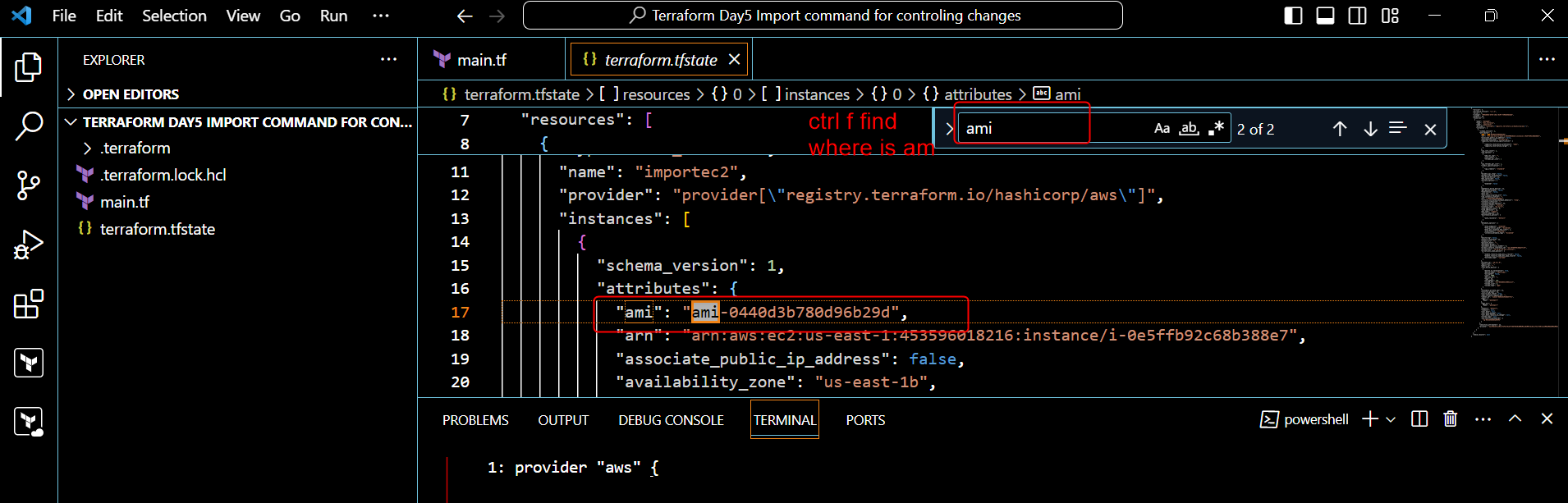
Now we will map the ec instance id with our local ec2 resource block

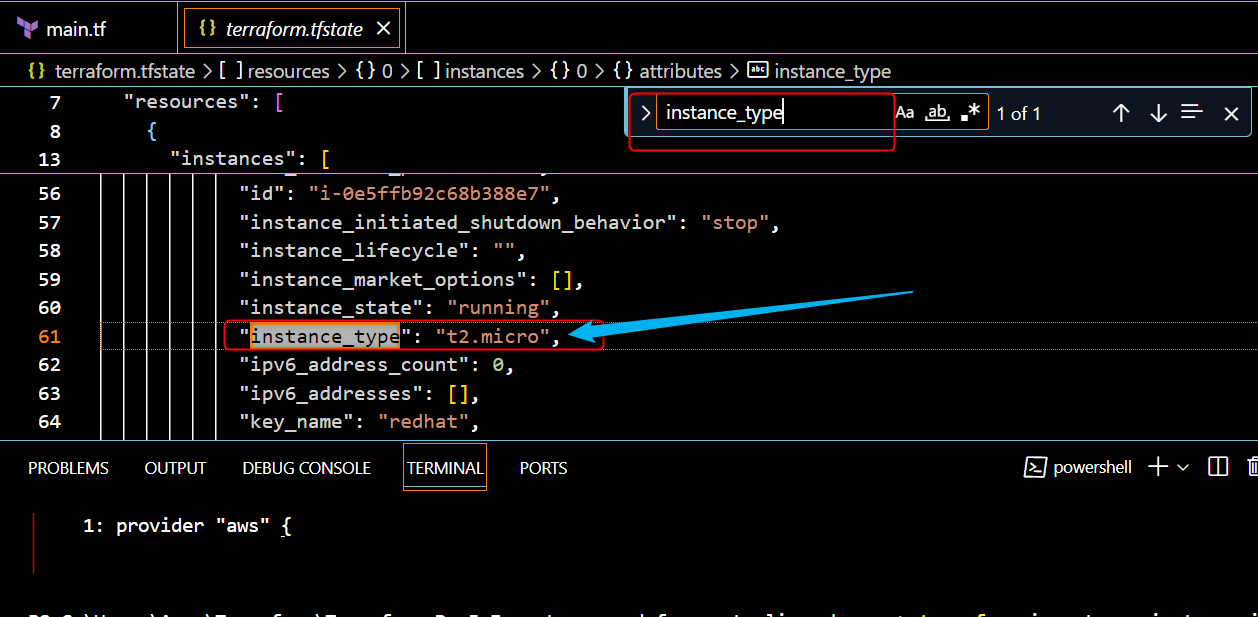
terraform import aws\_instance.importec2 i-0e5ffb92c68b388e7

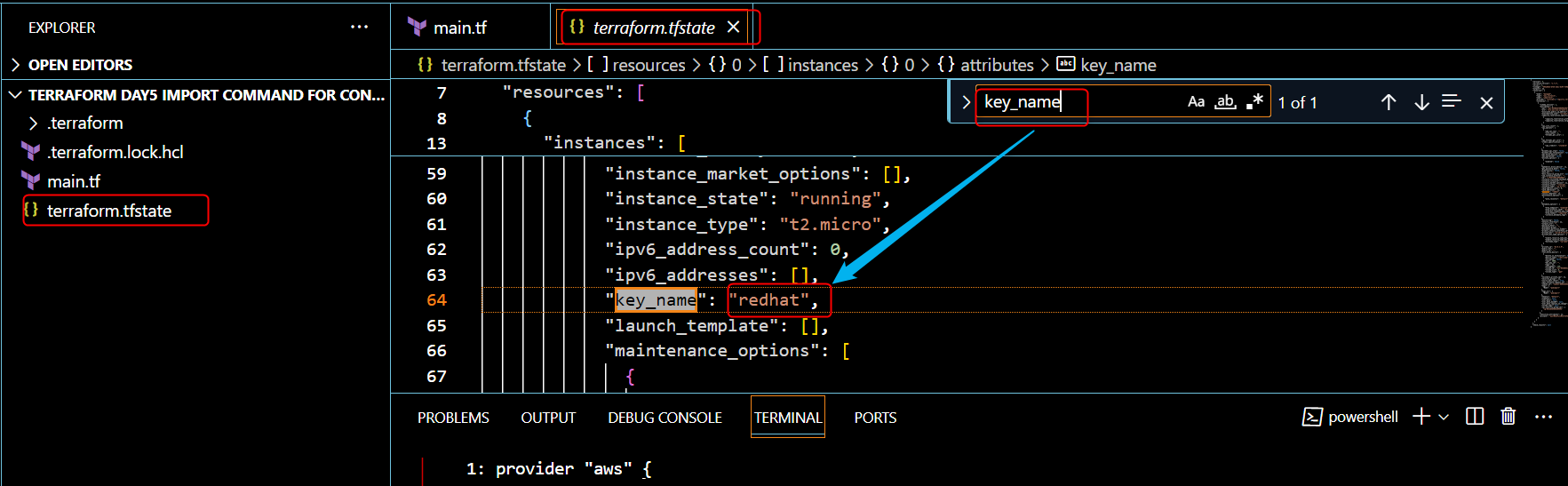




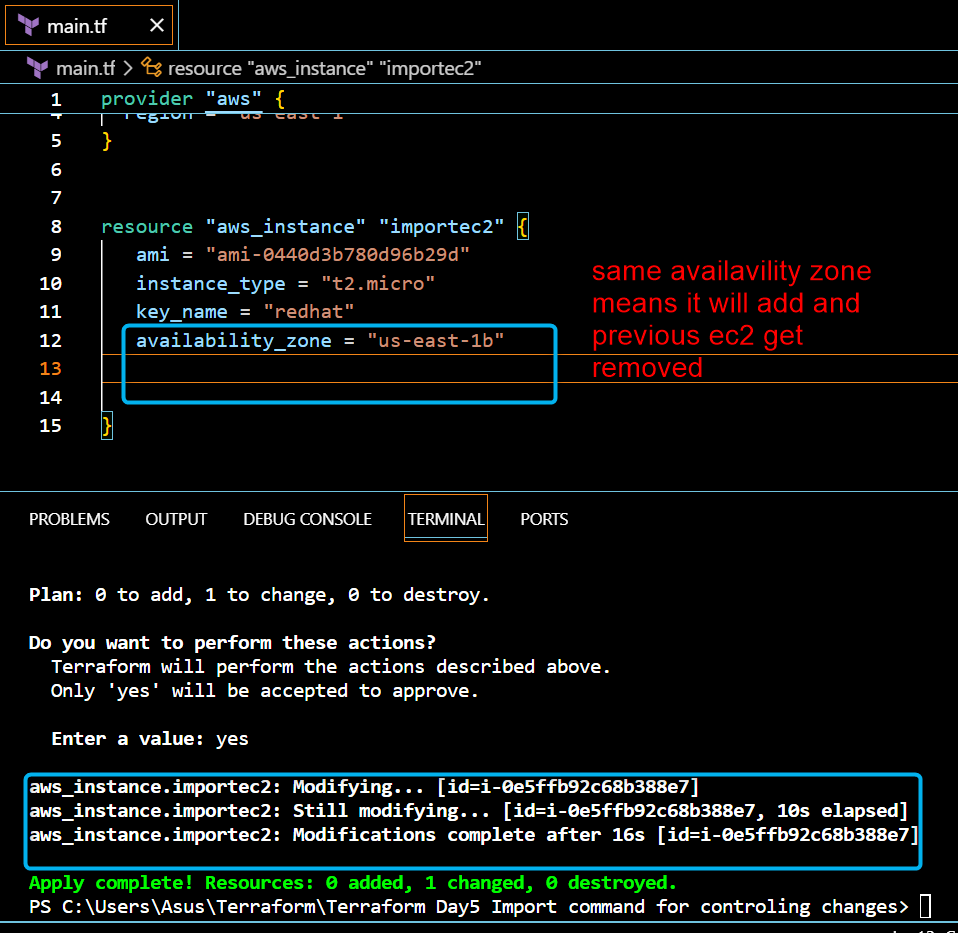
Now we can give all ami instance\_type key\_name by the refrence of statefile because state file recorded capture all details of that ec2 while importing to our local

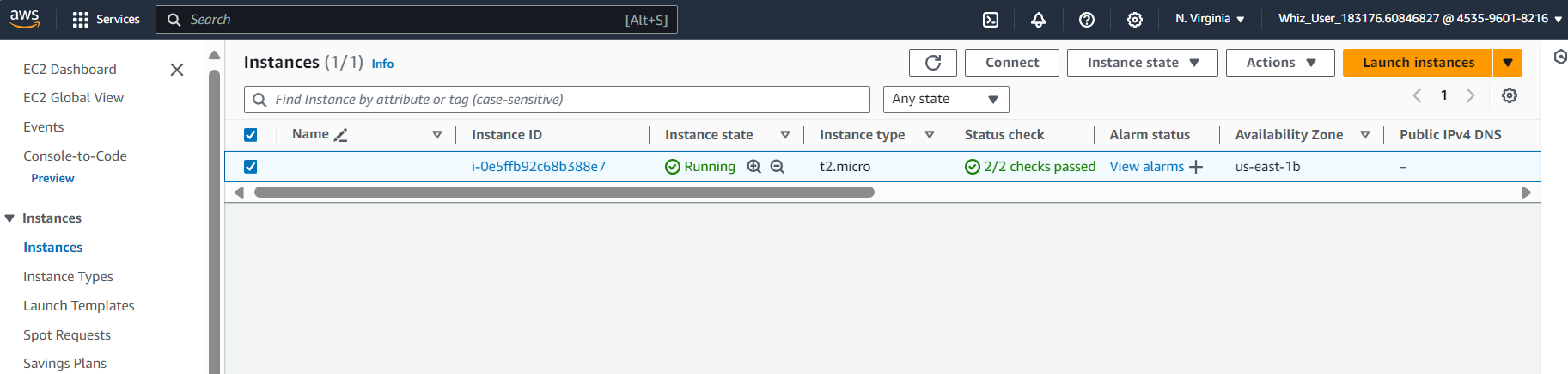






We can refer the details from statefile and code on our main resource block

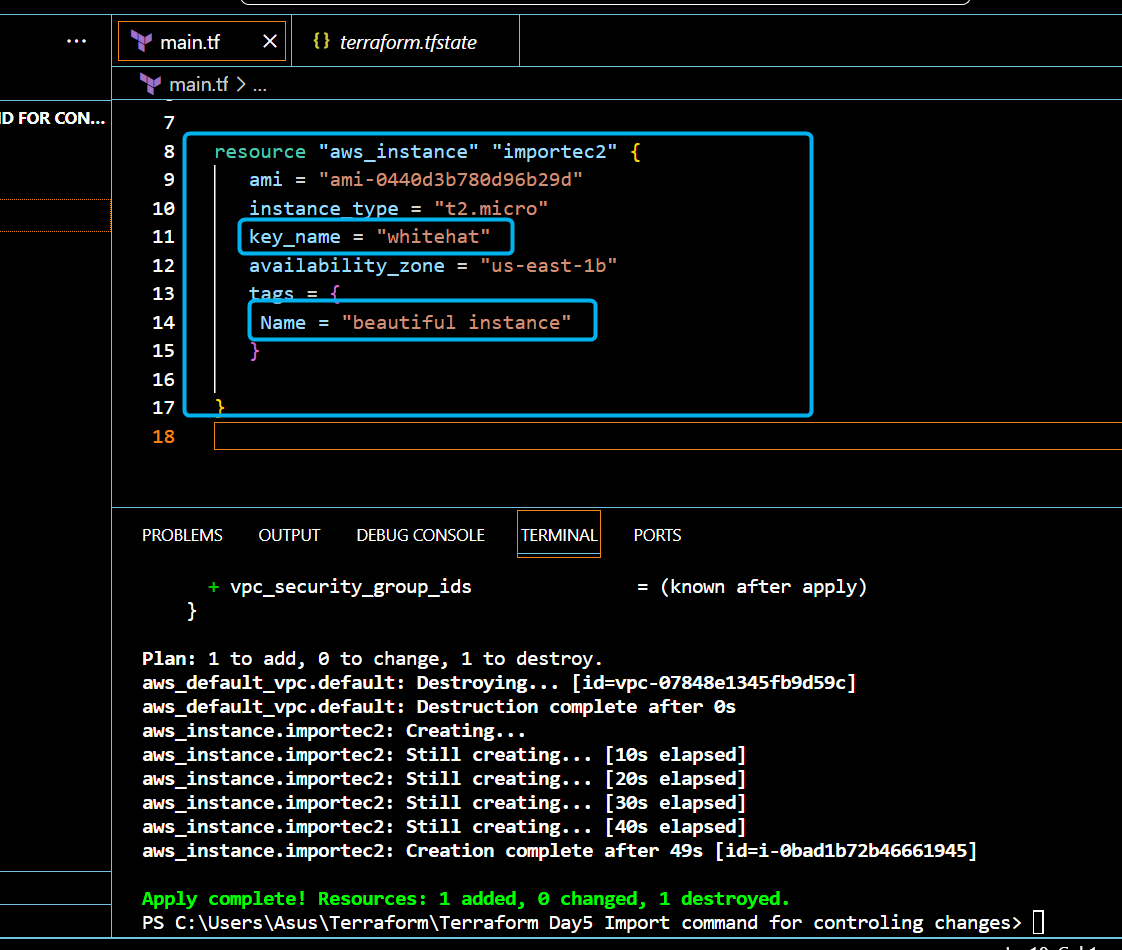


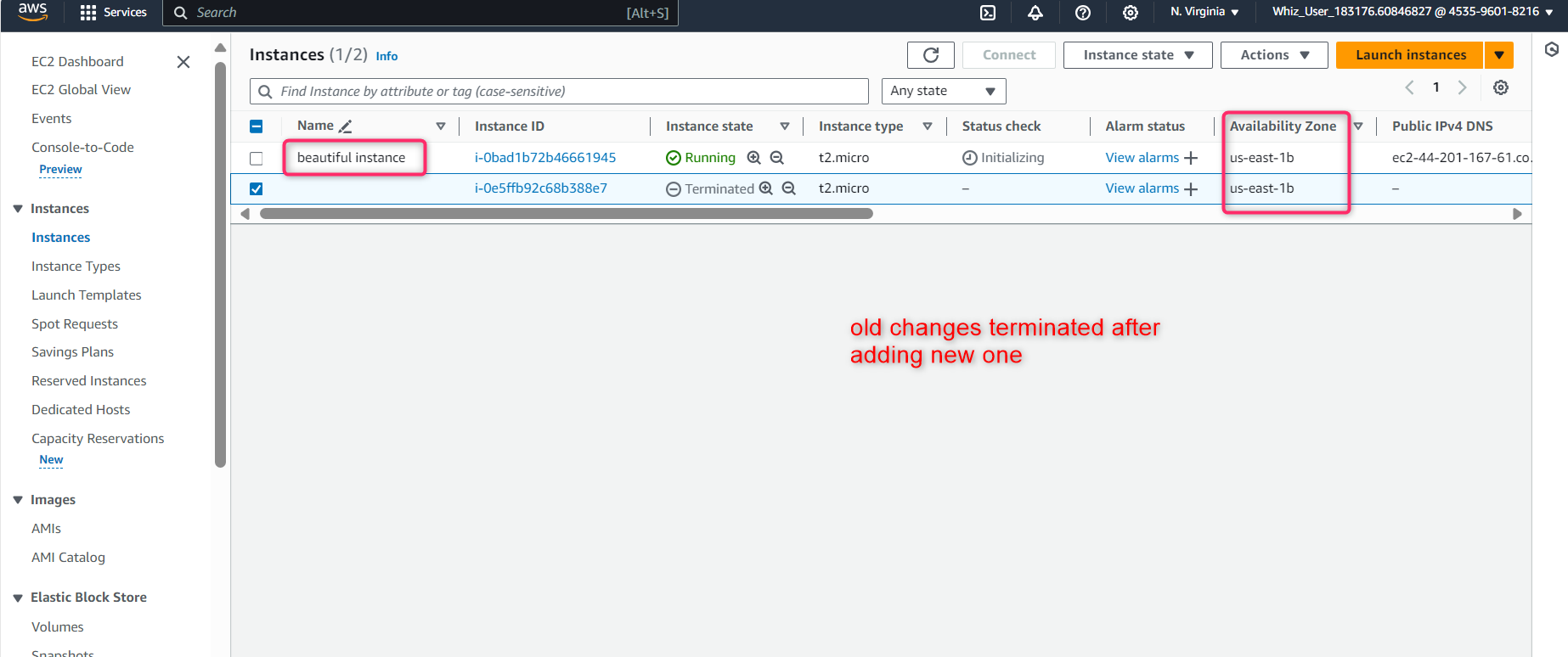


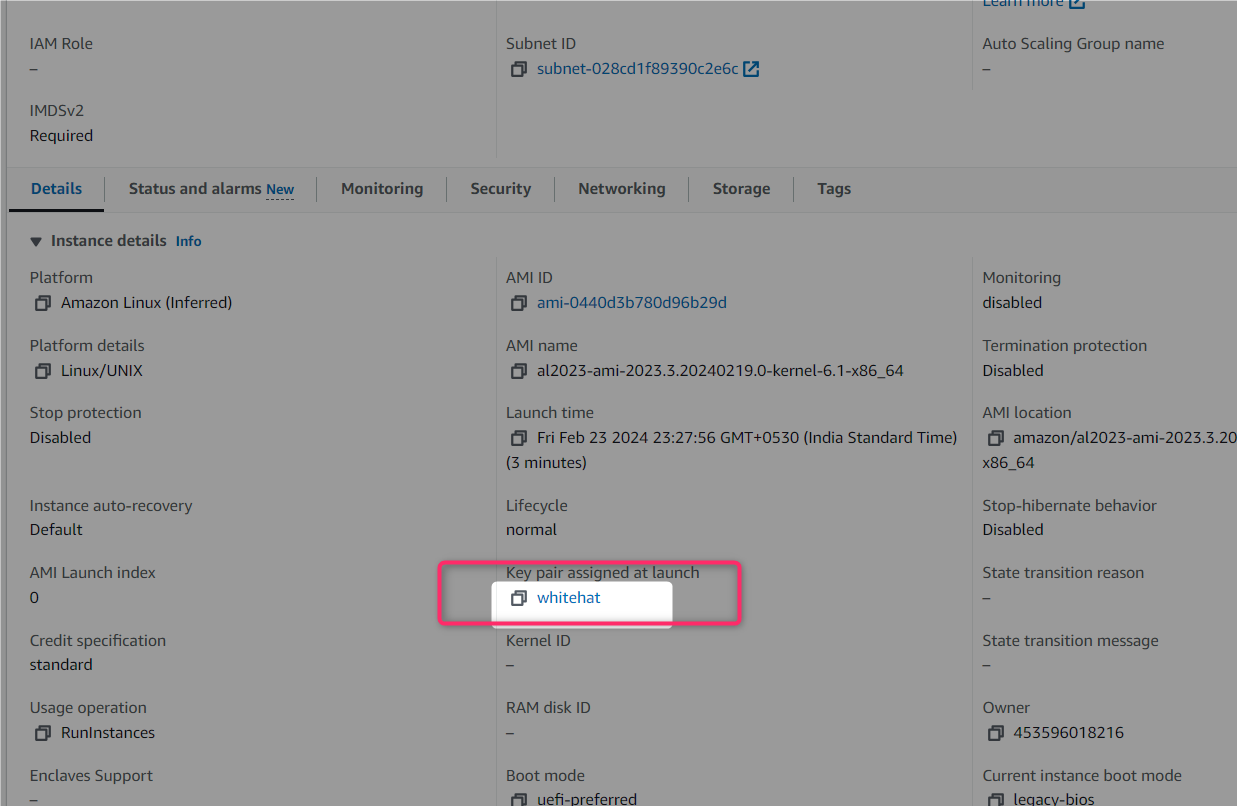
Now suppose I want to make further changes on it I will give another key pair previous at first before import it was redhat in statefile means first ec2 before import have redhat key\_name

As Now I have taken full control let me modify as per my wants

Let me change key name to “Whitehat(new key\_name)” from redhat (old key\_name) and also let **me tag a name** “beautiful instance” we can rule it because we own it now by taking control of it through import command.







DAY 6

* DATA SOURCE

Here we can use custom network where we already have vpc created and inside vpc  
my subnet , my internet g/w RT everything configured and that all vpc configuration attached to our placed ec2 public instance inside public subnet.

But here we can create any instance at any time and we can call same cust netwoek configuration where we already have our vpc details and all.

This can be done with help of data source.

So we already have the custom network configured we can copy the whole configuration and paste to our new folder.

#create vpc

resource "aws\_vpc" "custnw" {

    cidr\_block =  "10.0.0.0/16"

    tags = {

      Name = "ankit\_vpc"

    }

}

#create Internet Gateway and attach to VPC

resource "aws\_internet\_gateway" "custnw" {

vpc\_id = aws\_vpc.custnw.id

tags = {

  Name = "Ankit Internet Gateway"

}

}

#create subnet  attach to vpc

resource "aws\_subnet" "custnw" {

vpc\_id = aws\_vpc.custnw.id

cidr\_block = "10.0.0.0/24"

tags = {

  Name = "Ankit subnet"

}

}

#create RT and attach to vpc

resource "aws\_route\_table" "custnw" {

vpc\_id = aws\_vpc.custnw.id

tags = {

  Name = "Ankit Rt"

}

  #associate route table with internetgateway

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.custnw.id

}

}

#associate route table with subnet

resource "aws\_route\_table\_association" "custnw" {

route\_table\_id = aws\_route\_table.custnw.id

subnet\_id = aws\_subnet.custnw.id

}

#cust security group

resource "aws\_security\_group" "custnw\_sg" {

    name = "custnw\_sg"

    description = "Allow TLS inbound traffics"

    vpc\_id = aws\_vpc.custnw.id

    ingress {

    description = "TLS from VPC"

    from\_port        = 80

    to\_port          = 80

    protocol         = "tcp"

    cidr\_blocks      = ["0.0.0.0/0"]

  }

  ingress {

    description = "TLS from VPC"

    from\_port        = 22

    to\_port          = 22

    protocol         = "tcp"

    cidr\_blocks      = ["0.0.0.0/0"]

  }

  ingress {

    description = "TLS from VPC"

    from\_port        = 443

    to\_port          = 443

    protocol         = "tcp"

    cidr\_blocks      = ["0.0.0.0/0"]

  }

    egress {

    from\_port        = 0

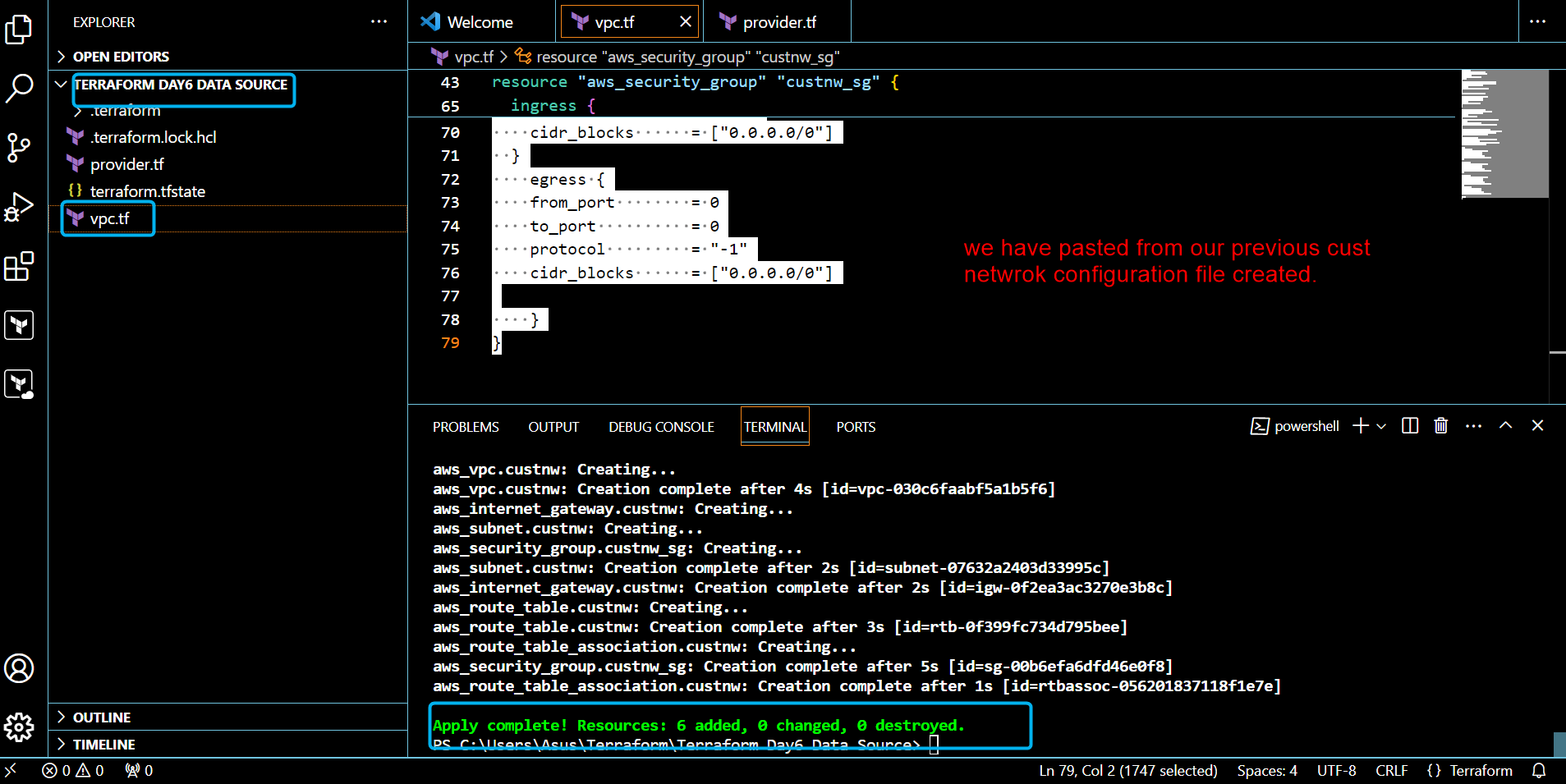
    to\_port          = 0

    protocol         = "-1"

    cidr\_blocks      = ["0.0.0.0/0"]

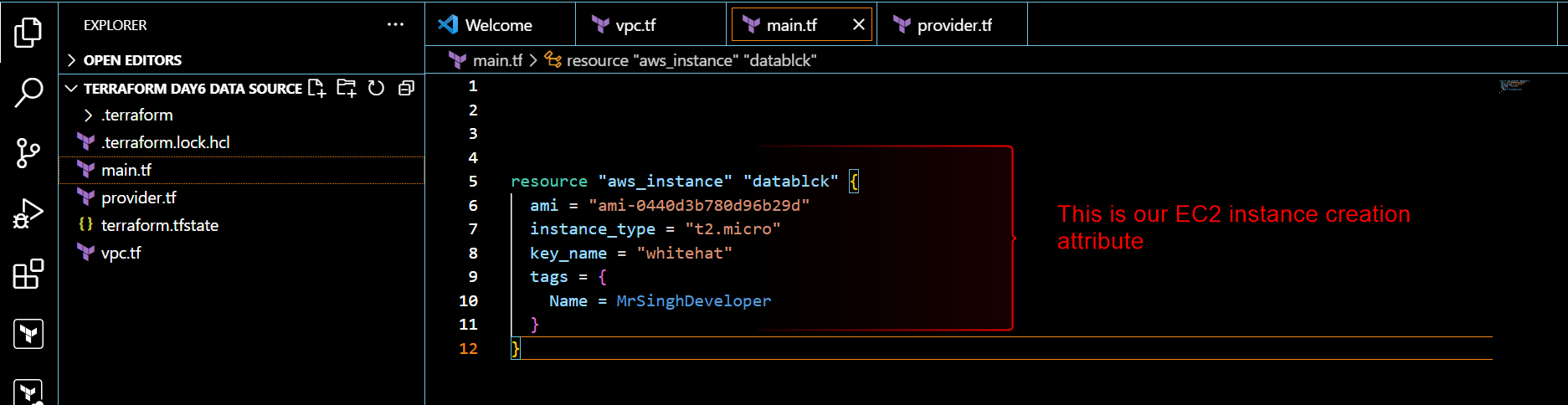
    }

}



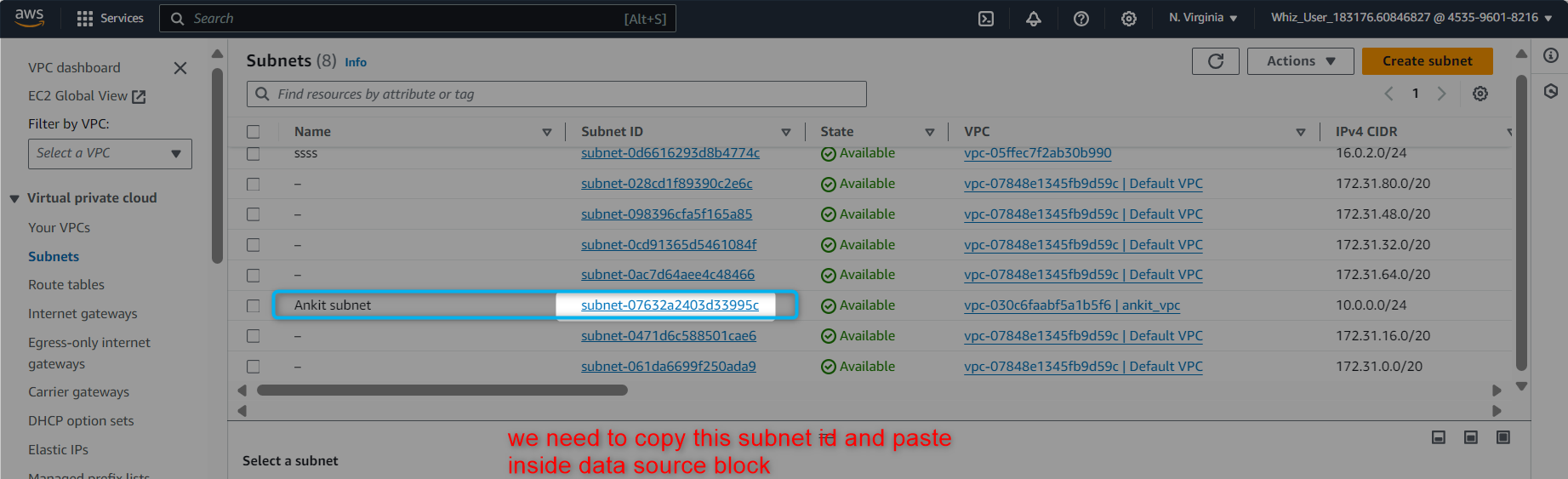
Now we can create fresh ec2 instance and inside it only we call vpc which we already taken from old and pasted to our new dir.

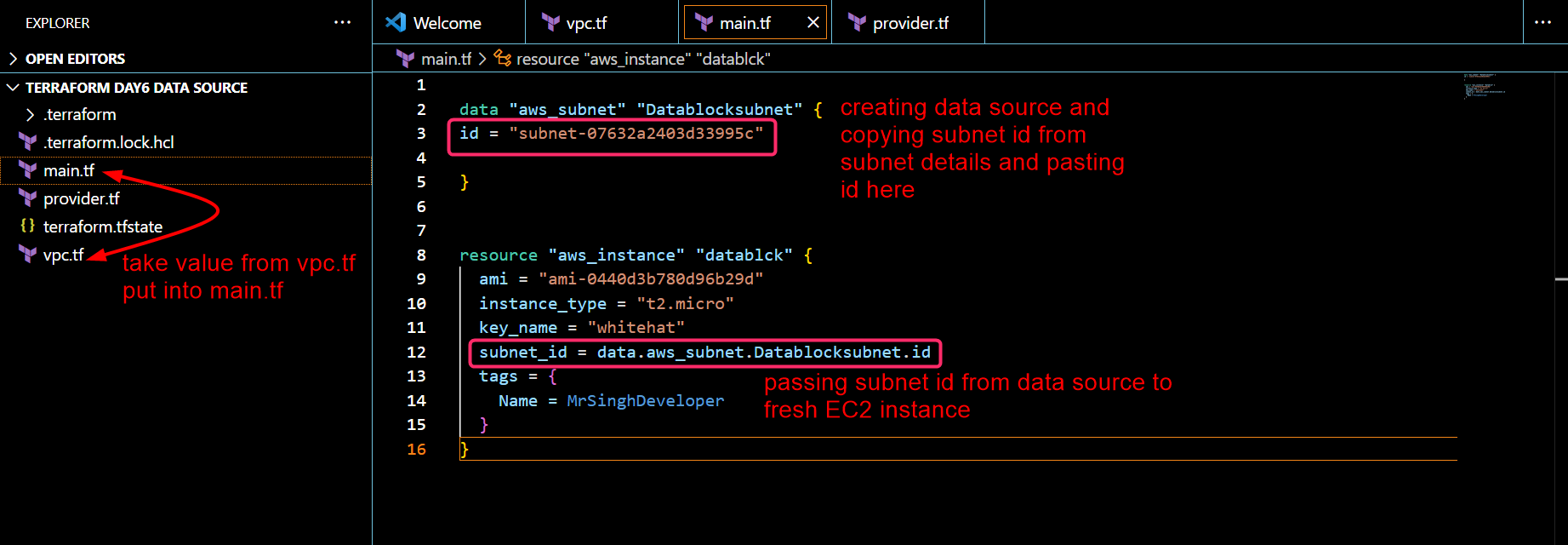
Lets create a fresh ec2 instance.



Go to vpc

Subnet ID (from here copying subnet id and pasting inside data source block)





Like above we passed the value of subnet into fresh ec2 via creating data source.

Same we can pass Security Group as well

Go to vpc

Security Group ID (from here copying SG id and pasting inside data source block)

