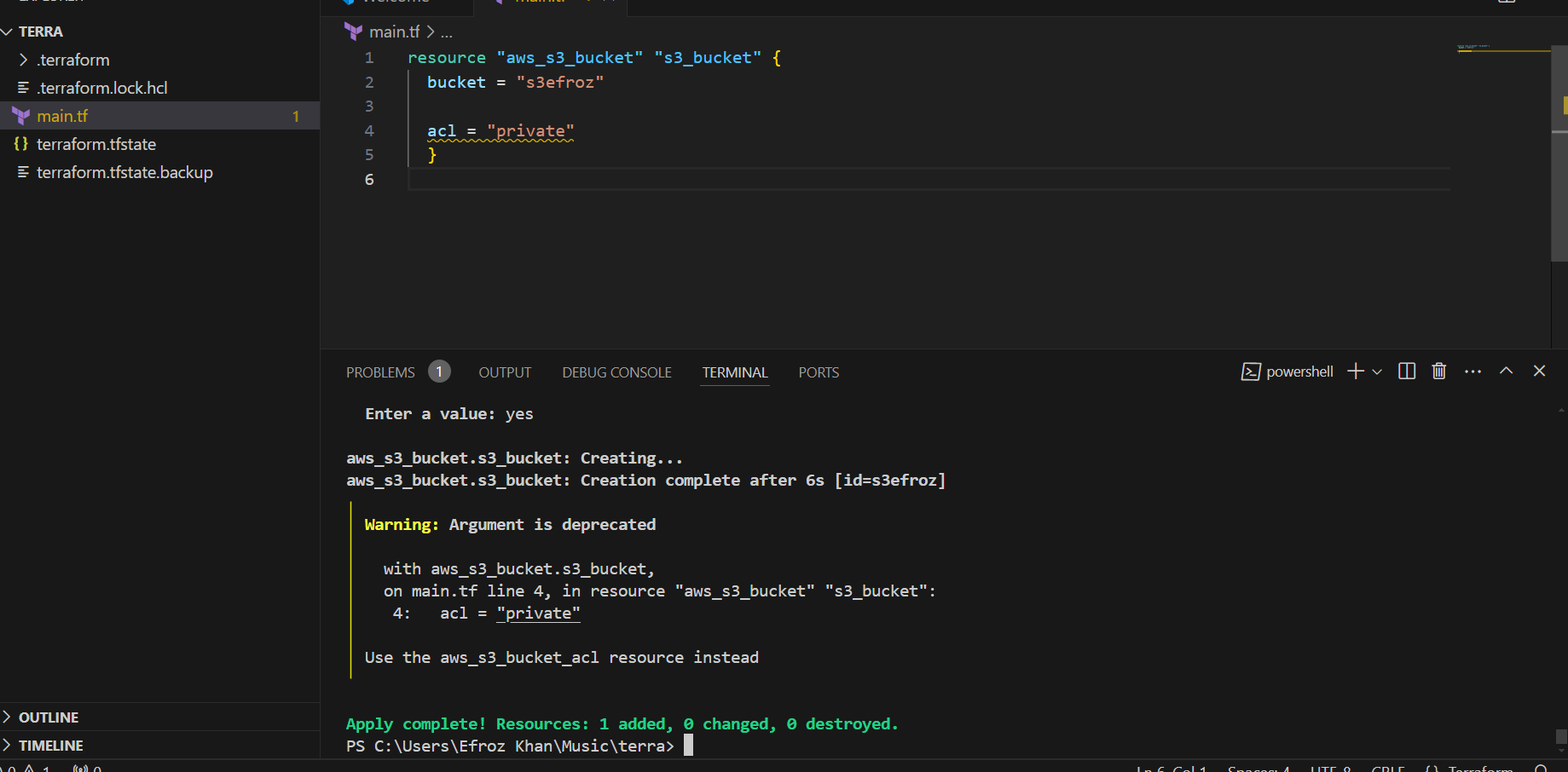
1. Watch terraform-05 video.

======= Watched ================

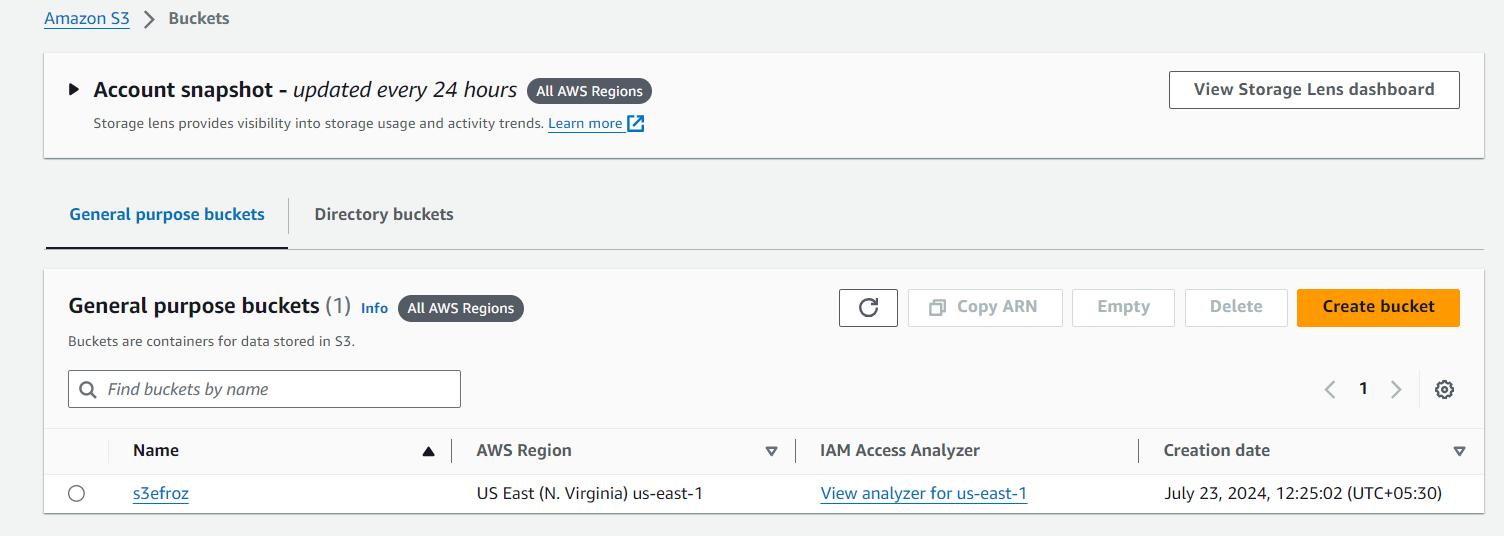
1. Execute the script shown in video.

Terraform apply –all-approve (this command is used to without asking yes permission to when terraform apply execute )

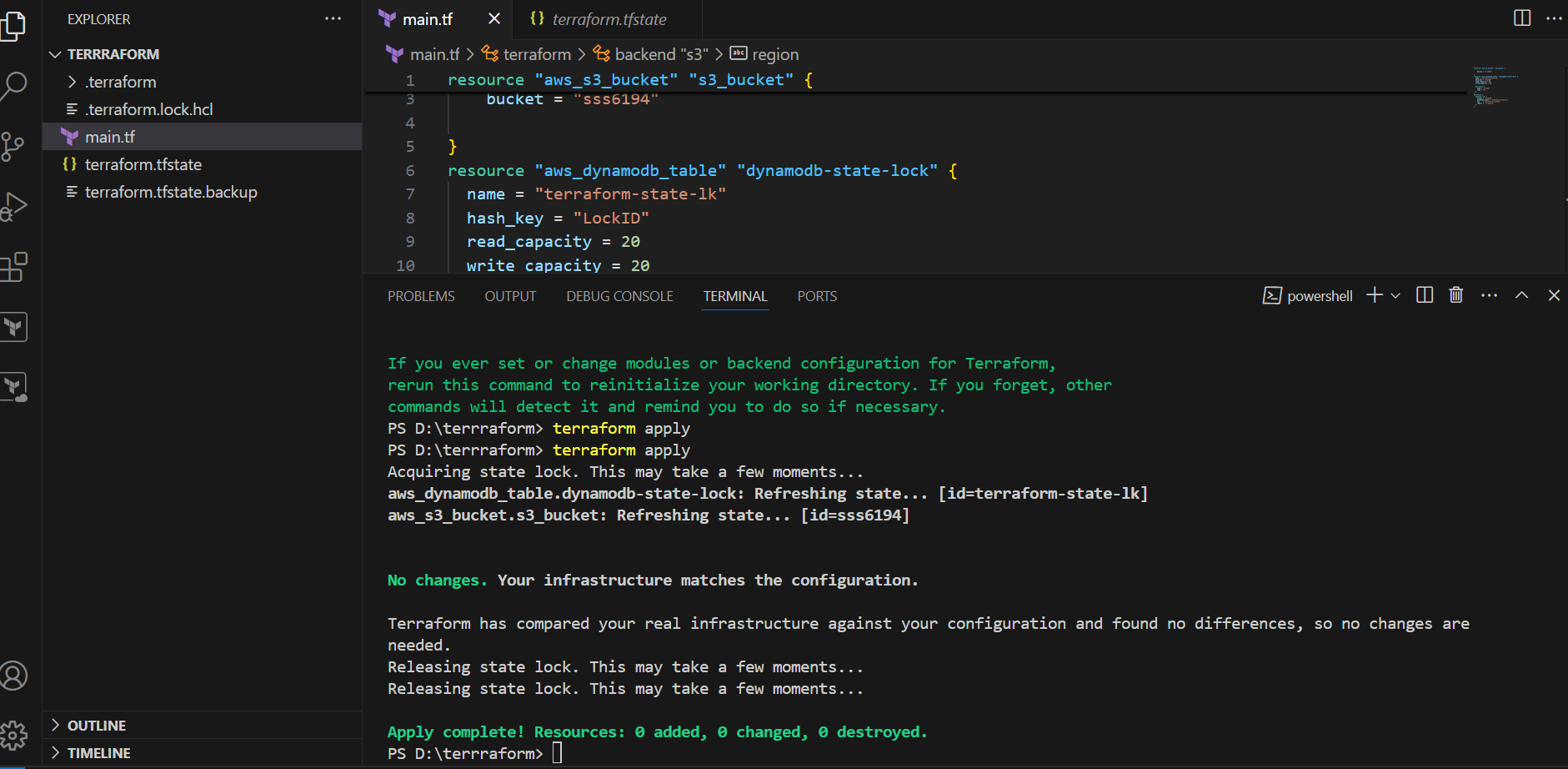
Creating s3 bucket using s3 resource



Out of s3 bucket created



Creating S3 bucket, dynamoDB and S3 as backend for terraform.tfstate file:-



resource "aws\_s3\_bucket" "s3\_bucket" {

    bucket = "sss6194"

}

resource "aws\_dynamodb\_table" "dynamodb-state-lock" {

  name = "terraform-state-lk"

  hash\_key = "LockID"

  read\_capacity = 20

  write\_capacity = 20

  attribute {

    name = "LockID"

    type = "S"

  }

}

terraform {

  backend "s3" {

    bucket = "sss6194"

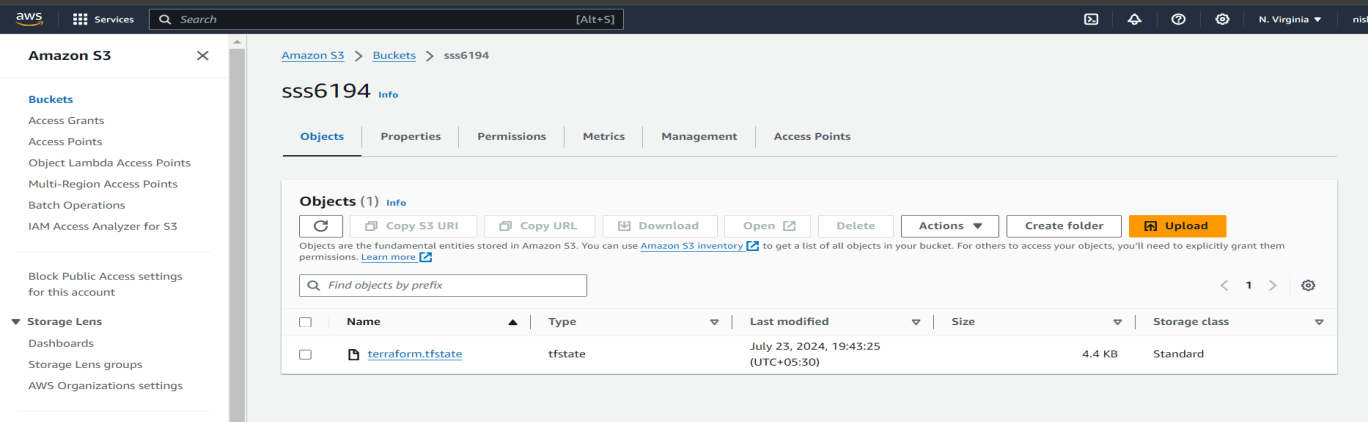
    dynamodb\_table = "terraform-state-lk"

    key    = "terraform.tfstate"

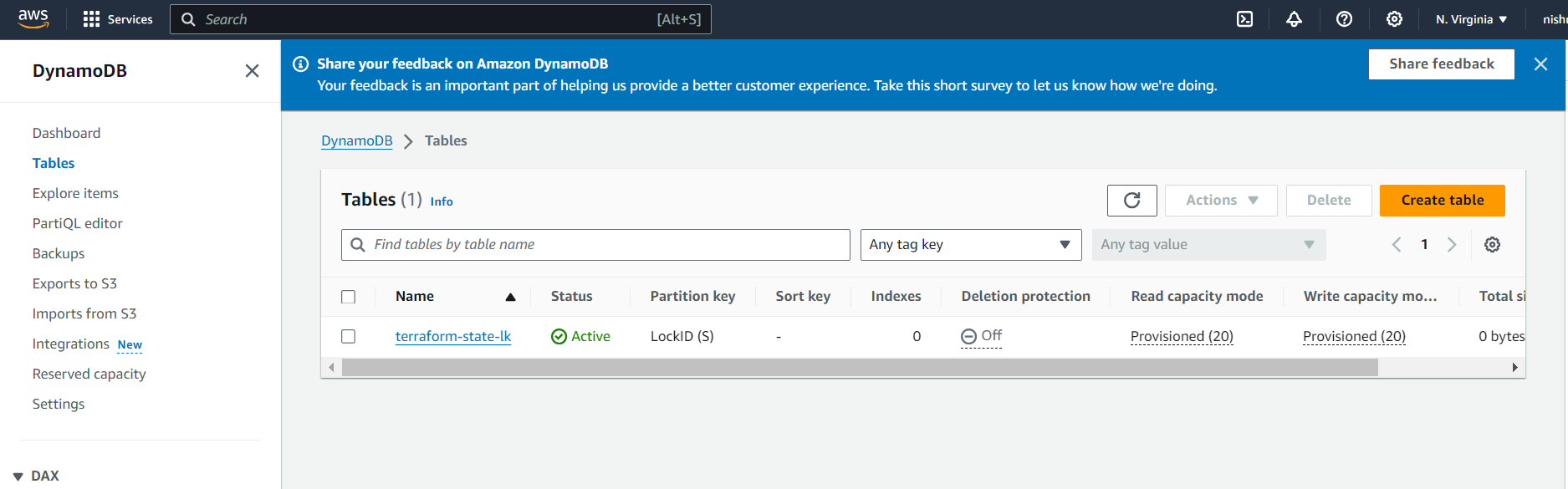
    region = "us-east-1"

  }

}

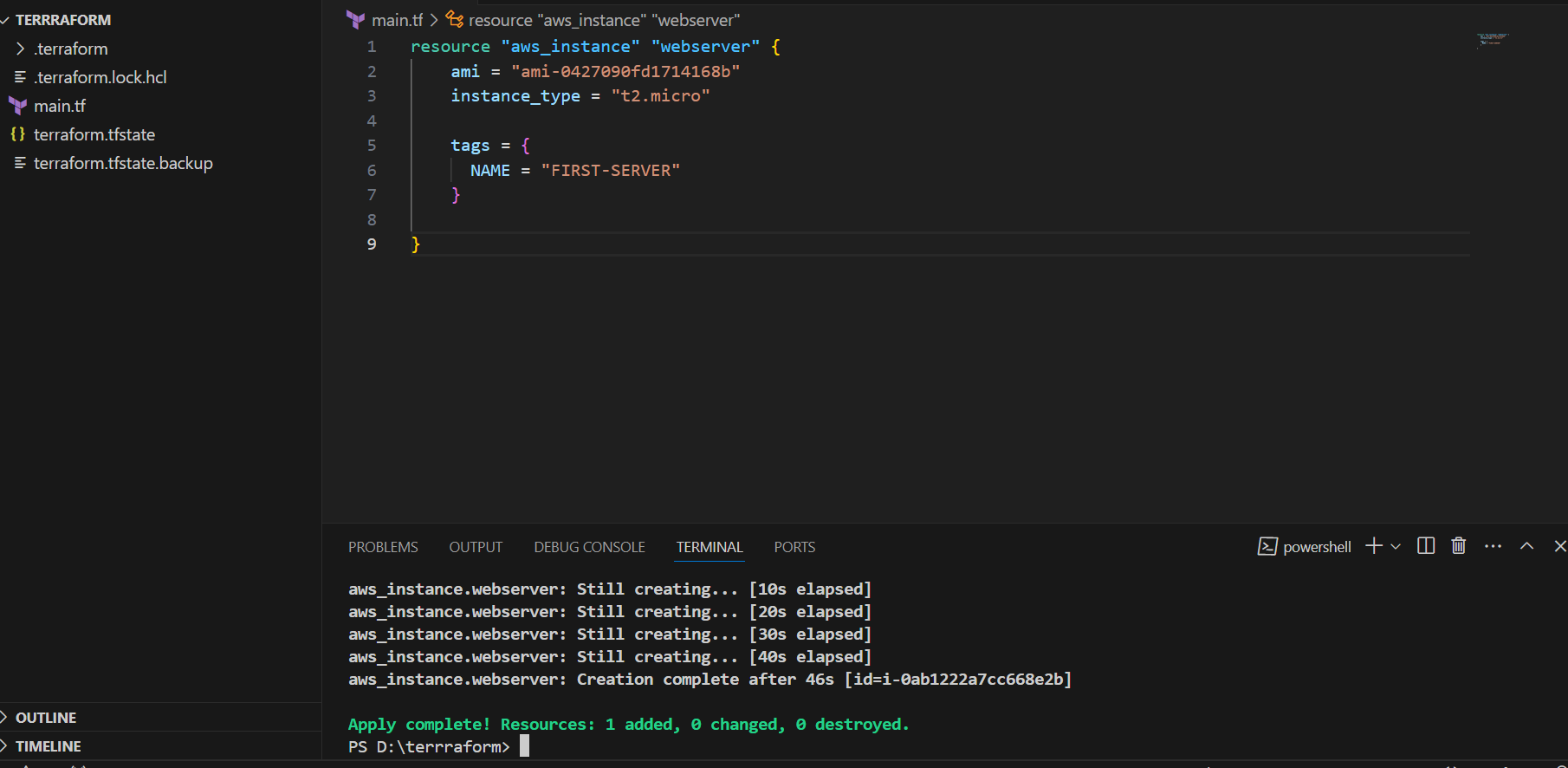
S3 output

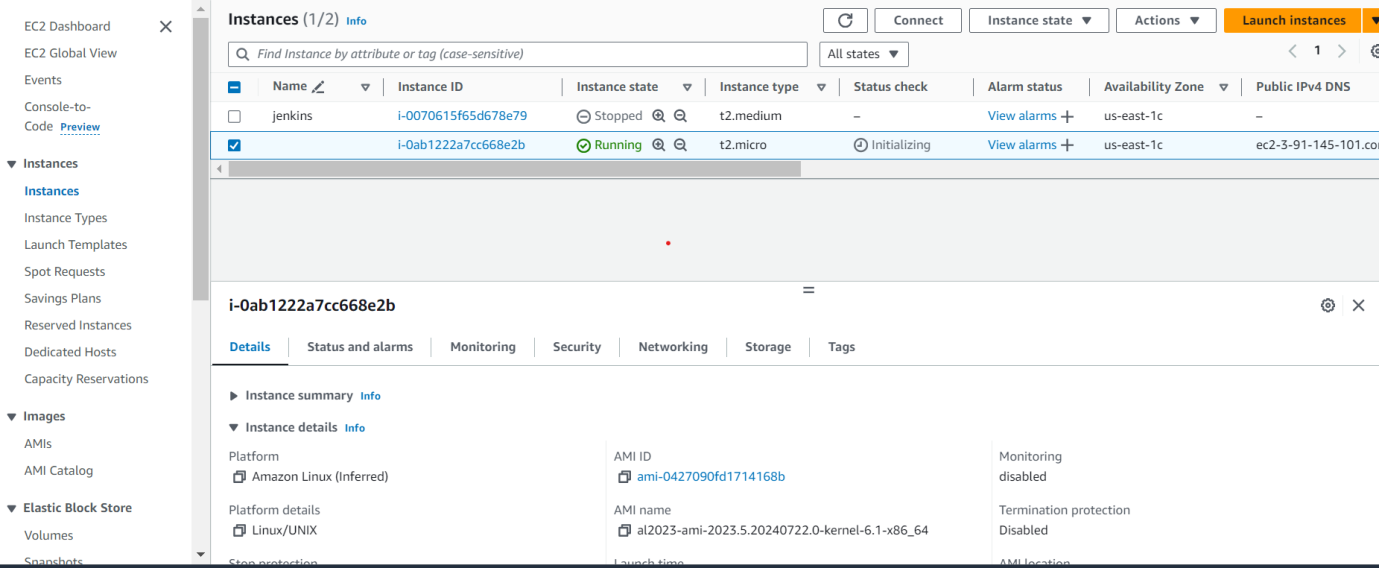
DynamoDB output



**Terraform** **Provisioner**:-

Creating Ec2 instance using resource





Remote Provisioner Example:

Example of Local Provisioner:-

provider "aws" {

  region = "us-east-1"

}

resource "aws\_instance" "test-server" {

  ami           = "ami-0427090fd1714168b"

  instance\_type = "t2.micro"

  provisioner "local-exec" {

    command = "echo ${self.public\_ip} Created! > public\_ips.txt"

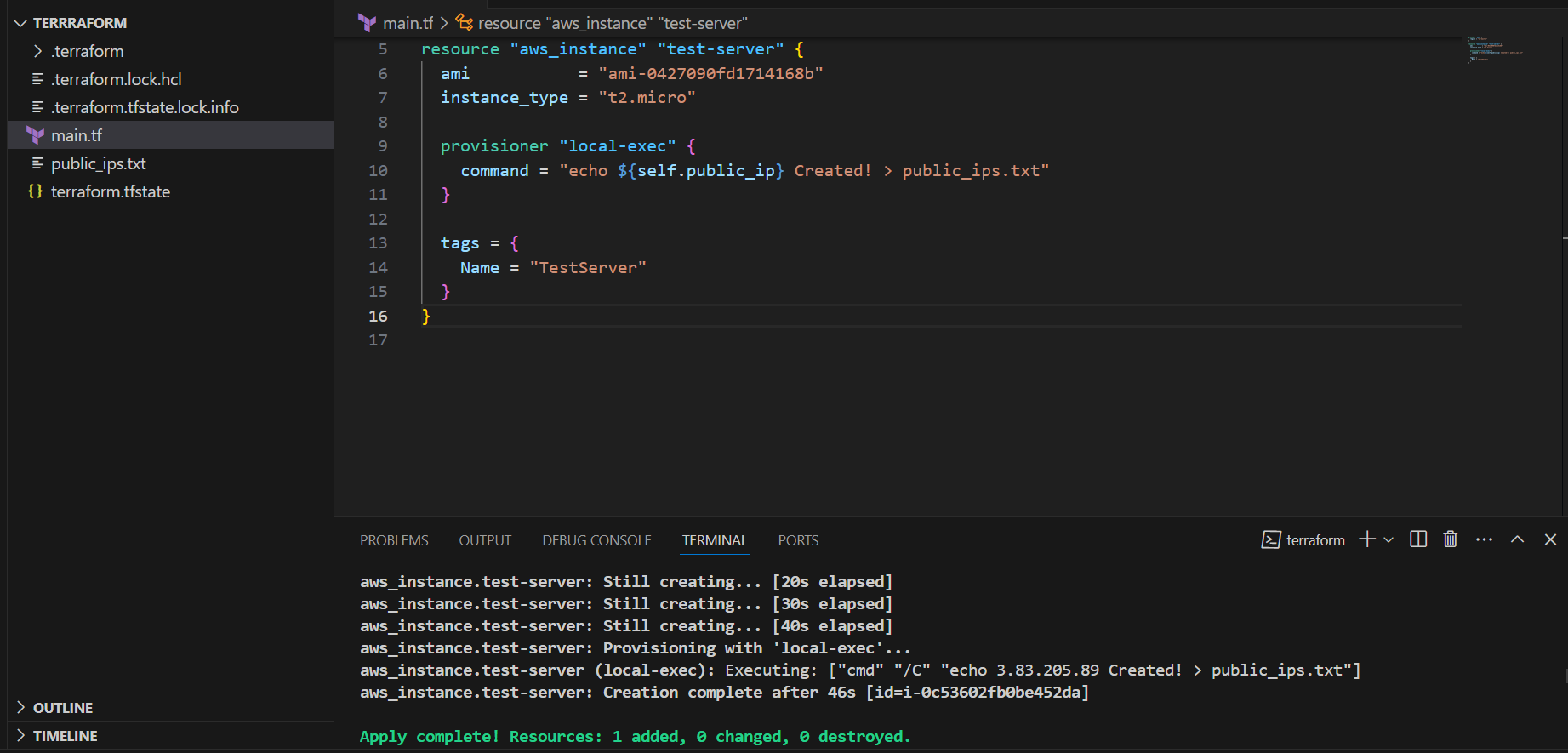
  }

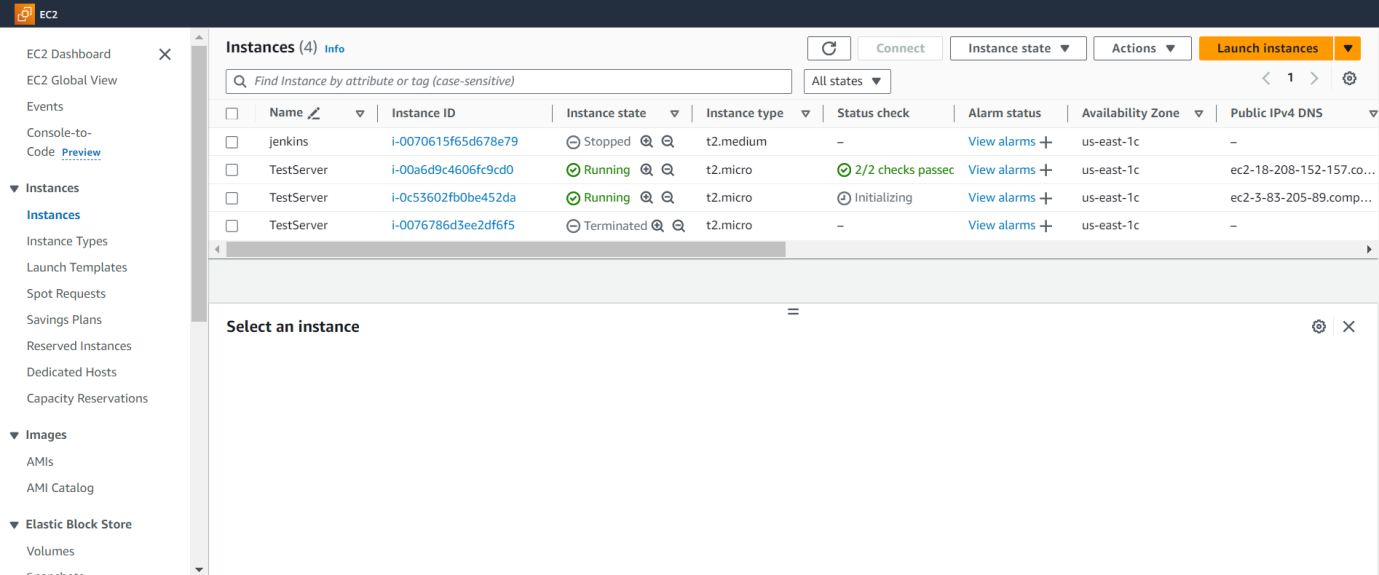
  tags = {

    Name = "TestServer"

  }

}





We can use local provider after creating/destroying respource.

**Terraform provisioner behaviours**:

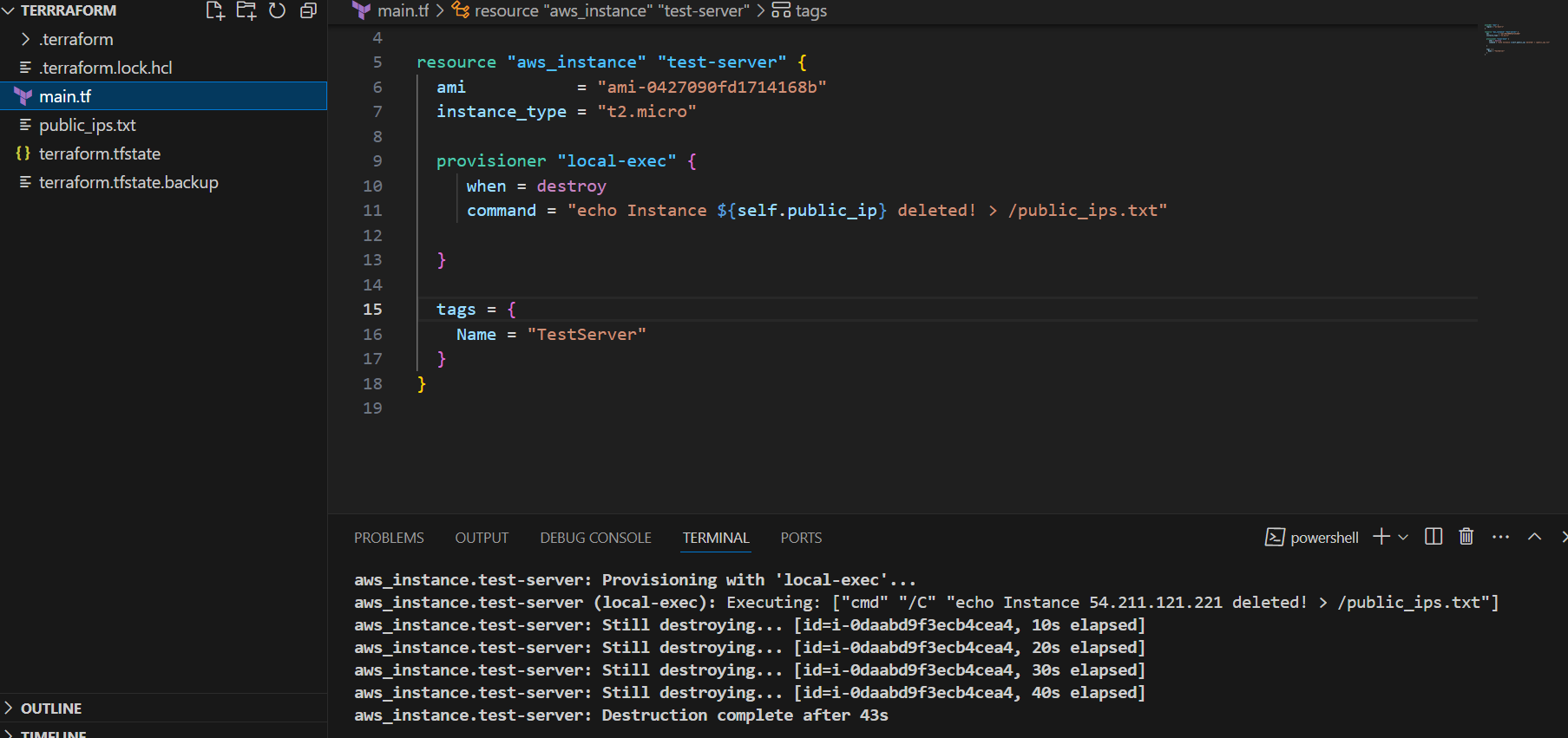
1) Create at the time creating resource (Default)

2) create at the time of destroy in resource (when = destroy)

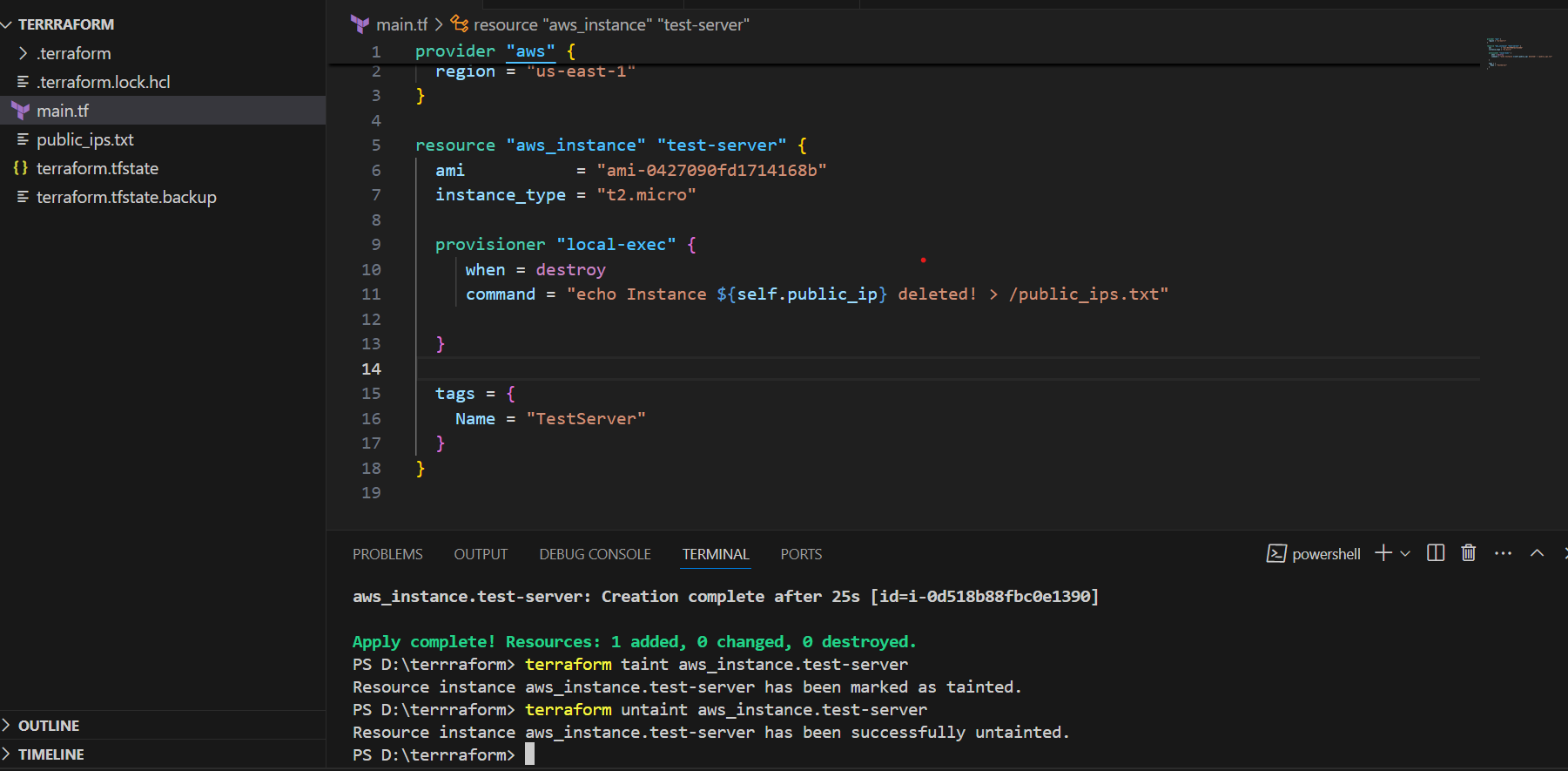
3) on\_failure = fail --> to create the resource if the script gets failed.(But terraform will mark the reource as tainted)

4) On\_fail = continue --> to create the resource and ignore the changes.

After destroy we need to add when condition in provisioner.



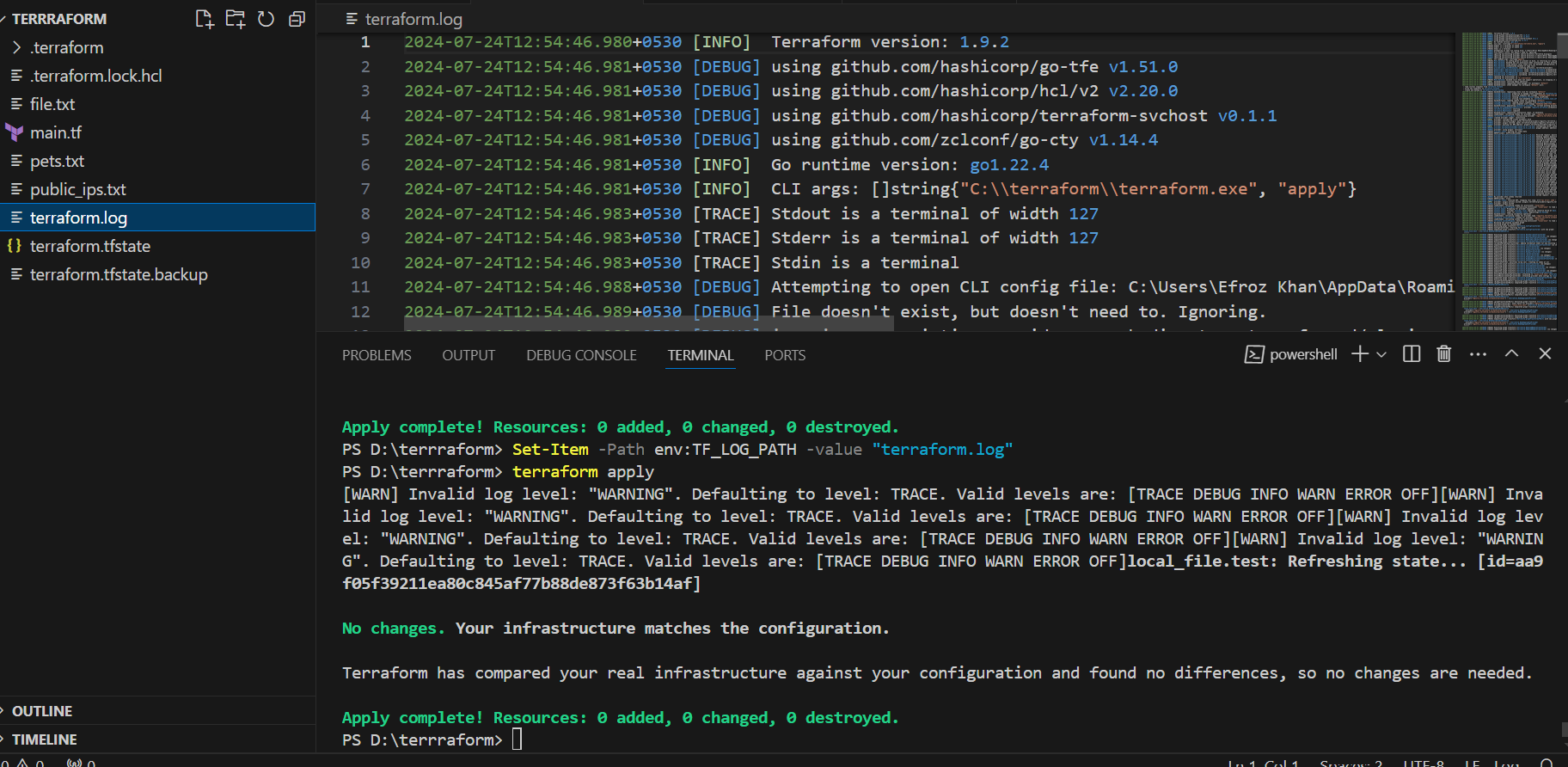
**Terraform taint and untaint:-**



**Debugging:-**

Terraform provides 5 levels of logs:

* INFO
* WARNING
* ERROR
* DEBUG
* TRACE



Here we storing logs in terraform.log floder by using below commands

Set-Item -Path env:TF\_LOG\_PATH -value "terraform.log"

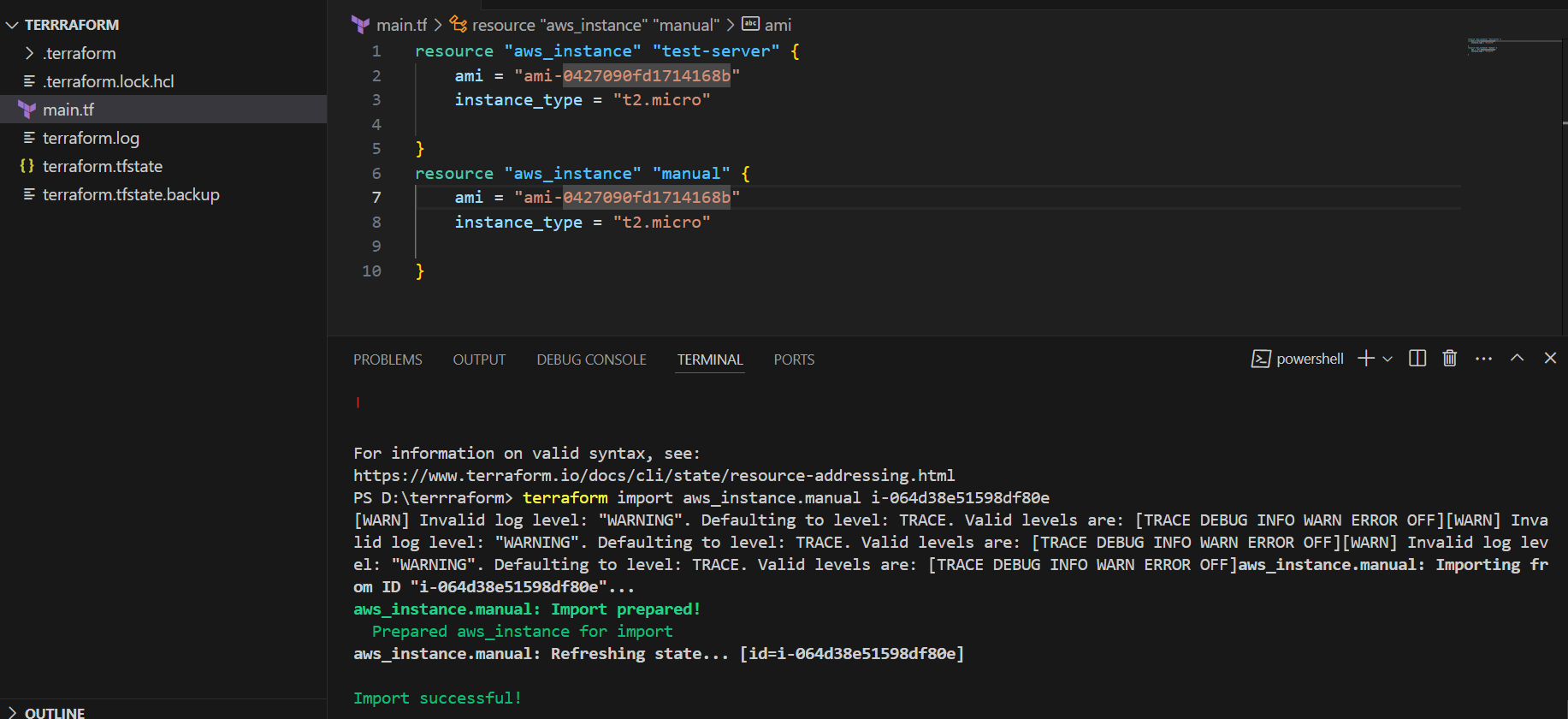
**Terraform import:-**

Terraform import is used to import the existing infrastructure in terraform state file.

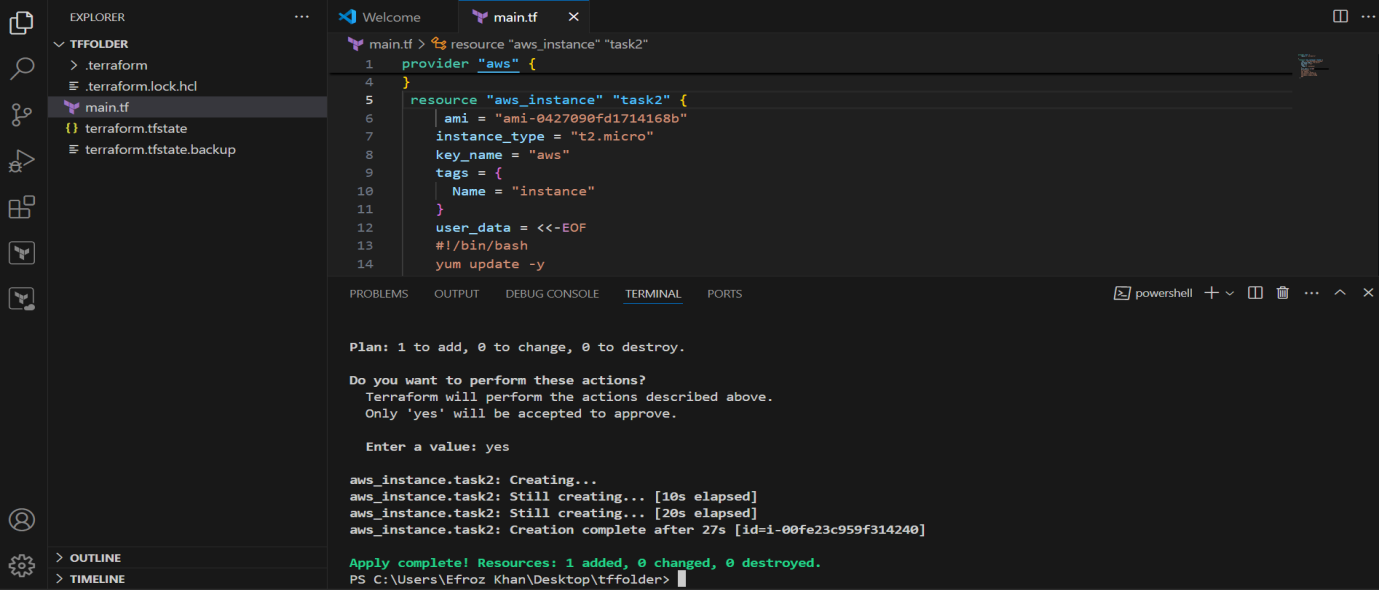
Once import is done then we can be able to create/delete and manage the infrastructure.

In order to import any resource we need to write the resource details in configuration file.

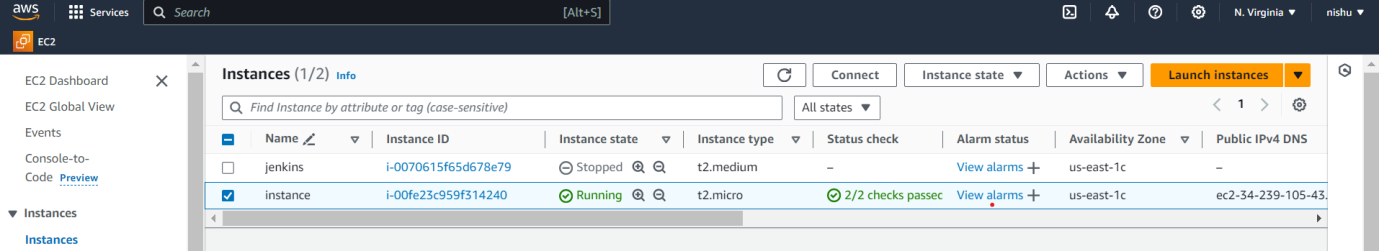
terraform import aws\_instance.<name> instance\_id



1. Create one ec2 instance with httpd installed using terraform script.
2. provider "aws" {
3. region = "us-east-1"
5. }
6. resource "aws\_instance" "task2" {
7. ami = "ami-0427090fd1714168b"
8. instance\_type = "t2.micro"
9. key\_name = "aws"
10. tags = {
11. Name = "instance"
12. }
13. user\_data = <<-EOF
14. #!/bin/bash
15. yum update -y
16. yum install httpd -y
17. systemctl start httpd
18. systemctl enable httpd
19. EOF
20. }

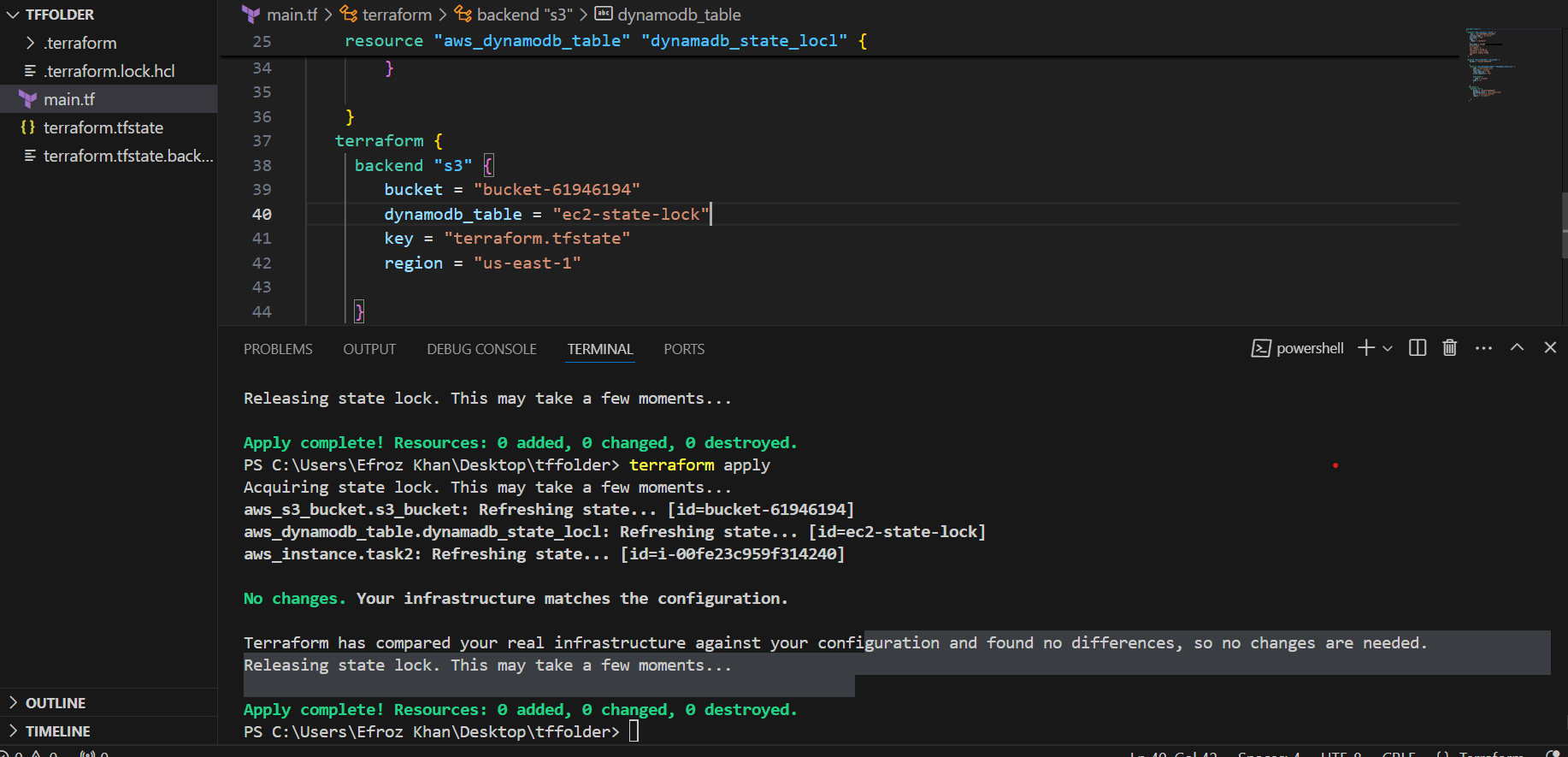


**Out put:-**

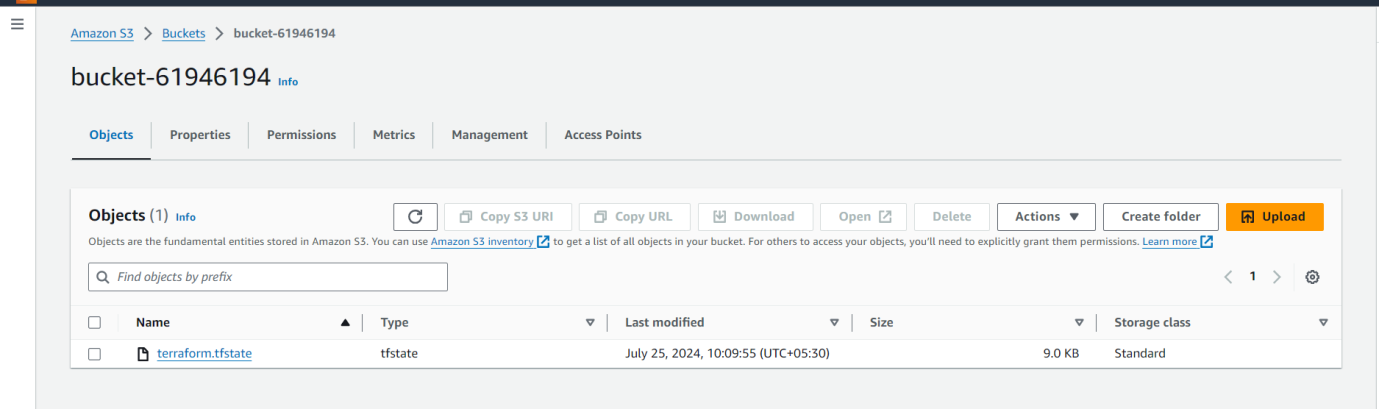




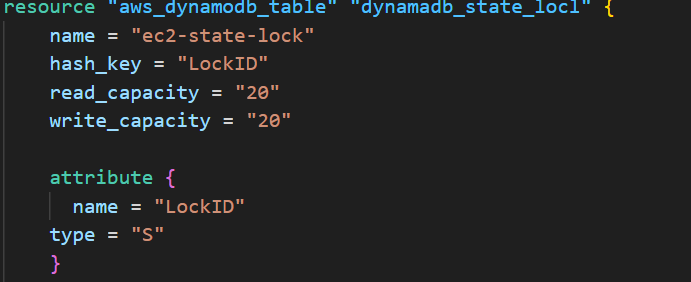
1. Setup s3 as backend to the task 3.
2. terraform {
3. backend "s3" {
4. bucket = "bucket-61946194"
5. dynamodb\_table = "ec2-state-lock"
6. key = "terraform.tfstate"
7. region = "us-east-1"
9. }
10. }

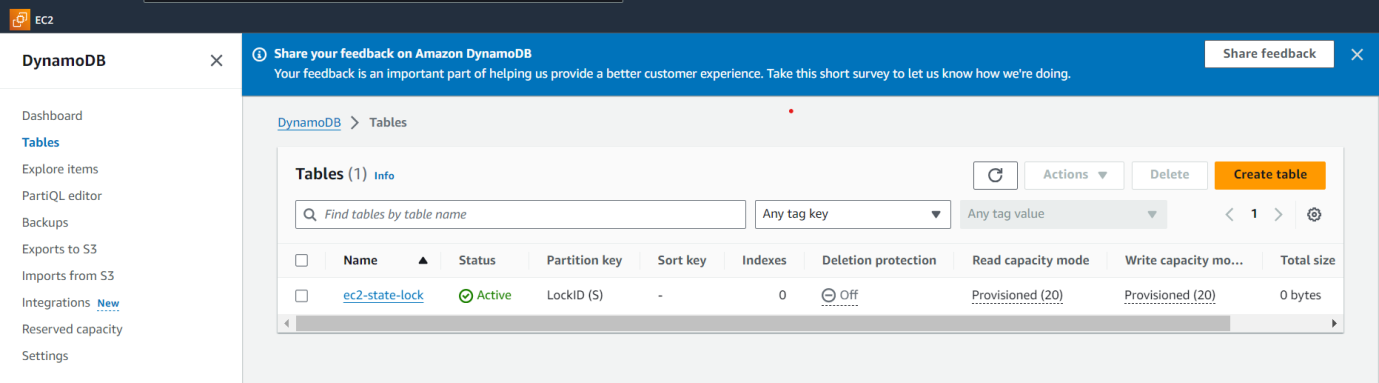


Out put:-



1. Setup dynamo db locking for task3.



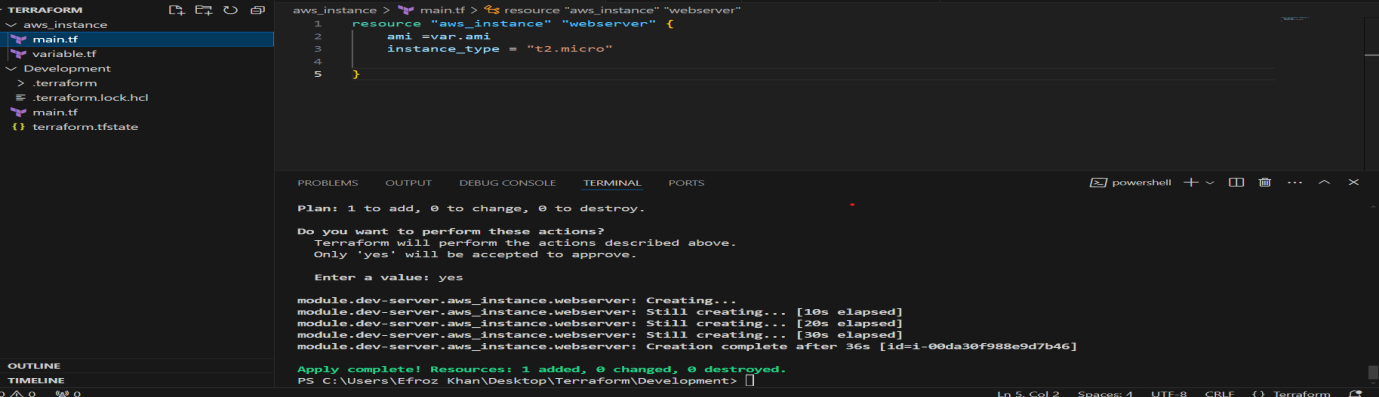


1. Watch terraform-06 video.

**Watched**

1. Execute the script shown in video.

Terraform module:-

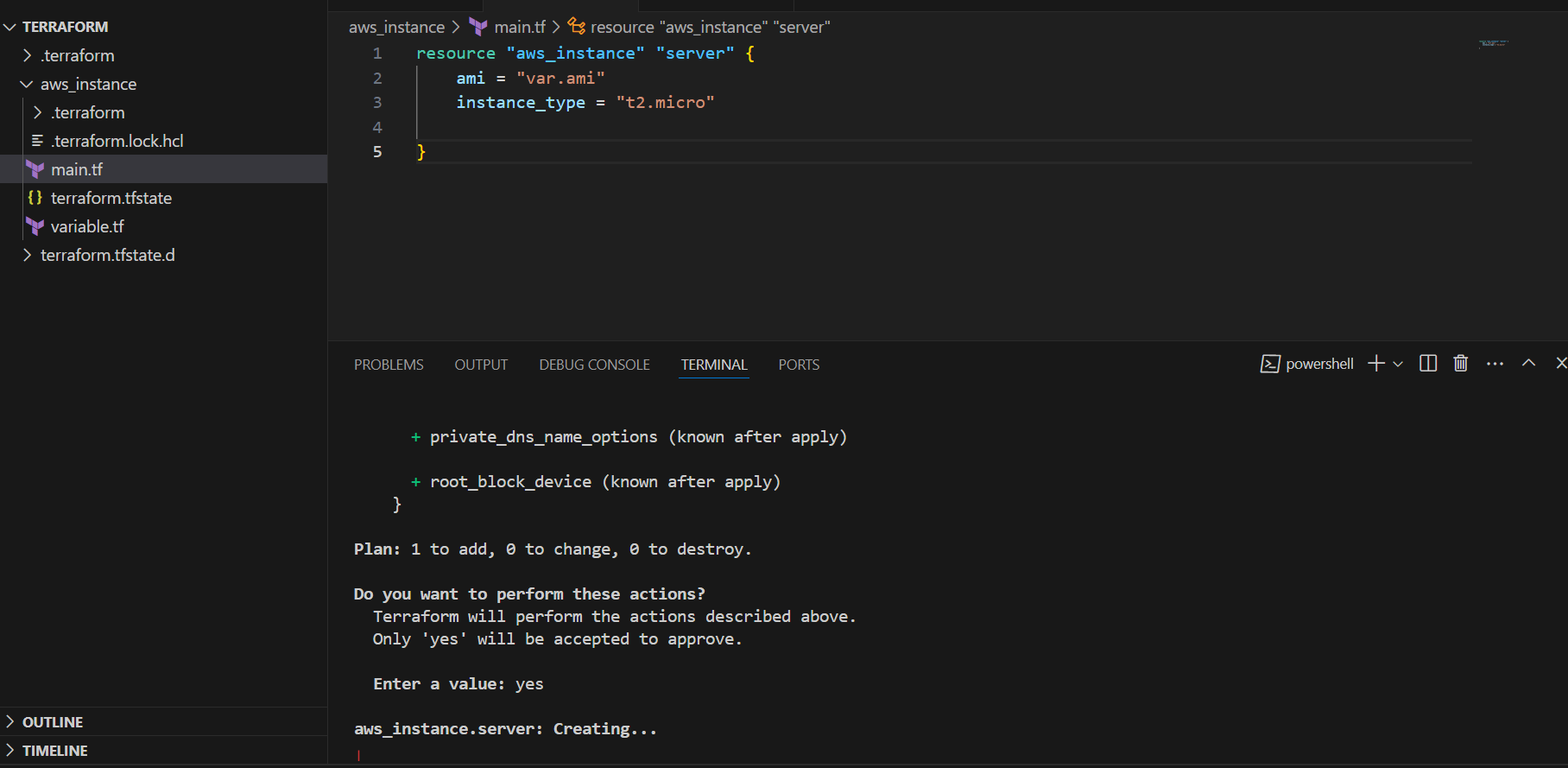


Terrafrom workspace:-

terraform workspace new projectA --> to create a workspace names projectA

terraform workspace list --> to list the workspaces available

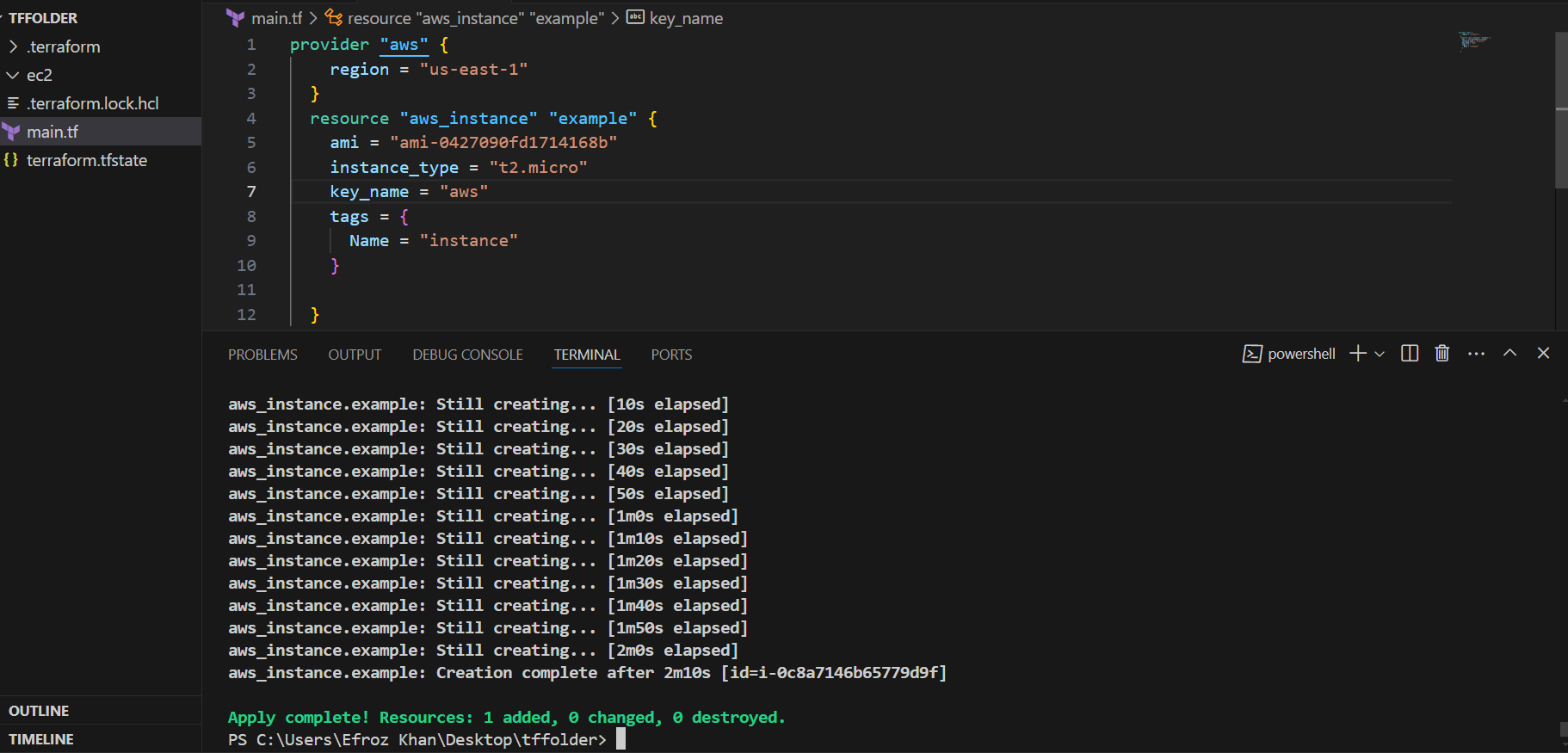
terraform workspace selecr projectA --> to swtich to specific workspace



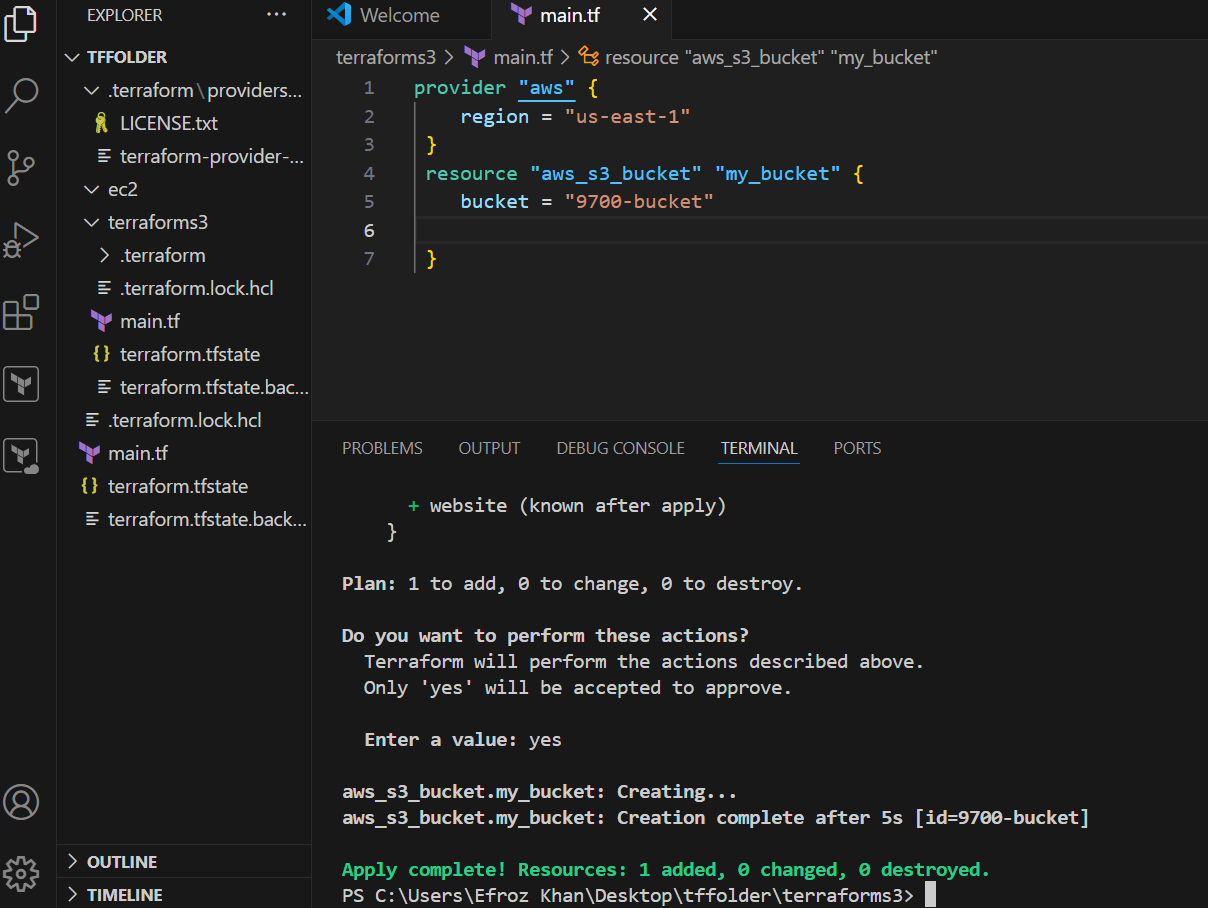
1. Provision ec2,s3 and vpc using Terraform modules.

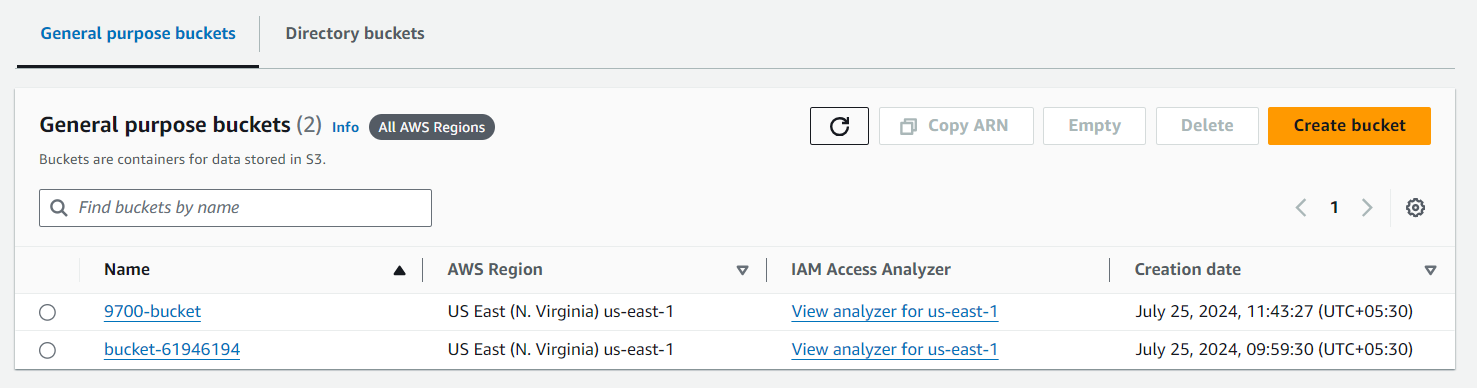
**EC2 Provision using Terraform:-**

1. provider "aws" {
2. region = "us-east-1"
3. }
4. resource "aws\_instance" "example" {
5. ami = "ami-0427090fd1714168b"
6. instance\_type = "t2.micro"
7. key\_name = "aws"
8. tags = {
9. Name = "instance"
10. }
12. }

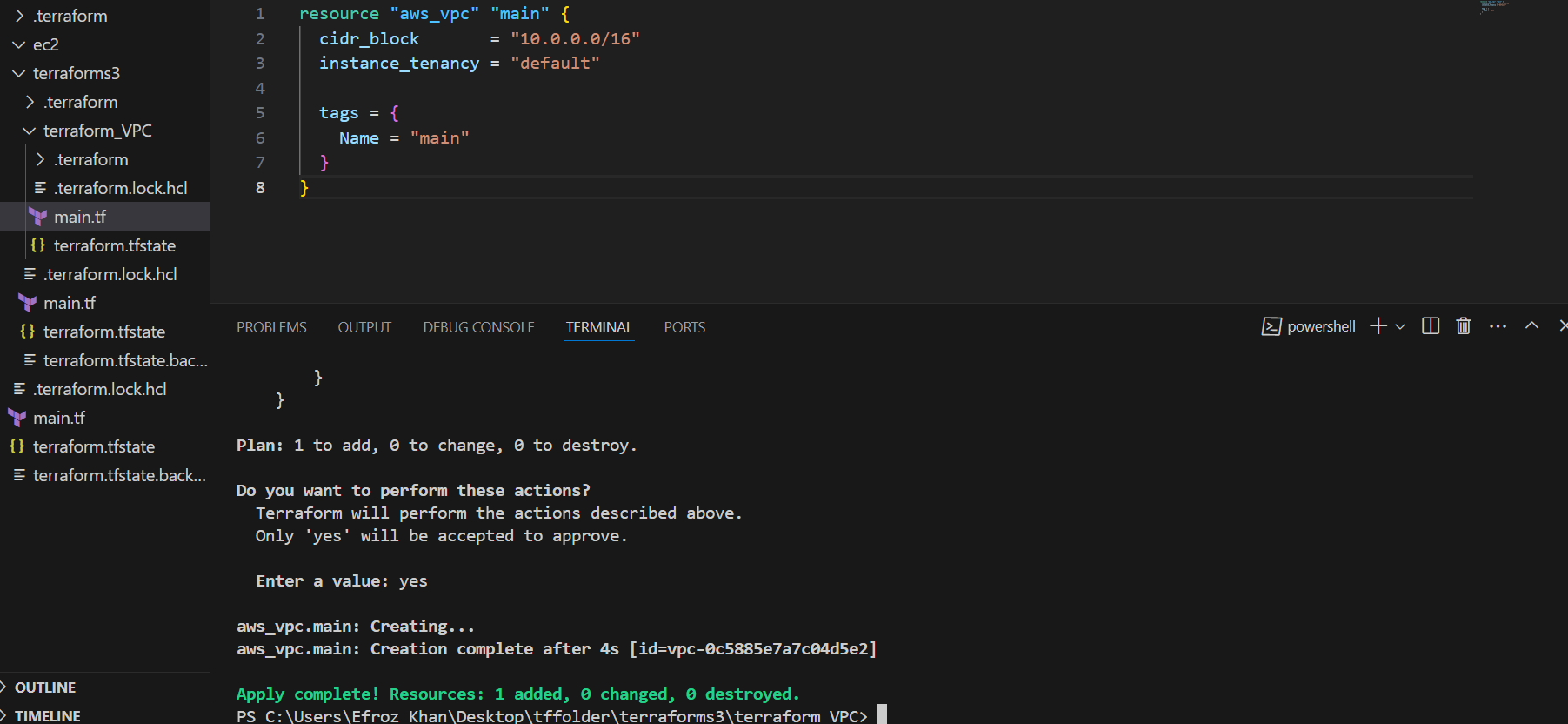


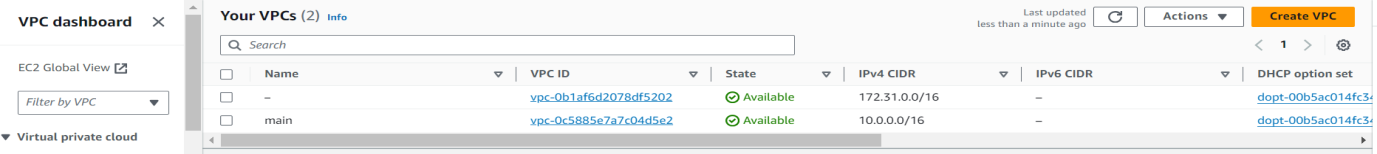
**S3 Provision using terraform** :-





**VPC Provisioning:-**

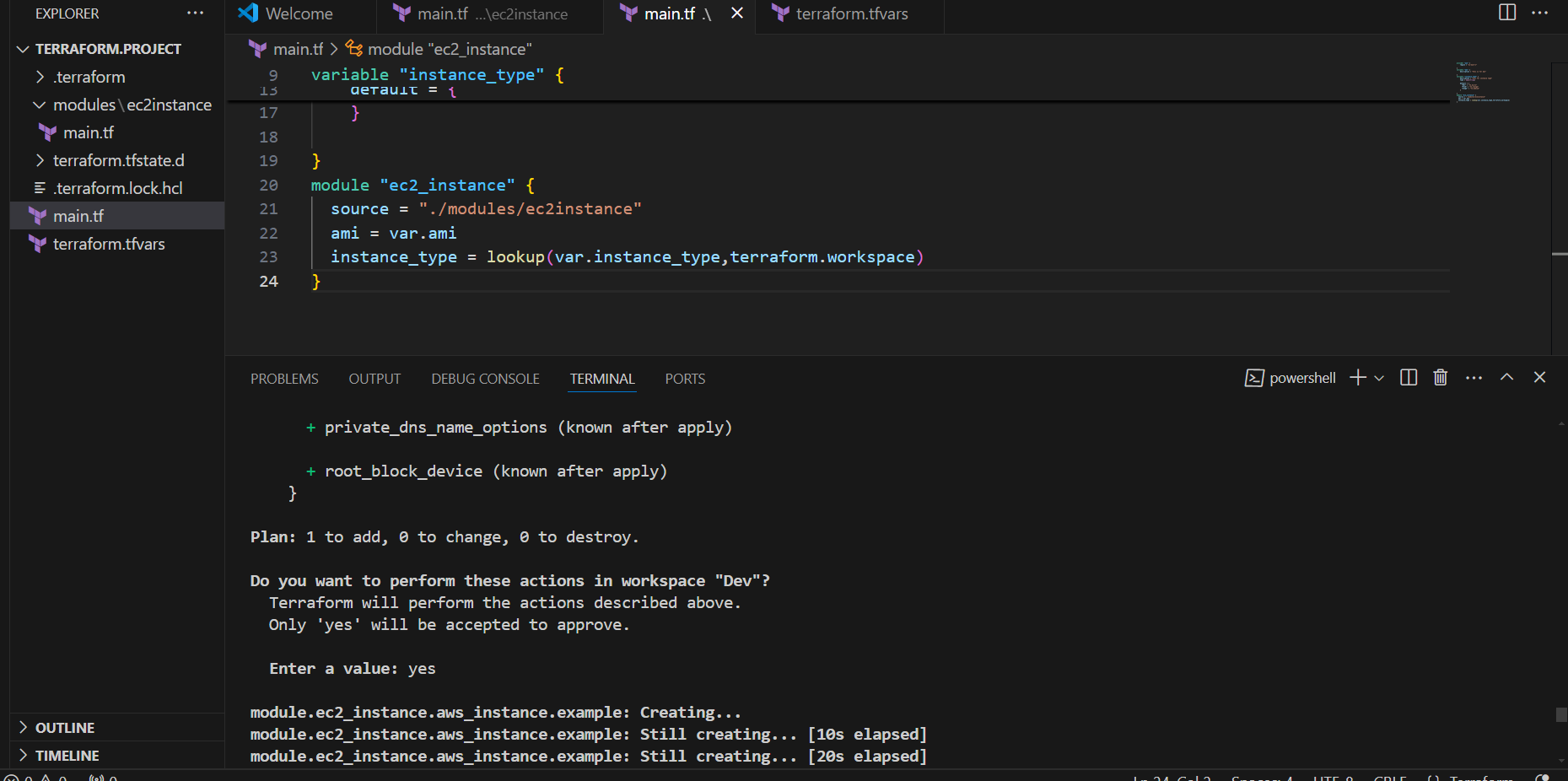




1. Provision ec2 for 3 different environments (Dev, Staging and Prod) using terraform

workspaces.

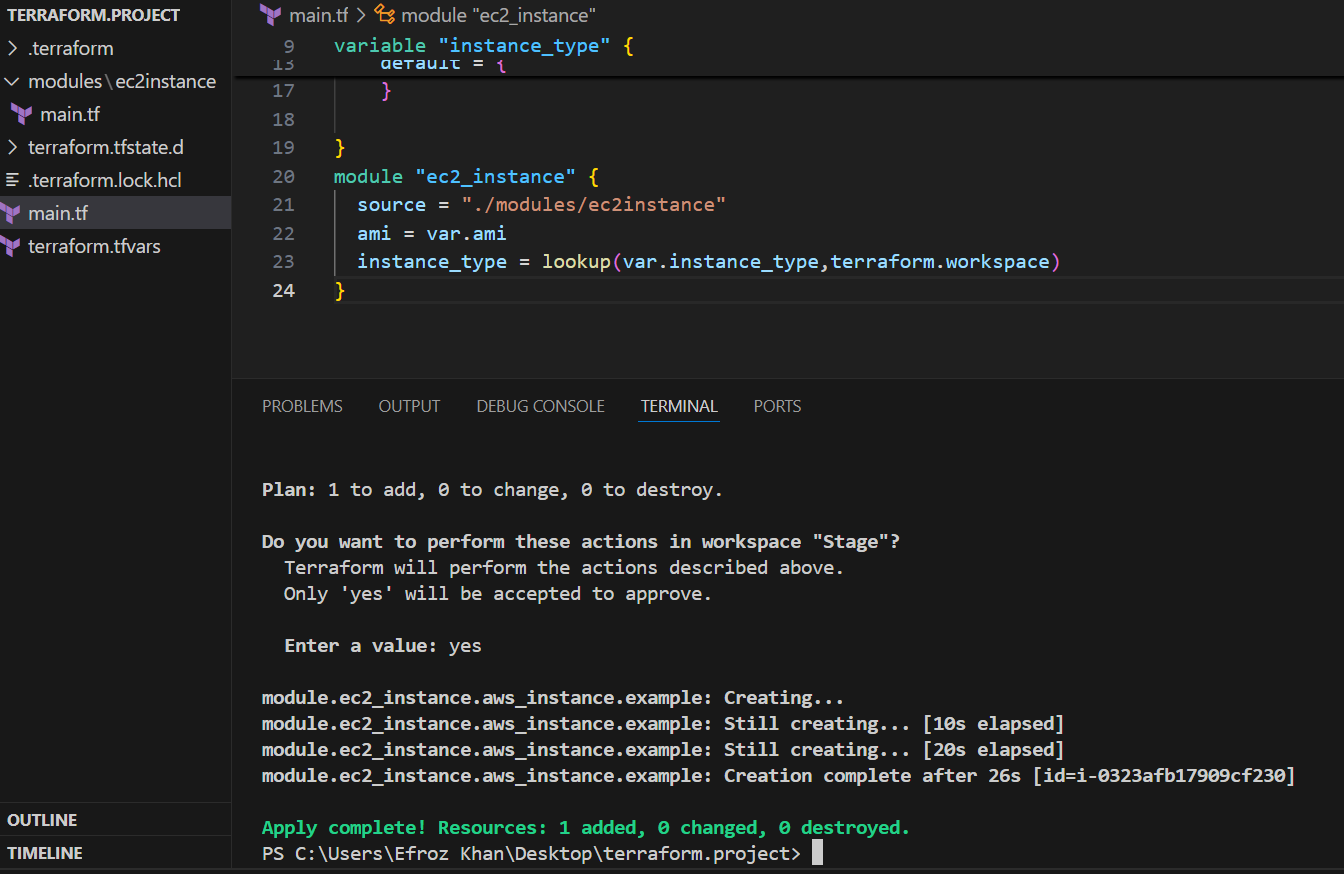
**Dev workspace Environment:-**



Dev environment output:-

C:\Users\Efroz Khan\Pictures\Screenshots\Screenshot 2024-07-25 175652.png

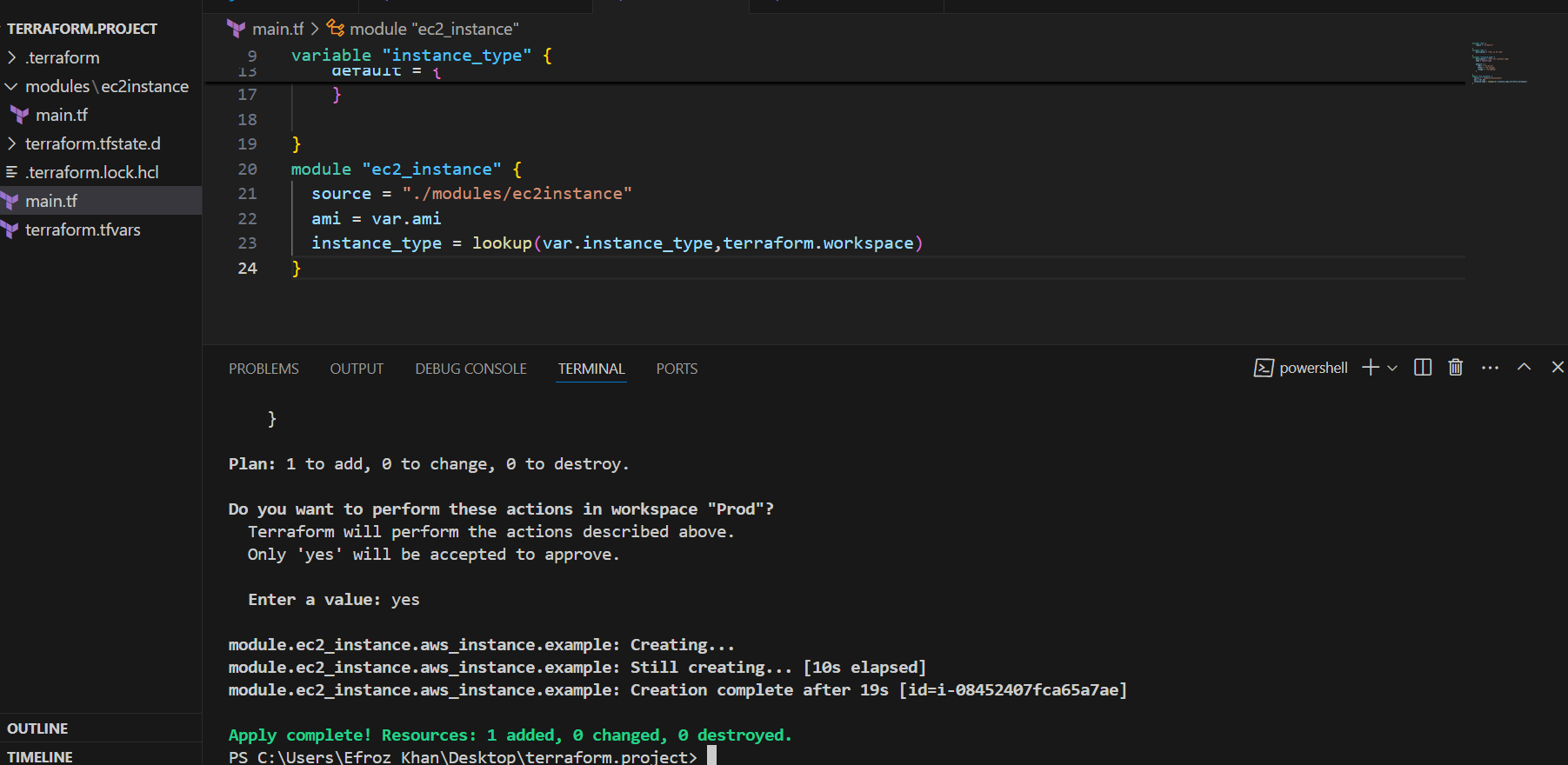
**Stage Workspace Environment:-**



**Stage Environment OutPut:-**

C:\Users\Efroz Khan\Pictures\Screenshots\Screenshot 2024-07-25 175848.png

**Prod Work space Environment:-**



**Prod Workspace OutPut:-**

C:\Users\Efroz Khan\Pictures\Screenshots\Screenshot 2024-07-25 180033.png