**1) Watch terraform-05 video.**

**2) Execute the script shown in video.**

\* Now am creating s3 bucket using s3-bucket resource

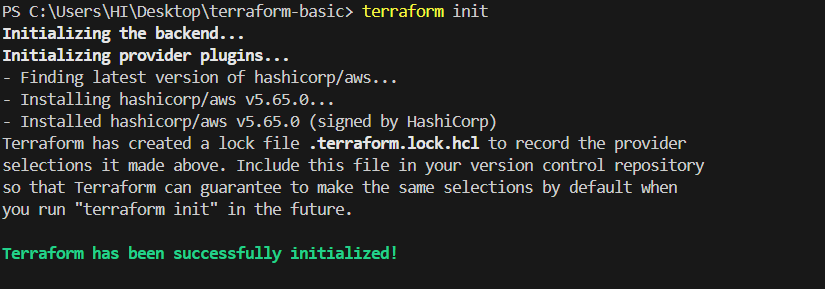
resource "aws\_s3\_bucket" "s3\_bucket" {

  bucket = "s3backend36"

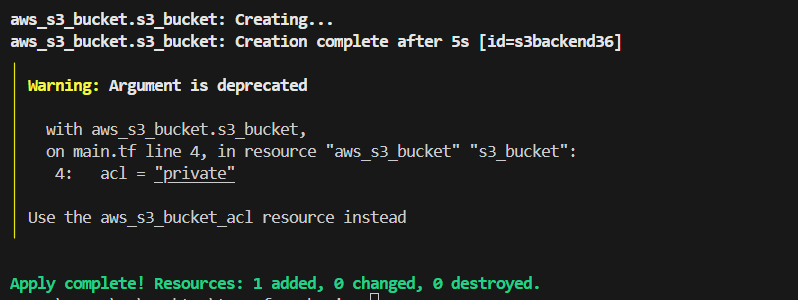
  acl = "private"

}

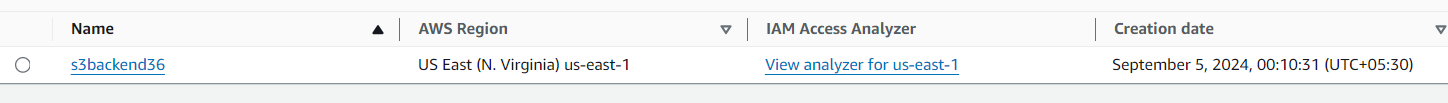
\* once initialize , >terraform file will create and it downloads the all dependencies of aws\_s3\_bucket



\* Executing **terraform apply ,** it will create s3 bucket named “s3backend36”



\* New s3 bucket has been created



\* Now I want to create dynamodb table using below resources and attributes

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

  bucket = "s3backend36"

  acl = "private"

}

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

  name = "terraform-state-lock-dynamo"

  hash\_key = "LockID"

  read\_capacity = 20

  write\_capacity = 20

  attribute {

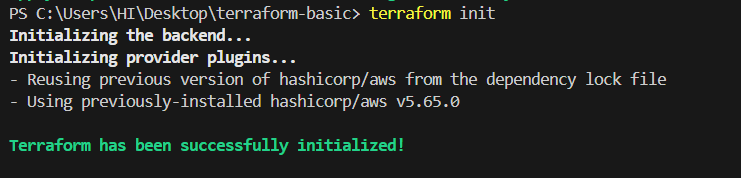
    name = "LockID"

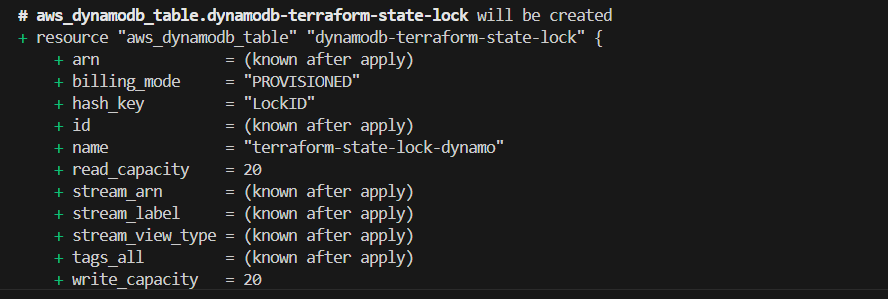
    type = "S"

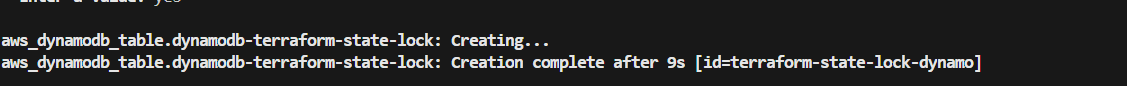
  }

}

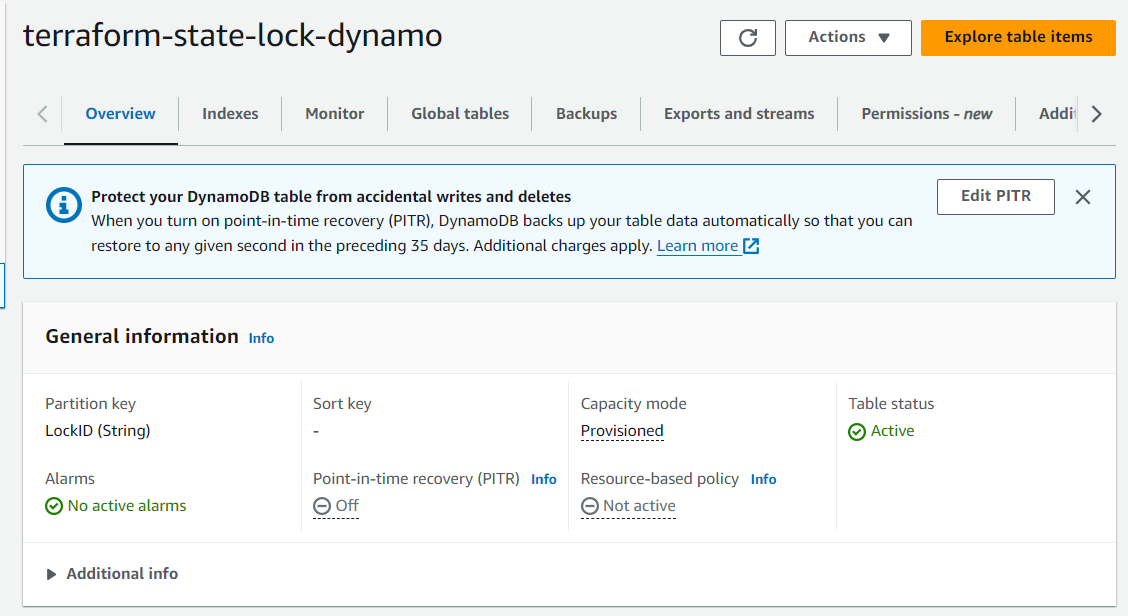
\* Enter **Terraform init** to install the dependencies for dynamodb provider







\* After enter **terraform apply** , it will creats dynamodb table



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

  bucket = "s3backend36"

  acl = "private"

}

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

  name = "terraform-state-lock-dynamo"

  hash\_key = "LockID"

  read\_capacity = 20

  write\_capacity = 20

  attribute {

    name = "LockID"

    type = "S"

  }

}

terraform {

  backend "s3" {

    bucket = "s3backend36"

    dynamodb\_table = "terraform-state-lock-dynamo"

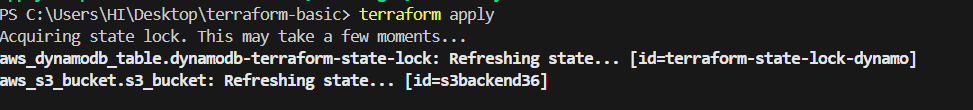
    key = "terraform.tfstate"

    region = "us-east-1"

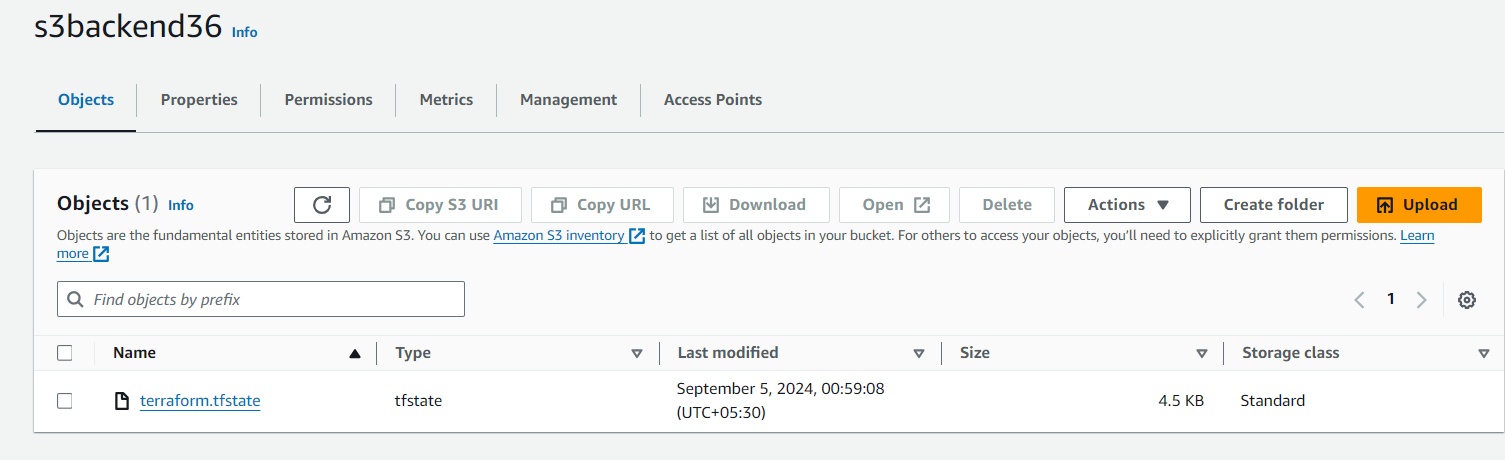
  }

}

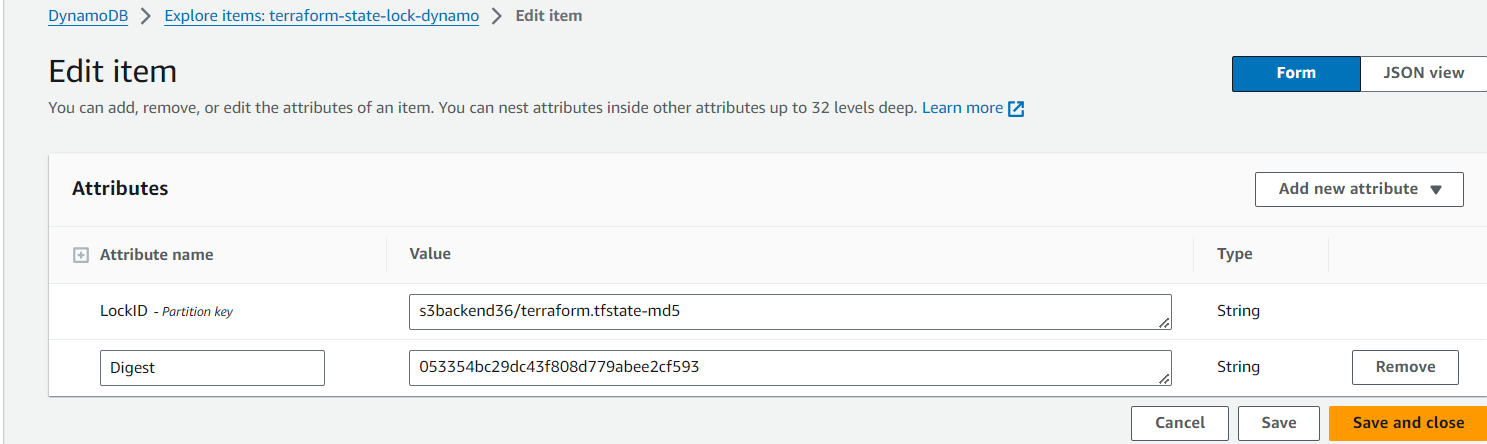
\*



\* terraform.tfstate has been pushed to the s3 and dynamodb table



\* we can find LockID in **terrafrm.tfstate** file which is in dynamodb table



\* Now am

resource "aws\_instance" "webserver" {

  ami = "ami-0182f373e66f89c85"

  instance\_type = "t2.micro"

  key\_name = "linux"

  subnet\_id     = aws\_subnet.my\_subnet.id

  # Security group specified by ID (not by GroupName)

  vpc\_security\_group\_ids = [aws\_security\_group.my\_sg.id]

  tags = {

    NAME = "First server"

  }

}

resource "aws\_vpc" "my\_vpc" {

  cidr\_block = "198.168.0.0/16"

}

resource "aws\_subnet" "my\_subnet" {

  vpc\_id     = aws\_vpc.my\_vpc.id

  cidr\_block = "198.168.0.0/28"

}

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

  vpc\_id = aws\_vpc.my\_vpc.id

  ingress {

    from\_port   = 22

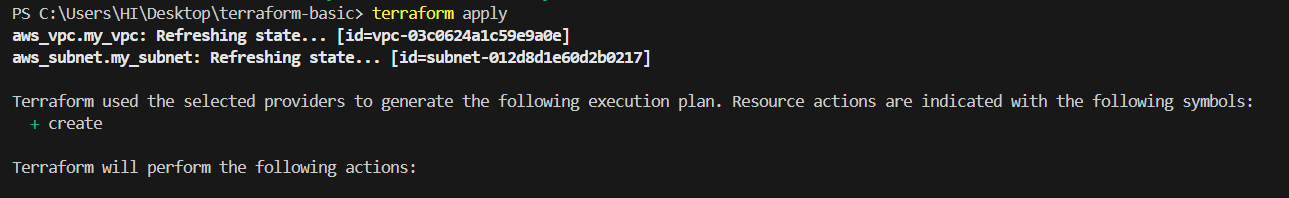
    to\_port     = 22

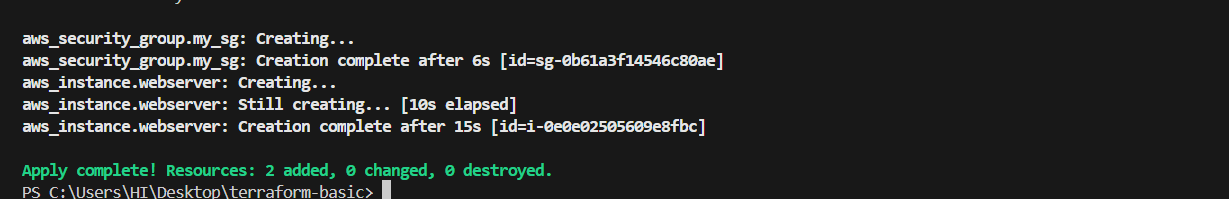
    protocol    = "tcp"

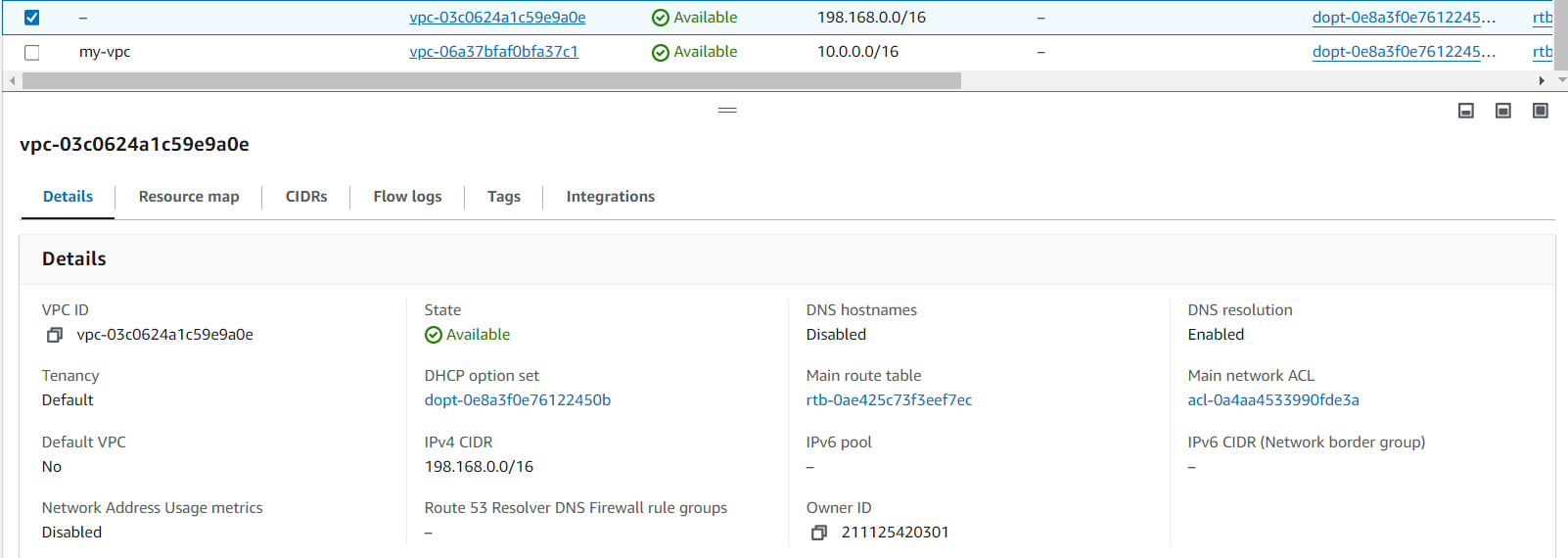
    cidr\_blocks = ["0.0.0.0/0"]

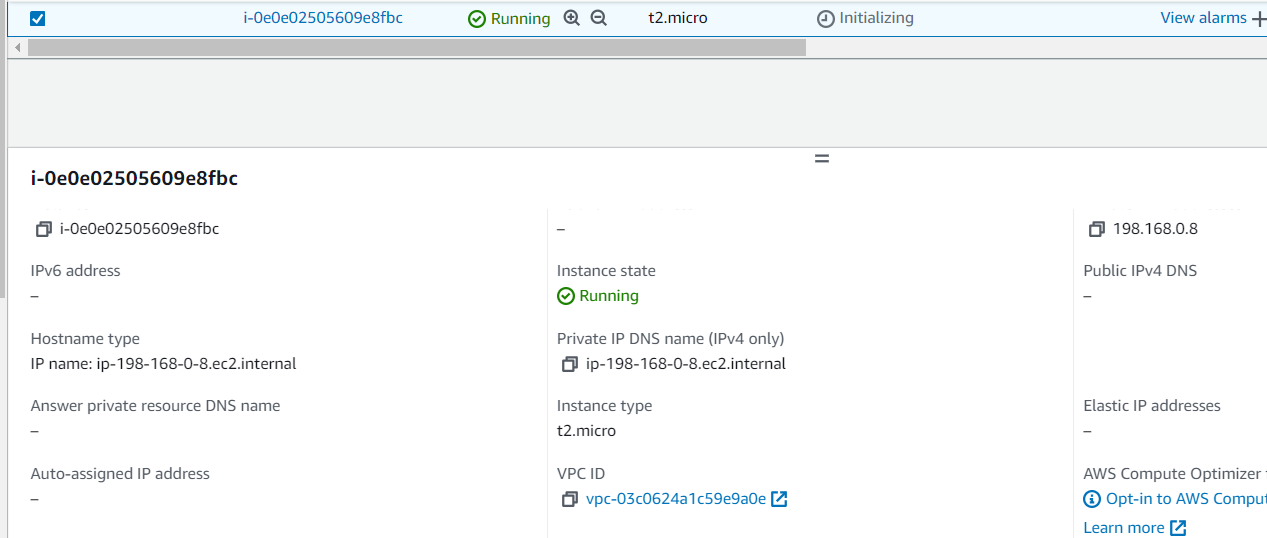
  }

}









================================================================

\* After creating new instance again I tried to create another one

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

  ami = "ami-02c21308fed24a8ab"

  instance\_type = "t2.medium"

  key\_name = "linux"

 subnet\_id = aws\_subnet.my\_subnet.id

 # Security group specified by ID (not by GroupName)

 vpc\_security\_group\_ids = [aws\_security\_group.my\_sg.id]

 tags = {

 NAME = "Second server"

 }

}

resource "aws\_vpc" "my\_vpc" {

 cidr\_block = "198.168.0.0/16"

}

resource "aws\_subnet" "my\_subnet" {

 vpc\_id = aws\_vpc.my\_vpc.id

 cidr\_block = "198.168.0.0/28"

}

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

 vpc\_id = aws\_vpc.my\_vpc.id

 ingress {

 from\_port = 22

 to\_port = 22

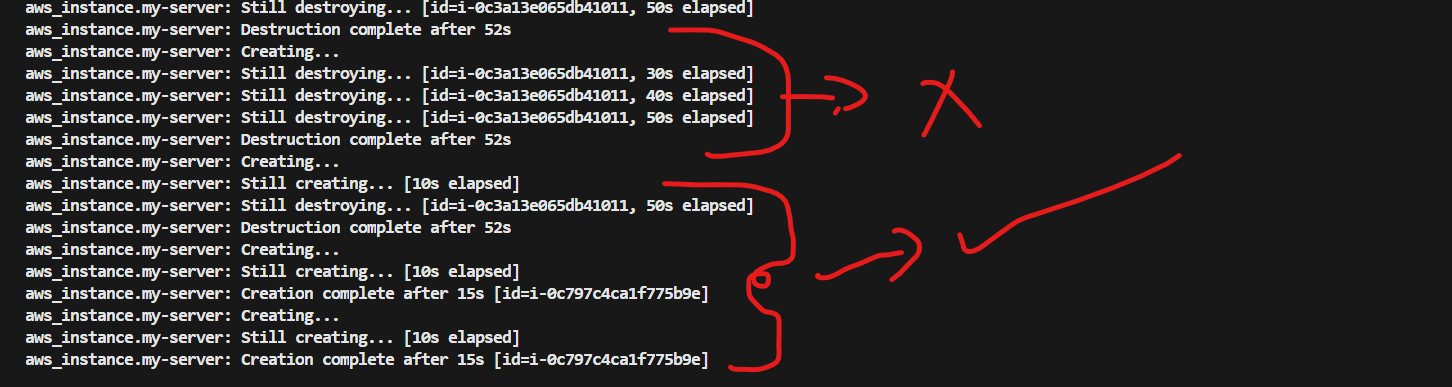
 protocol = "tcp"

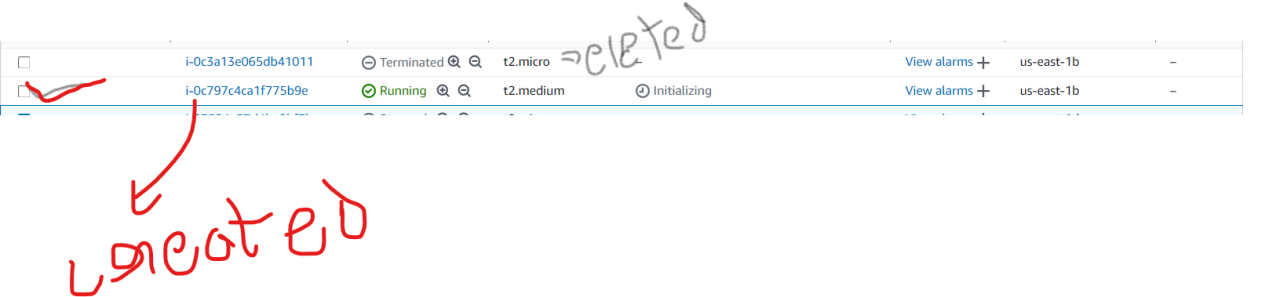
cidr\_blocks = ["0.0.0.0/0"]

 }

}

\* This time it is destroying the existing instance and creating the new instance





* Now am trying to **modify** the **instance type** and **ami id** by using life cycle(prevent\_destroy)

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

  ami = "ami-02c21308fed24a8ab"

  instance\_type = "t2.micro"

  key\_name = "linux"

lifecycle {

  prevent\_destroy = true

}

 subnet\_id = aws\_subnet.my\_subnet.id

 # Security group specified by ID (not by GroupName)

 vpc\_security\_group\_ids = [aws\_security\_group.my\_sg.id]

 tags = {

 NAME = "Second server"

 }

}

resource "aws\_vpc" "my\_vpc" {

 cidr\_block = "198.168.0.0/16"

}

resource "aws\_subnet" "my\_subnet" {

 vpc\_id = aws\_vpc.my\_vpc.id

 cidr\_block = "198.168.0.0/28"

}

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

 vpc\_id = aws\_vpc.my\_vpc.id

 ingress {

 from\_port = 22

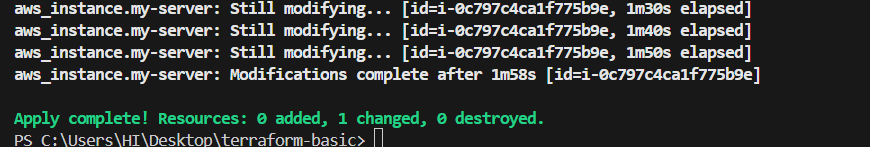
 to\_port = 22

 protocol = "tcp"

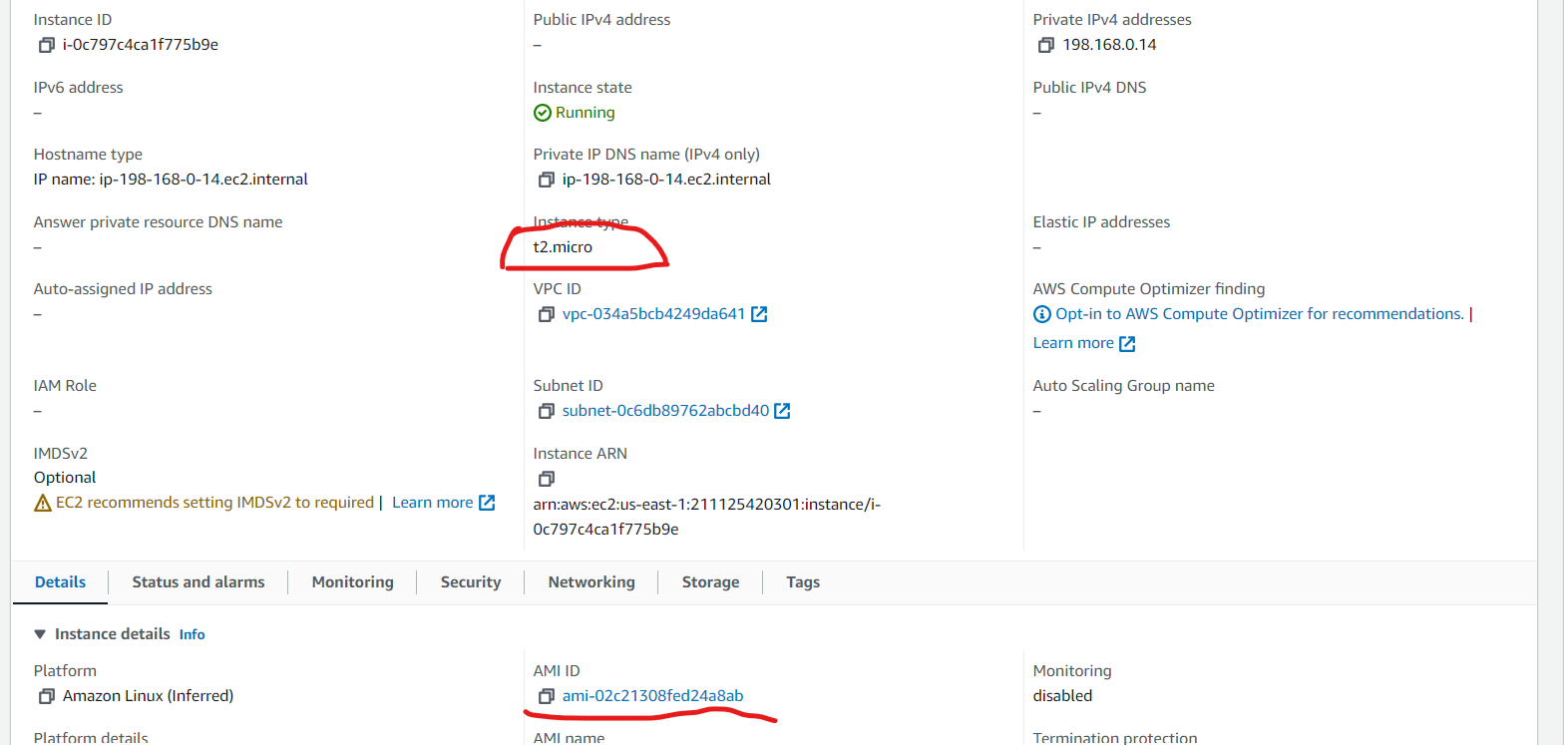
cidr\_blocks = ["0.0.0.0/0"]

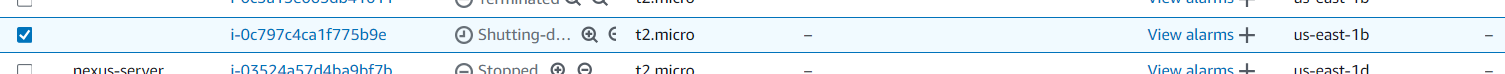
 }

}



\* the changes has been done





============================================================================

\*Now I want to see the public ip into the text file using **provisioner**

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

  ami = "ami-025258b26b492aec6"

  instance\_type = "t2.micro"

  key\_name = "america"

tags = {

  NAME = "first-server"

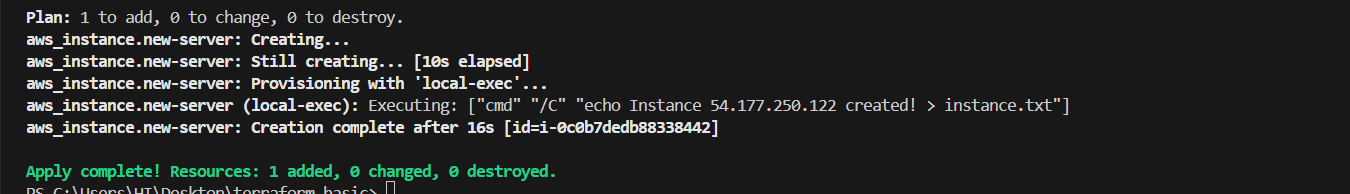
}

provisioner "local-exec" {

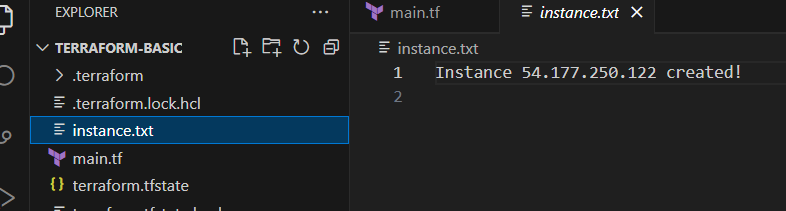
  command = "echo Instance ${aws\_instance.new-server.public\_ip} created! > instance.txt"

 }

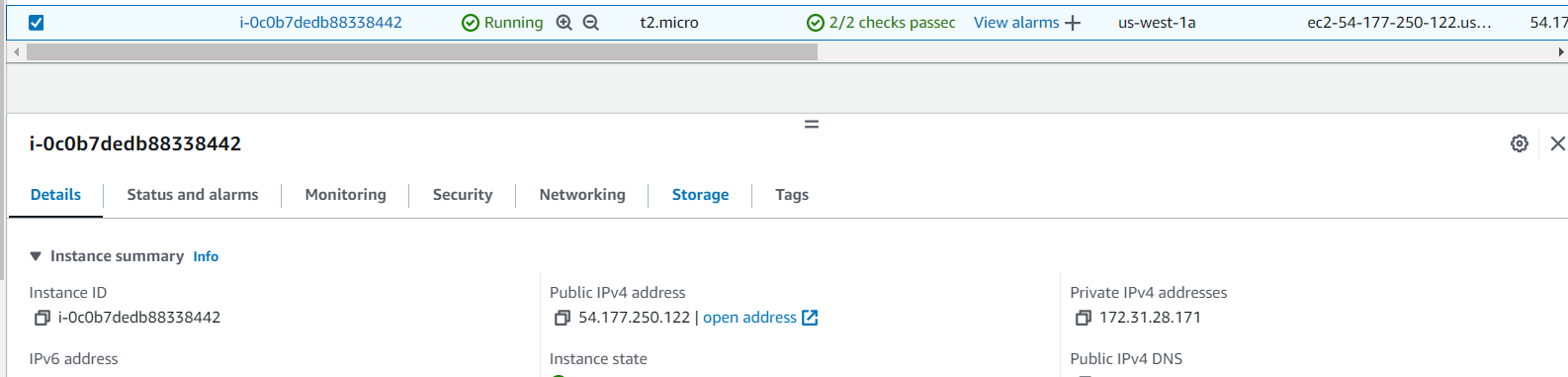
}



\* In the text file we can find the **public ip**



\* New Instance has been created



\* Now I want to see the public ip into the text file using **provisioner** ,but am given different filepath

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

  ami = "ami-025258b26b492aec6"

  instance\_type = "t2.micro"

  key\_name = "america"

tags = {

  NAME = "first-server"

}

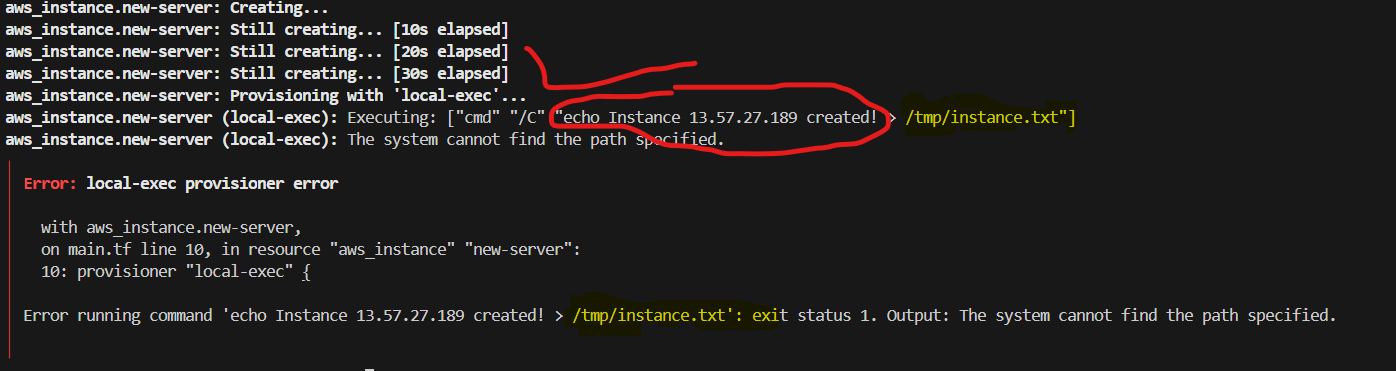
provisioner "local-exec" {

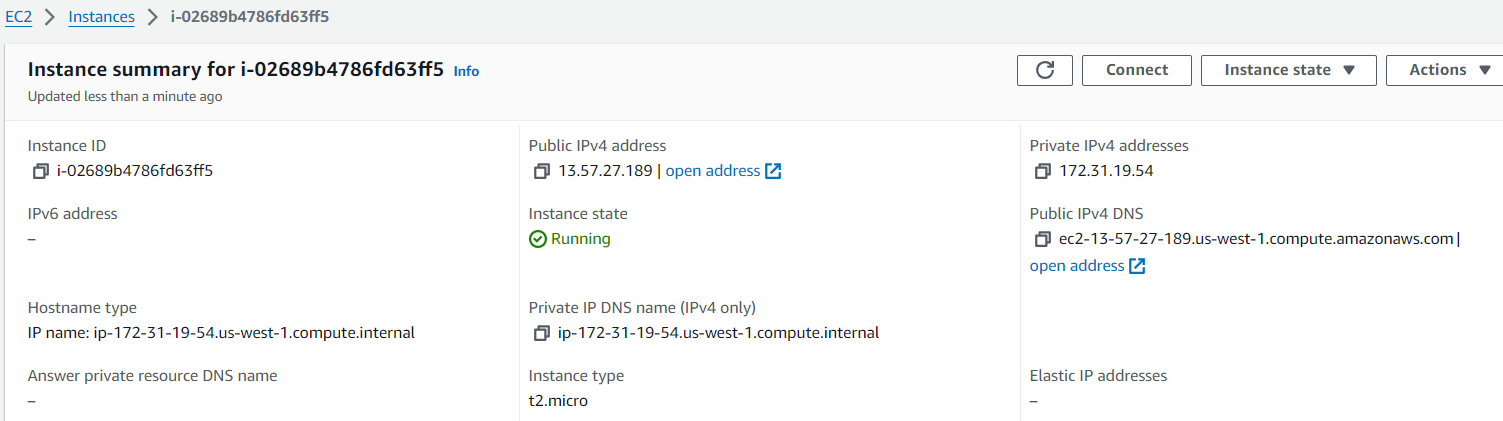
  command = "echo Instance ${aws\_instance.new-server.public\_ip} created! > /tmp/instance.txt"

 }

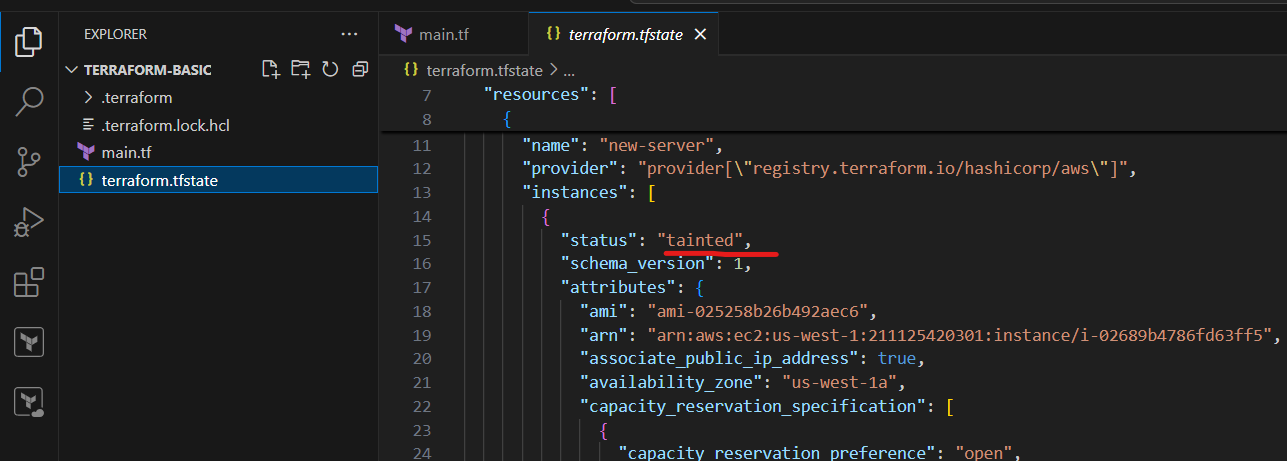
}

\* Now it will creates new instance, but throws an error for file path is not existed





\* when this case we can see the statefile status of infrastructure is showing as “**tainted**”,due to if error will came while creating instance it will ignore the provision block



\* By using this I have creating an instance using **provisioner** and shows the output in a text file(instance.txt)

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

  ami = "ami-025258b26b492aec6"

  instance\_type = "t2.micro"

  key\_name = "america"

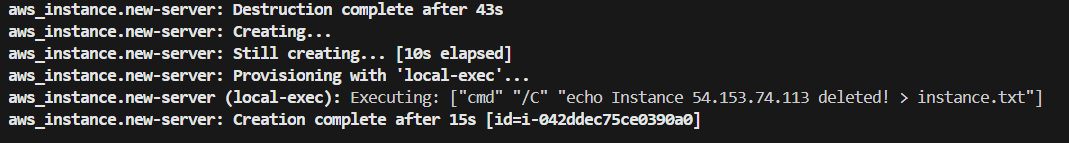
provisioner "local-exec" {

  command = "echo Instance ami created! > instance.txt"

 }

}

\* created an instance with tags



\* Then I have to adding “when = destroy” in **provisioner** to destroy the existing instance

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

  ami = "ami-025258b26b492aec6"

  instance\_type = "t2.micro"

  key\_name = "america"

provisioner "local-exec" {

  when = destroy

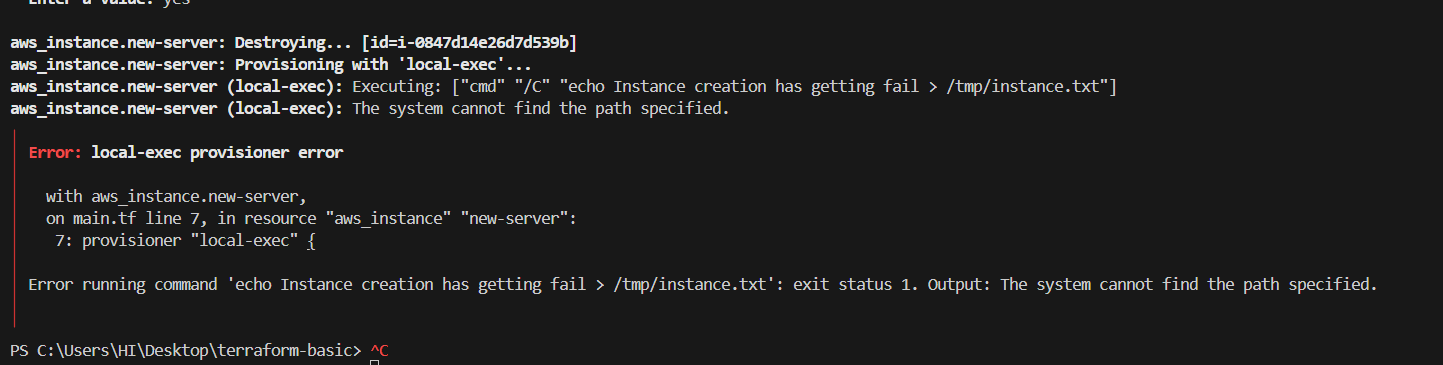
  command = "echo Instance ami deleted! > instance.txt"

 }

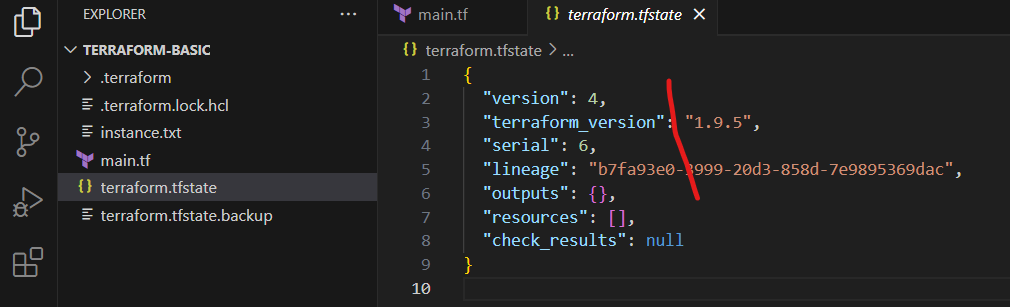
}

\* By using “**terraform destroy”** to it will deleting the exesting instance and it will shows output by local provision behaivour

\* By using this when to destroy if anything gets failed , it won’t delete the existing instance



\* The infrastructure as been destroyed



\* here am taking provisioner to take the output of public ip in a file while creating an instance

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

  ami = "ami-025258b26b492aec6"

  instance\_type = "t2.micro"

  key\_name = "america"

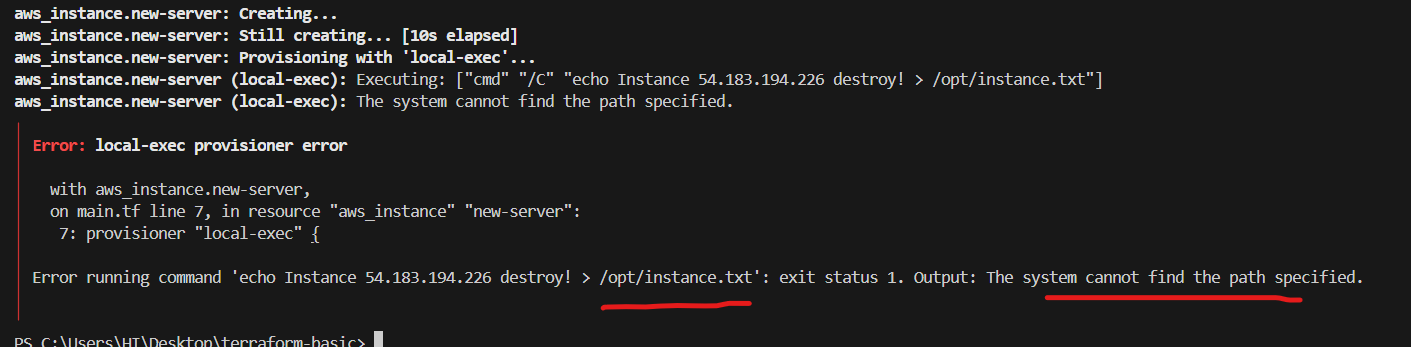
provisioner "local-exec" {

  command = "echo Instance ${aws\_instance.new-server.public\_ip} destroy! > /opt/instance.txt"

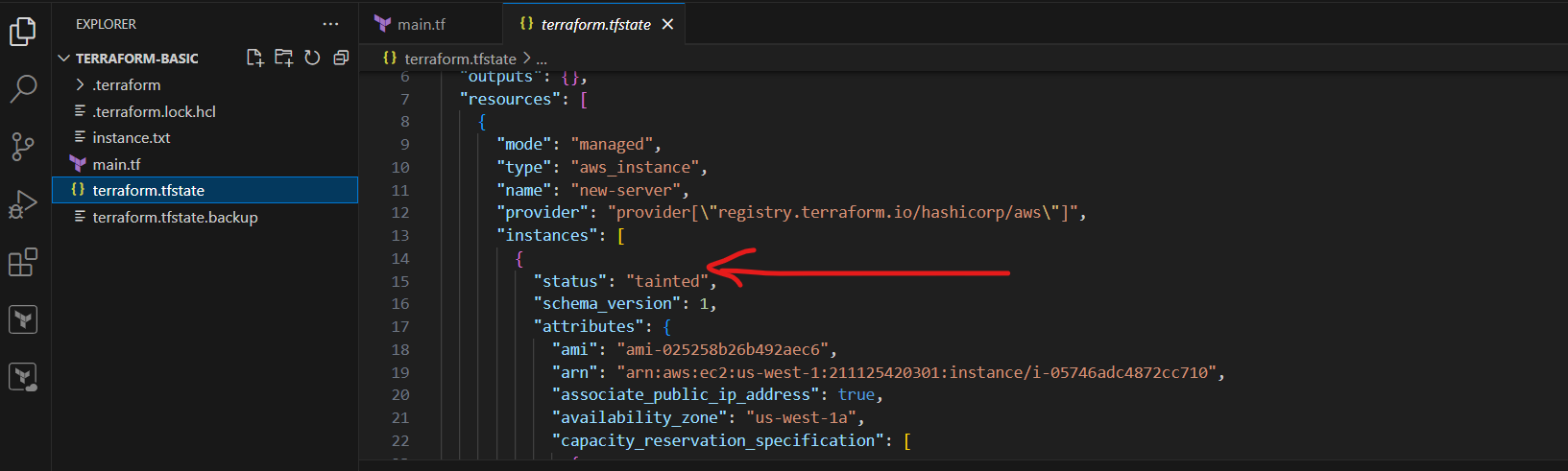
 }

}

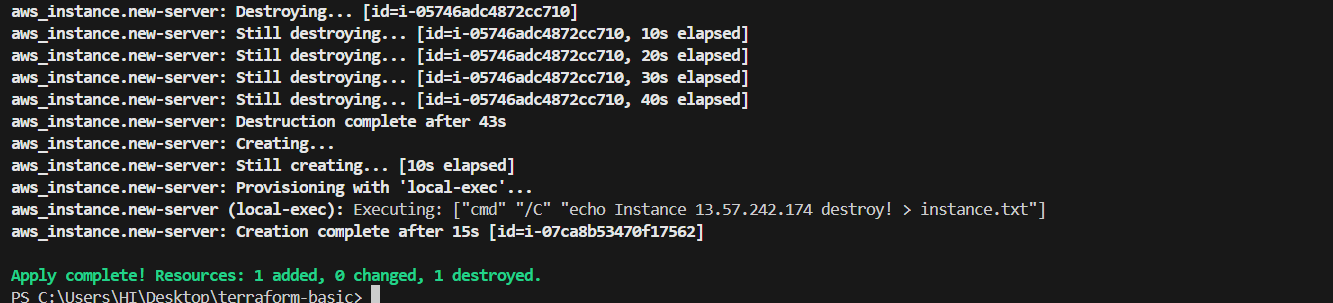
\* It will throws error due to command output file path is doesn’t exist.



\*When this case statefile status is showing as “tainted”

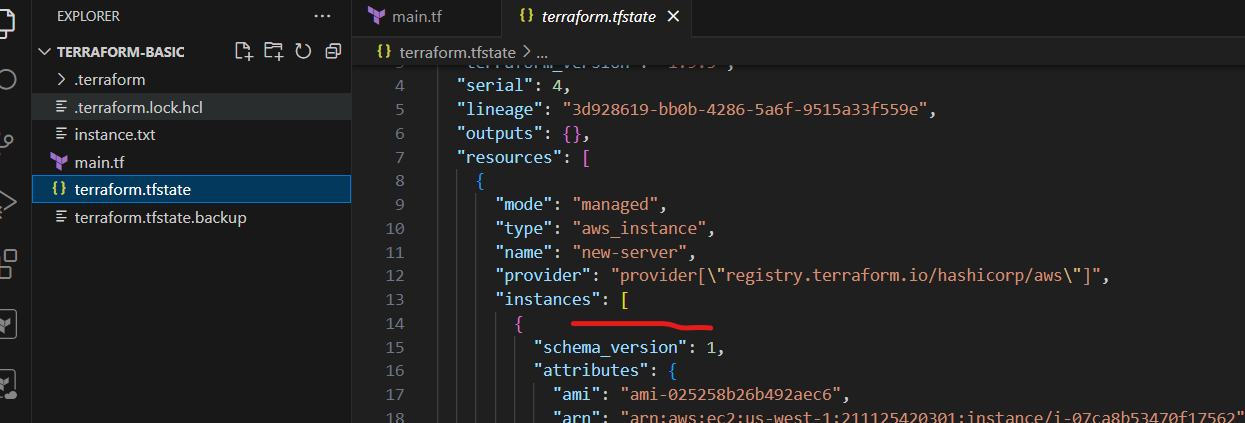


\* after giving the correct path in the privisioner command , Then execute **terraform apply** when it will executed successfully and it destroy the existing instance and creates new instance

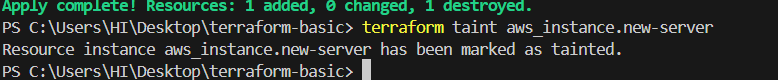


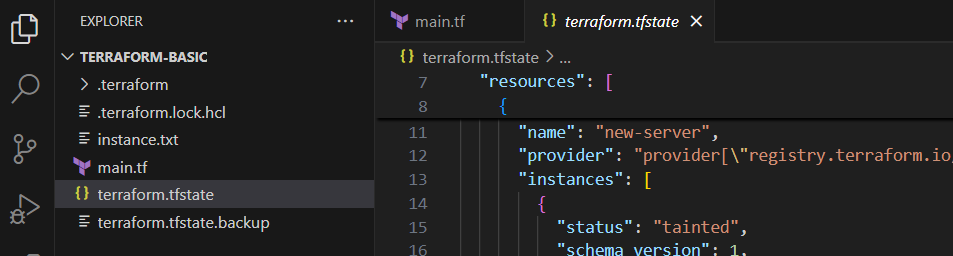
\* here we can see the **command output** of the present public ip in state file

\*the state file also updated with changes

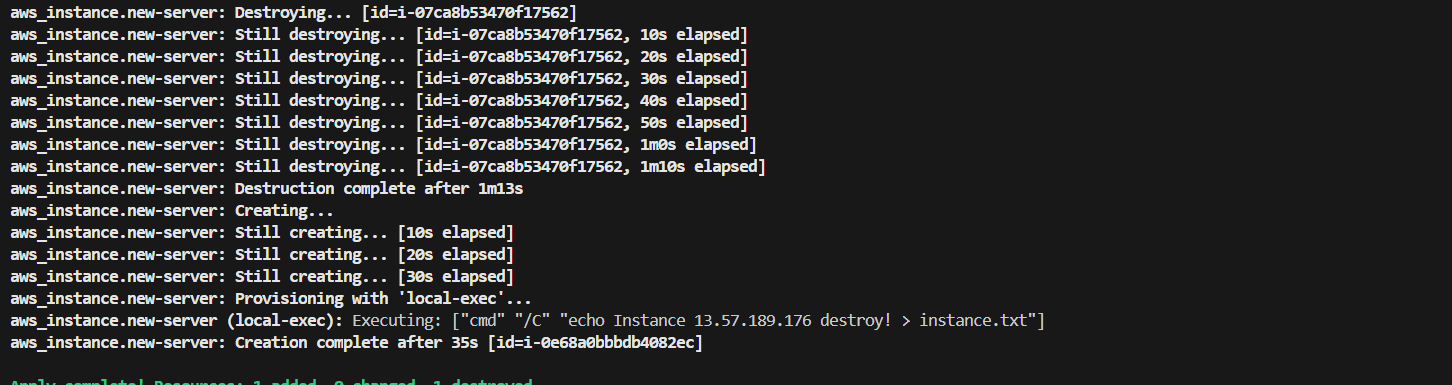


\* Now manually marked the resource as tainted



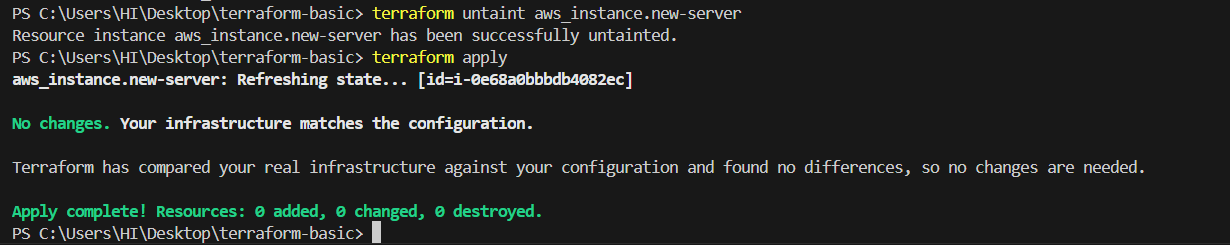


\*any of the moment statefile status marked as tainted, then we try to terraform apply it will destroy and create the instance



\* If in case the statefile status has marked as tainted, then changed to untainted the statefile status

Then apply it will shows as everything is clear(nothing to do the changes)



**Terraform Debugging**

**\***terraform apply will provide us logs/cause of the issue,

But still if we want to dig deeper then we need to export a variable

* Terraform providers 5 levels of logs

\* INFO

\* WARNING

\* ERROR

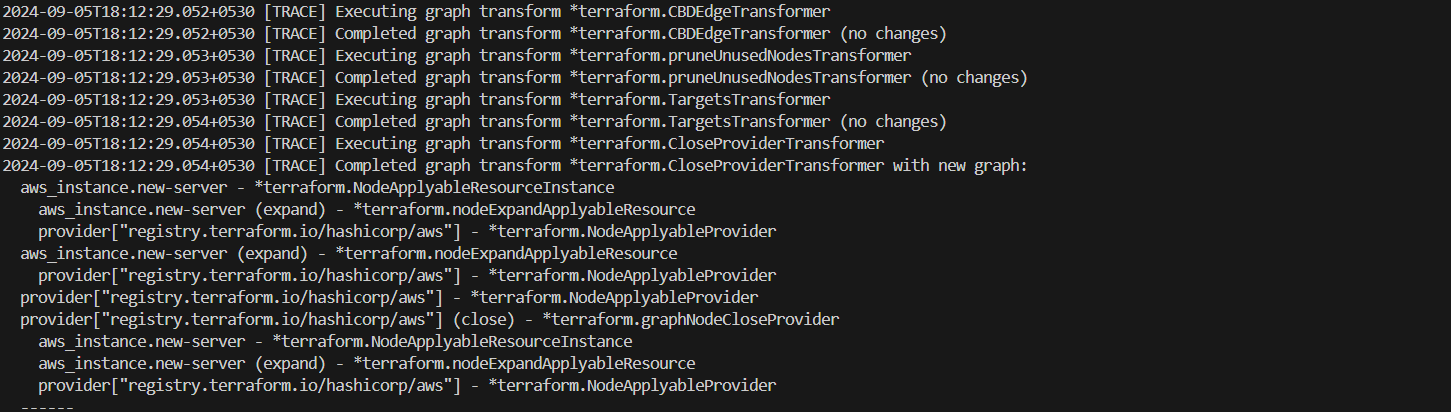
\* DEBUG

\* TRACE

**To store the logs as temporary**

Set-Item -Path env:TF\_LOG -value “TRACE”

To get the complete log files when eneter terraform plan

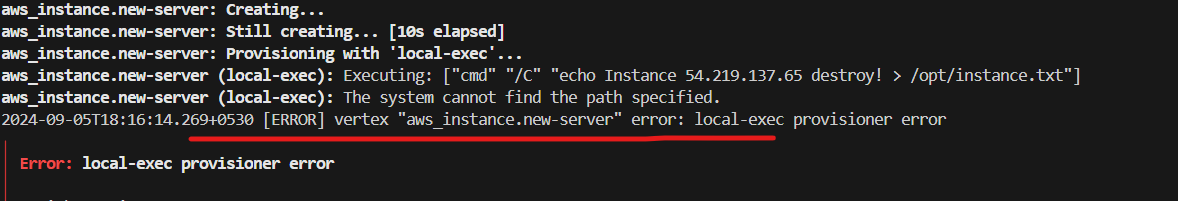


Set-Item -Path env:TF\_LOG -value “ERROR”

**\***To get the complete log files when enter **terraform apply**



\* here we can see the error log where It’s getting failed



**\* To store the logs as permanent:**

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

**\***to store the logs in terraform.log file as permanent

**Set-Item -Path env:TF\_LOG -value "ERROR"**

**\*** we can store error logs in the terraform.log file

By using the below **provisioner** we can do **terraform apply**

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

  ami = "ami-025258b26b492aec6"

  instance\_type = "t2.micro"

  key\_name = "america"

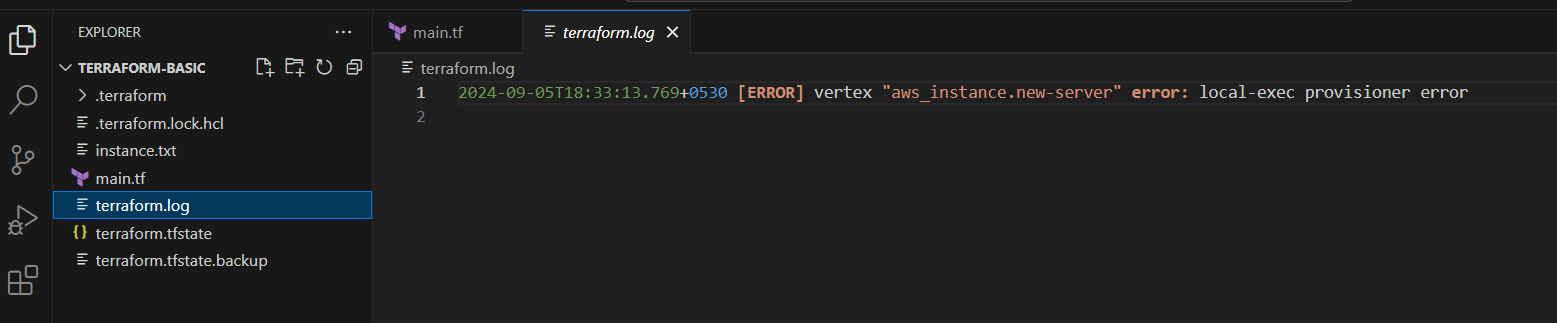
provisioner "local-exec" {

  command = "echo Instance ${aws\_instance.new-server.public\_ip} destroy! > /opt/instance.txt"

 }

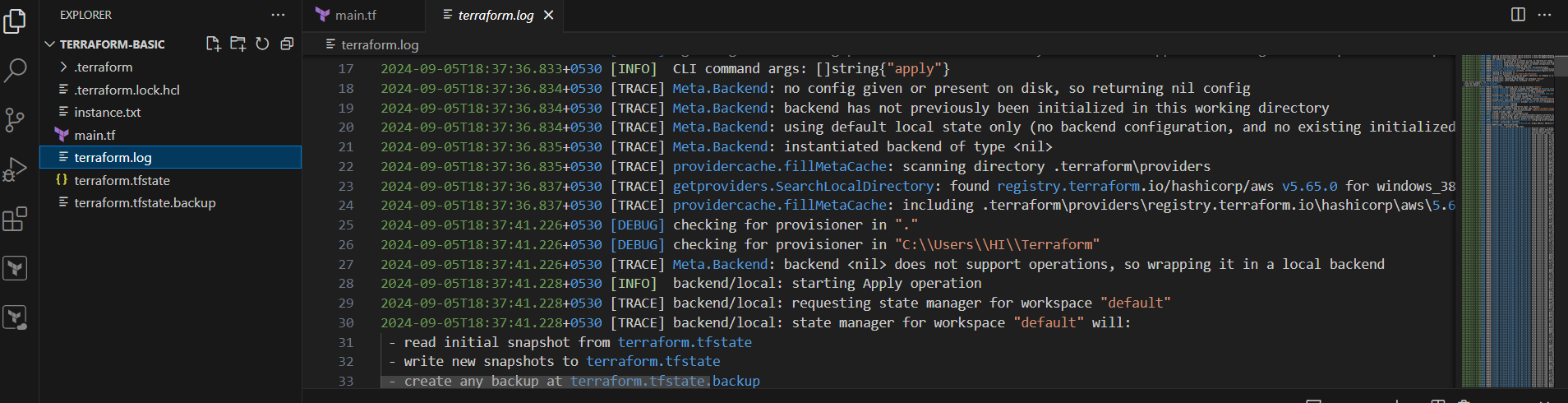
}

**\*** If any error logs will received save in the terraform logs file



**Set-Item -Path env:TF\_LOG -value "TRACE"**

\* After apply terraform the total logs will store in terraform log file



To unset or disable the logs

Unset TF\_LOG\_PATH

**\*** we need to import the existing infrastructure of instance to another instance configure the resource block as well using

**“terraform import aws\_instance.manual i-0b2c0fbbcbbd0390e”**

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

  ami = "ami-025258b26b492aec6"

  instance\_type = "t2.micro"

  key\_name = "america"

}

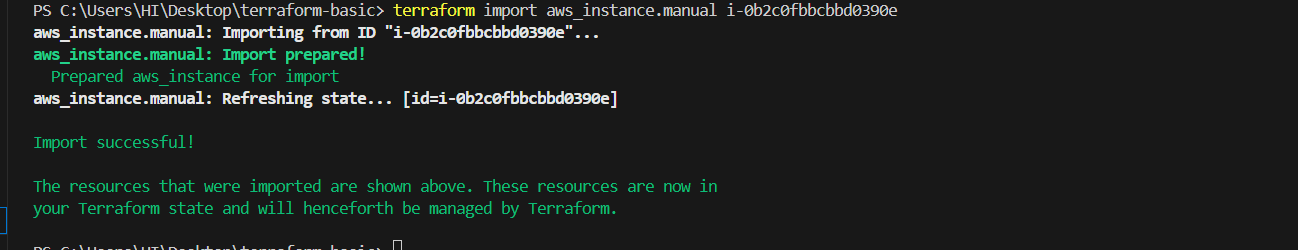
resource "aws\_instance" "manual" {

  ami = "ami-025258b26b492aec6"

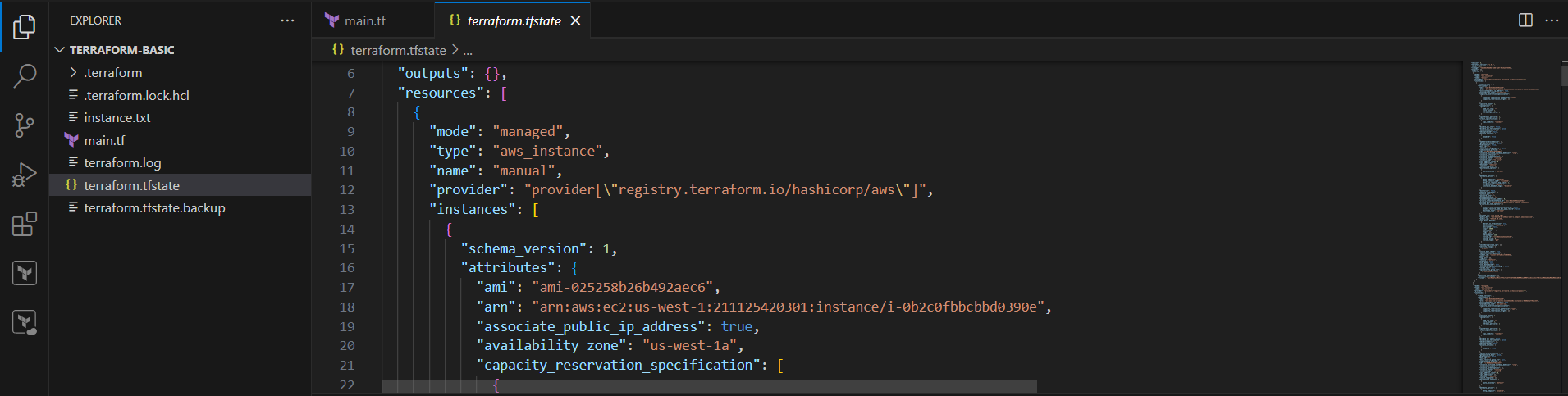
  instance\_type = "america"

}

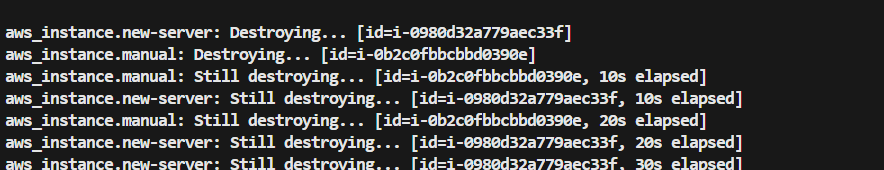
**\* Now it’s imported to anither instance**



**\* In the state file we can see the details**



\* Now I tried to destroy , then it will delete two instances



* **Now am trying to** **create multiple ec2-instances using count**

Main.tf

resource "aws\_instance" "myinstance" {

  ami                    = var.ami[count.index]

  instance\_type          = "t2.medium"

  key\_name               = "america"

  vpc\_security\_group\_ids = ["sg-0d955d1b514f784b4"]

count = length(var.ami)

}

Variabale.tf

variable "ami" {

  default = [

    "ami-025258b26b492aec6",

    "ami-025258b26b492aec6",

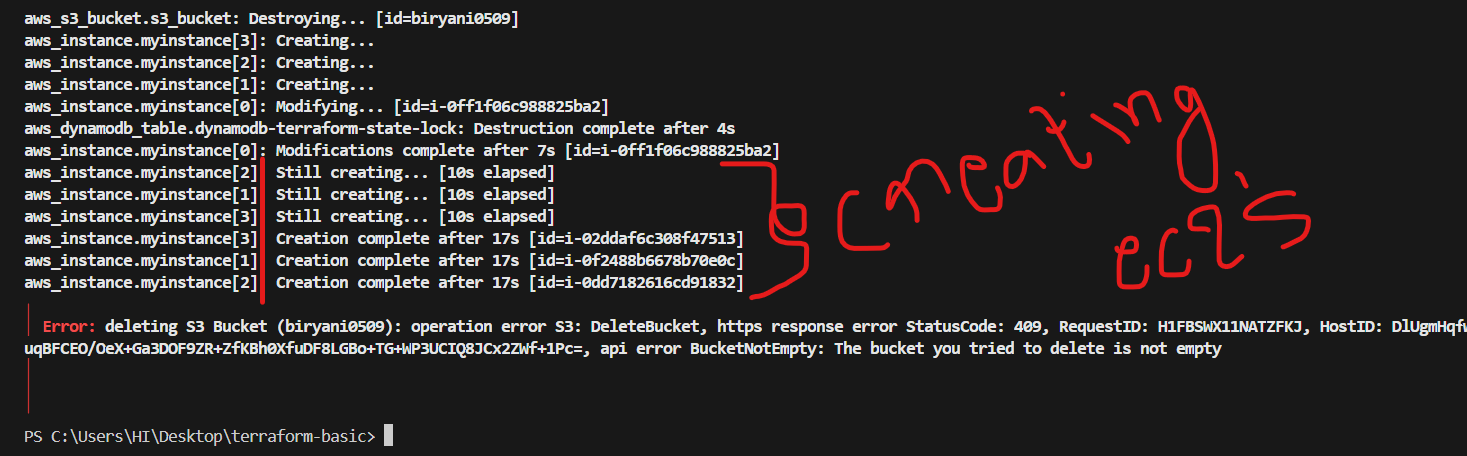
    "ami-025258b26b492aec6",

    "ami-025258b26b492aec6"

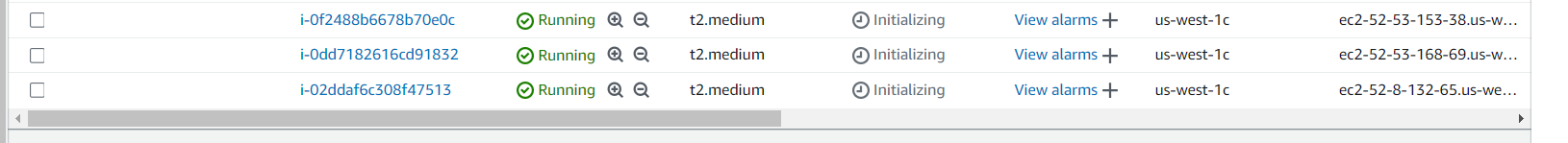
  ]

}

\* After entering apply it will destroy existing instances and createing 3 instances



\*3 Instances has been created with same **AMI-ID**



* **Now am creating instances using** **for-each Meta arguments**

\* Here am given for-each argument to create multiple instances

resource "aws\_instance" "myinstance" {

  ami                    = each.value

  instance\_type          = "t2.medium"

  key\_name               = "america"

  vpc\_security\_group\_ids = ["sg-0d955d1b514f784b4"]

for\_each = toset(var.ami)

}

variable "ami" {

  default = [

    "ami-025258b26b492aec6",

    "ami-025258b26b492aec6",

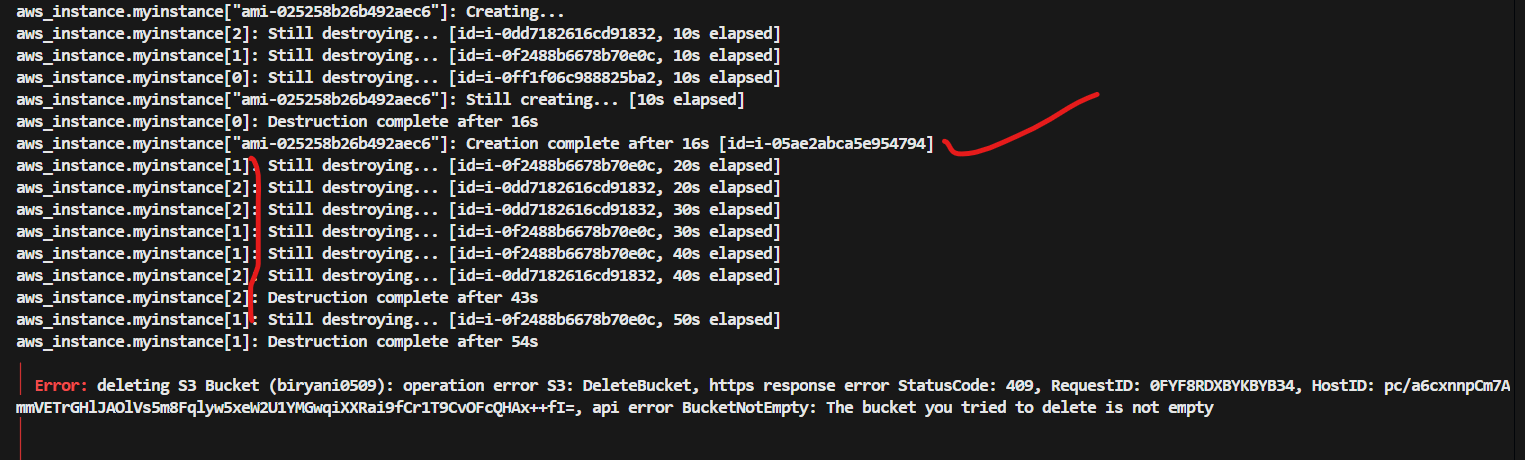
    "ami-025258b26b492aec6",

    "ami-025258b26b492aec6"

  ]

}

\* Here in this it’s created 1 instance due to which of 3 are already created in previous



1. **Create one ec2 instance with httpd installed using terraform script.**

**\*** First am giving resources for aws providers

\*for running the httpd using script in userdata

terraform {

  required\_providers {

    aws = {

      source  = "hashicorp/aws"

      version = "~> 2.70"

    }

  }

}

provider "aws" {

  region  = "us-west-1"

  access\_key = "AKIATCKAOIEGQREAP5OR"

  secret\_key = "pHlLznFJEGMLb4MjnwIWD7BJTnkrKU4hH32TjyKI"

}

resource "aws\_instance" "myinstance" {

  ami           = "ami-025258b26b492aec6"

  instance\_type = "t2.medium"

  key\_name = "america"

  tags = {

    Name = "My last instance"

  }

  vpc\_security\_group\_ids = [ "sg-0d955d1b514f784b4" ]

  user\_data = <<-EOF

    #!/bin/bash

    sudo yum -y install httpd

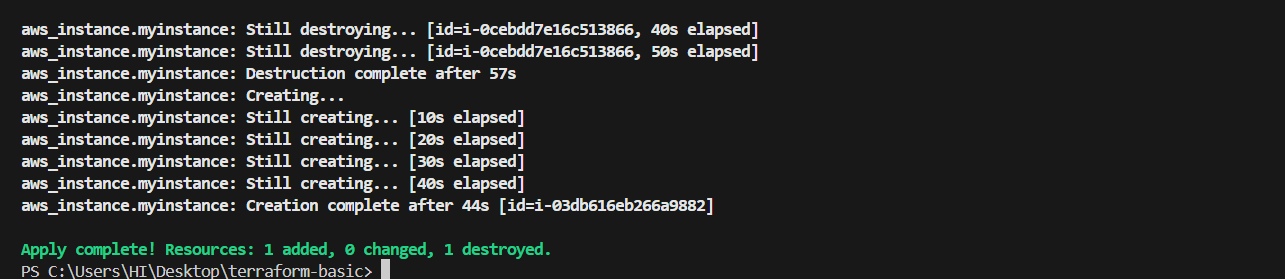
    sudo yum -y update

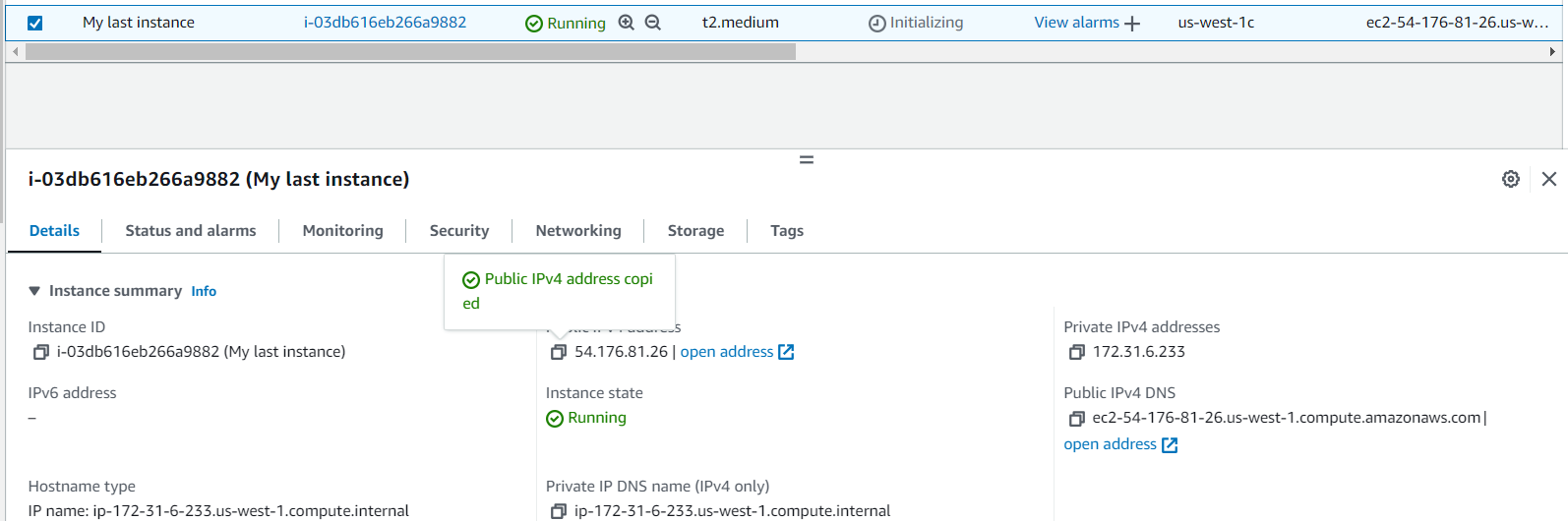
    systemctl start httpd

  EOF

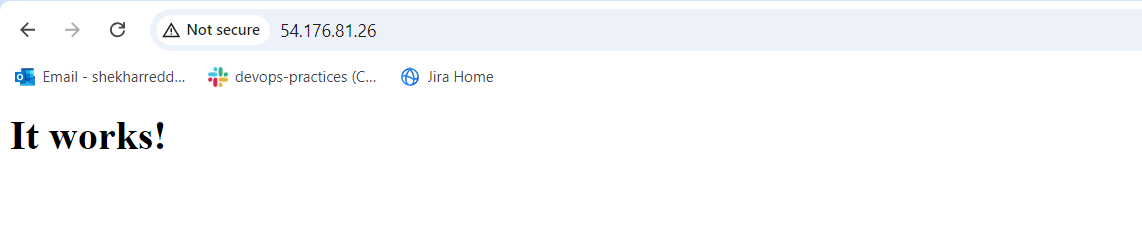
}

**\*** Now created an instance





\* it’s working on created instance public-IP



**4) Setup s3 as backend to the task 3.**

**\*** Using s3 bucket provider to install the dependencies and create the new bucket for **remote backend**

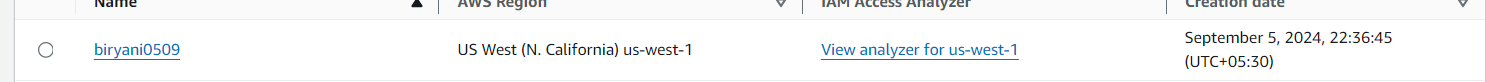
resource "aws\_s3\_bucket" "s3\_bucket" {

  bucket = "biryani0509"

  acl = "private"

}

**\*** New bucket has been created



**5)Setup dynamo db locking for task3.**

\*creating dynamodb table for state locking

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

  bucket = "biryani0509"

  acl = "private"

}

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

  name = "terraform-state-lock-dynamodb"

  hash\_key = "LockID"

  read\_capacity = 20

  write\_capacity = 20

  attribute {

    name = "LockID"

    type = "S"

  }

}

\* Dynamodb table has been created

