**1. Watch the Terraform-05 video.**

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

**3. Create one EC2 instance with httpd installed using a Terraform script.**

**4. Set up S3 as backend for task 3.**

**5. Set up DynamoDB locking for task 3.**

**1. Watch the Terraform-05 video.**

**Completed**

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

**TERRAFORM Remote state and state locking**

**S3 Bucket creation**

# Create the S3 bucket

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

bucket = "imran257s3backend" # change name if already taken globally

}

# Set ownership controls (to allow ACLs if needed)

resource "aws\_s3\_bucket\_ownership\_controls" "example" {

bucket = aws\_s3\_bucket.s3\_backend.id

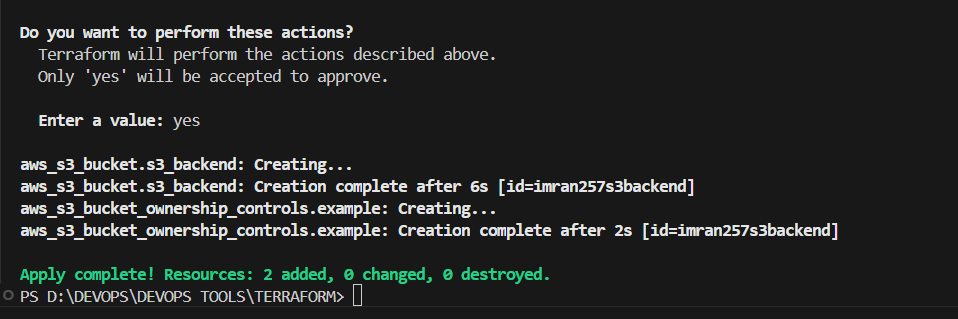
rule {

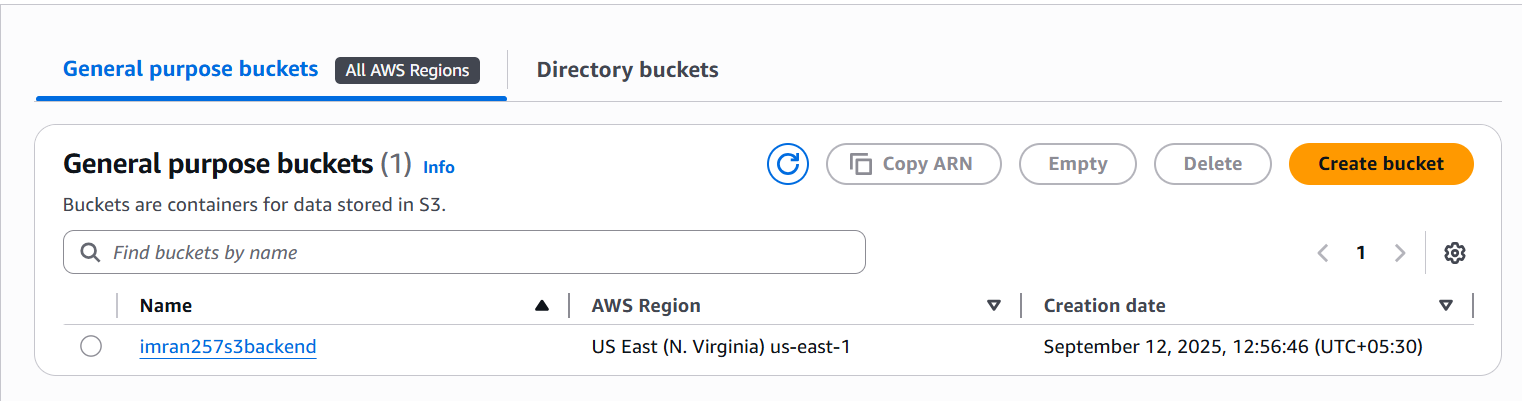
object\_ownership = "ObjectWriter"

}

}

Terraform apply





**Create dynamo db using terraform:**

**dynamodb-terraform-state-lock**

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"

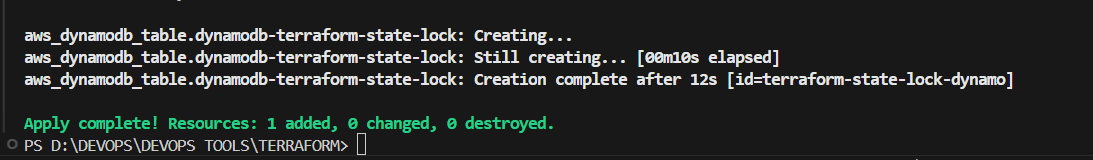
type = "S"

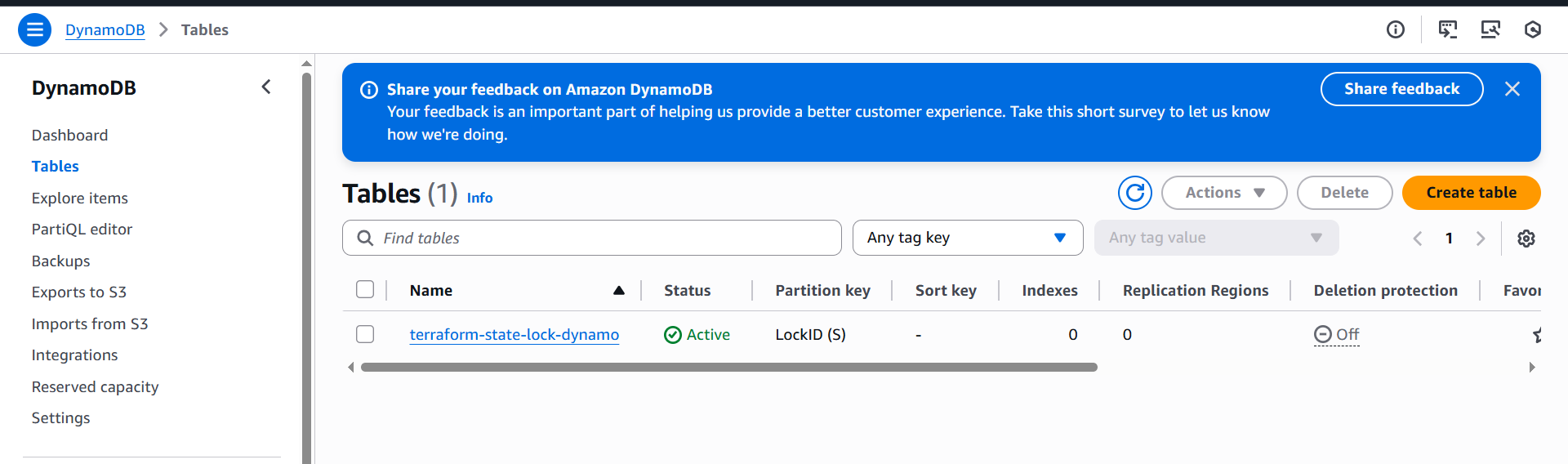
}

}

Terraform init

Terraform apply





**S3 as backend for terraform.tfstate file:**

terraform {

backend "s3" {

bucket = "imran1234s3backend"

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

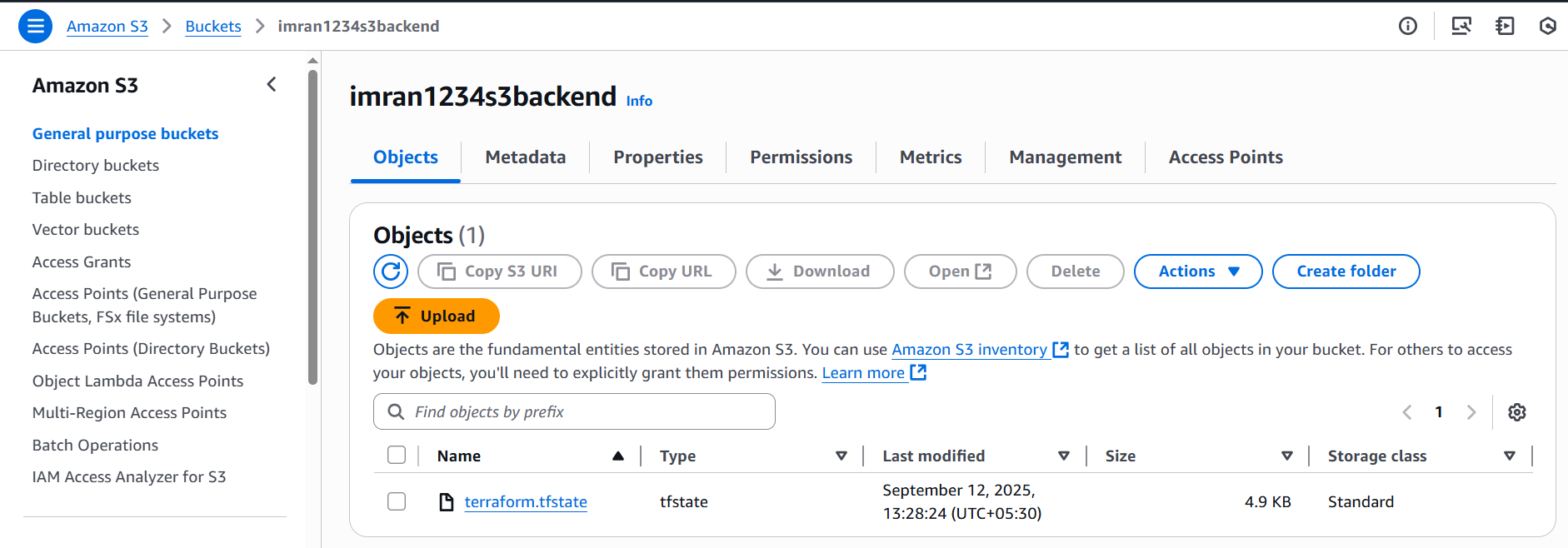
key = "terraform.tfstate"

region = "us-east-1"

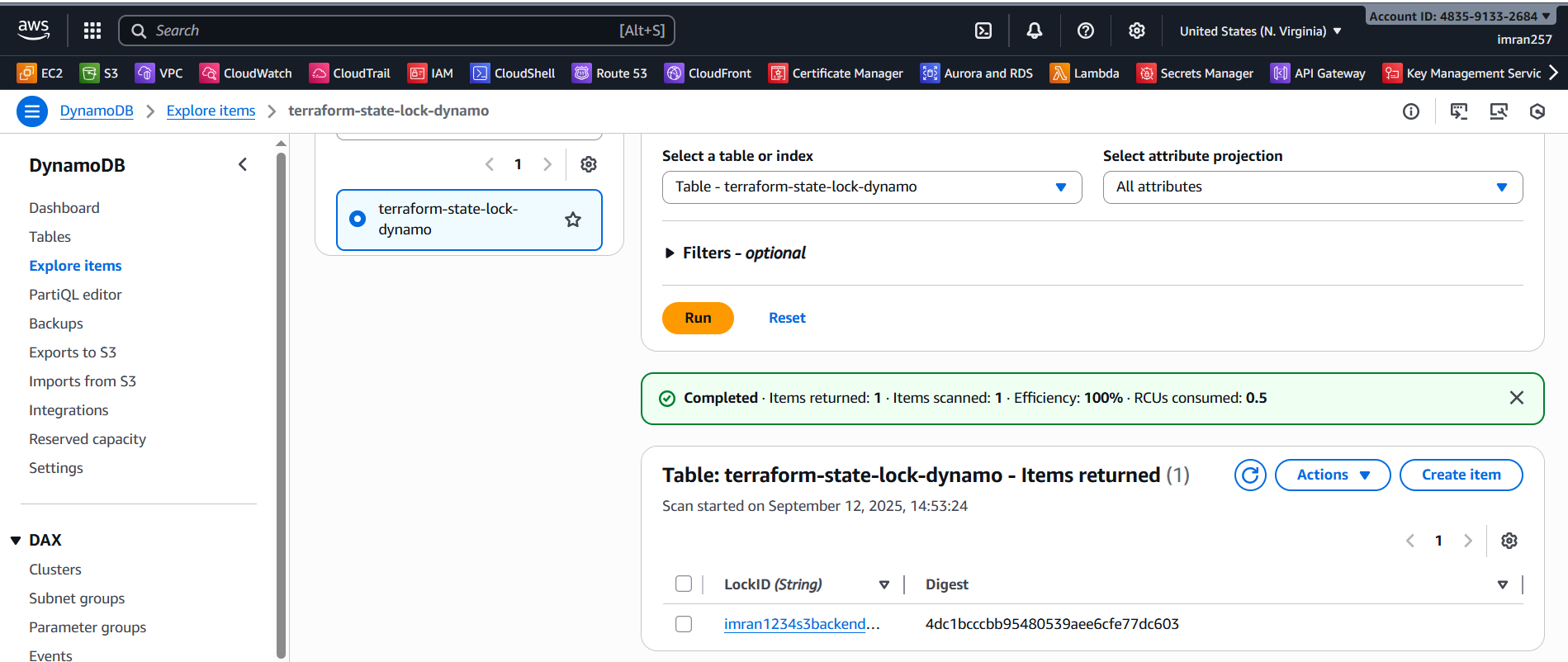
}

}

Statefile created in S3



Created Lock ID



I added one provider

resource "local\_file" "name" {

filename = "pets.txt"

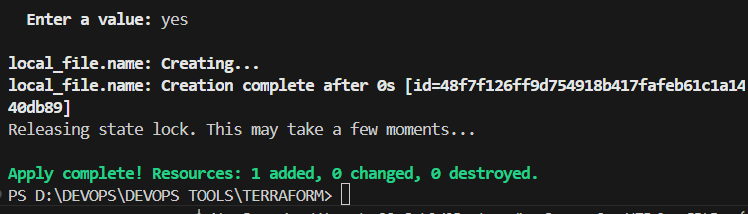
content = "i like cats"

}

Terraform init

Terraform apply

Now statefile is not created. Its directly updated in S3 during execution state lock is active.



TERRAFORM PROVISIONERS

Creating EC2 using Local Provisioner

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

ami = "ami-0d85d4f07a62e2969"

instance\_type = "t2.micro"

key\_name = "jenkins"

tags = {

Name = "Test-server"

}

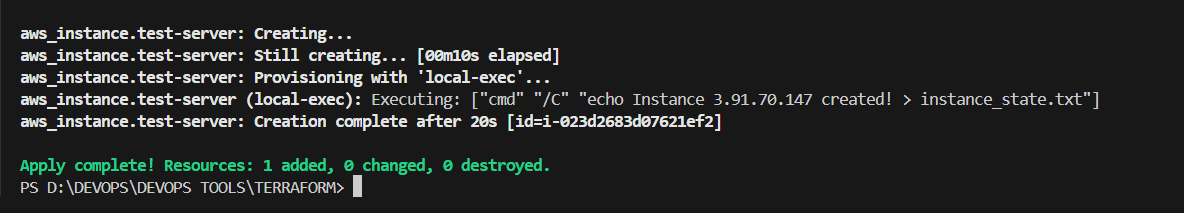
provisioner "local-exec" {

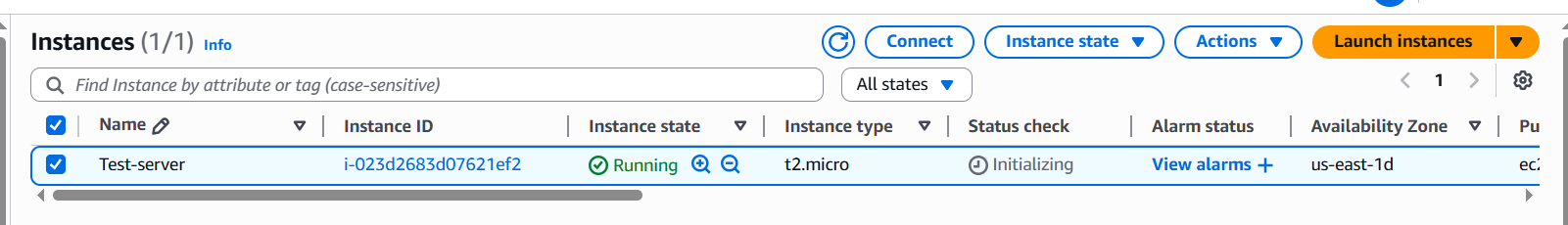
command = "echo Instance ${aws\_instance.test-server.public\_ip} created! > instance\_state.txt"

}

}

Terraform apply message shown in local machine.





Terraform provisioner behaviours:

when=destroy

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

ami = "ami-0d85d4f07a62e2969"

instance\_type = "t2.micro"

key\_name = "jenkins"

tags = {

Name = "Test-server"

}

provisioner "local-exec" {

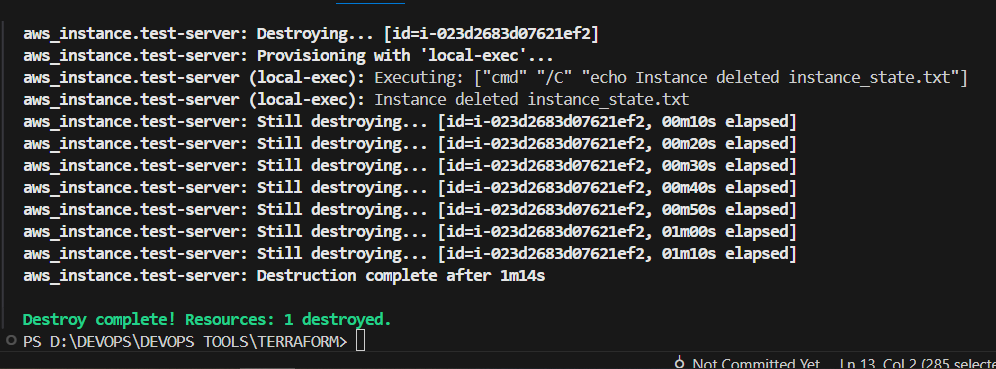
when = destroy

command = "echo Instance deleted instance\_state.txt"

}

}

destroyed



**on\_failure = fail**

If the script failed then terraform apply command will also throw error.

if we want the resource to be completed if the script is failed then we can use on\_failure module.

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

**ami = "ami-005f9685cb30f234b"**

**instance\_type = "t2.micro"**

**key\_name = "linux-01"**

**tags = {**

**Name = "Test-server"**

**}**

**provisioner "local-exec" {**

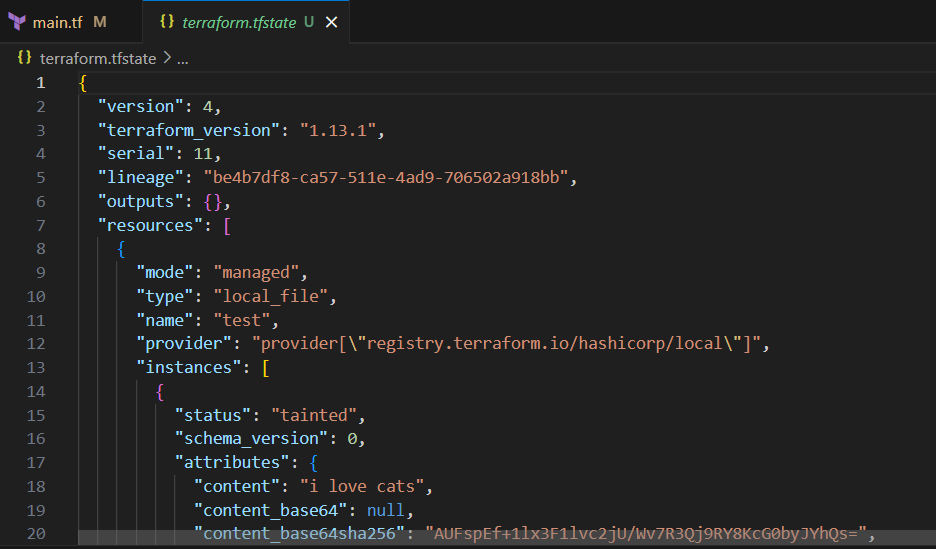
**on\_failure = fail**

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

**}**

**}**

**Terraform taint and untaint:**



resource "local\_file" "test" {

filename = "pets.txt"

content="i love cats"

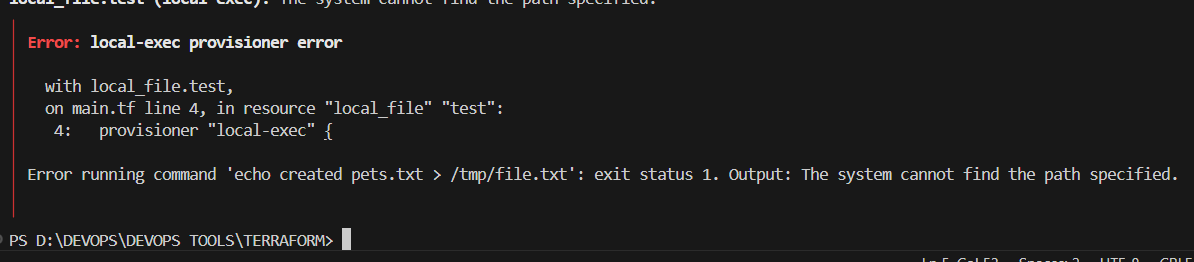
provisioner "local-exec" {

command = "echo created pets.txt > /tmp/file.txt"

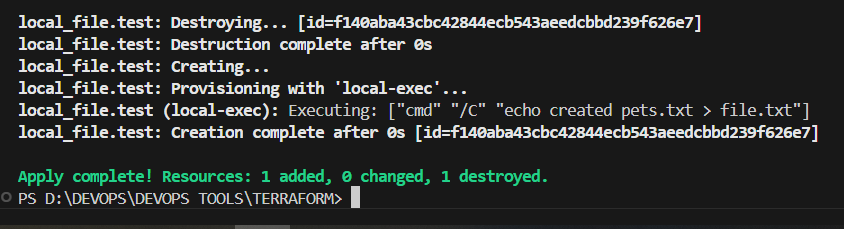
}

}

Terraform apply

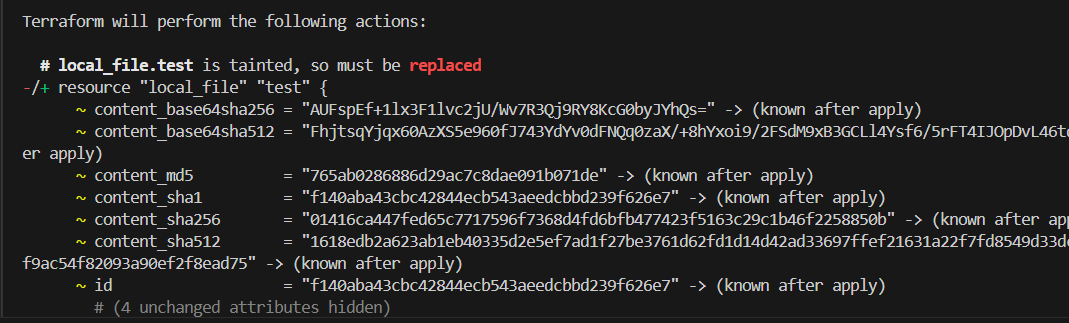


terraform=apply

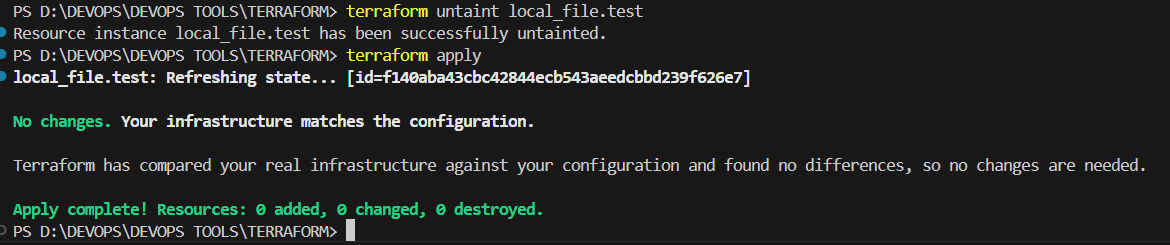


**terraform taint local\_file.test**

**It will keep status as tainted in our state file**

****

**terraform untaint local\_file.test**

****

DEBUGGING

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

but still if we want to dig deeper then we need to export a varibale

export TF\_LOG=TRACE

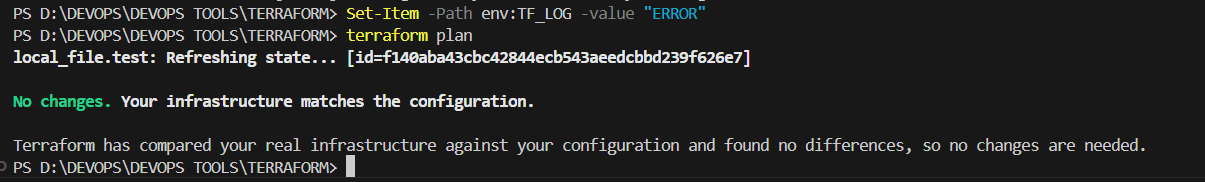
Set-Item -Path env:TF\_LOG -value "TRACE"

Terraform provides 5 levels of logs:

1) INFO

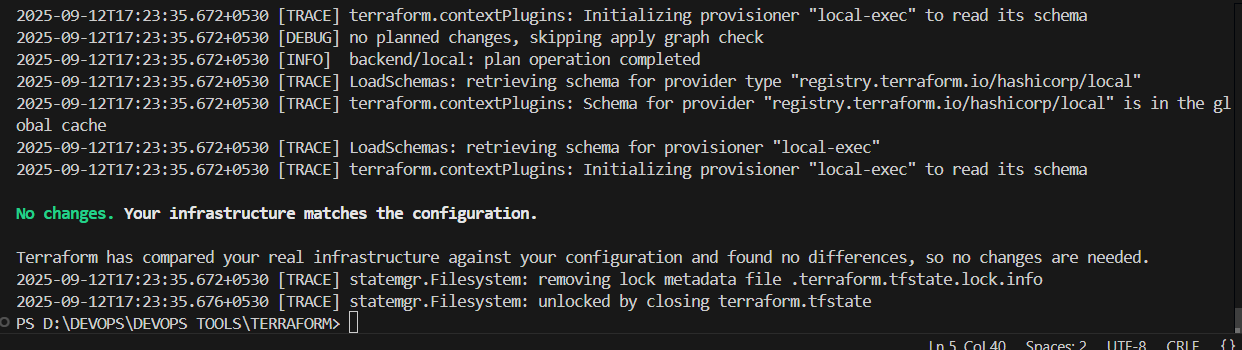
2) WARNING

3) ERROR



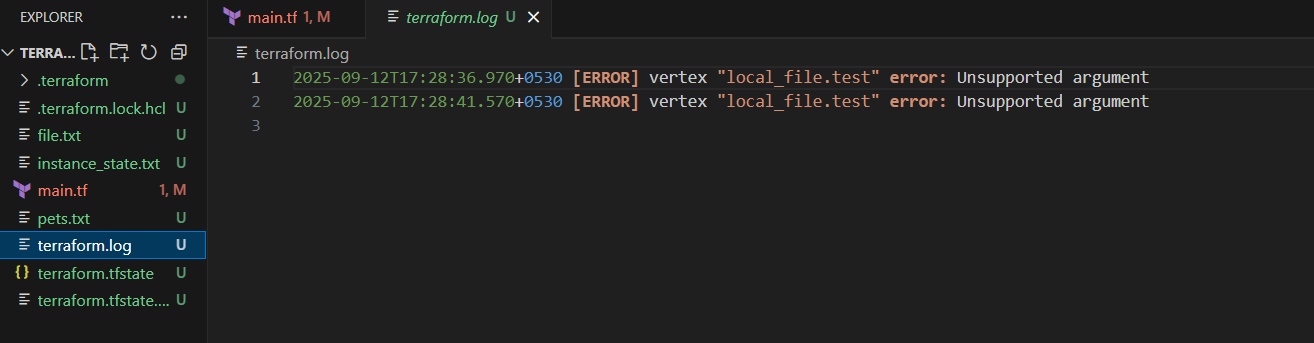
4) DEBUG

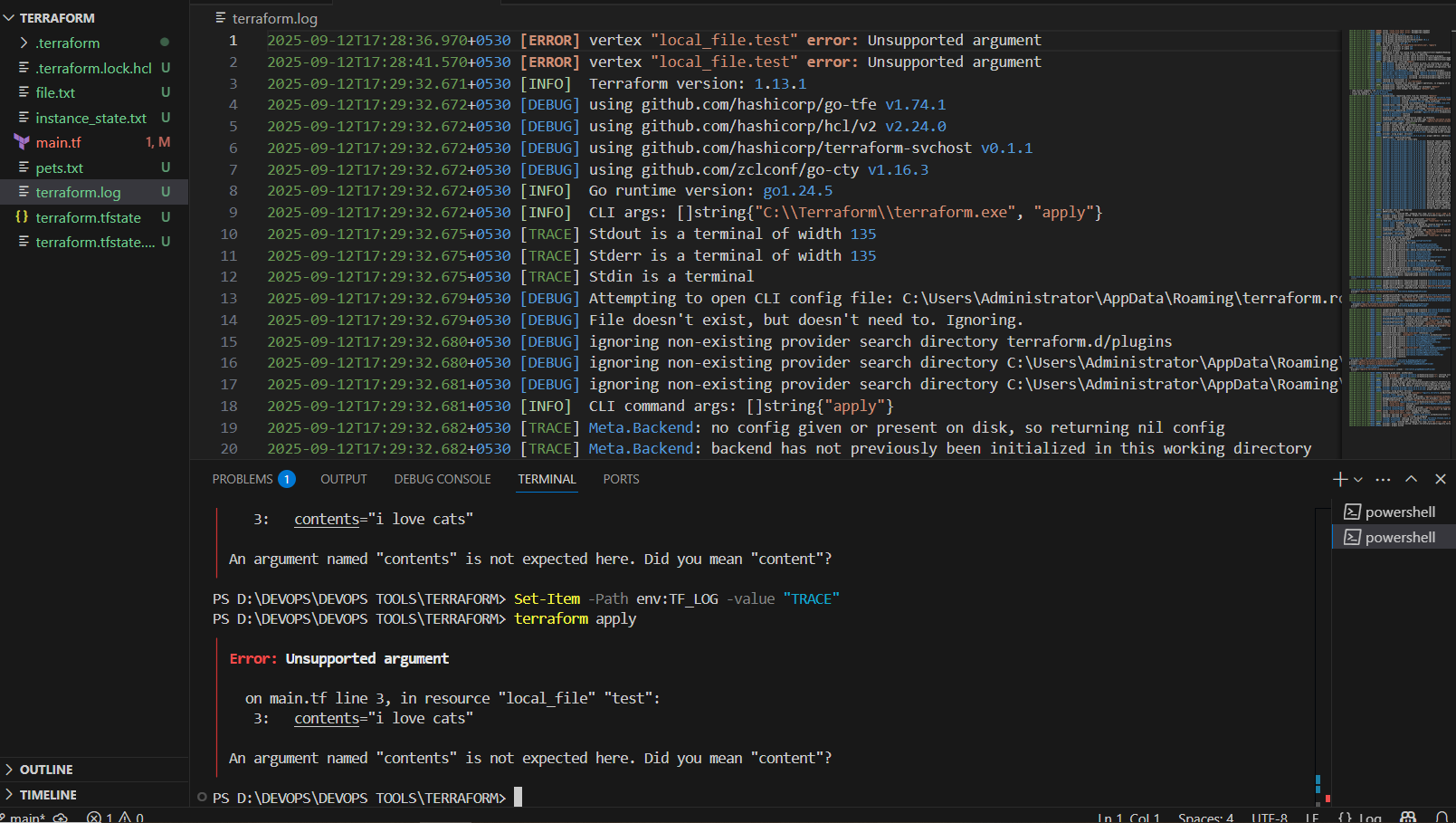
5) TRACE



**Store the logs permanently**

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





**Remove LOG file**

**Remove-Item Env:TF\_LOG\_PATH**

**Remove-Item Env:TF\_LOG**

**Terraform IMPORT**

**To import need to write the configuration block.**

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

**ami = "ami-0d85d4f07a62e2969"**

**instance\_type = "t2.micro"**

**key\_name = "jenkins"**

**}**

**resource "aws\_instance" "manual" {**

**ami = "ami-0d85d4f07a62e2969"**

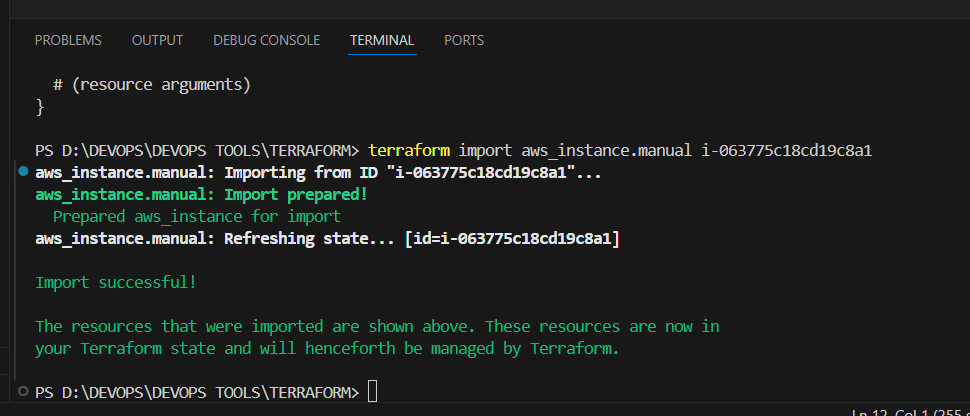
**instance\_type = "t2.micro"**

**key\_name = "jenkins"**

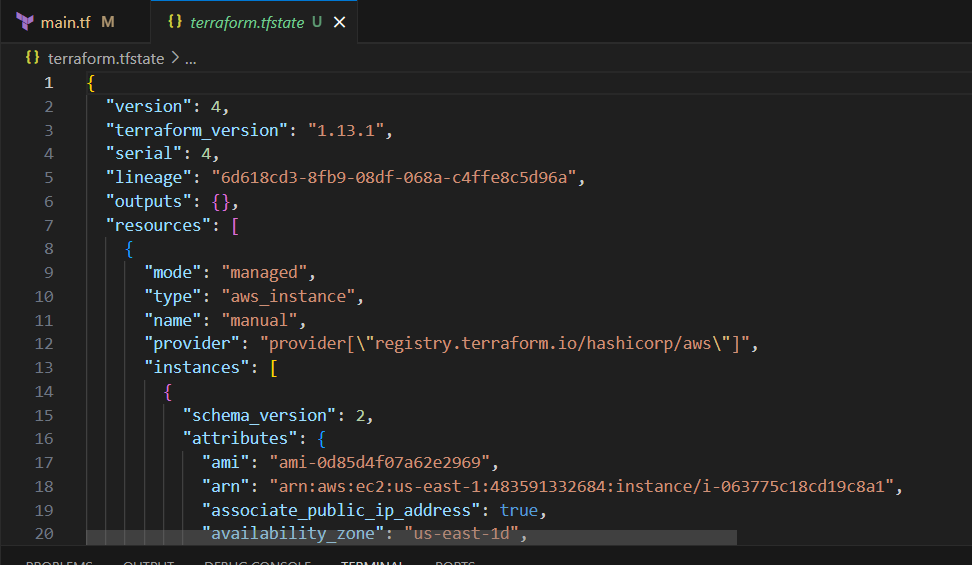
**}**

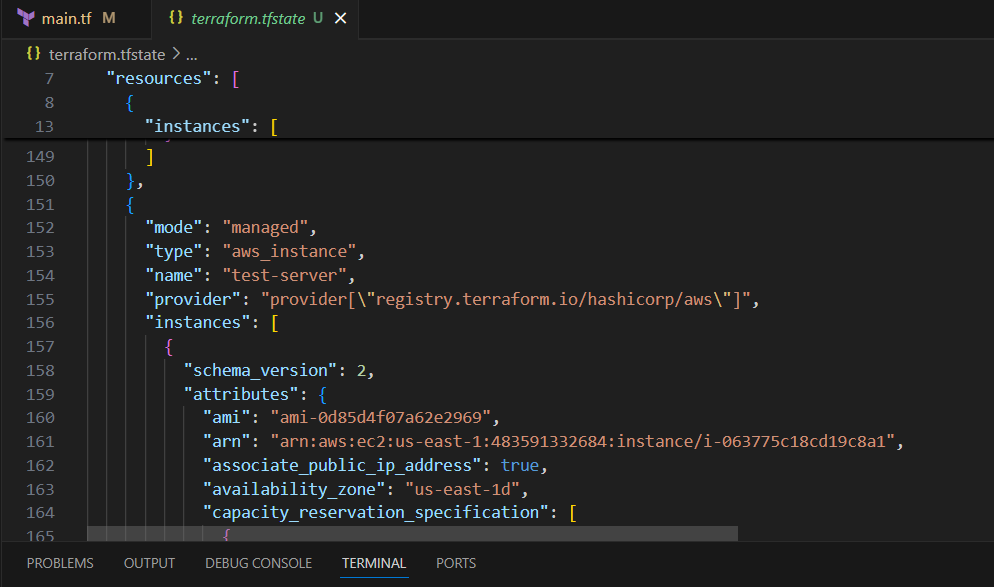
**terraform import aws\_instance.manual i-063775c18cd19c8a1**

**(last one is instance-id)**

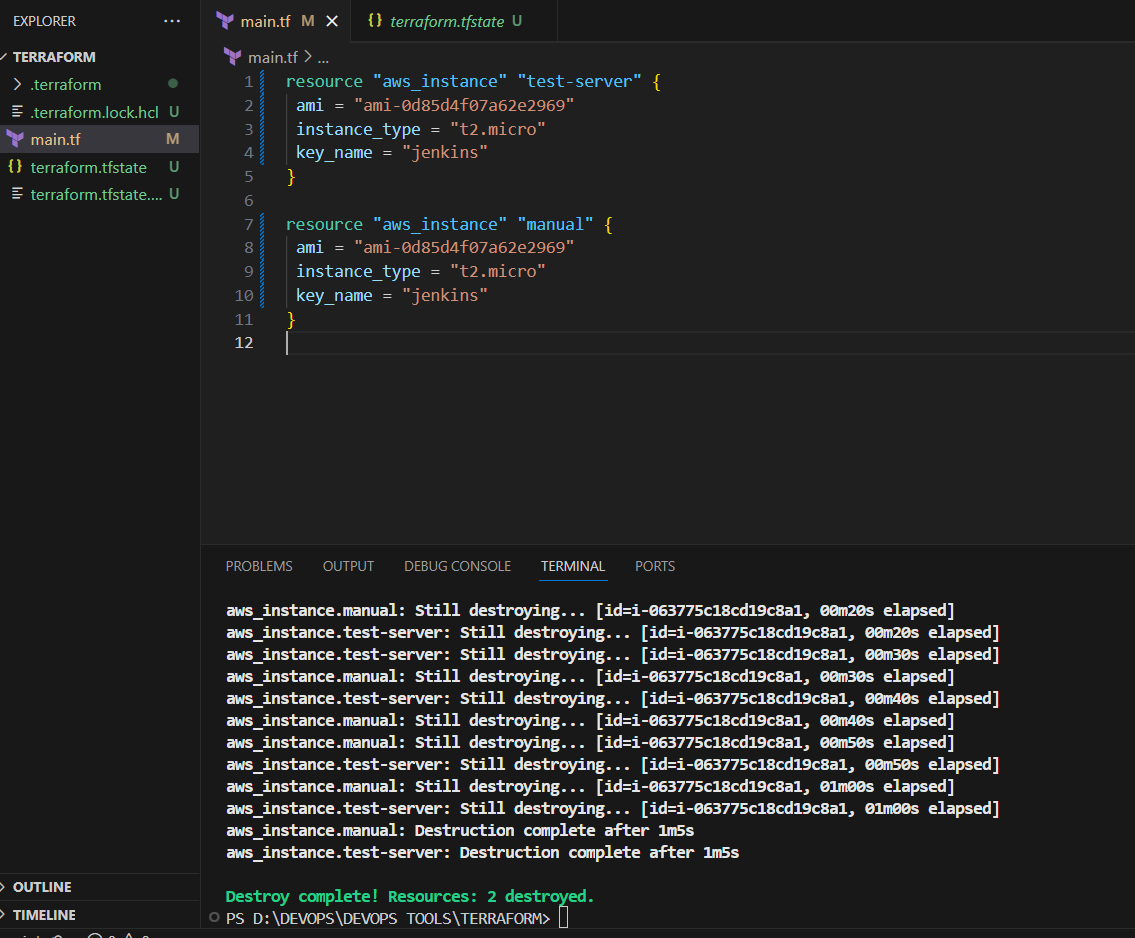
****

**In statefile**

****

****

**Terraform destroy**

****

**3. Create one EC2 instance with httpd installed using a Terraform script.**

[**main.tf**](http://main.tf)

**resource "aws\_instance" "web\_server" {**

**ami = "ami-0d85d4f07a62e2969" # Amazon Linux 2 (check region!)**

**instance\_type = "t2.micro"**

**key\_name = "jenkins" # Your key pair**

**tags = {**

**Name = "httpd-ec2"**

**}**

**user\_data = <<-EOF**

**#!/bin/bash**

**yum update -y**

**yum install -y httpd**

**systemctl enable httpd**

**systemctl start httpd**

**echo "<h1>Welcome to Terraform Apache Server</h1>" > /var/www/html/index.html**

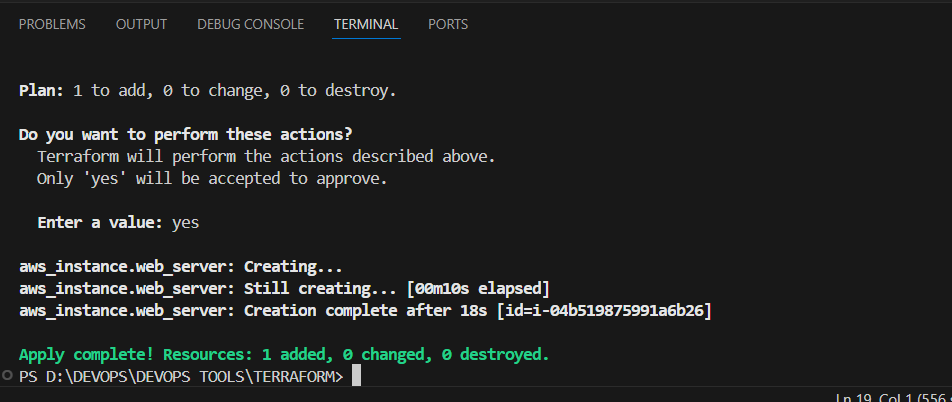
**EOF**

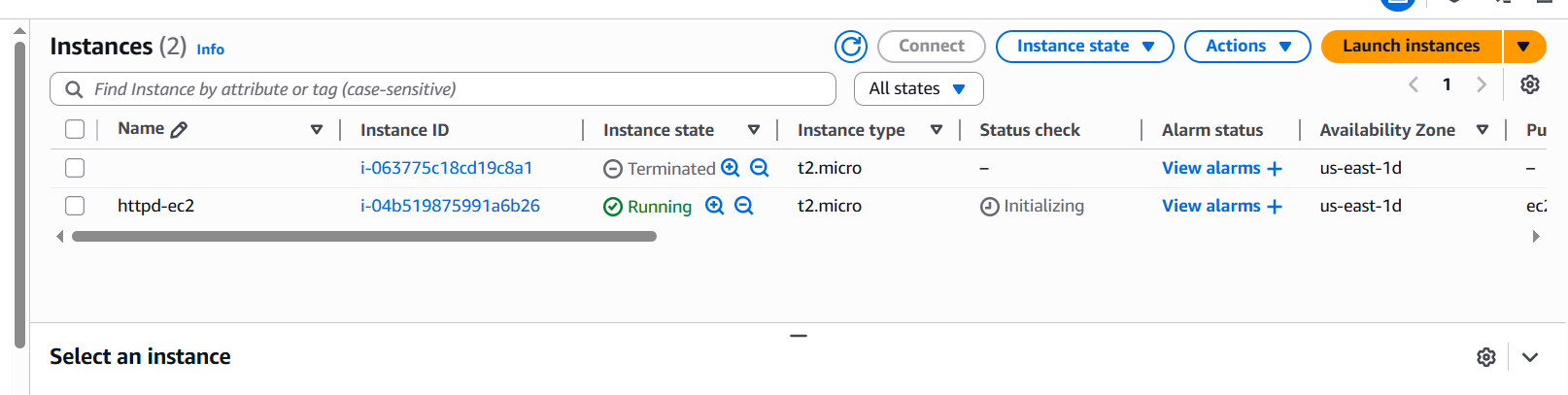
**}**

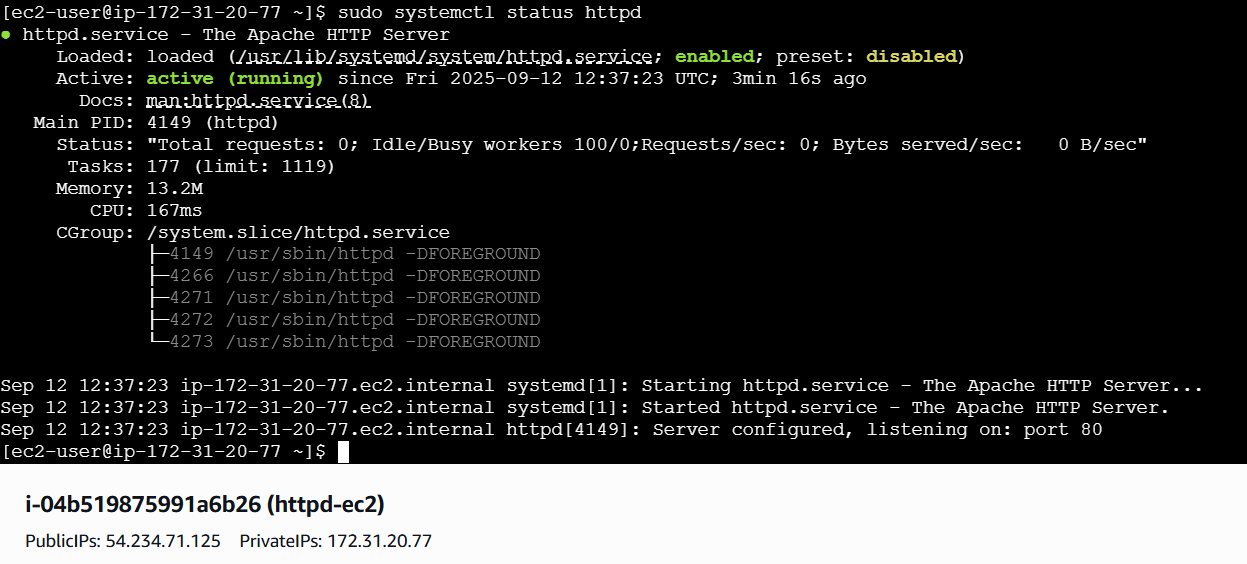
**Terraform init**

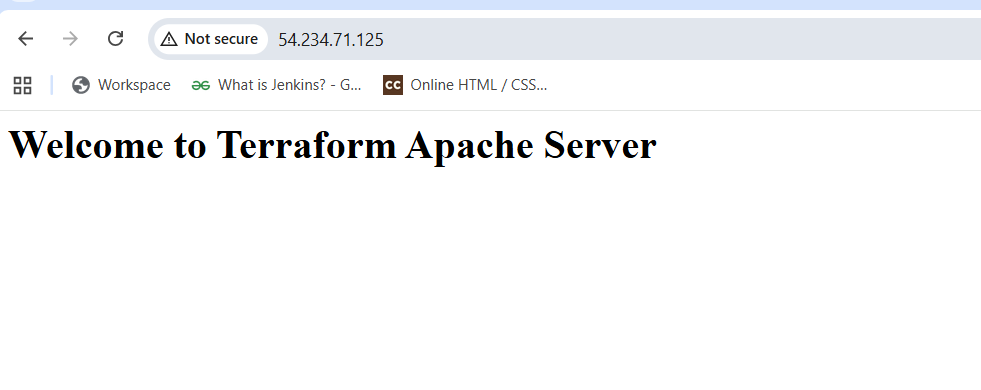
**Terraform plan**

**Terraform apply**

****

****

****

****

**4. Set up S3 as backend for task 3.**

[**main.tf**](http://main.tf)

**resource "aws\_s3\_bucket" "terraform\_backend" {**

**bucket = "terraform-backend-imran-2025" # must be globally unique**

**}**

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

**bucket = aws\_s3\_bucket.terraform\_backend.id**

**versioning\_configuration {**

**status = "Enabled"**

**}**

**}**

**resource "aws\_s3\_bucket\_public\_access\_block" "block" {**

**bucket = aws\_s3\_bucket.terraform\_backend.id**

**block\_public\_acls = true**

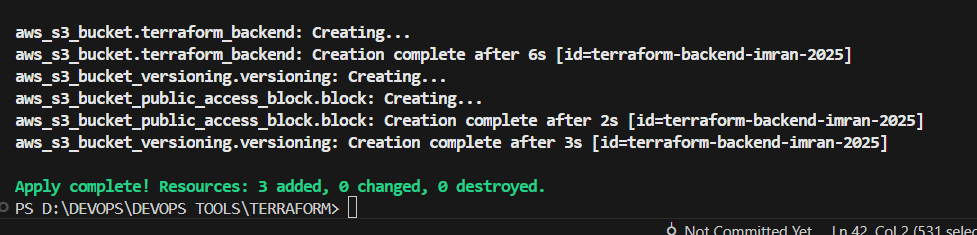
**block\_public\_policy = true**

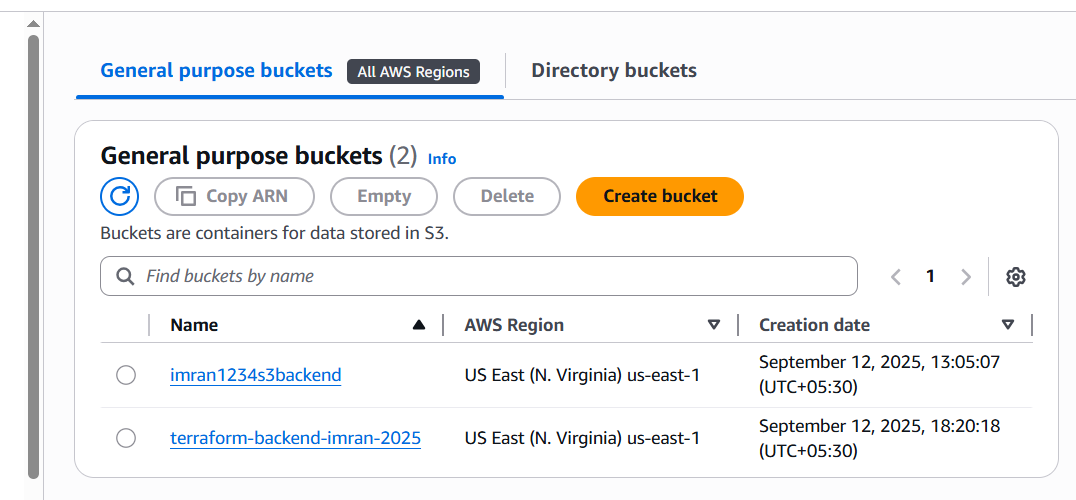
**ignore\_public\_acls = true**

**restrict\_public\_buckets = true**

**}**

**Terraform apply**

****

****

**5. Set up DynamoDB locking for task 3.**

**Add some more configurations in** [**main.tf**](http://main.tf)

**resource "aws\_instance" "web\_server" {**

**ami = "ami-0d85d4f07a62e2969" # Amazon Linux 2 (check region!)**

**instance\_type = "t2.micro"**

**key\_name = "jenkins" # Your key pair**

**tags = {**

**Name = "httpd-ec2"**

**}**

**user\_data = <<-EOF**

**#!/bin/bash**

**yum update -y**

**yum install -y httpd**

**systemctl enable httpd**

**systemctl start httpd**

**echo "<h1>Welcome to Terraform Apache Server</h1>" > /var/www/html/index.html**

**EOF**

**}**

**resource "aws\_s3\_bucket" "terraform\_backend" {**

**bucket = "terraform-backend-imran-2025" # must be globally unique**

**}**

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

**bucket = aws\_s3\_bucket.terraform\_backend.id**

**versioning\_configuration {**

**status = "Enabled"**

**}**

**}**

**resource "aws\_s3\_bucket\_public\_access\_block" "block" {**

**bucket = aws\_s3\_bucket.terraform\_backend.id**

**block\_public\_acls = true**

**block\_public\_policy = true**

**ignore\_public\_acls = true**

**restrict\_public\_buckets = true**

**}**

**resource "aws\_dynamodb\_table" "terraform\_locks" {**

**name = "terraform-state-lock-dynamo"**

**hash\_key = "LockID"**

**read\_capacity = 5**

**write\_capacity = 5**

**attribute {**

**name = "LockID"**

**type = "S"**

**}**

**}**

**terraform {**

**backend "s3" {**

**bucket = "terraform-backend-imran-2025" # same as above**

**key = "ec2-httpd/terraform.tfstate"**

**region = "us-east-1"**

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

**encrypt = true**

**}**

**}**

## **⚡ Workflow to Apply**

1. **Deploy backend resources (S3 + DynamoDB) first**
   * **Temporarily comment out the terraform { backend "s3" ... } block**
   * **Run:**

**terraform init**

**terraform apply**

**This creates the S3 bucket + DynamoDB table.**

**2. Enable backend**

* **Uncomment the backend block.**
* **Run:**

**terraform init -reconfigure**

**This migrates state to S3 with DynamoDB locking.**

**3. Deploy EC2 instance**

**terraform apply**

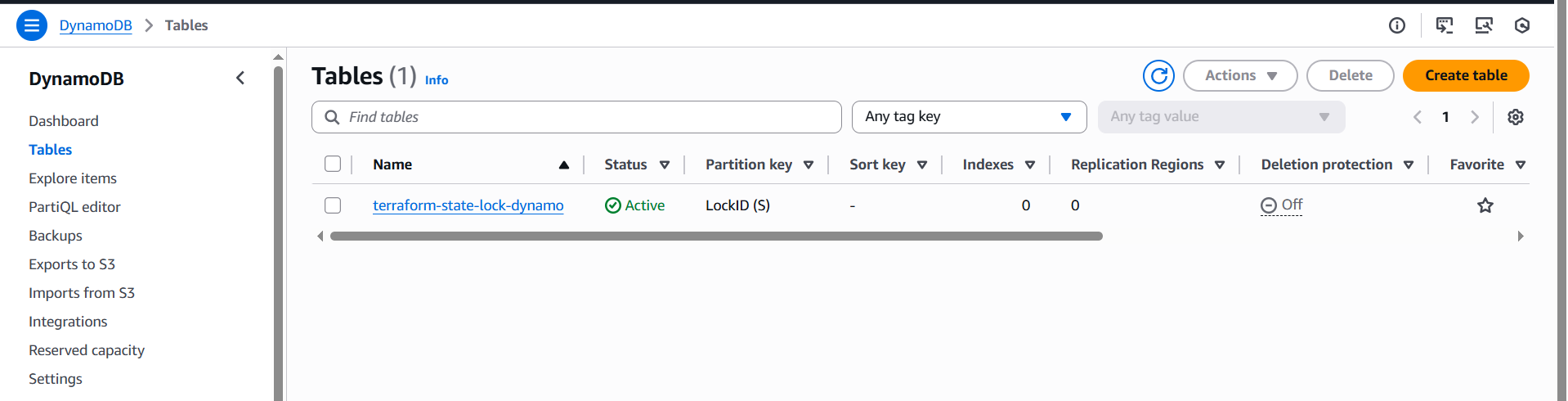
**✅ End result:**

* **An EC2 instance with Apache running.**
* **Terraform state stored in S3, versioned + encrypted.**
* **DynamoDB used for locking to avoid race conditions.**

**DynamoDB Locking done**

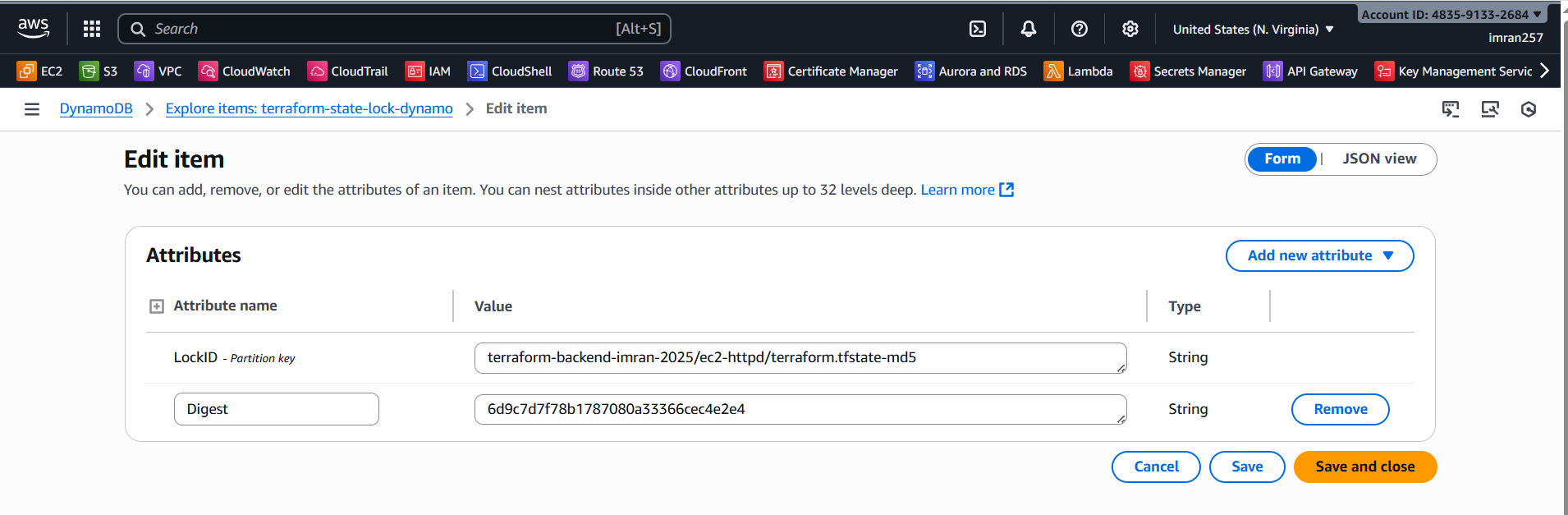
**State file created.**

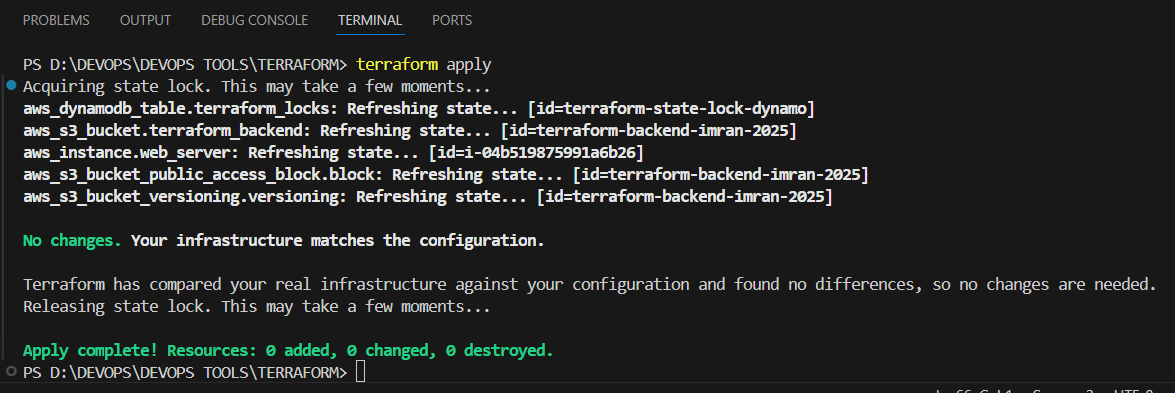
[**terraform-state-lock-dynamo**](https://us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#table?name=terraform-state-lock-dynamo)

****

**Configure Remote Backend done**

**terraform-backend-imran-2025/ec2-httpd/terraform.tfstate-md5**

****

****

## **🔹 What You Did**

1. **Created an EC2 instance**
   * **Amazon Linux 2 instance**
   * **Apache (httpd) installed and started**
   * **Accessible via public IP with a test page (<h1>Welcome to Terraform Apache Server</h1>)**
2. **Created an S3 bucket (terraform-backend-imran-2025)**
   * **Stores your Terraform state file (terraform.tfstate)**
   * **Versioning enabled → keeps history of state changes**
   * **Public access blocked (secure)**
3. **Created a DynamoDB table (terraform-state-lock-dynamo)**
   * **Used by Terraform to lock state so that if two people run terraform apply at the same time, only one will proceed.**
4. **Configured Terraform backend**
   * **State file is no longer local (terraform.tfstate in your project folder).**
   * **It now lives in S3 with locking in DynamoDB.**

## **🔹 Why This Is Important**

* **Collaboration → If multiple team members use the same infra code, they share one state in S3.**
* **Safety → DynamoDB prevents race conditions (two applies at once).**
* **Resilience → If your local machine is lost, the state still lives in S3.**
* **Auditability → With S3 versioning, you can roll back to a previous state file.**

**This is the standard enterprise setup for Terraform.**

**Verify Backend**

* **Check your AWS S3 console:**
  + **Go to the terraform-backend-imran-2025 bucket.**
  + **Navigate to ec2-httpd/terraform.tfstate.**
  + **Confirm that the state file is stored there.**
* **Go to DynamoDB console → Tables → terraform-state-lock-dynamo.**
  + **Try running terraform apply twice simultaneously → you’ll see a lock entry.**

**✅ You now have:**

* **Infrastructure (EC2 + Apache)**
* **Secure, shared Terraform state management**