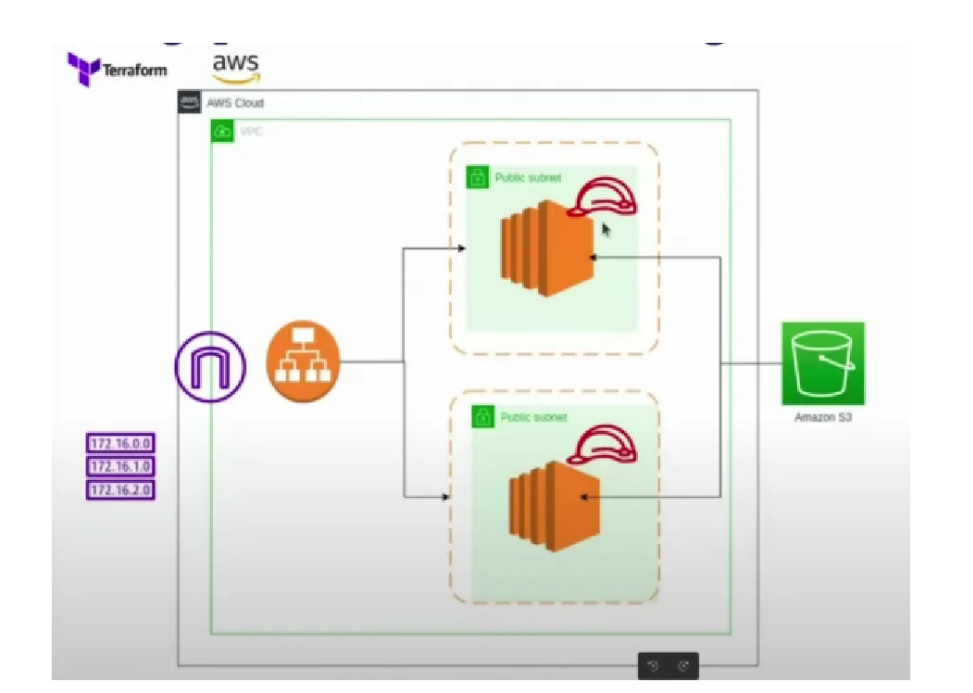
**Setting up infrastructure on AWS using Terraform**



**Introdution**

In this project, we are going to setting up a AWS infrastructure by using terraform.

This setup will includes creating a vpc, public subnets, security group, EC2, s3, application load balancer and target group. We’ll also configure EC2 instance is hosting the web page using http server.

First login to your aws acc.

Launch one instance for communicate with cloud using AWS CLI and terraform CLI.

After launching the instance ssh the instance and install awscli,terraform.

Create a access keys for access to cloud.

configure the terraform version file and provider file.

**Step -1:** setup the resource file configuration. Create the resource.tf file in terraform folder.

In that file first configure the vpc and cidr block.

resource "aws\_vpc" "myvpc" {

    cidr\_block = "10.0.0.0/16"

}

Create two public subnets in two availability\_zones.

resource "aws\_subnet" "sub1" {

  vpc\_id = aws\_vpc.myvpc.id

  cidr\_block = "10.0.1.0/24"

  availability\_zone = "us-east-1a"

  map\_public\_ip\_on\_launch = true

}

resource "aws\_subnet" "sub2" {

  vpc\_id = aws\_vpc.myvpc.id

  cidr\_block = "10.0.2.0/24"

  availability\_zone = "us-east-1b"

  map\_public\_ip\_on\_launch = true

}

Create internet gateway.

resource "aws\_internet\_gateway" "myigw" {

  vpc\_id = aws\_vpc.myvpc.id

}

Create route table and association

resource "aws\_route\_table" "myrt" {

  vpc\_id = aws\_vpc.myvpc.id

  route {

    cidr\_block = "0.0.0.0/0"

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

      }

}

resource "aws\_route\_table\_association" "rtass1" {

 subnet\_id = aws\_subnet.sub1.id

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

}

resource "aws\_route\_table\_association" "rtass2" {

 subnet\_id = aws\_subnet.sub2.id

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

}

Create the security group and inbound rules(ingress) and outbound rules(egress).

resource "aws\_security\_group" "mysg" {

   vpc\_id = aws\_vpc.myvpc.id

   ingress = {

    from\_port = 80

    to\_port =  80

    protocol = "tcp"

    cidr\_block = ["0.0.0.0/0"]

   }

   ingress = {

    from\_port = 22

    to\_port =  22

    protocol = "tcp"

    cidr\_block = ["0.0.0.0/0"]

   }

   egress = {

    from\_port = 0

    to\_port = 0

    protocol = "-1"

    cidr\_block = ["0.0.0.0/0"]

   }

tags = {

     name = "websg"

   }

}

Create s3 bucket to store the date.

resource "aws\_s3\_bucket" "mys3bucket" {

  bucket = "harishterraformproject"

}

Now create the two ec2 instance

resource "aws\_instance" "websev1" {

  ami = "ami-0182f373e66f89c85"

  instance\_type = "t2-micro"

  vpc\_security\_group\_ids = [ aws\_security\_group.mysg.id]

  subnet\_id = aws\_subnet.sub1.id

  user\_data = base64decode(file(userdata.sh))

}

resource "aws\_instance" "websev2" {

  ami = "ami-0182f373e66f89c85"

  instance\_type = "t2-micro"

  vpc\_security\_group\_ids = [ aws\_security\_group.mysg.id]

  subnet\_id = aws\_subnet.sub2.id

  user\_data = base64decode(file(userdata1.sh))

}

In userdata we going to write the what want to create after launch instance if use userdata wil be launched with instances.

Create the load balancer with target group and security group and target group attachment and listeners.

resource "aws\_lb" "mylb" {

  internal = "false"

  load\_balancer\_type = "application"

  security\_groups = [aws\_security\_group.mysg.id]

  subnets = [aws\_subnet.sub1.id,aws\_subnet.sub2.id]

  tags = {

    name = myalb

  }

}

resource "aws\_lb\_target\_group" "mytg" {

  port = 80

  protocol = "HTTP"

  vpc\_id = aws\_vpc.myvpc.id

  health\_check {

    path = "/"

    port = "traffic-port"

  }

}

resource "aws\_lb\_target\_group\_attachment" "tg1" {

   target\_group\_arn = aws\_lb\_target\_group.tg.arn

   target\_id = aws\_instance.websev1.id

   port = 80

}

resource "aws\_lb\_target\_group\_attachment" "tg2" {

   target\_group\_arn = aws\_lb\_target\_group.tg.arn

   target\_id = aws\_instance.websev2.id

   port = 80

}

resource "aws\_lb\_listener" "mylistener" {

    load\_balancer\_arn = aws\_lb.mylb.arn

    port = 80

    protocol = "HTTP"

    default\_action {

    target\_group\_arn = aws\_lb\_target\_group.tg.arn

    type             = "forward"

  }

}

Create the output.tf file it will give the lb dns when you enter terraform apply.

output "loadbalancerdns" {

value = aws\_lb.myalb.dns\_name

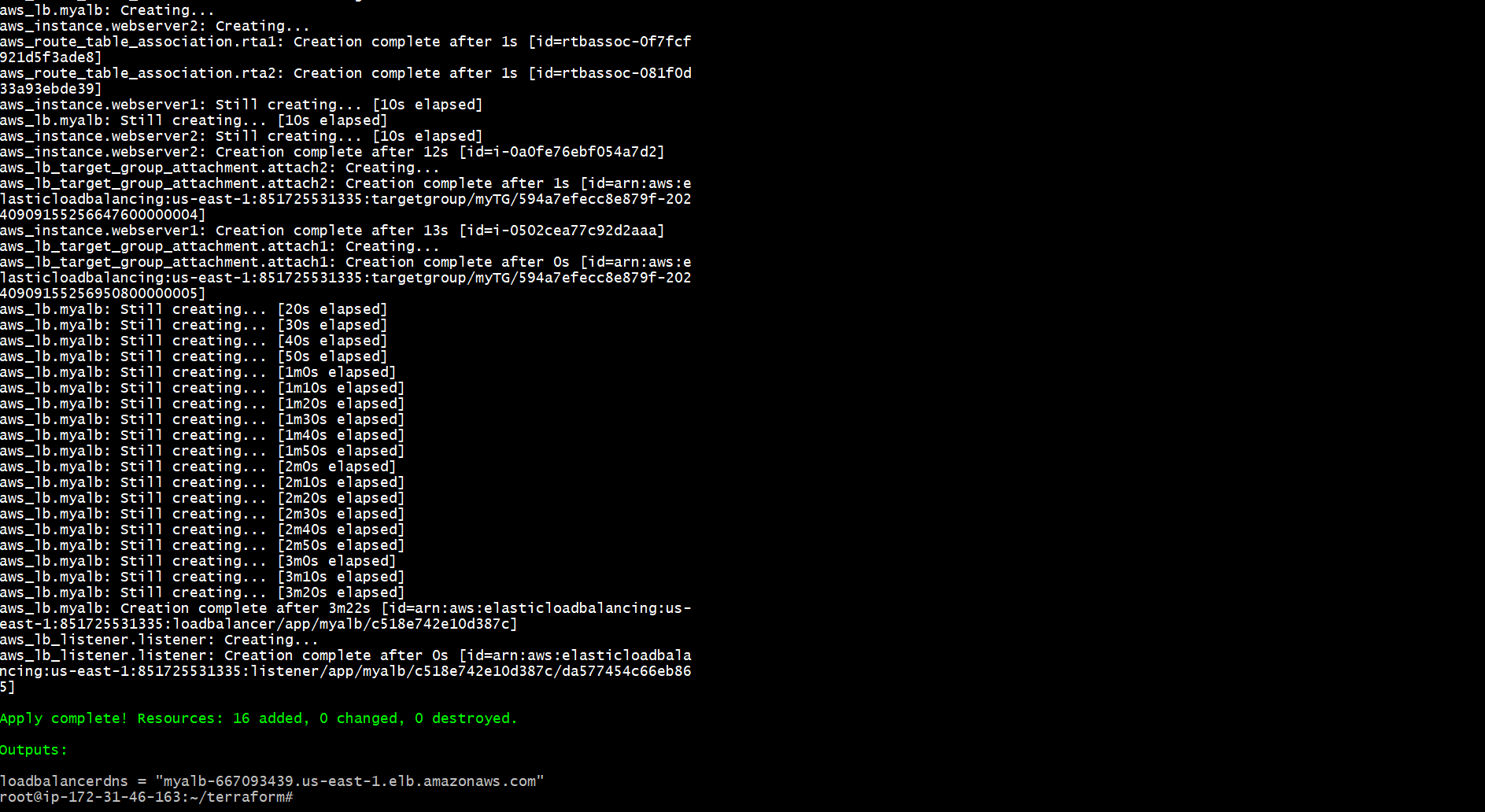
}

Terraform inti

Terraform validate

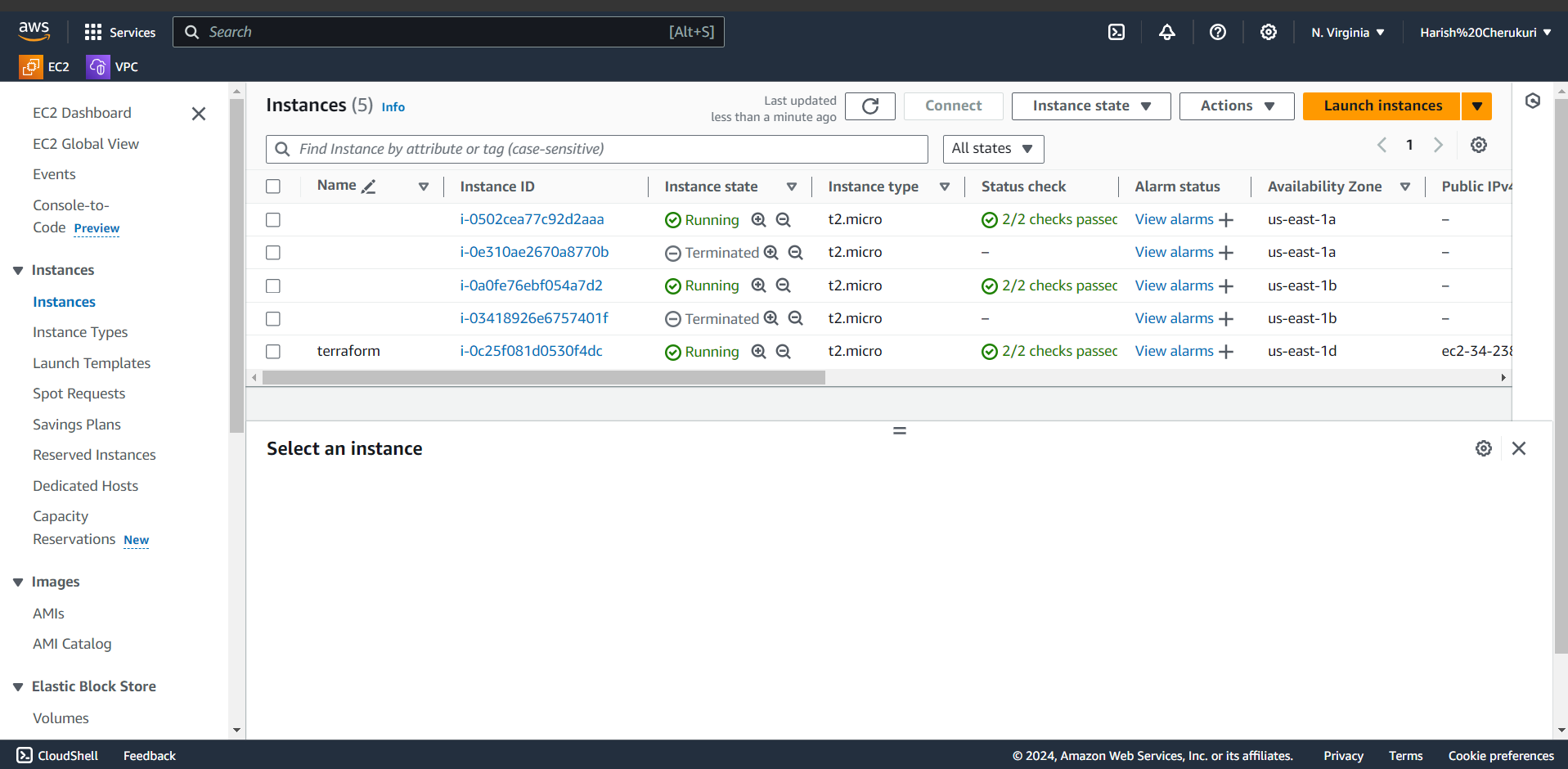
Terraform plan

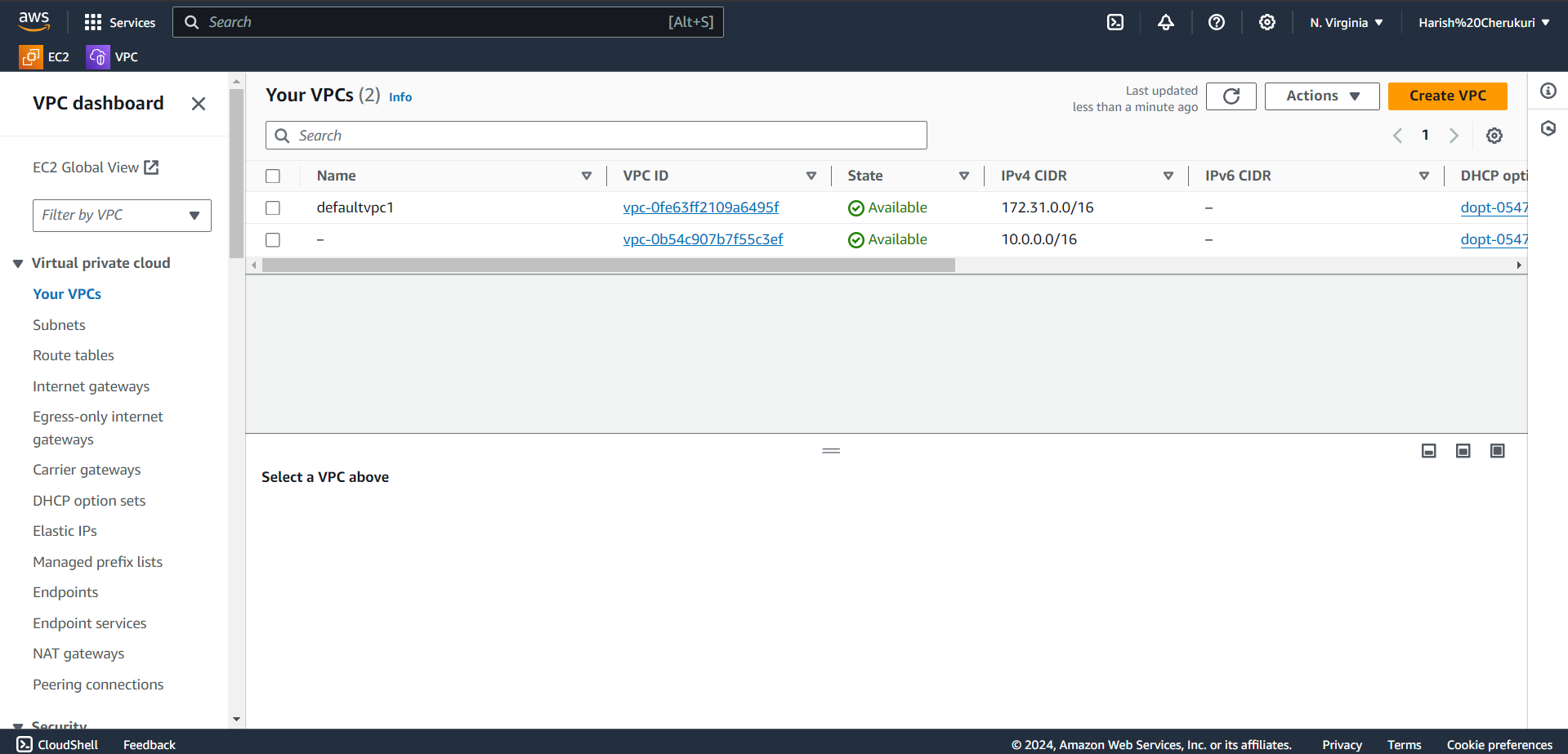
Terraform apply



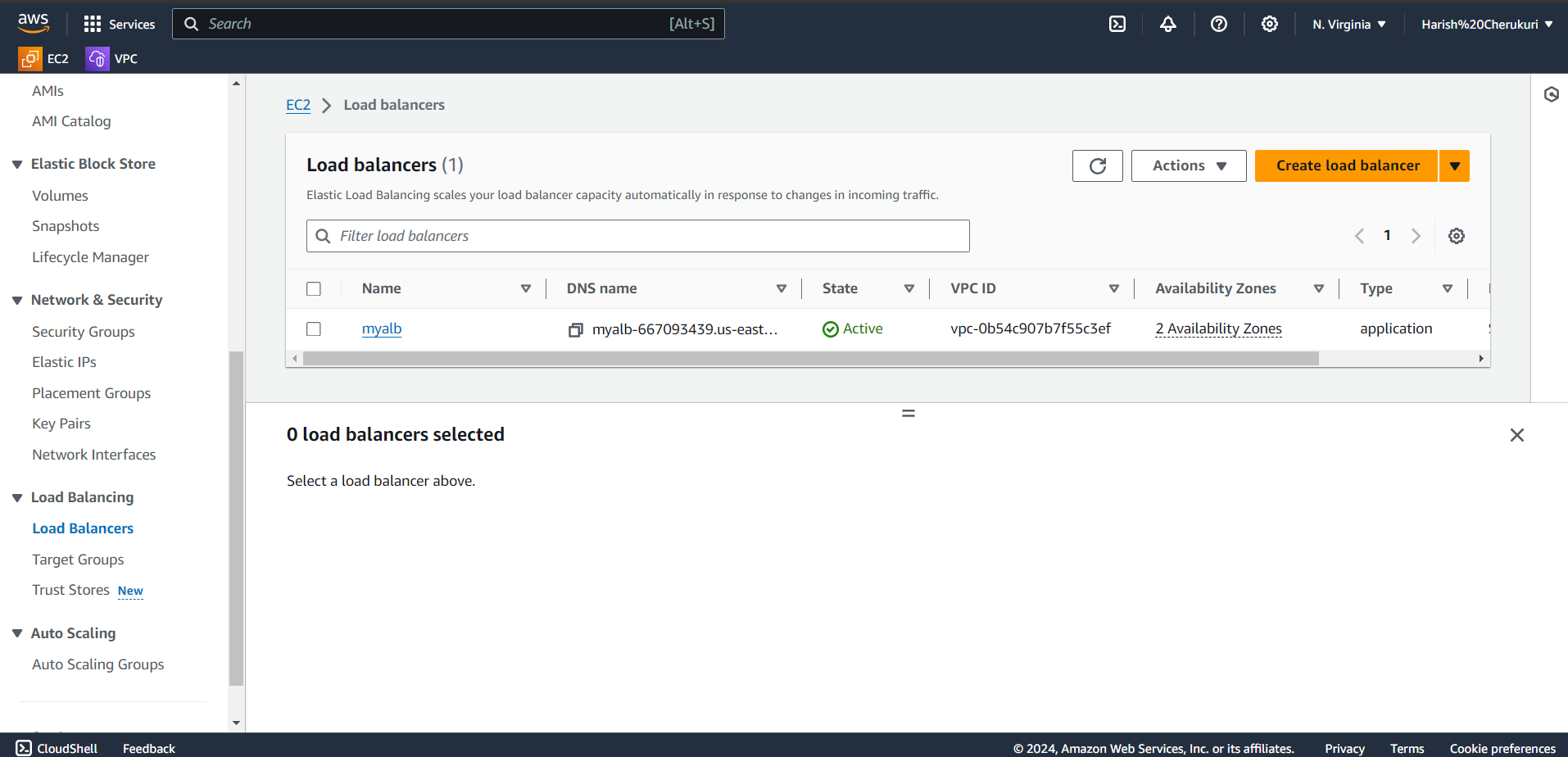
We got the lb dns

And the ec2 ,vpc, subnets , security group, load balancer , target group, also created.

ec2 instance is created.



Vpc also created.

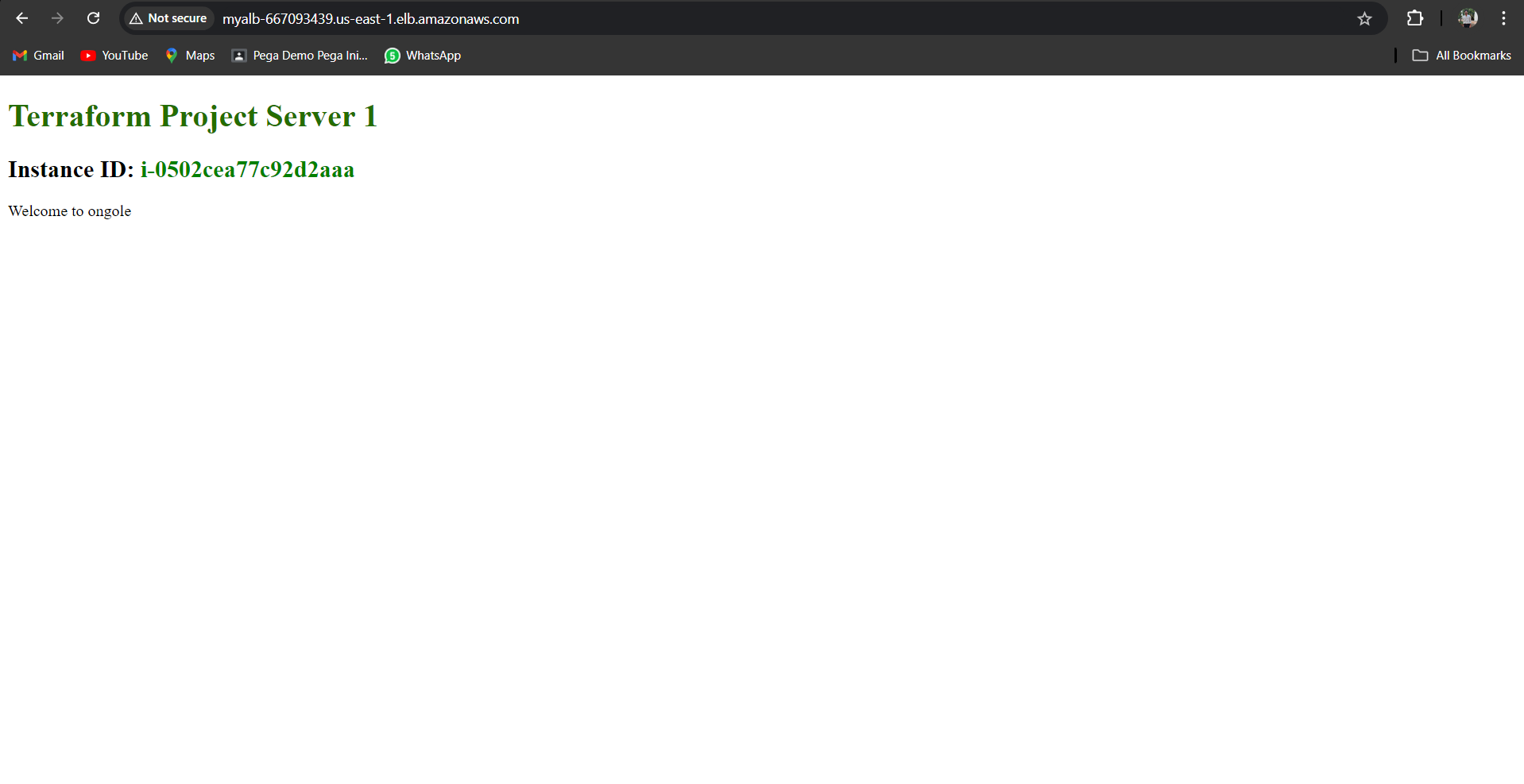


Load balancer also created.

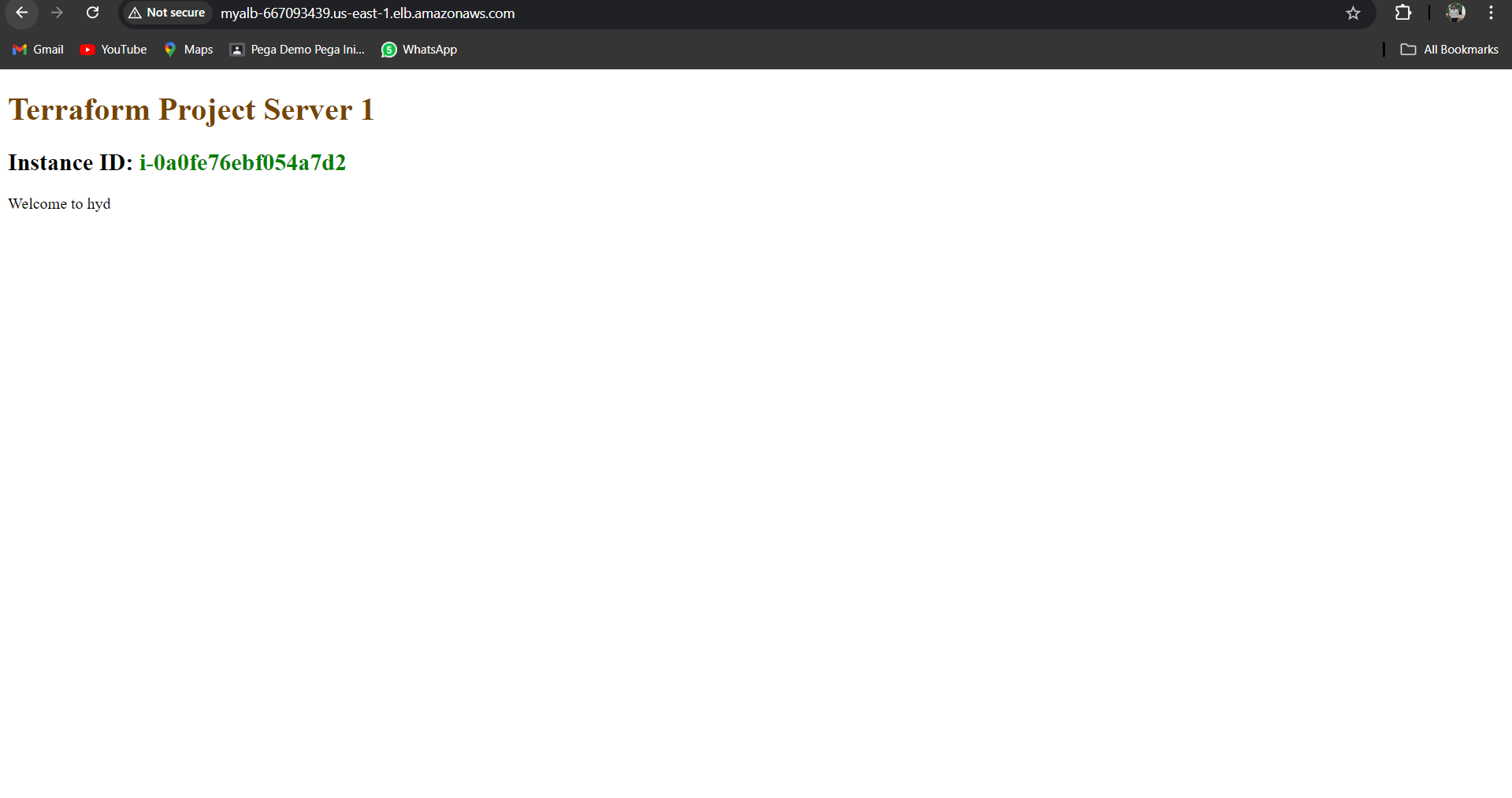
We also got the load balancer dns by configuring the output

myalb-667093439.us-east-1.elb.amazonaws.com

using this dns check lb is working are not.



Refresh the page it will redirect to another web server.



It is working.

Terraform destroy.

By using terraform we can setting up the AWS infrastructure.