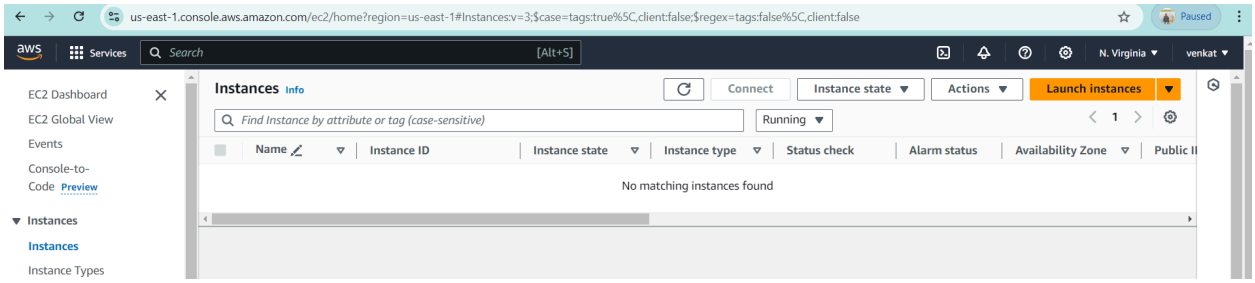
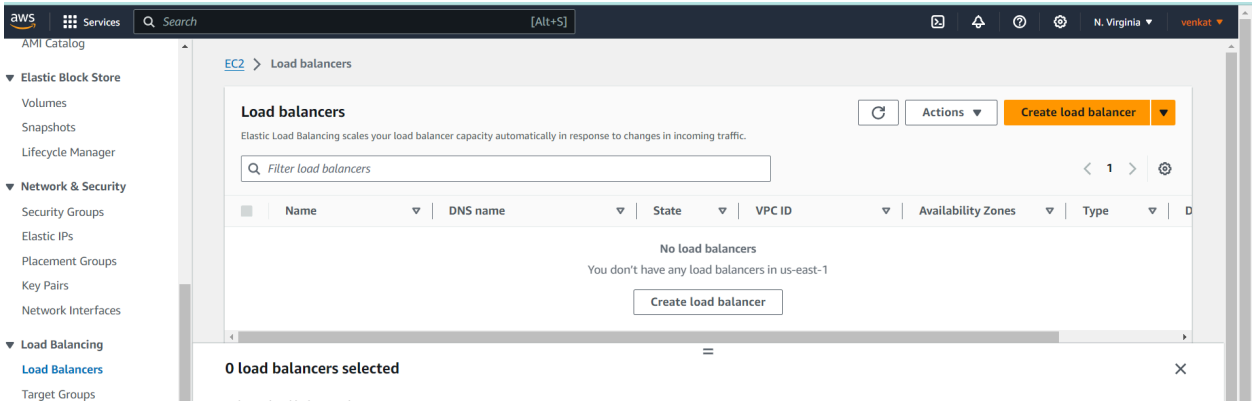
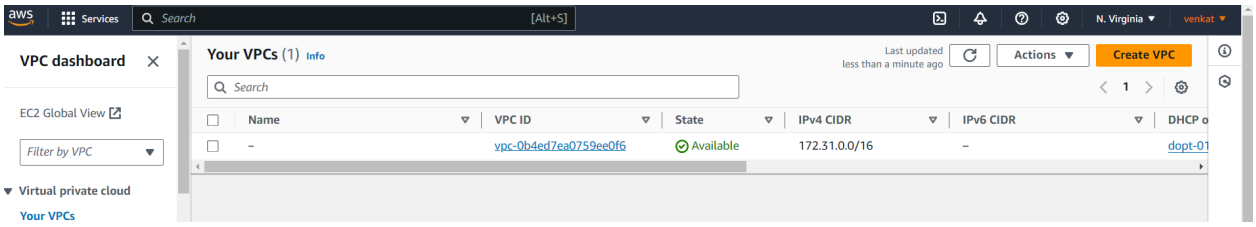
CREATE LOAD BALANCER USING TERRAFORM (us-east-1)

1) Login aws account





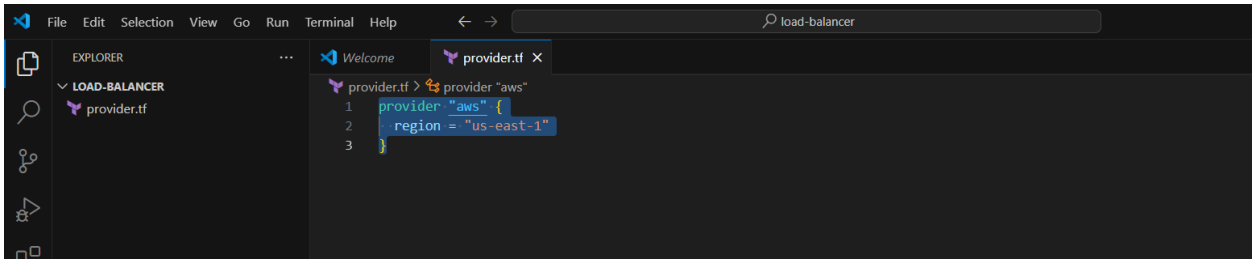


2) Now open visual studio code

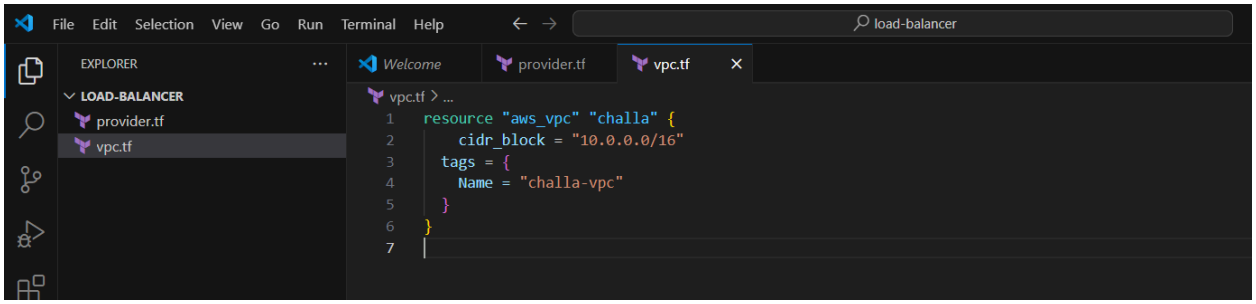
3) Select folder

4) Crete file provider.tf

```
provider "aws" {
  region = "us-east-1"
}
```



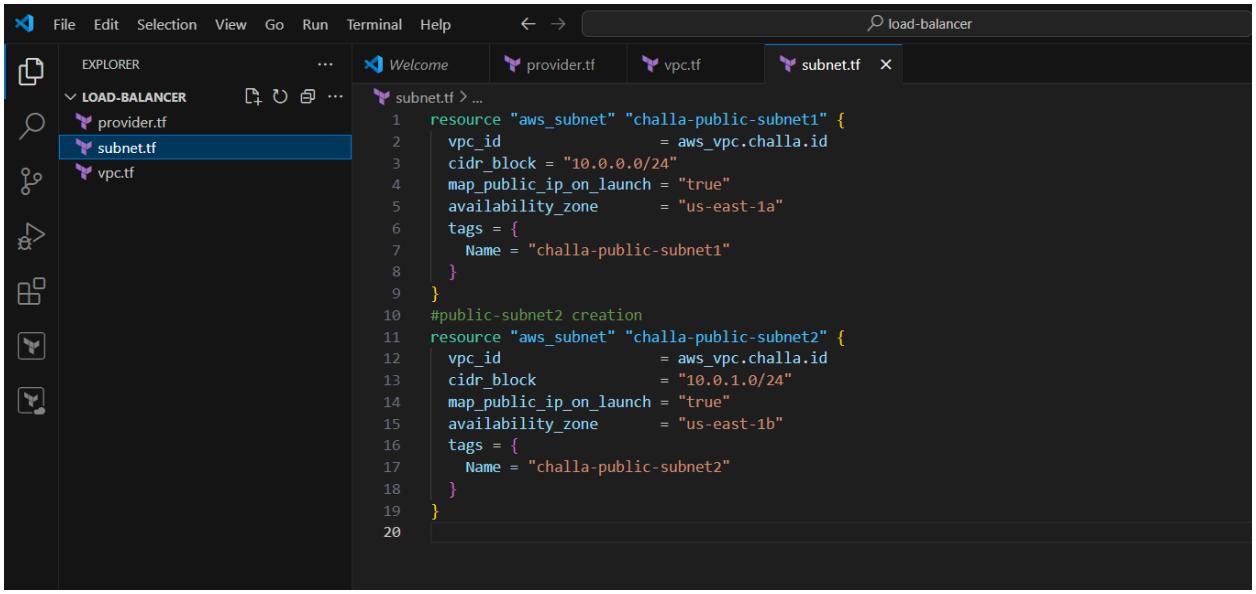
5) Now create vpc.tf




```
resource "aws_vpc" "challa" {
   cidr_block = "10.0.0.0/16"
  tags = {
    Name = "challa-vpc"
  }
}
```

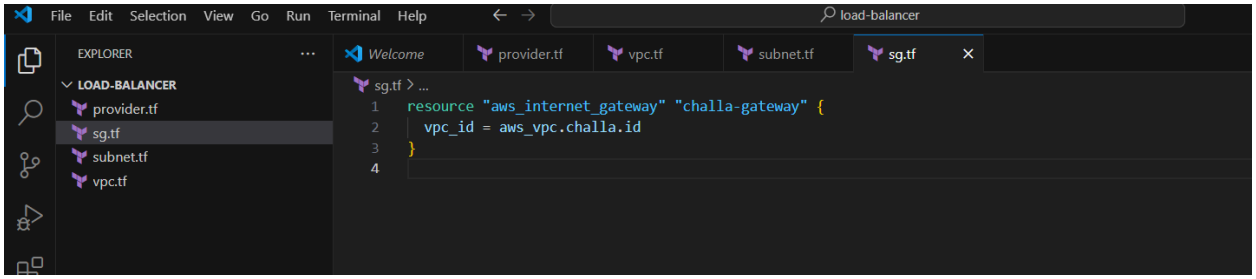
6) Now create subnet.tf

```
resource "aws_subnet" "challa-public-subnet1" {
  vpc_id                  = aws_vpc.challa.id
  cidr_block = "10.0.0.0/24"
  map_public_ip_on_launch = "true"
  availability_zone       = "us-east-1a"
  tags = {
    Name = "challa-public-subnet1"
  }
}
#public-subnet2 creation
resource "aws_subnet" "challa-public-subnet2" {
  vpc_id                  = aws_vpc.challa.id
  cidr_block              = "10.0.1.0/24"
  map_public_ip_on_launch = "true"
  availability_zone       = "us-east-1b"
  tags = {
    Name = "challa-public-subnet2"
  }
}
```



7) Now create igw.tf

```
resource "aws_internet_gateway" "challa-gateway" {
  vpc_id = aws_vpc.challa.id
}
```



8) Now create route table add route and subnet association

9) route.tf

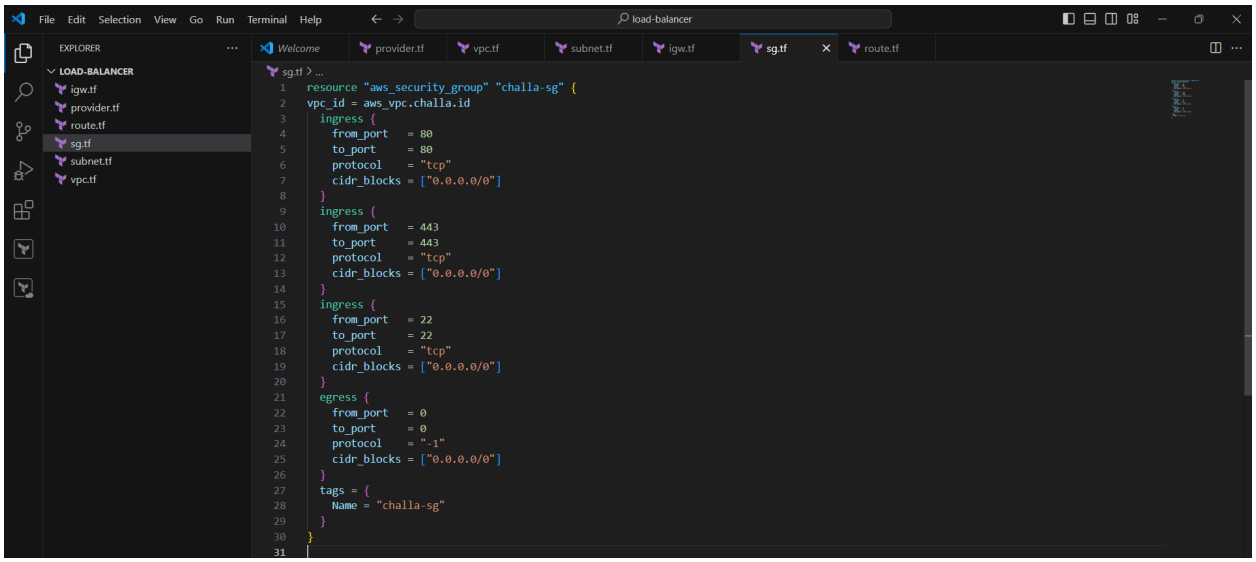
```
resource "aws_route_table" "challa-route" {
  vpc_id = aws_vpc.challa.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.challa-gateway.id
  }
  tags = {
    Name = "route to internet"
  }
}
#route 1
resource "aws_route_table_association" "challa-route1" {
  subnet_id      = aws_subnet.challa-public-subnet1.id
  route_table_id = aws_route_table.challa-route.id
```

```
}
#route 2
resource "aws_route_table_association" "challa-route2" {
  subnet_id      = aws_subnet.challa-public-subnet2.id
  route_table_id = aws_route_table.challa-route.id
}
```

10) Now create security group sg.tf

```
resource "aws_security_group" "challa-sg" {
vpc_id = aws_vpc.challa.id
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
```

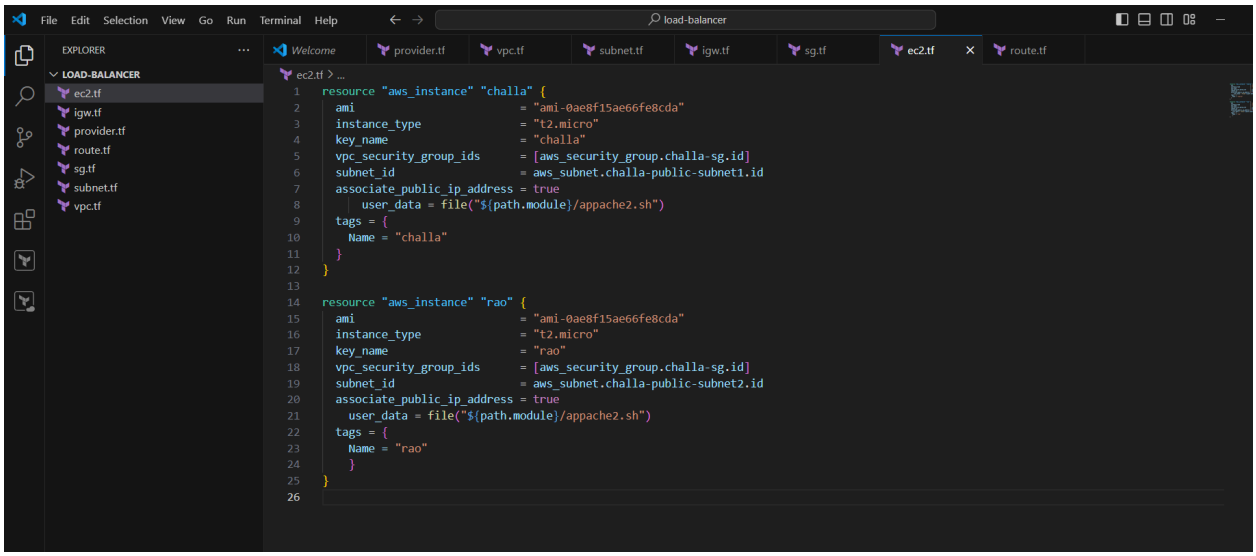
```
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "challa-sg"
  }
}
```



11) Now create 2 ec2 instance ec2.tf

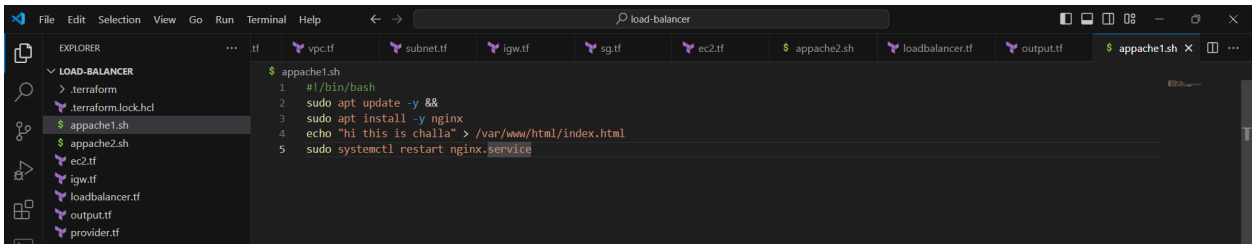
```
resource "aws_instance" "challa" {
  ami                    = "ami-0ae8f15ae66fe8cda"
  instance_type          = "t2.micro"
  key_name               = "challa"
  vpc_security_group_ids = [aws_security_group.challa-sg.id]
  subnet_id              = aws_subnet.challa-public-subnet1.id
  associate_public_ip_address = true
    user_data = file("${path.module}/appache1.sh")
  tags = {
    Name = "challa"
  }
}

resource "aws_instance" "rao" {
  ami                    = "ami-0ae8f15ae66fe8cda"
  instance_type          = "t2.micro"
  key_name               = "rao"
  vpc_security_group_ids = [aws_security_group.challa-sg.id]
  subnet_id              = aws_subnet.challa-public-subnet2.id
  associate_public_ip_address = true
    user_data = file("${path.module}/appache2.sh")
  tags = {
```

```
  Name = "rao"
  }
}
```



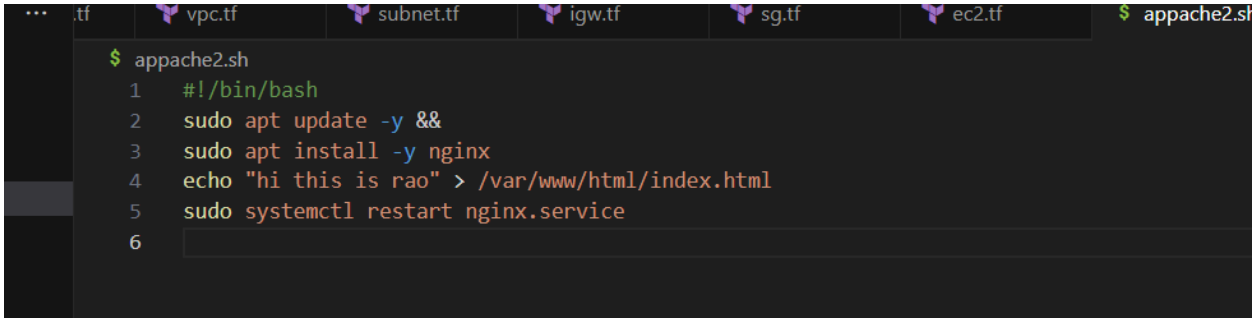
12) Now create appache1.sh to install nginx for challa instance

```
#appache1.sh
#!/bin/bash
sudo apt update -y &&
sudo apt install -y nginx
echo "hi this is challa" > /var/www/html/index.html
sudo systemctl restart nginx.service
```



13) Now create appache2.sh to install nginx for rao instance

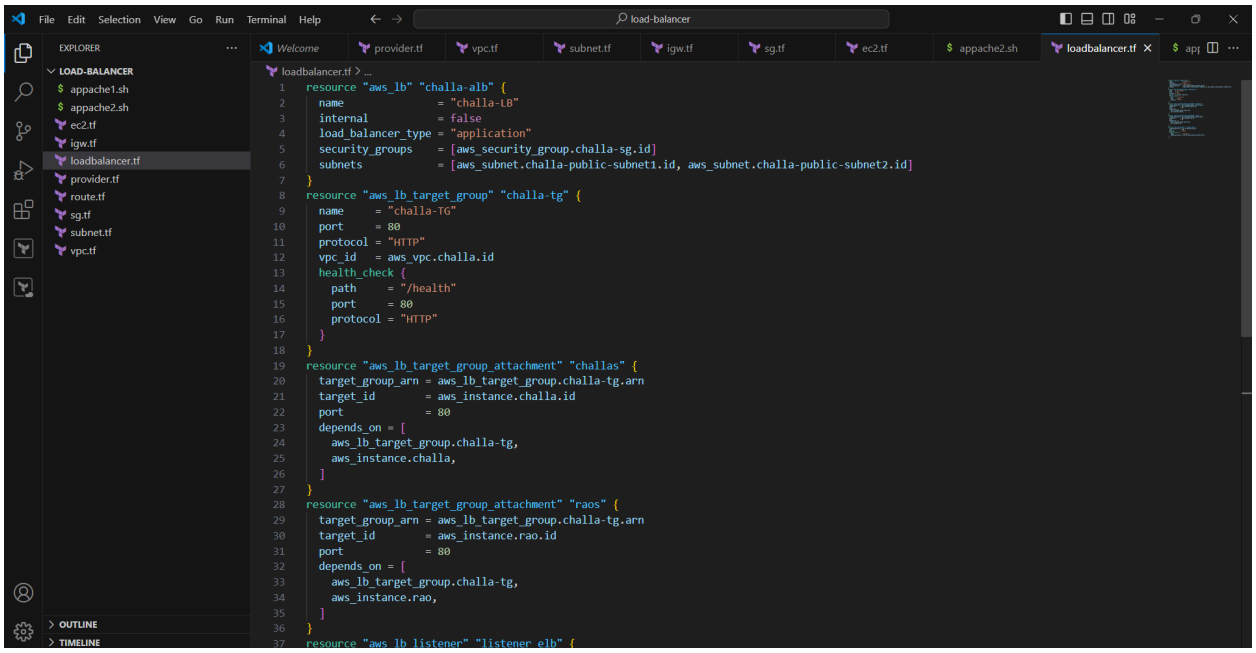
```
#appache2.sh
#!/bin/bash
sudo apt update -y &&
sudo apt install -y nginx
echo "hi this is rao" > /var/www/html/index.html
sudo systemctl restart nginx.service
```

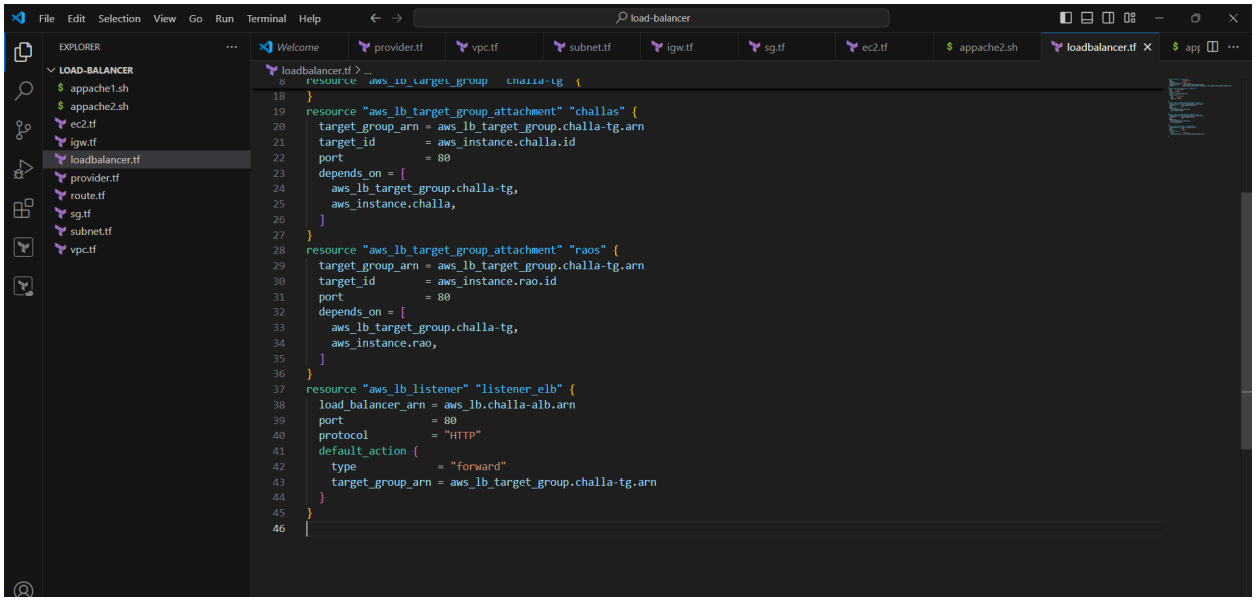


14) Now create load balancer and target group

```
resource "aws_lb" "challa-alb" {
  name               = "challa-LB"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.challa-sg.id]
  subnets            = [aws_subnet.challa-public-subnet1.id, aws_subnet.challa-public-subnet2.id]
}
resource "aws_lb_target_group" "challa-tg" {
  name     = "challa-TG"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.challa.id
  health_check {
    path     = "/health"
    port     = 80
    protocol = "HTTP"
  }
}
resource "aws_lb_target_group_attachment" "challas" {
  target_group_arn = aws_lb_target_group.challa-tg.arn
  target_id        = aws_instance.challa.id
  port             = 80
  depends_on = [
    aws_lb_target_group.challa-tg,
    aws_instance.challa,
  ]
}
resource "aws_lb_target_group_attachment" "raos" {
  target_group_arn = aws_lb_target_group.challa-tg.arn
  target_id        = aws_instance.rao.id
  port             = 80
  depends_on = [
    aws_lb_target_group.challa-tg,
    aws_instance.rao,
  ]
```

```
}
resource "aws_lb_listener" "listener_elb" {
  load_balancer_arn = aws_lb.challa-alb.arn
  port              = 80
  protocol          = "HTTP"
  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.challa-tg.arn
  }
}
```

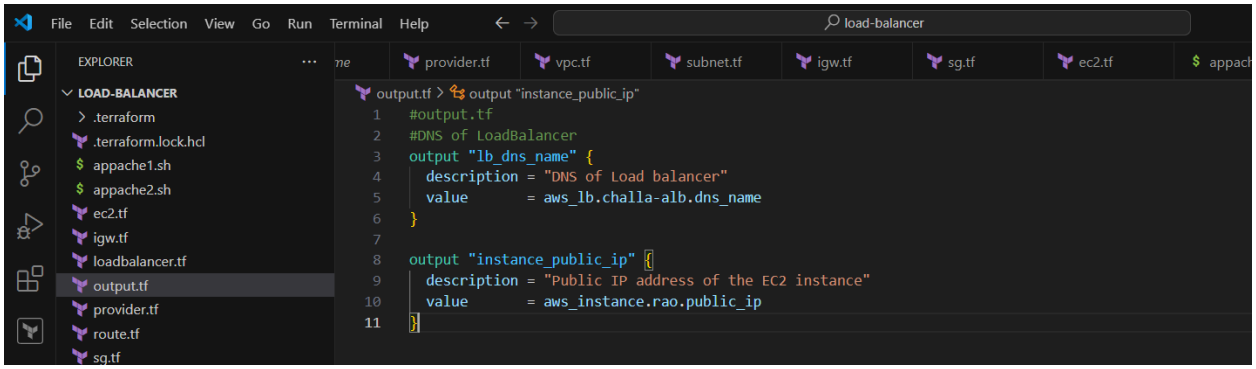




15) Now create output.tf

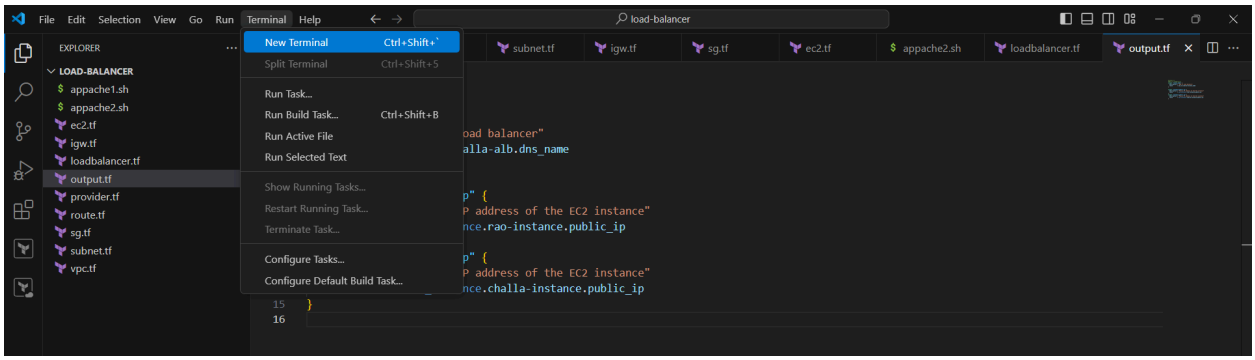
```
#output.tf
```

```
#DNS of LoadBalancer
output "lb_dns_name" {
  description = "DNS of Load balancer"
  value       = aws_lb.challa-alb.dns_name
}

output "instance_public_ip" {
  description = "Public IP address of the EC2 instance"
  value       = aws_instance.rao.public_ip
}
```

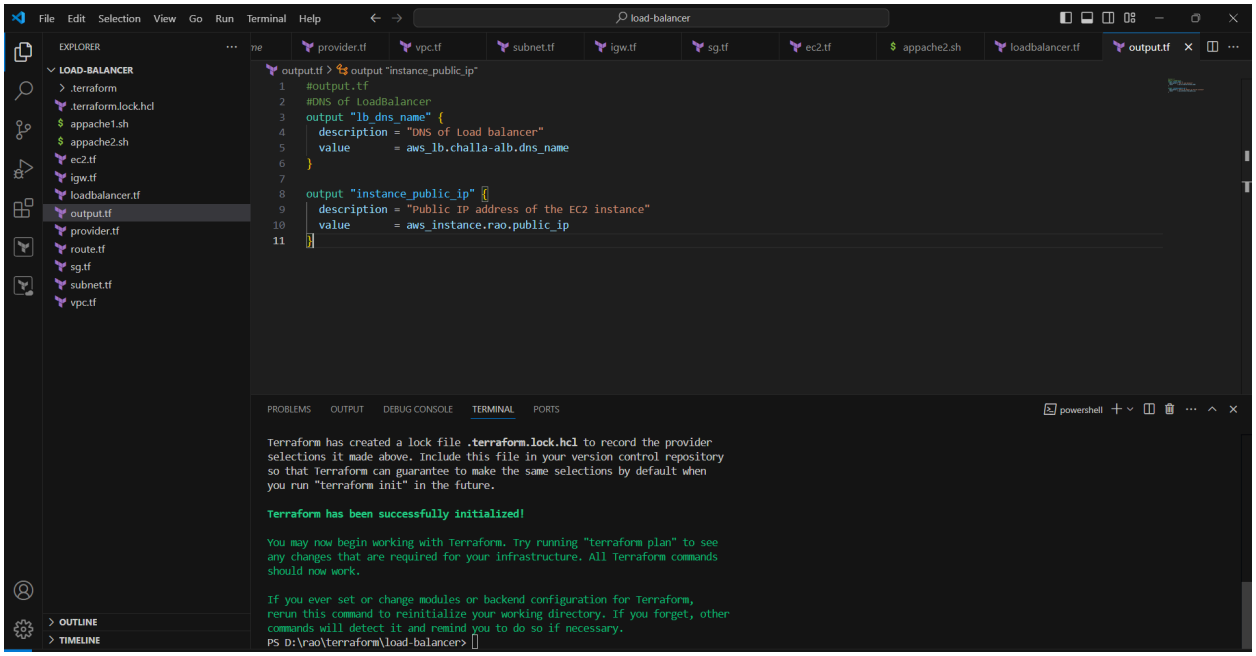


16) Now save all

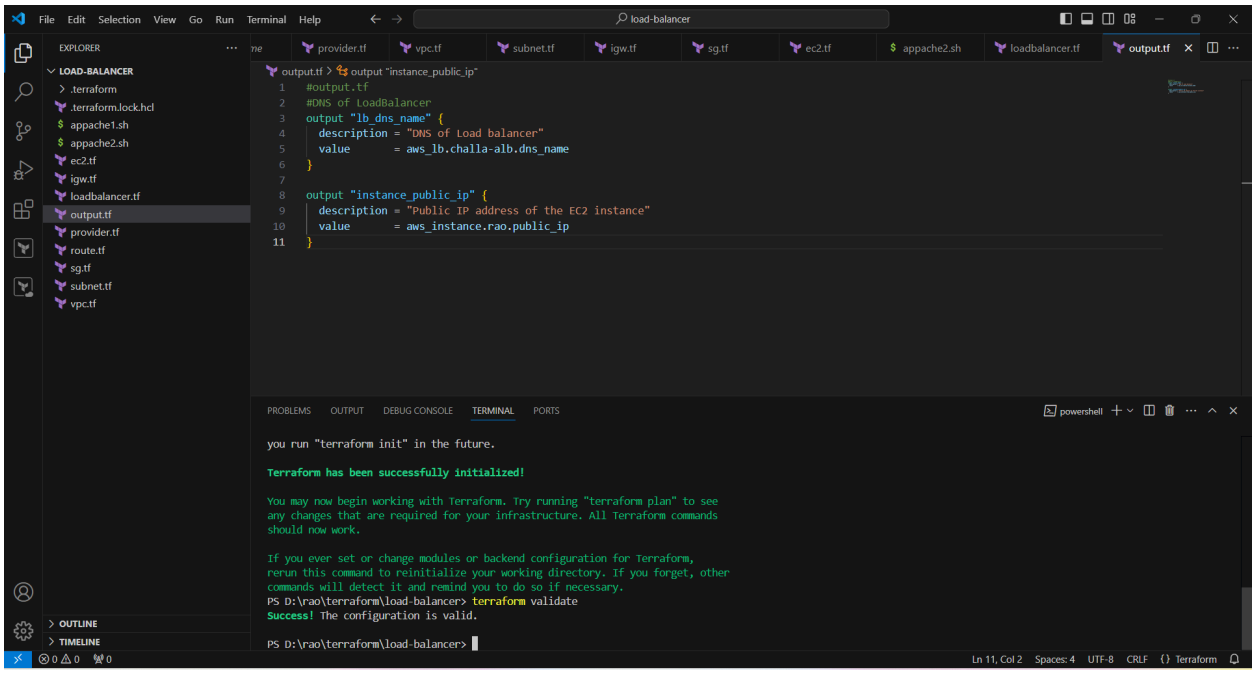
17) Now click on terminal select new terminal



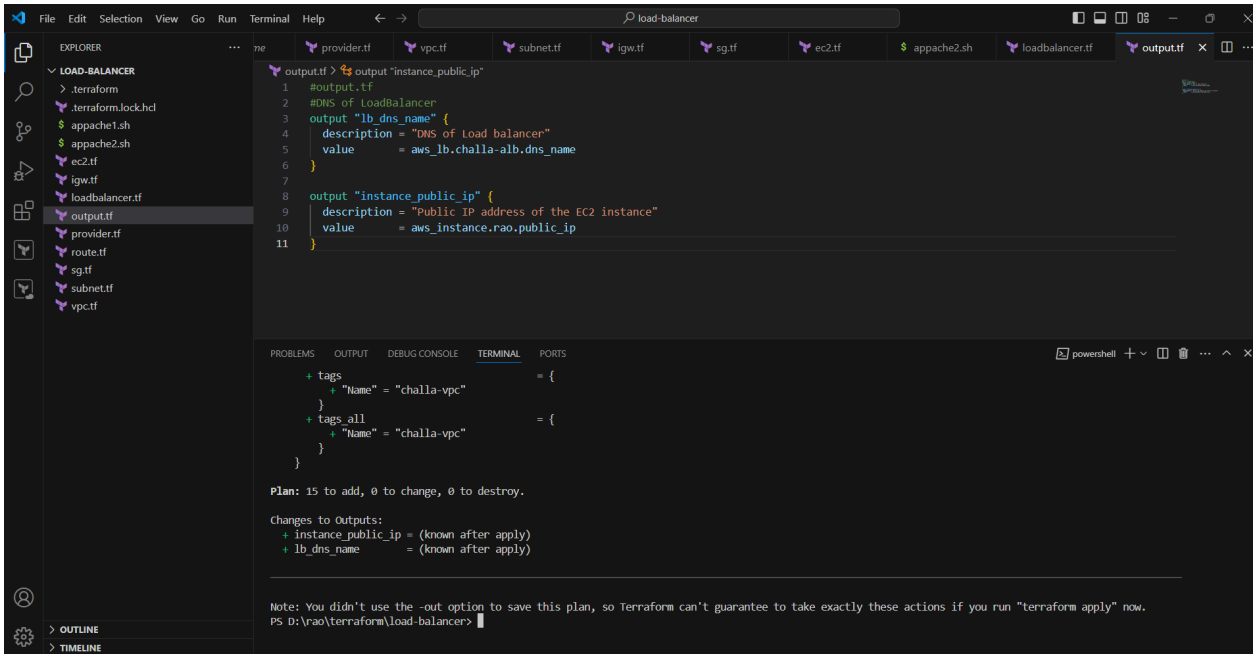
18) Now click on terminal and use #terraform init



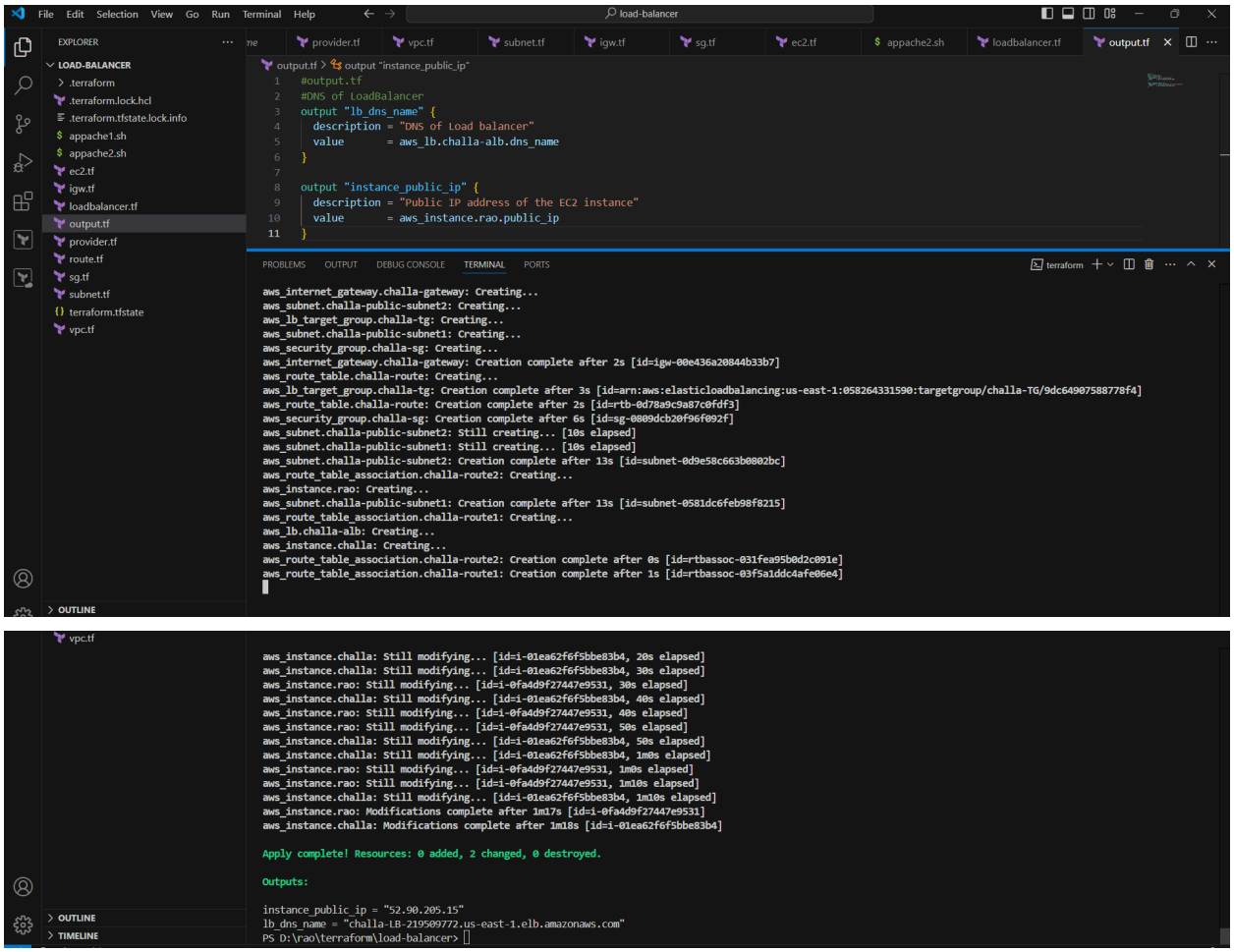
19) #terraform validate



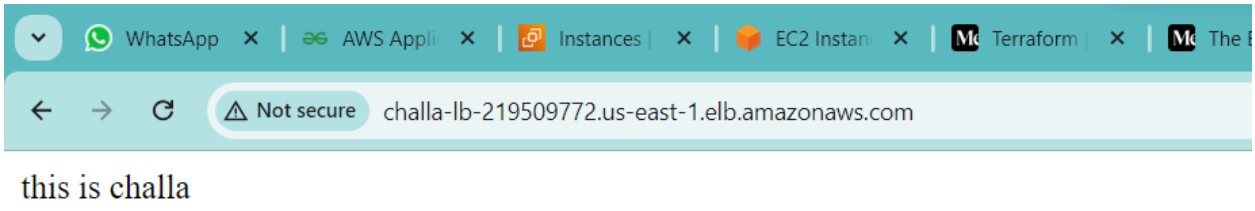
20) Now terraform plan #terraform plan



21) Now terraform apply #terraform apply --auto-aprove

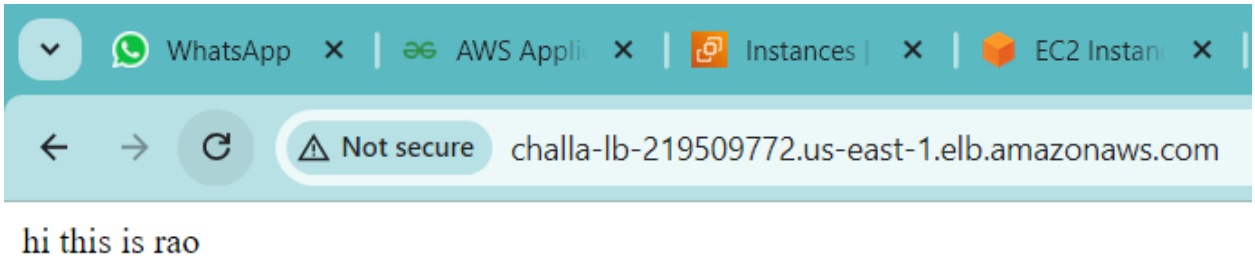


22) Copy dns and paste in google



23)

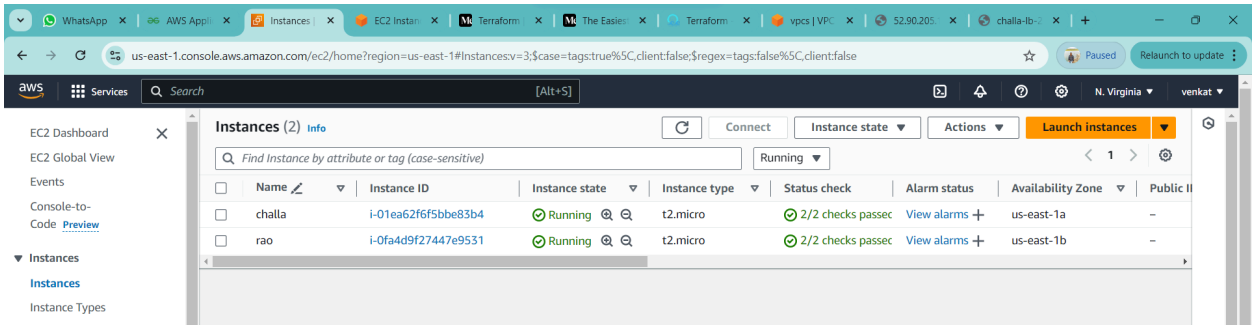
24) Refresh



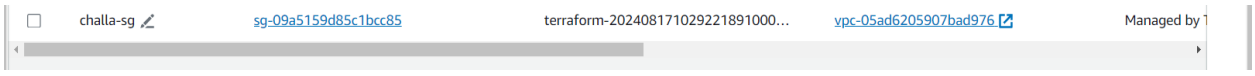
25)

26) Once go and check instances and load balancers

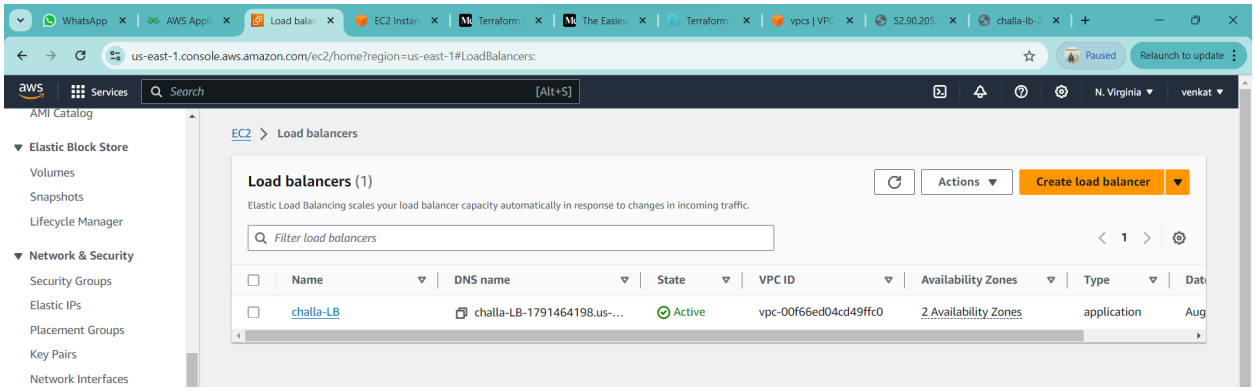
27)



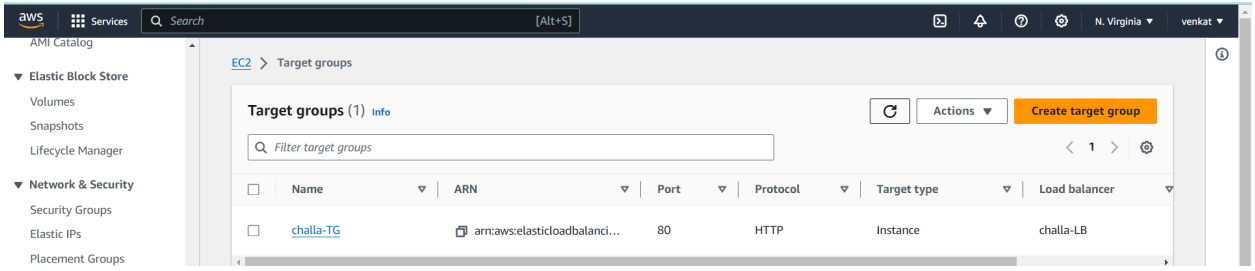
28)



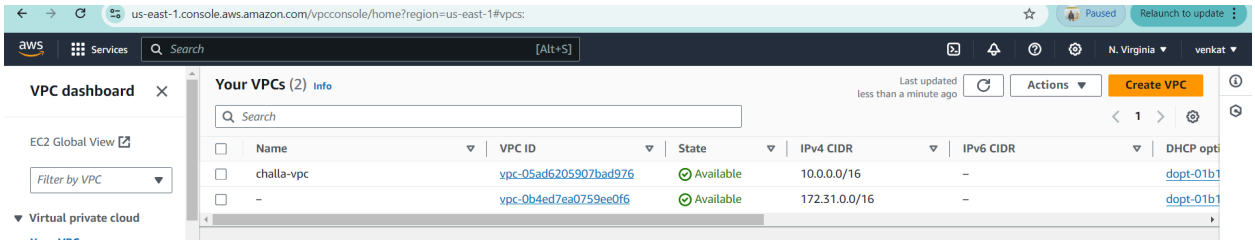
29)



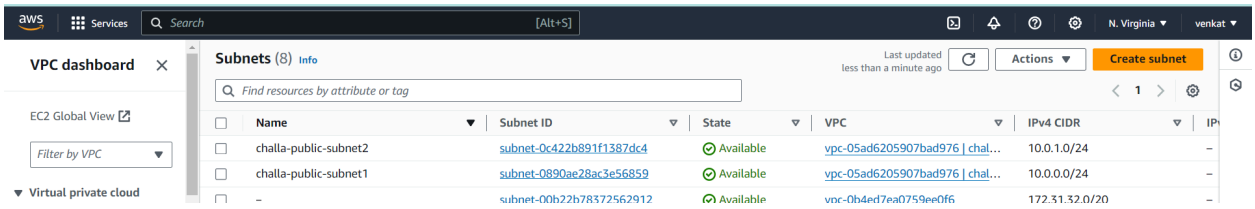
30)



31)



32)

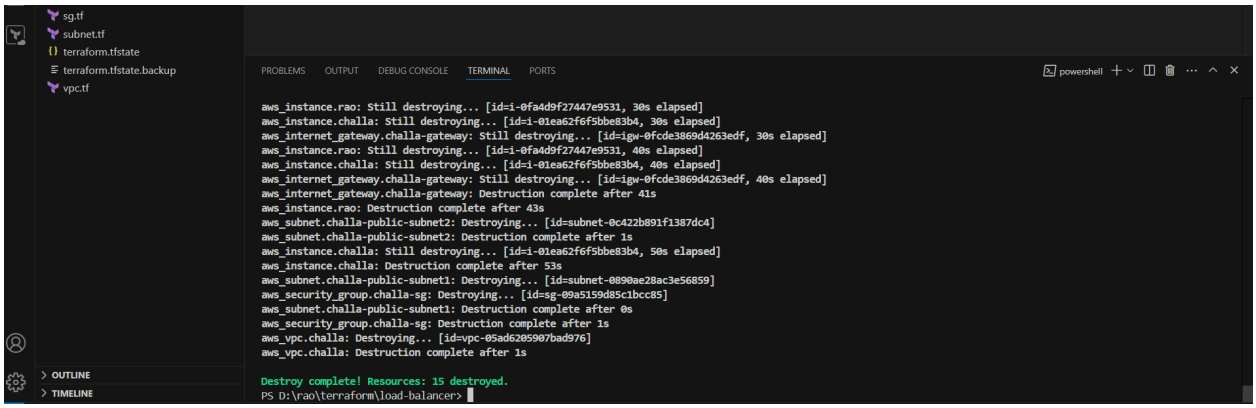


==========

If you want destroy use #terraform destroy

```
aws_instance.rao: Still modifying... [id=i-0fa4d9f27447e9531, 30s elapsed]
aws_instance.challa: Still modifying... [id=i-01ea62f6f5bbe83b4, 40s elapsed]
aws_instance.rao: Still modifying... [id=i-0fa4d9f27447e9531, 40s elapsed]
aws_instance.rao: Still modifying... [id=i-0fa4d9f27447e9531, 50s elapsed]
aws_instance.challa: Still modifying... [id=i-01ea62f6f5bbe83b4, 50s elapsed]
aws_instance.challa: Still modifying... [id=i-01ea62f6f5bbe83b4, 1m0s elapsed]
aws_instance.rao: Still modifying... [id=i-0fa4d9f27447e9531, 1m0s elapsed]
aws_instance.rao: Still modifying... [id=i-0fa4d9f27447e9531, 1m10s elapsed]
aws_instance.challa: Still modifying... [id=i-01ea62f6f5bbe83b4, 1m10s elapsed]
aws_instance.rao: Modifications complete after 1m17s [id=i-0fa4d9f27447e9531]
aws_instance.challa: Modifications complete after 1m18s [id=i-01ea62f6f5bbe83b4]

Apply complete! Resources: 0 added, 2 changed, 0 destroyed.

Outputs:

instance_public_ip = "52.90.205.15"
lb_dns_name = "challa-LB-219509772.us-east-1.elb.amazonaws.com"
PS D:\rao\terraform\load-balancer> terraform destroy
aws_vpc.challa: Refreshing state... [id=vpc-05ad6205907bad976]
```

```
      - owner_id                            = "058264331590" -> null
      - tags                                = {
          - "Name" = "challa-vpc"
        } -> null
      - tags_all                            = {
          - "Name" = "challa-vpc"
        } -> null
        # (4 unchanged attributes hidden)
    }

Plan: 0 to add, 0 to change, 15 to destroy.

Changes to Outputs:
  - instance_public_ip = "54.91.167.113" -> null
  - lb_dns_name        = "challa-LB-219509772.us-east-1.elb.amazonaws.com" -> null

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes
```



==========