Terraform Task: ===============

1) Create VPC

2) Create Internet gateway

3) Create Custom Route Table

4) Create Subnet

5) Associate subnet with Route Table

6) Create Security Group to allow port 22.80,443

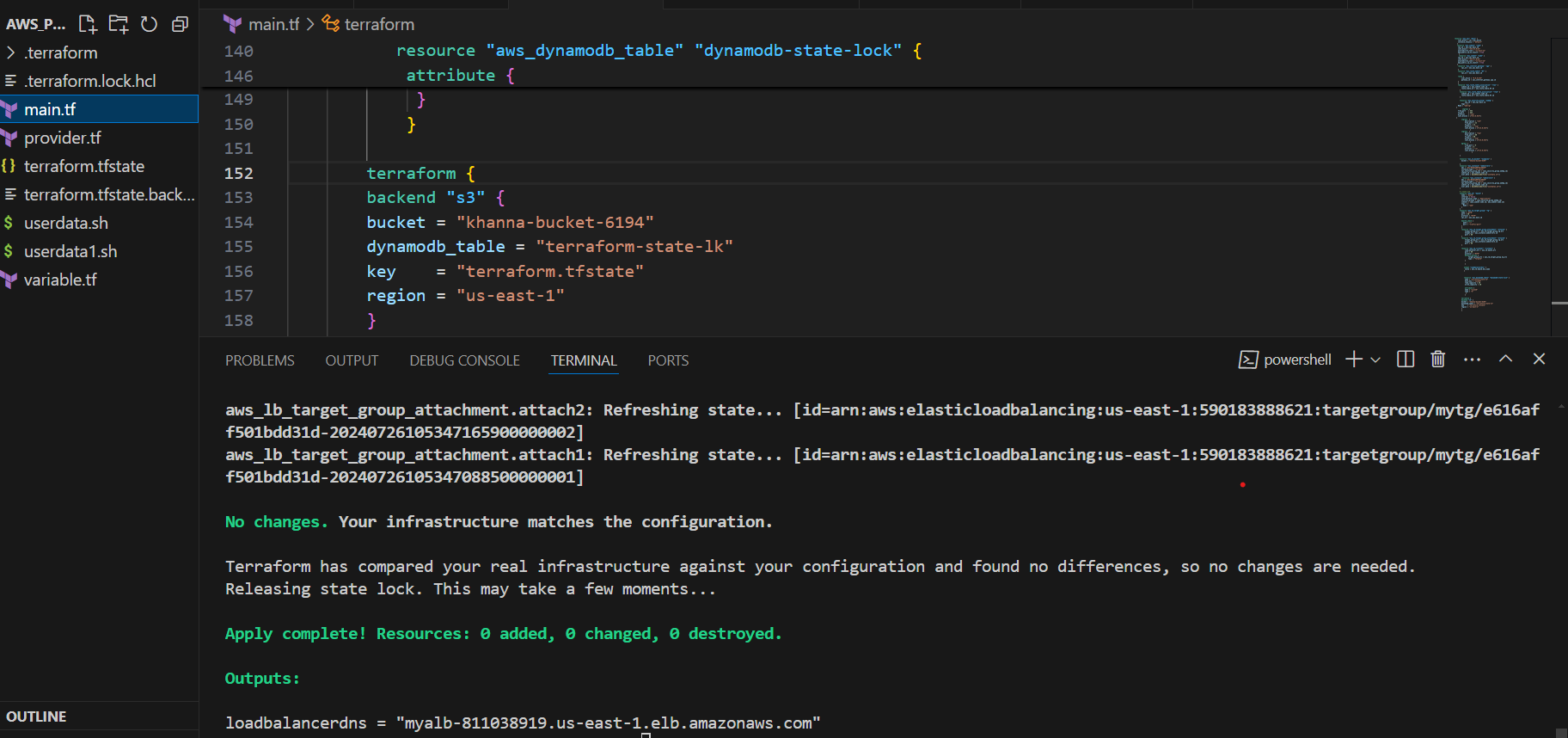
7) Create a network interface with an ip in the subnet that was created in step 4

8) Assign an elastic IP to the network interface created in step 7

9) Create Ubuntu server and install/enable apache2

Note: 1) Create single main.tf which will be created the above resources and do not hardcode the id's.

2) Configure s3 as backend and dynamo db locking for multi user execution.



**Provide.tf:-**

terraform {

  required\_providers {

    aws = {

      source = "hashicorp/aws"

      version = "5.60.0"

    }

  }

}

provider "aws" {

  # Configuration options

  region = "us-east-1"

}

----------------------------------------------------------------------------------

**----------------------------------------------------------------------------------------------------------------**

**Main.tf :- File**

**# Creating vpc using terraform resource**

resource "aws\_vpc" "main" {

    cidr\_block = var.cidr\_block

    instance\_tenancy = "default"

   }

**# Creating subnets sub1 and sub2 using terraform resource**

resource "aws\_subnet" "sub1" {

    vpc\_id = aws\_vpc.main.id

    cidr\_block = "10.0.0.0/24"

    availability\_zone = "us-east-1a"

    map\_public\_ip\_on\_launch = true

     }

     resource "aws\_subnet" "sub2" {

    vpc\_id = aws\_vpc.main.id

    cidr\_block = "10.0.1.0/24"

    availability\_zone = "us-east-1b"

    map\_public\_ip\_on\_launch = true

   }

**# Creating internet gate way using terraform resource**

    resource "aws\_internet\_gateway" "igw" {

        vpc\_id = aws\_vpc.main.id

    }

**# Creating route table using terraform resource**

    resource "aws\_route\_table" "RT" {

        vpc\_id = aws\_vpc.main.id

    route {

        cidr\_block = "0.0.0.0/0"

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

    }

    }

**# Creating route table association using terraform resource**

    resource "aws\_route\_table\_association" "rta1" {

        subnet\_id = aws\_subnet.sub1.id

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

      }

      resource "aws\_route\_table\_association" "rta2" {

        subnet\_id = aws\_subnet.sub2.id

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

      }

**# Creating Security group using terraform resource**

      resource "aws\_security\_group" "webSg" {

            vpc\_id = aws\_vpc.main.id

        tags = {

    Name = "web-sg"

  }

         ingress {

    from\_port   = 443

    to\_port     = 443

    protocol    = "tcp"

    cidr\_blocks = ["0.0.0.0/0"]

  }

        ingress  {

            description = "ssh"

            from\_port = 22

            to\_port = 22

            protocol = "tcp"

            cidr\_blocks = ["0.0.0.0/0"]

                  }

        ingress  {

            description = "tcp"

            from\_port = 80

            to\_port = 80

            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"]

                    }

      }

**# Creating s3 bucket using terraform resource**

      resource "aws\_s3\_bucket" "example1" {

        bucket = "khanna-bucket-6194"

       }

**# Creating instance using terraform resource**

      resource "aws\_instance" "webserver1" {

        ami = "ami-04a81a99f5ec58529"

        instance\_type = "t2.micro"

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

        subnet\_id = aws\_subnet.sub1.id

        user\_data = base64encode(file("userdata.sh"))

      }

         resource "aws\_instance" "webserver2" {

        ami = "ami-04a81a99f5ec58529"

        instance\_type = "t2.micro"

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

        subnet\_id = aws\_subnet.sub2.id

        user\_data = base64encode(file("userdata1.sh"))

      }

**# create alb using terraform resource**

      resource "aws\_lb" "myalb" {

        name = "myalb"

        internal = false

        load\_balancer\_type = "application"

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

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

        tags = {

          Name = "web"

        }

      }

**# Creating target group using terraform**

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

        name = "mytg"

        port = 80

        protocol = "HTTP"

        vpc\_id = aws\_vpc.main.id

        health\_check {

          path = "/"

          port = "traffic-port"

        }

        }

**# Creating attachment load balancer to the targets webserver1 and webserver2**

        resource "aws\_lb\_target\_group\_attachment" "attach1" {

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

            target\_id = aws\_instance.webserver1.id

            port = 80

        }

        resource "aws\_lb\_target\_group\_attachment" "attach2" {

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

            target\_id = aws\_instance.webserver2.id

            port = 80

        }

# Creating load balancer listener using terraform.

        resource "aws\_lb\_listener" "listener" {

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

            port = 80

            protocol = "HTTP"

            default\_action {

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

                type = "forward"

            }

            }

# Output of load balancer

           output "loadbalancerdns" {

            value = aws\_lb.myalb.dns\_name

           }

# Creating dynamodb\_table using terraform resource

           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"

             }

            }

**# Creating backend to terraform.tfstate file using terraform resource**

        terraform {

        backend "s3" {

        bucket = "khanna-bucket-6194"

        dynamodb\_table = "terraform-state-lk"

        key    = "terraform.tfstate"

        region = "us-east-1"

        }

        }

**Variable.tf(file)**

variable "cidr\_block" {

    default = "10.0.0.0/16"

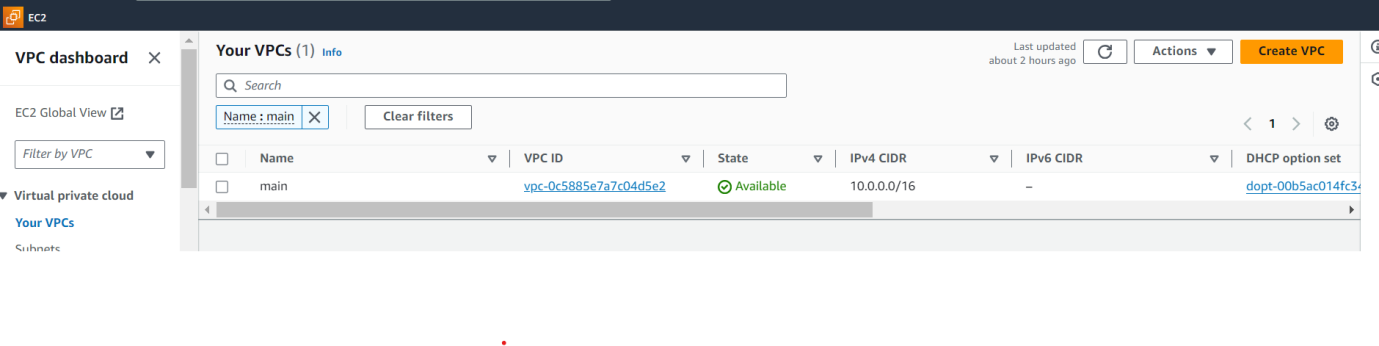
}

----------------------------------------------------------------------------------

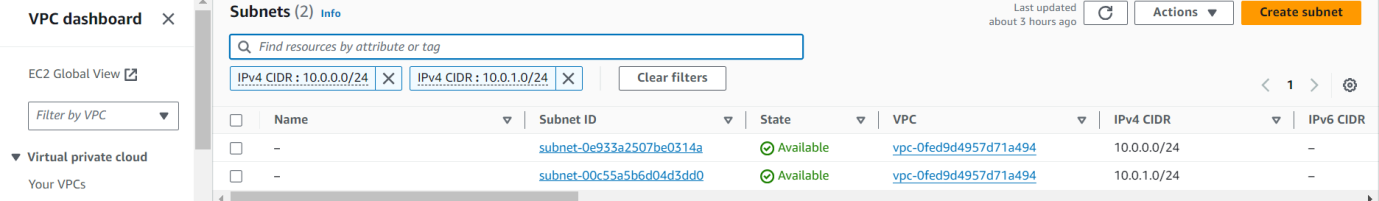
**Output of Project**:-

**VPVP**

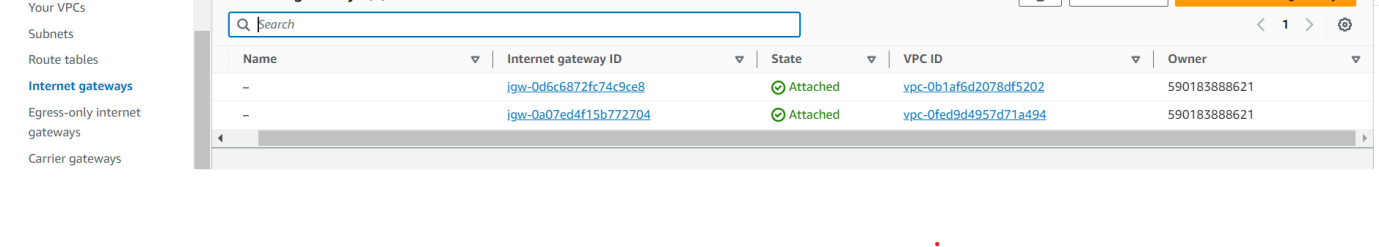
**VPC OUTPUT:-**

****

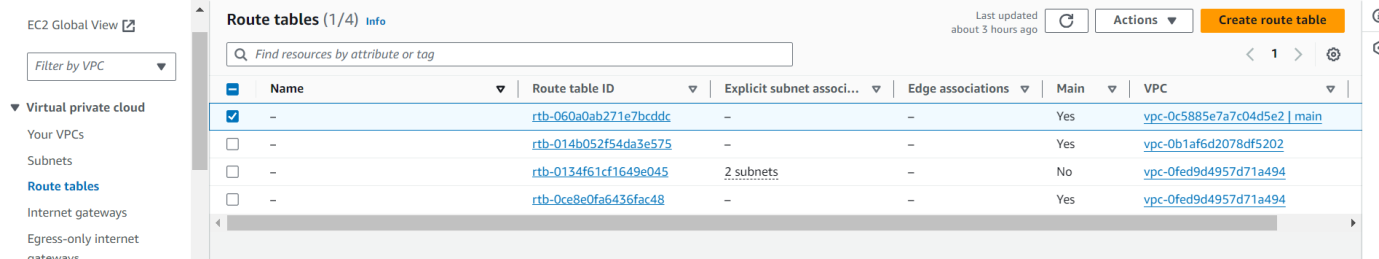
**SSUBNET OUTPUT:-**

****

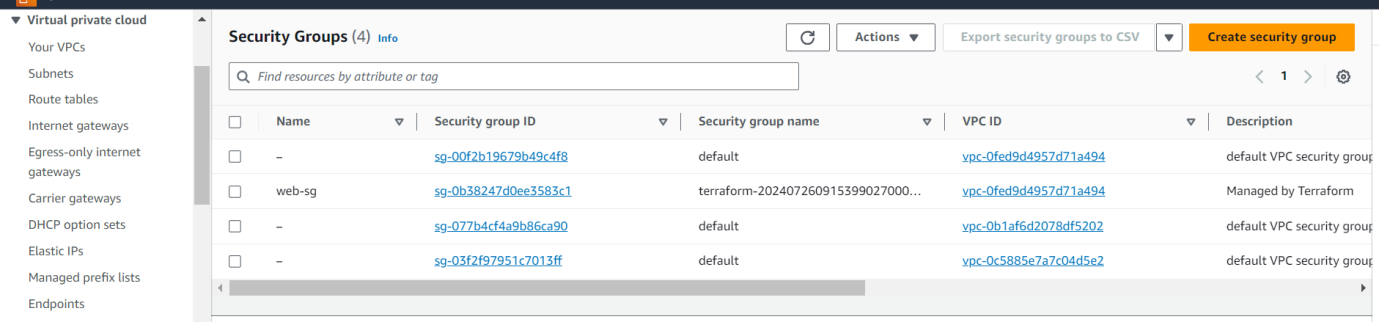
**INTERNETGATE WAY:-**

****

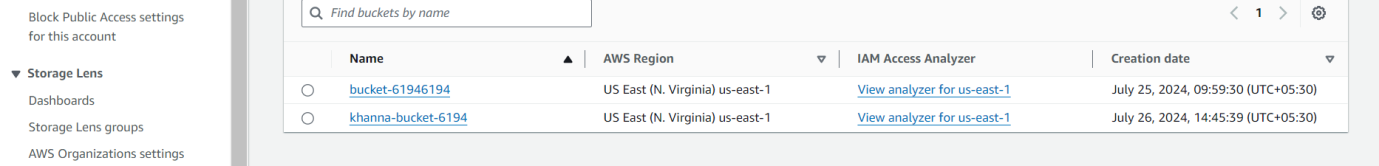
**RrROUTE TABLE:-**

****

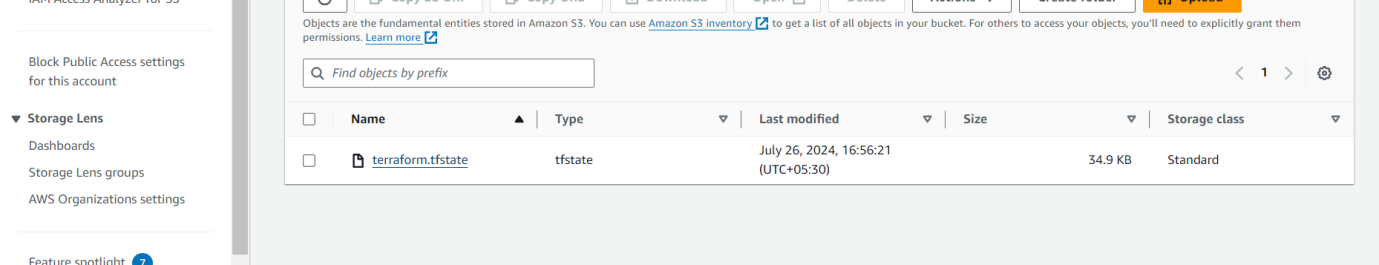
**SSECURITY GROUP:-**

****

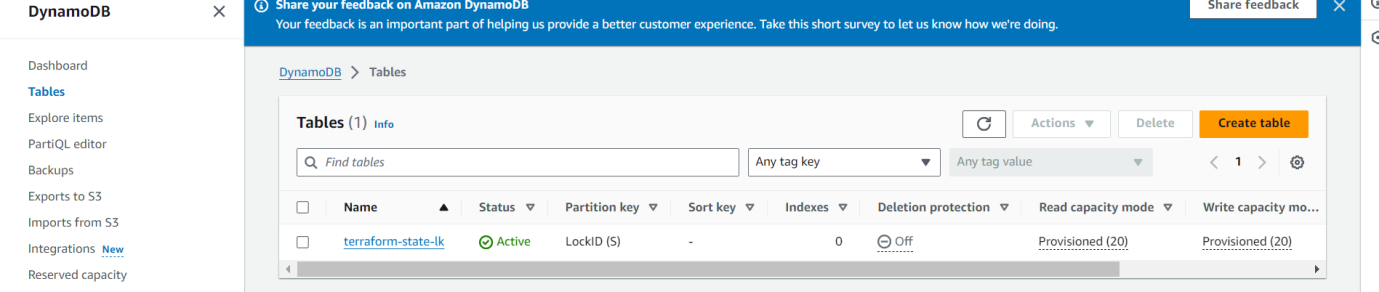
S**S3 BUCKET:-**

****

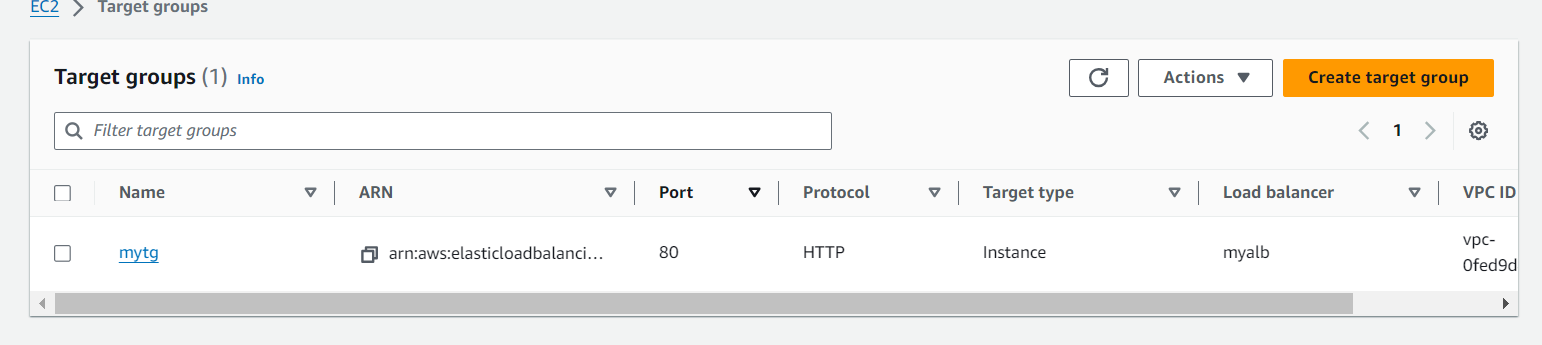
**TTERRAFORM.TFSTATE FILE SEND TO BACKEND:-**

****

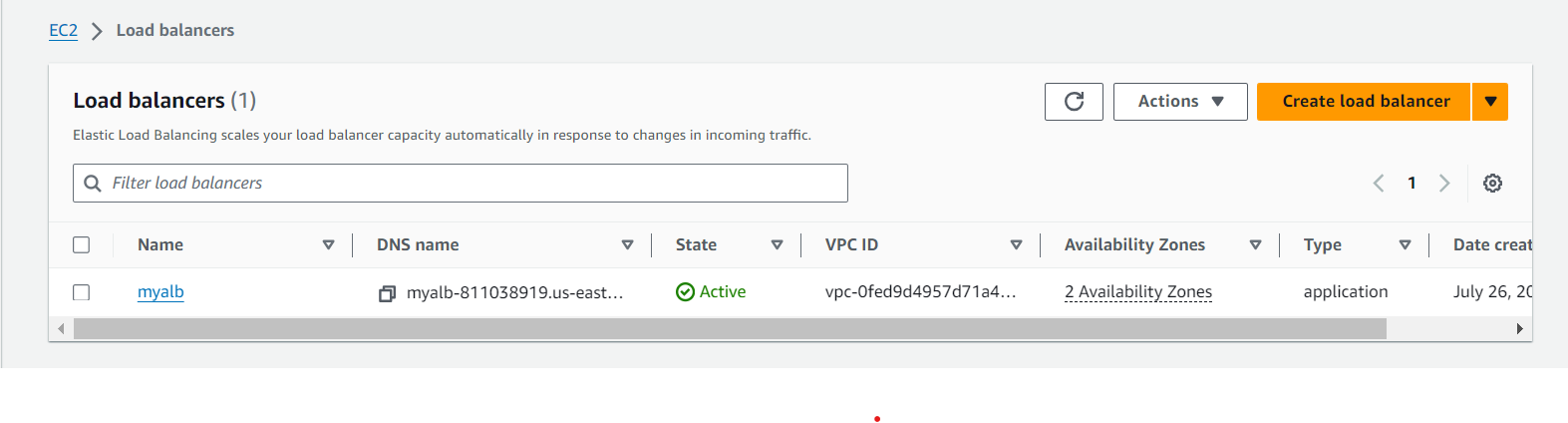
**DYNAMODB:-**

****

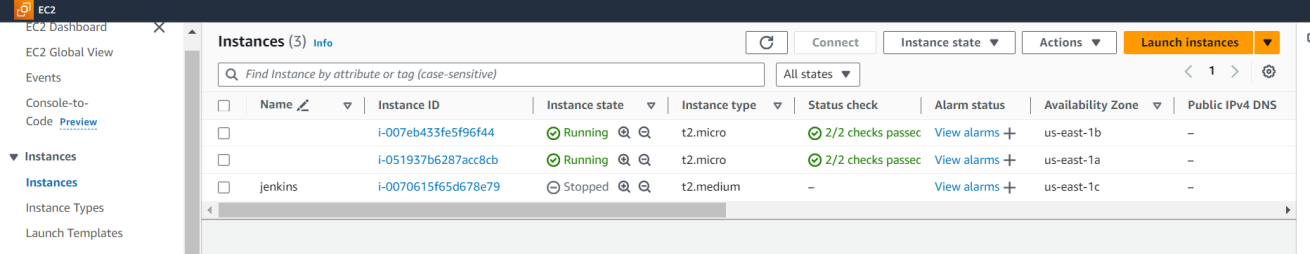
**TARGET GROUP:-**

****

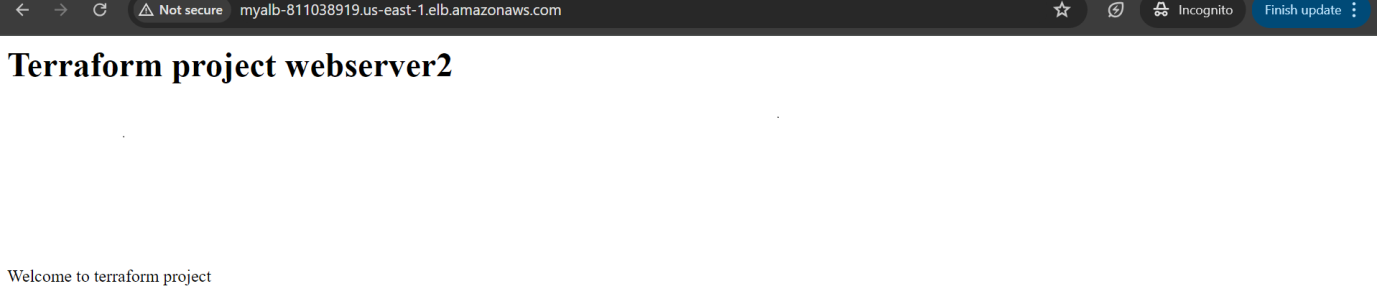
**LOAD BALANCER:-**

****

**EC2 INSTANCES OUT PUT:-**



**LOADBALANCER OUTPUT:-**

****