# Project-1

Terraform :  Terraform is an infrastructure as code tool that lets you build, change, and version cloud and on-prem resources safely and efficiently.

Firstly install terraform in the local machine from terraform website

1. sudo yum install -y yum-utils (utilities).
2. sudo yum-config-manager --add-repo (official HC repository)
3. sudo yum -y install terraform (installs terraform from repo)
4. make directory with the name terraform and initialise it.

First give programatic access and take the user and access keys and configure the local machine and files that stores terraform scripts should have extension ".tf" so that terraform will identify and executes the terraform scripts

Give the aws provider details

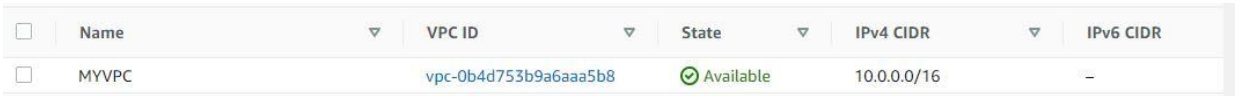
```
provider "aws" {
  region     = "us-east-1"
  access_key = "AKIAXI6JIOP5VD[illegible]BE"
  secret_key = "oJhtSvqBhg7+gph4bxyT0ZQ1mv3s7gI9wu3Wj[illegible]"
}
```

Here we have given the details of the user with access key and secret key along with the region to perform the actions.

**Creating a VPC and Internet Gateway:**

Here we createa vpc and attach it to internet gate way from terraform and confirms it from gui.

```
resource "aws_vpc" "myvpc" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "MYVPC"
  }
}
resource "aws_internet_gateway" "myigw" {
  vpc_id = aws_vpc.myvpc.id

  tags = {
    Name = "MYIGW"
  }
}
```



| Name | VPC ID | State | IPv4 CIDR | IPv6 CIDR |
|---|---|---|---|---|
| MYVPC | vpc-0b4d753b9a6aaa5b8 | Available | 10.0.0.0/16 | – |

Here we can see the vpc and internet gateway is created and it is attached to vpc.

| MYIGW | igw-0d40a1a76c7795222 | Attached | vpc-0b4d753b9a6aaa5b8 \| MYVPC | 5002489 |
|---|---|---|---|---|

Creating subnets using terraform:

Here we create public and private subnets in different availabilty zones and attaching to the vpc. Auto enable ipv4 for public subnets.

Here we have created public and private subnet in (1a) availability zone and apply it through terraform.

```
resource "aws_subnet" "pub-sub" {
  vpc_id                  = aws_vpc.myvpc.id
  cidr_block              = "10.0.1.0/24"
  availability_zone       = "us-east-1a"
  map_public_ip_on_launch = "true"
  tags = {
    Name = "PUB-SUB"
  }
}

resource "aws_subnet" "pvt-sub" {
  vpc_id                  = aws_vpc.myvpc.id
  cidr_block              = "10.0.2.0/24"
  availability_zone       = "us-east-1a"
  map_public_ip_on_launch = "false"

  tags = {
    Name = "PVT-SUB"
  }
}
```

Again launch another public and private subnets in (1b) availabilty zone and apply through it terraform.

```
resource "aws_subnet" "pub-sub1" {
  vpc_id                  = aws_vpc.myvpc.id
  cidr_block              = "10.0.3.0/24"
  availability_zone       = "us-east-1b"
  map_public_ip_on_launch = "true"
  tags = {
    Name = "PUB-SUB1"
  }
}

resource "aws_subnet" "pvt-sub1" {
  vpc_id                  = aws_vpc.myvpc.id
  cidr_block              = "10.0.4.0/24"
  availability_zone       = "us-east-1b"
  map_public_ip_on_launch = "false"

  tags = {
    Name = "PVT-SUB1"
  }
}
```

apply the scripts through terraform and confirm it has created in gui

| | | | | | |
|---|---|---|---|---|---|
| ☐ | PVT-SUB1 | subnet-0147d58561265fa72 | Available | vpc-0b4d753b9a6aaa5b8 \| MY... | 10.0.4.0/24 |
| ☐ | PUB-SUB1 | subnet-05fc49148c92bd758 | Available | vpc-0b4d753b9a6aaa5b8 \| MY... | 10.0.3.0/24 |
| ☐ | – | subnet-04a742dcab725b309 | Available | vpc-0bc0f4d0b4cb83037 | 172.31.16.0/20 |
| ☐ | – | subnet-0716013bb125e5a87 | Available | vpc-0bc0f4d0b4cb83037 | 172.31.64.0/20 |
| ☐ | – | subnet-0ba9e2b8bec2bf2dd | Available | vpc-0bc0f4d0b4cb83037 | 172.31.80.0/20 |
| ☐ | PUB-SUB | subnet-02764688be476c1f7 | Available | vpc-0b4d753b9a6aaa5b8 \| MY... | 10.0.1.0/24 |
| ☐ | – | subnet-0692e3929a3f303be | Available | vpc-0bc0f4d0b4cb83037 | 172.31.0.0/20 |
| ☐ | – | subnet-00af7fd060b76e714 | Available | vpc-0bc0f4d0b4cb83037 | 172.31.32.0/20 |
| ☐ | PVT-SUB | subnet-0f2f43a530d7f1689 | Available | vpc-0b4d753b9a6aaa5b8 \| MY... | 10.0.2.0/24 |

**Creating a route tables :**

Create route tables for the both the subnets and also attch igw to public route table to route traffic in public subnet.

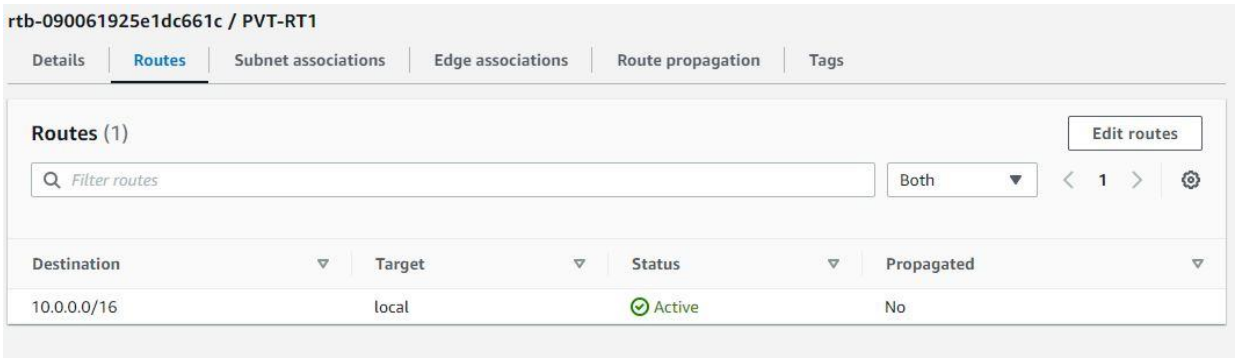
```
resource "aws_route_table" "pub-rt" {
  vpc_id = aws_vpc.myvpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.myigw.id
  }
  tags = {
    Name = "PUB-RT"
  }
}

resource "aws_route_table" "pvt-rt" {
  vpc_id = aws_vpc.myvpc.id
  tags = {
    Name = "PVT-RT"
  }
}
```

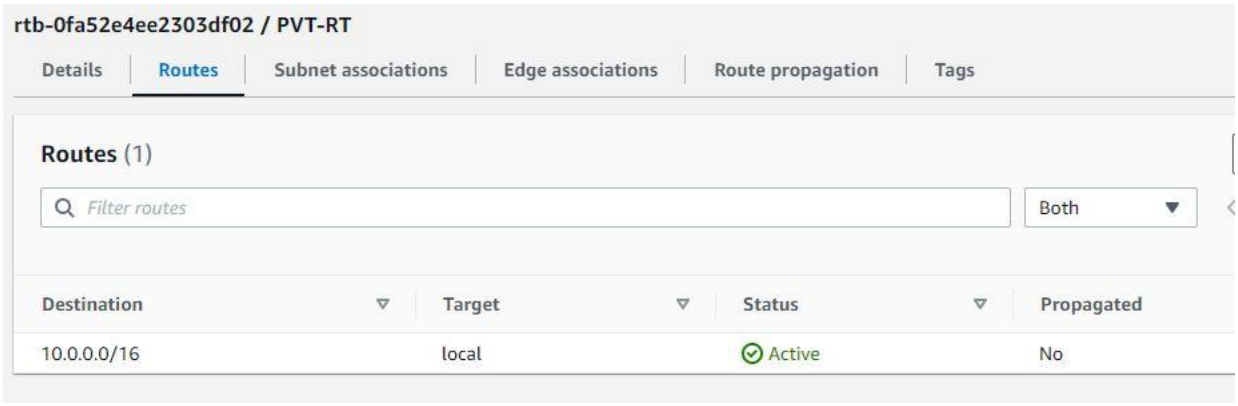
```
resource "aws_route_table" "pub-rt1" {
  vpc_id = aws_vpc.myvpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.myigw.id
  }
  tags = {
    Name = "PUB-RT1"
  }
}

resource "aws_route_table" "pvt-rt1" {
  vpc_id = aws_vpc.myvpc.id
  tags = {
    Name = "PVT-RT1"
  }
}
```

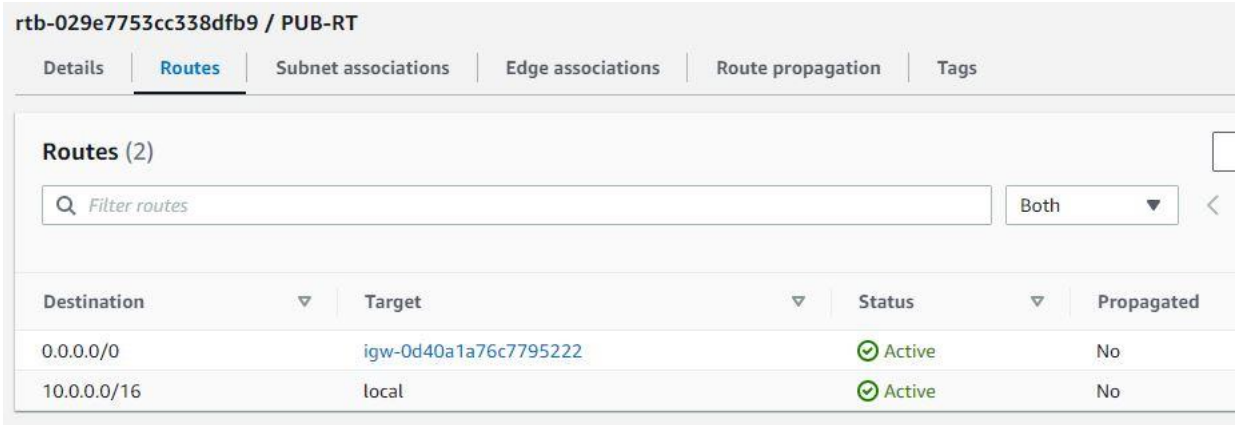
Here we can see the route tables are created

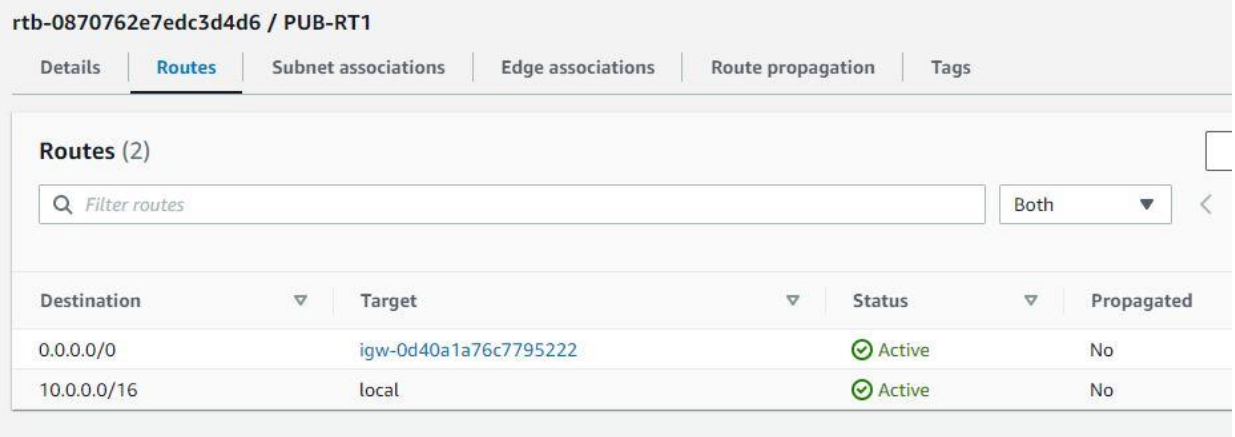
| | | | | | | |
|---|---|---|---|---|---|---|
| ☐ | PVT-RT1 | rtb-090061925e1dc661c | subnet-0147d58561265... | – | No | vpc-0b4d753b |
| ☐ | PVT-RT | rtb-0fa52e4ee2303df02 | subnet-0f2f43a530d7f1... | – | No | vpc-0b4d753b |
| ☐ | – | rtb-08ec05b0ed1417359 | – | – | Yes | vpc-0b4d753b |
| ☐ | PUB-RT | rtb-029e7753cc338dfb9 | subnet-02764688be476... | – | No | vpc-0b4d753b |
| ☐ | PUB-RT1 | rtb-0870762e7edc3d4d6 | subnet-05fc49148c92b... | – | No | vpc-0b4d753b |

verify the route tables routes from gui.






**Associating route tables to subnets:**

Here we associate route tables to subnets and check the routes. So the route tables can route the traffic from subnets to igw.

```
resource "aws_route_table_association" "pub-asso" {
  subnet_id      = aws_subnet.pub-sub.id
  route_table_id = aws_route_table.pub-rt.id
}

resource "aws_route_table_association" "pvt-asso" {
  subnet_id      = aws_subnet.pvt-sub.id
  route_table_id = aws_route_table.pvt-rt.id
}
```

Here public route table is associated to public subnet and vice-versa

```
resource "aws_route_table_association" "pub-asso1" {
  subnet_id      = aws_subnet.pub-sub1.id
  route_table_id = aws_route_table.pub-rt1.id
}

resource "aws_route_table_association" "pvt-asso1" {
  subnet_id      = aws_subnet.pvt-sub1.id
  route_table_id = aws_route_table.pvt-rt1.id
}
```

rtb-0fa52e4ee2303df02 / PVT-RT

Details | Routes | Subnet associations | Edge associations | Route propagation | Tags

**Explicit subnet associations** (1)

Find subnet association

| Name | Subnet ID | IPv4 CIDR |
|---|---|---|
| PVT-SUB | subnet-0f2f43a530d7f1689 | 10.0.2.0/24 |

rtb-090061925e1dc661c / PVT-RT1

Details | Routes | Subnet associations | Edge associations | Route propagation | Tags

**Explicit subnet associations** (1)

Edit subnet associations

Find subnet association

1

| Name | Subnet ID | IPv4 CIDR | IPv6 CIDR |
|---|---|---|---|
| PVT-SUB1 | subnet-0147d58561265fa72 | 10.0.4.0/24 | – |

rtb-029e7753cc338dfb9 / PUB-RT

Details | Routes | Subnet associations | Edge associations | Route propagation | Tags

**Explicit subnet associations** (1)

Edit subnet associations

Find subnet association

1

| Name | Subnet ID | IPv4 CIDR | IPv6 CIDR |
|---|---|---|---|
| PUB-SUB | subnet-02764688be476c1f7 | 10.0.1.0/24 | – |

rtb-0870762e7edc3d4d6 / PUB-RT1

Details | Routes | Subnet associations | Edge associations | Route propagation | Tags

**Explicit subnet associations** (1)

Edit subnet associations

Find subnet association

1

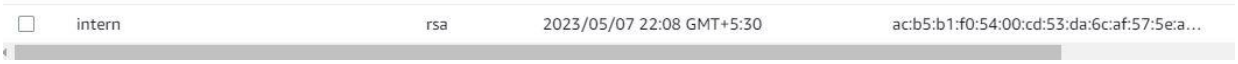
| Name | Subnet ID | IPv4 CIDR | IPv6 CIDR |
|---|---|---|---|
| PUB-SUB1 | subnet-05fc49148c92bd758 | 10.0.3.0/24 | – |

**Creating Ec2 instance using Terraform:**

First we have to generate a new key pair using terraform

```
resource "tls_private_key" "rsa" {
  algorithm = "RSA"
}

resource "local_file" "intern" {
  content  = tls_private_key.rsa.private_key_pem
  filename = "intern.pem"
}

resource "aws_key_pair" "key121" {
  key_name   = "intern"
  public_key = tls_private_key.rsa.public_key_openssh
}
```

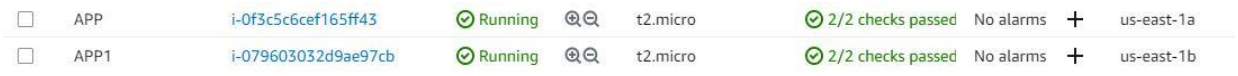
run the terraform script and it generates the new key pair.




Creating a terraform file and launch new instances from it and launching it in different availability zones for high availability.

```
resource "aws_instance" "app" {
  ami                    = "ami-03c7d01cf4dedc891"
  instance_type          = "t2.micro"
  subnet_id              = aws_subnet.pub-sub.id
  availability_zone      = "us-east-1a"
  key_name               = "intern"
  user_data              = file("httpd.sh")
  vpc_security_group_ids = [aws_security_group.ssh-http.id]
  tags = {
    Name = "APP"
  }
}

#2nd instance

resource "aws_instance" "app1" {
  ami                    = "ami-03c7d01cf4dedc891"
  instance_type          = "t2.micro"
  subnet_id              = aws_subnet.pub-sub1.id
  availability_zone      = "us-east-1b"
  key_name               = "intern"
  user_data              = file("httpd.sh")
  vpc_security_group_ids = [aws_security_group.ssh-http.id]
  tags = {
    Name = "APP1"
  }
}
```

we can see the instances running in ec2 console.

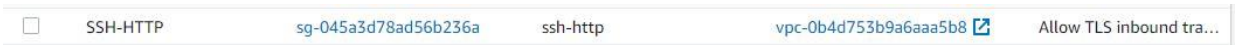



**Creating a security group for front end tier:**

Here we are giving inbound rules 22 and 80 and accepting all connections from outbound rules.

```
resource "aws_security_group" "ssh-http" {
  name        = "ssh-http"
  description = "Allow TLS inbound traffic"
  vpc_id      = aws_vpc.myvpc.id

  ingress {
    description = "TLS from VPC"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    description = "TLS from VPC"
    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"]
    ipv6_cidr_blocks = ["::/0"]
  }

  tags = {
    Name = "SSH-HTTP"
  }
}
```

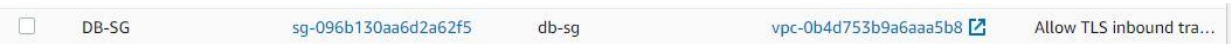
Check the security groups in gui



**Creating security group for database tier:**

Here we have to 3306 port for database to access it and accepting all connections from outside.

```
resource "aws_security_group" "db-sg" {
  name        = "db-sg"
  description = "Allow TLS inbound traffic"
  vpc_id      = aws_vpc.myvpc.id

  ingress {
    description = "TLS from VPC"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    description = "TLS from VPC"
    from_port   = 3306
    to_port     = 3306
    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"]
    ipv6_cidr_blocks = ["::/0"]
  }

  tags = {
    Name = "DB-SG"
  }
}
```

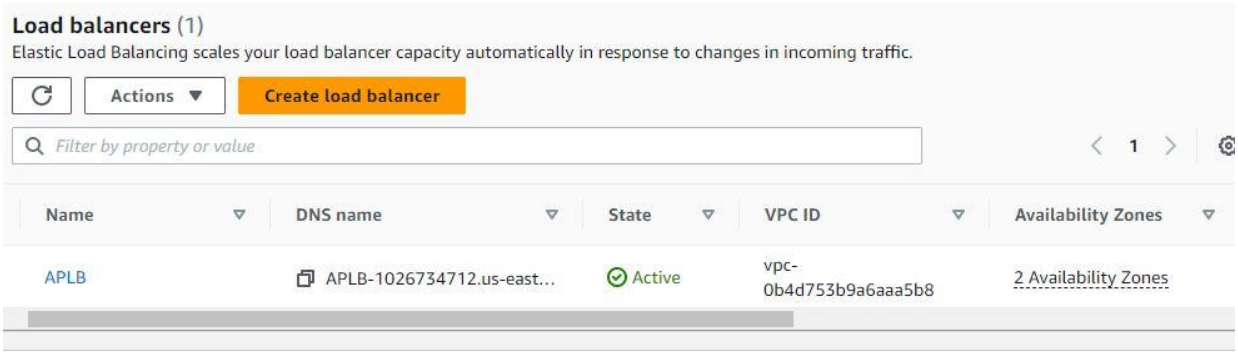


**Creating a Application Load Balancer:**

Create a application load balancer using terraform and attach it to the target groups.

```
resource "aws_lb" "aplb" {
  name               = "APLB"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.ssh-http.id]
  subnets            = [aws_subnet.pub-sub.id, aws_subnet.pub-sub1.id]
}
```

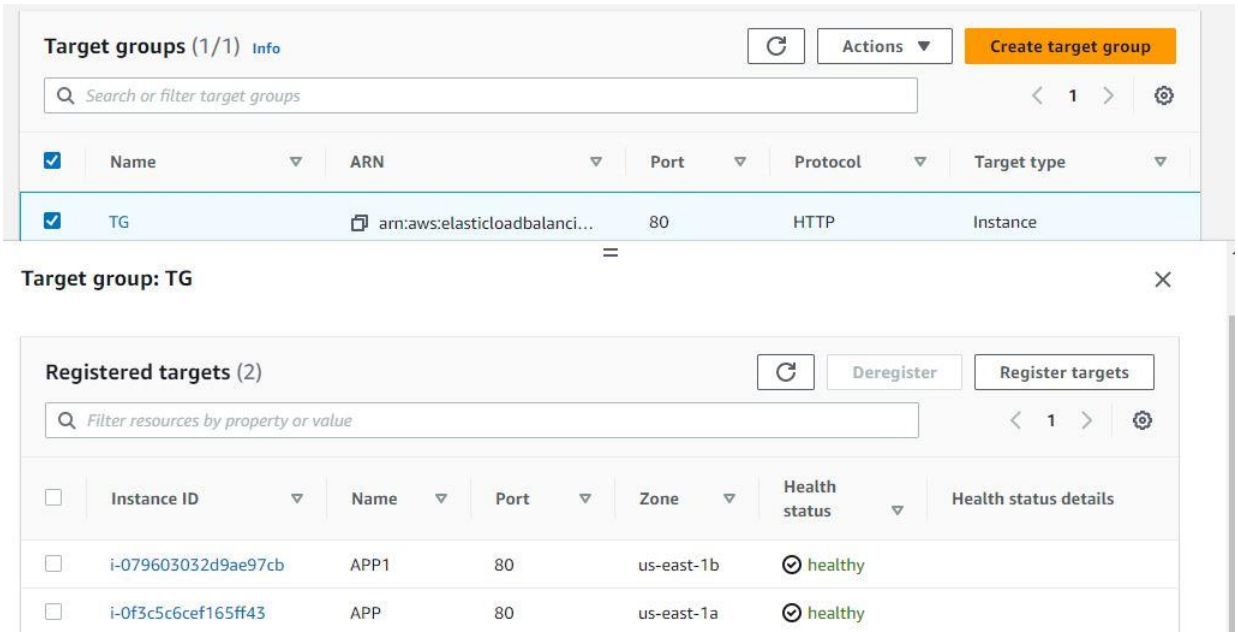
adding the subnets to load balancer where applications are running and also add security groups too.



Creating target groups and attaching it to load balancer.

```
resource "aws_lb_target_group" "tg" {
  name     = "TG"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.myvpc.id
}

resource "aws_lb_target_group_attachment" "attach" {
  target_group_arn = aws_lb_target_group.tg.arn
  target_id        = aws_instance.app.id
  port             = 80
  depends_on       = [aws_instance.app]
}

resource "aws_lb_target_group_attachment" "attach1" {
  target_group_arn = aws_lb_target_group.tg.arn
  target_id        = aws_instance.app1.id
  port             = 80
  depends_on       = [aws_instance.app1]
}

resource "aws_lb_listener" "fro" {
  load_balancer_arn = aws_lb.aplb.arn
  port              = 80
  protocol          = "HTTP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.tg.arn
  }
}
```

Create a target group on port 80 with protocol HTTP and register the targets to the target groups. Give the listeners so the loadbalancer gets associated with target groups and distributes the traffic to desired instances.




**Creating a file for rds instance:**

```
resource "aws_db_subnet_group" "rds" {
  name       = "rds"
  subnet_ids = [aws_subnet.pvt-sub.id, aws_subnet.pvt-sub1.id]

  tags = {
    Name = "RDS"
  }
}
```

Here we have created a database in two private subnets in different availability zones

```
resource "aws_db_instance" "rds" {
  allocated_storage    = 10
  db_subnet_group_name = aws_db_subnet_group.rds.id
  db_name              = "mydb"
  engine               = "mysql"
  engine_version       = "8.0.23"
  instance_class       = "db.t2.micro"
  multi_az             = true
  username             = "admini"
  password             = "admin123"
  #parameter_group_name   = "default.mysql5.7"
  skip_final_snapshot    = true
  vpc_security_group_ids = [aws_security_group.db-sg.id]
}
```

here database storage was allocated to 10gb and we have used the mysql verison of 8.0.23.
we have to give username and password to access it from ec2 console.

**Creating a file for outputs:**

Here we will store the dns address of the load balancer and it displays output after the terraform script gets excutes.

```
output "lb_dns_name" {
  description = "URL of load balancer"
  value       = aws_lb.aplb.dns_name
}
```

**creating user data for ec2:**

First we have to write a script file with name "httpd.sh" and writes the code in this and we have to change the file permissions and give execution permission.

```
#!/bin/bash
sudo yum -y install git
sudo yum -y install httpd
sudo systemctl start httpd
sudo systemctl enable httpd
sudo git clone https://github.com/charan675/ecomm.git /var/www/html
```

this script file should be given  in ec2 userdata

```
resource "aws_instance" "app" {
  ami                    = "ami-03c7d01cf4dedc891"
  instance_type          = "t2.micro"
  subnet_id              = aws_subnet.pub-sub.id
  availability_zone      = "us-east-1a"
  key_name               = "intern"
  user_data              = file("httpd.sh")
  vpc_security_group_ids = [aws_security_group.ssh-http.id]
  tags = {
    Name = "APP"
  }
}
```

Here we have given the script file in user data. Run terraform apply to see the changes in the web page

B.Vidya Charan