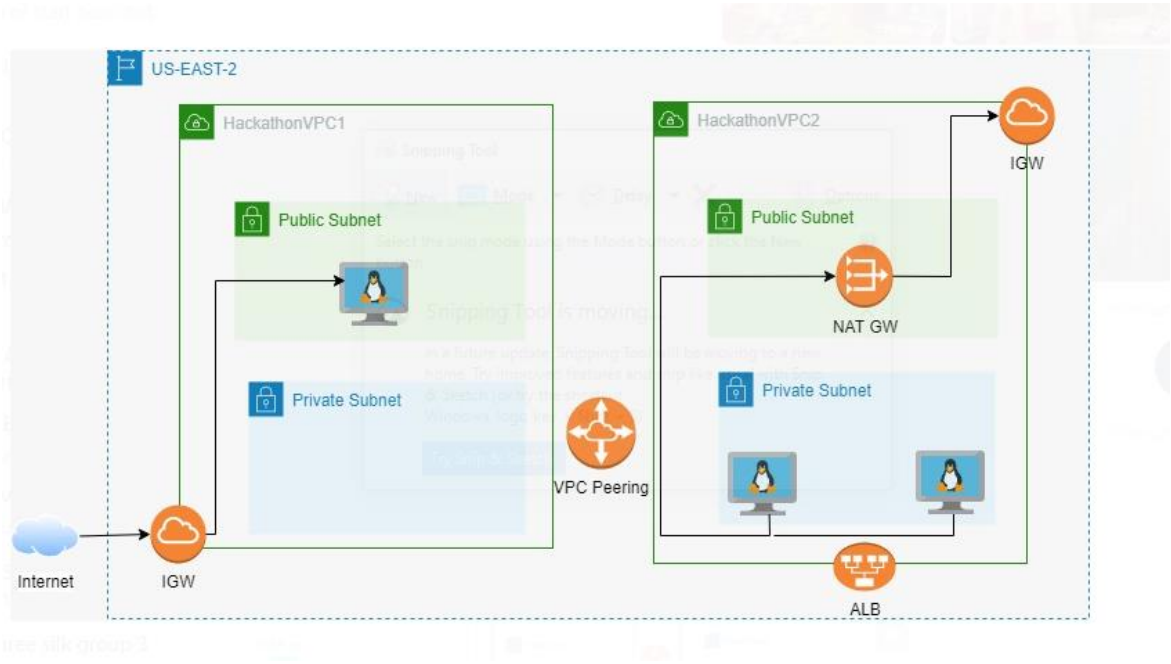
Creating the below AWS architecture using terraform



Step 1 : Create VPC1 and Public subnet assosiate root table with internet gateway

```
resource "aws_vpc" "vpc1" {
  cidr_block = "192.178.0.0/16"

  tags = {
     Name = "VPC1"
    }

}
#PUBLIC SUBNET
resource "aws_subnet" "vpc1_public1" {
  vpc_id     = aws_vpc.vpc1.id
  map_public_ip_on_launch = true
  cidr_block = "192.178.1.0/24"

  tags = {
    Name = "public_1"
  }
}

# internet gateway for public subnet
```

```
resource "aws_internet_gateway" "igw1" {
  vpc_id = aws_vpc.vpc1.id

  tags = {
    Name = "internet-gateway-vpc1"
  }
}

#root table and assosiation with subnet

resource "aws_route_table" "route-public1" {
  vpc_id = aws_vpc.vpc1.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw1.id
  }

  tags = {
    Name = "public-route-table"
  }
}

resource "aws_route_table_association" "public" {
  subnet_id      = aws_subnet.vpc1_public1.id
  route_table_id = aws_route_table.route-public1.id
}
```

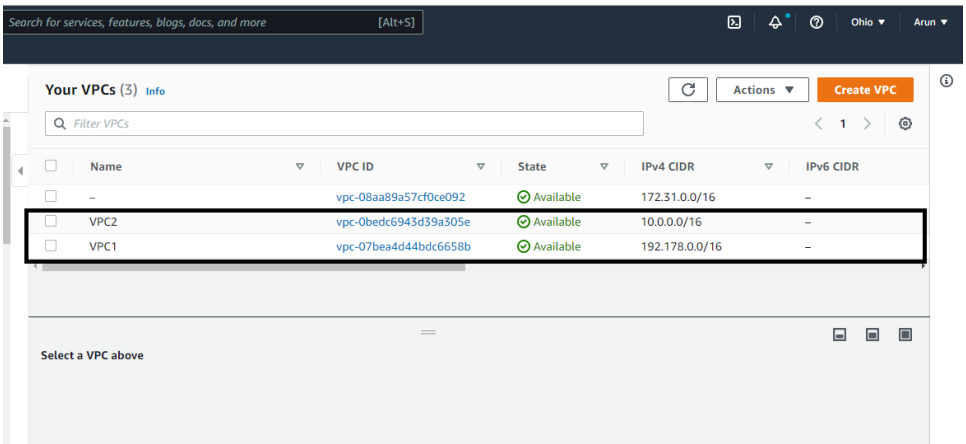
Step 2 : Create VPC2 and Private and public subnets, assosiate root table

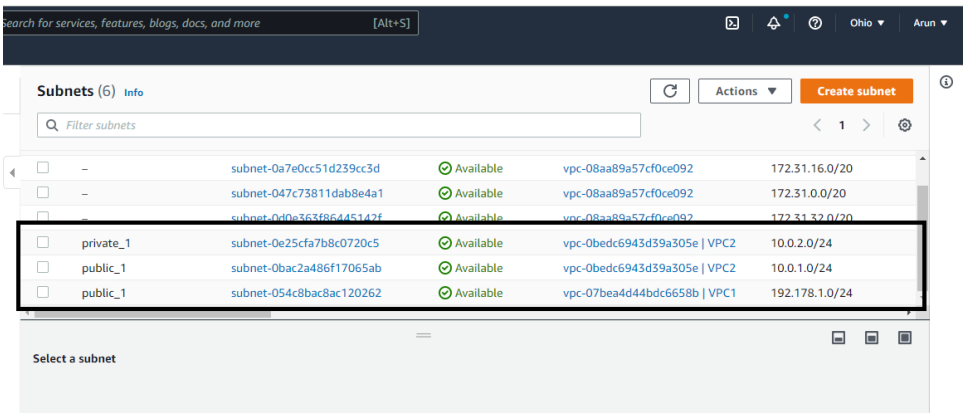
```
resource "aws_vpc" "vpc2" {
  cidr_block = "10.0.0.0/16"

  tags = {
      Name = "VPC2"
    }

}
#PUBLIC SUBNET
resource "aws_subnet" "vpc2_public1" {
  vpc_id     = aws_vpc.vpc2.id
  map_public_ip_on_launch = true
  cidr_block = "10.0.1.0/24"

  tags = {
    Name = "public_1"
```

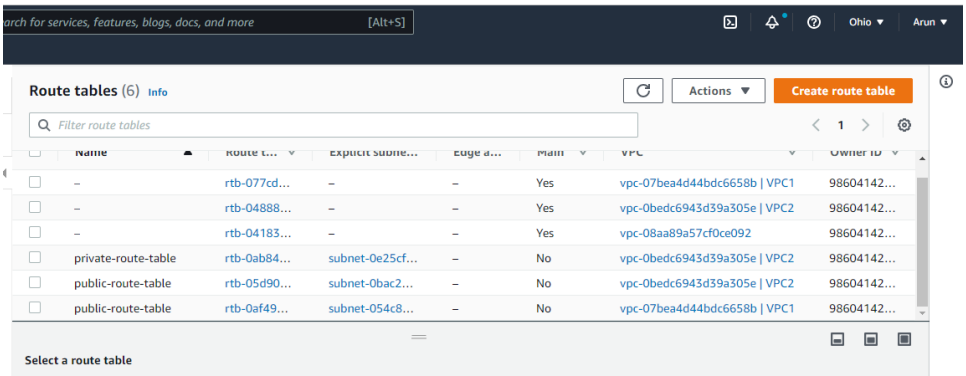
```
  }
}

# internet gateway for public subnet

resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.vpc2.id

  tags = {
    Name = "internet-gateway-vpc2"
  }
}

#root table and assosiation with subnet

resource "aws_route_table" "route-public" {
  vpc_id = aws_vpc.vpc2.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }

  tags = {
    Name = "public-route-table"
  }
}

resource "aws_route_table_association" "public_1" {
  subnet_id      = aws_subnet.vpc2_public1.id
  route_table_id = aws_route_table.route-public.id
}


# Private Subnet
resource "aws_subnet" "vpc2_private1" {
  vpc_id     = aws_vpc.vpc2.id
   availability_zone = "us-east-2a"
   map_public_ip_on_launch = false
  cidr_block = "10.0.2.0/24"

  tags = {
    Name = "private_1"
  }
}

# nat gateway for Private subnet
resource "aws_eip" "nat" {
```

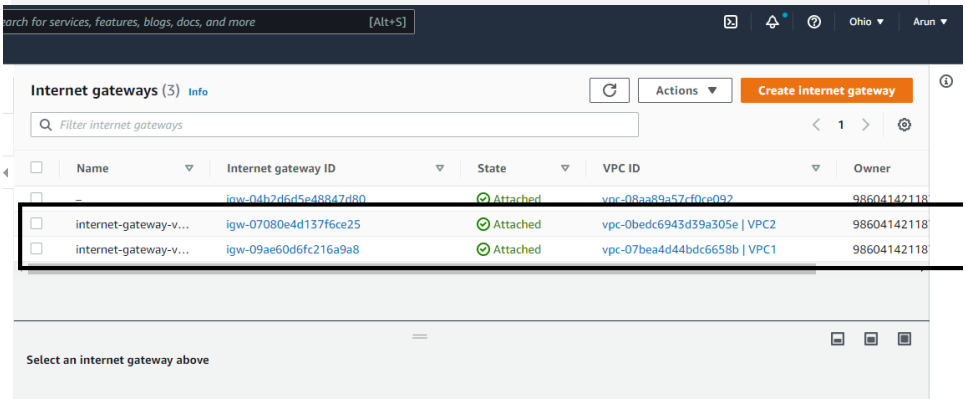
```
  vpc      = true
}

resource "aws_nat_gateway" "nat_gw" {
  allocation_id = aws_eip.nat.id
  subnet_id     = aws_subnet.vpc2_public1.id #nat gateway placed in public subnet
  depends_on    = [aws_internet_gateway.igw] #nat gateway depends on igw
}

resource "aws_route_table" "route_private" {
  vpc_id = aws_vpc.vpc2.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_nat_gateway.nat_gw.id
  }

  tags = {
    Name = "private-route-table"
  }
}

resource "aws_route_table_association" "private_1" {
  subnet_id      = aws_subnet.vpc2_private1.id
  route_table_id = aws_route_table.route_private.id
}

#private subnet 2
resource "aws_subnet" "vpc2_private2" {
  vpc_id     = aws_vpc.vpc2.id
  availability_zone = "us-east-2b"
  map_public_ip_on_launch = false
  cidr_block = "10.0.3.0/24"

  tags = {
    Name = "private_2"
  }
}

# nat gateway for Private subnet
resource "aws_eip" "nat2" {
  vpc      = true
}

resource "aws_nat_gateway" "nat_gw2" {
  allocation_id = aws_eip.nat2.id
  subnet_id     = aws_subnet.vpc2_public1.id #nat gateway placed in public subnet
  depends_on    = [aws_internet_gateway.igw] #nat gateway depends on igw
```

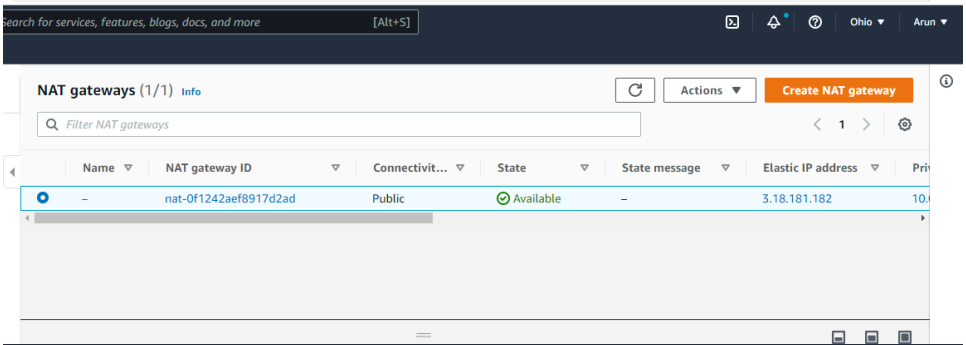
```
}

resource "aws_route_table" "route_private2" {
  vpc_id = aws_vpc.vpc2.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_nat_gateway.nat_gw2.id
  }

  tags = {
    Name = "private-route-table"
  }
}

resource "aws_route_table_association" "private_2" {
  subnet_id      = aws_subnet.vpc2_private2.id
  route_table_id = aws_route_table.route_private2.id
}
```

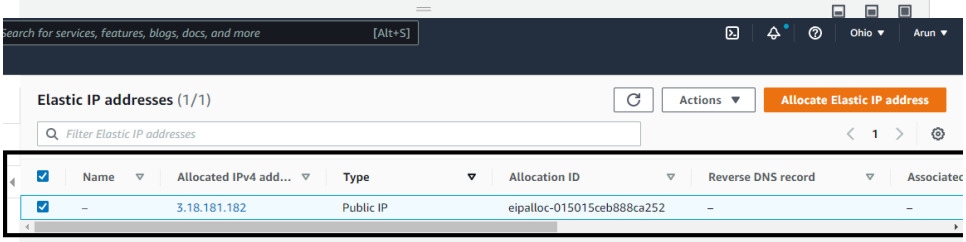












Step 3 : create security groups

```
resource "aws_security_group" "sg_vpc1" {
  name        = "allow_SSH"
  description = "Allow SSH inbound traffic"
  vpc_id      = aws_vpc.vpc1.id
```

```
  ingress {
    # SSH Port 22 allowed from any IP
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

    ingress {
      # SSH Port 22 allowed from any IP
      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"]
  }
}

resource "aws_security_group" "sg_vpc2" {
  name        = "allow_SSH"
  description = "Allow SSH inbound traffic"
  vpc_id      = aws_vpc.vpc2.id

  ingress {
    # SSH Port 22 allowed from any IP
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

    ingress {
      # SSH Port 22 allowed from any IP
      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"]
  }
}
resource "aws_security_group" "sg_vpc2_private" {
  name        = "allow_SSH2"
  description = "Allow SSH inbound traffic"
  vpc_id      = aws_vpc.vpc2.id

  ingress {
    # SSH Port 22 allowed from any IP
    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"]
  }
}
```

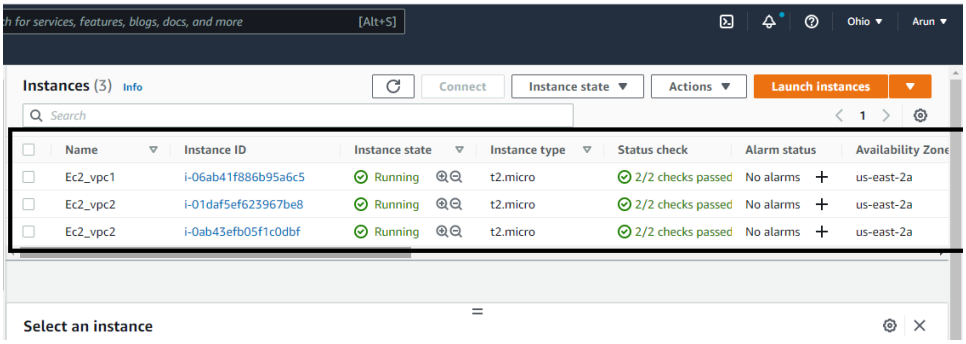
Step 4 : create instances in both vpcs

In VPC1 - public subnet – 1 instance
In VPC2 – Private subnet – 2 instance

```
provider "aws" {
    region= var.region
    access_key = var.access_key
    secret_key = var.secret_key
}
resource "aws_instance" "ec2_Vpc1" {
  ami           = var.ami
  instance_type = var.instance_type
  subnet_id = aws_subnet.vpc1_public1.id
  # Security group assign to instance
  vpc_security_group_ids = [aws_security_group.sg_vpc1.id]

  # key name
  key_name = var.key_name

  user_data = <<EOF
#! /bin/bash
```

```
            sudo yum update -y
sudo yum install -y httpd.x86_64
sudo service httpd start
sudo service httpd enable
echo "<h1>Deployed via Terraform</h1>" | sudo tee /var/www/html/index.html
   yum install java-1.8.0-openjdk-devel -y
   curl --silent --location http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo | sudo tee
/etc/yum.repos.d/jenkins.repo
   sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key
   yum install -y jenkins
   systemctl start jenkins
   systemctl status jenkins
   systemctl enable jenkins
  EOF

  tags = {
   Name = "Ec2_vpc1"
  }
}

resource "aws_instance" "ec2_Vpc2" {
  ami           = var.ami
  instance_type = var.instance_type
  subnet_id = aws_subnet.vpc2_private1.id
  # Security group assign to instance
  vpc_security_group_ids = [aws_security_group.sg_vpc2_private.id]

  # key name
  key_name = var.key_name

  user_data = <<EOF
#! /bin/bash
            sudo yum update -y
sudo yum install -y httpd.x86_64
sudo service httpd start
sudo service httpd enable
echo "<h1>Deployed via Terraform</h1>" | sudo tee /var/www/html/index.html
   yum install java-1.8.0-openjdk-devel -y
   curl --silent --location http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo | sudo tee
/etc/yum.repos.d/jenkins.repo
   sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key
   yum install -y jenkins
   systemctl start jenkins
   systemctl status jenkins
   systemctl enable jenkins
  EOF

  tags = {
```

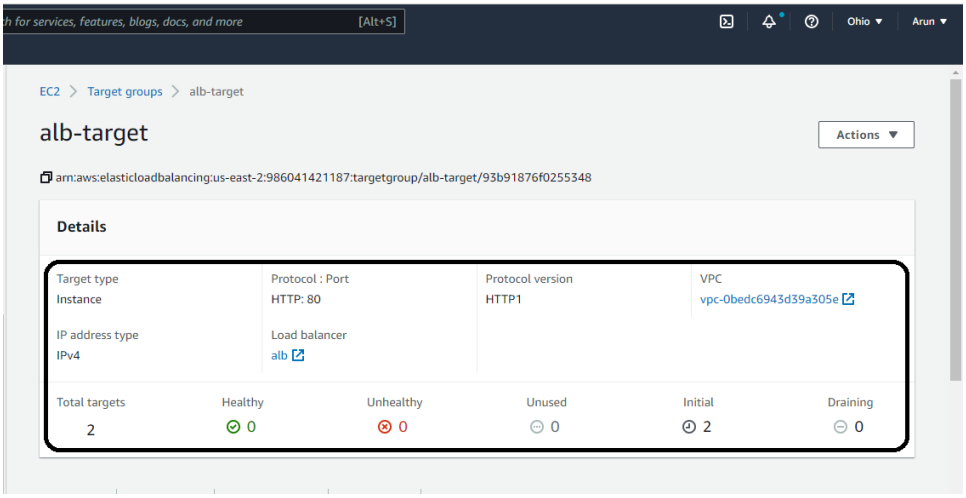
```
  Name = "Ec2_vpc2"
 }
}
resource "aws_instance" "ec2_Vpc2-2" {
 ami          = var.ami
 instance_type = var.instance_type
 subnet_id = aws_subnet.vpc2_private2.id
 # Security group assign to instance
 vpc_security_group_ids = [aws_security_group.sg_vpc2_private.id]

 # key name
 key_name = var.key_name

 user_data = <<EOF
#! /bin/bash
            sudo yum update -y
sudo yum install -y httpd.x86_64
sudo service httpd start
sudo service httpd enable
echo "<h1>Deployed via Terraform</h1>" | sudo tee /var/www/html/index.html
   yum install java-1.8.0-openjdk-devel -y
   curl --silent --location http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo | sudo tee
/etc/yum.repos.d/jenkins.repo
   sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key
   yum install -y jenkins
   systemctl start jenkins
   systemctl status jenkins
   systemctl enable jenkins
  EOF

 tags = {
  Name = "Ec2_vpc2 - 2"
 }
}
```



Step 5 : create ALB in Private subnet in VPC2

```
resource "aws_lb" "alb" {
 name            = "alb"
 internal        = false
```

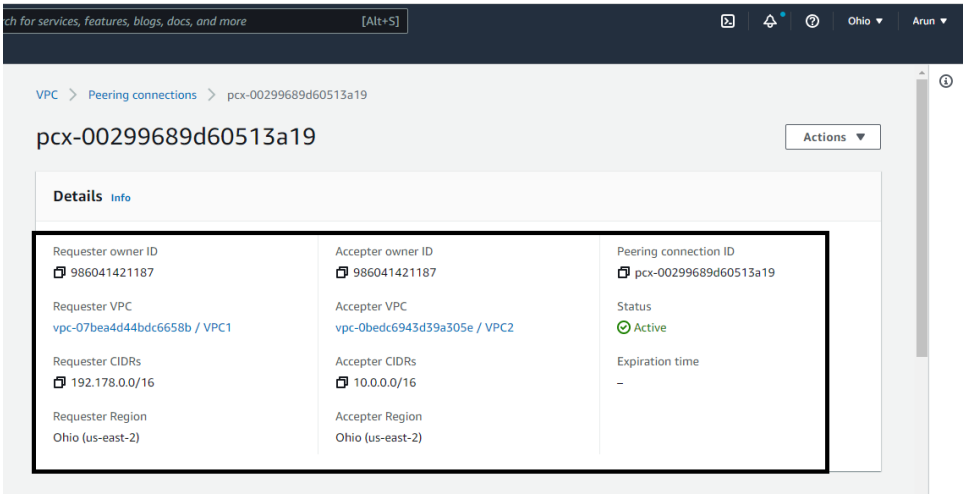
```
  load_balancer_type = "application"
  security_groups    = [aws_security_group.sg_vpc2.id]
  subnets            = [aws_subnet.vpc2_private1.id, aws_subnet.vpc2_private2.id]

  enable_deletion_protection = false


  tags = {
    Environment = "load balancer"
  }
}


resource "aws_alb_target_group" "group" {
  name     = "alb-target"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.vpc2.id
  stickiness {
    type = "lb_cookie"
  }
  # Alter the destination of the health check to be the login page.
  health_check {
    path = "/login"
    port = 80
  }
}
resource "aws_lb_target_group_attachment" "test" {
  target_group_arn = aws_alb_target_group.group.arn
  target_id        = aws_instance.ec2_Vpc2.id
  port             = 80
}
resource "aws_lb_target_group_attachment" "test2" {
  target_group_arn = aws_alb_target_group.group.arn
  target_id        = aws_instance.ec2_Vpc2-2.id
  port             = 80
}
# An example of a Listener
resource "aws_alb_listener" "my-alb-listener" {
  default_action {
    target_group_arn = aws_alb_target_group.group.arn
    type = "forward"
  }
  load_balancer_arn = aws_lb.alb.arn
  port = 80
  protocol = "HTTP"
}
```



Step 6 : Vpc peering

```
# Requester's side of the connection.
resource "aws_vpc_peering_connection" "peer" {
  vpc_id       = aws_vpc.vpc1.id
  peer_vpc_id   = aws_vpc.vpc2.id
  peer_owner_id = var.account_id
  peer_region   = var.region
  auto_accept   = false

  tags = {
    Side = "Requester"
  }
}

# Accepter's side of the connection.
resource "aws_vpc_peering_connection_accepter" "peer" {
  vpc_peering_connection_id = aws_vpc_peering_connection.peer.id
  auto_accept               = true

  tags = {
    Side = "Accepter"
  }
```

}


[Alt+S]
Ohio
Arun
VPC
Peering connections
pcx-00299689d60513a19
pcx-00299689d60513a19
Actions
Details
Info
Requester owner ID
986041421187
Accepter owner ID
986041421187
Peering connection ID
pcx-00299689d60513a19
Requester VPC
vpc-07bea4d44bdc6658b / VPC1
Accepter VPC
vpc-0bedc6943d39a305e / VPC2
Status
Active
Requester CIDRs
192.178.0.0/16
Accepter CIDRs
10.0.0.0/16
Expiration time
–
Requester Region
Ohio (us-east-2)
Accepter Region
Ohio (us-east-2)

Variables.tf

```
variable "access_key" {
   default = "AKIA6LFFGBWBXRXXEWI"
}
variable "secret_key" {
   default = "FjA+mHt0OZS7ofUn4JHkOtANr7hWdIqyjwclNzv"
}

variable "region" {
   default = "us-east-2"
}

variable "account_id" {
   default ="98604142187"
}
variable "ami" {
   default = "ami-064ff912f78e3e561"
}

variable "instance_type" {
   default = "t2.micro"
}
variable "key_name" {
   default ="ec2_key"
}
```