**Objective:**

**Create 2 vpcs and 2 subnets in one vpc and one in another vpc. All the subnets needs to be in different regions. Setup VPC peering connection to communicate between both VPC’s using private ip’s.Provision aws ec2 instances in each vpc and install apache2 and browse the instance ips to see the application.**

Created 2vpc’s each in different regions

creates 3subnet 2 in vpc1 and 1 in vpc2

created 2internet gateway

Create VPC Peering connection.

created 2routtable and added rout to internet gateway and peering connections of peer vpc’s

added subnet association to route table

Created security groups one for each vpc

Create Keypair for instances

Created 3 instances in each subnet

Code:

provider "aws" {

  alias = "ap-south-1"

  region = "ap-south-1"

}

provider "aws" {

  alias = "us-east-1"

  region = "us-east-1"

}

##VPC ap-south-1 config

resource "aws\_vpc" "vpc01" {

  provider = aws.ap-south-1

  cidr\_block = "10.0.0.0/16"

  tags = {

    Name = "VPC01"

  }

}

resource "aws\_subnet" "pubsub\_vpc01" {

  provider = aws.ap-south-1

  vpc\_id = aws\_vpc.vpc01.id

  cidr\_block = "10.0.1.0/24"

  tags = {

    Name = "Public Subnet"

  }

}

resource "aws\_subnet" "prvsub\_vpc01" {

  provider = aws.ap-south-1

  vpc\_id = aws\_vpc.vpc01.id

  cidr\_block = "10.0.2.0/24"

  tags = {

    Name = "Private Subnet"

  }

}

resource "aws\_internet\_gateway" "igwvpc01" {

  provider = aws.ap-south-1

  vpc\_id = aws\_vpc.vpc01.id

  tags = {

    Name = "igwvpc01"

  }

}

resource "aws\_route\_table" "pubroutevpc01" {

  provider = aws.ap-south-1

  vpc\_id = aws\_vpc.vpc01.id

  route {

    cidr\_block = "0.0.0.0/0"

    gateway\_id = aws\_internet\_gateway.igwvpc01.id

  }

  tags = {

    Name = "pubroutevpc01"

  }

}

resource "aws\_route\_table\_association" "routeassoc1" {

    provider = aws.ap-south-1

    subnet\_id = aws\_subnet.pubsub\_vpc01.id

    route\_table\_id = aws\_route\_table.pubroutevpc01.id

}

resource "aws\_eip" "nateipvpc01" {

  provider = aws.ap-south-1

  domain = "vpc"

}

resource "aws\_nat\_gateway" "natvpc01" {

  provider = aws.ap-south-1

  allocation\_id = aws\_eip.nateipvpc01.id

  subnet\_id = aws\_subnet.pubsub\_vpc01.id

  tags = {

    Name = "Nat VPC01"

  }

}

resource "aws\_route\_table" "prvroutevpc01" {

  provider = aws.ap-south-1

  vpc\_id = aws\_vpc.vpc01.id

  route {

    cidr\_block = "0.0.0.0/0"

    nat\_gateway\_id = aws\_nat\_gateway.natvpc01.id

  }

  tags = {

    Name = "prvroutevpc01"

  }

}

resource "aws\_route\_table\_association" "routeassoc2" {

  provider = aws.ap-south-1

  subnet\_id = aws\_subnet.prvsub\_vpc01.id

  route\_table\_id = aws\_route\_table.prvroutevpc01.id

}

##VPC us-east-1  config

resource "aws\_vpc" "vpc02" {

  provider = aws.us-east-1

  cidr\_block = "192.168.0.0/16"

  tags = {

    Name = "vpc02"

  }

}

resource "aws\_subnet" "pubsub\_vpc02" {

  provider = aws.us-east-1

  vpc\_id = aws\_vpc.vpc02.id

  cidr\_block = "192.168.1.0/24"

  tags = {

    Name = "Public Subnet"

  }

}

resource "aws\_subnet" "prvsub\_vpc02" {

  provider = aws.us-east-1

  vpc\_id = aws\_vpc.vpc02.id

  cidr\_block = "192.168.2.0/24"

  tags = {

    Name = "Private Subnet"

  }

}

resource "aws\_internet\_gateway" "igwvpc02" {

  provider = aws.us-east-1

  vpc\_id = aws\_vpc.vpc02.id

  tags = {

    Name = "igwvpc02"

  }

}

resource "aws\_route\_table" "pubroutevpc02" {

  provider = aws.us-east-1

  vpc\_id = aws\_vpc.vpc02.id

  route {

    cidr\_block = "0.0.0.0/0"

    gateway\_id = aws\_internet\_gateway.igwvpc02.id

  }

  tags = {

    Name = "pubroutevpc02"

  }

}

resource "aws\_route\_table\_association" "routeassoc3" {

    provider = aws.us-east-1

    subnet\_id = aws\_subnet.pubsub\_vpc02.id

    route\_table\_id = aws\_route\_table.pubroutevpc02.id

}

resource "aws\_eip" "nateipvpc02" {

  provider = aws.us-east-1

  domain = "vpc"

}

resource "aws\_nat\_gateway" "natvpc02" {

  provider = aws.us-east-1

  allocation\_id = aws\_eip.nateipvpc02.id

  subnet\_id = aws\_subnet.pubsub\_vpc02.id

  tags = {

    Name = "Nat vpc02"

  }

}

resource "aws\_route\_table" "prvroutevpc02" {

  provider = aws.us-east-1

  vpc\_id = aws\_vpc.vpc02.id

  route {

    cidr\_block = "0.0.0.0/0"

    nat\_gateway\_id = aws\_nat\_gateway.natvpc02.id

  }

  tags = {

    Name = "prvroutevpc02"

  }

}

resource "aws\_route\_table\_association" "routeassoc4" {

  provider = aws.us-east-1

  subnet\_id = aws\_subnet.prvsub\_vpc02.id

  route\_table\_id = aws\_route\_table.prvroutevpc02.id

}

## VPC Peering between ap-south-1 and us-east-1

resource "aws\_vpc\_peering\_connection" "peer" {

    provider = aws.ap-south-1

    vpc\_id = aws\_vpc.vpc01.id

    peer\_vpc\_id = aws\_vpc.vpc02.id

    peer\_region = "us-east-1"

    auto\_accept = false

    tags = {

      Name = "vpc01-vpc02"

      Side = "Requester"

    }

}

resource "aws\_vpc\_peering\_connection\_accepter" "peer\_accept" {

    provider = aws.us-east-1

    vpc\_peering\_connection\_id = aws\_vpc\_peering\_connection.peer.id

    auto\_accept = true

    tags = {

      Name = "vpc01-vpc02"

      Side = "Accepter"

    }

}

## Add the vpc peering to the route tables

resource "aws\_route" "r1" {

  provider = aws.ap-south-1

  route\_table\_id = aws\_route\_table.pubroutevpc01.id

  destination\_cidr\_block = aws\_vpc.vpc02.cidr\_block

  vpc\_peering\_connection\_id = aws\_vpc\_peering\_connection.peer.id

}

resource "aws\_route" "r2" {

  provider = aws.ap-south-1

  route\_table\_id = aws\_route\_table.prvroutevpc01.id

  destination\_cidr\_block = aws\_vpc.vpc02.cidr\_block

  vpc\_peering\_connection\_id = aws\_vpc\_peering\_connection.peer.id

}

resource "aws\_route" "r3" {

  provider = aws.us-east-1

  route\_table\_id = aws\_route\_table.pubroutevpc02.id

  destination\_cidr\_block = aws\_vpc.vpc01.cidr\_block

  vpc\_peering\_connection\_id = aws\_vpc\_peering\_connection.peer.id

}

resource "aws\_route" "r4" {

  provider = aws.us-east-1

  route\_table\_id = aws\_route\_table.prvroutevpc02.id

  destination\_cidr\_block = aws\_vpc.vpc01.cidr\_block

  vpc\_peering\_connection\_id = aws\_vpc\_peering\_connection.peer.id

}

#Generate keypair for instances

resource "aws\_key\_pair" "keypairmum" {

  provider = aws.ap-south-1

  key\_name = "terraform-key"

  public\_key = file("C:/Users/Vijesh/.ssh/id\_rsa.pub")

}

resource "aws\_key\_pair" "keypairus" {

  provider = aws.us-east-1

  key\_name = "terraform-key"

  public\_key = file("C:/Users/Vijesh/.ssh/id\_rsa.pub")

}

# Security group vpc1

resource "aws\_security\_group" "sgvpc01" {

  provider = aws.ap-south-1

  name = "web"

  vpc\_id = aws\_vpc.vpc01.id

  ingress {

    description = "HTTP from VPC"

    from\_port   = 80

    to\_port     = 80

    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 = "HTTPS"

    from\_port   = 443

    to\_port     = 443

    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 = "sgvpc01"

  }

}

# Security groups vpc2

resource "aws\_security\_group" "sgvpc02" {

  provider = aws.us-east-1

  name = "sgvpc02"

  vpc\_id = aws\_vpc.vpc02.id

  ingress {

    description = "HTTP from VPC"

    from\_port   = 80

    to\_port     = 80

    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 = "HTTPS"

    from\_port   = 443

    to\_port     = 443

    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 = "sgvpc02"

  }

}

## Creating instances 1 public bastion and  2 private instances in each private subnets

resource "aws\_instance" "instance1vpc01" {

  provider = aws.us-east-1

  ami = "ami-07d9b9ddc6cd8dd30"

  instance\_type = "t2.micro"

  subnet\_id = aws\_subnet.pubsub\_vpc02.id

  key\_name = aws\_key\_pair.keypairus.key\_name

  vpc\_security\_group\_ids = [aws\_security\_group.sgvpc02.id]

  associate\_public\_ip\_address = true

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

  tags = {

    Name = "instance1vpc01"

  }

}

resource "aws\_instance" "instance2vpc01" {

  provider = aws.us-east-1

  ami = "ami-07d9b9ddc6cd8dd30"

  instance\_type = "t2.micro"

  subnet\_id = aws\_subnet.prvsub\_vpc02.id

  key\_name = aws\_key\_pair.keypairus.key\_name

  vpc\_security\_group\_ids = [aws\_security\_group.sgvpc02.id]

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

  tags = {

    Name = "instance2vpc01"

  }

}

resource "aws\_instance" "instance1vpc02" {

  provider = aws.ap-south-1

  ami = "ami-03bb6d83c60fc5f7c"

  instance\_type = "t2.micro"

  subnet\_id = aws\_subnet.prvsub\_vpc01.id

  key\_name = aws\_key\_pair.keypairmum.key\_name

  vpc\_security\_group\_ids = [aws\_security\_group.sgvpc01.id]

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

  tags = {

    Name = "instance1vpc02"

  }

}

Terraform init

Terraform validate

terraform plan -out demo.tfplan

A screenshot of a computer program

Description automatically generated

terraform apply "demo.tfplan"

A screen shot of a computer

Description automatically generated

A screenshot of a computer

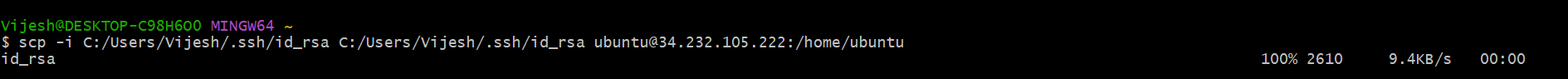
Description automatically generated

A screenshot of a computer

Description automatically generated

Now we will connect to the bastion server in virginia which is in public domain and then login to the private subnet of Virginia and try to connect to the private instance in Mumbai.

First copy the key from local to public instance:

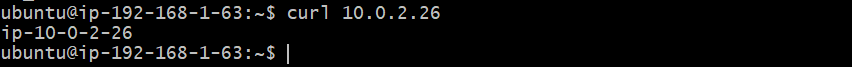


Login to the public instance:

A screenshot of a computer program

Description automatically generated

From public instance of virginia try to curl the private ip of the instance in Mumbai:



Now get into the private instance of virginia and curl to private instance of Mumbai:

A screenshot of a computer screen

Description automatically generated

A black background with white text

Description automatically generated

Hence the applications are able to connect to each other using private instances using VPC Peering.