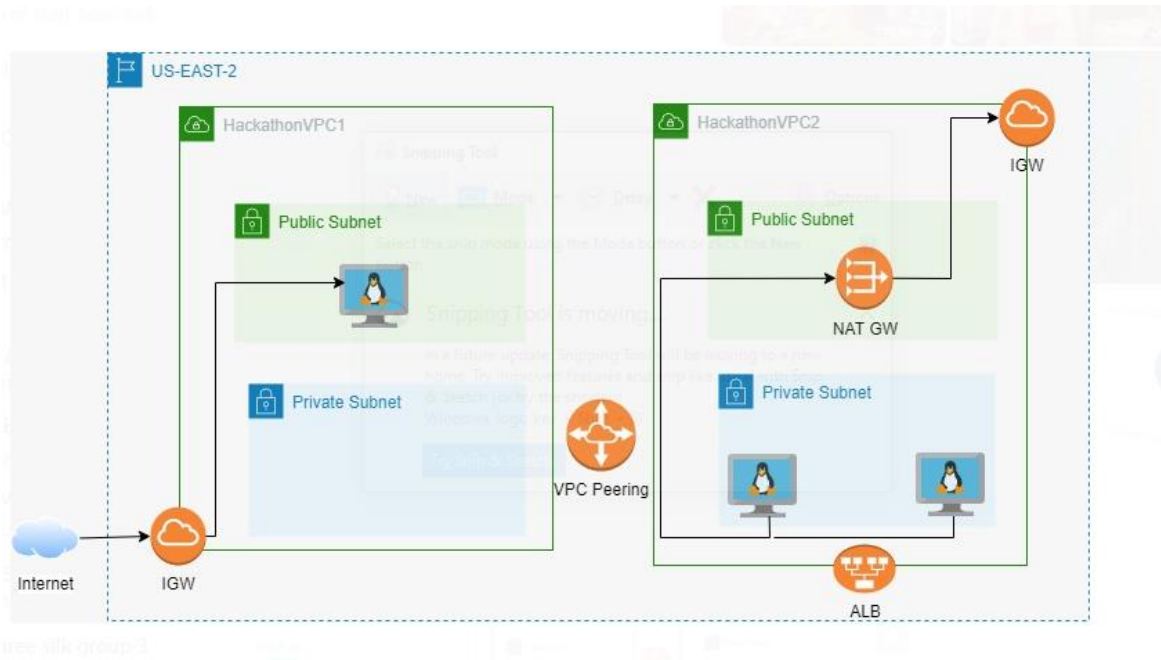


Creating the below AWS architecture using terraform



Step 1 : Create VPC1 and Public subnet associate route 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 association 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, associate 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 association 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
}

```

Search for services, features, blogs, docs, and more [Alt+S]

Ohio Arun

Your VPCs (3) Info

Filter VPCs

<input type="checkbox"/>	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
<input type="checkbox"/>	-	vpc-08aa89a57cf0ce092	Available	172.31.0.0/16	-
<input type="checkbox"/>	VPC2	vpc-0bedc6943d39a305e	Available	10.0.0.0/16	-
<input type="checkbox"/>	VPC1	vpc-07bea4d44bdc6658b	Available	192.178.0.0/16	-

Select a VPC above

Search for services, features, blogs, docs, and more [Alt+S]

Ohio Arun

Subnets (6) Info

Filter subnets

<input type="checkbox"/>	Name	Subnet ID	State	VPC	IPv4 CIDR
<input type="checkbox"/>	-	subnet-0a7e0cc51d239cc3d	Available	vpc-08aa89a57cf0ce092	172.31.16.0/20
<input type="checkbox"/>	-	subnet-047c73811dab8e4a1	Available	vpc-08aa89a57cf0ce092	172.31.0.0/20
<input type="checkbox"/>	-	subnet-040e363f86445142f	Available	vpc-08aa89a57cf0ce092	172.31.32.0/20
<input type="checkbox"/>	private_1	subnet-0e25cfa7b8c0720c5	Available	vpc-0bedc6943d39a305e VPC2	10.0.2.0/24
<input type="checkbox"/>	public_1	subnet-0bac2a486f17065ab	Available	vpc-0bedc6943d39a305e VPC2	10.0.1.0/24
<input type="checkbox"/>	public_1	subnet-054c8bac8ac120262	Available	vpc-07bea4d44bdc6658b VPC1	192.178.1.0/24

Select a subnet

Search for services, features, blogs, docs, and more

[Alt+S]

Ohio

Arun

Route tables (6)

Info

Filter route tables

1

>

⚙️

<input type="checkbox"/>	Name	Route table ID	Explicit route table	Edge route table	Main route table	VPC	Owner ID
<input type="checkbox"/>	-	rtb-077cd...	-	-	Yes	vpc-07bea4d44bdc6658b VPC1	98604142...
<input type="checkbox"/>	-	rtb-04888...	-	-	Yes	vpc-0bedc6943d39a305e VPC2	98604142...
<input type="checkbox"/>	-	rtb-04183...	-	-	Yes	vpc-08aa89a57cf0ce092	98604142...
<input type="checkbox"/>	private-route-table	rtb-0ab84...	subnet-0e25cf...	-	No	vpc-0bedc6943d39a305e VPC2	98604142...
<input type="checkbox"/>	public-route-table	rtb-05d90...	subnet-0bac2...	-	No	vpc-0bedc6943d39a305e VPC2	98604142...
<input type="checkbox"/>	public-route-table	rtb-0af49...	subnet-054c8...	-	No	vpc-07bea4d44bdc6658b VPC1	98604142...

Select a route table

Internet gateways (3)

Info

Filter internet gateways

1

>

⚙️

<input type="checkbox"/>	Name	Internet gateway ID	State	VPC ID	Owner
<input type="checkbox"/>	-	igw-04b2d6d5e48847d80	Attached	vpc-08aa89a57cf0ce092	98604142118
<input type="checkbox"/>	internet-gateway-v...	igw-07080e4d137f6ce25	Attached	vpc-0bedc6943d39a305e VPC2	98604142118
<input type="checkbox"/>	internet-gateway-v...	igw-09ae60d6fc216a9a8	Attached	vpc-07bea4d44bdc6658b VPC1	98604142118

Select an internet gateway above

NAT gateways (1/1)

Info

Filter NAT gateways

1

>

⚙️

<input type="checkbox"/>	Name	NAT gateway ID	Connectivity	State	State message	Elastic IP address	Priority
<input checked="" type="checkbox"/>	-	nat-0f1242aef8917d2ad	Public	Available	-	3.18.181.182	10

Elastic IP addresses (1/1)

Info

Filter Elastic IP addresses

1

>

⚙️

<input checked="" type="checkbox"/>	Name	Allocated IPv4 address	Type	Allocation ID	Reverse DNS record	Associated
<input checked="" type="checkbox"/>	-	3.18.181.182	Public IP	eipalloc-015015ceb888ca252	-	-

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"
  }
}

```

th for services, features, blogs, docs, and more [Alt+S]

Instances (3) Info

Search

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	Ec2_vpc1	i-06ab41f886b95a6c5	Running	t2.micro	2/2 checks passed	No alarms	us-east-2a
<input type="checkbox"/>	Ec2_vpc2	i-01daf5ef623967be8	Running	t2.micro	2/2 checks passed	No alarms	us-east-2a
<input type="checkbox"/>	Ec2_vpc2	i-0ab43efb05f1c0dbf	Running	t2.micro	2/2 checks passed	No alarms	us-east-2a

Select an instance

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"
}
```

h for services, features, blogs, docs, and more [Alt+S]

Create Load Balancer Actions

Filter by tags and attributes or search by keyword |< < 1 to 1 of 1 > >|

Name	DNS name	State	VPC ID	Availability Zones	Type
alb	alb-1357377402.us-east-2.el...	Provisioning	vpc-0bedc6943d39a305e	us-east-2b, us-east-2a	application

h for services, features, blogs, docs, and more [Alt+S]

EC2 > Target groups > alb-target

alb-target

arn:aws:elasticloadbalancing:us-east-2:986041421187:targetgroup/alb-target/93b91876f0255348

Actions

Details

Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-0bedc6943d39a305e
IP address type IPv4	Load balancer alb		
Total targets 2	Healthy 0	Unhealthy 0	Unused 0
	Initial 2		Draining 0

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"
}
```

Acceptor'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 = "Acceptor"
}
```

}

ch for services, features, blogs, docs, and more

[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-07bea4d444bdc6658b / 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 = "AKIA6LFFGBWBXRXEWI"  
}  
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"  
}
```