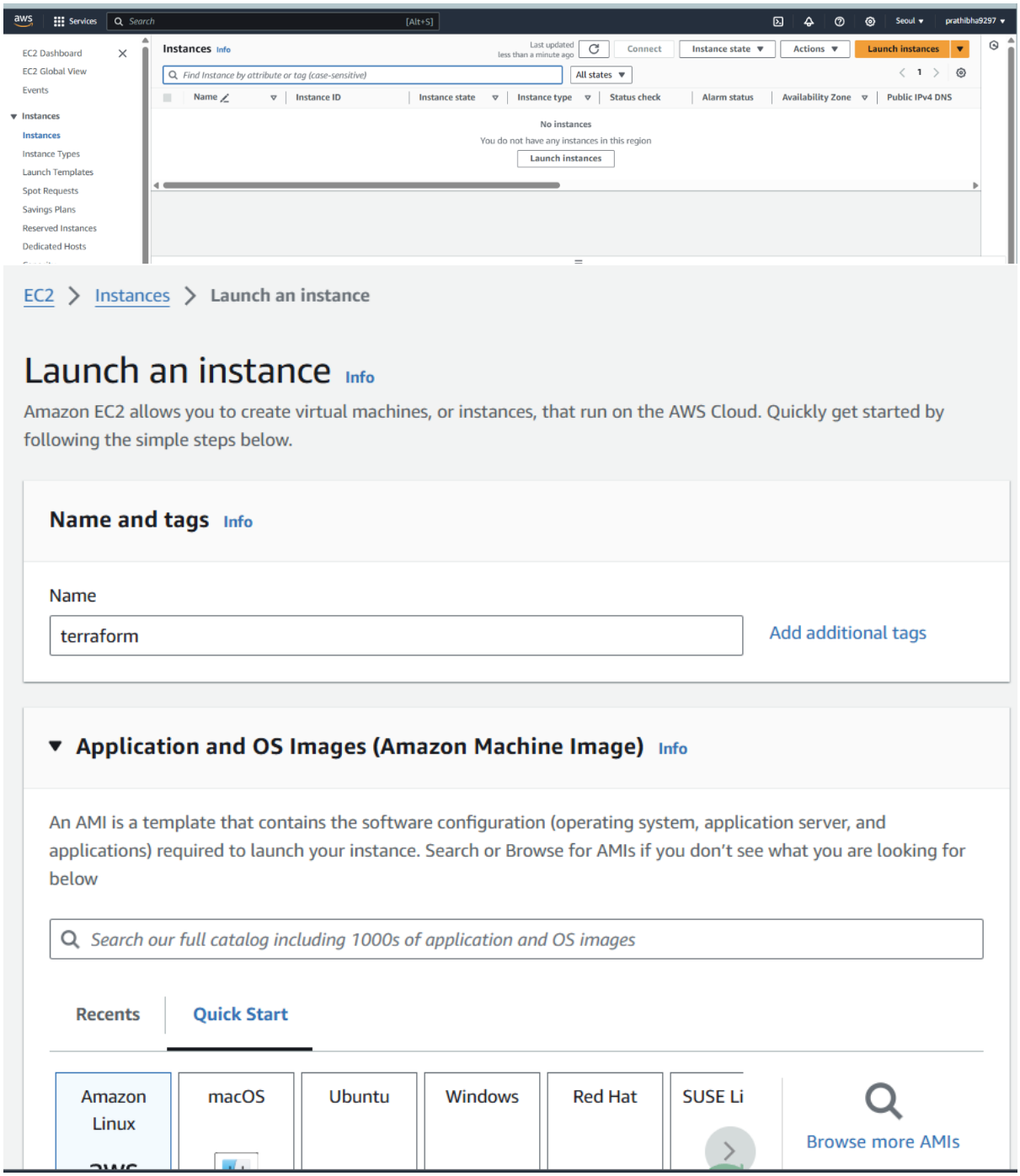
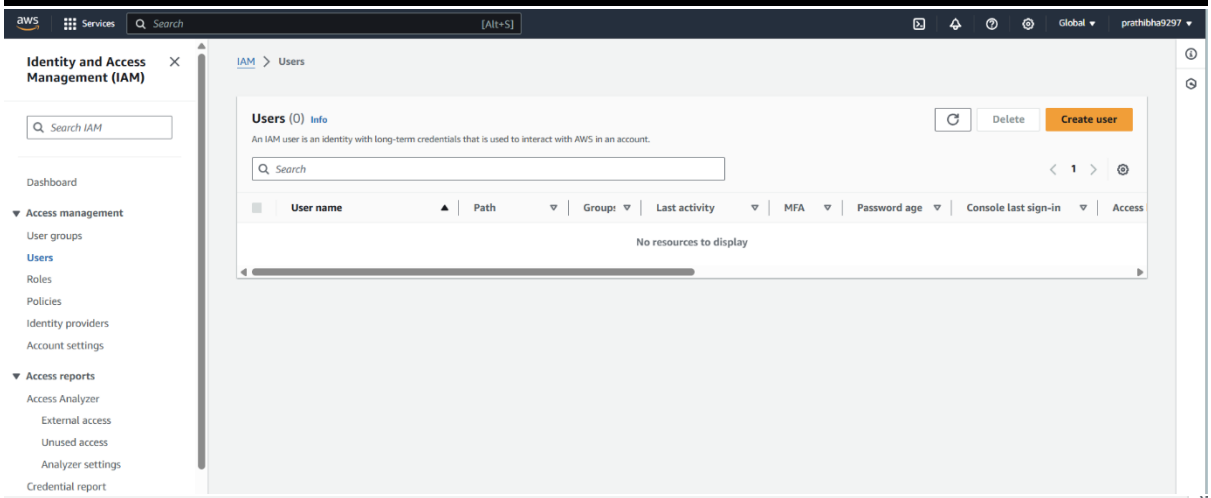
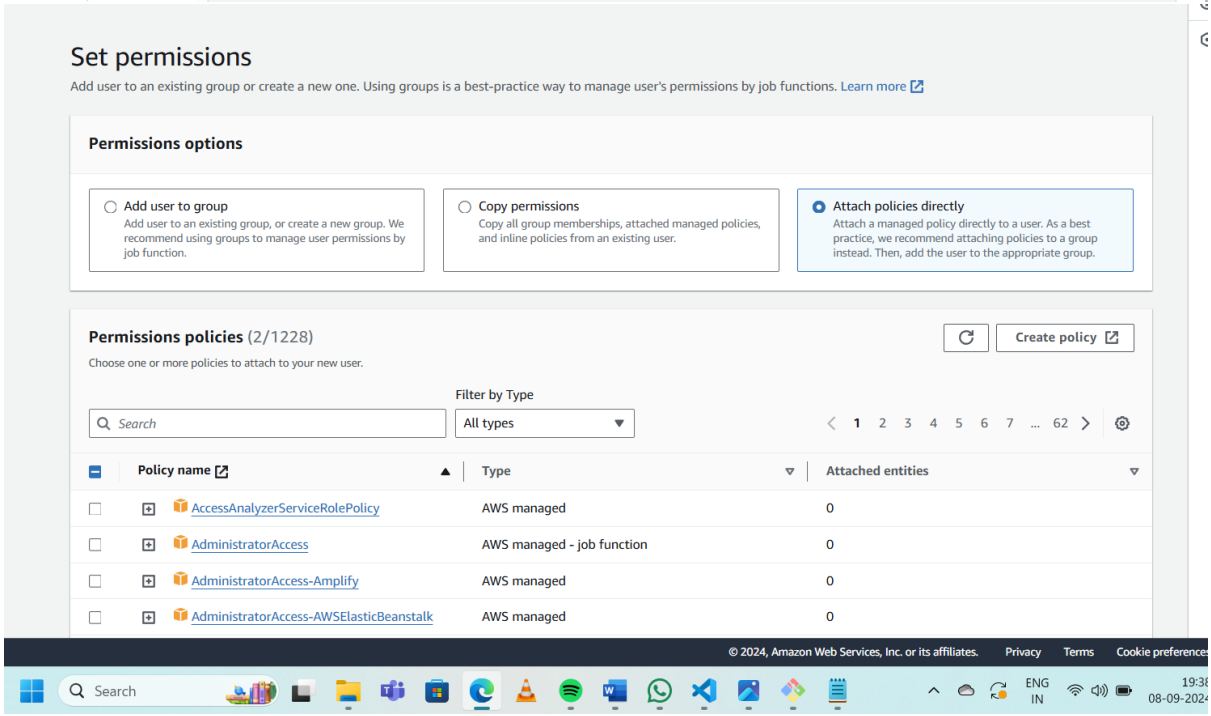
VPC PEERING USING TERRAFORM

- Open aws console and go to seoul region
- Create EC2 instance
- Connect to that EC2 instance in git bash
- Go to IAM service and create an user ‘terra’ and then create access key as shown in below slides



```
prath@Chinni MINGW64 ~/OneDrive/Desktop
$ ssh -i "project.pem" ec2-user@ec2-18-212-87-129.compute-1.amazonaws.com
The authenticity of host 'ec2-18-212-87-129.compute-1.amazonaws.com (18.212.87.129)' can't be established.
ED25519 key fingerprint is SHA256:QV3T4OK3vLdKUiwrw2QtTl1zxgDfBxjqPcOl86IdCdQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-212-87-129.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
       #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-26-237 ~]$ |
```





Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name	Console password type	Require password reset
terra	None	No

Permissions summary

Name	Type	Used as
AmazonEC2FullAccess	AWS managed	Permissions policy
AmazonVPCFullAccess	AWS managed	Permissions policy

Tags - *optional*

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel | Previous | Create user

IAM > Users

Users (1) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Delete | Create user

Search

User name	Path	Groups	Last activity	MFA	Password age	Console last sign-in	Access
terra	/	0	-	-	-	-	-

IAM > Users > terra

terra Info

Delete

Summary

ARN	Console access	Access key 1
arn:aws:iam::891377141179:user/terra	Disabled	Create access key
Created	Last console sign-in	
September 08, 2024, 19:38 (UTC+05:30)	-	

Permissions | Groups | Tags | Security credentials | Access Advisor

Permissions policies (2)

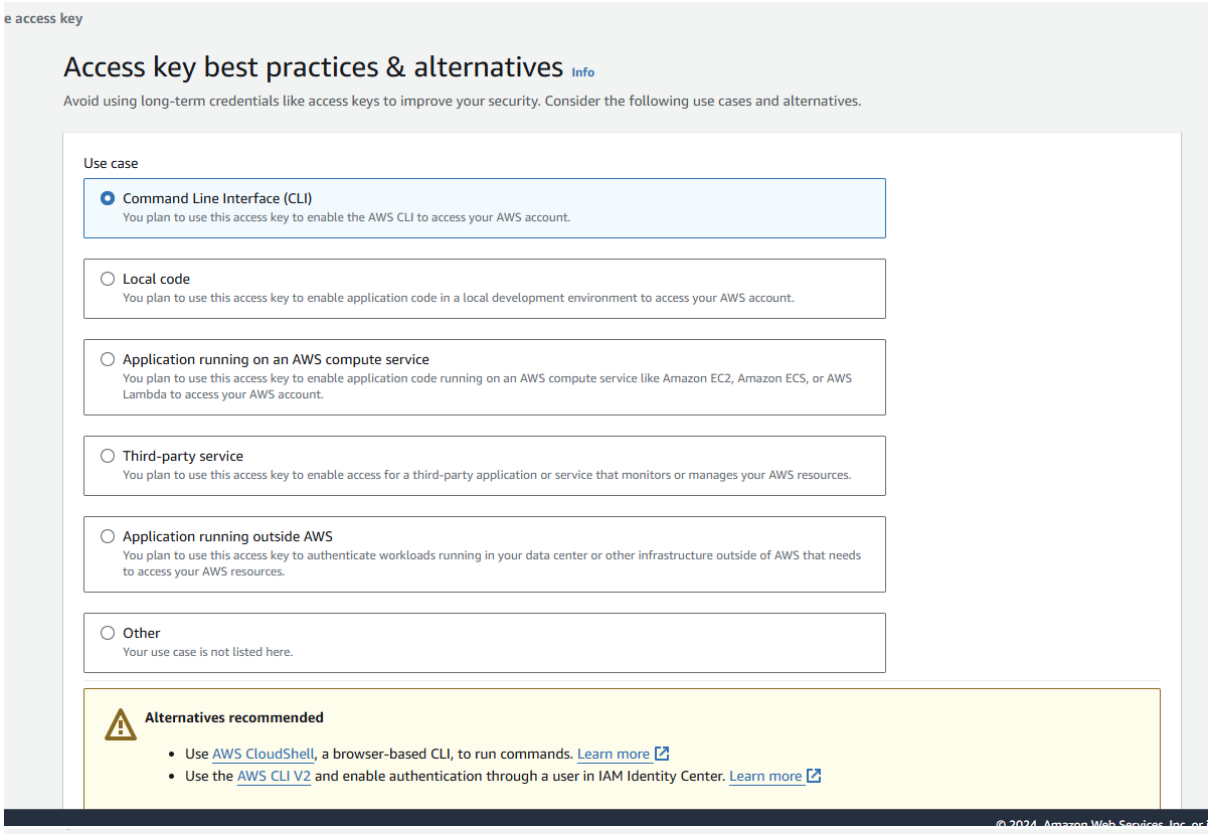
Permissions are defined by policies attached to the user directly or through groups.

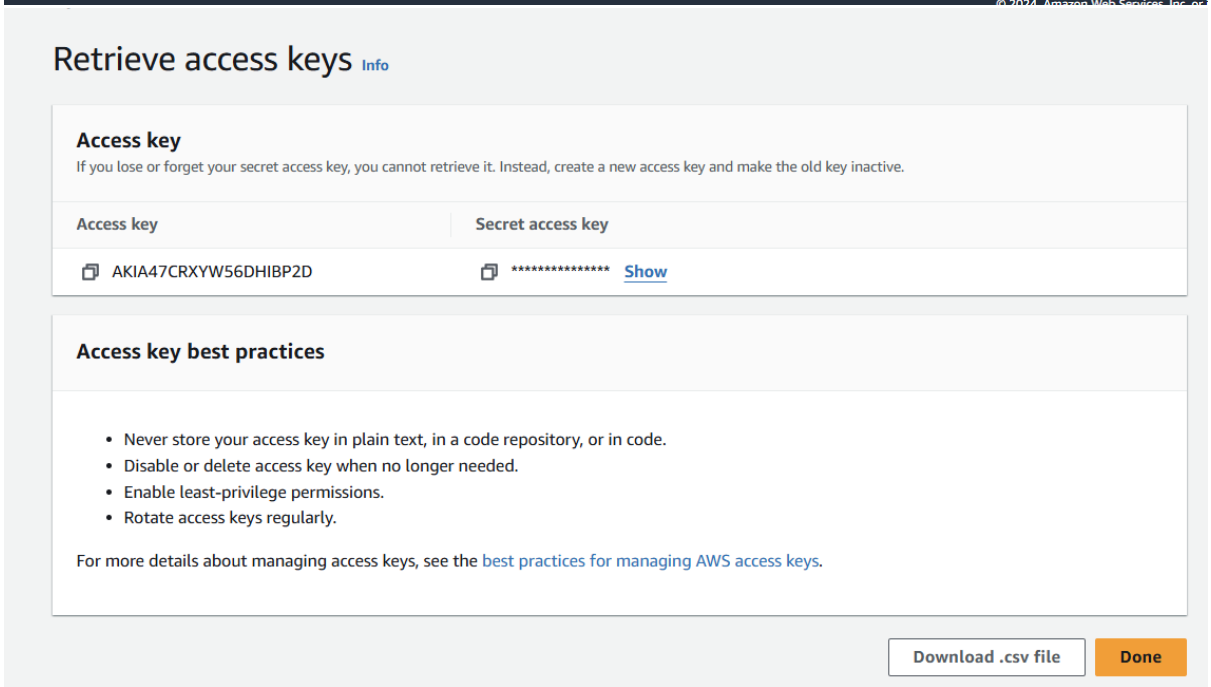
Remove | Add permissions

Filter by Type

Search | All types

Policy name	Type	Attached via
AmazonEC2FullAccess	AWS managed	Directly
AmazonVPCFullAccess	AWS managed	Directly





- Go to EC2 server and configure the access key
- Command for configure: aws configure
- Enter the access key, secret access key and region code
- Install the AWS CLI and Terraform
- Create directory – mkdir terraform
- Change to that directory – cd terraform
- In terraform directory create 3 block i.e, 1.terrafromblock.tf, 2.provider.tf, 3.resource.tf

1.terraformblock.tf:

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.65.0"
    }
  }
}
```

2.provider.tf:

```
provider "aws" {
    region = "us-east-1"
    profile = "default"
}
```

3.resource.tf:

```
# Create VPC 1
resource "aws_vpc" "vpc1" {
  cidr_block = "10.0.0.0/16"
  tags = {
    Name = "VPC1"
  }
}

# Create Subnet for VPC 1
resource "aws_subnet" "subnet1" {
  vpc_id                  = aws_vpc.vpc1.id
  cidr_block              = "10.0.1.0/24"
  availability_zone       = "us-east-1a"
  map_public_ip_on_launch = true
  tags = {
    Name = "Subnet1"
```

```
  }
}

# Create Internet Gateway for VPC 1
resource "aws_internet_gateway" "igw1" {
  vpc_id = aws_vpc.vpc1.id
  tags = {
    Name = "IGW1"
  }
}

# Create Route Table for VPC 1 and associate with the subnet
resource "aws_route_table" "rt1" {
  vpc_id = aws_vpc.vpc1.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw1.id
  }
}

resource "aws_route_table_association" "rta1" {
  subnet_id      = aws_subnet.subnet1.id
  route_table_id = aws_route_table.rt1.id
}

# Create Security Group for VPC 1
resource "aws_security_group" "sg1" {
  name   = "vpc1-sg"
  vpc_id = aws_vpc.vpc1.id

  ingress {
```

```
    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"]
  }
}
# Create EC2 Instance in VPC 1
resource "aws_instance" "instance1" {
  ami                         = "ami-0e86e20dae9224db8" # Replace with a valid AMI ID
  instance_type               = "t2.micro"
  subnet_id                   = aws_subnet.subnet1.id
  key_name                    = "project"  # Replace with your AWS key pair name
  vpc_security_group_ids      = [aws_security_group.sg1.id]
  associate_public_ip_address = true
 tags = {
    Name = "Instance1"
  }
}
# Create VPC 2
resource "aws_vpc" "vpc2" {
  cidr_block = "10.1.0.0/16"
  tags = {
    Name = "VPC2"
  }
```

```
}
# Create Subnet for VPC 2
resource "aws_subnet" "subnet2" {
  vpc_id                  = aws_vpc.vpc2.id
  cidr_block              = "10.1.1.0/24"
  availability_zone       = "us-east-1b"
  map_public_ip_on_launch = true
  tags = {
    Name = "Subnet2"
  }
}
# Create Internet Gateway for VPC 2
resource "aws_internet_gateway" "igw2" {
  vpc_id = aws_vpc.vpc2.id
  tags = {
    Name = "IGW2"
  }
}
# Create Route Table for VPC 2 and associate with the subnet
resource "aws_route_table" "rt2" {
  vpc_id = aws_vpc.vpc2.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw2.id
  }
}
resource "aws_route_table_association" "rta2" {
  subnet_id      = aws_subnet.subnet2.id
  route_table_id = aws_route_table.rt2.id
}
```

```
# Create Security Group for VPC 2
resource "aws_security_group" "sg2" {
  name   = "vpc2-sg"
  vpc_id = aws_vpc.vpc2.id
 ingress {
    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"]
  }
}
# Create EC2 Instance in VPC 2
resource "aws_instance" "instance2" {
  ami                         = "ami-0e86e20dae9224db8" # Replace with a valid AMI ID
  instance_type               = "t2.micro"
  subnet_id                   = aws_subnet.subnet2.id
  key_name                    = "project"  # Replace with your AWS key pair name
  vpc_security_group_ids      = [aws_security_group.sg2.id]
  associate_public_ip_address = true
tags = {
    Name = "Instance2"
  }
}
# Create VPC Peering Connection
resource "aws_vpc_peering_connection" "peer" {
```

```
  vpc_id        = aws_vpc.vpc1.id
  peer_vpc_id   = aws_vpc.vpc2.id
  auto_accept   = true
  tags = {
    Name = "VPC1-to-VPC2-Peering"
  }
}
# Update Route Tables to include Peering Connection
resource "aws_route" "route_to_vpc2" {
  route_table_id         = aws_route_table.rt1.id
  destination_cidr_block = aws_vpc.vpc2.cidr_block
  vpc_peering_connection_id = aws_vpc_peering_connection.peer.id
}

resource "aws_route" "route_to_vpc1" {
  route_table_id         = aws_route_table.rt2.id
  destination_cidr_block = aws_vpc.vpc1.cidr_block
  vpc_peering_connection_id = aws_vpc_peering_connection.peer.id
}
```

```
root@ip-172-31-26-237:~
[root@ip-172-31-26-237 ~]# aws configure
AWS Access Key ID [None]: AKIA47CRXYW56DHIBP2D
AWS Secret Access Key [None]: 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
Default region name [None]: table
```

```
root@ip-172-31-26-237:~/.aws
[root@ip-172-31-26-237 ~]# cd .aws
[root@ip-172-31-26-237 .aws]# ls
config  credentials
[root@ip-172-31-26-237 .aws]# cat credentials
[default]
aws_access_key_id = AKIA47CRXYW56DHIBP2D
aws_secret_access_key = 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
[root@ip-172-31-26-237 .aws]#
```

```
[root@ip-172-31-26-237 ~]# cd terraform
[root@ip-172-31-26-237 terraform]# vi terraformblock.tf
[root@ip-172-31-26-237 terraform]# vi provider.tf
[root@ip-172-31-26-237 terraform]# vi resource.tf
[root@ip-172-31-26-237 terraform]# vi userdata.sh
[root@ip-172-31-26-237 terraform]# ls
provider.tf  resource.tf  terraformblock.tf  userdata.sh
[root@ip-172-31-26-237 terraform]#
```

- Once blocks created give command-terraform init
- Terraform validate
- Terraform plan
- Terraform apply
- After terraform apply 17 resouce will add as shown in below figure

```
      + requester (known after apply)
    }

Plan: 17 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_vpc.vpc2: Creating...
aws_vpc.vpc1: Creating...
aws_vpc.vpc1: Creation complete after 1s [id=vpc-04f40dd44c82202a9]
aws_vpc.vpc2: Creation complete after 1s [id=vpc-01dcf746567da5690]
aws_subnet.subnet1: Creating...
aws_internet_gateway.igw1: Creating...
aws_security_group.sg1: Creating...
aws_security_group.sg2: Creating...
aws_vpc_peering_connection.peer: Creating...
aws_internet_gateway.igw2: Creating...
aws_subnet.subnet2: Creating...
aws_internet_gateway.igw1: Creation complete after 0s [id=igw-09833c987a9f9b0b3]
aws_route_table.rt1: Creating...
aws_internet_gateway.igw2: Creation complete after 0s [id=igw-07c692b1f8565c099]
aws_route_table.rt2: Creating...
aws_vpc_peering_connection.peer: Creation complete after 1s [id=pcx-0d06002cb3ebfef58]
aws_route_table.rt1: Creation complete after 1s [id=rtb-07d7bd735471c5add]
aws_route.route_to_vpc2: Creating...
aws_route_table.rt2: Creation complete after 1s [id=rtb-083ce9f3a207544c6]
aws_route.route_to_vpc1: Creating...
aws_route.route_to_vpc1: Creation complete after 0s [id=r-rtb-083ce9f3a207544c6179966490]
aws_security_group.sg1: Creation complete after 2s [id=sg-0b9ebcea47efcefad]
aws_security_group.sg2: Creation complete after 2s [id=sg-03ce024d8d8cf29ad]
aws_route.route_to_vpc2: Creation complete after 1s [id=r-rtb-07d7bd735471c5add3322942084]
aws_subnet.subnet1: Still creating... [10s elapsed]
aws_subnet.subnet2: Still creating... [10s elapsed]
aws_subnet.subnet1: Creation complete after 11s [id=subnet-081891b64a575e852]
aws_route_table_association.rta1: Creating...
aws_instance.instance1: Creating...
aws_subnet.subnet2: Creation complete after 11s [id=subnet-005fd0f688db9cb54]
aws_route_table_association.rta2: Creating...
aws_instance.instance2: Creating...
aws_route_table_association.rta2: Creation complete after 0s [id=rtbassoc-058082127bcb43e32]
aws_route_table_association.rta1: Creation complete after 0s [id=rtbassoc-036fe4e0886febf0f]
aws_instance.instance1: Still creating... [10s elapsed]
aws_instance.instance2: Still creating... [10s elapsed]
aws_instance.instance1: Still creating... [20s elapsed]
aws_instance.instance2: Still creating... [20s elapsed]
aws_instance.instance1: Still creating... [30s elapsed]
aws_instance.instance2: Still creating... [30s elapsed]
aws_instance.instance2: Creation complete after 32s [id=i-04fa7bea7bff29939]
aws_instance.instance1: Still creating... [40s elapsed]
aws_instance.instance1: Creation complete after 42s [id=i-0ec6c3184205b6410]

Apply complete! Resources: 17 added, 0 changed, 0 destroyed.
root@ip-172-31-17-248:~/terraform#
```

- Go to aws console and check resources created or not
- Once created open the instances and connect to the server and check both instances are peering or not by using curl command as shown in below figure

```
root@ip-10-0-1-94:/var/www/html# rm index.html
root@ip-10-0-1-94:/var/www/html# vi index.html
root@ip-10-0-1-94:/var/www/html# systemctl restart apache2
root@ip-10-0-1-94:/var/www/html# curl 10.0.1.94:80
hello this is webserver 1
root@ip-10-0-1-94:/var/www/html#
```

i-02eee3e7708c52cce (Instance1)

PublicIPs: 3.80.230.62 PrivateIPs: 10.0.1.94

aws Services Search

```
root@ip-10-1-1-68:/var/www/html# curl 10.1.1.68:80
hello this is from webserver2
root@ip-10-1-1-68:/var/www/html#
```

ASUS Software Port... MyASUS Software -... McAfee LiveSafe Gmail

aws Services Search

```
root@ip-10-0-1-248:~# cd /var/www/html
root@ip-10-0-1-248:/var/www/html# ls
index.html
root@ip-10-0-1-248:/var/www/html# rm index.html
root@ip-10-0-1-248:/var/www/html# vi index.html
root@ip-10-0-1-248:/var/www/html# systemctl restart apache2
root@ip-10-0-1-248:/var/www/html# curl 10.0.1.248
hello this is from website1
root@ip-10-0-1-248:/var/www/html#
```

aws Services Search

```
root@ip-10-1-1-217:~# cd /var/www/html
root@ip-10-1-1-217:/var/www/html# ls
index.html
root@ip-10-1-1-217:/var/www/html# rm index.html
root@ip-10-1-1-217:/var/www/html# vi index.html
root@ip-10-1-1-217:/var/www/html# systemctl restart apache2
root@ip-10-1-1-217:/var/www/html# curl 10.1.1.217:80
hello this is from webserver 2
root@ip-10-1-1-217:/var/www/html#
```

aws Services Search

```
root@ip-10-0-1-248:/var/www/html# curl 10.0.1.248
hello this is from website1
root@ip-10-0-1-248:/var/www/html# curl 10.1.1.217:80
hello this is from webserver 2
root@ip-10-0-1-248:/var/www/html#
```

`

```
root@ip-10-1-1-217:~# cd /var/www/html
root@ip-10-1-1-217:/var/www/html# ls
index.html
root@ip-10-1-1-217:/var/www/html# rm index.html
root@ip-10-1-1-217:/var/www/html# vi index.html
root@ip-10-1-1-217:/var/www/html# systemctl restart apache2
root@ip-10-1-1-217:/var/www/html# curl 10.1.1.217:80
hello this is from webserver 2
root@ip-10-1-1-217:/var/www/html# curl 10.0.1.248:80
hello this is from website1
root@ip-10-1-1-217:/var/www/html#
```

- As shown in above slide we can able to connect from one instance to another so vpc peering was successfully done