Q : Multiple users accessing the module

main .tf

```
module "Employee1" {
  source = "./Ec2_module"
}

module "Employee2" {
  source = "./Ec2_module"
}

module "Employee3" {
  source = "./Ec2_module"
}

module "Employee4" {
  source = "./Ec2_module"
}

module "Employee5" {
  source = "./Ec2_module"
}
```

Outputs.tf

```
output "instance_ip_address1" {
 description = "Public Ip of the instance"
 value       = module.Employee1.instance_public_ip
}

output "instance_ip_address2" {
 description = "Private Ip of the instance"
 value       = module.Employee2.instance_private_ip
}

output "instance_ip_address3" {
 description = "Public Ip of the instance"
 value       = module.Employee3.instance_public_ip
}
output "instance_ip_address4" {
 description = "Private Ip of the instance"
 value       = module.Employee4.instance_private_ip
}

output "instance_ip_address5" {
 description = "Public Ip of the instance"
 value       = module.Employee5.instance_public_ip
}
```

Ec2_module

main.tf

```
data "aws_ami" "ubuntu" {
  most_recent = true

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }

  owners = ["099720109477"] # Canonical
}

resource "aws_instance" "bala" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t2.micro"

  tags = {
    Name = "HelloWorld"
  }
}


resource "aws_instance" "champu" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t2.micro"

  tags = {
    Name = "palguni"
  }
}
```

Outputs.tf

```
output "instance_id" {
  description = "ID of the EC2 instance"
  value       = aws_instance.bala.id
}

output "instance_public_ip" {
  description = "Public IP address of the EC2 instance"
  value       = aws_instance.bala.public_ip
}

output "instance_private_ip" {
  description = "Private IP address of the EC2 instance"
  value       = aws_instance.champu.private_ip
}
```

Provider.tf

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.9.0"
    }
  }
}
```

Q : Ec2 Instance creation, S3 bucket and vpc creation

main .tf

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.9.0"
    }
  }
}


data "aws_ami" "ubuntu" {
  most_recent = true

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }

  owners = ["099720109477"] # Canonical
}

resource "aws_instance" "web" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t2.micro"

  tags = {
    Name = "HelloWorld"
  }
}
```

```
provider "aws" {
  region = "ap-south-1"  # Replace this with your desired AWS region
}

resource "aws_s3_bucket" "my_bucket" {
  bucket = "chopala123"  # Replace this with a unique bucket name of your choice

  # Additional optional configurations:
  force_destroy = true  # Setting this to true allows Terraform to destroy the bucket on
delete

  # Uncomment the following block to enable versioning for the bucket
  # versioning {
  #   enabled = true
  # }
}

resource "aws_subnet" "tiki" {
  vpc_id     = aws_vpc.tiki.id
  cidr_block = "10.0.0.0/16"

  tags = {
    Name = "Tiki"
  }
}


resource "aws_vpc" "tiki" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "tiki"
  }
}


resource "aws_vpc_ipv4_cidr_block_association" "secondary_cidr" {
  vpc_id     = aws_vpc.tiki.id
  cidr_block = "172.2.0.0/16"
}

resource "aws_subnet" "in_secondary_cidr" {
  vpc_id     = aws_vpc_ipv4_cidr_block_association.secondary_cidr.vpc_id
  cidr_block = "172.2.0.0/24"
}
```

Q : Vpc peering

Main.tf

```
# Configure the AWS Provider
```

```
provider "aws" {
  region = "ap-south-1"
}

data "aws_ami" "ubuntu" {
  most_recent = true

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }

  owners = ["099720109477"] # Canonical
}


resource "aws_vpc" "main1" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "vpc-1"
  }
}
resource "aws_vpc" "main2" {
  cidr_block       = "11.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "vpc-2"
  }
}

#security group
resource "aws_security_group" "example_sg" {
  name_prefix = "example_sg"
  description = "Example Security Group"

  # Replace with your desired VPC ID
  vpc_id = aws_vpc.main1.id

  # Allow inbound TCP traffic on port 22 (SSH) and 80 (HTTP)
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Allow inbound ICMP (ping) traffic
  ingress {
    from_port   = -1
    to_port     = -1
    protocol    = "icmp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Allow outbound traffic to all destinations (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" "example_sg1" {
  name_prefix = "example_sg"
  description = "Example Security Group"

  # Replace with your desired VPC ID
  vpc_id = aws_vpc.main2.id

  # Allow inbound TCP traffic on port 22 (SSH) and 80 (HTTP)
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Allow inbound ICMP (ping) traffic
  ingress {
    from_port   = -1
    to_port     = -1
    protocol    = "icmp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Allow outbound traffic to all destinations (0.0.0.0/0)
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
#subnet

resource "aws_subnet" "sub-1" {
  vpc_id     = aws_vpc.main1.id
  cidr_block = "10.0.1.0/24"
  availability_zone = "ap-south-1a"

  tags = {
    Name = "subnet-1"
  }
}
resource "aws_subnet" "sub-2" {
  vpc_id     = aws_vpc.main2.id
  cidr_block = "11.0.1.0/24"
  availability_zone = "ap-south-1b"

  tags = {
    Name = "subnet-2"
  }
}

#IGW
resource "aws_internet_gateway" "gw1" {
  vpc_id = aws_vpc.main1.id

  tags = {
    Name = "igw-1"
  }
}
resource "aws_internet_gateway" "gw2" {
  vpc_id = aws_vpc.main2.id

  tags = {
    Name = "igw-2"
  }
}

#route table
resource "aws_route_table" "rt1" {
  vpc_id = aws_vpc.main1.id
 route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.gw1.id
  }


  tags = {
    Name = "routetable-1"
  }
}
resource "aws_route_table" "rt2" {
  vpc_id = aws_vpc.main2.id
 route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.gw2.id
  }


  tags = {
    Name = "routetable-2"
  }
}

#route table association

resource "aws_route_table_association" "a1" {
  subnet_id      = aws_subnet.sub-1.id
  route_table_id = aws_route_table.rt1.id
}
resource "aws_route_table_association" "a2" {
  subnet_id      = aws_subnet.sub-2.id
  route_table_id = aws_route_table.rt2.id
}
#keypair creation
resource "aws_key_pair" "keypair_peer" {
  key_name   = "deployer-key"
  public_key = "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQD3F6tyPEFEzV0LX3X8BsXdMsQz1x2cEikKD
EY0aIj41qgxMCP/iteneqXSIFZBp5vizPvaoIR3Um9xK7PGoW8giupGn+EPuxIA4cDM4vzO
qOkiMPhz5XK0whEjkVzTo4+S0puvDZuwIsdiW9mxhJc7tgBNL0cYIWSYVkz4G/fsINfRPW
5mYAM49f4fhtxPb5ok4Q2Lg9dPKVHO/Bgeu5woMc7RY0p1ej6D4CKFE6lymSDJpW0YH
X/wqE9+cfEauh7xZcG0q9t2ta6F6fmX0agvpFyZo8aFbXeUBr7osSCJNgvavWbM/06niWrO
vYX2xwWdhXmXSrbX8ZbabVohBK41 email@example.com"
}


#ec2 instance creationnnnnn

resource "aws_instance" "example-1" {
  ami           = data.aws_ami.ubuntu.id

  instance_type = "t2.micro"              # Replace with your desired instance type

          vpc_security_group_ids = [aws_security_group.example_sg.id]
          subnet_id = aws_subnet.sub-1.id
          associate_public_ip_address = true
          key_name      = aws_key_pair.keypair_peer.key_name
  tags = {
    Name = "Instance-1"
  }
}
```

```
resource "aws_instance" "example-2" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t2.micro"

 vpc_security_group_ids = [aws_security_group.example_sg1.id]
  subnet_id = aws_subnet.sub-2.id
 associate_public_ip_address = true
 key_name      = aws_key_pair.keypair_peer.key_name
  tags = {
    Name = "Instance-2"
  }
}

#vpc peering connection
resource "aws_vpc_peering_connection" "peering" {
  peer_owner_id = "xxxxxxxxxxxxxxxxxx"
  peer_vpc_id   = aws_vpc.main2.id
  vpc_id        = aws_vpc.main1.id
  auto_accept = true
}

resource "aws_route" "route_to_peer_vpc" {
  route_table_id         = aws_route_table.rt1.id
  destination_cidr_block = aws_vpc.main2.cidr_block
  vpc_peering_connection_id = aws_vpc_peering_connection.peering.id
}
resource "aws_route" "route_to_peer_vpcc" {
  route_table_id         = aws_route_table.rt2.id
  destination_cidr_block = aws_vpc.main1.cidr_block
  vpc_peering_connection_id = aws_vpc_peering_connection.peering.id
}
```

Provider.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}
```

Q : To ssh into a instance

```
provider "aws" {
  region = "ap-south-1"
}

resource "aws_instance" "example_instance" {
  ami           =” "
  instance_type = "t2.micro"
  tags = {
    Name = "tiki"
  }
}

output "public_ip" {
  value = aws_instance.example_instance.public_ip
}
```

Bash script

```
#!/bin/bash
terraform init
instance_ip=$(terraform output public_ip)

ssh -i "key-pair.pem" ec2-user@$instance_ip "sudo yum update -y && sudo
amazon-linux-extras install nginx1.12 -y"

ssh -i "key-pair.pem" ec2-user@$instance_ip "curl 127.0.0.1"
```

Q : File provisioner provision the file to the EC2 instance.

Bash script

```
#!/bin/bash

sudo apt update
sudo apt upgrade -y

sudo apt install nginx -y

sudo systemctl start nginx

sudo systemctl enable nginx
```

Main.tf

```
provider "aws" {
  region = "ap-south-1"
}

resource "aws_instance" "example" {
  ami           = ""
  instance_type = "t2.micro"

  tags = {
    Name = "tiki"
  }
  provisioner "file" {
    source      = "/home/admin/scripts/day1/num2.sh"
    destination = /home/admin/Desktop/num1.sh"
  }
}
```

Q : To kill all resources after creation

Main.tf

```
provider "aws" {
  region = "ap-south-1"
}

resource "aws_vpc" "my_vpc" {
  cidr_block = "10.0.0.0/16"
}

resource "aws_subnet" "my_subnet" {
  vpc_id     = aws_vpc.my_vpc.id
  cidr_block = "10.0.1.0/24"
}

resource "aws_internet_gateway" "my_ig" {
  vpc_id = aws_vpc.my_vpc.id
}

resource "aws_route_table" "my_rt" {
  vpc_id = aws_vpc.my_vpc.id
}

resource "aws_route" "my_route" {
  route_table_id         = aws_route_table.my_rt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.my_ig.id
}

resource "aws_route_table_association" "subnet_assoc" {
  subnet_id      = aws_subnet.my_subnet.id
  route_table_id = aws_route_table.my_rt.id
}
```

Outputs.tf

```
output "vpc_id" {
  value = aws_vpc.my_vpc.id
}
```

```
output "subnet_id" {
  value = aws_subnet.my_subnet.id
}
```

Bash scripts

```
#!/bin/bash

terraform init

terraform apply -auto-approve


VPC_ID=$(terraform output vpc_id)
SUBNET_ID=$(terraform output subnet_id)

echo "VPC ID: $VPC_ID"
echo "Subnet ID: $SUBNET_ID"

terraform destroy -auto-approve

rm -rf .terraform terraform.tfstate terraform.tfstate.backup
```