#VPC-

#main.tf

Create the VPC

```
resource "aws_vpc" "Main" {              # Creating VPC here
  cidr_block       = var.main_vpc_cidr     # Defining the CIDR block use 10.0.0.0/24 for demo
  instance_tenancy = "default"
}
#Create Internet Gateway and attach it to VPC
resource "aws_internet_gateway" "IGW" {    # Creating Internet Gateway
   vpc_id =  aws_vpc.Main.id               # vpc_id will be generated after we create VPC
}
#Create a Public Subnets.
resource "aws_subnet" "publicsubnets" {    # Creating Public Subnets
  vpc_id =  aws_vpc.Main.id
  cidr_block = "${var.public_subnets}"       # CIDR block of public subnets
}
#Create a Private Subnet                   # Creating Private Subnets
resource "aws_subnet" "privatesubnets" {
  vpc_id =  aws_vpc.Main.id
  cidr_block = "${var.private_subnets}"         # CIDR block of private subnets
}
#Route table for Public Subnet's
resource "aws_route_table" "PublicRT" {    # Creating RT for Public Subnet
   vpc_id =  aws_vpc.Main.id
        route {
   cidr_block = "0.0.0.0/0"               # Traffic from Public Subnet reaches Internet via Internet Gateway
   gateway_id = aws_internet_gateway.IGW.id
    }
}
```

```
#Route table for Private Subnet's
resource "aws_route_table" "PrivateRT" {    # Creating RT for Private Subnet
  vpc_id = aws_vpc.Main.id
  route {
  cidr_block = "0.0.0.0/0"            # Traffic from Private Subnet reaches Internet via NAT Gateway
  nat_gateway_id = aws_nat_gateway.NATgw.id
  }
}
#Route table Association with Public Subnet's
resource "aws_route_table_association" "PublicRTassociation" {
   subnet_id = aws_subnet.publicsubnets.id
   route_table_id = aws_route_table.PublicRT.id
}
#Route table Association with Private Subnet's
resource "aws_route_table_association" "PrivateRTassociation" {
   subnet_id = aws_subnet.privatesubnets.id
   route_table_id = aws_route_table.PrivateRT.id
}
resource "aws_eip" "nateIP" {
  vpc   = true
}
#Creating the NAT Gateway using subnet_id and allocation_id
resource "aws_nat_gateway" "NATgw" {
  allocation_id = aws_eip.nateIP.id
  subnet_id = aws_subnet.publicsubnets.id
}
```

#variables.tf

```
variable "region" {
```

```
type =string
description="region for ec2"
default = "us-east-1"
}
variable "main_vpc_cidr" {
type =string
description="cidr block for vpc"
default = "10.0.0.0/16"
}
variable "public_subnets" {
type =string
description="cidr block for public subnet"
default = "10.0.1.0/24"

}
variable "private_subnets" {
type =string
description="cidr block for private subnet"
default = "10.0.2.0/24"

}
```

#EC2 instance in vpc

#main.tf

```
resource "aws_instance" "webserver-input" {
ami = var.ami
instance_type = var.type
```

```
tags =var.tags
subnet_id = var.subnet_id
key_name = var.keyname
}
```

#variables.tf

```
variable "keyname" {
type = string
description="pem key anem"
default="24marchAfternoon"
}
variable "subnet_id" {
type = string
description="subnet id for public instance"
default="subnet-00bae1f697fa8e291"
}
variable "ami" {
type =string
description="AMI idd for the ec2"
default = "ami-0c02fb55956c7d316"
validation {
condition = length(var.ami) > 4 && substr(var.ami, 0, 4) == "ami-"
error_message = "Please provide a valid value for variable AMI."
}
}
variable "type"{
type =string
description="Instnce type"
default="t2.micro"
}
```

```
variable "tags" {
type = object({
name = string
env = string
})
description = "Tags for the EC2 instance"
default = {
name = "public instance"
env = "Dev"
}
}
```

EFS

terraform.tfvars

```
region = "us-east-1"
main_vpc_cidr = "10.0.0.0/16"
public_subnets = "10.0.1.0/24"
private_subnets = "10.0.2.0/24"

engine              = "mysql"
engine_version      = "5.7"
instance_class      = "db.t3.micro"
name                = "mydatabase"
username            = "dbuser"
password            = "password"
parameter_group_name = "default.mysql5.7"
```

efsmain.tf

```
resource "aws_efs_file_system" "efs" {
  creation_token = "efs"
  performance_mode = "generalPurpose"
  throughput_mode = "bursting"
  encrypted = "true"
tags = {
    Name = "EFS"
  }
}
resource "aws_efs_mount_target" "efs-mt" {
  file_system_id  = aws_efs_file_system.efs.id
  subnet_id = aws_subnet.publicsubnets.id
  security_groups = [aws_security_group.efs.id]
}
```

securitymain.tf

```
resource "aws_security_group" "ec2" {
 name       = "allow_efs"
 description = "Allow efs outbound traffic"
 vpc_id      = aws_vpc.Main.id
 ingress {
   cidr_blocks = ["0.0.0.0/0"]
   from_port = 22
   to_port = 22
   protocol = "tcp"
```

```
  }
  egress {
    from_port       = 0
    to_port         = 0
    protocol        = "-1"
    cidr_blocks     = ["0.0.0.0/0"]
  }
  tags = {
    Name = "allow_efs"
  }
}

resource "aws_security_group" "efs" {
   name = "efs-sg"
   description= "Allos inbound efs traffic from ec2"
   vpc_id = aws_vpc.Main.id

   ingress {
     security_groups = [aws_security_group.ec2.id]
     from_port = 2049
     to_port = 2049
     protocol = "tcp"
   }

   egress {
     security_groups = [aws_security_group.ec2.id]
     from_port = 0
```

```
    to_port = 0
    protocol = "-1"
  }
}
```

Rdsmain.tf

```
resource "aws_db_instance" "default" {
 allocated_storage    = 10
 engine               = var.engine
 engine_version       = var.engine_version
 instance_class       = var.instance_class
 name                 = var.name
 username             = var.username
 password             = var.password
 parameter_group_name = var.parameter_group_name
 db_subnet_group_name   = aws_db_subnet_group.default.name
 vpc_security_group_ids = [ aws_security_group.ec2.id ]
 skip_final_snapshot      = true


}

resource "aws_db_subnet_group" "default" {
 name      = "main"
 subnet_ids = [aws_subnet.publicsubnets.id, aws_subnet.privatesubnets.id]
```

```
  tags = {
    Name = "My DB subnet group"
  }
}
```

elbmain.tf

```
resource "aws_elb" "classicbar" {
  name               = "classicelb"
  availability_zones = ["us-east-1a", "us-east-1b", "us-east-1c"]



listener {
    instance_port     = 8000
    instance_protocol = "http"
    lb_port           = 80
    lb_protocol       = "http"
  }



  health_check {
    healthy_threshold   = 2
    unhealthy_threshold = 2
    timeout             = 3
    target              = "HTTP:8000/"
    interval            = 30
  }



  cross_zone_load_balancing   = true
```

```
  idle_timeout                = 400
  connection_draining         = true
  connection_draining_timeout = 400


  tags = {
    Name = "classicelb"
  }
}
```

Add two roles to your ec2 instance ec2fullaccess and amazonrdsfullaccess

Run terraform cmds