PROJECT : 1

TERRAFORM

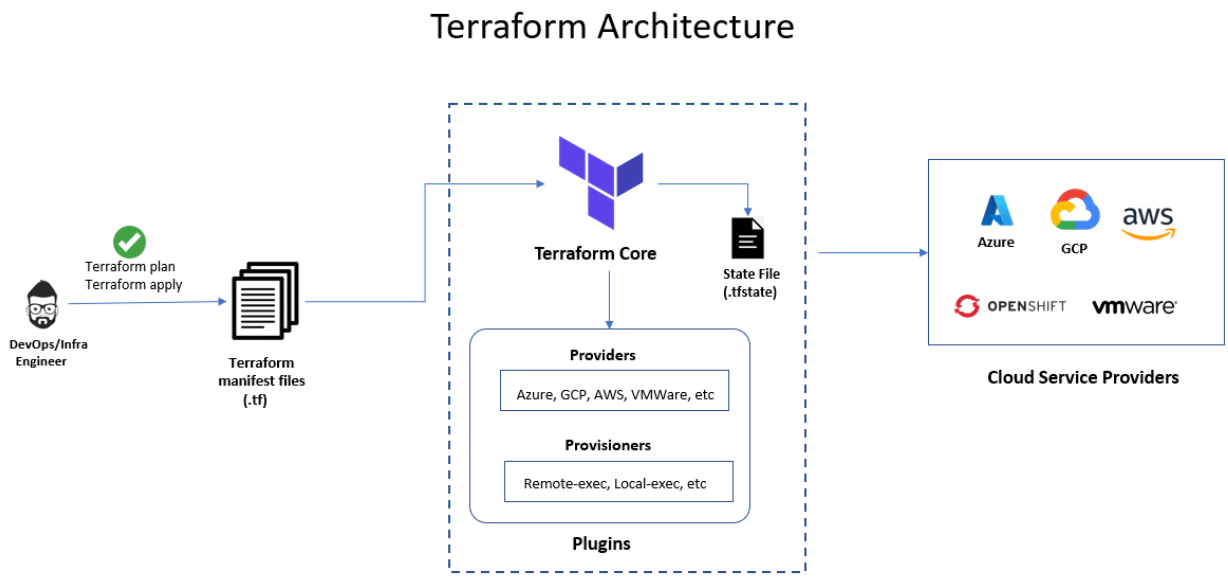
What is terraform ?

- Terraform is an Infrastructure as Code tool, meaning it allows you to manage and provision infrastructure resources (like virtual machines, networks, storage, etc.) using code rather than manual processes.
- With Terraform, you define the desired state of your infrastructure in configuration files using a declarative language called HashiCorp Configuration Language (HCL).
- Terraform supports provisioning infrastructure across various cloud providers such as AWS, Azure, Google Cloud Platform, as well as other services like Kubernetes, Docker, etc.
- Terraform automatically manages dependencies between resources
- Before making any changes to your infrastructure, Terraform generates an execution plan.
- Terraform maintains a state file that keeps track of the current state of your infrastructure.
- Terraform follows the idempotent principle, meaning running the same configuration multiple times results in the same outcome.

BASIC COMMANDS OF TERRAFORM :

- **terraform init**:
    -
    - Initializes a Terraform working directory by downloading necessary plugins and modules defined in the configuration.
- **terraform plan**:
    - Generates an execution plan showing what actions Terraform will take to change the current infrastructure to match the desired state defined in the configuration files.
- **terraform apply**:
    - Applies the changes required to reach the desired state specified in the Terraform configuration files. This command executes the execution plan generated by **`terraform plan`**.
- **terraform destroy**:

    - Destroys all resources managed by Terraform. It reads the Terraform state to determine what resources exist and then deletes them.
- **terraform validate**:
    - Validates the configuration files in the current directory to ensure they are syntactically valid and internally consistent.
- **terraform fmt**:
    - Rewrites Terraform configuration files to a canonical format and style. This command ensures consistent formatting across the configuration files.
- **terraform get**:
    - Downloads and installs modules and plugins defined in the configuration files. This command is deprecated in newer versions of Terraform (0.13 and later) as **`terraform init`** automatically handles this functionality.
- **terraform show**:
    - Outputs the current state or a specific resource's state in a human-readable format. It provides information about the resources managed by Terraform.
- **terraform state**:
    - Provides various subcommands to view or manipulate Terraform's state files. It can be used to inspect the current state, perform operations like moving or deleting state files, etc.



PRE REQUISITES:

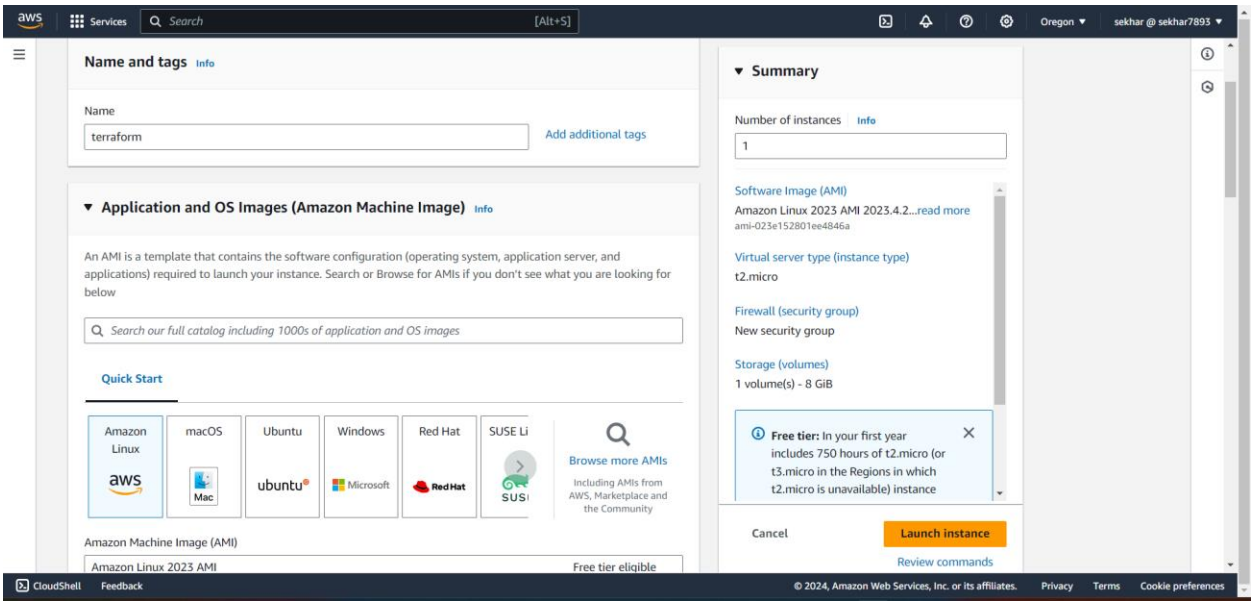
- Basic knowledge of aws & Terraform
- AWS Account
- IAM Account
- GitHub Account
- AWS Access & Secret Key

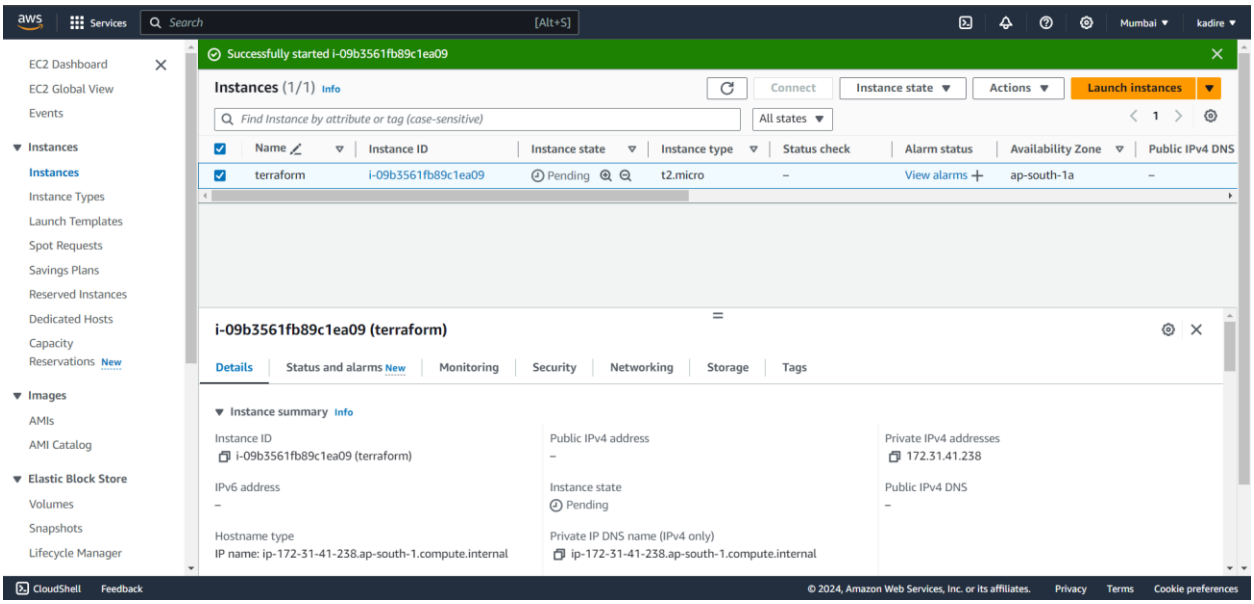
Lists of steps in the pipeline:

- STEP:1 à Create a file for the VPC
- STEP:2 à Create a file for the Subnet
- STEP:3 à Create a file for the Internet Gateway
- STEP:4 à Create a file for the Route table
- STEP:5 à Create a file for the EC2 Instance
- STEP:6 à Create a file for the Security group for the front-end tier
- STEP:7 à Create a file for the Security group for the Database tier
- STEP:8 à Create a file for the Application Load Balancer
- STEP:9 à Create a file for the RDS Instance
- STEP:10 à Create a file for the Outputs
- STEP:11 à Create a file for the Variables
- STEP:12 à Create a file for the User data
- STEP:13 à Create a file for the Resources

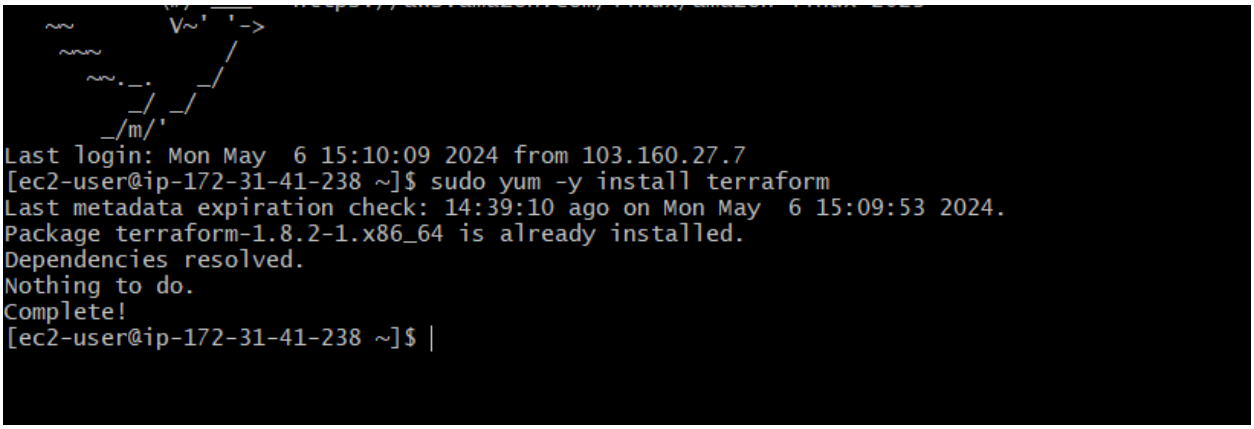
PROCEDURE:

- Open your AWS management console and Login to your AWS account.
- Click on EC2 and start launching an Instance.
- Give a name to your Instance and can add additional tags.
- Now the Instance is Launched.

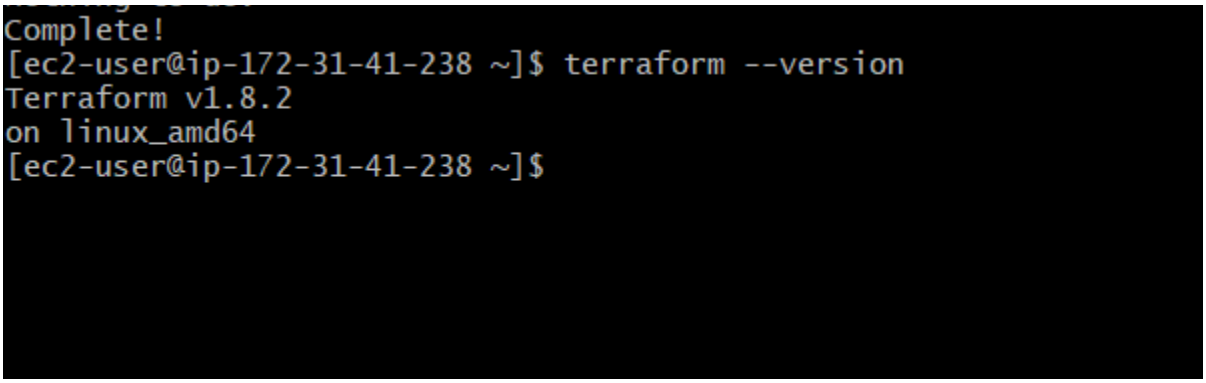




- Connect to the **Git bash Terminal**
- Go to google and search for **Install Terraform**
- **Copy** the code and **paste** it to the git bash
- Here we can see the **Terraform** was successfully **installed**



- Here we can check that terraform was installed or not using the command “terraform -version”



Creation of AWS Access key & Secert key

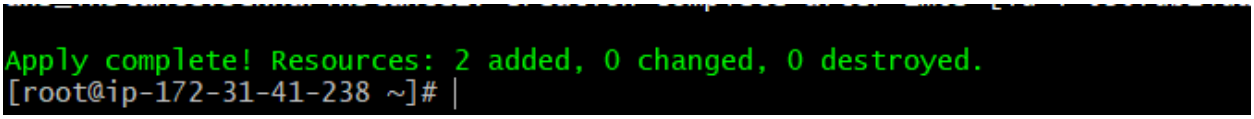
- Console the AWS account
- Go to IAM dashboard
- Go to Quick Links à Go to My security credentials
- Navigate the Create access key option and after creation access& secret key that we should save safely some place

Create Providers:

- Create a provider.tf and add given code

```
provider "aws" {
  region     = "us-east-1"
  access_key = "********** "
  secret_key = "**********"
}
```

Provider.tf file is apply is completed as shown given below.



STEP:1 → Create a file for the VPC:

- Create "vpc.tf" file and add below the code

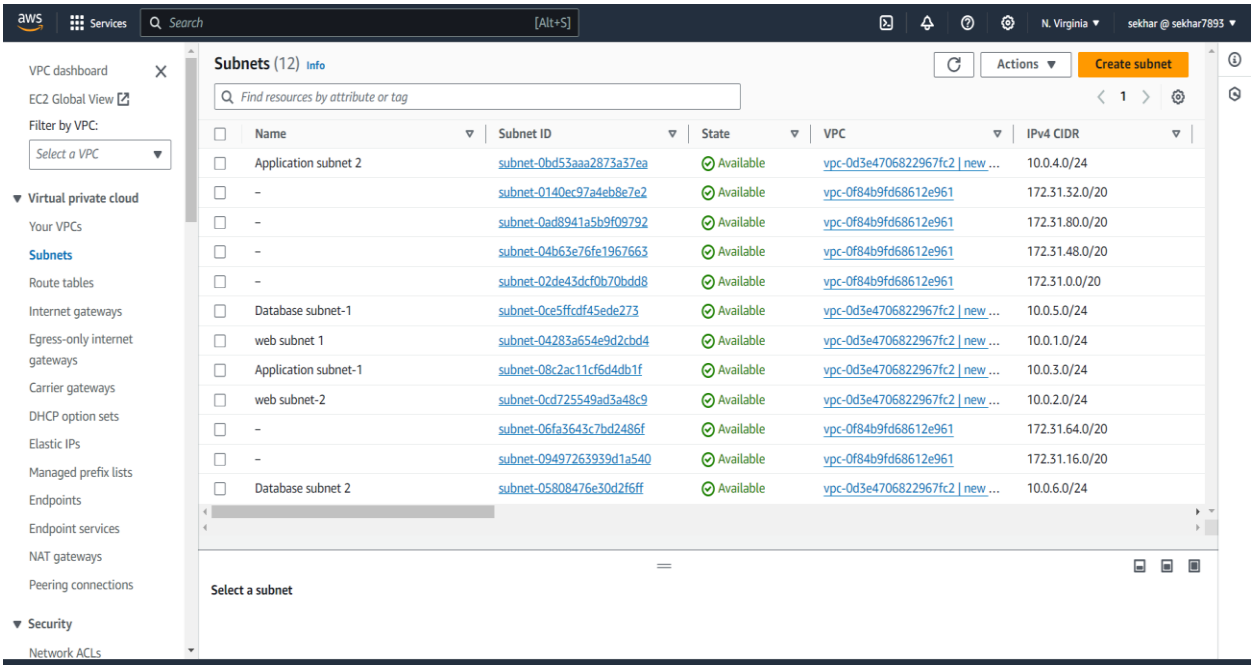
```
resource "aws_vpc" "demovpc" {
  cidr_block       = var.vpc_cidr
  instance_tenancy = "default"
  tags = {
    Name = "demo vpc"
  }
}

variable "vpc_cidr" {
  default = "10.0.0.0/16"

}
```

- Terraform init à terraform fmt à terraform plan à terraform apply above commands are running after that we found the output as shown below

```
aws_security_group.demosg: Creating...
aws_security_group.demosg: Creation complete after 5s [id=sg-09b219df6da3c1d26]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-41-238 ~]#
```

STEP:2 à Create a file for the Subnet:

- Create a “subnet.tf” file add given code to it




```
resource "aws_subnet" "public_subnet-1" {
  vpc_id                  = aws_vpc.demovpc.id
```

```
  cidr_block              = var.subnet_cidr
  map_public_ip_on_launch = true
  availability_zone       = "us-east-1a"
  tags = {
    Name = "web subnet 1"
  }
}


resource "aws_subnet" "public_subnet-2" {
  vpc_id                  = aws_vpc.demovpc.id
  cidr_block              = var.subnet1_cidr
  map_public_ip_on_launch = true
  availability_zone       = "us-east-1b"
  tags = {
    Name = "web subnet 2"
  }
}

resource "aws_subnet" "application_subnet-1" {
  vpc_id                  = aws_vpc.demovpc.id
  cidr_block              = var.subnet2_cidr
  map_public_ip_on_launch = false
  availability_zone       = "us-east-1a"
  tags = {


   Name = "appilcation subnet 1"
  }
}

resource "aws_subnet" "application_subnet-2" {
  vpc_id                  = aws_vpc.demovpc.id
  cidr_block              = var.subnet3_cidr
  map_public_ip_on_launch = false
  availability_zone       = "us-east-1b"
```

```
  tags = {
    Name = "appilcation subnet 2"
  }
}

resource "aws_subnet" "database_subnet-1" {
  vpc_id            = aws_vpc.demovpc.id
  cidr_block        = var.subnet4_cidr
  availability_zone = "us-east-1a"
  tags = {
    Name = "database subnet 1"
  }
}

resource "aws_subnet" "database_subnet-2" {
  vpc_id            = aws_vpc.demovpc.id
  cidr_block        = var.subnet5_cidr
  availability_zone = "us-east-1b"
  tags = {
    Name = "database subnet 2"
  }
}



variable "subnet_cidr" {
  default = "10.0.1.0/24"
}

variable "subnet1_cidr" {
  default = "10.0.2.0/24"
}

variable "subnet2_cidr" {
  default = "10.0.3.0/24"
}
```

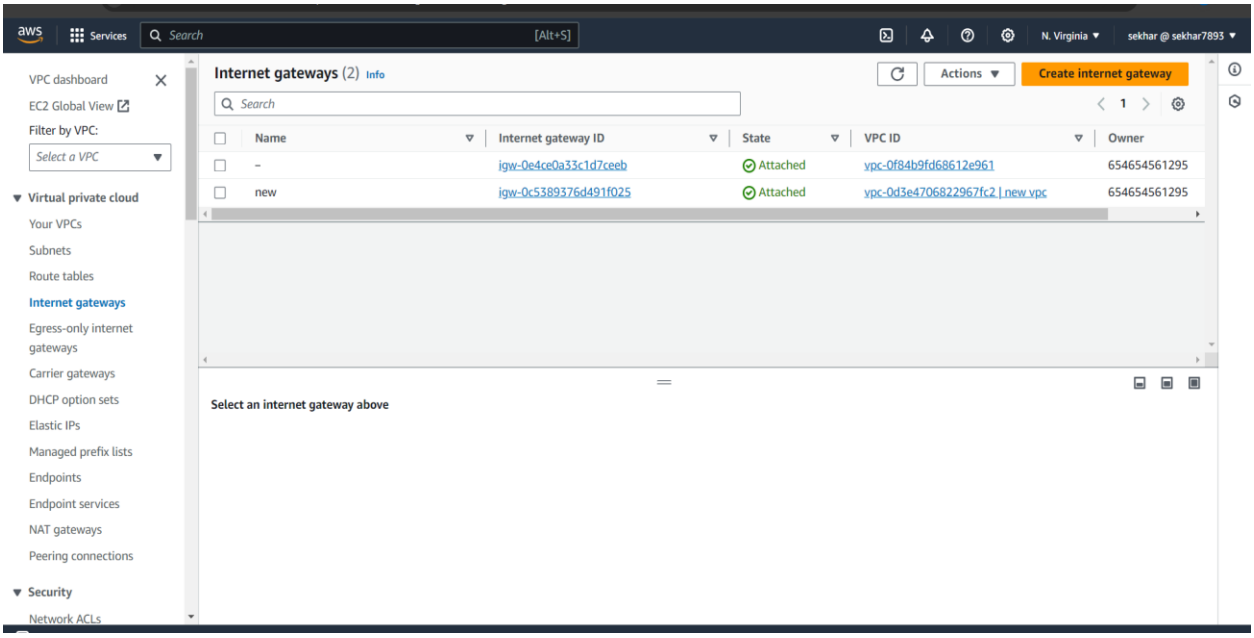
```
variable "subnet3_cidr" {


 default = "10.0.4.0/24"

}

variable "subnet4_cidr" {
  default = "10.0.5.0/24"
}

variable "subnet5_cidr" {
  default = "10.0.6.0/24"
}
```

- Here we were created total 6 subnets for the front-end tier and back-end tier with a mixture of public and private subnets
- Terraform init à terraform fmt à terraform validate à terraform apply –auto-approve (run the commands one by one respectively)

```
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
[root@ip-172-31-41-238 ~]# terraform apply
aws_vpc.newvpc: Refreshing state... [id=vpc-0d3e4706822967fc2]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-05808476e30d2f6ff]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-0cd725549ad3a48c9]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0ce5ffcdf45ede273]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-08c2ac11cf6d4db1f]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-04283a654e9d2cbd4]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-0bd53aaa2873a37ea]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_internet_gateway.gw will be created
  + resource "aws_internet_gateway" "gw" {
      + arn      = (known after apply)
      + id       = (known after apply)
      + owner_id = (known after apply)
      + tags     = {
          + "Name" = "new"
        }
      + tags_all = {
          + "Name" = "new"
        }
      + vpc_id   = "vpc-0d3e4706822967fc2"
    }

Plan: 1 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_internet_gateway.gw: Creating...
aws_internet_gateway.gw: Creation complete after 2s [id=igw-0c5389376d491f025]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-41-238 ~]#
```

STEP:3 Create a file for the Internet Gateway:

- Create a file “igw.tf” and add the below code to it



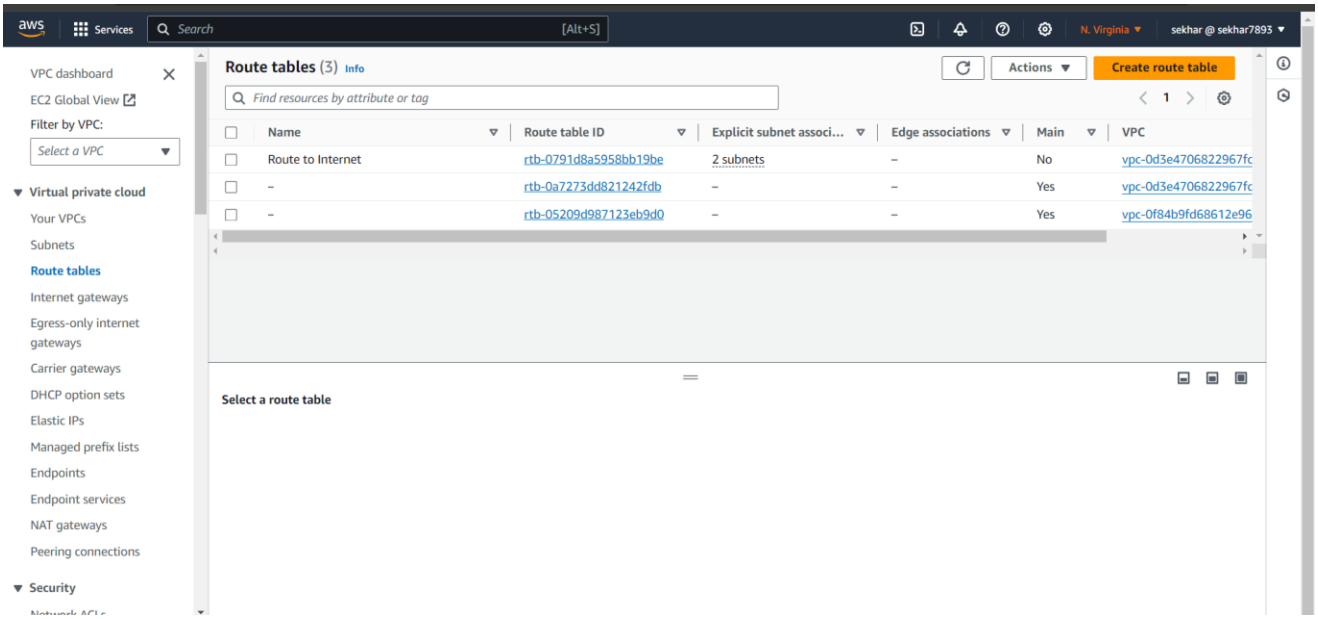
```
resource "aws_internet_gateway" "demogateway" {
  vpc_id = aws_vpc.demovpc.id
}
```

- Terraform fmt à terraform validate à terraform apply –auto-approve (run the commands one by one respectively)

```
aws_internet_gateway.gw: Creating...
aws_internet_gateway.gw: Creation complete after 2s [id=igw-0c5389376d491f025]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-41-238 ~]#
```

STEP:4 à Create a file for the Route table:

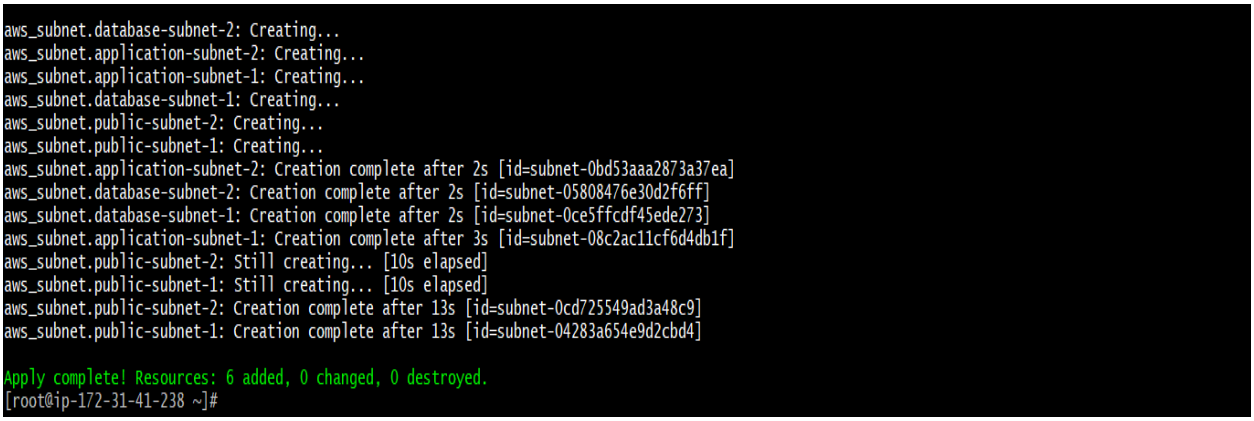
- Create a file “route_table_public.tf” and add the below code to it




```
resource "aws_route_table" "route" {
```

```
  vpc_id = aws_vpc.demovpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.demogateway.id
  }
  tags = {
    Name = "Route to internet"
  }
}
resource "aws_route_table_association" "rt1" {
  subnet_id      = aws_subnet.public_subnet-1.id
  route_table_id = aws_route_table.route.id
}

resource "aws_route_table_association" "rt2" {
  subnet_id      = aws_subnet.public_subnet-2.id
  route_table_id = aws_route_table.route.id
}
```

Terraform fmt à terraform validate à terraform apply -auto-approve (run the commands one by one respectively)



STEP:5 à Create a file for the EC2 Instance:

- Create a file “ec2.tf” & “ec2-1.tf” and add below the code to it

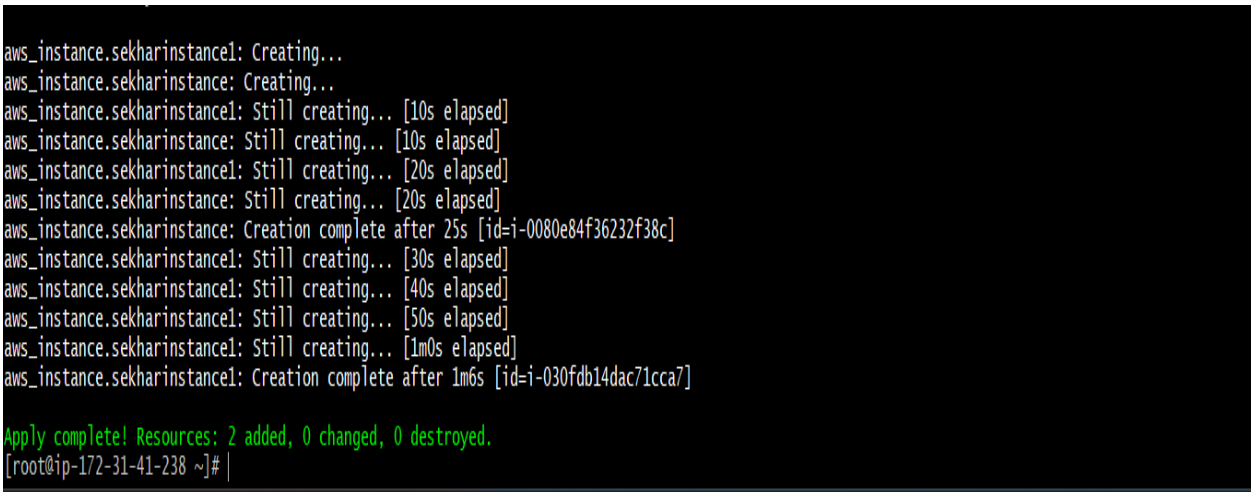
- For “ec” instance code:

```
resource "aws_instance" "demoinstance" {
  ami                    = "ami-0a1179631ec8933d7"
  instance_type          = "t2.micro"
  count                  = 1
  key_name               = "pp"
  vpc_security_group_ids = [aws_security_group.demosg.id]
  subnet_id              = aws_subnet.public_subnet-1.id
  associate_public_ip_address = "true"
  user_data              = file("data.sh")
  tags = {
    Name = "ec"

  }

}
```

- For “ec2” instance code

```
resource "aws_instance" "demoinstance1" {
  ami                    = "ami-0a1179631ec8933d7"
  instance_type          = "t2.micro"
  count                  = 1
  key_name               = "pp"
  vpc_security_group_ids = [aws_security_group.demosg.id]
  subnet_id              = aws_subnet.public_subnet-2.id
  associate_public_ip_address = "true"
  user_data              = file("dataa.sh")
  tags = {
    Name = "ec2"
  }
}
```

Terraform fmt à terraform validate à terraform apply -auto-approve (run the commands one by one respectively)



STEP:6

A Create a file for Security group for the front-end tier:

- Create a file " web_sg.tf " and add below the code to it

```
resource "aws_security_group" "demosg" {
  vpc_id = aws_vpc.demovpc.id
  #inbound rules
  #http access from any where
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
```

```
  #http access from any where
  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  #ssh access from any where
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  #outbound rules
  #internet access to anywhere
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "websg"
  }
}
```

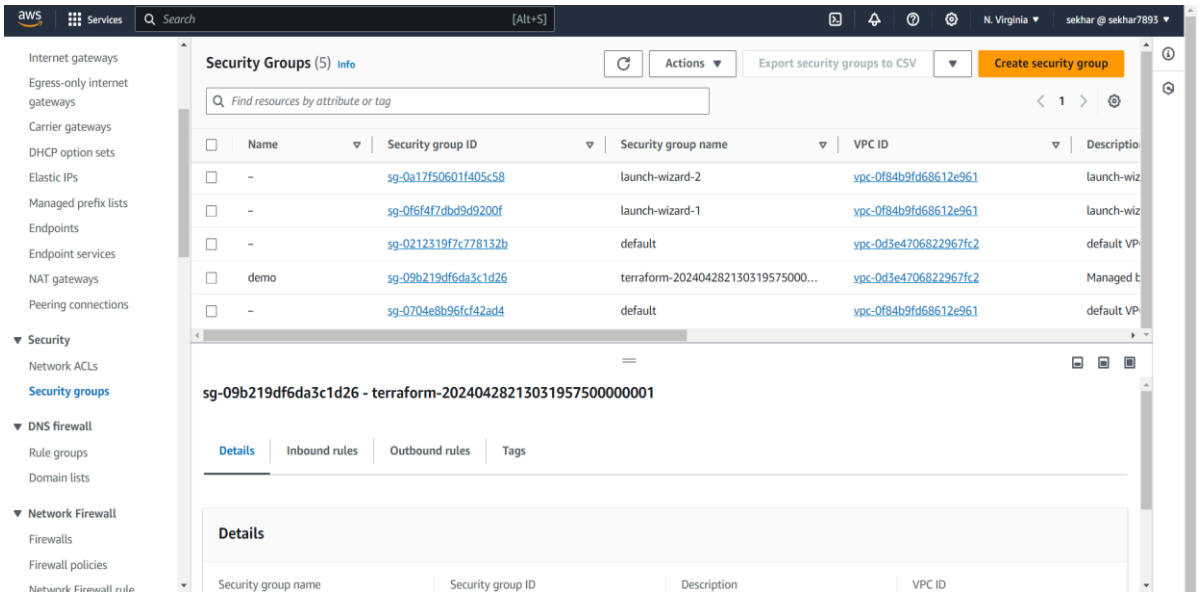
- Terraform fmt à terraform validate à terraform apply -auto-approve (run the commands one by one respectively)

Here we opened 80,443,22 ports for the inbound connection and we are opened all the ports for outbound connection

```
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_security_group.demosg: Creating...
aws_security_group.demosg: Creation complete after 5s [id=sg-09b219df6da3c1d26]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-41-238 ~]#
```

STEP:7 à Create a file for Security group for the Database tier:

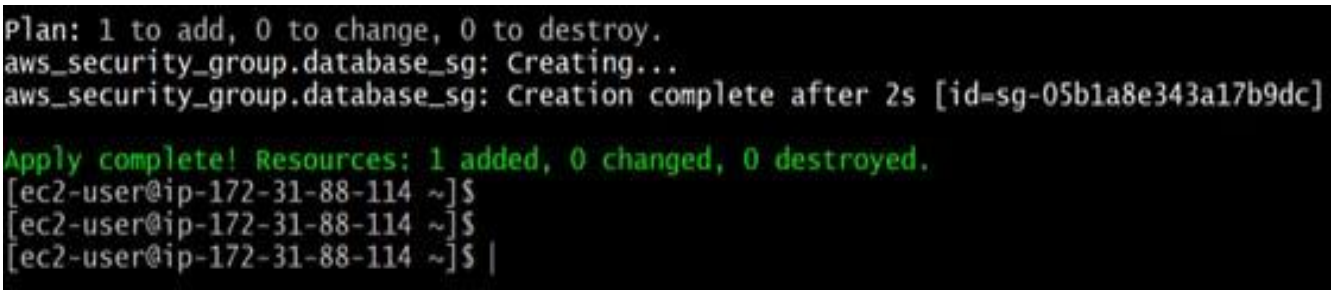
- Create a file “ database_sg.tf ” and add below the code to it




```
resource "aws_security_group" "database_sg" {
  name        = "Database SG" # Use lowercase "name" instead of "Name"
```

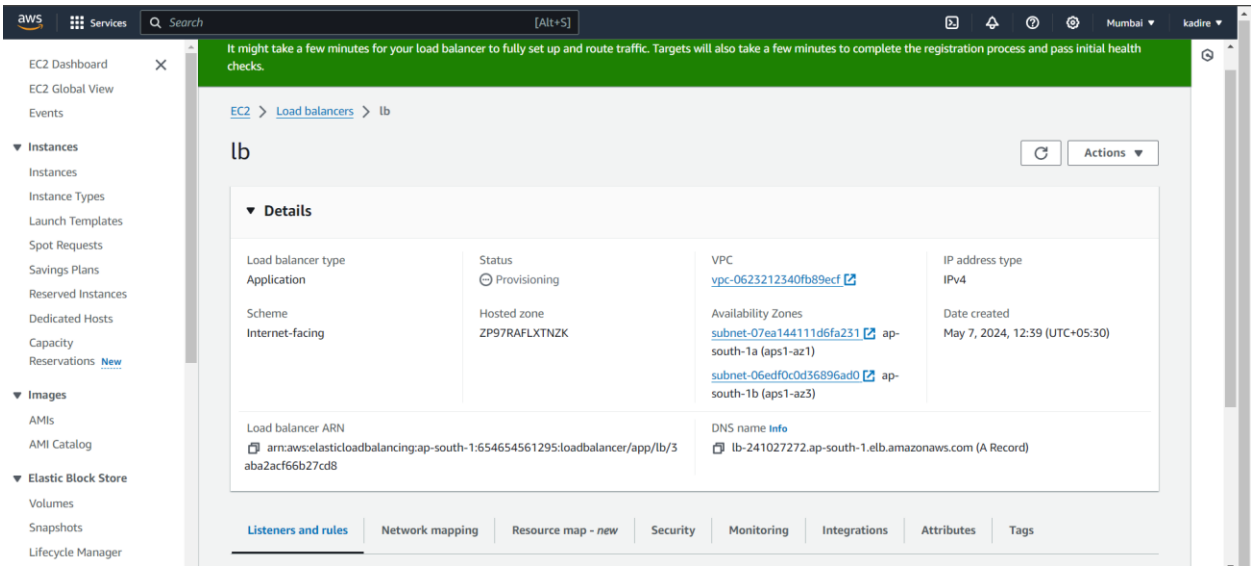
```
  description = "Allow inbound traffic from application layer"
  vpc_id      = aws_vpc.demovpc.id

  ingress {
    description     = "Allow traffic from application layer"
    from_port       = 3306
    to_port         = 3306
    protocol        = "tcp"
    security_groups = [aws_security_group.demosg.id]
  }

  egress {
    from_port   = 32768
    to_port     = 65535 # Corrected typo: "to_ port" to "to_port"
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "Database SG"
  }
}
```

- Terraform fmt à terraform validate à terraform apply –auto-approve (run the commands one by one respectively)
- We opened 3306 ports for the inbound connection and we are opened all the ports for the outbound connection.



STEP:8 à Create a file for Application Load Balancer:

- Create a file “ alb.tf ” and add below the code to it.




```
resource "aws_lb" "new_elb" {
  name               = "External-LB"
  internal           = false
  load_balancer_type = "application"
```

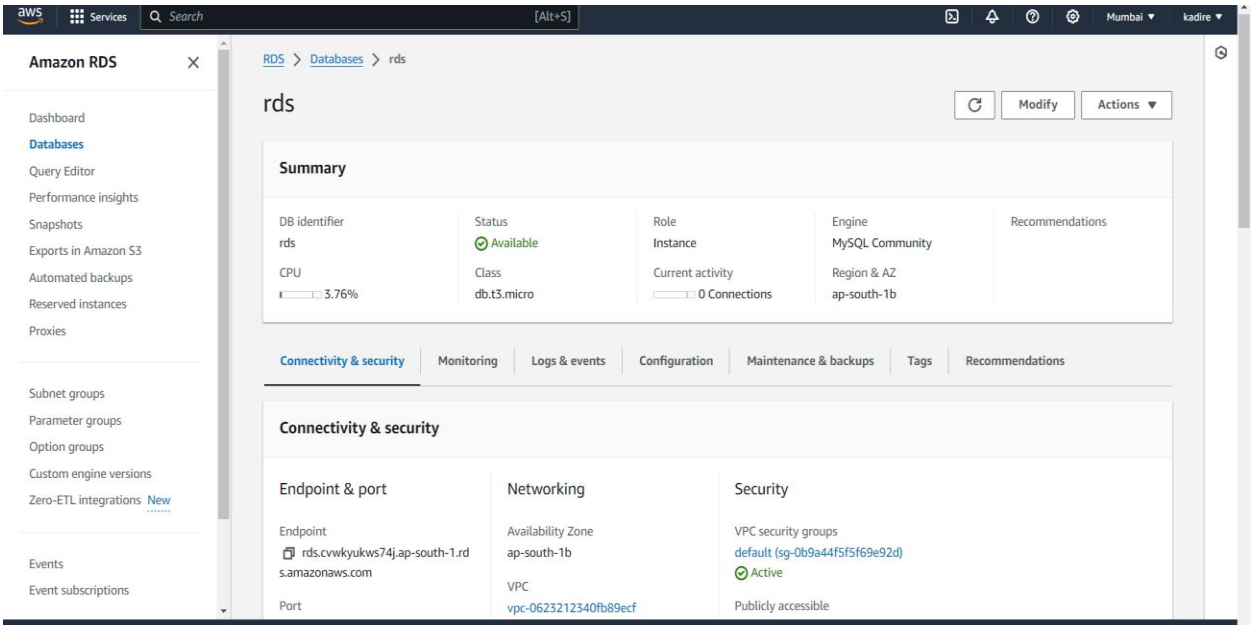
```
  security_groups    = [aws_security_group.demosg.id]
  subnets            = [aws_subnet.public_subnet-1.id,
aws_subnet.public_subnet-2.id]
}

resource "aws_lb_target_group" "target_elb" {
  name     = "ALB-TG"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.demovpc.id
}

resource "aws_lb_target_group_attachment" "attachment" {
  count            = length(aws_instance.demoinstance)
  target_group_arn = aws_lb_target_group.target_elb.arn
  target_id        = aws_instance.demoinstance[count.index].id
  port             = 80
  depends_on       = [aws_instance.demoinstance]
}

resource "aws_lb_target_group_attachment" "attachment1" {
  count            = length(aws_instance.demoinstance1)
  target_group_arn = aws_lb_target_group.target_elb.arn
  target_id        = aws_instance.demoinstance1[count.index].id
  port             = 80
  depends_on       = [aws_instance.demoinstance1]
}

resource "aws_lb_listener" "new_elb_listener" {
  load_balancer_arn = aws_lb.new_elb.arn
  port              = 80
  protocol          = "HTTP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.target_elb.arn
```

```
  }
}
```

- Terraform fmt à terraform validate à terraform apply –auto-approve (run the commands one by one respectively)
- The above load balancer is of type external
- Load balancer type is set to application
- The aws_lb_target_group_attachment resource will attach will attach our instances to the Target group
- The Load balancer will listen request on port 80

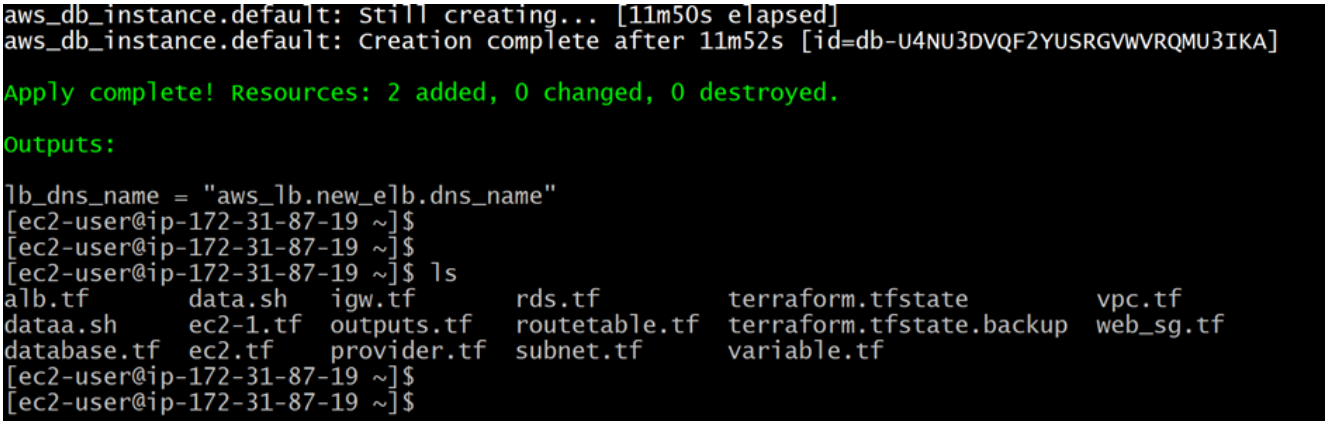
STEP:9 → Create a file for the RDS Instance:

- Create a file “rds.tf” and add below the code to it




```
resource "aws_db_subnet_group" "default" {
  name       = "new"
  subnet_ids = [aws_subnet.database_subnet-1.id,
aws_subnet.database_subnet-2.id]
  tags = {
    Name = "My DB Subnet Group"
  }
}

resource "aws_db_instance" "default" {
  allocated_storage      = 10
  db_subnet_group_name   = aws_db_subnet_group.default.id
  engine                 = "mysql"
  engine_version         = "8.0.34"
  instance_class         = "db.t3.micro"
  multi_az               = "true"
  db_name                = "db"
  username               = "sekhar"
  password               = "123456789"
  skip_final_snapshot    = "true"
  vpc_security_group_ids = [aws_security_group.database_sg.id]
}
```

- Terraform fmt à terraform validate à terraform apply –auto-approve (run the commands one by one respectively)



STEP:10 à Create a file for Outputs:

- Create a file “ outputs.tf ” and add below the code to it

```
output "lb_dns_name" {

  description ="Name of the load balancer"

  value = "aws_lb.new_elb.dns_name"

}
```

- Terraform fmt à terraform validate à terraform apply –auto-approve (run the commands one by one respectively)

From above code ,we will get the DNS of the application.

STEP:11 à Create a file for Variables:

- Create a file “variables.tf” and add below the code to it

```
variable "subnet-cidr" {
  default = "10.0.1.0/24"
}

variable "subnet1-cidr" {
  default = "10.0.2.0/24"
}

variable "subnet2-cidr" {
  default = "10.0.3.0/24"
}

variable "subnet3-cidr" {
  default = "10.0.4.0/24"
}

variable "subnet4-cidr" {
  default = "10.0.5.0/24"
}

variable "subnet5-cidr" {
  default = "10.0.6.0/24"
}
```

- Terraform fmt à terraform validate à terraform apply -auto-approve (run the commands one by one respectively)

STEP:12 à Create a file for User data:

- Create a files " data.sh" & "data.sh" and add below the code to it

- For "data.sh"

```
#!/bin/bash
yum -y install git
yum -y install httpd
sudo systemctl start httpd
sudo systemctl enable httpd
sudo git clone https://github.com/GOUSERABBANI44/Mario.git
/var/www/html/
```

- For "data1.sh"

```
#!/bin/bash
yum -y install git
yum -y install httpd
sudo systemctl start httpd
sudo systemctl enable httpd
sudo git clone https://github.com/GOUSERABBANI44/ecomm.git
/var/www/html/
```

- Terraform fmt à terraform validate à terraform apply -auto-approve (run the commands one by one respectively)

**********THE END **********