PROJECT-1

Deploy Three-Tier Architecture in AWS using Terraform

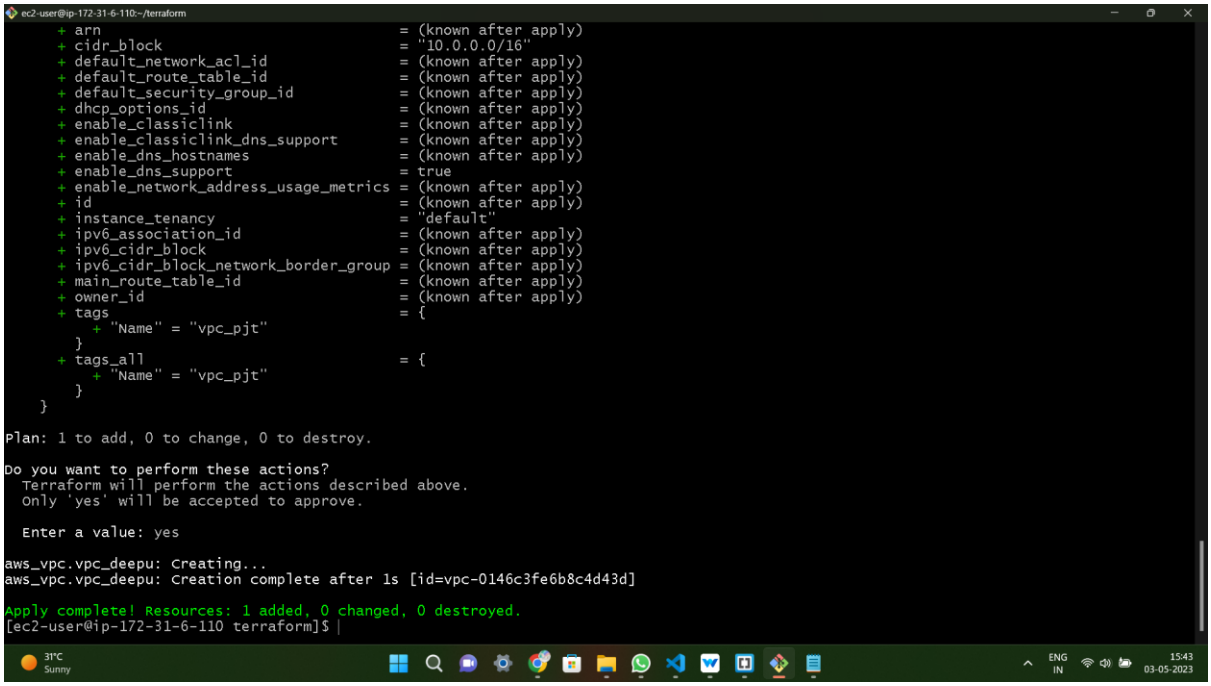
Step1: Create a file for Provider

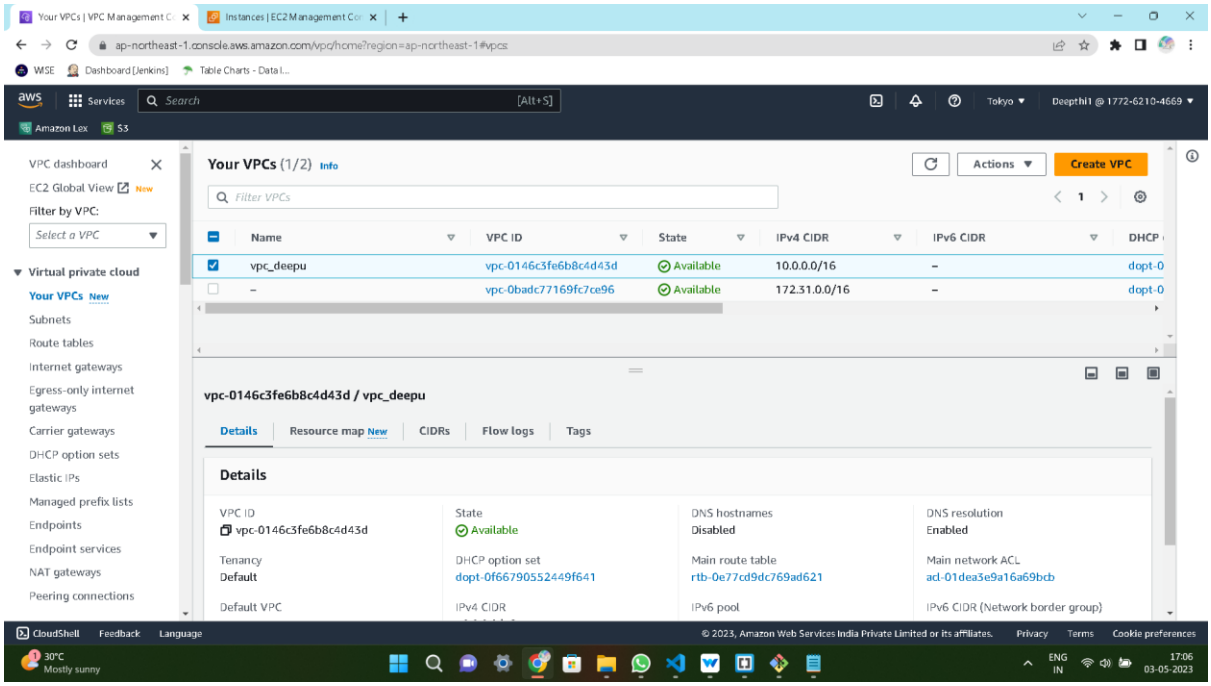
```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.65.0"

    }
  }
}

provider "aws" {
  profile    = "default"
  region     = "ap-south-1"
  access_key = "AKIAVMQJROITQWSD727D"
  secret_key = "B02gg+udlnUwwm8isWlz+btMViPmP/A/RprFSpf/"

}
```

```
ec2-user@ip-172-31-92-53:~/
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[ec2-user@ip-172-31-92-53 teraform]$ ls
provider.tf
[ec2-user@ip-172-31-92-53 teraform]$ vi provider.tf
[ec2-user@ip-172-31-92-53 teraform]$ terraform fmt
[ec2-user@ip-172-31-92-53 teraform]$ terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-31-92-53 teraform]$ terraform plan

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
[ec2-user@ip-172-31-92-53 teraform]$ terraform apply

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
```

Step2: Create a file for the vpc

```
# creating VPC
resource "aws_vpc" "vpc_deepu" {
  cidr_block       = "${var.vpc_cidr}"
  instance_tenancy = "default"

  tags = {
    Name = "vpc_deepu"
  }


}
```



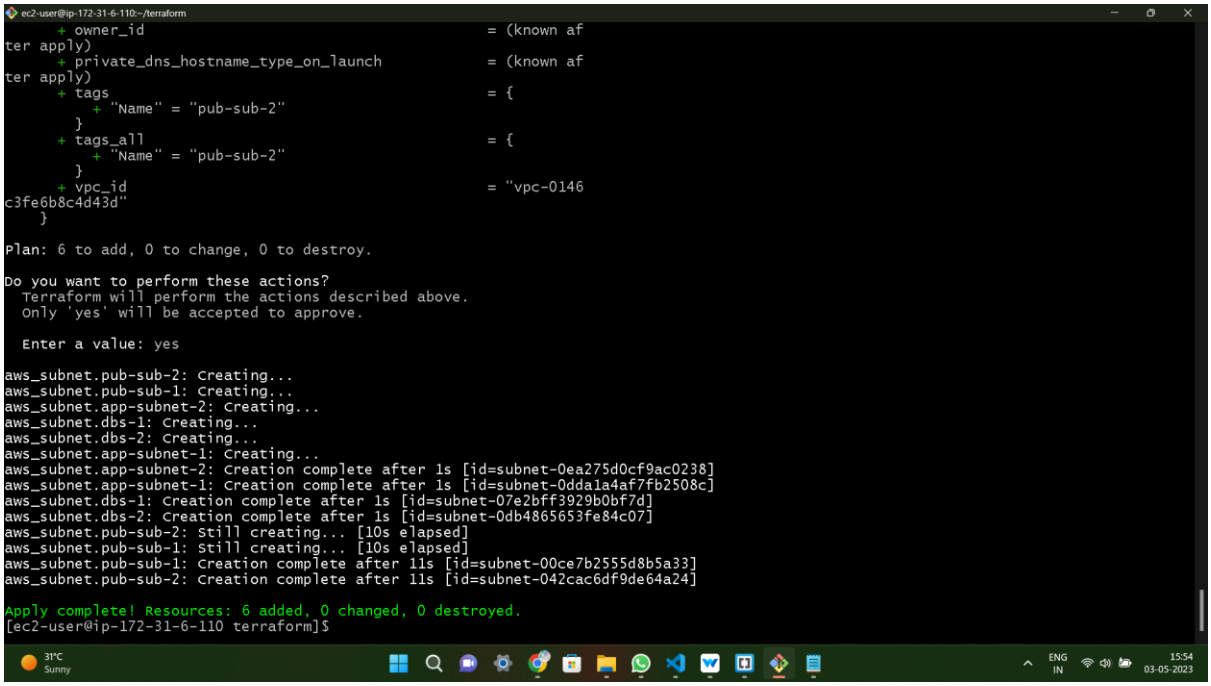


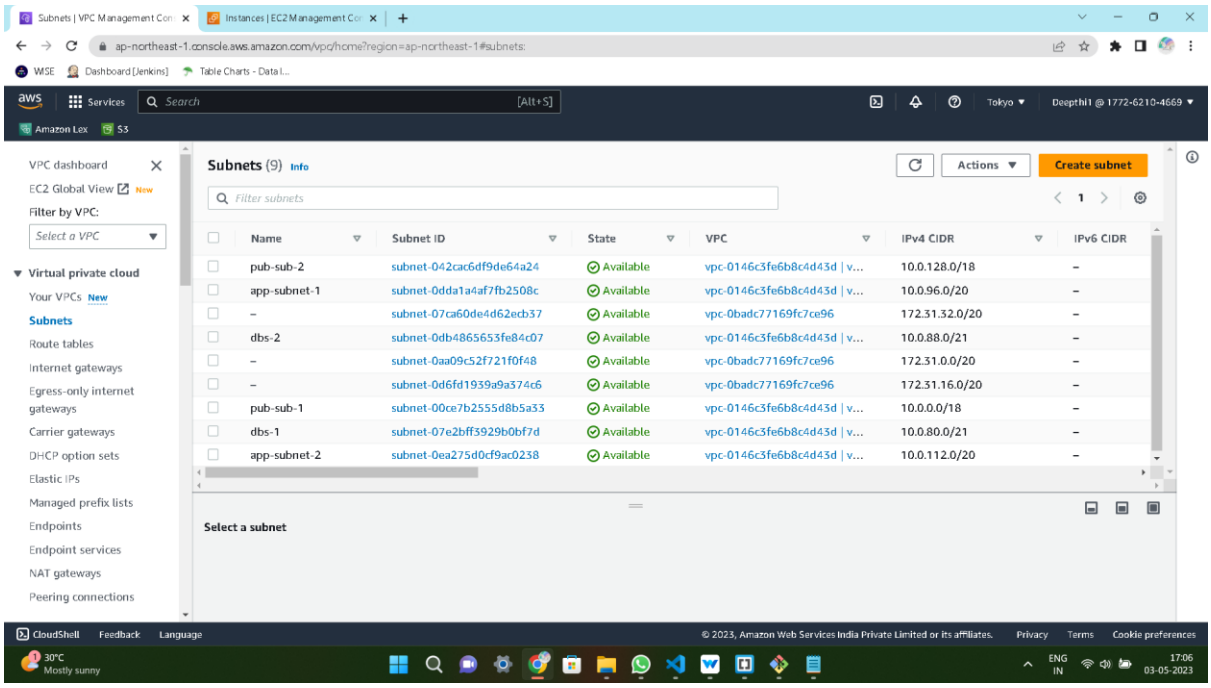
Step3: Create a file for the Subnet

```
#creating 1st web subnet
resource "aws_subnet" "pub-sub-1" {
  vpc_id                  = aws_vpc.vpc_deepu.id
  cidr_block              = "${var.websub_1}"
  availability_zone       = "ap-northeast-1a"
  map_public_ip_on_launch = true

  tags = {
    Name = "pub-sub-1"
  }

}

#creating 2nd web subnet
resource "aws_subnet" "pub-sub-2" {
  vpc_id                  = aws_vpc.vpc_deepu.id
  cidr_block              = "${var.websub_2}"
  availability_zone       = "ap-northeast-1c"
  map_public_ip_on_launch = true

  tags = {
    Name = "pub-sub-2"
  }

}

```

```
#creating 1st application subnet
resource "aws_subnet" "app-subnet-1" {
  vpc_id                  = aws_vpc.vpc_deepu.id
  cidr_block              = "${var.appsub_1}"
  availability_zone       = "ap-northeast-1a"
  map_public_ip_on_launch = false

  tags = {
    Name = "app-subnet-1"
  }

}

#creating 2nd application subnet
resource "aws_subnet" "app-subnet-2" {
  vpc_id                  = aws_vpc.vpc_deepu.id
  cidr_block              = "${var.appsub_2}"
  availability_zone       = "ap-northeast-1c"
  map_public_ip_on_launch = false

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

}

```

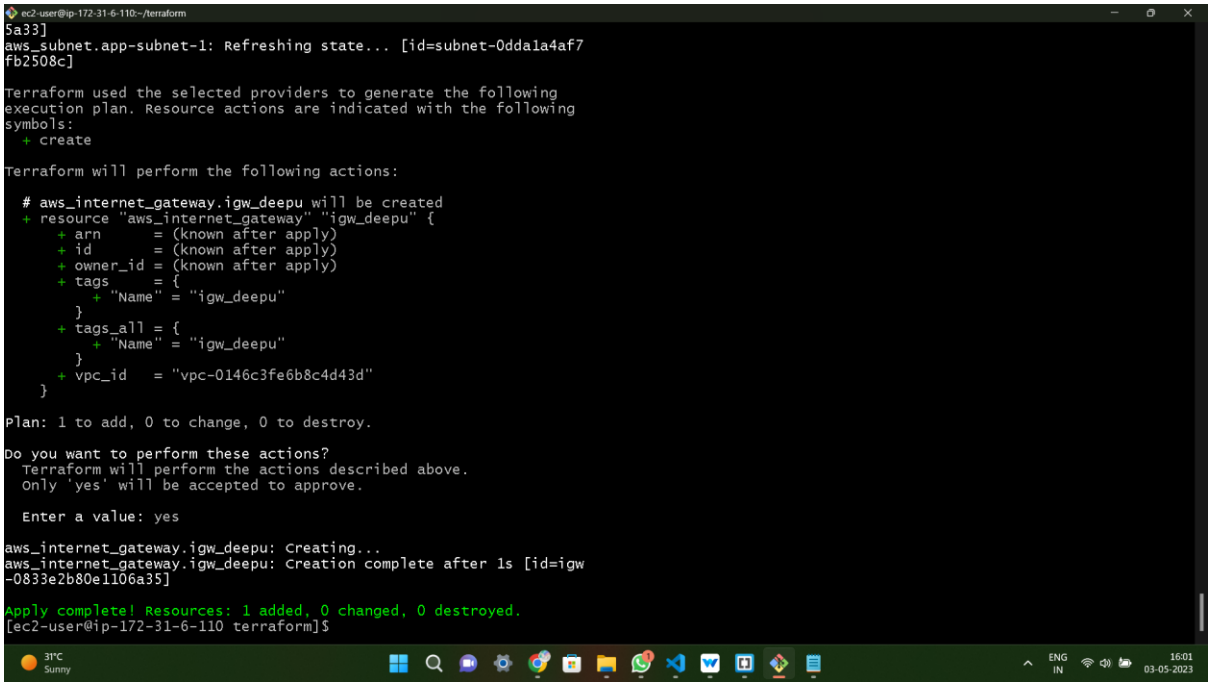
```
#creating 1st data base subnet
resource "aws_subnet" "dbs-1" {
  vpc_id                  = aws_vpc.vpc_deepu.id
  cidr_block              = "${var.dbsub_1}"
  availability_zone       = "ap-northeast-1a"
  map_public_ip_on_launch = false

  tags = {
    Name = "dbs-1"
  }

}

#creating 2nd  data base subnet
resource "aws_subnet" "dbs-2" {
  vpc_id                  = aws_vpc.vpc_deepu.id
  cidr_block              = "${var.dbsub_2}"
  availability_zone       = "ap-northeast-1c"
  map_public_ip_on_launch = false

  tags = {
    Name = "dbs-2"
  }

}

```

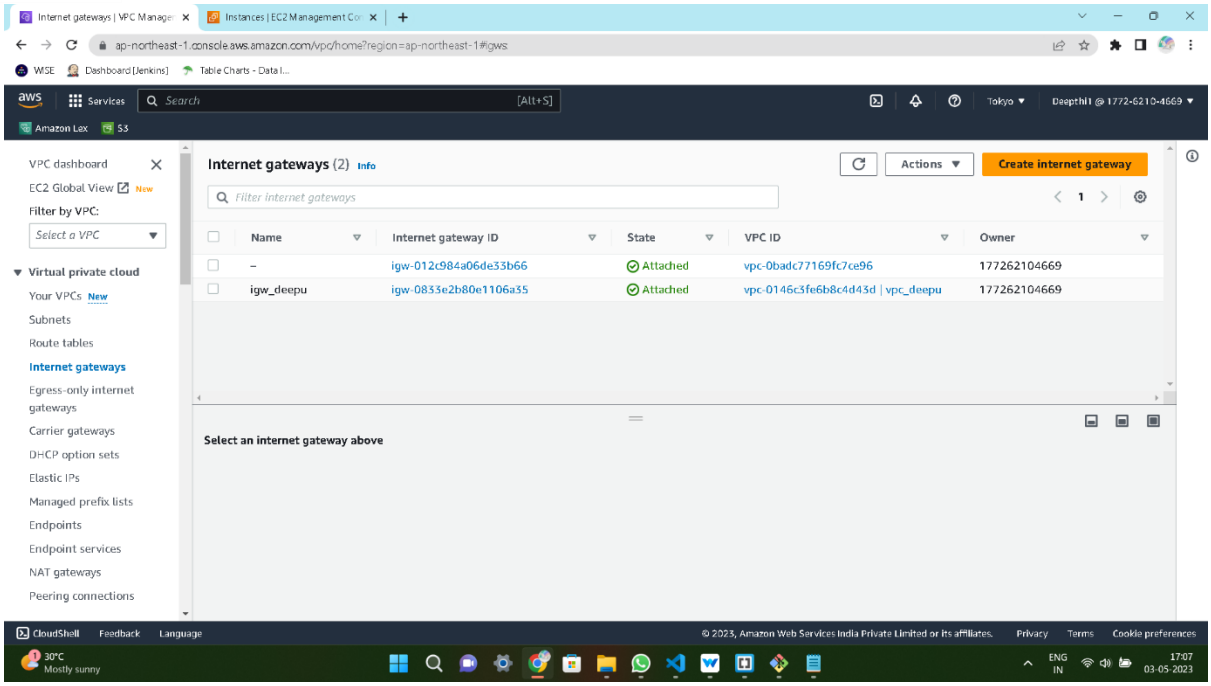




Step4: Create a file for the Internet Gateway

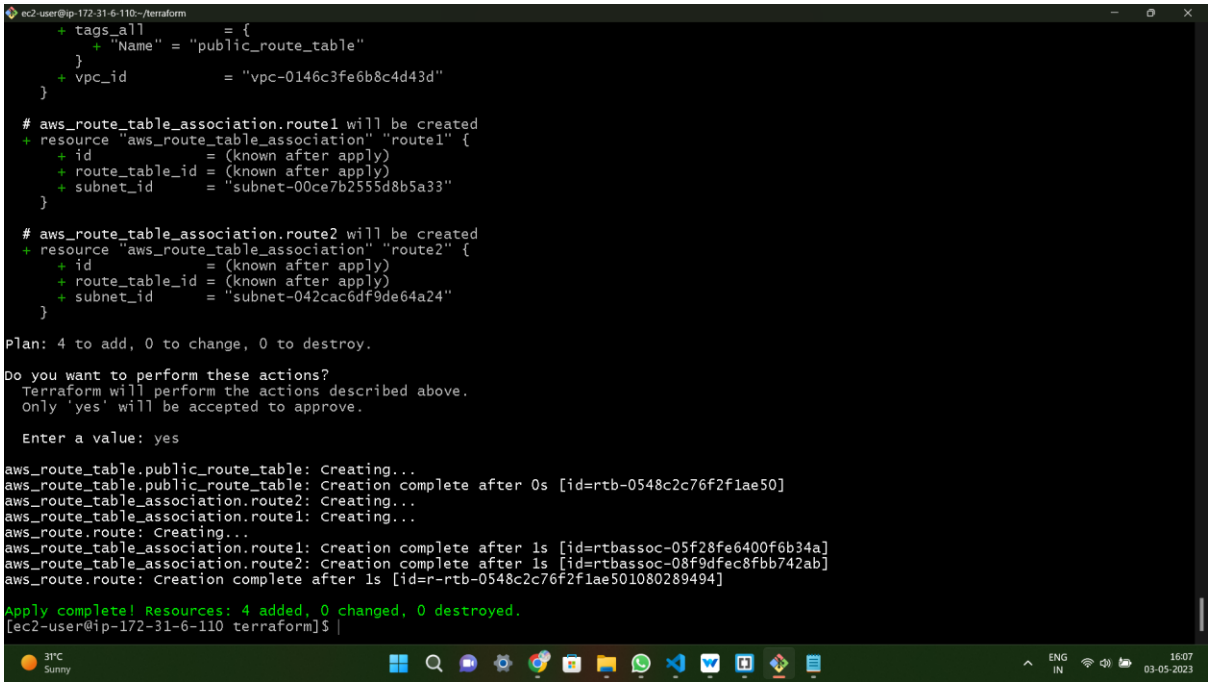
```terraform
# creating internet gateway
resource "aws_internet_gateway" "igw_deepu" {
  vpc_id = aws_vpc.vpc_deepu.id

  tags = {
    Name = "igw_deepu"
  }

}
```

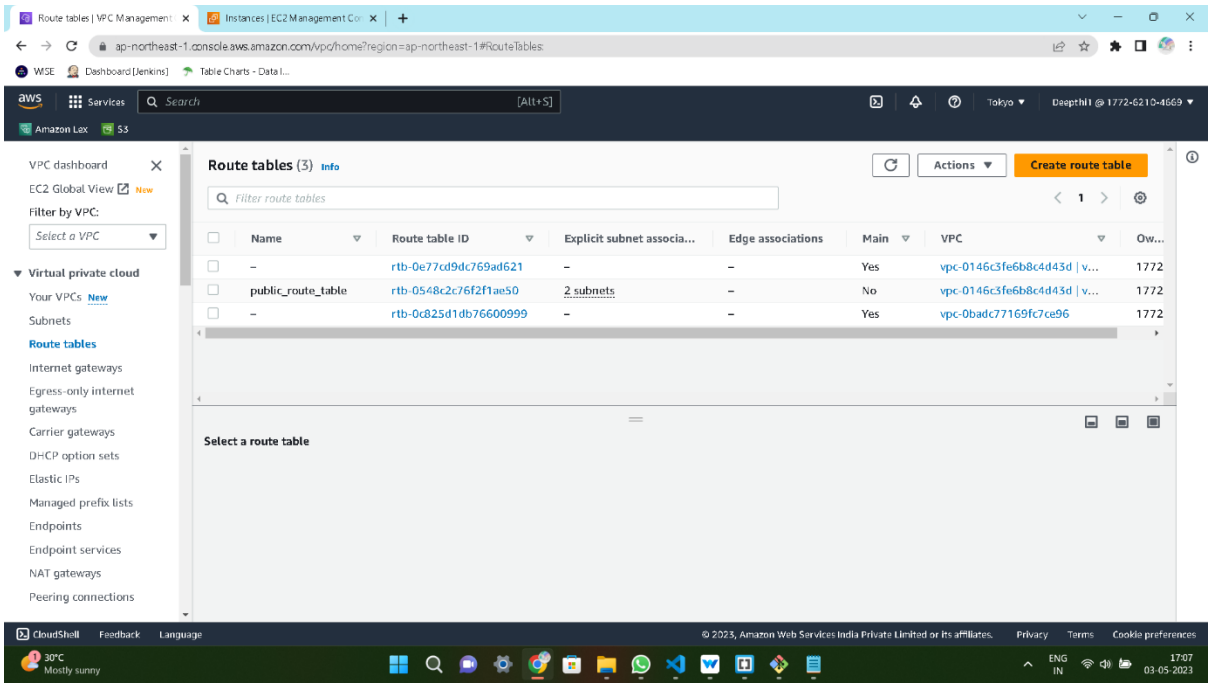




Step5: Create a file for the Route Table

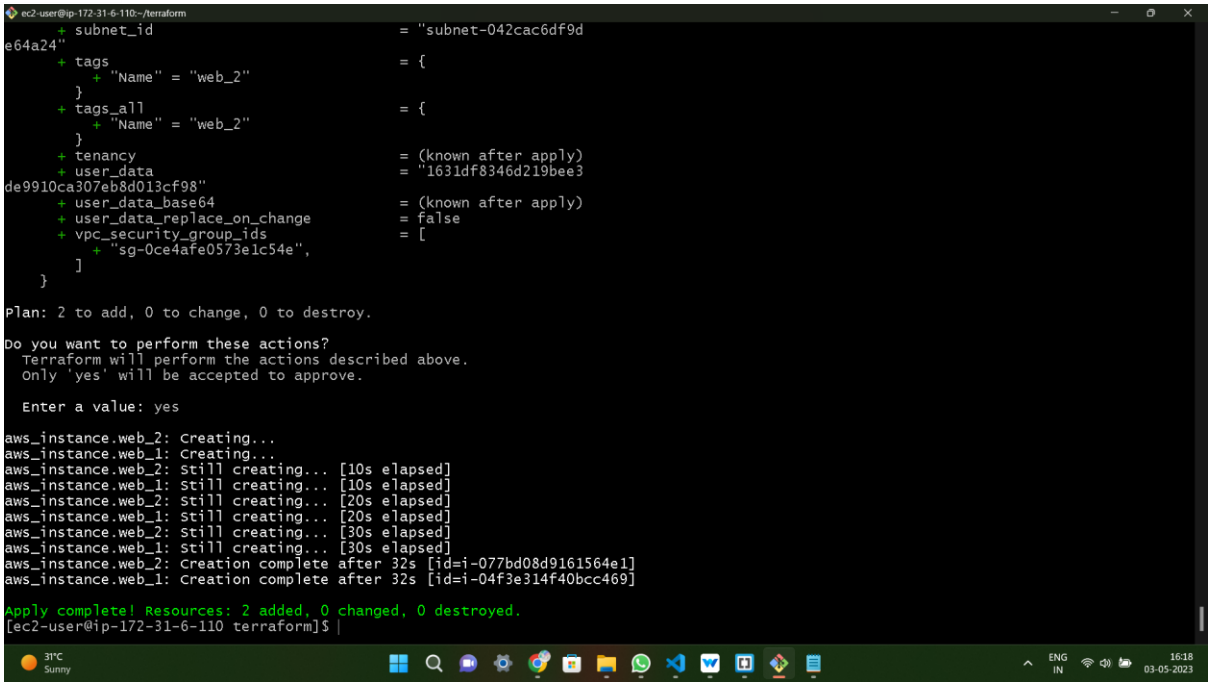
```
#creating route table
resource "aws_route_table" "public_route_table" {
    vpc_id = aws_vpc.vpc_deepu.id

    tags = {
        Name = "public_route_table"
    }


}

resource "aws_route" "route" {
    route_table_id = aws_route_table.public_route_table.id
    destination_cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw_deepu.id


}

#subnet associations

resource "aws_route_table_association" "route1" {
    subnet_id = aws_subnet.pub-sub-1.id
    route_table_id = aws_route_table.public_route_table.id

}

resource "aws_route_table_association" "route2" {
    subnet_id = aws_subnet.pub-sub-2.id
    route_table_id = aws_route_table.public_route_table.id

}
```

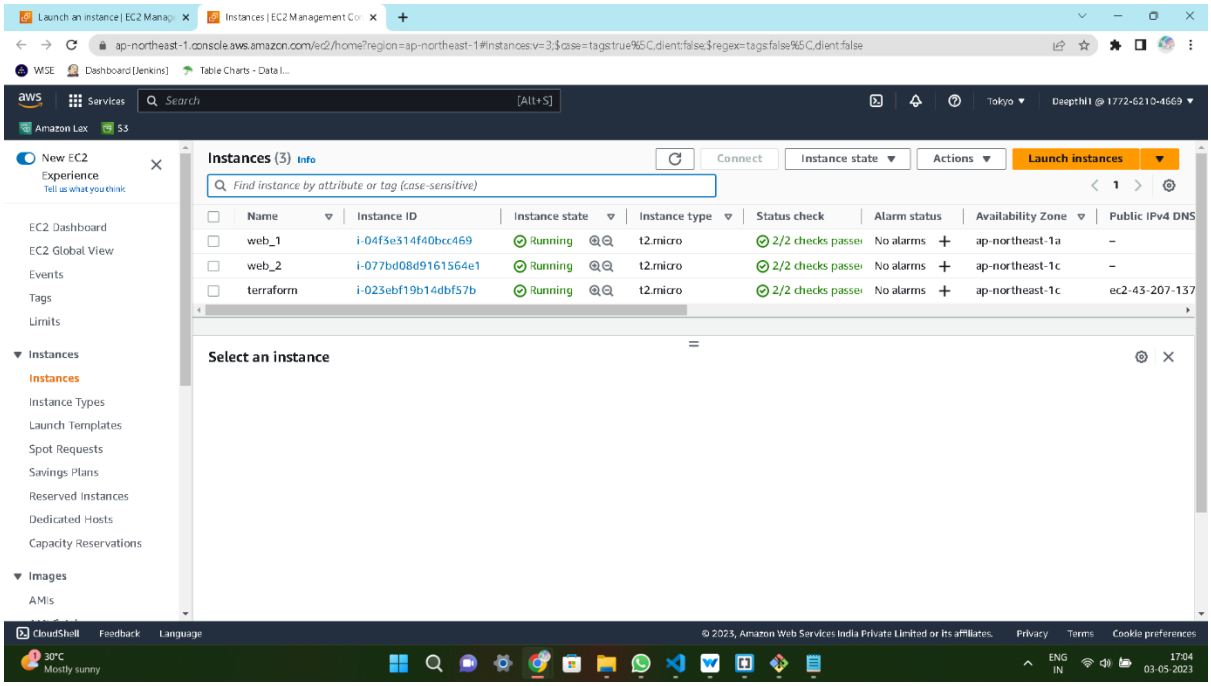




Step6: Create a file for the EC2 Instance

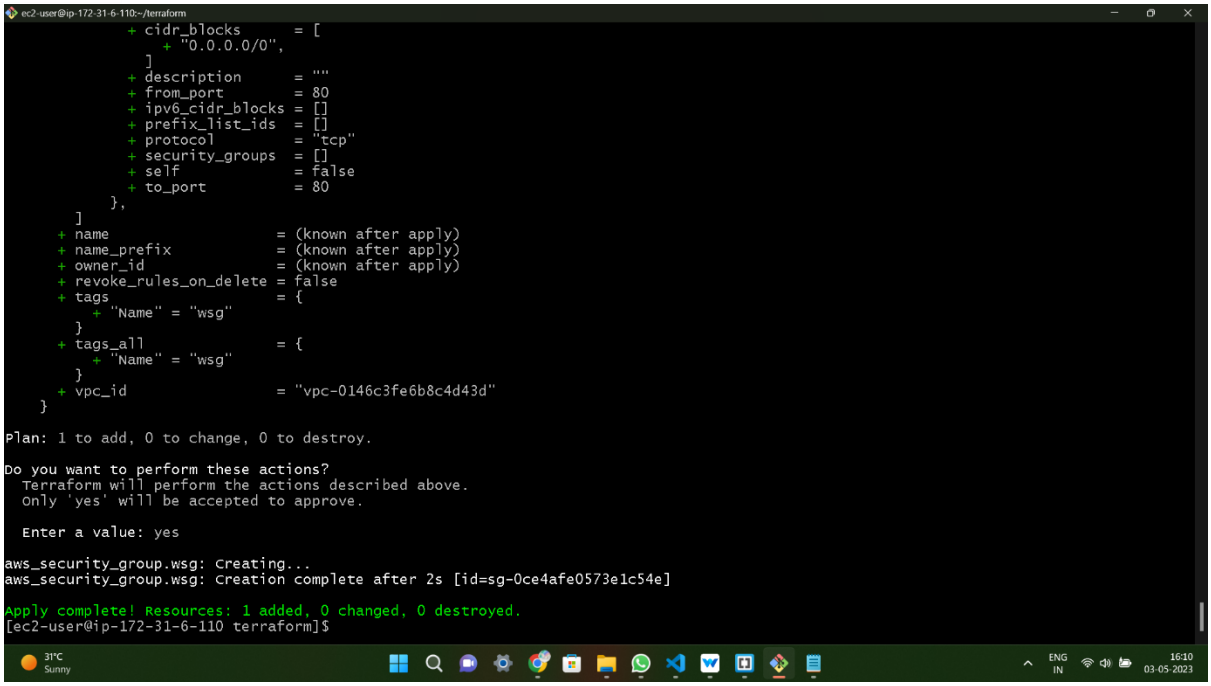
```
# creating ec2 instance in public subnet 1
resource "aws_instance" "web_1" {
    ami = "ami-0df2ca8a354185e1e"
    instance_type = "t2.micro"
    key_name = "deepu"
    vpc_security_group_ids = [aws_security_group.wsg.id]
    subnet_id = aws_subnet.pub-sub-1.id
    associate_public_ip_address = true
    user_data = "${file("userdata.sh")}"

    tags = {
      Name = "web_1"
    }

}

#creating ec2 instance in 2nd public subnet
resource "aws_instance" "web_2" {
    ami = "ami-0df2ca8a354185e1e"
    instance_type = "t2.micro"
    key_name = "deepu"
    vpc_security_group_ids = [aws_security_group.wsg.id]
    subnet_id = aws_subnet.pub-sub-2.id
    associate_public_ip_address = true
    user_data = "${file("userdata.sh")}"

    tags = {
      Name = "web_2"
    }


}
```

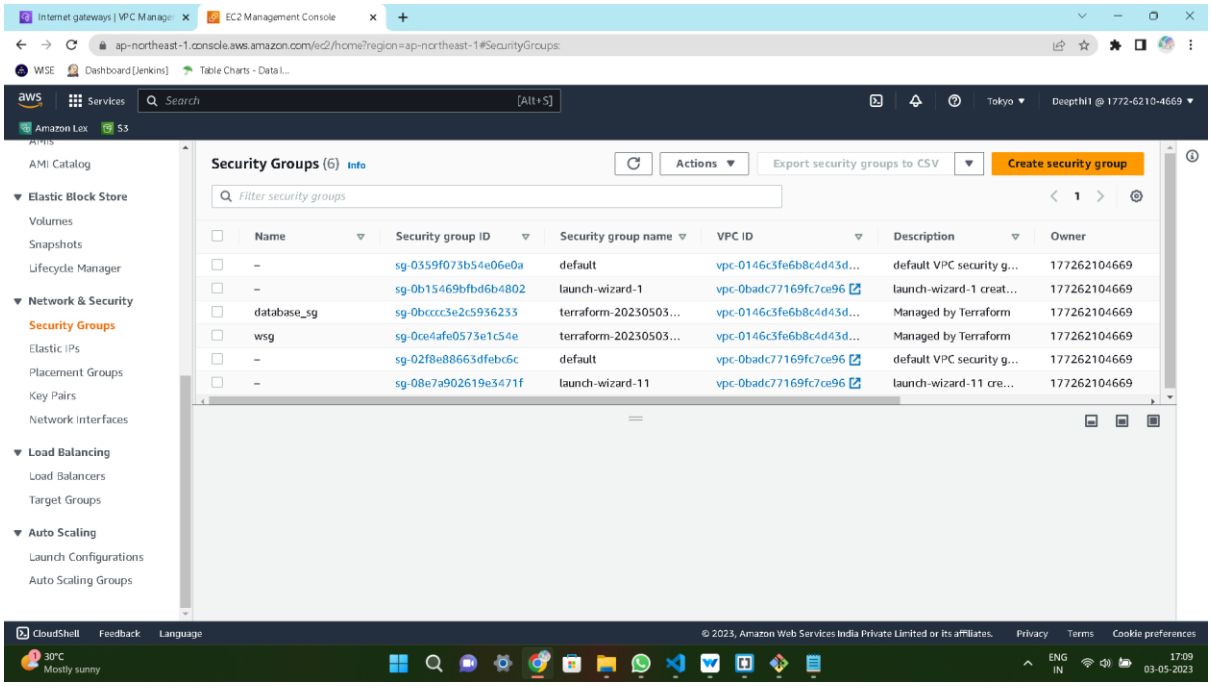




Step7: Create a file for Security Group for the Front-end tier

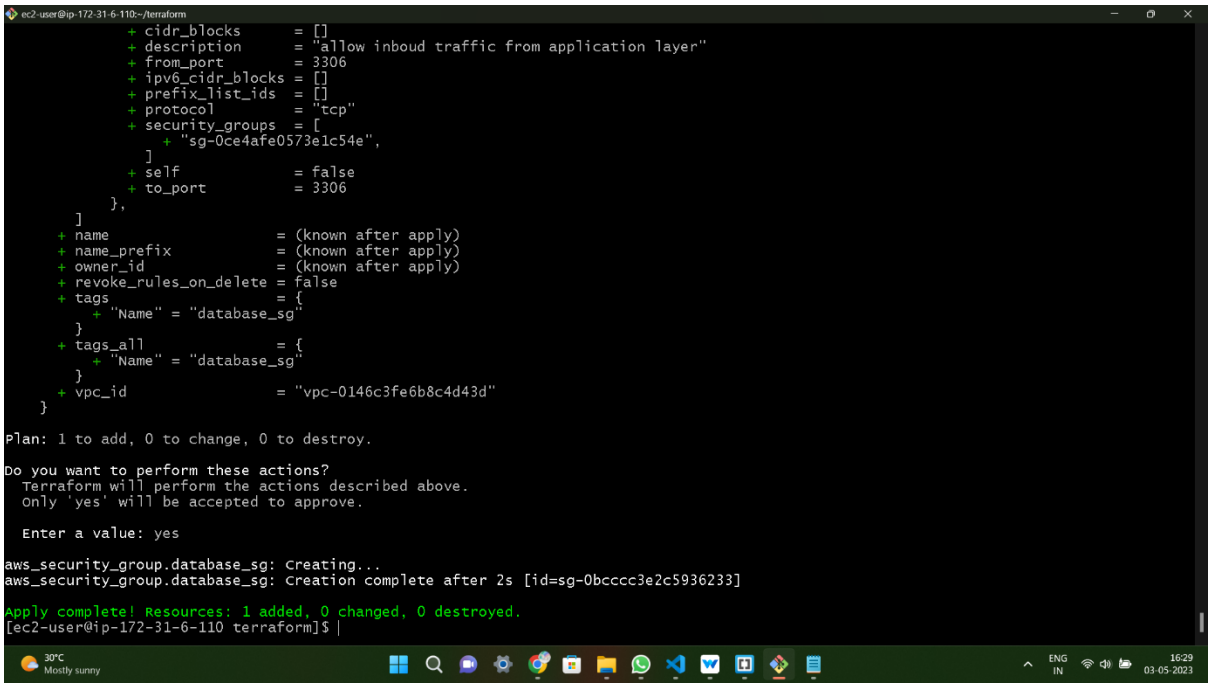
```
# creating security group for frontend tier
resource "aws_security_group" "wsg" {
  vpc_id = aws_vpc.vpc_deepu.id

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

  #HTTPS access from anywhere

  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]

  }

  #ssh access from anywhere

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  #outbound rules
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]


  }
  tags = {
    Name = "wsg"
  }



}
```

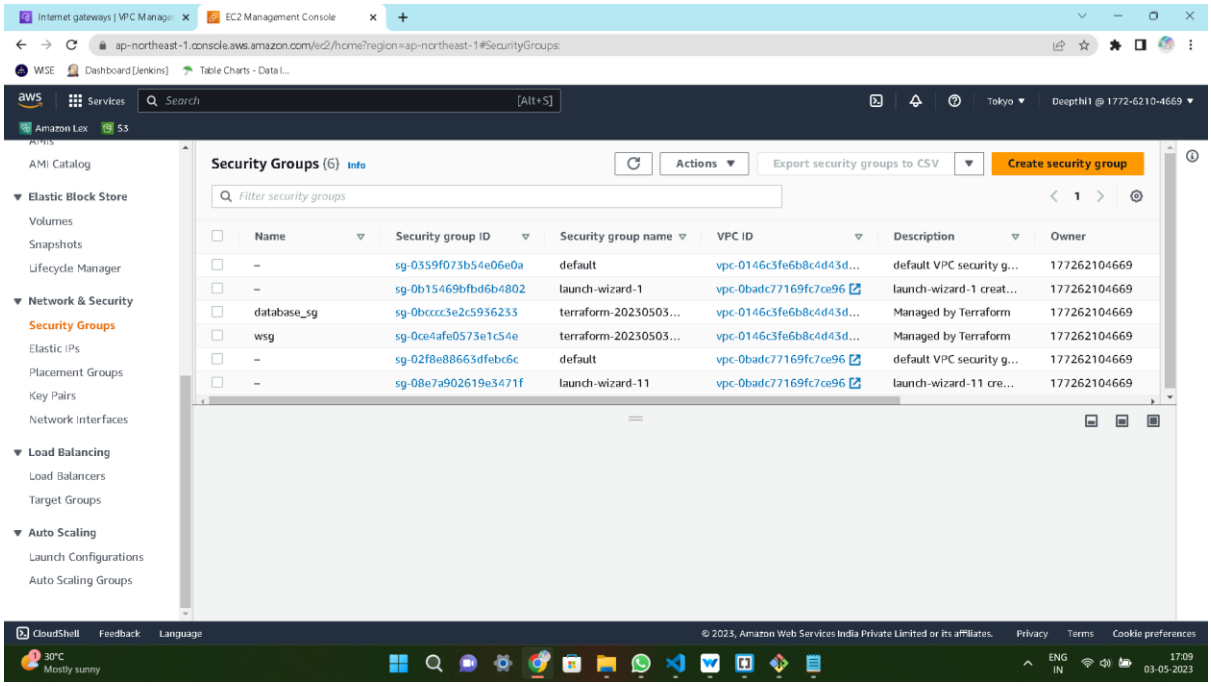




Step8: Create a file for Security Group for the Database tier

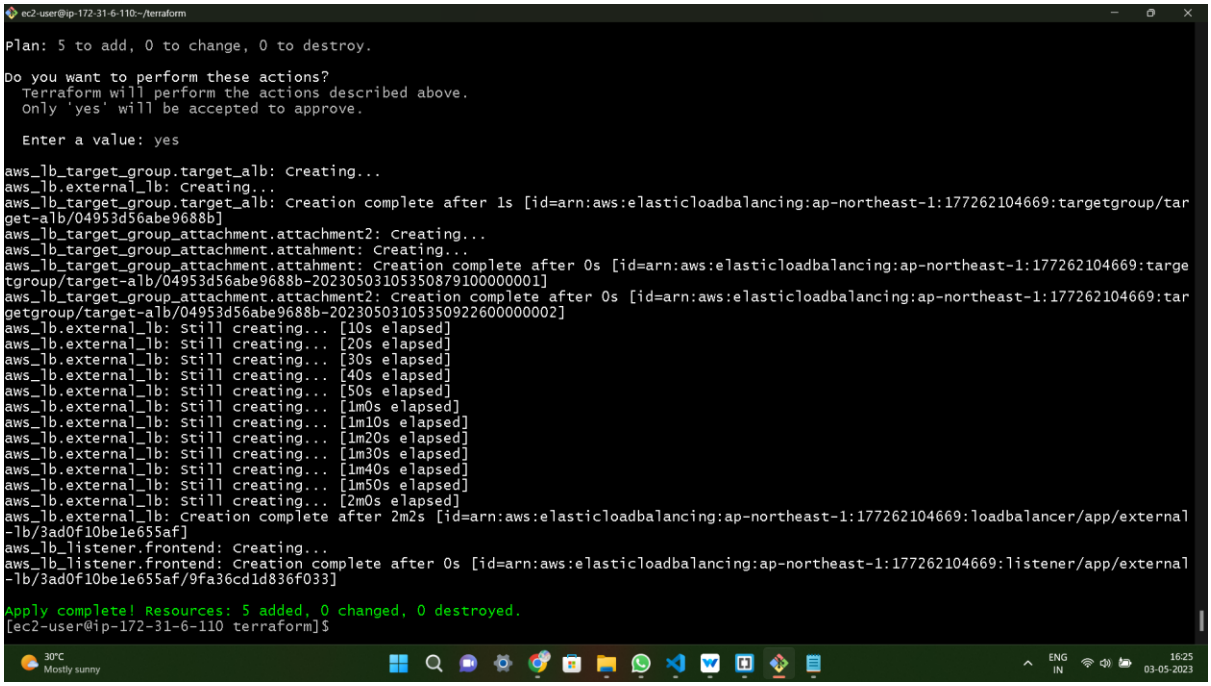
```hcl
#creating database security group
resource "aws_security_group" "database_sg" {
    vpc_id = aws_vpc.vpc_deepu.id

    ingress  {
        description = "allow inboud traffic from application layer"
        from_port = 3306
        to_port = 3306
        protocol = "tcp"
        security_groups = [aws_security_group.wsg.id]

    }
    egress {
        from_port = 32768
        to_port = 65535
        protocol = "tcp"
        cidr_blocks = [ "0.0.0.0/0" ]
    }

    tags = {
      Name = "database_sg"
    }


}
```

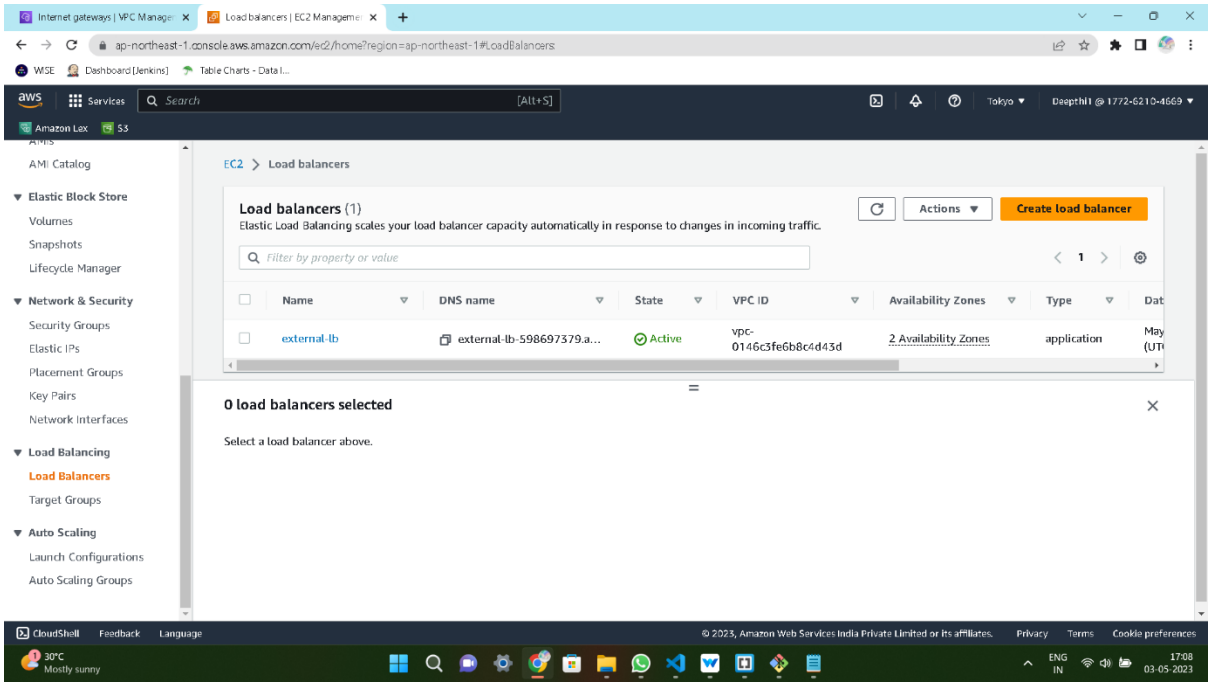




Step9: Create a file for Application Load Balancer

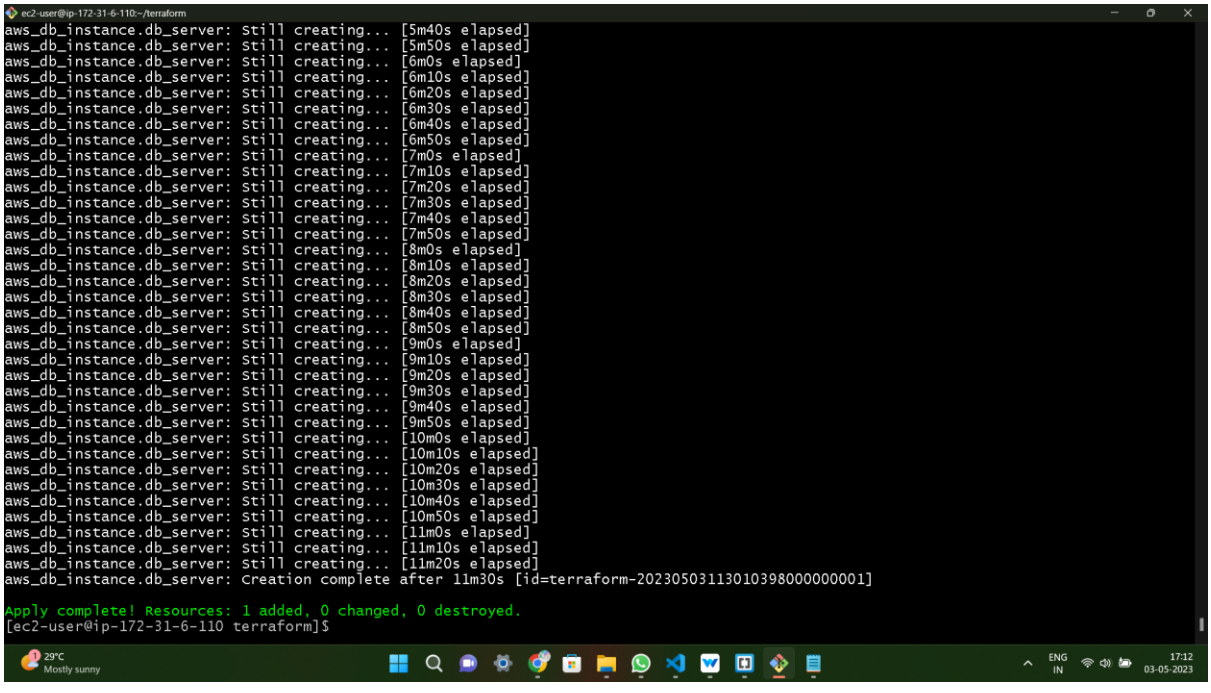
```
resource "aws_lb" "external_lb" {
  name               = "external-lb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.wsg.id]
  subnets            = [aws_subnet.pub-sub-1.id,aws_subnet.pub-sub-2.id]

}
resource "aws_lb_target_group" "target_alb" {
  name     = "target-alb"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.vpc_deepu.id

}

resource "aws_lb_target_group_attachment" "attahment" {
  target_group_arn = aws_lb_target_group.target_alb.arn
  target_id        = aws_instance.web_1.id
  port             = 80
  depends_on       = [aws_instance.web_1]

}

resource "aws_lb_target_group_attachment" "attachment2" {
  target_group_arn = aws_lb_target_group.target_alb.id
  target_id        = aws_instance.web_2.id
  port             = 80
  depends_on       = [aws_instance.web_1]

}



resource "aws_lb_listener" "frontend" {
  load_balancer_arn = aws_lb.external_lb.arn
  port              = 80
  protocol          = "HTTP"
  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.target_alb.arn
  }
}

```

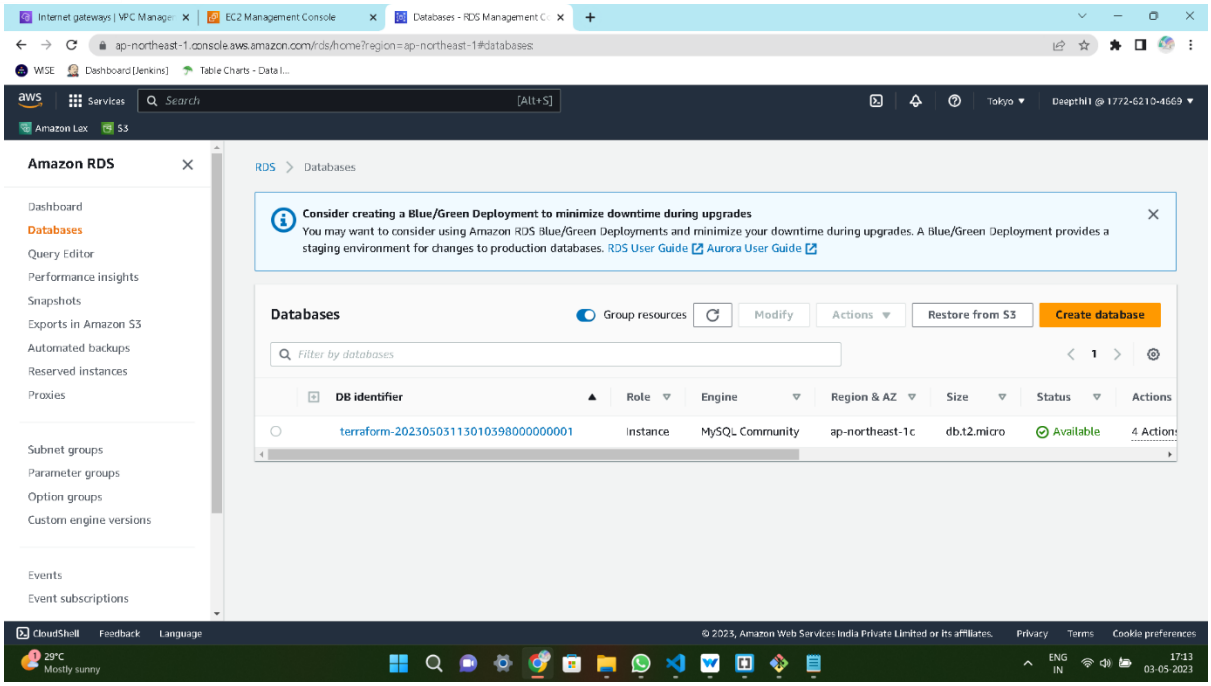




Step10: Create a file for RDS Instance

```
#creating rds instance
resource "aws_db_subnet_group" "rds" {
    name               = "main"
    subnet_ids         = [aws_subnet.dbs-1.id,aws_subnet.dbs-2.id]
    tags               = {
      Name             = "my DB subnet group"
    }
    }
    resource "aws_db_instance" "db_server" {
  allocated_storage    = 10
  db_subnet_group_name = aws_db_subnet_group.rds.id
  engine               = "mysql"
  engine_version       = "5.7"
  instance_class       = "db.t2.micro"
  multi_az             = true
  username             = "username"
  password             = "password"
  skip_final_snapshot  = true
  vpc_security_group_ids = [aws_security_group.wsg.id]
}
```





Step11: Create a file for Outputs

```
output "lb_dns_name" {
    description = "DNS OF LOAD BALANCER"
    value = "${aws_lb.external_lb.dns_name}"

}
```

Step12: Create a file for Variable

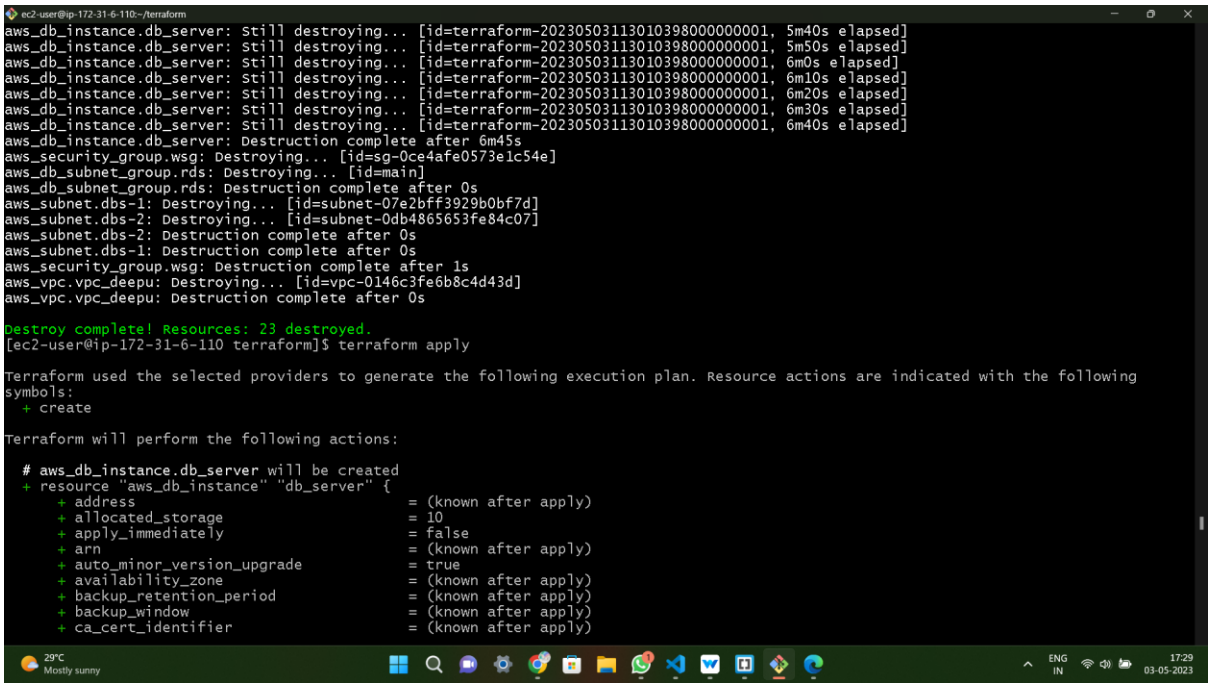
```
# defining cidr block for vpc

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

}


variable "websub_1" {
    default = "10.0.0.0/18"

}


variable "websub_2" {
    default = "10.0.128.0/18"

}


variable "appsub_1" {
    default = "10.0.96.0/20"

}


variable "appsub_2" {
    default = "10.0.112.0/20"

}
variable "dbsub_1" {
    default = "10.0.80.0/21"

}
variable "dbsub_2" {
    default = "10.0.88.0/21"

}

```

Step13: Create a file for User Data

```
#!/bin/bash
sudo yum update -y
sudo yum install httpd -y
sudo yum install git -y
sudo systemctl start httpd
sudo systemctl enable httpd
echo "hello world from $(hostname -f)" > /var/www/html/index.html
```

Step14: Verify the Resources

```
pply complete! Resources: 17 added, 0 changed, 0 destroyed.

utputs:
```



- All the resoures like vpc,subnets,internet gateway, route table,ec2 instances, rds,alb,security groups have been applied by terraform and verified as shown in above screenshot.