PROJECT-I

Deploy Three-Tier Architecture in AWS using Terraform

Prerequisites:

- Basic knowledge of AWS & Terraform.
- AWS account.
- I AM user.
- GitHub account.
- AWS Access & Secret key.

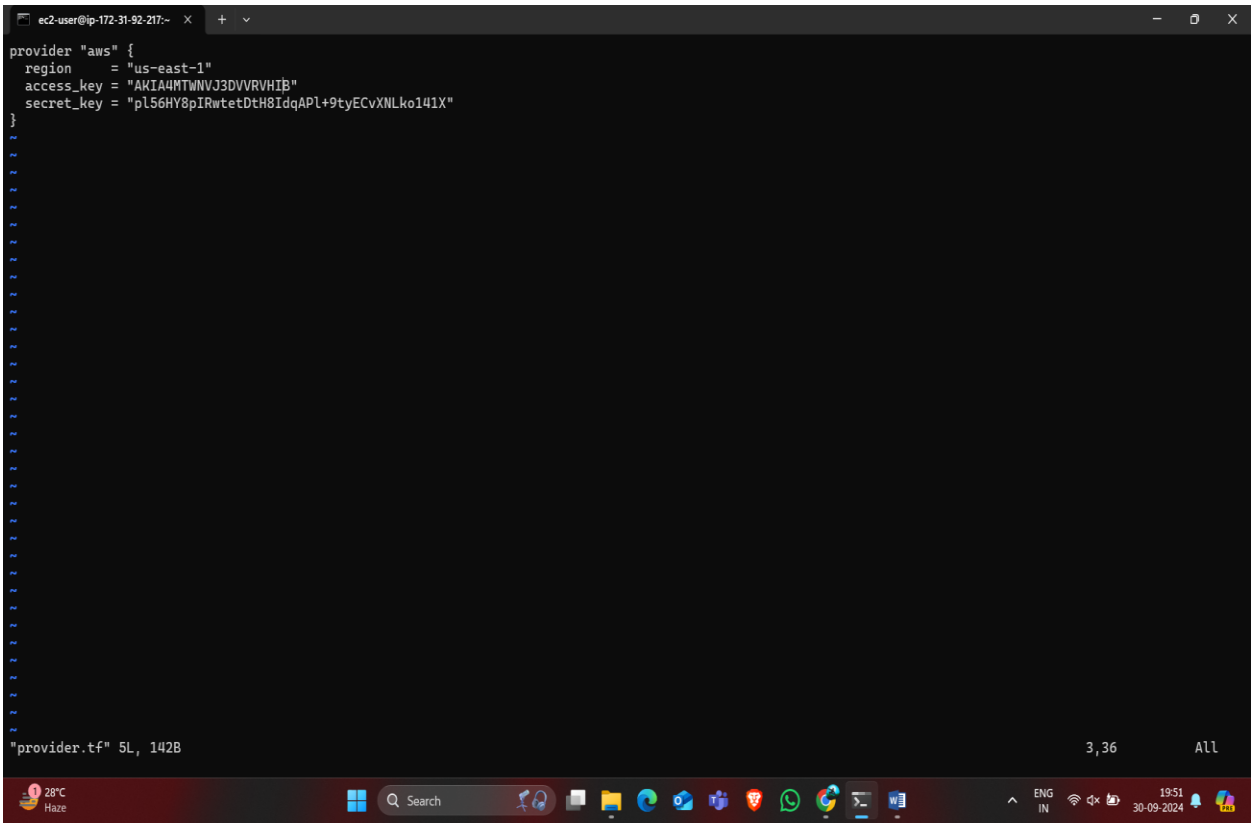
List of Steps in the Pipeline:

- Create a provider file.
- Create a variable file.
- Create a file for VPC.
- Create a file for subnet.
- Create a file for Internet gatway.
- Create a file for Route table
- Create a user data files.
- Create a file for EC2 instances.
- Create a file for Security groups for the front-end tier.
- Create a file for security groups for database tier.
- Create a file for Application load balancer.
- Create a file for Auto scaling group.
- Create a file for the RDS instance.
- Create a file for outputs.
- Verify the resources.

- We have to login to the AWS account and open the IAM account than create the access & secret keys.
- After that we have to store that keys in the file.
- Launch a new instance and connect to the terminal than provide the files in that instance.
- Before that install the terraform in the terminal.

❖ Creating a provider file :

- Create the provider.tf file in the terminal by using command

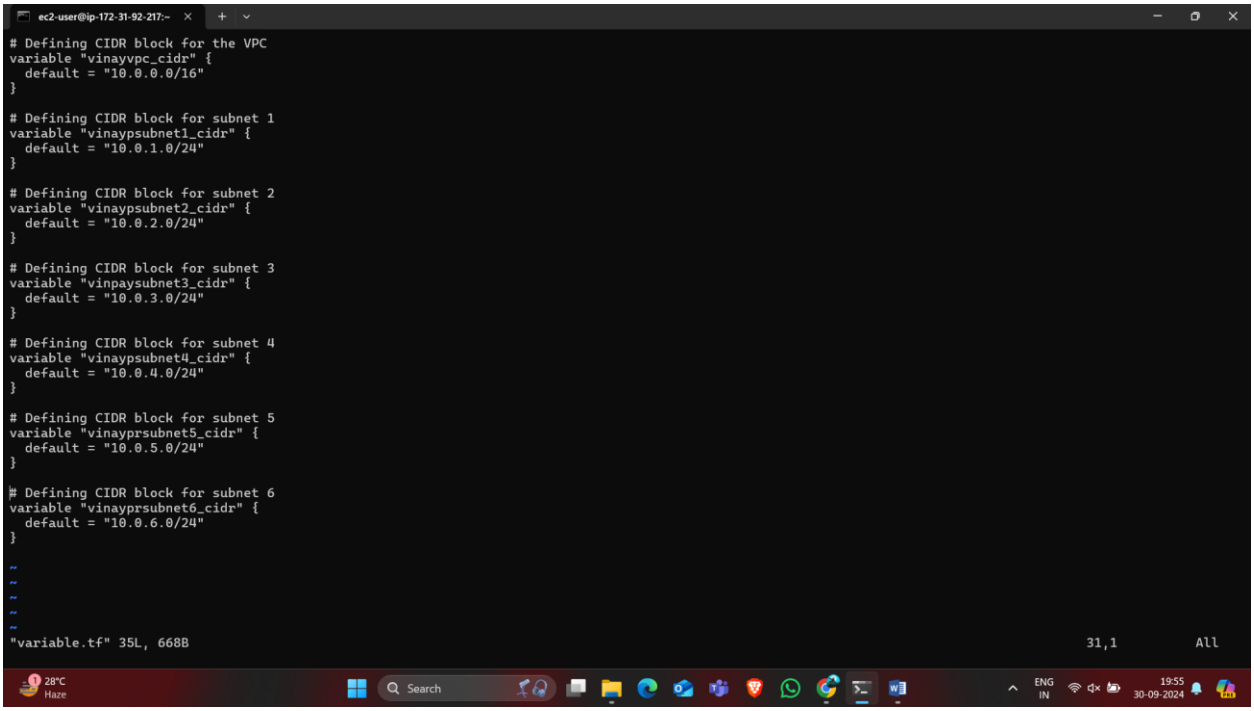


- Provide the terraform init ,validate,plan and apply.

❖ Creating the variable file :

- Create the variable file for storing the cidr blocks information and vpc,subnets information by using command

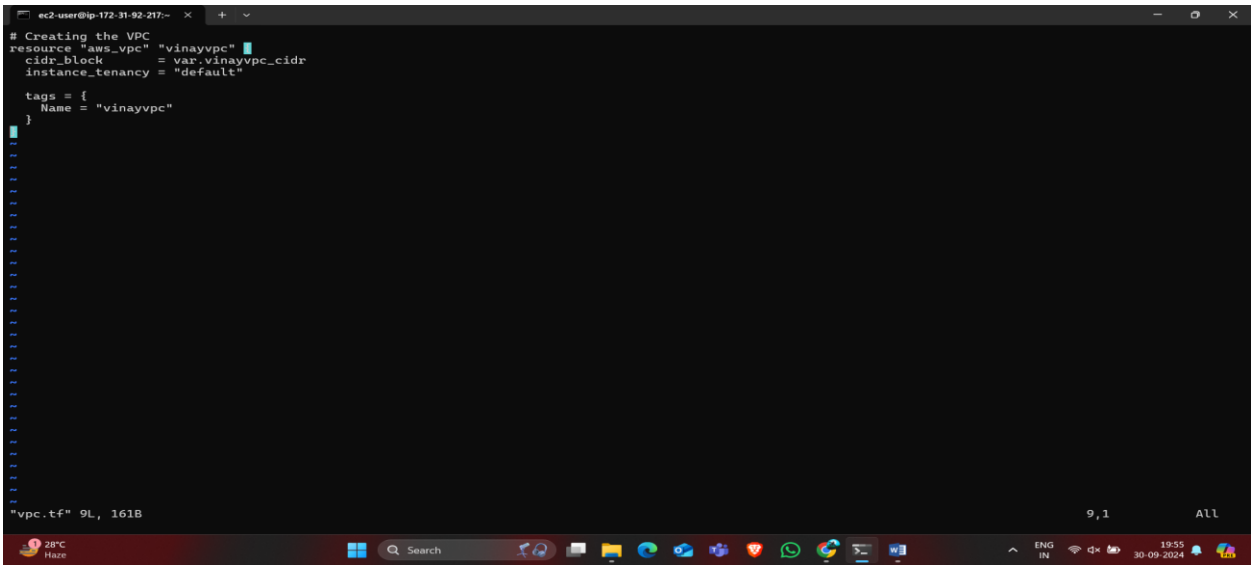
“vi variable.tf” and write below code.



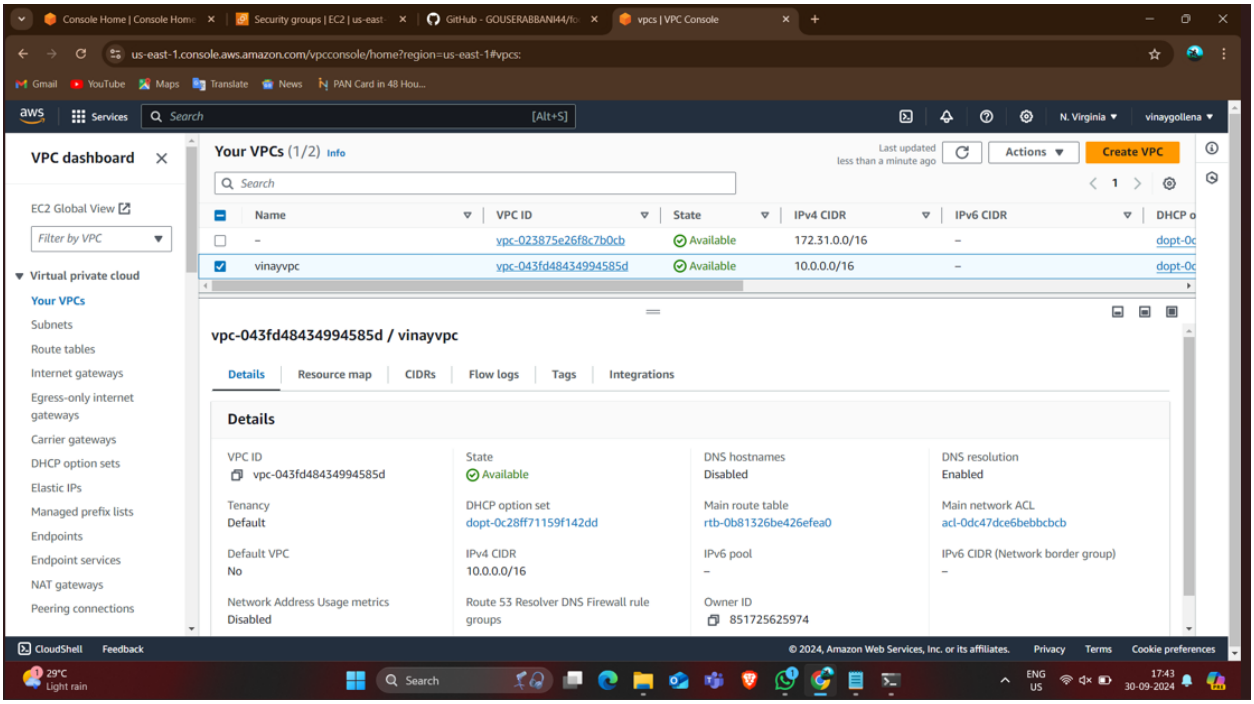
- Give the terraform init,validate,plan and apply.

❖ Creating the vpc file :

- Create the vpc.tf and write the below code,this is for creating the vpc in the aws console by using terraform code.

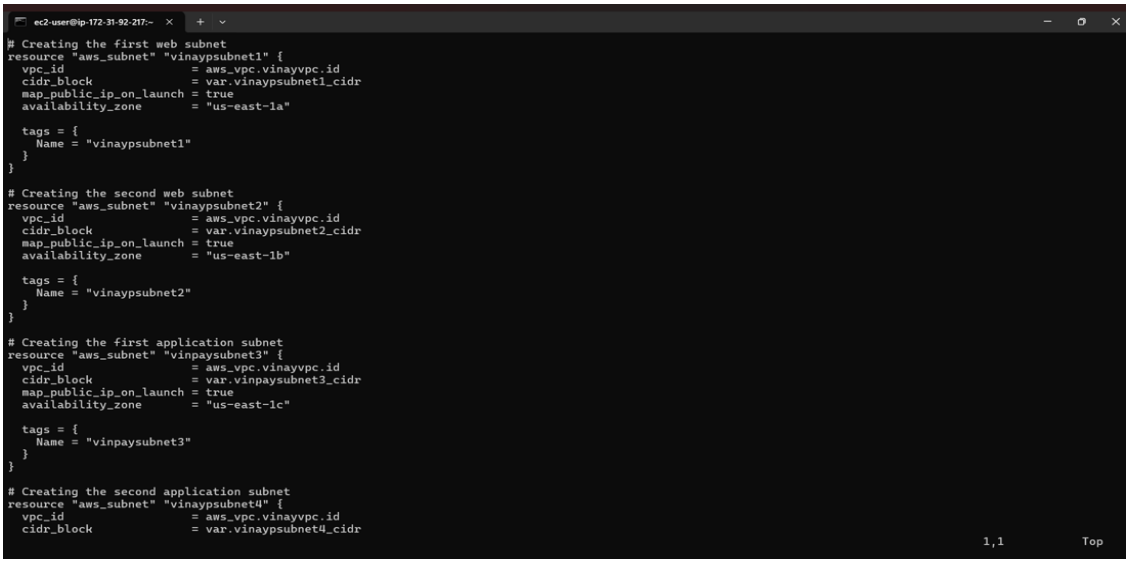


- Provide the terraform commands.
- And we can observe the created vpc in the below image.

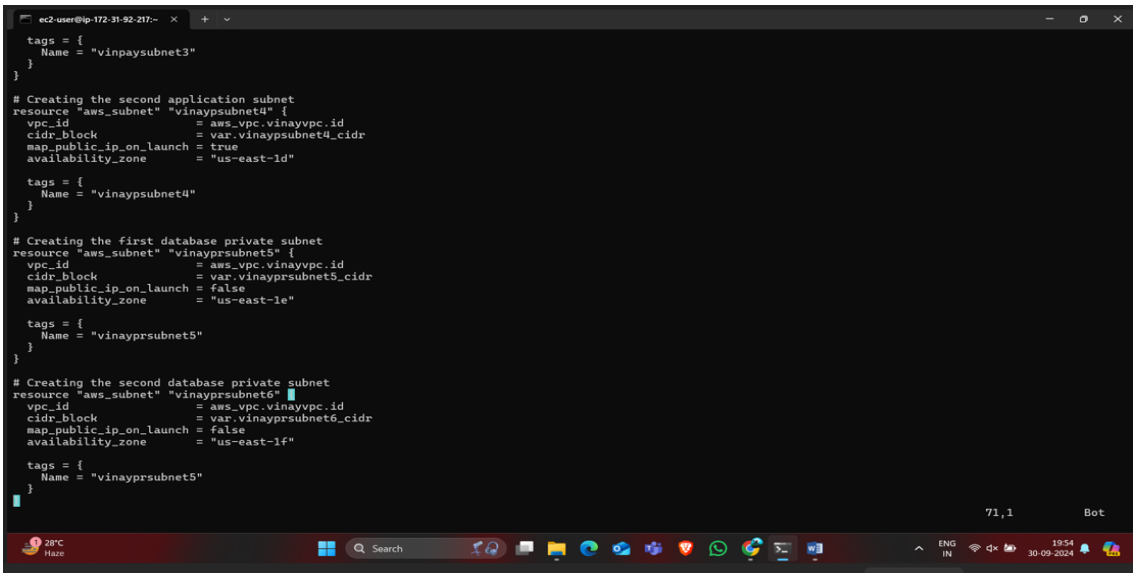


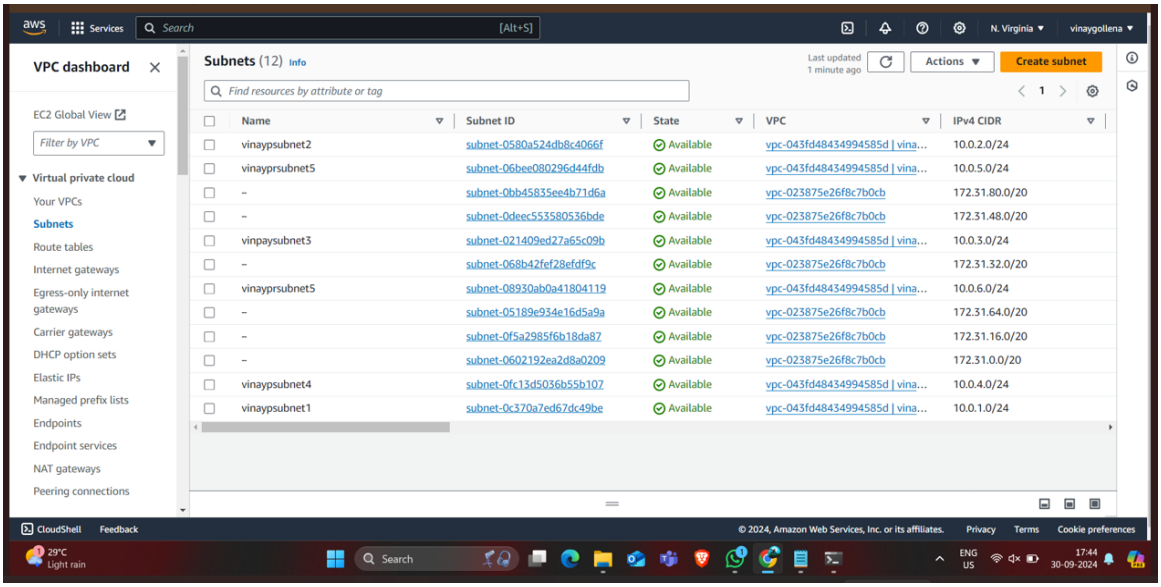
❖ Creating the subnets file:

- Create the subnet.tf file in the terminal by using “vi subnets.tf”.
- Writhe below code.



```
# Creating the first web subnet
resource "aws_subnet" "vinaypsubnet1" {
  vpc_id                  = aws_vpc.vinayvpc.id
  cidr_block              = var.vinaypsubnet1_cidr
  map_public_ip_on_launch = true
  availability_zone       = "us-east-1a"

  tags = {
    Name = "vinaypsubnet1"
  }
}

# Creating the second web subnet
resource "aws_subnet" "vinaypsubnet2" {
  vpc_id                  = aws_vpc.vinayvpc.id
  cidr_block              = var.vinaypsubnet2_cidr
  map_public_ip_on_launch = true
  availability_zone       = "us-east-1b"

  tags = {
    Name = "vinaypsubnet2"
  }
}

# Creating the first application subnet
resource "aws_subnet" "vinpaysubnet3" {
  vpc_id                  = aws_vpc.vinayvpc.id
  cidr_block              = var.vinpaysubnet3_cidr
  map_public_ip_on_launch = true
  availability_zone       = "us-east-1c"

  tags = {
    Name = "vinpaysubnet3"
  }
}

# Creating the second application subnet
resource "aws_subnet" "vinaypsubnet4" {
  vpc_id                  = aws_vpc.vinayvpc.id
  cidr_block              = var.vinaypsubnet4_cidr
                                                    1,1           Top
```

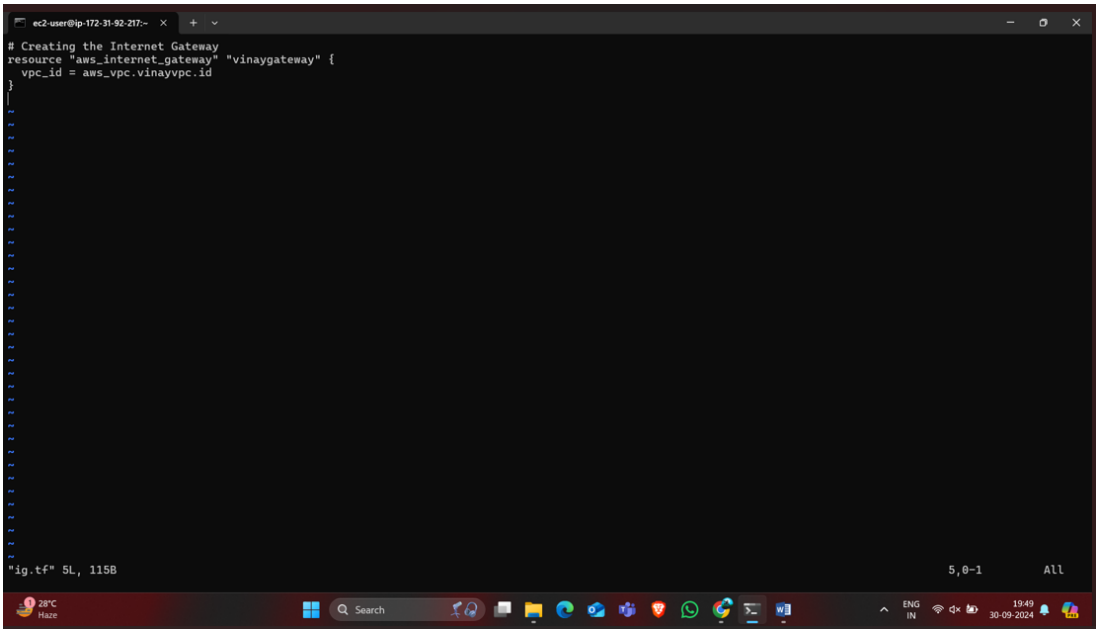




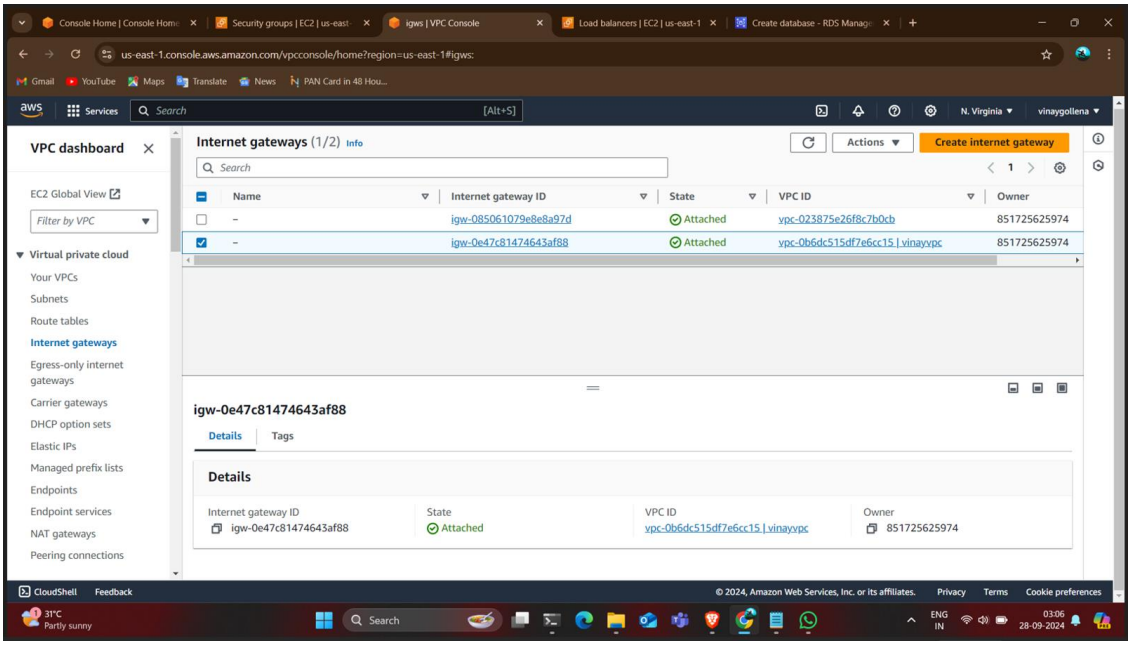
- We can observe the created subnets by using terraform code.

❖ Creating the internet gateway file:

- Create the internet gateway file “vi igw.tf”
- Write the below code



- Provide the terraform commands.
- Than observe the created internet gatway in the image.

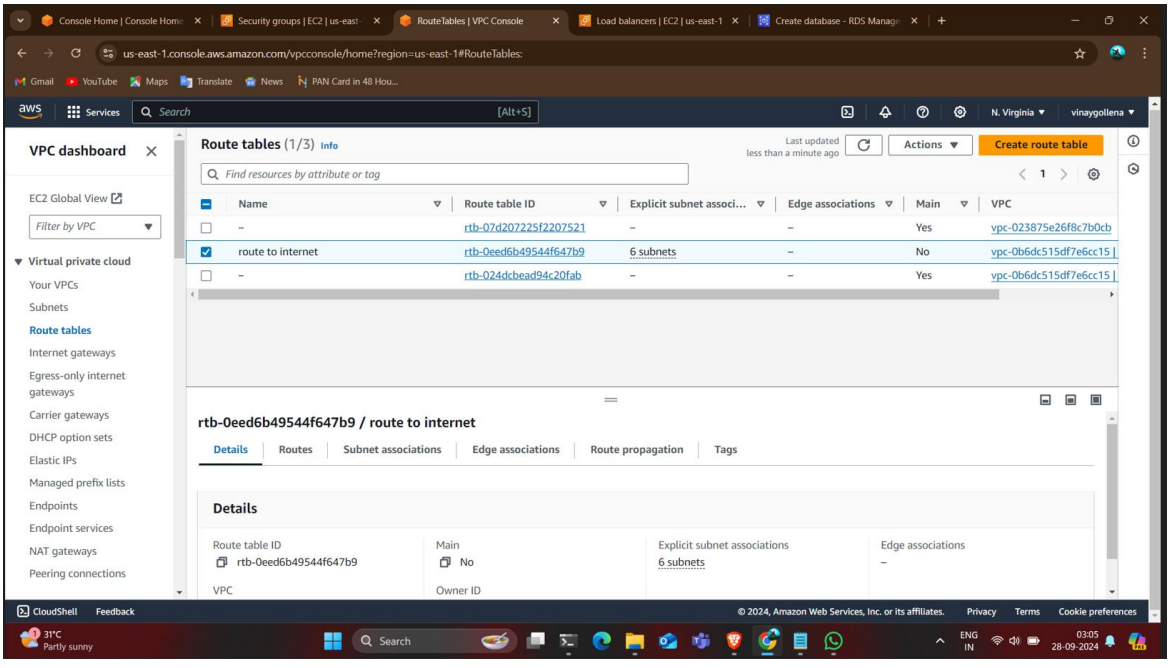


❖ Creating the route table file :

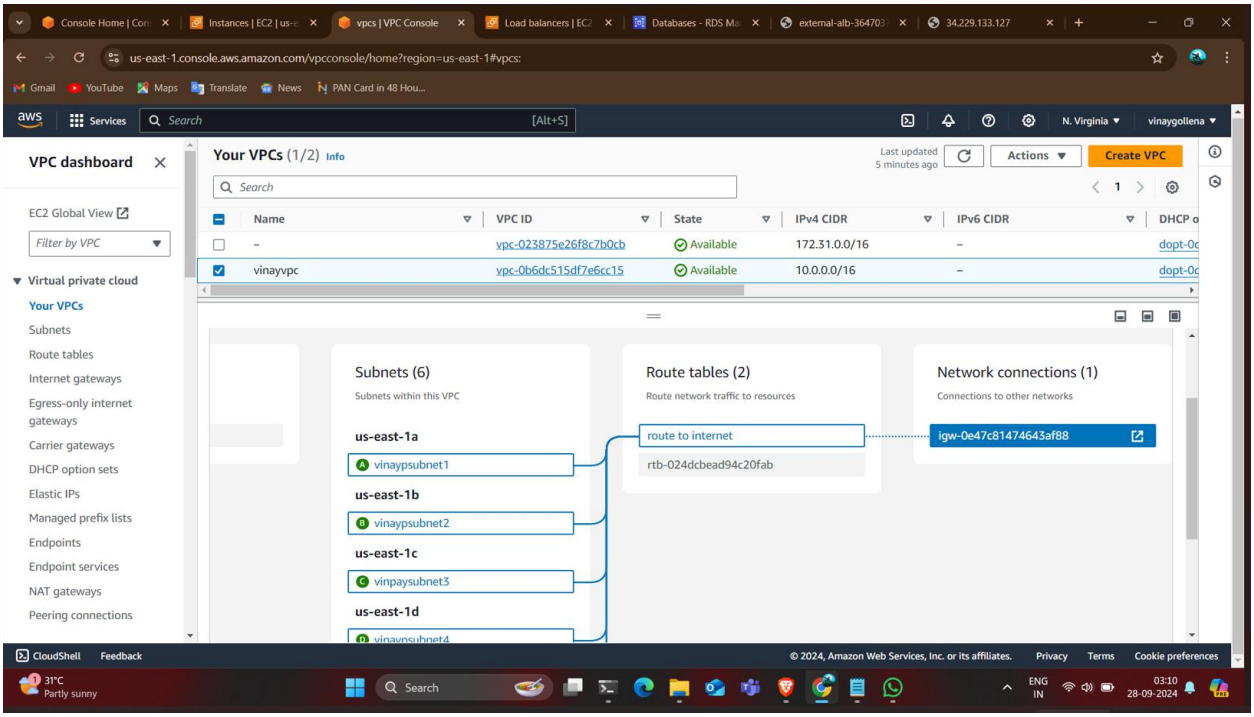
- Create the route table file “vi rt.tf”.than write the code.

```
# Creating a route table
resource "aws_route_table" "vinayroute" {
  vpc_id = aws_vpc.vinayvpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.vinaygateway.id
  }
  tags = {
    Name = "route to internet"
  }
}
# Associating route table with the first subnet
resource "aws_route_table_association" "vinayrt1" {
  subnet_id      = aws_subnet.vinaypsubnet1.id
  route_table_id = aws_route_table.vinayroute.id
}
# Associating route table with the second subnet
resource "aws_route_table_association" "vinayrt2" {
  subnet_id      = aws_subnet.vinaypsubnet2.id
  route_table_id = aws_route_table.vinayroute.id
}
# Associating route table with the 3rd subnet
resource "aws_route_table_association" "vinayrt3" {
  subnet_id      = aws_subnet.vinpaysubnet3.id
  route_table_id = aws_route_table.vinayroute.id
}
# Associating route table with the 4th subnet
resource "aws_route_table_association" "vinayrt4" {
  subnet_id      = aws_subnet.vinaypsubnet4.id
  route_table_id = aws_route_table.vinayroute.id
}
resource "aws_route_table_association" "vinayrt5" {
  subnet_id      = aws_subnet.vinayprsubnet5.id
  route_table_id = aws_route_table.vinayroute.id
}
resource "aws_route_table_association" "vinayrt6" {
  subnet_id      = aws_subnet.vinayprsubnet6.id
  route_table_id = aws_route_table.vinayroute.id
}

-- INSERT --                                                    24,2          All
```

- Provide the terraform commands.
- Observe the below image that route tablke has been created.



- After creating the VPC,subnets,Internet gateway and route table check the resources
- Check the connections if the network is passing or not.

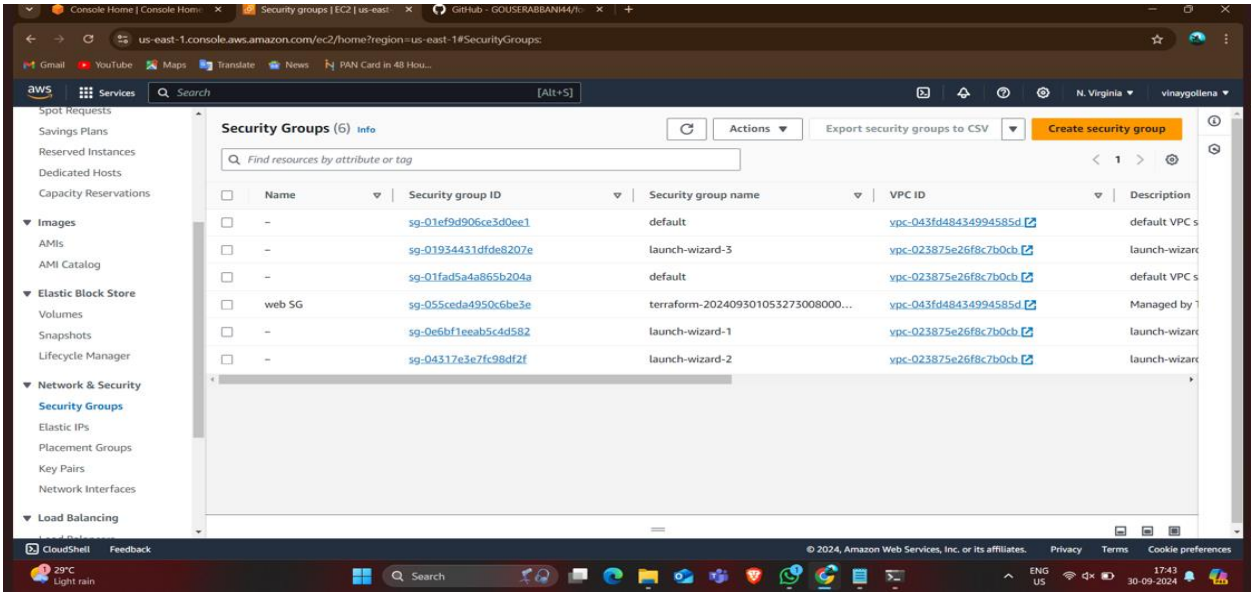


❖ Creating the Security groups for the Frint-end tier:

- Create the security groups file “vi sg.tf”.
- Write the code for the security groups

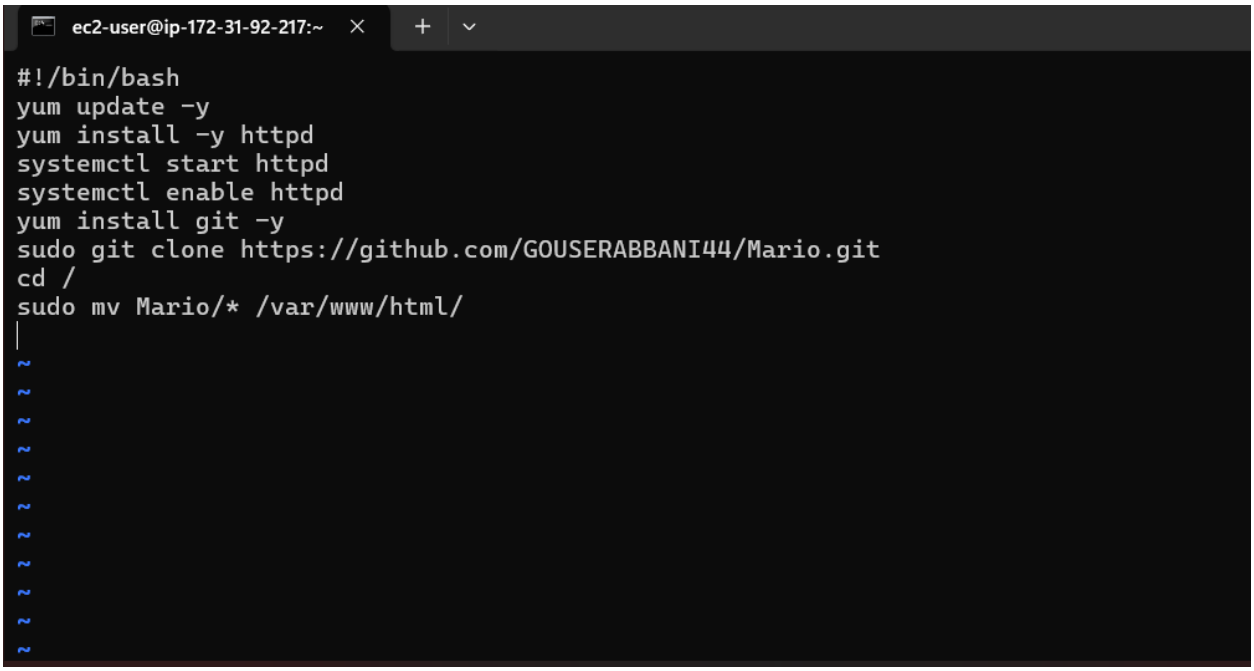
```
resource "aws_security_group" "vinaysg" {
  vpc_id = aws_vpc.vinayvpc.id

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

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

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

  # Egress rule for all outbound traffic
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "web SG"
  }
}
~
~
~
"sg.tf" 37L, 628B                                             1,1           All
```

- Provide the terraform commands.
- Observe the below image that creating the security groups.



❖ Creating the user data files :

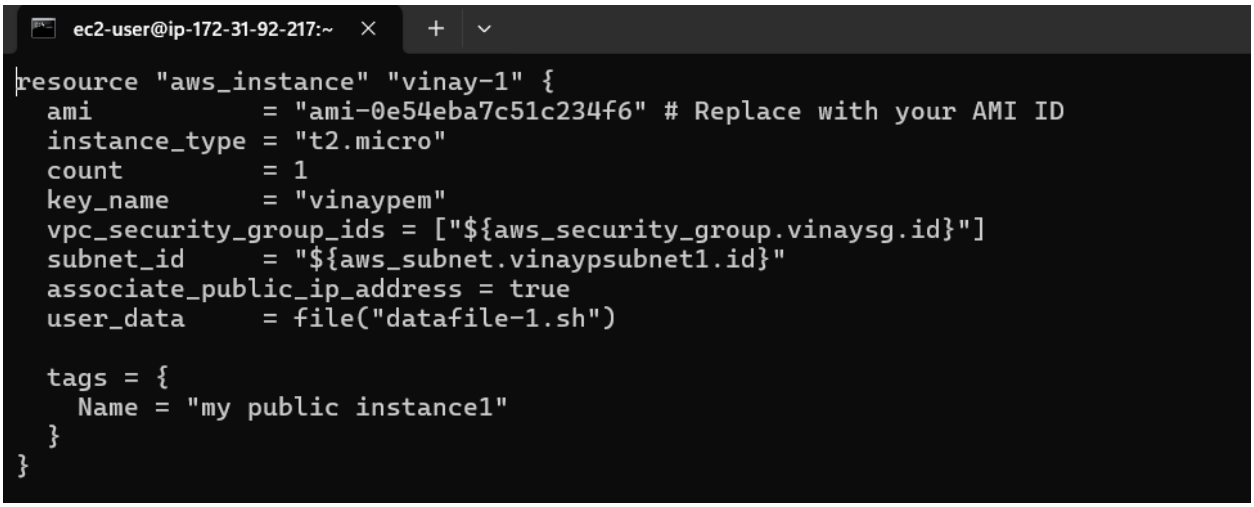
- Create the data file-1 and data file-2 for 2 instances with "vi data1.sh" and "vi data2.sh".
- Write the codes given below
- Data file-1



- Data file-2

❖ Creating the EC2 instances file:

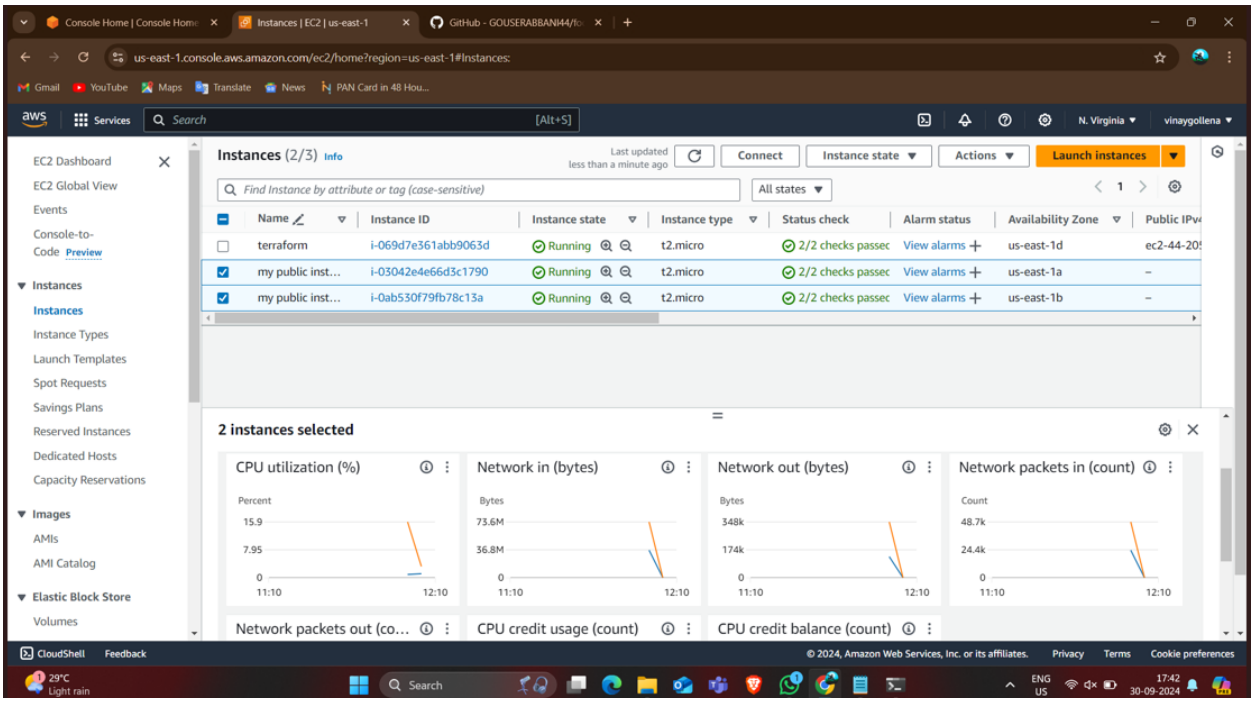
- Create the file “vi instances.tf”.
- Writhe code in the terminal



Creating 2nd EC2 instance in public subnet

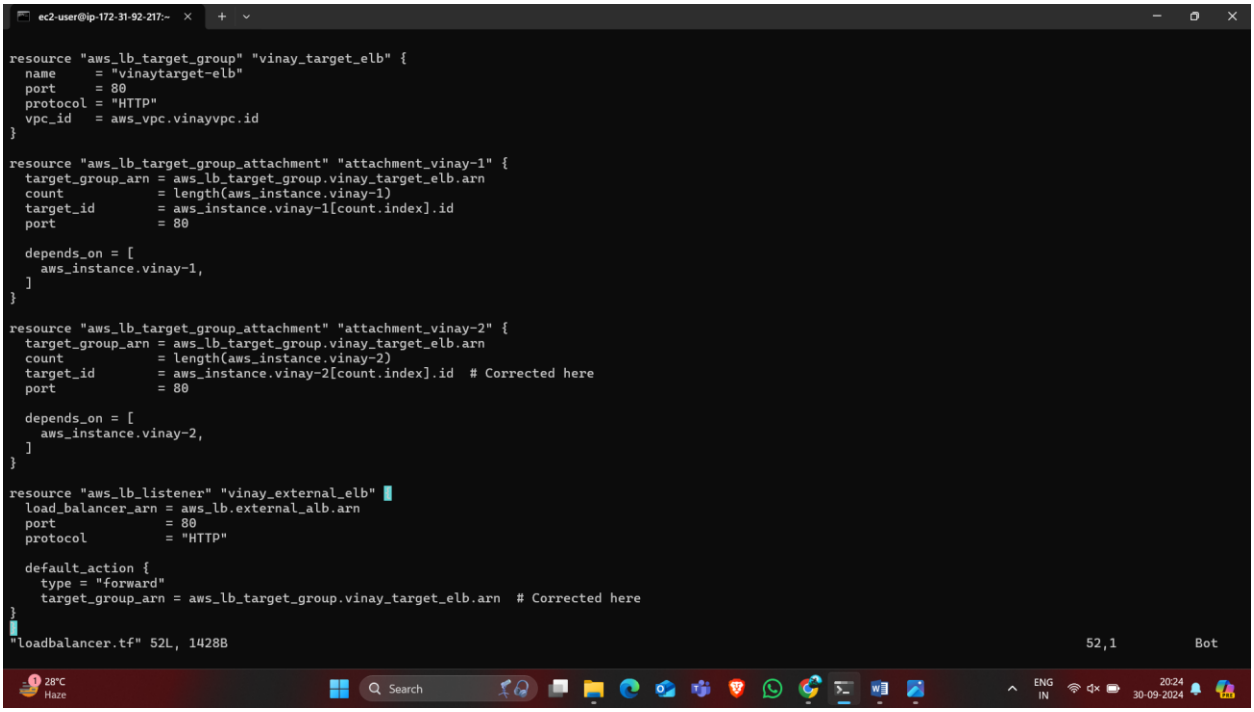
```
ec2-user@ip-172-31-92-217:~
# Creating 2nd EC2 instance in public subnet
resource "aws_instance" "vinay-2" {
  ami           = "ami-0e54eba7c51c234f6" # Replace with your AMI ID
  instance_type = "t2.micro"
  count         = 1
  key_name      = "vinaypem"
  vpc_security_group_ids = ["${aws_security_group.vinaysg.id}"]
  subnet_id     = "${aws_subnet.vinaypsubnet2.id}"
  associate_public_ip_address = true
  user_data     = file("datafile-2.sh")

  tags = {
    Name = "my public instance2"
  }
}
```

- Provide the terraform command.
- Observe the below image that instances has been created by using terraform script.

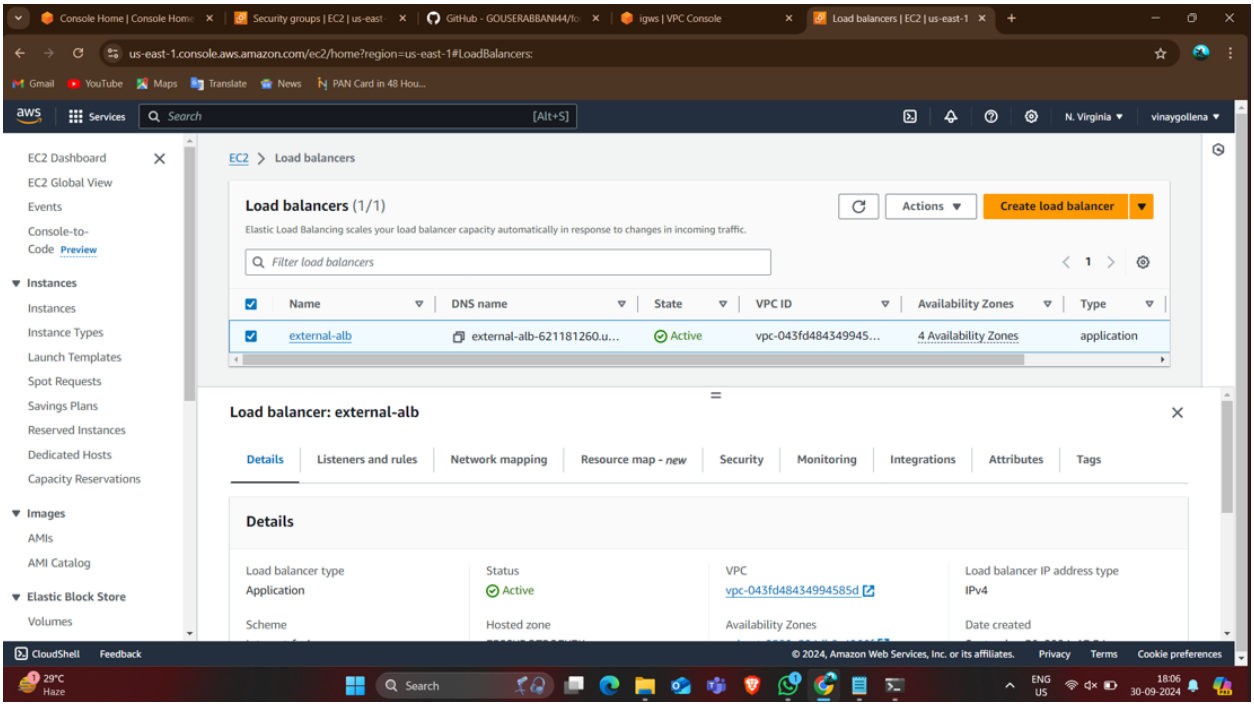


❖ Creating the Application Load balancer file:

- Create the Application load balancer file “vi ALB.tf”.
- Write the code in the terminal

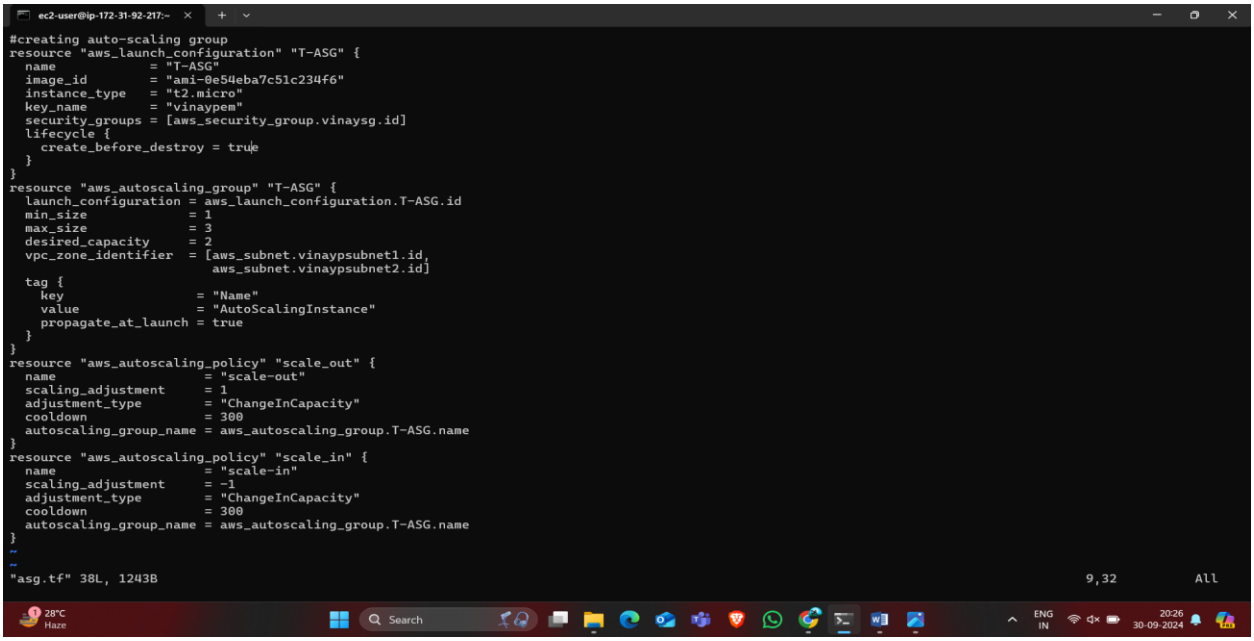


- Provide the terraform commands and observe the below image.

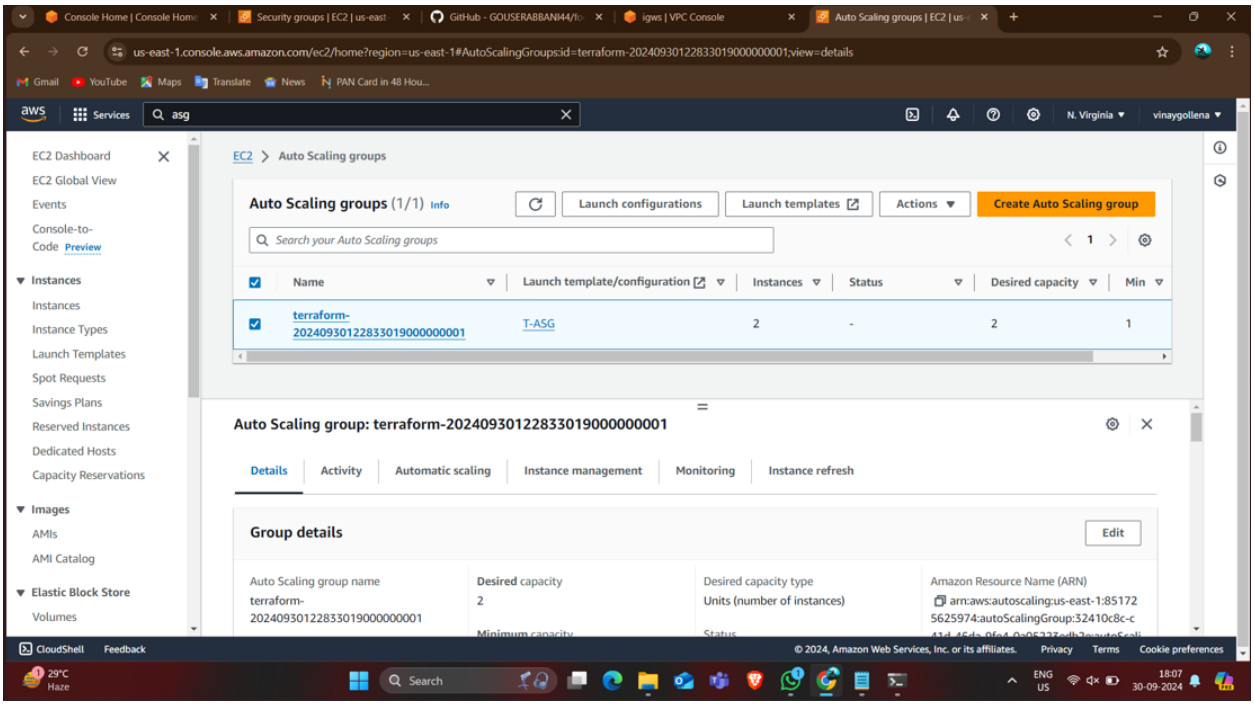


❖ Creating the Auto Scaling Group :

- Create the Auto scaling group “vi ASG.tf”
- Write the code given below

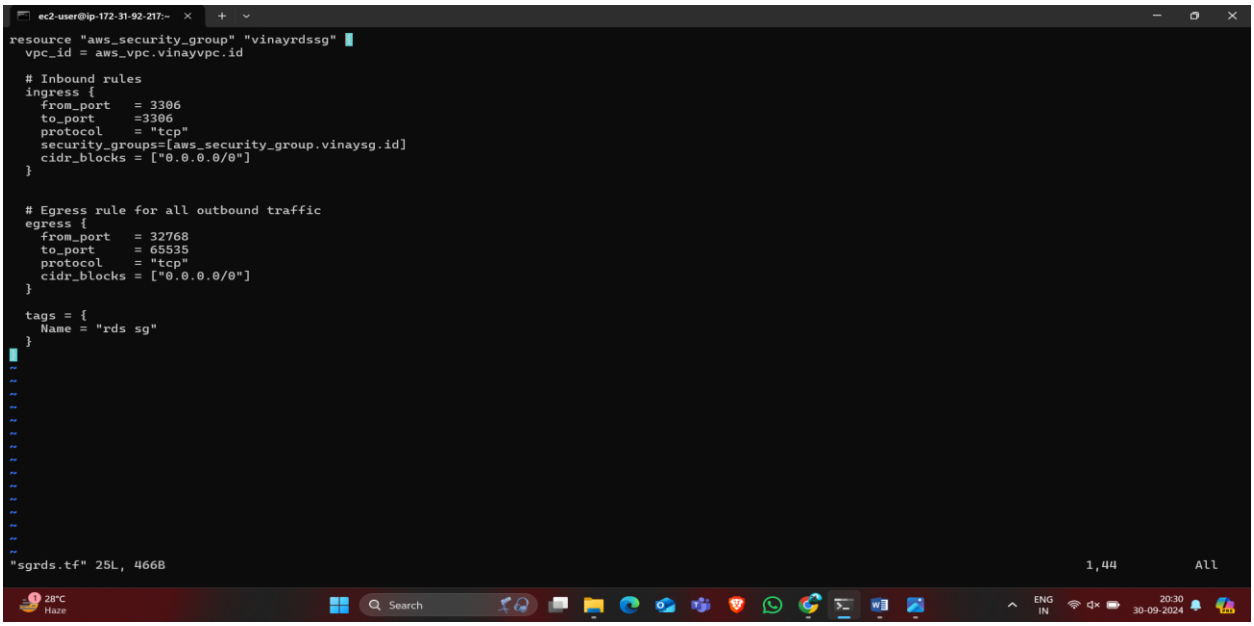


- Provide the terrafrom commands and observe in the aws consol.
- Instances are also created.



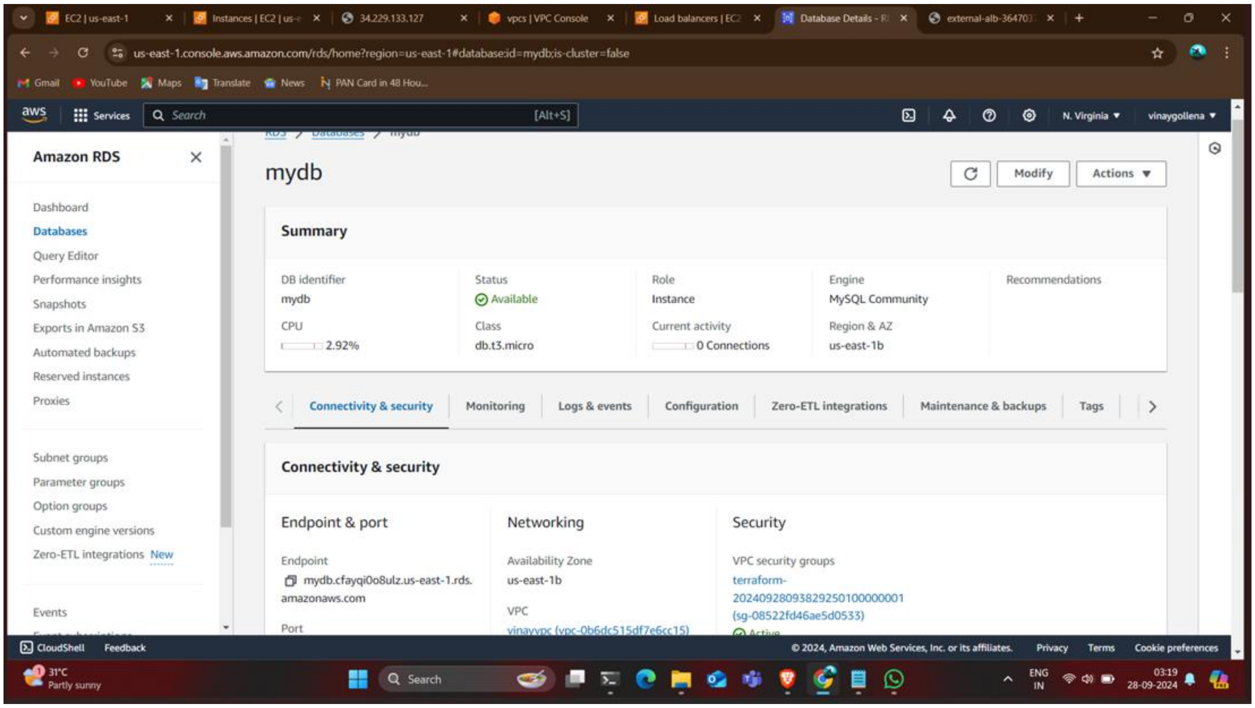
❖ Creating the RDS security group for the database tier :

- Create the Security groups file for RDS “vi RDSsg.tf”.
- Write the code and provide the appropriate port numbers.



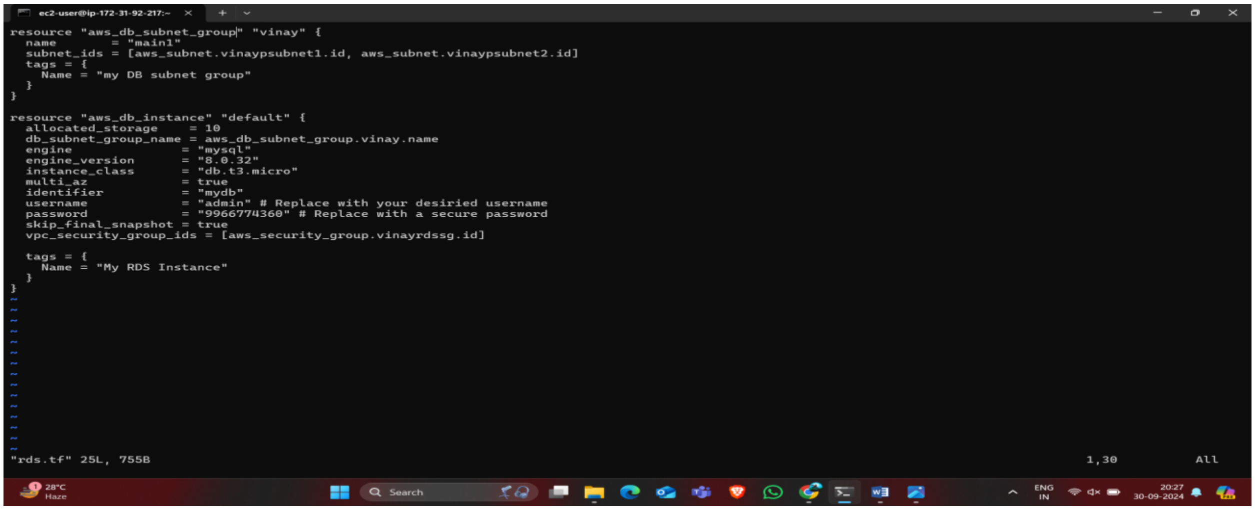
- Provide the terraform commands.

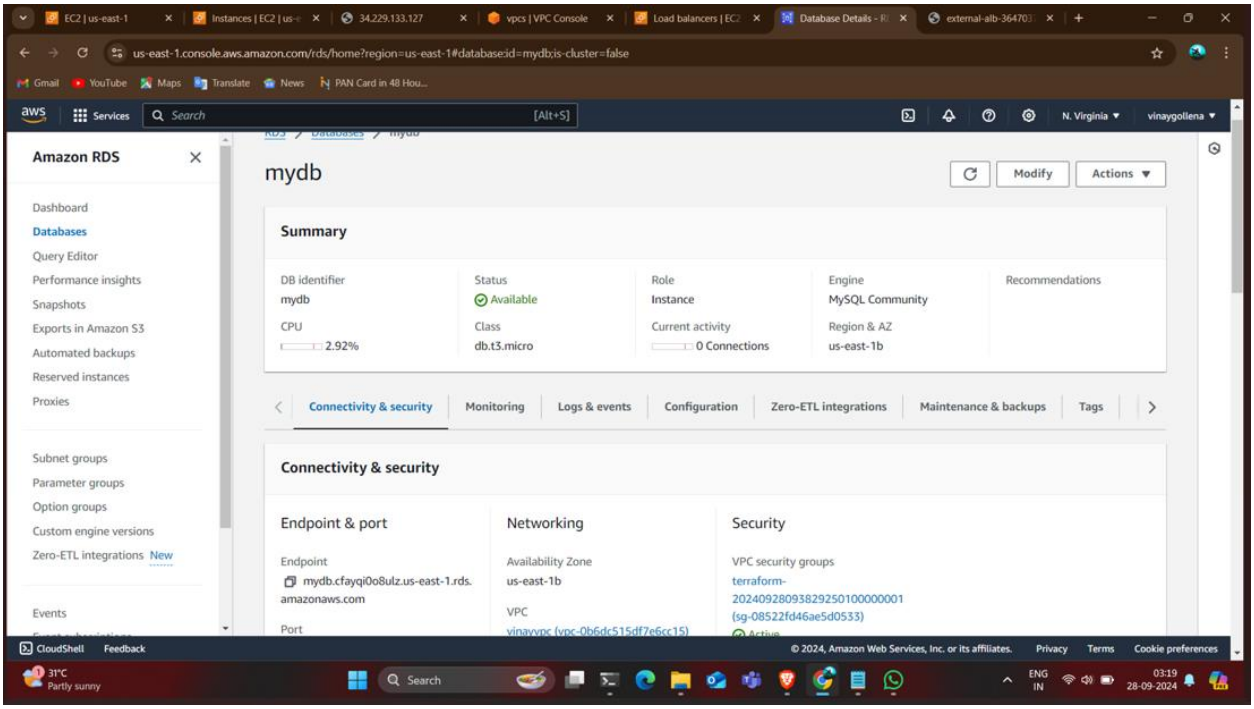
- Observe in the aws cansol.



❖ Creating the RDS :

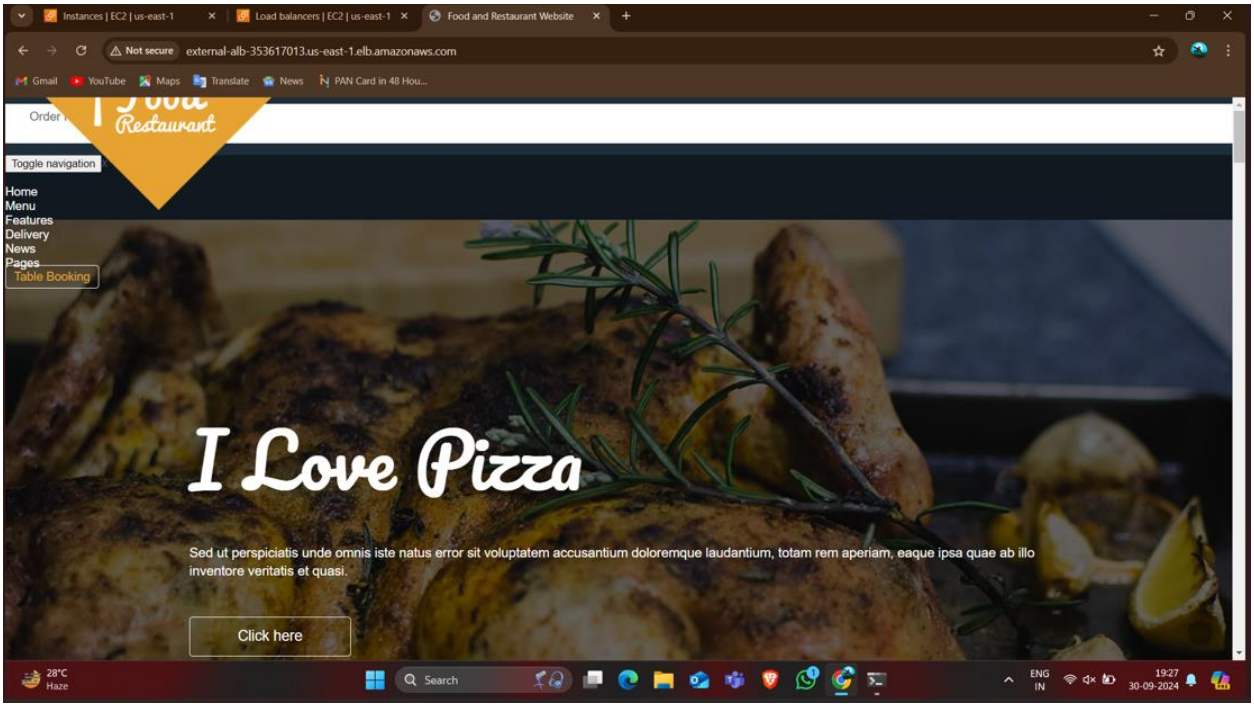
- Create the file for RDS “vi RDS.tf”.
- Write the terraform code for RDS

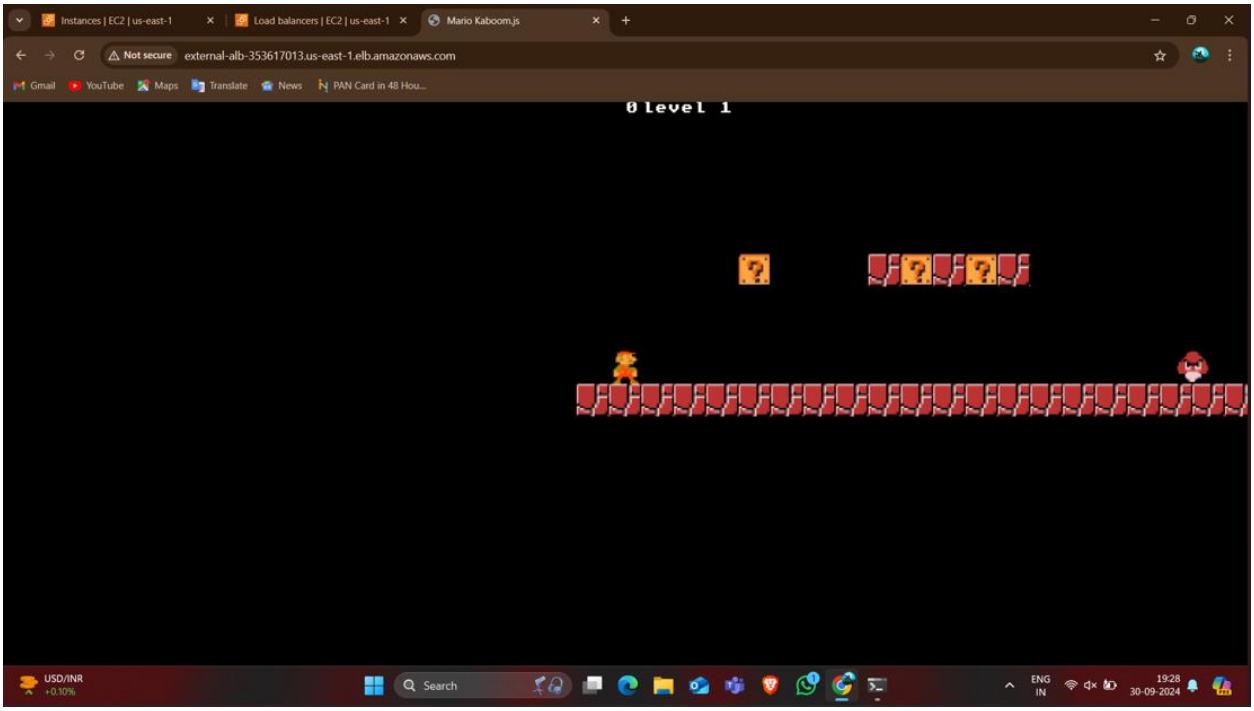




❖ OUTPUT:

- After creating the all resources and observed the all created resources in the aws consol.
- Than I have browse the public IP's of the instances than I got the output, below images.
- Than I copied the end url of loadbalancer and start browsing in the google.
- And increased the load to check the changes in webpage.
- When the load increased the webpages are changes.
- This is loadbalancer end url results.


Instances | EC2 | us-east-1
Load balancers | EC2 | us-east-1
Food and Restaurant Website
Not secure
external-alb-353617013.us-east-1.elb.amazonaws.com
Gmail
YouTube
Maps
Translate
News
PAN Card in 48 Hou...
Restaurant
Toggle navigation
Home
Menu
Features
Delivery
News
Pages
Table Booking
I Love Pizza
Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi.
Click here
28°C
Haze
Search
ENG
IN
19:27
30-09-2024


Instances | EC2 | us-east-1
Load balancers | EC2 | us-east-1
Mario Kaboom.js
Not secure
external-alb-353617013.us-east-1.elb.amazonaws.com
Gmail
YouTube
Maps
Translate
News
PAN Card in 48 Hou...
0 level 1
USD/INR
+0.10%
Search
ENG
IN
19:28
30-09-2024


Thank
you!