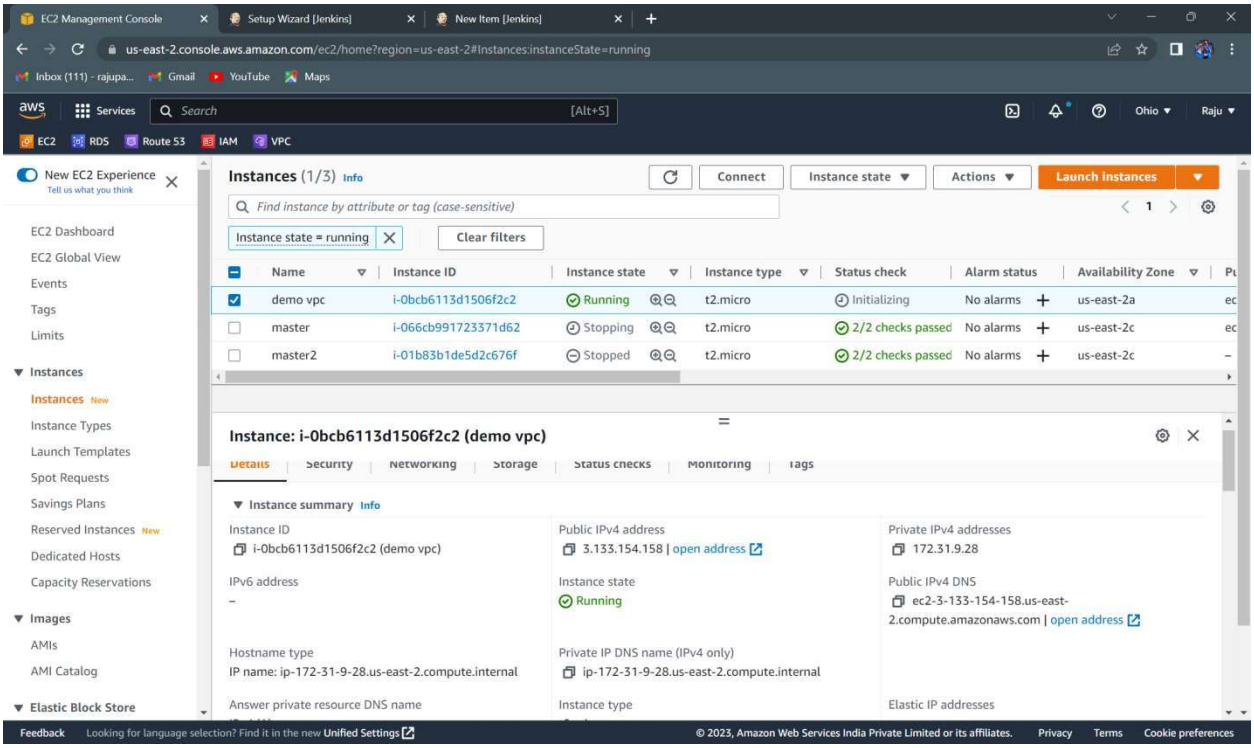
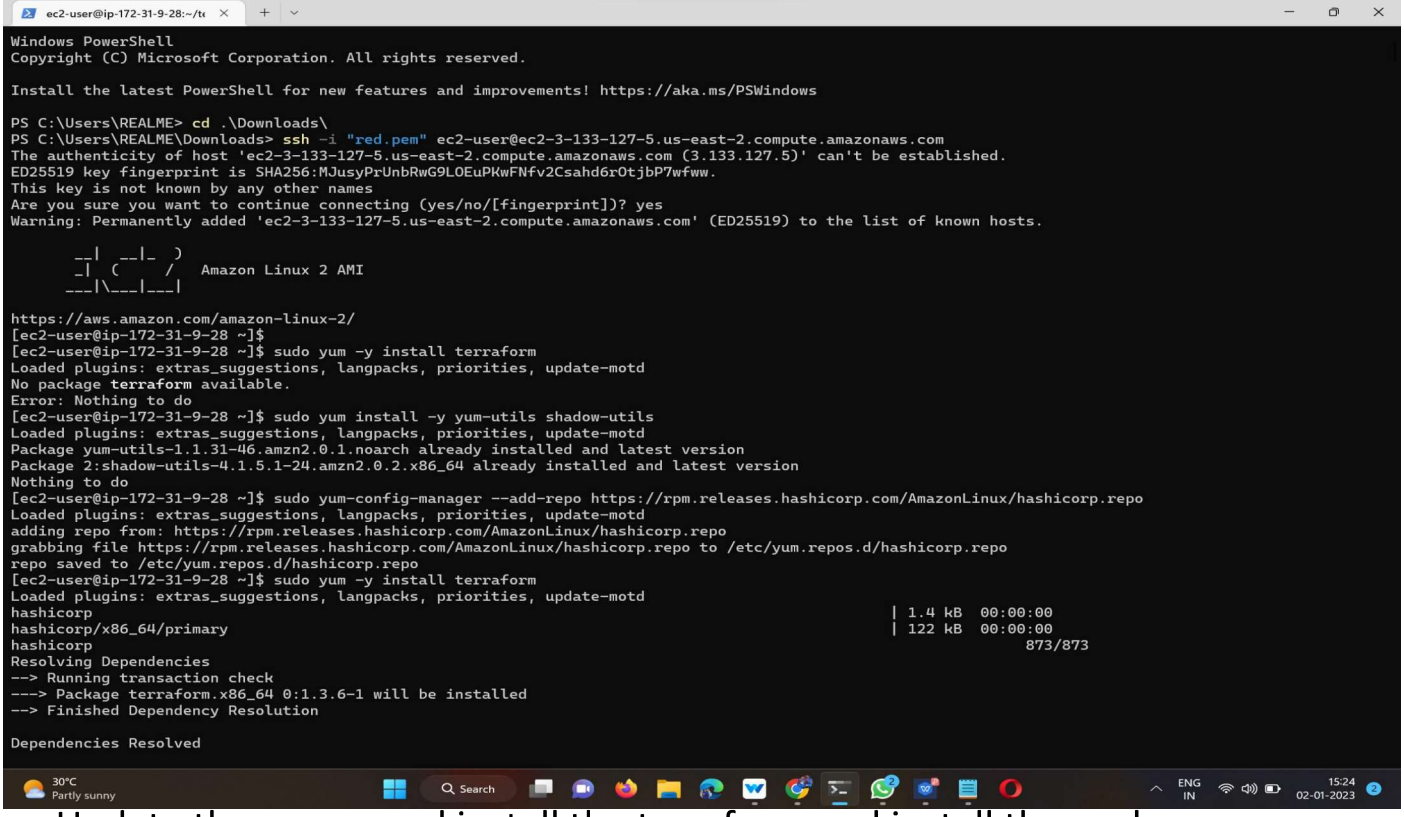
TERRAFORM

Terraform- Terraform is an open-source infrastructure as a code (IAC) tool that allows to create, manage & deploy the production-ready environment. Terraform codifies cloud APIs into declarative configuration files. Terraform can manage both existing service providers and custom in-house solutions.

Create a ec2 with amazon linux server and security group as ssh and keypair.



And the instance is connected to terminal.

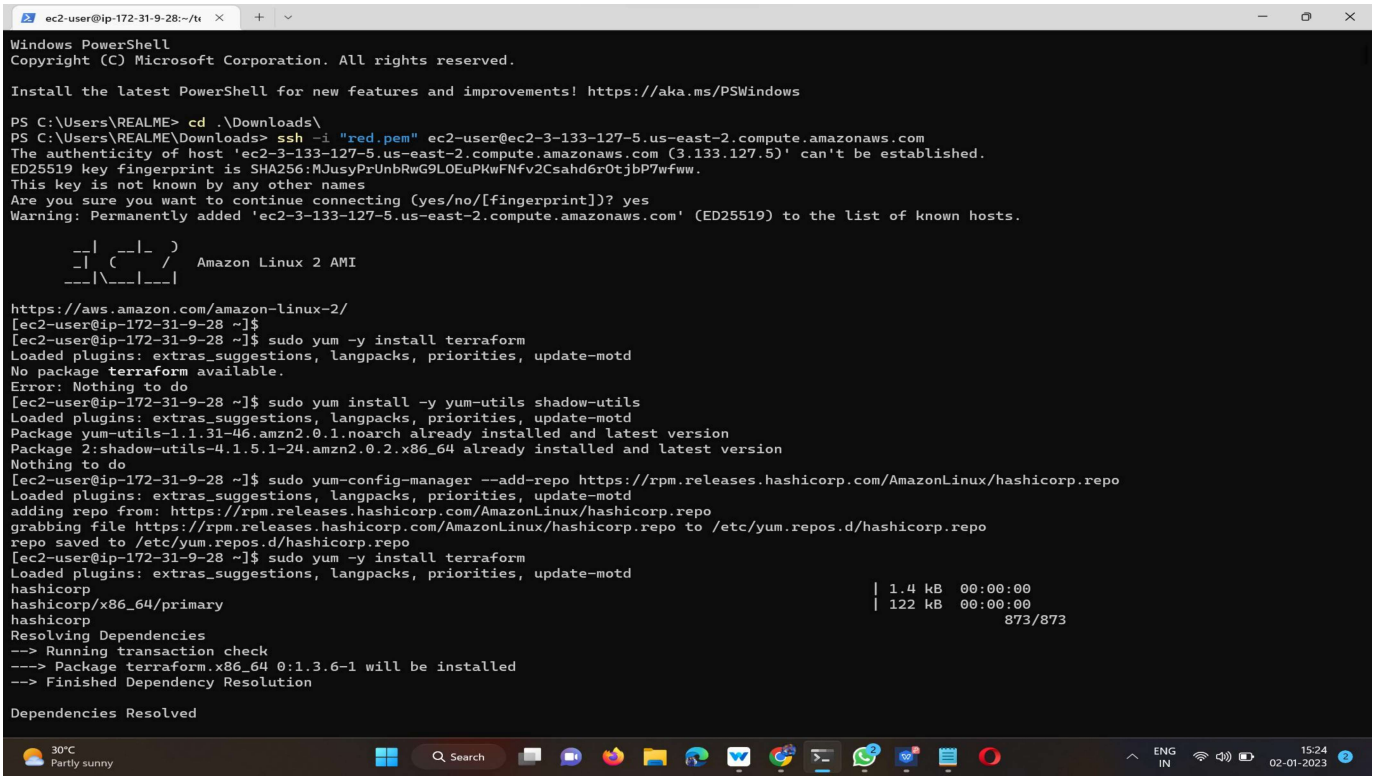


Update the server-and install the terraform and install the packages

sudo yum install -y yum-utils shadow-utils

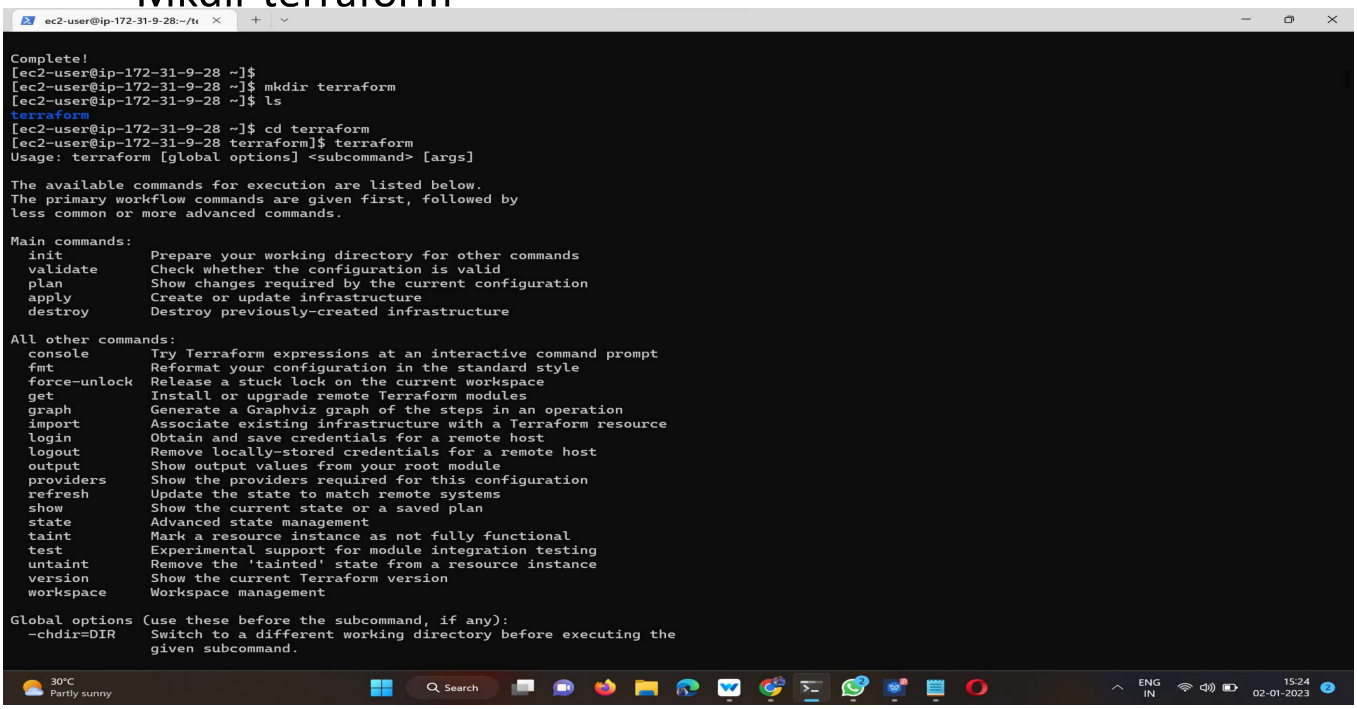
sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo

Sudo yum –y install terraform

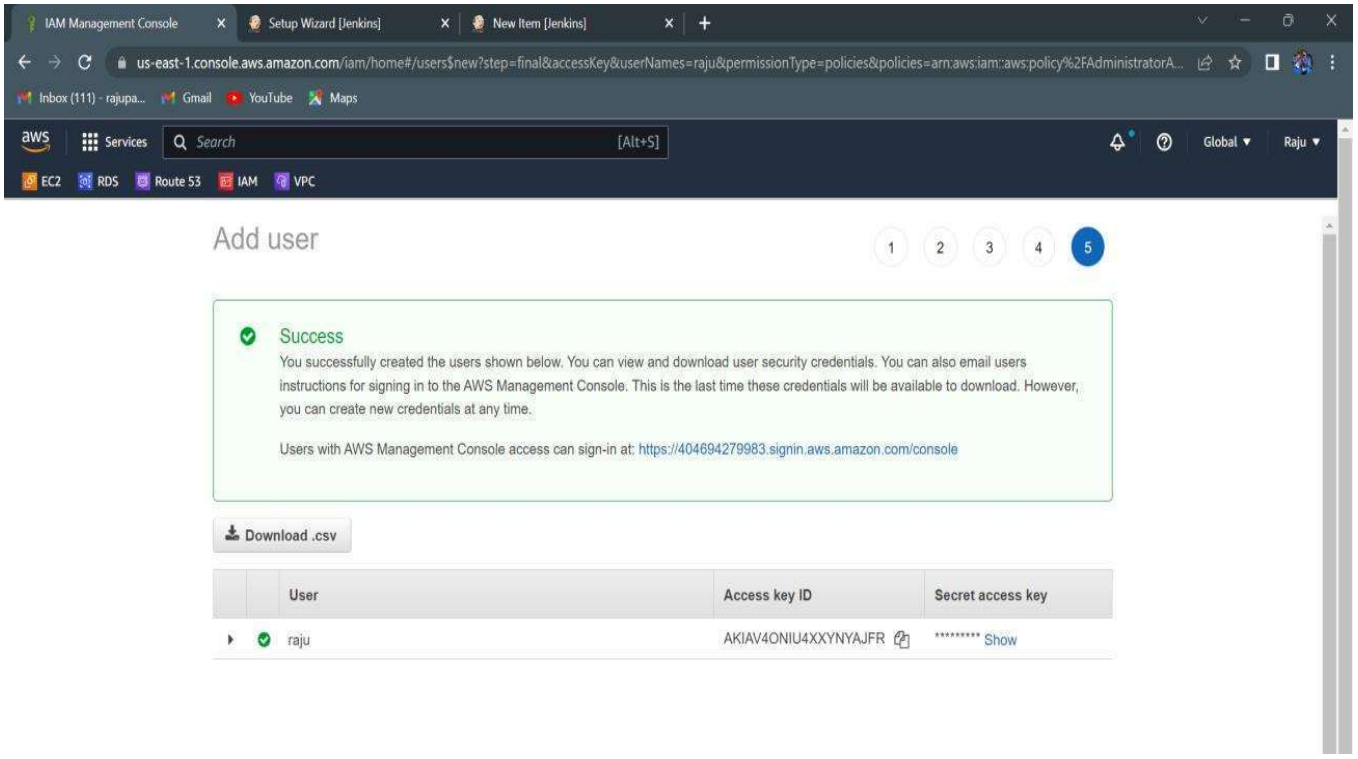


And make a directory by using the command

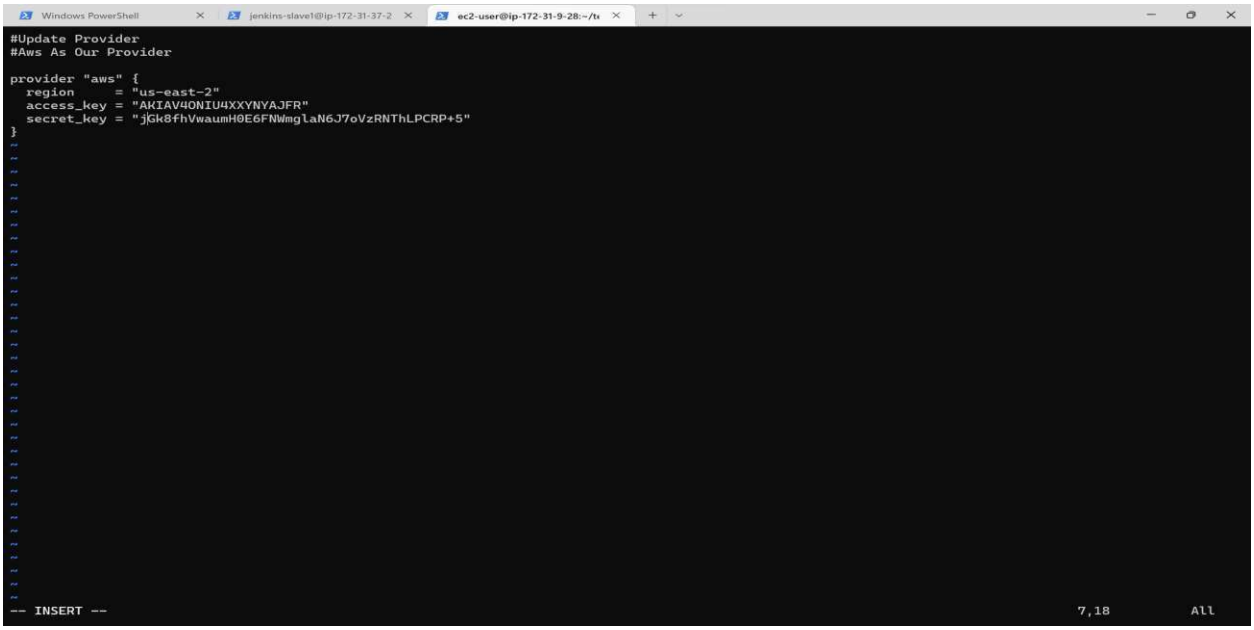
Mkdir terraform



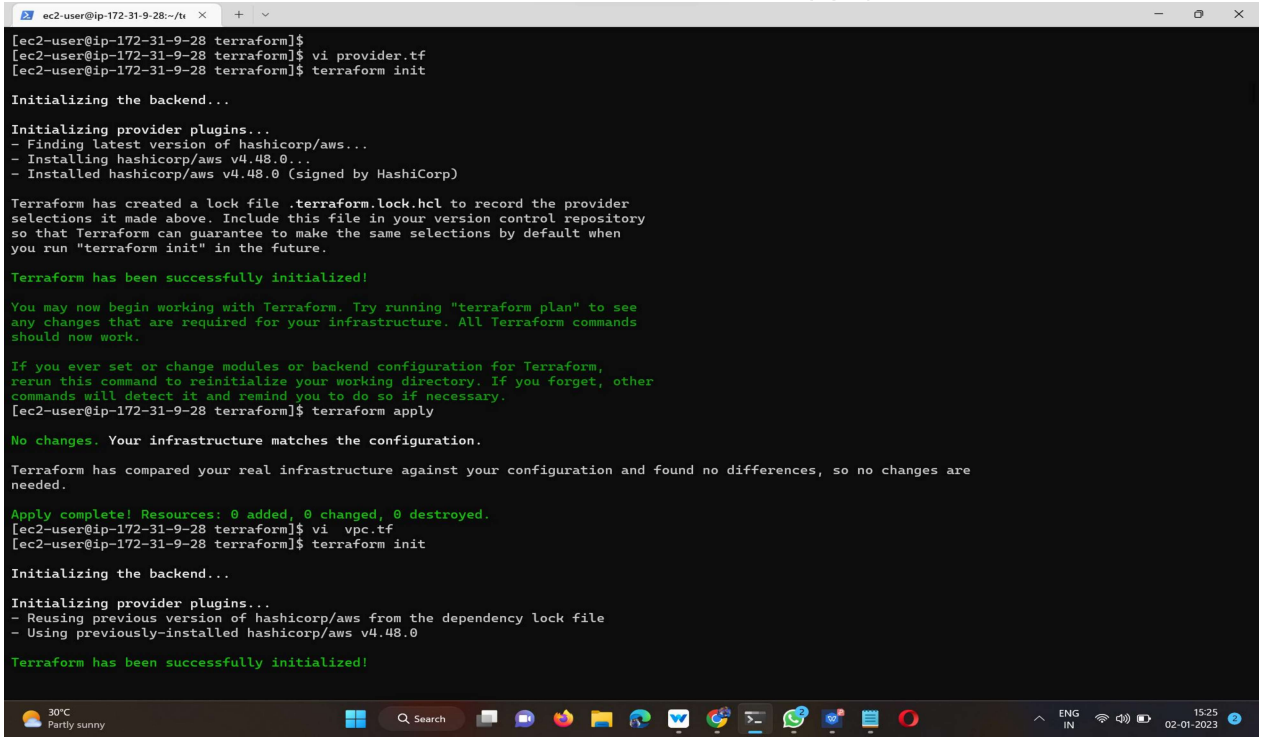
And create a iam user account and give the programmatic access and give the administration access



And in terminal create a provider.tf file and give the aws as a provider and give the access key and secret key created in iam

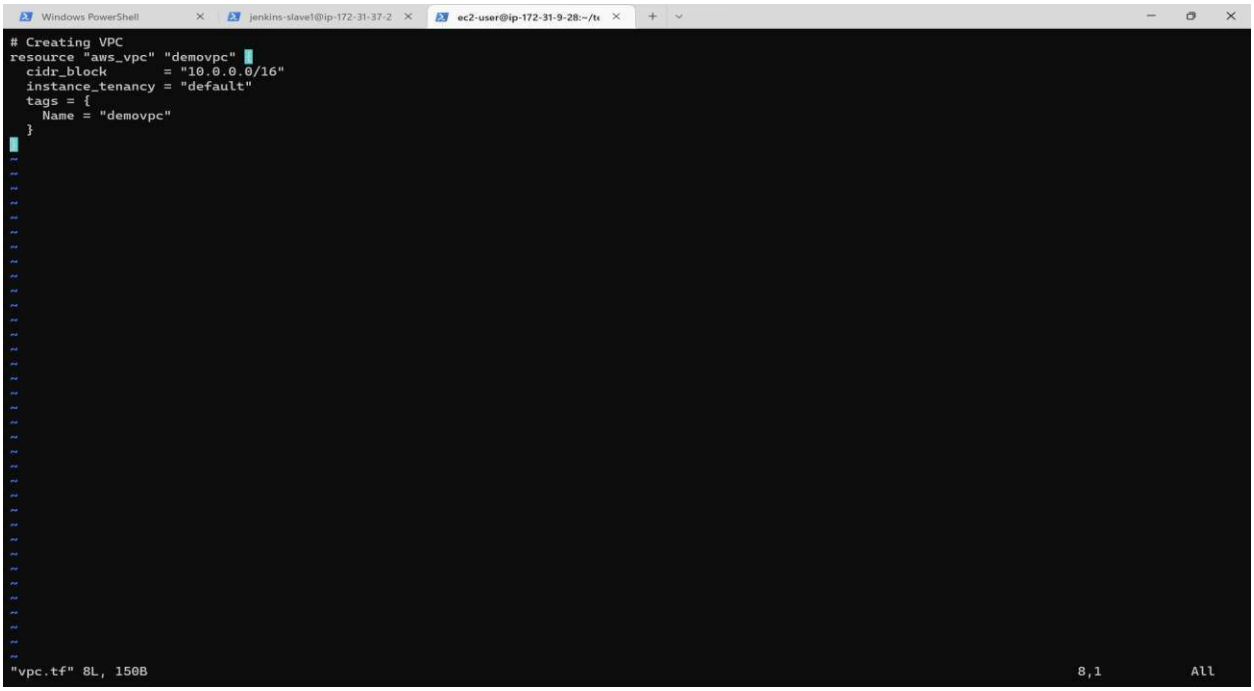


Then initialiaze the terraform and terraform apply



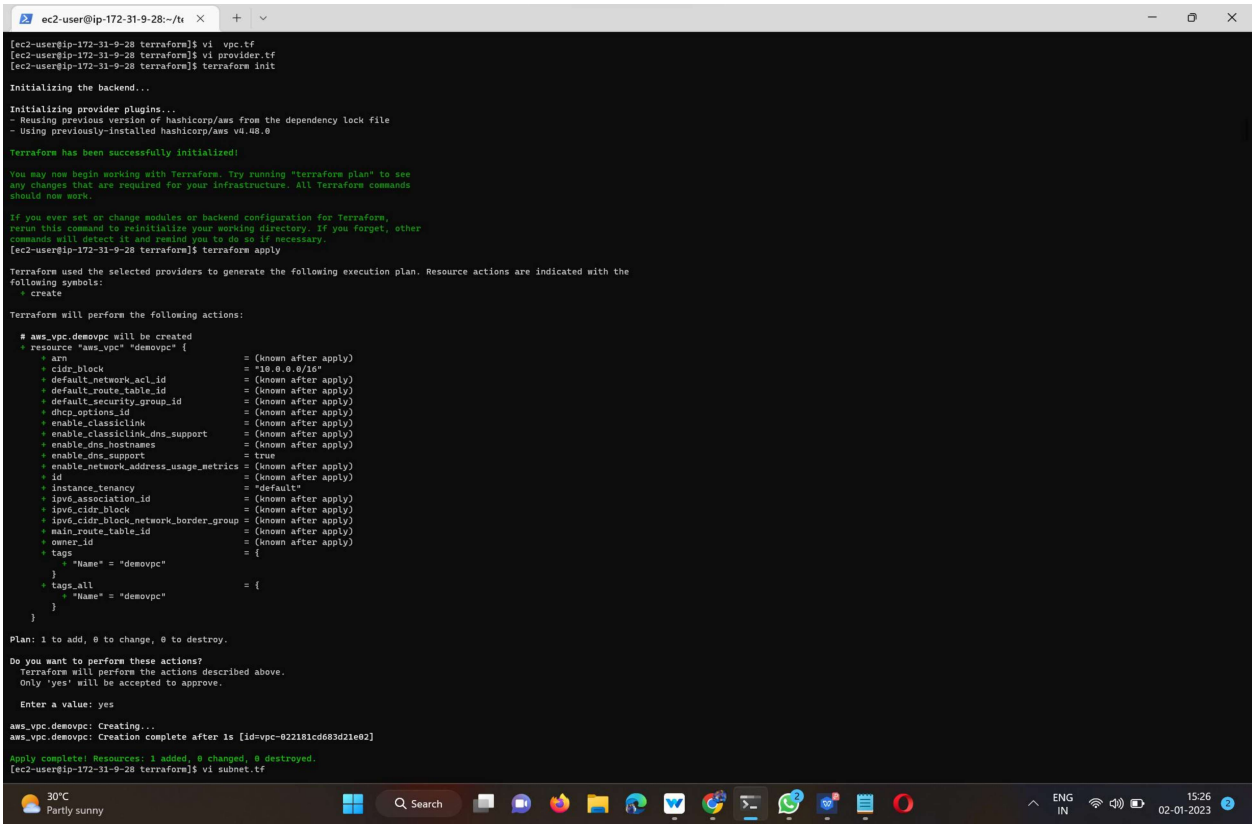
Create vpc.tf file by using the command

Vi vpc.tf

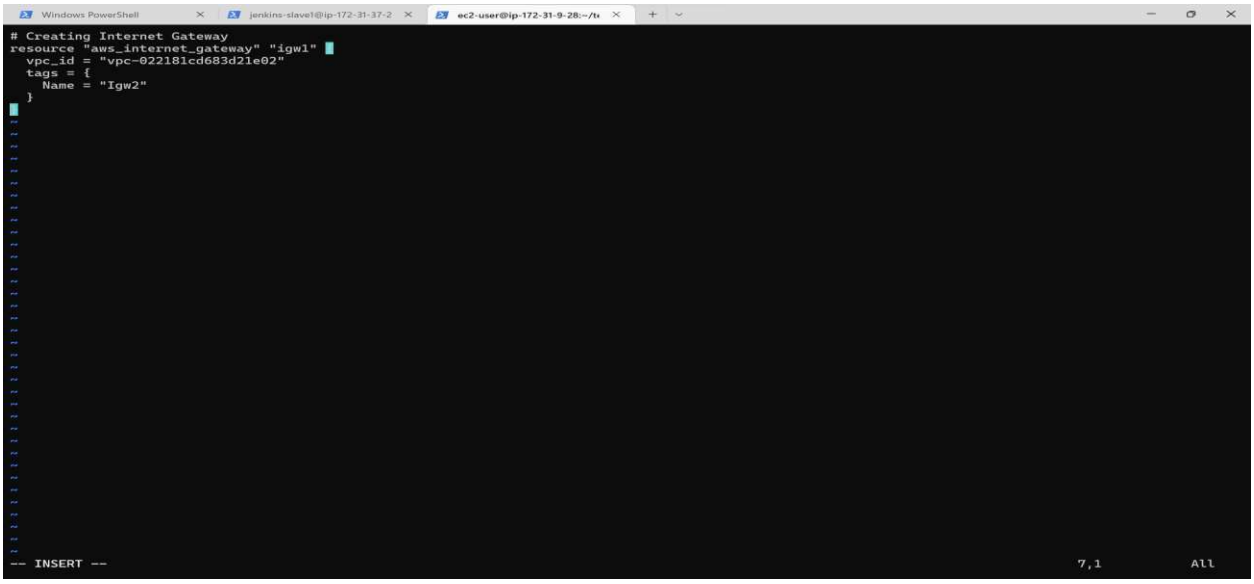


Save the script and init the terraform-terraform init

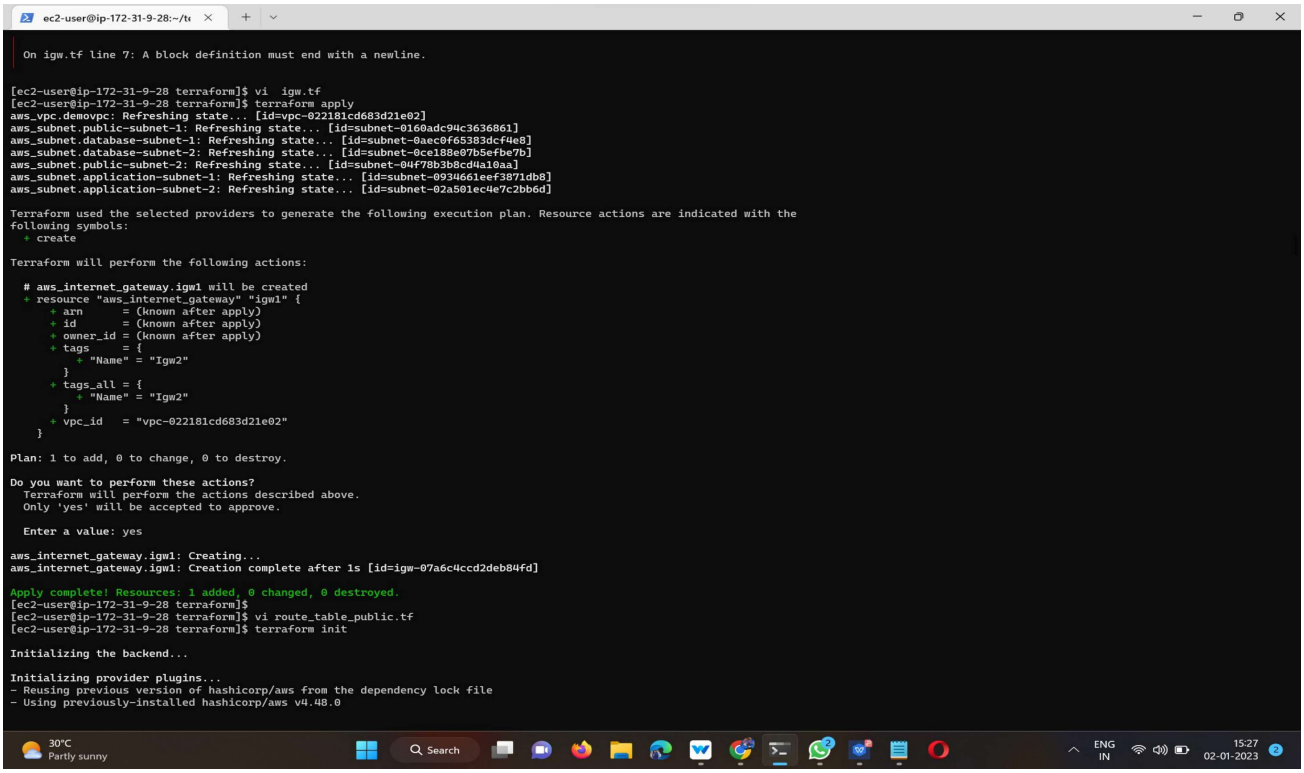
And terraform apply



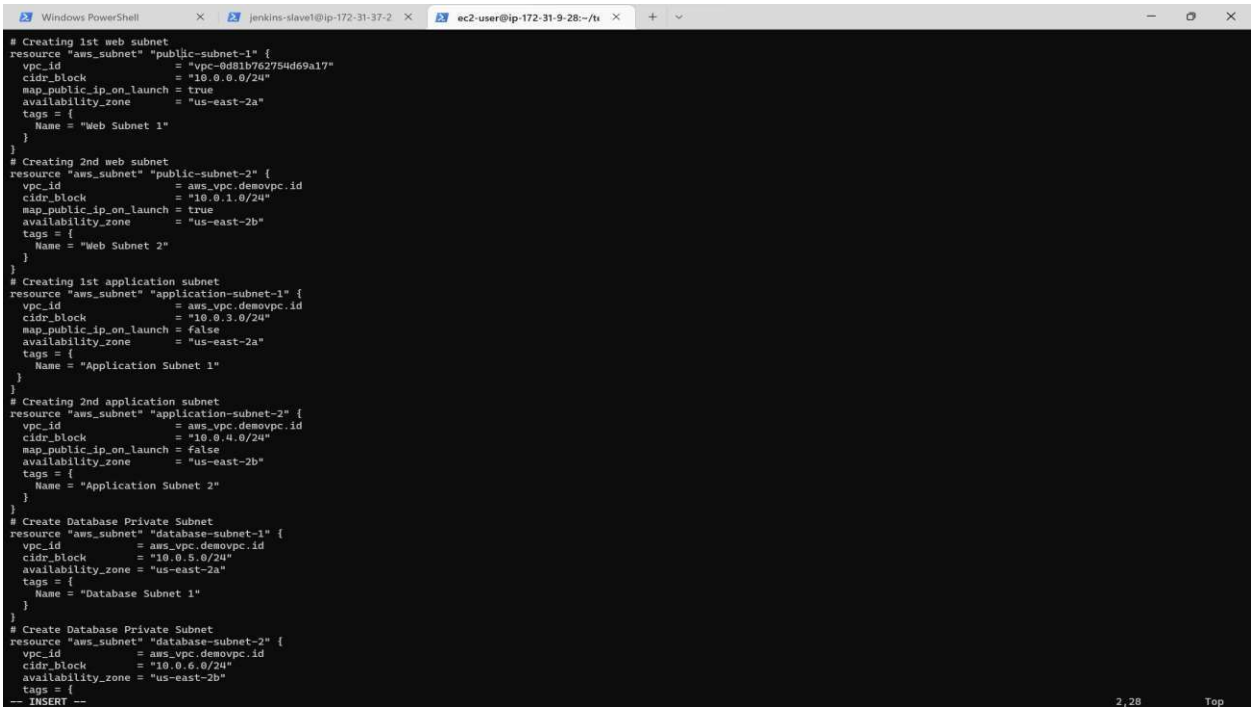
Create a internet gateway and the gateway is connected to the vpc and create file –vi igw.tf



And again initialise the terraform and terraform apply the internet gateway created it is connected to vpc

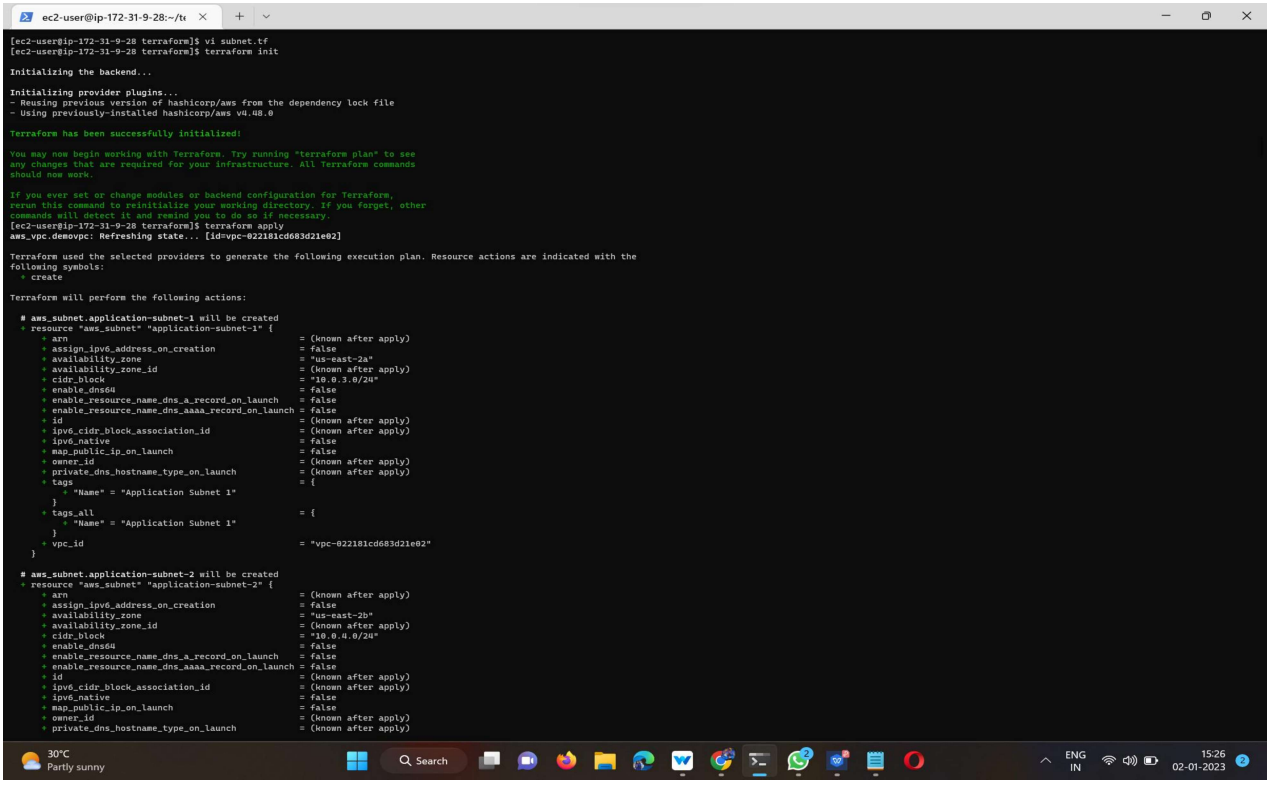


Create a subnet-vi subnet.tf

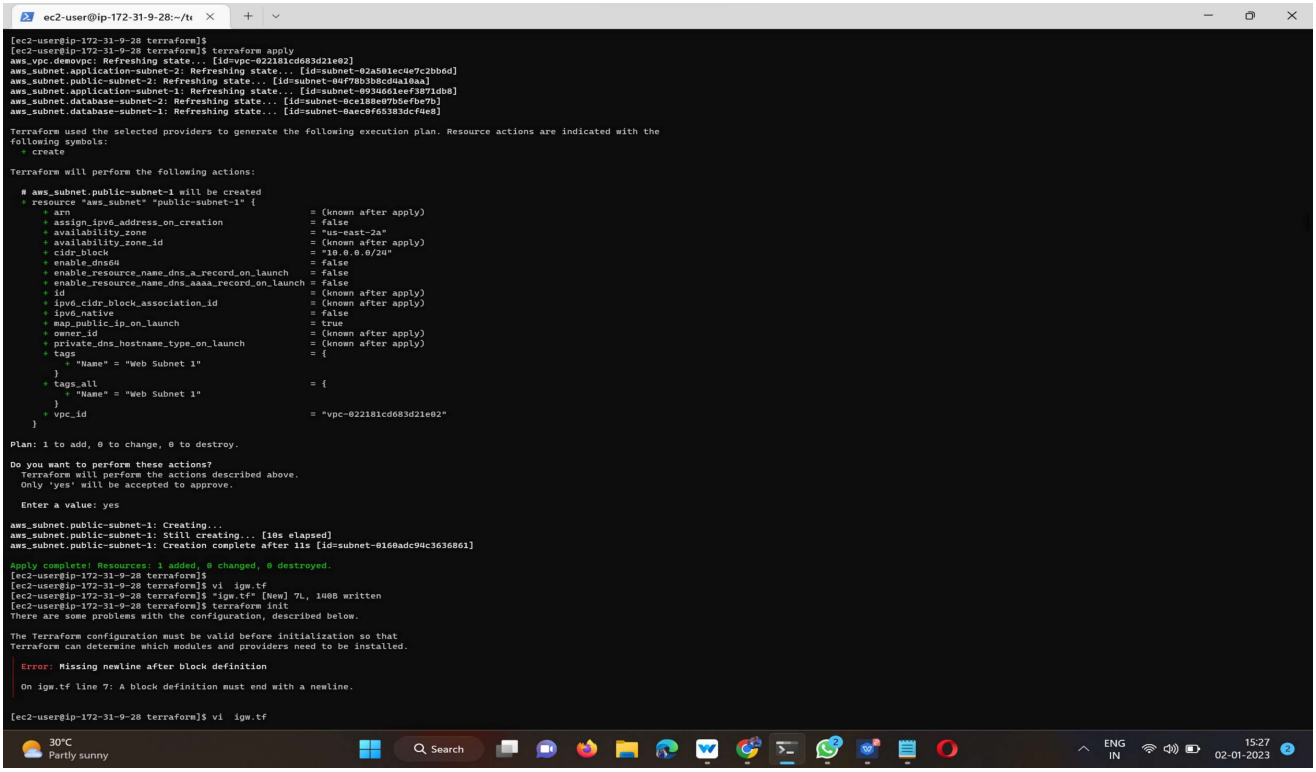


```
# Creating 1st web subnet
resource "aws_subnet" "public-subnet-1" {
  vpc_id                  = "vpc-0d81b762754d69a17"
  cidr_block              = "10.0.0.0/24"
  map_public_ip_on_launch = true
  availability_zone       = "us-east-2a"
  tags = {
    Name = "Web Subnet 1"
  }
}
# Creating 2nd web subnet
resource "aws_subnet" "public-subnet-2" {
  vpc_id                  = aws_vpc.demovpc.id
  cidr_block              = "10.0.1.0/24"
  map_public_ip_on_launch = true
  availability_zone       = "us-east-2b"
  tags = {
    Name = "Web Subnet 2"
  }
}
# Creating 1st application subnet
resource "aws_subnet" "application-subnet-1" {
  vpc_id                  = aws_vpc.demovpc.id
  cidr_block              = "10.0.3.0/24"
  map_public_ip_on_launch = false
  availability_zone       = "us-east-2a"
  tags = {
    Name = "Application Subnet 1"
  }
}
# Creating 2nd application subnet
resource "aws_subnet" "application-subnet-2" {
  vpc_id                  = aws_vpc.demovpc.id
  cidr_block              = "10.0.4.0/24"
  map_public_ip_on_launch = false
  availability_zone       = "us-east-2b"
  tags = {
    Name = "Application Subnet 2"
  }
}
# Create Database Private Subnet
resource "aws_subnet" "database-subnet-1" {
  vpc_id            = aws_vpc.demovpc.id
  cidr_block        = "10.0.5.0/24"
  availability_zone = "us-east-2a"
  tags = {
    Name = "Database Subnet 1"
  }
}
# Create Database Private Subnet
resource "aws_subnet" "database-subnet-2" {
  vpc_id            = aws_vpc.demovpc.id
  cidr_block        = "10.0.6.0/24"
  availability_zone = "us-east-2b"
  tags = {
-- INSERT --                                                    2,28          Top
```

Terraform init and terraform apply subnets can be created

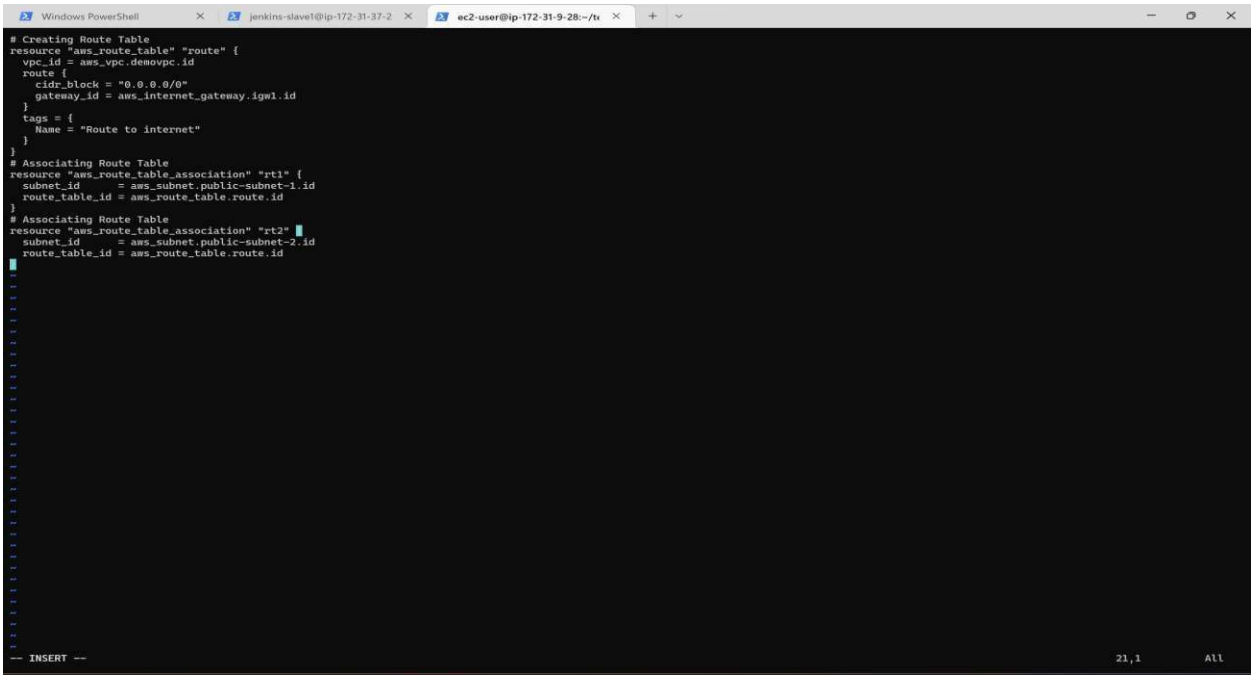


```
[ec2-user@ip-172-31-9-28 terraform]$ vi subnet.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

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-172-31-9-28 terraform]$ terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-022181cd683d21e02]

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_subnet.application-subnet-1 will be created
  + resource "aws_subnet" "application-subnet-1" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "us-east-2a"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "10.0.3.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
      + id                                             = (known after apply)
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
      + map_public_ip_on_launch                        = false
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
      + tags                                           = {
          + "Name" = "Application Subnet 1"
        }
      + tags_all                                       = {
          + "Name" = "Application Subnet 1"
        }
      + vpc_id                                         = "vpc-022181cd683d21e02"
    }

  # aws_subnet.application-subnet-2 will be created
  + resource "aws_subnet" "application-subnet-2" {
      + arn                                            = (known after apply)
      + assign_ipv6_address_on_creation                = false
      + availability_zone                              = "us-east-2b"
      + availability_zone_id                           = (known after apply)
      + cidr_block                                     = "10.0.4.0/24"
      + enable_dns64                                   = false
      + enable_resource_name_dns_a_record_on_launch    = false
      + enable_resource_name_dns_aaaa_record_on_launch = false
      + id                                             = (known after apply)
      + ipv6_cidr_block_association_id                 = (known after apply)
      + ipv6_native                                    = false
      + map_public_ip_on_launch                        = false
      + owner_id                                       = (known after apply)
      + private_dns_hostname_type_on_launch            = (known after apply)
```



Create a route table with the name

Vi route_public_table.tf



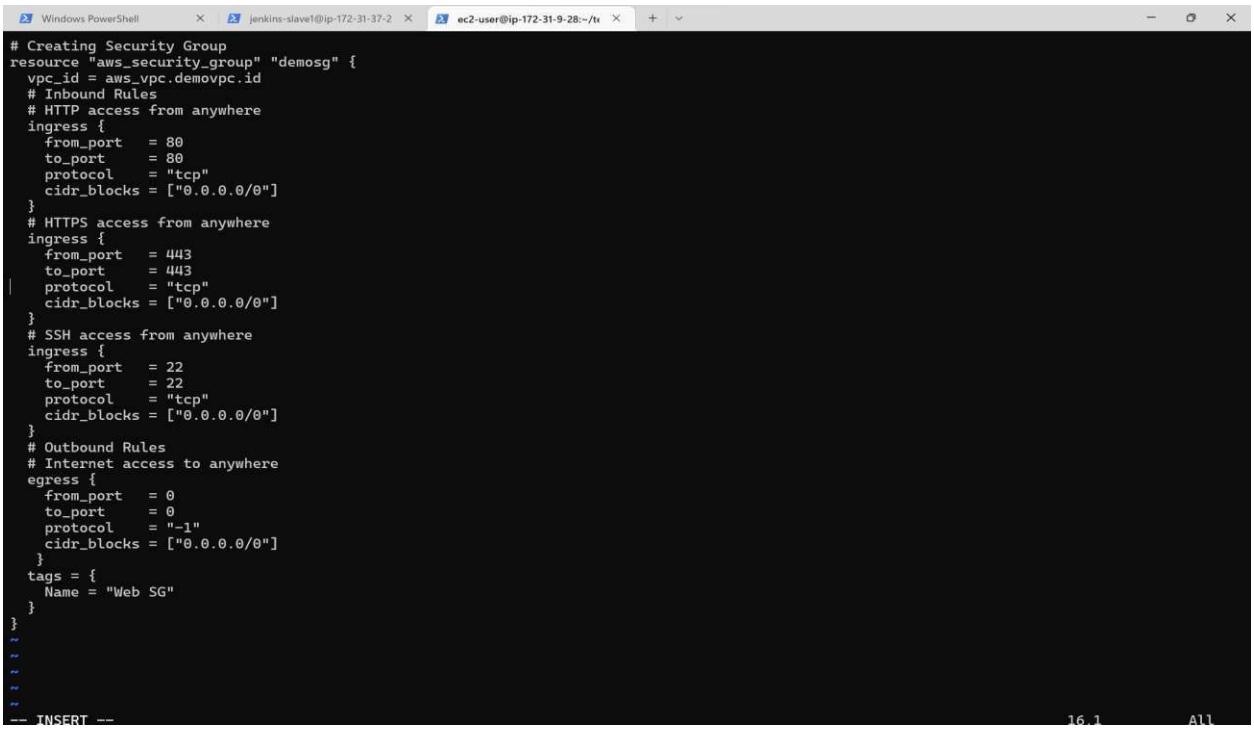
Initilaze the terraform and apply whether the script is write or wrong

And it creates the route tables and associate with subnets and the route is connecte to internet gateway

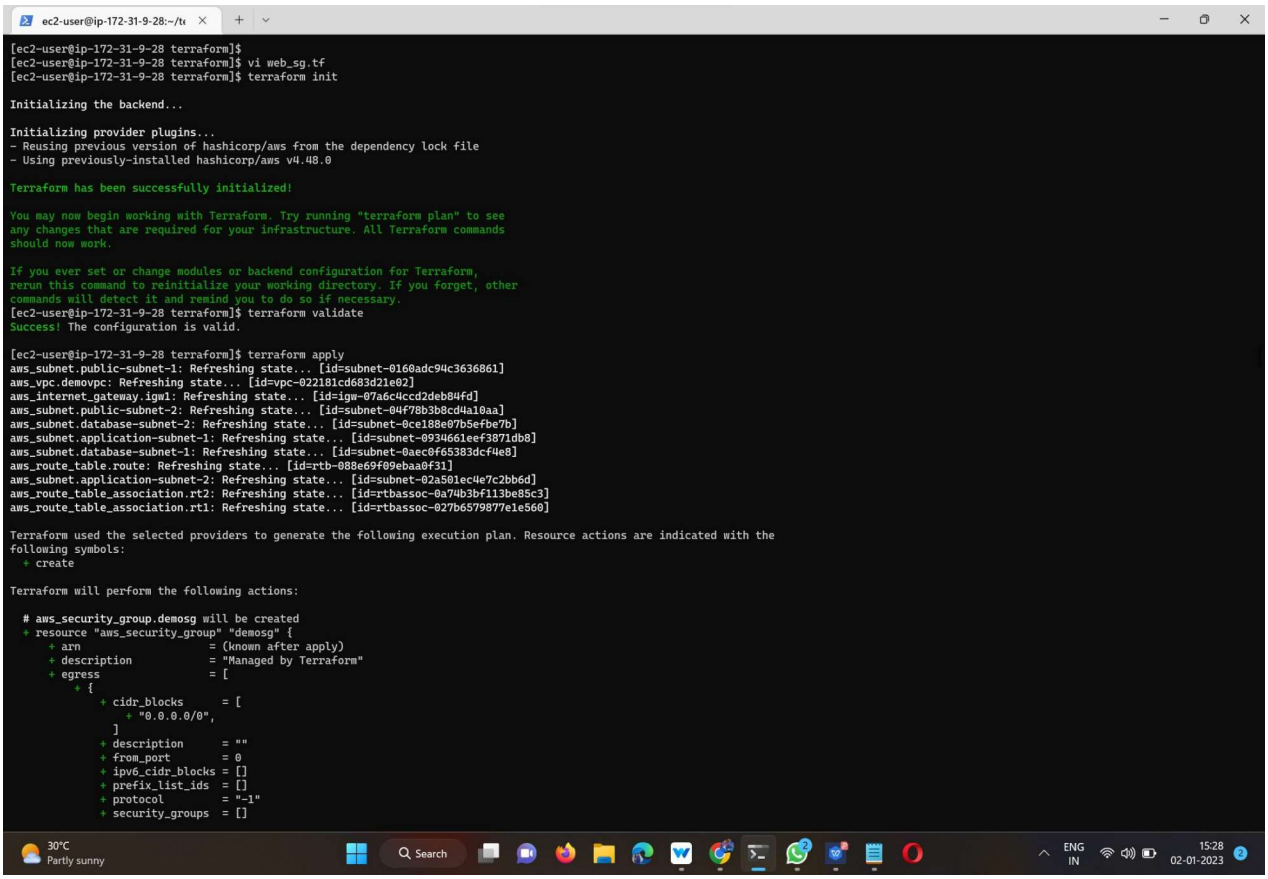
```
[ec2-user@ip-172-31-9-28 terraform]$
[ec2-user@ip-172-31-9-28 terraform]$ vi route_table_public.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

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-172-31-9-28 terraform]$ terraform apply
aws_internet_gateway.igw1: Refreshing state... [id=igw-07a6c4ccd2deb84fd]
aws_vpc.demovpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0934661eef3871db8]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04f78b3b8cd4a10aa]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0aec0f65383dcf4e8]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]

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_route_table.route will be created
  + resource "aws_route_table" "route" {
      + arn              = (known after apply)
      + id               = (known after apply)
      + owner_id         = (known after apply)
      + propagating_vgws = (known after apply)
      + route            = [
          + {
              + carrier_gateway_id         = ""
              + cidr_block                 = "0.0.0.0/0"
              + core_network_arn           = ""
              + destination_prefix_list_id = ""
              + egress_only_gateway_id     = ""
              + gateway_id                 = "igw-07a6c4ccd2deb84fd"
              + instance_id                = ""
              + ipv6_cidr_block            = ""
              + local_gateway_id           = ""
              + nat_gateway_id             = ""
              + network_interface_id       = ""
              + transit_gateway_id         = ""
              + vpc_endpoint_id            = ""
```

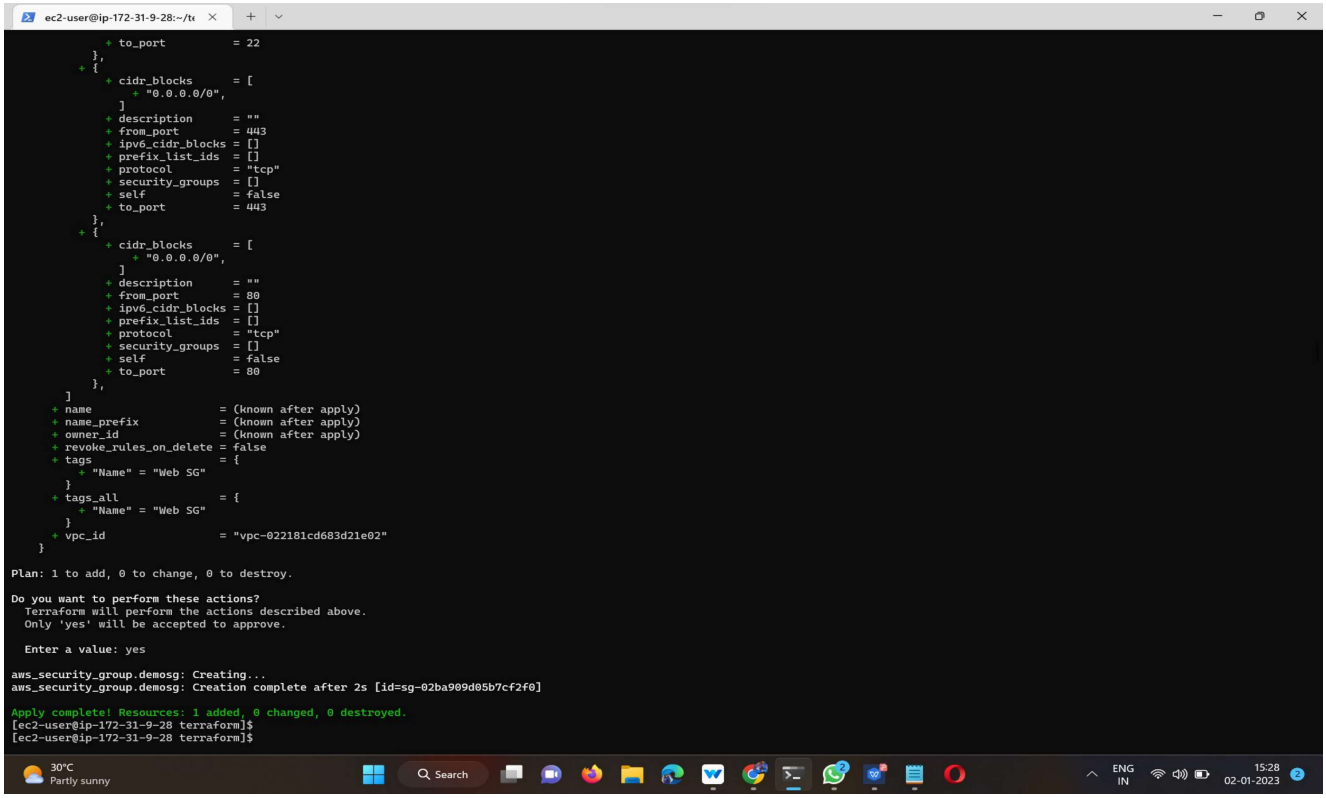

```
              + cidr_block                 = "0.0.0.0/0"
              + core_network_arn           = ""
              + destination_prefix_list_id = ""
              + egress_only_gateway_id     = ""
              + gateway_id                 = "igw-07a6c4ccd2deb84fd"
              + instance_id                = ""
              + ipv6_cidr_block            = ""
              + local_gateway_id           = ""
              + nat_gateway_id             = ""
              + network_interface_id       = ""
              + transit_gateway_id         = ""
              + vpc_endpoint_id            = ""
              + vpc_peering_connection_id  = ""
            },
        ]
      + tags             = {
          + "Name" = "Route to internet"
        }
      + tags_all         = {
          + "Name" = "Route to internet"
        }
      + vpc_id           = "vpc-022181cd683d21e02"
    }

  # aws_route_table_association.rt1 will be created
  + resource "aws_route_table_association" "rt1" {
      + id             = (known after apply)
      + route_table_id = (known after apply)
      + subnet_id      = "subnet-0160adc94c3636861"
    }

  # aws_route_table_association.rt2 will be created
  + resource "aws_route_table_association" "rt2" {
      + id             = (known after apply)
      + route_table_id = (known after apply)
      + subnet_id      = "subnet-04f78b3b8cd4a10aa"
    }

Plan: 3 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_route_table.route: Creating...
aws_route_table.route: Creation complete after 0s [id=rtb-088e69f09ebaa0f31]
aws_route_table_association.rt1: Creating...
aws_route_table_association.rt2: Creating...
aws_route_table_association.rt2: Creation complete after 0s [id=rtbassoc-0a74b3bf113be85c3]
aws_route_table_association.rt1: Creation complete after 0s [id=rtbassoc-027b6579877e1e560]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-9-28 terraform]$
[ec2-user@ip-172-31-9-28 terraform]$
```

Create a security groups –vi web_sg.tf

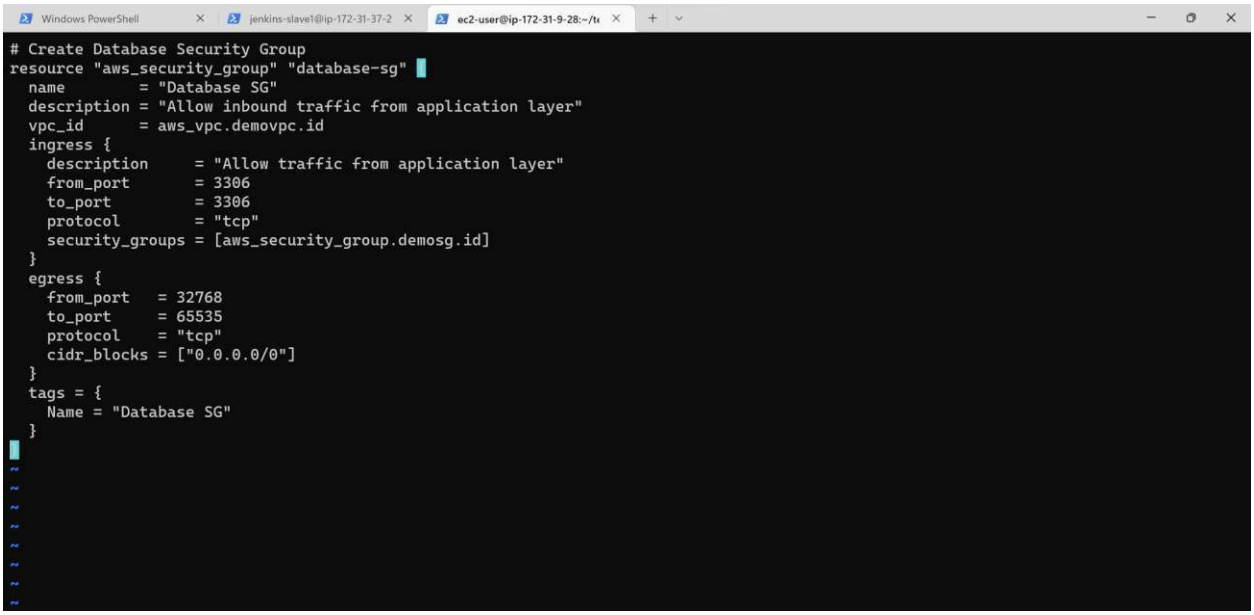


Terraform init and apply the terraform



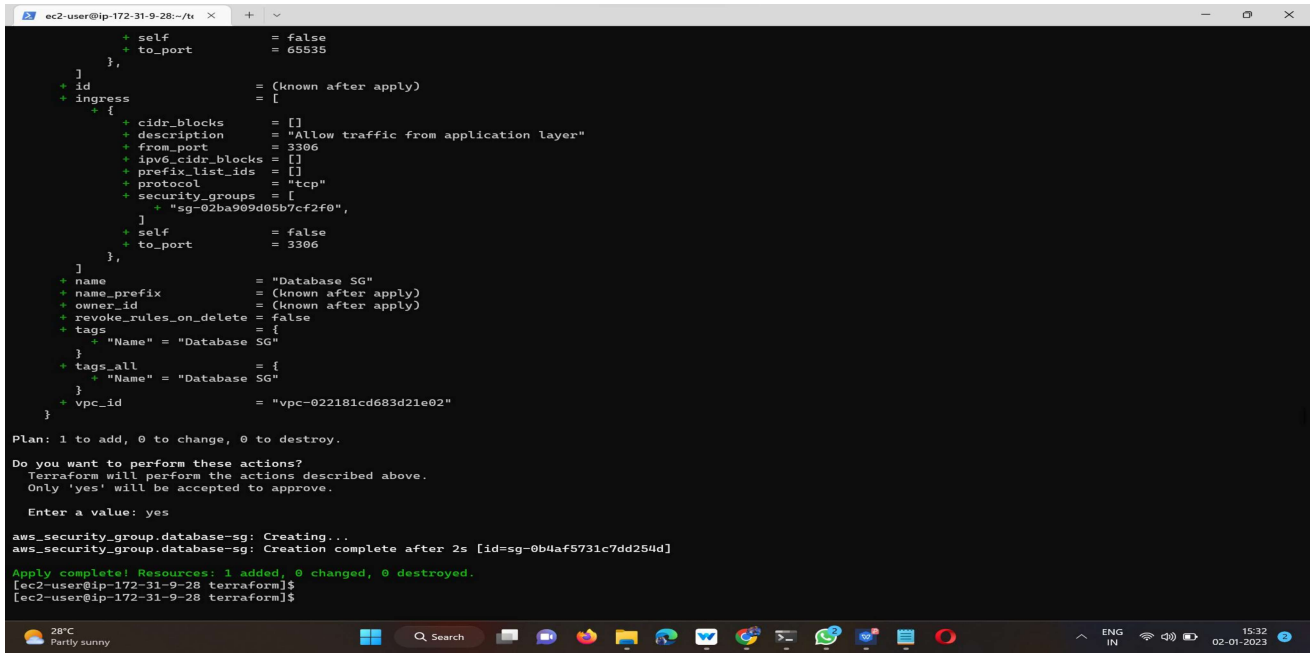


And again create a security group vi database_sg.tf



Terraform init and apply

```
[ec2-user@ip-172-31-9-28 terraform]$
[ec2-user@ip-172-31-9-28 terraform]$ vi  database_sg.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

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-172-31-9-28 terraform]$ terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_internet_gateway.igw1: Refreshing state... [id=igw-07a6c4ccd2deb84fd]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0aec0f65383dcf4e8]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04f78b3b8cd4a10aa]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0934661eef3871db8]
aws_security_group.demosg: Refreshing state... [id=sg-02ba909d05b7cf2f0]
aws_route_table.route: Refreshing state... [id=rtb-088e69f09ebaa0f31]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_instance.demoinstance: Refreshing state... [id=i-04f152afcd678f0ad]
aws_route_table_association.rt1: Refreshing state... [id=rtbassoc-027b6579877e1e560]
aws_instance.demoinstance1: Refreshing state... [id=i-0fecb1145449837b9]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-0a74b3bf113be85c3]

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_security_group.database-sg will be created
  + resource "aws_security_group" "database-sg" {
      + arn                    = (known after apply)
      + description            = "Allow inbound traffic from application layer"
      + egress                 = [
          + {
```

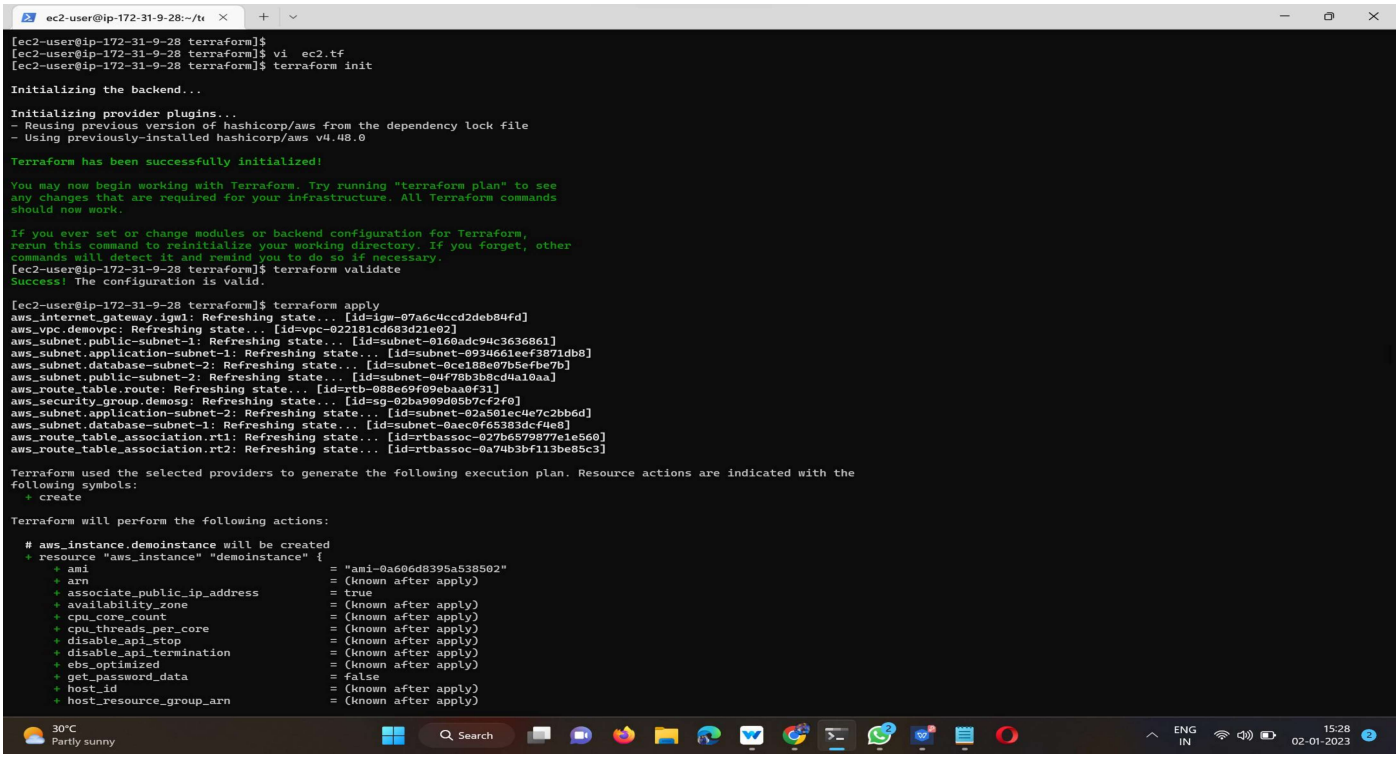


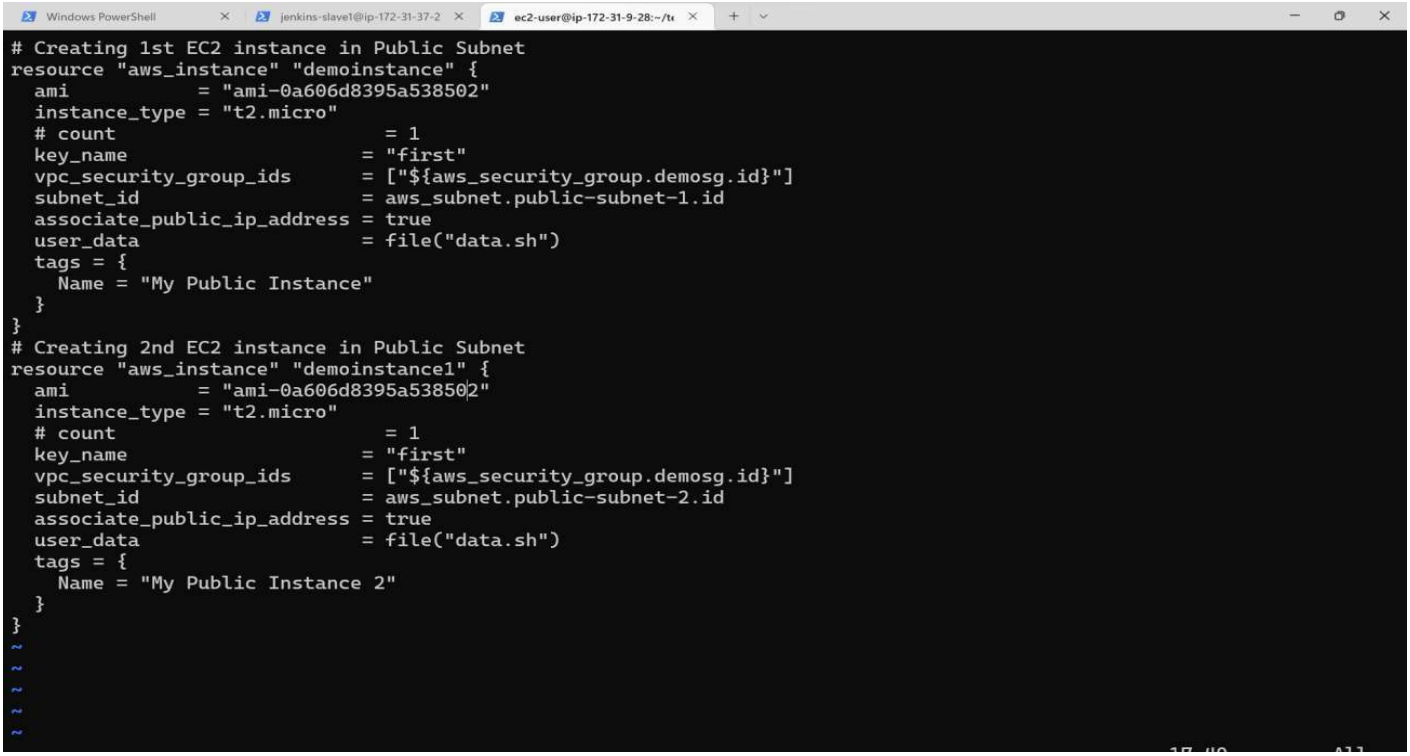
Create data.sh file-vi data.sh

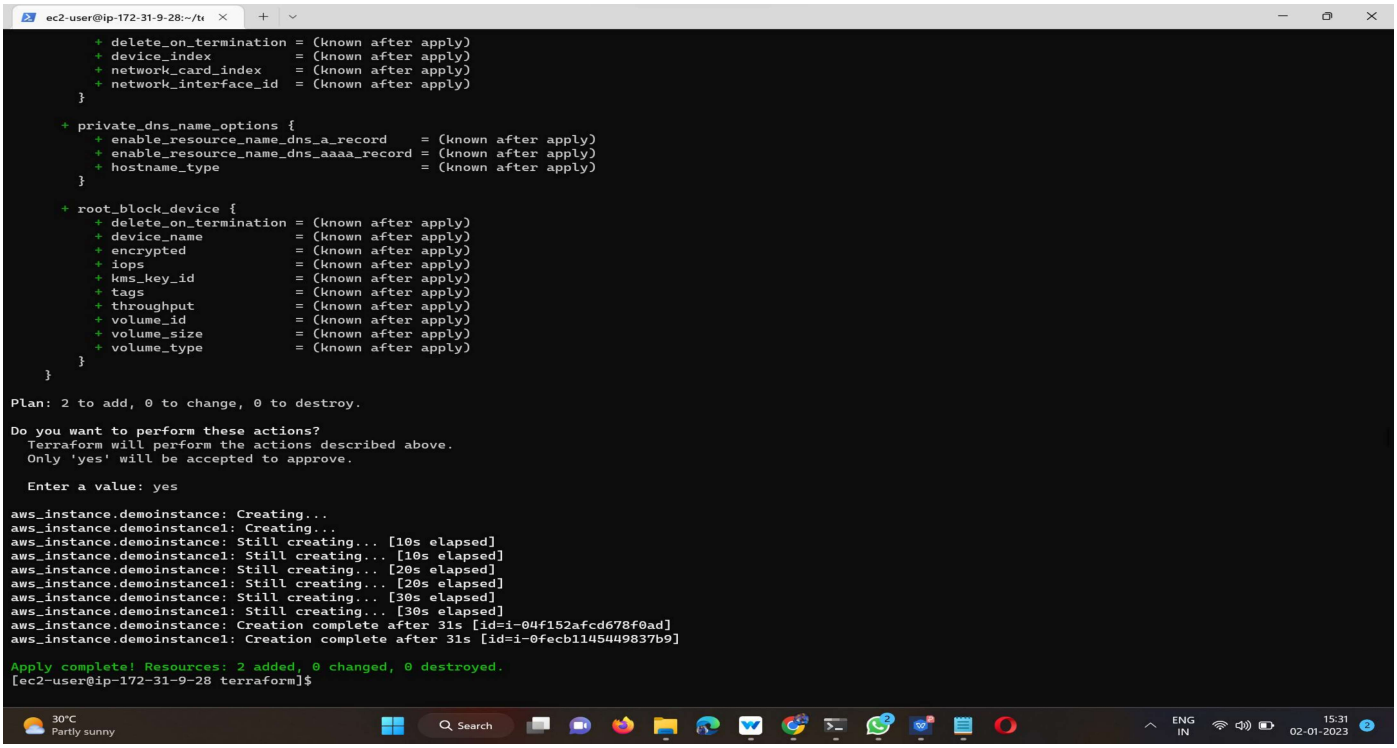
```
#!/bin/bash
yum update -y
yum install -y httpd.x86_64
systemctl start httpd.service
systemctl enable httpd.service
echo "Hello World from $(hostname -f)" > /var/www/html/index.html
```

```
[ec2-user@ip-172-31-9-28 terraform]$
[ec2-user@ip-172-31-9-28 terraform]$ vi data.sh
[ec2-user@ip-172-31-9-28 terraform]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

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-172-31-9-28 terraform]$ terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-31-9-28 terraform]$ terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_internet_gateway.igw1: Refreshing state... [id=igw-07a8c4ccd2deb84fd]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0aec0f65383dcf4e8]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-093466leef3871db8]
aws_route_table.route: Refreshing state... [id=rtb-088e69f09ebaa0f31]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04f78b3b8cd4a10aa]
aws_route_table_association.rt1: Refreshing state... [id=rtbassoc-027b65798f7e1e560]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-0a74b3bf113be85c3]

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.
[ec2-user@ip-172-31-9-28 terraform]$
[ec2-user@ip-172-31-9-28 terraform]$ vi web_sg.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

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.
```

And then create the ec2 instance and the give the user data in this instance-vi ec2.tf


ec2-user@ip-172-31-9-28:~/te
[ec2-user@ip-172-31-9-28 terraform]$
[ec2-user@ip-172-31-9-28 terraform]$ vi ec2.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0
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-172-31-9-28 terraform]$ terraform validate
Success! The configuration is valid.
[ec2-user@ip-172-31-9-28 terraform]$ terraform apply
aws_internet_gateway.igw1: Refreshing state... [id=igw-07a6c4ccd2deb84fd]
aws_vpc.demovpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0934661eef3871db8]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04f78b3b8cd4a10aa]
aws_route_table.route: Refreshing state... [id=rtb-088e69f09ebaa0f31]
aws_security_group.demosg: Refreshing state... [id=sg-02ba909d05b7cf2f0]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0aec0f65383dcf4e8]
aws_route_table_association.rt1: Refreshing state... [id=rtbassoc-027b6579877e1e560]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-0a74b3bf113be85c3]
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_instance.demoinstance will be created
+ resource "aws_instance" "demoinstance" {
+ ami = "ami-0a606d8395a538502"
+ arn = (known after apply)
+ associate_public_ip_address = true
+ availability_zone = (known after apply)
+ cpu_core_count = (known after apply)
+ cpu_threads_per_core = (known after apply)
+ disable_api_stop = (known after apply)
+ disable_api_termination = (known after apply)
+ ebs_optimized = (known after apply)
+ get_password_data = false
+ host_id = (known after apply)
+ host_resource_group_arn = (known after apply)
30°C
Partly sunny
Search
ENG
IN
15:28
02-01-2023


Windows PowerShell
jenkins-slave1@ip-172-31-37-2
ec2-user@ip-172-31-9-28:~/te
# Creating 1st EC2 instance in Public Subnet
resource "aws_instance" "demoinstance" {
ami = "ami-0a606d8395a538502"
instance_type = "t2.micro"
# count = 1
key_name = "first"
vpc_security_group_ids = ["${aws_security_group.demosg.id}"]
subnet_id = aws_subnet.public-subnet-1.id
associate_public_ip_address = true
user_data = file("data.sh")
tags = {
Name = "My Public Instance"
}
}
# Creating 2nd EC2 instance in Public Subnet
resource "aws_instance" "demoinstance1" {
ami = "ami-0a606d8395a538502"
instance_type = "t2.micro"
# count = 1
key_name = "first"
vpc_security_group_ids = ["${aws_security_group.demosg.id}"]
subnet_id = aws_subnet.public-subnet-2.id
associate_public_ip_address = true
user_data = file("data.sh")
tags = {
Name = "My Public Instance 2"
}
}



Go to the ec2 instances and copy the public ip adress and paste it on browser.

The above load balancer is of type external Load balancer type is set to application The aws_lb_target_group_attachment resource will attach our instances to the Target Group. The load balancer will listen requests on port 80

Vi alb.tf

```
[ec2-user@ip-172-31-9-28 terraform]$
[ec2-user@ip-172-31-9-28 terraform]$ vi alb.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

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-172-31-9-28 terraform]$ terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_internet_gateway.igw1: Refreshing state... [id=igw-07a6c4ccd2deb84fd]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04f78b3b8cd4a10aa]
aws_route_table.route: Refreshing state... [id=rtb-088e69f09ebaa0f31]
aws_security_group.demosg: Refreshing state... [id=sg-02ba909d05b7cf2f0]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0934661eef3871db8]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0aec0f65383dcf4e8]
aws_route_table_association.rt1: Refreshing state... [id=rtbassoc-027b6579877e1e560]
aws_security_group.database-sg: Refreshing state... [id=sg-0b4af5731c7dd254d]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-0a74b3bf113be85c3]
aws_instance.demoinstance1: Refreshing state... [id=i-0fecb1145449837b9]
aws_instance.demoinstance: Refreshing state... [id=i-04f152afcd678f0ad]

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_lb.external-alb will be created
  + resource "aws_lb" "external-alb" {
      + arn                        = (known after apply)
      + arn_suffix                 = (known after apply)
      + desync_mitigation_mode     = "defensive"
```

```
# Creating External LoadBalancer
resource "aws_lb" "external-alb" {
  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" {
  target_group_arn = aws_lb_target_group.target-elb.arn
  target_id        = aws_instance.demoinstance.id
  port             = 80
  depends_on = [
    aws_instance.demoinstance,
  ]
}
resource "aws_lb_target_group_attachment" "attachment1" {
  target_group_arn = aws_lb_target_group.target-elb.arn
  target_id        = aws_instance.demoinstance1.id
  port             = 80
  depends_on = [
    aws_instance.demoinstance1,
  ]
}
resource "aws_lb_listener" "external-elb" {
  load_balancer_arn = aws_lb.external-alb.arn
  port              = "80"
  protocol          = "HTTP"
  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.target-elb.arn
  }
}
~
~
~
```

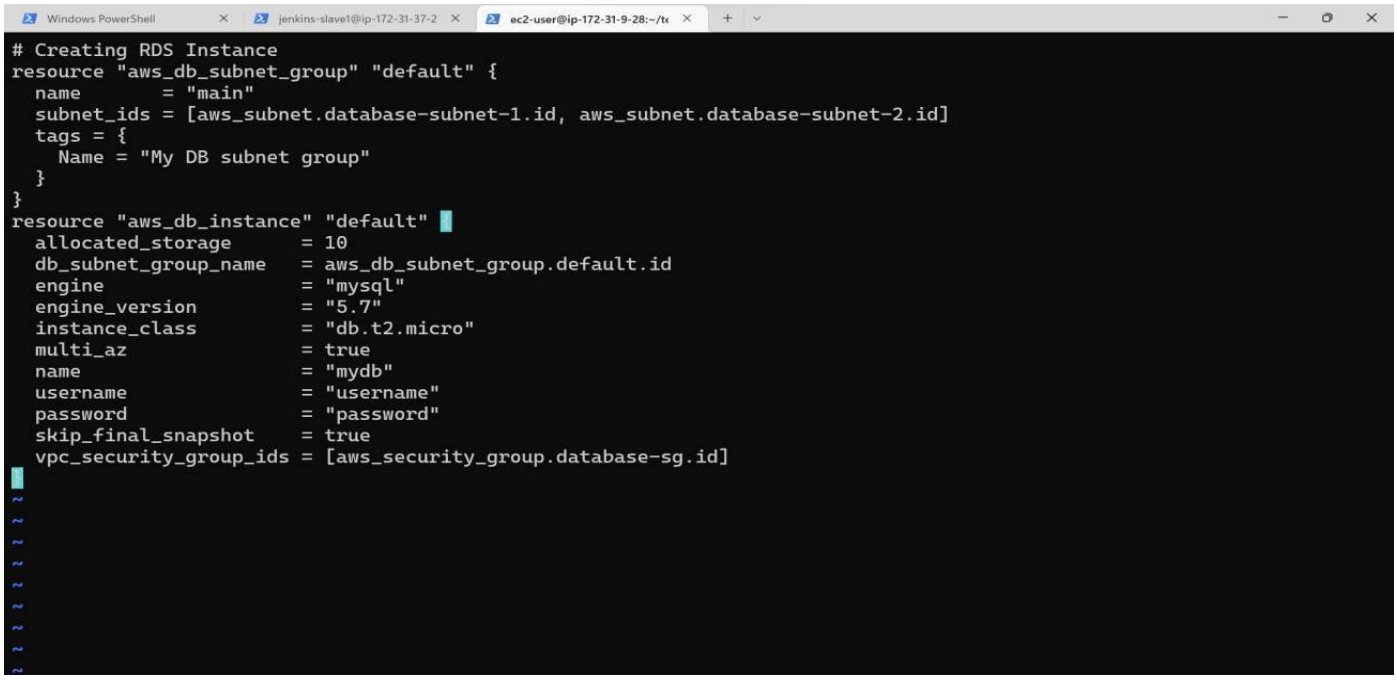
Terraform apply

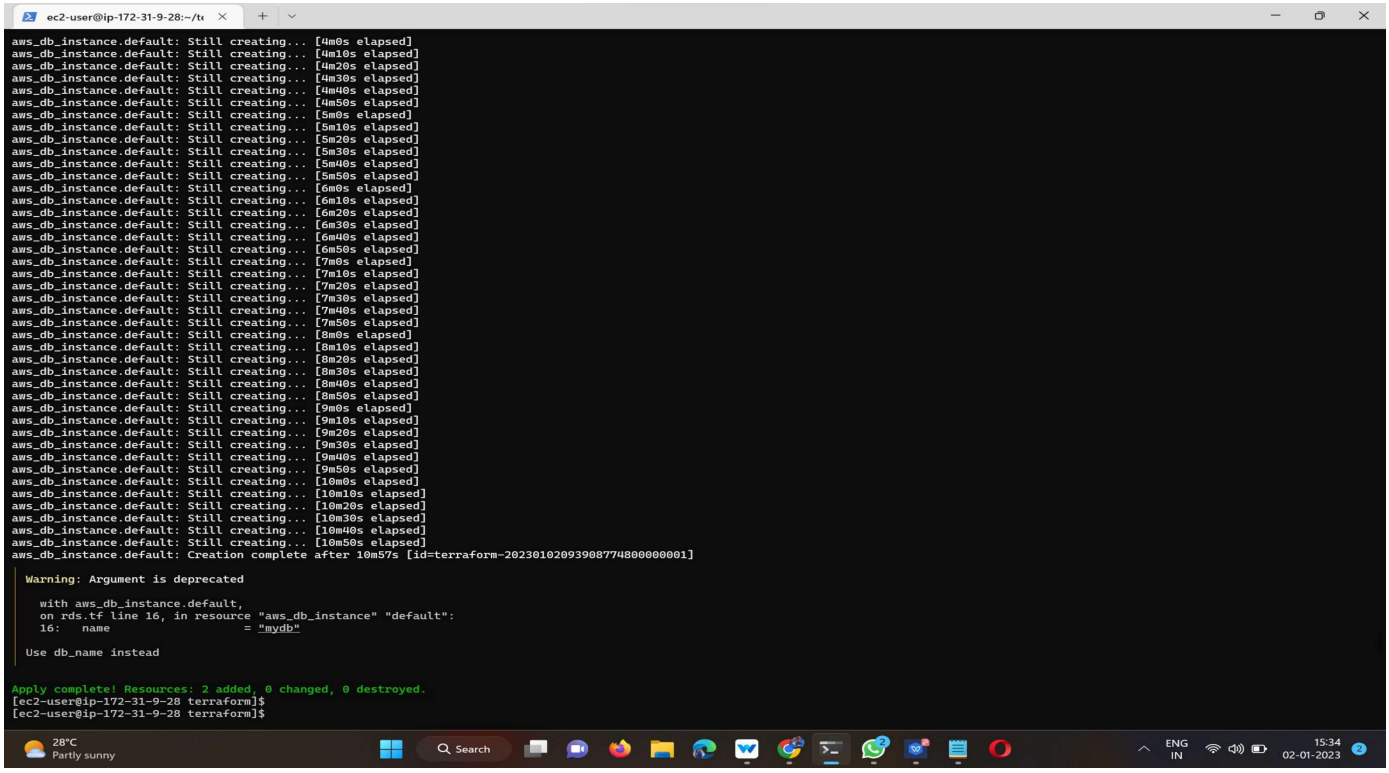
```
      + target_id        = "i-04f152afcd678f0ad"
    }

  # aws_lb_target_group_attachment.attachment1 will be created
  + resource "aws_lb_target_group_attachment" "attachment1" {
      + id               = (known after apply)
      + port             = 80
      + target_group_arn = (known after apply)
      + target_id        = "i-0fecb1145449837b9"
    }

Plan: 5 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_lb_target_group.target-elb: Creating...
aws_lb.external-alb: Creating...
aws_lb_target_group.target-elb: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4]
aws_lb_target_group_attachment.attachment: Creating...
aws_lb_target_group_attachment.attachment1: Creating...
aws_lb_target_group_attachment.attachment1: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4
-20230102093533080100000001]
aws_lb_target_group_attachment.attachment: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4-
20230102093533098600000002]
aws_lb.external-alb: Still creating... [10s elapsed]
aws_lb.external-alb: Still creating... [20s elapsed]
aws_lb.external-alb: Still creating... [30s elapsed]
aws_lb.external-alb: Still creating... [40s elapsed]
aws_lb.external-alb: Still creating... [50s elapsed]
aws_lb.external-alb: Still creating... [1m0s elapsed]
aws_lb.external-alb: Still creating... [1m10s elapsed]
aws_lb.external-alb: Still creating... [1m20s elapsed]
aws_lb.external-alb: Still creating... [1m30s elapsed]
aws_lb.external-alb: Still creating... [1m40s elapsed]
aws_lb.external-alb: Still creating... [1m50s elapsed]
aws_lb.external-alb: Still creating... [2m0s elapsed]
aws_lb.external-alb: Creation complete after 2m1s [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:loadbalancer/app/External-LB/4d513aa75b7adcd5]
aws_lb_listener.external-elb: Creating...
aws_lb_listener.external-elb: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:listener/app/External-LB/4d513aa75b7adcd5/a7ce67a
d51b9ca57]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-9-28 terraform]$
```

Create rds.tf and write the following script

multi-az is set to true for the high availability

```
[ec2-user@ip-172-31-9-28 terraform]$ vi  rds.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

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-172-31-9-28 terraform]$ terraform apply
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_vpc.demovpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_internet_gateway.igw1: Refreshing state... [id=igw-07a6c4ccd2deb84fd]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]
aws_lb_target_group.target-elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0aec0f65383dcf4e8]
aws_route_table.route: Refreshing state... [id=rtb-088e69f09ebaa0f31]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04f78b3b8cd4a10aa]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0934661eef3871db8]
aws_security_group.demosg: Refreshing state... [id=sg-02ba909d05b7cf2f0]
aws_instance.demoinstance: Refreshing state... [id=i-04f152afcd678f0ad]
aws_security_group.database-sg: Refreshing state... [id=sg-0b4af5731c7dd254d]
aws_route_table_association.rt1: Refreshing state... [id=rtbassoc-027b6579877e1e560]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-0a74b3bf113be85c3]
aws_lb.external-alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:loadbalancer/app/External-LB/4d513aa75b7adcd5]
aws_instance.demoinstance1: Refreshing state... [id=i-0fecb1145449837b9]
aws_lb_listener.external-elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:listener/app/External-LB/4d513aa75b7adcd5/a7ce67ad51b9ca
57]
aws_lb_target_group_attachment.attachment1: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4-202301
02093533080100000001]
aws_lb_target_group_attachment.attachment: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4-2023010
209353309860000002]

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





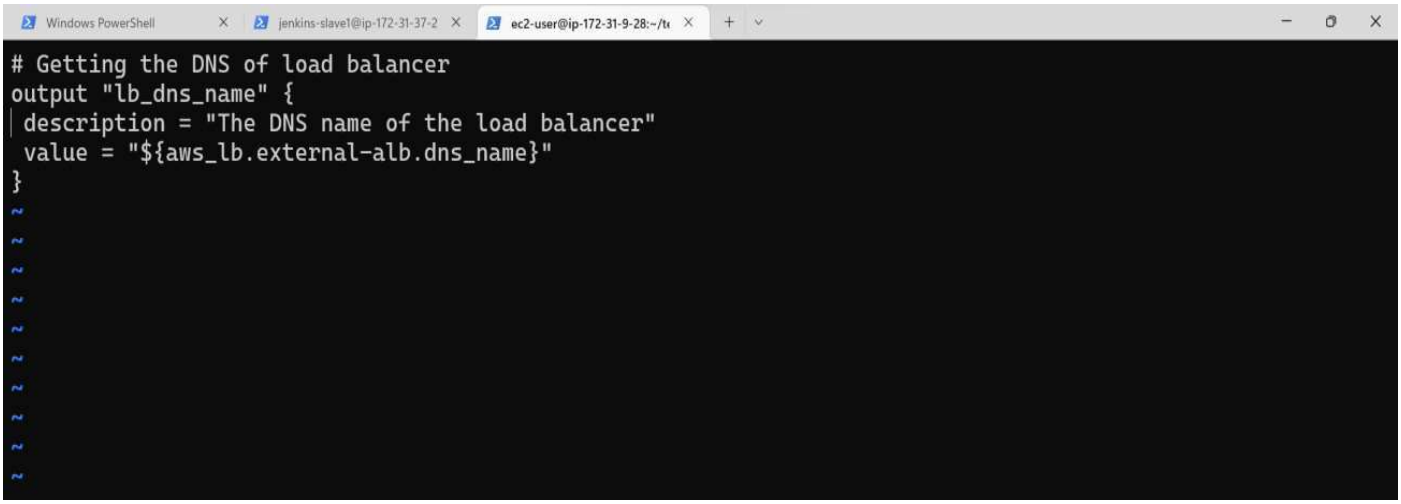
And create the file vi variable.tf

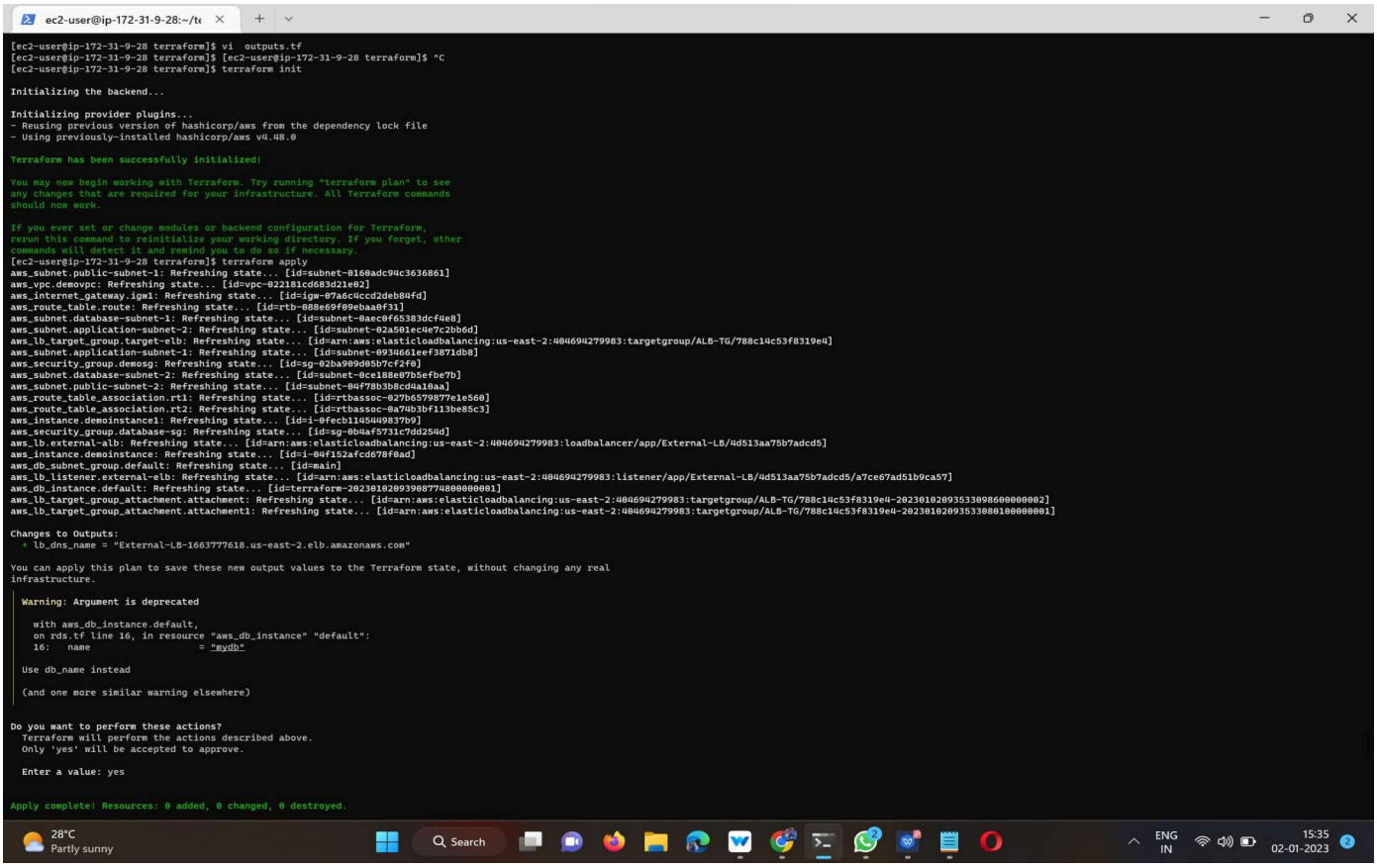
```
# Defining CIDR Block for VPC
variable "vpc_cidr" {
  default = "10.0.0.0/16"
}
# Defining CIDR Block for 1st Subnet
variable "subnet_cidr" {
  default = "10.0.1.0/24"
}
# Defining CIDR Block for 2nd Subnet
variable "subnet1_cidr" {
  default = "10.0.2.0/24"
}
# Defining CIDR Block for 3rd Subnet
variable "subnet2_cidr" {
  default = "10.0.3.0/24"
}
# Defining CIDR Block for 3rd Subnet
variable "subnet3_cidr" {
  default = "10.0.4.0/24"
}
# Defining CIDR Block for 3rd Subnet
variable "subnet4_cidr" {
  default = "10.0.5.0/24"
}
# Defining CIDR Block for 3rd Subnet
variable "subnet5_cidr" {
  default = "10.0.6.0/24"
}
```

```
[ec2-user@ip-172-31-9-28 terraform]$
[ec2-user@ip-172-31-9-28 terraform]$ vi vars.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

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-172-31-9-28 terraform]$ terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_internet_gateway.igw1: Refreshing state... [id=igw-07a6c4ccd2deb84fd]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_lb_target_group.target-elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0aec0f65383dcf4e8]
aws_security_group.demosg: Refreshing state... [id=sg-02ba909d05b7cf2f0]
aws_route_table.route: Refreshing state... [id=rtb-088e69f09ebaa0f31]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0934661eef3871db8]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04f78b3b8cd4a10aa]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_route_table_association.rt1: Refreshing state... [id=rtbassoc-027b65798f7e1e560]
aws_security_group.database-sg: Refreshing state... [id=sg-0b4af5731c7dd254d]
aws_db_subnet_group.default: Refreshing state... [id=main]
aws_instance.demoinstance: Refreshing state... [id=i-04f152afcd678f0ad]
aws_lb.external-alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:loadbalancer/app/External-LB/4d513aa75b7adcd5]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-0a74b3bf113be85c3]
aws_instance.demoinstance1: Refreshing state... [id=i-0fecb1145449837b9]
aws_lb_listener.external-elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:listener/app/External-LB/4d513aa75b7adcd5/a7ce67ad51b9ca57]
aws_db_instance.default: Refreshing state... [id=terraform-20230102093908774800000001]
aws_lb_target_group_attachment.attachment: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4-20230102093533098600000002]
aws_lb_target_group_attachment.attachment1: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4-20230102093533080100000001]

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.

  Warning: Argument is deprecated

    with aws_db_instance.default,
    on rds.tf line 16, in resource "aws_db_instance" "default":
    16:   name                  = "mydb"

  Use db_name instead

  (and one more similar warning elsewhere)

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

Outputs:

lb_dns_name = "External-LB-1663777618.us-east-2.elb.amazonaws.com"
[ec2-user@ip-172-31-9-28 terraform]$
[ec2-user@ip-172-31-9-28 terraform]$
```

We will get the DNS of the application load balancer.

And create vi output.tf


Windows PowerShell
jenkins-slave1@ip-172-31-37-2
ec2-user@ip-172-31-9-28:~/te
# Getting the DNS of load balancer
output "lb_dns_name" {
description = "The DNS name of the load balancer"
value = "${aws_lb.external-alb.dns_name}"
}


ec2-user@ip-172-31-9-28:~/te
[ec2-user@ip-172-31-9-28 terraform]$ vi outputs.tf
[ec2-user@ip-172-31-9-28 terraform]$ [ec2-user@ip-172-31-9-28 terraform]$ ^C
[ec2-user@ip-172-31-9-28 terraform]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0
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-172-31-9-28 terraform]$ terraform apply
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_vpc.demovpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_internet_gateway.igw1: Refreshing state... [id=igw-07a6c4ccd2deb84fd]
aws_route_table.route: Refreshing state... [id=rtb-088e69f09ebaa0f31]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0aec0f65383dcf4e8]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]
aws_lb_target_group.target-elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-093466leef3871db8]
aws_security_group.demosg: Refreshing state... [id=sg-02ba909d05b7cf2f0]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04f78b3b8cd4a10aa]
aws_route_table_association.rt1: Refreshing state... [id=rtbassoc-027b65799877e1e560]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-0a70b3bf113be85c3]
aws_instance.demoinstance1: Refreshing state... [id=i-0fecb1145449837b9]
aws_security_group.database-sg: Refreshing state... [id=sg-0b4af5731c7dd254d]
aws_lb.external-alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:loadbalancer/app/External-LB/4d513aa75b7adcd5]
aws_instance.demoinstance: Refreshing state... [id=i-04f152afcd678f0ad]
aws_db_subnet_group.default: Refreshing state... [id=main]
aws_lb_listener.external-elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:listener/app/External-LB/4d513aa75b7adcd5/a7ce67ad51b9ca57]
aws_db_instance.default: Refreshing state... [id=terraform-20230102093908774800000001]
aws_lb_target_group_attachment.attachment: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4-20230102093533098600000002]
aws_lb_target_group_attachment.attachment1: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4-20230102093533080100000001]
Changes to Outputs:
+ lb_dns_name = "External-LB-1663777618.us-east-2.elb.amazonaws.com"
You can apply this plan to save these new output values to the Terraform state, without changing any real
infrastructure.
Warning: Argument is deprecated
with aws_db_instance.default,
on rds.tf line 16, in resource "aws_db_instance" "default":
16: name = "mydb"
Use db_name instead
(and one more similar warning elsewhere)
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
Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
28°C
Partly sunny
Search
ENG
IN
15:35
02-01-2023