PROJECT-1

Deploy Three Tier Architecture In AWS Using Terraform

Task-1:- Create a provider file and initialize the terraform folder by using “terraform init” command.

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.6"
    }
  }
}
provider "aws" {
  profile    = "default"
  region     = "us-east-1c"
  access_key = "AKIA5PWVUBL44F2DTFM2"
  secret_key = "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"
}
```

```
[ec2-user@ip-172-31-82-108 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 v4.67.0

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-172-31-82-108 Terraform]$ terraform fmt
[ec2-user@ip-172-31-82-108 Terraform]$ terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-31-82-108 Terraform]$ 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-172-31-82-108 Terraform]$ 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.
[ec2-user@ip-172-31-82-108 Terraform]$ ll
total 8
```

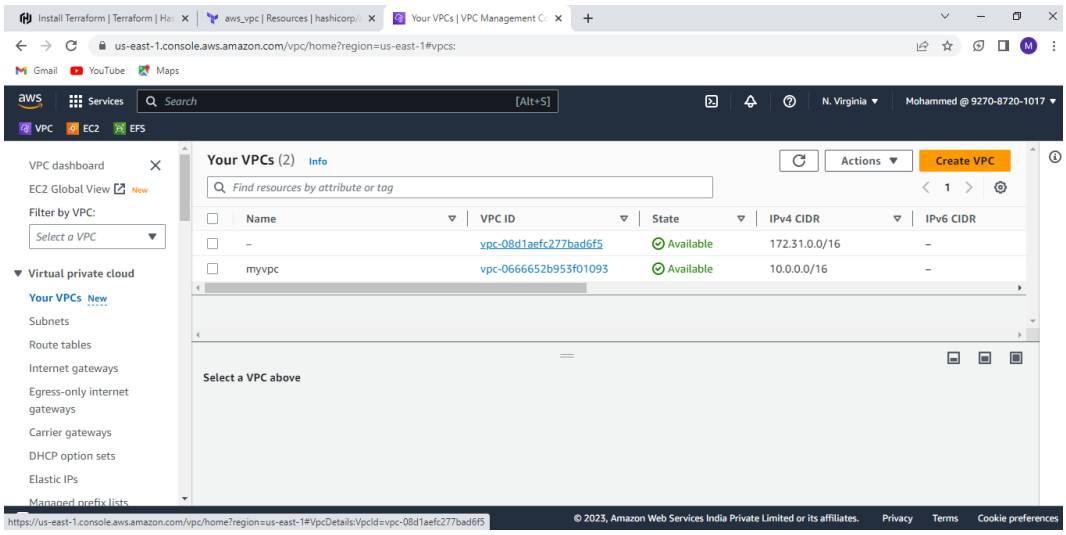
Task-2:- Create a file for aws vpc (vpc.tf) and write a script for vpc in the file by using vi command.

- Apply Terraform fmt, terraform validate, terraform plan and terraform apply commands after writing the script.

ec2-user@ip-172-31-82-108:~/Terraform

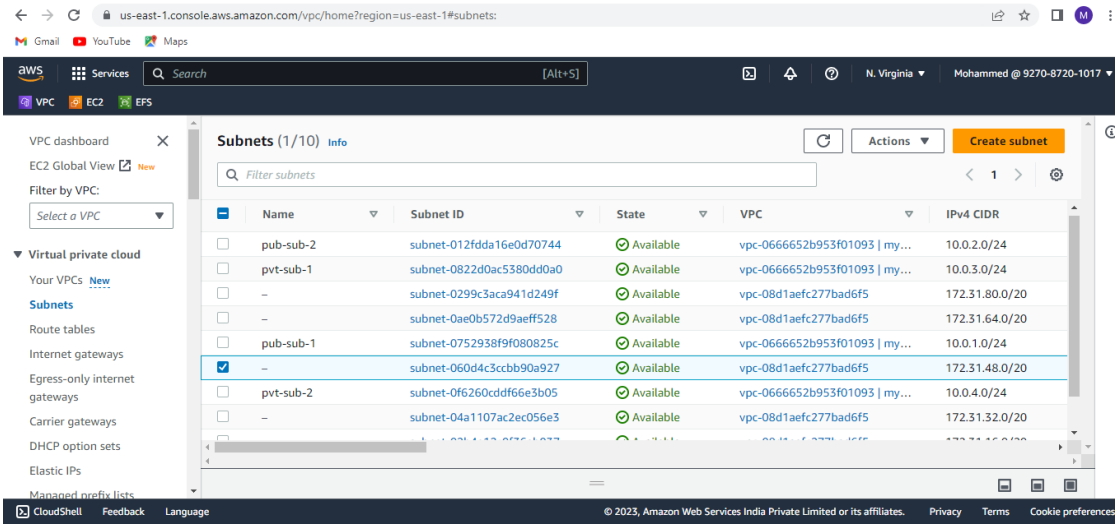
```
resource "aws_vpc" "mmyvpc" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "myvpc"
  }
}
~
~
~
~
```

```
ec2-user@ip-172-31-82-108:~/Terraform
[ec2-user@ip-172-31-82-108 Terraform]$ cat provider
cat: provider: No such file or directory
[ec2-user@ip-172-31-82-108 Terraform]$ vi provider.tf
[ec2-user@ip-172-31-82-108 Terraform]$ cat provider.tf
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.6"
    }
  }
}
provider "aws" {
  profile    = "default"
  region     = "us-east-1c"
  access_key = "AKIASPWVUBL44F2DTFM2"
  secret_key = "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"
}
[ec2-user@ip-172-31-82-108 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 v4.67.0

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-172-31-82-108 Terraform]$ terraform fmt
[ec2-user@ip-172-31-82-108 Terraform]$ terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-31-82-108 Terraform]$ 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-172-31-82-108 Terraform]$ 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.
```

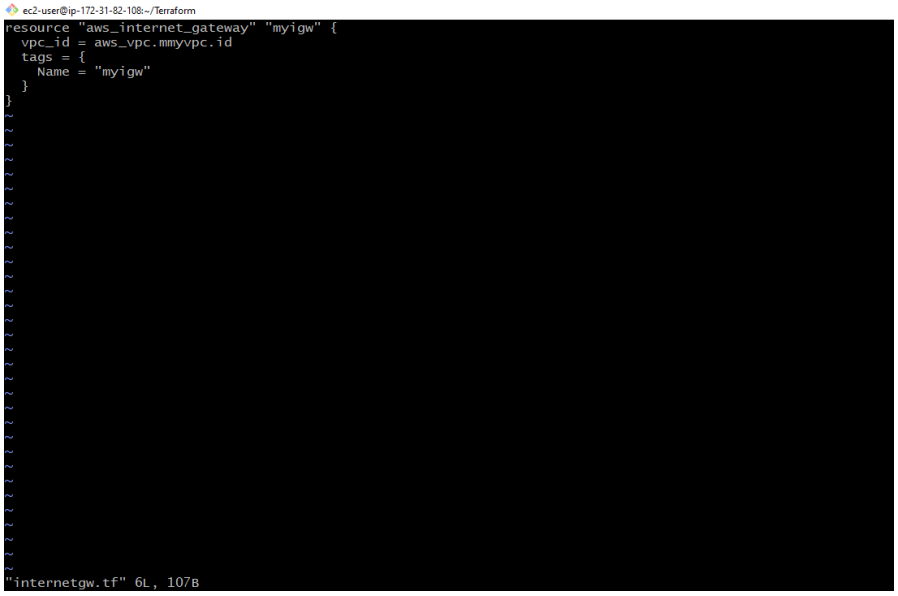


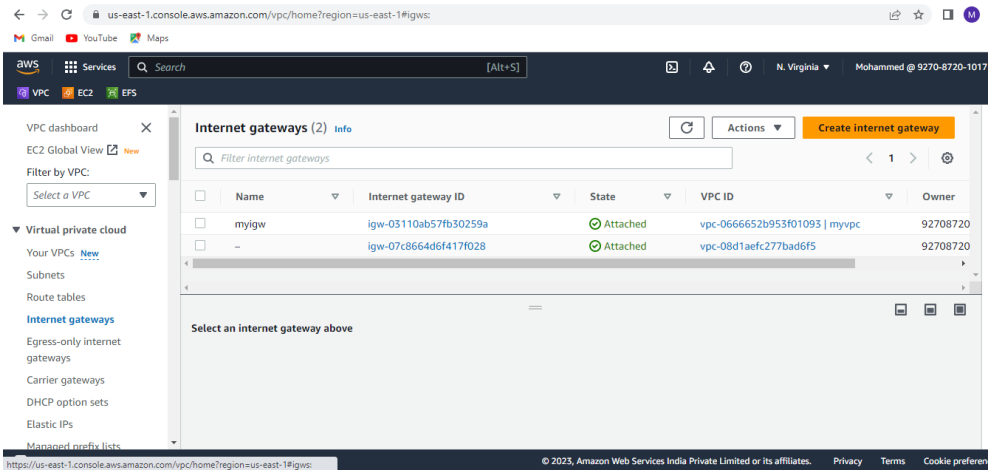
Task-3:- Create a file AWS subnet resource.

```
ec2-user@ip-172-31-82-108:~/Terraform
#creating 1st web subnet
resource "aws_subnet" "pub-sub-1" {
  vpc_id                  = "aws_vpc.mmyvpc.id"
  cidr_block              = "10.0.1.0/24"
  map_public_ip_on_launch = true
  availability_zone       = "us-east-1a"
  tags = {
    Name = "pub-sub-1"
  }
}

#creating 2nd web subnet
resource "aws_subnet" "pub-sub-2" {
  vpc_id                  = "aws_vpc.mmyvpc.id"
  cidr_block              = "10.0.2.0/24"
  map_public_ip_on_launch = true
  availability_zone       = "us-east-1b"
  tags = {
    Name = "pub-sub-2"
  }
}
#creating 3rd web subnet
resource "aws_subnet" "pvt-sub-1" {
  vpc_id                  = "aws_vpc.mmyvpc.id"
  cidr_block              = "10.0.3.0/24"
  map_public_ip_on_launch = false
  availability_zone       = "us-east-1a"
  tags = {
    Name = "pvt-sub-1"
  }
}
#creating 4th web subnet
resource "aws_subnet" "pvt-sub-2" {
  vpc_id                  = "aws_vpc.mmyvpc.id"
  cidr_block              = "10.0.4.0/24"
  map_public_ip_on_launch = false
  availability_zone       = "us-east-1b"
  tags = {
    Name = "pvt-sub-2"
```

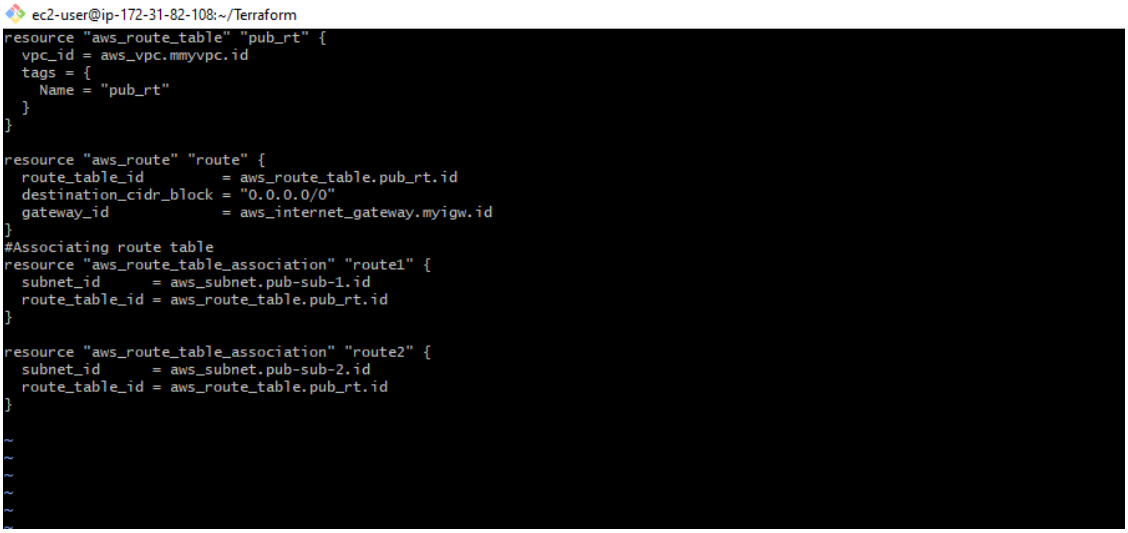


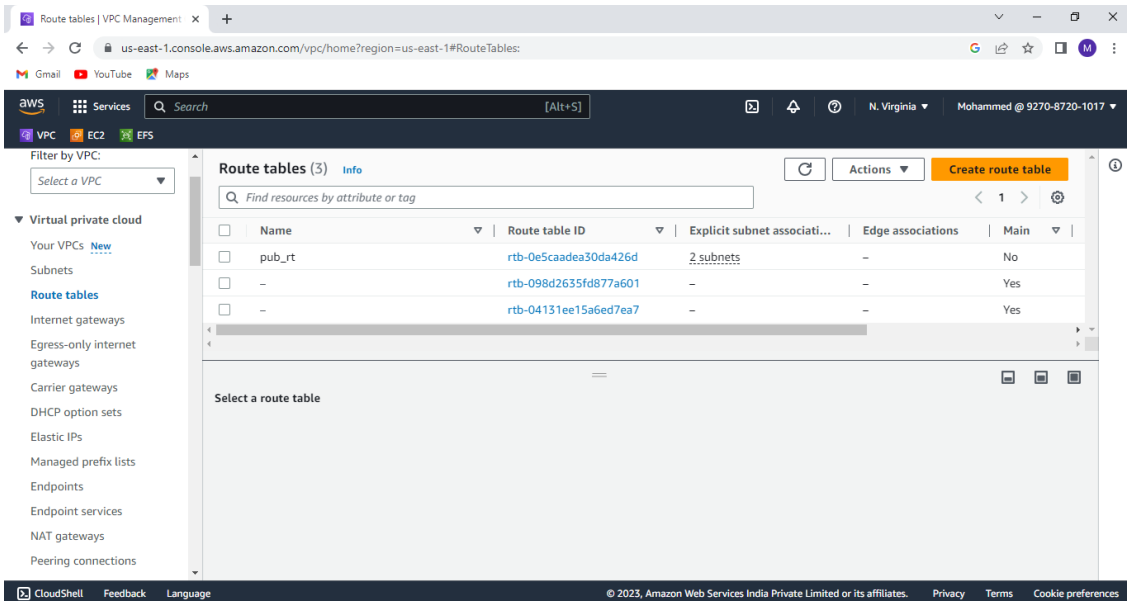
Task-4:- Create a file for Internet gateway.





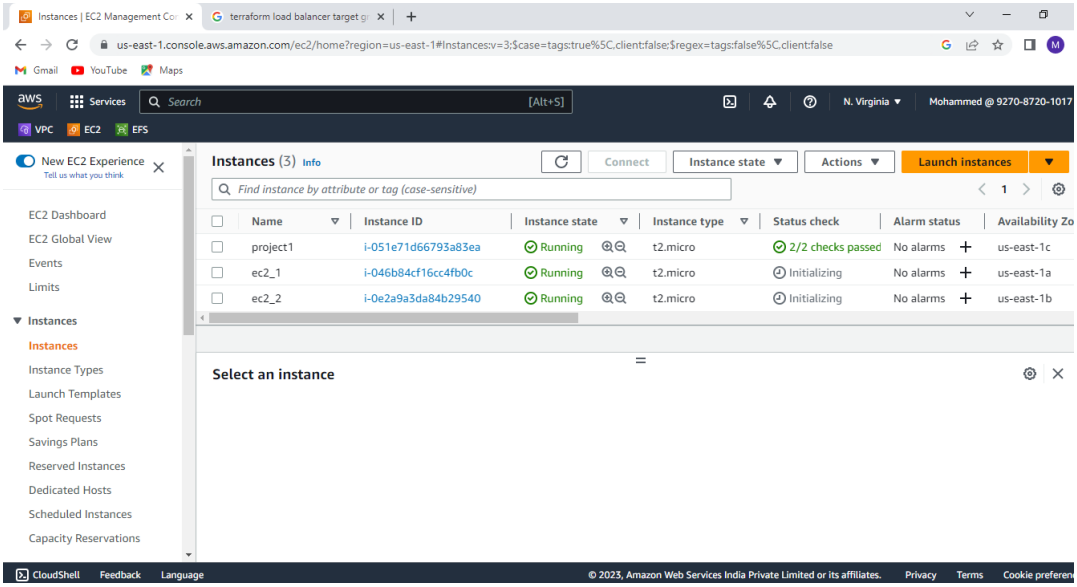
Task-5:- Create a file for Routetable.





Task-6:- Create a file for AWS EC2 Instance Resource.

```
ec2-user@ip-172-31-82-108:~/Terraform
resource "aws_instance" "ec2_1" {
  ami                         = "ami-0715c1897453cabd1"
  instance_type               = "t2.micro"
  count                       = 1
  key_name                    = "mykey"
  vpc_security_group_ids      = [aws_security_group.websg.id]
  subnet_id                   = aws_subnet.pub-sub-1.id
  associate_public_ip_address = true

  tags = {
    Name = "ec2_1"
  }
}

resource "aws_instance" "ec2_2" {
  ami                         = "ami-0715c1897453cabd1"
  instance_type               = "t2.micro"
  count                       = 1
  key_name                    = "mykey"
  vpc_security_group_ids      = [aws_security_group.websg.id]
  subnet_id                   = aws_subnet.pub-sub-2.id
  associate_public_ip_address = true

  tags = {
    Name = "ec2_2"
  }
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"ec2.tf" 28L, 798B
```



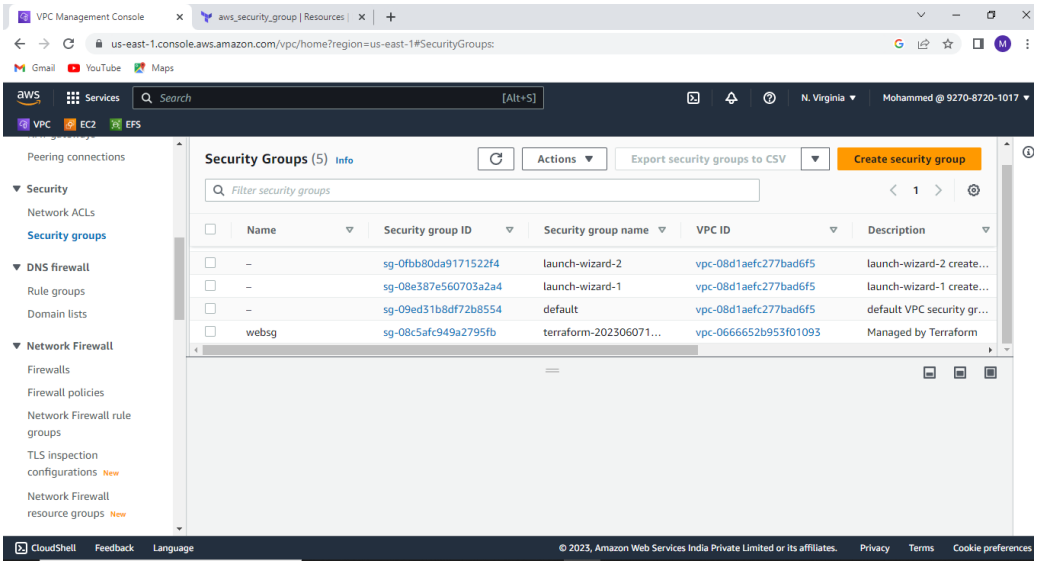
Task-7:- Create a file for Security Group for the Frontend Tier.

ec2-user@ip-172-31-82-108:~/Terraform

```
resource "aws_security_group" "websg" {
   vpc_id = aws_vpc.mmyvpc.id

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

#HTTPS access from anywhere
 ingress {
    from_port       = 443
    to_port         = 443
    protocol        = "tcp"
    cidr_block      = ["0.0.0.0/0"]
}

#ssh access from anywhere
 ingress {
    from_port       = 22
    to_port         = 22
    protocol        = "tcp"
    cidr_block      = ["0.0.0.0/0"]
}

#outbound rules
 egress {
    from_port       = 0
    to_port         = 0
    protocol        = "-1"
    cidr_block      = ["0.0.0.0/0"]
}
tags = {
  Name = "websg"
  }
}
~
~
~
~
~
~
-- INSERT --
```

ec2-user@ip-172-31-82-108:~/Terraform

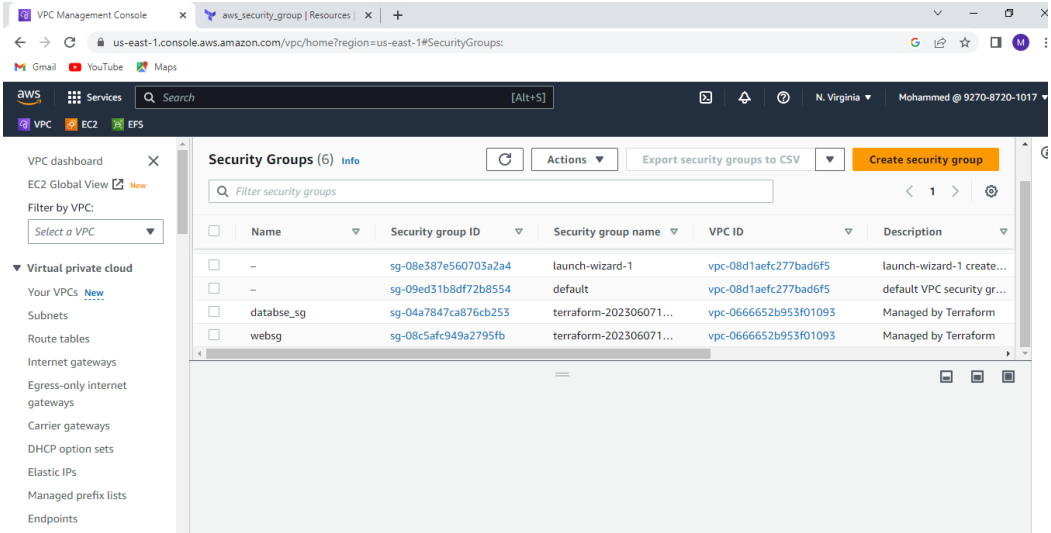
```
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 443
          },
        + {
            + cidr_blocks      = [
              + "0.0.0.0/0",
              ]
            + description      = ""
            + from_port        = 80
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "tcp"
            + security_groups  = []
            + self             = false
            + to_port          = 80
          },
      ]
    + name                   = (known after apply)
    + name_prefix            = (known after apply)
    + owner_id               = (known after apply)
    + revoke_rules_on_delete = false
    + tags                   = {
        + "Name" = "websg"
      }
    + tags_all               = {
        + "Name" = "websg"
      }
    + vpc_id                 = "vpc-0666652b953f01093"
  }

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_security_group.websg: Creating...
aws_security_group.websg: Creation complete after 2s [id=sg-08c5afc949a2795fb]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-82-108 Terraform]$
```



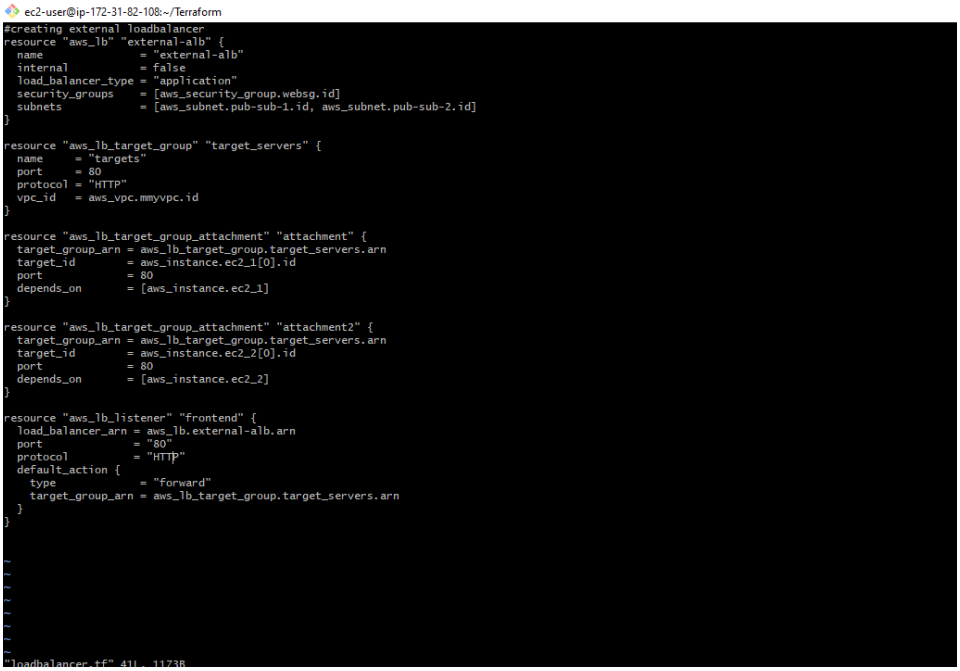
Task-8:- Create a file for Security Group for the Database Tier.

```
ec2-user@ip-172-31-82-108:~/Terraform
resource "aws_security_group" "databasesg" {
    vpc_id = aws_vpc.mmyvpc.id


    ingress {
         description      = "TLS from VPC"
         from_port        = 3306
         to_port          = 3306
         protocol         = "tcp"
         security_groups  = [aws_security_group.websg.id]
    }

    egress {
        from_port        = 32768
        to_port          = 65535
        protocol         = "-1"
        cidr_blocks      = ["0.0.0.0/0"]
    }

    tags = {
       Name = "databse_sg"
    }
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

```
ec2-user@ip-172-31-82-108:~/Terraform
Terraform will perform the following actions:

  # aws_security_group.databasesg is tainted, so must be replaced
-/+ resource "aws_security_group" "databasesg" {
      ~ arn                    = "arn:aws:ec2:us-east-1:927087201017:security-group/sg-073db4e2b0f73ae06" -> (known after apply)
      ~ egress                 = [
          + {
              + cidr_blocks      = [
                  + "0.0.0.0/0",
                ]
              + description      = ""
              + from_port        = 0
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "-1"
              + security_groups  = []
              + self             = false
              + to_port          = 0
            },
        ]
      ~ id                     = "sg-073db4e2b0f73ae06" -> (known after apply)
      ~ name                   = "terraform-20230607113920852700000001" -> (known after apply)
      ~ name_prefix            = "terraform-" -> (known after apply)
      ~ owner_id               = "927087201017" -> (known after apply)
        tags                   = {
            "Name" = "databse_sg"
        }
        # (5 unchanged attributes hidden)
    }

Plan: 1 to add, 0 to change, 1 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_security_group.databasesg: Destroying... [id=sg-073db4e2b0f73ae06]
aws_security_group.databasesg: Destruction complete after 0s
aws_security_group.databasesg: Creating...
aws_security_group.databasesg: Creation complete after 2s [id=sg-04a7847ca876cb253]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
[ec2-user@ip-172-31-82-108 Terraform]$
```

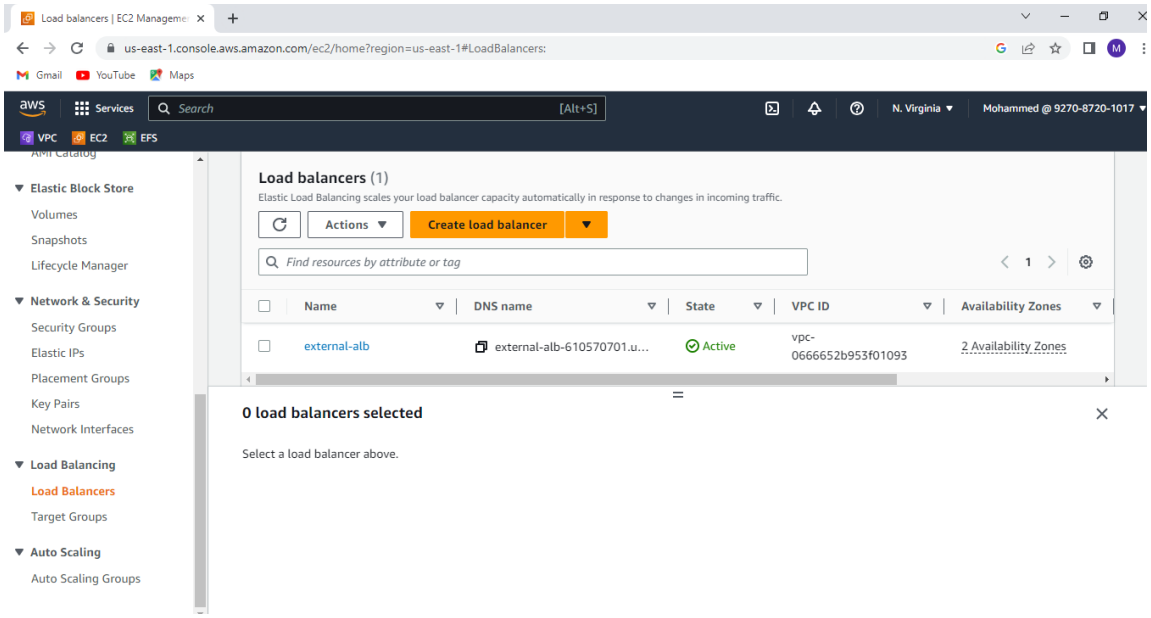


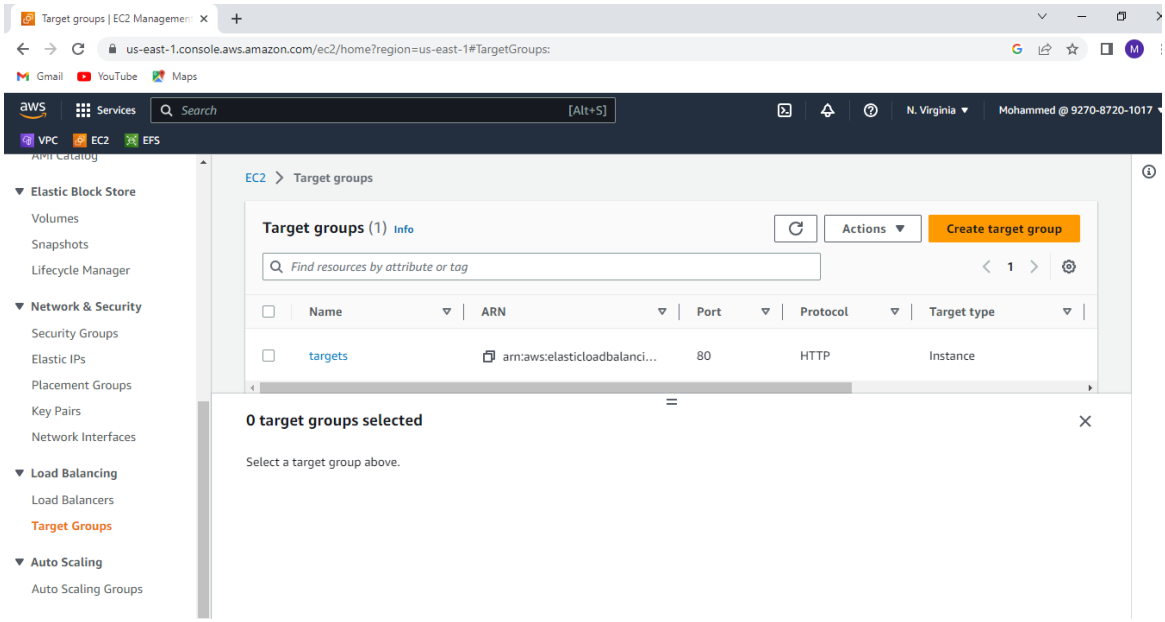
Task-9:- Create a file Application Load Balancer.



ec2-user@ip-172-31-82-108:~/Terraform

```
#creating external loadbalancer
resource "aws_lb" "external-alb" {
  name               = "external-alb"
  internal           = false
  load_balancer_type = "application"
  security_groups    = [aws_security_group.websg.id]
  subnets            = [aws_subnet.pub-sub-1.id, aws_subnet.pub-sub-2.id]
}

resource "aws_lb_target_group" "target_servers" {
  name     = "targets"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.mmyvpc.id
}

resource "aws_lb_target_group_attachment" "attachment" {
  target_group_arn = aws_lb_target_group.target_servers.arn
  target_id        = aws_instance.ec2_1[0].id
  port             = 80
  depends_on       = [aws_instance.ec2_1]
}

resource "aws_lb_target_group_attachment" "attachment2" {
  target_group_arn = aws_lb_target_group.target_servers.arn
  target_id        = aws_instance.ec2_2[0].id
  port             = 80
  depends_on       = [aws_instance.ec2_2]
}

resource "aws_lb_listener" "frontend" {
  load_balancer_arn = aws_lb.external-alb.arn
  port              = "80"
  protocol          = "HTTP"
  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.target_servers.arn
  }
}


~
~
~
~
~
~
~
~
"loadbalancer.tf" 41L, 1173B
```

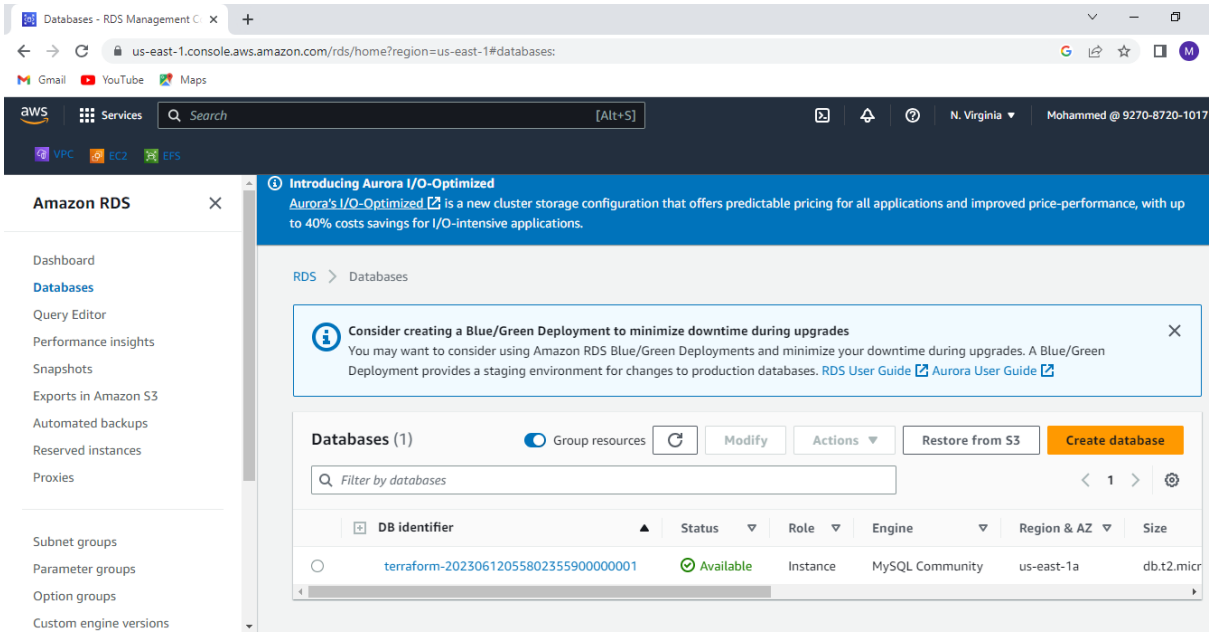

Load balancers | EC2 Manageme
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers:
Gmail
YouTube
Maps
aws
Services
Search
[Alt+S]
N. Virginia
Mohammed @ 9270-8720-1017
VPC
EC2
EFS
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Auto Scaling
Auto Scaling Groups
Load balancers (1)
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.
Actions
Create load balancer
Find resources by attribute or tag
Name
DNS name
State
VPC ID
Availability Zones
external-alb
external-alb-610570701.u...
Active
vpc-0666652b953f01093
2 Availability Zones
0 load balancers selected
Select a load balancer above.


Target groups | EC2 Managemen
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroups:
Gmail
YouTube
Maps
aws
Services
Search
[Alt+S]
N. Virginia
Mohammed @ 9270-8720-1017
VPC
EC2
EFS
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Auto Scaling
Auto Scaling Groups
EC2 > Target groups
Target groups (1) Info
Actions
Create target group
Find resources by attribute or tag
Name
ARN
Port
Protocol
Target type
targets
arn:aws:elasticloadbalanci...
80
HTTP
Instance
0 target groups selected
Select a target group above.

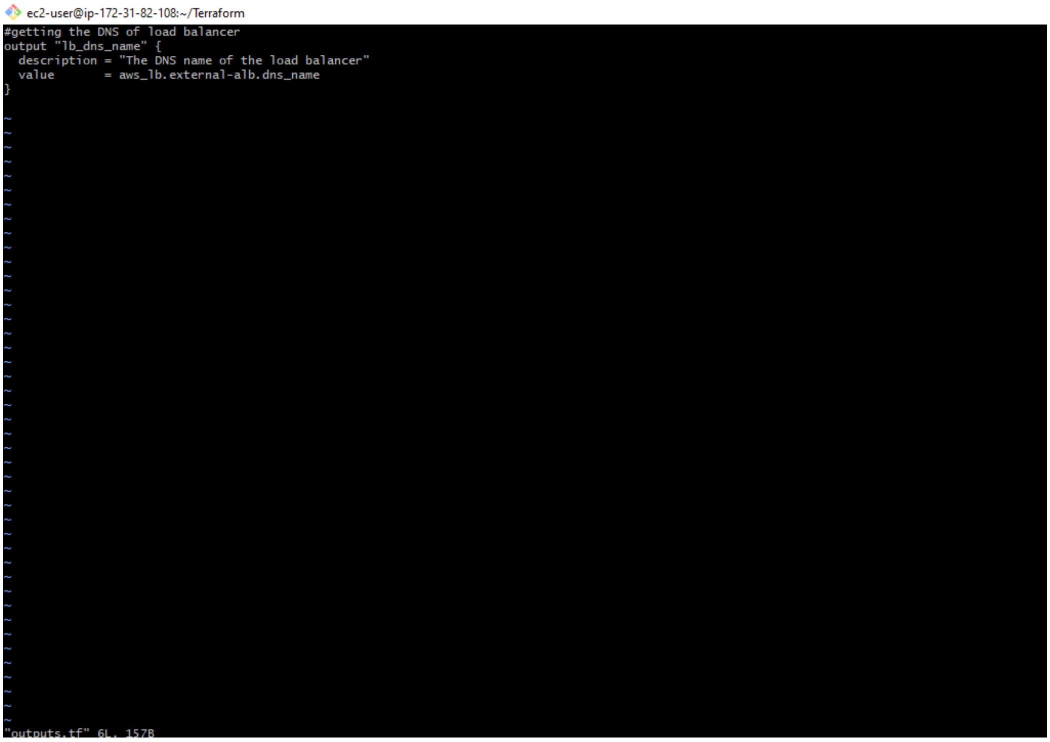
Task-10:- Create a file for RDS Instance.

ec2-user@ip-172-31-82-108:~/Terraform

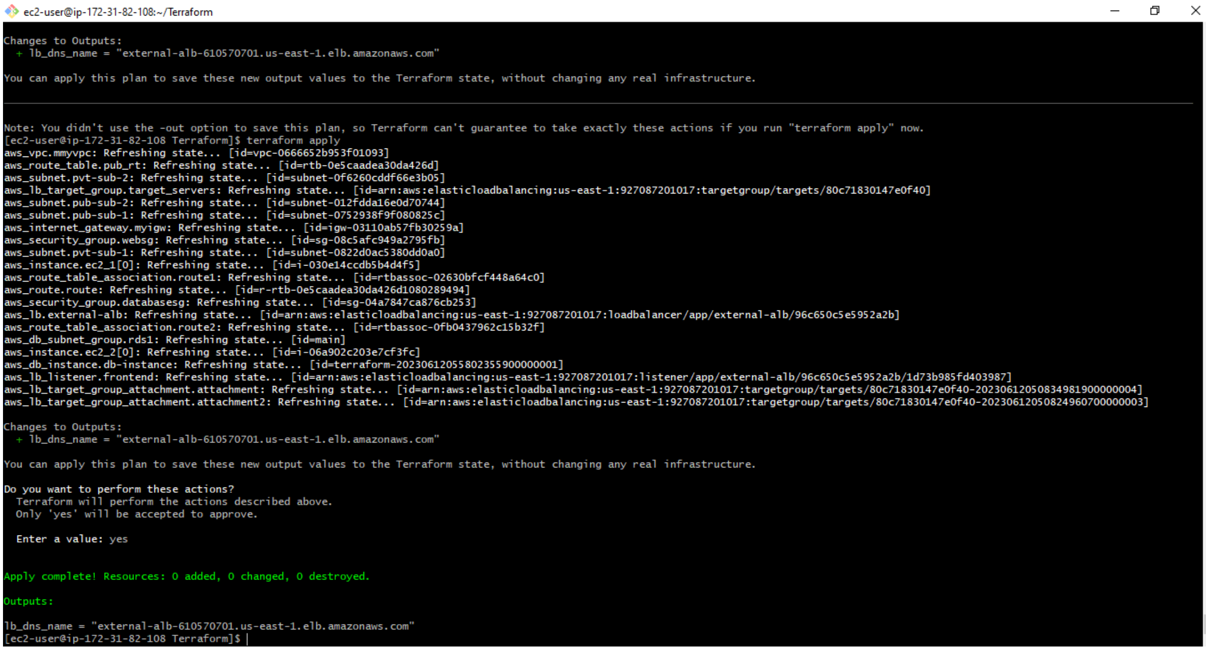
```
#creating RDS instance
resource "aws_db_subnet_group" "rds1" {
  name       = "main"
  subnet_ids = [aws_subnet.pvt-sub-1.id, aws_subnet.pvt-sub-2.id]
  tags = {
    Name = " DB subnet group"
  }
}
resource "aws_db_instance" "db-instance" {
  allocated_storage      = 10
  db_subnet_group_name   = aws_db_subnet_group.rds1.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.databasesg.id]
  tags = {
    Name = "mydb"
  }
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"RDS.tf" 24L, 673B
```



Task-11:- Create a file for Outputs.



```
#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
}

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"outputs.tf" 6L, 157B
```



```
Changes to Outputs:
  + lb_dns_name = "external-alb-610570701.us-east-1.elb.amazonaws.com"

You can apply this plan to save these new output values to the Terraform state, without changing any real infrastructure.

─────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────

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.
[ec2-user@ip-172-31-82-108 Terraform]$ terraform apply
aws_vpc.mmyvpc: Refreshing state... [id=vpc-0666652b953f01093]
aws_route_table.pub_rt: Refreshing state... [id=rtb-0e5caadea30da426d]
aws_subnet.pvt-sub-2: Refreshing state... [id=subnet-0f6260cddf66e3b05]
aws_lb_target_group.target_servers: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:927087201017:targetgroup/targets/80c71830147e0f40]
aws_subnet.pub-sub-2: Refreshing state... [id=subnet-012fdda16e0d70744]
aws_subnet.pub-sub-1: Refreshing state... [id=subnet-0752938f9f080825c]
aws_internet_gateway.myigw: Refreshing state... [id=igw-03110ab57fb30259a]
aws_security_group.websg: Refreshing state... [id=sg-08c5afc949a2795fb]
aws_subnet.pvt-sub-1: Refreshing state... [id=subnet-0822d0ac5380dd0a0]
aws_instance.ec2_1[0]: Refreshing state... [id=i-030e14ccdb5b4d4f5]
aws_route_table_association.route1: Refreshing state... [id=rtbassoc-02630bfcf448a64c0]
aws_route.route: Refreshing state... [id=r-rtb-0e5caadea30da426d1080289494]
aws_security_group.databasesg: Refreshing state... [id=sg-04a7847ca876cb253]
aws_lb.external-alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:927087201017:loadbalancer/app/external-alb/96c650c5e5952a2b]
aws_route_table_association.route2: Refreshing state... [id=rtbassoc-0fb0437962c15b32f]
aws_db_subnet_group.rds1: Refreshing state... [id=main]
aws_instance.ec2_2[0]: Refreshing state... [id=i-06a902c203e7cf3fc]
aws_db_instance.db-instance: Refreshing state... [id=terraform-20230612055802355900000001]
aws_lb_listener.frontend: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:927087201017:listener/app/external-alb/96c650c5e5952a2b/1d73b985fd403987]
aws_lb_target_group_attachment.attachment: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:927087201017:targetgroup/targets/80c71830147e0f40-20230612050834981900000004]
aws_lb_target_group_attachment.attachment2: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:927087201017:targetgroup/targets/80c71830147e0f40-20230612050824960700000003]

Changes to Outputs:
  + lb_dns_name = "external-alb-610570701.us-east-1.elb.amazonaws.com"

You can apply this plan to save these new output values to the Terraform state, without changing any real infrastructure.

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.

Outputs:

lb_dns_name = "external-alb-610570701.us-east-1.elb.amazonaws.com"
[ec2-user@ip-172-31-82-108 Terraform]$ |
```

Task-12:- Create a file for user data with .sh extension.

ec2-user@ip-172-31-82-108:~/Terraform

```
#!/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
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Task-13:- Verify the resource.

Terraform created below resources

- Vpc
- Public&private subnets
- Route tables
- Internet Gateway
- EC2 instances
- RDS instance
- Application Load Balancer
- Security Groups for web & RDS instance

TERRAFORM SCRIPT AUTOMATION WITH JENKIN

Step-1:- Launch a EC2 instance by giving jenkins port number 8080 in security group.

Step-2:- Connect the EC2 Instance.

Install Jenkins in EC2 instance by executing following commads.

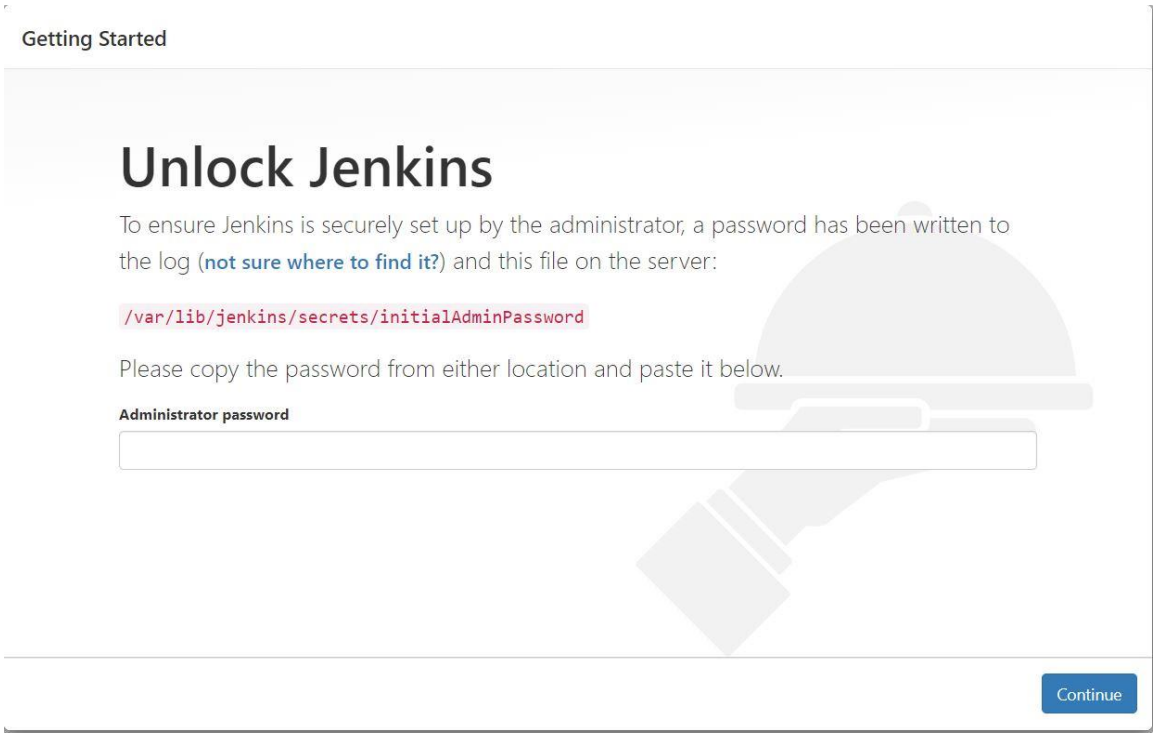
- `sudo yum update -y`
- `sudo wget -O /etc/yum.repos.d/jenkins.repo \ https://pkg.jenkins.io/redhat-stable/jenkins.repo`
- `sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key`
- `sudo yum upgrade`
- `sudo amazon-linux-extras install java-openjdk11 -y`
- `sudo yum install jenkins -y`
- `sudo systemctl enable Jenkins`
- `sudo systemctl start Jenkins`

Step-3:- Connect to http://<your_server_public_DNS>:8080 from your browser. You will be able to access Jenkins through its management interface

Step-4:- Enter the password found in **/var/lib/jenkins/secrets/initialAdminPassword**.

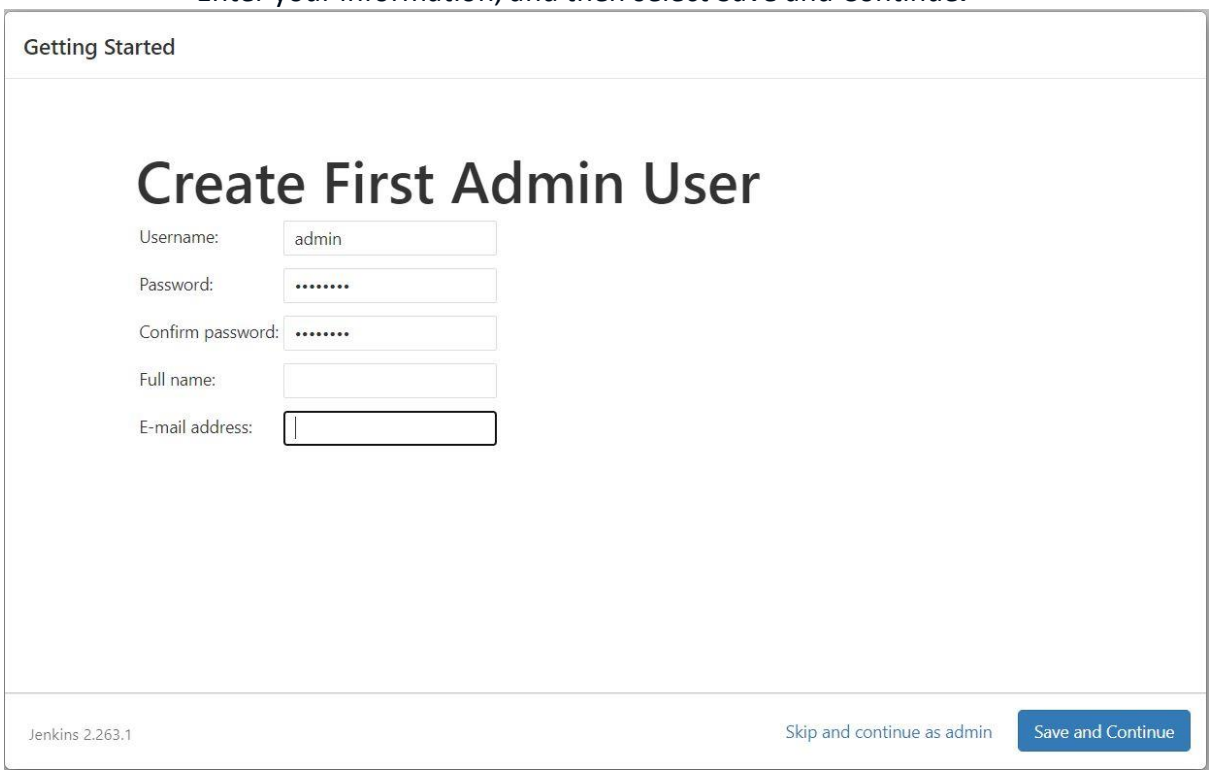
Use the following command to display this password:

$ sudo cat /var/lib/Jenkins/secrets/InitialAdminPassword

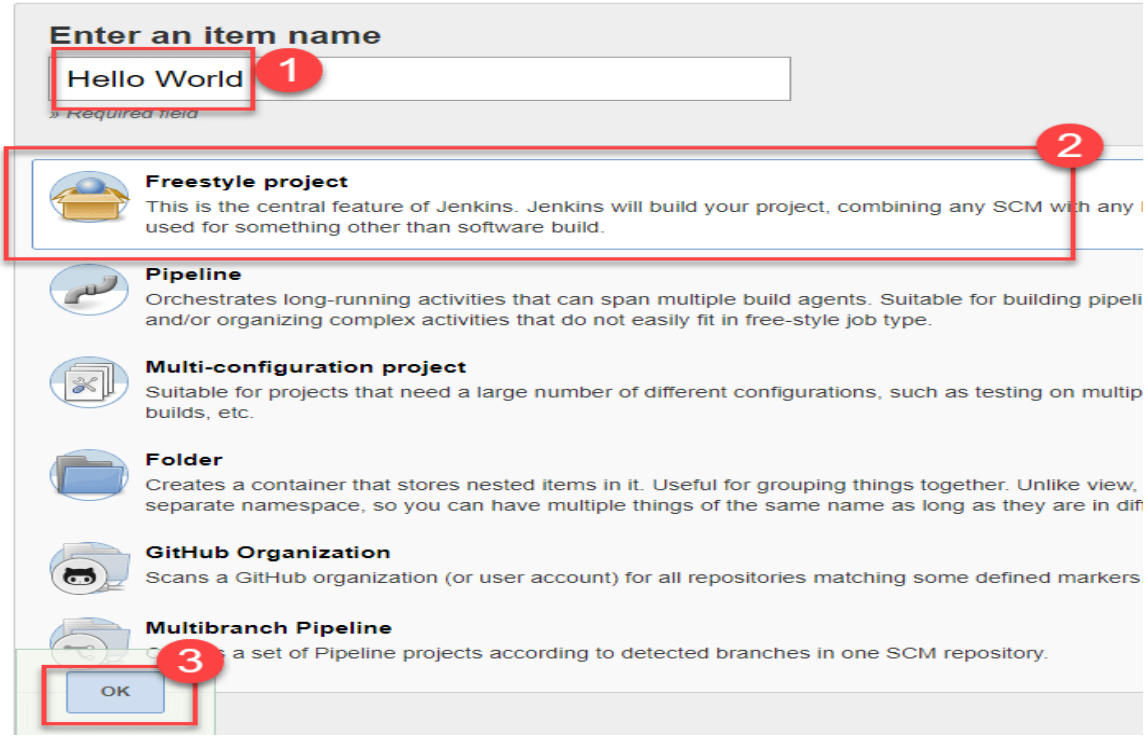


Step-5:- The Jenkins installation script directs you to the Customize Jenkins page. Click Install suggested plugins.

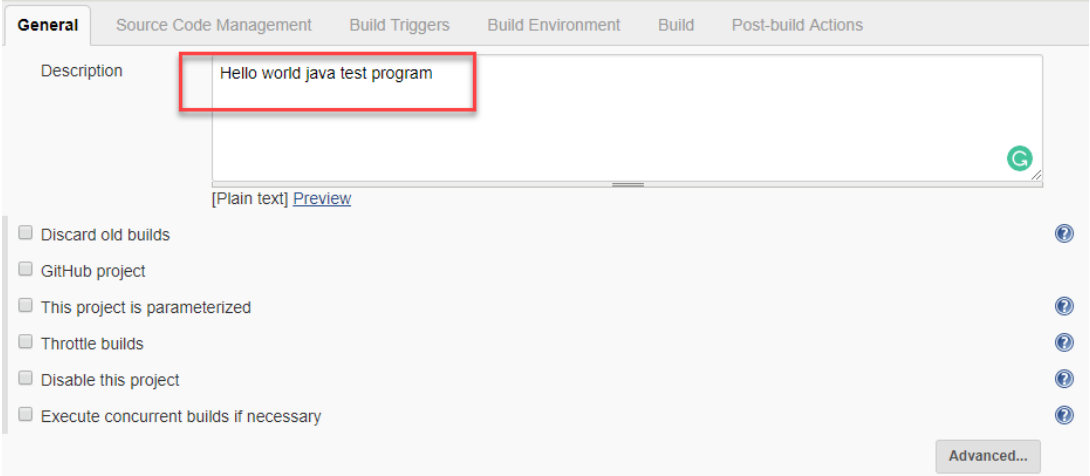
- Once the installation is complete, the Create First Admin User will open. Enter your information, and then select Save and Continue.



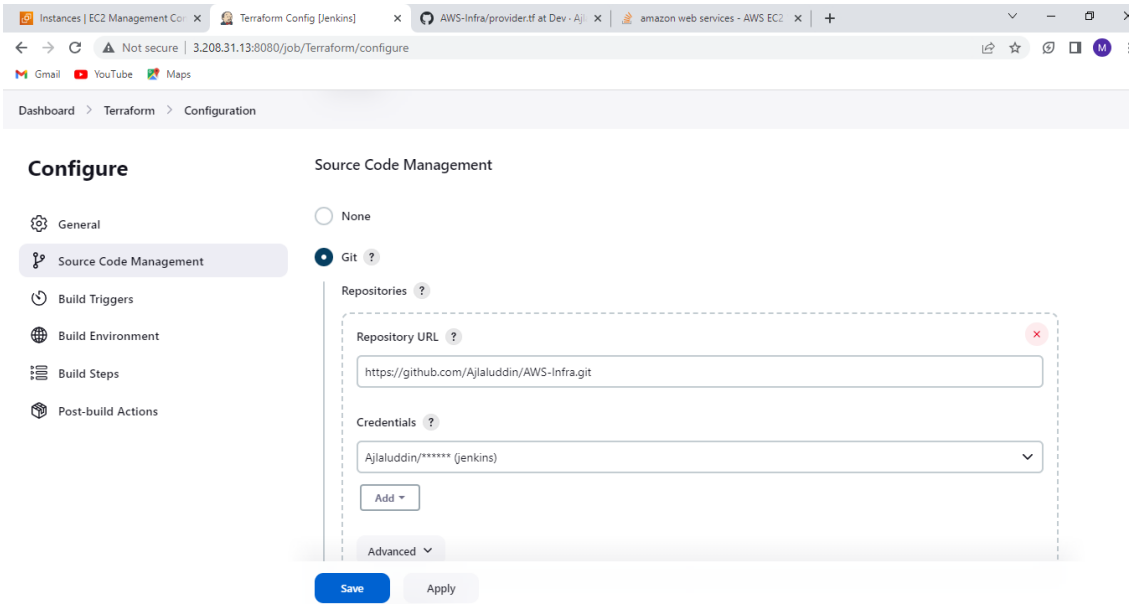
Step-6:- Once the Jenkins profile setup is completed create a new job by clicking on new item. Select free style project.



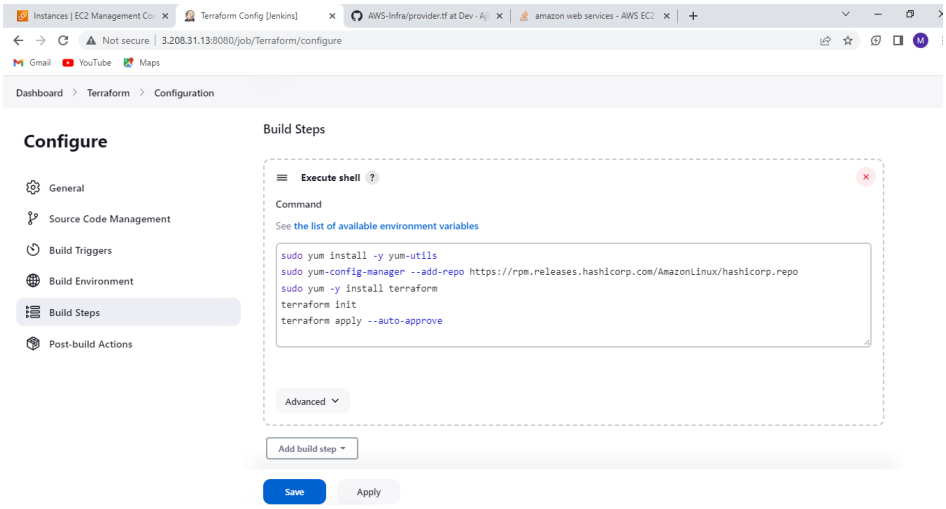
Step-7:- Enter the details of the project you want to test.



Step-8:- Under Source Code Management, Enter your repository URL.



Step-9:- Under the build section select “Add build step” and click on “Execute Shell” and add the commands which you want to execute during the build process.

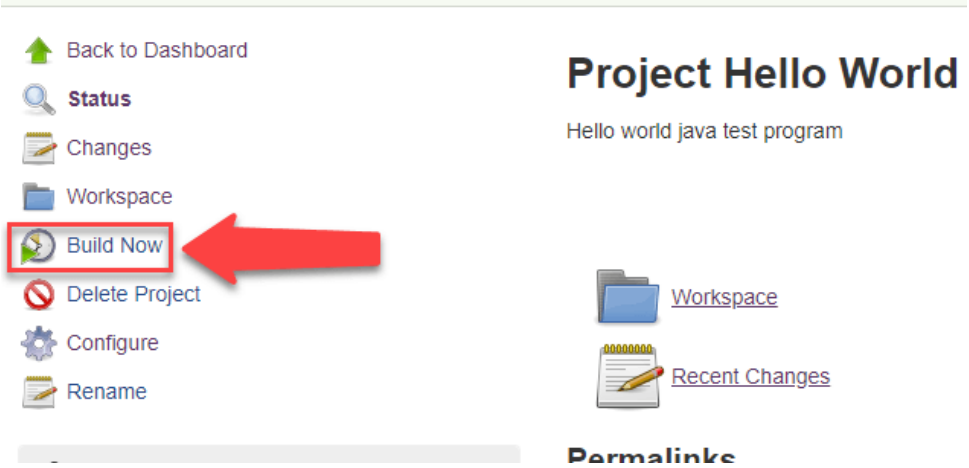


Step-10:- Click apply and save the project.

Step-11:- Build source code.

Now, in the main screen, Click the **Build Now** button on the left-hand side to build the source code.

After clicking on **Build now,** you can see the status of the build you run under **Build History**.



Step-12:- Click on **console output** to see the status of the build you run.