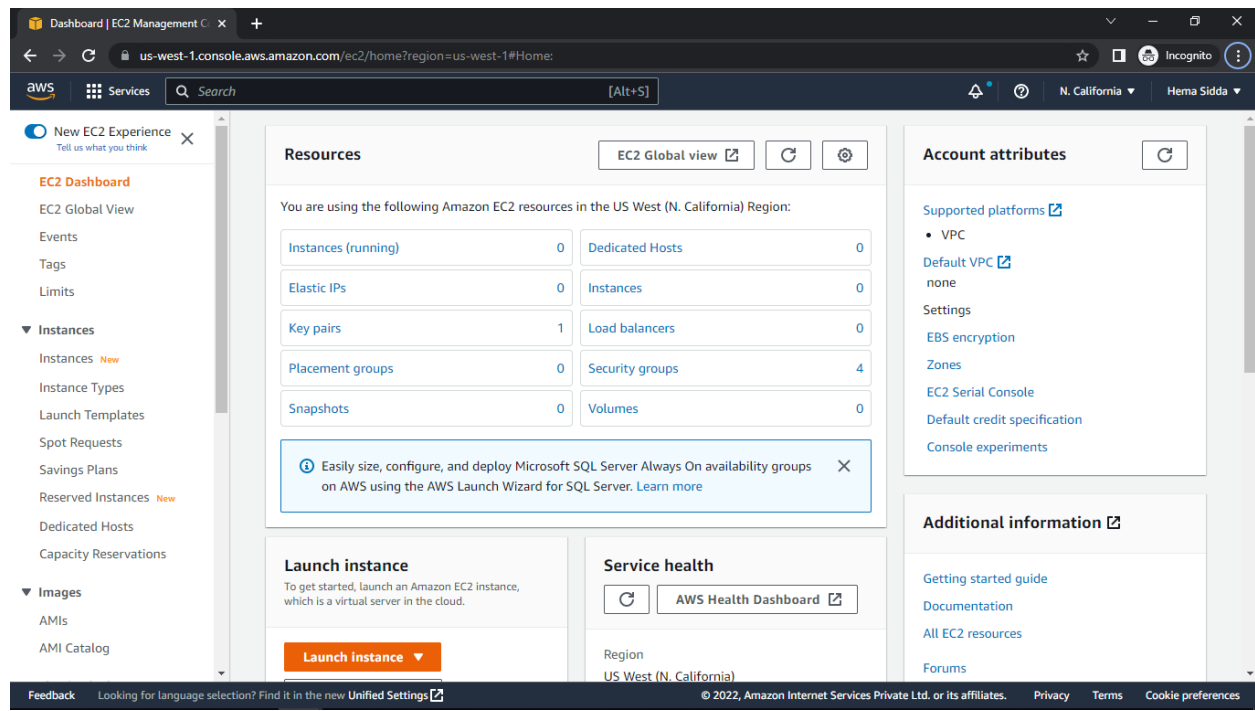


DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

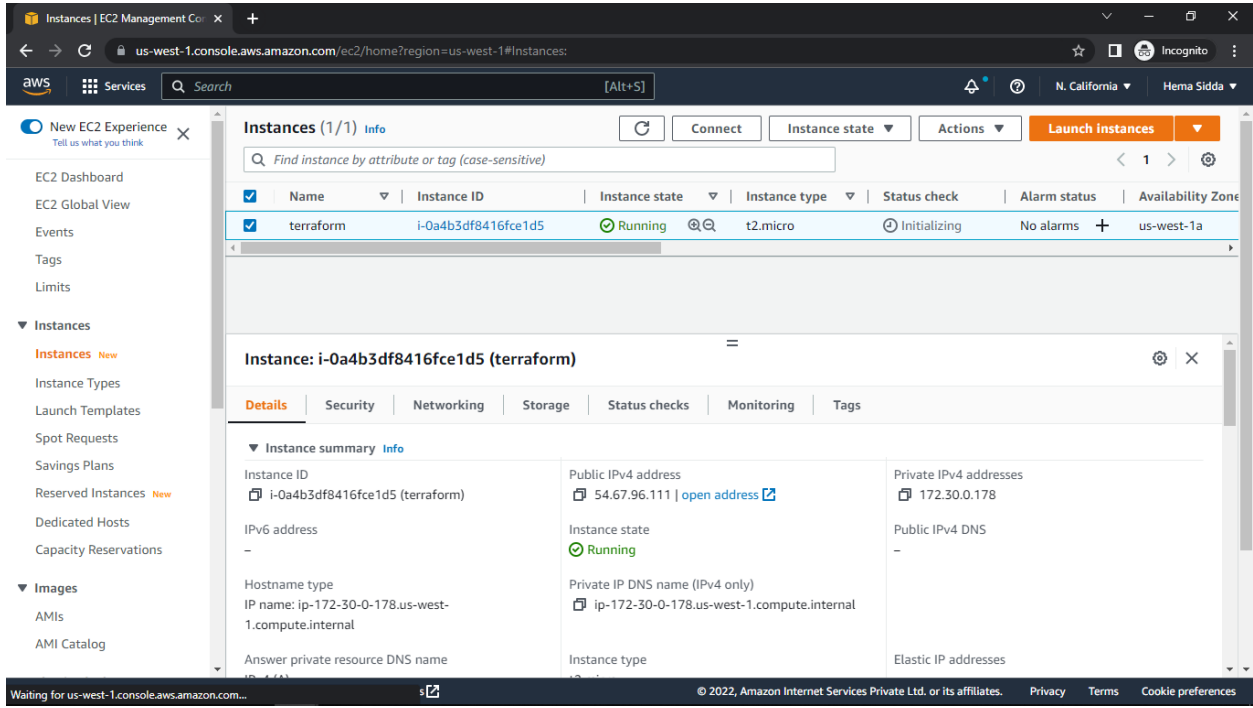
Method-1

- First open the AWS account by using credentials and go with EC2 instances.



- Now create or launch the server by selecting AMI's and key-pairs and also give the port ranges like SSH(22) and HTTP(80).

USING TERRAFORM



➤ Now connect the launched server through GIT-Bash terminal.

```
ec2-user@ip-172-30-0-178:~  
acer@DESKTOP-ONJL693 MINGW64 ~  
$ cd Downloads  
acer@DESKTOP-ONJL693 MINGW64 ~/Downloads  
$ ssh -i "terraform.pem" ec2-user@54.67.96.111  
The authenticity of host '54.67.96.111 (54.67.96.111)' can't be established.  
ED25519 key fingerprint is SHA256:0bvB2/AckDV74E4SwUaOceptA/Ry8vI1zGlbj9TIOxI.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '54.67.96.111' (ED25519) to the list of known hosts.  
  
  _ |  _ |  _ )  
 _ |  _ |  _ /  
 _ |  _ |  _ |  
      Amazon Linux 2 AMI  
  
https://aws.amazon.com/amazon-linux-2/  
1 package(s) needed for security, out of 1 available  
Run "sudo yum update" to apply all updates.  
[ec2-user@ip-172-30-0-178 ~]$
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY

USING TERRAFORM

- Now install the terraform by using commands those are available in Google browse it by the name <Terraform download>.

```
ec2-user@ip-172-30-0-178:~  
=====
```

Package	Arch	Version	Repository	Size
Installing: terraform	x86_64	1.3.5-1	hashicorp	13 M

```
=====
```

Transaction Summary

```
=====
```

Install 1 Package

Total download size: 13 M
Installed size: 58 M
Downloading packages:
warning: /var/cache/yum/x86_64/2/hashicorp/packages/terraform-1.3.5-1.x86_64.rpm: Header V4 RSA/SHA512 Signature, key ID a3219f7b: NOKEY
Public key for terraform-1.3.5-1.x86_64.rpm is not installed
terraform-1.3.5-1.x86_64.rpm | 13 MB 00:00:00
Retrieving key from https://rpm.releases.hashicorp.com/gpg
Importing GPG key 0xA3219F7B:
 Userid : "HashiCorp Security (HashiCorp Package Signing) <security+packaging@hashicorp.com>"
 Fingerprint: e8a0 32e0 94d8 eb4e a189 d270 da41 8c88 a321 9f7b
 From : https://rpm.releases.hashicorp.com/gpg
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
 Installing : terraform-1.3.5-1.x86_64 1/1
 Verifying : terraform-1.3.5-1.x86_64 1/1
Installed:
 terraform.x86_64 0:1.3.5-1
Complete!
[ec2-user@ip-172-30-0-178 ~]\$ ls
[ec2-user@ip-172-30-0-178 ~]\$ ls

- Now create vpc by using yaml scripting for that create a file as <vi file.tf> and write the yaml script to create vpc and save it by using “:wq”.

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~$
provider "aws" {
  region      = "us-west-1"
  access_key  = "AKIA352V5INCBUYKAMX2"
  secret_key  = "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"
}
#Create the vpc
resource "aws_vpc" "myvpc" {
  cidr_block      = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "myvpc"
  }
}

-- INSERT --
```

- Now change the environment into terraform mode by using command as <terraform init>.

```
ec2-user@ip-172-30-0-178:~$
Running transaction
  Installing : terraform-1.3.5-1.x86_64
  Verifying   : terraform-1.3.5-1.x86_64

Installed:
  terraform.x86_64 0:1.3.5-1

Complete!
[ec2-user@ip-172-30-0-178 ~]$ ls
[ec2-user@ip-172-30-0-178 ~]$ ls
[ec2-user@ip-172-30-0-178 ~]$ vi vpc.tf
[ec2-user@ip-172-30-0-178 ~]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.41.0...
- Installed hashicorp/aws v4.41.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
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-172-30-0-178 ~]$
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY

USING TERRAFORM

- Now set the yaml script alignment by using command as <terraform fmt>.

```
ec2-user@ip-172-30-0-178:~$ cat vpc.tf
provider "aws" {
  region      = "us-west-1"
  access_key  = "AKIA352V5INCBUYKAMX2"
  secret_key  = "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"
}
#Create the vpc
resource "aws_vpc" "myvpc" {
  cidr_block     = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "myvpc"
  }
}

[ec2-user@ip-172-30-0-178 ~]$ terraform fmt
vpc.tf
[ec2-user@ip-172-30-0-178 ~]$ cat vpc.tf
provider "aws" {
  region      = "us-west-1"
  access_key  = "AKIA352V5INCBUYKAMX2"
  secret_key  = "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"
}
#Create the vpc
resource "aws_vpc" "myvpc" {
  cidr_block     = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "myvpc"
  }
}

[ec2-user@ip-172-30-0-178 ~]$
```

- Now validate the script for correcting the spelling mistakes by using command as <terraform validate>.

```
access_key = "AKIA352V5INCBUYKAMX2"
secret_key = "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"
}
#Create the vpc
resource "aws_vpc" "myvpc" {
  cidr_block     = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "myvpc"
  }
}

[ec2-user@ip-172-30-0-178 ~]$ terraform fmt
vpc.tf
[ec2-user@ip-172-30-0-178 ~]$ cat vpc.tf
provider "aws" {
  region      = "us-west-1"
  access_key  = "AKIA352V5INCBUYKAMX2"
  secret_key  = "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"
}
#Create the vpc
resource "aws_vpc" "myvpc" {
  cidr_block     = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "myvpc"
  }
}

[ec2-user@ip-172-30-0-178 ~]$ terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-30-0-178 ~]$
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY

USING TERRAFORM

➤ Now create the vpc by using command as <terraform apply>.

```
ec2-user@ip-172-30-0-178:~$ terraform apply
# aws_vpc.mvpc will be created
+ resource "aws_vpc" "mvpc" {
+   arn                               = (known after apply)
+   cidr_block                        = "10.0.0.0/16"
+   default_network_acl_id           = (known after apply)
+   default_route_table_id           = (known after apply)
+   default_security_group_id        = (known after apply)
+   dhcp_options_id                  = (known after apply)
+   enable_classiclink                = (known after apply)
+   enable_classiclink_dns_support    = (known after apply)
+   enable_dns_hostnames              = (known after apply)
+   enable_dns_support                = true
+   enable_network_address_usage_metrics = (known after apply)
+   id                               = (known after apply)
+   instance_tenancy                  = "default"
+   ipv6_association_id              = (known after apply)
+   ipv6_cidr_block                   = (known after apply)
+   ipv6_cidr_block_network_border_group = (known after apply)
+   main_route_table_id              = (known after apply)
+   owner_id                         = (known after apply)
+   tags                             = {
+     "Name" = "myvpc"
+   }
+   tags_all                         = {
+     "Name" = "myvpc"
+   }
+ }

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: |
```

```
ec2-user@ip-172-30-0-178:~$ terraform apply
+ default_security_group_id          = (known after apply)
+ dhcp_options_id                    = (known after apply)
+ enable_classiclink                 = (known after apply)
+ enable_classiclink_dns_support      = (known after apply)
+ enable_dns_hostnames                = (known after apply)
+ enable_dns_support                  = true
+ enable_network_address_usage_metrics = (known after apply)
+ id                                 = (known after apply)
+ instance_tenancy                    = "default"
+ ipv6_association_id                = (known after apply)
+ ipv6_cidr_block                     = (known after apply)
+ ipv6_cidr_block_network_border_group = (known after apply)
+ main_route_table_id                = (known after apply)
+ owner_id                           = (known after apply)
+ tags                               = {
+   "Name" = "myvpc"
+ }
+ tags_all                           = {
+   "Name" = "myvpc"
+ }
+ }

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.

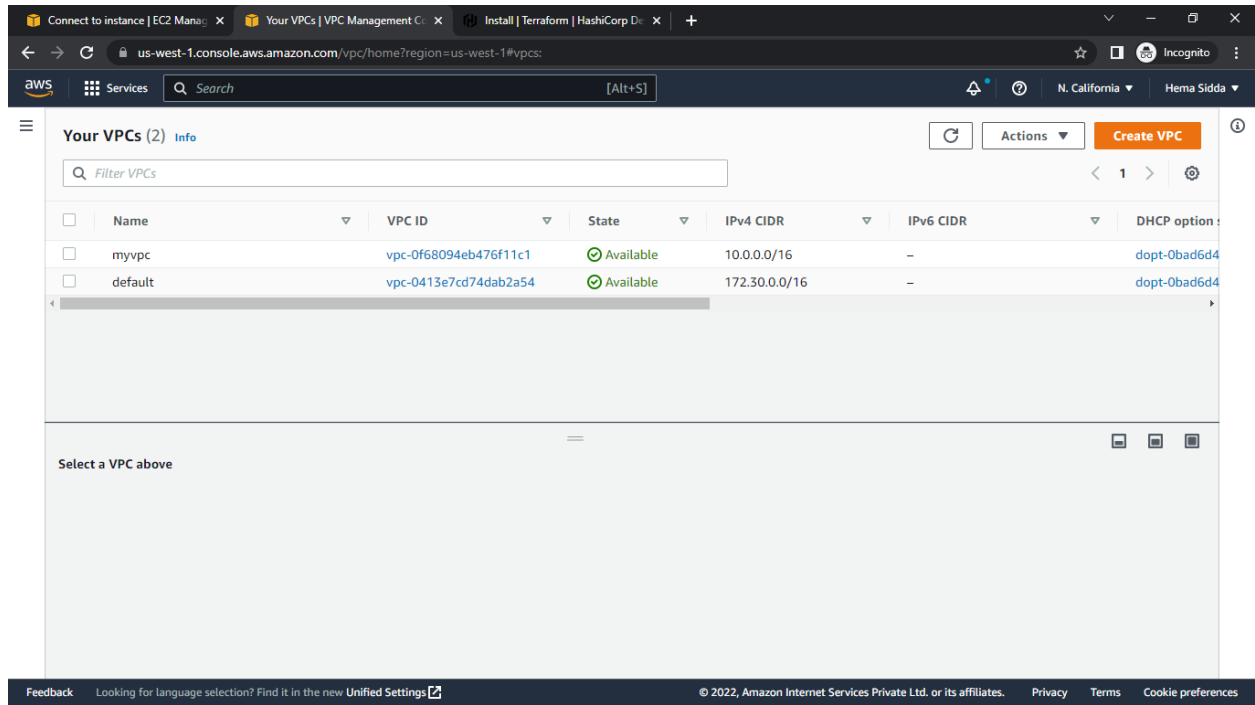
Enter a value: yes

aws_vpc.mvpc: Creating...
aws_vpc.mvpc: Creation complete after 0s [id=vpc-0f68094eb476f11c1]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

- Now go to your AWS account and open vpc services and check your vpc created or not.



- Now create the IGW and attach it to created vpc for that create a file as <vi igw.tf>

USING TERRAFORM

```

ec2-user@ip-172-30-0-178:~
#Creating IGW
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.mvpc.id
  tags = {

    Name = "my-igw"
  }
}

-- INSERT --
9,1 A11

```

```

ec2-user@ip-172-30-0-178:~$ terraform apply --auto-approve
aws_vpc.mvpc: Creating...
aws_vpc.mvpc: Creation complete after 0s [id=vpc-0f68094eb476f11c1]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$ vi igw.tf
[ec2-user@ip-172-30-0-178 ~]$ terraform apply --auto-approve
aws_vpc.mvpc: Refreshing state... [id=vpc-0f68094eb476f11c1]

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

Terraform will perform the following actions:

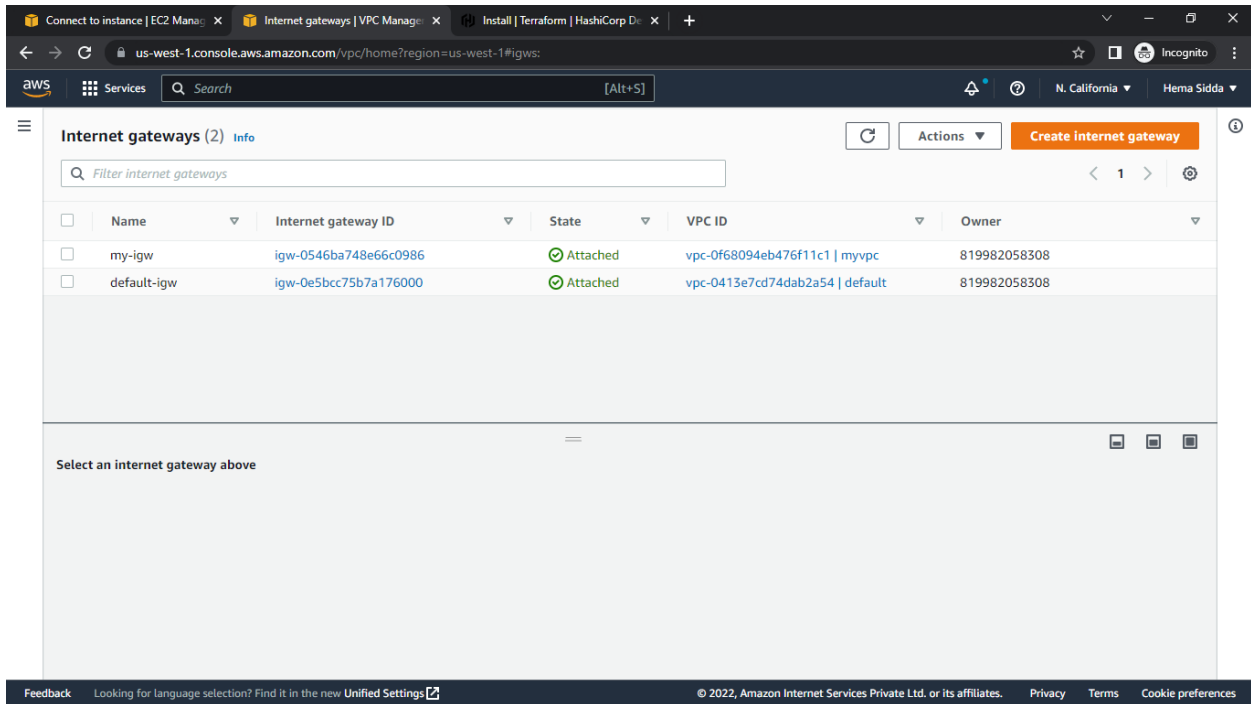
# aws_internet_gateway.igw will be created
+ resource "aws_internet_gateway" "igw" {
  + arn      = (known after apply)
  + id       = (known after apply)
  + owner_id = (known after apply)
  + tags     = {
    + "Name" = "my-igw"
  }
  + tags_all = {
    + "Name" = "my-igw"
  }
  + vpc_id   = "vpc-0f68094eb476f11c1"
}

Plan: 1 to add, 0 to change, 0 to destroy.
aws_internet_gateway.igw: Creating...
aws_internet_gateway.igw: Creation complete after 0s [id=igw-0546ba748e66c0986]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$ |

```


DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM



- Now create the subnets by using yaml script for that use this command <vi subnets.tf>

```
ec2-user@ip-172-30-0-178:~$
#Creating pub-subnet1
resource "aws_subnet" "pub-sub-1" {
  vpc_id            = aws_vpc.mvpc.id
  cidr_block        = "10.0.1.0/24"
  availability_zone  = "us-west-1a"
  map_public_ip_on_launch = "true"
  tags = {
    Name = "pub-subnet-1"
  }
}
#Creating pvt-subnet1
resource "aws_subnet" "pvt-sub-1" {
  vpc_id            = aws_vpc.mvpc.id
  cidr_block        = "10.0.2.0/24"
  availability_zone  = "us-west-1a"
  map_public_ip_on_launch = "false"
  tags = {
    Name = "pvt-subnet-1"
  }
}
#Creating pub-subnet2
resource "aws_subnet" "pub-sub-2" {
  vpc_id            = aws_vpc.mvpc.id
  cidr_block        = "10.0.3.0/24"
  availability_zone  = "us-west-1b"
  map_public_ip_on_launch = "true"
  tags = {
    Name = "pub-subnet-2"
  }
}
-- INSERT --
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY

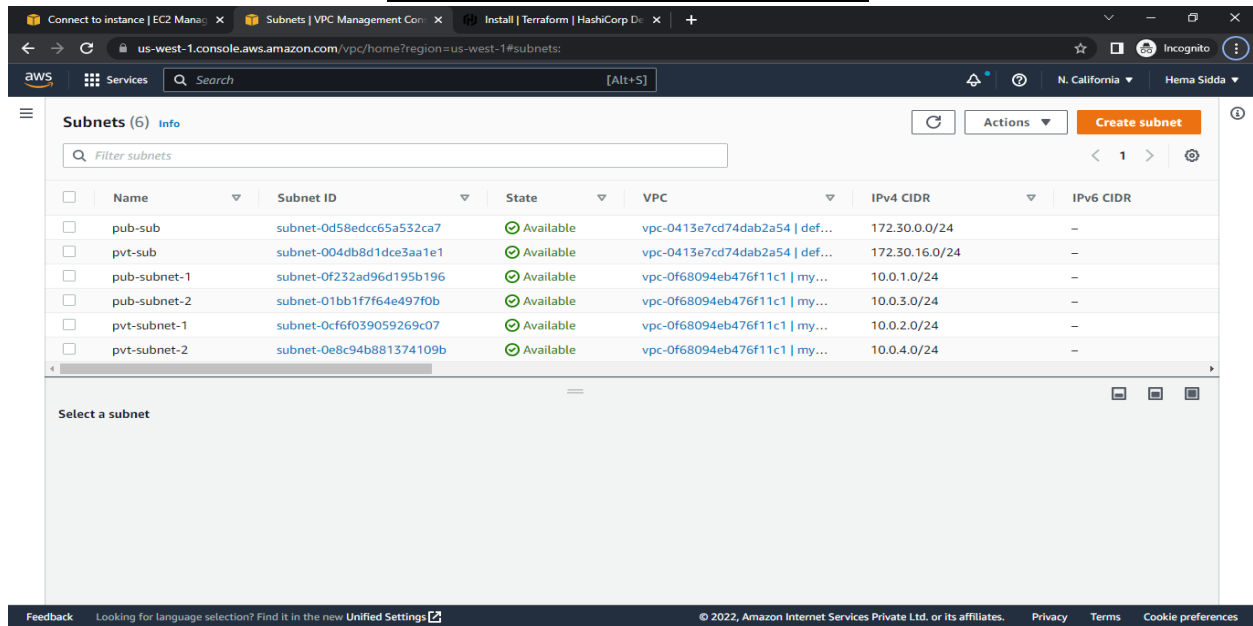
USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~  
cidr_block           = "10.0.2.0/24"  
availability_zone     = "us-west-1a"  
map_public_ip_on_launch = "false"  
  
tags = {  
  Name = "pvt-subnet-1"  
}  
}  
  
#Creating pub-subnet2  
resource "aws_subnet" "pub-sub-2" {  
  vpc_id           = aws_vpc.mvpc.id  
  cidr_block       = "10.0.3.0/24"  
  availability_zone = "us-west-1b"  
  map_public_ip_on_launch = "true"  
  tags = {  
    Name = "pub-subnet-2"  
  }  
}  
  
#Creating pvt-subnet2  
resource "aws_subnet" "pvt-sub-2" {  
  vpc_id           = aws_vpc.mvpc.id  
  cidr_block       = "10.0.4.0/24"  
  availability_zone = "us-west-1b"  
  map_public_ip_on_launch = "false"  
  tags = {  
    Name = "pvt-subnet-2"  
  }  
}  
  
-- INSERT --  
49,1 Bot
```

- Now by using single command I can create the resource that command is <terraform apply --auto-approve>

```
ec2-user@ip-172-30-0-178:~  
+ availability_zone           = "us-west-1b"  
+ 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)  
+ tags                       = {  
  + "Name" = "pvt-subnet-2"  
}  
+ tags_all                   = {  
  + "Name" = "pvt-subnet-2"  
}  
+ vpc_id                     = "vpc-0f68094eb476f11c1"  
}  
  
Plan: 4 to add, 0 to change, 0 to destroy.  
aws_subnet.pub-sub-1: Creating...  
aws_subnet.pub-sub-2: Creating...  
aws_subnet.pvt-sub-1: Creating...  
aws_subnet.pvt-sub-2: Creating...  
aws_subnet.pvt-sub-1: Creation complete after 0s [id=subnet-0cf6f039059269c07]  
aws_subnet.pvt-sub-2: Creation complete after 0s [id=subnet-0e8c94b881374109b]  
aws_subnet.pub-sub-1: Still creating... [10s elapsed]  
aws_subnet.pub-sub-2: Still creating... [10s elapsed]  
aws_subnet.pub-sub-1: Creation complete after 11s [id=subnet-0f232ad96d195b196]  
aws_subnet.pub-sub-2: Creation complete after 11s [id=subnet-01bb1f7f64e497f0b]  
  
Apply complete! Resources: 4 added, 0 changed, 0 destroyed.  
[ec2-user@ip-172-30-0-178 ~]$
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM



- Now create the NAT-gateway for that create a .tf file and enter the yaml script.

```
ec2-user@ip-172-30-0-178:~$
# Create elastic ip
resource "aws_eip" "elastic" {
  vpc = true
}

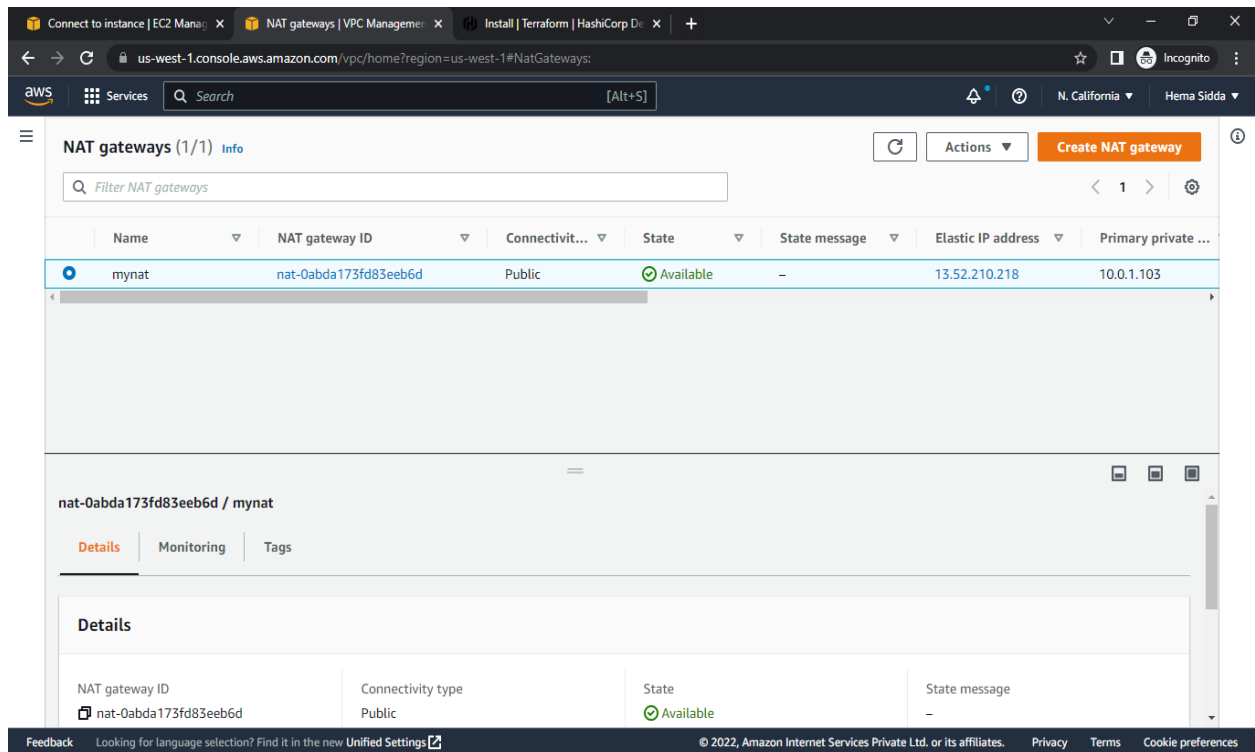
# Create Nat-gateway
resource "aws_nat_gateway" "nat" {
  allocation_id    = aws_eip.elastic.id
  subnet_id        = aws_subnet.pub-sub-1.id
  connectivity_type = "public"
  tags = {
    Name = "mynat"
  }
}
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~$ terraform apply
+ id = (known after apply)
+ network_interface_id = (known after apply)
+ private_ip = (known after apply)
+ public_ip = (known after apply)
+ subnet_id = "subnet-0f232ad96d195b196"
+ tags = {
  + "Name" = "mynat"
}
+ tags_all = {
  + "Name" = "mynat"
}

Plan: 2 to add, 0 to change, 0 to destroy.
aws_eip.elastic: Creating...
aws_eip.elastic: Creation complete after 0s [id=eipalloc-034630da659b77654]
aws_nat_gateway.nat: Creating...
aws_nat_gateway.nat: Still creating... [10s elapsed]
aws_nat_gateway.nat: Still creating... [20s elapsed]
aws_nat_gateway.nat: Still creating... [30s elapsed]
aws_nat_gateway.nat: Still creating... [40s elapsed]
aws_nat_gateway.nat: Still creating... [50s elapsed]
aws_nat_gateway.nat: Still creating... [1m0s elapsed]
aws_nat_gateway.nat: Still creating... [1m10s elapsed]
aws_nat_gateway.nat: Still creating... [1m20s elapsed]
aws_nat_gateway.nat: Still creating... [1m30s elapsed]
aws_nat_gateway.nat: Still creating... [1m40s elapsed]
aws_nat_gateway.nat: Still creating... [1m50s elapsed]
aws_nat_gateway.nat: Still creating... [2m0s elapsed]
aws_nat_gateway.nat: Still creating... [2m10s elapsed]
aws_nat_gateway.nat: Still creating... [2m20s elapsed]
aws_nat_gateway.nat: Creation complete after 2m25s [id=nat-0abda173fd83eeb6d]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$
```



DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

- Now go to create the route table and associate subnets, Internet gateway and NAT gate way for that create a .tf file by using the vi mode as <vi route.tf>

```
ec2-user@ip-172-30-0-178:~$ vi route.tf
#Create Pub-Route-table
resource "aws_route_table" "pub-route" {
  vpc_id = aws_vpc.mvpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }
  tags = {
    Name = "pub-rot"
  }
}

#Create Pvt-Route-table
resource "aws_route_table" "pvt-route" {
  vpc_id = aws_vpc.mvpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_nat_gateway.nat.id
  }
  tags = {
    Name = "pvt-rot"
  }
}

#subnets Associations
#pub-subnet-association
resource "aws_route_table_association" "a" {
  subnet_id      = aws_subnet.pub-sub-1.id
  route_table_id = aws_route_table.pub-route.id
}

-- INSERT --
```

```
ec2-user@ip-172-30-0-178:~$ vi route.tf
}

}

#subnets Associations
#pub-subnet-association
resource "aws_route_table_association" "a" {
  subnet_id      = aws_subnet.pub-sub-1.id
  route_table_id = aws_route_table.pub-route.id
}

#pvt subnet-association
resource "aws_route_table_association" "b" {
  subnet_id      = aws_subnet.pvt-sub-1.id
  route_table_id = aws_route_table.pvt-route.id
}

#subnets Associations
#pub-subnet-association
resource "aws_route_table_association" "c" {
  subnet_id      = aws_subnet.pub-sub-2.id
  route_table_id = aws_route_table.pub-route.id
}

#pvt subnet-association
resource "aws_route_table_association" "d" {
  subnet_id      = aws_subnet.pvt-sub-2.id
  route_table_id = aws_route_table.pvt-route.id
}

-- INSERT --
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

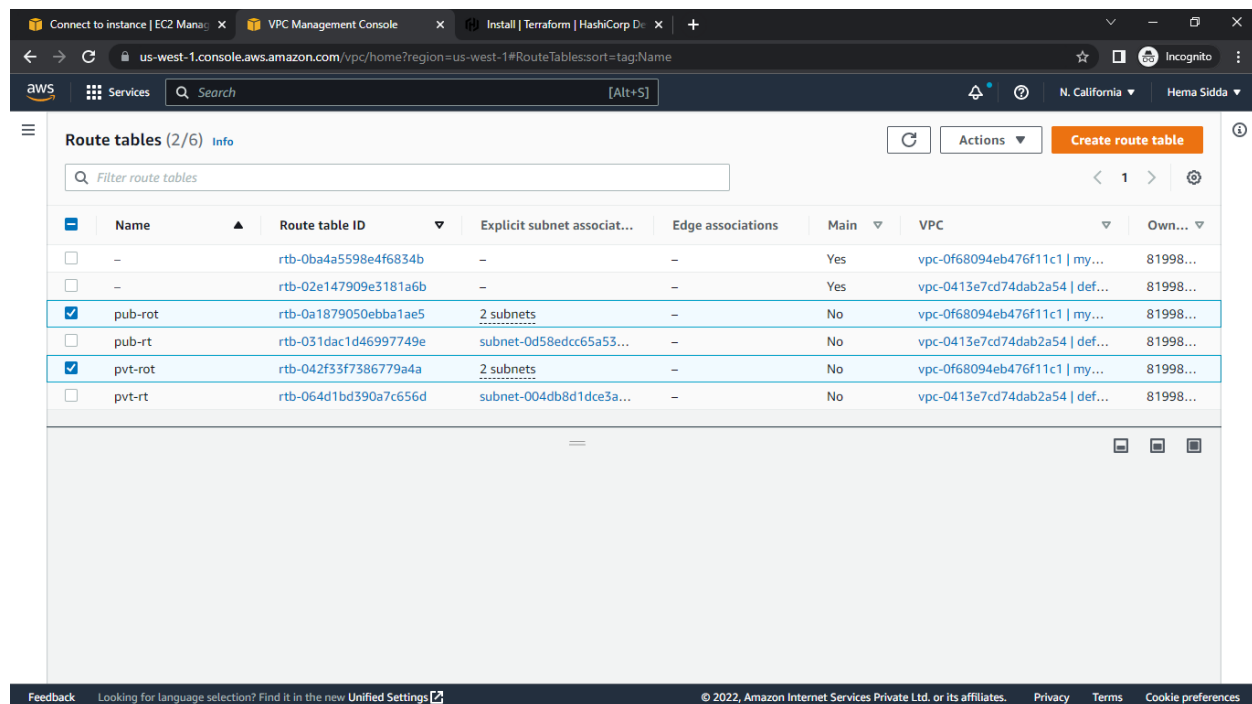
```
ec2-user@ip-172-30-0-178:~$ terraform apply
+ id = (known after apply)
+ route_table_id = (known after apply)
+ subnet_id = "subnet-0cf6f039059269c07"
}

# aws_route_table_association.c will be created
+ resource "aws_route_table_association" "c" {
+ id = (known after apply)
+ route_table_id = (known after apply)
+ subnet_id = "subnet-01bb1f7f64e497f0b"
}

# aws_route_table_association.d will be created
+ resource "aws_route_table_association" "d" {
+ id = (known after apply)
+ route_table_id = (known after apply)
+ subnet_id = "subnet-0e8c94b881374109b"
}

Plan: 6 to add, 0 to change, 0 to destroy.
aws_route_table.pub-route: Creating...
aws_route_table.pvt-route: Creating...
aws_route_table.pvt-route: Creation complete after 0s [id=rtb-042f33f7386779a4a]
aws_route_table_association.d: Creating...
aws_route_table_association.b: Creating...
aws_route_table.pub-route: Creation complete after 0s [id=rtb-0a1879050ebb1ae5]
aws_route_table_association.c: Creating...
aws_route_table_association.a: Creating...
aws_route_table_association.b: Creation complete after 0s [id=rtbassoc-0579536724ad2da7a]
aws_route_table_association.a: Creation complete after 1s [id=rtbassoc-00aeb0e5d8f6862]
aws_route_table_association.d: Creation complete after 1s [id=rtbassoc-0533d5fd5d09e2a50]
aws_route_table_association.c: Creation complete after 1s [id=rtbassoc-0222db84d94adb75c]

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$
```



The screenshot shows the AWS VPC Management Console for the us-west-1 region. The 'Route tables (2/6)' page is displayed, showing a list of route tables. The 'pub-rot' and 'pvt-rot' tables are selected, indicated by blue checkmarks in the selection column.

	Name	Route table ID	Explicit subnet associat...	Edge associations	Main	VPC	Own...
<input type="checkbox"/>	-	rtb-0ba4a5598e4f6834b	-	-	Yes	vpc-0f68094eb476f11c1 my...	81998...
<input type="checkbox"/>	-	rtb-02e147909e3181a6b	-	-	Yes	vpc-0413e7cd74dab2a54 def...	81998...
<input checked="" type="checkbox"/>	pub-rot	rtb-0a1879050ebb1ae5	2 subnets	-	No	vpc-0f68094eb476f11c1 my...	81998...
<input type="checkbox"/>	pub-rt	rtb-031dac1d46997749e	subnet-0d58edcc65a53...	-	No	vpc-0413e7cd74dab2a54 def...	81998...
<input checked="" type="checkbox"/>	pvt-rot	rtb-042f33f7386779a4a	2 subnets	-	No	vpc-0f68094eb476f11c1 my...	81998...
<input type="checkbox"/>	pvt-rt	rtb-064d1bd390a7c656d	subnet-004db8d1dce3a...	-	No	vpc-0413e7cd74dab2a54 def...	81998...

➤ Now create the bash script file by the data.sh

DEPLOY THREE-TIER ARCHITECHTURE IN AWS BY USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~$ # Create bash script or user-data for instance
#!/bin/bash
sudo yum -y update
sudo yum -y install httpd
sudo systemctl start httpd
sudo systemctl enable httpd
echo "hello world from $(hostname -f)" > index.html
sudo mv index.html /var/www/html/
```

8,33 A11

- Now create the public instance by using yaml scripting.

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY

USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~$
# Create instance
resource "aws_instance" "app" {
  ami           = "ami-0f5e8a042c8bfcd5e"
  instance_type = "t2.micro"
  count         = 1
  key_name      = "terraform"
  subnet_id    = aws_subnet.pub-sub-1.id
  vpc_security_group_ids = ["${aws_security_group.sg.id}"]
  user_data    = "${file("data.sh")}"
  tags = {
    Name = "webapplication"
  }
}

# Create Security-group
resource "aws_security_group" "sg" {
  vpc_id = aws_vpc.mvpc.id

  #Inbound Rules
  # HTTP access from anywhere
  ingress {
    from_port = 80
    to_port   = 80
    protocol  = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  # HTTPS access from any where
  ingress {
    from_port = 443
    to_port   = 443
    protocol  = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

-- INSERT --
```

```
ec2-user@ip-172-30-0-178:~$
  from_port = 80
  to_port   = 80
  protocol  = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}
# HTTPS access from any where
ingress {
  from_port = 443
  to_port   = 443
  protocol  = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}

# Ssh access from any where
ingress {
  from_port = 22
  to_port   = 22
  protocol  = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}

#outbound Rules
#internet access to anywhere
egress {
  from_port = 0
  to_port   = 0
  protocol  = "-1"
  cidr_blocks = ["0.0.0.0/0"]
}
tags = {
  Name = "web-sg"
}
}

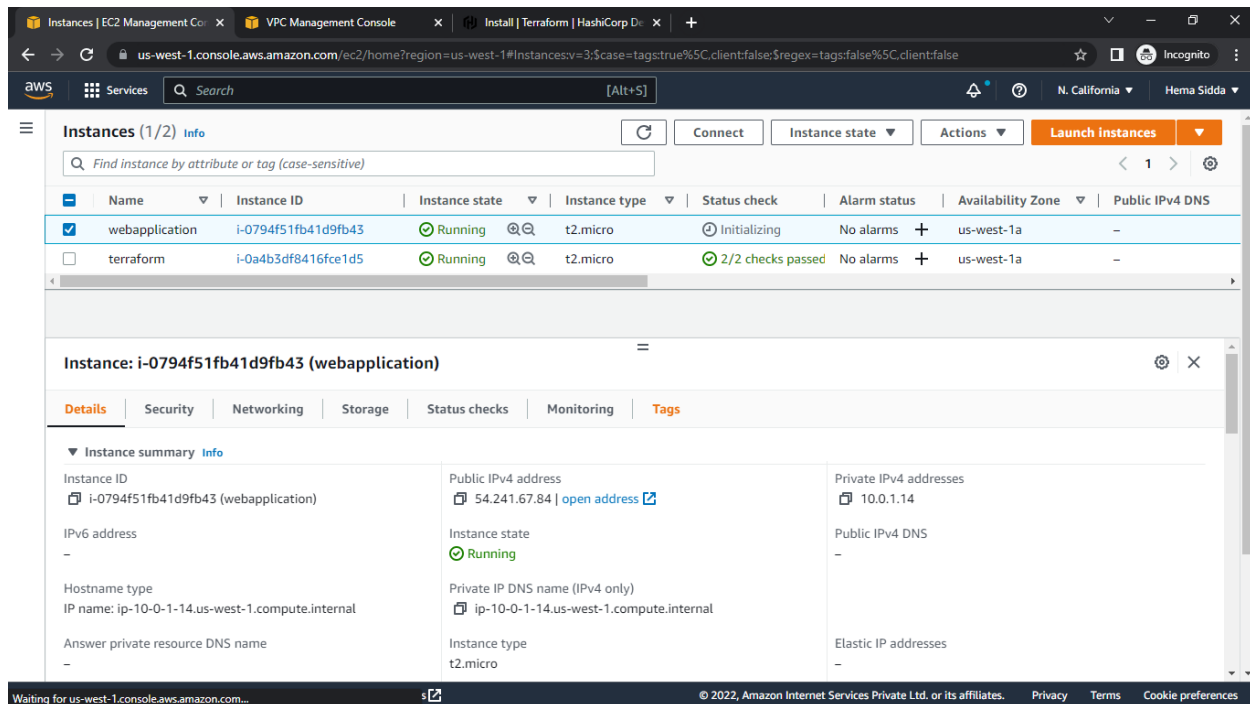
-- INSERT --
```


DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~$ terraform apply
+ 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" = "web-sg"
+ }
+ tags_all          = {
+   "Name" = "web-sg"
+ }
+ vpc_id            = "vpc-0f68094eb476f11c1"
}

Plan: 2 to add, 1 to change, 0 to destroy.
aws_security_group.sg: Creating...
aws_route_table.pvt-route: Modifying... [id=rtb-042f33f7386779a4a]
aws_route_table.pvt-route: Modifications complete after 0s [id=rtb-042f33f7386779a4a]
aws_security_group.sg: Creation complete after 1s [id=sg-04db0c061609b85bd]
aws_instance.app[0]: Creating...
aws_instance.app[0]: Still creating... [10s elapsed]
aws_instance.app[0]: Still creating... [20s elapsed]
aws_instance.app[0]: Still creating... [30s elapsed]
aws_instance.app[0]: Still creating... [40s elapsed]
aws_instance.app[0]: Creation complete after 41s [id=i-0794f51fb41d9fb43]

Apply complete! Resources: 2 added, 1 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$
```



The screenshot displays the AWS Management Console 'Instances' page. At the top, there are tabs for 'Instances (1/2)', 'Info', 'Connect', 'Instance state', 'Actions', and 'Launch instances'. Below these, a table lists the instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
webapplication	i-0794f51fb41d9fb43	Running	t2.micro	Initializing	No alarms	us-west-1a	-
terraform	i-0a4b3df8416fce1d5	Running	t2.micro	2/2 checks passed	No alarms	us-west-1a	-

Below the table, the details for the 'webapplication' instance (i-0794f51fb41d9fb43) are shown. The 'Details' tab is selected, displaying the following information:

- Instance ID:** i-0794f51fb41d9fb43 (webapplication)
- Public IPv4 address:** 54.241.67.84 | [open address](#)
- Private IPv4 addresses:** 10.0.1.14
- Instance state:** Running
- Public IPv4 DNS:** -
- IPv6 address:** -
- Instance type:** t2.micro
- Private IP DNS name (IPv4 only):** ip-10-0-1-14.us-west-1.compute.internal
- Hostname type:** IP name: ip-10-0-1-14.us-west-1.compute.internal
- Answer private resource DNS name:** -
- Elastic IP addresses:** -

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

- Now create the pvt instance by using the yaml script of pvt instance .

```
ec2-user@ip-172-30-0-178:~$
# Create instance
resource "aws_instance" "web-app" {
  ami           = "ami-0f5e8a042c8bfcd5e"
  instance_type = "t2.micro"
  count         = 1
  key_name      = "terraform"
  subnet_id    = aws_subnet.pvt-sub-1.id
  vpc_security_group_ids = ["${aws_security_group.demo.id}"]
  tags = {
    Name = "web-pvt"
  }
}

# Create Security-group
resource "aws_security_group" "demo" {
  vpc_id = aws_vpc.mvpc.id

  #Inbound Rules
  # HTTP access from anywhere
  ingress {
    from_port = 80
    to_port   = 80
    protocol  = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  # HTTPS access from any where
  ingress {
    from_port = 443
    to_port   = 443
    protocol  = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Ssh access from any where
  -- INSERT --
1,1 Top
```

```
ec2-user@ip-172-30-0-178:~$
to_port = 80
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}
# HTTPS access from any where
ingress {
  from_port = 443
  to_port   = 443
  protocol  = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}

# Ssh access from any where
ingress {
  from_port = 22
  to_port   = 22
  protocol  = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}
#outbound Rules
#internet access to anywhere
egress {
  from_port = 0
  to_port   = 0
  protocol  = "-1"
  cidr_blocks = ["0.0.0.0/0"]
}
tags = {
  Name = "web-sg"
}
}

-- INSERT --
55,1 Bot
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~$ terraform apply
+ 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" = "web-sg"
}
+ tags_all = {
+   "Name" = "web-sg"
}
+ vpc_id = "vpc-0f68094eb476f11c1"
}

Plan: 2 to add, 1 to change, 0 to destroy.
aws_security_group.demo: Creating...
aws_route_table.pvt-route: Modifying... [id=rtb-042f33f7386779a4a]
aws_route_table.pvt-route: Modifications complete after 1s [id=rtb-042f33f7386779a4a]
aws_security_group.demo: Creation complete after 2s [id=sg-051056bffa830437]
aws_instance.web-app[0]: Creating...
aws_instance.web-app[0]: Still creating... [10s elapsed]
aws_instance.web-app[0]: Still creating... [20s elapsed]
aws_instance.web-app[0]: Still creating... [30s elapsed]
aws_instance.web-app[0]: Creation complete after 31s [id=i-0b325a5b527c830bc]

Apply complete! Resources: 2 added, 1 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$
```

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
webapplication	i-0794f51fb41d9fb43	Running	t2.micro	2/2 checks passed	No alarms	us-west-1a	-
web-pvt	i-0b325a5b527c830bc	Initializing	t2.micro	1/2 checks passed	No alarms	us-west-1a	-
terraform	i-0a4b3df8416fce1d5	Running	t2.micro	2/2 checks passed	No alarms	us-west-1a	-

Instance: i-0b325a5b527c830bc (web-pvt)		
Details	Security	Networking
Instance summary Instance ID: i-0b325a5b527c830bc (web-pvt) IPv6 address: - Hostname type: IP name: ip-10-0-2-207.us-west-1.compute.internal Answer private resource DNS name: -	Public IPv4 address: - Instance state: Running Private IP DNS name (IPv4 only): ip-10-0-2-207.us-west-1.compute.internal Instance type: t2.micro	Private IPv4 addresses: 10.0.2.207 Public IPv4 DNS: - Elastic IP addresses: -

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

- Now create the RDS(relational database) service by using yaml scripting.

```
ec2-user@ip-172-30-0-178:~$
# Create subnet groups
resource "aws_db_subnet_group" "db-sub" {
  subnet_ids = [aws_subnet.pvt-sub-1.id, aws_subnet.pvt-sub-2.id]
  tags = {
    Name = "dbsubnet"
  }
}

# Create the data-base
resource "aws_db_instance" "data" {
  allocated_storage = "10"
  db_subnet_group_name = aws_db_subnet_group.db-sub.id
  db_name = "mydata"
  engine = "mysql"
  engine_version = "8.0.28"
  instance_class = "db.t2.micro"
  multi_az = "true"
  username = "admin"
  password = "admin1230"
  vpc_security_group_ids = ["${aws_security_group.db-sg.id}"]
  skip_final_snapshot = "true"
}

# Create the security groups
resource "aws_security_group" "db-sg" {
  vpc_id = aws_vpc.mvpc.id

  ingress {
    from_port = 3306
    to_port = 3306
    protocol = "tcp"
    security_groups = [aws_security_group.demo.id]
  }
}

-- INSERT --
```

```
ec2-user@ip-172-30-0-178:~$
  engine = "mysql"
  engine_version = "8.0.28"
  instance_class = "db.t2.micro"
  multi_az = "true"
  username = "admin"
  password = "admin1230"
  vpc_security_group_ids = ["${aws_security_group.db-sg.id}"]
  skip_final_snapshot = "true"
}

# Create the security groups
resource "aws_security_group" "db-sg" {
  vpc_id = aws_vpc.mvpc.id

  ingress {
    from_port = 3306
    to_port = 3306
    protocol = "tcp"
    security_groups = [aws_security_group.demo.id]
  }

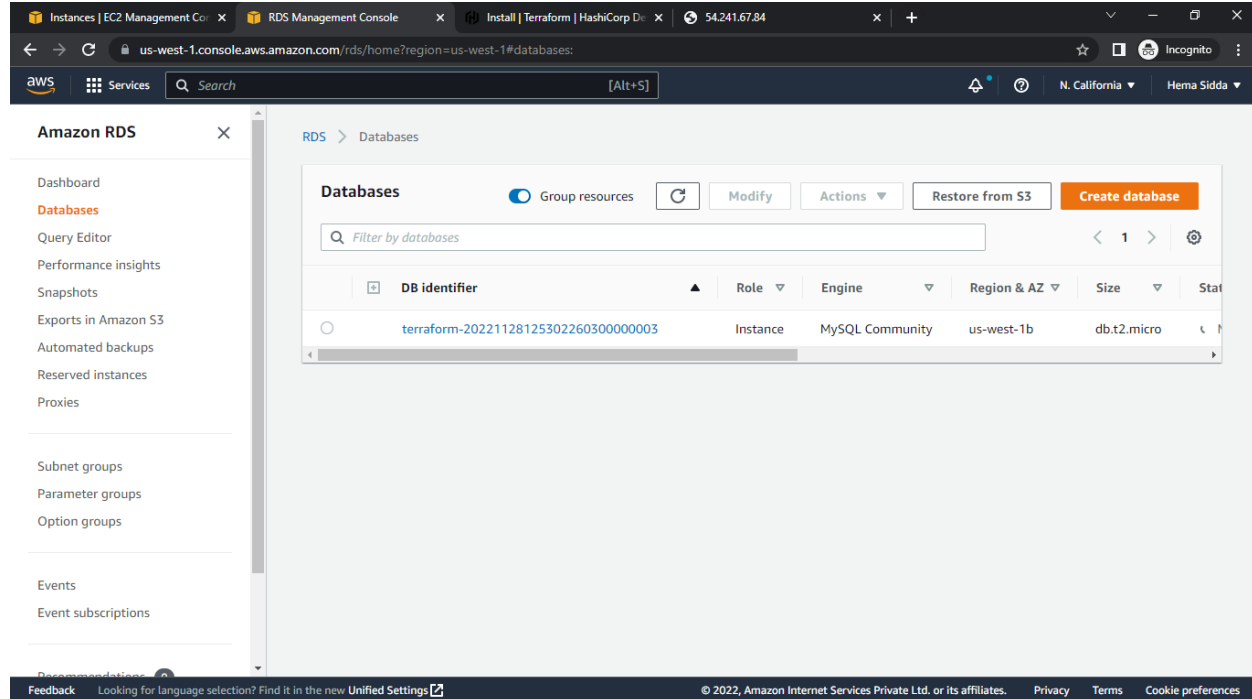
  egress {
    from_port = 32768
    to_port = 65535
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "database-sg"
  }
}

-- INSERT --
```

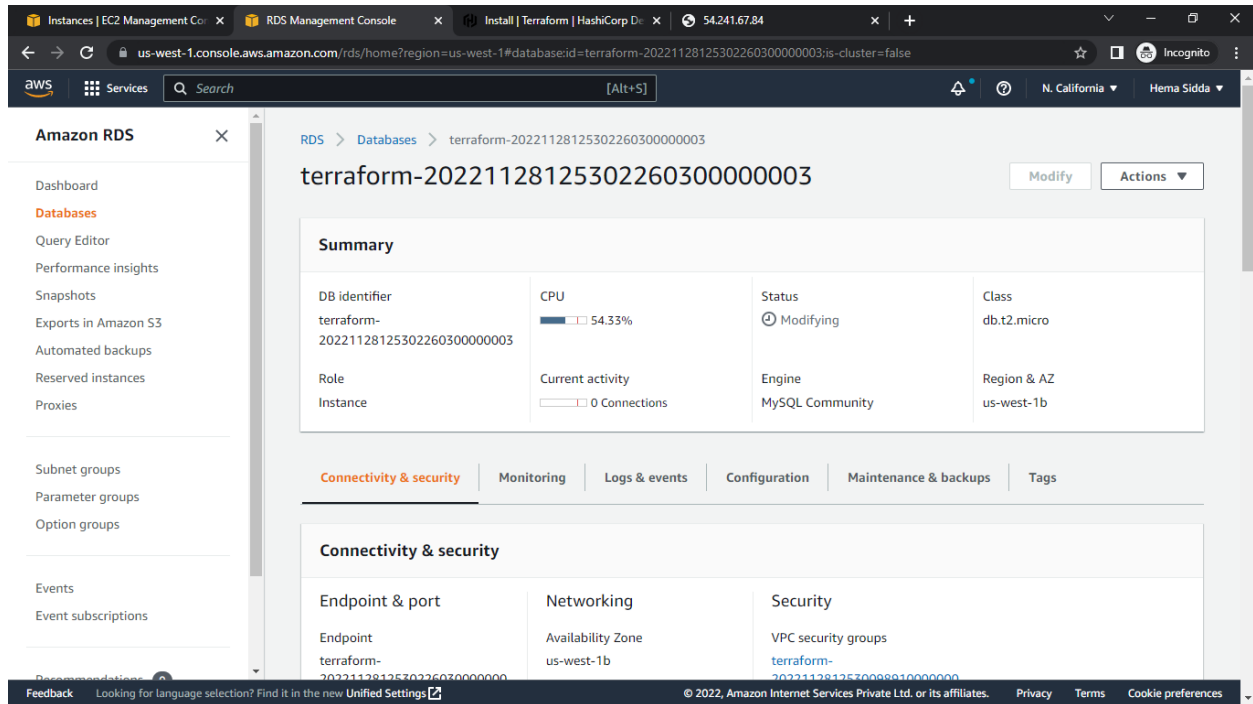
DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~$ terraform apply
aws_db_instance.data: Still creating... [5m20s elapsed]
aws_db_instance.data: Still creating... [5m30s elapsed]
aws_db_instance.data: Still creating... [5m40s elapsed]
aws_db_instance.data: Still creating... [5m50s elapsed]
aws_db_instance.data: Still creating... [6m0s elapsed]
aws_db_instance.data: Still creating... [6m10s elapsed]
aws_db_instance.data: Still creating... [6m20s elapsed]
aws_db_instance.data: Still creating... [6m30s elapsed]
aws_db_instance.data: Still creating... [6m40s elapsed]
aws_db_instance.data: Still creating... [6m50s elapsed]
aws_db_instance.data: Still creating... [7m0s elapsed]
aws_db_instance.data: Still creating... [7m10s elapsed]
aws_db_instance.data: Still creating... [7m20s elapsed]
aws_db_instance.data: Still creating... [7m30s elapsed]
aws_db_instance.data: Still creating... [7m40s elapsed]
aws_db_instance.data: Still creating... [7m50s elapsed]
aws_db_instance.data: Still creating... [8m0s elapsed]
aws_db_instance.data: Still creating... [8m10s elapsed]
aws_db_instance.data: Still creating... [8m20s elapsed]
aws_db_instance.data: Still creating... [8m30s elapsed]
aws_db_instance.data: Still creating... [8m40s elapsed]
aws_db_instance.data: Still creating... [8m50s elapsed]
aws_db_instance.data: Still creating... [9m0s elapsed]
aws_db_instance.data: Still creating... [9m10s elapsed]
aws_db_instance.data: Still creating... [9m20s elapsed]
aws_db_instance.data: Still creating... [9m30s elapsed]
aws_db_instance.data: Still creating... [9m40s elapsed]
aws_db_instance.data: Still creating... [9m50s elapsed]
aws_db_instance.data: Still creating... [10m0s elapsed]
aws_db_instance.data: Still creating... [10m10s elapsed]
aws_db_instance.data: Still creating... [10m20s elapsed]
aws_db_instance.data: Creation complete after 10m25s [id=terraform-20221128125302260300000003]

Apply complete! Resources: 3 added, 1 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$
```



DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM



➤ Now create the load-balancer by using yaml script.

```
ec2-user@ip-172-30-0-8:~$ cat load_balancer.yaml
# Create load_balancer
resource "aws_lb" "application" {
  internal        = false
  name            = "APPLI-LB"
  load_balancer_type = "application"
  security_groups = [aws_security_group.sg.id]
  subnets        = [aws_subnet.pub-sub-1.id, aws_subnet.pub-sub-2.id]
}

resource "aws_lb_target_group" "tar" {
  name        = "target"
  port        = 80
  protocol    = "HTTP"
  vpc_id      = aws_vpc.mvpc.id
  health_check {
    healthy_threshold   = 2
    unhealthy_threshold = 2
    timeout              = 3
    interval             = 30
    protocol             = "HTTP"
    port                 = 80
    path                 = "/ping"
    matcher              = 200
  }
}

resource "aws_lb_target_group_attachment" "att" {
  target_group_arn = aws_lb_target_group.tar.arn
  count            = 0
  target_id        = "aws_instance.app2.id[count.index]"
  port             = 80
  depends_on       = [aws_instance.app,]
}

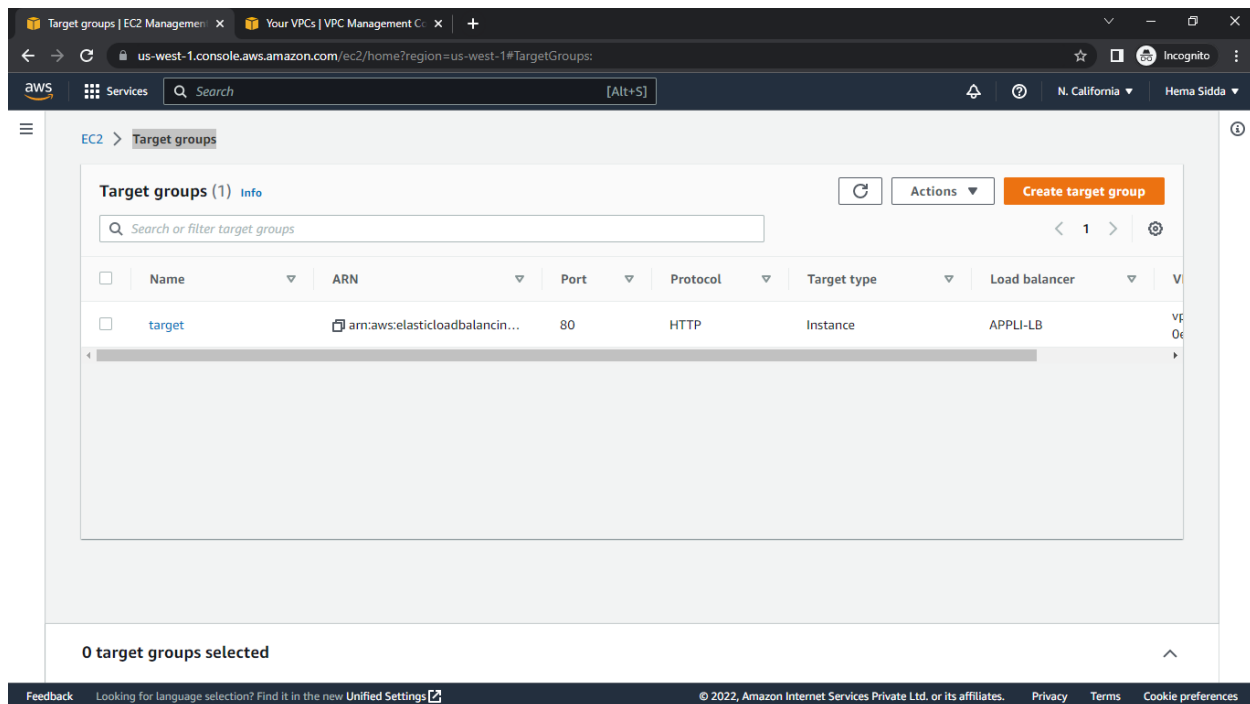
resource "aws_lb_target_group_attachment" "att2" {
  target_group_arn = aws_lb_target_group.tar.arn
  count            = 0
  target_id        = "aws_instance.app2.id[count.index]"
  port             = 80
  depends_on       = [aws_instance.app,]
}

-- INSERT --
```

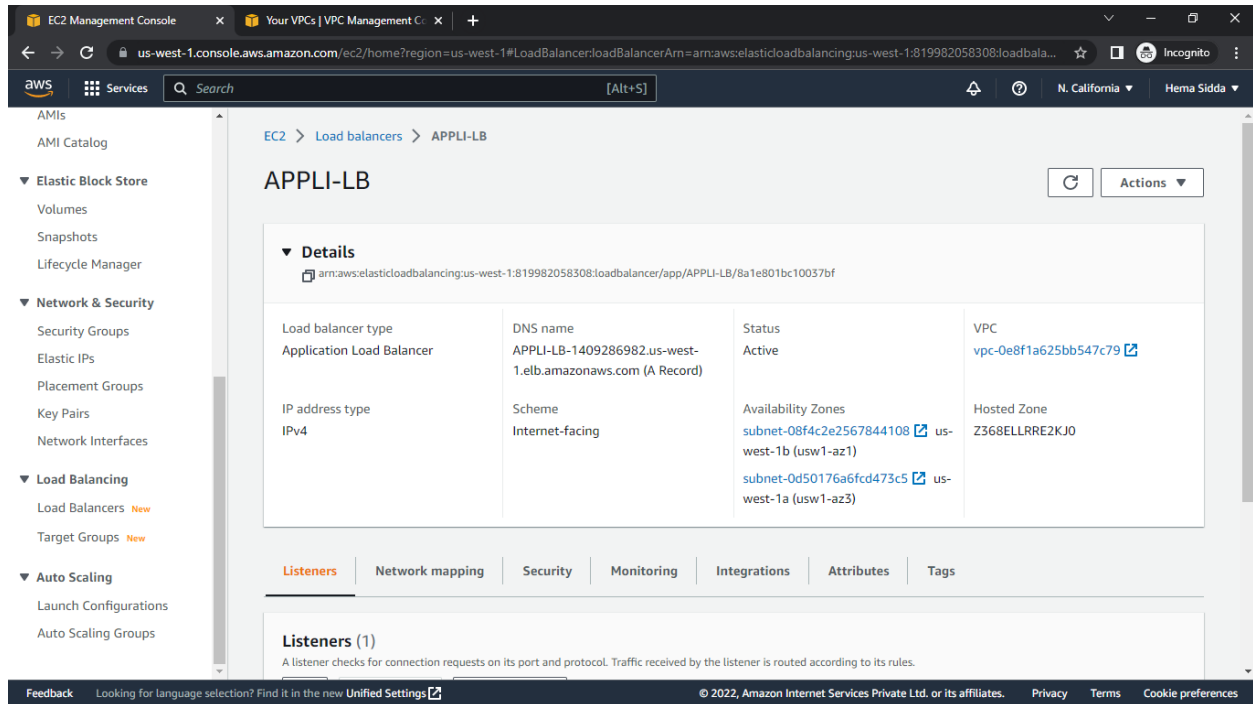
DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

```
ec2-user@ip-172-30-0-8:~  
} matcher = 200  
}  
resource "aws_lb_target_group_attachment" "att" {  
  target_group_arn = aws_lb_target_group.tar.arn  
  count            = 0  
  target_id        = "aws_instance.app2.id[count.index]"  
  port             = 80  
  depends_on       = [aws_instance.app,]  
}  
resource "aws_lb_target_group_attachment" "att2" {  
  target_group_arn = aws_lb_target_group.tar.arn  
  count            = 0  
  target_id        = "aws_instance.app2.id[count.index]"  
  port             = 80  
  depends_on       = [aws_instance.app2,]  
}  
resource "aws_lb_listener" "lb-lis" {  
  load_balancer_arn = aws_lb.application.arn  
  port              = 80  
  protocol           = "HTTP"  
  default_action {  
    type = "forward"  
    target_group_arn = aws_lb_target_group.tar.arn  
  }  
}  
#getting the DNS of load-balancer  
output "lb_dns_name" {  
  description = "the name of the loadbalancer"  
  value       = "${aws_lb.application.dns_name}"  
}  
-- INSERT --
```

418,1 Bot



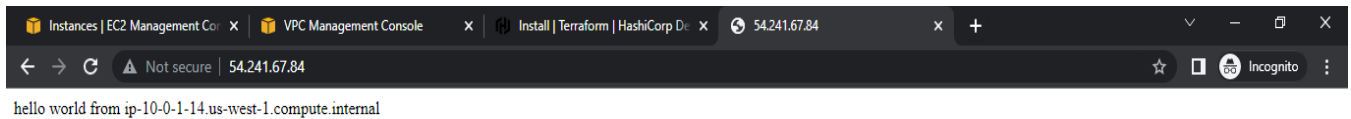
DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM



```
ec2-user@ip-172-30-0-8~  
aws_db_instance.data: Still creating... [7m20s elapsed]  
aws_db_instance.data: Still creating... [7m30s elapsed]  
aws_db_instance.data: Still creating... [7m40s elapsed]  
aws_db_instance.data: Still creating... [7m50s elapsed]  
aws_db_instance.data: Still creating... [8m0s elapsed]  
  
aws_db_instance.data: Still creating... [8m10s elapsed]  
aws_db_instance.data: Still creating... [8m20s elapsed]  
aws_db_instance.data: Still creating... [8m30s elapsed]  
aws_db_instance.data: Still creating... [8m40s elapsed]  
aws_db_instance.data: Still creating... [8m50s elapsed]  
aws_db_instance.data: Still creating... [9m0s elapsed]  
aws_db_instance.data: Still creating... [9m10s elapsed]  
aws_db_instance.data: Still creating... [9m20s elapsed]  
aws_db_instance.data: Still creating... [9m30s elapsed]  
aws_db_instance.data: Still creating... [9m40s elapsed]  
aws_db_instance.data: Still creating... [9m50s elapsed]  
aws_db_instance.data: Still creating... [10m0s elapsed]  
aws_db_instance.data: Creation complete after 10m6s [id=terraform-20221130125933895100000007]  
  
Apply complete! Resources: 26 added, 0 changed, 0 destroyed.  
  
Outputs:  
  
lb_dns_name = "APPLI-LB-1409286982.us-west-1.elb.amazonaws.com"  
[ec2-user@ip-172-30-0-8 ~]$  
[ec2-user@ip-172-30-0-8 ~]$  
[ec2-user@ip-172-30-0-8 ~]$  
[ec2-user@ip-172-30-0-8 ~]$  
[ec2-user@ip-172-30-0-8 ~]$  
[ec2-user@ip-172-30-0-8 ~]$
```


DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

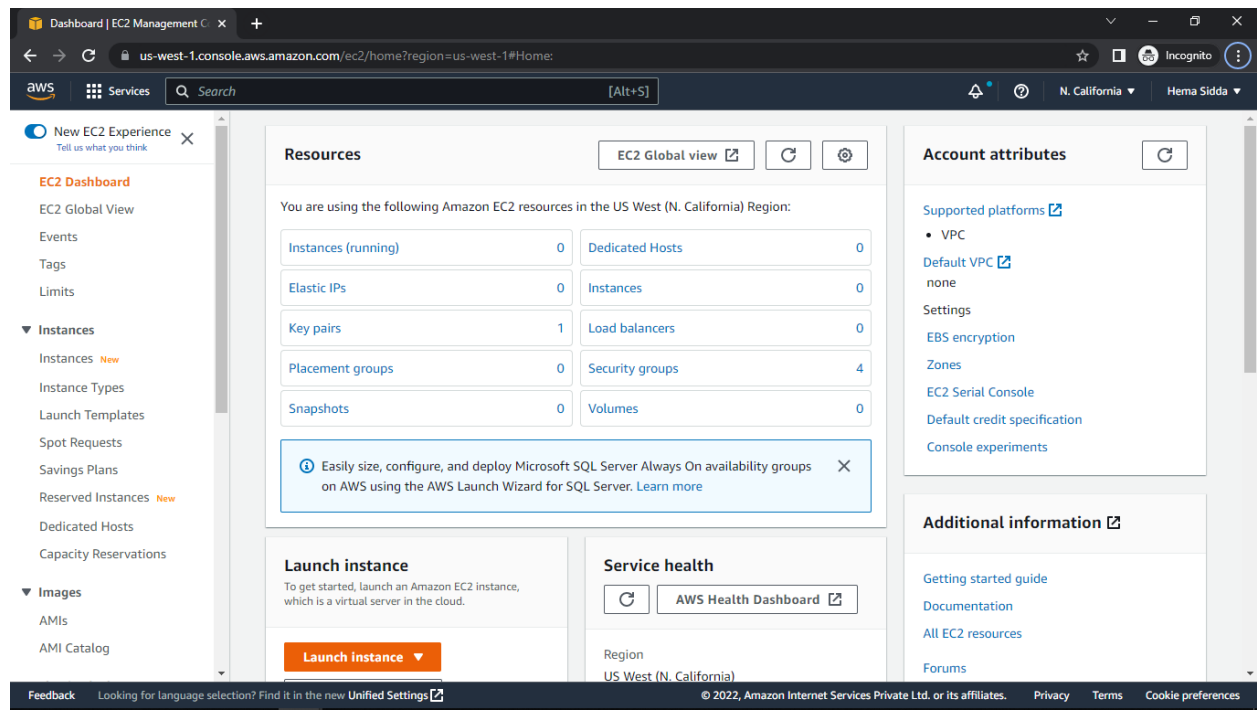
- Now go to EC2 instances and select the public ip(ipv4) and browse it on Google.



DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

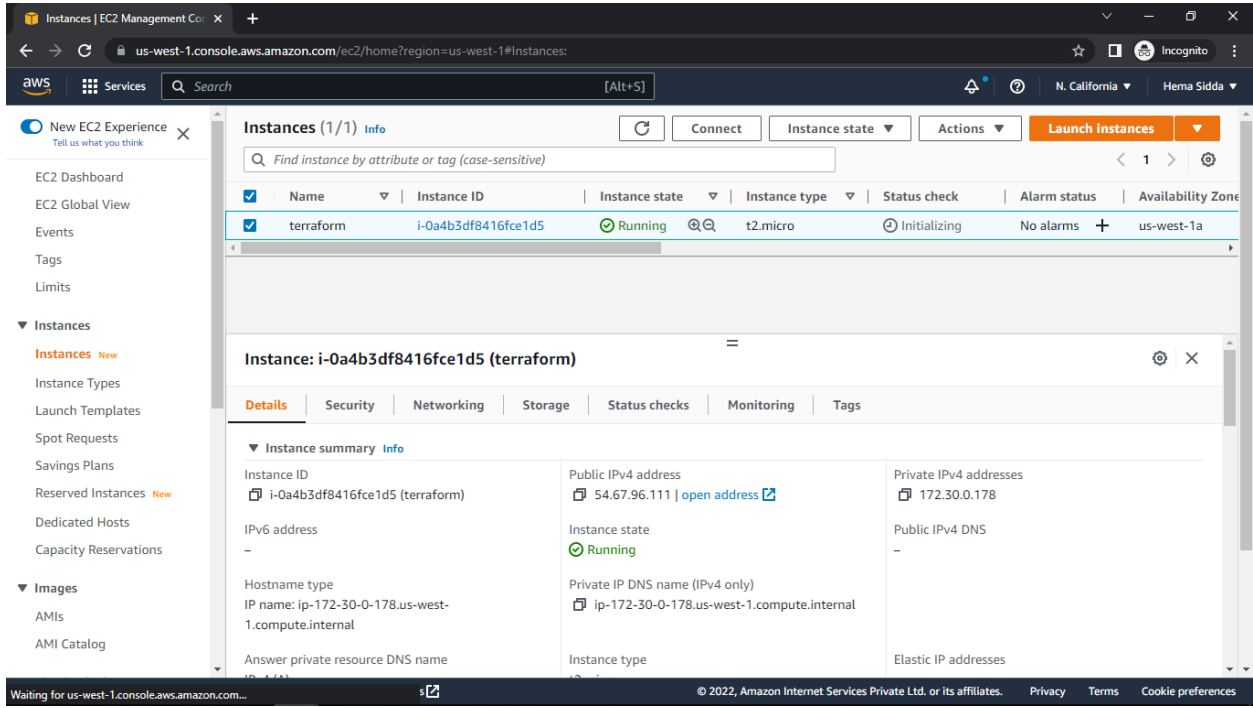
METHOD-2

- First open the AWS account by using credentials and go with EC2 instances.



- Now create or launch the server by selecting AMI's and key-pairs and also give the port ranges like SSH(22) and HTTP(80).

USING TERRAFORM



➤ Now connect the launched server through GIT-Bash terminal.

```
ec2-user@ip-172-30-0-178:~  
acer@DESKTOP-ONJL693 MINGW64 ~  
$ cd Downloads  
acer@DESKTOP-ONJL693 MINGW64 ~/Downloads  
$ ssh -i "terraform.pem" ec2-user@54.67.96.111  
The authenticity of host '54.67.96.111 (54.67.96.111)' can't be established.  
ED25519 key fingerprint is SHA256:0bvB2/AckDV74E4SwUaOceptA/Ry8vI1zGlbj9TioXI.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '54.67.96.111' (ED25519) to the list of known hosts.  
  
  _ |  _ | _ )  
 _ |  _ | _ /  
 _ | _ | _ |  
Amazon Linux 2 AMI  
  
https://aws.amazon.com/amazon-linux-2/  
1 package(s) needed for security, out of 1 available  
Run "sudo yum update" to apply all updates.  
[ec2-user@ip-172-30-0-178 ~]$
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY

USING TERRAFORM

- Now install the terraform by using commands those are available in Google browse it by the name <Terraform download>.

```
ec2-user@ip-172-30-0-178:~  
=====
```

Package	Arch	Version	Repository	Size
Installing: terraform	x86_64	1.3.5-1	hashicorp	13 M

```
=====
```

Transaction Summary

```
=====
```

Install 1 Package

Total download size: 13 M
Installed size: 58 M
Downloading packages:
warning: /var/cache/yum/x86_64/2/hashicorp/packages/terraform-1.3.5-1.x86_64.rpm: Header V4 RSA/SHA512 Signature, key ID a3219f7b: NOKEY
Public key for terraform-1.3.5-1.x86_64.rpm is not installed
terraform-1.3.5-1.x86_64.rpm | 13 MB 00:00:00
Retrieving key from https://rpm.releases.hashicorp.com/gpg
Importing GPG key 0xA3219F7B:
 Userid : "HashiCorp Security (HashiCorp Package Signing) <security+packaging@hashicorp.com>"
 Fingerprint: e8a0 32e0 94d8 eb4e a189 d270 da41 8c88 a321 9f7b
 From : https://rpm.releases.hashicorp.com/gpg
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
 Installing : terraform-1.3.5-1.x86_64 1/1
 Verifying : terraform-1.3.5-1.x86_64 1/1
Installed:
 terraform.x86_64 0:1.3.5-1
Complete!
[ec2-user@ip-172-30-0-178 ~]\$ ls
[ec2-user@ip-172-30-0-178 ~]\$ ls

- Now create vpc by using yaml scripting for that create a file as <vi file.tf> and write the yaml script to create vpc and save it by using “:wq”.

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~$
provider "aws" {
  region      = "us-west-1"
  access_key  = "AKIA352V5INCBUYKAMX2"
  secret_key  = "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"
}
#Create the vpc
resource "aws_vpc" "myvpc" {
  cidr_block     = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "myvpc"
  }
}

-- INSERT --
```

- Now change the environment into terraform mode by using command as <terraform init>.

```
ec2-user@ip-172-30-0-178:~$
Running transaction
  Installing : terraform-1.3.5-1.x86_64
  Verifying  : terraform-1.3.5-1.x86_64
Installed:
  terraform.x86_64 0:1.3.5-1

Complete!
[ec2-user@ip-172-30-0-178 ~]$ ls
[ec2-user@ip-172-30-0-178 ~]$ ls
[ec2-user@ip-172-30-0-178 ~]$ vi vpc.tf
[ec2-user@ip-172-30-0-178 ~]$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.41.0...
- Installed hashicorp/aws v4.41.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
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-172-30-0-178 ~]$
```

DEPLOY THREE-TIER ARCHITECHTURE IN AWS BY

USING TERRAFORM

- Now set the yaml script alignment by using command as <terraform fmt>.

```
ec2-user@ip-172-30-0-178:~$ cat vpc.tf
provider "aws" {
  region     = "us-west-1"
  access_key = "AKIA352V5INCBUYKAMX2"
  secret_key = "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"
}
#Create the vpc
resource "aws_vpc" "myvpc" {
  cidr_block     = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "myvpc"
  }
}

[ec2-user@ip-172-30-0-178 ~]$ terraform fmt
vpc.tf
[ec2-user@ip-172-30-0-178 ~]$ cat vpc.tf
provider "aws" {
  region     = "us-west-1"
  access_key = "AKIA352V5INCBUYKAMX2"
  secret_key = "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"
}
#Create the vpc
resource "aws_vpc" "myvpc" {
  cidr_block     = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "myvpc"
  }
}

[ec2-user@ip-172-30-0-178 ~]$
```

- Now validate the script for correcting the spelling mistakes by using command as <terraform validate>.

```
ec2-user@ip-172-30-0-178:~$ cat vpc.tf
access_key = "AKIA352V5INCBUYKAMX2"
secret_key = "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"
}
#Create the vpc
resource "aws_vpc" "myvpc" {
  cidr_block     = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "myvpc"
  }
}

[ec2-user@ip-172-30-0-178 ~]$ terraform fmt
vpc.tf
[ec2-user@ip-172-30-0-178 ~]$ cat vpc.tf
provider "aws" {
  region     = "us-west-1"
  access_key = "AKIA352V5INCBUYKAMX2"
  secret_key = "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"
}
#Create the vpc
resource "aws_vpc" "myvpc" {
  cidr_block     = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "myvpc"
  }
}

[ec2-user@ip-172-30-0-178 ~]$ terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-30-0-178 ~]$
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY

USING TERRAFORM

➤ Now create the vpc by using command as <terraform apply>.

```
ec2-user@ip-172-30-0-178:~$ terraform apply
# aws_vpc.mvpc will be created
+ resource "aws_vpc" "mvpc" {
+   arn                               = (known after apply)
+   cidr_block                        = "10.0.0.0/16"
+   default_network_acl_id           = (known after apply)
+   default_route_table_id           = (known after apply)
+   default_security_group_id        = (known after apply)
+   dhcp_options_id                  = (known after apply)
+   enable_classiclink                = (known after apply)
+   enable_classiclink_dns_support    = (known after apply)
+   enable_dns_hostnames              = (known after apply)
+   enable_dns_support                = true
+   enable_network_address_usage_metrics = (known after apply)
+   id                               = (known after apply)
+   instance_tenancy                  = "default"
+   ipv6_association_id               = (known after apply)
+   ipv6_cidr_block                   = (known after apply)
+   ipv6_cidr_block_network_border_group = (known after apply)
+   main_route_table_id               = (known after apply)
+   owner_id                          = (known after apply)
+   tags                              = {
+     "Name" = "myvpc"
+   }
+   tags_all                          = {
+     "Name" = "myvpc"
+   }
+ }

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: |
```

```
ec2-user@ip-172-30-0-178:~$ terraform apply
+ default_security_group_id           = (known after apply)
+ dhcp_options_id                     = (known after apply)
+ enable_classiclink                  = (known after apply)
+ enable_classiclink_dns_support       = (known after apply)
+ enable_dns_hostnames                 = (known after apply)
+ enable_dns_support                   = true
+ enable_network_address_usage_metrics = (known after apply)
+ id                                   = (known after apply)
+ instance_tenancy                     = "default"
+ ipv6_association_id                  = (known after apply)
+ ipv6_cidr_block                      = (known after apply)
+ ipv6_cidr_block_network_border_group = (known after apply)
+ main_route_table_id                  = (known after apply)
+ owner_id                            = (known after apply)
+ tags                                 = {
+   "Name" = "myvpc"
+ }
+ tags_all                             = {
+   "Name" = "myvpc"
+ }
+ }

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.

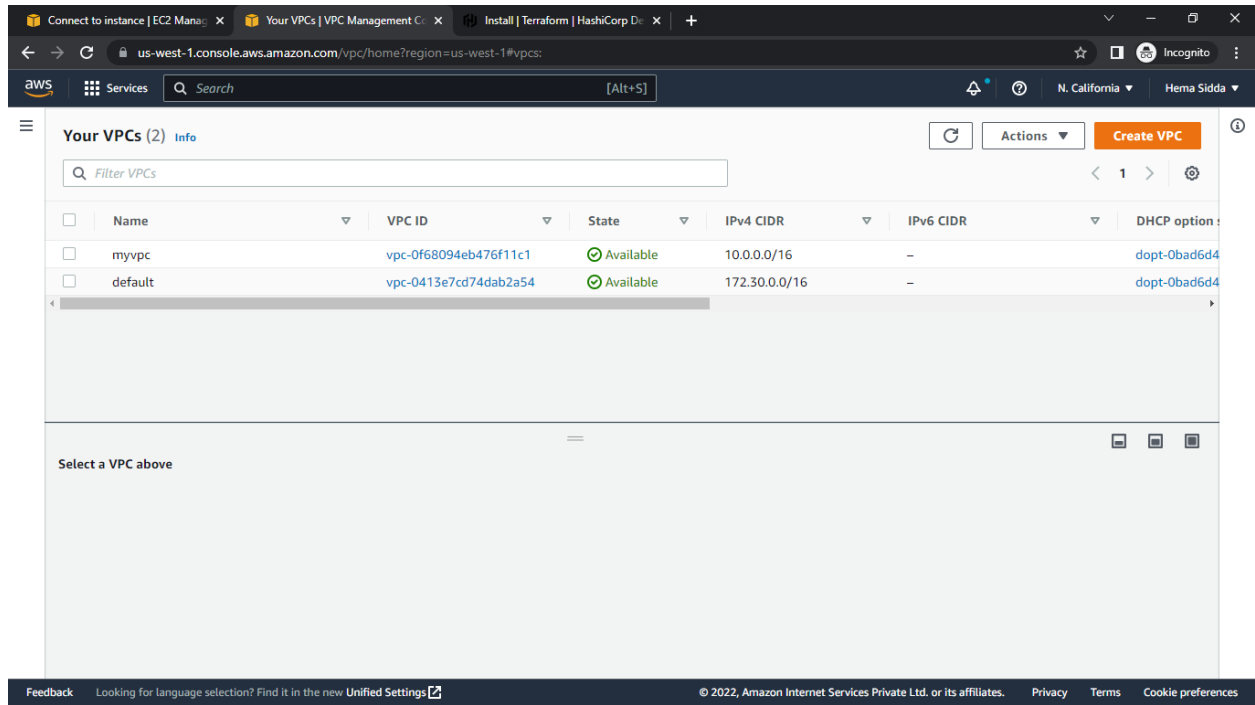
Enter a value: yes

aws_vpc.mvpc: Creating...
aws_vpc.mvpc: Creation complete after 0s [id=vpc-0f68094eb476f11c1]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

- Now go to your AWS account and open vpc services and check your vpc created or not.



- Now create the IGW and attach it to created vpc for that create a file as <vi igw.tf>

USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~  
#Creating IGW  
resource "aws_internet_gateway" "igw" {  
  vpc_id = aws_vpc.mvpc.id  
  tags = {  
    Name = "my-igw"  
  }  
}
```

-- INSERT --

9,1 A11

```

ec2-user@ip-172-30-0-178:~
aws_vpc.mvpc: Creating...
aws_vpc.mvpc: Creation complete after 0s [id=vpc-0f68094eb476f11c1]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$ vi igw.tf
[ec2-user@ip-172-30-0-178 ~]$ terraform apply --auto-approve
aws_vpc.mvpc: Refreshing state... [id=vpc-0f68094eb476f11c1]

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

Terraform will perform the following actions:

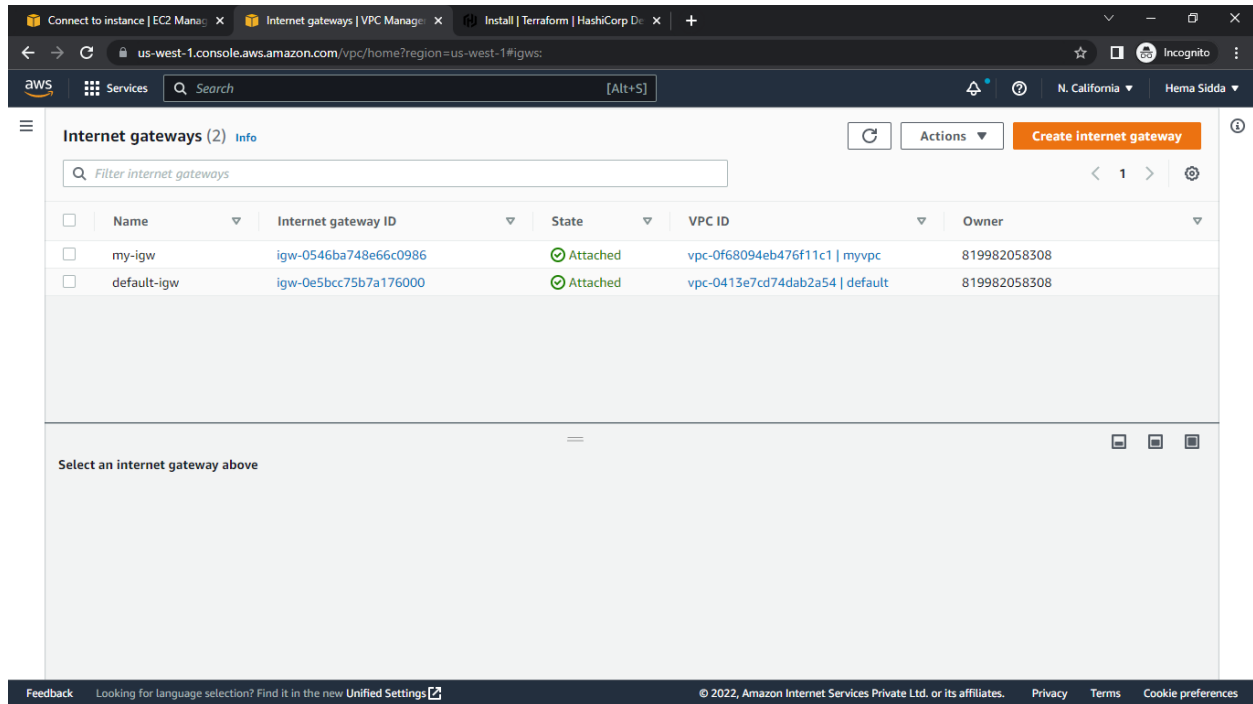
# aws_internet_gateway.igw will be created
+ resource "aws_internet_gateway" "igw" {
+   arn          = (known after apply)
+   id           = (known after apply)
+   owner_id     = (known after apply)
+   tags         = {
+     + "Name" = "my-igw"
+   }
+   tags_all     = {
+     + "Name" = "my-igw"
+   }
+   vpc_id       = "vpc-0f68094eb476f11c1"
+ }

Plan: 1 to add, 0 to change, 0 to destroy.
aws_internet_gateway.igw: Creating...
aws_internet_gateway.igw: Creation complete after 0s [id=igw-0546ba748e66c0986]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$ |

```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM



- Now create the subnets by using yaml script for that use this command <vi subnets.tf>

```
ec2-user@ip-172-30-0-178:~$
#Creating pub-subnet1
resource "aws_subnet" "pub-sub-1" {
  vpc_id            = aws_vpc.myvpc.id
  cidr_block        = "10.0.1.0/24"
  availability_zone  = "us-west-1a"
  map_public_ip_on_launch = "true"
  tags = {
    Name = "pub-subnet-1"
  }
}
#Creating pvt-subnet1
resource "aws_subnet" "pvt-sub-1" {
  vpc_id            = aws_vpc.myvpc.id
  cidr_block        = "10.0.2.0/24"
  availability_zone  = "us-west-1a"
  map_public_ip_on_launch = "false"
  tags = {
    Name = "pvt-subnet-1"
  }
}
#Creating pub-subnet2
resource "aws_subnet" "pub-sub-2" {
  vpc_id            = aws_vpc.myvpc.id
  cidr_block        = "10.0.3.0/24"
  availability_zone  = "us-west-1b"
  map_public_ip_on_launch = "true"
  tags = {
    Name = "pub-subnet-2"
  }
}
-- INSERT --
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY

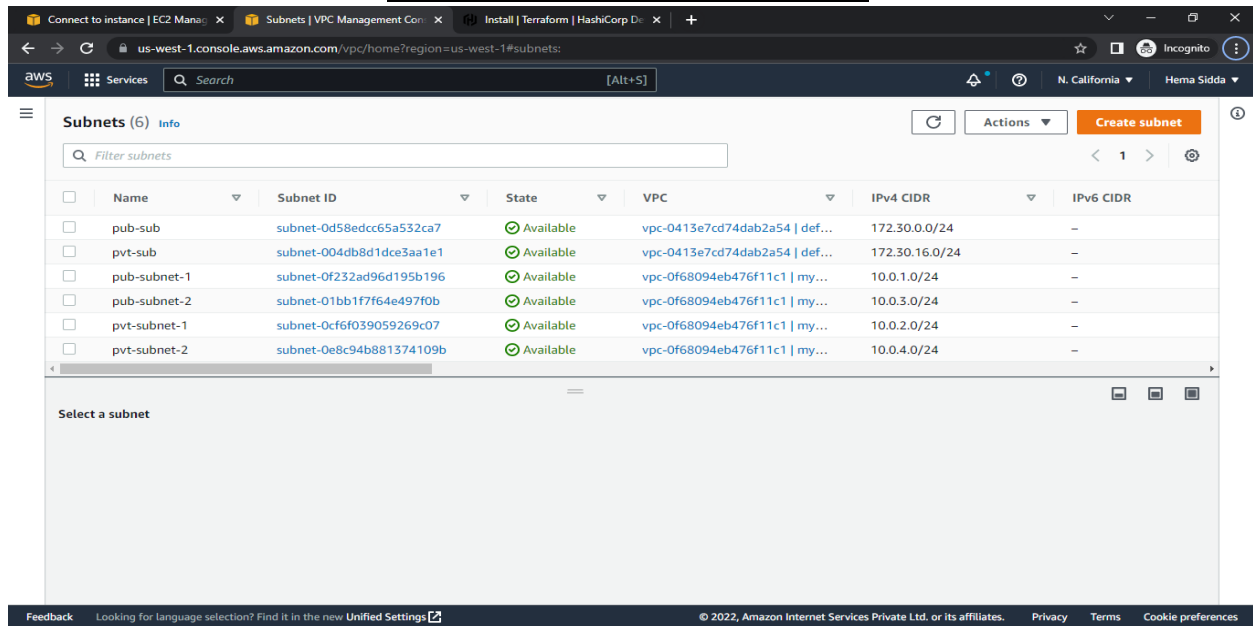
USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~  
cidr_block           = "10.0.2.0/24"  
availability_zone     = "us-west-1a"  
map_public_ip_on_launch = "false"  
  
tags = {  
  Name = "pvt-subnet-1"  
}  
}  
  
#Creating pub-subnet2  
resource "aws_subnet" "pub-sub-2" {  
  vpc_id           = aws_vpc.mvpc.id  
  cidr_block       = "10.0.3.0/24"  
  availability_zone = "us-west-1b"  
  map_public_ip_on_launch = "true"  
  tags = {  
    Name = "pub-subnet-2"  
  }  
}  
  
#Creating pvt-subnet2  
resource "aws_subnet" "pvt-sub-2" {  
  vpc_id           = aws_vpc.mvpc.id  
  cidr_block       = "10.0.4.0/24"  
  availability_zone = "us-west-1b"  
  map_public_ip_on_launch = "false"  
  tags = {  
    Name = "pvt-subnet-2"  
  }  
}  
  
-- INSERT --
```

- Now by using single command I can create the resource that commad is <terraform apply --auto-approve>

```
ec2-user@ip-172-30-0-178:~  
+ availability_zone           = "us-west-1b"  
+ 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)  
+ tags                        = {  
  + "Name" = "pvt-subnet-2"  
}  
+ tags_all                    = {  
  + "Name" = "pvt-subnet-2"  
}  
+ vpc_id                      = "vpc-0f68094eb476f11c1"  
}  
  
Plan: 4 to add, 0 to change, 0 to destroy.  
aws_subnet.pub-sub-1: Creating...  
aws_subnet.pub-sub-2: Creating...  
aws_subnet.pvt-sub-1: Creating...  
aws_subnet.pvt-sub-2: Creating...  
aws_subnet.pvt-sub-1: Creation complete after 0s [id=subnet-0cf6f039059269c07]  
aws_subnet.pvt-sub-2: Creation complete after 0s [id=subnet-0e8c94b881374109b]  
aws_subnet.pub-sub-1: Still creating... [10s elapsed]  
aws_subnet.pub-sub-2: Still creating... [10s elapsed]  
aws_subnet.pub-sub-1: Creation complete after 11s [id=subnet-0f232ad96d195b196]  
aws_subnet.pub-sub-2: Creation complete after 11s [id=subnet-01bb1f7f64e497f0b]  
  
Apply complete! Resources: 4 added, 0 changed, 0 destroyed.  
[ec2-user@ip-172-30-0-178 ~]$
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM



- Now create the NAT-gateway for that create a .tf file and enter the yaml script.

```
ec2-user@ip-172-30-0-178:~$
# Create elastic ip
resource "aws_eip" "elastic" {
  vpc = true
}

# Create Nat-gateway
resource "aws_nat_gateway" "nat" {
  allocation_id = aws_eip.elastic.id
  subnet_id     = aws_subnet.pub-sub-1.id
  connectivity_type = "public"
  tags = {
    Name = "mynat"
  }
}

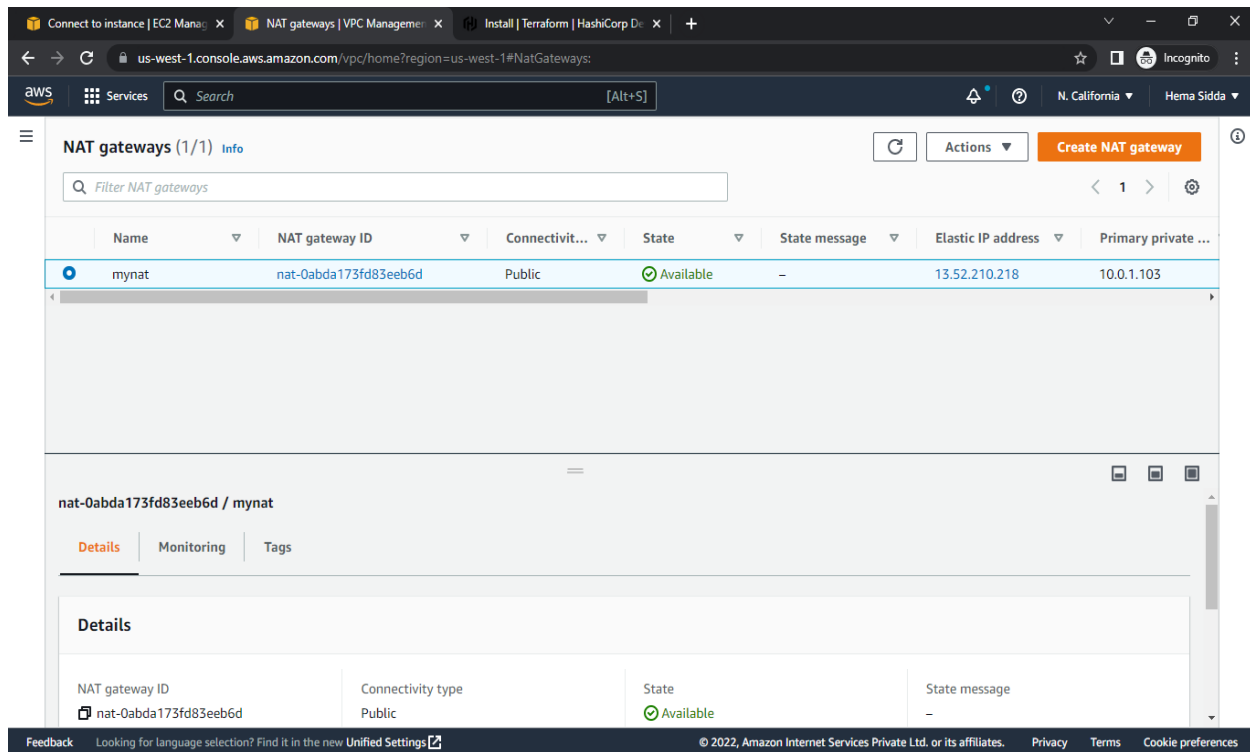
-- INSERT --
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~$ terraform apply
+ id = (known after apply)
+ network_interface_id = (known after apply)
+ private_ip = (known after apply)
+ public_ip = (known after apply)
+ subnet_id = "subnet-0f232ad96d195b196"
+ tags = {
  + "Name" = "mynat"
}
+ tags_all = {
  + "Name" = "mynat"
}

Plan: 2 to add, 0 to change, 0 to destroy.
aws_eip.elastic: Creating...
aws_eip.elastic: Creation complete after 0s [id=eipalloc-034630da659b77654]
aws_nat_gateway.nat: Creating...
aws_nat_gateway.nat: Still creating... [10s elapsed]
aws_nat_gateway.nat: Still creating... [20s elapsed]
aws_nat_gateway.nat: Still creating... [30s elapsed]
aws_nat_gateway.nat: Still creating... [40s elapsed]
aws_nat_gateway.nat: Still creating... [50s elapsed]
aws_nat_gateway.nat: Still creating... [1m0s elapsed]
aws_nat_gateway.nat: Still creating... [1m10s elapsed]
aws_nat_gateway.nat: Still creating... [1m20s elapsed]
aws_nat_gateway.nat: Still creating... [1m30s elapsed]
aws_nat_gateway.nat: Still creating... [1m40s elapsed]
aws_nat_gateway.nat: Still creating... [1m50s elapsed]
aws_nat_gateway.nat: Still creating... [2m0s elapsed]
aws_nat_gateway.nat: Still creating... [2m10s elapsed]
aws_nat_gateway.nat: Still creating... [2m20s elapsed]
aws_nat_gateway.nat: Creation complete after 2m25s [id=nat-0abda173fd83eeb6d]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$
```



DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

- Now go to create the route table and associate subnets, Internet gateway and NAT gate way for that create a .tf file by using the vi mode as <vi route.tf>

```
ec2-user@ip-172-30-0-178:~$ vi route.tf
#Create Pub-Route-table
resource "aws_route_table" "pub-route" {
  vpc_id = aws_vpc.mvpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }
  tags = {
    Name = "pub-rot"
  }
}

#Create Pvt-Route-table
resource "aws_route_table" "pvt-route" {
  vpc_id = aws_vpc.mvpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_nat_gateway.nat.id
  }
  tags = {
    Name = "pvt-rot"
  }
}

#subnets Associations
#pub-subnet-association
resource "aws_route_table_association" "a" {
  subnet_id      = aws_subnet.pub-sub-1.id
  route_table_id = aws_route_table.pub-route.id
}

-- INSERT --
```

```
ec2-user@ip-172-30-0-178:~$ vi route.tf
}

}

#subnets Associations
#pub-subnet-association
resource "aws_route_table_association" "a" {
  subnet_id      = aws_subnet.pub-sub-1.id
  route_table_id = aws_route_table.pub-route.id
}

#pvt subnet-association
resource "aws_route_table_association" "b" {
  subnet_id      = aws_subnet.pvt-sub-1.id
  route_table_id = aws_route_table.pvt-route.id
}

#subnets Associations
#pub-subnet-association
resource "aws_route_table_association" "c" {
  subnet_id      = aws_subnet.pub-sub-2.id
  route_table_id = aws_route_table.pub-route.id
}

#pvt subnet-association
resource "aws_route_table_association" "d" {
  subnet_id      = aws_subnet.pvt-sub-2.id
  route_table_id = aws_route_table.pvt-route.id
}

-- INSERT --
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

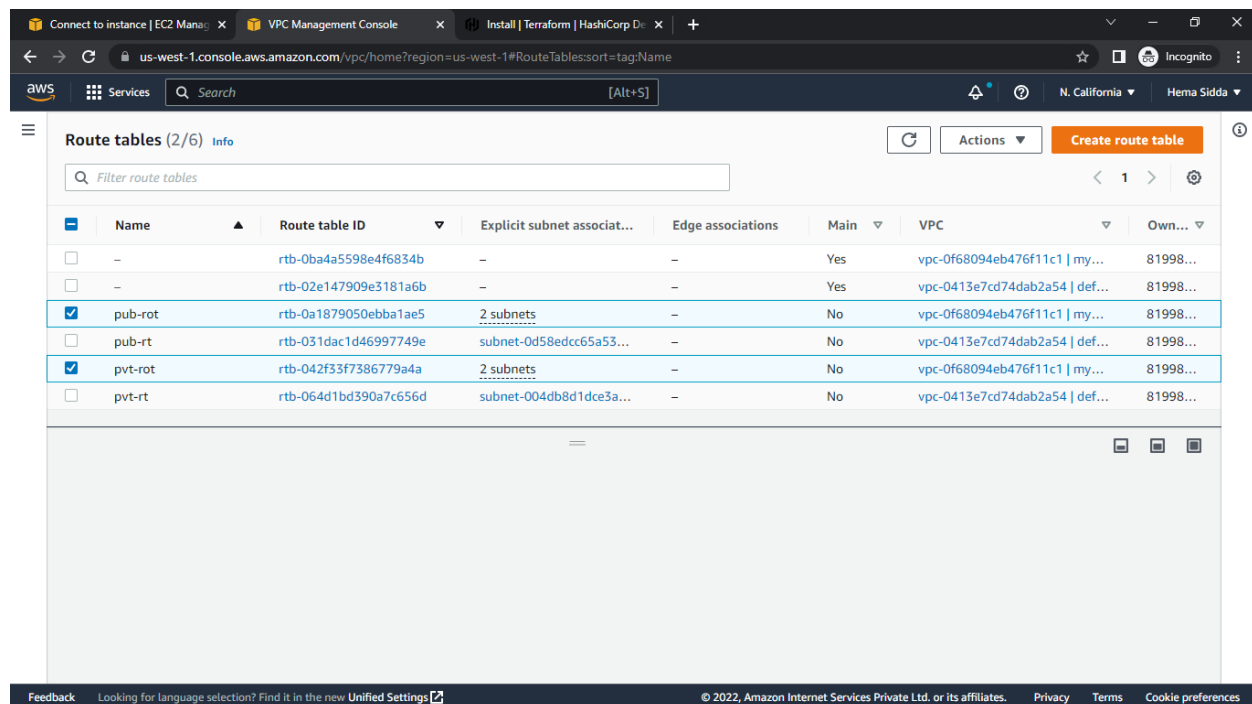
```
ec2-user@ip-172-30-0-178:~$ terraform apply
+ id = (known after apply)
+ route_table_id = (known after apply)
+ subnet_id = "subnet-0cf6f039059269c07"
}

# aws_route_table_association.c will be created
+ resource "aws_route_table_association" "c" {
+ id = (known after apply)
+ route_table_id = (known after apply)
+ subnet_id = "subnet-01bb1f7f64e497f0b"
}

# aws_route_table_association.d will be created
+ resource "aws_route_table_association" "d" {
+ id = (known after apply)
+ route_table_id = (known after apply)
+ subnet_id = "subnet-0e8c94b881374109b"
}

Plan: 6 to add, 0 to change, 0 to destroy.
aws_route_table.pub-route: Creating...
aws_route_table.pvt-route: Creating...
aws_route_table.pvt-route: Creation complete after 0s [id=rtb-042f33f7386779a4a]
aws_route_table_association.d: Creating...
aws_route_table_association.b: Creating...
aws_route_table.pub-route: Creation complete after 0s [id=rtb-0a1879050ebba1ae5]
aws_route_table_association.c: Creating...
aws_route_table_association.a: Creating...
aws_route_table_association.b: Creation complete after 0s [id=rtbassoc-0579536724ad2da7a]
aws_route_table_association.a: Creation complete after 1s [id=rtbassoc-00aeb0e5d8f6862]
aws_route_table_association.d: Creation complete after 1s [id=rtbassoc-0533d5fd5d09e2a50]
aws_route_table_association.c: Creation complete after 1s [id=rtbassoc-0222db84d94adb75c]

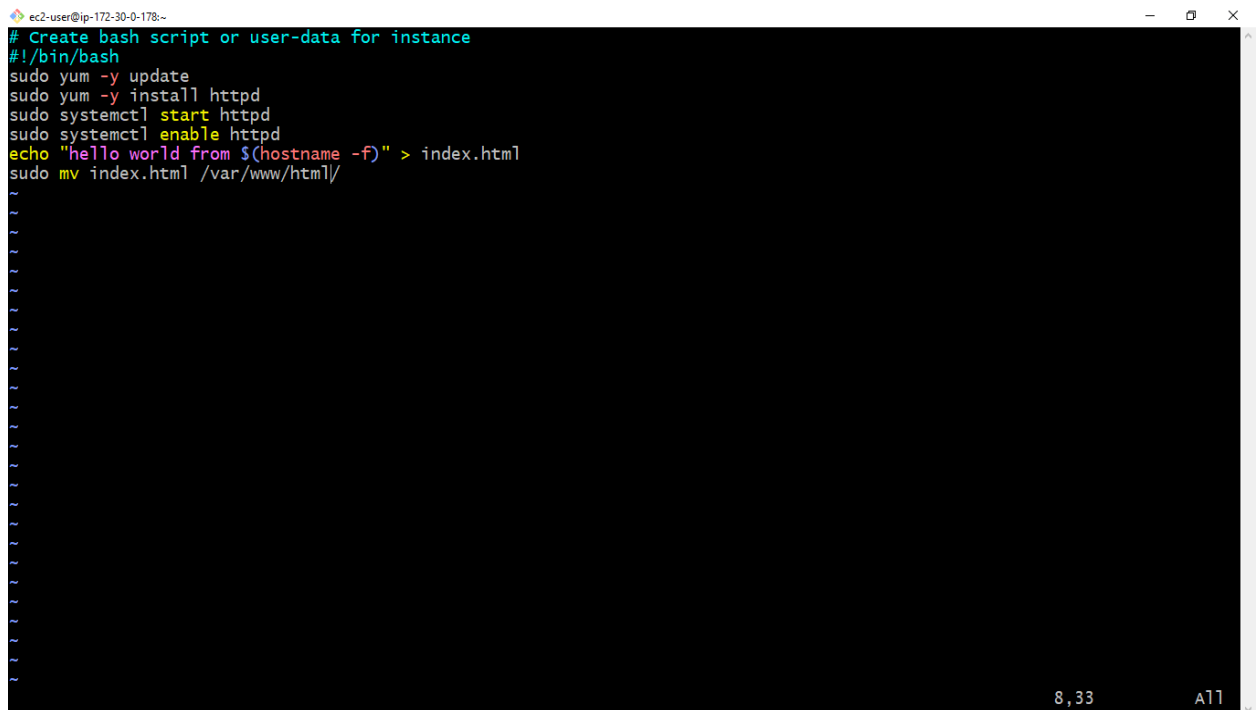
Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$
```



	Name	Route table ID	Explicit subnet associat...	Edge associations	Main	VPC	Own...
<input type="checkbox"/>	-	rtb-0ba4a5598e4f6834b	-	-	Yes	vpc-0f68094eb476f11c1 my...	81998...
<input type="checkbox"/>	-	rtb-02e147909e3181a6b	-	-	Yes	vpc-0413e7cd74dab2a54 def...	81998...
<input checked="" type="checkbox"/>	pub-rot	rtb-0a1879050ebba1ae5	2 subnets	-	No	vpc-0f68094eb476f11c1 my...	81998...
<input type="checkbox"/>	pub-rt	rtb-031dac1d46997749e	subnet-0d58edcc65a53...	-	No	vpc-0413e7cd74dab2a54 def...	81998...
<input checked="" type="checkbox"/>	pvt-rot	rtb-042f33f7386779a4a	2 subnets	-	No	vpc-0f68094eb476f11c1 my...	81998...
<input type="checkbox"/>	pvt-rt	rtb-064d1bd390a7c656d	subnet-004db8d1dce3a...	-	No	vpc-0413e7cd74dab2a54 def...	81998...

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

- Now create the bash script file by the data.sh

A terminal window with a black background and white text. The title bar shows 'ec2-user@ip-172-30-0-178:~'. The script content is as follows:

```
# Create bash script or user-data for instance
#!/bin/bash
sudo yum -y update
sudo yum -y install httpd
sudo systemctl start httpd
sudo systemctl enable httpd
echo "hello world from $(hostname -f)" > index.html
sudo mv index.html /var/www/html/
```

The script ends with several tilde characters (~) and a cursor. The bottom right corner of the terminal shows '8,33' and 'A11'.

- Now create the public instance by using yaml scripting.

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY

USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~$
# Create instance
resource "aws_instance" "app" {
  ami           = "ami-0f5e8a042c8bfcd5e"
  instance_type = "t2.micro"
  count         = 1
  key_name      = "terraform"
  subnet_id    = aws_subnet.pub-sub-1.id
  vpc_security_group_ids = ["${aws_security_group.sg.id}"]
  user_data    = "${file("data.sh")}"
  tags = {
    Name = "webapplication"
  }
}

# Create Security-group
resource "aws_security_group" "sg" {
  vpc_id = aws_vpc.mvpc.id

  #Inbound Rules
  # HTTP access from anywhere
  ingress {
    from_port = 80
    to_port   = 80
    protocol  = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  # HTTPS access from any where
  ingress {
    from_port = 443
    to_port   = 443
    protocol  = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

-- INSERT --
```

1,1 Top

```
ec2-user@ip-172-30-0-178:~$
  from_port = 80
  to_port   = 80
  protocol  = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}
# HTTPS access from any where
ingress {
  from_port = 443
  to_port   = 443
  protocol  = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}

# Ssh access from any where
ingress {
  from_port = 22
  to_port   = 22
  protocol  = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}
#outbound Rules
#internet access to anywhere
egress {
  from_port = 0
  to_port   = 0
  protocol  = "-1"
  cidr_blocks = ["0.0.0.0/0"]
}
tags = {
  Name = "web-sg"
}
}

-- INSERT --
```

55,1 Bot

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~$ terraform apply
+ 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" = "web-sg"
+ }
+ tags_all          = {
+   "Name" = "web-sg"
+ }
+ vpc_id            = "vpc-0f68094eb476f11c1"
}

Plan: 2 to add, 1 to change, 0 to destroy.
aws_security_group.sg: Creating...
aws_route_table.pvt-route: Modifying... [id=rtb-042f33f7386779a4a]
aws_route_table.pvt-route: Modifications complete after 0s [id=rtb-042f33f7386779a4a]
aws_security_group.sg: Creation complete after 1s [id=sg-04db0c061609b85bd]
aws_instance.app[0]: Creating...
aws_instance.app[0]: Still creating... [10s elapsed]
aws_instance.app[0]: Still creating... [20s elapsed]
aws_instance.app[0]: Still creating... [30s elapsed]
aws_instance.app[0]: Still creating... [40s elapsed]
aws_instance.app[0]: Creation complete after 41s [id=i-0794f51fb41d9fb43]

Apply complete! Resources: 2 added, 1 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$
```

The screenshot displays the AWS Management Console interface. At the top, there are tabs for 'Instances | EC2 Management Console', 'VPC Management Console', and 'Install | Terraform | HashiCorp Dev'. The main content area shows the 'Instances (1/2)' page. A table lists two instances: 'webapplication' (ID: i-0794f51fb41d9fb43) and 'terraform' (ID: i-0a4b3df8416fce1d5). Both are in a 'Running' state. Below the table, the details for the 'webapplication' instance are shown, including its public IP address (54.241.67.84) and private IP address (10.0.1.14).

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
webapplication	i-0794f51fb41d9fb43	Running	t2.micro	Initializing	No alarms	us-west-1a	-
terraform	i-0a4b3df8416fce1d5	Running	t2.micro	2/2 checks passed	No alarms	us-west-1a	-

Instance: i-0794f51fb41d9fb43 (webapplication)

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

Instance summary Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0794f51fb41d9fb43 (webapplication)	54.241.67.84 open address	10.0.1.14

IPv6 address: -

Hostname type: IP name: ip-10-0-1-14.us-west-1.compute.internal

Private IP DNS name (IPv4 only): ip-10-0-1-14.us-west-1.compute.internal

Answer private resource DNS name: -

Instance type: t2.micro

Elastic IP addresses: -

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

- Now create the pvt instance by using the yaml script of pvt instance .

```
ec2-user@ip-172-30-0-178:~$
# Create instance
resource "aws_instance" "web-app" {
  ami           = "ami-0f5e8a042c8bfcd5e"
  instance_type = "t2.micro"
  count         = 1
  key_name      = "terraform"
  subnet_id     = aws_subnet.pvt-sub-1.id
  vpc_security_group_ids = ["${aws_security_group.demo.id}"]
  tags = {
    Name = "web-pvt"
  }
}

# Create Security-group
resource "aws_security_group" "demo" {
  vpc_id = aws_vpc.mvpc.id

  #Inbound Rules
  # HTTP access from anywhere
  ingress {
    from_port = 80
    to_port   = 80
    protocol  = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  # HTTPS access from any where
  ingress {
    from_port = 443
    to_port   = 443
    protocol  = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Ssh access from any where
  -- INSERT --
1,1 Top
```

```
ec2-user@ip-172-30-0-178:~$
to_port = 80
protocol = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}
# HTTPS access from any where
ingress {
  from_port = 443
  to_port   = 443
  protocol  = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}

# Ssh access from any where
ingress {
  from_port = 22
  to_port   = 22
  protocol  = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}
#outbound Rules
#internet access to anywhere
egress {
  from_port = 0
  to_port   = 0
  protocol  = "-1"
  cidr_blocks = ["0.0.0.0/0"]
}
tags = {
  Name = "web-sg"
}
}

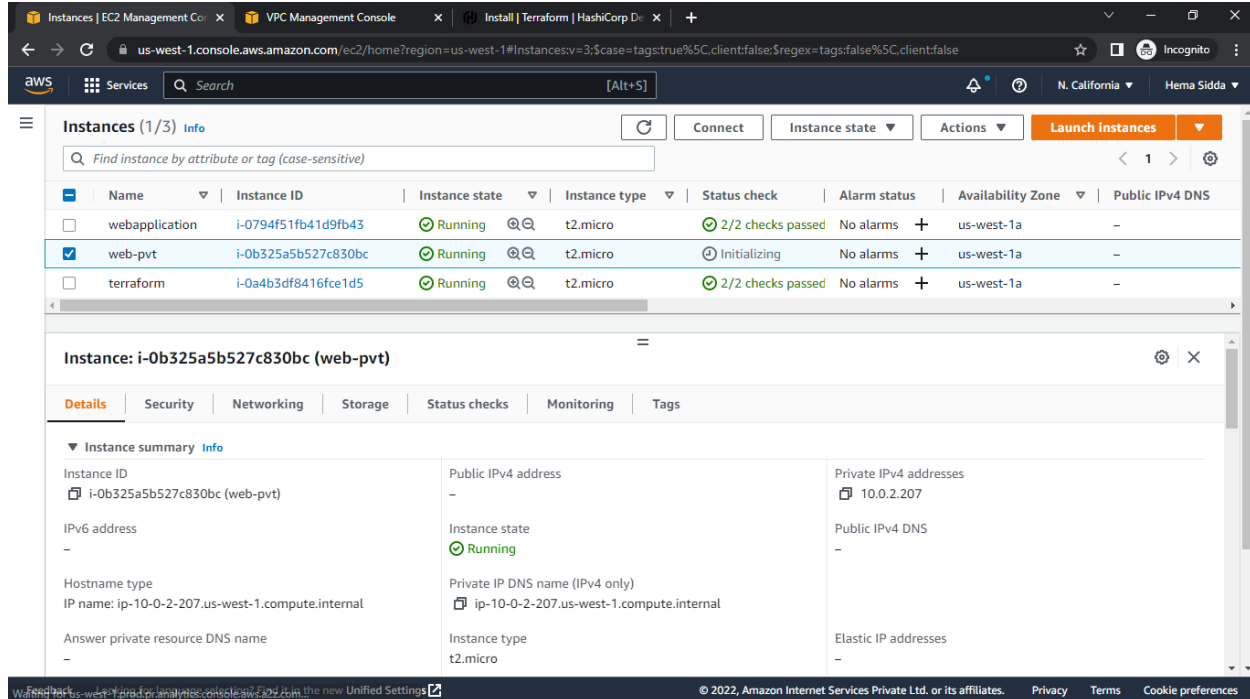
-- INSERT --
55,1 Bot
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~$ terraform apply
+ 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" = "web-sg"
+ }
+ tags_all       = {
+   + "Name" = "web-sg"
+ }
+ vpc_id         = "vpc-0f68094eb476f11c1"
}

Plan: 2 to add, 1 to change, 0 to destroy.
aws_security_group.demo: Creating...
aws_route_table.pvt-route: Modifying... [id=rtb-042f33f7386779a4a]
aws_route_table.pvt-route: Modifications complete after 1s [id=rtb-042f33f7386779a4a]
aws_security_group.demo: Creation complete after 2s [id=sg-051056bffa830437]
aws_instance.web-app[0]: Creating...
aws_instance.web-app[0]: Still creating... [10s elapsed]
aws_instance.web-app[0]: Still creating... [20s elapsed]
aws_instance.web-app[0]: Still creating... [30s elapsed]
aws_instance.web-app[0]: Creation complete after 31s [id=i-0b325a5b527c830bc]

Apply complete! Resources: 2 added, 1 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$
```



DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

- Now create the RDS(relational database) service by using yamI scripting.

```
ec2-user@ip-172-30-0-178:~$
# Create subnet groups
resource "aws_db_subnet_group" "db-sub" {
  subnet_ids = [aws_subnet.pvt-sub-1.id, aws_subnet.pvt-sub-2.id]
  tags = {
    Name = "dbsubnet"
  }
}

# Create the data-base
resource "aws_db_instance" "data" {
  allocated_storage = "10"
  db_subnet_group_name = aws_db_subnet_group.db-sub.id
  db_name = "mydata"
  engine = "mysql"
  engine_version = "8.0.28"
  instance_class = "db.t2.micro"
  multi_az = "true"
  username = "admin"
  password = "admin1230"
  vpc_security_group_ids = ["${aws_security_group.db-sg.id}"]
  skip_final_snapshot = "true"
}

# Create the security groups
resource "aws_security_group" "db-sg" {
  vpc_id = aws_vpc.mvpc.id

  ingress {
    from_port = 3306
    to_port = 3306
    protocol = "tcp"
    security_groups = [aws_security_group.demo.id]
  }
}

-- INSERT --
```

```
ec2-user@ip-172-30-0-178:~$
  engine = "mysql"
  engine_version = "8.0.28"
  instance_class = "db.t2.micro"
  multi_az = "true"
  username = "admin"
  password = "admin1230"
  vpc_security_group_ids = ["${aws_security_group.db-sg.id}"]
  skip_final_snapshot = "true"
}

# Create the security groups
resource "aws_security_group" "db-sg" {
  vpc_id = aws_vpc.mvpc.id

  ingress {
    from_port = 3306
    to_port = 3306
    protocol = "tcp"
    security_groups = [aws_security_group.demo.id]
  }

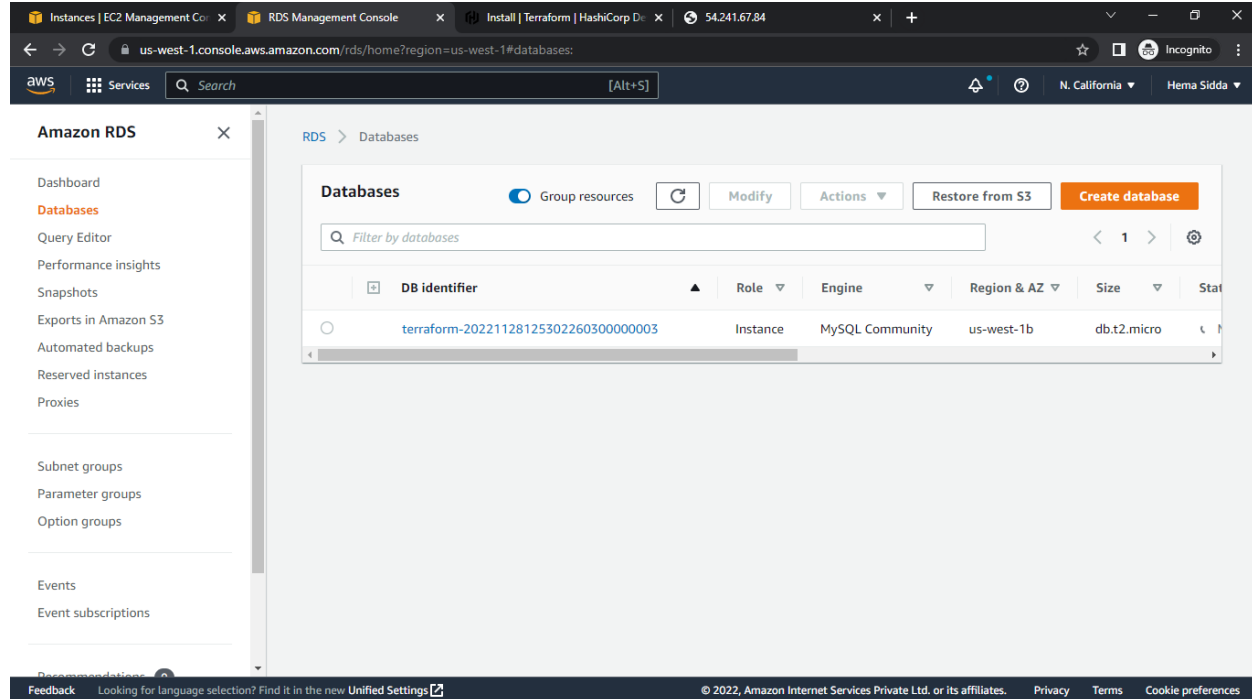
  egress {
    from_port = 32768
    to_port = 65535
    protocol = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "database-sg"
  }
}

-- INSERT --
```

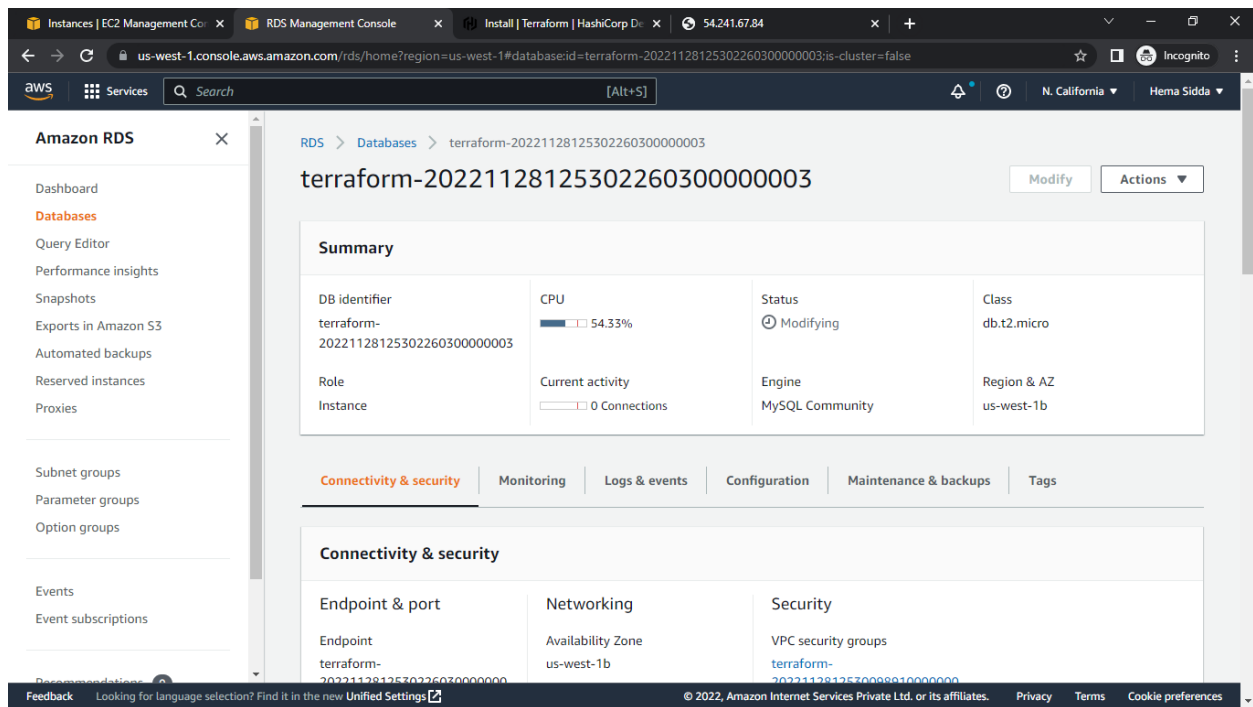
DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

```
ec2-user@ip-172-30-0-178:~$ terraform apply
aws_db_instance.data: Still creating... [5m20s elapsed]
aws_db_instance.data: Still creating... [5m30s elapsed]
aws_db_instance.data: Still creating... [5m40s elapsed]
aws_db_instance.data: Still creating... [5m50s elapsed]
aws_db_instance.data: Still creating... [6m0s elapsed]
aws_db_instance.data: Still creating... [6m10s elapsed]
aws_db_instance.data: Still creating... [6m20s elapsed]
aws_db_instance.data: Still creating... [6m30s elapsed]
aws_db_instance.data: Still creating... [6m40s elapsed]
aws_db_instance.data: Still creating... [6m50s elapsed]
aws_db_instance.data: Still creating... [7m0s elapsed]
aws_db_instance.data: Still creating... [7m10s elapsed]
aws_db_instance.data: Still creating... [7m20s elapsed]
aws_db_instance.data: Still creating... [7m30s elapsed]
aws_db_instance.data: Still creating... [7m40s elapsed]
aws_db_instance.data: Still creating... [7m50s elapsed]
aws_db_instance.data: Still creating... [8m0s elapsed]
aws_db_instance.data: Still creating... [8m10s elapsed]
aws_db_instance.data: Still creating... [8m20s elapsed]
aws_db_instance.data: Still creating... [8m30s elapsed]
aws_db_instance.data: Still creating... [8m40s elapsed]
aws_db_instance.data: Still creating... [8m50s elapsed]
aws_db_instance.data: Still creating... [9m0s elapsed]
aws_db_instance.data: Still creating... [9m10s elapsed]
aws_db_instance.data: Still creating... [9m20s elapsed]
aws_db_instance.data: Still creating... [9m30s elapsed]
aws_db_instance.data: Still creating... [9m40s elapsed]
aws_db_instance.data: Still creating... [9m50s elapsed]
aws_db_instance.data: Still creating... [10m0s elapsed]
aws_db_instance.data: Still creating... [10m10s elapsed]
aws_db_instance.data: Still creating... [10m20s elapsed]
aws_db_instance.data: Creation complete after 10m25s [id=terraform-20221128125302260300000003]

Apply complete! Resources: 3 added, 1 changed, 0 destroyed.
[ec2-user@ip-172-30-0-178 ~]$
```



DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM



➤ Now create the load-balancer by using yaml script.

```
ec2-user@ip-172-30-0-8:~$ cat load_balancer.yaml
# Create load_balancer
resource "aws_lb" "application" {
  internal        = false
  name            = "APPLI-LB"
  load_balancer_type = "application"
  security_groups = [aws_security_group.sg.id]
  subnets        = [aws_subnet.pub-sub-1.id, aws_subnet.pub-sub-2.id]
}

resource "aws_lb_target_group" "tar" {
  name        = "target"
  port        = 80
  protocol    = "HTTP"
  vpc_id      = aws_vpc.mvpc.id
  health_check {
    healthy_threshold   = 2
    unhealthy_threshold = 2
    timeout              = 3
    interval             = 30
    protocol             = "HTTP"
    port                 = 80
    path                 = "/ping"
    matcher              = 200
  }
}

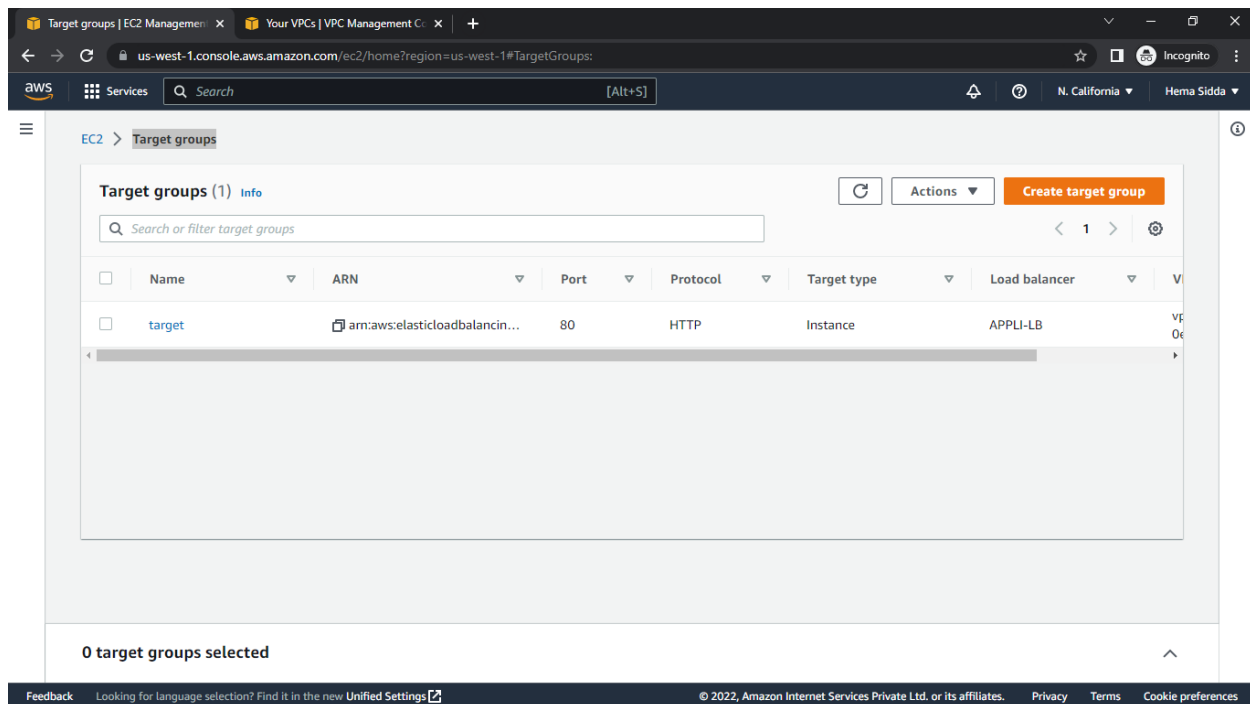
resource "aws_lb_target_group_attachment" "att" {
  target_group_arn = aws_lb_target_group.tar.arn
  count            = 0
  target_id        = "aws_instance.app2.id[count.index]"
  port             = 80
  depends_on       = [aws_instance.app,]
}

resource "aws_lb_target_group_attachment" "att2" {
  target_group_arn = aws_lb_target_group.tar.arn
  count            = 0
  target_id        = "aws_instance.app2.id[count.index]"
  port             = 80
  depends_on       = [aws_instance.app,]
}

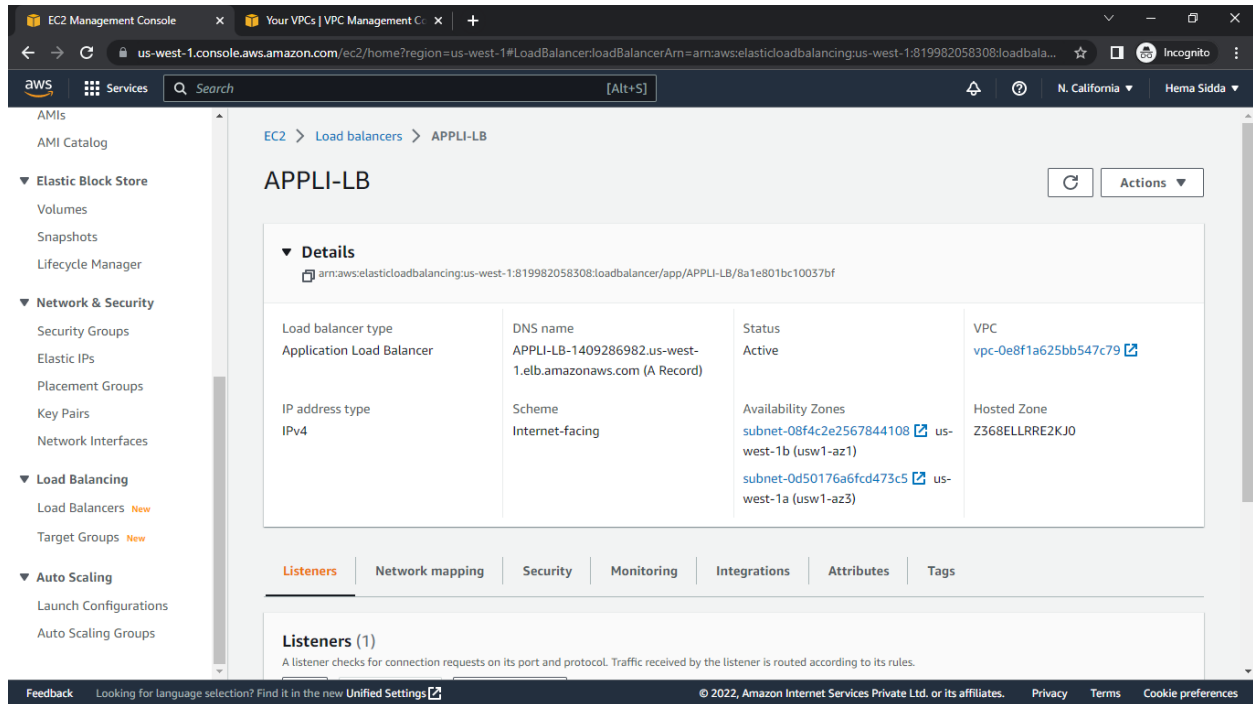
-- INSERT --
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

```
ec2-user@ip-172-30-0-8:~$ cat terraform.tf
matcher = 200
}
}
resource "aws_lb_target_group_attachment" "att" {
  target_group_arn = aws_lb_target_group.tar.arn
  count            = 0
  target_id       = "aws_instance.app2.id[count.index]"
  port            = 80
  depends_on      = [aws_instance.app,]
}
resource "aws_lb_target_group_attachment" "att2" {
  target_group_arn = aws_lb_target_group.tar.arn
  count            = 0
  target_id       = "aws_instance.app2.id[count.index]"
  port            = 80
  depends_on      = [aws_instance.app2,]
}
resource "aws_lb_listener" "lb-lis" {
  load_balancer_arn = aws_lb.application.arn
  port              = 80
  protocol          = "HTTP"
  default_action {
    type = "forward"
    target_group_arn = aws_lb_target_group.tar.arn
  }
}
#getting the DNS of load-balancer
output "lb_dns_name" {
  description = "the name of the loadbalancer"
  value       = "${aws_lb.application.dns_name}"
}
-- INSERT --
```



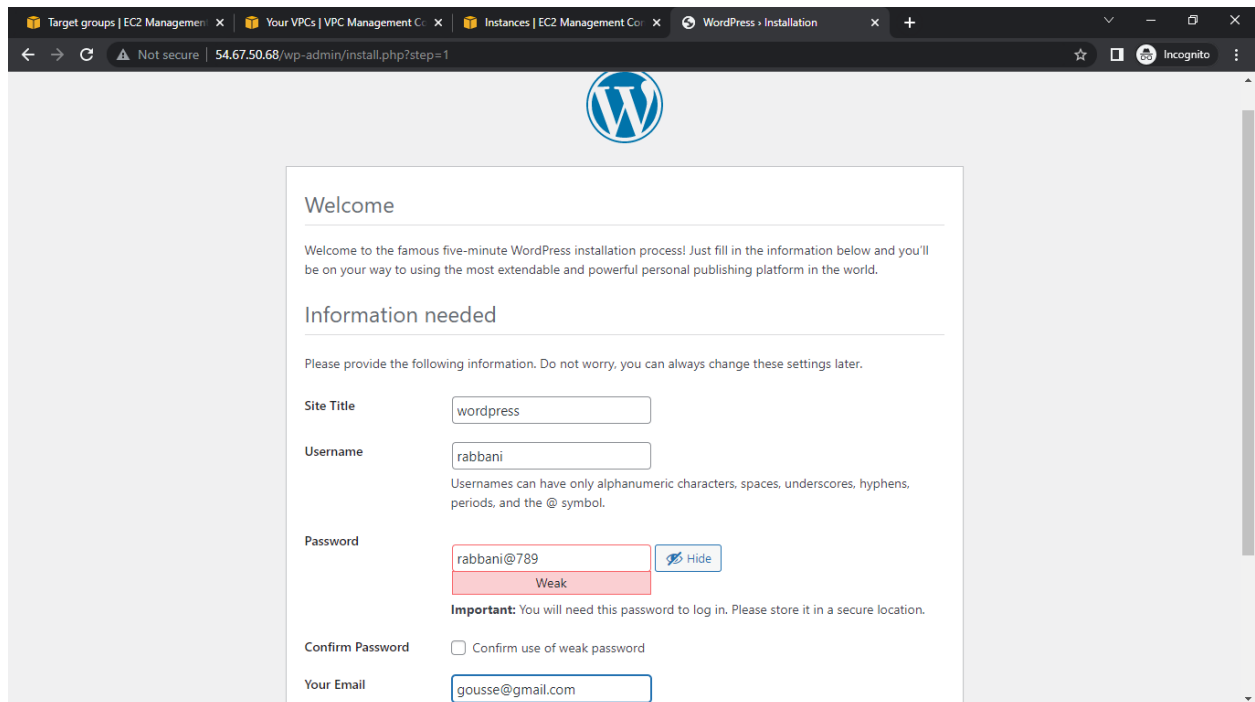
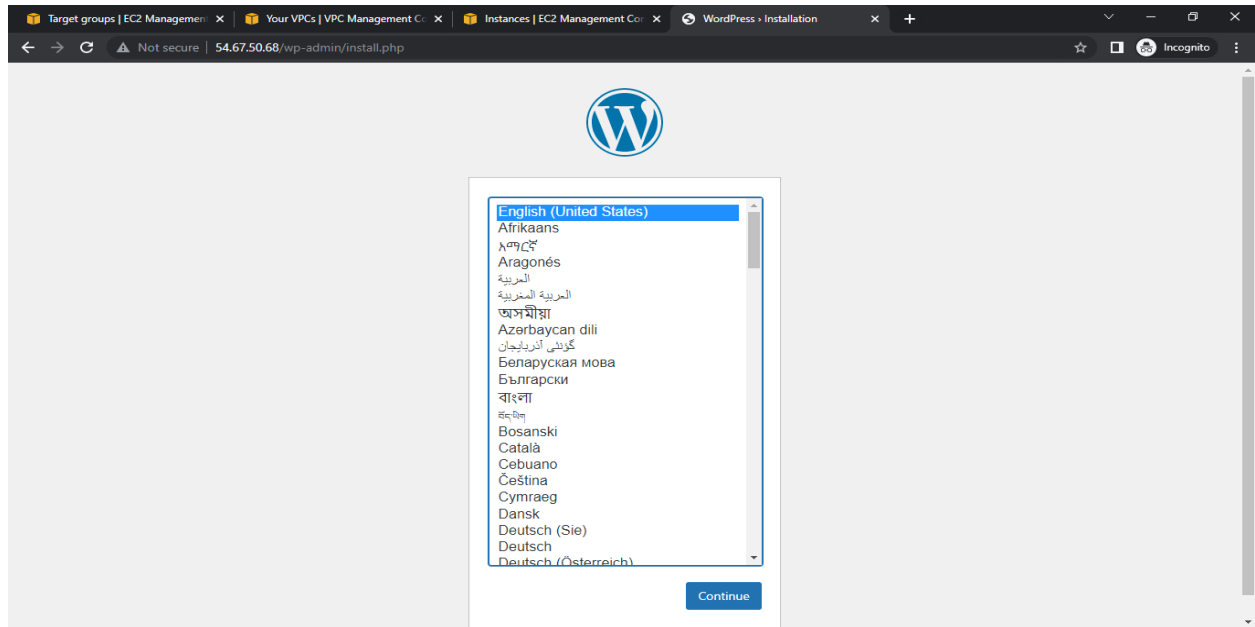
DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM



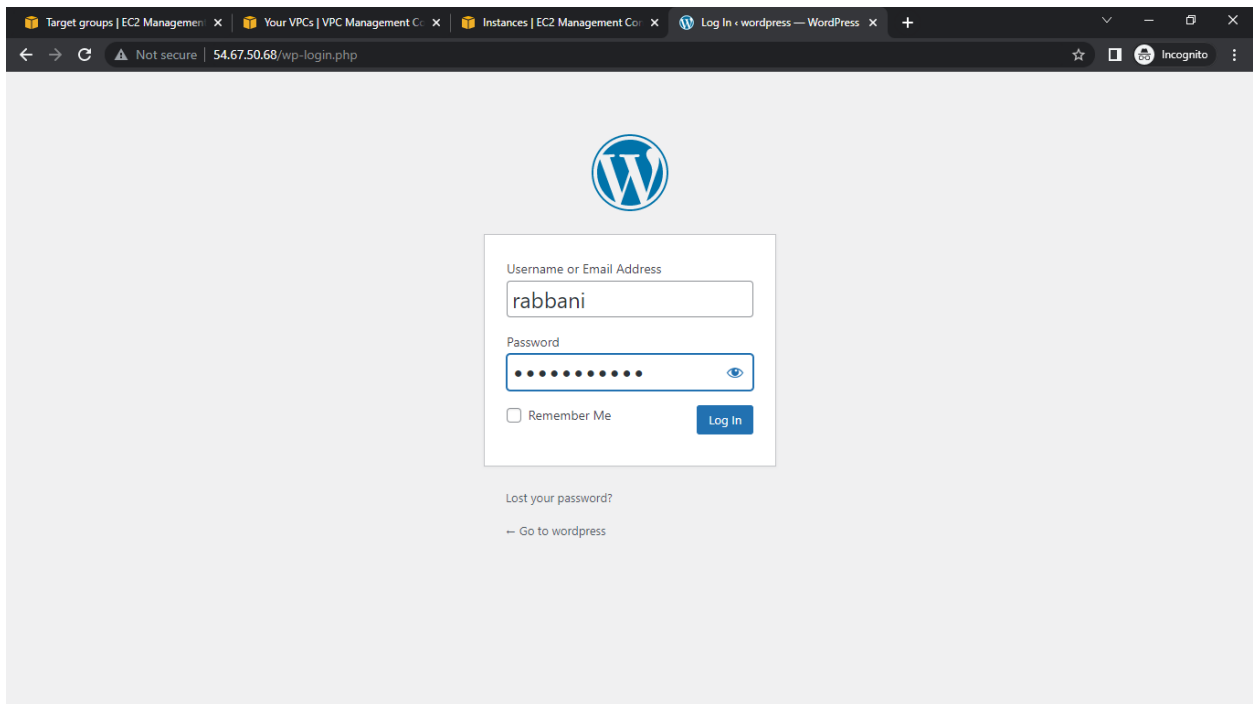
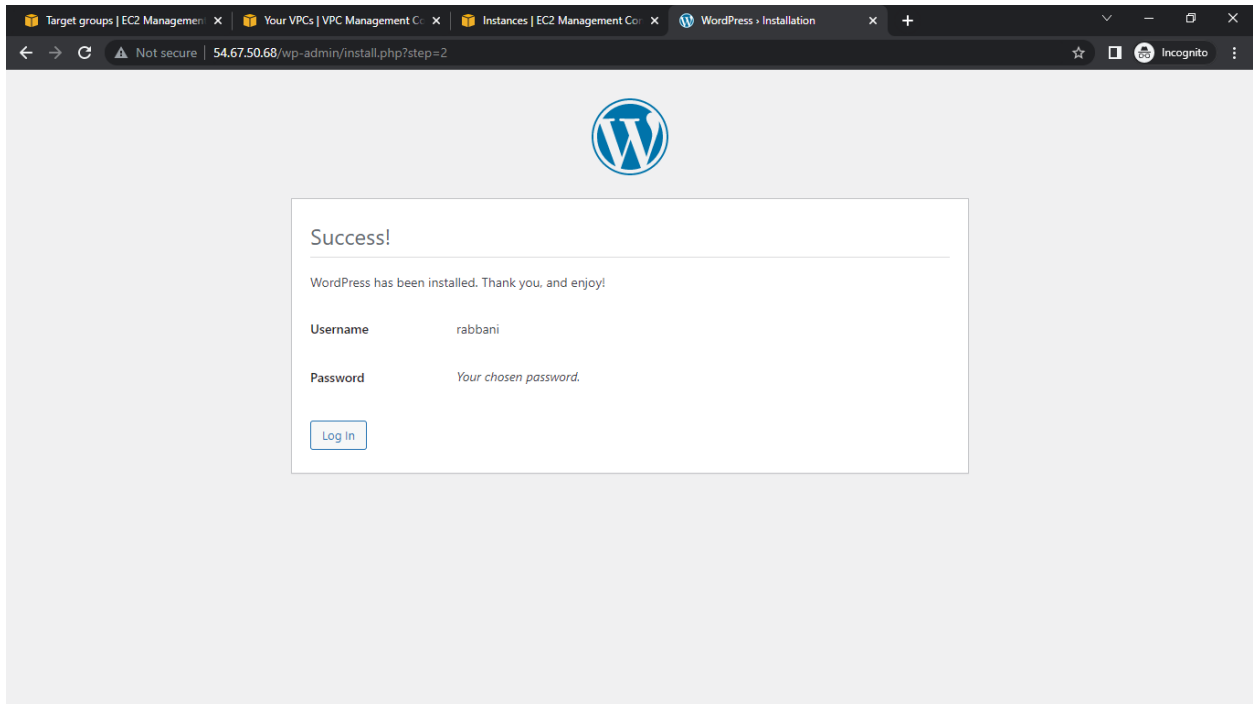
```
ec2-user@ip-172-30-0-8~  
aws_db_instance.data: Still creating... [7m20s elapsed]  
aws_db_instance.data: Still creating... [7m30s elapsed]  
aws_db_instance.data: Still creating... [7m40s elapsed]  
aws_db_instance.data: Still creating... [7m50s elapsed]  
aws_db_instance.data: Still creating... [8m0s elapsed]  
  
aws_db_instance.data: Still creating... [8m10s elapsed]  
aws_db_instance.data: Still creating... [8m20s elapsed]  
aws_db_instance.data: Still creating... [8m30s elapsed]  
aws_db_instance.data: Still creating... [8m40s elapsed]  
aws_db_instance.data: Still creating... [8m50s elapsed]  
aws_db_instance.data: Still creating... [9m0s elapsed]  
aws_db_instance.data: Still creating... [9m10s elapsed]  
aws_db_instance.data: Still creating... [9m20s elapsed]  
aws_db_instance.data: Still creating... [9m30s elapsed]  
aws_db_instance.data: Still creating... [9m40s elapsed]  
aws_db_instance.data: Still creating... [9m50s elapsed]  
aws_db_instance.data: Still creating... [10m0s elapsed]  
aws_db_instance.data: Creation complete after 10m6s [id=terraform-20221130125933895100000007]  
  
Apply complete! Resources: 26 added, 0 changed, 0 destroyed.  
  
Outputs:  
  
lb_dns_name = "APPLI-LB-1409286982.us-west-1.elb.amazonaws.com"  
[ec2-user@ip-172-30-0-8 ~]$  
[ec2-user@ip-172-30-0-8 ~]$  
[ec2-user@ip-172-30-0-8 ~]$  
[ec2-user@ip-172-30-0-8 ~]$  
[ec2-user@ip-172-30-0-8 ~]$  
[ec2-user@ip-172-30-0-8 ~]$
```

DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM

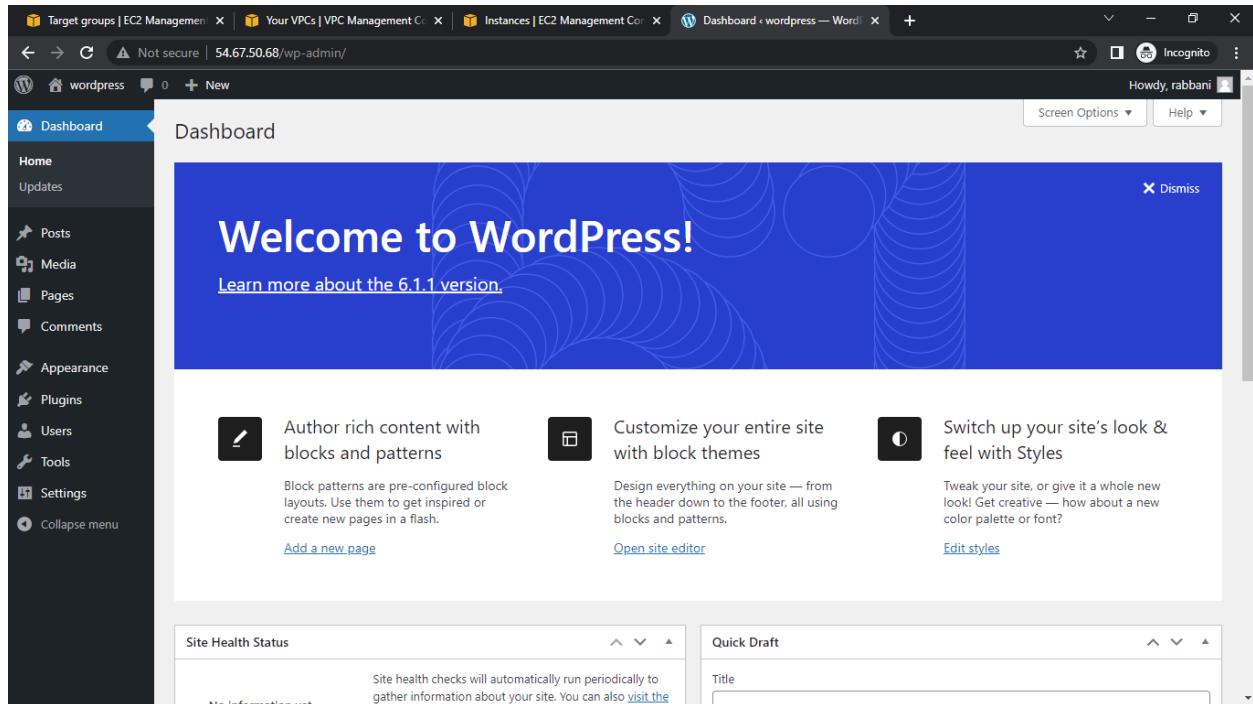
- Now go to EC2 instances and select the public ip(ipv4) and browse it on Google.



DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM



DEPLOY THREE-TIER ARCHITECTURE IN AWS BY USING TERRAFORM



- Now copy the DNS of load-balancer browse in Google and see the result.