

## TERRAFORM PROJECT

### AWS INFRASTRUCTURE AUTOMATION USING TERRAFORM

#### **TERRAFORM :**

Terraform is an open-source Infrastructure as Code (IaC) tool developed by HashiCorp. It allows you to define and manage your infrastructure (servers, databases, networking, etc.) using a declarative configuration language. With Terraform, you can automate the provisioning, scaling, and management of resources across multiple cloud providers (like AWS, Azure, Google Cloud) and on-premises environments.

#### **Simple Use Cases of Terraform:**

##### **➤ Cloud Infrastructure Management:**

Automate the setup of servers, databases, networks, and storage across cloud providers like AWS, Azure, or GCP.

##### **➤ Multi-Cloud Deployments:**

Manage resources in multiple cloud platforms simultaneously, such as hosting an app on AWS while storing data in Azure.

##### **➤ Auto-Scaling Applications:**

Set up rules to automatically scale your servers up or down based on demand, ensuring performance and cost-efficiency.

##### **➤ Infrastructure as Code (Iac):**

Write code to define your infrastructure, making it easy to replicate environments (e.g., dev, test, prod) consistently.

##### **➤ Resource Management:**

Terraform excels at resource management, allowing users to create, update, and delete infrastructure resources programmatically and consistently.

#### **Terraform Commands:**

##### **❖ Terraform init:**

Terraform initialization downloads the required plugins for the providers defined in the configuration files. Without this step Terraform wouldn't know how to interact with the desired infrastructure platform.

##### **❖ Terraform validate:**

It ensures that the files are syntactically correct. This includes checking for common mistakes like missing braces or incorrect argument formats.

❖ **Terraform plan (terraform plan -lock=false):**

It helps to ensure that the changes you intend to make to your infrastructure are accurate and safe before applying them.

❖ **Terraform apply:**

It is used to apply the changes to your infrastructure like creating, updating, or destroying resources as specified in our configuration files.

❖ **Terraform destroy:**

It is used to remove all the resources managed by terraform. It helps to delete resources that are no longer needed.

## PREREQUISITES:

- ✓ Basic knowledge of aws & terraform
- ✓ Aws account
- ✓ Iam user
- ✓ Github account
- ✓ Aws access & secret key

## Basic knowledge of aws & terraform

To create a files using terraform first we need to install terraform using the below commands

- sudo yum install -y yum-utils shadow-utils
- sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
- sudo yum -y install terraform

```
root@ip-172-31-18-121:~#
[ec2-user@ip-172-31-18-121 ~]$ sudo su
[ec2-user@ip-172-31-18-121 ~]# aws configure
AWS Access Key ID [None]: ARKASPTZDQCQWDXEQ4W
AWS Secret Access Key [None]: eN1Czrs4sYU0U8hX1xb7nhQBxRJzPM7mP3OpiKNT
Default region name [None]:
Default output format [None]:
[ec2-user@ip-172-31-18-121 ~]# ll
total 4
rv 2--r-- 1 root root 110 Nov 12 05:55 vpc.tf
[ec2-user@ip-172-31-18-121 ~]# terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.75.1

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,
re-run 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-172-31-18-121 ~]# terraform plan
Planning failed: Terraform encountered an error while generating this plan.

Error: Invalid provider configuration
Provider "registry.terraform.io/hashicorp/aws" requires explicit configuration. Add a provider block to the root module and configure the provider's required arguments as described in the provider documentation.

[ec2-user@ip-172-31-18-121 ~]#
```

## IAM USER

With the help of IAM user I have created access keys and secret access keys to provide security for my AWS account from unauthorized usage and unintended data loss.

The screenshot shows the AWS IAM User Access Keys page for the user 'Bhargavi-user'. It displays one active access key:

Access Key ID	Description	Status	Last used	Last used region	Last used service
AKIA5FTZD3CQWPDXEQ4W	creating access keys	Active	1 hour ago	us-east-1	ec2

Below this, there are sections for SSH public keys and HTTPS Git credentials, both currently empty.

## GITHUB ACCOUNT

I have created a github account to store the data and code related to the terraform, so that I can access the code immediately whenever I want to access.

The screenshot shows the GitHub repository 'Terraform-project' for the user 'bhargavibairagi'. The repository has 1 branch and 0 tags. The commit history shows the following commits:

Commit	Author	Date	
23bb99f	root committing	last week	
1	Terraform-project	committing	2 weeks ago
2	alb.tf	committing	last week
3	data.sh	committing	last week
4	database_sg.tf	committing	last week
5	ec2.tf	committing	2 weeks ago
6	igw.tf	committing	2 weeks ago
7	outputs.tf	committing	2 weeks ago
8	rds.tf	committing	last week
9	route_table_public.tf	committing	2 weeks ago
10	subnet.tf	committing	last week

The repository details section includes:

- About:** No description, website, or topics provided.
- Activity:** 0 stars, 1 watching, 0 forks.
- Releases:** No releases published. Create a new release.
- Packages:** No packages published. Publish your first package.
- Languages:** A progress bar indicating language usage.

## AWS ACCESS & SECRET KEY

- ✓ Access keys and secret access keys are necessary when using Terraform to manage AWS resources because they provide authentication and authorization for Terraform to interact with AWS on your behalf. AWS uses these credentials to verify that you have the appropriate permissions to create, modify, or delete resources in your account.
- ✓ Use <"aws configure"> to set access and secret keys.

## STEP 1: CREATE A FILE FOR VPC

A Virtual Private Cloud (VPC) is a logically isolated section of the AWS cloud where we can launch AWS resources in a virtual isolated network. Here I am creating a file for vpc with the name of “vpc.tf”, and with the specific cidr.

We need to configure the aws for authentication. AWS configuration provides Terraform with the necessary permissions to provision and manage these resources.

If we don't provide access key and secret access key terraform shows error while generating plan.

Here I did not provided any authentication so it is showing error. To resolve this I need to generate access keys using the user from IAM.

```
root@ip-172-31-80-26:~  
provider "aws" {  
    access_key = "AKIA5FTZD3CQWPDXEQ4W"  
    secret_key = "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"  
    region     = "us-east-1"  
}  
resource "aws_vpc" "demovpc" {  
    cidr_block="10.0.0.0/16"  
    instance_tenancy="default"  
    tags={  
        Name="DEMO VPC"  
    }  
}  
~
```

When using Terraform to create a Virtual Private Cloud (VPC) in AWS, you need to ensure proper authentication and configuration of the AWS provider. The process involves defining the VPC in a Terraform configuration file (vpc.tf) and authenticating Terraform with AWS to enable resource provisioning.

This Terraform configuration sets up a provider for AWS and creates a Virtual Private Cloud (VPC) within the AWS environment. Cidr block defines the IP range for the VPC. This allows for a maximum of 65,536 IP addresses

```

root@ip-172-31-18-121:~#
[root@ip-172-31-18-121 ~]# aws configure
AWS Access Key ID [*****]:*****SQW:
AWS Secret Access Key [*****]:*****1KNT:
Default region name [None]: us-east-1
Default output format [None]:
[root@ip-172-31-18-121 ~]# terraform plan

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_vpc.demovpc will be created
  + resource "aws_vpc" "demovpc" {
      + 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_dns_hostnames = "(known after apply)"
      + enable_dns_support = true
      + enable_network_address_usage_metrics = "(known after apply)"
      + id = "(known after apply)"
      + instance_tenancy = "default"
      + ipv4_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" = "DEMO VPC"
        }
      + tags_all = [
          + "Name" = "DEMO VPC"
        ]
    }

Plan: 1 to add, 0 to change, 0 to destroy.

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.
[root@ip-172-31-18-121 ~]# terraform apply

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_vpc.demovpc will be created
  + resource "aws_vpc" "demovpc" {
      + 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_dns_hostnames = "(known after apply)"
      + enable_dns_support = true
      + enable_network_address_usage_metrics = "(known after apply)"
      + id = "(known after apply)"
      + instance_tenancy = "default"
      + ipv4_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" = "DEMO VPC"
        }
      + tags_all = [
          + "Name" = "DEMO VPC"
        ]
    }

Plan: 1 to add, 0 to change, 0 to destroy.

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.

```

Plan for creating VPC is successful. Resource is used to create the service whatever we mention with the name. I have provided the resource with “aws\_vpc” ,”demovpc” is the name given to the vpc to reference our vpc while using it. For example if we want to create subnets from vpc we can use this reference name.

```

root@ip-172-31-18-121:~#
Apply cancelled.
[root@ip-172-31-18-121 ~]# ^C
[root@ip-172-31-18-121 ~]# terraform apply

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_vpc.demovpc will be created
  + resource "aws_vpc" "demovpc" {
      + 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_dns_hostnames = "(known after apply)"
      + enable_dns_support = true
      + enable_network_address_usage_metrics = "(known after apply)"
      + id = "(known after apply)"
      + instance_tenancy = "default"
      + ipv4_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" = "DEMO VPC"
        }
      + tags_all = [
          + "Name" = "DEMO VPC"
        ]
    }

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.demovpc: Creating...
aws_vpc.demovpc: Creation complete after 1s [id=vpc-0ee254eb03ed9ecc]
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-18-121 ~]#

```

After planning to create a vpc we need to apply the changes so that vpc or any services /infrastructure is created automatically. Here we need to run the command

“<terraform apply>”. It is showing whatever we mentioned in the code and there is no error, so it is asking for confirmation , we need to say “yes” if we want to create the vpc otherwise say “no”.

We can see DEMO VPC is created successfully in aws management console.

## STEP 2: CREATE A FILE FOR THE SUBNET

A subnet is a subdivision of the IP address range within a VPC. Subnets allow you to allocate smaller, more manageable address spaces for groups of resources.

- After creating vpc we need to create subnets, here am creating 6 subnets. Creating multiple subnets in a Virtual Private Cloud (VPC) allows you to organize and secure your resources based on different requirements, such as availability, security, and access control. Each subnet will be allocated a specific range of IP addresses, and you can deploy your resources (like EC2 instances) into these subnets.
- Subnets in AWS are defined by CIDR blocks, and we need to decide the size of each subnet. A common practice is to divide the VPC into subnets with /24 CIDR blocks, each providing 256 IP addresses (of which 251 are usable). You can adjust the sizes as needed, such as /25 or /23, depending on the number of IPs required per subnet.
- It's recommended to distribute your subnets across different Availability Zones (AZs) to ensure high availability and fault tolerance. AWS regions have multiple AZs (e.g., us-east-1a, us-east-1b, us-east-1c), and each subnet should be assigned to one AZ.

```

root@ip-172-31-80-26:~#
resource "aws_subnet" "public_subnet-1" {
  vpc_id = "${aws_vpc.demoVpc.id}"
  cidr_block = "10.0.1.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1a"
  tags = {
    Name = "Web Subnet 1"
  }
}
#creating public subnet
resource "aws_subnet" "public_subnet-2" {
  vpc_id = "${aws_vpc.demoVpc.id}"
  cidr_block = "10.0.2.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1b"
  tags = {
    Name = "WEB SUBNET2"
  }
}
#creating application subnet 1
resource "aws_subnet" "application-subnet-1" {
  vpc_id = "${aws_vpc.demoVpc.id}"
  cidr_block = "10.0.3.0/24"
  map_public_ip_on_launch = false
  availability_zone = "us-east-1a"
  tags = {
    Name = "Application Subnet 1"
  }
}
#creating application subnet
resource "aws_subnet" "application-subnet-2" {
  vpc_id = "${aws_vpc.demoVpc.id}"
  cidr_block = "10.0.4.0/24"
  map_public_ip_on_launch = false
  availability_zone = "us-east-1b"
  tags = {
    Name = "Application Subnet 2"
  }
}

resource "aws_subnet" "database-subnet-1" {
  vpc_id = "${aws_vpc.demoVpc.id}"
  cidr_block = "10.0.5.0/24"
  availability_zone = "us-east-1a"
  tags = {
    Name = "Database Subnet 1"
  }
}
-- INSERT --

```

```

resource "aws_subnet" "database-subnet-2" {
  vpc_id = "${aws_vpc.demoVpc.id}"
  cidr_block = "10.0.6.0/24"
  availability_zone = "us-east-1b"
  tags = {
    Name = "Database Subnet 2"
  }
}
-- INSERT --

```

This Terraform configuration defines **six subnets** within a Virtual Private Cloud (VPC) ,each subnet is configured with a CIDR block, availability zone, and attributes suitable for its purpose.

- ✓ Frontend Tier (Public Subnets): Accessible from the internet.

- ✓ Application Tier (Private Application Subnets): For internal processing, no direct internet access.
- ✓ Database Tier (Private Database Subnets): Fully private, isolated from the internet for security.

```

root@ip-172-31-18-121:~#
Error: Unsupported argument
  on subnet.tf line 4, in resource "aws_subnet" "public_subnet-1":
  4: map_ip_on_launch=true

An argument named "map_ip_on_launch" is not expected here.

[root@ip-172-31-18-121 ~]# vim subnet.tf
[root@ip-172-31-18-121 ~]# terraform plan
aws_vpc.demovpc: Refreshing state... [id=vpc-0ee254eb03ed9ecc]

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

Terraform will perform the following actions:

# aws_subnet.public_subnet-1 will be created
+ resource "aws_subnet" "public_subnet-1" {
    + arn                                = (Known after apply)
    + assign_ipv6_address_on_creation     = false
    + availability_zone                  = "us-east-1a"
    + availability_zone_id               = (Known after apply)
    + cidr_block                         = "10.0.1.0/24"
    + enable_dns64                      = false
    + enable_resource_name_dns_a_record_on_launch = false
    + id                                 = (Known after apply)
    + ipv6_cidr_block_association_id     = (Known after apply)
    + ipv6_native                        = false
    + map_public_ip_on_launch            = true
    + owner_id                           = (Known after apply)
    + private_dns_hostname_type_on_launch = (Known after apply)
    + tags                               = [
        + "Name" = "Web Subnet1"
    ]
    + tags_all                           = [
        + "Name" = "Web Subnet1"
    ]
    + vpc_id                             = "vpc-0ee254eb03ed9ecc"
}

Plan: 1 to add, 0 to change, 0 to destroy.

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.

[root@ip-172-31-18-121 ~]#

```

Creating plan for the subnet by using “terraform plan”. We can see what we are configured in the code and what is going to create in the first subnet.

```

root@ip-172-31-18-121:~#
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.
[root@ip-172-31-18-121 ~]# terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-0ee254eb03ed9ecc]

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

Terraform will perform the following actions:

# aws_subnet.public_subnet-1 will be created
+ resource "aws_subnet" "public_subnet-1" {
    + arn                                = (Known after apply)
    + assign_ipv6_address_on_creation     = false
    + availability_zone                  = "us-east-1a"
    + availability_zone_id               = (Known after apply)
    + cidr_block                         = "10.0.1.0/24"
    + enable_dns64                      = false
    + enable_resource_name_dns_a_record_on_launch = false
    + id                                 = (Known after apply)
    + ipv6_cidr_block_association_id     = (Known after apply)
    + ipv6_native                        = false
    + map_public_ip_on_launch            = true
    + owner_id                           = (Known after apply)
    + private_dns_hostname_type_on_launch = (Known after apply)
    + tags                               = [
        + "Name" = "Web Subnet1"
    ]
    + tags_all                           = [
        + "Name" = "Web Subnet1"
    ]
    + vpc_id                             = "vpc-0ee254eb03ed9ecc"
}

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_subnet.public_subnet-1: Creating...
aws_subnet.public_subnet-1: Still creating... [10s elapsed]
aws_subnet.public_subnet-1: Creation complete after 11s [id=subnet-04f4ad2c9d49e50a1]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-18-121 ~]#

```

After using “terraform apply” we can see subnet1 is creating.

The screenshot shows the AWS VPC dashboard with the Subnets page open. A new subnet named "Web Subnet1" has been created and is listed in the table. The subnet details are shown in a modal.

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 ...	IPv6 CIDR association ID	Available IPv4 addresses
-	subnet-01657ffad5375852e	Available	vpc-03f5f...	172.31.48.0/20	-	-	4091
-	subnet-0a0407bd5e37258df	Available	vpc-03f5f...	172.31.0.0/20	-	-	4091
-	subnet-0bf30e5b00722470	Available	vpc-03f5f...	172.31.64.0/20	-	-	4091
<b>Web Subnet1</b>	<b>subnet-04f4ad2c9d49e50a1</b>	<b>Available</b>	<b>vpc-0ee2...</b>	<b>10.0.1.0/24</b>	<b>-</b>	<b>-</b>	<b>251</b>
-	subnet-0b93ee5856f66287	Available	vpc-03f5f...	172.31.80.0/20	-	-	4091
-	subnet-0f875dfebb85c315	Available	vpc-03f5f...	172.31.16.0/20	-	-	4090
-	subnet-0db0d55fcf675b12	Available	vpc-03f5f...	172.31.32.0/20	-	-	4091

The first subnet with the name "Web Subnet1" is created

```

root@ip-172-31-18-121:~#
+ enable_dns64
+ enable_resource_name_dns_a_record_on_launch
enable_resource_name_dns_aaaa_record_on_launch = false
id = (known after apply)
ipv6_cidr_block_association_id = (known after apply)
ipv6_native
map_public_ip_on_launch
owner_id
private_dns_hostname_type_on_launch = (known after apply)
tags_all["Name"] = "Application Subnet 1"
tags_all["Name"] = "Application Subnet 1"
vpc_id = "vpc-0ee254eb03ed9ecc"
aws_subnet.public_subnet-2 will be created
resource "aws_subnet" "public_subnet-2" {
  arn = (known after apply)
  assign_ipv6_address_on_creation = false
  availability_zone = "us-east-1b"
  availability_zone_id = (known after apply)
  cidr_block = "10.0.2.0/24"
  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
  map_public_ip_on_launch = true
  owner_id = (known after apply)
  private_dns_hostname_type_on_launch = (known after apply)
  tags_all["Name"] = "WEB SUBNET2"
  tags_all["Name"] = "WEB SUBNET2"
  vpc_id = "vpc-0ee254eb03ed9ecc"
}

Plan: 2 to add, 0 to change, 0 to destroy.

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.
[root@ip-172-31-18-121:~]#

```

Here I have planned to create second subnet by writing code for second subnet.

```

root@ip-172-31-10-121:~#
  + "Name" = "Application Subnet 1"
  + tags.all
    + "Name" = "Application Subnet 1"
  + vpc_id
  |
  + aws_subnet.public_subnet-2 will be created
resource "aws_subnet" "public_subnet-2" {
  arn
  + assign_ipv6_address_on_creation
  + availability_zone
  + availability_zone_id
  + cidr_block
  + enable_dns64
  + enable_resource_name_dns_a_record_on_launch
  + enable_resource_name_dns_aaaa_record_on_launch
  + id
  + ipv6_cidr_block_association_id
  + ipv6_native
  + ipam_public_ip_on_launch
  + owner_id
  + private_dns_hostname_type_on_launch
  + tags
    + "Name" = "WEB SUBNET2"
  + tags.all
    + "Name" = "WEB SUBNET2"
  + vpc_id
}

Plan: 2 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_subnet.public_subnet-2: Creating...
aws_subnet.application_subnet-1: Creating...
aws_subnet.application_subnet-1: Creation complete after 0s [id=subnet-040a13a48dbd45a69]
aws_subnet.public_subnet-2: Still creating... [10s elapsed]
aws_subnet.public_subnet-2: Creation complete after 11s [id=subnet-0460e9c08f41b58f1]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
[root@ip-172-31-10-121 ~]# 

```

High UV Now 12:19 12-11-2024

I have created second subnet with the command “terraform apply”.

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 ...	IPv6 CIDR association ID	Available IPv4 addresses
-	subnet-01657ffa45375852e	Available	vpc-03f5f...	172.31.48.0/20	-	-	4091
Application Subnet 1	subnet-040a13a48dbd45a69	Available	vpc-0ee2...	10.0.3.0/24	-	-	251
-	subnet-040407bd5e37258df	Available	vpc-03f5f...	172.31.0.0/20	-	-	4091
-	subnet-0b303e5b00722470	Available	vpc-03f5f...	172.31.64.0/20	-	-	4091
Web Subnet1	subnet-0ff4ad29d19e50a1	Available	vpc-0ee2...	10.0.1.0/24	-	-	251
-	subnet-0b9c3ee5856f6287	Available	vpc-03f5f...	172.31.80.0/20	-	-	4091
-	subnet-0f875dbfebb85c315	Available	vpc-03f5f...	172.31.16.0/20	-	-	4090
-	subnet-0eb0455fcf675b12	Available	vpc-03f5f...	172.31.32.0/20	-	-	4091
<b>WEB SUBNET2</b>	<b>subnet-0460e9c08f41b58f1</b>	<b>Available</b>	<b>vpc-0ee2...</b>	<b>10.0.2.0/24</b>	<b>-</b>	<b>-</b>	<b>251</b>

**subnet-0460e9c08f41b58f1 / WEB SUBNET2**

**Details**

Subnet ID	subnet-0460e9c08f41b58f1	Subnet ARN	arn:aws:ec2:us-east-1:905418365089:subnet/subnet-0460e9c08f41b58f1	State	Available	IPv4 CIDR	10.0.2.0/24
Available IPv4 addresses	251	IPv6 CIDR	-	IPv6 CIDR association ID	-	Availability Zone	us-east-1b
Availability Zone ID	use1-az2	Network border group	ut-east-1	VPC	vpc-0ee254eb03ed9ecc   DEMO VPC	Route table	rtb-0ab2c3189dafe8639
Network ACL	-					Auto assign public IPv4 address	

We can see the second subnet in the aws management console.

**Subnets (1/3) Info**

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 ...	IPv6 CIDR association ID	Available IPv4 addresses
Application Subnet 1	subnet-040a13a48dbd45a69	Available	vpc-03f5f...	172.31.48.0/20	-	-	4091
	subnet-040a13a48dbd45a69	Available	vpc-0ee2...	10.0.3.0/24	-	-	251
	subnet-0a0407bd37258df	Available	vpc-03f5f...	172.31.0.0/20	-	-	4091
	subnet-0b303e5b00722470	Available	vpc-03f5f...	172.31.64.0/20	-	-	4091
WEB Subnet1	subnet-0f4fa42c9d19e50a1	Available	vpc-03f5f...	10.0.1.0/24	-	-	251
	subnet-0b93ed5856f6287	Available	vpc-03f5f...	172.31.80.0/20	-	-	4091
	subnet-0f875dbfeb85c315	Available	vpc-03f5f...	172.31.16.0/20	-	-	4090
	subnet-0bb0d55ffcf675b12	Available	vpc-03f5f...	172.31.32.0/20	-	-	4091
	WEB SUBNET2	Available	vpc-0ee2...	10.0.2.0/24	-	-	251

**subnet-040a13a48dbd45a69 / Application Subnet 1**

**Details**

Subnet ID	subnet-040a13a48dbd45a69	Subnet ARN	arn:aws:ec2:us-east-1:905418365089:subnet/subnet-040a13a48dbd45a69	State	Available	IPv4 CIDR	10.0.3.0/24
Available IPv4 addresses	251	IPv6 CIDR	-	IPv6 CIDR association ID	-	Availability Zone	us-east-1a
Availability Zone ID	use1-az1	Network border group	us-east-1	VPC	vpc-0ee254eb03ed9ecc   DEMO VPC	Route table	rtb-0ab2c3189dafe8639
Network ACL	-	-	-	Auto assign public IPv4 address	-	Auto assign IPv6 address	-

Created third subnet with the name of “application subnet 1”.

```

root@ip-172-31-18-121:~#
[root@ip-172-31-18-121 ~]# vim subnet.tf
[root@ip-172-31-18-121 ~]# terraform plan
Error: Unsupported block type

  on subnet.tf line 37, in resource "aws_subnet" "application-subnet-2":
  37: tags{
Blocks of type "tags" are not expected here. Did you mean to define argument "tags"? If so, use the equals sign to assign it a value.
[root@ip-172-31-18-121 ~]# vim subnet.tf
[root@ip-172-31-18-121 ~]# terraform plan
aws_vpc.demovpc: Refreshing state... [id=vpc-0ee254eb03ed9ecc]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-040a13a48dbd45a69]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-0460e9c08f41b50ff]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-04f4ad2c9d49e50a1]

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

Terraform will perform the following actions:

# aws_subnet.application-subnet-2 will be created
resource "aws_subnet" "application-subnet-2" {
    + arn
    + assign_ipv6_address_on_creation
    + availability_zone
    + availability_zone_id
    + cidr_block
    + enable_dns64
    + enable_resource_name_dns_a_record_on_launch
    + enable_resource_name_dns_aaaa_record_on_launch
    id
    ipv6_cidr_block_association_id
    ipv6_native
    map_public_ip_on_launch
    owner_id
    private_dns_hostname_type_on_launch
    tags
    + "Name" = "Application Subnet 2"
    + tags_all
    + "Name" = "Application Subnet 2"
    + vpc_id
}

```

Plan: 1 to add, 0 to change, 0 to destroy.

Creating fourth subnet with the help of code and used “terraform plan” to see the configurations that are going to create in the aws console are showing in the terminal.

```

root@ip-172-31-18-121:~#
[root@ip-172-31-18-121 ~]# terraform apply
aws_vpc: Refreshing state... [id=vpc-0ee254eb03ed9ecc]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-04f4ad2c9d94e50a]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-04f4ad2c9d94e50a]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-0460e9c08f41b58f]

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

Terraform will perform the following actions:

# aws_subnet.application-subnet-2 will be created
+ resource "aws_subnet" "application-subnet-2" (
  + arn
  + assign_ipv6_address_on_creation
  + availability_zone
  + availability_zone_id
  + cidr_block
  + enable_dns_64
  + enable_resource_name_dns_a_record_on_launch
  + enable_resource_name_dns_aaaa_record_on_launch
  + id
  + ipv6_cidr_block_association_id
  + ipv6_native
  + map_public_ip_on_launch
  + owner_id
  + private_dns_hostname_type_on_launch
  + tags
    + "Name" = "Application Subnet 2"
  )
+ tags all
  + "Name" = "Application Subnet 2"
)
+ vpc_id

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_subnet.application-subnet-2: Creating...
aws_subnet.application-subnet-2: Creation complete after 1s [id=subnet-0dd23b41ff07368c4]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-18-121 ~]# 

```

I have used “terraform apply” to create fourth subnet with the specified configurations.

Name	Subnet ID	State	IPv4 CIDR	IPv6 CIDR association ID	Available IPv4 address	
Application Subnet 1	subnet-040a13a48dhd45a59	Available	vpc-0ee2...	10.0.0.0/24	-	251
-	subnet-0a0407bd5e37258df	Available	vpc-03f5f...	172.31.0.0/20	-	4091
-	subnet-0bf303e5b00722470	Available	vpc-03f5f...	172.31.64.0/20	-	4091
Web Subnet1	subnet-04f4ad2c9d94e50a	Available	vpc-03f5f...	10.0.1.0/24	-	251
-	subnet-0bf9c1ee5856f66287	Available	vpc-03f5f...	172.31.80.0/20	-	4091
-	subnet-0f857d1fbefb85c315	Available	vpc-03f5f...	172.31.16.0/20	-	4090
-	subnet-0eb0d155ffff679b12	Available	vpc-03f5f...	172.31.32.0/20	-	4091
WEB SUBNET2	subnet-0460e9c08f41b58f1	Available	vpc-0ee2...	10.0.2.0/24	-	251
<b>Application Subnet 2</b>	<b>subnet-0dd23b41ff07368c4</b>	<b>Available</b>	<b>vpc-0ee2...</b>	<b>10.0.4.0/24</b>	<b>-</b>	<b>251</b>

**subnet-0dd23b41ff07368c4 / Application Subnet**

**Details**

Subnet ID: subnet-0dd23b41ff07368c4  
 Available IPv4 addresses: 251  
 Availability Zone ID: us-east-1a2  
 Network ACL: n/a

Subnet ARN: arn:aws:vpc:us-east-1:905418365089:subnet/subnet-0dd23b41ff07368c4  
 IPv6 CIDR: -  
 Network border group: un-east-1  
 VPC: vpc-0ee254eb03ed9ecc | DEMO VPC  
 Auto-assign public IPv4 address: Yes

Fourth subnet with the name of “Application Subnet 2” is created successfully with the defined cidr block.

```

root@ip-172-31-18-121:~#
An argument named "vpc_ip" is not expected here. Did you mean "vpc_id"?
[root@ip-172-31-18-121 ~]# vim subnet.tf
[root@ip-172-31-18-121 ~]# terraform plan
aws_vpc.demovpc: Refreshing state... [id=vpc-0ee254eb03ed9ecc]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-040a13a48dbd45a69]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-0dd23b41ff07368c4]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-04f4ad2c9d49e50a]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-0460e9c08f41b58f1]

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

Terraform will perform the following actions:

# aws_subnet.database-subnet-1 will be created
resource "aws_subnet" "database-subnet-1" {
    arn = (known after apply)
    assign_ipv6_address_on_creation = false
    availability_zone = "us-east-1a"
    availability_zone_id = (known after apply)
    cidr_block = "10.0.5.0/24"
    enable_dns64 = false
    enable_resource_name_dns_a_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" = "Database Subnet 1"
    ]
    tags_all = {
        + "Name" = "Database Subnet 1"
    }
    + vpc_id = "vpc-0ee254eb03ed9ecc"
}

Plan: 1 to add, 0 to change, 0 to destroy.

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.
[root@ip-172-31-18-121 ~]#
```

Creating a new subnet with different cidr 10.0.5.0/24, the name of the subnet is “database-subnet-1”.

```

root@ip-172-31-18-121:~#
aws_vpc.demovpc: Refreshing state... [id=vpc-0ee254eb03ed9ecc]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-04f4ad2c9d49e50a]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-0460e9c08f41b58f1]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-040a13a48dbd45a69]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-0dd23b41ff07368c4]

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

Terraform will perform the following actions:

# aws_subnet.database-subnet-1 will be created
resource "aws_subnet" "database-subnet-1" {
    arn = (known after apply)
    assign_ipv6_address_on_creation = false
    availability_zone = "us-east-1a"
    availability_zone_id = (known after apply)
    cidr_block = "10.0.5.0/24"
    enable_dns64 = false
    enable_resource_name_dns_a_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" = "Database Subnet 1"
    ]
    tags_all = {
        + "Name" = "Database Subnet 1"
    }
    + vpc_id = "vpc-0ee254eb03ed9ecc"
}

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_subnet.database-subnet-1: Creating...
aws_subnet.database-subnet-1: Creation complete after 1s [id=subnet-0c53d1c5497bf626e]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-18-121 ~]#
```

To apply the above mentioned configurations we need use terraform apply, then the subnet is going to create. We can see the creation of the subnet is successful.

**VPC dashboard**

**Subnets (1/11)**

Name	Subnet ID	State	IPv4 CIDR	IPv6 ...	IPv6 CIDR association ID	Available IPv4 address
Application Subnet 1	subnet-040a13a48bd5a5a9	Available	vpc-0ee2...	10.0.3.0/24	-	-
<b>Database Subnet 1</b>	<b>subnet-0c53d1c5497bf626e</b>	<b>Available</b>	<b>vpc-0ee2...</b>	<b>10.0.5.0/24</b>	-	<b>4091</b>
-	subnet-0d407bd5e57758df	Available	vpc-09f5f...	172.31.48.0/20	-	-
-	subnet-0f303eb0722470	Available	vpc-09f5f...	172.31.64.0/20	-	-
Web Subnet1	subnet-0af4ad2c9e49e50a1	Available	vpc-0ee2...	10.0.1.0/24	-	-
-	subnet-0b9c9ee58566f6287	Available	vpc-09f5f...	172.31.80.0/20	-	-
-	subnet-0b75dbfebb85c315	Available	vpc-09f5f...	172.31.16.0/20	-	-
-	subnet-0eb055ff4f759b12	Available	vpc-09f5f...	172.31.32.0/20	-	-

**subnet-0c53d1c5497bf626e / Database Subnet 1**

**Details**

Subnet ID	subnet-0c53d1c5497bf626e	Subnet ARN	arn:aws:vpc:us-east-1:1905418365089:subnet/subnet-0c53d1c5497bf626e	State	Available
Available IPv4 addresses	251	IPv6 CIDR	-	IPv6 CIDR association ID	-
Availability Zone ID	use1-az1	Network border group	us-east-1	VPC	vpc-0ee254eb03ed9eccc   DEMO VPC
Network ACL	-	Auto assign public IPv4 address	-	Route table	rta-bab2c5189daef8639

Database Subnet 1 is created in aws management console.

```

root@ip-172-31-80-26:~#
aws_subnet.database-subnet-1: Creation complete after 0s [id=subnet-0e27c8fb172c55ff1]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-80-26 ~]# vim subnet.tf
[root@ip-172-31-80-26 ~]# terraform plan
aws_vpc.demovpc: Refreshing state... [id=vpc-0d6bf814a7ba4f881]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0e27c8fb172c55ff1]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-020c456a22740b763]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-08c496538a16858d0]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-059c00b76febe80d]

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

Terraform will perform the following actions:

# aws_subnet.database-subnet-2 will be created
resource "aws_subnet" "database-subnet-2" {
    + arn
    + assign_ipv6_address_on_creation
    + availability_zone
    + availability_zone_id
    + cidr_block
    + enable_dns64
    + enable_resource_name_dns_a_record_on_launch
    + enable_resource_name_dns_aaa_record_on_launch
    + id
    + ipv6_cidr_block_association_id
    + ipv6_native
    + map_public_ip_on_launch
    + owner_id
    + private_dns_hostname_type_on_launch
    + tags
        + "Name" = "Database Subnet 2"
    }
    + tags_all
        + "Name" = "Database Subnet 2"
    }
    + vpc_id
        = "vpc-0d6bf814a7ba4f881"
}

Plan: 1 to add, 0 to change, 0 to destroy.

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.
[root@ip-172-31-80-26 ~]# terraform apply

```

Creating 6<sup>th</sup> subnet with availability zone us-east-1b, cidr block of 10.0.6.0/24. The Terraform plan command executed successfully with no mistake in the code.

```

root@ip-172-31-80-26:~#
aws subnet.database-subnet-1: Refreshing state... [id=subnet-0e27c0fb172c55ff1]
aws subnet.application-subnet-1: Refreshing state... [id=subnet-05cc00b76febeb0d]
aws subnet.application-subnet-2: Refreshing state... [id=subnet-08c496538a1e6858d0]
aws subnet.public_subnet-2: Refreshing state... [id=subnet-020c456a22740b763]
aws subnet.public_subnet-1: Refreshing state... [id=subnet-02b6448acf8835e18]

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

Terraform will perform the following actions:

# aws_subnet.database-subnet-2 will be created
+ resource "aws_subnet" "database-subnet-2" {
    arn                                = (known after apply)
    assign_ipv6_address_on_creation     = false
    availability_zone                   = "us-east-1b"
    availability_zone_id               = (known after apply)
    cidr_block                         = "10.0.6.0/24"
    enable_dns64                      = 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
    name_id                            = (known after apply)
    private_dns_hostname_type_on_launch = (known after apply)
    tags                               = [
        + "Name" = "Database Subnet 2"
    ]
    tags_all                           = [
        + "Name" = "Database Subnet 2"
    ]
    vpc_id                             = "vpc-0d6bf814a7ba4f881"
}

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_subnet.database-subnet-2: Creating...
aws_subnet.database-subnet-2: Creation complete after 0s [id=subnet-07bd0013517ea3edb]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-80-26 ~]#

```

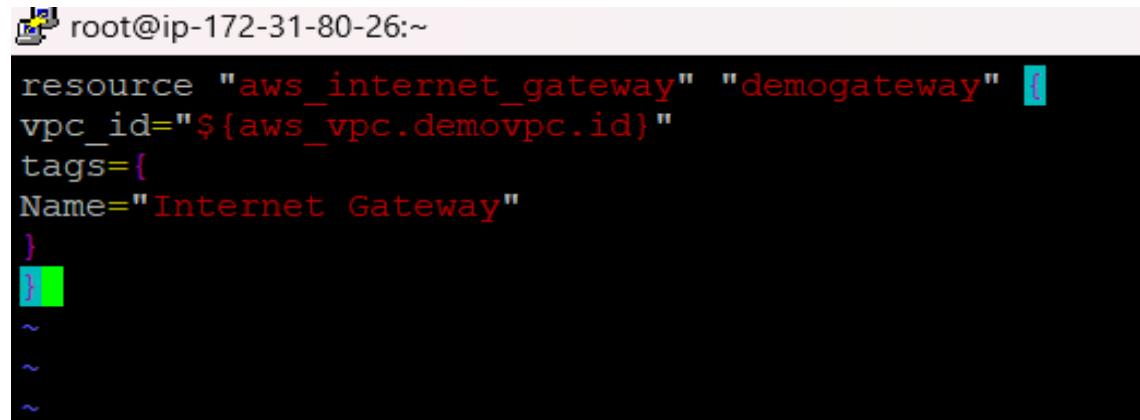
After using “terraform apply” the database subnet is creating with the sunet id.

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	IPv6 CIDR association ID	Available IPv4 addresses	Availability
-	subnet-01657ffa537...	Available	vpc-035f...	172.31.48.0/20	-	-	4091	us-east-1e
Application Subnet 2	subnet-08c496538a1e...	Available	vpc-035f...	10.0.4.0/24	-	-	251	us-east-1b
-	subnet-0a0407bd5e37...	Available	vpc-035f...	172.31.0.0/20	-	-	4091	us-east-1a
-	subnet-0b5103e5b007...	Available	vpc-035f...	172.31.64.0/20	-	-	4091	us-east-1f
Web Subnet 1	subnet-02b6448acf88...	Available	vpc-0d6b...	10.0.1.0/24	-	-	251	us-east-1a
Database Subnet 1	subnet-0e27c0fb172c...	Available	vpc-0d6b...	10.0.5.0/24	-	-	251	us-east-1a
-	subnet-0f893ee5856f...	Available	vpc-035f...	172.31.80.0/20	-	-	4090	us-east-1b
-	subnet-0f8175d8febb8...	Available	vpc-035f...	172.31.16.0/20	-	-	4091	us-east-1c
<b>Database Subnet 2</b>	<b>subnet-07bd0013517ea3edb</b>	<b>Available</b>	<b>vpc-0d6b...</b>	<b>10.0.6.0/24</b>	<b>-</b>	<b>-</b>	<b>251</b>	<b>us-east-1b</b>
-	subnet-0b0405f5ff67...	Available	vpc-035f...	172.31.32.0/20	-	-	4091	us-east-1d
Application Subnet 1	subnet-05cc00b76fe...	Available	vpc-0d6b...	10.0.3.0/24	-	-	251	us-east-1a
WEB SUBNET2	subnet-020c456a22740b...	Available	vpc-0d6b...	10.0.2.0/24	-	-	251	us-east-1b

Aws management console shows that the 6<sup>th</sup> subnet is created successfully.

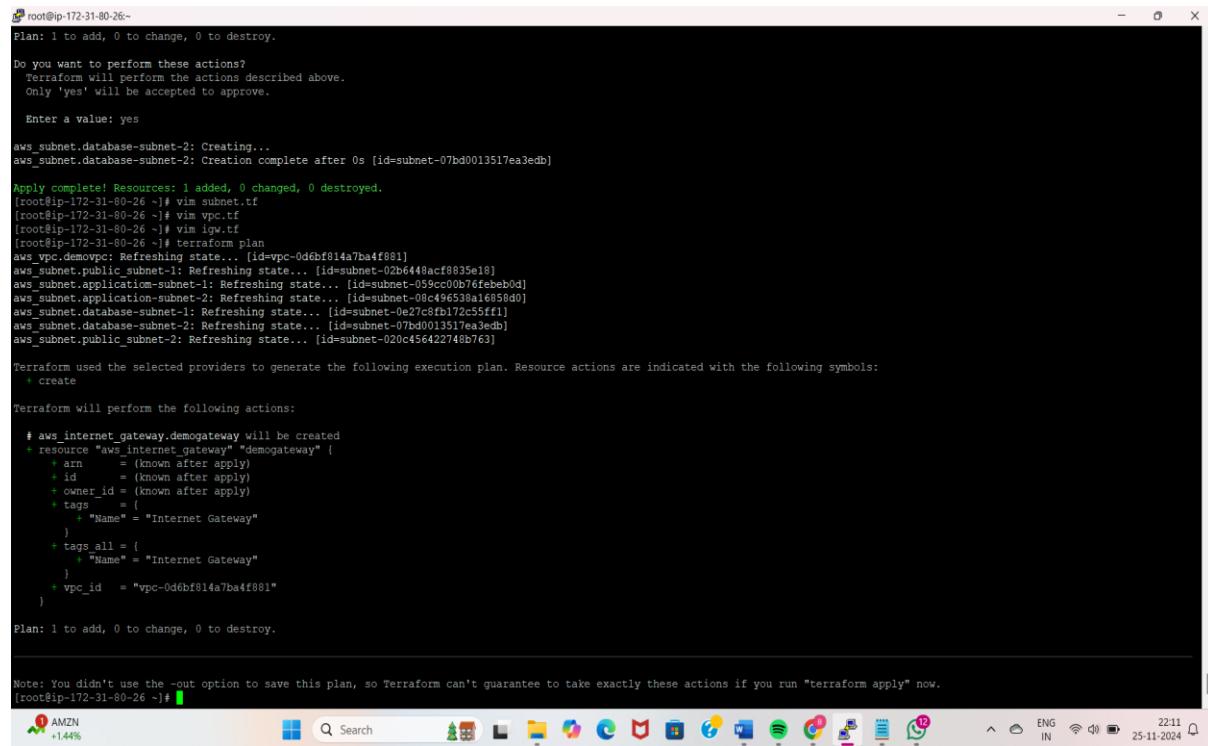
### STEP 3: CREATE INTERNET GATE WAY

Internet Gateway is used to connect our vpc to the internet, so that we can send and receive data from the resources of virtual private cloud.



```
root@ip-172-31-80-26:~  
resource "aws_internet_gateway" "demogateway" {  
vpc_id = "${aws_vpc.demovpc.id}"  
tags = {  
Name = "Internet Gateway"  
}  
}  
~  
~  
~
```

This is the code for creating internet gateway. Here we use the aws\_internet\_gateway resource type to create internet gateway for sending and receiving data from the resources.



```
root@ip-172-31-80-26:~# terraform plan  
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_subnet.database-subnet-2: Creating...  
aws_subnet.database-subnet-2: Creation complete after 0s [id= subnet-07bd0013517ea3edb]  
  
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.  
[root@ip-172-31-80-26:~]# vim subnet.tf  
[root@ip-172-31-80-26:~]# vim vpc.tf  
[root@ip-172-31-80-26:~]# vim igw.tf  
[root@ip-172-31-80-26:~]# terraform plan  
aws_vpc.demovpc: Refreshing state... [id=vpc-0d6bf814a7ba4ff881]  
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-02b6440acf8835e19]  
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-059cc0007ffeb0d]  
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-08c496530a16858d0]  
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0e27c5fb172c55ff1]  
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-07bd0013517ea3edb]  
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-020c456422740b763]  
  
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.demogateway will be created  
+ resource "aws_internet_gateway" "demogateway" {  
+   arn      = (known after apply)  
+   id       = (known after apply)  
+   owner_id = (known after apply)  
+   tags     = {  
+     "Name" = "Internet Gateway"  
+   }  
+   tags_all = {  
+     "Name" = "Internet Gateway"  
+   }  
+   vpc_id   = "vpc-0d6bf814a7ba4ff881"  
}  
  
Plan: 1 to add, 0 to change, 0 to destroy.  
  
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.  
[root@ip-172-31-80-26:~]#
```

This terminal shows that “terraform plan” command is performed to create internet gate way. There is no any mistake in the configurations so the plan is successful.

```

root@ip-172-31-18-121:~#
# aws_internet_gateway.demogateway will be created
+ resource "aws_internet_gateway" "demogateway" {
    + arn      = (known after apply)
    + id       = (known after apply)
    + owner_id = (known after apply)
    + tags_all = (known after apply)
    + vpc_id   = "vpc-0ee254eb03ed9ecc"
}

Plan: 1 to add, 0 to change, 0 to destroy.

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.
[root@ip-172-31-18-121 ~]# terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-0ee254eb03ed9ecc]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-040a13a49dbd45a69]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-0460e9c08f41b50ff]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0c53dc5497bf626e]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-04f4ad2c9d49e50a1]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-0dd23d41ff07368c4]

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.demogateway will be created
+ resource "aws_internet_gateway" "demogateway" {
    + arn      = (known after apply)
    + id       = (known after apply)
    + owner_id = (known after apply)
    + tags_all = (known after apply)
    + vpc_id   = "vpc-0ee254eb03ed9ecc"
}

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_internet_gateway.demogateway: Creating...
aws_internet_gateway.demogateway: Creation complete after 0s [id=igw-0bfa9094c244a8455]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-18-121 ~]#

```

Whatever internet gateway is going to create in aws console, the configurations are shown here.

Name	Internet gateway ID	State	VPC ID	Owner
-	igw-0d0ae3a0ff0e294a	Attached	vpc-03f5fe3fc53d1d03	905418365089
Internet Gateway	igw-0bfa9094c244a8455	Attached	vpc-0ee254eb03ed9ecc   DEMO VPC	905418365089

Internet gateway is created successfully.

#### STEP 4: CREATING THE ROUTING TABLE

Route table is used to determine how the traffic is directed within the VPC.

To create route table and associate it with the VPC and Internet gateway we need to create a route table file with the name of route\_table\_public.tf.

```
root@ip-172-31-80-26:~  
resource "aws_route_table" "route" {  
vpc_id = "${aws_vpc.demoVpc.id}"  
route{  
cidr_block = "0.0.0.0/0"  
gateway_id = "${aws_internet_gateway.demoGateway.id}"  
}  
tags = {  
Name = "Route to Internet"  
}  
}  
  
resource "aws_route_table_association" "rt1" {  
subnet_id = "${aws_subnet.public_subnet-1.id}"  
route_table_id = "${aws_route_table.route.id}"  
}  
  
resource "aws_route_table_association" "rt2" {  
subnet_id = "${aws_subnet.public_subnet-2.id}"  
route_table_id = "${aws_route_table.route.id}"  
}  
~  
~  
~
```

```
[root@ip-172-31-80-26 ~]# vim route_table_public.tf  
[root@ip-172-31-80-26 ~]# terraform plan  
aws_vpc.demoVpc: Refreshing state... [id=vpc-0d6bf814a7ba4f881]  
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-02b6448acf8835e18]  
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-08c45638a16858d0]  
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-02c456a22740b763]  
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-055cc0076fe6eb0d]  
aws_internet_gateway.demoGateway: Refreshing state... [id=igw-01a109d57b717d720]  
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0e27c8fb172c55ff1]  
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-07bd0013517ea3edb]  
  
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:  
+ create  
  
Terraform will perform the following actions:  
  
# aws_route_table.route will be created  
+ resource "aws_route_table" "route" {  
+ arn = (known after apply)  
+ id = (known after apply)  
+ owner_id = (known after apply)  
+ propagating_vgws = (known after apply)  
+ route = [  
+ {  
+ cidr_block = "0.0.0.0/0"  
+ gateway_id = "igw-01a109d57b717d720"  
# (11 unchanged attributes hidden)  
},  
]  
+ tags = {  
+ "Name" = "Route to Internet"  
}  
+ tags_all = {  
+ "Name" = "Route to Internet"  
}  
+ vpc_id = "vpc-0d6bf814a7ba4f881"  
}  
  
# aws_route_table_association.rt1 will be created  
+ resource "aws_route_table_association" "rt1" {  
+ id = (known after apply)  
+ route_table_id = (known after apply)  
+ subnet_id = "subnet-02b6448acf8835e18"  
}  
  
Plan: 2 to add, 0 to change, 0 to destroy.  
[root@ip-172-31-80-26 ~]#
```

The plan for route table is created using the command “terraform plan”.

```

root@ip-172-31-80-26:~#
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_route_table.route will be created
+ resource "aws_route_table" "route" {
    + arn          = (Known after apply)
    + id          = (Known after apply)
    + owner_id    = (Known after apply)
    + propagating_vgws = (Known after apply)
    + route        = [
        + {
            + cidr_block      = "0.0.0.0/0"
            + gateway_id     = "igw-0la109d57b717d720"
            # (11 unchanged attributes hidden)
        },
    ]
    + tags          = {
        + "Name" = "Route to Internet"
    }
    + tags_all     = [
        + "Name" = "Route to Internet"
    ]
    + vpc_id       = "vpc-0d6bf814a7ba4f881"
}

# aws_route_table_association.rtl will be created
+ resource "aws_route_table_association" "rtl" {
    + id          = (Known after apply)
    + route_table_id = (Known after apply)
    + subnet_id   = "subnet-02b648acf8835e18"
}

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_route_table.route: Creating...
aws_route_table.route: Creation complete after 1s [id=rtb-0bd9ff0c7ad3f250a2]
aws_route_table_association.rtl: Creating...
aws_route_table_association.rtl: Creation complete after 1s [id=rthassoc-0efc0c03db51dce4d]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
[root@ip-172-31-80-26 ~]# █

```

18°C Haze ENG IN 22:18 25-11-2024

Here I have executed the command terraform apply to create a route table and associate it with the vpc.

```

root@ip-172-31-18-121:~#
Plan: 0 to add, 1 to change, 0 to destroy.

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.
[root@ip-172-31-18-121 ~]# terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-0ee254eb03d9ecc0]
aws_internet_gateway.demogateway: Refreshing state... [id=igw-0fa9094c244a8455]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-0460e9c08f41b5bfff]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-04f4ad2c9d949e50a1]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0c53d1c5497bf626e]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-040a13a48dbd45a69]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-0dd23b41ff07368c4]
aws_route_table.route: Refreshing state... [id=rtb-0a89f6ab43b67c718]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- update in-place

Terraform will perform the following actions:

# aws_route_table.route will be updated in-place
- resource "aws_route_table" "route" {
    - id          = "rtb-0a89f6ab43b67c718"
    - route        = [
        - {
            - cidr_block      = "0.0.0.0/0"
            - gateway_id     = "igw-0bfa9094c244a8455"
            # (11 unchanged attributes hidden)
        },
    ]
    - tags          = {
        - "Name" = "Route to Internet"
    }
    # (5 unchanged attributes hidden)
}

Plan: 0 to add, 1 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_route_table.route: Modifying... [id=rtb-0a89f6ab43b67c718]
aws_route_table.route: Modifications complete after 0s [id=rtb-0a89f6ab43b67c718]

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
[root@ip-172-31-18-121 ~]# █

```

AFG - BAN Game score ENG IN 12:52 12-11-2024

This show that there is no error in the configurations to create the route table.

Route table is created to allow the traffic from the internet gateway to subnet.

We can see the route table is associated with the internet gateway and web subnet1 successfully.

```

root@ip-172-31-18-121:~#
# aws route_table_association.rt2 will be created
+ resource "aws_route_table_association" "rt2" {
+   id          = (known after apply)
+   route_table_id = "rtb-0a89f6ab43b67c718"
+   subnet_id   = "subnet-0460e9c08f41b58f1"
}

Plan: 1 to add, 0 to change, 0 to destroy.

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.
[root@ip-172-31-18-121 ~]# terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-0ee254eb03ed9ecc]
aws_internet_gateway.demogateway: Refreshing state... [id=igw-0fa9094c244a8455]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-0460e9c08f41b58f1]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-04fa4d2c9d4950a1]
aws_subnet.private_subnet-1: Refreshing state... [id=subnet-04fa4d2c9d4950a1]
aws_subnet.dataplane_subnet-1: Refreshing state... [id=subnet-04fa4d2c9d4950a1]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-04fa4d2c9d4950a1]
aws_subnet.application_subnet-2: Refreshing state... [id=subnet-04fa13a404db45a69]
aws_route_table.route: Refreshing state... [id=rtrt-0a89f6ab43b67c718]
aws_route_table_association.rt1: Refreshing state... [id=rtrbassoc-089f19cadedb2f5139]

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

Terraform will perform the following actions:

# aws_route_table_association.rt2 will be created
+ resource "aws_route_table_association" "rt2" {
+   id          = (known after apply)
+   route_table_id = "rtb-0a89f6ab43b67c718"
+   subnet_id   = "subnet-0460e9c08f41b58f1"
}

Plan: 1 to add, 0 to change, 0 to destroy.

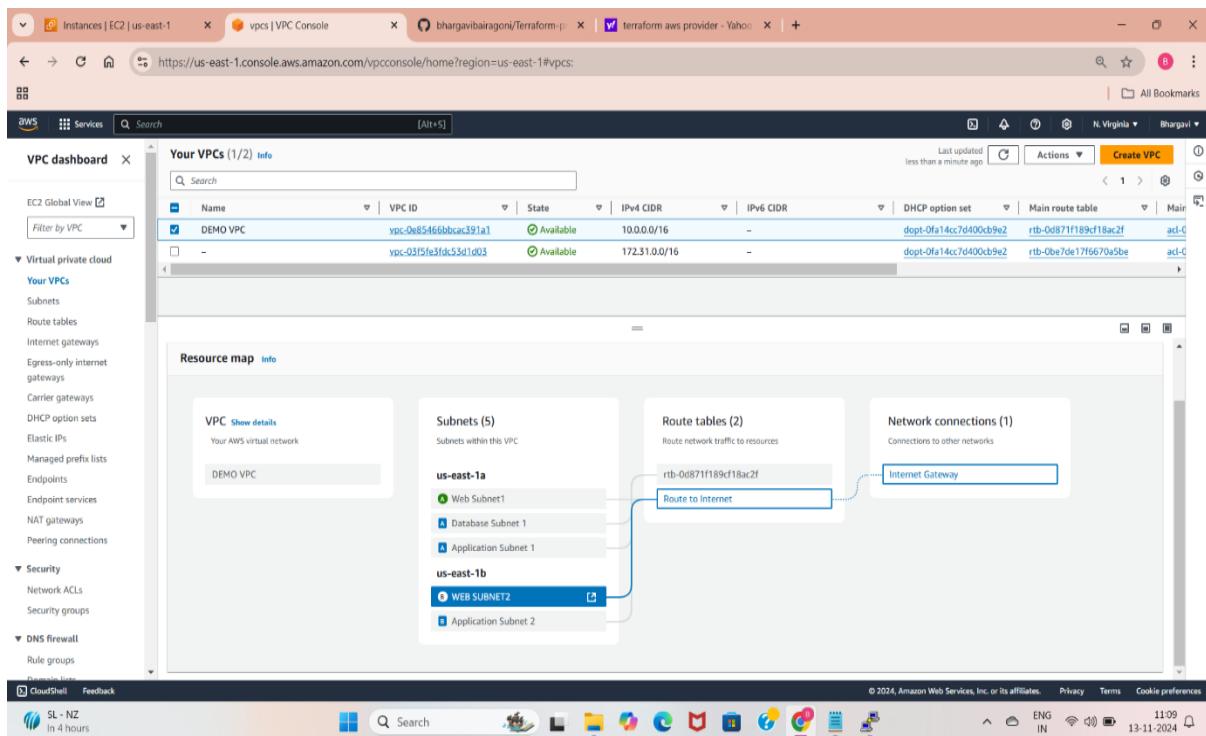
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_route_table_association.rt2: Creating...
aws_route_table_association.rt2: Creation complete after 0s [id=rtrbassoc-02915ffff9b010402d]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-18-121 ~]#
```

Creating the second subnet to associate it with the other web subnet.



Here second route table is associated with the internet gateway and web subnet2.

## STEP 5: CREATE THE EC2 INSTANCES

To create ec2 instance we need to specify the instance type, key pair and AMI in the file which is having the configurations.

```
root@ip-172-31-88-64:~  
  
resource "aws_instance" "public_subnet-1" {  
ami="ami-063d43db0594b521b"  
instance_type="t2.micro"  
count=1  
key_name="new-key"  
vpc_security_group_ids=["${aws_security_group.demosg.id}"]  
subnet_id ="${aws_subnet.public_subnet-1.id}"  
associate_public_ip_address=true  
user_data ="${file("data.sh")}"  
tags=[  
Name="My public Instance 1 "  
}  
}  
  
resource "aws_instance" "public_subnet-2" {  
ami="ami-063d43db0594b521b"  
instance_type="t2.micro"  
count=1  
key_name="new-key"  
vpc_security_group_ids=["${aws_security_group.demosg.id}"]  
subnet_id ="${aws_subnet.public_subnet-2.id}"  
associate_public_ip_address=true  
user_data ="${file("data_d.sh")}"  
tags=[  
Name="My public Instance 2 "  
}  
}
```

This terraform code creates two Amazon EC2 instances, each in a different public subnet. These two instances are configured with a public IP address and uses a single security group for network access.

```

root@ip-172-31-80-26:~# vim ec2.tf
[root@ip-172-31-80-26 ~]# terraform plan
aws_vpc.devovpc: Refreshing state... [id=vpc-0d6bf814a7ba4f881]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-054c00b76febeb0d]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-054c00b76febeb0d]
aws_internet_gateway.devovpgw: Refreshing state... [id=intgw-01a10d57b1747420]
aws_db_subnet_group.database-subnet-2: Refreshing state... [id=subnet-group-07b0013517e3e9db]
aws_security_group.devossg: Refreshing state... [id=sg-04fe87a9ed57b669b]
aws_subnet.public.subnet-1: Refreshing state... [id=subnet-02b6448acf8835e18]
aws_subnet.public.subnet-2: Refreshing state... [id=subnet-02c456422746b763]
aws_route_table.route: Refreshing state... [id=rtb-0bd9f8c7adcf258a2]
aws_route_table_association.rtl1: Refreshing state... [id=rtbassoc-0efc0c07db51dce4d]
aws_route_table_association.rtl2: Refreshing state... [id=rtbassoc-05378d1d9fc4515fa]

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

Terraform will perform the following actions:

# aws_instance.public_subnet-1[0] will be created
+ resource "aws_instance" "public_subnet-1" {
    ami                               = "ami-063d43db0594b521b"
    arn                             = (known after apply)
    associate_public_ip_address     = true
    availability_zone                = (known after apply)
    cpu_core_count                   = (known after apply)
    cpu_threads_per_core            = (known after apply)
    disable_api_stop                 = (known after apply)
    disable_api_termination          = (known after apply)
    ebs_optimized                    = (known after apply)
    get_password_data               = false
    host_id                          = (known after apply)
    host_resource_group_arn         = (known after apply)
    instance_profile                = (known after apply)
    id                               = (known after apply)
    instance_initiated_shutdown_behavior = (known after apply)
    instance_lifecycle              = (known after apply)
    instance_state                  = "t2.micro"
    instance_type                   = (known after apply)
    ipv6_address_count              = (known after apply)
    ipv6_addresses                  = (known after apply)
    key_name                        = "new-key"
    monitoring                      = (known after apply)
    outpost_arn                     = (known after apply)
    password_data                   = (known after apply)
    placement_group                 = (known after apply)
    placement_partition_number       = (known after apply)
    primary_network_interface_id    = (known after apply)
}

Breaking news
IPL auction: 13-y...

```

The plan specifies that an EC2 instance resource (public\_subnet-1, public\_subnet-2 ) will be created.

```

root@ip-172-31-80-26:~#
  + tenancy           = (known after apply)
  + user_data        = (known after apply)
  + user_data_base64 = (known after apply)
  + user_data_replace_on_change = false
  + vpc_security_group_ids = [
      + "sg-04fe87a9ed57b669b",
    ]
  + capacity_reservation_specification (known after apply)
  + cpu_options (known after apply)
  + ebs_block_device (known after apply)
  + enclave_options (known after apply)
  + ephemeral_block_device (known after apply)
  + instance_market_options (known after apply)
  + maintenance_options (known after apply)
  + metadata_options (known after apply)
  + network_interface (known after apply)
  + private_dns_name_options (known after apply)
  + root_block_device (known after apply)
  !

Plan: 2 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_instance.public_subnet-2[0]: Creating...
aws_instance.public_subnet-1[0]: Creating...
aws_instance.public_subnet-2[0]: Still creating... [10s elapsed]
aws_instance.public_subnet-1[0]: Still creating... [10s elapsed]
aws_instance.public_subnet-2[0]: Creation complete after 12s (id=i-00f5ee0048bbcd4b)
aws_instance.public_subnet-1[0]: Creation complete after 13s (id=i-0a441188255e4da71)

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

```

The terraform apply command is executed to create the infrastructure defined in the Terraform configuration (ec2.tf).

The screenshot shows the AWS EC2 Instances console. The left sidebar navigation includes: Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups), and Key Pair. The main content area displays the 'Instances (1/3) Info' table with two rows:

Name	Instance ID	Instance state	Public IPv4 DNS
My public Instance 2	i-0a61c08d11659...	Running	3.95.230.78
My public Instance 1	i-0a4abd96a44bc0...	Running	3.234.182.100

Details for 'My public Instance 1' are shown in the modal: Security details (IAM Role: -, Owner ID: 905418365089, Launch time: Thu Nov 14 2024 12:30:21 GMT+0530 (India Standard Time)), Inbound rules (three entries for ports 443, 22, and 80), and Outbound rules (three entries for ports 443, 22, and 80). The status bar at the bottom indicates: 28°C Haze, ENG IN, 12:38, 14-11-2024.

The console displays the list of EC2 instances currently running in the AWS environment. These two ec2 instances are created with the terraform code.

The screenshot shows a web browser window with the URL <http://3.85.146.207>. The page content is: "Hello World from EC2 instance This is Bhargavi Working on second project Second project is terraform ip-10-0-2-66.ec2.internal". The status bar at the bottom indicates: 21°C Haze, ENG IN, 22:37, 26-11-2024.

I have used user data in the terraform code, so it is hosting a application in apache server. I am able access this application using above created instance public IP.

## STEP 6: CREATE A FILE FOR SECURITY GROUP FOR THE FRONT-END-TIER

To create any type of ec2 instance we need to create a security group to access those instances. A Security Group acts as a virtual firewall that controls inbound and outbound traffic for your EC2 instances. Here am creating a security group with the name of web\_sg.tf which allows HTTP, HTTPS, and SSH. These ports helps us to connect with the terminals, access any applications through browsers.

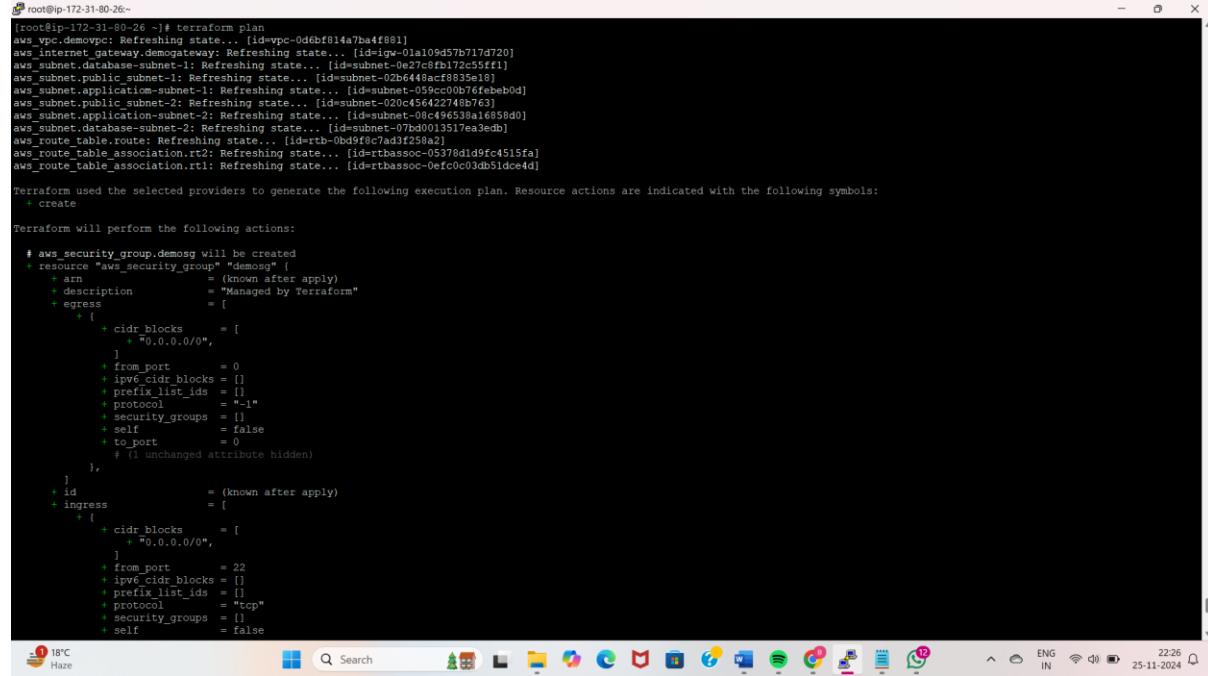
```
root@ip-172-31-80-26:~  
resource "aws_security_group" "demosg" {  
  vpc_id = "${aws_vpc.demovpc.id}"  
  
  ingress{  
    from_port=80  
    to_port=80  
    protocol="tcp"  
    cidr_blocks=["0.0.0.0/0"]  
  }  
  
  ingress{  
    from_port=443  
    to_port=443  
    protocol="tcp"  
    cidr_blocks=["0.0.0.0/0"]  
  }  
  
  ingress{  
    from_port=22  
    to_port=22  
    protocol="tcp"  
    cidr_blocks=["0.0.0.0/0"]  
  }  
  
  egress{  
    from_port=0  
    to_port=0  
    protocol="-1"  
    cidr_blocks=["0.0.0.0/0"]  
  }  
  tags={  
    Name="WEB SG"  
  }  
}
```

This picture shows the inbound and outbound rules allowed in the security group for frontend tier.

22- Allows secure shell (SSH) access to the EC2 instance.

443- This port is essential for secure communications between clients (e.g., web browsers) and the server.

80- Typically used for public websites where secure encryption (HTTPS) is not required.



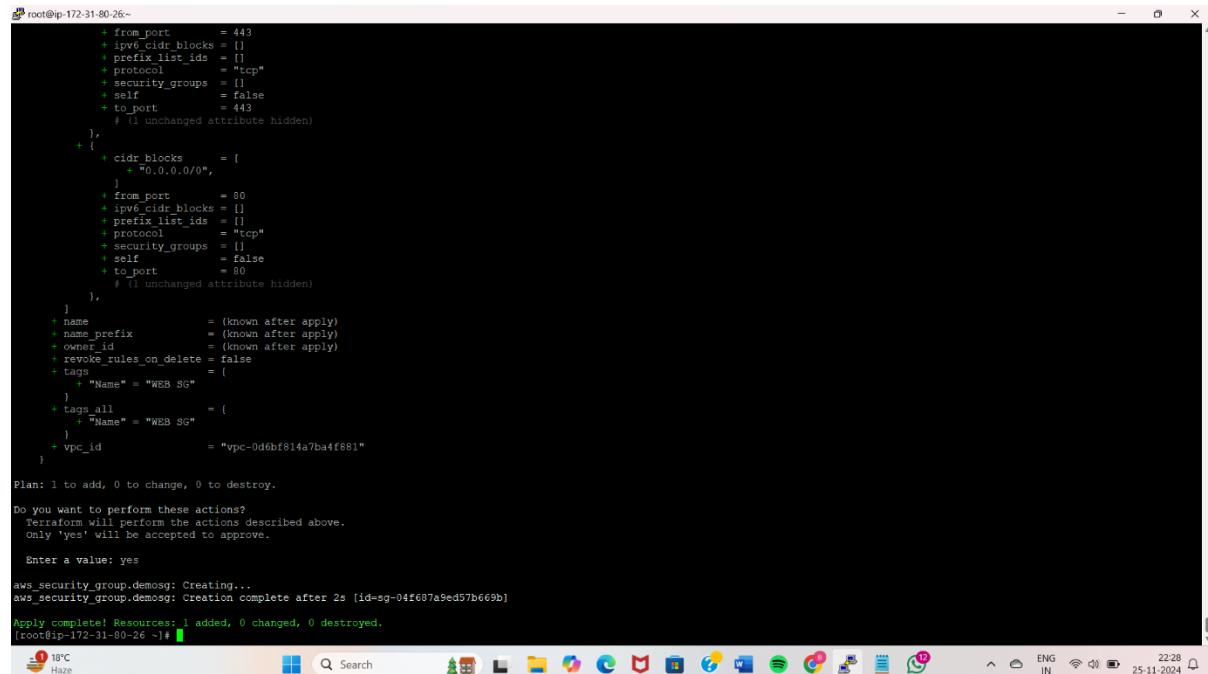
```
[root@ip-172-31-80-26 ~]# terraform plan
aws_vpc.vpc: Refreshing state... [id=vpc-0d6bf814a7ba4ff881]
aws_internet_gateway.gateway: Refreshing state... [id=igw-01a109d57b717d720]
aws_subnet.database_subnet-1: Refreshing state... [id=subnet-0e27c8fb172c55ff1]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-02b6448acf8835e18]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-059c00b76febe0d]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-020c456a22748763]
aws_subnet.application_subnet-2: Refreshing state... [id=subnet-066538a16858d0]
aws_route_table.route: Refreshing state... [id=rtb-097a0013517ea3ed8]
aws_route_table_association.rt2: Refreshing state... [id=rbassoc-05378d1d9fc4515fa]
aws_route_table_association.rt1: Refreshing state... [id=rbassoc-0efc0c03db51dce4d]

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

Terraform will perform the following actions:

# aws_security_group.demosg will be created
+ resource "aws_security_group" "demosg" {
    + arn = "(known after apply)"
    + description = "Managed by Terraform"
    + egress = [
        + {
            + cidr_blocks = [
                + "#0.0.0.0/0",
            ]
            + from_port = 0
            + ipv6_cidr_blocks = []
            + prefix_list_ids = []
            + protocol = "-1"
            + security_groups = []
            + self = false
            + to_port = 0
            # (1 unchanged attribute hidden)
        },
        + {
            + id = "(known after apply)"
            + ingress = [
                + {
                    + cidr_blocks = [
                        + "#0.0.0.0/0",
                    ]
                    + from_port = 22
                    + ipv6_cidr_blocks = []
                    + prefix_list_ids = []
                    + protocol = "tcp"
                    + security_groups = []
                    + self = false
                }
            ]
        }
    ]
}
```

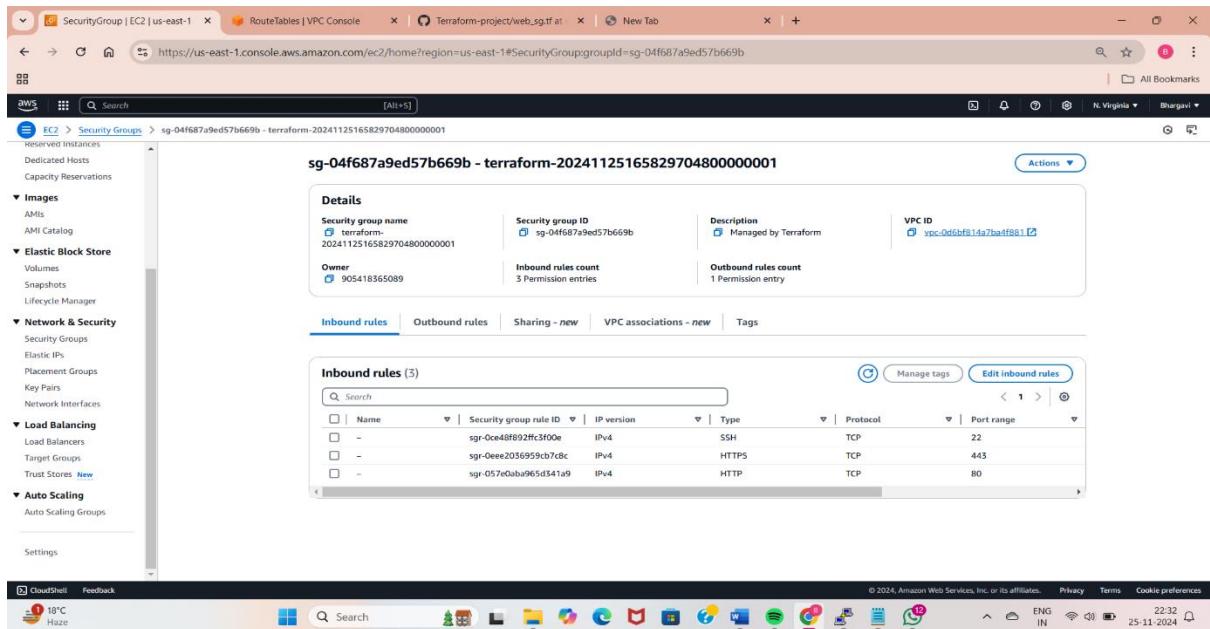
This plan specifies that an security group resource for instances public\_subnet-1, public\_subnet-2 will be created.



```
[root@ip-172-31-80-26 ~]# terraform apply
aws_security_group.demosg: Creating...
aws_security_group.demosg: Creation complete after 2s [id=sg-04f607a9ed57b669b]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-80-26 ~]#
```

The terraform apply command is executed to create the infrastructure defined in the Terraform configuration for creating a security group(web\_sg.tf).



The aws console shows that the security group for frontend tier is created with specified configurations.

## STEP 7: CREATE A FILE FOR SECURITY GROUP FOR THE DATABASE TIER

To create a Security Group for the Database Tier in Terraform we need to allow mysql port with the port number 3306. Here am creating a file with the name of database\_sg.tf for database tier configurations.

```
root@ip-172-31-80-26:~#
resource "aws_security_group" "database-sg" {
  name      = "Database SG"
  description = "Allow inbound traffic from application layer"
  vpc_id    = aws_vpc.demovpc.id
  ingress {
    description = "allow traffic from application layer"
    from_port  = 3306
    to_port   = 3306
    protocol  = "tcp"
    security_groups = [aws_security_group.demosg.id]
  }
  egress {
    from_port = 32768
    to_port   = 65535
    protocol  = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = [
    { Name = "Database SG" }
  ]
}
```

This resource creates an AWS Security Group named database-sg, which is intended for controlling network access to resources in the Database like MySQL or similar database. The security group is

associated with the above created VPC. This security group is specifically designed to allow database instances to receive traffic from application servers with the demosg security group on port 3306. It is also configured to allow outbound communication over a range of 32768–65535.

```

root@ip-172-31-80-26:~$ vim ec2.tf
root@ip-172-31-80-26:~$ vim database_sg.tf
root@ip-172-31-80-26:~$ terraform plan
aws_vpc.demosvg: Refreshing state... [id=vpc-0d6bf814a7ba4ff881]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-08c496538a16658d0]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-02b6448acf8835e18]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-020c456422748b763]
aws_internet_gateway.demogateway: Refreshing state... [id=igw-01a109d57b717d720]
aws_security_group.demosg: Refreshing state... [id=sg-04f687a9ed57b669b]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-059cc00876febeb0d]
aws_subnet.database_subnet-1: Refreshing state... [id=subnet-0e27c8fb172c55ff1]
aws_subnet.database_subnet-2: Refreshing state... [id=subnet-07bd4d517e3edb]
aws_instance.featured: Refreshing state... [id=rb-0b957a3138a02]
aws_instance_public_subnet-1[0]: Refreshing state... [id=ip-0a45600048bcc44b]
aws_instance_public_subnet-1[0]: Refreshing state... [id=ip-0a441188255e4d71]
aws_route_table_association.rt1: Refreshing state... [id=rtbassoc-0efc0c03db51dce4d]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-05378d1d9fc4515fa]

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

Terraform will perform the following actions:

# aws_security_group.database_sg will be created
+ resource "aws_security_group" "database_sg" {
    + arn          = (known after apply)
    + description  = "Allow inbound traffic from application layer"
    + egress       = [
        + {
            + cidr_blocks = [
                + "0.0.0.0/0",
            ]
            + from_port   = 32768
            + ipv4_cidr_blocks = []
            + prefix_list_ids = []
            + protocol    = "tcp"
            + security_groups = []
            + self        = false
            + to_port     = 65535
            # (1 unchanged attribute hidden)
        },
    ]
    + id          = (known after apply)
    + ingress     = [
        + {
            + cidr_blocks = []
            + description = "allow traffic from application layer"
            + from_port   = 3306
            + ipv6_cidr_blocks = []
        },
    ],
    + name        = "Database SG"
    + name_prefix = (known after apply)
    + owner_id    = (known after apply)
    + revoke_rules_on_delete = false
    + tags        = [
        + "Name" = "Database SG"
    ]
    + tags_all    = [
        + "Name" = "Database SG"
    ]
    + vpc_id      = "vpc-0d6bf814a7ba4ff881"
  ]
}

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.database_sg: Creating...
aws_security_group.database_sg: Creation complete after 3s [id=sg-0cfcd5073c0acf73c]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

```

The plan for creating security group specifies the configurations of database security group.

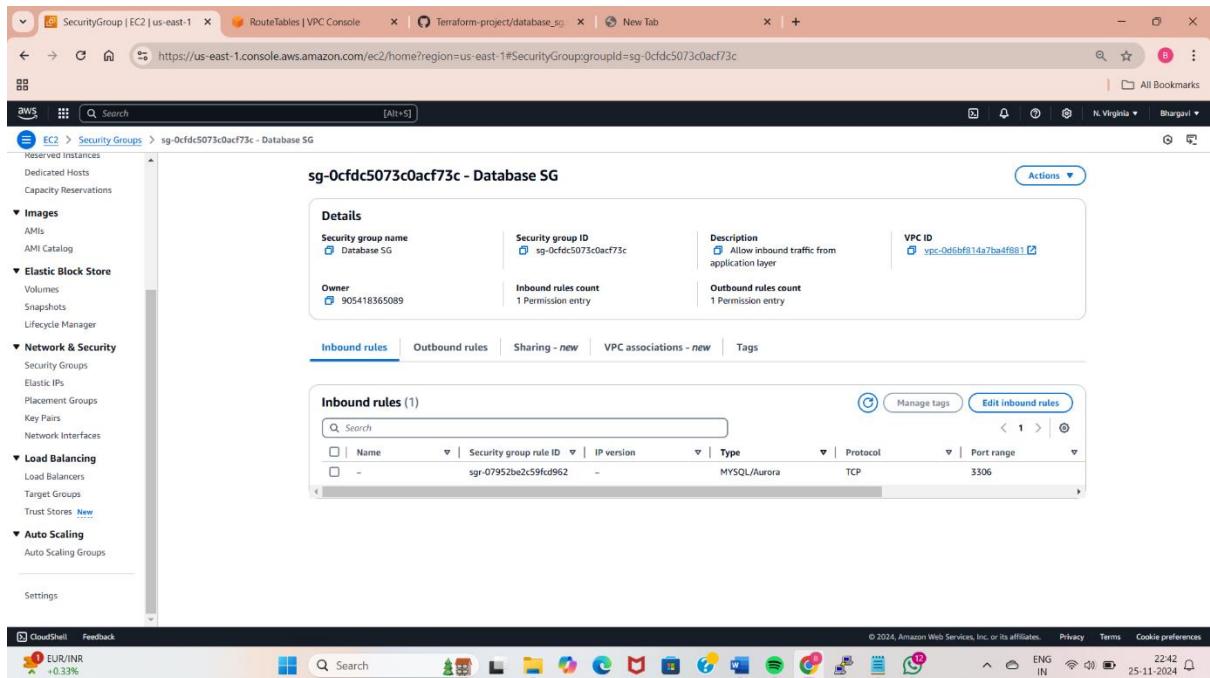
```

root@ip-172-31-80-26:~$ terraform apply
aws_security_group.database_sg: Creating...
aws_security_group.database_sg: Creation complete after 3s [id=sg-0cfcd5073c0acf73c]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

```

The terraform apply command is executed to create the infrastructure defined in the Terraform code which creates a security group for database tier(database\_sg.tf).



We can see in AWS management console that database security group is created with mentioned configurations.

## **STEP 8: CREATE A FILE FOR APPLICATION LOAD BALANCER**

An Application Load Balancer (ALB) is a service provided by Amazon Web Services that distributes incoming traffic across multiple targets, such as EC2 instances, containers, or IP addresses, within one or more Availability Zones. To create a load balancer we need to specify configurations in a file. Here am creating a `alb.tf` file for load balancer and target group.

```
root@ip-172-31-80-26:~  
resource "aws_lb" "external-alb" {  
  name="External-LB"  
  internal=false  
  load_balancer_type="application"  
  security_groups=[aws_security_group.demosg.id]  
  subnets=[aws_subnet.public_subnet-1.id,aws_subnet.public_subnet-2.id]  
}  
  
resource "aws_lb_target_group" "external-alb" {  
  name="ALB-TG"  
  port=80  
  protocol="HTTP"  
  vpc_id=aws_vpc.demoVPC.id  
}  
  
resource "aws_lb_target_group_attachment" "attachment1" {  
  target_group_arn=aws_lb_target_group.external-alb.arn  
  target_id="#[aws_instance.public_subnet-1[0].id]"  
  port=80  
  depends_on=[aws_instance.public_subnet-1]  
}  
  
resource "aws_lb_target_group_attachment" "attachment2" {  
  target_group_arn=aws_lb_target_group.external-alb.arn  
  target_id="#[aws_instance.public_subnet-2[0].id]"  
  port=80  
  depends_on=[  
    aws_instance.public_subnet-2  
  ]  
}  
  
resource "aws_lb_listener" "external-alb" {  
  load_balancer_arn=aws_lb.external-alb.arn  
  port=80  
  protocol="HTTP"  
  default_action{  
    type="forward"  
    target_group_arn=aws_lb_target_group.external-alb.arn  
  }  
}  
alb.tf" 42L, 970B  
17°C ENG 01:11 42,1 All  
Haze IN 26-11-2024
```

The Terraform configuration provided defines an Application Load Balancer (ALB) setup that includes the load balancer itself, target groups, target group attachments, and a listener for handling incoming HTTP traffic. Associating the load balancer with a security group, demosg, which controls inbound and outbound traffic rules for the ALB.

```

root@ip-172-31-80-26:~$ terraform plan
Terraform will perform the following actions:

# aws_lb.external-alb will be created
+ resource "aws_lb" "external-alb" {
  arn
  arn_suffix
  alias_arns
  alive_check_interval
  desync_mitigation_mode
  dns_name
  drop_invalid_header_fields
  enable_deletion_protection
  enable_http2
  enable_tls_versions_and_cipher_suites_headers
  idle_timeout
  internal
  ip_address_type
  load_balancer_type
  name
  name_prefix
  preserve_host_header
  security_groups
    + "sg-04f607a9ed57b669b",
  subnet_ids
    + "subnet-020c456422740b763",
    + "subnet-02b644acf0835e10",
}
+ subnets = [
  + "subnet-020c456422740b763",
  + "subnet-02b644acf0835e10",
]
+ tags_all = (known after apply)
+ vpc_id = (known after apply)
+ xff_header_processing_mode = (known after apply)
+ zone_id = (known after apply)

+ subnet_mapping (known after apply)

# aws_lb_listener.external-alb will be created
+ resource "aws_lb_listener" "external-alb" {
  arn = (known after apply)
  id = (known after apply)
  load_balancer_arn = (known after apply)
  port = 80
  protocol = "HTTP"
  ssl_policy = (known after apply)
  tags_all = (known after apply)
}

# aws_lb_target_group.external-alb: Creation complete after 1s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:targetgroup/ALB-TG/6920dc6e0298684c]
# aws_lb_target_group_attachment.attachment1: Creation complete after 1s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:targetgroup/ALB-TG/6920dc6e0298684c-20241125171620894900000001]
# aws_lb_target_group_attachment.attachment2: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:targetgroup/ALB-TG/6920dc6e0298684c-20241125171620913300000002]
]

aws_lb_target_group_attachment.attachment1: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:targetgroup/ALB-TG/6920dc6e0298684c-20241125171620913300000002]

aws_lb_external_alb: Still creating... [10s elapsed]
aws_lb_external_alb: Still creating... [20s elapsed]
aws_lb_external_alb: Still creating... [30s elapsed]
aws_lb_external_alb: Still creating... [40s elapsed]
aws_lb_external_alb: Still creating... [50s elapsed]
aws_lb_external_alb: Still creating... [1m0s elapsed]
aws_lb_external_alb: Still creating... [1m10s elapsed]
aws_lb_external_alb: Still creating... [1m20s elapsed]
aws_lb_external_alb: Still creating... [1m30s elapsed]
aws_lb_external_alb: Still creating... [1m40s elapsed]
aws_lb_external_alb: Still creating... [1m50s elapsed]
aws_lb_external_alb: Still creating... [2m0s elapsed]
aws_lb_external_alb: Still creating... [2m10s elapsed]
aws_lb_external_alb: Still creating... [2m20s elapsed]
aws_lb_external_alb: Still creating... [2m30s elapsed]
aws_lb_external_alb: Still creating... [2m40s elapsed]
aws_lb_external_alb: Still creating... [2m50s elapsed]
aws_lb_external_alb: Still creating... [3m0s elapsed]
aws_lb_external_alb: Still creating... [3m10s elapsed]
aws_lb_external_alb: Still creating... [3m20s elapsed]
aws_lb_external_alb: Still creating... [3m30s elapsed]
aws_lb_external_alb: Still creating... [3m40s elapsed]
aws_lb_external_alb: Still creating... [4m0s elapsed]
aws_lb_external_alb: Still creating... [4m10s elapsed]
aws_lb_external_alb: Still creating... [4m20s elapsed]
aws_lb_external_alb: Still creating... [4m30s elapsed]
aws_lb_external_alb: Creation complete after 4m33s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:loadbalancer/app/External-LB/8536e31850d6c172]
aws_lb_listener.external-alb: Creating...
aws_lb_listener.external-alb: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:listener/app/External-LB/8536e31850d6c172/e4bb12392602111a]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
[root@ip-172-31-80-26 ~]# 
```

The plan for creating load balancer specifies the required configurations to distribute the load across the servers.

```

root@ip-172-31-80-26:~$ terraform apply
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_lb.external-alb: Creating...
aws_lb_target_group.external-alb: Creating...
aws_lb_target_group_attachment.attachment1: Creation complete after 1s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:targetgroup/ALB-TG/6920dc6e0298684c]
aws_lb_target_group_attachment.attachment2: Creating...
aws_lb_target_group_attachment.attachment2: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:targetgroup/ALB-TG/6920dc6e0298684c-20241125171620894900000001]
aws_lb_target_group_attachment.attachment1: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:targetgroup/ALB-TG/6920dc6e0298684c-20241125171620913300000002]

aws_lb_external_alb: Still creating... [10s elapsed]
aws_lb_external_alb: Still creating... [20s elapsed]
aws_lb_external_alb: Still creating... [30s elapsed]
aws_lb_external_alb: Still creating... [40s elapsed]
aws_lb_external_alb: Still creating... [50s elapsed]
aws_lb_external_alb: Still creating... [1m0s elapsed]
aws_lb_external_alb: Still creating... [1m10s elapsed]
aws_lb_external_alb: Still creating... [1m20s elapsed]
aws_lb_external_alb: Still creating... [1m30s elapsed]
aws_lb_external_alb: Still creating... [1m40s elapsed]
aws_lb_external_alb: Still creating... [1m50s elapsed]
aws_lb_external_alb: Still creating... [2m0s elapsed]
aws_lb_external_alb: Still creating... [2m10s elapsed]
aws_lb_external_alb: Still creating... [2m20s elapsed]
aws_lb_external_alb: Still creating... [2m30s elapsed]
aws_lb_external_alb: Still creating... [2m40s elapsed]
aws_lb_external_alb: Still creating... [2m50s elapsed]
aws_lb_external_alb: Still creating... [3m0s elapsed]
aws_lb_external_alb: Still creating... [3m10s elapsed]
aws_lb_external_alb: Still creating... [3m20s elapsed]
aws_lb_external_alb: Still creating... [3m30s elapsed]
aws_lb_external_alb: Still creating... [3m40s elapsed]
aws_lb_external_alb: Still creating... [4m0s elapsed]
aws_lb_external_alb: Still creating... [4m10s elapsed]
aws_lb_external_alb: Still creating... [4m20s elapsed]
aws_lb_external_alb: Still creating... [4m30s elapsed]
aws_lb_external_alb: Creation complete after 4m33s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:loadbalancer/app/External-LB/8536e31850d6c172]
aws_lb_listener.external-alb: Creating...
aws_lb_listener.external-alb: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:listener/app/External-LB/8536e31850d6c172/e4bb12392602111a]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
[root@ip-172-31-80-26 ~]# 
```

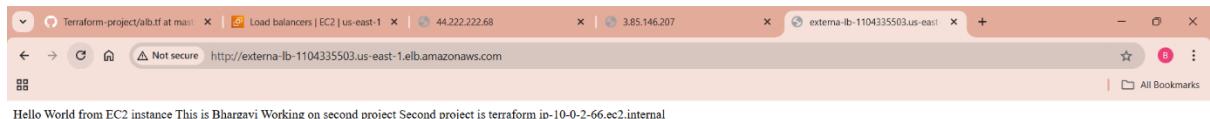
The Terraform apply command creating the load balancer and target group with two instances, including the specified configurations.

The screenshot shows the AWS EC2 Target groups console. On the left, a navigation sidebar lists various services like Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, 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 (which is selected), Trust Stores, Auto Scaling, and Auto Scaling Groups. The main content area displays a table for 'Target groups (1/1)'. One target group, 'ALB-TG', is listed with details: ARN: arn:aws:elasticloadbalancing:us-east-1:80, Port: 80, Protocol: HTTP, Target type: Instance, Load balancer: None associated, and VPC ID: vpc-080ed1259c760a309. Below this, a detailed view for 'Target group: ALB-TG' shows the 'Targets' tab selected, displaying two registered targets: 'i-004bd96aa44bc05f7' and 'i-0a61c08d116591721', both of which are in a healthy state.

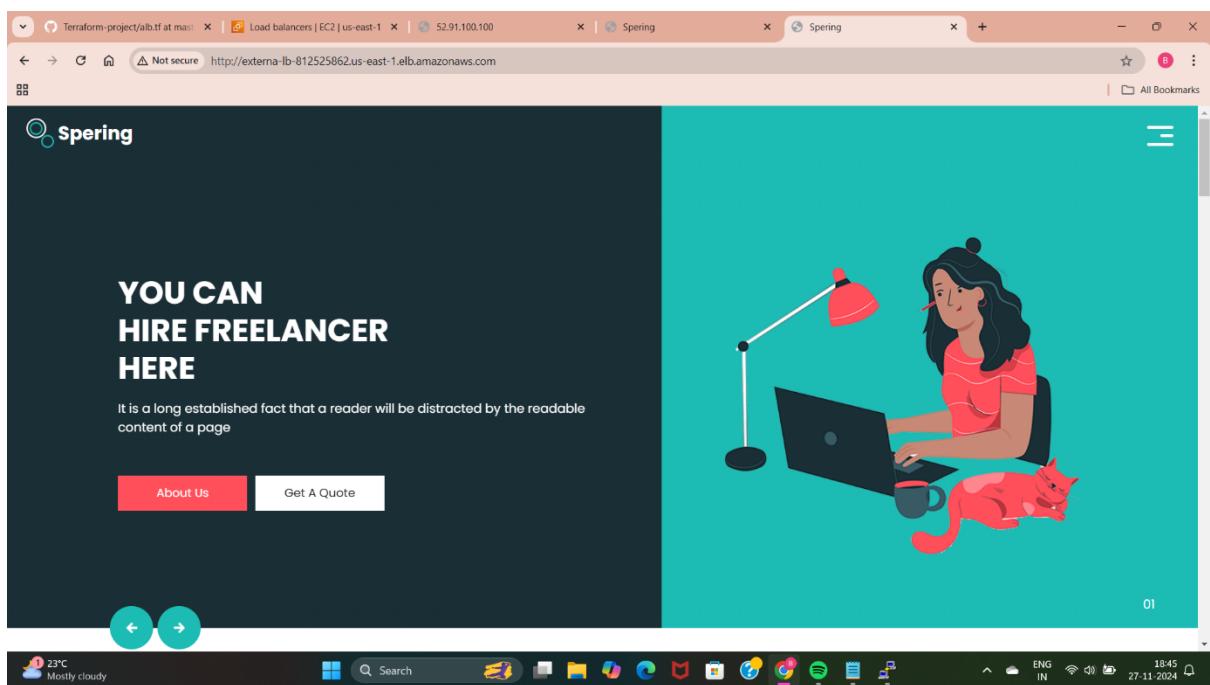
Above console shows the target group which is created with two instances and the instances are in healthy state.

The screenshot shows the AWS EC2 Load balancers console. The left sidebar includes options for Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Load Balancing, Load Balancers (selected), Target Groups, Trust Stores, and Auto Scaling. The main area shows a table for 'Load balancers (1/1)' with one entry: 'External-LB' (ARN: arn:aws:elb:us-east-1:1644513520.us-east-1.elb.amazonaws.com, State: Active, VPC ID: vpc-0d6bf814a7ba4ff881, Availability Zones: 2, Type: application, Date created: November 25, 2024, 22:46 (UTC+0)). Below this, a detailed view for 'Load balancer: External-LB' shows the 'Listeners and rules' tab selected, listing a single rule for 'HTTP:80' with a forward action to target group 'ALB-TG' at 100% weight and target group stickiness off.

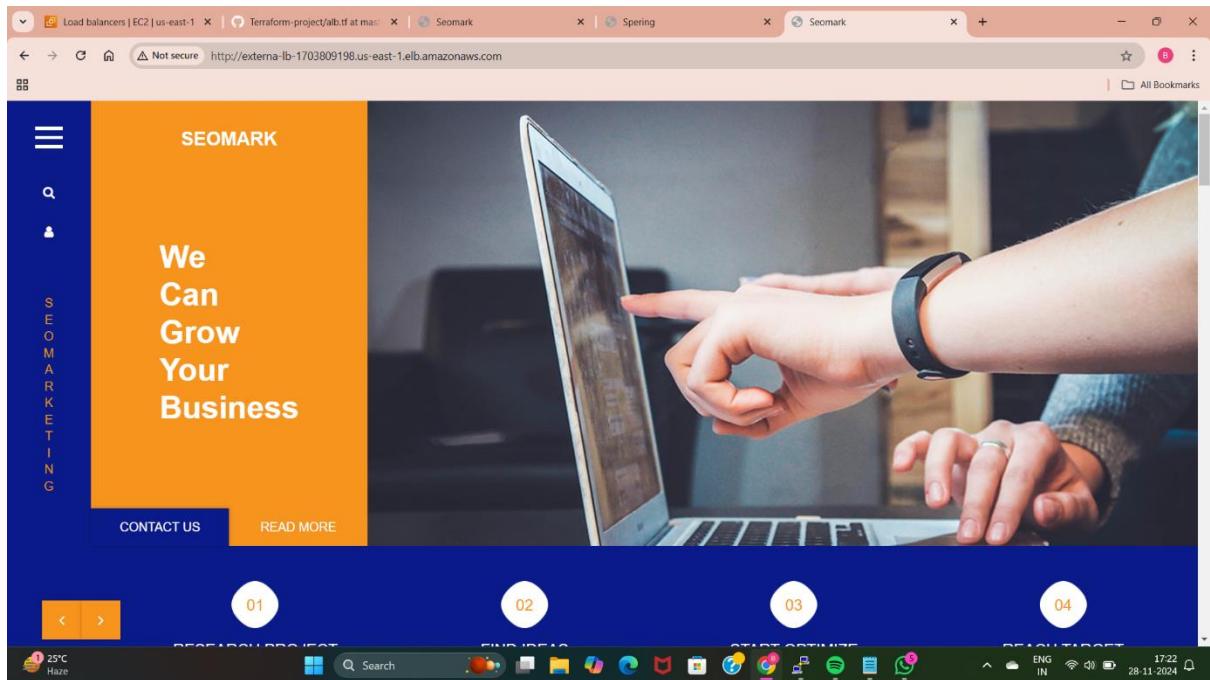
This is the load balancer which is created with the name of External-LB, we can also observe the DNS name which helps us to access any application which is used in the registered instances.



When I am trying to access the DNS of the load balancer am able to access the application which is hosted while we creating the instance using(data.sh)user data.



When I am trying to access the above created DNS name of load balancer am able to access the spering website which is stored in one of the ec2 instances.



When I am trying to increase the load then the traffic is distributed to other instance which is having other application with the name of Browny.

#### STEP 9: CREATE A FILE FOR THE RDS INSTANCE

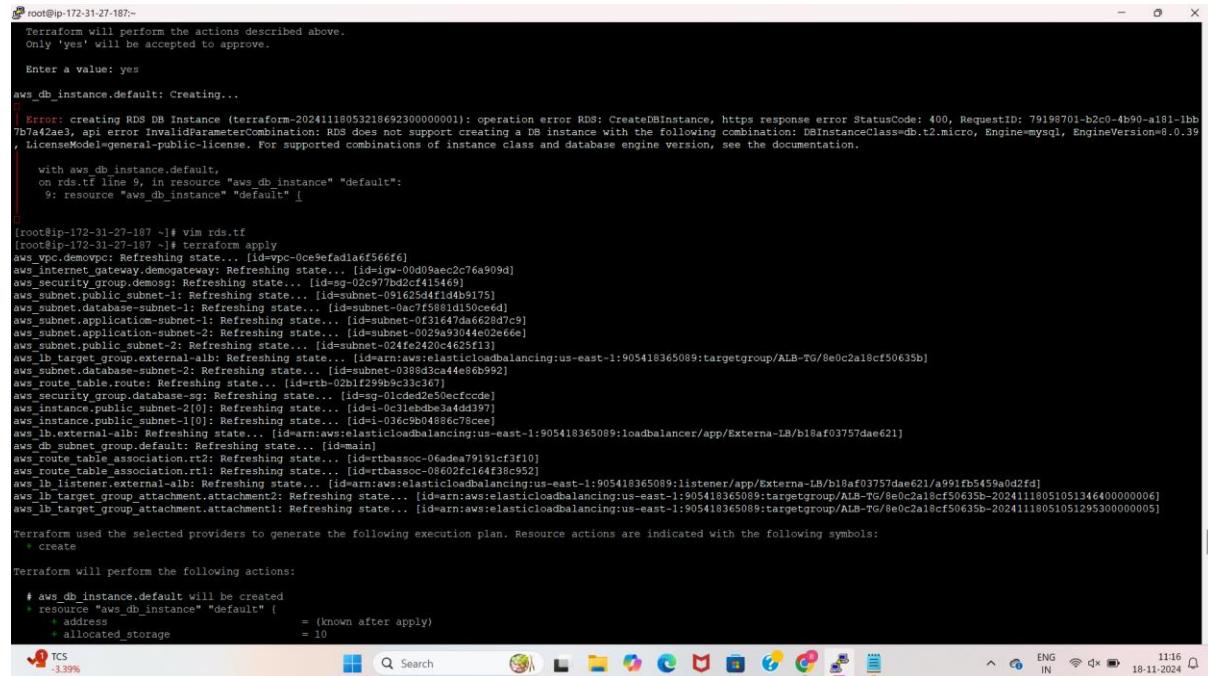
Amazon RDS (Relational Database Service) is a managed database service provided by AWS that makes it easier to set up, operate, and scale relational databases in the cloud. RDS handles routine database tasks such as provisioning, patching, backup, recovery, and scaling, allowing developers and businesses to focus on application development rather than managing the underlying infrastructure.

To create rds we need to take any database engine like mysql, oracle etc, for this we need to create a file so that database is automatically created when we run the file. Here am creating a file with the name of rds.tf.

```
root@ip-172-31-80-26:~  
resource "aws_db_subnet_group" "default" {  
  name = "main"  
  subnet_ids = [aws_subnet.database-subnet-1.id, aws_subnet.database-subnet-2.id]  
  tags = [  
    {Name = "My DB subnet group"}  
  ]  
  
  resource "aws_db_instance" "default" {  
    allocated_storage = 10  
    db_subnet_group_name = aws_db_subnet_group.default.id  
    engine = "MySQL"  
    engine_version = "8.0.39"  
    instance_class = "db.t3.micro"  
    multi_az = true  
    db_name = "mydb"  
    username = "username"  
    password = "password"  
    skip_final_snapshot = true  
    vpc_security_group_ids = [aws_security_group.database-sg.id]  
  }  
}
```

This Terraform configuration creates an RDS instance for a MySQL database in a specific subnet group within a VPC. This ensures that the database is launched within specified subnets for better control and availability. The `aws_db_instance` creates an RDS instance with MySQL as the database engine.

Here, am using 8.0.39 MySQL version, db.t3.micro instance, which is cost-effective and suitable for development and testing purposes. Username and password define the credentials for the database admin user. Snapshot prevent the data loss when (we set it to true) the data is deleted unintentionally.



```

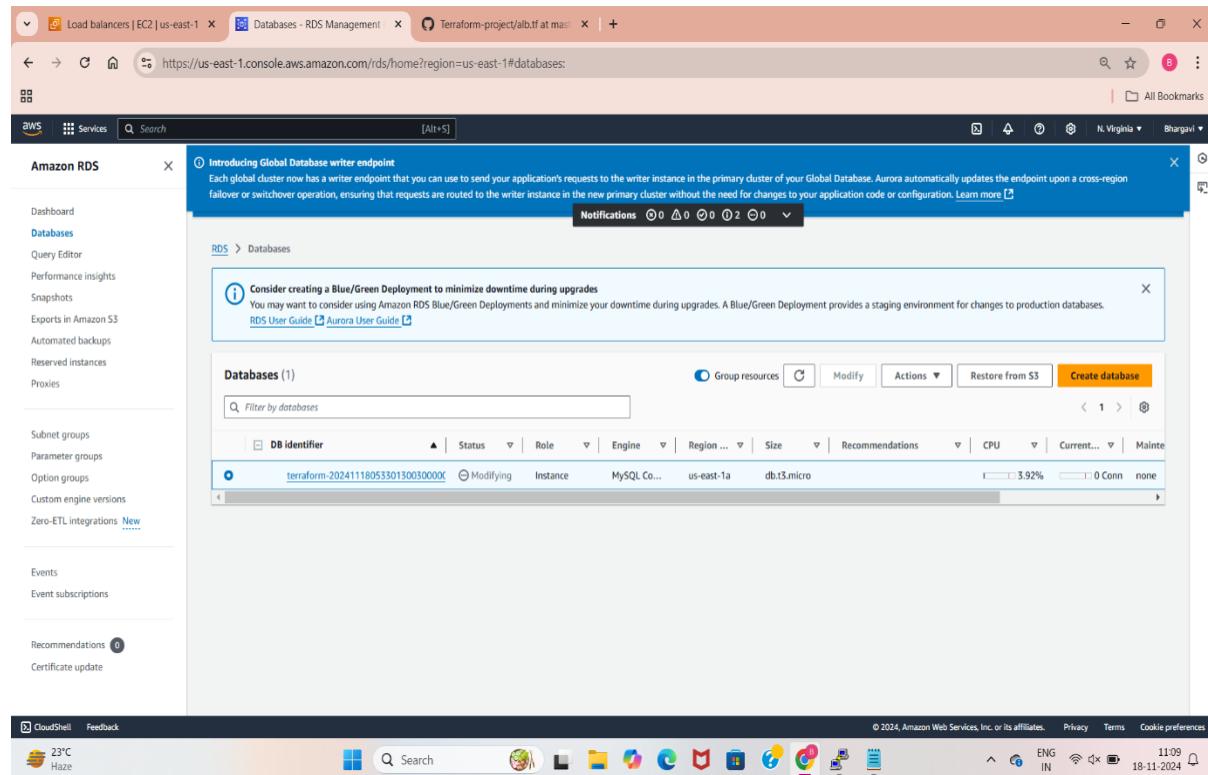
root@ip-172-31-27-187:~#
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_db_instance.default: Creating...
[...] Error: creating RDS DB Instance (terraform-2024111803218692300000001): operation error RDS: CreateDBInstance, https response error StatusCode: 400, RequestID: 79198701-b2c0-4b90-a181-lbb7b7a2ae3, api error InvalidParameterCombination: RDS does not support creating a DB instance with the following combination: DBInstanceClass=db.t2.micro, Engine=mysql, EngineVersion=8.0.39, LicenseModel=general-public-license. For supported combinations of instance class and database engine version, see the documentation.

with aws_db_instance.default,
on rds.tf line 9, in resource "aws_db_instance" "default":
  9: resource "aws_db_instance" "default" {
```

The plan for creating RDS with the specified configuration(rds.tf) is successful.



DB identifier	Status	Role	Engine	Region ...	Size	Recommendations	CPU	Current...	Maintain...
terraform-2024111803218692300000001	Modifying	Instance	MySQL Co...	us-east-1a	db.t3.micro				

We can see the RDS database is creating in aws console. It takes some time to create and make the database to available state.

The screenshot shows the AWS RDS service page. On the left, there's a sidebar with options like Dashboard, Databases, Query Editor, etc. The main area is titled 'Databases' and shows a table with one row. The row details a database named 'terraform-2024111805330130030000001'. The status is 'Available', and it's an 'Instance' type using MySQL Community Engine. The table includes columns for DB identifier, Status, Role, Engine, Region, Size, Recommendations, CPU, Current, and Maintenance. A message at the top encourages using a Blue/Green deployment. The bottom right corner shows the AWS CloudShell icon and the date/time: 18-11-2024.

Here the database is in available state, so we can get endpoint to access this database.

This screenshot shows the 'Database Details' page for the database 'terraform-2024111805330130030000001'. The top navigation bar includes 'Load balancers | EC2 | us-east-1', 'Database Details - RDS Management', 'Terraform-project/alg.tf at master · Terraform', and a '+' button. The main content area has tabs for 'Summary', 'Connectivity & security', 'Monitoring', 'Logs & events', 'Configuration', 'Zero-ETL integrations', 'Maintenance & backups', 'Tags', and 'Recommendations'. The 'Summary' tab is active, displaying details such as DB identifier, Status (Modifying), Role (Instance), Engine (MySQL Community), and Recommendations. The 'Connectivity & security' tab is also visible, showing the Endpoint (terraform-2024111805330130030000001.cla.u2u6sigra.us-east-1.rds.amazonaws.com), Port (3306), Networking (Availability Zone: us-east-1a, VPC: DEMO VPC (vpc-0ce9efad1a6f56f6)), and Security (VPC security groups: Database SG (sg-01c1ded2e50ecfcde), Active). The bottom right corner shows the AWS CloudShell icon and the date/time: 18-11-2024.

Using the endpoint of this database we can connect our mysql engine to instance to create any database and store the database.

## STEP 10: CREATE A FILE FOR OUTPUT

This Terraform output block is used to display the DNS name of the load balancer created in the Terraform configuration. To create the DNS name we need to create a file with the configurations. Am creating a outputs.tf file for DNS name.

```
root@ip-172-31-80-26:~  
  
output "lb_dns_name" {  
  description = "The DNS Name of the Load Balancer"  
  value=aws_lb.external-alb.dns_name  
}  
~
```

Outputs are often used to display useful information, such as resource IDs, URLs(DNS).

```
root@ip-172-31-80-26:  
  
Changes to outputs:  
+ lb_dns_name = (known after apply)  
  
Do you want to perform these actions?  
  Terraform will perform the actions described above.  
  Only 'yes' will be accepted to approve.  
  
Enter a value: yes  
  
aws_lb.external-alb: Creating...  
aws_lb_target_group.external-alb: Creating...  
aws_lb_target_group.external-alb: Creation complete after 1s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:targetgroup/ALB-TG/d4161e63c3d1356b]  
aws_lb_target_group_attachment.attachment1: Creating...  
aws_lb_target_group_attachment.attachment2: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:targetgroup/ALB-TG/d4161e63c3d1356b-20241125175341128500000001]  
aws_lb_target_group_attachment.attachment1: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:targetgroup/ALB-TG/d4161e63c3d1356b-20241125175341199800000002]  
aws_lb.external-alb: Still creating... [10s elapsed]  
aws_lb.external-alb: Still creating... [20s elapsed]  
aws_lb.external-alb: Still creating... [30s elapsed]  
aws_lb.external-alb: Still creating... [40s elapsed]  
aws_lb.external-alb: Still creating... [50s elapsed]  
aws_lb.external-alb: Still creating... [60s elapsed]  
aws_lb.external-alb: Still creating... [1m10s elapsed]  
aws_lb.external-alb: Still creating... [1m20s elapsed]  
aws_lb.external-alb: Still creating... [1m30s elapsed]  
aws_lb.external-alb: Still creating... [1m40s elapsed]  
aws_lb.external-alb: Still creating... [1m50s elapsed]  
aws_lb.external-alb: Still creating... [2m0s elapsed]  
aws_lb.external-alb: Still creating... [2m10s elapsed]  
aws_lb.external-alb: Still creating... [2m20s elapsed]  
aws_lb.external-alb: Still creating... [2m30s elapsed]  
aws_lb.external-alb: Still creating... [2m40s elapsed]  
aws_lb.external-alb: Still creating... [2m50s elapsed]  
aws_lb.external-alb: Still creating... [3m0s elapsed]  
aws_lb.external-alb: Still creating... [3m10s elapsed]  
aws_lb.external-alb: Still creating... [3m20s elapsed]  
aws_lb.external-alb: Creation complete after 3m23s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:loadbalancer/app/Externa-LB/d97b2fe95c88bb13]  
aws_lb_listener.external-alb: Creating...  
aws_lb_listener.external-alb: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-1:905418365089:listener/app/Externa-LB/d97b2fe95c88bb13/ef46e93e5cfce23e0]  
  
Apply complete! Resources: 5 added, 0 changed, 0 destroyed.  
  
Outputs:  
  
lb_dns_name = "Externa-LB-1844294251.us-eat-1.elb.amazonaws.com"  
[root@ip-172-31-80-26 ~]#
```

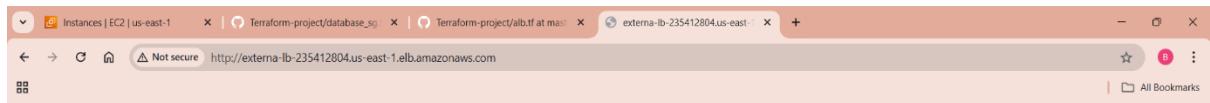
Here we can see the DNS for load balancer “Externa-LB-1844294251.us-eat-1.elb.amazonaws.com”. If we use this we can access the applications which are in the registered instances.

```
Apply complete! Resources: 20 added, 0 changed, 0 destroyed.
```

Outputs:

```
lb_dns_name = "External-LB-235412804.us-east-1.elb.amazonaws.com"  
[root@ip-172-31-28-241 ~]#
```

This is the DNS name of the load balancer.



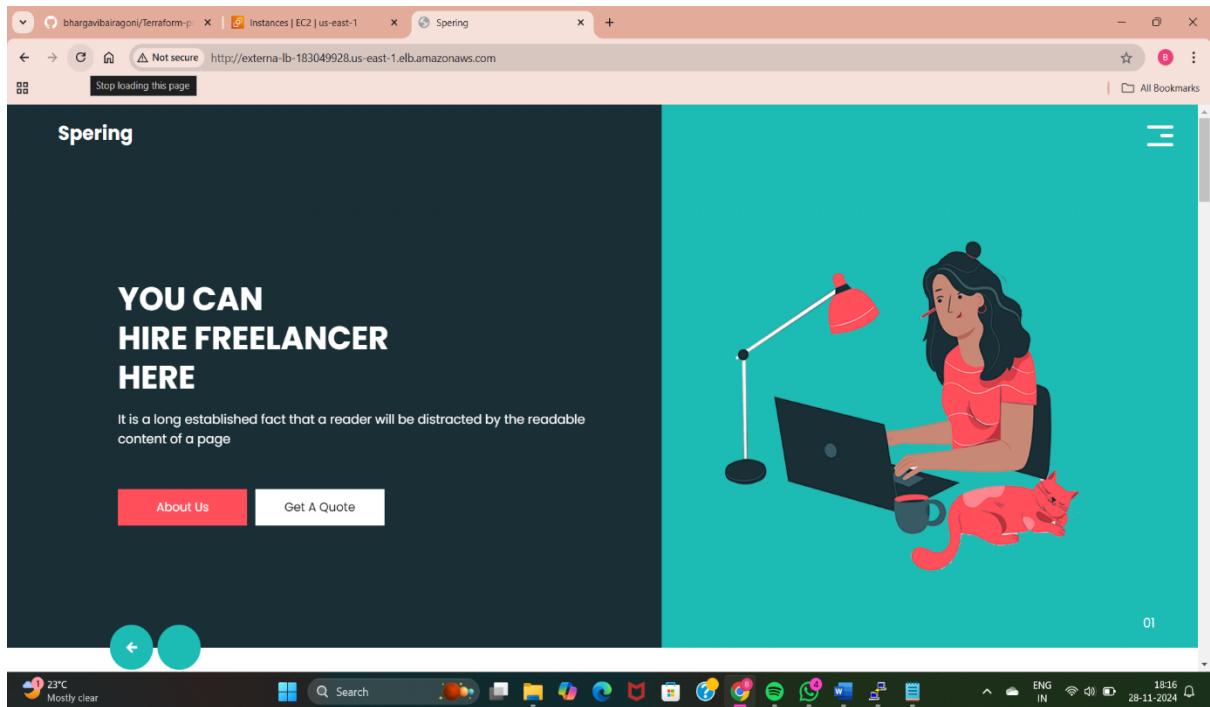
When am using the above DNS am getting this web application.

```
Apply complete! Resources: 20 added, 0 changed, 0 destroyed.
```

Outputs:

```
lb_dns_name = "External-LB-183049928.us-east-1.elb.amazonaws.com"
```

I have created new instances with the new configurations in the userdata and added the instances to the load balancer. Am getting this DNS for the load balancer.



By copying the above created load balancer DNS we can access the application .

#### **STEP11: CREATE A FILE FOR VARIABLE**

Instead of writing the same value multiple times (like an IP range or instance type), we can define it once as a variable and use multiple times. This makes our setup flexible and easier to update or use in different environments.

Here am creating a file for variables to give cidr block for vpc and subnet. The file is created with the name of vars.tf.

The vars.tf file is used to declare all the variables that can be referenced in other Terraform configuration files. This helps centralize variable definitions, making the code cleaner and easier to manage.

- ➔ We can change the CIDR blocks without modifying multiple places in your code.
- ➔ The same configuration can be reused in different environments (e.g., dev, test, prod) by changing the variable values.
- ➔ The code becomes cleaner and easier to understand.

```
root@ip-172-31-80-26:~  
variable "vpc_cidr" {  
  default = "10.0.0.0/16"  
}  
  
variable "subnet_cidr" {  
  default = "10.0.1.0/24"  
}  
  
variable "subnet1_cidr" {  
  default = "10.0.2.0/24"  
}  
  
variable "subnet2_cidr" {  
  default = "10.0.3.0/24"  
}  
  
variable "subnet3_cidr" {  
  default = "10.0.4.0/24"  
}  
  
variable "subnet4_cidr" {  
  default = "10.0.5.0/24"  
}  
  
variable "subnet5_cidr" {  
  default = "10.0.6.0/24"  
}
```

The `vpc_cidr` is a variable that holds a default CIDR block value for a VPC, `subnet_cidr` defines the CIDR block for a specific subnet within the VPC.

#### STEP12: CREATE A FILE FOR USER DATA

User data in the context of AWS EC2 instances refers to a script or set of commands provided to an instance at launch. It is used to automate tasks during the bootstrapping of the instance. This data can include shell scripts, or configuration scripts to set up and initialize software, install updates, or configure the operating system.

User data is a powerful way to automate the initialization of EC2 instances, making deployments more efficient and consistent.

```
root@ip-172-31-89-153:~  
#!/bin/bash  
yum update -y  
yum install -y httpd  
systemctl enable httpd  
systemctl start httpd  
echo "Hello World from EC2 instance  
This is Bhargavi  
Working on second project  
Second project is terraform $(hostname -f)" > /var/www/html/index.html  
  
~  
~
```

This is the userdata with the simple text. This will display the text mention in the echo.

```
root@ip-172-31-89-117:~  
#!/bin/bash  
# Update packages and install Apache and Git  
yum update -y  
yum install -y httpd  
systemctl start httpd  
systemctl enable httpd  
yum install git -y  
sudo git clone https://github.com/bhargavibairagoni/spering.git  
sudo mv spering/spering-html/* /var/www/html/  
~
```

This is the script for userdata with the name of “data.sh” Here am using a github project and cloning into the userdata and moving the files of the website to the apache document root directory.

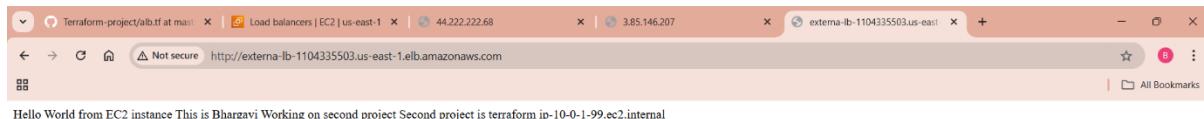
```
root@ip-172-31-89-117:~  
#!/bin/bash  
# Update packages and install Apache and Git  
yum update -y  
yum install -y httpd  
systemctl start httpd  
systemctl enable httpd  
yum install git -y  
sudo git clone https://github.com/bhargavibairagoni/seomark.git  
sudo mv seomark/seomark-html/* /var/www/html/  
~
```

This is the other file with the name of “data\_u.sh”. This clones the repository into the current working directory and files are moved to the directory served by Apache.

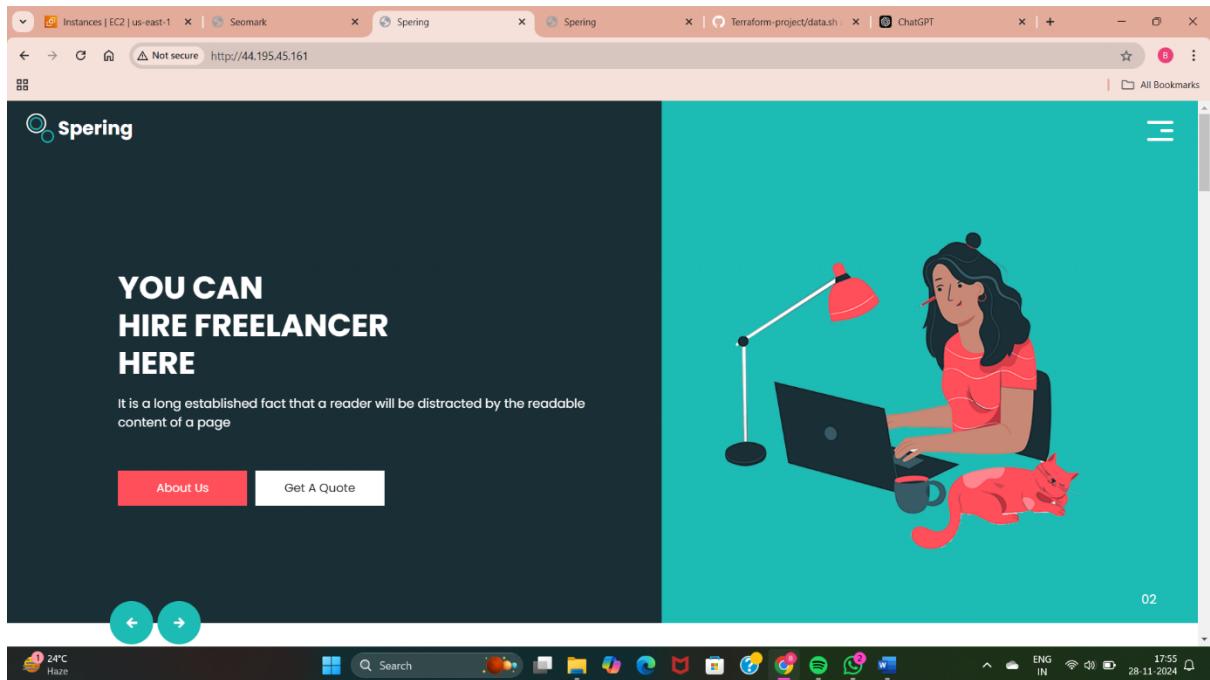
- Yum update -y : Updates the system's package repository and installs the latest updates for all installed packages.
- Installs the Apache HTTP server
- Configures Apache to start automatically whenever the instance is rebooted.
- Displays a basic welcome message by including the IP address of the instances
- Displays the websites which is cloned from the github.



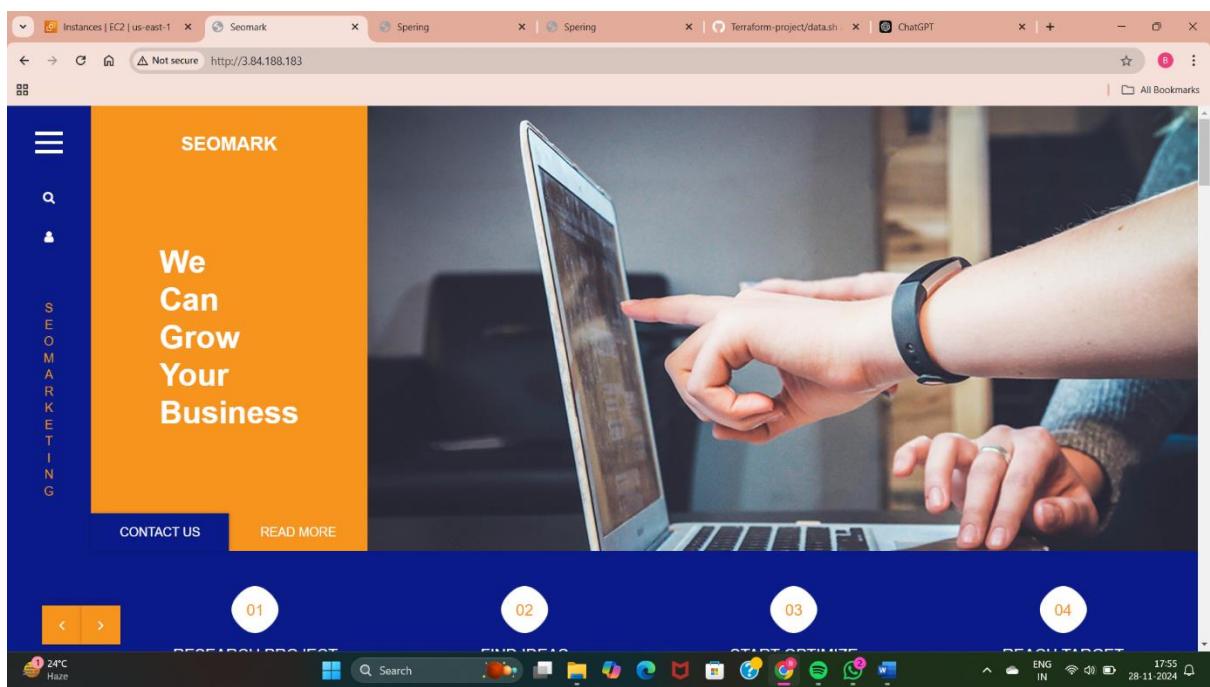
We can access the user data by using the public IP address of the instance.



This is the second instance in which I have used the user data.



This is the first website I have cloned from the github website. When I run the user data script (data.sh) one instance is created with the above website. When I taken the Public IP address and used this in the browser we can see and access the website.



This is the other application I have hosted using the userdata by specifying configurations in the userdata script(data\_u.sh)