

TERRAFORM PROJECT

TERRAFORM:

Terraform is an open-source infrastructure as code (IaC) tool created by HashiCorp that allows users to define and provision infrastructure using a high-level configuration language known as HashiCorp Configuration Language (HCL) or JSON.

KEY FEATURES OF TERRAFORM:

- **Infrastructure as Code:** Write and manage infrastructure in a declarative configuration file, allowing for version control and collaborative development.
- **Provider Ecosystem:** Terraform supports a wide range of cloud providers (like AWS, Azure, GCP) and other services, enabling users to manage both cloud and on-premises resources.
- **Resource Management:** Define resources such as virtual machines, databases, and networking components. Terraform can create, update, and delete resources based on the configuration.
- **Execution Plans:** Terraform generates an execution plan before applying changes, showing what actions it will take. This allows users to review and understand changes before they are executed.

COMMON USE CASES:

- **Provisioning Cloud Resources:** Automate the setup of virtual machines, storage, databases, and networking in cloud environments.
- **Multi-Cloud Deployment:** Manage resources across multiple cloud providers from a single configuration file.
- **Infrastructure Automation:** Streamline the process of infrastructure updates, scaling, and configuration management.
- **Collaboration and Versioning:** Use version control systems (like Git) to track changes in infrastructure configurations.

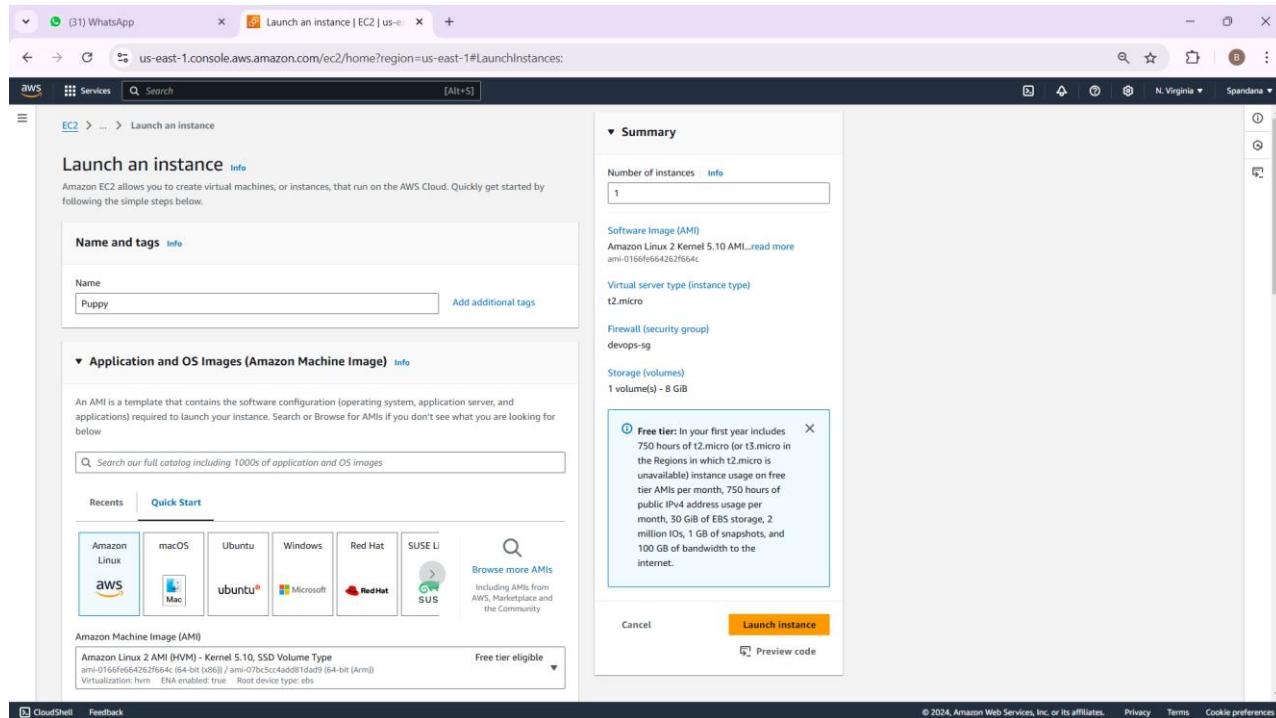
Terraform - infrastructure:

```
└── var.tf
└── vpc.tf
└── igw.tf
└── route_table_public.tf
└── ec2.tf
└── web_sg.tf
└── database_sg.tf
└── alb.tf
└── rds.tf
└── outputs.tf
```

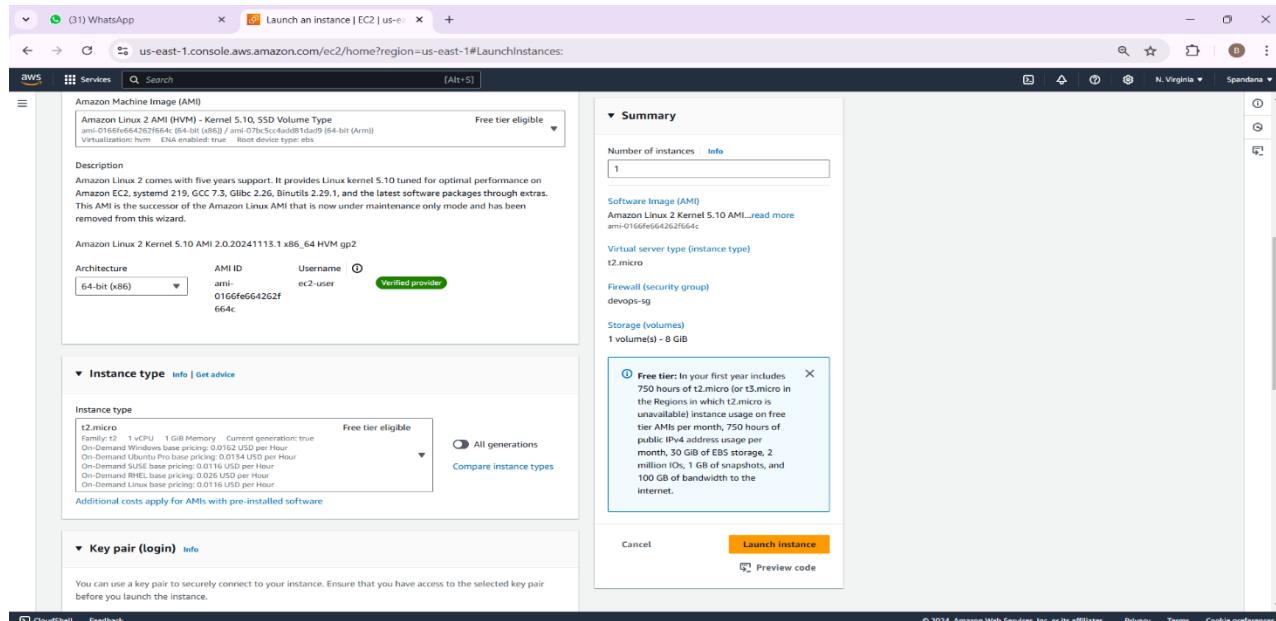
TERRAFORM COMMANDS:

- Terraform init
- Terraform validate
- Terraform plan
- Terraform apply
- Terraform show
- Terraform destroy

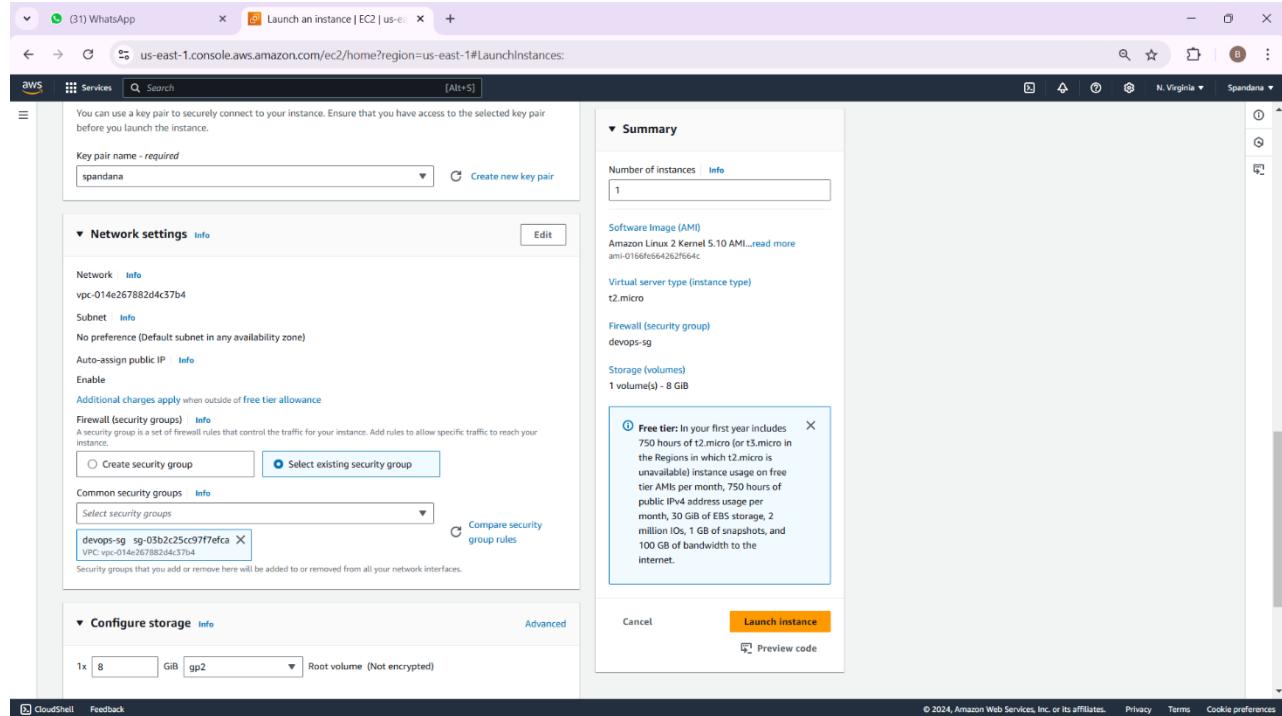
First step we need to create one instance using own configuration, On the EC2 Dashboard, click **Launch Instance**. Enter a name for your instance (e.g., "Puppy") .



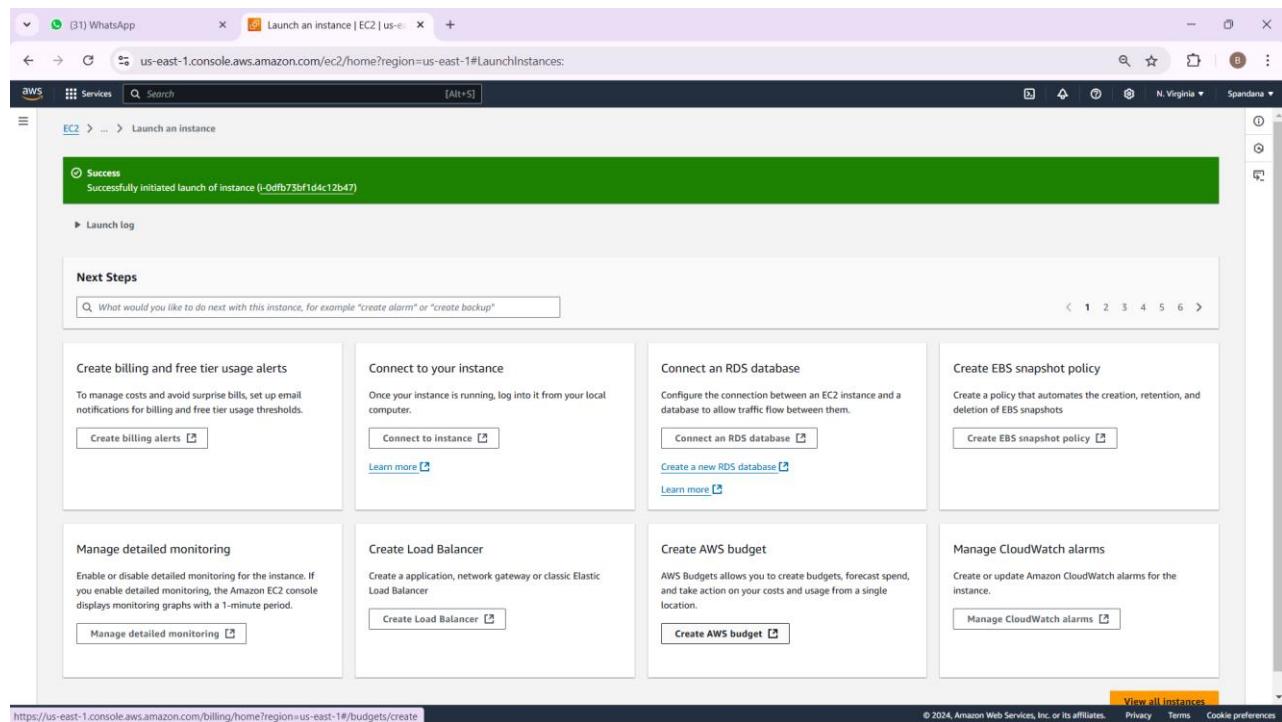
Choose an operating system for your instance, such as Amazon Linux, Ubuntu, or Windows Server based on our preference I have taken an amazon Linux, Select a free-tier eligible. Pick an instance type based on your requirements. For basic usage, choose **t2.micro**, which is free-tier eligible.



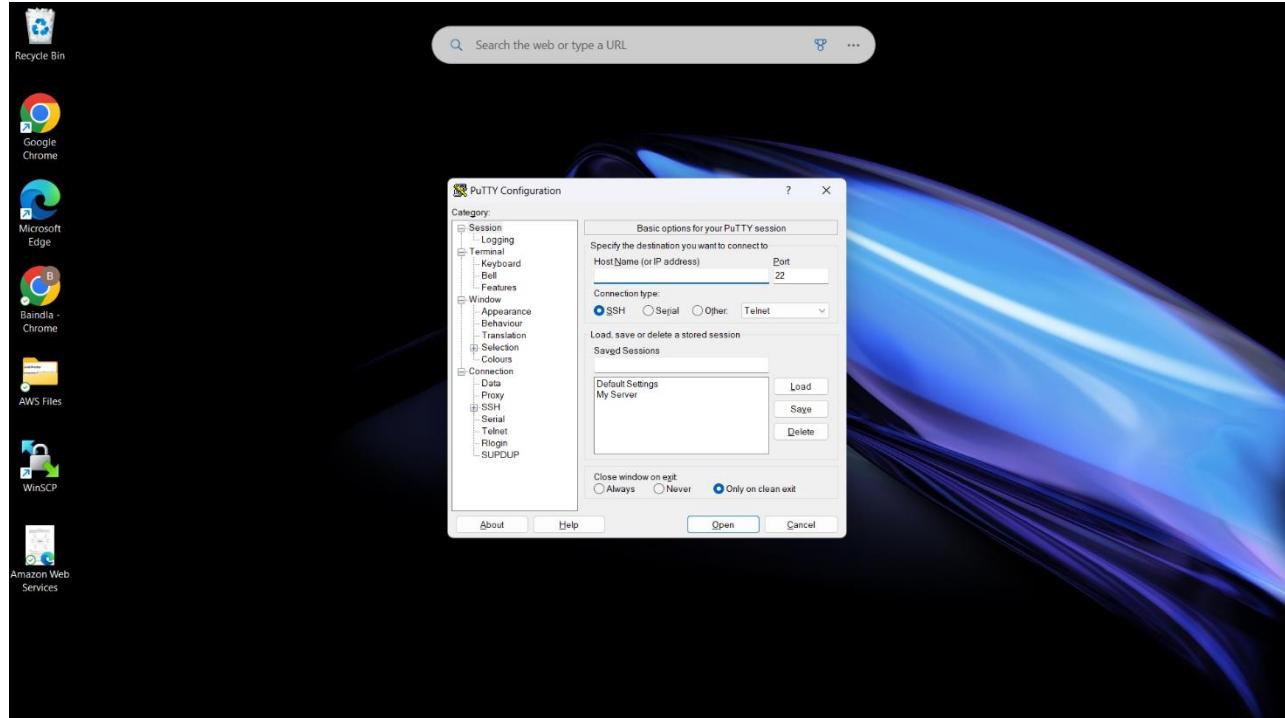
Select a key pair name if u don't have an key pair we have one option on the right hand side click on that and create a new key pair. I already have one key pair so I selected that key pair.



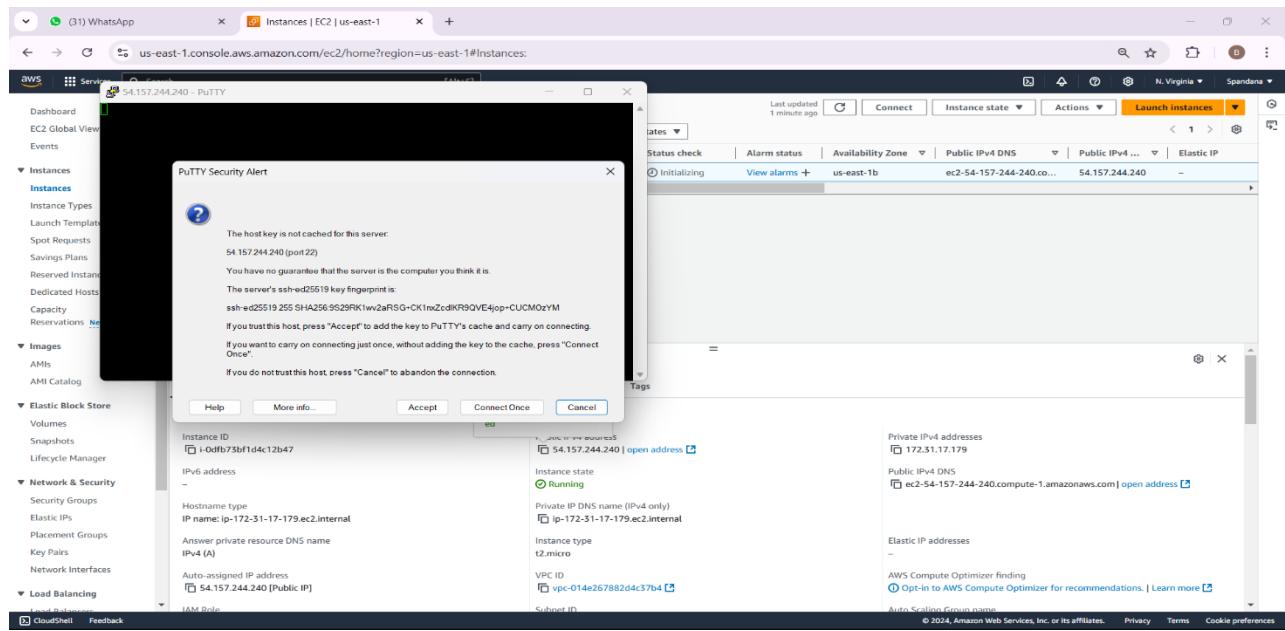
After configuring all the configurations then click on the launch instance.



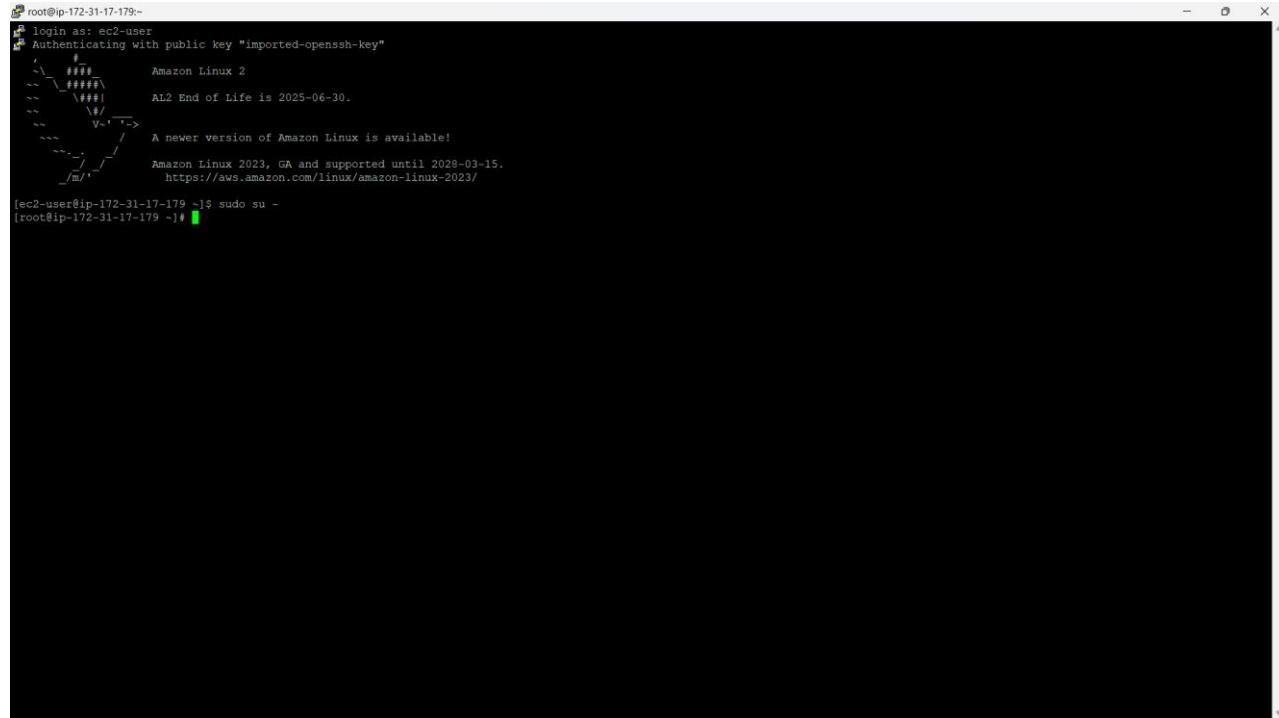
After creating an instance click on that instance in detail section we can able to see the public ip copy that public IP and open PuTTY. We can able to see basic option session of PuTTY in that we need to provide and Public IP or DNS of an instance.



After providing an public IP, we need to click on SSH we can able to see in the left side after that we can able to see Auth clicking on that we can able to see the credentials after clicking on that we need to provide .ppk file click on browse option directly it will navigate to the folder where we have saved .pk file select that file it directly open one terminal.



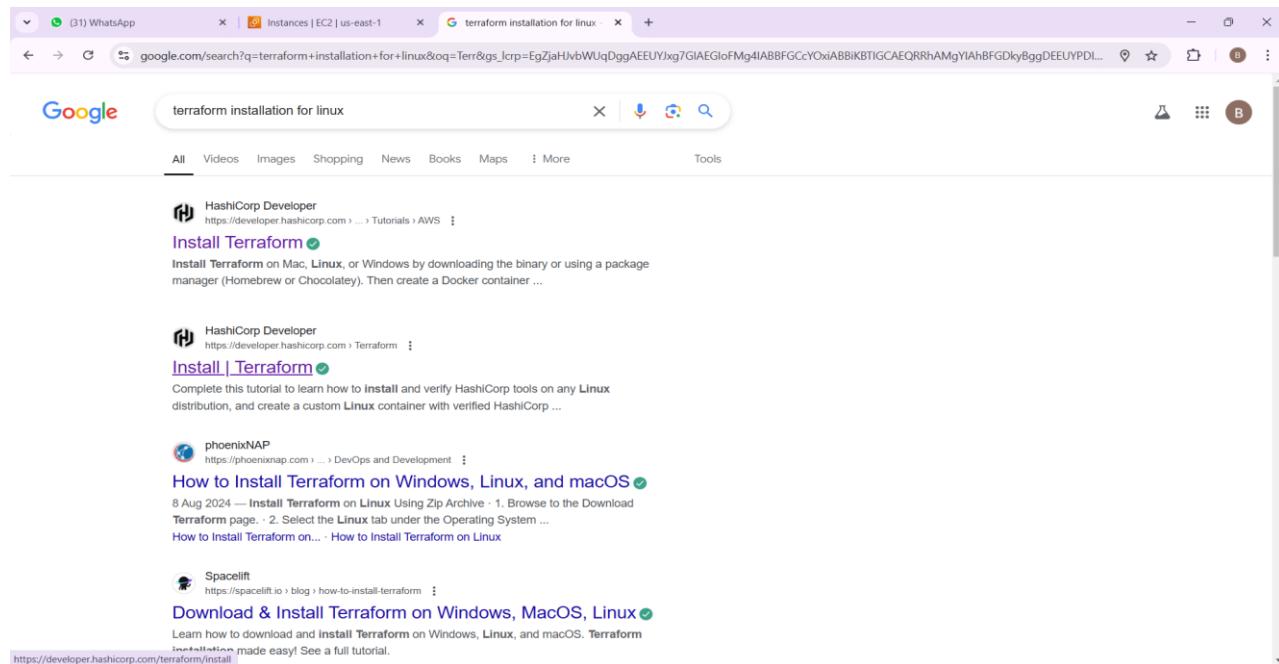
Click on accept button that shown in above page. We need to login as a ec2-user if we select amazon linux2 AMI, We need to login as based on our configuration of AMI.



```
root@ip-172-31-17-179:~#
login as: ec2-user
Authenticating with public key "imported-openssh-key"
Amazon Linux 2
AL2 End of Life is 2025-06-30.
A newer version of Amazon Linux is available!
Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/
[ec2-user@ip-172-31-17-179 ~]$ sudo su -
[root@ip-172-31-17-179 ~]#
```

STEP 1: CREATE A VARIABLES

If we want to create an any file first we need to install an terraform in our local machine so for that we need to visit an official website as sown in below.



Go to that website and click on amazon Linux as shown in below image click on that we can able to see some commands run that commands in the terminal and terraform has been installed successfully.

In the below image we can able to see that the downloading process of the terraform.

```
root@ip-172-31-17-179:~# sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
adding repo from: https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
grabbing file https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo to /etc/yum.repos.d/hashicorp.repo
[root@ip-172-31-17-179 ~]# sudo yum -y install terraform
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
hashicorp
hashicorp/x86_64/primary
hashicorp
Resolving Dependencies
--> Running transaction check
--> Package terraform.x86_64 0:1.9.8-1 will be installed
--> Processing Dependency: git for package: terraform-1.9.8-1.x86_64
--> Running transaction check
--> Package git.x86_64 0:2.40.1-1.amzn2.0.3 will be installed
--> Processing Dependency: git-core = 2.40.1-1.amzn2.0.3 for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Processing Dependency: git-core-doc = 2.40.1-1.amzn2.0.3 for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Processing Dependency: perl-Git = 2.40.1-1.amzn2.0.3 for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Processing Dependency: perl-Git for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Processing Dependency: perl-Term::ReadKey for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Running transaction check
--> Package git-core.x86_64 0:2.40.1-1.amzn2.0.3 will be installed
--> Package git-core-doc.noarch 0:2.40.1-1.amzn2.0.3 will be installed
--> Package perl-Git.noarch 0:2.40.1-1.amzn2.0.3 will be installed
--> Package perl-TermReadKey.x86_64 0:2.30-20.amzn2.0.2 will be installed
--> Running transaction check
--> Package perl-Noarch 1:0.17020-2.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

-----
Package           Arch      Version            Repository        Size
-----
Installing:
terraform         x86_64   1.9.8-1           hashicorp       27 M
Installing for dependencies:
git               x86_64   2.40.1-1.amzn2.0.3    amzn2-core     54 k
git-core          x86_64   2.40.1-1.amzn2.0.3    amzn2-core     10 M
git-core-doc      noarch   2.40.1-1.amzn2.0.3    amzn2-core     3.0 M
perl-error        noarch   1:0.17020-2.amzn2    amzn2-core     32 k
perl-git          noarch   2.40.1-1.amzn2.0.3    amzn2-core     42 k
perl-TermReadKey x86_64   2.30-20.amzn2.0.2  amzn2-core     31 k

Transaction Summary
-----
Install 1 Package (+6 Dependent packages)

Total download size: 40 M
Installed size: 129 M
```

```

root@ip-172-31-17-179:~#
[root@ip-172-31-17-179 ~]# sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
adding repo from: https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo
grabbing file https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo to /etc/yum.repos.d/hashicorp.repo
repo saved to /etc/yum.repos.d/hashicorp.repo
[root@ip-172-31-17-179 ~]# sudo yum -y install terraform
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
hashicorp
hashicorp/x86_64/primary
hashicorp
Resolving Dependencies
--> Running transaction check
--> Package terraform.x86_64 0:1.9.8-1 will be installed
--> Processing Dependency: git for package: terraform-1.9.8-1.x86_64
--> Running transaction check
--> Package git.x86_64 2.40.1-1.amzn2.0.3 will be installed
--> Processing Dependency: git-core = 2.40.1-1.amzn2.0.3 for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Processing Dependency: git-core-doc = 2.40.1-1.amzn2.0.3 for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Processing Dependency: perl-Git = 2.40.1-1.amzn2.0.3 for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Processing Dependency: perl(Git) for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Processing Dependency: perl(TermReadKey) for package: git-2.40.1-1.amzn2.0.3.x86_64
--> Running transaction check
--> Package git-core.x86_64 0:2.40.1-1.amzn2.0.3 will be installed
--> Package git-core-doc.noarch 0:2.40.1-1.amzn2.0.3 will be installed
--> Package perl-Git.noarch 0:2.40.1-1.amzn2.0.3 will be installed
--> Processing Dependency: perl(Error) for package: perl-Git-2.40.1-1.amzn2.0.3.noarch
--> Package perl-TermReadKey.x86_64 0:1.20.30-20.amzn2.0.2 will be installed
--> Running transaction check
--> Package perl-Error.noarch 1:0.17020-2.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch      Version            Repository        Size
=====
Installing:
terraform          x86_64   1.9.8-1           hashicorp       27 M
Installing for dependencies:
git                x86_64   2.40.1-1.amzn2.0.3    amzn2-core      54 k
git-core           x86_64   2.40.1-1.amzn2.0.3    amzn2-core      10 M
git-core-doc        noarch   2.40.1-1.amzn2.0.3    amzn2-core      3.0 M
perl-Error          noarch   1:0.17020-2.amzn2   amzn2-core      32 k
perl-git            noarch   2.40.1-1.amzn2.0.3    amzn2-core      42 k
perl-TermReadKey   x86_64   2.30-20.amzn2.0.2   amzn2-core      31 k

Transaction Summary
=====
Install 1 Package (+6 Dependent packages)

Total download size: 40 M
Installed size: 129 M

=====

```

In the below image we can able to see that terraform installation completed successfully.

```

root@ip-172-31-17-179:~#
Transaction Summary
=====
Install 1 Package (+6 Dependent packages)

Total download size: 40 M
Installed size: 129 M
Downloading packages:
(1/7): git-2.40.1-1.amzn2.0.3.x86_64.rpm                                | 54 kB 00:00:00
(2/7): git-core-doc-2.40.1-1.amzn2.0.3.noarch.rpm                         | 3.0 MB 00:00:00
(3/7): git-core-2.40.1-1.amzn2.0.3.x86_64.rpm                            | 10 MB 00:00:00
(4/7): perl-Error-0.17020-2.amzn2.noarch.rpm                         | 32 kB 00:00:00
(5/7): perl-TermReadKey-2.30-20.amzn2.0.2.x86_64.rpm                     | 31 kB 00:00:00
(6/7): perl-Git-2.40.1-1.amzn2.0.3.noarch.rpm                         | 42 kB 00:00:00
(7/7): terraform-1.9.8-1.x86_64.rpm                                     | 27 MB 00:00:00
                                                               63 MB/s | 40 MB 00:00:00

Total
Retrieving key from https://rpm.releases.hashicorp.com/gpg
Importing GPG key 0x462E701:
  Userid : "Hashicorp Security (Hashicorp Package Signing) <security+packaging@hashicorp.com>"
  Fingerprint: 798a ecf5 4e5c 1542 8c9e 42ee aa16 fcbc a621 e701
  From: https://rpm.releases.hashicorp.com/gpg
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : git-core-2.40.1-1.amzn2.0.3.x86_64                               1/7
  Installing : git-core-doc-2.40.1-1.amzn2.0.3.noarch                          2/7
  Installing : perl-Error-0.17020-2.amzn2.noarch                           3/7
  Installing : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64                      4/7
  Installing : perl-Git-2.40.1-1.amzn2.0.3.noarch                          5/7
  Installing : git-2.40.1-1.amzn2.0.3.x86_64                                6/7
  Verifying : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64                      1/7
  Verifying : git-2.40.1-1.amzn2.0.3.x86_64                                2/7
  Verifying : perl-Error-0.17020-2.amzn2.noarch                           3/7
  Verifying : git-core-2.40.1-1.amzn2.0.3.x86_64                          4/7
  Verifying : git-core-doc-2.40.1-1.amzn2.0.3.noarch                        5/7
  Verifying : perl-Git-2.40.1-1.amzn2.0.3.noarch                         6/7
  Installed: terraform.x86_64 0:1.9.8-1                                    7/7

Dependency Installed:
  git.x86_64 0:2.40.1-1.amzn2.0.3          git-core.x86_64 0:2.40.1-1.amzn2.0.3          git-core-doc.noarch 0:2.40.1-1.amzn2.0.3          perl-Error.noarch 1:0.17020-2.amzn2

Complete!
[root@ip-172-31-17-179 ~]# [root@ip-172-31-17-179 ~]# 

```

Keeps your configuration clean and organized. We need to create a variables file **vars.tf** which allows us to update values in one place without changing multiple files. Makes it easier to reuse configurations for different environments This setup makes our Terraform setup flexible and manageable, especially for large projects. After that we need to initiate the terraform then we need to run the **vars.tf** using **terraform plan**. After that we need to apply it using **terraform apply** command.

```
root@ip-172-31-80-26: ~
variable "vpc_cidr" {
  default = "10.0.0.0/16"
}

variable "subnet1_cidr" {
  default = "10.0.1.0/24"
}

variable "subnet1_cidr2" {
  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"
}

#
```

STEP 2: CREATING A FILE FOR VPC

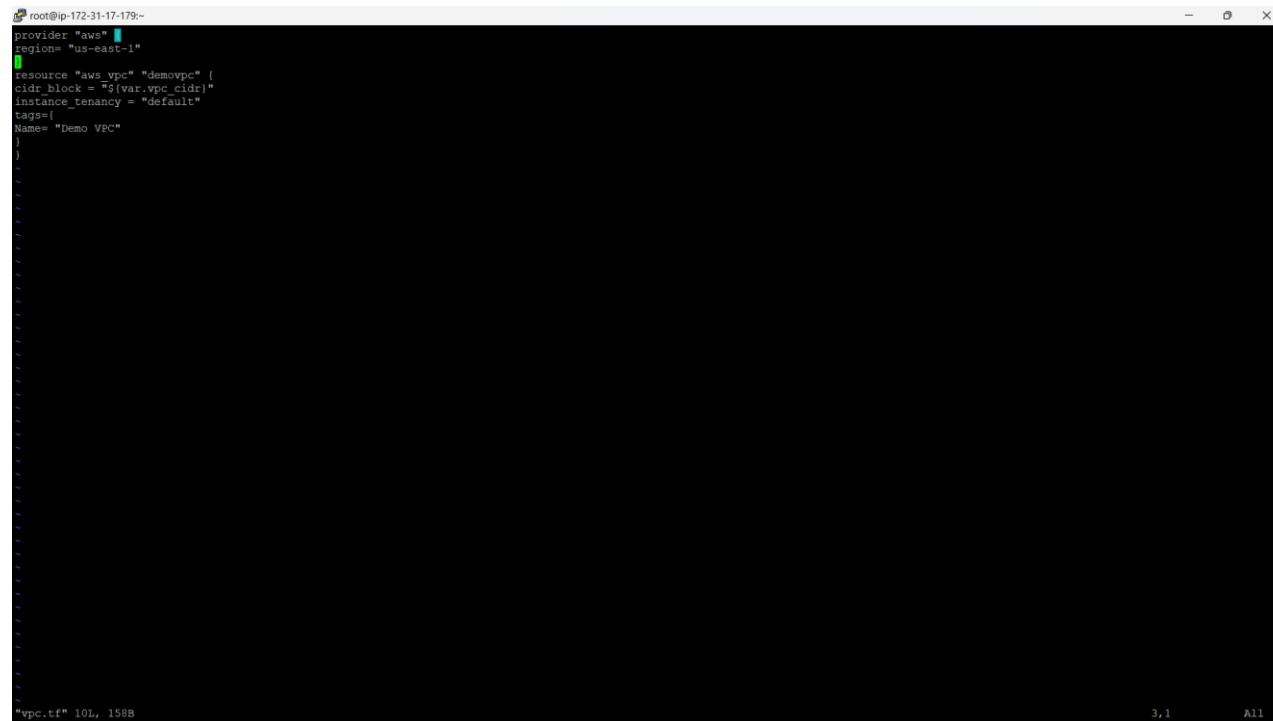
To interact with AWS using Terraform, we need two key files: provider.tf and vpc.tf. These files ensure Terraform can authenticate to your AWS account and create the required resources, like a Virtual Private Cloud (VPC). The provider.tf file defines the provider (in this case, AWS) and sets up access credentials. It acts as a bridge between Terraform and AWS.

Note: Storing keys directly in the file is not secure. For production, use environment variables or AWS CLI credentials instead.

After providing credentials in CLI we can able to access the AWS account for creating an services using terraform.

If we want to create an VPC in the AWS account first we need to create an `vpc.tf` file using **vim vpc.tf** and then we need to write a script for the creation of the VPC.

Specifies the resource type (`aws_vpc`) and a logical name (`demovpc`). `cidr_block` defines the range of IP addresses for the VPC (e.g., "10.0.0.0/16"). The value is dynamic and passed using a variable (`var.vpc_cidr`) to allow flexibility. `instance_tenancy` is set to "default", meaning EC2 instances launched in the VPC will share hardware with other AWS customers unless specified otherwise. The `tags` section assigns a name (Demo vpc) to the VPC for easy identification in the AWS Management Console.



```
root@ip-172-31-17-179:~$ vim vpc.tf
provider "aws" {
  region = "us-east-1"
}

resource "aws_vpc" "demovpc" {
  cidr_block = "${var.vpc_cidr}"
  instance_tenancy = "default"
  tags = {
    Name = "Demo VPC"
  }
}

vpc.tf: 10L, 158B
```

After creating an `vpc.tf` file then we need to run the **terraform plan** command because the terraform plan command is used to preview the changes in Terraform will make to your infrastructure. It shows you what will be added, modified, or deleted based on your Terraform configuration files, without actually making any changes where Terraform tells you what it's about to do, and you can check if everything looks good before applying the changes within the file before applying changes.

After running an **terraform plan** the below process will be run. Then we need to run **terraform apply** then that can be applied.

```
[root@ip-172-31-17-179 ~]# vim vpc.tf
[root@ip-172-31-17-179 ~]# 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
  + cidr_block
  + default_network_acl_id
  + default_route_table_id
  + default_security_group_id
  + dhcp_options_id
  + enable_dns_hostnames
  + enable_dns_support
  + enable_network_address_usage_metrics
  + id
  + instance_tenancy
  + ipv6_association_id
  + ipv6_cidr_block
  + ipv6_cidr_block_network_border_group
  + main_route_table_id
  + owner_id
  + 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-172-31-17-179 ~]# 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)
}
```

In the below image we can able to see the terraform apply has been completed successfully then go to the AWS account and check we can able to see VPC has been created successfully in our AWS account.

```
[root@ip-172-31-17-179 ~]

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-172-31-17-179 ~]# 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
  + cidr_block
  + default_network_acl_id
  + default_route_table_id
  + default_security_group_id
  + dhcp_options_id
  + enable_dns_hostnames
  + enable_dns_support
  + enable_network_address_usage_metrics
  + id
  + instance_tenancy
  + ipv6_association_id
  + ipv6_cidr_block
  + ipv6_cidr_block_network_border_group
  + main_route_table_id
  + owner_id
  + 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-0ebc47832fcff0045]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-17-179 ~]# vim vpc.tf
[root@ip-172-31-17-179 ~]#
```

In the below image we can able to see that newly created VPC has been created.

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set	Main route table
Demo VPC	vpc-03922f43caeeb447e	Available	10.0.0.0/16	-	dopt-08010d2dc2ef4a5ce	rtb-067aba57ab7984573
	vpc-014e267882d4c37bd	Available	172.31.0.0/16	-	dopt-08010d2dc2ef4a5ce	rtb-02a08c64822c08da8

STEP 3: CREATING A FILE FOR SUBNETS

After creating the VPC, the next step is to create subnets within that VPC. I have created **subnet.tf** file for Subnets allow you to organize your network into smaller sections, each with its own CIDR block. I have created 6 subnets for an existing VPC. The subnet is linked to an existing VPC by specifying the `vpc_id`.

```
root@ip-172-31-35-214:~#
resource "aws_subnet" "public_subnet-1" {
  vpc_id = "${aws_vpc.demoVpc.id}"
  cidr_block = "${var.subnet1_cidr}"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1a"
  tags = [
    Name = "Web Subnet 1"
  ]
}
resource "aws_subnet" "public_subnet-2" {
  vpc_id = "${aws_vpc.demoVpc.id}"
  cidr_block = "${var.subnet2_cidr}"
  map_public_ip_on_launch = true
  availability_zone = "us-east-1b"
  tags = [
    Name = "Web subnet 2"
  ]
}
resource "aws_subnet" "application_subnet-1" {
  vpc_id = "${aws_vpc.demoVpc.id}"
  cidr_block = "${var.subnet3_cidr}"
  map_public_ip_on_launch = false
  availability_zone = "us-east-1a"
  tags = [
    Name = "Application Subnet 1"
  ]
}
resource "aws_subnet" "application-subnet-2" {
  vpc_id = "${aws_vpc.demoVpc.id}"
  cidr_block = "${var.subnet4_cidr}"
  map_public_ip_on_launch = false
  availability_zone = "us-east-1b"
  tags = [
    Name = "Application Subnet 2"
  ]
}
resource "aws_subnet" "database-subnet-1" {
  vpc_id = "${aws_vpc.demoVpc.id}"
  cidr_block = "${var.subnet5_cidr}"
  availability_zone = "us-east-1a"
  tags = [
    Name = "Database Subnet 1"
  ]
}
resource "aws_subnet" "database-subnet-2" {
  vpc_id = "${aws_vpc.demoVpc.id}"
  cidr_block = "${var.subnet6_cidr}"
  availability_zone = "us-east-1b"
  tags = [
    Name = "Database subnet 2"
  ]
}
```

Each subnet needs its own IP range, so you define a CIDR block (like 10.0.1.0/24). This places the subnet in a specific availability zone. This is set to true to assign public IPs to instances launched in this subnet, making it a public subnet. Adds a tag (Name = "My Subnet") for easier identification in the AWS console. After creating an subnet.tf file we need to run the terraform plan command.

```
[root@ip-172-31-35-214 ~]# vim subnet.tf
[root@ip-172-31-35-214 ~]# terraform plan
aws_vpc.demoVPC: Refreshing state... [id=vpc-03922f43caeeb447e]

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 = (known after apply)
    assign_ipv6_address_on_creation = false
    availability_zone = "us-east-1b"
    availability_zone_id = (known after apply)
    cidr_block = "10.0.4.0/24"
    enable_dns64 = false
    enable_resource_name_dns_a_record_on_launch = false
    enable_resource_name_dns_aaaa_record_on_launch = false
    id = (known after apply)
    ipv6_cidr_block_association_id = (known after apply)
    ipv6_native = false
    map_public_ip_on_launch = false
    owner_id = (known after apply)
    private_dns_hostname_type_on_launch = (known after apply)
    tags = [
        + "Name" = "Application Subnet 2"
    ]
    tags_all = [
        + "Name" = "Application Subnet 2"
    ]
    vpc_id = "vpc-03922f43caeeb447e"
}

# aws_subnet.application_subnet-1 will be created
+ resource "aws_subnet" "application_subnet-1" {
    arn = (known after apply)
    assign_ipv6_address_on_creation = false
    availability_zone = "us-east-1a"
    availability_zone_id = (known after apply)
    cidr_block = "10.0.3.0/24"
    enable_dns64 = false
    enable_resource_name_dns_a_record_on_launch = false
    enable_resource_name_dns_aaaa_record_on_launch = false
    id = (known after apply)
    ipv6_cidr_block_association_id = (known after apply)
    ipv6_native = false
    map_public_ip_on_launch = false
    owner_id = (known after apply)
    private_dns_hostname_type_on_launch = (known after apply)
    tags = [
        + "Name" = "Application Subnet 1"
    ]
}
```

It will check every subnet configuration is correct or not.

```
[root@ip-172-31-35-214 ~]
+ "Name" = "Application Subnet 1"
+ vpc_id = "vpc-03922f43caeeb447e"
}

# 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
    enable_resource_name_dns_aaaa_record_on_launch = false
    id = (known after apply)
    ipv6_cidr_block_association_id = (known after apply)
    ipv6_native = false
    map_public_ip_on_launch = false
    owner_id = (known after apply)
    private_dns_hostname_type_on_launch = (known after apply)
    tags = [
        + "Name" = "Database Subnet 1"
    ]
    tags_all = [
        + "Name" = "Database Subnet 1"
    ]
    vpc_id = "vpc-03922f43caeeb447e"
}

# 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_a_record_on_launch = false
    enable_resource_name_dns_aaaa_record_on_launch = false
    id = (known after apply)
    ipv6_cidr_block_association_id = (known after apply)
    ipv6_native = false
    map_public_ip_on_launch = false
    owner_id = (known after apply)
    private_dns_hostname_type_on_launch = (known after apply)
    tags = [
        + "Name" = "Database Subnet 2"
    ]
    tags_all = [
        + "Name" = "Database subnet 2"
    ]
}
```

```

root@ip-172-31-35-214:~#
+ 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
+   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           = true
+   owner_id                           = (known after apply)
+   private_dns_hostname_type_on_launch = (known after apply)
+   tags
+     + "Name" = "Web Subnet 1"
+   tags_all                          = [
+     + "Name" = "Web Subnet 1"
+   ]
+   vpc_id                            = "vpc-03922f43caeeb447e"
}

# aws_subnet.public_subnet-1 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                       = false
+   map_public_ip_on_launch           = true
+   owner_id                           = (known after apply)
+   private_dns_hostname_type_on_launch = (known after apply)
+   tags
+     + "Name" = "Web Subnet 2"
+   tags_all                          = [
+     + "Name" = "Web subnet 2"
+   ]
+   vpc_id                            = "vpc-03922f43caeeb447e"
}

Plan: 6 to add, 0 to change, 0 to destroy.

```

After successfully completion of terraform plan then we need to apply using terraform apply command we can see in the below image.

```

root@ip-172-31-35-214:~#
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-172-31-35-214 ~]# terraform apply
aws_vpc.demoVPC: Refreshing state... [id=vpc-03922f43caeeb447e]

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                               = (known after apply)
+   assign_ipv6_address_on_creation    = false
+   availability_zone                 = "us-east-1b"
+   availability_zone_id              = (known after apply)
+   cidr_block                       = "10.0.4.0/24"
+   enable_dns64                      = false
+   enable_resource_name_dns_a_record_on_launch = false
+   enable_resource_name_dns_aaaa_record_on_launch = false
+   id                                = (known after apply)
+   ipv6_cidr_block_association_id    = (known after apply)
+   ipv6_native                       = false
+   map_public_ip_on_launch           = false
+   owner_id                           = (known after apply)
+   private_dns_hostname_type_on_launch = (known after apply)
+   tags
+     + "Name" = "Application Subnet 2"
+   tags_all                          = [
+     + "Name" = "Application Subnet 2"
+   ]
+   vpc_id                            = "vpc-03922f43caeeb447e"
}

# aws_subnet.application-subnet-1 will be created
+ resource "aws_subnet" "application-subnet-1" {
+   arn                               = (known after apply)
+   assign_ipv6_address_on_creation    = false
+   availability_zone                 = "us-east-1a"
+   availability_zone_id              = (known after apply)
+   cidr_block                       = "10.0.3.0/24"
+   enable_dns64                      = false
+   enable_resource_name_dns_a_record_on_launch = false
+   enable_resource_name_dns_aaaa_record_on_launch = false
+   id                                = (known after apply)
+   ipv6_cidr_block_association_id    = (known after apply)
+   ipv6_native                       = false
+   map_public_ip_on_launch           = false
+   owner_id                           = (known after apply)
+   private_dns_hostname_type_on_launch = (known after apply)

```

```

root@ip-172-31-35-214:~#
}
# aws_subnet.database-subnet-1 will be created
+ resource "aws_subnet" "database-subnet-1" {
+   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" = "Database Subnet 1"
+   tags_all
+   + "Name" = "Database Subnet 1"
}
+ vpc_id
= "vpc-03922f43caeeb447e"

# 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_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" = "Database subnet 2"
+   tags_all
+   + "Name" = "Database subnet 2"
}
+ vpc_id
= "vpc-03922f43caeeb447e"

```

```

root@ip-172-31-35-214:~#
}
+ tags_all
+ + "Name" = "Database subnet 2"
}
+ vpc_id
= "vpc-03922f43caeeb447e"

# aws_subnet.public_subnet-1 will be created
+ resource "aws_subnet" "public_subnet-1" {
+   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" = "Web Subnet 1"
}
+ tags_all
+ + "Name" = "Web Subnet 1"
}
+ vpc_id
= "vpc-03922f43caeeb447e"

# 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
+   map_public_ip_on_launch
+   owner_id
+   private_dns_hostname_type_on_launch
+   tags
+   + "Name" = "Web subnet 2"
}
+ tags_all
= [

```

In the below image we can able to see that terraform apply was successful. Then we need to go AWS account and search for VPC in that we can able to see subnets click on that and check in the subnets.

```
root@ip-172-31-35-214:~#
# 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                       = false
    + map_public_ip_on_launch           = true
    + owner_id                          = (known after apply)
    + private_dns_hostname_type_on_launch = (known after apply)
    + tags
      + "Name" = "Web subnet 2"
    + tags_all                         = [
        + "Name" = "Web subnet 2"
      ]
    + vpc_id                            = "vpc-03922f43caeeb447e"
  }

Plan: 6 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-1: Creating...
aws_subnet.application-subnet-1: Creating...
aws_subnet.public-subnet-2: Creating...
aws_subnet.application-subnet-2: Creating...
aws_subnet.public-subnet-1: Creating...
aws_subnet.database-subnet-1: Creation complete after 1s [id=subnet-0a03df81f43dd4fbe]
aws_subnet.application-subnet-2: Creation complete after 1s [id=subnet-0c75f48d9411b7dcb]
aws_subnet.database-subnet-2: Creation complete after 1s [id=subnet-0a9b7ff3f3ce7b0b2d]
aws_subnet.application-subnet-1: Creation complete after 1s [id=subnet-050b74d17f913947d]
aws_subnet.public-subnet-2: Still creating... [10s elapsed]
aws_subnet.public-subnet-1: Still creating... [10s elapsed]
aws_subnet.public-subnet-1: Creation complete after 12s [id=subnet-0e3fd1ae4d150bed]
aws_subnet.public-subnet-2: Creation complete after 12s [id=subnet-08950bb909c05f17b]

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.
[root@ip-172-31-35-214 ~] vim subnet.tf
[root@ip-172-31-35-214 ~] #
```

We can able to see that our first web subnet in the below image.

The screenshot shows the AWS VPC Subnets console with the following details:

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	IPv6 CIDR association ID
subnet-085ac199r0740bf9	Available	vpc-014e267882d4c37b4	172.31.16.0/20	-	-	-
subnet-05b7cfca1a1fe49e8e	Available	vpc-014e267882d4c37b4	172.31.32.0/20	-	-	-
subnet-0c060992779b0d64	Available	vpc-014e267882d4c37b4	172.31.64.0/20	-	-	-
Web subnet 2	Available	vpc-03922f43caeeb447e Demo...	10.0.2.0/24	-	-	-
subnet-08950bb909c05f17b	Available	vpc-014e267882d4c37b4	172.31.48.0/20	-	-	-
subnet-003fa9df930e7bd3	Available	vpc-014e267882d4c37b4	10.0.3.0/24	-	-	-
Application Subnet 1	Available	vpc-03922f43caeeb447e Demo...	10.0.5.0/24	-	-	-
Database Subnet 1	Available	vpc-03922f43caeeb447e Demo...	10.0.1.0/24	-	-	-
Web Subnet 1	Available	vpc-03922f43caeeb447e Demo...	10.0.1.0/24	-	-	-
Application Subnet 2	Available	vpc-03922f43caeeb447e Demo...	10.0.4.0/24	-	-	-

Details for Web Subnet 1:

- Subnet ID: subnet-0e3fd1ae4d150bed
- Subnet ARN: arn:aws:ec2:us-east-1:53742550105:subnet/subnet-0e3fd1ae4d150bed
- State: Available
- IPv4 CIDR: 10.0.1.0/24
- IPv6 CIDR: -
- IPv6 CIDR association ID: -
- Availability Zone: us-east-1a
- VPC: vpc-03922f43caeeb447e | Demo Demo
- Route table: rtb-067ab57ab7984573
- Auto-assign public IPv4 address: Yes
- Auto-assign IPv6 address: No
- IPV4 CIDR reservations: -
- Customer-owned IPv4 pool: -

This is the Web subnet 2.

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	IPv6 CIDR association ID
Web subnet 2	subnet-08950bb909c05f17b	Available	vpc-03922f43caebe447e Demo...	10.0.2.0/24	-	-
	subnet-0038fa9df930e78d3	Available	vpc-014e267882d4c37b4	172.31.48.0/20	-	-
	subnet-05b7fc1a1fe49e8e	Available	vpc-014e267882d4c37b4	172.31.32.0/20	-	-
	subnet-0x0a60992779b0d64	Available	vpc-014e267882d4c37b4	172.31.64.0/20	-	-
	subnet-085ac199c0740b9f	Available	vpc-014e267882d4c37b4	172.31.16.0/20	-	-

subnet-08950bb909c05f17b / Web subnet 2

Details

Subnet ID	subnet-08950bb909c05f17b	Subnet ARN	arn:aws:ec2:us-east-1:637423550105:subnet/subnet-08950bb909c05f17b	State	Available	IPv4 CIDR	10.0.2.0/24
Available IPv4 addresses	251					IPv6 CIDR association ID	-
Availability Zone ID	use1-az2					Availability Zone	us-east-1b
Network ACL	acl-0d37e77e5d73fc140	Network border group	us-east-1	VPC	vpc-03922f43caebe447e Demo VPC	Route table	rtb-067aba57ab7984573
Auto-assign customer-owned IPv4 address	No	Default subnet	No	Auto-assign public IPv4 address	Yes	Auto-assign IPv6 address	No
				Outpost ID	-	IPv4 CIDR reservations	-
				Customer-owned IPv4 pool	-		

Application subnet 1 has been created successfully as shown in below image.

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	IPv6 CIDR association ID
Application Subnet 1	subnet-050b74d17f913947d	Available	vpc-03922f43caebe447e Demo...	10.0.3.0/24	-	-
	subnet-0038fa9df930e78d3	Available	vpc-014e267882d4c37b4	172.31.48.0/20	-	-
	subnet-05b7fc1a1fe49e8e	Available	vpc-014e267882d4c37b4	172.31.32.0/20	-	-
	subnet-0x0a60992779b0d64	Available	vpc-014e267882d4c37b4	172.31.64.0/20	-	-
	subnet-085ac199c0740b9f	Available	vpc-014e267882d4c37b4	172.31.16.0/20	-	-
	subnet-0038fa9df930e78d3	Available	vpc-014e267882d4c37b4	172.31.48.0/20	-	-

subnet-050b74d17f913947d / Application Subnet 1

Details

Subnet ID	subnet-050b74d17f913947d	Subnet ARN	arn:aws:ec2:us-east-1:637423550105:subnet/subnet-050b74d17f913947d	State	Available	IPv4 CIDR	10.0.3.0/24
Available IPv4 addresses	251					IPv6 CIDR association ID	-
Availability Zone ID	use1-az2					Availability Zone	us-east-1a
Network ACL	acl-0d37e77e5d73fc140	Network border group	us-east-1	VPC	vpc-03922f43caebe447e Demo VPC	Route table	rtb-067aba57ab7984573
Auto-assign customer-owned IPv4 address	No	Default subnet	No	Auto-assign public IPv4 address	Yes	Auto-assign IPv6 address	No
				Outpost ID	-	IPv4 CIDR reservations	-
				Customer-owned IPv4 pool	-		

This is the Application subnet 2.

The screenshot shows the AWS VPC Subnets page with the following details:

Name	Subnet ID	VPC	IPv4 CIDR	IPv6 CIDR	IPv6 CIDR association ID
Application Subnet 2	subnet-0c75f48d9411b7dbc	vpc-03922f43caeab447e Demo VPC	10.0.4.0/24	-	-

subnet-0c75f48d9411b7dbc / Application Subnet 2

Details

Subnet ID	subnet-0c75f48d9411b7dbc	Subnet ARN	arn:aws:ec2:us-east-1:637423550105:subnet/subnet-0c75f48d9411b7dbc	State	Available	IPv4 CIDR	10.0.4.0/24
Available IPv4 addresses	251	IPv6 CIDR	-	IPv6 CIDR association ID	-	Availability Zone	us-east-1b
Availability Zone ID	use1-az4	Network border group	us-east-1	VPC	vpc-03922f43caeab447e Demo VPC	Route table	rtb-067aba57ab7984573
Network ACL	acl-0d37e77e5d73fc140	Default subnet	No	Auto-assign public IPv4 address	No	Auto-assign IPv6 address	No
Auto-assign customer-owned IPv4 address	No	Customer-owned IPv4 pool	-	Outpost ID	-	IPv4 CIDR reservations	-

In the below image we can able to see Database subnet 1.

The screenshot shows the AWS VPC Subnets page with the following details:

Name	Subnet ID	VPC	IPv4 CIDR	IPv6 CIDR	IPv6 CIDR association ID
Database Subnet 1	subnet-0a03df81f43ddf4be	vpc-03922f43caeab447e Demo VPC	10.0.5.0/24	-	-

subnet-0a03df81f43ddf4be / Database Subnet 1

Details

Subnet ID	subnet-0a03df81f43ddf4be	Subnet ARN	arn:aws:ec2:us-east-1:637423550105:subnet/subnet-0a03df81f43ddf4be	State	Available	IPv4 CIDR	10.0.5.0/24
Available IPv4 addresses	251	IPv6 CIDR	-	IPv6 CIDR association ID	-	Availability Zone	us-east-1a
Availability Zone ID	use1-az2	Network border group	us-east-1	VPC	vpc-03922f43caeab447e Demo VPC	Route table	rtb-067aba57ab7984573
Network ACL	acl-0d37e77e5d73fc140	Default subnet	No	Auto-assign public IPv4 address	No	Auto-assign IPv6 address	No
Auto-assign customer-owned IPv4 address	No	Customer-owned IPv4 pool	-	Outpost ID	-	IPv4 CIDR reservations	-

This is the Database Subnet 2.

The screenshot shows the AWS VPC Subnets console. On the left, there's a sidebar with navigation links for EC2 Global View, Filter by VPC, Virtual private cloud, Subnets, Security, DNS Firewall, Network Firewall, and CloudShell/Feedback. The main area displays a table of subnets with columns: Name, Subnet ID, State, VPC, IPv4 CIDR, IPv6 CIDR, and IPv6 CIDR association ID. A search bar at the top says "Find resources by attribute or tag". Below the table, a specific subnet is selected: "subnet-0a9b7ff33ce7b0b2d / Database subnet 2". The "Details" tab is active, showing the following configuration:

Subnet ID	Subnet ARN	State	IPv4 CIDR
subnet-0a9b7ff33ce7b0b2d	arn:aws:ec2:us-east-1:637423550105:subnet:subnet-0a9b7ff33ce7b0b2d	Available	10.0.6.0/24
Available IPv4 addresses	251	IPv6 CIDR association ID	-
Availability Zone ID	use1-az4	VPC	vpc-03922f43caebe447e Demo VPC
Network ACL	ac-0d57e77e5d73fc140	Default subnet	No
Auto-assign customer-owned IPv4 address	No	Customer-owned IPv4 pool	-
		Outpost ID	-

STEP 4: CREATE INTERNET GATE WAY

To create an Internet Gateway in Terraform, you need to define it in a file, such as `igw.tf`. This file contains the code to create the Internet Gateway and attach it to your VPC. The Internet Gateway acts as a connection point, allowing instances in your VPC to communicate with the internet.

1. `aws_internet_gateway`:

This is the Terraform resource used to create the Internet Gateway.

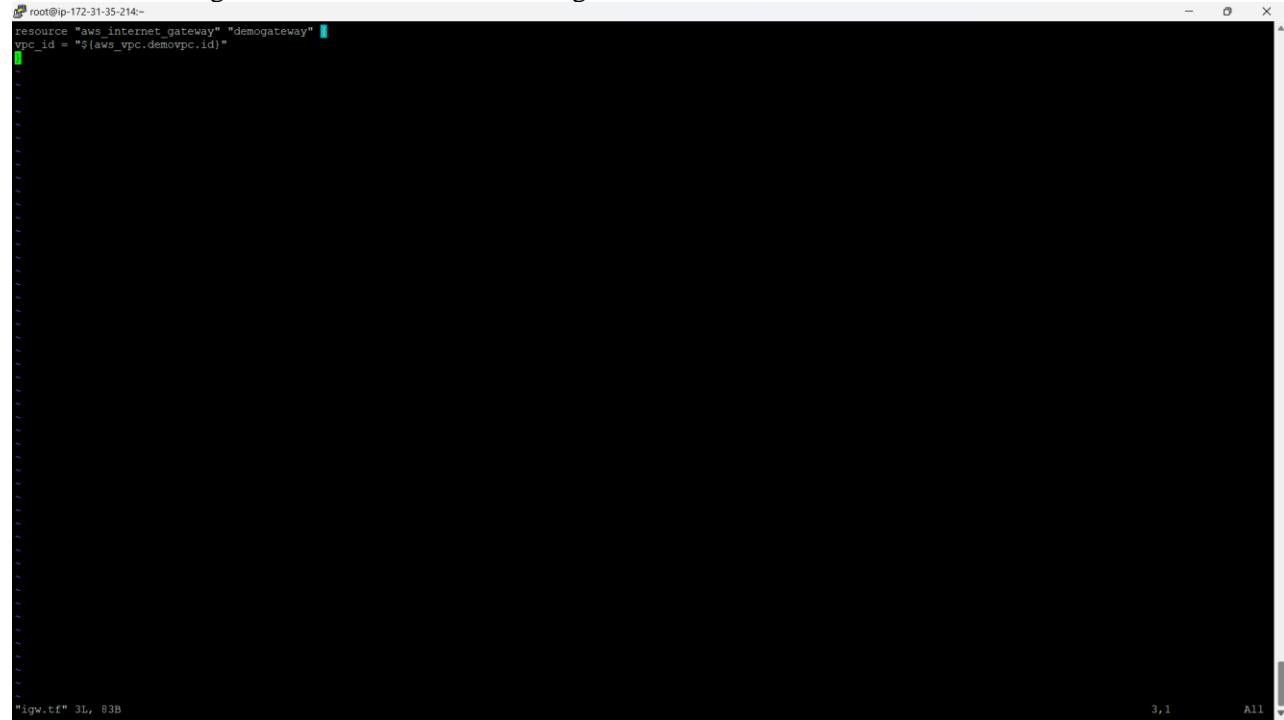
2. `vpc_id`:

This connects the Internet Gateway to the specific VPC you've already created (referenced as `aws_vpc.my_vpc.id`).

3. `Tags`:

Adding a Name tag makes it easier to find and manage the Internet Gateway in the AWS Console.

In the below image we can able to see that our igw.tf file.



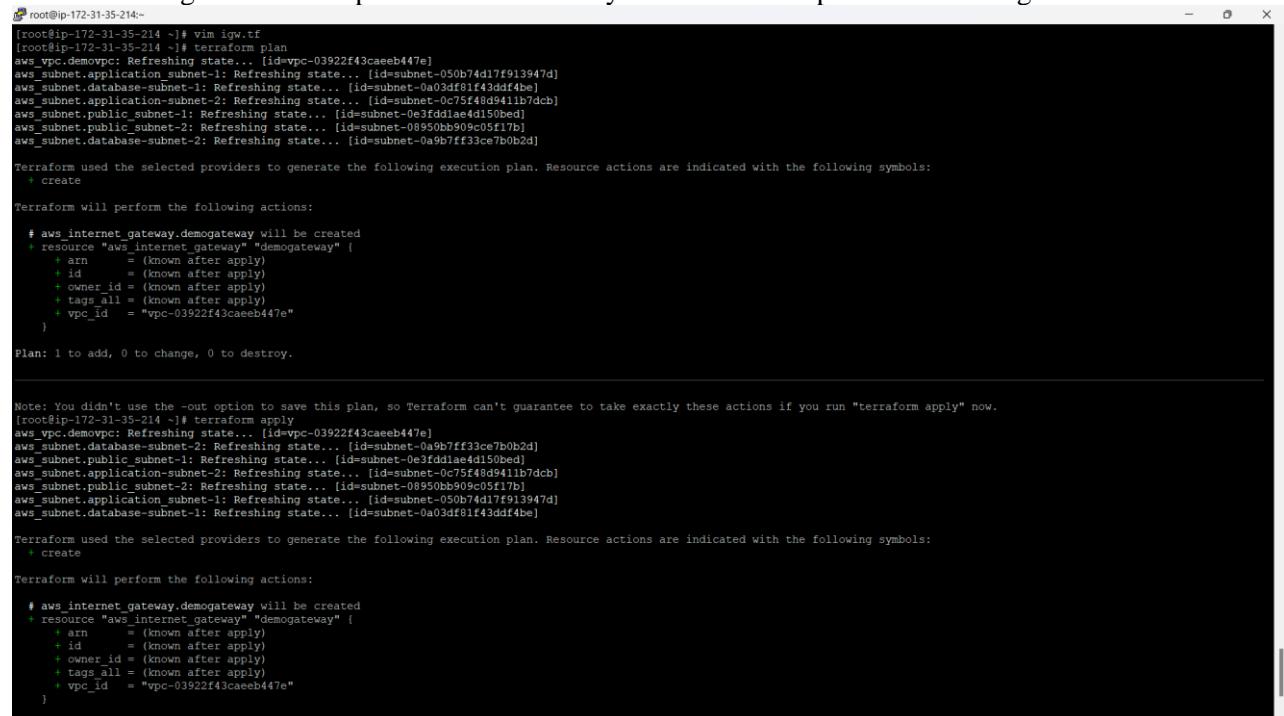
```
[root@ip-172-31-35-214 ~]# vim igw.tf
resource "aws_internet_gateway" "demogateway"
vpc_id = "${aws_vpc.demovpc.id}"

```

"igw.tf" 3L, 83B

3,1 All

We are running an terraform plan command for any modification or preview the changes.



```
[root@ip-172-31-35-214 ~]# vim igw.tf
[root@ip-172-31-35-214 ~]# terraform plan
aws_vpc.demovpc: Refreshing state... [id=vpc-03922f43caeef447e]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-050b74d17ff913947d]
aws_subnet.database_subnet-1: Refreshing state... [id=subnet-0a03df81f43dd4f4be]
aws_subnet.application_subnet-2: Refreshing state... [id=subnet-0c75f48d9411b7dc]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-0e3fd1ae4d150bed]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-08950bb909c05f17b]
aws_subnet.database_subnet-2: Refreshing state... [id=subnet-0a9b7ff33ce7b0b2d]

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-03922f43caeef447e"
}

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-172-31-35-214 ~]# terraform plan
aws_vpc.demovpc: Refreshing state... [id=vpc-03922f43caeef447e]
aws_subnet.database_subnet-2: Refreshing state... [id=subnet-0a9b7ff33ce7b0b2d]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-0e3fd1ae4d150bed]
aws_subnet.application_subnet-2: Refreshing state... [id=subnet-0c75f48d9411b7dc]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-08950bb909c05f17b]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-050b74d17ff913947d]
aws_subnet.database_subnet-1: Refreshing state... [id=subnet-0a03df81f43dd4f4be]

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-03922f43caeef447e"
}
```

In the below image we can able to see that **terraform apply** for the applying the changes to create a internet gateway.

```
root@ip-172-31-35-214:~#
# 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-03922f43caeab447e"
}

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-172-31-35-214 ~]# terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-03922f43caeab447e]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0a9b7ff33ce7b0b2d]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0e3fd1ae4d150bed]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-0c75f48d941b7dcb]
aws_subnet.private-subnet-2: Refreshing state... [id=subnet-08950bb909c05f17b]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-050b74d17f913947d]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0a03df81f43dd14be]

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-03922f43caeab447e"
}

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 value: yes

aws_internet_gateway.demogateway: Creating...
aws_internet_gateway.demogateway: Creation complete after 0s [id=igw-004abd1f8ac6397f8]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-35-214 ~]# vim igw.tf
[root@ip-172-31-35-214 ~]#
```

In the below image we can able to see that our Internet gateway has been created successfully and attached to VPC.

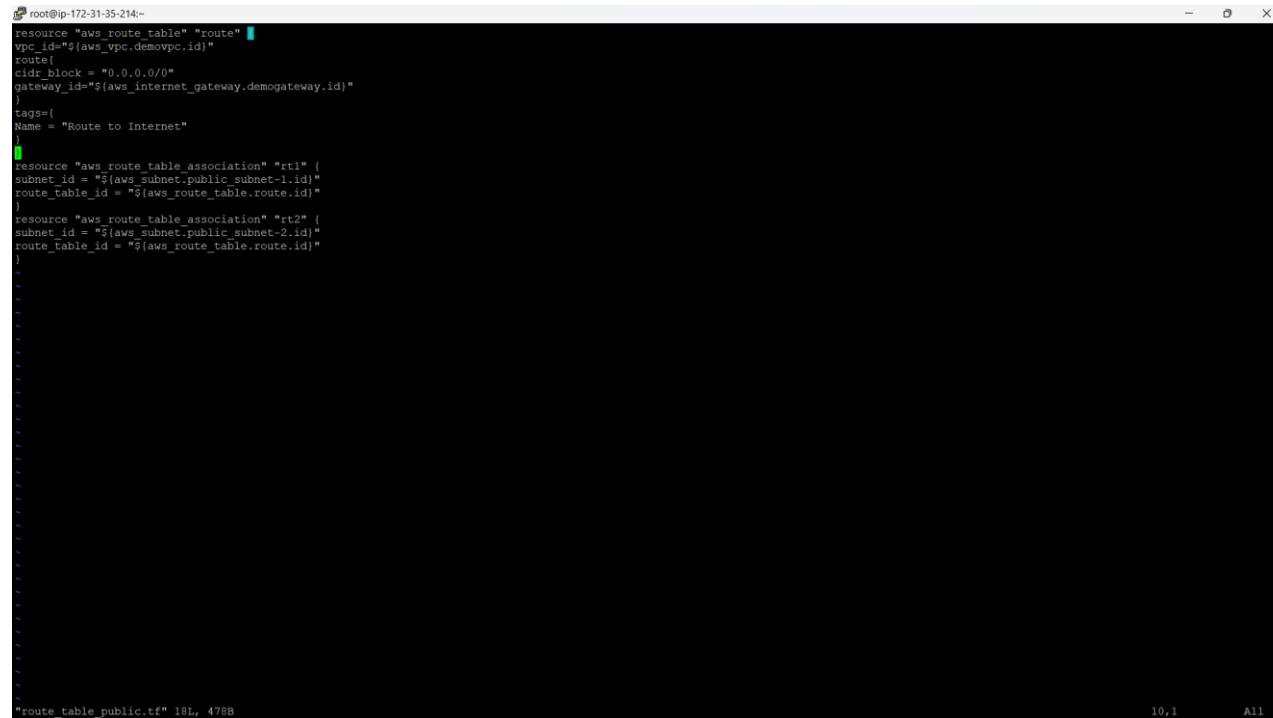
Name	Internet gateway ID	State	VPC ID	Owner
-	igw-004abd1f8ac6397f8	Attached	vpc-03922f43caeab447e Demo_VPC	637423550105
-	igw-0917bbbd60434e117	Attached	vpc-014e267882d4c37b4	637423550105

igw-004abd1f8ac6397f8

Details	Tags
Internet gateway ID igw-004abd1f8ac6397f8	State Attached
VPC ID vpc-03922f43caeab447e Demo_VPC	Owner 637423550105

STEP 5: CREATING THE ROUTING TABLE

A Route Table is like a map that guides the flow of network traffic within your VPC (Virtual Private Cloud) in AWS. It decides where data should go when it leaves a resource, such as an EC2 instance. For example, it can direct traffic to the internet, other subnets, or a VPN connection. I have created an **route_table_public.tf** file to create a route table.



```
root@ip-172-31-35-214:~# 
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" "rtt1" {
  subnet_id = "${aws_subnet.public_subnet-1.id}"
  route_table_id = "${aws_route_table.route.id}"
}

resource "aws_route_table_association" "rtt2" {
  subnet_id = "${aws_subnet.public_subnet-2.id}"
  route_table_id = "${aws_route_table.route.id}"
}

route_table_public.tf: 18L, 478B
10,1 All
```

0.0.0.0/0 → Internet Gateway (for internet access).

Subnet CIDRs → Local traffic within the VPC.

Private IP ranges → NAT Gateway for outbound internet access without exposing resources.

Control Traffic Flow: Ensures data reaches the correct destination.

Enables Internet Access: Connects subnets to the internet or other networks.

Network Segmentation: Helps isolate and secure parts of your network.

In simple terms, the route table is the brain behind how network traffic moves in and out of your VPC!

In the below image I have run the terraform plan command because I have review the changes and if any error occurs I can able to find it before applying.

```
root@ip-172-31-35-214:~# vim route_table_public.tf
[root@ip-172-31-35-214 ~]# terraform plan
aws_vpc.demoVPC: Refreshing state... [id=vpc-03922f43caeef447e]
aws_subnet.database_subnet-2: Refreshing state... [id=subnet-0a9b7ff33ce7b0b2d]
aws_internet_gateway.demoGateway: Refreshing state... [id=igw-004abd1f8ac6397f8]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-050b74d17f913947d]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-08950bb909c05f17b]
aws_subnet.database_subnet-1: Refreshing state... [id=subnet-0a03df81f43dd44be]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-0e3fd1ae4d150bed]
aws_subnet.application_subnet-2: Refreshing state... [id=subnet-0c75f48d9411b7dcb]

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-004abd1f8ac6397f8"
      # (11 unchanged attributes hidden)
    },
  + tags         = {
      + "Name" = "Route to Internet"
    }
  + tags_all     = {
      + "Name" = "Route to Internet"
    }
  + vpc_id       = "vpc-03922f43caeef447e"
}

# 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-0e3fd1ae4d150bed"
}

# aws_route_table_association.rt2 will be created
resource "aws_route_table_association" "rt2" {
  + id          = (known after apply)
  + route_table_id = (known after apply)
  + subnet_id    = "subnet-08950bb909c05f17b"
}
```

After completion of terraform plan there is no error or any mistake so it is ready for applying so I used terraform apply command to make changes and to create a route table.

```
root@ip-172-31-35-214:~# terraform apply
# aws_route_table_association.rt2 will be created
resource "aws_route_table_association" "rt2" {
  + id          = (known after apply)
  + route_table_id = (known after apply)
  + subnet_id    = "subnet-08950bb909c05f17b"
}

Plan: 3 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-172-31-35-214 ~]# terraform apply
aws_vpc.demoVPC: Refreshing state... [id=vpc-03922f43caeef447e]
aws_subnet.application_subnet-2: Refreshing state... [id=subnet-0c75f48d9411b7dcb]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-08950bb909c05f17b]
aws_subnet.database_subnet-2: Refreshing state... [id=subnet-0a9b7ff33ce7b0b2d]
aws_internet_gateway.demoGateway: Refreshing state... [id=igw-004abd1f8ac6397f8]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-050b74d17f913947d]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-0e3fd1ae4d150bed]
aws_subnet.database_subnet-1: Refreshing state... [id=subnet-0a03df81f43dd44be]

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-004abd1f8ac6397f8"
      # (11 unchanged attributes hidden)
    },
  + tags         = {
      + "Name" = "Route to Internet"
    }
  + tags_all     = {
      + "Name" = "Route to Internet"
    }
  + vpc_id       = "vpc-03922f43caeef447e"
}

# aws_route_table_association.rtl will be created
resource "aws_route_table_association" "rtl" {
  + id          = (known after apply)
```

In the below image we can able to see that our apply is successful then we need to go to AWS VPC and then click on route table then we can able to see that our route table as been created successfully.

```
root@ip-172-31-35-214:~#
+ 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-004abd1f8ac6397f8"
|   # (11 unchanged attributes hidden)
|
+ tags         =
+ "Name"      = "Route to Internet"
+ tags_all    =
+ "Name"      = "Route to Internet"
+ vpc_id      = "vpc-03922f43caebe447e"
}

# 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-0e3fd1aae4d150bed"
}

# aws_route_table_association.rt2 will be created
resource "aws_route_table_association" "rt2" {
+ id           = (known after apply)
+ route_table_id = (known after apply)
+ subnet_id    = "subnet-08950bb909c05f17b"
}

Plan: 3 to add, 0 to change, 0 to destroy.

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

Enter a value: yes

aws_route_table.route: Creating...
aws_route_table.route: Creation complete after 1s [id=rtb-02f7f23dd541a0e83]
aws_route_table_association.rtl: Creating...
aws_route_table_association.rt2: Creating...
aws_route_table_association.rt2: Creation complete after 0s [id=rtbassoc-00a769dbaba8bf54f]
aws_route_table_association.rtl: Creation complete after 0s [id=rtbassoc-04f3c1b154f815c79]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
[root@ip-172-31-35-214 ~]# vim route_table_public.tf
[root@ip-172-31-35-214 ~]#
```

In the below image we can able to see that our route table as been created successfully.

The screenshot shows the AWS VPC dashboard with the 'Route tables' section selected. A table lists one route table:

Name	Route table ID	Explicit subnet associations	Main	VPC	Owner ID
Route to Internet	rtb-02f7f23dd541a0e83	2 subnets	No	vpc-03922f43caebe447e Demo VPC	637423550105

Below the table, a detailed view of the route table 'rtb-02f7f23dd541a0e83' is shown. The 'Details' tab is selected, displaying the following information:

- Route table ID: rtb-02f7f23dd541a0e83
- Main: No
- Explicit subnet associations: 2 subnets
- VPC: vpc-03922f43caebe447e | Demo VPC
- Owner ID: 637423550105

In the below image we can able to see that the route association.

The screenshot shows the AWS VPC Route Tables console. On the left, there's a sidebar with various VPC-related options like EC2 Global View, Virtual private cloud, Route tables, Security, DNS Firewall, Network Firewall, and CloudShell. The main area displays a table of route tables:

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC	Owner ID
-	rtb-067aba57ab7984573	-	-	Yes	vpc-03922f43caeab447e Demo VPC	637423550105
<input checked="" type="checkbox"/> Route to Internet	rtb-02f7f23dd541a0e83	2 subnets	-	No	vpc-03922f43caeab447e Demo VPC	637423550105
-	rtb-02a08c64822c08da8	-	-	Yes	vpc-014e267882d4c37b4	637423550105

When you click on the 'Route to Internet' row, it takes you to a detailed view of the route table:

rtb-02f7f23dd541a0e83 / Route to Internet

- Details
- Routes
- Subnet associations** (selected)
- Edge associations
- Route propagation
- Tags

Explicit subnet associations (2)

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
Web subnet 2	subnet-08950bb90c05f17b	10.0.2.0/24	-
Web Subnet 1	subnet-0e5fd1ae4d150bed	10.0.1.0/24	-

Subnets without explicit associations (4)

The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
Database subnet 2	subnet-0a9b7f3ce7b0b2d	10.0.5.0/24	-
Application Subnet 1	subnet-050b74d17f913947d	10.0.3.0/24	-
Database Subnet 1	subnet-0a3df1f43df4be	10.0.5.0/24	-
Application Subnet 2	subnet-0c75f48d9411b7dc	10.0.4.0/24	-

After creating VPC, Subnets, Internet gateway, Route table then go to VPC and click on VPC in that we can able to see Route map click on that and we can able to see a map for the creation of our VPC and subnet attachment to which route table and connected to which network connection.

The screenshot shows the AWS VPC dashboard. The sidebar includes EC2 Global View, Virtual private cloud, Route tables, Security, DNS Firewall, Network Firewall, and CloudShell. The main area shows a table of VPCs:

Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR	DHCP option set	Main route table
<input checked="" type="checkbox"/> Demo VPC	vpc-0894591850cce1377	Available	Off	10.0.0.0/16	-	dept-08010d2dc2ef4a5ce	rtb-048de6637
-	vpc-014e267882d4c37b4	Available	Off	172.31.0.0/16	-	dept-08010d2dc2ef4a5ce	rtb-02a08c64822c08da8

When you click on the 'Demo VPC' row, it takes you to its detailed view:

vpc-0894591850cce1377 / Demo VPC

- Details
- Resource map** (selected)
- CIDRs
- Flow logs
- Tags
- Integrations

Resource map

```

graph LR
    VPC[VPC] --- Subnets[Subnets]
    Subnets --- usEast1a[us-east-1a]
    Subnets --- usEast1b[us-east-1b]
    usEast1a --- WebSubnet1[Web Subnet 1]
    usEast1a --- DatabaseSubnet1[Database Subnet 1]
    usEast1a --- ApplicationSubnet1[Application Subnet 1]
    usEast1b --- WebSubnet2[Web subnet 2]
    usEast1b --- DatabaseSubnet2[Database subnet 2]
    usEast1b --- ApplicationSubnet2[Application Subnet 2]
    RT[Route tables] --- RouteToInternet[Route to Internet]
    RouteToInternet --- rtb[rtb-048de6637da15eda1]
    NC[Network connections] --- igw[igw-09f16a122297c5ce2]
  
```

STEP 6: CREATING A SECURITY GROUP

A Security Group in AWS acts like a virtual firewall for your resources, such as EC2 instances. It controls what kind of network traffic is allowed to or from these resources. Think of it as a set of rules that decides who can communicate with your resource and how. I have created a **web_sg.tf** file for the security group creation before the creation of ec2.

```
[root@ip-172-31-35-214-]# vim web_sg.tf
[root@ip-172-31-35-214-]# terraform plan
aws_vpc.demovpc: Refreshing state... [id=vpc-03922f43caeb447e]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-050b74d17f913947d]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-08950bb90c05f17b]
aws_internet_gateway.demogateway: Refreshing state... [id=igw-004abd1f8ac6397f8]
aws_subnet.application_subnet-2: Refreshing state... [id=subnet-0c75f48d94e057dc]
aws_subnet.database_subnet-1: Refreshing state... [id=subnet-0a03d801f43dd4be]
aws_subnet.database_subnet-2: Refreshing state... [id=subnet-0a03d801f43dd4be]
aws_subnet_public_subnet-1: Refreshing state... [id=subnet-0cf1d14d150bed]
aws_route_table.route: Refreshing state... [id=rtb-0247f23dd541a0e83]
aws_route_table_association.rtl: Refreshing state... [id=rthassoc-04f3c1b154f815c79]
aws_route_table_association.rt2: Refreshing state... [id=rthassoc-00a769bdbab0bf54f]

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
          + to_port        = 22
          # (1 unchanged attribute hidden)
        ],
      + name         = (known after apply)
      + name_prefix   = (known after apply)
      + owner_id      = (known after apply)
      + revoke_rules_on_delete = false
      + tags          =
          + "Name" = "Web SG"
      + tags_all      =
          + "Name" = "Web SG"
    )
    + vpc_id        = "vpc-03922f43caeb447e"

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-172-31-35-214-]# terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-03922f43caeb447e]
aws_internet_gateway.demogateway: Refreshing state... [id=igw-004abd1f8ac6397f8]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-050b74d17f913947d]
aws_subnet.database_subnet-1: Refreshing state... [id=subnet-0a03d801f43dd4be]
aws_subnet_public_subnet-1: Refreshing state... [id=subnet-08950bb90c05f17b]
```

I have used terraform apply command to make changes and create an security group for an ec2 instance.

```
root@ip-172-31-35-214:~# terraform apply
      + vpc_id          = "vpc-03922f43caeeb447e"
    }

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-172-31-35-214 ~]# terraform apply
aws_vpc.demosvc: Refreshing state... [id=vpc-03922f43caeeb447e]
aws_subnet.gateway: Refreshing state... [id=subnet-004abd1f8ac6397f8]
aws_subnet.application-1: Refreshing state... [id=subnet-050b74d17f913947d]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0a03df81f434df4be]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-08950bb909c05f17b]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0a9b7ff33ce7b0b2d]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-0c75f48d941b7dcb]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-0e3fd1ae4d150bed]
aws_route_table.route: Refreshing state... [id=rtb-0247f23d541a0e83]
aws_route_table_association.rt1: Refreshing state... [id=rtbassoc-04f3c1b154f815c79]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-00a7699dab8bf54f]

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",
            ]
        }
    ],
  + name          = (known after apply)
  + name_prefix   = (known after apply)
  + owner_id      = (known after apply)
  + revoke_rules_on_delete = false
  + tags          = [
      + {
          + "Name" = "Web SG"
        }
    ]
  + tags_all      = [
      + {
          + "Name" = "Web SG"
        }
    ]
  + vpc_id        = "vpc-03922f43caeeb447e"
}
```

In the below image we can able to see that the terraform apply as been successfully applied. So now we can able to go and check in the security group we can able to see our new security group called **Web SG**.

```
root@ip-172-31-35-214:~# terraform apply
      + cidr_blocks  = [
          + "0.0.0.0/0",
        ]
      + from_port     = 443
      + ipv6_cidr_blocks = []
      + prefix_list_ids = []
      + protocol      = "tcp"
      + security_groups = []
      + self          = false
      + to_port       = 443
      # (1 unchanged attribute hidden)
    },
    + {
      + cidr_blocks  = [
          + "0.0.0.0/0",
        ]
      + from_port     = 80
      + ipv6_cidr_blocks = []
      + prefix_list_ids = []
      + protocol      = "tcp"
      + security_groups = []
      + self          = false
      + to_port       = 80
      # (1 unchanged attribute hidden)
    },
  + name          = (known after apply)
  + name_prefix   = (known after apply)
  + owner_id      = (known after apply)
  + revoke_rules_on_delete = false
  + tags          = [
      + {
          + "Name" = "Web SG"
        }
    ]
  + tags_all      = [
      + {
          + "Name" = "Web SG"
        }
    ]
  + vpc_id        = "vpc-03922f43caeeb447e"
}

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.demosg: Creating...
aws_security_group.demosg: Creation complete after 2s [id=sg-0f483fd95c06b7406]

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

In the below image we can able to see that our new security group has been created successfully.

The screenshot shows the AWS EC2 Security Groups page. A new security group named "Web SG" has been created, highlighted with a blue selection box. The details for "Web SG" are shown in the main content area:

Security group name	Security group ID	Description	VPC ID
terraform-2024111905561163860000001	sg-0f483fd95c06b7486	Managed by Terraform	vpc-03922f43caeef447e
Owner	637423550105	Inbound rules count	3 Permission entries
		Outbound rules count	1 Permission entry

Below are the inbound rules for the security group that we have created.

The screenshot shows the AWS EC2 Security Groups page with the "Inbound rules" tab selected for the "Web SG" security group. Three inbound rules are listed:

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
sgr-0408c2ec99969e4fb	IPv4	HTTP	TCP	80	0.0.0.0/0	-	
sgr-04f2456d3159cdb2	IPv4	HTTPS	TCP	443	0.0.0.0/0	-	
sgr-0956e04f87fe3ef65	IPv4	SSH	TCP	22	0.0.0.0/0	-	

STEP 7: CREATING AN USER DATA

We are creating an data.sh because if we want to create an instance. The data.sh file is a shell script that you can use to automate the setup of your EC2 instance when it starts. By including this file in the User Data section, you ensure that the necessary software and configurations are automatically applied without having to manually log into the instance after it's launched.

```
#!/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/Spandanall5/Anticafe.git
sudo mv Anticafe/* /var/www/html/
```

```
#!/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/Spandanall5/kider.git
sudo mv kider/preschool-website-template/* /var/www/html/
```

I have used terraform plan and apply for the data.sh.

```
root@ip-172-31-35-214:~#
}
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.demosg: Creating...
aws_security_group.demosg: Creation complete after 2s [id=sg-0f483fd95c06b7406]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-35-214 ~]# vim data.sh
[root@ip-172-31-35-214 ~]# terraform plan
aws_vpc.demovpc: Refreshing state... [id=vpc-03922f43caeef447e]
aws_internet_gateway.demogateway: Refreshing state... [id=intgw-004abd1f8ac6397f8]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0a03df81f43dd4f4be]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-0c75f48d411b7dc]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-08950bb909c05f17b]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0a9b7ff33ce7b0bd]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-0e1fd1iae4d150bed]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-09850bb909c05f17d]
aws_security_group.demosec: Refreshing state... [id=sg-043d65c06b7406]
aws_route_table.outs: Refreshing state... [id=rtb-02f7f23d51a0e083]
aws_route_table_association.rt2: Refreshing state... [id=rthassoc-00a769bdbaba8bf54f]
aws_route_table_association.rt1: Refreshing state... [id=rthassoc-04fc1b154f815c79]

No changes. Your infrastructure matches the configuration.

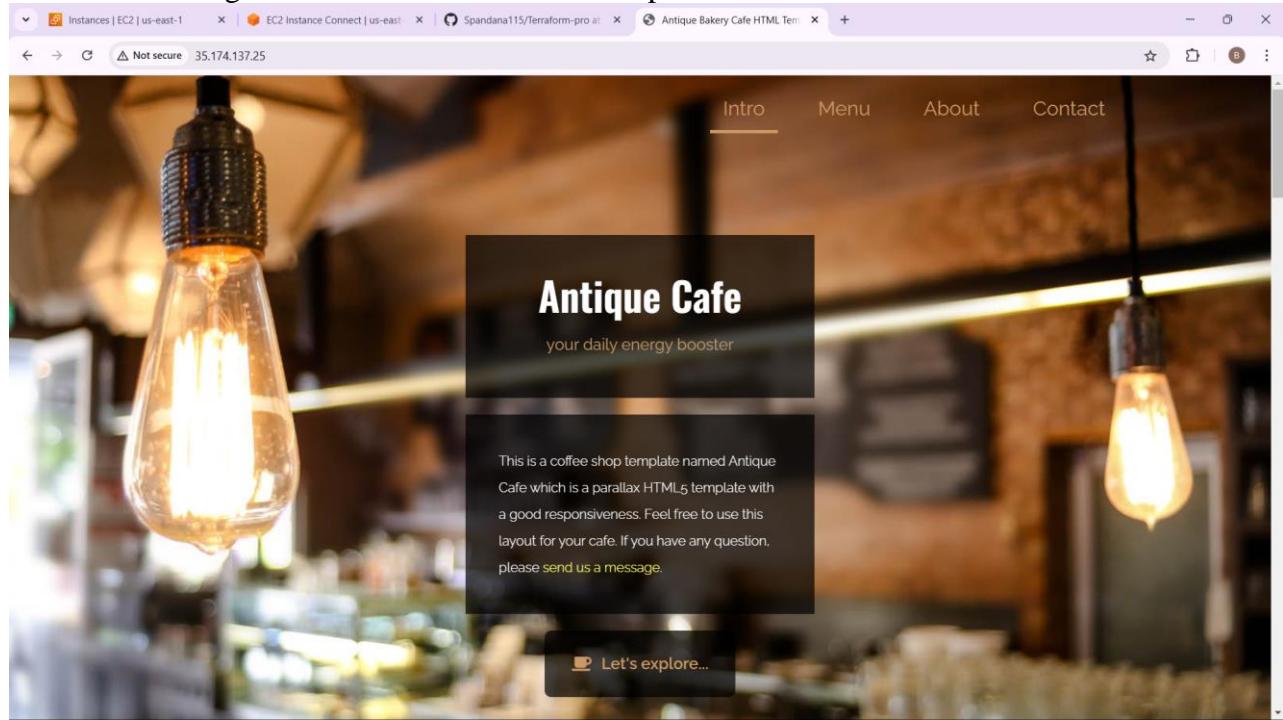
Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
[root@ip-172-31-35-214 ~]# vim data.sh
aws_vpc.demovpc: Refreshing state... [id=vpc-03922f43caeef447e]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-0e1fd1iae4d150bed]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-050b74d1f7913947d]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-08950bb909c05f17b]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0a9b7ff33ce7b0bd]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0a03df81f43dd4f4be]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-0c75f48d411b7dc]
aws_security_group.demosec: Refreshing state... [id=sg-043d65c06b7406]
aws_route_table.outs: Refreshing state... [id=rtb-004abd1f8ac6397f8]
aws_route_table.outs: Refreshing state... [id=rthassoc-00a769bdbaba8bf54f]
aws_route_table_association.rt1: Refreshing state... [id=rthassoc-04fc1b154f815c79]
aws_route_table_association.rt2: Refreshing state... [id=rthassoc-00a769bdbaba8bf54f]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
[root@ip-172-31-35-214 ~]# vim d
```

In the below image we can able to see that the output for the user data.



STEP 8: CREATING AN EC2 INSTANCES

To create an EC2 instance in AWS using Terraform, we need to define the configuration in a file **ec2.tf** file. This file contains all the details about your EC2 instance, such as the Amazon Machine Image (AMI), instance type, security groups, and other settings. I have created 2 instances.

```
root@ip-172-31-44-73:~#
resource"aws_instance" "demoinstance" {
ami = "ami-0166fe664262f664c"
instance_type = "t2.micro"
counts = 2
key_name = "spandana"
vpc_security_group_ids = ["${aws_security_group.demosg.id}"]
subnet_id = "${aws_subnet.public_subnet-1.id}"
associate_public_ip_address = true
user_data = "${file("data.sh")}"
tags={
Name = "My Public Instance"
}
}
resource"aws_instance" "demoinstance1" {
ami = "ami-0166fe664262f664c"
instance_type = "t2.micro"
counts = 2
key_name = "spandana"
vpc_security_group_ids = ["${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"
}
}

"ec2.tf" 26L, 609B
15,28 All
```

After creating an ec2.tf file then i have run a terraform plan command and terraform apply command to check and create an ec2 instances based on our configuration.

```
[root@ip-172-31-44-73 ~]# vim ec2.tf
[roo[root@ip-172-31-44-73 ~]# terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-0de869c7768593a13]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-0a1b5cefd9c1167d3]
aws_subnet.database_subnet-1: Refreshing state... [id=subnet-0dc3a1a8e297b0a04]
aws_subnet.database_subnet-2: Refreshing state... [id=subnet-011e651374282cac0]
aws_internet_gateway.demogateway: Refreshing state... [id=igw-0c3d4ea9e4afcdcf5]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-0cdccdaeb4258556]
aws_security_group.demosg: Refreshing state... [id=sg-093066f501b2b1696]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-0b79ddaa2fb330914]
aws_route_table.route: Refreshing state... [id=rtb-083edbf85433216f]
aws_instance.demoinstance[0]: Refreshing state... [id=i-075dd0ee9db2f7c47]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-090a642ca2057429f]
aws_route_table_association.rtl: Refreshing state... [id=rtbassoc-06d3c8e3752522298]

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

Terraform will perform the following actions:

  # aws_instance.demoinstance[0] will be created
+ resource "aws_instance" "demoinstance" {
    + ami                                = "ami-0166fe664262f664c"
    + 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)
    + do_optimize                        = (known after apply)
    + get_password_data                 = false
    + host_id                            = (Known after apply)
    + host_resource_groupArn            = (Known after apply)
    + iam_instance_profile              = (Known after apply)
    + id                                 = (Known after apply)
    + instance_initiated_shutdown_behavior = (Known after apply)
    + instance_lifecycle                = (Known after apply)
    + instance_state                    = (Known after apply)
    + instance_type                     = "t2.micro"
    + ipv6_address_count                = (Known after apply)
    + ipv6_addresses                     = (Known after apply)
    + key_name                           = "spandana"
    + 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)
    + private_dns                         = (Known after apply)
    + private_ip                          = (Known after apply)
}
```

```
[root@ip-172-31-44-73 ~]# vim data.sh
[roo[root@ip-172-31-44-73 ~]# terraform plan
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_instance.demoinstance[0]: Creating...
aws_instance.demoinstance[0]: Still creating... [10s elapsed]
aws_instance.demoinstance[0]: Creation complete after 12s [id=i-01553d92ac4b9dfc4]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[roo[root@ip-172-31-44-73 ~]# vim data.sh
[roo[root@ip-172-31-44-73 ~]# terraform plan
aws_vpc.demovpc: Refreshing state... [id=vpc-0de869c7768593a13]
aws_subnet.database_subnet-1: Refreshing state... [id=subnet-0dc3a1a8e297b0a04]
aws_subnet.database_subnet-2: Refreshing state... [id=subnet-011e651374282cac0]
aws_subnet.application_subnet-2: Refreshing state... [id=subnet-0c3d4ea9e4afcdcf5]
aws_internet_gateway.demogateway: Refreshing state... [id=igw-0c3d4ea9e4afcdcf5]
aws_subnet.public_subnet-1: Refreshing state... [id=sg-093066f501b2b1696]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-0a1b5cefd9c1167d3]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-0b79ddaa2fb330914]
aws_route_table.route: Refreshing state... [id=rtb-083edbf85433216f]
aws_instance.demoinstance[0]: Refreshing state... [id=i-01553d92ac4b9dfc4]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-090a642ca2057429f]
aws_route_table_association.rtl: Refreshing state... [id=rtbassoc-06d3c8e3752522298]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.
[roo[root@ip-172-31-44-73 ~]# terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-0de869c7768593a13]
aws_subnet.application_subnet-2: Refreshing state... [id=subnet-0c3d4ea9e4afcdcf5]
aws_security_group.demosg: Refreshing state... [id=sg-093066f501b2b1696]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-0a1b5cefd9c1167d3]
aws_subnet.database_subnet-1: Refreshing state... [id=subnet-0dc3a1a8e297b0a04]
aws_subnet.database_subnet-2: Refreshing state... [id=subnet-011e651374282cac0]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-0b79ddaa2fb330914]
aws_internet_gateway.demogateway: Refreshing state... [id=igw-0c3d4ea9e4afcdcf5]
aws_route_table.route: Refreshing state... [id=rtb-083edbf85433216f]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-090a642ca2057429f]
aws_instance.demoinstance[0]: Refreshing state... [id=i-01553d92ac4b9dfc4]
aws_instance.demoinstance[0]: Refreshing state... [id=i-075dd0ee9db2f7c47]
aws_route_table_association.rtl: Refreshing state... [id=rtbassoc-06d3c8e3752522298]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
[roo[root@ip-172-31-44-73 ~]#
```

In the below image we can able to see that our newly created 2 instances has been created successfully.the below image is the first instance details.

Instances (1/3) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
My Public Inst...	i-0758d0ee9db2f7c47	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	-	54.173.104.126	-
My Public Inst...	i-01553d92ac4b9dfc4	Running	t2.micro	Initializing	View alarms +	us-east-1a	-	54.208.67.157	-
Puppy	i-093f6e44b64c874be	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1c	ec2-100-29-9-34.comp...	100.29.9.34	-

i-01553d92ac4b9dfc4 (My Public Instance)

Details Status and alarms Monitoring Security Networking Storage Tags

Instance summary Info

Instance ID	i-01553d92ac4b9dfc4	Public IPv4 address	54.208.67.157 [open address]	Private IPv4 addresses	10.0.1.169
IPv6 address	-	Instance state	Running	Public IPv4 DNS	-
Hostname type	IP name: ip-10-0-1-169.ec2.internal	Private IP DNS name (IPv4 only)	ip-10-0-1-169.ec2.internal	Elastic IP addresses	-
Answer private resource DNS name	-	Instance type	t2.micro	AWS Compute Optimizer finding	-
Auto-assigned IP address	-	VPC ID	vpc-0de869c7768593a13 (Demo VPC)	Auto Scaling Group name	-
IAM Role	-	Subnet ID	subnet-0cdcdabc425855e6 (Web Subnet 1)		

In the below image we can see that second instance details.

Instances (1/3) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
My Public Inst...	i-0758d0ee9db2f7c47	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	-	54.173.104.126	-
My Public Inst...	i-01553d92ac4b9dfc4	Running	t2.micro	Initializing	View alarms +	us-east-1a	-	54.208.67.157	-
Puppy	i-093f6e44b64c874be	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1c	ec2-100-29-9-34.comp...	100.29.9.34	-

i-0758d0ee9db2f7c47 (My Public Instance)

Details Status and alarms Monitoring Security Networking Storage Tags

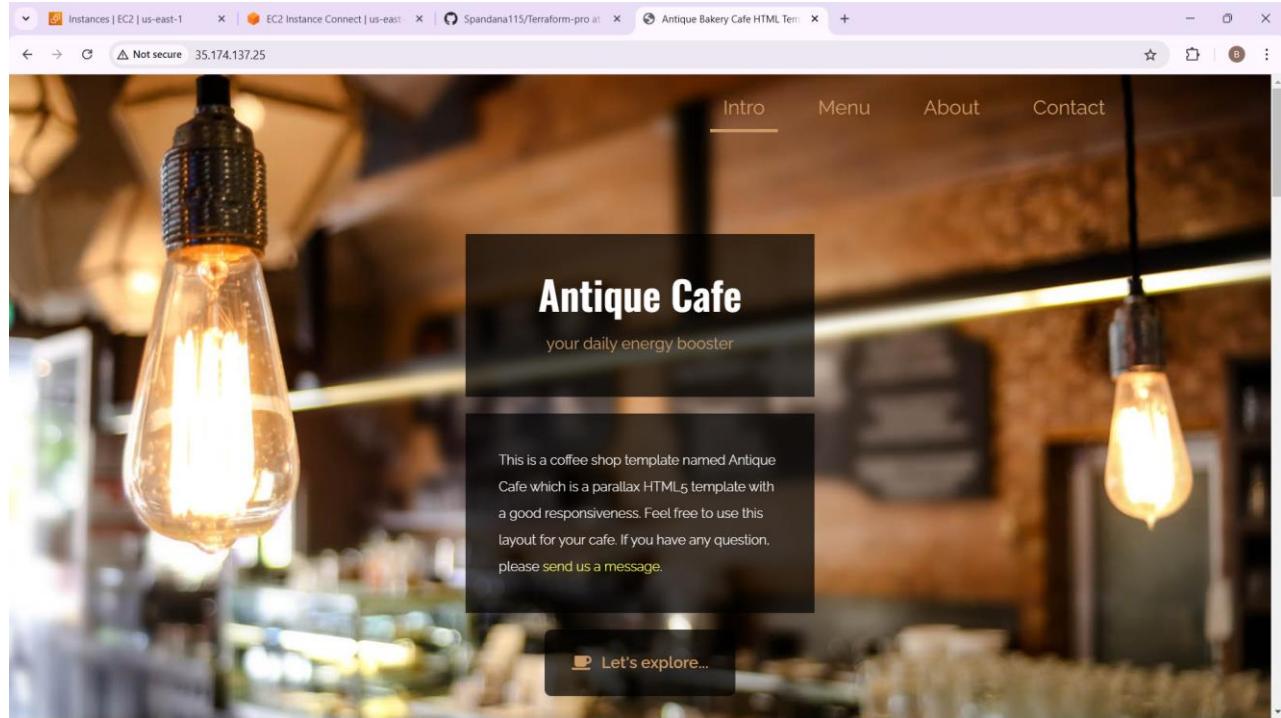
Instance summary Info

Instance ID	i-0758d0ee9db2f7c47	Public IPv4 address	54.173.104.126 [open address]	Private IPv4 addresses	10.0.1.27				
IPv6 address	-	Instance state	Running	Public IPv4 DNS	-				
Hostname type	IP name: ip-10-0-1-27.ec2.internal	Private IP DNS name (IPv4 only)	ip-10-0-1-27.ec2.internal	Elastic IP addresses	-				
Answer private resource DNS name	-	Instance type	t2.micro	AWS Compute Optimizer finding	-				
Auto-assigned IP address	54.173.104.126 [Public IP]	VPC ID	vpc-0de869c7768593a13 (Demo VPC)	Opt-in to AWS Compute Optimizer for recommendations. [Learn more]					
IAM Role	-	Subnet ID	subnet-0cdcdabc425855e6 (Web Subnet 1)	Auto Scaling Group name	-				
IMDSv2	-	Instance ARN	arn:aws:ec2:us-east-1:637423550105:instance/i-0758d0ee9db2f7c47						
Optional	-								
EC2 recommends setting IMDSv2 to required Learn more									

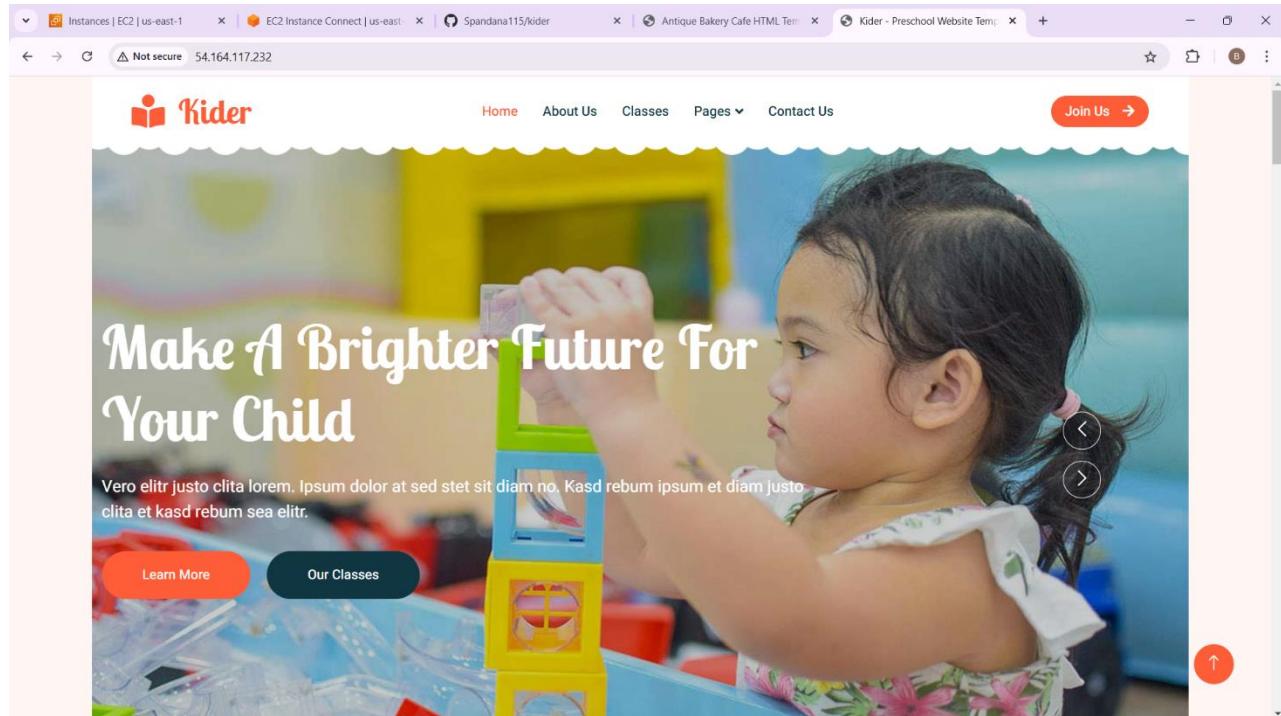
Instance details Info

Platform	AMI ID	Monitoring
----------	--------	------------

After creating an instances if we want to check that user data is working or not we need to copy the public IP of the instance and paster it in browser it throughs an error because it is https:// we need to remove the “s” http:// and again reload we can able to see the user data content.

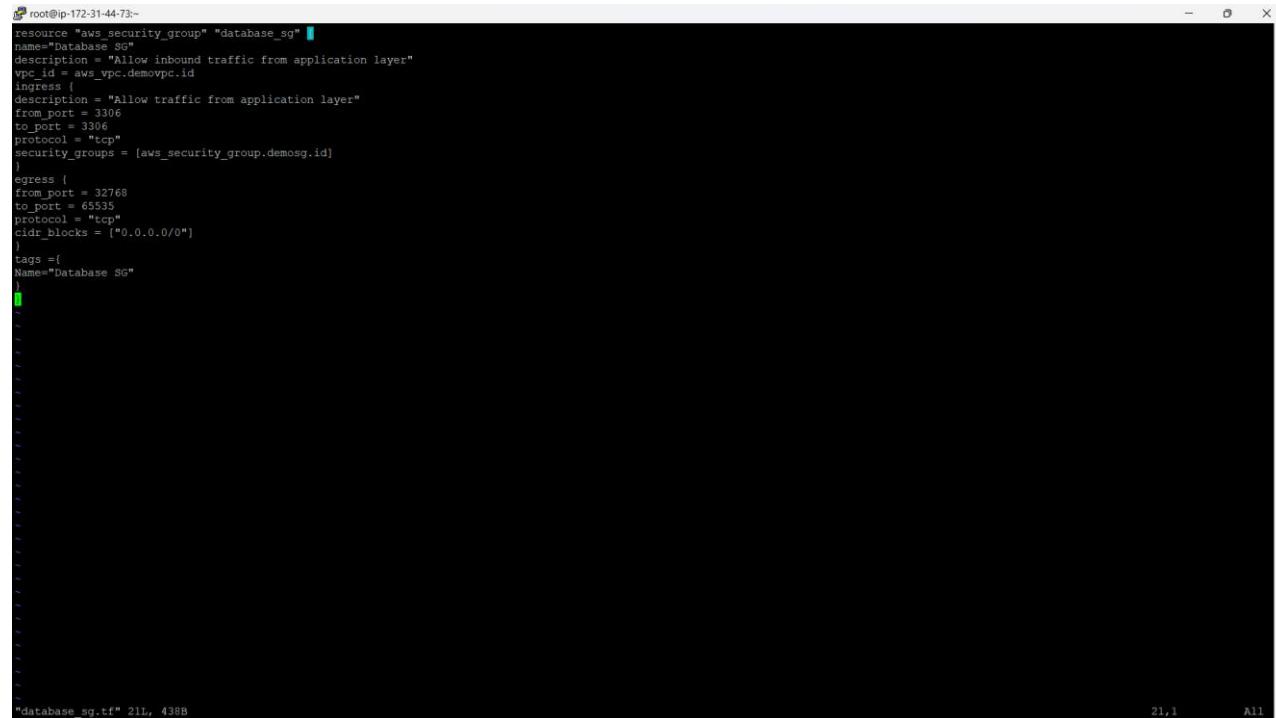


We need to do the same process for the second instance then we can able to see like a below image.

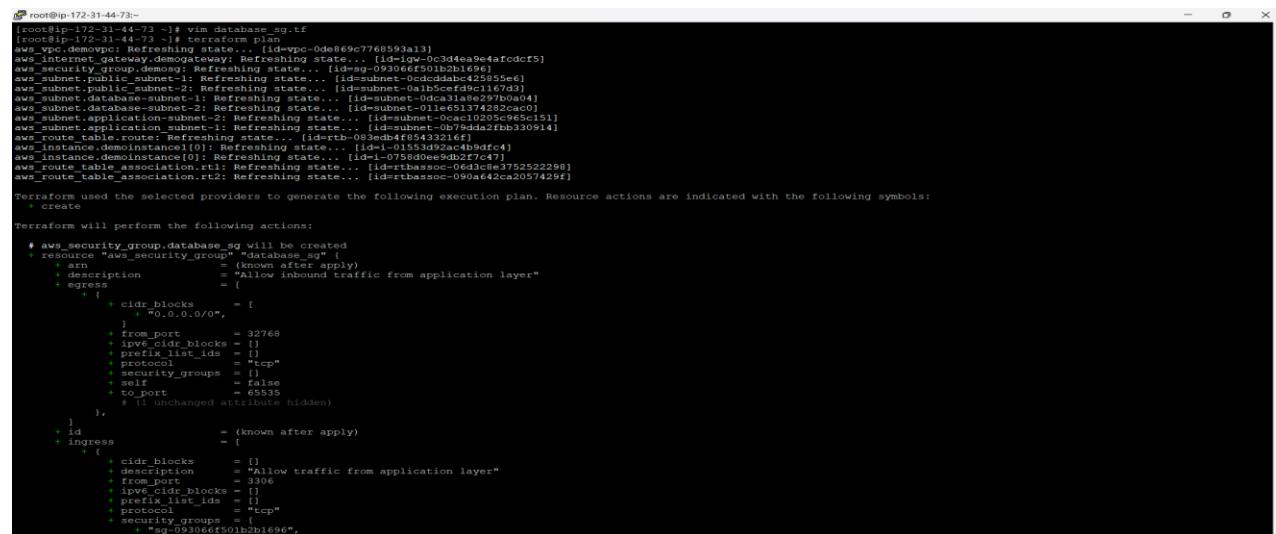


STEP 9: CREATING A FILE FOR SECURITY GROUP FOR THE DATABASE TIER

The Security Group for the Database Tier is like a gatekeeper. It allows only specific resources (like your Application Tier) to access the database. We set rules to allow only trusted traffic (e.g., from specific IP addresses or subnets) and block all others. This ensures your database is protected and only accessible by the parts of your system that need it. So because of that I have created an **database_sg.tf** file.



```
root@ip-172-31-44-73:~# vim database_sg.tf
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"
  }
}
~
```



```
root@ip-172-31-44-73:~# vim database_sg.tf
[root@ip-172-31-44-73 ~]# terraform plan
aws_vpc.demovpc: Refreshing state... [id=vpc-0d8e69c776d598a13]
aws_security_group.demosg: Refreshing state... [id=sg-0c3d4ea9e4afcdcf5]
aws_subnet.public_subnet_1: Refreshing state... [id=subnet-0cdcdabdc425855e]
aws_subnet.private_subnet_1: Refreshing state... [id=subnet-0dca31a4e297b0a04]
aws_subnet.database_subnet_1: Refreshing state... [id=subnet-011e651374282caco]
aws_subnet.application_subnet_1: Refreshing state... [id=subnet-0b79e22f8330914]
aws_route_table.route: Refreshing state... [id=rtb-083edb4f85433216f]
aws_instance.demoinstance[0]: Refreshing state... [id=l-01553d92a4fc4fc4]
aws_route_table_association.rtl: Refreshing state... [id=rthassoc-0643c8e375252298]
aws_route_table_association.rt2: Refreshing state... [id=rthassoc-090a642ca2057429f]

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

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,
          + ip_protocol_blocks = [],
          + prefix_list_ids = [],
          + protocol = "tcp",
          + security_groups = [],
          + state = "allow",
          + to_port = 65535,
        },
    ],
  + id = (known after apply)
  + ingress = [
      + {
          + cidr_blocks = []
          + description = "Allow traffic from application layer"
          + from_port = 3306
          + ip_protocol_blocks = []
          + prefix_list_ids = []
          + protocol = "tcp"
          + security_groups = [
              + "sg-093066f501b2b1696",
            ],
        },
    ],
}
```

I have created a Security Group that allows only specific sources (like the Application Tier) to access the database on specific port.

```
root@ip-172-31-44-73:~#
+ protocol          = "tcp"
+ security_groups  = [
+   + "sg-093066f501b2b1696",
+ ]
+ self              = false
+ to_port           = 3306
},
1
+ 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-0de869c7768593a13"
}

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-172-31-44-73 ~]# terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-0de869c7768593a13]
aws_security_group.demosg: Refreshing state... [id=sg-093066f501b2b1696]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-0a1b5cefd9c11e7d3]
aws_subnet.database_subnet-2: Refreshing state... [id=subnet-011e651374282cae0]
aws_subnet.database_subnet-1: Refreshing state... [id=subnet-0dcda1a8e297b0a04]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-0b79dd42fb5330914]
aws_subnet.application_subnet-2: Refreshing state... [id=subnet-0cac10205c965c151]
aws_internet_gateway.demogateway: Refreshing state... [id=igw-0c3d4ea9e4afcdcf]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-0a1b5cefd9c11e7d3]
aws_instance.demoinstance[0]: Refreshing state... [id=i-0750dd059b2a4247]
aws_route_table.route[0]: Refreshing state... [id=rtb-083eb4f5433216f]
aws_instance.demoinstance[0]: Refreshing state... [id=i-01553d92a2c4b9dfc4]
aws_route_table_association.rtl: Refreshing state... [id=rthasssoc-06d1c8e375252298]
aws_route_table_association.rt2: Refreshing state... [id=rthasssoc-09a642ca2057429f]

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             = [

```

I have applied it successfully using terraform apply command.

```
root@ip-172-31-44-73:~#
+ from_port         = 32768
+ ipv4.cidr_blocks = []
+ prefix_list_ids  = []
+ protocol          = "tcp"
+ security_groups   = []
+ self              = false
+ to_port           = 65535
  # (1 unchanged attribute hidden)
},
1
+ id                = "(known after apply)"
+ ingress           = [
+   +
+     + cidr_blocks      = []
+     + description       = "Allow traffic from application layer"
+     + from_port         = 3306
+     + ipv6.cidr_blocks = []
+     + prefix_list_ids  = []
+     + protocol          = "tcp"
+     + security_groups   = [
+       + "sg-093066f501b2b1696",
+     ]
+     + self              = false
+     + to_port           = 3306
   ],
+ 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-0de869c7768593a13"
}

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 2s [id=sg-018d4f3013081407b]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-44-73 ~]#
```

In the below image we can able to see that our database security group has been created successfully. We can able to see the inbound rules.

The screenshot shows the AWS EC2 Security Groups console. On the left, there's a navigation sidebar with options like Dashboard, Instances, Images, Elastic Block Store, Network & Security, and Load Balancing. The main area displays a table of security groups. One row is selected, labeled "Database SG" with the ID "sg-018d4f3013081407b". Below this, a detailed view for "sg-018d4f3013081407b - Database SG" is shown. The "Inbound rules" tab is active, displaying one rule: "sgr-0f7804183c18c7a25" which allows MySQL/Aurora traffic on port 3306 from source "sg-093066f501b2b16...". Other tabs include Details, Outbound rules, Sharing - new, VPC associations - new, and Tags.

In the below image we can able to see that the outbound rules for the Database security group.

This screenshot is identical to the previous one, showing the AWS EC2 Security Groups console. The "Outbound rules" tab is now active for the "Database SG" security group. It shows one rule: "sgr-07c5b409976508f0d" which uses Custom TCP on port range 32768-65535 to destination 0.0.0.0/0. The rest of the interface is the same, including the sidebar and the detailed view for the selected security group.

STEP 10: CREATE A FILE FOR THE APPLICATION LOAD BALANCER

I have created an Application Load Balancer because it distribute the incoming traffic to various servers. So I have created an alb.tf file for the load balancer and also I have created a target group.

Load Balancer (aws_lb):

name: The name of the ALB.

internal: Whether the ALB is public (false) or private (true).

load balancer type: Specifies it's an application load balancer.

security groups: The security group attached to the ALB to control traffic.

subnets: Defines the public subnets where the ALB will operate.

Listener (aws_lb_listener):

Specifies how the ALB listens for traffic (e.g., HTTP on port 80).

default action: Forwards traffic to the target group

Target Group (aws_lb_target_group):

Groups the backend instances (e.g. EC2 or containers) the ALB routes traffic to

Includes health checks to monitor the health of backend resources

After creation of alb.tf then I have used terraform plan

```
root@ip-172-31-44-73:~# vim alb.tf
[root@ip-172-31-44-73 ~]# terraform plan
aws_vpc.demo: Refreshing state... [id=vpc-0de869c7768593a13]
aws_subnet.public: Refreshing state... [id=subnet-0a1b5cefd9c1167d3]
aws_subnet.application: Refreshing state... [id=subnet-0cdccddabc425855e6]
aws_subnet.database: Refreshing state... [id=subnet-0dc3a1a8e297b0a04]
aws_internet_gateway.demogateway: Refreshing state... [id=igw-0c3d4ca8e4afcdcf5]
aws_subnet.database: Refreshing state... [id=subnet-011ee51374282ac0]
aws_security_group.demosg: Refreshing state... [id=msg-093066f501b2b1696]
aws_subnet.application: Refreshing state... [id=subnet-0b79ddaa2fb330914]
aws_subnet.public: Refreshing state... [id=subnet-0cdccddabc425855e6]
aws_route_table.route: Refreshing state... [id=rtb-083edbf4f5433216f]
aws_security_group.database sg: Refreshing state... [id=sg-018d4df3013081407b]
aws_instance.demoinstance[0]: Refreshing state... [id=i-0758d0ee9db2f7c47]
aws_instance.demoinstance[0]: Refreshing state... [id=i-01553d92a4b9dfc4]
aws_route_table_association.rtl: Refreshing state... [id=rtbassoc-063dc8e3752522298]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-090a642ca2057429f]

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

Terraform will perform the following actions:

# aws_lb.external_lb will be created
+ resource "aws_lb" "external_lb" {
    + arn = (known after apply)
    + arn_suffix = (known after apply)
    + client_keep_alive = 60
    + connection_drift_mode = "defensive"
    + dns_name = (known after apply)
    + drop_invalid_header_fields = false
    + enable_deletion_protection = false
    + enable_http2 = true
    + enable_tls_version_and_cipher_suite_headers = false
    + enable_waf_fail_open = false
    + enable_xff_client_port = false
    + enforce_security_group_inbound_rules_on_private_link_traffic = (known after apply)
    + id = 60
    + idle_timeout = false
    + internal = (known after apply)
    + ip_address_type = "application"
    + load_balancer_type = "external-lb"
    + name = (known after apply)
    + name_prefix = (known after apply)
    + preserve_host_header = false
    + security_groups = [
        + "sg-093066f501b2b1696",
    ]
    + subnets = [
        + "subnet-0a1b5cefd9c1167d3",
        + "subnet-0cdccddabc425855e6",
    ]
}
```

```
root@ip-172-31-44-73:~#
+ preserve_host_header = false
+ security_groups = [
    + "sg-093066f501b2b1696",
]
+ subnets = [
    + "subnet-0a1b5cefd9c1167d3",
    + "subnet-0cdccddabc425855e6",
]
+ tags_all = (known after apply)
+ vpc_id = (known after apply)
+ xff_header_processing_mode = "append"
+ zone_id = (known after apply)

+ subnet_mapping (known after apply)
}

# aws_lb.listener.external_elb will be created
+ resource "aws_lb_listener" "external_elb" {
    + 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)
    + tcp_idle_timeout_seconds = (known after apply)

    + default_action {
        + order = (known after apply)
        + target_group_arn = (known after apply)
        + type = "forward"
    }

    + mutual_authentication (known after apply)
}

# aws_lb.target_group.target_elb will be created
+ resource "aws_lb_target_group" "target_elb" {
    + arn = (known after apply)
    + arn_suffix = (known after apply)
    + connection_termination = (known after apply)
    + deregistration_delay = "300"
    + id = (known after apply)
    + ip_address_type = (known after apply)
    + lambda_multi_value_headers_enabled = false
    + load_balancer_arns = (known after apply)
    + load_balancing_algorithm_type = (known after apply)
    + load_balancing_anomaly_mitigation = (known after apply)
    + load_balancing_cross_zone_enabled = (known after apply)
    + name = "alb-tg"
    + name_prefix = (known after apply)
    + port = 80
}
```

```

root@ip-172-31-44-73:~#
+ arn_suffix = "(known after apply)"
+ connection_termination = "(known after apply)"
+ deregistration_delay = "300"
+ id = "(known after apply)"
+ ip_address_type = "(known after apply)"
+ lambda_multi_value_headers_enabled = false
+ load_balancer_arns = "(known after apply)"
+ load_balancing_algorithm_type = "(known after apply)"
+ load_balancing_anomaly_mitigation = "(known after apply)"
+ load_balancing_cross_zone_enabled = "(known after apply)"
+ name = "alb-tg"
+ name_prefix = "(known after apply)"
+ port = 80
+ preserve_client_ip = "(known after apply)"
+ protocol = "HTTP"
+ protocol_version = "(known after apply)"
+ proxy_protocol_v2 = false
+ slot_start = 0
+ tag_all = "(known after apply)"
+ target_type = "instance"
+ vpc_id = "vpc-0de869c7768593a13"

+ health_check (known after apply)
+ stickiness (known after apply)
+ target_failover (known after apply)
+ target_group_health (known after apply)
+ target_health_state (known after apply)
}

# aws_lb_target_group_attachment.attachment will be created
resource "aws_lb_target_group_attachment" "attachment" {
  id      = "(known after apply)"
  port    = 80
  target_group_arn = "(known after apply)"
  target_id = "i-0758d0ee9db2f7c47"
}

# aws_lb_target_group_attachment.attachment2 will be created
resource "aws_lb_target_group_attachment" "attachment2" {
  id      = "(known after apply)"
  port    = 80
  target_group_arn = "(known after apply)"
  target_id = "i-01553d92ac4b9dfc4"
}

Plan: 5 to add, 0 to change, 0 to destroy.

```

Now we need to perform an terraform apply for the creation of the application load balancer.

```

root@ip-172-31-44-73:~#
Plan: 5 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".
[root@ip-172-31-44-73 ~]# terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-0de869c7768593a13]
aws_security_group.demosg: Refreshing state... [id=sg-093066f501bcb1696]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-011e65137422cae0]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0234c834f9de4f5]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0ba1a8e297bfa04]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0b793da2fb330914]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-0a1b5ccfd9c1167a43]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-0cc4cdabc425855e6]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-0cac10205c965c151]
aws_security_group.database_sg: Refreshing state... [id=sg-0184df3013081407b]
aws_route_table.route: Refreshing state... [id=rtb-083edb4f05433216f]
aws_instance.demoinstance[0]: Refreshing state... [id=i-01553d92ac4b9dfc4]
aws_route_table_association.rtl: Refreshing state... [id=rbassoc-0643c8e3752522298]
aws_route_table_association.rt2: Refreshing state... [id=rbassoc-090a642ca2057429f]

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

Terraform will perform the following actions:

# aws_lb.external_lb will be created
resource "aws_lb" "external_lb" {
  arn           = "(known after apply)"
  arn_suffix    = "(known after apply)"
  client_keep_alive = 3600
  desync_mitigation_mode = "defensive"
  dns_name     = "(known after apply)"
  drop_invalid_header_fields = false
  enable_deletion_protection = false
  enable_http2   = true
  enable_tls_version_and_cipher_suite_headers = false
  enable_waf_fail_open = false
  enable_xff_client_port = false
  enforce_security_group_inbound_rules_on_private_link_traffic = "(known after apply)"
  id            = "(known after apply)"
  idle_timeout  = 60
  internal      = false
  ip_address_type = "application"
  load_balancer_type = "external-lb"
  name          = "(known after apply)"
  name_prefix   = false
  preserve_host_header = false
  security_groups = [
}
```

```

root@ip-172-31-44-73:~#
[+] subnets = [
  + "subnet-0a1b5cefd9c1167d3",
  + "subnet-0cdcdabc42585e6",
]
tags_all = (known after apply)
vpc_id = (known after apply)
xff_header_processing_mode = "append"
zone_id = (known after apply)

+ subnet_mapping (known after apply)
}

# aws_lb_listener.external_elb will be created
resource "aws_lb_listener" "external_elb" {
  + 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)
  + tcp_idle_timeout_seconds = (known after apply)

  + default_action {
    + order = (known after apply)
    + target_group_arn = (known after apply)
    + type = "forward"
  }

  + mutual_authentication (known after apply)
}

# aws_lb_target_group.target_elb will be created
resource "aws_lb_target_group" "target_elb" {
  + arn = (known after apply)
  + arn_suffix = (known after apply)
  + connection_termination = (known after apply)
  + deregistration_delay = "300"
  + id = (known after apply)
  + ip_address_type = (known after apply)
  + lambda_multi_value_headers_enabled = false
  + load_balancer_arns = (known after apply)
  + load_balancing_algorithm_type = (known after apply)
  + load_balancing_anomaly_mitigation = (known after apply)
  + load_balancing_cross_zone_enabled = (known after apply)
  + name = "lb-cg"
  + name_prefix = (known after apply)
  + port = 80
  + preserve_client_ip = (known after apply)
  + protocol = "HTTP"
  + protocol_version = (known after apply)
}

```

```

root@ip-172-31-44-73:~#
[+] port = 80
[+] preserve_client_ip = (known after apply)
[+] protocol = "HTTP"
[+] protocol_version = (known after apply)
[+] proxy_protocol_v2 = false
[+] slow_start = 0
[+] tags_all = (known after apply)
[+] target_type = "instance"
[+] vpc_id = "vpc-0de869c7768593a13"

+ health_check (known after apply)
+ stickiness (known after apply)
+ target_failover (known after apply)
+ target_group_health (known after apply)
+ target_health_state (known after apply)
}

# aws_lb_target_group_attachment.attachment will be created
resource "aws_lb_target_group_attachment" "attachment" {
  + id = (known after apply)
  + port = 80
  + target_group_arn = (known after apply)
  + target_id = "i-0758d0ee9db2f7c47"
}

# aws_lb_target_group_attachment.attachment2 will be created
resource "aws_lb_target_group_attachment" "attachment2" {
  + id = (known after apply)
  + port = 80
  + target_group_arn = (known after apply)
  + target_id = "i-01553d92ac4b9dfc4"
}

Plan: 5 to add, 0 to change, 0 to destroy.

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

Enter a value: yes

aws_lb_external_elb: Creating...
aws_lb_target_group_target_elb: Creating...
aws_lb_target_group_target_elb: Creation complete after 1s [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/alb-tg/bebbde15c770c149]
aws_lb_target_group_attachment_attachment2: Creating...
aws_lb_target_group_attachment_attachment: Creating...
aws_lb_target_group_attachment_attachment: Creation complete after 0s [id=arn:aws:elasticload

```

In the below image we can able to see that load balancer has been created successfully.

```
root@ip-172-31-44-73:~#
+ id          = (known after apply)
+ port        = 80
+ target_group_arn = (known after apply)
+ target_id   = "i-01553d92ac4b5dfc4"
+
Plan: 5 to add, 0 to change, 0 to destroy.

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

Enter a value: yes

aws_lb.external_lb: Creating...
aws_lb.target_group.target_elb: Creating...
aws_lb.target_group.target_elb: Creation complete after 1s [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/tg/bebbde15c770c149]
aws_lb.target_group.attachment.attachment2: Creating...
aws_lb.target_group.attachment.attachment: Creating...
aws_lb.target_group.attachment.attachment: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/tg/bebbde15c770c149-20241119161705933700000001]
aws_lb.target_group.attachment.attachment2: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/tg/bebbde15c770c149-20241119161706082400000002]
aws_lb.external_lb: Still creating... [10s elapsed]
aws_lb.external_lb: Still creating... [20s elapsed]
aws_lb.external_lb: Still creating... [30s elapsed]
aws_lb.external_lb: Still creating... [40s elapsed]
aws_lb.external_lb: Still creating... [50s elapsed]
aws_lb.external_lb: Still creating... [1m0s elapsed]
aws_lb.external_lb: Still creating... [1m10s elapsed]
aws_lb.external_lb: Still creating... [1m20s elapsed]
aws_lb.external_lb: Still creating... [1m30s elapsed]
aws_lb.external_lb: Still creating... [1m40s elapsed]
aws_lb.external_lb: Still creating... [1m50s elapsed]
aws_lb.external_lb: Still creating... [2m0s elapsed]
aws_lb.external_lb: Still creating... [2m10s elapsed]
aws_lb.external_lb: Still creating... [2m20s elapsed]
aws_lb.external_lb: Still creating... [2m30s elapsed]
aws_lb.external_lb: Still creating... [2m40s elapsed]
aws_lb.external_lb: Still creating... [2m50s elapsed]
aws_lb.external_lb: Still creating... [3m0s elapsed]
aws_lb.external_lb: Still creating... [3m10s elapsed]
aws_lb.external_lb: Creation complete after 3m12s [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:loadbalancer/app/external-lb/b838d4c726e0b907]
aws_lb.listener.external_elb: Creating...
aws_lb.listener.external_elb: Creation complete after 1s [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:listener/app/external-lb/b838d4c726e0b907/1ad49ea479806168]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
[root@ip-172-31-44-73 ~]# vim alb.tf
[root@ip-172-31-44-73 ~]#
```

Now we can able to see the application load balancer in the below image.

Load balancers (1/1)

Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
external-lb	external-lb-1471902147.us-east-1.elb.amazonaws...	Active	vpc-0de869c7768593...	2 Availability Zones	application	November 19, 2024, 21:47 (U...

Load balancer: external-lb

Listeners and rules (1) info

Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Tr...
HTTP:80	Forward to target group	1 rule	ARN	Not applicable	Not applicable	Not applicable	No...
	* tg (100%)						
	* Target group stickiness: Off						

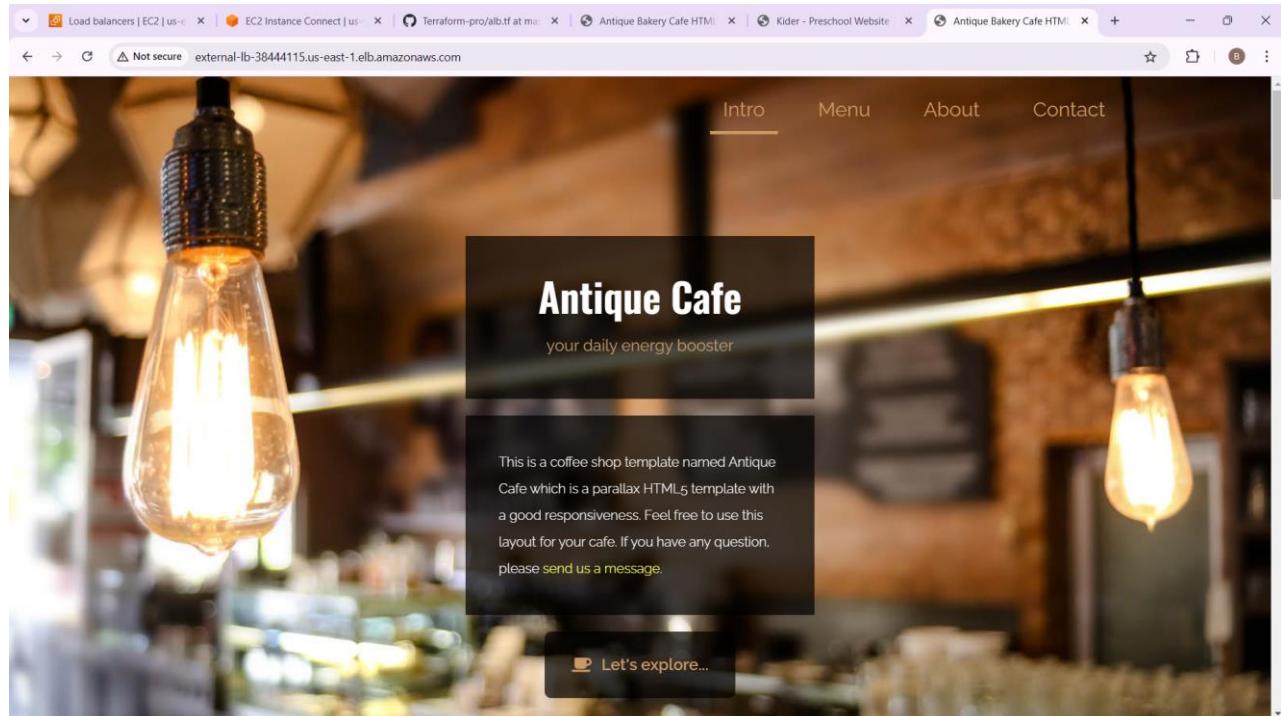
In the below image we can able to see the route map for the load balancer.

The screenshot shows the AWS CloudWatch Metrics console with the URL <https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LoadBalancers>. The left sidebar shows the navigation menu for EC2, including Instances, Network & Security, and Load Balancing. The main content area displays the 'Load balancers' page for 'external-lb'. The 'Resource map - new' tab is selected. The diagram shows a Listener (HTTP:80) connected to a Rule (Priority default: Forward to target group). The Target groups section shows one target group named 'alb-tg' with two targets, both marked as healthy. The Targets section lists two instances: i-01553d92ac4b9dfc4 and i-0758d0ee9db2f7c47, both with Port 80.

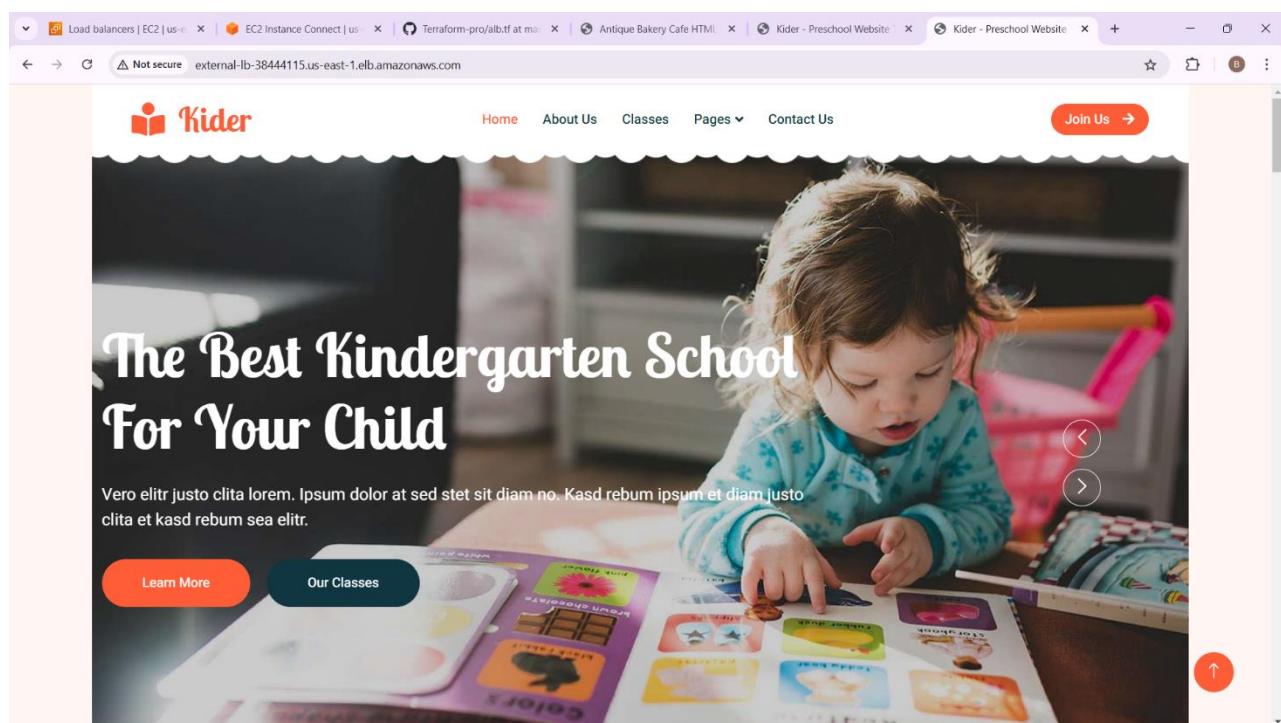
In the below image we can able to see that the target group for the load balancer and both the servers are healthy.

The screenshot shows the AWS CloudWatch Metrics console with the URL <https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#TargetGroups>. The left sidebar shows the navigation menu for EC2, including Instances, Network & Security, and Load Balancing. The main content area displays the 'Target groups' page for 'alb-tg'. The 'Targets' tab is selected, showing two registered targets: 'i-01553d92ac4b9dfc4' and 'i-0758d0ee9db2f7c47', both of which are marked as healthy.

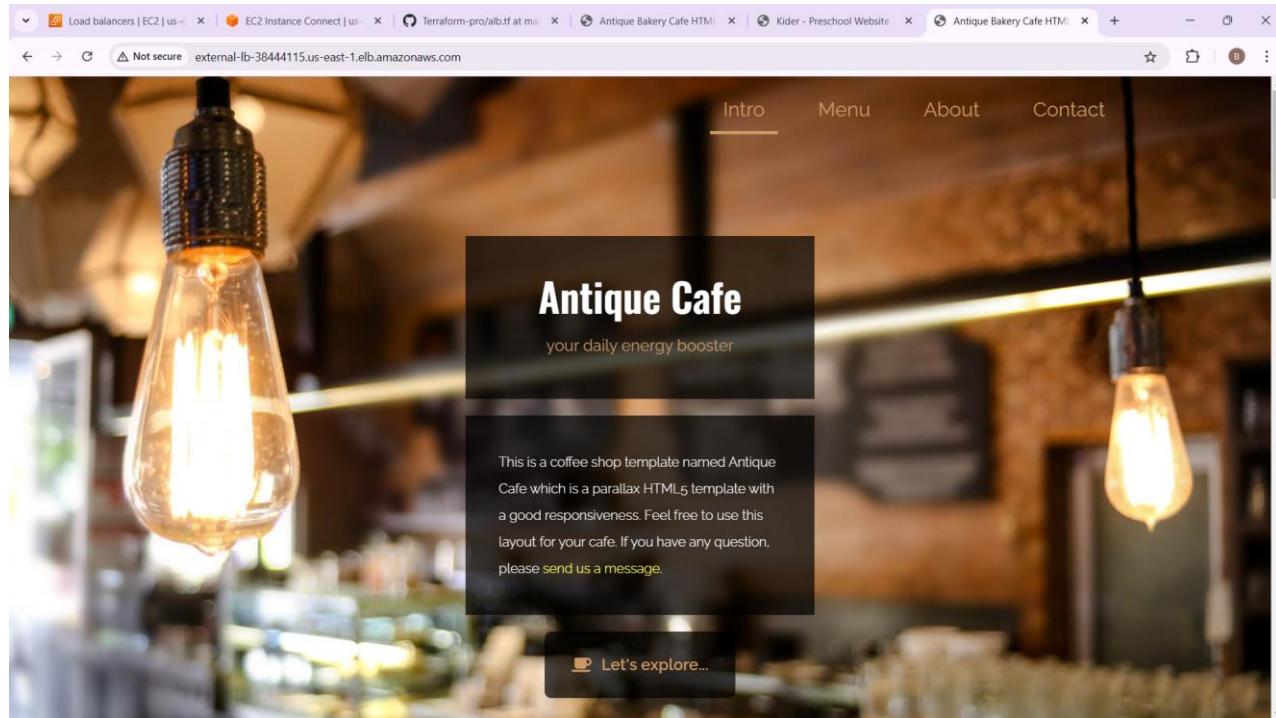
Copy the DNS of the load balancer and paste it in browser we can able to see the servers are working properly it will fluctuate the servers.



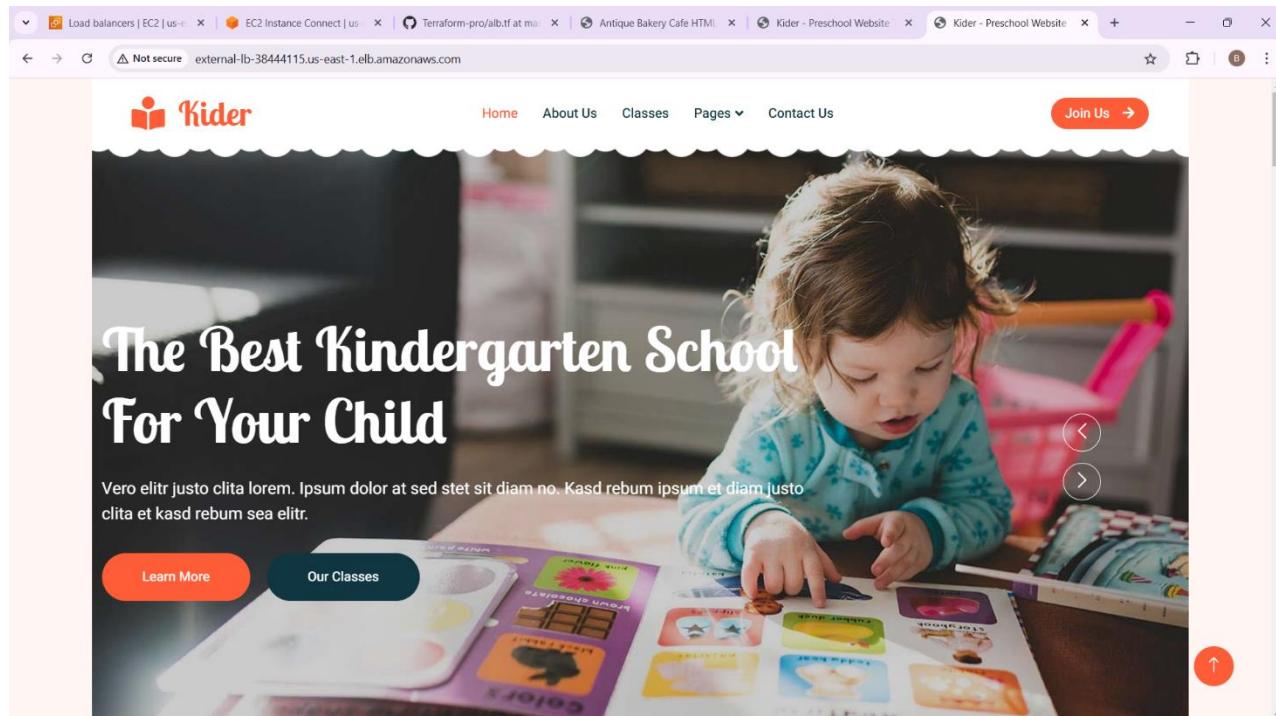
It shows the second server when I reload.



After reloading it shows the below web application

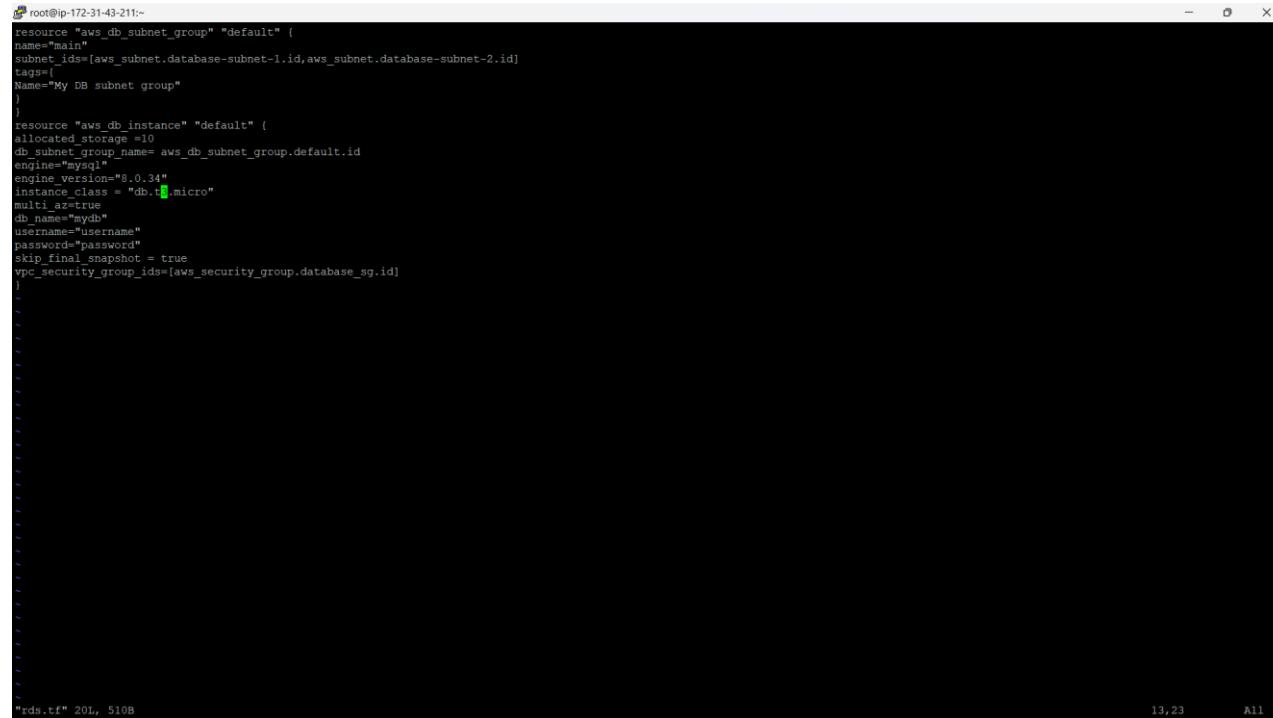


The load is distributing through two servers.



STEP 11: CREATING A FILE FOR AN RDS INSTANCE

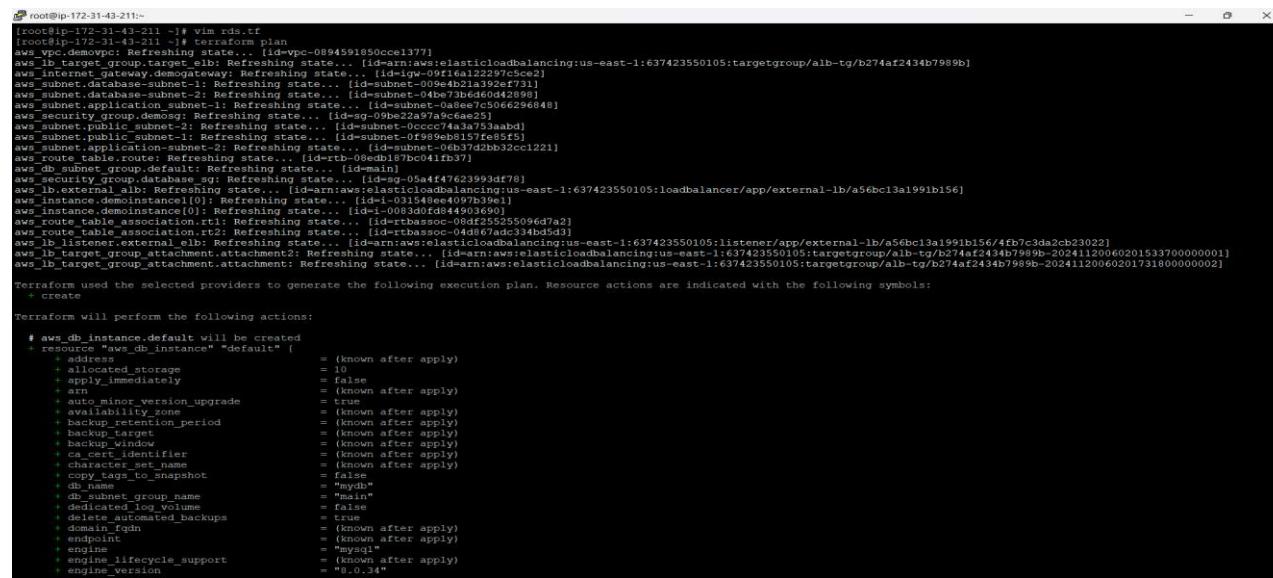
The **rds.tf** file defines a managed database instance in AWS. It specifies the type of database, storage size, and networking rules. This setup ensures your database is secure and only accessible by the required application or resources.



```
root@ip-172-31-43-211:~#
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 = "8.0.34"
  instance_class = "db.t2.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]
}

rds.tf" 20L, 510B
```

Then i have run the terraform plan command to check the configuration before creating an rds.



```
[root@ip-172-31-43-211:~]# vim rds.tf
[root@ip-172-31-43-211:~]# terraform plan
aws_vpc.demoVPC: Refreshing state... [id=vpc-0894591850cce1377]
aws_lb.target_group.target_elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/alb-tg/b274af2434b7989b]
aws_internet_gateway.demoInternetGateway: Refreshing state... [id=igw-09f112297c5ce2]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-04be73b6d6042898]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-04be73b6d6042898]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0a9e87c5066296848]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-0a9e87c5066296848]
aws_security_group.demoSG: Refreshing state... [id=sg-022a7a9c9c28]
aws_subnet_public.demoSubnetPublic: Refreshing state... [id=subnet-04837a4935aa8bd]
aws_subnet_public_subnet-1: Refreshing state... [id=subnet-0f89eb8157fe85f5]
aws_subnet_application-subnet-2: Refreshing state... [id=subnet-06b37d2bb32cc1221]
aws_route_table.route1: Refreshing state... [id=rtb-08ed07bc041fb37]
aws_route_table.route2: Refreshing state... [id=rtb-08ed07bc041fb37]
aws_security_group.database_sg: Refreshing state... [id=sg-05a447476239393df78]
aws_lb_external_lb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:loadbalancer/app/external-lb/a56bc13a1991b156]
aws_db_instance.demoDB: Refreshing state... [id=dbi-0083ddfd84903490]
aws_internet_gateway.attachment[0]: Refreshing state... [id=tbassoc-08df255255096d7a2]
aws_route_table_association.association1: Refreshing state... [id=tbassoc-08df255255096d7a2]
aws_route_table_association.association2: Refreshing state... [id=tbassoc-04d867ad334bd5d3]
aws_lb_listener.external_lb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:listener/app/external-lb/a56bc13a1991b156/4fb7c3de2cb3922]
aws_lb_target_group_attachment.attachment1: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/alb-tg/b274af2434b7989b-20241120060201533700000001]
aws_lb_target_group_attachment.attachment2: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/alb-tg/b274af2434b7989b-20241120060201731800000002]

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

Terraform will perform the following actions:

# aws_db_instance.default will be created
+ resource "aws_db_instance" "default" {
    + address = (known after apply)
    + allocated_storage = 10
    + arn = (known after apply)
    + auto_minor_version_upgrade = true
    + availability_zone = (known after apply)
    + backup_retention_period = (known after apply)
    + backup_target = (known after apply)
    + backup_window = (known after apply)
    + ca_cert_identifier = (known after apply)
    + character_set_name = (known after apply)
    + copy_tags_to_snapshot = false
    + db_name = "mydb"
    + db_subnet_group_name = "main"
    + deleted_auto_backups = false
    + delete_automated_backups = true
    + domain_fqdn = (known after apply)
    + endpoint = (known after apply)
    + engine = "mysql"
    + engine_version = "8.0.34"
```

```

root@ip-172-31-43-211:~#
+ db_name          = "mydb"
+ db_subnet_group_name = "main"
+ dedicated_log_volume = false
+ delete_automated_backups = true
+ domain_fqdn      = (known after apply)
+ engine           = "mysql"
+ engine_version   = (known after apply)
+ engine_version_actual = "8.0.34"
+ hosted_zone_id   = (known after apply)
+ id                = (known after apply)
+ identifier        = (known after apply)
+ identifier_prefix = (known after apply)
+ instance_class    = "db.t3.micro"
+ iops              = (known after apply)
+ kms_key_id        = (known after apply)
+ latest_restorable_time = (known after apply)
+ license_model     = (known after apply)
+ listener_endpoint = (known after apply)
+ maintenance_window = (known after apply)
+ master_user_secret = (known after apply)
+ monitoring_user_secret_kms_key_id = (known after apply)
+ monitoring_interval = 0
+ monitoring_role_arn = (known after apply)
+ multi_az          = true
+ nchar_character_set_name = (known after apply)
+ network_type       = (known after apply)
+ option_group_name = (known after apply)
+ parameter_group_name = (known after apply)
+ password          = (sensitive value)
+ performance_insights_enabled = false
+ performance_insights_kms_key_id = (known after apply)
+ performance_insights_retention_period = (known after apply)
+ port               = (known after apply)
+ publicly_accessible = false
+ replica_mode       = (known after apply)
+ replicas          = (known after apply)
+ resource_id        = (known after apply)
+ skip_final_snapshot = true
+ snapshot_identifier = (known after apply)
+ status             = (known after apply)
+ storage_throughput = (known after apply)
+ storage_type       = (known after apply)
+ tags_all          = (known after apply)
+ timezone          = (known after apply)
+ username           = "username"
+ vpc_security_group_ids = [
  + "sg-05a4f47e623993df78",
]
}

```

In the below image we can see terraform apply to create an rds.

```

root@ip-172-31-43-211:~#
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-172-31-43-211 ~]# terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-0894591850ccc1377]
aws_lb_target_group.target_elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/alb-tg/b274af2434b7989b]
aws_internet_gateway.demogateway: Refreshing state... [id=igw-09f16a122297c5ce2]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-0cccc74a3a753aa9d]
aws_security_group.demosg: Refreshing state... [id=sg-09be22a97a9c6ae25]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-009e4b21a392ef731]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-04be73b6d6042898]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-0f899eb8157fe85f5]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-0637d2b32cc1221]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0a8ee7c5066296648]
aws_security_group.database_sg: Refreshing state... [id=sg-05a4f47623993df78]
aws_route_table.route: Refreshing state... [id=rtb-08edb167bc041fb37]
aws_instance.demonstance1[0]: Refreshing state... [id=i-031548e4e097b39e1]
aws_db_subnet_group.default: Refreshing state... [id=main]
aws_instance.demonstance0[0]: Refreshing state... [id=i-0083ad0fd844903690]
aws_lb_listener.external_elb: Refreshing state... [id=lb-tbassoc-08df255255096d7a2]
aws_route_table_association.rt1: Refreshing state... [id=rtaassoc-04d867adc334hd5d1]
aws_lb_listener.external_elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:listener/app/external-lb/a56bc13a1991b156]
aws_lb_target_group.attachment.attachment2: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/alb-tg/f2434b7989b-2024112006020153370000000002]
aws_lb_target_group_attachment.attachment: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/alb-tg/b274af2434b7989b-202411200602017318000000002]

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_db_instance.default will be created
+ resource "aws_db_instance" "default" {
  + address          = (known after apply)
  + allocated_storage = 10
  + apply_immediately = false
  + arn               = (known after apply)
  + auto_minor_version_upgrade = true
  + availability_zone = (known after apply)
  + backup_retention_period = (known after apply)
  + backup_target      = (known after apply)
  + backup_window      = (known after apply)
  + ca_cert_identifier = (known after apply)
  + character_set_name = (known after apply)
  + copy_tags_to_snapshot = false
  + db_name           = "mydb"
  + db_subnet_group_name = "main"
  + dedicated_log_volume = false
  + delete_automated_backups = true
}

```

```

root@ip-172-31-43-211:~#
+ backup_retention_period      = (known after apply)
+ backup_target                = (known after apply)
+ backup_window                = (known after apply)
+ ca_cert_identifier           = (known after apply)
+ character_set_name          = (known after apply)
+ character_set_noconvert     = (known after apply)
+ character_set_noconvert_c   = (known after apply)
+ db_name                      = "mydb"
+ db_subnet_group_name         = "main"
+ dedicated_log_volume        = false
+ delete_automated_backups    = true
+ domain_fqdn                 = (known after apply)
+ endpoint                     = (known after apply)
+ engine                       = "mysql"
+ engine.lifecycle_support     = (known after apply)
+ engine_version               = "8.0.34"
+ engine.version_actual        = (known after apply)
+ hosted_zone_id              = (known after apply)
+ id                           = (known after apply)
+ identifier                   = (known after apply)
+ identifier_prefix            = (known after apply)
+ instance_class               = "db.t3.micro"
+ iops                         = (known after apply)
+ key_id                      = (known after apply)
+ latest_restorable_time       = (known after apply)
+ license_model                = (known after apply)
+ listener_endpoint             = (known after apply)
+ maintenance_window           = (known after apply)
+ master_user_secret           = (known after apply)
+ master_user_secret_kms_key_id= 0
+ monitoring_interval          = (known after apply)
+ monitoring_role_arn          = (known after apply)
+ multi_az                     = true
+ nchar_character_set_name     = (known after apply)
+ network_type                 = (known after apply)
+ option_group_name            = (known after apply)
+ parameter_group_name         = (known after apply)
+ password                     = (sensitive value)
+ performance_insights_enabled= false
+ performance_insights_kms_key_id= (known after apply)
+ performance_insights_retention_period= (known after apply)
+ port                         = (known after apply)
+ publicly_accessible          = false
+ replica_mode                 = (known after apply)
+ replicas                     = (known after apply)
+ resource_id                  = (known after apply)
+ skip_final_snapshot          = true
+ snapshot_identifier           = (known after apply)
+ status                        = (known after apply)
+ storage_throughput            = (known after apply)
+ storage_type                 = (known after apply)
+ tags_all                     = (known after apply)
+ timezone                     = (known after apply)

root@ip-172-31-43-211:~#
+ storage_type                 = (known after apply)
+ tags_all                     = (known after apply)
+ timezone                     = (known after apply)
+ username                      = "username"
+ vpc_security_group_ids        = [
  ]
}

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_db_instance.default: Creating...
aws_db_instance.default: Still creating... [10s elapsed]
aws_db_instance.default: Still creating... [20s elapsed]
aws_db_instance.default: Still creating... [30s elapsed]
aws_db_instance.default: Still creating... [40s elapsed]
aws_db_instance.default: Still creating... [50s elapsed]
aws_db_instance.default: Still creating... [1m0s elapsed]
aws_db_instance.default: Still creating... [1m10s elapsed]
aws_db_instance.default: Still creating... [1m20s elapsed]
aws_db_instance.default: Still creating... [1m30s elapsed]
aws_db_instance.default: Still creating... [1m40s elapsed]
aws_db_instance.default: Still creating... [1m50s elapsed]
aws_db_instance.default: Still creating... [2m0s elapsed]
aws_db_instance.default: Still creating... [2m10s elapsed]
aws_db_instance.default: Still creating... [2m20s elapsed]
aws_db_instance.default: Still creating... [2m30s elapsed]
aws_db_instance.default: Still creating... [2m40s elapsed]
aws_db_instance.default: Still creating... [2m50s elapsed]
aws_db_instance.default: Still creating... [3m0s elapsed]
aws_db_instance.default: Still creating... [3m10s elapsed]
aws_db_instance.default: Still creating... [3m20s elapsed]
aws_db_instance.default: Still creating... [3m30s elapsed]
aws_db_instance.default: Still creating... [3m40s elapsed]
aws_db_instance.default: Still creating... [3m50s elapsed]
aws_db_instance.default: Still creating... [4m0s elapsed]
aws_db_instance.default: Still creating... [4m10s elapsed]
aws_db_instance.default: Still creating... [4m20s elapsed]
aws_db_instance.default: Still creating... [4m30s elapsed]
aws_db_instance.default: Still creating... [4m40s elapsed]
aws_db_instance.default: Still creating... [4m50s elapsed]
aws_db_instance.default: Still creating... [5m0s elapsed]
aws_db_instance.default: Still creating... [5m10s elapsed]
aws_db_instance.default: Still creating... [5m20s elapsed]
aws_db_instance.default: Still creating... [5m30s elapsed]
aws_db_instance.default: Still creating... [5m40s elapsed]
aws_db_instance.default: Still creating... [5m50s elapsed]
aws_db_instance.default: Still creating... [5m59s elapsed]

aws_db_instance.default: Still creating... [10s elapsed]
aws_db_instance.default: Still creating... [20s elapsed]
aws_db_instance.default: Still creating... [30s elapsed]
aws_db_instance.default: Still creating... [40s elapsed]
aws_db_instance.default: Still creating... [50s elapsed]
aws_db_instance.default: Still creating... [1m0s elapsed]
aws_db_instance.default: Still creating... [1m10s elapsed]
aws_db_instance.default: Still creating... [1m20s elapsed]
aws_db_instance.default: Still creating... [1m30s elapsed]
aws_db_instance.default: Still creating... [1m40s elapsed]
aws_db_instance.default: Still creating... [1m50s elapsed]
aws_db_instance.default: Still creating... [2m0s elapsed]
aws_db_instance.default: Still creating... [2m10s elapsed]
aws_db_instance.default: Still creating... [2m20s elapsed]
aws_db_instance.default: Still creating... [2m30s elapsed]
aws_db_instance.default: Still creating... [2m40s elapsed]
aws_db_instance.default: Still creating... [2m50s elapsed]
aws_db_instance.default: Still creating... [3m0s elapsed]
aws_db_instance.default: Still creating... [3m10s elapsed]
aws_db_instance.default: Still creating... [3m20s elapsed]
aws_db_instance.default: Still creating... [3m30s elapsed]
aws_db_instance.default: Still creating... [3m40s elapsed]
aws_db_instance.default: Still creating... [3m50s elapsed]
aws_db_instance.default: Still creating... [4m0s elapsed]
aws_db_instance.default: Still creating... [4m10s elapsed]
aws_db_instance.default: Still creating... [4m20s elapsed]
aws_db_instance.default: Still creating... [4m30s elapsed]
aws_db_instance.default: Still creating... [4m40s elapsed]
aws_db_instance.default: Still creating... [4m50s elapsed]
aws_db_instance.default: Still creating... [5m0s elapsed]
aws_db_instance.default: Still creating... [5m10s elapsed]
aws_db_instance.default: Still creating... [5m20s elapsed]
aws_db_instance.default: Still creating... [5m30s elapsed]
aws_db_instance.default: Still creating... [5m40s elapsed]
aws_db_instance.default: Still creating... [5m50s elapsed]
aws_db_instance.default: Still creating... [5m59s elapsed]

```

Successfully created RDS.

```
root@ip-172-31-43-211:~# aws db instance.default: Still creating... [3m30s elapsed]
aws db instance.default: Still creating... [3m40s elapsed]
aws db instance.default: Still creating... [3m50s elapsed]
aws db instance.default: Still creating... [4m0s elapsed]
aws db instance.default: Still creating... [4m10s elapsed]
aws db instance.default: Still creating... [4m20s elapsed]
aws db instance.default: Still creating... [4m30s elapsed]
aws db instance.default: Still creating... [4m40s elapsed]
aws db instance.default: Still creating... [4m50s elapsed]
aws db instance.default: Still creating... [5m0s elapsed]
aws db instance.default: Still creating... [5m10s elapsed]
aws db instance.default: Still creating... [5m20s elapsed]
aws db instance.default: Still creating... [5m30s elapsed]
aws db instance.default: Still creating... [5m40s elapsed]
aws db instance.default: Still creating... [5m50s elapsed]
aws db instance.default: Still creating... [6m0s elapsed]
aws db instance.default: Still creating... [6m10s elapsed]
aws db instance.default: Still creating... [6m20s elapsed]
aws db instance.default: Still creating... [6m30s elapsed]
aws db instance.default: Still creating... [6m40s elapsed]
aws db instance.default: Still creating... [6m50s elapsed]
aws db instance.default: Still creating... [7m0s elapsed]
aws db instance.default: Still creating... [7m10s elapsed]
aws db instance.default: Still creating... [7m20s elapsed]
aws db instance.default: Still creating... [7m30s elapsed]
aws db instance.default: Still creating... [7m40s elapsed]
aws db instance.default: Still creating... [7m50s elapsed]
aws db instance.default: Still creating... [8m0s elapsed]
aws db instance.default: Still creating... [8m10s elapsed]
aws db instance.default: Still creating... [8m20s elapsed]
aws db instance.default: Still creating... [8m30s elapsed]
aws db instance.default: Still creating... [8m40s elapsed]
aws db instance.default: Still creating... [8m50s elapsed]
aws db instance.default: Still creating... [9m0s elapsed]
aws db instance.default: Still creating... [9m10s elapsed]
aws db instance.default: Still creating... [9m20s elapsed]
aws db instance.default: Still creating... [9m30s elapsed]
aws db instance.default: Still creating... [9m40s elapsed]
aws db instance.default: Still creating... [9m50s elapsed]
aws db instance.default: Creation complete after 11m19s (id=db-IGPHKB65QJP5KCFXTJURMSQIHY)

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-43-211 ~]# vim rds.tf
[root@ip-172-31-43-211 ~]#
```

In the below image we can able to see that the database has been created successfully.

The screenshot shows the AWS RDS console interface. On the left, there's a navigation sidebar with options like Dashboard, Databases (which is selected), Query Editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom engine versions, Zero-ETL integrations, Events, Event subscriptions, Recommendations (with a notification dot), and Certificate update. The main content area is titled "Databases (1)". It shows a table with one row of data:

DB identifier	Status	Role	Engine	Region ...	Size	Recommendations	CPU	Current...	Mainte...
terraform-2024112006553147790000C	Available	Instance	MySQL Co...	us-east-1b	db.t3.micro		4.14%	0 Conn	none

At the bottom of the page, there are links for CloudShell, Feedback, and a footer with copyright information: © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

Summary

DB identifier	Status	Role	Engine	Recommendations
terraform-20241120065531477900000001	Available	Instance	MySQL Community	
CPU	4.49%	Class	Region & AZ	us-east-1b
		Current activity		
		0 Connections		

Connectivity & security

Endpoint & port	Networking	Security
Endpoint: terraform-20241120065531477900000001.cts.umykgq23d.us-east-1.rds.amazonaws.com	Availability Zone: us-east-1b	VPC security groups: Database SG (sg-05a4ff47623993df78)
Port: 3306	VPC: Demo VPC (vpc-0894591850cce1377)	Active
	Subnet group: main	Publicly accessible: No
	Subnets: subnet-009e4b21a392ef731, subnet-04be73b6d60d42898	Certificate authority: Info rds-ca-rsa2048-g1
	Network type:	Certificate authority date: May 26, 2061, 05:04 (UTC+05:30)
		DB instance certificate expiration date: November 20, 2025, 12:27 (UTC+05:30)

In the below image we can able to see that the configuration of the RDS.

Summary

DB identifier	Status	Role	Engine	Recommendations
terraform-20241120065531477900000001	Available	Instance	MySQL Community	
CPU	4.49%	Class	Region & AZ	us-east-1b
		Current activity		
		0 Connections		

Instance

Configuration	Instance class	Storage	Performance Insights
DB instance ID: terraform-20241120065531477900000001	Instance class: db.t3.micro	Encryption: Not enabled	Performance Insights enabled: Turned off
Engine version: 8.0.34	vCPU: 2	Storage type: General Purpose SSD (gp2)	
RDS Extended Support: Enabled	RAM: 1 GB	Storage: 10 GiB	
DB name: mydb	Availability	Provisioned IOPS: -	
License model: General Public License	Master username: username	Storage throughput: -	
Option groups:	Master password:	Storage auto-scaling:	

STEP 12: CREATING A FILE FOR OUTPUTS

The **outputs.tf** file in Terraform is used to display the results of specific resources or attributes after applying the configuration. This helps users easily retrieve important information, such as instance IP addresses, database connection endpoints, or load balancer DNS names.

```
root@ip-172-31-43-211:~#
output "lb_dns_name" {
  description="The DNS name of the load balancer"
  value= "${aws_lb.external.alb.dns_name}"
}

outputs.tf" 5L, 115B
5,1      All

root@ip-172-31-43-211:~# vim outputs.tf
[root@ip-172-31-43-211 ~]# terraform plan
aws_vpc.demoVPC Refreshing state... [id=vpc-0894591850cce1377]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-04be73d2bb3cc1221]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-0cccc74a3e753aab]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-009e4b21a392ef731]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-04be73b6d60d42898]
aws_security_group.demoSG: Refreshing state... [id=sg-09be22a97a9c6ae25]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0a8ee7c5066296648]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0f89eb8157fe85f5]
aws_lb.target_group.target_elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/alb-tg/b274af2434b7989b]
aws_internet_gateway.demoIGW: Refreshing state... [id=igw-09f16a122297c5ce2]
aws_route_table.route_demoRTT1: Refreshing state... [id=rtb-05a4f47623993df78]
aws_security_group.demoSG: Refreshing state... [id=sg-05a4f47623993df78]
aws_instance.demoInstance[0]: Refreshing state... [id=i-0083d0fd844903690]
aws_lb.external.alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:loadbalancer/app/external-lb/a56bc13a1991b156]
aws_instance.demoInstance[0]: Refreshing state... [id=i-031548ee4097b39e1]
aws_route_table_association.rtl1: Refreshing state... [id=rtsassoc-08df25525596d7a2]
aws_route_table_association.rtl2: Refreshing state... [id=rtsassoc-04d867adc334bd5d3]
aws_db_subnet_group.default: Refreshing state... [id=main]
aws_lb_listener.external.elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:listener/app/external-lb/a56bc13a1991b156/4fb7c3da2cb23022]
aws_db_instance.default: Refreshing state... [id=db-IDPHK65QnP5KCFFXJURMS6HY]
aws_lb_target_group_attachment.attachment2: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/alb-tg/b274af2434b7989b-20241120060201533700000000]
aws_lb_target_group_attachment.attachment: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/alb-tg/b274af2434b7989b-20241120060201731800000002]

Changes to Outputs:
+ lb_dns_name = "external-lb-996635588.us-east-1.elb.amazonaws.com"

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

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

[root@ip-172-31-43-211 ~]# terraform apply
aws_vpc.demoVPC: Refreshing state... [id=vpc-0894591850cce1377]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-04be73b6d60d42898]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-06b37d2bb3cc1221]
aws_internet_gateway.demoIGW: Refreshing state... [id=igw-09f16a122297c5ce2]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-009e4b21a392ef731]
aws_security_group.demoSG: Refreshing state... [id=sg-09be22a97a9c6ae25]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0f89eb8157fe85f5]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0a8ee7c5066296648]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-0cccc74a3e753aab]
aws_lb.target_group.target_elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/alb-tg/b274af2434b7989b]
aws_db_subnet_group.default: Refreshing state... [id=main]
aws_security_group.demoSG: Refreshing state... [id=sg-05a4f47623993df78]
aws_instance.demoInstance[0]: Refreshing state... [id=i-0083d0fd844903690]
aws_instance.demoInstance[0]: Refreshing state... [id=i-031548ee4097b39e1]
aws_route_table_association.rtl1: Refreshing state... [id=rtsassoc-08df25525596d7a2]
aws_route_table_association.rtl2: Refreshing state... [id=rtsassoc-04d867adc334bd5d3]
```

```
root@ip-172-31-43-211:~# Changes to Outputs:
+ lb_dns_name = "external-lb-996635588.us-east-1.elb.amazonaws.com"

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

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
[root@ip-172-31-43-211 ~]# terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-0894591850ccc1777]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-04be73b6d60d42898]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-06b37d2bb32cc1221]
aws_internet_gateway.demogateway: Refreshing state... [id=igw-09f16a122297c5ce2]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-009e4b21a392e7f731]
aws_security_group.demosg: Refreshing state... [id=sg-09be22a57a9c6ae25]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-0f989eb0157fe85fx5]
aws_subnet.application_subnet-1: Refreshing state... [id=subnet-0a8ee7c5062596848]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-0ccc74aa3a753aa8d]
aws_lb_target_group.target_elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/alb-tg/b274af2434b7989b]
aws_route_table.route: Refreshing state... [id=rtb-08ed8197bc041fb37]
aws_lb_listener_group.default: Refreshing state... [id=lb-1]
aws_db_instance.demoinstance: Refreshing state... [id=db-05aff478239934df78]
aws_instance.demoinstance[0]: Refreshing state... [id=i-0083d0fd844903690]
aws_lb_external_lb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:loadbalancer/app/external-lb/a56bc13a1991b156]
aws_instance.demoinstance[0]: Refreshing state... [id=i-031548ee4097b39e1]
aws_route_table_association.rtl: Refreshing state... [id=rthasssoc-08df255255096d7a2]
aws_route_table_association.rt2: Refreshing state... [id=rthasssoc-04d67adec34bd5d3]
aws_db_instance.default: Refreshing state... [id=db-IGPHK965QnP5KCFTJURMSQHY]
aws_lb_listener.external_elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:listener/app/external-lb/a56bc13a1991b156/4fb7c3da2cb23022]
aws_lb_target_group_attachment.attachment: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/alb-tg/b274af2434b7989b-20241120060201731800000002]
aws_lb_target_group_attachment.attachment2: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:637423550105:targetgroup/alb-tg/b274af2434b7989b-20241120060201533700000001]

Changes to Outputs:
+ lb_dns_name = "external-lb-996635588.us-east-1.elb.amazonaws.com"

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

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

Enter a value: yes

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

Outputs:

lb dns name = "external-lb-996635588.us-east-1.elb.amazonaws.com"
[root@ip-172-31-43-211 ~]# vim outputs.tf
[root@ip-172-31-43-211 ~]#
```