

TERRAFORM

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

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

The screenshot shows the AWS EC2 Management Console interface. On the left, there's a sidebar with navigation links like EC2 Dashboard, EC2 Global View, Events, Tags, Limits, and Instances (with sub-options for Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations). The main content area has tabs for EC2, RDS, Route 53, IAM, and VPC. The 'Instances' tab is selected, showing a list of 1/3 instances. One instance, 'demo vpc' (ID: i-0bcb6113d1506f2c2), is selected and shown in a detailed view. This view includes tabs for Details, Security, Networking, Storage, Status Checks, Monitoring, and Tags. Under 'Details', it shows the Instance ID (i-0bcb6113d1506f2c2), Instance state (Running), and various network and DNS details. The instance is running in the us-east-2a availability zone. At the bottom of the main content area, there are links for Feedback, Unified Settings, and a footer with copyright information and links to Privacy, Terms, and Cookie preferences.

And the instance is connected to terminal.

```
ec2-user@ip-172-31-9-28:~$ 
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\REALME> cd .\Downloads\
PS C:\Users\REALME\Downloads> ssh -i "red.pem" ec2-user@ec2-3-133-127-5.us-east-2.compute.amazonaws.com
The authenticity of host 'ec2-3-133-127-5.us-east-2.compute.amazonaws.com (3.133.127.5)' can't be established.
ED25519 key fingerprint is SHA256:MjUsyPrUnbRwG9LOEuPkwFnfv2Csahd6r0tjbP7wfw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-133-127-5.us-east-2.compute.amazonaws.com' (ED25519) to the list of known hosts.

--| --|_ ) _| ( _/_ / Amazon Linux 2 AMI
---|_\---|___|_

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-9-28 ~]$ 
[ec2-user@ip-172-31-9-28 ~]$ sudo yum -y install terraform
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No package terraform available.
Error: Nothing to do
[ec2-user@ip-172-31-9-28 ~]$ sudo yum install -y yum-utils shadow-utils
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Package yum-utils-1.1.31-46.amzn2.0.1.noarch already installed and latest version
Package 2:shadow-utils-4.1.5.1-24.amzn2.0.2.x86_64 already installed and latest version
Nothing to do
[ec2-user@ip-172-31-9-28 ~]$ 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
[ec2-user@ip-172-31-9-28 ~]$ 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 0:1.3.6-1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

30°C Partly sunny Q Search ENG IN 15:24 02-01-2023
```

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

sudo yum install -y yum-utils shadow-utils

sudo yum-config-manager --add-repo

<https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo>

Sudo yum –y install terraform

```
ec2-user@ip-172-31-9-28:~$ 
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\REALME> cd .\Downloads\
PS C:\Users\REALME\Downloads> ssh -i "red.pem" ec2-user@ec2-3-133-127-5.us-east-2.compute.amazonaws.com
The authenticity of host 'ec2-3-133-127-5.us-east-2.compute.amazonaws.com (3.133.127.5)' can't be established.
ED25519 key fingerprint is SHA256:MjUsyPrUnbRwG9LOEuPkwFnfv2Csahd6r0tjbP7wfw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-133-127-5.us-east-2.compute.amazonaws.com' (ED25519) to the list of known hosts.

--| --|_ ) _| ( _/_ / Amazon Linux 2 AMI
---|_\---|___|_

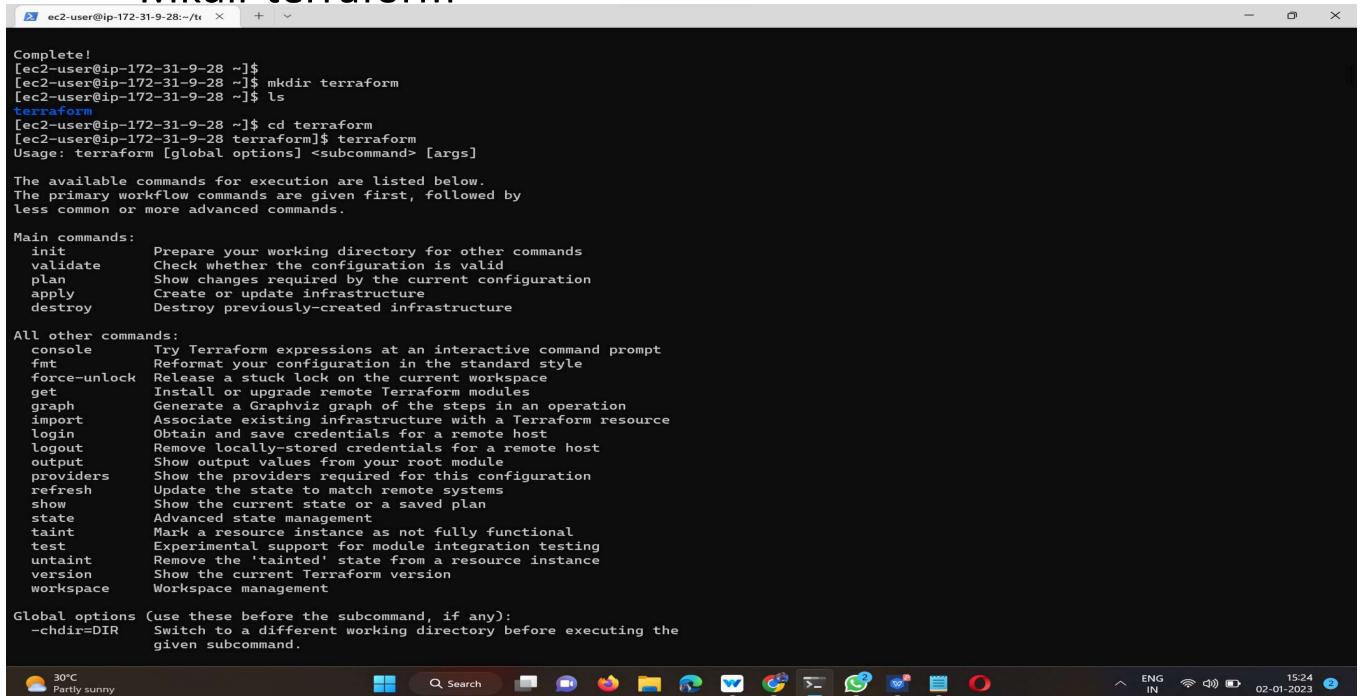
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-9-28 ~]$ 
[ec2-user@ip-172-31-9-28 ~]$ sudo yum -y install terraform
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No package terraform available.
Error: Nothing to do
[ec2-user@ip-172-31-9-28 ~]$ sudo yum install -y yum-utils shadow-utils
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Package yum-utils-1.1.31-46.amzn2.0.1.noarch already installed and latest version
Package 2:shadow-utils-4.1.5.1-24.amzn2.0.2.x86_64 already installed and latest version
Nothing to do
[ec2-user@ip-172-31-9-28 ~]$ 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
[ec2-user@ip-172-31-9-28 ~]$ 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 0:1.3.6-1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

30°C Partly sunny Q Search ENG IN 15:24 02-01-2023
```

And make a directory by using the command

Mkdir terraform



```
Complete!
[ec2-user@ip-172-31-9-28 ~]$ 
[ec2-user@ip-172-31-9-28 ~]$ mkdir terraform
[ec2-user@ip-172-31-9-28 ~]$ ls
terraform
[ec2-user@ip-172-31-9-28 ~]$ cd terraform
[ec2-user@ip-172-31-9-28 terraform]$ terraform
Usage: terraform [global options] <subcommand> [args]

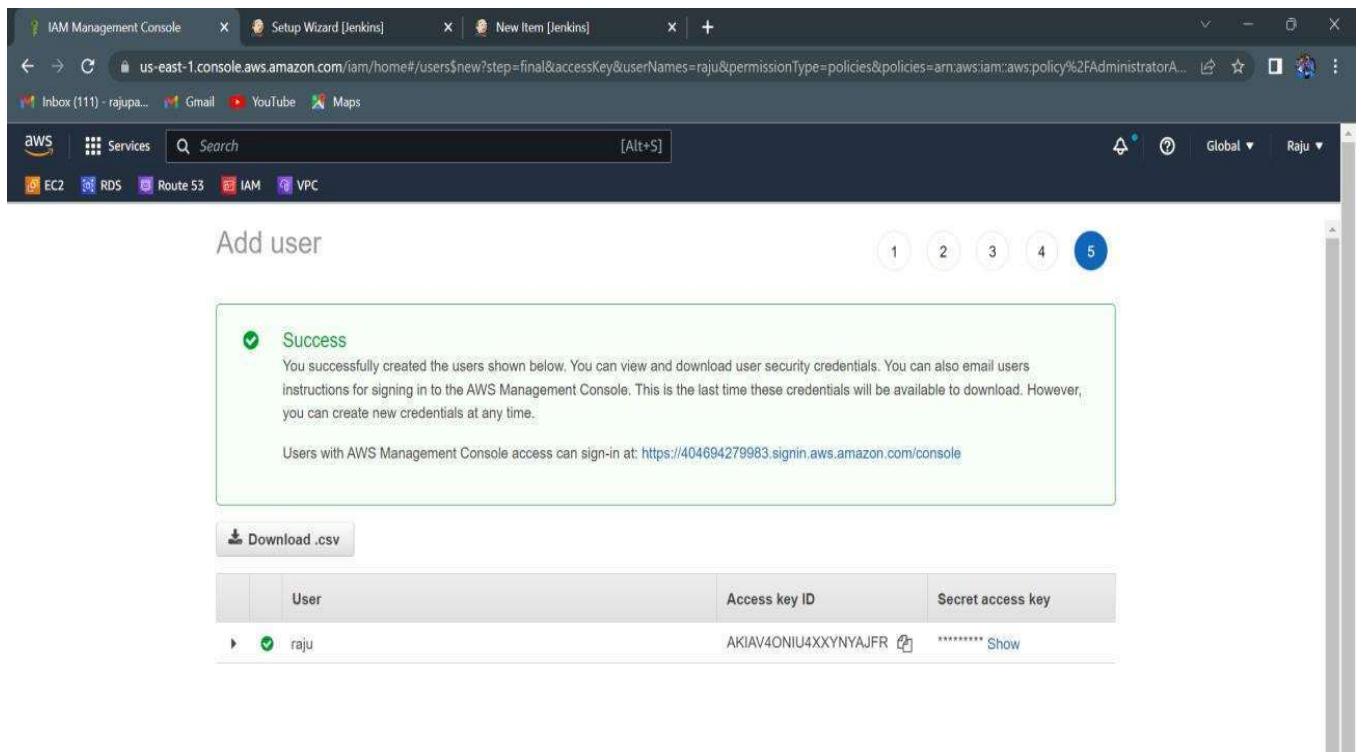
The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init      Prepare your working directory for other commands
  validate   Check whether the configuration is valid
  plan       Show changes required by the current configuration
  apply      Create or update infrastructure
  destroy    Destroy previously-created infrastructure

All other commands:
  console    Try Terraform expressions at an interactive command prompt
  fmt        Reformat your configuration in the standard style
  force-unlock Release a stuck lock on the current workspace
  get        Install or upgrade remote Terraform modules
  graph     Generate a Graphviz graph of the steps in an operation
  import    Associate existing infrastructure with a Terraform resource
  login     Obtain and save credentials for a remote host
  logout    Remove locally-stored credentials for a remote host
  output    Show output values from your root module
  providers Show the providers required for this configuration
  refresh   Update the state to match remote systems
  show      Show the current state or a saved plan
  state     Advanced state management
  taint     Mark a resource instance as not fully functional
  test      Experimental support for module integration testing
  untaint   Remove the 'tainted' state from a resource instance
  version   Show the current Terraform version
  workspace Workspace management

Global options (use these before the subcommand, if any):
  -chdir=DIR  Switch to a different working directory before executing the
              given subcommand.
```

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



The screenshot shows the AWS IAM Management Console interface. A new user named 'raju' has been successfully created. The success message indicates that the user can now view and download user security credentials, and provides a link for signing in to the AWS Management Console.

Add user

Success

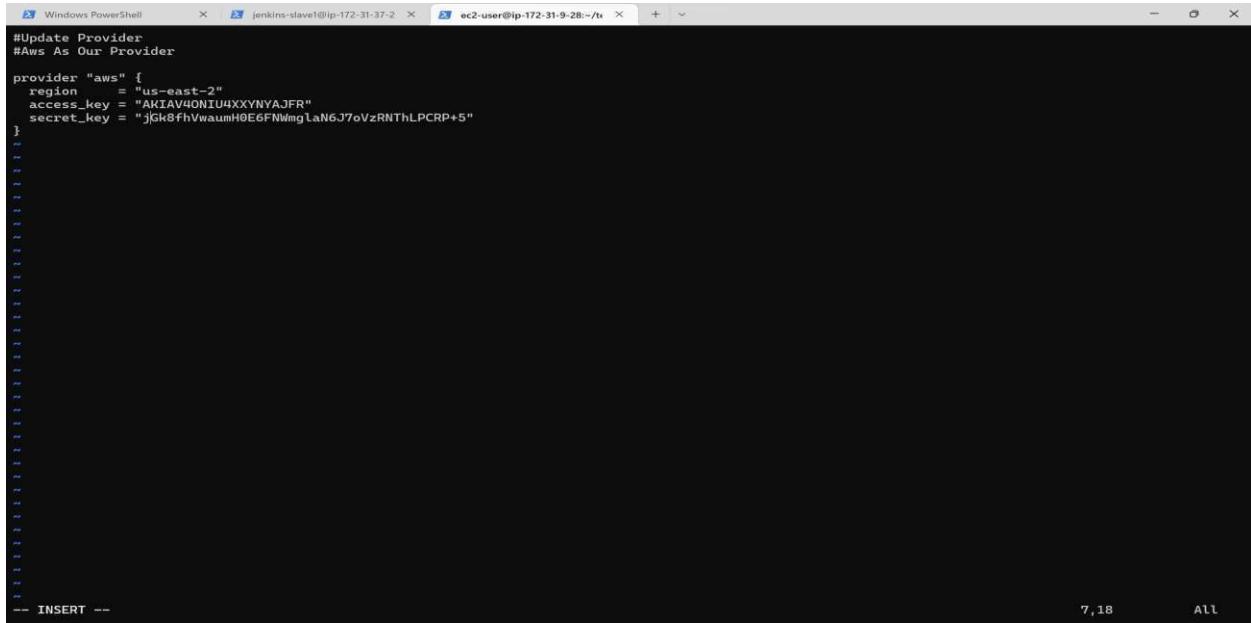
You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: <https://404694279983.signin.aws.amazon.com/console>

User

User	Access key ID	Secret access key
raju	AKIAV4ONIU4XXYNAYJFR	***** Show

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

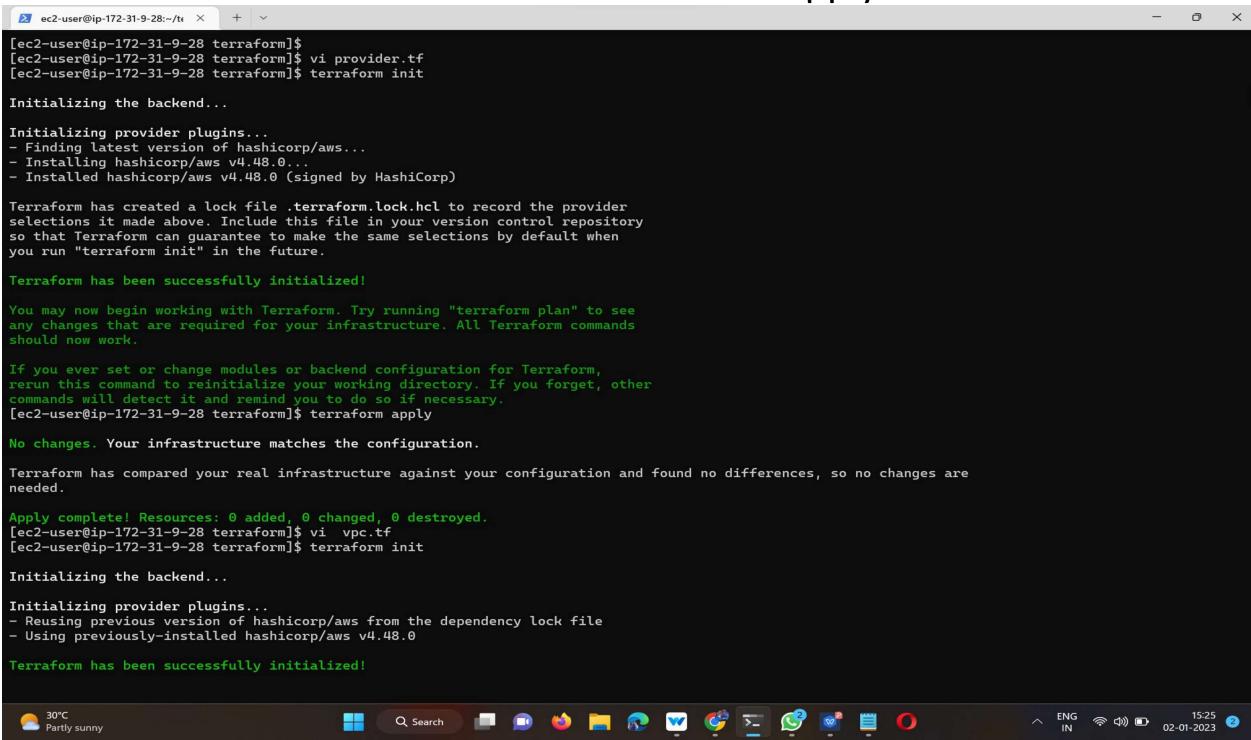


```
#Update Provider
#Aws As Our Provider

provider "aws" {
    region      = "us-east-2"
    access_key  = "AKIAV4ONIU4XXYNYAJFR"
    secret_key  = "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"
}

-- INSERT --
```

Then initialize the terraform and terraform apply



```
[ec2-user@ip-172-31-9-28:~/.tf] $ vi provider.tf
[ec2-user@ip-172-31-9-28:~/.tf] $ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.48.0...
- Installed hashicorp/aws v4.48.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[ec2-user@ip-172-31-9-28:~/.tf] $ terraform apply

No changes. Your infrastructure matches the configuration.

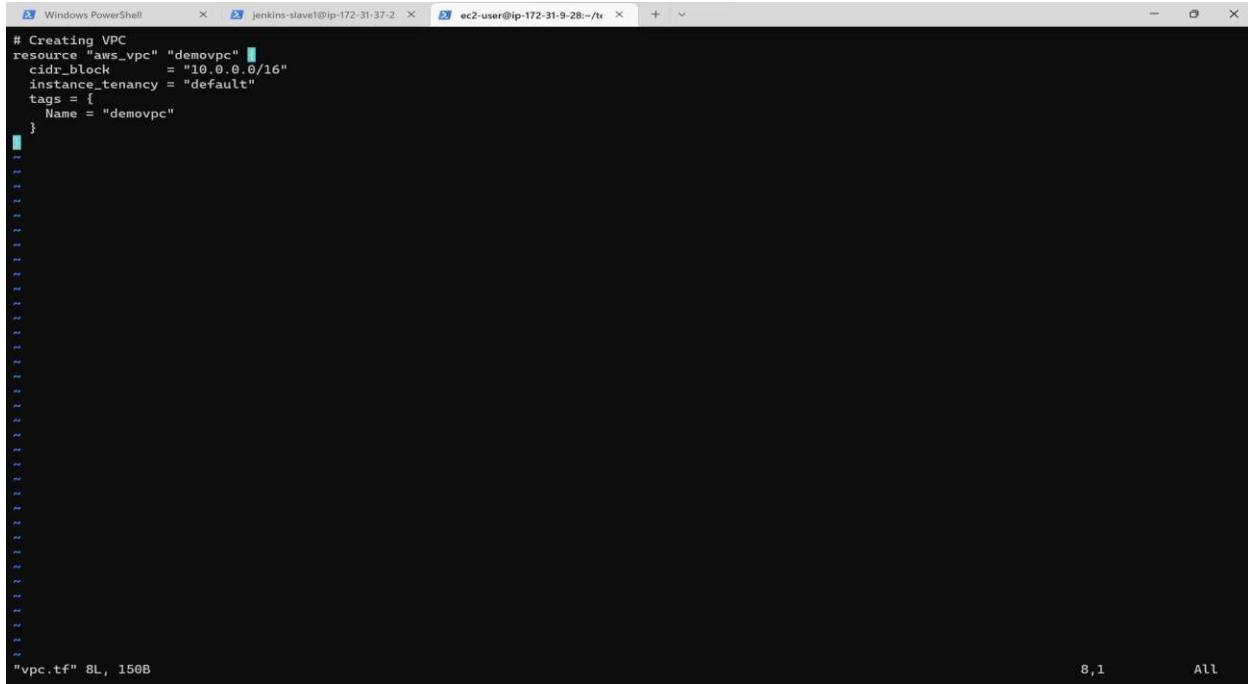
Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are
needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-9-28:~/.tf] $ vi vpc.tf
[ec2-user@ip-172-31-9-28:~/.tf] $ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

Terraform has been successfully initialized!
```

Create vpc.tf file by using the command

Vi vpc.tf

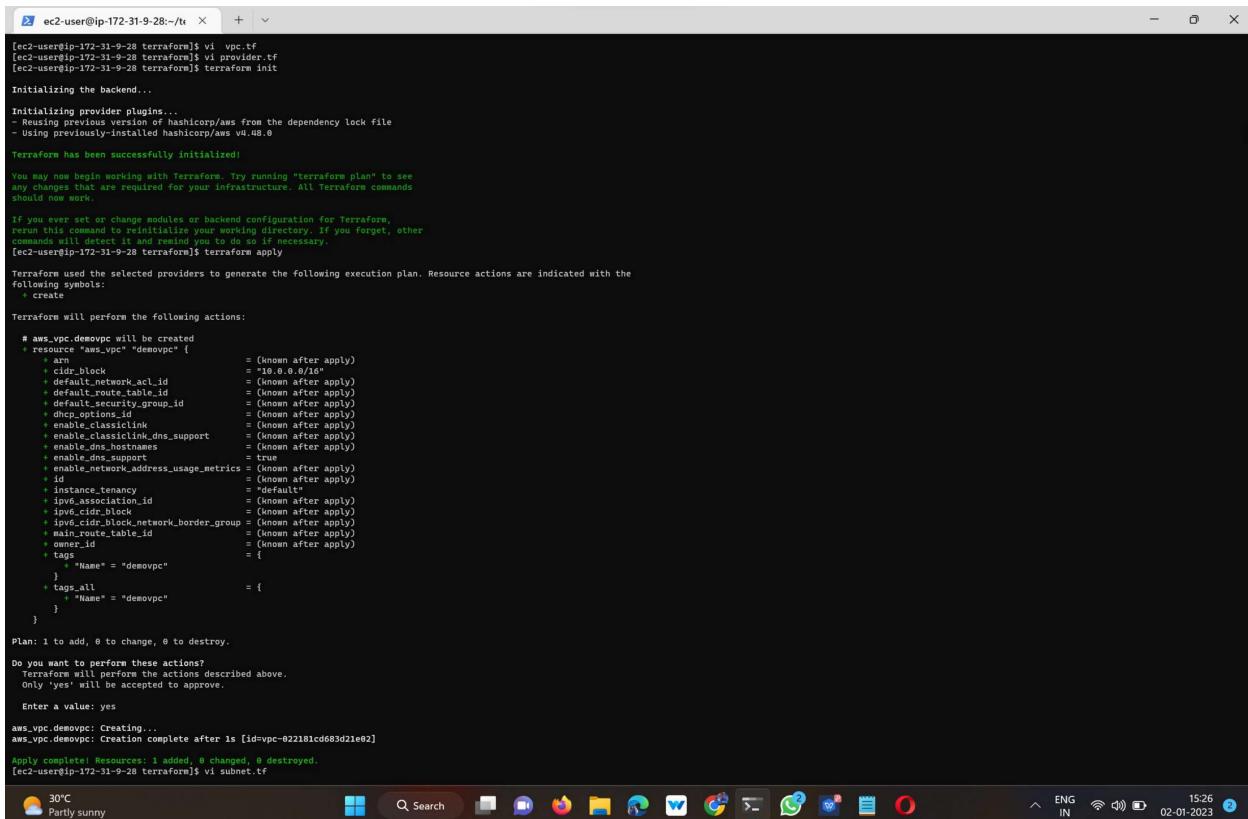


```
# Creating VPC
resource "aws_vpc" "demovpc"
  cidr_block      = "10.0.0.0/16"
  instance_tenancy = "default"
  tags = [
    Name = "demovpc"
  ]
```

"vpc.tf" 8L, 150B

Save the script and init the terraform-terraform init

And terraform apply



```
[ec2-user@ip-172-31-9-28:~]$ vi vpc.tf
[ec2-user@ip-172-31-9-28 terraform]$ vi provider.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
return to this directory and run "terraform init" to reinitialize your working
directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[ec2-user@ip-172-31-9-28 terraform]$ terraform apply

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

Terraform will perform the following actions:

  # aws_vpc.demovpc will be created
  + resource "aws_vpc" "demovpc" {
      + arn                                = (known after apply)
      + cidr_block                         = "10.0.0.0/16"
      + default_network_acl_id            = (known after apply)
      + default_route_table_id           = (known after apply)
      + default_security_group_id        = (known after apply)
      + enable_classiclink                = (known after apply)
      + enable_classiclink_dns_support   = (known after apply)
      + enable_dns_hostname               = (known after apply)
      + enable_dns_support                = true
      + enable_ip_transit               = (known after apply)
      + enable_network_address_usage_metrics = (known after apply)
      + id                                 = (known after apply)
      + instance_tenancy                  = "default"
      + ipv6_association_id              = (known after apply)
      + ipv6_cidr_block                  = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id              = (known after apply)
      + owner_id                           = (known after apply)
      + tags
          + "Name" = "demovpc"
      + tags_all                          = {
          + "Name" = "demovpc"
        }
    }

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

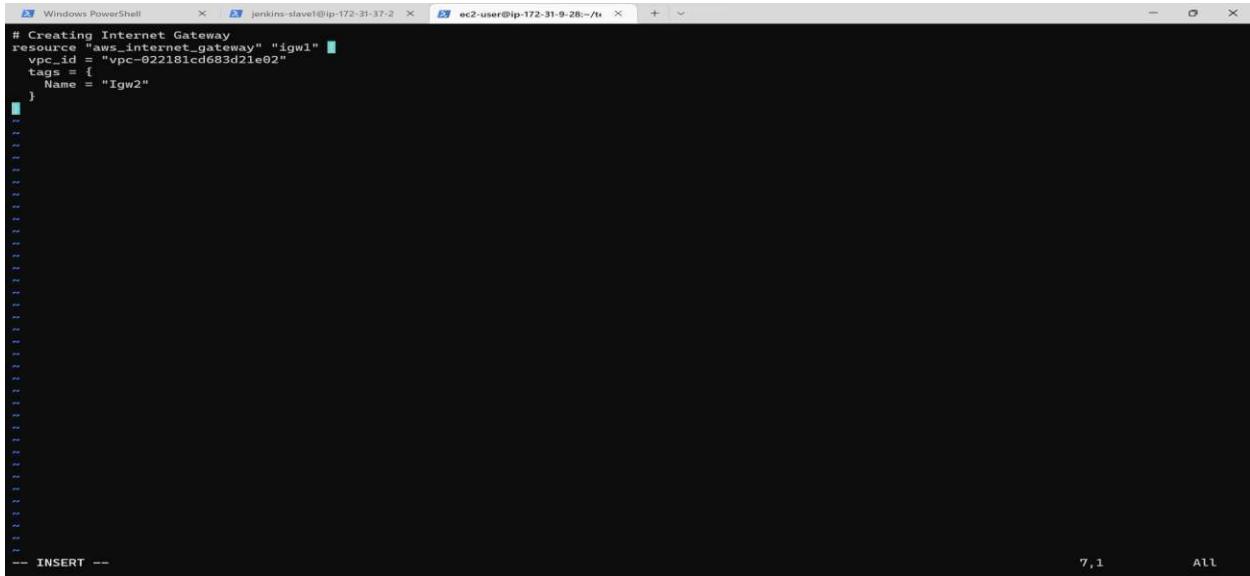
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_vpc.demovpc: Creating...
aws_vpc.demovpc: Creation complete after 1s [id=vpc-022181cd683d21e0]

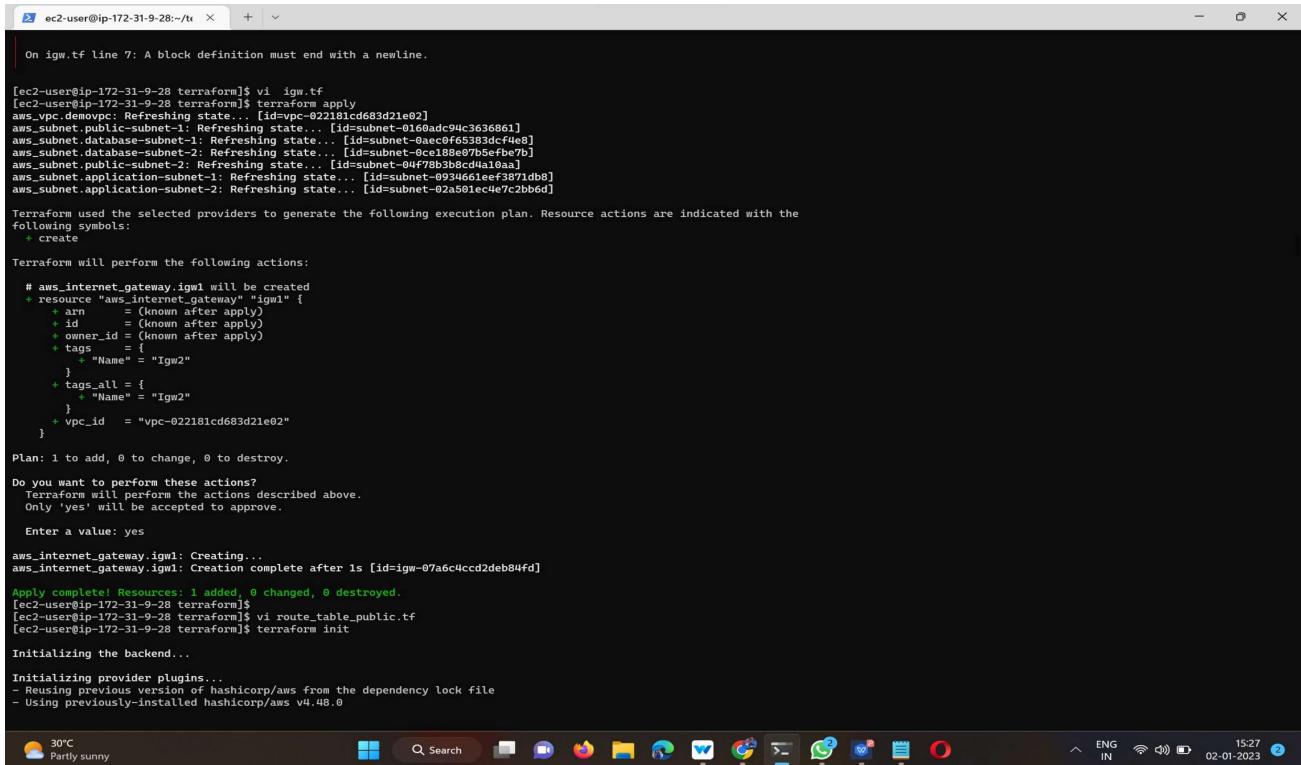
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-9-28 terraform]$ vi subnet.tf
```

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



```
# Creating Internet Gateway
resource "aws_internet_gateway" "igw1" {
  vpc_id = "vpc-022181cd683d21e02"
  tags = [
    { Name = "Igw2" }
  ]
}
```

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



```
[ec2-user@ip-172-31-9-28 ~]$ vi igw.tf
[ec2-user@ip-172-31-9-28 ~]$ terraform init
[ec2-user@ip-172-31-9-28 ~]$ terraform plan
On igw.tf line 7: A block definition must end with a newline.

[ec2-user@ip-172-31-9-28 ~]$ terraform apply
aws_vpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_subnet_public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_subnet_database-subnet-1: Refreshing state... [id=subnet-0aec0f65383dcf4e8]
aws_subnet_database-subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_subnet_public-subnet-2: Refreshing state... [id=subnet-04f78b3bbcd4a10aa]
aws_subnet_application-subnet-1: Refreshing state... [id=subnet-0934661eef3871dbd]
aws_subnet_application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]

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

Terraform will perform the following actions:

# aws_internet_gateway.igw1 will be created
+ resource "aws_internet_gateway" "igw1" {
  + arn      = (known after apply)
  + id       = (known after apply)
  + owner_id = (known after apply)
  + tags     = {
      + "Name" = "Igw2"
    }
  + tags_all = {
      + "Name" = "Igw2"
    }
  + vpc_id   = "vpc-022181cd683d21e02"
}

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

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

Enter a value: yes

aws_internet_gateway.igw1: Creating...
aws_internet_gateway.igw1: Creation complete after 1s [id=igw-07a6c4ccd2deb84fd]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-9-28 ~]$ terraform init
[ec2-user@ip-172-31-9-28 ~]$ terraform init

Initializing the backend...

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

Create a subnet-vi subnet.tf

```

Windows PowerShell Jenkins Slave1@ip-172-31-37-2 ec2-user@ip-172-31-9-28 ~ /t
# Creating 1st web subnet
resource "aws_subnet" "public-subnet-1" {
  vpc_id          = "vpc-0d81b762754d69a17"
  cidr_block     = "10.0.0.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-2a"
  tags = [
    { Name = "Web Subnet 1" }
  ]
}
# Creating 2nd web subnet
resource "aws_subnet" "public-subnet-2" {
  vpc_id          = aws_vpc.demo.vpc.id
  cidr_block     = "10.0.1.0/24"
  map_public_ip_on_launch = true
  availability_zone = "us-east-2b"
  tags = [
    { Name = "Web Subnet 2" }
  ]
}
# Creating 1st application subnet
resource "aws_subnet" "application-subnet-1" {
  vpc_id          = aws_vpc.demo.vpc.id
  cidr_block     = "10.0.3.0/24"
  map_public_ip_on_launch = false
  availability_zone = "us-east-2a"
  tags = [
    { Name = "Application Subnet 1" }
  ]
}
# Creating 2nd application subnet
resource "aws_subnet" "application-subnet-2" {
  vpc_id          = aws_vpc.demo.vpc.id
  cidr_block     = "10.0.4.0/24"
  map_public_ip_on_launch = false
  availability_zone = "us-east-2b"
  tags = [
    { Name = "Application Subnet 2" }
  ]
}
# Create Database Private Subnet
resource "aws_subnet" "database-subnet-1" {
  vpc_id          = aws_vpc.demo.vpc.id
  cidr_block     = "10.0.5.0/24"
  availability_zone = "us-east-2a"
  tags = [
    { Name = "Database Subnet 1" }
  ]
}
# Create Database Private Subnet
resource "aws_subnet" "database-subnet-2" {
  vpc_id          = aws_vpc.demo.vpc.id
  cidr_block     = "10.0.6.0/24"
  availability_zone = "us-east-2b"
  tags = [
    { Name = "Database Subnet 2" }
  ]
}
-- INSERT --

```

Terraform init and terraform apply subnets can be created

```

ec2-user@ip-172-31-9-28:~/t ~ + ~
[ec2-user@ip-172-31-9-28 terraform]$ vi subnet.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and require you to run this command as necessary.

[ec2-user@ip-172-31-9-28 terraform]$ terraform apply
aws_vpc.demo.vpc: Refreshing state... [id=vpc-022181cc683d21e02]

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

Terraform will perform the following actions:

# aws_subnet.application-subnet-1 will be created
resource "aws_subnet" "application-subnet-1" {
  # ...
  + assign_ipv6_address_on_creation      = (known after apply)
  + availability_zone                    = false
  + availability_zone_id                = "us-east-2a"
  + (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_aaa_record_on_launch = false
  + id                                  = (known after apply)
  + ipv6_cidr_block_association_id       = (known after apply)
  + ipv6_native                          = true
  + map_public_ip_on_launch              = (known after apply)
  + owner_id                            = (known after apply)
  + private_dns_hostname_type_on_launch = (known after apply)
  + tags                                = {
      + "Name" = "Application Subnet 1"
    }
  + tags_all                            = {
      + "Name" = "Application Subnet 1"
    }
  + vpc_id                             = "vpc-022181cc683d21e02"
}

# aws_subnet.application-subnet-2 will be created
resource "aws_subnet" "application-subnet-2" {
  # ...
  + assign_ipv6_address_on_creation      = (known after apply)
  + availability_zone                    = false
  + availability_zone_id                = "us-east-2b"
  + (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_aaa_record_on_launch = false
  + id                                  = (known after apply)
  + ipv6_cidr_block_association_id       = (known after apply)
  + ipv6_native                          = false
  + map_public_ip_on_launch              = (known after apply)
  + owner_id                            = (known after apply)
  + private_dns_hostname_type_on_launch = (known after apply)
}

30°C
Partly sunny

```

```

[ec2-user@ip-172-31-9-28 ~] terraform> + 
[ec2-user@ip-172-31-9-28 ~] terraform> terraform apply
aws_vpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_subnet_application_subnet-2: Refreshing state... [id=subnet-02a501cc67c2bb6d]
aws_subnet_application_subnet-1: Refreshing state... [id=subnet-0920661ef3871db8]
aws_subnet_database_subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_subnet_database_subnet-1: Refreshing state... [id=subnet-aac0f6538dcf4e8]
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
+ create

Terraform will perform the following actions:

# aws_subnet.public-subnet-1 will be created
+ resource "aws_subnet" "public-subnet-1" {
    arn                                = (known after apply)
    association_ipv6_address_on_creation = #<resource>
    availability_zone                   = "us-east-2a"
    availability_zone_id               = (known after apply)
    cidr_block                         = "10.0.0.0/24"
    enable_dhcp                         = true
    enable_ip_forwarding                = false
    enable_resource_name_dns_a_record_on_launch = false
    enable_resource_name_dns_aaa_record_on_launch = false
    ipv4                               = (known after apply)
    ipv6                               = (known after apply)
    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-022181cd683d21e02"

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

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

Enter a value: yes
aws_subnet.public-subnet-1: Creating...
aws_subnet.public-subnet-1: Still creating... [10s elapsed]
aws_subnet.public-subnet-1: Creation complete after 10s [id=subnet-0168adcf94c3636861]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-9-28 ~] terraform> vi igw.tf
[ec2-user@ip-172-31-9-28 ~] terraform> terraform init
[ec2-user@ip-172-31-9-28 ~] terraform> terraform init
There are some problems with the configuration, described below.

The Terraform configuration must be valid before initialization so that
Terraform can determine which modules and providers need to be installed.

Error: Missing newline after block definition
  On igw.tf line 7: A block definition must end with a newline.

[ec2-user@ip-172-31-9-28 ~] terraform> vi igw.tf

```

Create a route table with the name

Vi route_public_table.tf

```
[ec2-user@ip-172-31-9-28:~/t] + ~
[ec2-user@ip-172-31-9-28 terraform]$ vi route_table_public.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

Terraform has been successfully initialized!
```

```
Windows PowerShell      Jenkins-slave1@ip-172-31-37-2      ec2-user@ip-172-31-9-28:~/t] + ~
# Creating Route Table
resource "aws_route_table" "route" {
  vpc_id = aws_vpc.demoVPC.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw1.id
  }
  tags = [
    { Name = "Route to internet" }
  ]
}
# Associating Route Table
resource "aws_route_table_association" "rt1" {
  subnet_id     = aws_subnet.public-subnet-1.id
  route_table_id = aws_route_table.route.id
}
# Associating Route Table
resource "aws_route_table_association" "rt2" {
  subnet_id     = aws_subnet.public-subnet-2.id
  route_table_id = aws_route_table.route.id
}

-- INSERT --
```

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

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

```

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

Initializing the backend...

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

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[ec2-user@ip-172-31-9-28 ~]$ terraform apply
aws_internet_gateway.igw1: Refreshing state... [id=igw-07a6c4cccd2deb84fd]
aws_vpc.demovpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0934661eeff3871db8]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04f78b3b8cd4a10aa]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0aec0f65383dcf4e8]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]

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

Terraform will perform the following actions:

# aws_route_table.route will be created
+ resource "aws_route_table" "route" {
    + arn          = (known after apply)
    + id           = (known after apply)
    + owner_id     = (known after apply)
    + propagating_vgwss = (known after apply)
    + route        = [
        + {
            + carrier_gateway_id      = ""
            + cidr_block              = "0.0.0.0/0"
            + core_network_arn        = ""
            + destination_prefix_list_id = ""
            + egress_only_gateway_id   = "igw-07a6c4cccd2deb84fd"
            + gateway_id               = ""
            + instance_id              = ""
            + ipv6_cidr_block          = ""
            + local_gateway_id         = ""
            + nat_gateway_id           = ""
            + network_interface_id     = ""
            + transit_gateway_id       = ""
            + vpc_endpoint_id          = ""
            + vpc_peering_connection_id = ""
        },
    ],
    + tags          = {
        + "Name" = "Route to internet"
    }
    + tags_all      = {
        + "Name" = "Route to internet"
    }
    + vpc_id        = "vpc-022181cd683d21e02"
}

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

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

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

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

Enter a value: yes

aws_route_table.route: Creating...
aws_route_table.route: Creation complete after 0s [id=rtb-088e69f09ebaa0f31]
aws_route_table_association.rt1: Creating...
aws_route_table_association.rt2: Creating...
aws_route_table_association.rt2: Creation complete after 0s [id=rtbasssoc-0a7wb3bf113be85c3]
aws_route_table_association.rt1: Creation complete after 0s [id=rtbasssoc-027b6579877ele560]

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

```

30°C
Party sunny
Search
ENG IN 15:27 02-01-2023 ②

[ec2-user@ip-172-31-9-28 ~]$ terraform apply
aws_route_table_association.rt1: Refreshing state... [id=rtbasssoc-0a7wb3bf113be85c3]
aws_route_table_association.rt2: Refreshing state... [id=rtbasssoc-027b6579877ele560]

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

Terraform will perform the following actions:

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

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

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

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

Enter a value: yes

aws_route_table_association.rt1: Creating...
aws_route_table_association.rt1: Creation complete after 0s [id=rtbasssoc-0a7wb3bf113be85c3]
aws_route_table_association.rt2: Creating...
aws_route_table_association.rt2: Creation complete after 0s [id=rtbasssoc-027b6579877ele560]

Apply complete! Resources: 2 updated, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-9-28 ~]$ terraform
[ec2-user@ip-172-31-9-28 ~]$
```

Create a security groups –vi web_sg.tf

```

Windows PowerShell      Jenkins-slave1@ip-172-31-37-2      ec2-user@ip-172-31-9-28:~/tf
# Creating Security Group
resource "aws_security_group" "demosg" {
  vpc_id = aws_vpc.demoVPC.id
  # Inbound Rules
  # HTTP access from anywhere
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  # HTTPS access from anywhere
  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  # SSH access from anywhere
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  # Outbound Rules
  # Internet access to anywhere
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "Web SG"
  }
}
```
-- INSERT --

```

## Terraform init and apply the terraform

```

[ec2-user@ip-172-31-9-28:~/tf]$
[ec2-user@ip-172-31-9-28 terraform]$ vi web_sg.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[ec2-user@ip-172-31-9-28 terraform]$ terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-31-9-28 terraform]$ terraform apply
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_vpc.demoVPC: Refreshing state... [id=vpc-022181cd683d21e02]
aws_internet_gateway.igw1: Refreshing state... [id=igw-07a6c4cc2deb84fd]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04f78b3b8cd4a10aa]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0ce188e7b5efbe7b]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0934661ef3871d08]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0ae0f653b3dcf4e8]
aws_route_table.route: Refreshing state... [id=rtb-088e69f99ebaaf031]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-0a74b3bf13bbe85c3]
aws_route_table_association.rt1: Refreshing state... [id=rtbassoc-027b6579877ele560]

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",
]
 + description = ""
 + from_port = 0
 + ipv6_cidr_blocks = []
 + prefix_list_ids = []
 + protocol = "-1"
 + security_groups = []
 }
]
}
```

30°C  
Partly sunny



ENG IN 15:28 02-01-2023

```
ec2-user@ip-172-31-9-28:~/tr ~ + -
```

```
+ to_port = 22
},
+ {
+ cidr_blocks = [
+ "+ 0.0.0.0/0",
+]
+ description = ""
+ from_port = 443
+ ipv6_cidr_blocks = []
+ prefix_list_ids = []
+ protocol = "tcp"
+ security_groups = []
+ self = false
+ to_port = 443
},
+ {
+ cidr_blocks = [
+ "+ 0.0.0.0/0",
+]
+ description = ""
+ from_port = 80
+ ipv6_cidr_blocks = []
+ prefix_list_ids = []
+ protocol = "tcp"
+ security_groups = []
+ self = false
+ to_port = 80
},
]
name = (known after apply)
name_prefix = (known after apply)
owner_id = (known after apply)
revoke_rules_on_delete = false
tags = {
 + "Name" = "Web SG"
}
tags_all = {
 + "Name" = "Web SG"
}
vpc_id = "vpc-022181cd683d21e02"
}

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-02ba909d05b7cf2f0]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-9-28:~/tr]$
[ec2-user@ip-172-31-9-28 terraform]$
```

And again create a security group vi database\_sg.tf

```
Create Database Security Group
resource "aws_security_group" "database-sg" {
 name = "Database SG"
 description = "Allow inbound traffic from application layer"
 vpc_id = aws_vpc.deiovpc.id
 ingress {
 description = "Allow traffic from application layer"
 from_port = 3306
 to_port = 3306
 protocol = "tcp"
 security_groups = [aws_security_group.demossg.id]
 }
 egress {
 from_port = 32768
 to_port = 65535
 protocol = "tcp"
 cidr_blocks = ["0.0.0.0/0"]
 }
 tags = {
 Name = "Database SG"
 }
}
```

## Terraform init and apply

```

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

Initializing the backend...

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

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[ec2-user@ip-172-31-9-28 terraform]$ terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_internet_gateway.igwl: Refreshing state... [id=igw-07a6c4cccd2eb84fd]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0aec0f65383dcf4e8]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04f78b3b8cd4a10aa]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0934661ef3871db8]
aws_security_group.demosg: Refreshing state... [id=sg-02ba999d05b7cf2f0]
aws_route_table.route: Refreshing state... [id=rtb-088e69ff09ebaa0f31]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_instance.demoinstance: Refreshing state... [id=i-04f152afc678fb0ad]
aws_route_table_association.rtl1: Refreshing state... [id=rtbasoc-027b6579877ele560]
aws_instance.demoinstance1: Refreshing state... [id=i-0fecb1145449837b9]
aws_route_table_association.rt2: Refreshing state... [id=rtbasoc-0a74b3bf113be85c3]

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

Terraform will perform the following actions:

aws_security_group.database_sg will be created
+ resource "aws_security_group" "database_sg" {
 + arn = (Known after apply)
 + description = "Allow inbound traffic from application layer"
 + egress = [
 +
],
 + 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-02ba999d05b7cf2f0",
],
 + 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-022181cd683d21e02"
}

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-0b4af5731c7dd254d]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-9-28 terraform]$
[ec2-user@ip-172-31-9-28 terraform]$

28°C
Partly sunny
ENG IN 15:32 02-01-2023
```

```

[ec2-user@ip-172-31-9-28:~]$ terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_internet_gateway.igwl: Refreshing state... [id=igw-07a6c4cccd2eb84fd]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0aec0f65383dcf4e8]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04f78b3b8cd4a10aa]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0934661ef3871db8]
aws_security_group.demosg: Refreshing state... [id=sg-02ba999d05b7cf2f0]
aws_route_table.route: Refreshing state... [id=rtb-088e69ff09ebaa0f31]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_instance.demoinstance: Refreshing state... [id=i-04f152afc678fb0ad]
aws_route_table_association.rtl1: Refreshing state... [id=rtbasoc-027b6579877ele560]
aws_instance.demoinstance1: Refreshing state... [id=i-0fecb1145449837b9]
aws_route_table_association.rt2: Refreshing state... [id=rtbasoc-0a74b3bf113be85c3]

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-0b4af5731c7dd254d]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-9-28 terraform]$
[ec2-user@ip-172-31-9-28 terraform]$

28°C
Partly sunny
ENG IN 15:32 02-01-2023
```

Create data.sh file-vi data.sh

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

```
[ec2-user@ip-172-31-9-28 ~]$ terraform init
[ec2-user@ip-172-31-9-28 ~]$ vi data.sh
[ec2-user@ip-172-31-9-28 ~]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will remind you to do so if necessary.
[ec2-user@ip-172-31-9-28 ~]$ terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-31-9-28 ~]$ terraform apply
aws_vpc_demoVpc: Refreshing state... [id=vpc-022181cd6d83d21e82]
aws_internet_gateway_igw1: Refreshing state... [id=igw-07ad4cccd2deb84f]
aws_subnet_privateSubnet1: Refreshing state... [id=subnet-02a301c4e7cbb6d]
aws_subnet_applicationSubnet1: Refreshing state... [id=subnet-02a301c4e7cbb6d]
aws_subnet_privateSubnet1: Refreshing state... [id=subnet-0aec0f65383dcf4e8]
aws_subnet_applicationSubnet1: Refreshing state... [id=subnet-0934661eeef3871db8]
aws_route_table_route1: Refreshing state... [id=rtaassoc-027b6579877e1e560]
aws_route_table_route2: Refreshing state... [id=rtaassoc-027b6579877e1e560]
aws_subnet_publicSubnet2: Refreshing state... [id=subnet-04f78b3b8cd4a16aa]
aws_route_table_association_r1: Refreshing state... [id=rtaassoc-027b6579877e1e560]
aws_route_table_association_r2: Refreshing state... [id=rtaassoc-027b6579877e1e560]

No changes. Your infrastructure matches the configuration.

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

apply complete! Resources: 0 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-9-28 ~]$ vi web.ngn
[ec2-user@ip-172-31-9-28 ~]$ terraform init
Initialization the backend...

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

Terraform has been successfully initialized!
```

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

```
[ec2-user@ip-172-31-9-28:~/tr] + -
[ec2-user@ip-172-31-9-28 terraform]$ [ec2-user@ip-172-31-9-28 terraform]$ vi ec2.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.48.0
Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[ec2-user@ip-172-31-9-28 terraform]$ terraform validate
Success! The configuration is valid.

[ec2-user@ip-172-31-9-28 terraform]$ terraform apply
aws_internet_gateway.igw1: Refreshing state... [id=igw-07a6c4cccd2deb84fd]
aws_vpc.demovpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0934661eeef3871db8]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-04f5ec18e07b5eefbe7b]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04fb59440410aa]
aws_route_table.route: Refreshing state... [id=rtb-088e69f09ebaa0f31]
aws_security_group.demosg: Refreshing state... [id=sg-02ba999d05b7cf2f0]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0ae0f65383dcf4e8]
aws_route_table_association.rtl: Refreshing state... [id=rtbassoc-027b6579877e1e560]
aws_route_table_association.rtt2: Refreshing state... [id=rtbassoc-0a74b3bf113be85c3]

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

Terraform will perform the following actions:

aws_instance.demoinstance will be created
+ resource "aws_instance" "demoinstance" {
 ami = "ami-0a606d8395a538502"
 ami = (known after apply)
 arn = true
 associate_public_ip_address = true
 availability_zone = (known after apply)
 cpu_threads_per_core = (known after apply)
 disable_api_stop = (known after apply)
 disable_api_termination = (known after apply)
 ebs_optimized = (known after apply)
 get_password_data = false
 host_id = (known after apply)
 host_resource_group_arn = (known after apply)
}
```

30°C Partly sunny  ENG IN 15:28 02-01-2023

```
Creating 1st EC2 instance in Public Subnet
resource "aws_instance" "demoinstance" {
 ami = "ami-0a606d8395a538502"
 instance_type = "t2.micro"
 # count = 1
 key_name = "first"
 vpc_security_group_ids = ["${aws_security_group.demosg.id}"]
 subnet_id = aws_subnet.public-subnet-1.id
 associate_public_ip_address = true
 user_data = file("data.sh")
 tags = {
 Name = "My Public Instance"
 }
}
Creating 2nd EC2 instance in Public Subnet
resource "aws_instance" "demoinstance1" {
 ami = "ami-0a606d8395a538502"
 instance_type = "t2.micro"
 # count = 1
 key_name = "first"
 vpc_security_group_ids = ["${aws_security_group.demosg.id}"]
 subnet_id = aws_subnet.public-subnet-2.id
 associate_public_ip_address = true
 user_data = file("data.sh")
 tags = {
 Name = "My Public Instance 2"
 }
}
~
```

```
ec2-user@ip-172-31-9-28:~/tr ~ + -
```

```
+ delete_on_termination = (known after apply)
+ device_index = (known after apply)
+ network_card_index = (known after apply)
+ network_interface_id = (known after apply)
}

+ private_dns_name_options {
+ enable_resource_name_dns_a_record = (known after apply)
+ enable_resource_name_dns_aaaa_record = (known after apply)
+ hostname_type = (known after apply)
}

+ root_block_device {
+ delete_on_termination = (known after apply)
+ device_name = (known after apply)
+ encrypted = (known after apply)
+ iops = (known after apply)
+ kms_key_id = (known after apply)
+ tags = (known after apply)
+ throughput = (known after apply)
+ volume_id = (known after apply)
+ volume_size = (known after apply)
+ volume_type = (known after apply)
}

}

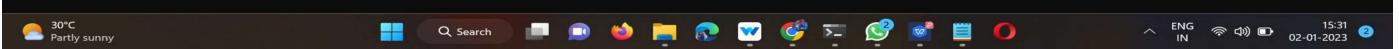
Plan: 2 to add, 0 to change, 0 to destroy.

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

Enter a value: yes

aws_instance.demoinstance: Creating...
aws_instance.demoinstance1: Creating...
aws_instance.demoinstance: Still creating... [10s elapsed]
aws_instance.demoinstance1: Still creating... [10s elapsed]
aws_instance.demoinstance: Still creating... [20s elapsed]
aws_instance.demoinstance1: Still creating... [20s elapsed]
aws_instance.demoinstance: Still creating... [30s elapsed]
aws_instance.demoinstance1: Still creating... [30s elapsed]
aws_instance.demoinstance: Creation complete after 31s [id=i-04f152afcd678f0ad]
aws_instance.demoinstance1: Creation complete after 31s [id=i-0fecb1145449837b9]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-9-28 terraform]$
```



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

The above load balancer is of type external. Load balancer type is set to application. The aws\_lb\_target\_group\_attachment resource will attach our instances to the Target Group. The load balancer will listen requests on port 80

Vi alb.tf

```
[ec2-user@ip-172-31-9-28:~/tr ~ + ^] [ec2-user@ip-172-31-9-28 terraform]$ vi alb.tf [ec2-user@ip-172-31-9-28 terraform]$ terraform init

Initializing the backend...

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

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[ec2-user@ip-172-31-9-28 terraform]$ terraform apply
aws_vpc.demoVPC: Refreshing state... [id=vpc-022181cd683d21e02]
aws_subnet_public-subnet-1: Refreshing state... [id=subnet-0160adcd94c3636861]
aws_internet_gateway.igw1: Refreshing state... [id=igw-07a6c4cccd2deb84fd]
aws_subnet_public-subnet-2: Refreshing state... [id=subnet-04f78b3b8cd4a10aa]
aws_route_table.route: Refreshing state... [id=rtb-088e69f09ebaaf31]
aws_security_group.demosg: Refreshing state... [id=sg-02ba909d05b7cf2f0]
aws_subnet_application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]
aws_subnet_application-subnet-1: Refreshing state... [id=subnet-0934661eeef3871db8]
aws_subnet_database-subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_subnet_database-subnet-1: Refreshing state... [id=subnet-0aec0f65383dcf4le8]
aws_route_table_association.rtl: Refreshing state... [id=rtbassoc-027b6579877e1e560]
aws_security_group.database-sg: Refreshing state... [id=sg-0b4af5731c7dd254d]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-0a74b3bf113be85c3]
aws_instance.demoinstance1: Refreshing state... [id=i-0fecb11454u4983tb9]
aws_instance.demoinstance: Refreshing state... [id=i-0uf152afcd678f0ad]

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

Terraform will perform the following actions:

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

```
Creating External LoadBalancer
resource "aws_lb" "external-alb" {
 name = "External-LB"
 internal = false
 load_balancer_type = "application"
 security_groups = [aws_security_group.demosg.id]
 subnets = [aws_subnet.public-subnet-1.id, aws_subnet.public-subnet-2.id]
}
resource "aws_lb_target_group" "target-elb" {
 name = "ALB-TG"
 port = 80
 protocol = "HTTP"
 vpc_id = aws_vpc.demovpc.id
}
resource "aws_lb_target_group_attachment" "attachment" {
 target_group_arn = aws_lb_target_group.target-elb.arn
 target_id = aws_instance.demoinstance.id
 port = 80
 depends_on = [
 aws_instance.demoinstance,
]
}
resource "aws_lb_target_group_attachment" "attachment1" {
 target_group_arn = aws_lb_target_group.target-elb.arn
 target_id = aws_instance.demoinstance1.id
 port = 80
 depends_on = [
 aws_instance.demoinstance1,
]
}
resource "aws_lb_listener" "external-elb" {
 load_balancer_arn = aws_lb.external-alb.arn
 port = "80"
 protocol = "HTTP"
 default_action {
 type = "forward"
 target_group_arn = aws_lb_target_group.target-elb.arn
 }
}
```

## Terraform apply

```

ec2-user@ip-172-31-9-28:~/.terraformrc ~ +
+ target_id = "i-04f152afcd678f0ad"
}

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

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

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

Enter a value: yes

aws_lb_target_group.target-elb: Creating...
aws_lb.external-alb: Creating...
aws_lb_target_group.target-elb: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4]
aws_lb_target_group_attachment.attachment: Creating...
aws_lb_target_group_attachment.attachment1: Creating...
aws_lb_target_group_attachment.attachment1: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4-2023010209353080100000001]
aws_lb_target_group_attachment.attachment: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4-20230102093533098600000002]
aws_lb.external-alb: Still creating... [10s elapsed]
aws_lb.external-alb: Still creating... [20s elapsed]
aws_lb.external-alb: Still creating... [30s elapsed]
aws_lb.external-alb: Still creating... [40s elapsed]
aws_lb.external-alb: Still creating... [50s elapsed]
aws_lb.external-alb: Still creating... [1m0s elapsed]
aws_lb.external-alb: Still creating... [1m10s elapsed]
aws_lb.external-alb: Still creating... [1m20s elapsed]
aws_lb.external-alb: Still creating... [1m30s elapsed]
aws_lb.external-alb: Still creating... [1m40s elapsed]
aws_lb.external-alb: Still creating... [1m50s elapsed]
aws_lb.external-alb: Still creating... [2m0s elapsed]
aws_lb.external-alb: Creation complete after 2m1s [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:loadbalancer/app/External-LB/4d513aa75b7adcd5]
aws_lb_listener.external-elb: Creating...
aws_lb_listener.external-elb: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:listener/app/External-LB/4d513aa75b7adcd5/a7ce67a51b9ca57]

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

28°C Partly sunny 15:33 IN 02-01-2023

Create rds.tf and write the following script

multi-az is set to true for the high availability

```

ec2-user@ip-172-31-9-28:~/.terraformrc ~ +
[ec2-user@ip-172-31-9-28 terraform]$ vi rds.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init

Initializing the backend...

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

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[ec2-user@ip-172-31-9-28 terraform]$ terraform apply
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160adc94c3636861]
aws_vpc.demovpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_internet_gateway.igw1: Refreshing state... [id=igw-07a6c4cccd2deb84fd]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-02a501ec4e7c2bb6d]
aws_lb_target_group.target-elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0ce188e07b5efbe7b]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0aec0f65383dcf4e68]
aws_route_table.route: Refreshing state... [id=rtb-088e69f09eaba0f31]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04f78b3b8cd4a10aa]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0934661eeef3871db8]
aws_security_group.demosg: Refreshing state... [id=sg-02ba990d05b7cf2f0]
aws_instance.demoinstance: Refreshing state... [id=i-04f152afcd678f0ad]
aws_security_group.database-sg: Refreshing state... [id=sg-0b4af5731c7dd254d]
aws_route_table_association.rti: Refreshing state... [id=rtbassoc-027b6579877e1e560]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-0a74b3bf13b8e85c3]
aws_lb.external-alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:loadbalancer/app/External-LB/4d513aa75b7adcd5]
aws_instance.demoinstance1: Refreshing state... [id=i-0fecb1145449837b9]
aws_lb_listener.external-elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:listener/app/External-LB/4d513aa75b7adcd5/a7ce67ad51b9ca57]
aws_lb_target_group_attachment.attachment1: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4-20230102093533098600000001]
aws_lb_target_group_attachment.attachment: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4-20230102093533098600000002]

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

28°C Partly sunny 15:33 IN 02-01-2023

```
Creating RDS Instance
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 = "5.7"
 instance_class = "db.t2.micro"
 multi_az = true
 name = "mydb"
 username = "username"
 password = "password"
 skip_final_snapshot = true
 vpc_security_group_ids = [aws_security_group.database-sg.id]
```

```
aws_db_instance.default: Still creating... [0m0s elapsed]
aws_db_instance.default: Still creating... [0m10s elapsed]
aws_db_instance.default: Still creating... [0m20s elapsed]
aws_db_instance.default: Still creating... [0m30s elapsed]
aws_db_instance.default: Still creating... [0m40s elapsed]
aws_db_instance.default: Still creating... [0m50s elapsed]
aws_db_instance.default: Still creating... [0m59s 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... [1m59s 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... [2m59s 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... [3m59s 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... [4m59s 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... [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... [6m59s 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... [7m59s 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... [8m59s 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: Still creating... [9m59s elapsed]
aws_db_instance.default: Still creating... [10m0s elapsed]
aws_db_instance.default: Still creating... [10m10s elapsed]
aws_db_instance.default: Still creating... [10m20s elapsed]
aws_db_instance.default: Still creating... [10m30s elapsed]
aws_db_instance.default: Still creating... [10m40s elapsed]
aws_db_instance.default: Still creating... [10m50s elapsed]
aws_db_instance.default: Still creating... [10m59s elapsed]
aws_db_instance.default: Creation complete after 10m57s [id=terraform-20230102093908774800000001]

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

Use db_name instead

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
[ec2-user@ip-172-31-9-28 terraform]$
```

And create the file vi variable.tf

```

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

```

```

[ec2-user@ip-172-31-9-28 ~] + | -
[ec2-user@ip-172-31-9-28 terraform]$ vi vars.tf
[ec2-user@ip-172-31-9-28 terraform]$ terraform init
initializing the backend...
initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.08.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
run this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
[ec2-user@ip-172-31-9-28 terraform]$ terraform apply
aws_vpc.demovpc: Refreshing state... [id=arn:aws:vpc:us-east-2:404694279983:vpc-02d4e4583821a1e8]
aws_subnet.demosubnet: Refreshing state... [id=arn:aws:ec2:us-east-2:404694279983:subnet-07a6c4cc2debaf4d]
aws_subnet_public.demosubnet: Refreshing state... [id=arn:aws:ec2:us-east-2:404694279983:subnet-0168adcd94c3636861]
aws_lb_target_group.target-lb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4]
aws_subnet.demobucket-subnet-1: Refreshing state... [id=arn:aws:ec2:us-east-2:404694279983:subnet-02a53dcf4f4e8]
aws_security_group.demosubnet-1: Refreshing state... [id=arn:aws:ec2:us-east-2:404694279983:security-group-02a53dcf4f4e8]
aws_route_table.demosubnet-1: Refreshing state... [id=arn:aws:ec2:us-east-2:404694279983:route-table-02a53dcf4f4e8]
aws_subnet.application-subnet-2: Refreshing state... [id=arn:aws:ec2:us-east-2:404694279983:subnet-02a51elec47c2bb6d]
aws_subnet.application-subnet-3: Refreshing state... [id=arn:aws:ec2:us-east-2:404694279983:subnet-093461eef3f71db8]
aws_subnet.demosecure-subnet: Refreshing state... [id=arn:aws:ec2:us-east-2:404694279983:subnet-0c188ef7b5efbe7b]
aws_subnet_database-subnet-2: Refreshing state... [id=arn:aws:ec2:us-east-2:404694279983:subnet-0c188ef7b5efbe7b]
aws_route_table_association.rtl: Refreshing state... [id=arn:aws:ec2:us-east-2:404694279983:route-table-association-027b6579877e1e568]
aws_security_group.demosecure-sql: Refreshing state... [id=arn:aws:ec2:us-east-2:404694279983:security-group-0b4af5731c7dd234d]
aws_subnet.demosecure-subnet: Refreshing state... [id=arn:aws:ec2:us-east-2:404694279983:subnet-0c188ef7b5efbe7b]
aws_instance.demoinstance: Refreshing state... [id=arn:aws:ec2:us-east-2:404694279983:instance-0d0f152afcd678f0ad]
aws_lb.external-alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:loadbalancer/app/External-LB/4d513aa75b7adcd5]
aws_route_table_association.rta: Refreshing state... [id=arn:aws:ec2:us-east-2:404694279983:route-table-association-0d513aa75b7adcd5]
aws_lb_listener.demosecure-alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:listener/app/External-LB/4d513aa75b7adcd5/a7ce07ad51b9ca57]
aws_db_instance.default: Refreshing state... [id=arn:aws:rds:us-east-2:404694279983:db-instance-202301020939087748800000001]
aws_lb_target_group_attachment.attachment: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4-20230102093533098600000002]
aws_lb_target_group_attachment.attachment1: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c53f8319e4-20230102093533080100000001]

No changes. Your infrastructure matches the configuration.

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

Warning: Argument is deprecated
 with aws_db_instance.default,
 on rds.tf line 16, in resource "aws_db_instance" "default":
 16: name = "mydb"
 Use db_name instead
 (and one more similar warning elsewhere)

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

Outputs:

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

We will get the DNS of the application load balancer.

And create vi output.tf

```
Getting the DNS of load balancer
output "lb_dns_name" {
| description = "The DNS name of the load balancer"
| value = "${aws_lb.external-alb.dns_name}"
}
~n
```

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

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

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
return this command to reinitialize your working directory. If you forget, other
commands will detect it and prompt you to run this command as necessary.

[ec2-user@ip-172-31-9-28 terraform]$ terraform apply
aws_subnet.public-subnet-1: Refreshing state... [id=subnet-0160ad9c94c5636860]
aws_vpc_demo:vpc: Refreshing state... [id=vpc-022181cd683d21e02]
aws_internet_gateway.igw1: Refreshing state... [id=igw-07accc2deb8df]
aws_route_table.rt1: Refreshing state... [id=routetable-0000000000000000]
aws_db_instance.databaseinst: Refreshing state... [id=dbi-0000000000000000]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-a2a501ec04c7chh6r]
aws_lb_target_group.target-elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c5f8319eu]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-0934661ef3871db8]
aws_security_group.demo: Refreshing state... [id=sg-02ba990905b7cf2f8]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-04f7bd3bcda10aa]
aws_subnet.public-subnet-2: Refreshing state... [id=subnet-04f7bd3bcda10aa]
aws_route_table_association.rtl: Refreshing state... [id=rtbassoc-0276579977e1e560]
aws_route_table_association.rtl2: Refreshing state... [id=rtbassoc-047a3bf113be85c3]
aws_instance.instance1: Refreshing state... [id=i-0fecb1145409837b9]
aws_security_group.database-sg: Refreshing state... [id=sg-004af731c7dd25ad]
aws_db_instance.databaseinst: Refreshing state... [id=dbi-0000000000000000]
aws_instance.instance1: Refreshing state... [id=i-0ufi152dc07ff0ad]
aws_db_subnet_group.default: Refreshing state... [id=main]
aws_lb_listener.external-elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:listener/app/External-LB/4d513aa75b7adc5]
aws_db_instance.default: Refreshing state... [id=terraform-202301020939077480000001]
aws_lb_target_group.attachment.attachment: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c5f8319eu-20230102093533098600000002]
aws_lb_target_group.attachment.attachment1: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-2:404694279983:targetgroup/ALB-TG/788c14c5f8319eu-20230102093533098600000001]

Changes to Outputs:
+ lb_dns_name = "External-LB-1663777618.us-east-2.elb.amazonaws.com"

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

Warning: Argument is deprecated

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

Use db_name instead

(And one more similar warning elsewhere)

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

Enter a value: yes

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