AWS Infrastructure Automation Using Terraform

Presented By

Amgoth Pravalika

Workflow Overview

1.Create VPC.

2.Create Subnet

3.Set up Internet Gateway.

4.Configure Route Table.

5.Provision EC2 Instances.

6.Configure Security Groups (Frontend and Database).

7.Set up Application Load Balancer.

8.Deploy RDS.

9.Define Outputs.

10.Utilize Variables.

11.Implement User Data for automation.

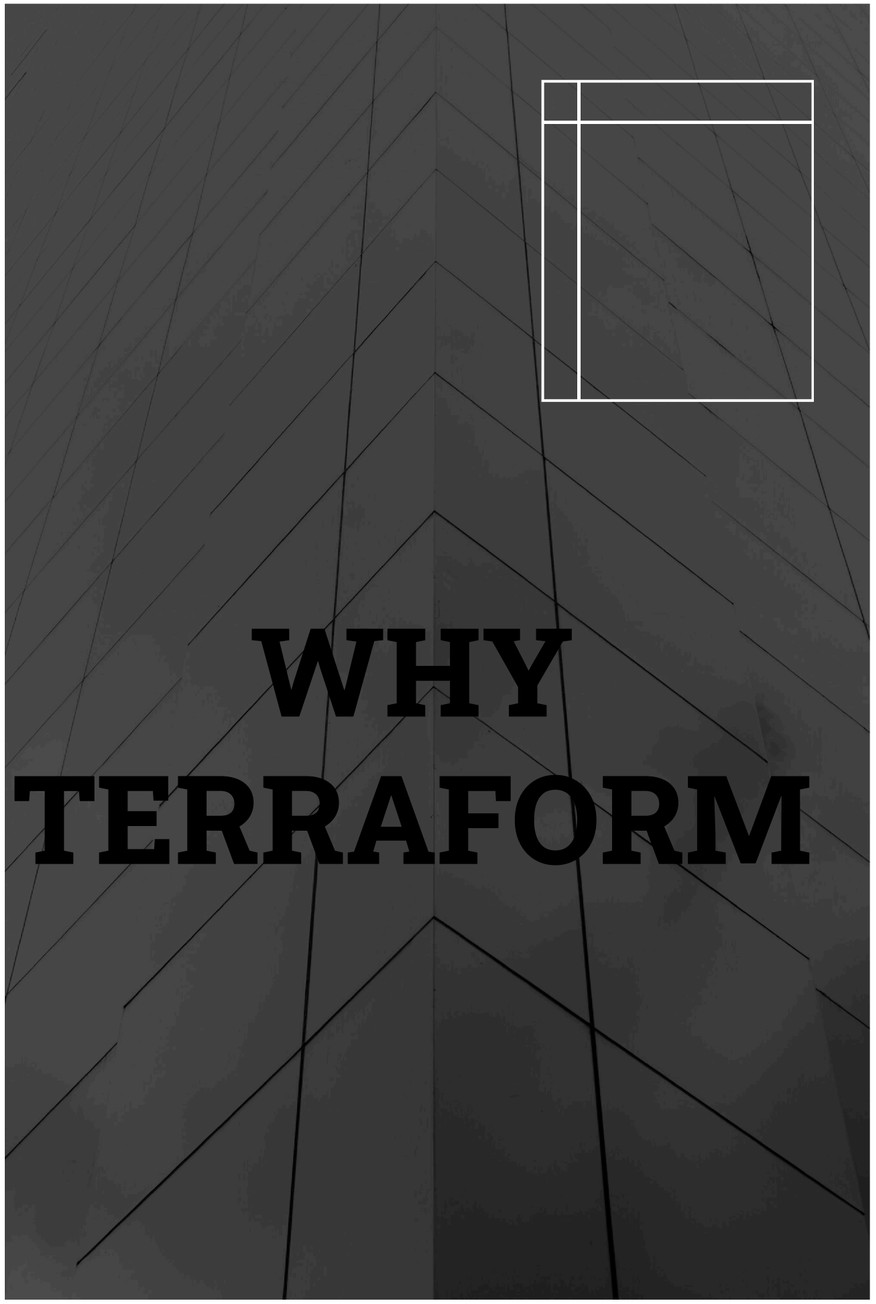
Project Objective

- Automate the creation and management of AWS infrastructure.

- Enhance scalability, consistency, and deployment speed using Terraform.

WHAT IS TERRAFORM

Terraform is an open-source infrastructure as a code(Iac) tool by Hashicorp.It helps define,provision and manage infrastructure across multiple platforms using a declarative configuration language(HCL)



WHY TERRAFORM

Infrastructure as Code (IaC)

Automation

Dependency Management

Execution Plans

State Management

Scalable & Cost-Effective

Terraform Commands

terraform init

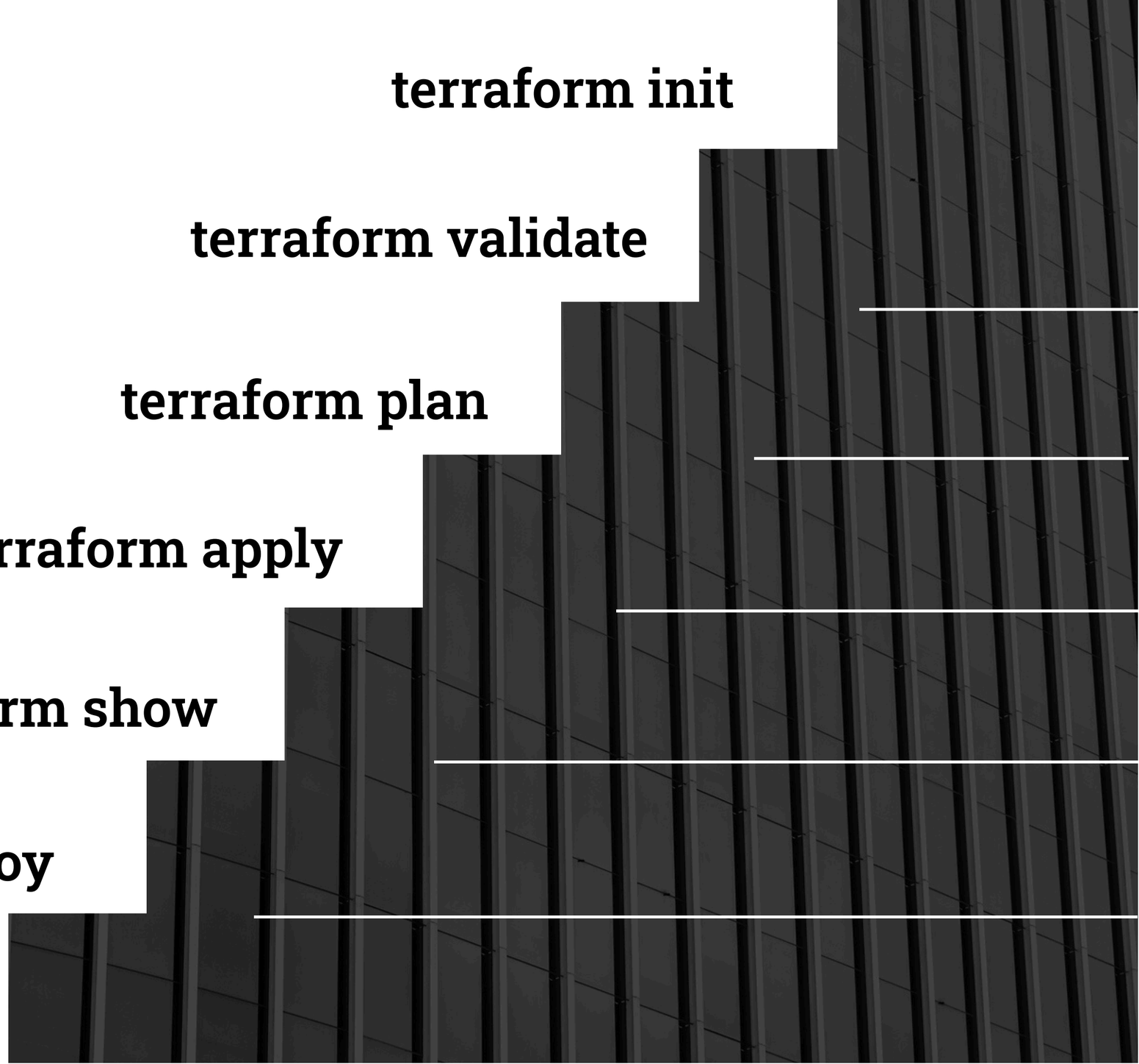
terraform validate

terraform plan

terraform apply

terraform show

terraform destroy

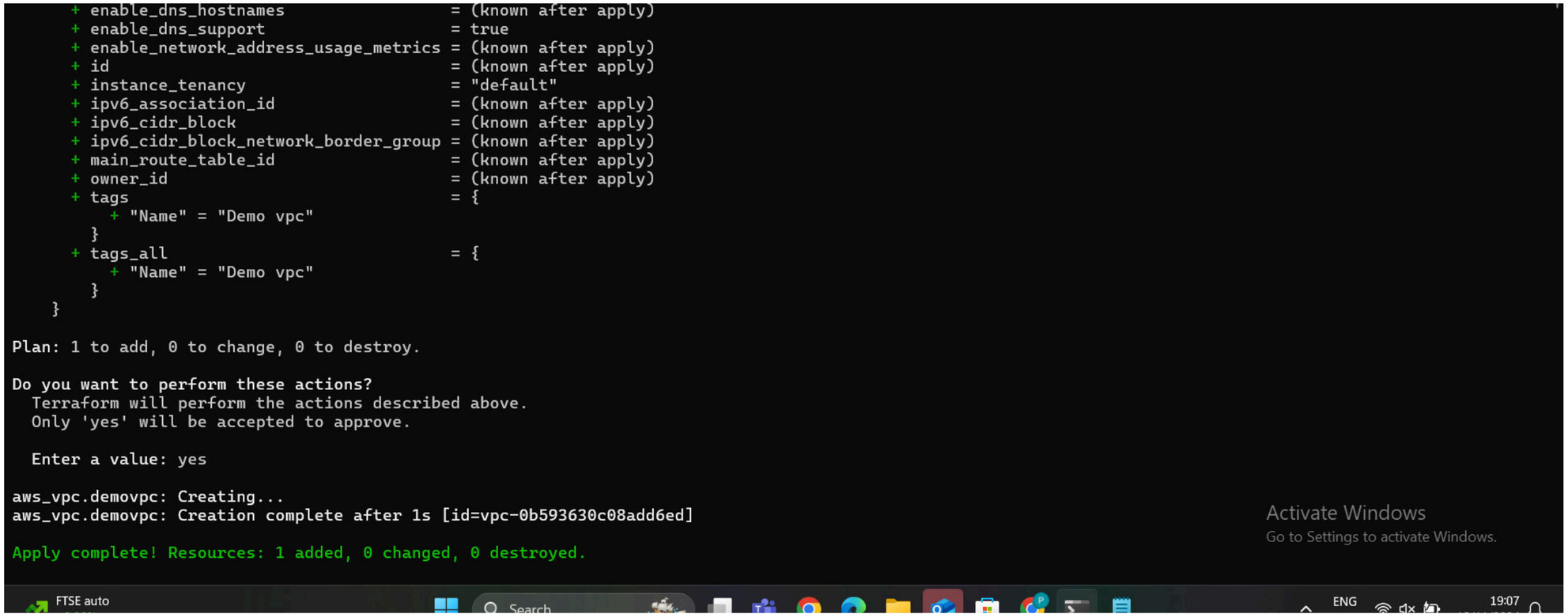


Process we follow

1. Install Terraform
2. Configure AWS CLI
3. Write Terraform Configuration Files [create.tf file]
4. Initialize Terraform-
5. Plan Infrastructure
6. Apply Configuration
7. Verify in AWS Console

step-1 :create vpc

A Virtual Private Cloud (VPC)is a private network within AWS that allows you to launch and manage resources in a secure and isolated environment. It enables you to customize IP address ranges, create subnets, configure route tables, and set up internet gateways, ensuring complete control over your network setup.

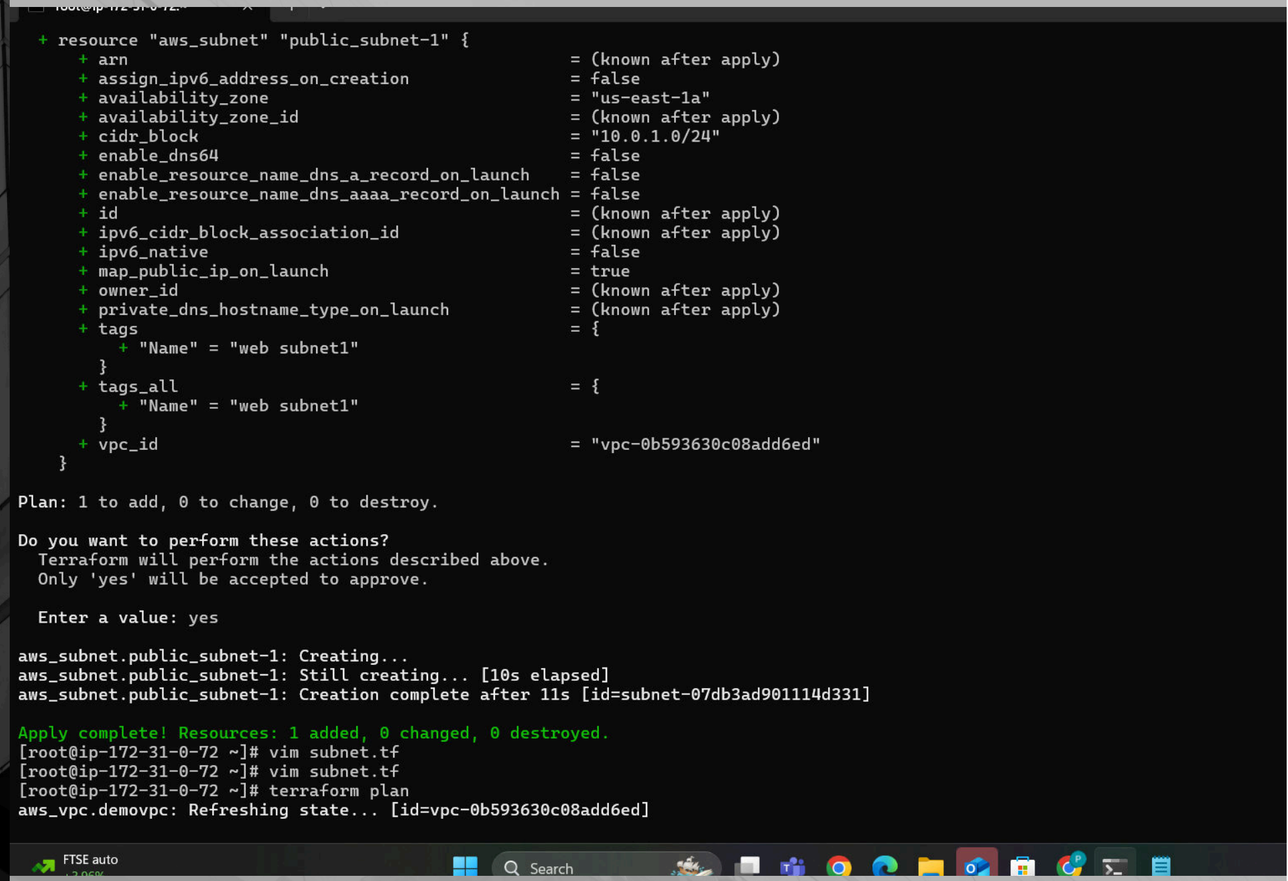


vpc.tf file

```
provider "aws" {
region="us-east-1"
}
resource "aws_vpc" "demovpc" {
cidr_block="${var.vpc_cidr}"
instance_tenancy="default"
tags={
Name="Demo vpc"
}
}
```

step-2:create subnet

ASubnet is a smaller network within a VPC, used to organize resources and improve security. It can be public(internet-accessible) or private(isolated from the internet).

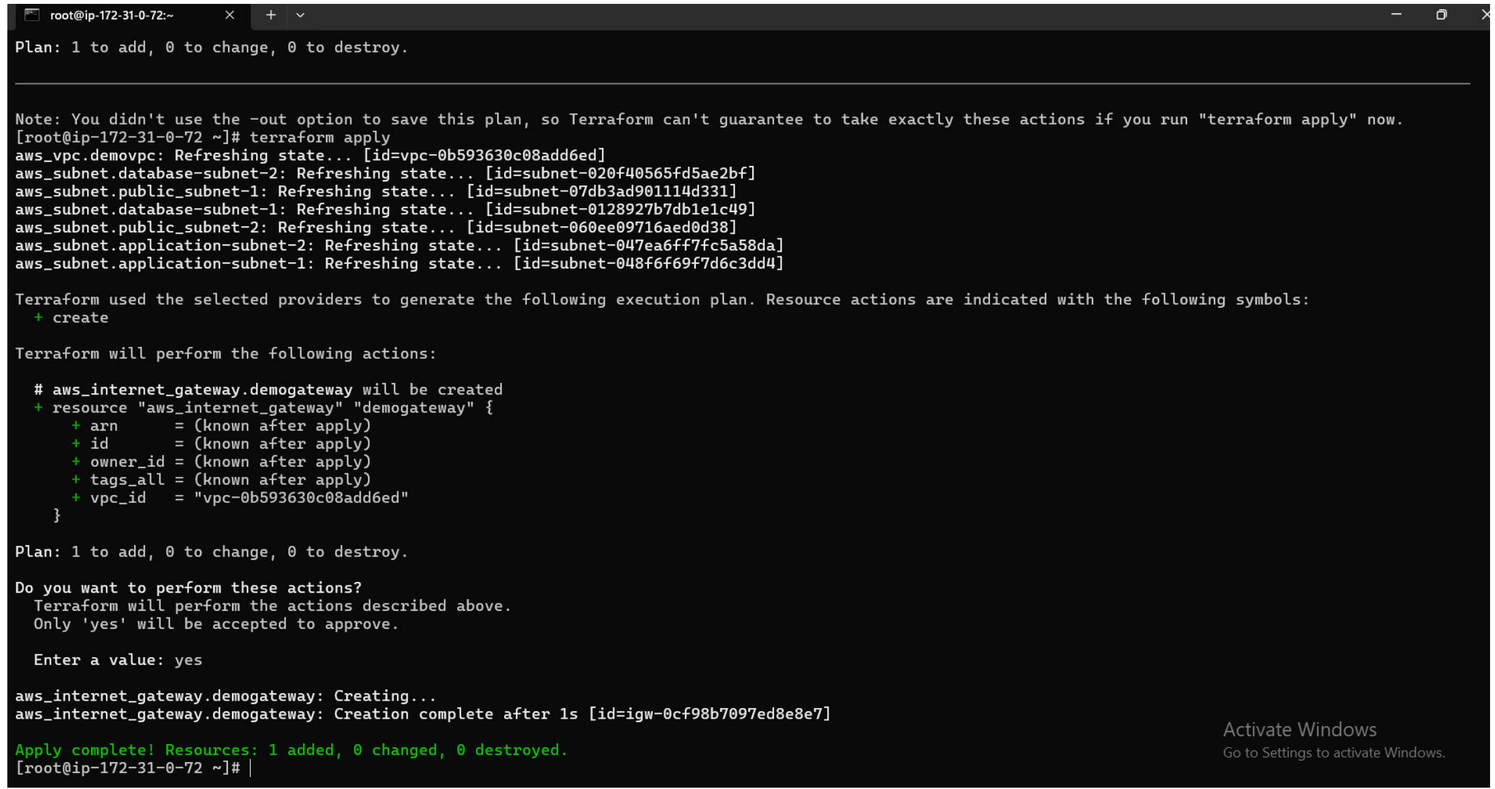


subnet.tf file

```
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 subnet1"
}
}
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}"
```

Step-3:Create internet gateway

An Internet Gatewayis a network device in AWS that allows communication between resources in a VPC and the internet. It enables instances in public subnets to send and receive traffic from the internet.



igf.tf file

```
resource "aws_internet_gateway" "demogateway" {
vpc_id="${aws_vpc.demovpc.id}"
}
```

Step-4:Create route table

A Route Table in AWS is a set of rules (routes) used to determine where network traffic from your VPC is directed. Each subnet in a VPC is associated with a route table, which specifies how traffic should flow to different destinations, such as the internet, other subnets, or on-premises networks.

```
root@ip-172-31-0-72:~

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-0cf98b7097ed8e8e7"
                # (11 unchanged attributes hidden)
            },
        ]
      + tags             = {
          + "Name" = "Route to Internet"
        }
      + tags_all         = {
          + "Name" = "Route to Internet"
        }
      + vpc_id           = "vpc-0b593630c08add6ed"
    }

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

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

  Enter a value: yes

aws_route_table.route: Creating...
aws_route_table.route: Creation complete after 1s [id=rtb-0f9de308660561356]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-0-72 ~]#
```

Route_table.tf file

```
resource "aws_route_table" "route" {
vpc_id="${aws_vpc.demovpc.id}"
route {
cidr_block="0.0.0.0/0"
gateway_id="${aws_internet_gateway.demogateway.id}"
}
tags={
Name="Route to Internet"
}
}

resource "aws_route_table_association" "rt1" {
subnet_id="${aws_subnet.public_subnet-1.id}"
route_table_id="${aws_route_table.route.id}"
}

resource "aws_route_table_association" "rt2" {
subnet_id="${aws_subnet.public_subnet-2.id}"
route_table_id="${aws_route_table.route.id}"
}
```

Step-5:Create Ec2-Instance

An EC2 instance (Elastic Compute Cloud) is a virtual server in AWS used to run applications and services. It provides scalable compute capacity, allowing you to choose different instance types based on performance needs (CPU, memory, storage) and to manage them easily via the AWS console or APIs. EC2 instances can run a variety of operating systems, including Linux and Windows.4o mini

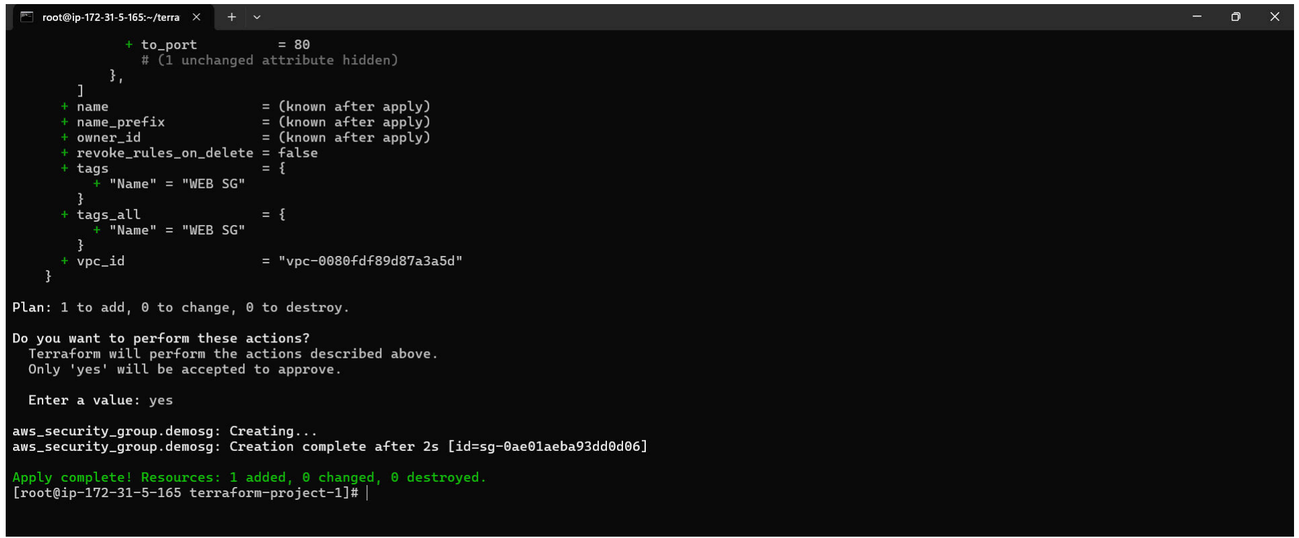
Ec2.tf file

```
resource "aws_instance" "public_subnet-1" {
ami="ami-012967cc5a8c9f891"
instance_type="t2.micro"
count=1
key_name="pinku"
vpc_security_group_ids=[aws_security_group.demosg.id]
subnet_id="${aws_subnet.public_subnet-1.id}"
associate_public_ip_address=true
user_data="${file("data.sh")}"
tags={
Name="My public Instance 1 "
}
}

resource "aws_instance" "public_subnet-2" {
ami="ami-012967cc5a8c9f891"
instance_type="t2.micro"
count=1
key_name="pinku"
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 "
}
}
```

Step-6:Security group:Frontend

A Frontend Security Group is a set of firewall rules in AWS that control inbound and outbound traffic to resources (like EC2 instances) in the frontend layer of your application.

HTTP (port 80) and HTTPS (port 443) traffic from the internet for web applications.
SSH (port 22) access from trusted IPs for administration purposes.



Web_SG.tf file

```
resource "aws_security_group" "demosg" {
vpc_id="${aws_vpc.demovpc.id}"

ingress{
from_port=80
to_port=80
protocol="tcp"
cidr_blocks=["0.0.0.0/0"]
}

ingress{
from_port=443
to_port=443
protocol="tcp"
cidr_blocks=["0.0.0.0/0"]
}

ingress{
from_port=22
to_port=22
protocol="tcp"
cidr_blocks=["0.0.0.0/0"]
}

egress{
from_port=0
to_port=0
protocol="-1"
cidr_blocks=["0.0.0.0/0"]
}
tags={
Name="WEB_SG"
```

Step-7:Security group(Data base)

A Database Security Groupis a set of firewall rules in AWS that controls inbound and outbound traffic to database resources (e.g., RDS instances). Typically, it allows:

Traffic from trusted sources(e.g., EC2 instances in the same VPC or specific IP ranges) on database ports (e.g.,MySQL: 3306,PostgreSQL: 5432).

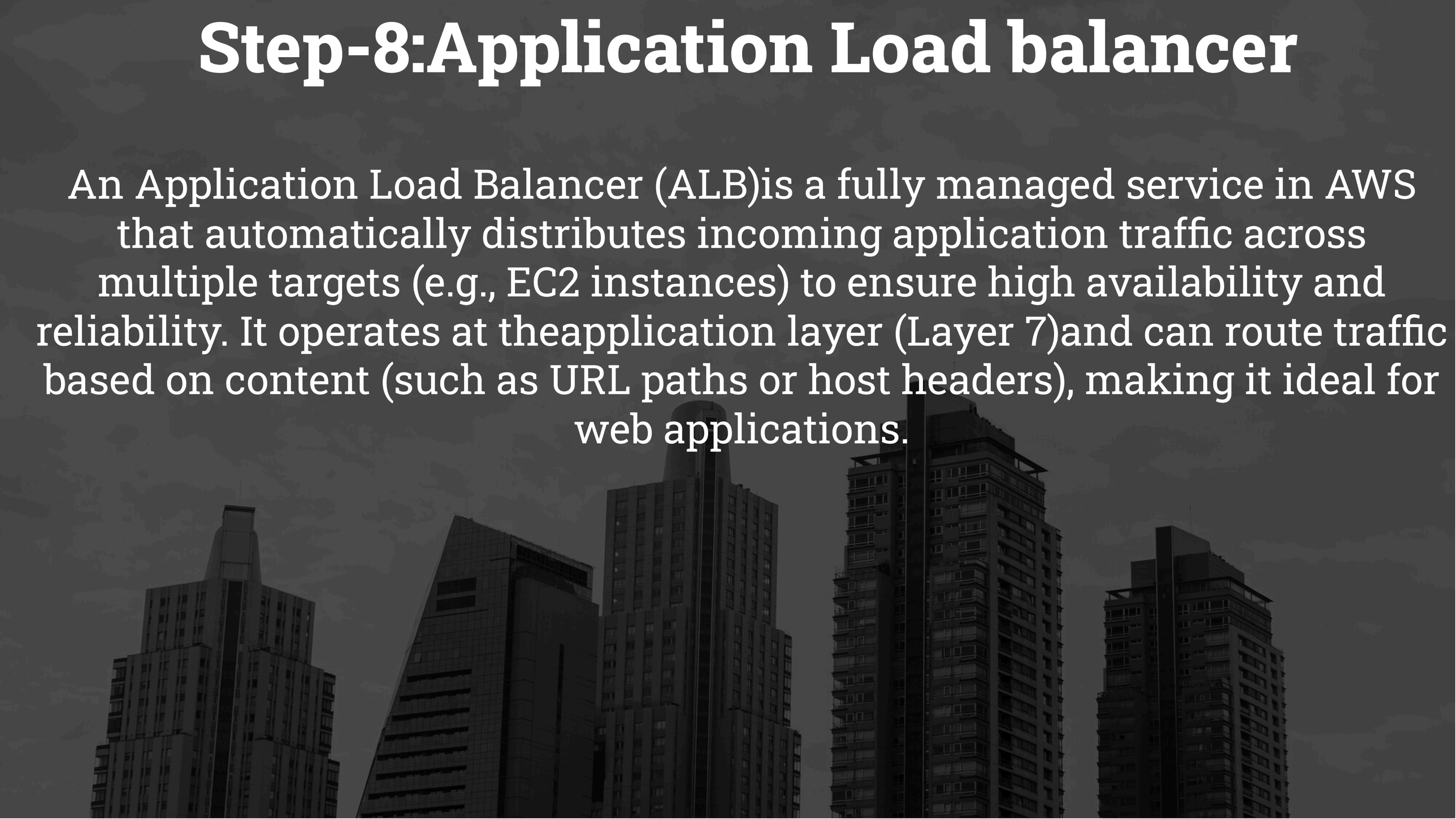
Deny all other trafficto secure the database from unauthorized access.

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=0
to_port=65535
protocol="tcp"
cidr_blocks=["0.0.0.0/0"]
}
tags={
Name="Databse SG"
}
}
```

Step-8:Application Load balancer

An Application Load Balancer (ALB)is a fully managed service in AWS that automatically distributes incoming application traffic across multiple targets (e.g., EC2 instances) to ensure high availability and reliability. It operates at theapplication layer (Layer 7)and can route traffic based on content (such as URL paths or host headers), making it ideal for web applications.



Alb.tf file

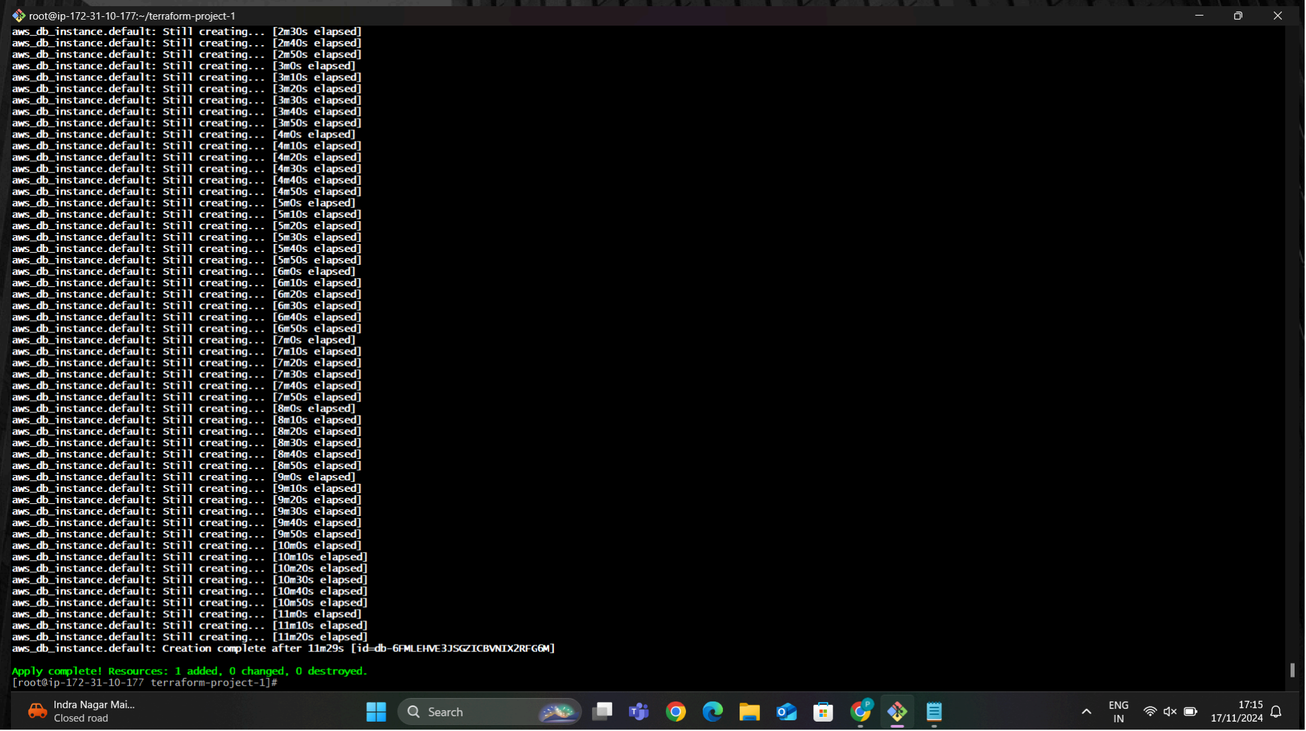
```
resource "aws_lb" "external-alb" {
name="Externa-LB"
internal=false
load_balancer_type="application"
security_groups=[aws_security_group.demosg.id]
subnets=[aws_subnet.public_subnet-1.id,aws_subnet.public_subnet-2.id]
}


resource "aws_lb_target_group" "external-alb" {
name="ALB-TG"
port=80
protocol="HTTP"
vpc_id=aws_vpc.demovpc.id
}

resource "aws_lb_target_group_attachment" "attachment1" {
target_group_arn=aws_lb_target_group.external-alb.arn
target_id="${aws_instance.public_subnet-1[0].id}"
port=80
depends_on=[aws_instance.public_subnet-1
]
}

resource "aws_lb_target_group_attachment" "attachment2" {
target_group_arn=aws_lb_target_group.external-alb.arn
target_id="${aws_instance.public_subnet-2[0].id}"
port=80
depends_on=[
aws_instance.public_subnet-2
]
```

$20,000

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

Step-9:RDS Instance

An RDS (Relational Database Service) instance is a managed database service in AWS that makes it easy to set up, operate, and scale a relational database. RDS supports multiple database engines, including MySQL, PostgreSQL, Oracle, SQL Server, and MariaDB.

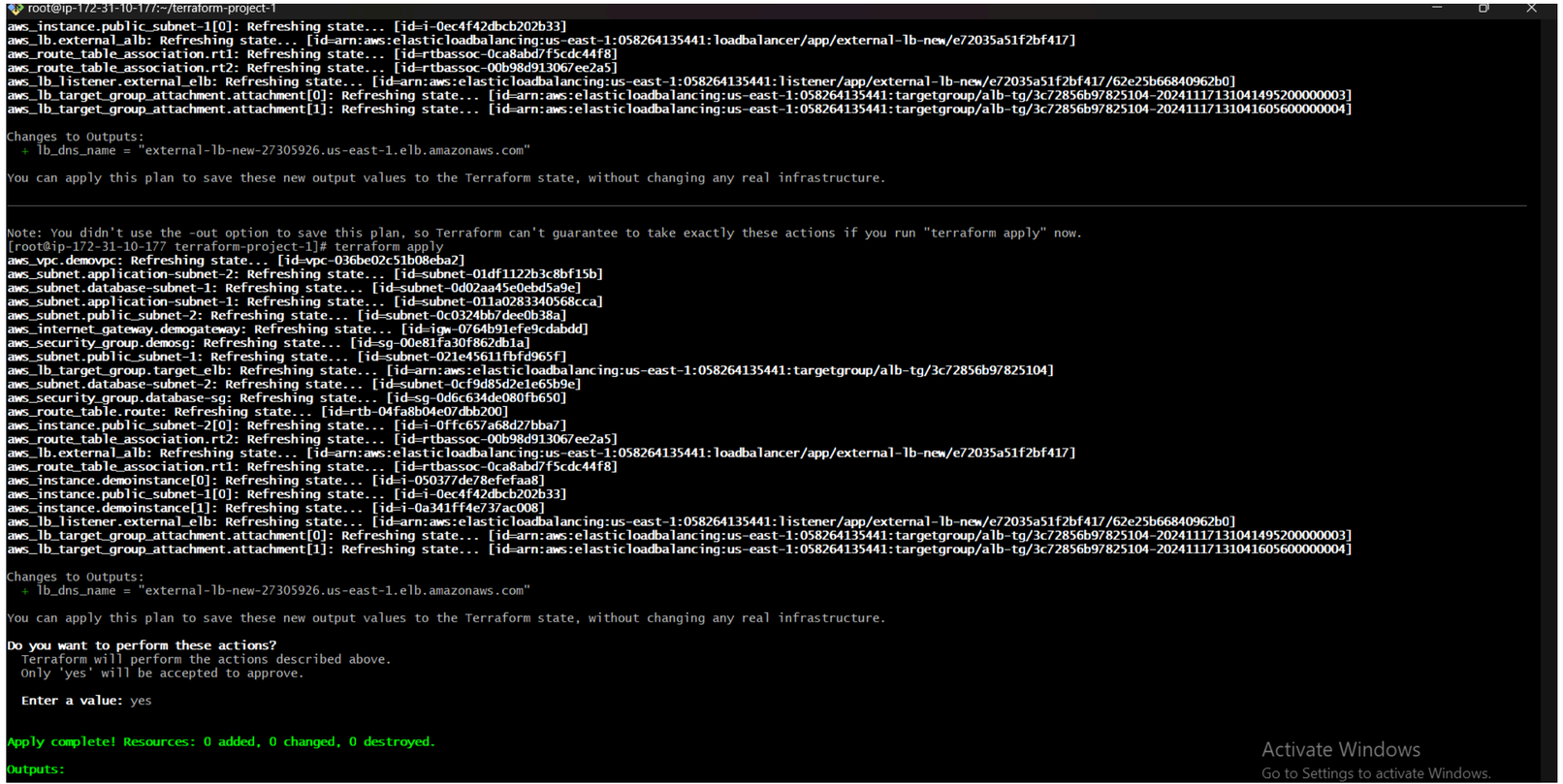


RDS.tf file

```
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="8.0.32"
instance_class="db.t3.micro"
multi_az=true
db_name="mydb"
username="username"
password="password"
skip_final_snapshot=true
vpc_security_group_ids=[aws_security_group.database-sg.id]
}
```

Step-10:Outputs

The outputblock in Terraform is used to display the value of a resource or variable after the infrastructure is created or modified. In this specific example, the output block will display the DNS name of an AWS Application Load Balancer (ALB).



Outputs.tf file

```
output "lb_dns_name" {
description ="The DNS Name of the Load Balancer"
value="${aws_lb.external_alb.dns_name}"
}
```

Step-11:Variables

In this Terraform configuration, thevariablesdefine the IP address ranges (CIDR blocks) for different network components, such as the VPC and subnets. These variables are used to parameterize the Terraform code, making it flexible and reusable.

Vars.tf file

```
variable "vpc_cidr" {
default ="10.0.0.0/16"
}

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

variable "subnet2_cidr" {
default="10.0.2.0/24"
}

variable "subnet3_cidr" {
default ="10.0.3.0/24"
}

variable "subnet4_cidr" {
default="10.0.4.0/24"
}

variable "subnet5_cidr" {
default="10.0.5.0/24"
}

```

Step-12:Users data

```
#!/bin/bash
yum update -y
yum install -y httpd.x86_64
systemctl start httpd.service
systemctl enable httpd.service
echo "Hello World from ec2 instance
This is Amgoth Pravalika
Working on second project
Second project is terraform $(hostname -f)" > /var/www/html/index.html
```


THANK
YOU