

---



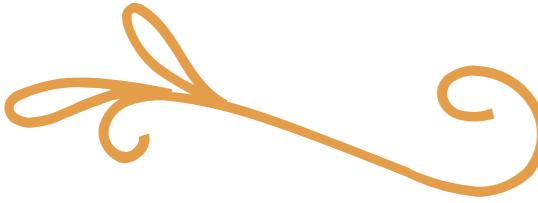
# AWS Infrastructure Automation Using Terraform

---



**Presented By**

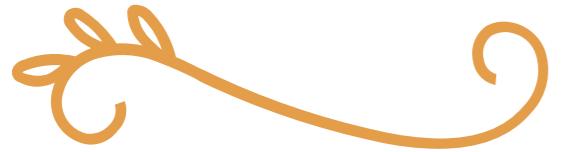
**Amgoth Pravalika**



# 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.



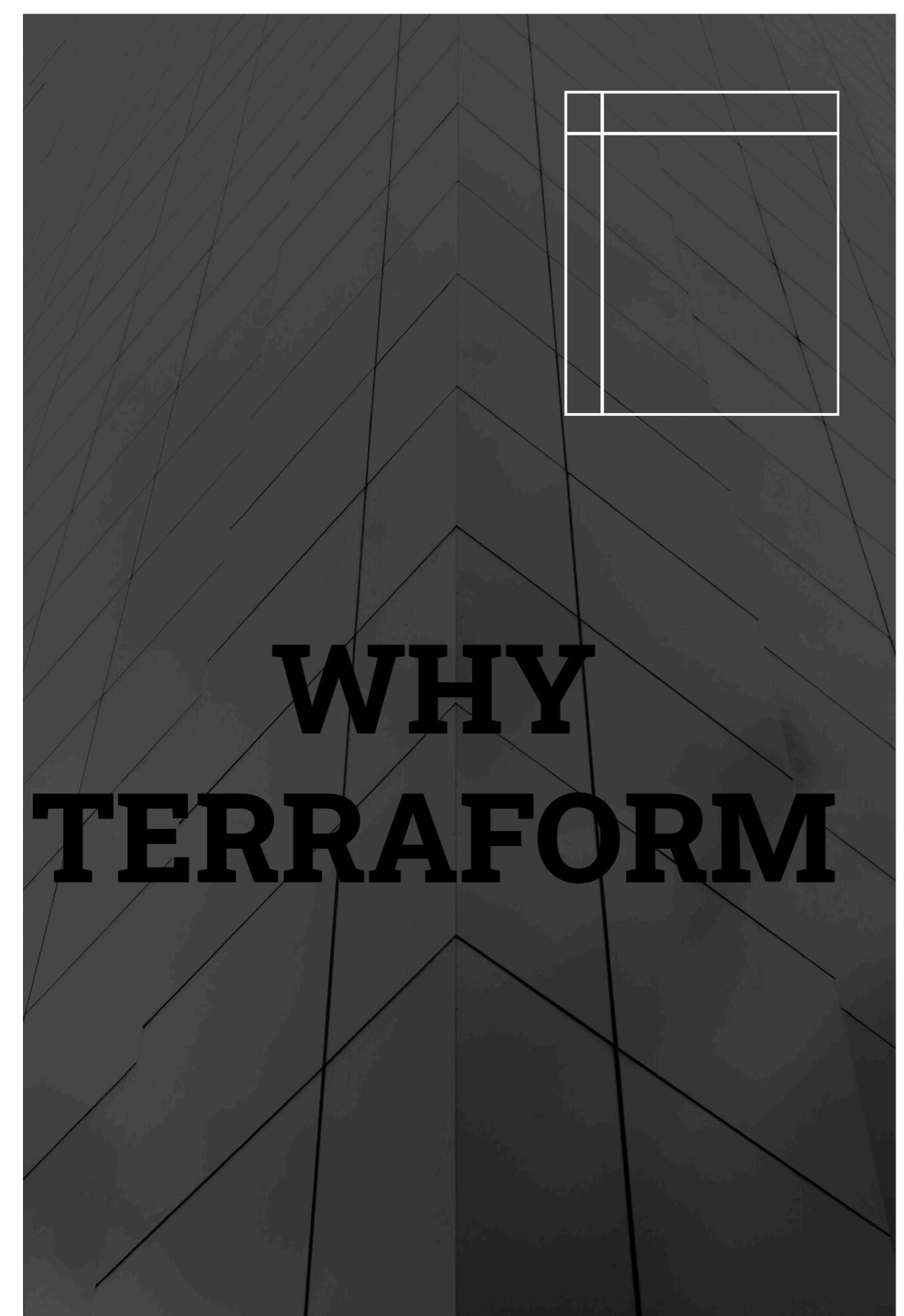
# Project Objective



- 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.

```
+ enable_dns_hostnames          = (known after apply)
+ enable_dns_support             = true
+ enable_network_address_usage_metrics = (known after apply)
+ id                            = (known after apply)
+ instance_tenancy               = "default"
+ ipv6_association_id           = (known after apply)
+ ipv6_cidr_block                = (known after apply)
+ ipv6_cidr_block_network_border_group = (known after apply)
+ main_route_table_id            = (known after apply)
+ owner_id                       = (known after apply)
+ tags
  + "Name" = "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-0b593630c08add6ed]
```

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

Activate Windows  
Go to Settings to activate Windows.



# vpc.tf file



```
1 provider "aws" {  
2   region="us-east-1"  
3 }  
4 resource "aws_vpc" "demovpc" {  
5   cidr_block="${var.vpc_cidr}"  
6   instance_tenancy="default"  
7   tags={  
8     Name="Demo vpc"  
9   }  
10 }
```

# step-2:create subnet

A Subnet 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).

```
+ 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 subnet1"
    }
    + tags_all                           = {
        + "Name" = "web subnet1"
    }
    + vpc_id                            = "vpc-0b593630c08add6ed"
}

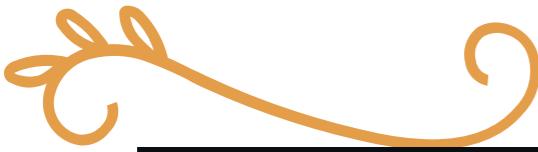
Plan: 1 to add, 0 to change, 0 to destroy.

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_subnet.public_subnet-1: Creating...
aws_subnet.public_subnet-1: Still creating... [10s elapsed]
aws_subnet.public_subnet-1: Creation complete after 11s [id=subnet-07db3ad901114d331]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-0-72 ~]# vim subnet.tf
[root@ip-172-31-0-72 ~]# vim subnet.tf
[root@ip-172-31-0-72 ~]# terraform plan
aws_vpc.demoVPC: Refreshing state... [id=vpc-0b593630c08add6ed]
```



# subnet.tf file



```
resource "aws_subnet" "public_subnet-1" {  
    vpc_id = "${aws_vpc демоврс.id}"  
    cidr_block = "${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 = "${aws_vpc демоврс.id}"  
    cidr_block = "${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 = "${aws_vpc демоврс.id}"  
    cidr_block = "${var.subnet3_cidr}"
```

# Step-3:Create internet gateway

An Internet Gateway is 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.

```
root@ip-172-31-0-72:~ x + v
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-172-31-0-72 ~]# terraform apply
aws_vpc.demovpc: Refreshing state... [id=vpc-0b593630c08add6ed]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-020f40565fd5ae2bf]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-07db3ad901114d331]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0128927b7db1e1c49]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-060ee09716aed0d38]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-047ea6ff7fc5a58da]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-048f6f69f7d6c3dd4]

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-0b593630c08add6ed"
}

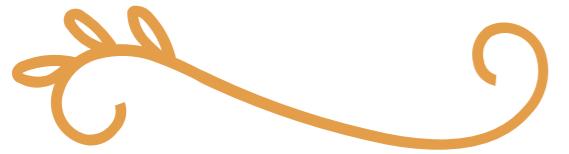
Plan: 1 to add, 0 to change, 0 to destroy.

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

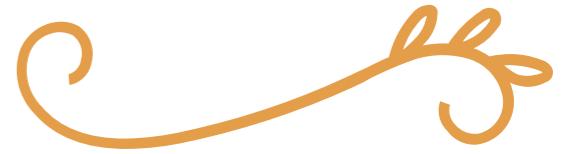
Enter a value: yes

aws_internet_gateway.demogateway: Creating...
aws_internet_gateway.demogateway: Creation complete after 1s [id=igw-0cf98b7097ed8e8e7]

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



# igf.tf file



```
1 resource "aws_internet_gateway" "demogateway" {
2   vpc_id = "${aws_vpc.demovpc.id}"
3 }
```

# Step-4:Create route table



```
root@ip-172-31-0-72:~ x + ~
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 ~]#
```

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.



# Route\_table.tf file



```
1   resource "aws_route_table" "route" {
2     vpc_id = "${aws_vpc.demovpc.id}"
3     route {
4       cidr_block = "0.0.0.0/0"
5       gateway_id = "${aws_internet_gateway.demogateway.id}"
6     }
7     tags = {
8       Name = "Route to Internet"
9     }
10   }
11
12   resource "aws_route_table_association" "rt1" {
13     subnet_id = "${aws_subnet.public_subnet-1.id}"
14     route_table_id = "${aws_route_table.route.id}"
15   }
16
17   resource "aws_route_table_association" "rt2" {
18     subnet_id = "${aws_subnet.public_subnet-2.id}"
19     route_table_id = "${aws_route_table.route.id}"
20   }
```

# 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.

# Ec2.tf file

```
1   resource "aws_instance" "public_subnet-1" {
2     ami="ami-012967cc5a8c9f891"
3     instance_type="t2.micro"
4     count=1
5     key_name="pinku"
6     vpc_security_group_ids=[aws_security_group.demosg.id]
7     subnet_id="${aws_subnet.public_subnet-1.id}"
8     associate_public_ip_address=true
9     user_data="${file("data.sh")}"
10    tags={
11      Name="My public Instance 1 "
12    }
13  }
14
15  resource "aws_instance" "public_subnet-2" {
16    ami="ami-012967cc5a8c9f891"
17    instance_type="t2.micro"
18    count=1
19    key_name="pinku"
20    vpc_security_group_ids=[aws_security_group.demosg.id]
21    subnet_id="${aws_subnet.public_subnet-2.id}"
22    associate_public_ip_address=true
23    user_data="${file("data.sh")}"
24    tags={
25      Name="My public Instance 2 "
26    }
27  }
```

# 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.

```
root@ip-172-31-5-165:~/terraform-project-1# terraform apply
aws_security_group.demosg: Creating...
aws_security_group.demosg: Creation complete after 2s [id=sg-0ae01aeba93dd0d06]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-5-165 terraform-project-1]#
```

# Web\_SG.tf file

```
1  resource "aws_security_group" "demosg" {
2    vpc_id = "${aws_vpc.demovpc.id}"
3
4    ingress{
5      from_port=80
6      to_port=80
7      protocol="tcp"
8      cidr_blocks=["0.0.0.0/0"]
9    }
10
11   ingress{
12     from_port=443
13     to_port=443
14     protocol="tcp"
15     cidr_blocks=["0.0.0.0/0"]
16   }
17
18   ingress{
19     from_port=22
20     to_port=22
21     protocol="tcp"
22     cidr_blocks=["0.0.0.0/0"]
23   }
24
25   egress{
26     from_port=0
27     to_port=0
28     protocol="-1"
29     cidr_blocks=["0.0.0.0/0"]
30   }
31   tags={
32     Name="WEB_SG"
```

# **Step-7:Security group(Data base)**

A Database Security Group is 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 traffic to secure the database from unauthorized access.

# Database\_sg.tf

```
1 resource "aws_security_group" "database-sg" {
2   name="Database SG"
3   description="Allow inbound traffic from application layer"
4   vpc_id=aws_vpc.demovpc.id
5   ingress{
6     description="allow traffic from application layer"
7     from_port=3306
8     to_port=3306
9     protocol="tcp"
10    security_groups=[aws_security_group.demosg.id]
11  }
12  egress {
13    from_port=0
14    to_port=65535
15    protocol="tcp"
16    cidr_blocks=["0.0.0.0/0"]
17  }
18  tags={
19    Name="Database SG"
20  }
21 }
```

# 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 the application 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

```
1 resource "aws_lb" "external-alb" {
2   name="External-LB"
3   internal=false
4   load_balancer_type="application"
5   security_groups=[aws_security_group.demosg.id]
6   subnets=[aws_subnet.public_subnet-1.id,aws_subnet.public_subnet-2.id]
7 }
8
9
0 resource "aws_lb_target_group" "external-alb" {
1   name="ALB-TG"
2   port=80
3   protocol="HTTP"
4   vpc_id=aws_vpc.demovpc.id
5 }
6
7 resource "aws_lb_target_group_attachment" "attachment1" {
8   target_group_arn=aws_lb_target_group.external-alb.arn
9   target_id="${aws_instance.public_subnet-1[0].id}"
0   port=80
1   depends_on=[aws_instance.public_subnet-1
2 ]
3 }
4
5 resource "aws_lb_target_group_attachment" "attachment2" {
6   target_group_arn=aws_lb_target_group.external-alb.arn
7   target_id="${aws_instance.public_subnet-2[0].id}"
8   port=80
9   depends_on=[
0   aws_instance.public_subnet-2
1 ]}
```

```
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
  }
}
```

\$20,000

# 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.

```
root@ip-172-31-10-177:~/terraform-project-1
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... [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: 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... [11m0s elapsed]
aws_db_instance.default: Still creating... [11m10s elapsed]
aws_db_instance.default: Still creating... [11m20s elapsed]
aws_db_instance.default: Creation complete after 11m29s [id=db-6FMLEHVE3JSGZICBVNIX2RFG6M]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
[root@ip-172-31-10-177 terraform-project-1]#
```

# RDS.tf file

```
1  resource "aws_db_subnet_group" "default" {
2    name="main"
3    subnet_ids=[aws_subnet.database-subnet-1.id, aws_subnet.database-subnet-2.id]
4    tags={
5      Name="My DB subnet group"
6    }
7  }
8  resource "aws_db_instance" "default" {
9    allocated_storage=10
10   db_subnet_group_name=aws_db_subnet_group.default.id
11   engine="MySQL"
12   engine_version="8.0.32"
13   instance_class="db.t3.micro"
14   multi_az=true
15   db_name="mydb"
16   username="username"
17   password="password"
18   skip_final_snapshot=true
19   vpc_security_group_ids=[aws_security_group.database-sg.id]
20 }
```

# 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).

```
root@ip-172-31-10-17:~/terraform-project-1
aws_instance.public_subnet-1[0]: Refreshing state... [id=i-0ec4f42dbcb202b33]
aws_lb.external_lb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:058264135441:loadbalancer/app/external-lb-new/e72035a51f2bf417]
aws_route_table_association.rt1: Refreshing state... [id=rtbassoc-0ca8abd/f5cdc44f8]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-00b98d913067ee2a5]
aws_lb_listener.external_elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:058264135441:listener/app/external-lb-new/e72035a51f2bf417/62e25b66840962b0]
aws_lb_target_group_attachment.attachment[0]: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:058264135441:targetgroup/alb-tg/3c72856b97825104-20241117131041495200000003]
aws_lb_target_group_attachment.attachment[1]: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:058264135441:targetgroup/alb-tg/3c72856b97825104-20241117131041605600000004]

Changes to Outputs:
+ lb_dns_name = "external-lb-new-27305926.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-172-31-10-17 terraform-project-1]# terraform apply
aws_vpc.demoVPC: Refreshing state... [id=vpc-036be02c51b08eba2]
aws_subnet.application-subnet-2: Refreshing state... [id=subnet-01df1122b3c8bf15b]
aws_subnet.database-subnet-1: Refreshing state... [id=subnet-0d02aa45e0ebd5a9e]
aws_subnet.application-subnet-1: Refreshing state... [id=subnet-011a0283340568cca]
aws_subnet.public_subnet-2: Refreshing state... [id=subnet-0c0324bb7dee0b38a]
aws_internet_gateway.demogateway: Refreshing state... [id=igw-0764b91efef9cdabdd]
aws_security_group.demosg: Refreshing state... [id=sg-00e81fa30f862db1a]
aws_subnet.public_subnet-1: Refreshing state... [id=subnet-021e45611fbfd965f]
aws_lb_target_group.target_elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:058264135441:targetgroup/alb-tg/3c72856b97825104]
aws_subnet.database-subnet-2: Refreshing state... [id=subnet-0cf9d85d2e1e65b9e]
aws_security_group.database-sg: Refreshing state... [id=sg-0d6c634de080fb650]
aws_route_table.route: Refreshing state... [id=rtb-04fa8b04e07fdb200]
aws_instance.public_subnet-2[0]: Refreshing state... [id=i-0ffcc657a68d27bba7]
aws_route_table_association.rt2: Refreshing state... [id=rtbassoc-00b98d913067ee2a5]
aws_lb.external_lb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:058264135441:loadbalancer/app/external-lb-new/e72035a51f2bf417]
aws_route_table_association.rt1: Refreshing state... [id=rtbassoc-0ca8abd/f5cdc44f8]
aws_instance.demoInstance[0]: Refreshing state... [id=i-050377de78efefaa8]
aws_instance.public_subnet-1[0]: Refreshing state... [id=i-0ec4f42dbcb202b33]
aws_instance.demoInstance[1]: Refreshing state... [id=i-0a341ff4e737ac008]
aws_lb_listener.external_elb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:058264135441:listener/app/external-lb-new/e72035a51f2bf417/62e25b66840962b0]
aws_lb_target_group_attachment.attachment[0]: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:058264135441:targetgroup/alb-tg/3c72856b97825104-20241117131041495200000003]
aws_lb_target_group_attachment.attachment[1]: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:058264135441:targetgroup/alb-tg/3c72856b97825104-20241117131041605600000004]

Changes to Outputs:
+ lb_dns_name = "external-lb-new-27305926.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:
```

# Outputs.tf file

```
1  output "lb_dns_name" {  
2      description = "The DNS Name of the Load Balancer"  
3      value = "${aws_lb.external_alb.dns_name}"  
4  }
```

# Step-11:Variables

In this Terraform configuration, the variables define 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

```
1  variable "vpc_cidr" {  
2    default = "10.0.0.0/16"  
3  }  
4  
5  variable "subnet1_cidr" {  
6    default = "10.0.1.0/24"  
7  }  
8  
9  variable "subnet2_cidr" {  
10   default = "10.0.2.0/24"  
11 }  
12 |  
13 variable "subnet3_cidr" {  
14   default = "10.0.3.0/24"  
15 }  
16  
17 variable "subnet4_cidr" {  
18   default = "10.0.4.0/24"  
19 }  
20  
21 variable "subnet5_cidr" {  
22   default = "10.0.5.0/24"  
23 }  
24  
25
```

# 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