PROJECT - 03

AWS MULTI-TIER ARCHITECTURE USING TERRAFORM

Presented by:

ATCHYUTHA SAI SOWMYA
Mail ID: sowmyaatchyutha7@gmail.com

Batch: 140-ONLINE

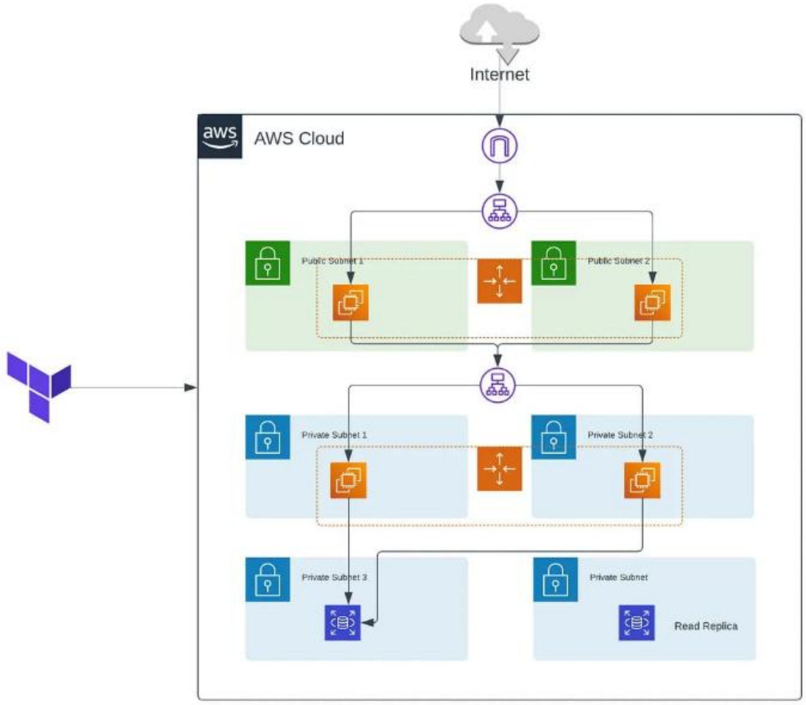


Fig: 3-TIER ARCHITECTURE USING TERRAFORM

Overview:

Terraform is an Infrastructure as Code (IaC) tool developed by HashiCorp widely used for provisioning and managing cloud and on-premise resources including those on Amazon Web Services (AWS). It allows users to define their infrastructure in human-readable configuration files using HashiCorp Configuration Language (HCL) which can then be versioned, reused and shared.

Deploying three-tier architecture on AWS using Terraform is a powerful way to build scalable, secure and modular cloud applications. The 3-tier architecture is a modular design pattern that separates an application into three distinct layers: Presentation**,** Application and Data.

1. Presentation Tier (Frontend Layer)

- It manages user interaction and delivers content
- It is typically placed in **public subnets** for internet accessibility
- It collects user input, validates data and formats the output received from the application tier before displaying it to the user

2. Application Tier (Business Logic Layer)

- **It p**rocesses user requests and enforces business rules
- It is located in **private subnets**, shielded from direct internet access
- It interacts with the data tier to retrieve, store, and manipulate data based on the processed requests

3. Data Tier (Storage Layer)

- **It** stores and retrieves structured or unstructured data
- It interacts primarily with the application layer
- It is the lowest tier and typically consists of database services and related components that manage data

Prerequisites:

- AWS Account
- AWS Access and Secret Key.
- Configure IAM user on Visual Studio Code.
- Terraform version- **v1.13.4**

PROCEDURE:

- Install Terraform
- Configure AWS CLI (aws configure)
- Create a project folder with subfolders in vscode or Gitbash.

STEP-1:

Create subfolders in terraform folder

1. Create terraformblock.tf file using the code below:

```
terraform {
  required providers {
    aws = {
      source = "hashicorp/aws"
      version = "6.17.0"
    }
  }
}
```

2. Create provider.tf file using the code below:

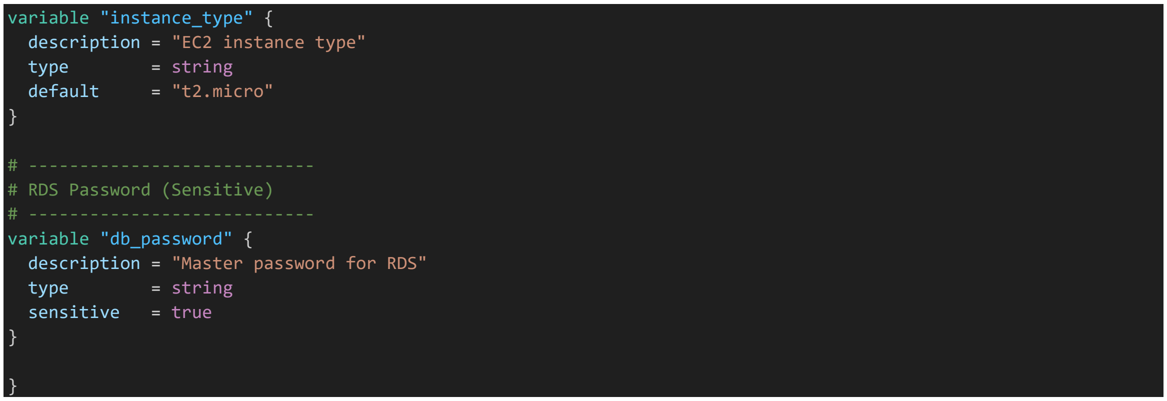
```
provider "aws" {
    region = "us-east-1"
}
```

3. Create vpc.tf file using the code below:

```
resource "aws_vpc" "main" {
  cidr_block           = var.vpc_cidr
  enable_dns_support   = true
  enable_dns_hostnames = true
  instance_tenancy     = "default"

  tags = {
    Name = "3tier-vpc"
  }
}
```

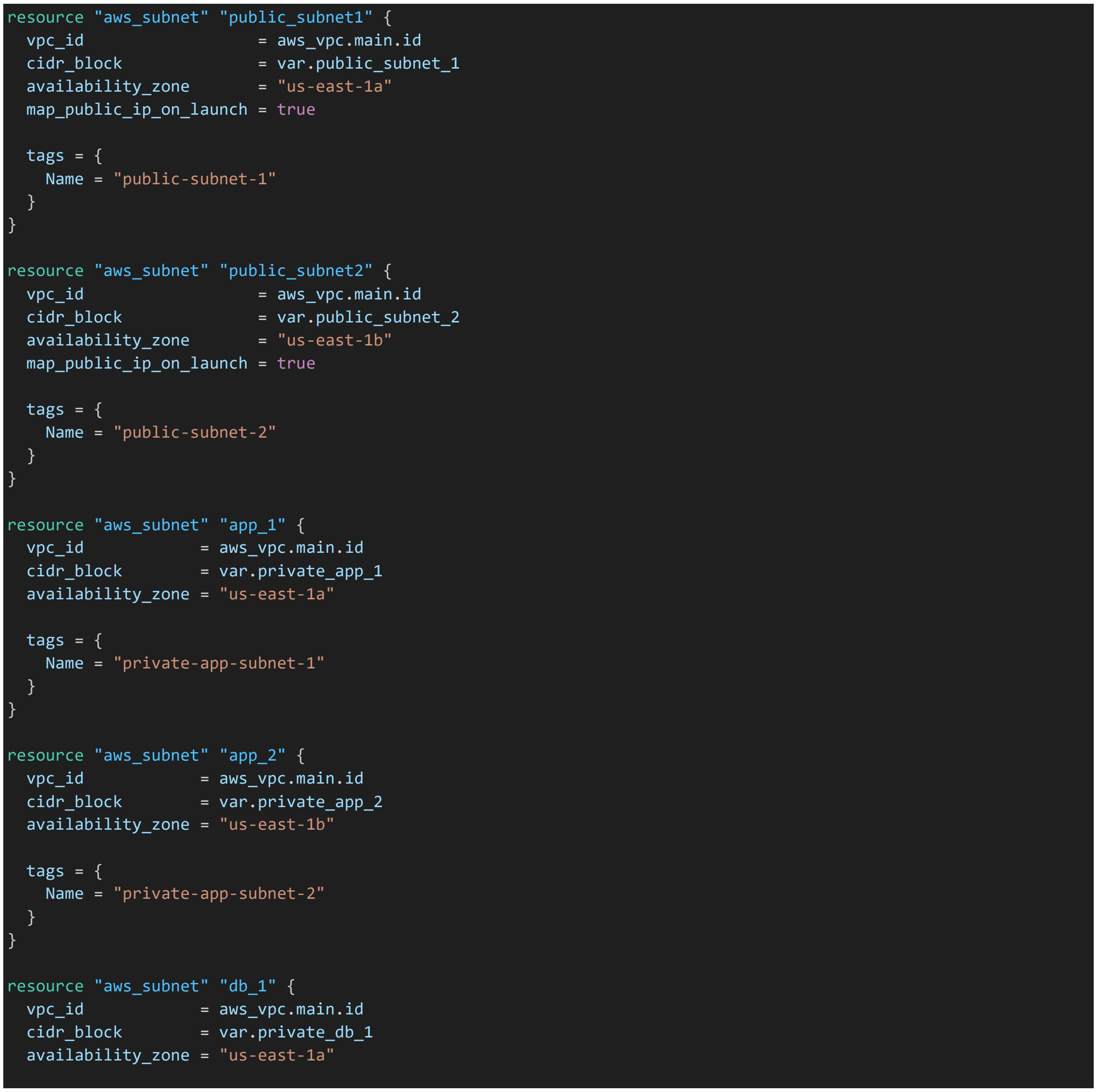
4. Create variable.tf file using the code below:

```
# ---------------------------
# VPC & Subnet CIDRs
# ---------------------------
variable "vpc_cidr" {
  description = "CIDR block for the VPC"
  type        = string
  default     = "10.0.0.0/16"
}

variable "public_subnet_1" {
  description = "CIDR block for public subnet 1"
  type        = string
  default     = "10.0.1.0/24"
}

variable "public_subnet_2" {
  description = "CIDR block for public subnet 2"
  type        = string
  default     = "10.0.2.0/24"
}

variable "private_app_1" {
  description = "CIDR block for private app subnet 1"
  type        = string
  default     = "10.0.5.0/24"
}

variable "private_app_2" {
  description = "CIDR block for private app subnet 2"
  type        = string
  default     = "10.0.6.0/24"
}

variable "private_db_1" {
  description = "CIDR block for private DB subnet 1"
  type        = string
  default     = "10.0.7.0/24"
}

variable "private_db_2" {
  description = "CIDR block for private DB subnet 2"
  type        = string
  default     = "10.0.8.0/24"
}

# ---------------------------
# EC2 Instance Type
# ---------------------------
```

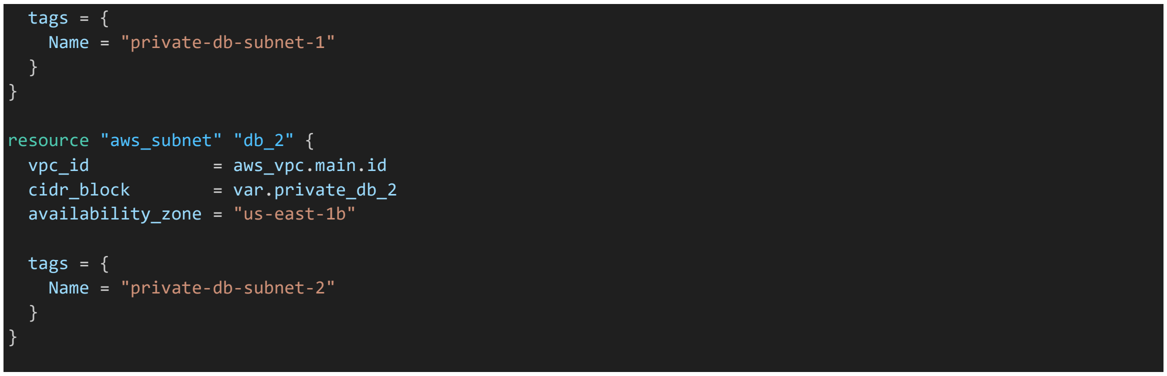


```
variable "instance_type" {
  description = "EC2 instance type"
  type        = string
  default     = "t2.micro"
}

# ----------------------------
# RDS Password (Sensitive)
# ----------------------------
variable "db_password" {
  description = "Master password for RDS"
  type        = string
  sensitive   = true
}

}
```

5. Create subnets.tf file for the public and private subnets using the code below:

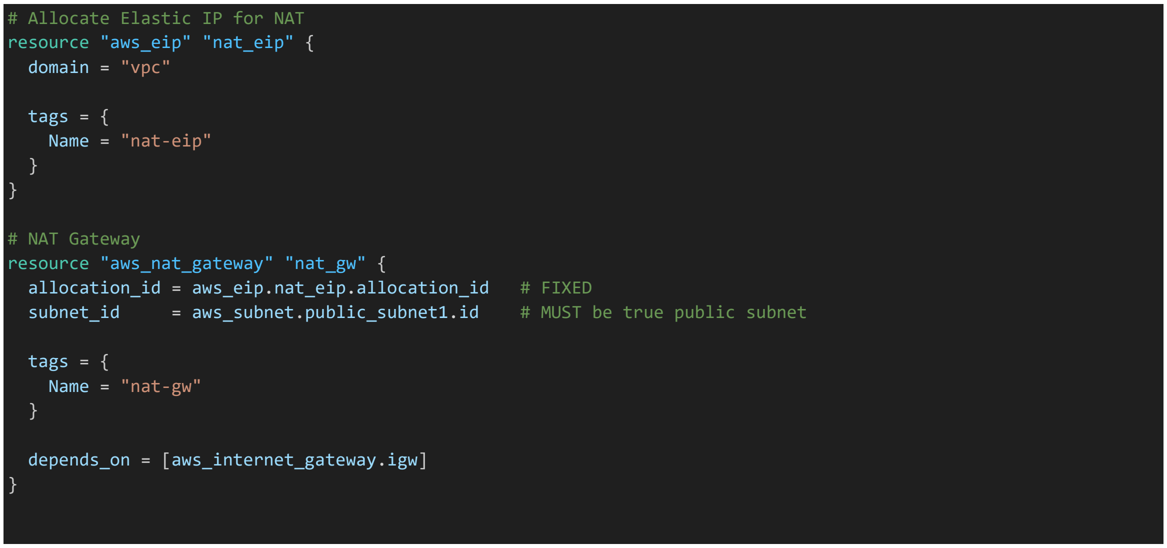


```
resource "aws_subnet" "public_subnet1" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = var.public_subnet_1
  availability_zone       = "us-east-1a"
  map_public_ip_on_launch = true

  tags = {
    Name = "public-subnet-1"
  }
}

resource "aws_subnet" "public_subnet2" {
  vpc_id                  = aws_vpc.main.id
  cidr_block              = var.public_subnet_2
  availability_zone       = "us-east-1b"
  map_public_ip_on_launch = true

  tags = {
    Name = "public-subnet-2"
  }
}

resource "aws_subnet" "app_1" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = var.private_app_1
  availability_zone = "us-east-1a"

  tags = {
    Name = "private-app-subnet-1"
  }
}

resource "aws_subnet" "app_2" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = var.private_app_2
  availability_zone = "us-east-1b"

  tags = {
    Name = "private-app-subnet-2"
  }
}

resource "aws_subnet" "db_1" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = var.private_db_1
  availability_zone = "us-east-1a"
```



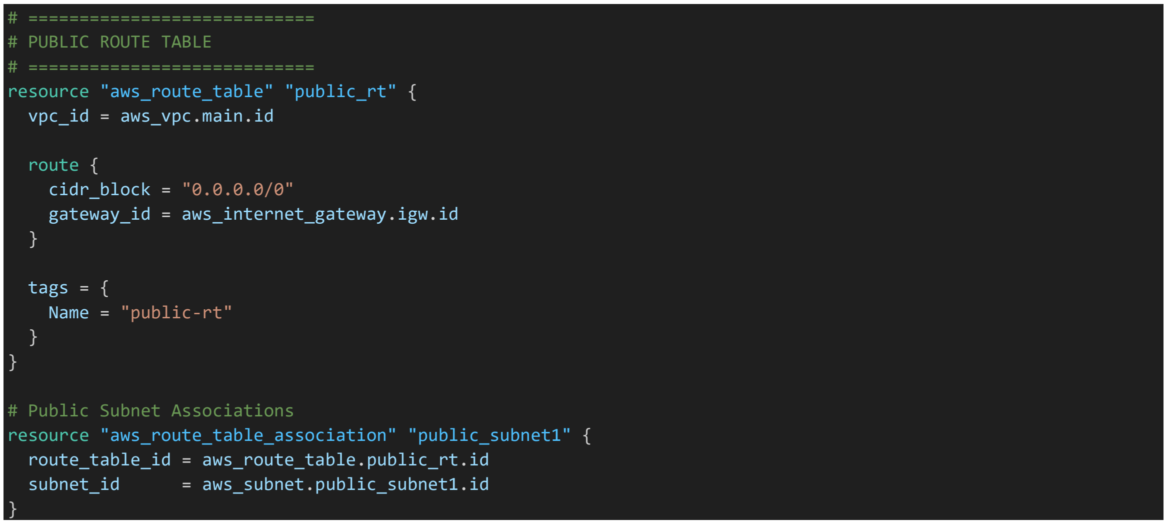
```
  tags = {
    Name = "private-db-subnet-1"
  }
}

resource "aws_subnet" "db_2" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = var.private_db_2
  availability_zone = "us-east-1b"

  tags = {
    Name = "private-db-subnet-2"
  }
}
```

6. Create igw_nat.tf file using the code below:

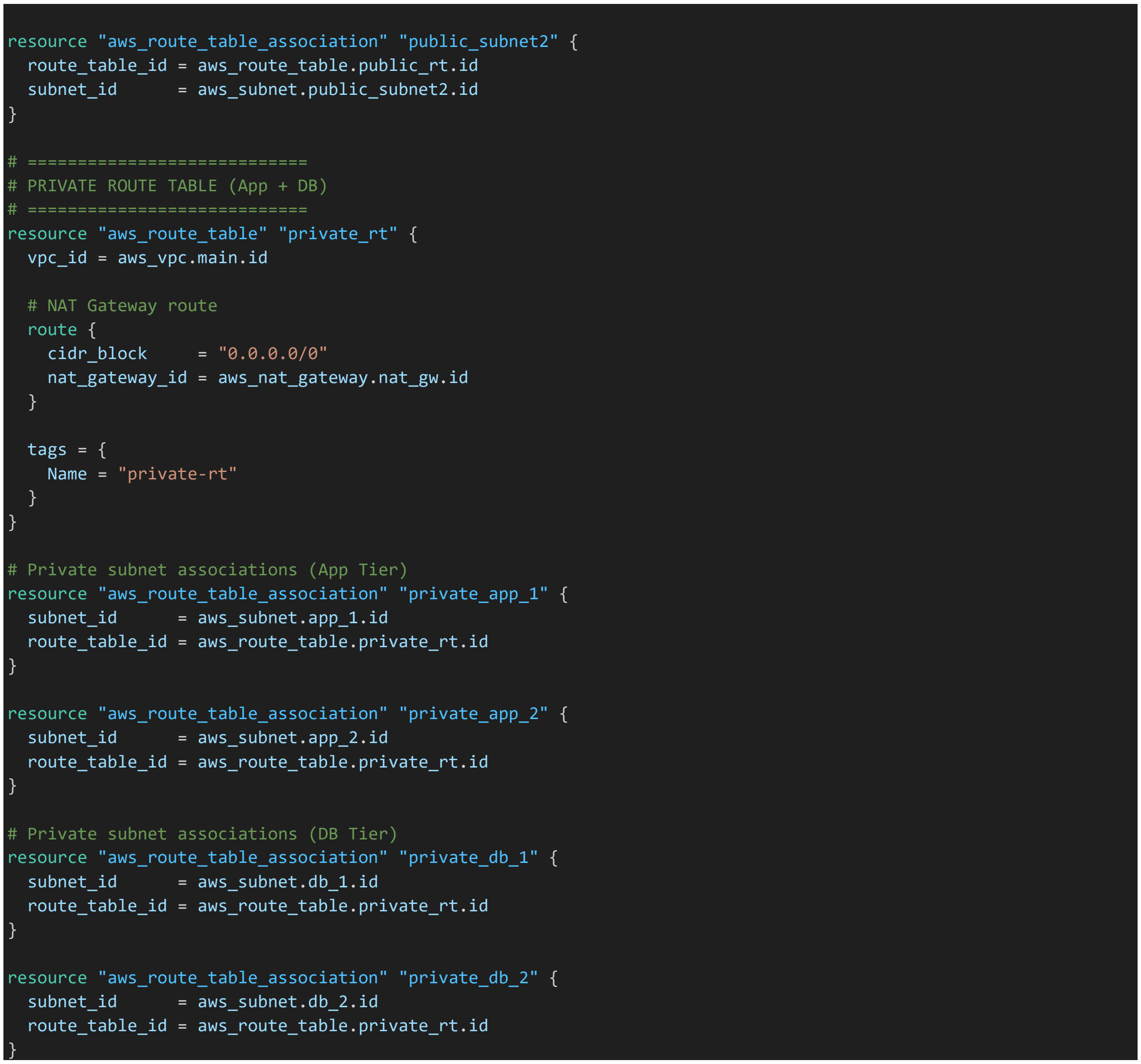


```
# Allocate Elastic IP for NAT
resource "aws_eip" "nat_eip" {
  domain = "vpc"

  tags = {
    Name = "nat-eip"
  }
}

# NAT Gateway
resource "aws_nat_gateway" "nat_gw" {
  allocation_id = aws_eip.nat_eip.allocation_id   # FIXED
  subnet_id     = aws_subnet.public_subnet1.id    # MUST be true public subnet

  tags = {
    Name = "nat-gw"
  }

  depends_on = [aws_internet_gateway.igw]
}
```

7. Create route_tables.tf using the code below:

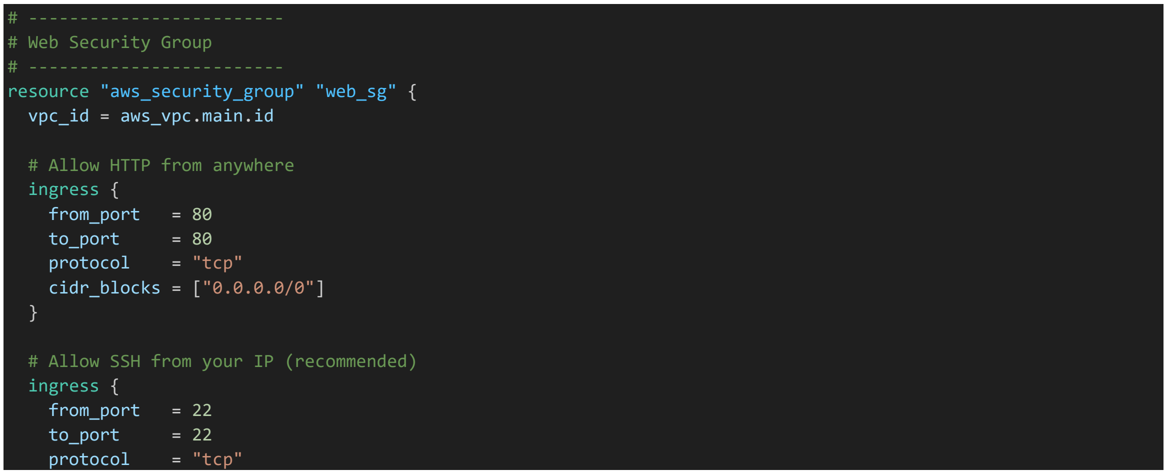


```
# ============================
# PUBLIC ROUTE TABLE
# ============================
resource "aws_route_table" "public_rt" {
  vpc_id = aws_vpc.main.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }

  tags = {
    Name = "public-rt"
  }
}

# Public Subnet Associations
resource "aws_route_table_association" "public_subnet1" {
  route_table_id = aws_route_table.public_rt.id
  subnet_id      = aws_subnet.public_subnet1.id
}
```

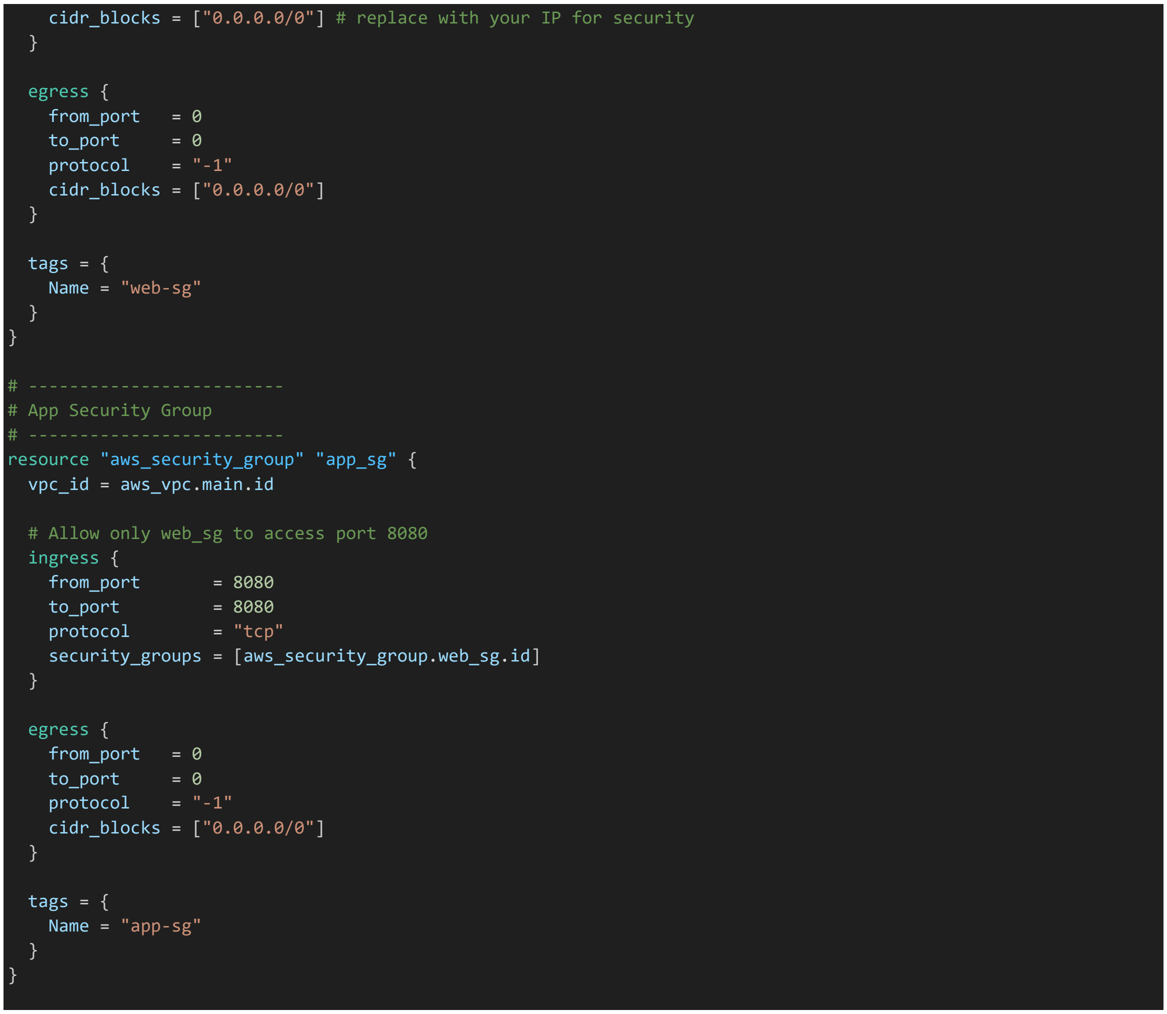


```
resource "aws_route_table_association" "public_subnet2" {
  route_table_id = aws_route_table.public_rt.id
  subnet_id      = aws_subnet.public_subnet2.id
}

# ============================
# PRIVATE ROUTE TABLE (App + DB)
# ============================
resource "aws_route_table" "private_rt" {
  vpc_id = aws_vpc.main.id

  # NAT Gateway route
  route {
    cidr_block     = "0.0.0.0/0"
    nat_gateway_id = aws_nat_gateway.nat_gw.id
  }

  tags = {
    Name = "private-rt"
  }
}

# Private subnet associations (App Tier)
resource "aws_route_table_association" "private_app_1" {
  subnet_id      = aws_subnet.app_1.id
  route_table_id = aws_route_table.private_rt.id
}

resource "aws_route_table_association" "private_app_2" {
  subnet_id      = aws_subnet.app_2.id
  route_table_id = aws_route_table.private_rt.id
}

# Private subnet associations (DB Tier)
resource "aws_route_table_association" "private_db_1" {
  subnet_id      = aws_subnet.db_1.id
  route_table_id = aws_route_table.private_rt.id
}

resource "aws_route_table_association" "private_db_2" {
  subnet_id      = aws_subnet.db_2.id
  route_table_id = aws_route_table.private_rt.id
}
```

8. Create security_groups.tf file using the code below:

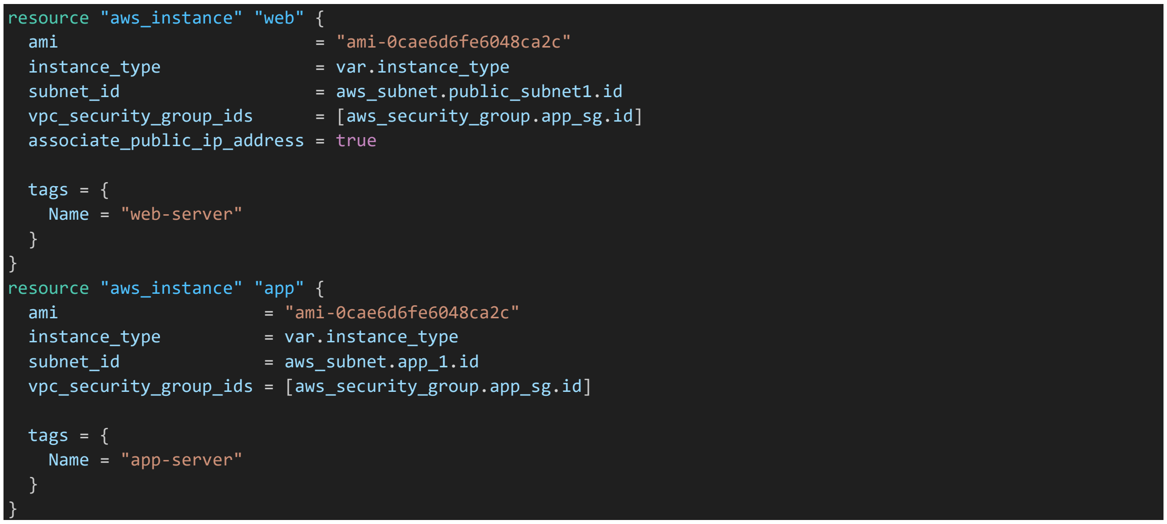


```
# -------------------------
# Web Security Group
# -------------------------
resource "aws_security_group" "web_sg" {
  vpc_id = aws_vpc.main.id

  # Allow HTTP from anywhere
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # Allow SSH from your IP (recommended)
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
```



```
    cidr_blocks = ["0.0.0.0/0"] # replace with your IP for security
  }

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

  tags = {
    Name = "web-sg"
  }
}

# ------------------------
# App Security Group
# ------------------------
resource "aws_security_group" "app_sg" {
  vpc_id = aws_vpc.main.id

  # Allow only web_sg to access port 8080
  ingress {
    from_port       = 8080
    to_port         = 8080
    protocol        = "tcp"
    security_groups = [aws_security_group.web_sg.id]
  }

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

  tags = {
    Name = "app-sg"
  }
}
```

9. Create ec2_app_web.tf using the code below:



```
resource "aws_instance" "web" {
  ami                         = "ami-0cae6d6fe6048ca2c"
  instance_type               = var.instance_type
  subnet_id                   = aws_subnet.public_subnet1.id
  vpc_security_group_ids      = [aws_security_group.app_sg.id]
  associate_public_ip_address = true

  tags = {
    Name = "web-server"
  }
}
resource "aws_instance" "app" {
  ami                    = "ami-0cae6d6fe6048ca2c"
  instance_type          = var.instance_type
  subnet_id              = aws_subnet.app_1.id
  vpc_security_group_ids = [aws_security_group.app_sg.id]

  tags = {
    Name = "app-server"
  }
}
```

10. Create alb.tf using the code below:

```
resource "aws_lb" "alb" {
  name               = "web-alb"
  load_balancer_type = "application"
  internal           = false               # ALB must be external for internet
  security_groups    = [aws_security_group.app_sg.id]

  subnets = [
    aws_subnet.public_subnet1.id,
    aws_subnet.public_subnet2.id
  ]

  tags = {
    Name = "web-alb"
  }
}

resource "aws_lb_target_group" "tg" {
  name     = "web-tg"
  port     = 80
  protocol = "HTTP"
  vpc_id   = aws_vpc.main.id

  health_check {
    path                = "/"
    interval            = 30
    timeout             = 5
    healthy_threshold   = 2
    unhealthy_threshold = 2
  }

  tags = {
    Name = "web-tg"
  }
}

resource "aws_lb_listener" "listener" {
  load_balancer_arn = aws_lb.alb.arn
  port              = 80
  protocol          = "HTTP"

  default_action {
    type             = "forward"
    target_group_arn = aws_lb_target_group.tg.arn
  }
}

resource "aws_lb_target_group_attachment" "web_attachment" {
  target_group_arn = aws_lb_target_group.tg.arn
  target_id        = aws_instance.web.id
  port             = 80
}
```

11. Create rds.tf using the code below:

```
resource "aws_security_group" "db_sg" {
  vpc_id = aws_vpc.main.id

  ingress {
    from_port       = 3306
    to_port         = 3306
    protocol        = "tcp"
    # Allow DB access ONLY from APP Layer
    security_groups = [aws_security_group.app_sg.id]
  }

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

  tags = {
    Name = "db-sg"
  }
}

}
```

12. Create outputs.tf using the code below:

```
output "alb_dns" {
  value = aws_lb.alb.dns_name
}
```

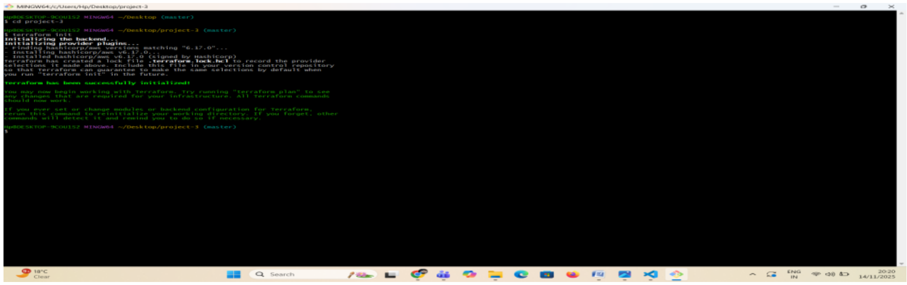
13. Create terraform.tfvars using the code below:

db_password = "Admin12345"

Step 2: Terraform commands:

By using all these commands we can create vpc and ec2 and their components.

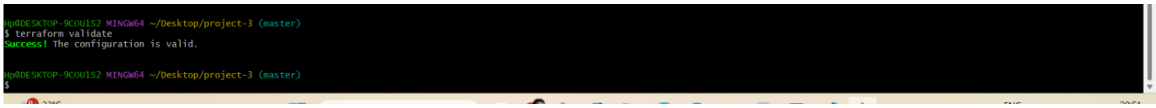
1. terraform init:



MINGW64:/c/Users/Hp/Desktop/project-3

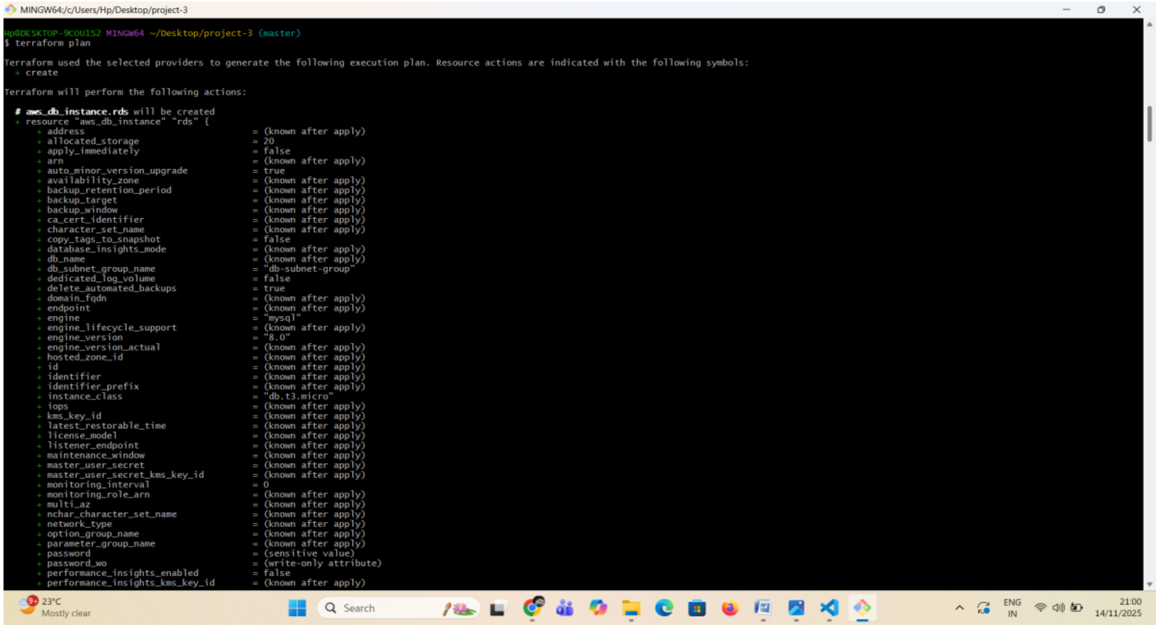
```
Hp@DESKTOP-9COU1S2 MINGW64 ~/Desktop (master)
$ cd project-3

Hp@DESKTOP-9COU1S2 MINGW64 ~/Desktop/project-3 (master)
$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "6.17.0"...
- Installing hashicorp/aws v6.17.0...
- Installed hashicorp/aws v6.17.0 (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.

Hp@DESKTOP-9COU1S2 MINGW64 ~/Desktop/project-3 (master)
$
```

2. terraform validate:



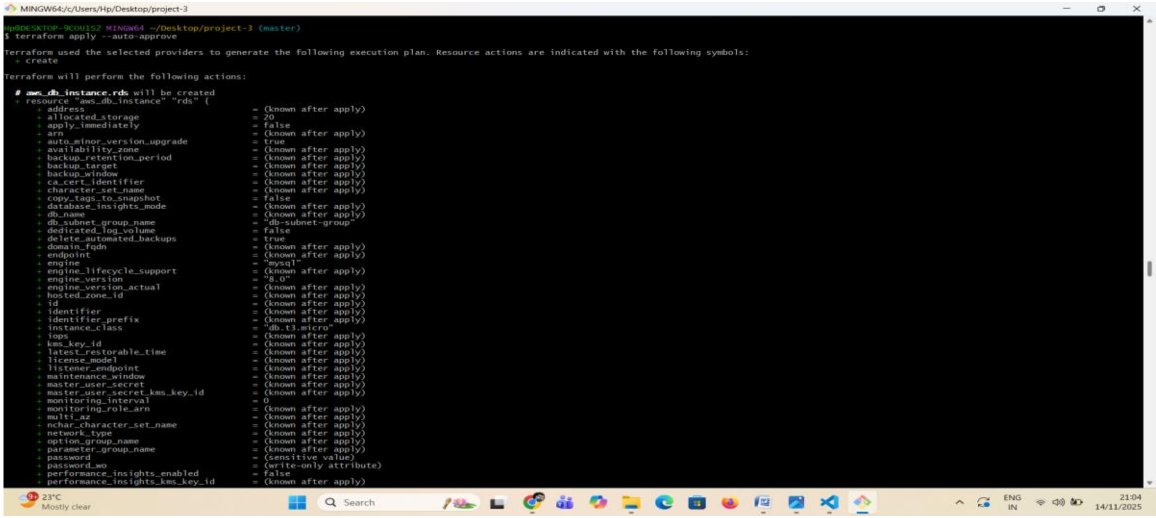
```
Hp@DESKTOP-9COU1S2 MINGW64 ~/Desktop/project-3 (master)
$ terraform validate
Success! The configuration is valid.

Hp@DESKTOP-9COU1S2 MINGW64 ~/Desktop/project-3 (master)
$
```

3. terraform plan:

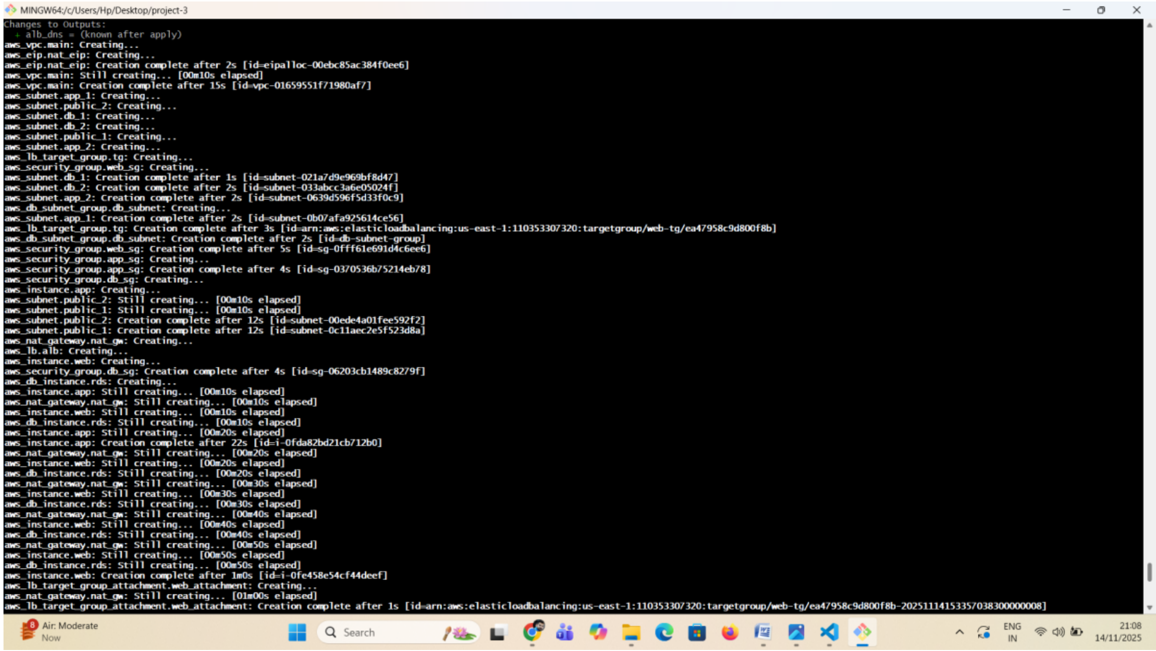


```
Hp@DESKTOP-9COU1S2 MINGW64 ~/Desktop/project-3 (master)
$ terraform plan

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

Terraform will perform the following actions:

  # aws_db_instance.rds will be created
  + resource "aws_db_instance" "rds" {
      + address                               = (known after apply)
      + allocated_storage                     = 20
      + apply_immediately                     = false
      + arn                                   = (known after apply)
      + auto_minor_version_upgrade            = true
      + availability_zone                     = (known after apply)
      + backup_retention_period               = (known after apply)
      + backup_target                         = (known after apply)
      + backup_window                         = (known after apply)
      + ca_cert_identifier                    = (known after apply)
      + character_set_name                    = (known after apply)
      + copy_tags_to_snapshot                 = false
      + database_insights_mode                = (known after apply)
      + db_name                               = (known after apply)
      + db_subnet_group_name                  = "db-subnet-group"
      + dedicated_log_volume                  = false
      + delete_automated_backups              = true
      + domain_fqdn                           = (known after apply)
      + endpoint                              = (known after apply)
      + engine                                = "mysql"
      + engine_lifecycle_support              = (known after apply)
      + engine_version                        = "8.0"
      + engine_version_actual                 = (known after apply)
      + hosted_zone_id                        = (known after apply)
      + id                                    = (known after apply)
      + identifier                            = (known after apply)
      + identifier_prefix                     = (known after apply)
      + instance_class                        = "db.t3.micro"
      + iops                                  = (known after apply)
      + kms_key_id                            = (known after apply)
      + latest_restorable_time                = (known after apply)
      + license_model                         = (known after apply)
      + listener_endpoint                     = (known after apply)
      + maintenance_window                    = (known after apply)
      + master_user_secret                    = (known after apply)
      + master_user_secret_kms_key_id         = (known after apply)
      + monitoring_interval                   = 0
      + monitoring_role_arn                   = (known after apply)
      + multi_az                              = (known after apply)
      + nchar_character_set_name              = (known after apply)
      + network_type                          = (known after apply)
      + option_group_name                     = (known after apply)
      + parameter_group_name                  = (known after apply)
      + password                              = (sensitive value)
      + password_wo                           = (write-only attribute)
      + performance_insights_enabled          = false
      + performance_insights_kms_key_id       = (known after apply)
```

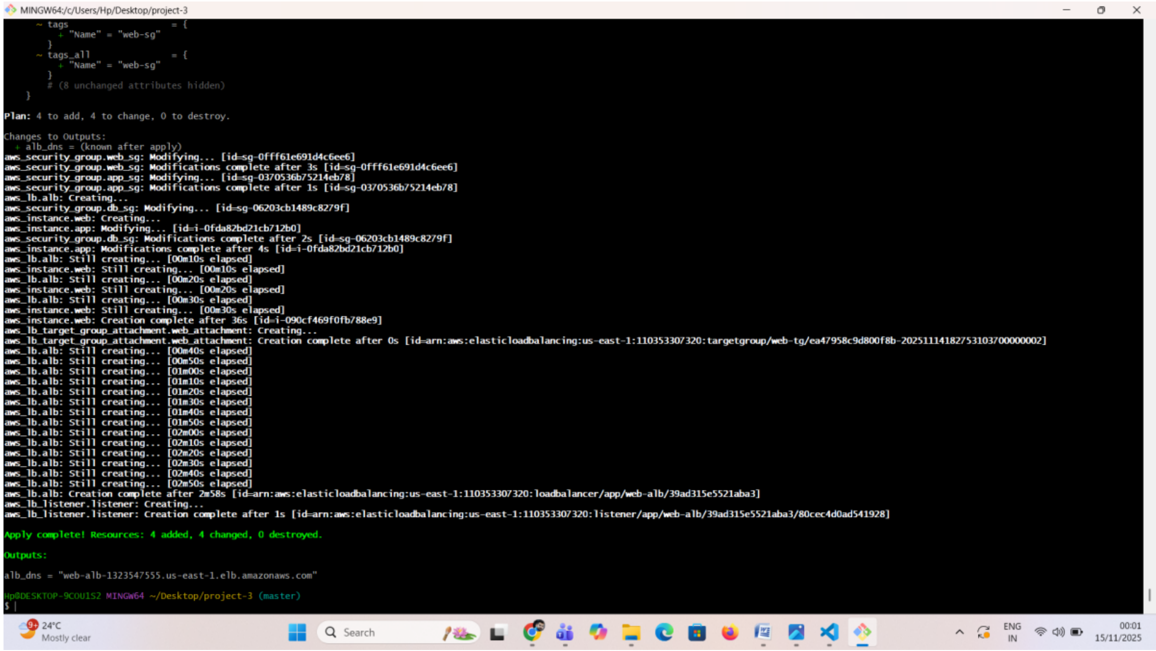
4. terraform apply:



```
Hp@DESKTOP-9COU1S2 MINGW64 ~/Desktop/project-3 (master)
$ terraform apply --auto-approve

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

Terraform will perform the following actions:

  # aws_db_instance.rds will be created
  + resource "aws_db_instance" "rds" {
      + address                               = (known after apply)
      + allocated_storage                     = 20
      + apply_immediately                     = false
      + arn                                   = (known after apply)
      + auto_minor_version_upgrade            = true
      + availability_zone                     = (known after apply)
      + backup_retention_period               = (known after apply)
      + backup_target                         = (known after apply)
      + backup_window                         = (known after apply)
      + ca_cert_identifier                    = (known after apply)
      + character_set_name                    = (known after apply)
      + copy_tags_to_snapshot                 = false
      + database_insights_mode                = (known after apply)
      + db_name                               = (known after apply)
      + db_subnet_group_name                  = "db-subnet-group"
      + dedicated_log_volume                  = false
      + delete_automated_backups              = true
      + domain_fqdn                           = (known after apply)
      + endpoint                              = (known after apply)
      + engine                                = "mysql"
      + engine_lifecycle_support              = (known after apply)
      + engine_version                        = "8.0"
      + engine_version_actual                 = (known after apply)
      + hosted_zone_id                        = (known after apply)
      + id                                    = (known after apply)
      + identifier                            = (known after apply)
      + identifier_prefix                     = (known after apply)
      + instance_class                        = "db.t3.micro"
      + iops                                  = (known after apply)
      + kms_key_id                            = (known after apply)
      + latest_restorable_time                = (known after apply)
      + license_model                         = (known after apply)
      + listener_endpoint                     = (known after apply)
      + maintenance_window                    = (known after apply)
      + master_user_secret                    = (known after apply)
      + master_user_secret_kms_key_id         = (known after apply)
      + monitoring_interval                   = 0
      + monitoring_role_arn                   = (known after apply)
      + multi_az                              = (known after apply)
      + nchar_character_set_name              = (known after apply)
      + network_type                          = (known after apply)
      + option_group_name                     = (known after apply)
      + parameter_group_name                  = (known after apply)
      + password                              = (sensitive value)
      + password_wo                           = (write-only attribute)
      + performance_insights_enabled          = false
      + performance_insights_kms_key_id       = (known after apply)
```



```
Changes to Outputs:
  + alb_dns = (known after apply)
aws_vpc.main: Creating...
aws_eip.nat_eip: Creating...
aws_eip.nat_eip: Creation complete after 2s [id=eipalloc-00ebc85ac384f0ee6]
aws_vpc.main: Still creating... [00m10s elapsed]
aws_vpc.main: Creation complete after 15s [id=vpc-01659551f71980af7]
aws_subnet.app_1: Creating...
aws_subnet.public_2: Creating...
aws_subnet.db_1: Creating...
aws_subnet.db_2: Creating...
aws_subnet.public_1: Creating...
aws_subnet.app_2: Creating...
aws_lb_target_group.tg: Creating...
aws_security_group.web_sg: Creating...
aws_subnet.db_1: Creation complete after 1s [id=subnet-021a7d9e969bf8d47]
aws_subnet.db_2: Creation complete after 2s [id=subnet-033abcc3a6e05024f]
aws_subnet.app_2: Creation complete after 2s [id=subnet-0639d596f5d33f0c9]
aws_db_subnet_group.db_subnet: Creating...
aws_subnet.app_1: Creation complete after 2s [id=subnet-0b07afa925614ce56]
aws_lb_target_group.tg: Creation complete after 3s [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:targetgroup/web-tg/ea47958c9d800f8b]
aws_db_subnet_group.db_subnet: Creation complete after 2s [id=db-subnet-group]
aws_security_group.web_sg: Creation complete after 5s [id=sg-0fff61e691d4c6ee6]
aws_security_group.app_sg: Creating...
aws_security_group.app_sg: Creation complete after 4s [id=sg-0370536b75214eb78]
aws_security_group.db_sg: Creating...
aws_instance.app: Creating...
aws_subnet.public_2: Still creating... [00m10s elapsed]
aws_subnet.public_1: Still creating... [00m10s elapsed]
aws_subnet.public_2: Creation complete after 12s [id=subnet-00ede4a01fee592f2]
aws_subnet.public_1: Creation complete after 12s [id=subnet-0c11aec2e5f523d8a]
aws_nat_gateway.nat_gw: Creating...
aws_lb.alb: Creating...
aws_instance.web: Creating...
aws_security_group.db_sg: Creation complete after 4s [id=sg-06203cb1489c8279f]
aws_db_instance.rds: Creating...
aws_instance.app: Still creating... [00m10s elapsed]
aws_nat_gateway.nat_gw: Still creating... [00m10s elapsed]
aws_instance.web: Still creating... [00m10s elapsed]
aws_db_instance.rds: Still creating... [00m10s elapsed]
aws_instance.app: Still creating... [00m20s elapsed]
aws_instance.app: Creation complete after 22s [id=i-0fda82bd21cb712b0]
aws_nat_gateway.nat_gw: Still creating... [00m20s elapsed]
aws_instance.web: Still creating... [00m20s elapsed]
aws_db_instance.rds: Still creating... [00m20s elapsed]
aws_nat_gateway.nat_gw: Still creating... [00m30s elapsed]
aws_instance.web: Still creating... [00m30s elapsed]
aws_db_instance.rds: Still creating... [00m30s elapsed]
aws_nat_gateway.nat_gw: Still creating... [00m40s elapsed]
aws_instance.web: Still creating... [00m40s elapsed]
aws_db_instance.rds: Still creating... [00m40s elapsed]
aws_nat_gateway.nat_gw: Still creating... [00m50s elapsed]
aws_instance.web: Still creating... [00m50s elapsed]
aws_db_instance.rds: Still creating... [00m50s elapsed]
aws_instance.web: Creation complete after 1m0s [id=i-0fe458e54cf44deef]
aws_lb_target_group_attachment.web_attachment: Creating...
aws_nat_gateway.nat_gw: Still creating... [01m00s elapsed]
aws_lb_target_group_attachment.web_attachment: Creation complete after 1s [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:targetgroup/web-tg/ea47958c9d800f8b-20251114153357038300000008]
```

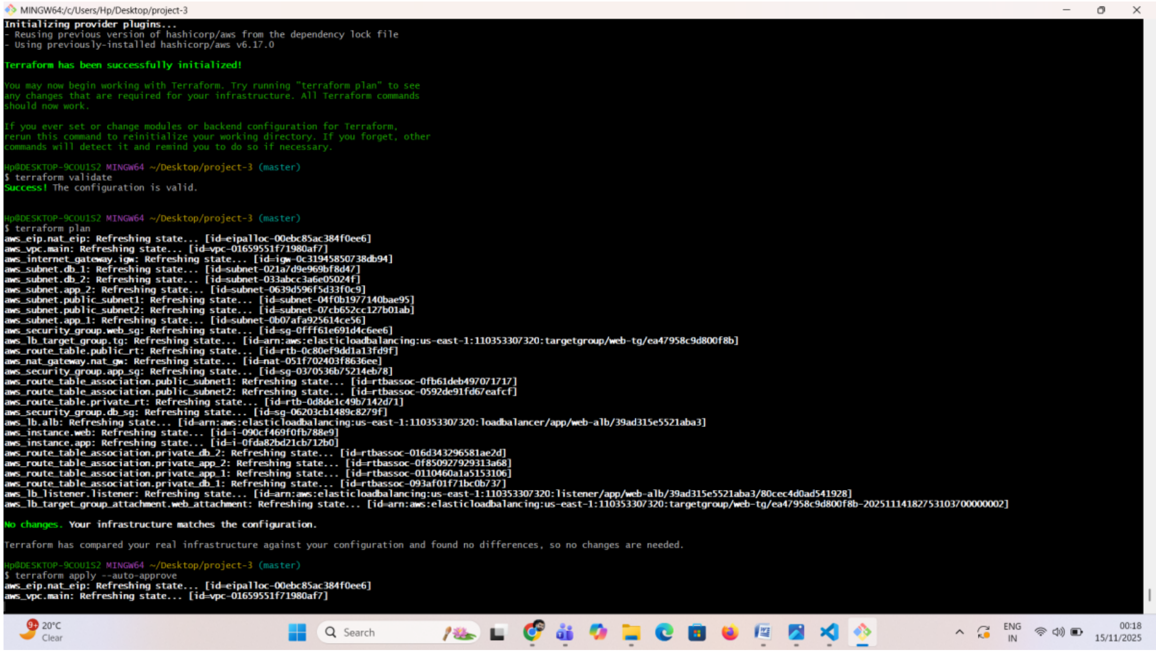


```
      ~ tags                   = {
          + "Name" = "web-sg"
        }
      ~ tags_all               = {
          + "Name" = "web-sg"
        }
        # (8 unchanged attributes hidden)
    }

Plan: 4 to add, 4 to change, 0 to destroy.

Changes to Outputs:
  + alb_dns = (known after apply)
aws_security_group.web_sg: Modifying... [id=sg-0fff61e691d4c6ee6]
aws_security_group.web_sg: Modifications complete after 3s [id=sg-0fff61e691d4c6ee6]
aws_security_group.app_sg: Modifying... [id=sg-0370536b75214eb78]
aws_security_group.app_sg: Modifications complete after 1s [id=sg-0370536b75214eb78]
aws_lb.alb: Creating...
aws_security_group.db_sg: Modifying... [id=sg-06203cb1489c8279f]
aws_instance.web: Creating...
aws_instance.app: Modifying... [id=i-0fda82bd21cb712b0]
aws_security_group.db_sg: Modifications complete after 2s [id=sg-06203cb1489c8279f]
aws_instance.app: Modifications complete after 4s [id=i-0fda82bd21cb712b0]
aws_lb.alb: Still creating... [00m10s elapsed]
aws_instance.web: Still creating... [00m10s elapsed]
aws_lb.alb: Still creating... [00m20s elapsed]
aws_instance.web: Still creating... [00m20s elapsed]
aws_lb.alb: Still creating... [00m30s elapsed]
aws_instance.web: Still creating... [00m30s elapsed]
aws_instance.web: Creation complete after 36s [id=i-090cf469f0fb788e9]
aws_lb_target_group_attachment.web_attachment: Creating...
aws_lb_target_group_attachment.web_attachment: Creation complete after 0s [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:targetgroup/web-tg/ea47958c9d800f8b-20251114182753103700000002]
aws_lb.alb: Still creating... [00m40s elapsed]
aws_lb.alb: Still creating... [00m50s elapsed]
aws_lb.alb: Still creating... [01m00s elapsed]
aws_lb.alb: Still creating... [01m10s elapsed]
aws_lb.alb: Still creating... [01m20s elapsed]
aws_lb.alb: Still creating... [01m30s elapsed]
aws_lb.alb: Still creating... [01m40s elapsed]
aws_lb.alb: Still creating... [01m50s elapsed]
aws_lb.alb: Still creating... [02m00s elapsed]
aws_lb.alb: Still creating... [02m10s elapsed]
aws_lb.alb: Still creating... [02m20s elapsed]
aws_lb.alb: Still creating... [02m30s elapsed]
aws_lb.alb: Still creating... [02m40s elapsed]
aws_lb.alb: Still creating... [02m50s elapsed]
aws_lb.alb: Creation complete after 2m58s [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:loadbalancer/app/web-alb/39ad315e5521aba3]
aws_lb_listener.listener: Creating...
aws_lb_listener.listener: Creation complete after 1s [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:listener/app/web-alb/39ad315e5521aba3/80cec4d0ad541928]

Apply complete! Resources: 4 added, 4 changed, 0 destroyed.

Outputs:

alb_dns = "web-alb-1323547555.us-east-1.elb.amazonaws.com"

Hp@DESKTOP-9COU1S2 MINGW64 ~/Desktop/project-3 (master)
$ |
```

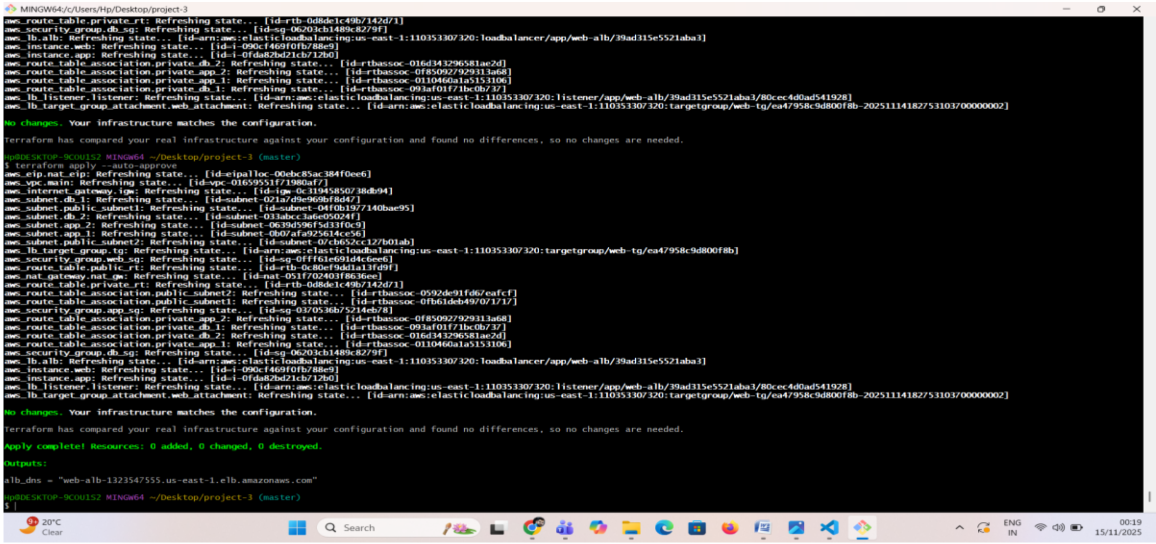
14. Create a file for the user data: (install-apache.sh)

```
#!/bin/bash

# Update system
yum update -y

# Install Apache Web Server
```

```
yum install -y httpd

# Start and enable Apache
systemctl start httpd
systemctl enable httpd

# Create sample webpage
echo "<html>
<body>
<h1>WEB TIER SUCCESS</h1>
</body>
</html>" > /var/www/html/index.html
```



```
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.17.0

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.

Hp@DESKTOP-9COU1S2 MINGW64 ~/Desktop/project-3 (master)
$ terraform validate
Success! The configuration is valid.


Hp@DESKTOP-9COU1S2 MINGW64 ~/Desktop/project-3 (master)
$ terraform plan
aws_eip.nat_eip: Refreshing state... [id=eipalloc-00ebc85ac384f0ee6]
aws_vpc.main: Refreshing state... [id=vpc-01659551f71980af7]
aws_internet_gateway.igw: Refreshing state... [id=igw-0c319458507 38db94]
aws_subnet.db_1: Refreshing state... [id=subnet-021a7d9e969bf8d47]
aws_subnet.db_2: Refreshing state... [id=subnet-033abcc3a6e05024f]
aws_subnet.app_2: Refreshing state... [id=subnet-0639d596f5d33f0c9]
aws_subnet.public_subnet1: Refreshing state... [id=subnet-04f0b1977140bae95]
aws_subnet.public_subnet2: Refreshing state... [id=subnet-07cb652cc127b01ab]
aws_subnet.app_1: Refreshing state... [id=subnet-0b07afa925614ce56]
aws_security_group.web_sg: Refreshing state... [id=sg-0fff61e691d4c6ee6]
aws_lb_target_group.tg: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:targetgroup/web-tg/ea47958c9d800f8b]
aws_route_table.public_rt: Refreshing state... [id=rtb-0c80ef9dd1a13fd9f]
aws_nat_gateway.nat_gw: Refreshing state... [id=nat-051f702403f8636ee]
aws_security_group.app_sg: Refreshing state... [id=sg-0370536b75214eb78]
aws_route_table_association.public_subnet1: Refreshing state... [id=rtbassoc-0fb61deb497071717]
aws_route_table_association.public_subnet2: Refreshing state... [id=rtbassoc-0592de91fd67eafcf]
aws_route_table.private_rt: Refreshing state... [id=rtb-0d8de1c49b7142d71]
aws_security_group.db_sg: Refreshing state... [id=sg-06203cb1489c8279f]
aws_lb.alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:loadbalancer/app/web-alb/39ad315e5521aba3]
aws_instance.web: Refreshing state... [id=i-090cf469f0fb788e9]
aws_instance.app: Refreshing state... [id=i-0fda82bd21cb712b0]
aws_route_table_association.private_db_2: Refreshing state... [id=rtbassoc-016d343296581ae2d]
aws_route_table_association.private_app_2: Refreshing state... [id=rtbassoc-0f850927929313a68]
aws_route_table_association.private_app_1: Refreshing state... [id=rtbassoc-0110460a1a5153106]
aws_route_table_association.private_db_1: Refreshing state... [id=rtbassoc-093af01f71bc0b737]
aws_lb_listener.listener: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:listener/app/web-alb/39ad315e5521aba3/80cec4d0ad541928]
aws_lb_target_group_attachment.web_attachment: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:targetgroup/web-tg/ea47958c9d800f8b-20251114182753103700000002]

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.

Hp@DESKTOP-9COU1S2 MINGW64 ~/Desktop/project-3 (master)
$ terraform apply --auto-approve
aws_eip.nat_eip: Refreshing state... [id=eipalloc-00ebc85ac384f0ee6]
aws_vpc.main: Refreshing state... [id=vpc-01659551f71980af7]
```

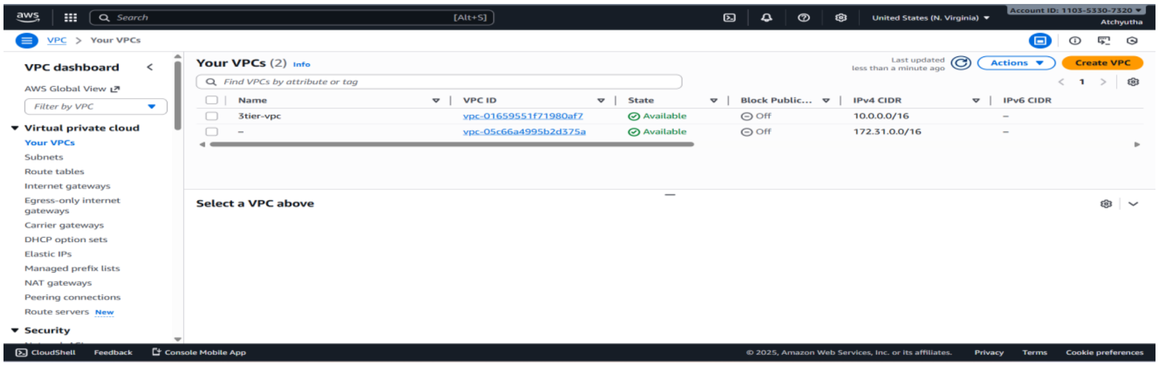


```
aws_route_table.private_rt: Refreshing state... [id=rtb-0d8de1c49b7142d71]
aws_security_group.db_sg: Refreshing state... [id=sg-06203cb1489c8279f]
aws_lb.alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:loadbalancer/app/web-alb/39ad315e5521aba3]
aws_instance.web: Refreshing state... [id=i-090cf469f0fb788e9]
aws_instance.app: Refreshing state... [id=i-0fda82bd21cb712b0]
aws_route_table_association.private_db_2: Refreshing state... [id=rtbassoc-016d343296581ae2d]
aws_route_table_association.private_app_2: Refreshing state... [id=rtbassoc-0f850927929313a68]
aws_route_table_association.private_app_1: Refreshing state... [id=rtbassoc-0110460a1a5153106]
aws_route_table_association.private_db_1: Refreshing state... [id=rtbassoc-093af01f71bc0b737]
aws_lb_listener.listener: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:listener/app/web-alb/39ad315e5521aba3/80cec4d0ad541928]
aws_lb_target_group_attachment.web_attachment: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:targetgroup/web-tg/ea47958c9d800f8b-20251114182753103700000002]

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.

Hp@DESKTOP-9COU1S2 MINGW64 ~/Desktop/project-3 (master)
$ terraform apply --auto-approve
aws_eip.nat_eip: Refreshing state... [id=eipalloc-00ebc85ac384f0ee6]
aws_vpc.main: Refreshing state... [id=vpc-01659551f71980af7]
aws_internet_gateway.igw: Refreshing state... [id=igw-0c319458507 38db94]
aws_subnet.db_1: Refreshing state... [id=subnet-021a7d9e969bf8d47]
aws_subnet.public_subnet1: Refreshing state... [id=subnet-04f0b1977140bae95]
aws_subnet.db_2: Refreshing state... [id=subnet-033abcc3a6e05024f]
aws_subnet.app_2: Refreshing state... [id=subnet-0639d596f5d33f0c9]
aws_subnet.app_1: Refreshing state... [id=subnet-0b07afa925614ce56]
aws_subnet.public_subnet2: Refreshing state... [id=subnet-07cb652cc127b01ab]
aws_lb_target_group.tg: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:targetgroup/web-tg/ea47958c9d800f8b]
aws_security_group.web_sg: Refreshing state... [id=sg-0fff61e691d4c6ee6]
aws_route_table.public_rt: Refreshing state... [id=rtb-0c80ef9dd1a13fd9f]
aws_nat_gateway.nat_gw: Refreshing state... [id=nat-051f702403f8636ee]
aws_route_table.private_rt: Refreshing state... [id=rtb-0d8de1c49b7142d71]
aws_route_table_association.public_subnet2: Refreshing state... [id=rtbassoc-0592de91fd67eafcf]
aws_route_table_association.public_subnet1: Refreshing state... [id=rtbassoc-0fb61deb497071717]
aws_security_group.app_sg: Refreshing state... [id=sg-0370536b75214eb78]
aws_route_table_association.private_app_2: Refreshing state... [id=rtbassoc-0f850927929313a68]
aws_route_table_association.private_db_1: Refreshing state... [id=rtbassoc-093af01f71bc0b737]
aws_route_table_association.private_db_2: Refreshing state... [id=rtbassoc-016d343296581ae2d]
aws_route_table_association.private_app_1: Refreshing state... [id=rtbassoc-0110460a1a5153106]
aws_security_group.db_sg: Refreshing state... [id=sg-06203cb1489c8279f]
aws_lb.alb: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:loadbalancer/app/web-alb/39ad315e5521aba3]
aws_instance.web: Refreshing state... [id=i-090cf469f0fb788e9]
aws_instance.app: Refreshing state... [id=i-0fda82bd21cb712b0]
aws_lb_listener.listener: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:listener/app/web-alb/39ad315e5521aba3/80cec4d0ad541928]
aws_lb_target_group_attachment.web_attachment: Refreshing state... [id=arn:aws:elasticloadbalancing:us-east-1:110353307320:targetgroup/web-tg/ea47958c9d800f8b-20251114182753103700000002]

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.

Outputs:

alb_dns = "web-alb-1323547555.us-east-1.elb.amazonaws.com"

Hp@DESKTOP-9COU1S2 MINGW64 ~/Desktop/project-3 (master)
$ |
```

Web-tier output:

After applying the commands it automatically provides vpc and their components.

VPC:

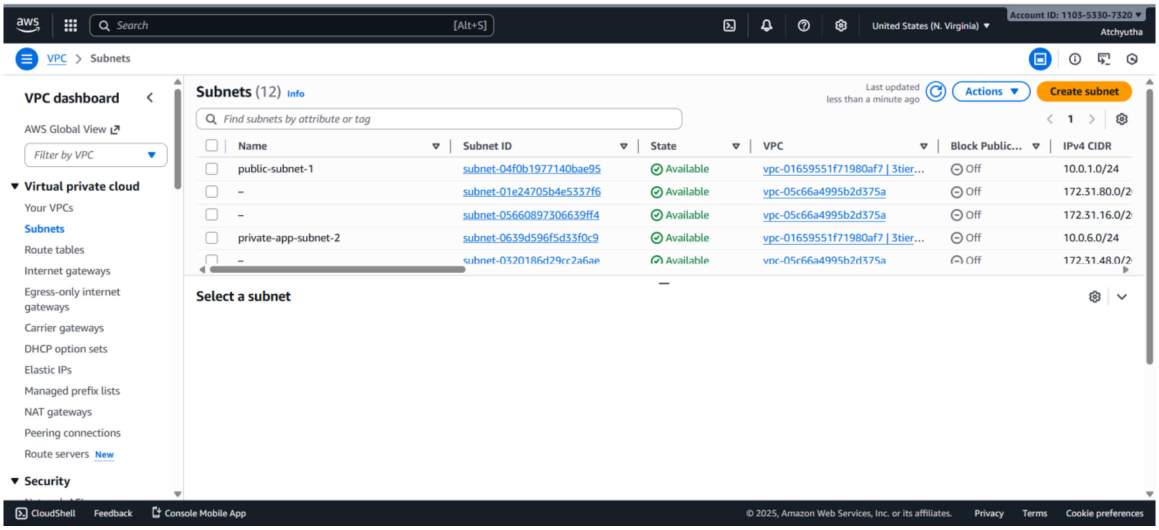


Your VPCs (2)

Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR
3tier-vpc	vpc-01659551f71980af7	Available	Off	10.0.0.0/16	–
–	vpc-05c66a4995b2d375a	Available	Off	172.31.0.0/16	–

Select a VPC above

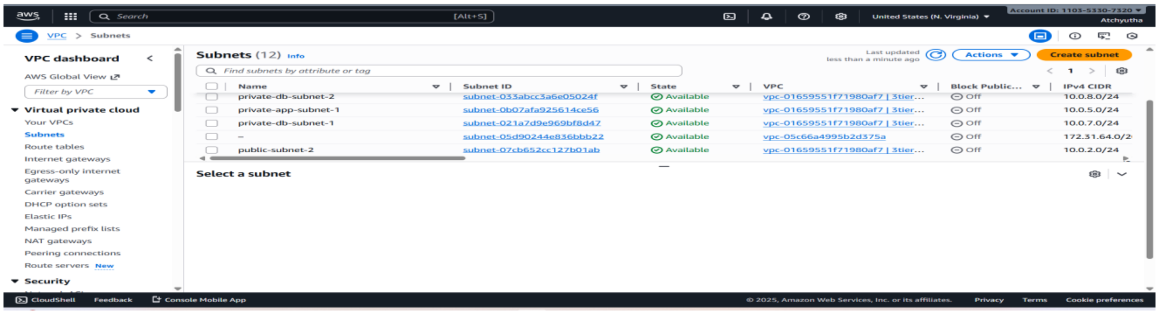
Subnet:



Subnets (12)

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
public-subnet-1	subnet-04f0b1977140bae95	Available	vpc-01659551f71980af7 | 3tier...	Off	10.0.1.0/24
–	subnet-01e24705b4e5337f6	Available	vpc-05c66a4995b2d375a	Off	172.31.80.0/2
–	subnet-05660897306639ff4	Available	vpc-05c66a4995b2d375a	Off	172.31.16.0/2
private-app-subnet-2	subnet-0639d596f5d33f0c9	Available	vpc-01659551f71980af7 | 3tier...	Off	10.0.6.0/24
–	subnet-0320186d29cc2a6ae	Available	vpc-05c66a4995b2d375a	Off	172.31.48.0/2

Select a subnet

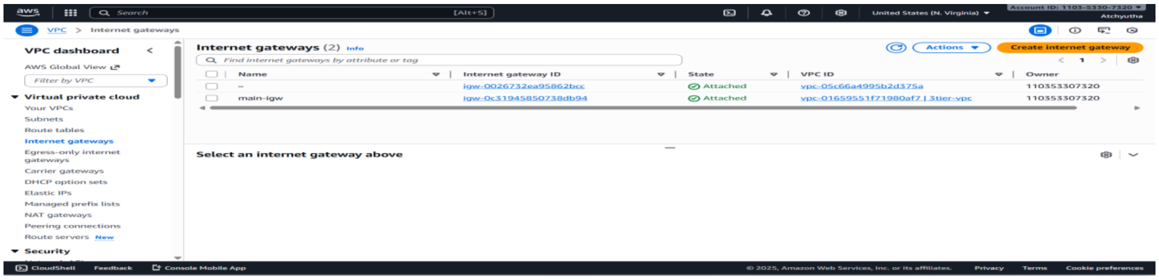


Subnets (12)

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
private-db-subnet-2	subnet-033abcc3a6e05024f	Available	vpc-01659551f71980af7 | 3tier...	Off	10.0.8.0/24
private-app-subnet-1	subnet-0b07afa925614ce56	Available	vpc-01659551f71980af7 | 3tier...	Off	10.0.5.0/24
private-db-subnet-1	subnet-021a7d9e969bf8d47	Available	vpc-01659551f71980af7 | 3tier...	Off	10.0.7.0/24
–	subnet-05d90244e836bbb22	Available	vpc-05c66a4995b2d375a	Off	172.31.64.0/2
public-subnet-2	subnet-07cb652cc127b01ab	Available	vpc-01659551f71980af7 | 3tier...	Off	10.0.2.0/24

Select a subnet

Internet gateway:



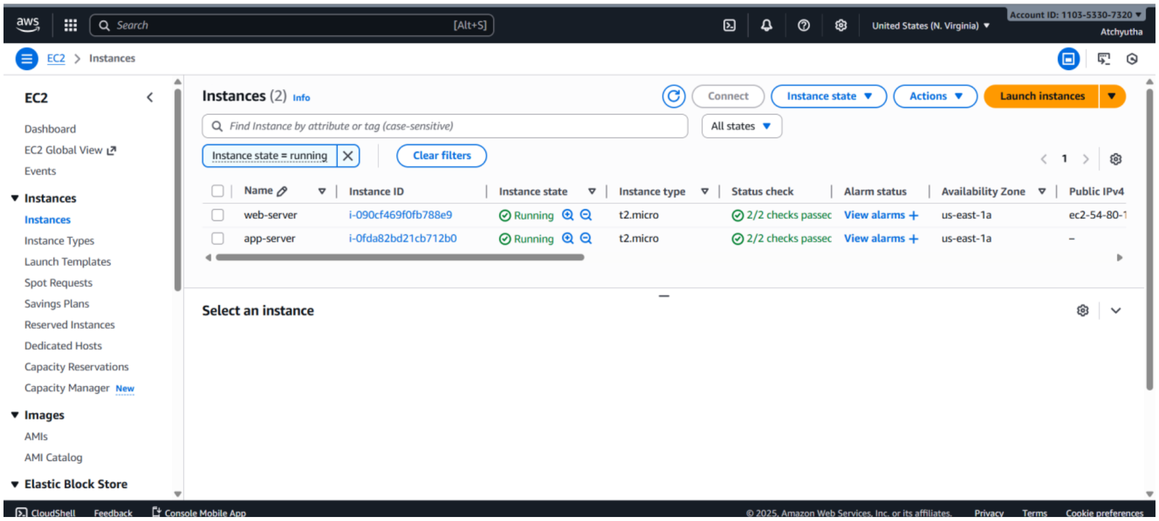
Internet gateways (2) Info

Find internet gateways by attribute or tag

Name	Internet gateway ID	State	VPC ID	Owner
–	igw-0026732ea95862bcc	Attached	vpc-05c66a4995b2d375a	110353307320
main-igw	igw-0c31945850738db94	Attached	vpc-01659551f71980af7 | 3tier-vpc	110353307320

Select an internet gateway above

APP-tier output:



Instances (2) Info

Find Instance by attribute or tag (case-sensitive)

Instance state = running

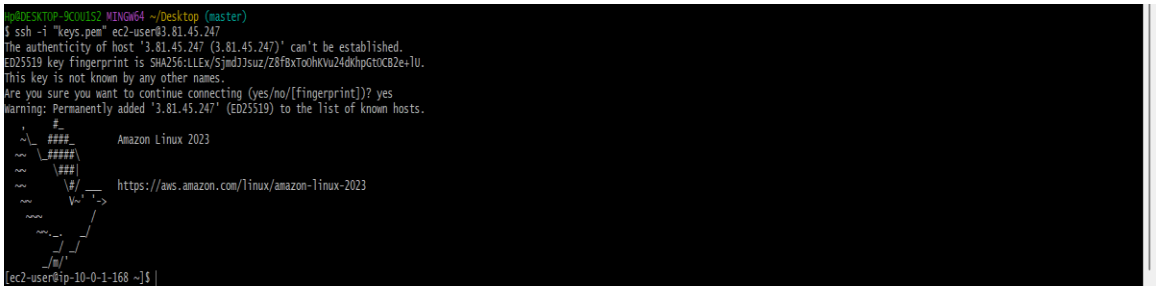
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
web-server	i-090cf469f0fb788e9	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	ec2-54-80-1
app-server	i-0fda82bd21cb712b0	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1a	–

Select an instance

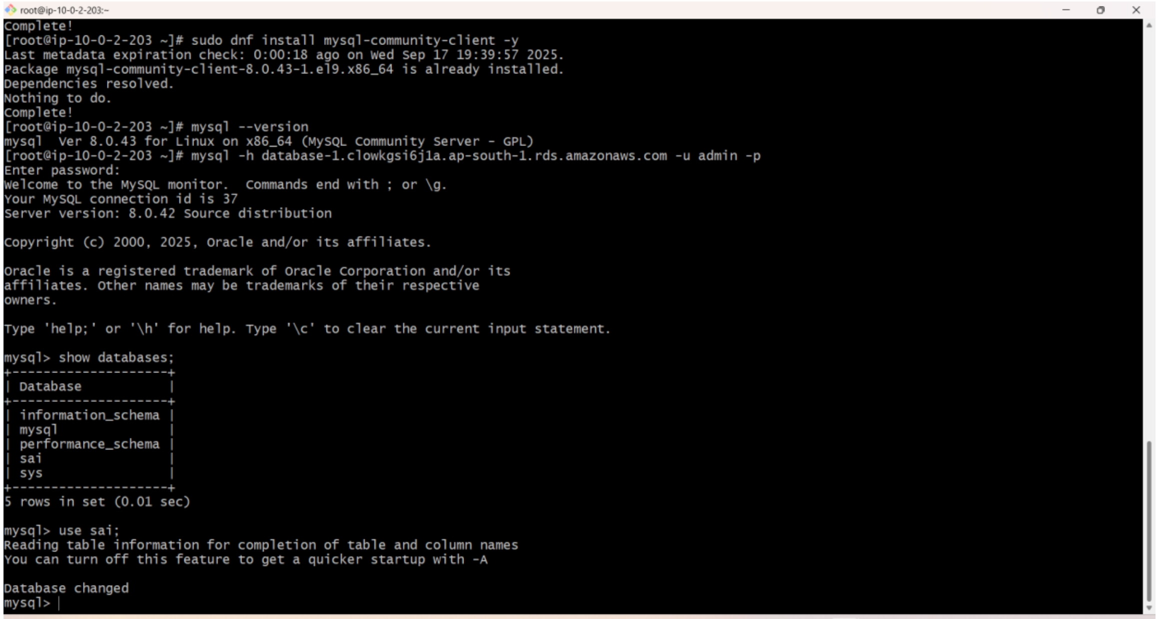
Connect to the server:

Outputs:

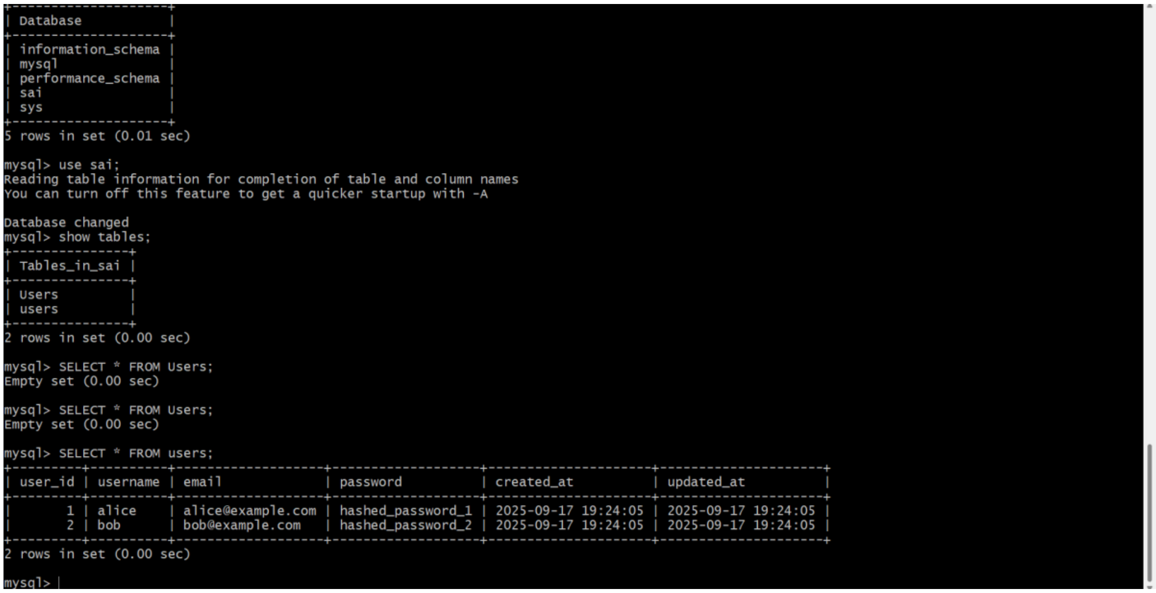
1. Connect to web EC2 (public subnet):



```
Hp@DESKTOP-9COU1S2 MINGW64 ~/Desktop (master)
$ ssh -i "keys.pem" ec2-user@3.81.45.247
The authenticity of host '3.81.45.247 (3.81.45.247)' can't be established.
ED25519 key fingerprint is SHA256:LLEx/SjmdJJsuz/Z8fBxToOhKVu24dKhpGtOCB2e+lU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.81.45.247' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-1-168 ~]$ 
```

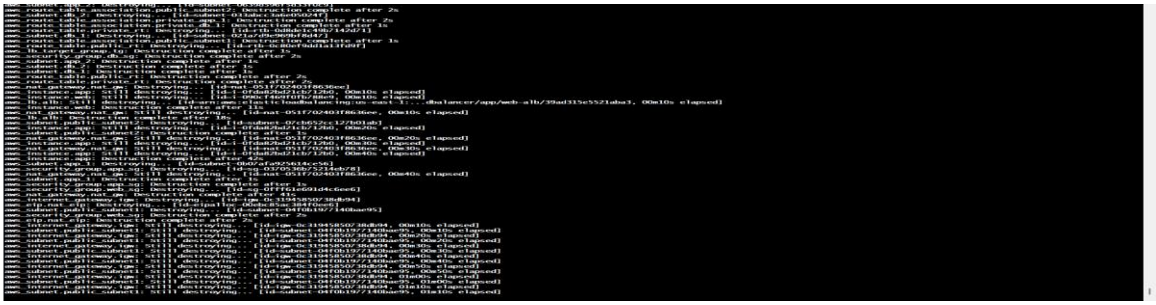


```
Complete!
[root@ip-10-0-2-203 ~]# sudo dnf install mysql-community-client -y
Last metadata expiration check: 0:00:18 ago on Wed Sep 17 19:39:57 2025.
Package mysql-community-client-8.0.43-1.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-10-0-2-203 ~]# mysql --version
mysql  Ver 8.0.43 for Linux on x86_64 (MySQL Community Server - GPL)
[root@ip-10-0-2-203 ~]# mysql -h database-1.clowkgsi6j1a.ap-south-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 37
Server version: 8.0.42 Source distribution

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sai                |
| sys                |
+--------------------+
5 rows in set (0.01 sec)

mysql> use sai;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> |
```



```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sai                |
| sys                |
+--------------------+
5 rows in set (0.01 sec)

mysql> use sai;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+---------------+
| Tables_in_sai |
+---------------+
| Users         |
| users         |
+---------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM Users;
Empty set (0.00 sec)

mysql> SELECT * FROM Users;
Empty set (0.00 sec)

mysql> SELECT * FROM users;
+---------+----------+-------------------+-------------------+---------------------+---------------------+
| user_id | username | email             | password          | created_at          | updated_at          |
+---------+----------+-------------------+-------------------+---------------------+---------------------+
|       1 | alice    | alice@example.com | hashed_password_1 | 2025-09-17 19:24:05 | 2025-09-17 19:24:05 |
|       2 | bob      | bob@example.com   | hashed_password_2 | 2025-09-17 19:24:05 | 2025-09-17 19:24:05 |
+---------+----------+-------------------+-------------------+---------------------+---------------------+
2 rows in set (0.00 sec)

mysql> |
```

15. terraform destroy:



```
aws_subnet.app_2: Destroying...
aws_route_table_association.public_subnet2: Destruction complete after 2s
aws_subnet.db_2: Destroying... [id=subnet-033abcc3a6e05024f]
aws_route_table_association.private_app_1: Destruction complete after 2s
aws_route_table_association.private_db_1: Destruction complete after 1s
aws_route_table.private_rt: Destroying... [id=rtb-0d8de1c49b7142d71]
aws_subnet.db_1: Destroying... [id=subnet-021a7d9e969bf8d47]
aws_route_table_association.public_subnet1: Destruction complete after 1s
aws_route_table.public_rt: Destroying... [id=rtb-0c80ef9dd1a13fd9f]
aws_lb_target_group.tg: Destruction complete after 1s
aws_security_group.db_sg: Destruction complete after 2s
aws_subnet.app_2: Destruction complete after 1s
aws_subnet.db_2: Destruction complete after 1s
aws_subnet.db_1: Destruction complete after 1s
aws_route_table.public_rt: Destruction complete after 2s
aws_route_table.private_rt: Destruction complete after 2s
aws_nat_gateway.nat_gw: Destroying... [id=nat-051f702403f8636ee]
aws_instance.app: Still destroying... [id=i-0fda82bd21cb712b0, 00m10s elapsed]
aws_instance.web: Still destroying... [id=i-090cf469f0fb788e9, 00m10s elapsed]
aws_lb.alb: Still destroying... [id=arn:aws:elasticloadbalancing:us-east-1:...dbalancer/app/web-alb/39ad315e5521aba3, 00m10s elapsed]
aws_instance.web: Destruction complete after 11s
aws_nat_gateway.nat_gw: Still destroying... [id=nat-051f702403f8636ee, 00m10s elapsed]
aws_lb.alb: Destruction complete after 18s
aws_subnet.public_subnet2: Destroying... [id=subnet-07cb652cc127b01ab]
aws_instance.app: Still destroying... [id=i-0fda82bd21cb712b0, 00m20s elapsed]
aws_subnet.public_subnet2: Destruction complete after 1s
aws_nat_gateway.nat_gw: Still destroying... [id=nat-051f702403f8636ee, 00m20s elapsed]
aws_instance.app: Still destroying... [id=i-0fda82bd21cb712b0, 00m30s elapsed]
aws_nat_gateway.nat_gw: Still destroying... [id=nat-051f702403f8636ee, 00m30s elapsed]
aws_instance.app: Still destroying... [id=i-0fda82bd21cb712b0, 00m40s elapsed]
aws_instance.app: Destruction complete after 42s
aws_subnet.app_1: Destroying... [id=subnet-0b07afa925614ce56]
aws_security_group.app_sg: Destroying... [id=sg-0370536b75214eb78]
aws_nat_gateway.nat_gw: Still destroying... [id=nat-051f702403f8636ee, 00m40s elapsed]
aws_subnet.app_1: Destruction complete after 1s
aws_security_group.app_sg: Destruction complete after 1s
aws_security_group.web_sg: Destroying... [id=sg-0fff61e691d4c6ee6]
aws_nat_gateway.nat_gw: Destruction complete after 41s
aws_internet_gateway.igw: Destroying... [id=igw-0c319458507 38db94]
aws_eip.nat_eip: Destroying... [id=eipalloc-00ebc85ac384f0ee6]
aws_subnet.public_subnet1: Destroying... [id=subnet-04f0b1977140bae95]
aws_security_group.web_sg: Destruction complete after 2s
aws_eip.nat_eip: Destruction complete after 2s
aws_internet_gateway.igw: Still destroying... [id=igw-0c31945850738db94, 00m10s elapsed]
aws_subnet.public_subnet1: Still destroying... [id=subnet-04f0b1977140bae95, 00m10s elapsed]
aws_internet_gateway.igw: Still destroying... [id=igw-0c31945850738db94, 00m20s elapsed]
aws_subnet.public_subnet1: Still destroying... [id=subnet-04f0b1977140bae95, 00m20s elapsed]
aws_internet_gateway.igw: Still destroying... [id=igw-0c31945850738db94, 00m30s elapsed]
aws_subnet.public_subnet1: Still destroying... [id=subnet-04f0b1977140bae95, 00m30s elapsed]
aws_internet_gateway.igw: Still destroying... [id=igw-0c31945850738db94, 00m40s elapsed]
aws_subnet.public_subnet1: Still destroying... [id=subnet-04f0b1977140bae95, 00m40s elapsed]
aws_internet_gateway.igw: Still destroying... [id=igw-0c31945850738db94, 00m50s elapsed]
aws_subnet.public_subnet1: Still destroying... [id=subnet-04f0b1977140bae95, 00m50s elapsed]
aws_internet_gateway.igw: Still destroying... [id=igw-0c31945850738db94, 01m00s elapsed]
aws_subnet.public_subnet1: Still destroying... [id=subnet-04f0b1977140bae95, 01m00s elapsed]
aws_internet_gateway.igw: Still destroying... [id=igw-0c31945850738db94, 01m10s elapsed]
aws_subnet.public_subnet1: Still destroying... [id=subnet-04f0b1977140bae95, 01m10s elapsed]
```

Summary

A 3-tier architecture consists of a Web Tier**,** Application Tier and Database Tier each isolated in its own subnets for security and scalability. Terraform is used to automate the provisioning of all AWS resources. Using Terraform, I automated a complete 3-tier AWS architecture by provisioning a secure VPC, isolated subnets, ALB for web traffic, EC2 instances for web and app layers, a private RDS database, NAT/IGW routing and security groups delivering a fully automated, modular, and production-ready infrastructure.