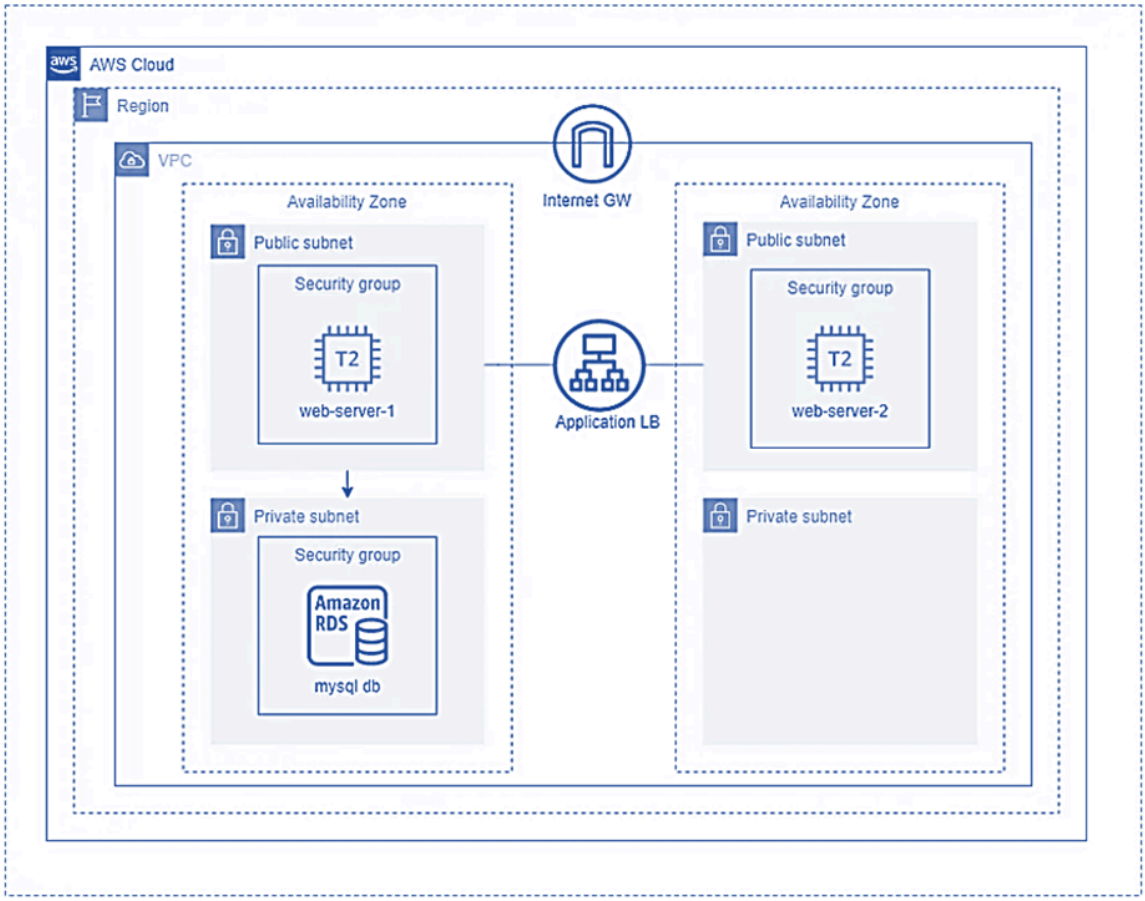
15 Days of Terraform – Curated by Shaik Hari Sadia Anjum

# Deploy Two Tier Architecture in AWS using Terraform




Two-Tier Architecture diagram

**Components:**

1. **VPC**
2. **Internet Gateway**
3. **Subnets - Private and public subnets**
4. **Load balancer**
5. **Route tables**
6. **RDS MySQL**

## ***Creating Provider.tf file:***

```
provider "aws" {

  region = "ap-south-1"

}
```

## ***Creating VPC.tf***

```
resource "aws_vpc" "vpc" {
  cidr_block = "10.0.0.0/16"
  tags = {
    name = "Two-tier-vpc"
  }

}

resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.vpc.id
  tags = {
    name = "two-tier-vpc-igw"
  }

}

resource "aws_subnet" "public_1" {
  vpc_id                  = aws_vpc.vpc.id
  cidr_block              = "10.0.1.0/24"
  availability_zone       = "ap-south-1a"
  map_public_ip_on_launch = true
  tags = {
    name = "public-1"
  }

}

resource "aws_subnet" "public_2" {
  vpc_id                  = aws_vpc.vpc.id
  cidr_block              = "10.0.2.0/24"
```

```
  availability_zone       = "ap-south-1b"
  map_public_ip_on_launch = true
  tags = {
    name = "public-2"
  }
}

resource "aws_subnet" "private_1" {
  vpc_id                  = aws_vpc.vpc.id
  cidr_block              = "10.0.3.0/24"
  availability_zone       = "ap-south-1a"
  map_public_ip_on_launch = false
  tags = {
    name = "private-1"
  }

}

resource "aws_subnet" "private_2" {
  vpc_id                  = aws_vpc.vpc.id
  cidr_block              = "10.0.4.0/24"
  availability_zone       = "ap-south-1b"
  map_public_ip_on_launch = false
  tags = {
    name = "private-2"
  }
}
```

## ***Creating Security-resources.tf***

```
resource "aws_security_group" "alb-sg" {

 name       = "alb-sg"

 description = "security grp for ALB"

 vpc_id      = aws_vpc.vpc.id

 ingress {

   from_port   = "0"

   to_port     = "0"

   protocol    = "-1"

   cidr_blocks = ["0.0.0.0/0"]

 }

 egress {

   from_port   = "0"

   to_port     = "0"

   protocol    = "-1"

   cidr_blocks = ["0.0.0.0/0"]

 }


}

# create ALB

resource "aws_lb" "two_tier_alb" {

 name            = "two-tier-alb"

 internal        = false

 load_balancer_type = "application"
```

```
  security_groups    = [aws_security_group.alb-sg.id]

  subnets            = [aws_subnet.public_1.id, aws_subnet.public_2.id]

}

# Create ALB target group

resource "aws_lb_target_group" "alb-tg" {

  name      = "alb-tg"

  port      = 80

  protocol   = "HTTP"

  vpc_id     = aws_vpc.vpc.id

  depends_on = [aws_vpc.vpc]

}


resource "aws_lb_target_group_attachment" "tg-attach1" {

  target_group_arn = aws_lb_target_group.alb-tg.arn

  target_id        = aws_instance.webserver_1.id

  port             = 80

  depends_on       = [aws_instance.webserver_1]

}

resource "aws_lb_target_group_attachment" "tg-attach2" {

  target_group_arn = aws_lb_target_group.alb-tg.arn

  target_id        = aws_instance.webserver_2.id

  port             = 80

  depends_on       = [aws_instance.webserver_2]

}
```

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

  }

}

# Create route table to internet gateway

resource "aws_route_table" "rt-igw" {
  vpc_id = aws_vpc.vpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }
  tags = {
    name = "rt-igw"
  }

}

# Associate public subnets with route table

resource "aws_route_table_association" "public_route_1" {
  subnet_id      = aws_subnet.public_1.id
  route_table_id = aws_route_table.rt-igw.id
```

```
}

resource "aws_route_table_association" "public_route_2" {
  subnet_id      = aws_subnet.public_2.id
  route_table_id = aws_route_table.rt-igw.id
}

# Create security groups
resource "aws_security_group" "public_sg" {
  name        = "public-sg"
  description = "Allow web and ssh traffic"
  vpc_id      = aws_vpc.vpc.id

  ingress {
    from_port   = 80
    to_port     = 80
    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"]

  }
}

resource "aws_security_group" "private_sg" {
  name        = "private-sg"
  description = "Allow web tier and ssh traffic"
  vpc_id      = aws_vpc.vpc.id

  ingress {
    from_port       = 3306
    to_port         = 3306
    protocol        = "tcp"
    cidr_blocks     = ["10.0.0.0/16"]
    security_groups = [aws_security_group.public_sg.id]
  }
  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"]

 }

}
```

## ***Creating EC2.tf***

```
resource "aws_instance" "webserver_1" {

 ami                    = "ami-0e35ddab05955cf57"

 instance_type          = "t2.micro"

 key_name               = "two-tier-aws-terraform"

 availability_zone      = "ap-south-1a"

 vpc_security_group_ids     = [aws_security_group.public_sg.id]

 associate_public_ip_address = true

 subnet_id              = aws_subnet.public_1.id

 user_data = <<-EOF

        #!/bin/bash

        sudo apt update -y

        sudo apt install nginx -y

        sudo systemctl enable nginx

        sudo systemctl start nginx

        EOF

 tags = {

  name = "webserver-1"

 }
}
resource "aws_instance" "webserver_2" {

 ami                    = "ami-0e35ddab05955cf57"
```

```
  instance_type           = "t2.micro"

  key_name                = "two-tier-aws-terraform"

  availability_zone       = "ap-south-1b"

  vpc_security_group_ids     = [aws_security_group.public_sg.id]

  associate_public_ip_address = true

  subnet_id               = aws_subnet.public_2.id


  user_data = <<-EOF

          #!/bin/bash

          sudo apt update -y

          sudo apt install nginx -y

          sudo systemctl enable nginx

          sudo systemctl start nginx

          EOF

  tags = {

    name = "webserver-2"

  }
}
```

## ***Creating DB.tf***

```
resource "aws_db_subnet_group" "db_subnet" {

  name      = "db_subnet"

  subnet_ids = [aws_subnet.private-1.id, aws_subnet.private_2.id]


}


resource "aws_db_instance" “mydatabase" {

  allocated_storage     = 5
```

```
engine              = "mysql"

engine_version      = "5.7"

instance_class      = "db.t2.micro"

identifier          = "db-instance"

db_name             = "mydatabase"

username            = "admin"

password            = "password"

db_subnet_group_name   = aws_db_subnet_group.db_subnet.id

vpc_security_group_ids = [aws_security_group.private_sg.id]

publicly_accessible    = false

skip_final_snapshot    = true

}
```

## Let’s deploy!

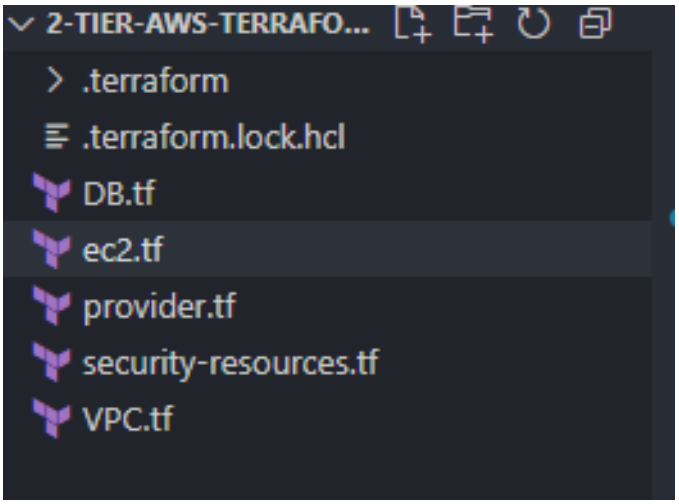
configure aws through Vs code with Access Key ID and Secret Access Key

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS                                  pwsh

PS C:\Users\admin\Desktop\2-tier-aws-terraform> aws configure
AWS Access Key ID [****************4AOX]: AKIASVLKCI3AMBJR7PPU
AWS Secret Access Key [****************R9ZY]: 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
Default region name [us-east-1]: us-east-1
Default output format [yes]: yes
```

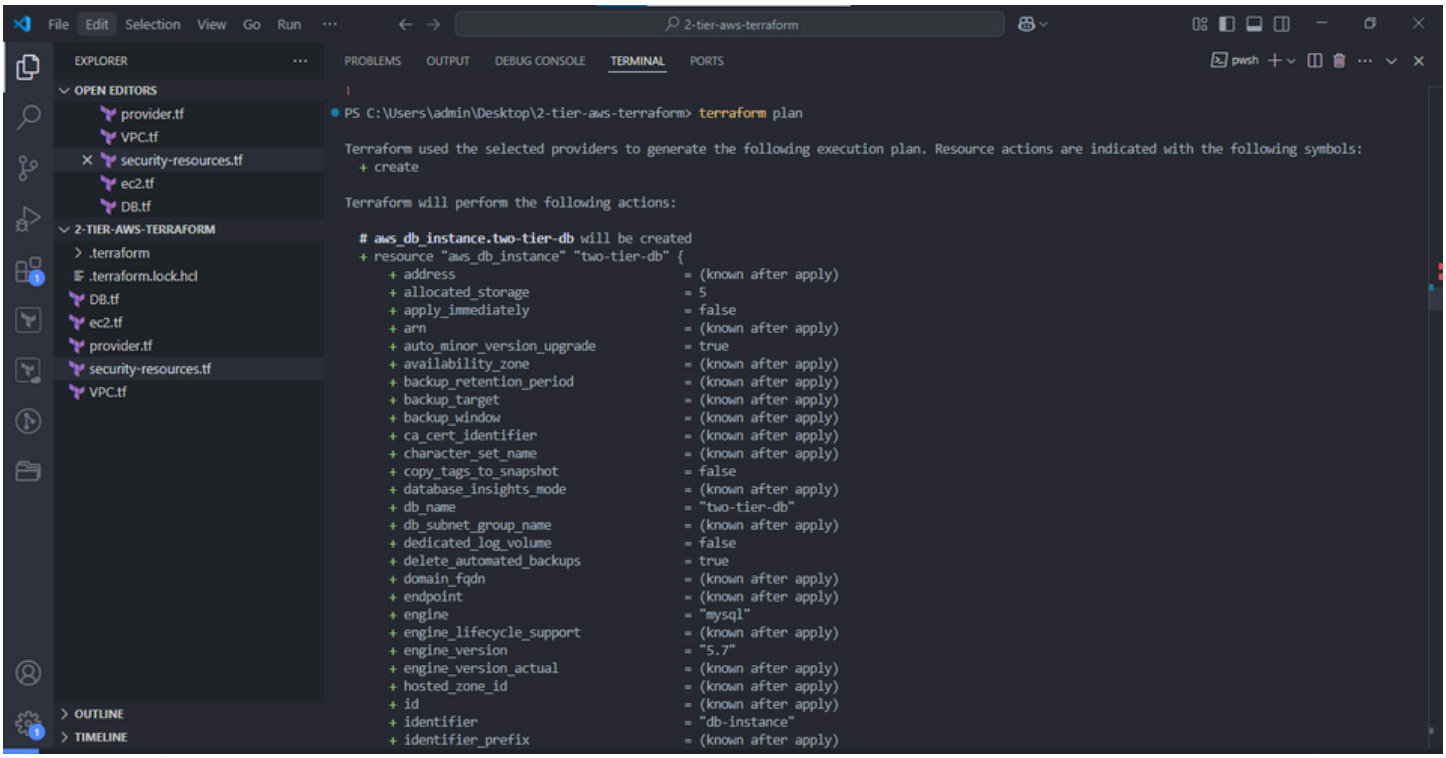
- **terraform init**

```
PS C:\Users\admin\Desktop\2-tier-aws-terraform> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.98.0...
- Installed hashicorp/aws v5.98.0 (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.
```

we can see after initializing lock.hcl file has created




- **terraform plan**



```
      + identifier                          = "db-instance"
      + identifier_prefix                   = (known after apply)
      + instance_class                      = "db.t2.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)
      + performance_insights_retention_period = (known after apply)
      + port                                = (known after apply)
      + publicly_accessible                 = false
      + replica_mode                        = (known after apply)
      + replicas                            = (known after apply)
      + resource_id                         = (known after apply)
      + skip_final_snapshot                 = true
      + snapshot_identifier                 = (known after apply)
      + status                              = (known after apply)
      + storage_throughput                  = (known after apply)
      + storage_type                        = (known after apply)
      + tags_all                            = (known after apply)
      + timezone                            = (known after apply)
      + username                            = "admin"
      + vpc_security_group_ids              = (known after apply)
```

```
# aws_db_subnet_group.db_subnet will be created
+ resource "aws_db_subnet_group" "db_subnet" {
    + arn                    = (known after apply)
    + description            = "Managed by Terraform"
    + id                     = (known after apply)
    + name                   = "db_subnet"
    + name_prefix            = (known after apply)
    + subnet_ids             = (known after apply)
    + supported_network_types = (known after apply)
    + tags_all               = (known after apply)
    + vpc_id                 = (known after apply)
  }

# aws_instance.webserver_1 will be created
+ resource "aws_instance" "webserver_1" {
    + ami                                  = "ami-0429d68a1cd41ca80"
    + arn                                  = (known after apply)
    + associate_public_ip_address          = true
    + availability_zone                    = "ap-south-1a"
    + cpu_core_count                       = (known after apply)
    + cpu_threads_per_core                 = (known after apply)
    + disable_api_stop                     = (known after apply)
    + disable_api_termination              = (known after apply)
    + ebs_optimized                        = (known after apply)
    + enable_primary_ipv6                  = (known after apply)
    + get_password_data                    = false
    + host_id                              = (known after apply)
    + host_resource_group_arn              = (known after apply)
    + iam_instance_profile                 = (known after apply)
    + id                                   = (known after apply)
    + instance_initiated_shutdown_behavior = (known after apply)
    + instance_lifecycle                   = (known after apply)
    + instance_state                       = (known after apply)
    + instance_type                        = "t2.micro"
    + ipv6_address_count                   = (known after apply)
    + ipv6_addresses                       = (known after apply)
```

```
    + private_dns_name_options (known after apply)

    + root_block_device (known after apply)
  }

# aws_instance.webserver_2 will be created
+ resource "aws_instance" "webserver_2" {
    + ami                                  = "ami-0429d68a1cd41ca80"
    + arn                                  = (known after apply)
    + associate_public_ip_address          = true
    + availability_zone                    = "ap-south-1b"
    + cpu_core_count                       = (known after apply)
    + cpu_threads_per_core                 = (known after apply)
    + disable_api_stop                     = (known after apply)
    + disable_api_termination              = (known after apply)
    + ebs_optimized                        = (known after apply)
    + enable_primary_ipv6                  = (known after apply)
    + get_password_data                    = false
    + host_id                              = (known after apply)
    + host_resource_group_arn              = (known after apply)
    + iam_instance_profile                 = (known after apply)
    + id                                   = (known after apply)
    + instance_initiated_shutdown_behavior = (known after apply)
    + instance_lifecycle                   = (known after apply)
    + instance_state                       = (known after apply)
    + instance_type                        = "t2.micro"
    + ipv6_address_count                   = (known after apply)
    + ipv6_addresses                       = (known after apply)
    + key_name                             = "two-tier-aws-terraform"
    + monitoring                           = (known after apply)
    + outpost_arn                          = (known after apply)
    + password_data                        = (known after apply)
    + placement_group                      = (known after apply)
    + placement_partition_number           = (known after apply)
    + primary_network_interface_id         = (known after apply)
    + private_dns                          = (known after apply)
    + private_ip                           = (known after apply)
```

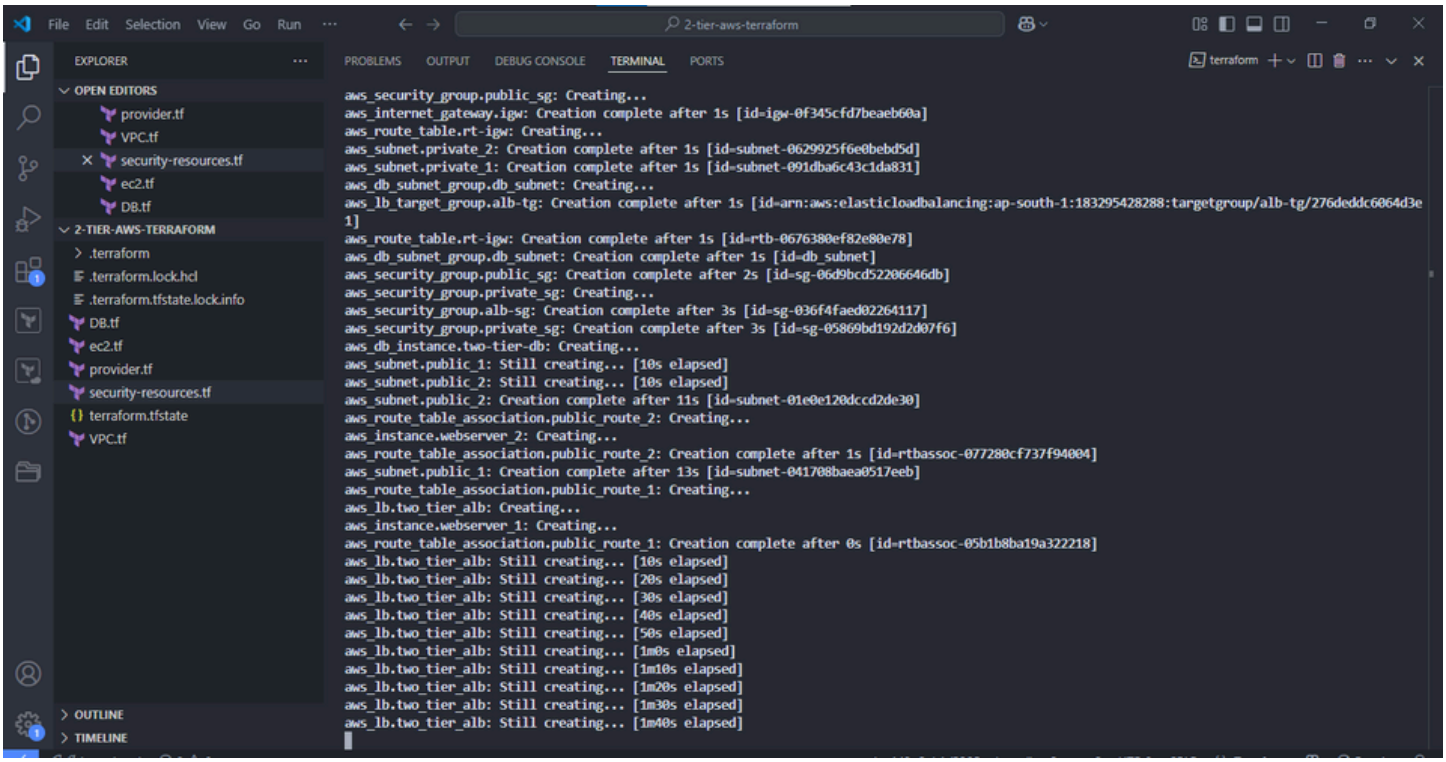
```
    }
    + vpc_id                                     = (known after apply)
  }

# aws_vpc.vpc will be created
+ resource "aws_vpc" "vpc" {
    + 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)
    + dhcp_options_id                      = (known after apply)
    + 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" = "Two-tier-vpc"
      }
    + tags_all                             = {
        + "name" = "Two-tier-vpc"
      }
  }

Plan: 21 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.
PS C:\Users\admin\Desktop\2-tier-aws-terraform>
```

- **terraform apply**

```
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.
PS C:\Users\admin\Desktop\2-tier-aws-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_db_instance.two-tier-db will be created
  + resource "aws_db_instance" "two-tier-db" {
      + address                               = (known after apply)
      + allocated_storage                     = 5
      + 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                               = "two-tier-db"
      + db_subnet_group_name                  = (known after apply)
      + 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                        = "5.7"
      + engine_version_actual                 = (known after apply)
      + hosted_zone_id                        = (known after apply)
      + id                                    = (known after apply)
```

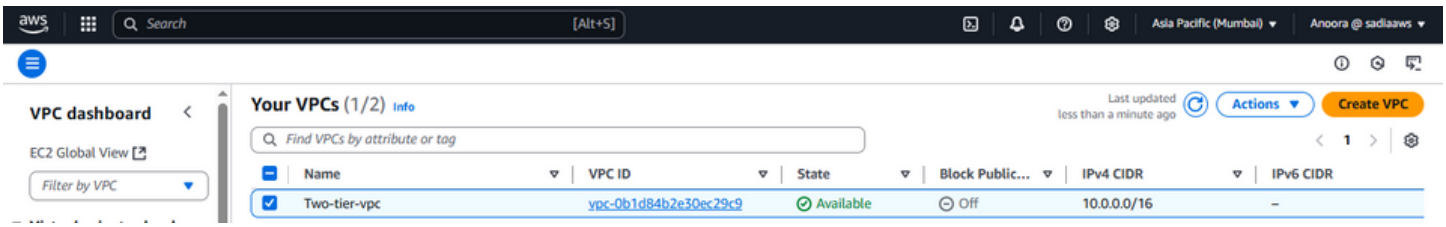
Confirm yes to create all resources

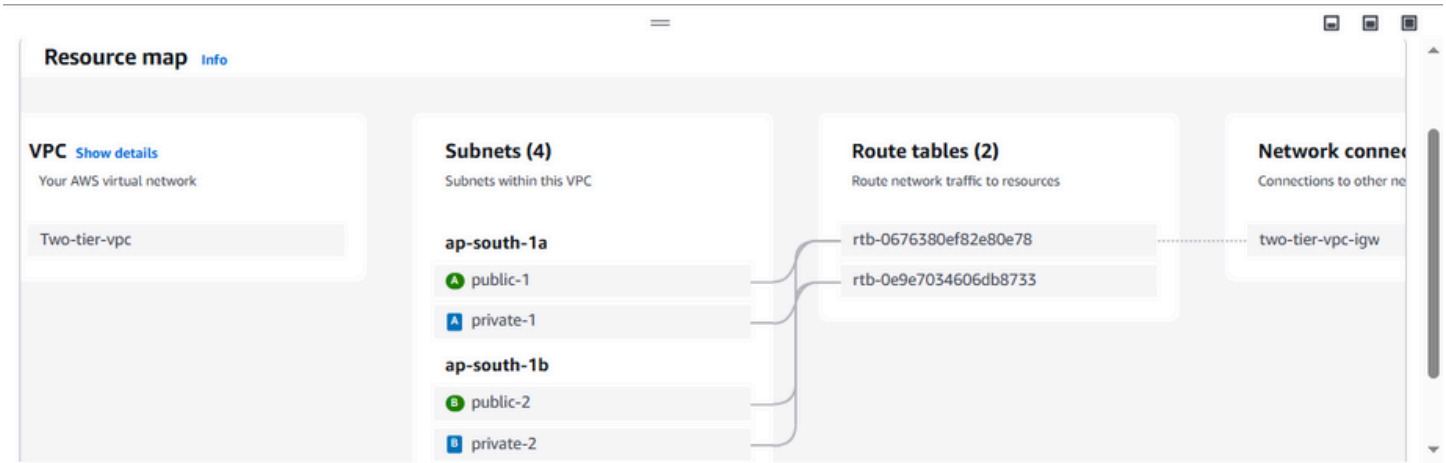
```
      + id                                    = (known after apply)
      + instance_tenancy                      = "default"
      + ipv6_association_id                   = (known after apply)
      + ipv6_cidr_block                       = (known after apply)
      + ipv6_cidr_block_network_border_group  = (known after apply)
      + main_route_table_id                   = (known after apply)
      + owner_id                              = (known after apply)
      + tags                                  = {
          + "name" = "Two-tier-vpc"
        }
      + tags_all                              = {
          + "name" = "Two-tier-vpc"
        }
    }

Plan: 21 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.vpc: Creating...
aws_vpc.vpc: Creation complete after 1s [id=vpc-0b1d84b2e30ec29c9]
aws_subnet.private_2: Creating...
aws_internet_gateway.igw: Creating...
aws_subnet.private_1: Creating...
aws_lb_target_group.alb-tg: Creating...
aws_subnet.public_1: Creating...
aws_subnet.public_2: Creating...
aws_security_group.alb-sg: Creating...
aws_security_group.public_sg: Creating...
aws_internet_gateway.igw: Creation complete after 1s [id=igw-0f345cfd7beaeb60a]
aws_route_table.rt-igw: Creating...
aws_subnet.private_2: Creation complete after 1s [id=subnet-0629925f6e0bebd5d]
aws_subnet.private_1: Creation complete after 1s [id=subnet-091dba6c43c1da831]
```



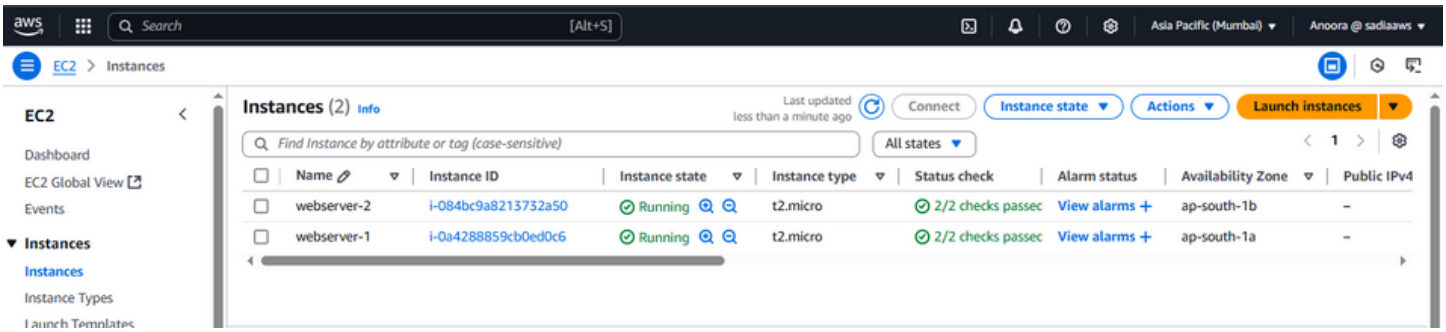

## Go to the AWS console and verify

1. VPC and Network resources

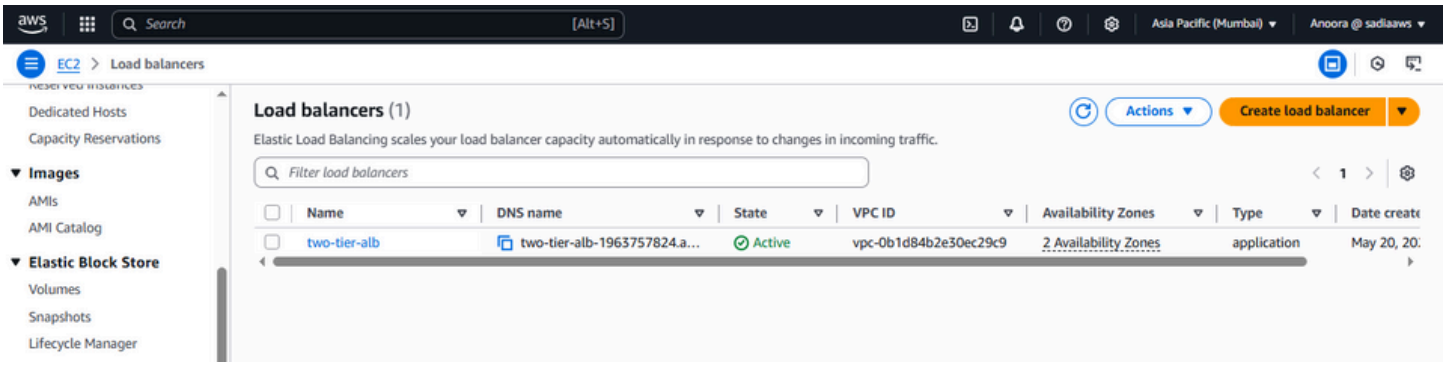


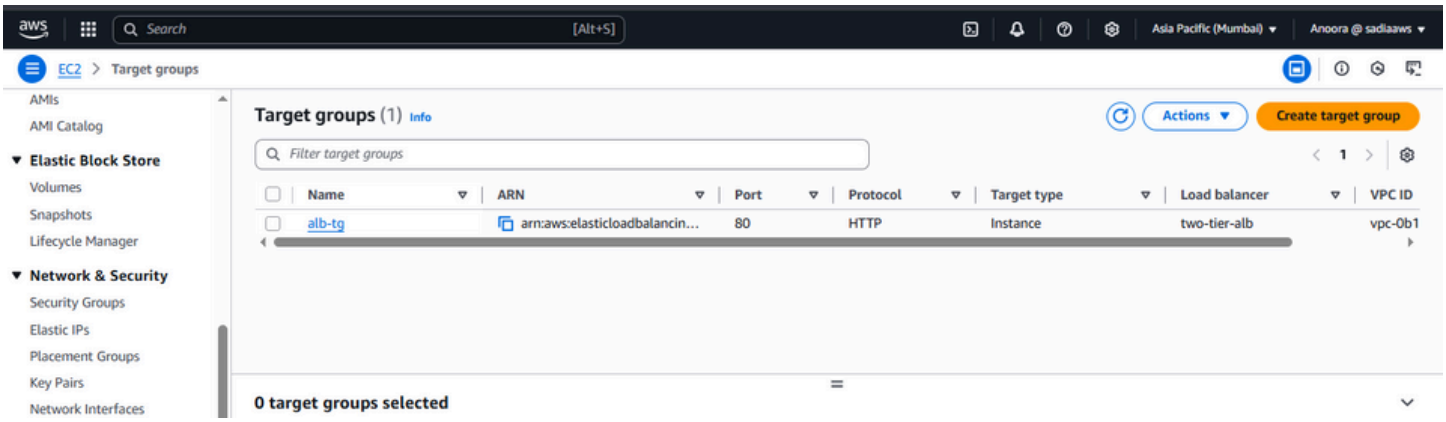

2. EC2 instances



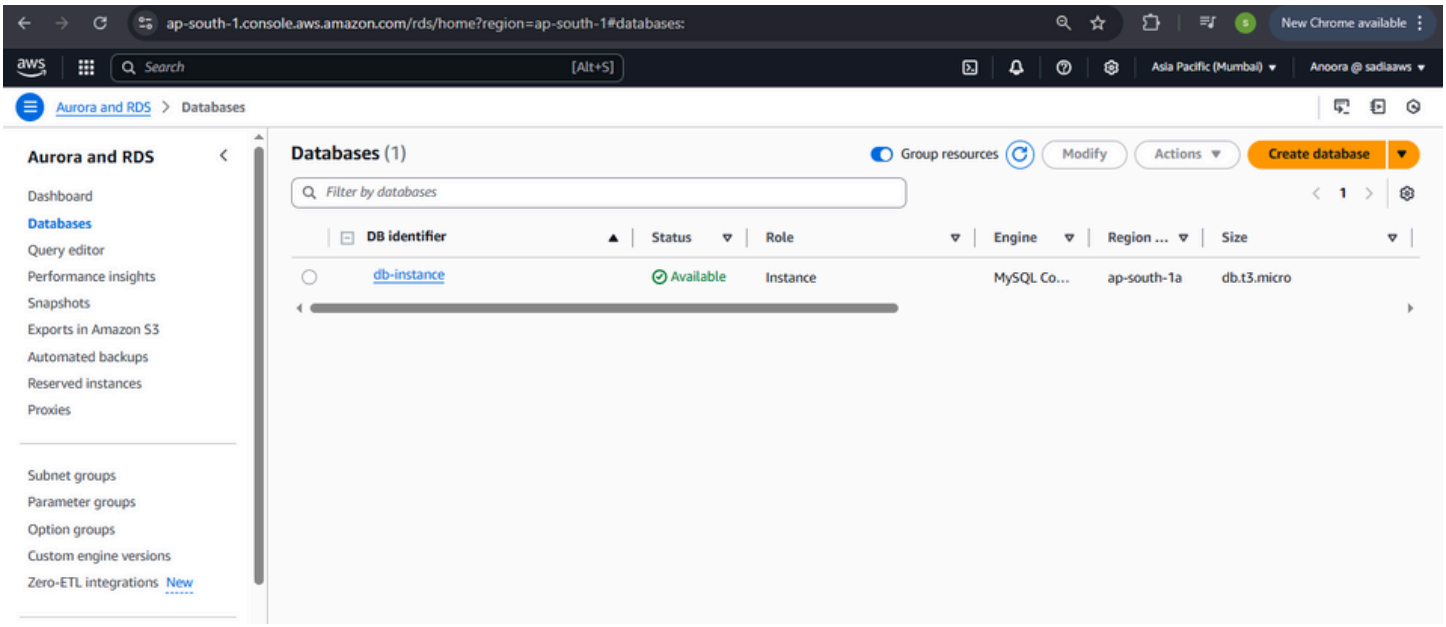

3. Load Balancer

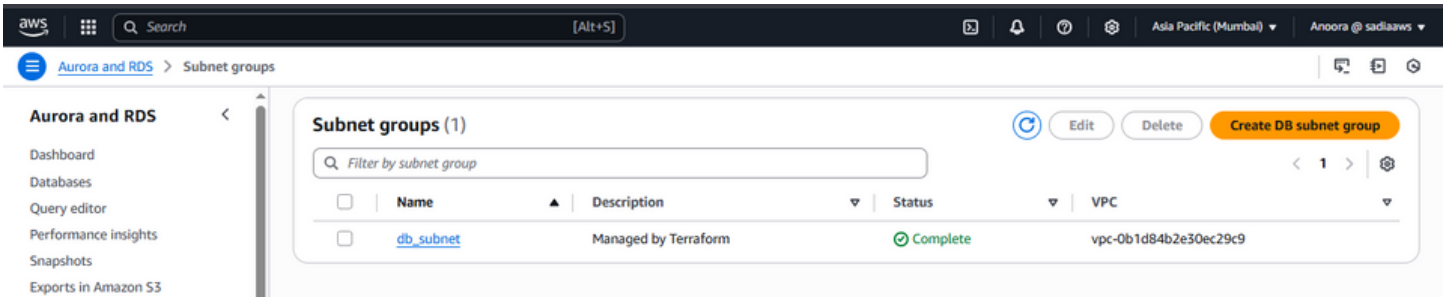


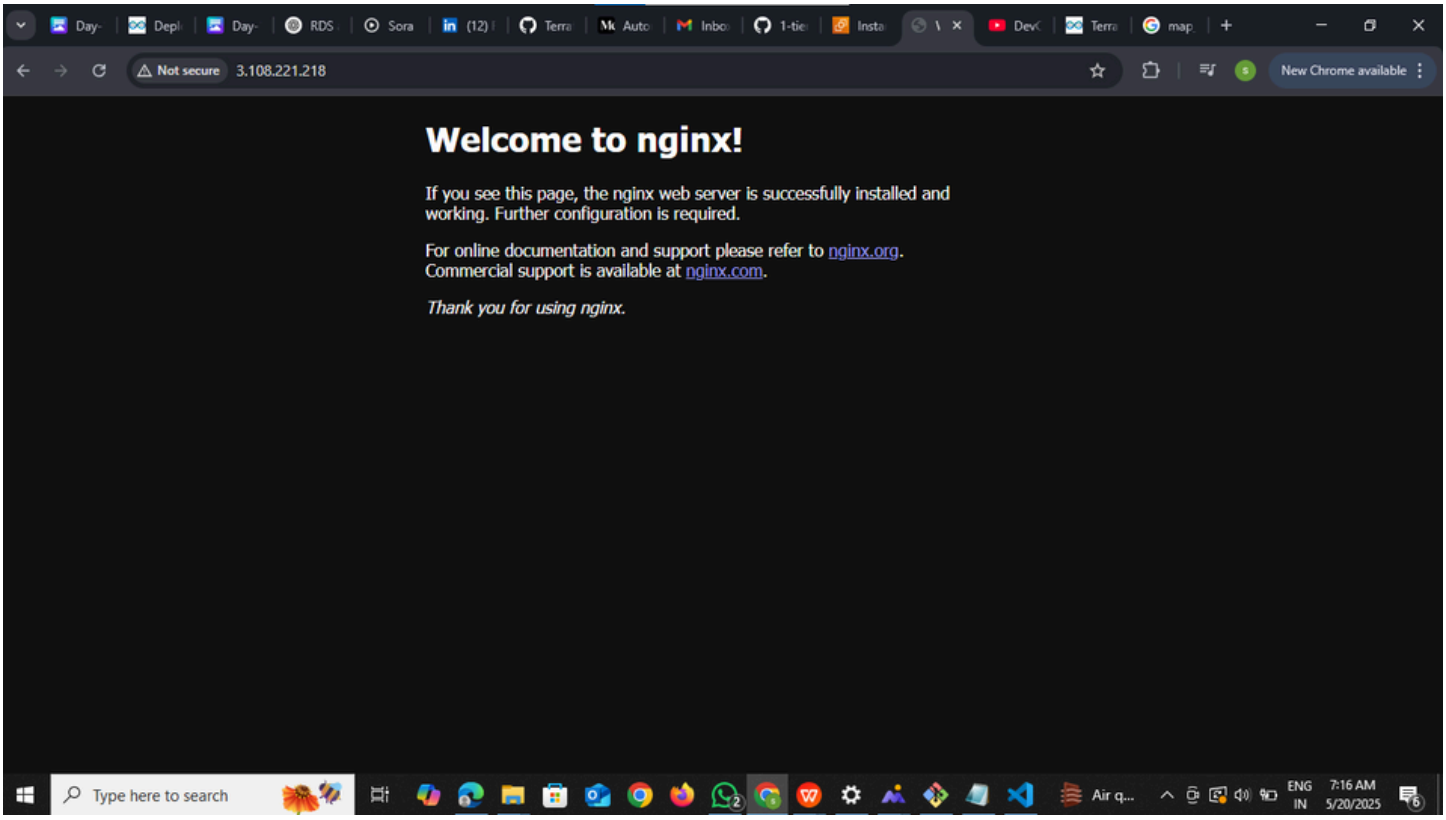

4. Target Group

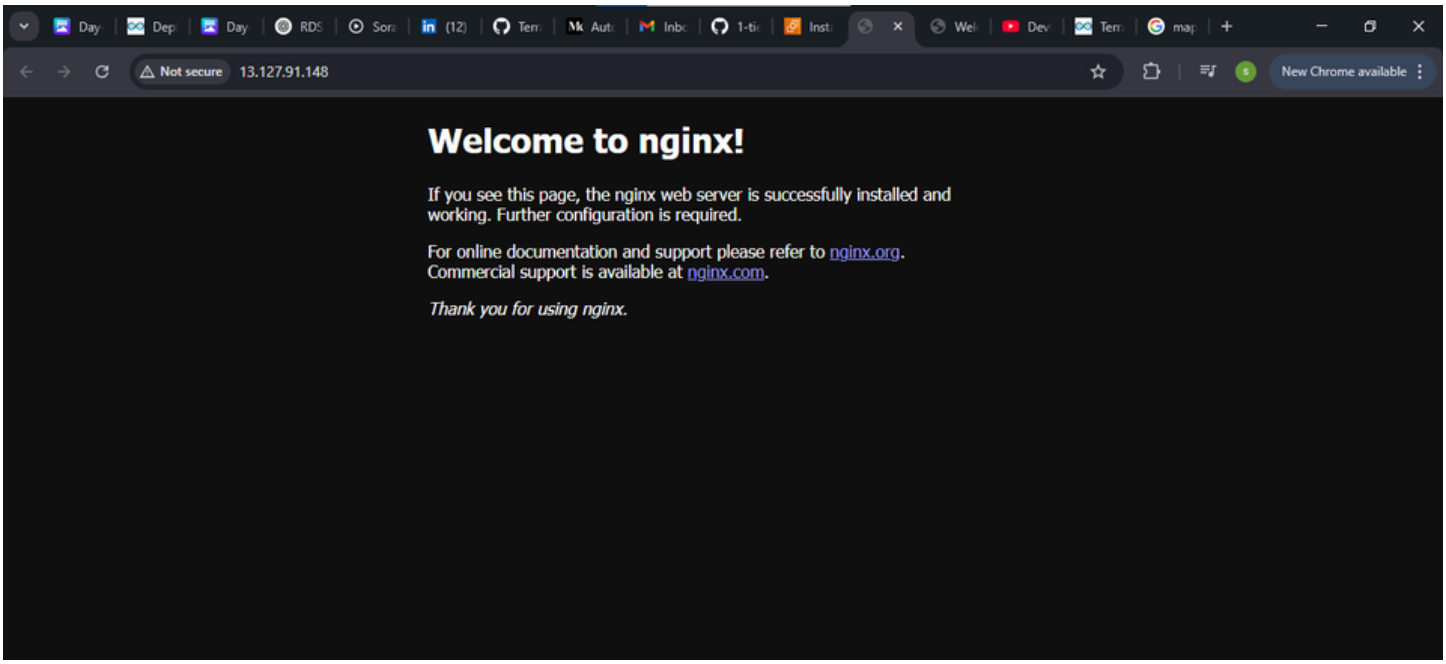


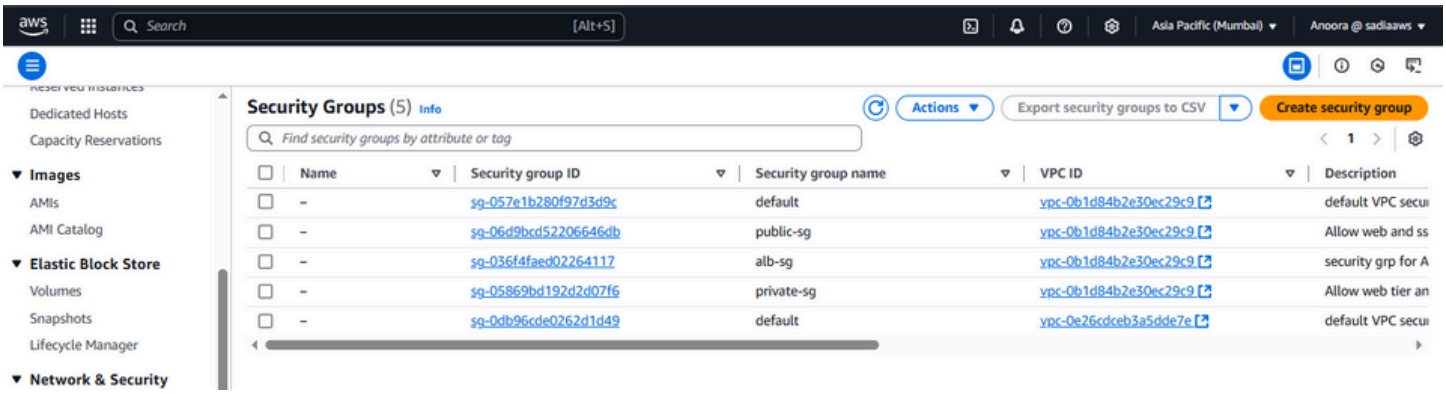

5. RDS MYSQL Database

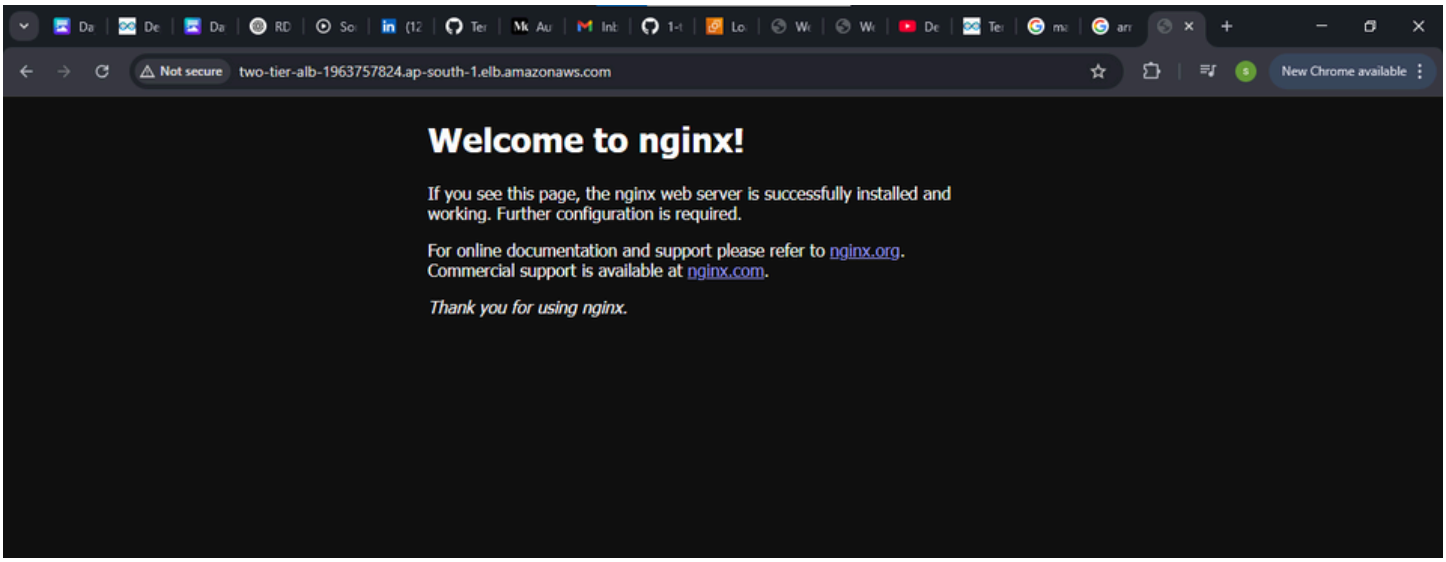

ap-south-1.console.aws.amazon.com/rds/home?region=ap-south-1#databases:
Aurora and RDS > Databases
Aurora and RDS
Dashboard
Databases
Query editor
Performance insights
Snapshots
Exports in Amazon S3
Automated backups
Reserved instances
Proxies
Subnet groups
Parameter groups
Option groups
Custom engine versions
Zero-ETL integrations New
Databases (1)
Group resources
Modify
Actions
Create database
Filter by databases
DB identifier
Status
Role
Engine
Region ...
Size
db-instance
Available
Instance
MySQL Co...
ap-south-1a
db.t3.micro

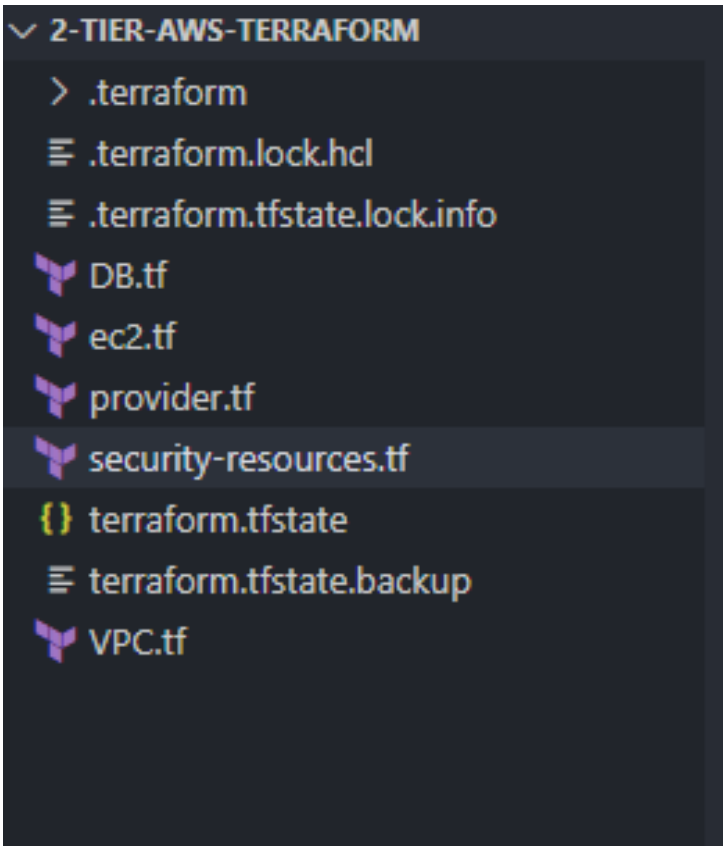

Aurora and RDS > Subnet groups
Aurora and RDS
Dashboard
Databases
Query editor
Performance insights
Snapshots
Exports in Amazon S3
Subnet groups (1)
Edit
Delete
Create DB subnet group
Filter by subnet group
Name
Description
Status
VPC
db_subnet
Managed by Terraform
Complete
vpc-0b1d84b2e30ec29c9


Not secure 3.108.221.218
Welcome to nginx!
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.

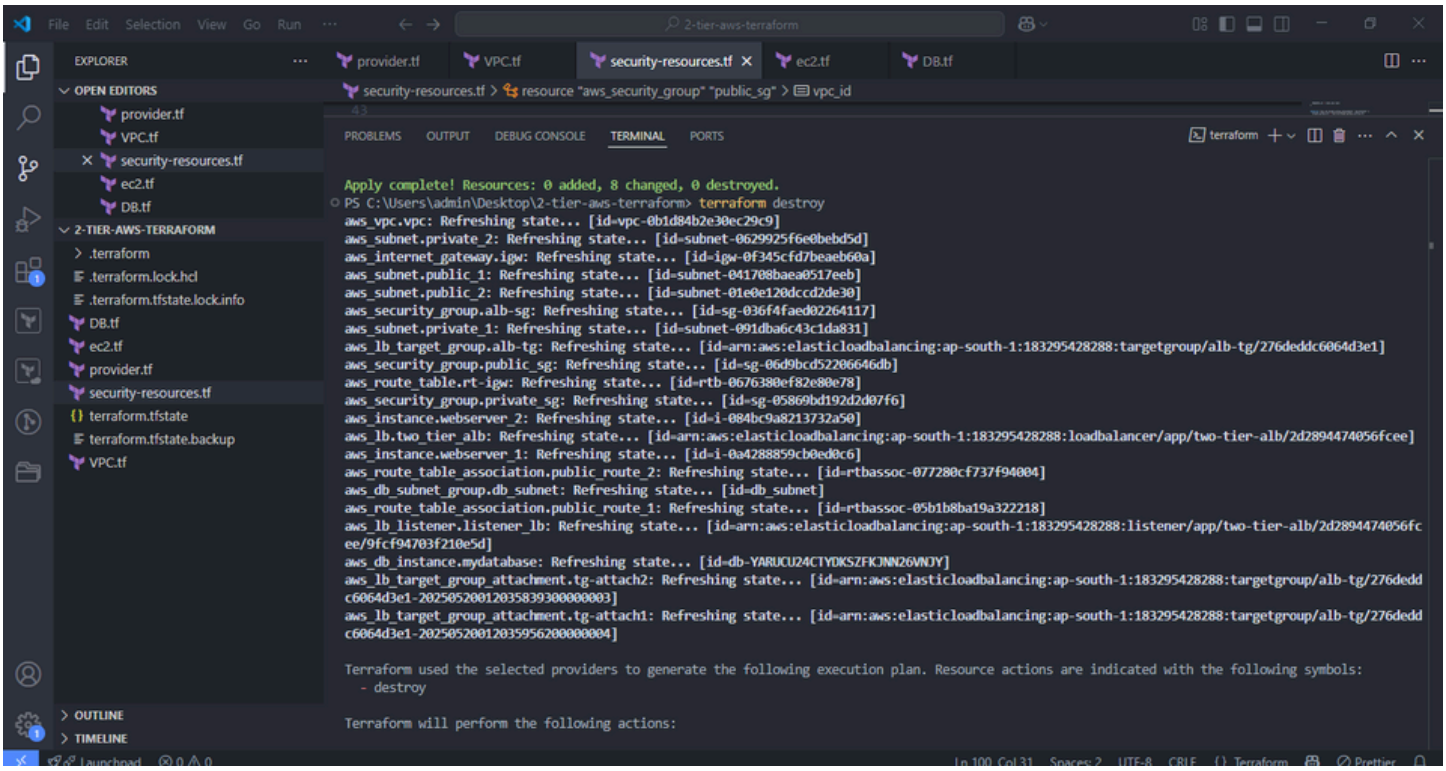

Not secure
13.127.91.148
New Chrome available
Welcome to nginx!
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.


aws
Search
[Alt+S]
Asia Pacific (Mumbai)
Anoora @ sadiaaws
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups (5) Info
Actions
Export security groups to CSV
Create security group
Find security groups by attribute or tag
Name
Security group ID
Security group name
VPC ID
Description
sg-057e1b280f97d3d9c default vpc-0b1d84b2e30ec29c9 default VPC secu
sg-06d9bcd52206646db public-sg vpc-0b1d84b2e30ec29c9 Allow web and ss
sg-036f4faed02264117 alb-sg vpc-0b1d84b2e30ec29c9 security grp for A
sg-05869bd192d2d07f6 private-sg vpc-0b1d84b2e30ec29c9 Allow web tier an
sg-0db96cde0262d1d49 default vpc-0e26cdceb3a5dde7e default VPC secu


Not secure
two-tier-alb-1963757824.ap-south-1.elb.amazonaws.com
New Chrome available
Welcome to nginx!
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.




- **terraform destroy**



```
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 1m30s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 1m40s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 1m50s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 2m0s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 2m10s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 2m20s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 2m30s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 2m40s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 2m50s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 3m0s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 3m10s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 3m20s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 3m30s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 3m40s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 3m50s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 4m0s elapsed]
aws_db_instance.mydatabase: Still destroying... [id=db-YARUCU24CTYDKSZFKJNN26VNJY, 4m10s elapsed]
aws_db_instance.mydatabase: Destruction complete after 4m13s
aws_db_subnet_group.db_subnet: Destroying... [id=db_subnet]
aws_security_group.private_sg: Destroying... [id=sg-05869bd192d2d07f6]
aws_db_subnet_group.db_subnet: Destruction complete after 0s
aws_subnet.private_1: Destroying... [id=subnet-091dba6c43c1da831]
aws_subnet.private_2: Destroying... [id=subnet-0629925f6e0bebd5d]
aws_subnet.private_1: Destruction complete after 1s
aws_subnet.private_2: Destruction complete after 1s
aws_security_group.private_sg: Destruction complete after 1s
aws_security_group.public_sg: Destroying... [id=sg-06d9bcd52206646db]
aws_security_group.public_sg: Destruction complete after 1s
aws_vpc.vpc: Destroying... [id=vpc-0b1d84b2e30ec29c9]
aws_vpc.vpc: Destruction complete after 0s

Destroy complete! Resources: 21 destroyed.
PS C:\Users\admin\Desktop\2-tier-aws-terraform>
```