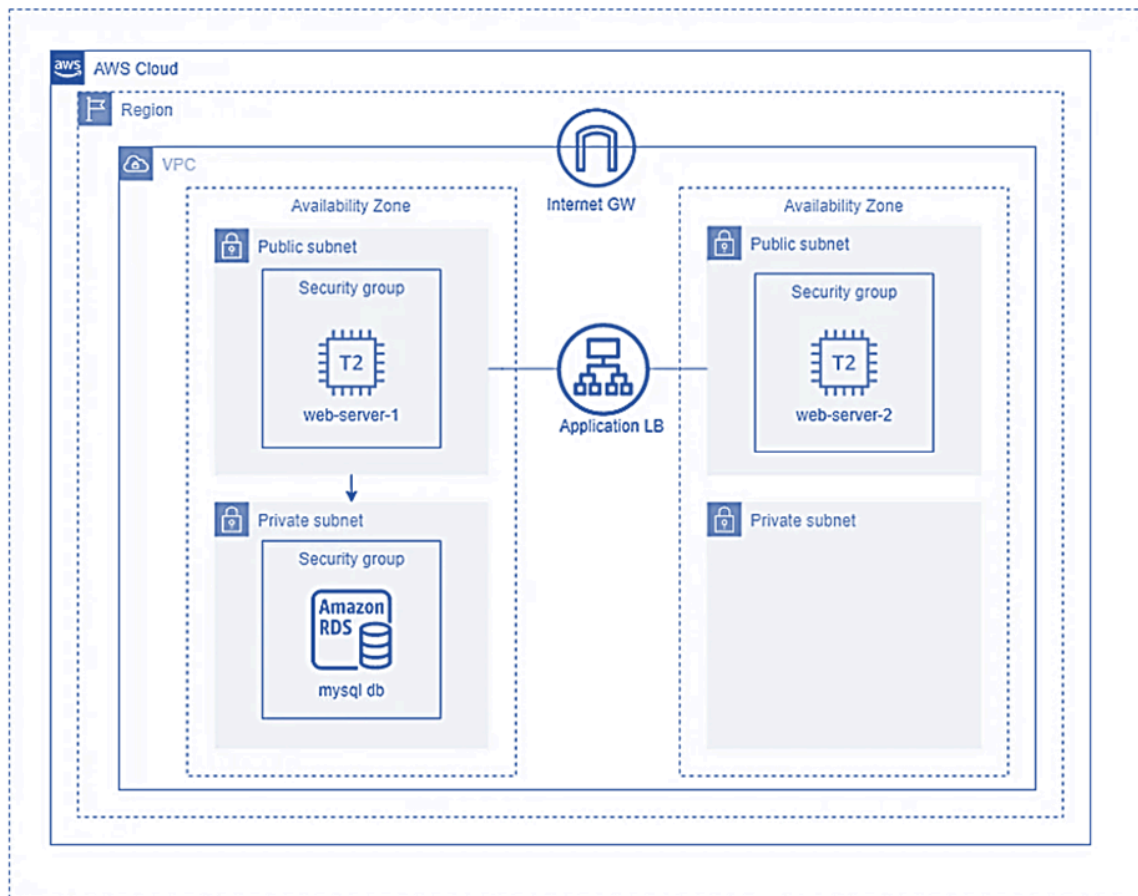


# Deploy Two Tier Architecture in AWS using Terraform



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]
```

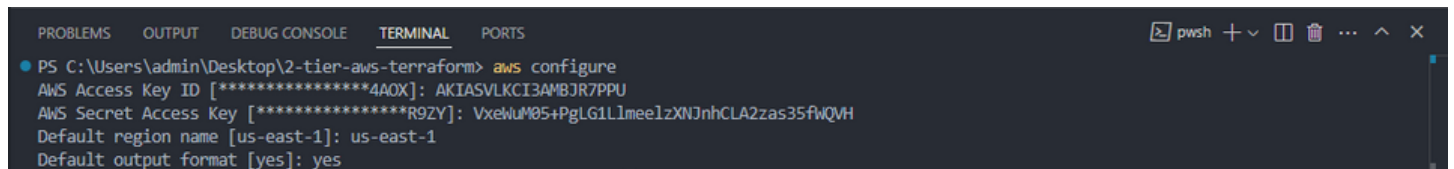
```
publicly_accessible = false
```

```
skip_final_snapshot = true
```

```
}
```

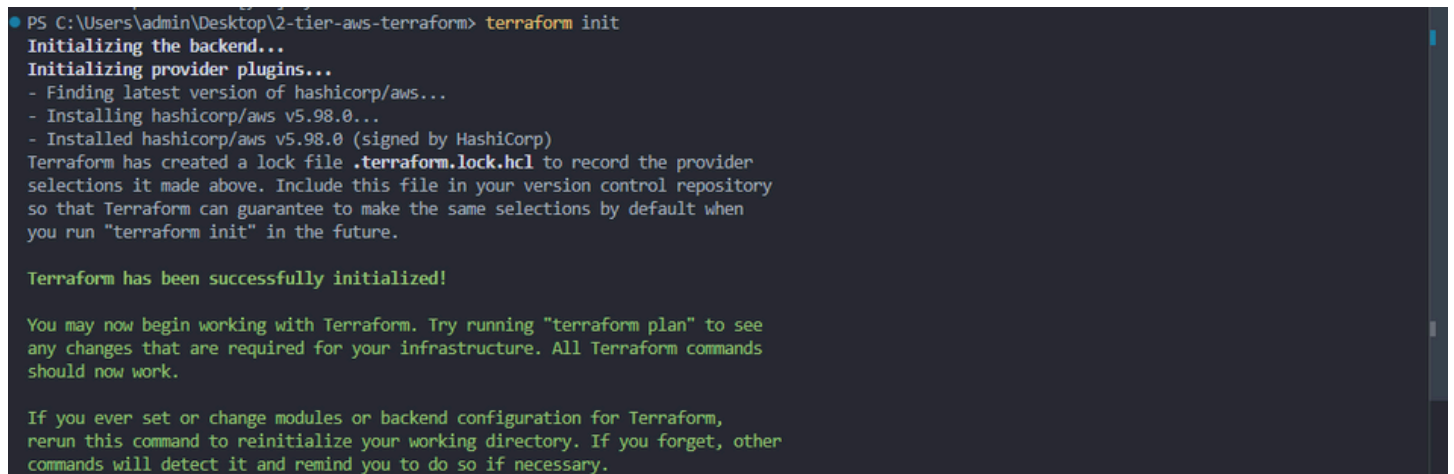
## Let's deploy!

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



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
● 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



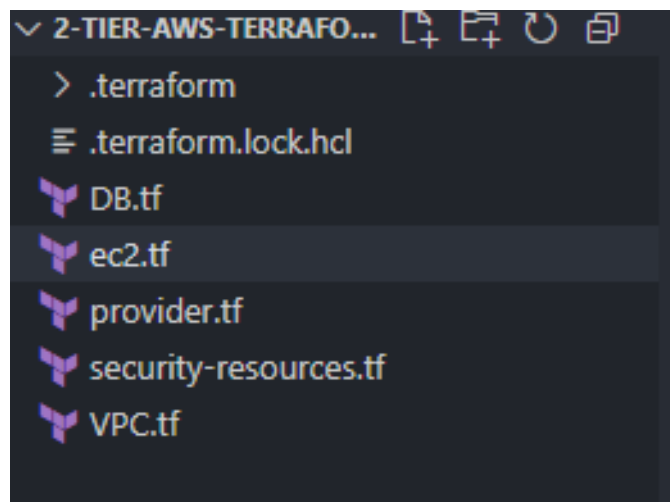
```
● PS C:\Users\admin\Desktop\2-tier-aws-terraform> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.98.0...
- Installed hashicorp/aws v5.98.0 (signed by HashiCorp)
Terraform has created a lock file .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!

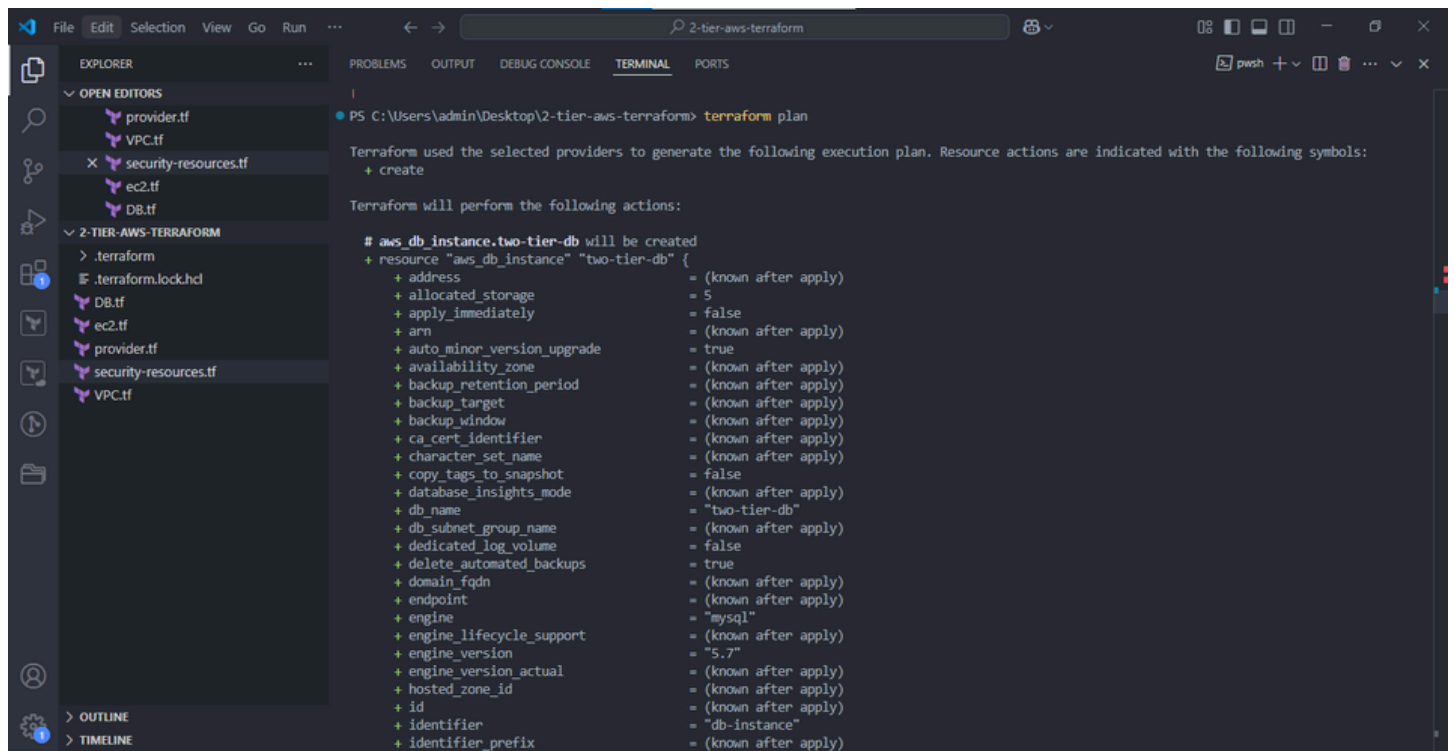
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

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



This screenshot shows the VS Code interface with the Explorer, Problems, Output, Debug Console, and Terminal panels. The Explorer panel on the left shows the project structure with files like provider.tf, VPC.tf, security-resources.tf, ec2.tf, DB.tf, and 2-TIER-AWS-TERRAFORM. The main editor displays the Terraform configuration for an AWS database instance, with the following content:

```
+ 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)
```

This screenshot shows the VS Code interface with the Explorer, Problems, Output, Debug Console, and Terminal panels. The Explorer panel on the left shows the project structure with files like provider.tf, VPC.tf, security-resources.tf, ec2.tf, DB.tf, and 2-TIER-AWS-TERRAFORM. The main editor displays the Terraform configuration for an AWS subnet and instance, with the following content:

```
# 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)
```

This screenshot shows the VS Code interface with the Explorer, Output, and Terminal panels. The Explorer panel on the left shows a project structure for '2-TIER-AWS-TERRAFORM' with files like 'provider.tf', 'VPC.tf', 'security-resources.tf', 'ec2.tf', and 'DB.tf'. The Output panel in the center displays the Terraform configuration for the 'aws\_instance.webserver\_2' resource. The configuration includes various attributes such as 'ami', 'arn', 'associate\_public\_ip\_address', 'availability\_zone', 'cpu\_core\_count', 'cpu\_threads\_per\_core', 'disable\_api\_stop', 'disable\_api\_termination', 'ebs\_optimized', 'enable\_primary\_ipv6', 'get\_password\_data', 'host\_id', 'host\_resource\_group\_arn', 'iam\_instance\_profile', 'id', 'instance\_initiated\_shutdown\_behavior', 'instance\_lifecycle', 'instance\_state', 'instance\_type', 'ipv6\_address\_count', 'ipv6\_addresses', 'key\_name', 'monitoring', 'outpost\_arn', 'password\_data', 'placement\_group', 'placement\_partition\_number', 'primary\_network\_interface\_id', 'private\_dns', and 'private\_ip'. The Terminal panel on the right shows the command 'terraform apply' being executed.

```
File Edit Selection View Go Run ... 2-tier-aws-terraform
```

EXPLORER

- OPEN EDITORS
  - provider.tf
  - VPC.tf
  - security-resources.tf
  - ec2.tf
  - DB.tf
- 2-TIER-AWS-TERRAFORM
  - .terraform
  - .terraform.lock.hcl
  - DB.tf
  - ec2.tf
  - provider.tf
  - security-resources.tf
  - VPC.tf
- OUTLINE
- TIMELINE

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
+ 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)
}
```

psush + - ...

This screenshot shows the VS Code interface with the Explorer, Output, and Terminal panels. The Explorer panel on the left shows the same project structure as the previous screenshot. The Output panel in the center displays the Terraform configuration for the 'aws\_vpc.vpc' resource. The configuration includes various attributes such as 'arn', 'cidr\_block', 'default\_network\_acl\_id', 'default\_route\_table\_id', 'default\_security\_group\_id', 'dhcp\_options\_id', 'enable\_dns\_hostnames', 'enable\_dns\_support', 'enable\_network\_address\_usage\_metrics', 'id', 'instance\_tenancy', 'ipv6\_association\_id', 'ipv6\_cidr\_block', 'ipv6\_cidr\_block\_network\_border\_group', 'main\_route\_table\_id', 'owner\_id', 'tags', and 'tags\_all'. The Terminal panel on the right shows the command 'terraform apply' being executed. Below the configuration, a note states: '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.'

```
File Edit Selection View Go Run ... 2-tier-aws-terraform
```

EXPLORER

- OPEN EDITORS
  - provider.tf
  - VPC.tf
  - security-resources.tf
  - ec2.tf
  - DB.tf
- 2-TIER-AWS-TERRAFORM
  - .terraform
  - .terraform.lock.hcl
  - DB.tf
  - ec2.tf
  - provider.tf
  - security-resources.tf
  - VPC.tf
- OUTLINE
- TIMELINE

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
}
+ 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>
```

psush + - ...

Ln 149, Col 1 (3385 selected) Spaces: 2 UTF-8 CRLF () Terraform Prettier

- terraform apply

The screenshot shows the Visual Studio Code interface with the Explorer pane on the left displaying a project structure for '2-TIER-AWS-TERRAFORM'. The main editor shows the terminal output of the 'terraform apply' command. The output includes a note about the '-out' option, the command execution path, and a list of resources to be created. The resource 'aws\_db\_instance.two-tier-db' is highlighted with its configuration details.

```
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

The screenshot shows the Visual Studio Code interface with the terminal output of the 'terraform apply' command. The output displays the plan for creating various resources, followed by a confirmation prompt 'Do you want to perform these actions?'. The user enters 'yes', and the resources are created successfully. The output lists the creation of 'aws\_vpc.vpc', 'aws\_subnet.private\_2', 'aws\_subnet.private\_1', 'aws\_internet\_gateway.igw', 'aws\_lb\_target\_group.alb-tg', 'aws\_subnet.public\_1', 'aws\_subnet.public\_2', 'aws\_security\_group.alb-sg', 'aws\_security\_group.public-sg', 'aws\_internet\_gateway.igw', 'aws\_route\_table.rt-igw', 'aws\_subnet.private\_2', and 'aws\_subnet.private\_1'.

```
+ 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"
}

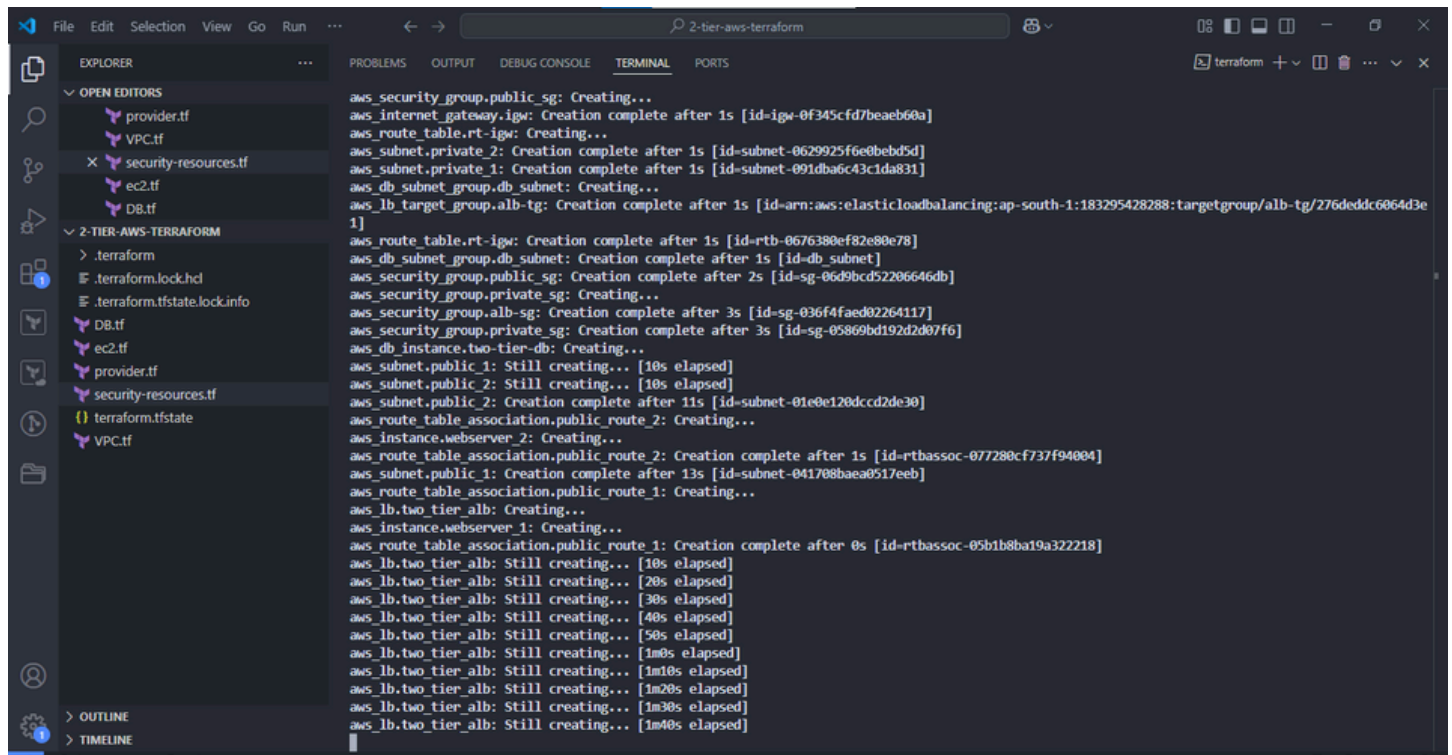
Plan: 21 to add, 0 to change, 0 to destroy.

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]
```

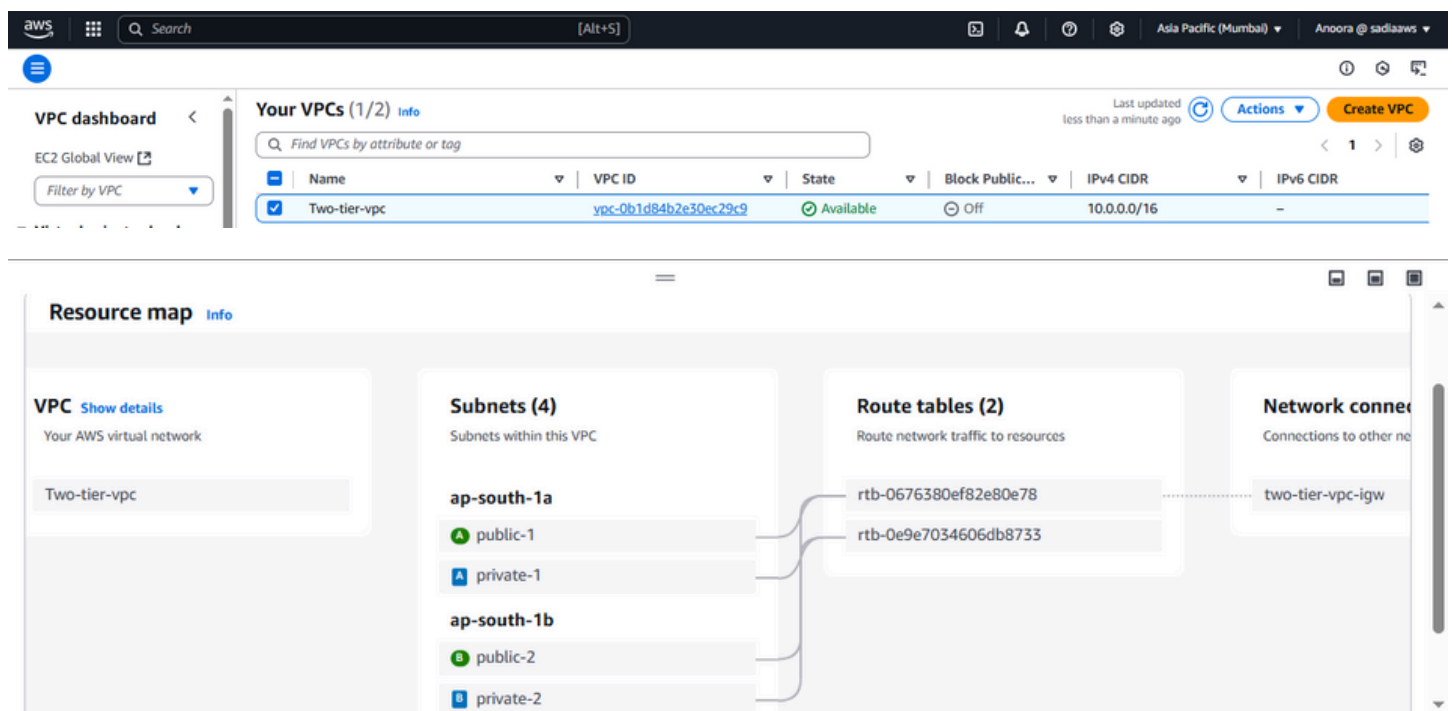




```
aws_security_group.public_sg: Creating...
aws_internet_gateway.igw: Creation complete after 1s [id=igw-0f345cfd7beaeb60a]
aws_route_table.rt-igw: Creating...
aws_subnet.private_2: Creation complete after 1s [id=subnet-0629925f6e0ebd5d]
aws_subnet.private_1: Creation complete after 1s [id=subnet-091dba6c43c1da831]
aws_db_subnet_group.db_subnet: Creating...
aws_lb_target_group.alb-tg: Creation complete after 1s [id=arn:aws:elasticloadbalancing:ap-south-1:183295428288:targetgroup/alb-tg/276deddc6064d3e1]
aws_route_table.rt-igw: Creation complete after 1s [id=rtb-0676380ef82e80e78]
aws_db_subnet_group.db_subnet: Creation complete after 1s [id=db_subnet]
aws_security_group.public_sg: Creation complete after 2s [id=sg-06d9bcd52206646db]
aws_security_group.private_sg: Creating...
aws_security_group.alb_sg: Creation complete after 3s [id=sg-036f4faed02264117]
aws_security_group.private_sg: Creation complete after 3s [id=sg-05869bd192d2d07f6]
aws_db_instance.two-tier-db: Creating...
aws_subnet.public_1: Still creating... [10s elapsed]
aws_subnet.public_2: Still creating... [10s elapsed]
aws_subnet.public_2: Creation complete after 11s [id=subnet-01e0e120dcd2de30]
aws_route_table_association.public_route_2: Creating...
aws_instance.websvr_2: Creating...
aws_route_table_association.public_route_2: Creation complete after 1s [id=rtbassoc-077280cf737f94004]
aws_subnet.public_1: Creation complete after 13s [id=subnet-041708bae0517eeb]
aws_route_table_association.public_route_1: Creating...
aws_lb.two_tier_alb: Creating...
aws_instance.websvr_1: Creating...
aws_route_table_association.public_route_1: Creation complete after 0s [id=rtbassoc-05b1b8ba19a322218]
aws_lb.two_tier_alb: Still creating... [10s elapsed]
aws_lb.two_tier_alb: Still creating... [20s elapsed]
aws_lb.two_tier_alb: Still creating... [30s elapsed]
aws_lb.two_tier_alb: Still creating... [40s elapsed]
aws_lb.two_tier_alb: Still creating... [50s elapsed]
aws_lb.two_tier_alb: Still creating... [1m0s elapsed]
aws_lb.two_tier_alb: Still creating... [1m10s elapsed]
aws_lb.two_tier_alb: Still creating... [1m20s elapsed]
aws_lb.two_tier_alb: Still creating... [1m30s elapsed]
aws_lb.two_tier_alb: Still creating... [1m40s elapsed]
```

## Go to the AWS console and verify

1. VPC and Network resources



2. EC2 instances



**Instances (2)**

Find Instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
webserver-2	i-084bc9a8213732a50	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b	-
webserver-1	i-0a4288859cb0ed0c6	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a	-

### 3. Load Balancer

**Load balancers (1)**

Filter load balancers

Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
two-tier-alb	two-tier-alb-1963757824.a...	Active	vpc-0b1d84b2e30ec29c9	2 Availability Zones	application	May 20, 20...

### 4. Target Group

**Target groups (1)**

Filter target groups

Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
alb-tg	arn:aws:elasticloadbalancin...	80	HTTP	Instance	two-tier-alb	vpc-0b1

0 target groups selected

### 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)

Filter by databases

Group resources

Modify

Actions

Create database

DB identifier	Status	Role	Engine	Region	Size
<a href="#">db-instance</a>	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)

Filter by subnet group

Edit

Delete

Create DB subnet group

Name	Description	Status	VPC
<a href="#">db_subnet</a>	Managed by Terraform	Complete	vpc-0b1d84b2e30ec29c9

Day- | Dep- | Day- | RDS | Sora | (12) | Terra | Mk Auto | Inbo | 1-tie | Insta | Dev | Terra | map | +

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.

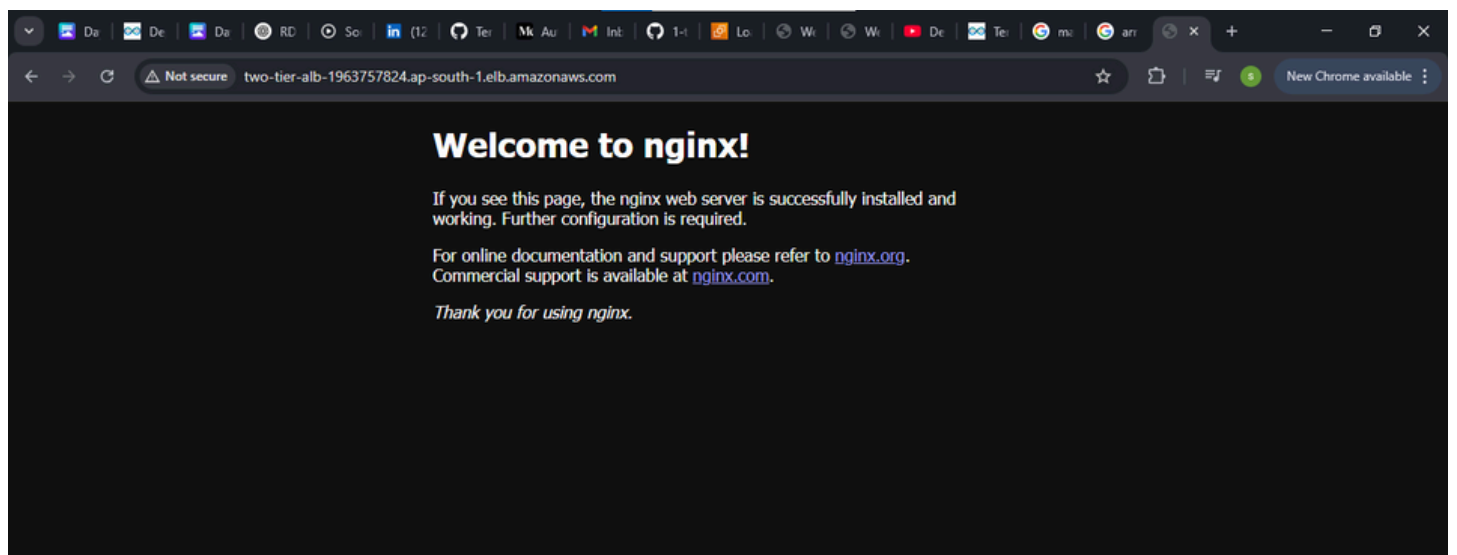
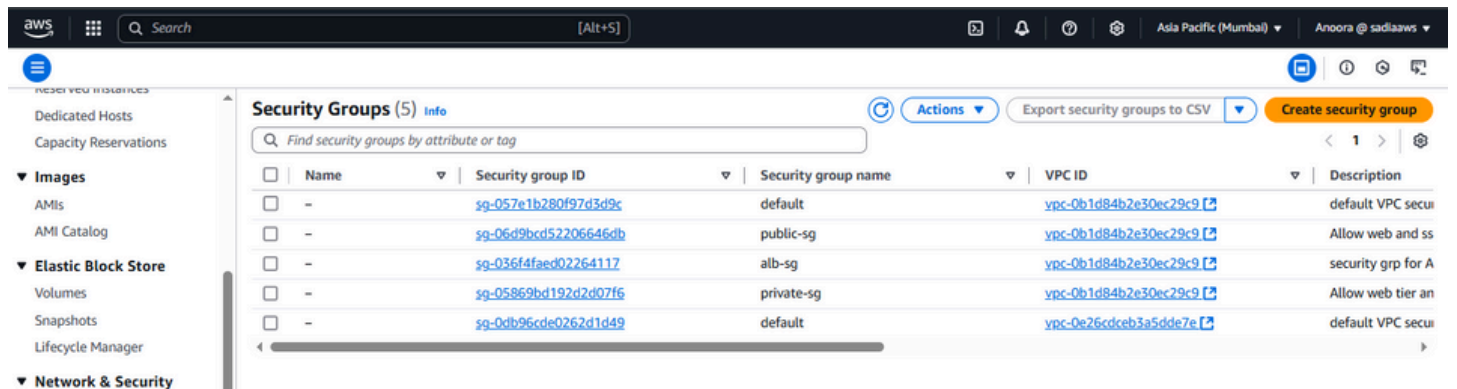
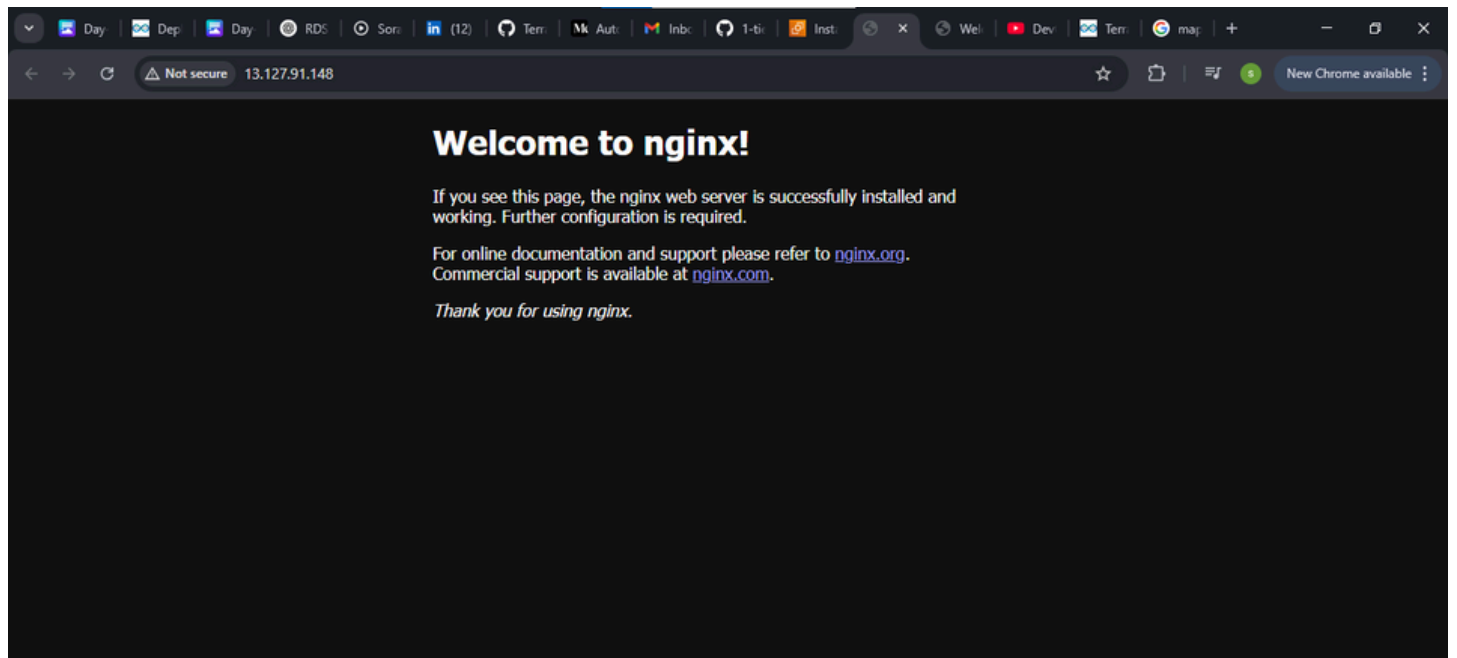
For online documentation and support please refer to [nginx.org](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

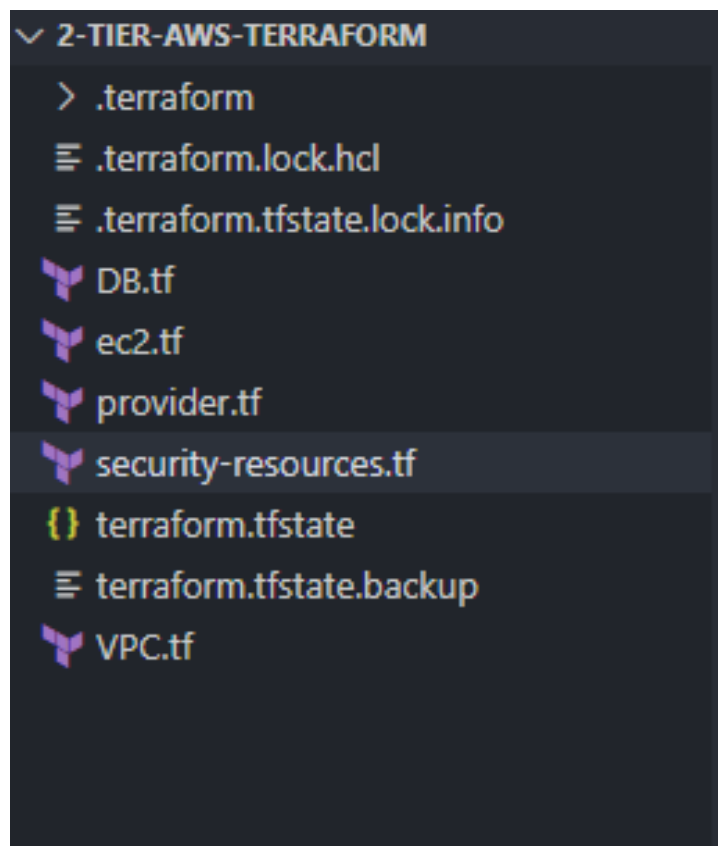
*Thank you for using nginx.*

Type here to search

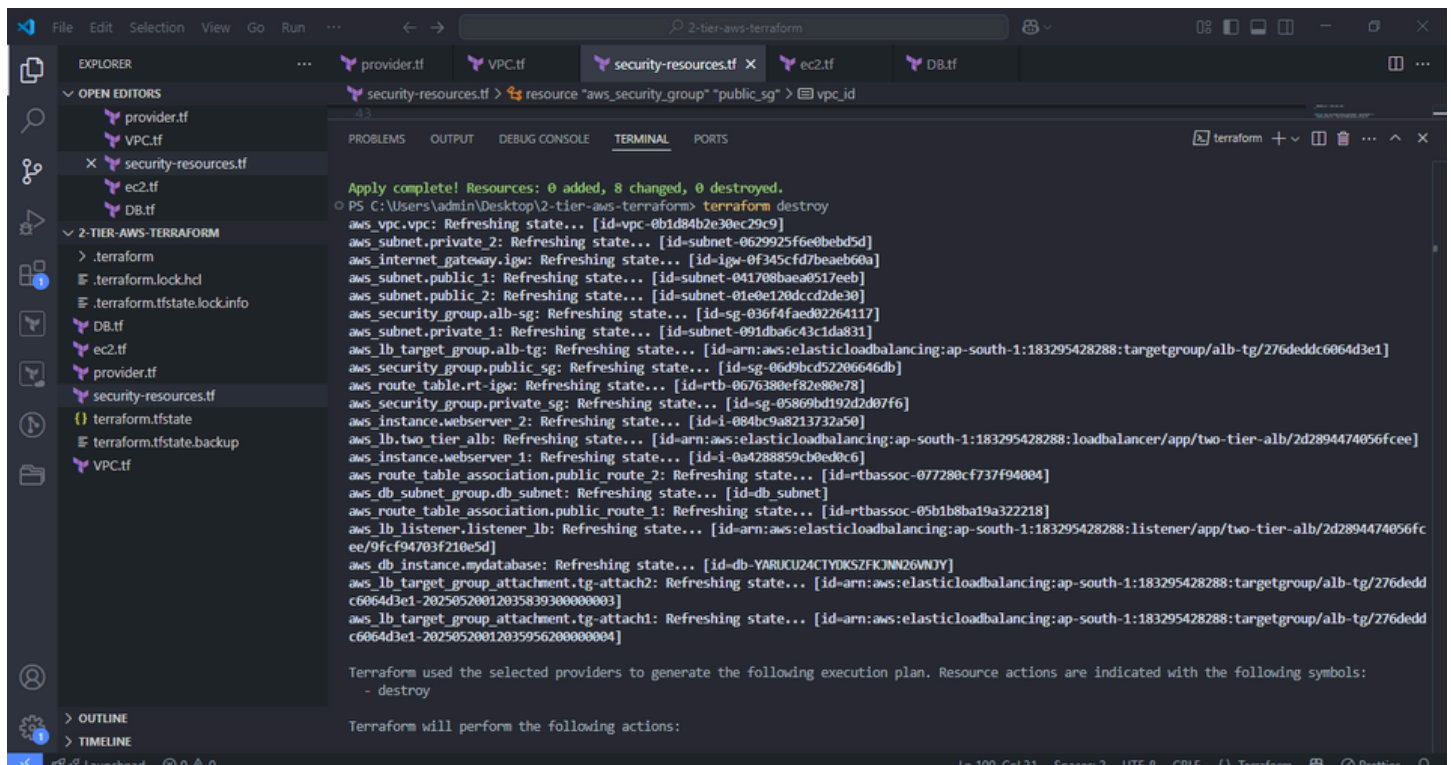
Air q...

ENG IN 7:16 AM 5/20/2025





- terraform destroy



File Edit Selection View Go Run ... 2-tier-aws-terraform

EXPLORER

- OPEN EDITORS
  - provider.tf
  - VPC.tf
  - security-resources.tf x
  - ec2.tf
  - DB.tf
- 2-TIER-AWS-TERRAFORM
  - .terraform
  - .terraform.lock.hcl
  - DB.tf
  - ec2.tf
  - provider.tf
  - security-resources.tf
  - terraform.tfstate
  - terraform.tfstate.backup
  - VPC.tf
- OUTLINE
- TIMELINE

security-resources.tf > resource "aws\_security\_group" "public\_sg" > vpc\_id

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
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: 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>
```

Ln 100, Col 31 Spaces: 2 UTF-8 CRLF Terraform Prettier