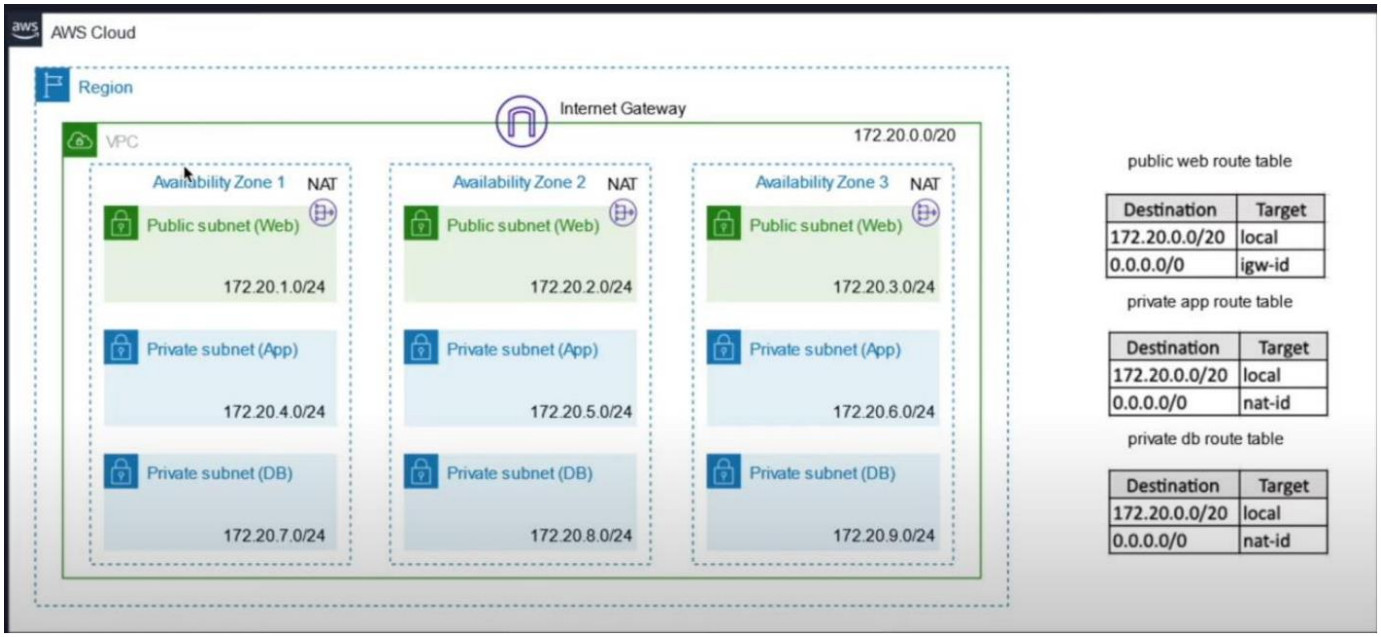
3-TIER ARCHITECHTURE PROJECT USING TERRAFORM

The aim of this project is to use terraform infrastructure as a code (IAC) to design and deploy a highly secure and scalable three-tier application infrastructure on Amazon Web Services (AWS) for an online pharmaceutical store. The primary objective is to establish a robust Virtual Private Cloud (VPC) that ensures the web application's continuous operation while adhering to stringent healthcare sector privacy and security standards.

The infrastructure required the creation of distinct network layers, including public subnets for internet-facing components and private subnets for the application and database layers. Key components of this architecture included an Internet Gateway (IGW), a Network Address Translation (NAT) Gateway, and an Application Load Balancer (ALB), all strategically positioned across multiple Availability Zones (AZs) to guarantee high availability and fault tolerance.

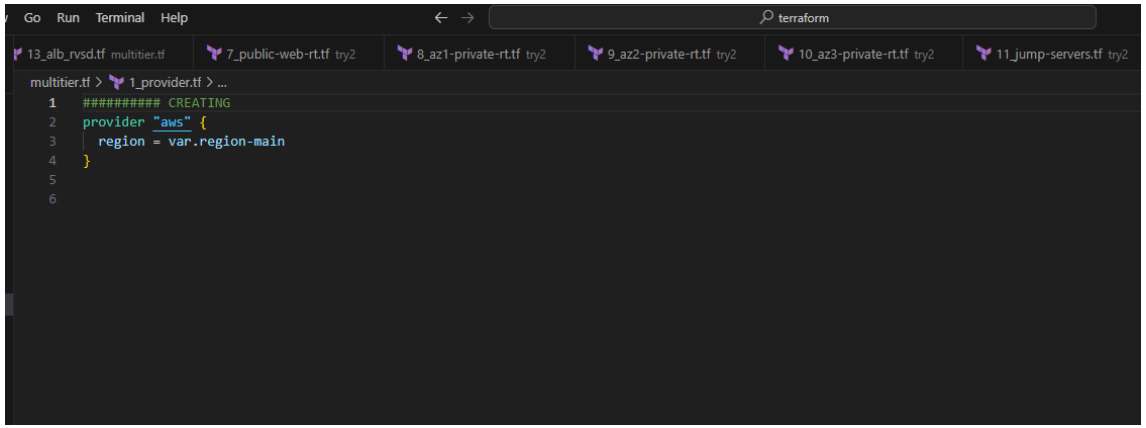
As part of the deployment process, the project involved setting up EC2 instances within these subnets, with a public-facing jump server in the public subnet and application and database servers in the private subnets. The application servers were bootstrapped with a lampstack script, excluding the database setup, as an RDS instance was employed to meet the database needs.



`

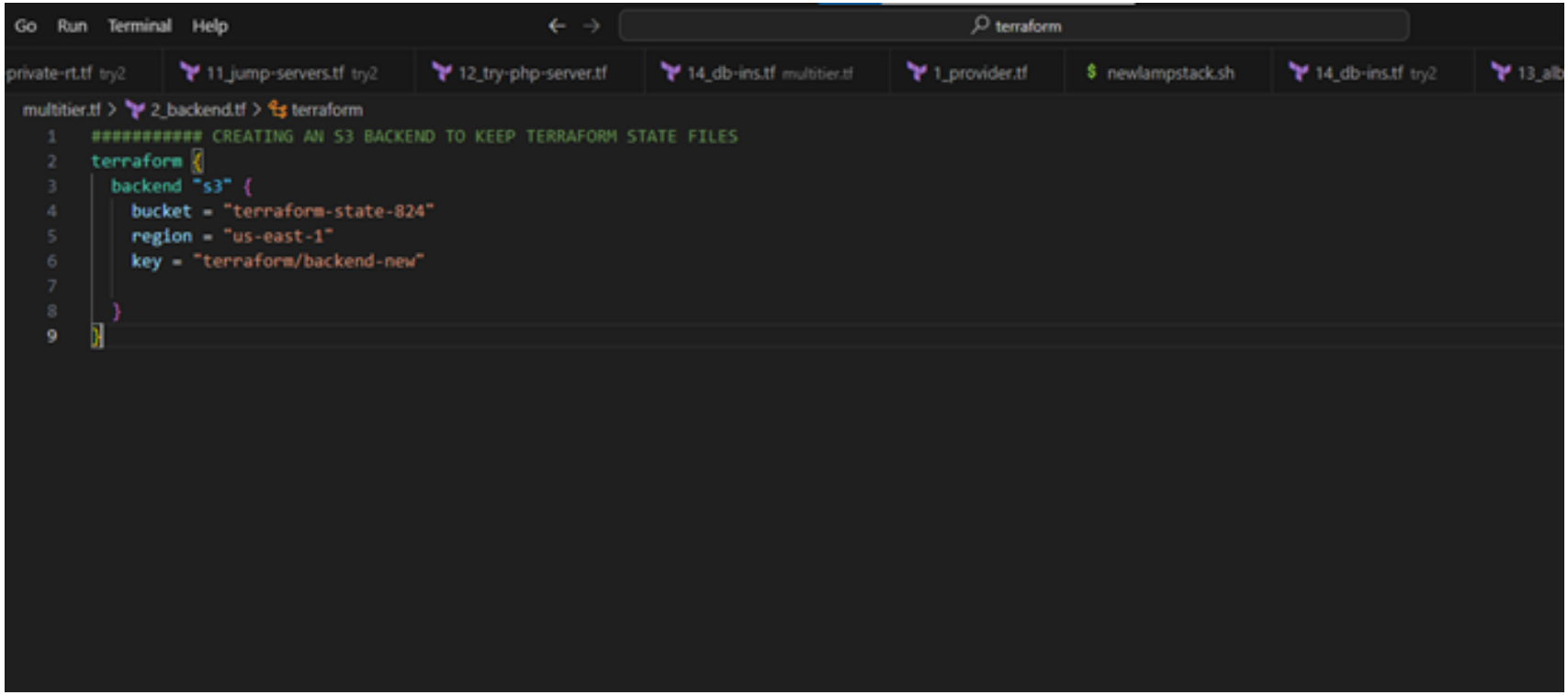
1. **<u>CREATED A PROVISIONER:</u> *<u>1_provider.tf</u>***

The provider block configures Terraform to interact with AWS, specifying the region where all resources will be provisioned. By setting the region to “var.region”-main, this block ensures that all resources are deployed in the designated AWS region, aligning with the project's geographical and compliance requirements.



`

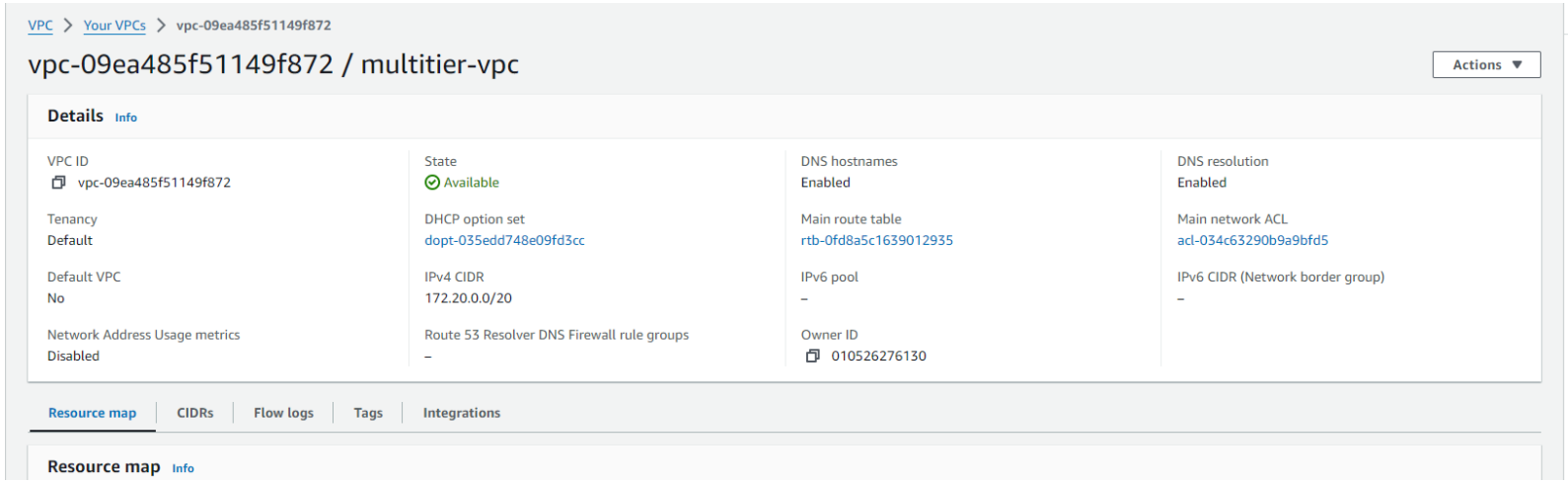
2. **<u>CREATING S3 BACKEND TO STORE TERRAFORM STATE FILES – *2_backend.tf*</u>**
   The S3 backend is crucial for storing the Terraform state files, which maintain the mapping between the resources in the configuration and the real-world resources in AWS. By storing these state files in an S3 bucket, the Terraform environment is set up for collaboration, allowing multiple team members to access the state information in a secure and scalable manner. This setup also facilitates the use of versioning to manage changes to the infrastructure over time.



`

3. **CREATING A VPC TO ACCOMMODATE THE ARCHITECTURE: *3_vpc.tf***

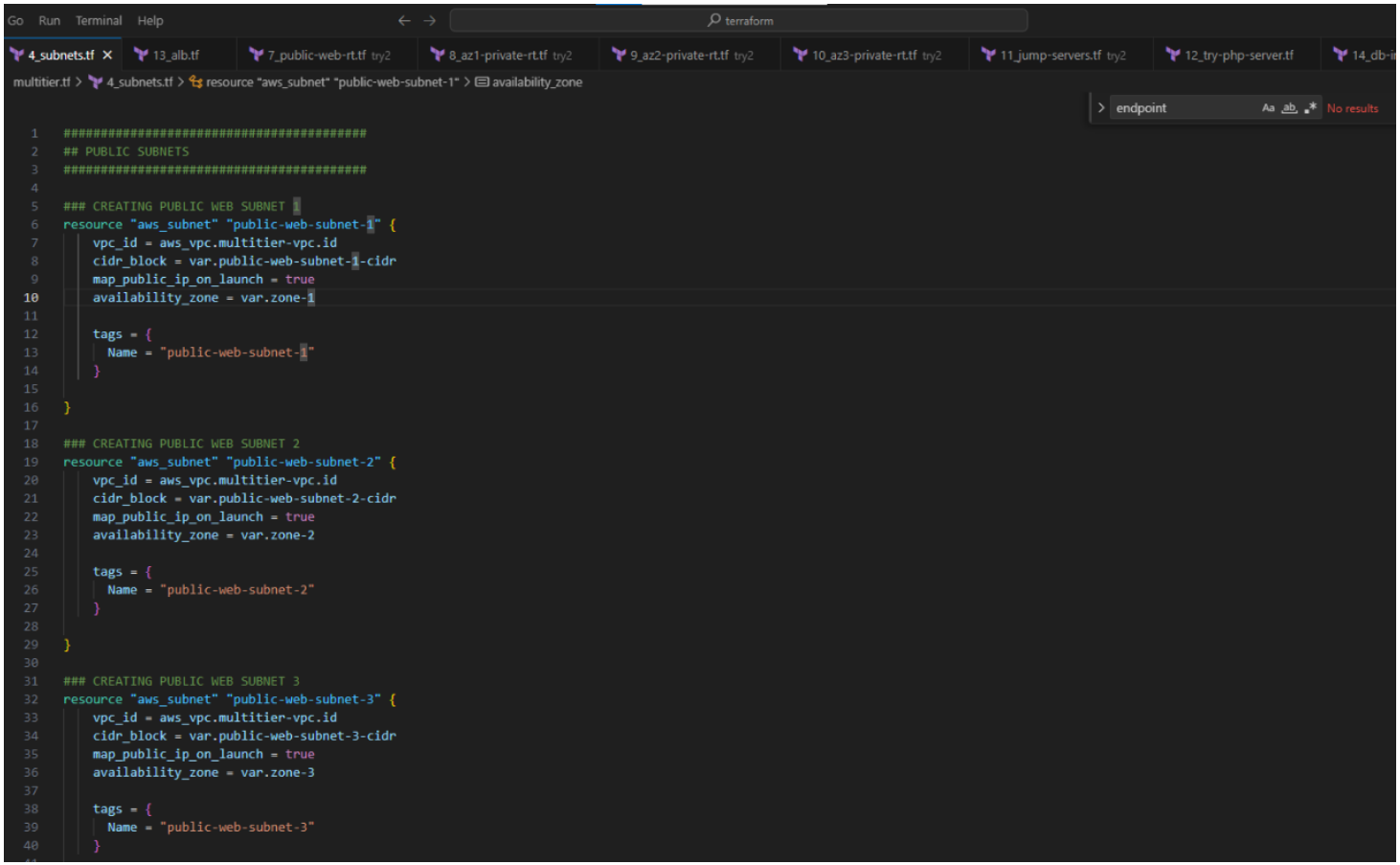
Virtual Private Cloud (VPC) will be used to set up an isolated network environment that provides full control over IP addressing, subnets, routing, and network gateways. This VPC serves as the environment where all other resources in architecture will reside.



`

4. CREATING PUBLIC SUBNETS TO HOST JUMP SERVERS: *4_subnets.tf*

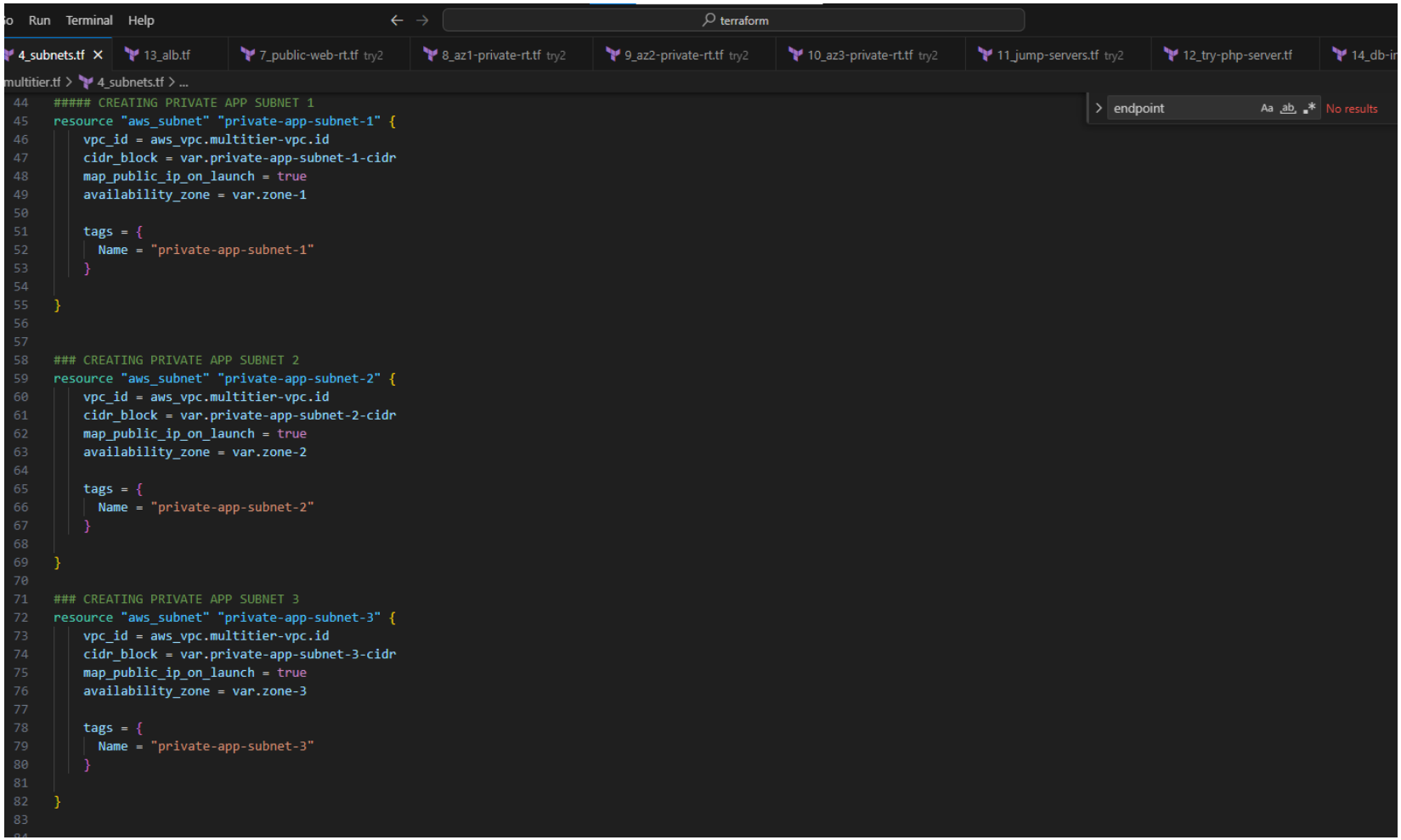
Public subnets are designed to host the jump-servers that need to be accessible from the internet. The jump server acts as a secure gateway through which administrators can access servers in private subnets. By placing the jump server in a public subnet, you allow SSH access to it, while keeping your application and database servers isolated in private subnets.



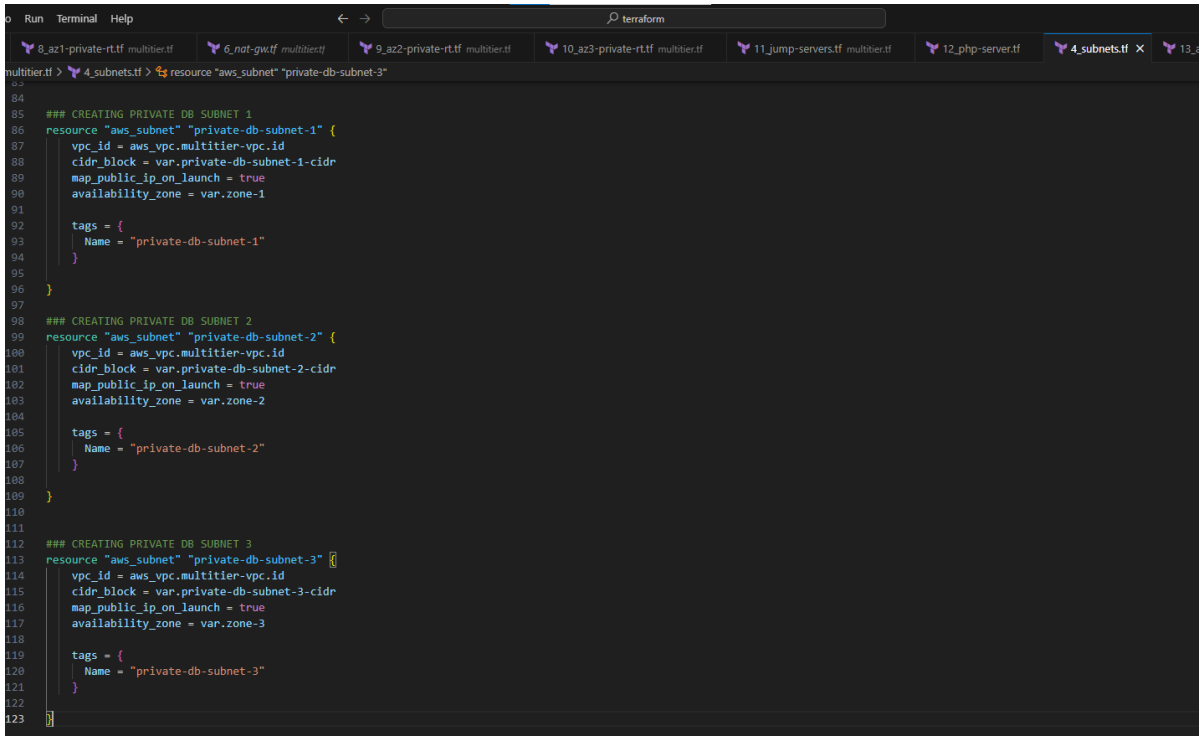
`

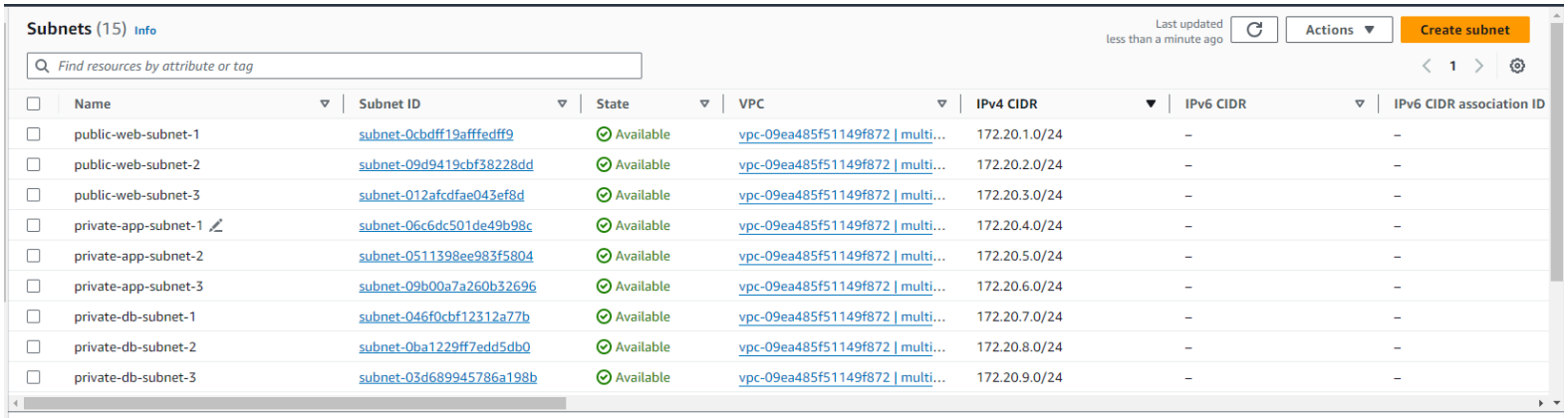
5. **<u>CREATING PRIVATE SUBNETS TO HOST PHP AND DB SERVERS: *4_subnet.tf*</u>**

Private subnets are used to host sensitive components of the application that should not be directly accessible from the internet. In this project, the PHP application server and the database server are placed in private subnets to ensure they are protected from external threats, while still being able to communicate with each other and the public subnet via appropriate routing rules.



`

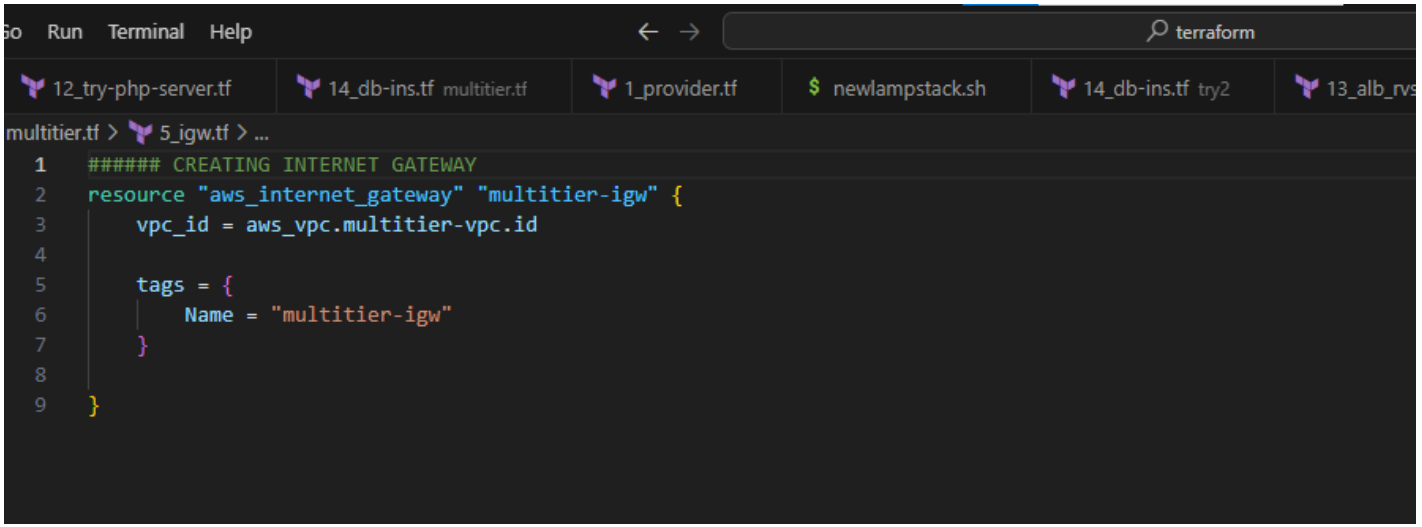

Run Terminal Help
terraform
8_az1-private-rt.tf multitier.tf
6_nat-gw.tf multitier.tf
9_az2-private-rt.tf multitier.tf
10_az3-private-rt.tf multitier.tf
11_jump-servers.tf multitier.tf
12_php-server.tf
4_subnets.tf
multitier.tf > 4_subnets.tf > resource "aws_subnet" "private-db-subnet-3"
CREATING PRIVATE DB SUBNET 1
resource "aws_subnet" "private-db-subnet-1" {
vpc_id = aws_vpc.multitier-vpc.id
cidr_block = var.private-db-subnet-1-cidr
map_public_ip_on_launch = true
availability_zone = var.zone-1
tags = {
Name = "private-db-subnet-1"
}
}
CREATING PRIVATE DB SUBNET 2
resource "aws_subnet" "private-db-subnet-2" {
vpc_id = aws_vpc.multitier-vpc.id
cidr_block = var.private-db-subnet-2-cidr
map_public_ip_on_launch = true
availability_zone = var.zone-2
tags = {
Name = "private-db-subnet-2"
}
}
CREATING PRIVATE DB SUBNET 3
resource "aws_subnet" "private-db-subnet-3" {
vpc_id = aws_vpc.multitier-vpc.id
cidr_block = var.private-db-subnet-3-cidr
map_public_ip_on_launch = true
availability_zone = var.zone-3
tags = {
Name = "private-db-subnet-3"
}
}

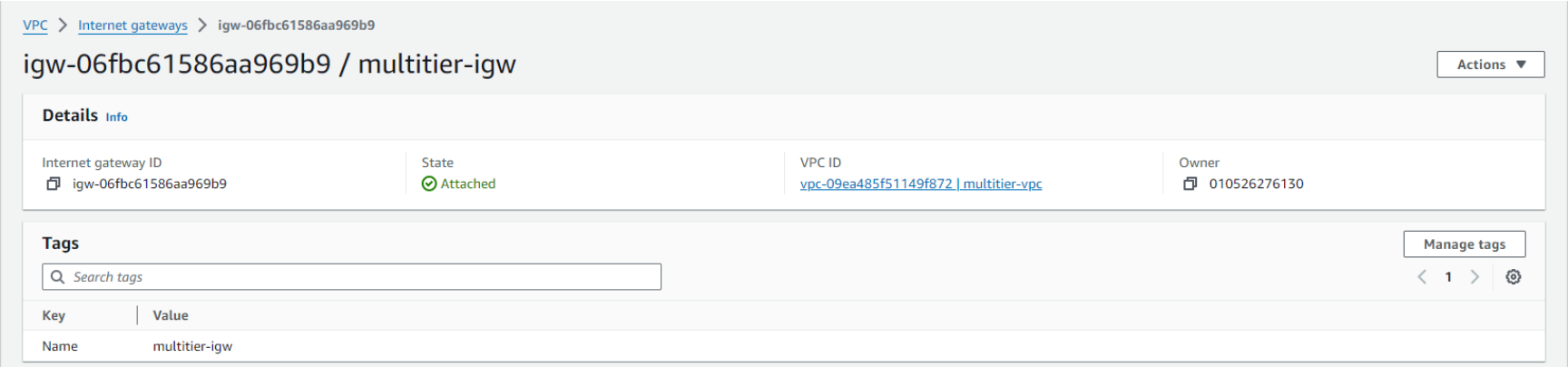

Subnets (15) Info
Last updated less than a minute ago
Actions
Create subnet
Find resources by attribute or tag
1
Name
Subnet ID
State
VPC
IPv4 CIDR
IPv6 CIDR
IPv6 CIDR association ID
public-web-subnet-1 subnet-0cbdff19afffedff9 Available vpc-09ea485f51149f872 | multi... 172.20.1.0/24 – –
public-web-subnet-2 subnet-09d9419cbf38228dd Available vpc-09ea485f51149f872 | multi... 172.20.2.0/24 – –
public-web-subnet-3 subnet-012afcdfae043ef8d Available vpc-09ea485f51149f872 | multi... 172.20.3.0/24 – –
private-app-subnet-1 subnet-06c6dc501de49b98c Available vpc-09ea485f51149f872 | multi... 172.20.4.0/24 – –
private-app-subnet-2 subnet-0511398ee983f5804 Available vpc-09ea485f51149f872 | multi... 172.20.5.0/24 – –
private-app-subnet-3 subnet-09b00a7a260b32696 Available vpc-09ea485f51149f872 | multi... 172.20.6.0/24 – –
private-db-subnet-1 subnet-046f0cbf12312a77b Available vpc-09ea485f51149f872 | multi... 172.20.7.0/24 – –
private-db-subnet-2 subnet-0ba1229ff7edd5db0 Available vpc-09ea485f51149f872 | multi... 172.20.8.0/24 – –
private-db-subnet-3 subnet-03d689945786a198b Available vpc-09ea485f51149f872 | multi... 172.20.9.0/24 – –

`

6. **<u>CREATING INTERNET GATEWAY TO ALLOW CONNECTION BETWEEN THE INTERNET AND THE VPC: *5_igw.tf*</u>**

The Internet Gateway (IGW) is attached to the VPC to enable communication between the instances in the VPC and the internet. This is essential for allowing users to access the application hosted in your VPC and for the jump server to communicate with the outside world. The IGW provides a path for internet traffic to reach the public subnet.

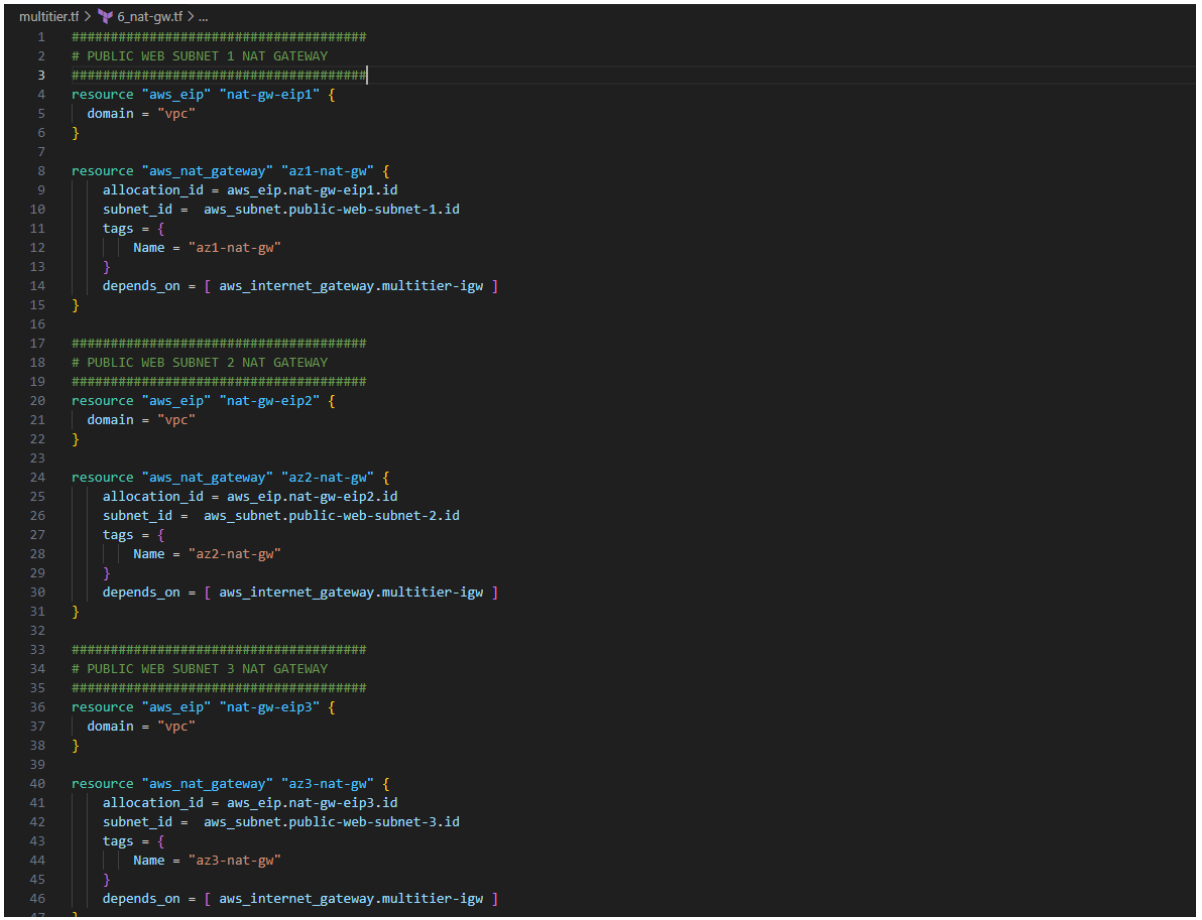




`

7. **<u>CREATING EIPS AND MAPPING THEM ONTO INDIVIDUAL NAT GATEWAYS: *6-nat-gw:*</u>**

The NAT Gateways will allow instances in private subnets to connect to the internet while preventing the internet from initiating connections with those instances. By strategically positioning NAT Gateways in each of your public subnets across different Availability Zones (AZs), high availability and fault tolerance is ensured. Each private subnet routes its outbound traffic through the NAT Gateway, Each NAT Gateway is assigned an EIP to ensure consistent outbound IP addresses by mapping an EIP to each NAT Gateway, you ensure that the gateway has a persistent address that does not change even if the NAT Gateway is replaced or restarted.",

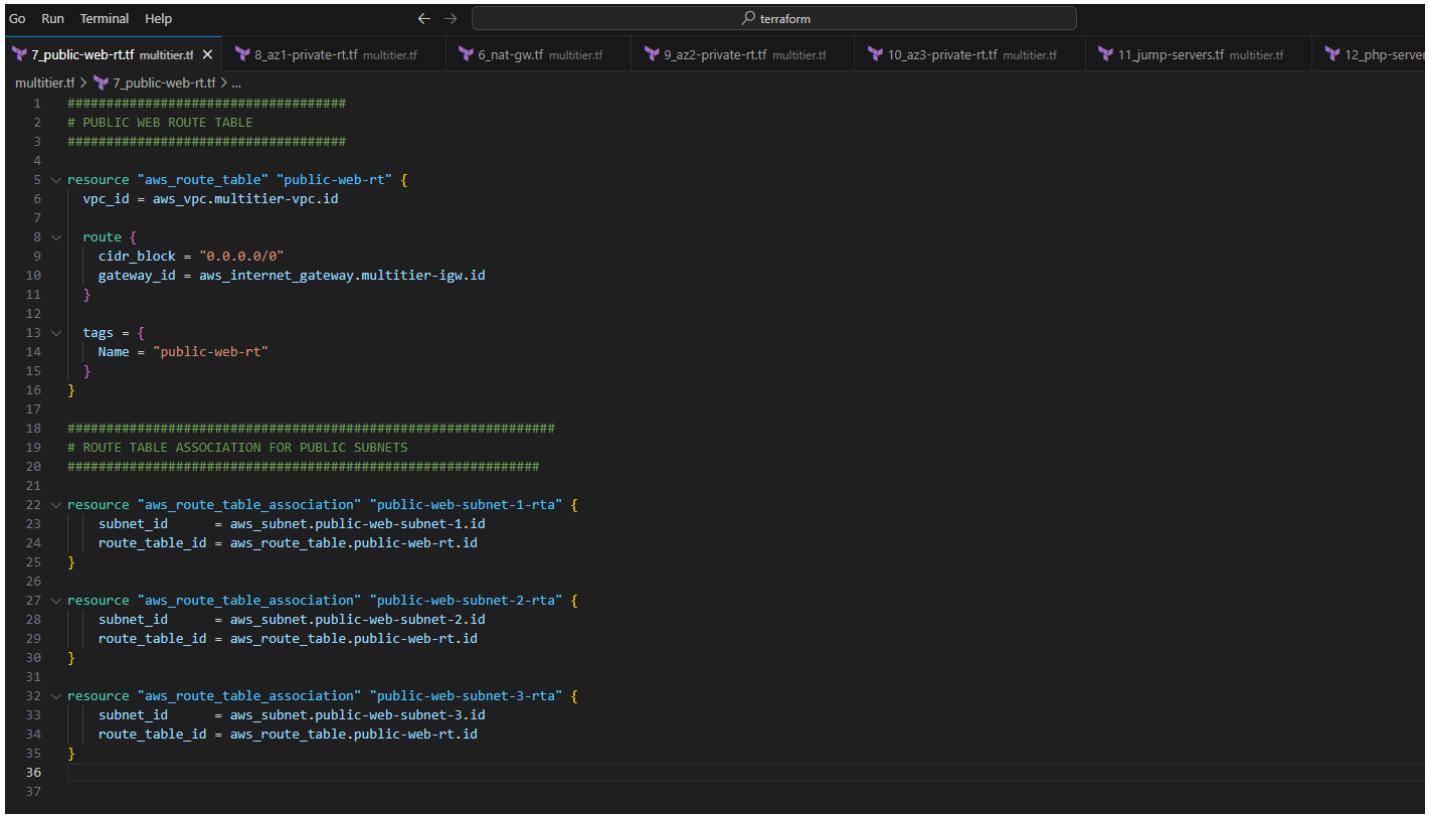


`

8. CREATING 4 ROUTE TABLES TO ROUTE TRAFFIC TO DESIRED DESTINATIONS.

Route tables will control the routing of traffic within the VPC. In this setup, one public route table is associated with the public subnets, which includes a route that directs internet-bound traffic through the Internet Gateway (IGW). This enables resources like the jump server and NAT Gateway in the public subnet to communicate with the internet. Three private route tables, each associated with a different group of two private subnets (PHP and DB) within a specific Availability Zone. Each private route table routes traffic destined for the internet through the corresponding NAT Gateway in the same Availability Zone. " This design ensures that the private instances have internet access while maintaining security and isolation from direct public internet exposure.

PUBLIC WEB ROUT TABLES



`

VPC > Route tables > rtb-04bd426e391802fb4

rtb-04bd426e391802fb4 / public-web-rt

Actions

Details Info

Route table ID	Main	Explicit subnet associations	Edge associations
rtb-04bd426e391802fb4	No	3 subnets	–

VPC	Owner ID
vpc-09ea485f51149f872 | multitier-vpc	010526276130

Routes | Subnet associations | Edge associations | Route propagation | Tags

Explicit subnet associations (3)

Edit subnet associations

Find subnet association

1

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
public-web-subnet-1	subnet-0cbdff19afffedff9	172.20.1.0/24	–
public-web-subnet-3	subnet-012afcdfae043ef8d	172.20.3.0/24	–
public-web-subnet-2	subnet-09d9419cbf38228dd	172.20.2.0/24	–

Subnets without explicit associations (0)

Edit subnet associations

The following subnets have not been explicitly associated with any route tables and are therefore associated with the main route table:

Find subnet association

1

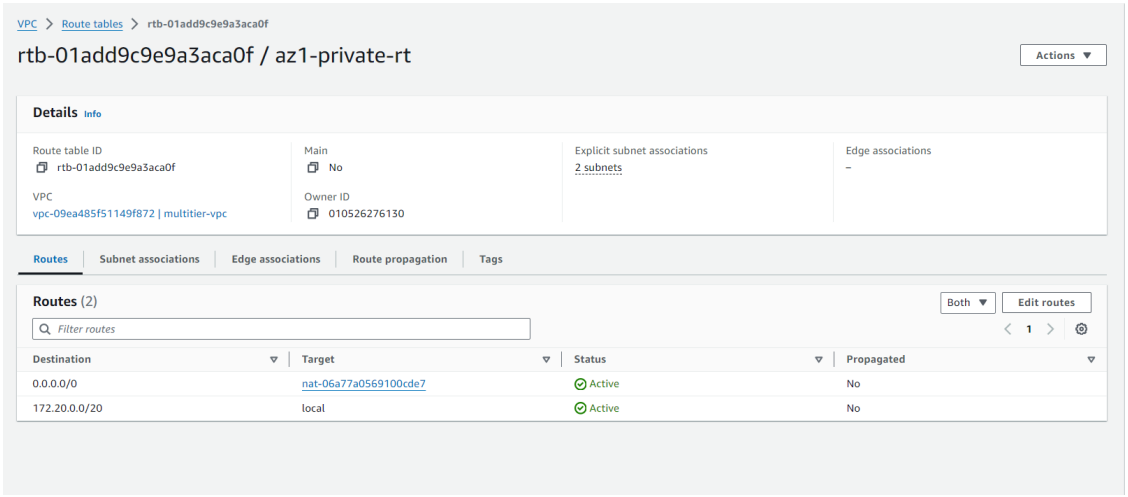
Name	Subnet ID	IPv4 CIDR	IPv6 CIDR

`

PRIVATE SUBNETS' ROUTE TABLES: *8_az1-private-rt.tf*

multitier.tf > 8_az1-private-rt.tf > ...

```
1  ###################################
2  # AZ-1 PRIVATE ROUTE TABLE
3  ###################################
4
5  resource "aws_route_table" "az1-private-rt" {
6    vpc_id = aws_vpc.multitier-vpc.id
7
8    route {
9      cidr_block = "0.0.0.0/0"
10     gateway_id = aws_nat_gateway.az1-nat-gw.id
11   }
12
13   tags = {
14     Name = "az1-private-rt"
15   }
16 }
17
18 ############################################################
19 # ROUTE TABLE ASSOCIATION FOR AZ1 PRIVATE SUBNETS
20 ############################################################
21
22 resource "aws_route_table_association" "az1-private-app-subnet-1-rta" {
23         subnet_id      = aws_subnet.private-app-subnet-1.id
24         route_table_id = aws_route_table.az1-private-rt.id
25 }
26
27 resource "aws_route_table_association" "az1-private-db-subnet-1-rta" {
28         subnet_id      = aws_subnet.private-db-subnet-1.id
29         route_table_id = aws_route_table.az1-private-rt.id
30 }
31
32
```

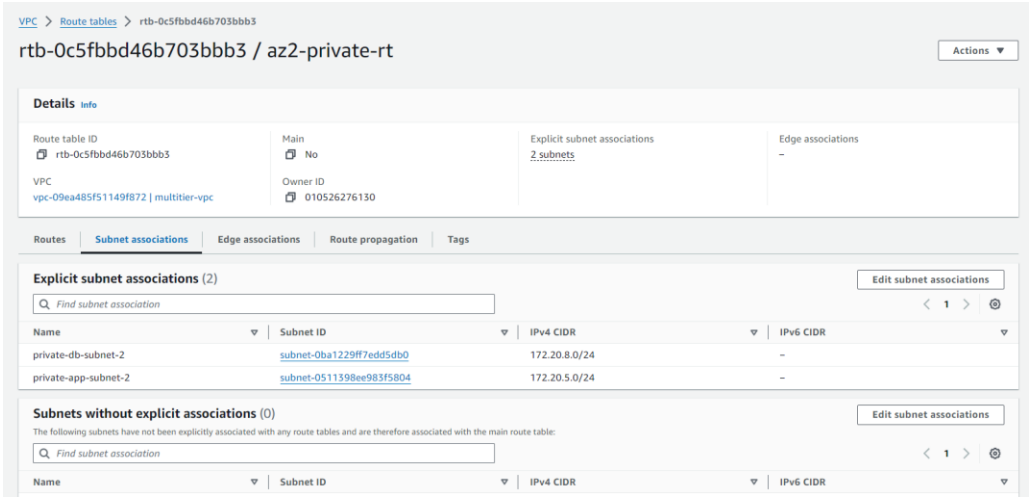


`

PRIVATE SUBNETS' ROUTE TABLES: *9_az2-private-rt*

multitier.tf > 9_az2-private-rt.tf > ...

```
1  ###################################
2  # AZ-2 PRIVATE ROUTE TABLE
3  ###################################
4
5  resource "aws_route_table" "az2-private-rt" {
6    vpc_id = aws_vpc.multitier-vpc.id
7
8    route {
9      cidr_block = "0.0.0.0/0"
10     gateway_id = aws_nat_gateway.az2-nat-gw.id
11   }
12
13   tags = {
14     Name = "az2-private-rt"
15   }
16 }
17
18 ############################################################
19 # ROUTE TABLE ASSOCIATION FOR AZ2 PRIVATE SUBNETS
20 ############################################################
21
22 resource "aws_route_table_association" "az2-private-app-subnet-2-rta" {
23         subnet_id      = aws_subnet.private-app-subnet-2.id
24         route_table_id = aws_route_table.az2-private-rt.id
25 }
26
27 resource "aws_route_table_association" "az2-private-db-subnet-2-rta" {
28         subnet_id      = aws_subnet.private-db-subnet-2.id
29         route_table_id = aws_route_table.az2-private-rt.id
30 }
31
```

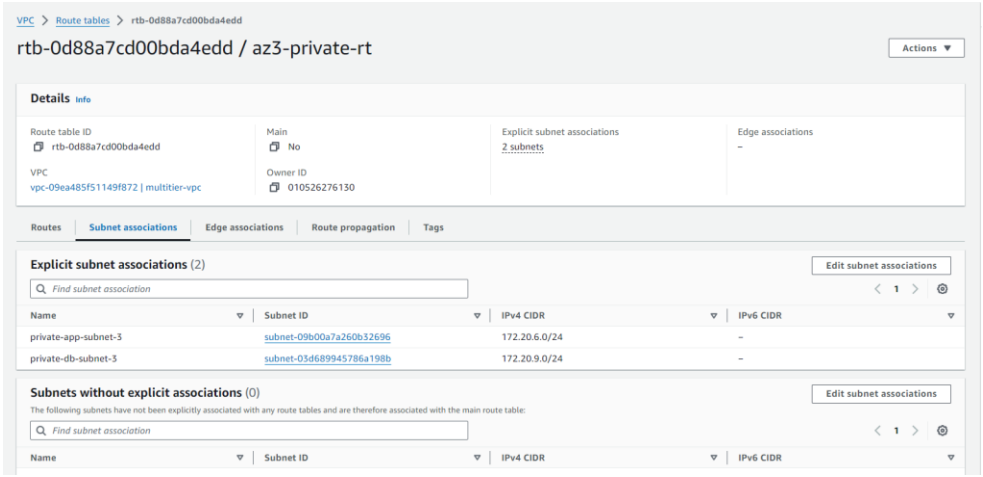


`

PRIVATE SUBNETS' ROUTE TABLES: *10_az3-private-rt*

multitier.tf > 10_az3-private-rt.tf > resource "aws_route_table_association" "az3-private-app-subnet-3-rta" > subnet_id

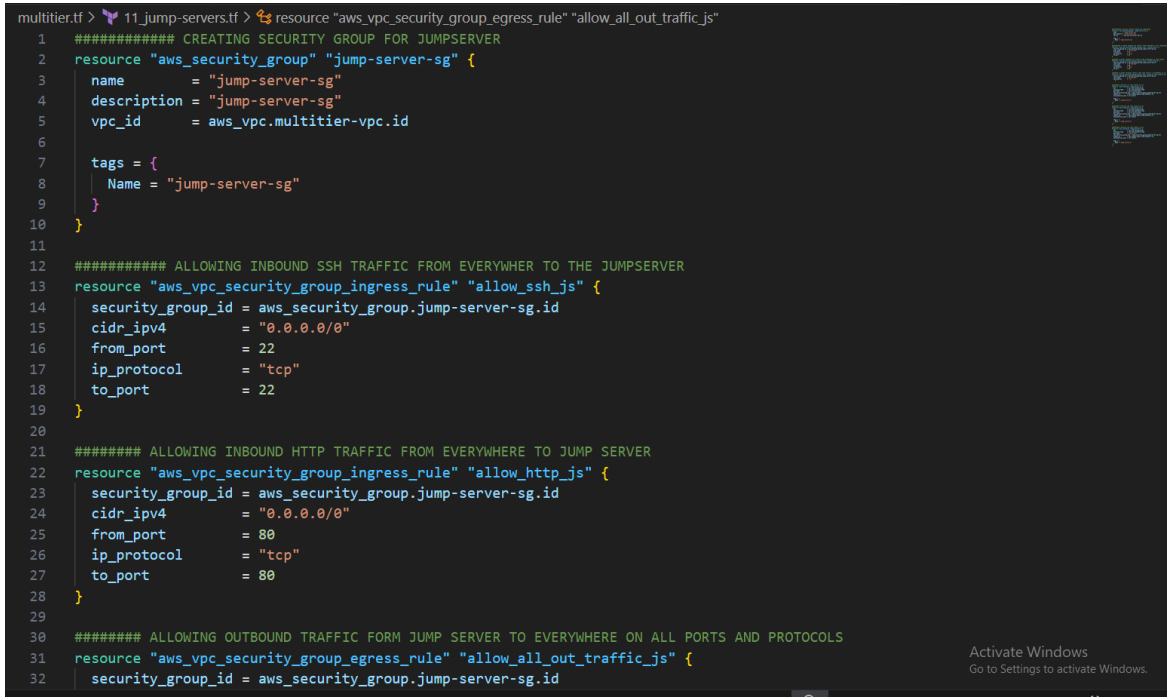
```
####################################
# AZ-3 PRIVATE ROUTE TABLE
####################################

resource "aws_route_table" "az3-private-rt" {
  vpc_id = aws_vpc.multitier-vpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_nat_gateway.az3-nat-gw.id
  }

  tags = {
    Name = "az3-private-rt"
  }
}

#############################################################
# ROUTE TABLE ASSOCIATION FOR AZ3 PRIVATE SUBNETS
#############################################################

resource "aws_route_table_association" "az3-private-app-subnet-3-rta" {
        subnet_id      = aws_subnet.private-app-subnet-3.id
        route_table_id = aws_route_table.az3-private-rt.id
}

resource "aws_route_table_association" "az3-private-db-subnet-3-rta" {
        subnet_id      = aws_subnet.private-db-subnet-3.id
        route_table_id = aws_route_table.az3-private-rt.id
}
```



`

9. CREATING JUMP SERVERS : *11_jump-servers.tf*

The jump server serves as an intermediary device which will be used to connect to other servers within the private subnets. By creating and configuring the jump server in a public subnet, a secure point of access is provided to manage the private infrastructure, reducing the need to expose sensitive components directly to the internet. To ensure secure and controlled access, a dedicated security group was created specifically for the jump servers, allowing inbound SSH traffic on port 22 and HTTP traffic on port 80 from any IP address. This setup ensures that any other administrator, can securely access the jump server while also enabling basic web traffic if necessary. The security group was configured to permit outbound traffic on all ports and protocols, allowing the jump server to communicate freely with other resources and the internet. The first jump servers were provisioned in each availability zone, configuring it with the appropriate security group, AMI, instance type, and key pair for secure SSH access. I then deployed additional jump servers in the remaining availability zones to ensure redundancy and high availability across the infrastructure.



`

```
}

######## ALLOWING OUTBOUND TRAFFIC FORM JUMP SERVER TO EVERYWHERE ON ALL PORTS AND PROTOCOLS
resource "aws_vpc_security_group_egress_rule" "allow_all_out_traffic_js" {
  security_group_id = aws_security_group.jump-server-sg.id
  cidr_ipv4         = "0.0.0.0/0"
  ip_protocol       = "-1"
    }


########## CREATING 1ST JUMP SERVER IN AZ1
resource "aws_instance" "jump-server-1" {
  ami               = var.ec2-instance-ami
  instance_type     = var.ec2-instance-type
  key_name          = var.ec2-key-name
  vpc_security_group_ids = [aws_security_group.jump-server-sg.id]
  subnet_id         = aws_subnet.public-web-subnet-1.id
  availability_zone = var.zone-1

  tags = {
    Name = "jump-server-1"
  }
}

########## CREATING 2ND JUMP SERVER IN AZ2
resource "aws_instance" "jump-server-2" {
  ami               = var.ec2-instance-ami
  instance_type     = var.ec2-instance-type
  key_name          = var.ec2-key-name
  vpc_security_group_ids = [aws_security_group.jump-server-sg.id]
  subnet_id         = aws_subnet.public-web-subnet-2.id
  availability_zone = var.zone-2

  tags = {
    Name = "jump-server-2"
  }
}

########## CREATING 3RD JUMP SERVER IN AZ2
resource "aws_instance" "jump-server-3" {
  ami               = var.ec2-instance-ami
  instance_type     = var.ec2-instance-type
  key_name          = "terra-key"
  vpc_security_group_ids = [aws_security_group.jump-server-sg.id]
  subnet_id         = aws_subnet.public-web-subnet-3.id
  availability_zone = var.zone-3

  tags = {
    Name = "jump-server-3"
  }
}
```

EC2 > Security Groups > sg-04a551b81453005d1 - jump-server-sg

sg-04a551b81453005d1 - jump-server-sg

Actions

Details

Security group name	Security group ID	Description	VPC ID
jump-server-sg	sg-04a551b81453005d1	jump-server-sg	vpc-09ea485f51149f872

Owner	Inbound rules count	Outbound rules count
010526276130	2 Permission entries	1 Permission entry

Inbound rules | Outbound rules | Tags

Inbound rules (2)

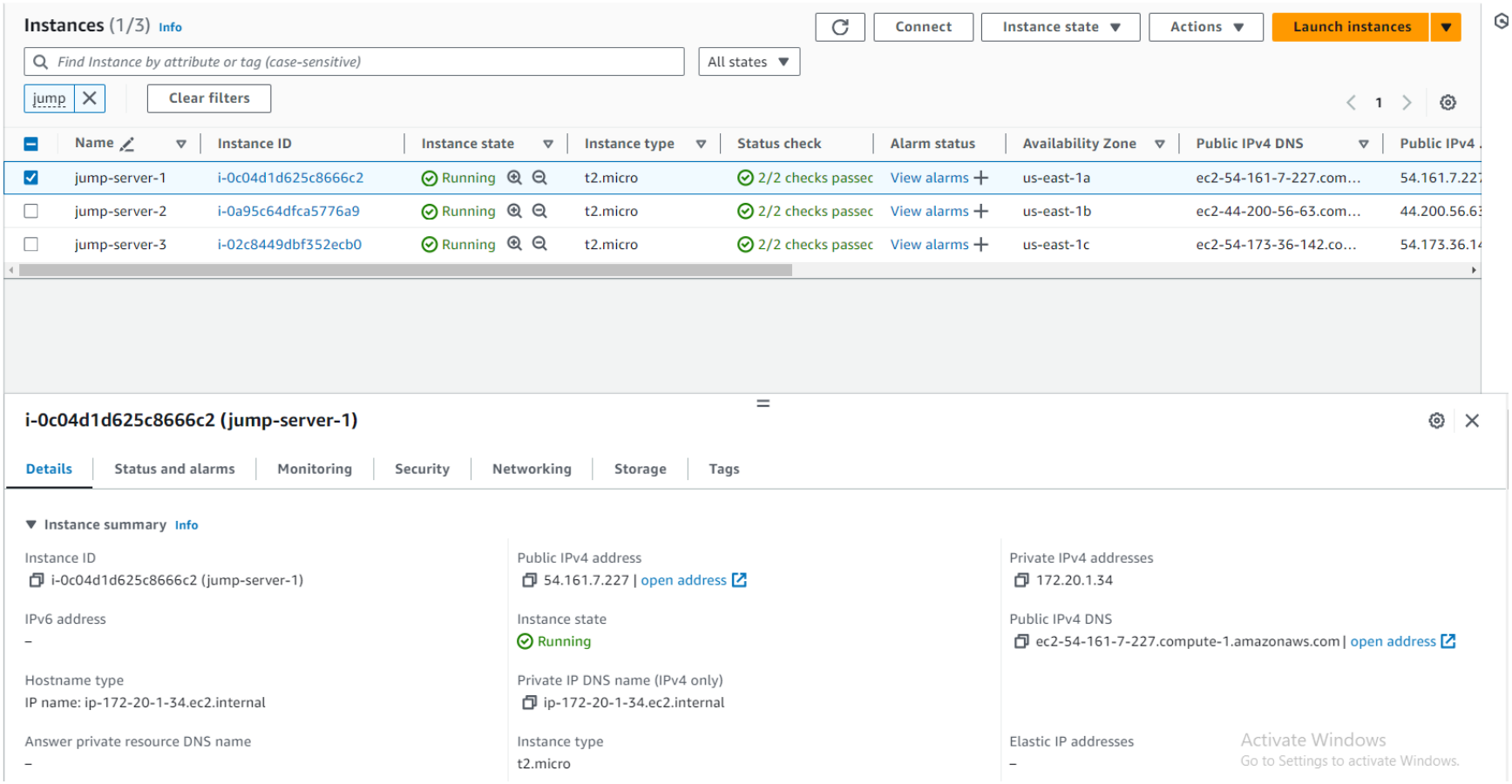
Manage tags | Edit inbound rules

Search

1

Name	Security group rule...	IP version	Type	Protocol	Port range	Source
-	sgr-01be1dfed6c9fec08	IPv4	HTTP	TCP	80	0.0.0.0/0
-	sgr-053a0575497898...	IPv4	SSH	TCP	22	0.0.0.0/0

`


Instances (1/3) Info
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
jump
Clear filters
1
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
Public IPv4
jump-server-1
i-0c04d1d625c8666c2
Running
t2.micro
2/2 checks passec
View alarms
us-east-1a
ec2-54-161-7-227.com...
54.161.7.227
jump-server-2
i-0a95c64dfca5776a9
Running
t2.micro
2/2 checks passec
View alarms
us-east-1b
ec2-44-200-56-63.com...
44.200.56.63
jump-server-3
i-02c8449dbf352ecb0
Running
t2.micro
2/2 checks passec
View alarms
us-east-1c
ec2-54-173-36-142.co...
54.173.36.14
i-0c04d1d625c8666c2 (jump-server-1)
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance summary Info
Instance ID
i-0c04d1d625c8666c2 (jump-server-1)
IPv6 address
–
Hostname type
IP name: ip-172-20-1-34.ec2.internal
Answer private resource DNS name
–
Public IPv4 address
54.161.7.227 | open address
Instance state
Running
Private IP DNS name (IPv4 only)
ip-172-20-1-34.ec2.internal
Instance type
t2.micro
Private IPv4 addresses
172.20.1.34
Public IPv4 DNS
ec2-54-161-7-227.compute-1.amazonaws.com | open address
Elastic IP addresses
–
Activate Windows
Go to Settings to activate Windows.

`

10. <u>CREATING PHP SERVER:</u> *<u>12_php-server.tf</u>*

The PHP server hosts the application logic of the e-commerce platform and is crucial for serving dynamic content to users. By deploying the PHP server in a private subnet, it is shielded from direct internet exposure, ensuring enhanced security while still being able to handle incoming requests routed through the Application Load Balancer (ALB). To ensure secure and controlled access, a dedicated security group was created specifically for the PHP servers, allowing inbound HTTP traffic from the ALB security group and any IP address, as well as SSH traffic from the jump server. This setup ensures that administrators can securely manage the PHP server while also enabling it to communicate with other essential components. The security group was configured to permit outbound traffic on all ports and protocols, allowing the PHP server to freely interact with other resources and the internet.

The PHP servers were provisioned in each availability zone, configured with the appropriate security group, AMI, instance type, and private IP address to maintain security within the private subnet. A key pair was created and associated with the servers for secure SSH access. A file provisioner was used to transfer the LAMP stack script to the server, followed by a remote-exec provisioner that executed the script to install and configure the necessary software components, including the integration of the database server's endpoint into the application configuration. This setup was replicated across the remaining availability zones to ensure redundancy and high availability throughout the infrastructure.

`

```
multitier.tf > 12_php-server.tf > resource "aws_instance" "php-server-1" > provisioner "file" > destination
1  ########## CREATING KEY PAIR FOR INSTANCES
2  resource "aws_key_pair" "terra-key" {
3    key_name   = var.ec2-key-name
4    public_key = file("terra-key.pub")
5  }
6
7  ############ CREATING SECURITY GROUP FOR PHP-SERVERS
8  resource "aws_security_group" "php-server-sg" {
9    name        = "php-server-sg"
10   description = "php-server-sg"
11   vpc_id      = aws_vpc.multitier-vpc.id
12
13
14 ############ ALLOWING INBOUND HTTP TRAFFIC FORM ALB SECURITY GROUP
15   ingress {
16     from_port = 80
17     to_port = 80
18     protocol = "tcp"
19     security_groups = [ aws_security_group.alb-sg.id ]
20   }
21
22 ############ ALLOWING INBOUND HTTP TRAFFIC FORM EVERYWHERE
23     ingress {
24     cidr_blocks = [ "0.0.0.0/0"]
25     from_port = 80
26     to_port = 80
27     protocol = "tcp"
28
29   }
30
31 ############ ALLOWING INBOUND SSH TRAFFIC FORM JUMP SERVER
32   ingress {
33     security_groups = [ aws_security_group.jump-server-sg.id ]
34     from_port = 22
35     to_port = 22
36     protocol = "tcp"
37   }
38
39 ############ ALLOWING INBOUND HTTP TRAFFIC FORM ALB
40   egress {
41     cidr_blocks = [ "0.0.0.0/0"]
42     from_port = 0
43     to_port = 0
44     protocol = "-1"
45   }
46
47
48   tags = {
49     Name = "pho-server-se"
                                                    Activate Windows
                                                    Go to Settings to activate
                                                    Ln 72, Col 34   Spaces: 2   UTF-8
```

EC2 > Security Groups > sg-03861aa786fb6da33 - php-server-sg

sg-03861aa786fb6da33 - php-server-sg

Actions

Details

Security group name	Security group ID	Description	VPC ID
php-server-sg	sg-03861aa786fb6da33	php-server-sg	vpc-09ea485f51149f872

Owner	Inbound rules count	Outbound rules count
010526276130	3 Permission entries	1 Permission entry

Inbound rules | Outbound rules | Tags

Inbound rules (3)

Manage tags | Edit inbound rules

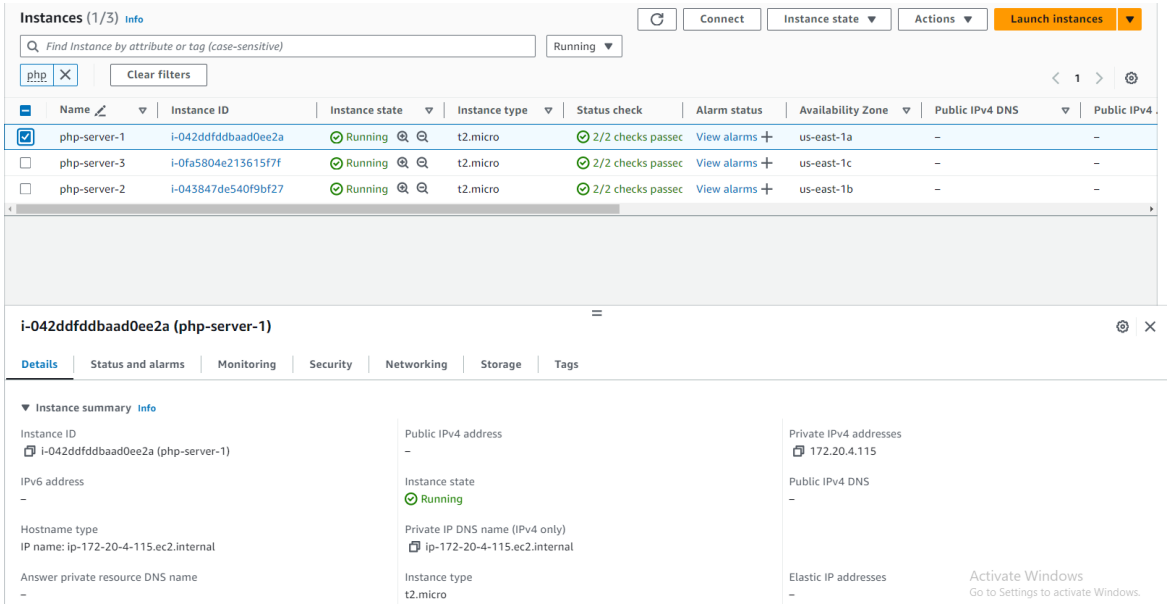
Search

1

Name	Security group rule...	IP version	Type	Protocol	Port range	Source
–	sgr-0956faaae3533182d	–	HTTP	TCP	80	sg-0f3ac5ca0c972
–	sgr-05d2bc8e85260cf21	IPv4	HTTP	TCP	80	0.0.0.0/0
–	sgr-098ff3304daf2699d	–	SSH	TCP	22	sg-04a551b81453

`

```
multitier.tf > 12_php-server.tf > resource "aws_instance" "php-server-1" > provisioner "file" > destination
resource "aws_instance" "php-server-1" {
  key_name               = var.ec2-key-name
  vpc_security_group_ids = [aws_security_group.php-server-sg.id]
  subnet_id              = aws_subnet.private-app-subnet-1.id
  availability_zone = var.zone-1
  associate_public_ip_address = false

  tags = {
    Name = "php-server-1"
  }

############ CREATING FILE PROVISIONER TO TRANSFER LAMPSTACK FILE TO PHP SERVER
  provisioner "file" {
    source      = "lampstack.sh"
    destination = "/tmp/lampstack.sh"
    on_failure  = continue
    connection {
      type        = "ssh"
      user        = "ec2-user"
      host        = self.private_ip
      private_key = file("terra-key")
      bastion_host = aws_instance.jump-server-1.public_ip
      bastion_user = "ec2-user"
      bastion_private_key = file("terra-key")
    }
  }


############ CREATING A REMOTE-EXEC PROVISIONER TO EXECUTE LAMPSTACK SCRIPT AND INSERT ENDPOINT OF DB SERVER
  provisioner "remote-exec" {
    inline = [
      "chmod u+x /tmp/lampstack.sh",
      "sudo /tmp/lampstack.sh",
      "db_endpoint=${data.aws_db_instance.db_instance.endpoint}",
      "sudo sed -i 's/localhost/'\"$db_endpoint\"'/g' /var/www/html/config.sample.inc.php",
      "sudo mv /var/www/html/config.sample.inc.php /var/www/html/config.inc.php "
    ]

    connection {
      type        = "ssh"
      user        = "ec2-user"
      host        = self.private_ip
      private_key = file("terra-key")
      bastion_host = aws_instance.jump-server-1.public_ip
      bastion_user = "ec2-user"
      bastion_private_key = file("terra-key")
    }
  }
```



`

11. Creating ALB : *13_alb.tf*

The Application Load Balancer (ALB) will be used for distributing incoming web traffic across multiple targets, ensuring that the application remains highly available and can handle varying levels of traffic efficiently. The ALB is configured to distribute traffic to the PHP servers that host the application.

To secure the ALB, a dedicated security group was created, which allows inbound HTTP traffic on port 80 from any IP address. This setup ensures that the ALB can receive traffic from the internet and distribute it to the appropriate backend servers. The security group also permits outbound traffic on all ports and protocols, allowing the ALB to communicate freely with the registered targets, such as the PHP servers.

A target group was created to manage the PHP servers that the ALB will forward traffic to. Each PHP server was registered as a target in this group, enabling the ALB to distribute incoming requests evenly across the available servers, enhancing load distribution and fault tolerance.

The ALB itself was provisioned across multiple public subnets, ensuring high availability by distributing traffic across different availability zones. An HTTP listener was configured for the ALB, which listens for incoming traffic on port 80 and forwards it to the registered PHP servers via the target group. This configuration ensures that incoming requests are efficiently managed and routed to the appropriate resources within the infrastructure.

`

multitier.tf > 13_alb.tf > resource "aws_lb_listener" "http" > default_action > target_group_arn

```
######### CREATING ALB SECURITY GROUP
resource "aws_security_group" "alb-sg" {
  name        = "alb-sg"
  description = "security_group_for_alb"
  vpc_id      = aws_vpc.multitier-vpc.id

  tags = {
    Name = "alb-sg"
  }


}

########## ALLOWING INBOUND HTTP TRAFFIC FROM THE INTERNET TO THE ALB
resource "aws_vpc_security_group_ingress_rule" "allow_http_alb" {
  security_group_id = aws_security_group.alb-sg.id
  cidr_ipv4         = "0.0.0.0/0"
  from_port         = 80
  ip_protocol       = "tcp"
  to_port           = 80
}

########## ALLOWING OUTBOUND TRAFFIC FROM THE ALB TO EVERYWHERE
resource "aws_vpc_security_group_egress_rule" "allow_all_out_traffic_alb" {
  security_group_id   = aws_security_group.alb-sg.id
  cidr_ipv4           = "0.0.0.0/0"
  ip_protocol         = "-1"
}

########### CREATING TARGET GROUP FOR ALB
resource "aws_lb_target_group" "alb-tg" {
  name        = "alb-tg"
  target_type = "instance"
  port        = 80
  protocol    = "HTTP"
  vpc_id      = aws_vpc.multitier-vpc.id
}


#############################################
# REGISTERING TARGETS
#############################################

######## REGISTERING PHP SERVER 1
resource "aws_lb_target_group_attachment" "register-php-server-1" {
  target_group_arn = aws_lb_target_group.alb-tg.arn
  target_id        = aws_instance.php-server-1.id
  port             = 80
}
```

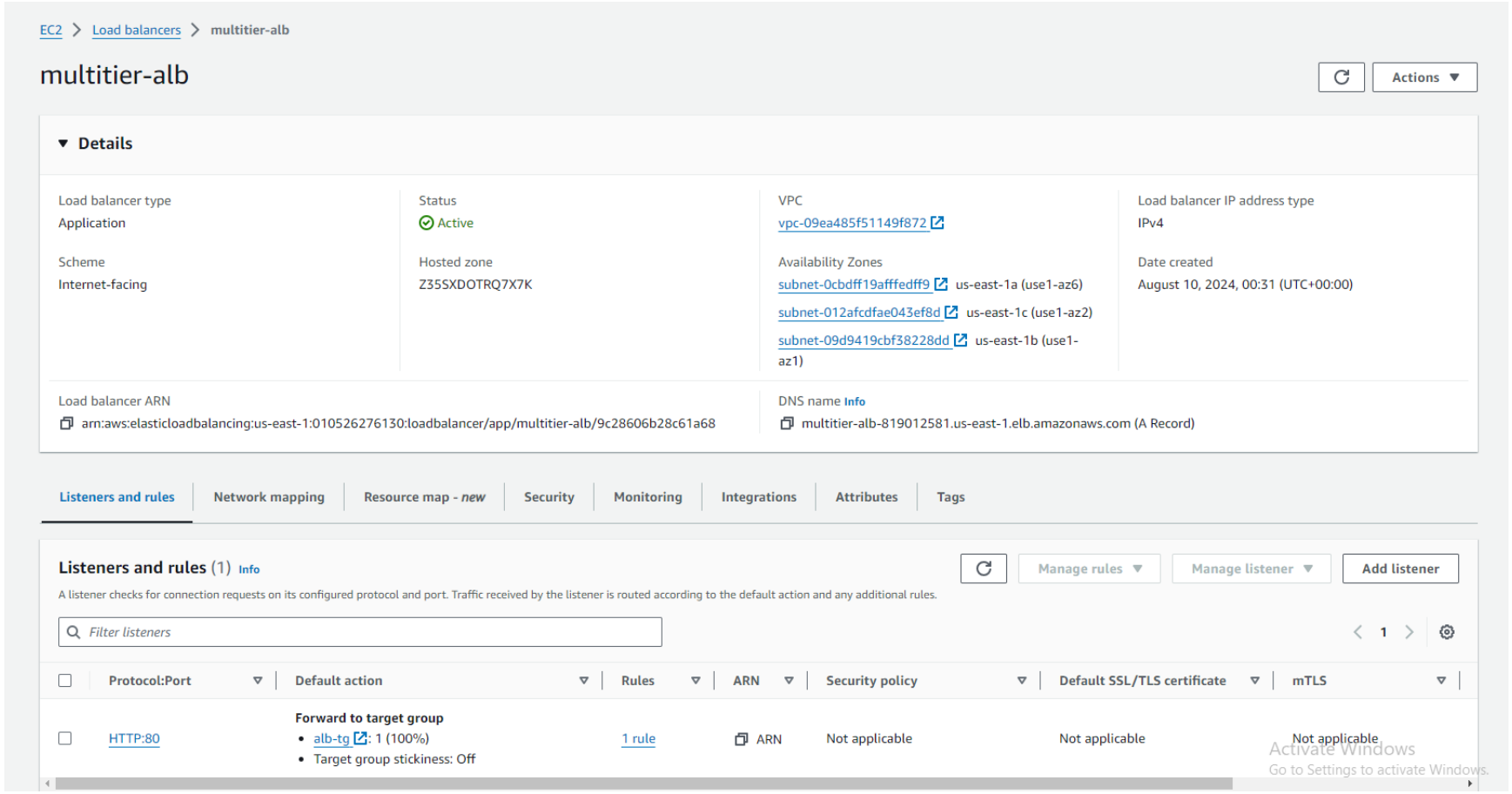
Activat
Go to Se

Ln 87, Col 54

`

```
43
44  ######## REGISTERING PHP SERVER 1
45  resource "aws_lb_target_group_attachment" "register-php-server-1" {
46    target_group_arn = aws_lb_target_group.alb-tg.arn
47    target_id        = aws_instance.php-server-1.id
48    port             = 80
49  }
50
51  ######## REGISTERING PHP SERVER 2
52  resource "aws_lb_target_group_attachment" "register-php-server-2" {
53    target_group_arn = aws_lb_target_group.alb-tg.arn
54    target_id        = aws_instance.php-server-2.id
55    port             = 80
56  }
57
58  ######## REGISTERING PHP SERVER 3
59  resource "aws_lb_target_group_attachment" "register-php-server-3" {
60    target_group_arn = aws_lb_target_group.alb-tg.arn
61    target_id        = aws_instance.php-server-3.id
62    port             = 80
63  }
64
65  ############# CREATING ALB
66  resource "aws_lb" "multitier-alb" {
67    name               = "multitier-alb"
68    internal           = false
69    load_balancer_type = "application"
70    security_groups    = [aws_security_group.alb-sg.id]
71    subnets            = [
72      aws_subnet.public-web-subnet-1.id,
73      aws_subnet.public-web-subnet-2.id,
74      aws_subnet.public-web-subnet-3.id
75    ]
76    enable_deletion_protection = false
77  }
78
79  ############## CRATING ALB LISTENER FOR HTTP
80  resource "aws_lb_listener" "http" {
81    load_balancer_arn = aws_lb.multitier-alb.arn
82    port              = 80
83    protocol          = "HTTP"
84
85    default_action {
86      type             = "forward"
87      target_group_arn = aws_lb_target_group.alb-tg.arn
88    }
89  }
90
```

`


EC2 > Load balancers > multitier-alb
multitier-alb
Actions
Details
Load balancer type
Application
Status
Active
VPC
vpc-09ea485f51149f872
Load balancer IP address type
IPv4
Scheme
Internet-facing
Hosted zone
Z35SXDOTRQ7X7K
Availability Zones
subnet-0cbdff19afffedff9 us-east-1a (use1-az6)
subnet-012afcdfae043ef8d us-east-1c (use1-az2)
subnet-09d9419cbf38228dd us-east-1b (use1-az1)
Date created
August 10, 2024, 00:31 (UTC+00:00)
Load balancer ARN
arn:aws:elasticloadbalancing:us-east-1:010526276130:loadbalancer/app/multitier-alb/9c28606b28c61a68
DNS name Info
multitier-alb-819012581.us-east-1.elb.amazonaws.com (A Record)
Listeners and rules
Network mapping
Resource map - new
Security
Monitoring
Integrations
Attributes
Tags
Listeners and rules (1) Info
Manage rules
Manage listener
Add listener
A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.
Filter listeners
1
Protocol:Port
Default action
Rules
ARN
Security policy
Default SSL/TLS certificate
mTLS
HTTP:80
Forward to target group
alb-tg: 1 (100%)
Target group stickiness: Off
1 rule
ARN
Not applicable
Not applicable
Not applicable
Activate Windows
Go to Settings to activate Windows.

`

12. CREATING DB INSTANCE: :*14_db-server.tf*

The database instance is a critical component for storing all data required by the e-commerce platform. An Amazon RDS (Relational Database Service) instance was used, configured with MySQL as the database engine, to ensure the database is secure, highly available, and capable of handling the application's demands.

The database instance was provisioned with 20 GB of allocated storage and configured to use the MySQL 8.0 engine. It was set up with a unique identifier (db-server-1) and configured with secure credentials. The instance was made non-publicly accessible, ensuring that it can only be accessed within the VPC, thereby enhancing its security.

To ensure high availability, the database was deployed in a Multi-AZ (Availability Zone) configuration, which allows the database to automatically failover to a standby instance in another availability zone if an outage occurs. This setup provides resilience and minimizes downtime.

A database subnet group was created, encompassing subnets from three different availability zones. This ensures that the database instance is isolated within the private network, further enhancing its security and availability by distributing it across multiple zones.

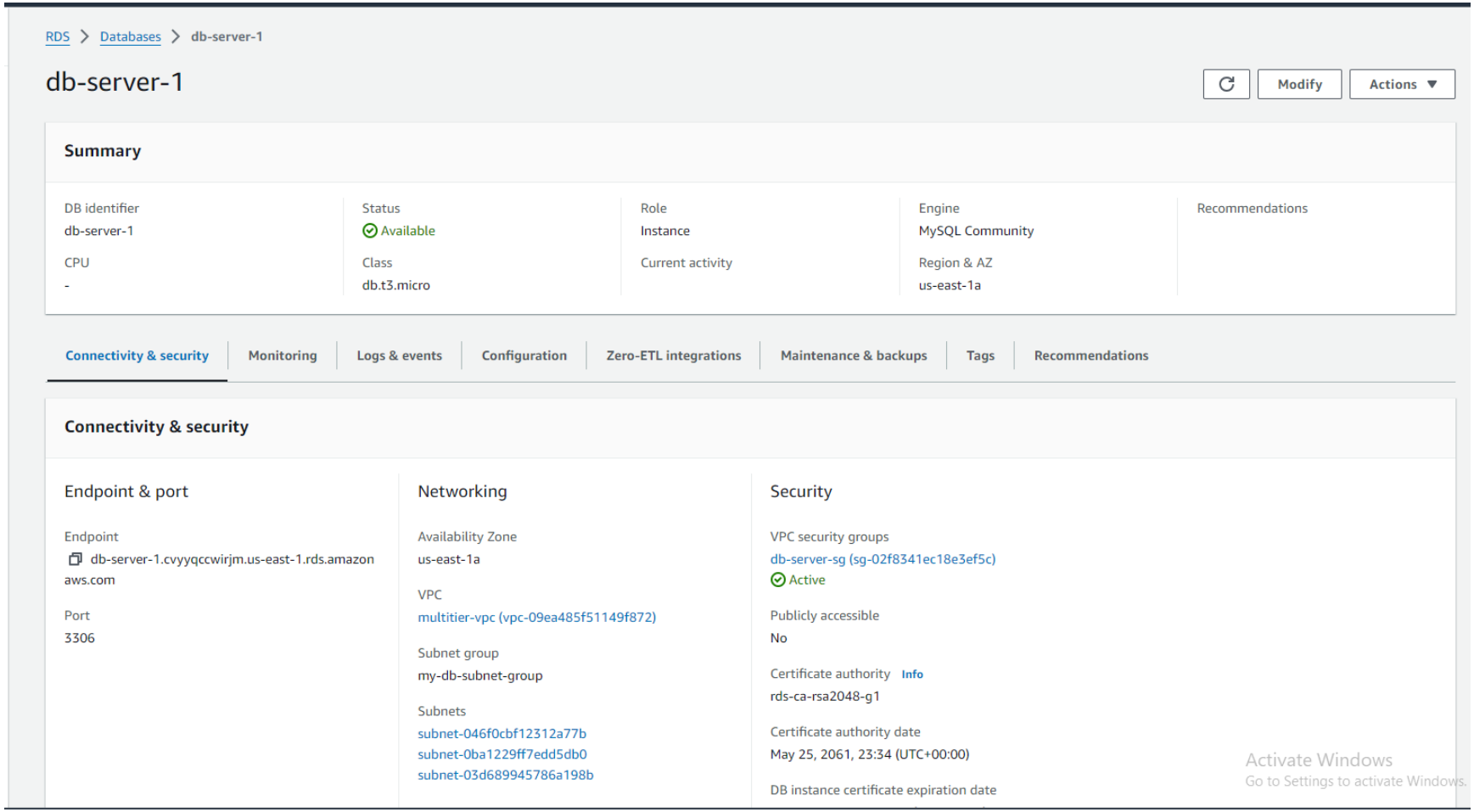
The security group for the database server was configured to allow inbound and outbound traffic on port 3306, which is used for MySQL communication. This traffic is restricted to the security group associated with the PHP servers, ensuring that only the application servers can communicate with the database. This configuration secures the database while maintaining efficient communication with the application servers.

`

multitier.tf > 14_db-ins.tf > resource "aws_db_instance" "db-server-1" > publicly_accessible

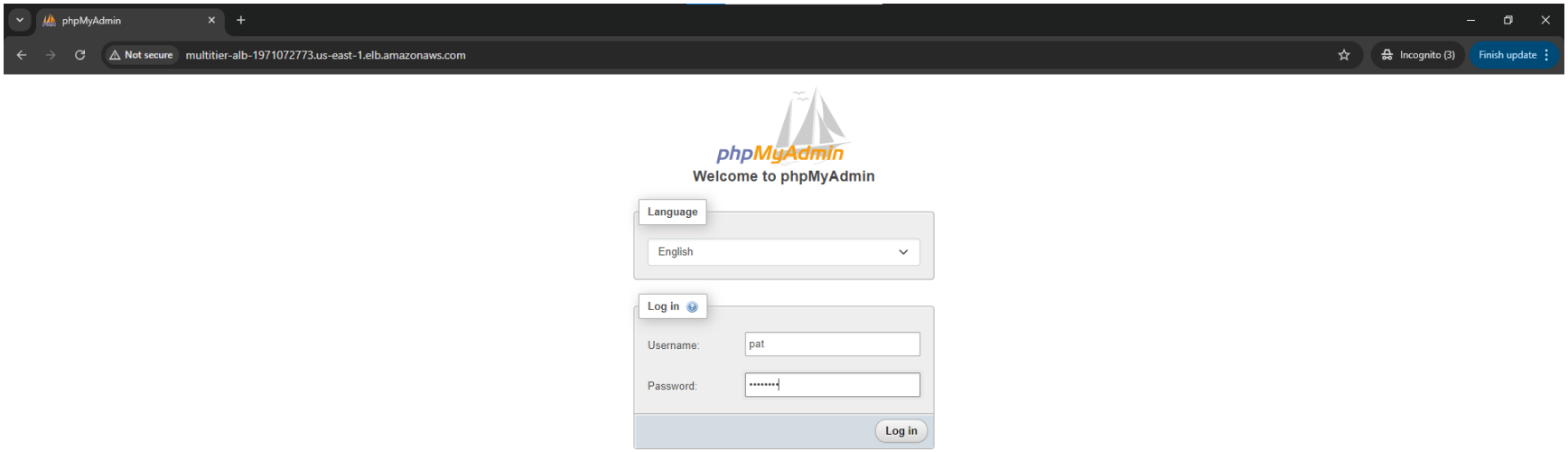
```
######### CREATING DB INSTANCE
resource "aws_db_instance" "db-server-1" {
  allocated_storage       = 20
  db_name                 = "mydb"
  identifier              = "db-server-1"
  engine                  = var.db-engine
  engine_version          = var.engine-version
  instance_class          = var.db-instance-class
  username                = var.db-username
  password                = var.db-username-pswd
  parameter_group_name    = "default.mysql8.0"
  skip_final_snapshot     = true
  db_subnet_group_name    = aws_db_subnet_group.my-db-subnet-group.id
  vpc_security_group_ids  = [ aws_security_group.db-server-sg.id ]
  multi_az                = true
  publicly_accessible     = false

  tags = {
    Name = "db-server-1"
  }

}


#### CREATING DB SUBNET GROUP
resource "aws_db_subnet_group" "my-db-subnet-group" {
  name = "my-db-subnet-group"
  subnet_ids = [
    aws_subnet.private-db-subnet-1.id,
    aws_subnet.private-db-subnet-2.id,
    aws_subnet.private-db-subnet-3.id
  ]

  tags = {
    Name = "multitier-DB-Subnet-Group"
  }
}

######## CREATING DB SERVER SECURITY GROUP
resource "aws_security_group" "db-server-sg" {
  name        = "db-server-sg"
  description = "db-server-sg"
  vpc_id      = aws_vpc.multitier-vpc.id

  ingress {
    from_port = 3306
    to_port = 3306
    protocol = "tcp"
    security_groups = [ aws_security_group.php-server-sg.id ]
```

`

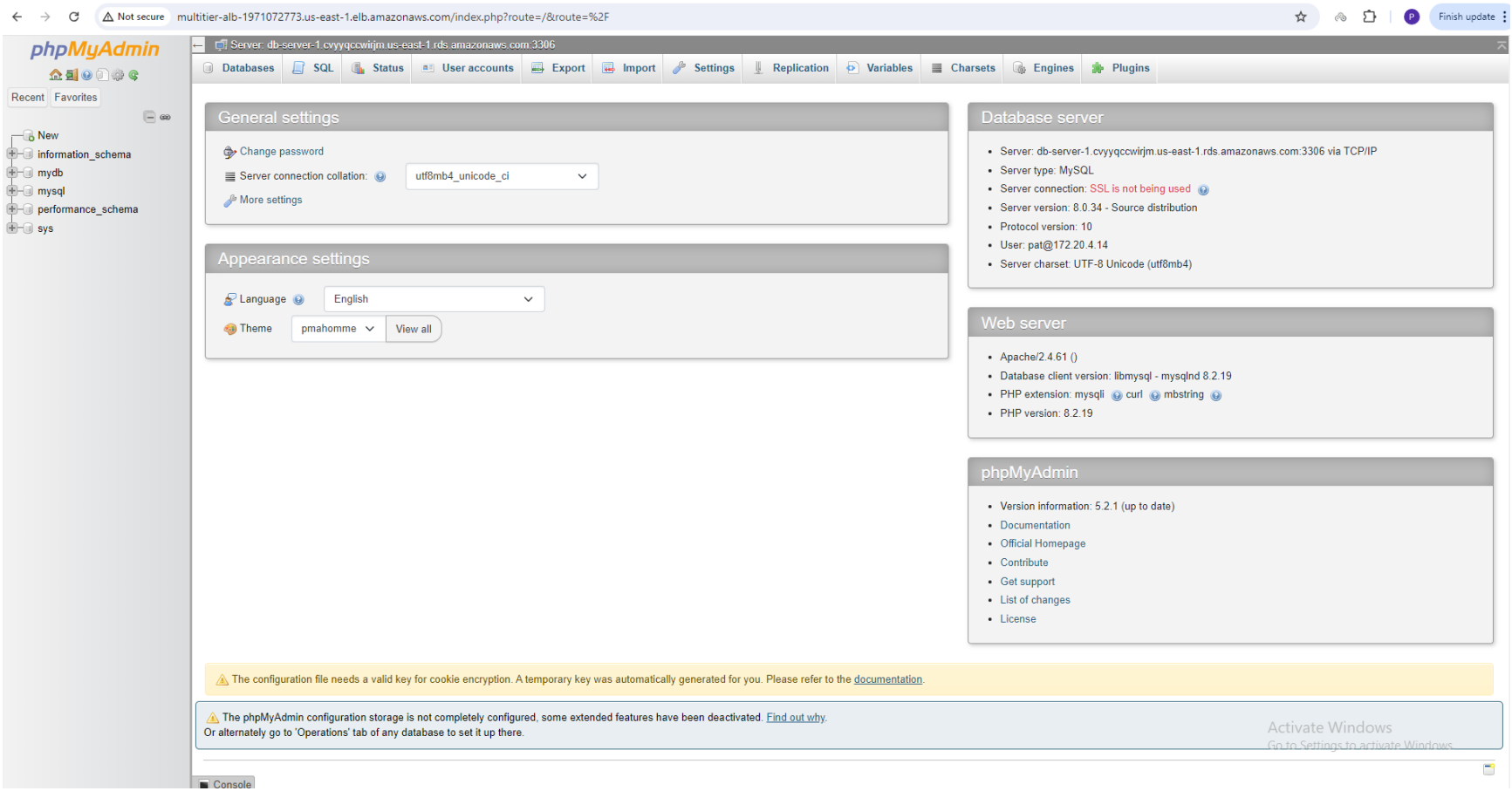

RDS > Databases > db-server-1
db-server-1
Modify
Actions
Summary
DB identifier
db-server-1
Status
Available
Role
Instance
Engine
MySQL Community
Recommendations
CPU
-
Class
db.t3.micro
Current activity
Region & AZ
us-east-1a
Connectivity & security
Monitoring
Logs & events
Configuration
Zero-ETL integrations
Maintenance & backups
Tags
Recommendations
Connectivity & security
Endpoint & port
Endpoint
db-server-1.cvyyqccwirjm.us-east-1.rds.amazonaws.com
Port
3306
Networking
Availability Zone
us-east-1a
VPC
multitier-vpc (vpc-09ea485f51149f872)
Subnet group
my-db-subnet-group
Subnets
subnet-046f0cbf12312a77b
subnet-0ba1229ff7edd5db0
subnet-03d689945786a198b
Security
VPC security groups
db-server-sg (sg-02f8341ec18e3ef5c)
Active
Publicly accessible
No
Certificate authority Info
rds-ca-rsa2048-g1
Certificate authority date
May 25, 2061, 23:34 (UTC+00:00)
DB instance certificate expiration date
Activate Windows
Go to Settings to activate Windows.

`

13. To confirm the application is running the DNS name of the ALB was entered in the web server and logged in with the credentials of the database.



`


Not secure
multitier-alb-1971072773.us-east-1.elb.amazonaws.com/index.php?route=/&route=%2F
Finish update
phpMyAdmin
Recent
Favorites
New
information_schema
mydb
mysql
performance_schema
sys
Server: db-server-1.cvyyqccwirjm.us-east-1.rds.amazonaws.com:3306
Databases
SQL
Status
User accounts
Export
Import
Settings
Replication
Variables
Charsets
Engines
Plugins
General settings
Change password
Server connection collation:
utf8mb4_unicode_ci
More settings
Appearance settings
Language
English
Theme
pmahomme
View all
Database server
Server: db-server-1.cvyyqccwirjm.us-east-1.rds.amazonaws.com:3306 via TCP/IP
Server type: MySQL
Server connection: SSL is not being used
Server version: 8.0.34 - Source distribution
Protocol version: 10
User: pat@172.20.4.14
Server charset: UTF-8 Unicode (utf8mb4)
Web server
Apache/2.4.61 ()
Database client version: libmysql - mysqlnd 8.2.19
PHP extension: mysqli curl mbstring
PHP version: 8.2.19
phpMyAdmin
Version information: 5.2.1 (up to date)
Documentation
Official Homepage
Contribute
Get support
List of changes
License
The configuration file needs a valid key for cookie encryption. A temporary key was automatically generated for you. Please refer to the documentation.
The phpMyAdmin configuration storage is not completely configured, some extended features have been deactivated. Find out why.
Or alternately go to 'Operations' tab of any database to set it up there.
Activate Windows
Go to Settings to activate Windows.
Console

`

CHALLENGES ENCOUNTERED

1. Automating Access to the PHP Server through the Jump Server

While developing the Terraform script, I faced a challenge with automating the login process to the PHP server via the jump server. The Terraform documentation didn't readily provide a solution, so I had to research on my own. I eventually found that the connection block could be enhanced with specific attributes, such as ***"bastion_host", "bastion user",*** and ***"bastion_private_key"***. By configuring these attributes, I was able to securely connect to the PHP server through the jump server using SSH, which was crucial for maintaining the security of the infrastructure.

2. Automating the Insertion of the Database Endpoint into the PHP Configuration

For the application to function correctly, it was necessary to insert the database endpoint into the PHP application's configuration file located at ***"/var/www/html/config.inc.php"***. Automating this process within the Terraform script was challenging. I addressed this by capturing the database endpoint as a variable within Terraform and then used basic sed commands in an inline script to automatically replace the placeholder in the configuration file with the actual endpoint. This ensured that the application was correctly configured during deployment.

3. Correcting the Apache Directory Path after LAMP Stack Installation

After executing the LAMP stack installation script, I discovered that the PHP application files were placed inside a phpMyAdmin folder within the ***"/var/www/html/"*** directory. However, Apache was configured to serve files directly from the ***"/var/www/html/"*** directory, causing issues when testing the application via the load balancer. To address this, I added an inline script to the ***"remote exec "***provisioner in Terraform, which automatically moved the files from the phpMyAdmin folder back to the root of the ***"/var/www/html/"*** directory. This adjustment allowed the application to become accessible through the web interface.

`