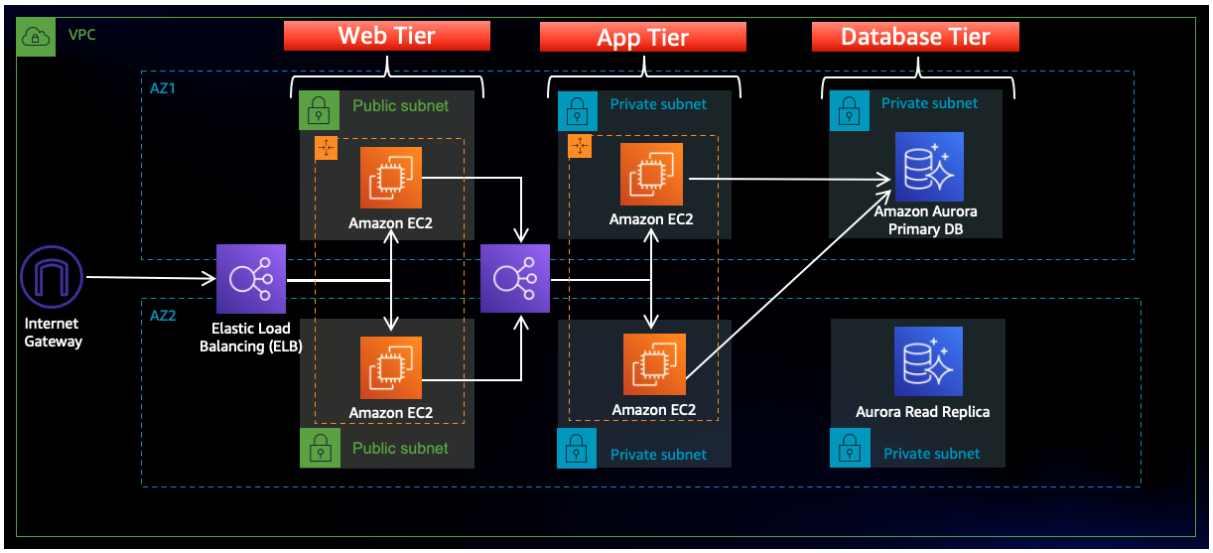
Building Scalable, Secure, and High-Performance Web Applications with AWS 3-Tier Architecture

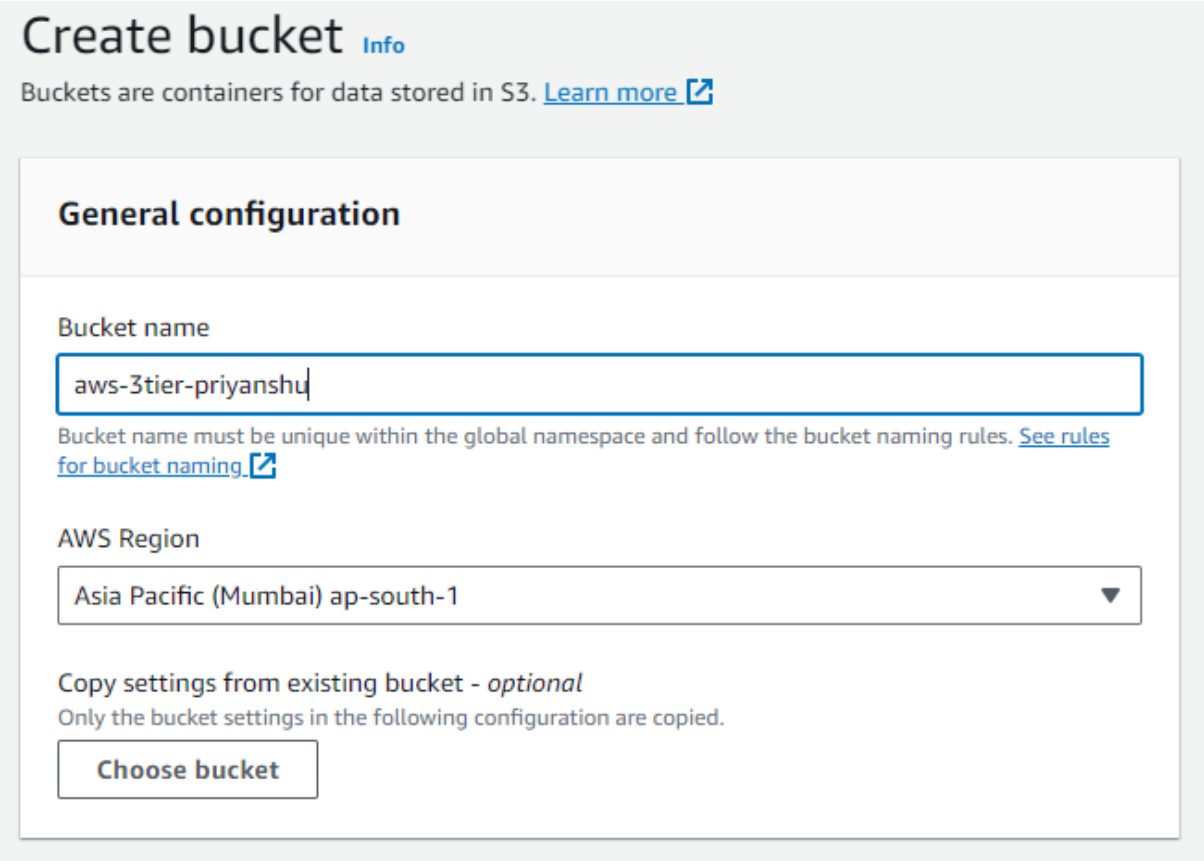
Architecture Overview



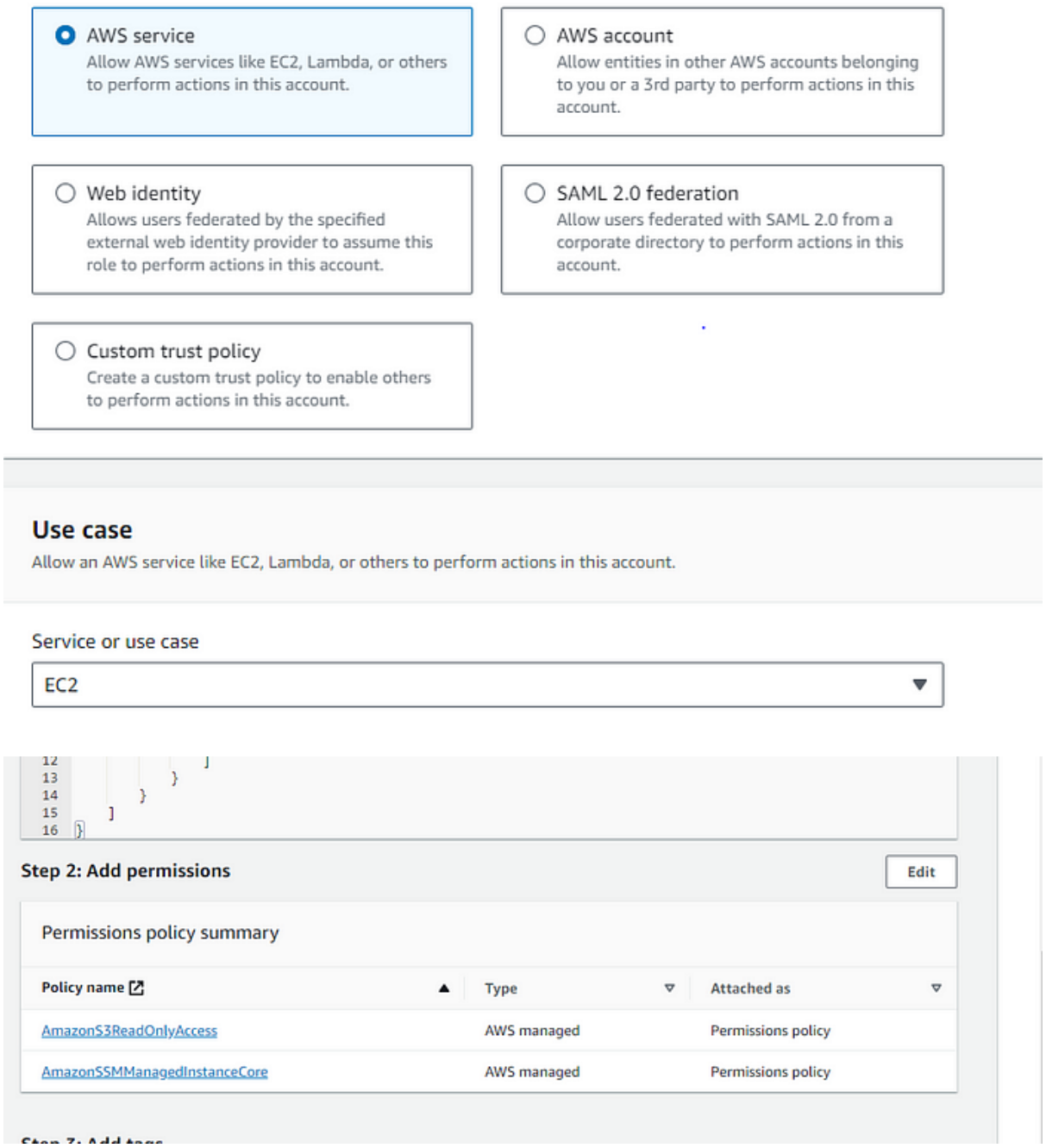
In today's digital age, building a scalable and secure web application is crucial for business success. The AWS 3-Tier Web Application Architecture stands out as a best practice for deploying cloud-based applications with efficiency and reliability. By leveraging Amazon Web Services (AWS) to create a three-tier architecture, businesses can achieve enhanced scalability, security, and performance. This architecture divides the application into three layers: the Web Tier, Application Tier, and Database Tier — each designed to handle specific tasks, ensuring seamless operation and optimal user experience. In this blog post, we'll delve into how the AWS 3-Tier architecture can be leveraged to build resilient and high-performing web applications. You can are a Automation Geek, checkout my ***2 tier architecture*** *in AWS using Terraform.*

Practical Implementation

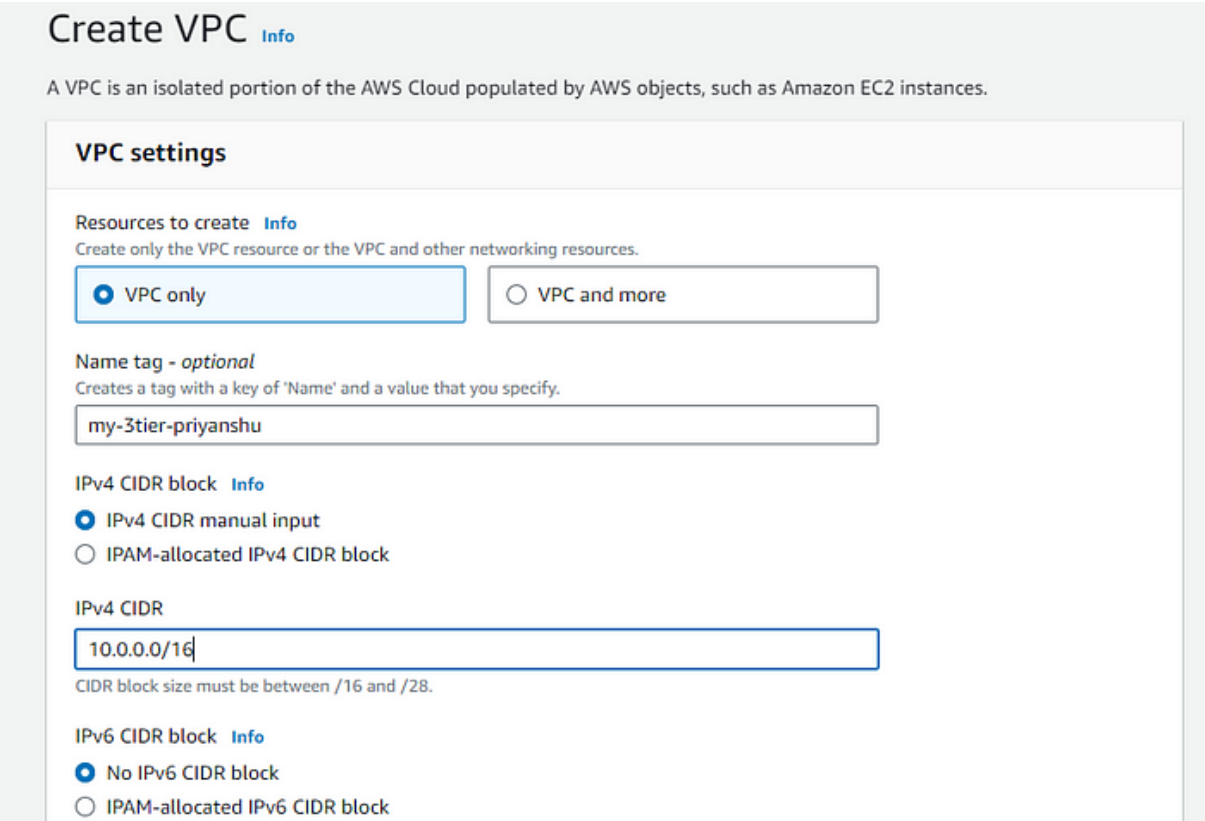
Step 1: Start by creating an S3 bucket which will hold the code for you tier.



STEP 2: Create an instance role for ec2 service, allow access to S3 but only restricted read access.



STEP 3: Create an Separate VPC for this architecture, Separate VPC provides network Isolation.



STEP 4: Create Subnets in different AZ, to create a multi AZ setup.We Will be having two Public and Two private subnets.

Subnet 1 of 1

Subnet name

Create a tag with a key of 'Name' and a value that you specify.

public-web-subnet-AZ1

The name can be up to 256 characters long.

Availability Zone Info

Choose the zone in which your subnet will reside, or let Amazon choose one for you.

Asia Pacific (Mumbai) / ap-south-1a

IPv4 VPC CIDR block Info

Choose the IPv4 VPC CIDR block to create a subnet in.

10.0.0.0/16

IPv4 subnet CIDR block

10.0.0.0/24 256 IPs

▼ Tags - *optional*

Key Value - *optional*

Subnet 2 of 2

Subnet name

Create a tag with a key of 'Name' and a value that you specify.

public-web-subnet-az2

The name can be up to 256 characters long.

Availability Zone Info

Choose the zone in which your subnet will reside, or let Amazon choose one for you.

Asia Pacific (Mumbai) / ap-south-1b

IPv4 VPC CIDR block Info

Choose the IPv4 VPC CIDR block to create a subnet in.

10.0.0.0/16

IPv4 subnet CIDR block

10.0.1.0/24 256 IPs

▼ Tags - *optional*

Subnet 3 of 3

Subnet name

Create a tag with a key of 'Name' and a value that you specify.

private-subnet-AZ1

The name can be up to 256 characters long.

Availability Zone **Info**

Choose the zone in which your subnet will reside, or let Amazon choose one for you.

Asia Pacific (Mumbai) / ap-south-1a

IPv4 VPC CIDR block **Info**

Choose the IPv4 VPC CIDR block to create a subnet in.

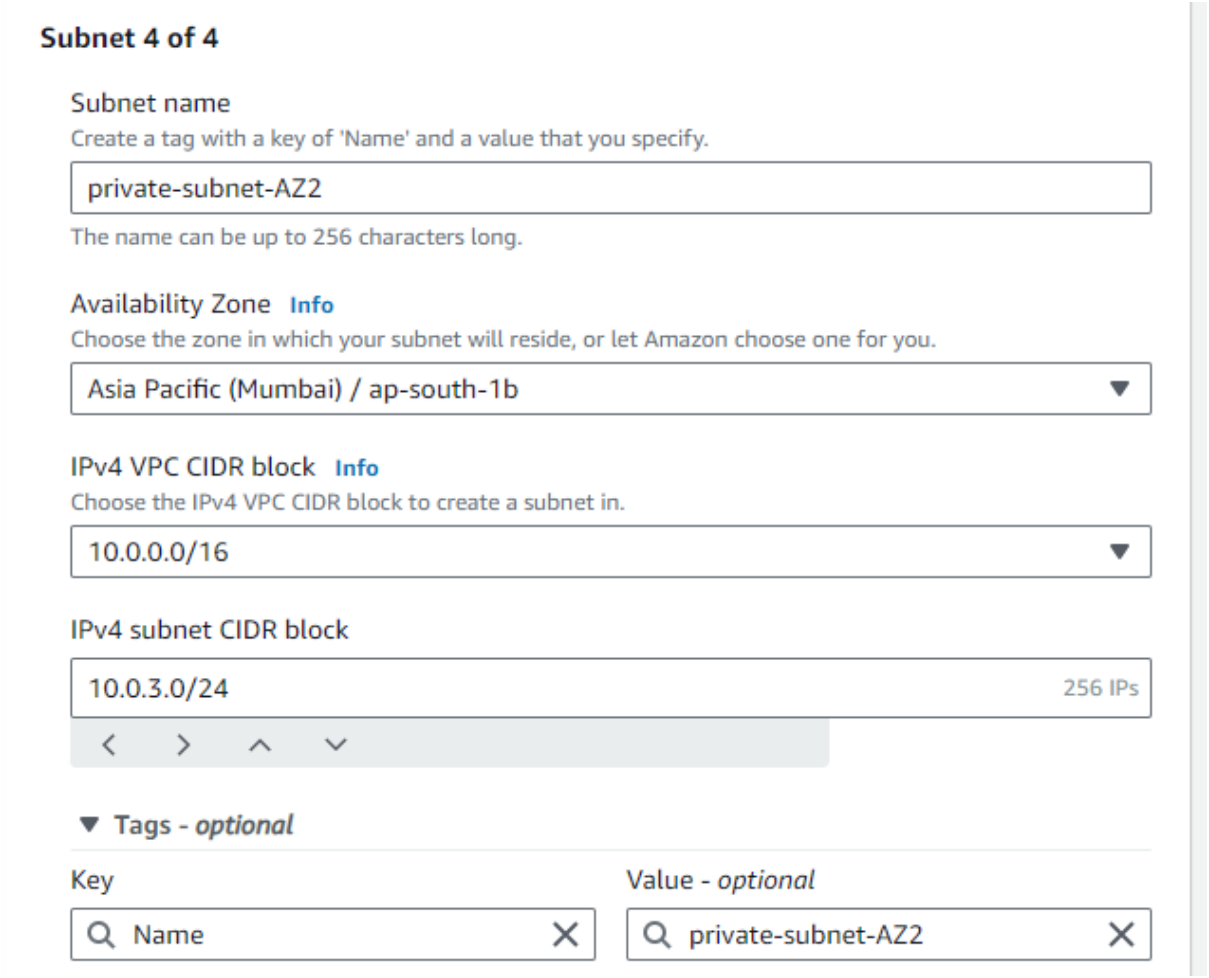
10.0.0.0/16

IPv4 subnet CIDR block

10.0.2.0/24 256 IPs

▼ Tags - *optional*

Key	Value - *optional*
Name	private-subnet-AZ1



STEP 5: Create a 5th and 6th subnet for the private Database, again in multi AZ.

Subnet 5 of 5

Subnet name

Create a tag with a key of 'Name' and a value that you specify.

DB-private-subnet-AZ1

The name can be up to 256 characters long.

Availability Zone Info

Choose the zone in which your subnet will reside, or let Amazon choose one for you.

Asia Pacific (Mumbai) / ap-south-1a

IPv4 VPC CIDR block Info

Choose the IPv4 VPC CIDR block to create a subnet in.

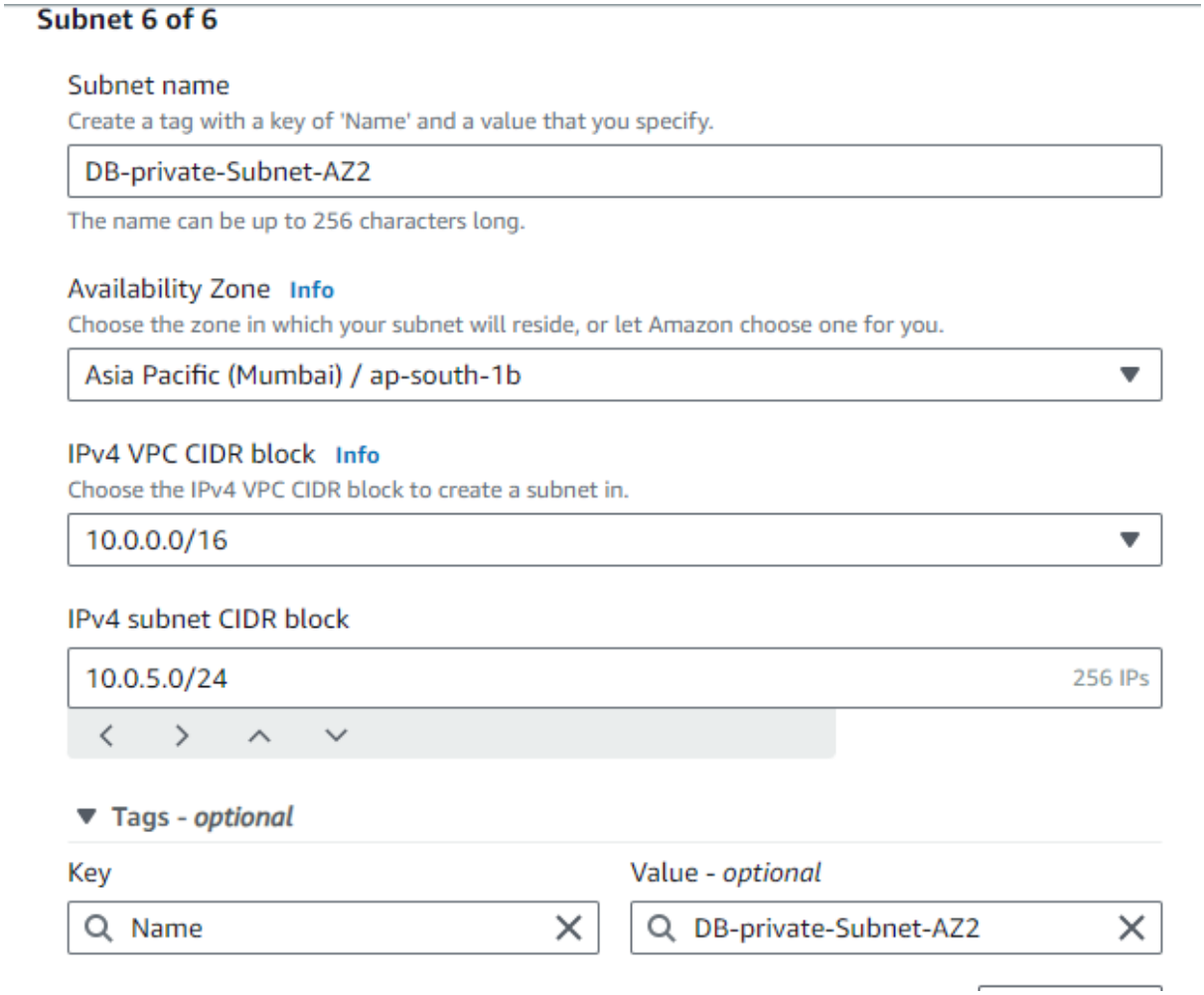
10.0.0.0/16

IPv4 subnet CIDR block

10.0.4.0/24 256 IPs

▼ Tags - *optional*

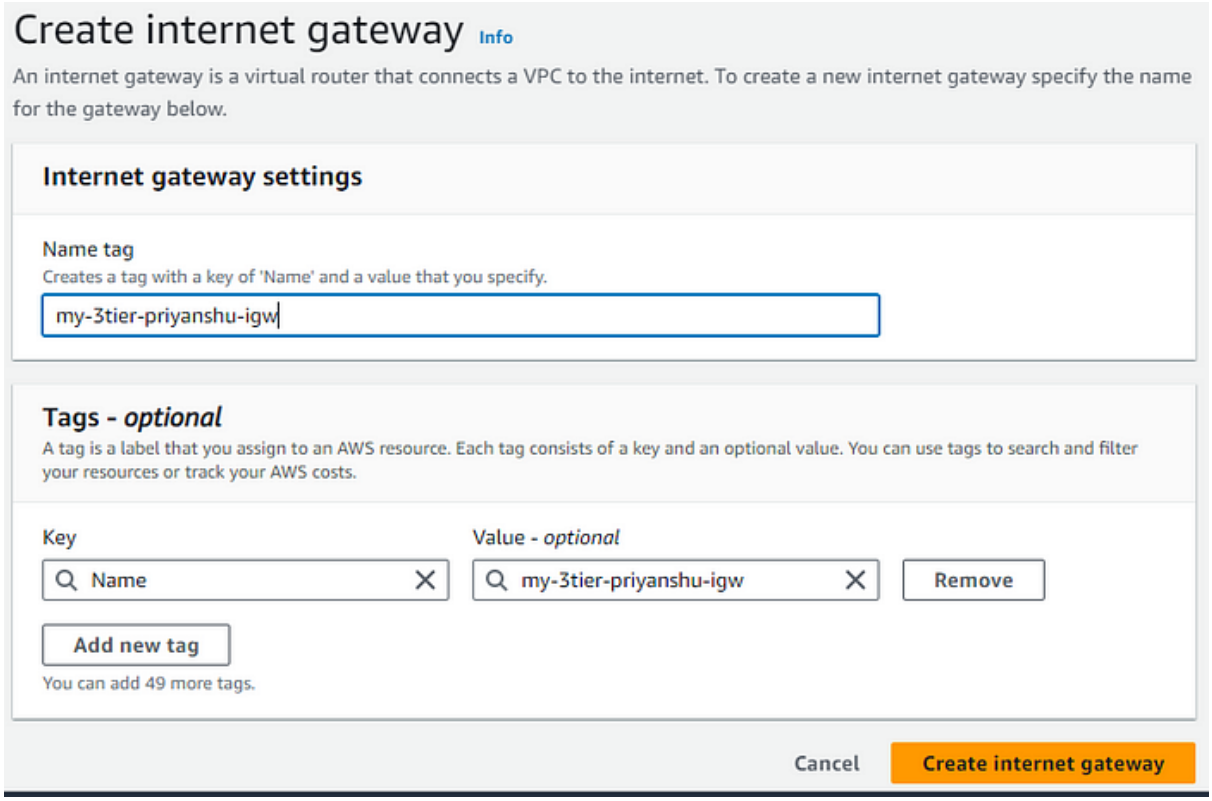
Key	Value - *optional*
Name	DB-private-subnet-AZ1



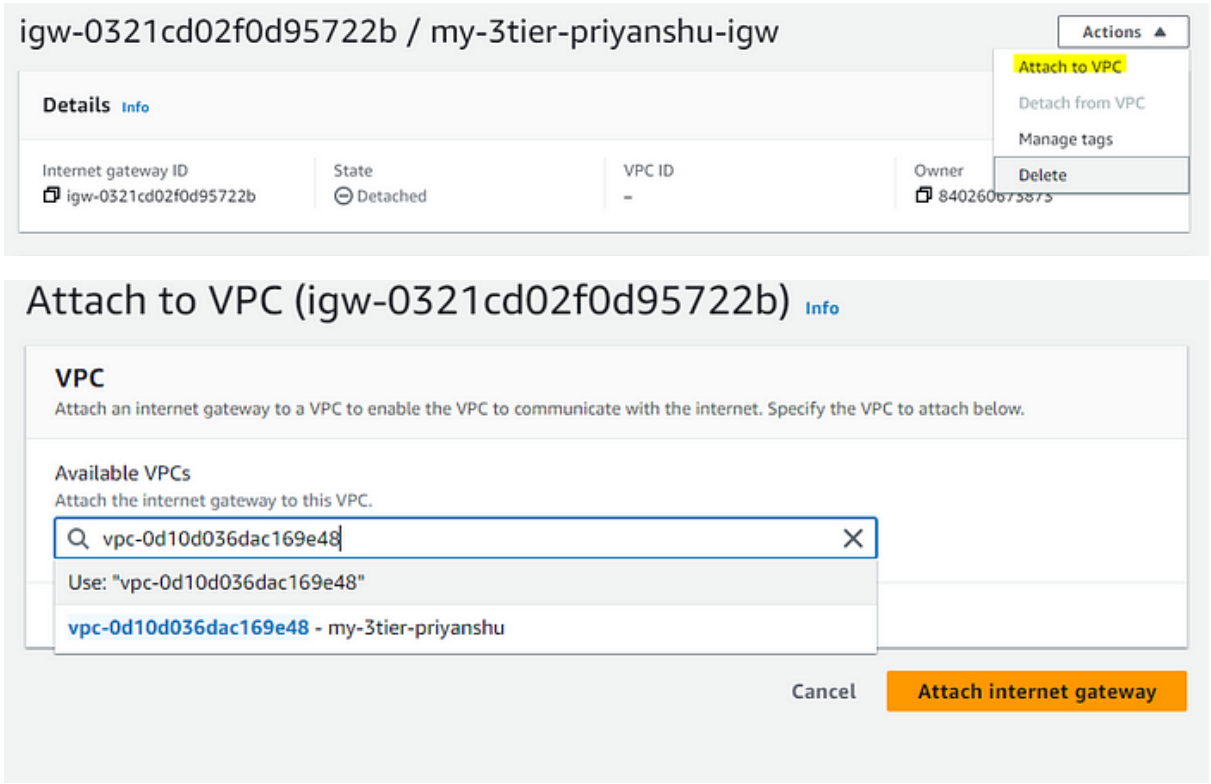
After all the configuration we'll be having 6 subnets, As shown below:

Name	Subnet ID	State
DB-private-Subnet-AZ2	subnet-0d91f1533863a65c5	Available
DB-private-subnet-AZ1	subnet-016c28dc844de2a02	Available
public-web-subnet-az2	subnet-001a75db22f70898d	Available
public-web-subnet-AZ1	subnet-076df2f16efa400e7	Available
private-subnet-AZ2	subnet-071856a3d56840d6e	Available
private-subnet-AZ1	subnet-0e5a9b907a0f3d15b	Available

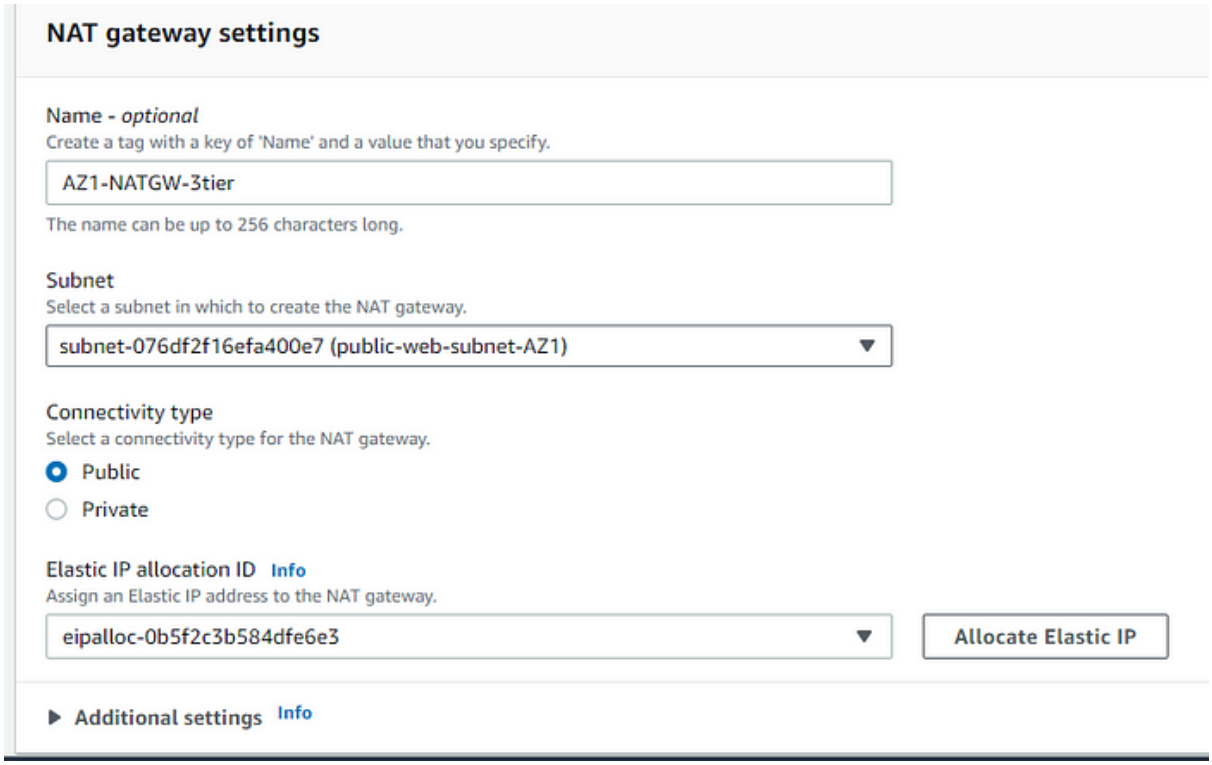
STEP 6: Create an internet Gateway, IGW is used to route traffic from internet to instance and vice versa, if all the newtork restrictions are allowed.



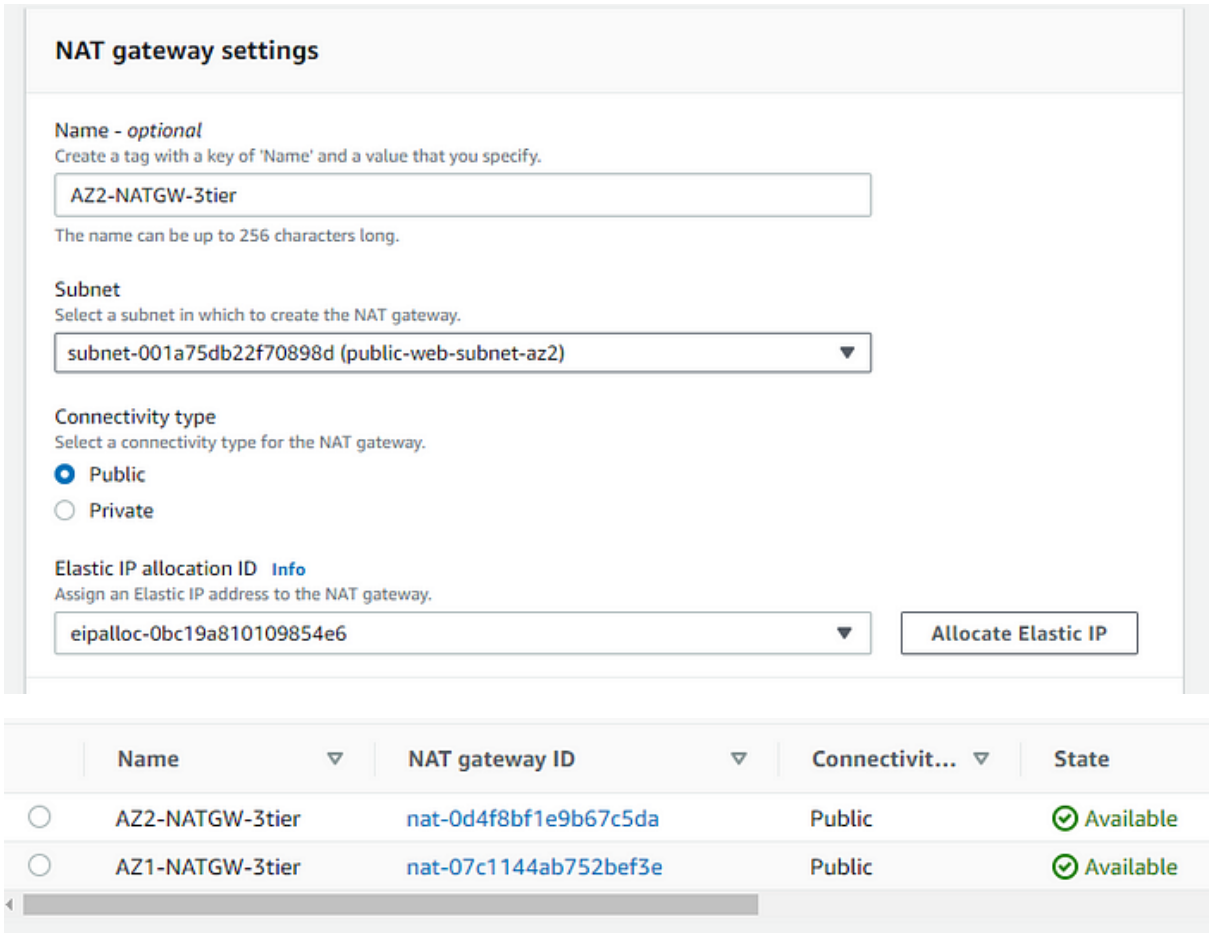
STEP 7: Attach this to the VPC which we just created above.



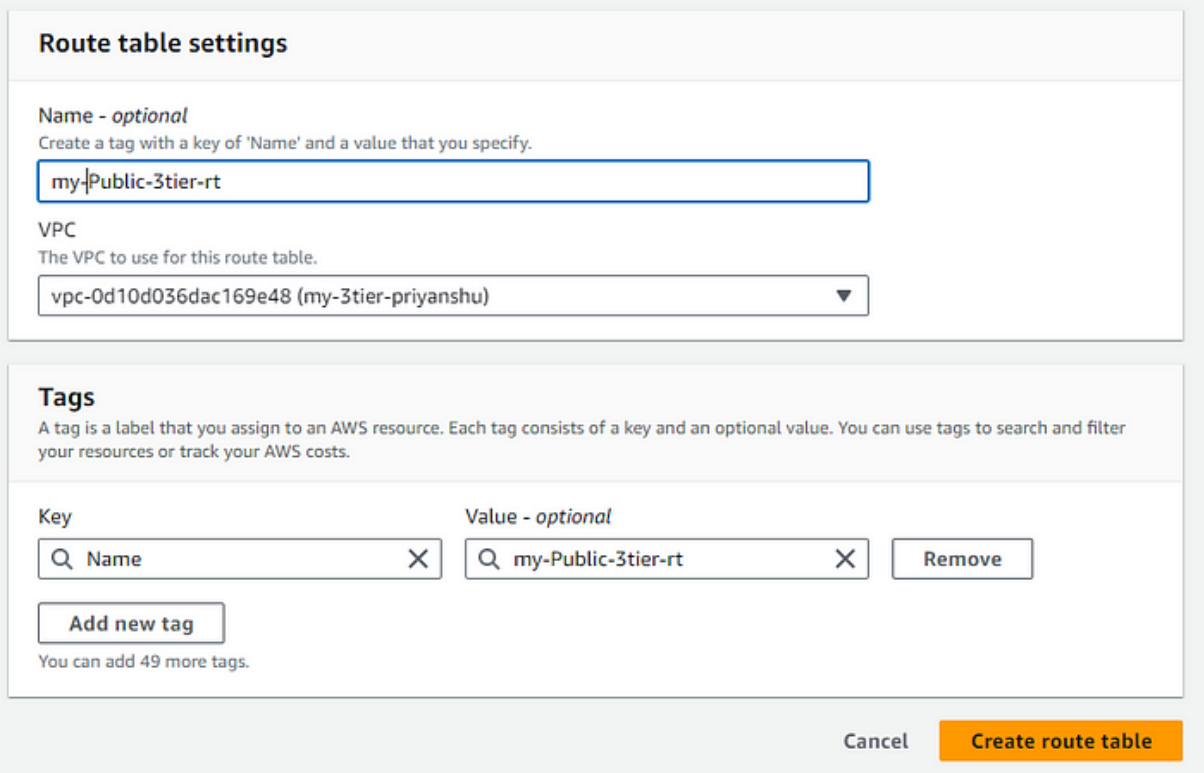
STEP 8: Create NAT Gateway, NAT in simpler allows instance to access network outside the instance, but doesn't allow external Traffic to come inside the instance.As its A subnet level resource we attach the first NAT to the public subnet 1 in AZ1.



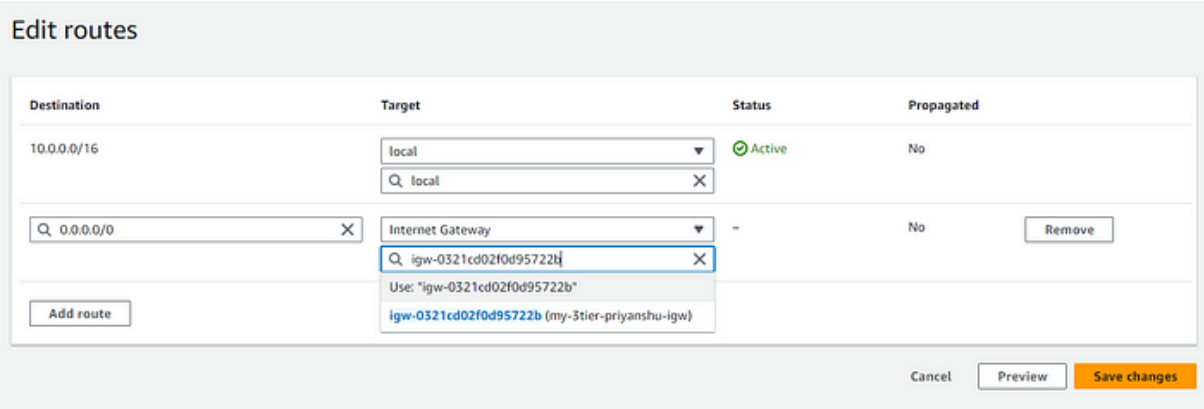
STEP 9: Create a second NAT Gateway and attach it to the second public AZ subnet.



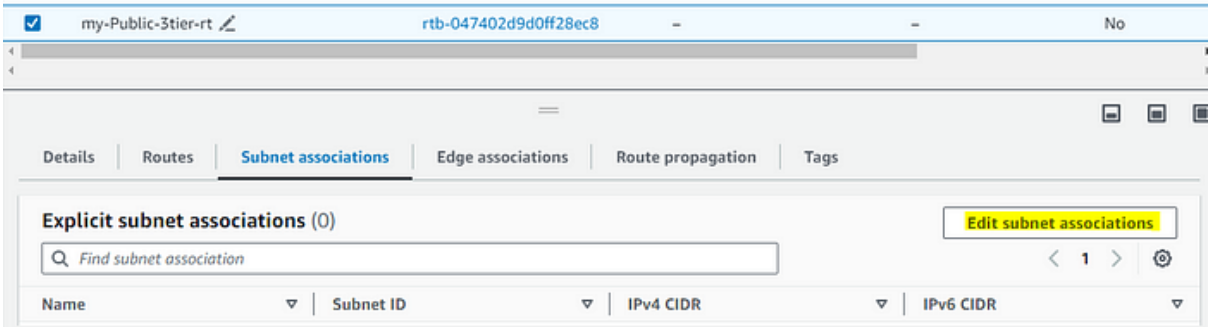
STEP 10: Create the route Tables with the specified subnet and routes.

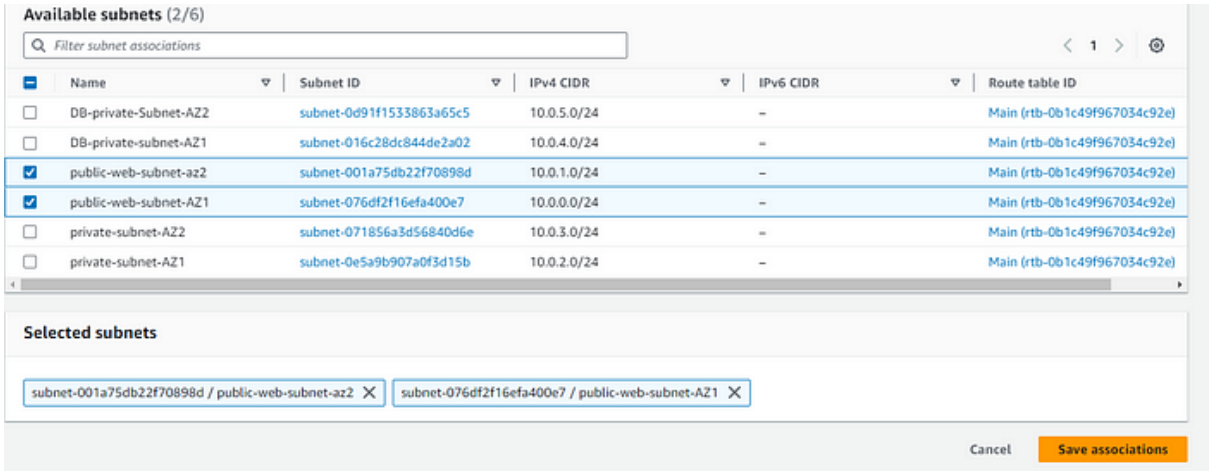


STEP 11: Edit route table for public destination internet and source IGW.

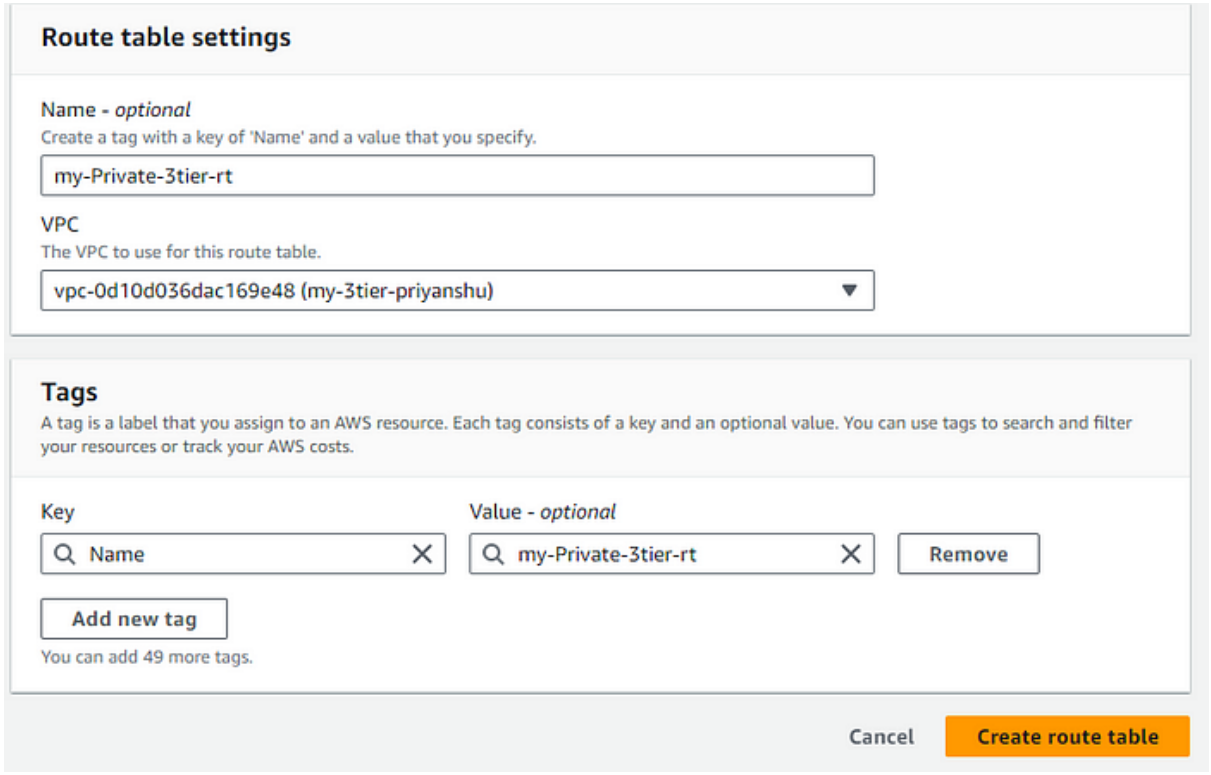


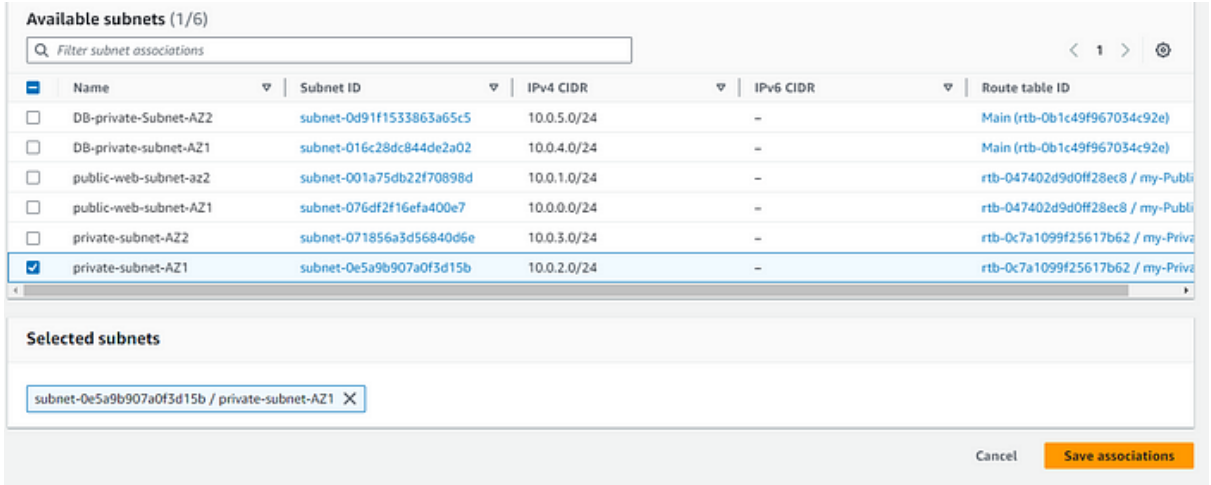
STEP 12: Subnet Association with the public subnet.

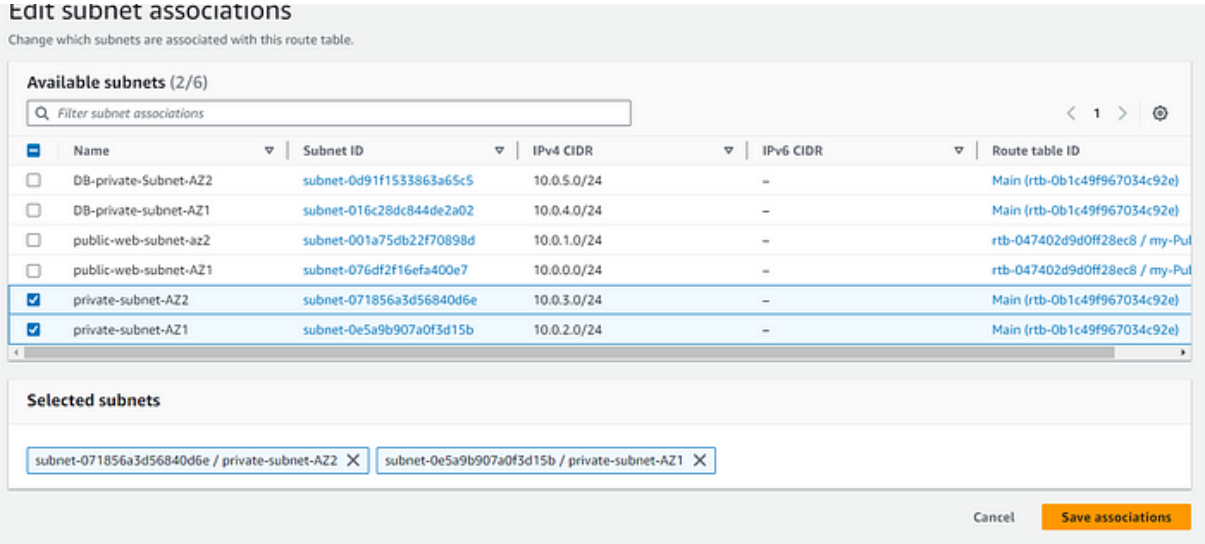


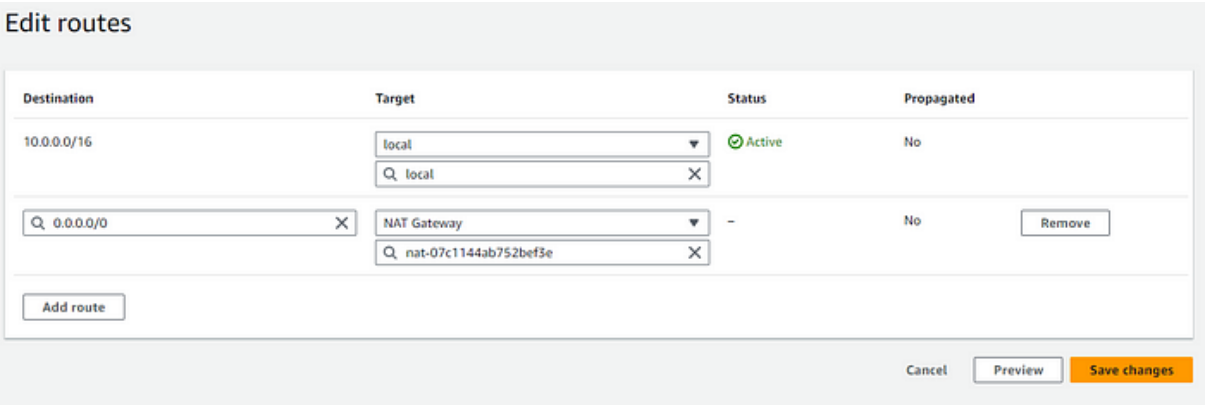


Create route table for the private subnet, but this time in routes add Nat Gateway, Private Subnet.







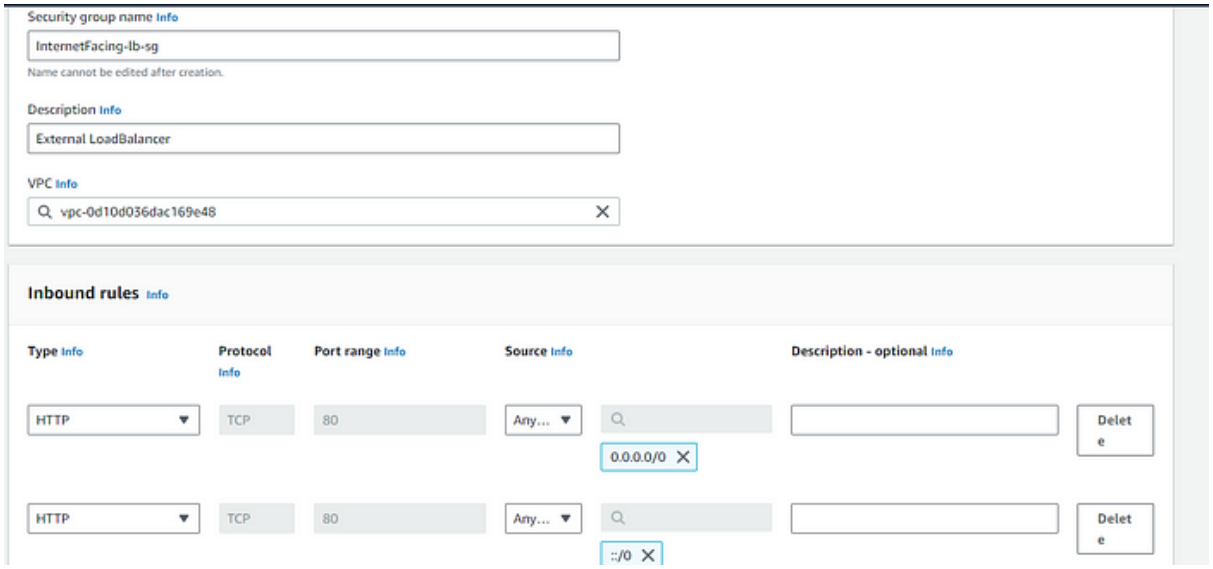


CREATING SECURITY GROUPS

In this section we will be creating security groups for all requirments. Security groups plays an integral part of the AWS security best practices, Regulating them can be benificial for you overall security.

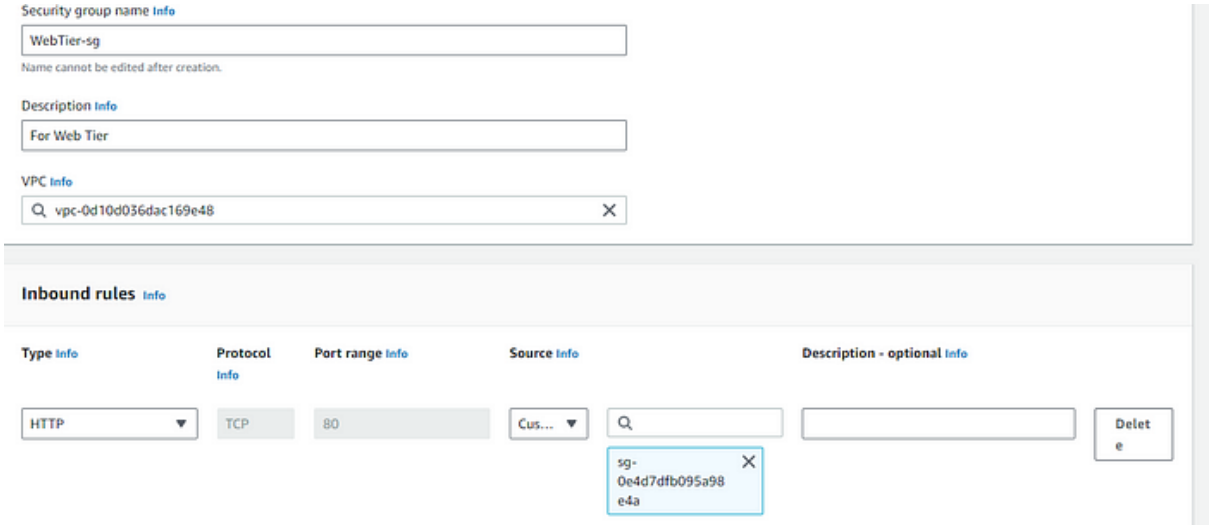
1. ALB security group:

This allow's internet traffic to reach you ALB.



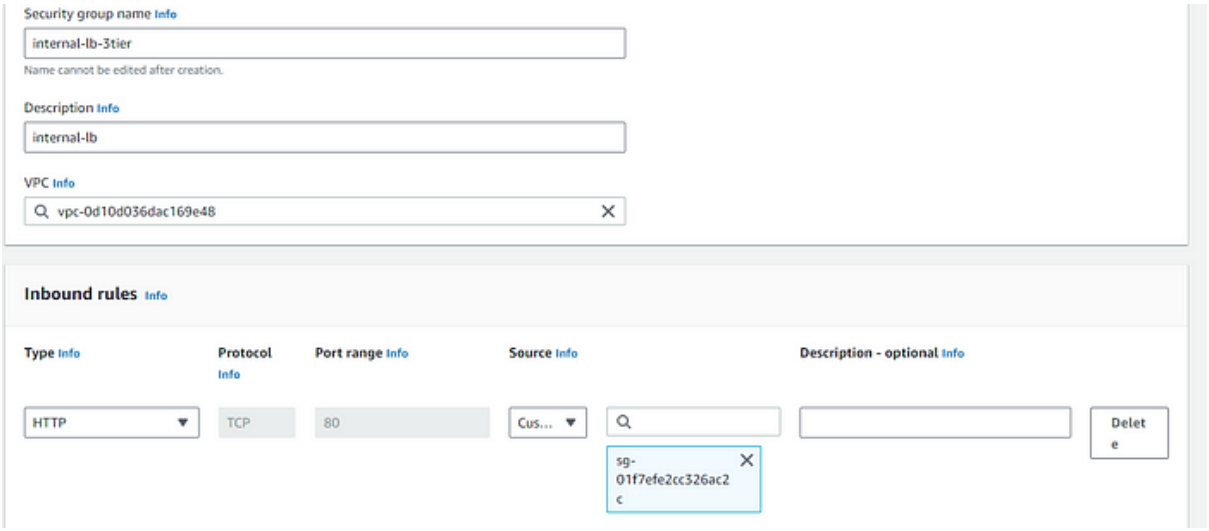
2. Web Tier Security Group:

This allows traffic from ALB to web Tier.



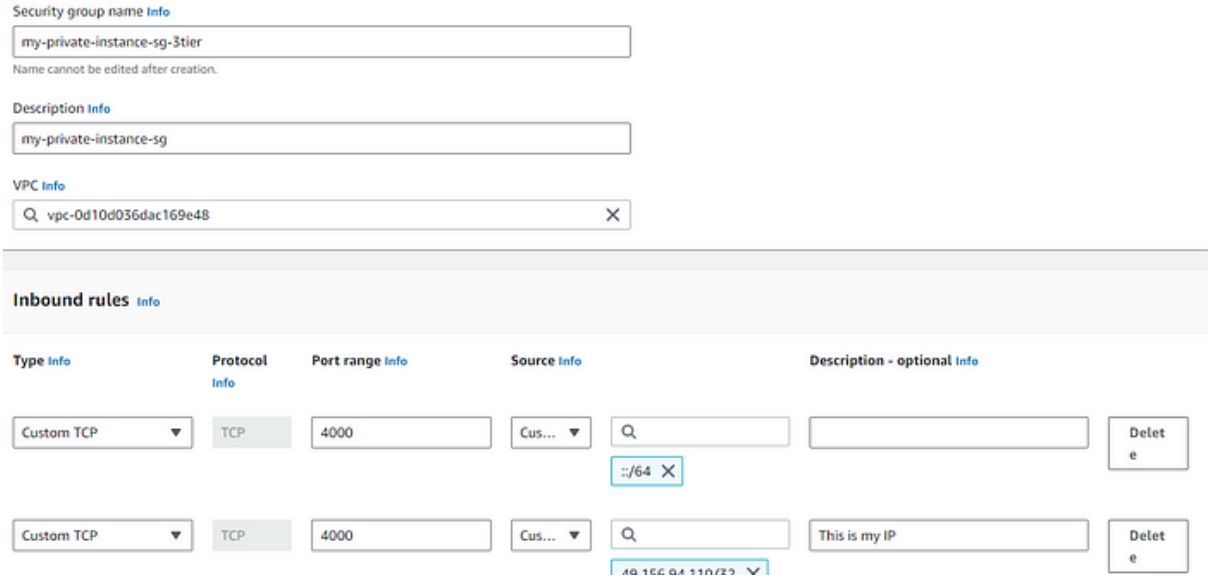
3. Internal LB SG:

There's an Internal ALB security group, which internally used for networking.



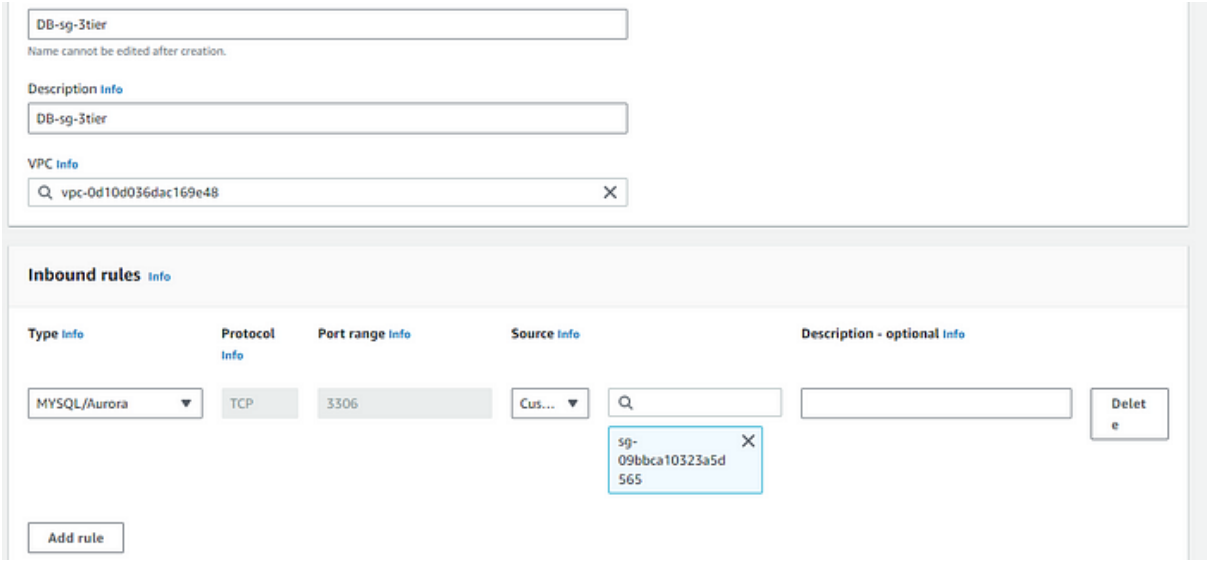
4. SG for private instance:

These allows only Certain IP(MY IP) to access the private instance.



5. DB private SG:

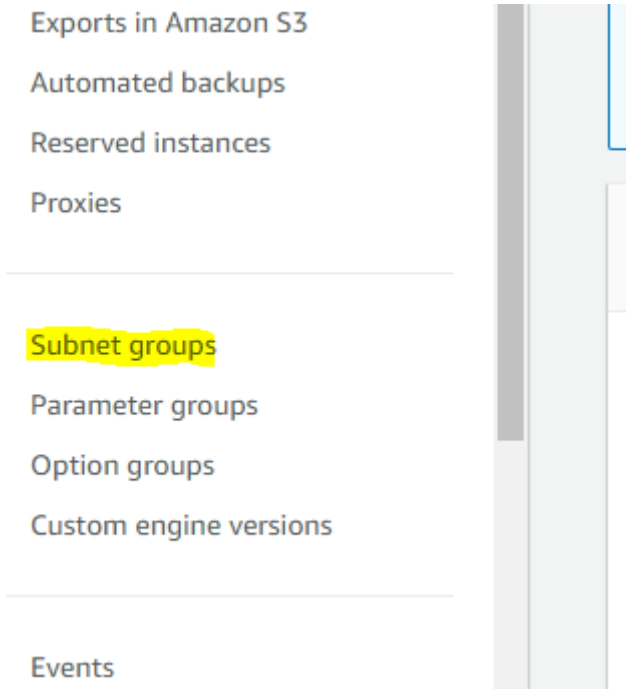
This allows the DB to be accessible from the instance.

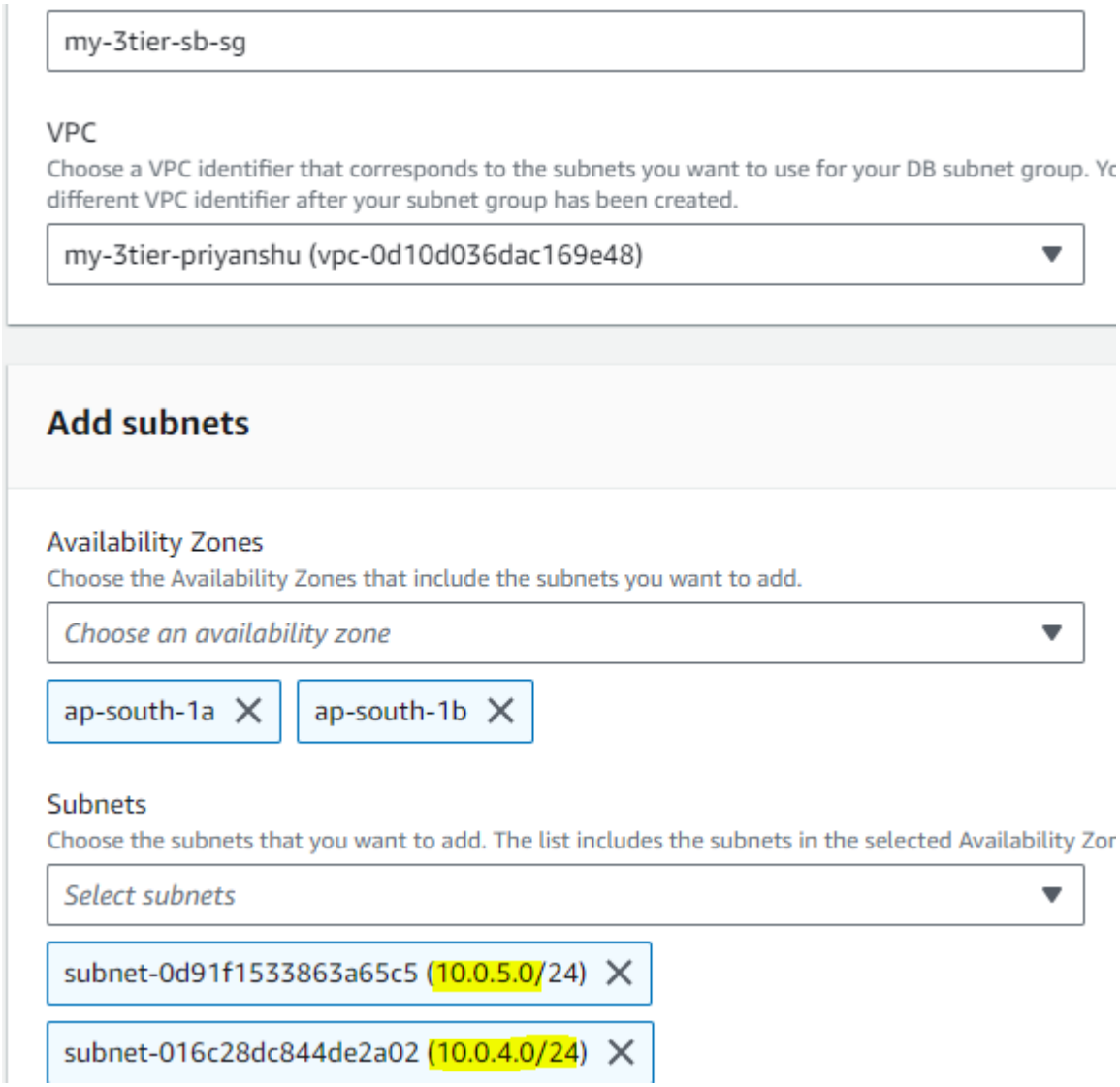


Creating Database

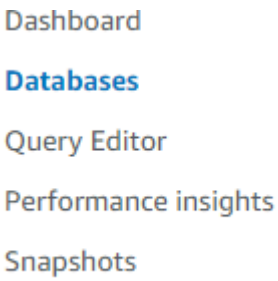
In this Setup We are using RDS as a Database, In This secrion we'll be doing configurations related to the RDS intance.

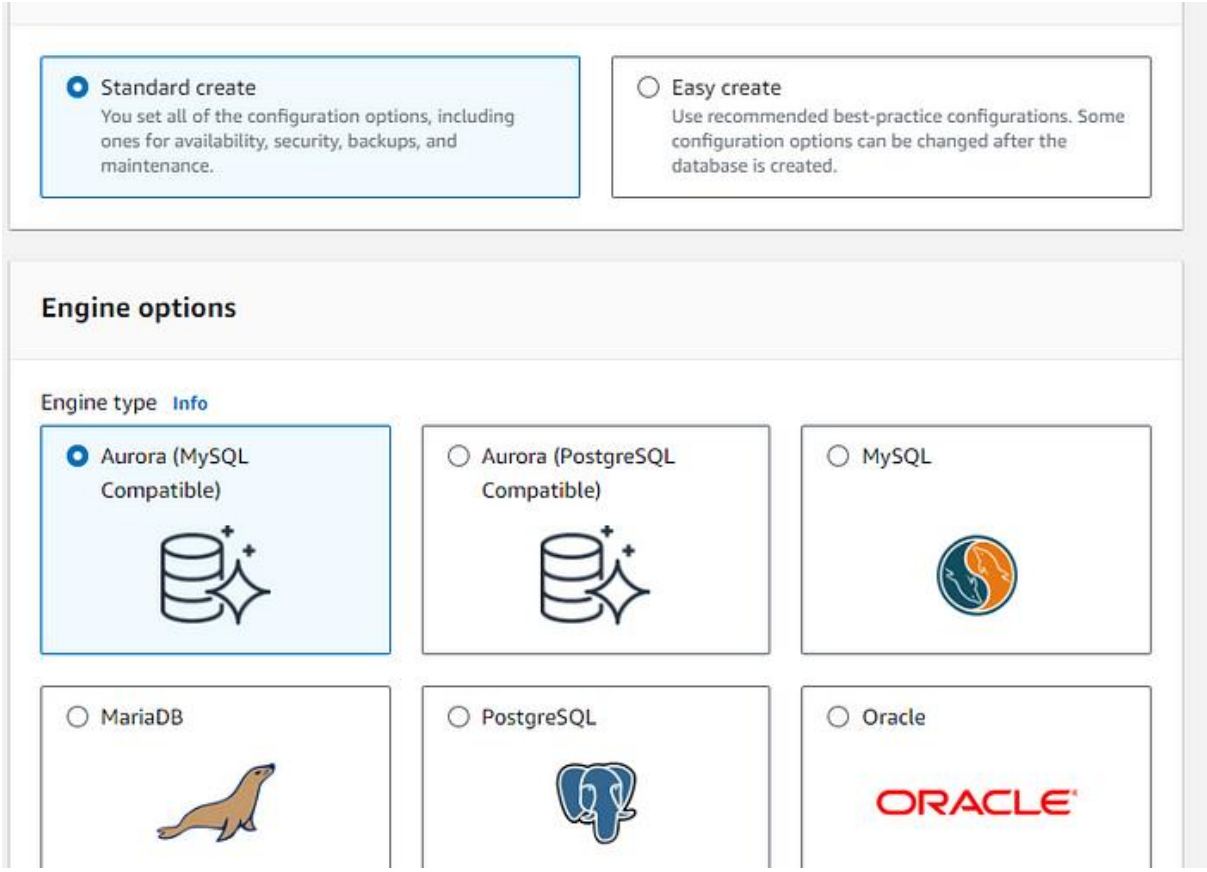
STEP 1: Creating Subnet Group for RDS



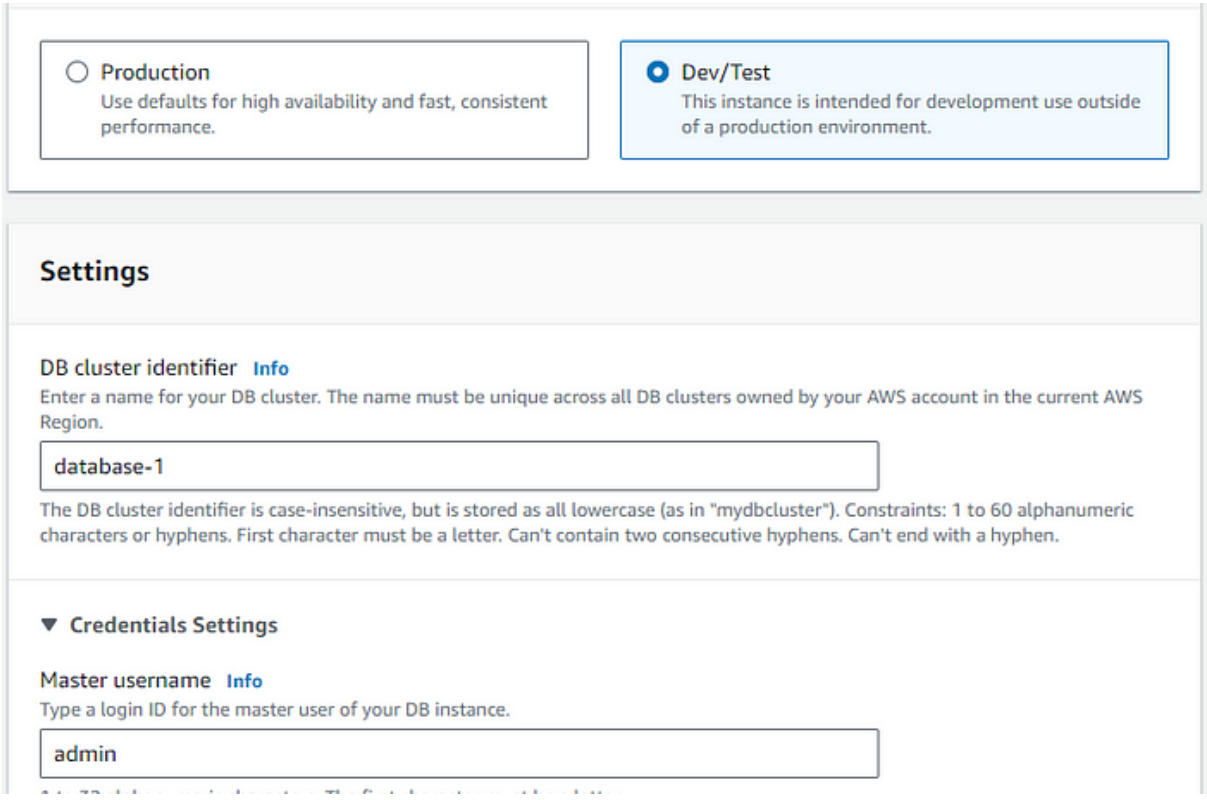


STEP 2: After Subnet Group create Database.We'll be creating a minimal setup of the RDS database as this is for testing purpose, you should not use this for a production setup.

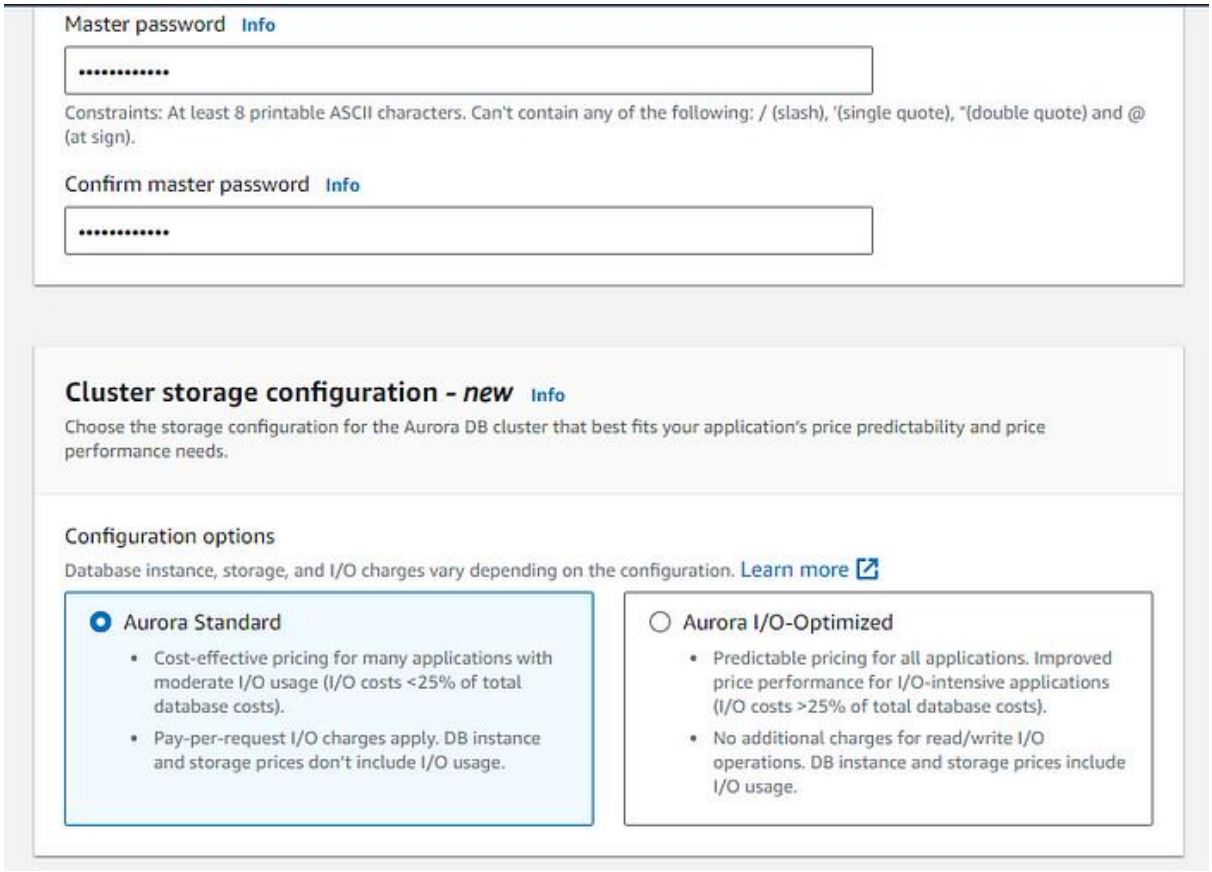


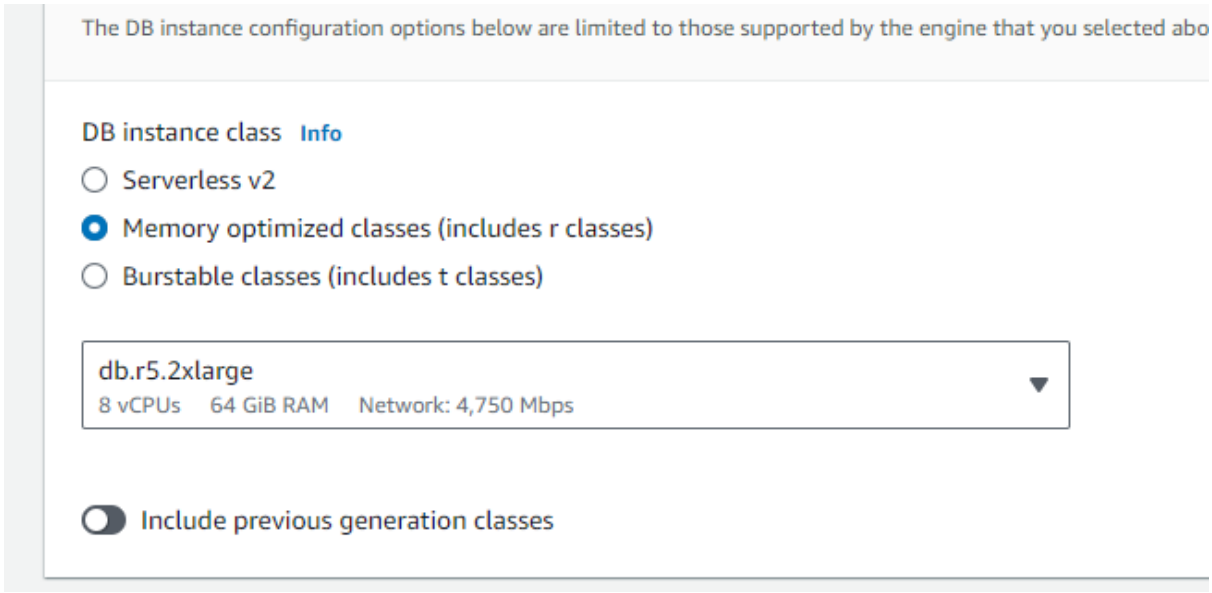


2. Using the Dev Test version for lesser costing.

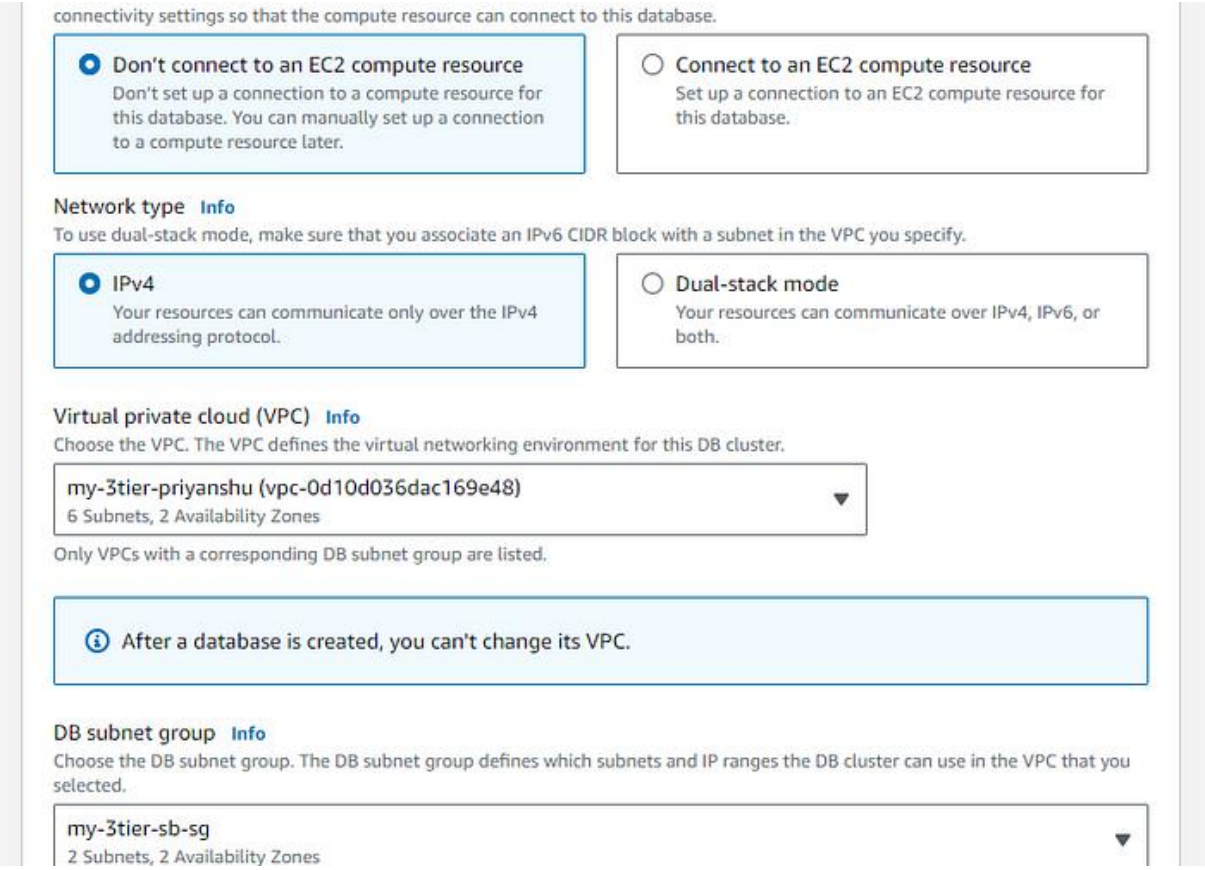


STEP 3: Create a solid Master password for your database access.

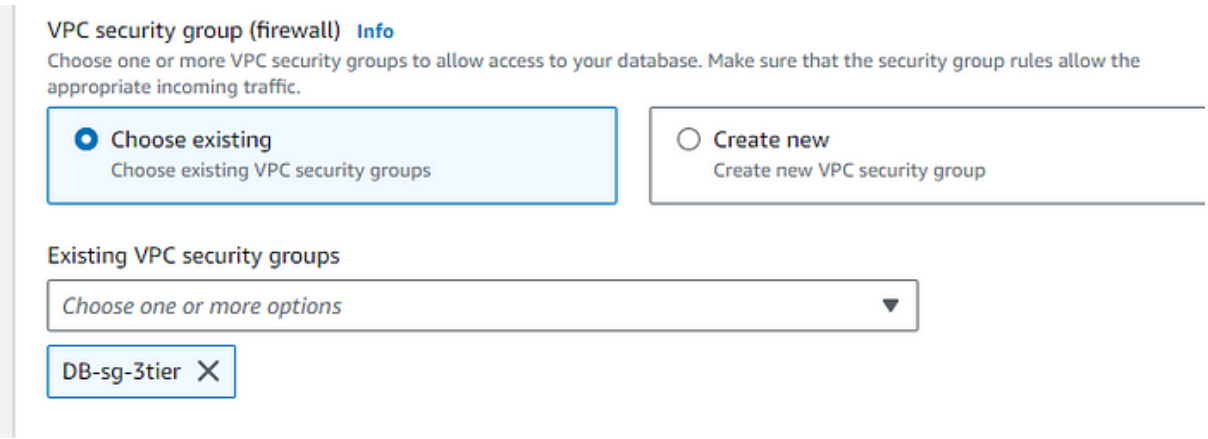




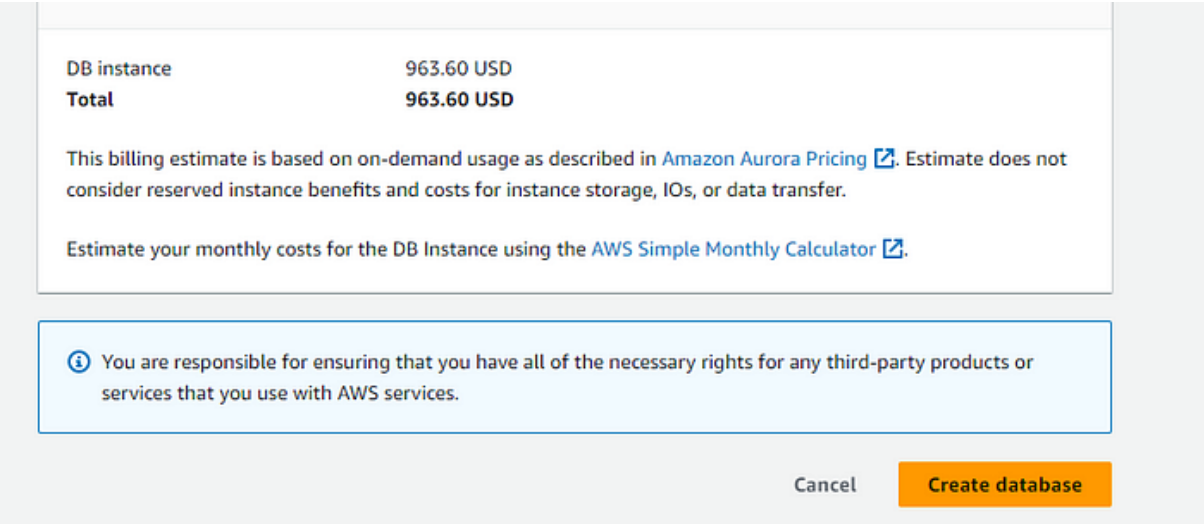
STEP 4: Disable the ec2 connect options as we'll be managing that independently using security groups.



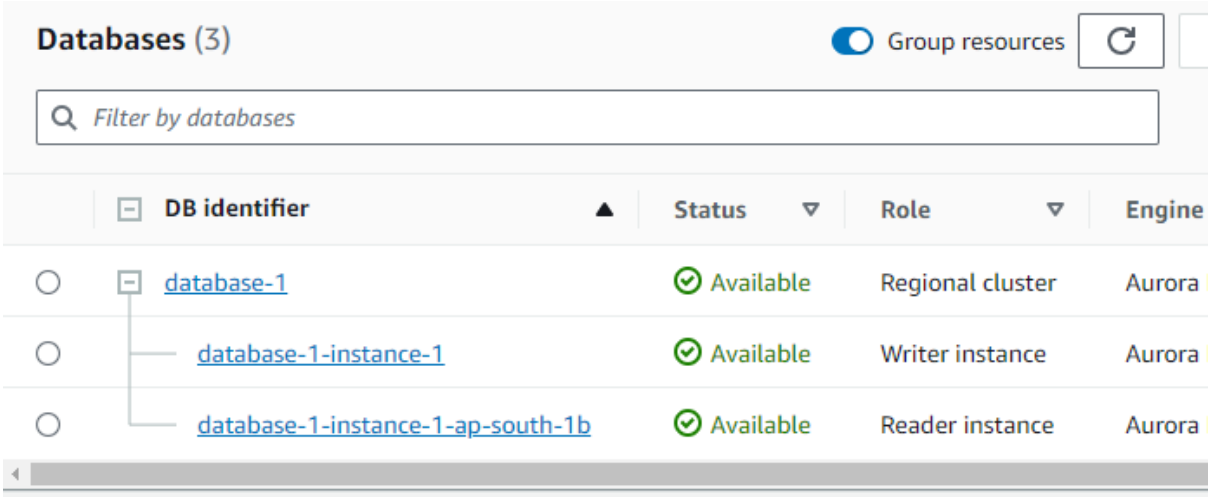
STEP 5: Attach the VPC and security group for the database that we just created in the security group section.



As we have used most of the things for demo, This setup can cost you for database. You can fine tune more settings for reducing you infra cost.

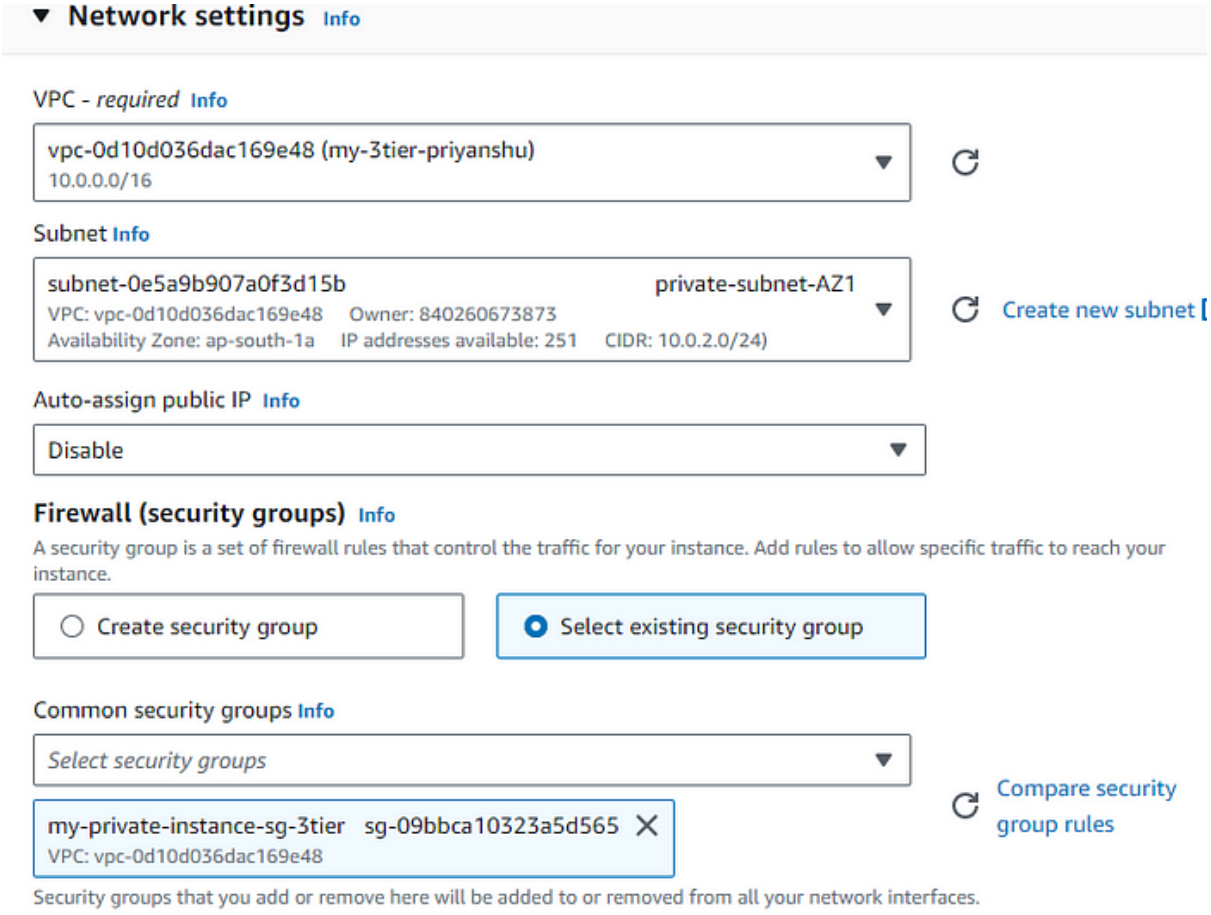


Wait for some time and you can see your database instance:

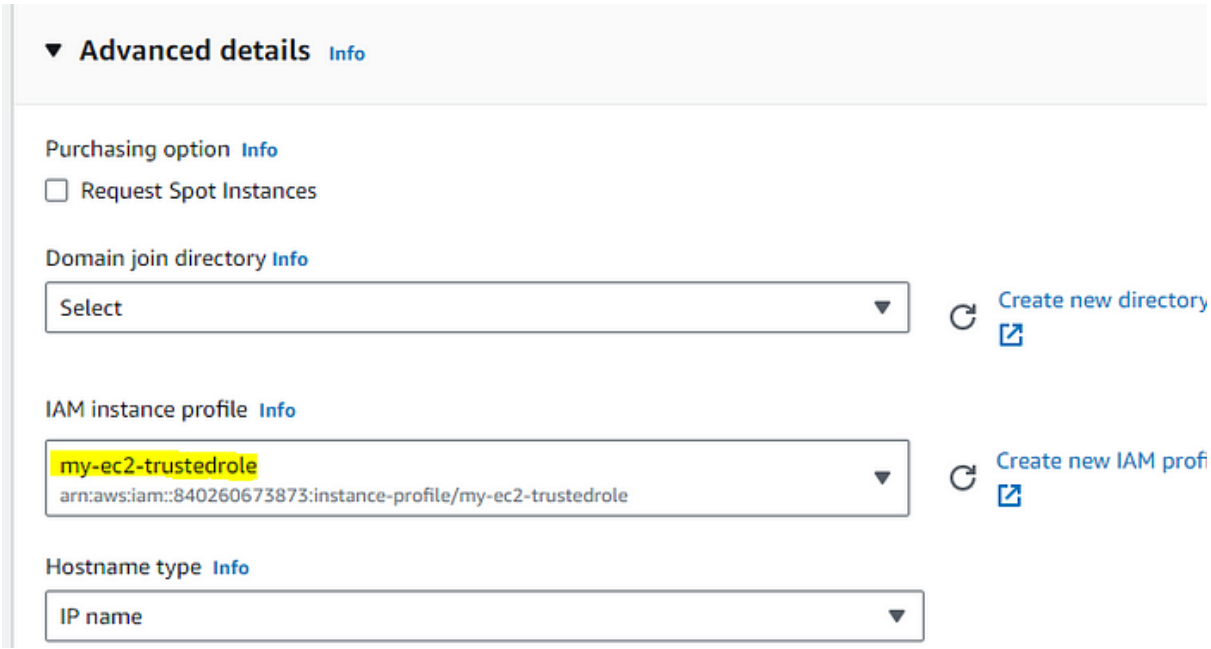


Instance Management

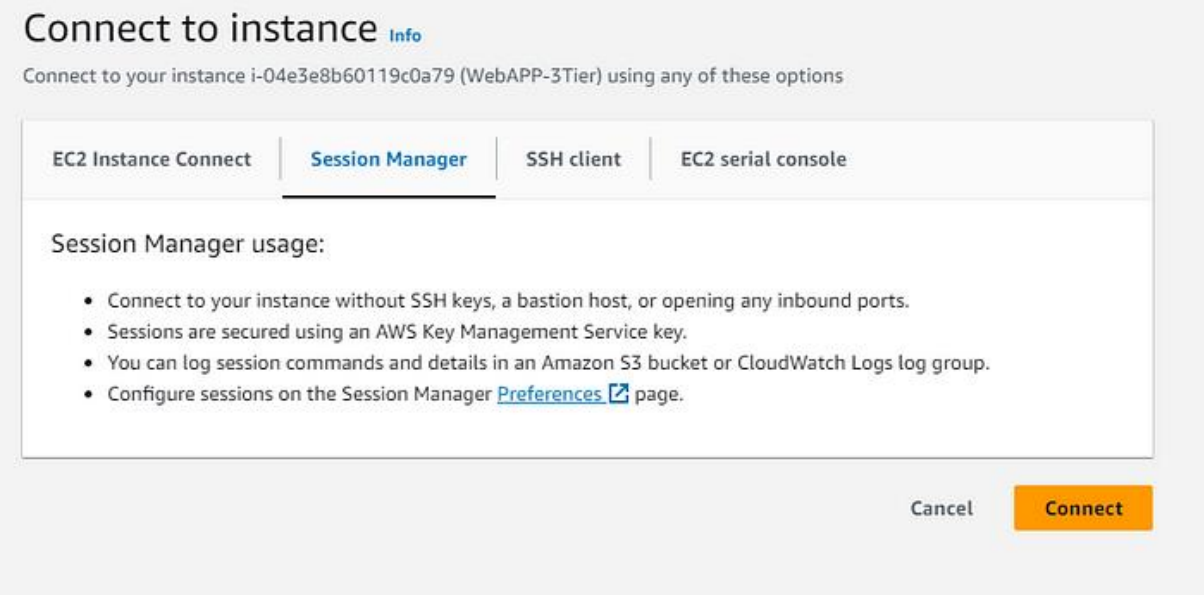
STEP 1: Create first instance in the private subnet with no public IP, attach the created security group to it.



STEP 2: Attach an role we just created in the first part.



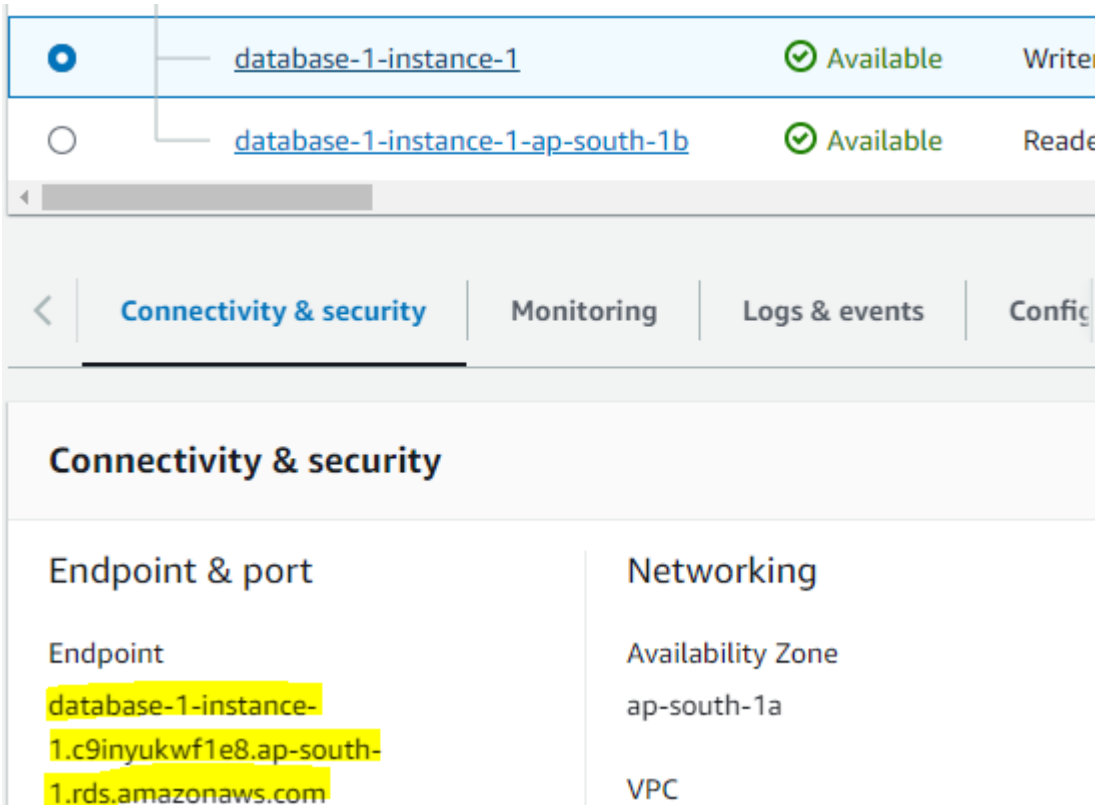
We'll be connecting this instance using the session manager, for better management of the instance accessibility without using the keys, we can use the session manager.



STEP 3: For Testing purpose, Download the myswl dependencies for testing your RDS Database accessibility.

```
[root@ip-10-0-2-134 ~]# sudo yum install https://dev.mysql.com/get/mysql57-community-release-e
l7-11.noarch.rpm
Last metadata expiration check: 0:14:53 ago on Tue Oct 17 11:54:53 2023.
mysql57-community-release-el7-11.noarch.rpm                  66 kB/s |  25 kB     00:00
Dependencies resolved.
================================================================================
 Package                    Architecture   Version          Repository          Size
================================================================================
Installing:
 mysql57-community-release  noarch         el7-11           @commandline        25 k

Transaction Summary
================================================================================
Install  1 Package

Total size: 25 k
Installed size: 31 k
Is this ok [y/N]: y
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                        1/1
```

STEP 4: Copy the writer endpoint from the RDS service page.



STEP 5: login the database using the endpoint. If you are unable to login then you must check your database security group settings.

```
[root@ip-10-0-2-134 ~]# mysql -h database-1-instance-1.c9inyukwf1e8.ap-south
-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 163
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

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

Type 'help;' or '\h' for help. Type '\c' to clear the current input statemen
t.

mysql>
```

STEP 6 Adding data in the database.

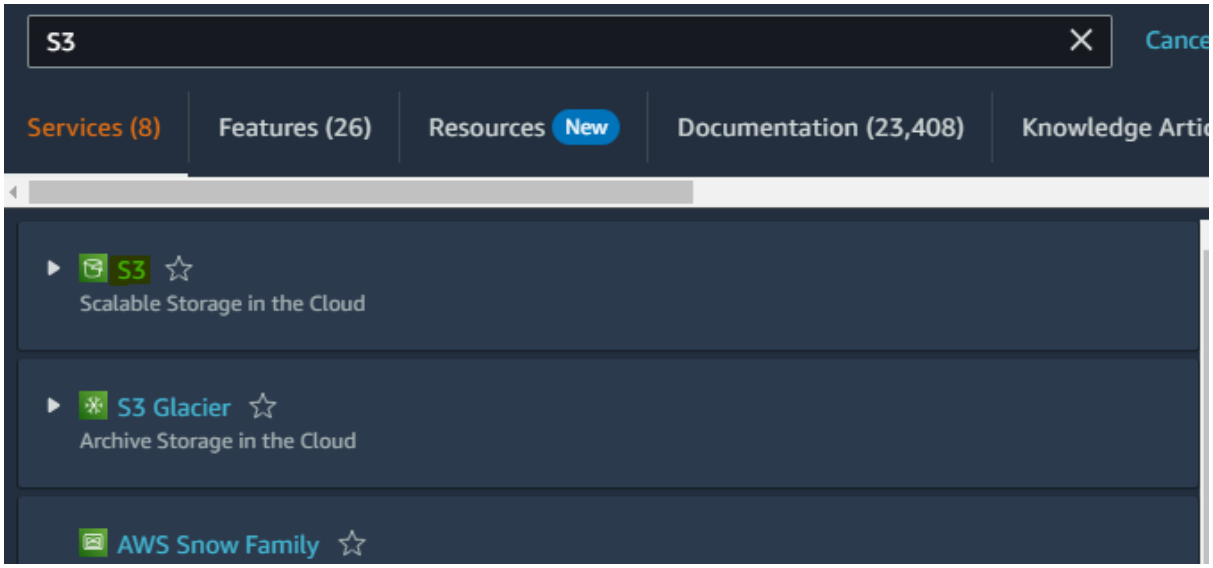
```
mysql> create DATABASE webappdb;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| webappdb           |
+--------------------+
5 rows in set (0.00 sec)

mysql> use webappdb;
Database changed
mysql> CREATE TABLE IF NOT EXISTS transactions(id INT NOT NULL
    -> AUTO_INCREMENT, amount DECIMAL(10,2), description
    -> VARCHAR(100), PRIMARY KEY(id));
Query OK, 0 rows affected (0.02 sec)

mysql> show tables
    -> ^C
mysql> show tables;
+--------------------+
| Tables_in_webappdb |
+--------------------+
| transactions       |
+--------------------+
1 row in set (0.00 sec)
```

```
mysql> INSERT INTO transactions (amount,description) VALUES ('400','grocerie
s');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM transactions;
+----+--------+-------------+
| id | amount | description |
+----+--------+-------------+
|  1 | 400.00 | groceries   |
+----+--------+-------------+
1 row in set (0.00 sec)

mysql>
```

Refer to this code repository: *git clone* https://github.com/aws-samples/aws-three-tier-web-architecture-workshop.git

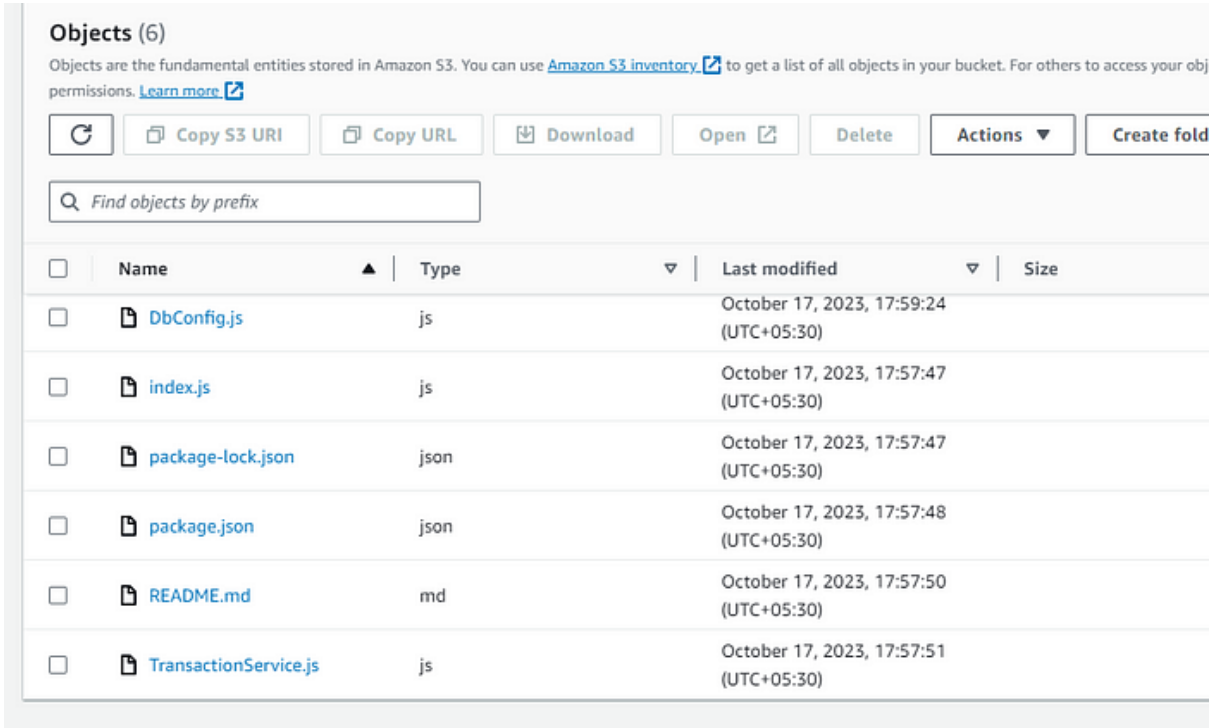
Setting up S3 bucket Code Section

In this section we will be adding the code repository in the s3 bucket storage.



STEP 1: edit the DB config file with the db specific argument, like writer endpoints.

STEP 2: Upload the files of the app tier code in this S3 bucket.



STEP 3: Testing this code Successfully running node application using pm2.

```
[root@ip-10-0-2-134 app-tier]# pm2 start index.js
[PM2] Applying action restartProcessId on app [index](ids: [ 0 ])
[PM2] [index](0) ✓
[PM2] Process successfully started
```

id	name	mode	↺	status	cpu	memory
0	index	fork	15	online	0%	12.4mb

```
[root@ip-10-0-2-134 app-tier]# pm2 list
```

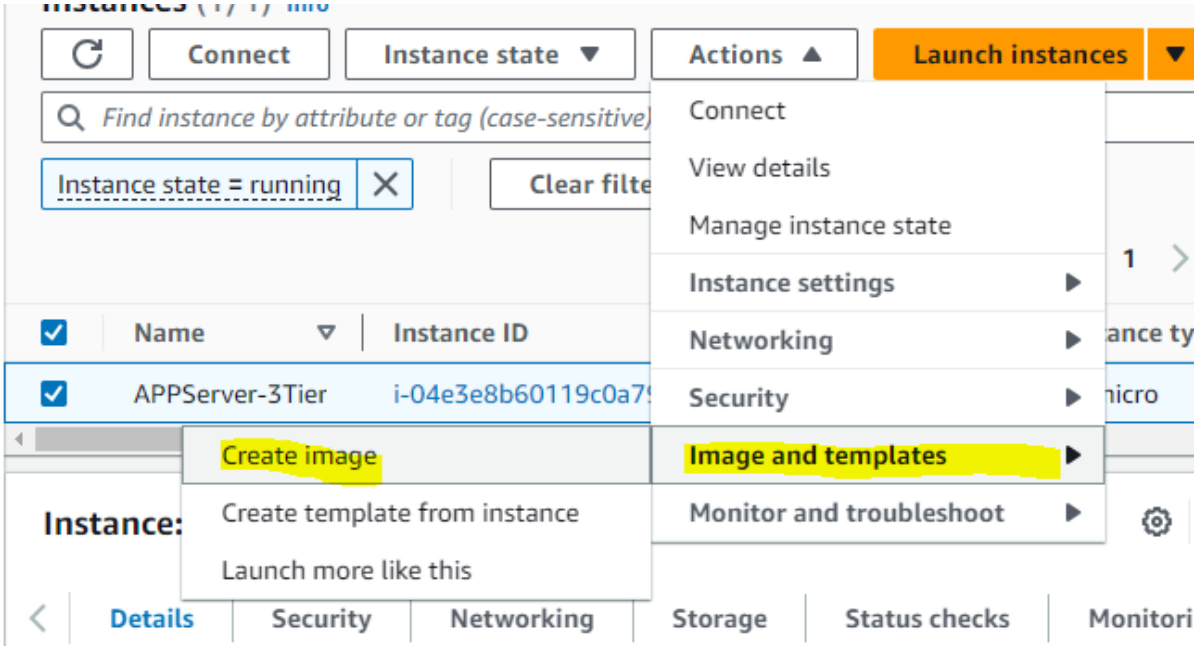
id	name	mode	↺	status	cpu	memory
0	index	fork	15	online	0%	56.8mb

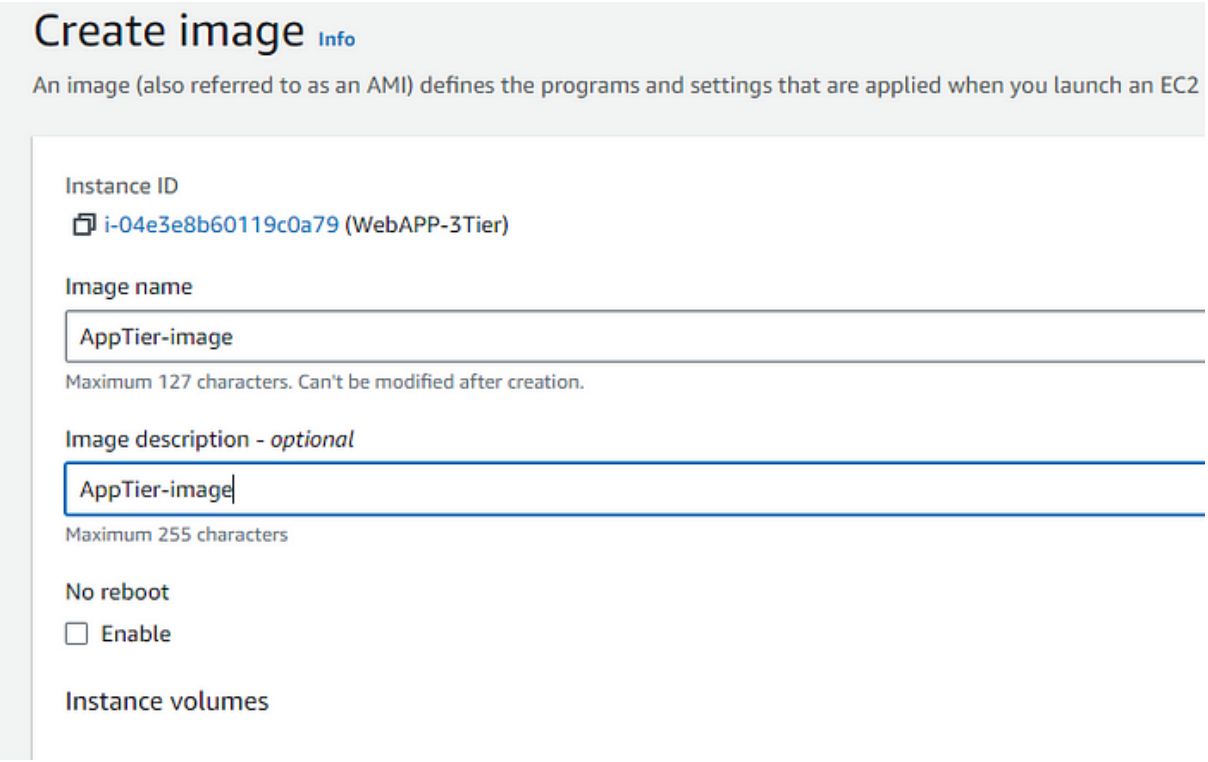
STEP 4: Testing Endpoints.

```
[root@ip-10-0-2-134 app-tier]# curl http://localhost:4000/health
"This is the health check"[root@ip-10-0-2-134 app-tier]#
```

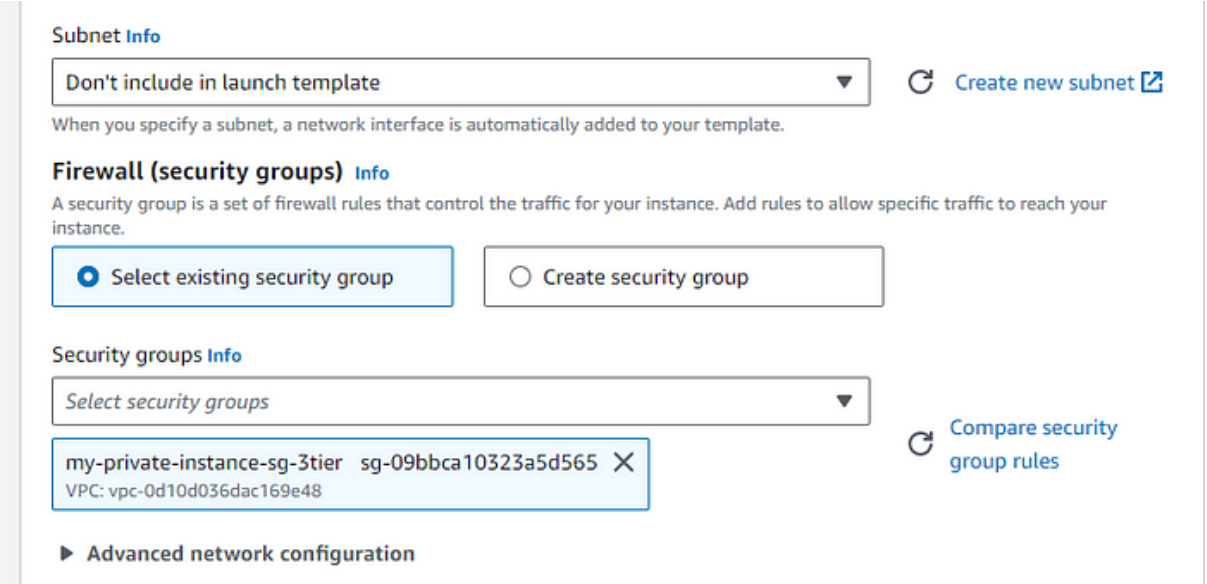
```
[root@ip-10-0-2-134 app-tier]# curl http://localhost:4000/transaction
{"result":[{"id":1,"amount":400,"description":"groceries"}]}[root@ip-10-0-2-134 ap
p-tier]#
```

STEP 5: AS this is an multi AZ setup hence, We have to do the same setup in the second AZ, Hence we'll be creating an Image out of this, And then Creating the Instance in the second AZ.

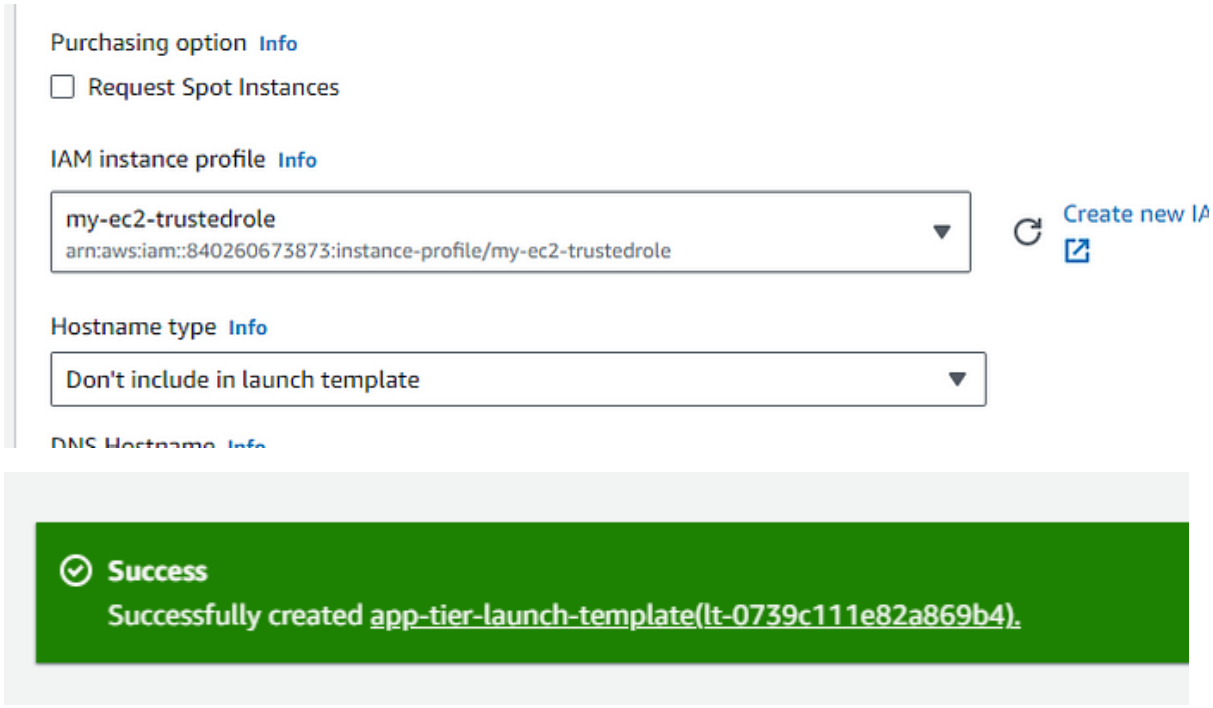




Security group for private instance and as its app tier



choose the instance profile from the advance section



STEP 6: Creating Target Groups, for load balancing of the app tier.

Choosing the cutom VPC we created in the earlier steps, 4000 is the target port.

Target group name

App-Tier-TG

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

Protocol: HTTP : Port: 4000

1-65535

IP address type

Only targets with the indicated IP address type can be registered to this target group.

- (•) IPv4
  Each instance has a default network interface (eth0) that is assigned the primary private IPv4 address. The instance's primary private IPv4 address is the one that will be applied to the target.
- ( ) IPv6
  Each instance you register must have an assigned primary IPv6 address. This is configured on the instance's default network interface (eth0). Learn more

VPC

Select the VPC with the instances that you want to include in the target group. Only VPCs that support the IP address type selected abov are available in this list.

my-3tier-priyanshu
vpc-0d10d036dac169e48
IPv4: 10.0.0.0/16

Protocol version

- (•) HTTP1

/health as the health check endpoint.

Health checks

The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

Health check protocol

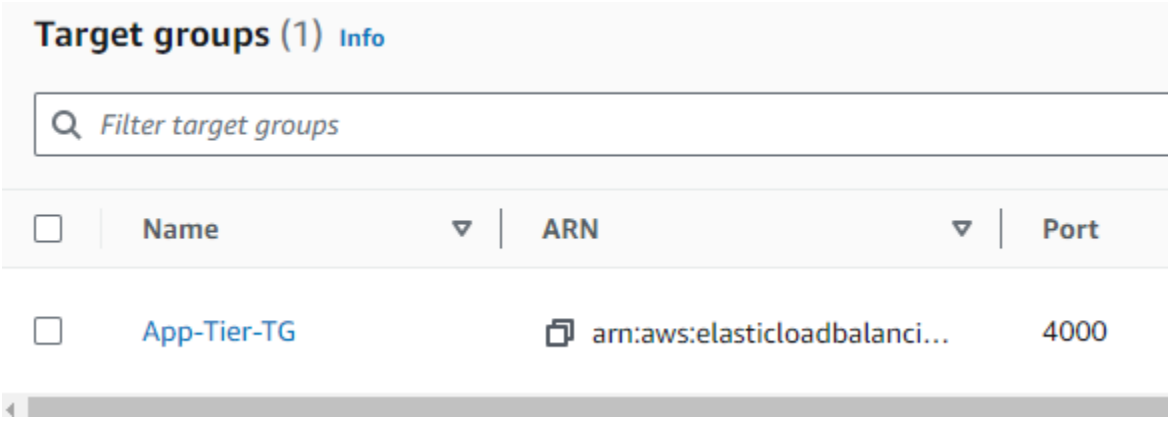
HTTP

Health check path

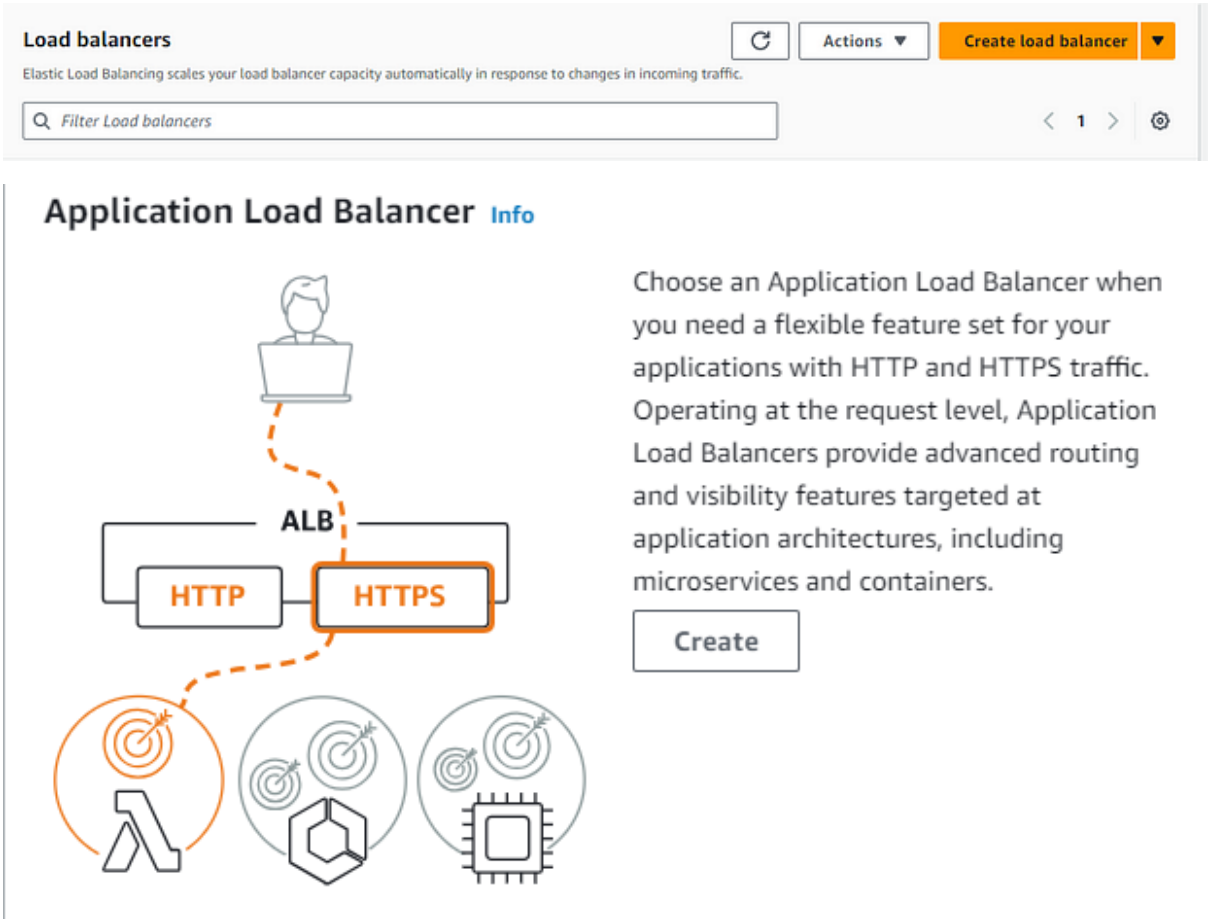
Use the default path of "/" to ping the root, or specify a custom path if preferred.

/health

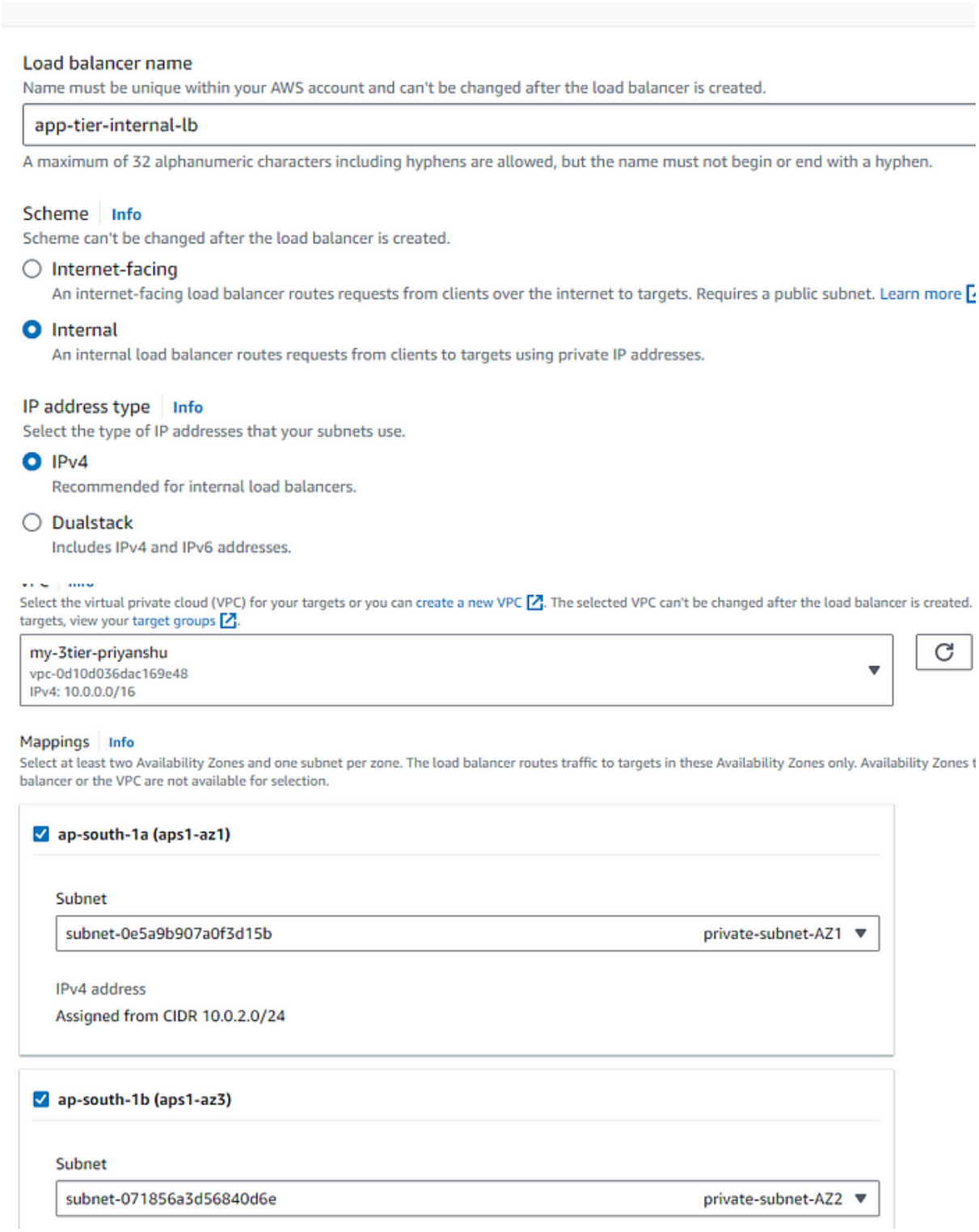
Up to 1024 characters allowed.



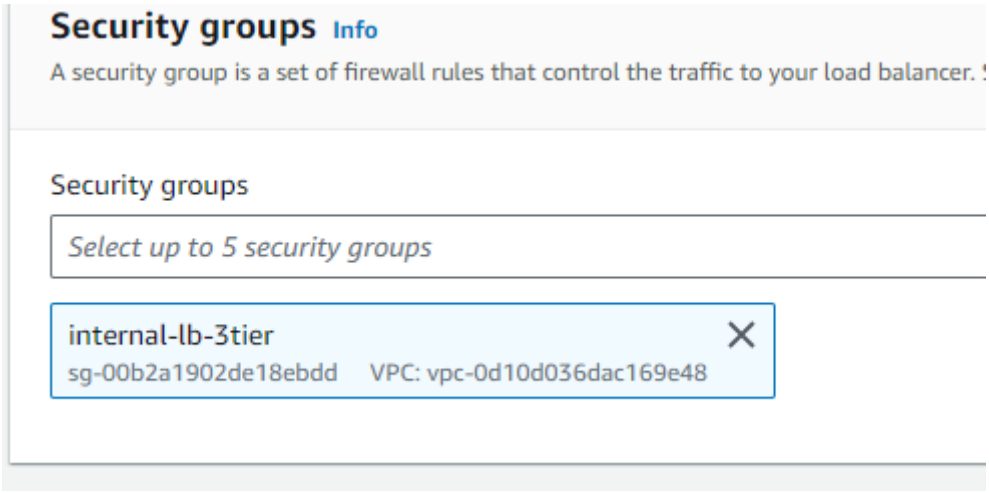
STEP 7: Now create an Load Balancer and attach this Target group to this LB.

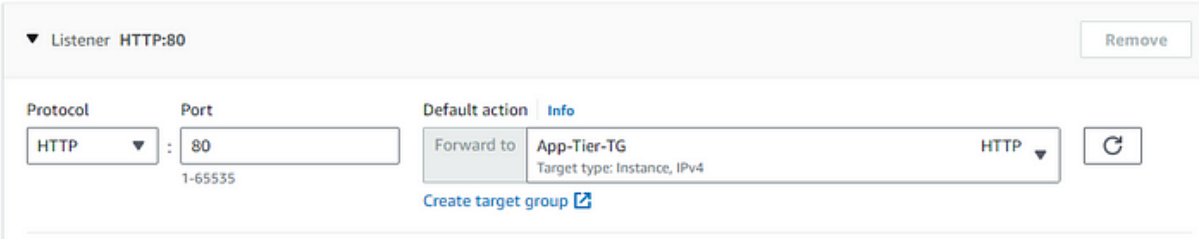


STEP 8: Select internal as its not an internet facing load balancer.Use the private subnet as its internal.

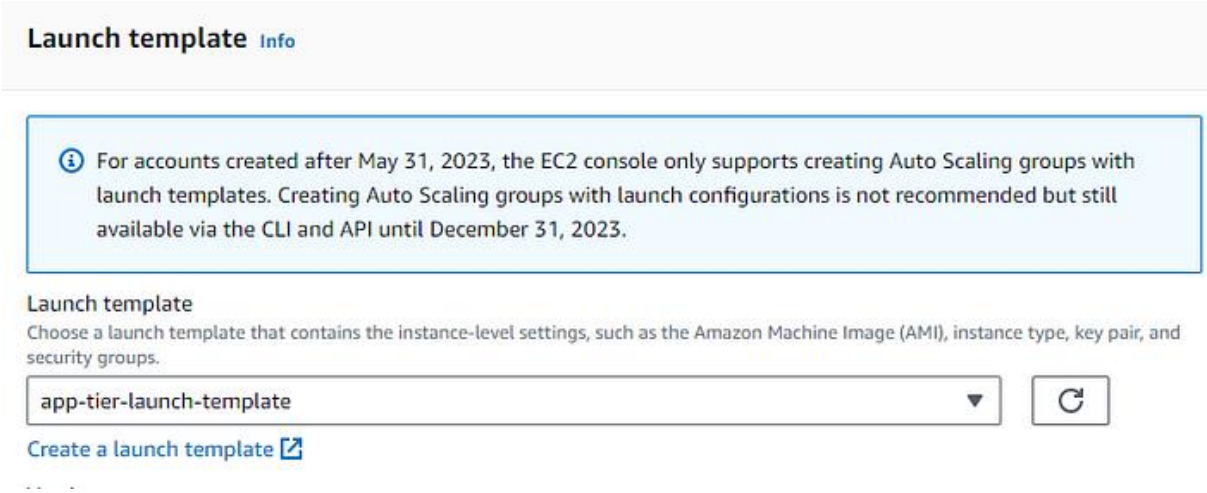


STEP 9: Attach the internal ALB sg which we created to this ALB.

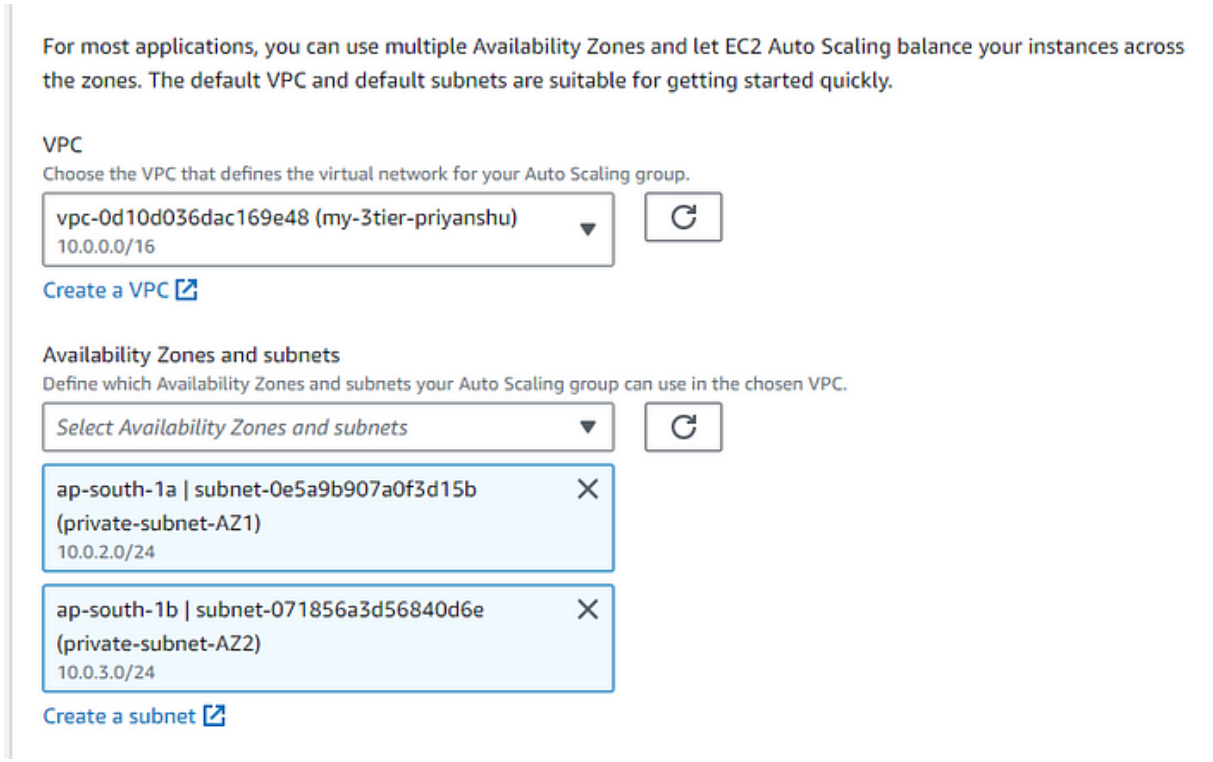




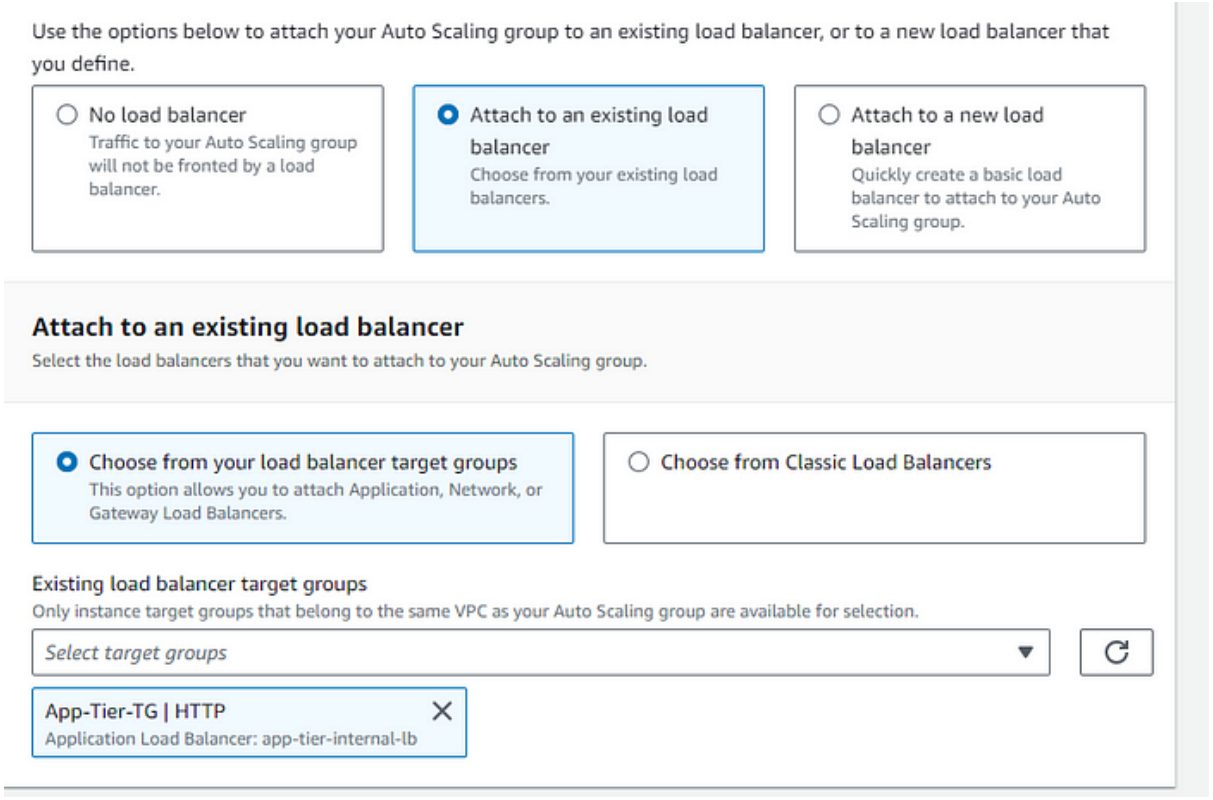
STEP 10: Creating Launch template for auto scaling for the app tier.



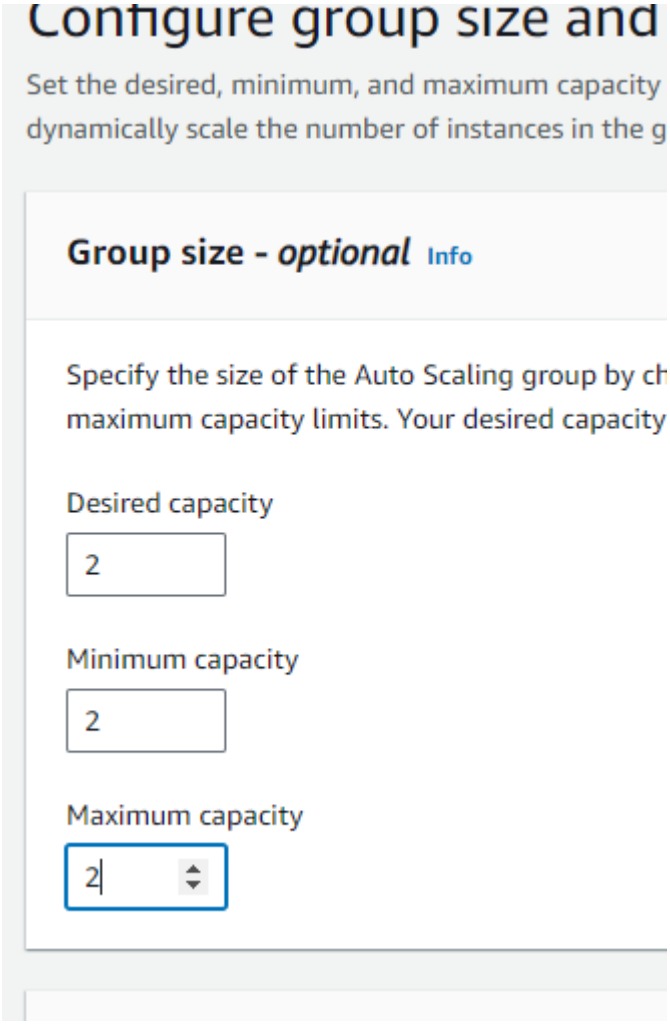
STEP 11: Select Subnet for the app tier private.



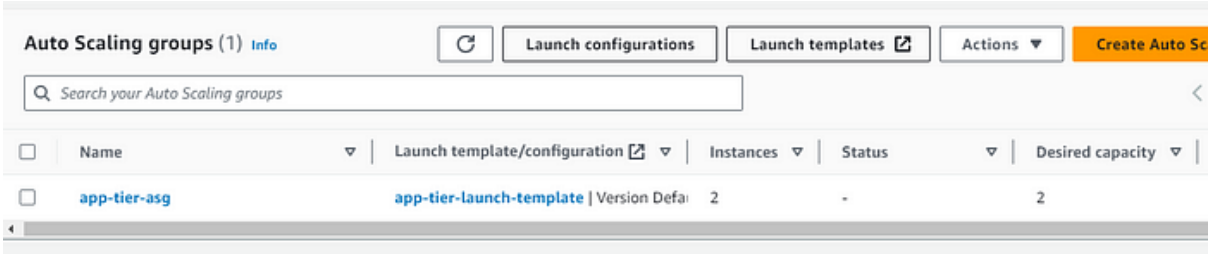
STEP 12: Select the existing Load Balancer for the intenal routing , and the taraget group we created above.



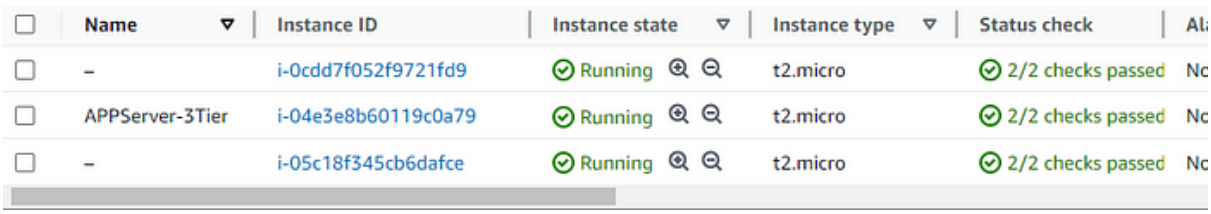
STEP 13: Setup Dynamic scaling , keeping desired and all as 2 for maximum fault tolerance and avalability.



Autoscaling group for the app tier will be created successfully.



So the summary for the app tier was that we did all the setup in the ec2 from which we created an image and then put that image as the ami image for the other instance in the launch template , and this launch template will be used by the autoscaling group.. The load balancer wil contact to this instance.



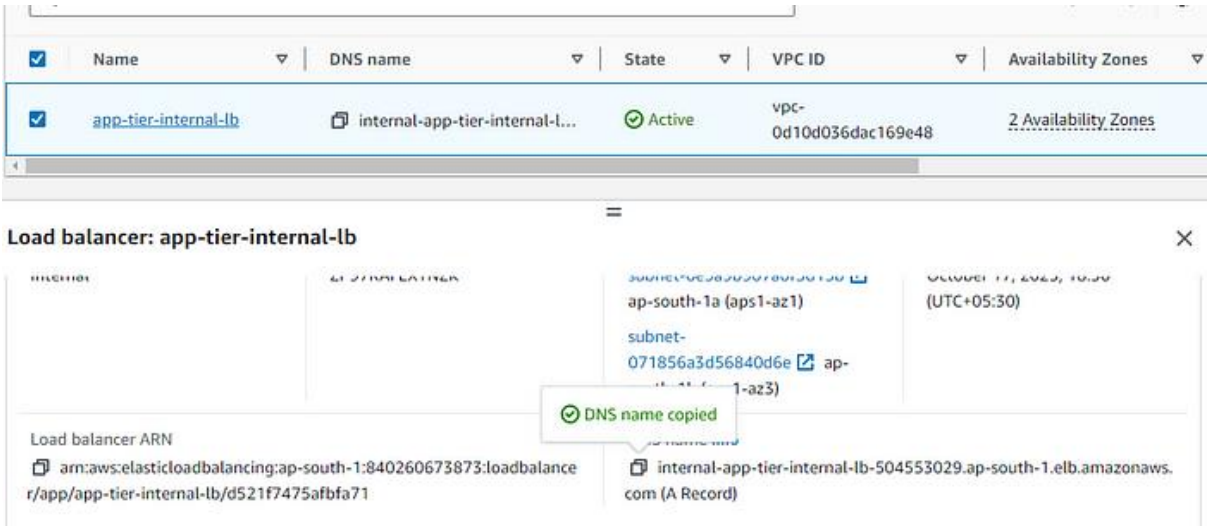
Setting up the web tier

This Tier will be holding our frontend Tier.

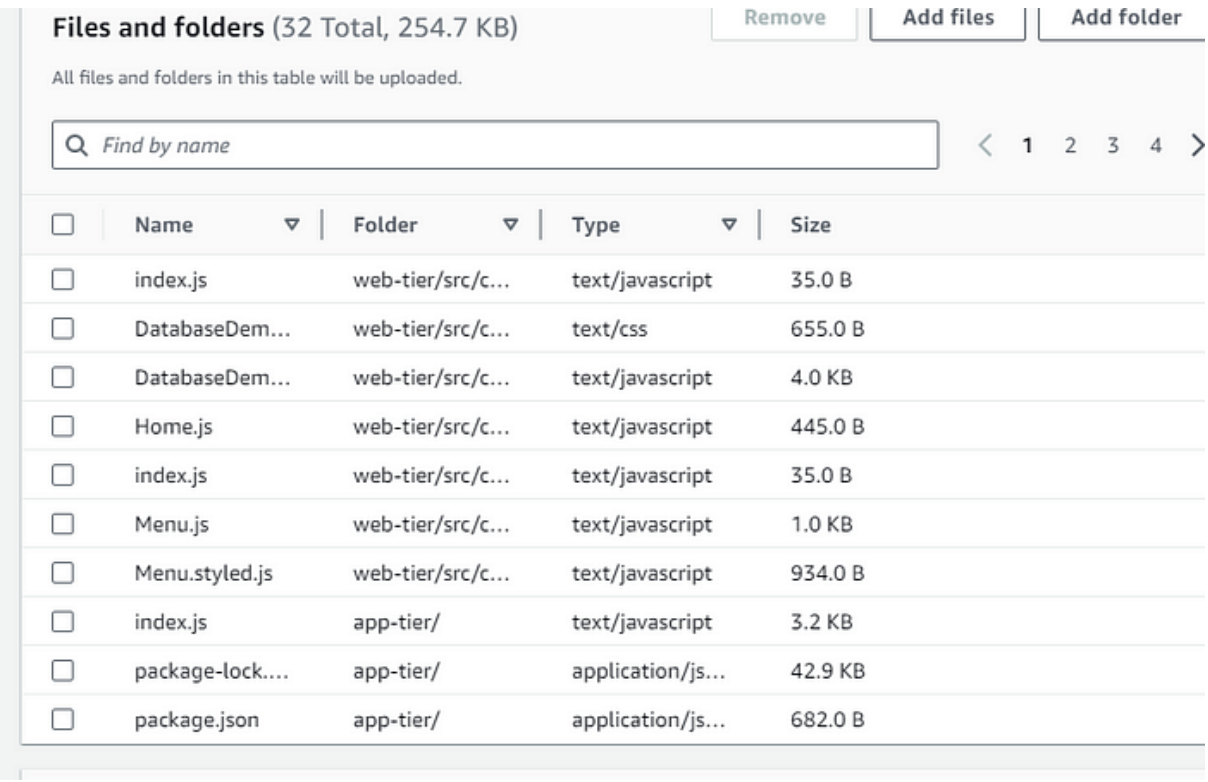
STEP 1: Setup nginx,Edit the nginx conf file, add the proxy pass as the endpoint for the ALB.

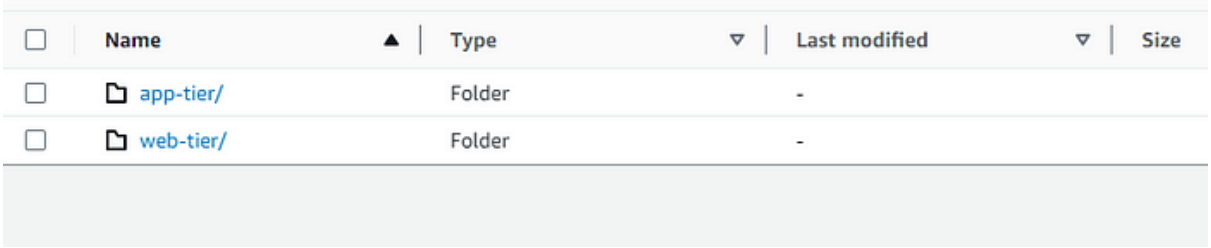
```
    default_type text/html;
    return 200 "<!DOCTYPE html><p>Web Tier Health Check</p>\n";
    }

    #react app and front end files
    location / {
    root    /home/ec2-user/web-tier/build;
    index index.html index.htm
    try_files $uri /index.html;
    }

    #proxy for internal lb
    location /api/{
            proxy_pass http://[REPLACE-WITH-INTERNAL-LB-DNS]:80/;
    }
```

Copy the DNS name for the Internal LB.

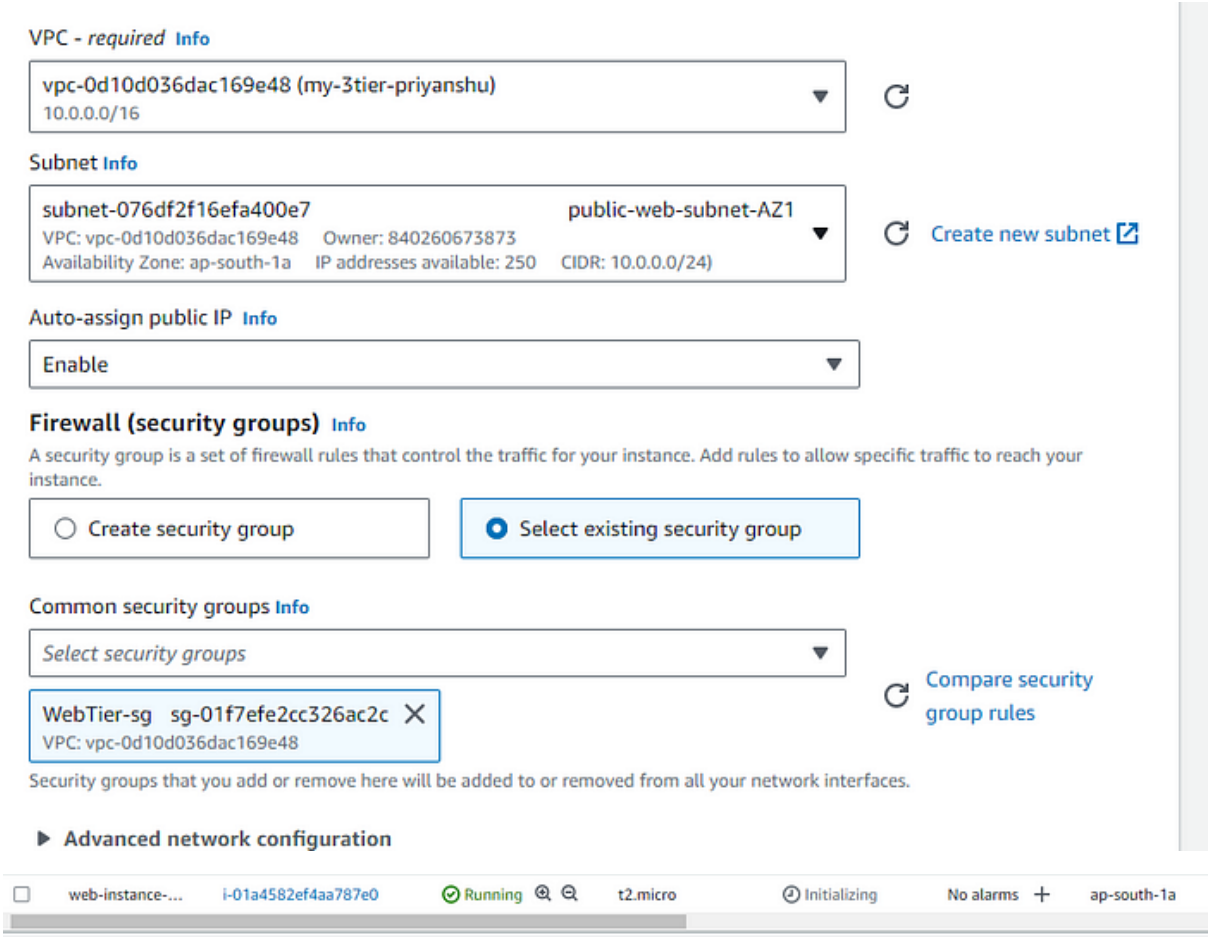


STEP 2: Edit these files with the endpoint and database connections, and upload this to the S3 bucket under the web-tier folder.

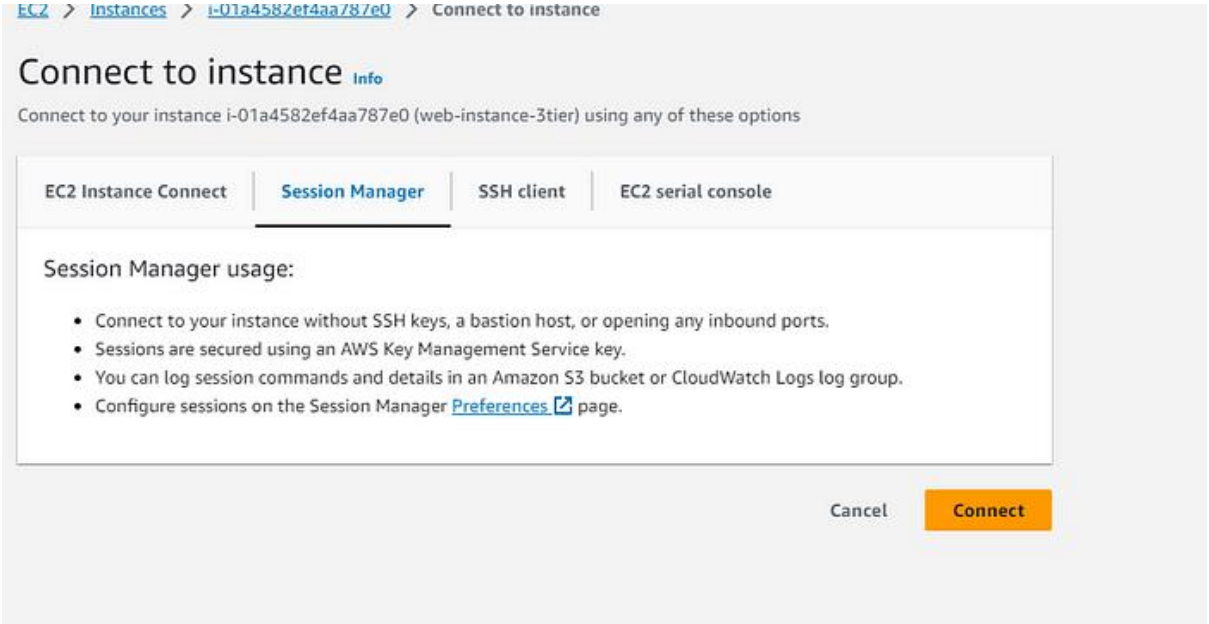




STEP 3: Now setting up instances for the web tier, all the rest configuration is same as for the app tier just change the seurity groups.



STEP 4: Connect it using the session manager.



STEP 5: Setup the instance web tier application same as we did it in the app tier.

```
Creating an optimized production build...
One of your dependencies, babel-preset-react-app, is importing the
"@babel/plugin-proposal-private-property-in-object" package without
declaring it in its dependencies. This is currently working because
"@babel/plugin-proposal-private-property-in-object" is already in your
node_modules folder for unrelated reasons, but it may break at any time.

babel-preset-react-app is part of the create-react-app project, which
is not maintianed anymore. It is thus unlikely that this bug will
ever be fixed. Add "@babel/plugin-proposal-private-property-in-object" to
your devDependencies to work around this error. This will make this message
go away.

Compiled successfully.

File sizes after gzip:

  77.42 kB  build/static/js/main.19a32700.js
  1.79 kB   build/static/js/787.1f63e066.chunk.js
  493 B     build/static/css/main.b20b6ac4.css

The project was built assuming it is hosted at ./.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.

Find out more about deployment here:

  https://cra.link/deployment
```

STEP 6: Install nginx.

```
[root@ip-10-0-0-245 web-tier]# yum install nginx -y
Last metadata expiration check: 0:07:17 ago on Tue Oct 17 13:32:49 2023.
Dependencies resolved.
================================================================================
 Package                         Architecture            Version
================================================================================
Installing:
 nginx                           x86_64                  1:1.24.0-1.amzn2023.0.1
Installing dependencies:
 generic-logos-httpd             noarch                  18.0.0-12.amzn2023.0.3
 gperftools-libs                 x86_64                  2.9.1-1.amzn2023.0.2
 libunwind                       x86_64                  1.4.0-5.amzn2023.0.2
 nginx-core                      x86_64                  1:1.24.0-1.amzn2023.0.1
 nginx-filesystem                noarch                  1:1.24.0-1.amzn2023.0.1
```

replacing the ngnix.conf file from the edit one which we uploaded in the s3 in the /etc/nginx.

```
[root@ip-10-0-0-245 web-tier]# aws s3 cp s3://aws-3tier-priyanshu/nginx.conf .
download: s3://aws-3tier-priyanshu/nginx.conf to ./nginx.conf
[root@ip-10-0-0-245 web-tier]# cd /etc/nginx/
[root@ip-10-0-0-245 nginx]# aws s3 cp s3://aws-3tier-priyanshu/nginx.conf .
download: s3://aws-3tier-priyanshu/nginx.conf to ./nginx.conf
[root@ip-10-0-0-245 nginx]# ls
conf.d        fastcgi.conf.default    koi-utf     mime.types.default  nginx.conf_bkp       uwsgi_params
default.d     fastcgi_params          koi-win     nginx.conf          scgi_params          uwsgi_params.default
fastcgi.conf  fastcgi_params.default  mime.types  nginx.conf.default  scgi_params.default  win-utf
[root@ip-10-0-0-245 nginx]# vi nginx.conf
[root@ip-10-0-0-245 nginx]# ls *.conf
fastcgi.conf  nginx.conf
```

STEP 7: starting nginx server and checking the status.

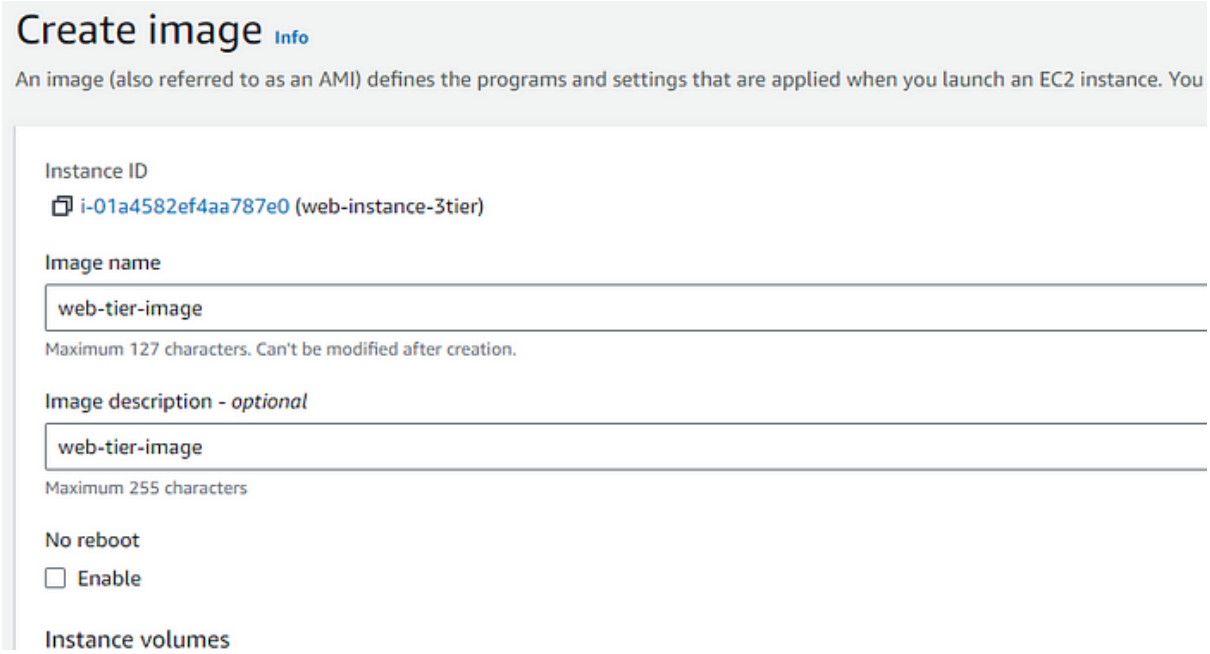
```
[root@ip-10-0-0-245 nginx]# service nginx.service restart
Redirecting to /bin/systemctl restart nginx.service
[root@ip-10-0-0-245 nginx]# systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; preset:
     Active: active (running) since Tue 2023-10-17 13:48:51 UTC; 7s ago
    Process: 26914 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, st
    Process: 26916 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCC
    Process: 26917 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 26918 (nginx)
      Tasks: 2 (limit: 1114)
     Memory: 2.2M
        CPU: 58ms
     CGroup: /system.slice/nginx.service
             ├─26918 "nginx: master process /usr/sbin/nginx"
             └─26919 "nginx: worker process"

Oct 17 13:48:51 ip-10-0-0-245.ap-south-1.compute.internal systemd[1]: Starting
Oct 17 13:48:51 ip-10-0-0-245.ap-south-1.compute.internal nginx[26916]: nginx:
Oct 17 13:48:51 ip-10-0-0-245.ap-south-1.compute.internal nginx[26916]: nginx:
Oct 17 13:48:51 ip-10-0-0-245.ap-south-1.compute.internal systemd[1]: Started
```

```
ip-10-0-0-245 nginx]# chmod -R 775 /home/ec2-user
ip-10-0-0-245 nginx]# sudo chkconfig nginx on
Forwarding request to 'systemctl enable nginx.service'.
d symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /u
systemd/system/nginx.service.
ip-10-0-0-245 nginx]#
```

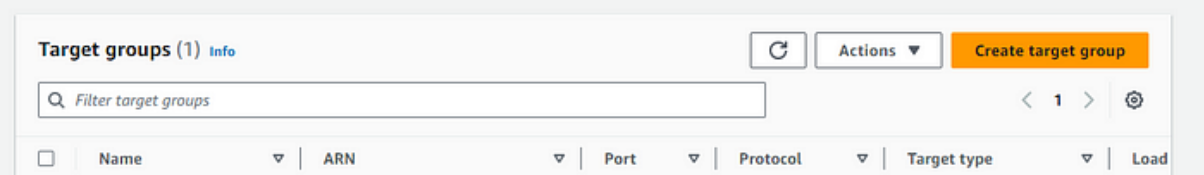
Now setting up the load balancer

In this section we'll be creating web Tier autoscaling groups and Load Balancer configurations.

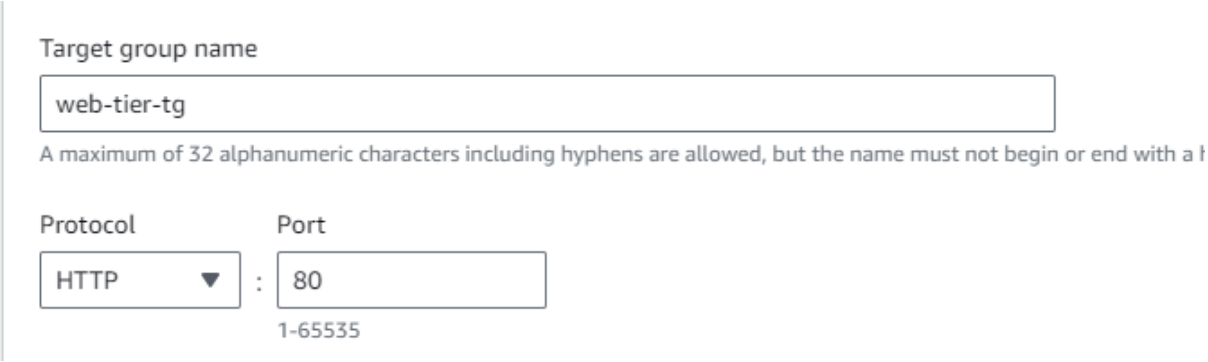
STEP 1: Create ami image for the web instance image as as we did in the app tier.

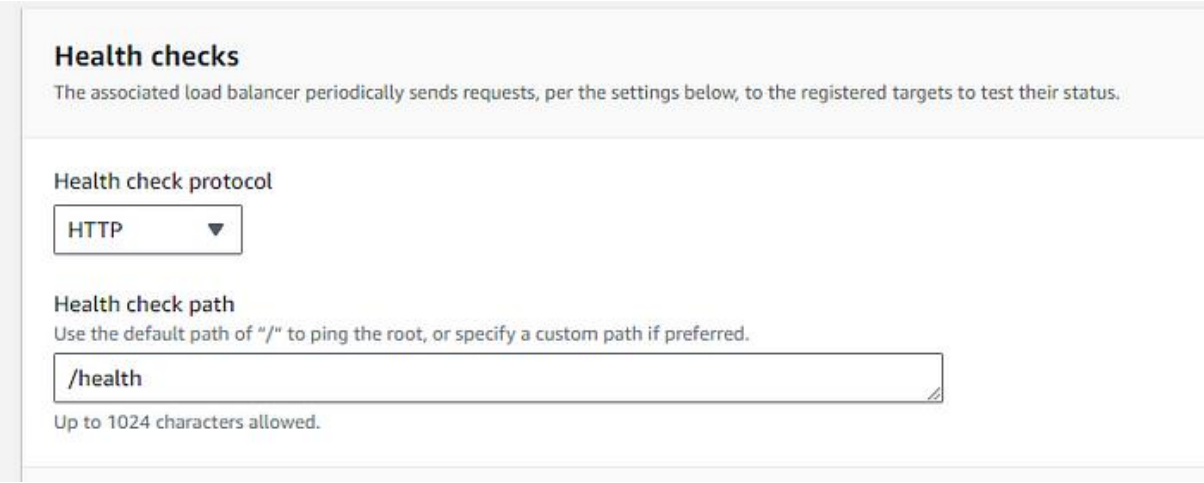


STEP 2: After successfullt creating the image,Create Target group for this load balancer.

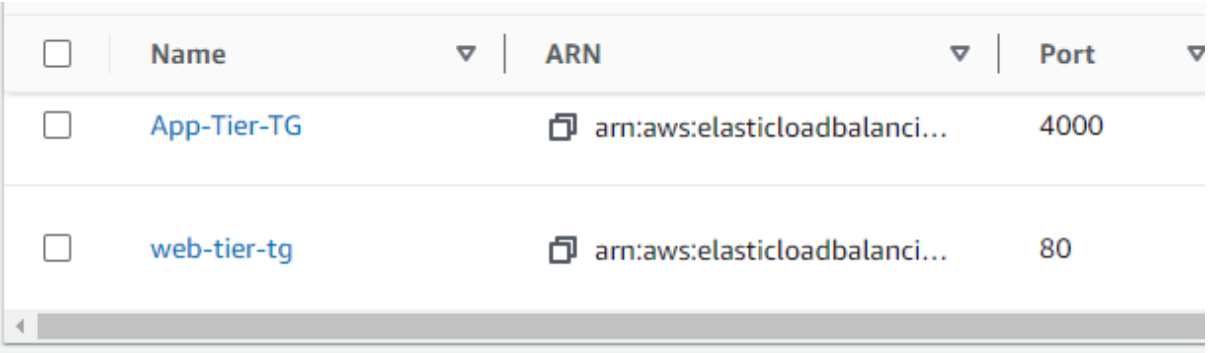


STEP 3: As its an web tier , this time the port is 80.

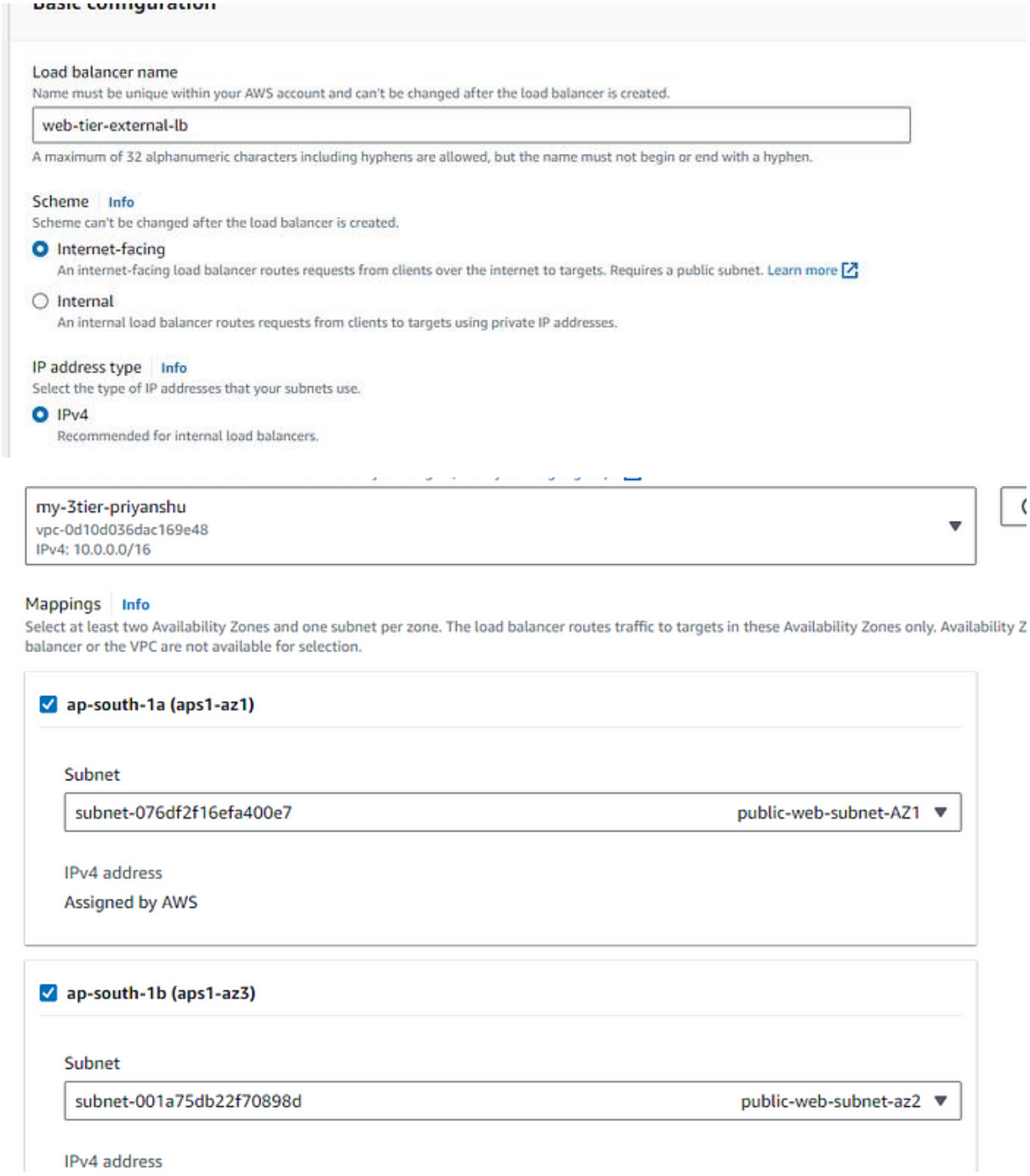




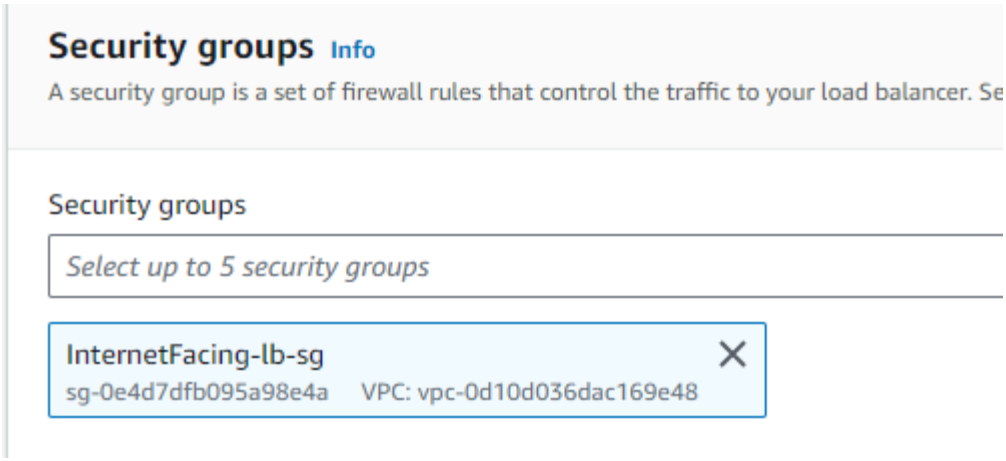
Now we have two target group one for app tier and other for web tier

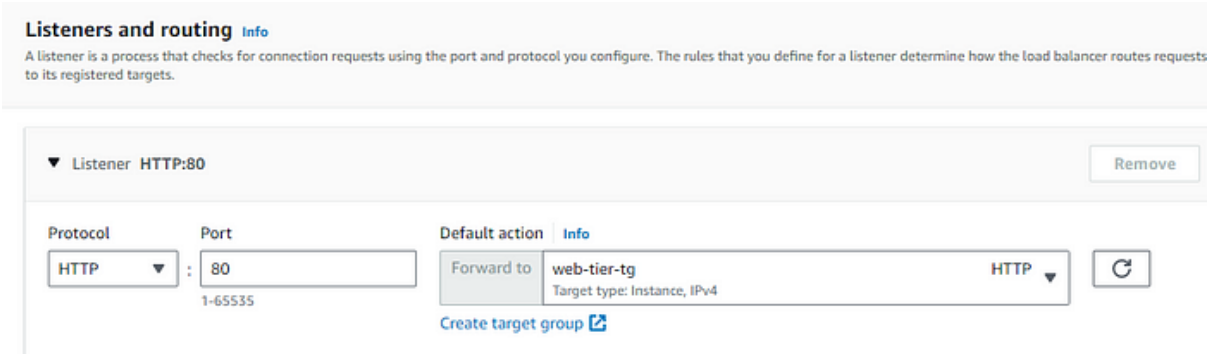


STEP 4: As it is internet facing load balancer enable intenet facing and select the public subnet for both the AZ.

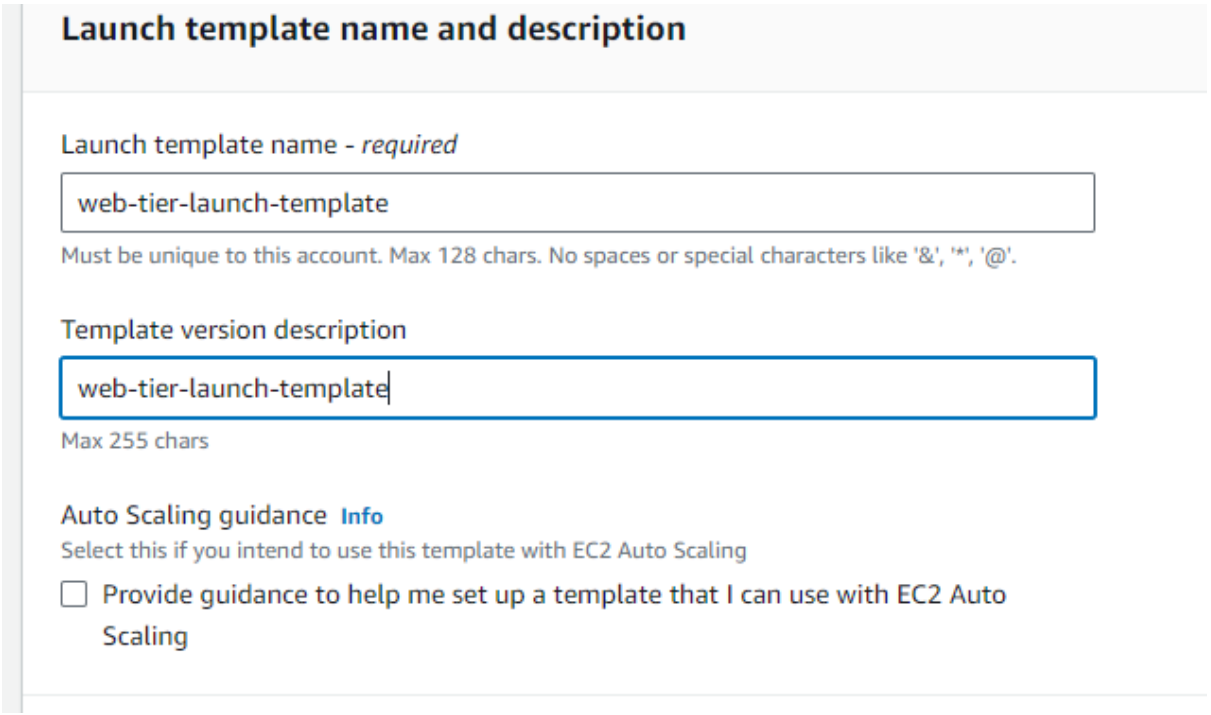


STEP 5: Select the public facing Load Balancer security group.

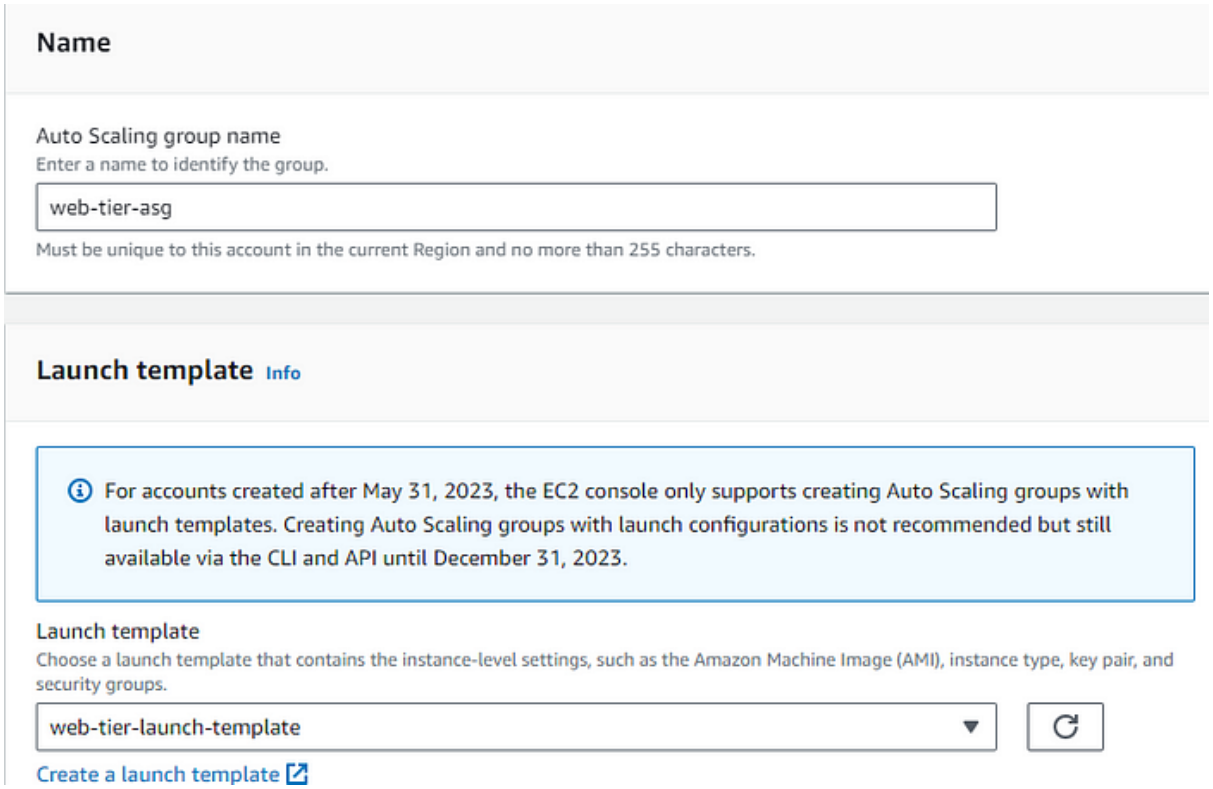




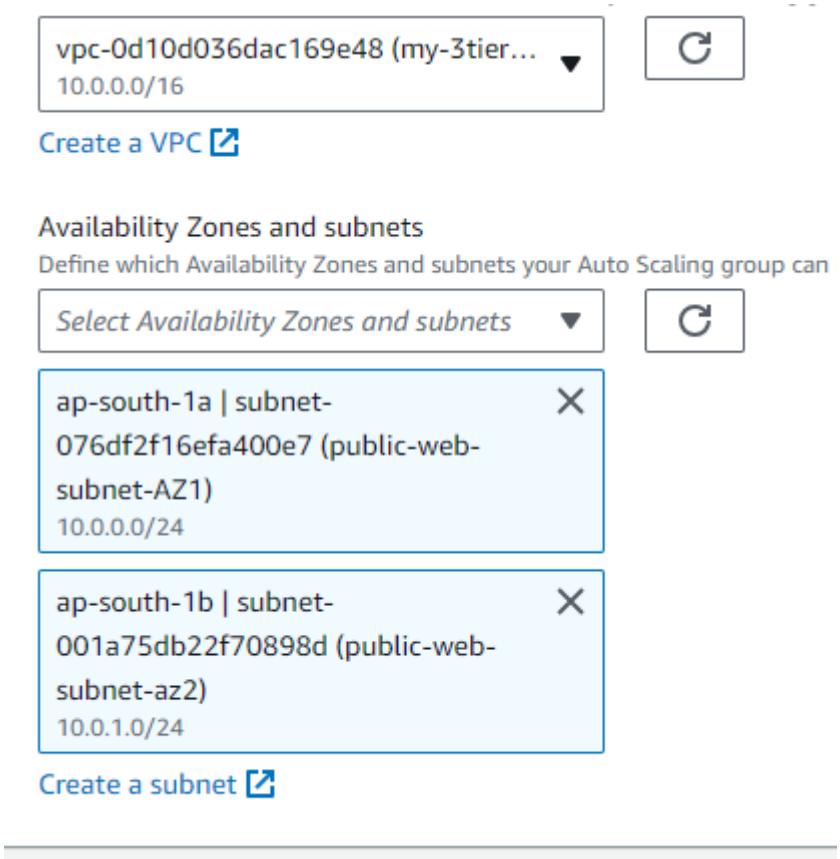
Now create the launch template similarlly to the app tier one jst change the images and the security group related to the web tier.



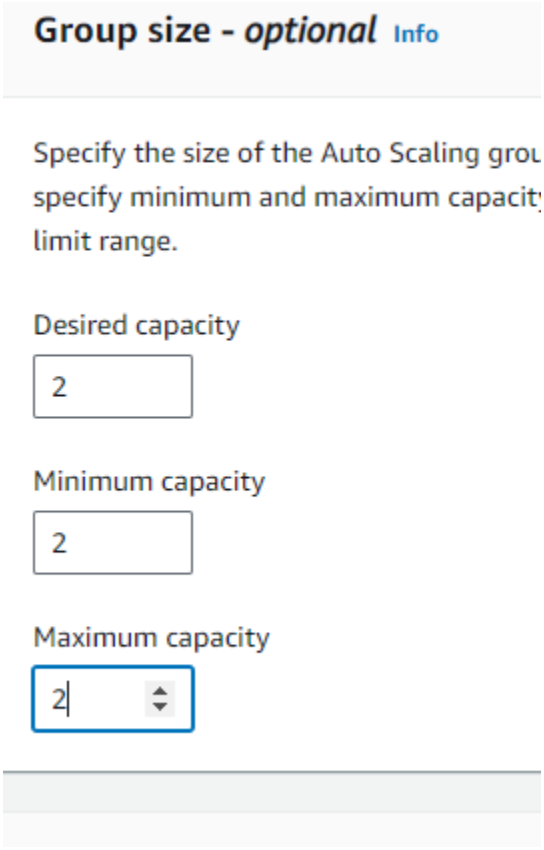
STEP 6: Creating auto sclaing group for web tier.



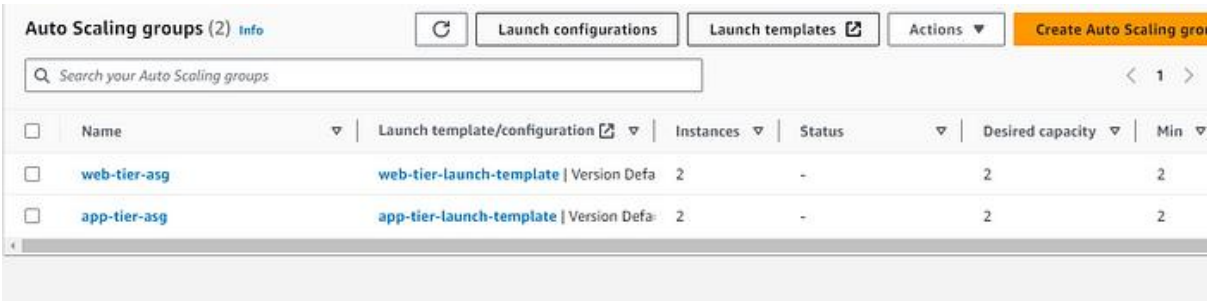
As its an web tier and public hence the Subnet will be Public for both AZ



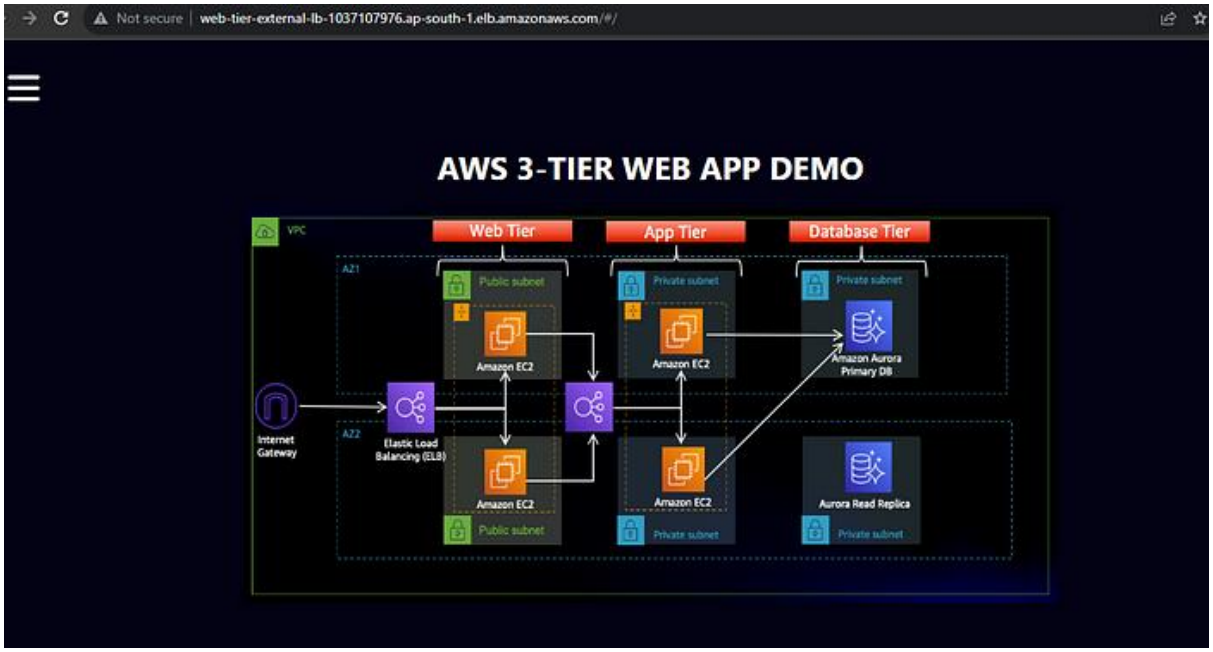
STEP 7: Add the dynamic scaling policy for this web tier auto scaling group.



Launched the web tier load balancer with the 2 capacity.



After the complete setup of both the app and web tier, copy the dns of the web tier load balancer and hit it it will display this page.



The database demo can also be done here, and the amount and description can be directly added to the database from the UI.