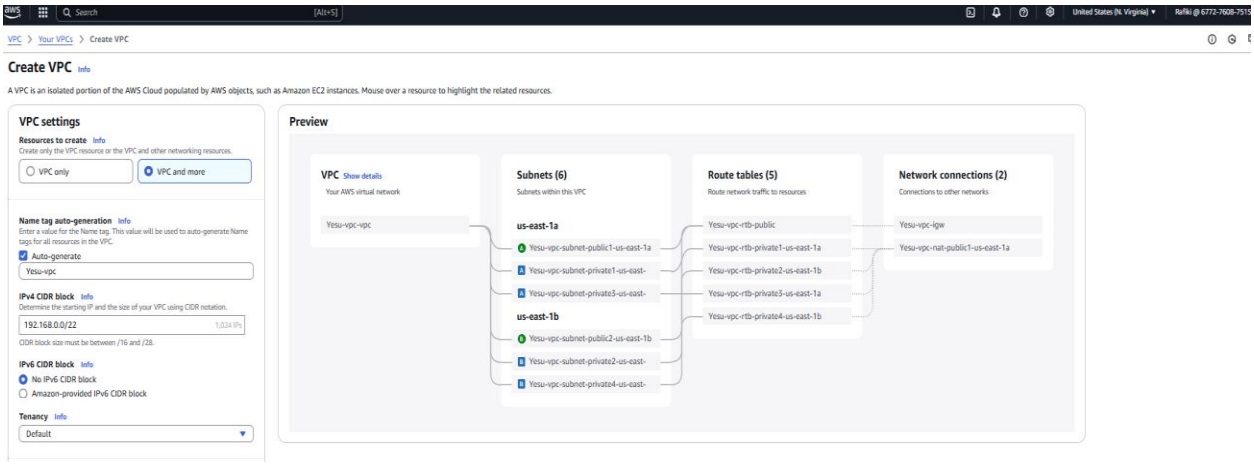
AWS 3 TIER PROJEC T

Link to folk project

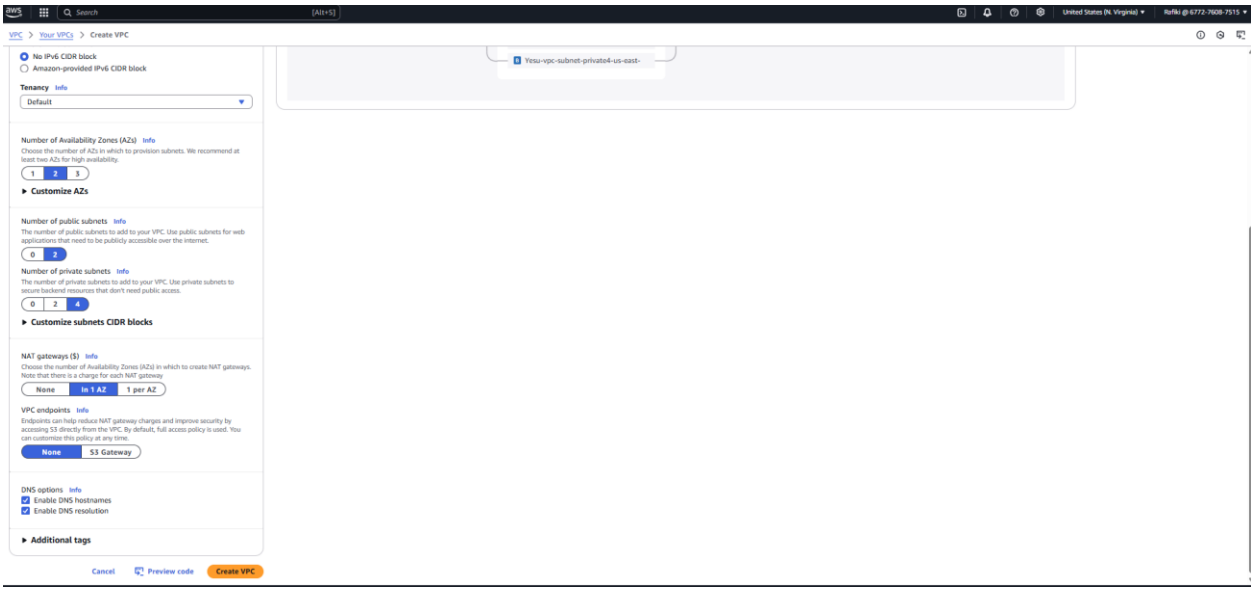
https://github.com/kibush/3TierAWSArchitectureApp/tree/main/application-code

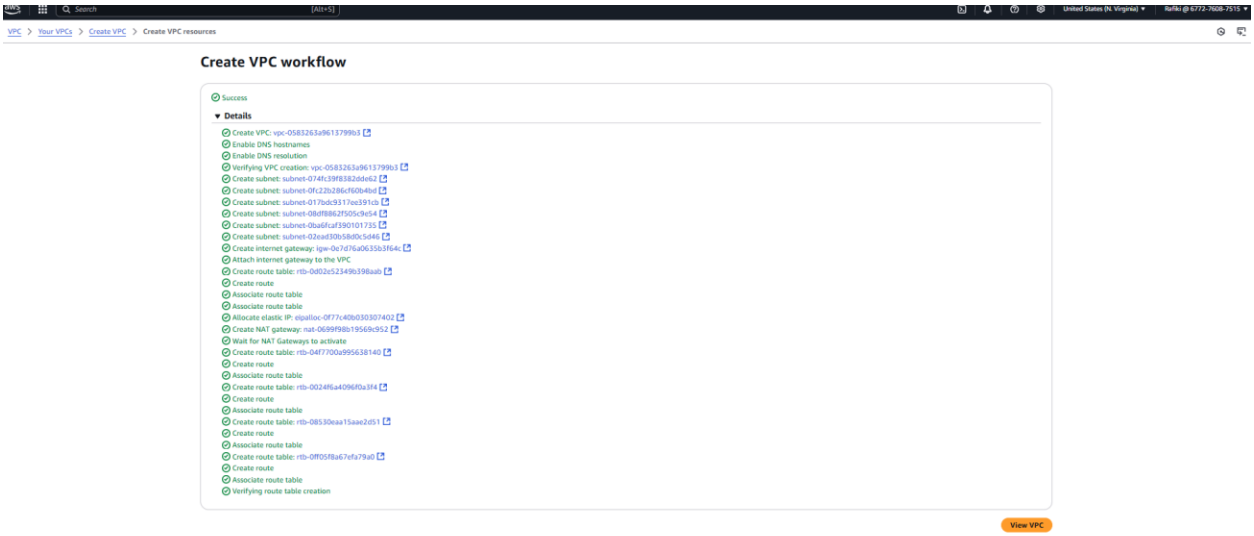
Real time snap shots

3/10/25

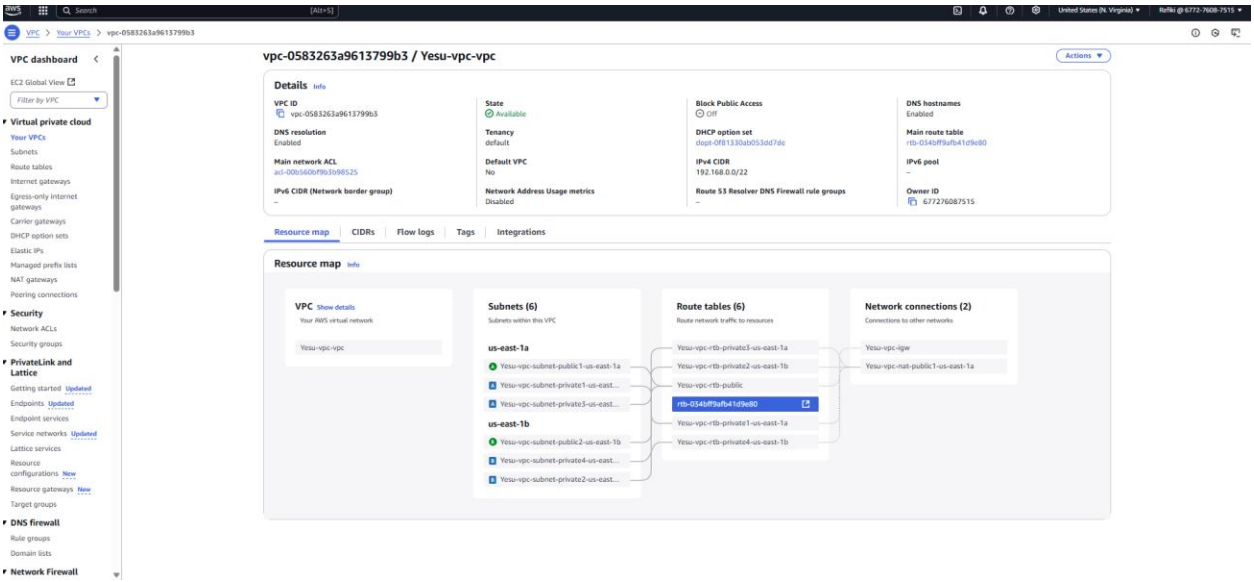


-





View VPC

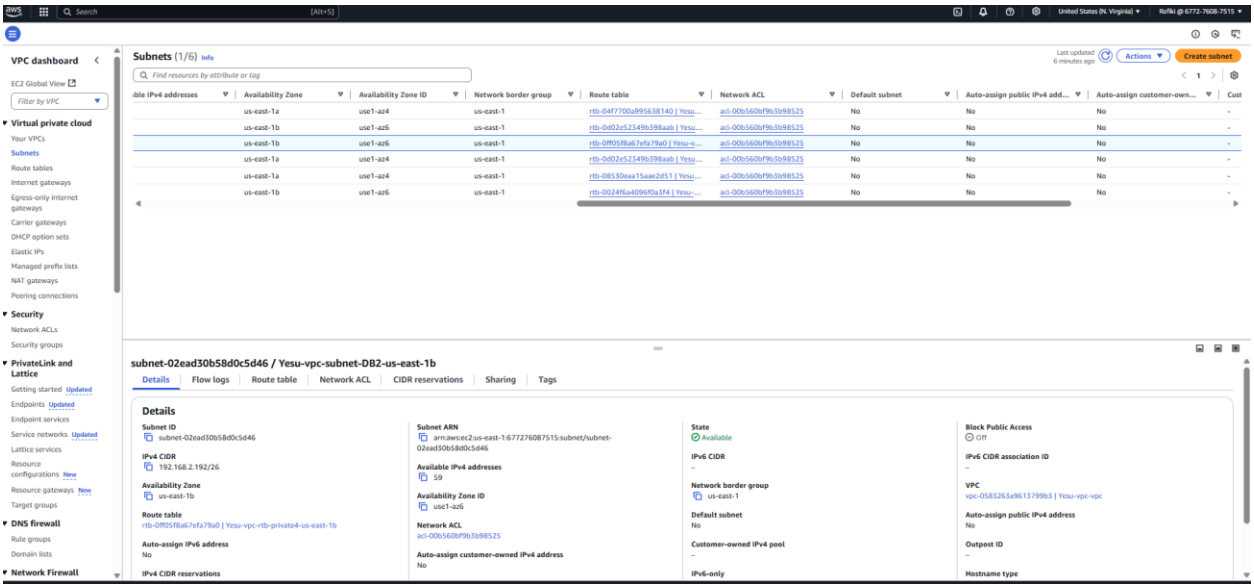


Customize Subnets

Here we are going to edit the names of the subnets to fit their function.

The public subnets will be used for Web layer because we need web connectivity for them.

The private subnets for App layer and Database layer



-Security groups

Next we are going to create security groups. Total five security groups.

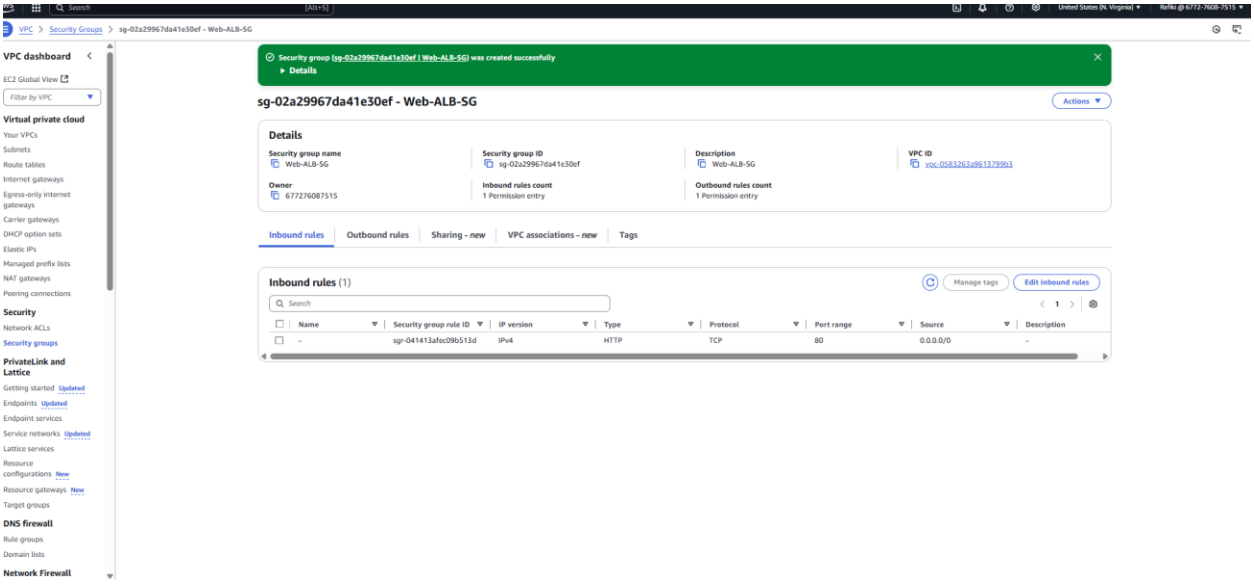
-One for web servers, another for internet facing load balancer(external load balancer) , one for app servers and another one for database servers.

1) Create Security group name -- As Web-ALB-SG , description the same then attache the vpc we created i.e Yesu-vpc

Inbound traffic

http traffic – protocol TCP port range 80 , source type Anywhere –IPV4 (that means open it to anybody) .

Outbound rules remain the same. Create security group



2)

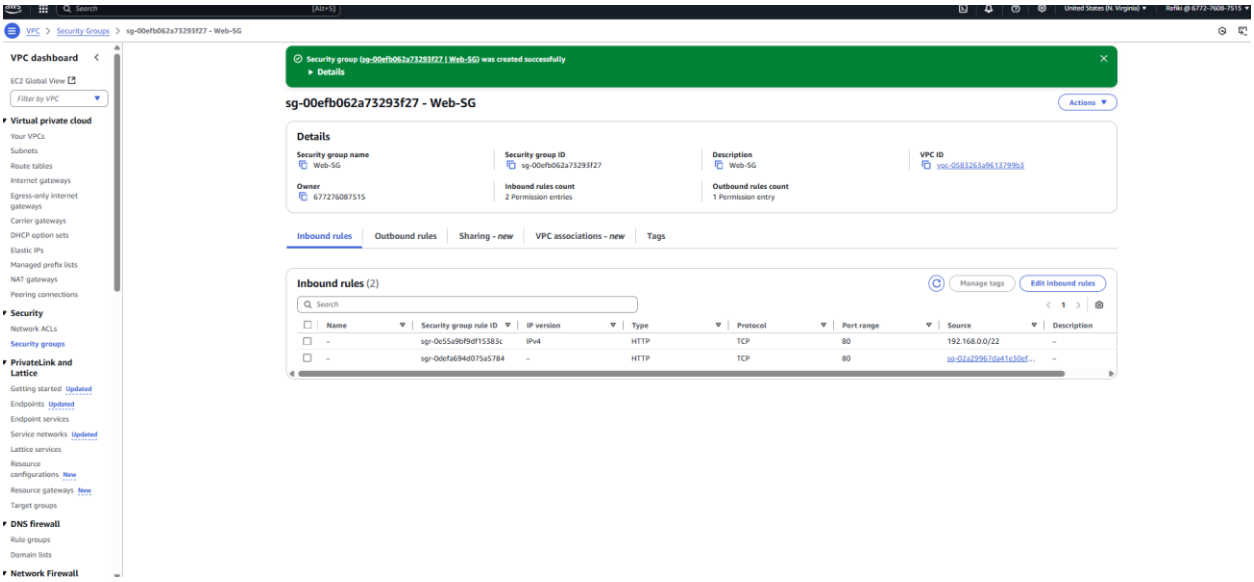
Next security group for the web tier

-Name it Web-SG ,description the same , attached to Yesu-vpc .

-Add inbound rule as HTTP , protocol TCP port 80 . However , source should be custom and attached to Web-ALB-SG security group we created earlier.

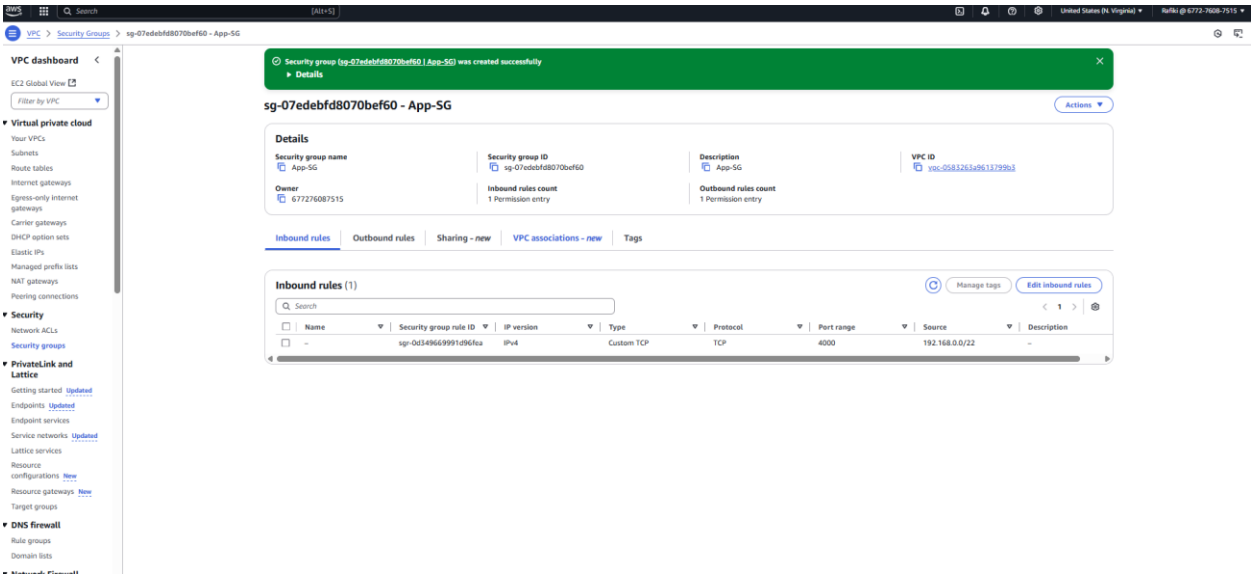
-In addition , we can create another rule with HTTP traffic, protocol TCP , port 80 , custom IP 192.168.0.0/22

IF you recall we used the IP address 192.168.0.0/22 to create vpc CIDR block range.



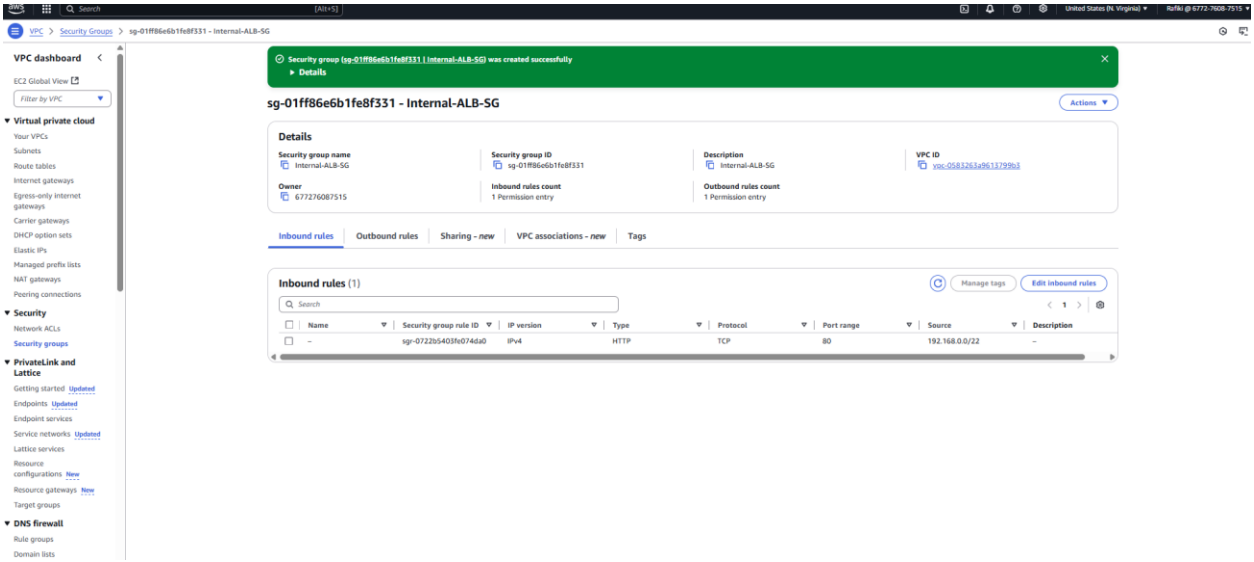
3)Now we need to create security group for private Apps running in the app server. Create security group for the App layer or App tier.

* React- js based application run Custom traffic TCP , protocol TCP on port range 4000 in the inbound rule and attach to vpc CDIR block 192.168.0.0/22.



4) Create security group for internal load balancer . Name it internal-ALB-SG. Next attach our created vpc Yesu-vpc.

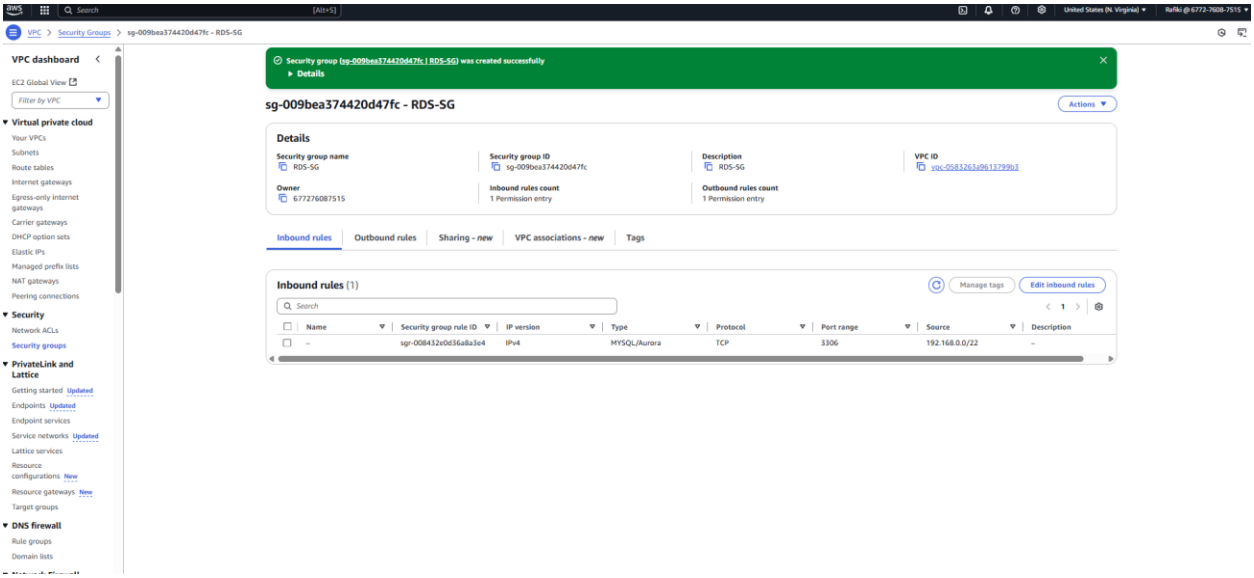
Here we open HTTP traffic, TCP protocol on port 80 , destination custom to 192.168.0.0/22 (vpc CDIR block)



5)

Next create security group for the database . Name it RDS-SG , attached to the vpc I created named Yesu-vpc

Now click on the inbound rule. Here for the traffic is going to be Mysql/aurora . The protocol is TCP and it runs of default port 3306 , source custom is custom network at IP 192.168.0.0/22. Recall this is vpc CIDR block. The resources communicating with the database will be restricted to the private subnets in the App tier.



Note that Web-ALB-SG in internet facing if you examine the inbound rules. We have given the source as anywhere meaning it is public facing network. Anybody from outside can access the application which is running in the web tier.

II

1) The second step is to create the s3 bucket. Reason we need to upload the our application code in the S3 bucket. Why ? To avoid the complexity of uploading the application code directly to our EC2 instances.

The source of our application code may be found in the Github repository using the link provided below.

https://github.com/kibush/3TierAWSArchitectureApp/tree/main/application-code

In my case I cloned the application code on my local machine via git bash. As follows below step by step

Owner@DESKTOP-MATAKO MINGW64 ~/Desktop/AWS/3TierKastro

$ git clone https://github.com/kibush/3TierAWSArchitectureApp.git

Cloning into '3TierAWSArchitectureApp'...

remote: Enumerating objects: 93, done.

remote: Counting objects: 100% (93/93), done.

remote: Compressing objects: 100% (83/83), done.

Receiving objects: 50% (47/93)used 0 (delta 0), pack-reused 0 (from 0)

Receiving objects: 100% (93/93), 369.10 KiB | 14.20 MiB/s, done.

Resolving deltas: 100% (19/19), done.

Owner@DESKTOP-MATAKO MINGW64 ~/Desktop/AWS/3TierKastro

$ ls

3TierAWSArchitectureApp/

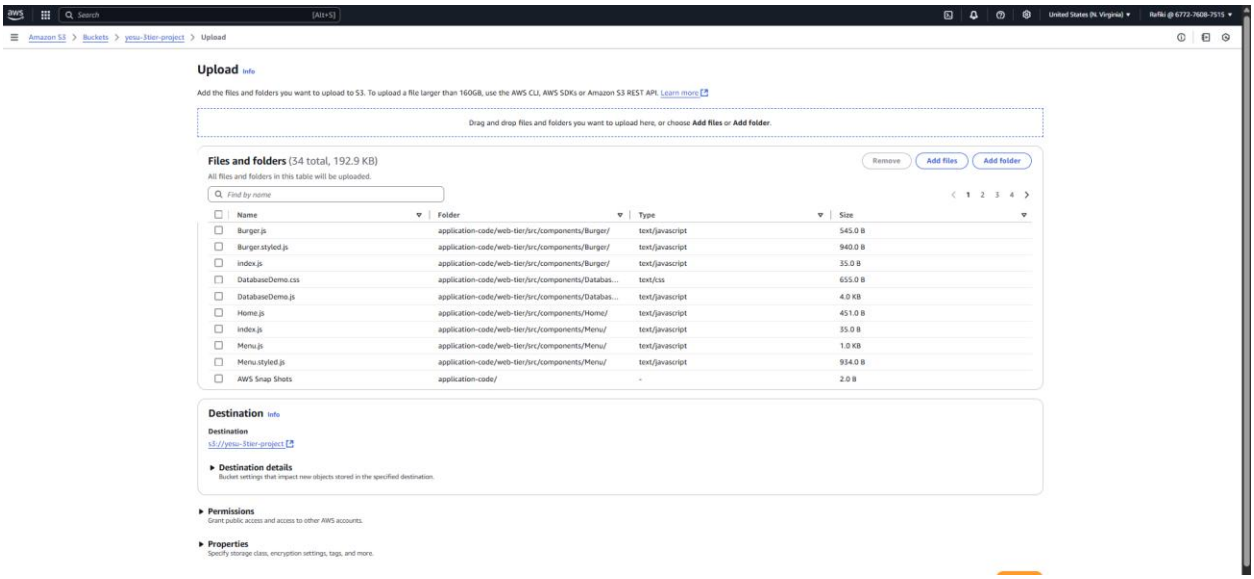
Owner@DESKTOP-MATAKO MINGW64 ~/Desktop/AWS/3TierKastro

2)

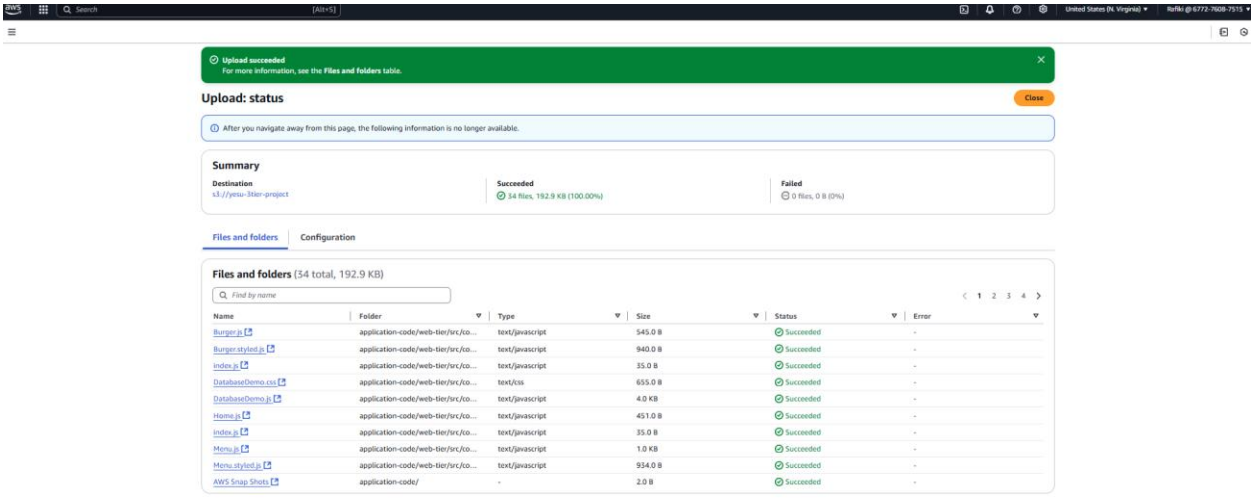
Here I am going upload the entire application folder into the s3 bucket on AWS. Type s3 in the search window to find the s3 service (Scalable storage in the Cloud). Here a bucket with the default settings.

Note it is important to remember the region of your service . In my case it is US East (N. Virginia) US-east-1. We will call out bucket name yesu-3tier-project.

Keep all settings in default mode . Now upload only application code folder. In the bucket you can drag and drop folder directly in the s3 bucket from the local machine.

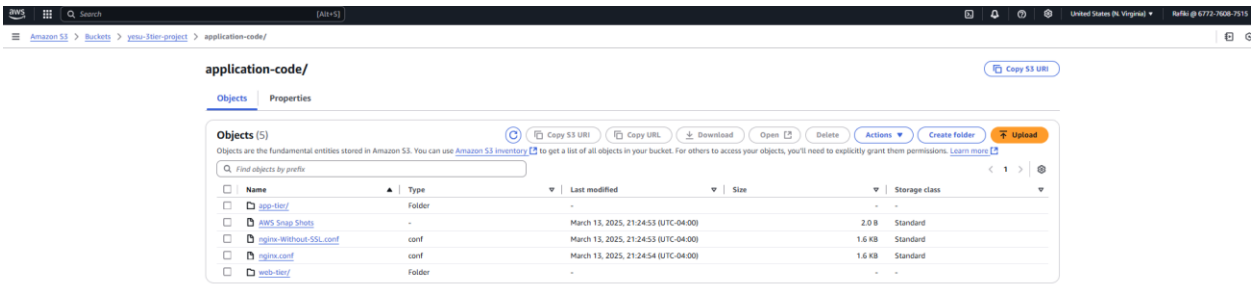


Click the up load file once you drag and drop file.



Here you can view the application code folder uploaded to s3 bucket.

-



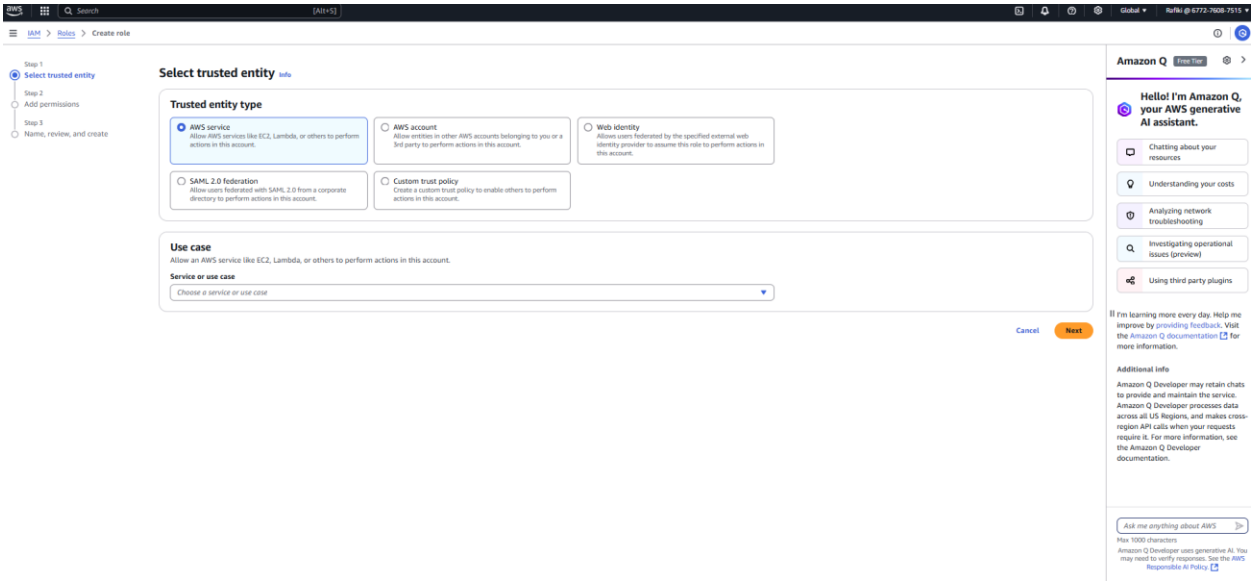
3) Next lets create an IAM role . What is the reason for creating IAM role. Remember we have two EC2 instances in the App tier. In addition they are in the private subnet . Also , the database is also unavailable in the private subnet. It is important to not give these 2 services access to outsiders. So , we are going to close any vulnerability. There may be 2 ways to connect to these two resources i.e. EC2 instances and the RDS. We can use the bastion host which the concept in vpc. That means we create a bastion host in the public subnet and by using the bastion host we will be able to connect to resources in the private subnet. The second way is instead of creating a bastion host an the complexities that come with it we can instead use the AWS service called the systems manager called SSM. That means if we attaché the appropriate role to EC2 instance which are there in the private subnet. Even though the

machines are there in the private subnet we access this virtual machines using the SSM agent. So, in our case we use the IAM role to be able to connect to these private layer or tier.

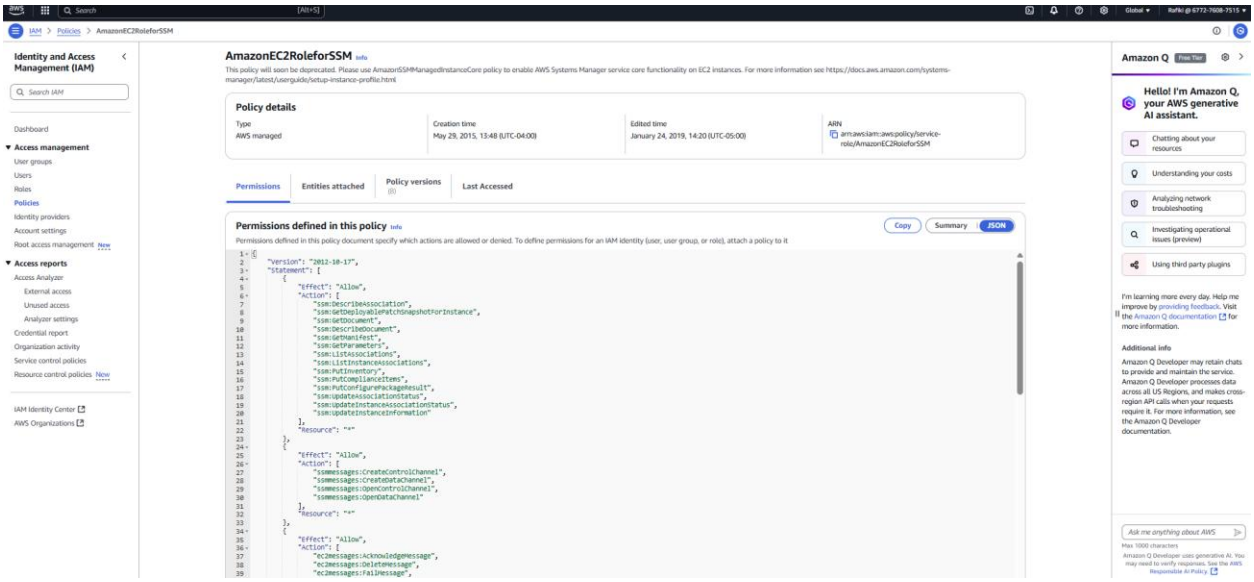
To, define our role we are going to create a policy and attach Amazon EC2 role for SSM.

4)

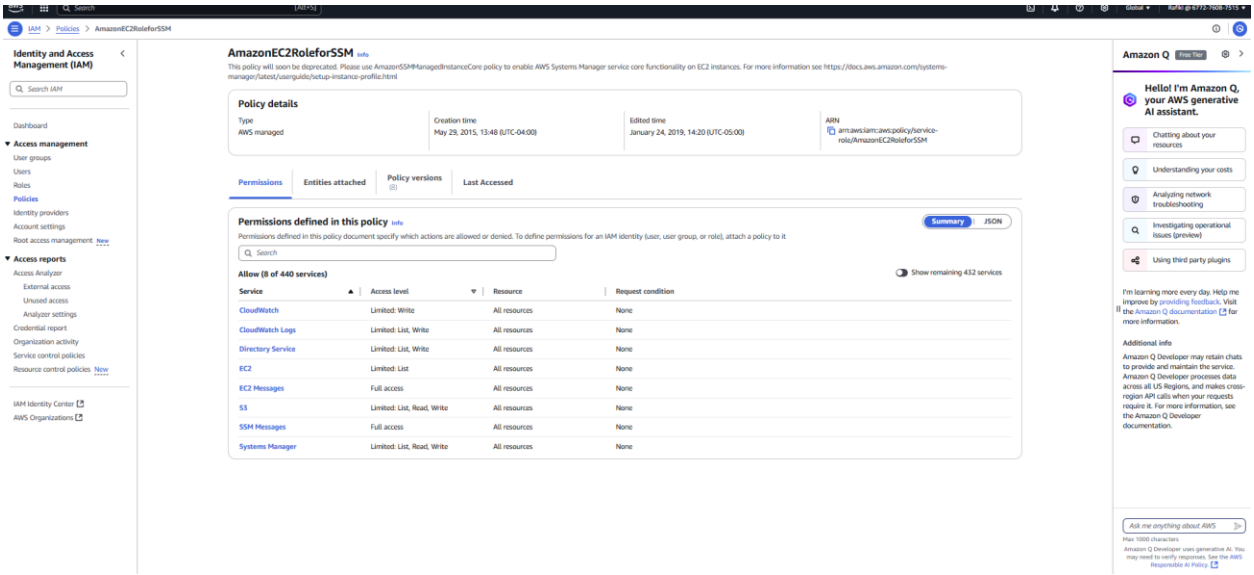
Go to IAM (Identity & Access Management) and click on roles to create one.



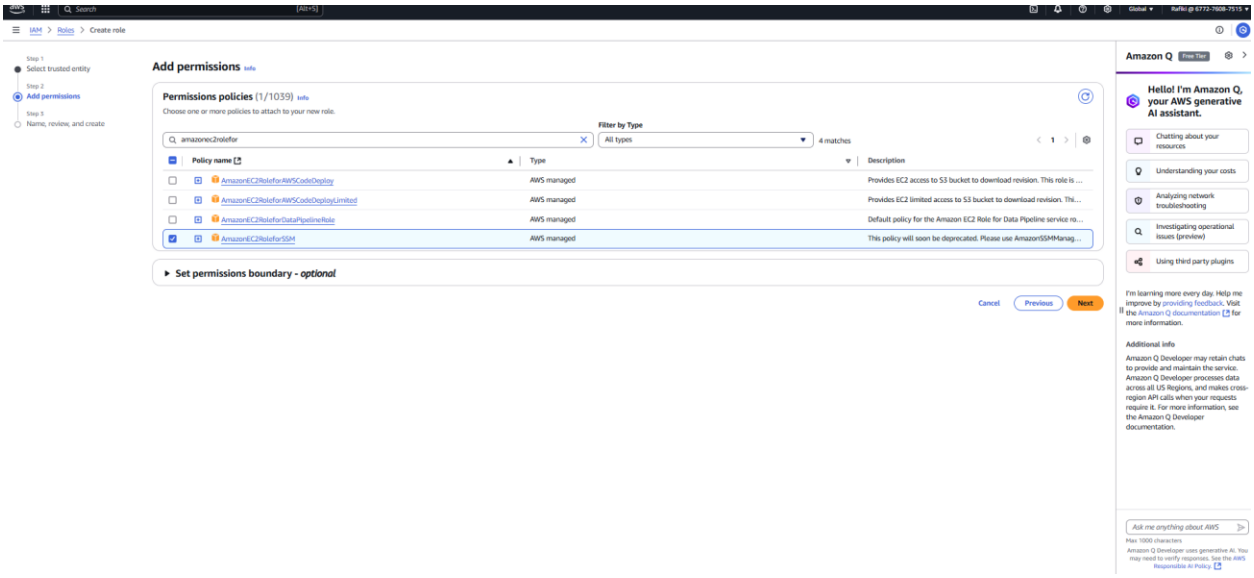
- Initially I select entity as AWS service . At the bottom option use case option I chose EC2 then next . Now which role or policy we need to access , in my case I choose **amazonec2rolefor SSM**.



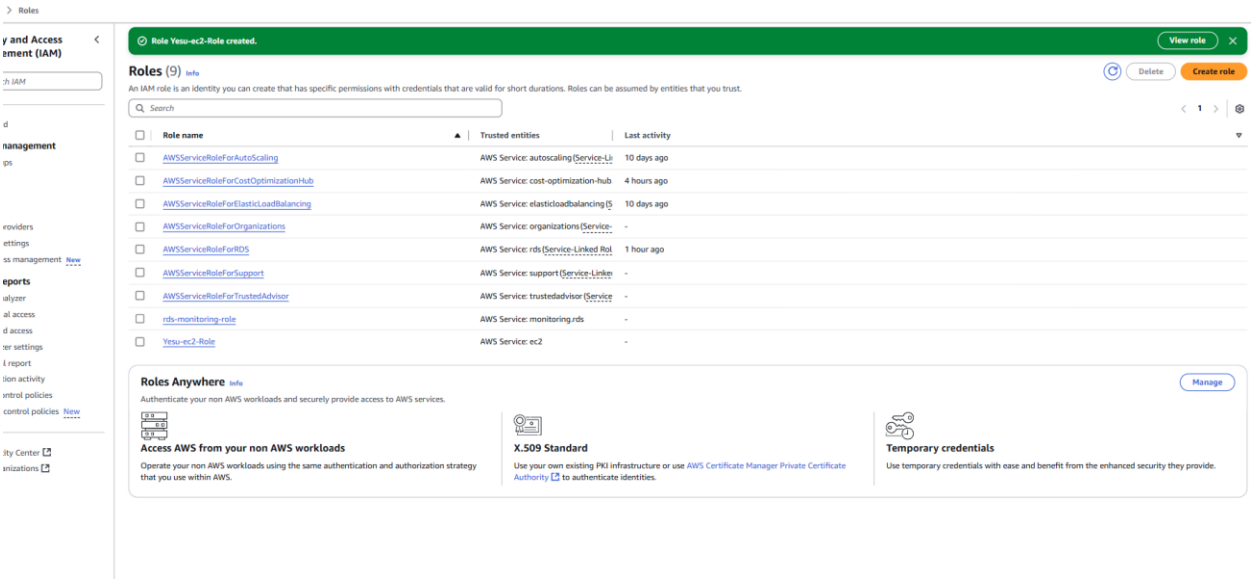
-Most importantly we access to s3 bucket.



-Also attach or add permission by clicking the box on the left to make sure it is attached.

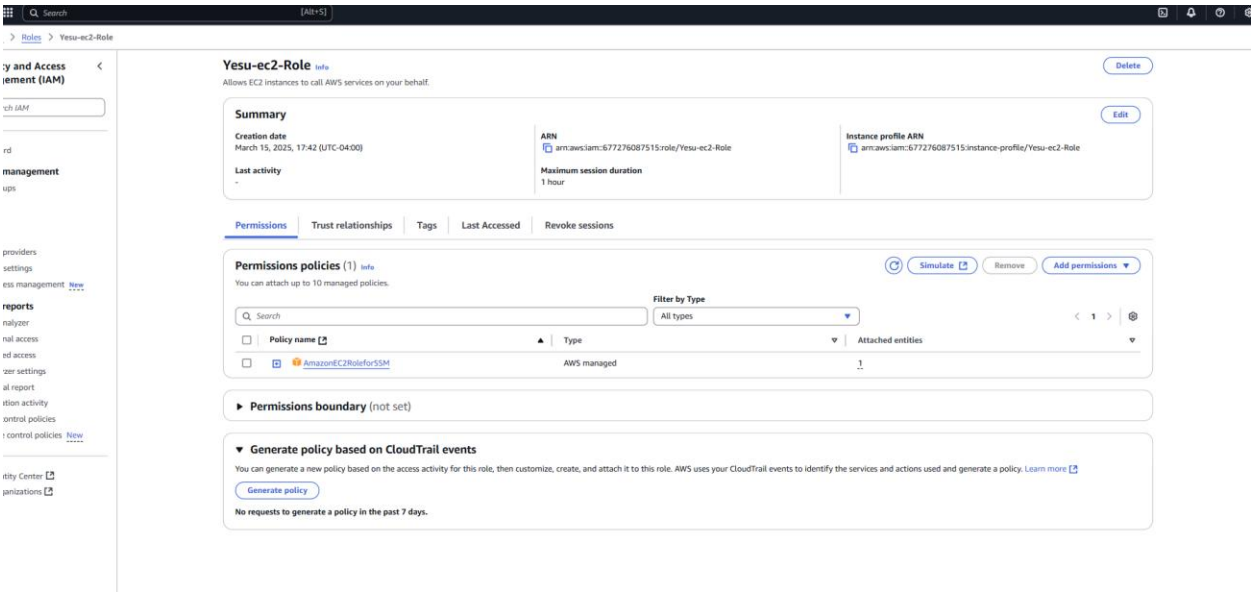


-Click next , give the role name as Yesu-EC2-Role. Scroll down and create role.



View role as shown below

5) Next we have to now create a database. Here we can use a service called RDS in the AWS cloud to create database instances. In the search bar on the AWS cloud type RDS.

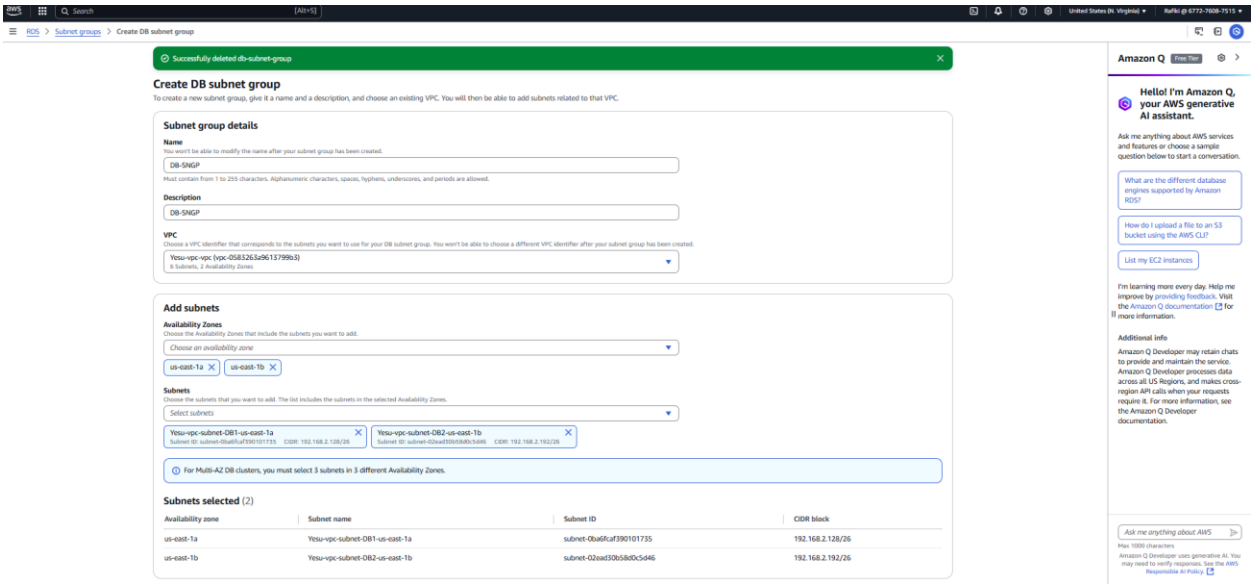


-Note in RDS billing is chargeable unless you are covered under free tier.

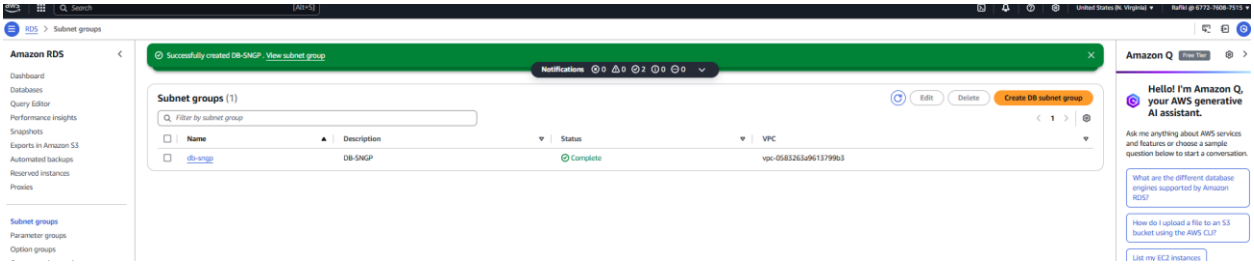
Now it important to create RDS with the specific subnets. Also, within the RDS we do not have permission to select specific subnets. So the other option we have is to use the subnet groups. Here , we will add our custom vpc and DB1 and DB2 subnets we have created then attaché DB-subnet- group to RDS instance.

Create subnet group . Name it DB-SNGP then chose custom vpc. Yesu-vpc we created before.

-Select availability zone available. I will chose us-east-1a and us-east -1b. Next you see option to select subnets .In our care we choose Yesu-vpc-DB1- us-east-1a and Yesu-vpc-DB2- us-east-1b private subnets previously created at the beginning of the project. So , click create .



-As shown below we have created subnet gropus.



-Hence, we should attached subnet group to the database. How ? Click database on the left select tab option. Create databases click standard create and select MySQL or Aurora (MySQL compatible) . The commands are the same. Edition is MySQL community choose an earlier version. Templates chose free tier option. Self managed password or AmazonRDS can create one for you and view on the credential details after database creation.

Next make sure Instance configuration chooses burstable classes as db.t4g.micro. Storage as GP2 allocation 20 GiB as the default allocation.

Connectivity to remain as default.

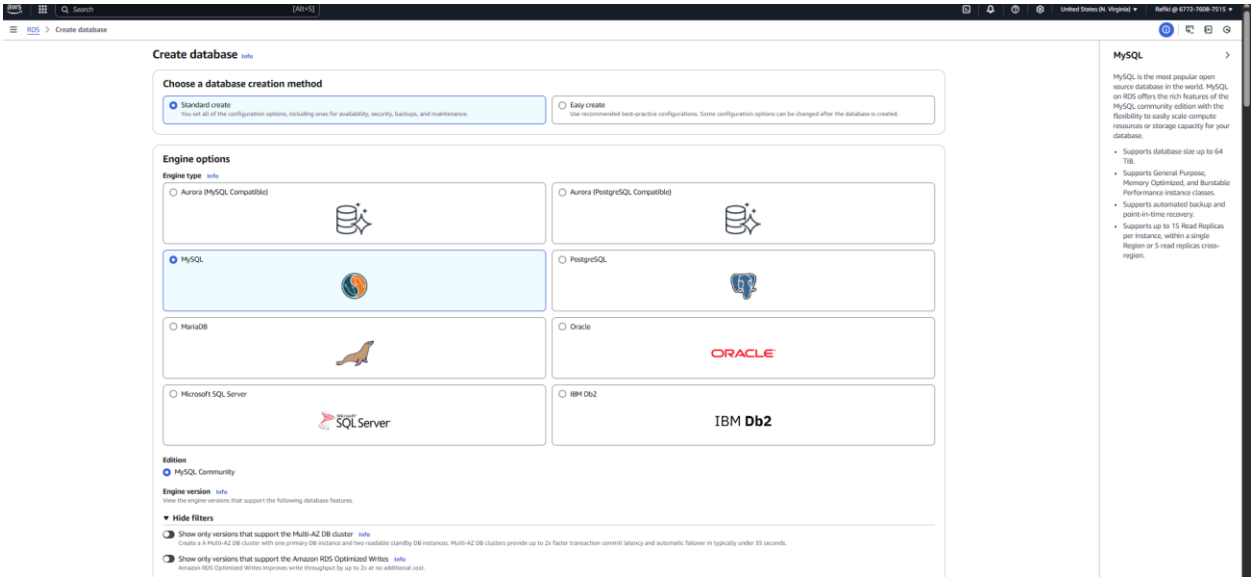
-Note it is important to chose the vpc we created Yesu-vpc at the beginning of the project and the DB subnet group called db-sngp we created. No public access too because have a BD as a private subnet.

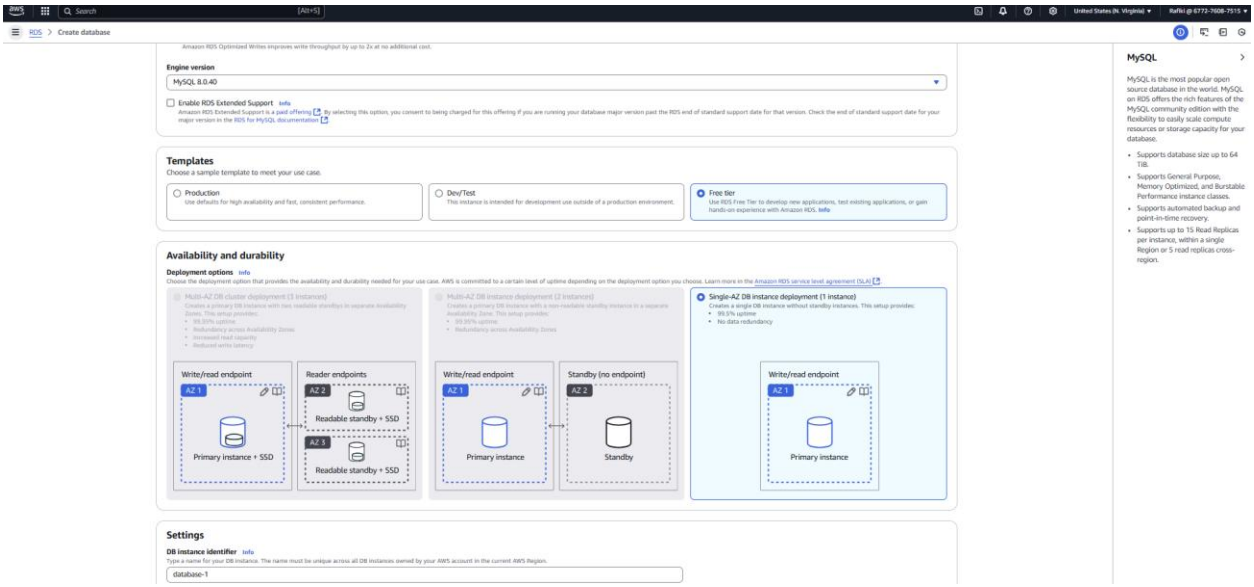
-Also, note we have a own security group we created before. So, need to uncheck default and chose RDS- SG security group we created earlier. Availability zone is not necessary for our project.

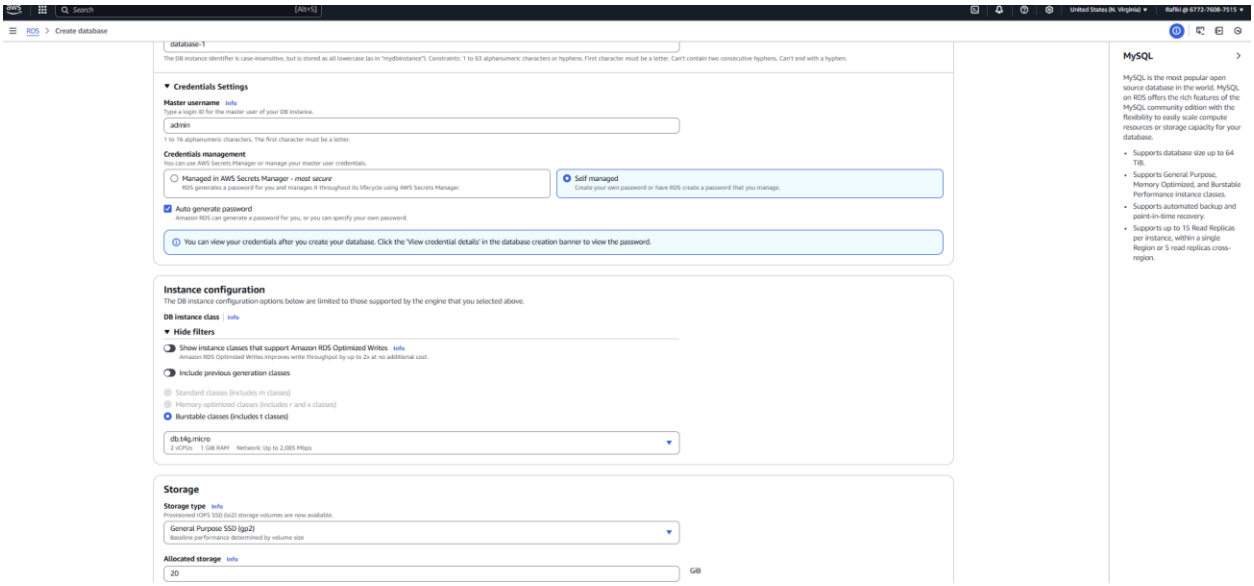
-Certificate of authority we go with the default one and monitoring remain default.

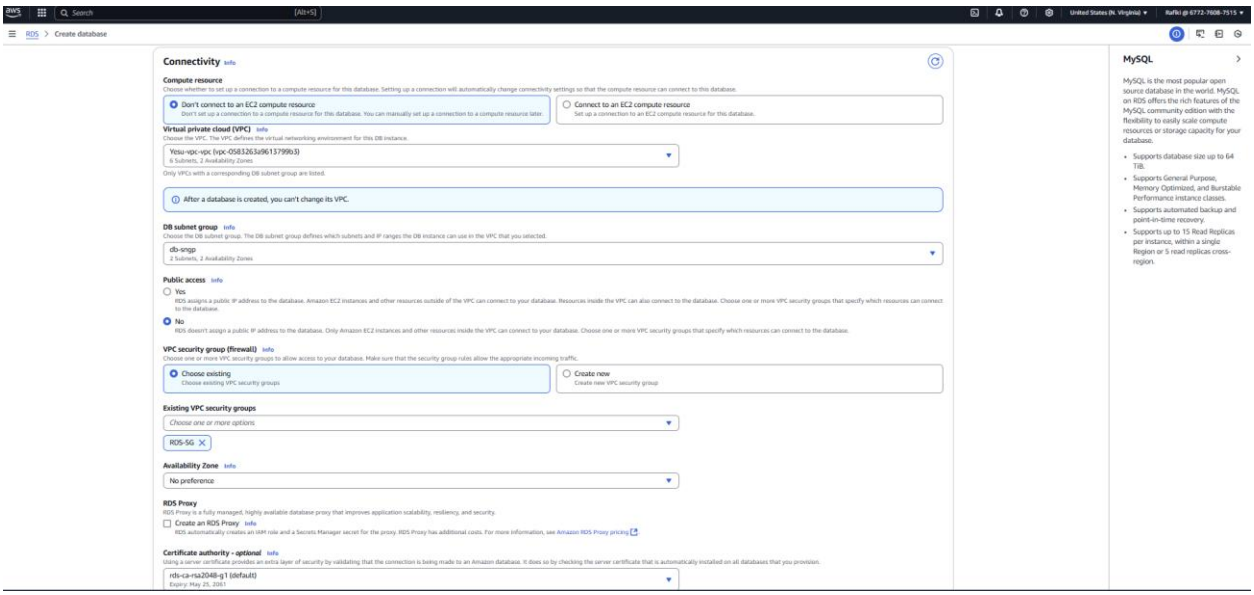
-For additional configuration ensure that option fall under the free tier estimated monthly cost.

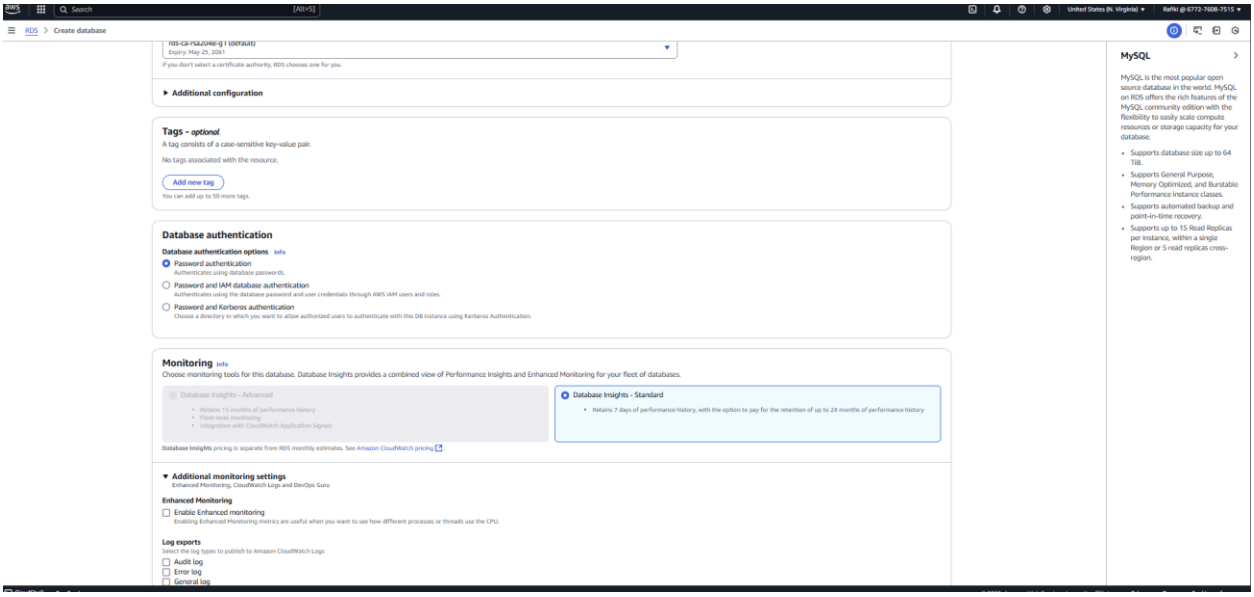
View set as follows ;

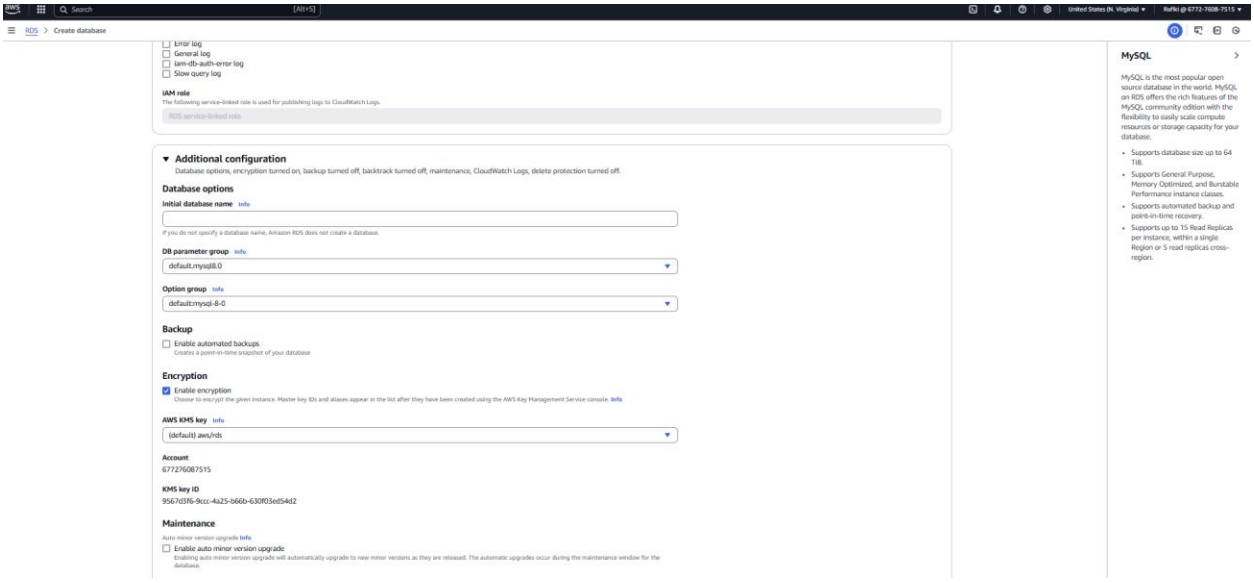


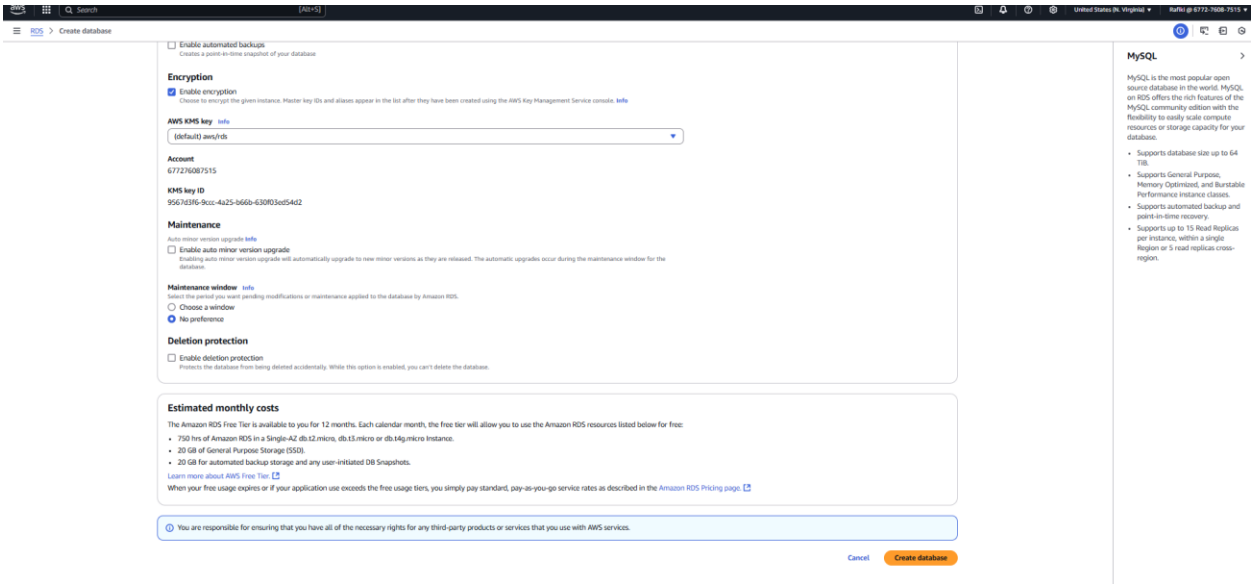



Search
[Alt+S]
United States (N. Virginia)
RDS > Create database
database-1
The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 63 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.
Credentials Settings
Master username Info
Type a login ID for the master user of your DB instance.
admin
1 to 16 alphanumeric characters. The first character must be a letter.
Credentials management
You can use AWS Secrets Manager or manage your master user credentials.
Managed in AWS Secrets Manager - most secure
RDS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.
Self managed
Create your own password or have RDS create a password that you manage.
Auto generate password
Amazon RDS can generate a password for you, or you can specify your own password.
You can view your credentials after you create your database. Click the 'View credential details' in the database creation banner to view the password.
Instance configuration
The DB instance configuration options below are limited to those supported by the engine that you selected above.
DB instance class Info
Hide filters
Show instance classes that support Amazon RDS Optimized Writes Info
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.
Include previous generation classes
Standard classes (includes m classes)
Memory optimized classes (includes r and x classes)
Burstable classes (includes t classes)
db.t4g.micro
2 vCPUs 1 GiB RAM Network: Up to 2,085 Mbps
Storage
Storage type Info
Provisioned IOPS SSD (io2) storage volumes are now available.
General Purpose SSD (gp2)
Baseline performance determined by volume size
Allocated storage Info
20
GiB
MySQL
MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.
Supports database size up to 64 TiB.
Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
Supports automated backup and point-in-time recovery.
Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

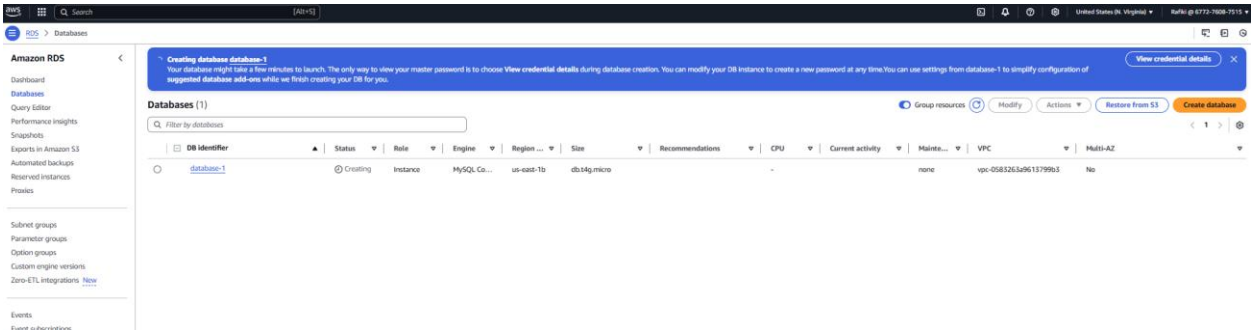

Search
[Alt+S]
United States (N. Virginia)
RDS > Create database
Connectivity Info
Compute resource
Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.
Don't connect to an EC2 compute resource
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.
Connect to an EC2 compute resource
Set up a connection to an EC2 compute resource for this database.
Virtual private cloud (VPC) Info
Choose the VPC. The VPC defines the virtual networking environment for this DB instance.
Yesu-vpc-vpc (vpc-0583263a96137990b3)
6 Subnets, 2 Availability Zones
Only VPCs with a corresponding DB subnet group are listed.
After a database is created, you can't change its VPC.
DB subnet group Info
Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.
db-sngp
2 Subnets, 2 Availability Zones
Public access Info
Yes
RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to this database.
No
RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.
VPC security group (firewall) Info
Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.
Choose existing
Choose existing VPC security groups
Create new
Create new VPC security group
Existing VPC security groups
Choose one or more options
RDS-SG
Availability Zone Info
No preference
RDS Proxy
RDS Proxy is a fully managed, highly available database proxy that improves application scalability, resiliency, and security.
Create an RDS Proxy Info
RDS automatically creates an IAM role and a Secrets Manager secret for the proxy. RDS Proxy has additional costs. For more information, see Amazon RDS Proxy pricing
Certificate authority - optional Info
Using a server certificate provides an extra layer of security by validating that the connection is being made to an Amazon database. It does so by checking the server certificate that is automatically installed on all databases that you provision.
rds-ca-rsa2048-g1 (default)
Expiry: May 25, 2061
MySQL
MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.
Supports database size up to 64 TiB.
Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
Supports automated backup and point-in-time recovery.
Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.


RDS > Create database
Additional configuration
Tags - optional
A tag consists of a case-sensitive key-value pair.
No tags associated with the resource.
Add new tag
You can add up to 50 more tags.
Database authentication
Database authentication options Info
Password authentication
Authenticates using database passwords.
Password and IAM database authentication
Authenticates using the database password and user credentials through AWS IAM users and roles.
Password and Kerberos authentication
Choose a directory in which you want to allow authorized users to authenticate with this DB instance using Kerberos Authentication.
Monitoring Info
Choose monitoring tools for this database. Database Insights provides a combined view of Performance Insights and Enhanced Monitoring for your fleet of databases.
Database Insights - Advanced
Database Insights - Standard
Retains 7 days of performance history, with the option to pay for the retention of up to 24 months of performance history
Additional monitoring settings
Enhanced Monitoring, CloudWatch Logs and DevOps Guru
Enhanced Monitoring
Enable Enhanced monitoring
Log exports
Audit log
Error log
General log
MySQL
MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.
Supports database size up to 64 TiB.
Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
Supports automated backup and point-in-time recovery.
Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.


RDS > Create database
Error log
General log
iam-db-auth-error log
Slow query log
IAM role
RDS service-linked role
Additional configuration
Database options, encryption turned on, backup turned off, backtrack turned off, maintenance, CloudWatch Logs, delete protection turned off.
Database options
Initial database name Info
If you do not specify a database name, Amazon RDS does not create a database.
DB parameter group Info
default.mysql8.0
Option group Info
default:mysql-8-0
Backup
Enable automated backups
Creates a point-in-time snapshot of your database
Encryption
Enable encryption
Choose to encrypt the given instance. Master key IDs and aliases appear in the list after they have been created using the AWS Key Management Service console. Info
AWS KMS key Info
(default) aws/rds
Account
677276087515
KMS key ID
9567d3f6-9ccc-4a25-b66b-630f03ed54d2
Maintenance
Auto minor version upgrade Info
Enable auto minor version upgrade
Enabling auto minor version upgrade will automatically upgrade to new minor versions as they are released. The automatic upgrades occur during the maintenance window for the database.
MySQL
MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.
Supports database size up to 64 TiB.
Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
Supports automated backup and point-in-time recovery.
Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.



-Enable database creation. The creation is not instantaneous so give the process of creation sometime.



6)

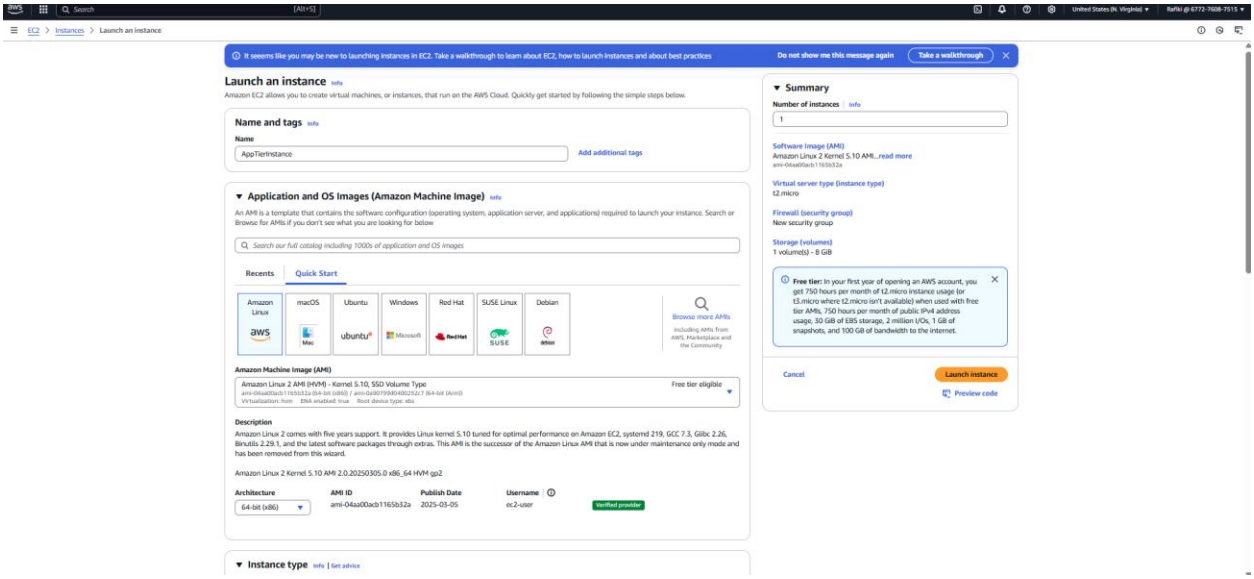
Next is lets create app tier resources.

-Deploy application-tier resources , include the configuration of an internal load balancer for traffic distribution within the tier.

-Here we are going to create ec2 instances to connect to (DB) database. When ever end user connect to database thru the App tier instances the goal is let the data get stored in the Database tier. Further, the required configuration will be implemented to connect instance to database.

-First, ec2 instance will be launched to connect to database. The configuration will produced from the first of 2 instances. The second instance will not be created manually , but will use an auto scaling group to launch the second instance. Then connect the auto scaling group to the load balancer. Load balancer will support and maintain AZ1 and AZ2 instances.

Below start by launching an instance.



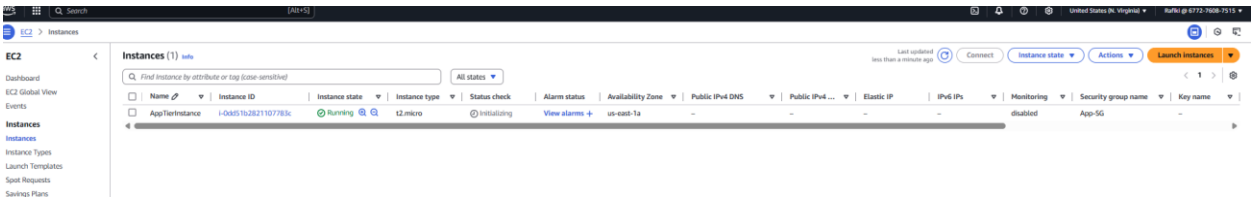
Here we will not connect with key pair.

In network settings, select the vpc we created. Yesu-vpc and should be connected to Yesu-vpc-App1-us-east-1a subnet only. Recall this is a private subnet only and should be connected to public subnet.

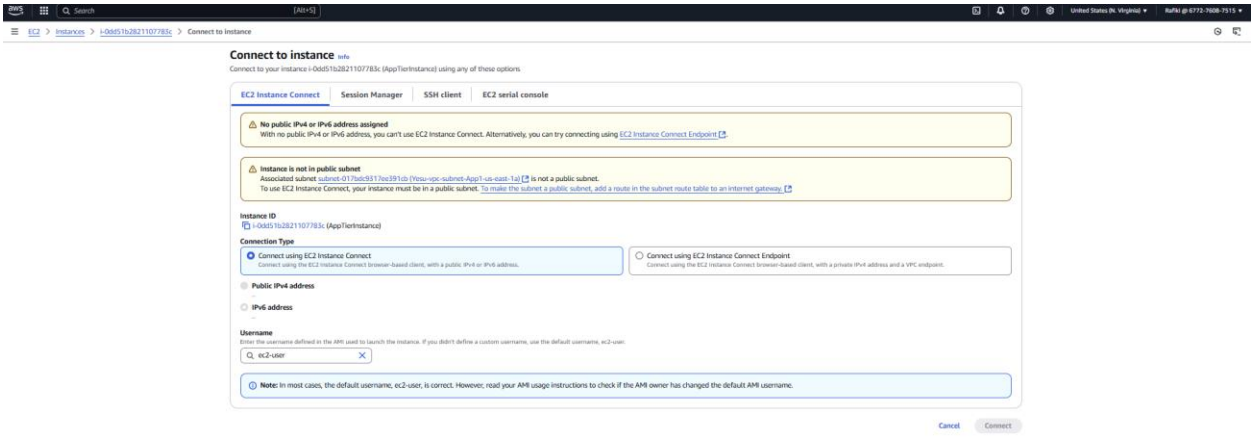
Firewall (security group) . Here recall we created the app security group called App-SG so attached to it.

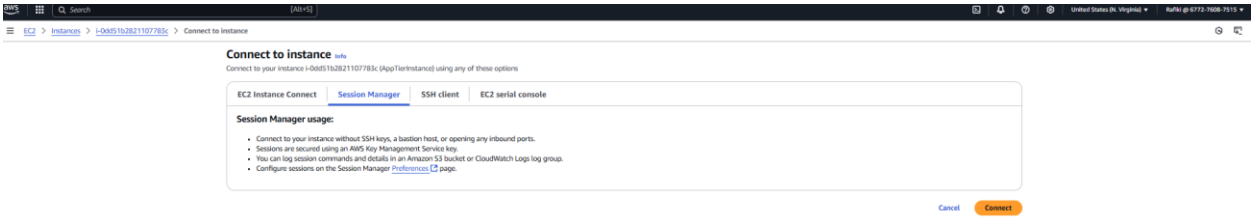
Storage can remain default for the free tier consumption.

-In the advanced details attaché the IAM role we created. **Yesu-ec2-Role.** Then go ahead and launch instance.



-In-order to connect instance to app tier or DB tier. Select ec2 and click connect session manager and connect tab. Here we will be connected ec2 instance session manager via a terminal.





-Now if we do not get to the part above , where you are not able to connect to session manager thru ec2 instance it means the security group has not been setup correctly. This also means Port 4000 is not open.

-On this instance terminal we have to setup configuration to connect the database.

-First lets switch to root user by typing the following command ; sudo su –

-To certain you are in the root user type ;**whoami** and the output should state **root**

-At the prompt type to reset to default directory ; prompt :$ **cd ..**

-Then type type pwd

-The goal is to be in the user directory or **/usr**

-From here we need to move to ec2user. So type ; **cd /home/ec2-user/**

-Type pwd to know where you are or current working directory

-Next test for connectivity by pinging google ; # 8.8.8.8 ; Therefore you should see a response

-So how is the internet connecting to this instance ? Via internet gateway.

-Next install mysql client

sudo yum install mysql –y

-For us to connect to database and perform basic configuration. Use the following command below ;

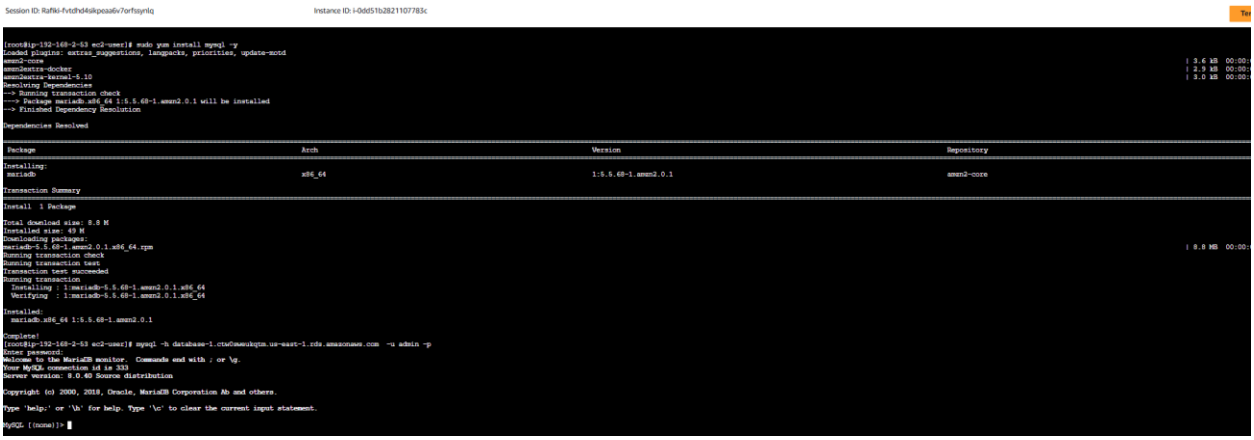
Db password : 3TqrzMz4T6lkKksUqY9u

Endpoints : database-1.ctw0sweukqtm.us-east-1.rds.amazonaws.com

Port : 3306

-Execute the following command to log into database ; mysql -h database-1.ctw0sweukqtm.us-east-1.rds.amazonaws.com -u admin –p

-Below we are logged into Mysql database



-Once logged into database create a database named webappdb . Once the database has been created also create a table.

-Also, check whether we created database or not ? Type SHOW DATABASES (command) .

-After database creation has been verified we also need to use it. So at the prompt type : USE webappdb

-Next we are going to create a table called ‘transactions’ using the commads from git hub repository.

-Using the following command create table ;

CREATE TABLE IF NOT EXISTS transactions(

id INT NOT NULL AUTO_INCREMENT,

amount DECIMAL(10,2),

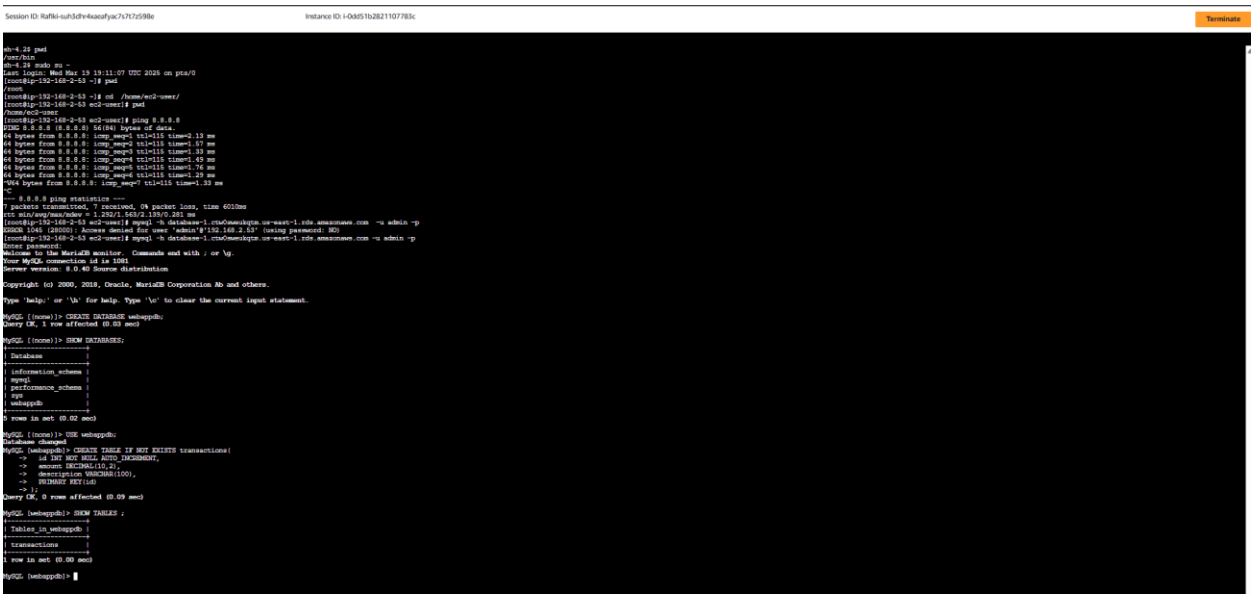
description VARCHAR(100),

PRIMARY KEY(id)

);

-Now verify the table was created type the following command > SHOW TABLES ;

-



```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 1081
Server version: 8.0.40 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MySQL [(none)]> CREATE DATABASE webappdb;
Query OK, 1 row affected (0.03 sec)

MySQL [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| webappdb           |
+--------------------+
5 rows in set (0.02 sec)

MySQL [(none)]> USE webappdb;
Database changed
MySQL [webappdb]> 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 (0.09 sec)

MySQL [webappdb]> SHOW TABLES ;
+------------------+
| Tables_in_webappdb |
+------------------+
| transactions       |
+------------------+
1 row in set (0.00 sec)

MySQL [webappdb]> INSERT INTO transactions (amount, description) VALUES ('400', 'groceries');
Query OK, 1 row affected (0.00 sec)
```

-Now we need to insert some dummy information in the table to test its functionality. Now check the entries in the table by executing the following command ; SELECT *FROM transactions;

- Now since we tests the entering contents from the instance this may not be the reality in the real time senario.

-In conclussion – We /I was able to able to connect to the database server through the appTier and able to write information via the database server.

Now recall database Tier is not required to conenct to appTier or webTier. It exists in the back end.

-Next we need to exit the database so type **exit** command.

7)

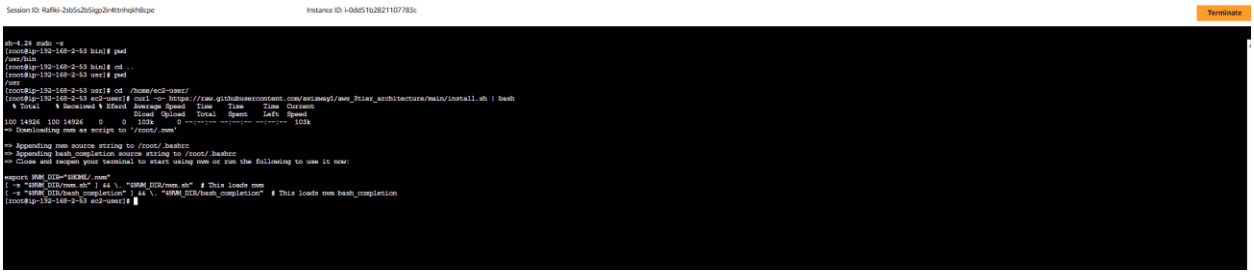
One more cofiguration is required once you exit database. We need to locate the application code on our host machine and find the file called **DBConfig.js.** Open this file with visual studio code and make some changes. Copy our the database endpoints and paste it to line number 2 my code. User remains default admin . Change the database password to current database. Database name should be the same as we created. Ensure file is saved sucessfully. Ctrl S

-Now lets go to the s3 service . Locate the application code we uploaded and update the DBconfig.js file as well. So we need to create a new one. So drag and drop the updated file into the s3 bucket to relpalce the old one. The will be no issues since we had not enable versioning in on our project folder.

8)

Next to need to install and configure nodejs. The instructions are in the 3 tier repository.

Using the following command; curl -o-
https://raw.githubusercontent.com/avizway1/aws_3tier_architecture/main/install.sh | bash.



-The we need to source it by going to the bash shell. Type the following command in the prompt ; source ~/ .bashrc

-Next **install nvm** , also known as node version manager. nvm install 16

-Now tell our manager to use nvm 16 type ; **nvm user 16**

-Now run node as a service. For that install ; **npm install –g pm2**

-We need to switch our path from root to user. How ? Type pwd to know where you are in the file system. **Type cd ..** .

The above command takes you home. Or type cd ~/

-Copy the path for the s3 bucket to app-tier folder

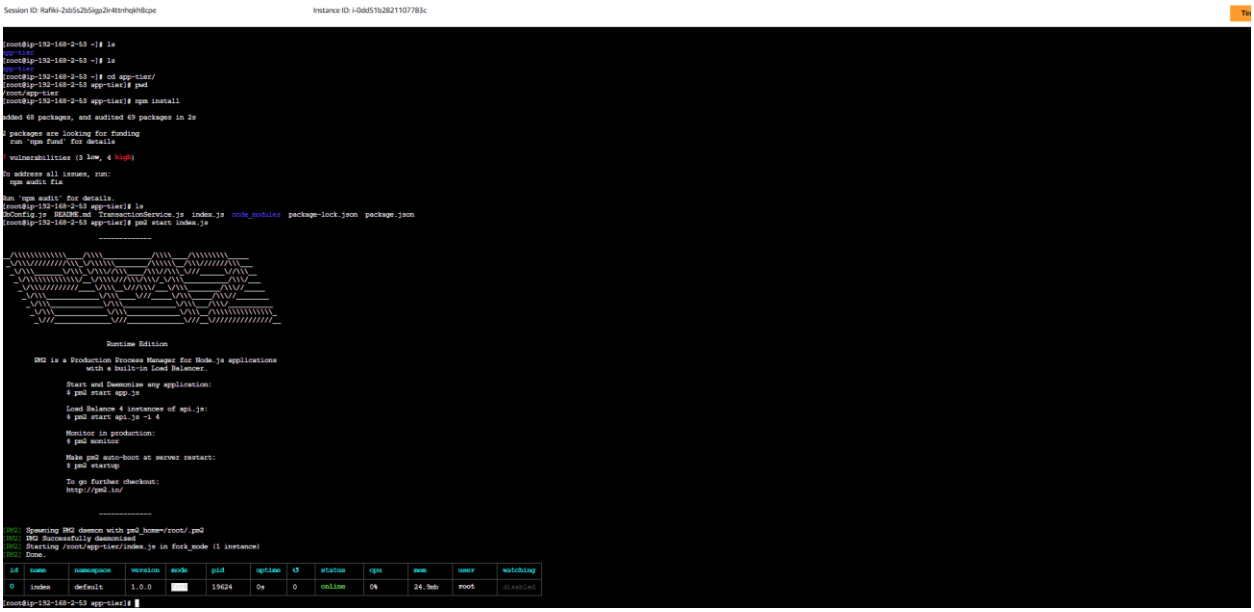
Root ~] # sudo aws s3 cp s3://**yesu-3tier-project**/application-code/app-tier/ app-tier –recursive

- Switch to app-tier directory type ; cd app-tier/

Install node port manager type ; npm install

- Type **ls** and you will able to view an **index.js** file.

-Start index.js file type ; pm2 start index.js



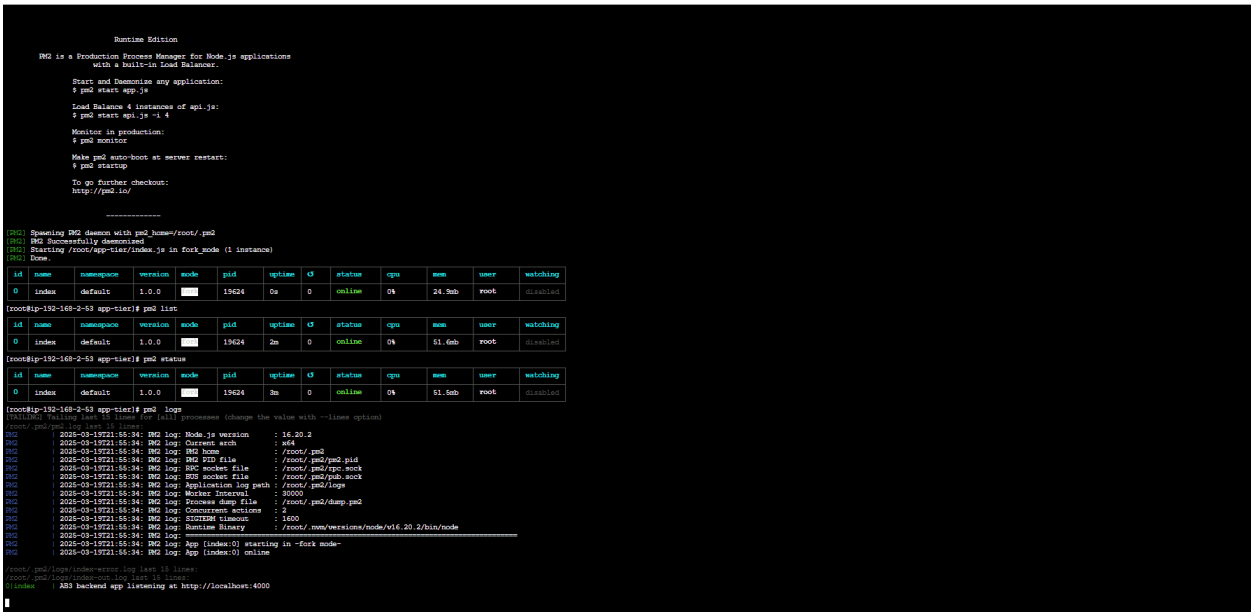
Now from the session manager terminal you should online status in the table described as the image above. Also type in the terminal ; **pm2 list**

-And **pm status**

- type ; pm2 logs

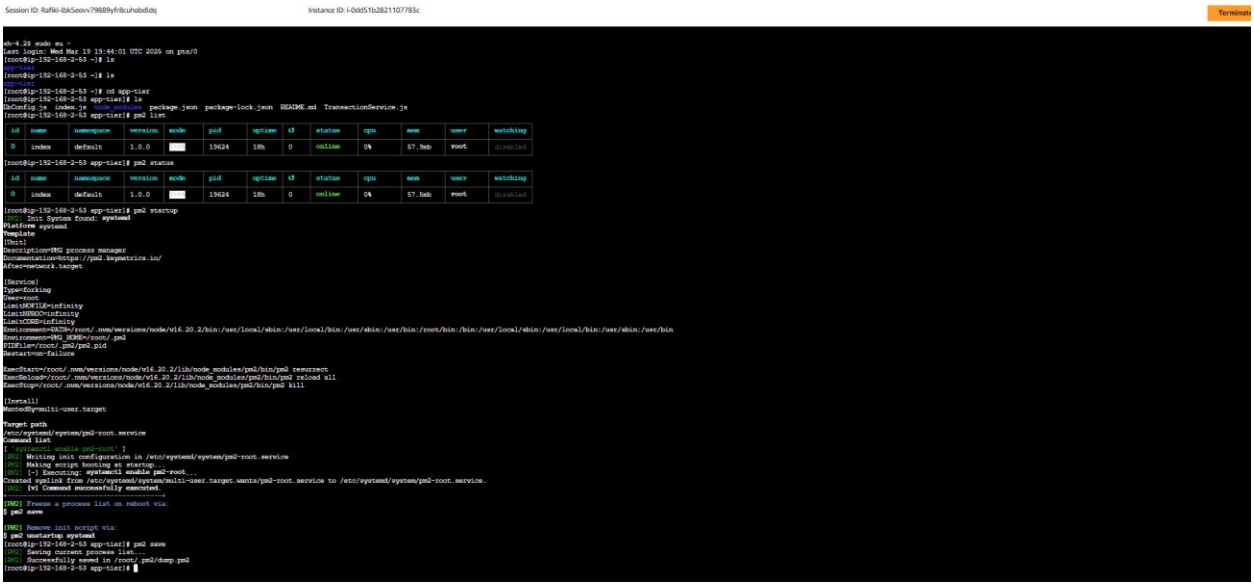
Type control C to return to terminal prompt.

Session ID: Rafiki-2sb5s2b5igp2ir4ttnhqkh8cpe Instance ID: i-0dd51b2821107783c



- Now type; **pm2 startup**

-Ensure that we save current configuration by typing ; **app-tier]# pm2 save**



- Verify that the application is running by executing the curl command ;
- Curl http://localhost: 4000/health
- It should return this is a health check.
-

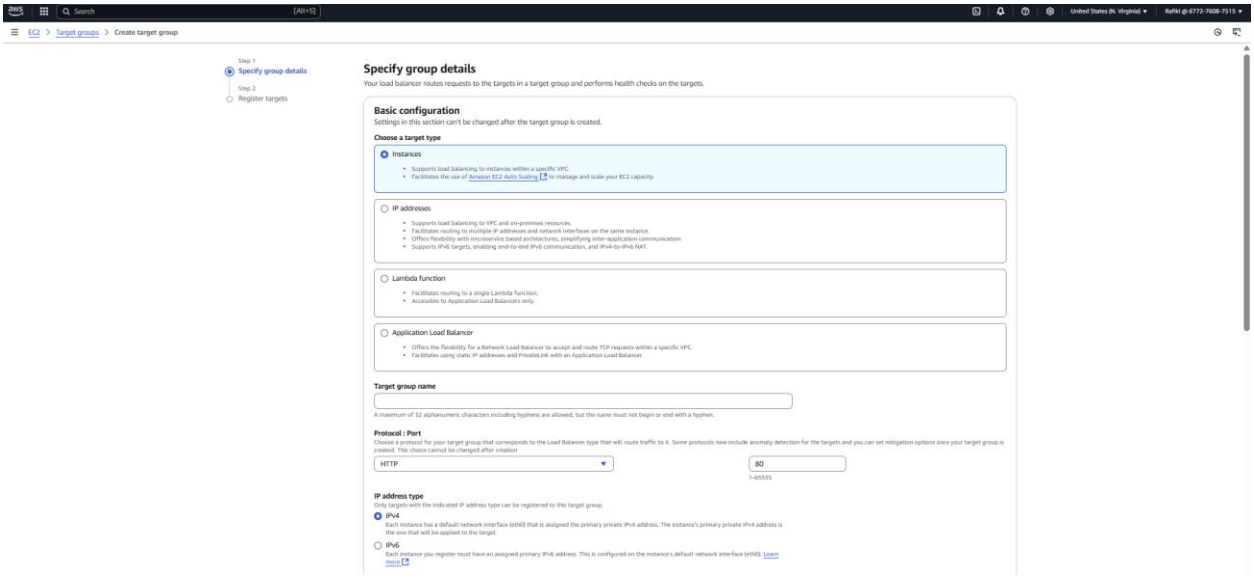
```
[PM2] Remove init script via:
$ pm2 unstartup systemd
[root@ip-192-168-2-53 app-tier]# pm2 save
[PM2] Saving current process list...
[PM2] Successfully saved in /root/.pm2/dump.pm2
[root@ip-192-168-2-53 app-tier]# curl http://localhost:4000/health
"This is the health check"[root@ip-192-168-2-53 app-tier]#
```

-Recall in the 3tier architecture rendering we have a internal load balancer that connects web-tier to app-tier. So, we need to create one. Prior to load balancer create target groups first . Click the target group tab chose target type as instances and give it a name. Say **App-Internal-TG.**

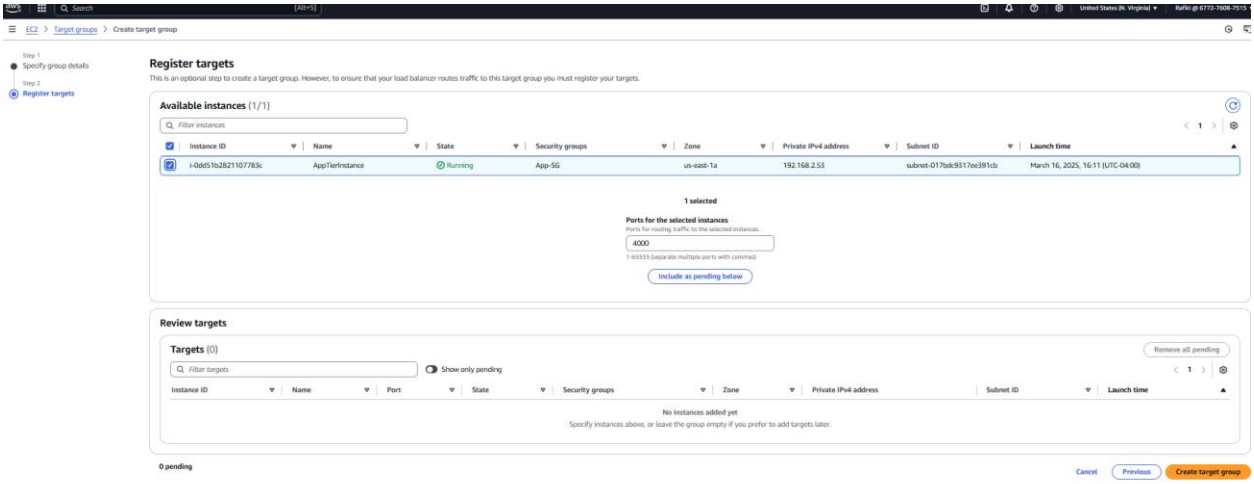
-Protocol as HTTP and recall our application runs on port 4000.

-IP address type ; IPv4

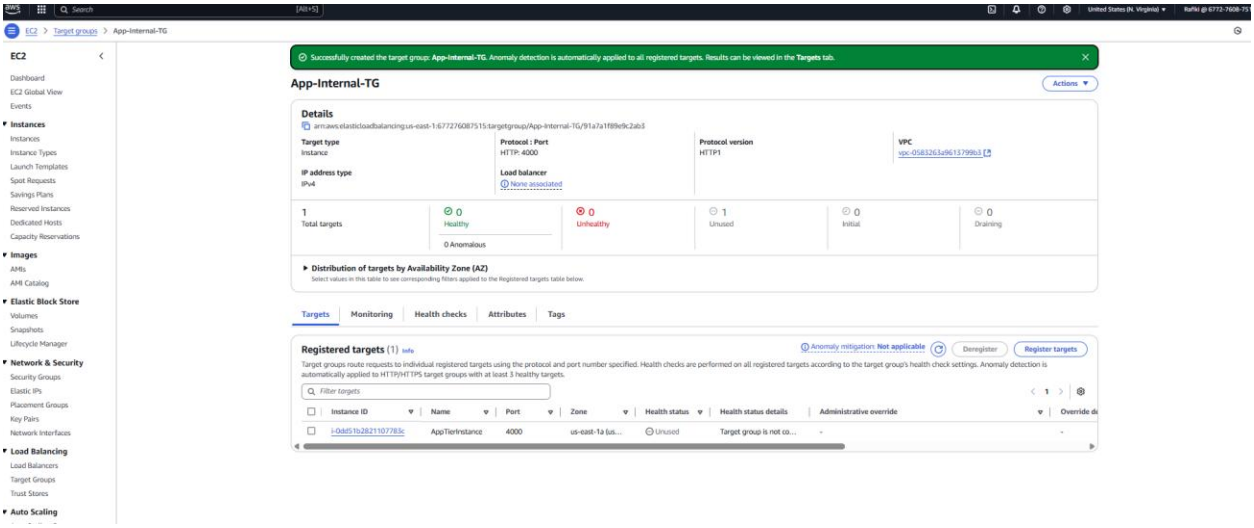
-Choose VPC is the same we created at beginning of project



-Health HTTP only . Health check path enter ; /health

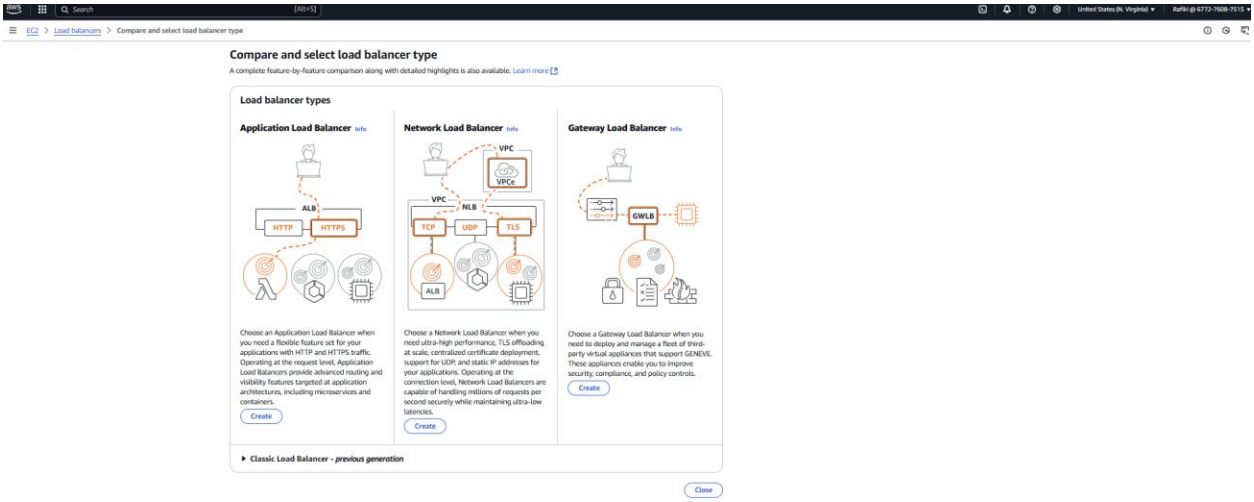


-Chose the target by clicking Register instance as **AppTierInstance .** Click include as pending below and activate create target group.



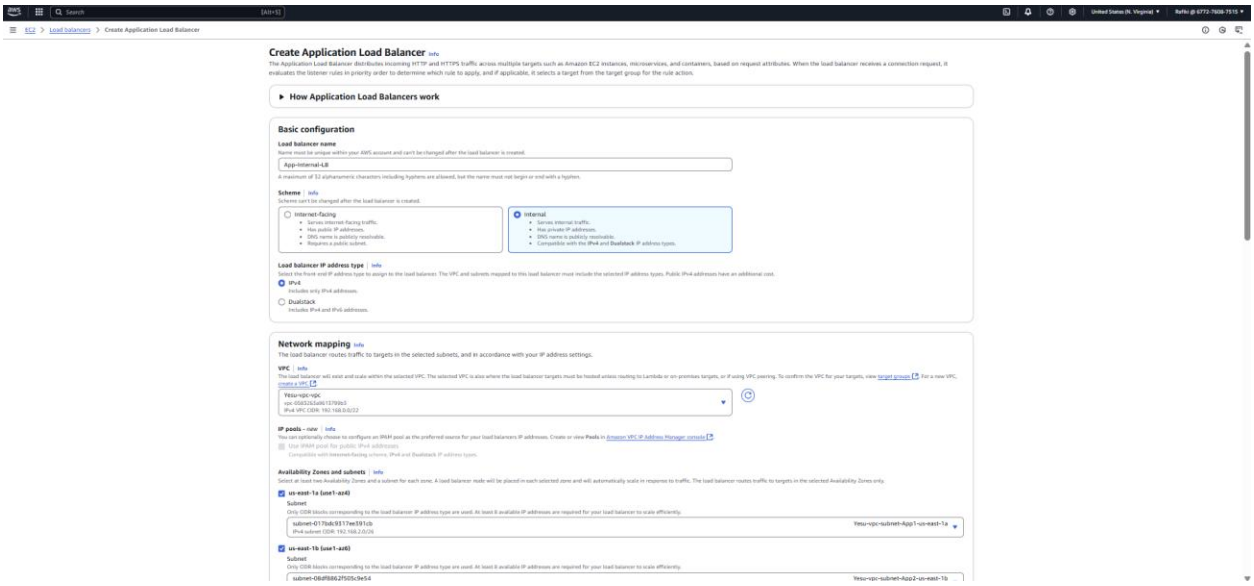
-As the image shows we have created the target group.

-Next create load balancer. We are going to chose application Load Balancer.



-We choose application load balancer . Provider a load balancer name ; **App-Internal-LB**

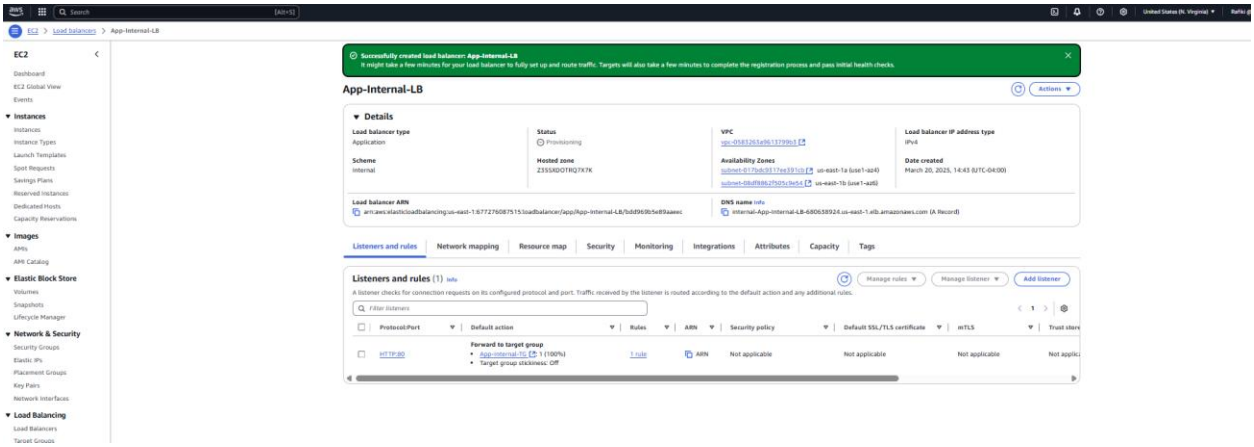
-Network mapping – here we choose vpc we created. **Yesu-vpc.** Also, make sure select the correct private Subnets. In my case I chose Yesu-vpc-App1-us-east-1a and Yesu-vpc-App2-us-east-1b.



-Recall we created our own internal group at the beginning of the project so select **internal-ALB-SG**

-For the listeners and routing direct protocol as HTTP , port 80 , default action ; **App-internal-TG**

-Click create load balancer.



-Now it is also important to note the DNS of the load balancer.

DNS name : internal-App-Internal-LB-680638924.us-east-1.elb.amazonaws.com

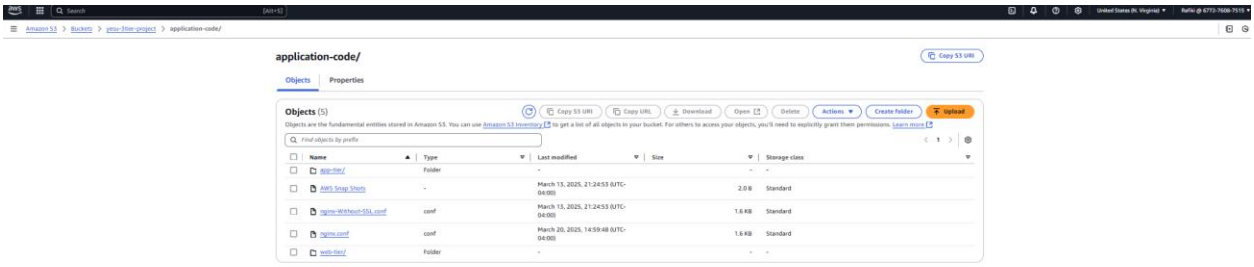
-So, with the DNS name we are going to configure the nginx.conf file. Open nginx.conf with code editor in my case is visual studio code. At the bottom of this file change the DNS name to name below and save file ;

internal-App-Internal-LB-680638924.us-east-1.elb.amazonaws.com.

The change should appear like this ;

```
  #proxy for internal lb
        location /api/ {
            proxy_pass http://internal-App-Internal-LB-680638924.us-east-
1.elb.amazonaws.com:80/;
        }
```

-Also ,remember to upload the updated nginx.conf file to s3 bucket. Drag and drop file and the file the file will automatically update.



** IF you still the app-tier ec2 via SSM (session manager) . Do a health check on the host app-tier.

-Verify that the application is running by executing

App-tier] # curl http://localhost:4000/health

```
[root@ip-192-168-2-53 app-tier]#  curl http://localhost:4000/health
"This is the health check"[root@ip-192-168-2-53 app-tier]#
```

_With the message it means we have completed the app-tier configuration.

-Eventually we need to place this file in the webserver or Web-tier. Thus we have successfully created the app-tier resources.

8)
Next, we going to implement web-tier resource creation.
-So lets create ec2 instance for the web-tier. First , are going to intall node.js and nginx.
-
Here we will create **web-tier instance**

-select Amazon as your goto OS
-Then Amazon linux 2 AMI –SSD Volume type (free tier eligible)
-instance type - t2.micro

-No key pair. Proceed without key pair

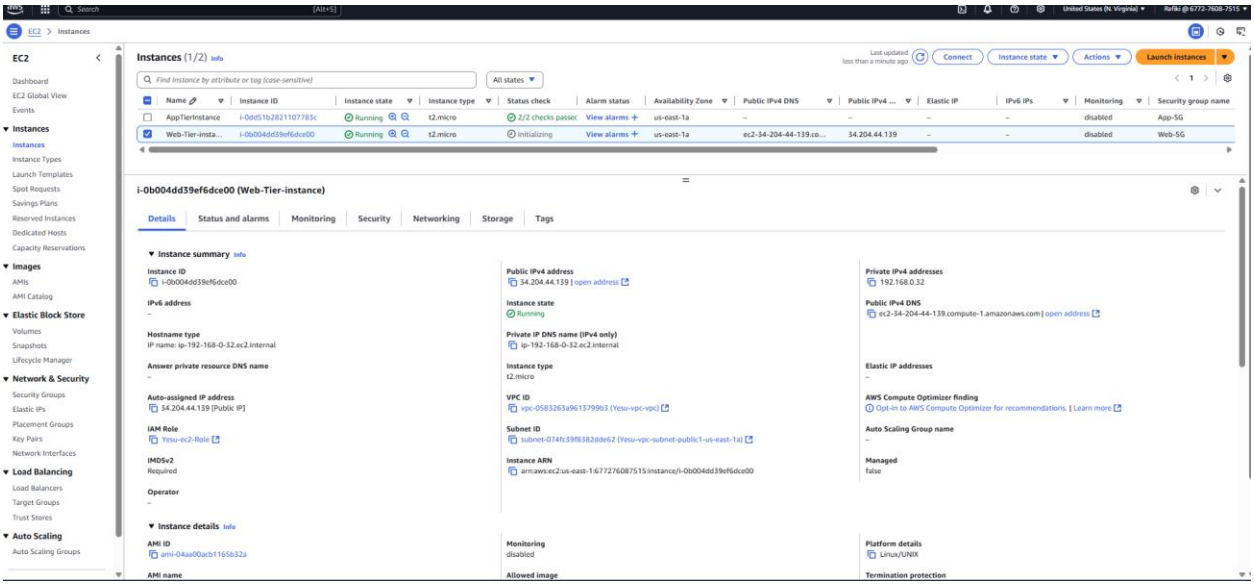
Network settings

***** In network setting make sure you edit and choose existing vpc we created at the beginning of the project **not default vpc**. In we recall the vpc we created is named Yesu-vpc.

-Also, recall we also created 2 public subnets. Yesu-vpc-subnet-public1-us-east-1a and Yesu-vpc-subnet-public2-us-east-1a . We can choose either one.
-In addition enable assign public IP option too.
-Next select existing security group we created at the beginning of the project. In this case the security we created we named Web-SG (Web security group)

-In advanced configuration make we attach IAM role called **Yesu-ec2-Role**

See image below of the Web-Tier instance.



-

-Now connect ec2 instance via session manager

-At the terminal switch to root user.] $ sudo –su ec2-user

```
sh-4.2$ sudo -su ec2-user
[ec2-user@ip-192-168-0-32 bin]$
[ec2-user@ip-192-168-0-32 bin]$
[ec2-user@ip-192-168-0-32 bin]$
```

-Switch to ec2-user HOME/ → type below command in terminal
] $ cd /home/ec2-user

```
[ec2-user@ip-192-168-0-32 bin]$ cd /home/ec2-user
[ec2-user@ip-192-168-0-32 ~]$ pwd
/home/ec2-user
[ec2-user@ip-192-168-0-32 ~]$
```

-type pwd to ensure you are in the right location
-Now use curl command to install NMV source environment in the bash script. Then install NVM 16 and use it.

```
[ec2-user@ip-192-168-0-32 ~]$ curl -o- https://raw.githubusercontent.com/avizway1/aws_3tier_architecture/main/install.sh | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14926  100 14926    0     0  71117      0 --:--:-- --:--:-- --:--:-- 71076
=> Downloading nvm as script to '/home/ec2-user/.nvm'

=> Appending nvm source string to /home/ec2-user/.bashrc
=> Appending bash_completion source string to /home/ec2-user/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash_completion
[ec2-user@ip-192-168-0-32 ~]$ source ~/.bashrc
[ec2-user@ip-192-168-0-32 ~]$ nvm install 16
Downloading and installing node v16.20.2...
Downloading https://nodejs.org/dist/v16.20.2/node-v16.20.2-linux-x64.tar.xz...
######################################################################################################################################################################
Computing checksum with sha256sum
Checksums matched!
Now using node v16.20.2 (npm v8.19.4)
Creating default alias: default -> 16 (-> v16.20.2)
[ec2-user@ip-192-168-0-32 ~]$ use nvm 16
bash: use: command not found
[ec2-user@ip-192-168-0-32 ~]$ nvm use 16
Now using node v16.20.2 (npm v8.19.4)
[ec2-user@ip-192-168-0-32 ~]$
```

-Next we need to copy the contents from the s3 bucket suing the following script below ;

Buket name : yesu-3tier-project

aws s3 cp s3://yesu-3tier-project/application-code/web-tier/ web-tier –recursive

```
[ec2-user@ip-192-168-0-32 ~]$  aws s3 cp s3://yesu-3tier-project/application-code/web-tier/ web-tier --recursive
download: s3://yesu-3tier-project/application-code/web-tier/README.md to web-tier/README.md
download: s3://yesu-3tier-project/application-code/web-tier/src/components/DatabaseDemo/DatabaseDemo.js to web-tier/src/components/DatabaseDemo/DatabaseDemo.js
download: s3://yesu-3tier-project/application-code/web-tier/public/robots.txt to web-tier/public/robots.txt
download: s3://yesu-3tier-project/application-code/web-tier/public/index.html to web-tier/public/index.html
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Burger/index.js to web-tier/src/components/Burger/index.js
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Menu/index.js to web-tier/src/components/Menu/index.js
download: s3://yesu-3tier-project/application-code/web-tier/src/components/index.js to web-tier/src/components/index.js
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Burger/Burger.js to web-tier/src/components/Burger/Burger.js
download: s3://yesu-3tier-project/application-code/web-tier/package.json to web-tier/package.json
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Burger/Burger.styled.js to web-tier/src/components/Burger/Burger.styled.js
download: s3://yesu-3tier-project/application-code/web-tier/src/components/DatabaseDemo/DatabaseDemo.css to web-tier/src/components/DatabaseDemo/DatabaseDemo.css
download: s3://yesu-3tier-project/application-code/web-tier/src/global.js to web-tier/src/global.js
download: s3://yesu-3tier-project/application-code/web-tier/src/setupTests.js to web-tier/src/setupTests.js
download: s3://yesu-3tier-project/application-code/web-tier/src/theme.js to web-tier/src/theme.js
download: s3://yesu-3tier-project/application-code/web-tier/src/reportWebVitals.js to web-tier/src/reportWebVitals.js
download: s3://yesu-3tier-project/application-code/web-tier/src/index.js to web-tier/src/index.js
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Menu/Menu.js to web-tier/src/components/Menu/Menu.js
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Menu/Menu.styled.js to web-tier/src/components/Menu/Menu.styled.js
download: s3://yesu-3tier-project/application-code/web-tier/src/components/Home/Home.js to web-tier/src/components/Home/Home.js
download: s3://yesu-3tier-project/application-code/web-tier/src/App.test.js to web-tier/src/App.test.js
download: s3://yesu-3tier-project/application-code/web-tier/src/hooks.js to web-tier/src/hooks.js
download: s3://yesu-3tier-project/application-code/web-tier/src/index.css to web-tier/src/index.css
download: s3://yesu-3tier-project/application-code/web-tier/src/App.css to web-tier/src/App.css
download: s3://yesu-3tier-project/application-code/web-tier/src/App.js to web-tier/src/App.js
download: s3://yesu-3tier-project/application-code/web-tier/src/assets/3TierArch.png to web-tier/src/assets/3TierArch.png
[ec2-user@ip-192-168-0-32 ~]$
```

To check that we created the web-tier folder with contents type the following command ;

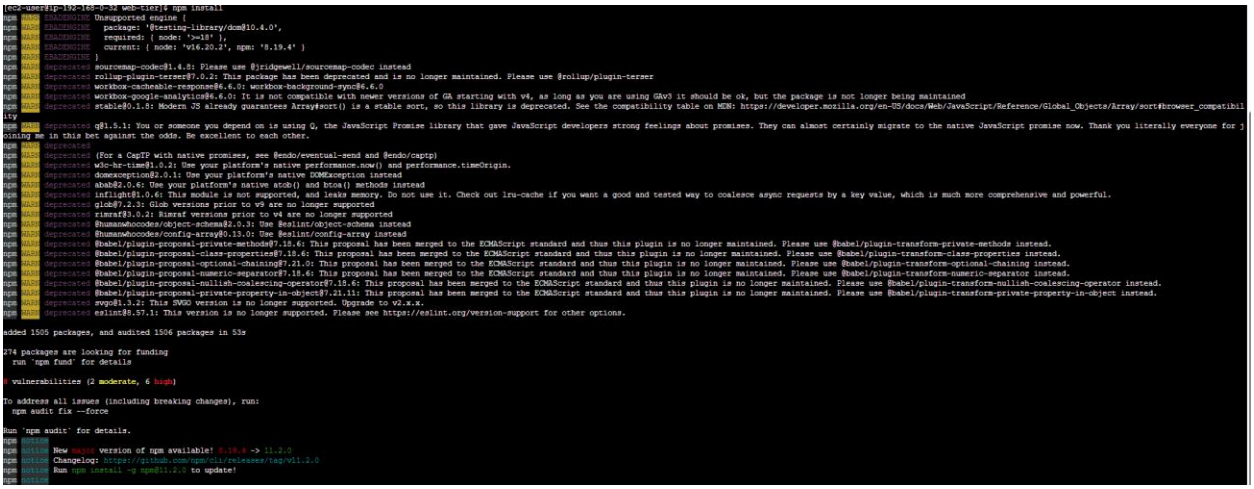
~] $ ls

```
[ec2-user@ip-192-168-0-32 ~]$ ls
web-tier
```

-Then switch to web-tier directory

```
[ec2-user@ip-192-168-0-32 ~]$ ls
web-tier
[ec2-user@ip-192-168-0-32 ~]$ cd web-tier/
[ec2-user@ip-192-168-0-32 web-tier]$
```

Webtier]$ npm install



Webtier]$ npm run build --- > thype this command to run it.

```
[ec2-user@ip-192-168-0-32 web-tier]$ npm run build

> aws-3tier-web-layer@0.1.0 build
> react-scripts build

Creating an optimized production build...
Compiled successfully.

File sizes after gzip:

  74.93 kB  build/static/js/main.4dff5ef7.js
  1.79 kB   build/static/js/453.a4ec9c9e.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
```

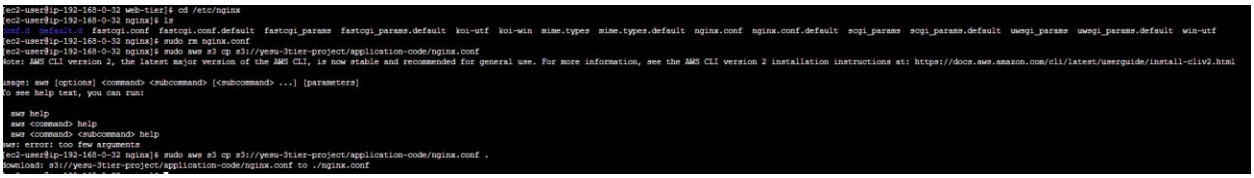
-Next we have to install nginx server using the following command ;
Web-tier] $ sudo amazon-linux-extras install nginx1 –y

-Once install update the nginx configuration. First switch to nginx path
Web-tier]$ cd /etc/nginx --→ Use the ls command and find file called nginx conf.
-In is important that we remove this file and update with the new one from the s3 bucket.
-Use the following command to remove it

Nginx] $ sudo rm nginx.conf.

Then use the below command to update with the new file.

] $ sudo aws s3 cp s3://yesu-3tier-project/application-code/nginx.conf . <…….Make sure in the end of the path there is a period or dot.



- Now lets restart nginx
- Nginx]$ sudo service nginx restart

Give permission to read , write an execute

Nginx] $ chmod -R 755 /home/ec2-user

-Now use the below following command to turn on nginx.

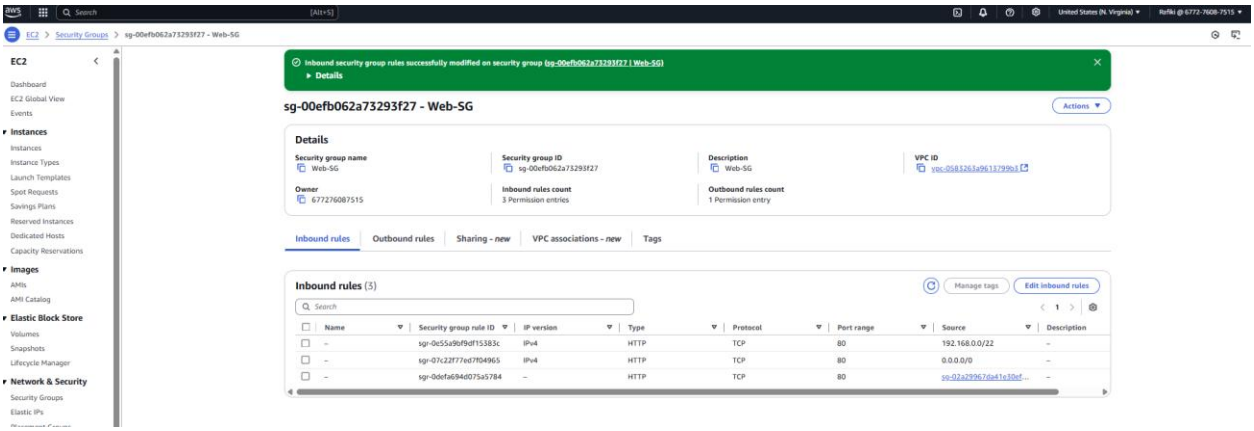
Nginx] $ sudo chkconfig nginx on

- To check the output of the App, we can check using the Web-Tier-Instance public IP. But before checking lets open port no 80 with http, Anywhere IPv4, 0.0.0.0/0 ---> Save rules ----> Now paste the pubic ip of Web-Tier-Instance in new tab of browser ----> You will see the app ----> Enter the data in the app

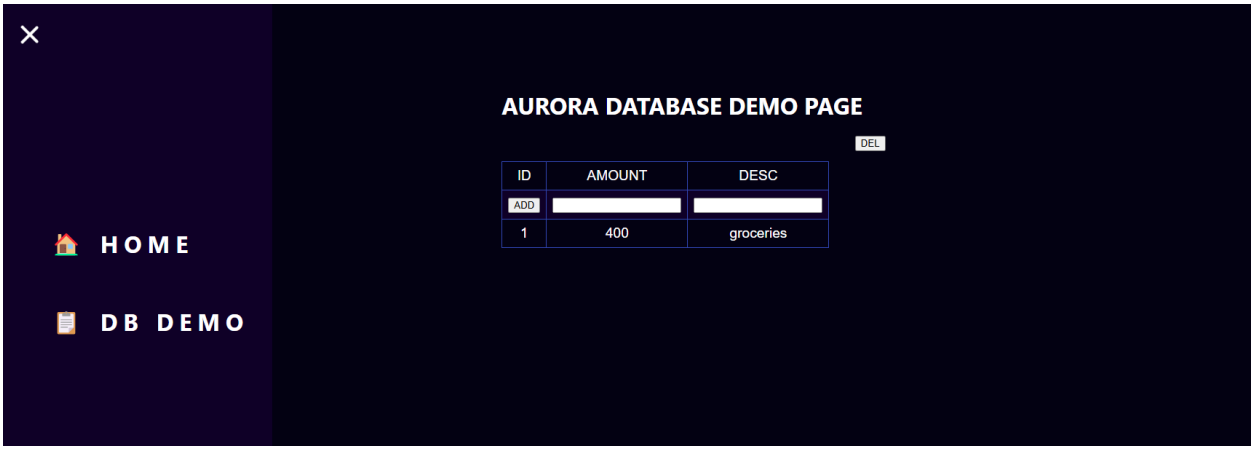
-Now go to web-tier instance select and click security tab. Click on security group link.

sg-00efb062a73293f27

-Edit inbound rules. Click add rule → protocol HTTP , anywhere IP , IP 0.0.0.0/0



-Next select instance copy the IP address paste it on the internet.



-Here our high availability fault tolerance 3-tier project is successfully complete aa demonstrated.

-In the real work or production environment we do not share IP address to the end user. Therefore, for security we have to create our own domain. This we can do by using route 53 in aws.

-So the first thing is create an external load balancer. Here also, it is not recommended to give the DNS address to the end user. So, map the dns name of the load balancer to route 53 custom domain.

-