Name :- Komal Mhetre
Role :- DevOps (Intern)
Task :- 3-Tier Application Architecture On AWS

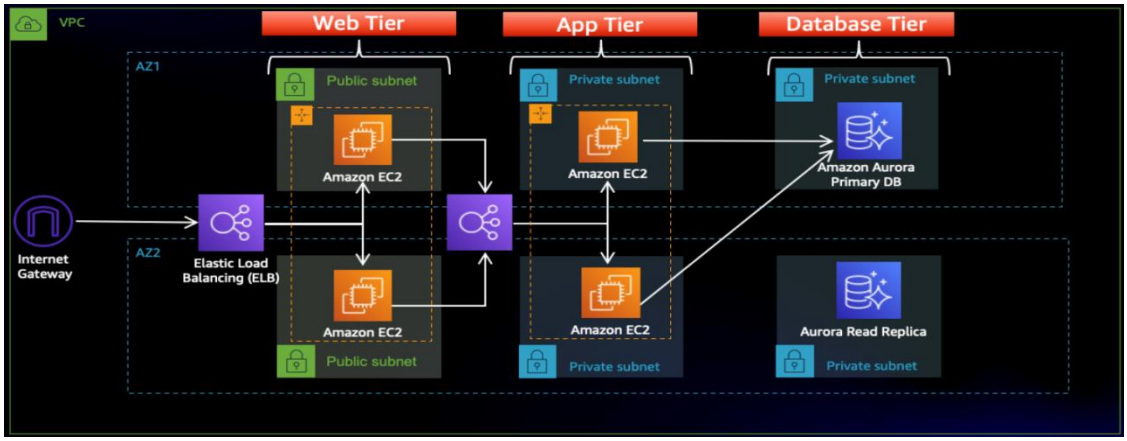
Introduction

A three-tier application architecture is a common design pattern used to develop and deploy applications, consisting of three layers: -Presentation Tier -Application Tier -Data Tier -Each tier serves a specific purpose and can be scaled, managed, and deployed independently.

Three Tiers

1. Presentation Tier :-
    - This tier is responsible for handling user interactions and presenting information to users.
    - Common components in this tier include web servers, load balancers, and content delivery networks (CDNs).
    - AWS services commonly used for the presentation tier include: -Amazon EC2 -Elastic Load Balancing (ELB) -Amazon CloudFront-Elastic Beanstalk
2. Application Tier :-
    - This tier contains the business logic and application processing components of the system.
    - Common components in this tier include application servers, APIs, and middleware.
    - AWS services commonly used for the application tier include: -Amazon EC2 or AWS Lambda -Amazon API Gateway
3. Data Tier :-
    - This tier stores and manages data used by the application.
    - Common components in this tier include databases, data warehouses, and data lakes.
    - AWS services commonly used for the data tier include: -Amazon RDS -Amazon DynamoDB -Amazon Redshift

Architecture



Setup

- Download source code from below GitHub repository to your local machine:
- "https://github.com/aws-samples/aws-three-tier-web-architectureworkshop.git"
- Extract files/folders from your zip folder and use accordingly

```
C:\Users\user7\Downloads>mkdir demo-three-tier

C:\Users\user7\Downloads>cd demo-three-tier

C:\Users\user7\Downloads\demo-three-tier>git clone https://github.com/aws-sampl
Cloning into 'aws-three-tier-web-architecture-workshop'...
remote: Enumerating objects: 133, done.
remote: Counting objects: 100% (26/26), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 133 (delta 15), reused 10 (delta 10), pack-reused 107
Receiving objects: 100% (133/133), 238.42 KiB | 144.00 KiB/s, done.
Resolving deltas: 100% (49/49), done.
```

IAM Role Setup

IAM EC2 Instance Role Creation:-

i. Navigate to the IAM dashboard in the AWS console and create an EC2 role.
ii. Select EC2 as the trusted entity.
iii. When adding permissions, include the following AWS managed policies. You can search for them and select them. These policies will allow our instances to download our code from S3 and use Systems Manager Session Manager to securely connect to our instances without SSH keys through the AWS console.
   - AmazonSSMManagedInstanceCore
   - AmazonS3ReadOnlyAccess

Give your role a name, and then click Create Role

Step 2: Add permissions Edit

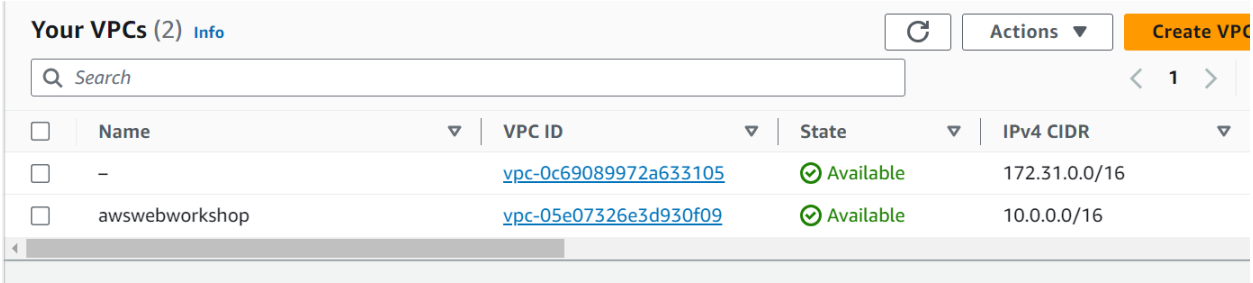
Permissions policy summary

Policy name	Type	Attached as
AmazonS3ReadOnlyAccess	AWS managed	Permissions policy
AmazonSSMManagedInstanceCore	AWS managed	Permissions policy

Networking Setup

1. VPC and Subnet Creation :-

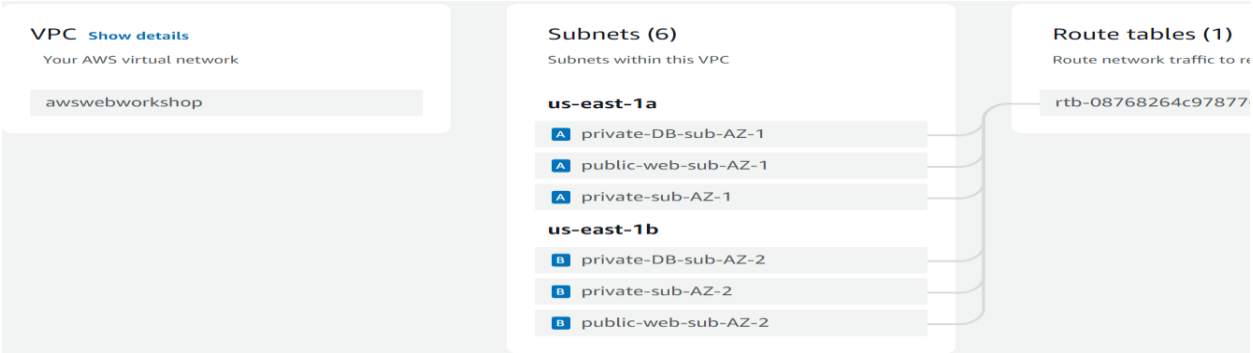
i) Make sure VPC only is selected, and fill out the VPC Settings with a Name tag and a CIDR range of your choice.



2.Subnet Creation :-

i) Create 6 subnets across 2 AZ.s within created VPC. That means that three subnets will be in one availability zone, and three subnets will be in another zone.

ii) Specify unique CIDR range for each subnet



Internet Connectivity

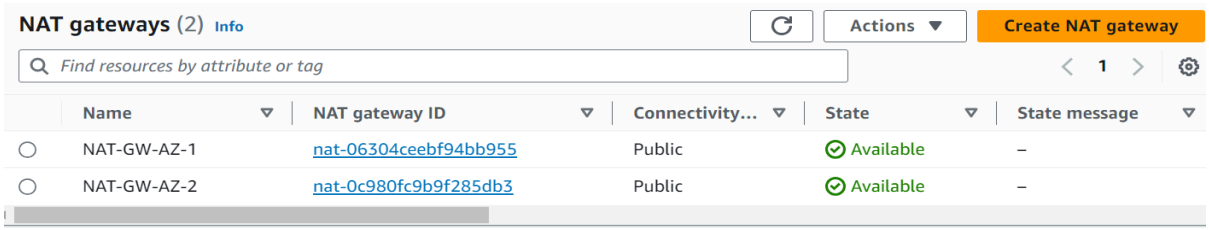
1.Create internet gateway:-

i) Create your internet gateway by simply giving it a name and clicking Create internet gateway.

ii) Attach internet gateway to your VPC.

2.Create NAT Gateway :-

i) Fill in the Name, choose one of the public subnets and then allocate an Elastic IP



Routing Configuration

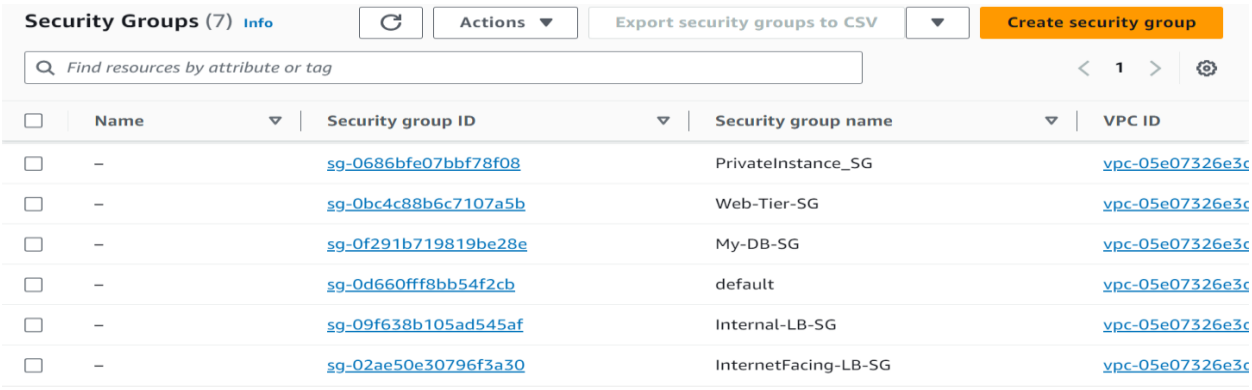
1.Create route table:-

i. First create one route table for the web layer public subnets and name it accordingly.
ii. After creating the route table, scroll down and click on the Routes tab and Edit routes.
iii. Add a route that directs traffic from the VPC to the internet gateway.
iv. Select Subnet Associations and click Edit subnet associations.
v. Select the two web layer public subnets and click Save associations.
vi. Now create 2 more route tables, one for each app layer private subnet in each AZ. These route tables will route app layer traffic destined for outside the VPC to the NAT gateway in the respective availability zone, so add the appropriate routes for that.
vii. Once the route tables are created and routes added, add the appropriate subnet associations for each of the app layer private subnets

Security Groups

1.Create Security Group :-

i. The first security group is for the public, type a name and description, add an inbound rule to allow HTTP type traffic for your IP.
ii. Create second security group is for the public instances in the web tier. Type a name and description, add an inbound rule that allows HTTP type traffic from your previously created security group. Then, add an additional rule that will allow HTTP type traffic for your IP.
iii. The third security group will be for our internal load balancer. Create new security group and add an inbound rule that allows HTTP type traffic from your public instance security group.
iv. The fourth security group is for our private instances. Type name and description, add an inbound rule that will allow TCP type traffic on port 4000 from the internal load balancer security group. You should also add another route for port 4000 that allows your IP for testing.
v. The fifth security group for protects our private database instances. Add an inbound rule that will allow traffic from the private instance security group to the MYSQL/Aurora port (3306)



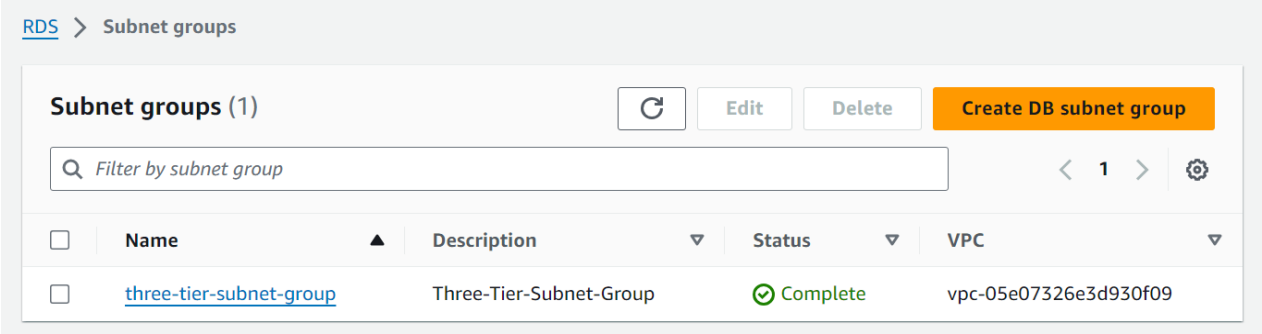
Name	Security group ID	Security group name	VPC ID
–	sg-0686bfe07bbf78f08	PrivateInstance_SG	vpc-05e07326e3c
–	sg-0bc4c88b6c7107a5b	Web-Tier-SG	vpc-05e07326e3c
–	sg-0f291b719819be28e	My-DB-SG	vpc-05e07326e3c
–	sg-0d660fff8bb54f2cb	default	vpc-05e07326e3c
–	sg-09f638b105ad545af	Internal-LB-SG	vpc-05e07326e3c
–	sg-02ae50e30796f3a30	InternetFacing-LB-SG	vpc-05e07326e3c

Database Deployment

1. Create Subnet Groups :-

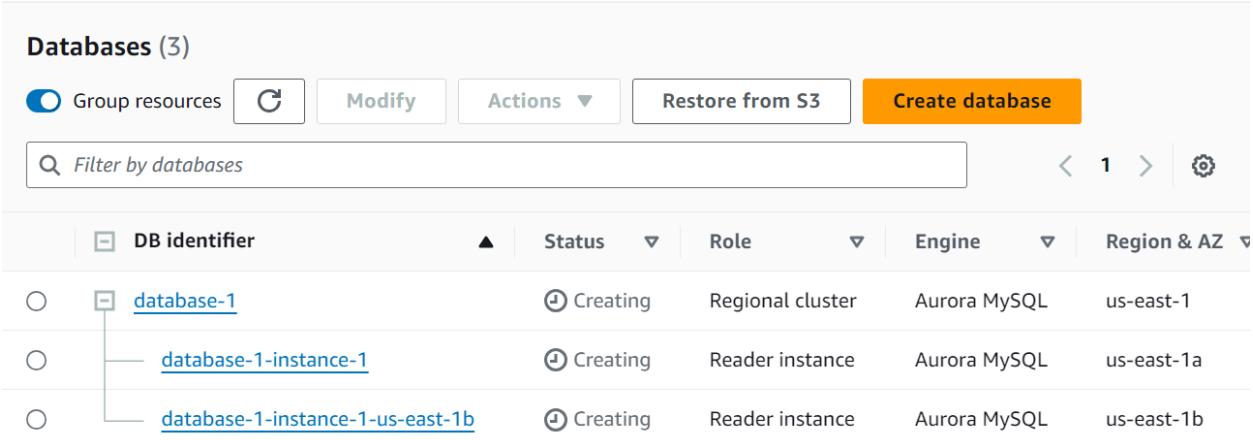
i. Click Create DB subnet group.
ii. Give your subnet group a name, description, and choose the VPC we created.

iii. Add subnets that we created for database.



2.Create Database :-

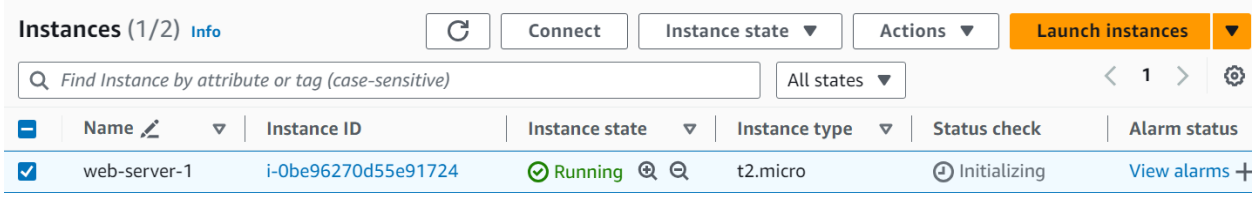
i. Navigate to RDS dashboard and click Create database.
ii. Start with a Standard create for this MySQL-Compatible Amazon Aurora database. Leave the rest of the defaults in the Engine options as default.
iii. Under the Templates section choose Dev/Test. Under Settings set a username and password.
iv. Next, under Availability and durability change the option to create an Aurora Replica or reader node in a different availability zone. Under Connectivity, set the VPC, choose the subnet group we created earlier, and select no for public access.
v. Set the security group we created for the database, make sure password authentication is selected as our authentication choice, and create the database.



App Tier Instance Deployment

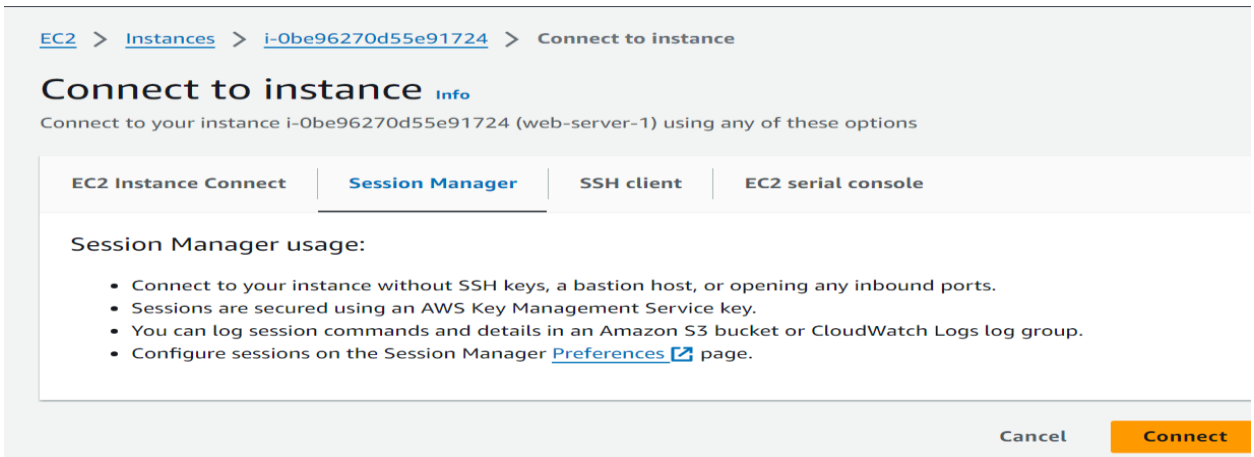
1.Create EC2 Instance :-

i. Navigate to the EC2 dashboard, click Launch Instances and add tag.
ii. Select the first Amazon Linux 2 AMI
iii. Select t.2 micro instance type.
iv. Configure Instance Details and make sure to select to correct Network, subnet, and IAM role. Note that this is the app layer, so use one of the private subnets we created for this layer.
v. Select key-pair as proceed without a keypair vi) Keep the defaults setting for storage.
vi. Select appropriate Security Group for private app tier.
vii. Click Launch



Connect to Instance

When the instance state is running, connect to your instance by clicking the checkmark box and click the connect button. Select the Session Manager tab, and click connect.



Configure Database

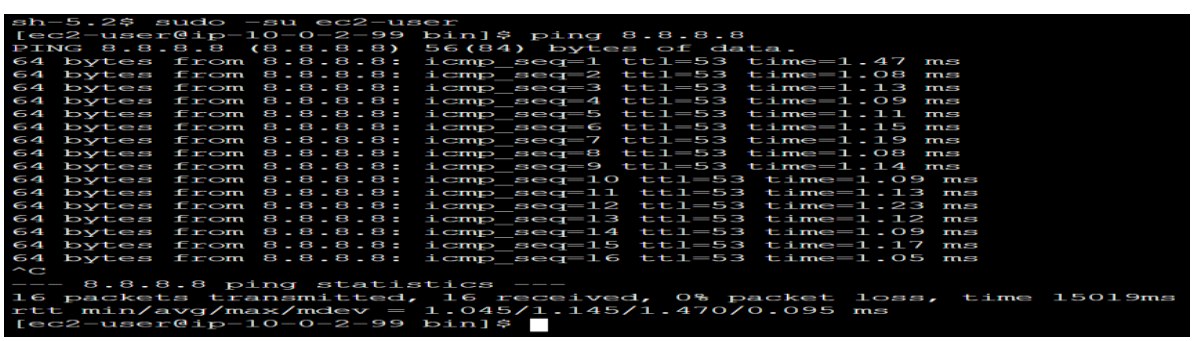
Firstly, you will be logged in as ssm-user which is the default user. Switch to ec2-user by executing the following command in the browser terminal:
sudo -su ec2-user
Then you will be loging as SSM user which is ec2 user then hit the command then try ping in 8.8.8.8 so that going to via internet or not.
Install following package for mysql server using wget command

1. Sudo wget https://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm

2. sudo rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022

3. sudo yum install https://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm

4. sudo yum install mysql -y



```
[ec2-user@ip-10-0-2-99 bin]$ sudo yum update
Last metadata expiration check: 0:10:23 ago on Fri Apr 19 06:42:46 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-10-0-2-99 bin]$ sudo wget https://repo.mysql.com/mysql57-community-release-el7.rpm
--2024-04-19 06:54:45--  https://repo.mysql.com/mysql57-community-release-el7.rpm
Resolving repo.mysql.com (repo.mysql.com)... 23.44.209.26, 2600:1408:20:c8f::1d68, 2600:1408:20:
Connecting to repo.mysql.com (repo.mysql.com)|23.44.209.26|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 25680 (25K) [application/x-redhat-package-manager]
Saving to: 'mysql57-community-release-el7.rpm'

mysql57-community-release-el7.rpm    100%[=================================================

2024-04-19 06:54:45 (3.65 MB/s) - 'mysql57-community-release-el7.rpm' saved [25680/25680]

[ec2-user@ip-10-0-2-99 bin]$ sudo yum install mysql -y
Last metadata expiration check: 0:14:08 ago on Fri Apr 19 06:42:46 2024.
No match for argument: mysql
No match for argument: -y
Error: Unable to find a match: mysql -y
[ec2-user@ip-10-0-2-99 bin]$ sudo rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022
[ec2-user@ip-10-0-2-99 bin]$ sudo yum install mysql -y
```

```
[ec2-user@ip-10-0-2-30 bin]$ sudo yum install mysql -y
MySQL Connectors Community
MySQL Tools Community
MySQL 5.7 Community Server
Dependencies resolved.
=================================================================================================
 Package                                  Architecture          Version
=================================================================================================
Installing:
 mysql-community-client                   x86_64                5.7.44-1.el7
Installing dependencies:
 mysql-community-common                   x86_64                5.7.44-1.el7
 mysql-community-libs                     x86_64                5.7.44-1.el7
 ncurses-compat-libs                      x86_64                6.2-4.20200222.amzn2023.0.6

Transaction Summary
=================================================================================================
Install  4 Packages

Total download size: 35 M
Installed size: 135 M
Downloading Packages:
(1/4): ncurses-compat-libs-6.2-4.20200222.amzn2023.0.6.x86_64.rpm
(2/4): mysql-community-common-5.7.44-1.el7.x86_64.rpm
(3/4): mysql-community-client-5.7.44-1.el7.x86_64.rpm
(4/4): mysql-community-libs-5.7.44-1.el7.x86_64.rpm
```

Initiate your DB connection with your Aurora RDS writer endpoint. In the following command, replace the RDS writer endpoint and the username, and then execute it in the browser terminal:

mysql -h CHANGE-TO-YOUR-RDS-ENDPOINT -u CHANGE-TO-USER-NAME –p

Enter password and connect to your database

```
[ec2-user@ip-10-0-2-30 bin]$ mysql -h database-1-instance-1.c3uwqae6gpa3.us-east-1.rds.amazonaws.com -u admin -p
ERROR 1045 (28000): Access denied for user 'admin'@'10.0.2.30' (using password: NO)
[ec2-user@ip-10-0-2-30 bin]$ mysql -h database-1-instance-1.c3uwqae6gpa3.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8413
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 statement.

mysql>
```

Database Operations

i. Create a database called webappdb with the following command using the MySQL CLI: CREATE DATABASE webappdb;

ii. You can verify that it was created correctly with the following command: SHOW DATABASES;

iii. Create a data table by first navigating to the database we just created: USE webappdb;

iv. Create the following transactions table by executing command: CREATE TABLE IF NOT EXISTS transactions(id INT NOT NULL AUTO_INCREMENT, amount DECIMAL(10,2), description VARCHAR(100), PRIMARY KEY(id))

v. Verify the table was created: SHOW TABLES;

vi. Insert data into table for use/testing later:

vii. INSERT INTO transactions (amount,description) VALUES ('400','groceries');

viii. Verify that your data was added by executing the following command: SELECT * FROM transactions;

ix. When finished, just type exit and hit enter to exit the MySQL client

Run the command to create an sample DATABASE.

```
+--------------------+
| 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;
+--------------------+
| Tables_in_webappdb |
+--------------------+
| transactions       |
+--------------------+
1 row in set (0.00 sec)

mysql> INSERT INTO transactions (amount,description) VALUES
    -> ('400','groceries');
```

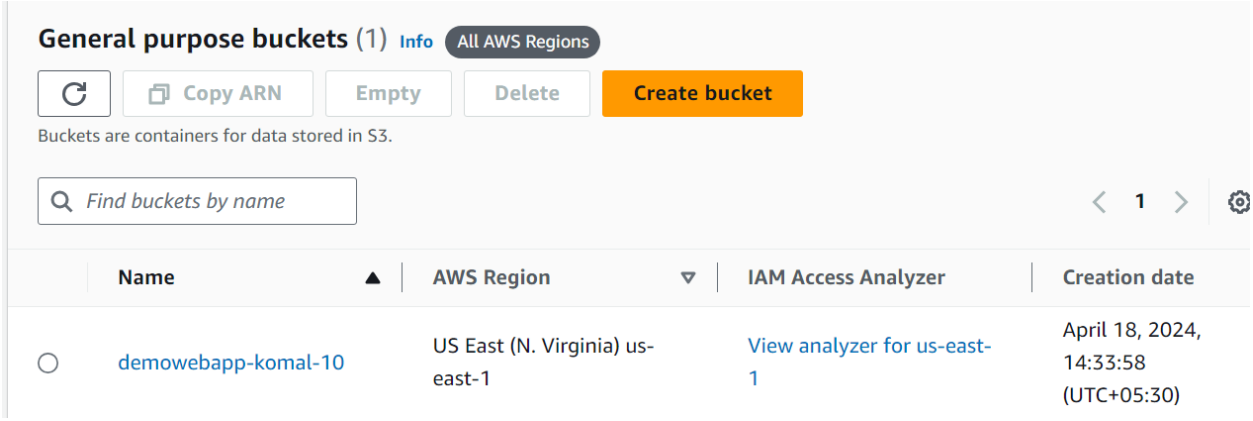
```
mysql> INSERT INTO transactions (amount,description) VALUES ('400','groceries');
Query OK, 1 row affected (0.01 sec)

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

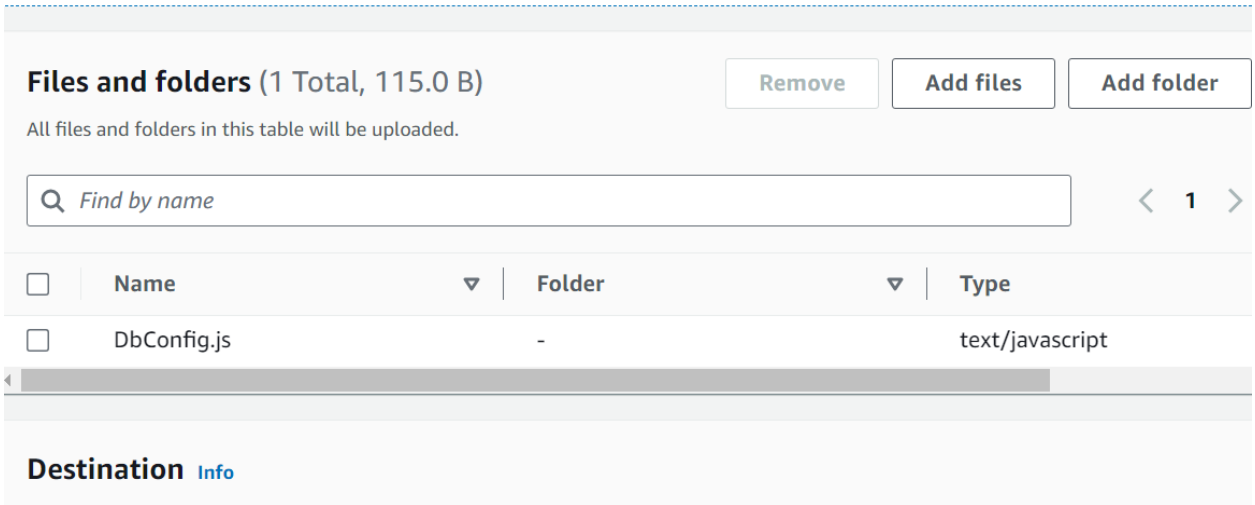
Configure App Instance

Open the application-code/app-tier/DbConfig.js file from the GitHub repo and edit for the hostname, user, password and database.

1.S3 Bucket Creation :-

i. Navigate to the S3 service and create a new S3 bucket.
ii. Give it a unique name, and then leave all the defaults as in.
iii. Upload the app-tier folder to the S3 bucket.
iv. Go back to your SSM session. Start by installing NVM (node version manager) using following command: curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash source ~/.bashrc
v. Next, install a compatible version of Node.js and make sure it's being used.
nvm install 16
nvm use 16
vi. PM2 is a daemon process manager that will keep our node.js app running when we exit the instance. npm install -g pm2





Files and folders (7 Total, 48.8 KB)

Remove | Add files | Add folder

All files and folders in this table will be uploaded.

Find by name

Name	Folder	Type
DbConfig.js	app-tier/	text/javascript
index.js	app-tier/	text/javascript
package-lock.json	app-tier/	application/json
package.json	app-tier/	application/json
README.md	app-tier/	-
TransactionService.js	app-tier/	text/javascript
DbConfig.js	-	text/javascript

Upload succeeded
View details below.

Files and folders (6 Total, 48.7 KB)

Find by name

Name	Folder	Type	Size	Status	Error
index.js	app-tier/	text/javascript	3.2 KB	Succeeded	-
package-loc...	app-tier/	application/...	42.9 KB	Succeeded	-
package.json	app-tier/	application/...	682.0 B	Succeeded	-
README.md	app-tier/	-	14.0 B	Succeeded	-
Transaction...	app-tier/	text/javascript	1.8 KB	Succeeded	-
DbConfig.js	-	text/javascript	115.0 B	Succeeded	-

Objects (2) Info

Copy S3 URI | Copy URL | Download | Open | Delete | Actions

Create folder | Upload

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Find objects by prefix

Name	Type	Last modified	Size	Storage class
app-tier/	Folder	-	-	-
DbConfig.js	js	April 19, 2024, 21:46:47 (UTC+05:30)	115.0 B	Standard

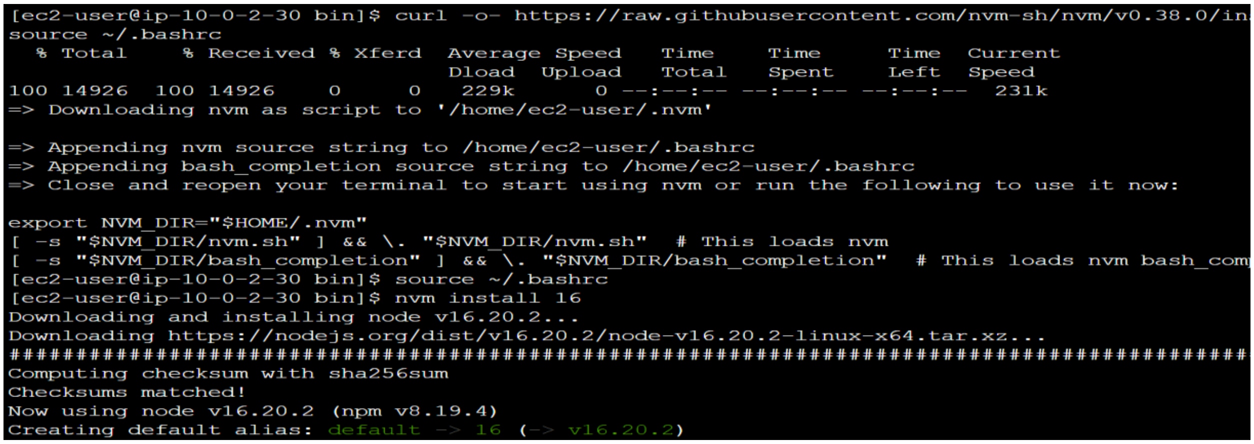
vii. Now we need to download our code from our s3 buckets onto our instance. In the command below, replace BUCKET_NAME with the name of the bucket: cd ~/ aws s3 cp s3://BUCKET_NAME/app-tier/ app-tier –recursive

viii. Navigate to the app directory, install dependencies, and start the app with pm2. cd ~/app-tier npm install pm2 start index.js

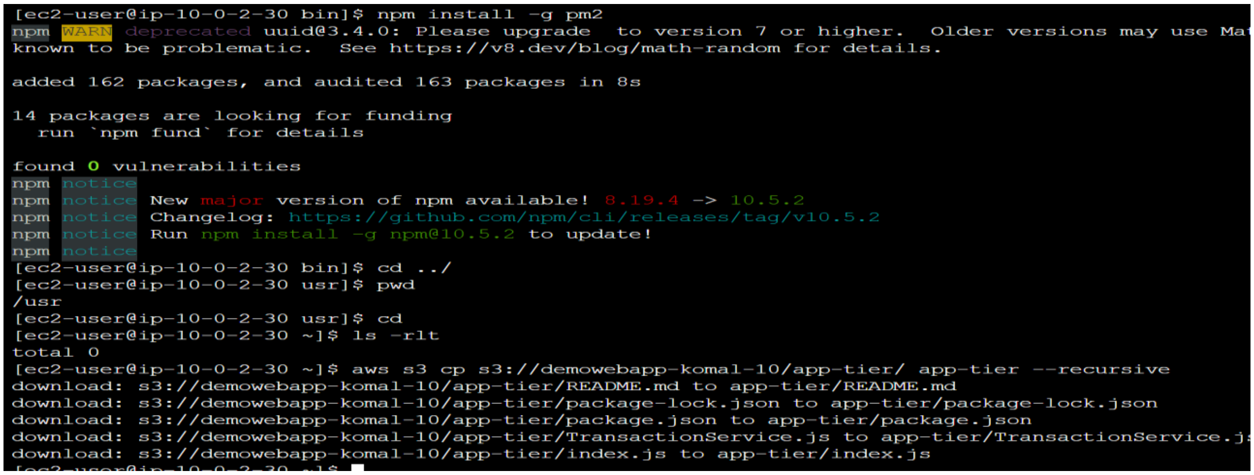
ix. To make sure the app is running correctly run the following: pm2 list To look at the latest errors, use this command: pm2 logs

x. Right now, pm2 is just making sure our app stays running when we leave the SSM session. pm2 startup

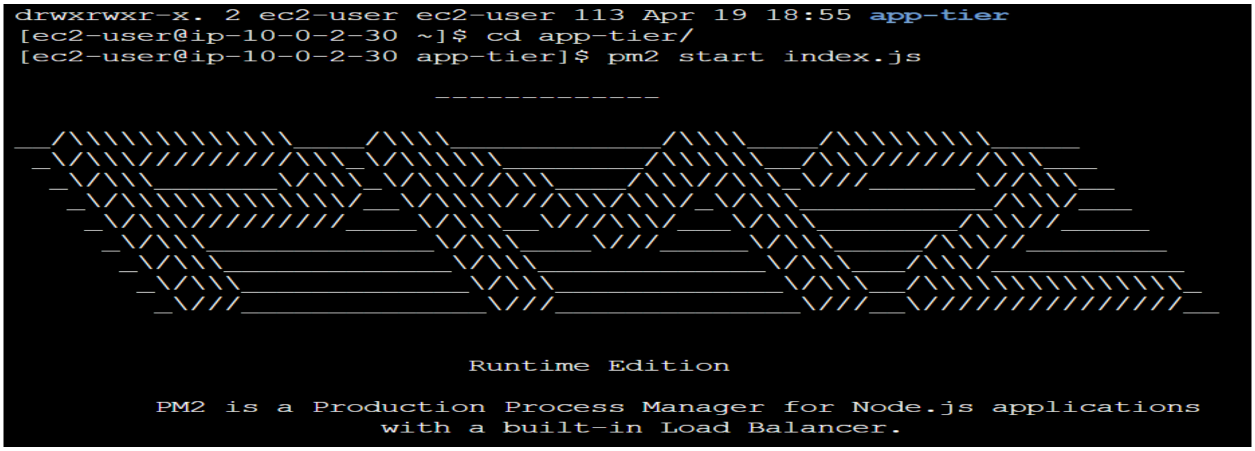
xi. Save the current list of node processes with the following command: pm2 save



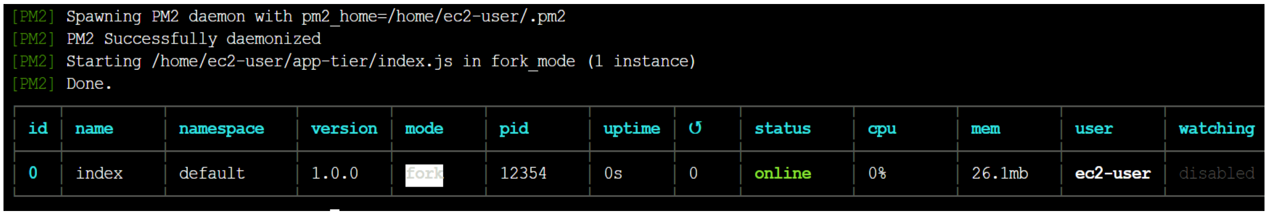
```
[ec2-user@ip-10-0-2-30 bin]$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/in
source ~/.bashrc
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14926  100 14926    0     0   229k      0 --:--:-- --:--:-- --:--:--  231k
=> 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" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash_com
[ec2-user@ip-10-0-2-30 bin]$ source ~/.bashrc
[ec2-user@ip-10-0-2-30 bin]$ 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-10-0-2-30 bin]$ npm install -g pm2
npm WARN deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  Older versions may use Ma
known to be problematic.  See https://v8.dev/blog/math-random for details.

added 162 packages, and audited 163 packages in 8s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New major version of npm available! 8.19.4 -> 10.5.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.5.2
npm notice Run npm install -g npm@10.5.2 to update!
npm notice
[ec2-user@ip-10-0-2-30 bin]$ cd ../
[ec2-user@ip-10-0-2-30 usr]$ pwd
/usr
[ec2-user@ip-10-0-2-30 usr]$ cd
[ec2-user@ip-10-0-2-30 ~]$ ls -rlt
total 0
[ec2-user@ip-10-0-2-30 ~]$ aws s3 cp s3://demowebapp-komal-10/app-tier/ app-tier --recursive
download: s3://demowebapp-komal-10/app-tier/README.md to app-tier/README.md
download: s3://demowebapp-komal-10/app-tier/package-lock.json to app-tier/package-lock.json
download: s3://demowebapp-komal-10/app-tier/package.json to app-tier/package.json
download: s3://demowebapp-komal-10/app-tier/TransactionService.js to app-tier/TransactionService.j
download: s3://demowebapp-komal-10/app-tier/index.js to app-tier/index.js
```



```
drwxrwxr-x. 2 ec2-user ec2-user 113 Apr 19 18:55 app-tier
[ec2-user@ip-10-0-2-30 ~]$ cd app-tier/
[ec2-user@ip-10-0-2-30 app-tier]$ pm2 start index.js

                        -------------

__/\\\\\\\\\\\\\____/\\\\____________/\\\\____/\\\\\\\\\_____
 _\/\\\/////////\\\_\/\\\\\\________/\\\\\\__/\\\///////\\\___
  _\/\\\_______\/\\\_\/\\\//\\\____/\\\//\\\_\///______\//\\\__
   _\/\\\\\\\\\\\\\/__\/\\\\///\\\/\\\/_\/\\\___________/\\\/___
    _\/\\\/////////____\/\\\__\///\\\/___\/\\\________/\\\//_____
     _\/\\\_____________\/\\\____\///_____\/\\\_____/\\\//________
      _\/\\\_____________\/\\\_____________\/\\\___/\\\/___________
       _\/\\\_____________\/\\\_____________\/\\\__/\\\\\\\\\\\\\\\_
        _\///______________\///______________\///__\///////////////__


                          Runtime Edition

        PM2 is a Production Process Manager for Node.js applications
                     with a built-in Load Balancer.
```



```
[PM2] Spawning PM2 daemon with pm2_home=/home/ec2-user/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /home/ec2-user/app-tier/index.js in fork_mode (1 instance)
[PM2] Done.
┌────┬──────────┬─────────────┬─────────┬─────────┬──────────┬────────┬──────┬───────────┬──────────┬──────────┬──────────┬──────────
│ id │ name     │ namespace   │ version │ mode    │ pid      │ uptime │ ↺    │ status    │ cpu      │ mem      │ user     │ watching
├────┼──────────┼─────────────┼─────────┼─────────┼──────────┼────────┼──────┼───────────┼──────────┼──────────┼──────────┼──────────
│ 0  │ index    │ default     │ 1.0.0   │ fork    │ 12354    │ 0s     │ 0    │ online    │ 0%       │ 26.1mb   │ ec2-user │ disabled
└────┴──────────┴─────────────┴─────────┴─────────┴──────────┴────────┴──────┴───────────┴──────────┴──────────┴──────────┴──────────
```

```
Target path
/etc/systemd/system/pm2-ec2-user.service
Command list
[ 'systemctl enable pm2-ec2-user' ]
[PM2] Writing init configuration in /etc/systemd/system/pm2-ec2-user.service
[PM2] Making script booting at startup...
[PM2] [-] Executing: systemctl enable pm2-ec2-user...
Created symlink /etc/systemd/system/multi-user.target.wants/pm2-ec2-user.service → /etc
[PM2] [v] Command successfully executed.
+---------------------------------------+
[PM2] Freeze a process list on reboot via:
$ pm2 save

[PM2] Remove init script via:
$ pm2 unstartup systemd
[ec2-user@ip-10-0-2-30 app-tier]$ pm2 save
[PM2] Saving current process list...
[PM2] Successfully saved in /home/ec2-user/.pm2/dump.pm2
```

Test App Tier

i) Run following command to simple health check endpoint that tells us if the app is simply running: curl http://localhost:4000/health

ii) Next, test your database connection: curl http://localhost:4000/transaction

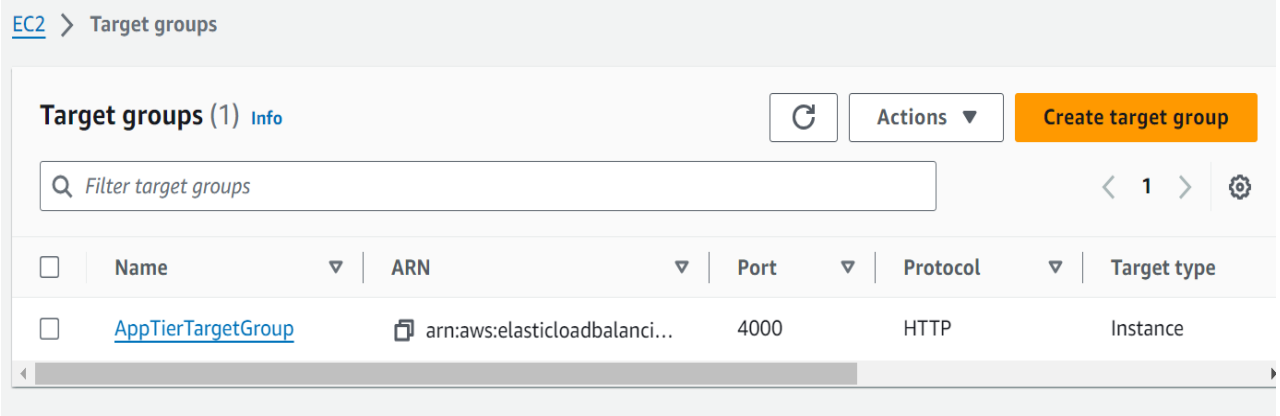
If these both commands shows appropriate output then proceed with further process.

```
PM2        | 2024-04-22T12:23:39: PM2 log: Application log path : /h
PM2        | 2024-04-22T12:23:39: PM2 log: Worker Interval      : 30
PM2        | 2024-04-22T12:23:39: PM2 log: Process dump file    : /h
PM2        | 2024-04-22T12:23:39: PM2 log: Concurrent actions   : 2
PM2        | 2024-04-22T12:23:39: PM2 log: SIGTERM timeout      : 16
PM2        | 2024-04-22T12:23:39: PM2 log: ========================
PM2        | 2024-04-22T12:23:39: PM2 log: App [index:0] starting in
PM2        | 2024-04-22T12:23:39: PM2 log: App [index:0] online

/home/ec2-user/.pm2/logs/index-error.log last 15 lines:
/home/ec2-user/.pm2/logs/index-out.log last 15 lines:
0|index    | AB3 backend app listening at http://localhost:4000
```

```
Target path
/etc/systemd/system/pm2-ec2-user.service
Command list
[ 'systemctl enable pm2-ec2-user' ]
[PM2] Writing init configuration in /etc/systemd/system/pm2-ec2-user.serv
[PM2] Making script booting at startup...
[PM2] [-] Executing: systemctl enable pm2-ec2-user...
Created symlink /etc/systemd/system/multi-user.target.wants/pm2-ec2-user.
[PM2] [v] Command successfully executed.
+---------------------------------------+
[PM2] Freeze a process list on reboot via:
$ pm2 save

[PM2] Remove init script via:
$ pm2 unstartup systemd
[ec2-user@ip-10-0-2-192 app-tier]$ pm2 save
[PM2] Saving current process list...
[PM2] Successfully saved in /home/ec2-user/.pm2/dump.pm2
[ec2-user@ip-10-0-2-192 app-tier]$ curl http://localhost:4000/health
"This is the health check"[ec2-usercurl http://localhost:4000/transactior
```

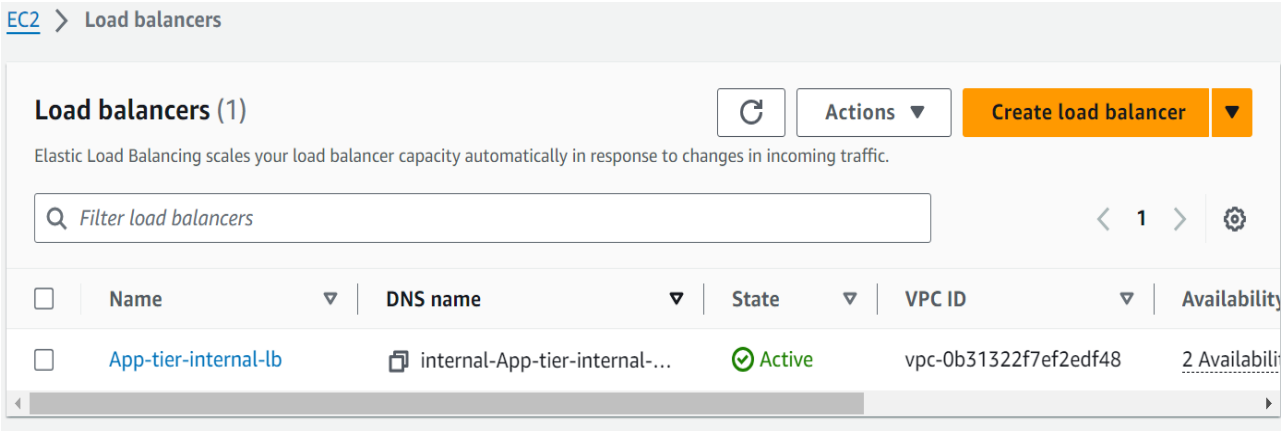
i. EC2 dashboard. Select the app tier instance we created and under Actions select Image and templates. Click Create Image.

ii. While the AMI is being created, we can create our target group to use with the load balancer. On the EC2 dashboard navigate to Target Groups under Load Balancing on the left hand side. Click on Create Target Group.

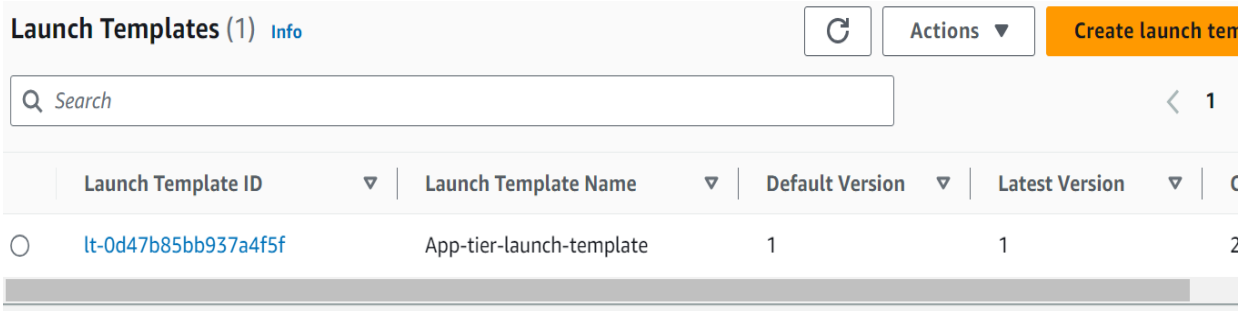


On EC2 dashboard select Load Balancers under Load Balancing and click Create Load Balancer.

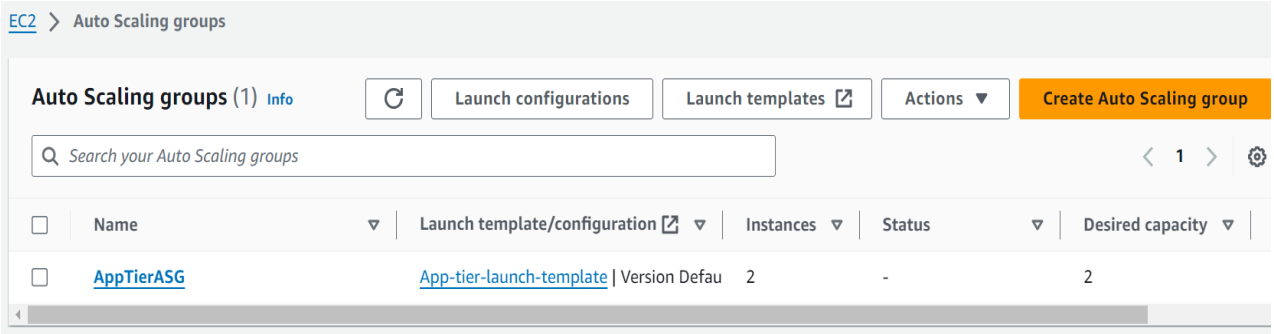
iii. Application Load Balancer is for our HTTP traffic so click the create button for that option.
iv. After giving the load balancer a name, be sure to select internal since this one will not be public facing, but rather it will route traffic from our web tier to the app tier.
v. Select the correct network configuration for VPC and private subnets.
vi. Select the security group we created for this internal ALB. Now, this ALB will be listening for HTTP traffic on port 80. It will be forwarding the traffic to our target group that we just created, so select it from the dropdown, and create the load balancer

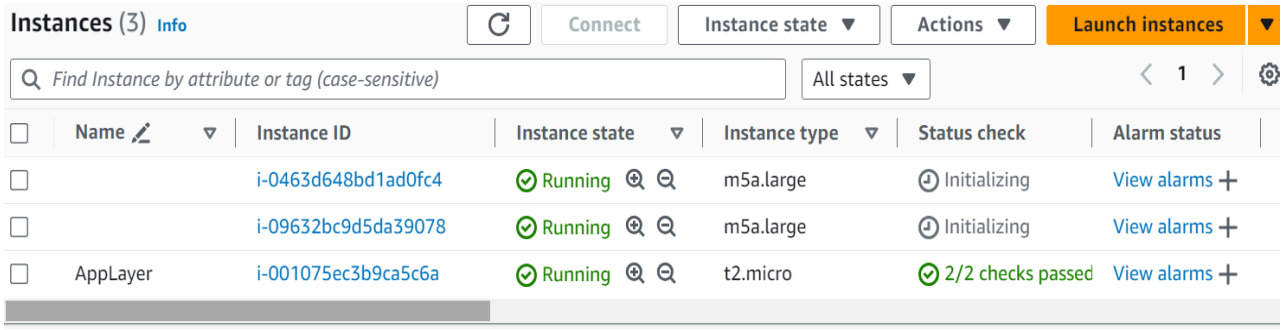


Before we configure Auto Scaling, we need to create a Launch template with the AMI. Name the Launch Template, and then under Application and OS Images include the app tier AMI you created.



EC2 dashboard navigate to Auto Scaling Groups under Auto Scaling and click Create Auto Scaling group.





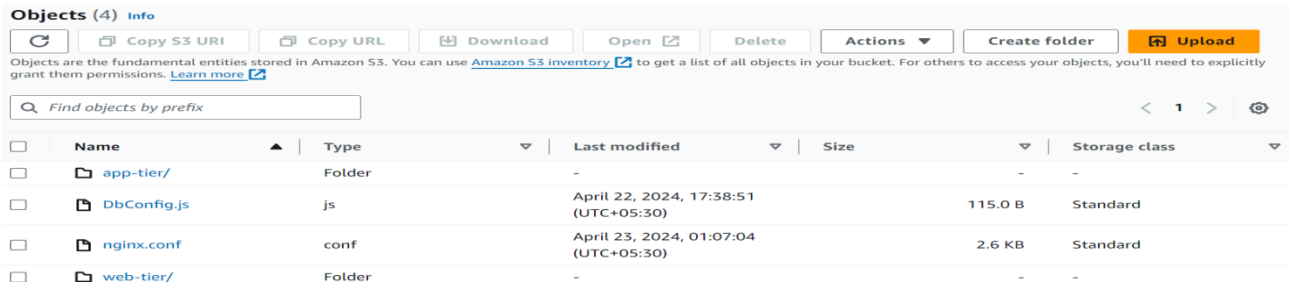
Web Tier Instance Deployment

App Tier Instance Deployment, with the exception of the subnet. We will be provisioning this instance in one of our public subnets. Make sure to select the correct network components, security group, and IAM role. This time, auto-assign a public ip on the Configure Instance Details page. Remember to tag the instance with a name so we can identify it more easily

Then connect to Instance, sudo -su ec2-user and ping 8.8.8.8

After that configure web Instance…

1. Start by installing NVM and node : curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash source ~/.bashrc
   nvm install 16
   nvm use 16
2. Now we need to download our web tier code from our s3 bucket: cd~/ aws s3 cp s3://BUCKET_NAME/web-tier/ web-tier –recursive
3. Navigate to the web-tier folder and create the build folder for the react app: cd ~/web-tier npm install npm run build
4. NGINX can be used for different use cases like load balancing, content caching etc, but we will be using it as a web server that we will configure to serve our application on port 80, as well as help direct our API calls to the internal load balancer. sudo amazon-linux-extras install nginx1 –y
   And then configure web Instance
5. Navigate to the Nginx configuration file with the following commands and list the files in the directory: cd /etc/nginx… ls
6. Then, restart Nginx with the following command: sudo service nginx restart
7. To make sure Nginx has permission to access our files execute this command: chmod -R 755 /home/ec2-user 8. And then to make sure the service starts on boot, run this command: sudo chkconfig nginx on



To test if your entire architecture is working, navigate to your external facing loadbalancer, and plug in the DNS name into your browser amd then hit enter.

It will display your web page by performing appropriate function.

```
download: s3://demowebapp-komal-10/web-tier/src/App.test.js to web-tier/src/App.test.js
download: s3://demowebapp-komal-10/web-tier/src/.DS_Store to web-tier/src/.DS_Store
download: s3://demowebapp-komal-10/web-tier/src/components/Menu/index.js to web-tier/src/compone
download: s3://demowebapp-komal-10/web-tier/src/App.css to web-tier/src/App.css
download: s3://demowebapp-komal-10/web-tier/src/assets/3TierArch.png to web-tier/src/assets/3Ti
download: s3://demowebapp-komal-10/web-tier/src/components/Burger/Burger.styled.js to web-tier/
download: s3://demowebapp-komal-10/web-tier/src/index.css to web-tier/src/index.css
download: s3://demowebapp-komal-10/web-tier/src/components/index.js to web-tier/src/components/
download: s3://demowebapp-komal-10/web-tier/src/theme.js to web-tier/src/theme.js
download: s3://demowebapp-komal-10/web-tier/src/index.js to web-tier/src/index.js
download: s3://demowebapp-komal-10/web-tier/src/setupTests.js to web-tier/src/setupTests.js
download: s3://demowebapp-komal-10/web-tier/src/reportWebVitals.js to web-tier/src/reportWebVita
download: s3://demowebapp-komal-10/web-tier/src/global.js to web-tier/src/global.js
download: s3://demowebapp-komal-10/web-tier/src/hooks.js to web-tier/src/hooks.js
download: s3://demowebapp-komal-10/web-tier/src/components/Menu/Menu.styled.js to web-tier/src/
download: s3://demowebapp-komal-10/web-tier/src/App.js to web-tier/src/App.js
```

```
264 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
npm notice
npm notice New major version of npm available! 8.19.4 -> 10.5.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.5.2
npm notice Run npm install -g npm@10.5.2 to update!
npm notice
[ec2-user@ip-10-0-0-129 web-tier]$
```

```
Compiled successfully.

File sizes after gzip:

  74.87 kB  build/static/js/main.0f3160bf.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
```

```
Complete!
[ec2-user@ip-10-0-0-129 web-tier]$ cd /etc/nginx/
[ec2-user@ip-10-0-0-129 nginx]$ ls -lrt
total 96
-rw-r--r--. 1 root root 35272 Feb  1  2023 mime.types
-rw-r--r--. 1 root root  2305 Oct 13  2023 nginx.conf
-rw-r--r--. 1 root root  3610 Oct 13  2023 win-utf
-rw-r--r--. 1 root root   664 Oct 13  2023 uwsgi_params.default
-rw-r--r--. 1 root root   664 Oct 13  2023 uwsgi_params
-rw-r--r--. 1 root root   636 Oct 13  2023 scgi_params.default
-rw-r--r--. 1 root root   636 Oct 13  2023 scgi_params
-rw-r--r--. 1 root root  2656 Oct 13  2023 nginx.conf.default
-rw-r--r--. 1 root root  5349 Oct 13  2023 mime.types.default
-rw-r--r--. 1 root root  2223 Oct 13  2023 koi-win
-rw-r--r--. 1 root root  2837 Oct 13  2023 koi-utf
-rw-r--r--. 1 root root  1007 Oct 13  2023 fastcgi_params.default
-rw-r--r--. 1 root root  1007 Oct 13  2023 fastcgi_params
-rw-r--r--. 1 root root  1077 Oct 13  2023 fastcgi.conf.default
-rw-r--r--. 1 root root  1077 Oct 13  2023 fastcgi.conf
drwxr-xr-x. 2 root root     6 Oct 13  2023 default.d
drwxr-xr-x. 2 root root     6 Oct 13  2023 conf.d
[ec2-user@ip-10-0-0-129 nginx]$ sudo cp nginx.conf nginx.conf_bkp
[ec2-user@ip-10-0-0-129 nginx]$ aws s3 cp s3://demowebapp-komal-10/nginx.conf .
download failed: s3://demowebapp-komal-10/nginx.conf to ./nginx.conf [Errno 13]
[ec2-user@ip-10-0-0-129 nginx]$ vi nginx.conf
[ec2-user@ip-10-0-0-129 nginx]$ sudo service nginx restart
Redirecting to /bin/systemctl restart nginx.service
```