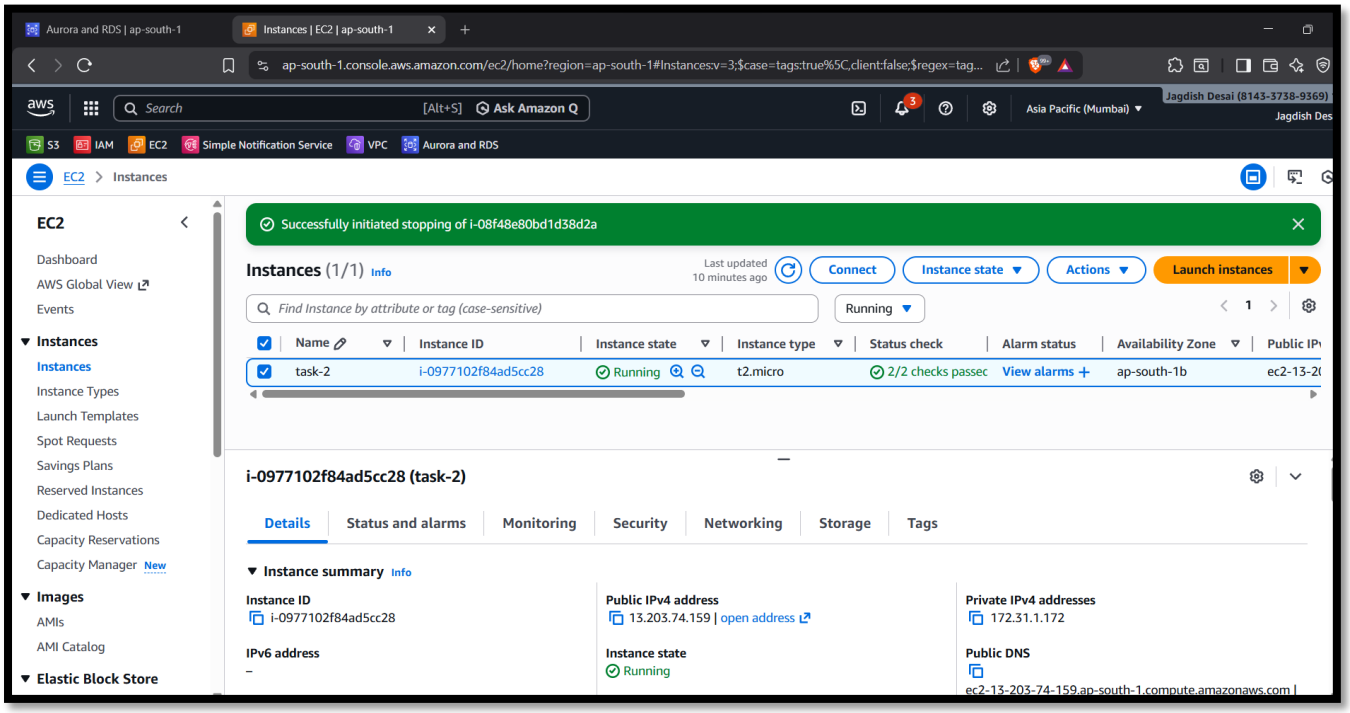
2. Docker

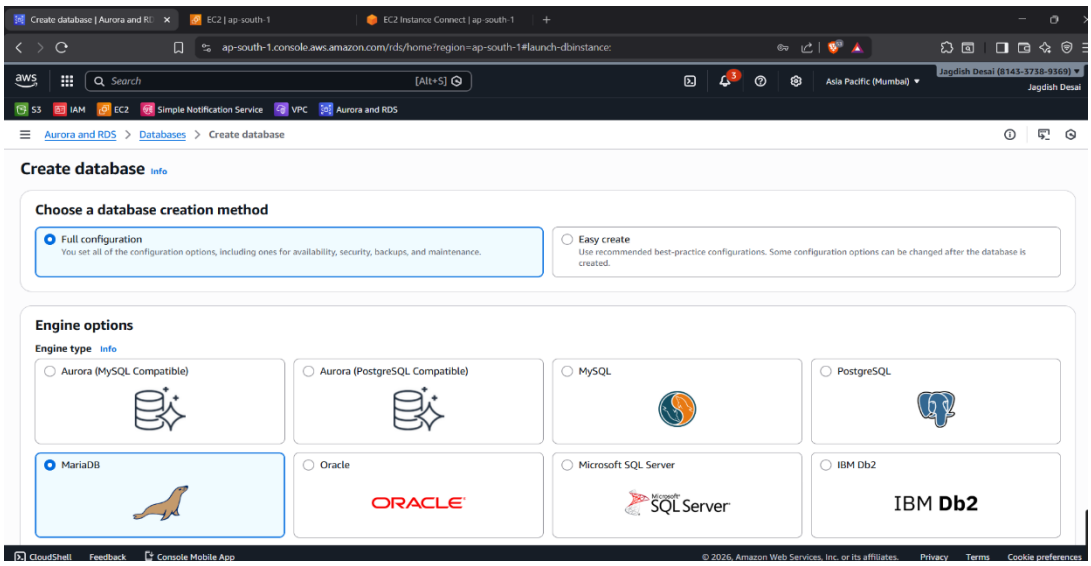
- ☐ Create Docker file(s)
- ☐ Run app using Docker containers
- ☐ Expose required ports
- ☐ Ensure containers auto-start on reboot

Steps:

Step 1: Launching a ec2 instance. Task-2

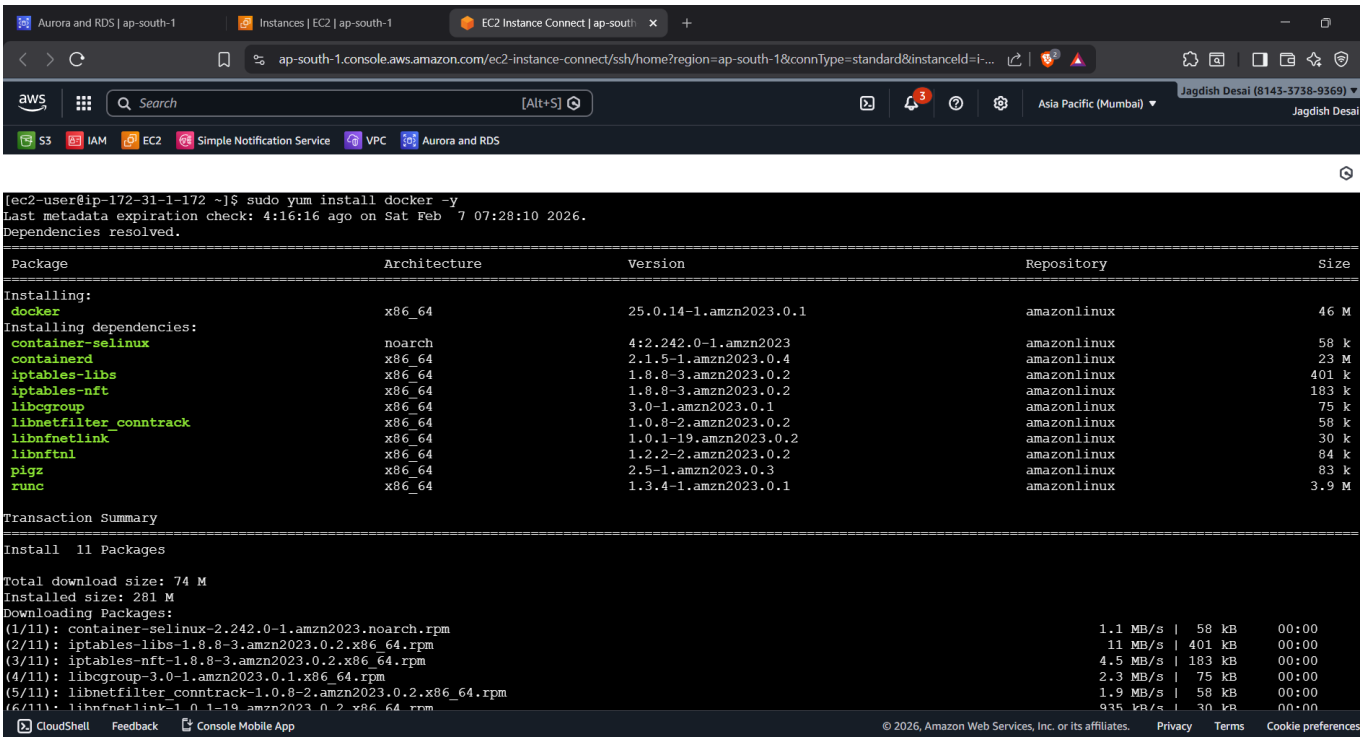


Creating Database using RDS:



Step 2:Installing Docker on task-2 instance

sudo yum install docker -y



After installing docker start docker using command:

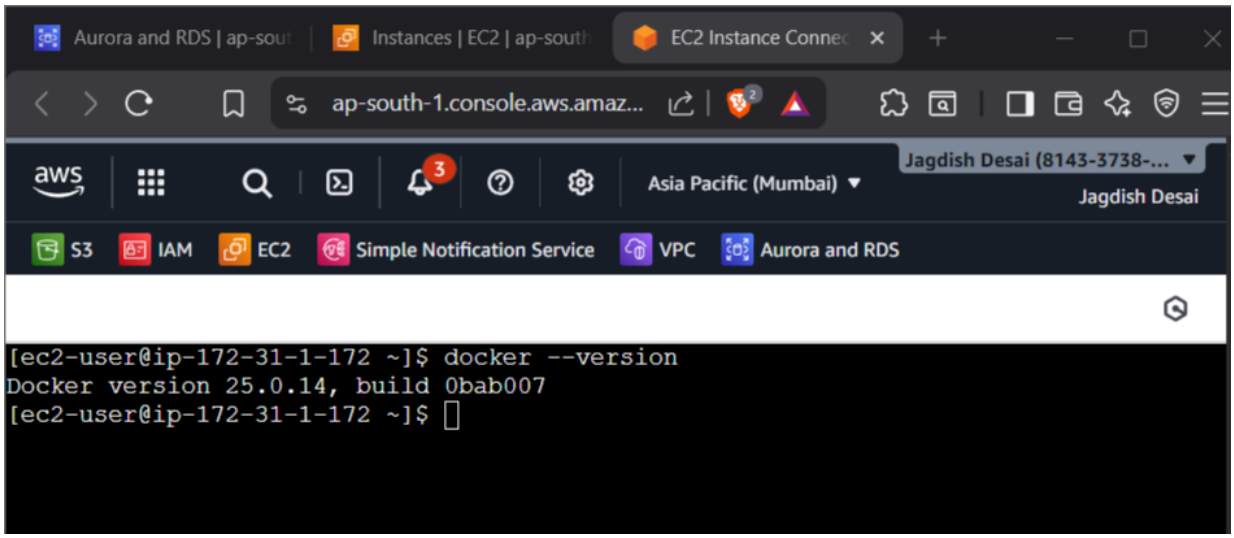
sudo systemctl start docker

After starting docker enable docker using command:

sudo systemctl enable docker

After that verify docker:

docker --version



Step 4 : Then starting creating our project by creating app directory.

Create Project Directory

mkdir app

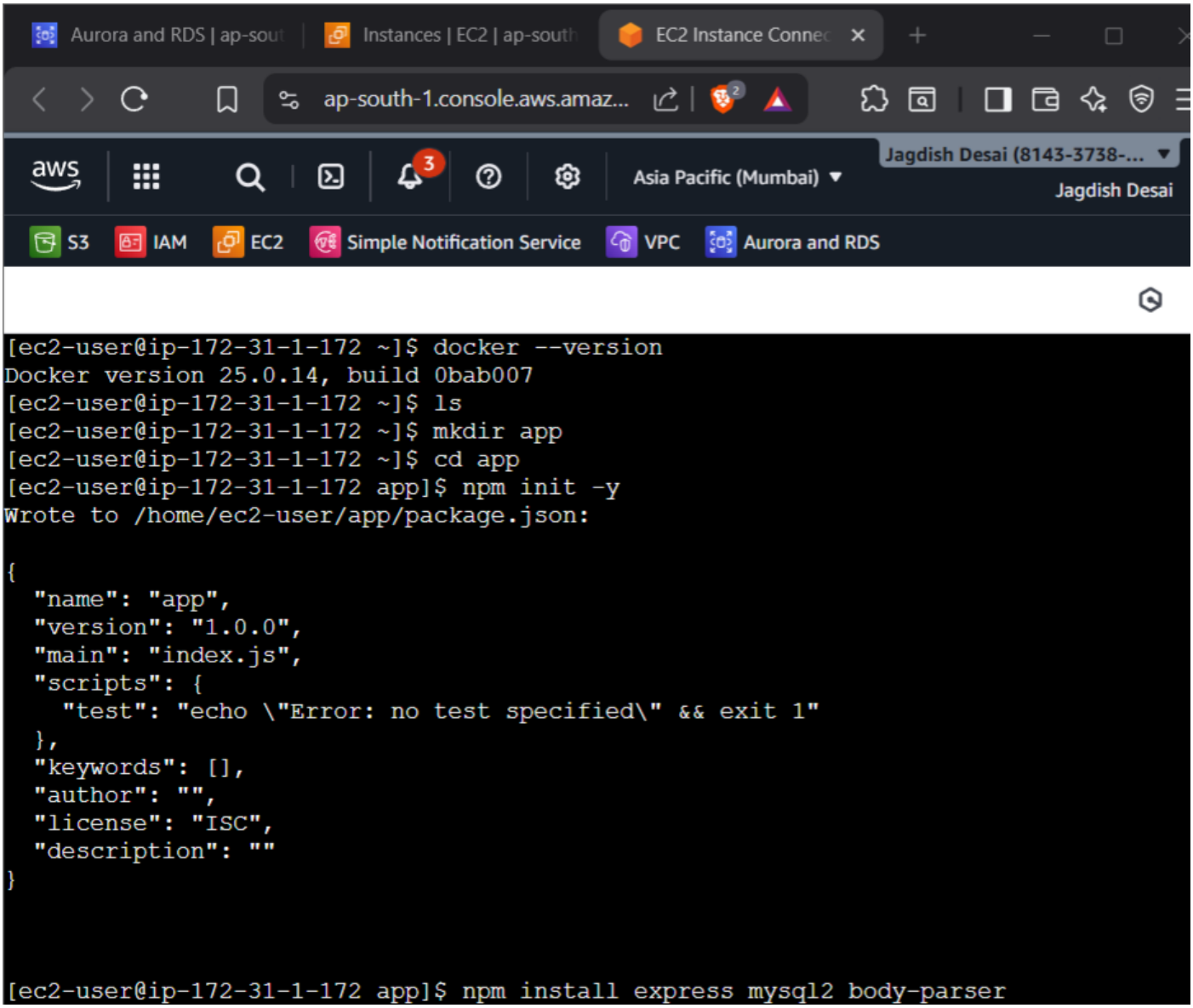
cd app

Create package.json

npm init -y

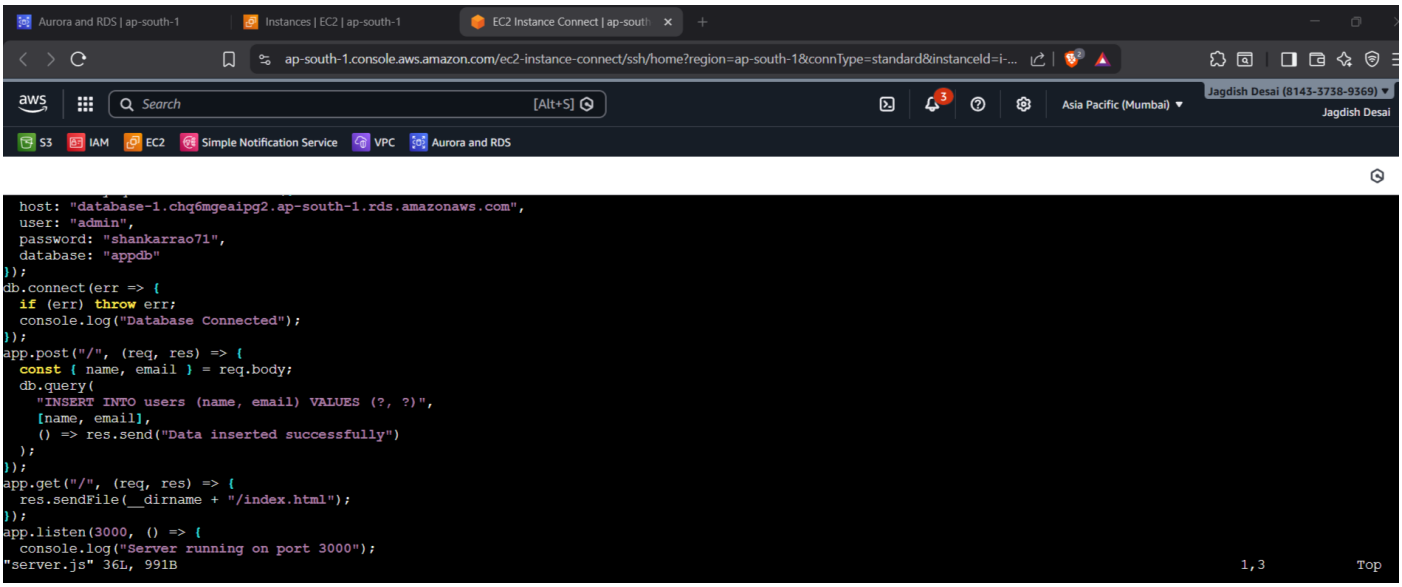
Install Required Packages

npm install express mysql2 body-parser



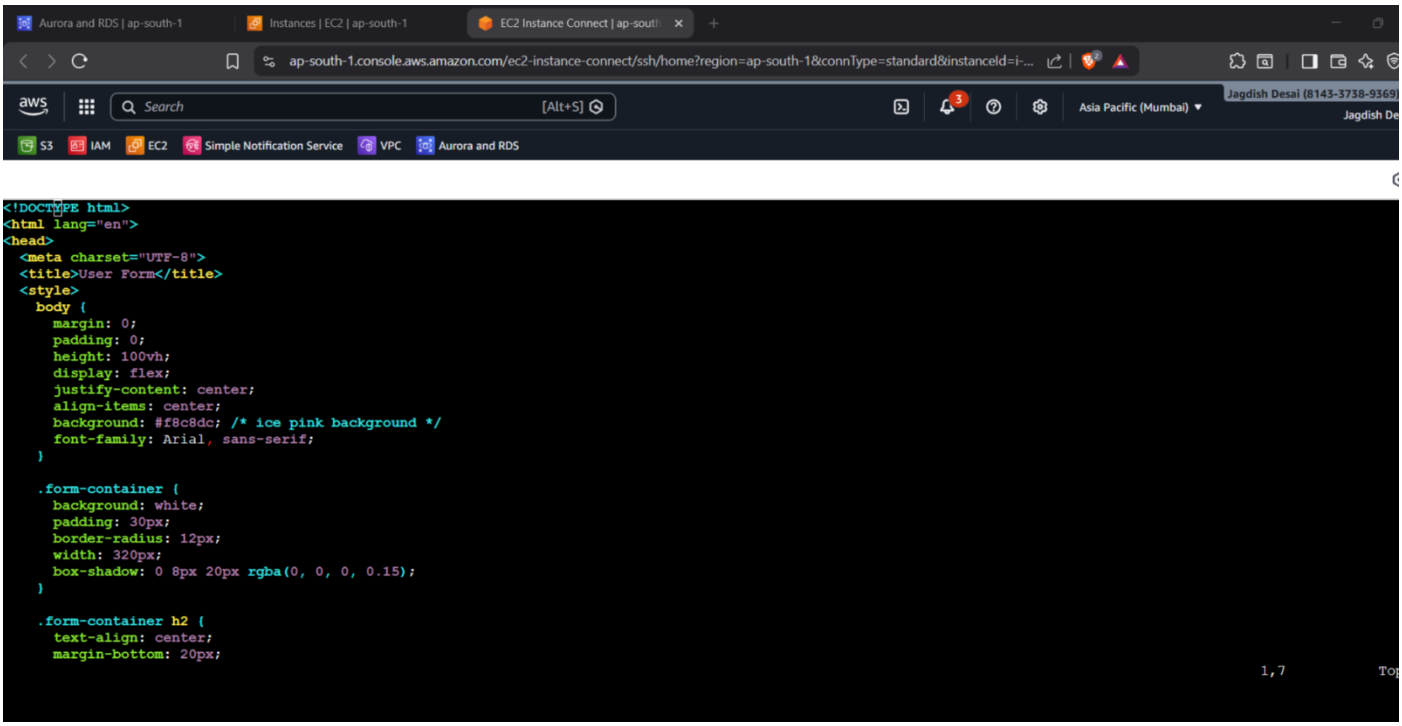
- **Create server.js:**

vi server.js



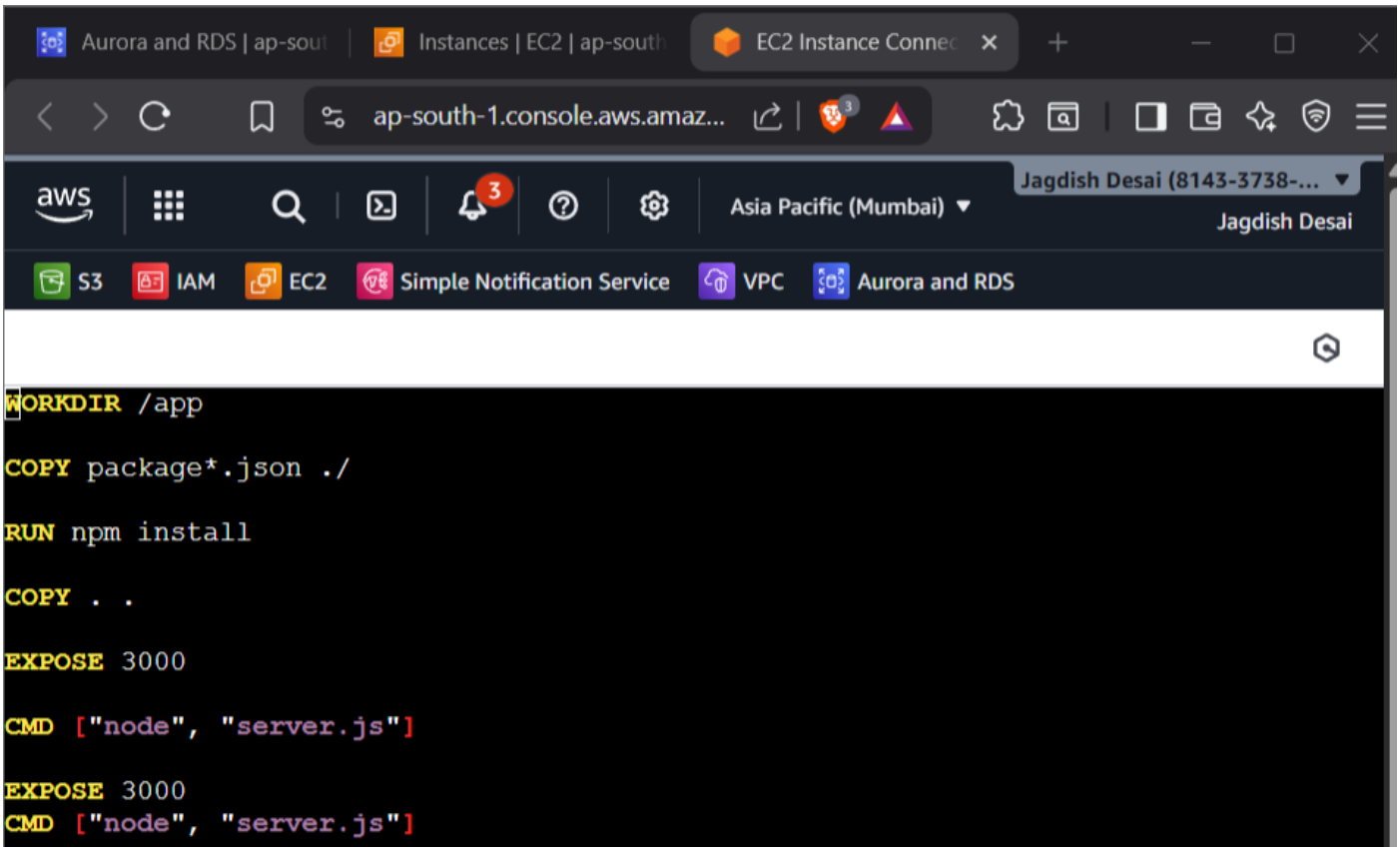
- **Creating index.html:**

vi index.html



- **Creating Dockerfile**

  vi dockerfile

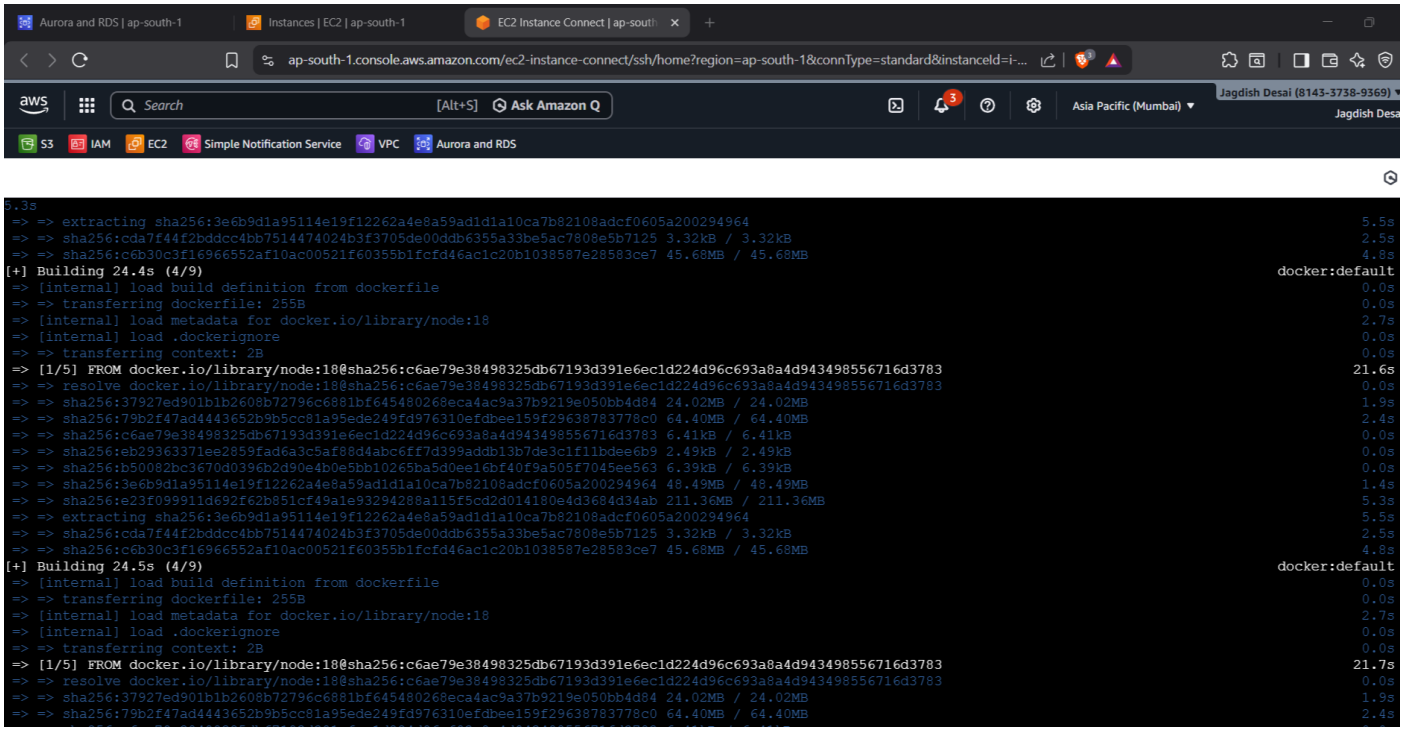


Dockerfile does (simple):

- Uses Node.js image
- Copies application files
- Installs dependencies
- Exposes port 3000
- Starts Node.js server

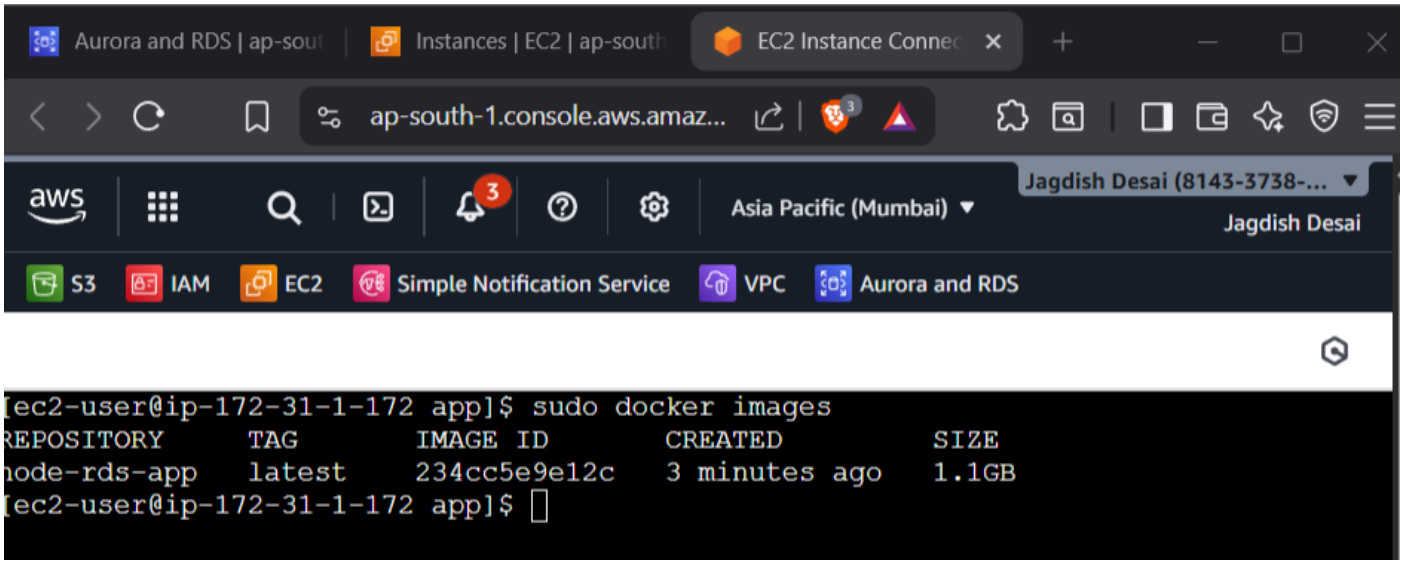
Step 5: Building Docker Image

docker build -t node-rds-app .

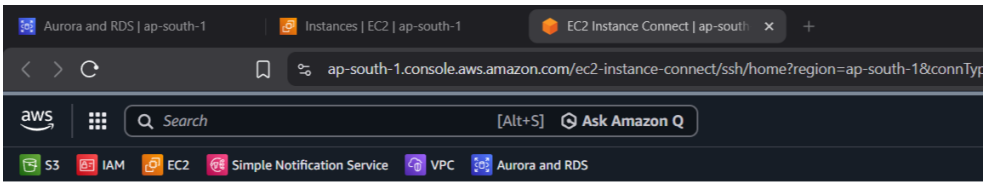


Check image:

docker images



- **Running Application Using Docker Container**

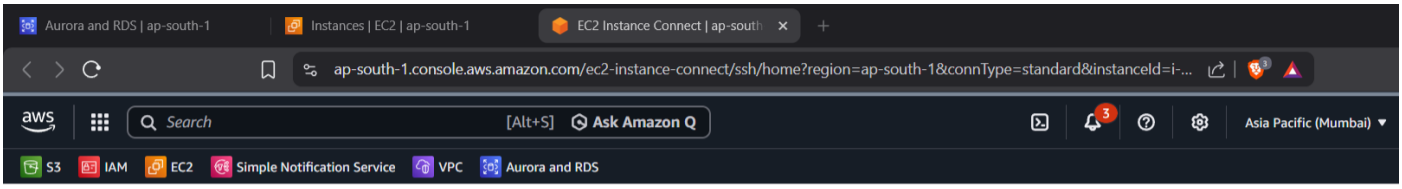



```
[ec2-user@ip-172-31-1-172 app]$ sudo docker run -d \
  -p 3000:3000 \
  --name node-app \
  node-rds-app
e34ee259d528cd25350dcff8af3e7d65ca70a814f922324844f8f143804dfae0
[ec2-user@ip-172-31-1-172 app]$
```

  - -d → run in background
  - -p 3000:3000 → expose port
  - --name → container name

- **Verifying Container is Running**

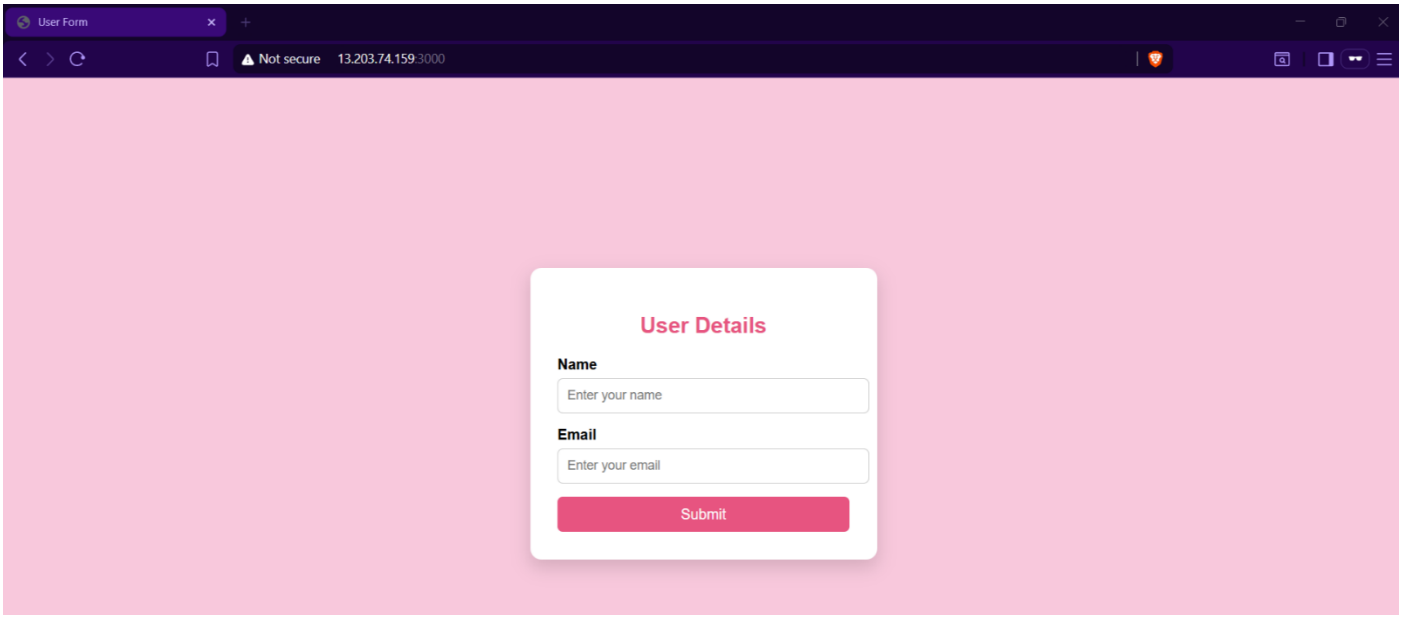
  docker ps




```
[ec2-user@ip-172-31-1-172 app]$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                                       NAMES
e34ee259d528   node-rds-app   "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   node-app
[ec2-user@ip-172-31-1-172 app]$
```

Step 6: Access Application using public ip

http://13.203.74.159:3000/



Step 7: Start Database on task2 instance

Allow port 3306 mysql port in security groups inbound rules.

systemctl start mariadb.service

sudo systemctl enable mariadb.service

sudo systemctl status mariadb.service

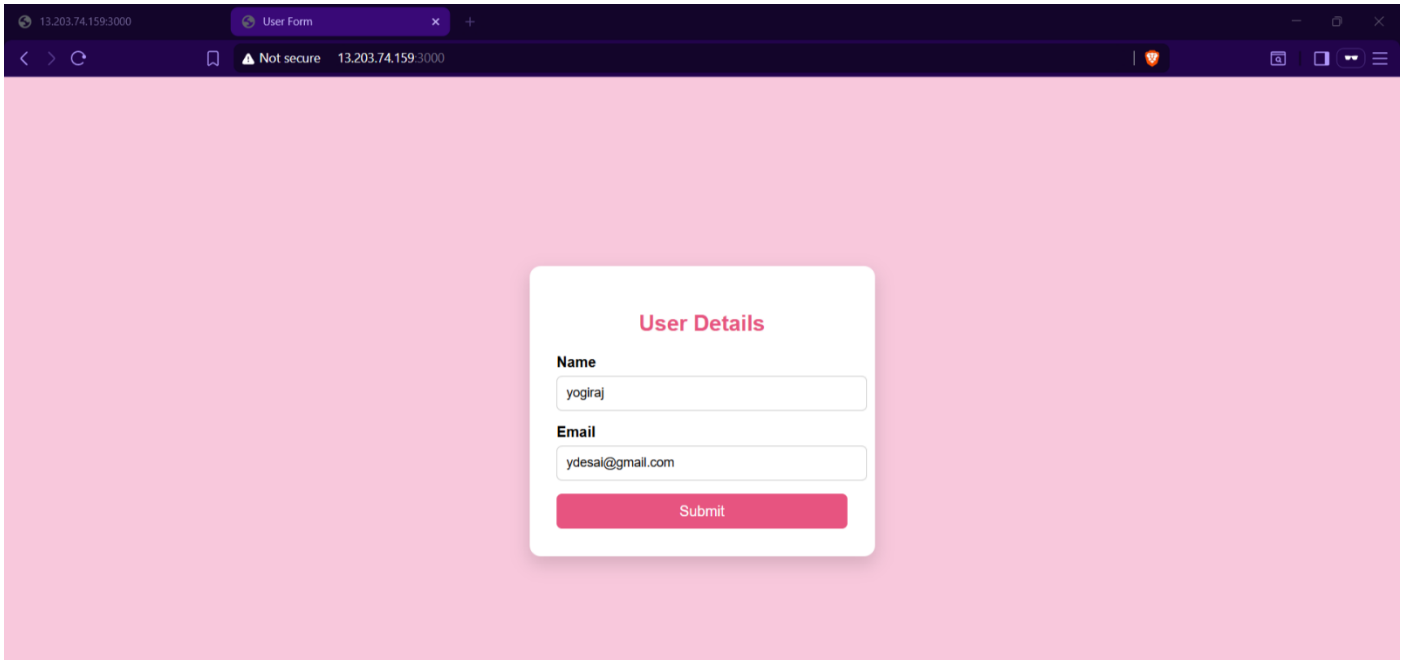
```
[ec2-user@ip-172-31-1-172 app]$ systemctl start mariadb.service
Failed to start mariadb.service: Access denied
See system logs and 'systemctl status mariadb.service' for details.
[ec2-user@ip-172-31-1-172 app]$ sudo !!
sudo systemctl start mariadb.service
[ec2-user@ip-172-31-1-172 app]$ sudo systemctl enable mariadb.service
[ec2-user@ip-172-31-1-172 app]$ sudo systemctl status mariadb.service
● mariadb.service - MariaDB 10.5 database server
     Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset: disabled)
     Active: active (running) since Sat 2026-02-07 08:30:53 UTC; 4h 27min ago
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
   Main PID: 2060 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 16 (limit: 1120)
     Memory: 65.5M
        CPU: 1.839s
     CGroup: /system.slice/mariadb.service
             └─2060 /usr/libexec/mariadbd --basedir=/usr

Feb 07 08:30:52 ip-172-31-1-172.ap-south-1.compute.internal systemd[1]: Starting mariadb.service - MariaDB 10.5 database server...
Feb 07 08:30:52 ip-172-31-1-172.ap-south-1.compute.internal mariadb-prepare-db-dir[2025]: Database MariaDB is probably initialized in /var/lib/mysql already, nothing >
Feb 07 08:30:52 ip-172-31-1-172.ap-south-1.compute.internal mariadb-prepare-db-dir[2025]: If this is not the case, make sure the /var/lib/mysql is empty before runnin>
Feb 07 08:30:53 ip-172-31-1-172.ap-south-1.compute.internal systemd[1]: Started mariadb.service - MariaDB 10.5 database server.
lines 1-17/17 (END)
```

create table users for storing user details:

```
CREATE TABLE users ( id INT AUTO_INCREMENT PRIMARY KEY,
                     name VARCHAR(100),
                     email VARCHAR(100) );
```

Step 8: Submit the above user form

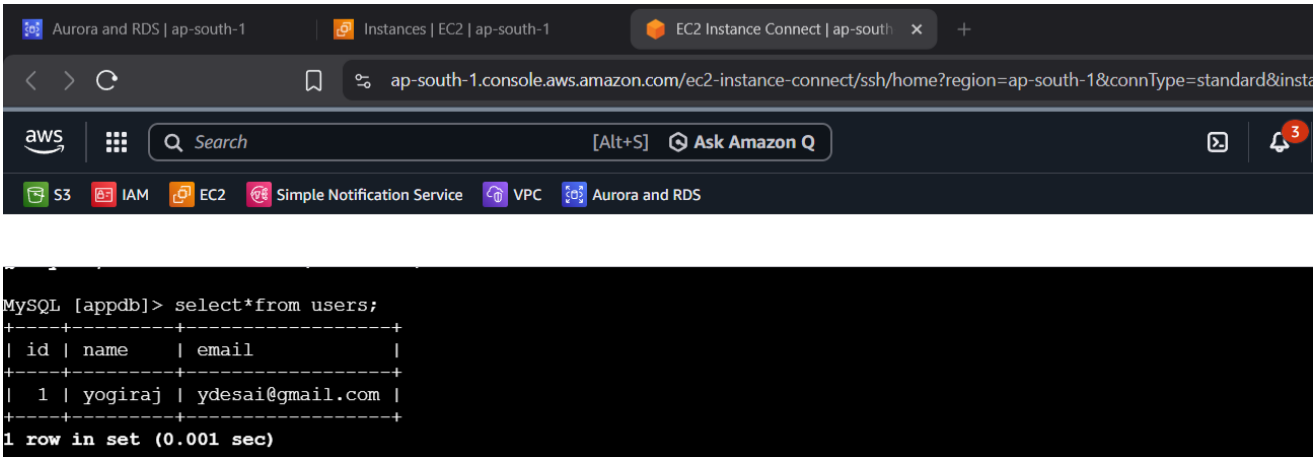


After inserting user details : data goes to our existing MySQL RDS appdb

Using command:

mysql -h database-1.chq6mgeaipg2.ap-south-1.rds.amazonaws.com -u admin -p

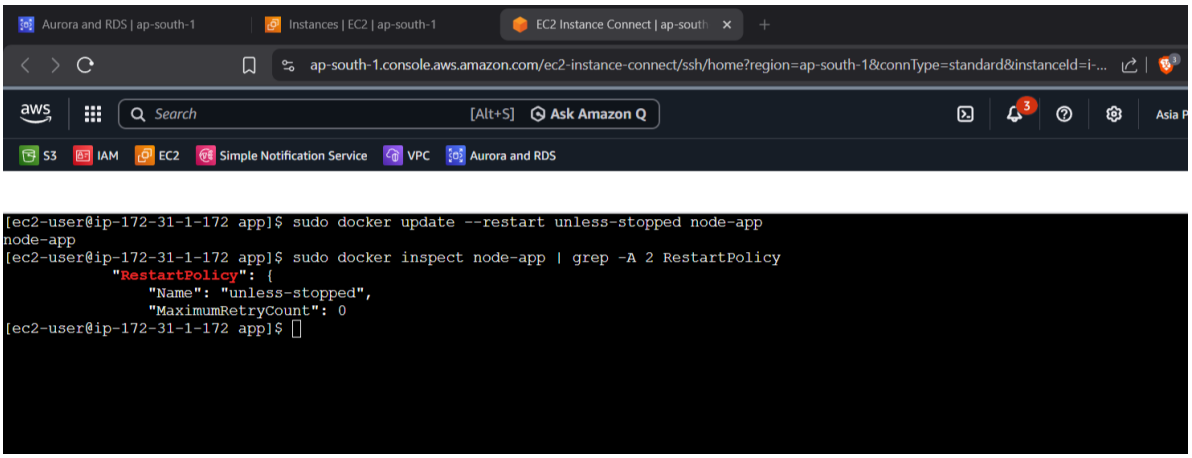
Select*from users;



Step 9: Ensuring Container Auto-Starts on Reboot

docker update --restart unless-stopped node-app

It tells Docker to automatically restart your container (node-app) whenever the EC2 instance reboots or the container stops unexpectedly, unless you explicitly stop it.



Then;

sudo docker inspect node-app | grep -A 2 RestartPolicy

It lets you **check the restart policy** of your container to confirm whether it will automatically restart after a reboot or on failure.

Container **automatically starts** when:

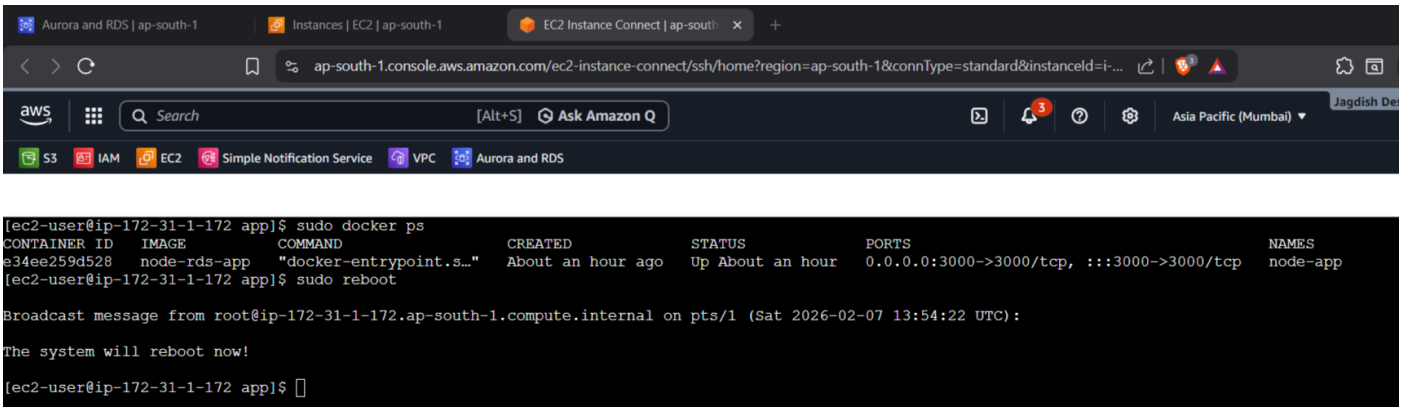
- EC2 reboots
- Docker service restarts

Container will NOT restart only if you manually stop it

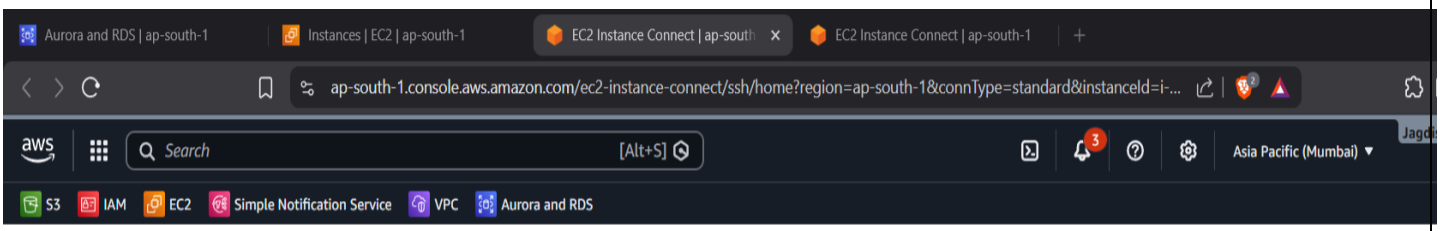
Step 10 :Test it by Restarting EC2

Sudo docker ps

sudo reboot



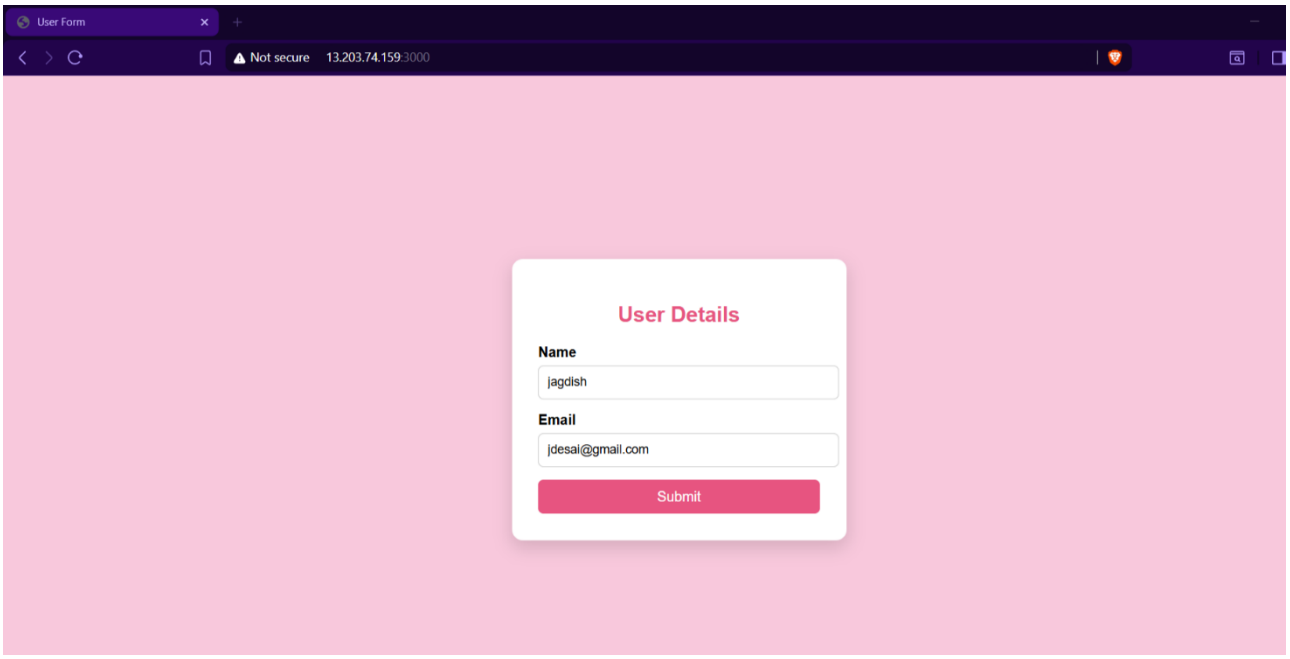
- **After Rebooting Reconnect to our EC2 instance**:



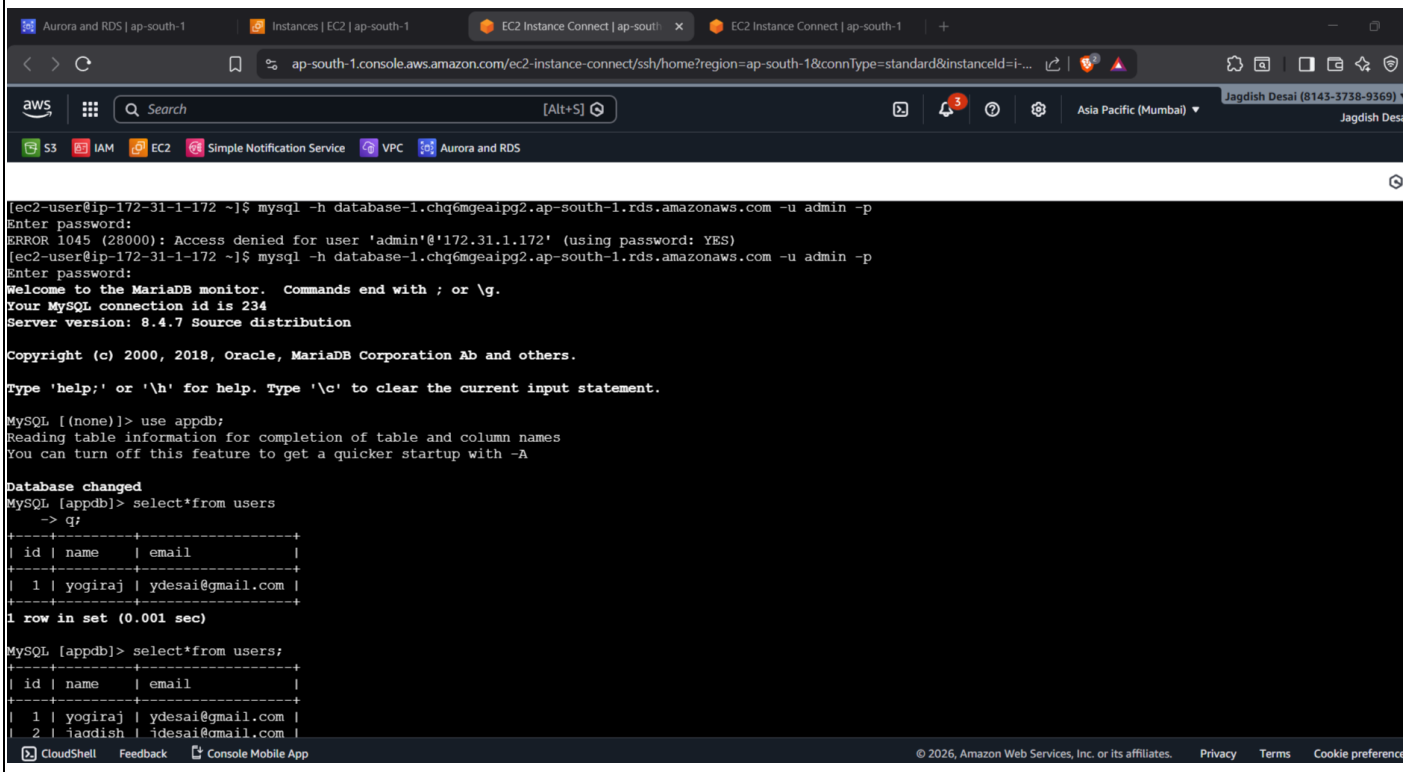

```
[ec2-user@ip-172-31-1-172 ~]$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED             STATUS          PORTS                                       NAMES
e34ee259d528   node-rds-app   "docker-entrypoint.s…"   About an hour ago   Up 10 minutes   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   node-app
[ec2-user@ip-172-31-1-172 ~]$
```

see node-app running, just like before, because you set the restart policy to unless-stopped.

- **ReTest our app after rebooting**:



Then You would see database are also properly working after rebooting.



- **What should I done in task 2**

Docker installed
Dockerfile created
App runs inside container
Port exposed
Auto-start enabled

Summary :

1. Created a Node.js application using nodejs.
2. Created a MySQL database using AWS RDS.
3. Installed Docker on AWS EC2.
4. Prepared the application files for Docker.
5. Created a Dockerfile for the Node.js app.
6. Built a Docker image from the Dockerfile.
7. Ran the application inside a Docker container.
8. Exposed the required port to access the app.
9. Connected the app to MySQL RDS database.
10. Enabled auto-start for the Docker container.
11. Rebooted EC2 and verified the app runs automatically.
12. Accessed the application using EC2 public IP.