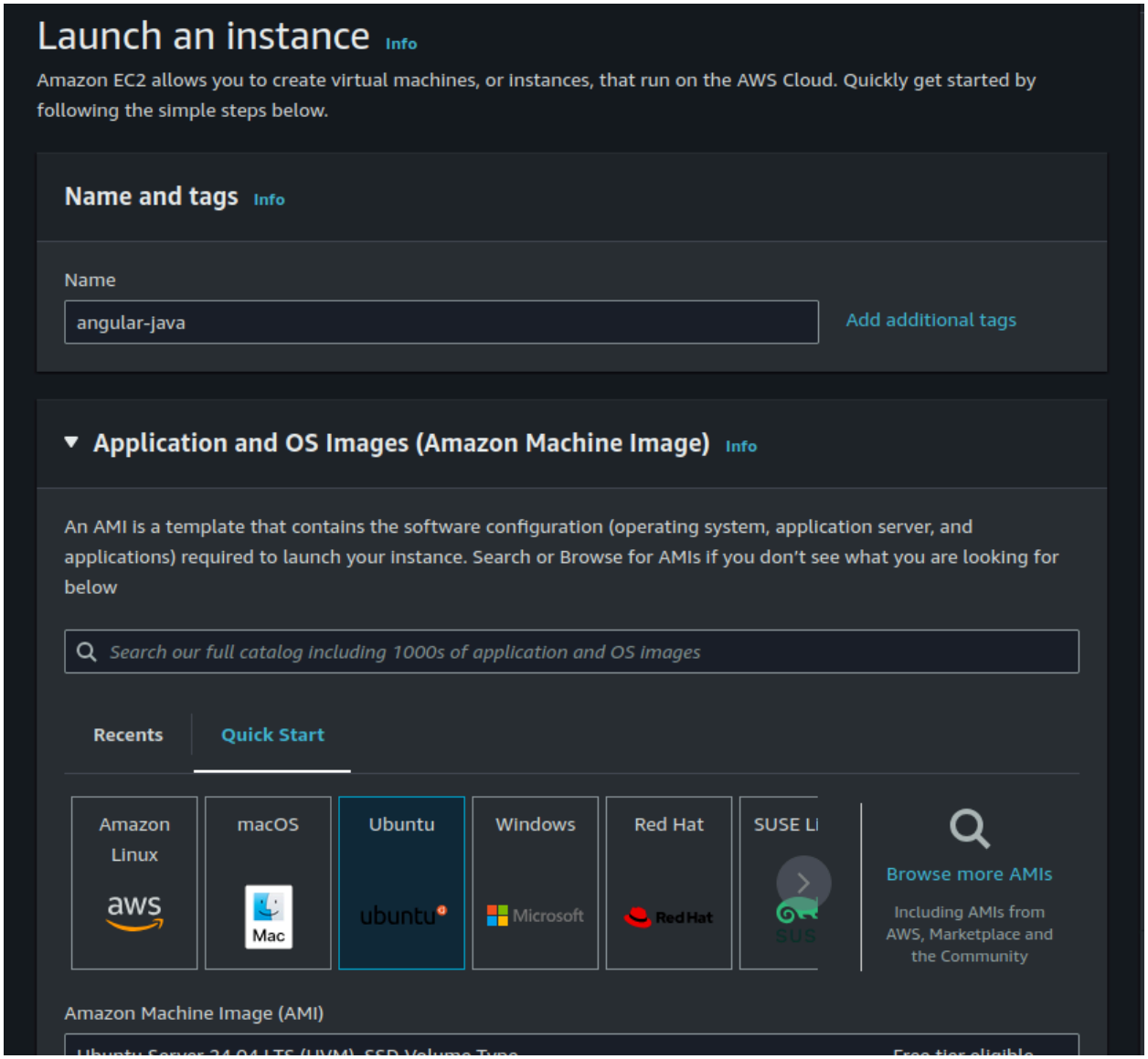
Task : 3-Tier Architecture: Dockerize Angular-Frontend and Java-Backend Application with RDS Integration

Step 1: Create and ec2 instance and launch it by using custom user data to install docker in it.



t2.medium
Family: t2 2 vCPU 4 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0496 USD per Hour
On-Demand Windows base pricing: 0.0676 USD per Hour
On-Demand RHEL base pricing: 0.0784 USD per Hour
On-Demand SUSE base pricing: 0.1496 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

seytan_cloud

Create new key pair

Network settings Info

Edit

Network | Info

vpc-09066077d7529c57f

Subnet | Info

No preference (Default subnet in any availability zone)

Auto-assign public IP | Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) | Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

For V2 requests, you must include a session token in all instance metadata requests. Applications or agents that use V1 for instance metadata access will break.

Metadata response hop limit | Info

2

Allow tags in metadata | Info

Select

User data - *optional* | Info

Upload a file with your user data or enter it in the field.

Choose file

```
#!/bin/bash


for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done

# Add Docker's official GPG key:
sudo apt-get update -y
sudo apt-get install ca-certificates curl -y
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
```

User data has already been base64 encoded

Step 2: Take ssh of the instance.

```
seytan@seytan-Inspiron-3501:~$ ssh -i seytan_cloud.pem ubuntu@3.108.194.171
The authenticity of host '3.108.194.171 (3.108.194.171)' can't be established.
ED25519 key fingerprint is SHA256:Tz4cGr40106agsgeLweFfMYmbF9UThRRB9FYOyqCzfQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.108.194.171' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1012-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Fri Sep 20 05:40:25 UTC 2024

  System load:  0.7               Processes:             131
  Usage of /:   24.7% of 6.71GB   Users logged in:       0
  Memory usage: 7%                IPv4 address for enX0: 172.31.34.179
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

Step 3: Now git clone the repository .

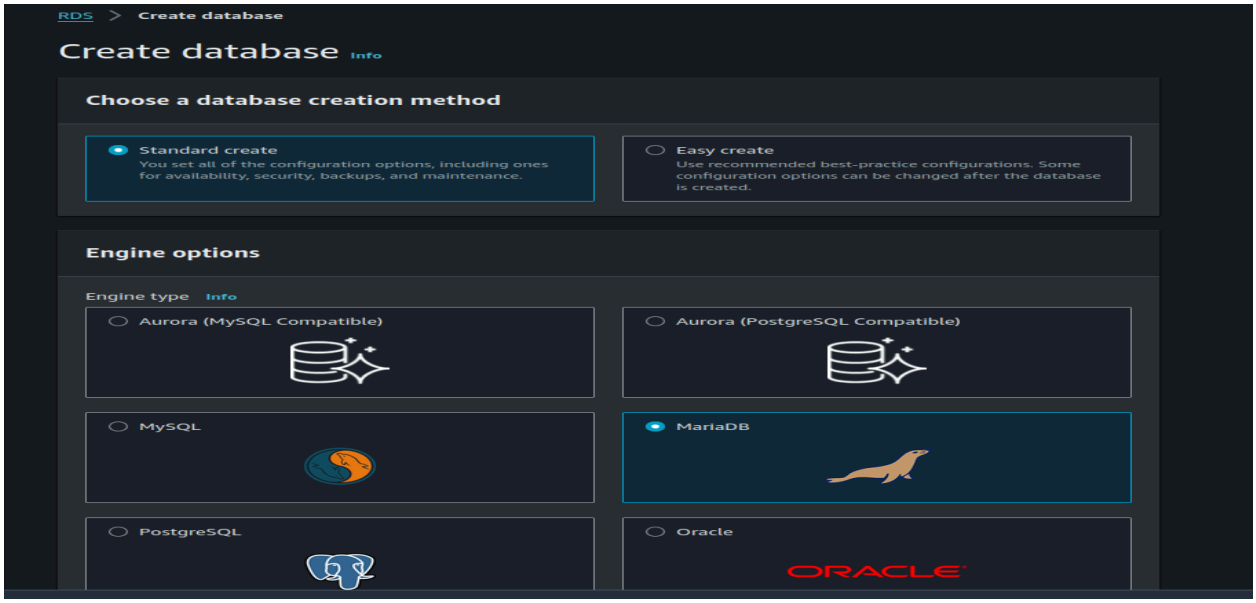
git clone https://github.com/rajatpzade/anguler-java.git

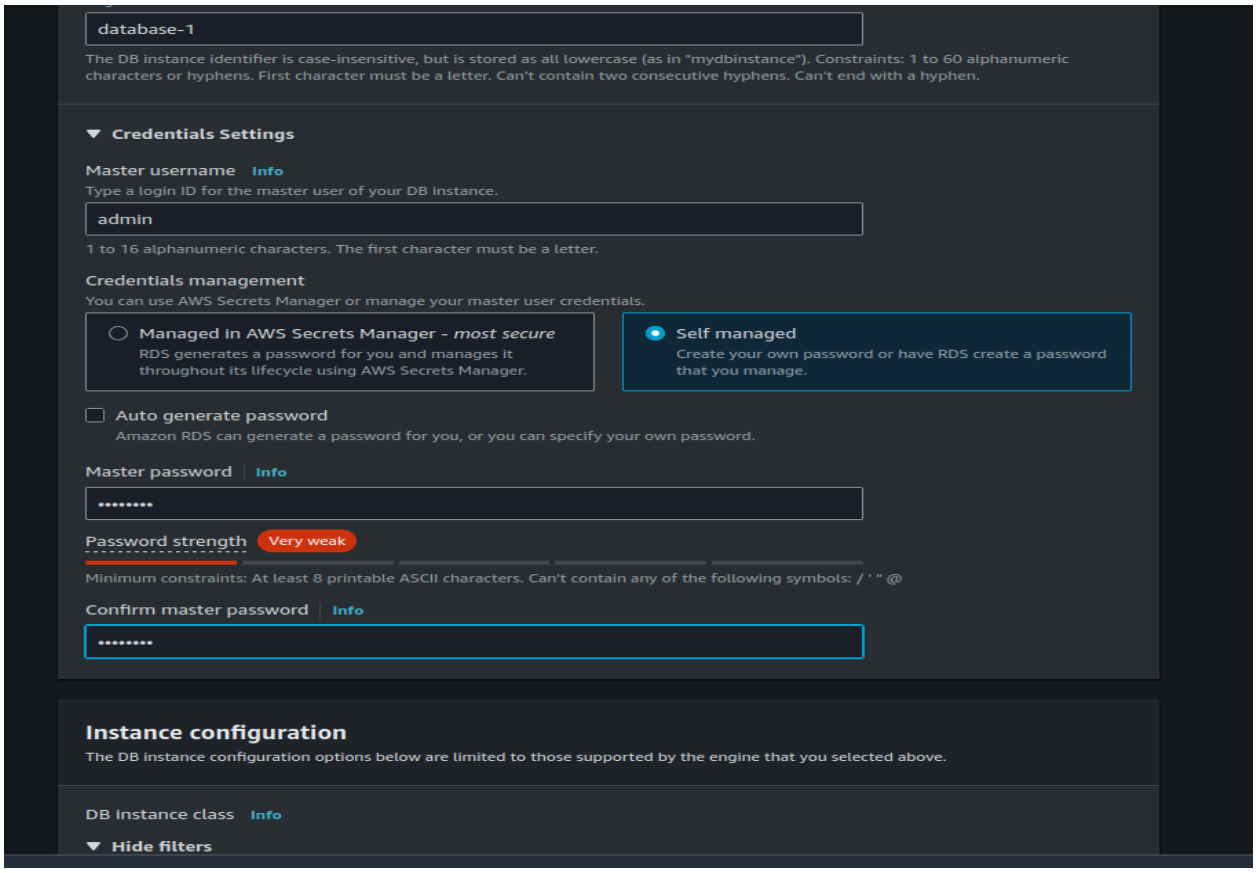
```
Sep 20  11:10
ubuntu@ip-172-31-3
ubuntu@ip-172-31-34-179:~$ git clone https://github.com/rajatpzade/anguler-java.git
Cloning into 'anguler-java'...
remote: Enumerating objects: 80, done.
remote: Counting objects: 100% (80/80), done.
remote: Compressing objects: 100% (62/62), done.
remote: Total 80 (delta 3), reused 80 (delta 3), pack-reused 0 (from 0)
Receiving objects: 100% (80/80), 268.11 KiB | 9.24 MiB/s, done.
Resolving deltas: 100% (3/3), done.
ubuntu@ip-172-31-34-179:~$ _
```

Step 4: Now update the os and install the mariadb -server .

```
Sep 20  11:11
ubuntu@ip-172-31-3
ubuntu@ip-172-31-34-179:~$ sudo apt update
sudo apt install mariadb-server
sudo systemctl start mariadb
sudo systemctl enable mariadb
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... 84%
```

Step 5: Now create a rds database with user admin and password .





Step 6:Connect to database using following and command and perform some actions.

mysql -h rds-endpoint -u username -p

```
ubuntu@ip-172-31-34-179: ~
ubuntu@ip-172-31-34-179:~$ mysql -h database-1.c7kqi8k66m89.ap-south-1.rds.amazonaws.com -u admin -p12345678
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 26
Server version: 10.11.8-MariaDB-log managed by https://aws.amazon.com/rds/

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

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

MariaDB [(none)]> CREATE DATABASE springbackend;
Query OK, 1 row affected (0.005 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON springbackend.* TO 'username'@'localhost' IDENTIFIED BY 'your_password';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]>
MariaDB [(none)]> GRANT ALL PRIVILEGES ON springbackend.* TO 'admin'@'3.108.194.171' IDENTIFIED BY '12345678';
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.009 sec)

MariaDB [(none)]> exit
Bye
ubuntu@ip-172-31-34-179:~$ mysql -h database-1.c7kqi8k66m89.ap-south-1.rds.amazonaws.com -u admin -p12345678 springbackend < anguler-java/springbackend.sql
ubuntu@ip-172-31-34-179:~$ _
```

Also copy the springbackend.sql file to the database springbackend in the rds.

Step 7: Now go to the spring-backend folder and create a Dockerfile .
Also make changes in the application.properties file and add the database endpoint and username and password there.

```
ubuntu@ip-172-31-34-179: ~/anguler-java/spring-backend
ubuntu@ip-172-31-34-179: ~/anguler-java/spring-backend        seytan@seytan-Inspiron-3501: ~
ubuntu@ip-172-31-34-179: $ cd anguler-java/spring-backend/
ubuntu@ip-172-31-34-179:~/anguler-java/spring-backend$ nano Dockerfile
ubuntu@ip-172-31-34-179:~/anguler-java/spring-backend$ cat Dockerfile
FROM ubuntu:latest

RUN apt update -y && \
    apt install openjdk-8-jdk -y && \
    apt install maven -y
RUN rm -rf /var/lib/apt/lists/*
WORKDIR /app


COPY . /app

# Create the directory for the properties file


RUN mvn clean package -Dmaven.test.skip=true

EXPOSE 8080

CMD ["java", "-jar", "target/spring-backend-v1.jar"]
ubuntu@ip-172-31-34-179:~/anguler-java/spring-backend$ nano src/main/resources/application.properties
ubuntu@ip-172-31-34-179:~/anguler-java/spring-backend$ cat src/main/resources/application.properties
spring.datasource.url=jdbc:mysql://database-1.c7kqi8k66m89.ap-south-1.rds.amazonaws.com:3306/springbackend?useSSL=false
spring.datasource.username=admin
spring.datasource.password=12345678

spring.jpa.generate-ddl=true
ubuntu@ip-172-31-34-179:~/anguler-java/spring-backend$ _
```

Step 8: Now run the command docker build -t angular:backend to build the backend docker image.

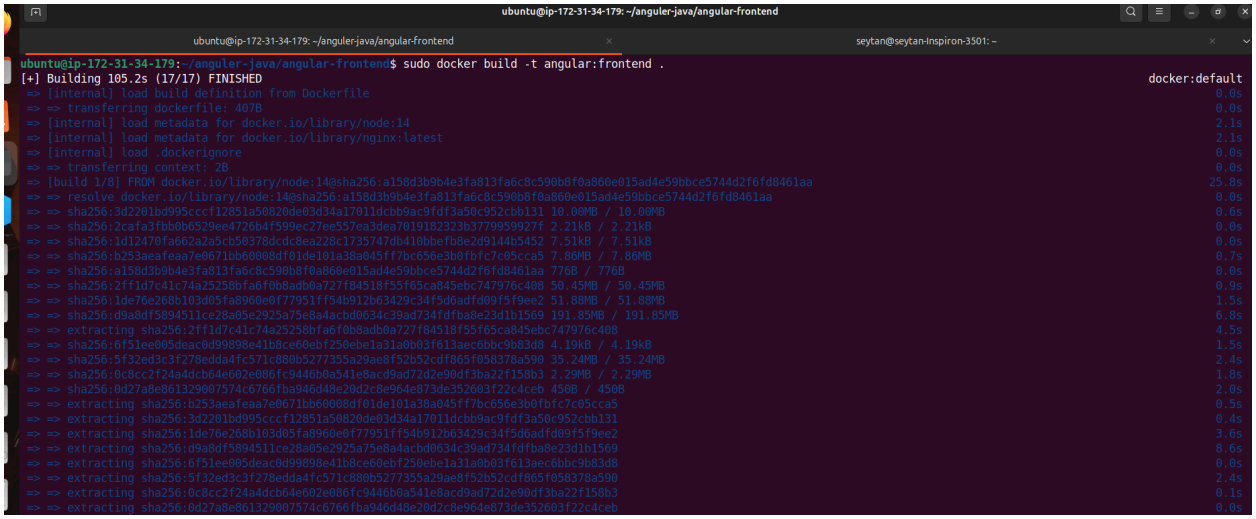
```
ubuntu@ip-172-31-34-179: ~/anguler-java/spring-backend
ubuntu@ip-172-31-34-179: ~/anguler-java/spring-backend        seytan@seytan-Inspiron-3501: ~
ubuntu@ip-172-31-34-179:~/anguler-java/spring-backend$ sudo docker build -t angualr:backend .
[+] Building 102.5s (13/13) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                                0.0s
 => => transferring dockerfile: 743B                                                                                                0.0s
 => [internal] load metadata for docker.io/library/ubuntu:latest                                                                    2.1s
 => [internal] load .dockerignore                                                                                                   0.0s
 => => transferring context: 2B                                                                                                     0.0s
 => [1/8] FROM docker.io/library/ubuntu:latest@sha256:dfc10878be8d8fc9c61cbff33166cb1d1fe44391539243703c72766894fa834a              2.3s
 => => resolve docker.io/library/ubuntu:latest@sha256:dfc10878be8d8fc9c61cbff33166cb1d1fe44391539243703c72766894fa834a              0.0s
 => => sha256:b1e9cef3f2977f8bdd19eb9ae04f83b315f80fe4f5c5651fedf41482c12432f7 2.30kB / 2.30kB                                      0.0s
 => => sha256:dafa2b0c44d2cfb0be6721f079092ddf15dc8bc537fb07fe7c3264c15cb2e8e6 29.75MB / 29.75MB                                    0.6s
 => => sha256:dfc10878be8d8fc9c61cbff33166cb1d1fe44391539243703c72766894fa834a 1.34kB / 1.34kB                                      0.0s
 => => sha256:77d57fd89366f7d16615794a5b53e124d742404e20f035c22032233f1826bd6a 424B / 424B                                          0.0s
 => => extracting sha256:dafa2b0c44d2cfb0be6721f079092ddf15dc8bc537fb07fe7c3264c15cb2e8e6                                           1.5s
 => [internal] load build context                                                                                                   0.0s
 => => transferring context: 87.19kB                                                                                                0.0s
 => [2/8] RUN apt update -y &&     apt install openjdk-8-jdk -y &&     apt install maven -y                                     76.4s
 => [3/8] RUN rm -rf /var/lib/apt/lists/*                                                                                           0.2s
 => [4/8] WORKDIR /app                                                                                                              0.1s
 => [5/8] COPY . /app                                                                                                               0.0s
 => [6/8] RUN rm -f ./src/main/resources/application.properties                                                                     0.2s
 => [7/8] RUN echo "spring.datasource.url=jdbc:mysql://database-1.c7kqi8k66m89.ap-south-1.rds.amazonaws.com:3306/springbackend?useSSL=false \nspring.datasource.username=admin \  0.2s
 => [8/8] RUN mvn clean package -Dmaven.test.skip=true                                                                             15.7s
 => exporting to image                                                                                                              5.0s
 => => exporting layers                                                                                                             4.9s
 => => writing image sha256:e802e66e4c7292cf2973582bf801474074d80b8b3a90422508fd1e24b1c25eb2                                        0.0s
 => => naming to docker.io/library/angualr:backend                                                                                  0.0s
ubuntu@ip-172-31-34-179:~/anguler-java/spring-backend$ sudo docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
angualr      backend   e802e66e4c72   42 seconds ago   858MB
ubuntu@ip-172-31-34-179:~/anguler-java/spring-backend$ _
```

Step 9: Now create another Dockerfile for the frontend in the folder angular-frontend.

In the line number 11 change the ip with your public ip of the instance.
OR you can change it manually in the worker.service.ts file .

```
ubuntu@ip-172-31-34-179: ~/anguler-java/angular-frontend
ubuntu@ip-172-31-34-179: ~/anguler-java/angular-frontend        seytan@seytan-Inspiron-3501: ~
ubuntu@ip-172-31-34-179:~/anguler-java/angular-frontend$ nano Dockerfile
ubuntu@ip-172-31-34-179:~/anguler-java/angular-frontend$ cat Dockerfile
FROM  node:14 AS build

RUN apt update -y
WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .
RUN sed -i 's/localhost/3.108.194.171/g' ./src/app/services/worker.service.ts

RUN npm run build

FROM nginx:latest

WORKDIR /usr/share/nginx/html

COPY --from=build /app/dist/angular-frontend /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
ubuntu@ip-172-31-34-179:~/anguler-java/angular-frontend$ _
```

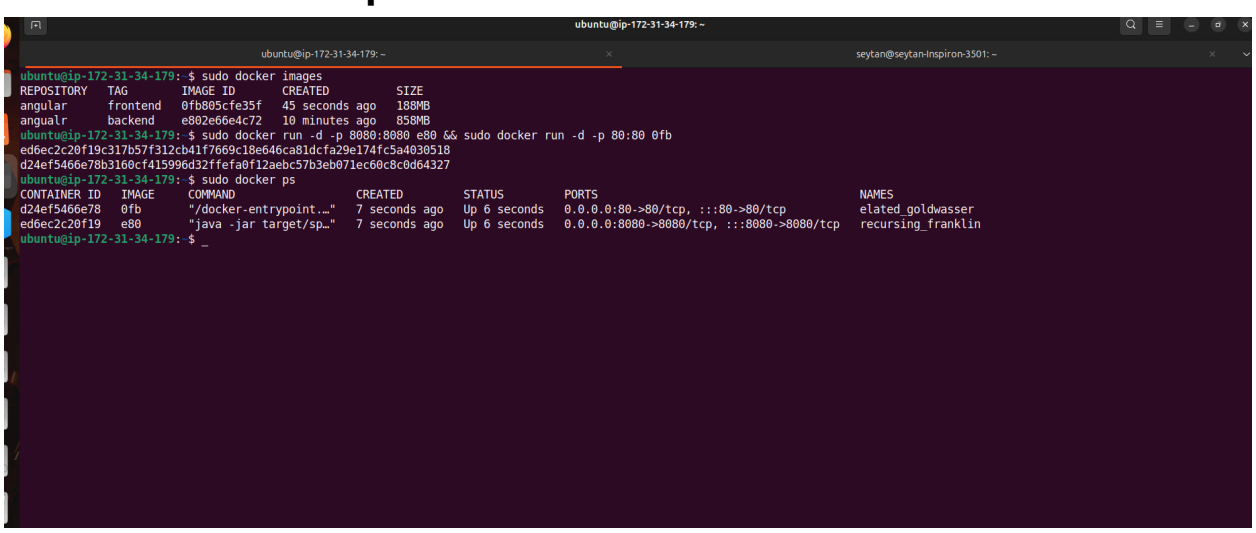
Step 10: Now Build the Docker image for the frontend also.



Step 11: Now check the images by command docker images . After that run the docker images of both backend and frontend .

sudo docker run -d -p 8080:8080 image_id/name && sudo docker run -d -p 80:80 image_id/name

Afer that check the containers are running or not by command docker ps.



Output: Once the both containers are running paste the public ip in the web browser and check the results.