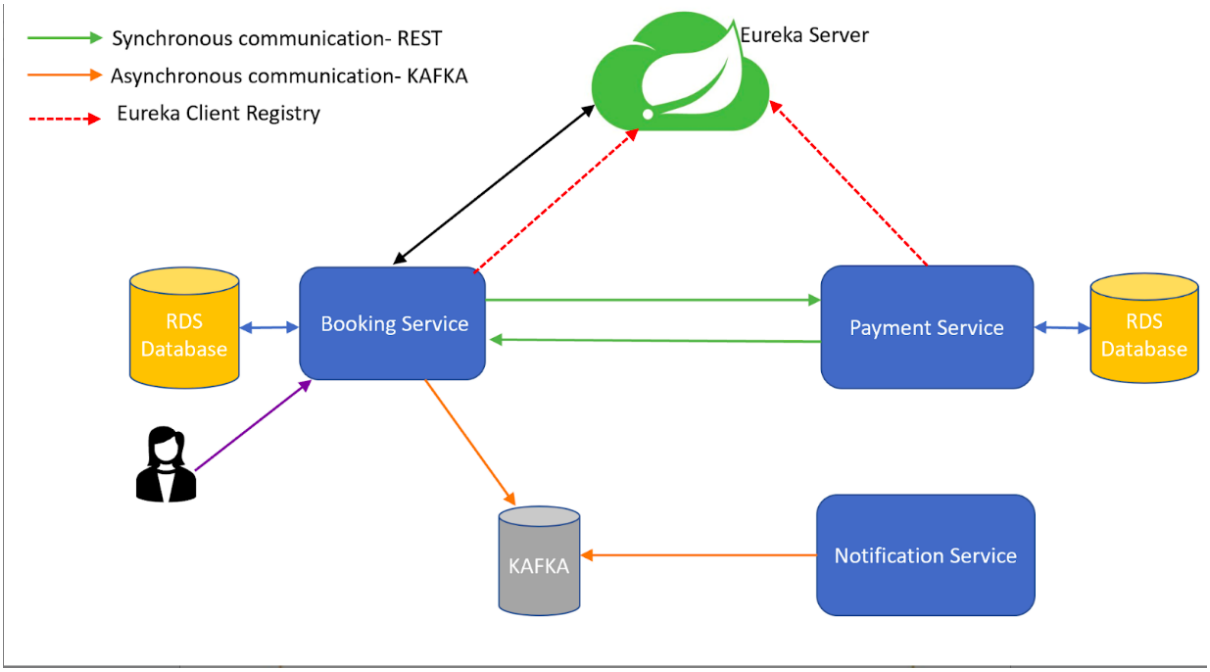
CodingLogic

> using external KAFKA details explained below

LOGIC



→ using external RDS database

→ using external Kafka service

→ passing details of kafa and database as environment in all the services so they can be picked up directly inside the pod

→ docker files are placed inside folder of each service

→ docker compose placed at parent of all the services folder

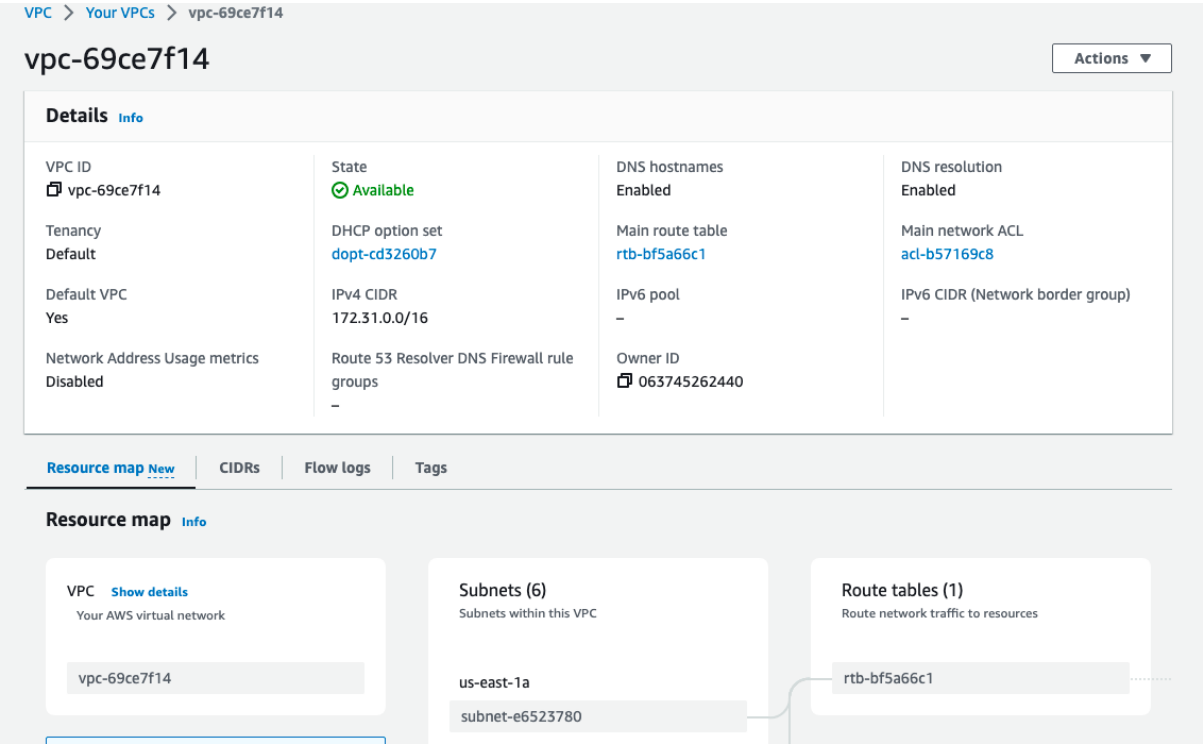
→ path to build image from which docker file specified in docker compose file

→ docker compose will take care of building images and running them by injecting environment variables as mentioned in the docker compose file

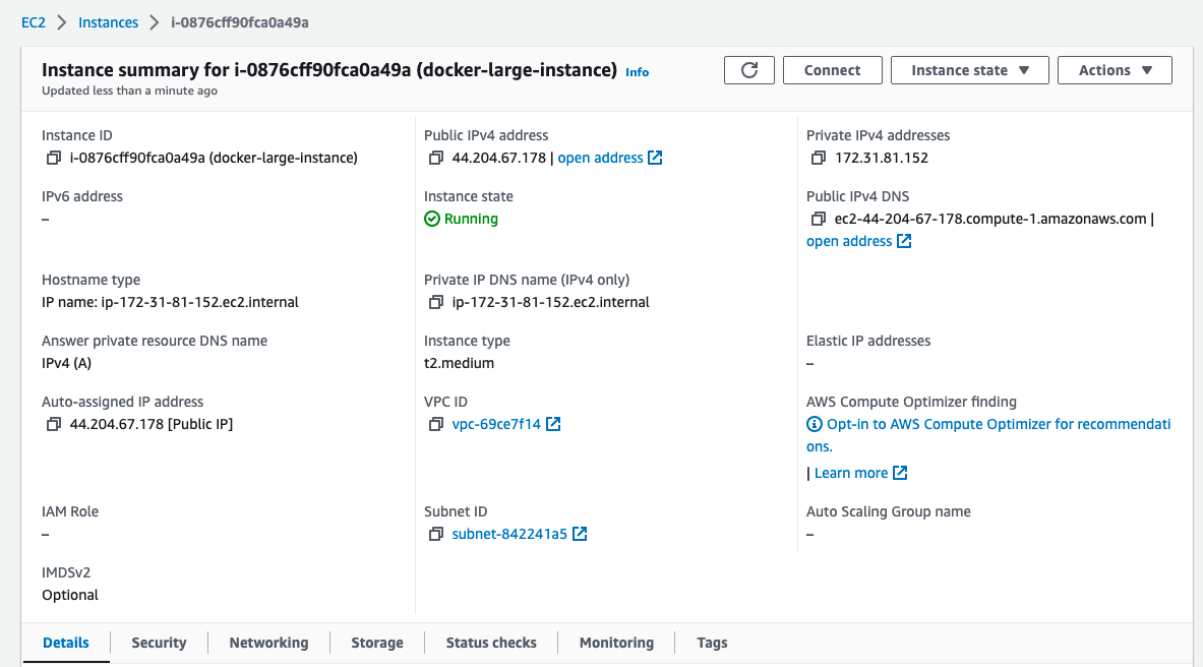
1. Installation and setup

→ **Docker setup**

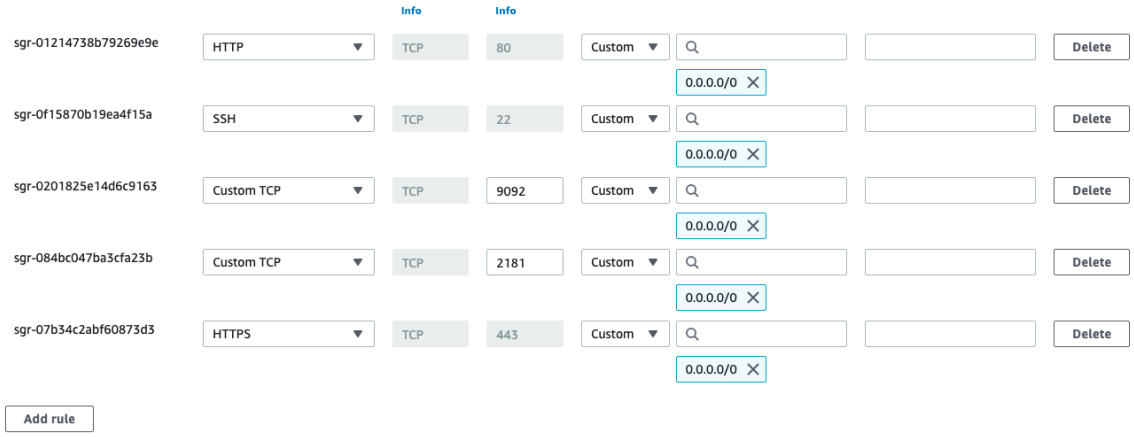
- created VPC with name `vpc-69ce7f14`



- Created `ubuntu` EC2 instance with `instance type medium` with above created VPC as shown in below image with name `socker-large-instance`



- updated security group inbound rules with required ports for connection from external as shown below



- Established `ssh connection to the instance` from public ip of Ec2 instance and public key

```
→  AWS ssh -i ./ec2_key_pair_1.pem ubuntu@ec2-44-204-67-178.compute-1.amazonaws.com
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1028-aws x86_64)

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

  System information as of Wed Jul  5 16:05:49 UTC 2023

  System load:                    0.13232421875
  Usage of /:                     29.8% of 23.08GB
  Memory usage:                   50%
  Swap usage:                     0%
  Processes:                      148
  Users logged in:                1
  IPv4 address for br-670ad7f45470: 172.24.0.1
  IPv4 address for docker0:       172.17.0.1
  IPv4 address for eth0:          172.31.81.152

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

24 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

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


Last login: Wed Jul  5 16:05:50 2023 from 171.76.87.84
ubuntu@ip-172-31-81-152:~$
```

- Installed docker

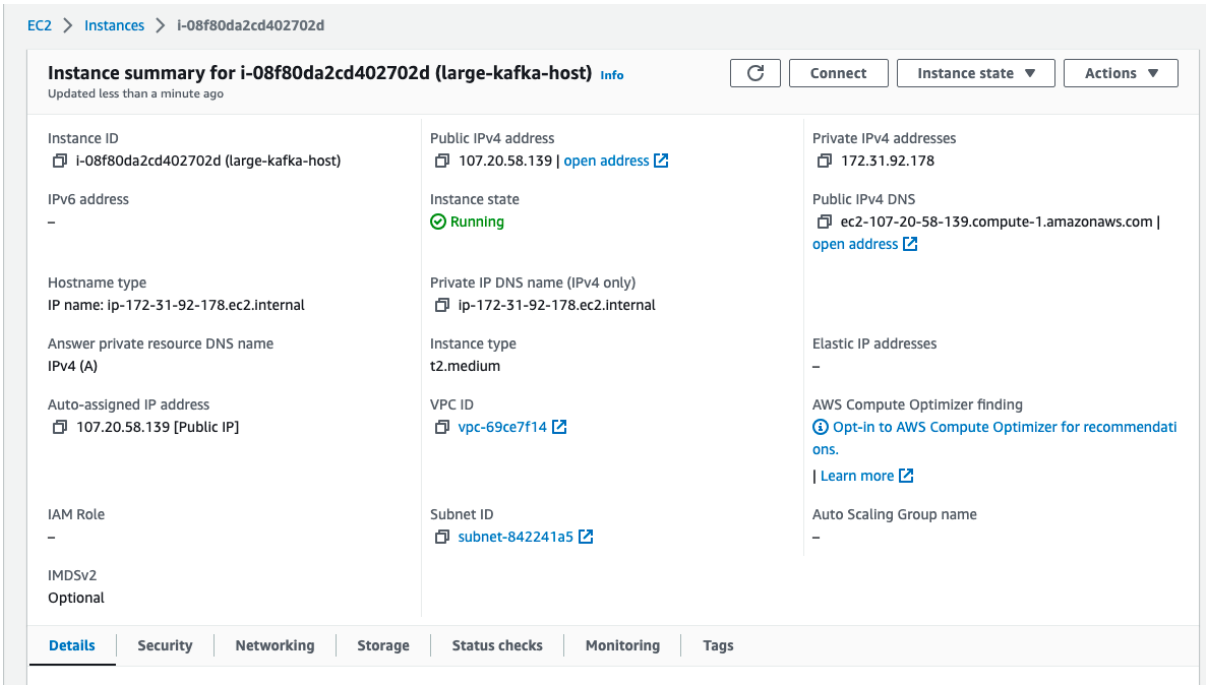
```
ubuntu@ip-172-31-81-152:~/LAB/FINAL_ASSIGNMENT/Sweet-home$ docker -v
Docker version 24.0.2, build cb74dfc
ubuntu@ip-172-31-81-152:~/LAB/FINAL_ASSIGNMENT/Sweet-home$
```

- transfer project files form local machine to Ec2

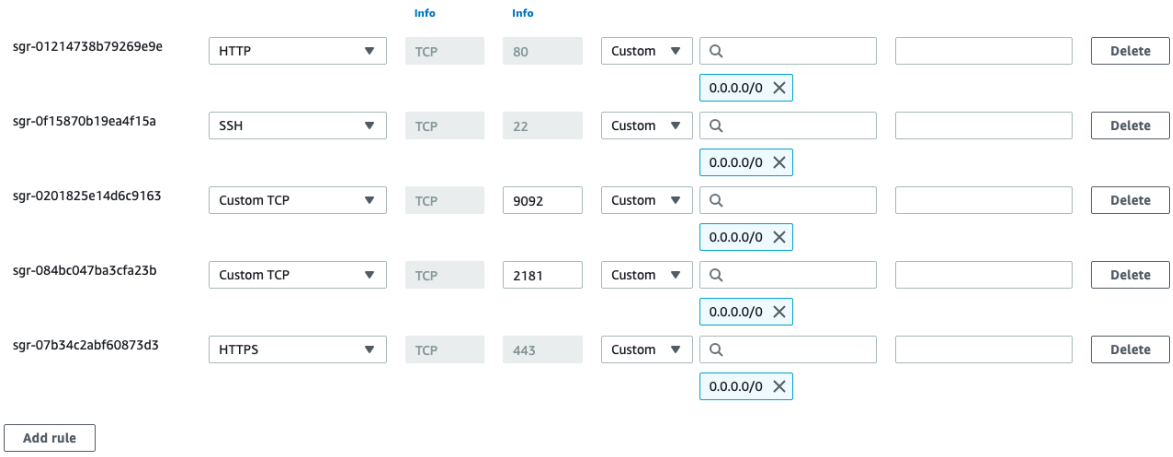
```
drwxr-xr-x 8 ubuntu ubuntu   4096 Jul  5 09:01 upgrad-sweet-home-docker-deployment/
-rw-r--r-- 1 ubuntu ubuntu 175822 Jul  4 05:26 upgrad-sweet-home-docker-deployment.tar.gz
ubuntu@ip-172-31-81-152:~/LAB$
```

→ **Kafka Setup**

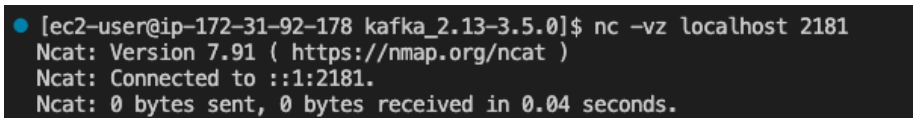
- using same VPC as created in docker setup `vpc-69ce7f14` and created a Ec2 instance with `t2.medium` named `large-kafka-host` for installing kafka



- updated security group inbound rules for external access of kafka ports `2181` and `9092`



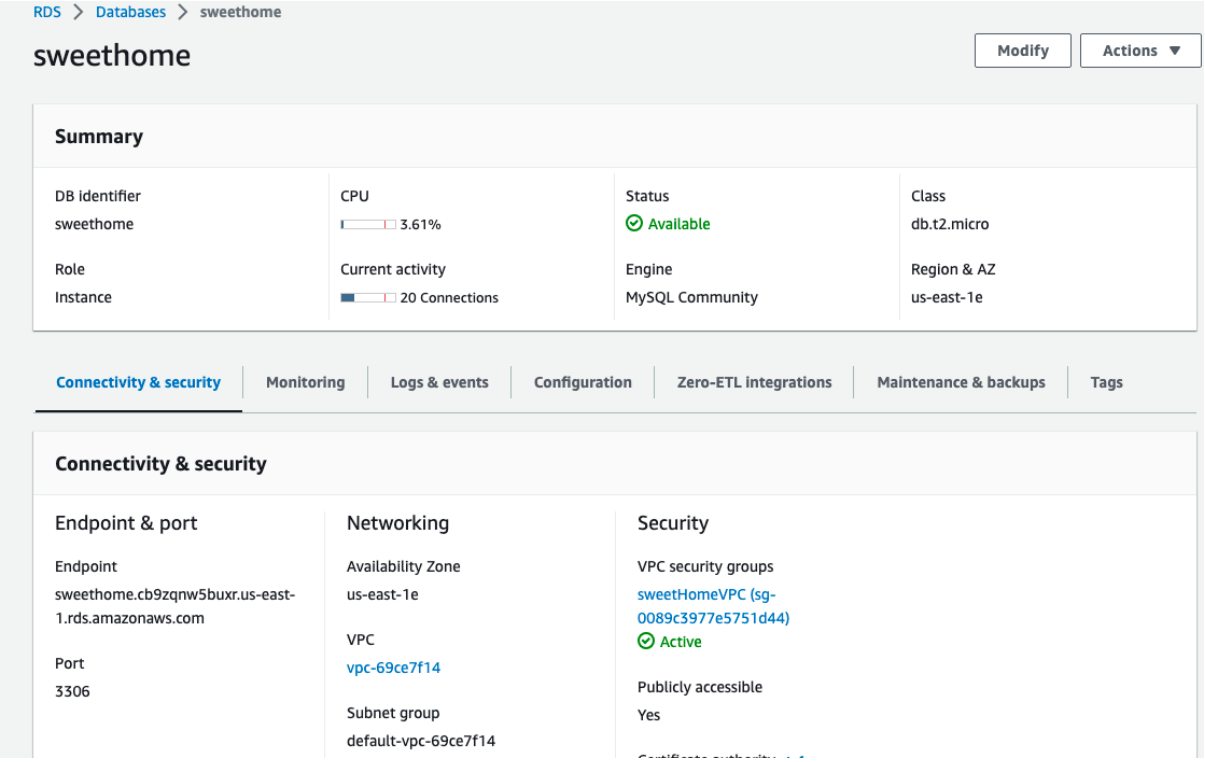
- installed `kafka` by referring to the document provided
- started `zookeeper` and `kafka`



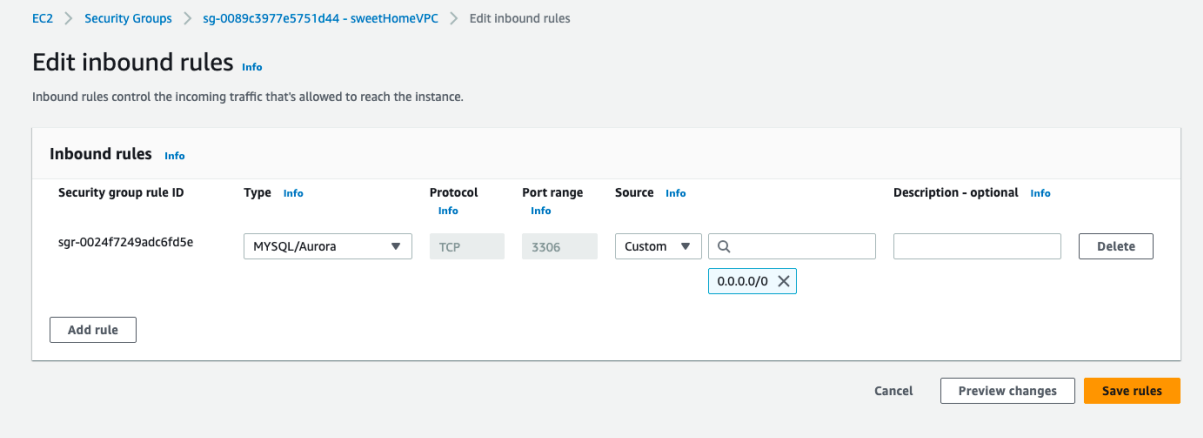
```
[ec2-user@ip-172-31-92-178 kafka_2.13-3.5.0]$ nc -vz localhost 9092
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Connected to ::1:9092.
Ncat: 0 bytes sent, 0 bytes received in 0.04 seconds.
```

→ **RDS setups**

- created `RDS` with Database `MySQL` names `sweethome`



- open port to access database from anywhere



- connect to mysql host from local host and create databases

```
→  ASSIGNMENT mysql --host=sweethome.cb9zqnw5buxr.us-east-1.rds.amazonaws.com --user=admin --passwor
d=upgrad123
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 491
Server version: 5.7.42-log 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> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| SweetHomeBooking   |
| SweetHomePayment   |
| innodb             |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
7 rows in set (0.34 sec)
```

---

2. Docker File creation

payment service

→ updated mysql connection made them to connect from environment variables passed

```
spring.datasource.url = jdbc:mysql://${MYSQL_HOST:localhost}:${MYSQL_HOST_PORT:3306}/SweetHomePayment
spring.datasource.username = ${MYSQL_USER:admin}
spring.datasource.password = ${MYSQL_PASSWORD:upgrad123}
```

→ wrote `Dockerfile` to build images

```
FROM maven:3.8.2-jdk-11 AS builder
WORKDIR /usr/src/app
COPY src ./src
COPY pom.xml .
RUN mvn clean install -DskipTests

FROM openjdk:11.0.11-jdk-slim
WORKDIR /usr/app
COPY --from=builder /usr/src/app/target/paymentService.jar /usr/app/paymentService.jar
ENV PATH="${PATH}:${JAVA_HOME}/bin"
EXPOSE 8083
ENTRYPOINT ["java", "-jar", "/usr/app/paymentService.jar"]
```

---

booking service

→ updated mysql connection made them to connect from environment variables passed

```
spring.datasource.url = jdbc:mysql://${MYSQL_HOST:localhost}:${MYSQL_HOST_PORT:3306}/SweetHomeBooking
spring.datasource.username = ${MYSQL_USER:admin}
spring.datasource.password = ${MYSQL_PASSWORD:upgrad123}
```

→ updated `kafka` connection details to be acceptable from environment variables

```
spring.datasource.url = jdbc:mysql://${MYSQL_HOST:localhost}:${MYSQL_HOST_PORT:3306}/SweetHomeBooking
spring.datasource.username = ${MYSQL_USER:admin}
spring.datasource.password = ${MYSQL_PASSWORD:upgrad123}
```

→ wrote `Dockerfile` to build images

```
FROM maven:3.8.2-jdk-11 AS builder
WORKDIR /usr/src/app
COPY src ./src
COPY pom.xml .
RUN mvn clean install -DskipTests

FROM openjdk:11.0.11-jdk-slim
WORKDIR /usr/app
COPY --from=builder /usr/src/app/target/bookingService.jar /usr/app/bookingService.jar
ENV PATH="${PATH}:${JAVA_HOME}/bin"
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/usr/app/bookingService.jar"]
```

---

notification service

→ updated `kafka` connection details to be acceptable from environment variables

```
String kafkaConnectionstring = System.getenv("KAFKA_HOST")+":"+System.getenv("KAFKA_HOST_PORT");
properties.put("bootstrap.servers", kafkaConnectionstring);
```

→ wrote docker file to build image

```
FROM maven:3.8.2-jdk-11 AS builder
WORKDIR /usr/src/app
COPY src ./src
COPY pom.xml .
RUN mvn clean compile assembly:single

FROM openjdk:11.0.11-jdk-slim
WORKDIR /usr/app
COPY --from=builder /usr/src/app/target/notificationService-jar-with-dependencies.jar /usr/app/notificationService.jar
ENV PATH="${PATH}:${JAVA_HOME}/bin"
#EXPOSE 8761
ENTRYPOINT ["java", "-jar", "/usr/app/notificationService.jar"]
```

eureka service

→ wrote docker file for eureka

```
FROM maven:3.8.2-jdk-11 AS builder
WORKDIR /usr/src/app
COPY src ./src
COPY pom.xml .
RUN mvn clean install -DskipTests

FROM openjdk:11.0.11-jdk-slim
WORKDIR /usr/app
COPY --from=builder /usr/src/app/target/eurekaServer.jar /usr/app/eurekaServer.jar
ENV PATH="${PATH}:${JAVA_HOME}/bin"
EXPOSE 8761
ENTRYPOINT ["java", "-jar", "/usr/app/eurekaServer.jar"]
```

3. Docker-compose file

- docker compose file will `build images` and `run them in specified network`
- docker compose file contains environment variables which are given at time of run
- these environment variables are the `ip for external kafka` and `IP of sql database` and respective `ports`
- while running these values to be filled and just fire `sudo docker compose up`

```
version: '3.3'

services:

  eurekaservice:
    build: eurekaserver
    container_name: eureka-server
    image: sweet-home/eurekaserver:1.0.1
    ports:
    - "8761:8761"
    networks:
      - "microservicesnet"
    hostname: eureka-service

  notificationservice:
    build: notificationservice
    container_name: notification-service
    image: sweet-home/notificationservice:1.0.1
    environment:
      KAFKA_HOST: "107.20.58.139"
      KAFKA_HOST_PORT: "9092"
    networks:
      - "microservicesnet"
    hostname: notification-service

  bookingservice:
    build: bookingservice
    container_name: booking-service
    image: sweet-home/bookingservice:1.0.1
    ports:
```

```
    - "8080:8080"
    networks:
      - "microservicesnet"
    environment:
      MYSQL_HOST: "sweethome.cb9zqnw5buxr.us-east-1.rds.amazonaws.com"
      MYSQL_HOST_PORT: 3306
      MYSQL_USER: admin
      MYSQL_PASSWORD: upgrad123
      KAFKA_HOST: "107.20.58.139"
      KAFKA_HOST_PORT: "9092"
    depends_on:
      - eurekaservice
    hostname: booking-service

  paymentservice:
    build: paymentservice
    container_name: payment-service
    image: sweet-home/paymentservice:1.0.1
    ports:
    - "8083:8083"
    networks:
      - "microservicesnet"
    environment:
      MYSQL_HOST: "sweethome.cb9zqnw5buxr.us-east-1.rds.amazonaws.com"
      MYSQL_HOST_PORT: 3306
      MYSQL_USER: admin
      MYSQL_PASSWORD: upgrad123
      KAFKA_HOST: "107.20.58.139"
      KAFKA_HOST_PORT: "9092"
    depends_on:
      - eurekaservice
    hostname: payment-service

networks:
  microservicesnet:
    driver: bridge
```

---

4. Service deployment

`Instruction to run`

Pre requisites

→ docker installed with compose plugin

→ RDS is up and running with the databases `SweetHomeBooking` and `SweetHomePayment`

→ Kafka is up and running as a external kafka

→ zip folder of the project

STEPS

1. unzip the project folder `Sweet-home.zip` then you will end up with `Sweet-home` folder
2. `cd Sweet-home`
3. open `docker-compose.yml` with any editor example: `vi docker-compose.yml`
4. update `KAFKA_HOST, KAFKA_HOST_PORT, MYSQL_HOST, MYSQL_HOST_PORT, MYSQL_USER, MYSQL_PASSWORD` with your proper working values.

5. after filling values by referring to samples given fire following command to run the application
6. `sudo docker compose up`

after successful execution of above command images will be built and services will be starting

running docker images

```
ubuntu@ip-172-31-81-152:~/LAB/FINAL_ASSIGNMENT/Sweet-home$ sudo docker compose ps
NAME                  IMAGE                                  COMMAND                  SERVICE              CREATED              STATUS              PORTS
booking-service       sweet-home/bookingservice:1.0.1        "java -jar /usr/app/…"   bookingservice       6 minutes ago        Up 6 minutes        0.0.0.0:80
80->8080/tcp, :::8080->8080/tcp
eureka-server         sweet-home/eurekaserver:1.0.1          "java -jar /usr/app/…"   eurekaservice        6 minutes ago        Up 6 minutes        0.0.0.0:87
61->8761/tcp, :::8761->8761/tcp
notification-service  sweet-home/notificationservice:1.0.1   "java -jar /usr/app/…"   notificationservice  6 minutes ago        Up 6 minutes
payment-service       sweet-home/paymentservice:1.0.1        "java -jar /usr/app/…"   paymentservice       6 minutes ago        Up 6 minutes        0.0.0.0:80
83->8083/tcp, :::8083->8083/tcp
ubuntu@ip-172-31-81-152:~/LAB/FINAL_ASSIGNMENT/Sweet-home$
```

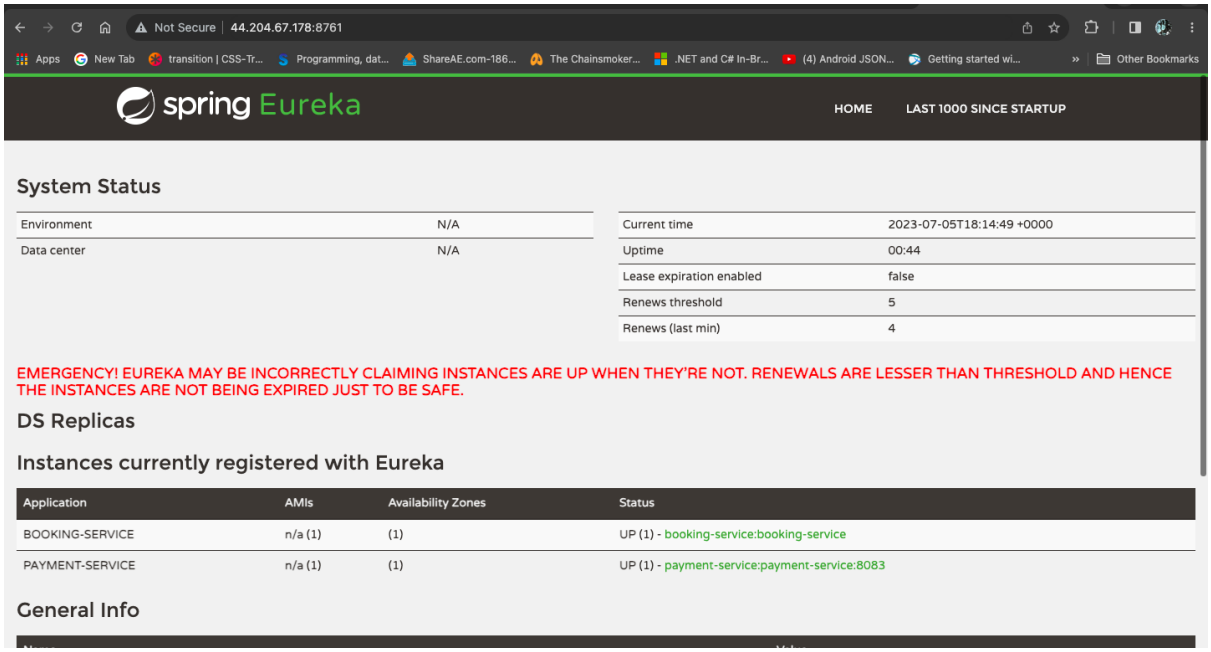
Connect to a power outlet while your backup disk is available.

docker images created after docker compose up

```
ubuntu@ip-172-31-81-152:~/LAB/FINAL_ASSIGNMENT/Sweet-home$ sudo docker images
REPOSITORY                          TAG       IMAGE ID       CREATED         SIZE
sweet-home/notificationservice      1.0.1     6fe850ba7f67   13 hours ago    435MB
sweet-home/bookingservice           1.0.1     fddc074d9332   15 hours ago    494MB
sweet-home/paymentservice           1.0.1     eef13b3e43da   23 hours ago    482MB
sweet-home/eurekaserver             1.0.1     66019c1c89fc   23 hours ago    466MB
ubuntu@ip-172-31-81-152:~/LAB/FINAL_ASSIGNMENT/Sweet-home$
```

eureka server

> after deploying the services they will be appearing in eureka

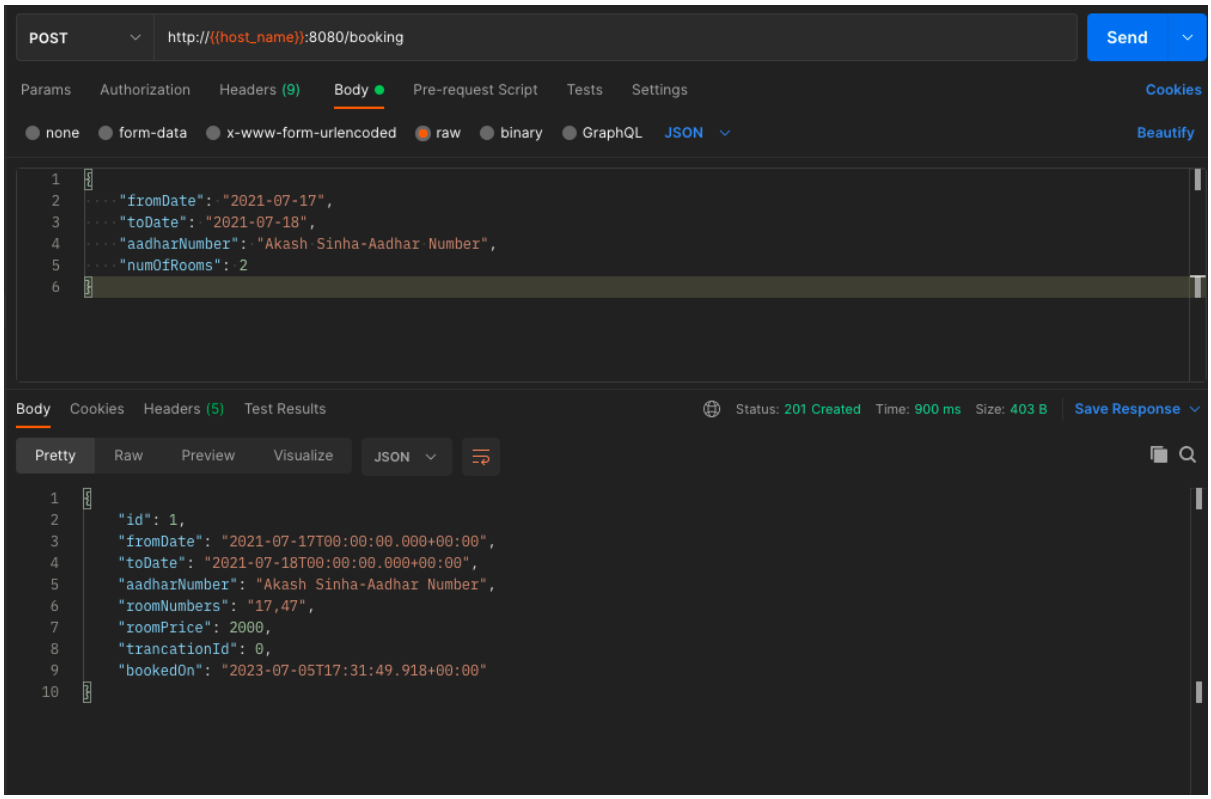


to see eureka service `http://<Ec2-instanceIP>:8761/` replace `Ec2-instanceIP` with ec2 public ip

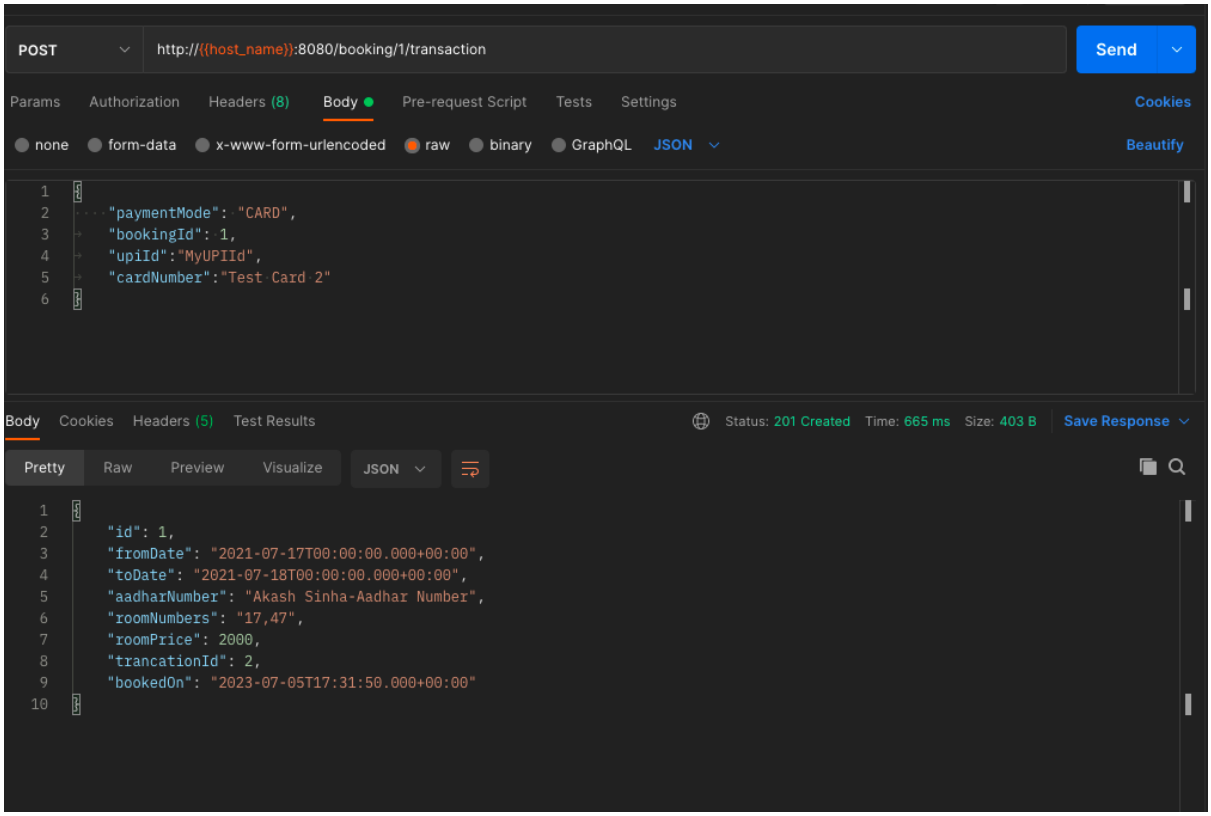
5. APIs testing

booking service

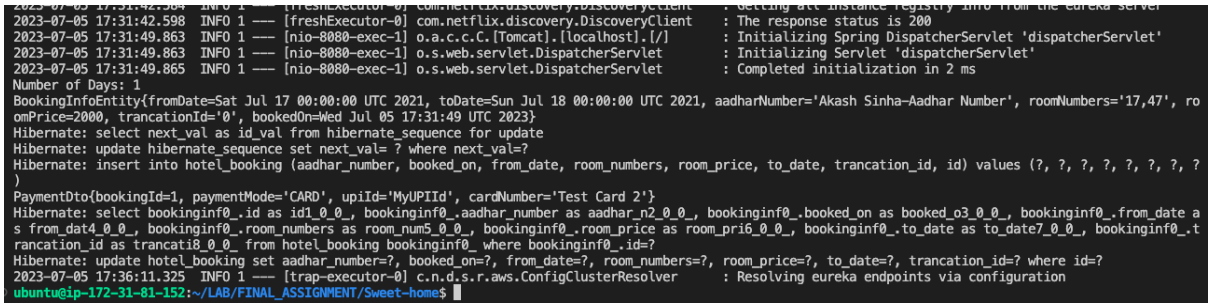
API for booking the rooms



api for making transaction for booked rooms



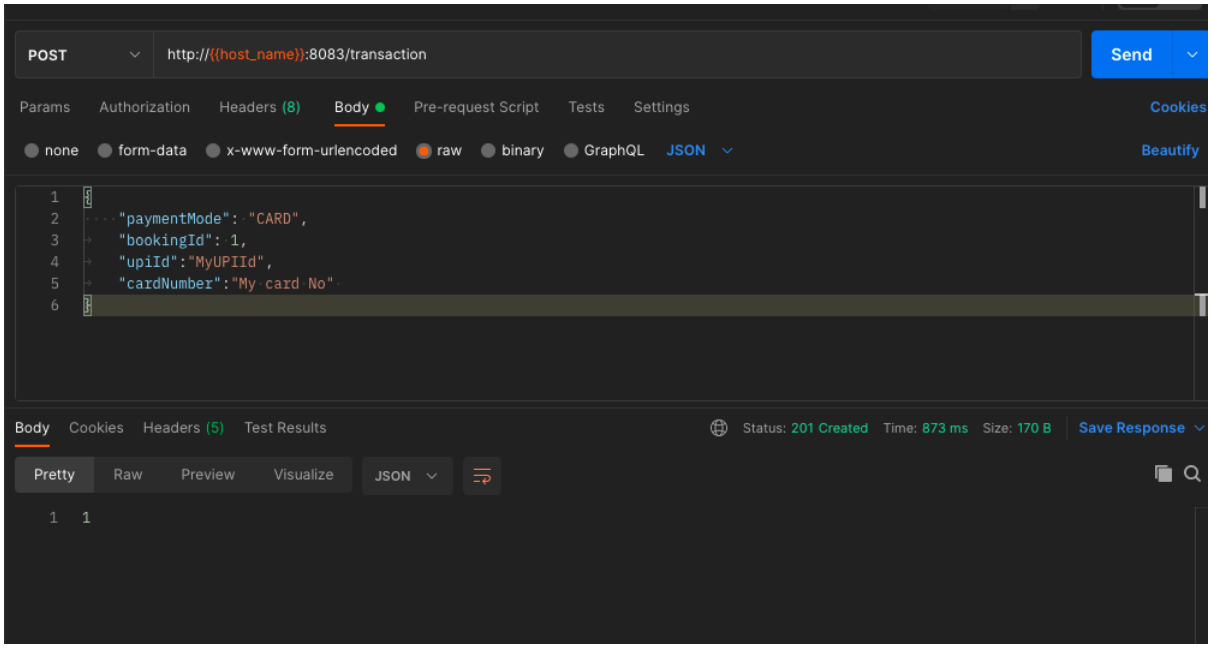
corresponding pod logs for booking service on request



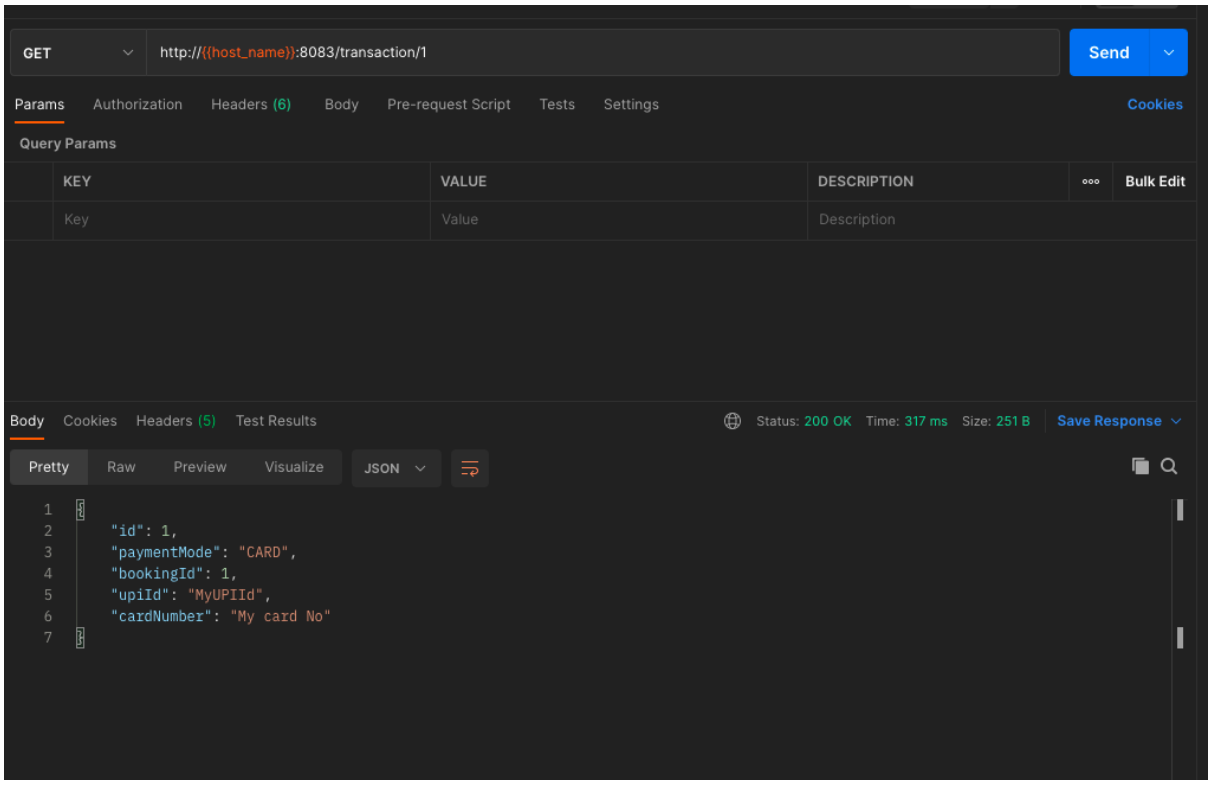
---

paymentservice

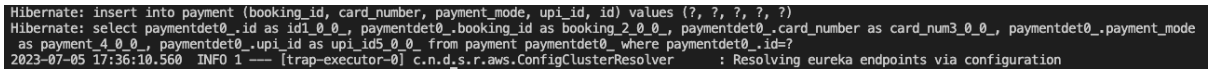
API for payment service payment details



API for payment service get payment details

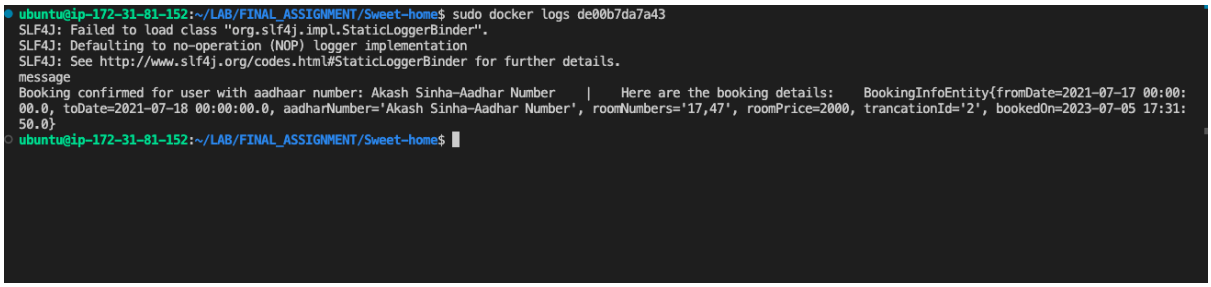


corresponding pod logs on firing request to service



notification service

logs of notification on getting message on registered topic in kafka



`Instruction to run`

Pre requisites

→ docker installed with compose plugin

→ RDS is up and running with the databases `SweetHomeBooking` and `SweetHomePayment`

→ Kafka is up and running as a external kafka

→ zip folder of the project

STEPS

1. unzip the project folder `Sweet-home.zip` then you will end up with `Sweet-home` folder
2. `cd Sweet-home`
3. open `docker-compose.yml` with any editor example: `vi docker-compose.yml`
4. update `KAFKA_HOST, KAFKA_HOST_PORT, MYSQL_HOST, MYSQL_HOST_PORT, MYSQL_USER, MYSQL_PASSWORD` with your proper working values.
5. after filling values by referring to samples, execute below command to run the application from sweet-home folder where `docker-compose.yml` is present
6. `sudo docker compose up`
7. application will be up after that

contributions

Vinod Patil:

→ written docker-compose.yml

→ written bookingservice docker file and changes in booking service

Rupa Cherlopalli:

→ written paymentservice docker file and changes in paymentservice