Solution Approach

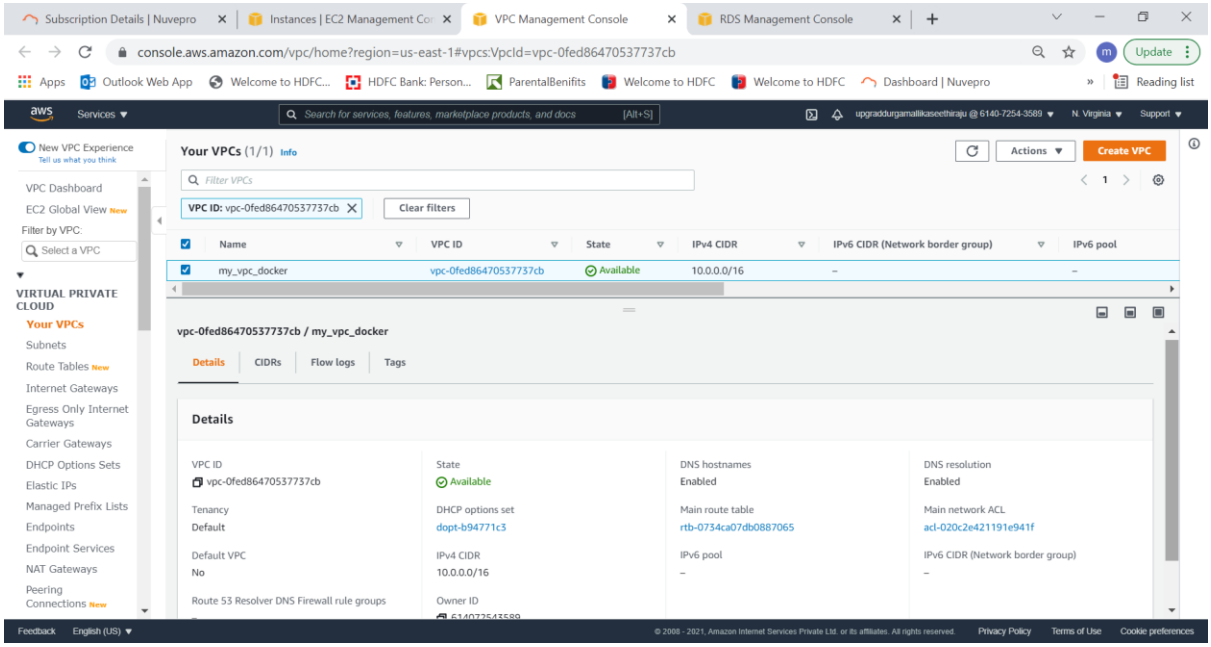
Overview of solution approach:

I have created EC2 instance with my own VPC to install docker. Externalized the database by creating RDS instance in AWS as per the problem statement. Externalized Kafka by creating separate EC2 instance for Kafka with Elastic IP. Downloaded the given codestub and made necessary changes in application.properties, application.yml which is clearly explained with screenshots in step by step solution approach. Also created docker files for each service and docker compose file. I have transferred all the files from local to EC2 instance. Installed Docker on EC2 instance ubuntu machine. Also installed java and maven on EC2 instance. I have followed the case 1 of checkpoint 3 and created jar files for the application. After that created docker images and containers by using “sudo docker-compose up -d”. After starting all the containers, checked whether all the applications are registered in Eureka service registry or not and also verified whether the application is working correctly by posting the request through postman.

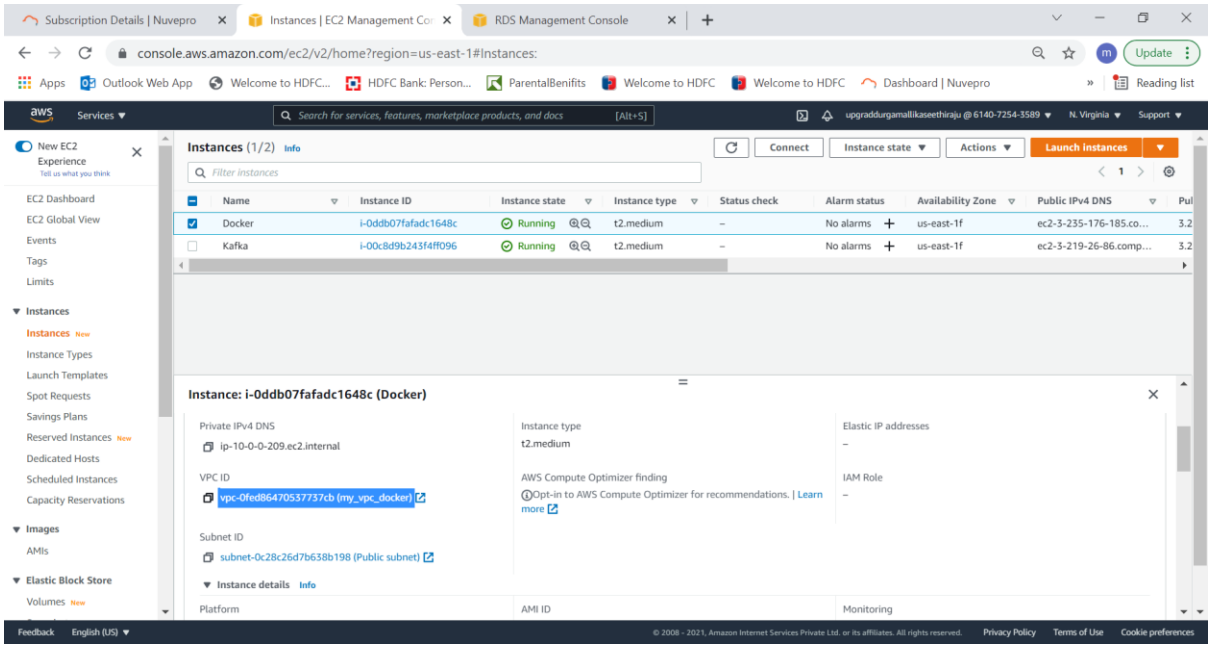
Step by step solution approach with screenshots:

Step 1: Created EC2 instance with my own VPC with ubuntu machine for installing docker.

VPC:

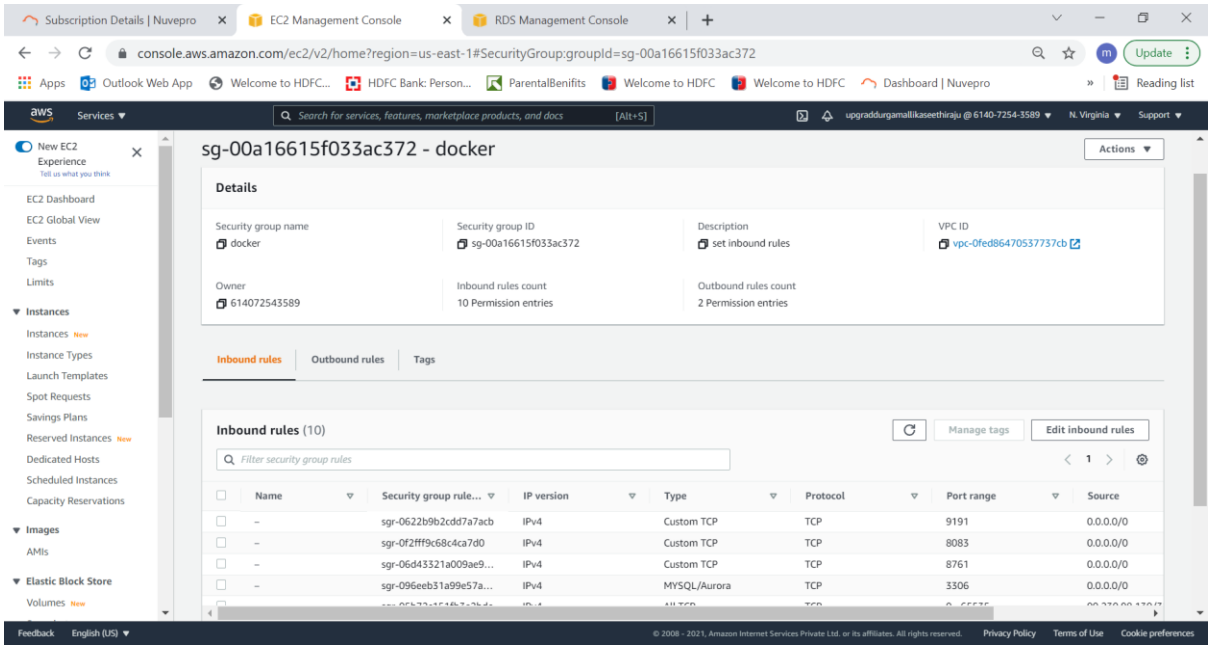


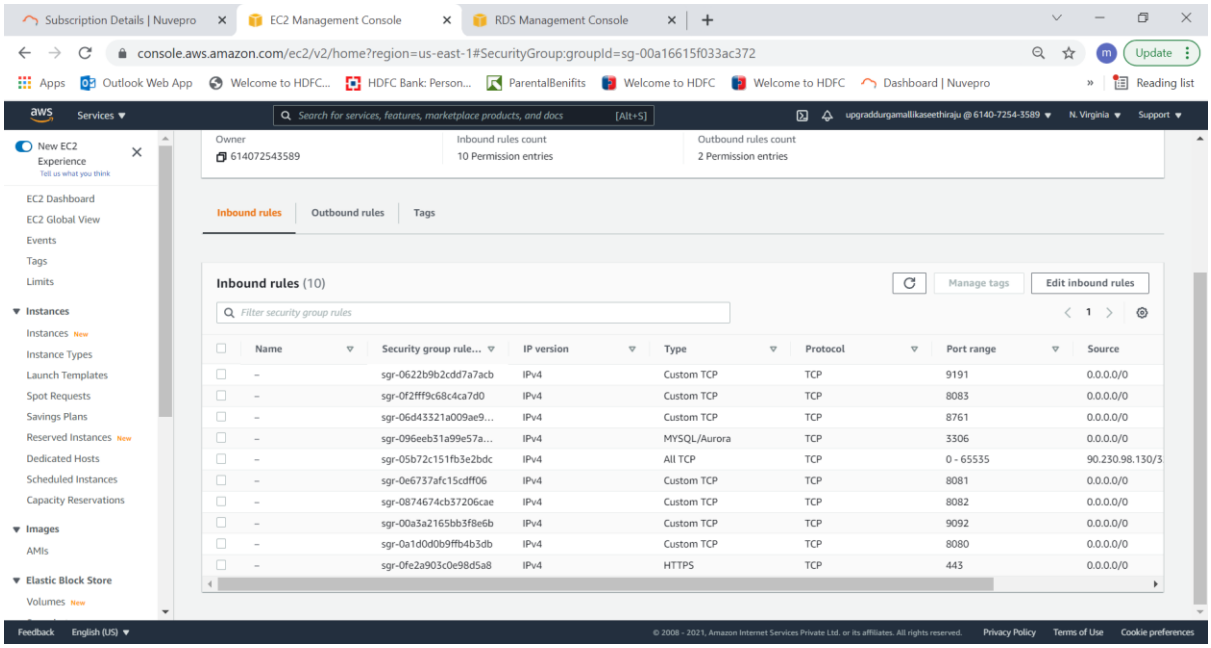
Docker EC2 instance:



Step 2: Added all the inbound rules and outbound rules to open the ports for docker Ec2 instance as required by the application.

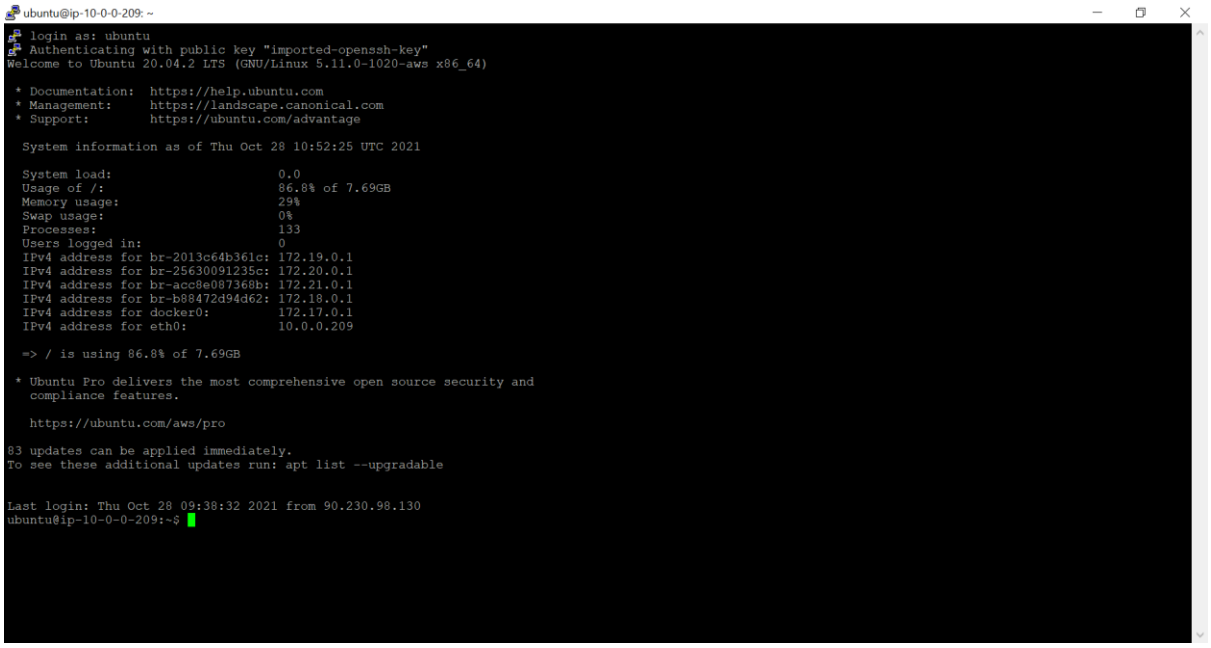
Inbound rules for Docker EC2 Instance:





Login to ubuntu machine is successful

EC2 Login as ubuntu user:

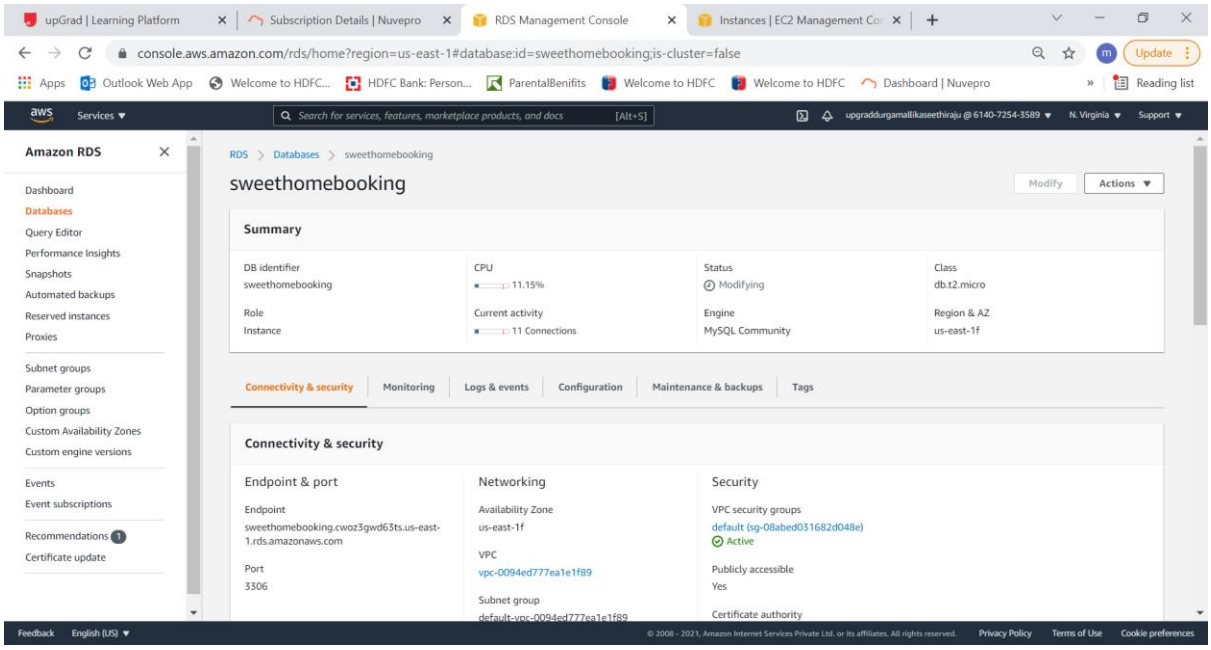


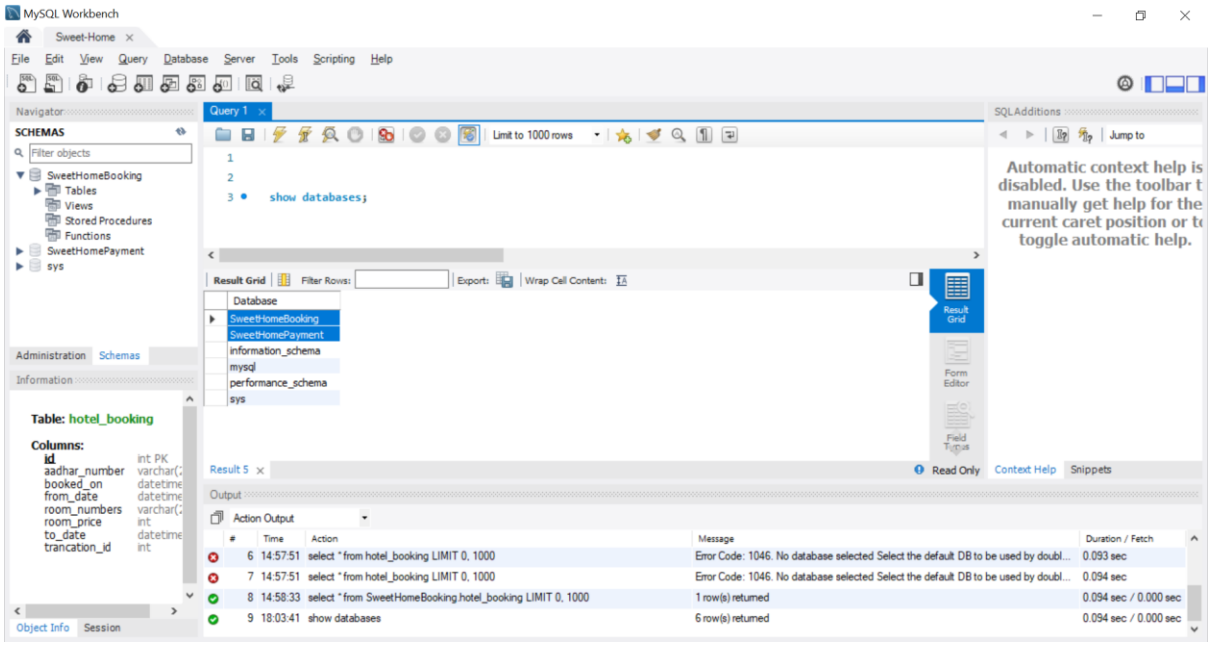
Step 3: RDS instance is created in AWS and created databases using mysql workbench. Added inbound rules as required. Also updated the RDS end point, user name and password in booking service and payment service application.properties file.

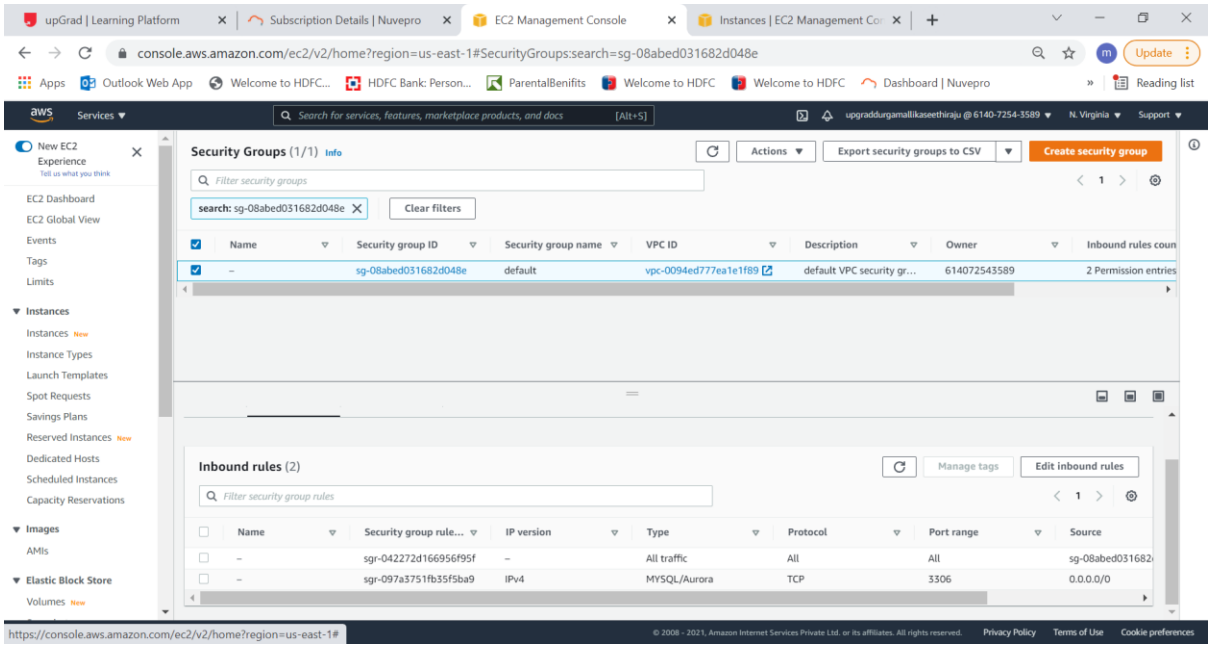
spring.datasource.url = jdbc:mysql://sweethomebooking.cwoz3gwd63ts.us-east-1.rds.amazonaws.com/SweetHomeBooking

spring.datasource.url = jdbc:mysql://sweethomebooking.cwoz3gwd63ts.us-east-1.rds.amazonaws.com/SweetHomePayment

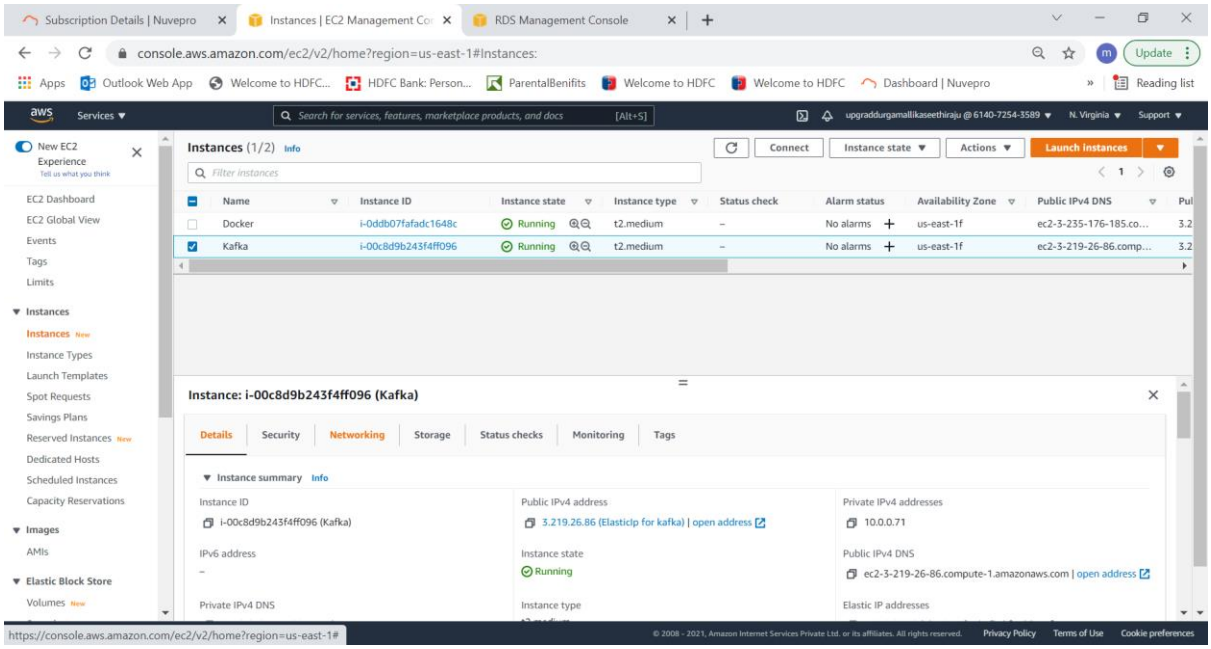
Removed H2 Database dependency from pom.xml file for booking and payment service.

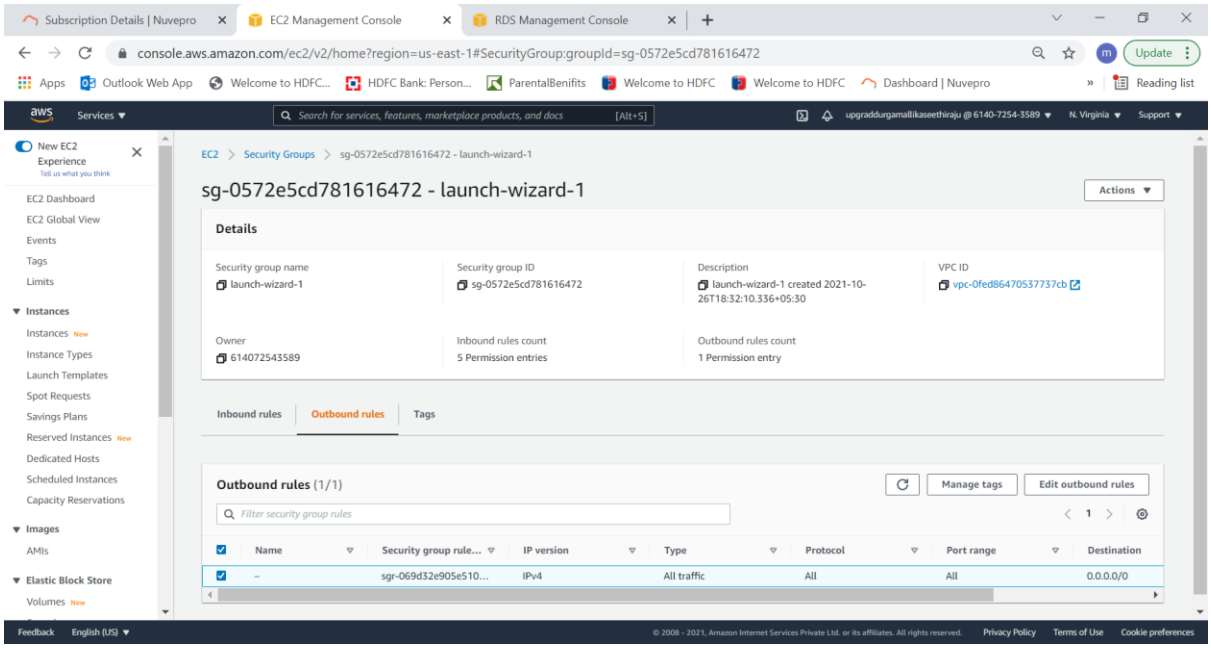


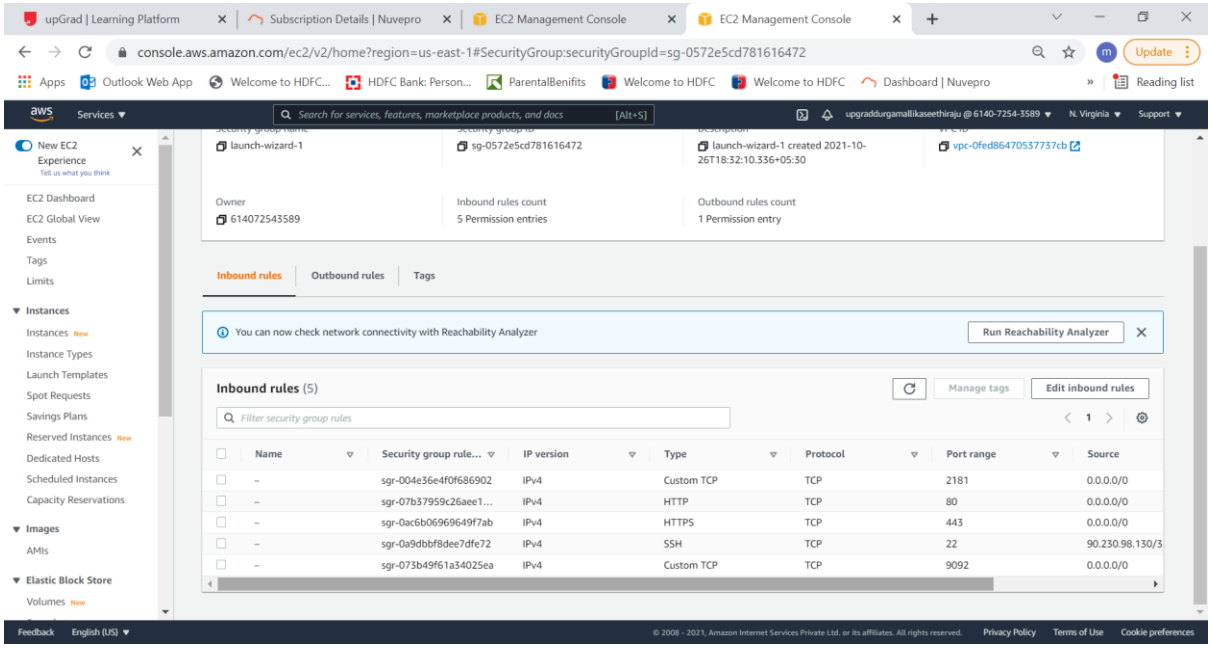




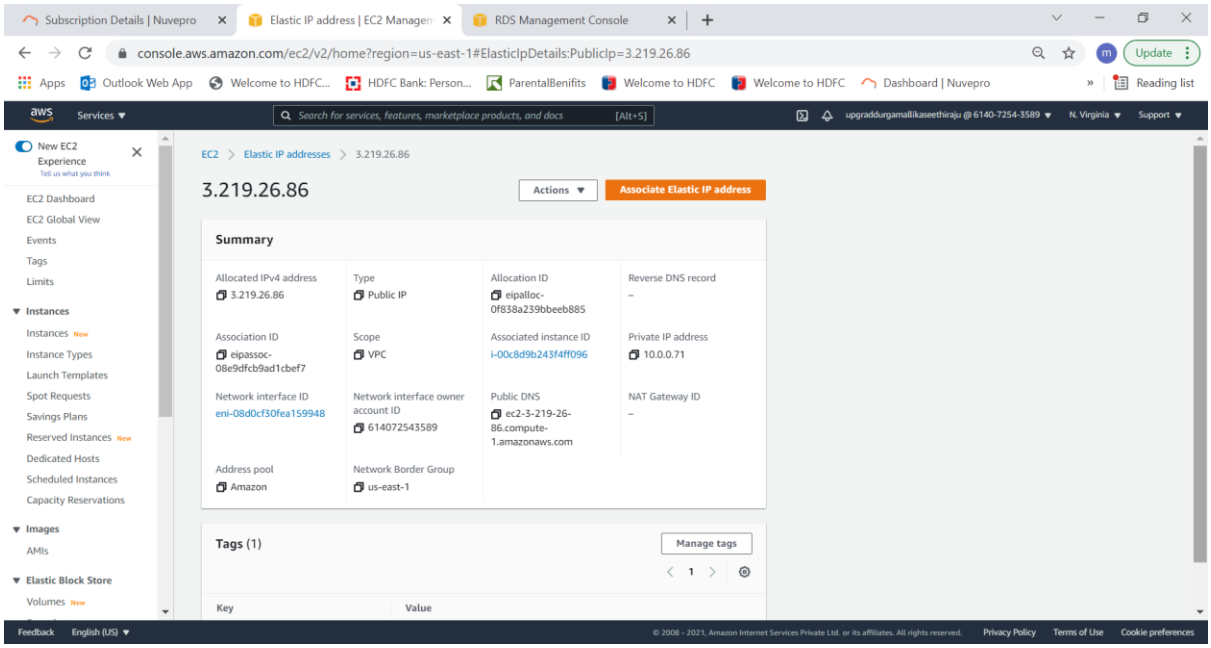
Step 4: Externalized Kafka by creating separate EC2 instance for Kafka with Elastic IP. Inbound rules are set as required and started Zookeeper and Kafka. Also updated the Public IPv4 DNS of kafka instance in KafkaConfig file in Config folder of booking service and Consumer file in notification service.



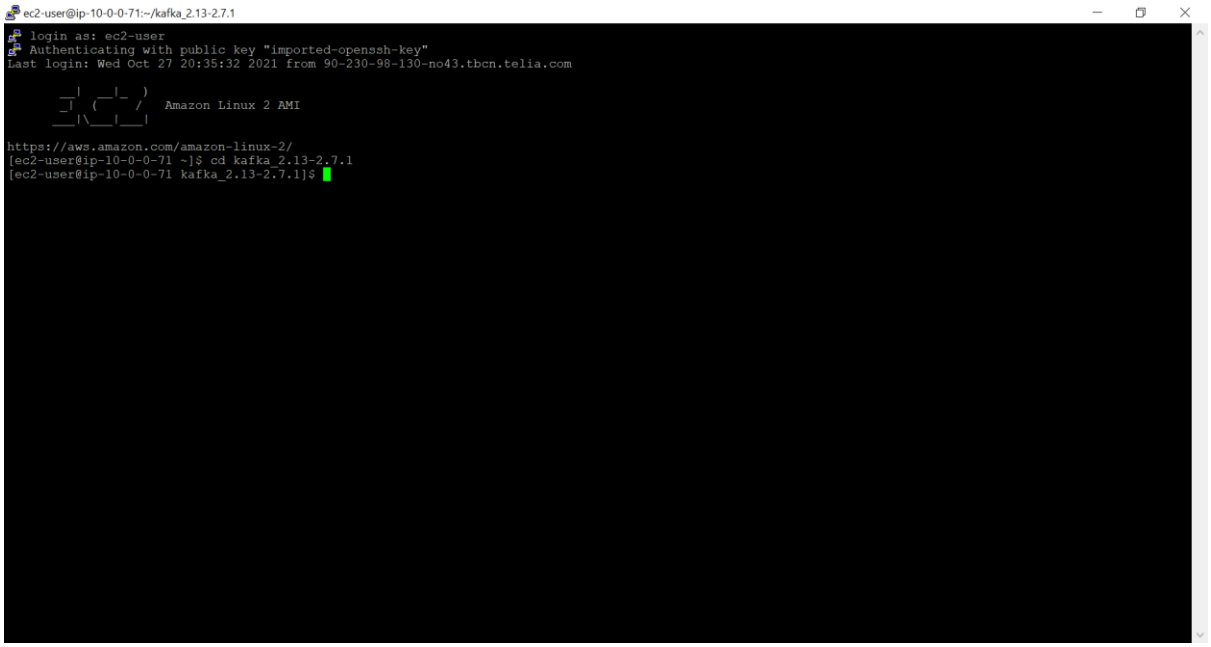




Kafka Elastic Ip Address:



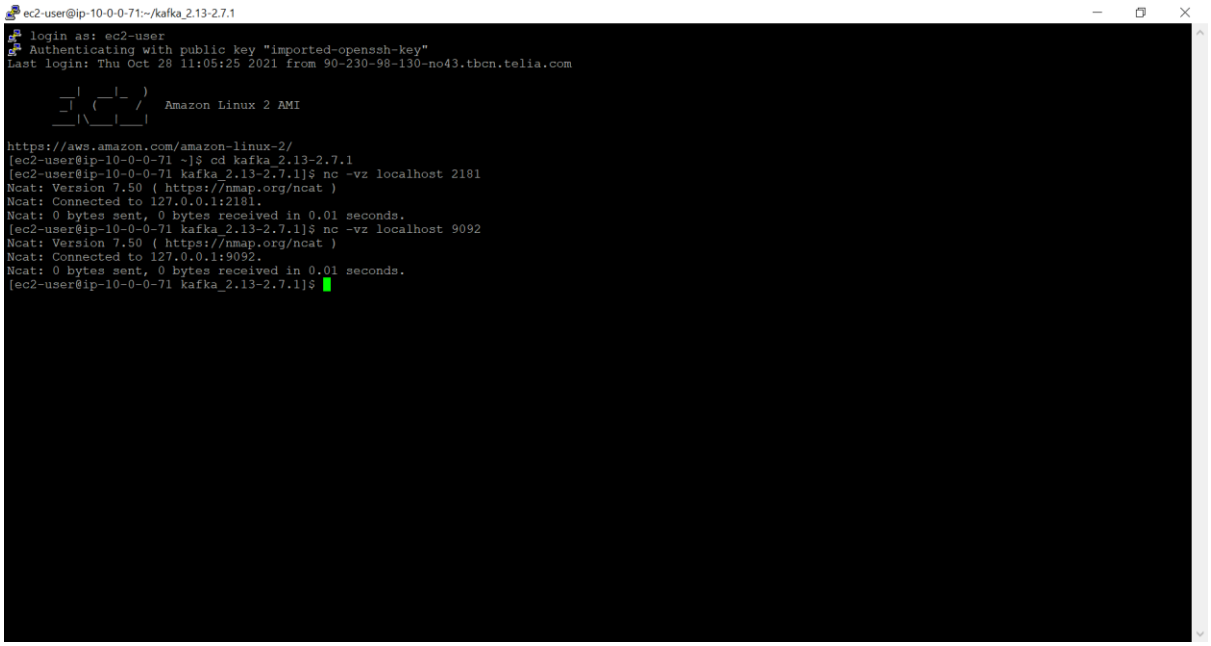
Login to Kafka instance is successful.



Verified whether zookeeper and kafka are started successfully by using commands:

nc -vz localhost 2181

nc -vz localhost 9092



Step 5: application.yml file is set as below for booking service, payment service and Eureka server.

Booking Service:

```
spring:
  cloud:
    loadbalancer:
      ribbon:
        enabled: false
    config:
      discovery:
        serviceId: booking-service
eureka:
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://${EUREKA_HOST_NAME:localhost}:8761/eureka/
  instance:
    hostname: ${EUREKA_HOST_NAME:localhost}
```

Payment service:

```
spring:
  cloud:
    loadbalancer:
      ribbon:
        enabled: false
    config:
      discovery:
        serviceId: payment-service
eureka:
  client:
    register-with-eureka: true
    fetch-registry: true
    service-url:
      defaultZone: http://${EUREKA_HOST_NAME:localhost}:8761/eureka/
  instance:
    hostname: ${EUREKA_HOST_NAME:localhost}
```

EurekaServer:

```
server:
  port: 8761

eureka:
  client:
    register-with-eureka: false
    fetch-registry: false
  instance:
    hostname: ${EUREKA_HOST_NAME:localhost}
```

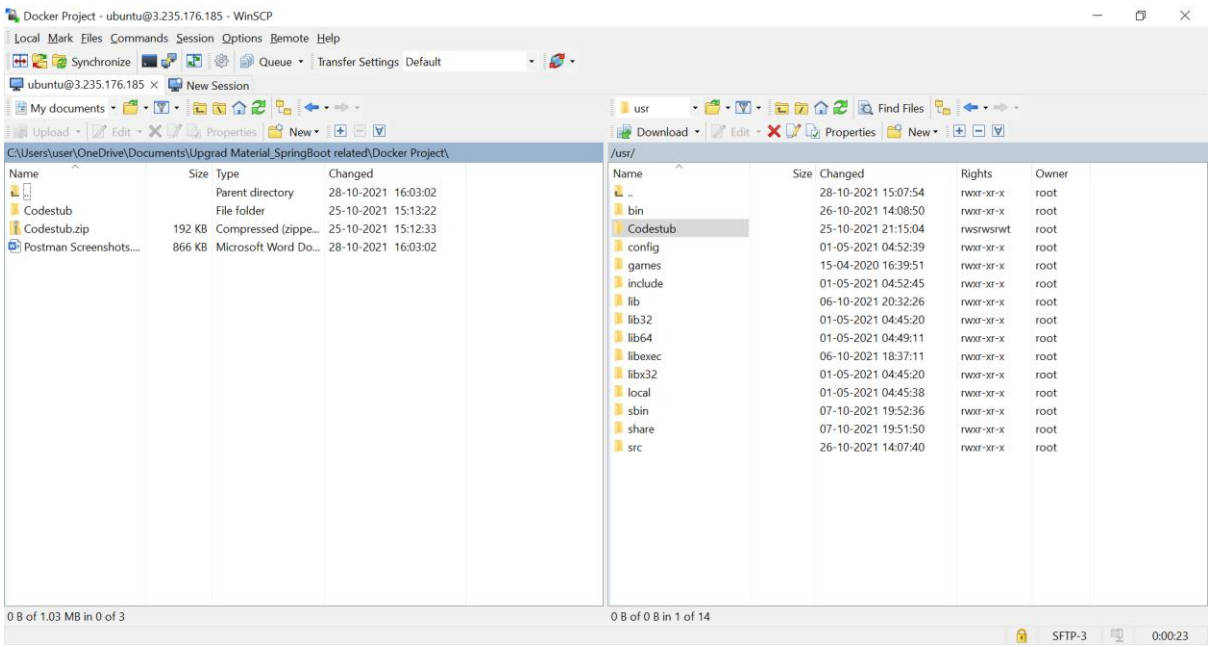
Step 6: Docker file is created for each service and Docker Compose file is created.

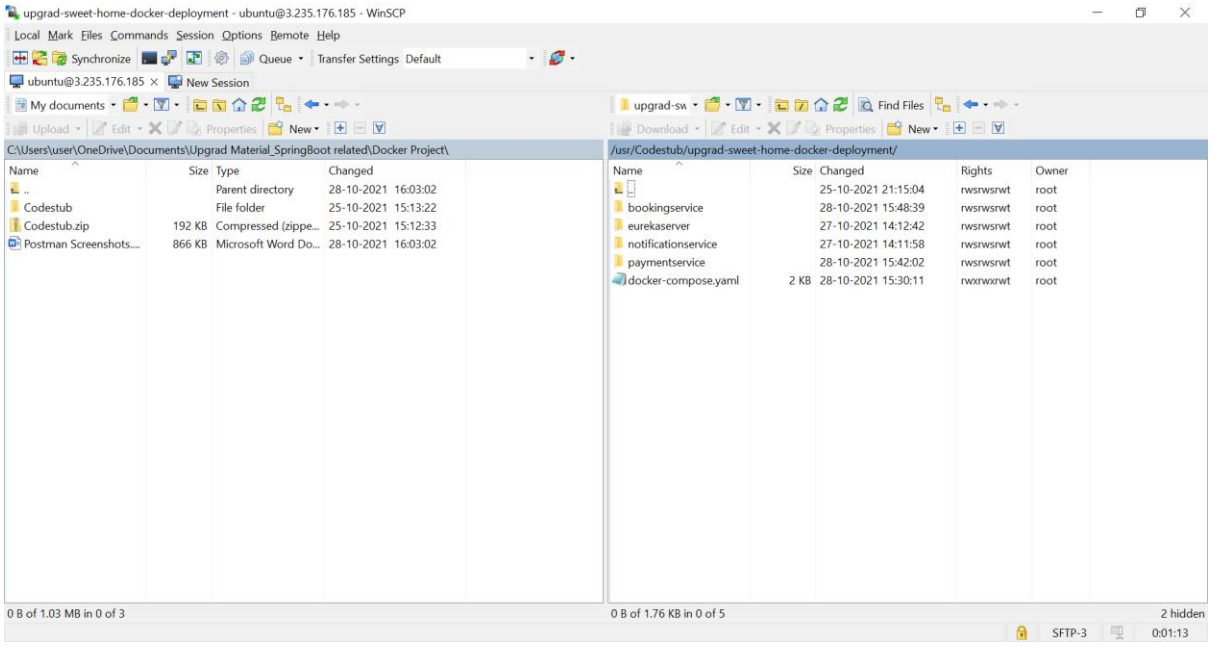
Sample Docker file for Booking service:

```
FROM openjdk:14-jdk-alpine
MAINTAINER upgrad
ADD ./target/bookingService.jar /opt/Codestub/upgrad-sweet-home-docker-deployment/bookingService.jar
WORKDIR /opt/Codestub/upgrad-sweet-home-docker-deployment
ENV PATH="${PATH}:${JAVA_HOME}/bin"
EXPOSE 8080
ENTRYPOINT [ "java", "-jar", "/opt/Codestub/upgrad-sweet-home-docker-deployment/bookingService.jar"]
```

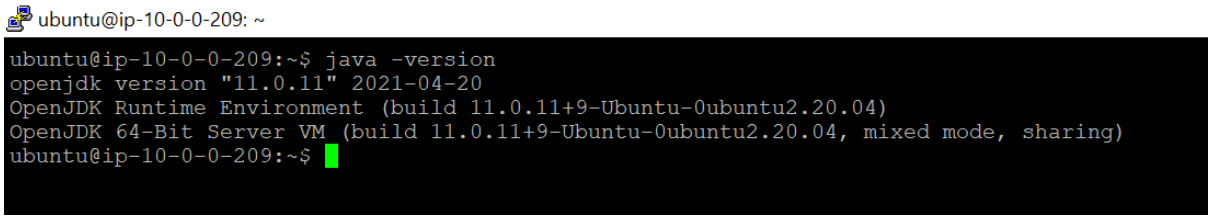
docker-compose.yaml

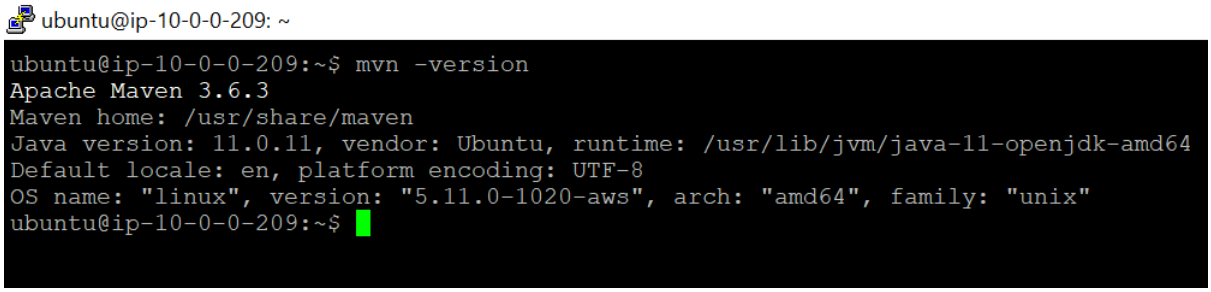
Step 7: All the files are transferred from local machine to EC2 instance (/usr folder) using WinSCP.

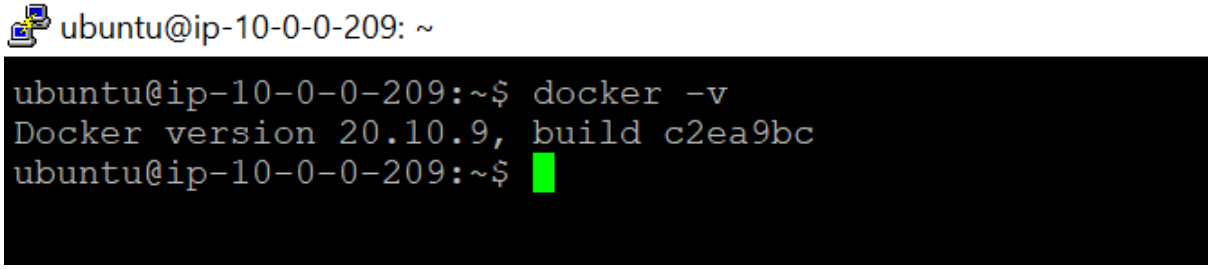




Step 8: Installed java, maven and docker on EC2 instance.







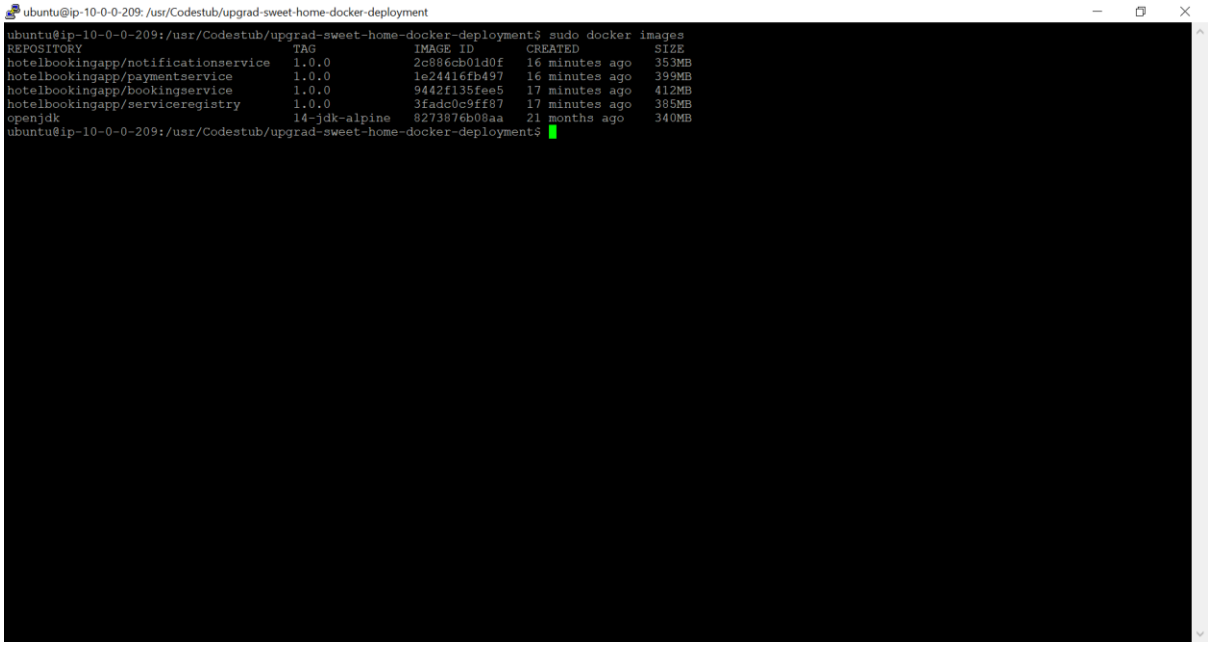
Step 9: Created Jar files for all the applications by using the command "**mvn clean install -DskipTests**" and verified whether jar files are created or not by using the command "**find . -name *.jar**".

Steps to create jar files for each microservice:

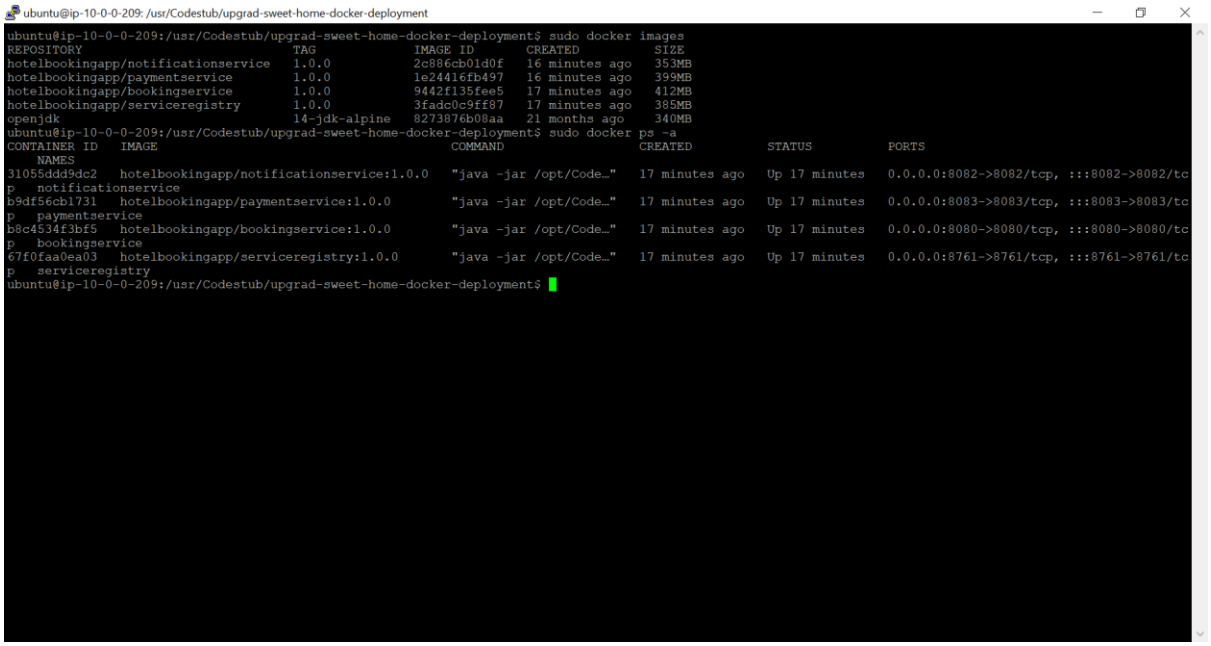
1) Login to ubuntu server and go to folder usr/Codestub/upgrad-sweet-home-docker-deployment
2) Go to the respective microservice for which you want to create the jar file by giving the command “cd bookingservice”.
3) Run the command “**mvn clean install -DskipTests”**.
4) Go to target folder by using “cd target” and check whether the jar is created for bookingservice or not.
5) Repeat the same procedure for all the microservices for which you want to create the jars.

Step 10: Created docker images and containers by using “sudo docker-compose up -d” and verified whether docker images and containers are created and running or not.

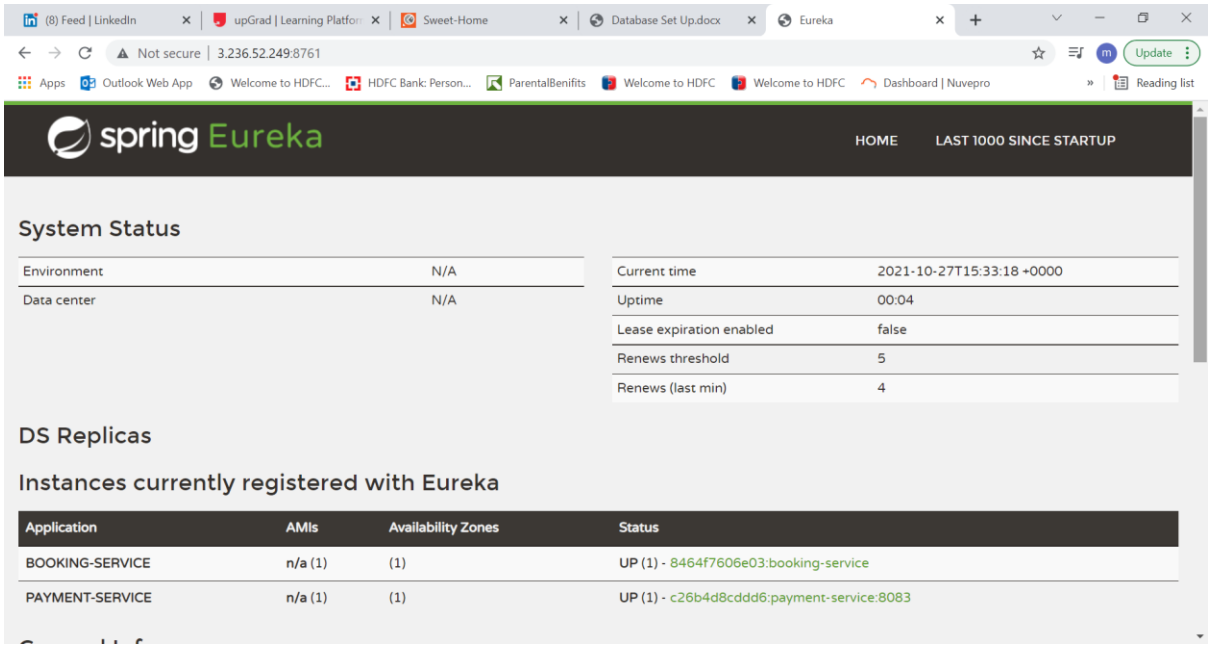
Sudo docker images:



Sudo docker ps -a:



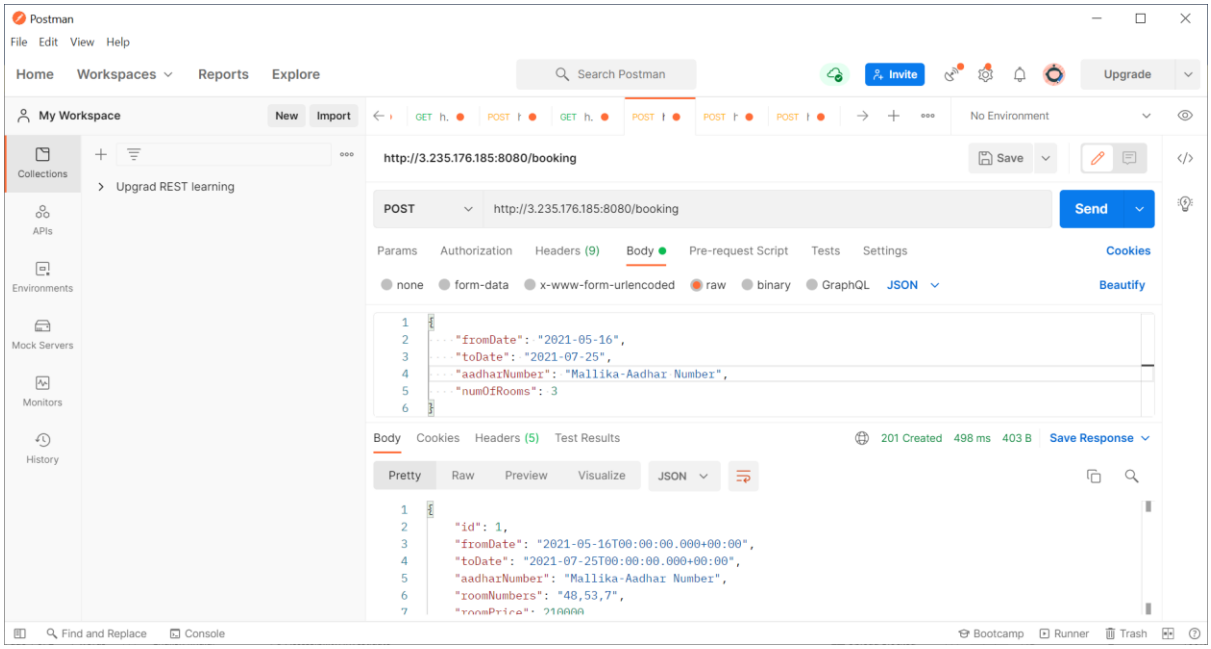
Step 11: Checked whether all the applications are registered in Eureka service registry or not.



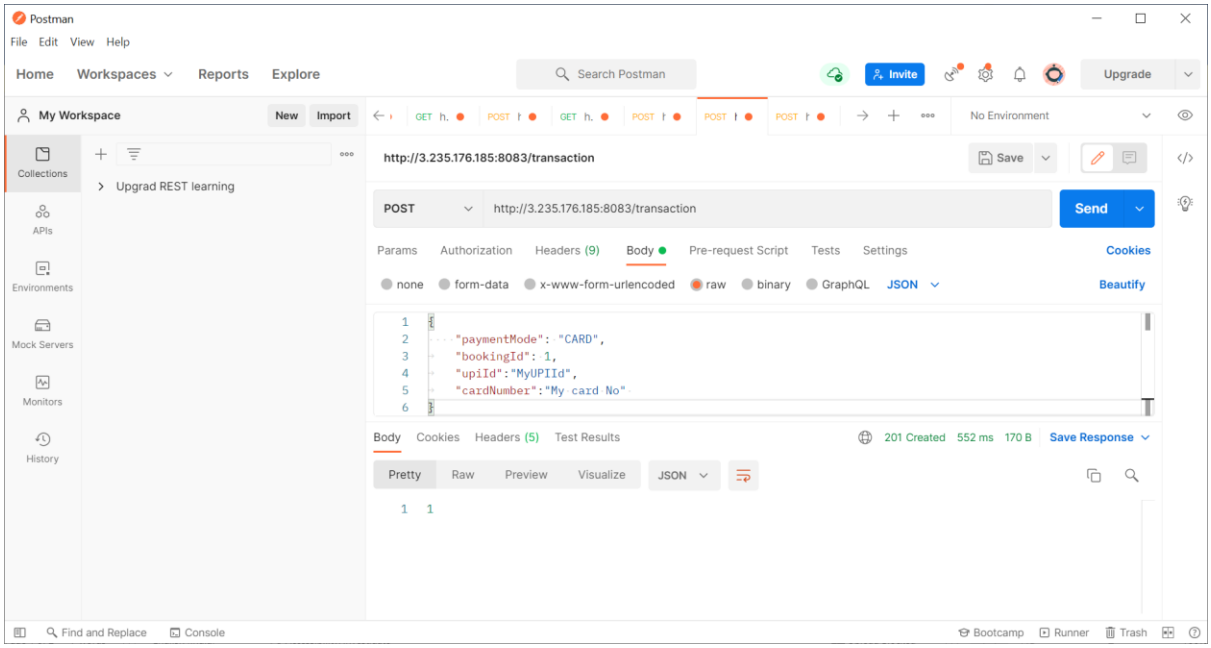
Step 12: Verified whether the application is working correctly by posting the request through postman and also verified the logs through EC2 instance once the booking is created.

Postman Screenshots:

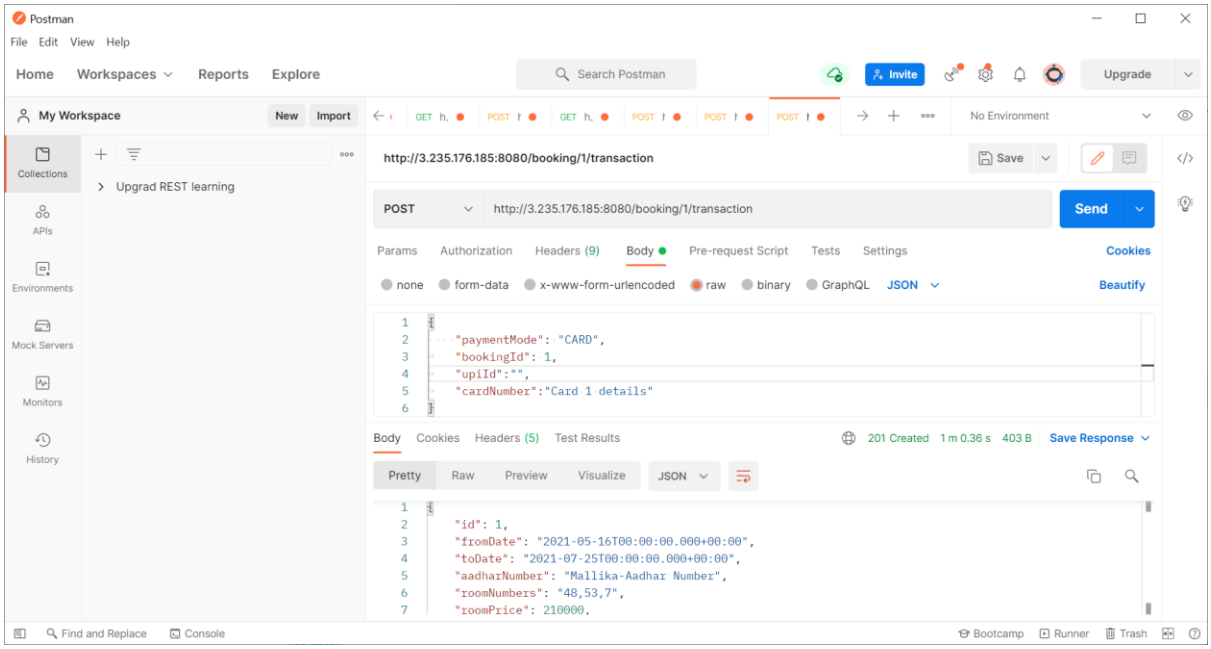
http:// 3.235.176.185:8080/booking



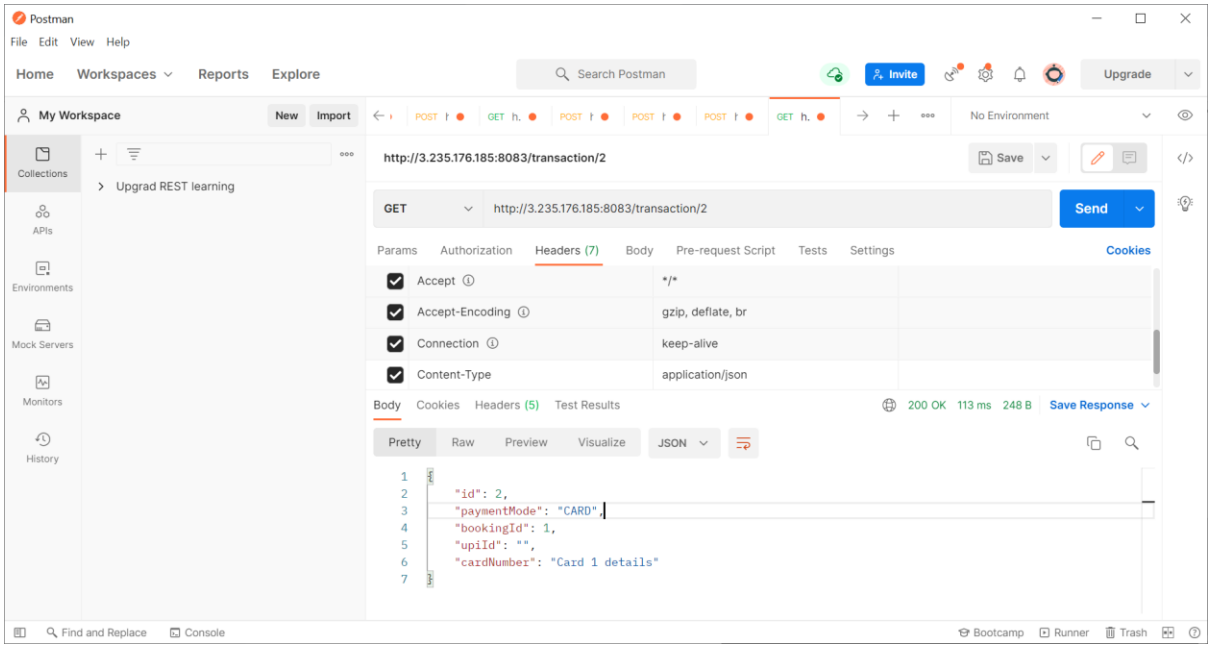
http:// 3.235.176.185:8083/transaction



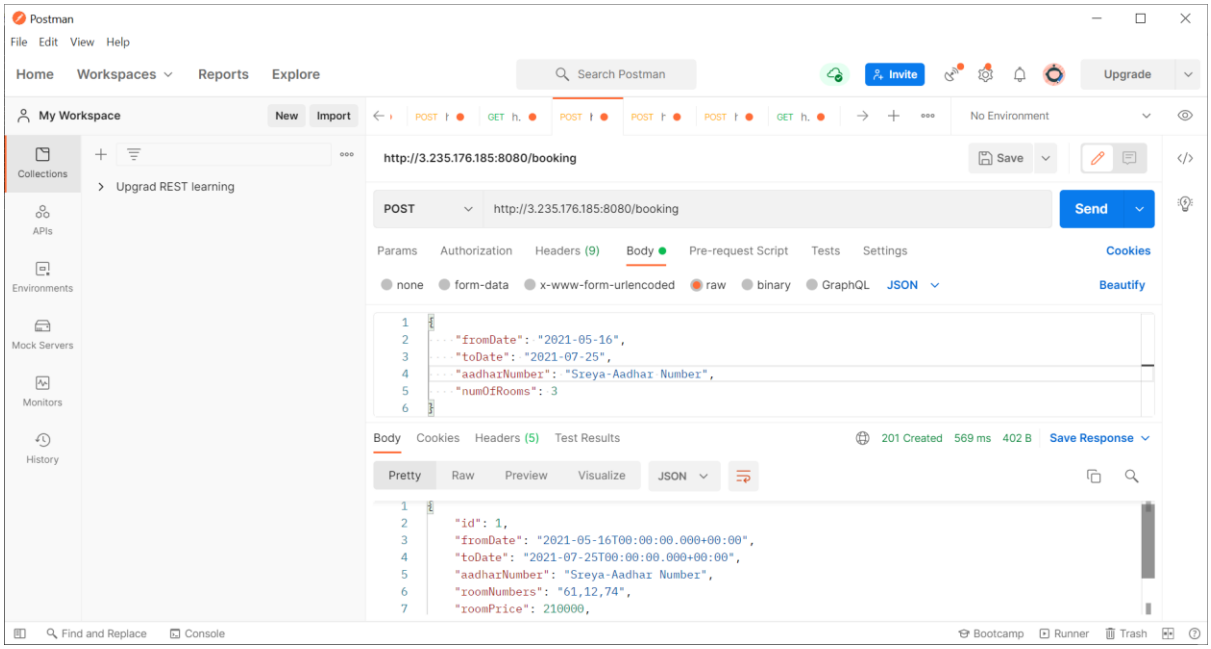
http://3.235.176.185:8080/booking/1/transaction

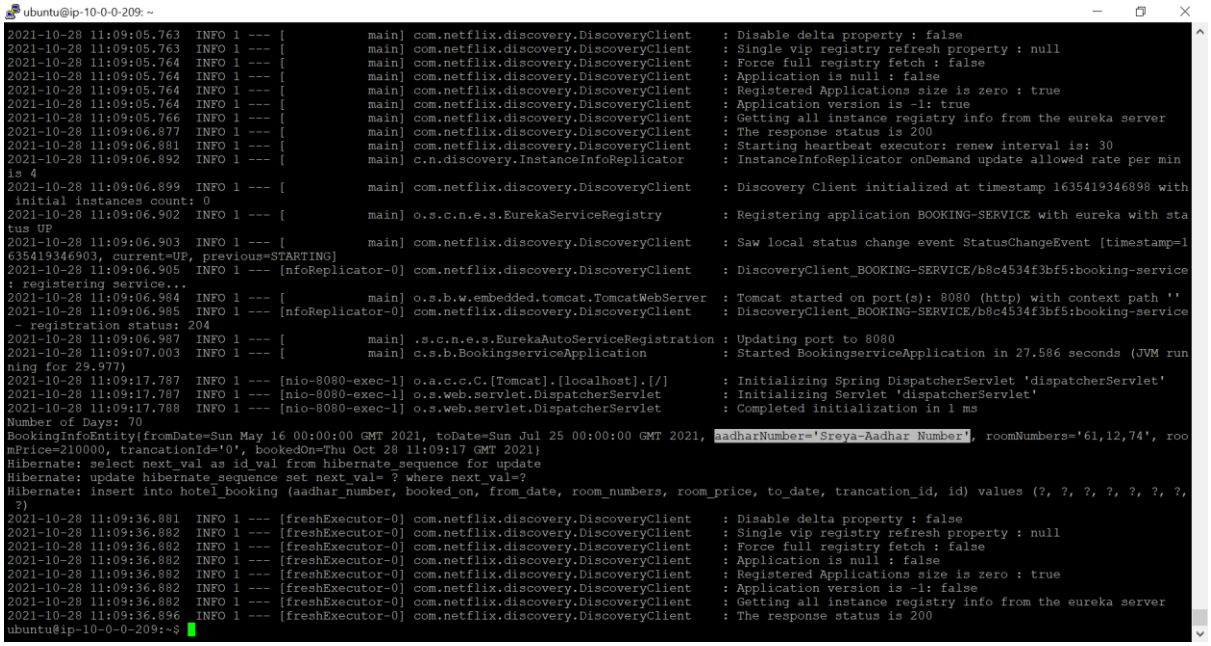


http://3.235.176.185:8083/transaction/2

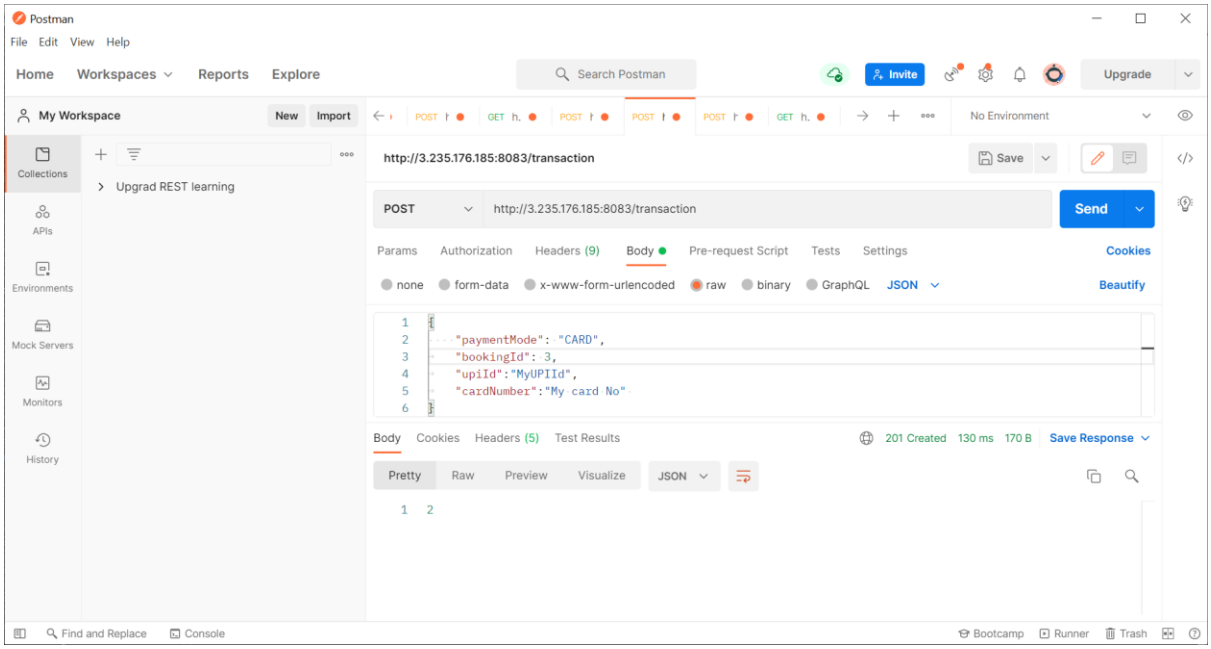


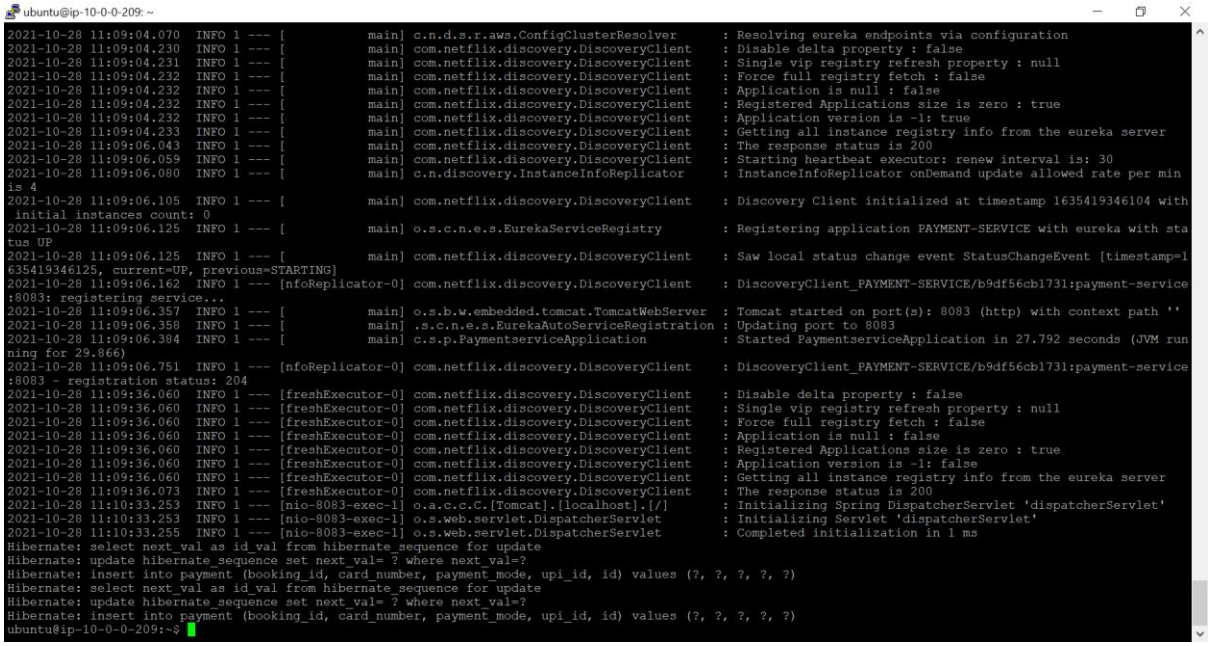
Bookingservice Logs:



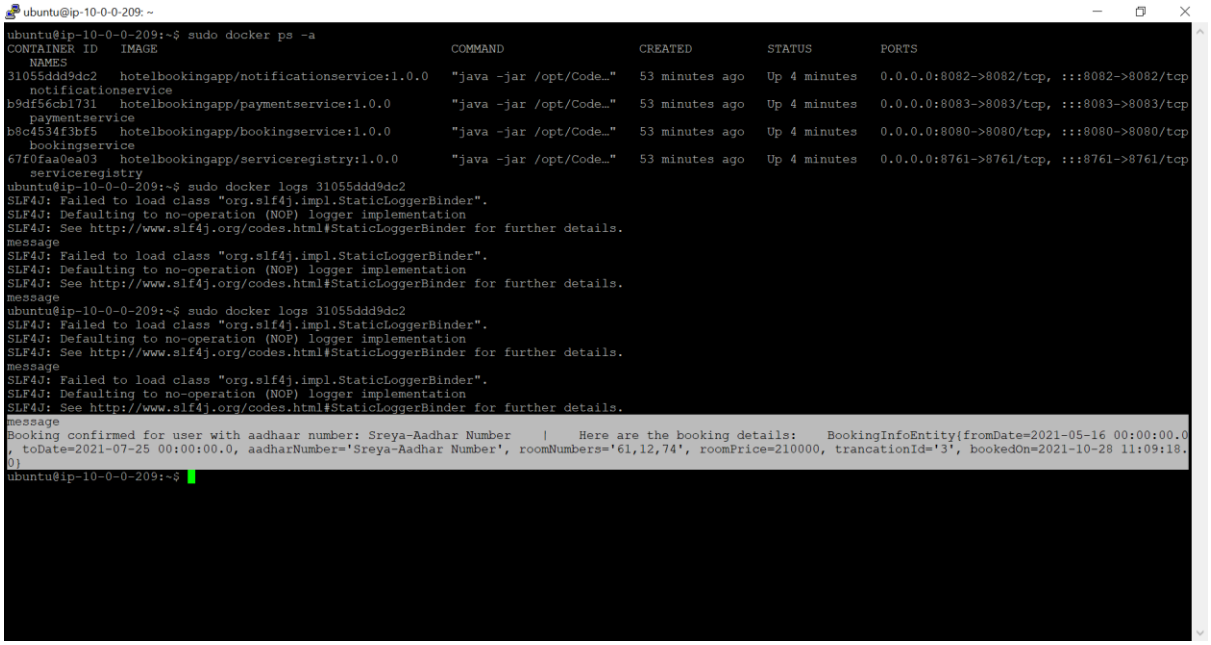


Payment service logs:


Postman
File Edit View Help
Home Workspaces Reports Explore
Search Postman
Invite
Upgrade
My Workspace
New Import
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
History
Upgrad REST learning
http://3.235.176.185:8083/transaction
Save
POST
http://3.235.176.185:8083/transaction
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
1
2 "paymentMode": "CARD",
3 "bookingId": 3,
4 "upiId":"MyUPIId",
5 "cardNumber":"My card No"
6
Body Cookies Headers (5) Test Results
201 Created 130 ms 170 B
Save Response
Pretty Raw Preview Visualize JSON
1 2
Find and Replace
Console
Bootcamp
Runner
Trash


ubuntu@ip-10-0-0-209: ~
2021-10-28 11:09:04.070 INFO 1 --- [main] c.n.d.s.r.aws.ConfigClusterResolver : Resolving eureka endpoints via configuration
2021-10-28 11:09:04.230 INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Disable delta property : false
2021-10-28 11:09:04.231 INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Single vip registry refresh property : null
2021-10-28 11:09:04.232 INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Force full registry fetch : false
2021-10-28 11:09:04.232 INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Application is null : false
2021-10-28 11:09:04.232 INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Registered Applications size is zero : true
2021-10-28 11:09:04.232 INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Application version is -1: true
2021-10-28 11:09:04.233 INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Getting all instance registry info from the eureka server
2021-10-28 11:09:06.043 INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : The response status is 200
2021-10-28 11:09:06.059 INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Starting heartbeat executor: renew interval is: 30
2021-10-28 11:09:06.080 INFO 1 --- [main] c.n.discovery.InstanceInfoReplicator : InstanceInfoReplicator onDemand update allowed rate per min is 4
2021-10-28 11:09:06.105 INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Discovery Client initialized at timestamp 1635419346104 with initial instances count: 0
2021-10-28 11:09:06.125 INFO 1 --- [main] o.s.c.n.e.s.EurekaServiceRegistry : Registering application PAYMENT-SERVICE with eureka with status UP
2021-10-28 11:09:06.125 INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Saw local status change event StatusChangeEvent [timestamp=1635419346125, current=UP, previous=STARTING]
2021-10-28 11:09:06.162 INFO 1 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryClient : DiscoveryClient_PAYMENT-SERVICE/b9df56cb1731:payment-service:8083: registering service...
2021-10-28 11:09:06.357 INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8083 (http) with context path ''
2021-10-28 11:09:06.358 INFO 1 --- [main] .s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8083
2021-10-28 11:09:06.384 INFO 1 --- [main] c.s.p.PaymentserviceApplication : Started PaymentserviceApplication in 27.792 seconds (JVM running for 29.866)
2021-10-28 11:09:06.751 INFO 1 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryClient : DiscoveryClient_PAYMENT-SERVICE/b9df56cb1731:payment-service:8083 - registration status: 204
2021-10-28 11:09:36.060 INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : Disable delta property : false
2021-10-28 11:09:36.060 INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : Single vip registry refresh property : null
2021-10-28 11:09:36.060 INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : Force full registry fetch : false
2021-10-28 11:09:36.060 INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : Application is null : false
2021-10-28 11:09:36.060 INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : Registered Applications size is zero : true
2021-10-28 11:09:36.060 INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : Application version is -1: false
2021-10-28 11:09:36.060 INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : Getting all instance registry info from the eureka server
2021-10-28 11:09:36.073 INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : The response status is 200
2021-10-28 11:10:33.253 INFO 1 --- [nio-8083-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2021-10-28 11:10:33.253 INFO 1 --- [nio-8083-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-10-28 11:10:33.255 INFO 1 --- [nio-8083-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
Hibernate: select next_val as id_val from hibernate_sequence for update
Hibernate: update hibernate_sequence set next_val= ? where next_val=?
Hibernate: insert into payment (booking_id, card_number, payment_mode, upi_id, id) values (?, ?, ?, ?, ?)
Hibernate: select next_val as id_val from hibernate_sequence for update
Hibernate: update hibernate_sequence set next_val= ? where next_val=?
Hibernate: insert into payment (booking_id, card_number, payment_mode, upi_id, id) values (?, ?, ?, ?, ?)
ubuntu@ip-10-0-0-209:~$

Notification service logs:



Eureka Server logs: