

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 codebase 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:

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR (Network border group)	IPv6 pool
my_vpc_docker	vpc-0fed86470537737cb	Available	10.0.0.0/16	-	-

Docker EC2 instance:

The screenshot shows the AWS EC2 Management Console interface. At the top, there are three tabs: "Subscription Details | Nuvapro", "Instances | EC2 Management Console", and "RDS Management Console". Below the tabs, the address bar shows the URL: "console.aws.amazon.com/ec2/v2/home?region=us-east-1#instances:". The navigation bar includes links for "Apps", "Outlook Web App", "Welcome to HDFC...", "HDFC Bank: Person...", "ParentalBenefits", "Welcome to HDFC", "Welcome to HDFC", "Dashboard | Nuvapro", and a "Reading list". On the left, the sidebar shows the "New EC2 Experience" and lists categories like "EC2 Dashboard", "EC2 Global View", "Events", "Tags", "Limits", "Instances" (selected), "Images", and "Elastic Block Store". Under "Instances", "Instances New" is selected. The main content area displays a table of instances with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, and Pulse. Two instances are listed: "Docker" (running, t2.medium, us-east-1f, ec2-3-235-176-185.co..., 3.2) and "Kafka" (running, t2.medium, us-east-1f, ec2-3-219-26-86.comp..., 3.2). A detailed view of the "Docker" instance is shown in a modal window, including its private IPv4 DNS (ip-10-0-0-209.ec2.internal), VPC ID (vpc-0fed86470537737cb (my_vpc_docker)), subnet ID (subnet-0c28c26d7b638b198 (Public subnet)), and instance details (AMI ID: AMI-20210615-0001, Platform: Amazon Linux 2). The bottom right of the modal shows monitoring and log metrics.

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:

The screenshot shows the AWS Security Groups page. The top navigation bar is identical to the previous screenshot. The sidebar on the left shows the "Instances" section with "Instances New" selected. The main content area shows a security group named "sg-00a16615f033ac372 - docker". The "Details" tab is selected, showing information such as Security group name (docker), Security group ID (sg-00a16615f033ac372), Description (set inbound rules), Owner (614072543589), Inbound rules count (10 Permission entries), and Outbound rules count (2 Permission entries). Below this, the "Inbound rules" tab is selected, showing a table of 10 rules. The columns are: Name, Security group rule..., IP version, Type, Protocol, Port range, and Source. The rules include various port ranges (e.g., 9191, 8083, 8761, 3506) and protocols (TCP, Custom TCP, MySQL/Aurora).

The screenshot shows the AWS EC2 Management Console with the RDS Management Console tab open. The main view displays the Inbound rules for a specific security group. There are 10 permission entries listed:

Name	Security group rule...	IP version	Type	Protocol	Port range	Source
-	sgr-0622b9b2cd7a7acb	IPv4	Custom TCP	TCP	9191	0.0.0.0/0
-	sgr-0ffff9c68c4ca7d0	IPv4	Custom TCP	TCP	8083	0.0.0.0/0
-	sgr-06d43321a009ae9...	IPv4	Custom TCP	TCP	8761	0.0.0.0/0
-	sgr-096eeb31a98e57a...	IPv4	MySQL/Aurora	TCP	3306	0.0.0.0/0
-	sgr-05b72c15fb3e2bdc	IPv4	All TCP	TCP	0 - 65535	90.230.98.130/3
-	sgr-0e6737afc15cff06	IPv4	Custom TCP	TCP	8081	0.0.0.0/0
-	sgr-0874674cb37206cae	IPv4	Custom TCP	TCP	8082	0.0.0.0/0
-	sgr-00a3a2165bb3f8e6b	IPv4	Custom TCP	TCP	9092	0.0.0.0/0
-	sgr-0a1d0d0b9ff4b3db	IPv4	Custom TCP	TCP	8080	0.0.0.0/0
-	sgr-0fe2a903c0e98d5a8	IPv4	HTTPS	TCP	443	0.0.0.0/0

Login to ubuntu machine is successful

EC2 Login as ubuntu user:

```
ubuntu@ip-10-0-0-209: ~
[1] login as: ubuntu
[2] Authenticating with public key "imported-openssh-key"
[3] Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.11.0-1020-aws x86_64)

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

System information as of Thu Oct 28 10:52:25 UTC 2021

System load:          0.0
Usage of /:           86.8% of 7.69GB
Memory usage:         29%
Swap usage:           0%
Processes:            133
Users logged in:     0
IPv4 address for br-2013c64b361c: 172.19.0.1
IPv4 address for br-25630091235c: 172.20.0.1
IPv4 address for br-acc8e087368b: 172.21.0.1
IPv4 address for br-b8472d94d62: 172.18.0.1
IPv4 address for docker0:    172.17.0.1
IPv4 address for eth0:      10.0.0.209

=> / is using 86.8% of 7.69GB

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

  https://ubuntu.com/aws/pro

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

Last login: Thu Oct 28 09:38:32 2021 from 90.230.98.130
ubuntu@ip-10-0-0-209:~$
```

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.

The screenshot displays two windows side-by-side. On the left is the AWS RDS Management Console showing the 'sweethomebooking' database details. The 'Summary' tab is selected, displaying metrics like CPU usage (11.15%), Status (Modifying), and Engine (MySQL Community). The 'Connectivity & security' tab shows the endpoint (sweethomebooking.cwoz5gwd63ts.us-east-1.rds.amazonaws.com), port (3306), networking (Availability Zone: us-east-1f), and security (VPC: vpc-0094ed777ea1e1f89, Subnet group: default-vpc-0094ed777ea1e1f89). On the right is MySQL Workbench showing the 'Sweet-Home' schema. The 'Schemas' tree shows 'SweetHomeBooking' and 'SweetHomePayment'. The 'Table: hotel_booking' is selected, showing its columns: id, aadhar_number, booked_on, from_date, room_numbers, room_price, to_date, and truncation_id. A query in 'Query 1' runs the command 'show databases;', and the result grid shows the databases: SweetHomeBooking, SweetHomePayment, information_schema, mysql, performance_schema, and sys. The 'Result 5' pane shows the execution history of the query.

The screenshot shows the AWS EC2 Management Console with the 'Security Groups' page open. A single security group named 'sg-08abed031682d048e' is listed under 'Inbound rules (2)', which contains two entries: 'All traffic' (IPV4) and 'MySQL/Aurora' (TCP port 3306).

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.

The screenshot shows the AWS EC2 Management Console with the 'Instances' page open. Two instances are listed: 'Docker' and 'Kafka'. The 'Kafka' instance is selected, showing its details. The 'Public IPv4 DNS' field for the Kafka instance is listed as 'ec2-3-219-26-86.compute-1.amazonaws.com'.

The screenshot shows the AWS EC2 Management Console with the URL <https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#SecurityGroup:groupId=sg-0572e5cd781616472>. The page displays the details of a security group named "sg-0572e5cd781616472 - launch-wizard-1". The "Outbound rules" tab is selected, showing one rule: "sgr-069d32e905e510..." with IP version IPv4, type All traffic, protocol All, port range All, and destination 0.0.0.0/0. The "Details" section includes fields for Security group ID (sg-0572e5cd781616472), Description (launch-wizard-1 created 2021-10-26T18:32:10.336+05:30), Owner (614072543589), Inbound rules count (5 Permission entries), and Outbound rules count (1 Permission entry). The left sidebar shows navigation links for EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances, Images, and Elastic Block Store.

The screenshot shows the AWS EC2 Management Console with the URL <https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#SecurityGroup:securityGroupId=sg-0572e5cd781616472>. The page displays the details of a security group named "sg-0572e5cd781616472 - launch-wizard-1". The "Inbound rules" tab is selected, showing five rules: "sgr-004e36e4f0f686902", "sgr-07b37959c26aae1...", "sgr-0ac6b069696497ab", "sgr-0a9dbfbff8de7dfef2", and "sgr-073b49f61a34025ea", all with IP version IPv4, type Custom TCP, protocol TCP, port range 2181, 80, 443, 22, and 9092 respectively, and source 0.0.0.0/0. A message at the top says "You can now check network connectivity with Reachability Analyzer". The "Details" section includes fields for Security group ID (sg-0572e5cd781616472), Description (launch-wizard-1 created 2021-10-26T18:32:10.336+05:30), Owner (614072543589), Inbound rules count (5 Permission entries), and Outbound rules count (1 Permission entry). The left sidebar shows navigation links for EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances, Images, and Elastic Block Store.

Kafka Elastic Ip Address:

The screenshot shows the AWS EC2 Elastic IP Management console. The main view displays the summary of an assigned IPv4 address (3.219.26.86). Key details include:

- Type:** Public IP
- Allocation ID:** eipalloc-0f838a239bbeb885
- Scope:** VPC
- Associated instance ID:** i-00c8d9b243f4ff096
- Private IP address:** 10.0.0.71
- Network interface ID:** eni-08d0cf50fea159948
- Network interface owner account ID:** 614072543589
- Public DNS:** ec2-3-219-26-86.compute-1.amazonaws.com
- NAT Gateway ID:** -
- Address pool:** Amazon
- Network Border Group:** us-east-1

Below the summary, there is a section for **Tags (1)** with one tag listed: **Key:** **Value:**

Login to Kafka instance is successful.

```

ec2-user@ip-10-0-0-71:~/kafka_2.13-2.7.1
[ec2-user@login ~]$ login as: ec2-user
[ec2-user@login ~]$ Authenticating with public key "imported-openssh-key"
Last login: Wed Oct 27 20:35:32 2021 from 90-230-98-130-no43.tbcn.telia.com
[ec2-user@login ~]$
[ec2-user@login ~]$ curl https://aws.amazon.com/amazon-linux-2/
[ec2-user@login ~]$ cd kafka_2.13-2.7.1
[ec2-user@login ~]$ ./kafka-server-start.sh config/server.properties
[ec2-user@login ~]$ nc -vz localhost 2181
[ec2-user@login ~]$ nc -vz localhost 9092
[ec2-user@login ~]$

```

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

nc -vz localhost 2181

nc -vz localhost 9092

```

ec2-user@ip-10-0-0-71:~/kafka_2.13-2.7.1
Login as: ec2-user
Authenticating with public key "imported-openssh-key"
Last login: Thu Oct 28 11:05:25 2021 from 90-230-98-130-no43.tbcn.telia.com
[ec2-user@ip-10-0-0-71 ~]$ cd kafka_2.13-2.7.1
[ec2-user@ip-10-0-0-71 kafka_2.13-2.7.1]$ nc -vz localhost 2181
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 127.0.0.1:2181
Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.
[ec2-user@ip-10-0-0-71 kafka_2.13-2.7.1]$ nc -vz localhost 9092
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 127.0.0.1:9092
Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.
[ec2-user@ip-10-0-0-71 kafka_2.13-2.7.1]$ 

```

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.

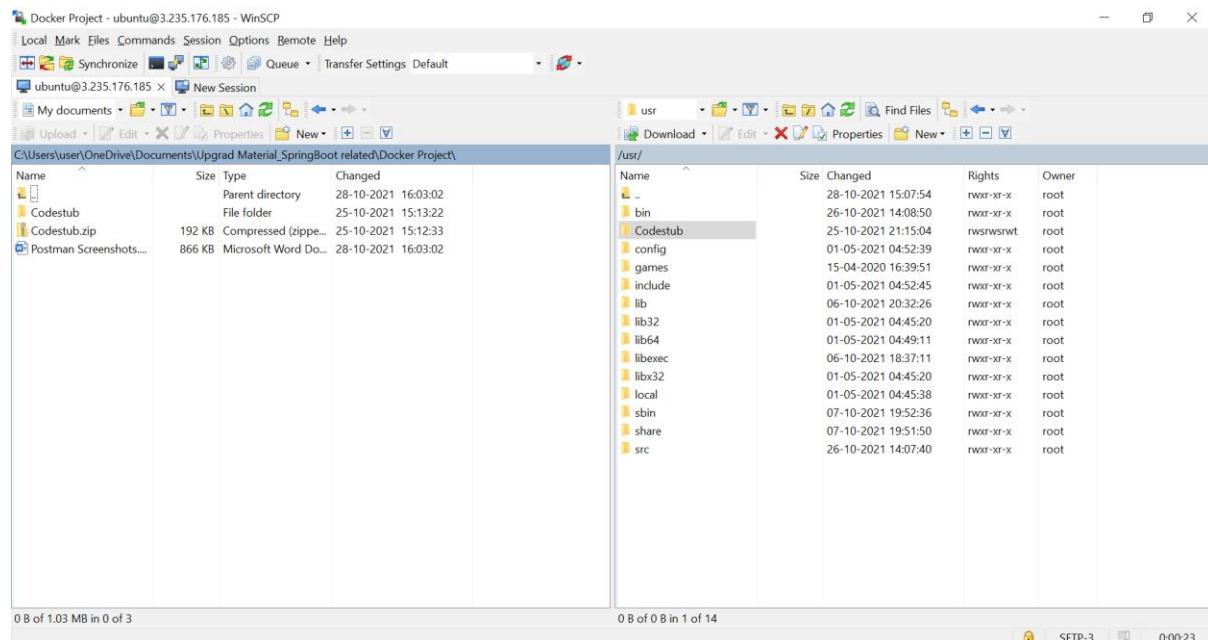
Sample Docker file for Booking service:

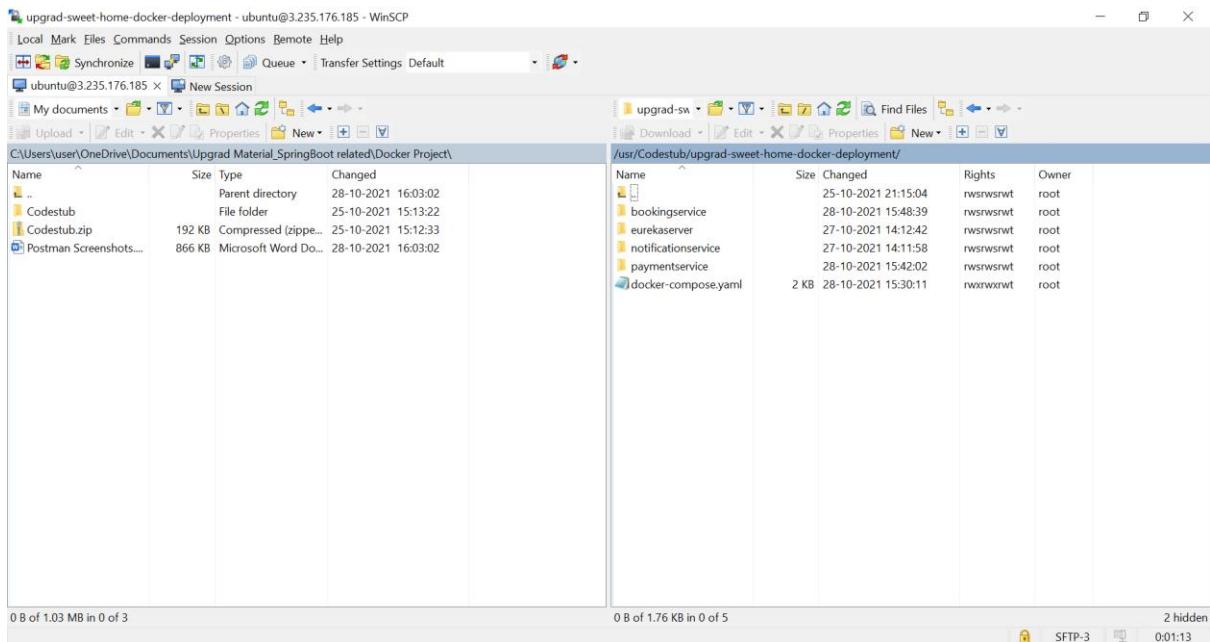
```
FROM openjdk:14-jdk-alpine
MAINTAINER upgrad
ADD ./target/bookingService.jar /opt/Codestub/upgrad-sweet-home-docker-deployment/bookingService.jar
WORKDIR /opt/Codestub/upgrad-sweet-home-docker-deployment
ENV PATH="${PATH}:${JAVA_HOME}/bin"
EXPOSE 8080
ENTRYPOINT [ "java", "-jar", "/opt/Codestub/upgrad-sweet-home-docker-deployment/bookingService.jar"]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



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.

```
ubuntu@ip-10-0-0-209:~$ java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.20.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04, mixed mode, sharing)
ubuntu@ip-10-0-0-209:~$
```

```
ubuntu@ip-10-0-0-209:~$ mvn -version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 11.0.11, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "5.11.0-1020-aws", arch: "amd64", family: "unix"
ubuntu@ip-10-0-0-209:~$
```

```
ubuntu@ip-10-0-0-209:~$ docker -v
Docker version 20.10.9, build c2ea9bc
ubuntu@ip-10-0-0-209:~$
```

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.

```
 ubuntu@ip-10-0-0-209: /usr/Codestub/upgrad-sweet-home-docker-deployment
ubuntu@ip-10-0-0-209:/usr/Codestub/upgrad-sweet-home-docker-deployment$ find . -name *.jar
./eurekaserver/target/eurekaServer.jar
./eurekaserver/.mvn/wrapper/maven-wrapper.jar
./bookingservice/target/bookingservice.jar
./bookingservice/.mvn/wrapper/maven-wrapper.jar
./notificationservice/target/notificationService.jar
./notificationservice/target/notificationService-jar-with-dependencies.jar
./paymentservice/target/paymentService.jar
./paymentservice/.mvn/wrapper/maven-wrapper.jar
ubuntu@ip-10-0-0-209:/usr/Codestub/upgrad-sweet-home-docker-deployment$
```

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:

```
 ubuntu@ip-10-0-0-209: /usr/Codestub/upgrad-sweet-home-docker-deployment
ubuntu@ip-10-0-0-209:/usr/Codestub/upgrad-sweet-home-docker-deployment$ sudo docker images
REPOSITORY          TAG      IMAGE ID   CREATED     SIZE
hotelbookingapp/notificationservice  1.0.0    2c886cb01d0f  16 minutes ago  353MB
hotelbookingapp/paymentservice       1.0.0    1e24416fb497  16 minutes ago  399MB
hotelbookingapp/bookingservice       1.0.0    9442f135fee5  17 minutes ago  412MB
hotelbookingapp/serviceregistry     1.0.0    3fad0c9ff87   17 minutes ago  385MB
openjdk              14-jdk-alpine  8273876b08aa  21 months ago  340MB
ubuntu@ip-10-0-0-209:/usr/Codestub/upgrad-sweet-home-docker-deployment$
```

Sudo docker ps -a:

```

ubuntu@ip-10-0-0-209:/usr/Codestub/upgrad-sweet-home-docker-deployment
ubuntu@ip-10-0-0-209:/usr/Codestub/upgrad-sweet-home-docker-deployment$ sudo docker images
REPOSITORY          TAG      IMAGE ID            CREATED             SIZE
hotelbookingapp/notificationservice  1.0.0    2e986cb01d0f  16 minutes ago   353MB
hotelbookingapp/paymentservice       1.0.0    1e24416fb497  16 minutes ago   399MB
hotelbookingapp/bookingservice      1.0.0    9442f135fe65  17 minutes ago   412MB
hotelbookingapp/serviceregistry     1.0.0    3fadcc0c9ef07  17 minutes ago   385MB
openjdk               14-jdk-alpine  8273876b08aa  21 months ago   340MB
ubuntu@ip-10-0-0-209:/usr/Codestub/upgrad-sweet-home-docker-deployment$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
 NAMES
31055ddd9dc2      hotelbookingapp/notificationservice:1.0.0   "java -jar /opt/Code..."   17 minutes ago   Up 17 minutes   0.0.0.0:8082->8082/tcp, :::8082->8082/tcp
b9df56cb1731      hotelbookingapp/paymentservice:1.0.0     "java -jar /opt/Code..."   17 minutes ago   Up 17 minutes   0.0.0.0:8083->8083/tcp, :::8083->8083/tcp
b8c4534f3bf5      hotelbookingapp/bookingservice:1.0.0     "java -jar /opt/Code..."   17 minutes ago   Up 17 minutes   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp
67fffaa0ea03      hotelbookingapp/serviceregistry:1.0.0     "java -jar /opt/Code..."   17 minutes ago   Up 17 minutes   0.0.0.0:8761->8761/tcp, :::8761->8761/tcp
ubuntu@ip-10-0-0-209:/usr/Codestub/upgrad-sweet-home-docker-deployment$ 

```

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

The screenshot shows a web browser window with multiple tabs open. The active tab is titled 'Eureka' and displays the Spring Eureka service registry interface. The page has a dark header with the 'spring Eureka' logo and navigation links for 'HOME' and 'LAST 1000 SINCE STARTUP'. Below the header, there are two main sections: 'System Status' and 'DS Replicas'.

System Status

Environment	N/A
Data center	N/A

Current time	2021-10-27T15:33:18 +0000
Uptime	00:04
Lease expiration enabled	false
Renews threshold	5
Renews (last min)	4

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
BOOKING-SERVICE	n/a (1)	(1)	UP (1) - 8464f7606e03:booking-service
PAYMENT-SERVICE	n/a (1)	(1)	UP (1) - c26b4d8cddd6:payment-service:8083

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>

Postman

File Edit View Help

Home Workspaces Reports Explore Search Postman

My Workspace New Import http://3.235.176.185:8080/booking

+

Upgrad REST learning

POST http://3.235.176.185:8080/booking

Params Authorization Headers (9) Body Body

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 "fromDate": "2021-05-16",
2 "toDate": "2021-07-25",
3 "aadharNumber": "Mallika-Aadhar Number",
4 "numOfRooms": 3

Body Cookies Headers (5) Test Results 201 Created 498 ms 403 B Save Response

Pretty Raw Preview Visualize JSON

1 "id": 1,
2 "fromDate": "2021-05-16T00:00:00.000+00:00",
3 "toDate": "2021-07-25T00:00:00.000+00:00",
4 "aadharNumber": "Mallika-Aadhar Number",
5 "roomNumbers": "48,53,7",
6 "transPrice": 210000

Find and Replace Console Bootcamp Runner Trash

The screenshot shows the Postman interface with a collection named 'Upgrad REST learning'. A POST request is made to 'http://3.235.176.185:8080/booking' with a JSON body containing room details. The response is a 201 Created status with a response body showing the booking ID and other details.

http:// 3.235.176.185:8083/transaction

Postman

File Edit View Help

Home Workspaces Reports Explore Search Postman

My Workspace New Import http://3.235.176.185:8083/transaction

+

Upgrad REST learning

POST http://3.235.176.185:8083/transaction

Params Authorization Headers (9) Body Body

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 "paymentMode": "CARD",
2 "bookingId": 1,
3 "upiId": "MyUPIId",
4 "cazidNumber": "My card No"

Body Cookies Headers (5) Test Results 201 Created 552 ms 170 B Save Response

Pretty Raw Preview Visualize JSON

1 1

Find and Replace Console Bootcamp Runner Trash

The screenshot shows the Postman interface with a collection named 'Upgrad REST learning'. A POST request is made to 'http://3.235.176.185:8083/transaction' with a JSON body containing payment mode and booking ID. The response is a 201 Created status with a response body showing the status code 1.

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

Postman interface showing a POST request to `http://3.235.176.185:8080/booking/1/transaction`. The request body is:

```

1
2   "paymentMode": "CARD",
3   "bookingId": 1,
4   "upId": "",
5   "cardNumber": "Card 1 details"
6

```

The response status is 201 Created.

http://3.235.176.185:8083/transaction/2

Postman interface showing a GET request to `http://3.235.176.185:8083/transaction/2`. The selected Headers are:

- Accept: */*
- Accept-Encoding: gzip, deflate, br
- Connection: keep-alive
- Content-Type: application/json

The response status is 200 OK.

Bookingservice Logs:

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing 'Collections', 'APIs', 'Environments', 'Mock Servers', 'Monitors', and 'History'. The main area shows a POST request to 'http://3.235.176.185:8080/booking'. The 'Body' tab is selected, displaying the following JSON payload:

```

1
2   ...
3     "fromDate": "2021-05-16",
4     "toDate": "2021-07-25",
5     "aadharNumber": "Sreya-Aadhar Number",
6     "numOfRooms": 3

```

Below the body, the response status is 201 Created with a response time of 569 ms. The response body is also shown in JSON format:

```

1
2   {
3     "id": 1,
4     "fromDate": "2021-05-16T00:00:00.000+00:00",
5     "toDate": "2021-07-25T00:00:00.000+00:00",
6     "aadharNumber": "Sreya-Aadhar Number",
7     "roomNumbers": "61,12,74",
8     "roomPrice": 210000

```

The screenshot shows a terminal window with the title 'ubuntu@ip-10-0-0-209: ~'. The logs are from a Java application using Netflix Discovery Client. Key log entries include:

- INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Disable delta property : false
- INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Single vip registry refresh property : null
- INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Force full registry fetch : false
- INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Application is null : false
- INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Registered Applications size is zero : true
- INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Application version is -1: true
- INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Getting all instance registry info from the eureka server
- INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : The response status is 200
- INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Starting heartbeat executor: renew interval is: 30
- INFO 1 --- [main] c.n.discovery.InstanceInfoReplicator : InstanceInfoReplicator onDemand update allowed rate per min
- INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Discovery Client initialized at timestamp 1635419346898 with initial instances count: 0
- INFO 1 --- [main] o.s.c.n.e.s.EurekaServiceRegistry : Registering application BOOKING-SERVICE with eureka with status UP
- INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Saw local status change event StatusChangeEvent [timestamp=1635419346903, current=UP, previous=STARTING]
- INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : DiscoveryClient_BOOKING-SERVICE/b8c4534f3bf5:booking-service : registering service...
- INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
- INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : DiscoveryClient_BOOKING-SERVICE/b8c4534f3bf5:booking-service : Registration status: 204
- INFO 1 --- [main] .s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8080
- INFO 1 --- [main] c.s.b.BookingserviceApplication : Started BookingserviceApplication in 27.586 seconds (JVM running for 29.977)
- INFO 1 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[] : : Initializing Spring DispatcherServlet 'dispatcherServlet'
- INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : : Initializing Servlet 'dispatcherServlet'
- INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : : Completed initialization in 1 ms
- Number of Days: 70
- BookingInfoEntity{fromDate=Sun May 16 00:00:00 GMT 2021, toDate=Sun Jul 25 00:00:00 GMT 2021, aadharNumber='Sreya-Aadhar Number', roomNumbers='61,12,74', roomPrice=210000, transactionId='0', bookedOn=Thu Oct 28 11:09:17 GMT 2021}
- 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 hotel_booking (aadhar_number, booked_on, from_date, room_numbers, room_price, to_date, transaction_id, id) values (?, ?, ?, ?, ?, ?, ?, ?)
- INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : Disable delta property : false
- INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : Single vip registry refresh property : null
- INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : Force full registry fetch : false
- INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : Application is null : false
- INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : Registered Applications size is zero : true
- INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : Application version is -1: false
- INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : Getting all instance registry info from the eureka server
- INFO 1 --- [freshExecutor-0] com.netflix.discovery.DiscoveryClient : The response status is 200

Payment service logs:

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections for Collections, APIs, Environments, Mock Servers, Monitors, and History. The main area displays a POST request to the URL `http://3.235.176.185:8083/transaction`. The request body is set to JSON and contains the following data:

```

1  ...
2  ...
3  ...
4  ...
5  ...
6  ...

```

The response status is 201 Created, with a timestamp of 130 ms and a size of 170 B. Below the response, there are tabs for Body, Cookies, Headers (5), Test Results, Pretty, Raw, Preview, Visualize, and JSON.

```

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 11:09:04.233 INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : The response status is 200
2021-10-28 11:09:06.043 INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Starting heartbeat executor: renew interval is: 30
2021-10-28 11:09:06.059 INFO 1 --- [main] c.n.d.s.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 sta
tus UP
2021-10-28 11:09:06.125 INFO 1 --- [main] com.netflix.discovery.DiscoveryClient : Saw local status change event StatusChangeEvent [timestamp=1
635419346125, 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 run
ning 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 : Disabling 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 11:09:36.253 INFO 1 --- [nio-8083-exec-1] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2021-10-28 11:09:33.253 INFO 1 --- [nio-8083-exec-1] o.s.web.servlet.DispatcherServlet : : Initialization Servlet 'dispatcherServlet'
2021-10-28 11:09:33.255 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:

```
ubuntu@ip-10-0-0-209:~$ sudo docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
 NAMES
31055ddd9dc2 hotelbookingapp/notificationservice:1.0.0 "java -jar /opt/Code_..." 53 minutes ago Up 4 minutes 0.0.0.0:8082->8082/tcp, :::8082->8082/tcp
notificationservice
b9df56cb1731 hotelbookingapp/paymentservice:1.0.0 "java -jar /opt/Code_..." 53 minutes ago Up 4 minutes 0.0.0.0:8083->8083/tcp, :::8083->8083/tcp
paymentservice
b8c4534f3bf5 hotelbookingapp/bookingservice:1.0.0 "java -jar /opt/Code_..." 53 minutes ago Up 4 minutes 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp
bookingservice
67ff0fa0ea03 hotelbookingapp/serviceregistry:1.0.0 "java -jar /opt/Code_..." 53 minutes ago Up 4 minutes 0.0.0.0:8761->8761/tcp, :::8761->8761/tcp
serviceregistry

ubuntu@ip-10-0-0-209:~$ sudo docker logs 31055ddd9dc2
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
message
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
message
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
message
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
message
Booking confirmed for user with aadhaar number: Sreya-Aadhar Number | Here are the booking details: BookingInfoEntity(fromDate=2021-05-16 00:00:00.0, toDate=2021-07-25 00:00:00.0, aadharNumber='Sreya-Aadhar Number', roomNumbers='61,12,74', roomPrice=210000, truncationId='3', bookedOn=2021-10-28 11:09:18.0)
ubuntu@ip-10-0-0-209:~$
```

Eureka Server logs:

```
2021-10-28 11:09:03.339 INFO 1 --- [           main] DiscoveryClientOptionalArgsConfiguration : Eureka HTTP Client uses Jersey
2021-10-28 11:09:03.692 WARN 1 --- [           main] iguration$LoadBalancerCaffeineWarnLogger : Spring Cloud LoadBalancer is currently working with the default cache. You can switch to using Caffeine cache, by adding it and org.springframework.cache.caffeine.CaffeineCacheManager to the classpath.
2021-10-28 11:09:03.791 INFO 1 --- [           main] o.s.c.n.eureka.InstanceInfoFactory : Setting initial instance status as: STARTING
2021-10-28 11:09:03.935 INFO 1 --- [           main] com.netflix.discovery.DiscoveryClient : Initializing Eureka in region us-east-1
2021-10-28 11:09:03.937 INFO 1 --- [           main] com.netflix.discovery.DiscoveryClient : Client configured to neither register nor query for data.
2021-10-28 11:09:03.976 INFO 1 --- [           main] com.netflix.discovery.DiscoveryClient : Discovery Client initialized at timestamp 1635419343975 with initial instances count: 0
2021-10-28 11:09:04.147 INFO 1 --- [           main] c.n.eureka.DefaultEurekaServerContext : Initializing ...
2021-10-28 11:09:04.152 WARN 1 --- [           main] c.n.eureka.cluster.PeerEurekaNodes : The replica size seems to be empty. Check the route 53 DNS Registry
2021-10-28 11:09:04.375 INFO 1 --- [           main] c.n.e.registry.AbstractInstanceRegistry : Finished initializing remote region registries. All known remote regions: []
2021-10-28 11:09:04.387 INFO 1 --- [           main] c.n.eureka.DefaultEurekaServerContext : Initialized
2021-10-28 11:09:04.438 INFO 1 --- [           main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 2 endpoint(s) beneath base path '/actuator'
2021-10-28 11:09:04.618 INFO 1 --- [           main] o.s.c.n.e.eureka.EurekaServiceRegistry : Registering application UNKNOWN with eureka with status UP
2021-10-28 11:09:04.649 INFO 1 --- [Thread-9] o.s.c.n.e.server.EurekaServerBootstrap : Setting the eureka configuration..
2021-10-28 11:09:04.738 INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8761 (http) with context path ''
2021-10-28 11:09:04.740 INFO 1 --- [Thread-9] o.s.c.n.e.server.EurekaServerBootstrap : isAWS returned false
2021-10-28 11:09:04.749 INFO 1 --- [           main] o.s.c.n.e.EurekaAutoServiceRegistration : Updating port to 8761
2021-10-28 11:09:04.746 INFO 1 --- [Thread-9] o.s.c.n.e.server.EurekaServerBootstrap : Initialized server context
2021-10-28 11:09:04.756 INFO 1 --- [Thread-9] c.n.e.r.PeerAwarenessRegistryImpl : Got 1 instances from neighboring DS node
2021-10-28 11:09:04.756 INFO 1 --- [Thread-9] c.n.e.r.PeerAwarenessRegistryImpl : Renew threshold is: 1
2021-10-28 11:09:04.761 INFO 1 --- [Thread-9] c.n.e.r.PeerAwarenessRegistryImpl : Changing status to UP
2021-10-28 11:09:04.823 INFO 1 --- [           main] o.s.e.EurekaServerApplication : Started EurekaServerApplication in 23.825 seconds (JVM running for 26.742)
2021-10-28 11:09:04.884 INFO 1 --- [Thread-9] o.s.EurekaServerInitializerConfiguration : Started Eureka Server
2021-10-28 11:09:05.820 INFO 1 --- [nio-8761-exec-2] o.s.c.o.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2021-10-28 11:09:05.828 INFO 1 --- [nio-8761-exec-2] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-10-28 11:09:05.831 INFO 1 --- [nio-8761-exec-2] o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms
2021-10-28 11:09:06.739 INFO 1 --- [nio-8761-exec-3] c.n.e.registry.AbstractInstanceRegistry : Registered instance PAYMENT-SERVICE/b9df56cb1731:payment-service:8083 with status UP (replication=false)
2021-10-28 11:09:06.979 INFO 1 --- [nio-8761-exec-4] c.n.e.registry.AbstractInstanceRegistry : Registered instance BOOKING-SERVICE/b8c4534f3bf5:booking-service with status UP (replication=false)
2021-10-28 11:10:04.807 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
2021-10-28 11:11:04.808 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
2021-10-28 11:12:04.808 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
2021-10-28 11:13:04.809 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
2021-10-28 11:14:04.808 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
2021-10-28 11:15:04.808 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
2021-10-28 11:16:04.808 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
2021-10-28 11:17:04.808 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
2021-10-28 11:18:04.808 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
ubuntu@ip-10-0-0-209:~$
```

