Project Explanation

So, we have created 5 Micro-services

1. Eureka Server: `eureka-server`
2. Booking Service: `booking-service`
3. Payment Service: `payment-service`
4. Notification Service: `notification-service`
5. API Gateway: `api-gateway`

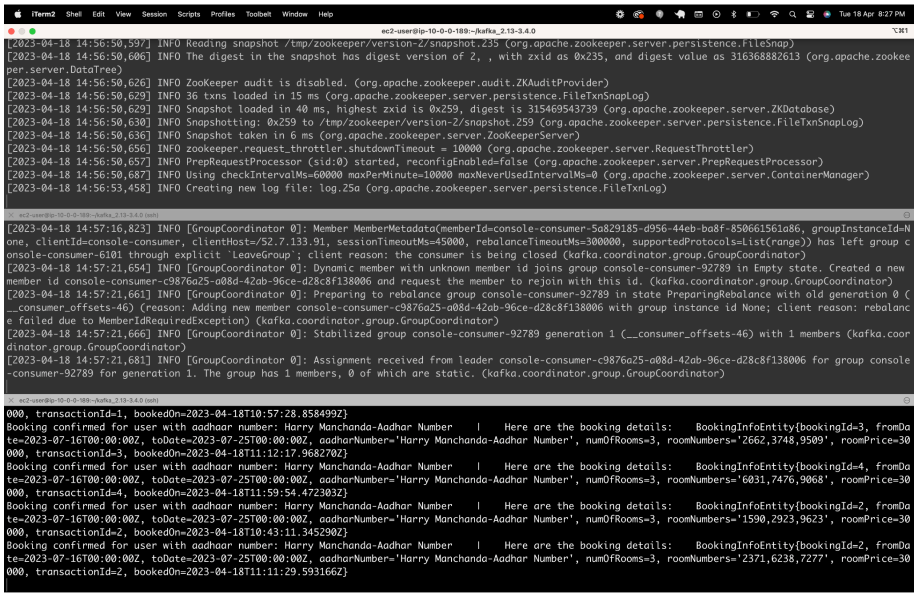
README Instructions

for running the entire project…

1. Create an Amazon RDS server and start the instance. Please don't forget to configure the VPC and security group properly. Please refer to https://cdn.upgrad.com/uploads/production/a1cc948c-b8c0-4421-bba1-087885df4b76/Database+Set+Up.pdf.
2. Create an Elastic IP and EC2 Instance (refer to https://cdn.upgrad.com/uploads/production/dd278192-d4c3-4035-9ff7-ed82a75e625f/EC2+MyIP+Setup.docx.pdf). Then to run Kafka on EC2, please refer to https://cdn.upgrad.com/uploads/production/96c2c559-89b9-43c1-af4b-7bb746feec5b/Kafka+Quickstart.pdf

3. Inside the kafka directory, run `bin/kafka-topics.sh --bootstrap-server localhost:9092 --create --topic message --partitions 1 --replication-factor 1` to create a `message` topic.

4. Optionally, Run `bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic message --from-beginning` to view the `message` topic flowing in the terminal. You should now have a terminal workspace like this below image attached. First tab being for running the zookeeper, second for kafka server, and third for the messages flowing in for the subscribed `message` topic.


iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
Tue 18 Apr 8:27 PM
ec2-user@ip-10-0-0-189:~/kafka_2.13-3.4.0
ec2-user@ip-10-0-0-189:~/kafka_2.13-3.4.0 (ssh)
ec2-user@ip-10-0-0-189:~/kafka_2.13-3.4.0 (ssh)

```
[2023-04-18 14:56:50,597] INFO Reading snapshot /tmp/zookeeper/version-2/snapshot.235 (org.apache.zookeeper.server.persistence.FileSnap)
[2023-04-18 14:56:50,606] INFO The digest in the snapshot has digest version of 2, , with zxid as 0x235, and digest value as 316368882613 (org.apache.zookee
per.server.DataTree)
[2023-04-18 14:56:50,626] INFO ZooKeeper audit is disabled. (org.apache.zookeeper.audit.ZKAuditProvider)
[2023-04-18 14:56:50,629] INFO 36 txns loaded in 15 ms (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2023-04-18 14:56:50,629] INFO Snapshot loaded in 40 ms, highest zxid is 0x259, digest is 315469543739 (org.apache.zookeeper.server.ZKDatabase)
[2023-04-18 14:56:50,630] INFO Snapshotting: 0x259 to /tmp/zookeeper/version-2/snapshot.259 (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2023-04-18 14:56:50,636] INFO Snapshot taken in 6 ms (org.apache.zookeeper.server.ZooKeeperServer)
[2023-04-18 14:56:50,656] INFO zookeeper.request_throttler.shutdownTimeout = 10000 (org.apache.zookeeper.server.RequestThrottler)
[2023-04-18 14:56:50,657] INFO PrepRequestProcessor (sid:0) started, reconfigEnabled=false (org.apache.zookeeper.server.PrepRequestProcessor)
[2023-04-18 14:56:50,687] INFO Using checkIntervalMs=60000 maxPerMinute=10000 maxNeverUsedIntervalMs=0 (org.apache.zookeeper.server.ContainerManager)
[2023-04-18 14:56:53,458] INFO Creating new log file: log.25a (org.apache.zookeeper.server.persistence.FileTxnLog)
```

```
[2023-04-18 14:57:16,823] INFO [GroupCoordinator 0]: Member MemberMetadata(memberId=console-consumer-5a829185-d956-44eb-ba8f-850661561a86, groupInstanceId=N
one, clientId=console-consumer, clientHost=/52.7.133.91, sessionTimeoutMs=45000, rebalanceTimeoutMs=300000, supportedProtocols=List(range)) has left group c
onsole-consumer-6101 through explicit `LeaveGroup`; client reason: the consumer is being closed (kafka.coordinator.group.GroupCoordinator)
[2023-04-18 14:57:21,654] INFO [GroupCoordinator 0]: Dynamic member with unknown member id joins group console-consumer-92789 in Empty state. Created a new
member id console-consumer-c9876a25-a08d-42ab-96ce-d28c8f138006 and request the member to rejoin with this id. (kafka.coordinator.group.GroupCoordinator)
[2023-04-18 14:57:21,661] INFO [GroupCoordinator 0]: Preparing to rebalance group console-consumer-92789 in state PreparingRebalance with old generation 0 (
__consumer_offsets-46) (reason: Adding new member console-consumer-c9876a25-a08d-42ab-96ce-d28c8f138006 with group instance id None; client reason: rebalanc
e failed due to MemberIdRequiredException) (kafka.coordinator.group.GroupCoordinator)
[2023-04-18 14:57:21,666] INFO [GroupCoordinator 0]: Stabilized group console-consumer-92789 generation 1 (__consumer_offsets-46) with 1 members (kafka.coor
dinator.group.GroupCoordinator)
[2023-04-18 14:57:21,681] INFO [GroupCoordinator 0]: Assignment received from leader console-consumer-c9876a25-a08d-42ab-96ce-d28c8f138006 for group console
-consumer-92789 for generation 1. The group has 1 members, 0 of which are static. (kafka.coordinator.group.GroupCoordinator)
```

```
000, transactionId=1, bookedOn=2023-04-18T10:57:28.858499Z}
Booking confirmed for user with aadhaar number: Harry Manchanda-Aadhar Number    |    Here are the booking details:    BookingInfoEntity{bookingId=3, fromDa
te=2023-07-16T00:00:00Z, toDate=2023-07-25T00:00:00Z, aadharNumber='Harry Manchanda-Aadhar Number', numOfRooms=3, roomNumbers='2662,3748,9509', roomPrice=30
000, transactionId=3, bookedOn=2023-04-18T11:12:17.968270Z}
Booking confirmed for user with aadhaar number: Harry Manchanda-Aadhar Number    |    Here are the booking details:    BookingInfoEntity{bookingId=4, fromDa
te=2023-07-16T00:00:00Z, toDate=2023-07-25T00:00:00Z, aadharNumber='Harry Manchanda-Aadhar Number', numOfRooms=3, roomNumbers='6031,7476,9068', roomPrice=30
000, transactionId=4, bookedOn=2023-04-18T11:59:54.472303Z}
Booking confirmed for user with aadhaar number: Harry Manchanda-Aadhar Number    |    Here are the booking details:    BookingInfoEntity{bookingId=2, fromDa
te=2023-07-16T00:00:00Z, toDate=2023-07-25T00:00:00Z, aadharNumber='Harry Manchanda-Aadhar Number', numOfRooms=3, roomNumbers='1590,2923,9623', roomPrice=30
000, transactionId=2, bookedOn=2023-04-18T10:43:11.345290Z}
Booking confirmed for user with aadhaar number: Harry Manchanda-Aadhar Number    |    Here are the booking details:    BookingInfoEntity{bookingId=2, fromDa
te=2023-07-16T00:00:00Z, toDate=2023-07-25T00:00:00Z, aadharNumber='Harry Manchanda-Aadhar Number', numOfRooms=3, roomNumbers='2371,6238,7277', roomPrice=30
000, transactionId=2, bookedOn=2023-04-18T11:11:29.593166Z}
```

5. Update `booking-service` => `application.yml` relative to below code

```yaml
spring:
 datasource:
   url: jdbc:mysql://<rds-instance>/booking-service-db?createDatabaseIfNotExist=true
   username: <rds-username>
   password: <rds-password>

notificationService:
 kafkaServerValue: ec2-user@<ec2-instance>9092
```

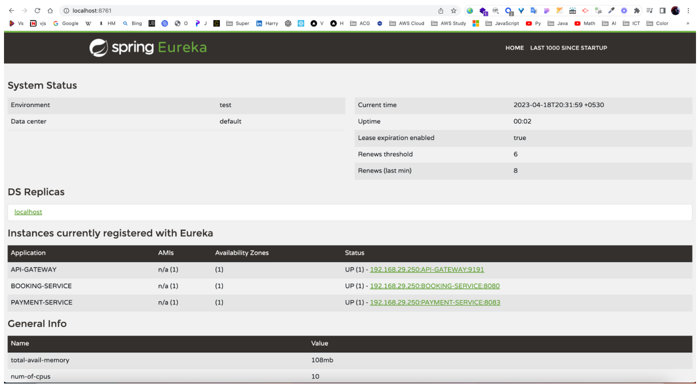
6. Update `payment-service` => `application.yml` relative to below code

```yaml
spring:
 datasource:
   url:
jdbc:mysql://<rds-instance>/payment-service-db?createDatabaseIfNotExist=true
   username: <rds-username>
   password: <rds-password>
```

7. Update `notification-service` => `consumers/KafkaMessageConsumer.java` relative to below code

```
properties.put("bootstrap.servers",
"ec2-user@<ec2-instance>:9092");
```

8. Start the `eureka-server` => `EurekaServerApplication.java`
9. Start the `booking-service` => `BookingServiceApplication.java`
10. Start the `payment-service` => `PaymentServiceApplication.java`
11. Start the `notification-service` => `consumers/KafkaMessageConsumer.java`
12. Start the `api-gateway` => `ApiGatewayApplication.java`
13. Now visit http://localhost:8761/ you should see the image below.



System Status

Environment	test
Data center	default

Current time	2023-04-18T20:31:59 +0530
Uptime	00:02
Lease expiration enabled	true
Renews threshold	6
Renews (last min)	8

DS Replicas

localhost

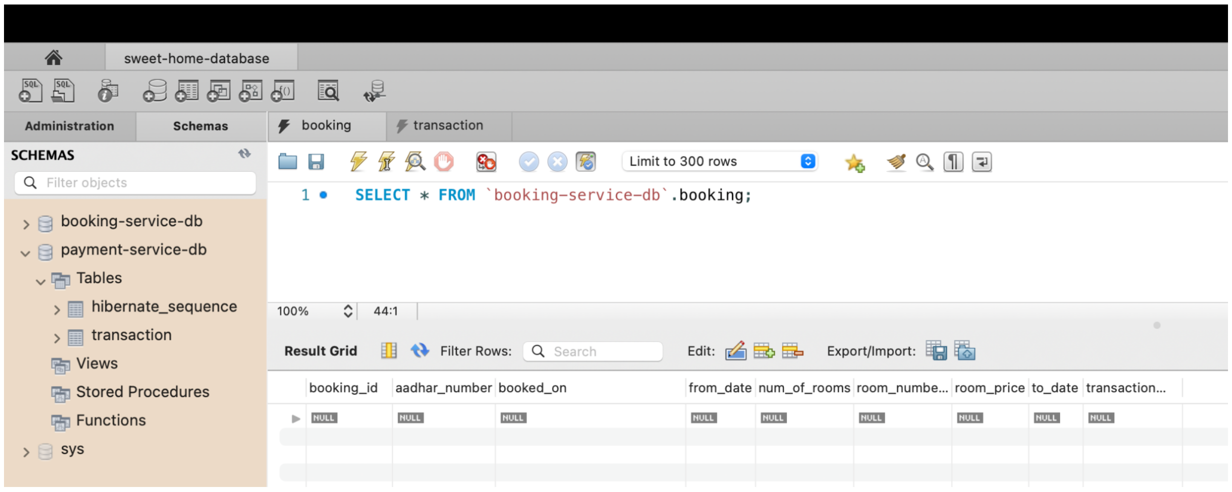
Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - 192.168.29.250:API-GATEWAY:9191
BOOKING-SERVICE	n/a (1)	(1)	UP (1) - 192.168.29.250:BOOKING-SERVICE:8080
PAYMENT-SERVICE	n/a (1)	(1)	UP (1) - 192.168.29.250:PAYMENT-SERVICE:8083

General Info

Name	Value
total-avail-memory	108mb
num-of-cpus	10

14. Optionally, open the Sweet Home Database in MySQL Workbench to view database change while we run API's. When you start the Sweet Home Database before hitting our API's you should see something like this as image attached below



```sql
SELECT * FROM `booking-service-db`.booking;
```

booking_id	aadhar_number	booked_on	from_date	num_of_rooms	room_numbe...	room_price	to_date	transaction...
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

15. Now let’s Run our User Facing API Endpoint for **Creating a Booking**

```
 Endpoint: {API_URL}/booking
 API URL (Booking Service): localhost:8080/booking
 API GATEWAY’s API URL: http://localhost:9191/booking
 HTTP METHOD: POST
 RequestBody: fromDate, toDate, aadharNumber, numOfRooms
 ResponseBody: id, fromDate, toDate, aadharNumber, roomNumbers, roomPrice,
transactionId, bookedOn
```

16. Now test our Internal Microservice communication based internal API Endpoint for **Creating a Transaction**

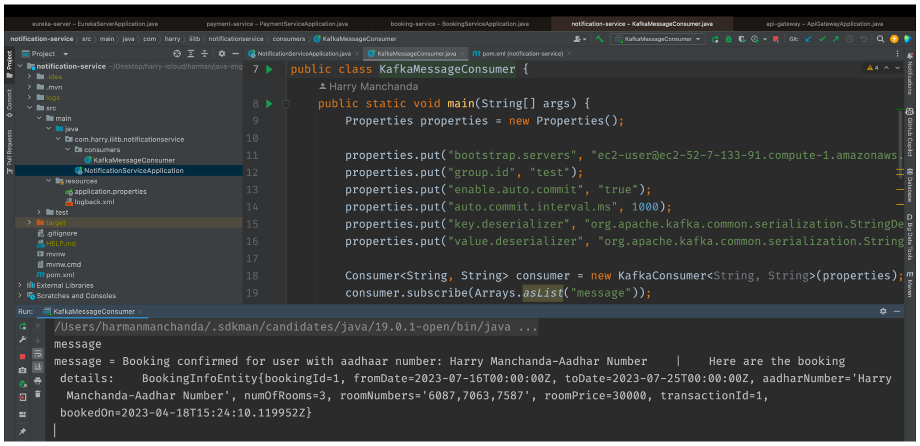
```
 Endpoint: {API_URL}/transaction
 API URL (Payment Service): localhost:8083/transaction
 API GATEWAY URL: http://localhost:9191/transaction
 HTTP METHOD: POST
 RequestBody: paymentMode, bookingId, upiId, cardNumber
 ResponseBody: id, transactionId, paymentMode, bookingId, upiId, cardNumber
```

Note that the Response body for this specific endpoint is for our Internal Microservice communication to send respective data from Payment Service to Booking Service. In an Real World Application, we will not expose this API Endpoint to the Frontend of our Application.

17. Now let’s Run our User Facing API Endpoint for **Creating a Booking Transaction**

```
 Endpoint: {API_URL}/booking/{bookingId}/transaction
 API URL (Booking Service): localhost:8080/booking/{bookingId}/transaction
 API GATEWAY URL: http://localhost:9191/booking/{bookingId}/transaction
 HTTP METHOD: POST
 RequestBody: paymentMode, bookingId, upiId, cardNumber
 ResponseBody: id, fromDate, toDate, aadharNumber, roomNumbers, roomPrice,
transactionId, bookedOn
```

18. Now let's confirm the notification message consumption. Please visit the running console in your Intellij for `notification-service` => `consumers/KafkaMessageConsumer.java`. You should see an image like attached below



```java
public class KafkaMessageConsumer {
    public static void main(String[] args) {
        Properties properties = new Properties();

        properties.put("bootstrap.servers", "ec2-user@ec2-52-7-133-91.compute-1.amazonaws.
        properties.put("group.id", "test");
        properties.put("enable.auto.commit", "true");
        properties.put("auto.commit.interval.ms", 1000);
        properties.put("key.deserializer", "org.apache.kafka.common.serialization.StringDe
        properties.put("value.deserializer", "org.apache.kafka.common.serialization.String

        Consumer<String, String> consumer = new KafkaConsumer<String, String>(properties);
        consumer.subscribe(Arrays.asList("message"));
```

```
/Users/harmanmanchanda/.sdkman/candidates/java/19.0.1-open/bin/java ...
message
message = Booking confirmed for user with aadhaar number: Harry Manchanda-Aadhar Number    |    Here are the booking
 details:    BookingInfoEntity{bookingId=1, fromDate=2023-07-16T00:00:00Z, toDate=2023-07-25T00:00:00Z, aadharNumber='Harry
  Manchanda-Aadhar Number', numOfRooms=3, roomNumbers='6087,7063,7587', roomPrice=30000, transactionId=1,
 bookedOn=2023-04-18T15:24:10.119952Z}
```

19. Now let's Run our User Facing API Endpoint for **Finding a Transaction by Id**

```
 Endpoint: {API_URL}/transaction/{transactionId}
 API URL (Payment Service): localhost:8083/transaction/{transactionId}
 API GATEWAY URL: http://localhost:9191/transaction/{transactionId}
 HTTP METHOD: GET
 RequestBody: N/A
 ResponseBody: id, paymentMode, bookingId, upiId, cardNumber
```

That's it, if you got here that means the project is running well properly and functionally.