**Book Catalogue - SpringBoot Microservices**

**Inside**

1. What – The Introduction
2. Why – The Use Case
3. How – The Design
4. Using – The Technologies
5. Build – The Packaging
6. View - The Sanpshot
7. Contributors – The Innovators

**What**

Book Catalogue is a spring boot application for a client to perform CRUD operations designed in a Microservices architectural style. This application provides an endpoint which enables client to

* Add a new Book(s)
* Retrieve Book(s) with title,author name, 13-digit ISBN
* Update a Book
* Delete a Book.

**Why**

Book Catalogue application is built as an internal exercise/request.

**How**

* + Book Catalogue application helps the client to perform CRUD actions.
  + Primarily Spring REST is used to perform these CRUD actions which are built on microservice architectural design.
  + Used Apache Kafka to publish messages into Kafka-Topic(queue) whenever the client performs a new action on Book Catalogue.

**Using**

Technologies or Packages helped this possible

1. Java 11
2. JDK1.8
3. Spring Boot(STS)
4. Apache Kafka
5. Git & GitHub
6. MySql
7. Postman

**Build**

**Rest APIs endpoints used in postman:**

*POST:* [*http://localhost:9191/bookcatalog/addBook*](http://localhost:9191/bookcatalog/addBook)

*POST:* [*http://localhost:9191/bookcatalog/addBooks*](http://localhost:9191/bookcatalog/addBooks)

*GET:* [*http://localhost:9191/bookcatalog/bookById/1*](http://localhost:9191/bookcatalog/bookById/1)

*GET:* [*http://localhost:9191/bookcatalog/bookByIsbn/978-93-8067-432-2*](http://localhost:9191/bookcatalog/bookByIsbn/978-93-8067-432-2)

*GET:* [*http://localhost:9191/bookcatalog/bookByAuthName/Lalit%20Kumar*](http://localhost:9191/bookcatalog/bookByAuthName/Lalit%20Kumar)

*PUT:* [*http://localhost:9191/bookcatalog/update*](http://localhost:9191/bookcatalog/update)

*DELETE:* [*http://localhost:9191/bookcatalog/deleteById/1*](http://localhost:9191/bookcatalog/deleteById/1)

**Spring Eureka used for service registry:**

<http://localhost:8761/>

**Cloud Api-Gateway:**

<http://localhost:9191/>

**GitHub:**

<https://github.com/naveendvnm/BookCatalogMicroservice.git>

**Kafka:**

To run ZooKepper:

C:\kafka>.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties

To start the Apache Kafka-

C:\kafka>.\bin\windows\kafka-server-start.bat .\config\server.properties

To create a topic with name **javatopic**, that has only one partition & one replica.

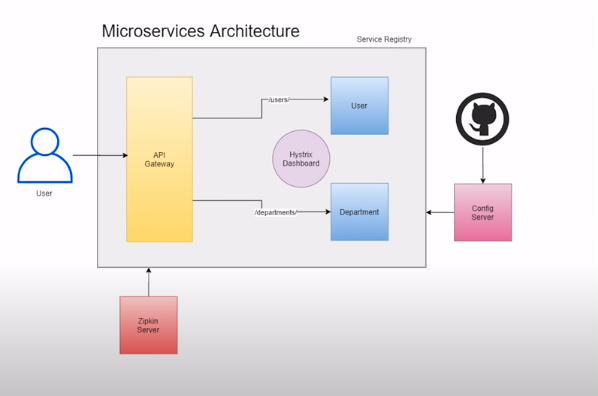
C:\kafka>.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic **javatopic**

Finally Open a new command prompt and start the consumer which listens to the topic **javatopic** we just created above. We will get the message we had sent using the producer

C:\kafka>.\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic **javatopic** --from-beginning

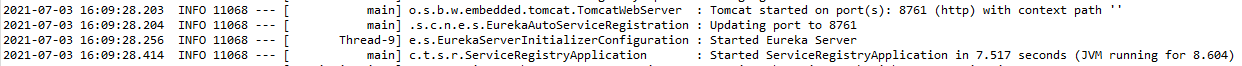
**Microservices:**

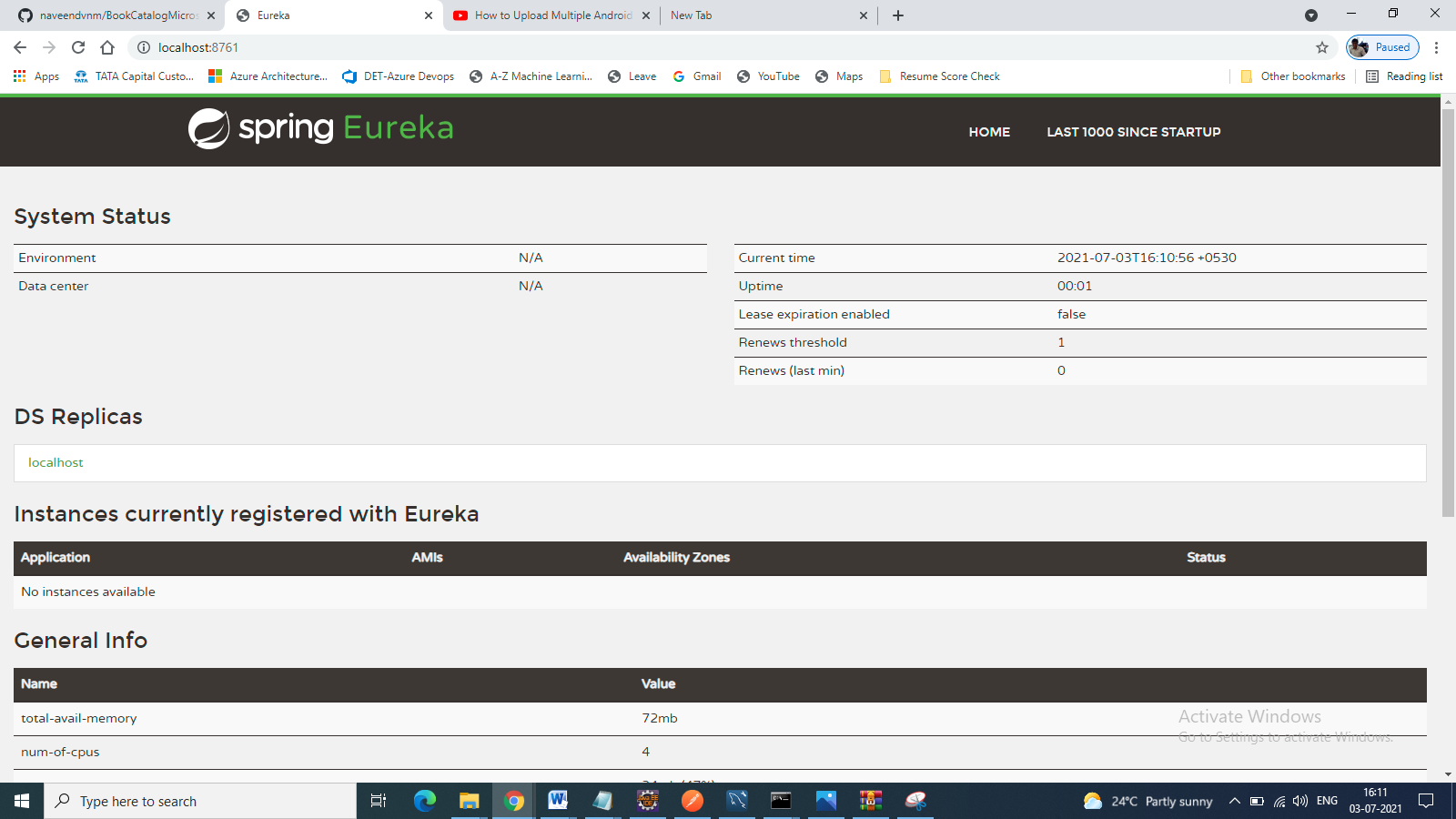
* Considering bookcatalog is a just single microservice and registered the same in service-registry.
* Built Api Cloud-Gateway to route user requests to the microservices(here it is bookcatalog).



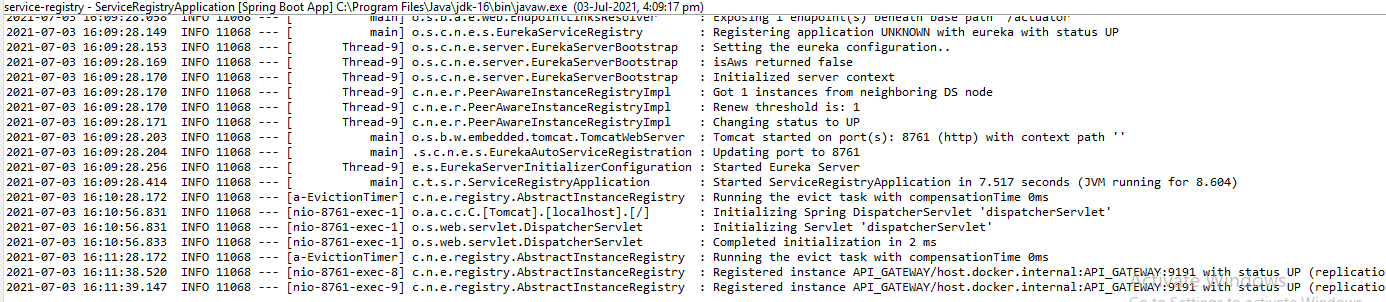
**View**

**Service\_Registry:**

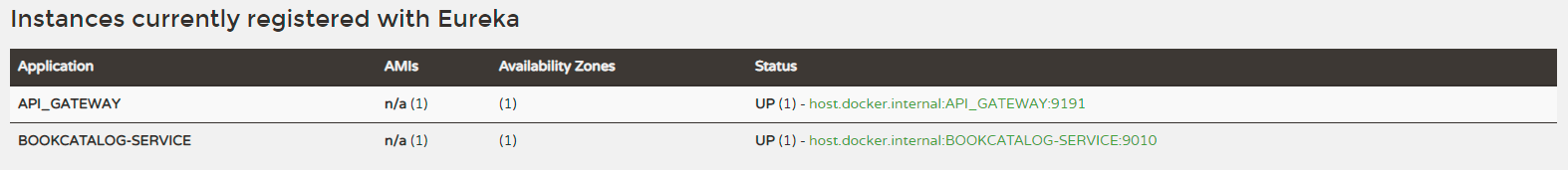




**Cloud\_Registry:**

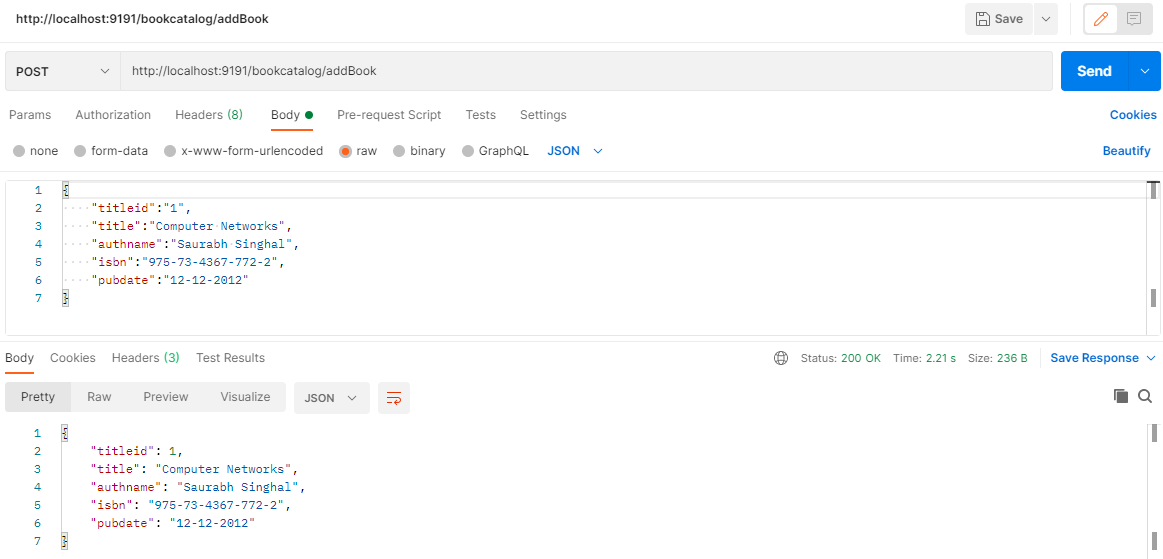


**BookCatalog\_Service:**

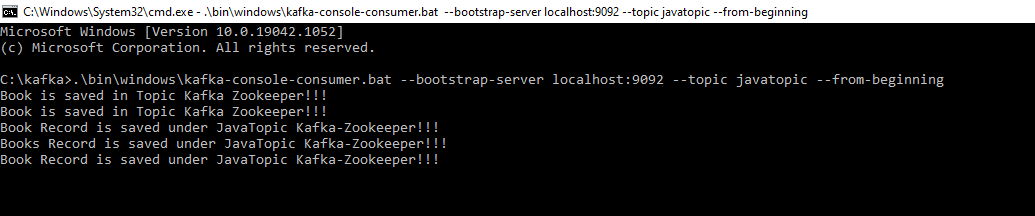


**After Run Zookeeper & Kafka**

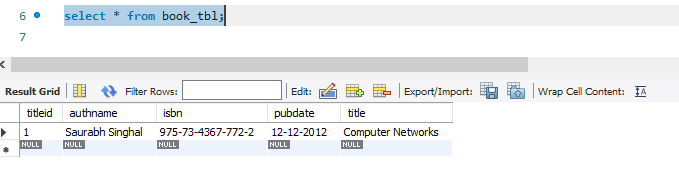
**Post the new Book from Postman:**



**We will get the message we had sent using the producer**



**Verify in MySql:**



**Contributors**

***Naveen Davanam***