# **APACHE KAFKA NOTES**

<http://blog.hampisoftware.com/index.php/2016/01/20/apache-kafka-differences-from-jms/>

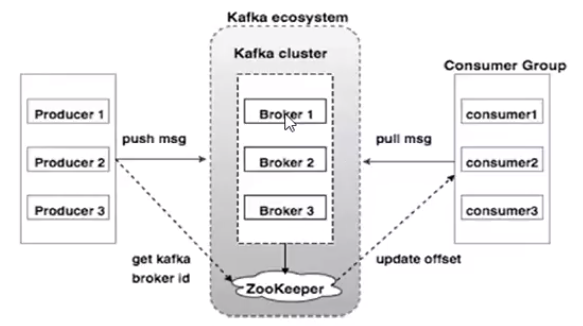
<https://kafka.apache.org/intro>

<https://kafka.apache.org/uses>

<https://gradle.org/maven-vs-gradle/>

<https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines>

<https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying>



Kafka has a Producer & a Consumer .

Kafka Cluster contains “n” number of brokers .

Broker is nothing but basically multiple instances of Kafka server .

Each broker is run on Kafka Server instance .

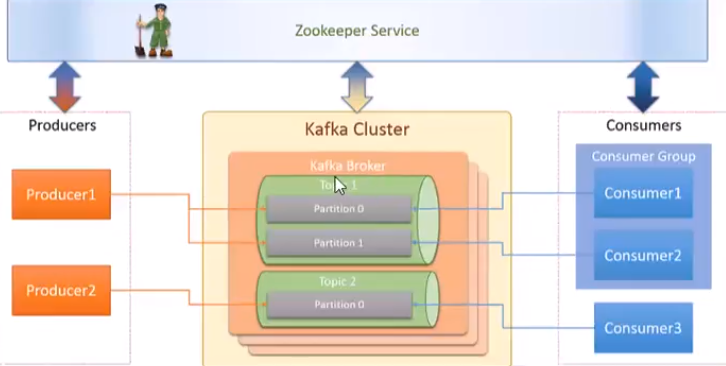
Inside Broker there are multiple “Topics” .

“Topic” is the key component in the Kafka Eco-system .

Producers push messages to the Topic .

Consumers pull messages from the Topic .

Topic is further categorized into Partitions .



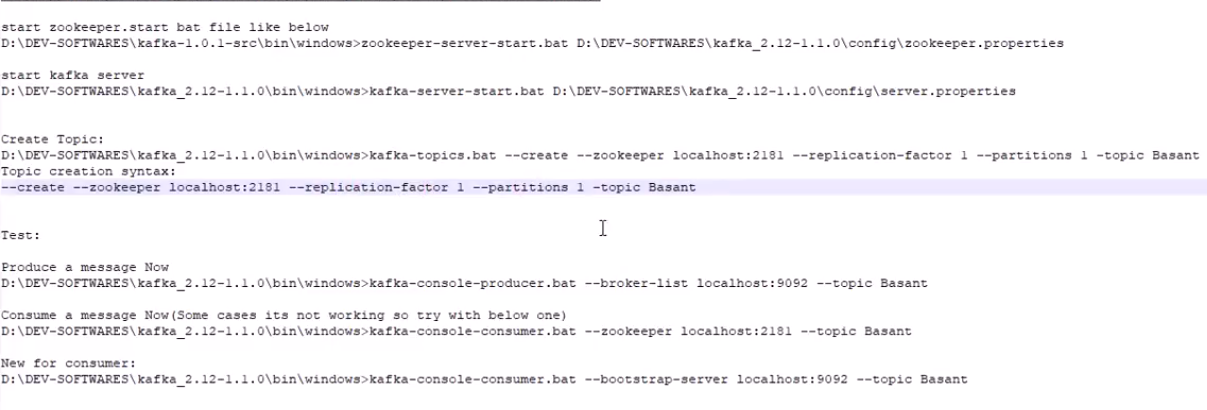
Zookeeper manages Kafka Server & scales it .

Zookeeper manages configurations for our Kafka Cluster .

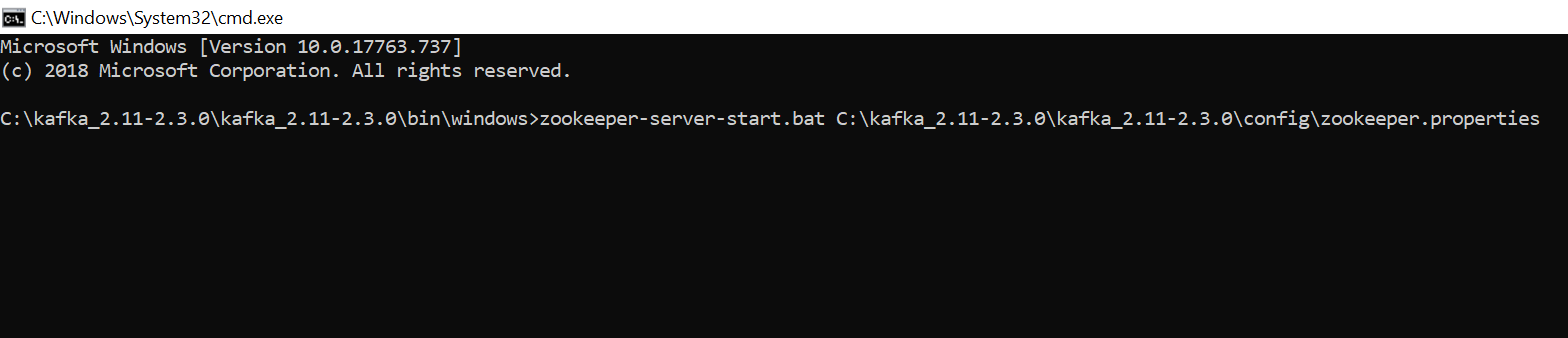
While creation of the topic , we need to define Replication Factor .

Replication Factor is nothing but shadow instance of your streaming data .

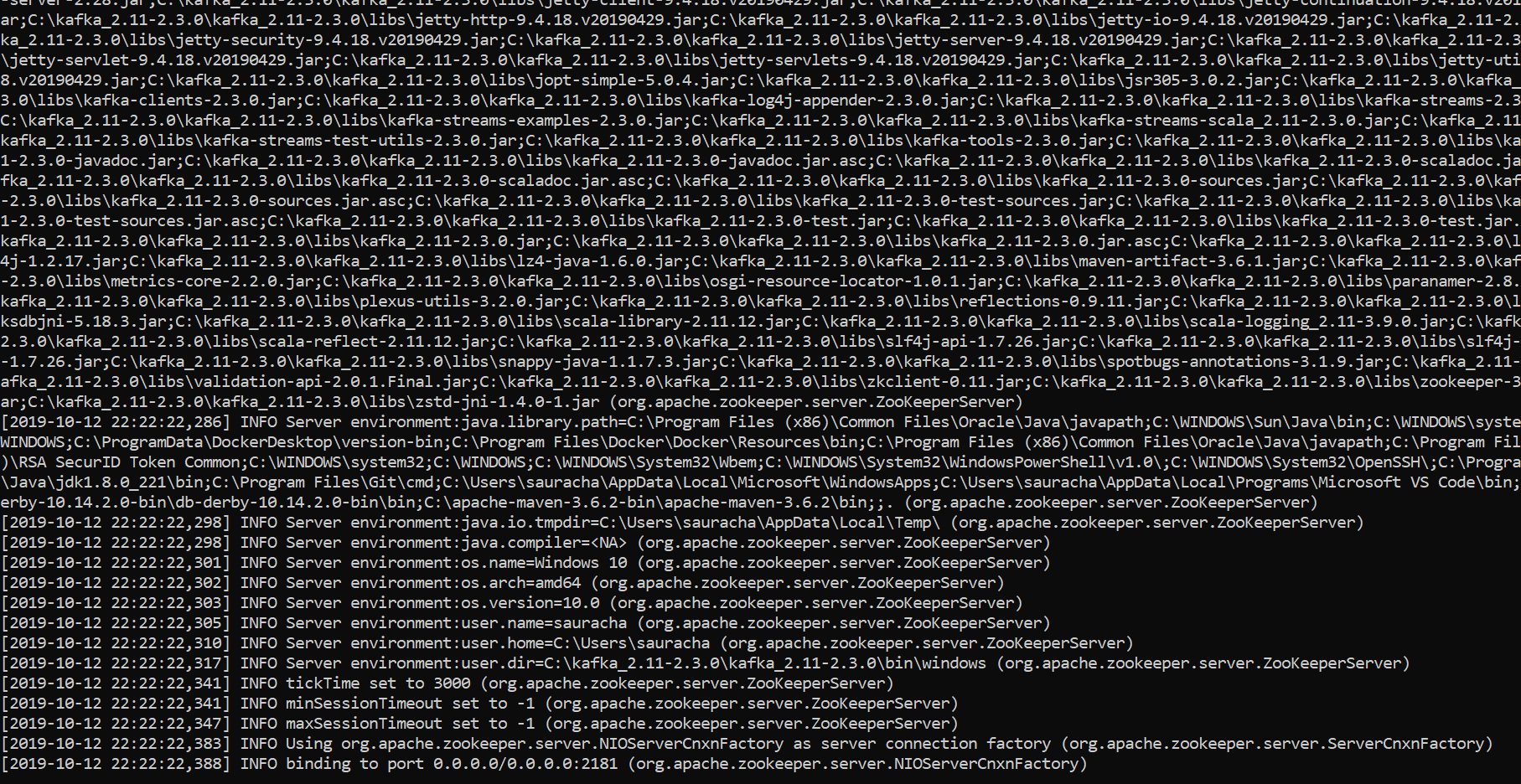
It helps in fault -tolerance if one or more nodes go down .



Syntax for Starting Zoo Keeper Server from CMD :



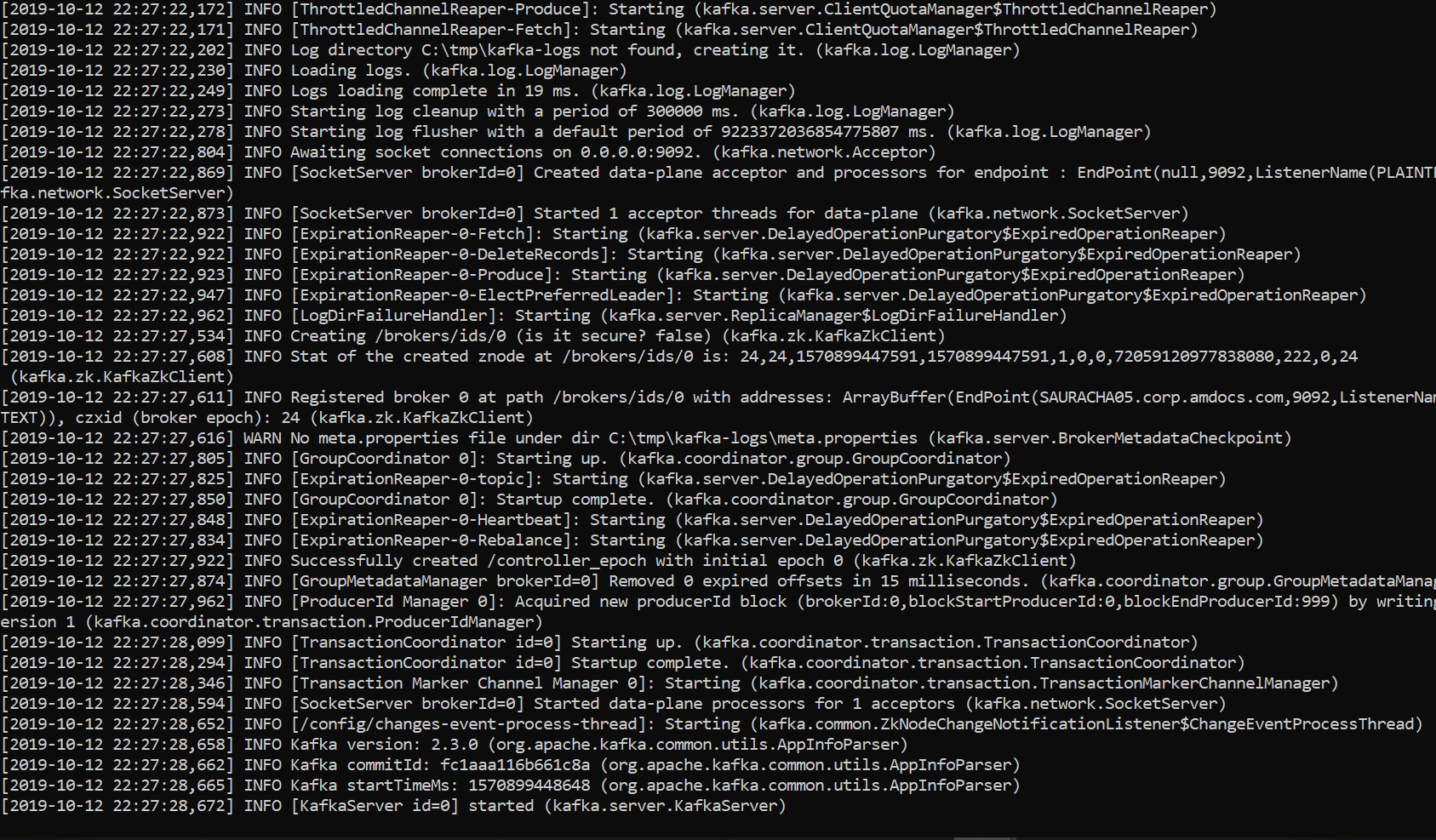
Successful Zookeeper Start-Up Logs



Start up Of Apache Kafka Server



Kafka Server Start – Up Logs



# start zookeeper.start bat file like below

zookeeper-server-start.bat D:\DEV-SOFTWARES\kafka\_2.12-1.1.0\config\zookeeper.properties

# start kafka server

kafka-server-start.bat D:\DEV-SOFTWARES\kafka\_2.12-1.1.0\config\server.properties

# Create Topic:

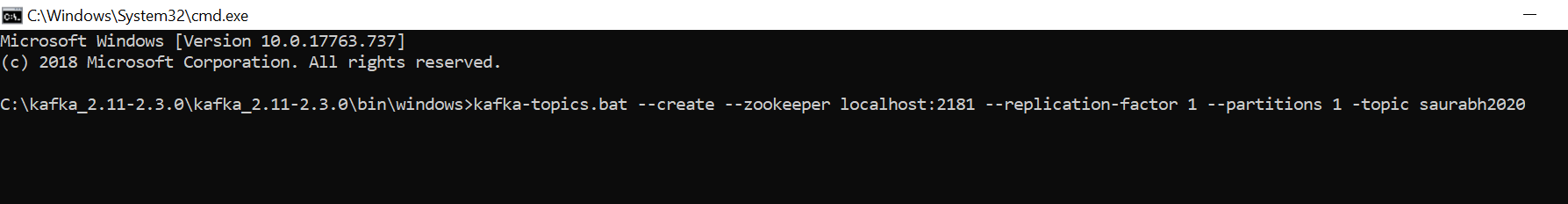
--create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 -topic javatechie

# Produce a message

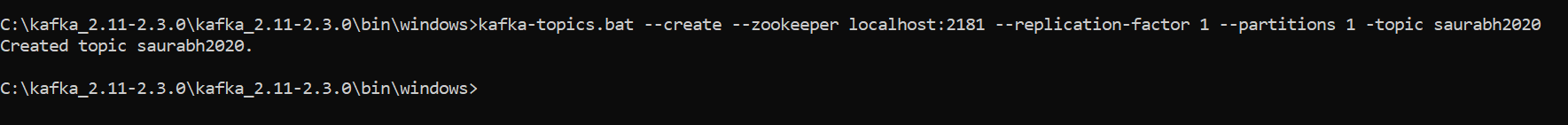
kafka-console-producer.bat --broker-list localhost:9092 --topic javatechie

# Consume a message

kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic topic\_name

Creating a Topic CMD

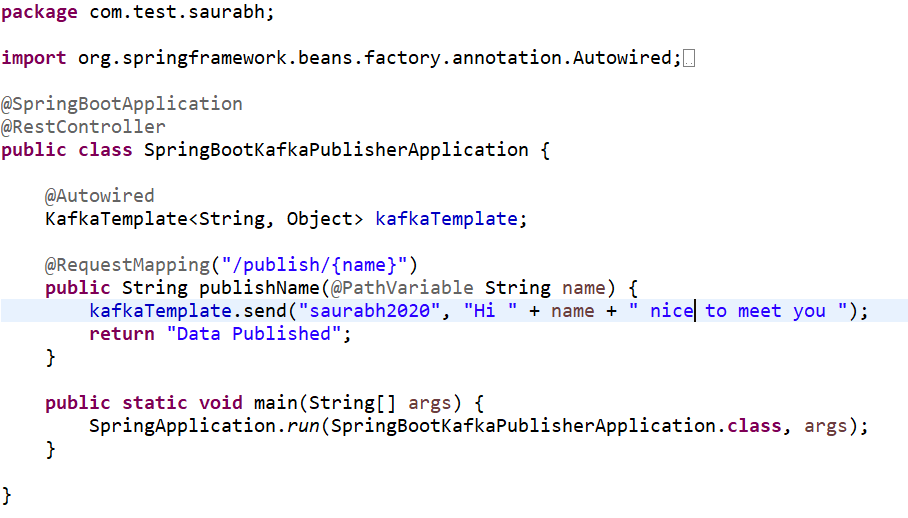
Topic Created !



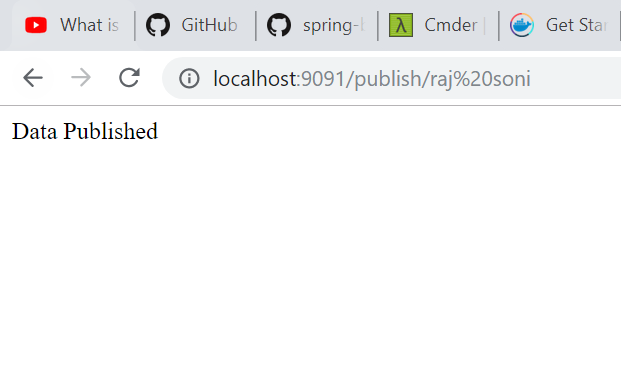
Check the data published on topic using below on CMD

kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic saurabh2020

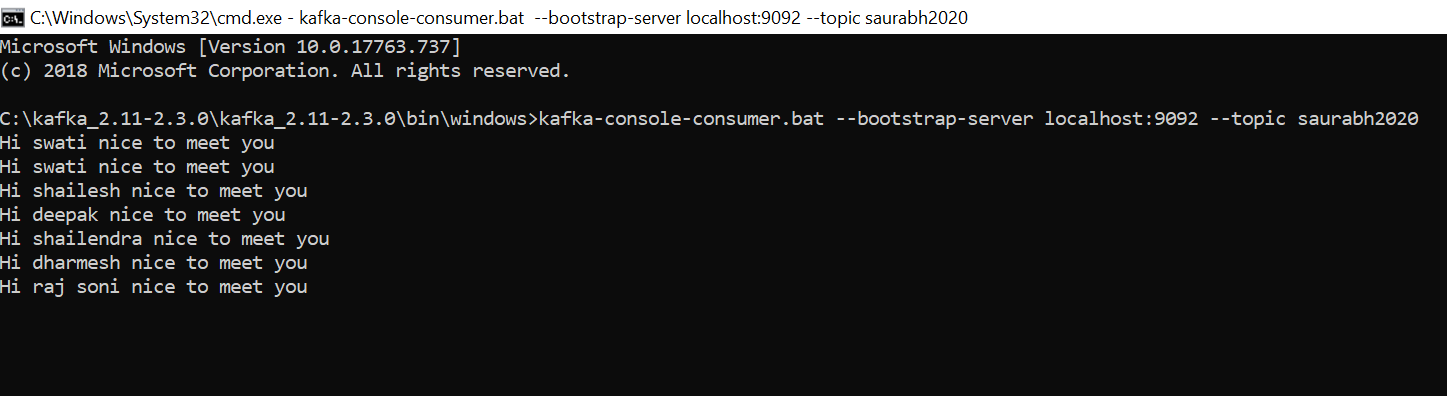
Barebones spring boot code to Publish a message using Apache Kafka



After Server is UP ; API on browser

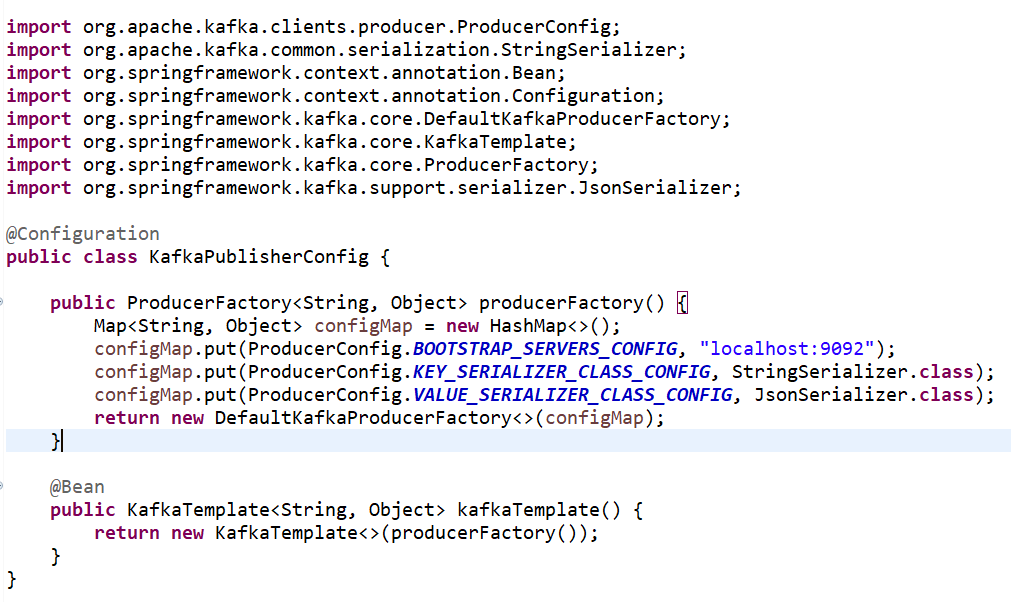


Messages being sent in real-time viewed on Apache Kafka Consumer Console as below :

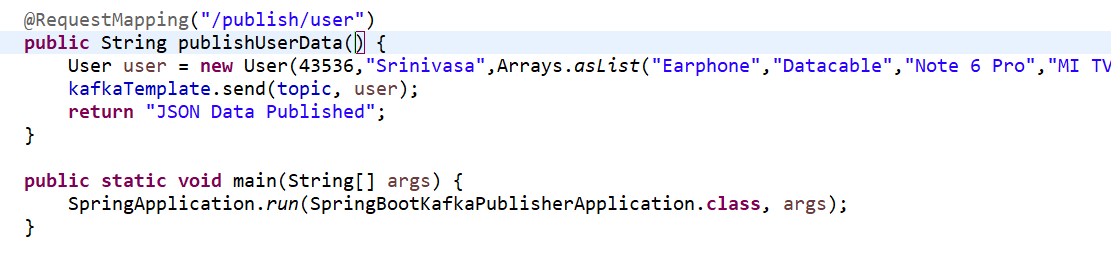


For Publishing Objects to a topic , we need to Override the KafkaTemplate default functionality ,

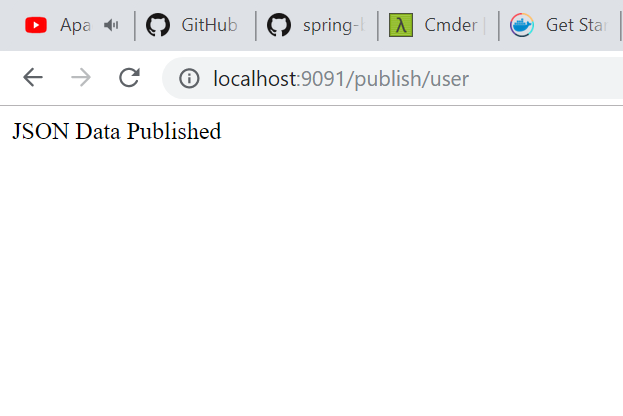
By provider configuration for Key Serialization & Value Serialization . Bare bones idea is below :



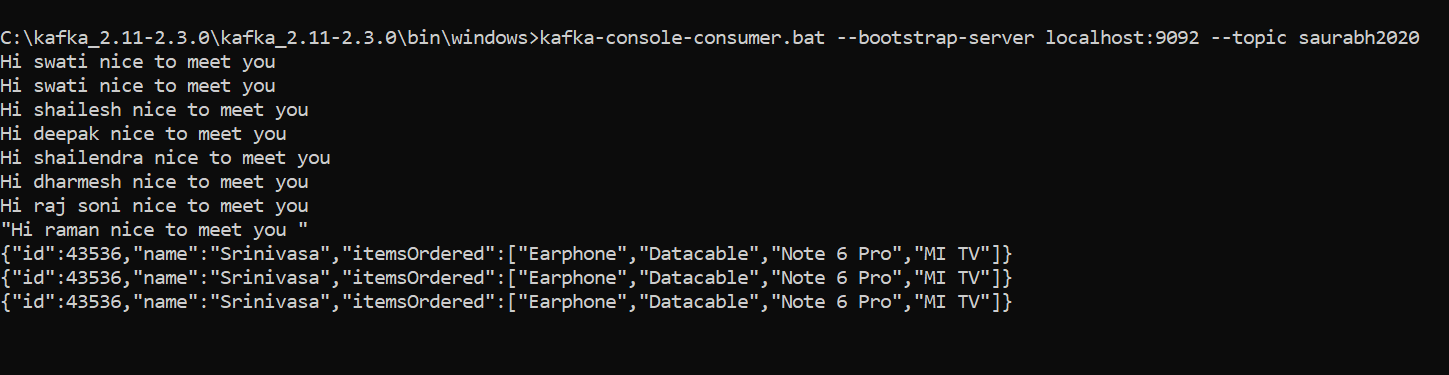
Added Controller to Return a Sample Object



Posting message using below endpoint :



JSON Data published on the consumer console along with the String data that we sent :



# NOW CONSUME THE PUBLISHED MESSAGE VIA SPRING BOOT CODE & NOT JUST THE CMD

While Consuming the data

We create a Config Class for Apache Consumer

We annotate it as @EnableKafka to listen for incoming messages

We create a consumerFactory that defines Deserializers for both types String , and Custom Object types .

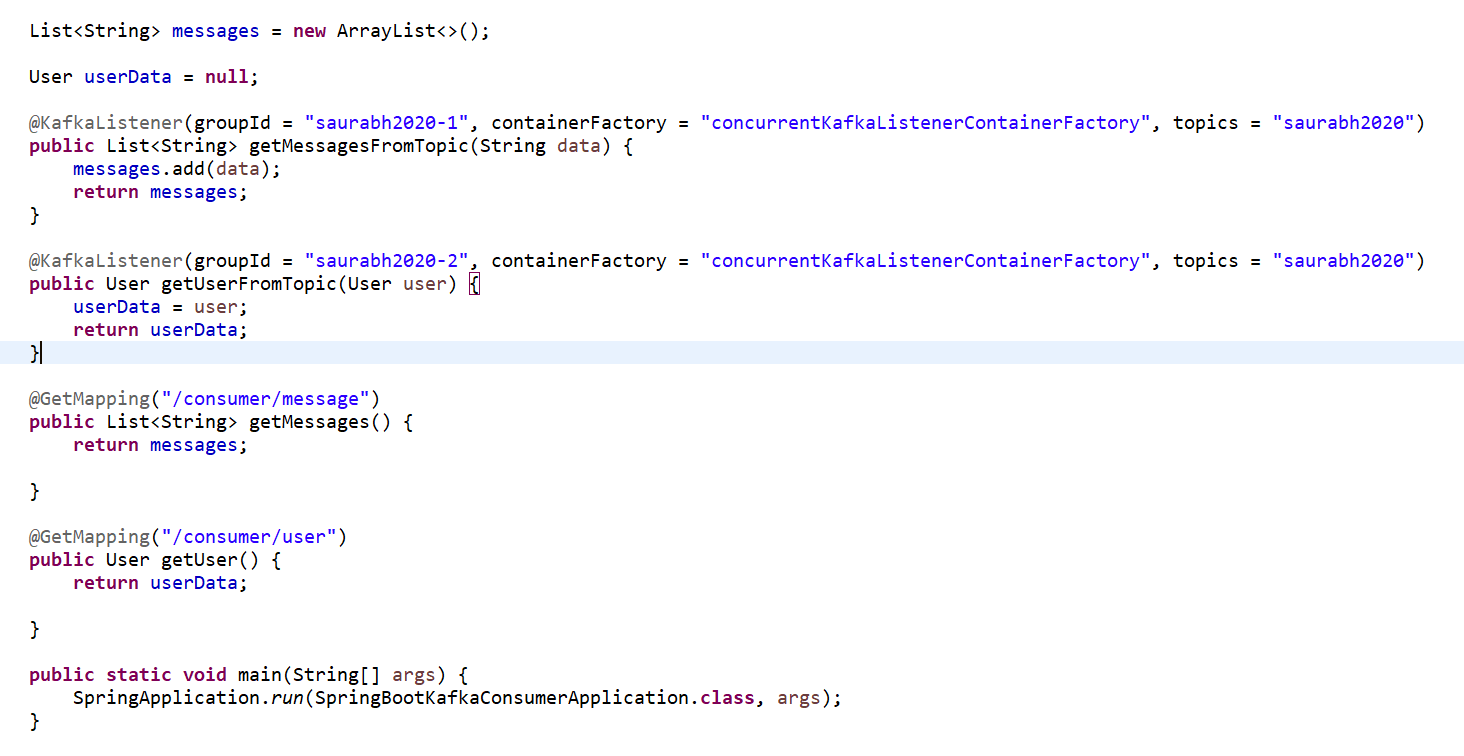
SNAPSHOT OF BARE BONES CONFIG



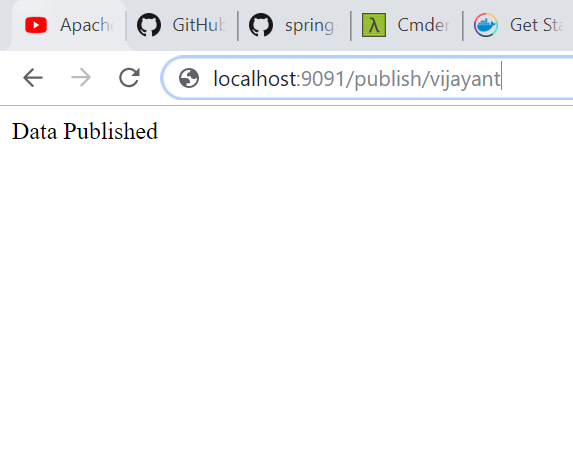
Then in the Controller/Service Layer we listen for String / Custom object based messages separately

By using @KafkaListener annotation at our methods and providing information like groupIds and topic name .

SNAPSHOT OF THE @KafkaListener Config

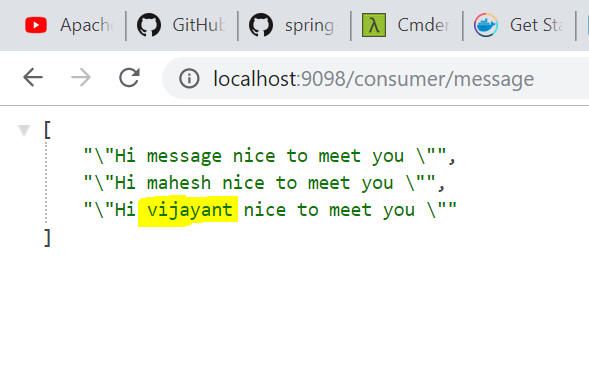


TESTING THE CONSUMER SIDE



Publishing a message String

Consumption Verified



Object type message consumed

