Task - create Multiple Consumer App

- producer one app (pub-sub true);

21-08-19

1. Spring Boot + Apache Kafka :-

* Apache Kafka supports AMQP [Advanced Message Queue Protocol] for large data transfer for MQ Applications over network.

* It supports data reading / writing
  - → Databases
  - → Flat file Systems
  - → Applications (Data Streams)

** - kafka supports integration with any type of application [langue independent + plugin required ; default Java]

* Kafka supports basic concept of MQ

1. Data Streaming 2. Web Activity 3. Log Aggrigation
4. Command Oriented Components

** Kafka follows below concept even

a. Kafka Manage Broker :- kafka uses Manage Broker also called as Broker Server which supports MQ Operations.

b. Load Balancing :- Kafka supports load Balancing to avoid more traffic, i.e. called as kafka Cluster.

→ In general cluster is group of Brooker Server [1 to n]

c. Topics :- kafka supports only Topics (Pub/Sub) Model

d. BootStrap Server :- kafka ~~for~~ cluster auto-scaling is done by Boot Strap Server (At Zookeeper).

→ It behaves as RPO server

e: Data is sent to Topic in <K,V> format

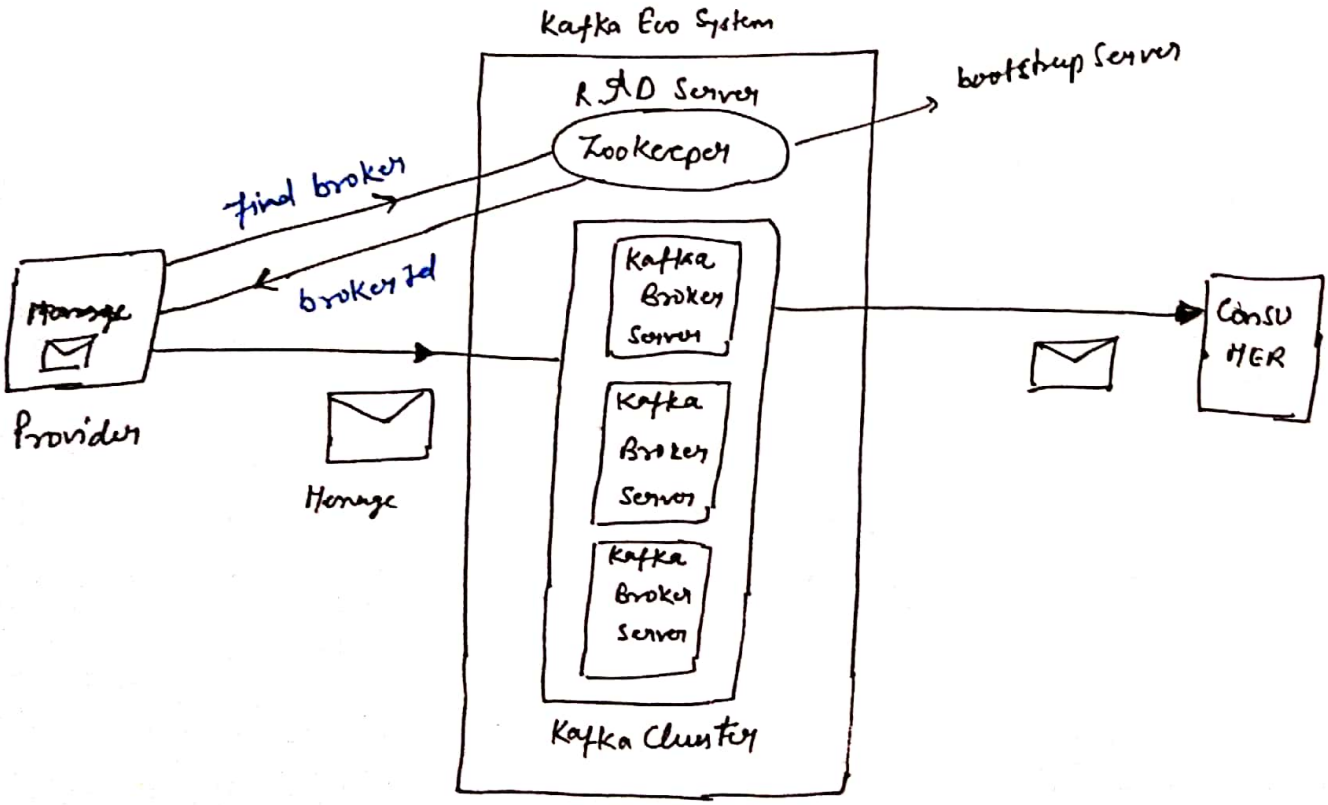
K = Topic Name [Destination] , V = Data.

f: Kafka Eco System = Kafka cluster + Bootstrap server

g: Data Partions :- Data partations concept is used by kafka to send large Data.

h: Data Persist :- Message Replica supports Data Persist Engine, to avoid data lose is case of "No consumer Available", network down, No broker response / restart ... etc".

Kafka Eco System



* Work Flow :- Producer Application communicates with bootstrap server, to get Instance Id of broker Server.

→ Producer use Kafka ~~Template~~ KafkaTemplate <K,V> to send data to one broker server.

→ Broker gets data into Topic (Partations) and stores Menage Instances

→ Based on Group Id and Topic Name broker sends menage to consumer App.

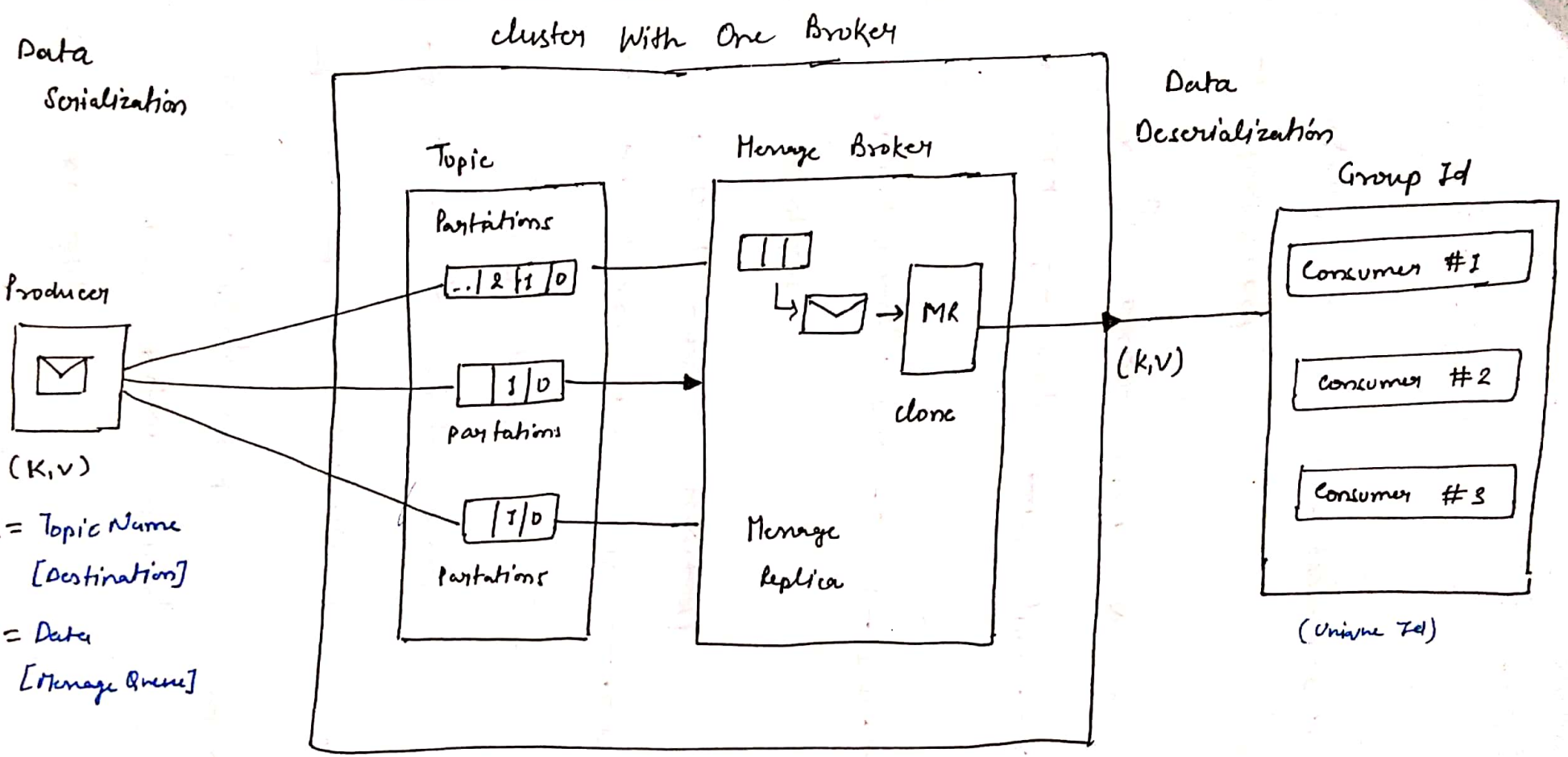
*→ Producer communicates with bootstrap Server i.e. Zookeeper to find broker. Based on load balancing it returns one broker id.

→ Then Producer send menage to this broker with the help of KafkaTemplate<K,V> . , K = topic Name, V = Menage.

→ First it goes to topic where the menage gets partioned into multiple partitions and send to broker server. called Serialization

→ Where it converted back into menage and goes to Menage Replica. , where it gets stored depending on no. of consumers the menage gets cloned.

→ After this the menage again converted to partations and send to consumers having same topic Name but every consumer must have unique id. This procen is called data Deserialization.



22-8-19

:- Kafka Setup :-

Apache Kafka software comes with unix based software called as Scala 2.x

→ GoTo → http:// to Kafka.apache.org/downloads

→ Choose one mirrors → Extract one → .tar → extract again → folder

→ Open folder and create — .bat file for Bootstrap Server (Zookeeper)

→ server.bat

.\bin\windows\zooker-server-start.bat .\config\zooker.properties

→ Create One more batch files .bat for kafka manage brokers group, known as kafka cluster

→ cluster.bat

.\bin\windows\kafka-server-start.bat .\config\server.properties

→ First start server.bat then cluster.bat

:- Kafka Application Steps :-

1. Choose spring apache kafka dependency while creating project (which gives integration of Spring with kafka)

```
<dependency>
    <g.Id> org.sf.boot kafka </groupId>
    <a.Id> spring-kafka </artifactId>
</dependency>
```

2. provide producer and consumer details with bootstrap server in application.properties

application.properties :-

```
server.port = 9988

kafka-topic-name = my-topic

spring.kafka.producer.bootstrap-servers = localhost:9092

spring.kafka.producer.key-serialization = org.apache.kafka.
                                          common.serialization.StringSerializer

spring.kafka.consumer.value-serializer = org.apache.kafka.
                                         common.serialization.StringSerializer
```

For consumer -

spring.kafka.consumer.bootstrap-servers = localhost:9092

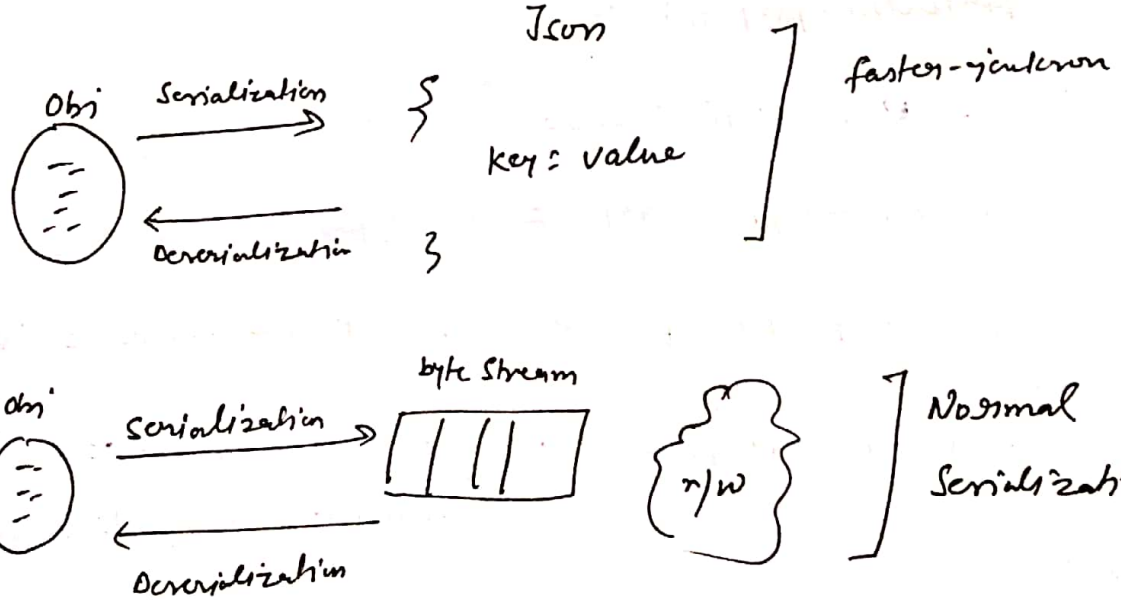
spring.kafka.consumer.group-id = group-id

spring.kafka.consumer.key-deserializer = org.apache.kafka.common.serialization.StringDeserializer

spring.kafka.consumer.value-deserializer = org.apache.kafka.common.serialization.StringDeserializer

3. Use KafkaTemplate<K,V> at producer application to send message (V) to given Topic (K)

→ Use @KafkaListener(topics = "---", groupId = "---") at consumer side to read message (V) using Topic (K) with groupId (G)



Coding Order →

1. application.properties
2. Message Storage.java
3. ConsumerService.java
4. ProducerService.java
5. KafkaRestController.java

Execution -

Start Zooker server

Start Kafka server (Cluster)

Run application Starter class

In browser -

http://localhost:9988/kafka/send?message=Hello

http://localhost:9988/kafka/all