**KAFKA INSTALLATION AND WORKING IN COMMAND LINE INTERFACE**

Step 1: Download Apache Kafka

Step 2: Extract the Kafka Archive and save in C drive

Step 3: Configure Environment Variables (System Variable)

KAFKA\_HOME to point to the kafka folder

Path %KAFKA\_HOME%\bin

Step 4: Edit config files in config folder as under

Change the path of Zookeeper.properties as follows

dataDir=/tmp/zookeeper🡪 dataDir=C:/????/zookeeper

Change the path of server.properties as follows

log.dirs=/tmp/kafka-logs🡪 log.dirs=C:/????/kafka-logs

???? stands for the folder name, press f2 to get folder name

Step 5: Start Zookeeper

Open cmd from kafak folder and type the following command

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

do not close this window – zookeeper should be running

Default port of zookeeper is 2181

Step 6: Start the Kafka Broker

Open another cmd from kafka folder and type

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

now both zookeeper and broker are running

Default port of kafka server (broker) is 9092

#### [CREATE A TOPIC TO STORE YOUR EVENTS](https://kafka.apache.org/documentation/#quickstart_createtopic)

In simple terms a topic is similar to a folder in a file system, and the events are the files in that folder.

bin\windows\kafka-topics.bat --create --topic myFirstTopic --bootstrap-server localhost:9092

bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 --create --topic SecondTopic --partitions 3 --replication-factor 1

DO NOT TRY TO DELETE A TOPIC

#### [TO](https://kafka.apache.org/documentation/#quickstart_createtopic) DESCRIBE THE TOPIC

bin\windows\kafka-topics.bat --describe --topic myFirstTopic --bootstrap-server localhost:9092

OR

bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 --describe --topic myFirstTopic

#### TO LIST ALL THE TOPICS

bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 --list

#### WRITE SOME EVENTS (messages) INTO THE TOPIC - Producer

bin\windows\kafka-console-producer.bat --topic myFirstTopic --bootstrap-server localhost:9092

#### READ THE EVENTS - Consumer

bin\windows\kafka-console-consumer.bat --topic myFirstTopic --from-beginning --bootstrap-server localhost:9092

**Kafka with Spring Boot**

1. Start the kafka-zookeeper.bat

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

2. Start the kafka-server-start.bat

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

3. Create a spring boot application with required dependencies. (web, Spring for Apache Kafka)

<dependency>

<groupId>org.springframework.kafka</groupId>

<artifactId>spring-kafka</artifactId>

</dependency>

In the application.properties file mention where kafka producer bootstrap is running

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

Note: if you have multiple brokers, you can list them separated by commas like

spring.kafka.producer.bootstrap-servers=broker1:9092,broker2:9092,broker3:9092

These two lines are for serialization. In both the cases StringSerializer is used. For Interger it will be IntegerSerializer and for JSON it will be JsonSerializer etc.

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

4. In this example, I created a class named KafkaMessagePublisher.

From our application in order to talk to kafka-server there is a class called KafkaTemplate.

In this example I have autowired a KafkaTemplate with String as key and Object as value

*@Autowired*

private KafkaTemplate<String, Object> template;

create a method to send message to the topic

public void sendMessageToTopic(String message) {

CompletableFuture<SendResult<String, Object>> future = template.send("FirstTopic", message);

future.whenComplete((result, ex)->{

if(ex==null) {

System.***out***.println("Sent Message = ["+message

+ "] with offset = [" + result.getRecordMetadata().offset()+"]");

}else {

System.***out***.println("Unable to Sent Message = [ "+message

+ "] due to = [" +ex.getMessage()+"]");

}

});

}

Run the application as usual. Hit the end point in postman five times giving different messages. It can be seen that all the five messages are stored in offsets starting from 0.

**Now for Bulk Message Transfer**

One of the major advantages of kafka server is load balancing. Load balancing means when huge data is produced kafka will store the data in different partitions on its own.

Create another topic from the command prompt with 3 partitions.

For that I have created another method named sendBulkMessage.

Run the application and hit the end point to see if the messages are stored in different partitions. Its upto the zookeeper.

**To create Topic programmatically**

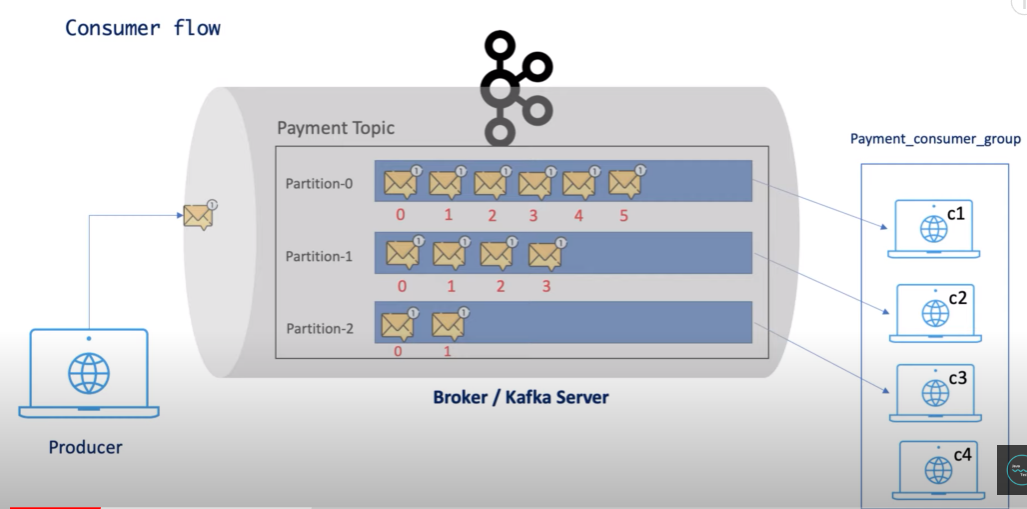
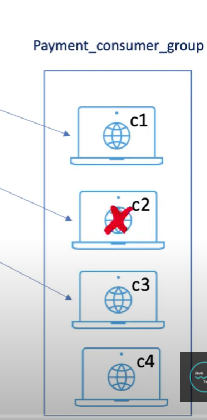
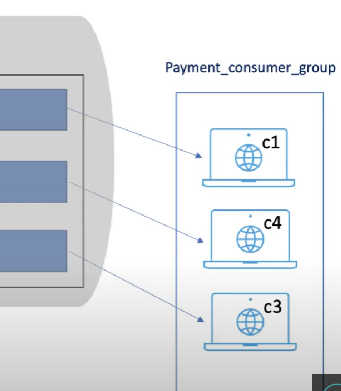
To create a topic programmatically we have a class called the NewTopic. Create a configuration class. Create a method to return a bean of type NewTopic. Run the application, a new topic will be created with the details given in the config class @Bean method.

Now Create a method in service giving the name of the topic created programmatically. Create an end point in the controller, send messages as required.

**Consumer**

Without having a consumer application to consume the messages in topics is meaningless.

For each partition there will only be one consumer listening. That is if we have three partitions, there should be at least three consumers. If we have a fourth consumer, he will be waiting for a future assignment. Suppose if any one of the three partitions die, then the fourth partition will take the place of that partition. This is known as **consumer Re-balancing**

Treasamma@Twilio1964

Recovery Code for Twilio Account = E5BJ5ALGSSRSEGE2ZGNF5JYP

Account SID - AC5b187a82792c4de6d681a18d86601f32

Auth Token - d69e5b346bf676b2ed57fce9639bbf73