**Kafka Windows Installation**

1. Download Kafka

Link 1: <https://www.apache.org/dyn/closer.cgi?path=/kafka/3.3.1/kafka_2.13-3.3.1.tgz>

1. Rename folder into “Kafka” and move into C drive
2. Go to Config and Change the server directory to C:/Kafka/kafka-logs also, zookeeper to C:/Kafka/zookeeper-data
3. Go to command prompt and get the Kafka directory path. Then enable zookeeper by typing:

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

Zookeeper starts on port 2181

1. Then start the Kafka Server in another Terminal

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

Server/broker starts on port 9092

1. Now Create topic. For that open another terminal and type:

C: \ Kafka> .\bin\windows\kafka-topics.bat --create --topic my-events --bootstrap-server localhost:9092

You can check the already created Kafka topics:

C: \ Kafka>.\bin\windows\kafka-topics.bat --bootstrap-server=localhost:9092 –list

my-events, is the created Kafka topic

1. Next, create events to do that run Kafka producer (to produce events) and write events to Kafka topic

C: \ Kafka>.\bin\windows\kafka-console-producer.bat --topic my-events --bootstrap-server localhost:9092

>Hello world

>welcome to Kafka

Exit? CTRL + C

1. Now read the written events from Kafka topic

C: \ Kafka>.\bin\windows\kafka-console-consumer.bat --topic my-events --from-beginning --bootstrap-server localhost:9092

Graphical user interface, text, application, email

Description automatically generated

**Set-up Spring boot Application in IntelliJ**

1. Create a Spring boot Application with following dependencies.

* Spring Web
* Spring for Apache Kafka

1. Kafka application. Properties

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

if there are multiple servers how to configure

**spring. kafka.consumer.bootstrap-servers**=**localhost:9092,localhost:8081**

Include following properties in the application. Properties

Graphical user interface, text, application

Description automatically generated

Created “Topic1” in Spring boot and build

Graphical user interface, text, application

Description automatically generated

Consumes Kafka Template to implement sendMessage () function

Graphical user interface, text, application, email

Description automatically generated

Rest Controller use *//http://localhost:8080/api/kafka/publish?message=hello world*

Text, application

Description automatically generated

Check the terminal and use kafka-console-consumer.bat to read Kafka topic and execute following command

C: \ Kafka>.\bin\windows\kafka-console-consumer.bat --topic topic1 --from-beginning --bootstrap-server localhost:9092

Output: Hello world

Consumer code:

Graphical user interface, text, application, email

Description automatically generated

Hit url *//http://localhost:8080/api/kafka/publish?message=hello world*

You can see the outputs in IntelliJ Terminal

**Configure Kafka Producer and Consumer for JSON Serializer and De- Serializer**

* This specifies how to send and receive a Java object as a JSON byte array to/from Apache Kafka. Apache Kafka stores and transports byte array.
* Change the application. Properties file for JSON serializer and De-serializer. (Comment value code of serialization and De-serialize)

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Since you are getting Listener exception, create a json new Kafka topic as below

Timeline

Description automatically generated

Then change the topic name in JasonProducer

Graphical user interface, text, application, email

Description automatically generated

Send the postman request again.

Later got to command-prompt and change the topic name to topic1\_json and check whether json object published in Kafka topic.



Create JsonKafkaConsumer to convert **Json byte array to User object**

Graphical user interface, text, application, email

Description automatically generated

**Kafka Wikimedia Project**

Create a multi-model package

First add <**packaging**>pom</**packaging**> in parent dependency to make multi-model project

<**groupId**>com.ab</**groupId**>  
<**artifactId**>wikimedia-kafka-project</**artifactId**>  
<**version**>0.0.1-SNAPSHOT</**version**>  
<**name**>wikimedia-kafka-project</**name**>  
<**description**>Demo project for Spring Boot</**description**>  
<**packaging**>pom</**packaging**>

Right click on parent project->new->module->select Maven->tick create from archetype check box -> give a name “kafka-producer-wikimedia”

Create a package->main method in “kafka-producer-wikimedia” and run the application

Then go to pom.xml in “kafka-producer-wikimedia” and add <**packaging**>jar</**packaging**> (basically modules convert to jar files)

Graphical user interface, text, application

Description automatically generated

Go to Execute Maven goal icon and type “mvn clean install” execute

Then go to “kafka-producer-wikimedia” resource and create a file called application. Properties file and add following configurations

Text

Description automatically generated

In “kafka-producer-wikimedia” create following packages and java classes

Text, timeline

Description automatically generated

//add event source dependency (retrieve stream data)

<https://mvnrepository.com/artifact/com.launchdarkly/okhttp-eventsource/2.7.1>

Text

Description automatically generated

Since all the data in json format -> have to provide Jackson json core and databind dependency as well (this dependency to deal with json data)

jackson-core:

<https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core/2.14.0>

jackson-databind:

<https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind/2.14.0>

Give dependency version as following (Strictly)

Text

Description automatically generated with medium confidence

Create EventHandler class as WikimediaChangesHandler

Graphical user interface, text, application, email

Description automatically generated

Implement sendMessage() using eventHandler

Text

Description automatically generated

Go to the main function

Graphical user interface, text, application, email

Description automatically generated

output

Text

Description automatically generated

Go to C:/Kafka> .\bin\windows\kafka-console-consumer.bat --topic wikimedia\_topic --from-beginning --bootstrap-server localhost:9092

**Consumer**

Follow the same steps, which have done for the producer

Create another module for kafka-consumer-database->include main method ->add packaging jar in pom also add application properties as following.

Text

Description automatically generated

Add following class in the services package

Graphical user interface, text, application, email

Description automatically generated

Then execute the kafka-consumer-database main application -> you see the output as following

Text

Description automatically generated

Store Consumer Data in Database

Create Wikimedia database in MYSQL.

Add dependencies(JPA and MySQL) into pom of kafka-consumer-database

<**dependencies**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-data-jpa</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>com.mysql</**groupId**>  
 <**artifactId**>mysql-connector-j</**artifactId**>  
 <**scope**>runtime</**scope**>  
 </**dependency**>  
</**dependencies**>

Add db configurations in application properties

Graphical user interface, text, application

Description automatically generated

Create a model class

Graphical user interface, text, application

Description automatically generated

Create an interface to implement JPA repository in repositories package

Graphical user interface, text, application, email

Description automatically generated

Change the service class accordingly by auto wiring the repository

Text, application

Description automatically generated

Execute the main and you will see following output

Graphical user interface

Description automatically generated with low confidence