**Kafka Documentation**

**How to Install and Run Apache Kafka on Windows?**

**Step 1:** Copy the path of the Kafka folder. Now go to *config* inside kafka folder and open *zookeeper.properties* file. Copy the path against the field *dataDir* and add */zookeeper-data* to the path.

A screen shot of a computer

Description automatically generated  
For example, if the path is *c:/kafka*

**Step 2:** Now in the same folder *config* open *server.properties* and scroll down to *log.dirs* and paste the path. To the path add */kafka-logs*

**Step 3: Start the ZooKeeper service**

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

**Step 4 :Start the Kafka broker service**

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

**Step 5: Create Topic**

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

Example:-

.\bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 --create --topic javatechi-demo2 --partitions 3 --replication-factor 1

**Step6: Describe Topic**

.\bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 --describe --topic javatechi-demo1

**Create Producer Application: -**

**1)As below snap, create project/Add dependencies from spring initialize then import into IntelliJ**

**A screenshot of a computer

Description automatically generated**

**2)Create Project structure as below.**

**A screenshot of a computer program

Description automatically generated**

**Files need to add/modify.**

**1)application.yml**

server:  
 port: 9191  
  
spring:  
 kafka:  
 producer:  
 bootstrap-servers: localhost:9092

**2)** KafkaMessagePublisher

package com.javatechi.service;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.kafka.core.KafkaTemplate;  
import org.springframework.kafka.support.SendResult;  
import org.springframework.stereotype.Service;  
import java.util.concurrent.CompletableFuture;  
@Service  
public class KafkaMessagePublisher {  
 @Autowired  
 private KafkaTemplate<String,Object> template;  
 public void sendMessageToTopic(String message){  
 CompletableFuture<SendResult<String, Object>> future = template.send("javatechi-demo1", message);  
 future.whenComplete((result,ex) ->{  
 if (ex==null){  
 System.*out*.println("Send Message=[" +message+  
 "] with offset=[" +  
 result.getRecordMetadata().offset() +"]");  
 }  
 else {  
 System.*out*.println("Unable to send message=["+  
 message + "] due to :" + ex.getMessage());  
 }  
 });

**3)EventController**

package com.javatechi.controller;  
import com.javatechi.service.KafkaMessagePublisher;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
@RequestMapping("/produce-app")  
public class EventController {  
  
 @Autowired  
 private KafkaMessagePublisher publisher;  
  
 @GetMapping("/publish/{message}")  
 public ResponseEntity<?> publishMessage(@PathVariable String message){  
 try {  
 publisher.sendMessageToTopic(message);  
 return ResponseEntity.*ok*("Message Published successfully...");  
 }catch (Exception ex){  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*)  
 .build();  
  
 }  
  
 }  
}

**4)KafkaProducerExampleApplication**

package com.javatechi;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class KafkaProducerExampleApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(KafkaProducerExampleApplication.class, args);  
 }  
  
}

**5)Published message through postmen rest**

**A screenshot of a computer

Description automatically generated**

**6)download offset explorer/configure**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**How to create a topic programmatically.**

**1) create below class in config folder**

package com.javatechi.config;  
import org.apache.kafka.clients.admin.NewTopic;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
  
@Configuration  
public class KafkaProducerConfig {  
 @Bean  
 public NewTopic createTopic(){  
  
 return new NewTopic("javatechi-demo3",5, (short) 1);  
 }  
}

**Create Consumer Application.**

**1)First case where one consumer instance assigned/listening from multiple partitioned (This is not good case and won’t give better throuput)**

**A screenshot of a video

Description automatically generated**

**1)application.yml.**

spring:  
 kafka:  
 consumer:  
 bootstrap-servers: localhost:9092  
 group-id: jt-group-1  
  
  
server:  
 port: 9292

**2)KafkaListener**

package com.javatechi.consumer;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.kafka.annotation.KafkaListener;  
import org.springframework.stereotype.Service;  
  
@Service  
public class KafkaMessageListener {  
  
 Logger log = LoggerFactory.*getLogger*(KafkaMessageListener.class);  
  
@KafkaListener(topics = "javatechi-demo4",groupId ="jt-group-1" )  
 public void consume(String message){  
  
 log.info("Consumer consume the messages{}" , message);  
  
  
 }  
  
  
}

**Process to see consuming messages.**

**1)Start Producer Application**

**2)Start Consumer Application**

**3)Send 10,000 messaged from postman as below snap.**

**A screenshot of a computer

Description automatically generated**

**2)check offset explorer regarding status of consuming message**

**A screenshot of a computer

Description automatically generated**

**2)Good Case:-**

**As per below snap, we have three partitions and each partition listening by different consumer Instance**

**A screenshot of a computer

Description automatically generated**

**Consumer rebalancing. Here the consumer instance four seating ideal and when if any consumer die, then it will start working.**

**It is just like; IT company hire software engineer for future project and he is seating ideal until future project started**

**A screen shot of a computer

Description automatically generated**

**Here are four consumer instances created.**

package com.javatechi.consumer;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.kafka.annotation.KafkaListener;  
import org.springframework.stereotype.Service;  
@Service  
public class KafkaMessageListener {  
 Logger log = LoggerFactory.*getLogger*(KafkaMessageListener.class);  
 @KafkaListener(topics = "javatechi-common",groupId ="jt-group-common" )  
 public void consume1(String message){  
 log.info("Consumer1 consume the messages{}" , message);  
 }  
 @KafkaListener(topics = "javatechi-common",groupId ="jt-group-common" )  
 public void consume2(String message){  
 log.info("Consumer2 consume the messages{}" , message);  
 }  
 @KafkaListener(topics = "javatechi-common",groupId ="jt-group-common" )  
 public void consume3(String message){  
 log.info("Consumer3 consume the messages{}" , message);  
 }  
 @KafkaListener(topics = "javatechi-common",groupId ="jt-group-common" )  
 public void consume4(String message){  
 log.info("Consumer4 consume the messages{}" , message);  
 }  
}

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**References: -**

[**https://www.youtube.com/watch?v=rVqAoUIPO7I**](https://www.youtube.com/watch?v=rVqAoUIPO7I)

[**https://www.geeksforgeeks.org/how-to-install-and-run-apache-kafka-on-windows/amp/**](https://www.geeksforgeeks.org/how-to-install-and-run-apache-kafka-on-windows/amp/)

[**https://www.kafkatool.com/documentation/connecting.html**](https://www.kafkatool.com/documentation/connecting.html)