<https://dzone.com/articles/running-apache-kafka-on-windows-os>

ZooKeeper:

Zkserver.cmd

**Running Kafka Server**

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

### Creating topics

kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic kafka\_example

### Creating a Producer and Consumer to Test Server

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

kafka-console-consumer.bat --zookeeper localhost:9092 --topic kafka\_example

## Some Other Useful Commands

1. List Topics: kafka-topics.bat --list --zookeeper localhost:2181
2. Describe Topic: kafka-topics.bat --describe --zookeeper localhost:2181 --topic [Topic Name]
3. Read messages from beginning: kafka-console-consumer.bat --zookeeper localhost:2181 --topic [Topic Name] --from-beginning
4. Delete Topic: kafka-run-class.bat kafka.admin.TopicCommand --delete --topic [topic\_to\_delete] --zookeeper localhost:2181

Kafka Consumer :

Pom.xml

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

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

<artifactId>spring-kafka</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

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

<artifactId>spring-kafka-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

Kafka Consumer :

**import** org.apache.kafka.clients.consumer.ConsumerConfig;

**import** org.apache.kafka.common.serialization.StringDeserializer;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.kafka.annotation.EnableKafka;

**import** org.springframework.kafka.config.ConcurrentKafkaListenerContainerFactory;

**import** org.springframework.kafka.core.ConsumerFactory;

**import** org.springframework.kafka.core.DefaultKafkaConsumerFactory;

**import** org.springframework.kafka.support.serializer.JsonDeserializer;

**import** com.sachin.kafka.consumer.kafkaconsumer.pojo.Person;

@EnableKafka

@Configuration

**public** **class** KafkaConsumerConfiguration {

@Bean

**public** ConsumerFactory<String, String> consumerFactory(){

Map<String, Object> configs = **new** HashMap<String,Object>();

configs.put(ConsumerConfig.***BOOTSTRAP\_SERVERS\_CONFIG***, "localhost:9092");

configs.put(ConsumerConfig.***GROUP\_ID\_CONFIG***, "group\_id");

configs.put(ConsumerConfig.***KEY\_DESERIALIZER\_CLASS\_CONFIG***, StringDeserializer.**class**);

configs.put(ConsumerConfig.***VALUE\_DESERIALIZER\_CLASS\_CONFIG***, StringDeserializer.**class**);

**return** **new** DefaultKafkaConsumerFactory<>(configs);

}

@Bean

**public** ConcurrentKafkaListenerContainerFactory<String, String> kafkaListenerContainerFactory(){

ConcurrentKafkaListenerContainerFactory<String, String> factory = **new** ConcurrentKafkaListenerContainerFactory<>();

factory.setConsumerFactory(consumerFactory());

**return** factory;

}

@Bean

**public** ConsumerFactory<String, Person> consumerFactoryJson(){

Map<String,Object> configs = **new** HashMap<>();

configs.put(ConsumerConfig.***BOOTSTRAP\_SERVERS\_CONFIG***, "localhost:9092");

configs.put(ConsumerConfig.***GROUP\_ID\_CONFIG***, "group\_json");

configs.put(ConsumerConfig.***KEY\_DESERIALIZER\_CLASS\_CONFIG***, StringDeserializer.**class**);

configs.put(ConsumerConfig.***VALUE\_DESERIALIZER\_CLASS\_CONFIG***, JsonDeserializer.**class**);

**return** **new** DefaultKafkaConsumerFactory<>(configs, **null** , **new** JsonDeserializer<Person>(Person.**class**));

}

@Bean

**public** ConcurrentKafkaListenerContainerFactory<String, Person> kafkaListenerContainerFactoryJson(){

ConcurrentKafkaListenerContainerFactory<String, Person> factory = **new** ConcurrentKafkaListenerContainerFactory<>();

factory.setConsumerFactory(consumerFactoryJson());

**return** factory;

}

}

**Service:**

import org.springframework.kafka.annotation.KafkaListener;

import org.springframework.stereotype.Service;

import com.sachin.kafka.consumer.kafkaconsumer.pojo.Person;

@Service

public class KafkaConsumer {

@KafkaListener(topics = "kafka\_consumer1", groupId = "group\_id")

public void consumer(String message) {

System.out.println("Kafka Listener:" + message);

}

@KafkaListener(topics = "kafka\_consumer1", groupId = "group\_json",

containerFactory = "kafkaListenerContainerFactoryJson")

public void consumeJson(Person person) {

System.out.println("Consumed JSON Message: " + person);

}

}

**Model:**

**public** **class** Person {

**private** String name;

**private** String last;

Person(){ //All way implement default constructor

}

**public** Person(String name, String last) {

**super**();

**this**.name = name;

**this**.last = last;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getLast() {

**return** last;

}

**public** **void** setLast(String last) {

**this**.last = last;

}

@Override

**public** String toString() {

**return** "Person [name=" + name + ", last=" + last + "]";

}

}

YML file configuration:

spring:

kafka:

consumer:

bootstrapservers: http://199.968.98.101:9092

group-id: groupid-QA-02

auto-offset-reset: latest

key-deserializer: org.apache.kafka.common.serialization.StringDeserializer

value-deserializer: org.apache.kafka.common.serialization.StringDeserializer

tpd:

  topic-name: advice-topic

  messages-per-request: 10

Producer:

Pom.xml

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

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

<artifactId>spring-kafka</artifactId>

</dependency>

Producer Configuration Class:

**import** org.apache.kafka.clients.producer.ProducerConfig;

**import** org.apache.kafka.common.serialization.StringSerializer;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.kafka.core.DefaultKafkaProducerFactory;

**import** org.springframework.kafka.core.KafkaTemplate;

**import** org.springframework.kafka.core.ProducerFactory;

**import** org.springframework.kafka.support.serializer.JsonSerializer;

**import** com.sachin.kafka.KafkaExample.pojo.User;

@Configuration

**public** **class** KafkaConfiguration {

//For the string message no need to explicit configuration comes as default

@Bean

**public** ProducerFactory<String, User> producerFactoryJson() {

Map<String, Object> configs = **new** HashMap<>();

configs.put(JsonSerializer.***ADD\_TYPE\_INFO\_HEADERS***, **false**);

configs.put(ProducerConfig.***BOOTSTRAP\_SERVERS\_CONFIG***, "127.0.0.1:9092");

configs.put(ProducerConfig.***KEY\_SERIALIZER\_CLASS\_CONFIG***, StringSerializer.**class**);

configs.put(ProducerConfig.***VALUE\_SERIALIZER\_CLASS\_CONFIG***, JsonSerializer.**class**);

**return** **new** DefaultKafkaProducerFactory(configs);

}

@Bean

**public** KafkaTemplate<String, User> kafkaTemplate(){

**return** **new** KafkaTemplate<>(producerFactoryJson());

}

}

Producer Controller:

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.kafka.core.KafkaTemplate;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.sachin.kafka.KafkaExample.pojo.User;

@RestController

@RequestMapping("/kafka")

**public** **class** UserController {

@Autowired

KafkaTemplate<String, User> kafka;

**private** **static** **final** String ***TOPICNAME*** = "kafka\_consumer1";

@RequestMapping(method = RequestMethod.***POST*** , value="/publishUser")

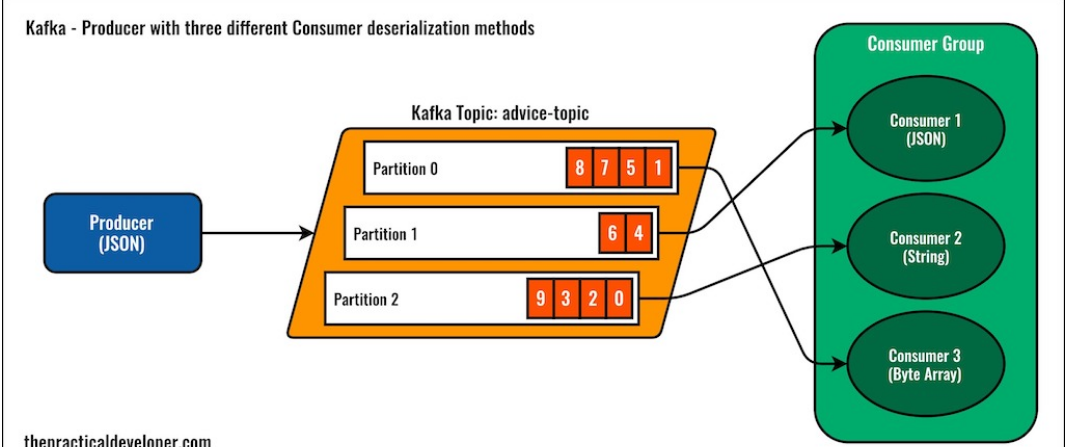
**public** User getUser(@RequestBody User user) {

kafka.send(***TOPICNAME***,user);

**return** user;

}

}



**Kafka messages with the same key are always placed in the same partitions**

**Message Class:**

import com.fasterxml.jackson.annotation.JsonProperty;

public class PracticalAdvice {

    private final String message;

    private final int identifier;

    public PracticalAdvice(@JsonProperty("message") final String message,

                           @JsonProperty("identifier") final int identifier) {

        this.message = message;

        this.identifier = identifier;

    }

    public String getMessage() {

        return message;

    }

    public int getIdentifier() {

        return identifier;

    }

    @Override

    public String toString() {

        return "PracticalAdvice::toString() {" +

                "message='" + message + '\'' +

                ", identifier=" + identifier +

                '}';

    }

}

**Producer Configuration:**

@SpringBootApplication

public class KafkaExampleApplication {

    public static void main(String[] args) {

        SpringApplication.run(KafkaExampleApplication.class, args);

    }

    @Autowired

    private KafkaProperties kafkaProperties;

    @Value("${tpd.topic-name}")

    private String topicName;

    // Producer configuration

    @Bean

    public Map<String, Object> producerConfigs() {

        Map<String, Object> props =

                new HashMap<>(kafkaProperties.buildProducerProperties());

        props.put(ProducerConfig.KEY\_SERIALIZER\_CLASS\_CONFIG,

                StringSerializer.class);

        props.put(ProducerConfig.VALUE\_SERIALIZER\_CLASS\_CONFIG,

                JsonSerializer.class);

        return props;

    }

    @Bean

    public ProducerFactory<String, Object> producerFactory() {

        return new DefaultKafkaProducerFactory<>(producerConfigs());

    }

    @Bean

    public KafkaTemplate<String, Object> kafkaTemplate() {

        return new KafkaTemplate<>(producerFactory());

    }

    @Bean

    public NewTopic adviceTopic() {

        return new NewTopic(topicName, 3, (short) 1);

    }

}

Controller:

@RestController

public class HelloKafkaController {

    private static final Logger logger =

            LoggerFactory.getLogger(HelloKafkaController.class);

    private final KafkaTemplate<String, Object> template;

    private final String topicName;

    private final int messagesPerRequest;

    private CountDownLatch latch;

    public HelloKafkaController(

            final KafkaTemplate<String, Object> template,

            @Value("${tpd.topic-name}") final String topicName,

            @Value("${tpd.messages-per-request}") final int messagesPerRequest) {

        this.template = template;

        this.topicName = topicName;

        this.messagesPerRequest = messagesPerRequest;

    }

    @GetMapping("/hello")

    public String hello() throws Exception {

        latch = new CountDownLatch(messagesPerRequest);

        IntStream.range(0, messagesPerRequest)

                .forEach(i -> this.template.send(topicName, String.valueOf(i),

                        new PracticalAdvice("A Practical Advice", i))

                );

        latch.await(60, TimeUnit.SECONDS);

        logger.info("All messages received");

        return "Hello Kafka!";

    }

}

**Consumer Configuration:**

@SpringBootApplication

public class KafkaExampleApplication {

    public static void main(String[] args) {

        SpringApplication.run(KafkaExampleApplication.class, args);

    }

    @Autowired

    private KafkaProperties kafkaProperties;

    @Value("${tpd.topic-name}")

    private String topicName;

    @Bean

    public ConsumerFactory<String, Object> consumerFactory() {

        final JsonDeserializer<Object> jsonDeserializer = new JsonDeserializer<>();

        jsonDeserializer.addTrustedPackages("\*");

        return new DefaultKafkaConsumerFactory<>(

                kafkaProperties.buildConsumerProperties(), new StringDeserializer(), jsonDeserializer

        );

    }

    @Bean

    public ConcurrentKafkaListenerContainerFactory<String, Object> kafkaListenerContainerFactory() {

        ConcurrentKafkaListenerContainerFactory<String, Object> factory =

                new ConcurrentKafkaListenerContainerFactory<>();

        factory.setConsumerFactory(consumerFactory());

        return factory;

    }

    // String Consumer Configuration

    @Bean

    public ConsumerFactory<String, String> stringConsumerFactory() {

        return new DefaultKafkaConsumerFactory<>(

                kafkaProperties.buildConsumerProperties(), new StringDeserializer(), new StringDeserializer()

        );

    }

    @Bean

    public ConcurrentKafkaListenerContainerFactory<String, String> kafkaListenerStringContainerFactory() {

        ConcurrentKafkaListenerContainerFactory<String, String> factory =

                new ConcurrentKafkaListenerContainerFactory<>();

        factory.setConsumerFactory(stringConsumerFactory());

        return factory;

    }

    // Byte Array Consumer Configuration

    @Bean

    public ConsumerFactory<String, byte[]> byteArrayConsumerFactory() {

        return new DefaultKafkaConsumerFactory<>(

                kafkaProperties.buildConsumerProperties(), new StringDeserializer(), new ByteArrayDeserializer()

        );

    }

    @Bean

    public ConcurrentKafkaListenerContainerFactory<String, byte[]> kafkaListenerByteArrayContainerFactory() {

        ConcurrentKafkaListenerContainerFactory<String, byte[]> factory =

                new ConcurrentKafkaListenerContainerFactory<>();

        factory.setConsumerFactory(byteArrayConsumerFactory());

        return factory;

    }

}

Consumer Class :

 @KafkaListener(topics = "advice-topic", clientIdPrefix = "json",

            containerFactory = "kafkaListenerContainerFactory")

    public void listenAsObject(ConsumerRecord<String, PracticalAdvice> cr,

                               @Payload PracticalAdvice payload) {

        logger.info("Logger 1 [JSON] received key {}: Type [{}] | Payload: {} | Record: {}", cr.key(),

                typeIdHeader(cr.headers()), payload, cr.toString());

        latch.countDown();

    }

    @KafkaListener(topics = "advice-topic", clientIdPrefix = "string",

            containerFactory = "kafkaListenerStringContainerFactory")

    public void listenasString(ConsumerRecord<String, String> cr,

                               @Payload String payload) {

        logger.info("Logger 2 [String] received key {}: Type [{}] | Payload: {} | Record: {}", cr.key(),

                typeIdHeader(cr.headers()), payload, cr.toString());

        latch.countDown();

    }

    @KafkaListener(topics = "advice-topic", clientIdPrefix = "bytearray",

            containerFactory = "kafkaListenerByteArrayContainerFactory")

    public void listenAsByteArray(ConsumerRecord<String, byte[]> cr,

                                  @Payload byte[] payload) {

        logger.info("Logger 3 [ByteArray] received key {}: Type [{}] | Payload: {} | Record: {}", cr.key(),

                typeIdHeader(cr.headers()), payload, cr.toString());

        latch.countDown();

    }

    private static String typeIdHeader(Headers headers) {

        return StreamSupport.stream(headers.spliterator(), false)

                .filter(header -> header.key().equals("\_\_TypeId\_\_"))

                .findFirst().map(header -> new String(header.value())).orElse("N/A");

    }