**Kafka Cluster**: Kafka is a distributed system, it acts as agent it consist of set of Broker / software, for production cluster must have 3 Broker.



**Producer**: Produce messages any in format. (JSON, XML, Text etc.) And send to Kafka cluster.

**Consumer**: Consume messages from Kafka broker.

**Kafka Broker**: it’s a Kafka server or s/w, the producer and Consumer don’t interact directly, they use Kafka broker / agent to exchange messages.

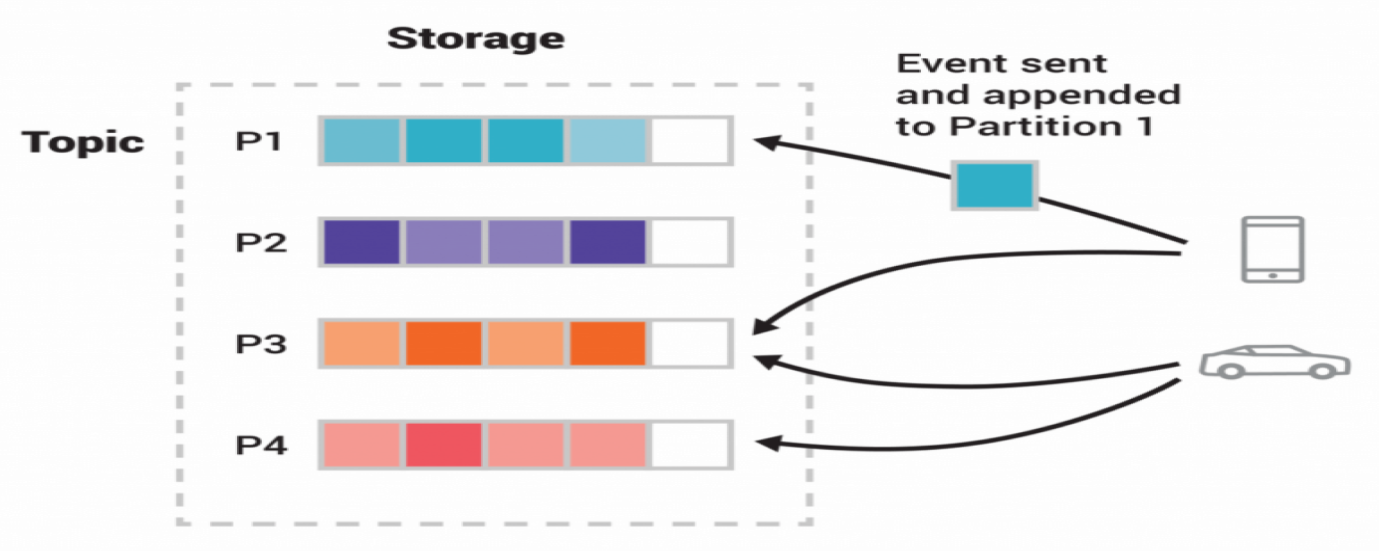
**Zookeeper**: Manages the state of the all Kafka broker in the Kafka cluster. It also maintains the configuration of all topics of producer and consumer.

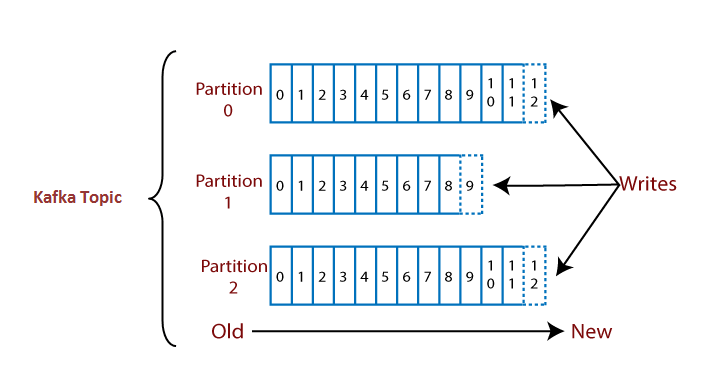
**Fault Tolerance**: If any broker gets down then automatically another broker handles the traffic as data is duplicated across the Kafka brokers in the cluster.

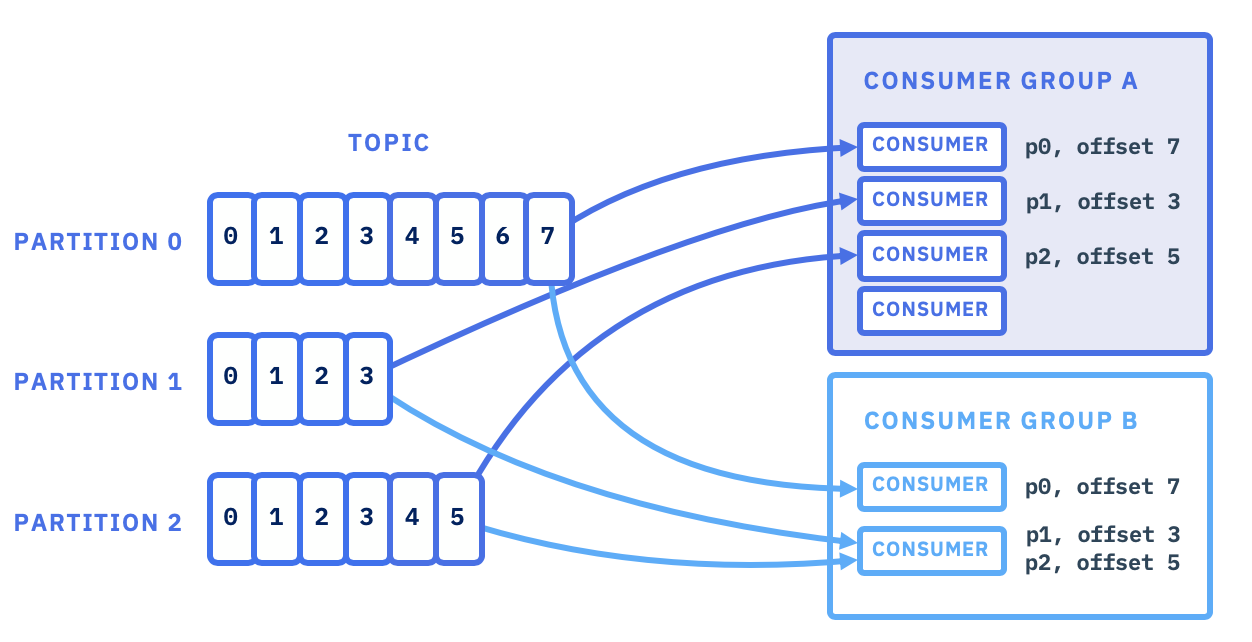
**Reference**: <https://kafka.apache.org/documentation/>

**Kafka Topic**: Is like Table in database or folder in file system, and is identified by name we can have any number of Topics in Kafka broker. Consumer subscribes the topic.

**Kafka Partition**: Topic has multiple partitions.



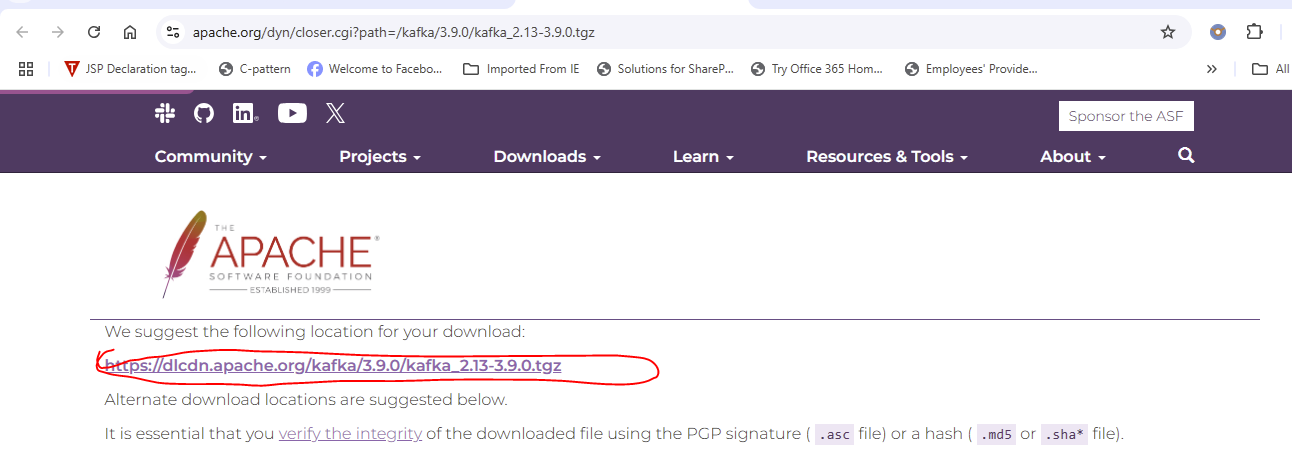
**Offset**: Is a sequence of Ids given to the messages at arrive position. Once the offset is assigned it will never changes.

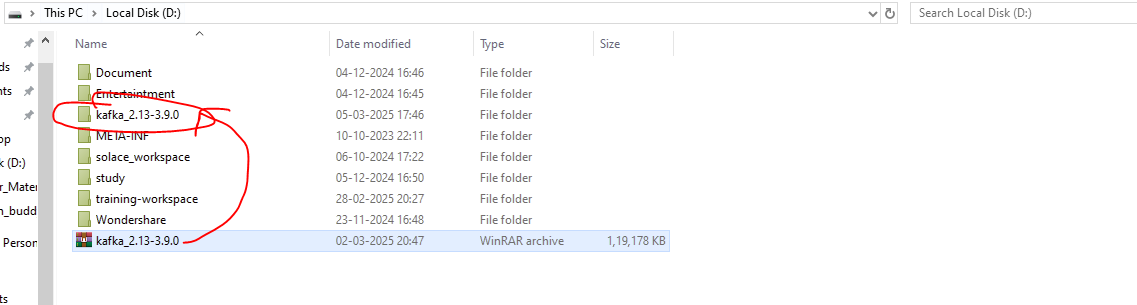
**Consumer Group**: A consumer group contains a one or more consumer’s working together to process the messages. 

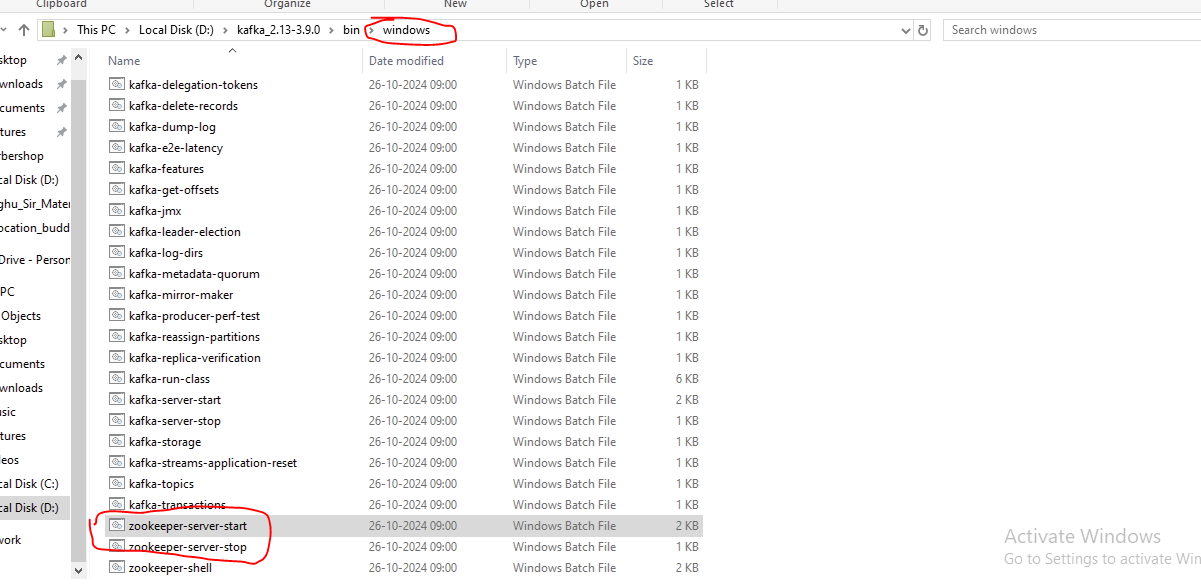
* **Can multiple consumers consume messages from same partition?**

|  |
| --- |
| Within same group: **NO**   * Two consumers (*Consumer 1, 2*) within the same group (*Group 1*) **CAN NOT** consume the same message from partition (*Partition 0*).   Across different groups: **YES**   * Two consumers in two groups (*Consumer 1* from *Group 1*, *Consumer 1* from *Group2*) CAN consume the same message from partition (*Partition 0*). |

**How to install Kafka:**

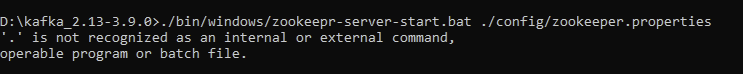
Reference: [**https://kafka.apache.org/quickstart**](https://kafka.apache.org/quickstart)Extract Here





**Start Zookeeper:**

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



**Check status if Zookeeper started**

.\bin\windows\zookeeper-shell.bat status **or find log** binding to port 0.0.0.0/0.0.0.0:2181

**Start Kafka broker:**

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



**Create Topic**

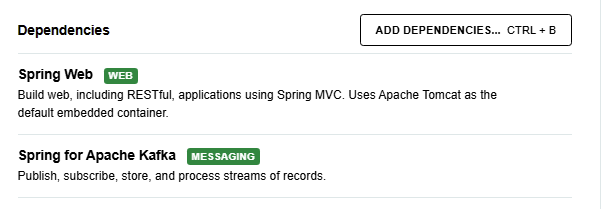
.\bin\windows\kafka-topics.bat --create --topic test-topic --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

**List Topic**

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

**List messages from topic**

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

**Create Spring Application**

Reference: <https://docs.spring.io/spring-kafka/reference/quick-tour.html>

**Create consumer and producer Application**

application.properties

|  |
| --- |
| #specify the Kafka broker(s) that the consumer will connect to for consuming message  spring.kafka.consumer.bootstrap-servers: localhost:9092 OR  spring.kafka.consumer.bootstrap-servers=broker1:9092,broker2:9092,broker3:9092 spring.kafka.consumer.group-id:myGroup spring.kafka.consumer.auto-offset-reset:earliest #Deserilise Key into Message spring.kafka.consumer.key-deserializer: org.apache.kafka.common.serialization.StringDeserializer #Deserilise value into Message spring.kafka.consumer.value-deserializer: org.apache.kafka.common.serialization.StringDeserializer  spring.kafka.producer.bootstrap-servers: localhost:9092 spring.kafka.producer.key-serializer: org.apache.kafka.common.serialization.StringSerializer spring.kafka.producer.value-serializer: org.apache.kafka.common.serialization.StringSerializer |

**Create Kafka Topic**

|  |
| --- |
| import org.apache.kafka.clients.admin.NewTopic; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.kafka.config.TopicBuilder;  @Configuration public class KafkaConfig {  @Bean  public NewTopic rlbinTopic() {  return TopicBuilder.name("rlbIn").build();  } } |

**List Kafka Topic**:

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



**Create Producer**

|  |
| --- |
| import org.springframework.kafka.core.KafkaTemplate; import org.springframework.stereotype.Service;  @Service public class KafkaProducer {  private KafkaTemplate<String, String> kafkaTemplate;   public KafkaProducer(KafkaTemplate<String, String> kafkaTemplate) {  this.kafkaTemplate = kafkaTemplate;  }   public void send(String message) {  kafkaTemplate.send("rlbIn", message);  } } |

**Create Rest API to publish Message**

|  |
| --- |
| import com.rlb.config.kafka.KafkaProducer; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.RestController;  @RestController @RequestMapping("/kafka/v1") public class KafkaController {   private KafkaProducer kafkaProducer;   public KafkaController(KafkaProducer kafkaProducer) {  this.kafkaProducer = kafkaProducer;  }   @RequestMapping("/publish")  public ResponseEntity<String> publish(@RequestParam("message") String msg) {  kafkaProducer.send(msg);  return new ResponseEntity<>("success", HttpStatus.*CREATED*);  } } |

**List Message:** .\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic rlbIn --from-beginning**Create Consumer**

|  |
| --- |
| package com.rlb.config.kafka;  import org.apache.logging.log4j.LogManager; import org.apache.logging.log4j.Logger; import org.springframework.kafka.annotation.KafkaListener; import org.springframework.stereotype.Service;  @Service public class KafkaConsumer {  private static final Logger *LOGGER* = LogManager.*getLogger*(KafkaConsumer.class);   @KafkaListener(topics = "rlbIn", groupId = "myGroup")  public void consume(String msg) {  *LOGGER*.info(String.*format*("Message received : %s",msg));  } } |

\*\*\*\*\*END\*\*\*\*\*

**JSON Deserializer and JSON Serializer**

Apache Kafka store and transport byte[], There are number of built in serializers and deserializers but it does not include any for JSON. Spring Kafka created JsonSerializer and JsonDeserializer which can use to convert Java Object to and from JSON.

application.properties

|  |
| --- |
| spring.kafka.consumer.value-deserializer: org.springframework.kafka.support.serializer.JsonDeserializer  spring.kafka.consumer.properties.spring.json.trusted.packages=\*  spring.kafka.producer.value-serializer: org.springframework.kafka.support.serializer.JsonSerializer |

**Create payload**

|  |
| --- |
| public class User {  private Integer id;  private String firstName;  private String lastName;  //getter setter  } |

**Create JsonProducer**

|  |
| --- |
| import com.rlb.payload.User; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.kafka.core.KafkaTemplate; import org.springframework.kafka.support.KafkaHeaders; import org.springframework.messaging.Message; import org.springframework.messaging.support.MessageBuilder; import org.springframework.stereotype.Service;  @Service public class JsonProducer {  private static final Logger *LOGGER* = LoggerFactory.*getLogger*(JsonProducer.class);   KafkaTemplate<String, User> kafkaTemplate;   public void sendMsg(User data) {  *LOGGER*.info(String.*format*("Message Send -> %s ", data.toString()));  Message<User> message = MessageBuilder.*withPayload*(data).setHeader(KafkaHeaders.*TOPIC*, "rlbIn").build();  kafkaTemplate.send(message);  } } |

**Create RestAPI to send Message**

|  |
| --- |
| import com.rlb.kafka.JsonProducer; import com.rlb.payload.User; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.PostMapping; import org.springframework.web.bind.annotation.RequestBody; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RestController;  @RestController @RequestMapping("/kafka/v1") public class JsonController {  private static final Logger *LOGGER* = LoggerFactory.*getLogger*(JsonController.class);  private JsonProducer jsonProducer;   public JsonController(JsonProducer jsonProducer) {  this.jsonProducer = jsonProducer;  }   @PostMapping("/publish/json")  public ResponseEntity<String> publish(@RequestBody User user) {  *LOGGER*.info(String.*format*("Message Sent- > %s", user.toString()));  jsonProducer.sendMsg(user);  return new ResponseEntity<>("JSON Message sent", HttpStatus.*CREATED*);  } } |

**JSON Consumer**

|  |
| --- |
| import com.rlb.payload.User; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.kafka.annotation.KafkaListener; import org.springframework.stereotype.Service;  @Service public class JsonConsumer {  private static final Logger LOGGER = LoggerFactory.getLogger(JsonConsumer.class);   @KafkaListener(topics = "rlbIn", groupId = "myGroup")  public void jsonConsume(User data) {  LOGGER.info("JSON ");  LOGGER.info(String.format("JSON Message received : %s", data.toString()));  } } |