**Apache Kafka**

**Key Features:**

Producer: It refers to the entity responsible for publishing data to Kafka topics. Producers push records to Kafka topics for further processing by consumers.

Kafka Cluster: A Kafka cluster is a group of one or more Kafka brokers working together to manage and store the data. It provides fault tolerance, scalability, and reliability for Kafka data.

Kafka Broker or Kafka Server: A Kafka broker is a single instance of Kafka running in a cluster. It manages the storage, handling, and distribution of messages. Kafka brokers store the published data in topics and serve it to consumers.

Topic: A topic is a category or feed name to which records are published by producers. It's the fundamental abstraction in Kafka for organizing and storing data streams. Topics are partitioned and distributed across Kafka brokers for scalability and fault tolerance.

Consumers: Consumers read data from Kafka topics. They subscribe to one or more topics and process records published to those topics by producers. Consumers can be part of a consumer group, enabling parallel processing of messages within a group.

A screenshot of a computer

Description automatically generated

The topic will receive the message from producer. If the message is large, it is partitioned and the consumer will receive the message in the order of partionid only.

A diagram of a diagram of a diagram

Description automatically generated with medium confidence

**Steps for Apache Kafka Startup**

* Download Kafka with the below link.

<https://kafka.apache.org/downloads.html>

* Extract the folder.
* Paste it in other than C Drive
* Go to that directory D:\kafka\_2.12-3.7.0 and cmd
* .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties for starting zookeeper.
* .\bin\windows\kafka-server-start.bat .\config\server.properties for starting Apache server.
* Hit the API

Demo Project was done to illustrate the Apache Kafka with Springboot.

**Dependencies used for the project:**

<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>

**Application.yml:**

Any message which is sent by the producer to the particular topic should be serialized.(String to bytes)

spring:

kafka:

producer:

bootstrap-servers: localhost:9092

key-serializer: org.apache.kafka.common.serialization.StringSerializer

value-serializer: org.apache.kafka.common.serialization.StringSerializer

**KafkaProducer class:**

This is the class which produces the messages. It will send the messages through Kafka Template. The two parameters in Kafka Template indicates the topic name and the message. sendMessageToTopic() methods sends the message from Producer to the topic.

@Service

public class KafkaProducer {

//Two parameters indicates the name of the topic and message

@Autowired

private KafkaTemplate<String, String> kafkaTemplate;

public void sendMessageToTopic(String message) {

kafkaTemplate.send("neokred", message);

}

}

**KafkaConsumer Class:**

This is where the consumer class receives the messages. @KafkaListener(topics = "neokred", groupId = "neokred-group") indicates that the topic name with which it should receive the message and the consumer Id. If the consumer id is neokred-group-1 in the logs, then first consumer is receiving the message.

@Service

public class KafkaConsumer {

@KafkaListener(topics = "neokred", groupId = "neokred-group")

public void listenToNeokredTopic(String messageReceived) {

System.out.println("Message received is " + messageReceived);

}

}

**Controller Class:**

@RestController

@RequestMapping("/rest/api")

public class FetchMessageFromClient {

@Autowired

KafkaProducer kafkaProducer;

@GetMapping(value = "/producerMsg")

public String sendMessage(@RequestParam("message") String message)

{

kafkaProducer.sendMessageToTopic(message);

return "Message sent Successfully to neokred topic";

}

}