

APACHE KAFKA USING SPRING-BOOT

What is kafka ?

Kafka Is a Distributed message broker or publisher subscriber system.

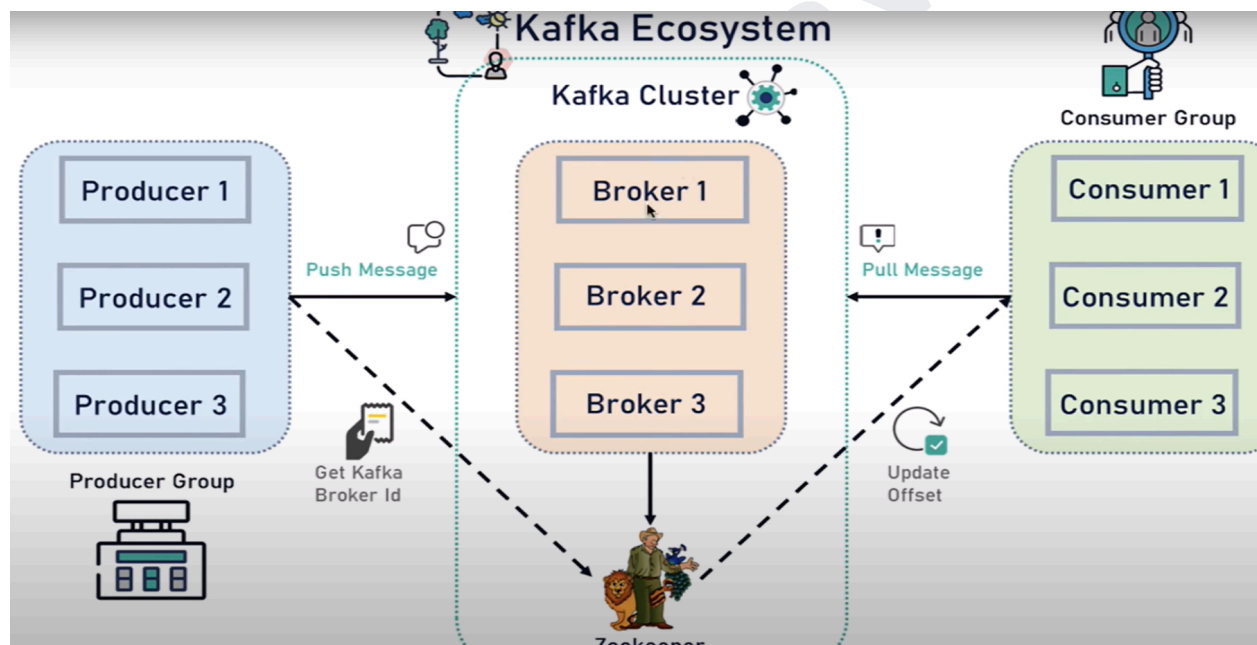
Kafka ecosystem :

Three Components

- 1) Publisher :who publish the data
- 2) Subscriber :who consumes a data
- 3) Broker : interface between both of them

Zookeeper : manage the state of kafka brokers

Publisher push the data and consumer pull the data they both read and communicate in the form of messages and the topics



Use Case:

- 1)messaging
- 2)website Activity Tracking
- 3)Metrics

Kafka Cluster :

Cluster consist of set of broker (minimum 3 brokers)

Kafka broker (kafka server)

The producer and consumer don't interact directly . they use kafka server as an agent or a broker to exchange messages

producer → kafka broker → consumer

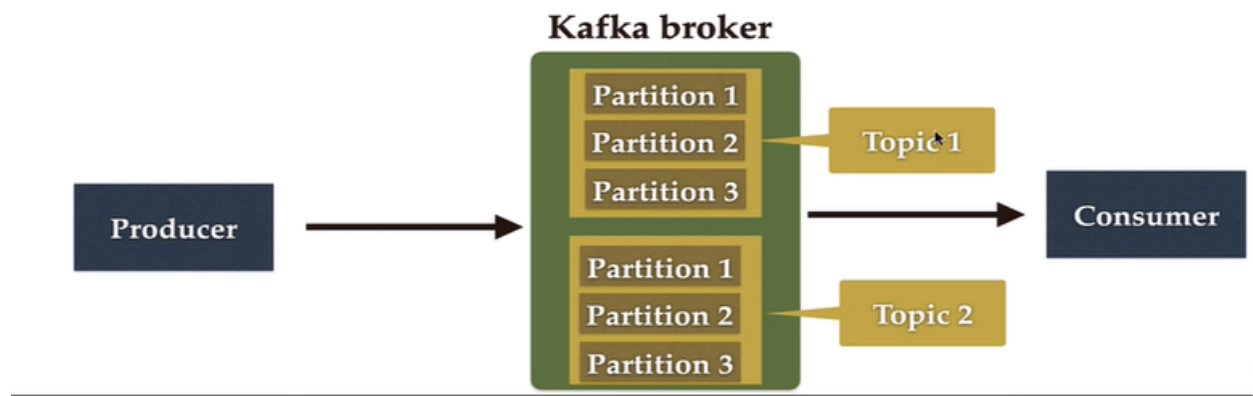
How is the data store in the broker ?

1) Kafka topic :

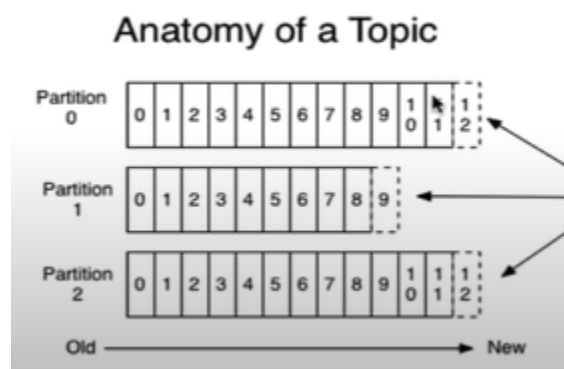
- It is like a table in a database or folder in a file system .
- it identified by a name
- can have any number of topic

Kafka Partitions

kafka topics divided into number of partitions which contain record in an unchangeable sequence



Offsets : sequence of id given to msg . Once the offset is assigned it will never be changed . the first message gets an offset zero



writes

Consumer Groups :

Contains one or more consumer working together to process the messages

INSTALL AND SETUP

1)apache kafka link : <https://kafka.apache.org/downloads>

2) after installation

<https://www.geeksforgeeks.org/how-to-install-and-run-apache-kafka-on-windows/>

Start the zookeeper :(using cmd)

C:\kafka_2.12-3.9.0\bin\windows>zookeeper-server-start.bat

C:\kafka_2.12-3.9.0\config\zookeeper.properties

Start

Create SPRING project using Spring IO :

Refer this website : <https://spring.io/projects/spring-kafka>

spring-boot application properties :

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

spring.kafka.consumer.group-id=myGroup

spring.kafka.consumer.auto-offset-reset : earliest

spring.kafka.consumer.key-deserializer :

org.apache.kafka.common.serialization.StringDeserializer

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

Controller :

@RestController

@RequestMapping("/kafka")

public class KafkaController {

 @Autowired

 private KafkaProducer producer;

```

    @PostMapping("/send")
    public ResponseEntity<String> send(@RequestBody String message) {
        producer.send("test-topic", message);
        return ResponseEntity.ok("Message sent!");
    }
}

```

Producer Service class :

```

@Service
public class KafkaProducer {
    @Autowired
    private KafkaTemplate<String, String> kafkaTemplate;

    public void send(String topic, String payload) {
        kafkaTemplate.send(topic, payload);
    }
}

```

Consumer Service Class

```

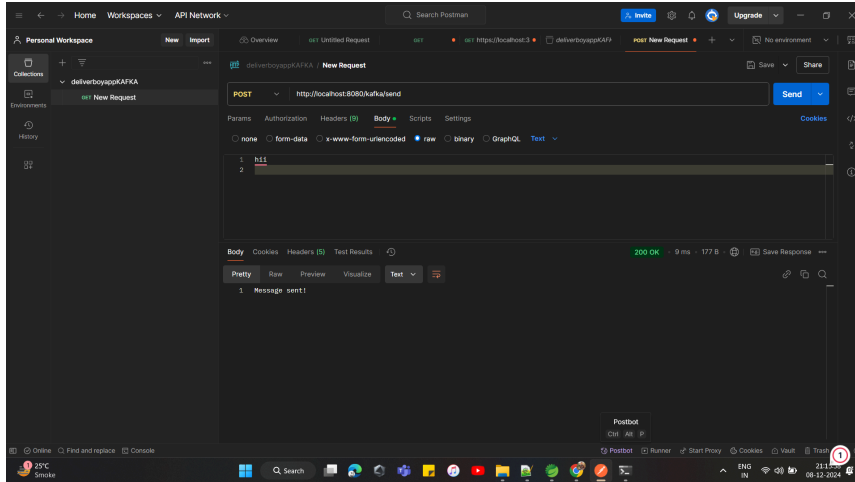
@Service
public class KafkaConsumer {

    private static Logger LOGEER = LoggerFactory.getLogger(KafkaConsumer.class);
    @KafkaListener(topics = "test-topic", groupId = "spring-kafka-poc-group")
    public void listen(String message) {
        LOGEER.info(String.format("message received - > %s", message));
    }
}

```

Testing :

1) Hit the API



2)Read the events in topic which we have created

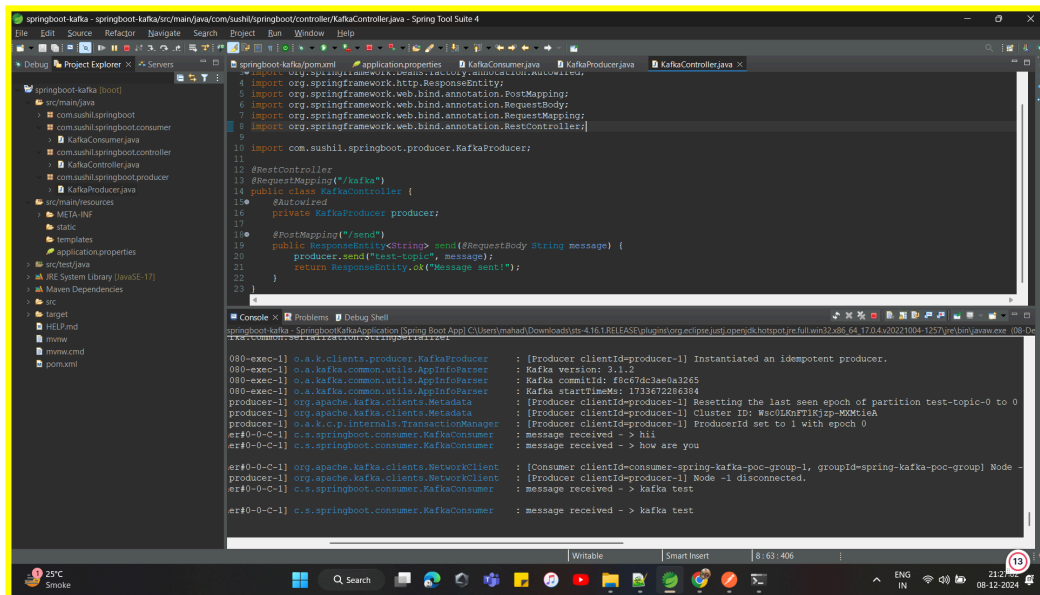
kafka-console-consumer.bat --topic <topic name> --from-beginning --bootstrap-server localhost:9092

Ex.

kafka-console-consumer.bat --topic test-topic --from-beginning --bootstrap-server localhost:9092

```
C:\kafka_2.12-3.9.0\bin\windows>kafka-console-consumer.bat --topic test-topic --from-beginning --bootstrap-server localhost:9092
hii
hii
hii
hii
hii
how are you
kafka test
```

3) check in console the msg is received :



The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project named 'springboot-kafka' with a package structure including 'com.sushil.springboot.controller'.
- Editor:** Displays the code for 'KafkaController.java'. It includes imports for Spring Framework annotations and a custom 'KafkaProducer' class. The controller has a REST endpoint for '/kafka' that sends a message to the producer.
- Console:** Shows the execution output. It includes logs for the KafkaProducer being instantiated, the message 'hi' being sent, and the message 'hi' being received by the consumer. It also shows a warning about a disconnected node.

CONFIGURE KAFKA PRODUCER AND CONSUMER FOR JSON SERIALIZER AND DESERIALIZER

Kafka stores and transport `byte[]`. There are a number of built-in serializers and deserializers but it doesn't include any for Json .

Spring Kafka simplifies this by offering `JsonSerializer` and `JsonDeserializer` to handle JSON data:

- **Sending JSON:** Use `JsonSerializer` to convert a Java object into JSON (`byte[]`) and send it to a Kafka topic.
- **Receiving JSON:** Use `JsonDeserializer` to convert JSON (`byte[]`) back into a Java object automatically.

Producer Service class for Json :

```
package com.sushil.springboot.kafka;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
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;

import com.sushil.springboot.jsondto.User;

@Service
public class JsonKafkaProducer {

    private static final Logger LOGGER =
        LoggerFactory.getLogger(JsonKafkaProducer.class);

    @Autowired
    private KafkaTemplate<String, User> kafkaTemplate;

    public void sendMessage(User data) {
        try {
            LOGGER.info(String.format("Message to be sent -> %s", data.toString()));

            Message<User> message = MessageBuilder
                .withPayload(data)
                .setHeader(KafkaHeaders.TOPIC, "test-topic")
                .build();

            kafkaTemplate.send(message);
            LOGGER.info("Message sent successfully to topic: sushil");
        } catch (Exception e) {
            LOGGER.error("Failed to send message to topic: sushil. Error: {}", e.getMessage(),
e);
        }
    }
}
```

The Consumer class remains the same as above Just add the following code:

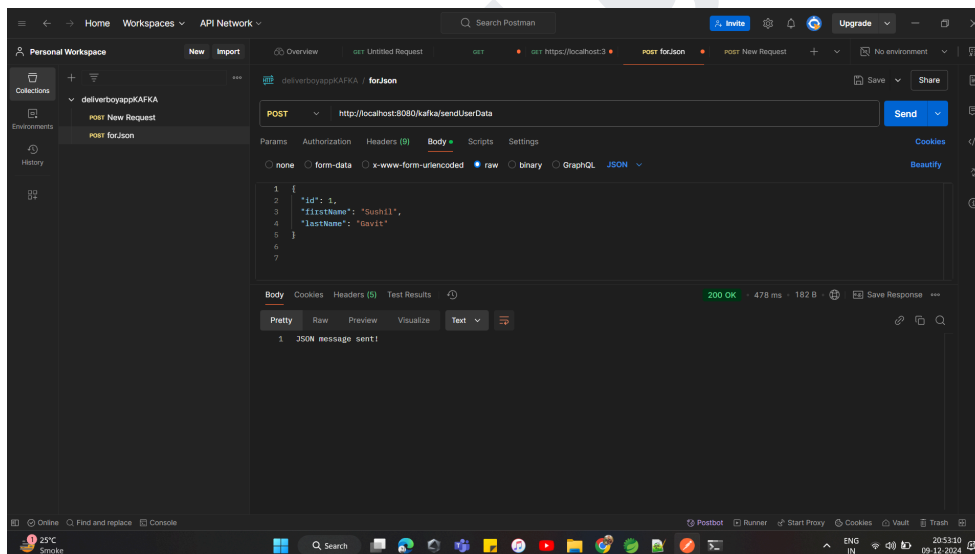
```
@KafkaListener(topics = "test-topic_json", groupId = "spring-kafka-poc-group")
public void listenJson(User user) {
    try {
        LOGGER.info(String.format("Message received -> %s",user));

    } catch (Exception e) {

        LOGGER.error("Error occurred while processing the message: {}", e.getMessage(),
e);
    }
}
```

Testing :

2) Hit the API

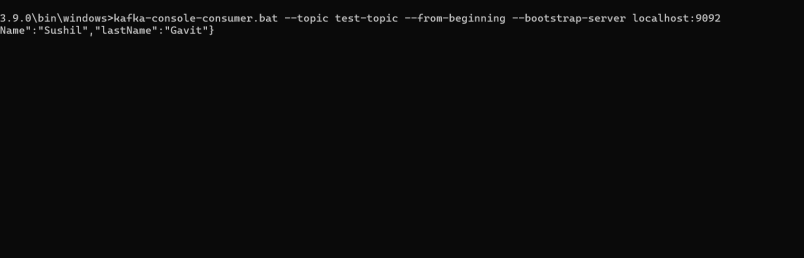


2)Read the events in topic which we have created

kafka-console-consumer.bat --topic <topic name> --from-beginning --bootstrap-server localhost:9092

Ex.


```
kafka-console-consumer.bat --topic test-topic --from-beginning --bootstrap-server localhost:90
```



The screenshot shows a Windows command prompt window with the following content:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.26100.2454]
(c) Microsoft Corporation. All rights reserved.

C:\kafka_2.12-3.9.0\bin\windows>kafka-console-consumer.bat --topic test-topic --from-beginning --bootstrap-server localhost:9092
{"id":1,"firstName":"Sushil","lastName":"Gavit"}
```

The taskbar at the bottom shows the system clock as 20:54:17 on 09-12-2024, and the language is set to ENG IN.

3) check in console the msg is received :

The screenshot displays an IDE with the `KafkaProducer.java` file open. The code defines a `NonKafkaProducer` class that implements `KafkaTemplate`. It includes a `sendMessage` method that constructs a `MessageUser` object and sends it to the `topic` using `send` and `flush` methods. The `main` method calls `sendMessage` with the topic `test-topic-1` and a user object.

The console output shows the successful execution of the application. It includes the following log entries:

```

[2024-12-09 21:54:17.337] INFO [ProducerClient] Producer client set to 1 with epoch 0
[2024-12-09 21:54:17.387] INFO [NonKafkaProducer] Message sent successfully to topic: sushil
[2024-12-09 21:54:17.387] INFO [NonKafkaProducer] Message sent successfully to topic: sushil
[2024-12-09 21:54:21.664] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-1, groupID=spring-kafka-poc-group-1, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.664] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-2, groupID=spring-kafka-poc-group-2, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.677] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-1, groupID=spring-kafka-poc-group-1, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.684] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-2, groupID=spring-kafka-poc-group-2, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.710] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-1, groupID=spring-kafka-poc-group-1, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.710] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-2, groupID=spring-kafka-poc-group-2, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.719] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-1, groupID=spring-kafka-poc-group-1, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.719] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-2, groupID=spring-kafka-poc-group-2, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.731] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-1, groupID=spring-kafka-poc-group-1, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.731] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-2, groupID=spring-kafka-poc-group-2, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.740] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-1, groupID=spring-kafka-poc-group-1, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.740] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-2, groupID=spring-kafka-poc-group-2, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.755] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-1, groupID=spring-kafka-poc-group-1, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.755] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-2, groupID=spring-kafka-poc-group-2, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.773] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-1, groupID=spring-kafka-poc-group-1, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.773] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-2, groupID=spring-kafka-poc-group-2, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.774] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-1, groupID=spring-kafka-poc-group-1, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.774] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-2, groupID=spring-kafka-poc-group-2, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.844] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-1, groupID=spring-kafka-poc-group-1, partitions assigned: [test-topic-1_0]
[2024-12-09 21:54:21.844] INFO [Consumer client] Consumer client=consumer-spring-kafka-poc-group-2, groupID=spring-kafka-poc-group-2, partitions assigned: [test-topic-1_0]

```

Connect :

lg : @sushil_gavit

github : <https://github.com/Sushilgavit>