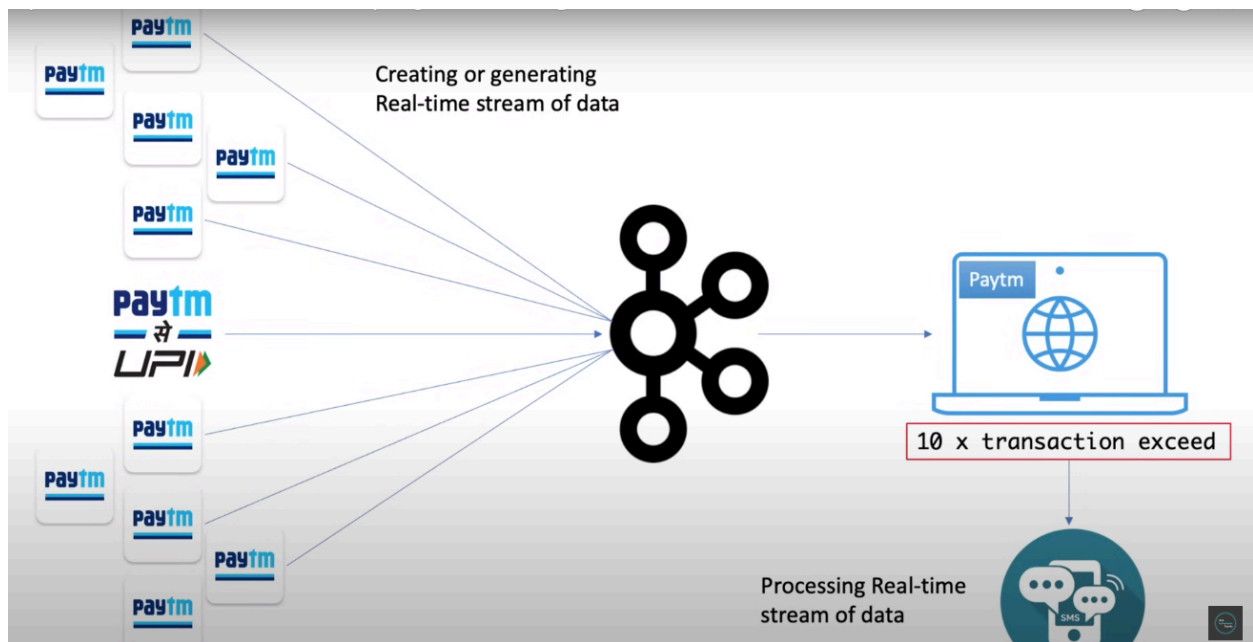


## What is Apache Kafka?

Apache Kafka is an open-source distributed event streaming platform.

1. Creating Real-time Stream
2. Processing Real-time Stream

### Creating Real-time Stream



Suppose we are purchasing a flight ticket by Paytm. Because Paytm provides multiple types of transactions. Like flight ticket booking. Movie ticket booking etc. When we will do any transaction then event will go to the kafka server but here I am not the only one Paytm user who is using the transaction there are millions of users and they are performing millions of transactions into milliseconds. **Kafka server receives millions or billions of events in each minute or each second or even in each millisecond. Sending the stream of continuous data from Paytm to Kafka. It's called Creating or generating Real-time stream of data.**

### Processing Real-time Stream

When the Kafka server received the data then needs to process the data for the operations. Like: Suppose Paytm wants to restrict 10x transaction to users per day. If they exceed the limit then Paytm wants to send a notification to the user. In such scenarios, Paytm needs to read the data from the Kafka server continuously for validating the user transaction limitation. This type of data processing is called **Processing Real-time stream of data.**

**What is the meaning of distributed?**

To the distributing of kafka server is called the distribution.

Suppose there are three kafka server running the different origin to perform event operations. In case if any server is goes down another server will come and pick-up the traffic to avoid the application downtime.

**Where does kafka come from?**

Kafka was originally developed at LinkedIn, and now it is open source sine 2011.

Now it is under of **Apache Kafka Foundation**.

## Why do need the Kafka?

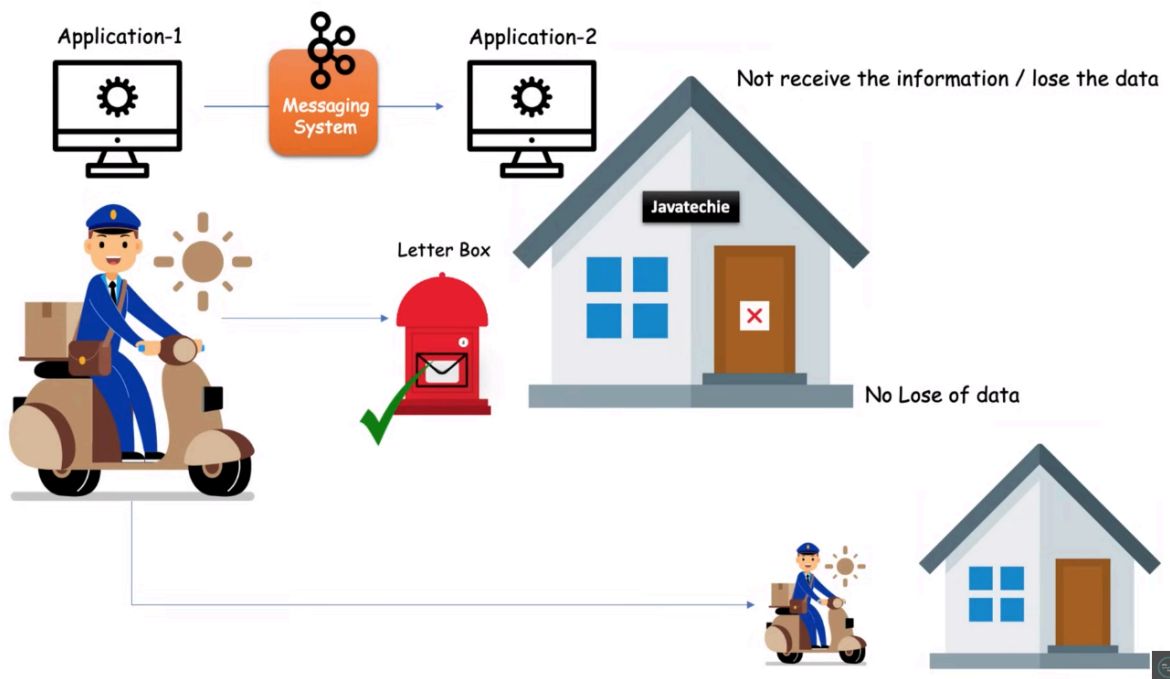
### Real life Example.

Suppose I have some parcel on my name and delivery man comes to my home for delivery but unfortunately I was not at my home and delivery man return from my home and they repeated same attempt three times and every time I was not at my home and finally delivery man return my parcel to main office. But I don't know who came to my home for parcel delivering may be in the parcel there are some important data. Like papers or money related paper but here I did lose the data because I was not available during the period when the postman came to my door. This could be a huge loss for me. So how we can overcome this issue.

If I have installed a letter box at my home so when delivery man will come and in my absence they put items into the letter box. Whenever I will be back then I can collect my order. In this case I will not lose the data. Here **Letter Box** acts as a middle layer between me and delivery man.

### Real time Example.

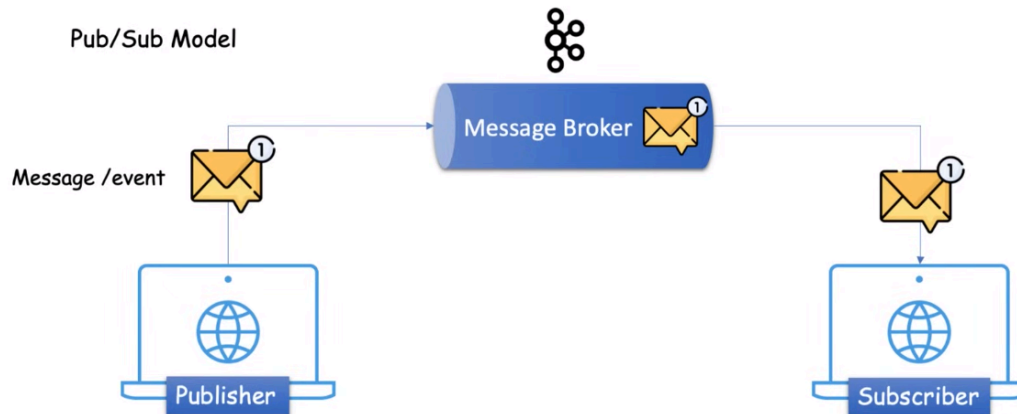
Suppose there are two applications and Application-1 wants to send some data to Application-2 but Application-2 was not available to take the data then Application-2 loses the data and which might impact business logic or future processing. So to overcome this issue, same as the letter box, we have to build a middle layer. In technical terms we can use a messaging system as a middle layer between two applications.



## How does kafka internally work?

It is specially works on Pub/Sub Model. (Publisher and subscriber model)

### How does it work (High-level overview)



## Kafka Architecture and Components.

Below core concept of Kafka.

1. Producer
2. Consumer
3. Broker
4. Cluster
5. Topic
6. Partitions
7. Offset
8. Consumer Groups
9. Zookeeper

### Producer.

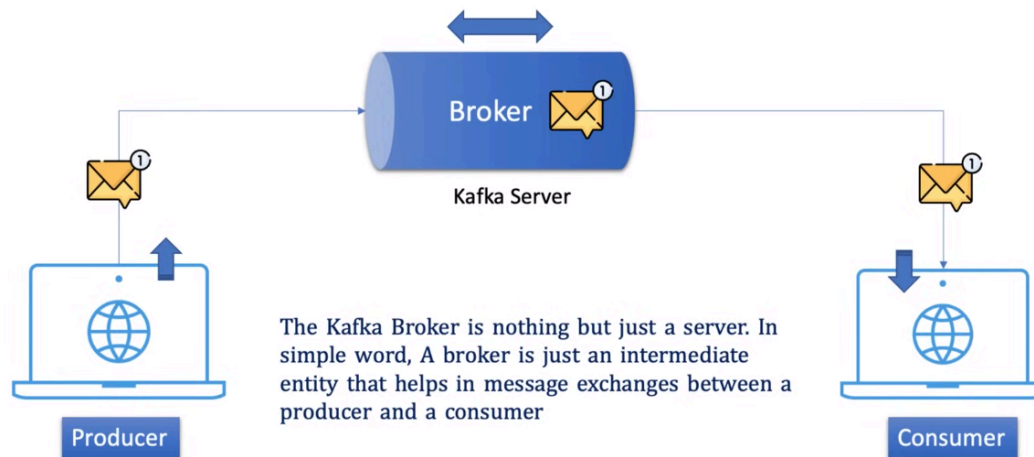
It is just responsible to produce the data to kafka server of Broker.

### Consumer.

It is just responsible to consume the data from kafka Broker.

### **Kafka Broker/Server.(Default port = 9092)**

It is just a server and works as intermediate between Producer and consumer to exchange the data



### **Cluster.**

Cluster is nothing. It is a common terminology in the distributed computing system. It is nothing just a group of computer or server that are working for a common purpose. A Kafka cluster consists of a set of brokers. A cluster has a minimum of 3 brokers.

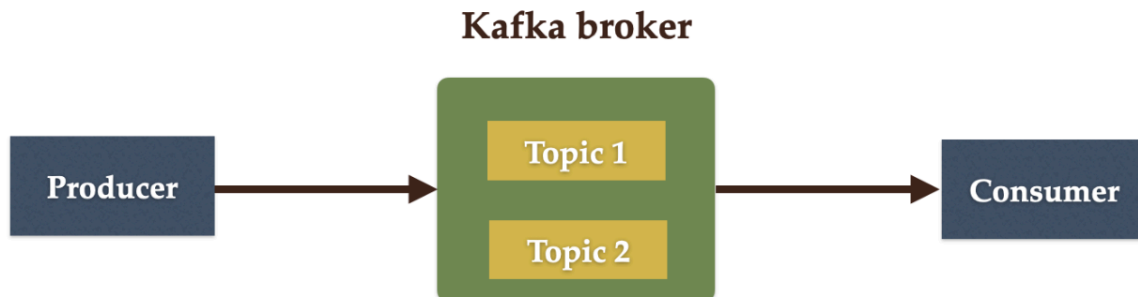
Suppose your producer publishing huge volume of the data so a single Kafka Broker may not handle the load then he might to need additional kafka server or broker. Then Kafka cluster adding group of brokers.

### Topic.

When the producer sends data to the Kafka broker. Then a consumer can ask for data from the Kafka broker. But the question is, Which data? We need to have some identification mechanism to request data from a broker. There comes the Kafka topic.

- The topic is like a table in a database or folder in a file system.
- The topic is identified by a name.
- You can have any number of topics.
- Topic acts like database tables.

The following diagram shows two Topics are created in a Kafka broker:



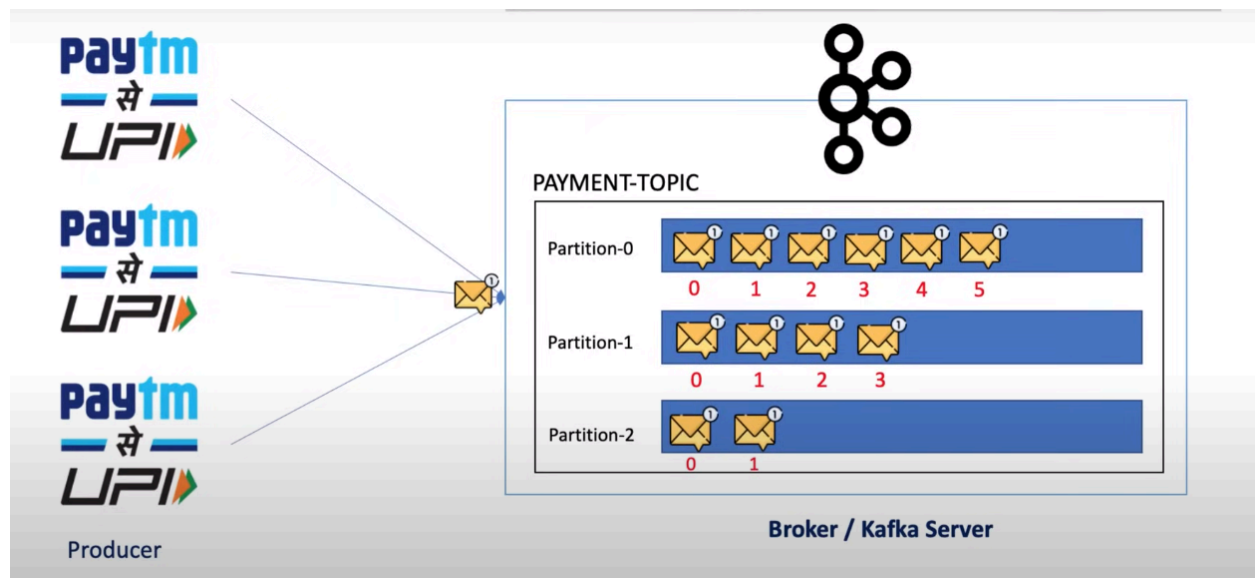
### Partition.

Partitioning in Kafka is the process of dividing a topic into multiple partitions, where each partition is an ordered, immutable sequence of records. Partitioning serves several purposes:

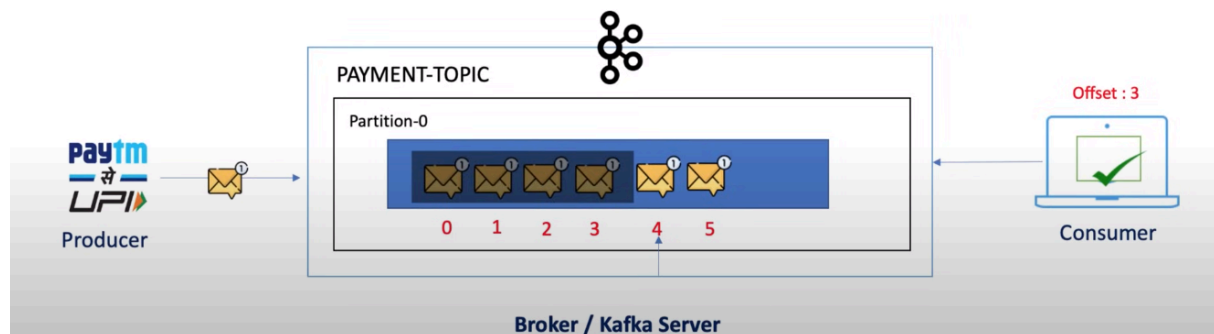
1. **Scalability:** Multiple consumers can read from different partitions in parallel, allowing Kafka to handle a high throughput of data.
2. **Distributed Processing:** Partitioning enables distributed processing by distributing the data across multiple brokers.

## Offset.

The data which is stored into Partition and a position or id is assigned that id or number or indexes is called offset.



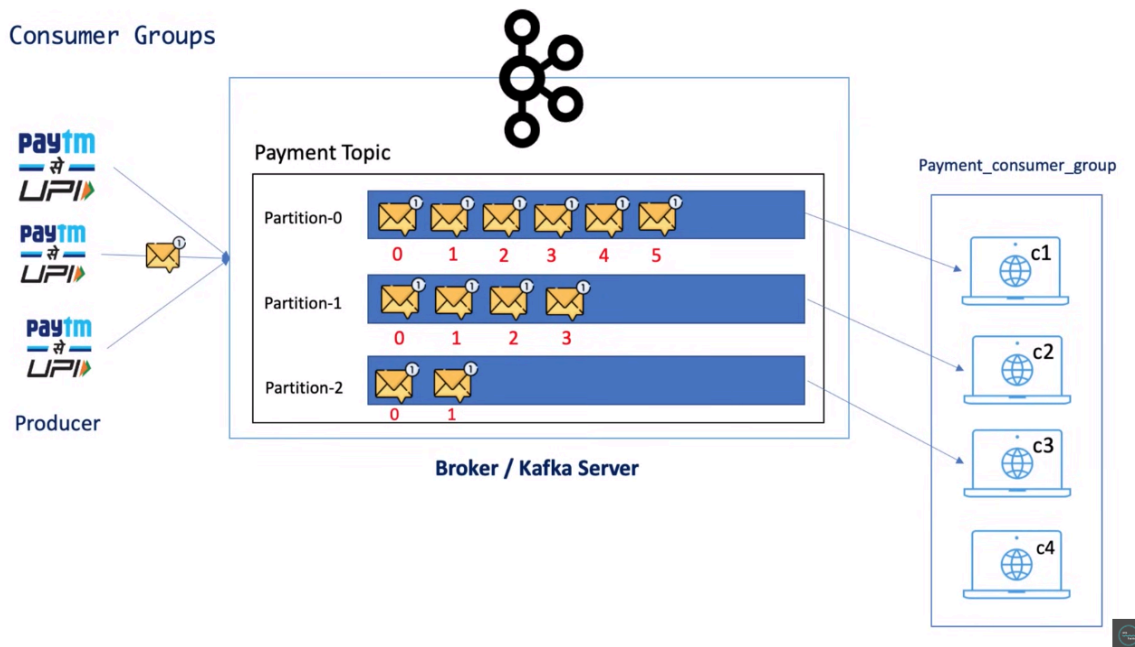
1. Offset is used for tracking like which message is consumed by consumer.
2. Suppose we have 5 offset in kafka partition and consumer consume all 3 offset data but after that consumer is down. Then consumer will back to the online so here offset value will help to the consumer exactly where consumer has to start consuming the messages.



## Consumer Group.

As below image suppose we have three partitions if we have one consumer so here performance is not fast because one consumer will read the data from all partitions but if we make a consumer group and make three consumer this time performance is fast because randomly each consumer consume the data from each partition and we do not have control that consumer can read the data from partition 1 because it is decided by coordinator.

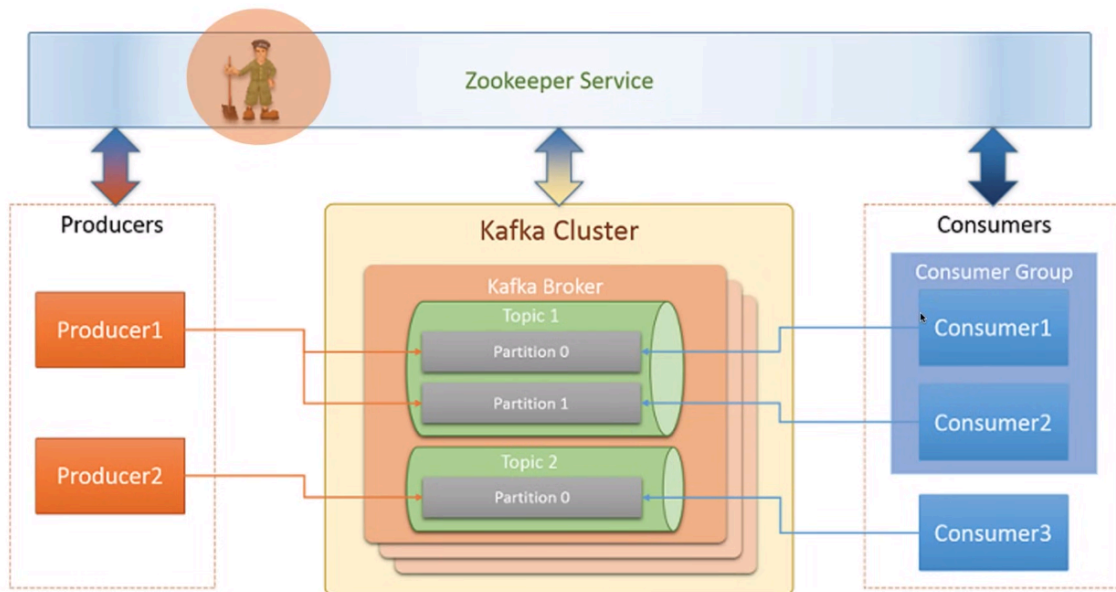
In Below image we have 4 consumer so what is the behaviour of this consumer group 4? The consumer 4 will sit on bench and there is no work for him but if any consumer instance is rejected or goes off then will get chance of the consumer 4 to connect any that is called the consumer rebalancing.





### Zookeeper (Default port = 2181)

Zookeeper acts as manager for tracking the status of Kafka cluster nodes. It is also track of the kafka topics, partitions, offset, etc.



# Kafka Installation

Open Source : Apache Kafka

Commercial distribution : Confluent Kafka

Managed Kafka service : confluent & AWS

# 1. Install and Setup Apache Kafka

1. Download Kafka from the official website at <https://kafka.apache.org/downloads>

->get satart -> quickstart -> download

2. Extract Kafka zip in the local file system by 7-zip application

3. Start Zookeeper service.

Note have to install it inside the extract Kafka file

Got to the below directive and hit enter

Start Zookeeper = `zookeeper-server-start.bat`

`..\..\config\zookeeper.properties`

```
D:\Application\kafka\bin\windows>zookeeper-server-start.bat ..\..\config\zookeeper.properties
[2024-07-13 14:19:13,038] INFO Reading configuration from: ..\..\config\zookeeper.properties (org.apa
[2024-07-13 14:19:13,042] WARN ..\..\config\zookeeper.properties is relative. Prepend .\ to indicate
```

4. Start Kafka Server/Broker

Start Kafka Server = `kafka-server-start.bat`

`..\..\config\server.properties`

```
D:\Application\kafka\bin\windows>kafka-server-start.bat ..\..\config\server.properties
[2024-07-13 14:24:50,337] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2024-07-13 14:24:50,743] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zoo
9Util)
[2024-07-13 14:24:50,898] INFO starting (kafka.server.KafkaServer)
[2024-07-13 14:24:50,898] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2024-07-13 14:24:50,920] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2024-07-13 14:24:50,935] INFO Client environment:zookeeper.version=3.6.3--6401e4ad2087061bc6b9f80d8c2d69f2e3c8660a, built on 04/08/2021 16:35 GMT (org
```

5. How to create a Topic

```
D:\Application\kafka\bin\windows>kafka-topics.bat --create --topic my-topic --bootstrap-server localhost:9092 --replication-factor 1 --partitions 3
Created topic my-topic.
D:\Application\kafka\bin\windows>
```

Creating Kafka Topic = `kafka-topics.bat --create --topic my-topic`  
`--bootstrap-server localhost:9092 --replication-factor 1`  
`--partitions 3`

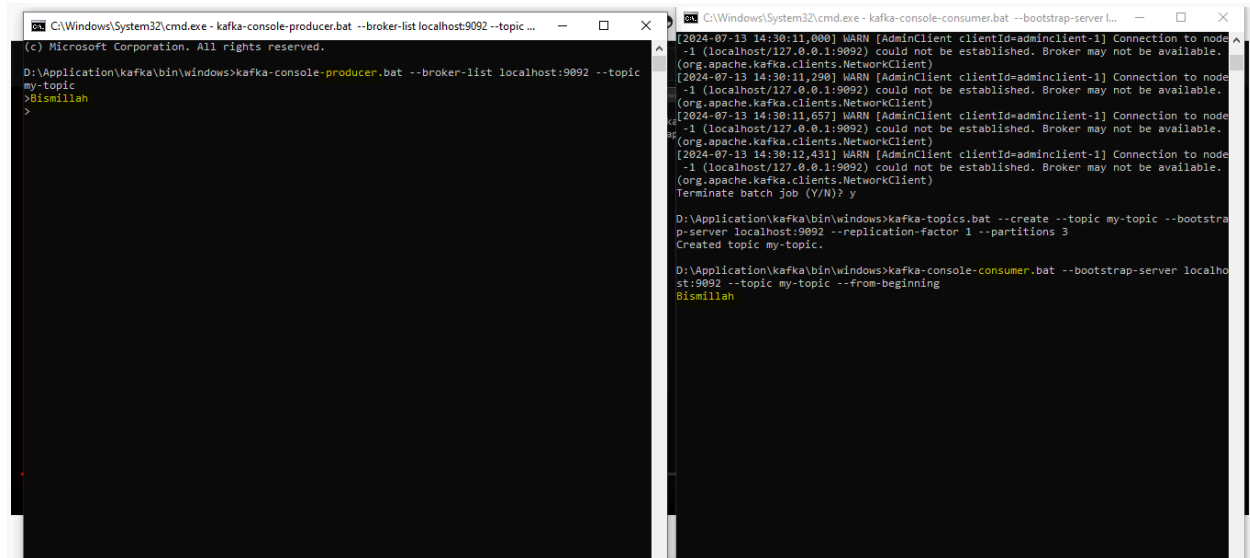
## 6. How to write some event or produce the topic

Creating Kafka Producer = `kafka-console-producer.bat --broker-list localhost:9092 --topic my-topic`

```
D:\Application\kafka\bin\windows>kafka-console-producer.bat --broker-list localhost:9092 --topic my-topic
```

## 7. How to read the event or producer data

Creating kafka consumer = `kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic my-topic --from-beginning`



The image shows two terminal windows side-by-side. The left window is titled 'C:\Windows\System32\cmd.exe - kafka-console-producer.bat --broker-list localhost:9092 --topic my-topic' and shows the command being executed. The right window is titled 'C:\Windows\System32\cmd.exe - kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic my-topic --from-beginning' and shows the command being executed along with several warning messages from the Kafka client about connection failures to the broker at localhost:9092.

```
C:\Windows\System32\cmd.exe - kafka-console-producer.bat --broker-list localhost:9092 --topic my-topic
(c) Microsoft Corporation. All rights reserved.
D:\Application\kafka\bin\windows>kafka-console-producer.bat --broker-list localhost:9092 --topic my-topic
>Bismillah
>
```

```
C:\Windows\System32\cmd.exe - kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic my-topic --from-beginning
[2024-07-13 14:30:11,000] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Broker may not be available.
(org.apache.kafka.clients.NetworkClient)
[2024-07-13 14:30:11,290] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Broker may not be available.
(org.apache.kafka.clients.NetworkClient)
[2024-07-13 14:30:11,657] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Broker may not be available.
(org.apache.kafka.clients.NetworkClient)
[2024-07-13 14:30:12,431] WARN [AdminClient clientId=adminclient-1] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Broker may not be available.
(org.apache.kafka.clients.NetworkClient)
Terminate batch job (Y/N)? y

D:\Application\kafka\bin\windows>kafka-topics.bat --create --topic my-topic --bootstrap-server localhost:9092 --replication-factor 1 --partitions 3
Created topic my-topic.

D:\Application\kafka\bin\windows>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic my-topic --from-beginning
Bismillah
```

```
D:\Application\kafka\bin\windows>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic my-topic --from-beginning
```

# Installing confluent Kafka.

1. Open confluent.io
2. Fill the form and submit but be care full do not choose **cloud**

study Translate study Mindcraft pune college form fi... sak soft vmedulife Software Forgot password in... Microservice Accou...


Unlock Real-time Decisions with Cloud-Native Microservices | [Join the Webinar](#) → with Michelin [Login](#) [Contact Us](#)

CONFLUENT WHY CONFLUENT PRODUCTS PRICING SOLUTIONS LEARN DEVELOPERS [GET STARTED FREE](#) 🔍 🌐

## Community

The open-source and community features of Confluent Platform can be installed manually by downloading zip, tar or docker archives. Use it for free forever.

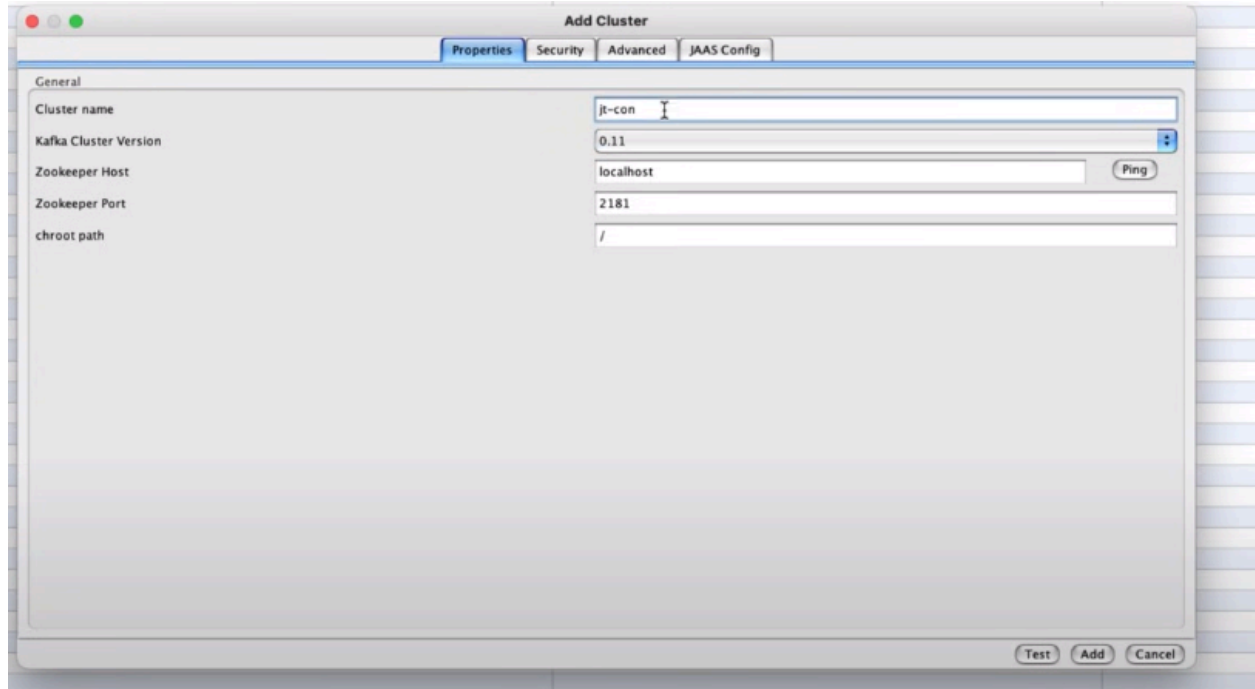
ZIP	<a href="#">Quick Start</a> <a href="#">Installation Guide</a>	<a href="#">DOWNLOAD</a>
TAR	<a href="#">Quick Start</a> <a href="#">Installation Guide</a>	<a href="#">DOWNLOAD</a>
Docker	<a href="#">Installation Guide</a> <a href="#">Browse Images</a>	<a href="#">QUICK START</a>



# Installing Kafka Offset Explorer.

This is use for monitoring kafka messaging system.

Below image how to add new connection in Kafka Offset explorer.

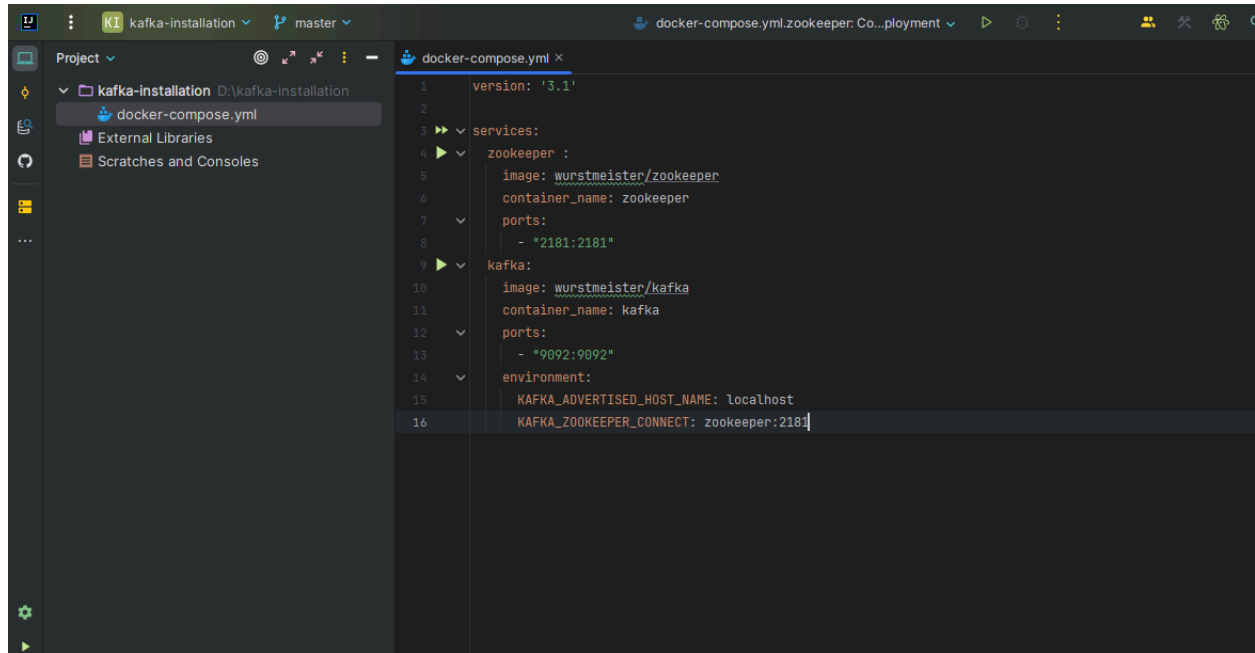


# Installing Kafka on Docker.

For installing Kafka on docker container We need two things.

1. Instance of zookeeper.
2. Instance of Kafka

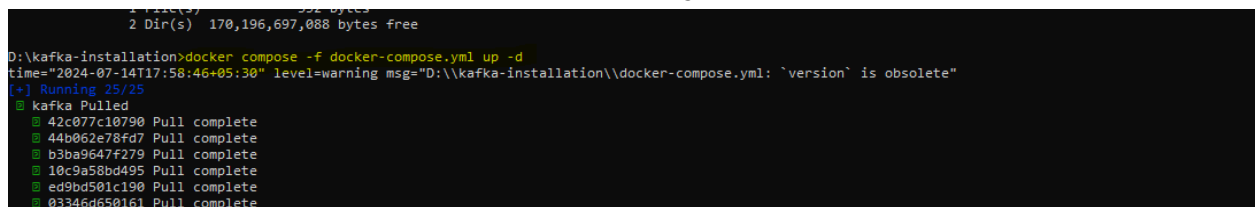
Create normal project as below image and follow docker-compose.yml



Below image to pull the docker image by commands

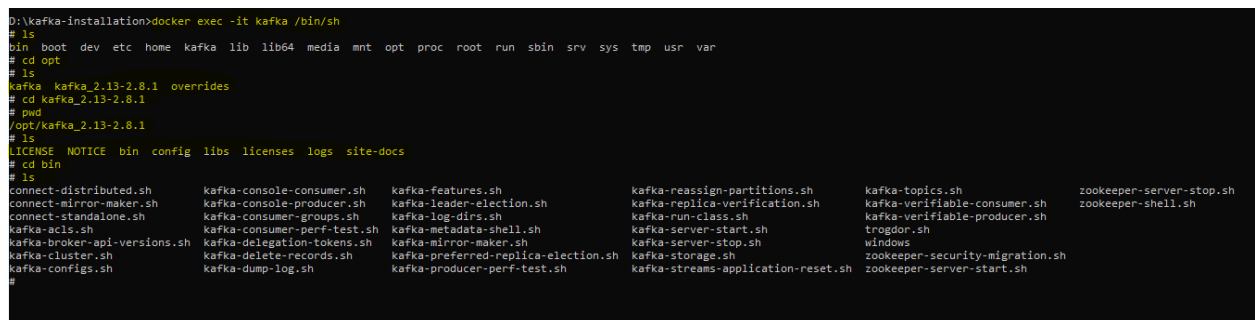
Note:= Docker desktop should be open.

Command = **docker compose -f docker-compose.yml up -d**



To ensure kafka is installed or not follow below steps.

1. **docker exec -it kafka /bin/sh**

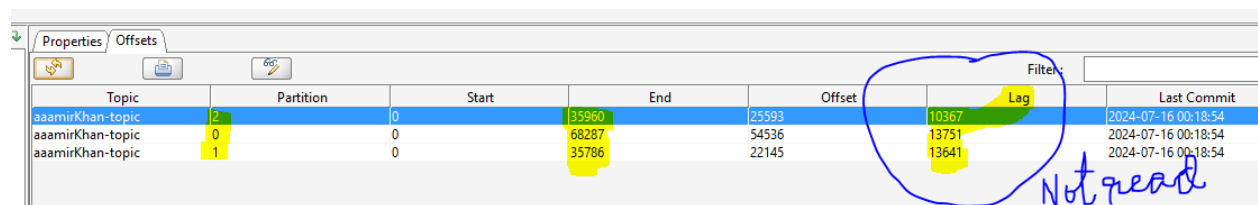


Command = **kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 --partitions 1 --topic test-topic**

```
#  
# kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 --partitions 1 --topic test-topic  
Created topic test-topic.  
# pwd  
/opt/kafka_2.13-2.8.1  
#
```

### What is the lag in apache kafka?

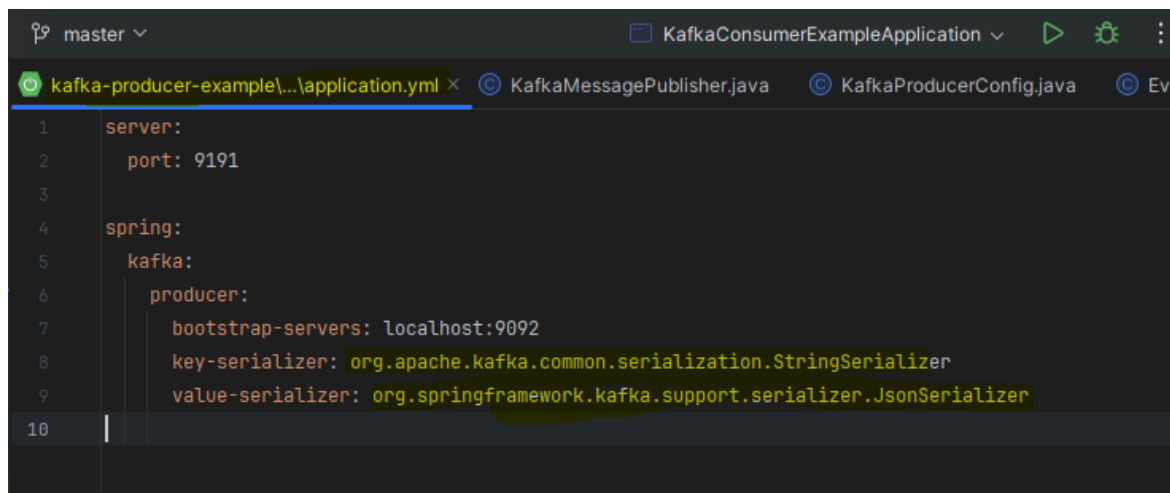
Suppose we produced 30k data and stored into a partition but we did not consume all data from the any of the partition that is called the lag.



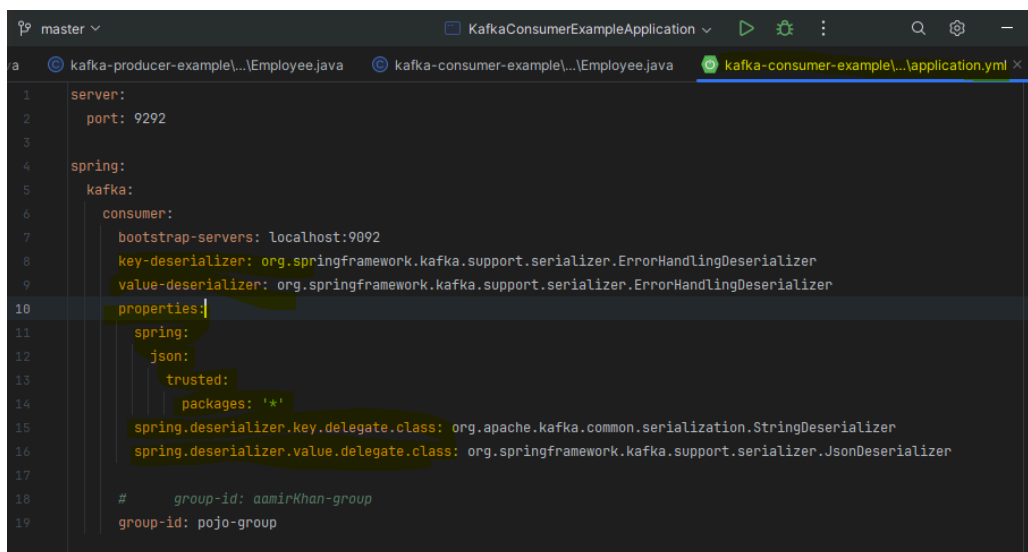
Topic	Partition	Start	End	Offset	Lag	Last Commit
aaamirKhan-topic	2	0	35960	25593	10367	2024-07-16 00:18:54
aaamirKhan-topic	0	0	68287	54536	13751	2024-07-16 00:18:54
aaamirKhan-topic	1	0	35786	22145	13641	2024-07-16 00:18:54

## Kafka Serialization & Deserialization?

- In Kafka if we send the data into string then do not need to use manually serialization or deserialization.
- In Kafka suppose we are producing the a data which is a Customer pojo class then we have to use serialization while producing and using deserialization while consuming the messages.
- The data always should be serialized while producing and consumer using the deserialization concept while consuming the messages.
- Producer produces the message or serialized the message over the network/topic into byte of array format.



```
1  server:
2    port: 9191
3
4  spring:
5    kafka:
6      producer:
7        bootstrap-servers: localhost:9092
8        key-serializer: org.apache.kafka.common.serialization.StringSerializer
9        value-serializer: org.springframework.kafka.support.serializer.JsonSerializer
10
```



```
1  server:
2    port: 9292
3
4  spring:
5    kafka:
6      consumer:
7        bootstrap-servers: localhost:9092
8        key-deserializer: org.springframework.kafka.support.serializer.ErrorHandlingDeserializer
9        value-deserializer: org.springframework.kafka.support.serializer.ErrorHandlingDeserializer
10       properties:
11         spring:
12           json:
13             trusted:
14               packages: '*'
15           spring.deserializer.key.delegate.class: org.apache.kafka.common.serialization.StringDeserializer
16           spring.deserializer.value.delegate.class: org.springframework.kafka.support.serializer.JsonDeserializer
17
18       # group-id: aamirKhan-group
19       group-id: pojo-group
```



```

properties:
  spring:
    json:
      trusted:
        packages : com.javatechie.dto

```

Instead of Above Yml configuration we can do Java base configuration.

**Note : Need to comment yml config.**

Below for producer.

```

package com.aamir.config;

import org.apache.kafka.clients.admin.NewTopic;
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 java.util.HashMap;
import java.util.Map;

@Configuration
public class KafkaProducerConfig {

    @Bean
    public NewTopic createTopic(){
        return new
            NewTopic("aaamirKhan-topic",3, (short) 1);
    }

    @Bean
    public NewTopic pojoTopic(){
        return new
            NewTopic("pojo-topic",3, (short) 1);
    }

    @Bean
    public Map<String,Object> producerConfiguration(){
        Map<String,Object> props = new HashMap<>();
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,"localhost:9092");
        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<>(producerConfiguration());
}

@Bean
public KafkaTemplate<String, Object> kafkaTemplate() {
    return new KafkaTemplate<>(producerFactory());
}
}

```

**Below for the consumer.**

```

package com.aamir.config;

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.config.ConcurrentKafkaListenerContainerFactory;
import org.springframework.kafka.config.KafkaListenerContainerFactory;
import org.springframework.kafka.core.ConsumerFactory;
import org.springframework.kafka.core.DefaultKafkaConsumerFactory;
import org.springframework.kafka.listener.ConcurrentMessageListenerContainer;
import org.springframework.kafka.support.serializer.JsonDeserializer;

import java.util.HashMap;
import java.util.Map;

@Configuration
public class KafkaConsumerConfig {

    @Bean
    public Map<String, Object> consumerConfiguration() {
        Map<String, Object> props = new HashMap<>();
        props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
        props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class);
        props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
JsonDeserializer.class);
        props.put(JsonDeserializer.TRUSTED_PACKAGES, "*"); // for all packages
//        props.put(JsonDeserializer.TRUSTED_PACKAGES, "com.aamir.entity"); // for
specific packages
        return props;
    }

    @Bean
    public ConsumerFactory<String, Object> consumerFactory() {
        return new DefaultKafkaConsumerFactory<>(consumerConfiguration());
    }

    @Bean

```

```

    public KafkaListenerContainerFactory<ConcurrentMessageListenerContainer<String,
Object>> kafkaListenerContainerFactory() {
        ConcurrentKafkaListenerContainerFactory<String, Object> factory = new
ConcurrentKafkaListenerContainerFactory<>();
        factory.setConsumerFactory(consumerFactory());
        return factory;
    }
}

```

Understand message routing with specific Partition in Kafka.

In Producer we can give the partitions. See below image

```

@Autowired
private KafkaTemplate<String, Object> template;

public void sendMessageToTopic(String message){
    CompletableFuture<SendResult<String, Object>> future = template.send(topic: "javatechie-topic", partition: 3, key: null, message);
    future.whenComplete((result, ex)->{
        if (ex == null) {
            System.out.println("Sent message=[" + message +
                "] with offset=[" + result.getRecordMetadata().offset() + "]");
        } else {
            System.out.println("Unable to send message=[" +
                message + "] due to : " + ex.getMessage());
        }
    });
}
}

```

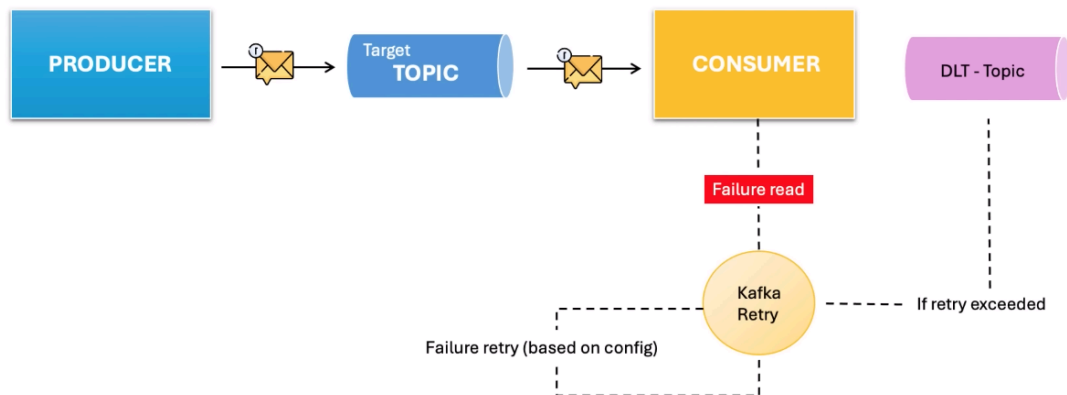
In Consumer we can read the data from the particular partitions.

```

@KafkaListener(topics = "javatechie-topic1", groupId = "jt-group",
    topicPartitions = {@TopicPartition(topic = "javatechie-topic1", partitions = {"2"})})
public void consumeEvents(String customer) {
    log.info("consumer consume the events {} ", customer.toString());
}
}

```

## Kafka Error Handling, Retry and Dead Letter topics.



If a Producer produces a message and consumer is not available for consuming those message may be some issue like db connection, or any validation Exception so we have to use Kafka retry mechanism based on configuration like 4 times or 5 times to consume those message who is not consumed while consumer had issue.

For this issue we can use `@RetryTopic` by default the value will be 3 but it will retry only 2 time bcz onf N-1 and after a particular retry if issue is exist then those topic will go mark as DLT (Dead Letter Topic) by using `@DltHandler`

If we `@RetryTopic(attempt = "3")` internally 2 topic created for retrying. See below images. Where Highlighted topic is main topic. Instead of main topic all extra created topic will be only one partition.

```
kafka-topics --zookeeper zk1 --list
```

- kafka-error-handling-demo
- kafka-error-handling-demo-dlt
- kafka-error-handling-demo-retry