**Kafka**

**Vedio link :** [**https://www.youtube.com/watch?v=clouToNoxGM&list=PLVz2XdJiJQxwoGuQb8lR-sTV26dz1SoXo**](https://www.youtube.com/watch?v=clouToNoxGM&list=PLVz2XdJiJQxwoGuQb8lR-sTV26dz1SoXo)

[**https://github.com/Java-Techie-jt/kafka-publisher**](https://github.com/Java-Techie-jt/kafka-publisher)

**The Saga design pattern is a way to manage data consistency across microservices in distributed transaction scenarios. A saga is a sequence of transactions that updates each service and publishes a message or event to trigger the next transaction step.**

**Kafka download Link**

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

https://www.tutorialspoint.com/apache\_kafka/apache\_kafka\_basic\_operations.htm

Start zookeeper.start

D:\Software\java setup\kafka\_2.12-1.1.0\bin\windows>zookeeper-server-start.bat D:\Software\java setup\kafka\_2.12-1.1.0\config\zookeeper.properties

Start Kafka server

D:\Software\java setup\kafka\_2.12-1.1.0\bin\windows>kafka-server-start.bat

D:\Software\java setup\kafka\_2.12-1.1.0\config>server.properties

**Create topics:**

D:\Software\java setup\kafka\_2.12-1.1.0\bin\windows>kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 –partitions 1 -topic <topic\_name>

**Create topic syntax:**

--create --zookeeper localhost:2181 --replication-factor 1 –partitions 1 -topic <topic\_name>

**Test:**

**Produce Message in kafka:**

D:\Software\java setup\kafka\_2.12-1.1.0\bin\windows> kafka-console-producer.bat --broker-list localhost:9092 -topic <topic\_name>

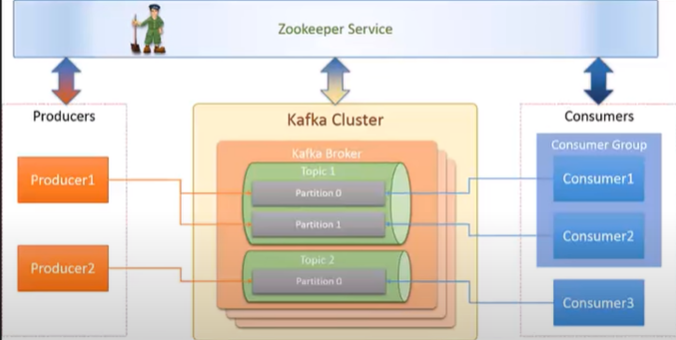
**Consume a message in Kafka:**

D:\Software\java setup\kafka\_2.12-1.1.0\bin\windows> kafka-console-consumer.bat --zookeeper localhost:2181 -topic <topic\_name>

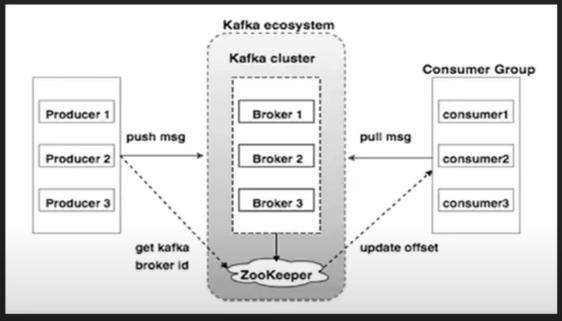
**New for consumer:**

D:\Software\java setup\kafka\_2.12-1.1.0\bin\windows>kafka-console-consumer.bat -bootstrap-server localhost:9092 --topic <topic\_name>

Note: Bootstrap server is the Kafka server, having port number=9092.



**Replication: It is nothing shadow instance of your streaming data.**

****

What is Apache Kafka? https://www.javatpoint.com/apache-kafka

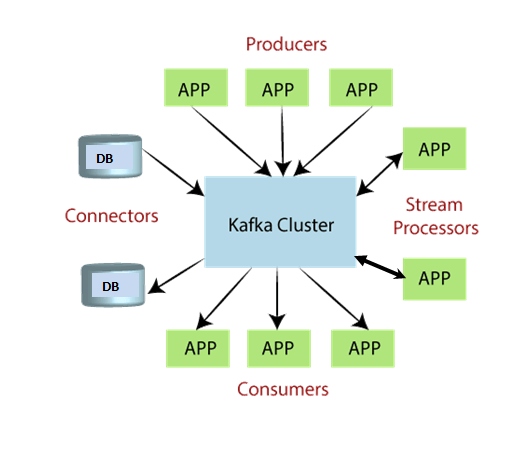
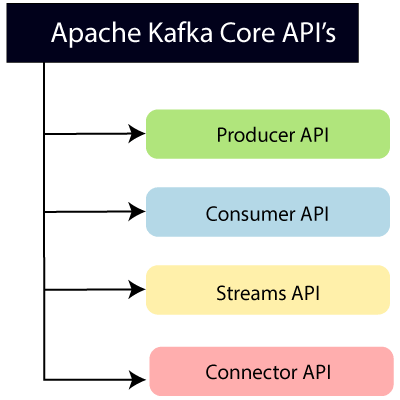
Apache Kafka is an open-source stream-processing software platform that is used to handle real-time data storage. It works as a broker between two parties, i.e., a sender and a receiver. It can handle trillions of data events in a day.

Apache Kafka is a software platform that is based on a distributed streaming process. It is a publish-subscribe messaging system that lets exchanging of data between applications, servers, and processors as well. Apache Kafka was originally developed by **LinkedIn**, and later it was donated to the Apache Software Foundation. Currently, it is maintained by **Confluent** under Apache Software Foundation. Apache Kafka has resolved the lethargic trouble of data communication between a sender and a receiver.

## What is the Streaming process?

A streaming process is the processing of data in parallelly connected systems. This process allows different applications to limit the parallel execution of the data, where one record executes without waiting for the output of the previous record. Therefore, a distributed streaming platform enables the user to simplify the task of the streaming process and parallel execution. Therefore, a streaming platform in Kafka has the following key capabilities:

* As soon as the streams of records occur, it processes it.
* It works similar to an enterprise messaging system where it publishes and subscribes streams of records.
* It stores the streams of records in a fault-tolerant durable way.

 ****

**Producer API:** This API allows/permits an application to publish streams of records to one or more topics.

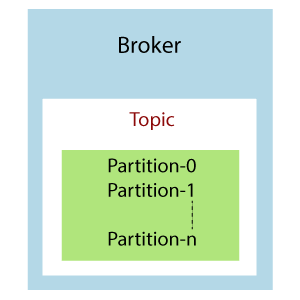
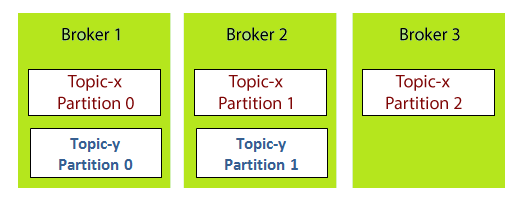
**Consumer API:** This API allows an application to subscribe to one or more topics and process the stream of records produced to them.

**Streams API:** This API allows an application to effectively transform the input streams to output streams. It permits an application to act as a stream processor which consumes an input stream from one or more topics and produces an output stream to one or more output topics.

**Connector API:** This API executes the reusable producer and consumer APIs with the existing data systems or applications.

## Why Apache Kafka

1. Apache Kafka is capable of handling millions of data or messages per second.
2. Apache Kafka works as a mediator between the source system and the target system. Thus, the source system (producer) data is sent to Apache Kafka, where it decouples the data, and the target system (consumer) consumes the data from Kafka.
3. Apache Kafka is having extremely high performance, i.e., it has a really low latency value of less than 10ms which proves it as a well-versed software.
4. Apache Kafka has a resilient architecture that has resolved unusual complications in data sharing.
5. Organizations such as NETFLIX, UBER, Walmart, etc., and over thousands of such firms make use of Apache Kafka.
6. Apache Kafka is able to maintain the fault-tolerance. Fault tolerance means that sometimes a consumer successfully consumes the message that was delivered by the producer. But, the consumer fails to process the message back due to backend database failure, or due to the presence of a bug in the consumer code. In such a situation, the consumer is unable to consume the message again. Consequently, Apache Kafka has resolved the problem by reprocessing the data.
7. Learning Kafka is a good source of income. So, those who wish to raise their income in the future in the IT sector can learn.

Each broker is holding a topic, namely Topic-x with three partitions 0,1 and 2. Remember, all partitions do not belong to one broker only, it is always distributed among each broker (depends on the quantity). Broker 1 and Broker 2 contains another topic-y having two partitions 0 and 1. Thus, Broker 3 does not hold any data from Topic-y. It is also concluded that no relationship ever exists between the broker number and the partition number.

# **Kafka Topic Replication**