**What is an Event Stream?**

An Event Stream is an ordered sequence of events representing important actions in a software domain. This can be something simple, like clicking on a link, or it might be something more complex, like transferring funds between two banks.

Event Stream Processing (ESP) takes a continuous stream of [events](https://developer.confluent.io/patterns/event-stream/event-streaming-platform/) and processes them as soon as a change happens. By processing single points of data rather than an entire batch, [event streaming platforms](https://developer.confluent.io/patterns/event-stream/event-streaming-platform/) provide an architecture that enable software to understand, react to, and operate as events occur.

**How Event Streaming Works**

In software, any significant action can be recorded as an event.

For example, it could be as simple as someone clicking a link or viewing a webpage, or something more involved like paying for an order, withdrawing money, or even communicating with numerous, distributed IoT devices at once.

These events can be organized into streams, essentially a series of events ordered by time. From there, events can be shared with other systems where they can be processed in real-time. Events are pushed and handled one at a time, as they happen. This allows the system to react in real-time, rather than waiting for batches to accumulate.

For example, each time someone clicks a link or views a webpage, we might push an event into a system such as [Apache Kafka](https://www.confluent.io/what-is-apache-kafka/?session_ref=https://www.google.com/). Downstream, a Flink job could consume those events to develop analytics about how many views and clicks our website is receiving.

**Benefits of Event Streaming**

The most obvious benefit of event streaming is that it allows systems to react in real time. For example, multi-player video games, a stock transaction, or personalized shopping recommendations. This gives users the kind of instant gratification that they crave while better reflecting the world we live in.

**What is kafka:**

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

Apache Kafka is a distributed data streaming platform that can publish, subscribe to, store, and process streams of records in real time.

It is designed to handle data streams from multiple sources and deliver them to multiple consumers. In short, it moves massive amounts of data, not just from point A to B, but from points A to Z and anywhere else you need, all at the same time.



## Kafka Producer

Producer is an application that sends messages. It does not send messages directly to the recipient. It sends messages only to the Kafka server.

## Kafka Consumer

Consumer is an application that reads messages from the Kafka server.

If producers are sending data, they must be sending it to someone, right? The consumers are the recipients. But remember that the producers don't send data to a recipient address. They just send it to the Kafka server.

## Kafka Broker

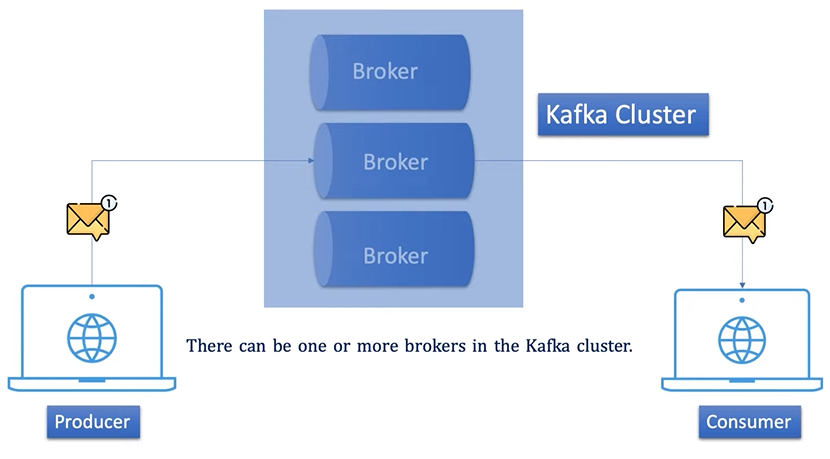
The broker is the Kafka server - act as a message broker between producer and consumer.

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



## Kafka Cluster

Since Kafka is a distributed system, it acts as a cluster. A Kafka cluster consists of a set of brokers. A cluster has a minimum of 3 brokers.

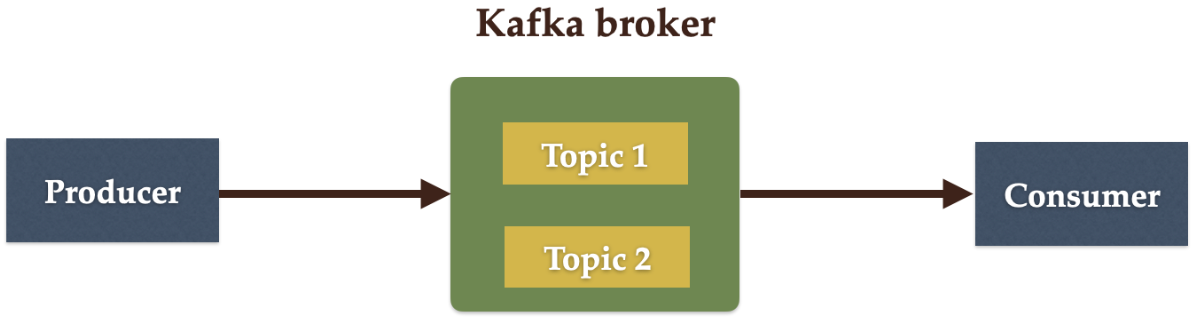


Data in the Kafka cluster is distributed amongst several brokers. There are several copies of the same data in the Kafka cluster. They are called **replicas**. This mechanism makes Kafka even more reliable, fault-tolerant, and stable. If an error occurs with one broker, another broker will start to perform the functions of the broken component. Hence, there are no chances of any information loss.

## Kafka Topic

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.

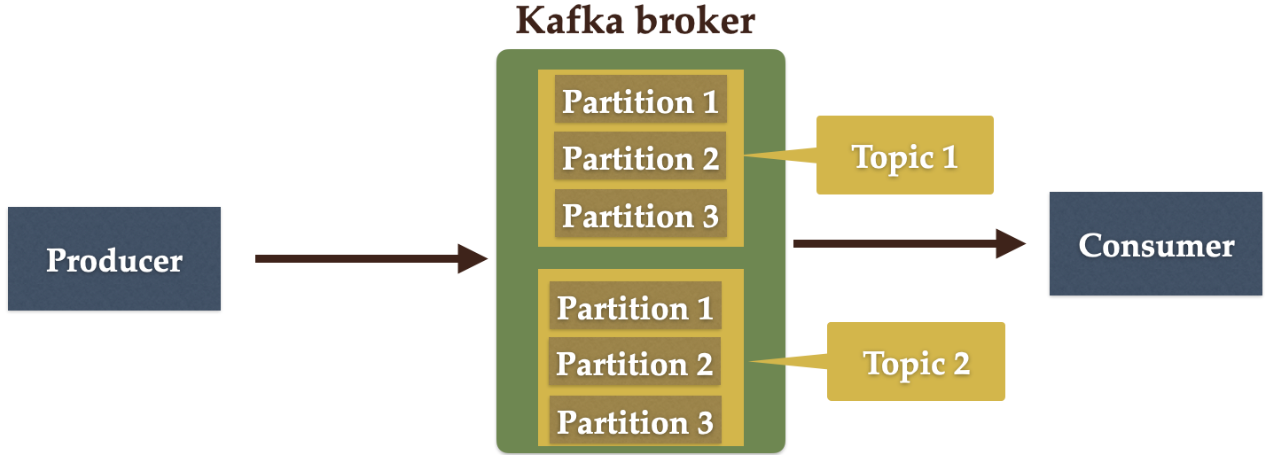
* Topic is like a table in a database or folder in a file system.
* Topic is identified by a name.
* You can have any number of topics.



## Kafka Partitions

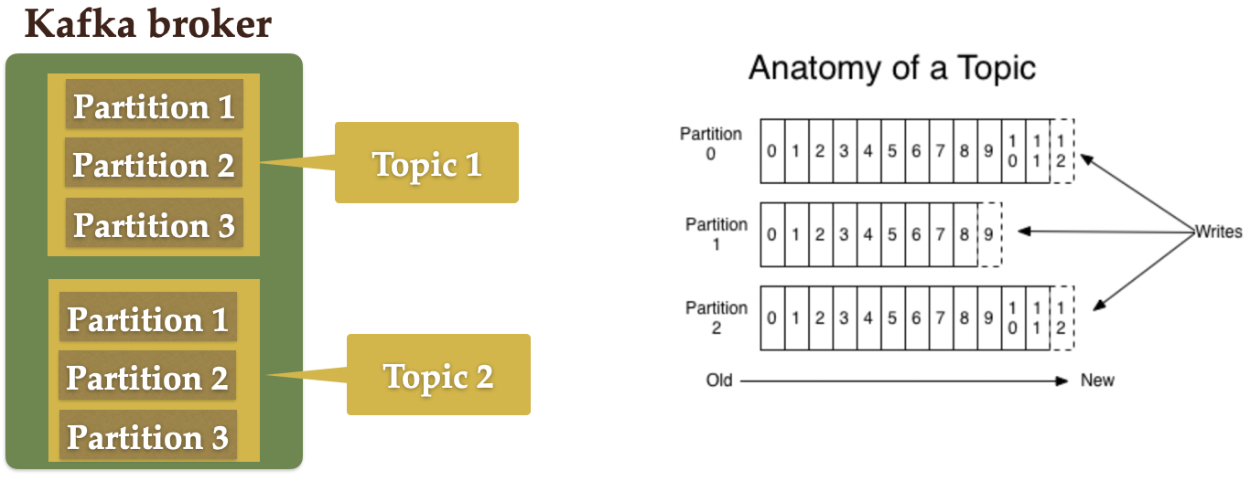
Kafka topics are divided into a number of partitions, which contain records in an unchangeable sequence.

Kafka Brokers will store messages for a topic. But the capacity of data can be enormous and it may not be possible to store in a single computer. Therefore, it will be partitioned into multiple parts and distributed among multiple computers since Kafka is a distributed system.



## Kafka Offsets

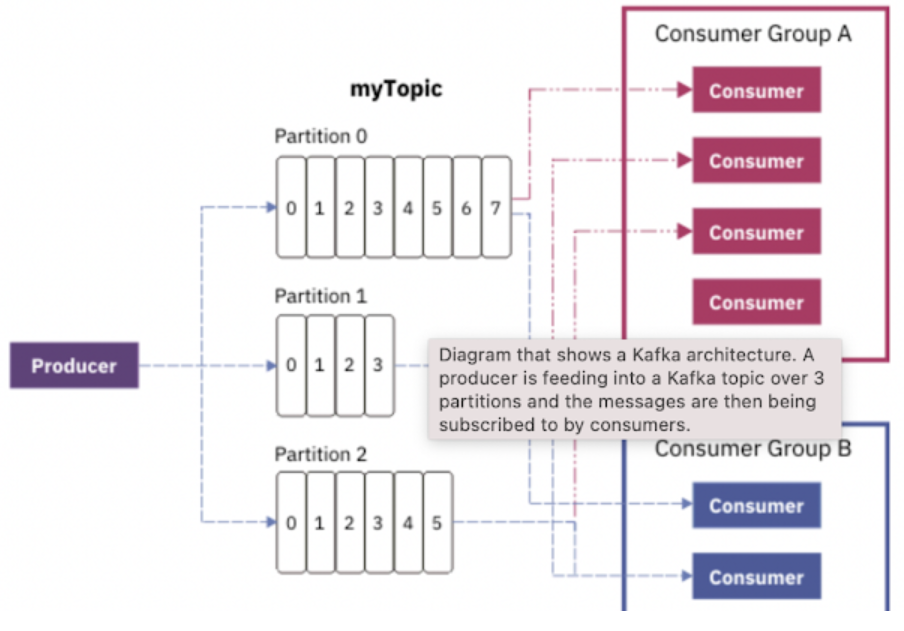
Offset is a sequence of ids given to messages as they arrive at a partition. Once the offset is assigned it will never be changed. The first message gets an offset zero. The next message receives an offset one and so on.



## Kafka Consumer Group

A consumer group contains one or more consumers working together to process the messages.

Consumers will belong to a consumer group. Each consumer within a particular consumer group will have responsibility for reading a subset of the partitions of each topic that it is subscribed to.

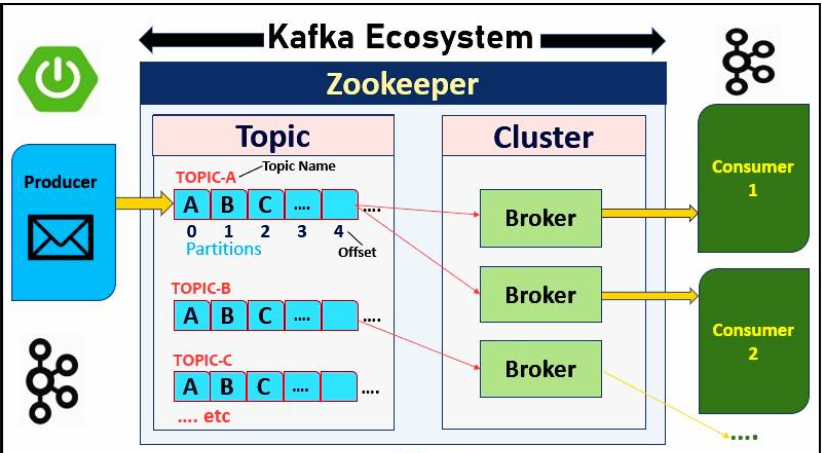


## Zookeeper

It provides a centralized service in distributed systems such as providing configuration information, synchronization, naming registry, and other group services over large clusters.

**Kafka** uses **Zookeeper** in order to track the status of nodes in the Kafka cluster.

The primary role of **ZooKeeper** is to track the status of nodes in the **Kafka** cluster and also maintain a list of **Kafka** topics and messages.



**Kafka Installation:**

Step 1: download kafka

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

step 2: Extract it into some drive

That’s all.

**start Zookeeper:**

D:\kafka\_2.12-3.7.0>bin\windows\zookeeper-server-start.bat config\zookeeper.properties

**start Kafa server:**

D:\kafka\_2.12-3.7.0>bin\windows\kafka-server-start.bat config\server.properties

Zookeeper : 2181

Kafka server/broker : 9092

**Create Topic:**

D:\kafka\_2.12-3.7.0>bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 --create --topic demo-topic --partitions 3 --replication-factor 1

Created topic demo-topic.

D:\kafka\_2.12-3.7.0>bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 --create --topic demo-topic2 --partitions 3 --replication-factor 1

Created topic demo-topic2.

**To List all the topics:**

D:\kafka\_2.12-3.7.0>bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 --list

demo-topic

demo-topic2

**To Describe the topic:**

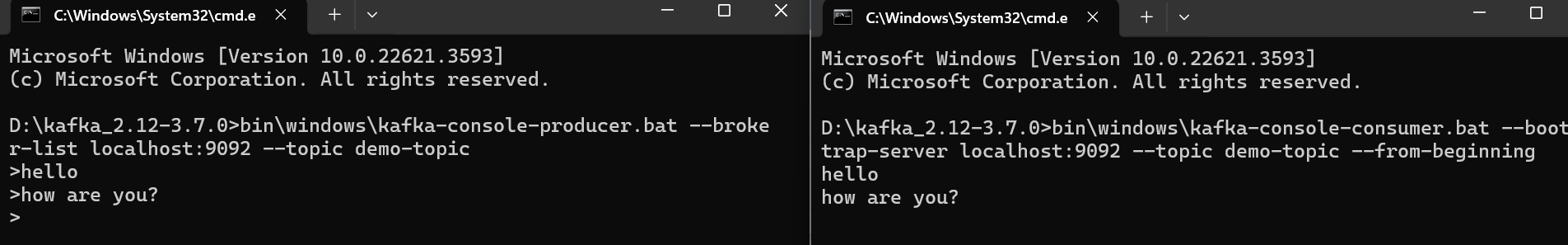
D:\kafka\_2.12-3.7.0>bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 --describe --topic demo-topic

**To send message from broker start the producer:**

D:\kafka\_2.12-3.7.0>bin\windows\kafka-console-producer.bat --broker-list localhost:9092 --topic demo-topic

**To consume message from broker start the producer:**

D:\kafka\_2.12-3.7.0>bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic demo-topic --from-beginning



**To read data from CSV file:**

D:\kafka\_2.12-3.7.0>bin\windows\kafka-console-producer.bat --broker-list localhost:9092 --topic demo-topic2 < D:\trainings\wipro\batch2\cadidates.csv

D:\kafka\_2.12-3.7.0>bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic demo-topic2 --from-beginning