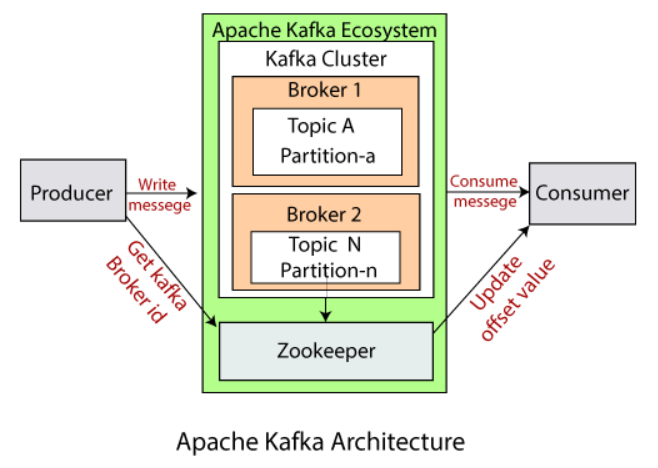
**Kafka:**

Kafka is a Distributed Streaming Platform or a Distributed Commit Log.

Kafka is a distributed streaming platform which can be used for building real-time data pipelines and streaming apps. It is a ***highly scalable, fault-tolerant distributed system***.

Though Kafka started as a publish-subscribe messaging system( similar to message queues or enterprise messaging systems but not exactly), over the years it has involved as a complete streaming platform.

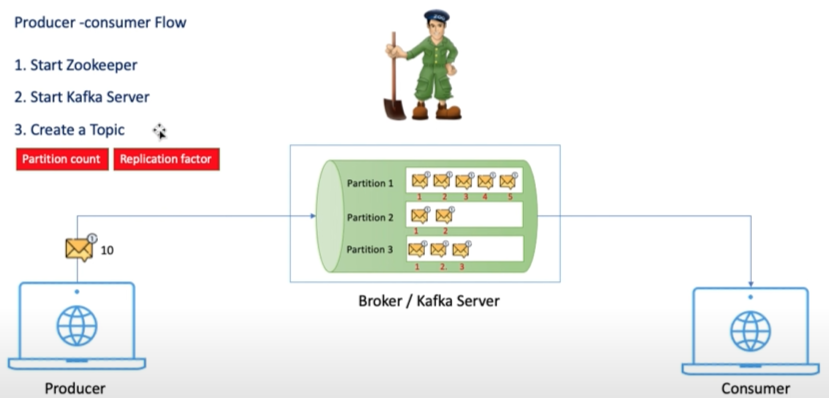


* **Kafka Cluster:** A Kafka cluster is a system that comprises of different brokers, topics, and their respective partitions. Data is written to the topic within the cluster and read by the cluster itself.
* **Producers:** A producer sends or writes data/messages to the topic within the cluster. In order to store a huge amount of data, different producers within an application send data to the Kafka cluster.
* **Consumers:** A consumer is the one that reads or consumes messages from the Kafka cluster. There can be several consumers consuming different types of data form the cluster. The beauty of Kafka is that each consumer knows from where it needs to consume the data.
* **Brokers:** A Kafka server is known as a broker. A broker is a bridge between producers and consumers. If a producer wishes to write data to the cluster, it is sent to the Kafka server. All brokers/server lie within a Kafka cluster itself. Also, there can be multiple brokers.
* **Topics:** It is a common name or a heading given to represent a similar type of data. In Apache Kafka, there can be multiple topics in a cluster. Each topic specifies different types of messages.
* **Partitions:** The data or message is divided into small subparts, known as partitions. Each partition carries data within it having an **offset** value. The data is always written in a sequential manner. We can have an infinite number of partitions with infinite offset values. However, it is not guaranteed that to which partition the message will be written.
* **ZooKeeper:** A ZooKeeper is used to store information about the Kafka cluster and details of the consumer clients. It manages brokers by maintaining a list of them. Also, a ZooKeeper is responsible for choosing a leader for the partitions. If any changes like a broker die, new topics, etc., occurs, the ZooKeeper sends notifications to Apache Kafka. A ZooKeeper is designed to operate with an odd number of Kafka servers. Zookeeper has a leader server that handles all the writes, and rest of the servers are the followers who handle all the reads. However, a user does not directly interact with the Zookeeper, but via brokers. No Kafka server can run without a zookeeper server. It is mandatory to run the zookeeper server.

#### MetaData: The data that describes and provides context for software components, including information about their creation, structure, purpose, and dependencies.

#### Note: With Kafka version 0.10 and above, a Zookeeper does not store the consumer offset value. It is stored in the Kafka topics (as seen in Kafka topics section).

**How to start server?**



Default Port:

Zookeeper: 2181



Kafka/Broker/MessageBroker: 9092



Cmds:

**To run zookeeper:**

zookeeper-server-start.bat ..\..\config\zookeeper.properties

**To run kafka srerver:**

kafka-server-start.bat ..\..\config\server.properties

**To create topic:**

kafka-topics.bat --create --topic my-topic --bootstrap-server localhost:9092 --replication-factor 1 --partitions 3

where:

kafka-topics.bat : used to create topic

my-topic : name of the topic

bootstrap-server localhost:9092 : basically it is a kafka server

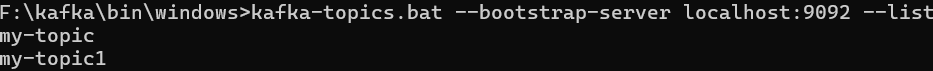
replication-factor 1 : means only one replica(instance) of current broker

partitions 3 : how many partition in a topic. here we have 3 partitions



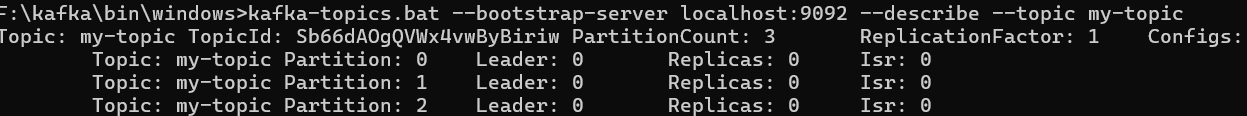
**To get list of topics:**

kafka-topics.bat --bootstrap-server localhost:9092 --list



**To describe about topic:**

kafka-topics.bat --bootstrap-server localhost:9092 --describe --topic my-topic



where:

“my-topic”: topic name

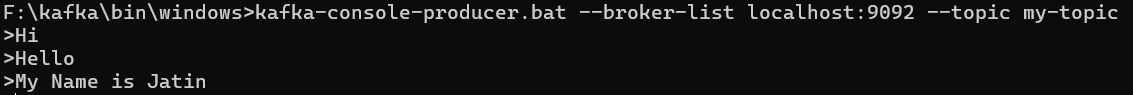
**To run producer:**

kafka-console-producer.bat --broker-list localhost:9092 --topic my-topic

where:

kafka-console-producer.bat: producer server

--broker-list: used to list write all server/broker like localhost:9092 at different port.



Hi, Hello,... are the msgs

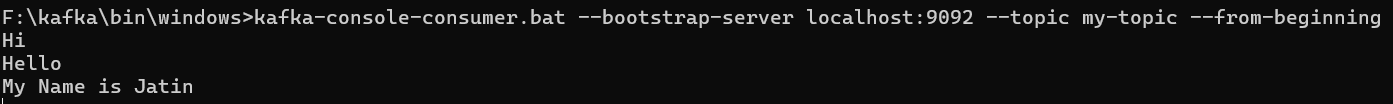
**To run consumer:**

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

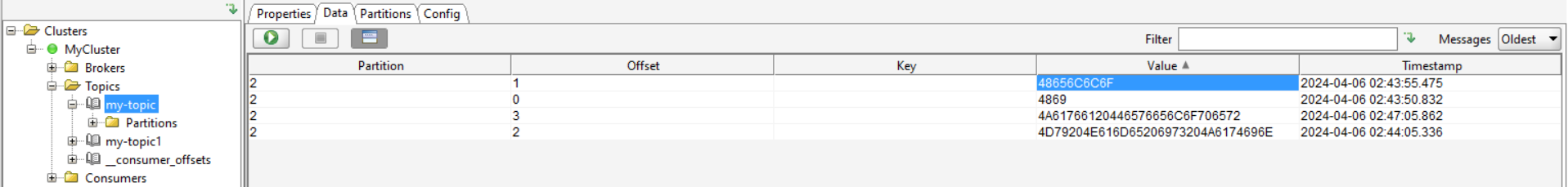
where:

kafka-console-consumer.bat : consumer server

--from-beginning : to read msg from beginning from topic.



Automatically read msg from topic because it is connected to same server and same topic



All the msgs are in partition 2 but it is decide by zookeeper, not in our hand

If you want to send any file then mention the path of that file, it will work

**How to run server without zookeeper?**

We can run using **Kraft** mode

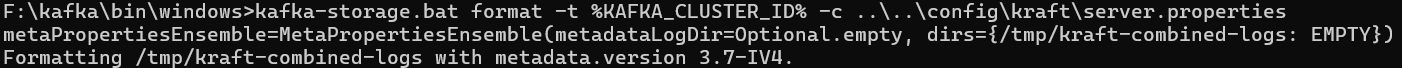
kafka-storage.bat random-uuid



set KAFKA\_CLUSTER\_ID = 1VzFAQz6RhGUM-9pmfdFOw



kafka-storage.bat format -t %KAFKA\_CLUSTER\_ID% -c ..\..\config\kraft\server.properties



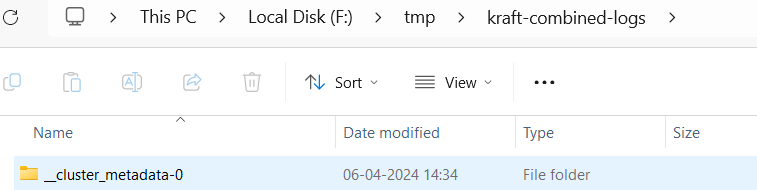
kafka-server-start.bat ..\..\config\kraft\server.properties

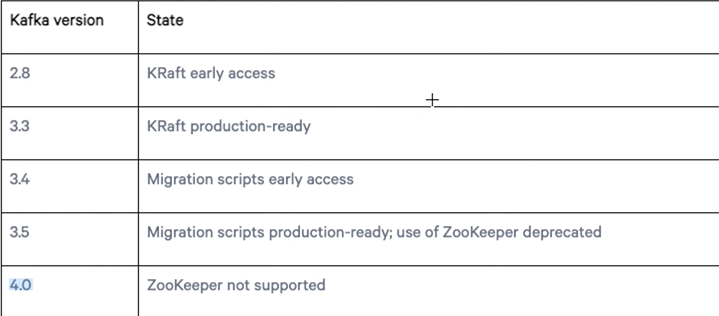


Kafka server started

If zookeeper is not running then where metadata store?

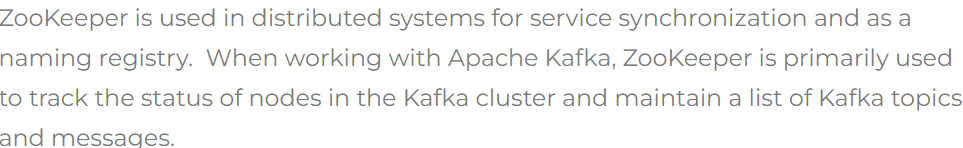
It stores in tmp folder in: **tmp/kraft-combined-logs\_cluster\_metadata-0**

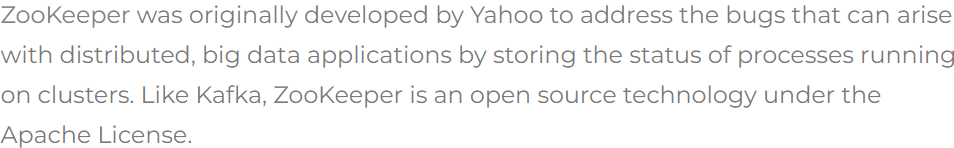


**Eliminate system complexities:** no need to start zooperker. No metadata in zookeeper

Since, zookeeper is a third party sofware. After new version we dont need to depend on third party softwares.





# Kafka Schema Registry and Avro:

Use of docker\_compose.yml?

Compose uses a docker-compose.yml file. It specifies what images are required, what ports they need to expose, whether they have access to the host filesystem, what commands should be run when they start up, and so on.