# Kafka

Apache Kafka is a distributed software system that allows developers to process and store large amounts of real-time data

For this tutorial, Landoop kafka distribution is used. Landoop docker container installation steps exist in install\_and\_config folder.

Open landoop docker bash in CMD and mount local dev folder to docker -

*Directories are generally mounted for consuming the files present in host system by docker.*

*If the below command does not work then, docker copy command can be executed to move files from host system to docker*

|  |
| --- |
| *docker run --rm -it -v %cd%:/main --net=host landoop/fast-data-dev bash* |

Open docker bash in CMD without mounting any directory - use below command

|  |
| --- |
| *docker run --rm -it --net=host landoop/fast-data-dev bash* |

## Test Kafka executing below commands

### Create new topic

|  |
| --- |
| *kafka-topics --create --topic test-topic --bootstrap-server localhost:9092 --replication-factor 1 --partitions 2* |

### List kafka topic

|  |
| --- |
| *kafka-topics --bootstrap-server localhost:9092 --list* |

### Console kafka producer - Starts console producer

|  |
| --- |
| *kafka-console-producer --bootstrap-server localhost:9092 --topic test-topic* |

### Console kafka Consumer - Starts console consumer

|  |
| --- |
| *kafka-console-producer --bootstrap-server localhost:9092 --topic test-topic* *kafka-console-consumer --bootstrap-server localhost:9092 --topic test-topic --from-beginning* |

# Kafka Architecture

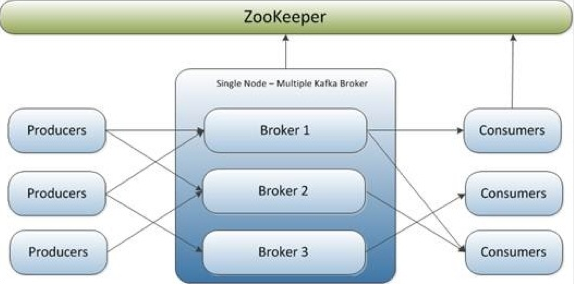
- In production systems, there will be multiple brokers and zookeepers.

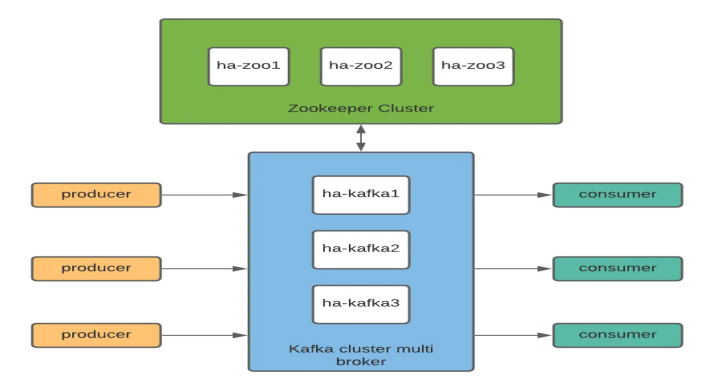
- While configuring the topics we can specify only 1 broker server and configuration property ***replication-factor*** integer value will decide that how many times the topics will be replicated across multiple other brokers.

- If any Broker fails then other brokers can manage the messages to and from the topics

- Zookeeper will manage the health of all the brokers and manage the communication across them

- Multiple Zookeeper servers will be there in case primary zookeeper server fails then other redundant server will take its place





# Topics, partitions, and offsets in Kafka

* Kafka **Topic** is the queue where Producer pushes the messages and Consumer consumes it.
* Each Topic can be divided into multiple **partitions**. Partition Id for a kafka topic start from 0.
* Each partition can have an incremental message counter called **Offset**. Each partition Offset starts from 0 and it will incrementally increase. Offset denotes the unique position of a message within a partition.
* Offset guarantees the chronological order of the messages in a partition, but it does not guarantee the order across different partitions i.e. In a same partition, Message with Offset 0 came before the Message with Offset 1 however, Message with Offset 1 in Partition 1 may occur before Offset 0 of Partition 2.

A screen shot of a screen

Description automatically generated

* Messages stored in the topics as immutable objects.
* Messages in a topic are stored in partitions. Each message is composed of a key, a value, and additional metadata (such as headers and timestamps).
* The key-value pair is the main component of each message, where the key is optional and is often used to determine the partition to which the message is assigned.
* Based on the key, Kafka stores messages in specific partitions, ensuring that messages with the same key are consistently placed in the same partition. If the key is absent, Kafka distributes the messages across partitions using a round-robin or another default strategy.
* Consumers are not required to consume messages in chronological order, especially when consuming from multiple partitions, as each partition can be processed independently and concurrently.
* Consumers can consume the messages from a specific Partition and can also be configured with the Offset (offset indicates the specific position from which a consumer starts or continues consuming messages in a partition)
* While configuring Offset in the consumers, Partition needs to be configured else error will occur.

***Bash Scripts:***

|  |
| --- |
| * **Create new topic with 3 partitions**   *kafka-topics --create --topic test-topic --bootstrap-server localhost:9092 --replication-factor 1 --partitions 3*   * **Console kafka Consumer - Configure consumer to get the messages from a specific partition**   *kafka-console-consumer --bootstrap-server localhost:9092 --topic test-topic --partition 1*   * **Console kafka Consumer - Configure consumer to get the messages from a specific Offset**   *kafka-console-consumer --bootstrap-server localhost:9092 --topic test-topic --partition 1 --offset 1*   * **Console kafka Consumer - configure consumer to get the messages from a specific Offset without configuring partition. Below code will throw error, while configuring offset, partition needs to be configured**   *kafka-console-consumer --bootstrap-server localhost:9092 --topic test-topic --offset 1* |

# Kafka Cluster

In production environment multiple brokers are used for fault tolerance.

For this exercise, Kafka is downloaded on Windows system and multiple copies of server.properties files will be used to spin up (mimic) cluster with multiple brokers.

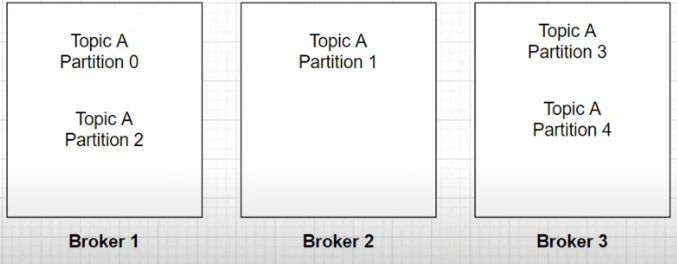
Each copy of server.properties file will have different listener, different broker Id, different server log folder path and Zookeeper host will remain same in all the files, so that Zookeeper can track all the brokers.

Installation steps exist in install\_and\_config folder.

***Scripts:***

|  |
| --- |
| * **Start Zookeeper using below code - change path of zookeeper.properties**   *zookeeper-server-start D:\kafka\config\zookeeper.properties*   * **Start kafka clsuter - multiple brokers**   *kafka-server-start D:\kafka\config\server0.properties*  *kafka-server-start D:\kafka\config\server1.properties*  *kafka-server-start D:\kafka\config\server2.properties*   * **Create a topic with 5 partitions**   *kafka-topics --create --topic test-cluster-topic --bootstrap-server localhost:9092,localhost:9093,localhost:9094 --replication-factor 1 --partitions 5*   * **Create kafka console Producer**   *kafka-console-producer --topic test-cluster-topic --bootstrap-server localhost:9092,localhost:9093,localhost:9094*   * **Create kafka console consumer**   *kafka-console-consumer --topic test-cluster-topic --bootstrap-server localhost:9092,localhost:9093,localhost:9094* |

In Kafka Cluster with multiple brokers (replication-factor = 1), whenever a topic is created with multiple partitions, those partitions will reside in different brokers so that if any broker crashes, then partitions in other brokers should still be receiving and sending messages.



In above screenshot, if Broker 1 crashes then Partition 1, 3 and 4 will still be receiving and sending messages.

However in standalone cluster i.e. cluster with 1 broker, if the broker crashes, the entire topic will be lost since all the partitions reside in 1 broker.