**Need of kafka:**

Apache Kafka is a powerful tool for handling real-time data streams and is used in a variety of use cases. Some of the main reasons why Kafka is often used include:

1. **Scalability:** Kafka can handle high volumes of data and can easily handle an large amount of data.
2. **Real-time streaming:** Kafka is designed to handle real-time data streams, allowing us to process and analyze data as it is generated.
3. **Durability:** kafka stores all published records for a configurable amount of time, this allows to handle situations like server failures, network issues, etc.
4. **Flexibility:** kafka can be integrated with a wide range of other tools and systems, including Apache Storm, Apache Hadoop, and Apache Spark, to build complex real-time data pipelines.
5. **Decoupling:** Kafka allows to decouple data producers and data consumers, so that if one system goes down, it does not affect other systems.
6. **Fault-tolerance:** kafka is built for fault-tolerance. It stores data on disk, and replicates data across multiple servers to ensure that data is not lost in case of server failure.
7. **Distributed:** Kafka is a distributed system, so it can handle a high volume of data from multiple sources, and it can be easily scaled out.
8. **High performance:** kafka is designed to handle high throughput, low latency, and high concurrency, making it a good fit for big data and real-time analytics use cases.

Whatever the system sends message is called **“Producer”** and whatever the system receives message is called **“Consumer”** with the help of **“kafka cluster”**

**Broker:** kafka brokers are the server’s that manage the conversations b//w two different systems. Brokers are responsible for the delivery of messages to the right party.

Kafka broker is having some ip and port for publishing the messages

**Message**: messages are simply byte arrays and any object can be stored in any format by developers. The format can be in string, json and etc

**Topic**: which is used to store the messages for temporary purpose when a producer sends, and the consumer will collect that data from the topic. Topics are identified by it’s name which was unique in nature.

**Cluster:** A cluster is a group of servers that work together to handle a larger amount of data and traffic. It can have one or more Kafka brokers, which can be added or removed as needed to handle changes in traffic or data volume.

**Producers**: producer is a client application that processes or sends messages to one or more topics in a Kafka cluster. Producers are the source of data stream in kafka basically.

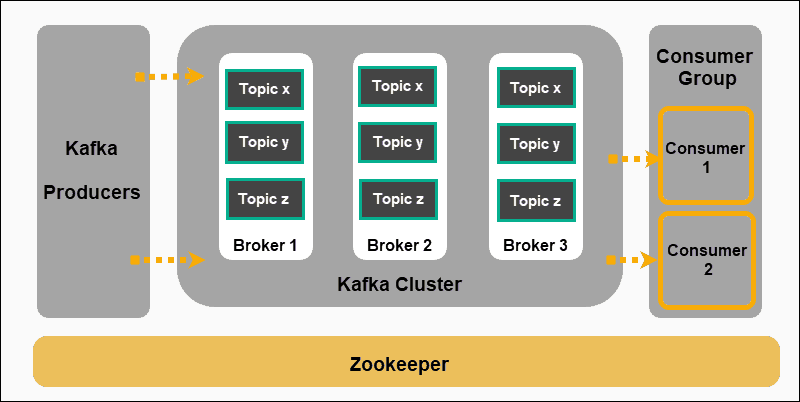
**Consumers:** consumer is a client application that reads messages from one or more topics in a Kafka cluster by subscribing to one or more topics in kafka cluster.

**Partitions:** every broker holds few partitions, and each partition can be either leader or replica for a topic. All writes and reads to a topic go via the leader. Leader is responsible for updating replicas with new data. If the leader fails replica acts as a leader.

**Offset:** Data in the offset will lasts for 7days by default we can also modify this by our requirement once the data is written it can ‘t be changed

There is a cluster 🡪broker🡪topics🡪messages🡪n-partitions 🡪n-offsets

**Architecture of kafka cluster**



Zookeeper is used to maintain the cluster how the services inside the cluster are running and how the different nodes inside the cluster are running are maintained by zookeeper. In simple terms there is Leader election for partitions and replicas Zookeeper is used to elect leaders and handle the failover process when a leader goes down.

**Cluster types:**

1. **Single node single broker** (mainly used in test env)
2. **Single node multiple brokers** (mainly used in dev env)
3. **Multiple nodes multiple brokers** (mainly used in prod)

**Kafka zookeeper configuration:**

1. Launch 3 instances (t2-medium)
2. Add the security rules 9092(Kafka port),2888(internal zookeeper forum creation port),3888(internal zookeeper forum creation port),2181(zookeeper port)
3. Connect to the instances and follow the below steps
4. Change as root user
5. apt-get update

sudo apt install openjdk-8-jdk -y

1. Go to [**https://kafka.apache.org/downloads**](https://kafka.apache.org/downloads)and download the required version of scala
2. Go to the kafka server

Wget <https://archive.apache.org/dist/kafka/2.8.0/kafka_2.13-2.8.0.tgz> in all kafka machines

1. Extract the file and rename the file

tar -xvf kafka\_2.13-2.8.0.tgz

mv kafka\_2.13-2.8.0 kafka

1. cd kafka/config

ls -lrt (we will find some files)

zookeeper.properties file is used to create zookeeper

server.properties file is used to create brokers

1. cd ..

cd /bin (all script files are available here)

1. create a data-dir/log-dir

mkdir -p data/zookeeper (create this in kafka folder)

1. create the id for zookeeper in kafka folder

echo “1” > data/zookeeper/myid

1. we need to specify the data-dir in zookeeper.properties

vi config/zookeeper.properties

dataDir=/root/kafka/data/zookeeper(reference point viii)

1. Start the zookeeper

cd /bin

./zookeeper-server-start.sh ../config/zookeeper.properties (run as a tar-ball)

Or

./zookeeper-server-start.sh -daemon ../config/zookeeper.properties(run as daemon in background)

1. To check zookeeper is connected or not

nc -vz localhost 2181 (or) ./zookeeper-shell.sh localhost:2181 than ls / to come out type quit

1. vi ../config/server.properties

specify logs-dir

log.dirs=/root/kafka/data/kafka (create kafka dir before doing this)

1. run the broker

cd /bin

./kafka-server-start.sh ../config/server.properties(run as tar ball)

Or

./kafka-server-start.sh -daemon ../config/server.properties(run in background)

1. to check the kafka is running or not

nc -vz localhost 9092

1. **creating a topic**

./kafka-topics.sh –bootstrap-server localhost:9092 –create –replication-factor[replics-num] –partitions[partitions-num] –topic [topic name]

**Eg:** ./kafka-topics.sh –bootstrap-server localhost:9092 –create –replication-factor 1

–partitions 3 –topic states

1. **check list of how many topics**

./kafka-topics.sh –bootstrap-server localhost:9092 --list

1. **check the detailed description of a topic**

./kafka-topics.sh –bootstrap-server localhost:9092 –describe –topic [topic name]

**Eg:** ./kafka-topics.sh –bootstrap-server localhost:9092 –describe –topic states

1. **deleting a topic**

./kafka-topics.sh –bootstrap-server localhost:9092 –delete –topic [topic name]

1. **producing messages to a producer**

./kafka-console-producer.sh –bootstrap-server localhost:9092 –topic [topic name]

**Eg:** ./kafka-console-producer.sh –bootstrap-server localhost:9092 –topic states

**Then enter the messages after that (ctrl+c)**

1. **consume the messages to a consumer**

./kafka-console-consumer.sh –bootstrap-server localhost:9092 –topic [topic name] –from-beginning

**Eg:** ./kafka-console-consumer.sh –bootstrap-server localhost:9092 –topic states –from-beginning

1. **to read a message from particular partition**

./kafka-console-consumer.sh –bootstrap-server localhost:9092 –partition [partition num] –topic [topic name] –from-beginning

**Eg:** ./kafka-console-consumer.sh –bootstrap-server localhost:9092 –partition 2 –topic states –from-beginning

1. **to read a message from particular partition and particular offset**

./kafka-console-consumer.sh –bootstrap-server localhost:9092 –partition [partition num] –topic [topic name] –offset [offset num]

**Eg:** ./kafka-console-consumer.sh –bootstrap-server localhost:9092 –partition 1 –topic states –offset 1

1. **To read the messages from a specific consumer group**

./kafka-console-consumer.sh –bootstrap-server localhost:9092 –topic [topic name]

--group [group name] –from-beginning

**Eg:** ./kafka-console-consumer.sh –bootstrap-server localhost:9092 –topic states

--group states –from-beginning

1. **To check the messages which are stored in consumer groups**

./kafka-consumer-group.sh –bootstrap-server localhost:9092 –list

1. **To check the logs for the created topics**

cd /root/kafka/data/kafa/logs

here we can find the topics enter into the topics and go for logs and see the logs

**3 brokers and 3 zookeepers in single machine**

1. Go to config dir and make the server.properties file and zookeeper.properties as many as for our requirement
2. Add these changes in config file

**# the directory where the snapshot is stored.**

**dataDir=/root/kafka/data/zookeeper1**

tickTime=2000

initLimit=10

syncLimit=5

**#the ports at which client will connect**

Clientport=2181 (change the port num for every client connection)

server1=localhost:2888:3888

server2=localhost:2889:3889

server3=localhost:2890:3890

1. Do the same for remaining zookeeper.properties (refer point 2)
2. Run all the zookeeper servers
3. Check the connections using

nc -vz localhost 2181/2182/2183

1. To configure 4 letters word to get the reply from kafka

Cd /config

Vi zookeeper.properties

4lw.commands.whitelist=\* (add this at the end of the file)

1. To get these changes restart the server
2. To check the 4letters words (conf,envi,cons,dump,srvr,stat,mntr)

Echo “ruok” | nc localhost 2181

1. Now add the brokers in config/server.properties

**# The id of the broker. This must be set to a unique integer for each broker.**

**broker.id=101**

log.dirs=/root/kafka/data/kafka-logs1

**advertised.listeners=PLAINTEXT://localhost:9092**

**listeners=PLAINTEXT://0.0.0.0:9092**

**zookeeper.connect=localhost:2181,localhost:2182,localhost:2183**

1. Start the kafka broker and Check the logs

cd logs/server.log

if the broker is shutting down then remove the kafka-logs1 in data dir and run as a daemon and check the logs now it will be fine

1. Do the same for the remaining brokers
2. Now create the topic

./kafka-topics.sh –bootstrap-server localhost:pn,localhost:pn,localhost:pn –create –replication-factor [replicas-num] –partitions [partitions-num] –topic [topic-name]

**Eg:** ./kafka-topics.sh –bootstrap-server localhost:9092,localhost:9093,localhost:9094 –create –replication-factor 3 –partitions 5 –topic colours

1. After that try to produce some messages to the created topic and check the logs for the topic in /root/kafka/data/
2. Now consume the messages

./kafka-console-consumer.sh –bootstrap-server localhost:pn,localhost:pn,localhost:pn –topic colours –from-beginning

**Eg:** ./kafka-console-consumersh –bootstrap-server localhost:9092,localhost:9093,localhost:9094 –topic colous --from-beginning

1. Check the logs whether the offsets are created or not

cd /root/kafka/data/kafka-logs1

1. Create another topic (topic name --numbers)
2. Procduce some messages to the topic and then describe it
3. After that kill one broker and then describe the topic once to check the diff
4. And also stop one more broker and check the describe for the topic after some time start the brokers

**Performing a perf-test in a particular topic**

1. Create a topic

**Eg:** ./kafka-topics.sh –bootstrap-server localhost:9092 –create –replication-factor 3 –partitions 100 –topic perf

1. Then once describe the topic created
2. Produce some messages

./kafka-producer-perf-test.sh –topic [topic name] –num-records [num of msgs to send] –throughput [no of msgs to go per second] –record-size [size] –producer-props bootstrap.server=localhost:9092

**Eg:** ./kafka-producer-perf-test.sh –topic perf –num-records 1000 –throughput 10 –record-size 1000 –producer-props bootstrap.server=localhost:9092