***Kafka Notes***

* by default, the retention policy is set to 7 days, of the records in a partition in kafka
* a consumer will specify the topic, partition no. and the offset from where it has to read
* partitions are there to support parallelism i.e multiple consumers can consume the data simultaneously

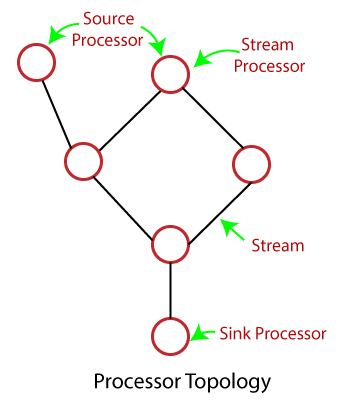
***Zookeeper performs 3 major functions***:

electing a controller, cluster membership & topic configuration

* electing a controller:
* Controller is one of the brokers and is responsible for maintaining leader-follower relationship for all partitions, if a node shuts down the controller tells other replicas to become partition leaders.
* Zookeeper, elects a controller and makes sure that there is only one controller in the kafka cluster, and also elects a new one if it crashes
* Cluster membership: Zookeeper checks which broker are alike and are part of the cluster
* Topic configuration:
* Zookeeper keeps the track of which topics exists, how many partitions each topics has, where are the replicas, who’s the preferred leader etc.
* Basically, the zookeeper keeps a lot of information, so that it helps in providing a reliable system, whenever a node fails.

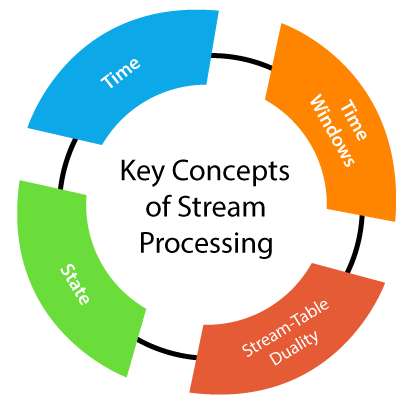
**Kafka Streams**

* streams are the continuous real-time flow of the facts or records (key-value pairs)
* Processor topologies are represented graphically where 'stream processors' are its nodes, and each node is connected by 'streams' as its edges.



* two major processors present in the topology:
* **Source Processor:** The type of stream processor which does not have any upstream processors. This processor consumes data from one or more topics and produces an input stream to its topologies.
* **Sink Processor:** This is the type of stream processor which does not have downstream processors. The work of this processor is to send the received data from its upstream processors to the specified topic.

**Key concepts of Stream Processing**



**TIME:**

1. **Event Time:** The time when an event had occurred, and the record was originally created. Thus, event time matters during the processing of stream data.
2. **Log append time:** It is that point of time when the event arrived for the broker to get stored.
3. **Processing Time:** The time when a stream-processing application received the event to apply some operations. The time can be in milliseconds, days, or hours. Here, different timestamps are assigned to the same event, depending on exactly when each stream processing application happened to read the event. Also, the timestamp can differ for two threads in the same application. Thus, the processing time is highly unreliable, as well as best avoided.

**STATE:**

1. **Internal or local state:** The state which can be accessed only by a specific stream-processing application’s instance. The internal state is managed and maintained with an embedded, in-memory database within the application. Although local states are extremely fast, the memory size is limited.
2. **External state:** It is the state which is maintained in an external data store such as a NoSQL database. Unlike the internal state, it provides virtually unlimited memory size. Also, it can be accessed either from different applications or from their instances. But it carries extra latency and complexity, which makes it avoidable by some applications.

**STREAM-TABLE DUALITY:**

if the user wants to convert streams into a table, it is required to convert all changes which a stream contains. This process of conversion is also called **materializing the stream**. So, we can have the dual process of changing streams into tables as well as tables to streams.

**TIME WINDOWS**

The term time windows means windowing the total time into parts. Therefore, there are some operations on streams which depend on the time window. Such operations are called **Windowed operations**. For example, join operation performed on two streams are windowed.