***Apache Kafka Metrics***

**Producer matrices**

Producers are responsible for producing data to Kafka topics. If the producer fails, the consumer will not have any new messages to consume and it will be left idle. The performance of the producer also plays an important role in achieving high throughput and latency. Let's look into a few important matrices of Kafka producer:

**Response rate**:

The producer sends records to the Kafka broker, and the broker acknowledges when a message is written to a replica in case of a request. Required .acks is set to -1. The response rate depends on the value assigned to this property. If set to, -0, the broker will immediately return a response when it receives a request from the producer before it writes data to disk. If set to 1, the producer first writes data to its disk and then returns a response. Obviously, less fewer write operations will lead to high performance, but there will be chances of losing data in such cases.

**Request rate**:

The request rate is the number of records the producer produces within a given time.

**I/O wait time**:

The producer sends data and then waits for data. It may wait for network resources when the producing rate is more than the sending rate. The reason for a low producing rate could be slow disk access, and checking the I/O wait time can help us identify the performance of reading the data. More waiting time means producers are not receiving data quickly. In such cases, we may want to use fast access storage such as SSD.

**Failed send rate**:

This gives the number of message requests failed per second. If more messages are failing, it triggers an alarm to find out the root cause of the problem and then fix it.

**Buffer total bytes**:

This represents the maximum memory the producer can use to buffer data before it sends it to brokers. The maximum buffer size will result in high throughput.

**Compression rate**:

This represents the average compression rate for batch records for topic. A higher compression rate triggers us to change the compression type or look for some way to reduce it.

**Broker matrices**

Brokers are responsible for serving producer and consumer requests. They also contain important matrices that can help you avoid some critical issues. There are a lot of metrics available, but we will only look into a few important ones.

*MetricsDescriptionkafka.server:type=ReplicaManager,name=UnderReplicatedPartitions*

This represents the number of under-replicated partitions. A higher number of under-replication partition may result in losing more data in case the broker fails.

*kafka.controller:type=KafkaController,name=OfflinePartitionsCount*

This represents the total number of partitions that are not available for read or write because of no active leader for those partitions.

*kafka.controller:type=KafkaController,name=ActiveControllerCount*

This defines the number of active controllers per cluster. There should not be more than one active controller per cluster.

*kafka.server:type=ReplicaManager,name=PartitionCount*

This represents the number of partitions on the broker. The value should be even across all brokers.

*kafka.server:type=ReplicaManager,name=LeaderCount*

This represents the number of leaders on the broker. This should also be even across all brokers; if not, we should enable auto rebalancer for the leader.

**Consumer metrics**

Consumers are responsible for consuming data from topic and doing some processing on it, if needed. Sometimes, your consumer may be slow, or it may behave unacceptably. The following are some important metrics that will help you identify some parameters that indicate optimization on the consumer side:

*records-lag-max*:

The calculated difference between the producer's current offset and the consumer's current offset is known as record lag. If the difference is very big, it's fairly indicative of the consumer processing data much slower than the producer. It sends alerts for suitable action to fix up this issue, either by adding more consumer instance or by increasing partitions and increasing consumers simultaneously.

*bytes-consumed-rate*:

This represents the number of bytes consumed per second by the consumer. It helps in identifying the network bandwidth of your consumer.

*records-consumed-rate*:

This defines the number of messages consumed per second. This value should be constant and generally helps when compared with byte-consumed-rate.

*fetch-rate*:

This represents the number of records fetched per second by the consumer.

*fetch-latency-max*:

This represents the maximum time taken for the fetch request. If it's high, it triggers to optimize the consumer application.