kakfa-broker :

..................

1. **Active Controller :**

Controller Broker (KafkaController) is a Kafka service that runs on every broker in a Kafka cluster,but only one can be active (elected) at any point in time. The process of promoting a broker to be the active controller is called Kafka Controller Election.

1. **Brokers online** : number of broker uptime
2. **Unclean Leader Replication :**

The Unclean Leader Elections metric shows the number of failures to elect a suitable leader per second.

Unclean leader elections are caused when there are no available in-sync replicas for a partition (either due to network issues,

lag causing the broker to fall behind, or brokers going down completely), so an out of sync replica is the only option for the leader.

When an out of sync replica is elected leader, all data not replicated from the previous leader is lost forever.

1. **.preferred replica imbalance count** :

imbalance count in the preferred replica.

1. **offline partition count** :

Offline partitions represent data stores unavailable to your applications due to a server failure or restart. In a Kafka cluster,

one of the brokers serves as the controller responsible for managing the states of partitions and replicas and to reassign partitions when needed

1. **Under Min ISR Partitions :**

The two most common causes of under-min ISR partitions are that one or more brokers is unresponsive, or the cluster is experiencing performance issues and one or more brokers are falling behind.Each partition has n replicas and can tolerate n-1 replica failures. One of the replicas is elected as the leader and the rest of the replicas are followers. The leader maintains a set of in-sync replicas (ISR): the set of replicas that have fully caught up with the leader. For each partition, we store in Zookeeper the current leader and the current ISR.

1. **time spent in gc :**

Displays the percentage of elapsed time that was spent performing a garbage collection since the last garbage collection cycle. This counter usually indicates the work done by the garbage collector to collect **and compact memory on behalf of the application**

1. **NetworkProcessorAvgIdlePercent :**

Average fraction of time the network processor threads are idle. Values are between 0 (all resources are used) and 1 (all resources are available).

1. **RequestHandlerAvgIdlePercent** :

Average fraction of time the request handler threads are idle. Values are between 0 (all resources are used) and 1 (all resources are available).

1. **MessagesInPerSec**

Aggregate incoming message rate.

1. **Zookeeperrequestlatency:**

Amount of time it takes for the server to respond to a client **request** (since the server was started). When **latency** > 10 (Ticks). Number of queued **requests** in the server. This goes up when the server receives more **requests** than it can process.

1. **ZooKeeperSyncConnectsPerSec**

ZooKeeper client is connected to the ensemble and ready to execute operations.

1. **ZooKeeperExpiresPerSec**

The ZooKeeper session has expired. When a session expires, we can have leader changes and even a new controller. It is important to keep an eye on the number of such events across a Kafka cluster and if the overall number is high, then we have a few recommendations:

1. Check the health of your network
2. Check for garbage collection issues and tune it accordingly
3. If necessary, increase the session time out by setting the value of zookeeper.session.timeout.ms.
4. **ZooKeeperDisconnectsPerSec**

ZooKeeper client is currently disconnected from the ensemble. The client lost its previous connection to a server and it is currently trying to reconnect. The session is not necessarily expired

1. **=ZooKeeperAuthFailuresPerSec**

An attempt to connect to the ensemble failed because the client has not provided correct credentials.

1. **IsrShrinksPerSec**

If a broker goes down, ISR for some of the partitions will shrink. When that broker is up again, ISR will be expanded once the replicas are fully caught up. Other than that, the expected value for both ISR shrink rate and expansion rate is 0.

1. **IsrExpandsPerSec**

When a broker is brought up after a failure, it starts catching up by reading from the leader. Once it is caught up, it gets added back to the ISR.

**bil**), which means “band interleaved by line / Binary Interleaved by Line(BIL) - image/bil. This is actually a very simple raw binary format

1. **RequestQueueTimeMs,request={Produce|FetchConsumer|FetchFollower}**

Time the request waits in the request queue.

1. **LocalTimeMs,request={Produce|FetchConsumer|FetchFollower}**

Time the request is processed at the leader.

1. **=RemoteTimeMs,request={Produce|FetchConsumer|FetchFollower}**

Time the request waits for the follower. This is non-zero for produce requests when acks=all.

1. **ResponseQueueTimeMs,request={Produce|FetchConsumer|FetchFollower}**

Time the request waits in the response queue.

1. **kafka.network:type=RequestMetrics,name=RequestsPerSec,request={Produce|FetchConsumer|FetchFollower}**

Request rate.