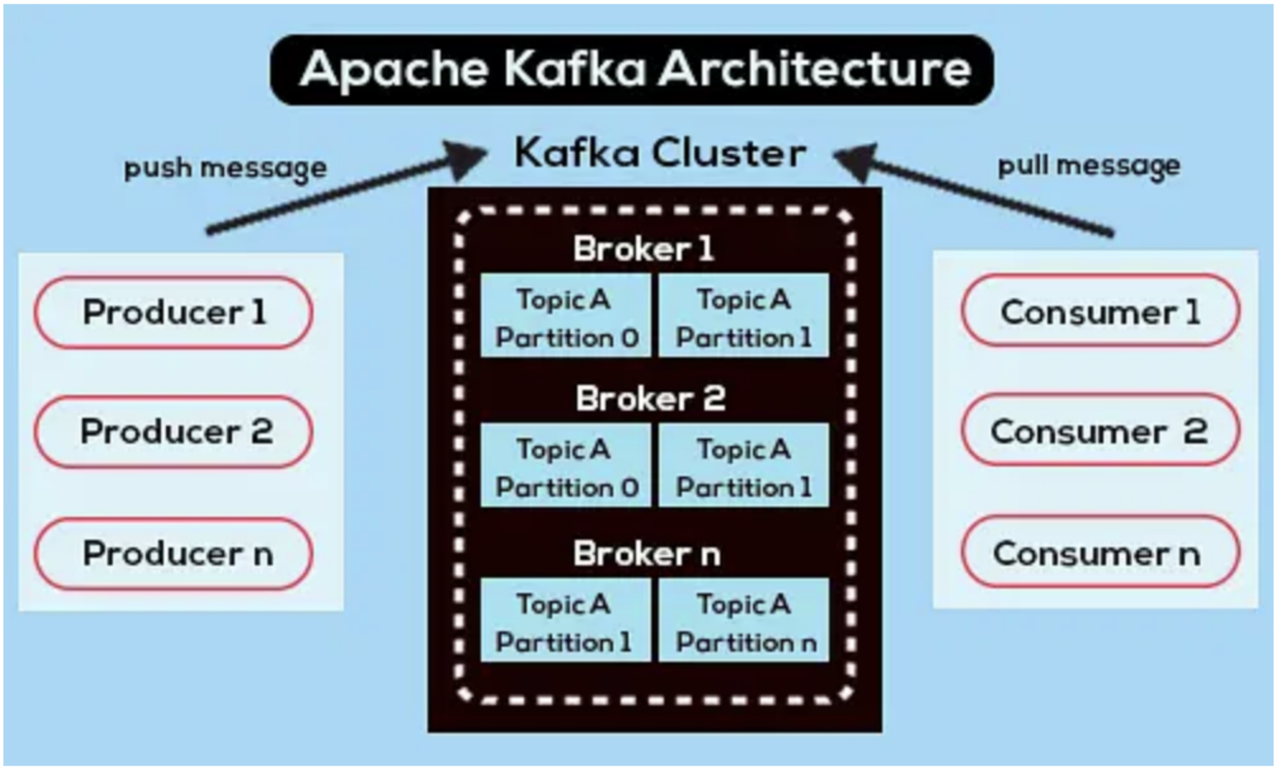
Kafka Learning Note

- **Intro**

Kafka combines three key capabilities for event streaming end-to-end solutions:

1. To **publish** (write) and **subscribe to** (read) streams of events, including continuous import/export of your data from other systems.
2. To **store** streams of events durably and reliably for as long as you want.
3. To **process** streams of events as they occur or retrospectively.



- **Producer**

The producer communicates with the Kafka broker hosts (worker nodes) and sends data to a Kafka topic.

0 1 Kafka 0 Topic 100 0 99

blocked URL

- **Broker**

A Kafka cluster consists of one or more servers (**Kafka brokers**).

Each **Broker** can have one or more **Topics**. Kafka topics are divided into a number of partitions, each partition can be placed on a single or separate machine to allow for multiple consumers to read from a topic in parallel.

1 Topic --N Partitions -distributed to N brokers

- **Partition, Replica**

  Each partition is ordered.

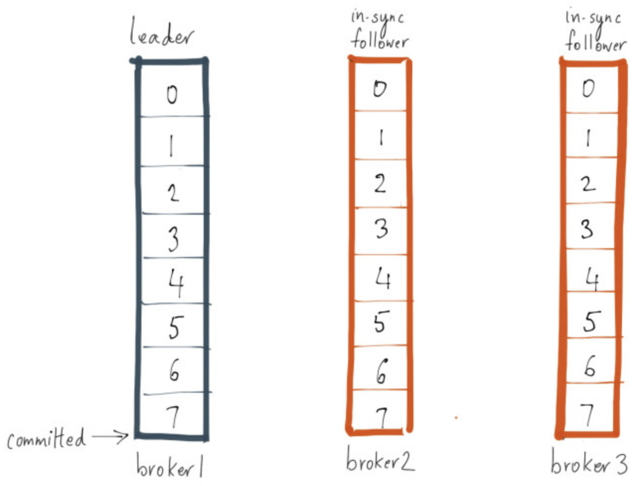
  1 lead partition --->N-1 replica

  In general, writes and reads happens to the leader.

  Kafka M N N N-1 0

https://time.geekbang.org/column/article/99318

- **ISR**
  - Definition: **ISR**In-Sync Replicas leader follower + leader
  - Purpose: Keep HA for the client side.
    - With the ISR list, Producer knows how many replicas in the list are waiting for the data.
    - New Leader will be elected from the ISR list.
  - MechanismIn-Sync Replica: follower --pull data-lead (data should be synchronized in 10 sec. (Alert?)



Acks (acknowledgments)

An ack is **an acknowledgment that the producer gets from a Kafka broker to ensure that the message has been successfully committed to that broker**. The config acks is the number of acknowledgments the producer needs to receive before considering a successful commit.

acks=0fire and forgetProducerProducer

acks=1

acks=all ISR

- **Controller**

  Controller is responsible for the lead replica election.

- **Consumer**

  N partitions --1 Consumer

- **Consumer Group**

A consumer group is **a set of consumers that cooperate to consume data from some topics**. The partitions of all the topics are divided among the consumers in the group.

Three features of Consumer Group

1 Topic-N partitions-lead replica- Consumer group1---->Consumer11

Consumer group2--Consumer21

1. *Consumer Group Consumer*
2. *Group ID Kafka Consumer Group*
3. *Consumer Group Consumer Group*

- **Rebalance1 partition-1 consumer**

  Rebalance Consumer Group Consumer Topic Group 20 Consumer 100 TopicKafka Consumer 5 Rebalance

  total partition number/total group consumer number= Ideal balanced number per consumer

  Consumer Consumer “”Consumer Group consumer.subscribe(Pattern.compile("t.*c")) Group t c Consumer Group Group RebalanceKafka Group Rebalance

  Pro Average assigned the partitions to consumer.

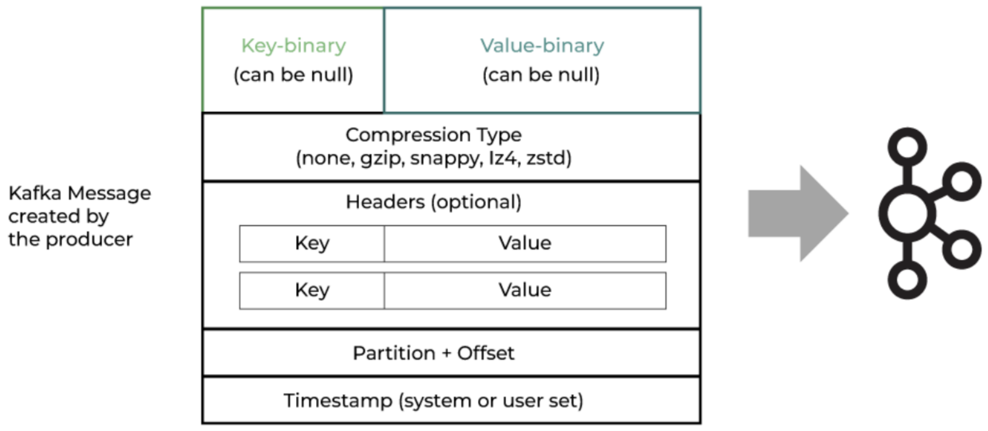
  Con: Rebalance Consumer Group , Rebalance Consumer Rebalance .

- **Topics**

  Virtual Groups or Logs that hold messages and events in a logical order, allowing users to send and receive data between Kafka Servers.

- **Message Header**

Message header could let each message contains more metadata such as the topic, partition, and offset, the format is <Key, Value> pair.


Kafka Message created by the producer
Key-binary (can be null)
Value-binary (can be null)
Compression Type (none, gzip, snappy, lz4, zstd)
Headers (optional)
Key
Value
Key
Value
Partition + Offset
Timestamp (system or user set)

- **Message Partitioner Key**

The message has same key always landing in the same partition, therefore messages are stored in order in partitions.

- **Throttle**

There are two types of throttle that can be implemented.

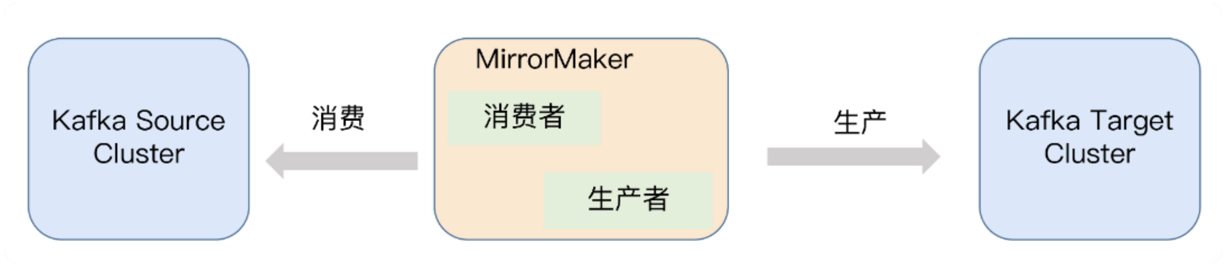
1. From the Producer and Consumer ends, we could user the *clientid* to differentiate the traffic quota.
2. In the broker cluster , we could set up the quota of traffic for follower replica to catch up the Leader replica.

   ref Kafka Replication Quotas during Bootstrapping#TheThrottleMechanism

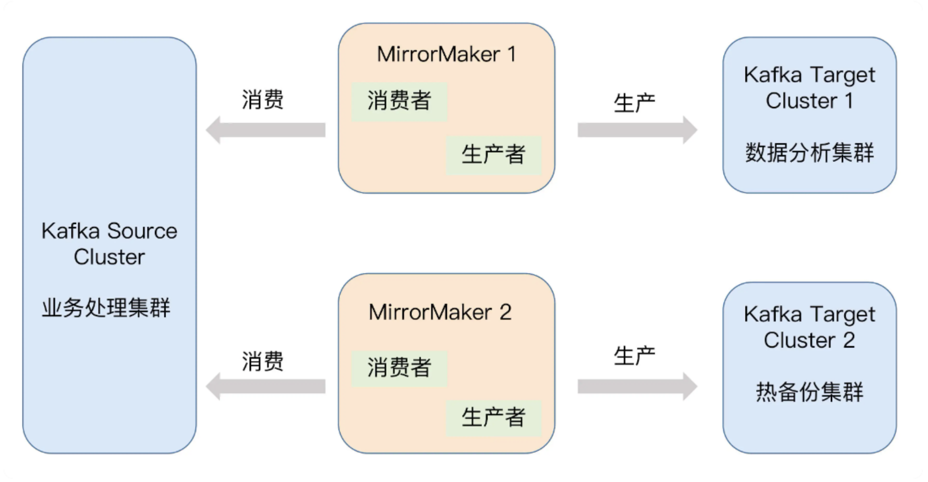
https://cwiki.apache.org/confluence/display/KAFKA/KIP-73+Replication+Quotas

- **Mirror Maker**

  Kafka MirrorMaker is **a stand-alone tool for copying data between two Apache Kafka® clusters**. It is little more than a Kafka consumer and producer hooked together. Data will be read from topics in the original cluster and written to a topic with the same name in the destination cluster.



Multi-cluster usage:

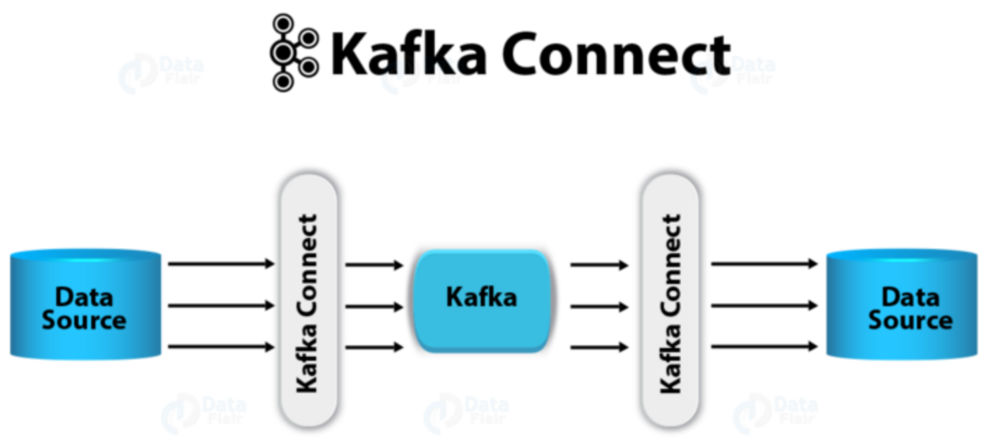


- **Connect**

  Kafka Connect is a framework for connecting Kafka with external systems like databases, key-value stores, search indexes, and file systems, using so-called Connectors.

- **Source connector**: Source connectors ingest entire databases and stream table updates to Kafka topics. Source connectors can also collect metrics from all your application servers and store the data in Kafka topics–making the data available for stream processing with low latency.
- **Sink connector**: Sink connectors deliver data from Kafka topics to secondary indexes, such as ElasticSearch, or batch systems such as Hadoop for offline analysis.
- Connector coordinates a set of **tasks** for piping the data from the source to the target cluster.

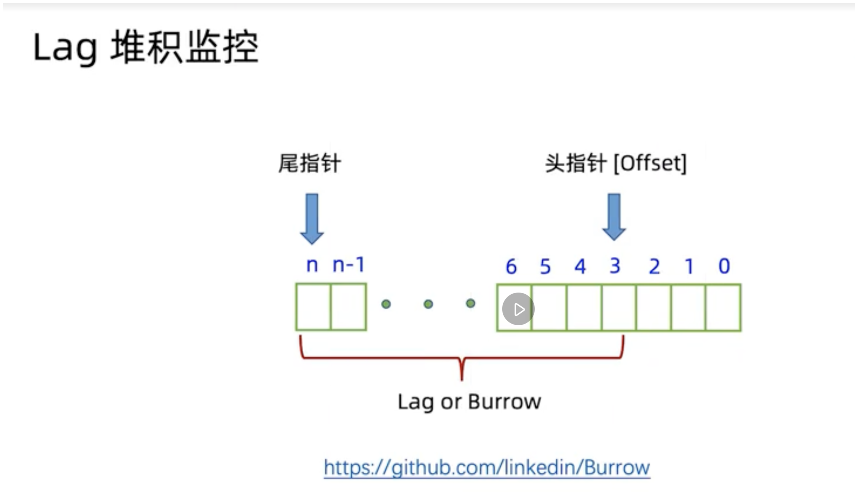
  Example: MySql --Connect ---->Kafka--->Connect-ElasticSearch (**S3**)



- **Consumer Lag**

The Consumer speed can not catch up with the Producer speed.

Kafka Lag Lag Lag

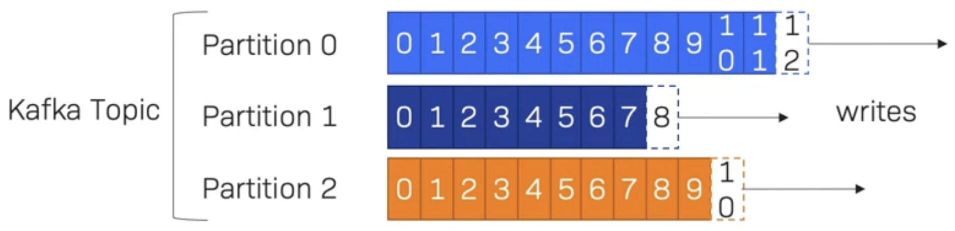


we also can check **records-lag-max , records-lead-min in JMX** for monitoring the consumer speed.

Lead Lag Lead **Lag Lead**

- **Offset   Partition level meaning)**

The offset is **a unique ID assigned to the partitions, which contains messages**. The most important use is that it identifies the messages through ID, which are available in the partitions. In other words, it is a position within a partition for the next message to be sent to a consumer.



__consumer_offset : Consumer Consumer

- **Coordinator**

The group coordinator is nothing but **one of the brokers which receives heartbeats (or polling for messages) from all consumers of a consumer group**. Every consumer group has a group coordinator. If a consumer stops sending heartbeats, the coordinator will trigger a rebalance.

Reference

https://www.projectpro.io/article/apache-kafka-architecture-/442

https://blog.csdn.net/sinat_39809957/article/details/121017220

https://www.confluent.io/blog/hands-free-kafka-replication-a-lesson-in-operational-simplicity/

Parameters adjustion: https://time.geekbang.org/column/article/101763

Consumer Lag:https://time.geekbang.org/column/article/109238