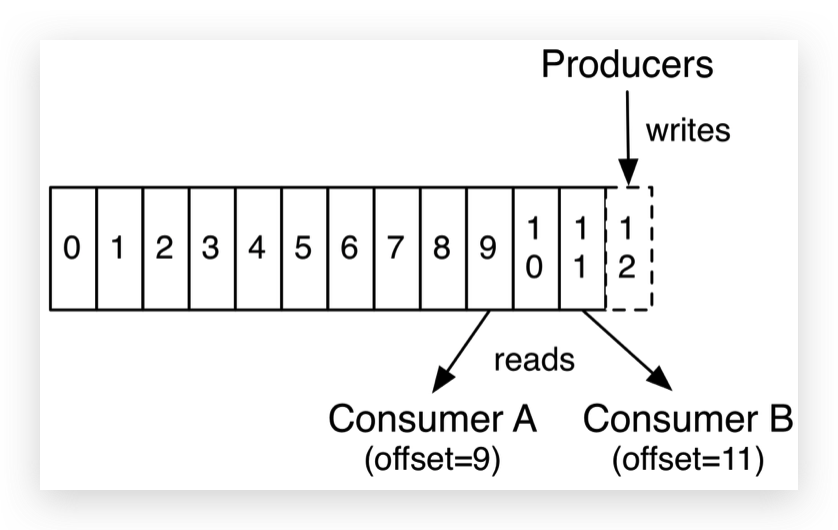
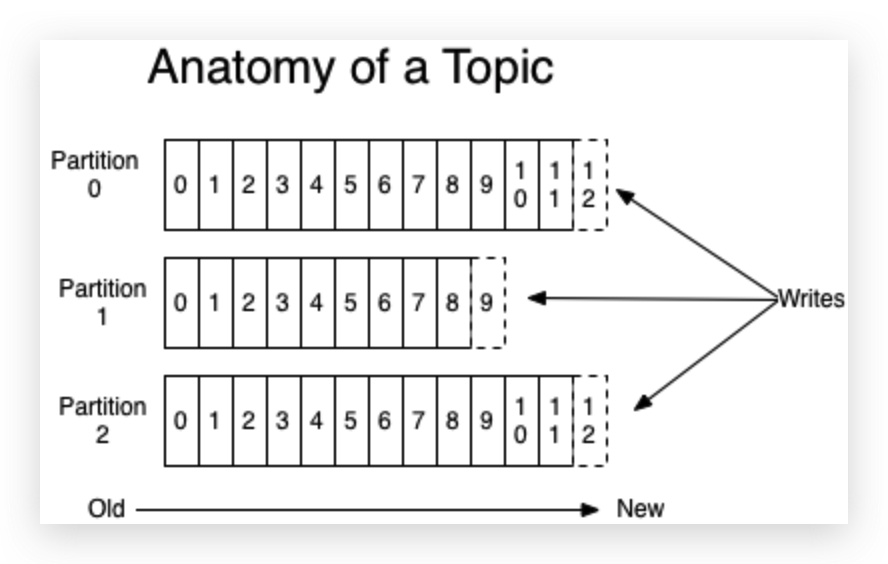
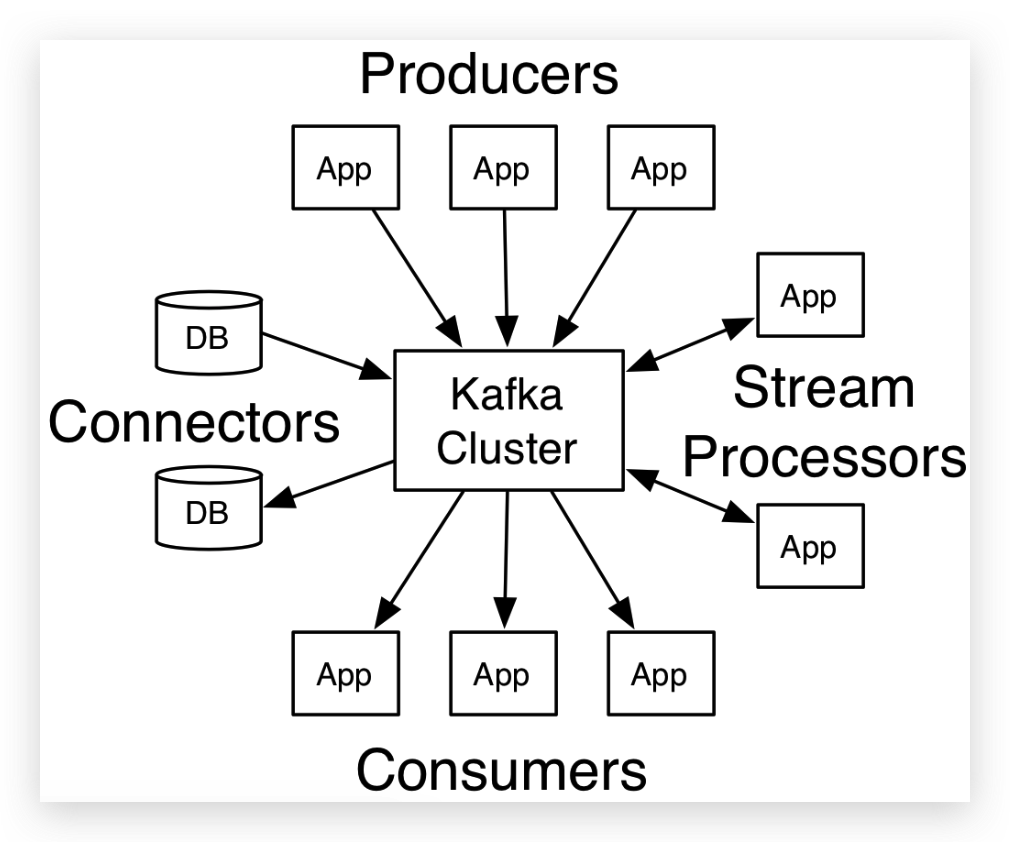
**distributed streaming platform**

消息结构与传输协议的设计

****

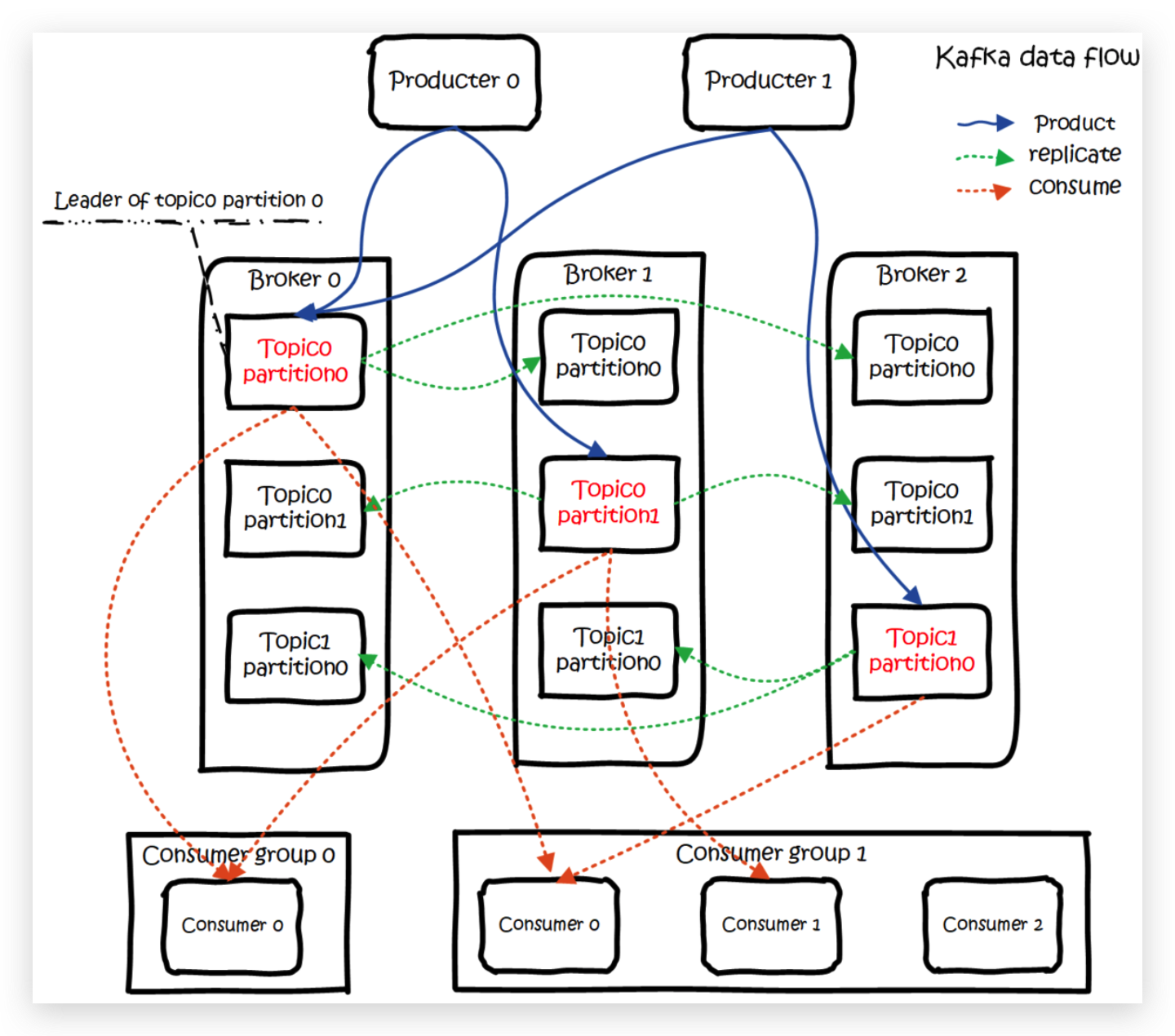
**Broker:** Topic的容器，数据备份。(一个进程/节点)

**Topic:** Partition的容器，数据分区。(一类目录)

**Partition**: an ordered, immutable sequence of records that is continually appended to—a structured commit log. The records in the partitions are each assigned a sequential id number called the *offset* that uniquely identifies each record within the partition. The Kafka cluster durably persists all published records—whether or not they have been consumed—using a configurable retention period. Each partition is replicated across a configurable number of servers for fault tolerance. 有序的连续不可变记录：以offset作为序列号，需配置记录的保留时间、冗余备份的个数。(一个目录)

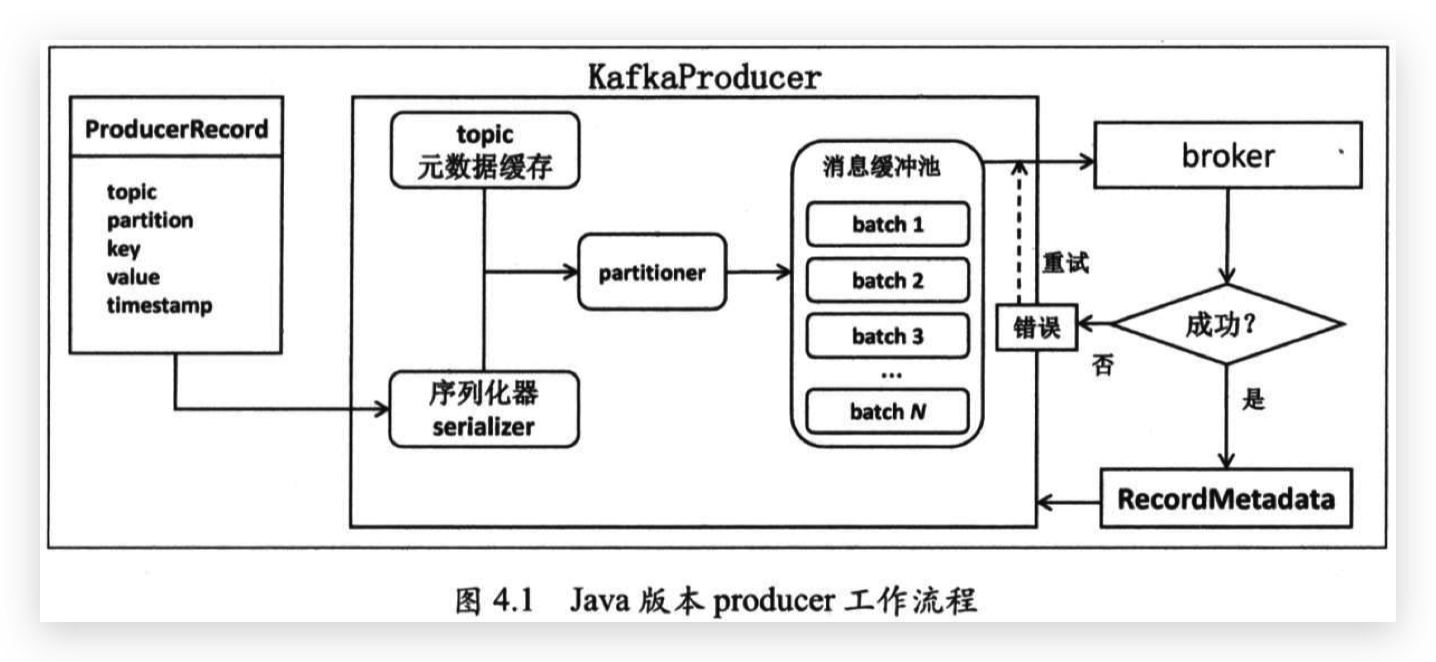
**Segment**:

**Record:** (CRC, version, timestamp, key, value)



**Producer**

1. 消息要素：(topic，key，value)
2. 发送流程：积攒一批消息，再发送



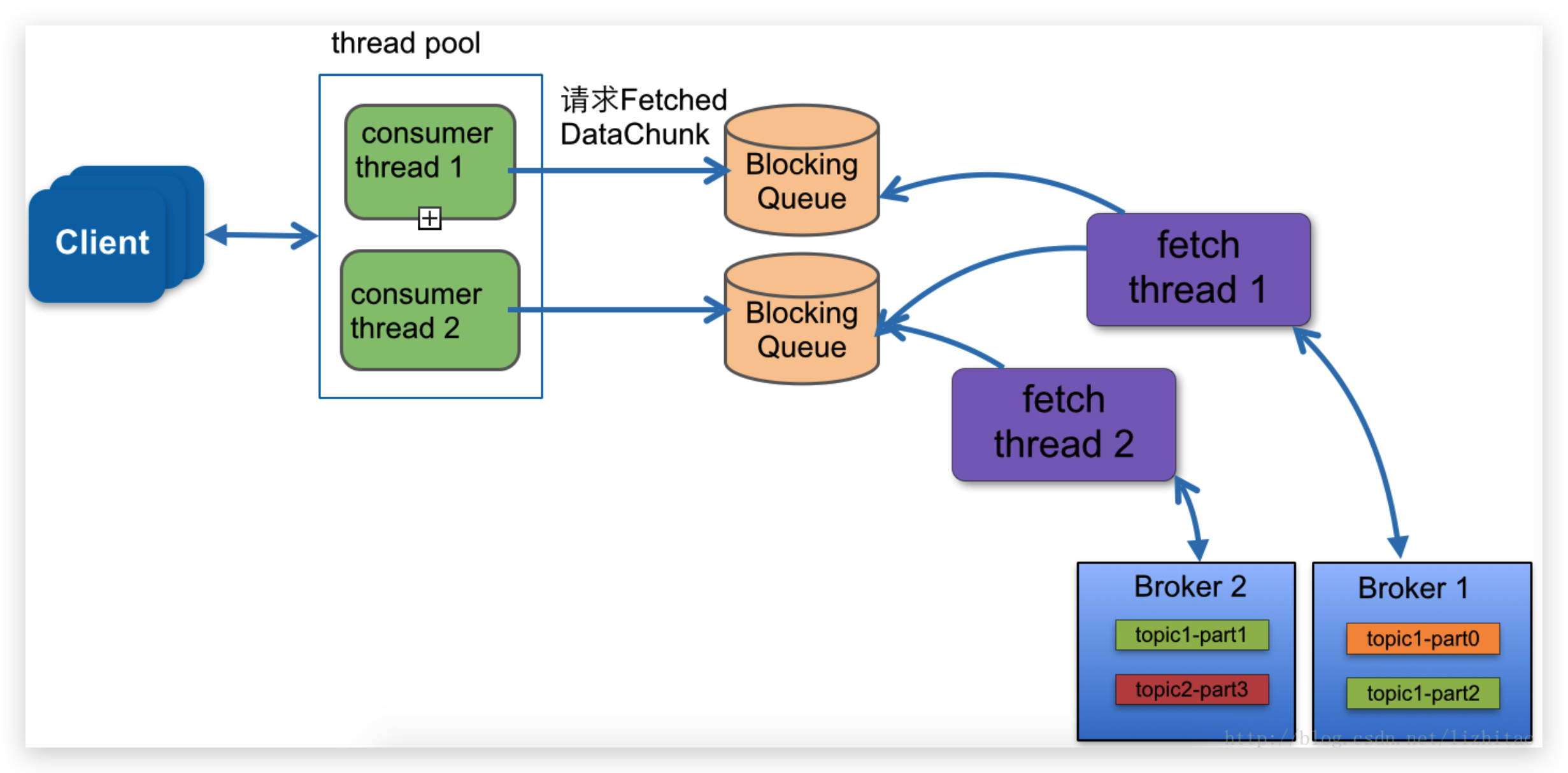
1. 分区策略：指定partition -> 按key哈希 -> round-robin
2. 节点选主：为partition选出一个对外服务的leader replica节点，ISR(in-sync replica)列表作为备用的follower replica节点。
3. 异常处理: RetriableException就重试，还不行就抛异常。结束进程前，需要producer.close()

**Consumer**

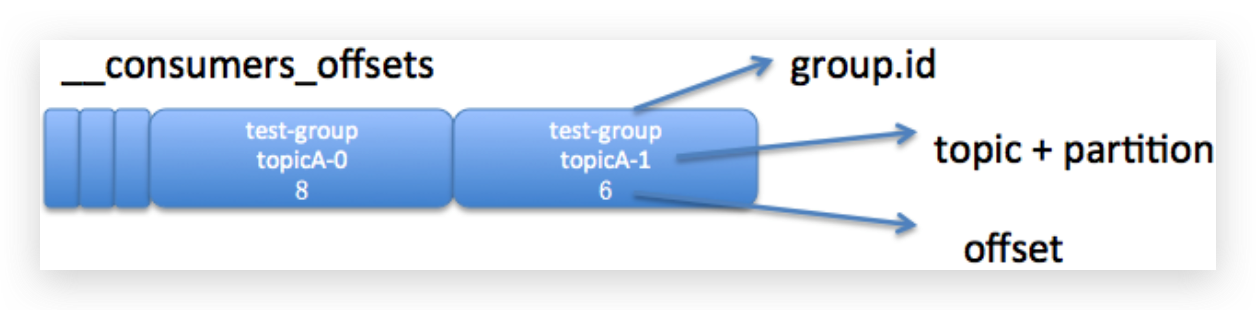
1. 订阅规则：Consumer Group <--> Topic <--> Producer

(Consumer) (Partition)

1. poll(timeout): 有消息立即返回，无消息等待至超时。



1. 位移管理： (group.id, topic, partition) -> offset

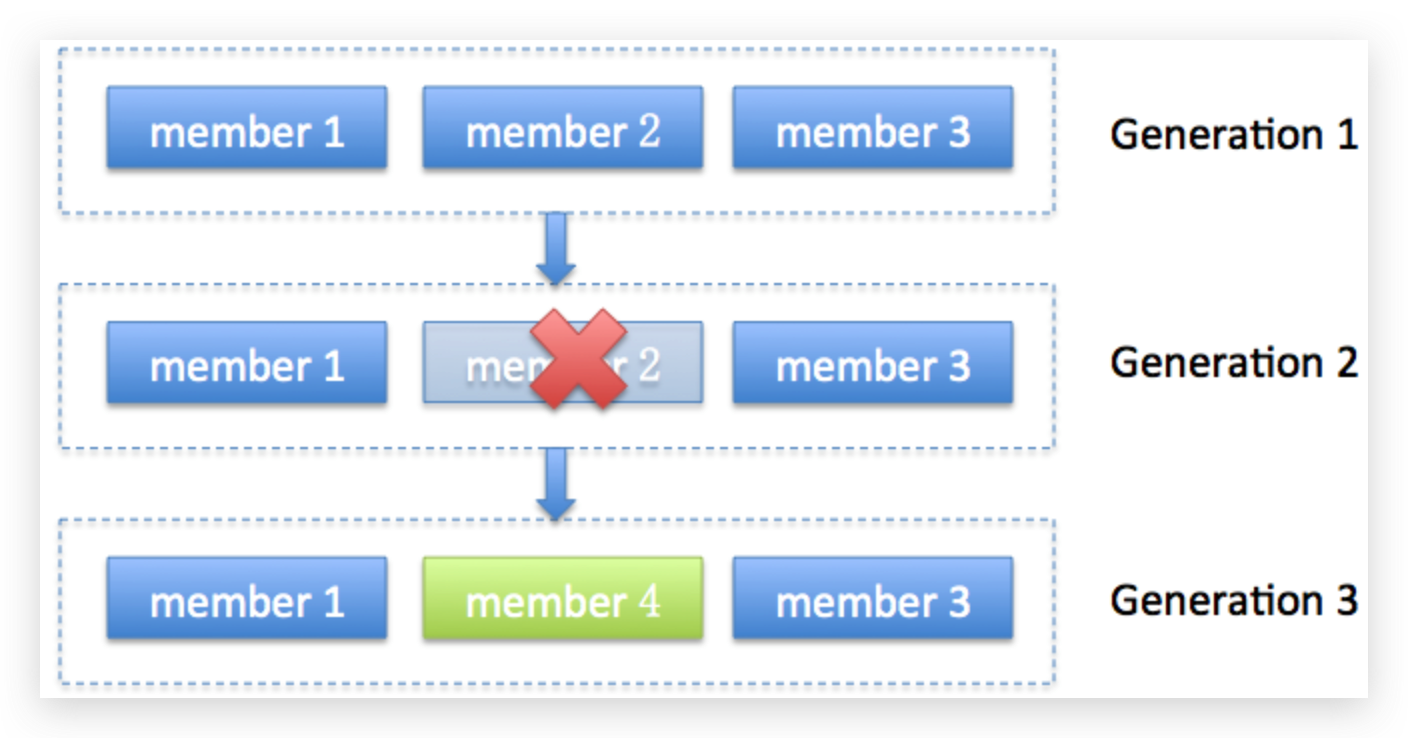
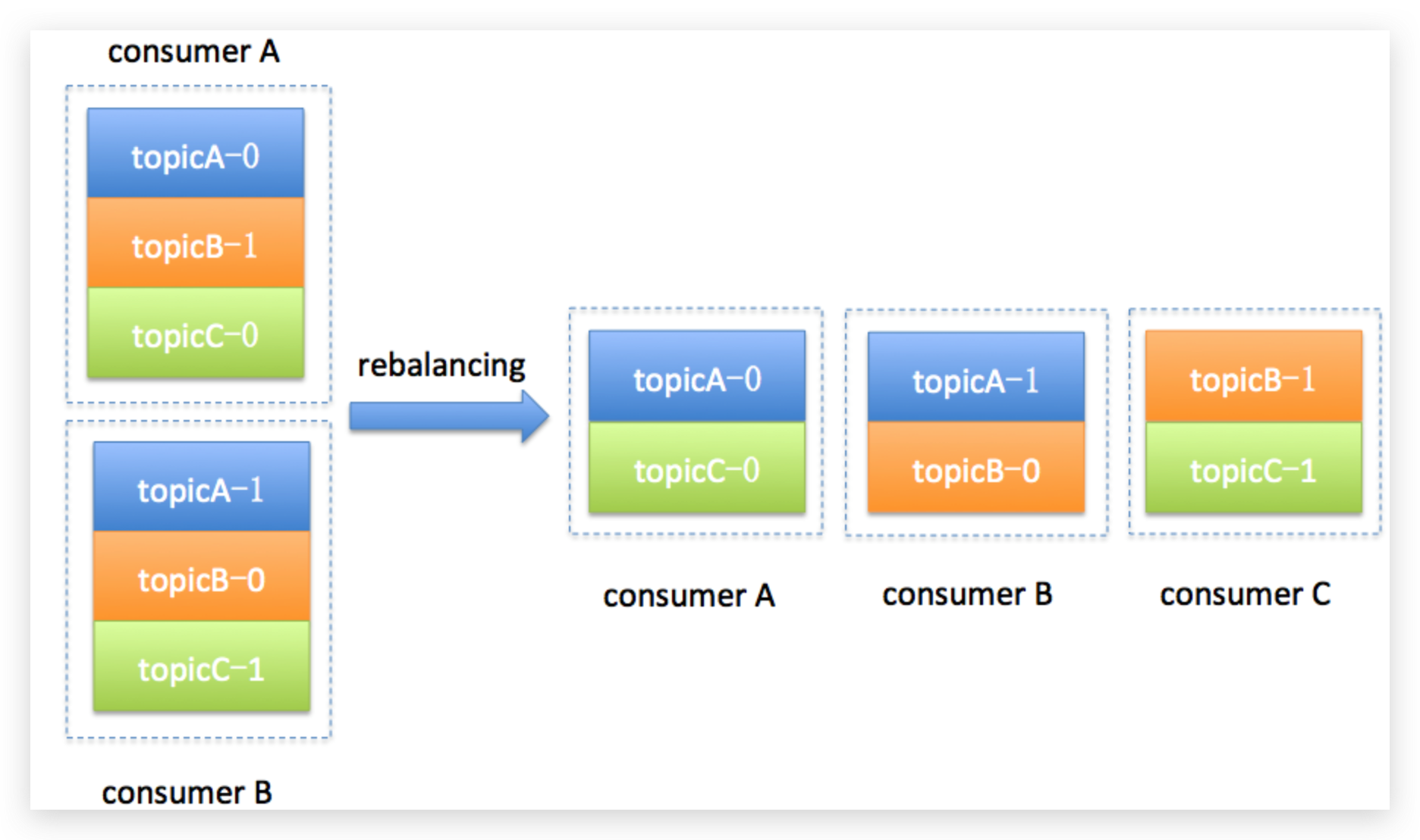
 

|  |  |
| --- | --- |
| Message Delivery Semantic | |
| at most once | 先提交位移，再消费消息。 |
| at least once | 消息消费之后，再提交位移。 |
| exactly once | 消息消费和提交位置在同一个事务中完成。 |

* 自动提交位移：后台周期性提交
* 手动提交位移：commitSync/commitAsync

|  |  |
| --- | --- |
| 位置信息 | |
| last committed offset | 最近一次提交位移的位置 |
| current position | 已读取但未提交的位置 |
| watermark | 已完成副本同步的位置 |
| log end offset | 只写入leader但未完成副本同步的位置 |

1. Consumer Group Rebalance重平衡规则：



* 触发条件：组成员consumer的变更、订阅的topic变更、topic的partition变更。
* coordinator选举：

a. Math.abs(groupID.hashCode) % offsets.topic.num.partitions，求到consumer group位于\_\_consumer\_offsets编号x的分区。

b. 分区x的leader副本所在的broker，即为coordinator(协调者)

* 分区分配策略：range(按区间分段), round-robin(逐个轮询), sticky。
* rebalance generation：每次重平衡后，生成一个连续递增的编号，来限制旧generation的位移提交。

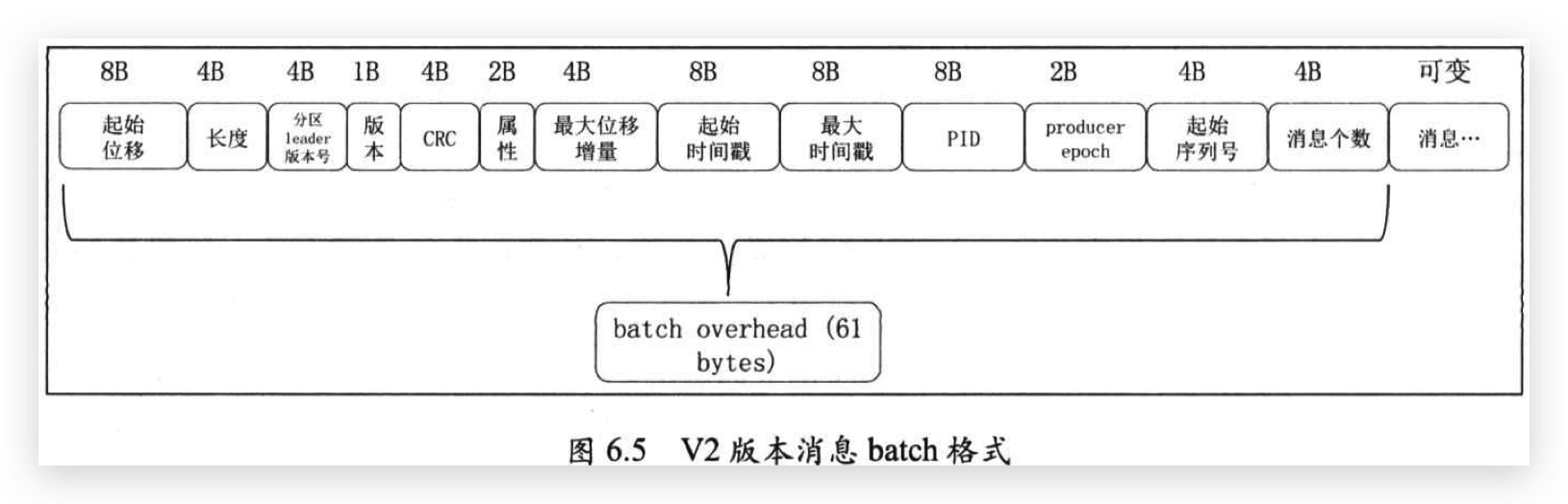
|  |  |
| --- | --- |
| Broker(注册到zk的节点) | |
| broker.id | 集群中的进程节点编号 |
| listeners | 网络监听器，协议://主机名:端口 |
| log.dirs | 消息持久化的目录 |
| zookeeper.connect | zk的节点列表 |
| unclean.leader.  election.enable | 非ISR节点，能否成为leader节点？  默认为false。若为true，则存在消息丢失的可能。 |
| log.retention.ms  log.retention.bytes | 保留消息的最久时间和最大容量，超过则自动清理。 |
| min.insync.replicas | When a producer sets acks to "all" (or "-1"), specifies the minimum number of replicas that must acknowledge a write for the write to be considered successful.  推荐值: min.insync.replicas = replicas - 1 |
| message.max.bytes | 消息的最大容量 |
| replica.fetch.max.bytes | replica同步的最大返回容量 |
| flush.messages | 消息写入文件的批次大小，=1表示同步写入硬盘 |
| Producer | |
| bootstrap.servers | 指定连接的IP:Port |
| key.serializer  value.serializer | 序列化，无默认值 |
| acks | 衡量消息成功提交的标准:  (若写入失败，则回抛异常)  acks=1(default), 等待leader broker返回的ack  acks=0, 不等待leader broker返回的ack  acks=all, 等待所有ISR中的replicas返回的acks |
| buffer.memory | The total bytes of memory the producer can use to buffer records waiting to be sent to the server. |
| max.block.ms | can be blocked either because the buffer is full or metadata unavailable. |
| batch.size | 衡量batch已满的大小 |
| linger.ms | batch未满时，发送的周期 |
| retries | 发生可恢复异常后的重试次数 |
| max.in.flight.requests  .per.connection | The maximum number of unacknowledged requests the client will send on a single connection before blocking. |
| compression.type | 消息的压缩类型，默认不压缩 |
| max.request.size | producer请求broker的大小 |
| request.timeout.ms | producer请求broker的超时时间 |
| partitioner.class | 自定义分区规则 |
| interceptor.classes | 拦截器链，默认空to intercept the records received by the producer before they are published |
| compression.type | 压缩类型：CPU(时间)换IO(空间)，权衡实际。 |
| Consumer | |
| group.id | consumer group id |
| key.deserializer  value.deserializer | 反序列化，无默认值 |
| session.timeout.ms | Consumer Group接收Consumer周期性心跳消息的超时时间。超时无响应，则触发rebalance。 |
| heartbeat.interval.ms | Consumer向Consumer Group发送心跳的间隔  建议: < 1/3 \* session.timeout.ms |
| max.poll.interval.ms | The maximum delay between invocations of poll(). 超过最大延迟，则触发rebalance。 |
| connections.max.idle.ms | Socket连接的最大空闲时间，超时关闭。 |
| auto.offset.reset | 找不到正确的位移时，应对策略：earliest, latest |
| enable.auto.commit | 是否自动周期性地提交位移 |
| fetch.max.bytes | 能返回的消息集合的大小，  建议: > message.max.bytes \* n |
| max.poll.records | 一次poll()的最大消息数 |

**设计理念**

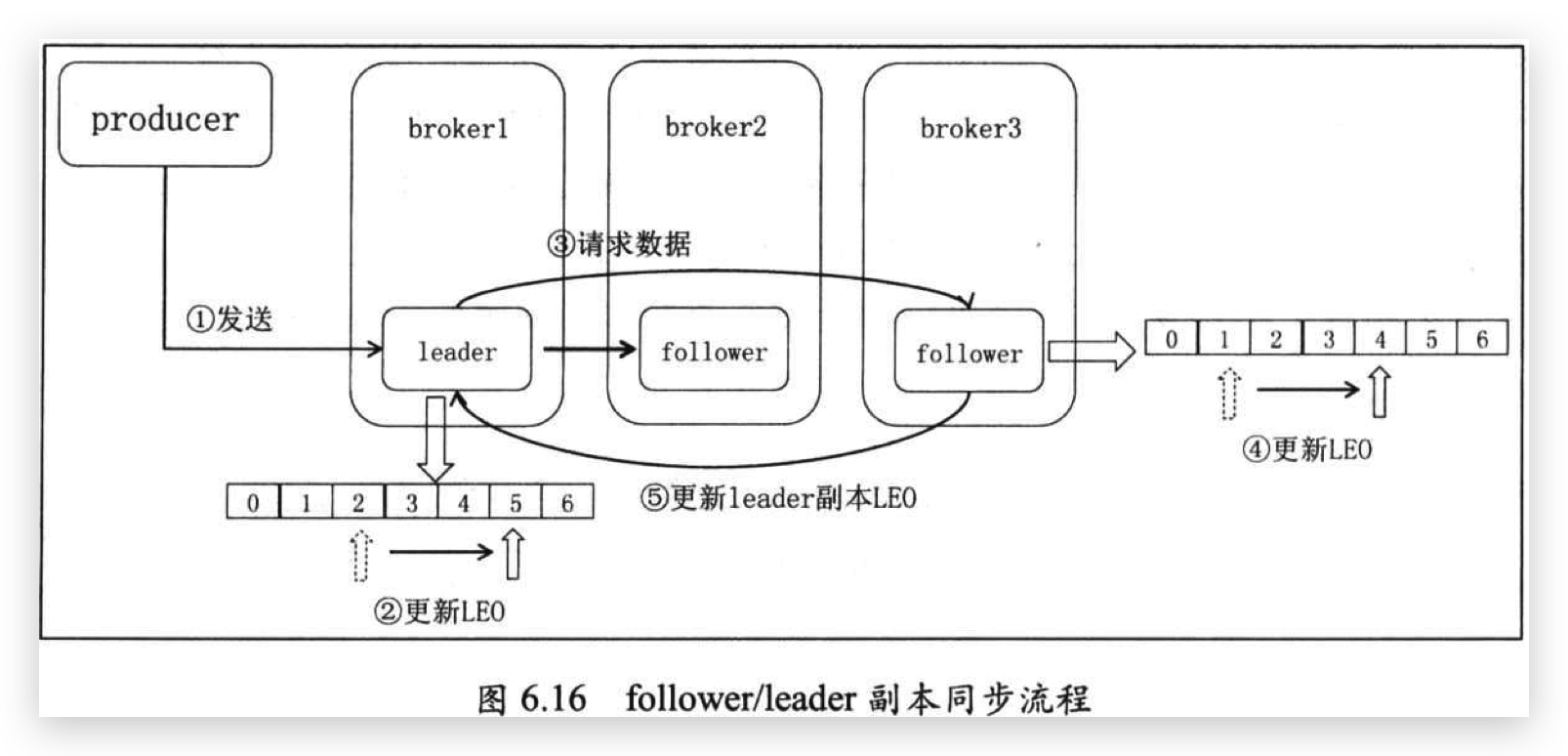
1. broker

* 消息结构：将Java Bean转换成紧凑的ByteBuffer存储。





* 持久化：
* 集群管理：基于zookeeper的ephemeral node实现失效转移。
* 数据备份：leader负责所有的对外读写，follower只是备份数据，争取进入ISR



**运维**

1. 列出所有topic



1. 查看topic详情



3. 查看brokers在zk中的状态

