kafka如何复制
kafka如何处理来自消费者和生产者的请求
kafka的存储细节，比如文件格式和索引

1.复制
kafka使用主题管理数据，每个主题若干个分区，每个分区多个副本。
副本包括：
1.首领副本：每个分区有且仅有一个。1. 所有生产者和消费者的请求都需要经过这个副本 2.搞清楚有哪些跟随者和自己状态一致
2.跟随者副本：不处理需求，只负责从首领复制消息，如果首领奔溃，提升为新首领

跟随者如何与首领保持同步？：
为了与首领保持同步，跟随者向首领发送获取数据的请求，这种请求与悄费者为了读取悄息而发送的请求是一样的。首领将响应消息发给跟随者。请求消息里包含了跟随者想要获取消息的偏移量，而且这些偏移量总是有序的。
比如一个跟随者可能会按序请求消息1，消息2，消息3，在收到这3个消息的返回时是不会发送第四个请求的。如果发送第四个请求首领就知道跟随者收到了前面3个请求的响应。通过查看跟随者请求的最新偏移量，首领可以知道该跟随者的复杂进度。如果跟随者10s（replica.lag.time.max.ms）内没有发送请求，或者虽然在请求消息，但是10s（replica.lag.time.max.ms）内没有请求最新的数据，这个跟随者就是不同步的。
同步的跟随者被称为**同步副本**，只有同步的副本能被选为新首领。
首选首领：创建主题时候选定的首领就是首选首领。在创建分区的时候，首选首领选出来考虑到了broker之间的均衡，我们希望首选首领成为首领时，broker的负载均衡会得到均衡。auto.leader.rebalance.enable=true会检查当前首领是不是首选首领，如果不是，并且该副本是同步的，就会让首选首领选举成为首领。

2.处理请求
broker 会在它所监听的每一个端口上运行一个acceptor线程，这个钱程会创建一个连接，并把它交给processor线程去处理。processor线程（也被叫作"网络线程"）的数量是可配置的。网络线程负责从客户端获取请求悄息，把它们放进请求队列，然后从响应队列获取响应消息，把它们发送给客户端。请求消息被放到请

求队列后，IO 线程会负责处理它们。

2.1 客户端发送请求前提：元数据请求

客户端先发送元数据请求到broker(每个broker都会缓存这些信息）或者一般情况下客户端本身就缓存了这些信息（间隔一段时间（meta.max.age.ms）刷新一次）。在新broker 加入集群时，或者如果客户端收到"非首领"错误，客户端会在尝试重发请求之前先刷新元数据

2.2 生产请求：生产者发送到请求，包含客户端需要写入到信息

1）包含首领副本的broker收到生产者请求后，先做如下验证

发送数据的用户是否拥有主题写入权限?

请求里包含的acks值是否有效?（只允许出现0、1、all）

如果acks=all，是否有足够多的同步副本保证消息已经被完全写入?（可以对broker进行配置，如果同步数量不总，broker可以拒接处理新消息）

2）消息被写入本地磁盘。在Linux系统上，消息会被写到文件系统缓存里，并不保证何时会被刷新到磁盘上，Kafka不会一直等待数据被写到磁盘上——它依赖复制功能来保证消息的持久性。

3）在消息被写入分区首领之后，broker开始检查acks配置参数——如果acks被设为0或1，那么broker立即返回响应；如果acks被设为all，那么请求会被保存在一个叫做炼狱(purgatory)的缓冲区，直到首领发现所有跟随者副本都复制了消息，响应才会返回给客户端。

2.3 获取请求：消费者和跟随者需要从broker读取消息时发送的请求

1)客户端发送获取请求，向Broker请求主题分区里具有**特定偏移量**的消息

2)获取请求需要先到达指定的**分区首领**上，然后客户端通过**查询元数据**来确保请求的路由是正确的

3)分区首领在收到获取请求时，分区首领首先会检查获取请求是否有效（例如指定的偏移量在分区上是否存在）

4)如果请求的偏移量存在，Broker将按照**客户端指定的数量上限**从分区里读取消息，再把消息返回给客户端

5)客户端除了可以设置Broker返回数据的上限外，还可以设置**下限以及设置一个超时等待时间**

注意：在首领副本没有把消息同步到所有的同步副本的时候，这些没被同步的数据是不会被客户端读取的，因为这样是不安全的，如果首领奔溃，新副本成为首

领，这些数据就丢失了，如果允许消费者读取数据，就破坏了一致性。比如一个消费者读取了这么一个消息，另外一个消费者却发现这消息不存在。

2.4 其他请求

3.物理存储

管理员指定来一个用于存储分区的目录清单，也就是**log.dirs**(这个值不是日志的路径，日志路径由**log4j.properties**指定)

a)数据是如何被分配到集群的broker上以及broker的目录里的。

b)broker如何管理这些文件，特别是如何进行数据保留的

c)了解文件和索引格式

d)**日志压缩**及工作原理

3.1 分区分配

1.为分区和副本选择broker:随机选择一个broker,使用轮询给每个broker分配分区来确定首领分区的位置（一种是按照数字顺序，如果设置了机架就按照交替机架来选择）

2.为分区选择目录：单独为每个分区分配目录，规则是计算每个分区的目录数量，新的分区总添加到数量最小的那个目录

3.2文件管理

kafka管理员可以设置每个主题的消息保留期限。

因为大文件查找和删除消息很消耗时间，所以分区被分成若干个**片段**，每个片段默认1GB或者一周数据。broker往分区写入数据时，达到了上限，那么就关闭该片段，打开一个新文件

正在写入的数据叫做活跃片段。活跃片段永远不会被删除。

3.3 文件格式

我们把kafka的消息和偏移量保存在文件中。格式和生产者发送过来或者消费者接受的格式一样，这样kafka可以零复制给消费者发消息，同时避免对生产者已经压缩过的消息再次压缩和解压。

消息包含：键，值，偏移量，消息大小，校验和，消息格式版本号，压缩算法（Snappy,GZip,LZ4），时间戳(生产者发送消息的时间或者broker接受到消息的时间)

DumpLogSegment：kafka自带工具，查看偏移量，消息大小，校验和，消息格

式版本号，压缩算法

bin/kafka-run-class.sh kafka.tools.DumpLogSegment

3.4 索引

为了帮助broker更快地定位到指定的偏移量，Kafka为每个分区维护了一个索引。索引把偏移量映射到片段文件和偏移量在文件里的位置。

索引也分成片段。

3.5 清理

Kafka通过改变主题的保留策略来满足这些使用场景。

3.6 清理工作原理

每个日志片段分成2部分：

干净的部分：这些消息之前被清理过，每个键只有一个对应的值。

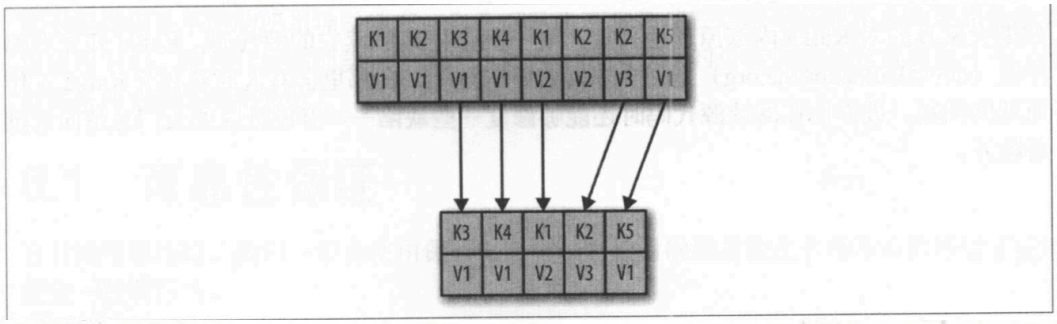
污浊的部分：消息是在上一次清理后写入的

清理过程

1）每个broker启动一个清理管理器线程和多个清理线程

2)清理线程读取分区污浊的部分，创建map，map里每个元素包括了消息键的散列值（16B）和消息的偏移量（8B），这也意味着，1G的日志片段，每个消息1KB，也就是1，000，000个数据，map 24MB就够了。

3）清理线程在创建好偏移盐 map后，开始从干净的片段处读取消息，从最旧的消息开始，把 它们的 内容与 map 里的内容进行比对。它会检查消息的键是否存在于 map 中，如果不存在， 那么说 明消息的值是最新的，就把消息复制到替换片段上 。 如果键已存在，消息会被忽略， 因为在分区 的后部已经有一个具有相同键的消息存在。在复制完所有的消息之后，我们就将 替换片段与原始片段进行交换，然后开始清理下一个片段。完成整个清理过程之后，每个键 对应一个不同的消息一寸主些消息的值都是最新的。



3.7 彻底清理

生产者发送值为null的消息，kafka会保留一段时间该消息，确保消费者能收到这个消息。