1.Kafka 的设计时什么样的呢？
Kafka 将消息以 topic 为单位进行归纳
将向 Kafka topic 发布消息的程序成为 producers.
将预订 topics 并消费消息的程序成为 consumer.
Kafka 以集群的方式运行，可以由一个或多个服务组成，每个服务叫做一个 broker.
producers 通过网络将消息发送到 Kafka 集群，集群向消费者提供消息

2.数据传输的事物定义有哪三种？
数据传输的事务定义通常有以下三种级别：
（1）最多一次: 消息不会被重复发送，最多被传输一次，但也有可能一次不传输
（2）最少一次: 消息不会被漏发送，最少被传输一次，但也有可能被重复传输.
（3）精确的一次（Exactly once）： 不会漏传输也不会重复传输,每个消息都传输被一次而且仅仅被传输一次，这是大家所期望的

3.Kafka 判断一个节点是否还活着有那两个条件？
（1）节点必须可以维护和 ZooKeeper 的连接，Zookeeper 通过心跳机制检查每个节点的连接
（2）如果节点是个 follower,他必须能及时的同步 leader 的写操作，延时不能太久

4.producer 是否直接将数据发送到 broker 的 leader(主节点)？
producer 直接将数据发送到 broker 的 leader(主节点)，不需要在多个节点进行分发，为了帮助 producer 做到这点，所有的 Kafka 节点都可以及时的告知:哪些节点是活动的，目标 topic 目标分区的 leader 在哪。这样 producer 就可以直接将消息发送到目的地了

5、Kafa consumer 是否可以消费指定分区消息？
Kafa consumer 消费消息时，向 broker 发出"fetch"请求去消费特定分区的消息，consumer 指定消息在日志中的偏移量（offset），就可以消费从这个位置开始的消息，customer 拥有了 offset 的控制权，可以向后回滚去重新消费之前的消息，这是很有意义的

6、Kafka 消息是采用 Pull 模式，还是 Push 模式？
Kafka 最初考虑的问题是，customer 应该从 brokes 拉取消息还是 brokers 将消息推送到 consumer，也就是 pull 还 push。在这方面，Kafka 遵循了一种大部分消息系统共同的传统的设计：producer 将消息推送到 broker，consumer 从 broker 拉取消息

一些消息系统比如 Scribe 和 Apache Flume 采用了 push 模式，将消息推送到下游的 consumer。这样做有好处也有坏处：由 broker 决定消息推送的速率，对于不同消费速率的 consumer 就不太好处理了。消息系统都致力于让 consumer 以最大的速率最快速的消费消息，但不幸的是，push 模式下，当 broker 推送的速率远大于 consumer 消费的速率时，consumer 恐怕就要崩溃了。最终 Kafka 还是选取了传统的 pull 模式

Pull 模式的另外一个好处是 consumer 可以自主决定是否批量的从 broker 拉取数据。Push 模式必须在不知道下游 consumer 消费能力和消费策略的情况下决定是立即推送每条消息还是缓存之后批量推送。如果为了避免 consumer 崩溃而采用较低的推送速率，将可能导致一次只推送较少的消息而造成浪费。Pull 模式下，consumer 就可以根据自己的消费能力去决

定这些策略

Pull 有个缺点是，如果 broker 没有可供消费的消息，将导致 consumer 不断在循环中轮询，直到新消息到 t 达。为了避免这点，Kafka 有个参数可以让 consumer 阻塞知道新消息到达(当然也可以阻塞知道消息的数量达到某个特定的量这样就可以批量发

7.Kafka 存储在硬盘上的消息格式是什么？
消息由一个固定长度的头部和可变长度的字节数组组成。头部包含了一个版本号和 CRC32 校验码。
- 消息长度: 4 bytes (value: 1+4+n)
- 版本号: 1 byte
- CRC 校验码: 4 bytes
- 具体的消息: n bytes

8.Kafka 高效文件存储设计特点：
(1).Kafka 把 topic 中一个 parition 大文件分成多个小文件段，通过多个小文件段，就容易定期清除或删除已经消费完文件，减少磁盘占用。
(2).通过索引信息可以快速定位 message 和确定 response 的最大大小。
(3).通过 index 元数据全部映射到 memory，可以避免 segment file 的 IO 磁盘操作。
(4).通过索引文件稀疏存储，可以大幅降低 index 文件元数据占用空间大小。

9.Kafka 与传统消息系统之间有三个关键区别
(1).Kafka 持久化日志，这些日志可以被重复读取和无限期保留
(2).Kafka 是一个分布式系统：它以集群的方式运行，可以灵活伸缩，在内部通过复制数据提升容错能力和高可用性
(3).Kafka 支持实时的流式处理

10.Kafka 创建 Topic 时如何将分区放置到不同的 Broker 中
- 副本因子不能大于 Broker 的个数；
- 第一个分区（编号为 0）的第一个副本放置位置是随机从 brokerList 选择的；
- 其他分区的第一个副本放置位置相对于第 0 个分区依次往后移。也就是如果我们有 5 个 Broker，5 个分区，假设第一个分区放在第四个 Broker 上，那么第二个分区将会放在第五个 Broker 上；第三个分区将会放在第一个 Broker 上；第四个分区将会放在第二个 Broker 上，依次类推；
- 剩余的副本相对于第一个副本放置位置其实是由 nextReplicaShift 决定的，而这个数也是随机产生的

11.Kafka 新建的分区会在哪个目录下创建
在启动 Kafka 集群之前，我们需要配置好 log.dirs 参数，其值是 Kafka 数据的存放目录，这个参数可以配置多个目录，目录之间使用逗号分隔，通常这些目录是分布在不同的磁盘上用于提高读写性能。
当然我们也可以配置 log.dir 参数，含义一样。只需要设置其中一个即可。
如果 log.dirs 参数只配置了一个目录，那么分配到各个 Broker 上的分区肯定只能在这个目录下创建文件夹用于存放数据。

但是如果 log.dirs 参数配置了多个目录，那么 Kafka 会在哪个文件夹中创建分区目录呢？答案是：Kafka 会在含有分区目录最少的文件夹中创建新的分区目录，分区目录名为 Topic 名+分区 ID。注意，是分区文件夹总数最少的目录，而不是磁盘使用量最少的目录！也就是说，如果你给 log.dirs 参数新增了一个新的磁盘，新的分区目录肯定是先在这个新的磁盘上创建直到这个新的磁盘目录拥有的分区目录不是最少为止。

12.partition 的数据如何保存到硬盘
topic 中的多个 partition 以文件夹的形式保存到 broker，每个分区序号从 0 递增，
且消息有序
Partition 文件下有多个 segment（xxx.index，xxx.log）
segment 文件里的 大小和配置文件大小一致可以根据要求修改 默认为 1g
如果大小大于 1g 时，会滚动一个新的 segment 并且以上一个 segment 最后一条消息的偏移量命名

13.kafka 的 ack 机制
request.required.acks 有三个值 0 1 -1
0:生产者不会等待 broker 的 ack，这个延迟最低但是存储的保证最弱当 server 挂掉的时候就会丢数据
1：服务端会等待 ack 值 leader 副本确认接收到消息后发送 ack 但是如果 leader 挂掉后他不确保是否复制完成新 leader 也会导致数据丢失
-1：同样在 1 的基础上 服务端会等所有的 follower 的副本受到数据后才会受到 leader 发出的 ack，这样数据不会丢失

14.Kafka 的消费者如何消费数据
消费者每次消费数据的时候，消费者都会记录消费的物理偏移量（offset）的位置
等到下次消费时，他会接着上次位置继续消费

15.消费者负载均衡策略
一个消费者组中的一个分片对应一个消费者成员，他能保证每个消费者成员都能访问，如果组中成员太多会有空闲的成员

16.数据有序
一个消费者组里它的内部是有序的
消费者组与消费者组之间是无序的

17.kafaka 生产数据时数据的分组策略
生产者决定数据产生到集群的哪个 partition 中
每一条消息都是以（key，value）格式
Key 是由生产者发送数据传入
所以生产者（key）决定了数据产生到集群的哪个 partition