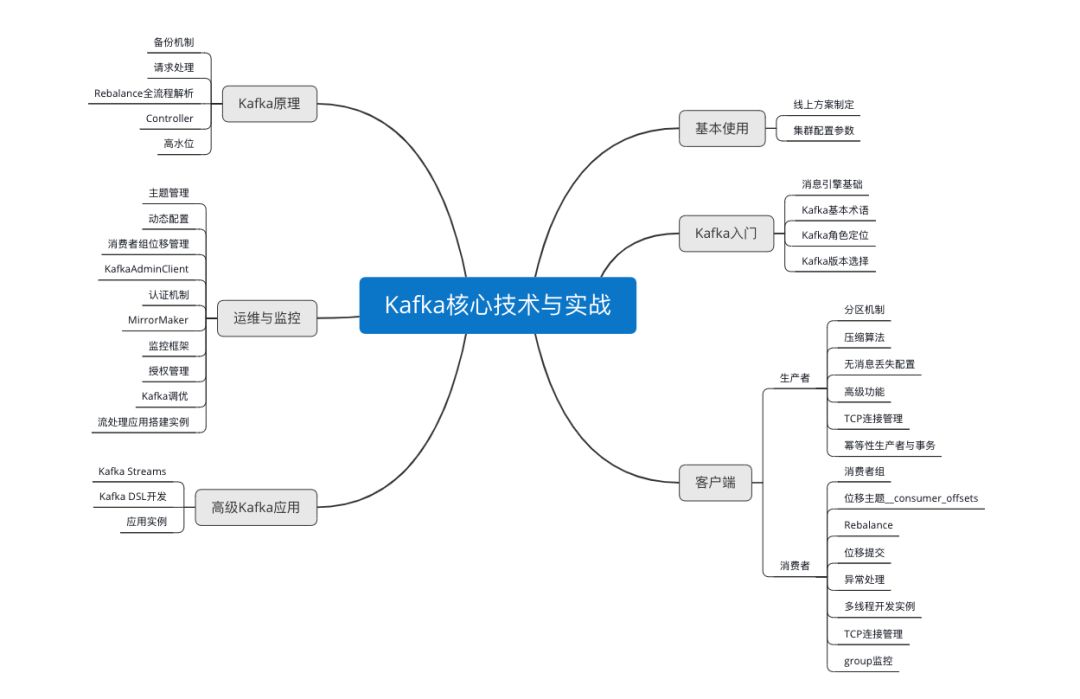
# Kafka基础

Kafka 核心技术：我们仅需要学习一套框架，就能在实际业务系统中实现消息队列应用、应用程序集成、分布式存储构建，甚至是流处理应用的开发与部署



极客时间新上线的《Kafka 核心技术与实战》

## Kafka消费组

什么是消费者组

consumer group是kafka提供的可扩展且具有容错性的消费者机制,组内的所有消费者协调在一起来消费订阅主题(subscribed topics)的所有分区(partition)。当然，每个分区只能由同一个消费组内的一个consumer来消费。

消费者在消费的过程中需要记录自己消费了多少数据，即消费位置信息。在Kafka中这个位置信息有个专门的术语：位移(offset)。

Kafka默认是定期帮你自动提交位移的(enable.auto.commit = true)，你当然可以选择手动提交位移实现自己控制。另外kafka会定期把group消费情况保存起来，做成一个offset map，

老版本的位移是提交到zookeeper中的，新版本的kafka提供了另一种解决方案：增加\_\_consumeroffsets topic，将offset信息写入这个topic，摆脱对zookeeper的依赖(指保存offset这件事情)。\_\_consumer\_offsets中的消息保存了每个consumer group某一时刻提交的offset信息。

# Kafka架构

## Kfaka如何应对大量的长连接

Kafka遇到的问题：应对大量客户端连接（kafka会与生产者消费者建立大量的长连接，及其耗内存）

针对这个问题，大名鼎鼎的Kafka采用的架构策略是Reactor多路复用模型。

简单来说，就是搞一个acceptor线程，基于底层操作系统的支持，实现连接请求监听。

如果有某个设备发送了建立连接的请求过来，那么那个线程就把这个建立好的连接交给processor线程。

每个processor线程会被分配N多个连接，一个线程就可以负责维持N多个连接，他同样会基于底层操作系统的支持监听N多连接的请求。

如果某个连接发送了请求过来，那么这个processor线程就会把请求放到一个请求队列里去。

接着后台有一个线程池，这个线程池里有工作线程，会从请求队列里获取请求，处理请求，接着将请求对应的响应放到每个processor线程对应的一个响应队列里去。

最后，processor线程会把自己的响应队列里的响应发送回给客户端。

说了这么多，还是来一张图，大家看下面的图，就可以理解上述整个过程了。

