旧的高级consumer依靠zk管理，而新的会使用kafka自带的分组协议。在协议中，broker中的一个将会被设置为协调者（coordinator），它负责管理组内成员以及为它们分配分区。

Coordinator是从每个topic的partition的leader中选出来的。这样cluster中每个broker上的coordinator数量大致相等，这样可以通过增加broker来扩展分组。

Consumer加入group之前需要向coordinator发送请求，然后coordinator进行partition分配（rebalance）产生新组。组内成员必须向coordinator发送heartbeats，如果coordinator没有在session timeout配置过期之前（超时）收到heartbeats，那么这个成员就会被清除出组，coordinator会重新分配资源。

Offset Management:每个成员都需要确定被分配到的partition的初始位置。在group第一次创建，还没有消息写入时，由auto.offset.reset规定初始位置，通常从最早或者最迟的offset开始。

Consumer read的过程中需要提交已读消息的offset，一旦crash，它的partition会分配给别的consumer，新consumer会从最后一次提交的offset开始read。如果没有任何offset就crash了，那么新consumer会按照上文reset初始化来接管parttion。

默认情况下consumer在指定的时间段内会自动commit，也有支持手动的API。

设置

核心设置：

bootstrap.servers：连接kafka cluster

client.id:可以通过其知道是broker中的哪个client发出的请求，同一组中的所有consumer共享一个client.id

组设置：

group.id

session.timeout.ms：会话超时的时间设置，默认30s。

heartbeat.interval.ms：consumer向coordinator发送心跳的频率。

Offset管理设置 ：

enable.auto.commit：设置是否自动提交，默认是。

auto.commit.interval.ms：自动提交的时间周期。默认5s。

auto.offset.reset：当没有提交或者提交的offset溢出时，这个设置可以选择reset commit的位置，从最早或最晚（默认）的offset开始。也可以设置为none，如果你想手动设置位置的话。

初始化

librdkafka需要通过事先设置好来直接处理配置产生的错误。

Basic Usage

librdkafka与java采用完全不同的方式处理线程模型和消费者活跃度

librdkafka采用多线程处理kafka消费。它所有的抓取和与coordinator的沟通都通过后台线程。但是client关闭了之后后台线程使consumer继续存活，需要手动确认它不是一个持有partition的zombie process。

poll()：从broker循环检索记录

subscribe()：poll过程中从哪个topic fetch

用完consumer记得调用close()。

同步提交

commitSync()：手动提交，这个方法将会一直阻塞直到提交成功

在librdkafka中，使用rd\_kafka\_commit()，用来同步和异步提交。rd\_kafka\_consumer\_poll返回单个消息而不是像java consumer返回一批。

传输保证

消息传递至多一次，不重复消费。

doCommitSync()：返回commit是否成功

fetch.min.bytes()：控制每次fetch返回的数据量。

Broker将保持fetch直到获取足够的可用数据（或者fetch.max.wait.ms时限到）但是这可能导致重复。

异步提交

不等请求完成，consumer就立即发送请求并返回。

保证了只提交一次，但是它的缺点在于不能重试也不能保证提交顺序，可以通过回调来重试提交，但还是需要面对顺序的问题。

为了防止提交失败往往采取同步异步结合，可是这样还是有可能产生重复消费。

ConsumerRebalanceListener()：与重平衡行为挂钩

Producer设置

bootstrap.servers：同consumer

持久化设置

acks：默认为1，从leader确认成功写入kafka。=all时保证不止leader并且所有replicas都成功同步备份。=0时吞吐量最大化，但是无法保证数据成功写入kafka broker因为在此时它不会返回响应。

该设置通常是每个topic各自的设置，但是也可以全局设置所有topic的该设置，通过default\_topic\_conf。

消息排序

为了防止重试（retries > 0）导致消息顺序重排，可以设置max.in.flight.requests.per.connection=1确保一次只有一个请求被发送到broker。

message.send.max.retries：最大重新发送消息的次数，默认为2。

批处理和压缩

batch.num.messages：设置每一批发送消息的数量。

compression.codec：是否压缩。

队列限制

queue.buffering.max.messages：在给定时间内，存在buffer中的未发送消息的队列的长度限制。

queue.buffering.max.ms：设置给定的时间。