链滴

kafka 学习纪录及与 spring boot 的整合

作者：wuhongxu
原文链接：https://ld246.com/article/1521088210005
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile
com/bing/20180120.jpg?imageView2/1/w/960/h/520/interlace/1/q/100">
<h2 id="kafka-基础知识及与spring配合使用介绍">kafka 基础知识及与 spring 配合使用介绍</h2

<h2 id="kafka基础知识梳理">kafka 基础知识梳理</h2>
<h3 id="名词解释">名词解释</h3>

producer：
消息生产者，发布消息到 kafka 集群的终端或服务。
broker：
kafka 集群中包含的服务器。
topic：
每条发布到 kafka 集群的消息属于的类别，即 kafka 是面向 topic 的。
partition：
partition 是物理上的概念，每个 topic 包含一个或多个 partition。kafka 分配的单位是 partitio
。
consumer：
从 kafka 集群中消费消息的终端或服务。
Consumer group：
high-level consumer API 中，每个 consumer 都属于一个 consumer group，每条消息只能被
onsumer group 中的一个 Consumer 消费，但可以被多个 consumer group 消费。
replica：
partition 的副本，保障 partition 的高可用。
leader：
replica 中的一个角色， producer 和 consumer 只跟 leader 交互。
follower：
replica 中的一个角色，从 leader 中复制数据。
controller：
kafka 集群中的其中一个服务器，用来进行 leader election 以及 各种 failover。
zookeeper：
kafka 通过 zookeeper 来存储集群的 meta 信息。

<h3 id="四个核心API">四个核心 API</h3>

应用程序使用 producer API 发布消息到 1 个或多个 topic 中。
应用程序使用 consumer API 来订阅一个或多个 topic,并处理产生的消息。
应用程序使用 streams API 充当一个流处理器,从 1 个或多个 topic 消费输入流,并产生一个输出
到 1 个或多个 topic,有效地将输入流转换到输出流。
connector API 允许构建或运行可重复使用的生产者或消费者,将 topic 链接到现有的应用程序或
据系统。

<h3 id="消息通信">消息通信</h3>
       通常来讲，消息模型可以分为两种：队列和发布-订阅式。队列
处理方式是一组消费者从服务器读取消息，一条消息只有其中的一个消费者来处理。在发布-订阅模
中，消息被广播给所有的消费者，接收到消息的消费者都可以处理此消息。Kafka 为这两种模型提供
单一的消费者抽象模型： 消费者组(consumer group)。消费者用一个消费者组名标记自己。
       一个发布在 Topic 上消息被分发给此消费者组中的一个消费者
假如所有的消费者都在一个组中，那么这就变成了 queue 模型。假如所有的消费者都在不同的组中
那么就完全变成了发布-订阅模型。更通用的， 我们可以创建一些消费者组作为逻辑上的订阅者。每
组包含数目不等的消费者，一个组内多个消费者可以用来扩展性能和容错。
nbsp;
       并且，kafka 能够保证生产者发送到一个特定的 Topic 的分区
，消息将会按照它们发送的顺序依次加入，也就是说，如果一个消息 M1 和 M2 使用相同的 producer
发送，M1 先发送，那么 M1 将比 M2 的 offset 低，并且优先的出现在日志中。消费者收到的消息

是此顺序。如果一个 Topic 配置了复制因子（replication facto）为 N,那么可以允许 N-1 服务器宕而不丢失任何已经提交（committed）的消息。此特性说明 kafka 有比传统的消息系统更强的顺序保。但是，相同的消费者组中不能有比分区更多的消费者，否则多出的消费者一直处于空等待，不会收消息。
<h3 id="主题和日志--Topic和Log-">主题和日志 (Topic 和 Log)</h3>
      每一个分区(partition)都是一个顺序的、不可变的消息队列,并且可以续的添加。分区中的消息都被分了一个序列号,称之为偏移量(offset),在每个分区中此偏移量都是唯一。Kafka 集群保持所有的消息,直到它们过期,无论消息是否被消费了。实际上消费者所持有的仅有的数据就是这个偏移量，也就是消费者在这个 log 中的位置。 这个偏移量由消费者控制：正常情况当费者消费消息的时候，偏移量也线性的的增加。但是实际偏移量由消费者控制，消费者可以将偏移量置为更老的一个偏移量，重新读取消息。 可以看到这种设计对消费者来说操作自如， 一个消费者的作不会影响其它消费者对此 log 的处理。 再说说分区。Kafka 中采用分区的设计有几个目的。一是可处理更多的消息，不受单台服务器的限制。Topic 拥有多个分区意味着它可以不受限的处理更多的数。第二，分区可以作为并行处理的单元，稍后会谈到这一点。
<h3 id="分布式-Distribution-">分布式(Distribution)</h3>
Log 的分区被分布到集群中的多个服务器上。每个服务器处理它分到的分区。根据配置每个分区可以复制到其它服务器作为备份容错。 每个分区有一个 leader，零或多个 follower。Leader 处理此区的所有的读写请求，而 follower 被动的复制数据。如果 leader 宕机，其它的一个 follower 会被举为新的 leader。 一台服务器可能同时是一个分区的 leader，另一个分区的 follower。 这样可以平负载，避免所有的请求都只让一台或者某几台服务器处理。
<h2 id="kafka的使用">kafka 的使用</h2>
<blockquote>
安装配置过程省略（zookeeper、kafka）
</blockquote>
<h3 id="kafka自身的api">kafka 自身的 api</h3>
官方文档
<h3 id="spring对于kafka的抽象支持">spring 对于 kafka 的抽象支持</h3>
spring-kafka 整合了 kafka 的 consumer 和 producer 包，所以可以同时做为生产者和消费者用

消费者： 主要需要提供 KafkaMessageListenerContainer 做为消费者客户端(可以理解为一个独启动的服务，需要调用 container.start()启动).spring boot 抽象之后，
会自动配置 container，简化配置之后，需要提供 bean KafkaListenerContainerFactory,为 factory 提供属性等。spring boot 继续提供默认值，简化配置，可以直接在 property 文件中，配置连接属即可

KafkaMessageListenerContainer 需要的 containerProps 中可以设置监听器 setMessageLister(Object),参数需要实现 MessageListener 监听,
spring boot 提供了 @KafkaListener 注解，可以在任意 bean 中的方法上使用，支持直接注入消息身或者 ConsumerRecord，操作消息体。

3.生产者：spring 为 kafka 的 producer 提供了 KafkaTemplate 类，主要封装了 kafka 生产者的 ProducerRecord 类,
生产者只需要配置 KafkaTemplate 即可.spring boot 为 property 提供了属性配置，所以可以配置属性之后直接使用 template