〇、回顾

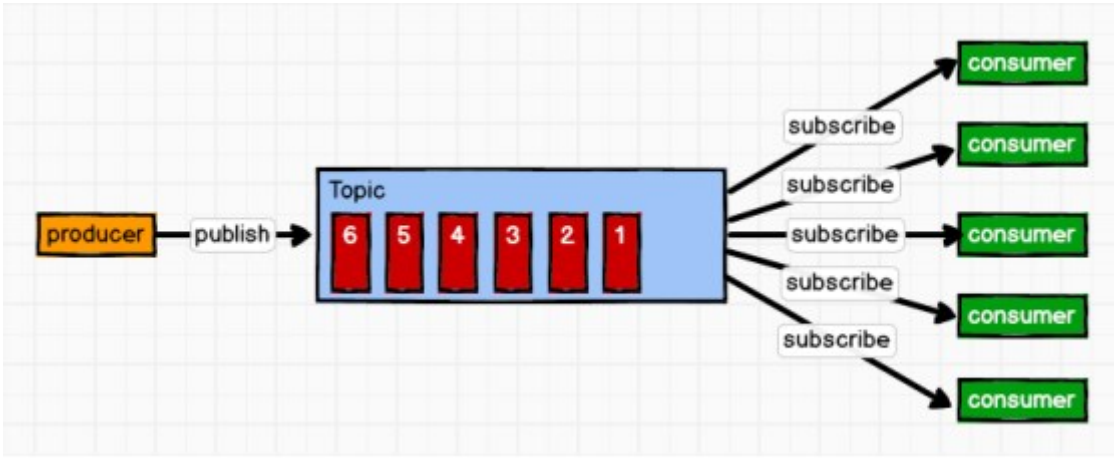
RabbitMQ的工作模式：一对一（点对点模式）、消息订阅模式、路由模式、消息通配符topic模式

一、Kafka基本知识

1、概念

分布式的基于**发布/订阅模式**的消息队列，是一个分布式的**数据流式传输平台。**

发布/订阅模式：生产者发送消息到topic，多个消费者消费同一topic的内容



2、介绍

Kafka一般用来**缓存数据**，**Spark**通过消费Kafka的数据进行**计算**

Apache Kafka是一个开源消息系统，由**Scala**写成

Kafka对消息保存时**根据Topic进行归类（按照类别记录存储，也被称为主题topic）**

一个集群有多个kafka实例组成，每个**实例**(server)称为**broker（掮客/中介）**。

kafka集群，还是consumer都依赖于**zookeeper**集群保存一些meta信息，来保证系统可用性。

每条**记录**由一个键，一个值和一个时间戳组成

3、特点

提供对**流式数据**的发布和订阅

持久的**容错的方式存储**流式数据

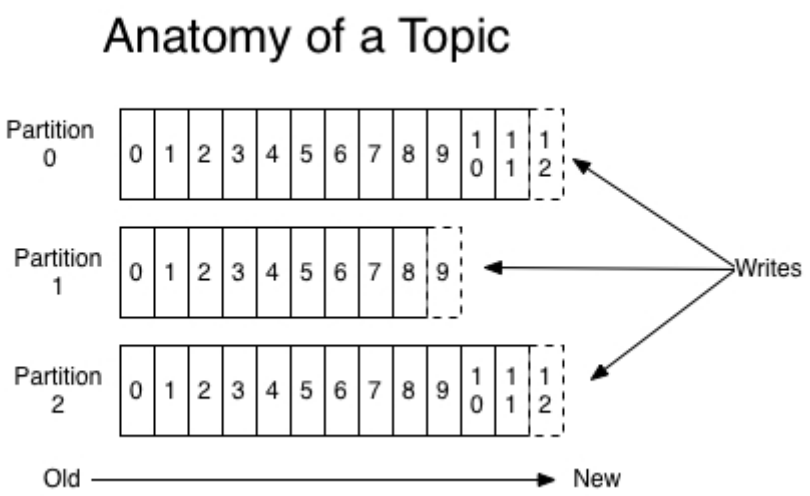
及时地处理流式数据

集群方式运行在可**跨多个数据中心**的服务器上

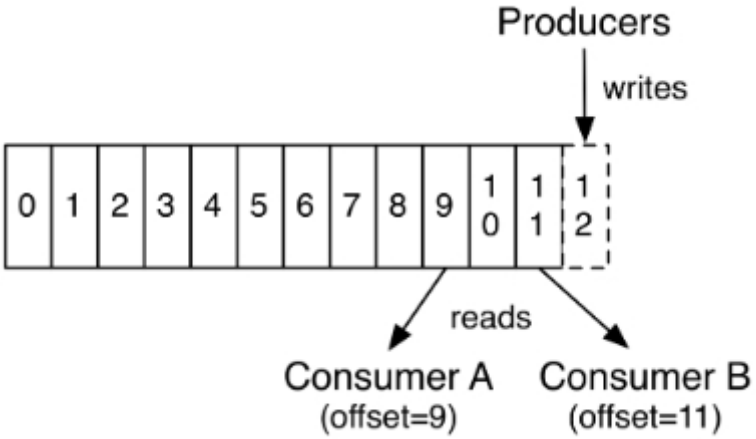
常两种**应用**：多应用间需要有可靠的实时数据通道&基于实时传输和及时计算的应用

4、核心概念

- Broker：Kafka的一台服务器，集群中有多个Broker
- Topic（相当于数据库）：代表不同的数据主题，基于多订阅者模式（一个topic可以由多个消费者订阅），一个大的topic可以分布式存储在多个broker中
- Partition（相当于数据库中的表）：
    - 表示topic的分区，通过分区，topic可以不断扩展
    - 一个topic的多个分区可以分布式存储在多个broker，一个topic也可以被多个消费者消费，实现并行处理。（partition内部有序发送，但一个topic的多个partition整体无序）
    - 拓扑结构：



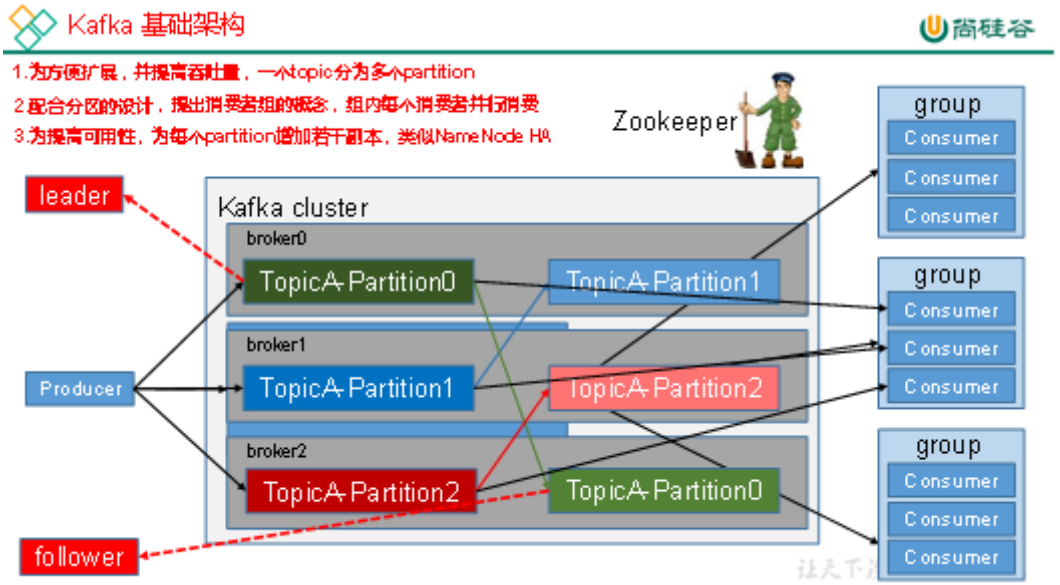
- Offset：偏移量
    - 每个分区数据有序，数据按**时间顺序追加**到分区的commit log中
    - **数据顺序通过offset的id标识**（offset有序且不可变）
    - 消费者端会保存offset元数据（在commit log中的位置），由消费者控制offset偏移量（因此消费者可以采用任何顺序消费数据），消费者的多少，对集群和其他消费者无影响
    - 数据流消费



- 持久化：
  - 通过可配置参数（保留期限）**保留发布的记录**
  - 期限内随时被消费，超期清除并释放磁盘空间
  - 长时间存储对Kafka性能无影响（性能与数据大小无关）
- 副本机制：
  - 日志的partition会分布在集群broker上，服务器处理数据和请求时共享分区，各分区会在**服务器上备份**，确保容错性
  - 对于每个partition，都有一台服务器叫leader，零到多台服务器叫followers
  - leader用于处理分区的读写请求，followers被动的同步leader的数据
  - leader宕机，某台followers会自动变为leader
  - leader和followers会放在不同的broker
- Producer
  - 生产者，向broker发消息的客户端
  - 负责将消息分配到topic的指定partition中
- Consumer
  - 消费者，从broker中取消息的客户端
  - 每个**消费者都要维护自己读取数据的offset**
  - 0.9前offset保存到zookeeper，0.9后保存到kafka的**__consumer_offsets**主题中
- Consumer Group
  - 消费者组，每个消费者都属于一个group
  - 同一组的不同消费者可以分布在多个进程/机器上
  - 如果所有消费者在同一group中，消息记录会负载均衡（**单播**）到每个消费者单例

- 如果消费者实例在不同的group中，每条消息会**广播**到所有消费者进程
- 一个topic可以有多个group，topic的消息会复制到CG中，但**每个partition只会发送到CG中的某一个consumer**

5、基础架构



二、Kafka快速入门

1、安装部署

- 集群规划：3个集群
- jar包下载：http://kafka.apache.org/downloads.html
- 集群部署
  - 解压
  - 改名
  - 创建logs文件夹
  - 配置config/server.properties
  - 配置环境变量source /etc/profile
  - 分发安装包xsync kafka/
  - 启动/关闭集群&群起脚本

2、命令行操作

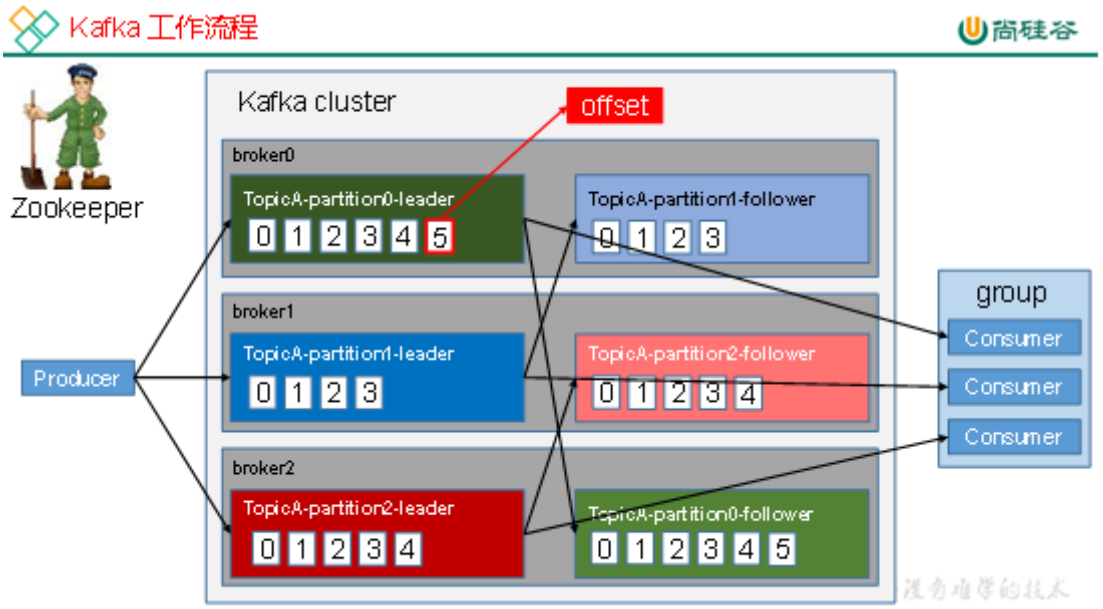
- 查看topic --list
- 创建topic --create
- 删除topic --delete
- 发送消息 kafka-console-producer.sh

- 消费消息 kafka-console-consumer.sh
- 查看topic详情 --describe
- 修改分区数 --alter

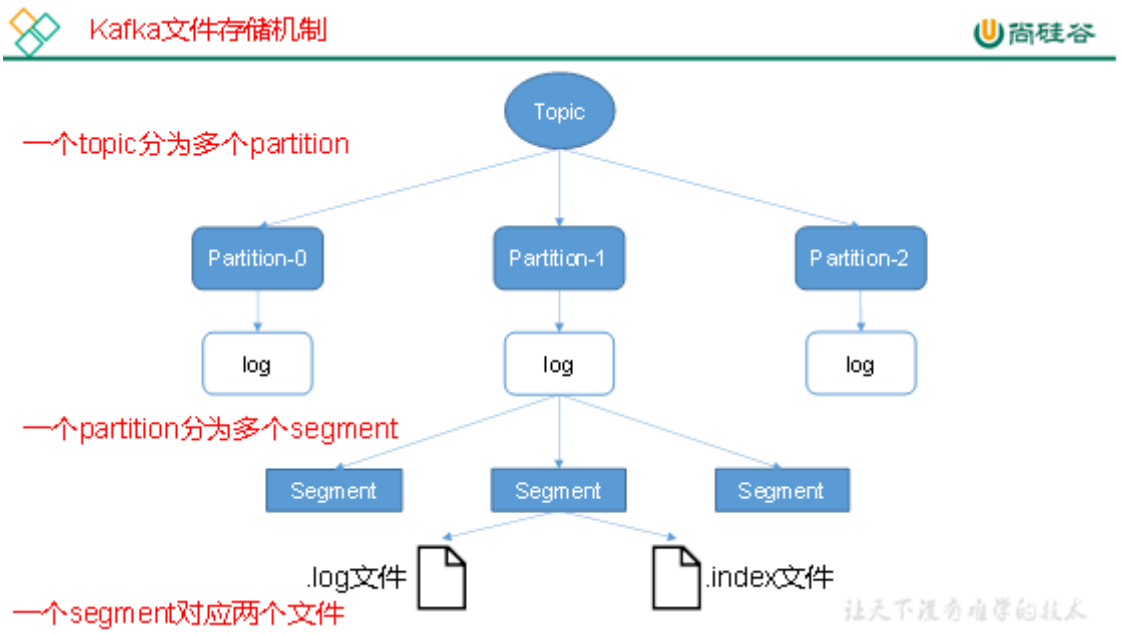
三、Kafka架构深入

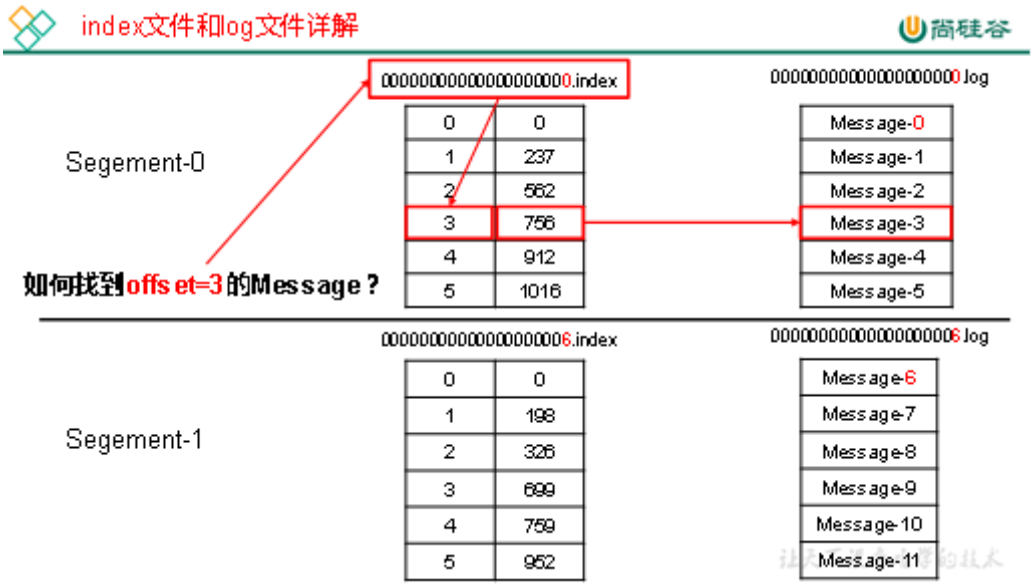
1、工作流程及文件存储机制

- 工作流程
  - 以topic进行分类，生产者与消费者都是面向topic的
  - topic是逻辑概念，partition是物理概念（与log文件一一对应）
  - log文件存储生产的数据，将数据追加到文件末端，每条数据都有对应的offset（相当于序列索引）
  - 消费者实时记录offset，便于出错恢复时，从上次位置继续消费



- 文件存储机制
  - 防止log文件过大，采取分片和索引机制（每个**partition分区被分为多个segment**段）
  - 每个segment文件中的消息数量不一定相同，方便已被消费的消息的清理
  - 每个segment对应两个文件——“.index”文件（存索引，二分定位message，元数据指向offset地址）和“.log”文件（存储数据）
  - 两文件以偏移量命名（同名）





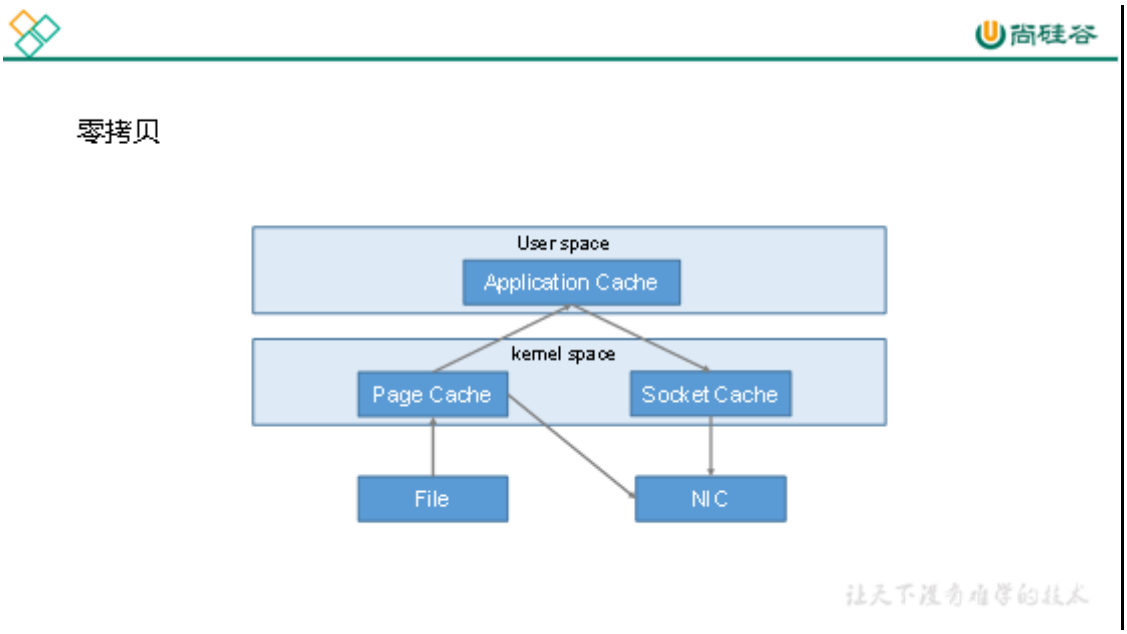
2、生产者

- 不同分区策略，将数据封装为**ProducerRecord**对象
- 数据可靠性保证：**partition向生产者**发送ack
  - 副本同步：全部完成才发送ack，选举leader只需要n+1个副本
  - leader维护in-sync replica set (ISR)，followers未完成与leader的同步，则被踢出ISR（出故障后会从ISR中选取leader）
  - ack应答机制：三个0（不等待ack） 1（leader成功ack） -1（leader和follower均成功ack）
  - 故障处理细节：LEO（每个副本的最后一个offset）和HW（所有副本中最小的LEO）

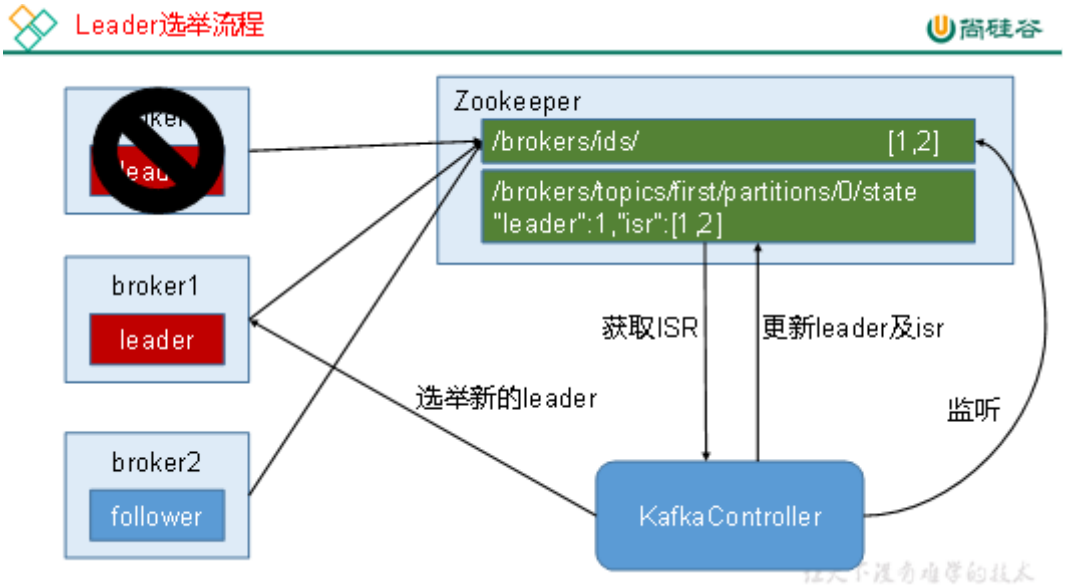
- Exactly Once语义：通过幂等性，实现每条消息只被发送一次

3、消费者

- 消费方式：
  - **pull**模式从broker读取
  - push难以适应速率不同的consumer，而pull可以根据消费能力消费数据
  - 不足：无数据会陷入循环，方式：通过时长参数timeout
- **分区分配**策略：那个partition由哪个consumer来消费，包括RoundRobin、Range
- offset维护：避免宕机，**实时记录**消费到了哪个offset，保存到内置的topic中，该topic为__consumer_offset
- 消费者组案例：测试同一个消费者组中的消费者，同一时刻只能有一个消费者消费。
- 高效读写数据
  - 顺序写磁盘：省去了大量磁头寻址的时间
  - **零拷贝**



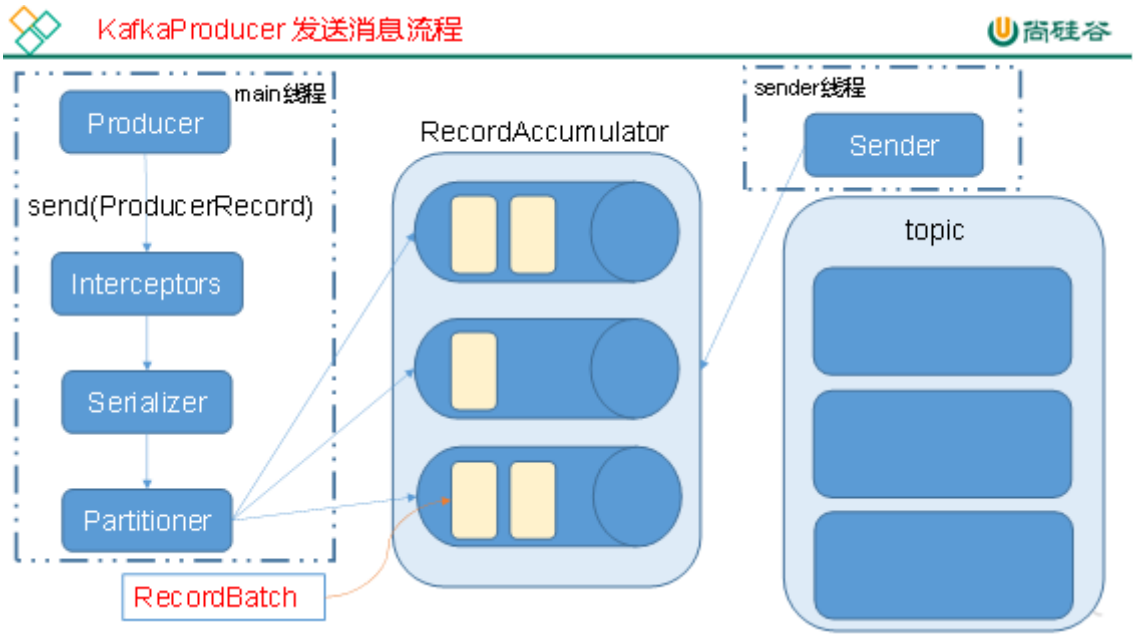
- Zookeeper的作用
  - Kafka集群中有一个broker会被选举为**Controller**，负责**管理**上下线、副本管理、leader选举等
  - Controller的管理工作都是**依赖于**Zookeeper的。



四、Kafka API

1、Producer API

- 消息发送流程
  - **异步发送**，涉及main和sender线程以及RecordAccumulator
  - main发送到RecordAccumulator，sender从RecordAccumulator拉取到broker
  - 根据batch.size（数据批次）和linger.ms（等待时间）判断发送数据的时间



- 异步发送api
  - 涉及的类：KafkaProducer、ProducerConfig、ProducerRecord（数据封装的对象）
  - 不带回调函数的api
  - 带回调函数的api：在producer**收到ack时调用**，为异步调用，ProducerRecord构造中多一个参数--Callback的匿名内部类的onCompletion方法，发送失败会自动重试
- 同步发送api：发送后阻塞进程，直至返回ack
  - 方式：producer.send()后加get()

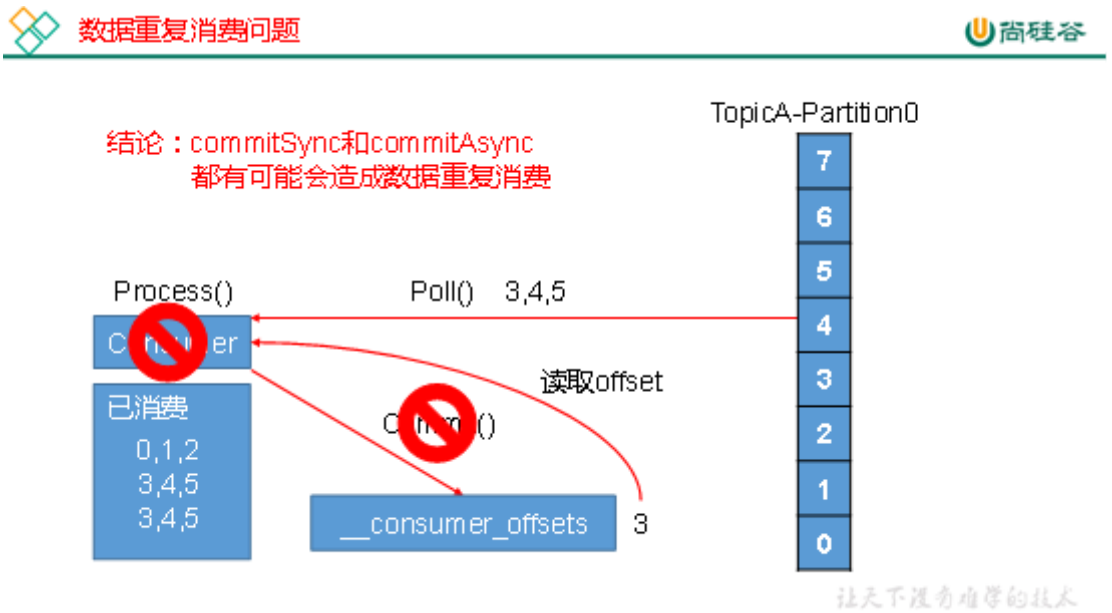
2、Consumer API

kafka数据持久化，不必担心数据丢失

实时记录，便于故障恢复，需要考虑offset的维护

- 自动提交offset
  - 需要用到的类：KafkaConsumer、ConsumerConfig、ConsuemrRecord
  - 自动提交的相关参数：enable.auto.commit、auto.commit.interval.ms（自动提交时间间隔）
- 手动提交offset
  - 自动提交基于时间，开发人员无法把握时机
  - 手动提交分为commitSync（同步提交）和commitAsync（异步提交）
    - 相同点：将本次poll的一批**数据最高的偏移量提交**

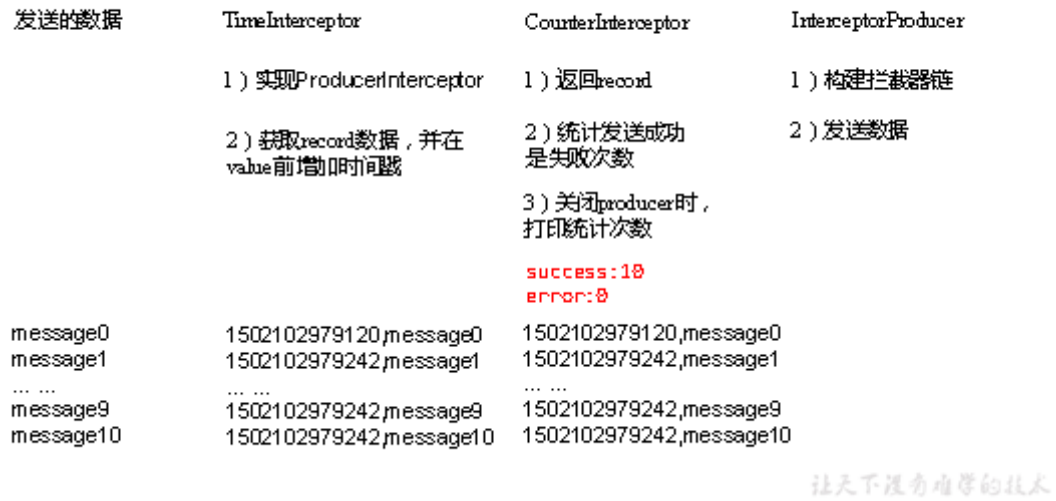
        - 不同点：同步提交阻塞进程直到成功，**自动失败重试**；异步提交无失败重试
    - 同步提交offset：可靠，阻塞线程，吞吐量受影响
    - 异步提交offset
    - 缺陷：数据的**漏消费**（先提交后消费）或者重复消费（先消费后提交，重复消费上一次的offset字段缓存）



- 自定义存储offset
    - 存储在内置topic中
    - offset维护过程繁琐，需要考虑消费者的Rebalace（分区变化、新消费者加入CG），需要重新获取对应的消费分区partition
    - 需要借助ConsumerRebalanceListener的两个方法，分别在Rebalace前后执行

3、自定义Interceptor拦截器

- 实现clients端的定制化控制逻辑，在消息发送前或producer回调逻辑前对消息做定制化需求
- 用户可以指定多个interceptor形成拦截链
- 实现接口是ProducerInterceptor，包括configure、onSend（可以对消息进行操作）、onAcknowledgement（成功或失败时调用）、close方法



- 配置对象prop.put(interceptorsList)

五、Kafka监控

1、Kafka Manager

上传解压kafka-manager，访问主机名:9000端口

2、Kafka Monitor

KafkaOffsetMonitor-assembly解压创建启动脚本，访问主机名:8086端口

六、对接Flume