法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象学院
  - 新浪微博：小象AI学院

大纲

- Kafka基本原理
- Kafka集群部署及演示操作
- Kafka API

大纲

Kafka基本原理

Kafka简介

- Kafka由Linkedin开发的消息队列，使用Scala语言编写
- 分布式、多分区、多副本、基于发布/订阅的消息系统
- Kafka设计的初衷是希望作为一个统一的信息收集平台，能够实时的收集反馈信息，并能够支撑较大的数据量，且具备良好的容错能力

Kafka特性

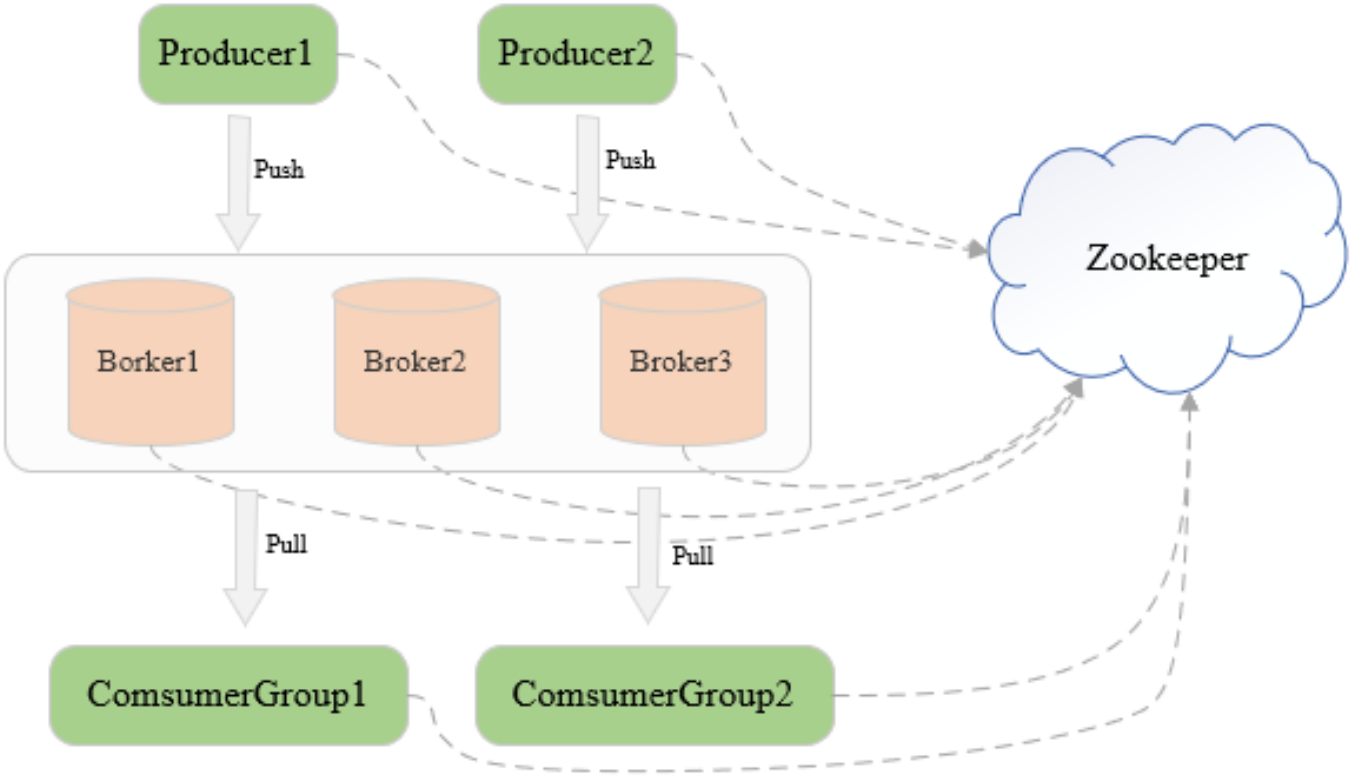
- 消息持久化：采用时间复杂度O(1)的磁盘结构顺序存储
- 高吞吐量：支持每秒百万级别的消息
- 易扩展：新增机器，集群无需停机，自动感知
- 高容错：通过多分区，多副本提供高容错性
- 多客户端支持：Java、PHP、Python等
- 实时性：进入到Kafka的消息能够立即被消费者消费

Kafka消息队列作用

- 消息队列的作用
  - 应用程序解耦并行处理
  - 顺序保证
  - 高吞吐率
  - 高容错、高可用
  - 可扩展
  - 峰值处理

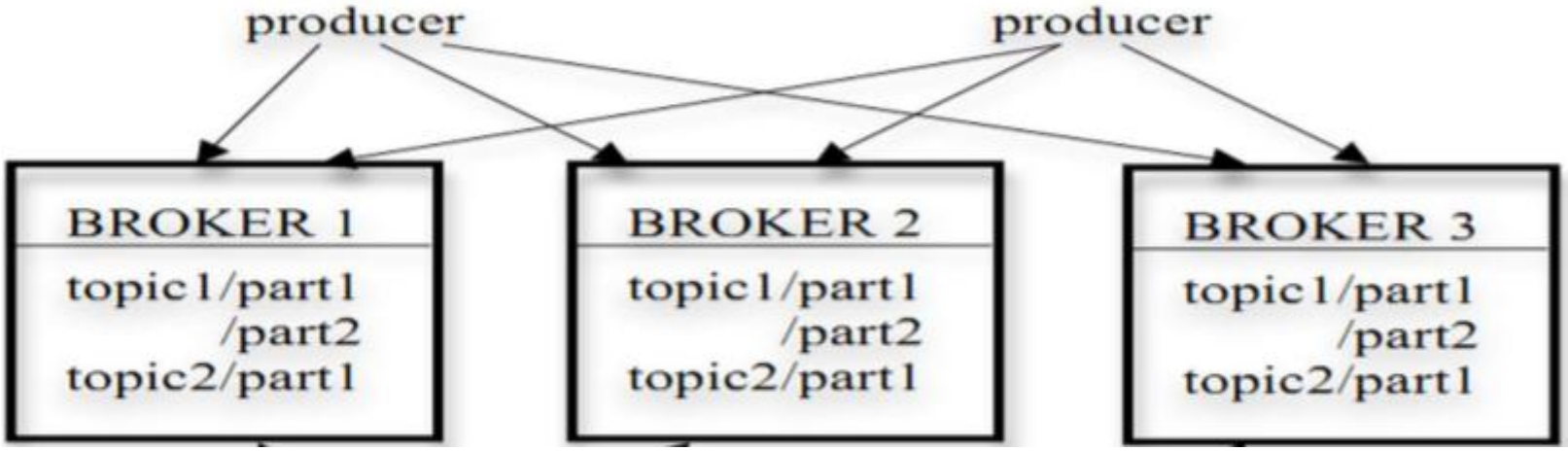
Kafka原理

- 在Kafka中，发送消息者称为Producer，而消息拉取者称为Consumer，通常consumer是被定义在Consumer Group里
- Kafka通过Zookeeper管理集群



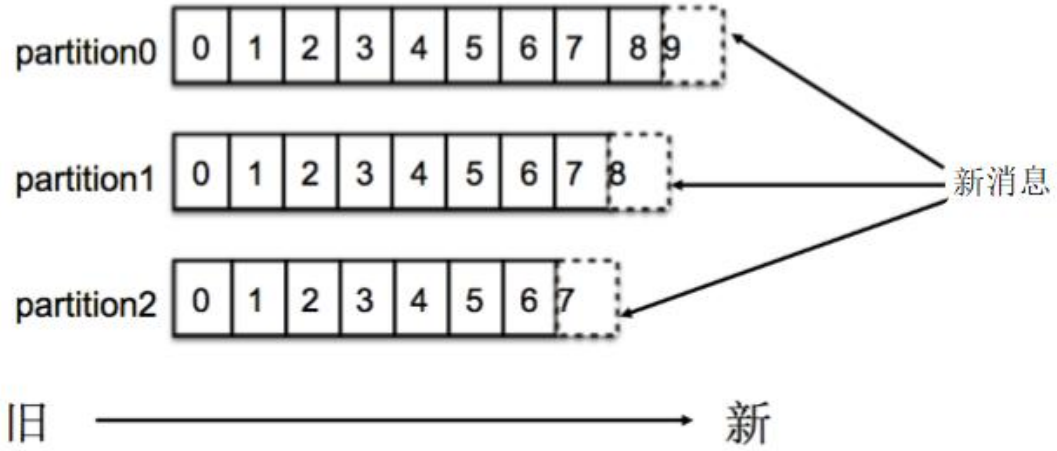
Kafka原理

- Kafka集群由多个实例组成，每个节点称为Broker，对消息保存时根据Topic进行归类
- 一个Topic可以被划分为多个Partition
- 每个Partition可以有多个副本



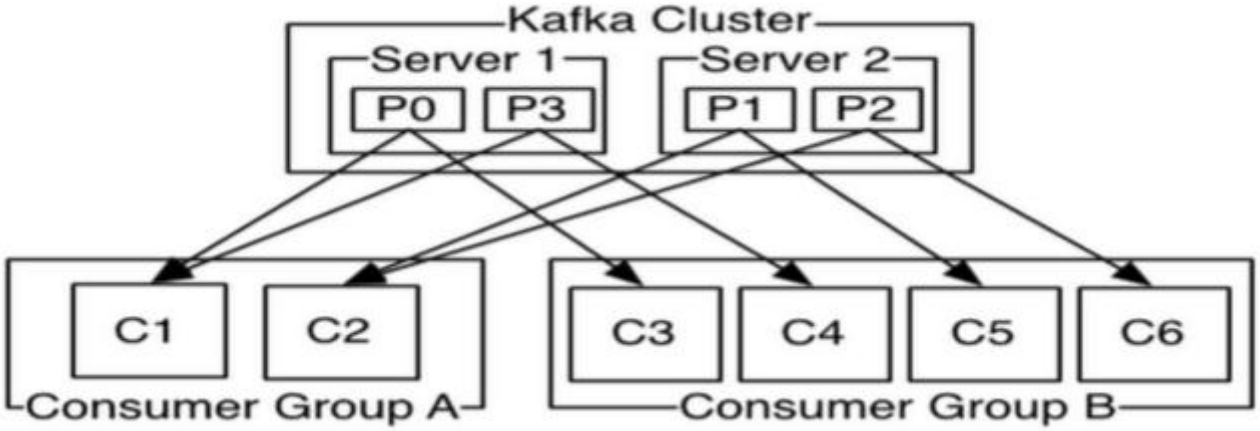
Kafka原理

- Partition内顺序存储，写入新消息采用追加的方式，消费消息采用FIFO的方式顺序拉取消息
- 一个Topic可以有多个分区，Kafka只保证同一个分区内有序，不保证Topic整体（多个分区之间）有序



Kafka原理

- Consumer Group（CG），为了加快读取速度，多个consumer可以划分为一个组，并行消费一个Topic
- 一个Topic可以由多个CG订阅，多个CG之间是平等的，同一个CG内可以有一个或多个consumer，同一个CG内的consumer之间是竞争关系，一个消息在一个CG内的只能被一个consumer消费



Kafka核心概念

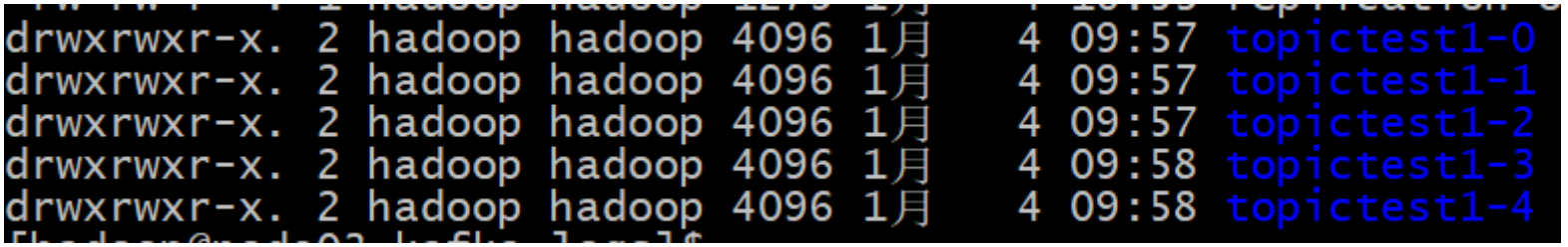
- Broker：启动kafka的一个实例就是一个broker，一个kafka集群可以启动多个broker
- Topic：kafka中同一种类型数据集的名称，相当于数据库中的表，productor将同一类型的数据写入的同一个topic下，consumer从同一个topic消费同一类型的数据
- Partition：一个topic可以设置多个分区，相当于把一个数据集分成多份分别放到不同的分区中存储，一个topic可以有一个或者多个分区，分区内消息有序
- Replication：副本，一个partition可以设置一个或者多个副本，副本主要保证系统能够持续不丢失的对外提供服务，提高系统的容错能力

Kafka核心概念

- Producer：消息生产者，负责向kafka中发布消息
- Consumer Group：消费者所属组，一个Consumer Group可以包含一个或者多个consumer，当一个topic被一个Consumer Group消费的时候，Consumer Group内只能有一个consumer消费同一条消息，不会出现同一个Consumer Group多个consumer同时消费一条消息造成一个消息被一个Consumer Group消费多次的情况
- Consumer：消息消费者，consumer从kafka指定的主题中拉取消息
- Zookeeper：Zookeeper在kafka集群中主要用于协调管理，Kafka将元数据信息保存在Zookeeper中，通过Zookeeper管理维护整个Kafka集群的动态扩展、各个Broker负载均衡、Partition leader选举等

Kafka存储

- 每个partition的副本是一个目录



- Segment：段文件，kafka中最小数据存储单位，一个partition包含多个segment文件，每个segment以message在partition中的起始偏移量命名以log结尾的文件
- Offset：消息在分区中的偏移量，用来在分区中唯一的标识这个消息

00000000000000000000.log
00000000000000368769.log
00000000000000737337.log

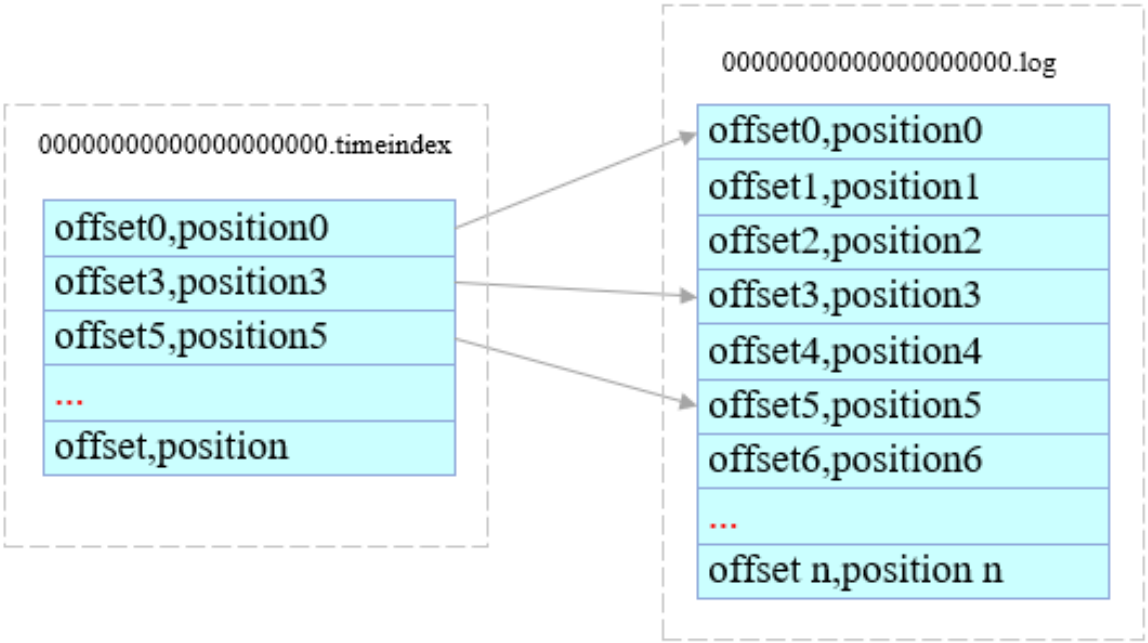
Kafka存储-索引文件

- Kafka为了提高写入、查询速度在partition文件夹下每一个segment log文件都有同名的索引文件，在kafka0.10以后的版本中会存在两个索引文件，一个用offset做名字以index结尾的索引文件，我们称为偏移量索引文件。一个是以消息写入的时间戳为做名字以timeinde结尾的索引文件，我们称为时间戳索引文件。

```
00000000000000000000.index
00000000000000000000.log
00000000000000000000.timeindex
```

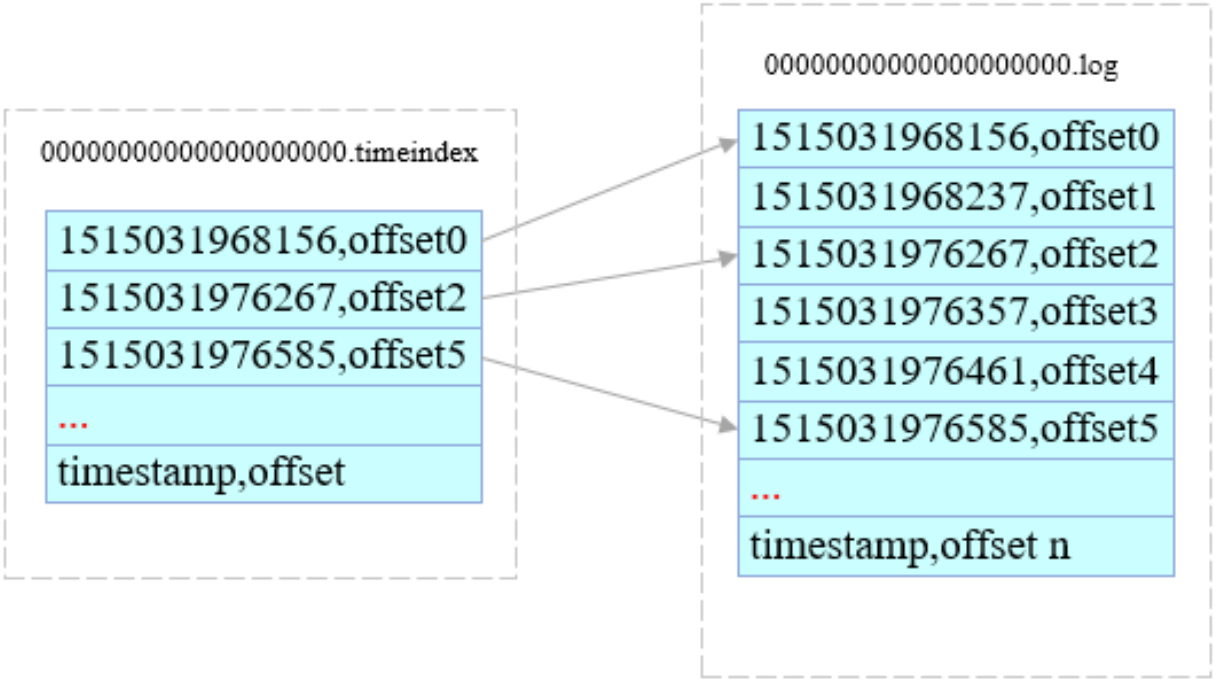
Kafka存储-偏移量索引文件

- 以偏移量作为名称，index为后缀
- 索引内容格式：offset,position
- 采用稀疏存储方式
- 通过log.index.interval.bytes设置索引跨度



Kafka存储-时间戳索引文件

- 以时间戳作为名称，以timeindex为后缀
- 索引内容格式：timestamp,offset
- 采用稀疏存储方式
- 通过log.index.interval.bytes设置索引跨度



Kafka 高可用实现

- 多分区多副本
  - 一个topic可以有多个分区，每个分区可以有多个副本
  - 0.8以前没有Replication，一旦某台brocker宕机，其上partition数据便丢失
  - 一个分区的多个副本选举一个leader，由leader负责读写，其他副本作为follower从leader同步消息
  - 同一个partition的不同副本分布到不同的broker

Kafka 高可用实现

- Kafka Controller选举
  - 从集群中的broker选举出一个Broker作为Controller控制节点
  - 负责整个集群的管理，如Broker管理、Topic管理、Partition Leader选举等
  - 选举过程通过向Zookeeper创建临时znode实现，为被选中的Broker监听Controller的znode，等待下次选举

Kafka 高可用实现

- Kafka Partition Leader选举
  - Controller负责分区Leader选举
  - ISR列表
    - ✓ Follower批量从Leader拖取数据
    - ✓ Leader跟踪不其 保持同步的flower列表ISR（ In Sync Replica），ISR作为下次选主的候选列表
    - ✓ Follower心跳超时或者消息落后太多，将被移除出ISR
  - Leader失败后，从ISR列表中选择一个Follower作为新的Leader

大纲

Kafka集群部署及演示操作

Kafka集群部署

- 集群规划
  - 使用3台机器部署，分别是node03、node04、node05
- 下载Kafka安装包
  - 下载地址http://kafka.apache.org/downloads，选择Kafka版本
- 使用hadoop用户将安装包上传到其中一台机器上，并解压到/home/hadoop/apps目录下
  - tar -zxvf kafka_2.11-0.10.2.1.tgz -C /home/hadoop/apps
- 修改配置文件
  - cd /home/hadoop/apps/kafka_2.11-0.10.2.1/config
  - vim server.properties

Kafka集群部署

- server.properties配置文件添加修改的内容如下

- 每个borker的id是唯一的，多个broker要设置不同的id
  broker.id=0
- 访问端口号
  port=9092
- 访问地址
  host.name=192.168.183.102
- 允许删除topic
  delete.topic.enable=true
- 存储数据路径，默认是在/tmp目录下，需要修改
  log.dirs=/home/hadoop/apps/kafka_2.11-0.10.2.1/kafka-logs
- 创建topic默认分区数
  num.partitions=1
- 数据保存时间，默认7天，单位小时
  log.retention.hours=168
- Zookeeper访问地址，多个地址用逗号隔开
  zookeeper.connect=192.168.183.100:2181,192.168.183.101:2181,192.168.183.102:2181

Kafka集群部署

- 在/home/hadoop/apps/kafka_2.11-0.10.2.1创建kafka-logs文件夹
  - mkdir /home/hadoop/apps/kafka_2.11-0.10.2.1/kafka-logs
- 使用scp将配置好的kafka安装包拷贝到node04和node05两个节点
  - scp -r /home/hadoop/apps/kafka_2.11-0.10.2.1 hadoop@node04:/home/hadoop/apps/
  - scp -r /home/hadoop/apps/kafka_2.11-0.10.2.1 hadoop@node05:/home/hadoop/apps/
- 分别修改node04和node05的配置文件server.properties
  - node04的server.properties修改项

broker.id=1

host.name=192.168.183.103

  - node05的server.properties修改项

broker.id=2

host.name=192.168.183.104

Kafka集群部署

- 分别在node03、node04、node05启动kafka
  - cd /home/hadoop/apps/kafka_2.11-0.10.2.1
  - 启动的时候使用-daemon选项，则kafka将以守护进程的方式启动

    bin/kafka-server-start.sh -daemon config/server.properties
- 日志目录
  - 默认在kafka安装路径生成的logs文件夹中

Kafka主题管理

- 使用kafka自带的Producer客户端创建topic
  - 创建topic名称为topictest1，3个分区，每个分区有2个副本

  bin/kafka-topics.sh --create --zookeeper 192.168.183.100:2181,192.168.183.101:2181,192.168.183.102:2181 --replication-factor 2 --partitions 3 --topic topictest1
- 查看kafka中已经创建的主题列表
  - bin/kafka-topics.sh --list --zookeeper 192.168.183.100:2181,192.168.183.101:2181,192.168.183.102:2181
- 使用describe查看主题信息
  - bin/kafka-topics.sh --describe --zookeeper 192.168.183.100:2181,192.168.183.101:2181,192.168.183.102:2181 --topic topictest1

Kafka主题管理

- 给指定主题增加分区，不支持减少分区的操作
  - 给topictest1主题分区由最初创建的3个分区，增加到5个分区

bin/kafka-topics.sh --alter --zookeeper 192.168.183.100:2181,
192.168.183.101:2181,192.168.183.102:2181 --topic topictest1
--partitions 5

- 使用Kafka自带的生产者客户端脚本向topictest1主题发送消息
  - bin/kafka-console-producer.sh --broker-list 192.168.183.102:9092,192.168.183.103:9092 --topic topictest1
- 使用kafka自带的消费者客户端脚本从topictest1主题拉取消息
  - bin/kafka-console-consumer.sh --zookeeper 192.168.183.100:2181 --from-beginning --topic topictest1

大纲

Kafka API

Producer关键参数

- bootstrap.servers ： Broker列表，定义格式host:port，多个Broker之间用逗号隔开
- key.serializer：key序列化方式，常用字符串序列化
  - org.apache.kafka.common.serialization.StringSerializer
- value.serializer：value序列化方式，常用字符串放序列化
  - org.apache.kafka.common.serialization.StringSerializer

Producer关键参数

- acks设置Producer发送消息到Borker是否等待接收Broker返回成功送达信号
  - 0 表示Producer发送消息到Broker之后不需要等待Broker返回成功送达的信号，这种方式吞吐量高，但是存在数据丢失的风险
  - 1 表示Broker接收到消息成功写入本地log文件后向Producer返回成功接收的信号，不需要等待所有的Follower全部同步完消息后再做回应，这种方式在数据丢失风险和吞吐量之间做了平衡，默认值1
  - all（或者-1）表示Broker接收到Producer的消息成功写入本地log并且等待所有的Follower成功写入本地log后向Producer返回成功接收的信号，这种方式能够保证消息不丢失，但是性能最差

Consumer关键参数

- bootstrap.servers：Broker列表，定义格式host:port，多个Broker之间用逗号隔开
- key.deserializer：key反序列化方式，常用字符串反序列化
  - org.apache.kafka.common.serialization.StringDeserializer
- value.deserializer：value反序列化方式，常用字符串反序列化
  - org.apache.kafka.common.serialization.StringDeserializer
- group.id：Consumer Group Id，该值应该是连接kafka集群的唯一值，同一组内可以有多个Consumer。
- auto.offset.reset：Consumer从Kafka拉取消息的方式
  - earliest表示从最早的偏移量开始拉取，
  - latest表示从最新的偏移量开始拉取，默认值latest
  - none表示如果没有发现该Consumer组之前拉取的偏移量则抛异常

Consumer关键参数

- enable.auto.commit
  - Consumer是否自动提交偏移量，默认值true
- auto.commit.interval.ms
  - 如果enable.auto.commit设置为true，自动提交的时间间隔，默认值5000毫秒（5秒）

疑问

- 小象问答官网
  - http://wenda.chinahadoop.cn

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：小象学院
- 新浪微博：小象AI学院