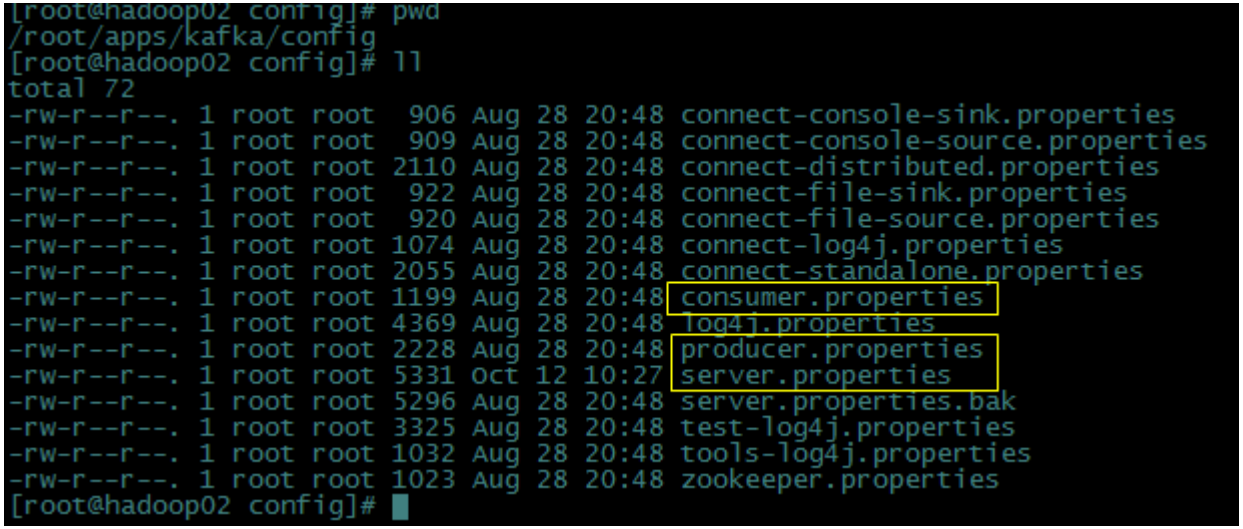

(1).producer.properties:生产端的配置文件

```
#指定kafka节点列表，用于获取metadata，不必全部指定
#需要kafka的服务器地址，来获取每一个topic的分片数等元数据信息。
metadata.broker.list=kafka01:9092,kafka02:9092,kafka03:9092

#生产者生产的消息被发送到哪个block，需要一个分组策略。
#指定分区处理类。默认kafka.producer.DefaultPartitioner，表通过key哈希到对应分区
#partitioner.class=kafka.producer.DefaultPartitioner

#生产者生产的消息可以通过一定的压缩策略（或者说压缩算法）来压缩。消息被压缩后发送到broker集群，
#而broker集群是不会进行解压缩的，broker集群只会把消息发送到消费者集群，然后由消费者来解压缩。
#是否压缩，默认0表示不压缩，1表示用gzip压缩，2表示用snappy压缩。
#压缩后消息中会有头来指明消息压缩类型，故在消费者端消息解压是透明的无需指定。
#文本数据会以1比10或者更高的压缩比进行压缩。
compression.codec=none

#指定序列化处理类，消息在网络上传输就需要序列化，它有String、数组等许多种实现。
serializer.class=kafka.serializer.DefaultEncoder

#如果要压缩消息，这里指定哪些topic要压缩消息，默认empty，表示不压缩。
#如果上面启用了压缩，那么这里就需要设置
```

10. windows下IntelliJ IDEA搭建kafka源码环境

最新评论

1. Re:使用Spring Cloud Feign作为HTTP客户端调用远程HTTP服务
不过现在 这个 FeignClient 里边的 ${url.xapi} 结合 spring cloud config 还不能动态刷新

--mushishi

2. Re:kafka常用操作命令
--from等这些参数除了begining外还有哪些选项呢，在哪可以看到完整选项。在官网不知道在哪

-----江北

3. Re:spring整合kafka项目生产和消费测试结果记录（一）
380659521@qq.com博主来一份源码，谢谢

--mjsjshhcbhchcecececec

4. Re:MediaType是application/x-www-form-urlencoded的接口测试方法
非常感谢，请问下楼主是在哪找到的相应的文档呢

--程序猿1111

5. Re:Spring的线程池ThreadPoolTaskExecutor使用案例
解决办法是手动新建一个bean，然后注入service，可保不污染

--魏子腾

阅读排行榜

1. hadoop集群之HDFS和YARN启动和停止命令(16472)
2. kafka消费者如何才能从头开始消费某个topic的全量数据(15253)
3. nginx虚拟主机配置(15094)

```
#compressed.topics=
#这是消息的确认机制，默认值是0。在面试中常被问到。
#producer有个ack参数，有三个值，分别代表：
#（1）不在乎是否写入成功；
#（2）写入leader成功；
#（3）写入leader和所有副本都成功；
#要求非常可靠的话可以牺牲性能设置成最后一种。
#为了保证消息不丢失，至少要设置为1，也就
#是说至少保证leader将消息保存成功。
#设置发送数据是否需要服务端的反馈,有三个值0,1,-1，分别代表3种状态：
#0： producer不会等待broker发送ack。生产者只要把消息发送给broker之后，就认为发送成功了，这是第1种情况；
#1： 当leader接收到消息之后发送ack。生产者把消息发送到broker之后，并且消息被写入到本地文件，才认为发送成功，这是第二种情况；#-1： 当所有的follower都同步消息成功后发送ack。不仅是主的分区将消息保存成功了，
#而且其所有的分区的副本数也都同步好了，才会被认为发动成功，这是第3种情况。
request.required.acks=0

#broker必须在该时间范围之内给出反馈，否则失败。
#在向producer发送ack之前,broker允许等待的最大时间 ，如果超时，
#broker将会向producer发送一个error ACK.意味着上一次消息因为某种原因
#未能成功(比如follower未能同步成功)
request.timeout.ms=10000

#生产者将消息发送到broker，有两种方式，一种是同步，表示生产者发送一条，broker就接收一条；
#还有一种是异步，表示生产者积累到一批的消息，装到一个池子里面缓存起来，再发送给broker，
#这个池子不会无限缓存消息，在下面，它分别有一个时间限制（时间阈值）和一个数量限制（数量阈值）的参数供我们来设置。
#一般我们会选择异步。
#同步还是异步发送消息，默认“sync”表同步，"async"表异步。异步可以提高发送吞吐量，
#也意味着消息将会在本地buffer中,并适时批量发送，但是也可能导致丢失未发送过去的消息
producer.type=sync

#在async模式下,当message被缓存的时间超过此值后,将会批量发送给broker，
#默认为5000ms
#此值和batch.num.messages协同工作.
queue.buffering.max.ms = 5000

#异步情况下，缓存中允许存放消息数量的大小。
#在async模式下,producer端允许buffer的最大消息量
#无论如何,producer都无法尽快的将消息发送给broker,从而导致消息在producer端大量沉积
#此时,如果消息的条数达到阀值,将会导致producer端阻塞或者消息被抛弃，默认为10000条消息。
queue.buffering.max.messages=20000
```

4. HBase启动和停止命令(12663)
5. Spring的线程池ThreadPoolTaskExecutor使用案例(12262)

评论排行榜

1. spring整合kafka项目生产和消费测试结果记录（一）(15)
2. Spring的线程池ThreadPoolTaskExecutor使用案例(4)
3. hadoop集群之HDFS和YARN启动和停止命令(3)
4. Kafka文件的存储机制(2)
5. windows下IntelliJ IDEA搭建kafka源码环境(2)

```
#如果是异步，指定每次批量发送数据量，默认为200
batch.num.messages=500

#在生产端的缓冲池中，消息发送出去之后，在没有收到确认之前，该缓冲池中的消息是不能被删除的，
#但是生产者一直在生产消息，这个时候缓冲池可能会被撑爆，所以这就需要有一个处理的策略。
#有两种处理方式，一种是让生产者先别生产那么快，阻塞一下，等会再生产；另一种是将缓冲池中的消息清空。
#当消息在producer端沉积的条数达到"queue.buffering.max.meesages"后阻塞一定时间后，
#队列仍然没有enqueue(producer仍然没有发送出任何消息)
#此时producer可以继续阻塞或者将消息抛弃,此timeout值用于控制"阻塞"的时间
#-1: 不限制阻塞超时时间，让produce一直阻塞,这个时候消息就不会被抛弃
#0: 立即清空队列,消息被抛弃
queue.enqueue.timeout.ms=-1


#当producer接收到error ACK,或者没有接收到ACK时,允许消息重发的次数
#因为broker并没有完整的机制来避免消息重复,所以当网络异常时(比如ACK丢失)
#有可能导致broker接收到重复的消息,默认值为3.
message.send.max.retries=3

#producer刷新topic metada的时间间隔,producer需要知道partition leader
#的位置,以及当前topic的情况
#因此producer需要一个机制来获取最新的metadata,当producer遇到特定错误时,
#将会立即刷新
#(比如topic失效,partition丢失,leader失效等),此外也可以通过此参数来配置
#额外的刷新机制，默认值600000
topic.metadata.refresh.interval.ms=60000
```

(2).consumer.properties:消费端的配置文件

```
#消费者集群通过连接Zookeeper来找到broker。
#zookeeper连接服务器地址
zookeeper.connect=zk01:2181,zk02:2181,zk03:2181
```

```
#zookeeper的session过期时间，默认5000ms，用于检测消费者是否挂掉
zookeeper.session.timeout.ms=5000

#当消费者挂掉，其他消费者要等该指定时间才能检查到并且触发重新负载均衡
zookeeper.connection.timeout.ms=10000

#这是一个时间阈值。
#指定多久消费者更新offset到zookeeper中。
#注意offset更新时基于time而不是每次获得的消息。
#一旦在更新zookeeper发生异常并重启，将可能拿到已拿到过的消息
zookeeper.sync.time.ms=2000

#指定消费
group.id=xxxxx

#这是一个数量阈值，经测试是500条。
#当consumer消费一定量的消息之后,将会自动向zookeeper提交offset信息#注意offset信息并不是每消费一次消息就向zk提交
#一次,而是现在本地保存(内存),并定期提交,默认为true
auto.commit.enable=true

# 自动更新时间。默认60 * 1000
auto.commit.interval.ms=1000

# 当前consumer的标识,可以设定,也可以有系统生成,
#主要用来跟踪消息消费情况,便于观察
conusmer.id=xxx

# 消费者客户端编号，用于区分不同客户端，默认客户端程序自动产生
client.id=xxxx

# 最大取多少块缓存到消费者(默认10)
queued.max.message.chunks=50

# 当有新的consumer加入到group时,将会reblance,此后将会
#有partitions的消费端迁移到新  的consumer上,如果一个
#consumer获得了某个partition的消费权限,那么它将会向zk
#注册 "Partition Owner registry"节点信息,但是有可能
#此时旧的consumer尚没有释放此节点， 此值用于控制,
#注册节点的重试次数.
```

```
rebalance.max.retries=5

#每拉取一批消息的最大字节数
#获取消息的最大尺寸,broker不会像consumer输出大于
#此值的消息chunk 每次feth将得到多条消息,此值为总大小,
#提升此值,将会消耗更多的consumer端内存
fetch.min.bytes=6553600

#当消息的尺寸不足时,server阻塞的时间,如果超时,
#消息将立即发送给consumer
#数据一批一批到达，如果每一批是10条消息，如果某一批还
#不到10条，但是超时了，也会立即发送给consumer。
fetch.wait.max.ms=5000
socket.receive.buffer.bytes=655360

# 如果zookeeper没有offset值或offset值超出范围。
#那么就给个初始的offset。有smallest、largest、
#anything可选，分别表示给当前最小的offset、
#当前最大的offset、抛异常。默认largest
auto.offset.reset=smallest

# 指定序列化处理类
derializer.class=kafka.serializer.DefaultDecoder
```

(3).server.properties:服务端的配置文件

```
#broker的全局唯一编号，不能重复
broker.id=0

#用来监听链接的端口，producer或consumer将在此端口建立连接
port=9092

#处理网络请求的线程数量，也就是接收消息的线程数。
```

```
#接收线程会将接收到的消息放到内存中，然后再从内存中写入磁盘。
num.network.threads=3

#消息从内存中写入磁盘是时候使用的线程数量。
#用来处理磁盘IO的线程数量
num.io.threads=8

#发送套接字的缓冲区大小
socket.send.buffer.bytes=102400

#接受套接字的缓冲区大小
socket.receive.buffer.bytes=102400

#请求套接字的缓冲区大小
socket.request.max.bytes=104857600

#kafka运行日志存放的路径
log.dirs=/export/servers/logs/kafka

#topic在当前broker上的分片个数
num.partitions=2

#我们知道segment文件默认会被保留7天的时间，超时的话就
#会被清理，那么清理这件事情就需要有一些线程来做。这里就是
#用来设置恢复和清理data下数据的线程数量
num.recovery.threads.per.data.dir=1

#segment文件保留的最长时间，默认保留7天（168小时），
#超时将被删除，也就是说7天之前的数据将被清理掉。
log.retention.hours=168

#滚动生成新的segment文件的最大时间
log.roll.hours=168

#日志文件中每个segment的大小，默认为1G
log.segment.bytes=1073741824

#上面的参数设置了每一个segment文件的大小是1G，那么
#就需要有一个东西去定期检查segment文件有没有达到1G，
#多长时间去检查一次，就需要设置一个周期性检查文件大小
```

```
#的时间（单位是毫秒）。
log.retention.check.interval.ms=300000

#日志清理是否打开
log.cleaner.enable=true

#broker需要使用zookeeper保存meta数据
zookeeper.connect=zk01:2181,zk02:2181,zk03:2181

#zookeeper链接超时时间
zookeeper.connection.timeout.ms=6000

#上面我们说过接收线程会将接收到的消息放到内存中，然后再从内存
#写到磁盘上，那么什么时候将消息从内存中写入磁盘，就有一个
#时间限制（时间阈值）和一个数量限制（数量阈值），这里设置的是
#数量阈值，下一个参数设置的则是时间阈值。
#partion buffer中，消息的条数达到阈值，将触发flush到磁盘。
log.flush.interval.messages=10000

#消息buffer的时间，达到阈值，将触发将消息从内存flush到磁盘，
#单位是毫秒。
log.flush.interval.ms=3000

#删除topic需要server.properties中设置delete.topic.enable=true否则只是标记删除
delete.topic.enable=true

#此处的host.name为本机IP(重要),如果不改,则客户端会抛出:
#Producer connection to localhost:9092 unsuccessful 错误!
host.name=kafka01

advertised.host.name=192.168.239.128
```

如果觉得本文对您有帮助，不妨扫描下方微信二维码打赏点，您的鼓励是我前进最大的动力：