kafka 的配置文件的说明

Server.properties 配置文件说明

```
#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
num.network.threads=3

#用来处理磁盘 IO 的线程数量 页缓存功能 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 确定我们磁盘的路径 df -lh 多个磁盘路径用逗号隔开
log.dirs=/export/data/kafka/

#topic 在当前 broker 上的分片个数，(一般都是在创建 topic 的时候手动指定好了分区的个数)
num.partitions=2

#用来恢复和清理 data 下数据的线程数量 文件什么时候删除 168h 后 7 天后
num.recovery.threads.per.data.dir=1
```

```
#segment 文件保留的最长时间，超时将被删除
log.retention.hours=1

#滚动生成新的 segment 文件的最大时间
log.roll.hours=1

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

#新的 segemnt 文件生成的策略：第一个时间长短，一个小时生成一个新的

第二个：文件大小，segement 文件达到 1G 也生成新的文件

```
#周期性检查文件大小的时间
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

#partion buffer 中，消息的条数达到阈值，将触发 flush 到磁盘(页缓存里面有多少条数据 开始发)
log.flush.interval.messages=10000

#消息 buffer 的时间，达到阈值，将触发 flush 到磁盘
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=node01
```

```
advertised.host.name=192.168.52.100//广播地址 一般用不到
```

producer 生产者配置文件说明

生产数据的时候，尽量使用异步模式，可以提高数据生产的效率

```
#指定 kafka 节点列表，用于获取 metadata，不必全部指定
metadata.broker.list=node01:9092,node02:9092,node03:9092
# 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
# 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。
compression.codec=none
```

```
# 指定序列化处理类
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。
#compressed.topics=
```

```
# 设置发送数据是否需要服务端的反馈,有三个值 0,1,-1
# 0: producer 不会等待 broker 发送 ack
# 1: 当 leader 接收到消息之后发送 ack
# -1: 当所有的 follower 都同步消息成功后发送 ack.
request.required.acks=1

# 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
```

```
# 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

# 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
queue.buffering.max.messages=20000

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

# 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: 无阻塞超时限制,消息不会被抛弃
# 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
```

consumer 消费者配置详细说明

```
# 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
# 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(原来正在消费的线程保护期 看死没死透)
zookeeper.sync.time.ms=2000
#指定消费
group.id=qst
# 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  (没什么用)
# 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  (没什么用)
# 最大取多少块缓存到消费者(默认 10)
queued.max.message.chunks=50(尽量一次多去点儿)
```

```
# 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

# 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
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
```