Kafka安装指导手册

# 一、环境准备：

**Kafka:** kafka\_2.11-2.0.0.tgz

**Zookeeper：**zookeeper-3.4.13.tar.gz

**Centos 7.X三台:**

172.18.31.201,172.18.31.202,172.18.31.203

**Jdk:**请自行安装1.8以上的jdk环境

*备注：三台机器均使用kafka用户，用户目录/home/kafka*

*useradd -d /home/kafka -g users kafka*

*passwd kafka*

# 二、Zookeeper安装

下载地址：

<https://archive.apache.org/dist/zookeeper/>

## 2.1创建使用路径

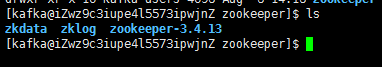
mkdir zookeeper

cd zookeeper

mkdir zkdata

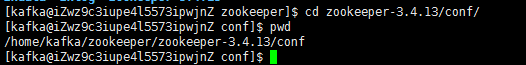
mkdir zklog

解压zookeeper-3-4.13.tar.gz到当前目录，目录结果如下图所示：



## 2.2 zookeeper配置

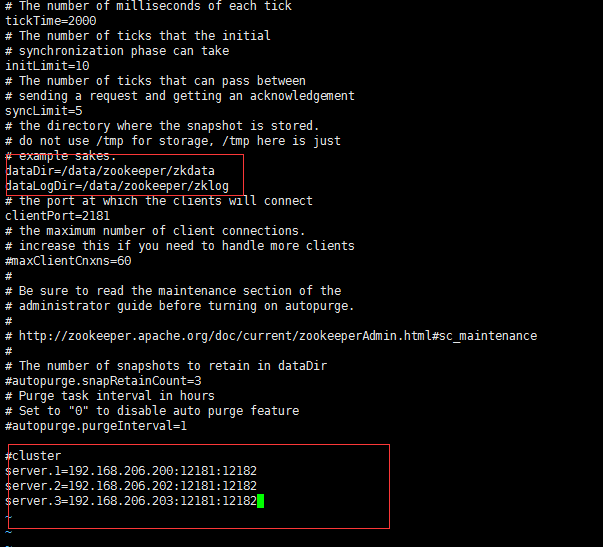
cd zookeeper-3.4.13/conf/



cp zoo\_sample.cfg zoo.cfg



vi zoo.cfg



a.修改数据路径

b.添加日志路径

c.集群配置

d.各项配置说明：

|  |
| --- |
| #tickTime：  这个时间是作为 Zookeeper 服务器之间或客户端与服务器之间维持心跳的时间间隔，也就是每个 tickTime 时间就会发送一个心跳。  #initLimit：  这个配置项是用来配置 Zookeeper 接受客户端（这里所说的客户端不是用户连接 Zookeeper 服务器的客户端，而是 Zookeeper 服务器集群中连接到 Leader 的 Follower 服务器）初始化连接时最长能忍受多少个心跳时间间隔数。当已经超过 5个心跳的时间（也就是 tickTime）长度后 Zookeeper 服务器还没有收到客户端的返回信息，那么表明这个客户端连接失败。总的时间长度就是 5\*2000=10 秒  #syncLimit：  这个配置项标识 Leader 与Follower 之间发送消息，请求和应答时间长度，最长不能超过多少个 tickTime 的时间长度，总的时间长度就是5\*2000=10秒  #dataDir：  快照日志的存储路径  #dataLogDir：  事物日志的存储路径，如果不配置这个那么事物日志会默认存储到dataDir制定的目录，这样会严重影响zk的性能，当zk吞吐量较大的时候，产生的事物日志、快照日志太多  #clientPort：  这个端口就是客户端连接 Zookeeper 服务器的端口，Zookeeper 会监听这个端口，接受客户端的访问请求。修改他的端口改大点  Eg：server.1=192.168.7.100:12888:13888  server.1=172.18.31.201:2888:3888  #server.1：  这个1是服务器的标识也可以是其他的数字， 表示这个是第几号服务器，用来标识服务器，这个标识要写到快照目录下面myid文件里  #192.168.7.107为集群里的IP地址，第一个端口是master和slave之间的通信端口，默认是2888，第二个端口是leader选举的端口，集群刚启动的时候选举或者leader挂掉之后进行新的选举的端口默认是3888 |

*注：zoo.cfg 文件是zookeeper配置文件 在conf目录里，初始提供的是zoo\_sample.cfg文件。*

## 2.3 创建myid文件

切换到zkdata目录



三台服务分别执行：

echo "1" > myid

echo "2" > myid

echo "3" > myid

1对应172.18.31.201服务器，2对应172.18.31.202服务器，3对应172.18.31.203服务器



注：myid文件和server.myid  在快照目录下存放的标识本台服务器的文件，他是整个zk集群用来发现彼此的一个重要标识

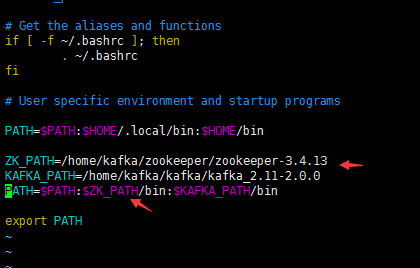
## 2.4 添加zk到环境变量中方便直接执行命令(可跳过本章节)

回到用户根目录更改.base\_profile文件



vi .bash\_profile

添加：ZK\_PATH=/home/kafka/zookeeper/zookeeper-3.4.13



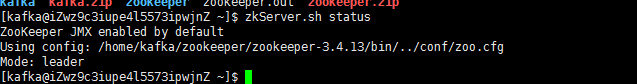
source .bash\_profile

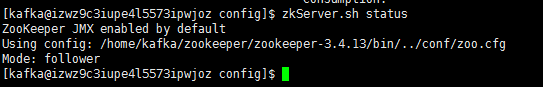
## ls2.5 分别启动三台服务器上的zk

zkServer.sh start

查看启动后状态

zkServer.sh status





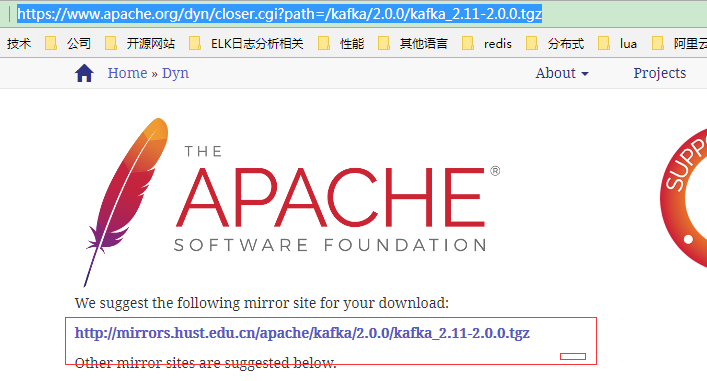
*注：*

*Mode:leader & follower 两种值*

# 三、kafka安装

## 3.1下载地址：

<https://www.apache.org/dyn/closer.cgi?path=/kafka/2.0.0/kafka_2.11-2.0.0.tgz>



可直接在linxu执行

wget <http://mirrors.hust.edu.cn/apache/kafka/2.0.0/kafka_2.11-2.0.0.tgz>

## 3.2 用户根目录下创建kafka目录

mkdir kafka

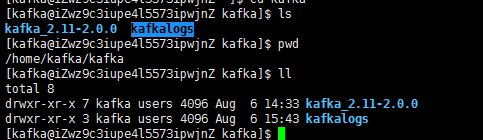


cd kafka

创建kafka日志目录

mkdir kafkalogs

下载kafka2.11解压到kafka目录，如下图结构



## 3.3环境变量配置，方便直接使用命令(可跳过)

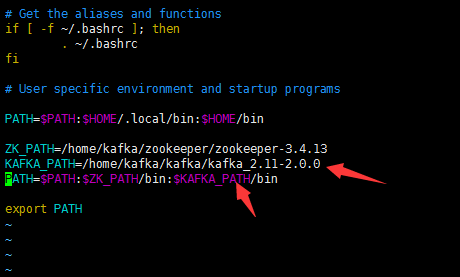
回到用户根目录编辑.bash\_profile文件



vi .bash\_profile

添加kafka目录

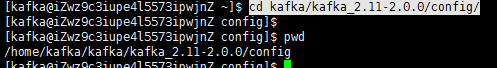
KAFKA\_PATH=/home/kafka/kafka/kafka\_2.11-2.0.0



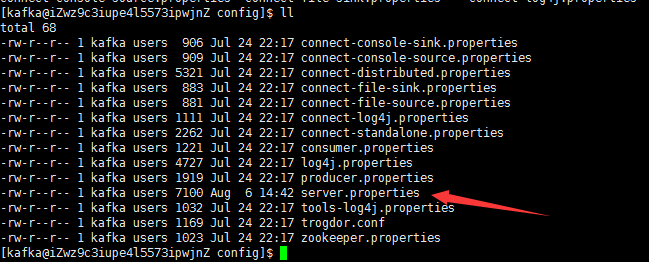
source .bash\_profile

## 3.4 kafka配置

cd kafka/kafka\_2.11-2.0.0/config/

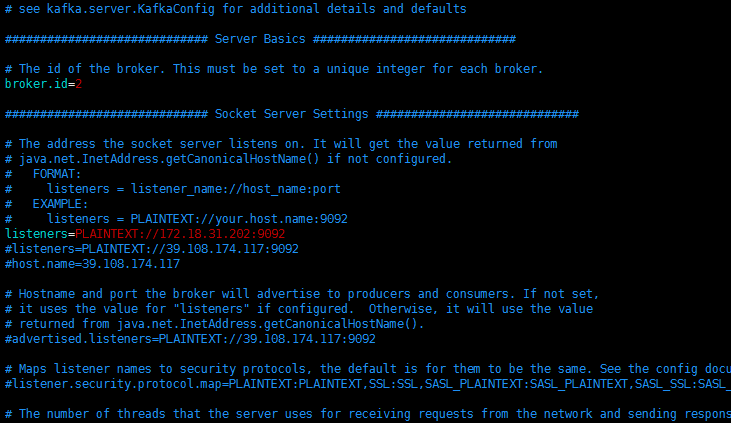


重点关注server.properties文件



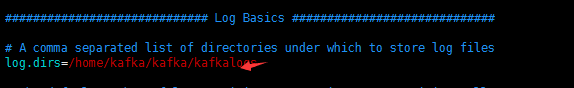
vi server.properties

主要修改地方如下几点：

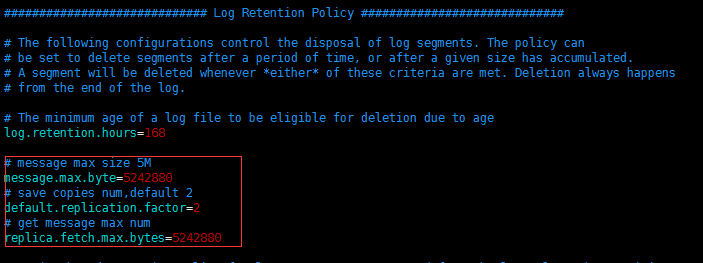


broker.id在集群内全局唯一

listeners kafka协议+IP+端口



log.dirs前步骤所创建的logs目录

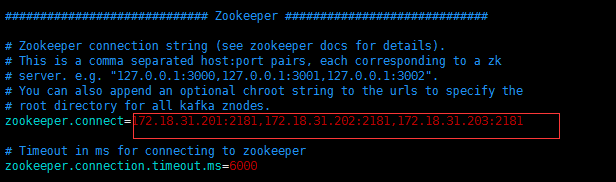


添加如下三项：

message.max.byte：消息保存最大值

default.repliaction.factor：消息副本数，如果其中一个失效，另一个还可以继续提供服务

replica.fetch.max.bytes：取消息的最大数



Zookeeper的配置

## 3.5 server.propeties配置文件各参数参考说明：

|  |
| --- |
| # Licensed to the Apache Software Foundation (ASF) under one or more  # contributor license agreements. See the NOTICE file distributed with  # this work for additional information regarding copyright ownership.  # The ASF licenses this file to You under the Apache License, Version 2.0  # (the "License"); you may not use this file except in compliance with  # the License. You may obtain a copy of the License at  #  # http://www.apache.org/licenses/LICENSE-2.0  #  # Unless required by applicable law or agreed to in writing, software  # distributed under the License is distributed on an "AS IS" BASIS,  # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  # See the License for the specific language governing permissions and  # limitations under the License.  # see kafka.server.KafkaConfig for additional details and defaults  ############################# Server Basics #############################  # The id of the broker. This must be set to a unique integer for each broker.  # 节点的ID，必须与其它节点不同  broker.id=0  # Switch to enable topic deletion or not, default value is false  # 选择启用删除主题功能，默认false  #delete.topic.enable=true  ############################# Socket Server Settings #############################  # The address the socket server listens on. It will get the value returned from  # java.net.InetAddress.getCanonicalHostName() if not configured.  # 套接字服务器坚挺的地址。如果没有配置，就使用java.net.InetAddress.getCanonicalHostName()的返回值  # FORMAT:  # listeners = listener\_name://host\_name:port  # EXAMPLE:  # listeners = PLAINTEXT://your.host.name:9092  #listeners=PLAINTEXT://:9092  # Hostname and port the broker will advertise to producers and consumers. If not set,  # it uses the value for "listeners" if configured. Otherwise, it will use the value  # returned from java.net.InetAddress.getCanonicalHostName().  # 节点的主机名会通知给生产者和消费者。如果没有设置，如果配置了"listeners"就使用"listeners"的值。  # 否则就使用java.net.InetAddress.getCanonicalHostName()的返回值  #advertised.listeners=PLAINTEXT://your.host.name:9092  # Maps listener names to security protocols, the default is for them to be the same. See the config documentation for more details  # 将侦听器的名称映射到安全协议，默认情况下它们是相同的。有关详细信息，请参阅配置文档  #listener.security.protocol.map=PLAINTEXT:PLAINTEXT,SSL:SSL,SASL\_PLAINTEXT:SASL\_PLAINTEXT,SASL\_SSL:SASL\_SSL  # The number of threads that the server uses for receiving requests from the network and sending responses to the network  # 服务器用来接受请求或者发送响应的线程数  num.network.threads=3  # The number of threads that the server uses for processing requests, which may include disk I/O  # 服务器用来处理请求的线程数，可能包括磁盘IO  num.io.threads=8  # The send buffer (SO\_SNDBUF) used by the socket server  # 套接字服务器使用的发送缓冲区大小  socket.send.buffer.bytes=102400  # The receive buffer (SO\_RCVBUF) used by the socket server  # 套接字服务器使用的接收缓冲区大小  socket.receive.buffer.bytes=102400  # The maximum size of a request that the socket server will accept (protection against OOM)  # 单个请求最大能接收的数据量  socket.request.max.bytes=104857600  ############################# Log Basics #############################  # A comma seperated list of directories under which to store log files  # 一个逗号分隔的目录列表，用来存储日志文件  log.dirs=/tmp/kafka-logs  # The default number of log partitions per topic. More partitions allow greater  # parallelism for consumption, but this will also result in more files across  # the brokers.  # 每个主题的日志分区的默认数量。更多的分区允许更大的并行操作，但是它会导致节点产生更多的文件  num.partitions=1  # The number of threads per data directory to be used for log recovery at startup and flushing at shutdown.  # This value is recommended to be increased for installations with data dirs located in RAID array.  # 每个数据目录中的线程数，用于在启动时日志恢复，并在关闭时刷新。  num.recovery.threads.per.data.dir=1  ############################# Internal Topic Settings #############################  # 内部主题设置  # The replication factor for the group metadata internal topics "\_\_consumer\_offsets" and "\_\_transaction\_state"  # For anything other than development testing, a value greater than 1 is recommended for to ensure availability such as 3.  # 对于除了开发测试之外的其他任何东西，group元数据内部主题的复制因子“\_\_consumer\_offsets”和“\_\_transaction\_state”，建议值大于1，以确保可用性(如3)。  offsets.topic.replication.factor=1  transaction.state.log.replication.factor=1  transaction.state.log.min.isr=1  ############################# Log Flush Policy #############################  # Messages are immediately written to the filesystem but by default we only fsync() to sync  # the OS cache lazily. The following configurations control the flush of data to disk.  # 消息直接被写入文件系统，但是默认情况下我们仅仅调用fsync()以延迟的同步系统缓存  # There are a few important trade-offs here:  # 这些有一些重要的权衡  # 1. Durability: Unflushed data may be lost if you are not using replication.  # 2. Latency: Very large flush intervals may lead to latency spikes when the flush does occur as there will be a lot of data to flush.  # 3. Throughput: The flush is generally the most expensive operation, and a small flush interval may lead to exceessive seeks.  # 1. 持久性:如果不使用复制，未刷新的数据可能会丢失。  # 2. 延迟:非常大的刷新间隔可能会在刷新时导致延迟，因为将会有大量数据刷新。  # 3. 吞吐量:刷新通常是最昂贵的操作，而一个小的刷新间隔可能会导致过多的搜索。  # The settings below allow one to configure the flush policy to flush data after a period of time or  # every N messages (or both). This can be done globally and overridden on a per-topic basis.  # 下面的设置允许你去配置刷新策略，每隔一段时间刷新或者一次N个消息（或者两个都配置）。这可以在全局范围内完成，并在每个主题的基础上重写。  # The number of messages to accept before forcing a flush of data to disk  # 在强制刷新数据到磁盘之前允许接收消息的数量  #log.flush.interval.messages=10000  # The maximum amount of time a message can sit in a log before we force a flush  # 在强制刷新之前，消息可以在日志中停留的最长时间  #log.flush.interval.ms=1000  ############################# Log Retention Policy #############################  # The following configurations control the disposal of log segments. The policy can  # be set to delete segments after a period of time, or after a given size has accumulated.  # 以下的配置控制了日志段的处理。策略可以配置为每隔一段时间删除片段或者到达一定大小之后。  # A segment will be deleted whenever \*either\* of these criteria are met. Deletion always happens  # from the end of the log.  # 当满足这些条件时，将会删除一个片段。删除总是发生在日志的末尾。  # The minimum age of a log file to be eligible for deletion  # 一个日志的最小存活时间，可以被删除  log.retention.hours=168  # A size-based retention policy for logs. Segments are pruned from the log as long as the remaining  # segments don't drop below log.retention.bytes.  # 一个基于大小的日志保留策略。段将被从日志中删除只要剩下的部分段不低于log.retention.bytes。  #log.retention.bytes=1073741824  # The maximum size of a log segment file. When this size is reached a new log segment will be created.  # 每一个日志段大小的最大值。当到达这个大小时，会生成一个新的片段。  log.segment.bytes=1073741824  # The interval at which log segments are checked to see if they can be deleted according  # to the retention policies  # 检查日志段的时间间隔，看是否可以根据保留策略删除它们  log.retention.check.interval.ms=300000  ############################# Zookeeper #############################  # Zookeeper connection string (see zookeeper docs for details).  # Zookeeper连接字符串（具体见Zookeeper文档）  # This is a comma separated host:port pairs, each corresponding to a zk  # 这是一个以逗号为分割的部分，每一个都匹配一个Zookeeper  # server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".  # You can also append an optional chroot string to the urls to specify the  # root directory for all kafka znodes.  # 您还可以将一个可选的chroot字符串附加到url，以指定所有kafka znode的根目录。  zookeeper.connect=localhost:2181  # Timeout in ms for connecting to zookeeper  # 连接到Zookeeper的超时时间  zookeeper.connection.timeout.ms=6000  ############################# Group Coordinator Settings #############################  # The following configuration specifies the time, in milliseconds, that the GroupCoordinator will delay the initial consumer rebalance.  # The rebalance will be further delayed by the value of group.initial.rebalance.delay.ms as new members join the group, up to a maximum of max.poll.interval.ms.  # The default value for this is 3 seconds.  # We override this to 0 here as it makes for a better out-of-the-box experience for development and testing.  # However, in production environments the default value of 3 seconds is more suitable as this will help to avoid unnecessary, and potentially expensive, rebalances during application startup.  group.initial.rebalance.delay.ms=0 |

## 3.6 启动集群(三台服务器同样的方式启动)

如进行环境变量的配置3.3章节，进入zookeeper/conf目录下启动



kafka-server-start.sh -daemon server.properties

## 3.6 测试

203服务器创建一个 Topic

kafka-topics.sh --create --zookeeper 172.18.31.202:2181 --replication-factor 2 --partitions 1 --topic omni-test

|  |
| --- |
| 集群版：  kafka-topics.sh --create --zookeeper 192.168.206.200:2181,192.168.206.201:2181,192.168.206.202:2181 --replication-factor 2 --partitions 1 --topic dev\_test |



|  |
| --- |
| #解释  --replication-factor 2 #复制两份  --partitions 1 #创建1个分区  --topic #主题为shuaige |

202服务器创一个 生产者

kafka-console-producer.sh --broker-list 172.18.31.202:9092 --topic omni

|  |
| --- |
| 集群版：  kafka-console-producer.sh --broker-list 192.168.206.200:9092,192.168.206.201:9092,192.168.206.202:9092 --topic |



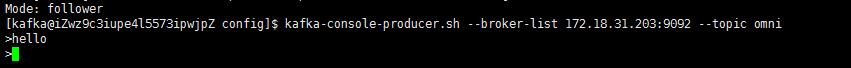
201服务器创建一个 消费者

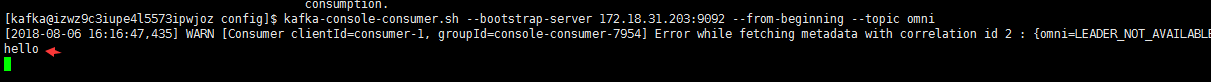
kafka-console-consumer.sh --bootstrap-server 172.18.31.202:9092 --from-beginning --topic omni

|  |
| --- |
| 集群版：  kafka-console-consumer.sh --bootstrap-server 192.168.206.200:9092,192.168.206.201:9092,192.168.206.202:9092 --from-beginning --topic dev\_test |



在生产端发送消息 hello，消费端即能收到消息：





至此，kafka集群搭建完成

## 3.7 遇到的问题

**案例使用的是阿里的内网地址，导致外网无法访问，即使设置一台外网IP在集群的环境下仍会堵到内网地址，导致外面无法发送数据到kafka，异常信息如下**：

**Springboot debug模式异常：**

|  |
| --- |
| 2018-08-07 17:00:16.616 DEBUG [omni-member-service,,,] [omni-member-service,,,] 9064 --- [ad | producer-2] o.apache.kafka.common.network.Selector : [Producer clientId=producer-2] Connection with /172.18.31.201 disconnected  java.net.ConnectException: Connection timed out: no further information  at sun.nio.ch.SocketChannelImpl.checkConnect(Native Method)  at sun.nio.ch.SocketChannelImpl.finishConnect(SocketChannelImpl.java:717)  at org.apache.kafka.common.network.PlaintextTransportLayer.finishConnect(PlaintextTransportLayer.java:50)  at org.apache.kafka.common.network.KafkaChannel.finishConnect(KafkaChannel.java:106)  at org.apache.kafka.common.network.Selector.pollSelectionKeys(Selector.java:458)  at org.apache.kafka.common.network.Selector.poll(Selector.java:412)  at org.apache.kafka.clients.NetworkClient.poll(NetworkClient.java:460)  at org.apache.kafka.clients.producer.internals.Sender.run(Sender.java:239)  at org.apache.kafka.clients.producer.internals.Sender.run(Sender.java:163)  at java.lang.Thread.run(Thread.java:748) |

发送日志到kafka异常：

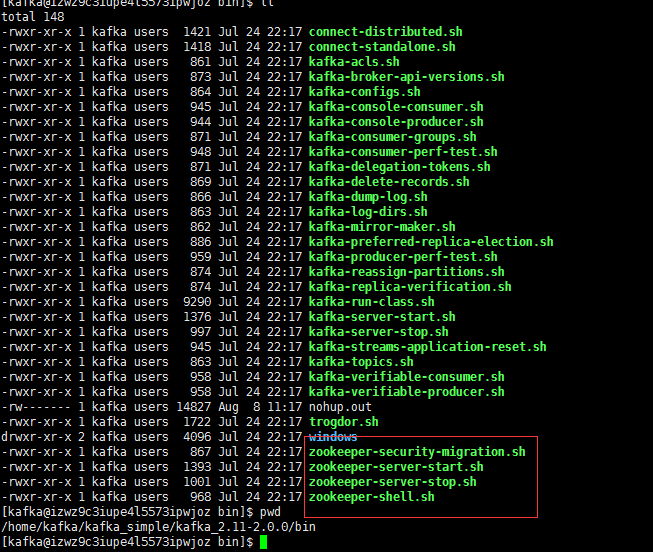
|  |
| --- |
| 2018-08-07 17:00:43.431 ERROR [omni-member-service,,,] [omni-member-service,,,] 9064 --- [ad | producer-2] o.s.k.support.LoggingProducerListener : Exception thrown when sending a message with key='null' and payload='hello,kafka from Producer test!!!!2018-08-07 17:00:13.392' to topic omni:  org.apache.kafka.common.errors.TimeoutException: Expiring 1 record(s) for omni-0: 30039 ms has passed since batch creation plus linger time |

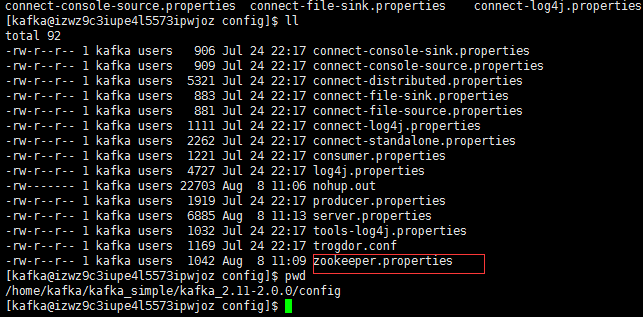
## 3.8 单机安装

***重点：阿里云外网接入，且zk和kafka需要的端口已经开通，本案(22181/19092)***

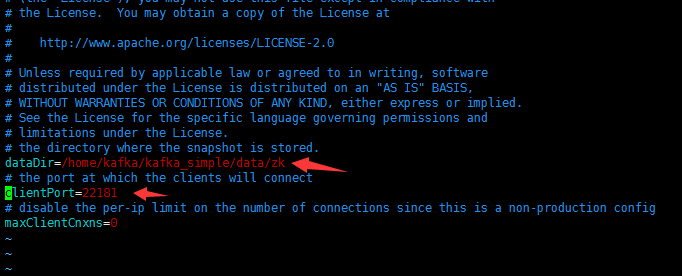
### 3.8.1 ZK单机配置和启动：

使用kafka安装包的zookeeper，kafka已提供zookeeper相关文件





修改zookeeper.properties文件配置zk的日志路径，如端口占用修改2181端口



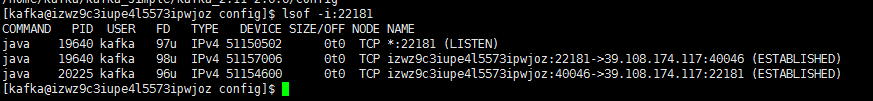
在当前目录下启动zk(后台启动)

nohup ./../bin/zookeeper-server-start.sh zookeeper.properties &



查看启动端口情况

lsof -i:22181

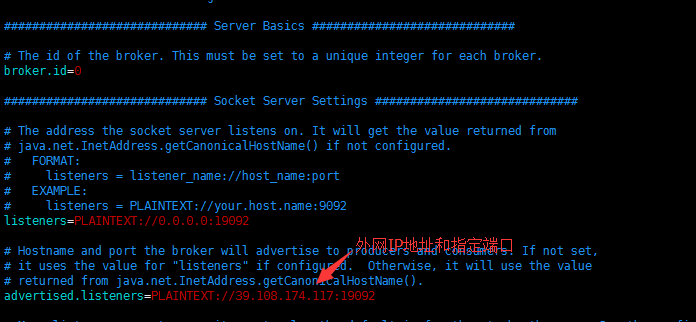


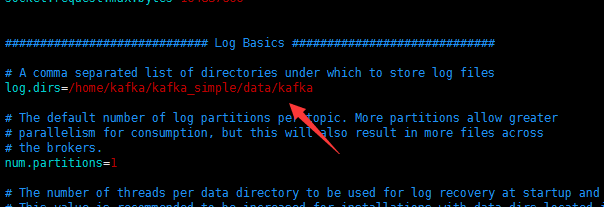
### 3.8.2 kafka单机配置

还是在当前conf目录下，修改server.properties文件



vi server.properties

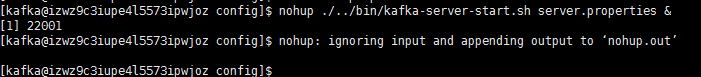






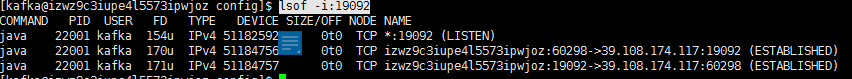
当前目录后台启动kafka

nohup ./../bin/kafka-server-start.sh server.properties &



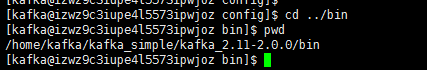
查看启动情况

lsof -i:19092



### 3.8.3 单机测试

回到kafka/bin目录



**1) 创建一个topic**

*要定：zk必须使用外网IP地址和端口，否则通过外网无法发送消息*

./kafka-topics.sh --create --zookeeper 39.108.174.117:22181 --replication-factor 1 --partitions 1 --topic kafkaTest



**2) 创建一个producer**

./kafka-console-producer.sh --broker-list 39.108.174.117:19092 --topic kafkaTest



**3) 创建一个 consumer**

./kafka-console-consumer.sh --bootstrap-server 39.108.174.117:19092 --topic kafkaTest --from-beginning



**4) springboot kafka代码测试**

**application.yml配置：**

|  |
| --- |
| spring:  kafka:  bootstrap-servers:  - 39.108.174.117:19092  consumer:  #默认组  group-id: test-group  key-deserializer:  org.apache.kafka.common.serialization.StringDeserializer  value-deserializer:  org.apache.kafka.common.serialization.StringDeserializer  #单节点不需要配置该参数  enable-auto-commit: **false**  auto-commit-interval: 100  auto-offset-reset: earliest  producer:  #批量发送消息  batch-size: 65536  buffer-memory: 524288  key-serializer:  org.apache.kafka.common.serialization.StringSerializer  value-serializer:  org.apache.kafka.common.serialization.StringSerializer |

**Producer 代码**

|  |
| --- |
| **import** java.time.LocalDateTime;  **import** java.time.format.DateTimeFormatter;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.kafka.core.KafkaTemplate;  **import** org.springframework.stereotype.Component;  @Component  **public** **class** KafkaProducerTest  {  @Autowired  **private** KafkaTemplate<String, String> kafkaTemplate;  /\*\*  \* 发送消息到kafka,主题为test  \*/  **public** **void** sendTest()  {  kafkaTemplate.send("kafkaTest",  "hello,kafka from Producer test!!!!" + LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*("yyyy-MM-dd HH:mm:ss.SSS")));  }  } |

**Consumer代码**

|  |
| --- |
| @KafkaListener(topics = "kafkaTest", id = "member-service")  **public** **void** listen(ConsumerRecord<?, ?> record) **throws** Exception  {  Optional<?> kafkaMessage = Optional.*ofNullable*(record.value());  **if** (kafkaMessage.isPresent())  {  Object message = kafkaMessage.get();  ***log***.info("----------------- record =" + record);  ***log***.info("------------------ re-message =" + message);  }  } |

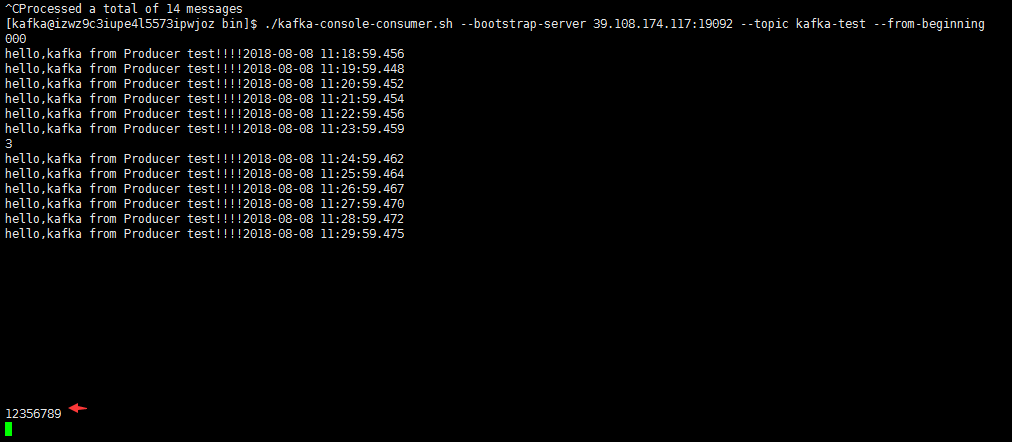
依赖

|  |
| --- |
| <!--kafka支持 -->  <dependency>  <groupId>org.springframework.kafka</groupId>  <artifactId>spring-kafka</artifactId>  </dependency> |

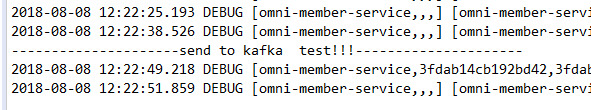
**测试结果：**

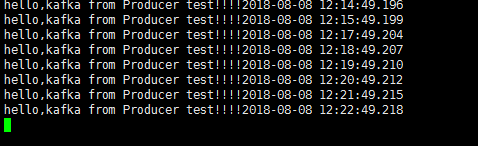
**1.控制台生成测试**





**2.springboot测试**





# 四、kafka web管理界面

简述：

使用雅虎开源的组件：https://github.com/yahoo/kafka-manager

## 4.1 下载

Github或git clone下载

## 4.2 sbt环境安装

curl https://bintray.com/sbt/rpm/rpm > bintray-sbt-rpm.repo

mv bintray-sbt-rpm.repo /etc/yum.repos.d/

yum install sbt

## 4.3 编辑打包

cd kafka-manager

sbt clean dist

注：安装过程很复杂加上网络原因，不一定能安装成功，会在target/universal下生成一个zip包。

另：本案是从网盘下载已经打包好的包：kafka-manager-1.3.3.7.zip

网盘地址：https://pan.baidu.com/s/1qYifoa4 密码：el4o

## 4.4 解压配置文件

unzip kafka-manager-1.3.3.7.zip

cd kafka-manager-1.3.3.7/conf

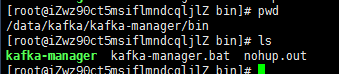
vi application.conf

修改如下：



## 4.5 启动

cd kafka-manager-1.3.3.7/bin



nohup ./kafka-manager

tips:

默认端口9000，修改端口命令如下:

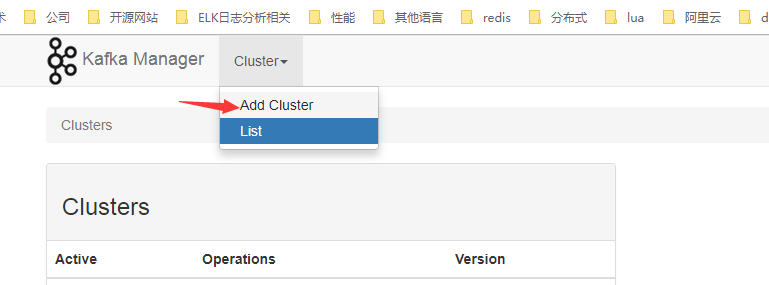
nohup bin/kafka-manager -Dconfig.file=conf/application.conf -Dhttp.port=8080 &

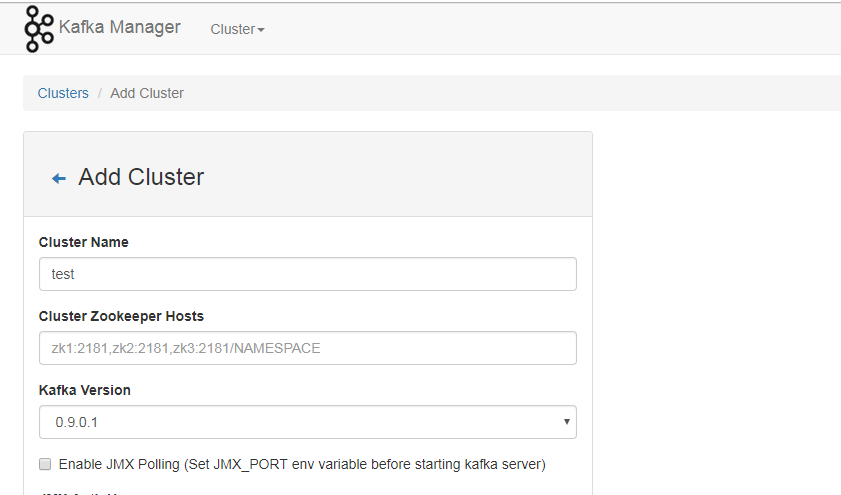
## 4.6 访问：

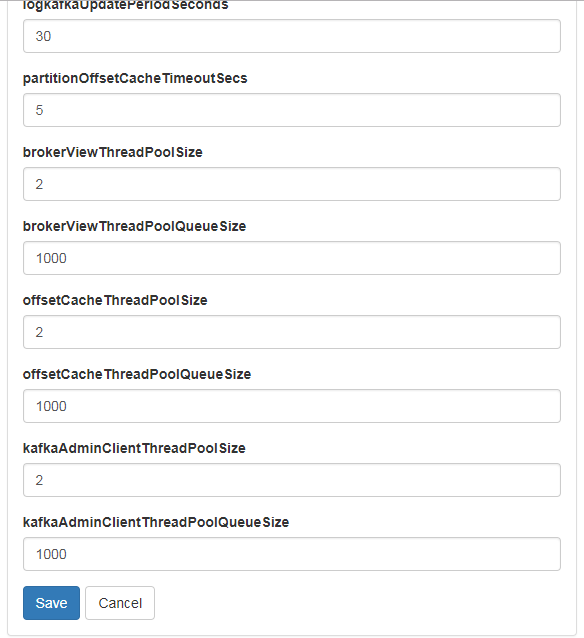
http://ip:9000



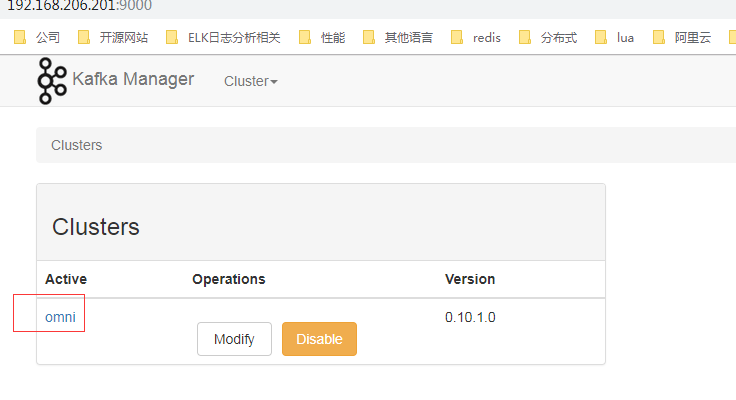
## 4.7 添加kafka集群







创建成功后，回到首页显示列表：



## 4.8 进入详情

