|  |
| --- |
| kafka和zookeeper安装 |
| **作者：张亚**  **归档：工作记录**  **2018/3/12** |
| **快捷键：**  Ctrl + 1 标题1  Ctrl + 2 标题2  Ctrl + 3 标题3  Ctrl + 4 实例  Ctrl + 5 程序代码  Ctrl + 6 正文 |
| **格式说明：**  蓝色字体：注释  黄色背景：重要  绿色背景：注意 |

目 录

[第1章 kafka简介 1](#_Toc508793698)

[第2章 安装环境准备 1](#_Toc508793699)

[第3章 下载安装并验证zookeeper 2](#_Toc508793700)

[3.1 zookeeper下载地址 2](#_Toc508793701)

[3.2 kafka下载地址 2](#_Toc508793702)

[3.3 安装zookeeper 2](#_Toc508793703)

[3.3.1 所有节点上传所有的软件包到指定目录 3](#_Toc508793704)

[3.3.2 节点1的配置 4](#_Toc508793705)

[3.3.3 节点2的配置 5](#_Toc508793706)

[3.3.4 节点3的配置 6](#_Toc508793707)

[3.3.5 各节点启动zookeeper 7](#_Toc508793708)

[3.3.6 查看各节点的zookeeper状态 7](#_Toc508793709)

[3.3.7 zookeeper简单操作命令 9](#_Toc508793710)

[3.4 安装并测试kafka 10](#_Toc508793711)

[3.4.1 节点1的配置 10](#_Toc508793712)

[3.4.2 节点2的配置 11](#_Toc508793713)

[3.4.3 节点3的配置 11](#_Toc508793714)

[3.4.4 各节点启动kafka 11](#_Toc508793715)

[3.4.5 验证进程 12](#_Toc508793716)

[3.4.6 测试创建topic 12](#_Toc508793717)

[3.4.7 测试获取toppid 13](#_Toc508793718)

[3.4.8 测试删除topic 13](#_Toc508793719)

[3.4.9 验证是否真的删除 13](#_Toc508793720)

[3.4.10 测试获取所有的topic列表 14](#_Toc508793721)

[3.4.11 kafka测试命令发送消息 14](#_Toc508793722)

[3.4.12 其他kafka服务器获取消息 15](#_Toc508793723)

[3.5 报错解决 15](#_Toc508793724)

[3.5.1 zookeeper配置文件里的server写错导致zookeeper状态为standalone 15](#_Toc508793725)

[3.5.2 发送消息失败 16](#_Toc508793726)

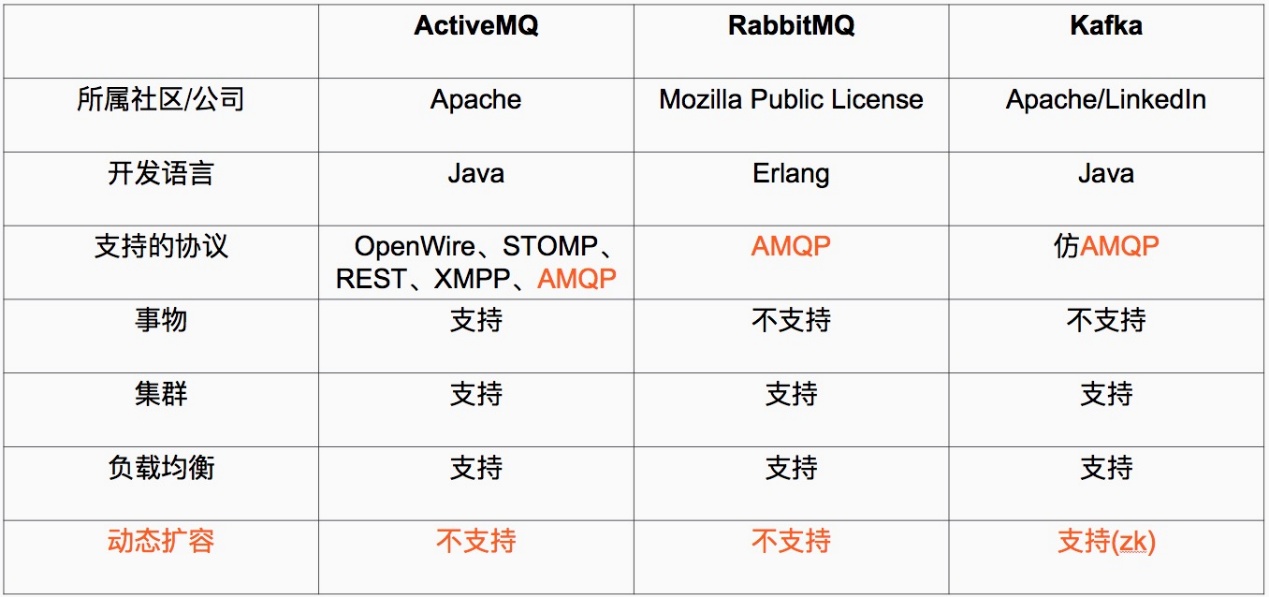
[3.5.3 接受消息失败报错 17](#_Toc508793727)

# kafka简介

Kafka 被称为下一代分布式消息系统，是非营利性组织ASF(Apache Software Foundation，简称为ASF)基金会中的一个开源项目，比如HTTP Server、Hadoop、ActiveMQ、Tomcat等开源软件都属于Apache基金会的开源软件，类似的消息系统还有RbbitMQ、ActiveMQ、ZeroMQ，最主要的优势是其具备分布式功能、并且结合zookeeper可以实现动态扩容。

相关链接介绍:

<http://www.infoq.com/cn/articles/apache-kafka>



# 安装环境准备

三台服务器配置hosts,并可以互相ping通,这里我选用debian8系统

[root@kafka70 ~]# vim /etc/hosts

[root@kafka70 ~]# cat /etc/hosts

127.0.0.1 localhost

10.0.0.200 debian

# The following lines are desirable for IPv6 capable hosts

::1 localhost ip6-localhost ip6-loopback

ff02::1 ip6-allnodes

ff02::2 ip6-allrouters

192.168.47.70 kafka70

192.168.47.71 kafka71

192.168.47.72 kafka72

[root@kafka70 ~]# ping kafka

PING kafka70 (192.168.47.70) 56(84) bytes of data.

64 bytes from kafka70 (192.168.47.70): icmp\_seq=1 ttl=64 time=0.019 ms

^C

--- kafka70 ping statistics ---

1 packets transmitted, 1 received, 0% packet loss, time 0ms

rtt min/avg/max/mdev = 0.019/0.019/0.019/0.000 ms

[root@kafka70 ~]# ping kafka71

PING kafka71 (192.168.47.71) 56(84) bytes of data.

64 bytes from kafka71 (192.168.47.71): icmp\_seq=1 ttl=64 time=0.197 ms

^C

--- kafka71 ping statistics ---

1 packets transmitted, 1 received, 0% packet loss, time 0ms

rtt min/avg/max/mdev = 0.197/0.197/0.197/0.000 ms

[root@kafka70 ~]# ping kafka72

PING kafka72 (192.168.47.72) 56(84) bytes of data.

64 bytes from kafka72 (192.168.47.72): icmp\_seq=1 ttl=64 time=0.505 ms

^C

--- kafka72 ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 1000ms

rtt min/avg/max/mdev = 0.435/0.470/0.505/0.035 ms

# 下载安装并验证zookeeper

## zookeeper下载地址

http://zookeeper.apache.org/releases.html

## kafka下载地址

http://kafka.apache.org/downloads.html

## 安装zookeeper

zookeeper集群特性:整个集群中只要有超过集群数量一半的zookeeper工作是正常的,那么整个集群对外就是可用的,例如有2台服务器做了一个zaookeeper,只要有任何一台故障或宕机,那么这个zookeeper集群就是不可用的了.因为剩下的一台没有超过集群的一半的数量,但是假如有三台zookeeper组成一个集群,那么损坏一台还剩两台,大于3台的一半,所以损坏一台还是可以正常运行的,但是再损坏一台就只剩下一台,集群就不可用了.

如果是4台组成,损坏一台正常,损坏两台还剩两台,不满足集群总数的一半,所以3台的集群和4台的集群算坏两台的结果都是集群不可用.所以这也是为什么集群一般是奇数的原因.

### 所有节点上传所有的软件包到指定目录

把所有的软件包都上传到/opt/soft目录,注意!所有节点都操作

[root@kafka70 ~]# mkdir /opt/soft

[root@kafka70 ~]# cd /opt/soft/

[root@kafka71 soft]# rz -E

rz waiting to receive.

[root@kafka70 soft]# ls -lh

total 264M

-rw-r--r-- 1 root root 181M Jan 9 18:33 jdk-8u151-linux-x64.tar.gz

-rw-r--r-- 1 root root 48M Dec 15 15:34 kafka\_2.11-1.0.0.tgz

-rw-r--r-- 1 root root 35M Dec 15 15:27 zookeeper-3.4.11.tar.gz

[root@kafka71 ~]# mkdir /opt/soft

[root@kafka71 ~]# cd /opt/soft/

[root@kafka71 soft]# rz -E

rz waiting to receive.

[root@kafka71 soft]# ls -lh

total 264M

-rw-r--r-- 1 root root 181M Jan 9 18:33 jdk-8u151-linux-x64.tar.gz

-rw-r--r-- 1 root root 48M Dec 15 15:34 kafka\_2.11-1.0.0.tgz

-rw-r--r-- 1 root root 35M Dec 15 15:27 zookeeper-3.4.11.tar.gz

[root@kafka72 ~]# mkdir /opt/soft

[root@kafka72 ~]# cd /opt/soft/

[root@kafka72 soft]# rz -E

rz waiting to receive.

[root@kafka72 soft]# ls -lh

total 264M

-rw-r--r-- 1 root root 181M Jan 9 18:33 jdk-8u151-linux-x64.tar.gz

-rw-r--r-- 1 root root 48M Dec 15 15:34 kafka\_2.11-1.0.0.tgz

-rw-r--r-- 1 root root 35M Dec 15 15:27 zookeeper-3.4.11.tar.gz

### 节点1的配置

安装配置java环境并确认

[root@kafka70 soft]# cd /opt/soft

[root@kafka70 soft]# tar zxf jdk-8u151-linux-x64.tar.gz -C /opt/

[root@kafka70 soft]# ln -s /opt/jdk1.8.0\_151/ /opt/jdk

[root@kafka70 soft]# sed -i.bak '$a export JAVA\_HOME=/opt/jdk\nexport PATH=$JAVA\_HOME/bin:$JAVA\_HOME/jre/bin:$PATH\nexport CLASSPATH=.$CLASSPATH:$JAVA\_HOME/lib:$JAVA\_HOME/jre/lib:$JAVA\_HOME/lib/tools.jar' /etc/profile

[root@kafka70 soft]# source /etc/profile

[root@kafka70 soft]# java -version

java version "1.8.0\_151"

Java(TM) SE Runtime Environment (build 1.8.0\_151-b12)

Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)

[root@kafka70 soft]# ls -lh /opt/

total 8.0K

lrwxrwxrwx 1 root root 18 Mar 12 14:05 jdk -> /opt/jdk1.8.0\_151/

drwxr-xr-x 8 uucp 143 4.0K Sep 6 2017 jdk1.8.0\_151

drwxr-xr-x 2 root root 4.0K Mar 12 13:53 soft

安装配置zookeeper

[root@kafka70 soft]# cd /opt/soft

[root@kafka70 soft]# tar zxf zookeeper-3.4.11.tar.gz -C /opt/

[root@kafka70 soft]# ln -s /opt/zookeeper-3.4.11/ /opt/zookeeper

[root@kafka70 soft]# tree -L 1 /opt/

/opt/

├── jdk -> /opt/jdk1.8.0\_151/

├── jdk1.8.0\_151

├── soft

├── zookeeper -> /opt/zookeeper-3.4.11/

└── zookeeper-3.4.11

5 directories, 0 files

[root@kafka70 soft]# mkdir -p /data/zookeeper

[root@kafka70 soft]# cp /opt/zookeeper/conf/zoo\_sample.cfg /opt/zookeeper/conf/zoo.cfg

[root@kafka70 soft]# vim /opt/zookeeper/conf/zoo.cfg

[root@kafka70 soft]# grep "^[a-Z]" /opt/zookeeper/conf/zoo.cfg

tickTime=2000

initLimit=10

syncLimit=5

dataDir=/data/zookeeper

clientPort=2181

server.1=192.168.47.70:2888:3888

server.2=192.168.47.71:2888:3888

server.3=192.168.47.72:2888:3888

[root@kafka70 soft]# echo "1" > /data/zookeeper/myid

[root@kafka70 soft]# ls -lh /data/zookeeper/

total 4.0K

-rw-r--r-- 1 root root 2 Mar 12 14:17 myid

[root@kafka70 soft]# cat /data/zookeeper/myid

1

### 节点2的配置

配置java: 步骤和节点1一样,只是最后myid不一样而已

[root@kafka71 soft]# cd /opt/soft

[root@kafka71 soft]# tar zxf jdk-8u151-linux-x64.tar.gz -C /opt/

[root@kafka71 soft]# ln -s /opt/jdk1.8.0\_151/ /opt/jdk

[root@kafka71 soft]# sed -i.bak '$a export JAVA\_HOME=/opt/jdk\nexport PATH=$JAVA\_HOME/bin:$JAVA\_HOME/jre/bin:$PATH\nexport CLASSPATH=.$CLASSPATH:$JAVA\_HOME/lib:$JAVA\_HOME/jre/lib:$JAVA\_HOME/lib/tools.jar' /etc/profile

[root@kafka71 soft]# source /etc/profile

[root@kafka71 soft]# java -version

java version "1.8.0\_151"

Java(TM) SE Runtime Environment (build 1.8.0\_151-b12)

Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)

配置zookeeper

[root@kafka71 soft]# cd /opt/soft

[root@kafka71 soft]# tar zxf zookeeper-3.4.11.tar.gz -C /opt/

[root@kafka71 soft]# ln -s /opt/zookeeper-3.4.11/ /opt/zookeeper

[root@kafka71 soft]# mkdir -p /data/zookeeper

[root@kafka71 soft]# cp /opt/zookeeper/conf/zoo\_sample.cfg /opt/zookeeper/conf/zoo.cfg

[root@kafka71 soft]# vim /opt/zookeeper/conf/zoo.cfg

[root@kafka71 soft]# grep "^[a-Z]" /opt/zookeeper/conf/zoo.cfg

tickTime=2000

initLimit=10

syncLimit=5

dataDir=/data/zookeeper

clientPort=2181

server.1=192.168.47.70:2888:3888

server.2=192.168.47.71:2888:3888

server.3=192.168.47.72:2888:3888

[root@kafka71 soft]# echo "2" > /data/zookeeper/myid

[root@kafka71 soft]# cat /data/zookeeper/myid

2

[root@kafka71 soft]# tree -L 1 /opt/

/opt/

├── jdk -> /opt/jdk1.8.0\_151/

├── jdk1.8.0\_151

├── soft

├── zookeeper -> /opt/zookeeper-3.4.11/

└── zookeeper-3.4.11

### 节点3的配置

配置java

[root@kafka72 soft]# cd /opt/soft

[root@kafka72 soft]# tar zxf jdk-8u151-linux-x64.tar.gz -C /opt/

[root@kafka72 soft]# ln -s /opt/jdk1.8.0\_151/ /opt/jdk

[root@kafka72 soft]# sed -i.bak '$a export JAVA\_HOME=/opt/jdk\nexport PATH=$JAVA\_HOME/bin:$JAVA\_HOME/jre/bin:$PATH\nexport CLASSPATH=.$CLASSPATH:$JAVA\_HOME/lib:$JAVA\_HOME/jre/lib:$JAVA\_HOME/lib/tools.jar' /etc/profile

[root@kafka72 soft]# vim /etc/profile

[root@kafka72 soft]# source /etc/profile

[root@kafka72 soft]# java -version

java version "1.8.0\_151"

Java(TM) SE Runtime Environment (build 1.8.0\_151-b12)

Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)

配置zookeeper

[root@kafka72 soft]# cd /opt/soft

[root@kafka72 soft]# tar zxf zookeeper-3.4.11.tar.gz -C /opt/

[root@kafka72 soft]# ln -s /opt/zookeeper-3.4.11/ /opt/zookeeper

[root@kafka72 soft]# mkdir -p /data/zookeeper

[root@kafka72 soft]# cp /opt/zookeeper/conf/zoo\_sample.cfg /opt/zookeeper/conf/zoo.cfg

[root@kafka72 soft]# vim /opt/zookeeper/conf/zoo.cfg

[root@kafka72 soft]# grep "^[a-Z]" /opt/zookeeper/conf/zoo.cfg

tickTime=2000

initLimit=10

syncLimit=5

dataDir=/tmp/zookeeper

clientPort=2181

server.1=192.168.47.70:2888:3888

server.2=192.168.47.71:2888:3888

server.3=192.168.47.72:2888:3888

[root@kafka72 soft]# echo "3" > /data/zookeeper/myid

[root@kafka72 soft]# cat /data/zookeeper/myid

3

[root@kafka72 soft]# tree -L 1 /opt/

/opt/

├── jdk -> /opt/jdk1.8.0\_151/

├── jdk1.8.0\_151

├── soft

├── zookeeper -> /opt/zookeeper-3.4.11/

└── zookeeper-3.4.11

### 各节点启动zookeeper

节点1

[root@kafka70 ~]# /opt/zookeeper/bin/zkServer.sh start

ZooKeeper JMX enabled by default

Using config: /opt/zookeeper/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

节点2

[root@kafka71 soft]# /opt/zookeeper/bin/zkServer.sh start

ZooKeeper JMX enabled by default

Using config: /opt/zookeeper/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

节点3

[root@kafka72 ~]# /opt/zookeeper/bin/zkServer.sh start

ZooKeeper JMX enabled by default

Using config: /opt/zookeeper/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

### 查看各节点的zookeeper状态

节点1

[root@kafka70 ~]# /opt/zookeeper/bin/zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /opt/zookeeper/bin/../conf/zoo.cfg

Mode: follower

节点2

[root@kafka71 soft]# /opt/zookeeper/bin/zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /opt/zookeeper/bin/../conf/zoo.cfg

Mode: leader

节点3

[root@kafka72 ~]# /opt/zookeeper/bin/zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /opt/zookeeper/bin/../conf/zoo.cfg

Mode: follower

### zookeeper简单操作命令

连接到任意节点生成数据:

我们在节点1生成数据,然后在其他节点验证数据

[root@kafka70 ~]# /opt/zookeeper/bin/zkCli.sh -server 192.168.47.70:2181

Connecting to 192.168.47.70:2181

=================

WATCHER::

WatchedEvent state:SyncConnected type:None path:null

[zk: 192.168.47.70:2181(CONNECTED) 0] create /test "hello"

Created /test

[zk: 192.168.47.70:2181(CONNECTED) 1]

在其他节点上验证数据

[root@kafka71 ~]# /opt/zookeeper/bin/zkCli.sh -server 192.168.47.71:2181

Connecting to 192.168.47.71:2181

===========================

WATCHER::

WatchedEvent state:SyncConnected type:None path:null

[zk: 192.168.47.71:2181(CONNECTED) 0] get /test

hello

cZxid = 0x100000002

ctime = Mon Mar 12 15:15:52 CST 2018

mZxid = 0x100000002

mtime = Mon Mar 12 15:15:52 CST 2018

pZxid = 0x100000002

cversion = 0

dataVersion = 0

aclVersion = 0

ephemeralOwner = 0x0

dataLength = 5

numChildren = 0

[zk: 192.168.47.71:2181(CONNECTED) 1]

在节点3上查看

[root@kafka72 ~]# /opt/zookeeper/bin/zkCli.sh -server 192.168.47.72:2181

Connecting to 192.168.47.72:2181

===========================

WATCHER::

WatchedEvent state:SyncConnected type:None path:null

[zk: 192.168.47.72:2181(CONNECTED) 0] get /test

hello

cZxid = 0x100000002

ctime = Mon Mar 12 15:15:52 CST 2018

mZxid = 0x100000002

mtime = Mon Mar 12 15:15:52 CST 2018

pZxid = 0x100000002

cversion = 0

dataVersion = 0

aclVersion = 0

ephemeralOwner = 0x0

dataLength = 5

numChildren = 0

[zk: 192.168.47.72:2181(CONNECTED) 1]

## 安装并测试kafka

### 节点1的配置

[root@kafka70 ~]# cd /opt/soft/

[root@kafka70 soft]# tar zxf kafka\_2.11-1.0.0.tgz -C /opt/

[root@kafka70 soft]# ln -s /opt/kafka\_2.11-1.0.0/ /opt/kafka

[root@kafka70 soft]# mkdir /opt/kafka/logs

[root@kafka70 soft]# vim /opt/kafka/config/server.properties

21 broker.id=1

31 listeners=PLAINTEXT://192.168.47.70:9092

60 log.dirs=/opt/kafka/logs

103 log.retention.hours=24   
123 zookeeper.connect=192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181

### 节点2的配置

[root@kafka71 ~]# cd /opt/soft/

[root@kafka71 soft]# tar zxf kafka\_2.11-1.0.0.tgz -C /opt/

[root@kafka71 soft]# ln -s /opt/kafka\_2.11-1.0.0/ /opt/kafka

[root@kafka71 soft]# mkdir /opt/kafka/logs

[root@kafka71 soft]# vim /opt/kafka/config/server.properties

21 broker.id=2

31 listeners=PLAINTEXT://192.168.47.71:9092

60 log.dirs=/opt/kafka/logs

103 log.retention.hours=24   
123 zookeeper.connect=192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181

### 节点3的配置

[root@kafka72 ~]# cd /opt/soft/

[root@kafka72 soft]# tar zxf kafka\_2.11-1.0.0.tgz -C /opt/

[root@kafka72 soft]# ln -s /opt/kafka\_2.11-1.0.0/ /opt/kafka

[root@kafka72 soft]# mkdir /opt/kafka/logs

[root@kafka72 soft]# vim /opt/kafka/config/server.properties

21 broker.id=3

31 listeners=PLAINTEXT://192.168.47.72:9092

60 log.dirs=/opt/kafka/logs

103 log.retention.hours=24   
123 zookeeper.connect=192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181

### 各节点启动kafka

节点1,可以先前台启动,方便查看错误日志

[root@kafka70 soft]# /opt/kafka/bin/kafka-server-start.sh /opt/kafka/config/server.properties

===========================

[2018-03-14 11:04:05,397] INFO Kafka version : 1.0.0 (org.apache.kafka.common.utils.AppInfoParser)

[2018-03-14 11:04:05,397] INFO Kafka commitId : aaa7af6d4a11b29d (org.apache.kafka.common.utils.AppInfoParser)

[2018-03-14 11:04:05,414] INFO [KafkaServer id=1] started (kafka.server.KafkaServer)

最后一行出现KafkaServer id和started字样,就表明启动成功了,然后就可以放到后台启动了

[root@kafka70 logs]# /opt/kafka/bin/kafka-server-start.sh -daemon /opt/kafka/config/server.properties

[root@kafka70 logs]# tail -f /opt/kafka/logs/server.log

=========================

[2018-03-14 11:04:05,414] INFO [KafkaServer id=1] started (kafka.server.KafkaServer)

节点2,我们这次直接后台启动然后查看日志

[root@kafka71 kafka]# /opt/kafka/bin/kafka-server-start.sh -daemon /opt/kafka/config/server.properties

[root@kafka71 kafka]# tail -f /opt/kafka/logs/server.log

====================================

[2018-03-14 11:04:13,679] INFO [KafkaServer id=2] started (kafka.server.KafkaServer)

节点3,一样后台启动然后查看日志

[root@kafka72 kafka]# /opt/kafka/bin/kafka-server-start.sh -daemon /opt/kafka/config/server.properties

[root@kafka72 kafka]# tail -f /opt/kafka/logs/server.log

=======================================

[2018-03-14 11:06:38,274] INFO [KafkaServer id=3] started (kafka.server.KafkaServer)

### 验证进程

节点1

[root@kafka70 ~]# /opt/jdk/bin/jps

4531 Jps

4334 Kafka

1230 QuorumPeerMain

节点2

[root@kafka71 kafka]# /opt/jdk/bin/jps

2513 Kafka

2664 Jps

1163 QuorumPeerMain

节点3

[root@kafka72 kafka]# /opt/jdk/bin/jps

2835 Jps

2728 Kafka

1385 QuorumPeerMain

### 测试创建topic

创建名为kafkatest，partitions(分区)为3，replication(复制)为3的topic(主题),在任意机器操作即可

[root@kafka70 ~]# /opt/kafka/bin/kafka-topics.sh --create --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --partitions 3 --replication-factor 3 --topic kafkatest

Created topic "kafkatest".

**Kafka2.13-3.1.0版本**

[root@kafka70 ~]#/opt/kafka/bin/kafka-topics.sh --create --bootstrap-server 192.168.91.55:9092,192.168.91.56:9092,192.168.91.57:9092 --partitions 3 --replication-factor 3 --topic kafkatest

### 测试获取toppid

可以在任意一台kafka服务器进行测试

节点1

[root@kafka70 ~]# /opt/kafka/bin/kafka-topics.sh --describe --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --topic kafkatest

Topic:kafkatest PartitionCount:3 ReplicationFactor:3 Configs:

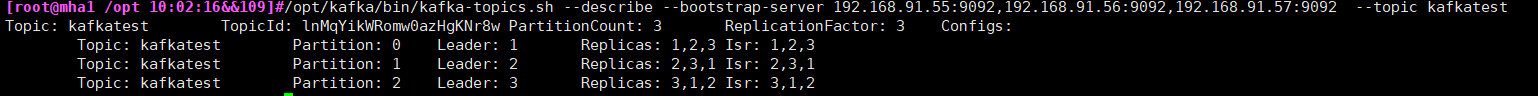
Topic: kafkatest Partition: 0 Leader: 2 Replicas: 2,3,1 Isr: 2,3,1

Topic: kafkatest Partition: 1 Leader: 3 Replicas: 3,1,2 Isr: 3,1,2

Topic: kafkatest Partition: 2 Leader: 1 Replicas: 1,2,3 Isr: 1,2,3

**Kafka2.13-3.1.0版本**

[root@kafka70 ~]#/opt/kafka/bin/kafka-topics.sh --describe --bootstrap-server 192.168.91.55:9092,192.168.91.56:9092,192.168.91.57:9092 --topic kafkatest



节点2

[root@kafka71 ~]# /opt/kafka/bin/kafka-topics.sh --describe --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --topic kafkatest

Topic:kafkatest PartitionCount:3 ReplicationFactor:3 Configs:

Topic: kafkatest Partition: 0 Leader: 2 Replicas: 2,3,1 Isr: 2,3,1

Topic: kafkatest Partition: 1 Leader: 3 Replicas: 3,1,2 Isr: 3,1,2

Topic: kafkatest Partition: 2 Leader: 1 Replicas: 1,2,3 Isr: 1,2,3

**Kafka2.13-3.1.0版本**

[root@kafka70 ~]#/opt/kafka/bin/kafka-topics.sh --describe --bootstrap-server 192.168.91.55:9092,192.168.91.56:9092,192.168.91.57:9092 --topic kafkatest

节点3

[root@kafka72 ~]# /opt/kafka/bin/kafka-topics.sh --describe --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --topic kafkatest

Topic:kafkatest PartitionCount:3 ReplicationFactor:3 Configs:

Topic: kafkatest Partition: 0 Leader: 2 Replicas: 2,3,1 Isr: 2,3,1

Topic: kafkatest Partition: 1 Leader: 3 Replicas: 3,1,2 Isr: 3,1,2

Topic: kafkatest Partition: 2 Leader: 1 Replicas: 1,2,3 Isr: 1,2,3

**Kafka2.13-3.1.0版本**

[root@kafka70 ~]#/opt/kafka/bin/kafka-topics.sh --describe --bootstrap-server 192.168.91.55:9092,192.168.91.56:9092,192.168.91.57:9092 --topic kafkatest

状态说明：kafkatest有三个分区分别为1、2、3，分区0的leader是2（broker.id），分区0有三个副本，并且状态都为lsr（ln-sync，表示可以参加选举成为leader）。

### 测试删除topic

[root@kafka70 ~]# /opt/kafka/bin/kafka-topics.sh --delete --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --topic kafkatest

Topic kafkatest is marked for deletion.

Note: This will have no impact if delete.topic.enable is not set to true.

**Kafka2.13-3.1.0版本**

/opt/kafka1/bin/kafka-topics.sh --delete --bootstrap-server 172.28.10.13:9091,172.28.10.13:9092,172.28.10.13:9093 --topic kafkatest

### 验证是否真的删除

[root@kafka70 ~]# /opt/kafka/bin/kafka-topics.sh --describe --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --topic kafkatest

[root@kafka70 ~]#

### 测试获取所有的topic列表

首先创建两个topic

[root@kafka70 ~]# /opt/kafka/bin/kafka-topics.sh --create --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --partitions 3 --replication-factor 3 --topic kafkatest

Created topic "kafkatest".

[root@kafka70 ~]# /opt/kafka/bin/kafka-topics.sh --create --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --partitions 3 --replication-factor 3 --topic kafkatest2

Created topic "kafkatest2".

然后查看所有的topic列表

[root@kafka70 ~]# /opt/kafka/bin/kafka-topics.sh --list --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181

kafkatest

kafkatest2

**Kafka2.13-3.1.0版本**

kafka-topics.sh --list --bootstrap-server 172.28.10.13:9091,172.28.10.13:9092,172.28.10.13:9093

[root@kafka70 ~]#

### kafka测试命令发送消息

创建一个名为messagetest的topic

[root@kafka70 ~]# /opt/kafka/bin/kafka-topics.sh --create --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --partitions 3 --replication-factor 3 --topic messagetest

Created topic "messagetest".

发送消息:注意,端口是 kafka的9092,而不是zookeeper的2181

[root@kafka70 ~]# /opt/kafka/bin/kafka-console-producer.sh --broker-list 192.168.47.70:9092,192.168.47.71:9092,192.168.47.72:9092 --topic messagetest

>hello

>mymy

>Yo!

>

### 其他kafka服务器获取消息

[root@kafka70 ~]# /opt/kafka/bin/kafka-console-consumer.sh --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --topic messagetest --from-beginning

Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-server] instead of [zookeeper].

mymy

Yo!

Hello

**Kafka2.13-3.1.0版本**

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server 192.168.91.55:9092,192.168.91.56:9092,192.168.91.57:9092 --topic messagetest --from-beginning

[root@kafka71 ~]# /opt/kafka/bin/kafka-console-consumer.sh --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --topic messagetest --from-beginning

Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-server] instead of [zookeeper].

mymy

Yo!

Hello

**Kafka2.13-3.1.0版本**

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server 192.168.91.55:9092,192.168.91.56:9092,192.168.91.57:9092 --topic messagetest --from-beginning

[root@kafka72 ~]# /opt/kafka/bin/kafka-console-consumer.sh --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --topic messagetest --from-beginning

Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-server] instead of [zookeeper].

hello

mymy

Yo!

**Kafka2.13-3.1.0版本**

/opt/kafka/bin/kafka-console-consumer.sh --bootstrap-server 192.168.91.55:9092,192.168.91.56:9092,192.168.91.57:9092 --topic messagetest --from-beginning

## 报错解决

### zookeeper配置文件里的server写错导致zookeeper状态为standalone

配置文件里zoo.cfg里的server地址写错了,导致启动的时候只会查找自己的节点

[root@kafka70 soft]# grep "^[a-Z]" /opt/zookeeper/conf/zoo.cfg

tickTime=2000

initLimit=10

syncLimit=5

dataDir=/data/zookeeper

clientPort=2181

server.1=192.168.47.70:2888:3888

server.1=192.168.47.71:2888:3888

server.1=192.168.47.72:2888:3888

[root@kafka70 ~]# /opt/zookeeper/bin/zkServer.sh start

ZooKeeper JMX enabled by default

Using config: /opt/zookeeper/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

[root@kafka70 ~]# /opt/zookeeper/bin/zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /opt/zookeeper/bin/../conf/zoo.cfg

Mode: standalone

解决:各节点修改标签为正确的数字,然后重启zookeeper服务,注意!所有节点都要操作!

[root@kafka70 soft]# grep "^[a-Z]" /opt/zookeeper/conf/zoo.cfg

tickTime=2000

initLimit=10

syncLimit=5

dataDir=/data/zookeeper

clientPort=2181

server.1=192.168.47.70:2888:3888

server.2=192.168.47.71:2888:3888

server.3=192.168.47.72:2888:3888

[root@kafka70 soft]# /opt/zookeeper/bin/zkServer.sh restart

ZooKeeper JMX enabled by default

Using config: /opt/zookeeper/bin/../conf/zoo.cfg

ZooKeeper JMX enabled by default

Using config: /opt/zookeeper/bin/../conf/zoo.cfg

Stopping zookeeper ... STOPPED

ZooKeeper JMX enabled by default

Using config: /opt/zookeeper/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

[root@kafka71 soft]# /opt/zookeeper/bin/zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /opt/zookeeper/bin/../conf/zoo.cfg

Mode: follower

### 发送消息失败

[root@kafka70 ~]# /opt/kafka/bin/kafka-console-producer.sh --broker-list 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --topic messagetest

>hellp

mymy

meme

[2018-03-14 11:47:31,269] ERROR Error when sending message to topic messagetest with key: null, value: 5 bytes with error: (org.apache.kafka.clients.producer.internals.ErrorLoggingCallback)

org.apache.kafka.common.errors.TimeoutException: Failed to update metadata after 60000 ms.

>hello

[2018-03-14 11:48:31,277] ERROR Error when sending message to topic messagetest with key: null, value: 0 bytes with error: (org.apache.kafka.clients.producer.internals.ErrorLoggingCallback)

org.apache.kafka.common.errors.TimeoutException: Failed to update metadata after 60000 ms.

>

报错原因.端口写错了,应该是kafka的9092,而不是zookeeper的2181

解决:使用正确的端口

[root@kafka70 ~]# /opt/kafka/bin/kafka-console-producer.sh --broker-list 192.168.47.70:9092,192.168.47.71:9092,192.168.47.72:9092 --topic messagetest

>hello

>mymy

>Yo!

>

### 接受消息失败报错

[root@kafka71 ~]# /opt/kafka/bin/kafka-console-consumer.sh --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --topic messagetest --from-beginning

Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-server] instead of [zookeeper].

[2018-03-14 12:02:01,648] ERROR Unknown error when running consumer: (kafka.tools.ConsoleConsumer$)

java.net.UnknownHostException: kafka71: kafka71: Name or service not known

at java.net.InetAddress.getLocalHost(InetAddress.java:1505)

at kafka.consumer.ZookeeperConsumerConnector.<init>(ZookeeperConsumerConnector.scala:135)

at kafka.consumer.ZookeeperConsumerConnector.<init>(ZookeeperConsumerConnector.scala:159)

at kafka.consumer.Consumer$.create(ConsumerConnector.scala:112)

at kafka.consumer.OldConsumer.<init>(BaseConsumer.scala:130)

at kafka.tools.ConsoleConsumer$.run(ConsoleConsumer.scala:72)

at kafka.tools.ConsoleConsumer$.main(ConsoleConsumer.scala:54)

at kafka.tools.ConsoleConsumer.main(ConsoleConsumer.scala)

Caused by: java.net.UnknownHostException: kafka71: Name or service not known

at java.net.Inet6AddressImpl.lookupAllHostAddr(Native Method)

at java.net.InetAddress$2.lookupAllHostAddr(InetAddress.java:928)

at java.net.InetAddress.getAddressesFromNameService(InetAddress.java:1323)

at java.net.InetAddress.getLocalHost(InetAddress.java:1500)

... 7 more

报错原因:主机名和hosts解析名不一致

[root@kafka71 ~]# cat /etc/hostname

kafka71

[root@kafka71 ~]# tail -3 /etc/hosts

192.168.47.70 kafka70

192.168.47.71 kafka71

192.168.47.72 kafka72

解决办法:所有主机的主机名和hosts解析名保持一致,然后重新获取  
修改所有主机的主机名

[root@kafka70 ~]# hostname

kafka70

[root@kafka70 ~]# tail -3 /etc/hosts

192.168.47.70 kafka70

192.168.47.71 kafka71

192.168.47.72 kafka72

[root@kafka71 ~]# hostname

kafka71

[root@kafka71 ~]# tail -3 /etc/hosts

192.168.47.70 kafka70

192.168.47.71 kafka71

192.168.47.72 kafka72

[root@kafka72 ~]# hostname

kafka72

[root@kafka72 ~]# tail -3 /etc/hosts

192.168.47.70 kafka70

192.168.47.71 kafka71

192.168.47.72 kafka72

重新获取消息

[root@kafka71 ~]# /opt/kafka/bin/kafka-console-consumer.sh --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --topic messagetest --from-beginning

Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-server] instead of [zookeeper].

mymy

Yo!

hello

[root@kafka72 ~]# /opt/kafka/bin/kafka-console-consumer.sh --zookeeper 192.168.47.70:2181,192.168.47.71:2181,192.168.47.72:2181 --topic messagetest --from-beginning

Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release. Consider using the new consumer by passing [bootstrap-server] instead of [zookeeper].

hello

mymy

Yo!