

《版本说明》

软件版本-

CentOS-7-x86_64-DVD-2003.iso

jdk-8u241-linux-x64.tar.gz

apache-zookeeper-3.6.1-bin.tar.gz

kafka_2.13-2.6.0.tgz

[软件安装教程](#)

一、搭建zookeeper集群

1.1 搭建集群环境

- 克隆虚拟机

克隆3台虚拟机，分别命名为node001，node002，node003

ip分别为 172.31.2.101，172.31.2.102，172.31.2.103

- 修改克隆虚拟机的静态IP

```
vi /etc/sysconfig/network-scripts/ifcfg-ens33
```

```
TYPE=Ethernet # 网络类型为以太网
BOOTPROTO=static # 手动分配ip
DEVICE=ens33 # 网卡设备名，设备名一定要跟文件名一致 我的是ens33，如果配置不一样，等下网络重启的时候会失败
ONBOOT=yes # 该网卡是否随网络服务启动
IPADDR=172.31.2.102 # 该网卡ip地址 和自己宿主机同一网段
NETMASK=255.255.240.0 # 子网掩码
GATEWAY=172.31.15.254 # 网关 和自己宿主机一样
DNS1=8.8.8.8 # 8.8.8.8为Google提供的免费DNS服务器的IP地址
```

配置网络工作

```
/etc/sysconfig/network文件里增加 NETWORKING=yes #网络是否工作，此处一定不能为no
```

配置公共DNS服务

```
/etc/resolv.conf 文件里增加 nameserver 8.8.8.8
```

关闭防火墙

```
systemctl stop firewalld # 临时关闭防火墙
systemctl disable firewalld # 禁止开机启动
```

重启网络服务,显示OK的话就配置OK了

```
service network restart
```

- 修改主机名

```
vi /etc/hostname # 改为想要设置的主机名
reboot # 重新启动服务器。
```

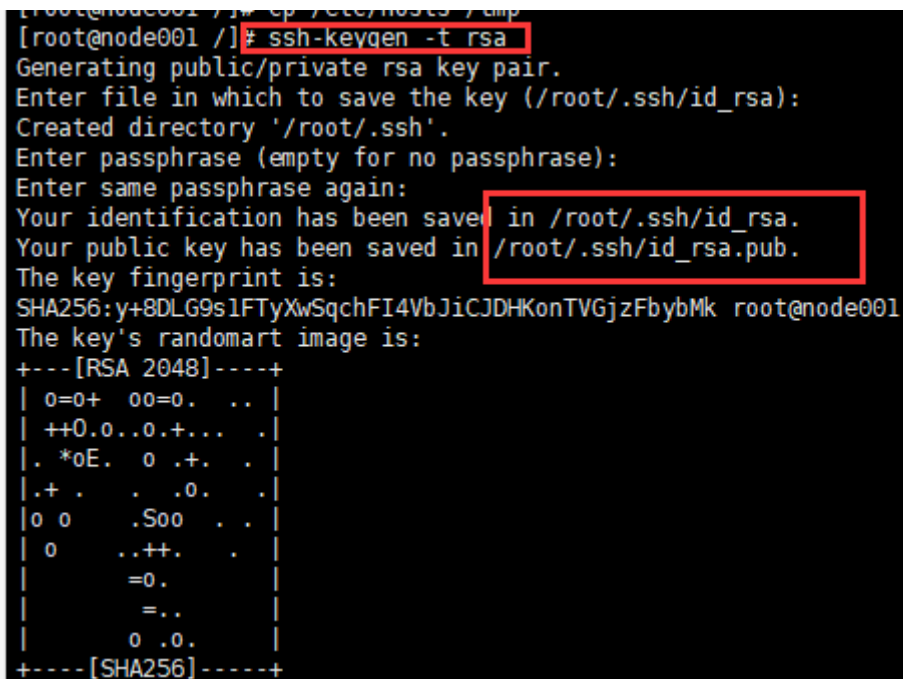
- 配置hosts系统文件

```
vi /etc/hosts
#追加 172.31.2.101 node001 172.31.2.102 node002 172.31.2.103 node003
vi /tmp/add.list #添加 node002 node003
cp /etc/hosts /tmp
#其他主机类似一样配置
```

1.2 SSH免密登录

- 创建密钥

```
ssh-keygen -t rsa
```



```
[root@node001 /]# cp /etc/hosts /tmp
[root@node001 /]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:y+8DLG9s1FTyXwSqchFI4VbJiCJDHKonTVGjzFbybMk root@node001
The key's randomart image is:
+---[RSA 2048]---+
| 0=0+ 00=0.  .. |
| ++0.0..0.+...  |
| . *oE. 0 .+.  . |
| .+ .  . .0.  . |
| 0 0  .Soo  .  . |
| 0  ..++ .  .  |
|      =0.      |
|      =..      |
|      0 .0.    |
|-----[SHA256]-----+
```

- 认证授权

将公钥 (id_rsa.pub) 文件的内容追加到 authorized_keys文件中

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

- 文件赋权

在当前账号下，需要给authorized_keys文件赋予600权限，否者会因为权限限制导致登录失败。

```
chmod 600 ~/.ssh/authorized_keys
```

- 给其他主机创建密钥

```
ssh-keygen -t rsa
```

将authorized_keys文件分发给其他主机相应文件夹下

- 测试

```
ssh node002
```

1.3 搭建zookeeper集群

```
vi zoo.cfg
# 追加
server.1=node001:2888:3888
server.2=node002:2888:3888
server.3=node003:2888:3888
# 修改配置项
dataDir=/usr/local/apache-zookeeper-3.6.1-bin/dataDir
# 添加配置项
dataLogDir=/usr/local/apache-zookeeper-3.6.1-bin/dataLogDir
# 保存退出后，创建目录
mkdir dataDir
mkdir dataLogDir
#其他主机也是一样操作
```

编写zookeeper集群启动脚本

在/usr/local/apache-zookeeper-3.6.1-bin/bin/目录下创建zk.sh

```
#!/bin/bash
case $1 in
"start"){
    for i in node001 node002 node003
    do
        echo "*****$i*****"
        ssh $i "/usr/local/apache-zookeeper-3.6.1-bin/bin/zkServer.sh start"
    done
};;
"stop"){
    for i in node001 node002 node003
    do
        echo "*****$i*****"
        ssh $i "/usr/local/apache-zookeeper-3.6.1-bin/bin/zkServer.sh stop"
    done
};;
"status"){
    for i in node001 node002 node003
    do
        echo "*****$i*****"
        ssh $i "/usr/local/apache-zookeeper-3.6.1-bin/bin/zkServer.sh status"
    done
};;
esac
```

```
#赋权
chmod 777 zk.sh
#启动
zk.sh start
```

编写jps查询集群脚本

在/root/bin/目录下创建xcall.sh

```
#!/bin/bash

for i in node001 node002 node003
do
    echo ----- $i -----
    ssh $i '/usr/local/java/jdk1.8.0_241/bin/jps'
done
```

```
#赋权
chmod 777 xcall.sh
#使用
xcall.sh jps
```

二、搭建kafka集群

2.1 搭建集群

- 修改配置文件server.properties

vi server.properties

三台机子分别修改broker.id=0 broker.id=1 broker.id=2

zookeeper.connect=172.31.2.101:2181,172.31.2.102:2181,172.31.2.103:2181

- 创建集群启动脚本

在/usr/local/kafka_2.13-2.6.0/bin下创建kk.sh

```
#!/bin/bash
case $1 in
"start"){
    for i in node001 node002 node003
    do
        echo "*****$i*****"
        ssh $i "/usr/local/kafka_2.13-2.6.0/bin/kafka-server-
start.sh -daemon /usr/local/kafka_2.13-2.6.0/config/server.properties"
    done
};;

"stop"){
    for i in node001 node002 node003
    do
        echo "*****$i*****"
        ssh $i "/usr/local/kafka_2.13-2.6.0/bin/kafka-server-
stop.sh"
```

```
done
```

```
};;
```

```
esac
```

```
#赋权
```

```
chmod 777 kk.sh
```

```
#启动
```

```
./kk.sh start
```

2.2 kafka操作命令

2.2.1命令行操作topic增删改查

1.查看所有topic命令

```
bin/kafka-topics.sh --list --zookeeper node001:2181
```

2.创建topic（在kafka中创建主题first，设置两个分区，两个副本）（副本数不能大于集群数）

```
bin/kafka-topics.sh --create --zookeeper node001:2181 --topic first --partitions  
2 --replication-factor 2
```

3.删除主题

```
bin/kafka-topics.sh --delete --zookeeper node001:2181 --topic first
```

4.查看主题描述

```
bin/kafka-topics.sh --describe --topic first --zookeeper node001:2181
```

5.修改主题

```
bin/kafka-topics.sh --alter --topic first --zookeeper node001:2181
```

2.2.2命令行测试生产者消费者消费

生产者执行命令

```
bin/kafka-console-producer.sh -topic first --broker-list node001:9092
```

消费者执行命令

```
bin/kafka-console-consumer.sh -topic first --bootstrap-server node002:9092 --  
from-beginning
```

```
1 node001 x +
4515 Kafka
4584 Jps
----- node003 -----
2148 QuorumPeerMain
3492 Jps
3421 Kafka
[root@node001 bin]# cd ..
[root@node001 kafka_2.13-2.6.0]# bin/kafka-console-producer.sh -topic first --broker-list node001:9092
>hhhhh
>oooo
>

1 node002 x +
[root@node002 kafka_2.13-2.6.0]# cd lo
-bash: cd: lo: 没有那个文件或目录
[root@node002 kafka_2.13-2.6.0]# cd logs/
[root@node002 logs]# ls
[root@node002 logs]# ll
总用量 0
[root@node002 logs]# cd ..
[root@node002 kafka_2.13-2.6.0]# bin/kafka-console-consumer.sh -topic first --bootstrap-server node002:9092
hhhhh
oooo

1 node003 x +
[root@node003 kafka_2.13-2.6.0]# bin/kafka-console-consumer.sh -topic first --bootstrap-server node002:9092 --from-beginning
hhhhh
oooo
```

三、springboot集成kafka

Producer有同步发送和异步发送2种策略：

kafka异步发送：

当Kafka返回错误的时候，onCompletion方法会收到一个非null的异常。上面的例子直接打印异常消息，但是如果是生产环境，需要做一些处理错误的操作。

```
producer.send(new ProducerRecord<String, String>(this.topic, value), new
Callback() {
    @Override
    public void onCompletion(RecordMetadata metadata, Exception exception) {
        if(exception != null){
            exception.printStackTrace();
        }
    }
});
```

kafka同步发送：

在send()方法中使用Future对象获取发送消息返回的信息

```
RecordMetadata recordMetadata = producer.send(new ProducerRecord<String,
String>(this.topic, value)).get();
```

