zookeeper安装以及配置HDFS和yarn的高可用

【注意】以下所有操作均在hadoop用户下， A,B,C代表3个ubuntu服务器的节点名称。

本文档默认读者已经学会使用gedit或者vim修改文件内容，所以仅描述了修改的文件以及文件内容，并未说明如何去修改。

1.【A】上传zookeeper安装包

```
hadoop@mashuai01:~/Downloads$ ls
程序                                eclipse-4.7.0-linux.gtk.x86_64.tar.gz
程序.zip                            eclipse-java-2024-03-R-linux-gtk-x86_64.tar.gz
apache-maven-3.6.3-bin.zip          hadoop-3.1.3.tar.gz
apache-zookeeper-3.7.2-bin.tar.gz   jdk-8u181-linux-x64.tar.gz
hadoop@mashuai01:~/Downloads$ sudo tar -zxvf apache-zookeeper-3.7.2-bin.tar.gz -C /usr/l
ocal/
[sudo] hadoop 的密码：
apache-zookeeper-3.7.2-bin/docs/
apache-zookeeper-3.7.2-bin/docs/images/
```

2 【A】在hadoop用户下解压缩zookeeper到指定路径

```
$ sudo tar -zxvf apache-zookeeper-3.7.2-bin.tar.gz -C /usr/local/
$ cd /usr/local/
$ sudo mv apache-zookeeper-3.7.2-bin/   zookeeper
$ sudo chown -R hadoop:hadoop zookeeper/
$ cd zookeeper/conf
$ cp zoo_sample.cfg   zoo.cfg
$ vim zoo.cfg
```

编辑zoo.cfg文件，修改dataDir如下图

```
# the directory where the snapshot is stor
# do not use /tmp for storage, /tmp here
# example sakes.
dataDir=/data/zookeeper
# the port at which the clients will conne
clientPort=2181
# the maximum number of client connections
# increase this if you need to handle mor
#maxClientCnxns=60
```

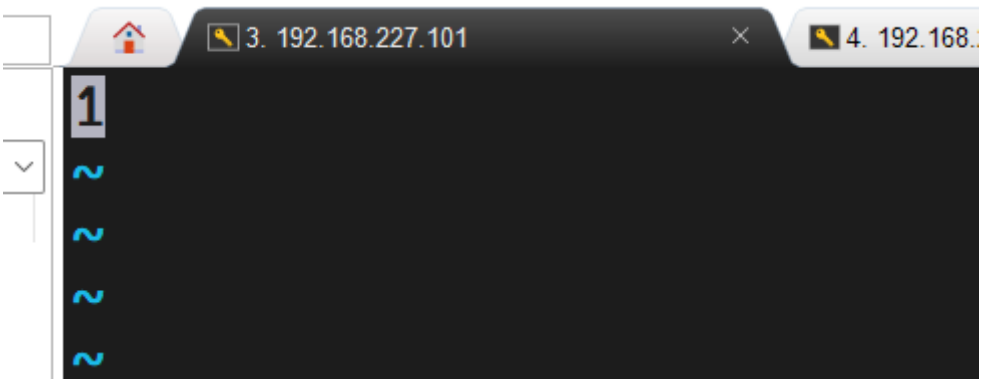
zoo.cfg文件最后追加如下

```
ider
#metricsProvider.httpPort=7000
#metricsProvider.exportJvmInfo=true
server.1=mashuai01:2888:3888
server.2=mashuai02:2888:3888
server.3=mashuai03:2888:3888
```

3.【A】创建zookeeper目录以及myid文件

```
cd  /
sudo mkdir  data
sudo mkdir  data/zookeeper
cd  /data/zookeeper
sudo vim myid
```

编辑myid文件，写入1，如下图。




```
sudo chown -R hadoop:hadoop /data
```

4.【B,C】在02,03节点上，相同位置，创建相同的myid，文件内容分别2,3

```
cd  /
sudo mkdir  data
sudo mkdir  data/zookeeper
cd  /data/zookeeper
sudo vim myid
```

```
6
7 sudo chown -R hadoop:hadoop /data
```

编辑myid文件，B服务器myid写2，C服务器myid写3。

5.【A】复制zookeeper文件到B/C上

```
1 scp -r /usr/local/zookeeper  hadoop@mashuai02:/usr/local/
2 scp -r /usr/local/zookeeper  hadoop@mashuai03:/usr/local/
```

6.【B、C】为zookeeper目录授权

```
1 sudo chown -R hadoop:hadoop /usr/local/zookeeper
```

7.【A,B,C】增加环境变量 在3台机器上

```
1 vim ~/.bashrc
```

添加如下内容

export ZK_HOME=/usr/local/zookeeper

export PATH=$PATH:$ZK_HOME/bin

```
1 source ~/.bashrc
```

8.【A、B、C】启动zookeeper

```
1  zkServer.sh start
```

```
hadoop@mashuai02:/usr/local/zookeeper/bin$ ./zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
```

9. 【A、B、C】查看zookeeper各节点状态

```
zkServer.sh status
```

```
hadoop@mashuai01:/usr/local/zookeeper/bin$ ./zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
```

```
hadoop@mashuai02:/usr/local/zookeeper/bin$ ./zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: leader
```

```
hadoop@mashuai03:/usr/local/zookeeper/bin$ ./zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/local/zookeeper/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
```

会看到，有一个节点的Mode是leader，另外两个是follower。 具体哪个会是leader是zookeeper自己选举出来的。

以下环境部署，基于前面课程修改配置，将原来的hadoop环境修改为HA

10. 【A、B、C】修改hadoop的配置文件

core-site.xml

```
<configuration>
<property>
    <name>fs.defaultFS</name>
    <value>hdfs://mycluster</value>
</property>
<property>
    <name>hadoop.tmp.dir</name>
    <value>file:/usr/local/hadoop/tmp</value>
</property>
<!-- 指定 zkfc 要连接的 zkServer 地址 -->
<property>
```

```
<name>ha.zookeeper.quorum</name>
<value>mashuai01:2181,mashuai02:2181,mashuai03:2181</value>
</property>
</configuration>

```

hdfs-site.xml

```

<configuration>
        <!--<property>
    <name>dfs.namenode.secondary.http-address</name>
    <value>mashuai02:9868</value>
        </property>-->
<property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/usr/local/hadoop/dfs/name</value>
</property>

<property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/usr/local/hadoop/dfs/data</value>
</property>

<property>
    <name>dfs.replication</name>
    <value>3</value>
</property>

<!-- JournalNode 数据存储目录 -->
<property>
<name>dfs.journalnode.edits.dir</name>
<value>/usr/local/hadoop/tmp/jn</value>
</property>
<!-- 完全分布式集群名称 -->
<property>
<name>dfs.nameservices</name>
<value>mycluster</value>
```

```
</property>
<!-- 集群中 NameNode 节点都有哪些 -->
<property>
<name>dfs.ha.namenodes.mycluster</name>
<value>nn1,nn2,nn3</value>
</property>
<!-- NameNode 的 RPC 通信地址 -->
<property>
<name>dfs.namenode.rpc-address.mycluster.nn1</name>
<value>mashuai01:8020</value>
</property>
<property>
<name>dfs.namenode.rpc-address.mycluster.nn2</name>
<value>mashuai02:8020</value>
</property>
<property>
<name>dfs.namenode.rpc-address.mycluster.nn3</name>
<value>mashuai03:8020</value>
</property>
<!-- NameNode 的 http 通信地址 -->
<property>
<name>dfs.namenode.http-address.mycluster.nn1</name>
<value>mashuai01:9870</value>
</property>
<property>
<name>dfs.namenode.http-address.mycluster.nn2</name>
<value>mashuai02:9870</value>
</property>
<property>
<name>dfs.namenode.http-address.mycluster.nn3</name>
<value>mashuai03:9870</value>
</property>
<!-- 指定 NameNode 元数据在 JournalNode 上的存放位置 -->
<property>
<name>dfs.namenode.shared.edits.dir</name>
<value>qjournal://mashuai01:8485;mashuai02:8485;mashuai03:8485/mycluster</value>
</property>
<!-- 访问代理类：client 用于确定哪个 NameNode 为 Active -->
<property>
<name>dfs.client.failover.proxy.provider.mycluster</name>
```

```
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value
>
</property>
<!-- 配置隔离机制，即同一时刻只能有一台服务器对外响应 -->
<property>
<name>dfs.ha.fencing.methods</name>
<value>sshfence</value>
</property>
<!-- 使用隔离机制时需要 ssh 秘钥登录  下面的路径改成自己的-->
<property>
<name>dfs.ha.fencing.ssh.private-key-files</name>
<value>/home/hadoop/.ssh/id_rsa</value>
</property>
<!-- 启用 nn 故障自动转移 -->
<property>
<name>dfs.ha.automatic-failover.enabled</name>
<value>true</value>
</property>
</configuration>

```

yarn-site.xml

```
<configuration>
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
<!--
<property>
    <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>

<property>
    <name>yarn.resourcemanager.address</name>
    <value>mashuai01:8032</value>
</property>
```

```
-->
<!-- 启用 resourcemanager ha -->
<property>
<name>yarn.resourcemanager.ha.enabled</name>
<value>true</value>
</property>
<!-- 声明两台 resourcemanager 的地址 -->
<property>
<name>yarn.resourcemanager.cluster-id</name>
<value>cluster-yarn1</value>
</property>
<!--指定 resourcemanager 的逻辑列表-->
<property>
<name>yarn.resourcemanager.ha.rm-ids</name>
<value>rm1,rm2,rm3</value>
</property>
<!-- ========== rm1 的配置 ========== -->
<!-- 指定 rm1 的主机名 -->
<property>
<name>yarn.resourcemanager.hostname.rm1</name>
<value>mashuai01</value>
</property>
<!-- 指定 rm1 的 web 端地址 -->
<property>
<name>yarn.resourcemanager.webapp.address.rm1</name>
<value>mashuai01:8088</value>
</property>
<!-- ========== rm2 的配置 ========== -->
<!-- 指定 rm2 的主机名 -->
<property>
<name>yarn.resourcemanager.hostname.rm2</name>
<value>mashuai02</value>
</property>
<property>
<name>yarn.resourcemanager.webapp.address.rm2</name>
<value>mashuai02:8088</value>
</property>
<!-- ========== rm3 的配置 ========== -->
<!-- 指定 rm1 的主机名 -->
<property>
```

```
<name>yarn.resourcemanager.hostname.rm3</name>
<value>mashuai03</value>
</property>
<!-- 指定 rm1 的 web 端地址 -->
<property>
<name>yarn.resourcemanager.webapp.address.rm3</name>
<value>mashuai03:8088</value>
</property>
<!-- 指定 rm1 的内部通信地址 -->
<!-- 指定 zookeeper 集群的地址 -->
<property>
<name>yarn.resourcemanager.zk-address</name>
<value>mashuai01:2181,mashuai02:2181,mashuai03:2181</value>
</property>
<!-- 启用自动恢复 -->
<property>
<name>yarn.resourcemanager.recovery.enabled</name>
<value>true</value>
</property>
<!-- 指定 resourcemanager 的状态信息存储在 zookeeper 集群 -->
<property>
<name>yarn.resourcemanager.store.class</name>
<value>org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore</value>
</property>
<!-- 环境变量的继承 -->
<property>
        <name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_
DISTCACHE,HADOOP_YARN_HOME,HADOOP_HOME,PATH,LANG,TZ,HADOOP_MAPRED_HOME</value>
</property>
<property>
        <name>yarn.nodemanager.vmem-check-enabled</name>
        <value>false</value>
</property>
<!-- Site specific YARN configuration properties -->

</configuration>

```

11. 【A】拷贝hadoop的配置文件到B,C

```
scp ../etc/hadoop/*  hadoop@mashuai02:/usr/local/hadoop/etc/hadoop
scp ../etc/hadoop/*  hadoop@mashuai03:/usr/local/hadoop/etc/hadoop
```

12.【A,B,C】在各个JournalNode节点上，输入一下命令启动iournalnode服务

```
 hdfs --daemon start journalnode
```

这时jps可以看到journalnode进程

13.【A】在 nn1 上，对其进行格式化，并启动

```
cd /usr/local/hadoop/
rm -r dfs/ logs/ tmp/
hdfs namenode -format
hdfs --daemon start namenode
```

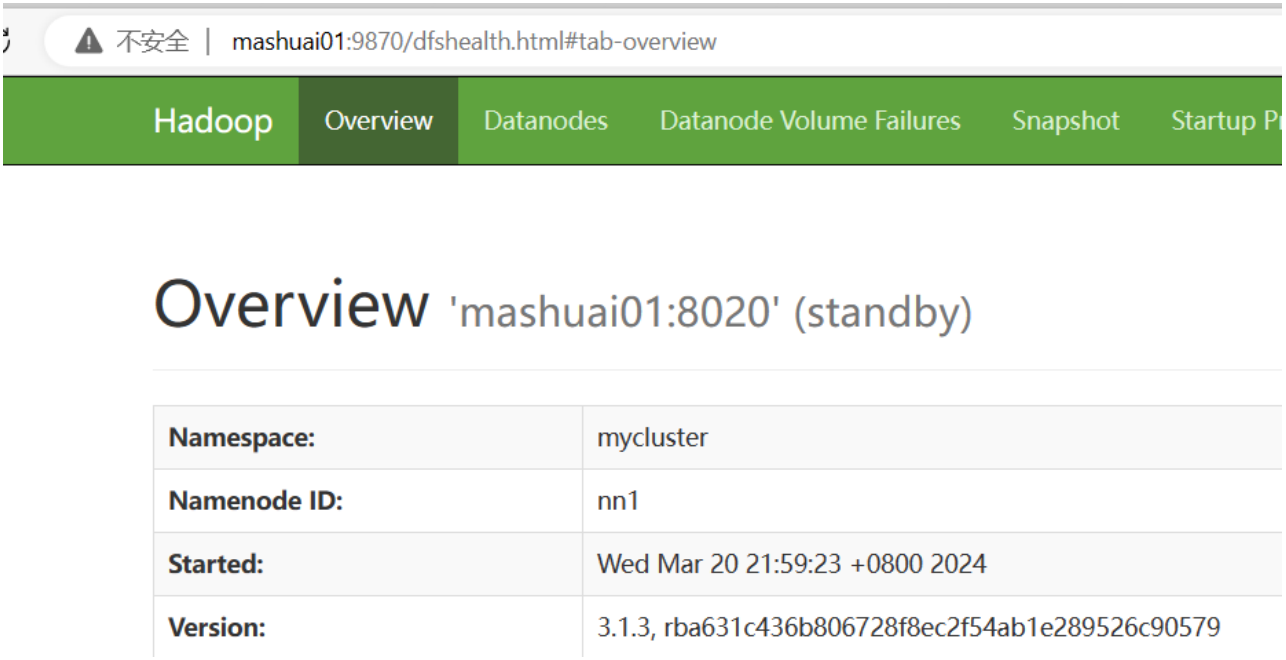
14.【B、C】在 nn2和 nn3 ，同步nn1 的元数据信息

```
cd /usr/local/hadoop/
rm -r dfs/ logs/ tmp/
hdfs namenode -bootstrapStandby
```

15.【B、C】启动 nn2 和 nn3

```
hdfs --daemon start namenode
```

16 【A】查看web界面 9870



17.【A,B,C】在所有节点，启动datanode

```
hdfs --daemon start datanode
```

18.【A】在A节点上格式化zkfc：

```
hdfs zkfc -formatZK
```

19.【A、B、C】在各个NameNode节点上启动DFSZK Failover Controller，
先在哪台机器启动，哪个机器的NameNode就是Active NameNode

```
hdfs --daemon start zkfc
手动切换故障节点，可不执行
hdfs haadmin -failover nn1 nn2
```

jps查看进程，发现zkfc进程正常运行。hdfs zkfc -formatZK，没有格式化整个Zookeeper，只是在Zookeeper中创建了路径/hadoop-ha/hacluster。

20.【A,B,C】 修改*start-dfs.sh*和stop-dfs.sh文件，添加下列参数
HDFS_NAMENODE_USER=hadoop
HDFS_DATANODE_USER=hadoop

HDFS_JOURNALNODE_USER=hadoop
HDFS_ZKFC_USER=hadoop
HADOOP_SECURE_DN_USER=hadoop
HDFS_SECONDARYNAMENODE_USER=hadoop
添加后使用source一下，生效

21.【A,B,C】在start-yarn.sh stop-yarn.sh中添加
YARN_RESOURCEMANAGER_USER=hadoop
YARN_NODEMANAGER_USER=hadoop
添加后使用source一下，生效

22.【A】启动yarn

```
`start-yarn.sh
```

- 查看服务状态

```
yarn rmadmin -getServiceState rm1
```

- 可以去 zkCli.sh 客户端查看 ResourceManager 选举锁节点内容

【补充】关机后启动方法
1.【A、B、C】启动zookeeper，每节点启动

```
zkServer.sh start
```

2.【A】启动hdfs，主节点启动

```
start-dfs.sh
```

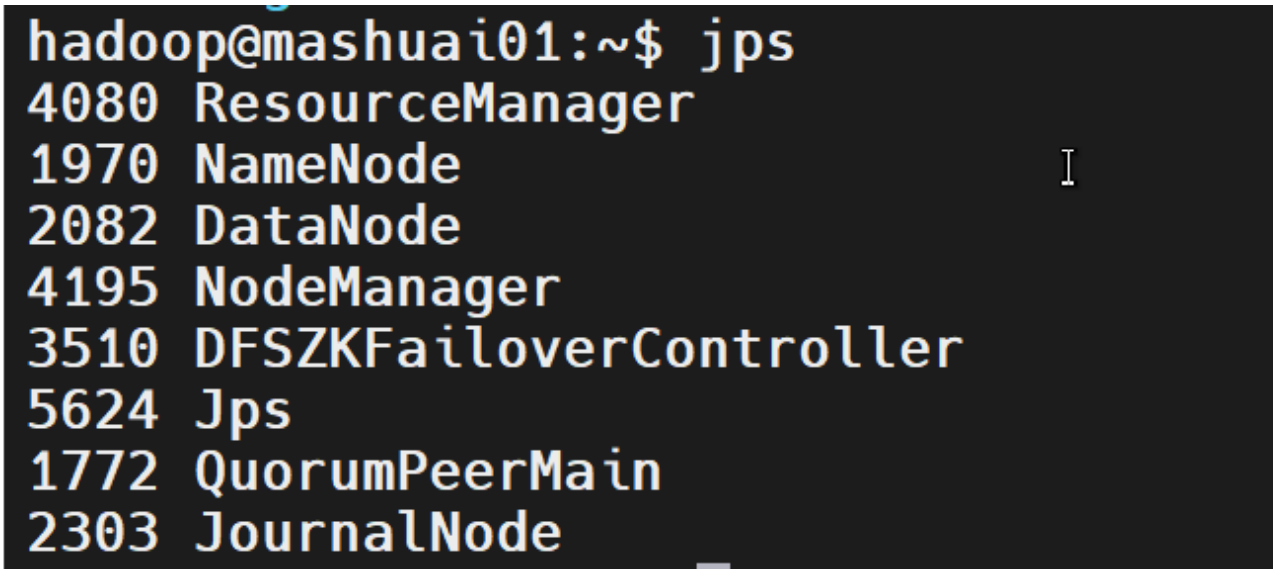
~~3.【A、B、C】启动zkfc，每节点启动~~

```
hdfs --daemon start zkfc
```

4.【A】启动yarn，主节点启动

```
start-yarn.sh
```

验证进程启动是否成功



```
hadoop@mashuai02:~$ jps
25136 DataNode
26035 DFSZKFailoverController
27316 Jps
26437 ResourceManager
25254 JournalNode
26520 NodeManager
19146 QuorumPeerMain
25039 NameNode
```

```
hadoop@mashuai03:~$ jps
19873 DFSZKFailoverController
21618 Jps
13732 QuorumPeerMain
18869 NameNode
18966 DataNode
20152 ResourceManager
20235 NodeManager
19083 JournalNode
```

验证hdfs的namenode的active是否可以自动迁移，使用

hdfs --daemon stop namenode

或者

ps -ef | grep namenode

kill -9 pid

停止namenode，但是发现另外两台机器并没有新的active出现，仍然是两个standby，经过查找资料，通过以下方式 修改hdfs-site.xml可以解决。

hadoop相关问题

序：namenode高可用问题

namenode的高可用是由QJM和zkfc加zk集群来实现的，当宕机再启动的时候，会切换很快，但是如果直接宕机或者是hang机，当ssh无法登录上去的时候，就会导致切换不成功，一直为standby状态。。。需要登录上去，进行探测namenode进程是否存在，如果存在则杀死，这就是sshfence的实现，但是要在宕机的时候，也可以切换，必须修改为如下配置：

```
<property>
        <name>dfs.ha.fencing.methods</name>
        <value>
        sshfence
        shell(/bin/true)
        </value>
</property>
```