# hadoop+hbase 部署和测试

Name：曲中岭

Email:zlingqu@126.com
Q Q :441869115

# 第一章 规划

## 1.1 版本

机器：Vsphere 虚拟机
操作系统：CentOS 6.6 x64，最小化安装
内存：4G
CPU：2vCPU
磁盘：/data—> 300GB
Jdk：1.8.0_171 x64
Hadoop：2.8.4
Hbase：2.0.0
Zookeeper：3.4.12

## 1.2模块分布

| 模块 | 节点 | 服务器 | | | | |
|---|---|---|---|---|---|---|
| | | hadoop01 | hadoop02 | hadoop03 | hadoop04 | hadoop05 |
| | | 10.1.5.201 | 10.1.5.202 | 10.1.5.203 | 10.1.5.204 | 10.1.5.205 |
| HDFS | NameNode | √ | √ | | | |
| | DFSZKFailoverController | √ | √ | | | |
| | JournalNode | | | √ | √ | √ |
| | DataNode | √ | √ | √ | √ | √ |
| Yarn | ResourceManager | √ | | | | |
| | NodeManager | √ | √ | √ | √ | √ |
| ZooKeeper | QuorumPeerMain | | | √ | √ | √ |
| HBase | HMaster | √ | √ | | | |
| | HRegionServer | √ | √ | √ | √ | √ |

# 第二章：环境准备

## 2.1 用户/目录/hostname 等

### 2.1.1 修改 hostname

按照规划，分别修改 hostname 为 hadoop01-05
并修改/etc/sysconfig/network 中内容，保证重启后仍然有效

### 2.1.2 修改 hosts 文件

hadoop01-05 上使用 root 操作:

```
 echo ″
10.1.5.201 hadoop01
10.1.5.202 hadoop02
10.1.5.203 hadoop03
10.1.5.204 hadoop04
10.1.5.205 hadoop05″ >> /etc/hosts
```

### 2.1.3 创建用户

hadoop01-05 上使用 root 操作:

```
    useradd hadoop
    echo ″hadoop″ | passwd --stdin hadoop
```

### 2.1.3 创建目录

hadoop01-05 上使用 root 操作

```
      mkdir /data
      chown hadoop.hadoop /data
```

### 2.1.4 修改环境变量

hadoop01-05 上使用 root 操作

保证/etc/profice 中有如下内容:

```
export JAVA_HOME=/usr/local/jdk
export HADOOP_HOME=/data/hadoop-2.8.4
```

export HBASE_HOME=/data/hbase-2.0.0
export PATH=$HADOOP_HOME/bin: $HADOOP_HOME/sbin:$HBASE_HOME/bin:$JAVA_HOME/bin:$PATH

## 2.2 主机信任

hadoop01 和 hadoop02 上使用 hadoop 操作，注意在 01 和 02 上都要操作。
ssh-keygen 生成密钥对，这里使用默认选项，一直下一步即可。
ssh-copy-id hadoop@hadoop01 #公钥分发
ssh-copy-id hadoop@hadoop02
ssh-copy-id hadoop@hadoop03
ssh-copy-id hadoop@hadoop04
ssh-copy-id hadoop@hadoop05
如果找不到 ssh-copy-id 命令，需要按照以下包
yum –y install openssh-clients

以上操作保证 01 和 02 对 01-05 均是信任的，hadoop 依赖于 rpc，必须要主机信任。

## 2.3 jdk 环境

hadoop01-05 上使用 root 操作，或者 01 操作后，scp 到其他机器
下载地址：http://www.oracle.com/technetwork/cn/java/archive-139210-zhs.html
tar xf jdk-8u171-linux-x64.tar –C /usr/local
mv /usr/local/ jdk1.8.0_171 /usr/local/jdk #也可以创建软连接，看个人习惯
保证/etc/profile 中有以下内容：
export JAVA_HOME=/usr/local/jdk
export PATH=$JAVA_HOME/bin:$PATH #$JAVA_HOME 一定要放到$PATH 前面，防止识别到系统环境中其他版本的 jdk

## 2.4 准备 rsync

这一步，只是为了同步的方便，可以使用 scp 等实现同样操作，看个人习惯。
hadoop01 上使用 root 操作：
提供配置文件：
vim /etc/rsyncd.conf

```
port = 873
uid = root
gid = root
use chroot = yes
max connections = 10
host allow=10.1.5.*
pid file = /var/run/rsyncd.pid
lock file = /var/run/rsync.lock
```

```
log file = /var/log/rsyncd.log
[hadoop]
path = /data/hadoop-2.8.4
ignore errors
read only = yes
list=yes
[hbase]
path = /data/hbase-2.0.0
ignore errors
read only = yes
service xinetd start
chkconfig xinetd on
```

启用 rsyncd:

```
sed -i 's/yes/no/g' /etc/xinetd.d/rsync
service xinetd start
chkconfig xinetd on
```

haoop02-05 上使用 hadoop 操作，保证有如下脚本:

```
[hadoop@hadoop02 ~]$ cat hbase.sh
/usr/bin/rsync -avzP --progress --delete 10.1.5.201::hbase /data/hbase-2.0.0
 [hadoop@hadoop02 ~]$ cat hadoop.sh
/usr/bin/rsync -avzP --progress --delete 10.1.5.201::hadoop /data/hadoop-2.8.4
```

hadoop01 上使用 hadoop 操作，保证有如下脚本:

```
[hadoop@hadoop01 ~]$ cat hbase.sh
#!/bin/bash
ssh hadoop@hadoop02 sh hbase.sh
ssh hadoop@hadoop03 sh hbase.sh
ssh hadoop@hadoop04 sh hbase.sh
ssh hadoop@hadoop05 sh hbase.sh
[hadoop@hadoop01 ~]$ cat hadoop.sh
#!/bin/bash
ssh hadoop@hadoop02 sh hadoop.sh
ssh hadoop@hadoop03 sh hadoop.sh
ssh hadoop@hadoop04 sh hadoop.sh
ssh hadoop@hadoop05 sh hadoop.sh
```

## 2.5 hadoop 下载

在 hadoop01 上使用 hadoop 操作:

下载地址:

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

下载后解压到/data/hadoop-2.8.4

```
tar xvzf hadoop-2.8.4.tar.gz -C /data
```

## 2.6 hbase 下载

在 hadoop01 上使用 hadoop 操作:
下载地址：http://archive.apache.org/dist/hbase/
注意 hbase 和 hadoop 版本要匹配，到以下链接查看匹配情况
http://hbase.apache.org/book.html#configuration
```
tar xf hbase-2.0.0-bin.tar.gz -C /data
```

## 2.7 zookeeper 集群安装和下载

部署到 hadoop03-05，比较简单，会的同学，可直接跳过，或者参考 6.1 节

# 第三章 hadoop 安装

在 hadoop01 上使用 hadoop 操作:

## 3.1 修改配置文件

### 3.1.1 hadoop-env.sh

cd /data/hadoop-2.8.4/etc/hadoop
vim hadoop-env.sh
#export JAVA_HOME=${JAVA_HOME} #注释掉这一行，格式不对
export JAVA_HOME=/usr/local/jdk 添加这一样，必须使用绝对路径

### 3.1.2 core-site.xml

```
cat core-site.xml #hadoop 核心配置
<configuration>
    <property>
        <name>fs.defaultFS</name> <!--NameNode 的 URI-->
        <value>hdfs://myha</value>
  </property>
  <property>
        <name>hadoop.tmp.dir</name> <!--hadoop 临时文件的存放目录-->
        <value>/data/hadoop-temp</value>
  </property>
  <property>
      <name>ha.zookeeper.quorum</name>
      <value>hadoop03:2181,hadoop04:2181,hadoop05:2181</value>
  </property>
</configuration>
```

### 3.1.3 hdfs-site.xml

```
cat hdfs-site.xml #hdfs 配置信息
<configuration>
    <!--指定数据副本数量-->
    <property>
      <name>dfs.replication</name>
      <value>3</value>
    </property>
```

```
    <!--指定 hdfs 的 nameserver 元数据路径-->
    <property>
      <name>dfs.namenode.name.dir</name>
      <value>file:/data/name-ha</value>
    </property>
    <!--指定 hdfs 的 datanode 的数据路径 -->
    <property>
      <name>dfs.datanode.data.dir</name>
      <value>file:/data/data-ha</value>
    </property>
    <!--指定 hdfs 的 nameservice 为 myha，需要和 core-site.xml 中的保持一致 -->
    <property>
      <name>dfs.nameservices</name>
      <value>myha</value>
    </property>
 <!-- 指定 nameservice 下面有哪些 namenode，hadoop3 支持两个以上的 namenode -
->
 <!--注意 myha,nn1,nn2 要和前后文、core.site.xml 中的配置对应 -->
    <property>
      <name>dfs.ha.namenodes.myha</name>
      <value>nn1,nn2</value>
    </property>

<!-- nn1、nn2 的 RPC 通信地址,注意 myha、nn1、nn2、hadoop01、haoop02 等 -->
    <property>
      <name>dfs.namenode.rpc-address.myha.nn1</name>
      <value>hadoop01:9000</value>
    </property>
    <property>
      <name>dfs.namenode.rpc-address.myha.nn2</name>
      <value>hadoop02:9000</value>
    </property>
<!-- nn1、nn2 的 http 地址,注意 myha、nn1、nn2、hadoop01、haoop02 等 -->

    <property>
      <name>dfs.namenode.http-address.myha.nn1</name>
      <value>hadoop01:50070</value>
    </property>
    <property>
      <name>dfs.namenode.http-address.myha.nn2</name>
      <value>hadoop02:50070</value>
    </property>
     <!-- 指定一组 journalNode 的 URI 地址,主 NN 将 edit log 写入，从 NN 读取并载入内
存,配置的节点将开启 journalnode 进程,注意 myha -->
```

```
    <property>
      <name>dfs.namenode.shared.edits.dir</name>
      <value>qjournal://hadoop03:8485;hadoop04:8485;hadoop05:8485/myha</value>
    </property>
    <!-- JournalNode 所在节点上的一个目录，用于存放 editlog 和其他状态信息-->
    <property>
      <name>dfs.journalnode.edits.dir</name>
      <value>/data/journaldata-ha</value>
    </property>
   <!-- 开启 NameNode 失败自动切换 -->
    <property>
      <name>dfs.ha.automatic-failover.enabled</name>
      <value>true</value>
    </property>
   <!-- 客户端与主 NN 进行交互的 Java 实现类,客户端通过该类查找谁是 active NN，除非你自定义了一个，否则都用这个，注意 myha -->
    <property>
      <name>dfs.client.failover.proxy.provider.myha</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
    </property>
    <!-- 配置隔离机制方法，多个机制用换行分割，即每个机制暂用一行,-->
    <property>
       <name>dfs.ha.fencing.methods</name>
     <value>
          sshfence
     </value>
   </property>
   <!-- 使用 sshfence 隔离机制时需要 ssh 免登陆 -->
    <property>
       <name>dfs.ha.fencing.ssh.private-key-files</name>
      <value>/home/hadoop/.ssh/id_rsa</value>
    </property>
    <!-- 配置 sshfence 隔离机制超时时间 -->
    <property>
      <name>dfs.ha.fencing.ssh.connect-timeout</name>
      <value>30000</value>
    </property>
</configuration>
```

### 3.1.4 mapred-site.xml

```
cat mapred-site.xml
<configuration>
```

```
<property>
    <!--告诉 hadoop 以后 MR 运行在 YARN 上-->
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
</configuration>
```

### 3.1.5 yarn-site.xml

```
cat yarn-site.xml
<configuration>
<property>
    <!--nomenodeManager 获取数据的方式是 shuffle-->
    <name>yarn.nodemanager.aux-services </name>
    <value>mapreduce_shuffle</value>
</property>
<property>
    <name>yarn.resourcemanager.webapp.address</name>
    <value>hadoop01:8088</value>
</property>
</configuration>
```

## 3.2 同步配置到 02-05

在 hadoop01 上使用 hadoop 操作
sh hadoop.sh #已经配置好 rsync，也可以使用 scp 等，看个人习惯

## 3.3 格式化文件系统

```
hdfs namenode -format
```

看到如下输出证明格式化成功
18/03/13 13:54:36 INFO common.Storage: Storage directory ****** has been successfully formatted.

## 3.4 初始化 zkfc

此步骤其实就是向 zookeeper 注册/hadoop-ha/myha

```
hdfs zkfc -formatZK
```

如果看到 Successfully created /hadoop-ha/myha in ZK.
证明注册成功

## 3.5 启动 JournalNode

在 hadoop03-05 上启动

```
hadoop-daemon.sh start journalnode
```

## 3.6 hadoop01 上启动 namenode

一定要启动后才可以执行下一步

```
hadoop-daemon.sh start namenode
```

## 3.7 hadoop02 同步元数据：

```
hdfs namenode -bootstrapStandby
```

看到如下内容，证明同步成功

Storage directory ***** has been successfully formatted.

## 3.8 启动 namenode2

在 hadoop02 上执行

```
hadoop-daemon.sh start namenode
```

## 3.9 启动所有的 datanode

可在各个 datanode 节点上一个一个启动

```
hadoop-daemon.sh start datanode
```

也可以在任一 namenode（对所有节点已做免认证）上启动：

start/stop-dfs.sh

这个操作就相当于同时启动/停止 namenode、dadanode、journalnode、zkfc，如果已经启动对应模块，会跳过

但第一次启动，要按照 3.3-3.8 的顺序来

启动 zkfc，相当于在 zookeeper 中创建了：

```
[zk: localhost:2181(CONNECTED) 25] ls /hadoop-ha/myha
[ActiveBreadCrumb, ActiveStandbyElectorLock]
```

而初始化 zkfc 相当于创建了

```
[zk: localhost:2181(CONNECTED) 25] ls /hadoop-ha/myha
```

## 3.10 启动 yarn

```
start-yarn.sh
```

## 3.11 启停方式汇总

全启动：start-all.sh = start-dfs.sh + start-yarn.sh
模块启动：

start-dfs.sh/stop-dfs.sh = namenode + datanode

start-yarn.sh/stop-yarn.sh = resourcemanager + nodemanager

单个进程启动：(以下，start 可以换为 stop)

hadoop-daemon.sh start namenode

hadoop-daemon.sh start datanode

hadoop-daemon.sh start zkfc

yarn-daemon.sh start nodemanager

yarn-daemon.sh start resourcemanager

hadoop-daemon.sh start journalnode

更多模块单独启停操作，可参考 hadoop-daemon.sh 脚本内容

# 第四章 hbase 安装

## 4.1 部署 hadoop 环境

hbase 依赖于 hdfs，确保已经部署好 hadoop 环境。hbase 集群依赖于 zk 集群，确保已经部署好 zk 集群（见 6.1）

## 4.2 修改配置文件

### 4.2.1 core-site.xml、hdfs-site.xml

cd /data/hbase-2.0.0/conf
这两个配置文件应该和 hdoop 中完全相同，实际中我创建了一个软连接

```
ln -sv /data/hadoop-2.8.4/etc/hadoop/core-site.xml core-site.xml
ln -sv /data/hadoop-2.8.4/etc/hadoop/hdfs-site.xml hdfs-site.xml
```

### 4.2.2 hbase-env.sh

保证有如下两行
export JAVA_HOME=/usr/local/jdk/ #指定 java_home
export HBASE_MANAGES_ZK=false #不使用 hbase 内置的 zk，如果使用内置的 zk，只会在 hmaster 上启动一个 zk，非集群模式

### 4.2.3 hbase-site.xml

```
cat hbase-site.xml
<configuration>
<property>
    <name>fs.default.name</name>
    <value>hdfs://myha/</value> #同 core-site.xml 相同，指定 hdfs 的 namenode
</property>
<property>
    <name>hbase.rootdir</name>
    <value>hdfs://myha/hbase</value> #配置 hbase 的存储路径
</property>
<property>
    <name>hbase.ZooKeeper.quorum</name> #指定 zk 中注册的名字
    <value>hbase</value>
</property>
```

```
<property>
    <name>hbase.cluster.distributed</name> #表示使用 hbase 的集群模式
    <value>true</value>
</property>
<property>
    <name>hbase.tmp.dir</name> #hbase 的临时目录，不用提前创建
    <value>/data/hbase-temp/</value>
</property>
<property>
    <name>hbase.zookeeper.quorum</name> #zk 集群路径
    <value>hadoop03:2181,hadoop04:2181,hadoop05:2181</value>
</property>
</configuration>
```

### 4.2.4 regionservers

```
cat regionservers
hadoop01
hadoop02
hadoop03
hadoop04
hadoop05
```

此配置表示 HRegionServer 节点

### 4.2.5 同步

hadoop01 上使用 hadoop 用户操作：

```
sh ~/hbase.sh   #见 2.4，已经配置好 rsync 脚本。也可使用 scp 等，看个人习惯。
```

## 4.3 启动 hbase 集群

在 hadoop01 上使用 hadoop 操作，在 01 上操作，则 01 就是 hmaster 节点：

```
sh /data/hbase-2.0.0/bin/start-hbase.sh
```

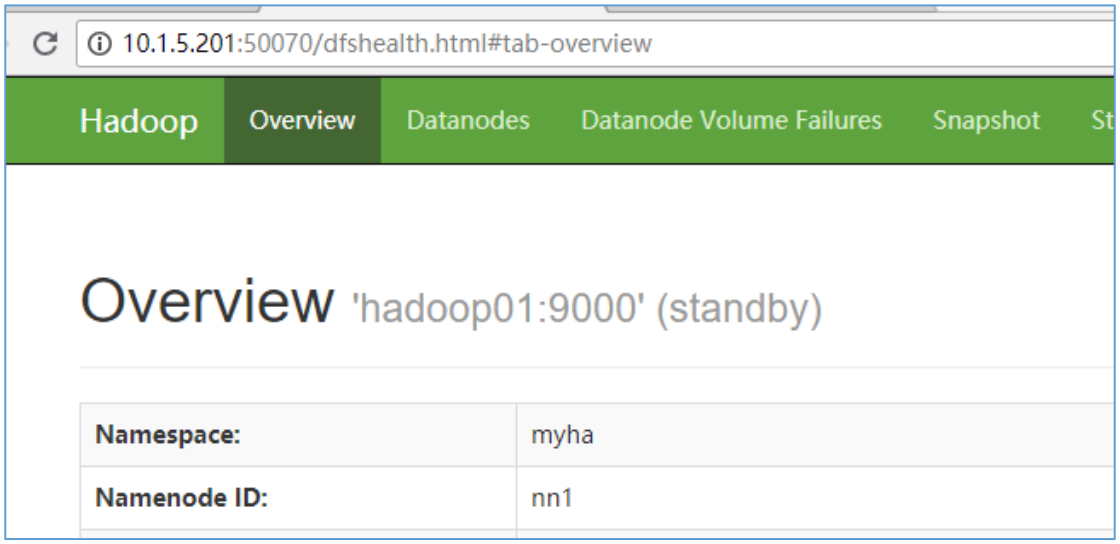
hadoop02 上启动备份 hmaster

```
sh /data/hbase-2.0.0/bin/hbase-daemon.sh start master
```

# 第五章 验证和测试

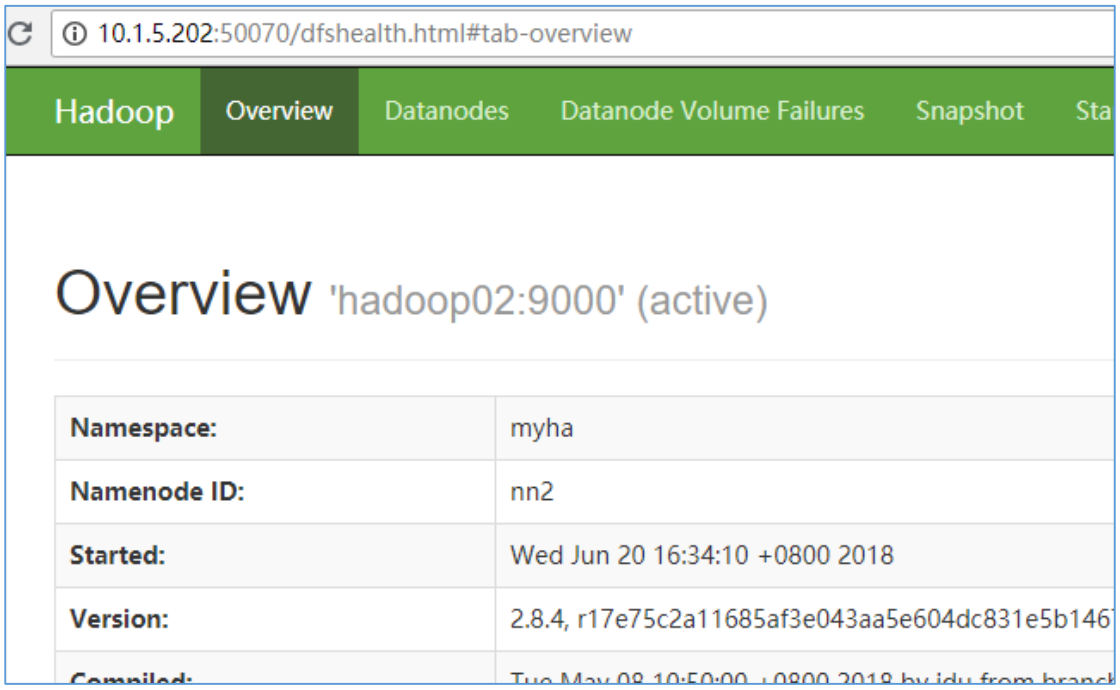
## 5.1 namenode web

http://10.1.5.201:50070/dfshealth.html#tab-overview




standyby 模式

http://10.1.5.202:50070/dfshealth.html#tab-overview




active 模式

In operation

Show 25 entries

| Node | Http Address | Last contact | Capacity |
|---|---|---|---|
| hadoop01:50010 (10.1.5.201:50010) | http://hadoop01:50075 | 0s | 245.18 GB |
| hadoop02:50010 (10.1.5.202:50010) | http://hadoop02:50075 | 2s | 245.18 GB |
| hadoop03:50010 (10.1.5.203:50010) | http://hadoop03:50075 | 2s | 245.18 GB |
| hadoop04:50010 (10.1.5.204:50010) | http://hadoop04:50075 | 0s | 245.18 GB |
| hadoop05:50010 (10.1.5.205:50010) | http://hadoop05:50075 | 2s | 245.18 GB |

Showing 1 to 5 of 5 entries

5 个 datanode 节点

也可以通过命令行查看状态，也得到同样的效果

hdfs haadmin -getServiceState nn1

atandby

hdfs haadmin -getServiceState nn2

standby

```
[hadoop@hadoop01 conf]$
[hadoop@hadoop01 conf]$ hdfs haadmin -getServiceState nn1
standby
[hadoop@hadoop01 conf]$ hdfs haadmin -getServiceState nn2
active
[hadoop@hadoop01 conf]$
```

使用如下命令可以强制转换为 active 状态：

hdfs haadmin -transitionToActive nn1 --forcemanual

另外也可以使用如下命令强制转换为 standby 状态：

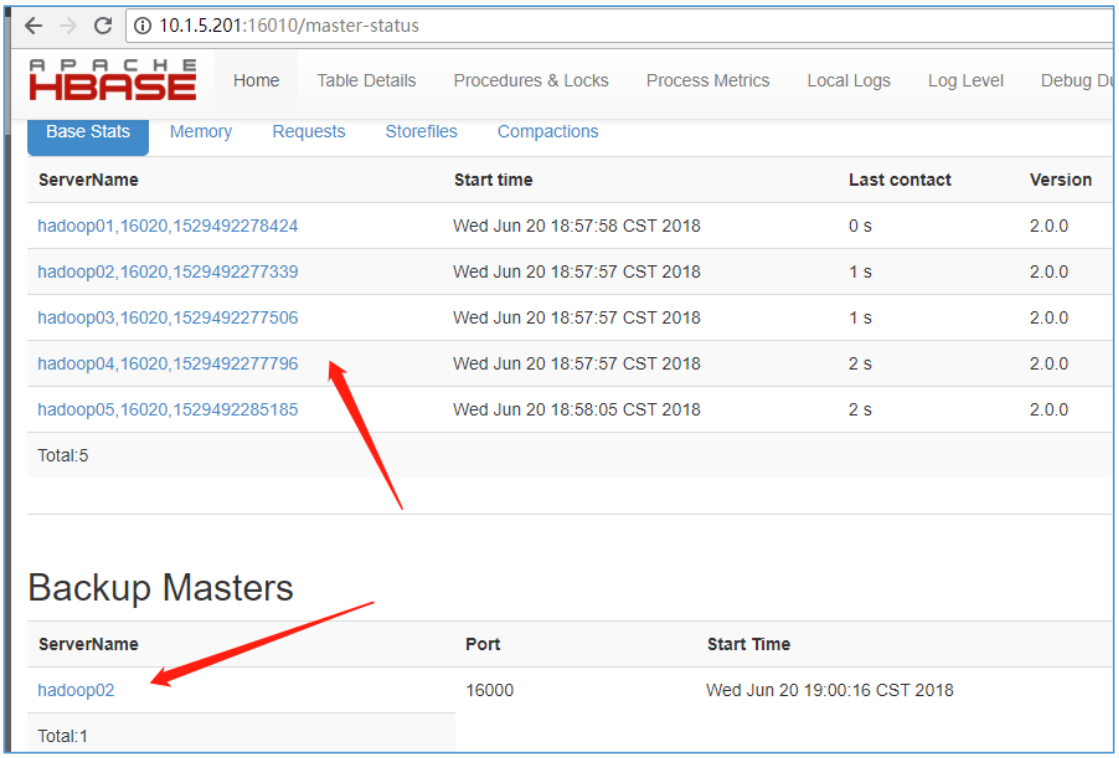
hdfs haadmin -transitionToStandby -nn1 --forcemanual

更多命令，可通过 hdfs haadmin --help 查看得到

## 5.2 hbase web

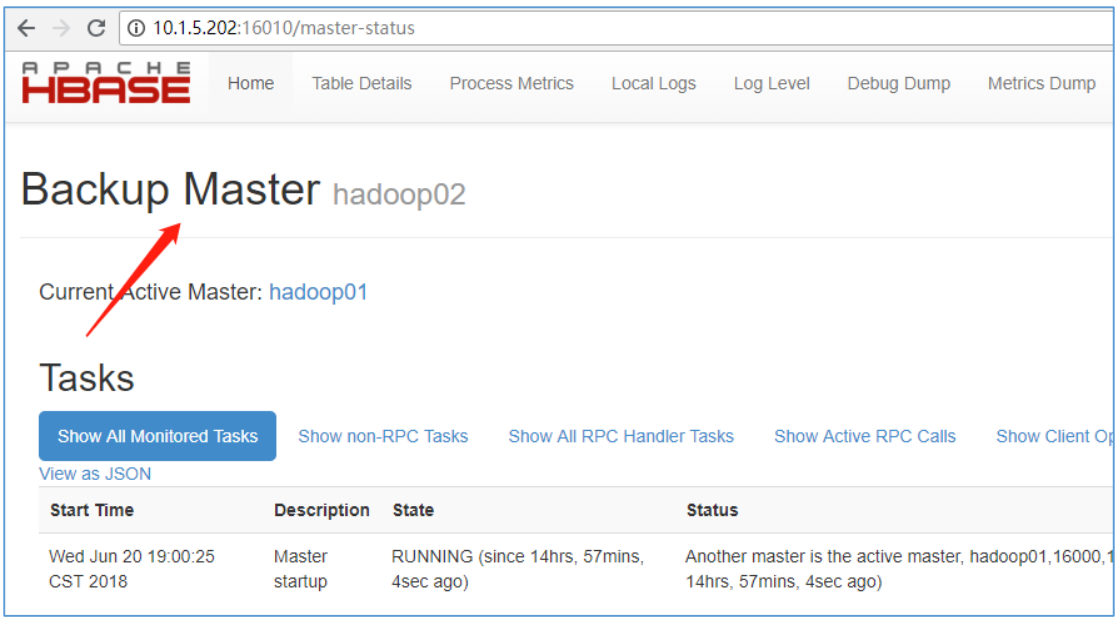
http://10.1.5.201:16010/master-status

看到 5 个从节点，1 个备份 hmater 节点




登陆 hadoop02，也可以看到处于 backup 状态。

http://10.1.5.202:16010/master-status




提示：备份的 hmaster 节点可以多于 1 个，主备的切换，由 zookeeper 控制。

## 5.3 进程确认

**hadooop01 上，7 个进程，和规划一致，见 1.2**

[hadoop@hadoop01 conf]$ jps
28832 DFSZKFailoverController
1137 ResourceManager
1331 NodeManager
612 HRegionServer
28516 DataNode
471 HMaster
28397 NameNode
1663 Jps

```
[hadoop@hadoop01 ~]$ jps
28832 DFSZKFailoverController
1137 ResourceManager
1331 NodeManager
612 HRegionServer
28516 DataNode
15526 Jps
471 HMaster
28397 NameNode
[hadoop@hadoop01 ~]$
[hadoop@hadoop01 ~]$
[hadoop@hadoop01 ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast s
    link/ether 00:0c:29:d1:29:35 brd ff:ff:ff:ff:ff:ff
    inet 10.1.5.201/24 brd 10.1.5.255 scope global eth0
    inet6 fe80::20c:29ff:fed1:2935/64 scope link
```

**hadoop02 上，6 个进程，和规划一致，见 1.2**

[hadoop@hadoop02 ~]$ jps
21539 DataNode
21460 NameNode
22964 HRegionServer
21685 DFSZKFailoverController
23431 Jps
23180 NodeManager
23357 HMaster

```
[hadoop@hadoop02 ~]$
[hadoop@hadoop02 ~]$ jps
21539 DataNode
21460 NameNode
22964 HRegionServer
21685 DFSZKFailoverController
28937 Jps
23357 HMaster
[hadoop@hadoop02 ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOW
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fa
    link/ether 00:0c:29:cd:19:5a brd ff:ff:ff:ff:ff:ff
    inet 10.1.5.202/24 brd 10.1.5.255 scope global eth0
    inet6 fe80::20c:29ff:fecd:195a/64 scope link
       valid_lft forever preferred_lft forever
```

注意：当没有作业进行时，从节点的 NodeManager 将停止，这是正常的。

**hadoop03-hadoop05 上，5 个进程，和规划一致，见 1.2**

[hadoop@hadoop03 ~]$ jps
11795 QuorumPeerMain
20389 HRegionServer
20742 Jps
19478 DataNode
20603 NodeManager
19583 JournalNode

## 5.4 zookeeper 确认

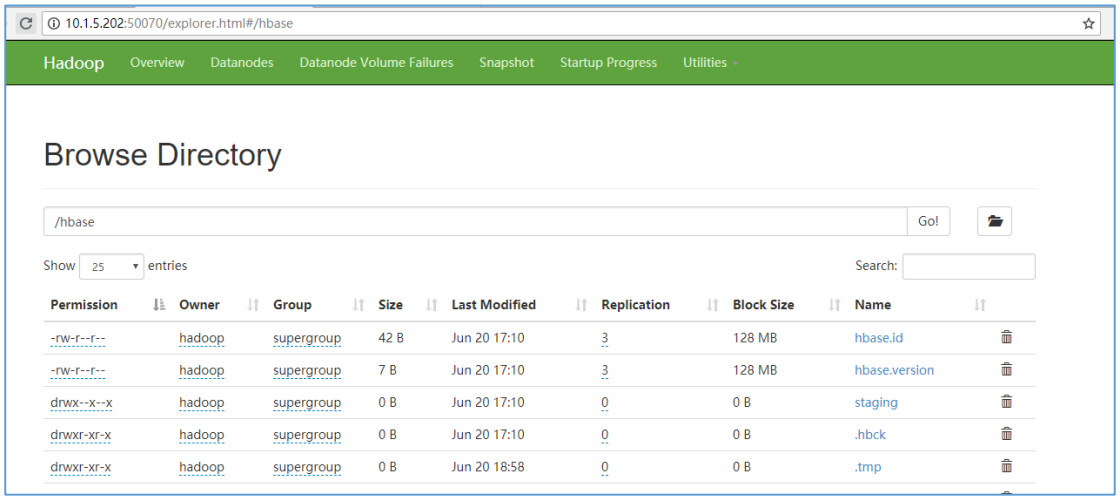
hadoop05 上操作，可以看到 hbase、hadoop-ha/myha 的注册信息。

```
[hadoop@hadoop05 bin]$
[hadoop@hadoop05 bin]$ ./zkcli.sh
Connecting to localhost:2181
2018-06-21 10:40:40,716 [myid:] - INFO  [main
```

```
[zk: localhost:2181(CONNECTED) 0] ls /
[zookeeper, hbase, hadoop-ha]
[zk: localhost:2181(CONNECTED) 1] ls /hbase
[replication, meta-region-server, rs, splitWAL, backup-masters, table-lock, flush-
amespace, hbaseid, table]
[zk: localhost:2181(CONNECTED) 2] ls /hadoop-ha
[myha]
[zk: localhost:2181(CONNECTED) 3] ls /hadoop-ha/myha
[ActiveBreadCrumb, ActiveStandbyElectorLock]
[zk: localhost:2181(CONNECTED) 4]
```

## 5.5 namenode 主备切换

切换前，hadoop01 备，hadoop02 主。查看 hdfs 文件：

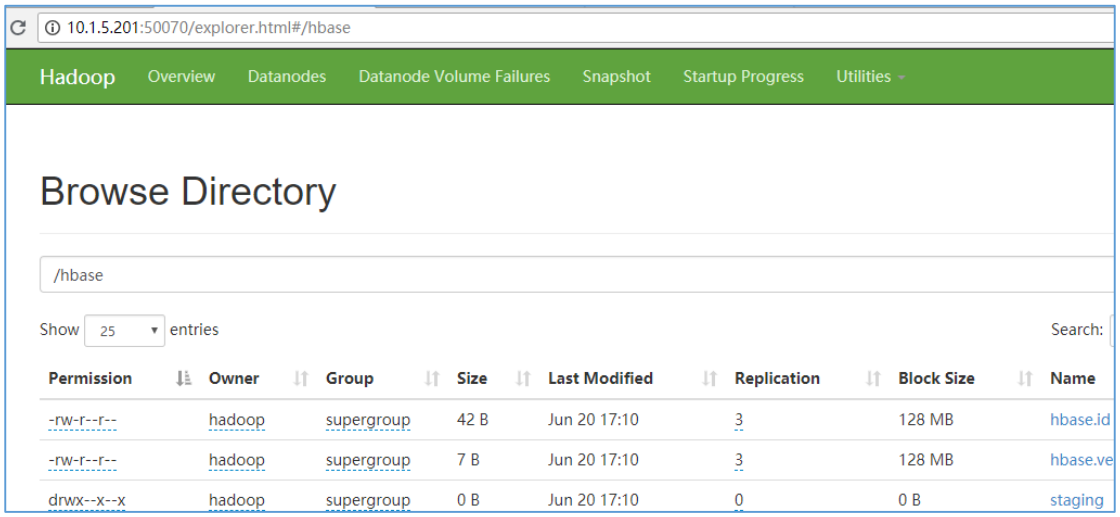



停止 hadoop02 上的 namenode

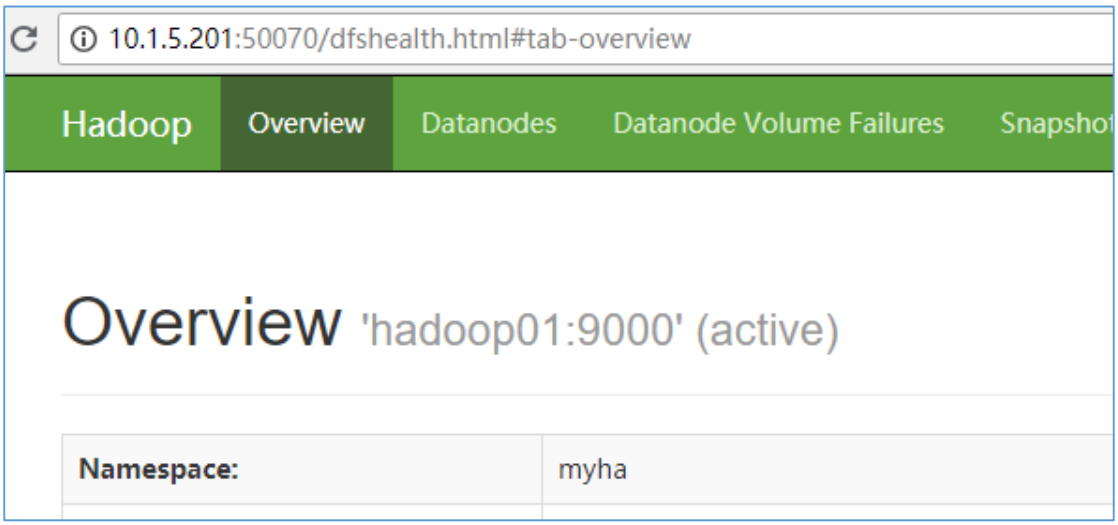
```
[hadoop@hadoop02 ~]$
[hadoop@hadoop02 ~]$ hadoop-daemon.sh stop namenode
stopping namenode
[hadoop@hadoop02 ~]$ jps
29074 Jps
21539 DataNode
22964 HRegionServer
21685 DFSZKFailoverController
23357 HMaster
[hadoop@hadoop02 ~]$
```

hadoop02 无法访问

通过 hadoop01 查看，文件都正常：




hadoop01 成了 active 状态




同时查看 zkfc 的日志 hadoop-hadoop-zkfc-hadoop01.log 有如下内容，也打印出了 hadoop01 成了主节点。

```
2018-06-20 16:34:35,291 INFO org.apache.hadoop.ha.ZKFailoverController: ZK Election indicated that NameNode at hadoop01/10.1.5.201:9000 should become standby
2018-06-20 16:34:35,307 INFO org.apache.hadoop.ha.ZKFailoverController: Successfully transitioned NameNode at hadoop01/10.1.5.201:9000 to standby state
2018-06-21 10:16:00,898 INFO org.apache.hadoop.ha.ActiveStandbyElector: Checking for any old active which needs to be fenced...
2018-06-21 10:16:00,909 INFO org.apache.hadoop.ha.ActiveStandbyElector: Old node exists: 0a046d7968611203666321a08686164f6f70303220a84628d33e
2018-06-21 10:16:00,912 INFO org.apache.hadoop.ha.ZKFailoverController: Should fence: NameNode at hadoop02/10.1.5.202:9000
2018-06-21 10:16:01,917 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: hadoop02/10.1.5.202:9000. Already tried 0 time(s); retry policy is RetryUpToMaxi
mumCountWithFixedSleep(maxRetries=1, sleepTime=1000 MILLISECONDS)
2018-06-21 10:16:01,919 WARN org.apache.hadoop.ipc.Client: Failed to connect to server: hadoop02/10.1.5.202:9000: retries get failed due to exceeded maximum allowed re
tries number: 1
java.net.ConnectException: Connection refused
        at sun.nio.ch.SocketChannelImpl.checkConnect(Native Method)
        at sun.nio.ch.SocketChannelImpl.finishConnect(SocketChannelImpl.java:717)
        at org.apache.hadoop.net.SocketIoWithTimeout.connect(SocketIoWithTimeout.java:206)
        at org.apache.hadoop.net.NetUtils.connect(NetUtils.java:531)
        at org.apache.hadoop.ipc.Client$Connection.setupConnection(Client.java:685)
        at org.apache.hadoop.ipc.Client$Connection.setupIoStreams(Client.java:788)
        at org.apache.hadoop.ipc.Client$Connection.access$3500(Client.java:410)
        at org.apache.hadoop.ipc.Client.getConnection(Client.java:1550)
        at org.apache.hadoop.ipc.Client.call(Client.java:1381)
```

```
2018-06-21 10:16:02,773 INFO org.apache.hadoop.ha.SshFenceByTcpPort: Connected to hadoop02
2018-06-21 10:16:02,773 INFO org.apache.hadoop.ha.SshFenceByTcpPort: Looking for process running on port 9000
2018-06-21 10:16:03,317 INFO org.apache.hadoop.ha.SshFenceByTcpPort: Indeterminate response from trying to kill service. Verifying whether it is running us
2018-06-21 10:16:03,339 WARN org.apache.hadoop.ha.SshFenceByTcpPort: nc -z hadoop02 9000 via ssh: bash: nc: command not found
2018-06-21 10:16:03,340 INFO org.apache.hadoop.ha.SshFenceByTcpPort: Verified that the service is down.
2018-06-21 10:16:03,340 INFO org.apache.hadoop.ha.SshFenceByTcpPort.jsch: Disconnecting from hadoop02 port 22
2018-06-21 10:16:03,343 INFO org.apache.hadoop.ha.NodeFencer: ====== Fencing successful by method org.apache.hadoop.ha.SshFenceByTcpPort(null) ======
2018-06-21 10:16:03,344 INFO org.apache.hadoop.ha.ActiveStandbyElector: Writing znode /hadoop-ha/myha/ActiveBreadCrumb to indicate that the local node is t
nt active...
2018-06-21 10:16:03,347 INFO org.apache.hadoop.ha.SshFenceByTcpPort.jsch: Caught an exception, leaving main loop due to Socket closed
2018-06-21 10:16:03,352 INFO org.apache.hadoop.ha.ZKFailoverController: Trying to make NameNode at hadoop01/10.1.5.201:9000 active...
2018-06-21 10:16:04,210 INFO org.apache.hadoop.ha.ZKFailoverController: Successfully transitioned NameNode at hadoop01/10.1.5.201:9000 to active state
[hadoop@hadoop01 logs]$
[hadoop@hadoop01 logs]$
[hadoop@hadoop01 logs]$
```

查看 zookeeper 中有关 namedate 的节点信息：

```
[zk: localhost:2181(CONNECTED) 23] get /hadoop-ha/myha/ActiveBreadCrumb

myhann1hadoop01 F(>
cZxid = 0x1000001b5
ctime = Thu Jun 21 10:46:48 CST 2018
mZxid = 0x1000001b5
mtime = Thu Jun 21 10:46:48 CST 2018
pZxid = 0x1000001b5
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 27
numChildren = 0
[zk: localhost:2181(CONNECTED) 24]
[zk: localhost:2181(CONNECTED) 24] get /hadoop-ha/myha/ActiveStandbyElectorLock

myhann1hadoop01 F(>
cZxid = 0x1000001b4
ctime = Thu Jun 21 10:46:48 CST 2018
mZxid = 0x1000001b4
mtime = Thu Jun 21 10:46:48 CST 2018
pZxid = 0x1000001b4
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x1001dced1400014
dataLength = 27
numChildren = 0
```

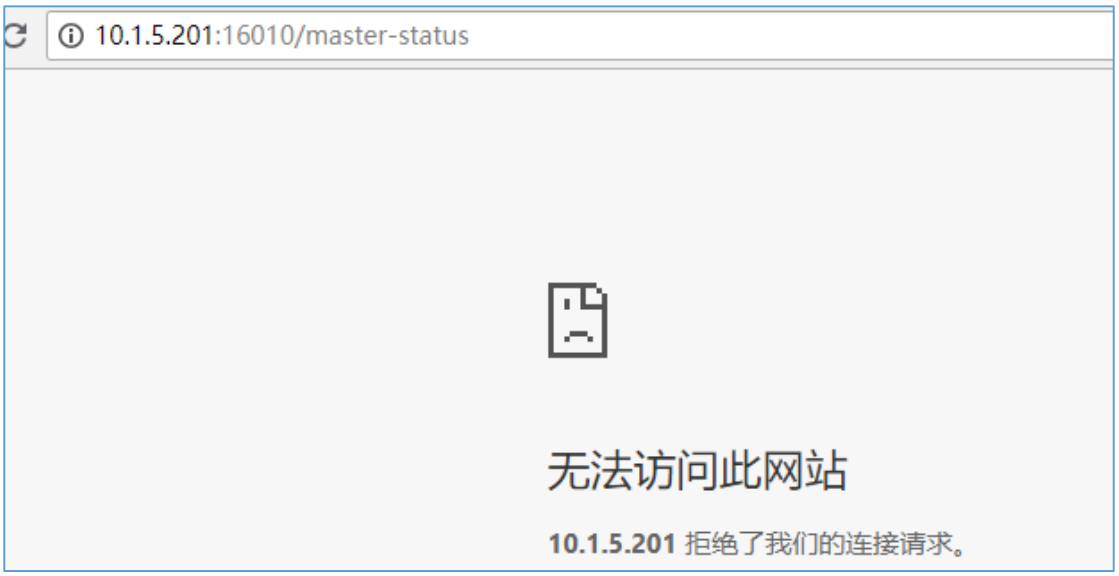
## 5.6 hbase-master 主备切换

切换前：hadoop01 主，hadoop02 从

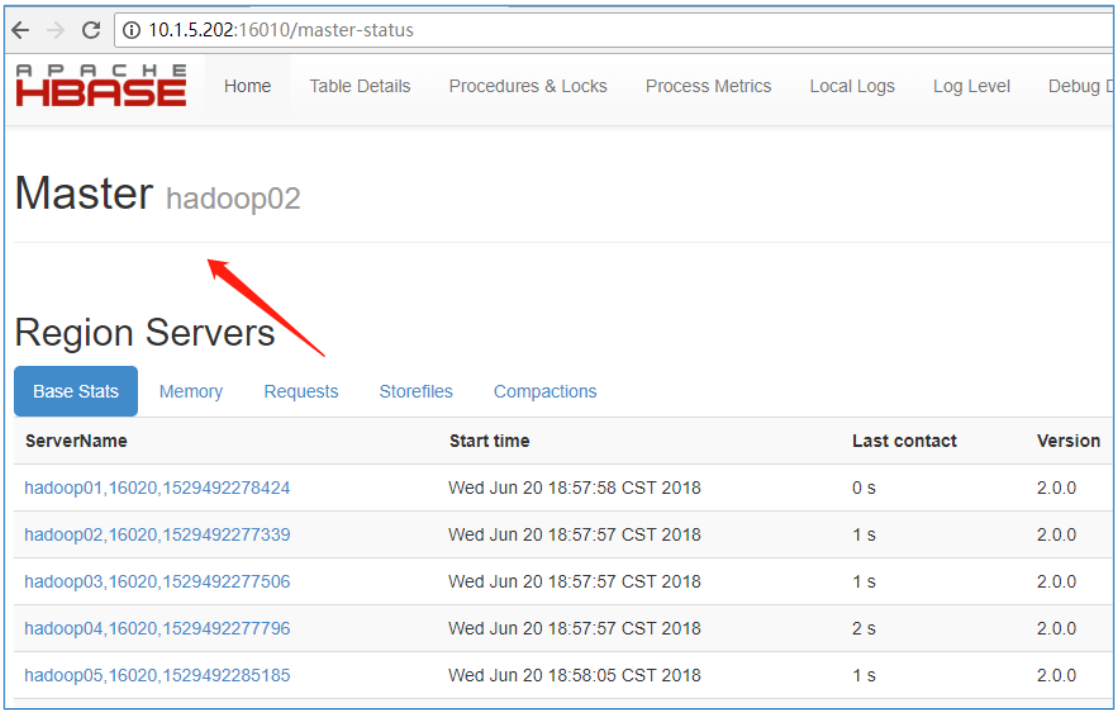
停止 hadoop01 的 hmaster

```
[hadoop@hadoop01 ~]$ hbase-daemon.sh stop master
running master, logging to /data/hbase-2.0.0/logs/hbase-hadoop-master-hadoop01.out
stopping master.
[hadoop@hadoop01 ~]$ jps
16129 DFSZKFailoverController
1137 ResourceManager
1331 NodeManager
612 HRegionServer
28516 DataNode
16360 Jps
28397 NameNode
[hadoop@hadoop01 ~]$
```

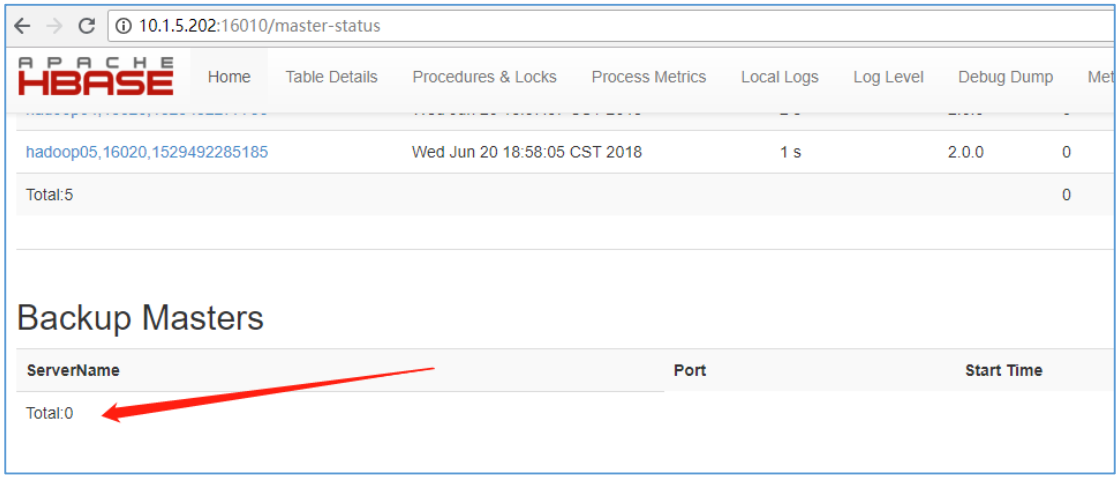
此时 hadoop01 无法访问




hadoop02 变成了 master

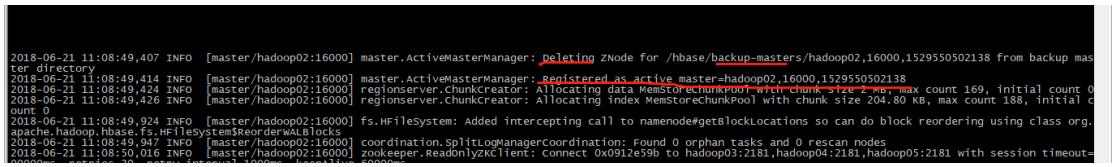



且没有从 master




查看日志：hbase-hadoop-master-hadoop02.log

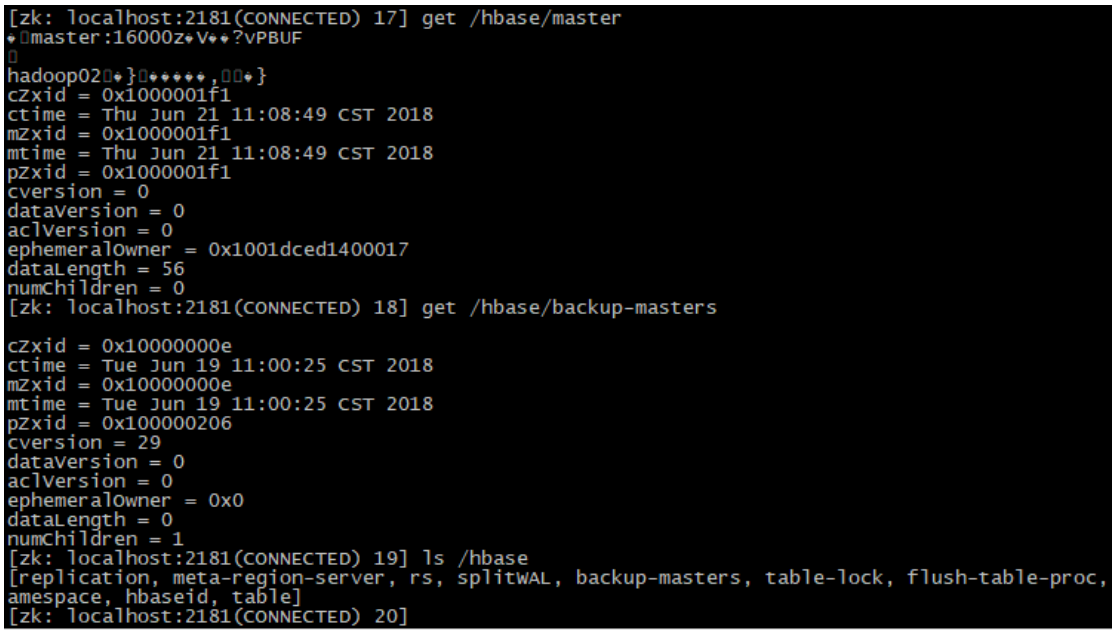
切换非常迅速，从这里也可以看出 zookeeper 的原理，先删除一个注册信息，再注册一条新信息，使之成为主 master.




2018-06-21 11:08:49,407 INFO [master/hadoop02:16000] master.ActiveMasterManager: Deleting ZNode for /hbase/backup-masters/hadoop02,16000,1529550502138 from backup master directory

2018-06-21 11:08:49,414 INFO [master/hadoop02:16000] master.ActiveMasterManager: Registered as active master=hadoop02,16000,1529550502138

zookeeper 中有关 hbase 的部分信息：



# 第六章 其他

## 6.1 zookeeper 集群安装

### 6.1.1 规划

zookeeper 集群至少需要三个几点，规划如下：
hadoop03，clientPort=2181
hadoop04，clientPort=2181
hadoop05，clientPort=2181

### 6.1.2 配置 jdk 环境

建议使用 oracle 的 jdk，当然使用系统自带的 openjdk 也是可以的，见 6.3。

### 6.1.3 下载 zookeeper

下载地址：http://www.apache.org/dist/zookeeper/

```
tar xvzf zookeeper-3.4.12.tgz -C /data/
```

### 6.1.4 修改配置文件

在 hadoop03 上：
cd /data/zookeeper-3.4.12/conf
cp zoo_sample.cfg zoo.cfg
修改配置文件如下：

```
cat zoo.cfg
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/data/zookeeper-data
clientPort=2181
server.1=hadoop03:2888:3888
server.2=hadoop04:2888:3888
server.3=hadoop05:2888:3888
```

### 6.1.5 配置其他节点

同 6.1.4 完全相同，配置 hadoop04 和 haoop05

### 6.1.6 新建好相关目录

hadoop03-05 上操作:

```
mkdir -pv /data/zookerper-data
```

### 6.1.7 新建 myid 文件

**hadoop03 上:**

```
echo 1 > /data/zookerper-data/myid
```

**hadoop04 上:**

```
echo 2 > /data/zookerper-data/myid
```

**hadoop05 上:**

```
echo 3 > /data/zookerper-data/myid
```

### 6.1.8 启动

```
cd /data/zookerper-3.4.12/bin
./zkServer.sh start
```

以此启动 3 个节点。

bin 目录下会生产 zookeeper.out 文件

### 6.1.9 查看状态

分布在 3 个节点上执行

```
/data/zookeeper-3.4.12/bin/zkServer.sh status
```

会有以下两种结果，leader 表示主，fllower 表示从:

Mode: leader

Mode: follower

3 点的分布输入 jps，会看下类似如下的进程

[yunMail@mail10-vmcore-38 ~]$ jps

43442 QuorumPeerMain

## 6.2 glibc 升级

### 6.2.1 背景

hadoop 机器部署完成之后，允许任何 hadoop 命令，比如 haoop fs -ls / ，如果出现有以下错误：

```
Failed to load native-hadoop with error: java.lang.UnsatisfiedLinkError: /home/hadoop/hadoop-2.9.0/lib/native/libhadoop.so.1.0.0: /lib64/libc.so.6: version `GLIBC_2.14' not found (required by /home/hadoop/hadoop-2.9.0/lib/native/libhadoop.so.1.0.0)
```

该错误表示 2.9.0 需要 glibc2.14 的支持，所以必须升级 glibc 库，如不升级，操作任何 hadoop 命令均有此提示。当然也可以安装 7.X 版本的 CentOS，glic 版本会比较高，就不需要升级。

查看当前 glibc 版本：

```
strings /lib64/libc.so.6 |grep GLIBC_
```

或者

```
getconf GNU_LIBC_VERSION
```

### 6.2.2 升级过程

```
wget http://ftp.gnu.org/gnu/glibc/glibc-2.14.tar.gz
tar xf glibc-2.14.tar.xz
cd glibc-2.14
mkdir build
cd build
../configure --prefix=/opt/glibc-2.14
make #这一步需要十几分钟
meke install
make localedata/install-locales #修改字符集，很重要，否则会各种报错，需要好几分钟

rm -f /lib64/libc.so.6 #先删除先前的 libc.so.6 软链，删除后所有的命令将不能使用
LD_PRELOAD=/opt/glibc-2.14/lib/libc-2.14.so ln -s /opt/glibc-2.14/lib/libc-2.14.so /lib64/libc.so.6 #创建新的软链

如果有问题，需要回退
rm -f /lib64/libc.so.6
LD_PRELOAD=/lib64/libc-2.12.so ln -s /lib64/libc-2.12.so /lib64/libc.so.6
```