**High Availability Bigdata**

**Cluster Solution**

**By Ivan**

目录

[大数据 HA集群设计方案 12](#_Toc495143520)

[一．先决条件 12](#_Toc495143521)

[1.安装环境 12](#_Toc495143522)

[2. 各主机节点职责分配 13](#_Toc495143523)

[3.目录规划 13](#_Toc495143524)

[4.架构图 14](#_Toc495143525)

[5.软件下载地址 14](#_Toc495143526)

[二．配置vm主机网络 14](#_Toc495143527)

[1.配置网络连接模式 14](#_Toc495143528)

[2.配置静态IP 15](#_Toc495143529)

[3.配置hosts 16](#_Toc495143530)

[4.配置DNS 17](#_Toc495143531)

[5.重启并验证网络服务 17](#_Toc495143532)

[6.关闭并禁止防火墙 17](#_Toc495143533)

[三．安装jdk与scala 19](#_Toc495143534)

[1.上传安装包 19](#_Toc495143535)

[2．解压安装jdk与scala 20](#_Toc495143536)

[3．配置jdk和scala环境变量 20](#_Toc495143537)

[4．使环境变量生效 21](#_Toc495143538)

[5．验证jdk与scala 21](#_Toc495143539)

[四．Clone虚拟机并配置 22](#_Toc495143540)

[1.关闭当前主机 22](#_Toc495143541)

[2.clone 4台主机分别命名为： 22](#_Toc495143542)

[3.依照以上步骤clone其它各节点 24](#_Toc495143543)

[4.启动各主机修改IP 24](#_Toc495143544)

[5.查看各节点 24](#_Toc495143545)

[6.分别重启网络服务 25](#_Toc495143546)

[7.如果遇到以下问题，请参考以下解决方案 26](#_Toc495143547)

[五．设置各主机节点免密码登录 27](#_Toc495143548)

[2.如果没有此命令执行下面命令安装： 27](#_Toc495143549)

[3.生成秘钥对配置 27](#_Toc495143550)

[4.将master-01上的“最全”公钥，复制到其它机器 30](#_Toc495143551)

[六．zookeeper安装与配置 32](#_Toc495143552)

[1.安装机器 32](#_Toc495143553)

[2.上传安装包 33](#_Toc495143554)

[3.在workspace目录中解压 33](#_Toc495143555)

[4.为zookeeper配置环境变量 33](#_Toc495143556)

[5.刷新配置 33](#_Toc495143557)

[6. 复制一份zookeeper配置文件 33](#_Toc495143558)

[7.修改配置文件 34](#_Toc495143559)

[8.在zookeeper-3.4.10/data目录下创建一个文件：myid 35](#_Toc495143560)

[9．为了方便集群管理脚本启动设置zkEnv.sh 35](#_Toc495143561)

[10.将zookeeper复制到slave-02,slave-03机器中 36](#_Toc495143562)

[11．分别设置各节点zookeeper下data目录中的myid 36](#_Toc495143563)

[12.分别修改slave-02与slave-03中的环境变量文件并刷新保存 37](#_Toc495143564)

[七．hadoop安装与配置 37](#_Toc495143565)

[1.上传hadoop安装包 37](#_Toc495143566)

[2.解压hadoop安装包到workspace中 38](#_Toc495143567)

[3.配置hadoop环境变量（hadoop用户级别） 38](#_Toc495143568)

[4.在hadoop目录中创建以下工作目录 38](#_Toc495143569)

[5.修改hadoop配置文件 39](#_Toc495143570)

[八． 启动服务 51](#_Toc495143571)

[1.启动Zookeeper集群 51](#_Toc495143572)

[2.创建命名空间（格式化bigdata） 52](#_Toc495143573)

[3.启动主NameNode节点 53](#_Toc495143574)

[4.格式备NameNode节点（namenode2） 54](#_Toc495143575)

[5.启动备NameNode节点（namenode2） 54](#_Toc495143576)

[6.启动集群 55](#_Toc495143577)

[7启动YARN（namenode1） 57](#_Toc495143578)

[9.重启集群 59](#_Toc495143579)

[10.查看hdfs 60](#_Toc495143580)

[九．spark安装与配置 60](#_Toc495143581)

[1.上传spark安装包 60](#_Toc495143582)

[2.解压spark包 60](#_Toc495143583)

[3.修改环境变量 60](#_Toc495143584)

[4.刷新环境变量 61](#_Toc495143585)

[5.配置$SPARK\_HOME/conf/spark-env.sh文件 61](#_Toc495143586)

[6．配置Spark History Server 61](#_Toc495143587)

[7.配置$SPARK\_HOME/conf/slaves文件 62](#_Toc495143588)

[8. 复制spark到各节点： 63](#_Toc495143589)

[9.修改各节点的hadoop用户的spark环境变量 63](#_Toc495143590)

[10.启动Spark(standlone,不启动就是yarn) 63](#_Toc495143591)

[11.启动Spark History Server 64](#_Toc495143592)

[十．配置spark HA 64](#_Toc495143593)

[1.说明 64](#_Toc495143594)

[2.修改配置文件 65](#_Toc495143595)

[3.将配置复制到其它节点上 65](#_Toc495143596)

[4.启动测试 66](#_Toc495143597)

[十一．测试集群 68](#_Toc495143598)

[1.测试hadoop 68](#_Toc495143599)

[2.测试spark 70](#_Toc495143600)

[十二．Kafka安装 71](#_Toc495143601)

[1.上传kafka安装包并解压 71](#_Toc495143602)

[2.创建kafka\_logs目录 71](#_Toc495143603)

[3.修改kafka配置文件 71](#_Toc495143604)

[4.配置kafka环境变量 73](#_Toc495143605)

[5.复制到slave-02与slave-03节点上 73](#_Toc495143606)

[6.修改slave-02与slave-03的配置文件 73](#_Toc495143607)

[7.分别配置节点slave-02与slave-03的环境变量 74](#_Toc495143608)

[8.测试kafka 75](#_Toc495143609)

[十三.hive安装 76](#_Toc495143610)

[1.上传并解压hive 76](#_Toc495143611)

[2.安装mysql 76](#_Toc495143612)

[3.解压hive 79](#_Toc495143613)

[4.配置hive环境变量 79](#_Toc495143614)

[5.上传mysql jdbc并copy到hive lib目录中 80](#_Toc495143615)

[6.配置hive-site.xml 80](#_Toc495143616)

[7.创建目录 82](#_Toc495143617)

[8.初始化数据库 82](#_Toc495143618)

[9.启动hive 82](#_Toc495143619)

[10.查看 83](#_Toc495143620)

[十四.sqoop安装 83](#_Toc495143621)

[1.上传并解压 83](#_Toc495143622)

[2.配置环境变量 83](#_Toc495143623)

[3.配置sqoop(conf) 83](#_Toc495143624)

[4.测试sqoop 84](#_Toc495143625)

[5.拷贝mysql jdbc到lib目录下 84](#_Toc495143626)

[6.测试 84](#_Toc495143627)

[十五．flume集群搭建 85](#_Toc495143628)

[1． 环境配置 85](#_Toc495143629)

[2． 架构图 85](#_Toc495143630)

[3． 集群角色分布 86](#_Toc495143631)

[4． Flume安装 86](#_Toc495143632)

[5． Flume配置 86](#_Toc495143633)

[6． 启动Flume 89](#_Toc495143634)

[7． 测试Flume集群 89](#_Toc495143635)

[8． 关闭Flume集群 90](#_Toc495143636)

[十六．集群管理 90](#_Toc495143637)

[1．正确的启动顺序 90](#_Toc495143638)

[2．脚本部署在下面的路径下 90](#_Toc495143639)

[创建Slaves 配置文件 90](#_Toc495143640)

[3.管理脚本 91](#_Toc495143641)

[5.集群访问说明： 97](#_Toc495143642)

# 大数据 HA集群设计方案

## 一．先决条件

### 1.安装环境

|  |  |  |
| --- | --- | --- |
| 软件名 | 版本 | 内核 |
| VirtualBox | Oracle 5.1.22 | R115126 |
| Linux | CentOS-6.9-x86\_64-bin-DVD | 6.9 |
| Java | jdk-8u131-linux-x64.tar.gz | build 1.8.0\_131-b11 |
| Scala | scala-2.10.6.tgz | 2.10.6 |
| Hadoop | hadoop-2.6.5.tar.gz | 2.6.5 |
| Spark | spark-1.6.3-bin-hadoop2.6.tgz | 1.6.3 |
| zookeeper | zookeeper-3.4.10.tar.gz | 3.4.10 |
| Hive | apache-hive-2.1.1-bin.tar.gz | 2.1.1 |
| kafka | kafka\_2.10-0.10.2.1.tgz | 2.10-0.10.2.1 |

### 2. 各主机节点职责分配

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IP | 主机名 | 安装软件 | 部署模块 | 运行进程 |
| **192.168.2.11** | **bigdata-master-01** | hadoop | NameNode | NameNode |
| Java/Scala | ResourceManager | DFSZKFailoverController |
| Spark | Spark | ResourceManager |
| Hive | HadoopHistory | Master |
| Mysql |  | JobHistoryServer |
| **192.168.2.12** | **bigdata-master-02** | hadoop | NameNode | NameNode |
| Java/Scala | ResourceManager | DFSZKFailoverController |
| Spark | Spark | ResourceManager |
|  | SparkHistory | HistoryServer |
| **192.168.2.13** | **bigdata-slave-01** | Hadoop | DataNode | DataNode |
| Java/Scala | NodeManager | NodeManager |
| Zookeeper | Zookeeper | JournalNode |
| Spark | Spark | QuorumPeerMain |
| Kafka | Kafka | Worker |
|  |  | Kafka |
| **192.168.2.14** | **bigdata-slave-02** | Hadoop | DataNode | NodeManager |
| Java/Scala | NodeManager | JournalNode |
| Zookeeper | Zookeeper | QuorumPeerMain |
| Spark | Spark | Worker |
| Kafka | Kafka | Kafka |
| **192.168.2.15** | **bigdata-slave-03** | Hadoop | DataNode | DataNode |
| Java/Scala | NodeManager | NodeManager |
| Zookeeper | Zookeeper | JournalNode |
| Spark | Spark | QuorumPeerMain |
| Kafka | Kafka | Worker |
|  |  | Kafka |

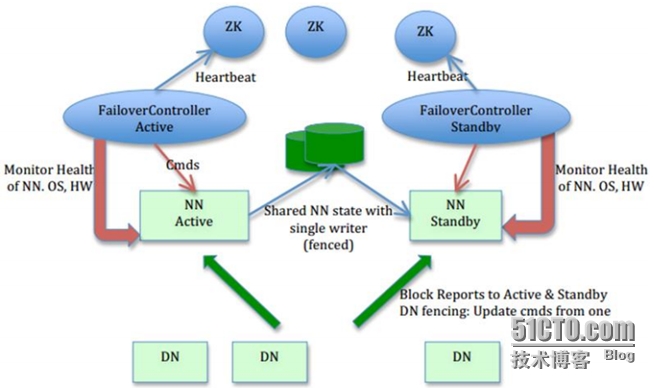
### 3.目录规划

bigdata---backup（安装包文件备份）

---tools（shell脚本）

---workspace（工作目录）

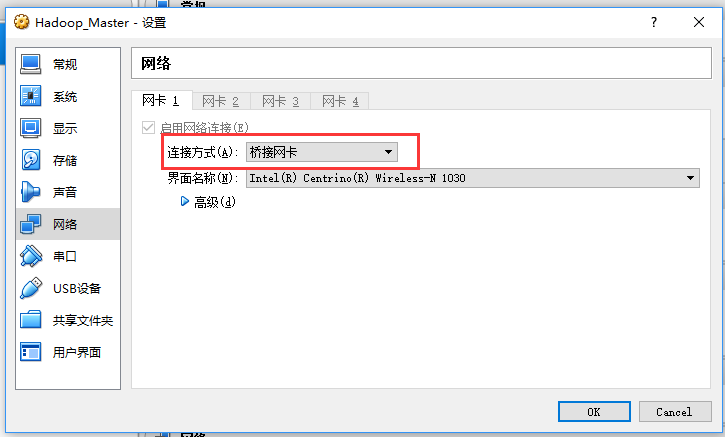
### 4.架构图



### 5.软件下载地址

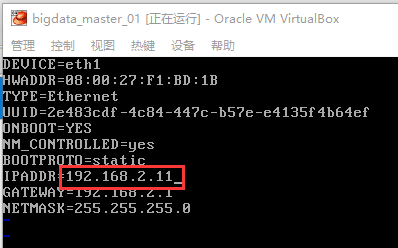
## 二．配置vm主机网络

### 1.配置网络连接模式



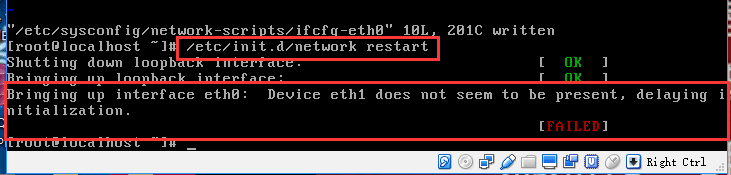
### 2.配置静态IP

vi /etc/sysconfig/network-scripts/incfg-eth0



刷新网络

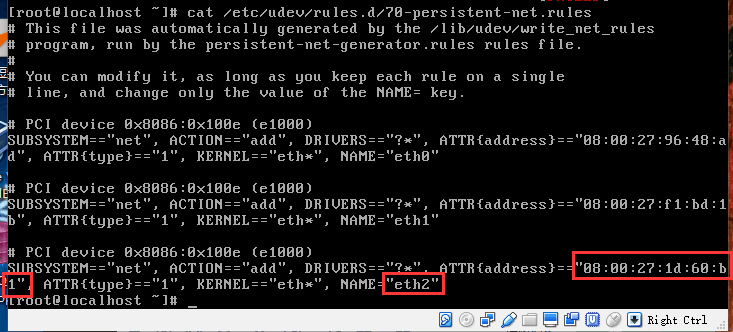
/etc/init.d/network restart



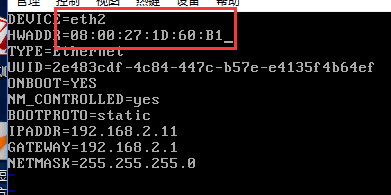
如果出现启动失败请参考以下方法解决

vi /etc/udev/rules.d/70-persistent-net.rules

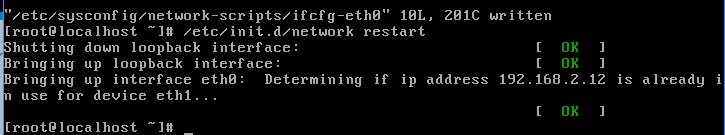
（记下name=xxx, mac=xxx）



将mac地址与name更新到网络配置文件中。

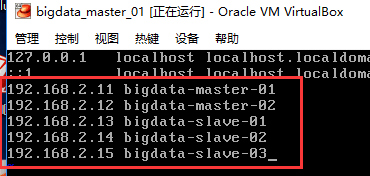


再次重启网络



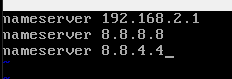
### 3.配置hosts

vi /etc/hosts



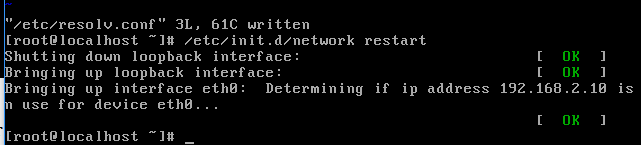
### 4.配置DNS

vi /etc/resolv.conf



### 5.重启并验证网络服务

/etc/init.d/network restart

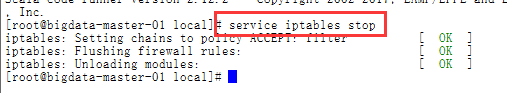


验证网络



### 6.关闭并禁止防火墙

关闭命令：service iptables stop



永久关闭命令：chkconfig iptables off



查看状态：service iptables status

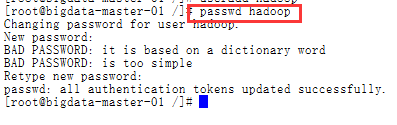


7.在各主机节点创建专门hadoop帐户

1）useradd hadoop (创建用户)



2）passwd hadoop （设置密码）



3）usermod -g root hadoop:为了方便，建议将hadoop加入root用户组，执行完后hadoop即归属于root组了



4）id hadoop 查看输出验证一下，如果看到类似下面的输出：

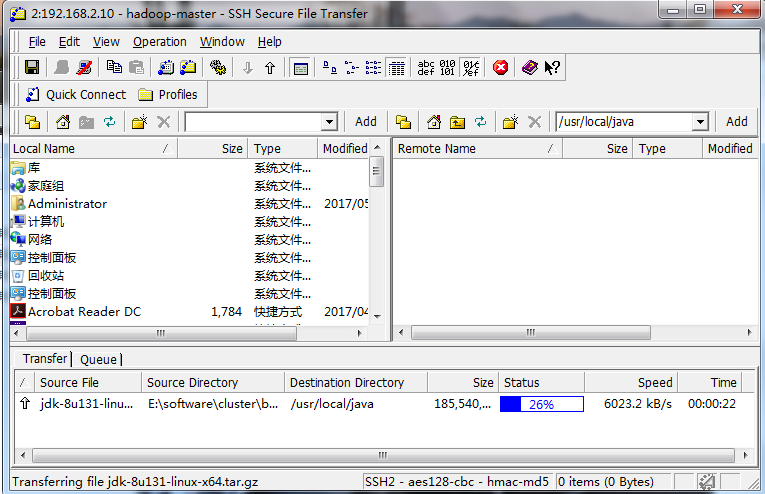
uid=500(hadoop) gid=0(root) 组=0(root)就表示OK了



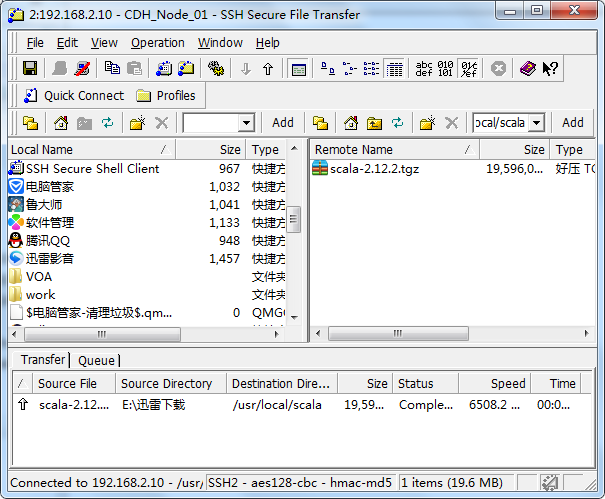
## 三．安装jdk与scala

### 1.上传安装包

jdk

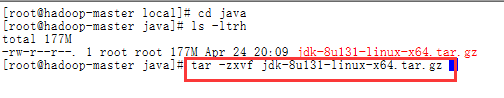


Scala

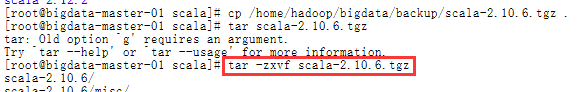


### 2．解压安装jdk与scala

jdk



scala



### 3．配置jdk和scala环境变量

jdk

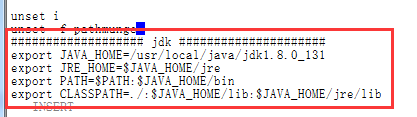
vi /etc/profile

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_131

export JRE\_HOME=$JAVA\_HOME/jre

export PATH=$PATH:$JAVA\_HOME/bin

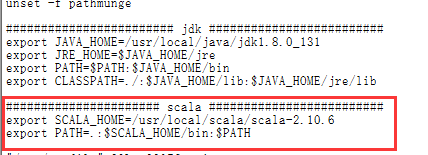
export CLASSPATH=./:$JAVA\_HOME/lib:$JAVA\_HOME/jre/lib



Scala

export SCALA\_HOME=/usr/local/scala/scala-2.10.6

export PATH=.:$SCALA\_HOME/bin:$PATH



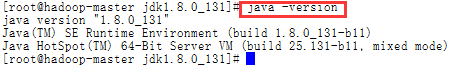
### 4．使环境变量生效

source /etc/profile

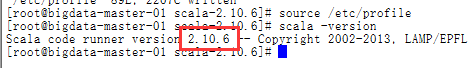


### 5．验证jdk与scala

java -version



scala -version



## 四．Clone虚拟机并配置

### 1.关闭当前主机

### 2.clone 4台主机分别命名为：

bigdata-master-02

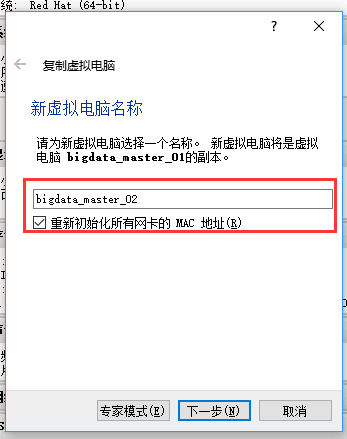
bigdata-slave-01

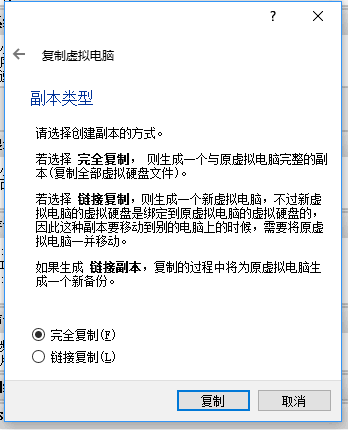
bigdata-slave-02

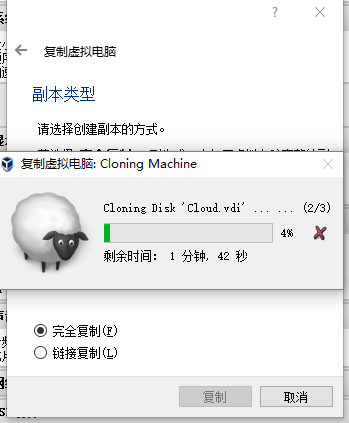
bigdata-slave-03

控制->复制

节点命名为：bigdata-xxx-xx



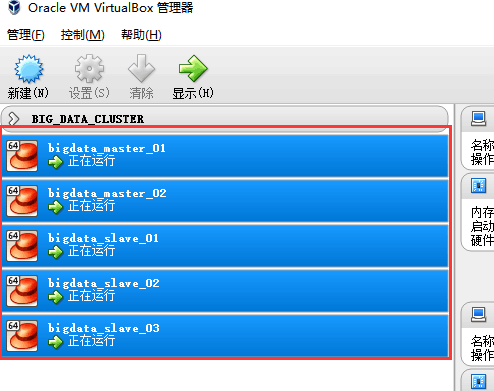




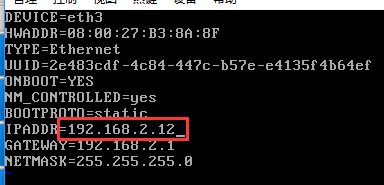
### 3.依照以上步骤clone其它各节点

### 4.启动各主机修改IP

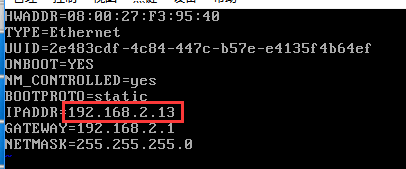
### 5.查看各节点



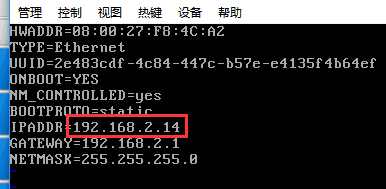
bigdata-master-02



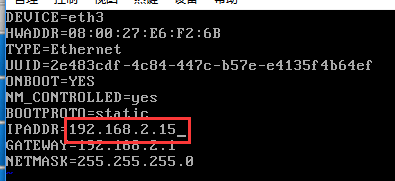
bigdata-slave-01



bigdata-slave-02



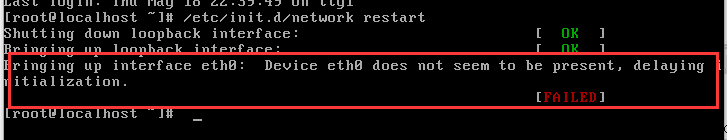
bigdata-slave-03



### 6.分别重启网络服务

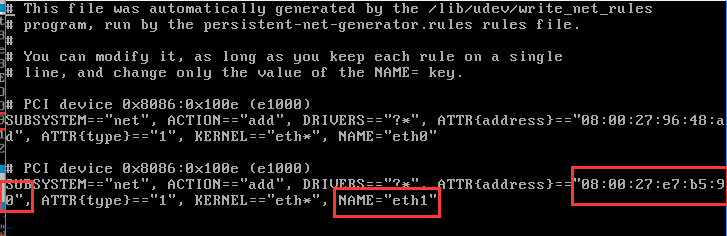
/etc/init.d/network restart

### 7.如果遇到以下问题，请参考以下解决方案

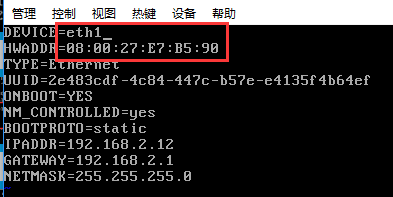


vi /etc/udev/rules.d/70-persistent-net.rules

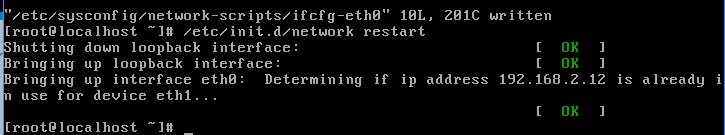
（记下name=thN,mac=08.00.27.e7.b5.90）



将mac地址与name更新到网络配置文件中。



再次重启网络



## 五．设置各主机节点免密码登录

Master（NameNode | JobTracker）作为客户端，要实现无密码公钥认证，连接到服务器Salve（DataNode | Tasktracker）上时，需要在Master上生成一个密钥对，包括一个公钥和一个私钥，而后将公钥复制到所有的Slave上。当Master通过SSH连接Salve时，Salve就会生成一个随机数并用Master的公钥对随机数进行加密，并发送给Master。Master收到加密数之后再用私钥解密，并将解密数回传给Slave，Slave确认解密数无误之后就允许Master进行连接了。这就是一个公钥认证过程，其间不需要用户手工输入密码。

**1.在各节点输入 scp 查找当前系统是否已经安装SSH**  


### 2.如果没有此命令执行下面命令安装：

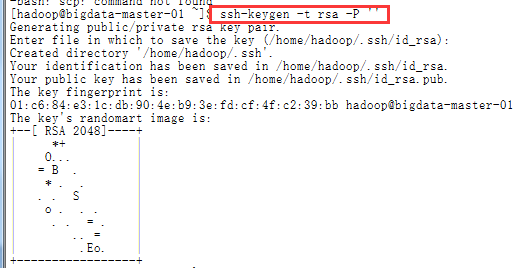
yum install openssh-clients



### 3.生成秘钥对配置

a) 以hadoop身份登录各节点机器（master-01,master-02,slave-01,slave-02,slave-03）

执行 ssh-keygen -t rsa -P '' 生成公钥、密钥



cat .ssh/id\_rsa.pub >> .ssh/authorized\_keys

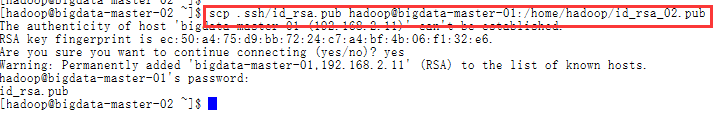
chmod 600 .ssh/authorized\_keys



b)然后用scp命令，把公钥文件发放给master-01节点

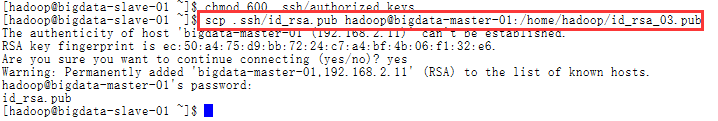
master-02上：

scp .ssh/id\_rsa.pub [hadoop@bigdata-master-01:/home/hadoop/id\_rsa\_02.pub](mailto:hadoop@bigdata-master-01:/home/hadoop/id_rsa_02.pub)



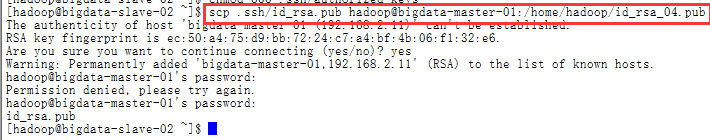
slave-01上：

scp .ssh/id\_rsa.pub [hadoop@bigdata-master-01:/home/hadoop/id\_rsa\_03.pub](mailto:hadoop@bigdata-master-01:/home/hadoop/id_rsa_03.pub)



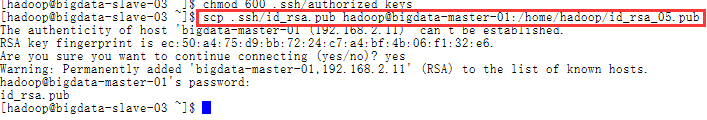
slave-02上：

scp .ssh/id\_rsa.pub [hadoop@bigdata-master-01:/home/hadoop/id\_rsa\_04.pub](mailto:hadoop@bigdata-master-01:/home/hadoop/id_rsa_04.pub)



slave-03上：

scp .ssh/id\_rsa.pub [hadoop@bigdata-master-01:/home/hadoop/id\_rsa\_05.pub](mailto:hadoop@bigdata-master-01:/home/hadoop/id_rsa_05.pub)



以上执行完后，回到master-01中，查看下/home/hadoop目录，应该有4个新文件id\_rsa\_02.pub、id\_rsa\_03.pub、id\_rsa\_04.pub、id\_rsa\_05.pub

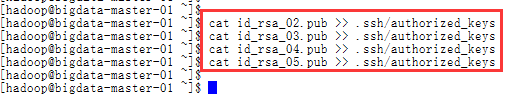
然后在master-01上，导入这4个公钥

cat id\_rsa\_02.pub >> .ssh/authorized\_keys

cat id\_rsa\_03.pub >> .ssh/authorized\_keys

cat id\_rsa\_04.pub >> .ssh/authorized\_keys

cat id\_rsa\_05.pub >> .ssh/authorized\_keys



这样，master-01这台机器上，就有所有5台机器的公钥了。

### 4.将master-01上的“最全”公钥，复制到其它机器

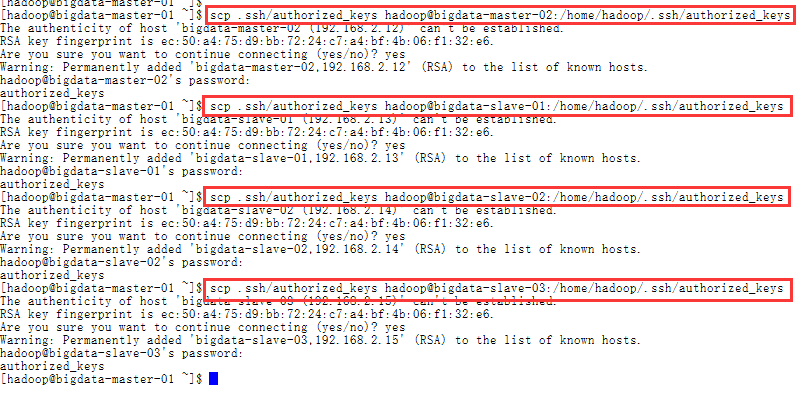
a) 继续保持在master-01上，执行以下命令：

scp .ssh/authorized\_keys [hadoop@bigdata-master-02:/home/hadoop/.ssh/authorized\_keys](mailto:hadoop@bigdata-master-02:/home/hadoop/.ssh/authorized_keys)

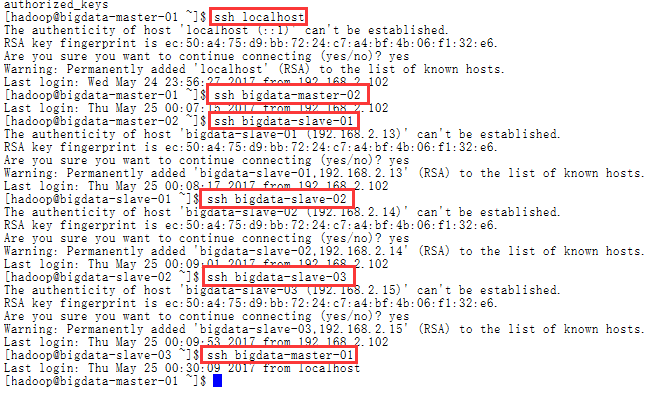
scp .ssh/authorized\_keys hadoop@bigdata-slave-01:/home/hadoop/.ssh/authorized\_keys

scp .ssh/authorized\_keys [hadoop@bigdata-slave-02:/home/hadoop/.ssh/authorized\_keys](mailto:hadoop@bigdata-slave-02:/home/hadoop/.ssh/authorized_keys)

scp .ssh/authorized\_keys hadoop@bigdata-slave-03:/home/hadoop/.ssh/authorized\_keys



b)验证免密SSH



c)创建大数据集群目录结构

在master-01创建以下目录

mkdir /home/hadoop/bigdata（主路径）



mkdir /home/hadoop/bigdata/backup（安装包备份）



mkdir /home/hadoop/bigdata/workspace（集群工作路径）



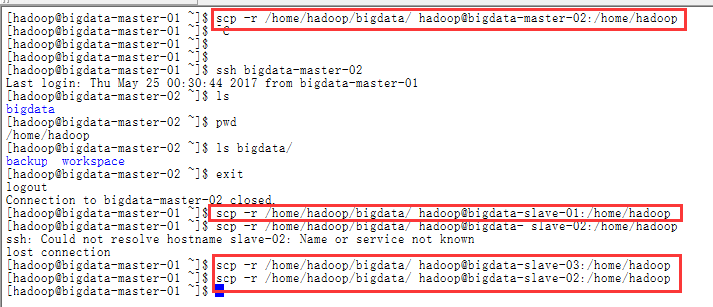
将目录结构复制到其它节点中

scp -r /home/hadoop/bigdata/ hadoop@bigdata-master-02:/home/hadoop

scp -r /home/hadoop/bigdata/ hadoop@bigdata-slave-01:/home/hadoop

scp -r /home/hadoop/bigdata/ hadoop@bigdata-slave-02:/home/hadoop

scp -r /home/hadoop/bigdata/ hadoop@bigdata-slave-03:/home/hadoop



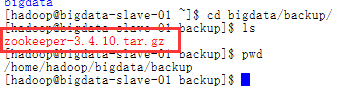
## 六．zookeeper安装与配置

### 1.安装机器

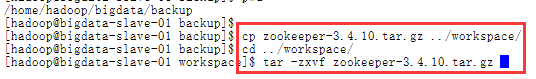
（**bigdata-slave-01，bigdata-slave-02，bigdata-slave-03**）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **192.168.2.13** | **bigdata-slave-01** | **Hadoop**  **Java/Scala**  **zookeeper** | **DataNode**  **NodeManager**  **Zookeeper** | **DataNode**  **NodeManager**  **JournalNode**  **QuorumPeerMain** |
| **192.168.2.14** | **bigdata-slave-02** | **Hadoop**  **Java/Scala**  **zookeeper** | **DataNode**  **NodeManager**  **Zookeeper** | **DataNode**  **NodeManager**  **JournalNode**  **QuorumPeerMain** |
| **192.168.2.15** | **bigdata-slave-03** | **Hadoop**  **Java/Scala**  **zookeeper** | **DataNode**  **NodeManager**  **Zookeeper** | **DataNode**  **NodeManager**  **JournalNode**  **QuorumPeerMain** |

### 2.上传安装包



### 3.在workspace目录中解压



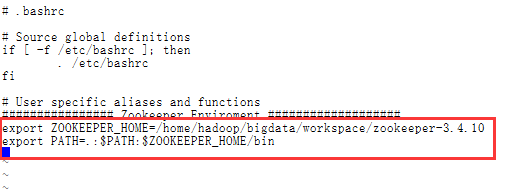
### 4.为zookeeper配置环境变量

vi ~/.bashrc

################ Zookeeper Enviroment ###################

export ZOOKEEPER\_HOME=/home/hadoop/bigdata/workspace/zookeeper-3.4.10

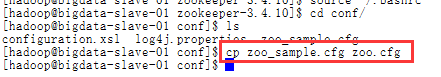
export PATH=.:$PATH:$ZOOKEEPER\_HOME/bin



### 5.刷新配置

source ~/.bashrc

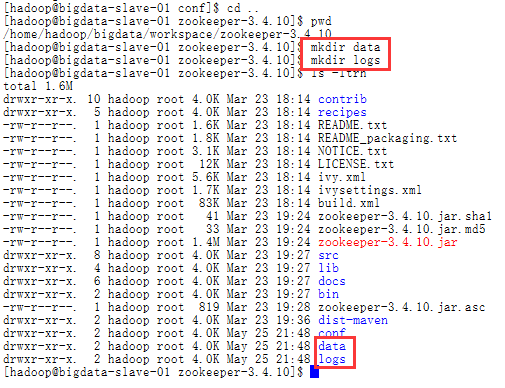
### 6. 复制一份zookeeper配置文件



在zookeeperk中创建工作目录

mkdir data

mkdir logs



### 7.修改配置文件

保存数据目录

dataDir=/home/hadoop/bigdata/workspace/zookeeper-3.4.10/data

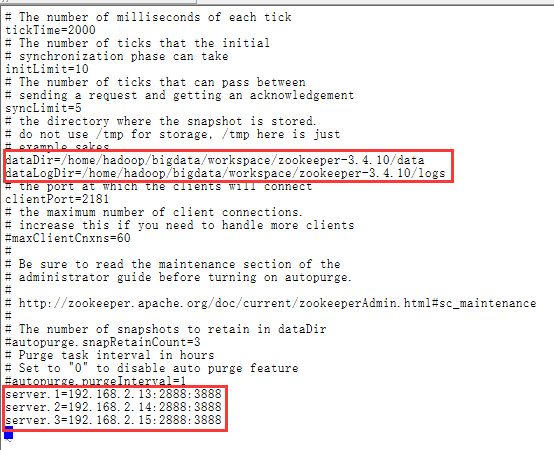
保存日志目录

dataLogDir=/home/hadoop/bigdata/workspace/zookeeper-3.4.10/logs

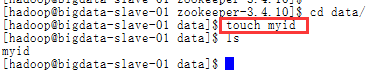
server.1=192.168.2.13:2888:3888

server.2=192.168.2.14:2888:3888

server.3=192.168.2.15:2888:3888



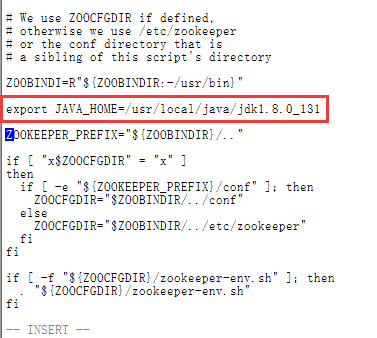
### 8.在zookeeper-3.4.10/data目录下创建一个文件：myid



### 9．为了方便集群管理脚本启动设置zkEnv.sh

vi /home/hadoop/bigdata/workspace/zookeeper-3.4.10/bin/zkEnv.sh

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_131



### 10.将zookeeper复制到slave-02,slave-03机器中

scp -r /home/hadoop/bigdata/workspace/zookeeper-3.4.10 hadoop@bigdata-slave-02:/home/hadoop/bigdata/workspace



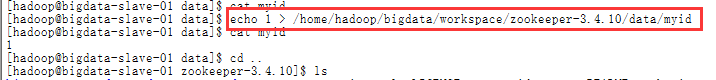
scp -r /home/hadoop/bigdata/workspace/zookeeper-3.4.10 hadoop@bigdata-slave-03:/home/hadoop/bigdata/workspace



### 11．分别设置各节点zookeeper下data目录中的myid

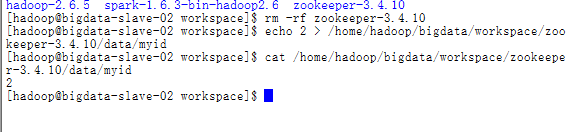
在slave-01 节点中 执行：

echo 1 > /home/hadoop/bigdata/workspace/zookeeper-3.4.10/data/myid



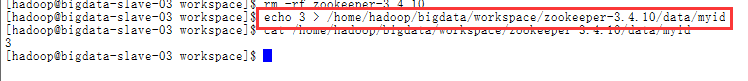
在slave-02 节点中 执行：

echo 2 > /home/hadoop/bigdata/workspace/zookeeper-3.4.10/data/myid



在slave-03 节点中 执行：

echo 3 > /home/hadoop/bigdata/workspace/zookeeper-3.4.10/data/myid



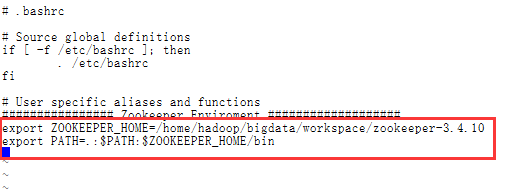
### 12.分别修改slave-02与slave-03中的环境变量文件并刷新保存

vi ~/.bashrc

################ Zookeeper Enviroment ###################

export ZOOKEEPER\_HOME=/home/hadoop/bigdata/workspace/zookeeper-3.4.10

export PATH=.:$PATH:$ZOOKEEPER\_HOME/bin

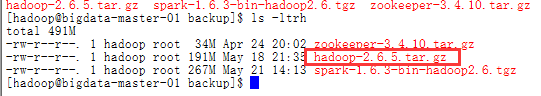


刷新配置

source ~/.bashrc

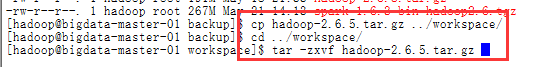
## 七．hadoop安装与配置

### 1.上传hadoop安装包



### 2.解压hadoop安装包到workspace中

tar -zxvf hadoop-2.6.5.tar.gz



### 3.配置hadoop环境变量（hadoop用户级别）

vi ~/.bashrc

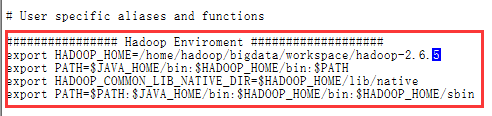
################ Hadoop Enviroment ###################

export HADOOP\_HOME=/home/hadoop/bigdata/workspace/hadoop-2.6.5

export PATH=$JAVA\_HOME/bin:$HADOOP\_HOME/bin:$PATH

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

export PATH=$PATH:$JAVA\_HOME/bin:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin



刷新配置

source ~/.bashrc

### 4.在hadoop目录中创建以下工作目录

mkdir tmp

mkdir dfs

mkdir logs

mkdir logs/hadoop

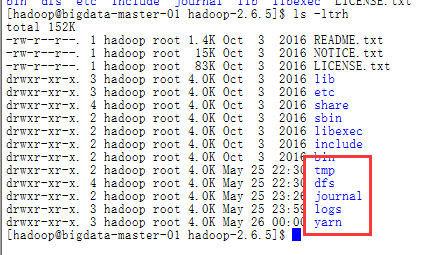
mkdir journal

mkdir dfs/data

mkdir dfs/name

mkdir yarn

mkdir yarn/local



### 5.修改hadoop配置文件

一共有7个文件要修改：

$HADOOP\_HOME/etc/hadoop/hadoop-env.sh

$HADOOP\_HOME/etc/hadoop/yarn-env.sh

$HADOOP\_HOME/etc/hadoop/core-site.xml

$HADOOP\_HOME/etc/hadoop/hdfs-site.xml

$HADOOP\_HOME/etc/hadoop/mapred-site.xml

$HADOOP\_HOME/etc/hadoop/yarn-site.xml

$HADOOP\_HOME/etc/hadoop/slaves

其中$HADOOP\_HOME表示hadoop根目录，本文中默认为/home/hadoop/bigdata/workspace/hadoop-2.6.5

a) 修改hadoop-env.sh 、yarn-env.sh

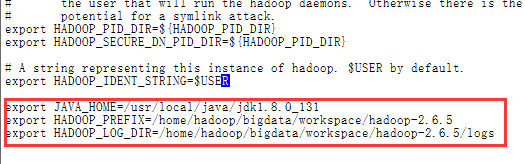
这二个文件主要是修改JAVA\_HOME后的目录，改成实际本机jdk所在目录位置vi etc/hadoop/hadoop-env.sh （及 vi etc/hadoop/yarn-env.sh）找到下面这行的位置，改成（jdk目录位置，大家根据实际情况修改）

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_131

export HADOOP\_PREFIX=/home/hadoop/bigdata/workspace/hadoop-2.6.5

export HADOOP\_LOG\_DIR=/home/hadoop/bigdata/workspace/hadoop-2.6.5/logs

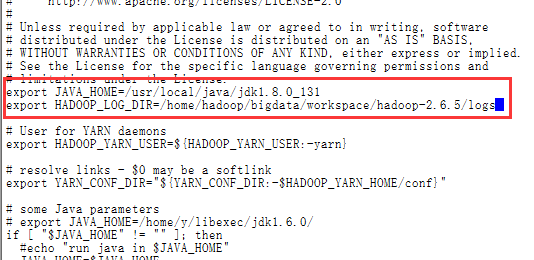
vi /home/hadoop/bigdata/workspace/hadoop-2.6.5/etc/hadoop/hadoop-env.sh



export JAVA\_HOME=/usr/local/java/jdk1.8.0\_131

export HADOOP\_PREFIX=/home/hadoop/bigdata/workspace/hadoop-2.6.5

vi /home/hadoop/bigdata/workspace/hadoop-2.6.5/etc/hadoop/yarn-env.sh



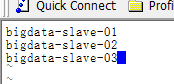
b) **配置文件:slaves**

bigdata-slave-01

bigdata-slave-02

bigdata-slave-03

vi /home/hadoop/bigdata/workspace/hadoop-2.6.5/etc/hadoop/slaves



c) core-site.xml 参考下面的内容修改：

<configuration>

　　<!--这里的值指的是默认的HDFS路径。该值来自于hdfs-site.xml中的配置-->

<property>

<name>fs.defaultFS</name>

<value>hdfs://bigdata</value>

</property>

<!-- 指定hadoop临时目录 -->

<property>

<name>hadoop.tmp.dir</name>

<value>/home/hadoop/bigdata/workspace/hadoop-2.6.5/tmp</value>

<description>Abase for other temporary directories.</description>

</property>

<!--指定可以在任何IP访问-->

<property>

<name>hadoop.proxyuser.hduser.hosts</name>

<value>\*</value>

</property>

<!--指定所有用户可以访问-->

<property>

<name>hadoop.proxyuser.hduser.groups</name>

<value>\*</value>

</property>

<!--这里是ZooKeeper集群的地址和端口。注意，数量一定是奇数，且不少于三个节点-->

<property>

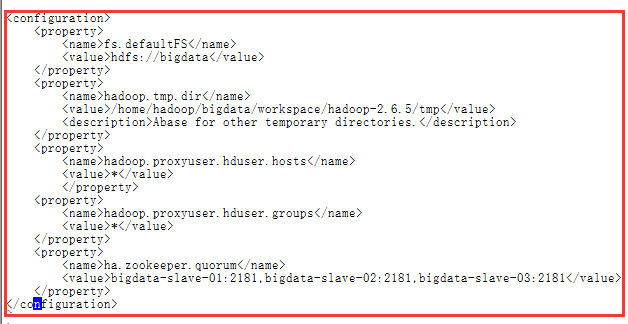
<name>ha.zookeeper.quorum</name>

<value>bigdata-slave-01:2181,bigdata-slave-02:2181,bigdata-slave-03:2181</value>

</property>

</configuration>

vi /home/hadoop/bigdata/workspace/hadoop-2.6.5/etc/hadoop/core-site.xml



d) hdfs-site.xml参考下面的内容修改：

<configuration>

<!--指定hdfs的集群名为bigdata，需要和core-site.xml中的保持一致 -->

<property>

<name>dfs.nameservices</name>

<value>bigdata</value>

</property>

<!-- ns1下面有两个NameNode，分别是nameNode1，nameNode2 -->

<property>

<name>dfs.ha.namenodes.bigdata</name>

<value>nameNode1, nameNode2</value>

</property>

<!-- nameNode1的RPC通信地址 -->

<property>

<name>dfs.namenode.rpc-address.bigdata.nameNode1</name>

<value>bigdata-master-01:9000</value>

</property>

<!-- nameNode2的RPC通信地址 -->

<property>

<name>dfs.namenode.rpc-address.bigdata.nameNode2</name>

<value>bigdata-master-02:9000</value>

</property>

<!-- nameNode1的http通信地址 -->

<property>

<name>dfs.namenode.http-address.bigdata.nameNode1</name>

<value>bigdata-master-01:50070</value>

</property>

<!-- nameNode2的http通信地址 -->

<property>

<name>dfs.namenode.http-address.bigdata.nameNode2</name>

<value>bigdata-master-02:50070</value>

</property>

<!--指定JournalNode集群在对NameNode的目录进行共享时，自己存储数据的磁盘路径-->

<property>

<name>dfs.journalnode.edits.dir</name>

<value>/home/hadoop/bigdata/workspace/hadoop-2.6.5/journal</value>

</property>

<!-- 指定cluster1的两个NameNode共享edits文件目录时，使用的JournalNode集群信息-->

<property>

<name>dfs.namenode.shared.edits.dir</name>

<value>qjournal://bigdata-slave-01:8485;bigdata-slave-02:8485;bigdata-slave-03:8485/bigdata</value>

</property>

<!-- qj方式共享edits。使用此方式-->

<property>

<name>dfs.namenode.edits.journal-plugin.qjournal</name>

<value>org.apache.hadoop.hdfs.qjournal.client.QuorumJournalManager</value>

</property>

<!--开启NameNode失败自动切换-->

<property>

<name>dfs.ha.automatic-failover.enabled</name>

<value>true</value>

</property>

<!--设置为true，允许NN尝试恢复之前失败的dfs.namenode.name.dir目录。在创建checkpoint是做此尝试。如果设置多个磁盘，建议允许-->

<property>

<name>dfs.namenode.name.dir.restore</name>

<value>true</value>

</property>

<!--指定出故障时，哪个实现类负责执行故障切换-->

<property>

<name>dfs.client.failover.proxy.provider.bigdata</name>

<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>

</property>

<!--一旦需要NameNode切换，使用ssh方式进行操作-->

<property>

<name>dfs.ha.fencing.methods</name>

<value>sshfence</value>

</property>

<!--如果使用ssh进行故障切换，使用ssh通信时用的密钥存储的位置-->

<property>

<name>dfs.ha.fencing.ssh.private-key-files</name>

<value>/home/hadoop/.ssh/id\_rsa</value>

</property>

<!--指定namenode名称空间的存储地址, 可以是按逗号分隔的目录列表-->

<property>

<name>dfs.namenode.name.dir</name>

<value>file:/home/hadoop/bigdata/workspace/hadoop-2.6.5/dfs/name</value>

</property>

<!--指定datanode数据存储地址, 可以是按逗号分隔的目录列表-->

<property>

<name>dfs.datanode.data.dir</name>

<value>file:/home/hadoop/bigdata/workspace/hadoop-2.6.5/dfs/data</value>

</property>

<!--指定数据冗余份数,不超过机器数即可-->

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

<!--指定可以通过web访问hdfs目录-->

<property>

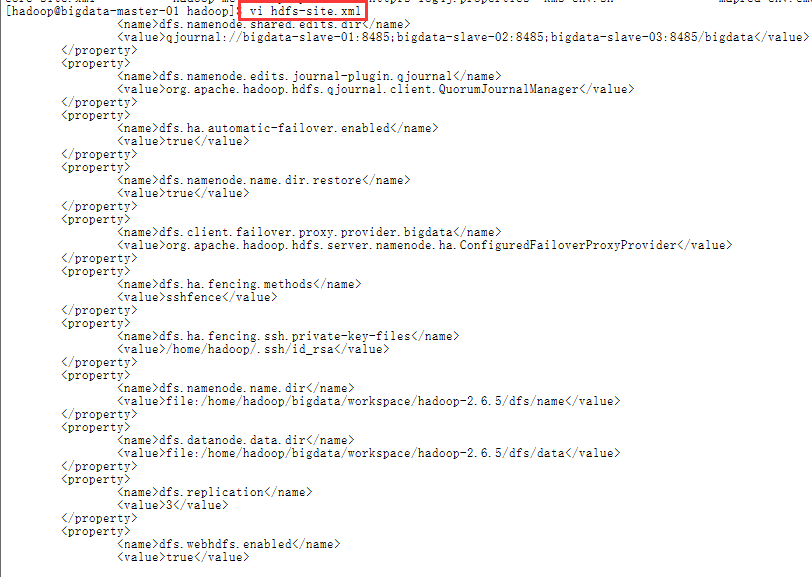
<name>dfs.webhdfs.enabled</name>

<value>true</value>

</property>

</configuration>

vi /home/hadoop/bigdata/workspace/hadoop-2.6.5/etc/hadoop/hdfs-site.xml



d) mapred-site.xml参考下面的内容修改：

<configuration>

<!-- 配置 MapReduce Applications -->

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

<!-- 配置 MapReduce JobHistory Server 地址 ，默认: 0.0.0.0:10020 -->

<property>

<name>mapreduce.jobhistory.address</name>

<value>bigdata-master-01:10020</value>

</property>

<!-- 配置 MapReduce JobHistory Server web ui 地址， 默认: 0.0.0.0:19888 -->

<property>

<name>mapreduce.jobhistory.webapp.address</name>

<value>bigdata-master-01:19888</value>

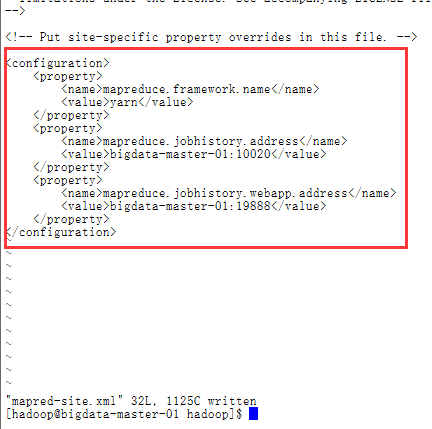
</property>

</configuration>

cd /home/hadoop/bigdata/workspace/hadoop-2.6.5/etc/hadoop/

cp mapred-site.xml.template mapred-site.xml

vi mapred-site.xml



e)yarn-site.xml参考下面的内容修改：

**注意：配置yarn.resourcemanager.ha.id时，在master-01上配置rm1,在master-02上配置rm2,注意：一般都喜欢把配置好的文件远程复制到其它机器上，但这个在YARN的另一个机器上一定要修改，其他机器上不配置此项**

<configuration>

<!--rm失联后重新链接的时间-->

<property>

<name>yarn.resourcemanager.connect.retry-interval.ms</name>

<value>2000</value>

</property>

<!--开启resource manager HA,默认为false-->

<property>

<name>yarn.resourcemanager.ha.enabled</name>

<value>true</value>

</property>

<!--配置resource manager 命名-->

<property>

<name>yarn.resourcemanager.ha.rm-ids</name>

<value>rm1,rm2</value>

</property>

<!--开启resourcemanager故障自动切换，指定机器-->

<property>

<name>yarn.resourcemanager.ha.automatic-failover.enabled</name>

<value>true</value>

</property>

<property>

<name>yarn.resourcemanager.hostname.rm1</name>

<value>bigdata-master-01</value>

</property>

<property>

<name>yarn.resourcemanager.hostname.rm2</name>

<value>bigdata-master-02</value>

</property>

<!--在master-01上配置rm1,在master-02上配置rm2,注意：一般都喜欢把配置好的文件远程复制到其它机器上，但这个在YARN的另一个机器上一定要修改，其他机器上不配置此项-->

<property>

<name>yarn.resourcemanager.ha.id</name>

<value>rm1</value>

<description>If we want to launch more than one RM in single node, we need this configuration</description>

</property>

<!--开启resourcemanager自动恢复功能-->

<property>

<name>yarn.resourcemanager.recovery.enabled</name>

<value>true</value>

</property>

<!--用于持久存储的类。尝试开启-->

<property>

<name>yarn.resourcemanager.store.class</name>

<value>org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore</value>

</property>

<property>

<name>yarn.resourcemanager.zk-address</name>

<value>bigdata-slave-01:2181,bigdata-slave-02:2181,bigdata-slave-03:2181</value>

</property>

<!--失联等待连接时间-->

<property>

<name>yarn.app.mapreduce.am.scheduler.connection.wait.interval-ms</name>

<value>5000</value>

</property>

<!-- 集群的Id，使用该值确保RM不会做为其它集群的active -->

<property>

<name>yarn.resourcemanager.cluster-id</name>

<value>bigdata</value>

</property>

<!--配置rm1-->

<property>

<name>yarn.resourcemanager.address.rm1</name>

<value>bigdata-master-01:8132</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address.rm1</name>

<value>bigdata-master-01:8130</value>

</property>

<!-- RM的网页接口地址：端口-->

<property>

<name>yarn.resourcemanager.webapp.address.rm1</name>

<value>bigdata-master-01:8188</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address.rm1</name>

<value>bigdata-master-01:8131</value>

</property>

<!-- RM管理接口地址：端口-->

<property>

<name>yarn.resourcemanager.admin.address.rm1</name>

<value>bigdata-master-01:8033</value>

</property>

<property>

<name>yarn.resourcemanager.ha.admin.address.rm1</name>

<value>bigdata-master-01:23142</value>

</property>

<!--配置rm2-->

<property>

<name>yarn.resourcemanager.address.rm2</name>

<value>bigdata-master-02:8132</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address.rm2</name>

<value>bigdata-master-02:8130</value>

</property>

<!-- RM的网页接口地址：端口-->

<property>

<name>yarn.resourcemanager.webapp.address.rm2</name>

<value>bigdata-master-02:8188</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address.rm2</name>

<value>bigdata-master-02:8131</value>

</property>

<!-- RM管理接口地址：端口-->

<property>

<name>yarn.resourcemanager.admin.address.rm2</name>

<value>bigdata-master-02:8033</value>

</property>

<property>

<name>yarn.resourcemanager.ha.admin.address.rm2</name>

<value>bigdata-master-02:23142</value>

</property>

<!--请配置为：mapreduce\_shuffle，在Yarn上开启MR的必须项-->

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<!-- nodemanager本地文件存储目录-->

<property>

<name>yarn.nodemanager.local-dirs</name>

<value>/home/hadoop/bigdata/workspace/hadoop-2.6.5/yarn/local</value>

</property>

<!--存储container日志的地方-->

<property>

<name>yarn.nodemanager.log-dirs</name>

<value>/home/hadoop/bigdata/workspace/hadoop-2.6.5/log/hadoop</value>

</property>

<property>

<name>yarn.nodemanager.resource.memory-mb</name>

<value>2048</value>

<discription>每个节点可用内存,单位MB</discription>

</property>

<property>

<name>yarn.scheduler.minimum-allocation-mb</name>

<value>258</value>

<discription>单个任务可申请最少内存，默认1024MB</discription>

</property>

<property>

<name>yarn.scheduler.maximum-allocation-mb</name>

<value>2048</value>

<discription>单个任务可申请最大内存，默认8192MB</discription>

</property>

<property>

<name>yarn.log-aggregation-enable</name>

<value>true</value>

<discription>是否启用日志聚集功能</discription>

</property>

</configuration>

**vi yarn-site.xml**



f)复制hadoop工程到其它各节点中

scp -r /home/hadoop/bigdata/workspace/hadoop-2.6.5 hadoop@bigdata-master-02:/home/hadoop/bigdata/workspace

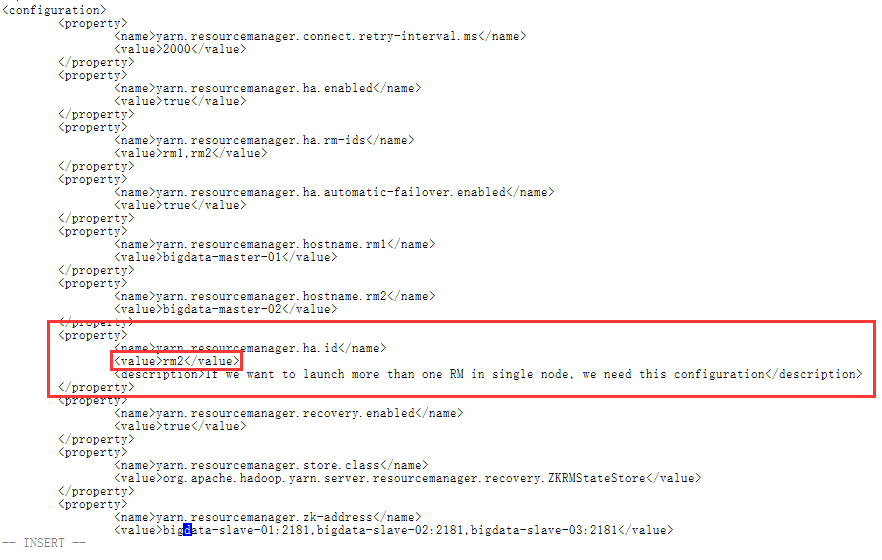
scp -r /home/hadoop/bigdata/workspace/hadoop-2.6.5 hadoop@bigdata-slave-01:/home/hadoop/bigdata/workspace

scp -r /home/hadoop/bigdata/workspace/hadoop-2.6.5 hadoop@bigdata-slave-02:/home/hadoop/bigdata/workspace

scp -r /home/hadoop/bigdata/workspace/hadoop-2.6.5 hadoop@bigdata-slave-03:/home/hadoop/bigdata/workspace

g)修改master-02节点yarn-site.xml文件中arn.resourcemanager.ha.id为rm2

vi /home/hadoop/bigdata/workspace/hadoop-2.6.5/etc/hadoop/yarn-site.xml



h)配置各节点hadoop的环境变量

################ Hadoop Enviroment ###################

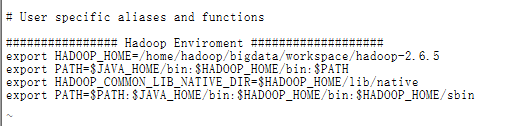
export HADOOP\_HOME=/home/hadoop/bigdata/workspace/hadoop-2.6.5

export PATH=$JAVA\_HOME/bin:$HADOOP\_HOME/bin:$PATH

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

export PATH=$PATH:$JAVA\_HOME/bin:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin

vi ~/.bashrc



source ~/.bashrc

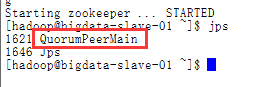
## 启动服务

### 1.启动Zookeeper集群

分别在slave-01,slave-02, slave-03上启动Zookeeper·

zkServer.sh start

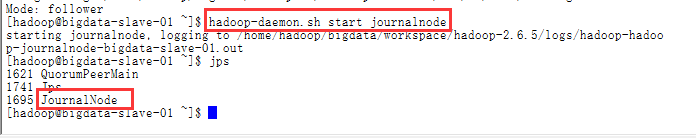
启动成功后slave-01,slave-02, slave-03会有**QuorumPeerMain**进程



### 2.创建命名空间（格式化bigdata）

格式化前,先在各journalnode(Slave)节点机器上先启动JournalNode进程

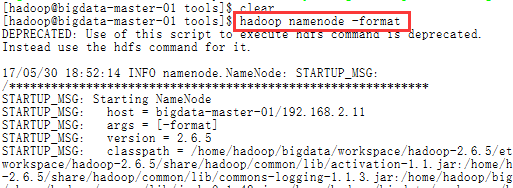
**hadoop-daemon.sh start journalnode**



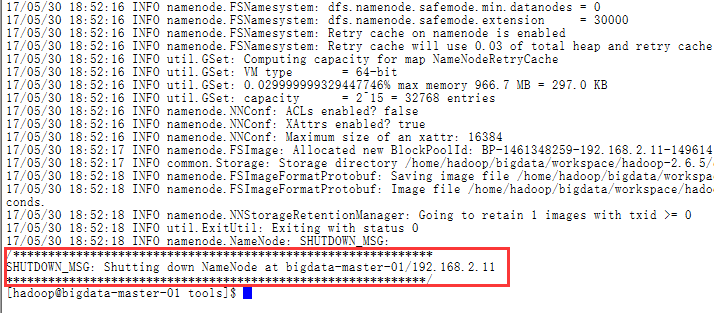
在master-01上执行：

NameNode格式化

hadoop namenode -format

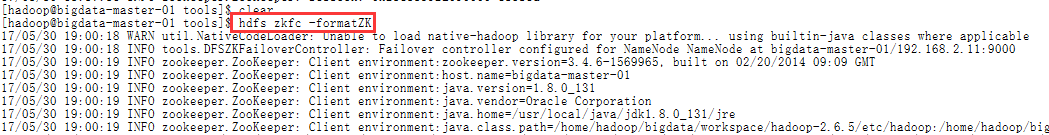


格式化完成

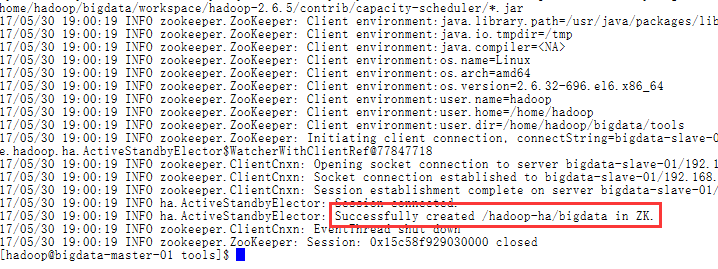


hdfs格式化:

hdfs zkfc -formatZK



格式化完成



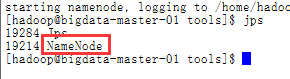
### 3.启动主NameNode节点

在master-01上执行：

hadoop-daemon.sh start namenode



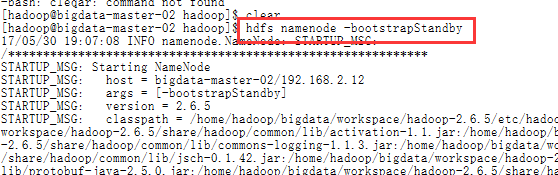
启动成功后master-01会有NameNode进程



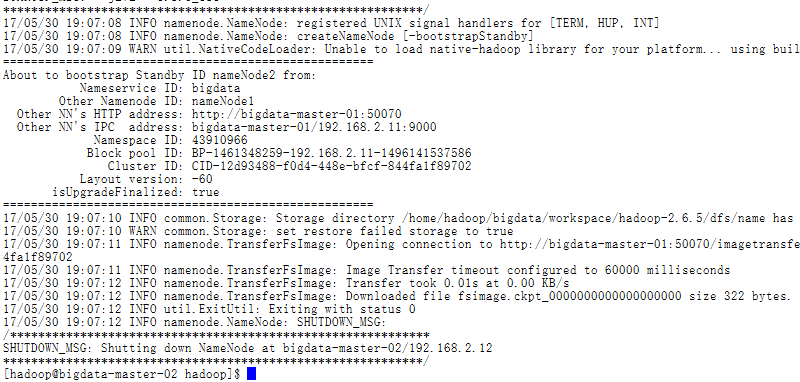
### 4.格式备NameNode节点（namenode2）

在master-02上执行命令：

hdfs namenode –bootstrapStandby



格式化完成



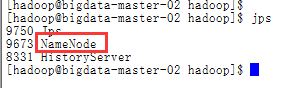
### 5.启动备NameNode节点（namenode2）

在master-02上执行：

hadoop-daemon.sh start namenode

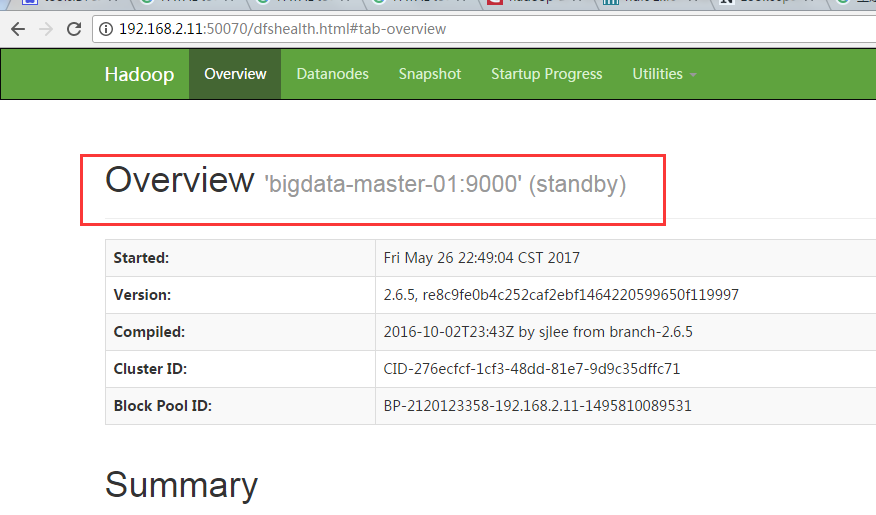


启动成功后master-02会有**NameNode**进程

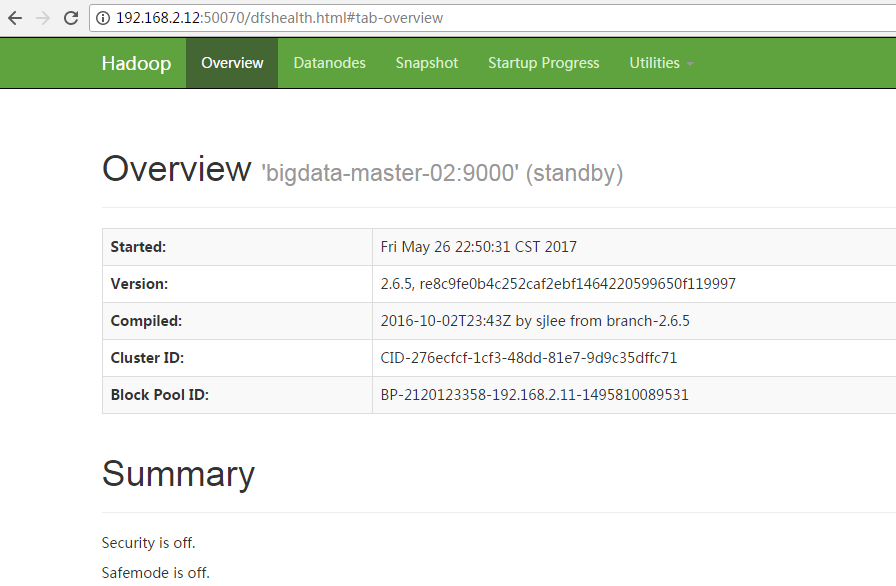


检查UI：

http://192.168.2.11:50070 standby



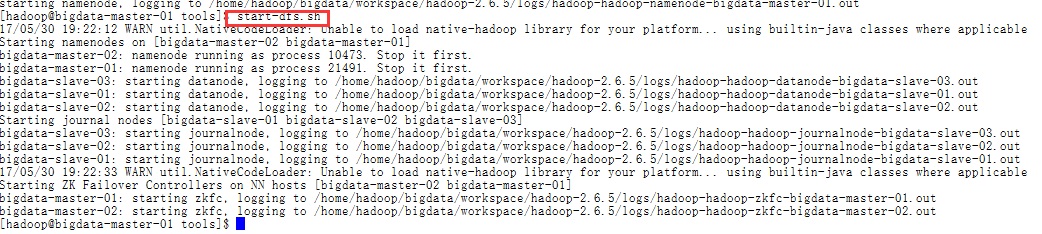
192.168.3.12:50070 standby



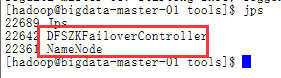
### 6.启动集群

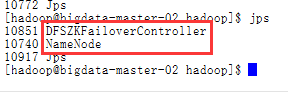
在master-01上执行

start-dfs.sh



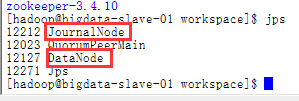
启动成功后master-01，master-02上会有**DFSZKFailoverController**进程

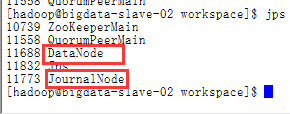


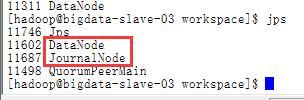


slave-01，slave-02，slave-03会有**JournalNode**进程

slave-01，slave-02，slave-03会有**DataNode**进程

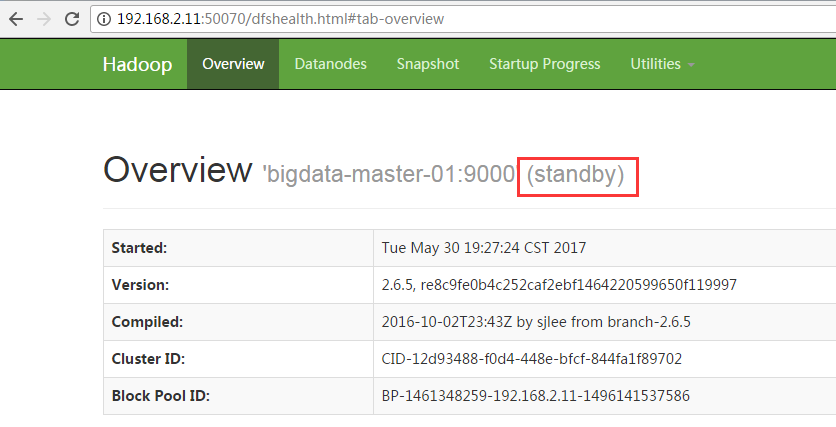




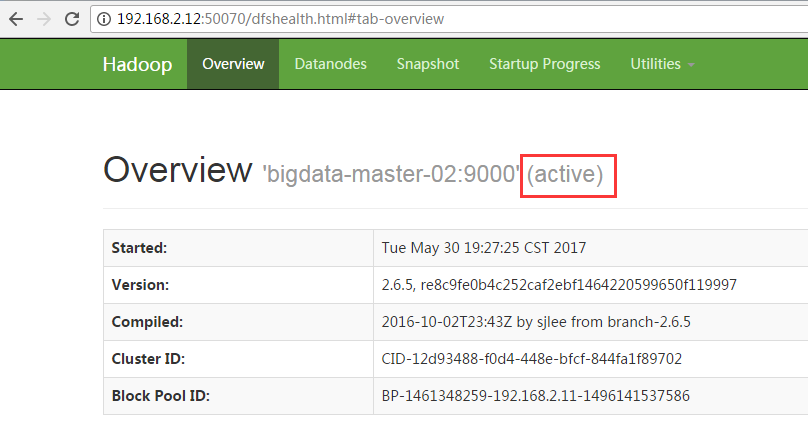


namenode检查：

192.168.3.11:50070 standby



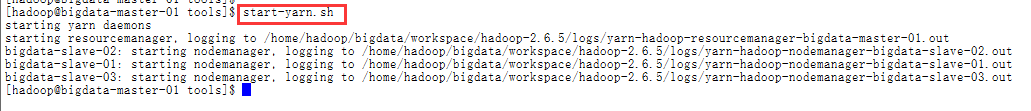
192.168.3.12:50070 active

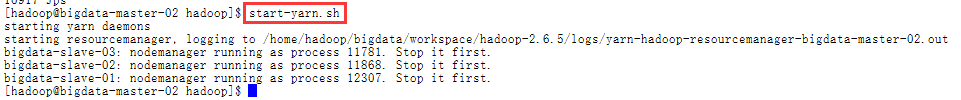


### 7启动YARN（namenode1）

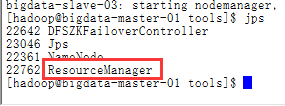
在master-01, master-02上执行：

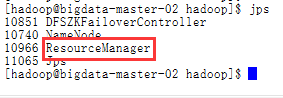
start-yarn.sh



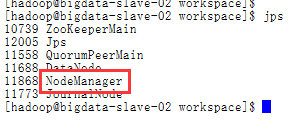


启动成功后master-01, master-02上会有resourcemanager进程



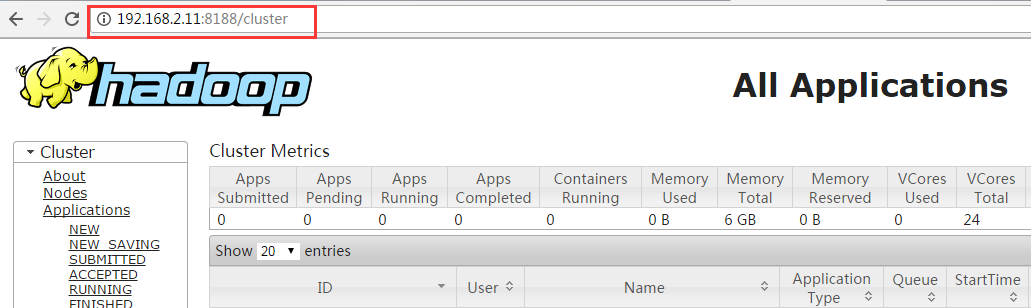


slave-01，slave-02，slave-03上会有nodemanager进程

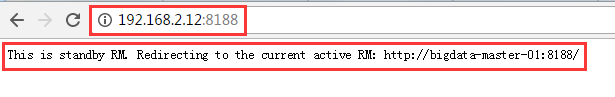


检查：

192.168.2.11:8188 active

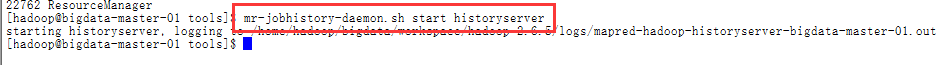


192.168.2.12:8188 standby



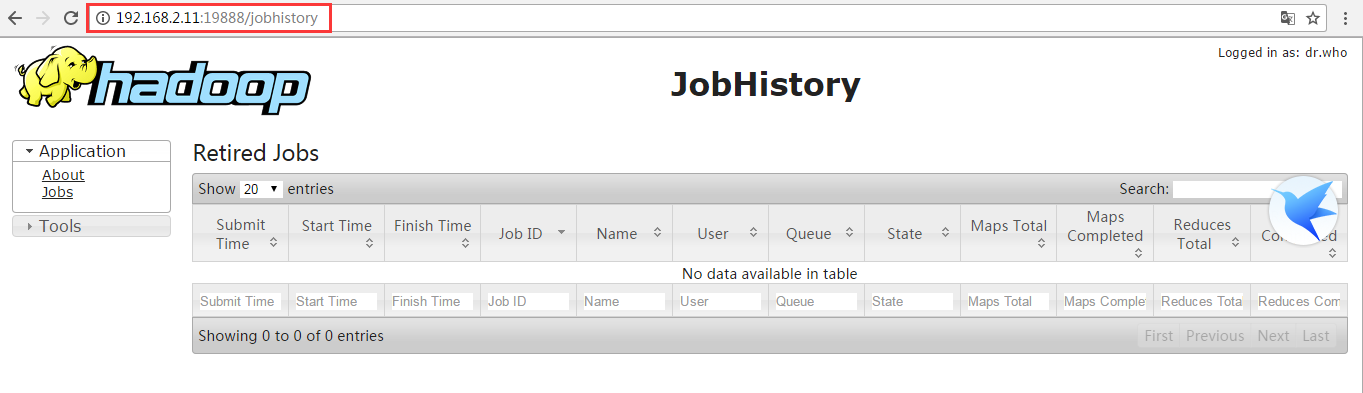
**8.启动hostory server**

mr-jobhistory-daemon.sh start historyserver



执行完后直接登录Job History Server 的 web console，

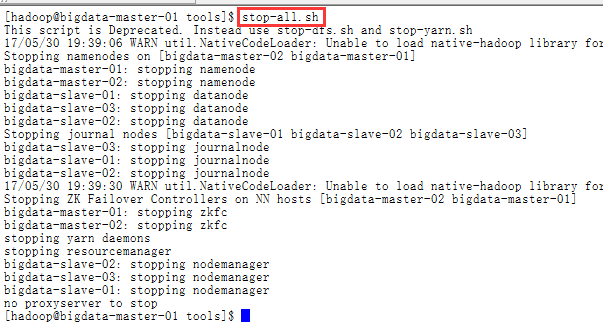
网址为“ <http://hostname:19888/jobhistory>



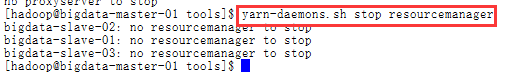
### 9.重启集群

在master-01上执行：

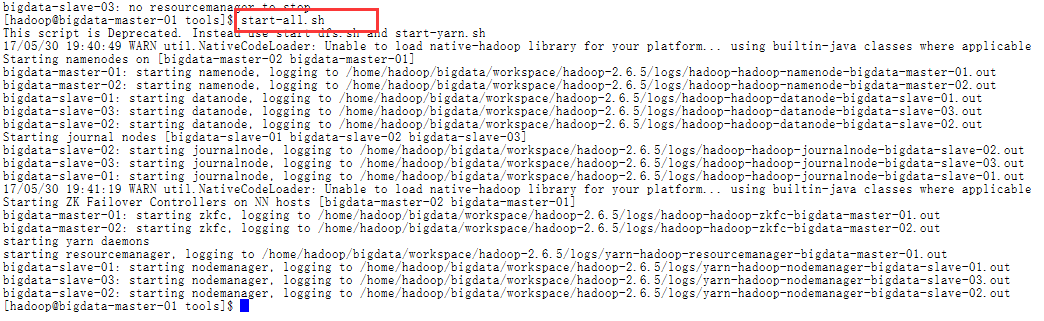
stop-all.sh



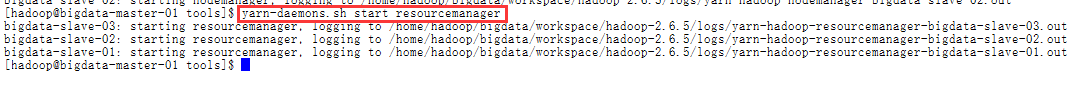
yarn-daemons.sh stop resourcemanager



start-all.sh

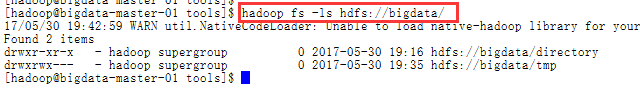


yarn-daemons.sh start resourcemanager



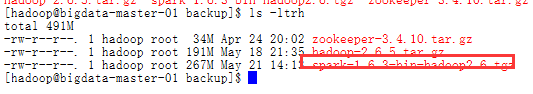
### 10.查看hdfs

hadoop fs -ls hdfs://bigdata/



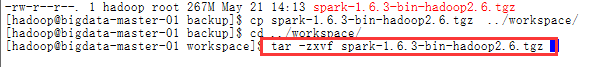
## 九．spark安装与配置

### 1.上传spark安装包



### 2.解压spark包

$SPARK\_HOME=/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6



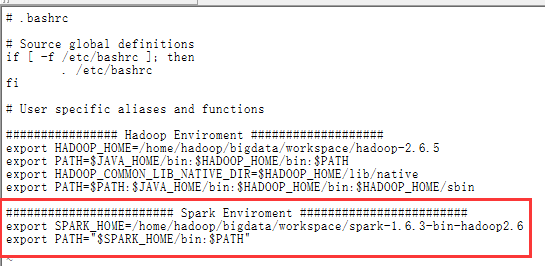
### 3.修改环境变量

######################## Spark Enviroment ########################

export SPARK\_HOME=/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6

export PATH="$SPARK\_HOME/bin:$PATH"

vi ~/.bashrc

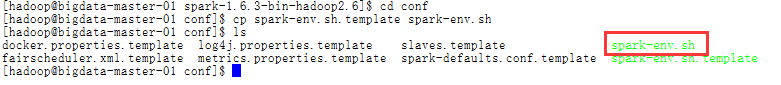


### 4.刷新环境变量

source ~/.bashrc

### 5.配置$SPARK\_HOME/conf/spark-env.sh文件

cd /home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/conf

cp spark-env.sh.template spark-env.sh

#配置内容如下：

export SCALA\_HOME=/usr/local/scala/scala-2.10.6

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_131

export HADOOP\_HOME=/home/hadoop/bigdata/workspace/hadoop-2.6.5

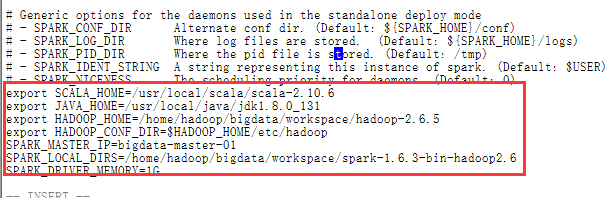
export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

SPARK\_MASTER\_IP=bigdata-master-01

SPARK\_LOCAL\_DIRS=/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6

SPARK\_DRIVER\_MEMORY=1G

vi /home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/conf/spark-env.sh



### 6．配置Spark History Server

cp spark-defaults.conf.template spark-defaults.conf

vi spark-default.conf

spark.eventLog.enabled true

spark.eventLog.compress true

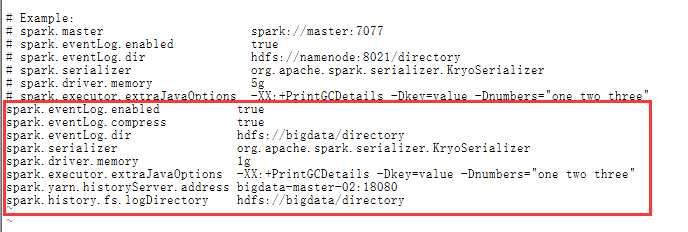
spark.eventLog.dir hdfs://bigdata/directory

spark.serializer org.apache.spark.serializer.KryoSerializer

spark.driver.memory 1g

spark.executor.extraJavaOptions -XX:+PrintGCDetails -Dkey=value -Dnumbers="one two three"

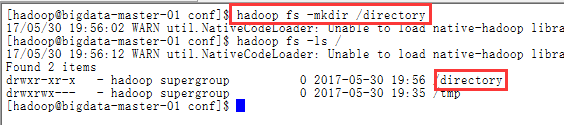
spark.yarn.historyServer.address bigdata-master-02:18080

spark.history.fs.logDirectory hdfs://bigdata/directory

创建directory目录

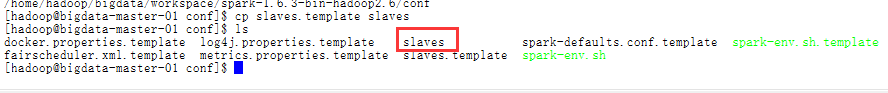
启动hadoop集群后执行以下命令：

hadoop fs –mkdir /directory



### 7.配置$SPARK\_HOME/conf/slaves文件

cp slaves.template slaves

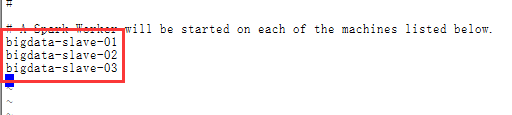


配置内容如下

bigdata-slave-01

bigdata-slave-02

bigdata-slave-03



### 8. 复制spark到各节点：

将配置好的spark文件复制到bigdata-slave-N节点

scp -r /home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6 hadoop@bigdata-master-02:/home/hadoop/bigdata/workspace

scp -r /home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6 hadoop@bigdata-slave-01:/home/hadoop/bigdata/workspace

scp -r /home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6 hadoop@bigdata-slave-02:/home/hadoop/bigdata/workspace

scp -r /home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6 hadoop@bigdata-slave-03:/home/hadoop/bigdata/workspace

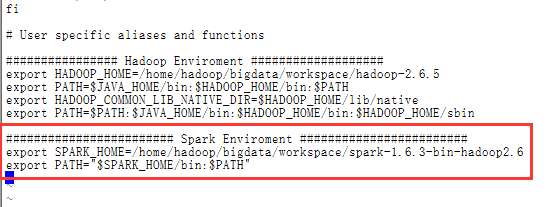
### 9.修改各节点的hadoop用户的spark环境变量

######################## Spark Enviroment ########################

export SPARK\_HOME=/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6

export PATH="$SPARK\_HOME/bin:$PATH"

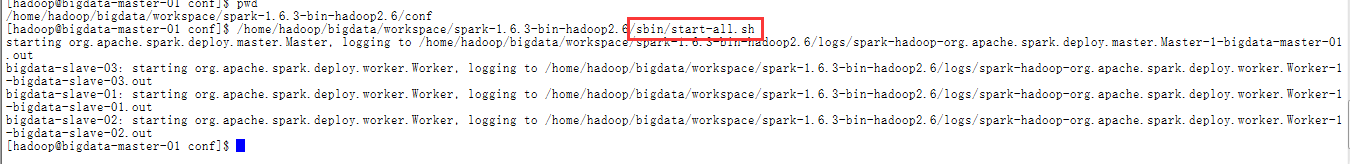
进入环境变量配置文件：vi ~/.bashrc



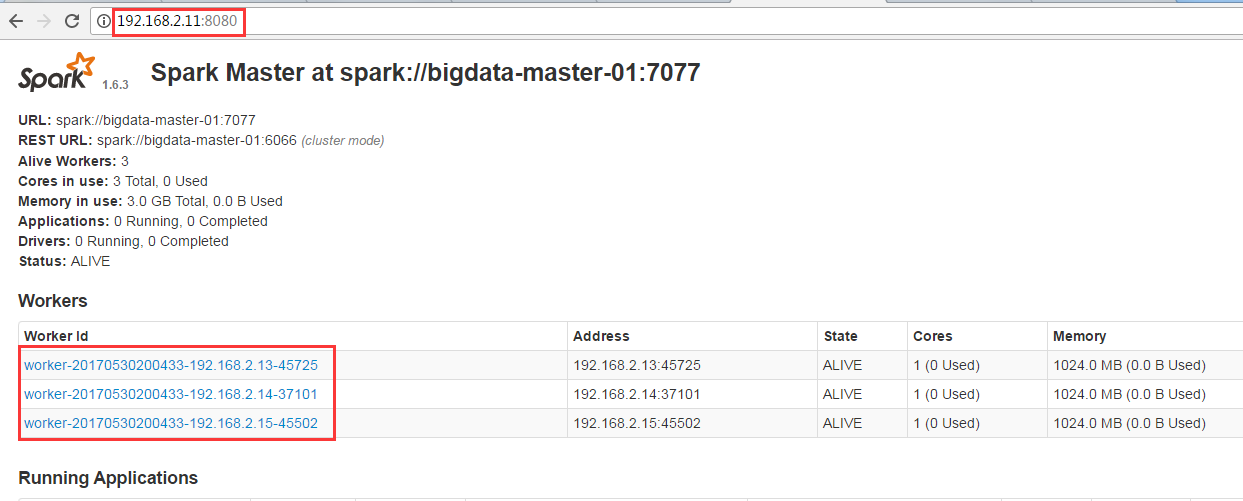
刷新配置：source ~/.bashrc

### 10.启动Spark(standlone,不启动就是yarn)

/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/start-all.sh

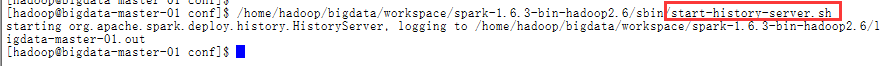


查看http://192.168.2.11:8080/

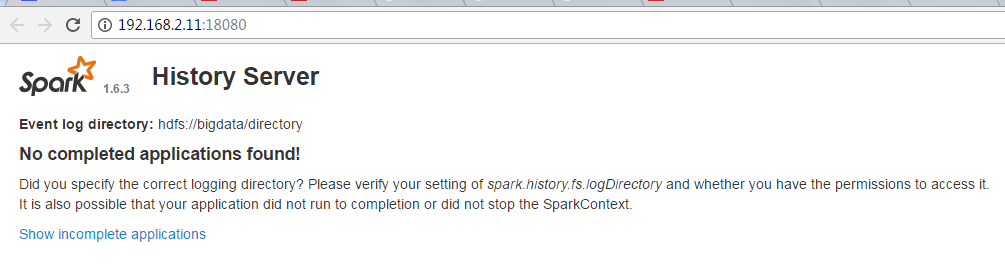


### 11.启动Spark History Server

/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/start-history-server.sh



查看集群：http://192.168.2.12:18080/



## 十．配置spark HA

### 1.说明

默认情况下，Standalone的Spark集群是Master-Slaves架构的集群模式，由一台master来调度资源，这就和大部分的Master-Slaves结构集群一样，存在着Master单点故障的问题。如何解决这个单点故障的问题呢？Spark提供了两种方案：基于文件系统的单点恢复(Single-Node Recovery with Local File system)和基于zookeeper的Standby Masters(Standby Masters with ZooKeeper)。其中ZooKeeper是生产环境下的最佳选择。

ZooKeeper提供了一个Leader Election机制，利用这个机制你可以在集群中开启多个master并使它们都注册到ZooKeeper实例，ZooKeeper会管理使其中只有一个是Active的，其他的都是Standby的，Active状态的master可以提供服务，standby状态的则不可以。ZooKeeper保存了集群的状态信息，该信息包括所有的Worker，Driver 和Application。当Active的Master出现故障时，ZooKeeper会从其他standby的master中选举出一台，然后该新选举出来的master会恢复挂掉了的master的状态信息，之后该Master就可以正常提供调度服务。整个恢复过程只需要1到2分钟。需要注意的是，在这1到2分钟内，只会影响新程序的提交，那些在master崩溃时已经运行在集群中的程序并不会受影响。

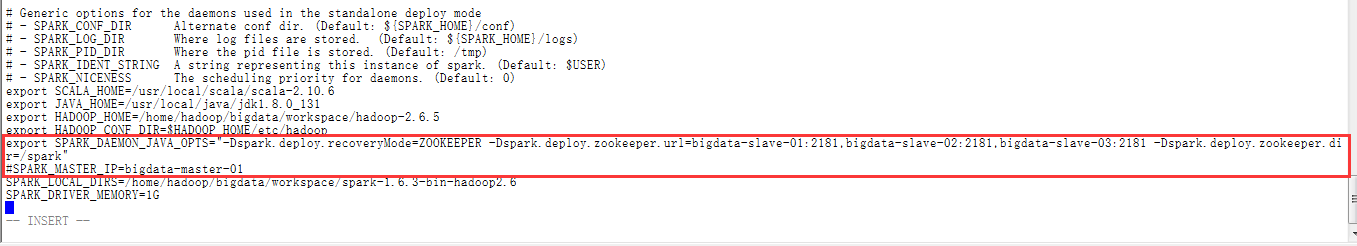
### 2.修改配置文件

vi /home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/conf/spark-env.sh

注释掉export SPARK\_MASTER\_IP=MASTER

添加export SPARK\_DAEMON\_JAVA\_OPTS="-Dspark.deploy.recoveryMode=ZOOKEEPER -Dspark.deploy.zookeeper.url=bigdata-slave-01:2181,bigdata-slave-02:2181,bigdata-slave-03:2181 -Dspark.deploy.zookeeper.dir=/spark"

注意：所有准备用来做master的节点都需要做这个修改。



### 3.将配置复制到其它节点上

scp /home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/conf/spark-env.sh hadoop@bigdata-master-01:/home/hadoop/bigdata/workspace/spark-1.6.3-bin-ha

doop2.6/conf/

scp /home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/conf/spark-env.sh hadoop@bigdata-master-02:/home/hadoop/bigdata/workspace/spark-1.6.3-bin-ha

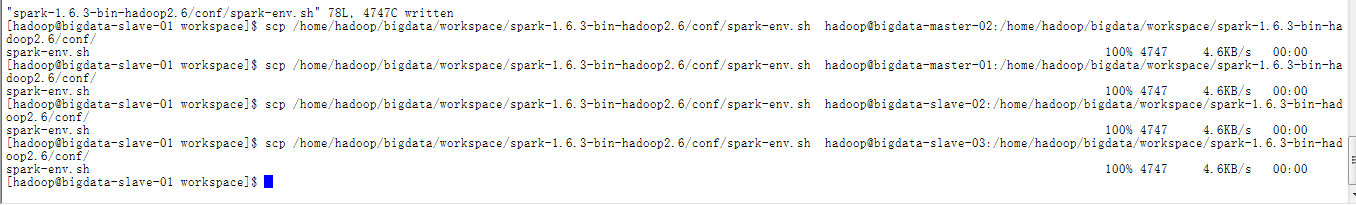
doop2.6/conf/

scp /home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/conf/spark-env.sh hadoop@bigdata-slave-02:/home/hadoop/bigdata/workspace/spark-1.6.3-bin-ha

doop2.6/conf/

scp /home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/conf/spark-env.sh hadoop@bigdata-slave-03:/home/hadoop/bigdata/workspace/spark-1.6.3-bin-ha

doop2.6/conf/



### 4.启动测试

先启动zookeeper与hadoop,再启动spark

在master-01上启动spark

/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/start-all.sh

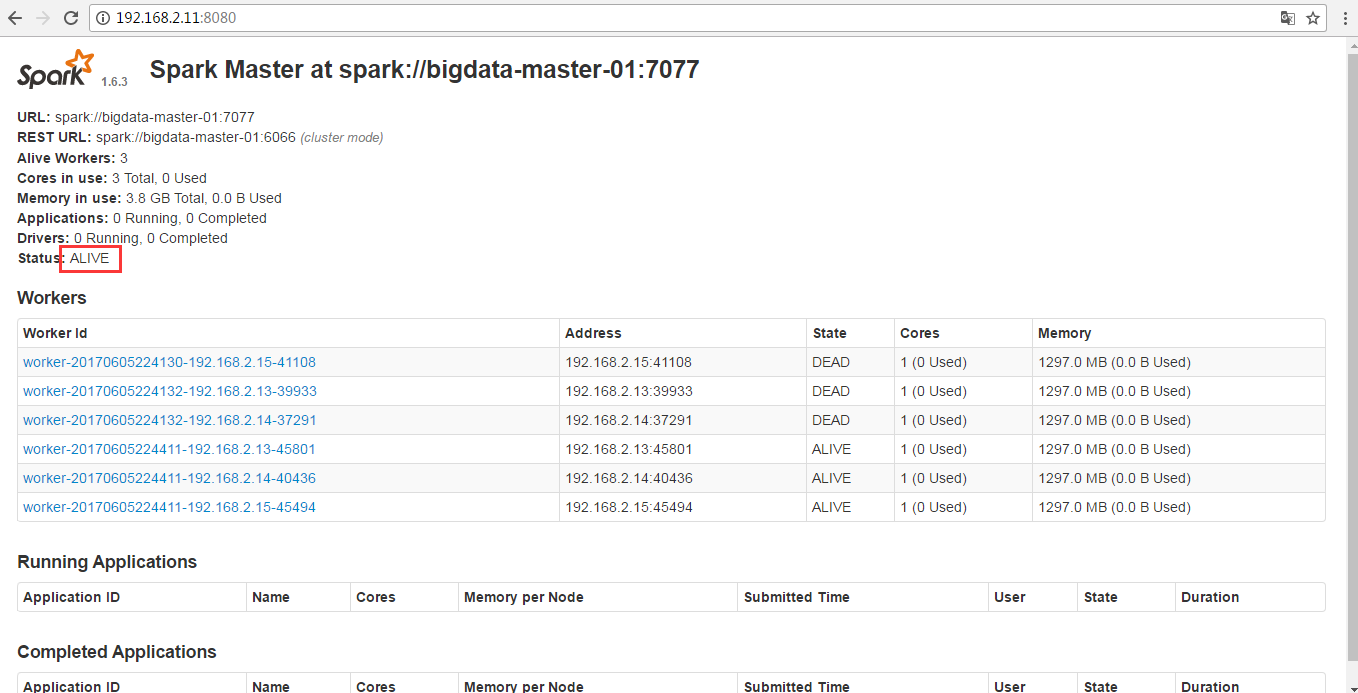
在master-02上启动spark

/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/start-all.sh

5.验证spark HA

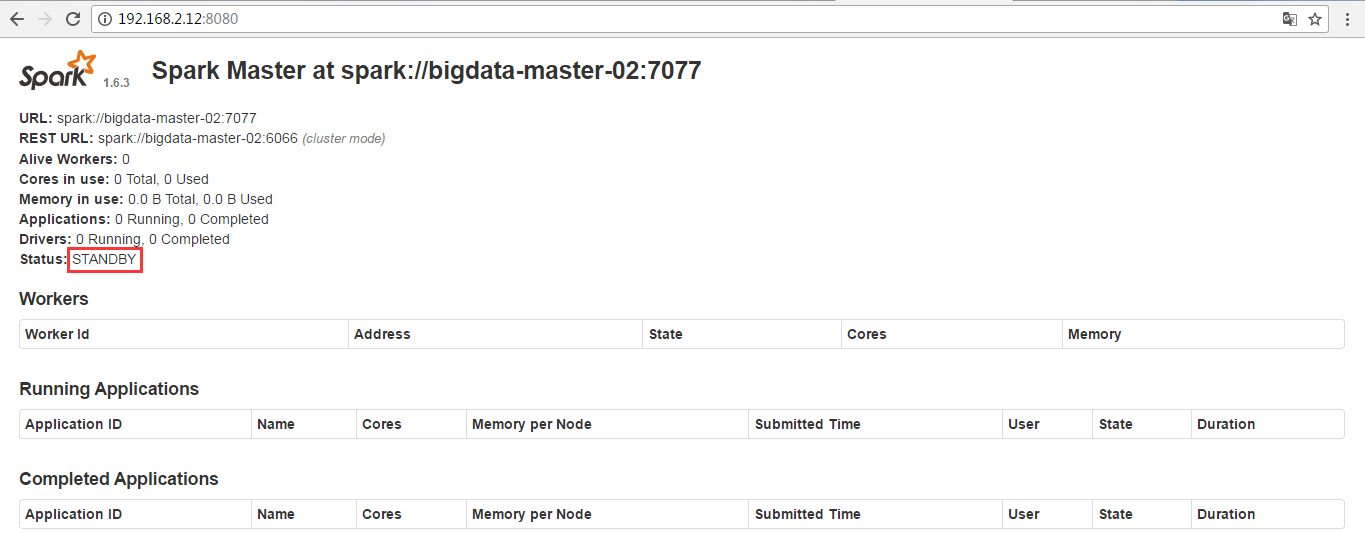
Master-01

<http://192.168.2.11:8080/>

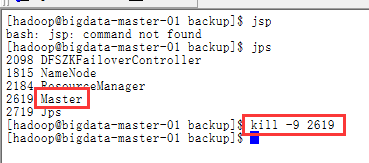


Master-02

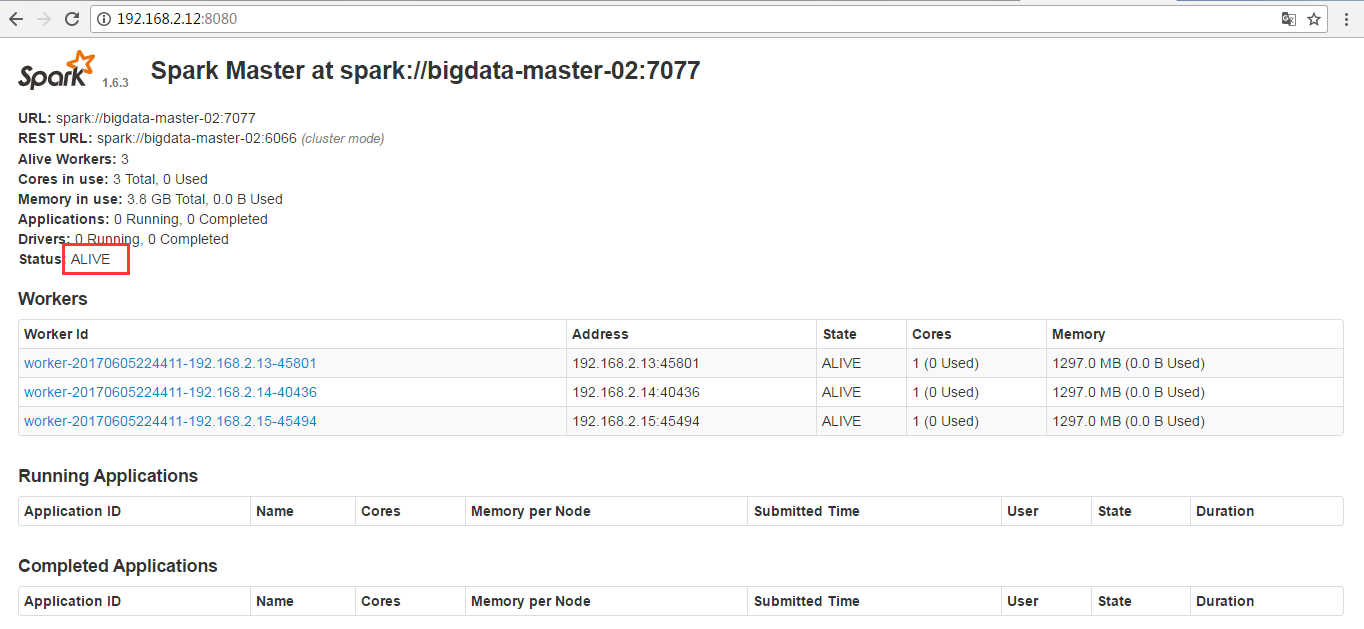
<http://192.168.2.12:8080/>



Kill掉master-01



Master-02变成了active



提交job

spark-submit --class org.apache.spark.examples.SparkPi --master spark://bigdata-master-01:7077,bigdata-master-02:7077 /home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/lib/spark-examples-1.6.3-hadoop2.6.0.jar

## 十一．测试集群

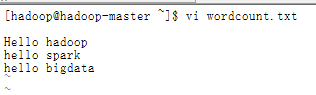
### 1.测试hadoop

测试的源文件的内容为:

Hello hadoop

hello spark

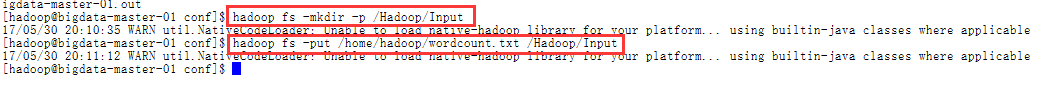
hello bigdata



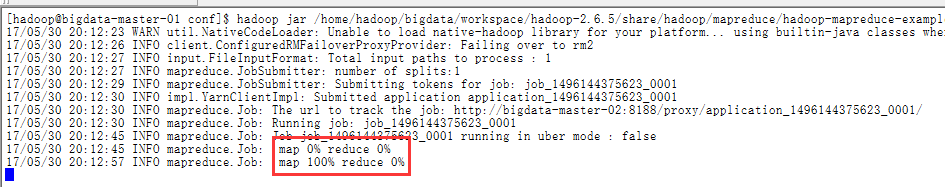
然后执行下列命令：

hadoop fs -mkdir -p /Hadoop/Input

hadoop fs -put /home/hadoop/wordcount.txt /Hadoop/Input

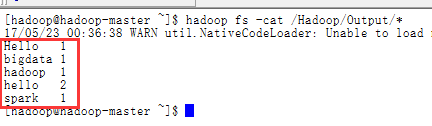


hadoop jar /home/hadoop/bigdata/workspace/hadoop-2.6.5/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.5.jar wordcount /Hadoop/Input /Hadoop/Output

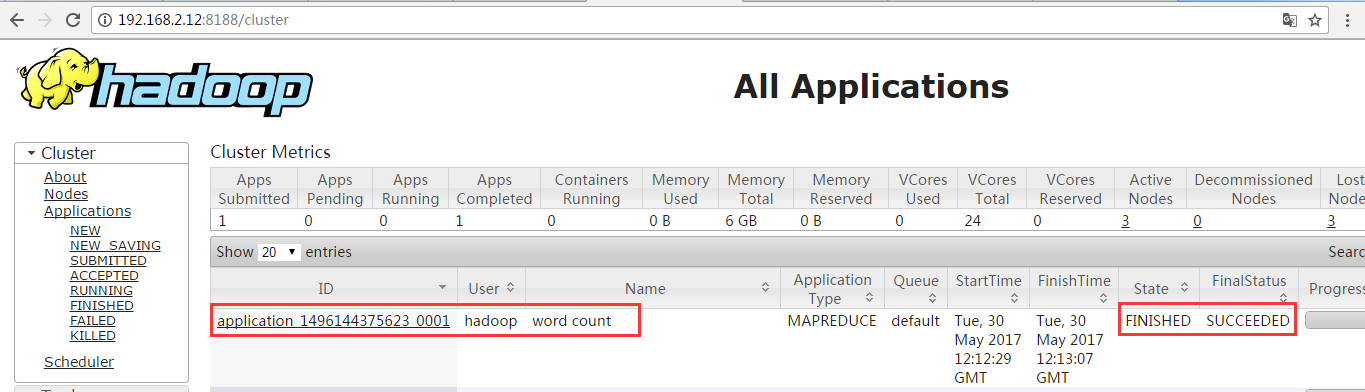


查看hadoop计算结果：

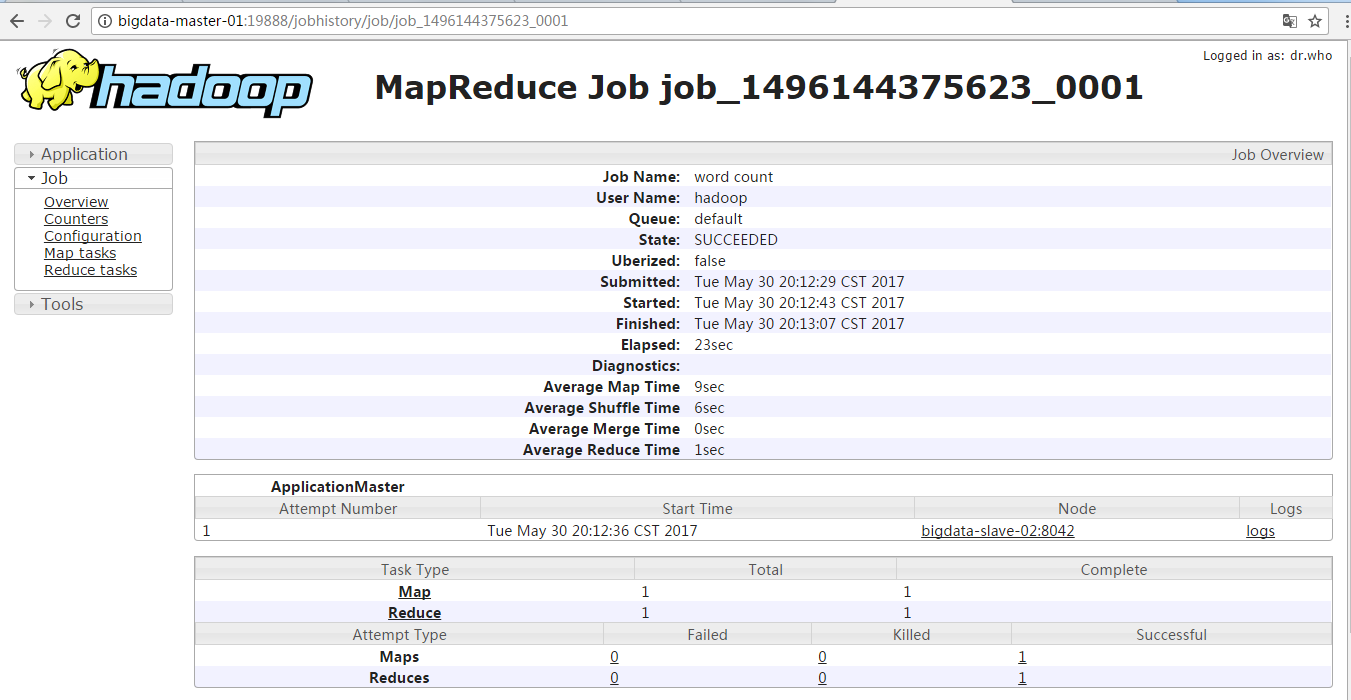
hadoop fs -cat /Hadoop/Output/\*

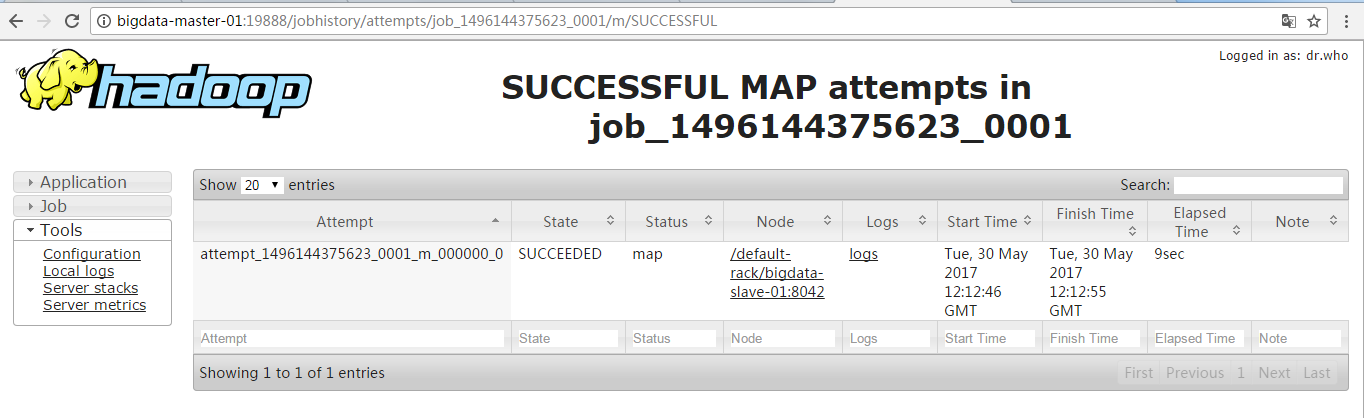


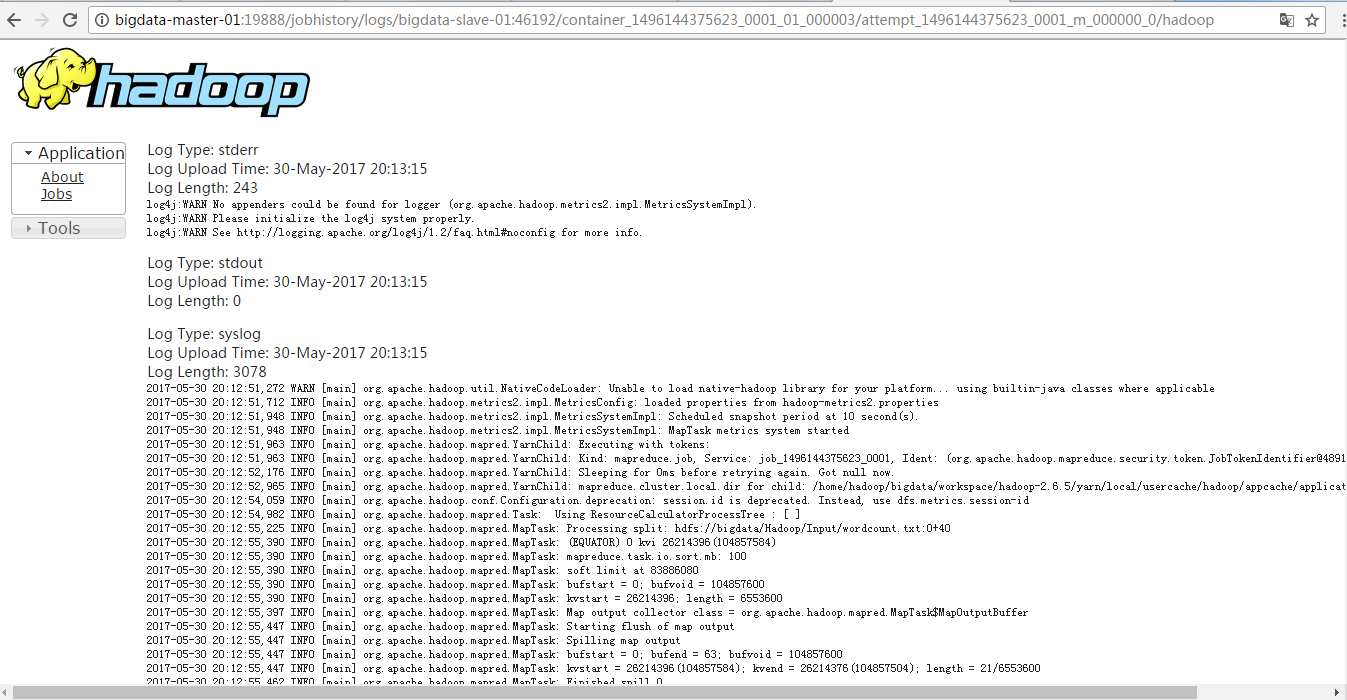
查看UI



查看log记录







hadoop集群搭建成功！

### 2.测试spark

为了避免麻烦这里我们使用spark-shell，做一个简单的worcount的测试

用于在测试hadoop的时候我们已经在hdfs上存储了测试的源文件，下面就是直接拿来用就好了！

spark-shell

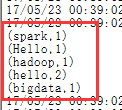
val file=sc.textFile("hdfs://bigdata/Hadoop/Input/wordcount.txt")

val rdd = file.flatMap(line => line.split(" ")).map(word => (word,1)).reduceByKey(\_+\_)

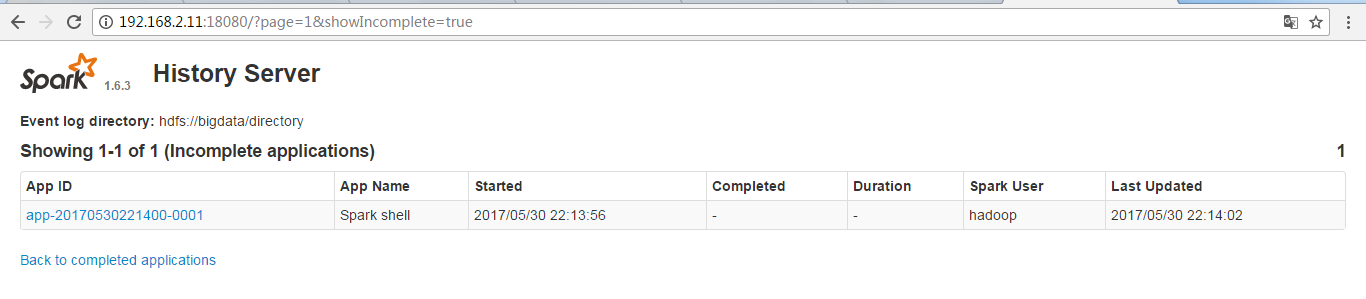
rdd.collect()

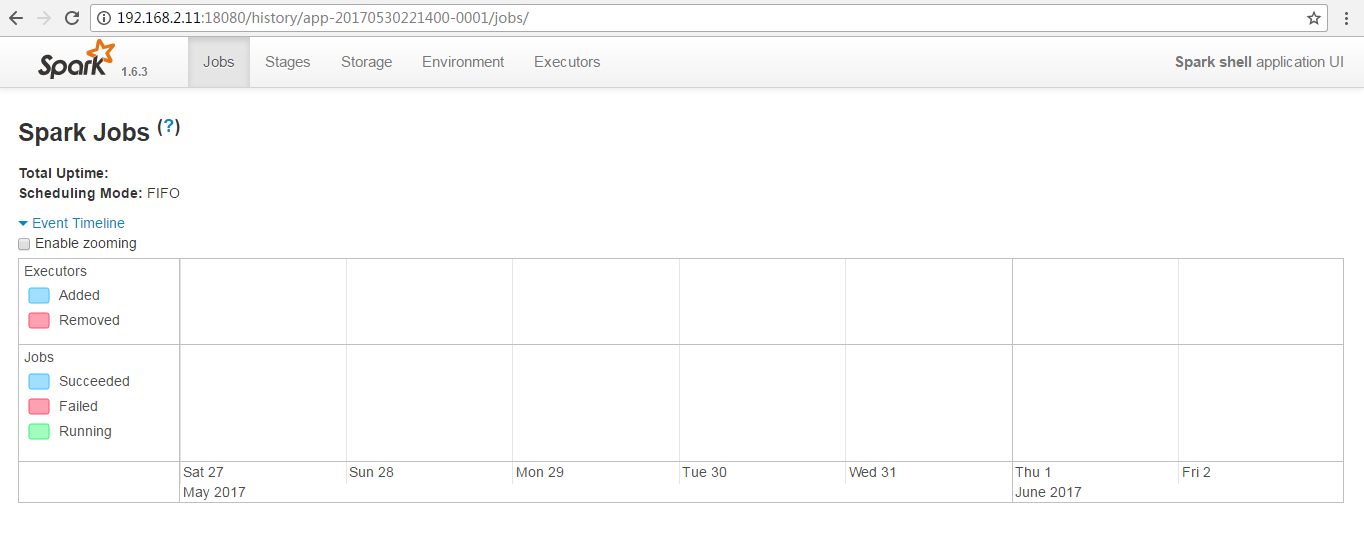
rdd.foreach(println)





查看spark job日志





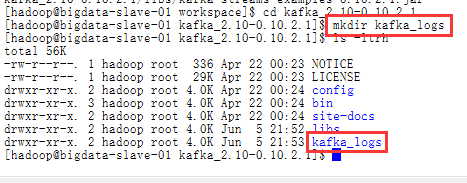
## 十二．Kafka安装

### 1.上传kafka安装包并解压



### 2.创建kafka\_logs目录

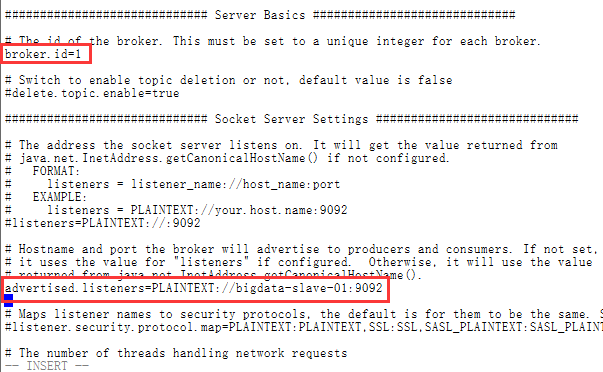
mkdir kafka\_logs



### 3.修改kafka配置文件

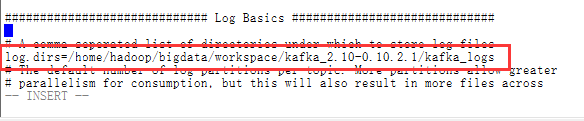
a)配置节点与端口

vi /home/hadoop/bigdata/workspace/kafka\_2.10-0.10.2.1/config/server.properties



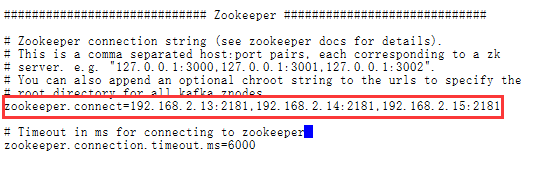
b）配置log目录

/home/hadoop/bigdata/workspace/kafka\_2.10-0.10.2.1/kafka\_logs



c)配置zookeeper连接

zookeeper.connect=192.168.2.13:2181,192.168.2.14:2181,192.168.2.15:2181



### 4.配置kafka环境变量

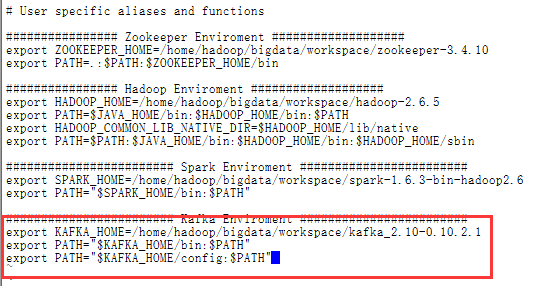
######################## Kafka Enviroment ########################

export KAFKA\_HOME=/home/hadoop/bigdata/workspace/kafka\_2.10-0.10.2.1

export PATH="$KAFKA\_HOME/bin:$PATH"

export PATH="$KAFKA\_HOME/config:$PATH"

vi ~/.bashrc



source ~/.bashrc

### 5.复制到slave-02与slave-03节点上

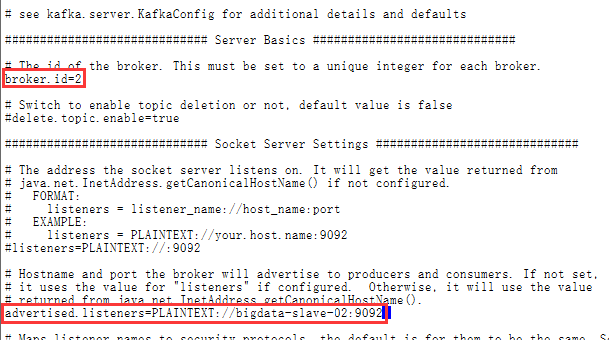
scp -r /home/hadoop/bigdata/workspace/kafka\_2.10-0.10.2.1 hadoop@bigdata-slave-02:/home/hadoop/bigdata/workspace

scp -r /home/hadoop/bigdata/workspace/kafka\_2.10-0.10.2.1 hadoop@bigdata-slave-03:/home/hadoop/bigdata/workspace

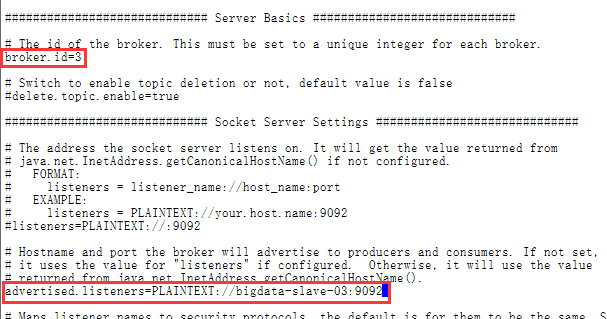
### 6.修改slave-02与slave-03的配置文件

vi /home/hadoop/bigdata/workspace/kafka\_2.10-0.10.2.1/config/server.properties

slave-02



slave-03



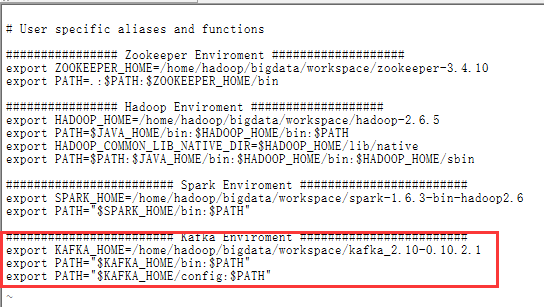
### 7.分别配置节点slave-02与slave-03的环境变量

######################## Kafka Enviroment ########################

export KAFKA\_HOME=/home/hadoop/bigdata/workspace/kafka\_2.10-0.10.2.1

export PATH="$KAFKA\_HOME/bin:$PATH"

export PATH="$KAFKA\_HOME/config:$PATH"



### 8.测试kafka

1. 启动各节点kafka(启动前先启动zookeeper)

nohup kafka-server-start.sh /home/hadoop/bigdata/workspace/kafka\_2.10-0.10.2.1/config/server.properties &

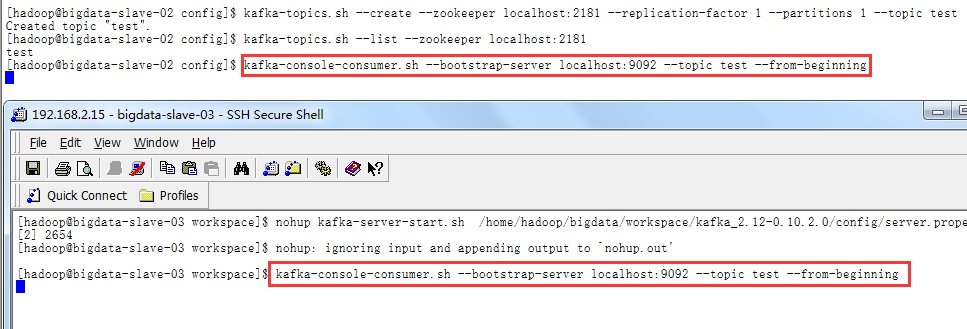
1. 创建topic

kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test



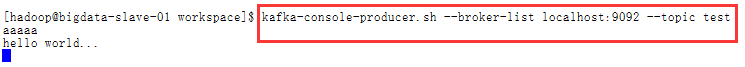
1. 在slave-02与slave-03上监听消息

kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning

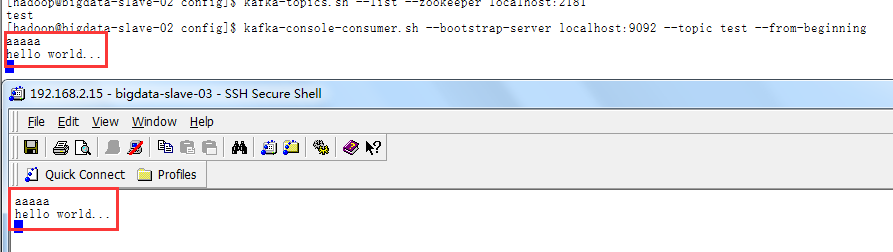


1. 发消息

kafka-console-producer.sh --broker-list localhost:9092 --topic test



1. 查看接收消息



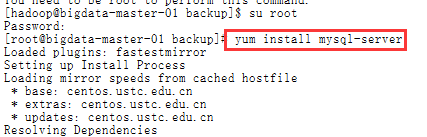
## 十三.hive安装

### 1.上传并解压hive

### 2.安装mysql

a)下载mysql

yum install mysql-server



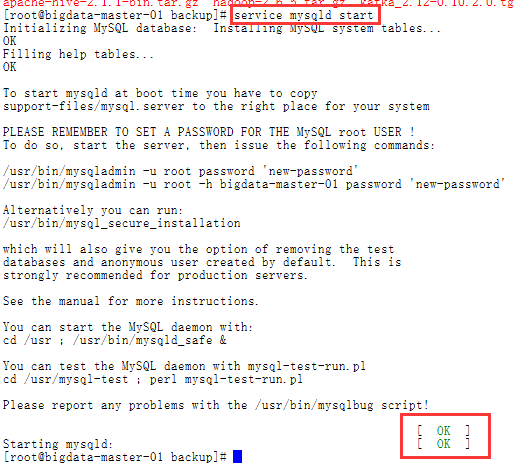
b)配置开机启动

chkconfig --add mysqld



c)启动mysql

service mysqld start



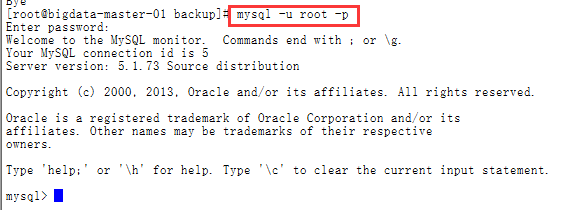
d)设置密码

mysqladmin -u root password "root"



e)连接数据库

mysql -u root -p



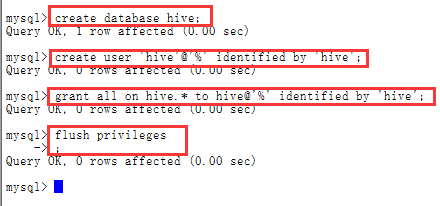
1. 建立[数据库](http://lib.csdn.net/base/mysql)[Hive](http://lib.csdn.net/base/hive)

create database hive;



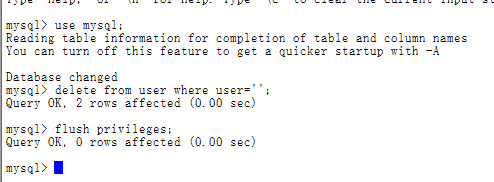
g)创建用户hive   
create user 'hive'@'%' identified by 'hive';  
h)创建hive用户,并授权   
grant all on hive.\* to hive@'%' identified by 'hive';

flush privileges;

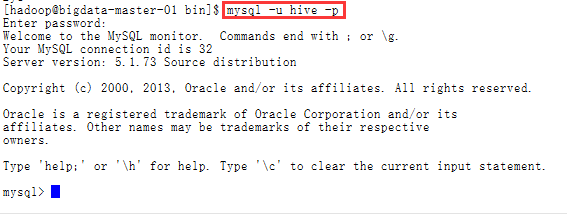


i)解决新用户不能登录问题

增加普通用户后，执行：   
mysql> use mysql   
mysql> delete from user where user='';   
mysql> flush privileges;



j)使用新用户登录



k)设置开机启动

添加执行权限

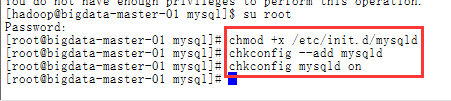
chmod +x /etc/init.d/mysqld

添加到服务

chkconfig --add mysqld

设置开机启动

chkconfig mysqld on



### 3.解压hive



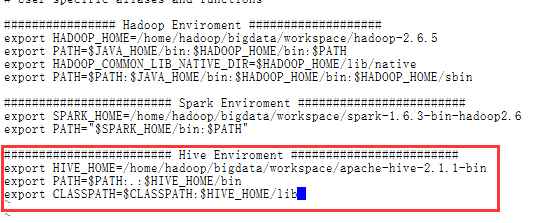
### 4.配置hive环境变量

export HIVE\_HOME=/home/hadoop/bigdata/workspace/apache-hive-2.1.1-bin

export PATH=$PATH:.:$HIVE\_HOME/bin

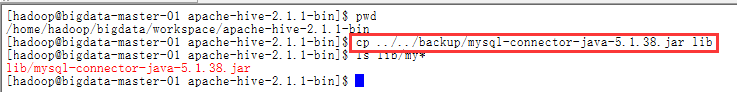
export CLASSPATH=$CLASSPATH:$HIVE\_HOME/lib

vi ~/.bashrc



source ~/.bashrc

### 5.上传mysql jdbc并copy到hive lib目录中



### 6.配置hive-site.xml

cp hive-default.xml.template hive-site.xml

vi /home/hadoop/bigdata/workspace/apache-hive-2.1.1-bin/conf/hive-site.xml

修改以下配置属性

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://localhost:3306/hive?characterEncoding=UTF-8</value>

<description>JDBC connect string for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

<description>Driver class name for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>hive</value>

<description>username to use against metastore database</description>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>hive</value>

<description>password to use against metastore database</description>

</property>

<property>

<name>hive.metastore.schema.verification</name>

<value>false</value>

<description>

Enforce metastore schema version consistency.

True: Verify that version information stored in is compatible with one from Hive jars. Also disable automatic

schema migration attempt. Users are required to manually migrate schema after Hive upgrade which ensures

proper metastore schema migration. (Default)

False: Warn if the version information stored in metastore doesn't match with one from in Hive jars.

</description>

<property>

<name>hive.exec.local.scratchdir</name>

<value>/home/hadoop/bigdata/workspace/apache-hive-2.1.1-bin/tmp</value>

<description>Local scratch space for Hive jobs</description>

</property>

<property>

<name>hive.downloaded.resources.dir</name>

<value>/home/hadoop/bigdata/workspace/apache-hive-2.1.1-bin/tmp</value>

<description>Temporary local directory for added resources in the remote file system.</description>

</property>

<property>

<name>hive.querylog.location</name>

<value>/home/hadoop/bigdata/workspace/apache-hive-2.1.1-bin/tmp</value>

<description>Location of Hive run time structured log file</description>

</property>

<property>

<name>hive.server2.logging.operation.log.location</name>

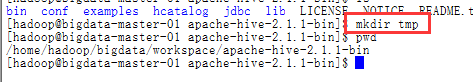
<value>/home/hadoop/bigdata/workspace/apache-hive-2.1.1-bin/tmp/operation\_logs</value>

<description>Top level directory where operation logs are stored if logging functionality is enabled</description>

</property>

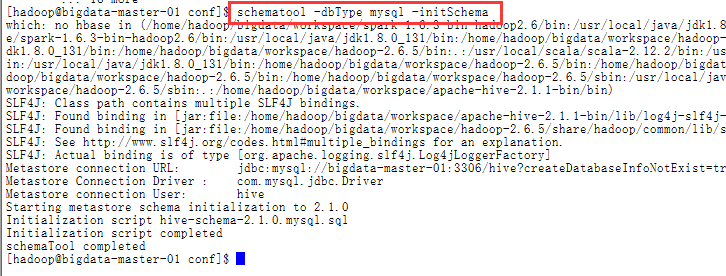
### 7.创建目录

mkdir /home/hadoop/bigdata/workspace/apache-hive-2.1.1-bin/tmp

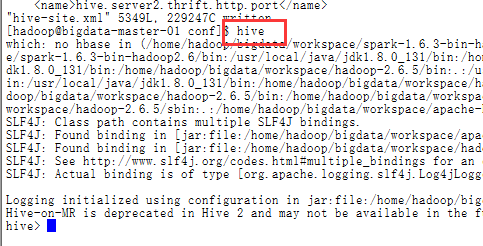


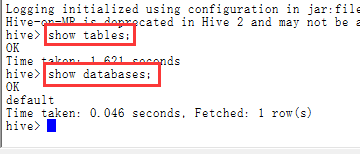
### 8.初始化数据库

schematool -dbType mysql –initSchema



### 9.启动hive





### 10.查看

## 十四.sqoop安装

### 1.上传并解压



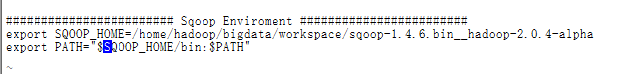
### 2.配置环境变量

vi ~/.bashrc

######################## Sqoop Enviroment ########################

export SQOOP\_HOME=/home/hadoop/bigdata/workspace/sqoop-1.4.6.bin\_\_hadoop-2.0.4-alpha

export PATH="$SQOOP\_HOME/bin:$PATH"



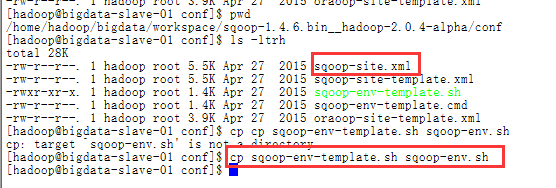
source ~/.bashrc

### 3.配置sqoop(conf)

先拷贝：

cp sqoop-site.xml.template sqoop-site.xml

cp sqoop-env.sh.template sqoop-env.sh



sqoop-site.xml不变

sqoop-env.sh修改如下：

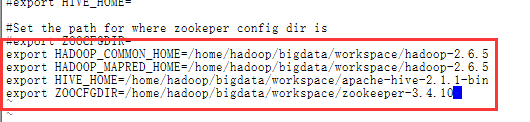
export HADOOP\_COMMON\_HOME=/home/hadoop/bigdata/workspace/hadoop-2.6.5

export HADOOP\_MAPRED\_HOME=/home/hadoop/bigdata/workspace/hadoop-2.6.5

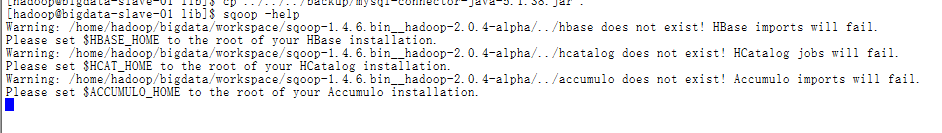
export HIVE\_HOME=/home/hadoop/bigdata/workspace/apache-hive-2.1.1-bin

#如果zookeeper是单独安装配置的则还要修改下面的内容

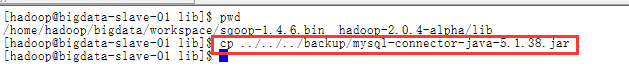
export ZOOCFGDIR=/home/hadoop/bigdata/workspace/zookeeper-3.4.10



### 4.测试sqoop

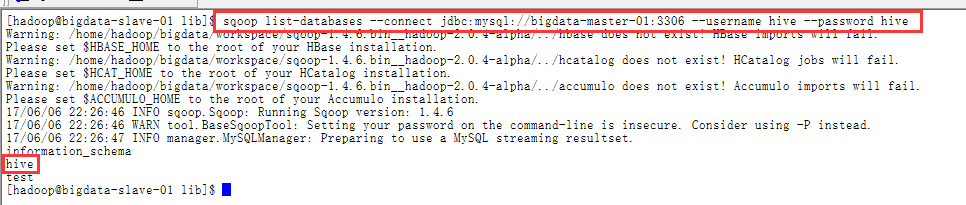


### 5.拷贝mysql jdbc到lib目录下



### 6.测试

sqoop list-databases --connect jdbc:mysql://bigdata-master-01:3306 --username hive --password hive

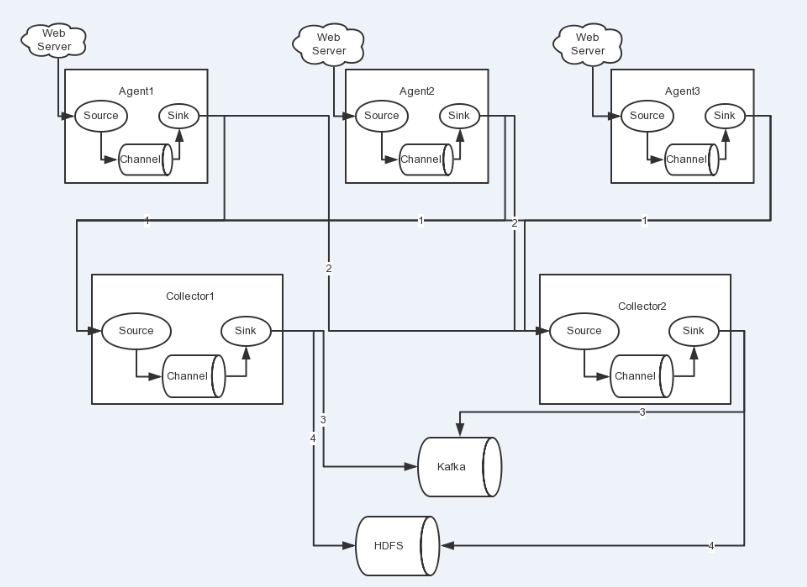


## 十五．flume集群搭建

### 环境配置

|  |  |
| --- | --- |
| 名称 | 版本 |
| Flume | 1.8 |
| Zookeeper | 3.4.7 |
| Kafka | kafka\_2.11-0.11.0.1 |

### 架构图

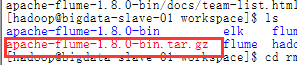


### 集群角色分布

|  |  |  |
| --- | --- | --- |
| 名称 | 节点 | 角色 |
| Agent1 | bigdata-slave-01 | Web Server |
| Agent2 | bigdata-slave-02 | Web Server |
| Agent3 | bigdata-slave-03 | Web Server |
| Collector1 | bigdata-master-01 | AgentMstr1 |
| Collector2 | bigdata-master-02 | AgentMstr2 |

### Flume安装

1. 解压



### Flume配置

进入配置文件目录(conf)建立以下文件：

* 1）flume-client.properties

#agent1 name

agent1.channels = c1

agent1.sources = r1

agent1.sinks = k1 k2

#set gruop

agent1.sinkgroups = g1

#set channel

agent1.channels.c1.type = memory

agent1.channels.c1.capacity = 1000

agent1.channels.c1.transactionCapacity = 100

agent1.sources.r1.channels = c1

agent1.sources.r1.type = spooldir

agent1.sources.r1.spoolDir = /home/hadoop/bigdata/data/logs

agent1.sources.r1.fileHeader = true

agent1.sources.r1.interceptors = i1 i2

agent1.sources.r1.interceptors.i1.type = static

agent1.sources.r1.interceptors.i1.key = Type

agent1.sources.r1.interceptors.i1.value = LOGIN

agent1.sources.r1.interceptors.i2.type = timestamp

# set sink1

agent1.sinks.k1.channel = c1

agent1.sinks.k1.type = avro

agent1.sinks.k1.hostname = bigdata-master-01

agent1.sinks.k1.port = 52020

# set sink2

agent1.sinks.k2.channel = c1

agent1.sinks.k2.type = avro

agent1.sinks.k2.hostname = bigdata-master-02

agent1.sinks.k2.port = 52020

#set sink group

agent1.sinkgroups.g1.sinks = k1 k2

#set failover

agent1.sinkgroups.g1.processor.type = failover

agent1.sinkgroups.g1.processor.priority.k1 = 10

agent1.sinkgroups.g1.processor.priority.k2 = 1

agent1.sinkgroups.g1.processor.maxpenalty = 10000

* 2）flume-server.properties

#set Agent name

a1.sources = r1

a1.channels = c1

a1.sinks = k1

#set channel

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

# other node,nna to nns

a1.sources.r1.type = avro

a1.sources.r1.bind = bigdata-master-01

a1.sources.r1.port = 52020

a1.sources.r1.interceptors = i1

a1.sources.r1.interceptors.i1.type = static

a1.sources.r1.interceptors.i1.key = Collector

a1.sources.r1.interceptors.i1.value = NNA

a1.sources.r1.channels = c1

#set sink to kafka

a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink

a1.sinks.k1.topic = test

a1.sinks.k1.brokerList = bigdata-slave-01:9092,bigdata-slave-02:9092,bigdata-slave-03:9092

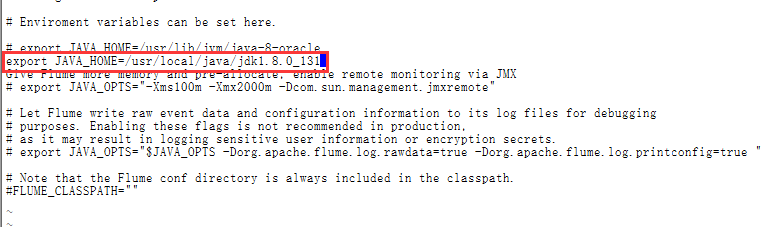
a1.sinks.k1.requiredAcks = 1

a1.sinks.k1.batchSize = 2

a1.sinks.k1.channel=c1

* 3) flume-env.sh

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_131



* 4)传送到各节点

scp -r apache-flume-1.8.0-bin hadoop@bigdata-master-01:/home/hadoop/bigdata/workspace/

scp -r apache-flume-1.8.0-bin hadoop@bigdata-master-02:/home/hadoop/bigdata/workspace/

scp -r apache-flume-1.8.0-bin hadoop@bigdata-slave-02:/home/hadoop/bigdata/workspace/

scp -r apache-flume-1.8.0-bin hadoop@bigdata-slave-03:/home/hadoop/bigdata/workspace/

* 5)修改master-02上的flume-server.properties

将主机名字改成：bigdata-master-02

#set Agent name

a1.sources = r1

a1.channels = c1

a1.sinks = k1

#set channel

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

# other node,nna to nns

a1.sources.r1.type = avro

a1.sources.r1.bind = bigdata-master-02

a1.sources.r1.port = 52020

a1.sources.r1.interceptors = i1

a1.sources.r1.interceptors.i1.type = static

a1.sources.r1.interceptors.i1.key = Collector

a1.sources.r1.interceptors.i1.value = NNA

a1.sources.r1.channels = c1

#set sink to kafka

a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink

a1.sinks.k1.topic = test

a1.sinks.k1.brokerList = bigdata-slave-01:9092,bigdata-slave-02:9092,bigdata-slave-03:9092

a1.sinks.k1.requiredAcks = 1

a1.sinks.k1.batchSize = 2

a1.sinks.k1.channel=c1

### 启动Flume

在Agent节点上启动命令如下所示：

/home/hadoop/bigdata/workspace/apache-flume-1.8.0-bin/bin/flume-ng agent -n agent1 -c conf -f /home/hadoop/bigdata/workspace/apache-flume-1.8.0-bin/conf/flume-client.properties -Dflume.root.logger=DEBUG,console

注：命令中的agent1表示配置文件中的Agent的Name，如配置文件中的agent1。flume-client.properties表示配置文件所在配置，需填写准确的配置文件路径。

在Collector节点上启动命令如下所示：

/home/hadoop/bigdata/workspace/apache-flume-1.8.0-bin/bin/flume-ng agent -n a1 -c conf -f /home/hadoop/bigdata/workspace/apache-flume-1.8.0-bin/conf/flume-server.properties -Dflume.root.logger=DEBUG,console

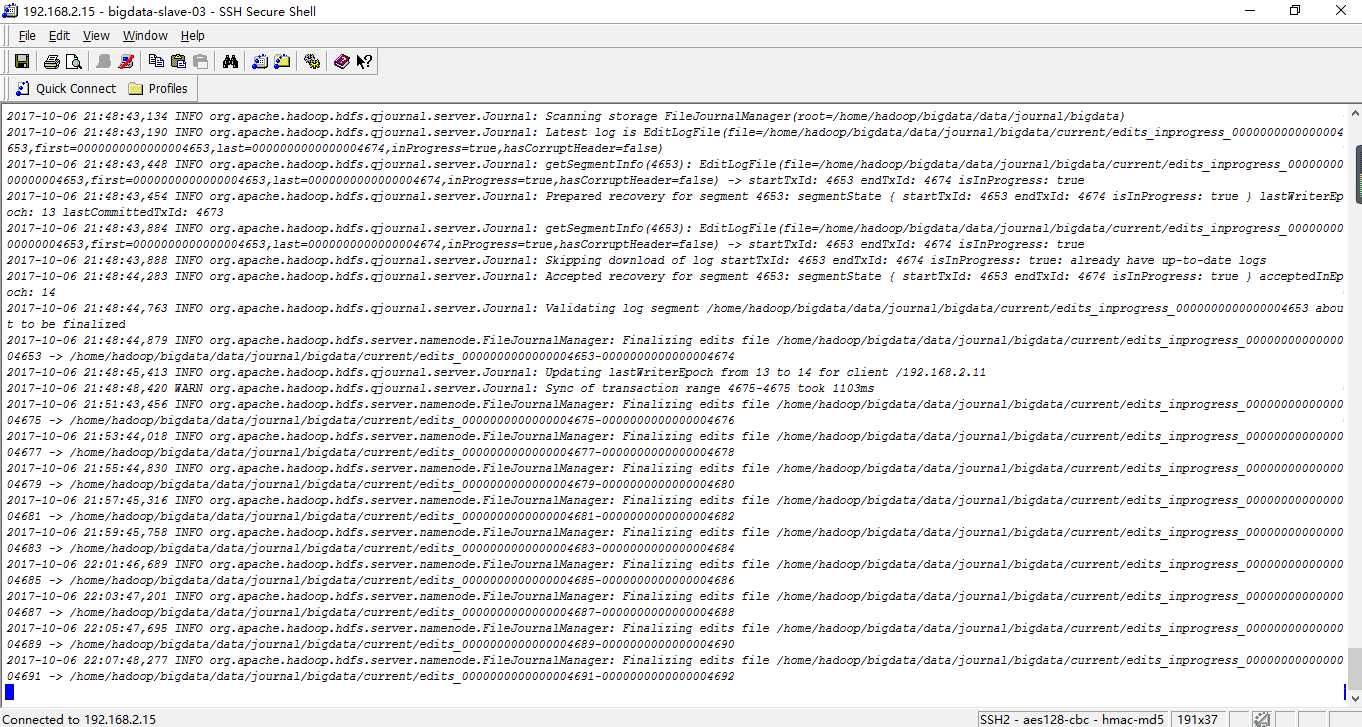
注：命令中的a1表示配置文件中的Agent的Name，如配置文件中的a1。flume-server.properties表示配置文件所在配置，需填写准确的配置文件路径。

### 测试Flume集群

1. 启动kafka集群和cusumer

/home/hadoop/bigdata/workspace/kafka\_2.11-0.11.0.1/bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic test --from-beginning

1. 依次启动flume Collector 和Agent(参考上面命令)
2. 查看接收的日志信息



### Flume集群命令

flume组件启动顺序：

channels——>sinks——>sources

关闭顺序：

sources——>sinks——>channels

* 1）启动

/home/hadoop/bigdata/workspace/apache-flume-1.8.0-bin/bin/flume-ng agent -n a1 -c conf -f /home/hadoop/bigdata/workspace/apache-flume-1.8.0-bin/conf/flume-server.properties -Dflume.root.logger=DEBUG,console 1>/dev/null 2>&1 &

/home/hadoop/bigdata/workspace/apache-flume-1.8.0-bin/bin/flume-ng agent -n agent1 -c conf -f /home/hadoop/bigdata/workspace/apache-flume-1.8.0-bin/conf/flume-client.properties -Dflume.root.logger=DEBUG,console 1>/dev/null 2>&1 &

* 2）关闭

## 十六．集群管理

### 1．正确的启动顺序

**1) ZooKeeper ->**[**Hadoop**](http://lib.csdn.net/base/hadoop)**-> Spark**

**2) ZooKeeper -> JournalNode (Hadoop) -> NameNode (Hadoop) -> DataNode (Hadoop) -> 主 ResourceManager/NodeManager (Hadoop) -> 备份 ResourceManager (Hadoop) -> ZKFC (Hadoop) -> MapReduce JobHistory (Hadoop) -> Spark**

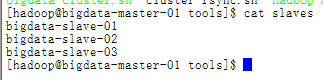
**关闭集群的顺序则相反**

### 2．脚本部署在下面的路径下

/home/hadoop/bigdata/tools

## 创建Slaves 配置文件

**文件名**：slaves   
**功能**：记录运行 datanode 、nodemanager、HRegionServer、QuorumPeerMain、JournalNode 的节点 IP 地址或主机名   
**内容**



### 3.管理脚本

1) ZooKeeper 管理脚本

文件名：zk-manager.sh

功能：启动、关闭与重启 ZooKeeper 集群，并可查看运行 ZK 服务的的模式（leader or follower？）

内容:

#!/bin/bash

SLAVES=$(cat slaves)

start\_time=`date +%s`

for slave in $SLAVES

do

case $1 in

start) ssh -t $slave "source ~/.bash\_profile;zkServer.sh start" 1>/dev/null;;

stop) ssh -t $slave "source ~/.bash\_profile;zkServer.sh stop" 1>/dev/null ;;

status) ssh -t $slave "source ~/.bash\_profile;zkServer.sh status" ;;

restart)ssh -t $slave "source ~/.bash\_profile;zkServer.sh restart" 1>/dev/null;;

\*) echo -e "Usage: sh zk-manager.sh {start|stop|restart} ^\_^\n" && exit ;;

esac

done

end\_time=`date +%s`

elapse\_time=$((${end\_time}-${start\_time}))

echo -e "\n$1 ZooKeeper Server takes ${elapse\_time} seconds\n"

2）JournalNode 管理脚本

文件名：journal-manager.sh

功能：启动、关闭运行在各个 slaves 上的 JournalNode 进程

内容

#!/bin/bash

start\_time=`date +%s`

SLAVES=$(cat slaves)

for slave in $SLAVES

do

case $1 in

start) ssh -t $slave "hadoop-daemon.sh start journalnode" ;;

stop) ssh -t $slave "hadoop-daemon.sh stop journalnode" ;;

\*) echo -e "Usage: sh journal-manager.sh {start|stop} ^\_^\n" && exit ;;

esac

done

end\_time=`date +%s`

elapse\_time=$((${end\_time}-${start\_time}))

echo -e "\n$1 JournalNode Server takes ${elapse\_time} seconds\n"

3） Hadoop 管理脚本

文件名：hadoop-manager.sh

功能：管理 hadoop 的启动与关闭

内容

#!/bin/bash

start\_time=`date +%s`

SLAVES=$(cat slaves)

for slave in $SLAVES

do

case $1 in

start) ssh -t $slave "hadoop-daemon.sh start journalnode" ;;

stop) ssh -t $slave "hadoop-daemon.sh stop journalnode" ;;

\*) echo -e "Usage: sh journal-manager.sh {start|stop} ^\_^\n" && exit ;;

esac

done

end\_time=`date +%s`

elapse\_time=$((${end\_time}-${start\_time}))

echo -e "\n$1 JournalNode Server takes ${elapse\_time} seconds\n"

[hadoop@bigdata-master-01 tools]$ clear

[hadoop@bigdata-master-01 tools]$ cat hadoop-manager.sh

# which machine to be active NameNode

NameNode\_1=bigdata-master-01

# which machine to be standy NameNode

NameNode\_2=bigdata-master-02

# which machine to be active ResourceManager

ResourceManager\_1=bigdata-master-01

# which machine to be standby ResourceManager

ResourceManager\_2=bigdata-master-02

# which machine to be JobHistoryServer

HistoryServer=bigdata-master-01

start\_time=`date +%s`

# make sure which namenode is active and which resourcemanager is active

function getServiceState () {

hdfs haadmin -getServiceState nameNode1 | grep 'active' >> /dev/null && NameNode\_Active=${NameNode1} && NameNode\_Standby=${NameNode2}

hdfs haadmin -getServiceState nameNode2 | grep 'active' >> /dev/null && NameNode\_Active=${NameNode2} && NameNode\_Standby=${NameNode1}

yarn rmadmin -getServiceState rm1 | grep 'active' >> /dev/null && ResourceManager\_Active=${ResourceManager\_1} && ResourceManager\_Standby=${ResourceManager\_2}

yarn rmadmin -getServiceState rm2 | grep 'active' >> /dev/null && ResourceManager\_Active=${ResourceManager\_2} && ResourceManager\_Standby=${ResourceManager\_1}

}

case $1 in

start) ssh -t ${NameNode\_1} "hadoop-daemon.sh start namenode";

ssh -t ${NameNode\_1} "hadoop-daemon.sh start zkfc";

ssh -t ${NameNode\_2} "hadoop-daemon.sh start namenode";

ssh -t ${NameNode\_2} "hadoop-daemon.sh start zkfc";

ssh -t ${NameNode\_1} "hadoop-daemons.sh start datanode";

ssh -t ${ResourceManager\_1} "start-yarn.sh";

ssh -t ${ResourceManager\_2} "yarn-daemon.sh start resourcemanager";

ssh -t ${HistoryServer} "mr-jobhistory-daemon.sh start historyserver";

;;

stop) getServiceState

ssh -t ${HistoryServer} "mr-jobhistory-daemon.sh stop historyserver";

ssh -t ${ResourceManager\_2} "yarn-daemon.sh stop resourcemanager";

ssh -t ${ResourceManager\_1} "stop-yarn.sh";

ssh -t ${NameNode\_1} "hadoop-daemons.sh stop datanode";

ssh -t ${NameNode\_2} "hadoop-daemon.sh stop namenode";

ssh -t ${NameNode\_1} "hadoop-daemon.sh stop namenode";

ssh -t ${NameNode\_2} "hadoop-daemon.sh stop zkfc";

ssh -t ${NameNode\_1} "hadoop-daemon.sh stop zkfc";

;;

\* ) echo -e "Usage: hadoop-manager.sh {start|stop} ^\_^\n" && exit;

;;

esac

end\_time=`date +%s`

elapse\_time=$((${end\_time}-${start\_time}))

echo -e "$1 Hadoop Server takes ${elapse\_time} seconds\n"

4）Spark 管理脚本

文件名：spark-manager.sh

功能：启动、关闭 Spark 集群

内容:

#!/bin/bash

spark\_01=bigdata-master-01

spark\_02=bigdata-master-02

spark\_03=bigdata-slave-01

spark\_04=bigdata-slave-02

spark\_05=bigdata-slave-03

start\_time=`date +%s`

case $1 in

start) ssh -t $spark\_01 "source ~/.bash\_profile;/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/start-all.sh" 1>/dev/null;

ssh -t $spark\_02 "source ~/.bash\_profile;/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/start-all.sh" 1>/dev/null;

ssh -t $spark\_03 "source ~/.bash\_profile;/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/start-all.sh" 1>/dev/null;

ssh -t $spark\_04 "source ~/.bash\_profile;/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/start-all.sh" 1>/dev/null;

ssh -t $spark\_05 "source ~/.bash\_profile;/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/start-all.sh" 1>/dev/null;

ssh -t $spark\_01 "source ~/.bash\_profile;/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/start-history-server.sh" 1>/dev/null;

;;

stop) ssh -t $spark\_01 "source ~/.bash\_profile;/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/stop-history-server.sh" 1>/dev/null;

ssh -t $spark\_05 "source ~/.bash\_profile;/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/stop-all.sh" 1>/dev/null;

ssh -t $spark\_04 "source ~/.bash\_profile;/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/stop-all.sh" 1>/dev/null;

ssh -t $spark\_03 "source ~/.bash\_profile;/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/stop-all.sh" 1>/dev/null;

ssh -t $spark\_03 "source ~/.bash\_profile;/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/stop-all.sh" 1>/dev/null;

ssh -t $spark\_01 "source ~/.bash\_profile;/home/hadoop/bigdata/workspace/spark-1.6.3-bin-hadoop2.6/sbin/stop-all.sh" 1>/dev/null;

;;

\*) echo -e "Usage: hadoop-manager.sh {start|stop} ^\_^\n" && exit;

;;

esac

end\_time=`date +%s`

elapse\_time=$((${end\_time}-${start\_time}))

echo -e "$1 Spark Server takes ${elapse\_time} seconds\n"

5)整个集群管理脚本

文件名：Bigdata-Cluster.sh

功能：统一启动、关闭及查看 ZooKeeper+Hadoop+Spark 大集群

内容:

#!/bin/bash

CLUSTER\_CONF\_PATH=$(cd "$(dirname "$0")"; pwd)

NameNode\_1=bigdata-master-01

NameNode\_2=bigdata-master-02

SLAVE\_1=bigdata-slave-01

SLAVE\_2=bigdata-slave-02

SLAVE\_3=bigdata-slave-03

start\_time=`date +%s`

function showJps() {

echo -e "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo -e "current the process on ${NameNode\_1} is:" && ssh -t ${NameNode\_1} source ~/.bash\_profile; jps

echo -e "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo -e "current the process on ${NameNode\_2} is:" && ssh -t ${NameNode\_2} source ~/.bash\_profile; jps

echo -e "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo -e "current the process on ${SLAVE\_1} is:" && ssh -t ${SLAVE\_1} source ~/.bash\_profile; jps

echo -e "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo -e "current the process on ${SLAVE\_2} is:" && ssh -t ${SLAVE\_2} source ~/.bash\_profile; jps

echo -e "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo -e "current the process on ${SLAVE\_3} is:" && ssh -t ${SLAVE\_3} source ~/.bash\_profile; jps

echo -e "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

}

case $1 in

start) sh $CLUSTER\_CONF\_PATH/zk-manager.sh start ;

sh $CLUSTER\_CONF\_PATH/journal-manager.sh start;

sh $CLUSTER\_CONF\_PATH/hadoop-manager.sh start;

#sh $CLUSTER\_CONF\_PATH/hbase-manager.sh start;

sh $CLUSTER\_CONF\_PATH/spark-manager.sh start;

showJps

;;

stop) sh $CLUSTER\_CONF\_PATH/spark-manager.sh stop;

#sh $CLUSTER\_CONF\_PATH/hbase-manager.sh stop;

sh $CLUSTER\_CONF\_PATH/spark-manager.sh stop;

sh $CLUSTER\_CONF\_PATH/hadoop-manager.sh stop;

sh $CLUSTER\_CONF\_PATH/journal-manager.sh stop;

sh $CLUSTER\_CONF\_PATH/zk-manager.sh stop;

showJps

;;

status) showJps

;;

\*) echo -e "Usage: sh bigdata-cluster.sh {start|stop|status} ^\_^\n" ;;

esac

end\_time=`date +%s`

elapse\_time=$((${end\_time}-${start\_time}))

echo -e "\n$1 Bigdata Cluster takes ${elapse\_time} seconds\n"

### 5.集群访问说明：

ResoureManager:192.168.2.11:50070

ClusterManager:192.168.2.11:8188

Spark UI:192.168.2.11:8080

Spark History Server:192.168.2.11:18080