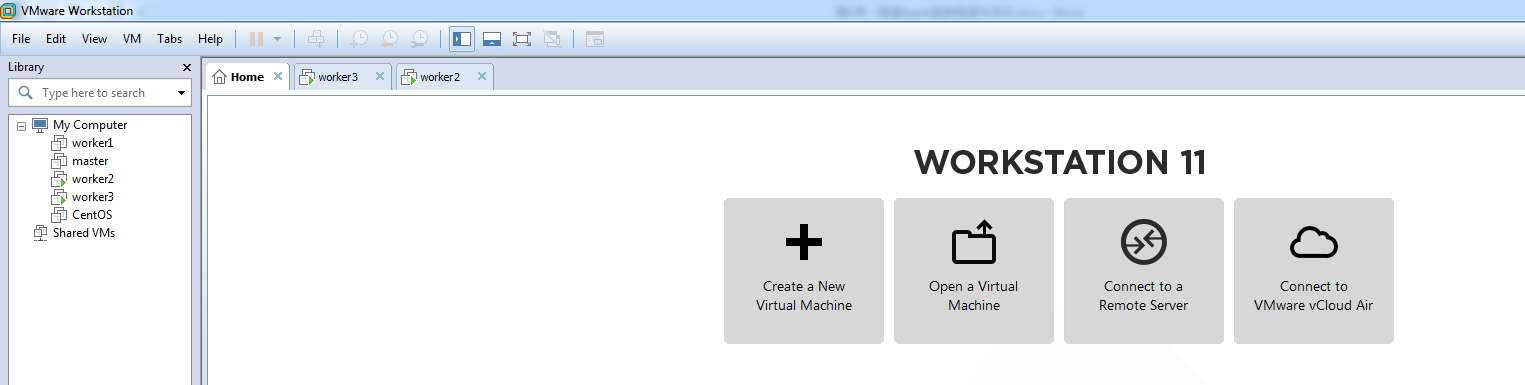
精通Spark集群搭建与测试

1. 安装 VMware Workstation 软件

推荐官方下载最新版，下载地址：

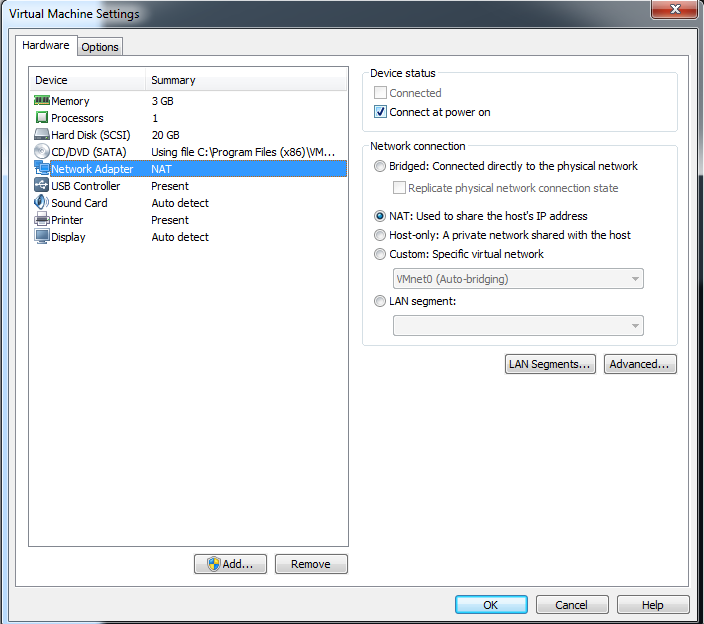
<https://my.vmware.com/cn/web/vmware/details?downloadGroup=WKST-1210-WIN&productId=524&rPId=9763>



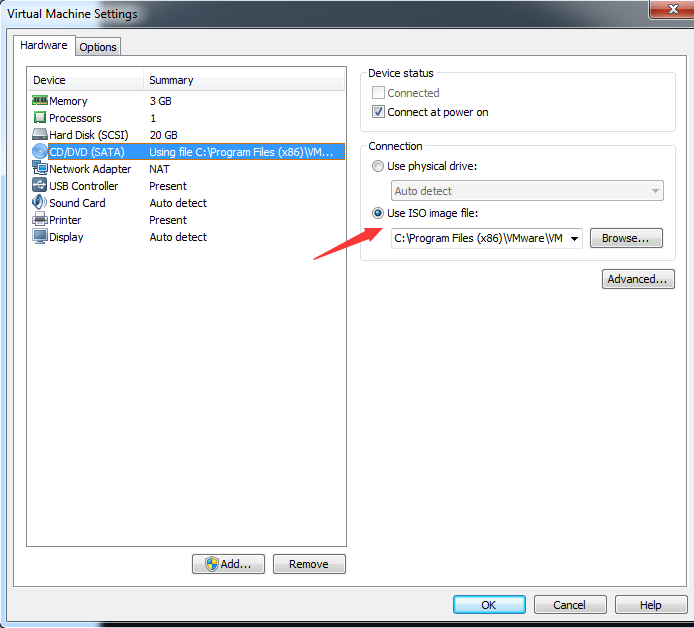
1. 运行VMware Workstation，新建4台虚拟机，并安装Ubuntu操作系统

Ubuntu下载地址:<http://www.ubuntu.org.cn/download/alternative-downloads>

需要配置虚拟机使之能够上网，在这里我们采用网络地址转换即NAT的方式，与宿主机共享IP上网：



按照下图指定Ubuntu的iso文件，然后点击‘power on this virtual machine’,然后按照提示一步步走下去即可完成操作系统的安装。



注1：可以先装好一台机器，然后通过VMware的克隆功能生成另外两台。

注2：安装完系统后，为了能从宿主机与虚拟机互相COPY文件，也为了能使虚拟机全屏显示，推荐安装VMwareTools.方法如下：

* 1. tar -xzvf VMwareTools-9.6.0-1294478.tar.gz
  2. cd vmware-tools-distrib/
  3. sudo ./vmware-install.pl
  4. 然后一路回车即可

1. 为了简化后续操作中的权限问题，我们在这里配置为root账户登录系统，方法如下：
2. 进入root用户权限模式：sudo –s
3. vim /etc/lightdm/lightdm.conf

[SeatDefaults]

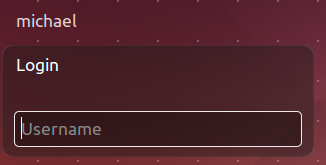
user-session=ubuntu

greeter-session=unity-greeter

greeter-show-manual-login=true

allow-guest=false

1. 为root账号设置密码: sudo passwd root
2. 重新启动系统后，即可用root账号登录： reboot -h now



注1：如果系统提示vim没有安装的话，可以通过apt-get install vim安装。

注2：切换为root账户登录后，如果遇到以下问题： 

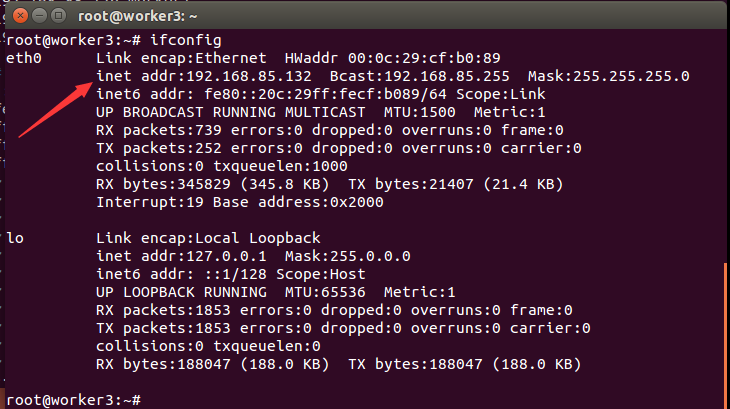
请按以下方法解决：vim /root/.profile，将“mesg n”，更改为“tty -s && mesg n”。

1. 在各个节点修改节点名称，并配置ip地址和hostname的对应关系：
   1. vim /etc/hostname, 将4台节点分别命名为master,worker1,worker2,worker3
   2. Restart使更改生效
   3. 验证hostname更改是否生效：用hostname命令
   4. 在各个节点修改ip地址和hostname的对应关系: vim /etc/hosts
   5. 验证hostname和ip的映射关系是否正确:用ping 命令

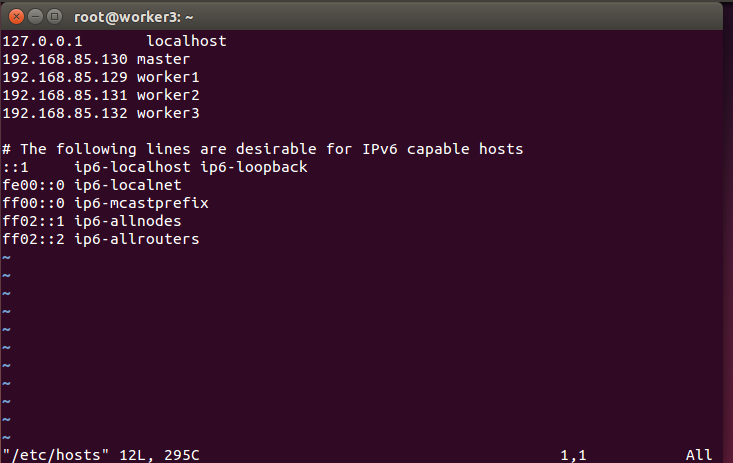
注1：用hostname命令验证更改是否生效：



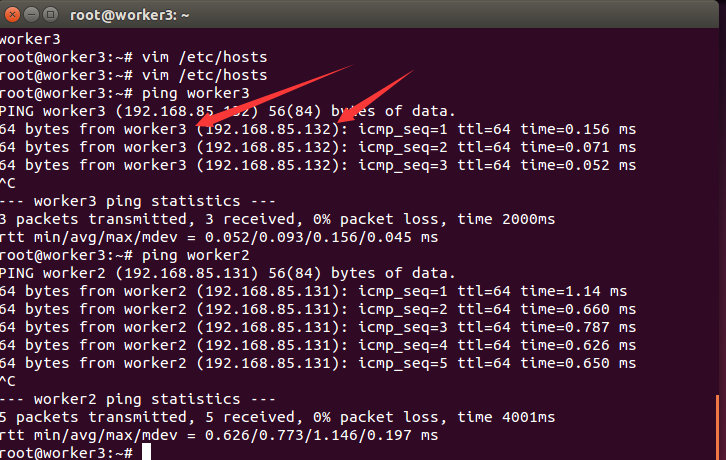
注2：用ifconfig来查看当前节点的ip:



注3：修改后，各节点的hosts文件如下所示：



注4：用Ping命令来验证hostname和ip的映射关系是否正确。



1. 在各个节点安装ssh并配置免密码登录。

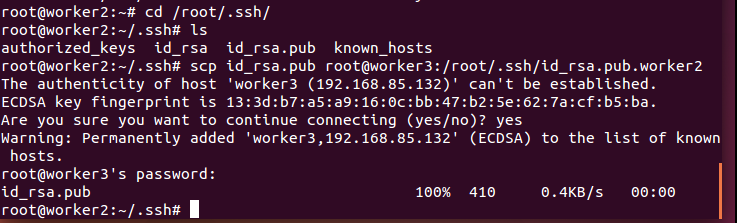
Hadoop是通过ssh进行通信的，所以需要安装SSH;

安装完SSH后还需设置免密码登录，才能省去每次通信时都输入密码的麻烦。

* 1. 安装ssh: apt-get install ssh;
  2. 启动ssh: /etc/init.d/ssh start
  3. 验证ssh是否正常启动：ps –e | grep ssh.若没有正常启动，可以尝试重启。
  4. 生产公钥和私钥：ssh-keygen -t rsa
  5. 将公钥追加到authorized\_keys中（authorized\_keys用于保存所有允许以当前用户身份登录到ssh客户端的用户的公钥内容）：cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys
  6. 验证单机上免密码登录是否配置成功：ssh localhost
  7. 重复以上步骤，在各个节点的/root/.ssh/目录下生成一个公钥id\_rsa.pub和一个私钥id\_rsa
  8. 将所有worker节点的公钥复制到master节点上：

scp id\_rsa.pub [root@master:/root/.ssh/id\_rsa.pub.worker1](mailto:root@master:/root/.ssh/id_rsa.pub.worker1)

scp id\_rsa.pub [root@master:/root/.ssh/id\_rsa.pub.worker2](mailto:root@master:/root/.ssh/id_rsa.pub.worker2)



* 1. 在master节点上，将所有worker的公钥追加到authorized\_keys中：

cat ~/.ssh/id\_rsa.pub.worker1 >> ~/.ssh/authorized\_keys

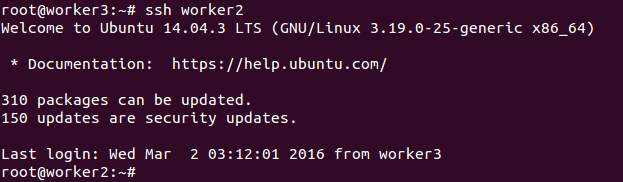
cat ~/.ssh/id\_rsa.pub.worker2 >> ~/.ssh/authorized\_keys

* 1. 将master节点的authorized\_keys（已经包含了所有节点的公钥）复制到各个worker节点的.ssh目录下：

scp authorized\_keys [root@worker1:/root/.ssh/authorized\_keys](mailto:root@worker1:/root/.ssh/authorized_keys)

scp authorized\_keys [root@worker2:/root/.ssh/authorized\_keys](mailto:root@worker2:/root/.ssh/authorized_keys)

* 1. 验证各个节点之间是否可以免密码登录：



1. 在各个节点安装rsync: apt-get install rsync
2. 在各个节点安装java,推荐从官方下载最新版本
   1. 新建java安装目录:mkdir /usr/lib/java
   2. 将下载的jdk文件解压到java安装目录中（安装了VMware tools后，可以直接将下载的安装包拖拉到安装目录中）：tar -xzf jdk-8u60-linux-x64.tar.gz
   3. 修改环境变量：vim ~/.bashrc

export JAVA\_HOME=/usr/lib/java/jdk1.8.0\_60

export JRE\_HOME=$JAVA\_HOME/jre

export CLASSPATH=$JAVA\_HOME/lib:$JRE\_HOME/lib:$CLASSPATH

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

* 1. 执行以下命令使配置文件的修改生效：source ~/.bashrc
  2. 验证java是否安装成功：java -version

1. 在各个节点安装hadoop，推荐从官网下载最新的稳定版本。

<http://apache.opencas.org/hadoop/common/hadoop-2.6.0/>

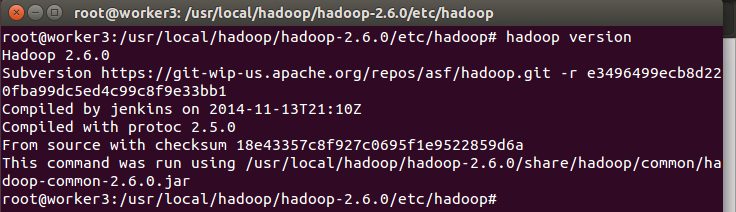


* 1. 创建hadoop安装目录：mkdir /usr/local/hadoop
  2. 把下载的hadoop拷贝到安装目录并解压（安装了VMware tools后，可以直接将下载的安装包拖拉到安装目录中）：tar -xzvf hadoop-2.6.0.tar.gz
  3. 为了可以在任意目录下使用hadoop命令，需要将hadoop的bin目录配置到.bashrc中,**vim ~/.bashrc**:

export HADOOP\_HOME=/usr/local/hadoop/hadoop-2.6.0

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

* 1. 使用source命令使更改生效：source ~/.bashrc
  2. verify the installation of hadoop: hadoop version



注：可以首先在一台节点上安装配置好，然后通过SCP命令拷贝到其他节点上。

1. 在各个节点配置hadoop：

可以首先在一台节点上配置好，然后通过SCP命令拷贝到其他节点上。

* 1. 具体操作前首先在hadoop根目录下创建以下文件夹：

mkdir tmp

mkdir dfs

mkdir dfs/data

mkdir dfs/name



注：这里新建的目录会在hdfs-site.xml中用到。实际上namenode节点上只需要dfs/name目录,datanode节点上只需要dfs/data 目录。

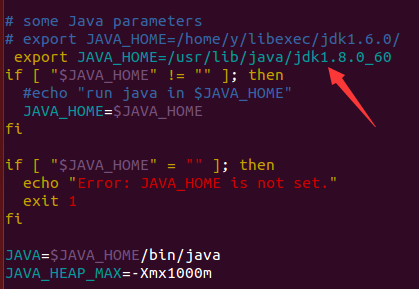
* 1. 修改Hadoop-env.sh，指定java安装目录:



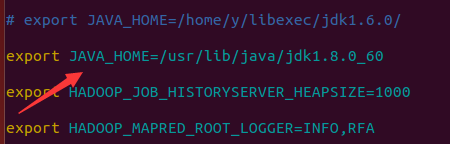
注：若没有配置JAVA\_HOME,后续执行hadoop命令时会报如下错误:



* 1. 修改yarn-env.sh, 指定java安装目录:



* 1. 修改mapred-env.sh, 指定java安装目录:



* 1. 修改配置文件slaves,将各个worker节点的hostname加进去：



* 1. **修改core-site.xml,以下是最小配置，更详细的信息可以参考官网：**

[**http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/core-default.xml**](http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/core-default.xml)

**vim core-site.xml**:

<configuration>

  <property>

    <name>fs.defaultFS</name>

    <value>hdfs://Master:9000</value>

<description>The name of the default file system</description>

  </property>

    <property>

    <name>hadoop.tmp.dir</name>

    <value>/usr/local/hadoop/hadoop-2.6.0/tmp</value>

    <description>A base for other temporary directories</description>

  </property>

</configuration>

<property>

<name>hadoop.native.lib</name>

<value>true</value>

<description>Should native hadoop libraries, if present, be used.</descripti

on>

</property>

* 1. **修改hdfs-site.xml,以下是最小配置，更详细的信息可以参考官网:**

<http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml>

**vim hdfs-site.xml:**

<configuration>

    <property>

        <name>dfs.replication</name>

        <value>2</value>

    </property>

<property>

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

        <value>/usr/local/hadoop/hadoop-2.6.0/dfs/name</value>

</property>

<property>

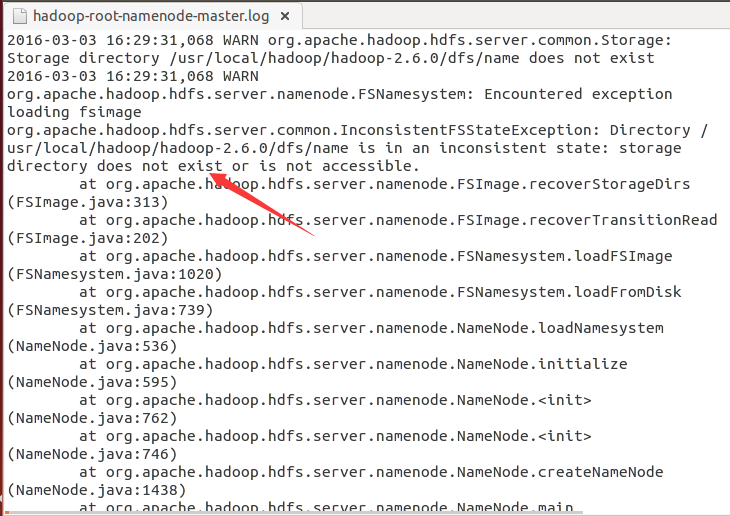
<name>dfs.datanode.data.dir</name>

<value>/usr/local/hadoop/hadoop-2.6.0/dfs/data</value>

</property>

<configuration>

注：这里指定的dfs.namenode.name.dir与dfs.datanode.data.dir若不存在的话，后续start-dfs时会报错：



* 1. **修改mapred-site.xml,以下是最小配置，更详细的信息可以参考官网:**

[**http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml**](http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml)

**注：MRv1的Hadoop没有使用yarn作为资源管理器，其配置如下：**

**vim mapred-site.xml：（without yarn）**

**<configuration>**

**<property>**

**<name>mapred.job.tracker</name>**

**<value>master:9001</value>**

**</property>**

**</configuration>**

**MRv2的hadoop使用yarn作为资源管理器,其配置如下：**

**vim mapred-site.xml：（with yarn）**

<configuration>

    <property>

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

        <value>yarn</value>

    </property>

</configuration>

* 1. **修改yarn-site.xml,以下是最小配置，更详细的信息可以参考官网:**

<http://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-common/yarn-default.xml>

**vim yarn-site.xml:**

<configuration>

<property>

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

<value>master</value>

</property>

<property>

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

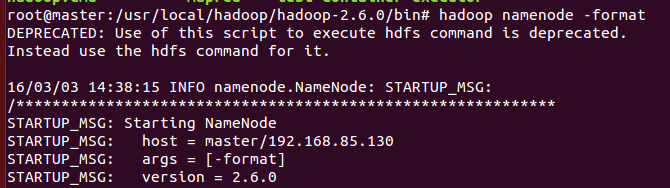
<value>mapreduce\_shuffle</value>

</property>

</configuration>

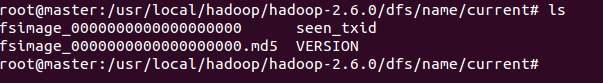
注：Yarn是Hadoop推出整个分布式（大数据）集群的资源管理器，负责资源的管理和分配，基于Yarn我们可以在同一个大数据集群上同时运行多个计算框架，例如Spark、MapReduce、Storm。

1. 启动并验证hadoop集群：
   1. 格式化hdfs文件系统：**hadoop namenode –format/hdfs namenode -format**

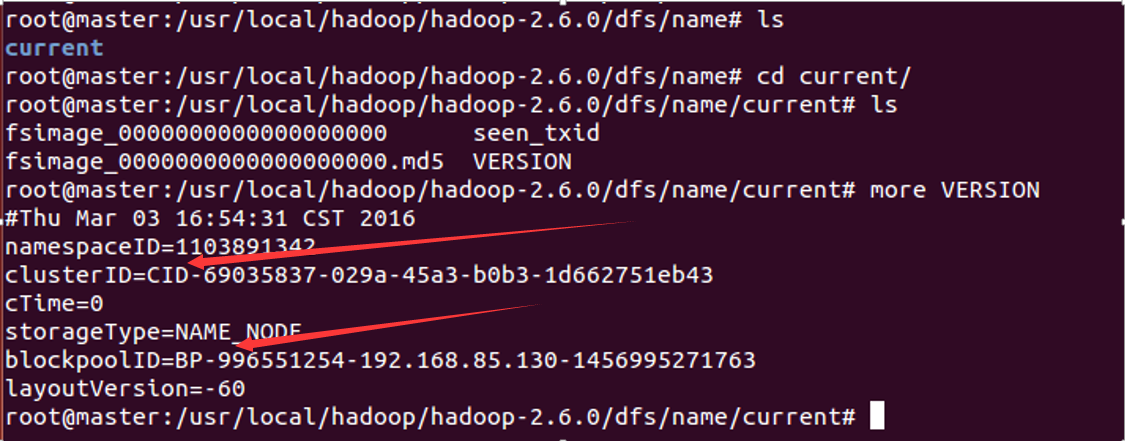


该命令会启动，格式化，然后关闭namenode。

实际上格式化后，在namenode上会生成以下文件：



其中VERSION文件的内容如下：

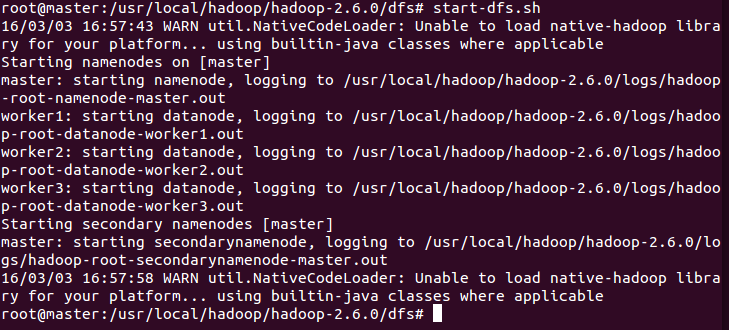


该命令不会在datanode的dfs.datanode.data.dir 对应的目录下生成任何文件：

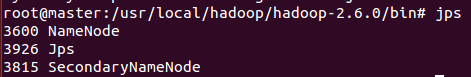


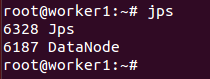
有关该命令的细节请参考官方文档：<http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html#namenode>

* 1. **启动hdfs: start-dfs.sh**



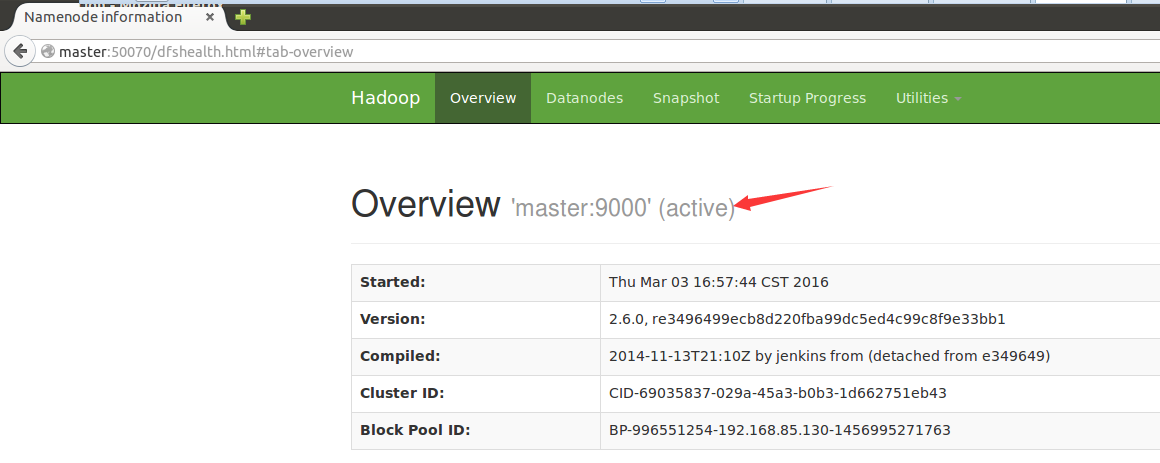
**使用jps验证HDFS是否启动成功：**





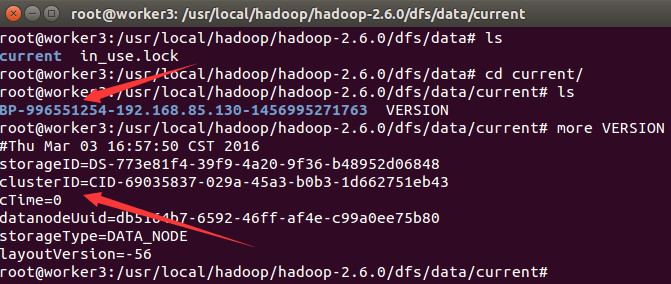
**通过webui检查HDFS是否启动成功**

<http://master:50070>

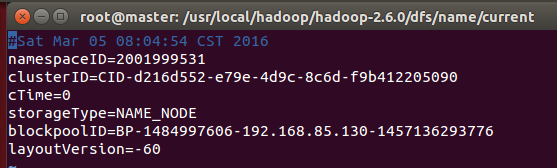




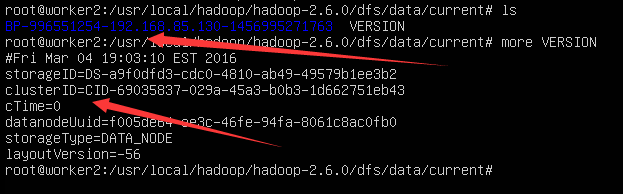
注1：实际上第一次启动hdfs后，在datanode 的dfs.datanode.data.dir 对应的目录下会生成current目录，该目录下的BP文件与namenode上dfs.namenode.name.dir对应的目录下的current子目录的VERSION文件中的blockpoolID字段的值一致；在该目录下也会生成VERSION文件，该VERSION文件中的clusterID和namenode的dfs.namenode.name.dir对应的目录下的current子目录的VERSION文件中的clusterID一致：



实际上在后续执行了hdfs namenode –format后，namenode的VERSION文件会改变：



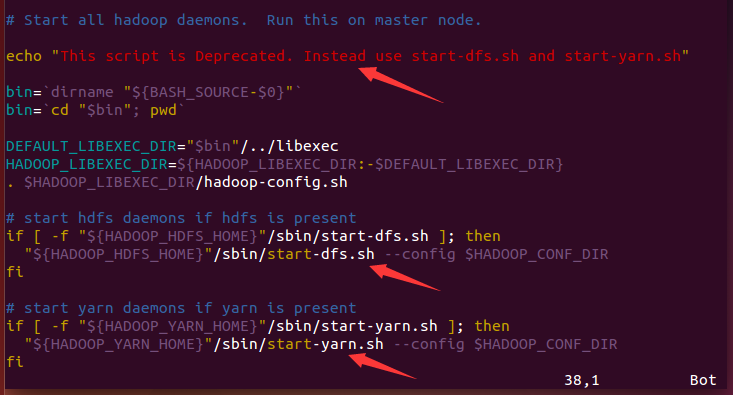
而dananode的BP和VERSION文件都不会改变：



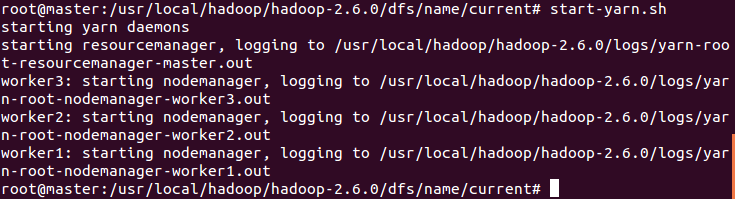
再次start-dfs.sh时，namenode可以成功启动，但在datanode上，因为version文件与namenode的不一致，datanode不能成功启动并成功注册到namenode！

所以：每次执行hdfs namenode –format前，必须清空datanode的data文件夹！(namenode的name文件夹不需要清空，namenode和datanode的tmp文件夹也不需要空。)

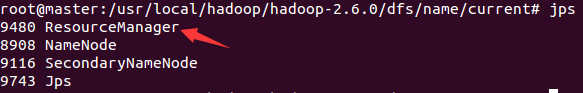
注2：注：有的朋友喜欢使用start-all.sh，其实质是执行了start-dfs.sh和start-yarn.sh,如下图可见，在提示中也可见，推荐分开使用start-dfs.sh和start-yarn.sh而不是直接使用start-all.sh:

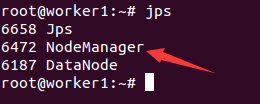


* 1. 启动yarn: start-yarn.sh



**使用jps验证yarn是否启动成功：**



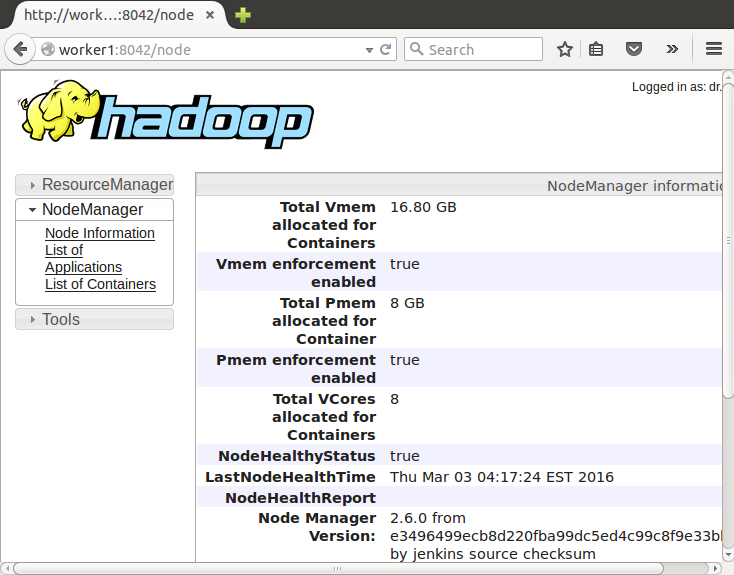


**通过webui检查yarn是否启动成功:**

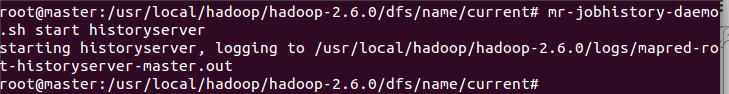
<http://master:8088/>

<http://worker1:8042/>

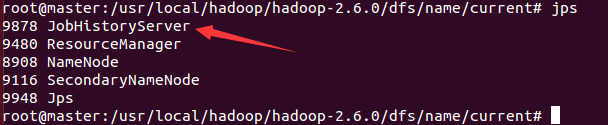




* 1. 启动JobHistory Server：mr-jobhistory-daemon.sh start historyserver

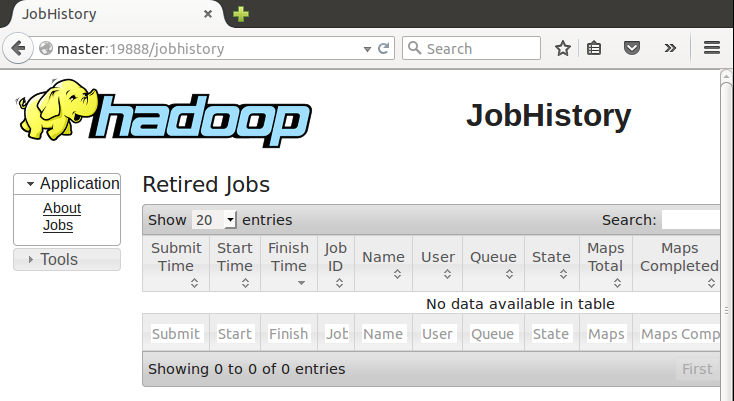


**使用jps验证**JobHistory Server**是否启动成功：**



**通过webui检查**JobHistory Server**是否启动成功:**

<http://master:19888>



* 1. 验证hadoop集群

创建文件夹：

hdfs dfs -mkdir -p /data/wordcount

**hdfs dfs -mkdir -p /output**

上传文件：

hdfs dfs -put /usr/local/hadoop/hadoop-2.6.0/etc/hadoop/\*.xml /data/wordcount

查看上传文件是否成功：

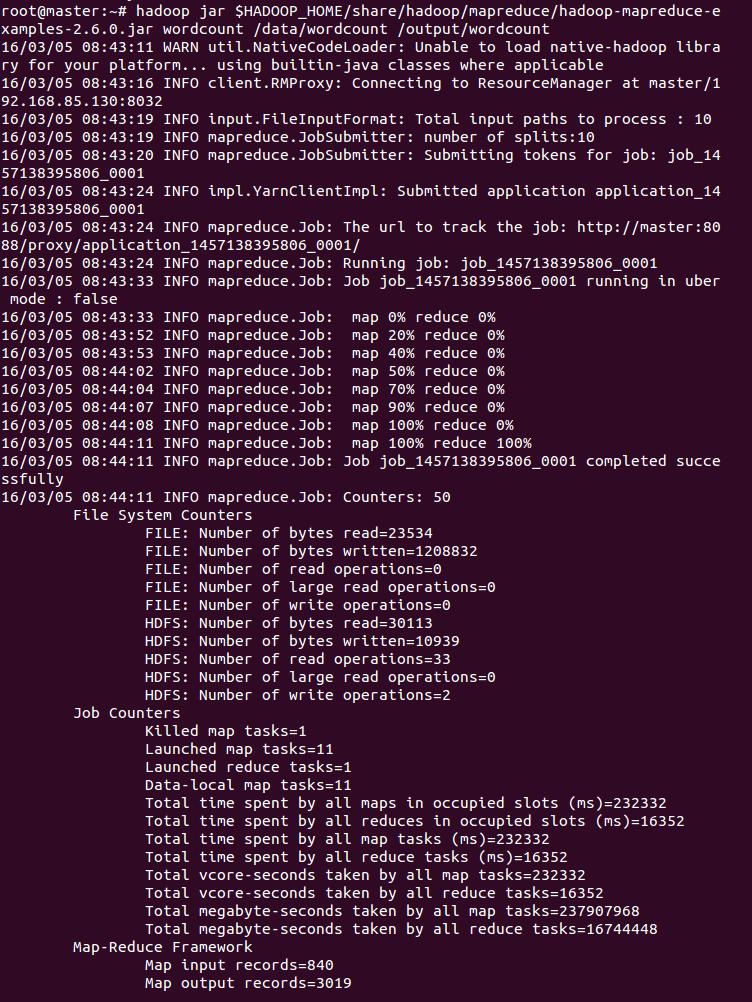
hdfs dfs -ls /data/wordcount

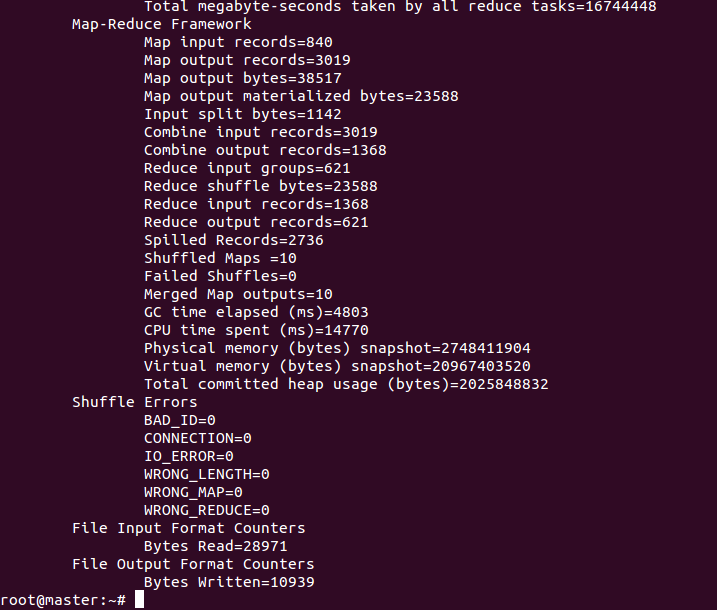


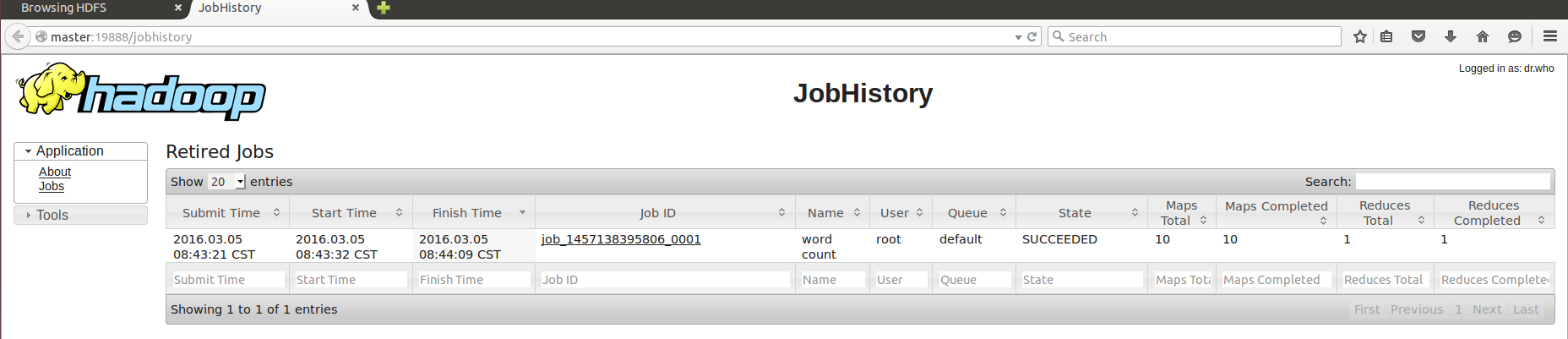
尝试执行hadoop自带的wordcount程序：

hadoop jar $HADOOP\_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar wordcount /data/wordcount /output/wordcount

下图可见，执行成功：





也可以去webui里查看jobhistory：

hadoop的常见命令细节请参考官方文档：

<http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html#dfs>

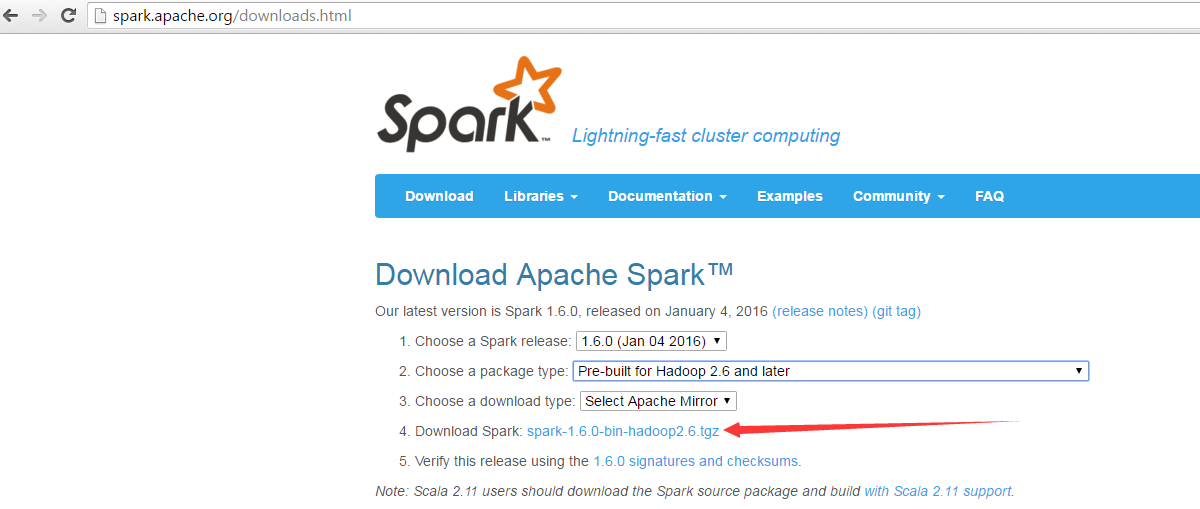
1. 在各个节点安装scala,推荐从官方下载最新版本
   1. 新建scala安装目录:mkdir /usr/lib/scala
   2. 将下载的scala文件解压到scala安装目录中（安装了VMware tools后，可以直接将下载的安装包拖拉到安装目录中）：tar -xzvf scala-2.10.4.tgz
   3. 修改环境变量：vim ~/.bashrc

export SCALA\_HOME=/usr/lib/scala/scala-2.10.4

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

* 1. 执行以下命令使配置文件的修改生效：source ~/.bashrc
  2. 验证scala是否安装成功：scala -version

1. 在各个节点安装spark，推荐从官网下载最新的稳定版本:

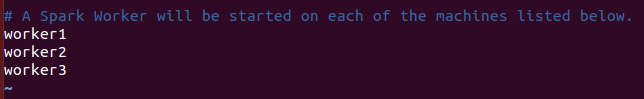


1. 创建spark安装目录：mkdir /usr/local/spark
2. 把下载的spark拷贝到安装目录并解压（安装了VMware tools后，可以直接将下载的安装包拖拉到安装目录中）：tar -xzvf spark-1.6.0-bin-hadoop2.6.tgz
3. 为了可以在任意目录下使用spark命令，需要将spark的bin和sbin目录配置到.bashrc中,**vim ~/.bashrc**:

export SPARK\_HOME =/usr/local/spark/spark-1.6.0-bin-hadoop2.6

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

1. 使用source命令使更改生效：source ~/.bashrc
2. 在各个节点配置spark,一下配置都是在$SPARK\_HOME/conf目录下：
   1. 修改slaves文件，（若没有slaves文件可以cp slaves.template slaves创建），添加worker节点的Hostname，修改后内容如下:



* 1. 配置spark-env.sh，（若没有该文件可以cp spark-env.sh.template spark-env.sh创建），添加如下内容：

export JAVA\_HOME=/usr/lib/java/jdk1.8.0\_60

export SCALA\_HOME=/usr/lib/scala/scala-2.10.4

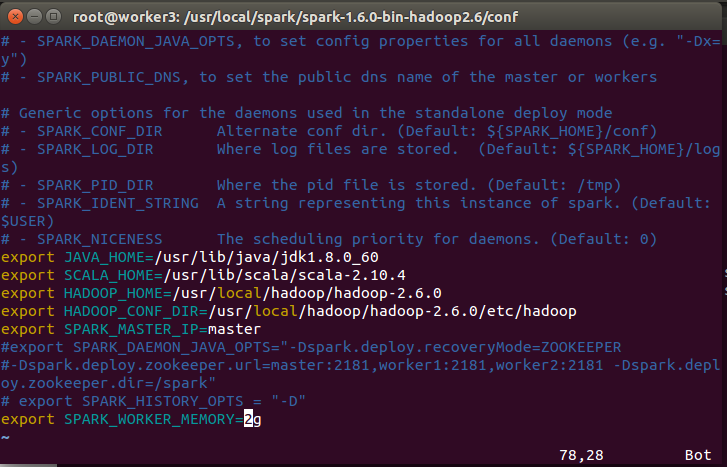
export HADOOP\_HOME=/usr/local/hadoop/hadoop-2.6.0

**export HADOOP\_CONF\_DIR=/usr/local/hadoop/hadoop-2.6.0/etc/hadoop**

**export SPARK\_MASTER\_IP=master**

export SPARK\_WORKER\_MEMORY=1g

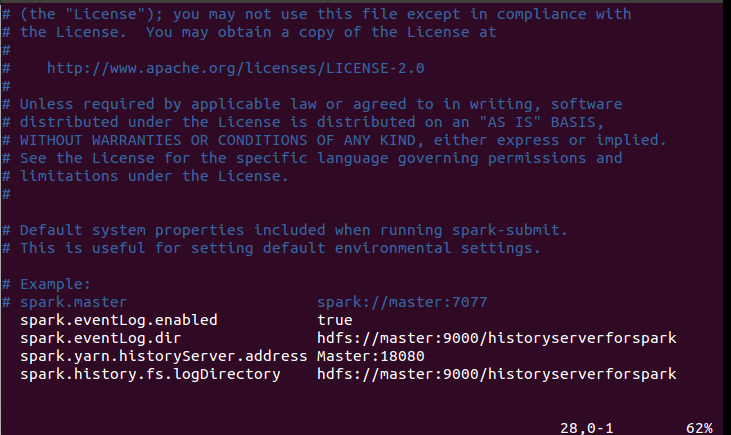
修改后的内容如下：



更详细的配置说明，请参考官方文档：

<http://spark.apache.org/docs/latest/spark-standalone.html#cluster-launch-scripts>

* 1. 可选，配置spark-defaults.sh:



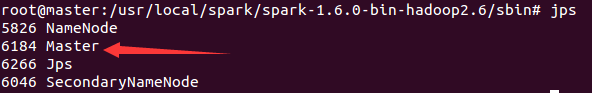
1. 启动并验证spark集群：

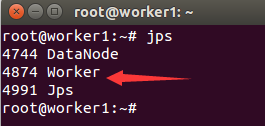
Spark只是一个计算框架，并不提供文件系统功能，故我们需要首先启动文件系统hdfs；在standalone模式下，我们并不需要启动yarn功能，故不需要启动yarn.

1. 用start-dfs.sh启动hdfs,参考step10.
2. 在hadoop集群启动成功的基础上，启动spark集群,常见的做法是在master节点上start-all.sh：

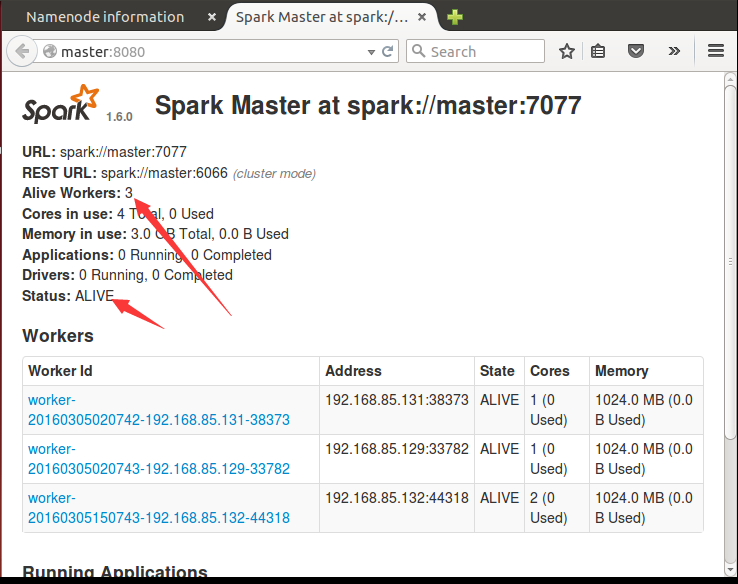


1. 使用jps在master和worker节点上验证spark集群是否正确启动：





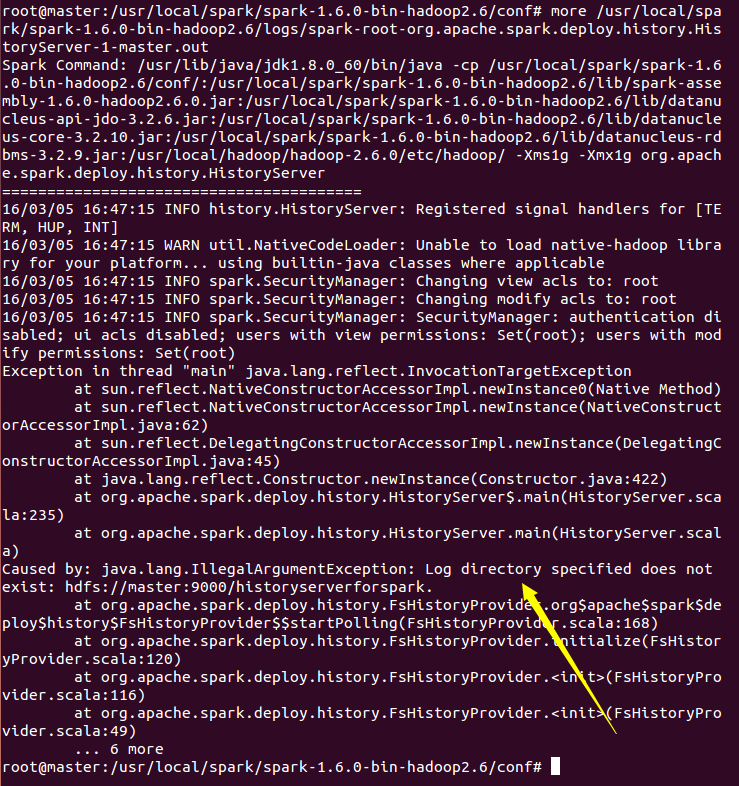
1. 通过webui查看spark集群是否启动成功：**http://master:8080**



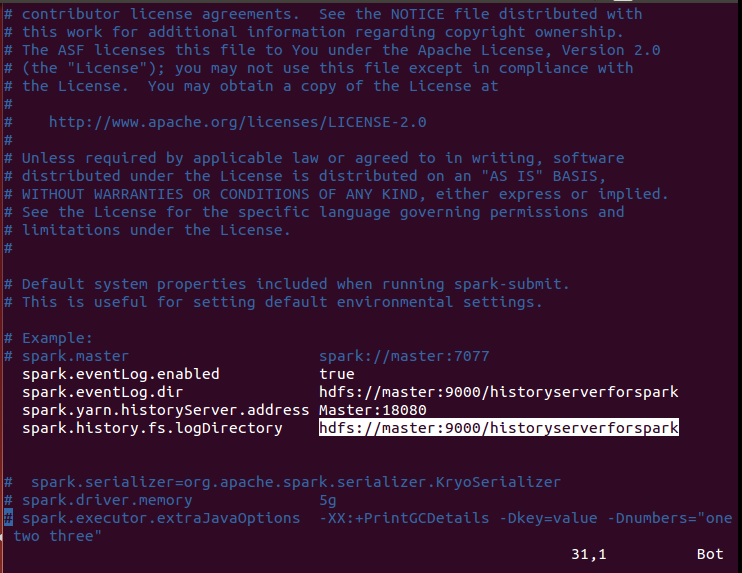
1. 启动spark的historyserver: start-history-server.sh



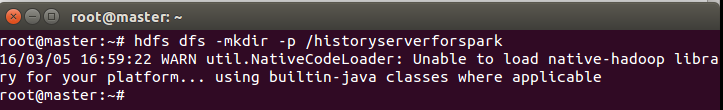
Jps可见，historyserver没能启动成功。仔细观察日志可见，指定的日志目录不存在:

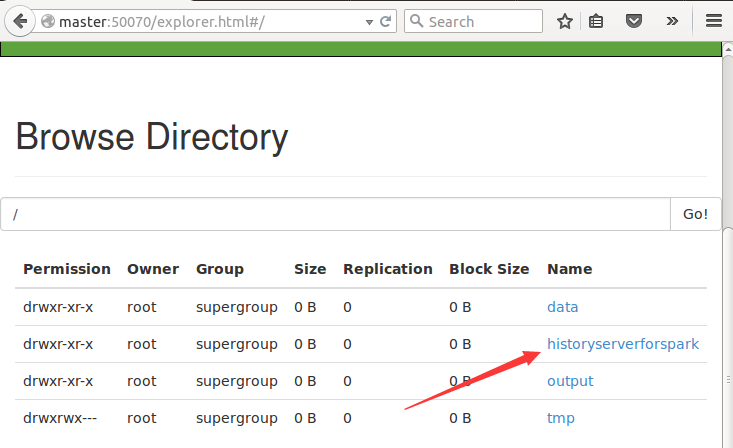


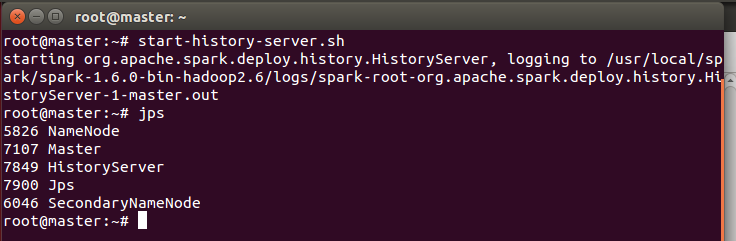
该目录是我们在spark-defaults.conf中指定的（事实上，若不指定spark.eventLog.dir和spark.history.fs.logDirectory，你也无法启动historyserver，因为系统不知道去哪里存储你的history信息）



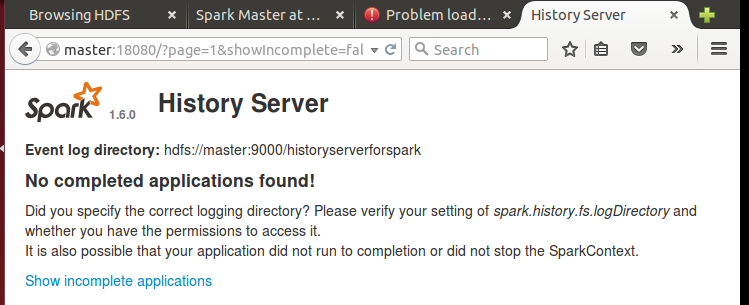
我们只需使用hdfs命令hdfs dfs -mkdir -p /historyserverforspark来创建该目录，再启动historyserver, jps可见多了个historyserver进程：



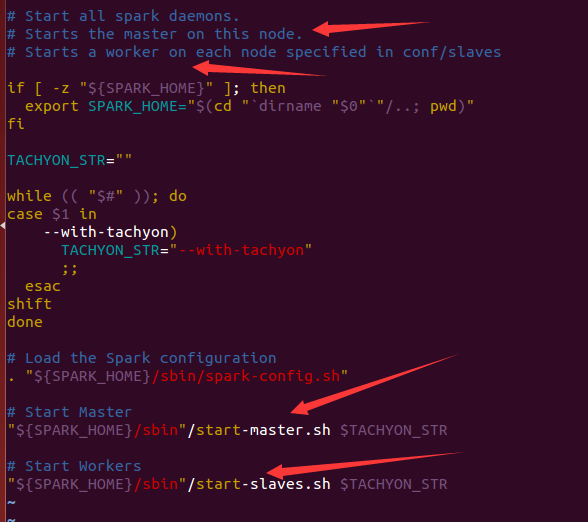




1. 使用webUI来查看historyserver是否成功启动 <http://master:8080>

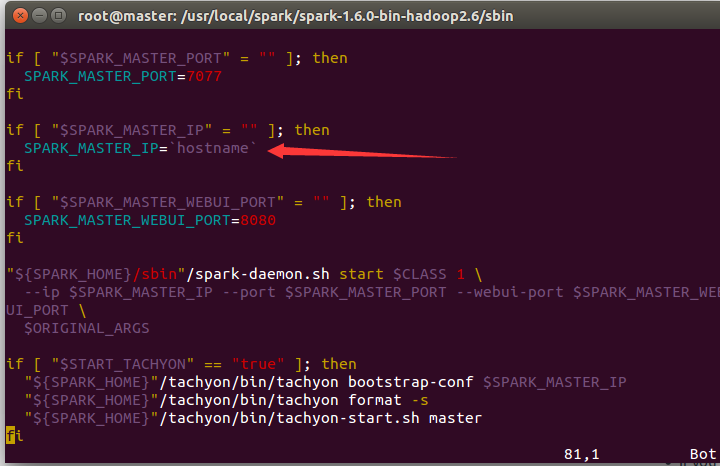


注1：start-all.sh其实质是先后执行了start-master.sh和start-slaves.sh:



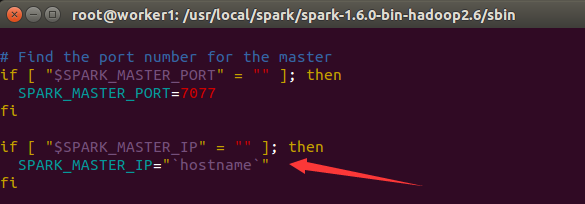
最佳实践是：在master节点上执行start-all.sh.

注2：start-master.sh会首先尝试从spark-env.sh中获取spark\_master\_ip，获取到的话，就在该参数对应的节点上（可以是当前节点也可以不是当期节点）启动master进程；如果没有获取到的话，则会在当前节点上启动master进程：



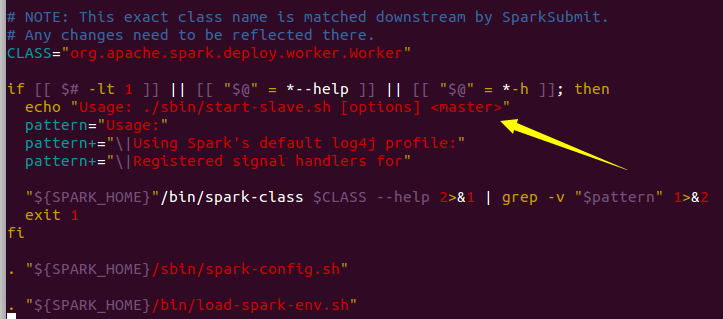
最佳实践是：在master节点上执行start-master.sh.

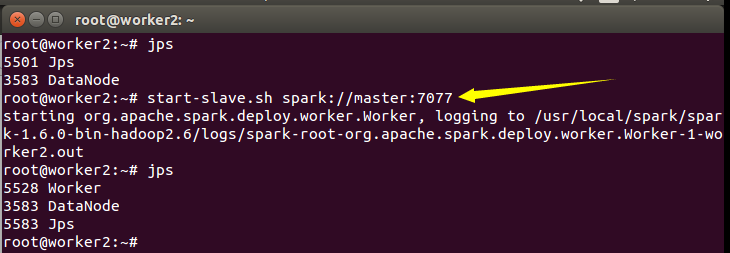
注3：start-slaves.sh会在slaves 文件中指定的每个节点上分别调用start-slave.sh来启动worker进程，并尝试注册到特定的master上。这个master通过以下方式获取：首先尝试从spark-env.sh中获取spark\_master\_ip，获取到的话，该参数对应的节点（可以是当前节点也可以不是当期节点）就是master节点；如果没有获取到的话，则会视当前节点为master节点。若该master节点上的master进程没有启动，这些worker节点上的worker进程会不断尝试注册到master上：



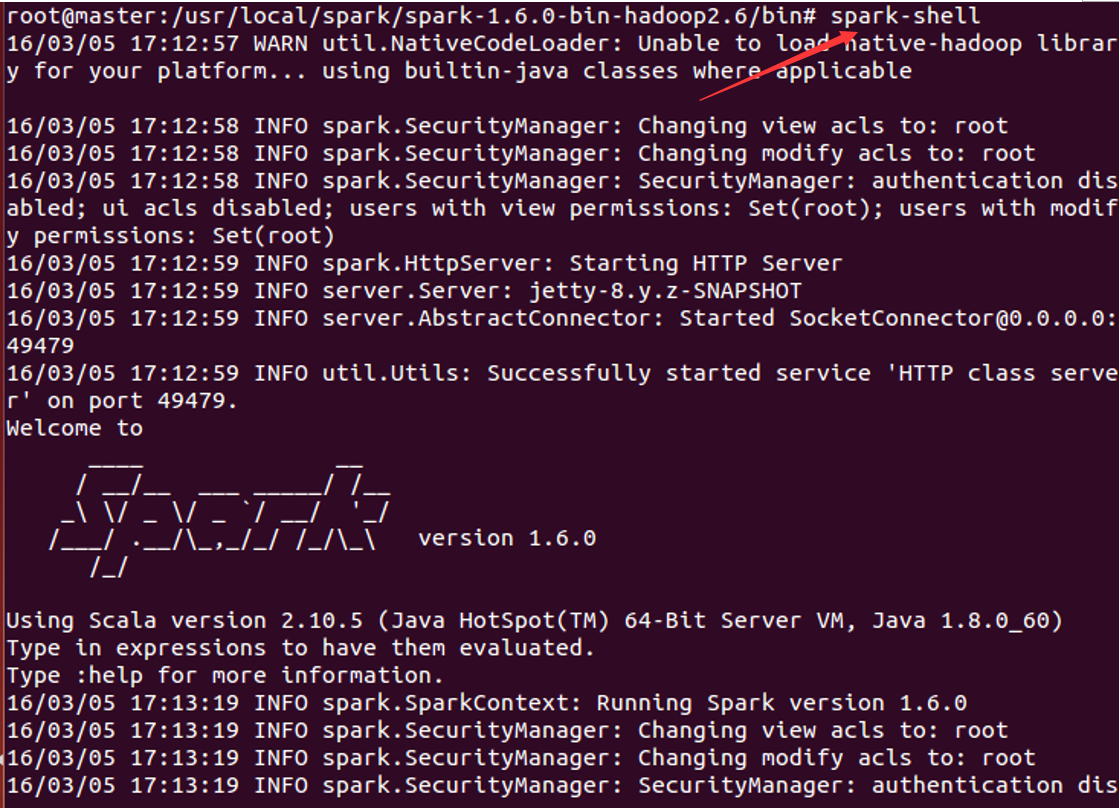
最佳实践是：在master节点上start-slaves.sh.

注4：start-slave.sh:该命令可以动态启动worker节点并注册到到master上，这样当我们已经启动了spark集群后，当后续有新的节点可用时，无需stop整个集群，只需要在新的可用节点上执行该命令就可以动态启动并注册到master上。需要注意的是，当使用该命令时，必须在命令行指定master:



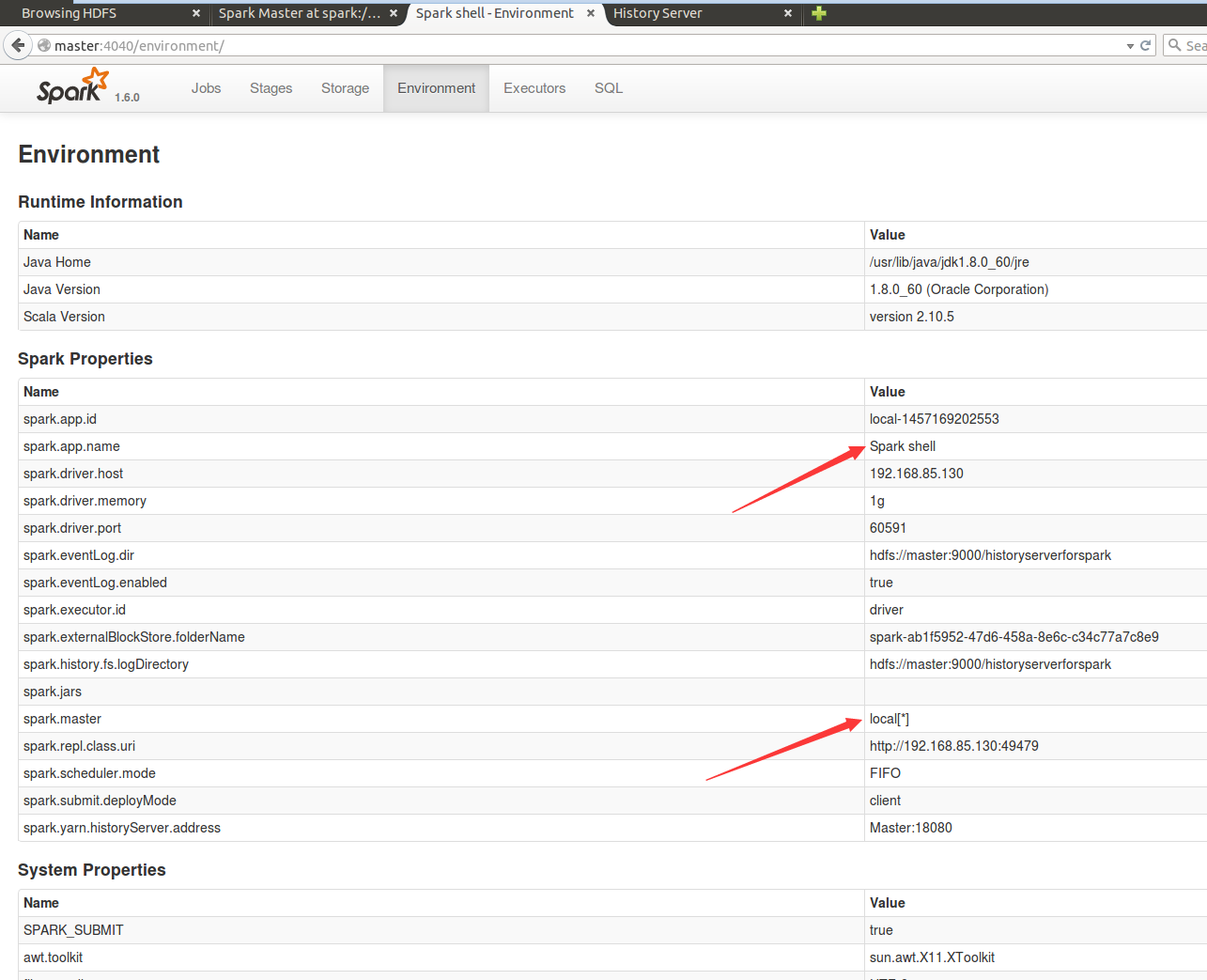


1. Spark为我们提供了spark-shell这个交互式脚本，它是我们学习spark的一个简单有效的途径。需要注意的是，在启动spark-shell时，若你没有指定参数--master, spark-shell是运行在本地单机模式的。
2. Spark-shell单机模式:

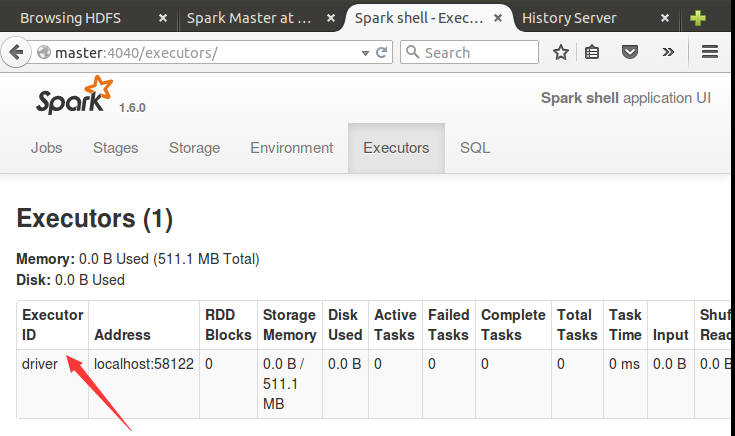


通过webUI的方式查看系统信息：<http://master:4040>

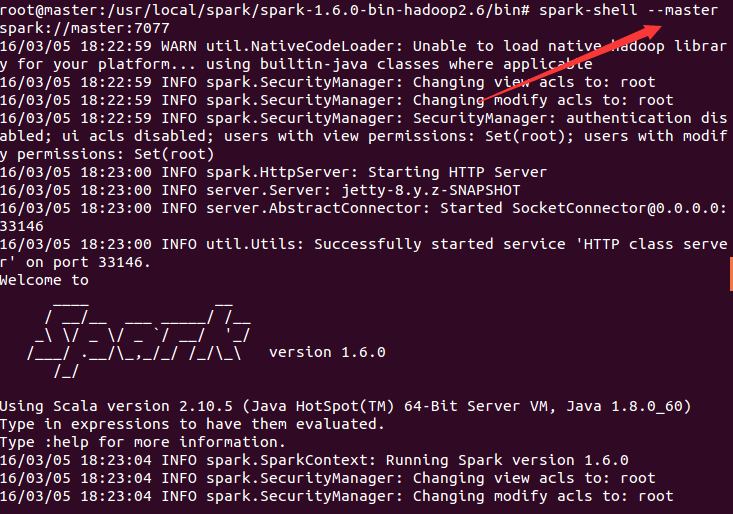
Environment信息：



Executors信息：

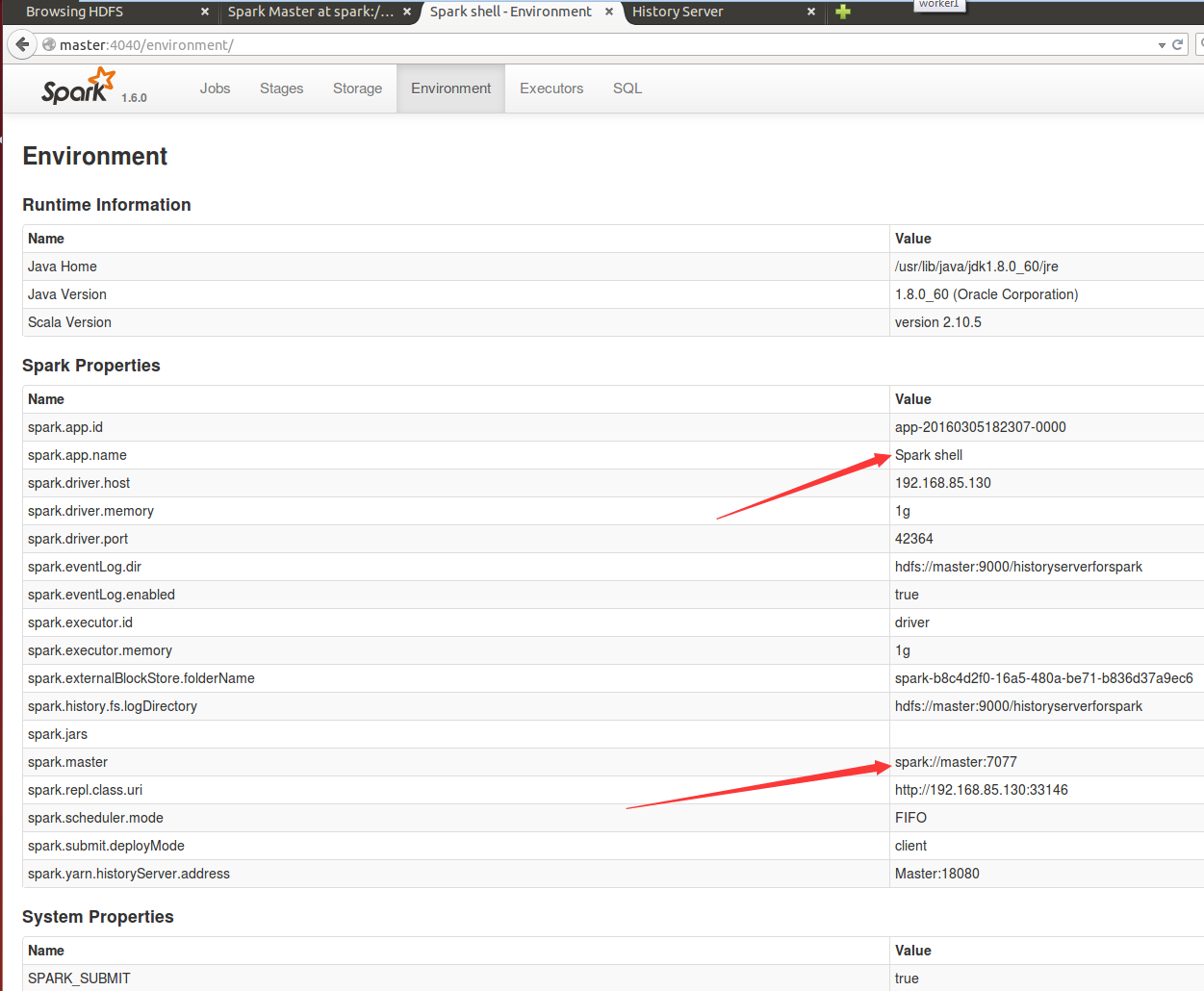


1. Spark-shell集群模式：

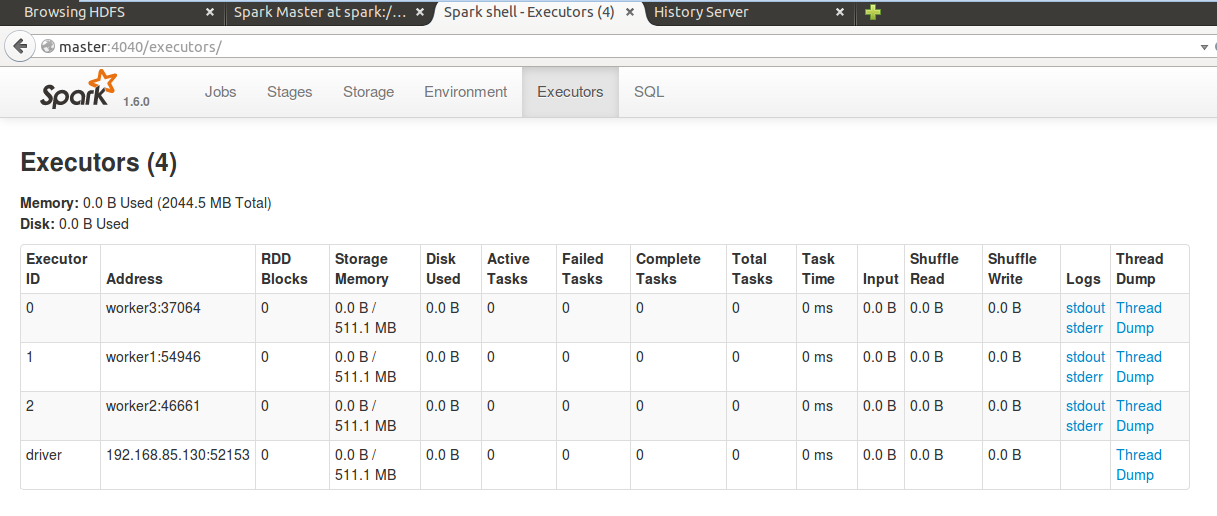


通过webUI的方式查看系统信息：<http://master:4040>

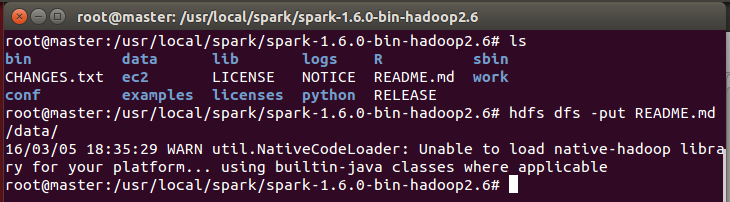
Environment信息：



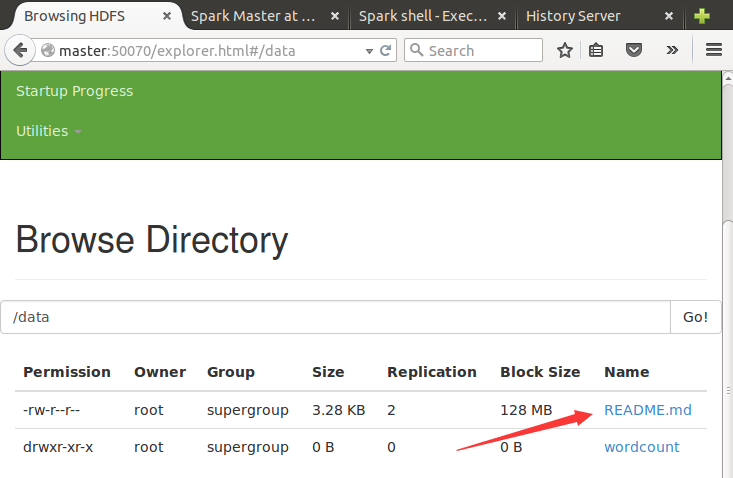
Executors信息：



1. 通过spark-shell测试SPARK集群:
2. 将spark安装包下的README.md上传到hdfs: hdfs dfs -put README.md /data/

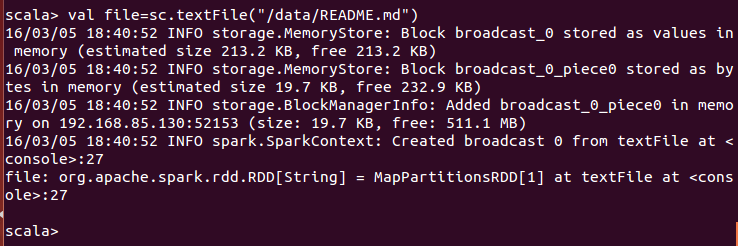


1. 通过hdfs的web控制台查看可见上传成功：

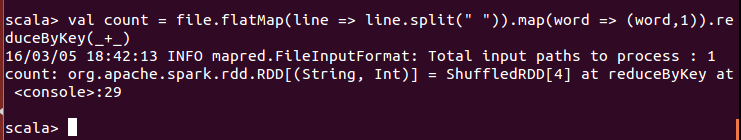


1. 在spark-shell中对上传的文件进行操作：

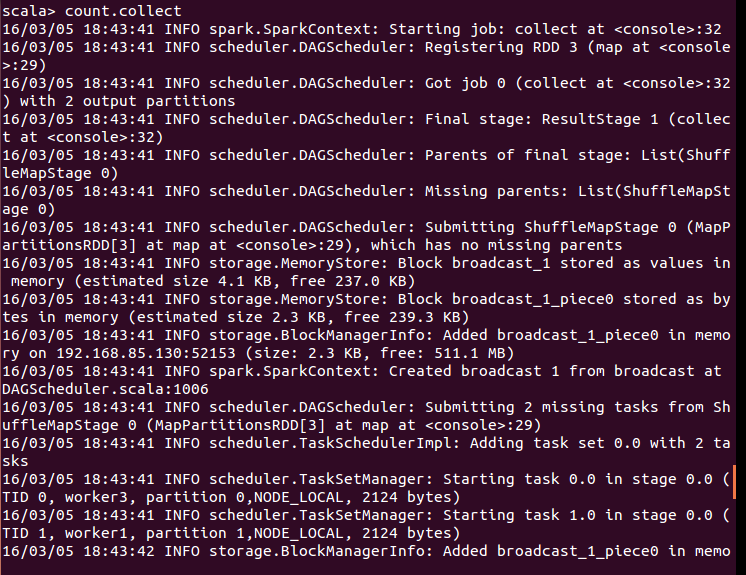
读取文件：val file=sc.textFile("/data/README.md")

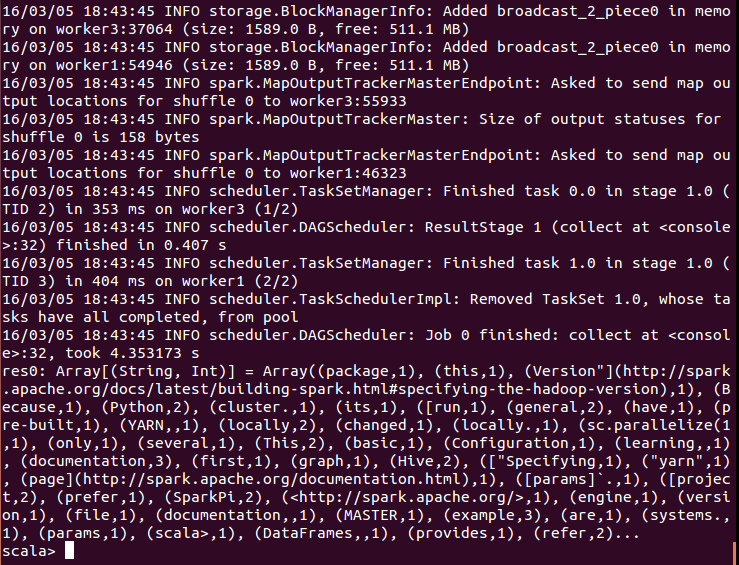


对读取的文件进行count操作：val count = file.flatMap(line => line.split(" ")).map(word => (word,1)).reduceByKey(\_+\_)

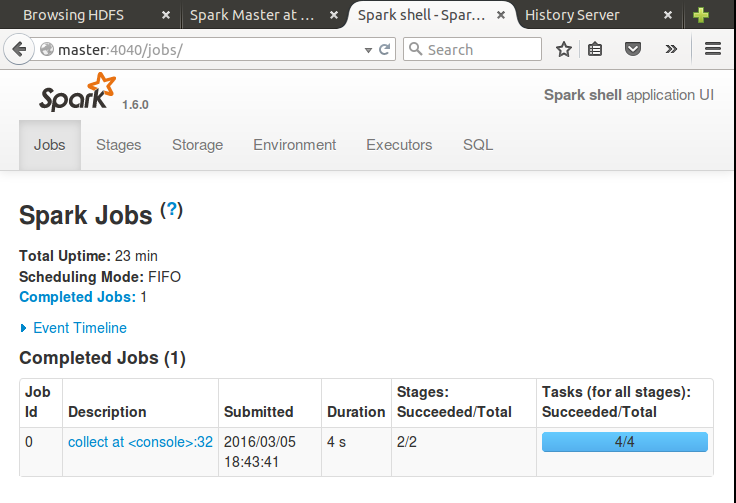


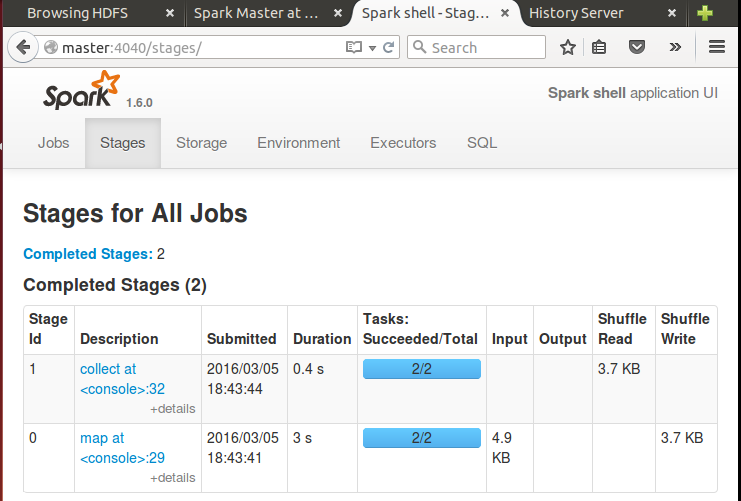
用collect命令提交并执行Job: count.collect



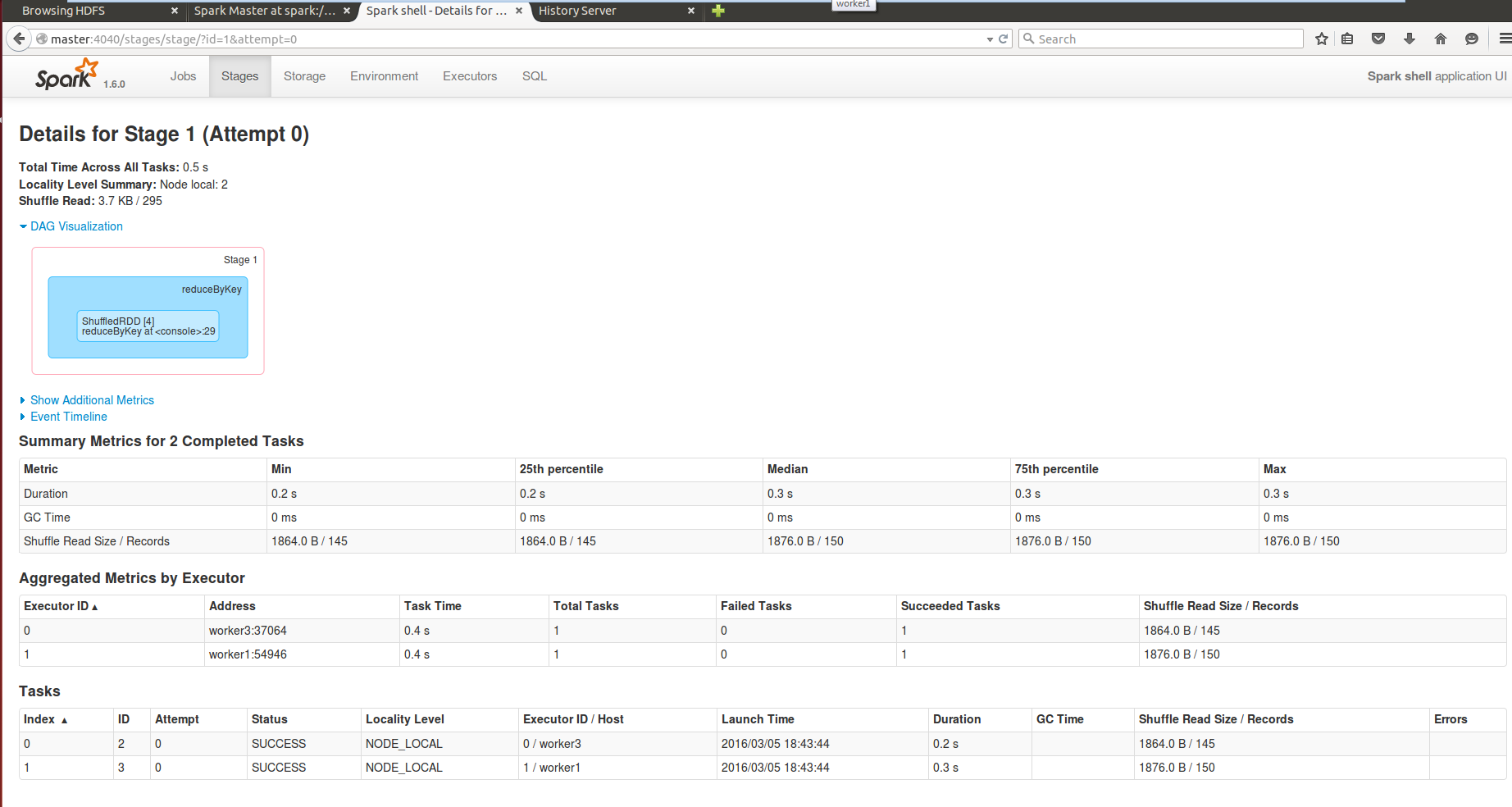


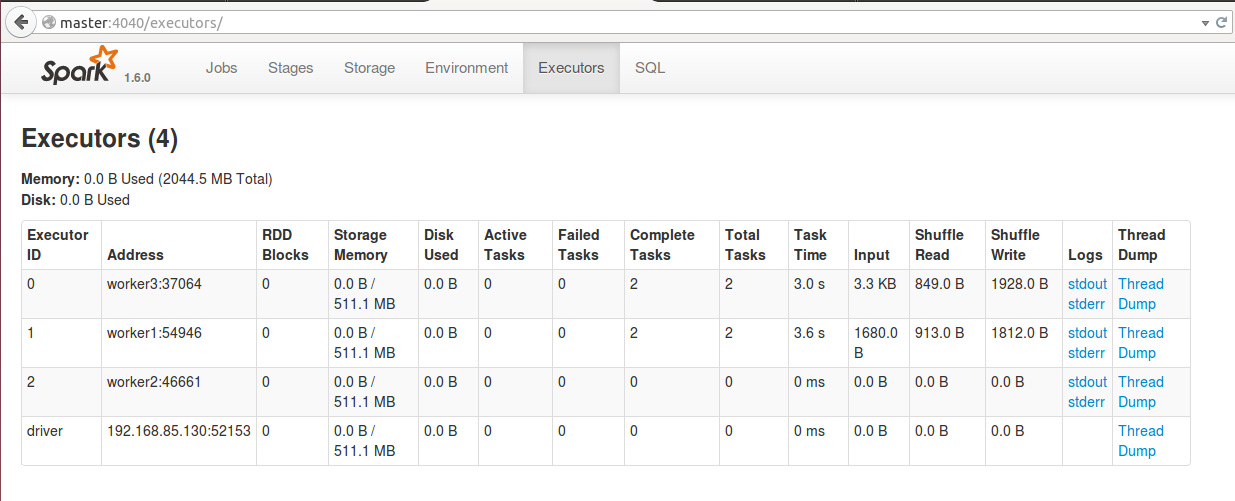
1. 从webUI可见job成功执行：<http://master:4040>





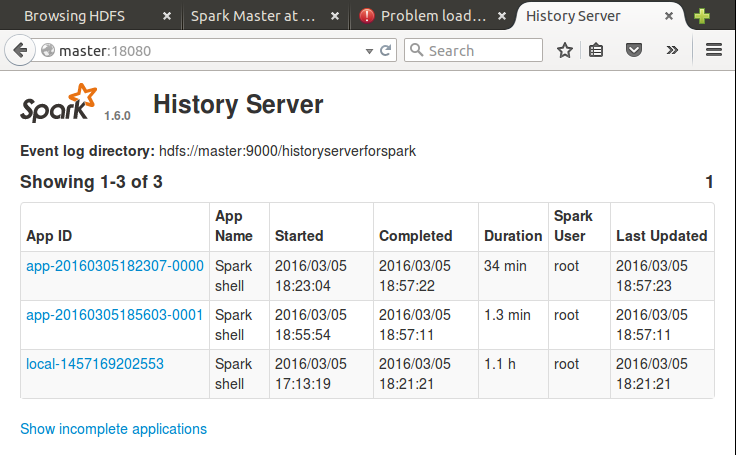
Stage1详细信息：





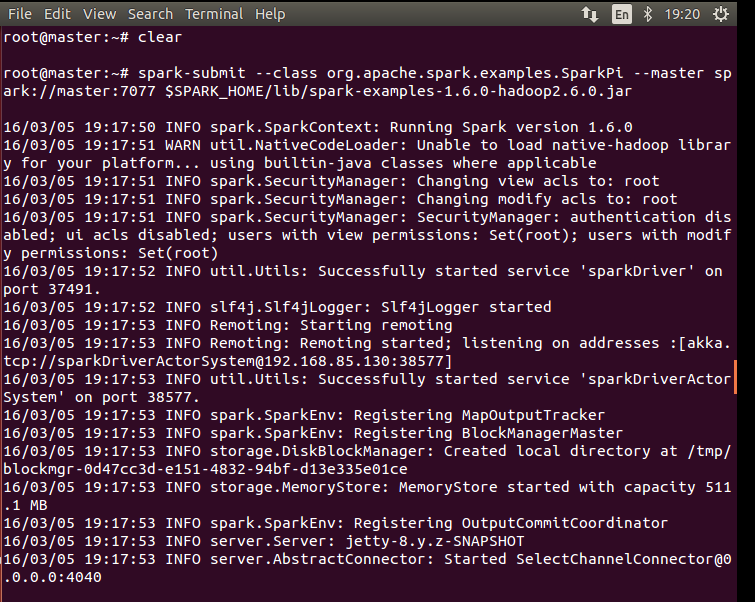
上述信息表明程序执行成功。

1. 通过jobhistoryserver可以查看程序执行的历史信息：<http://master:18080>

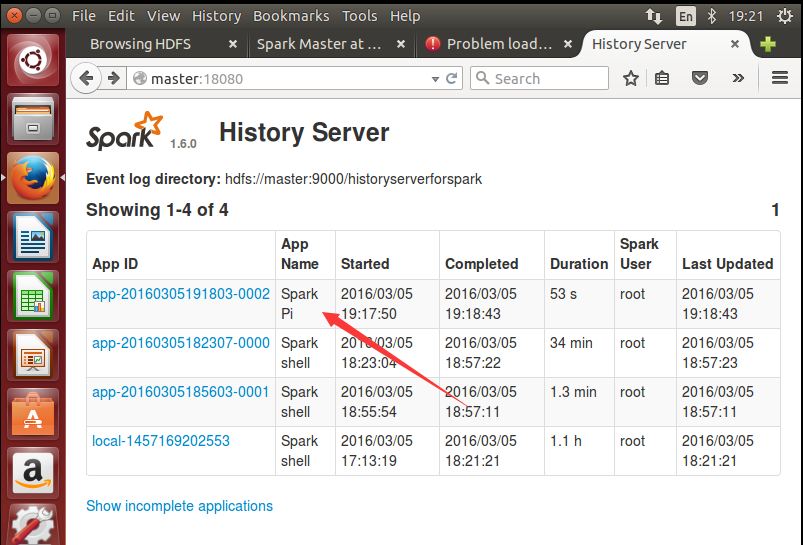


注：4040端口用来查看正在运行的程序的job信息，而18080用来查看执行完毕的程序的job信息。

1. 通过spark-submit提交程序到spark集群测试SPARK集群:
2. 通过以下命令提交spark自带的sparkPi程序：spark-submit --class org.apache.spark.examples.SparkPi --master spark://master:7077 $SPARK\_HOME/lib/spark-examples-1.6.0-hadoop2.6.0.jar



1. webUI查看：



注意：spark-submit更详细的信息请参考官方文档：

<http://spark.apache.org/docs/latest/submitting-applications.html>

1. 附录：常见错误与解决方法：
2. **错误**：安装 ubuntu时 出现错误:This kernel requires an X86-64 CPU, but only detected an i686 CPU错误。**解决方法**：安装64位的系统需要64位的cpu且cpu要允许硬件虚拟化。首先查看cpu是否是64 位，是的话去bios里, 在cpu configuration中找到Virtualization，将其状态改为enabled.
3. **错误**：E212:can't open file for writing。**解决方法**：权限不够，切换到ROOT.
4. **错误**：sudo时报错: unable to change to root gid: Operation not permitted**解决方法**：这是因为我们使用guest角色登录系统了，切换用户，使用普通用户登录系统后即可使用sudo命令。
5. **错误**：执行下面命令是报错：

root@worker1:~# scp id\_rsa.pub root@master:/root/.ssh/id\_rsa.pub.slave1

root@worker1's password: Permission denied, please try again.

root@worker1's password: Permission denied, please try again.

root@worker1's password: Permission denied (publickey,password).lost connection

**解决方法**：登录到scp的目标机，将/etc/ssh/sshd\_config 中注释掉 #PermitRootLogin without-password，添加 PermitRootLogin yes,重启 ssh服务即可。（sudo service ssh restart.）

1. 错误:启动spark history server时报错：

Exception in thread "main" java.lang.reflect.InvocationTargetException

at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstruct

orAccessorImpl.java:62) at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingC

onstructorAccessorImpl.java:45)at java.lang.reflect.Constructor.newInstance(Constructor.java:422)

at org.apache.spark.deploy.history.HistoryServer$.main(HistoryServer.sca

la:235)at org.apache.spark.deploy.history.HistoryServer.main(HistoryServer.scal

a)Caused by: java.lang.IllegalArgumentException: Log directory specified does not exist: file:/tmp/spark-events. Did you configure the correct one through spark.history.fs.logDirectory?

**解决方法**:在启动spark history server时需要在命令行动态指定目录,或提前配置好相关参数。以spark.history开头的需要配置在spark-env.sh中的SPARK\_HISTORY\_OPTS，以spark.eventLog开头的配置在spark-defaults.conf。

可以在spark-defaults.conf中添加如下配置：

  spark.eventLog.enabled           true

  spark.eventLog.dir               hdfs://spark/jobhistory

  spark.history.fs.logDirectory    hdfs://spark/jobhistory