**二、各类配置**

**①基础环境配置：**

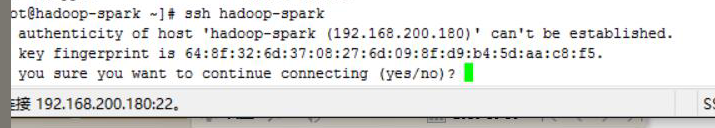
**（1）ssh的配置**

1、ssh-keygen -t rsa 生成一个私钥id\_rsa，和一个公钥id\_rsa.pub，在文件/root/.ssh/目录下

如果系统中已经生成了密钥，但是自己的主机ip，或者hostname改变的时候，需要重新生成，将原来的.ssh目录删除

2、cp ~/.ssh/id\_rsa.pub ~/.ssh/authorized\_keys 这一步是将公钥拷贝到.ssh目录下，并且新建一个文件authorized\_keys，将公钥写进去

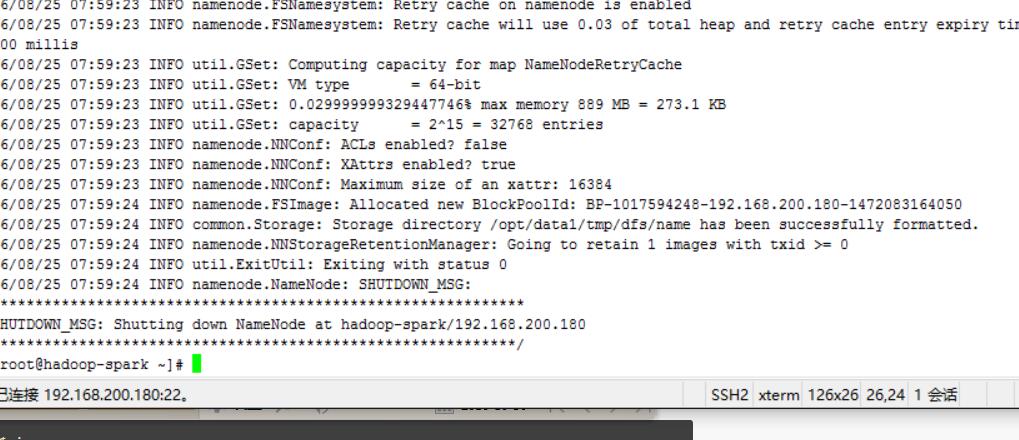
3、分别第一次用命令ssh localhost以及自己主机名ssh hadoop-spark来登录，在显示



后，输入虚拟机的登录密码，两次的第一次登录都需要输入，然后选yes即可.

**（2）namenode的formate**

成功的话显示如下：

****

**（3）各类配置文件的具体参数：**

**Core-site.xml:**

<configuration>

<property>

<name>fs.default.name</name>

<value>hdfs://hadoop-spark:9000</value>

</property>

<property>

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

<value>/opt/data1/tmp</value> (在重新format namenode时只需要把此文件目录修改

</property> 一下即可，不用删除其他东西)

</configuration>

**Hdfs-site.xml:**

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.permissions</name>

<value>false</value>

</property>

</configuration>

**Mapred-site.xml:**

<configuration>

<property>

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

<value>hadoop-spark:9001</value>

</property>

</configuration>

Hadoop的**masters**：

hadoop-spark

hadoop的**slaves**：

hadoop-spark

spark的**masters**：

hadoop-spark

spark的**spark-defaults.conf**可以不用配置

spark的**spark-env.sh**

export SCALA\_HOME=~/user/scala-2.10.4

export JAVA\_HOME=/root/user/jdk/jdk1.8.0\_101

export SPARK\_MASTER\_IP=hadoop-spark

export SPARK\_MASTER\_PORT=7077

export SPARK\_MASTER\_WEBUI\_PORT=8080

export SPARK\_WORKER\_CORES=1

export SPARK\_WORKER\_PORT=7078

export SPARK\_WORKER\_WEBUI\_PORT=8081

export SPARK\_WORKER\_INSTANCES=1

**etc/profile:**

export JAVA\_HOME=~/user/jdk/jdk1.8.0\_101

export JRE\_HOME=${JAVA\_HOME}/jre

export CLASSPATH=.:${JAVA\_HOME}/lib:${JRE\_HOME}/lib

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

export SCALA\_HOME=~/user/scala-2.10.4

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

export HADOOP\_HOME=~/user/hadoop-2.5.2

export HADOOP\_COMMON\_HOME=$HADOOP\_HOME

export HADOOP\_hadoop\_HOME=$HADOOP\_HOME

export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME

export HADOOP\_YARN\_HOME=$HADOOP\_HOME

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

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

**网络**

**sysconfig/network:**

NETWORKING=yes

HOSTNAME=hadoop-spark

**sysconfig/hosts:**

127.0.0.1 localhost

192.168.200.180 hadoop-spark （这里一个ip只能对应一个hostname，多个可能会出错）

**C:\Windows\System32\drivers\etc\HOSTS**

192.168.200.180 hadoop-spark

**修改时间：**

[root@hadoop-spark bin]# date -s 11/01/2016

2016年 11月 01日 星期二 00:00:00 CST

[root@hadoop-spark bin]# date -s 10:41:00

2016年 11月 01日 星期二 10:41:00 CST

1. **测试各个端口：**

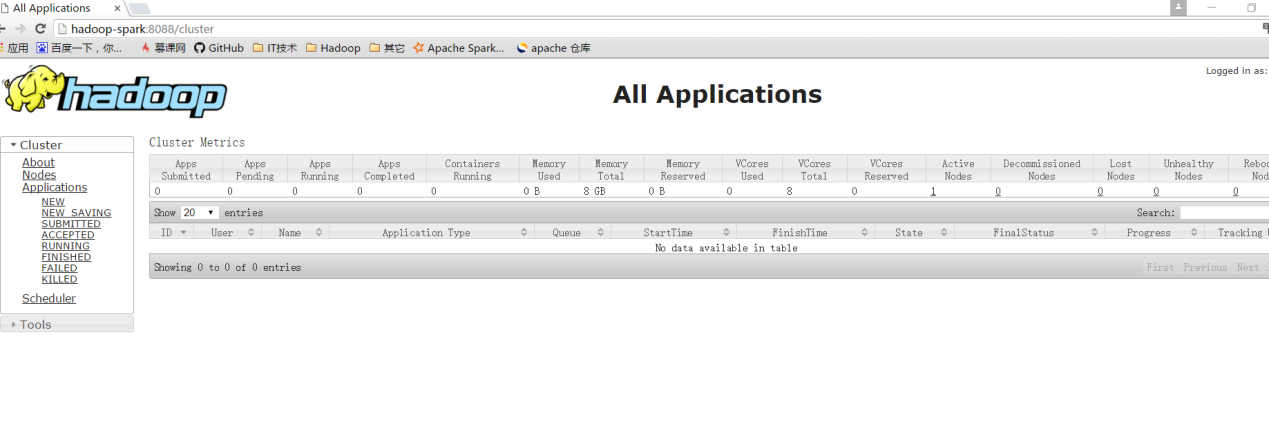
用于访问和监控[**Hadoop**](http://lib.csdn.net/base/20)系统运行状态

**namenode的webUI端口：50070**   
**yarn的web端口：8088**

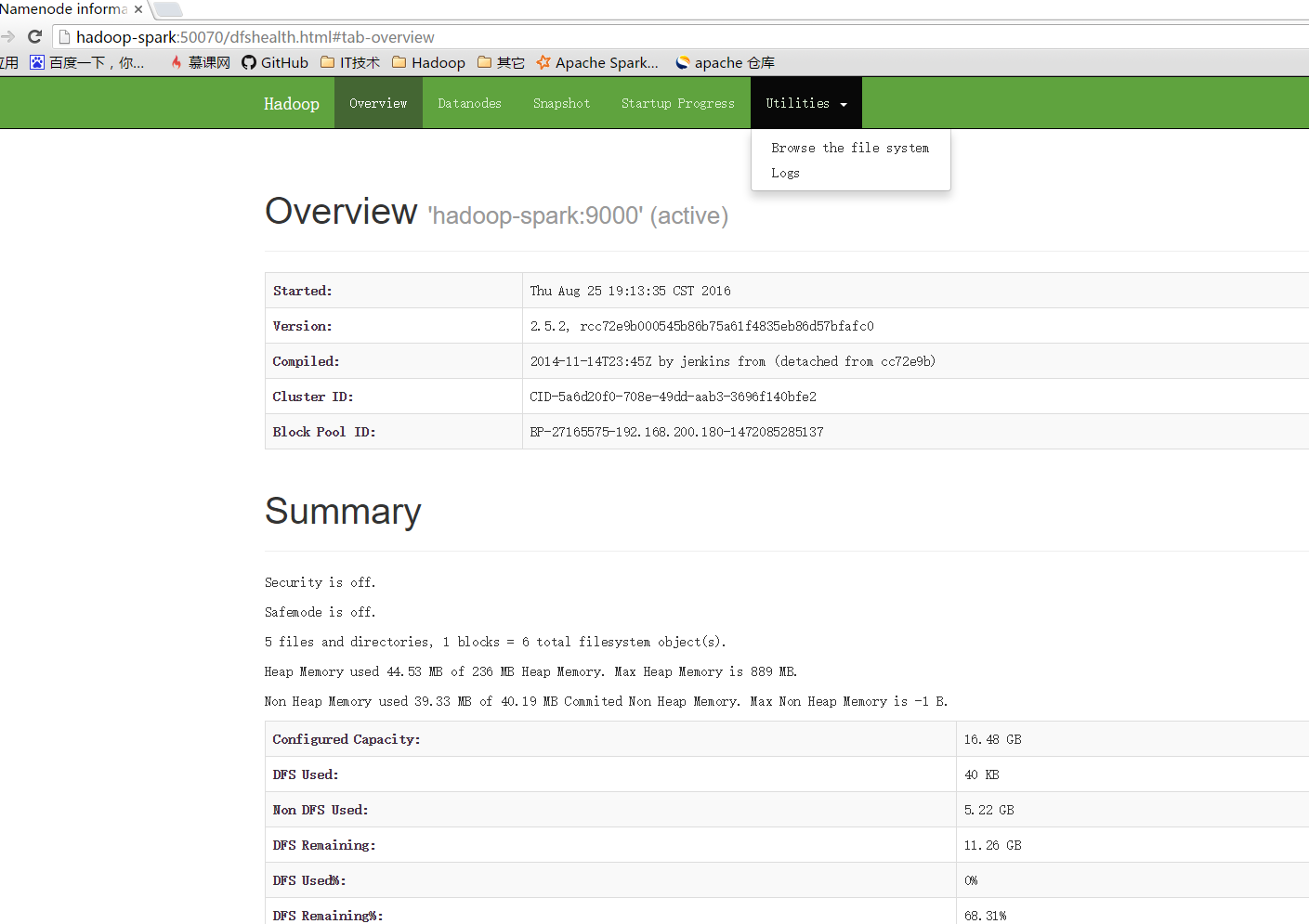
  
**spark集群的web端口：8080**   
**spark-job监控端口：4040**

**（单个具体的job的端口，如果被占用，会启用下一个4041…）**

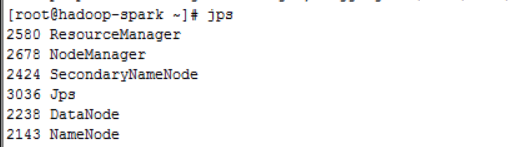
**Hadoop的8088：**



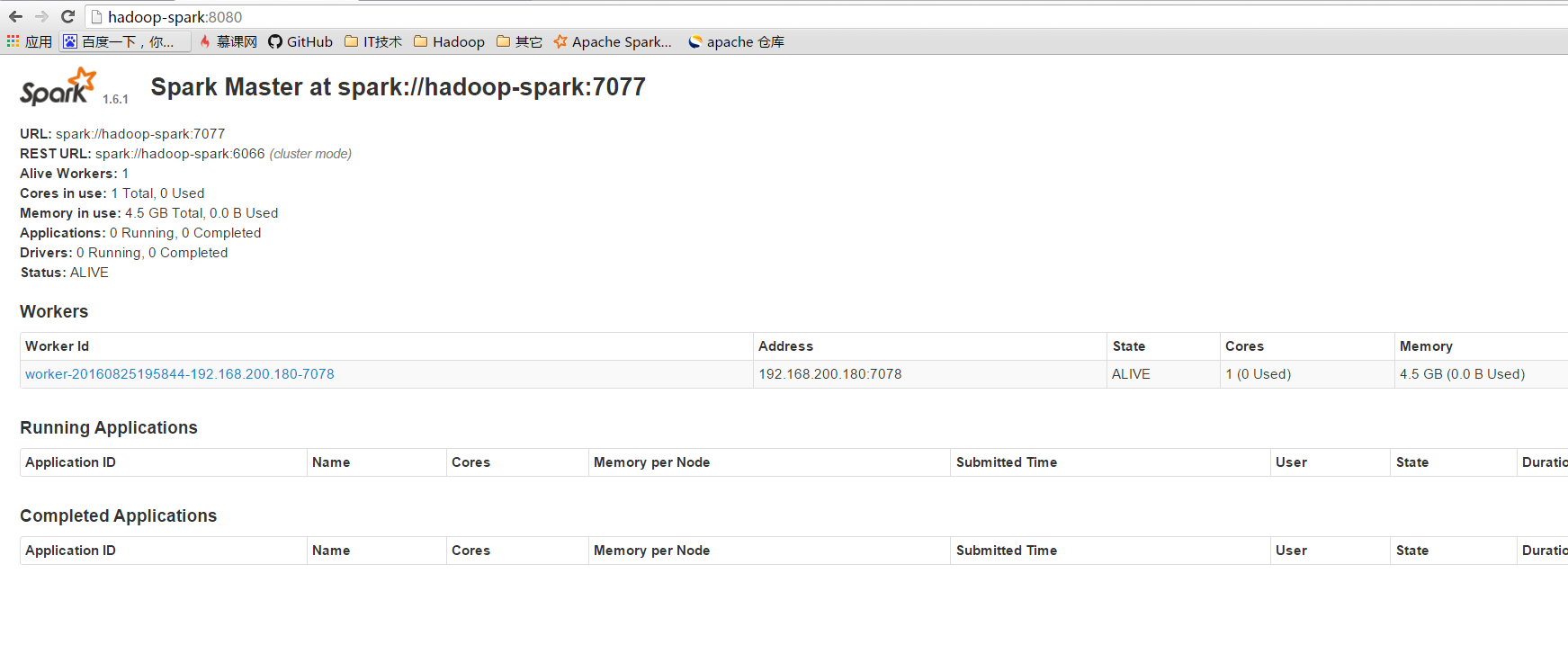
**Hadoop的50070：**



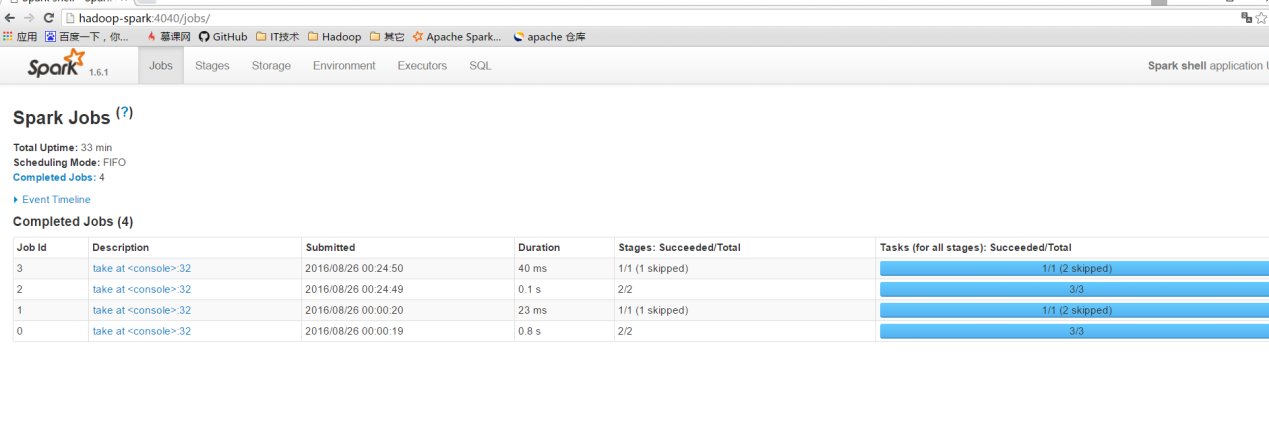
查看进程

如上所示，如果都测试成功，表示hadoop的配置已经完成。

**Spark的8080：**

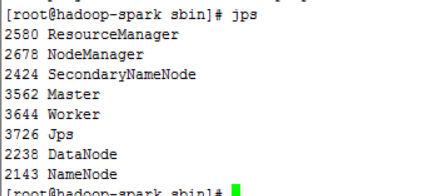
****

**Spark的4040端口：**

****

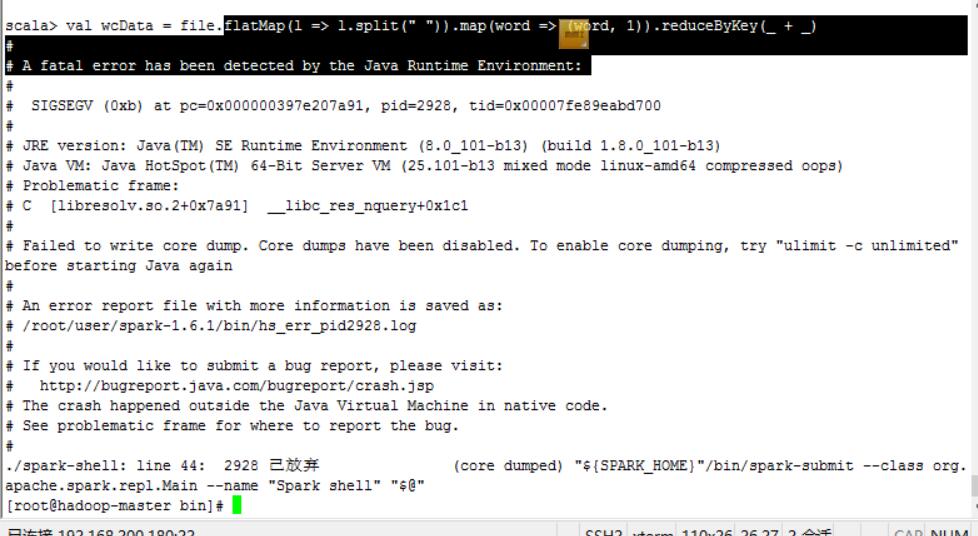
（spark的7077可以不用管）：

然后jps的端口如下：



至此，hadoop和spark环境搭建基本结束，以下是各类错误总结：

**错误（1）**



原因：

val rdd=sc.textFile("http://hdfs://hadoop-spark:9000/one")

多加了http

**错误（2）**

自己编译spark的各种错误，解决办法，刻意去网上根据自己的hadoop版本，找已经编译好的的（不用管hive的或者其他的），然后解压即可。

**错误（3）**

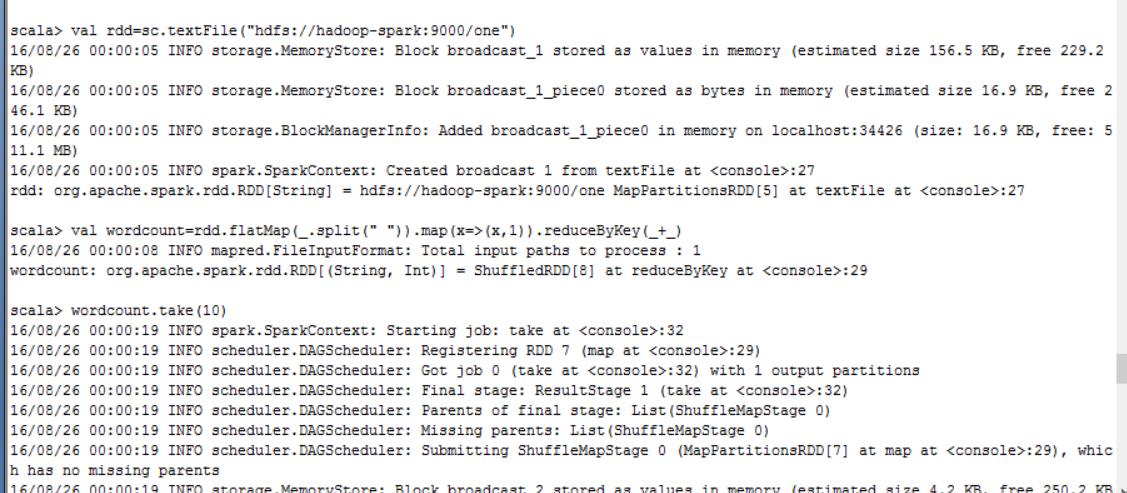
****

原因：val rdd=sc.textFile("hdfs://hadoop-spark:50070/one")

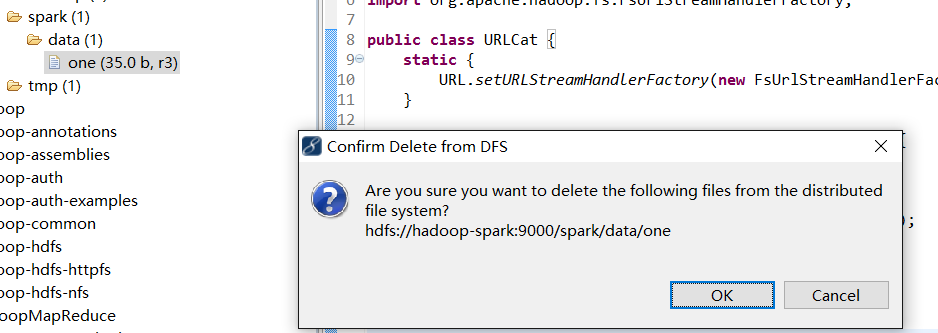
端口号写成了50070

正确的代码及显示如下：

val rdd=sc.textFile("hdfs://hadoop-spark:9000/one")



至于如何查看文件目录，可以通过myeclipse查看：



**其他错误及解决办法：**

**安装部署Spark 1.x Standalone模式集群**

【问题：

输入：./spark-shell

Java HotSpot(TM) 64-Bit Server VM warning: INFO: os::commit\_memory(0x00000000c0000000, 357892096, 0) failed; error='Cannot allocate memory' (errno=12)】

内存不足：

free -h

total used free shared buffers cached

Mem: 5.5G 312M 5.2G 244K 16M 160M

-/+ buffers/cache: 135M 5.4G

Swap: 999M 0B 999M

[root@hadoop-master ~]# start-all.sh

This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh

Starting namenodes on [hadoop-master.dragon.org]

hadoop-master.dragon.org: starting namenode, logging to /root/user/hadoop-2.5.2/logs/hadoop-root-namenode-hadoop-master.out

hadoop-master.dragon.org: starting datanode, logging to /root/user/hadoop-2.5.2/logs/hadoop-root-datanode-hadoop-master.out

Starting secondary namenodes [0.0.0.0]

0.0.0.0: starting secondarynamenode, logging to /root/user/hadoop-2.5.2/logs/hadoop-root-secondarynamenode-hadoop-master.out

starting yarn daemons

starting resourcemanager, logging to /root/user/hadoop-2.5.2/logs/yarn-root-resourcemanager-hadoop-master.out

hadoop-master.dragon.org: starting nodemanager, logging to /root/user/hadoop-2.5.2/logs/yarn-root-nodemanager-hadoop-master.out

[root@hadoop-master ~]# free -h

total used free shared buffers cached

Mem: 5.5G 967M 4.6G 244K 16M 178M

-/+ buffers/cache: 771M 4.8G

Swap: 999M 0B 999M

[root@hadoop-master ~]# free -h

total used free shared buffers cached

Mem: 5.5G 2.4G 3.1G 248K 20M 263M

-/+ buffers/cache: 2.1G 3.4G

Swap: 999M 0B 999M

[root@hadoop-master ~]#

【textFile.count默认读取的是/root/user/hadoop-2.5.2/下的REMADE.md】

可以在spark-env.sh设置自定义的读取位置：

export HADOOP\_CONF\_DIR=/root/user/hadoop-2.5.2/etc/hadoop

可能出现错误：Input path does not exist:

【运行textFile.count()报错 】

16/02/04 11:11:57 WARN scheduler.TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources

16/02/04 11:12:12 WARN scheduler.TaskSchedulerImpl: Initial job has not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources

原因：spark-env.sh中参数设置有问题：

无需设置：

export SPARK\_WORKER\_MEMORY=1000m

直接采用默认的即可，全部都用默认值

export SPARK\_MASTER\_IP=hadoop-master.dragon.org

export SPARK\_MASTER\_PORT=7077

export SPARK\_MASTER\_WEBUI\_PORT=8080

export SPARK\_WORKER\_CORES=1

export SPARK\_WORKER\_PORT=7078

export SPARK\_WORKER\_WEBUI\_PORT=8081

export SPARK\_WORKER\_INSTANCE

SPARK\_EXECUTOR\_MEMORY=200M

SPARK\_DRIVER\_MEMORY=200M

export SPARK\_WORKER\_MEMORY=1000m

网上的解释：

集群环境，每个节点只剩下500MB内存了，由于没有配置SPARK\_EXECUTOR\_MEMORY参数，默认会使用1G内存，所以会出现内存不足，从而出现上面日志报的警告信息。

所以解决办法是添加如下参数：export SPARK\_EXECUTOR\_MEMORY=100M

SPARK\_WORKER\_MEMORY一共才512M，这些--executor-memory、--driver-memory 指定得更小些（比如50M、100M）？

尤其是--driver-memory，应该用不着512M这么大。

spark.driver.memory

默认：512m

driver进程使用的内存数

spark.executor.memory

默认：512m

每个executor进程使用的内存数。和JVM内存串拥有相同的格式（如512m,2g）

**②**[**Spark开发环境配置(windows/Intellij IDEA 篇)**](http://blog.csdn.net/xiaoshunzi111/article/details/52538521)

### 配置开始

1. 安装IDEA 的scala插件

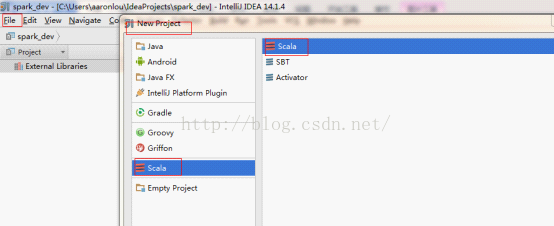
  可能存在的问题：在网上下载的插件，在IDEA导入时出现：

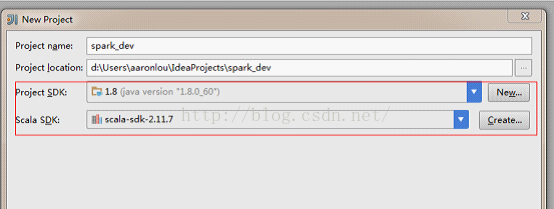
Plugin 'Scala' is incompatible with this installation

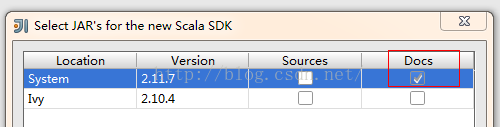
是因为此版本的IDEA不适合该插件，可以到官网上查看该版本支持的插件

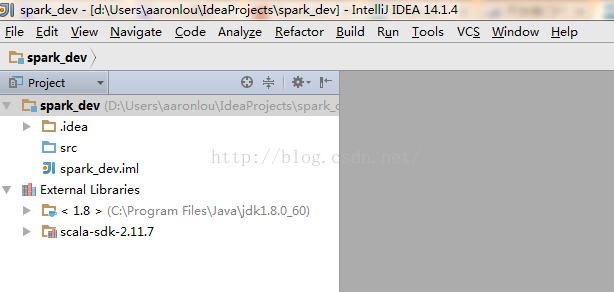
2. 创建项目并导入相应依赖包

    如下图指引，新建项目时，选择scala，然后点击“下一步”，这里Project SDK是java的jdk，如果没有默认加载出来，点击New，手工定位到jdk的目录提交上来即可。Scala SDK那里如果默认没有加载出来，点击Create，在弹出的窗口中安默认勾选的System点击OK即可。





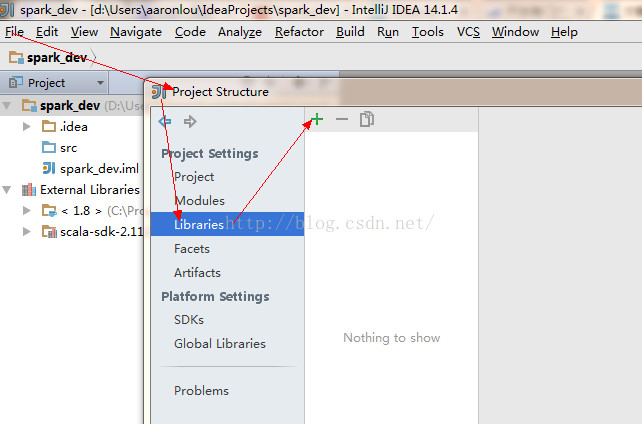


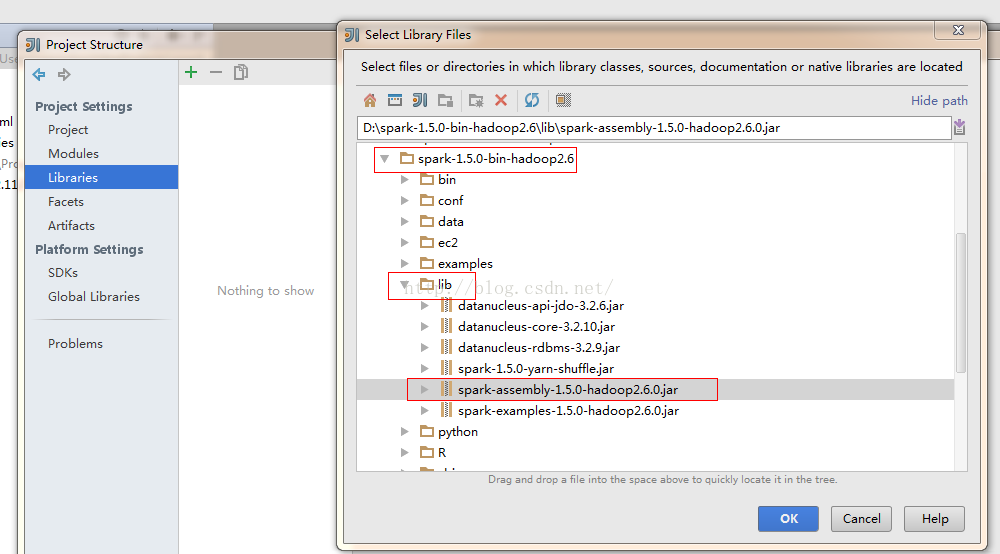


接下来，我们导入上面下载好的spark源码。按下图指引操作, 在+号处选择java, 然后定位到你上面步骤中将spark程序解压到的目录位置，选择lib目录下的spark-assembly-1.5.0-hadoop2.6.0.jar文件，确认。

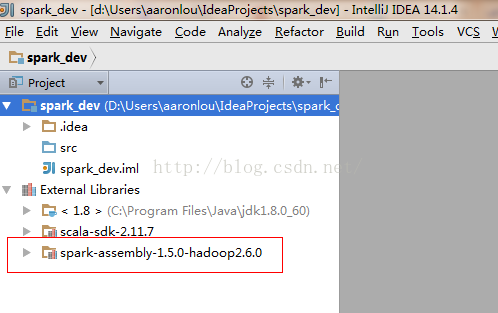
**其中源码包为：spark-1.6.1-bin-hadoop2.4**

**该文件在spark-1.6.1-bin-hadoop2.4\lib\ spark-assembly-1.6.1-hadoop2.4.0.jar目录下**





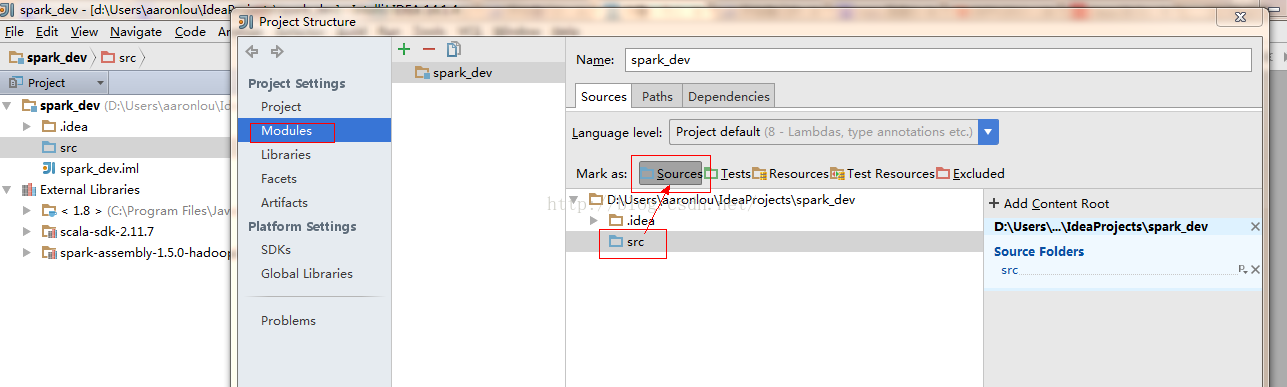
此时，俺的项目依赖的外部库文件中，就多出了spark源码这个文件，如下图所示：



### 程序开发

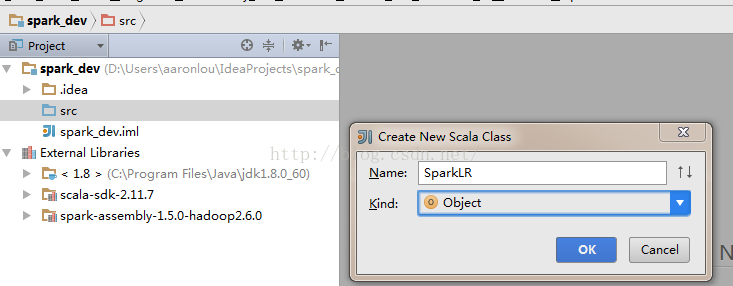
    搞好了基础配置，就开始coding吧。

    这里先提几个细节。如果你发现在src目录上右键找不到scala class这样的文件创建选项的话，说明src这个文件夹的属性不对，一定要确认下图所示src被选中为Sources文件夹属性。

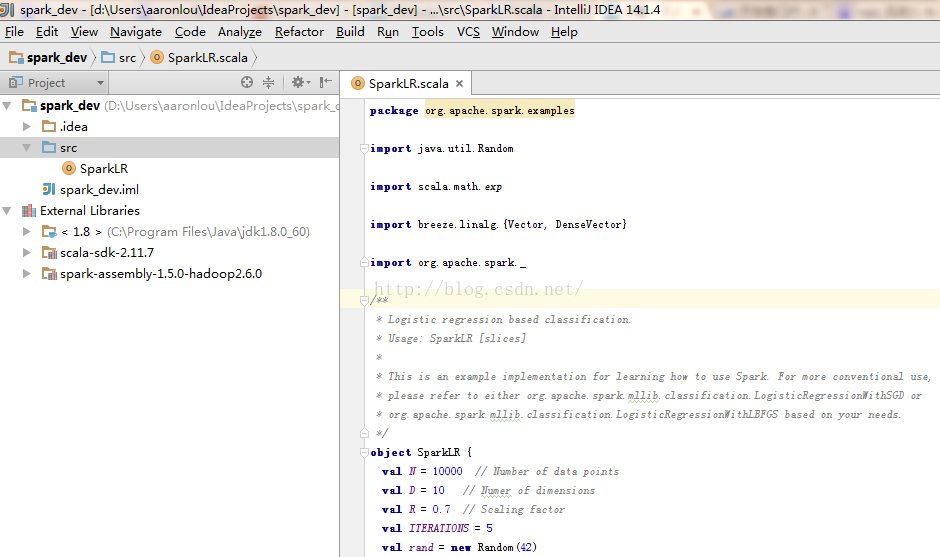


第2个细节，这里创建类文件时，一般大家喜欢按目录结构来，比如创建main/scala/或main/java等这样的子文件夹，当然这样方便了代码管理，但是笔者在这样操作时，后面编译文件时就报错了。google半天，发现不少人遇到同样问题，删除这些子文件夹，直接在src下创建类文件，就OK了。

     finally, 终于进入编码环节了。本文为示例方便，直接拿spark源码中的example程序-SparkLR来讲解（这些案例程序位于你的spark解压目录\examples\src\main\scala\org\apache\spark\examples）。在src文件夹上右击选择scala class， Kind这里选择Object，然后把上述案例程序粘贴进去。



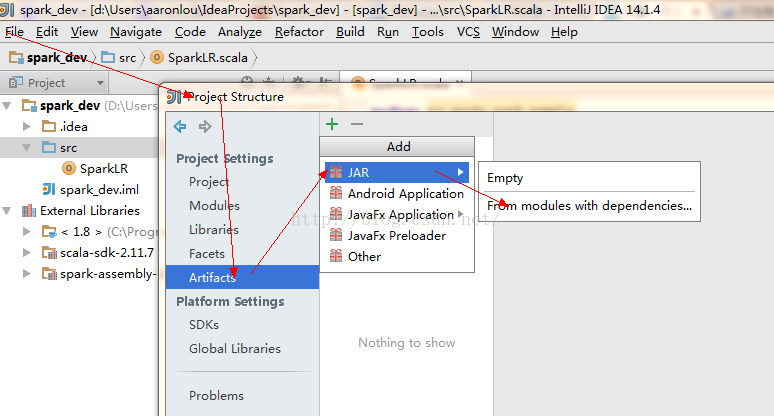
截止目前，我们的项目长成如下这个样子：

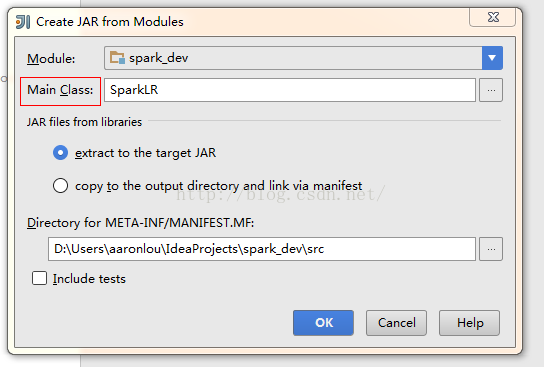


然后，还要再配置一下输出，也就是说一会你写的代码编译打包成jar包，放到哪里去，起个啥名字。操作如下：

**输出目录为：C:\Users\2281444815\IdeaProjects\untitled\out\artifacts\untitled\_jar**

**可以将untitled.jar的后缀该为.zip，即可查看内部的文件**

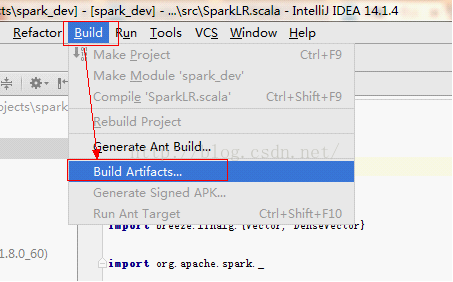


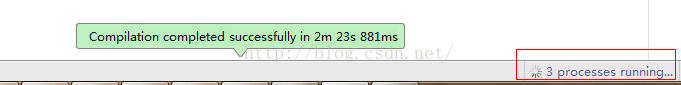


点击OK后，就配置好了输出形式。

### 编译&打包

    build -> build artifacts -> spark\_dev:jar -> build, 然后就开始编译了，在最下面可以看到进行的状态。如果没有报错，就是成功啦。





到输出JAR包的目录下去看看，发现确实成功生成了一个jar文件。【特别提示：此处打好的jar包如果要提交到数平的Spark集群上运行，请打开此jar包文件，观察其中是否有一个scala的文件夹，删除之！否则可能与线上的scala版本冲突】

到此，整个配置、开发、编译过程就算成功完成了。至于提交运行，就不在本文讨论范围内了，有时间的话再撰文记录吧。

