虚拟机Ubuntu 16.04 server上spark集群的搭建

环境准备

- VMware Workstation 15 Pro
- Ubuntu 16.04 server
- Xshell5
- Xftp5
- jdk-8u211-linux-x64.tar.gz
- scala-2.11.12.tgz
- hadoop-2.7.7.tar.gz
- spark-2.3.3-bin-hadoop2.7.tgz

一、Ubuntu server的安装

1、

2、

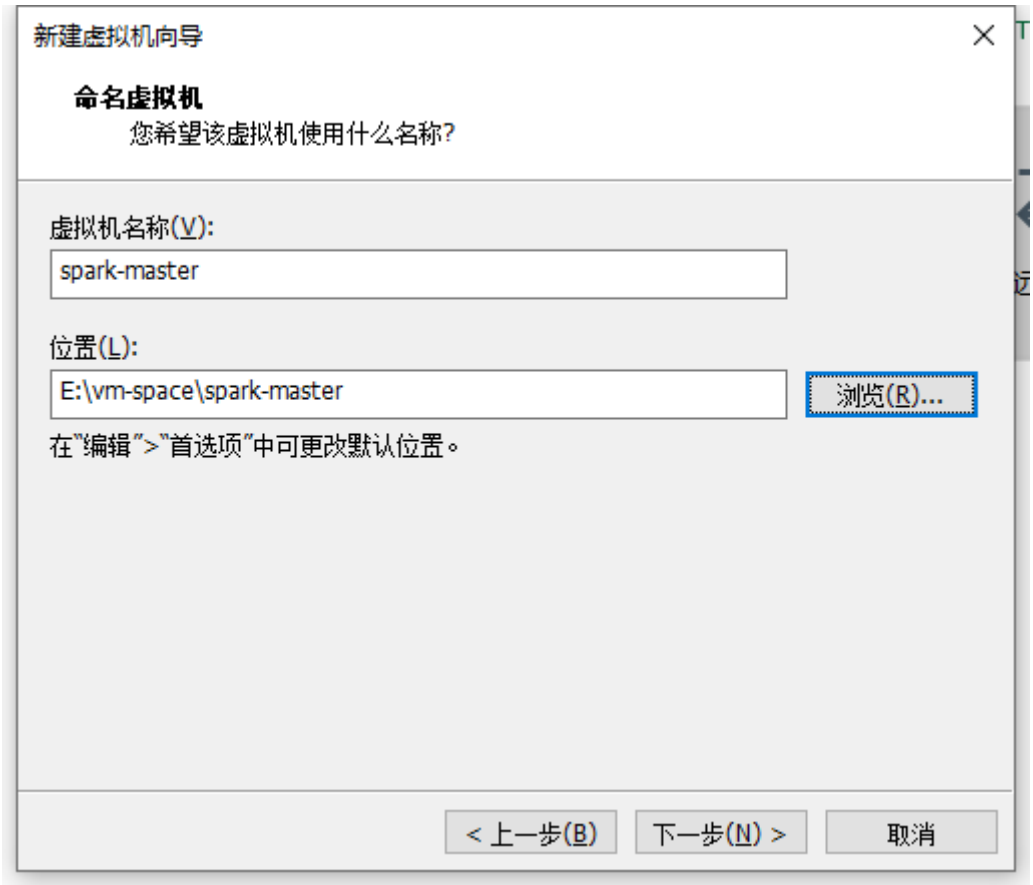

新建虚拟机向导
选择虚拟机硬件兼容性
该虚拟机需要何种硬件功能?
虚拟机硬件兼容性
硬件兼容性(H):
Workstation 15.x
兼容:
ESX Server(S)
兼容产品:
Fusion 11.x
Workstation 15.x
限制:
64 GB 内存
16 个处理器
10 个网络适配器
8 TB 磁盘大小
3 GB 共享图形内存
帮助
< 上一步(B)
下一步(N) >
取消

3、


新建虚拟机向导
安装客户机操作系统
虚拟机如同物理机，需要操作系统。您将如何安装客户机操作系统?
安装来源:
安装程序光盘(D):
DVD RW 驱动器 (F:)
安装程序光盘映像文件(iso)(M):
D:\迅雷下载\ubuntu-16.04.6-server-amd64.iso
浏览(R)...
已检测到 Ubuntu 64 位 Server 16.04.6。
该操作系统将使用简易安装。 (这是什么?)
稍后安装操作系统(S)。
创建的虚拟机将包含一个空白硬盘。
帮助
< 上一步(B)
下一步(N) >
取消

4、

5、



6、一直下一步

等待安装完成即可。

ps. 因为VMware和Ubuntu配合不是很好，中间如果有的地方卡了很长时间，建议重新安装（右键移除spark-master，再删除安装目录中所有内容）。

二、软件安装

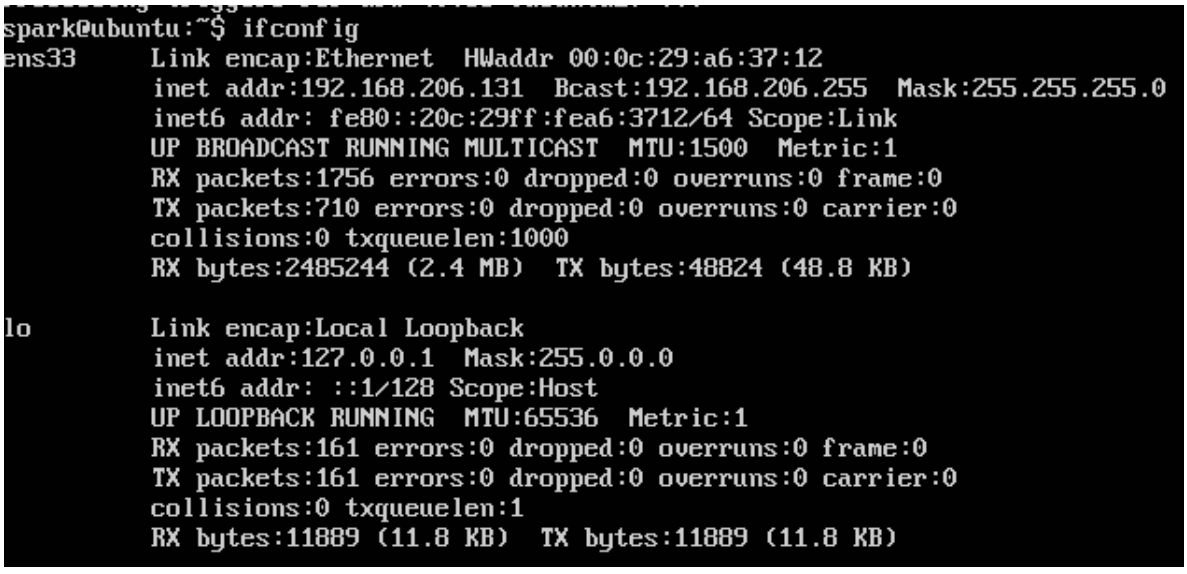
1、登录系统，安装ssh

在终端中输入命令：

```
sudo apt-get install openssh-server
```

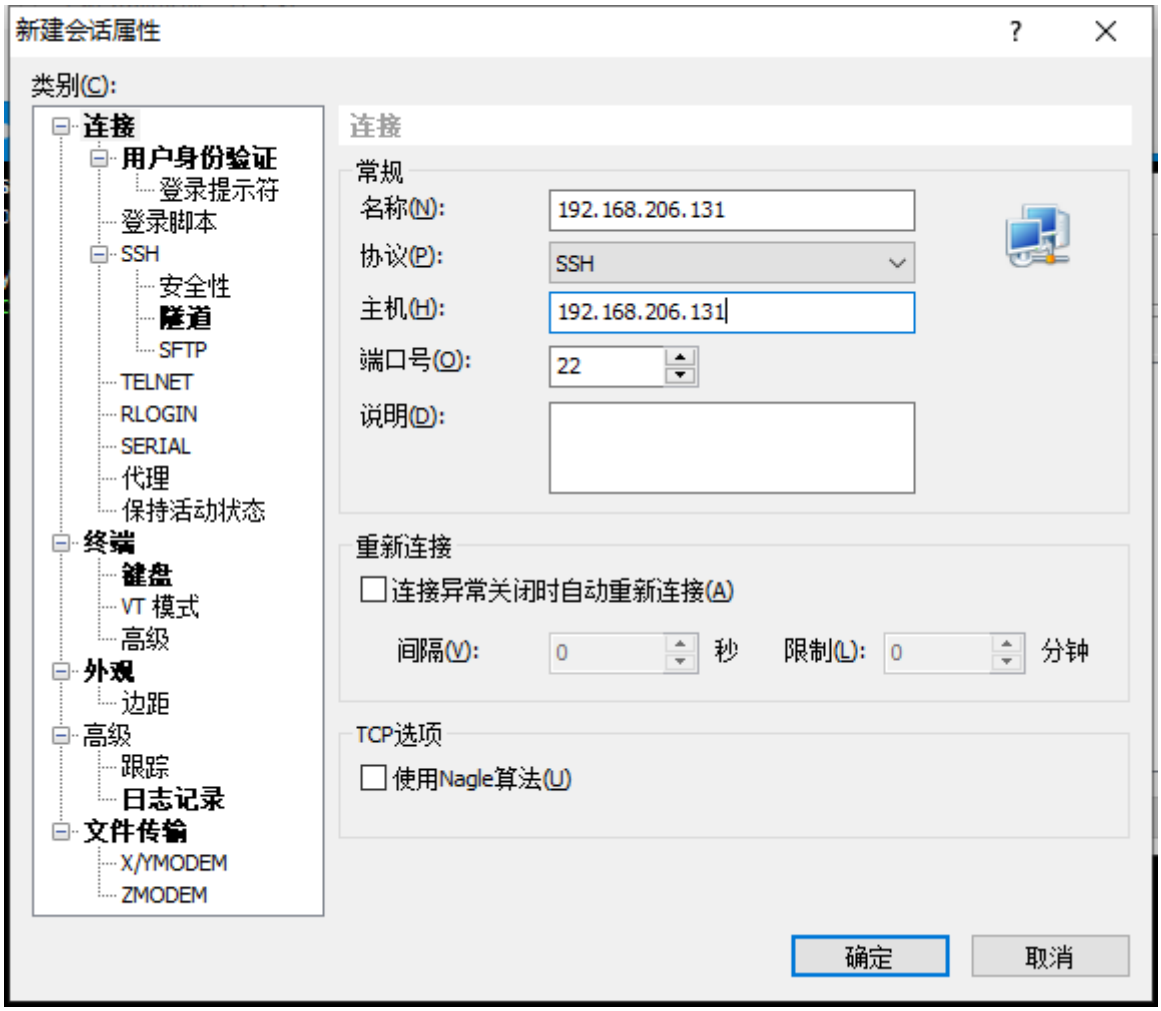
2、使用xshell连接

（1）输入ifconfig查看ip：

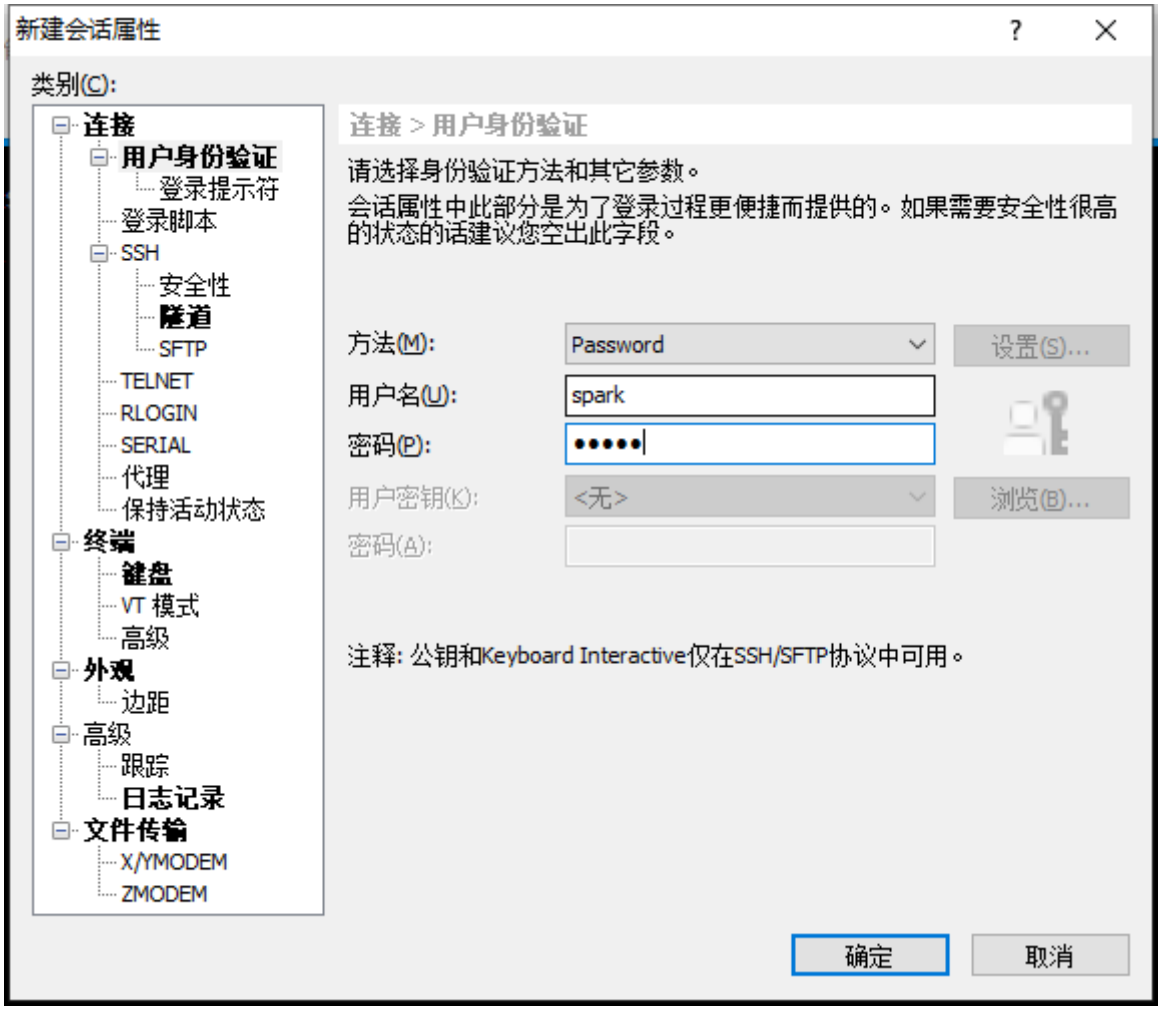


（2）打开xshell新建会话

在名称和主机输入ip（便于辨别）：

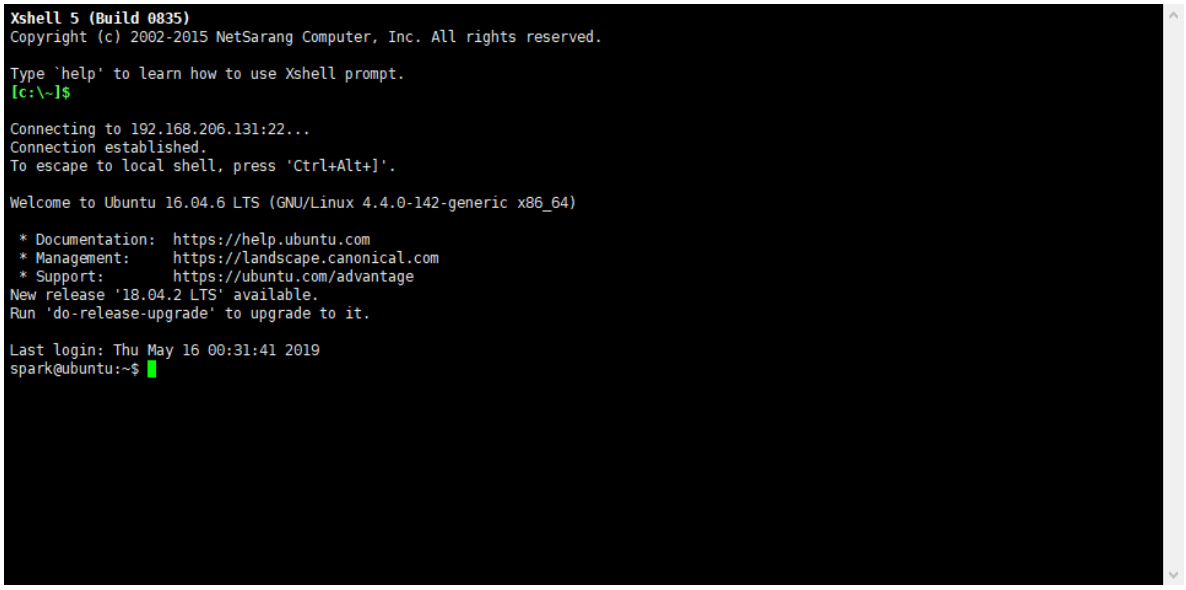


再点击**用户身份验证**，在用户名和密码中输入对应的账号密码：



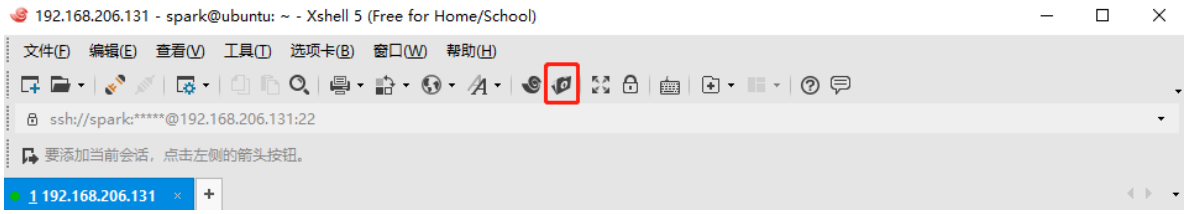
点击确定，再点击连接，再点击接受并保存：

出来这个界面表示已经连接上了：

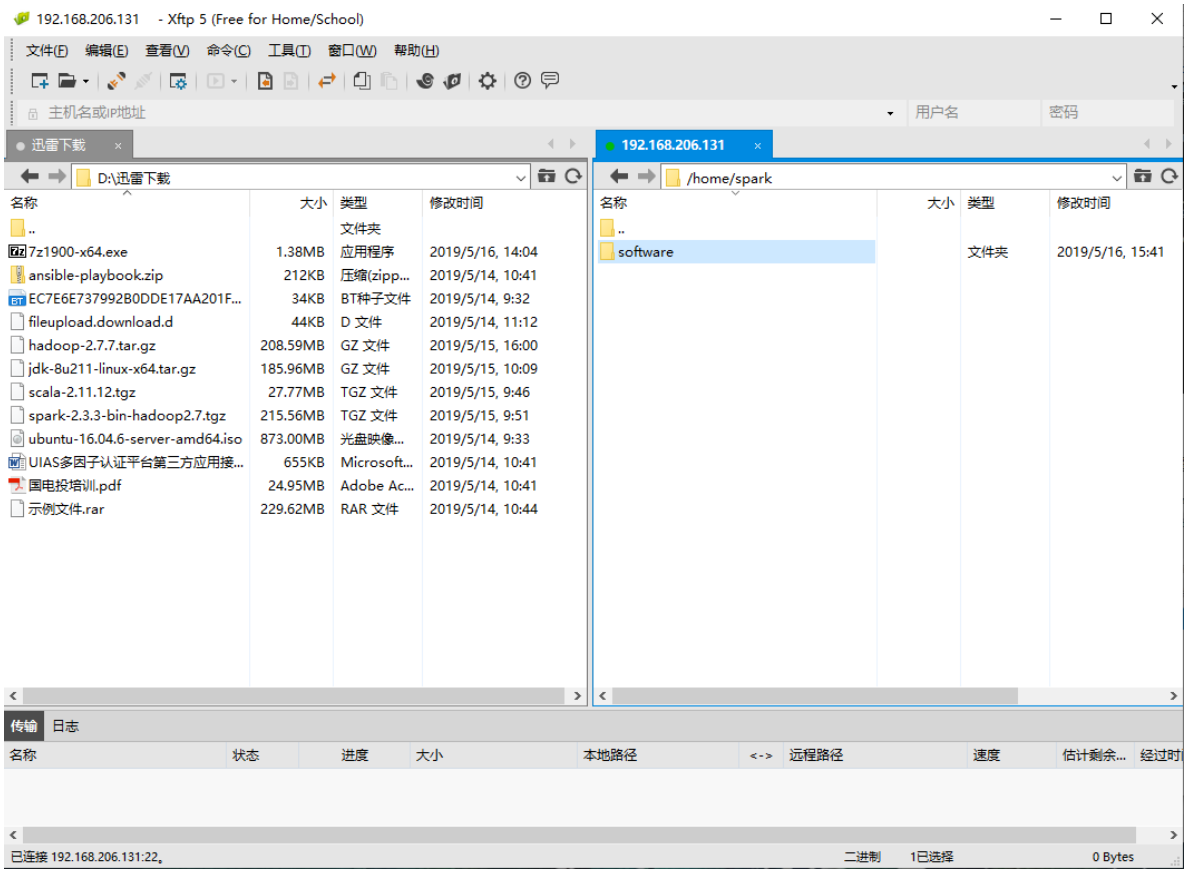


3、传输文件

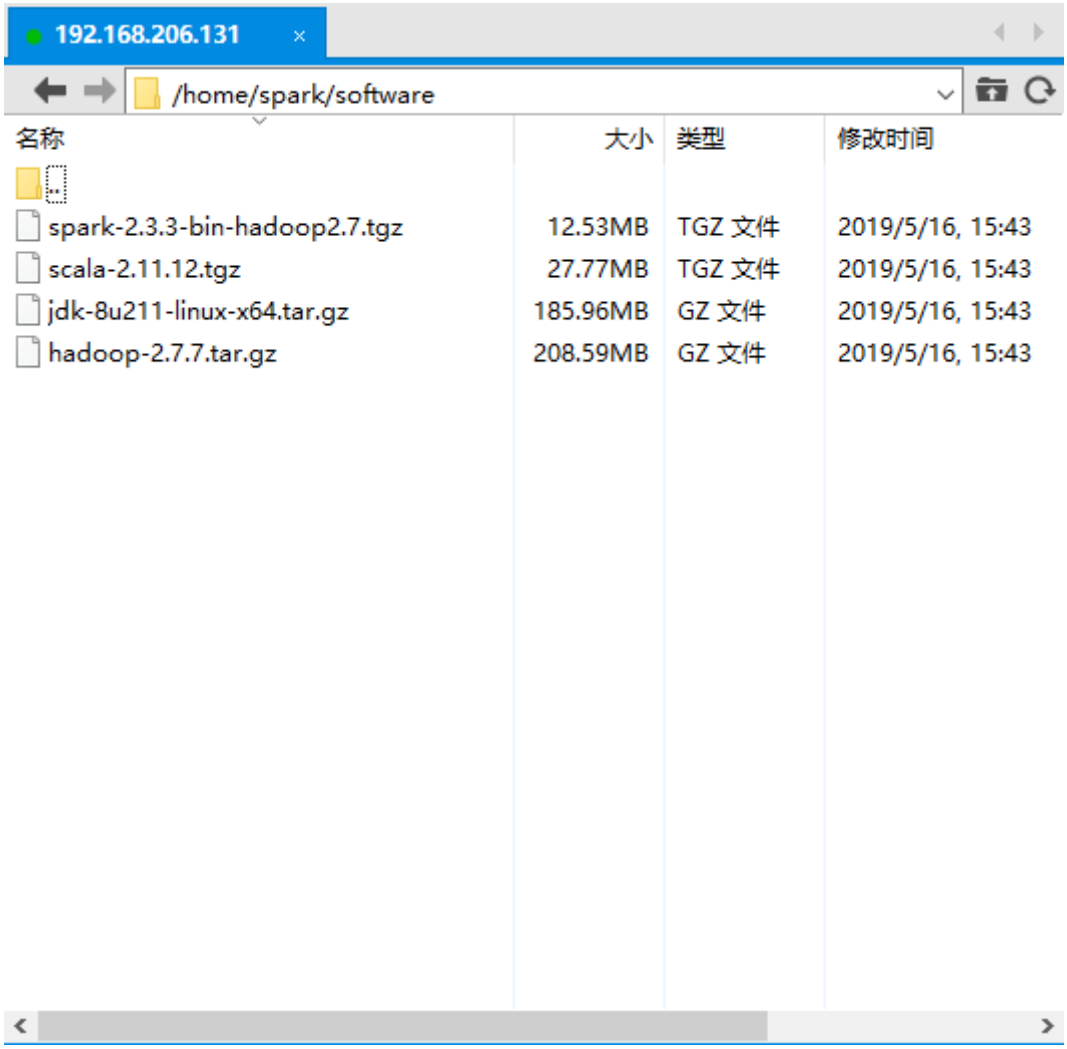
（1）点击下面这个图标，打开xftp：



（2）在右侧中右键新建文件夹software：



（3）点击进入software，将下载好的四个文件都托入右边，再关闭xftp：



4、安装jdk、scala和spark

（1）新建文件夹和解压软件

回到xshell中，输入以下命令：

```
sudo mkdir /usr/local/java
cd software
sudo tar -zxvf jdk-8u211-linux-x64.tar.gz -C /usr/local/java
sudo mkdir /usr/local/scala
sudo tar -zxvf scala-2.11.12.tgz -C /usr/local/scala
sudo mkdir /usr/local/spark
sudo tar -zxvf spark-2.3.3-bin-hadoop2.7.tgz -C /usr/local/spark
```

ps.第一个命令输入后需要输入密码。

（2）配置环境变量

个人习惯使用vim，在命令行中输入sudo apt-get install vim安装vim，中间输入y。

再输入命令打开环境变量文件：

```
sudo vim ~/.bashrc
```

shift+g快速到文件末尾，按o输入以下命令：

```
export JAVA_HOME=/usr/local/java/jdk1.8.0_211
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
export PATH=${JAVA_HOME}/bin:$PATH
export SCALA_HOME=/usr/local/scala/scala-2.11.12
export PATH=${SCALA_HOME}/bin:$PATH
export SPARK_HOME=/usr/local/spark/spark-2.3.3-bin-hadoop2.7
export PATH=${SPARK_HOME}/bin:$PATH
```

按Esc，再输入:wq保存退出，出来后在命令行输入命令使环境生效：

```
source ~/.bashrc
```

（3）测试

在终端中分别输入java -version、scala（退出按Ctrl+d）、spark-shell（退出按Ctrl+d）和pyspark来对jdk、scala和spark进行测试：

```
spark@ubuntu:~/software$ java -version
java version "1.8.0_211"
Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.211-b12, mixed mode)
```

```
spark@ubuntu:~/software$ scala
Welcome to Scala 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_211).
Type in expressions for evaluation. Or try :help.

scala>
```

```
spark@ubuntu:~/software$ spark-shell
2019-05-16 01:06:30 WARN  Utils:66 - Your hostname, ubuntu resolves to a loopback address: 127.0.1.1; using 192.168.206.131 instead (
on interface ens33)
2019-05-16 01:06:30 WARN  Utils:66 - Set SPARK_LOCAL_IP if you need to bind to another address
2019-05-16 01:06:30 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes
where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://192.168.206.131:4040
Spark context available as 'sc' (master = local[*], app id = local-1557993999294).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.3.3
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_211)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

输入pyspark会报错：

```
spark@ubuntu:~/software$ pyspark
/usr/local/spark/spark-2.3.3-bin-hadoop2.7/bin/pyspark: line 45: python: command not found
env: ‘ python’ : No such file or directory
spark@ubuntu:~/software$
```

解决方法：打开环境变量（sudo vim ~/.bashrc），在文末添加以下语句：

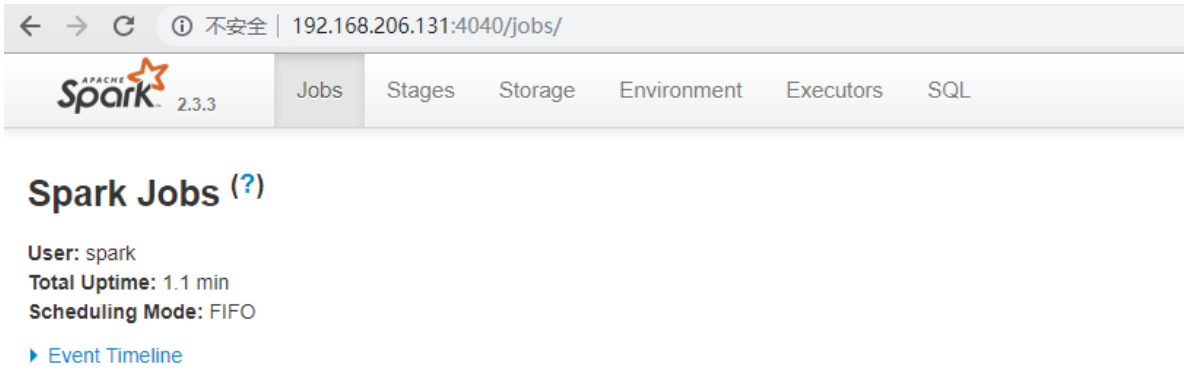
```
export PYTHONPATH=${SPARK_HOME}/python:${SPARK_HOME}/python/lib/py4j-0.10.7-
src.zip:
export PYSPARK_PYTHON=python3
```

保存并退出，再输入source ~/.bashrc使配置生效。

这次在终端中输入pyspark，成功进入：

```
spark@ubuntu:~/software$ pyspark
/usr/local/spark/spark-2.3.3-bin-hadoop2.7/bin/pyspark: line 45: python: command not found
Python 3.5.2 (default, Nov 12 2018, 13:43:14)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
2019-05-16 01:12:31 WARN  Utils:66 - Your hostname, ubuntu resolves to a loopback address: 127.0.1.1; using 192.168.206.131 instead (
on interface ens33)
2019-05-16 01:12:31 WARN  Utils:66 - Set SPARK_LOCAL_IP if you need to bind to another address
2019-05-16 01:12:31 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes
where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.3.3
      /_/

Using Python version 3.5.2 (default, Nov 12 2018 13:43:14)
SparkSession available as 'spark'.
>>>
```

在主机的浏览器中输入your's ip:4040会出来页面：



终端输入exit()退出pyspark。

5、修改本机名称

（1）查看本机ip

在终端中输入ifconfig查看ip：

```
spark@ubuntu:~/software$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:a6:37:12
          inet addr:192.168.206.131  Bcast:192.168.206.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fea6:3712/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:503133 errors:0 dropped:0 overruns:0 frame:0
          TX packets:27947 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:708216998 (708.2 MB)  TX bytes:4604940 (4.6 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:753 errors:0 dropped:0 overruns:0 frame:0
          TX packets:753 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:52729 (52.7 KB)  TX bytes:52729 (52.7 KB)
```

（2）修改本机名称

在终端中输入命令：

```
sudo vim /etc/hostname
```

将里面的ubuntu修改成master

（3）修改hosts

在终端输入命令：

```
sudo vim /etc/hosts
```

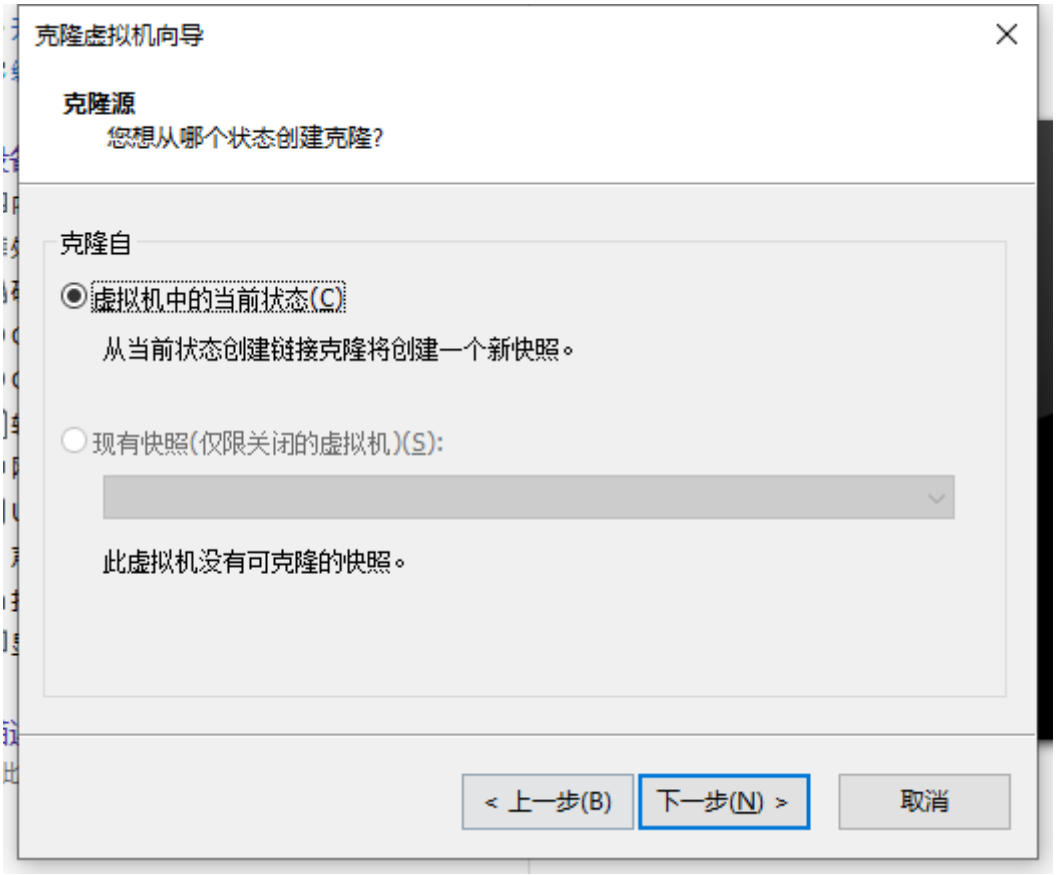
修改为下面这样：

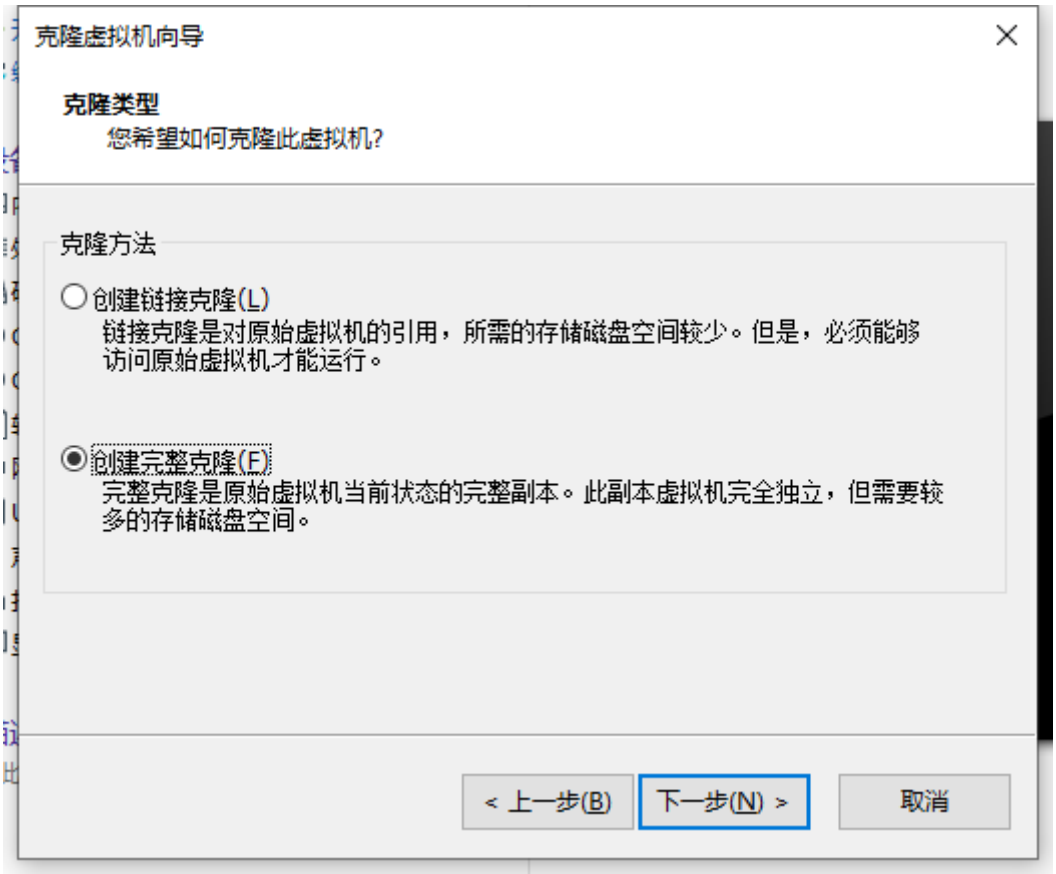
```
127.0.0.1       localhost
192.168.206.131         master
192.168.206.132         slave1
192.168.206.133         slave2

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
~
```

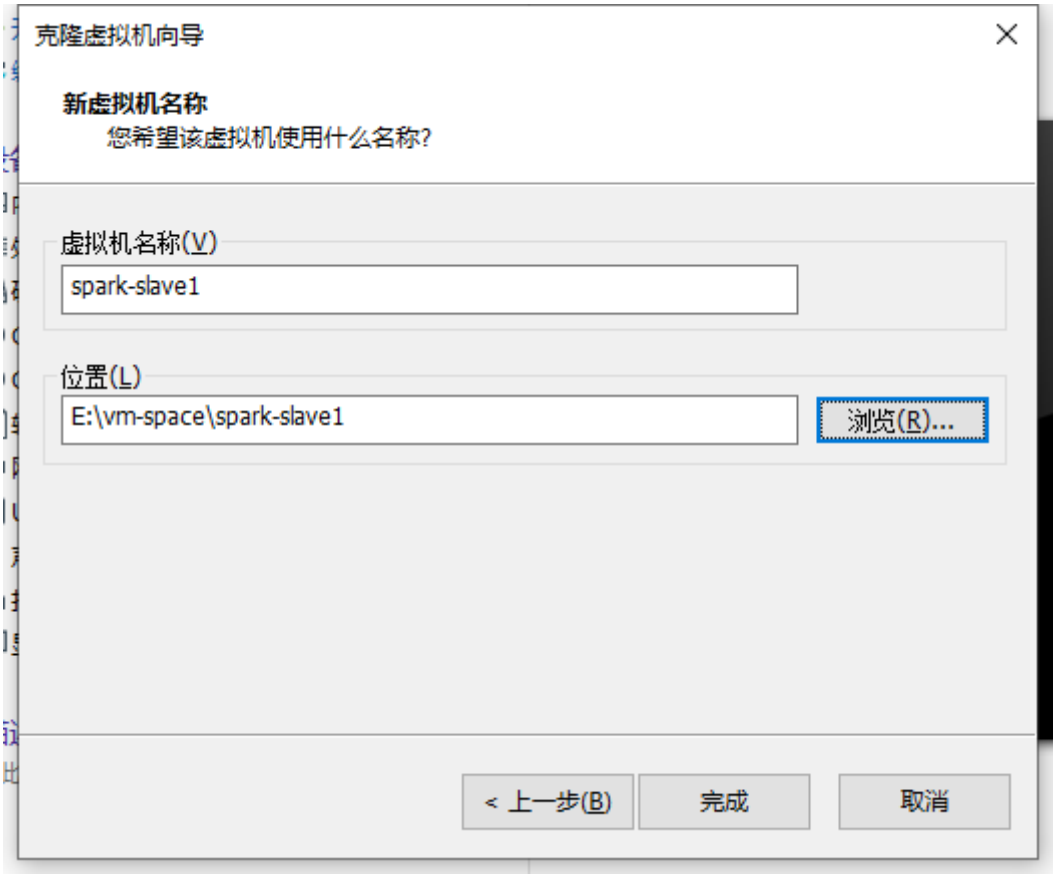
保存退出后关闭虚拟机。

6、克隆虚拟机

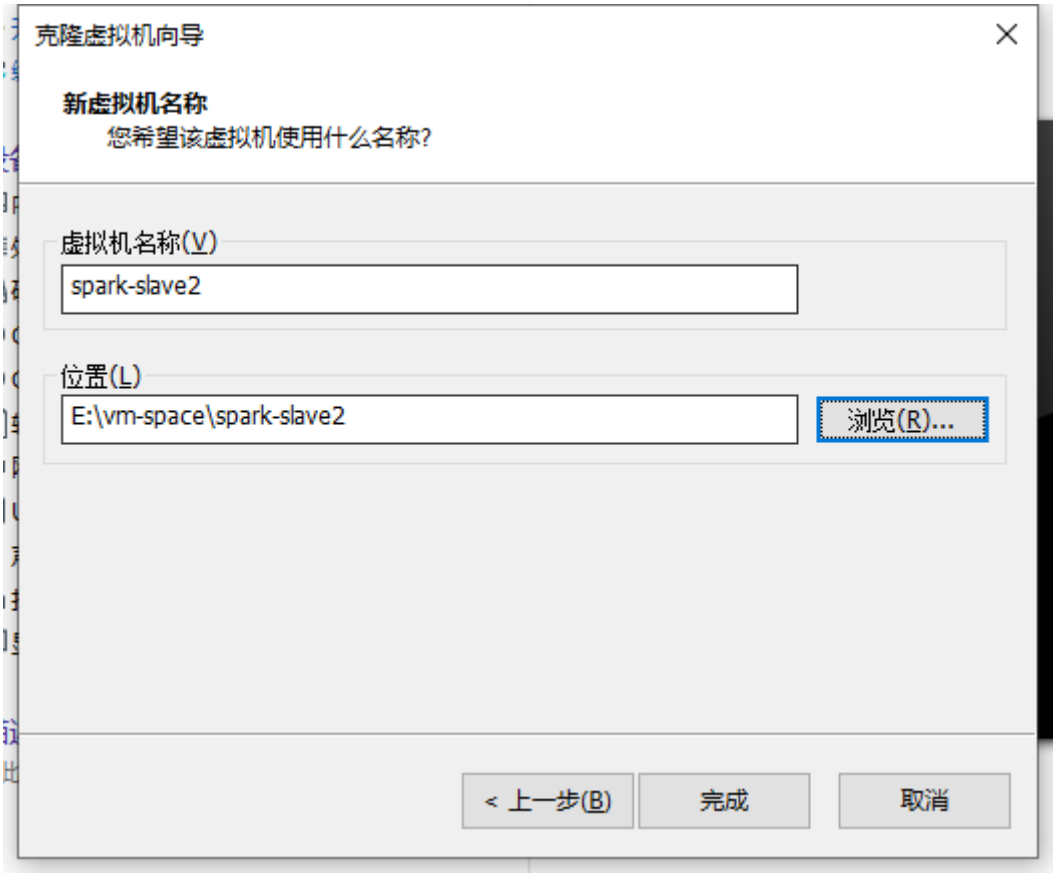
（1）右键spark-master→管理→克隆


克隆虚拟机向导
克隆源
您想从哪个状态创建克隆?
克隆自
虚拟机中的当前状态(C)
从当前状态创建链接克隆将创建一个新快照。
现有快照(仅限关闭的虚拟机)(S):
此虚拟机没有可克隆的快照。
< 上一步(B)
下一步(N) >
取消


克隆虚拟机向导
克隆类型
您希望如何克隆此虚拟机?
克隆方法
创建链接克隆(L)
链接克隆是对原始虚拟机的引用，所需的存储磁盘空间较少。但是，必须能够访问原始虚拟机才能运行。
创建完整克隆(F)
完整克隆是原始虚拟机当前状态的完整副本。此副本虚拟机完全独立，但需要较多的存储磁盘空间。
< 上一步(B)
下一步(N) >
取消



等待完成，第二个也这样克隆。



（2）开启三个虚拟机

依次启动并登录三个虚拟机，一定要按顺序（master→slave1→slave2）启动，不然ip顺序会乱。分别在三台虚拟机的终端中输入ifconfig，如果ip和hosts的一致就没问题，否则要修改成对应的。

（3）修改hostname

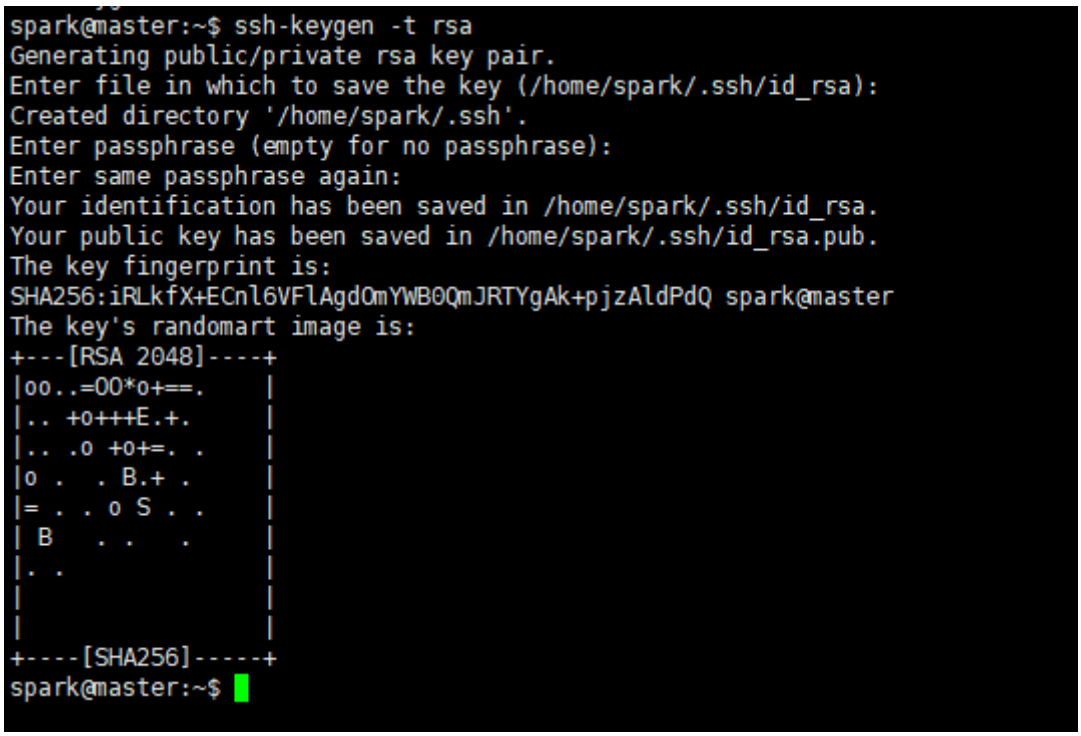
分别在spark-slave1和spark-slave2的终端输入sudo vim /etc/hostname，将里面的内容分别改为slave1和slave2，退出并保存，再输入命令sudo reboot重启两台虚拟机，名字就会变成对应的名字。

7、SHH免密登录

首先用xshell连接剩下两台虚拟机（个人觉得xshell更方便一点）。

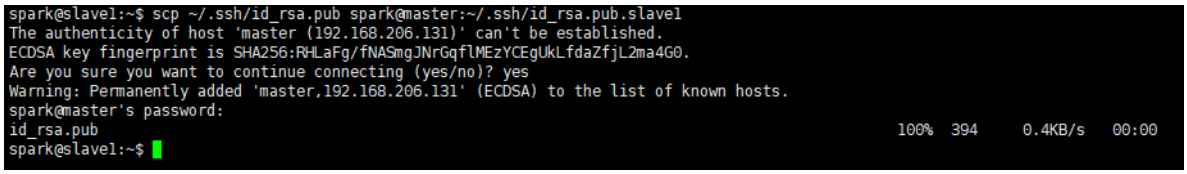
分别在三台终端中输入以下命令，一直按回车就行：

```
ssh-keygen -t rsa
```



然后分别将slave1与slave2上的id_rsa.pub用scp命令发送给master:

```
scp ~/.ssh/id_rsa.pub spark@master:~/.ssh/id_rsa.pub.slave1
scp ~/.ssh/id_rsa.pub spark@master:~/.ssh/id_rsa.pub.slave2
```



在master上，将所有公钥加到用于认证的公钥文件authorized_keys中：

```
cat ~/.ssh/id_rsa.pub* >> ~/.ssh/authorized_keys
```

在master上，将公钥文件authorized_keys分发给每台slave：

```
scp ~/.ssh/authorized_keys spark@slave1:~/.ssh/
scp ~/.ssh/authorized_keys spark@slave2:~/.ssh/
```

```
spark@master:~$ scp ~/.ssh/authorized_keys spark@slave1:~/.ssh/
The authenticity of host 'slave1 (192.168.206.132)' can't be established.
ECDSA key fingerprint is SHA256:RHLaFg/fNASmgJNrGqflMEzYCEgUkLfdaZfjL2ma4G0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave1,192.168.206.132' (ECDSA) to the list of known hosts.
spark@slave1's password:
authorized_keys                                                    100% 1182     1.2KB/s   00:00
spark@master:~$ scp ~/.ssh/authorized_keys spark@slave2:~/.ssh/
The authenticity of host 'slave2 (192.168.206.133)' can't be established.
ECDSA key fingerprint is SHA256:RHLaFg/fNASmgJNrGqflMEzYCEgUkLfdaZfjL2ma4G0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave2,192.168.206.133' (ECDSA) to the list of known hosts.
spark@slave2's password:
authorized_keys                                                    100% 1182     1.2KB/s   00:00
spark@master:~$
```

现在可以在三台机器上使用以下命令测试是否可以免密登陆了：

```
ssh master
ssh slave1
ssh slave2
```

8、安装hadoop

（1）在master中，输入以下命令创建文件夹和解压文件：

```
sudo mkdir /usr/local/hadoop
cd software/
sudo tar -zxvf hadoop-2.7.7.tar.gz -C /usr/local/hadoop
```

（2）配置文件

首先在三台机器的/home/spark路径下创建几个文件夹：

```
spark@master:~$ mkdir hadoop
spark@master:~$ mkdir hadoop/tmp
spark@master:~$ mkdir hadoop/namenode
spark@master:~$ mkdir hadoop/datanode
```

进入“/usr/local/hadoop/hadoop-2.7.7/etc/hadoop”目录：

```
cd /usr/local/hadoop/hadoop-2.7.7/etc/hadoop/
```

对以下文件进行配置：hadoop-env.sh、yarn-env.sh、core-site.xml

```
//hadoop-env.sh
sudo vim hadoop-env.sh
export JAVA_HOME=/usr/local/java/jdk1.8.0_211
export PATH=$PATH:/usr/local/hadoop/hadoop-2.7.7/bin
```

保存并退出，输入命令source hadoop-env.sh使配置生效，并输入hadoop version测试：

```
spark@master:/usr/local/hadoop/hadoop-2.7.7/etc/hadoop$ source hadoop-env.sh
spark@master:/usr/local/hadoop/hadoop-2.7.7/etc/hadoop$ hadoop version
Hadoop 2.7.7
Subversion Unknown -r c1aad84bd27cd79c3d1a7dd58202a8c3ee1ed3ac
Compiled by stevel on 2018-07-18T22:47Z
Compiled with protoc 2.5.0
From source with checksum 792e15d20b12c74bd6f19a1fb886490
This command was run using /usr/local/hadoop/hadoop-2.7.7/share/hadoop/common/hadoop-common-2.7.7.jar
spark@master:/usr/local/hadoop/hadoop-2.7.7/etc/hadoop$
```

```
//yarn-env.sh
sudo vim yarn-env.sh
export JAVA_HOME=/usr/local/java/jdk1.8.0_211
```

同样保存并退出，输入命令source yarn-env.sh使之生效。

```
//slaves（把文件里的localhost删掉）
sudo vim slaves
slave1
slave2
```

```
//core-site.xml（file后面的路径与创建的文件夹路径一致）
sudo vim core-site.xml
<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://master:9000/</value>
        </property>
        <property>
                 <name>hadoop.tmp.dir</name>
                <value>file:/home/spark/hadoop/tmp</value>
        </property>
</configuration>
```

```
//hdfs-site.xml（file后面的路径与创建的文件夹路径一致）
sudo vim hdfs-site.xml
<configuration>
        <property>
                <name>dfs.namenode.secondary.http-address</name>
                <value>master:9001</value>
        </property>
         <property>
                <name>dfs.namenode.name.dir</name>
                <value>file:/home/spark/hadoop/namenode</value>
        </property>
        <property>
                <name>dfs.datanode.data.dir</name>
                <value>file:/home/spark/hadoop/datanode</value>
        </property>
        <property>
                <name>dfs.replication</name>
                <value>3</value>
        </property>
        <property>
                <name>dfs.webhdfs.enabled</name>
                <value>true</value>
        </property>
</configuration>
```

```
//mapred-site.xml
sudo cp mapred-site.xml.template mapred-site.xml
sudo vim mapred-site.xml
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
</configuration>
```

```
//yarn-site.xml
sudo vim yarn-site.xml
<configuration>
<!-- Site specific YARN configuration properties -->
        <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>
        <property>
                <name>yarn.resourcemanager.address</name>
                <value>master:8032</value>
        </property>
        <property>
                <name>yarn.resourcemanager.scheduler.address</name>
                <value>master:8030</value>
        </property>
        <property>
                <name>yarn.resourcemanager.resource-tracker.address</name>
                <value>master:8035</value>
        </property>
        <property>
                <name>yarn.resourcemanager.admin.address</name>
                <value>master:8033</value>
        </property>
        <property>
                <name>yarn.resourcemanager.webapp.address</name>
                <value>master:8088</value>
        </property>
</configuration>
```

（3）发送配置

在两台slave中创建和master一致的hadoop文件夹，并在三台机器中均使用以下命令进行文件夹的权限设置：

```
sudo chmod -R 777 /usr/local/hadoop
```

此时可以将配置好的hadoop-2.7.7文件夹分发给所有slave了：

```
scp -r /usr/local/hadoop/hadoop-2.7.7 spark@slave1:/usr/local/hadoop/
scp -r /usr/local/hadoop/hadoop-2.7.7 spark@slave2:/usr/local/hadoop/
```

分发完成后，再次在两台slave中对hadoop文件夹进行权限设置：

```
sudo chmod -R 777 /usr/local/hadoop
```

在master中使用cd命令切换到hadoop主目录，格式化namenode后，启动：

```
cd /usr/local/hadoop/hadoop-2.7.7/
./bin/hadoop namenode -format
./sbin/start-all.sh
```

输入./sbin/start-all.sh会报错，解决方法：

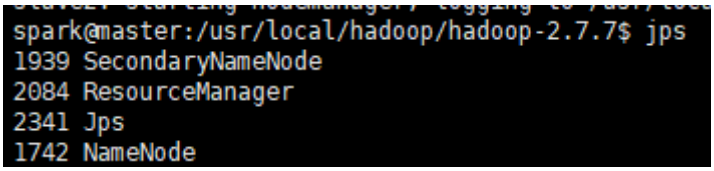
将hadoop-env.sh中的export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}

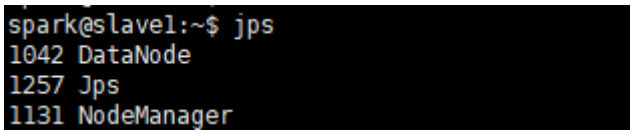
改为绝对路径：

```
export HADOOP_CONF_DIR=/usr/local/hadoop/hadoop-2.7.7/etc/hadoop
```

使用source hadoop-env.sh使之生效。

在三台机器的终端中分别输入jps命令（下图是master和slave1，slave2和slave1是一样的），看到如下图即代表启动成功，可以使用“./sbin/stop-all.sh”进行关闭：





9、二次配置spark

先使用如下命令将两台slave中的spark文件删除（一会再从master传过来）：

```
sudo rm -rf /usr/local/spark/spark-2.3.3-bin-hadoop2.7/
```

然后，在master上cd到spark的根目录下的conf目录中，修改以下文件（没有该文件则新建该文件）：

```
cd /usr/local/spark/spark-2.3.3-bin-hadoop2.7/conf/
sudo cp spark-env.sh.template spark-env.sh
sudo vim spark-env.sh
//spark-env.sh
export SCALA_HOME=/usr/local/scala/scala-2.11.12
export JAVA_HOME=/usr/local/java/jdk1.8.0_211
export HADOOP_HOME=/usr/local/hadoop/hadoop-2.7.7
export HADOOP_CONF_DIR=/usr/local/hadoop/hadoop-2.7.7/etc/hadoop
export SPARK_MASTER_IP=master
export SPARK_LOCAL_DIRS=/usr/local/spark/spark-2.3.3-bin-hadoop2.7
export SPARK_DRIVER_MEMORY=512M
```

```
sudo cp slaves.template slaves
sudo vim slaves
//slaves（把文件中自带的localhost删除）
master
slave1
slave2
```

同样的，在三台机器上使用chmod命令给文件夹赋予权限：

```
sudo chmod -R 777 /usr/local/spark
```

此时可以将配置好的spark-2.3.3-bin-hadoop2.7文件夹分发给所有slave了：

```
scp -r /usr/local/spark/spark-2.3.3-bin-hadoop2.7 spark@slave1:/usr/local/spark/
scp -r /usr/local/spark/spark-2.3.3-bin-hadoop2.7 spark@slave2:/usr/local/spark/
```

在两台slave中再次对文件夹赋予权限：

```
sudo chmod -R 777 /usr/local/spark
```

此时，配置已完成，在master上使用cd命令切换到spark主目录，进行启动：

```
cd /usr/local/spark/spark-2.3.3-bin-hadoop2.7/
./sbin/start-all.sh
```

在三台机器的终端中分别输入jps命令（图示是master和slave1），看到如下图即代表启动成功，可以使用“./sbin/stop-all.sh”进行关闭：

```
spark@master:/usr/local/spark/spark-2.3.3-bin-hadoop2.7$ jps
1939 SecondaryNameNode
2084 ResourceManager
3334 Master
3432 Worker
3482 Jps
1742 NameNode
```

```
spark@slave1:~$ jps
2018 Jps
1042 DataNode
1973 Worker
1131 NodeManager
```

10、集群测试

在spark启动成功后，在master的终端中cd到spark根目录，执行以下代码（spark自带的一个例子），出现结果即为成功：

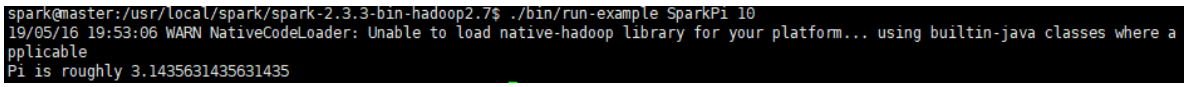
```
./bin/run-example SparkPi 10
```

```
Pi is roughly 3.1423191423191423
2019-05-16 19:50:00 INFO  AbstractConnector:318 - Stopped Spark@29b90898{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
2019-05-16 19:50:00 INFO  SparkUI:54 - Stopped Spark web UI at http://master:4040
2019-05-16 19:50:00 INFO  MapOutputTrackerMasterEndpoint:54 - MapOutputTrackerMasterEndpoint stopped!
2019-05-16 19:50:00 INFO  MemoryStore:54 - MemoryStore cleared
2019-05-16 19:50:00 INFO  BlockManager:54 - BlockManager stopped
2019-05-16 19:50:00 INFO  BlockManagerMaster:54 - BlockManagerMaster stopped
2019-05-16 19:50:00 INFO  OutputCommitCoordinator$OutputCommitCoordinatorEndpoint:54 - OutputCommitCoordinator stopped!
2019-05-16 19:50:00 INFO  SparkContext:54 - Successfully stopped SparkContext
2019-05-16 19:50:00 INFO  ShutdownHookManager:54 - Shutdown hook called
2019-05-16 19:50:00 INFO  ShutdownHookManager:54 - Deleting directory /tmp/spark-773f9d51-afe3-4b7b-b6f2-2f64d7a6dad3
2019-05-16 19:50:00 INFO  ShutdownHookManager:54 - Deleting directory /tmp/spark-bca12196-214c-4228-950a-910b08b14a16
```

PS.如果嫌界面输出的信息太多可以到conf/log4j.properties中修改配置：

```
cd conf/
sudo cp log4j.properties.template log4j.properties
sudo vim log4j.properties
```

找到log4j.rootCategory=INFO, console，将INFO改为WARN后保存并退出，重新输入代码出来结果如下：



结语

安装spark集群一定要耐心和细心，另外一定要注意文件的权限，不然一大堆的permission denied。

参考

1、https://blog.csdn.net/ms961516792/article/details/79115383#commentBox

2、https://blog.csdn.net/qq_42881421/article/details/88069211

3、https://blog.csdn.net/w3045872817/article/details/70940952

4、http://www.cnblogs.com/shishanyuan/p/4701646.html

5、https://blog.csdn.net/a602232180/article/details/74779641