```
./bin/spark-shell  --master spark://spark-master:7077  \
--conf spark.hadoop.fs.s3a.endpoint=http://172.26.0.6:9000 \
--conf spark.hadoop.fs.s3a.access.key=minio \
--conf spark.hadoop.fs.s3a.secret.key=minio123 \
--conf spark.hadoop.fs.s3a.path.style.access=true \
--conf
spark.hadoop.fs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSystem


./bin/spark-shell --master spark://spark-master:7077  --jars examples/
delta-core_2.11-0.4.0.jar \
--conf
spark.delta.logStore.class=org.apache.spark.sql.delta.storage.S3SingleDri
verLogStore \
--conf spark.hadoop.fs.s3a.endpoint=http://172.26.0.6:9000 \
--conf spark.hadoop.fs.s3a.access.key=minio \
--conf spark.hadoop.fs.s3a.secret.key=minio123 \
--conf spark.hadoop.fs.s3a.path.style.access=true \
--conf
spark.hadoop.fs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSystem

./bin/spark-shell --master spark://spark-master:7077 \
--conf
spark.delta.logStore.class=org.apache.spark.sql.delta.storage.S3SingleDri
verLogStore \
--conf spark.hadoop.fs.s3a.endpoint=http://172.26.0.6:9000 \
--conf spark.hadoop.fs.s3a.access.key=minio \
--conf spark.hadoop.fs.s3a.secret.key=minio123 \
--conf spark.hadoop.fs.s3a.path.style.access=true \
--conf
spark.hadoop.fs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSystem


./bin/spark-submit --master spark://spark-master:7077 \
--conf
spark.delta.logStore.class=org.apache.spark.sql.delta.storage.S3SingleDri
verLogStore \
--conf spark.hadoop.fs.s3a.endpoint=http://172.22.0.4:9000 \
--conf spark.hadoop.fs.s3a.access.key=minio \
--conf spark.hadoop.fs.s3a.secret.key=minio123 \
--conf spark.hadoop.fs.s3a.path.style.access=true \
--conf
```

```
spark.hadoop.fs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSystem \
--jars /spark/examples/delta-core_2.11-0.5.0.jar \
--class com.delta.Run examples/original-deltaLake2-1.0-SNAPSHOT.jar
s3a://spark-test/  delta21 schemaCheck21
```

```
./bin/spark-shell --master spark://spark-master:7077 \
--conf
spark.delta.logStore.class=org.apache.spark.sql.delta.storage.S3SingleDri
verLogStore \
--conf spark.hadoop.fs.s3a.endpoint=http://172.22.0.4:9000 \
--conf spark.hadoop.fs.s3a.access.key=minio \
--conf spark.hadoop.fs.s3a.secret.key=minio123 \
--conf spark.hadoop.fs.s3a.path.style.access=true \
--conf
spark.hadoop.fs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSystem \
--jars /spark/examples/delta-core_2.11-0.5.0.jar

val loans = spark.sql("""
          SELECT addr_state, CAST(rand(10)*count as bigint) AS count,
          CAST(rand(10) * 10000 * count AS double) AS amount
          FROM loan_by_state_delta
          """)
```

```
./bin/spark-submit --master spark://spark-master:7077 --jars examples/
delta-core_2.11-0.4.0.jar  --class com.delta.Run examples/original-
deltaLake2-1.0-SNAPSHOT.jar s3a://spark-test/
```

Book stack.cn/read/angel-v3.0

Spark 集群执行命令

1. 在 spark 的 conf 目录下的 aprk.env.sh(aprk.env.sh 是复制来于 aprk.env.sh.template) 里面有很多的相关的配置，目前初始阶段只涉及到
   spark_master_host = node-1
   spark_master_port=7077

2. 还是在 conf 目录下，这里定义了三个 子节点，node-2, node-3, node-4

3. Centos7 防火墙的操作
   systemctl status firewalld 相对于的 status -> stop -> start 来查看，关

闭，开启防火墙

Centos7 安装
https://www.bilibili.com/video/av76793551/
https://www.youtube.com/watch?v=ACypx1rwm6g&t=4s

4. 查看 master 的管理页面
   http://192.168.31.100:8080/

   http://192.168.31.100:7077/ master 和
目录所在地方： /opt/spark
5. 开启 spark 到 spark 文件的 sbin 文件下， 执行
   start-all.sh 开启， stop-all.sh 关闭

6. 开启 hdfs 文件系统
   在 hadoop 的 sbin 目标下执行 ./start-dfs.sh 命令

7. 查看 hdfs 文件
   hddfs 的命令都在 /opt/spark/hadoop/hadoop-2.6.5/bin 目录中
   ./hdfs dfs -ls /

   在 web 页面可以看见 http://192.168.31.100:50070

8. 启动 spark 的 shell
   在 /opt/spark/spark/spark-2.2.0-bin-hadoop2.7/bin 路径下

   ./spark-shell 如果直接执行这个启动 spark，那么这是本地模式启动 spark

   启动的时候指定 master 启动集群模式 ./spark-shell --master spark://node-1:7077

   并在 master 中会启动 SparkSubmit 进程，这个进程主要是提交任务到每个 worker 的 Executor 中去执行

   然后通过 master 的管理页面 http://192.168.31.100:8080/ 可以查看到 Spark shell 正在运行的 spark application

   启动之后， 再所有的 节点 node 中都会出现一个 CoarseGrainedExecutorBackend 进程 这就是真正的跑集群任务

9. http://192.168.31.100:8088/cluster ResourceManager UI 界面
   1. 启动 yarn ${HADOOP_HOME}./satrt-yarn.sh

9. 计算第一个任务

sc.textFile("hdfs://node-1:9000/wordcount.txt").flatMap(_.split(" ")).map((_, 1)).reduceByKey(_+_).sortBy(_._2).collect

注意关闭 所有集群上的防火墙，否则计算的时候，集群之间无法通信

10. Master 负责资源调度，也就是决定在哪些 worker 中启动 Executor 和监控 worker，这和 Yarn 是很相似的
这就相当于 yarn 中的 ResourceManager，都是去完成资源调度。

worker 负责启动对应的执行任务的进程 (Excutor)，并且监控这个进程 (Excutor),并且将当前集群的信息通过心跳汇报给
master，worker 相对于 yarn 而言就相当于是 NodeManager，Executor 进程负责执行计算任务，

spark-submit 负责向 master 提交任务并申请资源，然后该任务下的 Executor 和 sparkSubmit 进行通信，监控 Executor

而这个 spark-submit 就相当于是 yarn 中的 AppMater

11 提交任务到集群

打 jar 包并上传到集群机器 scp learning-1.0-SNAPSHOT.jar root@192.168.31.100:/opt/spark

在任何一台 node 中都可以提交任务

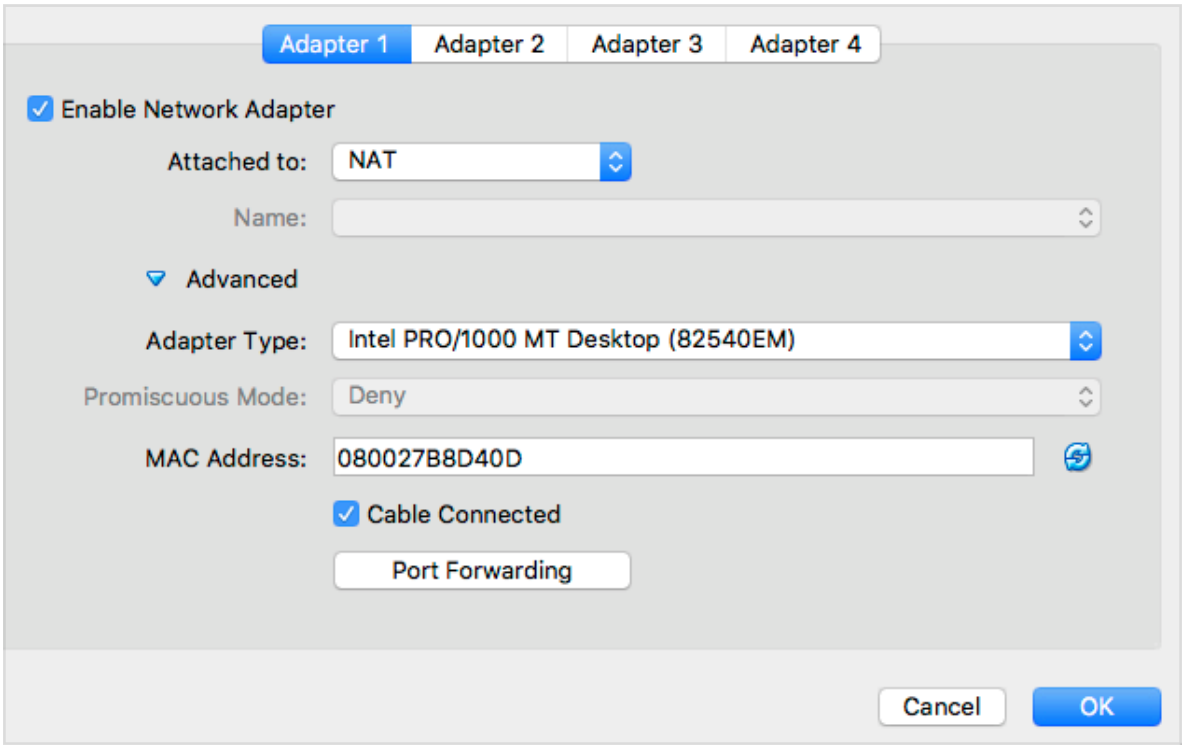
执行
./spark-submit --master spark://node-1:7077 --executor-memory 512m --total-executor-cores 2
--class com.bigData.spark.App /opt/spark/spark-code-jars/learning-1.0-SNAPSHOT.jar hdfs://node-1:9000/wordcount.txt hdfs://node-1:9000/wordcountResult121.txt

./spark-submit --class org.apache.spark.examples.SparkPi --master yarn --deploy-mode cluster --driver-memory 1G --executor-memory 1G --executor-cores 1 lib/spark-exampl

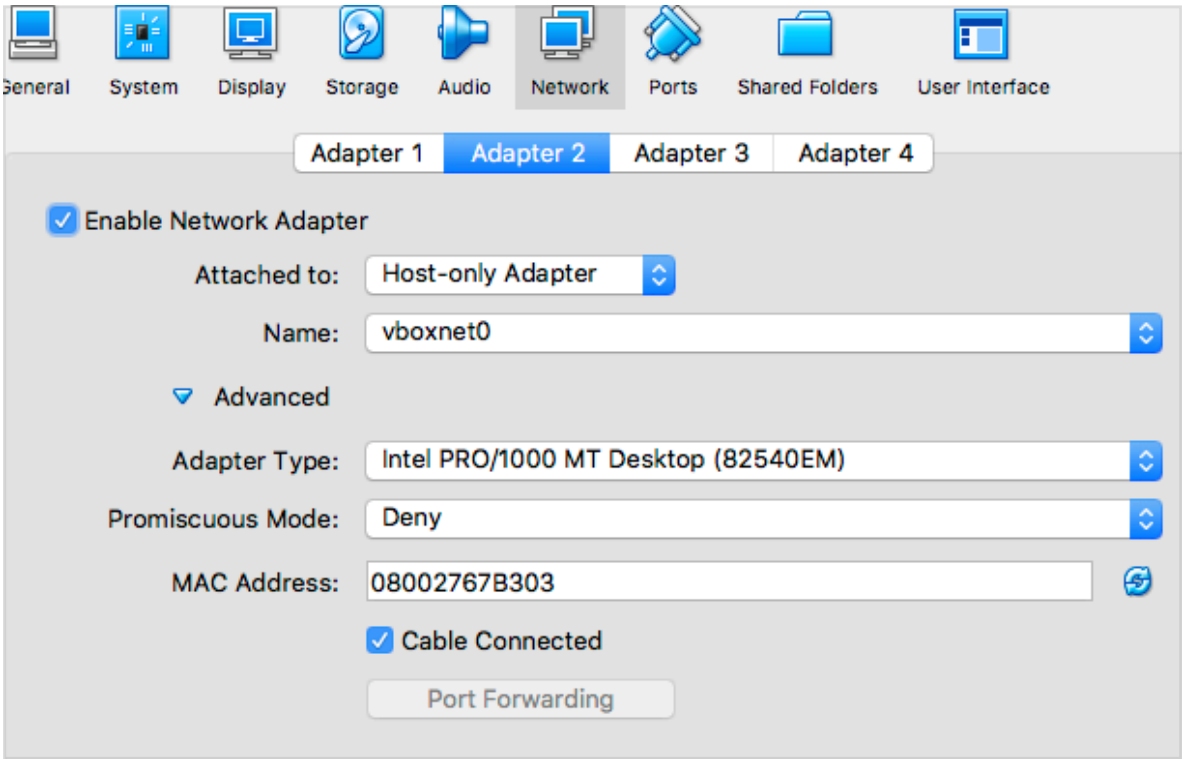
最后面几个参数分别是：jar 路径 执行的那个 class 锁需要的 2 个参数

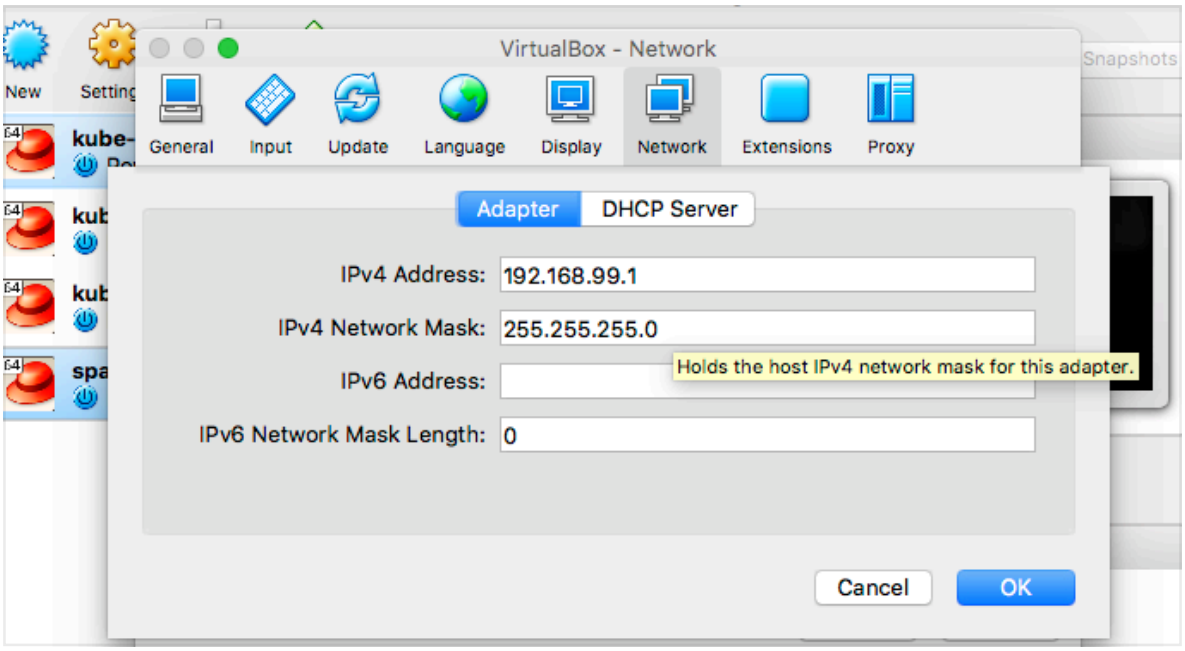
安装 Centos 虚拟机，关键的步骤是设置好网络，这里这是 2 个网络，一个 NAT 网络，用于连接网络，一个 HOST-ONLY，用于 物理机 ssh 连接到虚拟

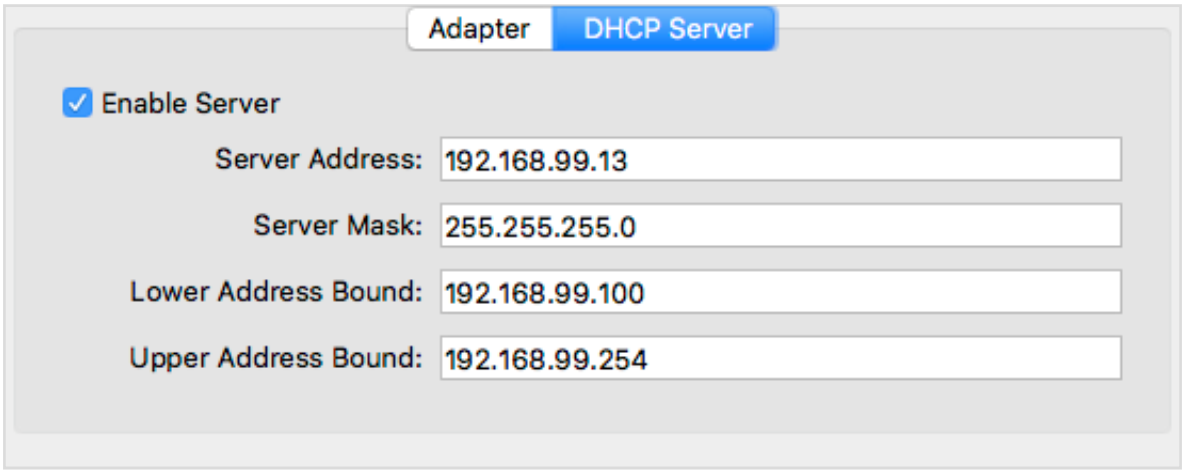
机，如果不使用物理机连接，那么就需要在虚拟机上安装增强功能，才可以用那些 复制粘贴等功能。



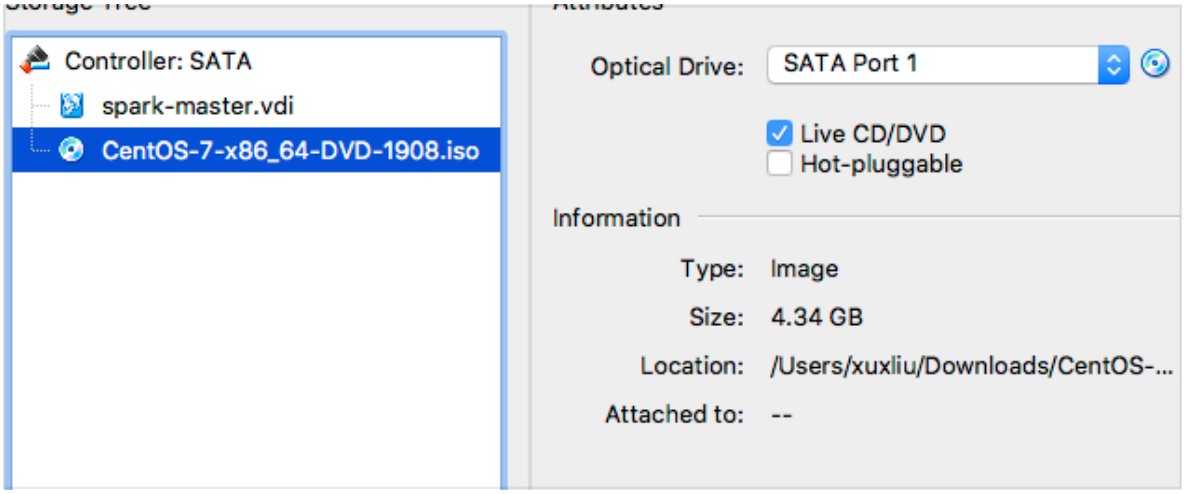
主机网络





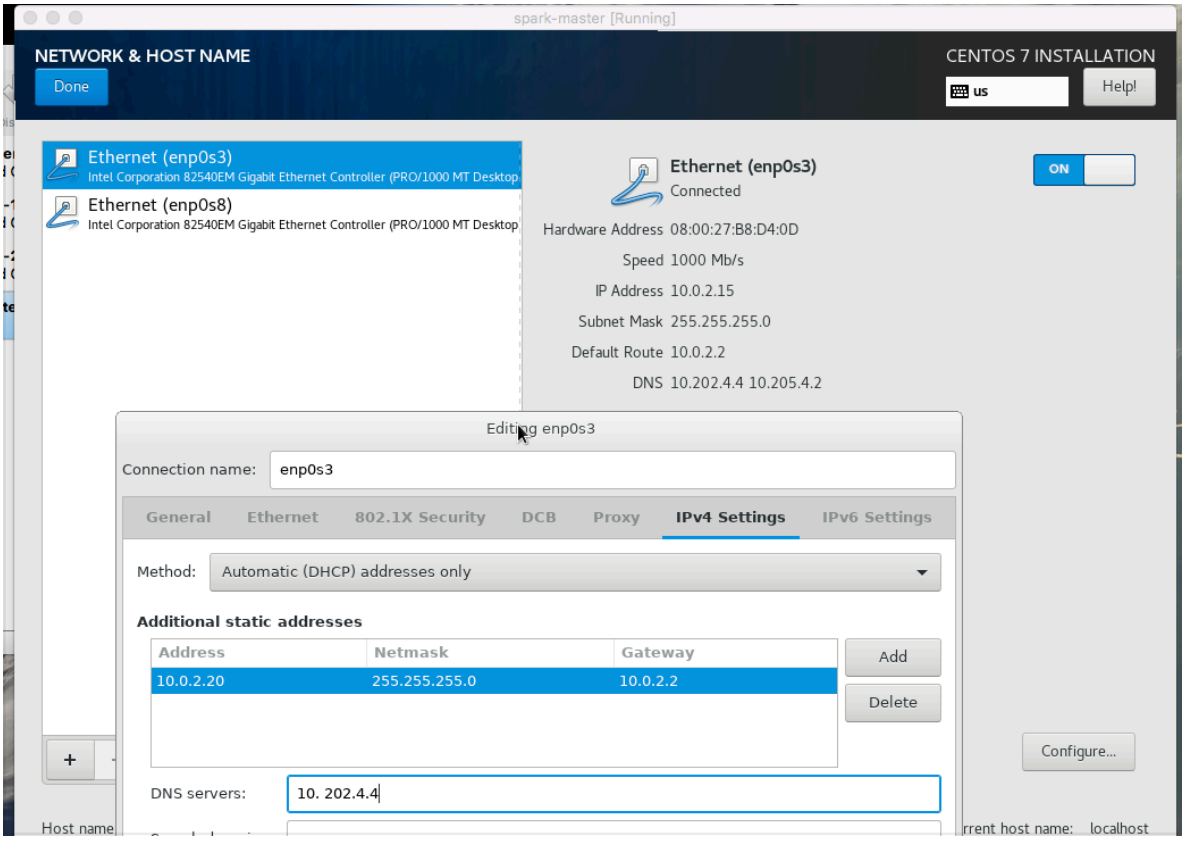


铁架iso光驱，去掉IDE

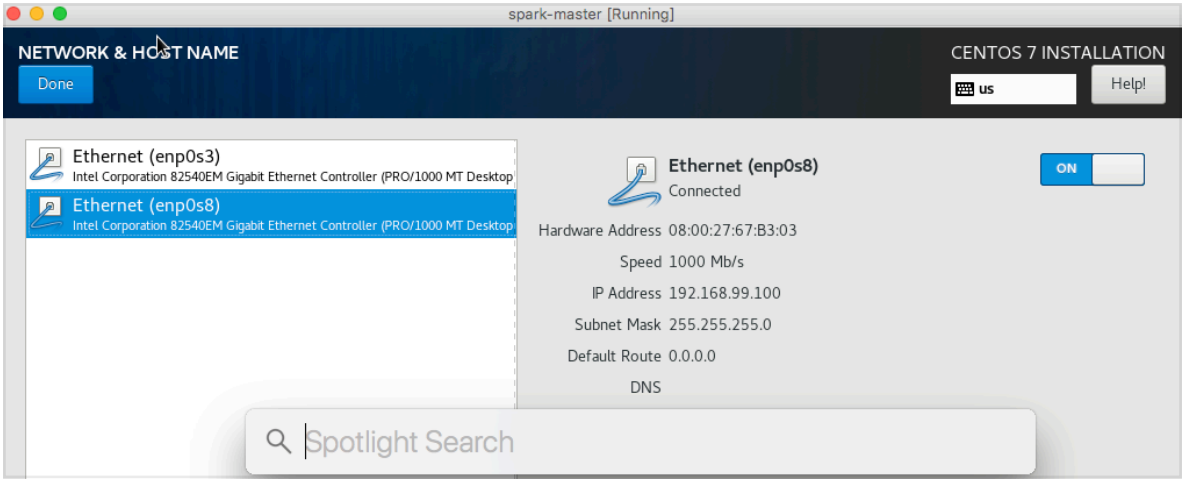


确定后，启动系统，开始安装

安装过程中配置网卡NAT，其他步骤省略，基本就是下一步下一步
NAT模式进入configure.. 配置静态网络。

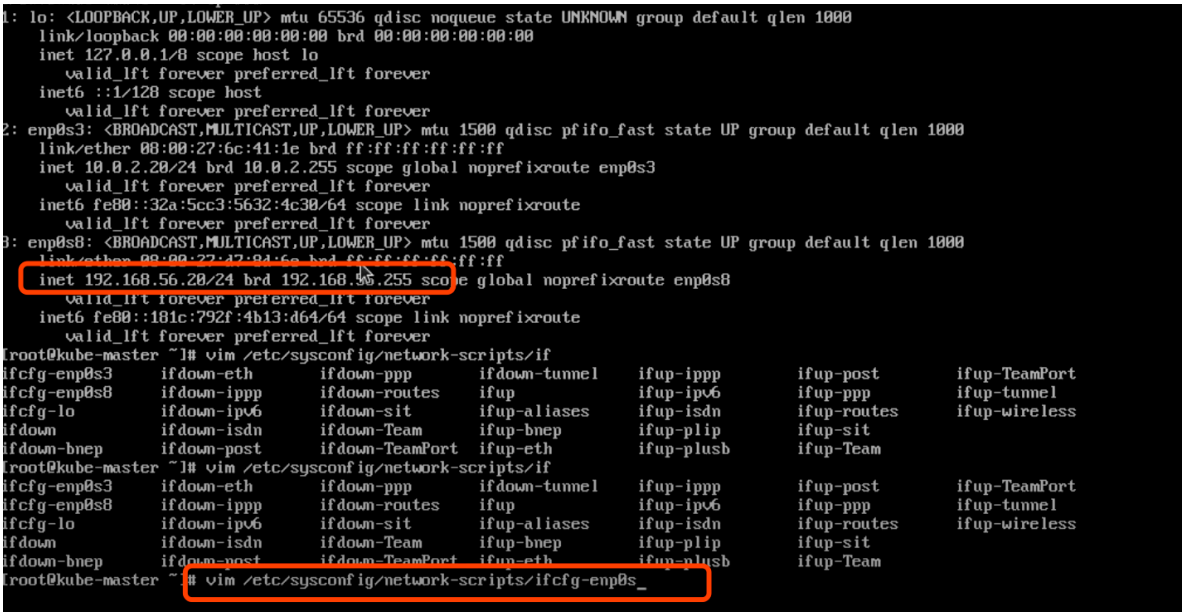


HOST-ONLY，这里没有配置静态IP，在安装完成之后，安装过几次有时候有IP，有时候没有的，如果没有的，那么安装完整之后配置网卡配置文件。

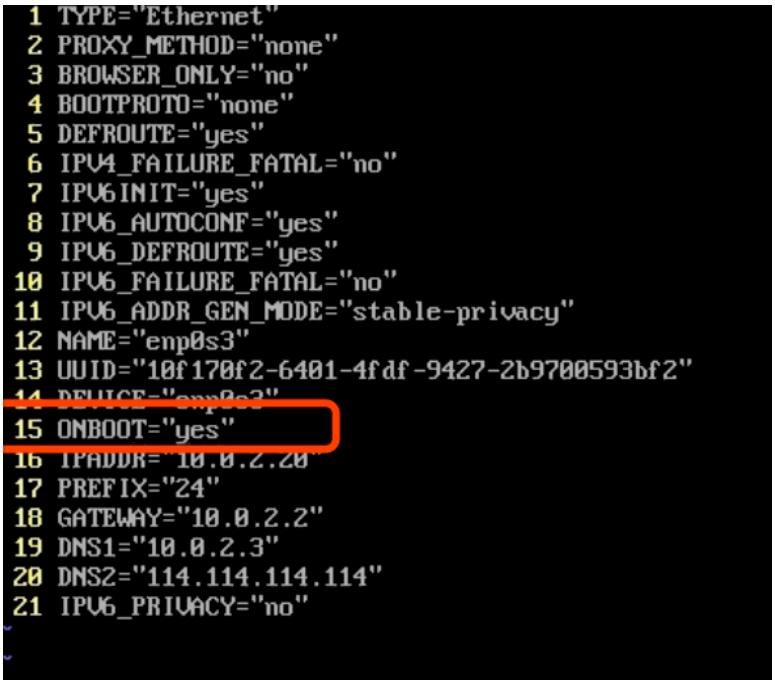


然后设置用户名密码即可

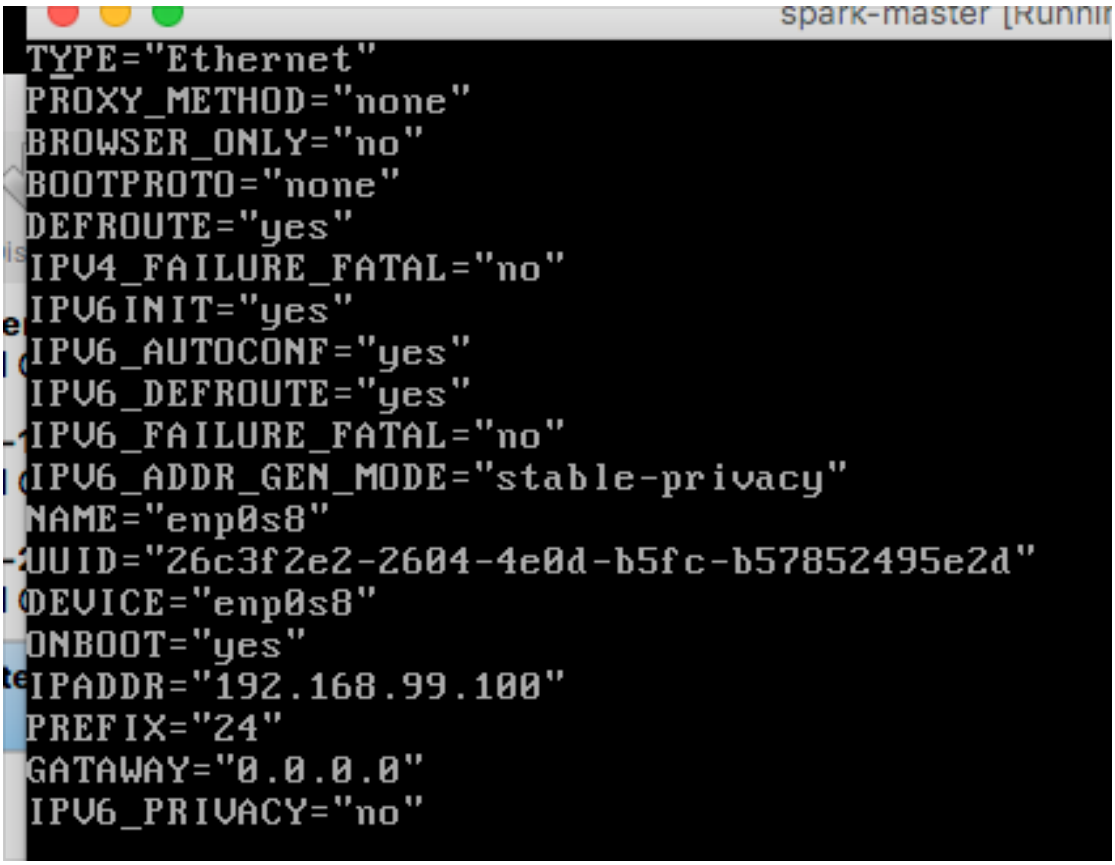
如果配置之后，发现主机模式下面没有IP地址那么进行如下配置



确认 onboot 是 yes



然后重启网络,
我的配置



网络重启后 service network restart 然后 ip addr 就能看到 网卡 8 上的 ip 了

更改国内镜像源

mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.backup
wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo
然后跟下 yum 安装包
yum -y update
大于出来使用 aliyun 的镜像源信息

下载 spark 2.4.5， 此次安装时要使用 Delta lake，它需要 spark 在 2.4.2 以上的版本
https://mirrors.tuna.tsinghua.edu.cn/apache/spark/spark-2.4.5/spark-2.4.5-bin-hadoop2.7.tgz
下载 hadoop，这里使用国内镜像下载
https://mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/hadoop-2.7.7/
在 spark 安装的过程中不需要安装 scala，原因是 spark 依赖中以及有 scala 了。

关闭防火墙

systemctl stop firewalld
systemctl disable firewalld

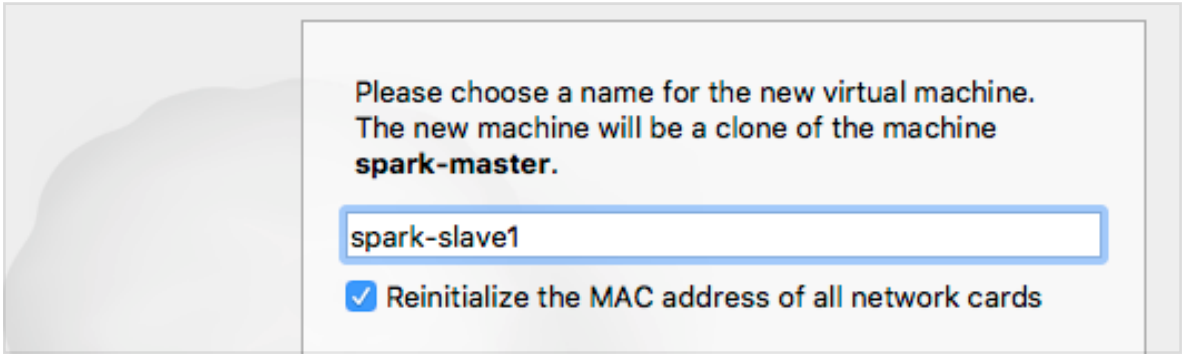
我是在本地下载好 spark，hadoop jdk 等然后 Scp 到虚拟机
scp spark-2.4.5-bin-hadoop2.7.tgz root@192.168.99.100:/

安装 Java，配置环境变量

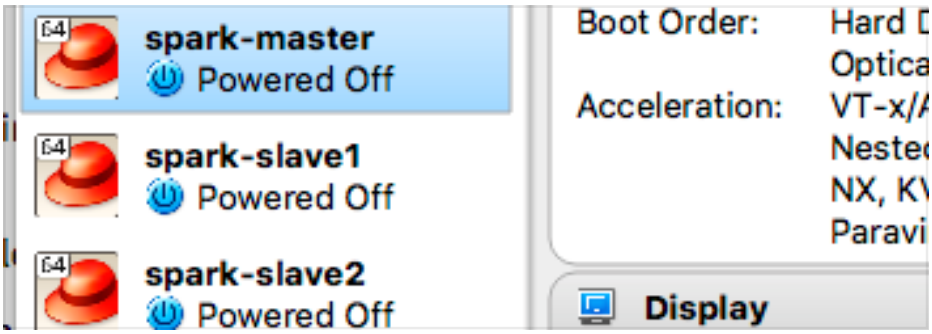
export JAVA_HOME=/opt/bigData/othersoftware/jdk1.8.0_221
export JRE_HOME=$JAVA_HOME/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
export PATH=${JAVA_HOME}/bin:$PATH
source ~/.bachrc

安装 slave 节点，

然后在 master 安装 hadoop 以及 spark，之间 scp 到 slave 节点中，就省去很多一一的配置
克隆 虚拟机，slave1，注意在克隆的时候 MAC 地址需要重新初始化



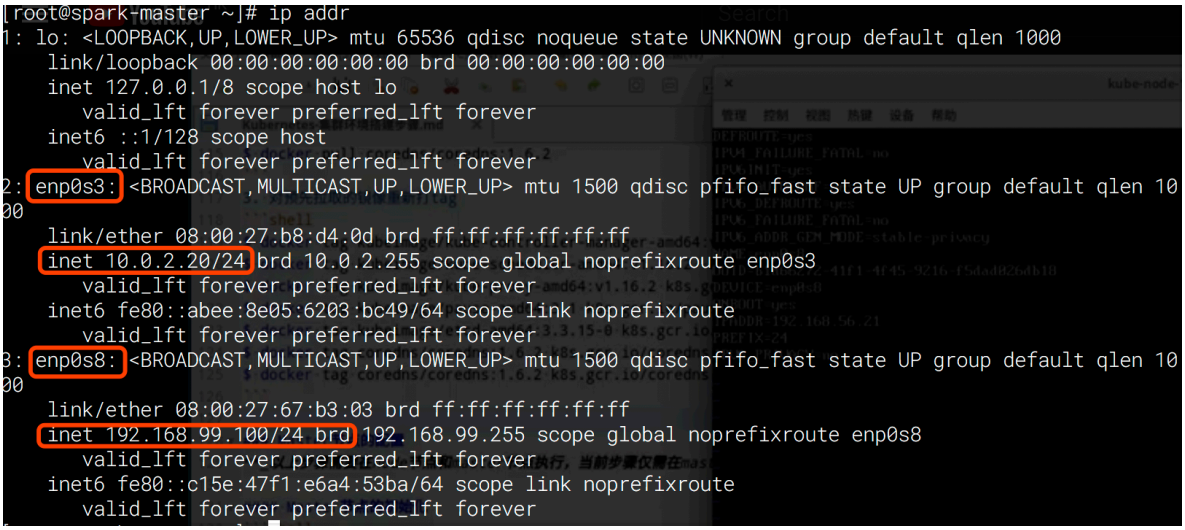
这样最终复制出来 2 个虚拟机 slave1, slave2,



克隆过来的虚拟机，系统修改 **hostname**

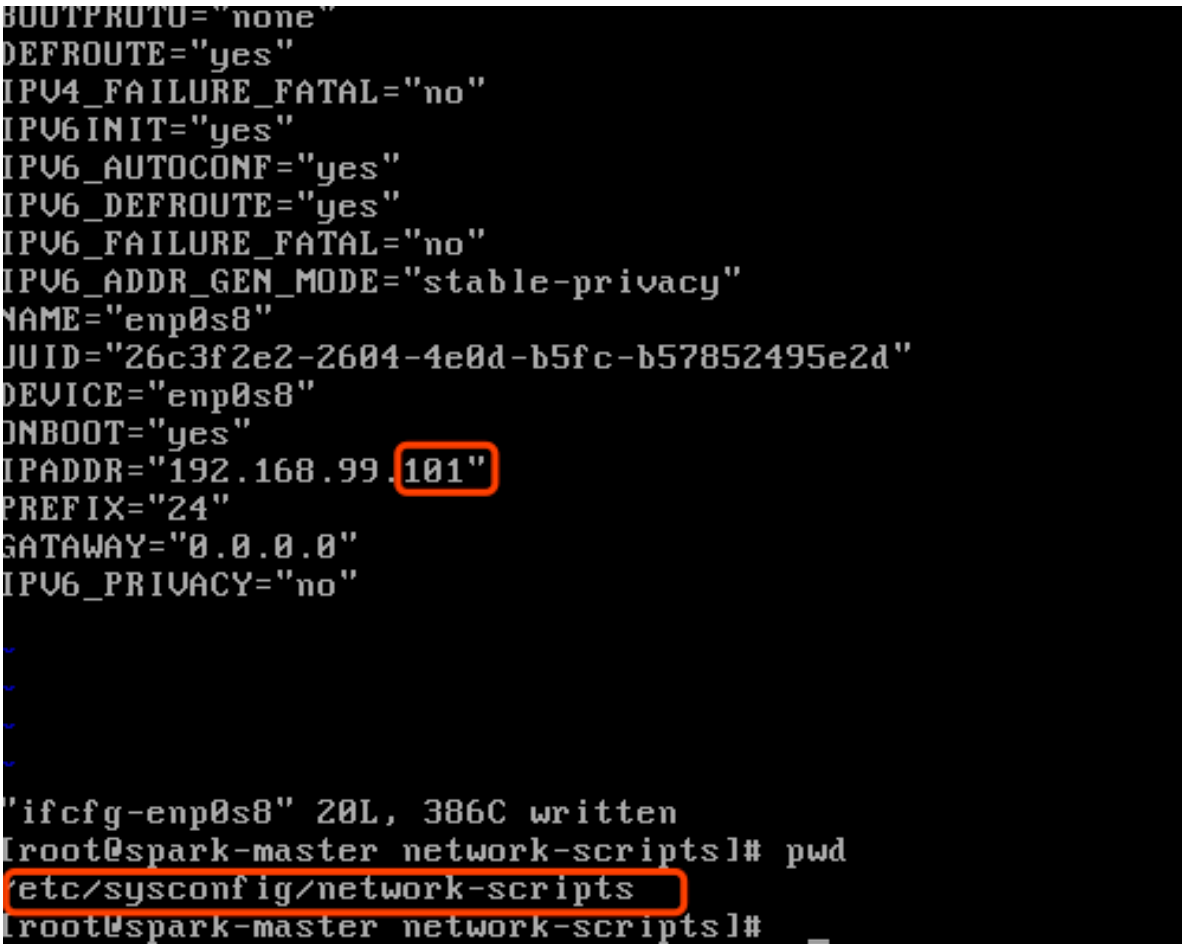
Master 使用的是静态 IP，所以 master 和 slave 之间的 IP 也是冲突的，需要修改

这是 master 的网卡

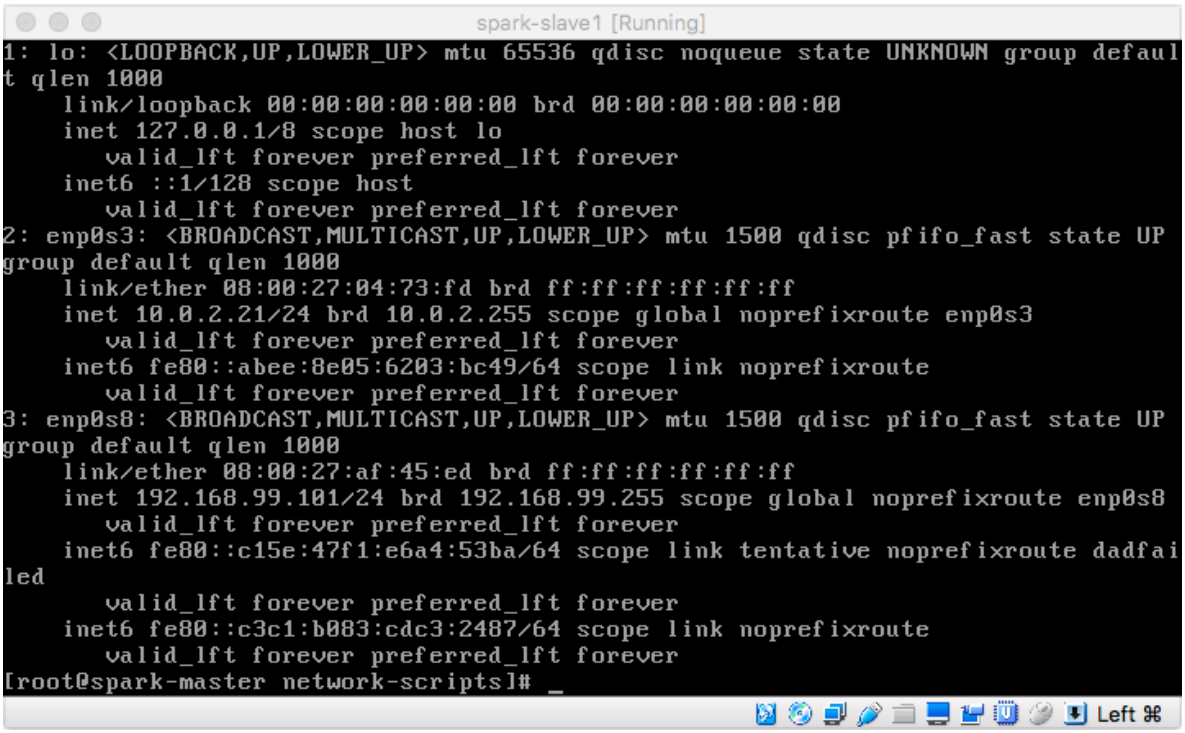


查看下 slave 的网卡和这里的 ip 都是一样的

进入目录将 2 张网卡的配置文件全部都重新配置，这里贴出来修改网卡 enp0s8 的，将 192.168.99.100 改为 92.168.99.101，同样，在网卡 enp0s3 中将，10.0.2.20 改为 10.0.2.21



service network restart 重启网络， 然后在查看 ip addr， 修改已全部生效



在2台slave 机器上执行相同的操作。

修改文件

```
[root@spark-master ~]# cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.99.100 spark-master
192.168.99.101 spark-slave1
192.168.99.102 spark-slave2
```

重新启动 1、reboot; 2、shutdown -r now 上文明配置的hosts 以及 hostname 生效

设置免密登录，

spark master 需要给 slave 节点分配任务, 需要确保 master 可以免密登录到其他机器上
ssh-keygen -t rsa 一路回车
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys 密匙是默认放在 ~/.ssh/id_rsa.pub，目录中，
然后将master 的id_rsa.pub scp 到slave目录中。并写入到
cat id_rsa.pub >> ~/.ssh/authorized_keys
然后master 就可以免密登录到 slave1 和 slave2， 中，如果需要slave 登录到master，同样将slave 的该文件写到 master 的authorized_keys文件中。

执行 exit，退出登录。

安装及配置 hadoop

tar -zxvf hadoop-2.7.7.tar.gz
export HADOOP_HOME=/opt/bigData/hadoop/hadoop-2.7.7
export PATH=$PATH:${HADOOP_HOME}/bin:${HADOOP_HOME}/sbin

进入 /opt/bigData/hadoop/hadoop-2.7.7/etc/hadoop

配置 core-site
<configuration>
<property>
    <name>fs.defaultFS</name>
    <value>hdfs://spark-master:9000</value>
</property>
<property>
    <name>hadoop.tmp.dir</name>
    <value>/opt/bigData/hadoop/tmpDir</value>
</property>
</configuration>

配置 hdfs-site
<configuration>
<property>
    <name>dfs.replication</name>
    <value>2</value>
</property>
</configuration>

配置 mr 以及 secondaryNameNode
<configuration>
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value> // 在 yarn 上跑 mapreduce 计算

</property>
<property>
        <name>dfs.namenode.secondary.http-address</name>
        <value>spark-slave1:50090</value>
</property>
</configuration>

配置 yarn
<property>
    <name>yarn.resourcemanager.hostname</name>
    <value>spark-master</value>
</property>
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
</configuration>

配置 slaves(感觉这里应该不写 spark-master 的)

```
[root@spark-master hadoop]# cat slaves
spark-master
spark-slave1
spark-slave2
```

注意一定要配置 JAVA_HOME 环境变量，因为 hadoop-env.sh 中会去读环境变量

```
# The java implementation to use.
export JAVA_HOME=${JAVA_HOME}
```

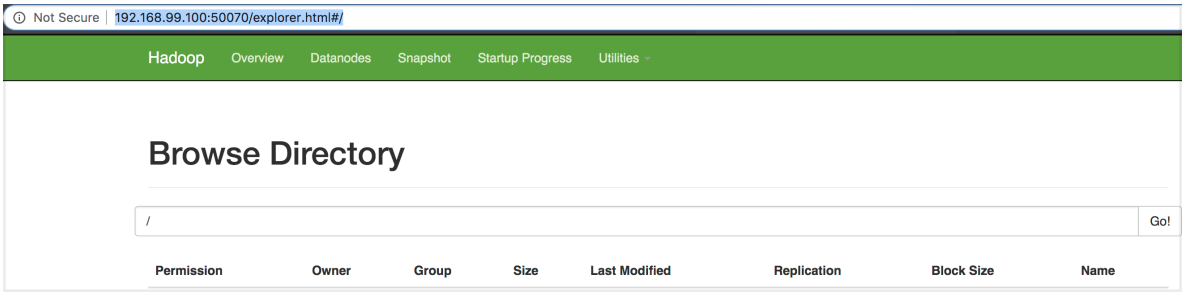
至此配置完 hadoop 将 hadoop 包 scp 到其他的 slave 机器。
scp -r hadoop-2.7.7 root@spark-slave2:/opt/bigData/hadoop

然后就可以启动 hadoop 了。

首次启动需要执行格式化 nameNode
hdfs namenode -format
启动
start-dfs.sh

```
[root@spark-master hadoop]# start-dfs.sh
Incorrect configuration: namenode address dfs.namenode.servicerpc-address or dfs.namenode.rpc-address is not configured.
Starting namenodes on []
spark-slave2: starting namenode, logging to /opt/bigData/hadoop/hadoop-2.7.7/logs/hadoop-root-namenode-spark-slave2.out
spark-master: starting namenode, logging to /opt/bigData/hadoop/hadoop-2.7.7/logs/hadoop-root-namenode-spark-master.out
spark-slave1: starting namenode, logging to /opt/bigData/hadoop/hadoop-2.7.7/logs/hadoop-root-namenode-spark-slave1.out
spark-slave1: starting datanode, logging to /opt/bigData/hadoop/hadoop-2.7.7/logs/hadoop-root-datanode-spark-slave1.out
spark-slave2: starting datanode, logging to /opt/bigData/hadoop/hadoop-2.7.7/logs/hadoop-root-datanode-spark-slave2.out
spark-master: starting datanode, logging to /opt/bigData/hadoop/hadoop-2.7.7/logs/hadoop-root-datanode-spark-master.out
Starting secondary namenodes [spark-slave1]
spark-slave1: starting secondarynamenode, logging to /opt/bigData/hadoop/hadoop-2.7.7/logs/hadoop-root-secondarynamenode-spark-slave1.out
```

可以查看 hdfs 文件系统了
http://192.168.99.100:50070/explorer.html#/



安装配置 spark

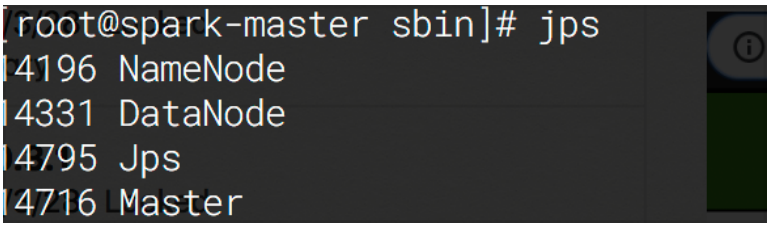
减压，添加环境变量
export SPARK_HOME=/opt/bigData/spark/spark-2.4.5-bin-hadoop2.7
export PATH=$PATH:${HADOOP_HOME}/bin:${HADOOP_HOME}/sbin:${SPARK_HOME}/bin:${SPARK_HOME}/sbin

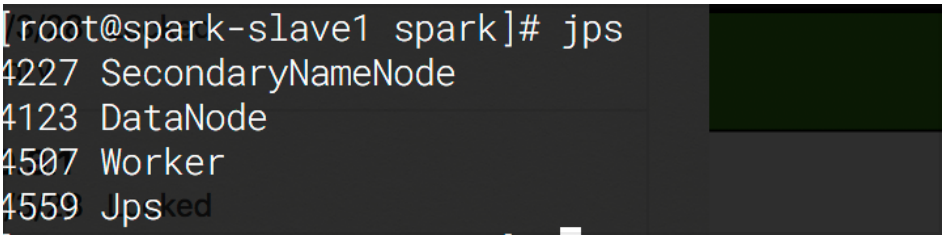
配置：
slaves
spark-slave1
spark-slave2

spark-env.sh
export JAVA_HOME=/opt/bigData/othersoftware/jdk1.8.0_221
export SPARK_MASTER_IP=192.168.99.100
export SPARK_WORKER_CORES=2
export SPARK_EXECUTOR_MEMORY=1g
export SPARK_WORKER_MEMORY=1g
export HADOOP_CONF_DIR=/opt/bigData/hadoop/hadoop-2.7.7/etc/hadoop

启动 spark start-all.sh

Spark-UI
http://192.168.31.100:8080/





启动 yarn
./satrt-yarn.sh

——

spark-shell(不加 --master spark://spark-master:7077 是 local 启动的)
spark-shell --master spark://spark-master:7077

spark.range(0,
50).select(**$"id"**.as(**"id"**)).repartition(1).write.mode(SaveMode.*Overwrite*)
.format(**"delta"**).save(path + table)