1. 大数据集群搭建

3.1 基本环境与 zookeeper 安装

本次集群搭建共有三个节点，包括一个主节点 master，和两个从节点 slave1 和 slave2。具体操作如下：

切换本地源

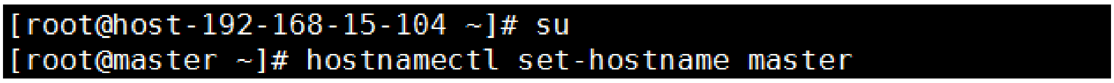
- 发信号给 yum 进程：pkill -9 yum
- 进入 yum 源配置文件：cd /etc/yum.repos.d
- 删除所有文件 rm-rf *
- 下载 yum 源：wget http://192.168.158.166/repofile/CentOS-Base.repo
- 清除 YUM 缓存：yum clean all

3.1.1 修改主机名（三台机器均执行）

（1）以主机点 master 为例，首次切换到 root 用户：su

（2）修改主机名为 master：

hostnamectl set-hostname master

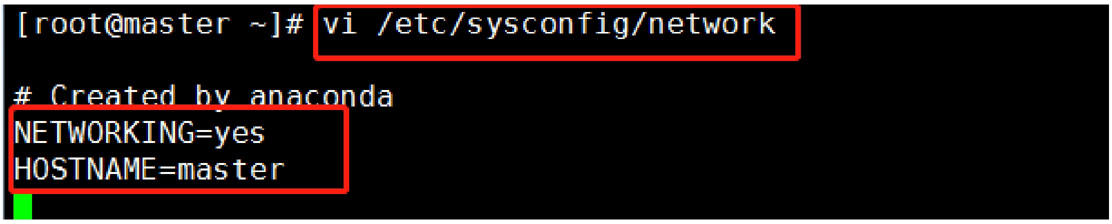


```
[root@host-192-168-15-104 ~]# su
[root@master ~]# hostnamectl set-hostname master
```

（3）永久修改主机名，编辑/etc/sysconfig/network 文件，内容如下：

NETWORKING=yes

HOSTNAME=master

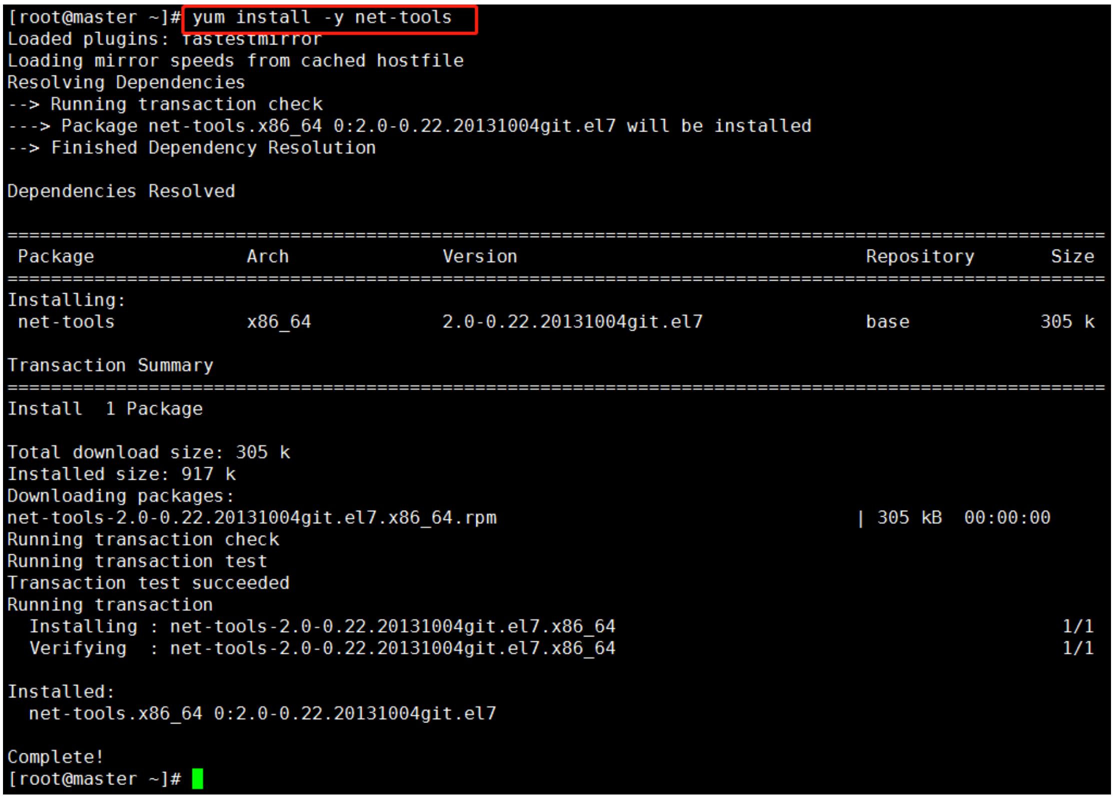


```
[root@master ~]# vi /etc/sysconfig/network

# Created by anaconda
NETWORKING=yes
HOSTNAME=master
```

注意保存退出。

（4）下载相关工具

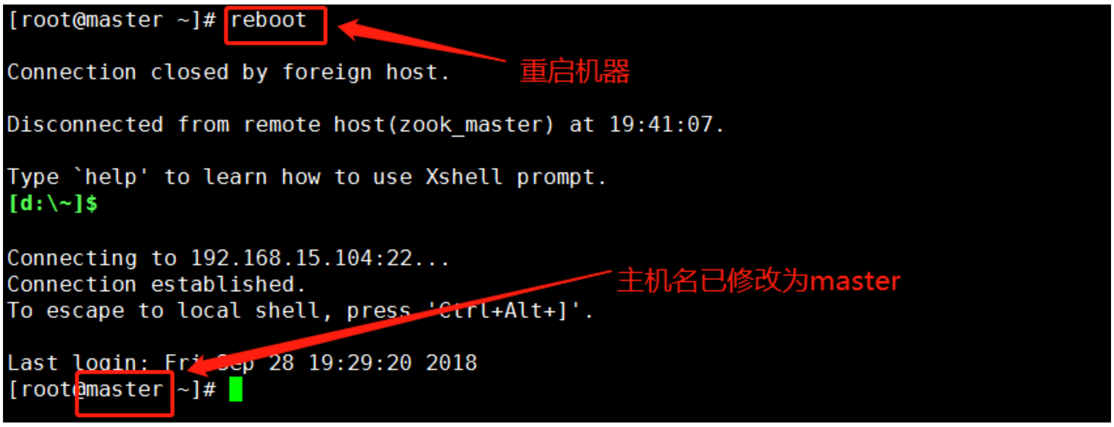
yum install -y net-tools



```
[root@master ~]# yum install -y net-tools
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
---> Package net-tools.x86_64 0:2.0-0.22.20131004git.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

==================================================================================================
 Package                Arch              Version                               Repository      Size
==================================================================================================
Installing:
 net-tools              x86_64            2.0-0.22.20131004git.el7              base            305 k

Transaction Summary
==================================================================================================
Install  1 Package

Total download size: 305 k
Installed size: 917 k
Downloading packages:
net-tools-2.0-0.22.20131004git.el7.x86_64.rpm                                 | 305 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : net-tools-2.0-0.22.20131004git.el7.x86_64                                      1/1
  Verifying  : net-tools-2.0-0.22.20131004git.el7.x86_64                                      1/1

Installed:
  net-tools.x86_64 0:2.0-0.22.20131004git.el7

Complete!
[root@master ~]#
```

（5）保存该文件，重启计算机：reboot

（6）查看是否生效：hostname



```
[root@master ~]# reboot

Connection closed by foreign host.

Disconnected from remote host(zook_master) at 19:41:07.

Type `help' to learn how to use Xshell prompt.
[d:\~]$

Connecting to 192.168.15.104:22...
Connection established.
To escape to local shell, press 'Ctrl+Alt+]'.

Last login: Fri Sep 28 19:29:20 2018
[root@master ~]#
```

重启机器

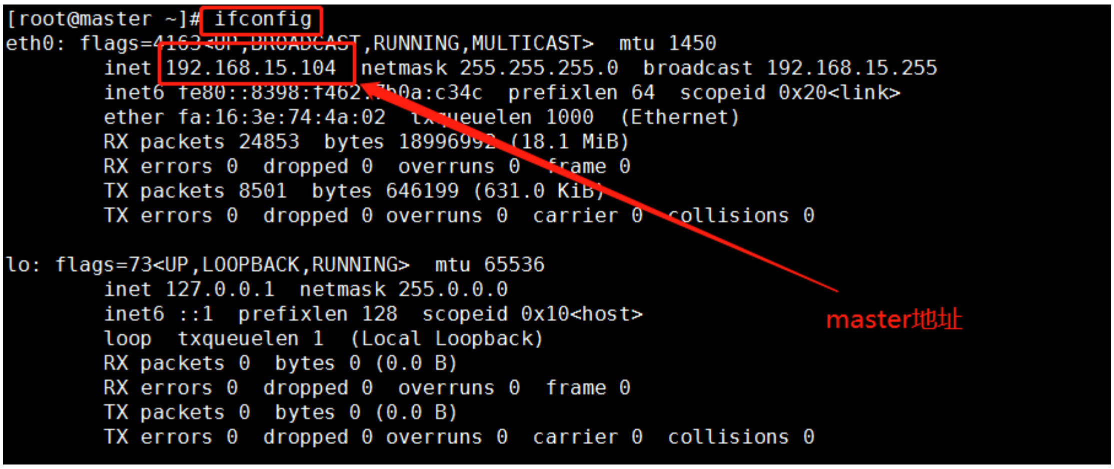
主机名已修改为master

3.1.2 配置 host 文件（三台机器）

使各个节点能使用对应的节点主机名连接对应的地址。

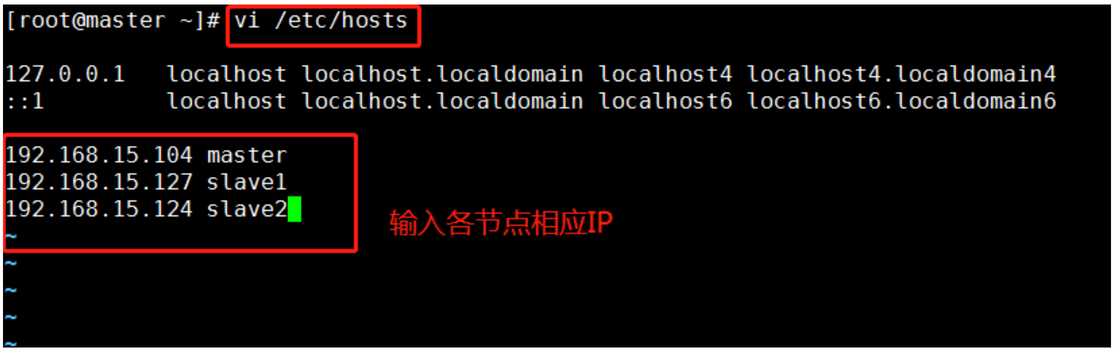
hosts 文件主要用于确定每个结点的 IP 地址，方便后续各结点能快速查到并访问。在上述 3 个虚机结点上均需要配置此文件。由于需要确定每个结点的 IP 地址，所以在配置 hosts 文件之前需要先查看当前虚机结点的 IP 地址是多少.

（1）可以通过 ifconfig 命令进行查看。



```
[root@master ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1450
        inet 192.168.15.104  netmask 255.255.255.0  broadcast 192.168.15.255
        inet6 fe80::8398:f462:7b0a:c34c  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:74:4a:02  txqueuelen 1000  (Ethernet)
        RX packets 24853  bytes 18996992 (18.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8501  bytes 646199 (631.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

（2）查看节点地址之后将三个节点的 ip 地址以及其对应的名称写进 hosts 文件。这里我们设置为 master、slave1、slave2。注意保存退出。



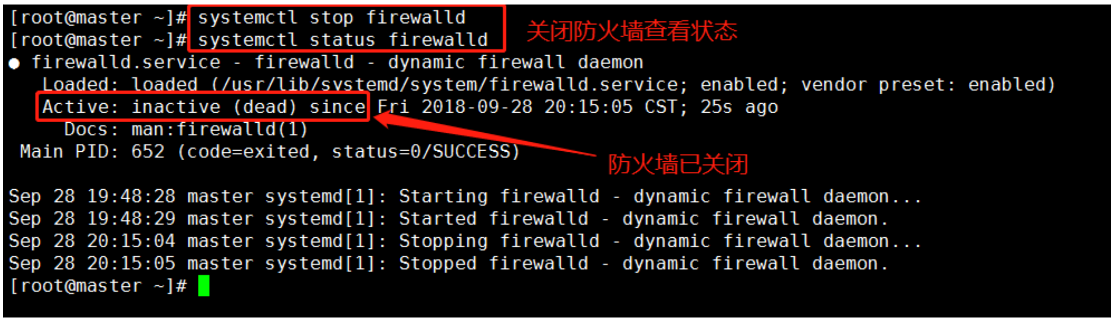
```
[root@master ~]# vi /etc/hosts

127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.15.104 master
192.168.15.127 slave1
192.168.15.124 slave2
~
~
~
~
~
```

3.1.3 关闭防火墙（三台机器）

centos7 中防火墙命令用 firewalld 取代了 iptables，当其状态是 dead 时，即防火墙关闭。

关闭防火墙：systemctl stop firewalld

查看状态：systemctl status firewalld

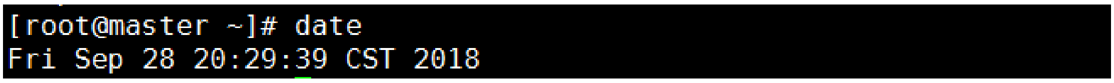


```
[root@master ~]# systemctl stop firewalld
[root@master ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Fri 2018-09-28 20:15:05 CST; 25s ago
     Docs: man:firewalld(1)
 Main PID: 652 (code=exited, status=0/SUCCESS)

Sep 28 19:48:28 master systemd[1]: Starting firewalld - dynamic firewall daemon...
Sep 28 19:48:29 master systemd[1]: Started firewalld - dynamic firewall daemon.
Sep 28 20:15:04 master systemd[1]: Stopping firewalld - dynamic firewall daemon...
Sep 28 20:15:05 master systemd[1]: Stopped firewalld - dynamic firewall daemon.
[root@master ~]#
```

3.1.4 时间同步

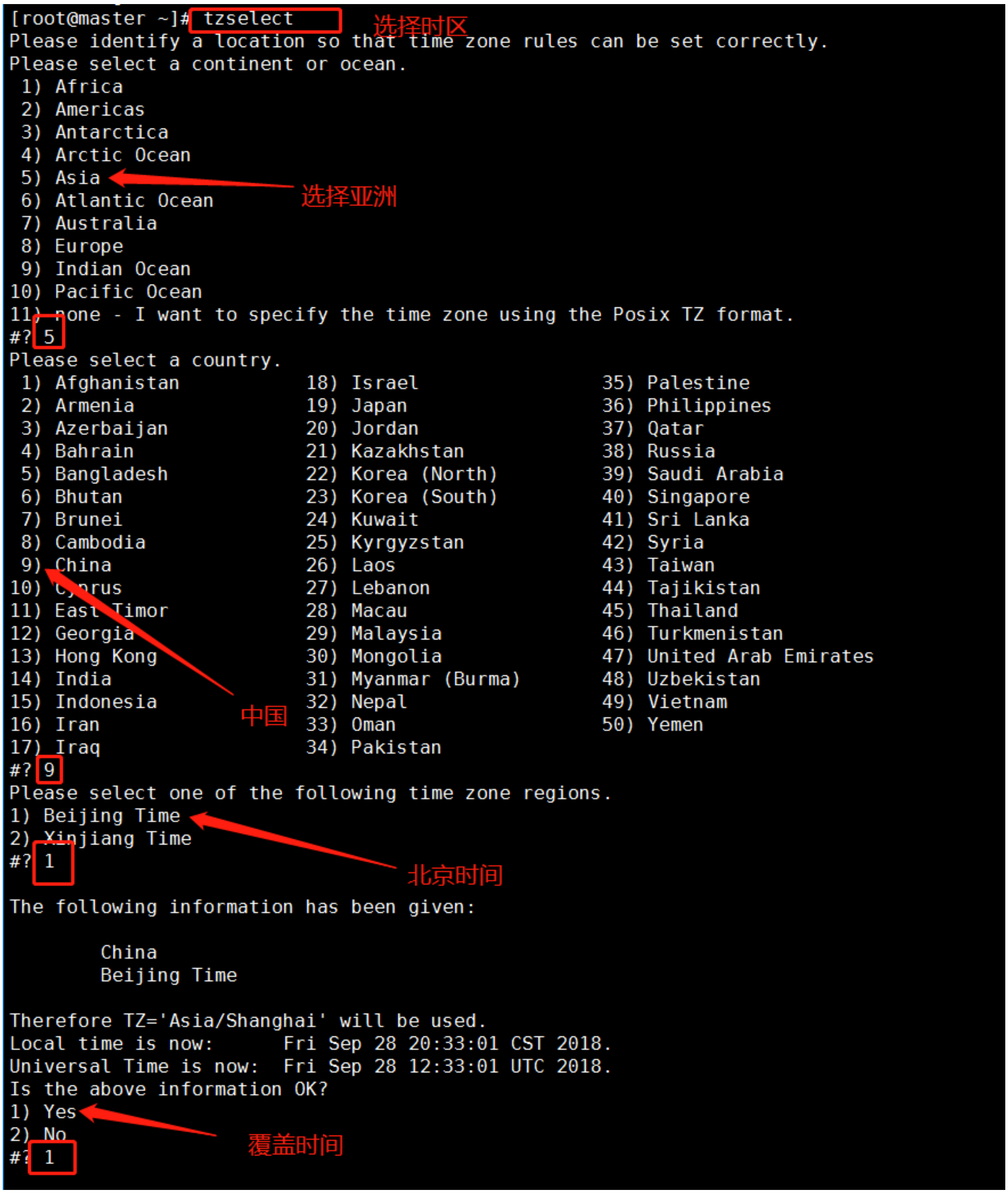
（1）时区一致。要保证设置主机时间准确，每台机器时区必须一致。实验中我们需要同步网络时间，因此要首先选择一样的时区。先确保时区一样，否则同步以后时间也是有时区差。

可以使用 date 命令查看自己的机器时间。



```
[root@master ~]# date
Fri Sep 28 20:29:39 CST 2018
```

（2）选择时区：tzselect

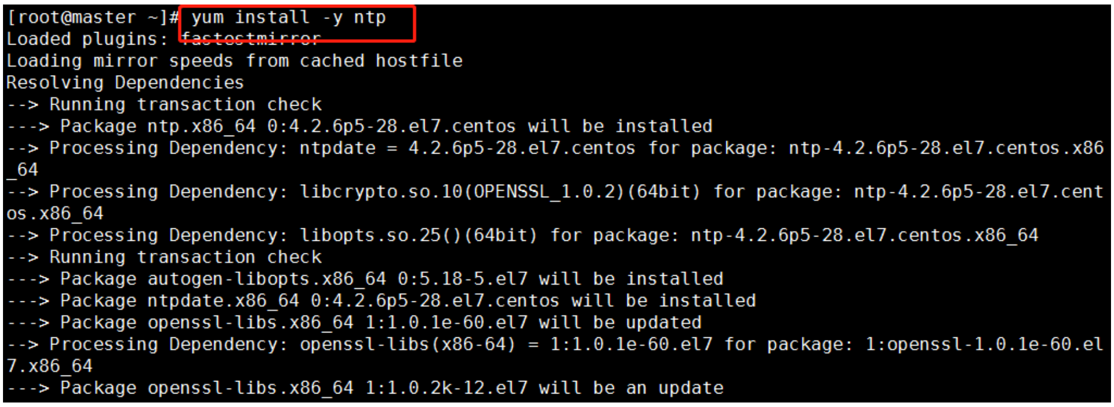


```
[root@master ~]# tzselect
Please identify a location so that time zone rules can be set correctly.
Please select a continent or ocean.
 1) Africa
 2) Americas
 3) Antarctica
 4) Arctic Ocean
 5) Asia
 6) Atlantic Ocean
 7) Australia
 8) Europe
 9) Indian Ocean
10) Pacific Ocean
11) none - I want to specify the time zone using the Posix TZ format.
#? 5
Please select a country.
 1) Afghanistan           18) Israel                35) Palestine
 2) Armenia               19) Japan                 36) Philippines
 3) Azerbaijan            20) Jordan                37) Qatar
 4) Bahrain               21) Kazakhstan            38) Russia
 5) Bangladesh            22) Korea (North)         39) Saudi Arabia
 6) Bhutan                23) Korea (South)         40) Singapore
 7) Brunei                24) Kuwait                41) Sri Lanka
 8) Cambodia              25) Kyrgyzstan            42) Syria
 9) China                 26) Laos                  43) Taiwan
10) Cyprus                27) Lebanon               44) Tajikistan
11) East Timor            28) Macau                 45) Thailand
12) Georgia               29) Malaysia              46) Turkmenistan
13) Hong Kong             30) Mongolia              47) United Arab Emirates
14) India                 31) Myanmar (Burma)       48) Uzbekistan
15) Indonesia             32) Nepal                 49) Vietnam
16) Iran                  33) Oman                  50) Yemen
17) Iraq                  34) Pakistan
#? 9
Please select one of the following time zone regions.
1) Beijing Time
2) Xinjiang Time
#? 1

The following information has been given:

        China
        Beijing Time

Therefore TZ='Asia/Shanghai' will be used.
Local time is now:      Fri Sep 28 20:33:01 CST 2018.
Universal Time is now:  Fri Sep 28 12:33:01 UTC 2018.
Is the above information OK?
1) Yes
2) No
#? 1
```

由于 hadoop 集群对时间要求很高，所以集群内主机要经常同步。我们使用 ntp 网络时间协议进行时间同步，master 作为 ntp 服务器，其余的当做 ntp 客户端。

（3）下载 ntp（三台机器）

yum install –y ntp



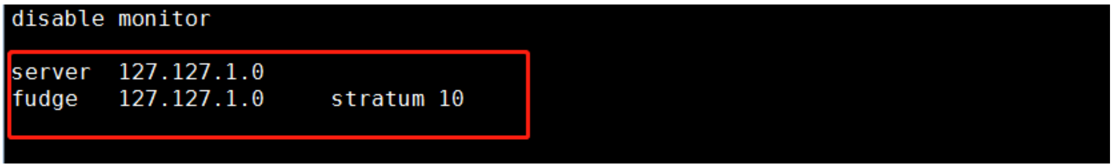
```
[root@master ~]# yum install -y ntp
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
Resolving Dependencies
--> Running transaction check
---> Package ntp.x86_64 0:4.2.6p5-28.el7.centos will be installed
--> Processing Dependency: ntpdate = 4.2.6p5-28.el7.centos for package: ntp-4.2.6p5-28.el7.centos.x86
_64
--> Processing Dependency: libcrypto.so.10(OPENSSL_1.0.2)(64bit) for package: ntp-4.2.6p5-28.el7.cent
os.x86_64
--> Processing Dependency: libopts.so.25()(64bit) for package: ntp-4.2.6p5-28.el7.centos.x86_64
--> Running transaction check
---> Package autogen-libopts.x86_64 0:5.18-5.el7 will be installed
---> Package ntpdate.x86_64 0:4.2.6p5-28.el7.centos will be installed
---> Package openssl-libs.x86_64 1:1.0.1e-60.el7 will be updated
--> Processing Dependency: openssl-libs(x86-64) = 1:1.0.1e-60.el7 for package: 1:openssl-1.0.1e-60.el
7.x86_64
---> Package openssl-libs.x86_64 1:1.0.2k-12.el7 will be an update
```

（4）master 作为 ntp 服务器，修改 ntp 配置文件。（master 上执行）

vi /etc/ntp.conf

server　127.127.1.0　　　　　　　　# local clock

fudge　　127.127.1.0　　stratum 10　　#stratum 设置为其它值也是可以的，其范围为 0~15



```
disable monitor

server  127.127.1.0
fudge   127.127.1.0     stratum 10
```

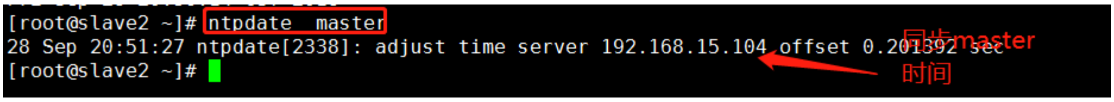
重启 ntp 服务。

/bin/systemctl restart　ntpd.service

（5）　其他机器同步（slave1，slave2）

等待大概五分钟，再到其他机上同步该 master 服务器时间。

ntpdate　master



```
[root@slave2 ~]# ntpdate  master
28 Sep 20:51:27 ntpdate[2338]: adjust time server 192.168.15.104 offset 0.201392 sec
[root@slave2 ~]# 
```

如果配置平台式没有外网连接可以将三台机器设为统一时间，输入命令：

date -s 10:00（时间）

3.1.5 配置 ssh 免密

SSH 主要通过 RSA 算法来产生公钥与私钥，在数据传输过程中对数据进行加密来保障数

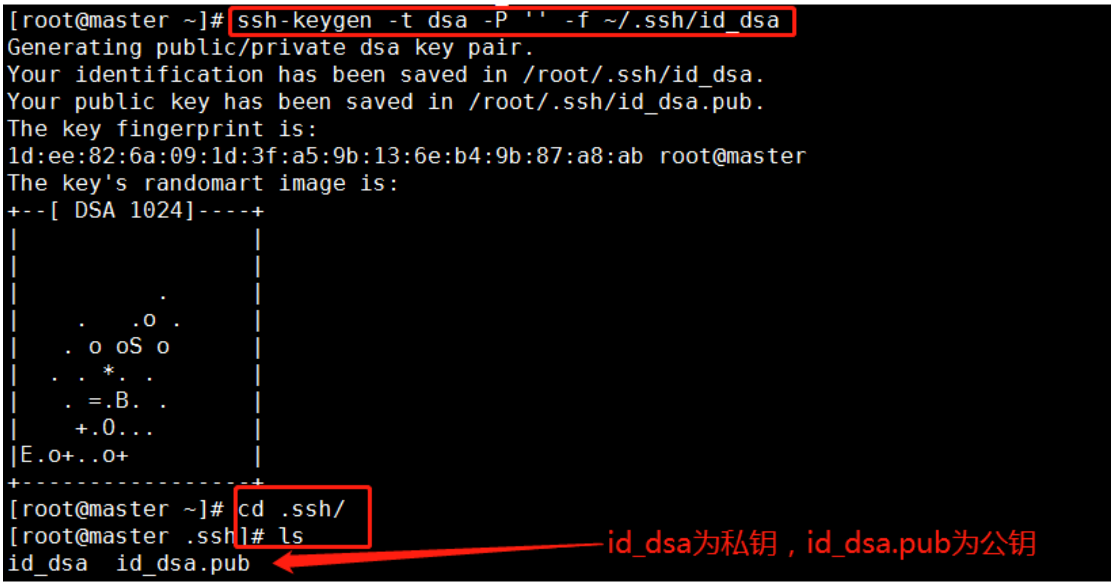
据的安全性和可靠性，公钥部分是公共部分，网络上任一结点均可以访问，私钥主要用于对数据进行加密，以防他人盗取数据。总而言之，这是一种非对称算法，想要破解还是非常有难度的。Hadoop 集群的各个结点之间需要进行数据的访问，被访问的结点对于访问用户结点的可靠性必须进行验证，hadoop 采用的是 ssh 的方法通过密钥验证及数据加解密的方式进行远程安全登录操作，当然，如果 hadoop 对每个结点的访问均需要进行验证，其效率将会大大降低，所以才需要配置 SSH 免密码的方法直接远程连入被访问结点，这样将大大提高访问效率。

（1） 每个结点分别产生公私密钥：

ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa（三台机器）

秘钥产生目录在用户主目录下的.ssh 目录中，进入相应目录查看：

cd .ssh/

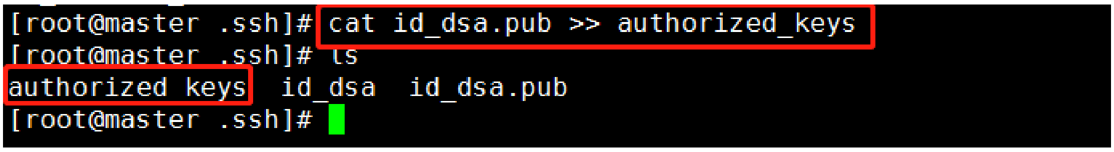


```
[root@master ~]# ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
Generating public/private dsa key pair.
Your identification has been saved in /root/.ssh/id_dsa.
Your public key has been saved in /root/.ssh/id_dsa.pub.
The key fingerprint is:
1d:ee:82:6a:09:1d:3f:a5:9b:13:6e:b4:9b:87:a8:ab root@master
The key's randomart image is:
+--[ DSA 1024]----+
|                 |
|                 |
|          .      |
|    .   .o .     |
|   . o oS o      |
|  . . *. .       |
|   . =.B. .      |
|    +.O...       |
|E.o+..o+         |
+-----------------+
[root@master ~]# cd .ssh/
[root@master .ssh]# ls
id_dsa  id_dsa.pub
```

id_dsa为私钥，id_dsa.pub为公钥

（2）Id_dsa.pub 为公钥，id_dsa 为私钥，紧接着将公钥文件复制成 authorized_keys 文件：（仅 master）

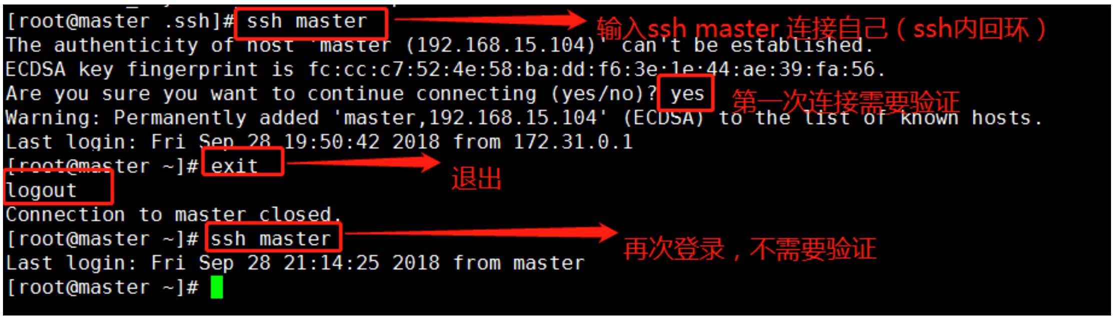
cat id_dsa.pub >> authorized_keys（注意在.ssh/路径下操作）



```
[root@master .ssh]# cat id_dsa.pub >> authorized_keys
[root@master .ssh]# ls
authorized_keys  id_dsa  id_dsa.pub
[root@master .ssh]# 
```

在主机上连接自己，也叫做 ssh 内回环。

ssh master



```
[root@master .ssh]# ssh master
The authenticity of host 'master (192.168.15.104)' can't be established.
ECDSA key fingerprint is fc:cc:c7:52:4e:58:ba:dd:f6:3e:1e:44:ae:39:fa:56.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'master,192.168.15.104' (ECDSA) to the list of known hosts.
Last login: Fri Sep 28 19:50:42 2018 from 172.31.0.1
[root@master ~]# exit
logout
Connection to master closed.
[root@master ~]# ssh master
Last login: Fri Sep 28 21:14:25 2018 from master
[root@master ~]# 
```

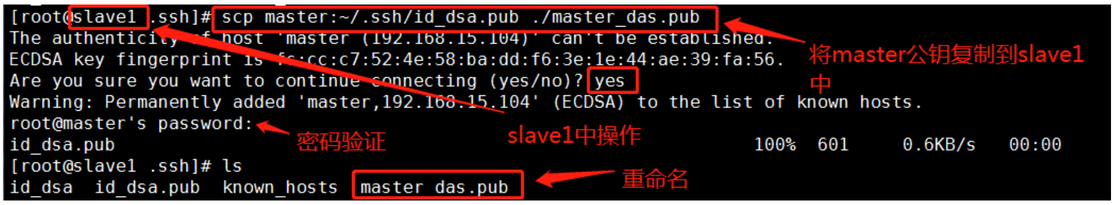
（3）让主结点 master 能通过 SSH 免密码登录两个子结点 slave。（slave 中操作）

为了实现这个功能，两个 slave 结点的公钥文件中必须要包含主结点的公钥信息，这样

当 master 就可以顺利安全地访问这两个 slave 结点了。

slave1 结点通过 scp 命令远程登录 master 结点，并复制 master 的公钥文件到当前的目录下，且重命名为 master_das.pub，这一过程需要密码验证。

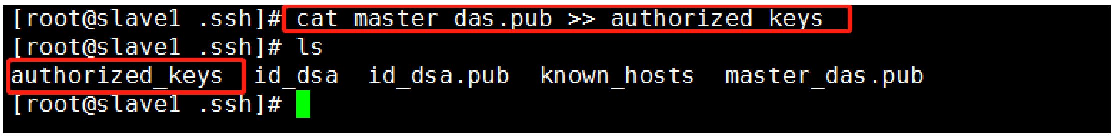
scp master:~/.ssh/id_dsa.pub ./master_das.pub



```
[root@slave1 .ssh]# scp master:~/.ssh/id_dsa.pub ./master_das.pub
The authenticity of host 'master (192.168.15.104)' can't be established.
ECDSA key fingerprint is fc:cc:c7:52:4e:58:ba:dd:f6:3e:1e:44:ae:39:fa:56.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'master,192.168.15.104' (ECDSA) to the list of known hosts.
root@master's password:
id_dsa.pub                                    100%  601     0.6KB/s   00:00
[root@slave1 .ssh]# ls
id_dsa  id_dsa.pub  known_hosts  master_das.pub
```

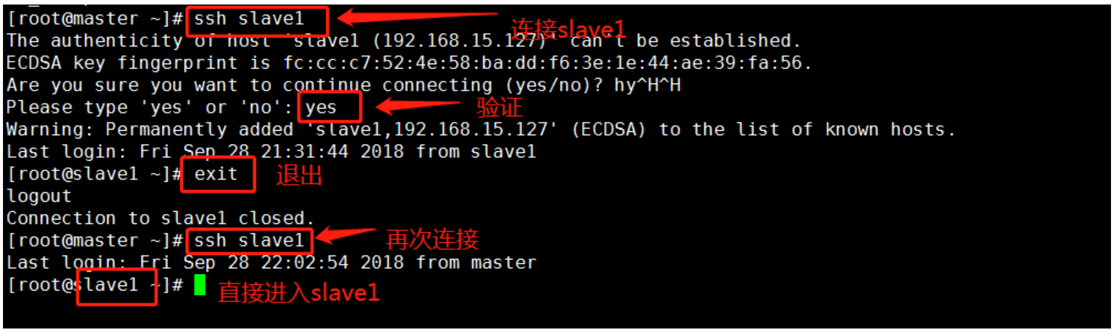
将 master 结点的公钥文件追加至 authorized_keys 文件：

cat master_dsa.pub >> authorized_keys



```
[root@slave1 .ssh]# cat master_das.pub >> authorized_keys
[root@slave1 .ssh]# ls
authorized_keys  id_dsa  id_dsa.pub  known_hosts  master_das.pub
[root@slave1 .ssh]#
```

这时，master 就可以连接 slave1 了。



```
[root@master ~]# ssh slave1
The authenticity of host 'slave1 (192.168.15.127)' can't be established.
ECDSA key fingerprint is fc:cc:c7:52:4e:58:ba:dd:f6:3e:1e:44:ae:39:fa:56.
Are you sure you want to continue connecting (yes/no)? hy^H^H
Please type 'yes' or 'no': yes
Warning: Permanently added 'slave1,192.168.15.127' (ECDSA) to the list of known hosts.
Last login: Fri Sep 28 21:31:44 2018 from slave1
[root@slave1 ~]# exit
logout
Connection to slave1 closed.
[root@master ~]# ssh slave1
Last login: Fri Sep 28 22:02:54 2018 from master
[root@slave1 ~]#
```

slave1 结点首次连接时需要，“yes”确认连接，这意味着 master 结点连接 slave1 结点时需要人工询问，无法自动连接，输入 yes 后成功接入，紧接着注销退出至 master 结点。

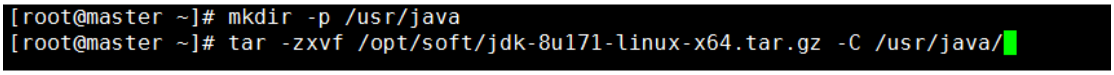
同理 slave2 中也是这么操作。

注意：两个结点的 ssh 免密码登录已经配置成功，还需要对主结点 master 也要进行上面的同样工作，因为 jobtracker 有可能会分布在其它结点上，jobtracker 有不存在 master 结点上的可能性。在上一步骤中，我们已经进行过此操作，这里仅做提醒。

3.1.6 安装 JDK（三台机器）

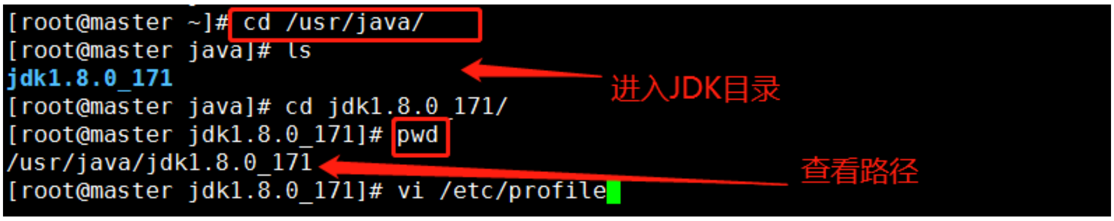
（1）首先建立工作路径/usr/java。

```
mkdir -p /usr/java

tar -zxvf /opt/soft/jdk-8u171-linux-x64.tar.gz -C /usr/java/
```



```
[root@master ~]# mkdir -p /usr/java
[root@master ~]# tar -zxvf /opt/soft/jdk-8u171-linux-x64.tar.gz -C /usr/java/
```

2.修改环境变量

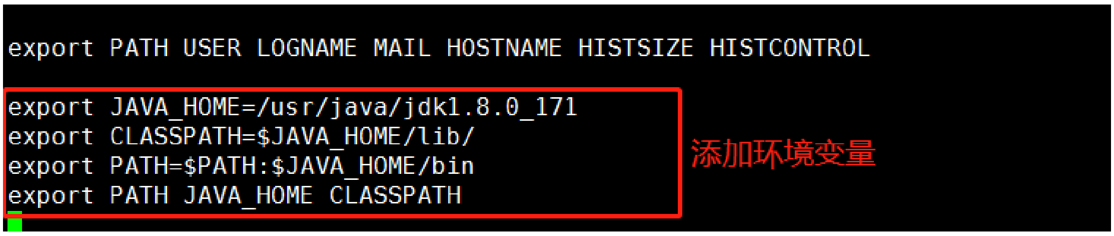


```
[root@master ~]# cd /usr/java/
[root@master java]# ls
jdk1.8.0_171
[root@master java]# cd jdk1.8.0_171/
[root@master jdk1.8.0_171]# pwd
/usr/java/jdk1.8.0_171
[root@master jdk1.8.0_171]# vi /etc/profile
```

修改环境变量：vi /etc/profile

添加内容如下：

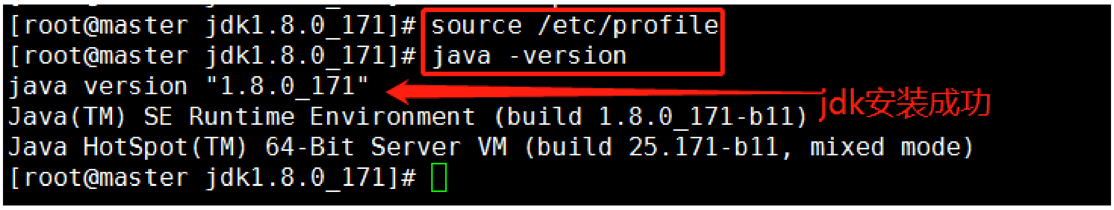
```
export JAVA_HOME=/usr/java/jdk1.8.0_171

export CLASSPATH=$JAVA_HOME/lib/

export PATH=$PATH:$JAVA_HOME/bin

export PATH JAVA_HOME CLASSPATH
```



```
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL

export JAVA_HOME=/usr/java/jdk1.8.0_171
export CLASSPATH=$JAVA_HOME/lib/
export PATH=$PATH:$JAVA_HOME/bin
export PATH JAVA_HOME CLASSPATH
```

生效环境变量：source /etc/profile

查看 java 版本：java -version

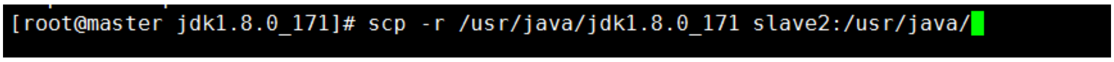


```
[root@master jdk1.8.0_171]# source /etc/profile
[root@master jdk1.8.0_171]# java -version
java version "1.8.0_171"
Java(TM) SE Runtime Environment (build 1.8.0_171-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.171-b11, mixed mode)
[root@master jdk1.8.0_171]#
```

同理 slave 节点，相同安装步骤。

注意：如果在 slave 节点中安装较慢，可以使用 scp 命令，将相同的文件从 master 中复制过来。

在 master 中将 JDK 复制到 slave2 中（要保证 slave2 中已有相应目录）。



```
[root@master jdk1.8.0_171]# scp -r /usr/java/jdk1.8.0_171 slave2:/usr/java/
```

3.2 安装 zookeeper

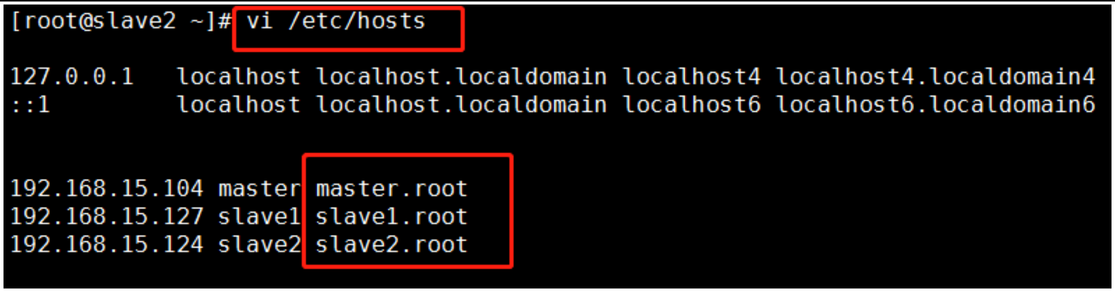
（1）修改主机名称到 IP 地址映射配置。

vi /etc/hosts

192.168.15.104 master master.root

192.168.15.127 slave1 slave1.root

192.168.15.124 slave2 slave2.root

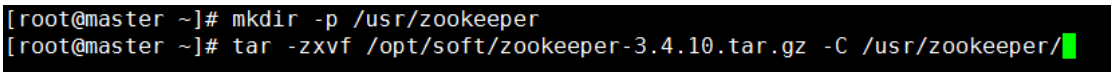


```
[root@slave2 ~]# vi /etc/hosts

127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6


192.168.15.104 master master.root
192.168.15.127 slave1 slave1.root
192.168.15.124 slave2 slave2.root
```

（2）修改 ZooKeeper 配置文件。在其中 master 机器上，用 tar -zxvf 命令解压缩 zookeeper-3.4.10.tar.gz。

创建工作目录：mkdir -p /usr/zookeeper

解压：tar -zxvf /opt/soft/zookeeper-3.4.10.tar.gz -C /usr/zookeeper/



```
[root@master ~]# mkdir -p /usr/zookeeper
[root@master ~]# tar -zxvf /opt/soft/zookeeper-3.4.10.tar.gz -C /usr/zookeeper/
```

（3）配置文件 conf/zoo.cfg

用 cd 命令进入 zookeeper-3.4.10/conf 目录下，将 zoo_sample.cfg 文件拷贝一份，命名为为“zoo.cfg”。

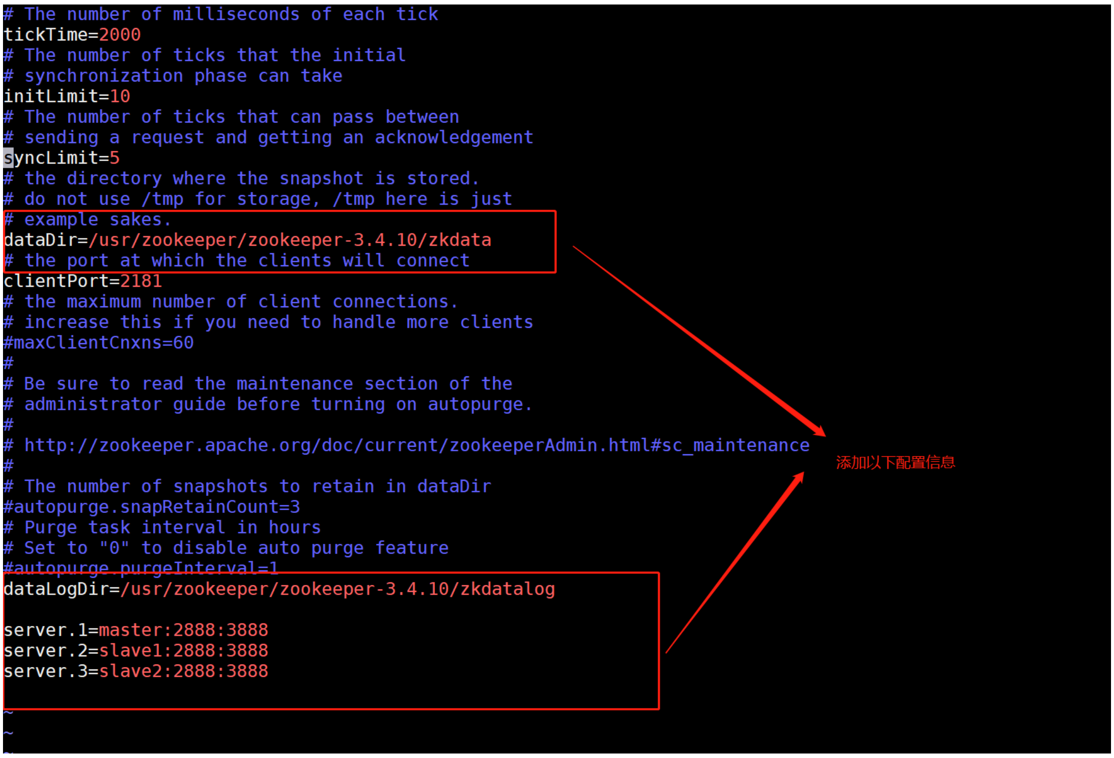
scp zoo_sample.cfg zoo.cfg

Zoo.cfg 文件配置

tickTime=2000

initLimit=10

```
syncLimit=5

dataDir=/usr/zookeeper/zookeeper-3.4.10/zkdata

clientPort=2181

dataLogDir=/usr/zookeeper/zookeeper-3.4.10/zkdatalog

server.1=master:2888:3888

server.2=slave1:2888:3888

server.3=slave2:2888:3888
```

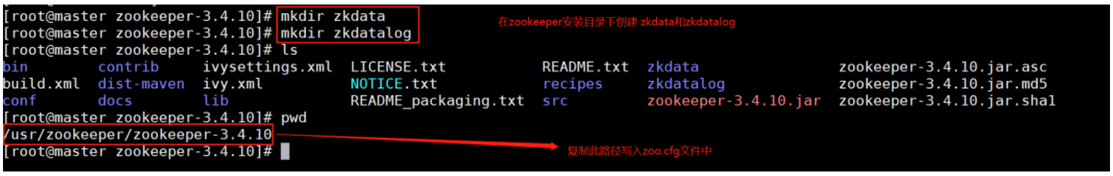


```
# The number of milliseconds of each tick
tickTime=2000
# The number of ticks that the initial
# synchronization phase can take
initLimit=10
# The number of ticks that can pass between
# sending a request and getting an acknowledgement
syncLimit=5
# the directory where the snapshot is stored.
# do not use /tmp for storage, /tmp here is just
# example sakes.
dataDir=/usr/zookeeper/zookeeper-3.4.10/zkdata
# the port at which the clients will connect
clientPort=2181
# the maximum number of client connections.
# increase this if you need to handle more clients
#maxClientCnxns=60
#
# Be sure to read the maintenance section of the
# administrator guide before turning on autopurge.
#
# http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_maintenance
#
# The number of snapshots to retain in dataDir
#autopurge.snapRetainCount=3
# Purge task interval in hours
# Set to "0" to disable auto purge feature
#autopurge.purgeInterval=1
dataLogDir=/usr/zookeeper/zookeeper-3.4.10/zkdatalog

server.1=master:2888:3888
server.2=slave1:2888:3888
server.3=slave2:2888:3888

~
~
```

添加以下配置信息

（4）在 zookeeper 的目录中，创建 zkdata 和 zkdatalog 两个文件夹。zkdatalog 文件夹，是为了指定 zookeeper 产生日志指定相应的路径。

```
mkdir zkdata

mkdir zkdatalog
```

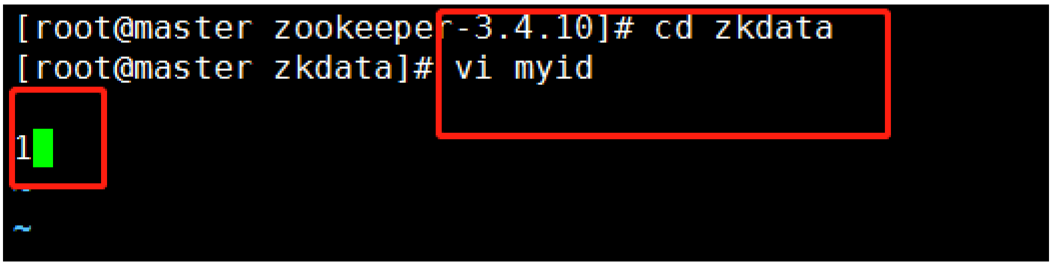


```
[root@master zookeeper-3.4.10]# mkdir zkdata
[root@master zookeeper-3.4.10]# mkdir zkdatalog
[root@master zookeeper-3.4.10]# ls
bin         contrib     ivysettings.xml  LICENSE.txt           README.txt  zkdata                    zookeeper-3.4.10.jar.asc
build.xml   dist-maven  ivy.xml          NOTICE.txt            recipes     zkdatalog                 zookeeper-3.4.10.jar.md5
conf        docs        lib              README_packaging.txt  src         zookeeper-3.4.10.jar      zookeeper-3.4.10.jar.sha1
[root@master zookeeper-3.4.10]# pwd
/usr/zookeeper/zookeeper-3.4.10
[root@master zookeeper-3.4.10]# 
```

在zookeeper安装目录下创建 zkdata和zkdatalog

复制此路径写入zoo.cfg文件中

（5）进入 zkdata 文件夹，创建文件 myid。

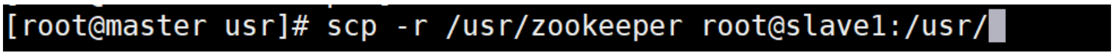


```
[root@master zookeeper-3.4.10]# cd zkdata
[root@master zkdata]# vi myid

1
~
~
```

（6）远程复制分发安装文件

上面已经在一台机器 master 上配置完成 ZooKeeper，现在可以将该配置好的安装文件远程拷贝到集群中的各个结点对应的目录下：

```
scp -r /usr/zookeeper root@slave1:/usr/

scp -r /usr/zookeeper root@slave2:/usr/
```

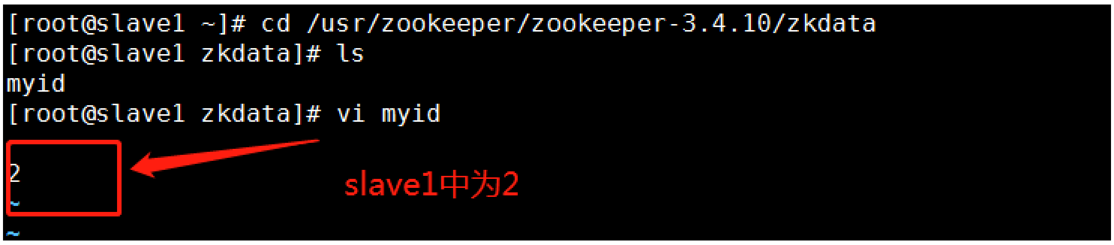


```
[root@master usr]# scp -r /usr/zookeeper root@slave1:/usr/
```

（7）设置 myid。在我们配置的 dataDir 指定的目录下面，创建一个 myid 文件，里面内容为一个数字，用来标识当前主机，conf/zoo.cfg 文件中配置的 server.X 中 X 为什么数字，则 myid 文件中就输入这个数字。

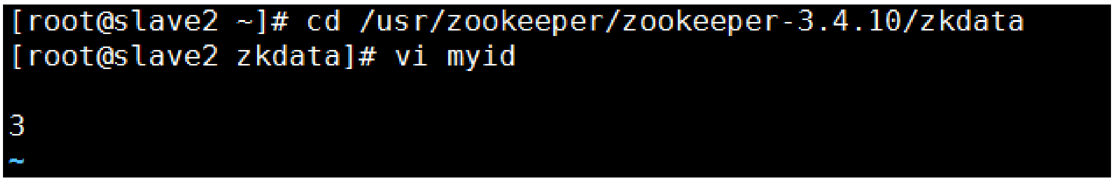
slave1 中为 2；slave2 中为 3。

```
cd /usr/zookeeper/zookeeper-3.4.10/zkdata
```



```
[root@slave1 ~]# cd /usr/zookeeper/zookeeper-3.4.10/zkdata
[root@slave1 zkdata]# ls
myid
[root@slave1 zkdata]# vi myid

2
~
~
```

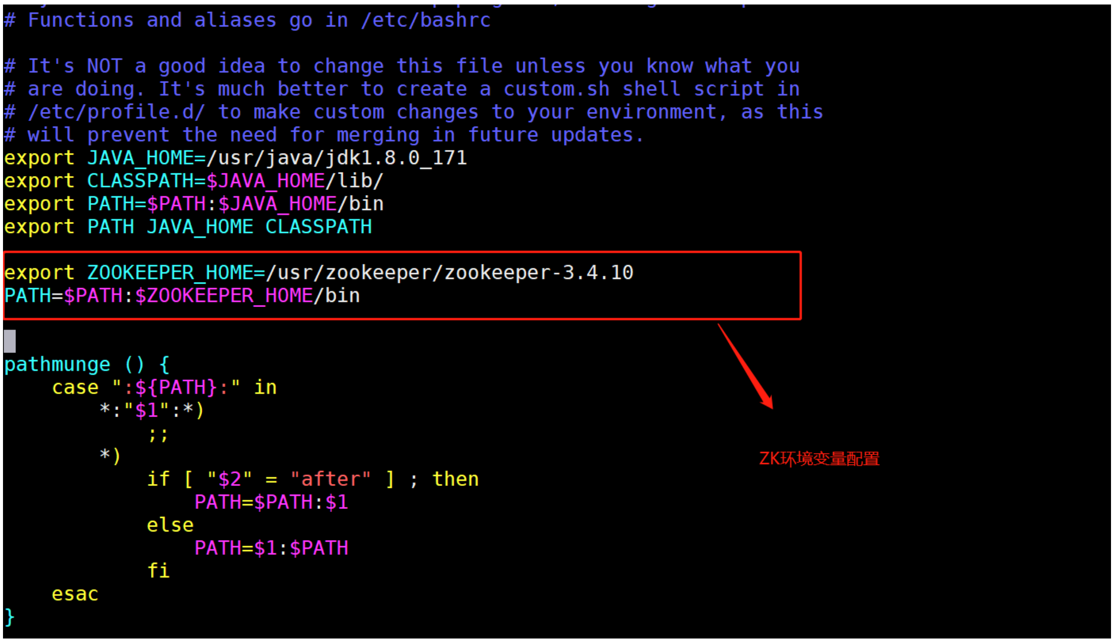
slave1中为2



```
[root@slave2 ~]# cd /usr/zookeeper/zookeeper-3.4.10/zkdata
[root@slave2 zkdata]# vi myid

3
~
```

（8）配置环境变量并启动 ZooKeeper。在每台机器上操作如下：

```
vi /etc/profile

#set zookeeper environment

export ZOOKEEPER_HOME=/usr/zookeeper/zookeeper-3.4.10

PATH=$PATH:$ZOOKEEPER_HOME/bin
```



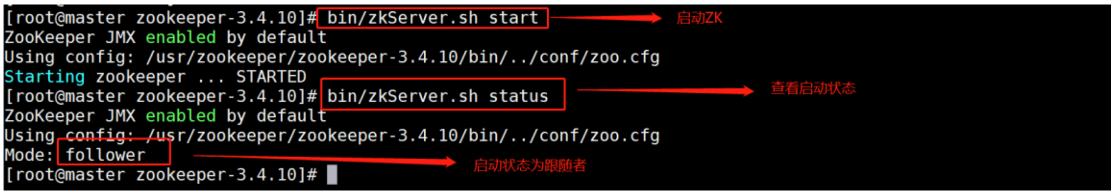
```
# Functions and aliases go in /etc/bashrc

# It's NOT a good idea to change this file unless you know what you
# are doing. It's much better to create a custom.sh shell script in
# /etc/profile.d/ to make custom changes to your environment, as this
# will prevent the need for merging in future updates.
export JAVA_HOME=/usr/java/jdk1.8.0_171
export CLASSPATH=$JAVA_HOME/lib/
export PATH=$PATH:$JAVA_HOME/bin
export PATH JAVA_HOME CLASSPATH

export ZOOKEEPER_HOME=/usr/zookeeper/zookeeper-3.4.10
PATH=$PATH:$ZOOKEEPER_HOME/bin


pathmunge () {
    case ":${PATH}:" in
        *:"$1":*)
            ;;
        *)
            if [ "$2" = "after" ] ; then
                PATH=$PATH:$1
            else
                PATH=$1:$PATH
            fi
    esac
}
```

生效：source /etc/profile

（9）启动 ZooKeeper 集群

在 ZooKeeper 集群的每个结点上，执行启动 ZooKeeper 服务的脚本，如下所示：

```
bin/zkServer.sh start

bin/zkServer.sh status
```

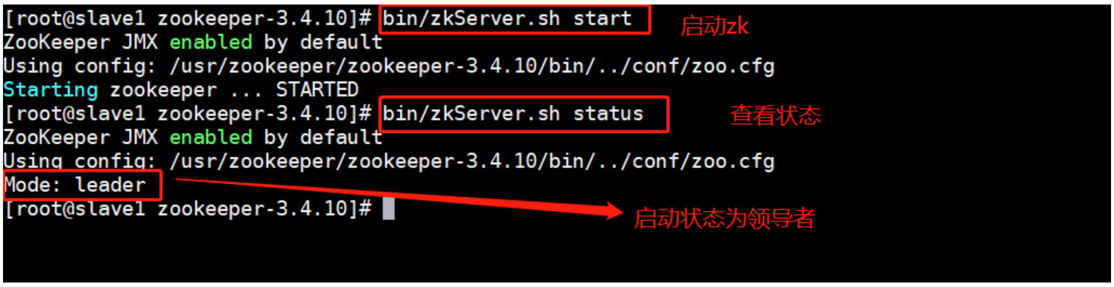


```
[root@master zookeeper-3.4.10]# bin/zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /usr/zookeeper/zookeeper-3.4.10/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
[root@master zookeeper-3.4.10]# bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/zookeeper/zookeeper-3.4.10/bin/../conf/zoo.cfg
Mode: follower
[root@master zookeeper-3.4.10]# 
```

启动ZK

查看启动状态

启动状态为跟随者

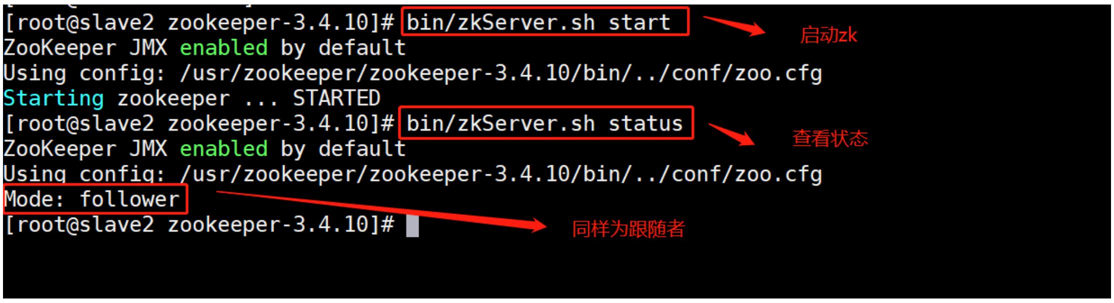


```
[root@slave1 zookeeper-3.4.10]# bin/zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /usr/zookeeper/zookeeper-3.4.10/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
[root@slave1 zookeeper-3.4.10]# bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/zookeeper/zookeeper-3.4.10/bin/../conf/zoo.cfg
Mode: leader
[root@slave1 zookeeper-3.4.10]# 
```

启动zk

查看状态

启动状态为领导者



```
[root@slave2 zookeeper-3.4.10]# bin/zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /usr/zookeeper/zookeeper-3.4.10/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
[root@slave2 zookeeper-3.4.10]# bin/zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /usr/zookeeper/zookeeper-3.4.10/bin/../conf/zoo.cfg
Mode: follower
[root@slave2 zookeeper-3.4.10]# 
```

启动zk

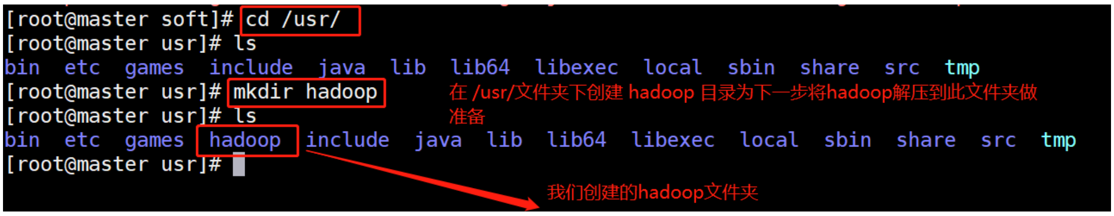
查看状态

同样为跟随者

通过上面状态查询结果可见，一个节点是 Leader，其余的结点是 Follower。

3.3 安装 hadoop

（1）创建对应工作目录/usr/hadoop：

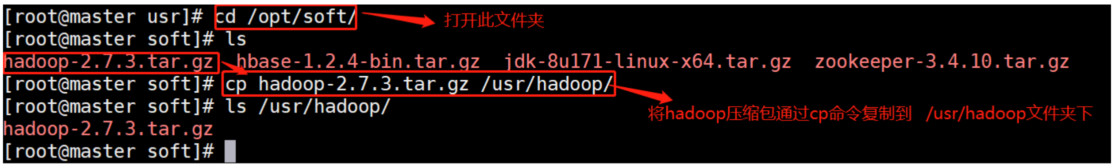


```
[root@master soft]# cd /usr/
[root@master usr]# ls
bin  etc  games  include  java  lib  lib64  libexec  local  sbin  share  src  tmp
[root@master usr]# mkdir hadoop
[root@master usr]# ls
bin  etc  games  hadoop  include  java  lib  lib64  libexec  local  sbin  share  src  tmp
[root@master usr]# 
```

在 /usr/文件夹下创建 hadoop 目录为下一步将hadoop解压到此文件夹做准备

我们创建的hadoop文件夹

解压 hadoop 到相应目录：

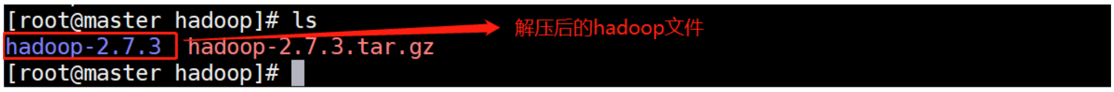


```
[root@master usr]# cd /opt/soft/
[root@master soft]# ls
hadoop-2.7.3.tar.gz  hbase-1.2.4-bin.tar.gz  jdk-8u171-linux-x64.tar.gz  zookeeper-3.4.10.tar.gz
[root@master soft]# cp hadoop-2.7.3.tar.gz /usr/hadoop/
[root@master soft]# ls /usr/hadoop/
hadoop-2.7.3.tar.gz
[root@master soft]# 
```

打开此文件夹

将hadoop压缩包通过cp命令复制到 /usr/hadoop文件夹下

解压后：

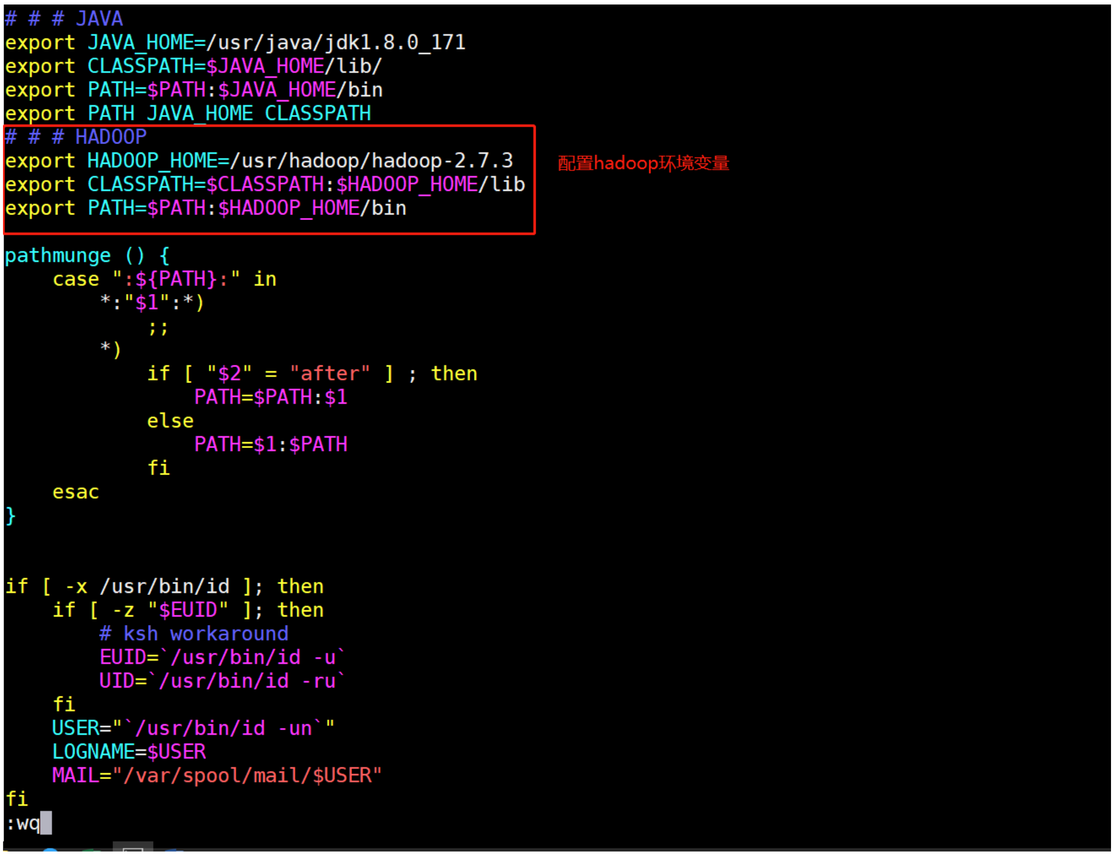


```
[root@master hadoop]# ls
hadoop-2.7.3  hadoop-2.7.3.tar.gz
[root@master hadoop]# 
```

解压后的hadoop文件

配置环境变量:

```
vim    /etc/profile

export HADOOP_HOME=/usr/hadoop/hadoop-2.7.3

export CLASSPATH=$CLASSPATH:$HADOOP_HOME/lib

export PATH=$PATH:$HADOOP_HOME/bin
```



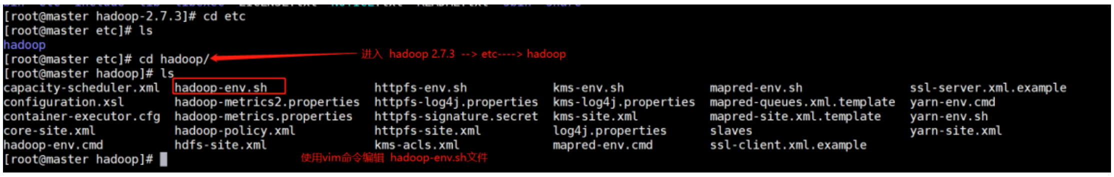
```
# # # JAVA
export JAVA_HOME=/usr/java/jdk1.8.0_171
export CLASSPATH=$JAVA_HOME/lib/
export PATH=$PATH:$JAVA_HOME/bin
export PATH JAVA_HOME CLASSPATH
# # # HADOOP
export HADOOP_HOME=/usr/hadoop/hadoop-2.7.3
export CLASSPATH=$CLASSPATH:$HADOOP_HOME/lib
export PATH=$PATH:$HADOOP_HOME/bin

pathmunge () {
    case ":${PATH}:" in
        *:"$1":*)
            ;;
        *)
            if [ "$2" = "after" ] ; then
                PATH=$PATH:$1
            else
                PATH=$1:$PATH
            fi
    esac
}


if [ -x /usr/bin/id ]; then
    if [ -z "$EUID" ]; then
        # ksh workaround
        EUID=`/usr/bin/id -u`
        UID=`/usr/bin/id -ru`
    fi
    USER="`/usr/bin/id -un`"
    LOGNAME=$USER
    MAIL="/var/spool/mail/$USER"
fi
:wq
```

配置hadoop环境变量

使用以下命令使 profile 生效:

source /etc/profile

（2）编辑 hadoop 环境配置文件 hadoop-env.sh

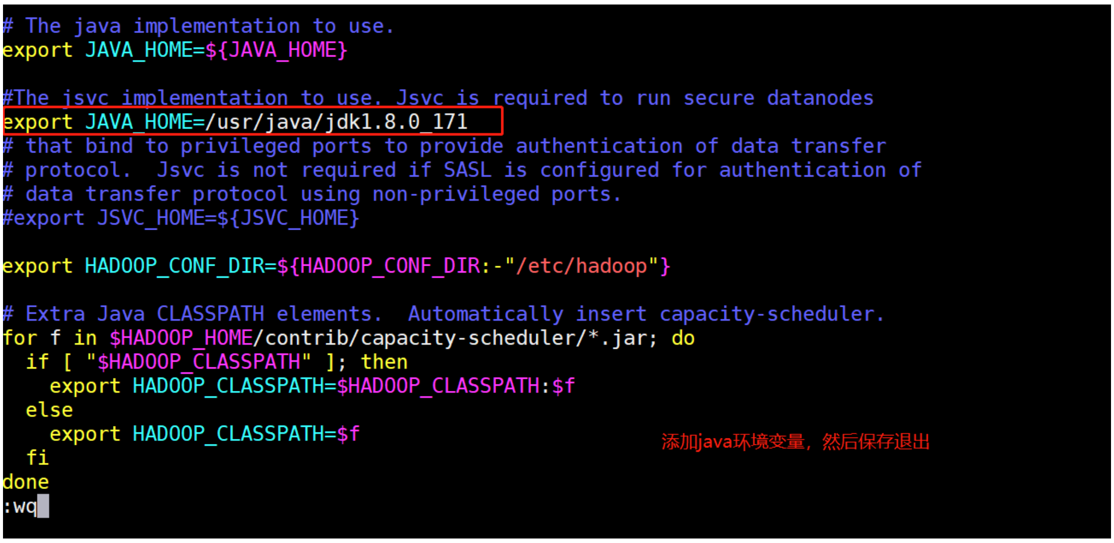


```
[root@master hadoop-2.7.3]# cd etc
[root@master etc]# ls
hadoop
[root@master etc]# cd hadoop/
[root@master hadoop]# ls
capacity-scheduler.xml  hadoop-env.sh                httpfs-env.sh              kms-env.sh             mapred-env.sh                ssl-server.xml.example
configuration.xsl       hadoop-metrics2.properties   httpfs-log4j.properties    kms-log4j.properties   mapred-queues.xml.template   yarn-env.cmd
container-executor.cfg  hadoop-metrics.properties    httpfs-signature.secret    kms-site.xml           mapred-site.xml.template     yarn-env.sh
core-site.xml           hadoop-policy.xml            httpfs-site.xml            log4j.properties       slaves                       yarn-site.xml
hadoop-env.cmd          hdfs-site.xml                kms-acls.xml               mapred-env.cmd         ssl-client.xml.example
[root@master hadoop]# 
```

进入 hadoop 2.7.3 --> etc----> hadoop

使用vim命令编辑 hadoop-env.sh文件

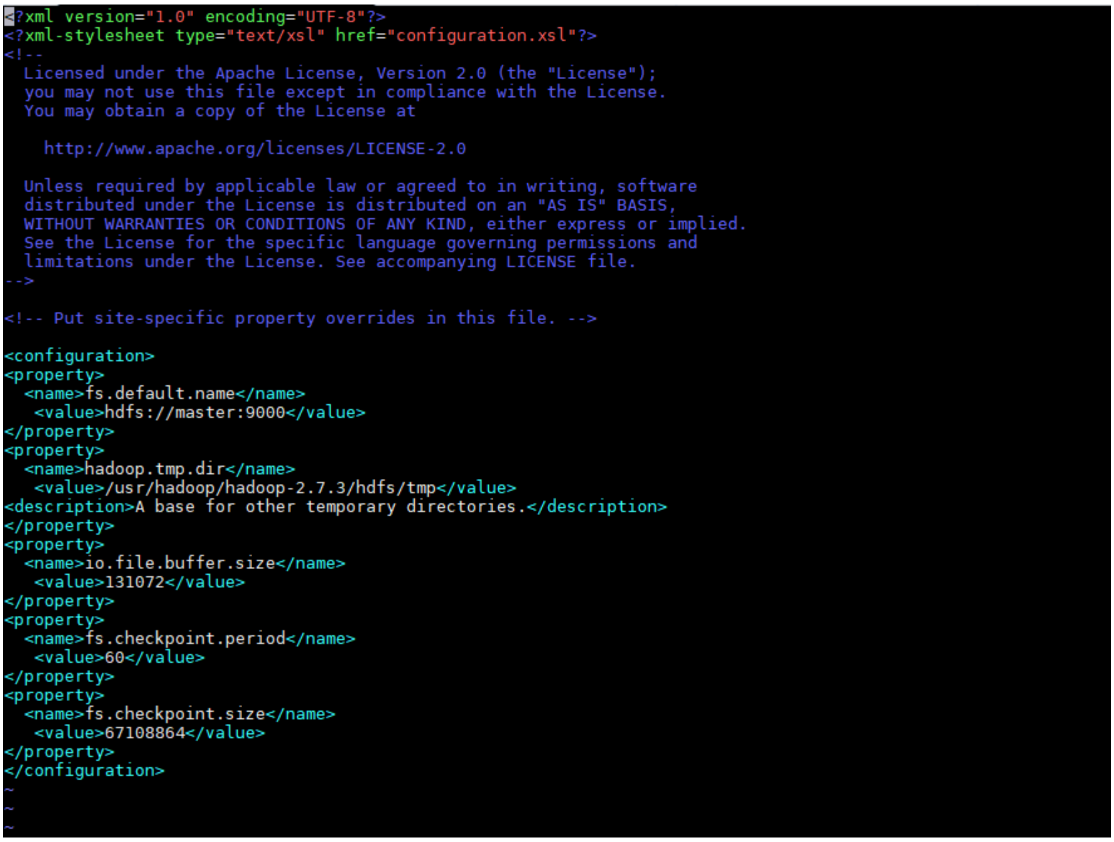
输入内容：export JAVA_HOME=/usr/java/jdk1.8.0_171



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

#The jsvc implementation to use. Jsvc is required to run secure datanodes
export JAVA_HOME=/usr/java/jdk1.8.0_171
# that bind to privileged ports to provide authentication of data transfer
# protocol.  Jsvc is not required if SASL is configured for authentication of
# data transfer protocol using non-privileged ports.
#export JSVC_HOME=${JSVC_HOME}

export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/etc/hadoop"}

# Extra Java CLASSPATH elements.  Automatically insert capacity-scheduler.
for f in $HADOOP_HOME/contrib/capacity-scheduler/*.jar; do
  if [ "$HADOOP_CLASSPATH" ]; then
    export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
  else
    export HADOOP_CLASSPATH=$f
  fi
done
:wq
```

添加java环境变量，然后保存退出

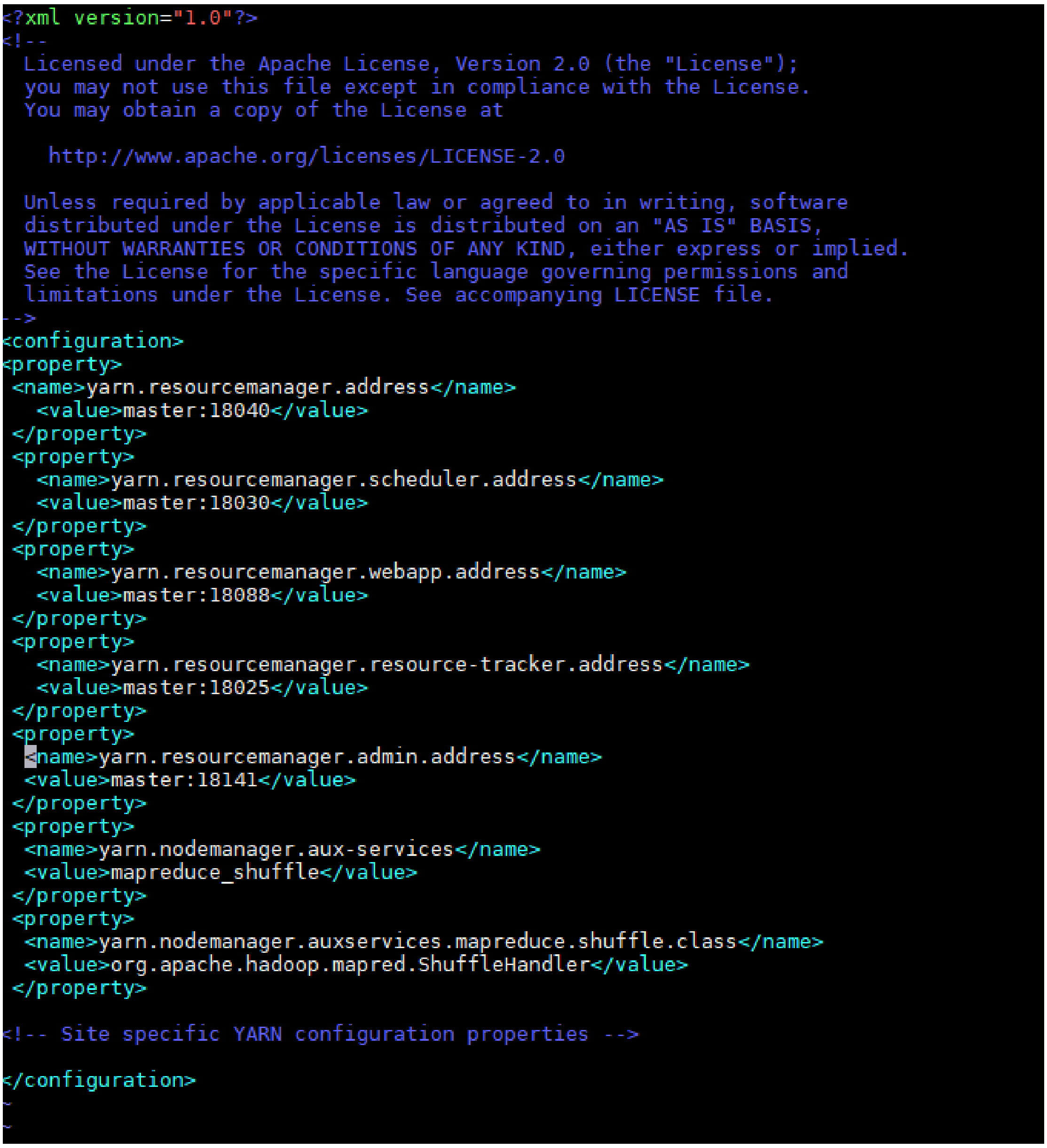
（3）core-site.xml



```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
  <name>fs.default.name</name>
   <value>hdfs://master:9000</value>
</property>
<property>
  <name>hadoop.tmp.dir</name>
   <value>/usr/hadoop/hadoop-2.7.3/hdfs/tmp</value>
<description>A base for other temporary directories.</description>
</property>
<property>
  <name>io.file.buffer.size</name>
   <value>131072</value>
</property>
<property>
  <name>fs.checkpoint.period</name>
   <value>60</value>
</property>
<property>
  <name>fs.checkpoint.size</name>
   <value>67108864</value>
</property>
</configuration>
~
~
~
```

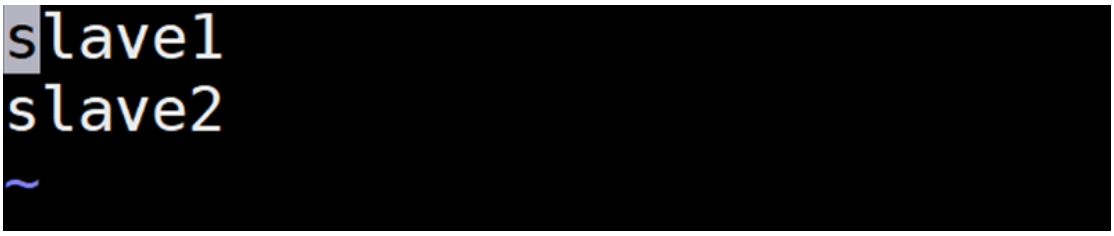
（4）yarn-site.xml

```
<configuration>
 <property>
 <name>yarn.resourcemanager.address</name>
   <value>master:18040</value>
 </property>
 <property>
   <name>yarn.resourcemanager.scheduler.address</name>
   <value>master:18030</value>
 </property>
 <property>
   <name>yarn.resourcemanager.webapp.address</name>
   <value>master:18088</value>
 </property>
 <property>
   <name>yarn.resourcemanager.resource-tracker.address</name>
   <value>master:18025</value>
 </property>
 <property>
  <name>yarn.resourcemanager.admin.address</name>
  <value>master:18141</value>
  </property>

  <property>

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

    <value>mapreduce_shuffle</value>

  </property>

  <property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>

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

  </property>



<!-- Site specific YARN configuration properties -->



</configuration>
```



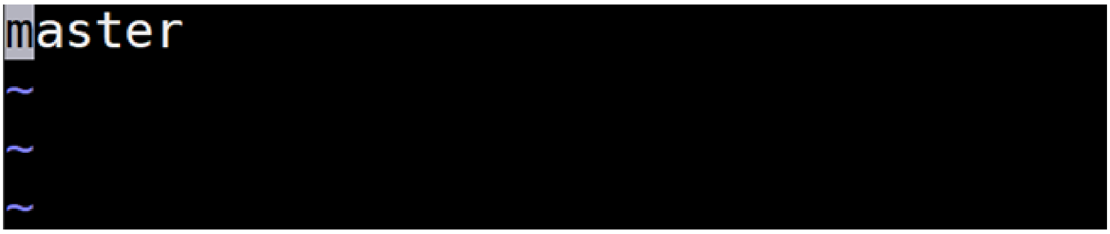
```
<?xml version="1.0"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<configuration>
<property>
 <name>yarn.resourcemanager.address</name>
   <value>master:18040</value>
 </property>
 <property>
   <name>yarn.resourcemanager.scheduler.address</name>
   <value>master:18030</value>
 </property>
 <property>
   <name>yarn.resourcemanager.webapp.address</name>
   <value>master:18088</value>
 </property>
 <property>
   <name>yarn.resourcemanager.resource-tracker.address</name>
   <value>master:18025</value>
 </property>
 <property>
  <name>yarn.resourcemanager.admin.address</name>
  <value>master:18141</value>
 </property>
 <property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
 </property>
 <property>
  <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
 </property>

<!-- Site specific YARN configuration properties -->

</configuration>
~
~
```

（5）编写 slavs 文件



```
slave1
slave2
~
```

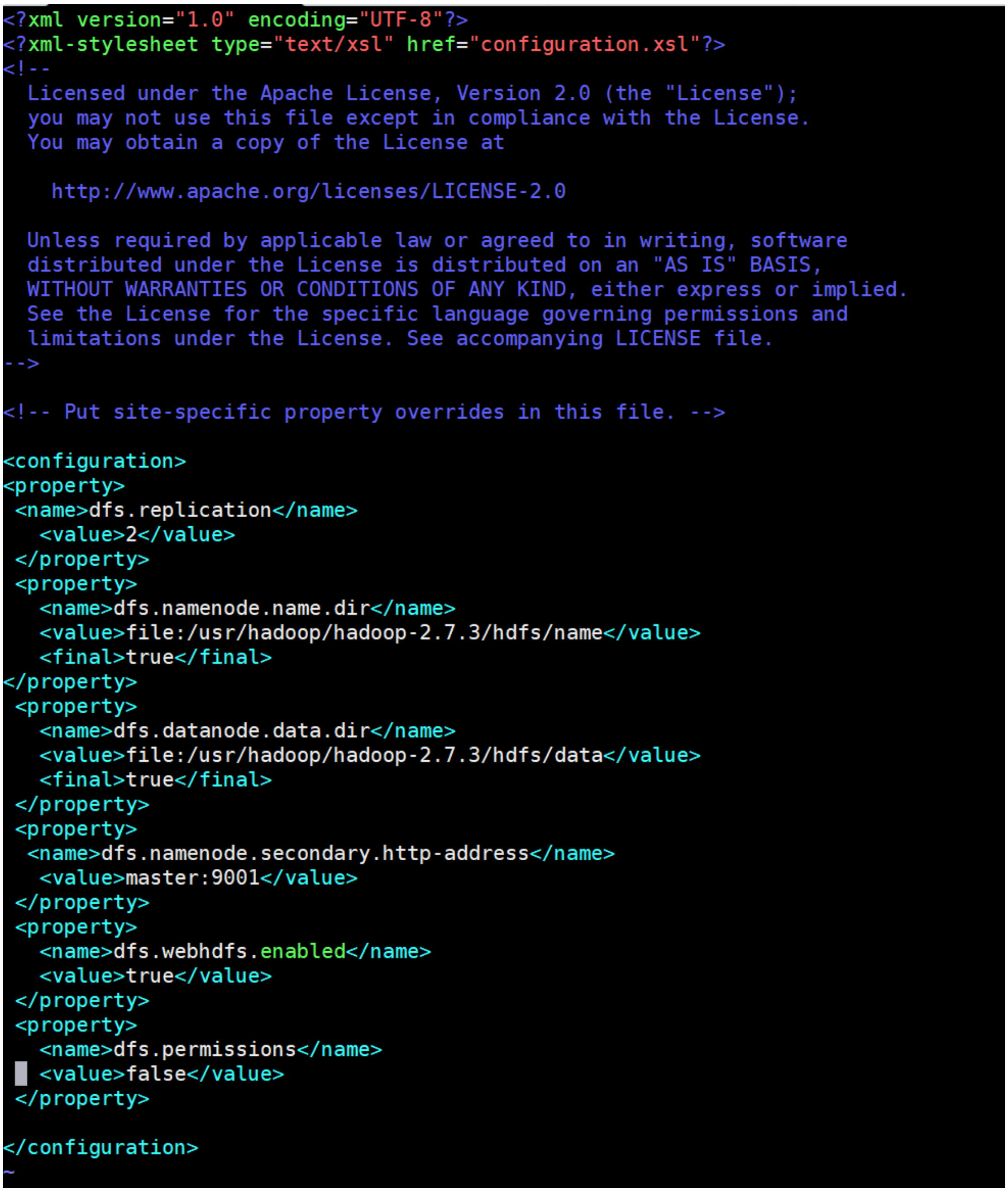
（6）master 文件



```
master
~
~
~
```

（7）hdfs-site.xml

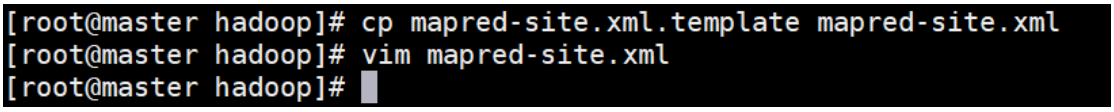
```
<configuration>
  <property>
 <name>dfs.replication</name>
   <value>2</value>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/usr/hadoop/hadoop-2.7.3/hdfs/name</value>
   <final>true</final>
</property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>file:/usr/hadoop/hadoop-2.7.3/hdfs/data</value>
   <final>true</final>
 </property>
 <property>
  <name>dfs.namenode.secondary.http-address</name>
   <value>master:9001</value>
 </property>
 <property>
   <name>dfs.webhdfs.enabled</name>
   <value>true</value>
 </property>
 <property>
   <name>dfs.permissions</name>
   <value>false</value>
 </property>
</configuration>
```



```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
 <name>dfs.replication</name>
   <value>2</value>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/usr/hadoop/hadoop-2.7.3/hdfs/name</value>
   <final>true</final>
</property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>file:/usr/hadoop/hadoop-2.7.3/hdfs/data</value>
   <final>true</final>
 </property>
 <property>
  <name>dfs.namenode.secondary.http-address</name>
   <value>master:9001</value>
 </property>
 <property>
   <name>dfs.webhdfs.enabled</name>
   <value>true</value>
 </property>
 <property>
   <name>dfs.permissions</name>
   <value>false</value>
 </property>

</configuration>
~
```

（8） mapred-site.xml

首先将模板文件复制为 xml 文件，对其进行编辑：



```
[root@master hadoop]# cp mapred-site.xml.template mapred-site.xml
[root@master hadoop]# vim mapred-site.xml
[root@master hadoop]# 
```

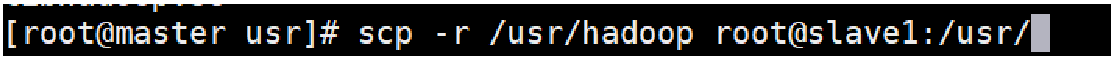
```
    <property>

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

        <value>yarn</value>
```

```
</property>
```

（9）分发 hadoop：

scp -r /usr/hadoop root@slave1:/usr/

scp -r /usr/hadoop root@slave2:/usr/

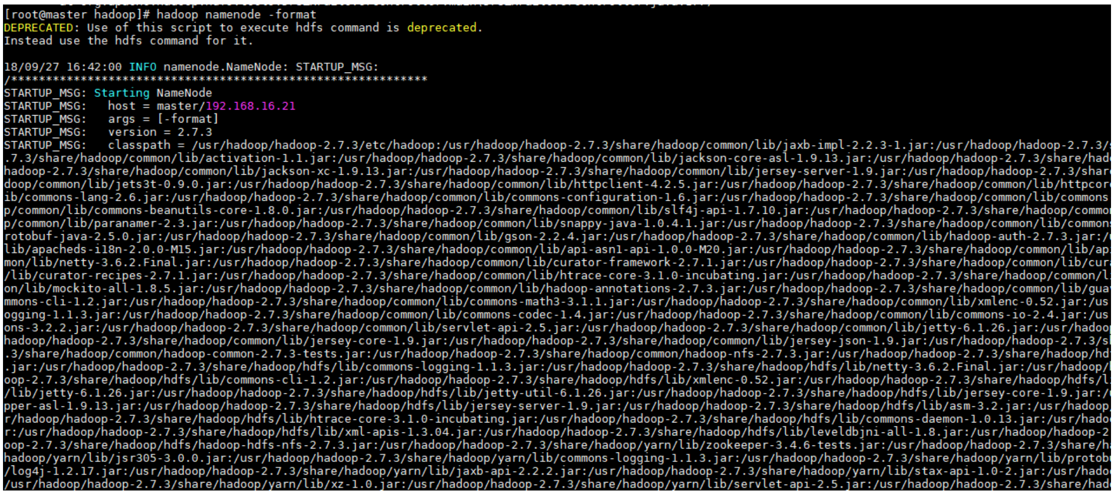


```
[root@master usr]# scp -r /usr/hadoop root@slave1:/usr/
```

注意：slave 节点上还需要配置环境变量，参考 hadoop 中第一个步骤。

（10）master 中格式化 hadoop

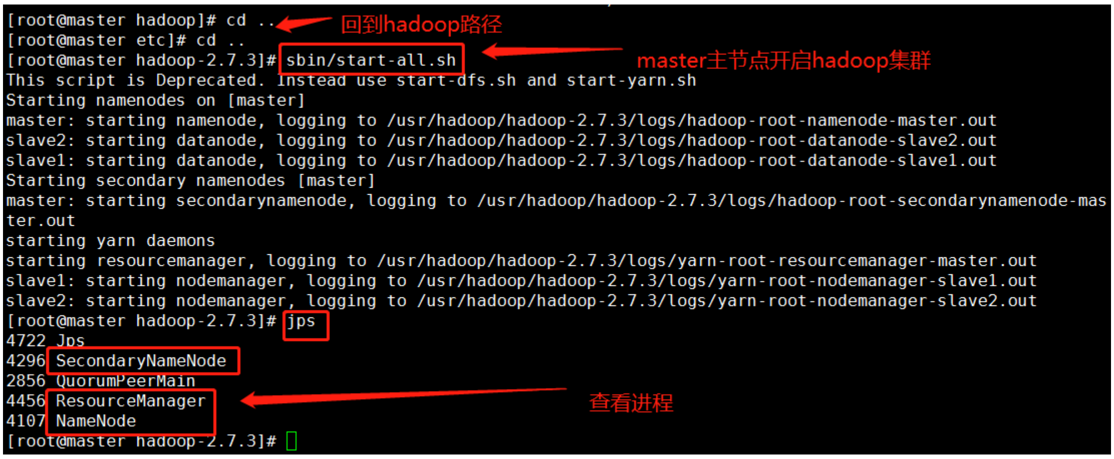
hadoo namenode -format



```
[root@master hadoop]# hadoop namenode -format
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

18/09/27 16:42:00 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = master/192.168.16.21
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.7.3
STARTUP_MSG:   classpath = /usr/hadoop/hadoop-2.7.3/etc/hadoop:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/jaxb-impl-2.2.3-1.jar:/usr/hadoop/hadoop-2.7.3/
.7.3/share/hadoop/common/lib/activation-1.1.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/jackson-core-asl-1.9.13.jar:/usr/hadoop/hadoop-2.7.3/share/had
hadoop-2.7.3/share/hadoop/common/lib/jackson-xc-1.9.13.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/jersey-server-1.9.jar:/usr/hadoop/hadoop-2.7.3/shar
doop/common/lib/jets3t-0.9.0.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/httpclient-4.2.5.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/httpcor
ib/commons-lang-2.6.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/commons-configuration-1.6.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/commons
p/common/lib/commons-beanutils-core-1.8.0.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/slf4j-api-1.7.10.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/commo
p/common/lib/paranamer-2.3.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/snappy-java-1.0.4.1.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/common
rotobuf-java-2.5.0.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/gson-2.2.4.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/hadoop-auth-2.7.3.jar:/
lib/apacheds-i18n-2.0.0-M15.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/api-asn1-api-1.0.0-M20.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/ap
mon/lib/netty-3.6.2.Final.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/curator-framework-2.7.1.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/cur
/lib/curator-recipes-2.7.1.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/htrace-core-3.1.0-incubating.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/l
on/lib/mockito-all-1.8.5.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/hadoop-annotations-2.7.3.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/gua
mmons-cli-1.2.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/commons-math3-3.1.1.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/xmlenc-0.52.jar:/us
ogging-1.1.3.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/commons-codec-1.4.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/commons-io-2.4.jar:/us
ons-3.2.2.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/servlet-api-2.5.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/jetty-6.1.26.jar:/usr/hadoo
hadoop/hadoop-2.7.3/share/hadoop/common/lib/jersey-core-1.9.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/jersey-json-1.9.jar:/usr/hadoop/hadoop-2.7.3/s
.3/share/hadoop/common/hadoop-common-2.7.3-tests.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/hadoop-nfs-2.7.3.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/hd
.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/hdfs/lib/commons-logging-1.1.3.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/hdfs/lib/netty-3.6.2.Final.jar:/usr/hadoop/
oop-2.7.3/share/hadoop/hdfs/lib/commons-cli-1.2.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/hdfs/lib/xmlenc-0.52.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/hdfs/l
/lib/jetty-6.1.26.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/hdfs/lib/jetty-util-6.1.26.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/hdfs/lib/jersey-core-1.9.jar:/
pper-asl-1.9.13.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/hdfs/lib/jersey-server-1.9.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/hdfs/lib/asm-3.2.jar:/usr/hadoop
r/hadoop/hadoop-2.7.3/share/hadoop/hdfs/lib/htrace-core-3.1.0-incubating.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/hdfs/lib/commons-daemon-1.0.13.jar:/usr/hado
r:/usr/hadoop/hadoop-2.7.3/share/hadoop/hdfs/lib/xml-apis-1.3.04.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/hdfs/lib/leveldbjni-all-1.8.jar:/usr/hadoop/hadoop-2
oop-2.7.3/share/hadoop/hdfs/hadoop-hdfs-nfs-2.7.3.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/yarn/lib/zookeeper-3.4.6-tests.jar:/usr/hadoop/hadoop-2.7.3/share/h
hadoop/yarn/lib/jsr305-3.0.0.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/yarn/lib/commons-logging-1.1.3.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/yarn/lib/protob
/log4j-1.2.17.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/yarn/lib/jaxb-api-2.2.2.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/yarn/lib/stax-api-1.0-2.jar:/usr/hado
/usr/hadoop/hadoop-2.7.3/share/hadoop/yarn/lib/xz-1.0.jar:/usr/hadoop/hadoop-2.7.3/share/hadoop/yarn/lib/servlet-api-2.5.jar:/usr/hadoop/hadoop-2.7.3/share/had
```

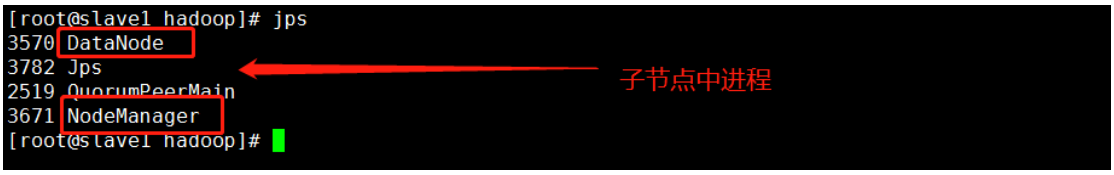
（11）各节点进行如下：

master：



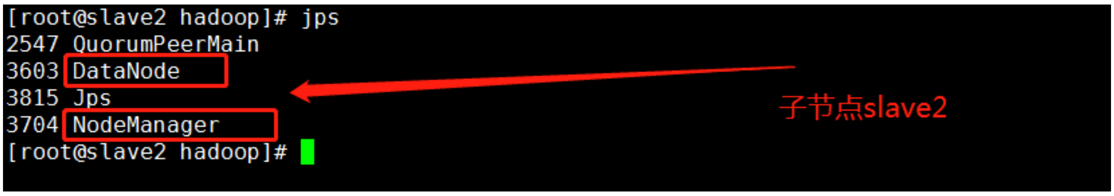
```
[root@master hadoop]# cd ..
[root@master etc]# cd ..
[root@master hadoop-2.7.3]# sbin/start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [master]
master: starting namenode, logging to /usr/hadoop/hadoop-2.7.3/logs/hadoop-root-namenode-master.out
slave2: starting datanode, logging to /usr/hadoop/hadoop-2.7.3/logs/hadoop-root-datanode-slave2.out
slave1: starting datanode, logging to /usr/hadoop/hadoop-2.7.3/logs/hadoop-root-datanode-slave1.out
Starting secondary namenodes [master]
master: starting secondarynamenode, logging to /usr/hadoop/hadoop-2.7.3/logs/hadoop-root-secondarynamenode-mas
ter.out
starting yarn daemons
starting resourcemanager, logging to /usr/hadoop/hadoop-2.7.3/logs/yarn-root-resourcemanager-master.out
slave1: starting nodemanager, logging to /usr/hadoop/hadoop-2.7.3/logs/yarn-root-nodemanager-slave1.out
slave2: starting nodemanager, logging to /usr/hadoop/hadoop-2.7.3/logs/yarn-root-nodemanager-slave2.out
[root@master hadoop-2.7.3]# jps
4722 Jps
4296 SecondaryNameNode
2856 QuorumPeerMain
4456 ResourceManager
4107 NameNode
[root@master hadoop-2.7.3]# 
```

slave1：



```
[root@slave1 hadoop]# jps
3570 DataNode
3782 Jps
2519 QuorumPeerMain
3671 NodeManager
[root@slave1 hadoop]#
```

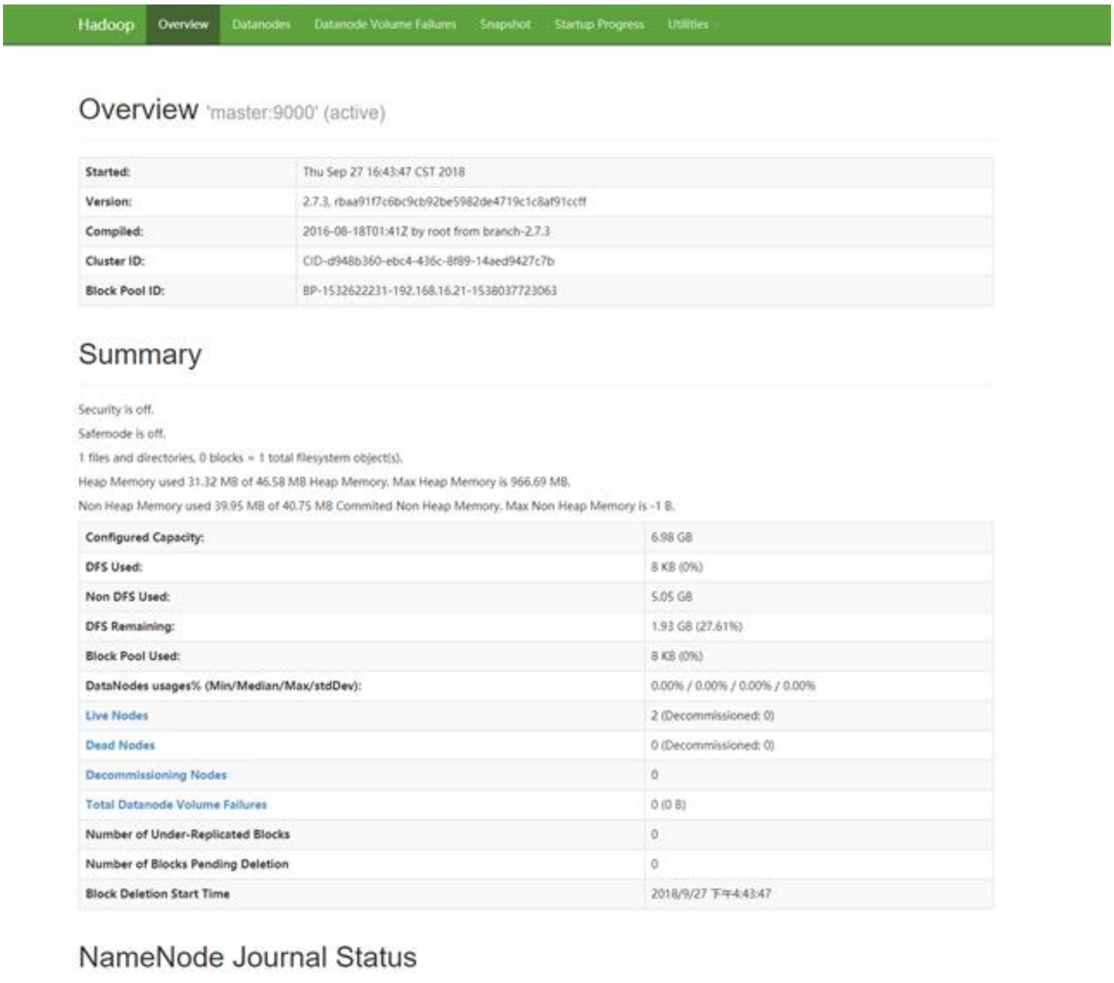
slave2：



```
[root@slave2 hadoop]# jps
2547 QuorumPeerMain
3603 DataNode
3815 Jps
3704 NodeManager
[root@slave2 hadoop]#
```

访问主节点 master：50070（50070 是 hdfs 的 web 管理页面）

注意，如果发现集群已启动，但是访问不了，可能是防火墙没有关闭。



Overview 'master:9000' (active)

Started:	Thu Sep 27 16:43:47 CST 2018
Version:	2.7.3, rbaa91f7c6bc9cb92be5982de4719c1c8af91ccff
Compiled:	2016-08-18T01:41Z by root from branch-2.7.3
Cluster ID:	CID-d948b360-ebc4-436c-8f89-14aed9427c7b
Block Pool ID:	BP-1532622231-192.168.16.21-1538037723063

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 31.32 MB of 46.58 MB Heap Memory. Max Heap Memory is 966.69 MB.

Non Heap Memory used 39.95 MB of 40.75 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

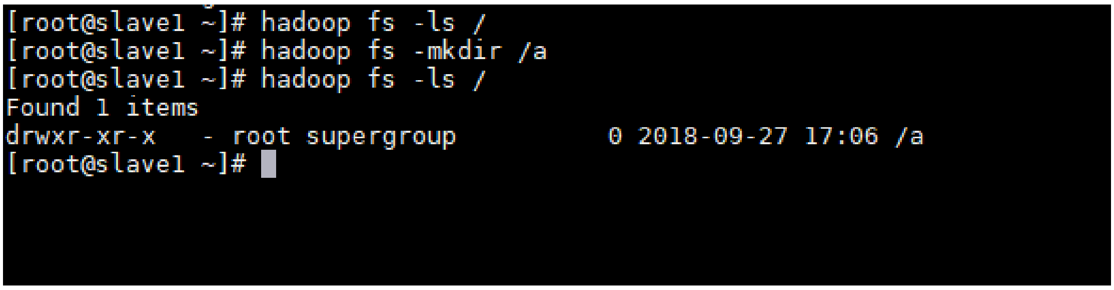
Configured Capacity:	6.98 GB
DFS Used:	8 KB (0%)
Non DFS Used:	5.05 GB
DFS Remaining:	1.93 GB (27.61%)
Block Pool Used:	8 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	2 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	2018/9/27 下午4:43:47

NameNode Journal Status

（12）查看 hdfs

Hadoop fs -ls /　　　　(最开始创建的是一个空的文件系统所以什么也没有)

Hadoop fs -mkdir /a　　(在 hdfs 上传到 a 文件夹)

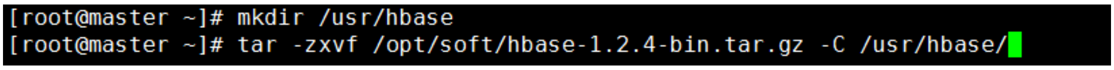


```
[root@slave1 ~]# hadoop fs -ls /
[root@slave1 ~]# hadoop fs -mkdir /a
[root@slave1 ~]# hadoop fs -ls /
Found 1 items
drwxr-xr-x   - root supergroup          0 2018-09-27 17:06 /a
[root@slave1 ~]# 
```

3.4hbase 安装

（1）同样先建立工作路径/usr/hbase,将/opt/soft 下的 hbase 解压到工作路径中。

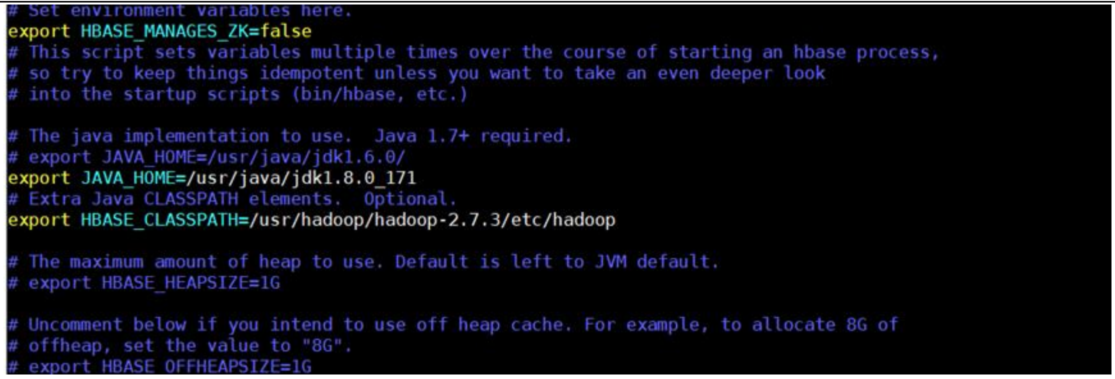
解压：tar -zxvf /opt/soft/hbase-1.2.4-bin.tar.gz -C /usr/hbase



```
[root@master ~]# mkdir /usr/hbase
[root@master ~]# tar -zxvf /opt/soft/hbase-1.2.4-bin.tar.gz -C /usr/hbase/
```

（2）修改配置文件：conf/hbase-env.sh

```
export HBASE_MANAGES_ZK=false

export JAVA_HOME=/usr/java/jdk1.8.0_171

export HBASE_CLASSPATH=/usr/hadoop/hadoop-2.7.3/etc/Hadoop
```



```
# Set environment variables here.
export HBASE_MANAGES_ZK=false
# This script sets variables multiple times over the course of starting an hbase process,
# so try to keep things idempotent unless you want to take an even deeper look
# into the startup scripts (bin/hbase, etc.)

# The java implementation to use.  Java 1.7+ required.
# export JAVA_HOME=/usr/java/jdk1.6.0/
export JAVA_HOME=/usr/java/jdk1.8.0_171
# Extra Java CLASSPATH elements.  Optional.
export HBASE_CLASSPATH=/usr/hadoop/hadoop-2.7.3/etc/hadoop

# The maximum amount of heap to use. Default is left to JVM default.
# export HBASE_HEAPSIZE=1G

# Uncomment below if you intend to use off heap cache. For example, to allocate 8G of
# offheap, set the value to "8G".
# export HBASE_OFFHEAPSIZE=1G
```

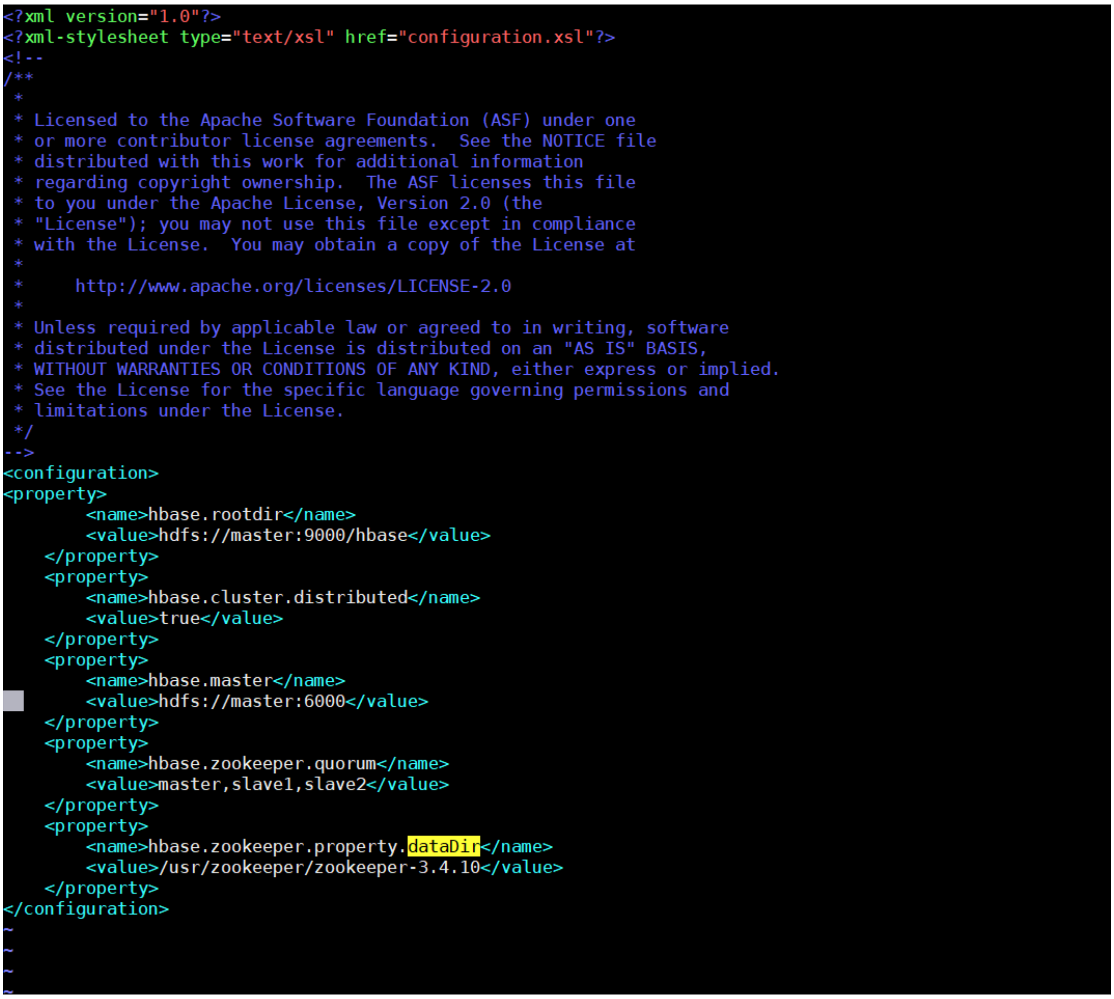
解释：一个分布式运行的 Hbase 依赖一个 zookeeper 集群。所有的节点和客户端都必须能够访问 zookeeper。默认的情况下 Hbase 会管理一个 zookeep 集群，即 Hbase 默认自带一个 zookeep 集群。这个集群会随着 Hbase 的启动而启动。而在实际的商业项目中通常自己管理一个 zookeeper 集群更便于优化配置提

高集群工作效率，但需要配置 Hbase。需要修改 conf/hbase-env.sh 里面的 HBASE_MANAGES_ZK 来切换。这个值默认是 true 的，作用是让 Hbase 启动的时候同时也启动 zookeeper.在本实验中，我们采用独立运行 zookeeper 集群的方式，故将其属性值改为 false。

（3）配置 conf/hbase-site.xml

```
<configuration>
  <property>
        <name>hbase.rootdir</name>
        <value>hdfs://master:9000/hbase</value>
    </property>
    <property>
        <name>hbase.cluster.distributed</name>
        <value>true</value>
    </property>
    <property>
        <name>hbase.master</name>
        <value>hdfs://master:6000</value>
    </property>
    <property>
        <name>hbase.zookeeper.quorum</name>
        <value>master,slave1,slave2</value>
    </property>
```

```
        <property>

            <name>hbase.zookeeper.property.dataDir</name>

            <value>/usr/zookeeper/zookeeper-3.4.10</value>

        </property>

    </configuration>
```



```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
/**
 *
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
-->
<configuration>
<property>
        <name>hbase.rootdir</name>
        <value>hdfs://master:9000/hbase</value>
    </property>
    <property>
        <name>hbase.cluster.distributed</name>
        <value>true</value>
    </property>
    <property>
        <name>hbase.master</name>
        <value>hdfs://master:6000</value>
    </property>
    <property>
        <name>hbase.zookeeper.quorum</name>
        <value>master,slave1,slave2</value>
    </property>
    <property>
        <name>hbase.zookeeper.property.dataDir</name>
        <value>/usr/zookeeper/zookeeper-3.4.10</value>
    </property>
</configuration>
~
~
~
~
```

解释：要想运行完全分布式模式，加一个属性 hbase.cluster.distributed 设置为 true 然后把 hbase.rootdir 设置为 HDFS 的 NameNode 的位置

hbase.rootdir：这个目录是 region server 的共享目录，用来持久化 Hbase。URL 需要是'完全正确'的，还要包含文件系统的 scheme

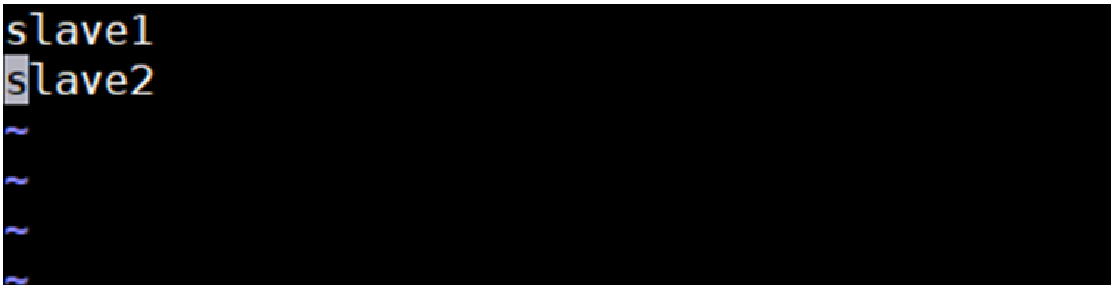
hbase.cluster.distributed ：Hbase 的运行模式。false 是单机模式，true 是分

布式模式。若为 false,Hbase 和 Zookeeper 会运行在同一个 JVM 里面。在 hbase-site.xml 配置 zookeeper：当 Hbase 管理 zookeeper 的时候，你可以通过修改 zoo.cfg 来配置 zookeeper，对于 zookeepr 的配置，你至少要在 hbase-site.xml 中列出 zookeepr 的 ensemble servers，具体的字段是 hbase.zookeeper.quorum. 在这里列出 Zookeeper 集群的地址列表，用逗号分割。

hbase.zookeeper.property.clientPort：ZooKeeper 的 zoo.conf 中的配置,客户端连接的端口。

hbase.zookeeper.property.dataDir：ZooKeeper 的 zoo.conf 中的配置。对于独立的 Zookeeper，要指明 Zookeeper 的 host 和端口。需要在 hbase-site.xml 中设置。

（4）配置 conf/regionservers



```
slave1
slave2
~
~
~
~
```

在这里列出了希望运行的全部 HRegionServer，一行写一个 host。列在这里的 server 会随着集群的启动而启动，集群的停止而停止。

（5）hadoop 配置文件拷入 hbase 的 conf 目录下：（当前位置为 hbased 的 conf 配置文件夹）

```
cp /usr/hadoop/hadoop-2.7.3/etc/hadoop/hdfs-site.xml .

cp /usr/hadoop/hadoop-2.7.3/etc/hadoop/core-site.xml .
```

（6）分发 hbase

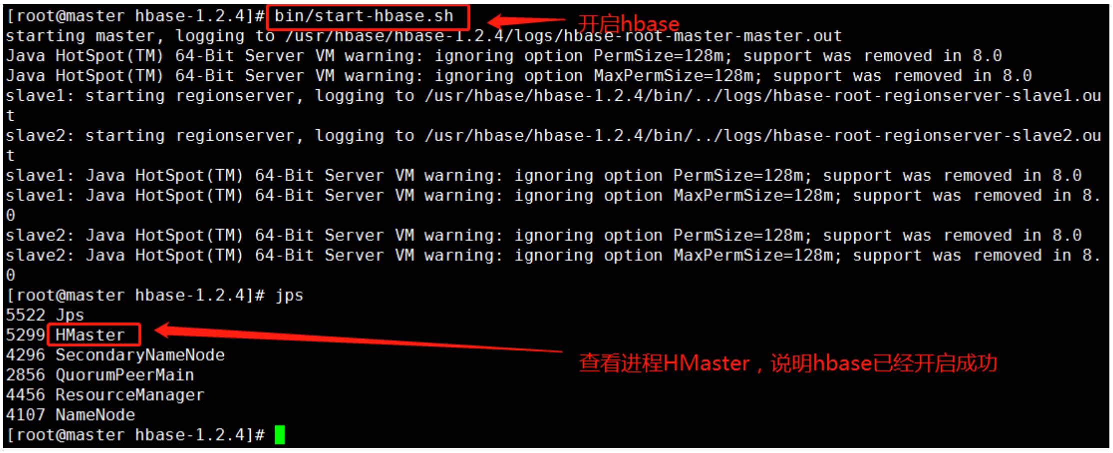
```
scp -r /usr/hbase root@slave1:/usr/

scp -r /usr/hbase root@slave2:/usr/
```

（7）配置环境变量

```
vi /etc/profile

配置环境变量 Hbase

# set hbase environment

export HBASE_HOME=/usr/hbase/hbase-1.2.4

export PATH=$PATH:$HBASE_HOME/bin

生效环境变量：source /etc/profile
```

（8）运行和测试

```
在 master 上执行(保证 hadoop 和 zookeeper 已开启)：

bin/start-hbase.sh
```

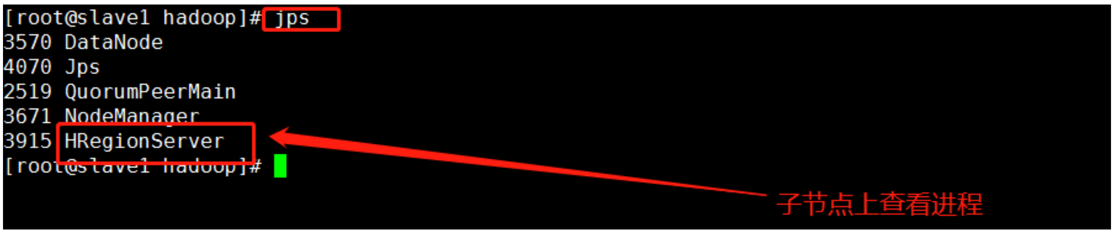


```
[root@master hbase-1.2.4]# bin/start-hbase.sh
starting master, logging to /usr/hbase/hbase-1.2.4/logs/hbase-root-master-master.out
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0
slave1: starting regionserver, logging to /usr/hbase/hbase-1.2.4/bin/../logs/hbase-root-regionserver-slave1.ou
t
slave2: starting regionserver, logging to /usr/hbase/hbase-1.2.4/bin/../logs/hbase-root-regionserver-slave2.ou
t
slave1: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0
slave1: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.
0
slave2: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0
slave2: Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.
0
[root@master hbase-1.2.4]# jps
5522 Jps
5299 HMaster
4296 SecondaryNameNode
2856 QuorumPeerMain
4456 ResourceManager
4107 NameNode
[root@master hbase-1.2.4]#
```

开启hbase

查看进程HMaster，说明hbase已经开启成功

子节点上查看进程：

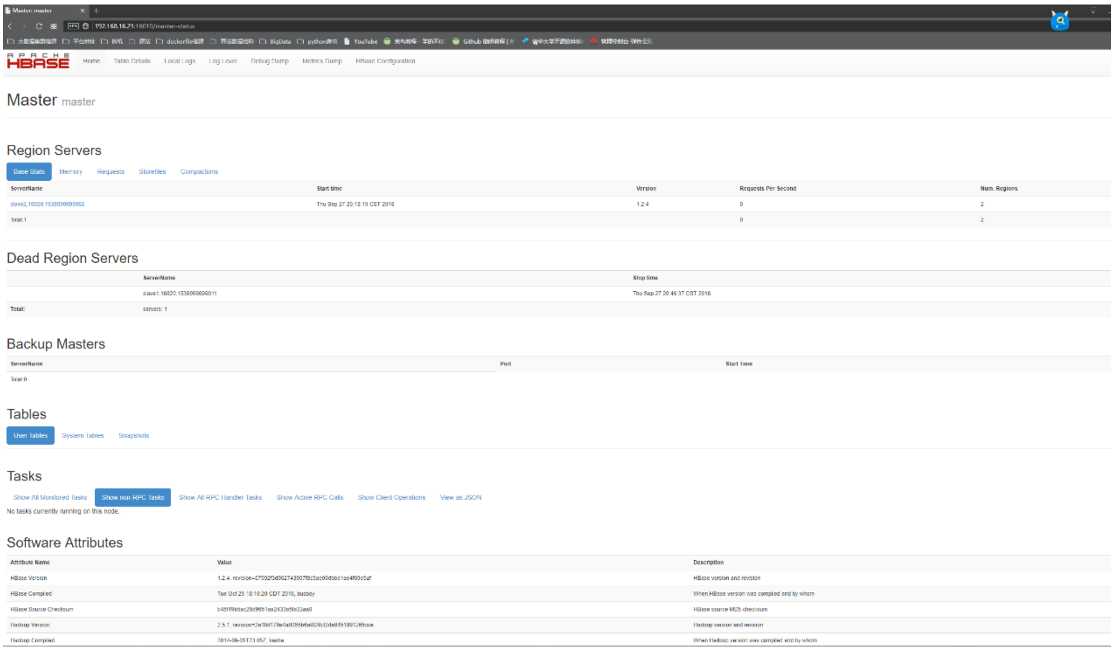


```
[root@slave1 hadoop]# jps
3570 DataNode
4070 Jps
2519 QuorumPeerMain
3671 NodeManager
3915 HRegionServer
[root@slave1 hadoop]#
```

子节点上查看进程

（9）访问 master 的 hbase web 界面

http://master IP:16010/master-status



Master master

Region Servers

ServerName	Start time	Version	Requests Per Second	Num. Regions
slave2,16020,1538050695962	Thu Sep 27 20:18:15 CST 2018	1.2.4	0	2
Total:1			0	2

Dead Region Servers

	ServerName	Stop time
	slave1,16020,1538050696011	Thu Sep 27 20:46:37 CST 2018
Total:	servers: 1	

Backup Masters

ServerName	Port	Start Time
Total:0		

Tables

Tasks

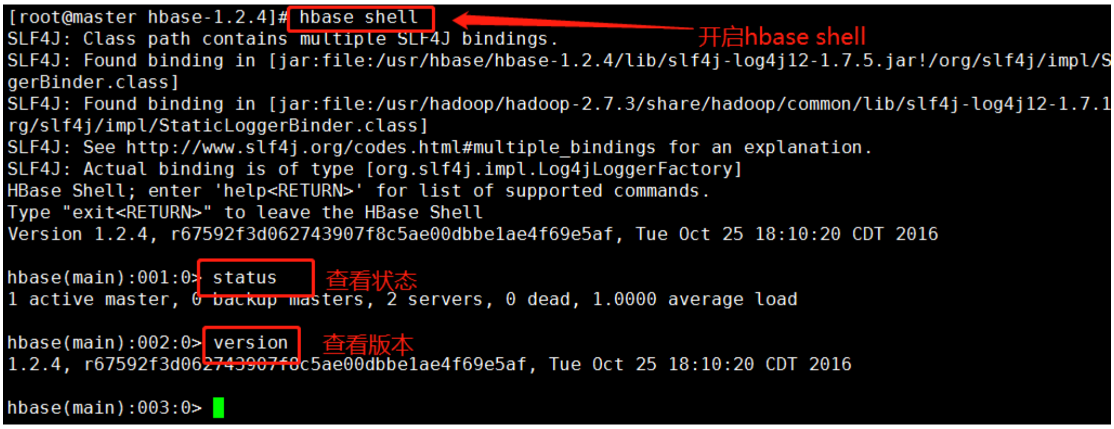
No tasks currently running on this node.

Software Attributes

Attribute Name	Value	Description
HBase Version	1.2.4, revision=67592f3d062743907f8c5ae00dbbe1ae4f69e5af	HBase version and revision
HBase Compiled	Tue Oct 25 18:10:20 CDT 2016, busbey	When HBase version was compiled and by whom
HBase Source Checksum	b45f19b5ac28d9651aa2433a5fa33aa0	HBase source MD5 checksum
Hadoop Version	2.5.1, revision=2e18d179e4a8065b6a9f29cf2de9451891265cce	Hadoop version and revision
Hadoop Compiled	2014-09-05T23:05Z, kasha	When Hadoop version was compiled and by whom

（10）进 hbase 交互界面，查看状态与版本

hbase shell



```
[root@master hbase-1.2.4]# hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hbase/hbase-1.2.4/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/S
gerBinder.class]
SLF4J: Found binding in [jar:file:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.1
rg/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.4, r67592f3d062743907f8c5ae00dbbe1ae4f69e5af, Tue Oct 25 18:10:20 CDT 2016

hbase(main):001:0> status
1 active master, 0 backup masters, 2 servers, 0 dead, 1.0000 average load

hbase(main):002:0> version
1.2.4, r67592f3d062743907f8c5ae00dbbe1ae4f69e5af, Tue Oct 25 18:10:20 CDT 2016

hbase(main):003:0>
```

开启hbase shell

查看状态

查看版本

2. 构建数据仓库

master 作为 client 客户端

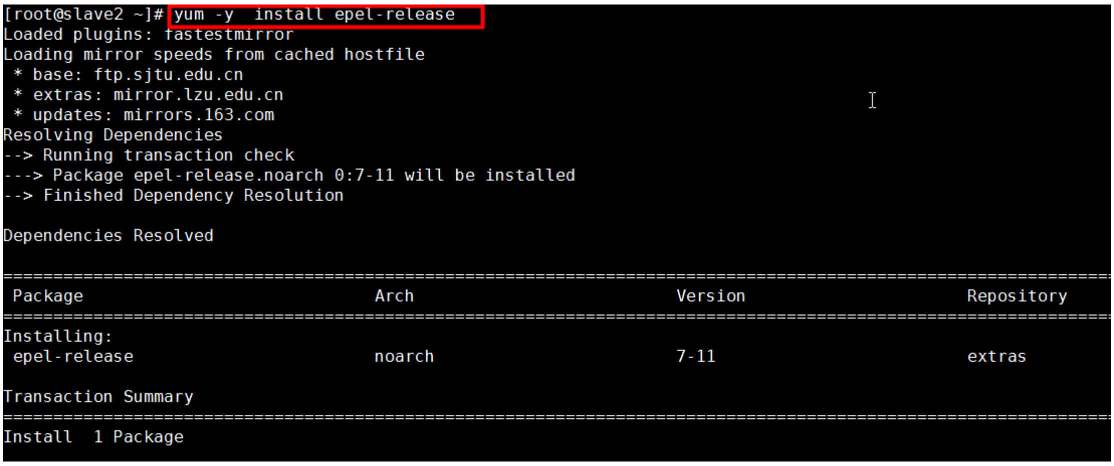
slave1 作为 hive server 服务器端

slave2 安装 mysql server

4.1slave2 上安装 mysql server

（1）安装 EPEL 源（实验中已给出本地源，下载源仅做参考）

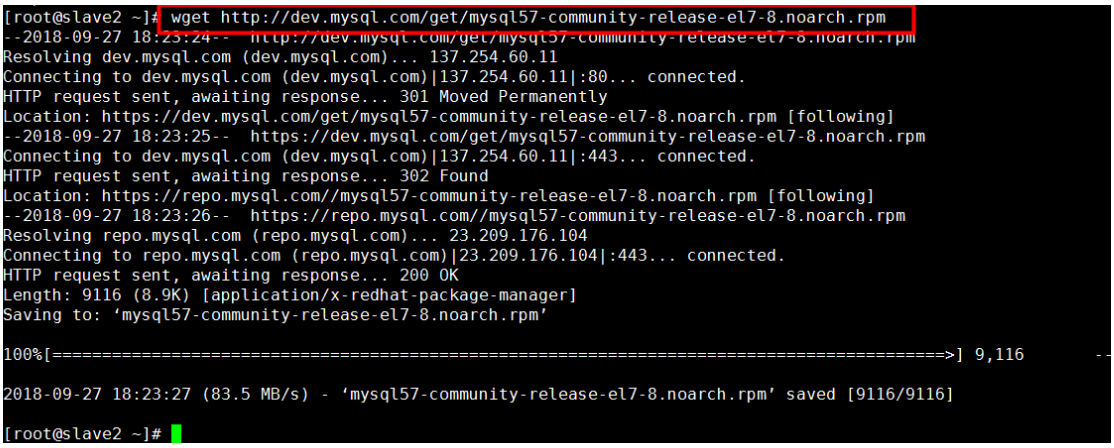
yum -y install epel-release



```
[root@slave2 ~]# yum -y  install epel-release
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: ftp.sjtu.edu.cn
 * extras: mirror.lzu.edu.cn
 * updates: mirrors.163.com
Resolving Dependencies
--> Running transaction check
---> Package epel-release.noarch 0:7-11 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

==========================================================================================================
 Package                              Arch                          Version                    Repository
==========================================================================================================
Installing:
 epel-release                         noarch                        7-11                       extras

Transaction Summary
==========================================================================================================
Install  1 Package
```

（2）安装 MySQL server 包，下载源安装包：

wget http://dev.mysql.com/get/mysql57-community-release-el7-8.noarch.rpm



```
[root@slave2 ~]# wget http://dev.mysql.com/get/mysql57-community-release-el7-8.noarch.rpm
--2018-09-27 18:23:24--  http://dev.mysql.com/get/mysql57-community-release-el7-8.noarch.rpm
Resolving dev.mysql.com (dev.mysql.com)... 137.254.60.11
Connecting to dev.mysql.com (dev.mysql.com)|137.254.60.11|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://dev.mysql.com/get/mysql57-community-release-el7-8.noarch.rpm [following]
--2018-09-27 18:23:25--  https://dev.mysql.com/get/mysql57-community-release-el7-8.noarch.rpm
Connecting to dev.mysql.com (dev.mysql.com)|137.254.60.11|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://repo.mysql.com//mysql57-community-release-el7-8.noarch.rpm [following]
--2018-09-27 18:23:26--  https://repo.mysql.com//mysql57-community-release-el7-8.noarch.rpm
Resolving repo.mysql.com (repo.mysql.com)... 23.209.176.104
Connecting to repo.mysql.com (repo.mysql.com)|23.209.176.104|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9116 (8.9K) [application/x-redhat-package-manager]
Saving to: ‘mysql57-community-release-el7-8.noarch.rpm’

100%[==========================================================================================>] 9,116       --

2018-09-27 18:23:27 (83.5 MB/s) - ‘mysql57-community-release-el7-8.noarch.rpm’ saved [9116/9116]

[root@slave2 ~]#
```

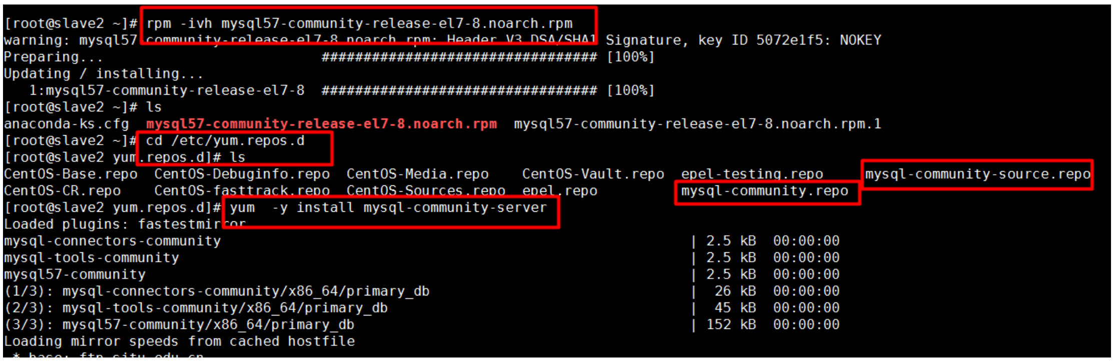
（3）安装源：rpm -ivh mysql57-community-release-el7-8.noarch.rpm

查看是否有包：cd /etc/yum.repos.d

mysql-community.repo

mysql-community-source.repo

安装 MySQL：yum -y install mysql-community-server



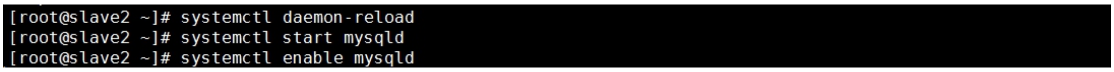
```
[root@slave2 ~]# rpm -ivh mysql57-community-release-el7-8.noarch.rpm
warning: mysql57-community-release-el7-8.noarch.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY
Preparing...                          ################################# [100%]
Updating / installing...
   1:mysql57-community-release-el7-8  ################################# [100%]
[root@slave2 ~]# ls
anaconda-ks.cfg  mysql57-community-release-el7-8.noarch.rpm  mysql57-community-release-el7-8.noarch.rpm.1
[root@slave2 ~]# cd /etc/yum.repos.d
[root@slave2 yum.repos.d]# ls
CentOS-Base.repo  CentOS-Debuginfo.repo  CentOS-Media.repo    CentOS-Vault.repo  epel-testing.repo      mysql-community-source.repo
CentOS-CR.repo    CentOS-fasttrack.repo  CentOS-Sources.repo  epel.repo          mysql-community.repo
[root@slave2 yum.repos.d]# yum  -y install mysql-community-server
Loaded plugins: fastestmirror
mysql-connectors-community                                                  | 2.5 kB  00:00:00
mysql-tools-community                                                       | 2.5 kB  00:00:00
mysql57-community                                                           | 2.5 kB  00:00:00
(1/3): mysql-connectors-community/x86_64/primary_db                         |  26 kB  00:00:00
(2/3): mysql-tools-community/x86_64/primary_db                              |  45 kB  00:00:00
(3/3): mysql57-community/x86_64/primary_db                                  | 152 kB  00:00:00
Loading mirror speeds from cached hostfile
```

（4）启动服务

重载所有修改过的配置文件：systemctl daemon-reload

开启服务：systemctl start mysqld

开机自启：systemctl enable mysqld

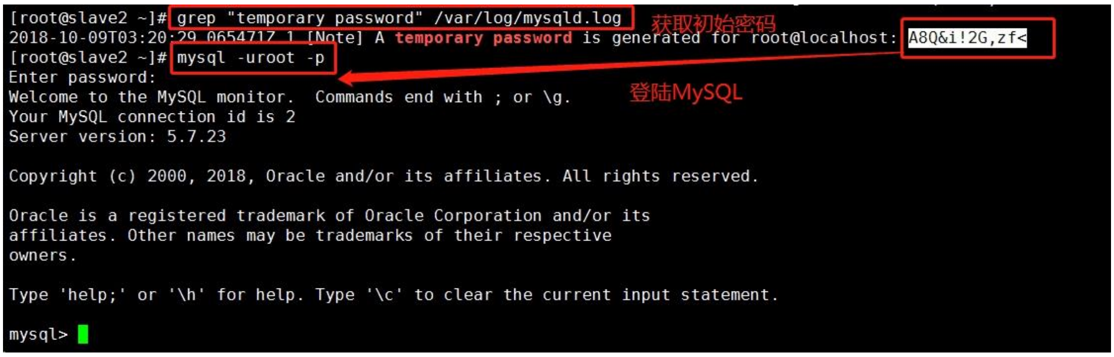


```
[root@slave2 ~]# systemctl daemon-reload
[root@slave2 ~]# systemctl start mysqld
[root@slave2 ~]# systemctl enable mysqld
```

（5）安装完毕后，/var/log/mysqld.log 文件中会自动生成一个随机的密码，我们需要先取得这个随机密码，以用于登录 MySQL 服务端：

获取初密码：grep "password" /var/log/mysqld.log

登陆 MySQL：mysql -uroot -p



```
[root@slave2 ~]# grep "temporary password" /var/log/mysqld.log
2018-10-09T03:20:29.065471Z 1 [Note] A temporary password is generated for root@localhost: A8Q&i!2G,zf<
[root@slave2 ~]# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.23

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```

（6）MySQL 密码安全策略：

设置密码强度为低级：set global validate_password_policy=0;

设置密码长度：set global validate_password_length=4;

修改本地密码：alter user 'root'@'localhost' identified by '123456';

退出：\q

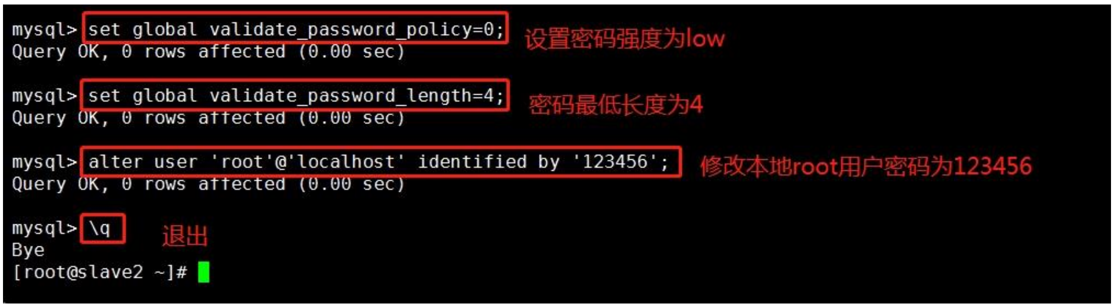
密码强度分级如下：

0 为 low 级别，只检查长度；

1 为 medium 级别（默认），符合长度为 8，且必须含有数字，大小写，特殊字符；

2 为 strong 级别，密码难度更大一些，需要包括字典文件。

密码长度最低长为 4，当设置长度为 1、2、3 时，其长度依然为 4。



```
mysql> set global validate_password_policy=0;
Query OK, 0 rows affected (0.00 sec)

mysql> set global validate_password_length=4;
Query OK, 0 rows affected (0.00 sec)

mysql> alter user 'root'@'localhost' identified by '123456';
Query OK, 0 rows affected (0.00 sec)

mysql> \q
Bye
[root@slave2 ~]# 
```

设置密码强度为low

密码最低长度为4

修改本地root用户密码为123456

退出

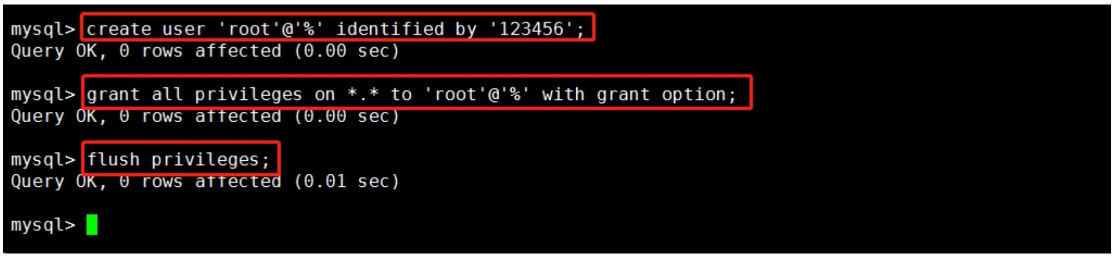
（7）设置远程登录

以新密码登陆 MySQL：mysql -uroot -p123456

创建用户：create user 'root'@'%' identified by '123456';

允许远程连接：grant all privileges on *.* to 'root'@'%' with grant option;

刷新权限：flush privileges;



```
mysql> create user 'root'@'%' identified by '123456';
Query OK, 0 rows affected (0.00 sec)

mysql> grant all privileges on *.* to 'root'@'%' with grant option;
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.01 sec)

mysql> 
```

4.2Slave1 上安装 hive

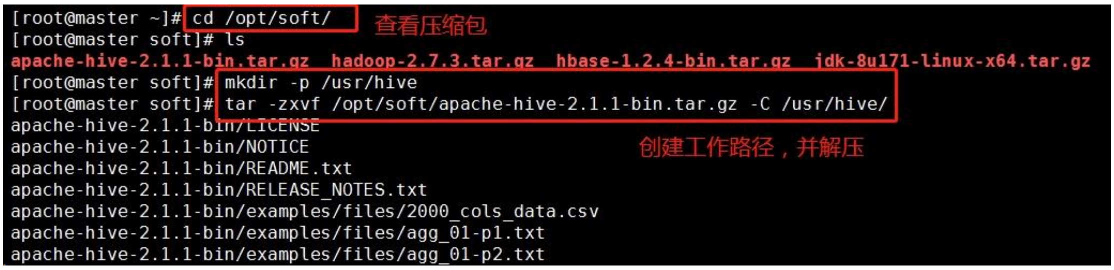
（1）首先我们需要创建工作路径，并将 hive 解压。环境中 master 作为客户端，slave1 作为服务器端，因此都需要使用到 hive。因为 hive 相关安装包存放在 master 中，因此我们先在 master 中对 hive 进行解压，然后将其复制到 slave1 中。

master 中操作如下：

cd /opt/soft

mkdir -p /usr/hive

tar -zxvf /opt/soft/apache-hive-2.1.1-bin.tar.gz -C /usr/hive/



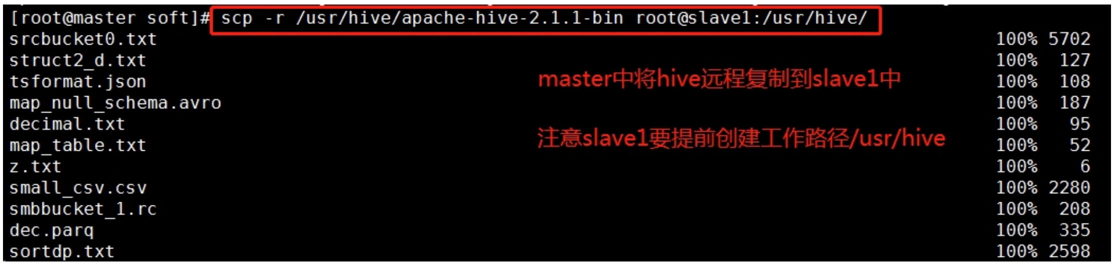
```
[root@master ~]# cd /opt/soft/
[root@master soft]# ls
apache-hive-2.1.1-bin.tar.gz  hadoop-2.7.3.tar.gz  hbase-1.2.4-bin.tar.gz  jdk-8u171-linux-x64.tar.gz
[root@master soft]# mkdir -p /usr/hive
[root@master soft]# tar -zxvf /opt/soft/apache-hive-2.1.1-bin.tar.gz -C /usr/hive/
apache-hive-2.1.1-bin/LICENSE
apache-hive-2.1.1-bin/NOTICE
apache-hive-2.1.1-bin/README.txt
apache-hive-2.1.1-bin/RELEASE_NOTES.txt
apache-hive-2.1.1-bin/examples/files/2000_cols_data.csv
apache-hive-2.1.1-bin/examples/files/agg_01-p1.txt
apache-hive-2.1.1-bin/examples/files/agg_01-p2.txt
```

查看压缩包

创建工作路径，并解压

同样 slave1 上建立文件夹/usr/hive，然后 master 中将安装包远程复制到 slave1。

scp -r /usr/hive/apache-hive-2.1.1-bin root@slave1:/usr/hive/



```
[root@master soft]# scp -r /usr/hive/apache-hive-2.1.1-bin root@slave1:/usr/hive/
srcbucket0.txt                                                           100% 5702
struct2_d.txt                                                            100%  127
tsformat.json                                                            100%  108
map_null_schema.avro                                                     100%  187
decimal.txt                                                              100%   95
map_table.txt                                                            100%   52
z.txt                                                                    100%    6
small_csv.csv                                                            100% 2280
smbbucket_1.rc                                                           100%  208
dec.parq                                                                 100%  335
sortdp.txt                                                               100% 2598
```

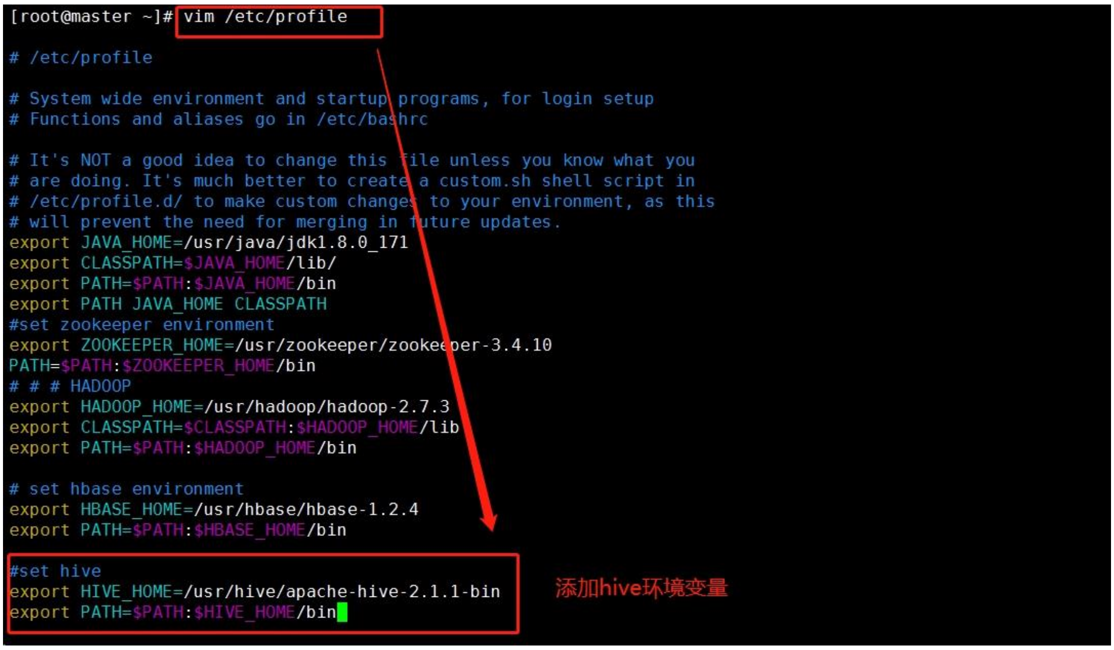
master中将hive远程复制到slave1中

注意slave1要提前创建工作路径/usr/hive

（2）修改/etc/profile 文件设置 hive 环境变量。（master 和 slave1 都执行）。

vim /etc/profile

```
#set hive

export HIVE_HOME=/usr/hive/apache-hive-2.1.1-bin

export PATH=$PATH:$HIVE_HOME/bin
```



```
[root@master ~]# vim /etc/profile

# /etc/profile

# System wide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc

# It's NOT a good idea to change this file unless you know what you
# are doing. It's much better to create a custom.sh shell script in
# /etc/profile.d/ to make custom changes to your environment, as this
# will prevent the need for merging in future updates.
export JAVA_HOME=/usr/java/jdk1.8.0_171
export CLASSPATH=$JAVA_HOME/lib/
export PATH=$PATH:$JAVA_HOME/bin
export PATH JAVA_HOME CLASSPATH
#set zookeeper environment
export ZOOKEEPER_HOME=/usr/zookeeper/zookeeper-3.4.10
PATH=$PATH:$ZOOKEEPER_HOME/bin
# # # HADOOP
export HADOOP_HOME=/usr/hadoop/hadoop-2.7.3
export CLASSPATH=$CLASSPATH:$HADOOP_HOME/lib
export PATH=$PATH:$HADOOP_HOME/bin

# set hbase environment
export HBASE_HOME=/usr/hbase/hbase-1.2.4
export PATH=$PATH:$HBASE_HOME/bin

#set hive
export HIVE_HOME=/usr/hive/apache-hive-2.1.1-bin
export PATH=$PATH:$HIVE_HOME/bin
```

生效环境变量:

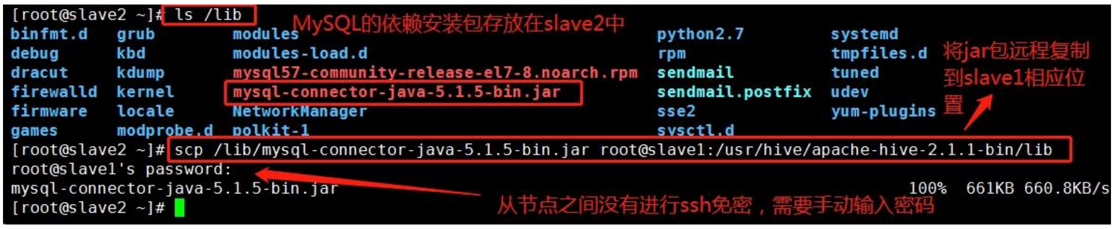
source /etc/profile

（3）因为服务端需要和 Mysql 通信，所以服务端需要 Mysql 的 lib 安装包到 Hive_Home/conf 目录下。

注意：mysql.jar 放在 slave2 中的/lib 目录下，需要将其远程复制到 slave1 的 hive 的 lib 中。

首先 slave2 中进行如下操作:

```
ls /lib

scp /lib/mysql-connector-java-5.1.5-bin.jar root@slave1:/usr/hive/apache-hive-2.1.1-bin/lib
```



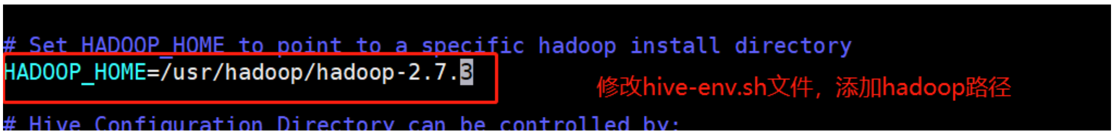
```
[root@slave2 ~]# ls /lib
binfmt.d    grub         modules                                python2.7         systemd
debug       kbd          modules-load.d                         rpm               tmpfiles.d
dracut      kdump        mysql57-community-release-el7-8.noarch.rpm sendmail      tuned
firewalld   kernel       mysql-connector-java-5.1.5-bin.jar     sendmail.postfix  udev
firmware    locale       NetworkManager                         sse2              yum-plugins
games       modprobe.d   polkit-1                               sysctl.d
[root@slave2 ~]# scp /lib/mysql-connector-java-5.1.5-bin.jar root@slave1:/usr/hive/apache-hive-2.1.1-bin/lib
root@slave1's password:
mysql-connector-java-5.1.5-bin.jar                                     100%  661KB 660.8KB/s
[root@slave2 ~]# 
```

MySQL的依赖安装包存放在slave2中

将jar包远程复制到slave1相应位置

从节点之间没有进行ssh免密，需要手动输入密码

（4）回到 slave1，修改 hive-env.sh 中 HADOOP_HOME 环境变量。



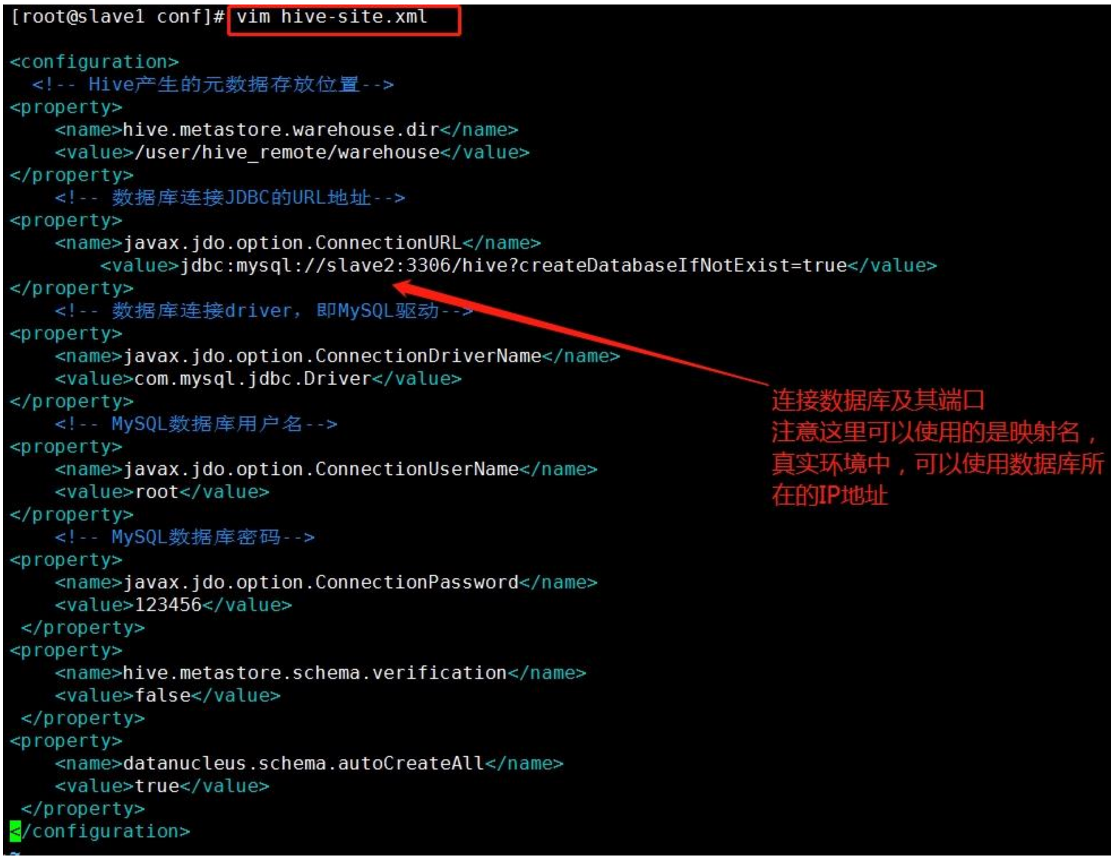
```
# Set HADOOP_HOME to point to a specific hadoop install directory
HADOOP_HOME=/usr/hadoop/hadoop-2.7.3
# Hive Configuration Directory can be controlled by:
```

修改hive-env.sh文件，添加hadoop路径

（5）修改 hive-site.xml 文件

```
<configuration>
  <!-- Hive 产生的元数据存放位置-->
<property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/user/hive_remote/warehouse</value>
</property>
    <!-- 数据库连接 JDBC 的 URL 地址-->
<property>
    <name>javax.jdo.option.ConnectionURL</name>
<value>jdbc:mysql://slave2:3306/hive?createDatabaseIfNotExist=true</value>
</property>
    <!-- 数据库连接 driver，即 MySQL 驱动-->
<property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
```

```
</property>
    <!-- MySQL 数据库用户名-->
<property>
    <name>javax.jdo.option.ConnectionUserName</name>
    <value>root</value>
</property>
    <!-- MySQL 数据库密码-->
<property>
    <name>javax.jdo.option.ConnectionPassword</name>
    <value>123456</value>
 </property>
<property>
    <name>hive.metastore.schema.verification</name>
    <value>false</value>
 </property>
<property>
    <name>datanucleus.schema.autoCreateAll</name>
    <value>true</value>
 </property>
</configuration>
```



```
[root@slave1 conf]# vim hive-site.xml

<configuration>
  <!-- Hive产生的元数据存放位置-->
<property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/user/hive_remote/warehouse</value>
</property>
    <!-- 数据库连接JDBC的URL地址-->
<property>
    <name>javax.jdo.option.ConnectionURL</name>
        <value>jdbc:mysql://slave2:3306/hive?createDatabaseIfNotExist=true</value>
</property>
    <!-- 数据库连接driver, 即MySQL驱动-->
<property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
</property>
    <!-- MySQL数据库用户名-->
<property>
    <name>javax.jdo.option.ConnectionUserName</name>
    <value>root</value>
</property>
    <!-- MySQL数据库密码-->
<property>
    <name>javax.jdo.option.ConnectionPassword</name>
    <value>123456</value>
 </property>
<property>
    <name>hive.metastore.schema.verification</name>
    <value>false</value>
 </property>
<property>
    <name>datanucleus.schema.autoCreateAll</name>
    <value>true</value>
 </property>
</configuration>
```

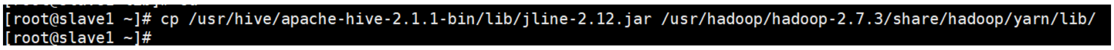
连接数据库及其端口
注意这里可以使用的是映射名，真实环境中，可以使用数据库所在的IP地址

4.3Master 作为客户端

（1）解决版本冲突和 jar 包依赖问题。

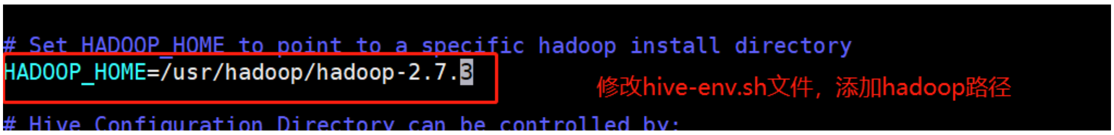
由于客户端需要和 Hadoop 通信，所以需要更改 Hadoop 中 jline 的版本。即保留一个高版本的 jline jar 包，从 hive 的 lib 包中拷贝到 Hadoop 中 lib 位置为 /usr/hadoop/hadoop-2.7.3/share/hadoop/yarn/lib。

```
cp /usr/hive/apache-hive-2.1.1-bin/lib/jline-2.12.jar /usr/hadoop/hadoop-2.7.3/share/hadoop/yarn/lib/
```



```
[root@slave1 ~]# cp /usr/hive/apache-hive-2.1.1-bin/lib/jline-2.12.jar /usr/hadoop/hadoop-2.7.3/share/hadoop/yarn/lib/
[root@slave1 ~]#
```

（2）修改 hive-env.sh

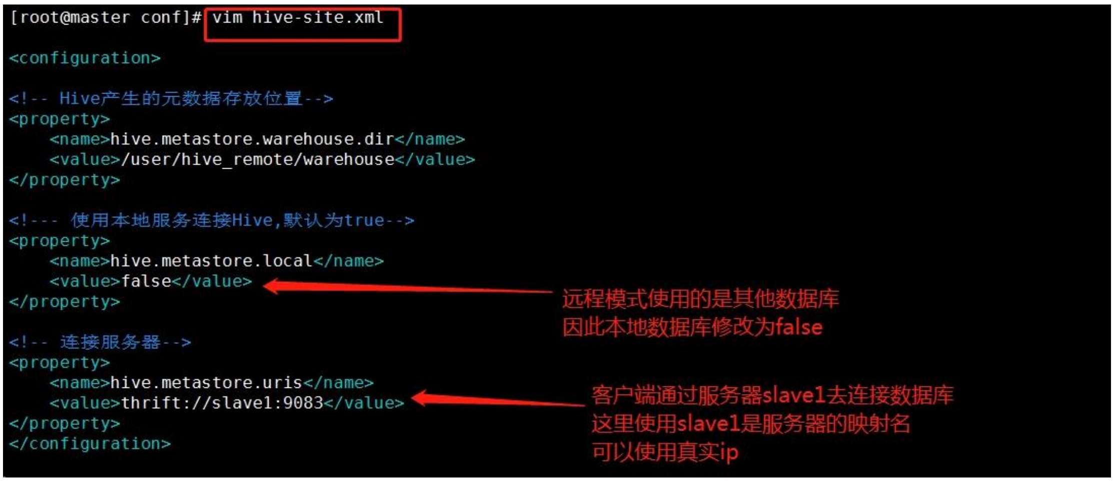


```
# Set HADOOP_HOME to point to a specific hadoop install directory
HADOOP_HOME=/usr/hadoop/hadoop-2.7.3

# Hive Configuration Directory can be controlled by:
```

修改hive-env.sh文件，添加hadoop路径

（3）修改 hive-site.xml

```
<configuration>
  <!-- Hive 产生的元数据存放位置-->
<property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/user/hive_remote/warehouse</value>
</property>
    <!--- 使用本地服务连接 Hive,默认为 true-->
<property>
    <name>hive.metastore.local</name>
    <value>false</value>
</property>
    <!-- 连接服务器-->
<property>
    <name>hive.metastore.uris</name>
<value>thrift://slave1:9083</value>
</property>
</configuration>
```



```
[root@master conf]# vim hive-site.xml

<configuration>

<!-- Hive产生的元数据存放位置-->
<property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/user/hive_remote/warehouse</value>
</property>

<!--- 使用本地服务连接Hive,默认为true-->
<property>
    <name>hive.metastore.local</name>
    <value>false</value>
</property>

<!-- 连接服务器-->
<property>
    <name>hive.metastore.uris</name>
    <value>thrift://slave1:9083</value>
</property>
</configuration>
```

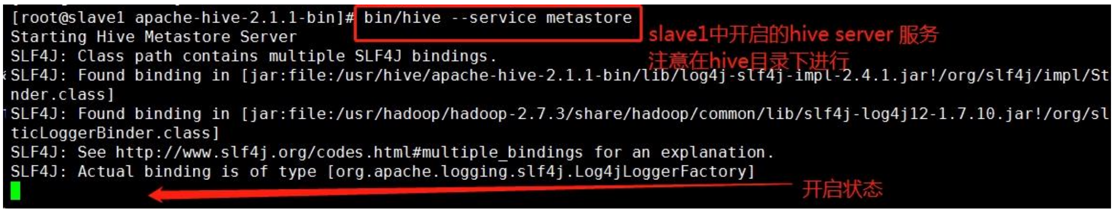
远程模式使用的是其他数据库
因此本地数据库修改为false

客户端通过服务器slave1去连接数据库
这里使用slave1是服务器的映射名
可以使用真实ip

4.4 成功启动 Hive

（1）启动 hive server（slave1 上）

bin/hive --service metastore



```
[root@slave1 apache-hive-2.1.1-bin]# bin/hive --service metastore
Starting Hive Metastore Server
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hive/apache-hive-2.1.1-bin/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/St
nder.class]
SLF4J: Found binding in [jar:file:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/sl
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
█
```

slave1中开启的hive server 服务
注意在hive目录下进行

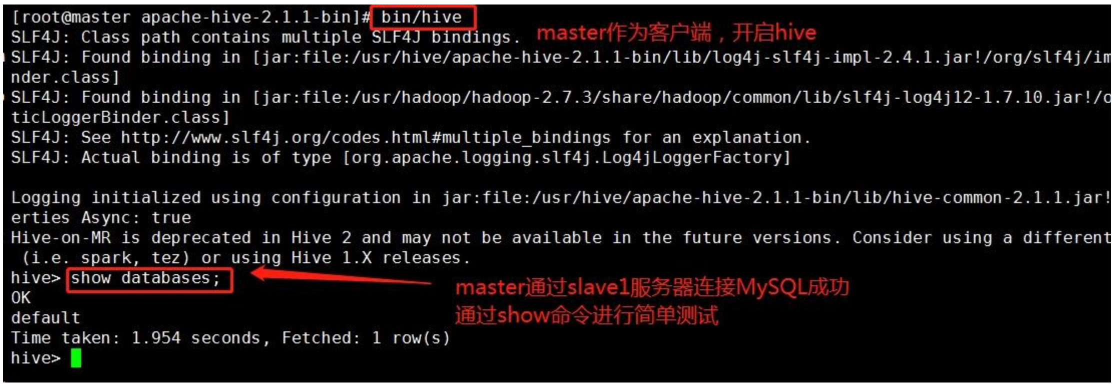
开启状态

（2）启动 hive client(master 上)

bin/hive

测试 hive 是否启动成功:

hive>show databases;

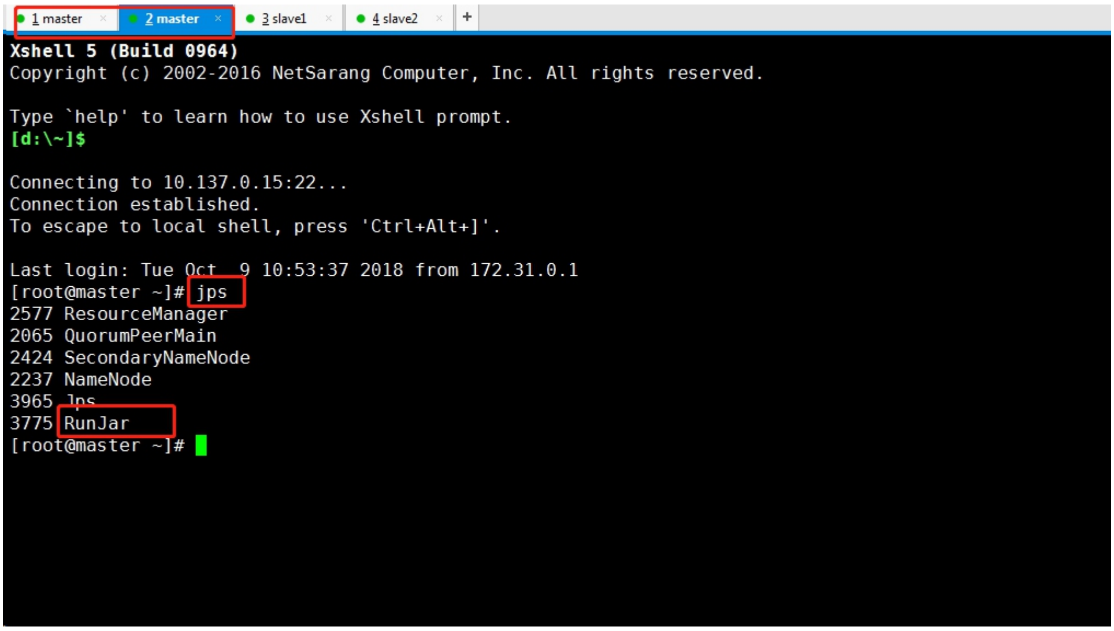


```
[root@master apache-hive-2.1.1-bin]# bin/hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hive/apache-hive-2.1.1-bin/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/im
nder.class]
SLF4J: Found binding in [jar:file:/usr/hadoop/hadoop-2.7.3/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/o
ticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/usr/hive/apache-hive-2.1.1-bin/lib/hive-common-2.1.1.jar!
erties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different
 (i.e. spark, tez) or using Hive 1.X releases.
hive> show databases;
OK
default
Time taken: 1.954 seconds, Fetched: 1 row(s)
hive> █
```

master作为客户端，开启hive

master通过slave1服务器连接MySQL成功
通过show命令进行简单测试

（3）最后 master 的进程如下:



```
Xshell 5 (Build 0964)
Copyright (c) 2002-2016 NetSarang Computer, Inc. All rights reserved.

Type `help' to learn how to use Xshell prompt.
[d:\~]$

Connecting to 10.137.0.15:22...
Connection established.
To escape to local shell, press 'Ctrl+Alt+]'.

Last login: Tue Oct  9 10:53:37 2018 from 172.31.0.1
[root@master ~]# jps
2577 ResourceManager
2065 QuorumPeerMain
2424 SecondaryNameNode
2237 NameNode
3965 Jps
3775 RunJar
[root@master ~]#
```

3. 数据采集

5.1 任务要求

（一）要求使用 Python 语言编写爬虫代码。

（二）要求爬取给定网站的所有发帖数据，包含回帖数据。

（三）将爬取的数据上传到 Hdfs 指定路径。

（四）将爬取的数据存放在制定 Hive 表中。

（五）统计总发帖数，将结果写入到 Hdfs 指定路径。

（六）统计总用户数，将结果写入到 Hdfs 指定路径。

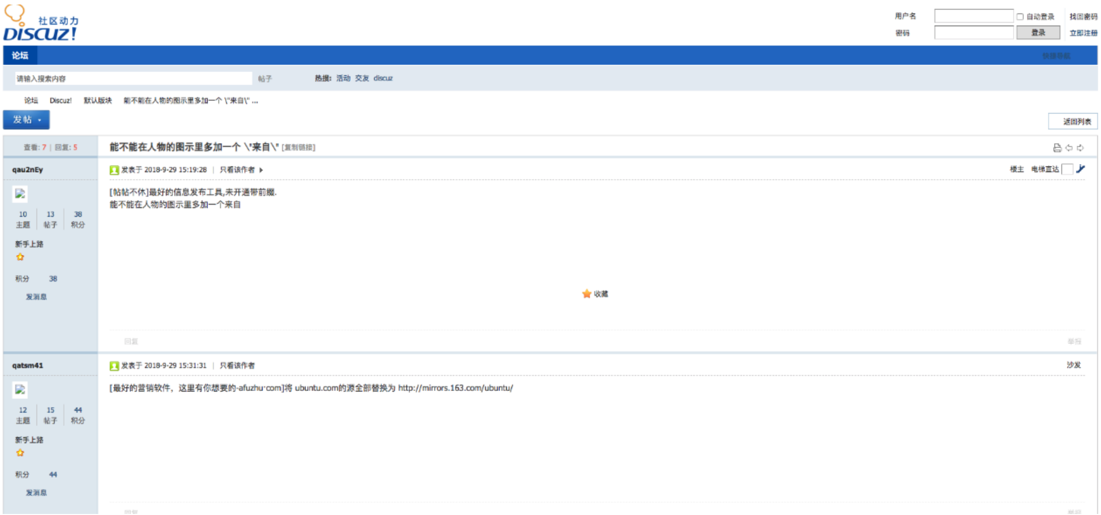
（七）统计发帖活跃用户 TOP10。

5.2 目标环境

Discuz!是目前国内知名的开源 php 社交系统。它的基础架构采用 PHP+MySQL 实现；适用于各种服务器环境的高效论坛系统。

直接访问目标站点 ip 即可进入论坛主页。论坛的默认模块包含 5800+条主题帖及 1700+条回复帖，共计 7500+条有效回复内容；包含 550+会员。

其中涉及到的信息包含：论坛版块、发帖人、回帖人、发帖人 ID、发帖人名称、回帖人 ID、回帖人名称、用户头像、发帖内容、回帖内容、发帖 ID、回帖 ID 等。



能不能在人物的图示里多加一个 \"来自\" [复制链接]

qau2nEy 发表于 2018-9-29 15:19:28 | 只看该作者

[帖帖不休]最好的信息发布工具,来开通带前缀.
能不能在人物的图示里多加一个来自

qatsm41 发表于 2018-9-29 15:31:31 | 只看该作者

[最好的营销软件，这里有你想要的-afuzhu·com]将 ubuntu.com的源全部替换为 http://mirrors.163.com/ubuntu/

图：论坛图

5.3 逻辑图

逻辑关系为：

（一）论坛版块对应多个帖子

（二）用户对应多个发帖

（三）用户对应多个回帖

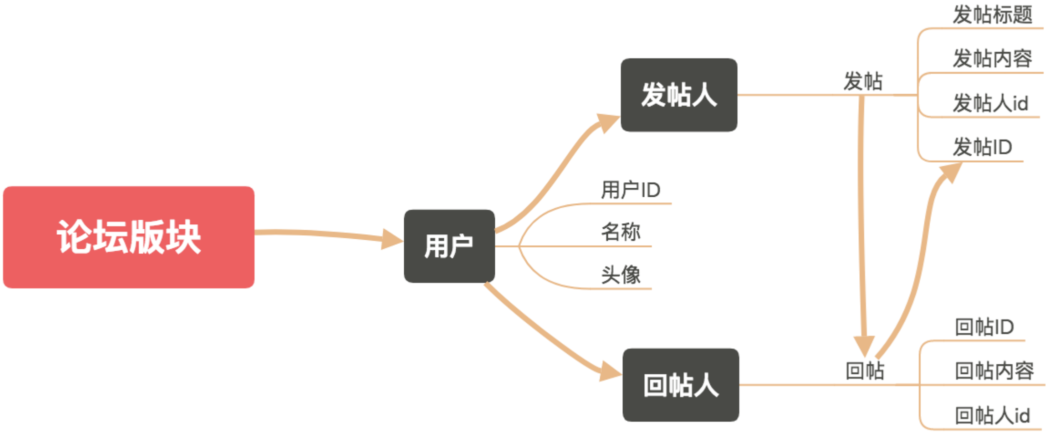
（四）发帖对应多个回帖

（五）发帖包含：发帖 id、发帖标题、发帖内容、发帖人 id

（六）回帖包含：发帖 id、回帖 id、回帖内容、回帖人 id

（七）用户包含：用户 id、名称、头像

逻辑关系图为下：

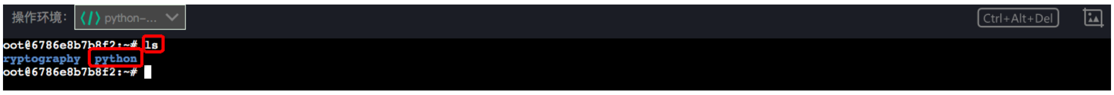


图：逻辑图

5.4 操作方法

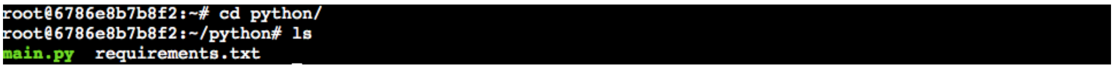
步骤一：代码实现

打开 python-env 的虚拟机，输入 ls 查看当前目录，找到 python 文件夹。



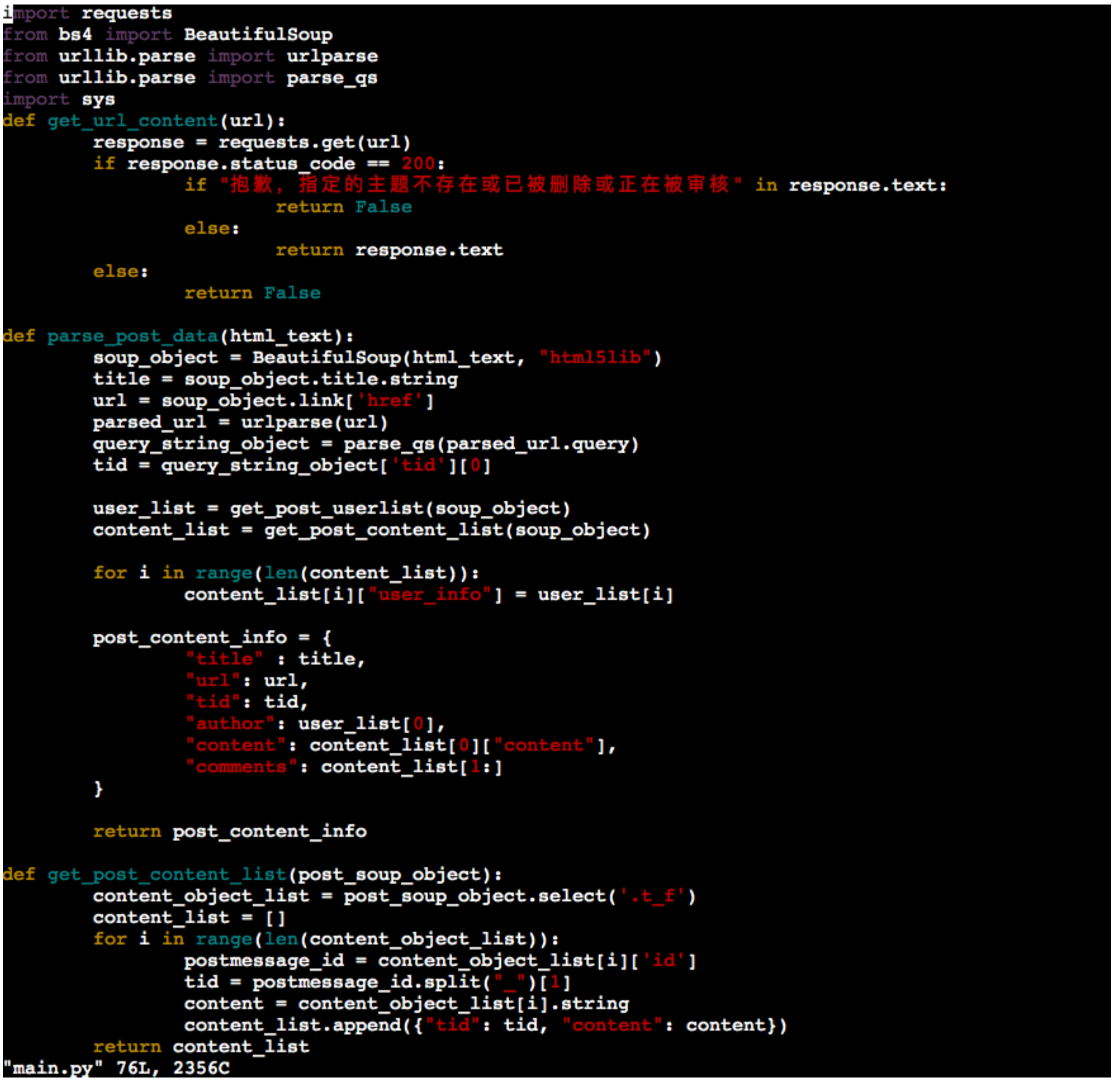
```
oot@6786e8b7b8f2:~# ls
ryptography  python
oot@6786e8b7b8f2:~# 
```

打开 python 文件夹，编写 main.py 文件。



```
root@6786e8b7b8f2:~# cd python/
root@6786e8b7b8f2:~/python# ls
main.py  requirements.txt
```

在 main.py 文件中编写爬虫代码。



```
import requests
from bs4 import BeautifulSoup
from urllib.parse import urlparse
from urllib.parse import parse_qs
import sys
def get_url_content(url):
        response = requests.get(url)
        if response.status_code == 200:
                if "抱歉，指定的主题不存在或已被删除或正在被审核" in response.text:
                        return False
                else:
                        return response.text
        else:
                return False

def parse_post_data(html_text):
        soup_object = BeautifulSoup(html_text, "html5lib")
        title = soup_object.title.string
        url = soup_object.link['href']
        parsed_url = urlparse(url)
        query_string_object = parse_qs(parsed_url.query)
        tid = query_string_object['tid'][0]

        user_list = get_post_userlist(soup_object)
        content_list = get_post_content_list(soup_object)

        for i in range(len(content_list)):
                content_list[i]["user_info"] = user_list[i]

        post_content_info = {
                "title" : title,
                "url": url,
                "tid": tid,
                "author": user_list[0],
                "content": content_list[0]["content"],
                "comments": content_list[1:]
        }

        return post_content_info

def get_post_content_list(post_soup_object):
        content_object_list = post_soup_object.select('.t_f')
        content_list = []
        for i in range(len(content_object_list)):
                postmessage_id = content_object_list[i]['id']
                tid = postmessage_id.split("_")[1]
                content = content_object_list[i].string
                content_list.append({"tid": tid, "content": content})
        return content_list
"main.py" 76L, 2356C
```

具体代码及解析如下：

```
import requests
```

##引入 request 库， requests 是 python 实现的简单易用的 HTTP 库，使用起来比 urllib 简洁很多

```
from bs4 import BeautifulSoup
```

##BS4 本身是一种对描述语言进行封装的函数操作模块，通过提供面向对象的操作方式将文档对象中的各种节点、标签、属性、内容等等都封装成了 python 中对象的属性，在查询操作过程中，通过调用指定的函数直接进行数据匹配检索操作，非常的简单非常的灵活。

```
from urllib.parse import urlparse
```

##使用 urlparse 包中的 urlparse 方法来解析 url

```
from urllib.parse import parse_qs
```

##主要用于分析 URL 中 query 组件的参数，返回一个 key-value 对应的字典格式

```
import sys
```

##这句语句告诉 Python，我们想要使用这个模块。sys 模块包含了与 Python 解释器和它的

环境有关的函数。

```
def get_url_content(url):
        response = requests.get(url)
##请求访问目标网站
```

```
        if response.status_code == 200:
##判断请求状态码（状态），返回值为 200 正常。
```

```
                if ″抱歉，指定的主题不存在或已被删除或正在被审核″ in
response.text:
                        return False
                else:
                        return response.text
        else:
                return False
##判断″抱歉，指定的主题不存在或已被删除或正在被审核″是否在 response.text 网页源
码（文本形式）中(是，返回 False，否，以文本形式返回网页源码)
```

```
def parse_post_data(html_text):
        soup_object = BeautifulSoup(html_text, ″html5lib″)
##解析网页源码
```

```
        title = soup_object.title.string
##获取 title 并转换成 string 类型
```

```
        url = soup_object.link[’href’]
##获取 href 后的链接里的内容
```

```
        parsed_url = urlparse(url)
##将 url 解析成 6 个部分（协议、位置、路径、参数、查询、片段）
```

```
        query_string_object = parse_qs(parsed_url.query)
##获取解析后元组中的 query 项
```

```
        tid = query_string_object[’tid’][0]
##获取解析后元组中第一个 tid，代表着发帖
```

```
        user_list = get_post_userlist(soup_object)
        content_list = get_post_content_list(soup_object)
```

```
        for i in range(len(content_list)):
                content_list[i][″user_info″] = user_list[i]
##获取发帖用户
```

```
  post_content_info = {
                ″title″ : title,
                ″url″: url,
                ″tid″: tid,
                ″author″: user_list[0],
                ″content″: content_list[0][″content″],
                ″comments″: content_list[1:]
          }

          return post_content_info
##将抓取的数据以字典的形式保存
```

```
def get_post_content_list(post_soup_object):
        content_object_list = post_soup_object.select(’.t_f’)
#选择器。源码中标有.t_f 的为获取对象
        content_list = []
#定义一个空列表
```

```
        for i in range(len(content_object_list)):
                postmessage_id = content_object_list[i][’id’]
##获取发帖 id
```

```
                tid = postmessage_id.split(″_″)[1]
##获取发帖 ID，以″_″进行切分，取第二个元素
```

```
                content = content_object_list[i].string
##将下角标为 i 的数据转换为 string 类型
```

```
                content_list.append({″tid″: tid, ″content″: content})
        return content_list
##向列表中添加元素
```

```
def get_post_userlist(post_soup_object):
```

```
        user_info_doms = post_soup_object.select(″.authi″)
##选择器，可以获得多条也可以获得单条数据
```

```
        user_list = []
##定义一个用户空列表
```

```
        for i in range(len(user_info_doms)):
                if i % 2 == 0:
                        user_name = user_info_doms[i].a.string
##将下角标为 i 的数据转换为 string 类型
```

```
                        uid = parse_qs(user_info_doms[i].a[’href’])[’uid’][0]
                        user_list.append({″user_name″: user_name, ″uid″: uid})
##向列表中添加元素
```

```
        return user_list
content=get_url_content(″http://10.135.0.8/forum.php?mod=viewthread&tid=5656″)
##调用 get_url_content 方法，请求″http://10.135.0.8/forum.php?mod=viewthread&tid=5656″，以文本形式返回网页源码
```

```
parsed_post_content_info = parse_post_data(content)
#调用 parse_post_data 方法
```

```
# print(json.dumps(parsed_post_content_info))
# print(content)
#parse_post_data(get_url_content(″http://192.168.15.122/forum.php?mod=viewthread&tid=5656″))
```

```
max_tid = sys.argv[1]
##sys.argv 实现从程序外部向程序传递参数
```

```
post_base_url = ″http://10.135.0.8/forum.php?mod=viewthread&tid=″
for i in range(int(max_tid)):

        ret = get_url_content(post_base_url + str(i))
#调用 get_url_content 方法请求目标网站
```

```
        if ret != False:
                parsed_post_data = parse_post_data(ret)
                print(″got post data,tid:%s″ % (parsed_post_data[″tid″]))
        else:
                print(″tid:%s not found,continue″ %(i))
##判断 ret 是否不等于 False
```

步骤二：数据上传（详细内容参照 Hadoop 安装）

（一）将爬取的数据上传到 Hdfs 指定路径。

（二）将爬取的数据存放在制定 Hive 表中。

步骤三：数据分析（详细内容参照数据分析）

（一）统计总发帖数，将结果写入到 Hdfs 指定路径。

（二）统计总用户数，将结果写入到 Hdfs 指定路径。

（三）统计发帖活跃用户 TOP10。

1. 数据分析

6.1 学习目标

（一）掌握将本地文件上传至 hdfs 指定路径技能

（二）掌握创建 hive 表，并将本地数据信息导入技能

（三）掌握创建表获取指定格式、指定信息技能

（四）掌握转化率计算等函数

6.2 数据集说明

该数据集为某购物平台在“双 11”之前和之后的过去 6 个月内的匿名用户的购物日志以及指示它们是否是重复购买者的标签信息。 由于隐私问题，数据采取的方式有偏差，所以这个数据集的统计结果会偏离平台购物的实际情况。 但是这不会影响解决方案的适用性。 在本次比赛阶段，数据集已经上传在平台节点 root 目录下，文件名称为 train_format2.csv，同学们可以练习使用。

数据格式的详细信息可以在下表中找到。

数据字段	定义
user_id	购物者的唯一 ID。
age_range	用户的年龄范围：1 为小于 18；2 为[18,24]；3 [25,29]；4 [30,34]；

	5 代表[35,39]； 6 为[40,49]； 7 和 8 为大于等于 50； 0 和 NULL 为未知。
gender	用户性别：0 为女；1 为男；2 和空为不详。
merchant_id	商家的唯一 ID。
label	值来自{0，1，-1，NULL}。 ’1’表示’user_id’是’merchant_id’的重复购买者，而’0’则相反。 “-1”表示“user_id”不是给定商家的新客户，因此超出了我们的预测。 但是，这些记录可能会提供更多信息。 “NULL”只在测试数据中出现，表明这是一对预测。
activity_log	{user_id，merchant_id}之间的交互记录集，其中每个记录是表示为“item_id：category_id：brand_id：time_stamp：action_type”的操作。 ’#’用来分隔两个相邻的元素。 记录没有以任何特定顺序排序。

activity_log 中字段含义为：

数据字段	定义
item_id	该项目的唯一 ID。
category _id	该项目所属类别的唯一 ID。
brand_id	该品牌的唯一 ID。
time_stamp	操作发生的日期（格式：mmdd）
action_type	它是一个枚举类型{0,1,2,3}，其中 0 表示点击，1 表示加入购物车，2 表示购买，3 表示加入收藏。

6.3 使用 hive 对数据进行操作

在 master 上执行：

（1）# start-all.sh （启动 hadoop）

（2）# zkServer.sh start（各个节点均执行）

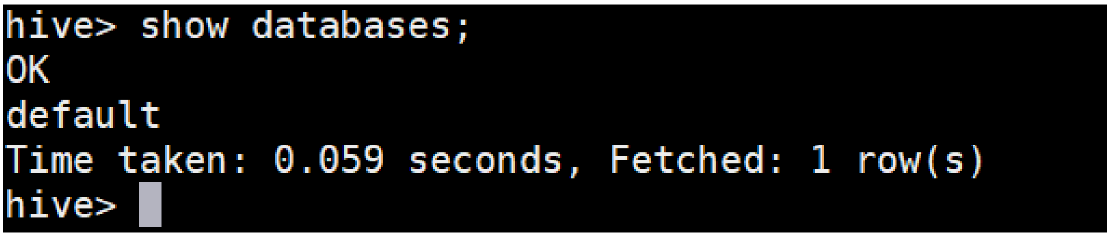
（3）# start-hbase.sh （直接运行这个命令需要将 HBASE 的 bin 目录也加入到/etc/environment 中）

（4）# 在 slave1 上输入命令：bin/hive --service metastore 启动 hive server，

然后在 master 节点上输入命令：bin/hive 启动 hive 客户端，当所有进程启动完全后方可执行以下操作。（所有命令需要在 hive 的安装目录下输入）

（一）查看数据库列表：（接下来操作我们每一步都可以看到返回值耗时）

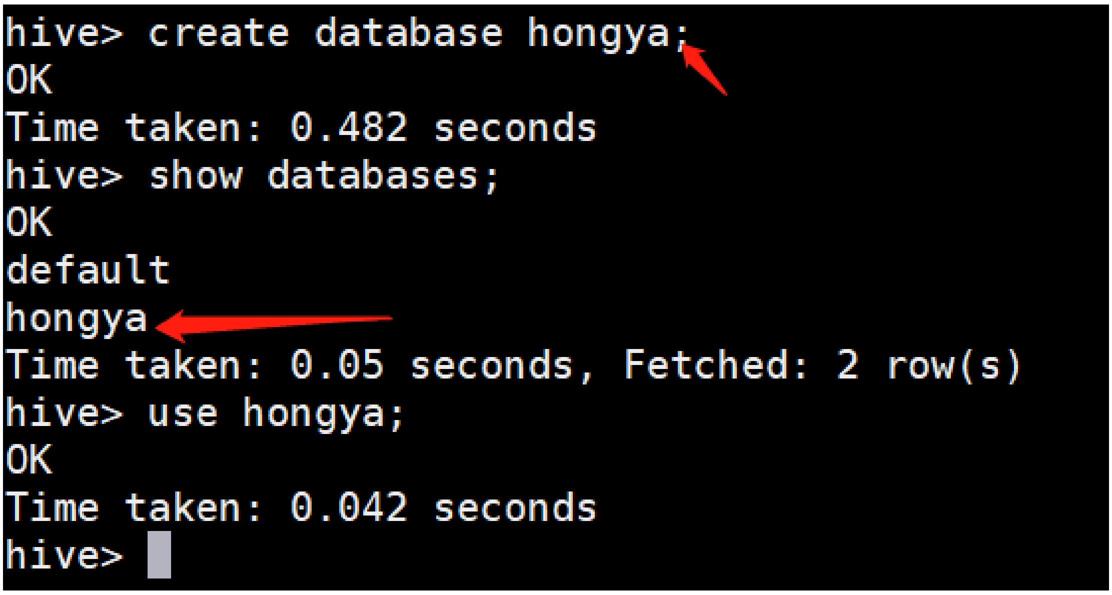
```
show databases;//查看数据库列表
```



```
hive> show databases;
OK
default
Time taken: 0.059 seconds, Fetched: 1 row(s)
hive>
```

（二）建数据库，建表：

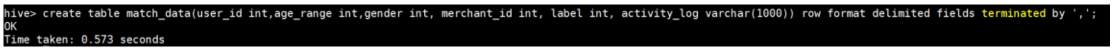
```
create database hongya; //创建数据库 hongya

show databases;//查看数据库，发现有库 hongya

use hongya;//使用 hongya 数据库
```



```
hive> create database hongya;
OK
Time taken: 0.482 seconds
hive> show databases;
OK
default
hongya
Time taken: 0.05 seconds, Fetched: 2 row(s)
hive> use hongya;
OK
Time taken: 0.042 seconds
hive>
```

创建比赛数据表 match_data，要求表结构与提供的数据结构一样，信息包含用户 iduser_id、用户性别信息 gender、商家唯一 id merchant_id、购物者标签 label，均为为 int 类型，用户与商家交互信息 activity_log 为 varchar 类型。

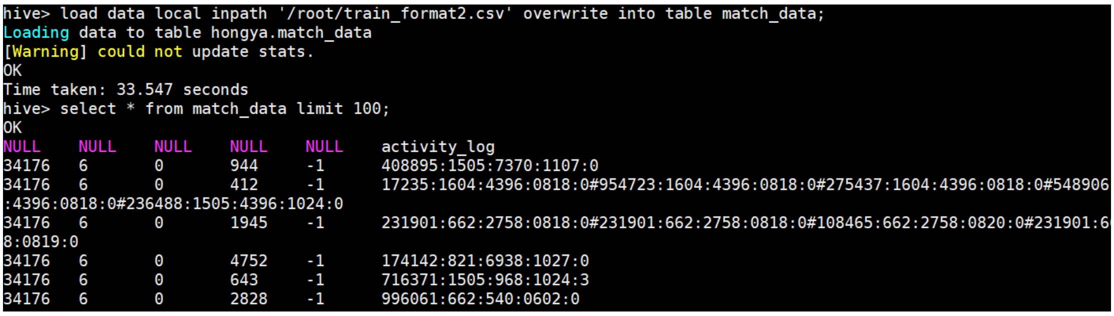
```
create table match_data(user_id int,age_range int,gender int, merchant_id int,
label int, activity_log varchar(1000)) row format delimited fields terminated by ',';
```



```
hive> create table match_data(user_id int,age_range int,gender int, merchant_id int, label int, activity_log varchar(1000)) row format delimited fields terminated by ',';
OK
Time taken: 0.573 seconds
```

将 root 下的 train_format2.csv 数据导入到创建的 match_data 表中

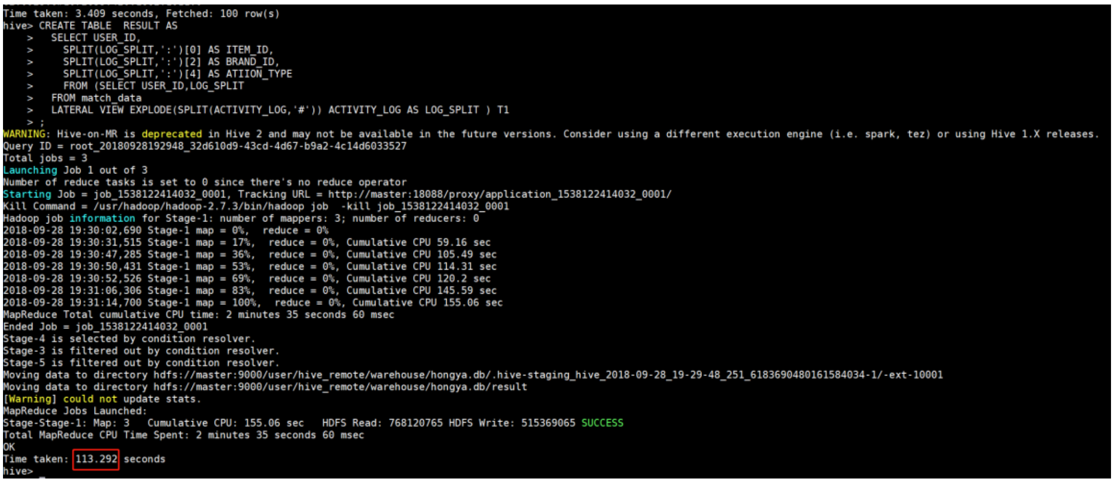
```
load data local inpath '/root/train_format2.csv' overwrite into table match_data;
select * from match_data limit 100;//查看 match_data 数据
```



```
hive> load data local inpath '/root/train_format2.csv' overwrite into table match_data;
Loading data to table hongya.match_data
[Warning] could not update stats.
OK
Time taken: 33.547 seconds
hive> select * from match_data limit 100;
OK
NULL    NULL    NULL    NULL    NULL    activity_log
34176   6       0       944     -1      408895:1505:7370:1107:0
34176   6       0       412     -1      17235:1604:4396:0818:0#954723:1604:4396:0818:0#275437:1604:4396:0818:0#548906
:4396:0818:0#236488:1505:4396:1024:0
34176   6       0       1945    -1      231901:662:2758:0818:0#231901:662:2758:0818:0#108465:662:2758:0820:0#231901:6
8:0819:0
34176   6       0       4752    -1      174142:821:6938:1027:0
34176   6       0       643     -1      716371:1505:968:1024:3
34176   6       0       2828    -1      996061:662:540:0602:0
```

```
    CREATE TABLE  RESULT AS //创建 RESULT 表并获取 match_data 的 USER_ID,
  SELECT USER_ID,
    SPLIT(LOG_SPLIT,':')[0] AS ITEM_ID, //将拆成行的数据以：为分隔符筛选字符
串第 0 位
    SPLIT(LOG_SPLIT,':')[2] AS BRAND_ID, //将拆成行的数据以：为分隔符筛选字
符串第 2 位
    SPLIT(LOG_SPLIT,':')[4] AS ATIION_TYPE //将拆成行的数据以：为分隔符筛选
字符串第 4 位
    FROM (SELECT USER_ID,LOG_SPLIT
  FROM match_data
  LATERAL VIEW EXPLODE(SPLIT(ACTIVITY_LOG,'#')) ACTIVITY_LOG AS
LOG_SPLIT ) T1
```

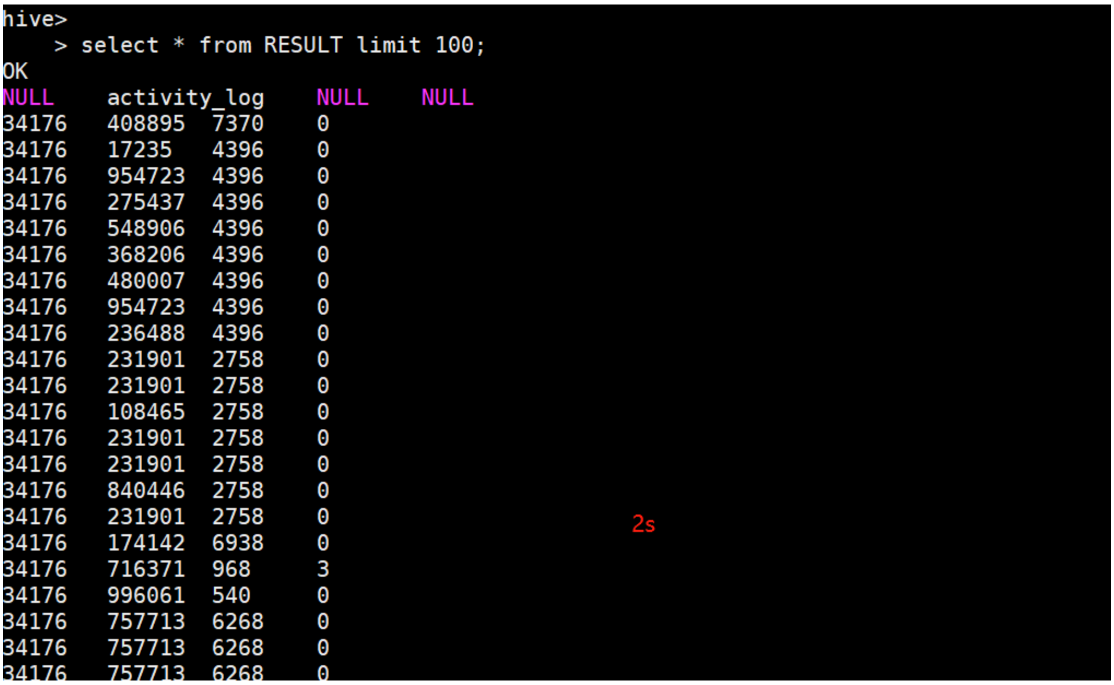
; //lateral view 和 split，explode 一起使用，以#为分隔符将一列数据拆成多行数据



```
Time taken: 3.409 seconds, Fetched: 100 row(s)
hive> CREATE TABLE  RESULT AS
    >   SELECT USER_ID,
    >     SPLIT(LOG_SPLIT,':')[0] AS ITEM_ID,
    >     SPLIT(LOG_SPLIT,':')[2] AS BRAND_ID,
    >     SPLIT(LOG_SPLIT,':')[4] AS ATIION_TYPE
    >     FROM (SELECT USER_ID,LOG_SPLIT
    >   FROM match_data
    >   LATERAL VIEW EXPLODE(SPLIT(ACTIVITY_LOG,'#')) ACTIVITY_LOG AS LOG_SPLIT ) T1
    > ;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = root_20180928192948_32d610d9-43cd-4d67-b9a2-4c14d6033527
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1538122414032_0001, Tracking URL = http://master:18088/proxy/application_1538122414032_0001/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0001
Hadoop job information for Stage-1: number of mappers: 3; number of reducers: 0
2018-09-28 19:30:02,690 Stage-1 map = 0%,  reduce = 0%
2018-09-28 19:30:31,515 Stage-1 map = 17%,  reduce = 0%, Cumulative CPU 59.16 sec
2018-09-28 19:30:47,285 Stage-1 map = 36%,  reduce = 0%, Cumulative CPU 105.49 sec
2018-09-28 19:30:50,431 Stage-1 map = 53%,  reduce = 0%, Cumulative CPU 114.31 sec
2018-09-28 19:30:52,526 Stage-1 map = 69%,  reduce = 0%, Cumulative CPU 120.2 sec
2018-09-28 19:31:06,306 Stage-1 map = 83%,  reduce = 0%, Cumulative CPU 145.59 sec
2018-09-28 19:31:14,700 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 155.06 sec
MapReduce Total cumulative CPU time: 2 minutes 35 seconds 60 msec
Ended Job = job_1538122414032_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://master:9000/user/hive_remote/warehouse/hongya.db/.hive-staging_hive_2018-09-28_19-29-48_251_6183690480161584034-1/-ext-10001
Moving data to directory hdfs://master:9000/user/hive_remote/warehouse/hongya.db/result
[Warning] could not update stats.
MapReduce Jobs Launched:
Stage-Stage-1: Map: 3   Cumulative CPU: 155.06 sec   HDFS Read: 768120765 HDFS Write: 515369065 SUCCESS
Total MapReduce CPU Time Spent: 2 minutes 35 seconds 60 msec
OK
Time taken: 113.292 seconds
hive>
```

查看表 RESULT 中前 100 行数据

select * from RESULT limit 100;//查看前 100 行数据



```
hive>
    > select * from RESULT limit 100;
OK
NULL    activity_log    NULL    NULL
34176   408895  7370    0
34176   17235   4396    0
34176   954723  4396    0
34176   275437  4396    0
34176   548906  4396    0
34176   368206  4396    0
34176   480007  4396    0
34176   954723  4396    0
34176   236488  4396    0
34176   231901  2758    0
34176   231901  2758    0
34176   108465  2758    0
34176   231901  2758    0
34176   231901  2758    0
34176   840446  2758    0
34176   231901  2758    0
34176   174142  6938    0
34176   716371  968     3
34176   996061  540     0
34176   757713  6268    0
34176   757713  6268    0
34176   757713  6268    0
```

2s

CREATE TABLE　CLICK AS //创建表 click，代表点击量

SELECT ITEM_ID,COUNT(1) COUNT_1//对所有的行 ITEM_ID 相同的进行统计

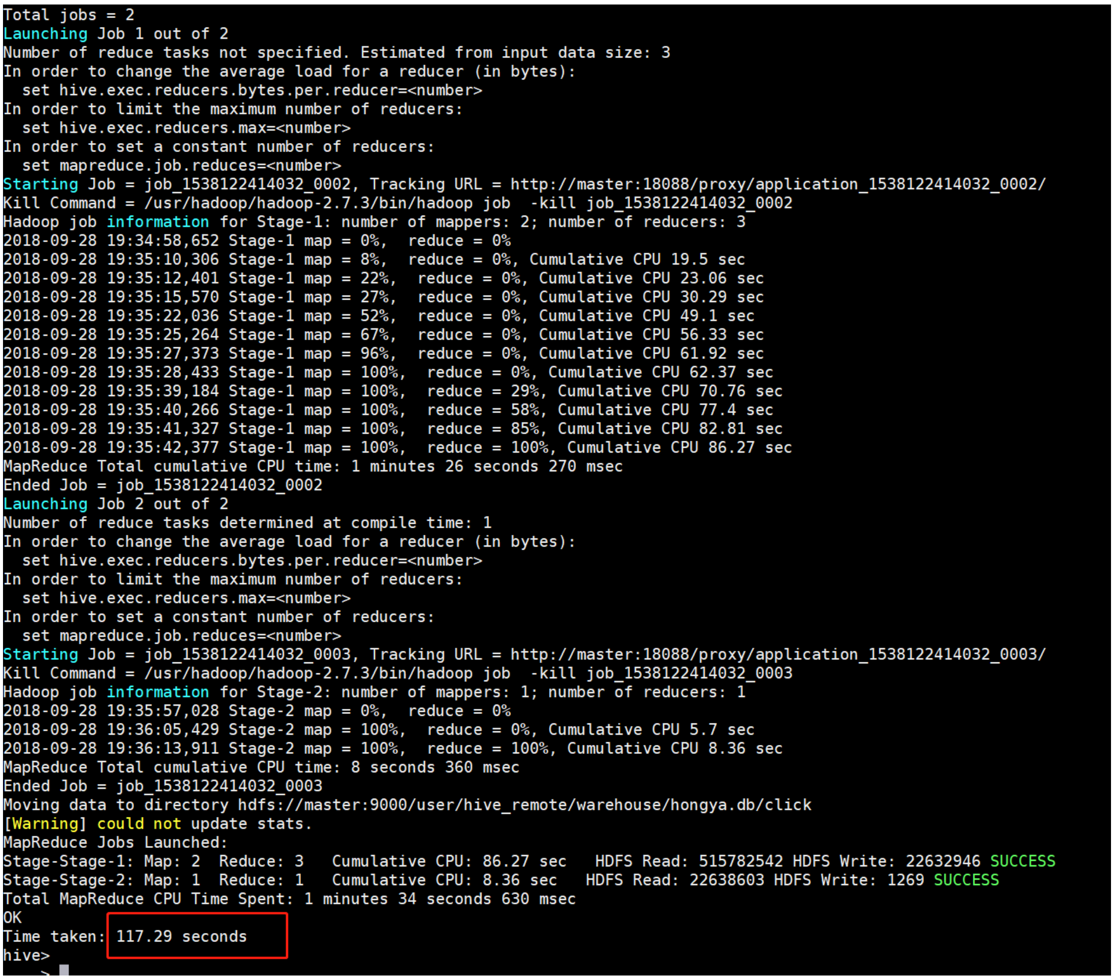
FROM RESULT

WHERE ATIION_TYPE = '0'//限定条件 ATIION_TYPE = '0'

GROUP BY ITEM_ID// group by 操作表示按照 ITEM_ID 字段的值进行分组，有相同的 ITEM_ID 值放到一起

ORDER BY COUNT_1 DESC//按照统计结果全局降序排序

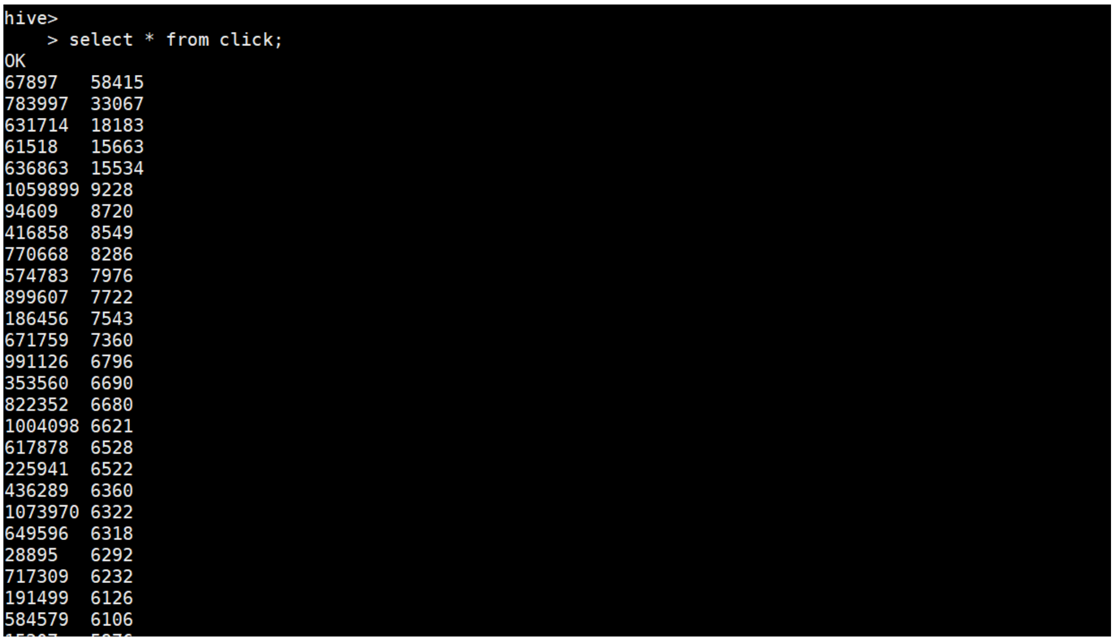
LIMIT 100;//限制数据 100 行



```
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1538122414032_0002, Tracking URL = http://master:18088/proxy/application_1538122414032_0002/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0002
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 3
2018-09-28 19:34:58,652 Stage-1 map = 0%,  reduce = 0%
2018-09-28 19:35:10,306 Stage-1 map = 8%,  reduce = 0%, Cumulative CPU 19.5 sec
2018-09-28 19:35:12,401 Stage-1 map = 22%,  reduce = 0%, Cumulative CPU 23.06 sec
2018-09-28 19:35:15,570 Stage-1 map = 27%,  reduce = 0%, Cumulative CPU 30.29 sec
2018-09-28 19:35:22,036 Stage-1 map = 52%,  reduce = 0%, Cumulative CPU 49.1 sec
2018-09-28 19:35:25,264 Stage-1 map = 67%,  reduce = 0%, Cumulative CPU 56.33 sec
2018-09-28 19:35:27,373 Stage-1 map = 96%,  reduce = 0%, Cumulative CPU 61.92 sec
2018-09-28 19:35:28,433 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 62.37 sec
2018-09-28 19:35:39,184 Stage-1 map = 100%,  reduce = 29%, Cumulative CPU 70.76 sec
2018-09-28 19:35:40,266 Stage-1 map = 100%,  reduce = 58%, Cumulative CPU 77.4 sec
2018-09-28 19:35:41,327 Stage-1 map = 100%,  reduce = 85%, Cumulative CPU 82.81 sec
2018-09-28 19:35:42,377 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 86.27 sec
MapReduce Total cumulative CPU time: 1 minutes 26 seconds 270 msec
Ended Job = job_1538122414032_0002
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1538122414032_0003, Tracking URL = http://master:18088/proxy/application_1538122414032_0003/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0003
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-09-28 19:35:57,028 Stage-2 map = 0%,  reduce = 0%
2018-09-28 19:36:05,429 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 5.7 sec
2018-09-28 19:36:13,911 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 8.36 sec
MapReduce Total cumulative CPU time: 8 seconds 360 msec
Ended Job = job_1538122414032_0003
Moving data to directory hdfs://master:9000/user/hive_remote/warehouse/hongya.db/click
[Warning] could not update stats.
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 3   Cumulative CPU: 86.27 sec   HDFS Read: 515782542 HDFS Write: 22632946 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 8.36 sec   HDFS Read: 22638603 HDFS Write: 1269 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 34 seconds 630 msec
OK
Time taken: 117.29 seconds
hive>
    >
```

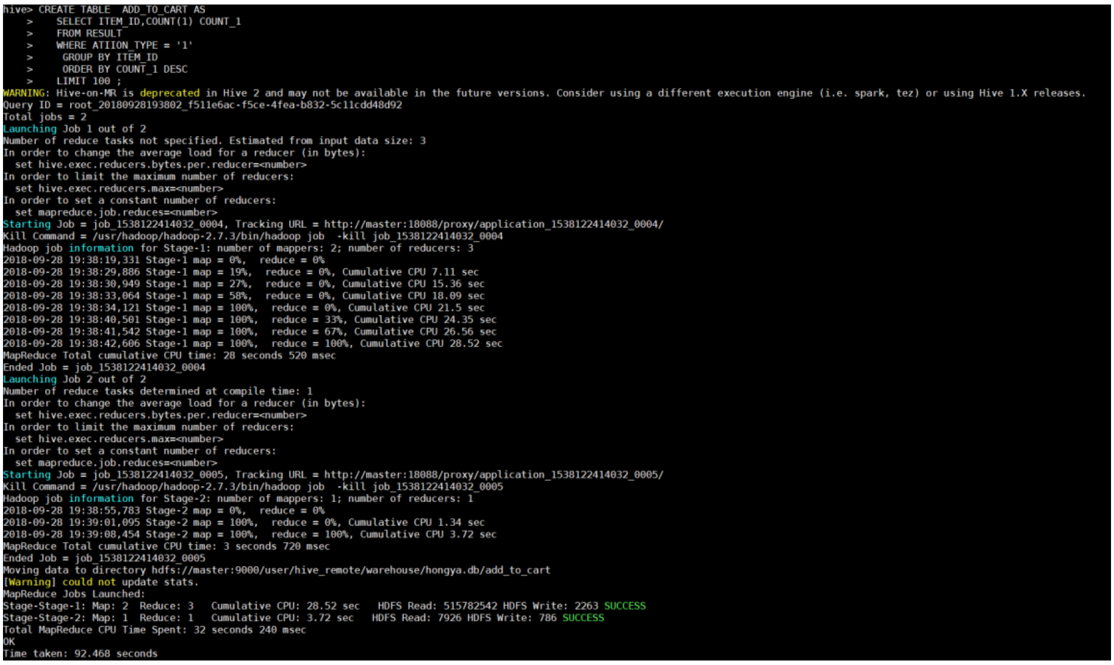
查看 click 表中所有数据

select * from click;// 查看 click 表中所有数据



```
hive>
    > select * from click;
OK
67897   58415
783997  33067
631714  18183
61518   15663
636863  15534
1059899 9228
94609   8720
416858  8549
770668  8286
574783  7976
899607  7722
186456  7543
671759  7360
991126  6796
353560  6690
822352  6680
1004098 6621
617878  6528
225941  6522
436289  6360
1073970 6322
649596  6318
28895   6292
717309  6232
191499  6126
584579  6106
```

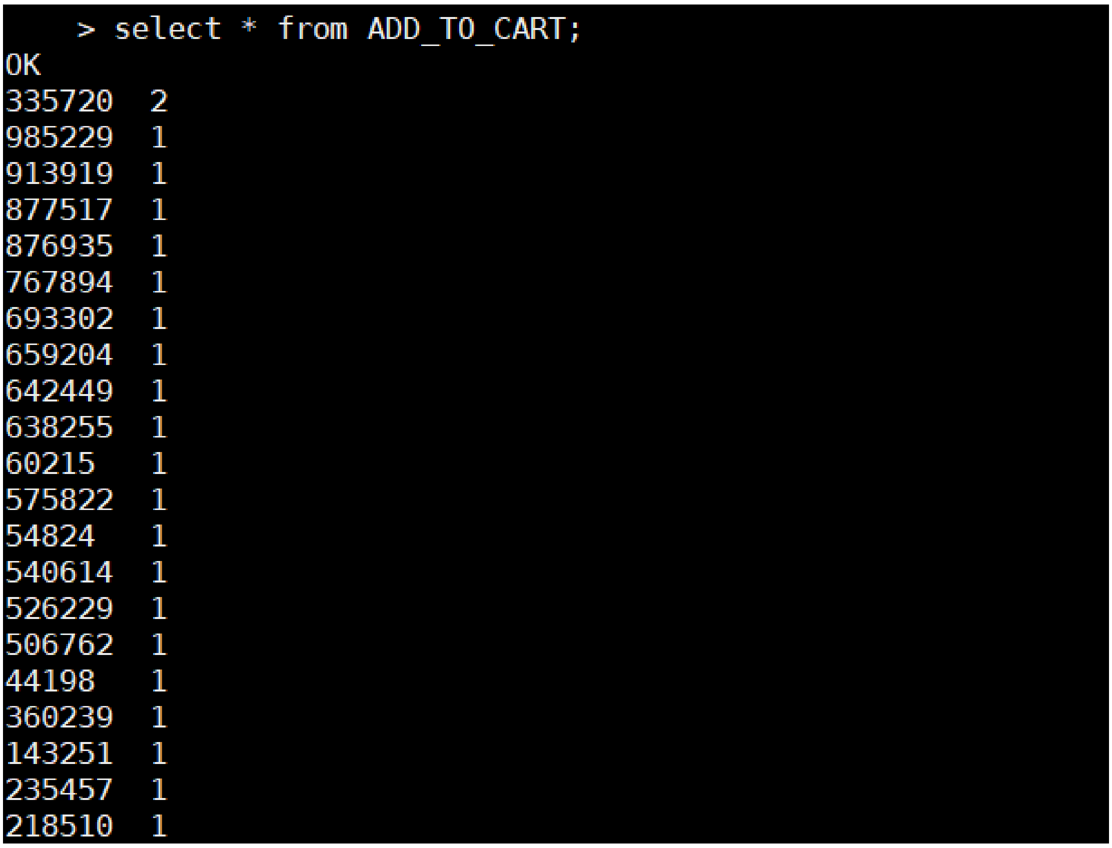
```
CREATE TABLE    ADD_TO_CART AS //创建表 ADD_TO_CART，代表加入购物车量
    SELECT ITEM_ID,COUNT(1) COUNT_1 对所有的行 ITEM_ID 相同的进行统计
    FROM RESULT
    WHERE ATIION_TYPE = '1' //限定条件 ATIION_TYPE = '1'
    GROUP BY ITEM_ID // group by 操作表示按照 ITEM_ID 字段的值进行分组，有相同的 ITEM_ID 值放到一起
    ORDER BY COUNT_1 DESC //按照统计结果全局降序排序
    LIMIT 100 ; //限制数据 100 行
```



```
hive> CREATE TABLE  ADD_TO_CART AS
    >    SELECT ITEM_ID,COUNT(1) COUNT_1
    >    FROM RESULT
    >    WHERE ATIION_TYPE = '1'
    >     GROUP BY ITEM_ID
    >     ORDER BY COUNT_1 DESC
    >    LIMIT 100 ;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = root_20180928193802_f511e6ac-f5ce-4fea-b832-5c11cdd48d92
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1538122414032_0004, Tracking URL = http://master:18088/proxy/application_1538122414032_0004/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0004
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 3
2018-09-28 19:38:19,331 Stage-1 map = 0%,  reduce = 0%
2018-09-28 19:38:29,886 Stage-1 map = 19%,  reduce = 0%, Cumulative CPU 7.11 sec
2018-09-28 19:38:30,949 Stage-1 map = 27%,  reduce = 0%, Cumulative CPU 15.36 sec
2018-09-28 19:38:33,064 Stage-1 map = 58%,  reduce = 0%, Cumulative CPU 18.09 sec
2018-09-28 19:38:34,121 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 21.5 sec
2018-09-28 19:38:40,501 Stage-1 map = 100%,  reduce = 33%, Cumulative CPU 24.35 sec
2018-09-28 19:38:41,542 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 26.56 sec
2018-09-28 19:38:42,606 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 28.52 sec
MapReduce Total cumulative CPU time: 28 seconds 520 msec
Ended Job = job_1538122414032_0004
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1538122414032_0005, Tracking URL = http://master:18088/proxy/application_1538122414032_0005/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0005
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-09-28 19:38:55,783 Stage-2 map = 0%,  reduce = 0%
2018-09-28 19:39:01,095 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.34 sec
2018-09-28 19:39:08,454 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 3.72 sec
MapReduce Total cumulative CPU time: 3 seconds 720 msec
Ended Job = job_1538122414032_0005
Moving data to directory hdfs://master:9000/user/hive_remote/warehouse/hongya.db/add_to_cart
[Warning] could not update stats.
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 3   Cumulative CPU: 28.52 sec   HDFS Read: 515782542 HDFS Write: 2263 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 3.72 sec   HDFS Read: 7926 HDFS Write: 786 SUCCESS
Total MapReduce CPU Time Spent: 32 seconds 240 msec
OK
Time taken: 92.468 seconds
```

查看 ADD_TO_CART 表中所有数据

```
select * from ADD_TO_CART;// 查看 ADD_TO_CART 表中所有数据
```



```
    > select * from ADD_TO_CART;
OK
335720  2
985229  1
913919  1
877517  1
876935  1
767894  1
693302  1
659204  1
642449  1
638255  1
60215   1
575822  1
54824   1
540614  1
526229  1
506762  1
44198   1
360239  1
143251  1
235457  1
218510  1
```

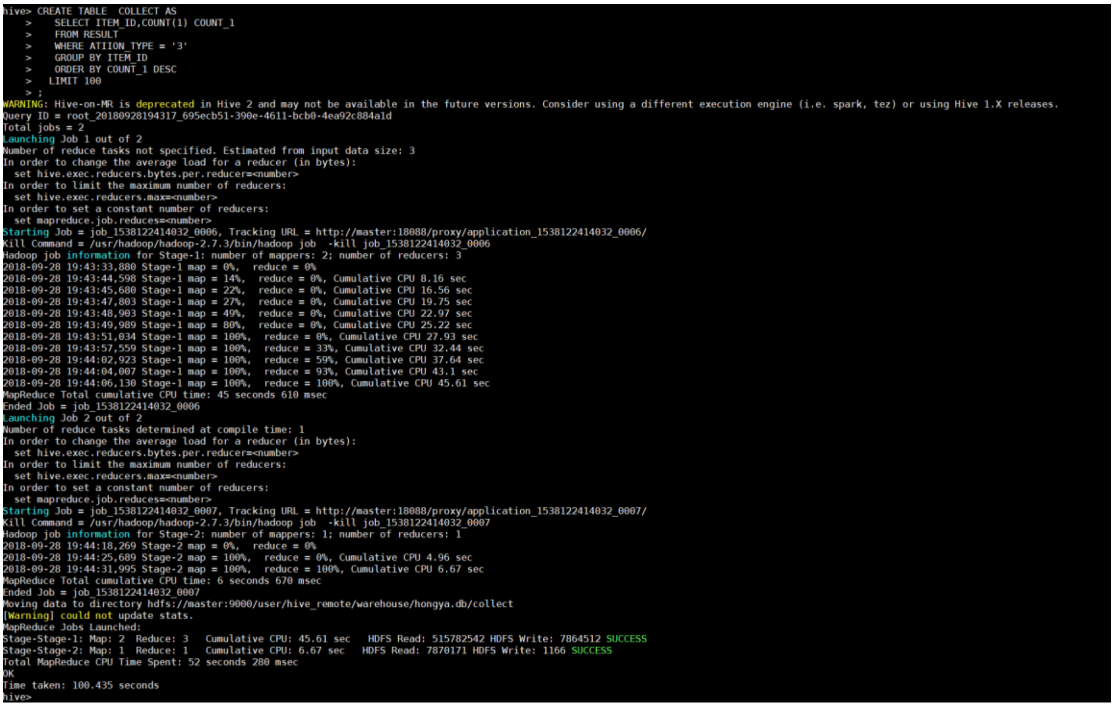
创建 collect 表

```
CREATE TABLE   COLLECT AS //创建表 COLLECT，代表收藏量
```

```
    SELECT ITEM_ID,COUNT(1) COUNT_1 //对所有的行 ITEM_ID 相同的进行统计

    FROM RESULT

    WHERE ATIION_TYPE = '3' //限定条件 ATIION_TYPE = '3'

    GROUP BY ITEM_ID // group by 操作表示按照 ITEM_ID 字段的值进行分组,

有相同的 ITEM_ID 值放到一起

    ORDER BY COUNT_1 DESC //按照统计结果全局降序排序

  LIMIT 100; //限制数据 100 行
```

查看 COLLECT 表中所有数据

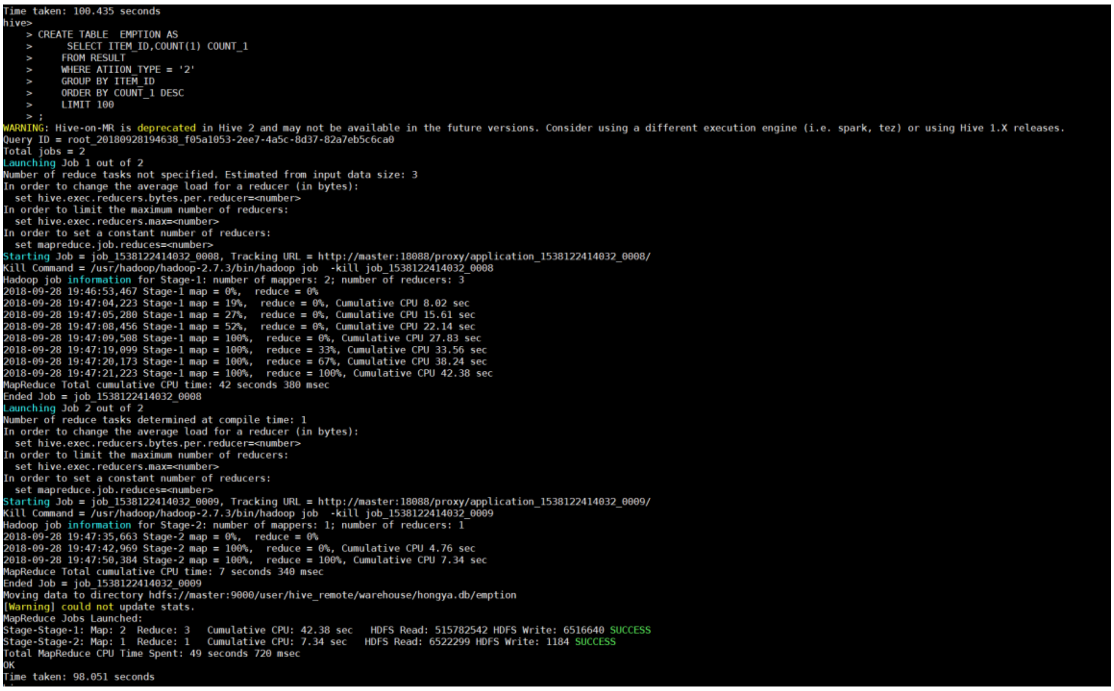
```
select * from COLLECT;     //查看 COLLECT 表中所有数据
```



```
hive> CREATE TABLE  COLLECT AS
    >    SELECT ITEM_ID,COUNT(1) COUNT_1
    >    FROM RESULT
    >    WHERE ATIION_TYPE = '3'
    >    GROUP BY ITEM_ID
    >    ORDER BY COUNT_1 DESC
    >   LIMIT 100
    > ;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = root_20180928194317_695ecb51-390e-4611-bcb0-4ea92c884a1d
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1538122414032_0006, Tracking URL = http://master:18088/proxy/application_1538122414032_0006/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0006
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 3
2018-09-28 19:43:33,880 Stage-1 map = 0%,  reduce = 0%
2018-09-28 19:43:44,598 Stage-1 map = 14%,  reduce = 0%, Cumulative CPU 8.16 sec
2018-09-28 19:43:45,680 Stage-1 map = 22%,  reduce = 0%, Cumulative CPU 16.56 sec
2018-09-28 19:43:47,803 Stage-1 map = 27%,  reduce = 0%, Cumulative CPU 19.75 sec
2018-09-28 19:43:48,903 Stage-1 map = 49%,  reduce = 0%, Cumulative CPU 22.97 sec
2018-09-28 19:43:49,989 Stage-1 map = 80%,  reduce = 0%, Cumulative CPU 25.22 sec
2018-09-28 19:43:51,034 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 27.93 sec
2018-09-28 19:43:57,559 Stage-1 map = 100%,  reduce = 33%, Cumulative CPU 32.44 sec
2018-09-28 19:44:02,923 Stage-1 map = 100%,  reduce = 59%, Cumulative CPU 37.64 sec
2018-09-28 19:44:04,007 Stage-1 map = 100%,  reduce = 93%, Cumulative CPU 43.1 sec
2018-09-28 19:44:06,130 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 45.61 sec
MapReduce Total cumulative CPU time: 45 seconds 610 msec
Ended Job = job_1538122414032_0006
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1538122414032_0007, Tracking URL = http://master:18088/proxy/application_1538122414032_0007/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0007
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-09-28 19:44:18,269 Stage-2 map = 0%,  reduce = 0%
2018-09-28 19:44:25,689 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 4.96 sec
2018-09-28 19:44:31,995 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 6.67 sec
MapReduce Total cumulative CPU time: 6 seconds 670 msec
Ended Job = job_1538122414032_0007
Moving data to directory hdfs://master:9000/user/hive_remote/warehouse/hongya.db/collect
[Warning] could not update stats.
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 3   Cumulative CPU: 45.61 sec   HDFS Read: 515782542 HDFS Write: 7864512 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 6.67 sec   HDFS Read: 7870171 HDFS Write: 1166 SUCCESS
Total MapReduce CPU Time Spent: 52 seconds 280 msec
OK
Time taken: 100.435 seconds
hive>
```

```
创建创建表 EMPTION, 插入购买数据

CREATE TABLE   EMPTION AS //创建表 EMPTION, 代表购买量

      SELECT ITEM_ID,COUNT(1) COUNT_1 //对所有的行 ITEM_ID 相同的进行统

计

     FROM RESULT
```

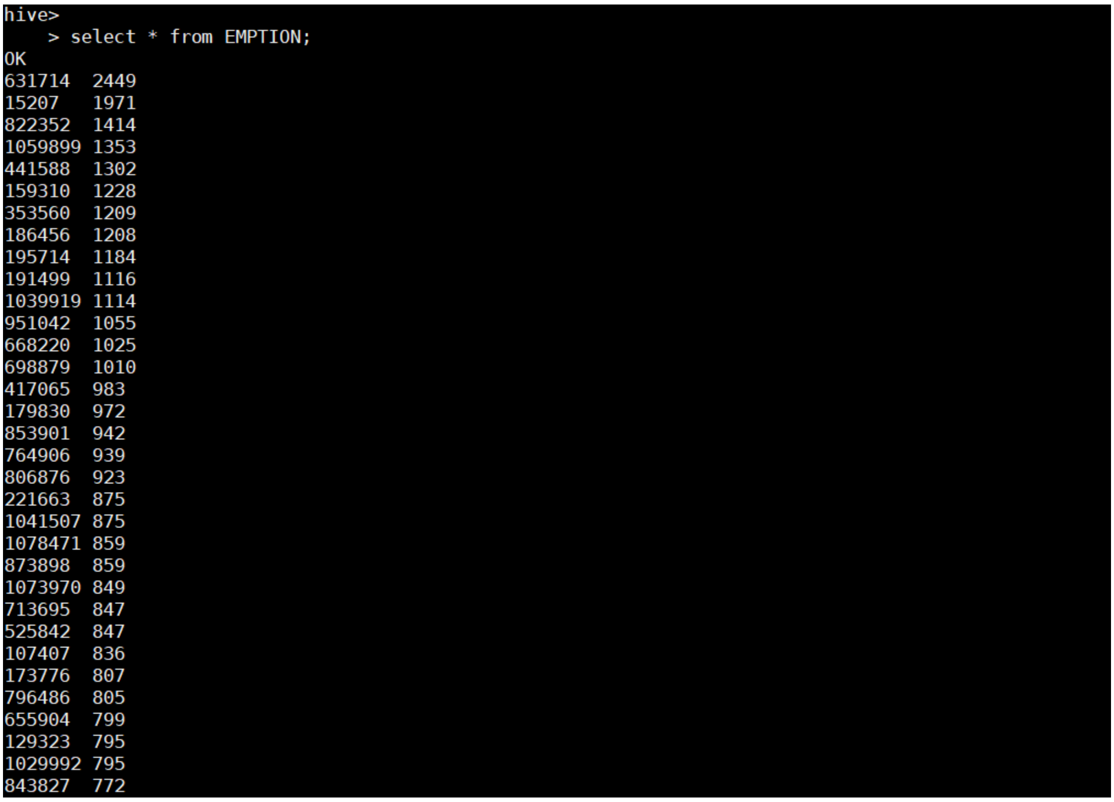
```
    WHERE ATIION_TYPE = '2' //限定条件 ATIION_TYPE = '2'
    GROUP BY ITEM_ID // group by 操作表示按照 ITEM_ID 字段的值进行分组，
有相同的 ITEM_ID 值放到一起
    ORDER BY COUNT_1 DESC //按照统计结果全局降序排序
    LIMIT 100;    //限制数据 100 行
```



```
Time taken: 100.435 seconds
hive>
    > CREATE TABLE  EMPTION AS
    >      SELECT ITEM_ID,COUNT(1) COUNT_1
    >     FROM RESULT
    >     WHERE ATIION_TYPE = '2'
    >     GROUP BY ITEM_ID
    >     ORDER BY COUNT_1 DESC
    >     LIMIT 100
    > ;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = root_20180928194638_f05a1053-2ee7-4a5c-8d37-82a7eb5c6ca0
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1538122414032_0008, Tracking URL = http://master:18088/proxy/application_1538122414032_0008/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0008
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 3
2018-09-28 19:46:53,467 Stage-1 map = 0%,  reduce = 0%
2018-09-28 19:47:04,223 Stage-1 map = 19%,  reduce = 0%, Cumulative CPU 8.02 sec
2018-09-28 19:47:05,280 Stage-1 map = 27%,  reduce = 0%, Cumulative CPU 15.61 sec
2018-09-28 19:47:08,456 Stage-1 map = 52%,  reduce = 0%, Cumulative CPU 22.14 sec
2018-09-28 19:47:09,508 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 27.83 sec
2018-09-28 19:47:19,099 Stage-1 map = 100%,  reduce = 33%, Cumulative CPU 33.56 sec
2018-09-28 19:47:20,173 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 38.24 sec
2018-09-28 19:47:21,223 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 42.38 sec
MapReduce Total cumulative CPU time: 42 seconds 380 msec
Ended Job = job_1538122414032_0008
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1538122414032_0009, Tracking URL = http://master:18088/proxy/application_1538122414032_0009/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0009
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-09-28 19:47:35,663 Stage-2 map = 0%,  reduce = 0%
2018-09-28 19:47:42,969 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 4.76 sec
2018-09-28 19:47:50,384 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 7.34 sec
MapReduce Total cumulative CPU time: 7 seconds 340 msec
Ended Job = job_1538122414032_0009
Moving data to directory hdfs://master:9000/user/hive_remote/warehouse/hongya.db/emption
[Warning] could not update stats.
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 3   Cumulative CPU: 42.38 sec   HDFS Read: 515782542 HDFS Write: 6516640 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 7.34 sec   HDFS Read: 6522299 HDFS Write: 1184 SUCCESS
Total MapReduce CPU Time Spent: 49 seconds 720 msec
OK
Time taken: 98.051 seconds
```

查看表 EMPTION 中所有数据

```
select * from EMPTION;//  查看表 EMPTION 中所有数据
```



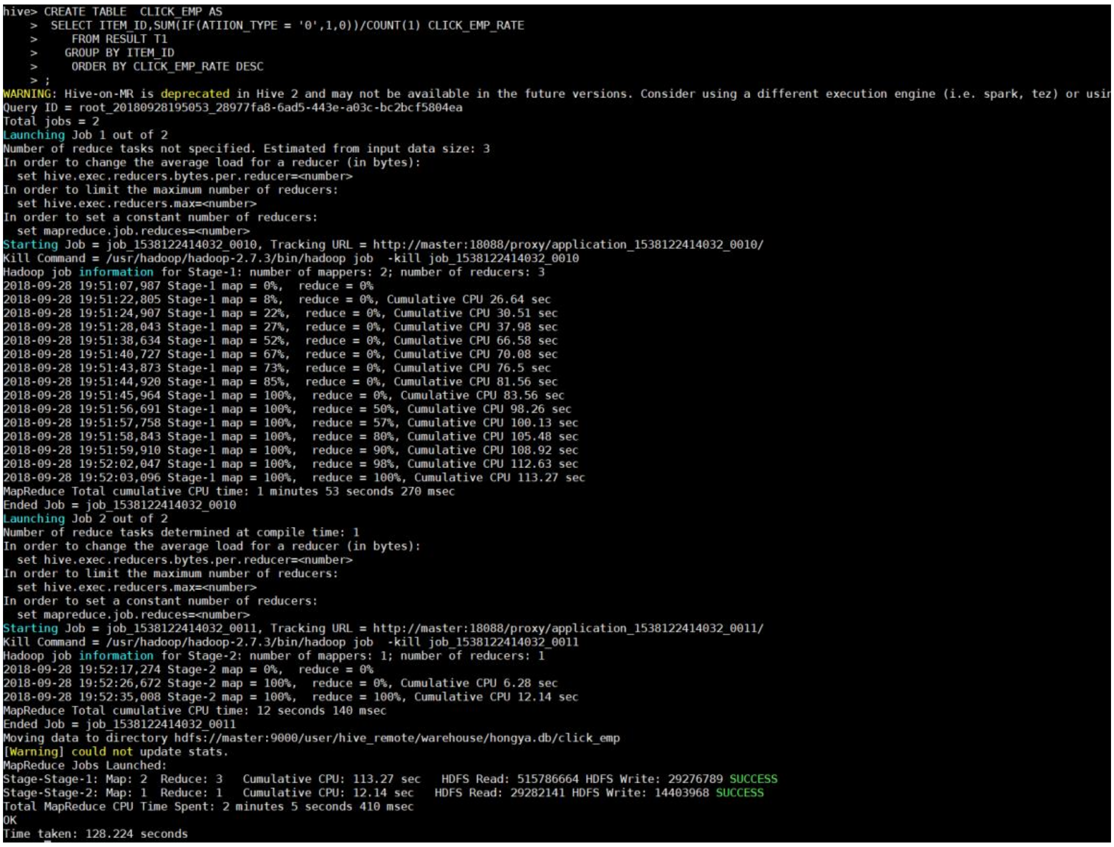
```
hive>
    > select * from EMPTION;
OK
631714  2449
15207   1971
822352  1414
1059899 1353
441588  1302
159310  1228
353560  1209
186456  1208
195714  1184
191499  1116
1039919 1114
951042  1055
668220  1025
698879  1010
417065  983
179830  972
853901  942
764906  939
806876  923
221663  875
1041507 875
1078471 859
873898  859
1073970 849
713695  847
525842  847
107407  836
173776  807
796486  805
655904  799
129323  795
1029992 795
843827  772
```

（三）转化率计算

（1）计算转化率

创建 click_emp，写入商品点击购买转化率

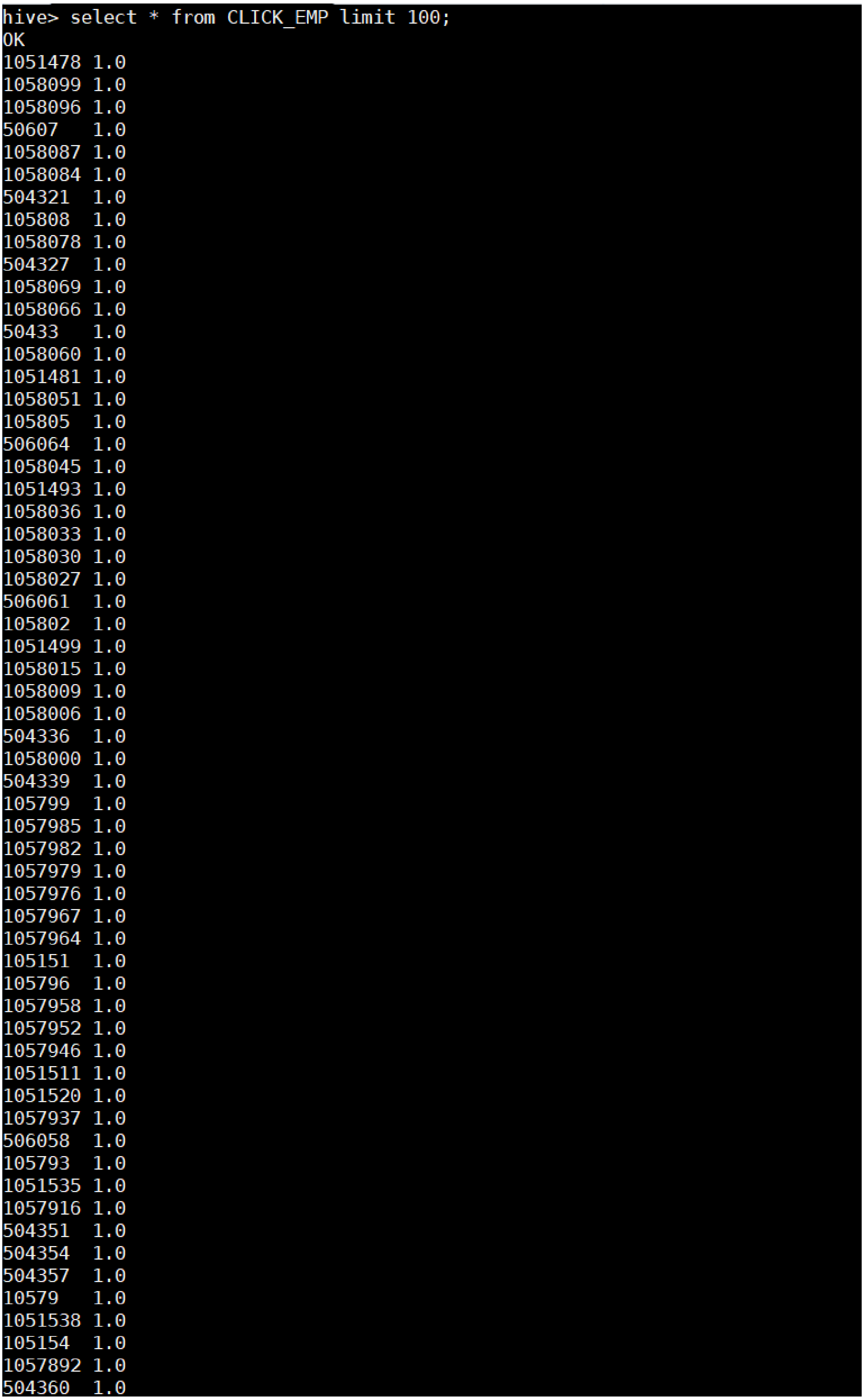
```
CREATE TABLE   CLICK_EMP AS //创建 CLICK_EMP 表

 SELECT ITEM_ID,SUM(IF(ATIION_TYPE = '0',1,0))/COUNT(1) CLICK_EMP_RATE//

点击总和除以该 ITEM_ID 的购买总和

      FROM RESULT T1

    GROUP BY ITEM_ID // group by 操作表示按照 ITEM_ID 字段的值进行分组，

有相同的 ITEM_ID 值放到一起

ORDER BY CLICK_EMP_RATE DESC;//按照点击购买转化率降序排序
```



```
hive> CREATE TABLE  CLICK_EMP AS
    >  SELECT ITEM_ID,SUM(IF(ATIION_TYPE = '0',1,0))/COUNT(1) CLICK_EMP_RATE
    >     FROM RESULT T1
    >    GROUP BY ITEM_ID
    >     ORDER BY CLICK_EMP_RATE DESC
    > ;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or usin
Query ID = root_20180928195053_28977fa8-6ad5-443e-a03c-bc2bcf5804ea
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1538122414032_0010, Tracking URL = http://master:18088/proxy/application_1538122414032_0010/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0010
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 3
2018-09-28 19:51:07,987 Stage-1 map = 0%,  reduce = 0%
2018-09-28 19:51:22,805 Stage-1 map = 8%,  reduce = 0%, Cumulative CPU 26.64 sec
2018-09-28 19:51:24,907 Stage-1 map = 22%,  reduce = 0%, Cumulative CPU 30.51 sec
2018-09-28 19:51:28,043 Stage-1 map = 27%,  reduce = 0%, Cumulative CPU 37.98 sec
2018-09-28 19:51:38,634 Stage-1 map = 52%,  reduce = 0%, Cumulative CPU 66.58 sec
2018-09-28 19:51:40,727 Stage-1 map = 67%,  reduce = 0%, Cumulative CPU 70.08 sec
2018-09-28 19:51:43,873 Stage-1 map = 73%,  reduce = 0%, Cumulative CPU 76.5 sec
2018-09-28 19:51:44,920 Stage-1 map = 85%,  reduce = 0%, Cumulative CPU 81.56 sec
2018-09-28 19:51:45,964 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 83.56 sec
2018-09-28 19:51:56,691 Stage-1 map = 100%,  reduce = 50%, Cumulative CPU 98.26 sec
2018-09-28 19:51:57,758 Stage-1 map = 100%,  reduce = 57%, Cumulative CPU 100.13 sec
2018-09-28 19:51:58,843 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 105.48 sec
2018-09-28 19:51:59,910 Stage-1 map = 100%,  reduce = 90%, Cumulative CPU 108.92 sec
2018-09-28 19:52:02,047 Stage-1 map = 100%,  reduce = 98%, Cumulative CPU 112.63 sec
2018-09-28 19:52:03,096 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 113.27 sec
MapReduce Total cumulative CPU time: 1 minutes 53 seconds 270 msec
Ended Job = job_1538122414032_0010
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1538122414032_0011, Tracking URL = http://master:18088/proxy/application_1538122414032_0011/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0011
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-09-28 19:52:17,274 Stage-2 map = 0%,  reduce = 0%
2018-09-28 19:52:26,672 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 6.28 sec
2018-09-28 19:52:35,008 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 12.14 sec
MapReduce Total cumulative CPU time: 12 seconds 140 msec
Ended Job = job_1538122414032_0011
Moving data to directory hdfs://master:9000/user/hive_remote/warehouse/hongya.db/click_emp
[Warning] could not update stats.
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 3   Cumulative CPU: 113.27 sec   HDFS Read: 515786664 HDFS Write: 29276789 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 12.14 sec   HDFS Read: 29282141 HDFS Write: 14403968 SUCCESS
Total MapReduce CPU Time Spent: 2 minutes 5 seconds 410 msec
OK
Time taken: 128.224 seconds
```

查看表 CLICK_EMP 中前 100 行数据

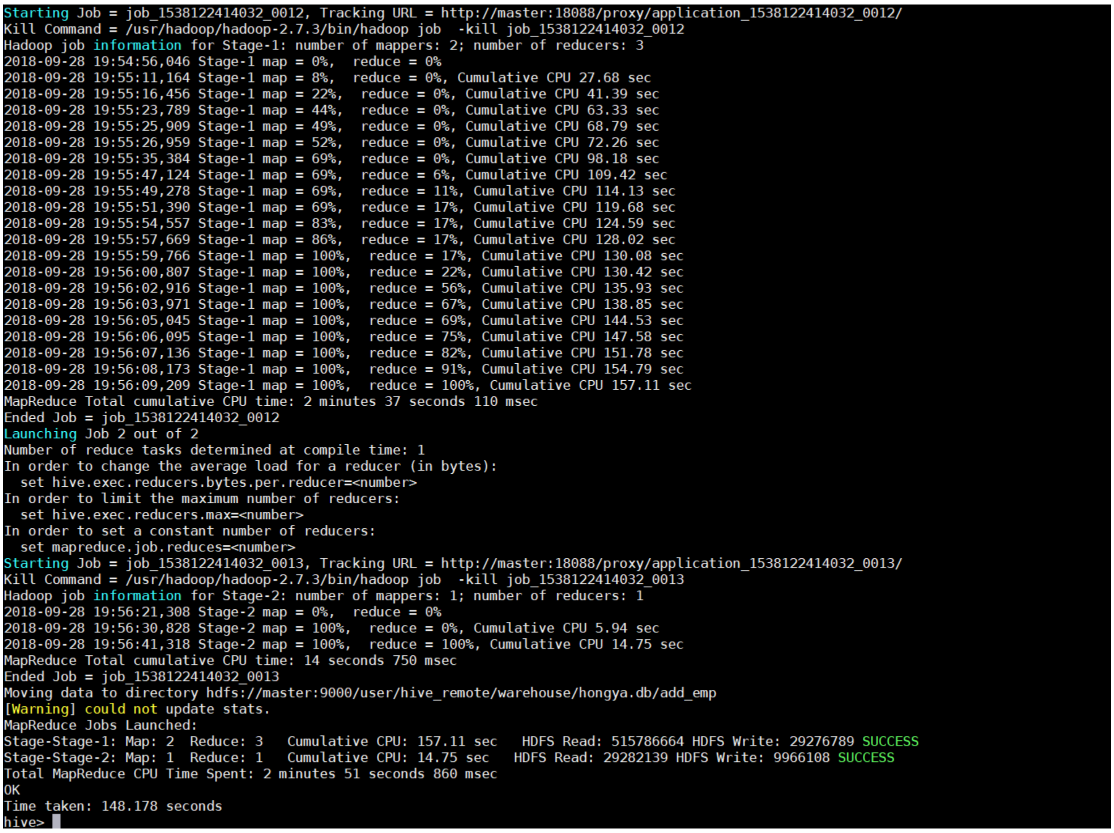
```
select * from CLICK_EMP limit 100;// 查看表 CLICK_EMP 中前 100 行数据

，第一列数据为 item_id，第二列数据为点击购买转化率
```



```
hive> select * from CLICK_EMP limit 100;
OK
1051478 1.0
1058099 1.0
1058096 1.0
50607   1.0
1058087 1.0
1058084 1.0
504321  1.0
105808  1.0
1058078 1.0
504327  1.0
1058069 1.0
1058066 1.0
50433   1.0
1058060 1.0
1051481 1.0
1058051 1.0
105805  1.0
506064  1.0
1058045 1.0
1051493 1.0
1058036 1.0
1058033 1.0
1058030 1.0
1058027 1.0
506061  1.0
105802  1.0
1051499 1.0
1058015 1.0
1058009 1.0
1058006 1.0
504336  1.0
1058000 1.0
504339  1.0
105799  1.0
1057985 1.0
1057982 1.0
1057979 1.0
1057976 1.0
1057967 1.0
1057964 1.0
105151  1.0
105796  1.0
1057958 1.0
1057952 1.0
1057946 1.0
1051511 1.0
1051520 1.0
1057937 1.0
506058  1.0
105793  1.0
1051535 1.0
1057916 1.0
504351  1.0
504354  1.0
504357  1.0
10579   1.0
1051538 1.0
105154  1.0
1057892 1.0
504360  1.0
```

创建表 add_emp，写入商品加入购物车-购买转化率

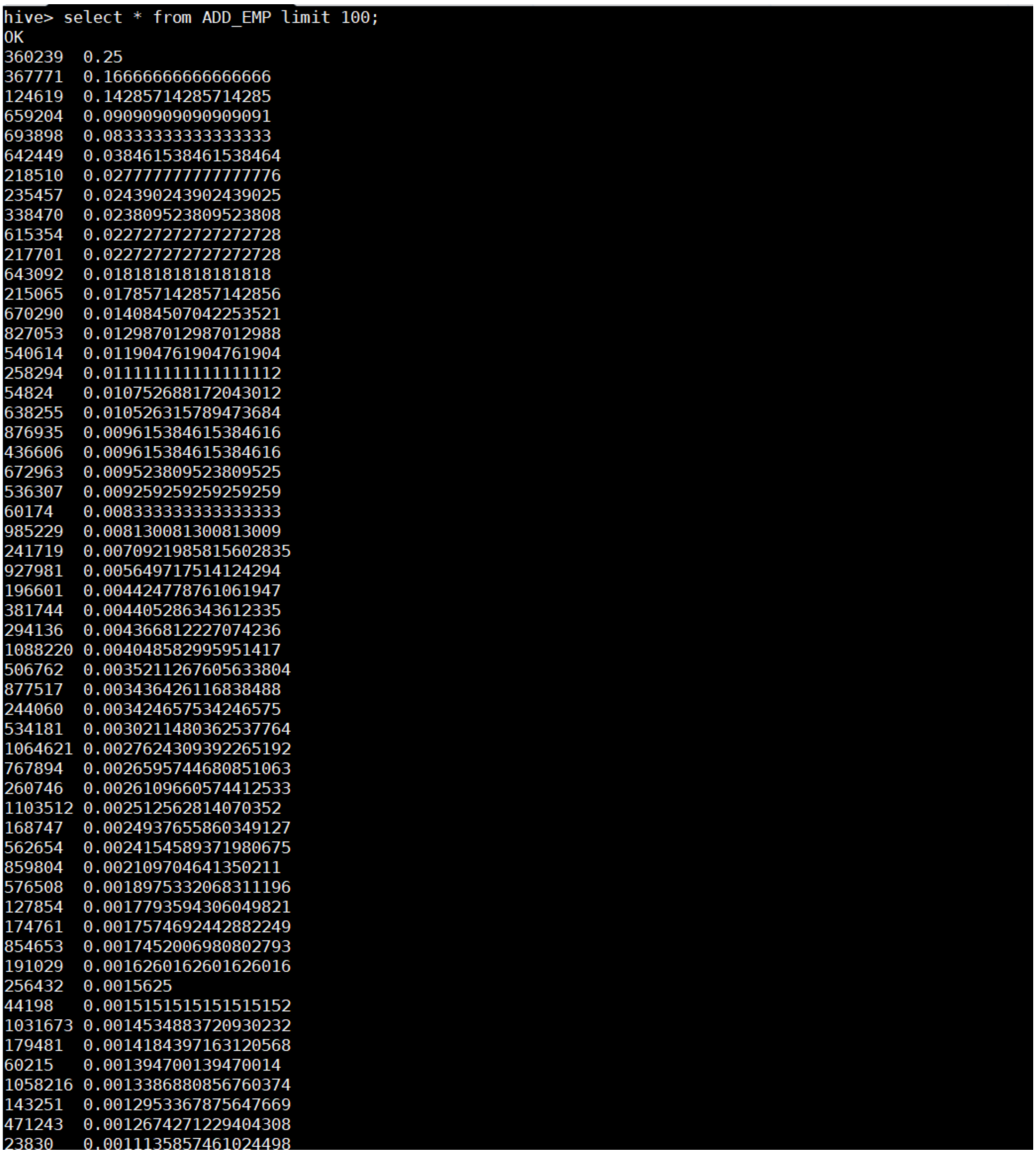
```
CREATE TABLE  ADD_EMP AS //创建 ADD_EMP P 表
    SELECT    ITEM_ID,SUM(IF(ATIION_TYPE    =    '1',1,0))/COUNT(1)
CLICK_EMP_RATE //加入购物车总和除以该 ITEM_ID 的购买总和
   FROM RESULT T1
   GROUP BY ITEM_ID // group by 操作表示按照 ITEM_ID 字段的值进行分组,
有相同的 ITEM_ID 值放到一起
   ORDER BY CLICK_EMP_RATE DESC; //按照点击购买转化率降序排序
```



```
Starting Job = job_1538122414032_0012, Tracking URL = http://master:18088/proxy/application_1538122414032_0012/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0012
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 3
2018-09-28 19:54:56,046 Stage-1 map = 0%,  reduce = 0%
2018-09-28 19:55:11,164 Stage-1 map = 8%,  reduce = 0%, Cumulative CPU 27.68 sec
2018-09-28 19:55:16,456 Stage-1 map = 22%,  reduce = 0%, Cumulative CPU 41.39 sec
2018-09-28 19:55:23,789 Stage-1 map = 44%,  reduce = 0%, Cumulative CPU 63.33 sec
2018-09-28 19:55:25,909 Stage-1 map = 49%,  reduce = 0%, Cumulative CPU 68.79 sec
2018-09-28 19:55:26,959 Stage-1 map = 52%,  reduce = 0%, Cumulative CPU 72.26 sec
2018-09-28 19:55:35,384 Stage-1 map = 69%,  reduce = 0%, Cumulative CPU 98.18 sec
2018-09-28 19:55:47,124 Stage-1 map = 69%,  reduce = 6%, Cumulative CPU 109.42 sec
2018-09-28 19:55:49,278 Stage-1 map = 69%,  reduce = 11%, Cumulative CPU 114.13 sec
2018-09-28 19:55:51,390 Stage-1 map = 69%,  reduce = 17%, Cumulative CPU 119.68 sec
2018-09-28 19:55:54,557 Stage-1 map = 83%,  reduce = 17%, Cumulative CPU 124.59 sec
2018-09-28 19:55:57,669 Stage-1 map = 86%,  reduce = 17%, Cumulative CPU 128.02 sec
2018-09-28 19:55:59,766 Stage-1 map = 100%,  reduce = 17%, Cumulative CPU 130.08 sec
2018-09-28 19:56:00,807 Stage-1 map = 100%,  reduce = 22%, Cumulative CPU 130.42 sec
2018-09-28 19:56:02,916 Stage-1 map = 100%,  reduce = 56%, Cumulative CPU 135.93 sec
2018-09-28 19:56:03,971 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 138.85 sec
2018-09-28 19:56:05,045 Stage-1 map = 100%,  reduce = 69%, Cumulative CPU 144.53 sec
2018-09-28 19:56:06,095 Stage-1 map = 100%,  reduce = 75%, Cumulative CPU 147.58 sec
2018-09-28 19:56:07,136 Stage-1 map = 100%,  reduce = 82%, Cumulative CPU 151.78 sec
2018-09-28 19:56:08,173 Stage-1 map = 100%,  reduce = 91%, Cumulative CPU 154.79 sec
2018-09-28 19:56:09,209 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 157.11 sec
MapReduce Total cumulative CPU time: 2 minutes 37 seconds 110 msec
Ended Job = job_1538122414032_0012
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1538122414032_0013, Tracking URL = http://master:18088/proxy/application_1538122414032_0013/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0013
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-09-28 19:56:21,308 Stage-2 map = 0%,  reduce = 0%
2018-09-28 19:56:30,828 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 5.94 sec
2018-09-28 19:56:41,318 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 14.75 sec
MapReduce Total cumulative CPU time: 14 seconds 750 msec
Ended Job = job_1538122414032_0013
Moving data to directory hdfs://master:9000/user/hive_remote/warehouse/hongya.db/add_emp
[Warning] could not update stats.
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 3   Cumulative CPU: 157.11 sec   HDFS Read: 515786664 HDFS Write: 29276789 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 14.75 sec   HDFS Read: 29282139 HDFS Write: 9966108 SUCCESS
Total MapReduce CPU Time Spent: 2 minutes 51 seconds 860 msec
OK
Time taken: 148.178 seconds
hive> 
```

查看表 ADD_EMP 中前 100 行数据

```
select * from ADD_EMP limit 100;//  查看表 ADD_EMP 中前 100 行数据
，第一列数据为 item_id，第二列数据为加入购物车购买转化率
```

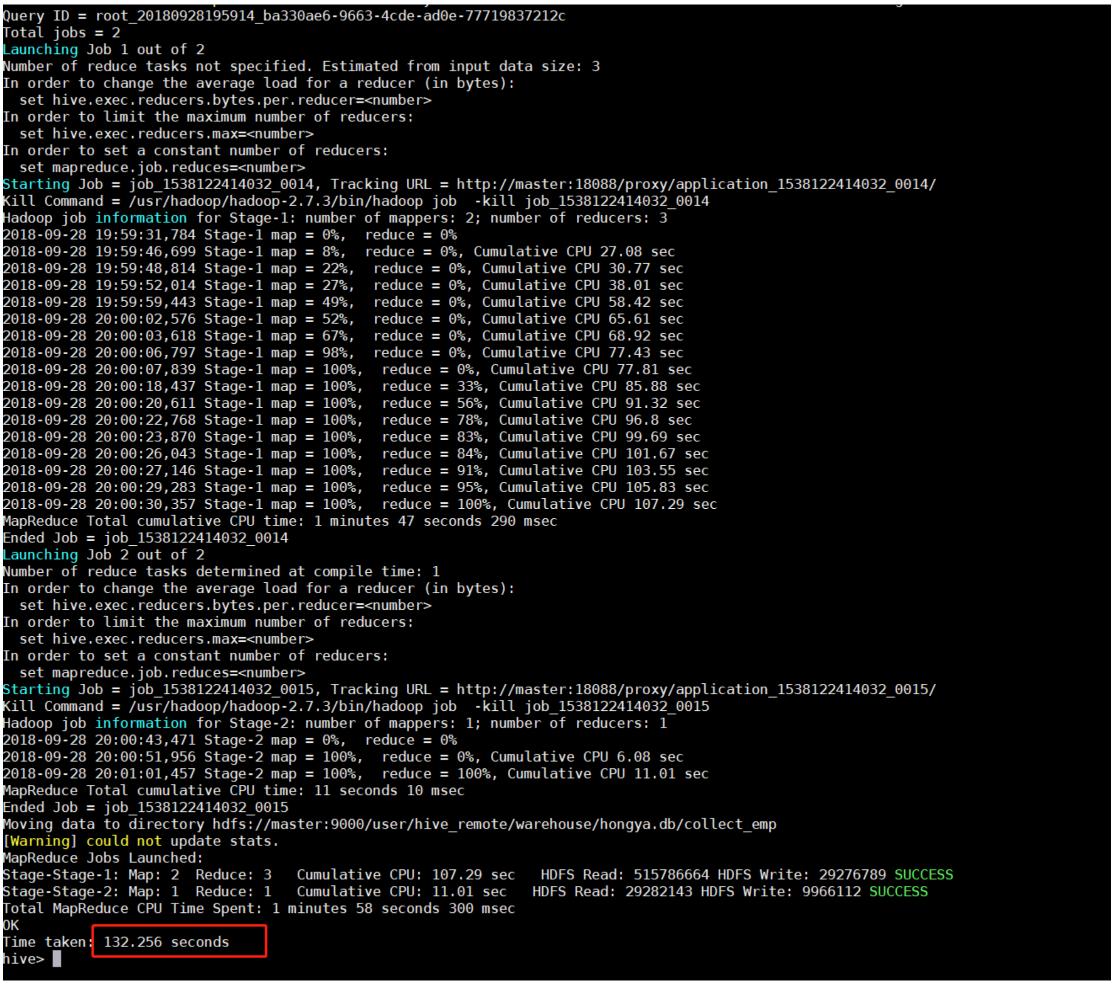


```
hive> select * from ADD_EMP limit 100;
OK
360239  0.25
367771  0.16666666666666666
124619  0.14285714285714285
659204  0.09090909090909091
693898  0.08333333333333333
642449  0.038461538461538464
218510  0.027777777777777776
235457  0.024390243902439025
338470  0.023809523809523808
615354  0.022727272727272728
217701  0.022727272727272728
643092  0.01818181818181818
215065  0.017857142857142856
670290  0.014084507042253521
827053  0.012987012987012988
540614  0.011904761904761904
258294  0.011111111111111112
54824   0.010752688172043012
638255  0.010526315789473684
876935  0.009615384615384616
436606  0.009615384615384616
672963  0.009523809523809525
536307  0.009259259259259259
60174   0.008333333333333333
985229  0.008130081300813009
241719  0.0070921985815602835
927981  0.005649717514124294
196601  0.004424778761061947
381744  0.004405286343612335
294136  0.004366812227074236
1088220 0.004048582995951417
506762  0.0035211267605633804
877517  0.003436426116838488
244060  0.003424657534246575
534181  0.0030211480362537764
1064621 0.0027624309392265192
767894  0.0026595744680851063
260746  0.0026109660574412533
1103512 0.002512562814070352
168747  0.0024937655860349127
562654  0.0024154589371980675
859804  0.002109704641350211
576508  0.0018975332068311196
127854  0.0017793594306049821
174761  0.0017574692442882249
854653  0.0017452006980802793
191029  0.0016260162601626016
256432  0.0015625
44198   0.0015151515151515152
1031673 0.0014534883720930232
179481  0.0014184397163120568
60215   0.001394700139470014
1058216 0.0013386880856760374
143251  0.0012953367875647669
471243  0.0012674271229404308
23830   0.0011135857461024498
```

创建表 collect_emp，写入商品收藏-购买转化率

```
CREATE TABLE  COLLECT_EMP AS //创建 COLLECT_EMP 表
    SELECT   ITEM_ID,SUM(IF(ATIION_TYPE   =   '1',1,0))/COUNT(1)
CLICK_EMP_RATE //收藏总和除以该 ITEM_ID 的购买总和
    FROM RESULT T1
    GROUP BY ITEM_ID // group by 操作表示按照 ITEM_ID 字段的值进行分组,
有相同的 ITEM_ID 值放到一起
```

```
ORDER BY CLICK_EMP_RATE DESC; //按照点击购买转化率降序排序
```

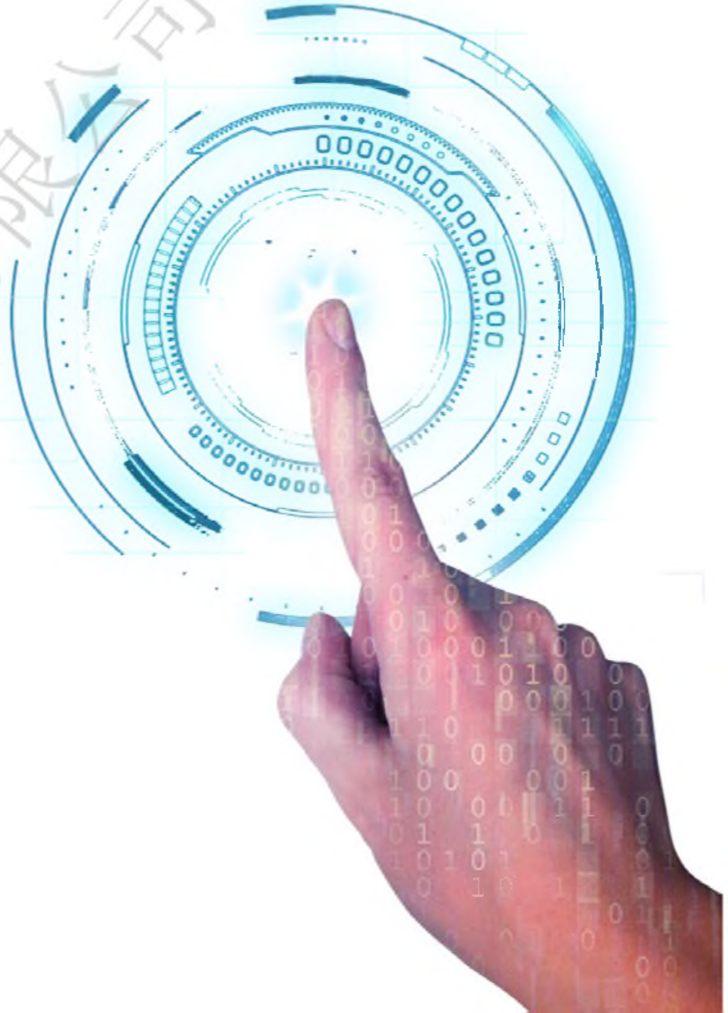


```
Query ID = root_20180928195914_ba330ae6-9663-4cde-ad0e-77719837212c
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 3
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1538122414032_0014, Tracking URL = http://master:18088/proxy/application_1538122414032_0014/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0014
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 3
2018-09-28 19:59:31,784 Stage-1 map = 0%,  reduce = 0%
2018-09-28 19:59:46,699 Stage-1 map = 8%,  reduce = 0%, Cumulative CPU 27.08 sec
2018-09-28 19:59:48,814 Stage-1 map = 22%,  reduce = 0%, Cumulative CPU 30.77 sec
2018-09-28 19:59:52,014 Stage-1 map = 27%,  reduce = 0%, Cumulative CPU 38.01 sec
2018-09-28 19:59:59,443 Stage-1 map = 49%,  reduce = 0%, Cumulative CPU 58.42 sec
2018-09-28 20:00:02,576 Stage-1 map = 52%,  reduce = 0%, Cumulative CPU 65.61 sec
2018-09-28 20:00:03,618 Stage-1 map = 67%,  reduce = 0%, Cumulative CPU 68.92 sec
2018-09-28 20:00:06,797 Stage-1 map = 98%,  reduce = 0%, Cumulative CPU 77.43 sec
2018-09-28 20:00:07,839 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 77.81 sec
2018-09-28 20:00:18,437 Stage-1 map = 100%,  reduce = 33%, Cumulative CPU 85.88 sec
2018-09-28 20:00:20,611 Stage-1 map = 100%,  reduce = 56%, Cumulative CPU 91.32 sec
2018-09-28 20:00:22,768 Stage-1 map = 100%,  reduce = 78%, Cumulative CPU 96.8 sec
2018-09-28 20:00:23,870 Stage-1 map = 100%,  reduce = 83%, Cumulative CPU 99.69 sec
2018-09-28 20:00:26,043 Stage-1 map = 100%,  reduce = 84%, Cumulative CPU 101.67 sec
2018-09-28 20:00:27,146 Stage-1 map = 100%,  reduce = 91%, Cumulative CPU 103.55 sec
2018-09-28 20:00:29,283 Stage-1 map = 100%,  reduce = 95%, Cumulative CPU 105.83 sec
2018-09-28 20:00:30,357 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 107.29 sec
MapReduce Total cumulative CPU time: 1 minutes 47 seconds 290 msec
Ended Job = job_1538122414032_0014
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1538122414032_0015, Tracking URL = http://master:18088/proxy/application_1538122414032_0015/
Kill Command = /usr/hadoop/hadoop-2.7.3/bin/hadoop job  -kill job_1538122414032_0015
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-09-28 20:00:43,471 Stage-2 map = 0%,  reduce = 0%
2018-09-28 20:00:51,956 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 6.08 sec
2018-09-28 20:01:01,457 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 11.01 sec
MapReduce Total cumulative CPU time: 11 seconds 10 msec
Ended Job = job_1538122414032_0015
Moving data to directory hdfs://master:9000/user/hive_remote/warehouse/hongya.db/collect_emp
[Warning] could not update stats.
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 3   Cumulative CPU: 107.29 sec   HDFS Read: 515786664 HDFS Write: 29276789 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 11.01 sec   HDFS Read: 29282143 HDFS Write: 9966112 SUCCESS
Total MapReduce CPU Time Spent: 1 minutes 58 seconds 300 msec
OK
Time taken: 132.256 seconds
hive> 
```

查看表 COLLECT_EMP 100 行数据，第一列数据为 item_id，第二列数据为收藏购买转化率

```
select * from COLLECT_EMP limit 100;
```

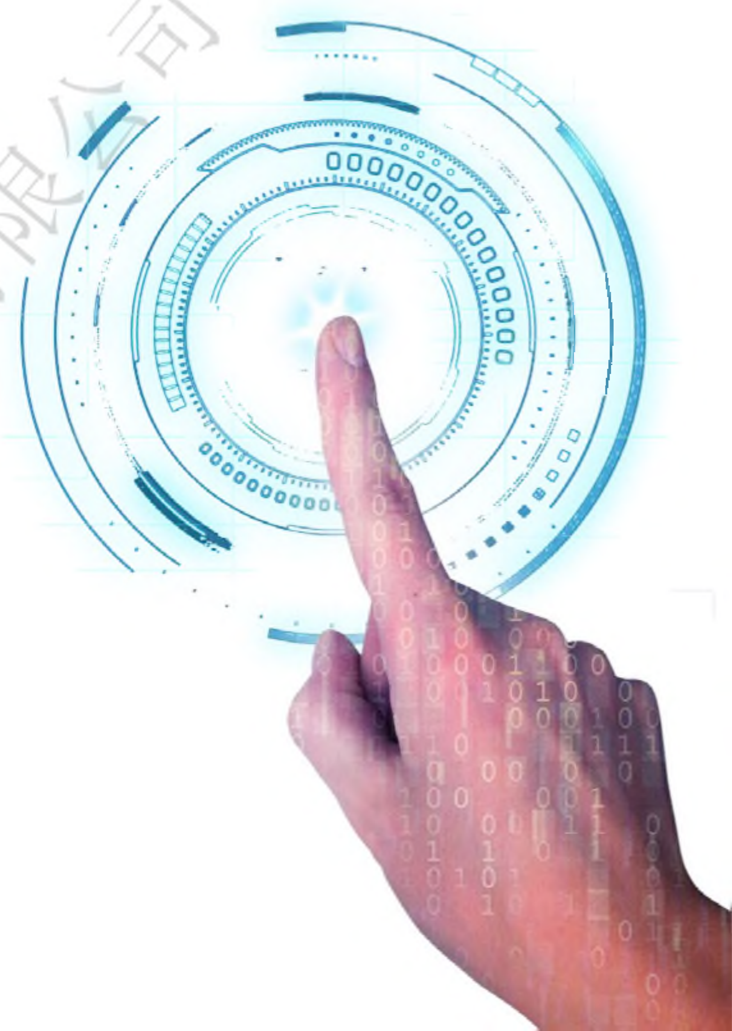
Spark on YARN安装



任务要求

1. 了解Scala的安装过程
2. 掌握Spark on YARN的安装过程
3. 掌握Spark-shell的使用

01 安装Scala

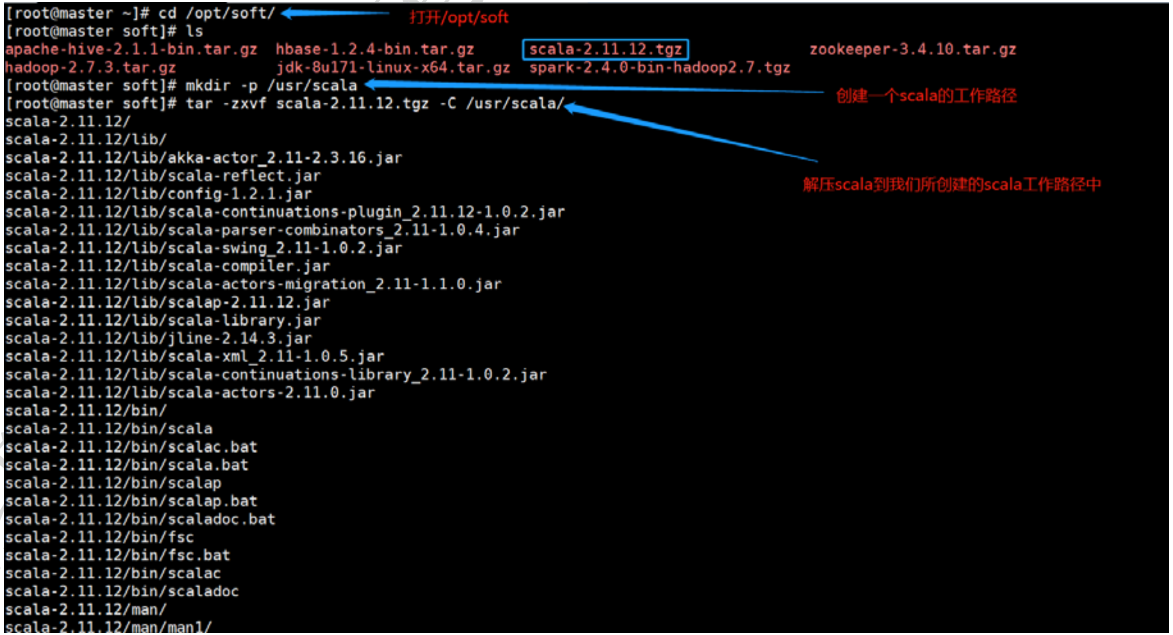


安装Scala

我们需要在拥有hadoop集群的所有节点中安装scala语言环境，因为spark的源代码为scala语言所编写，所以接下来我们进行安装scala。

解压scala的tar包：

首先我们进入到本系统的`/opt/soft`路径下可以看到我们所提供的scala安装包，接下来我们在`/usr/`下创建scala文件夹，然后解压scala到我们所创建的scala工作路径中，具体 操作如下图所示：



```
[root@master ~]# cd /opt/soft/
[root@master soft]# ls
apache-hive-2.1.1-bin.tar.gz  hbase-1.2.4-bin.tar.gz        scala-2.11.12.tgz                  zookeeper-3.4.10.tar.gz
hadoop-2.7.3.tar.gz           jdk-8u171-linux-x64.tar.gz    spark-2.4.0-bin-hadoop2.7.tgz
[root@master soft]# mkdir -p /usr/scala
[root@master soft]# tar -zxvf scala-2.11.12.tgz -C /usr/scala/
scala-2.11.12/
scala-2.11.12/lib/
scala-2.11.12/lib/akka-actor_2.11-2.3.16.jar
scala-2.11.12/lib/scala-reflect.jar
scala-2.11.12/lib/config-1.2.1.jar
scala-2.11.12/lib/scala-continuations-plugin_2.11.12-1.0.2.jar
scala-2.11.12/lib/scala-parser-combinators_2.11-1.0.4.jar
scala-2.11.12/lib/scala-swing_2.11-1.0.2.jar
scala-2.11.12/lib/scala-compiler.jar
scala-2.11.12/lib/scala-actors-migration_2.11-1.1.0.jar
scala-2.11.12/lib/scalap-2.11.12.jar
scala-2.11.12/lib/scala-library.jar
scala-2.11.12/lib/jline-2.14.3.jar
scala-2.11.12/lib/scala-xml_2.11-1.0.5.jar
scala-2.11.12/lib/scala-continuations-library_2.11-1.0.2.jar
scala-2.11.12/lib/scala-actors-2.11.0.jar
scala-2.11.12/bin/
scala-2.11.12/bin/scala
scala-2.11.12/bin/scalac.bat
scala-2.11.12/bin/scala.bat
scala-2.11.12/bin/scalap
scala-2.11.12/bin/scalap.bat
scala-2.11.12/bin/scaladoc.bat
scala-2.11.12/bin/fsc
scala-2.11.12/bin/fsc.bat
scala-2.11.12/bin/scalac
scala-2.11.12/bin/scaladoc
scala-2.11.12/man/
scala-2.11.12/man/man1/
```

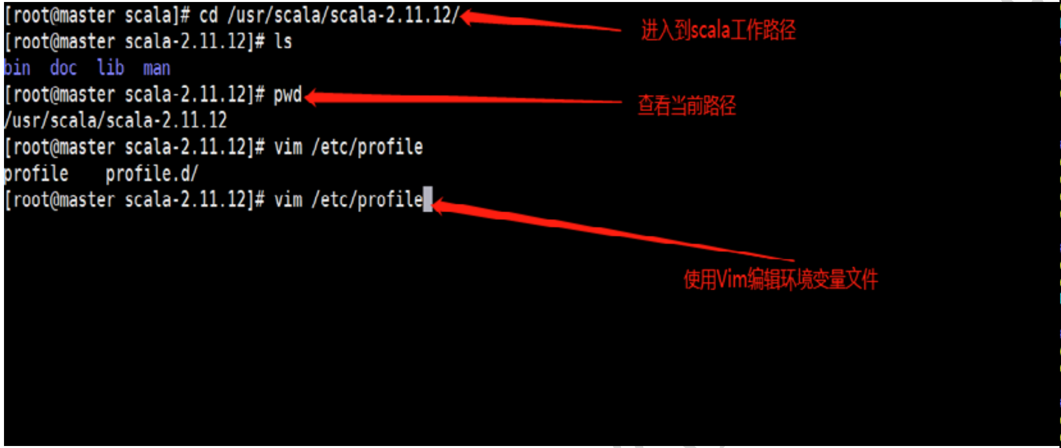
打开/opt/soft

创建一个scala的工作路径

解压scala到我们所创建的scala工作路径中

配置scala的环境变量：

当我们解压好scala安装包之后，我们需要对scala进行配置环境变量，我们需要将环境变量配置到`/etc/profile`文件中，首先我们进入scala的工作路径，然后使用`pwd`命令进行查看scala的安装路径，接下来就可以复制此路径到我们的profile文件中了，具体操作如下图所示：

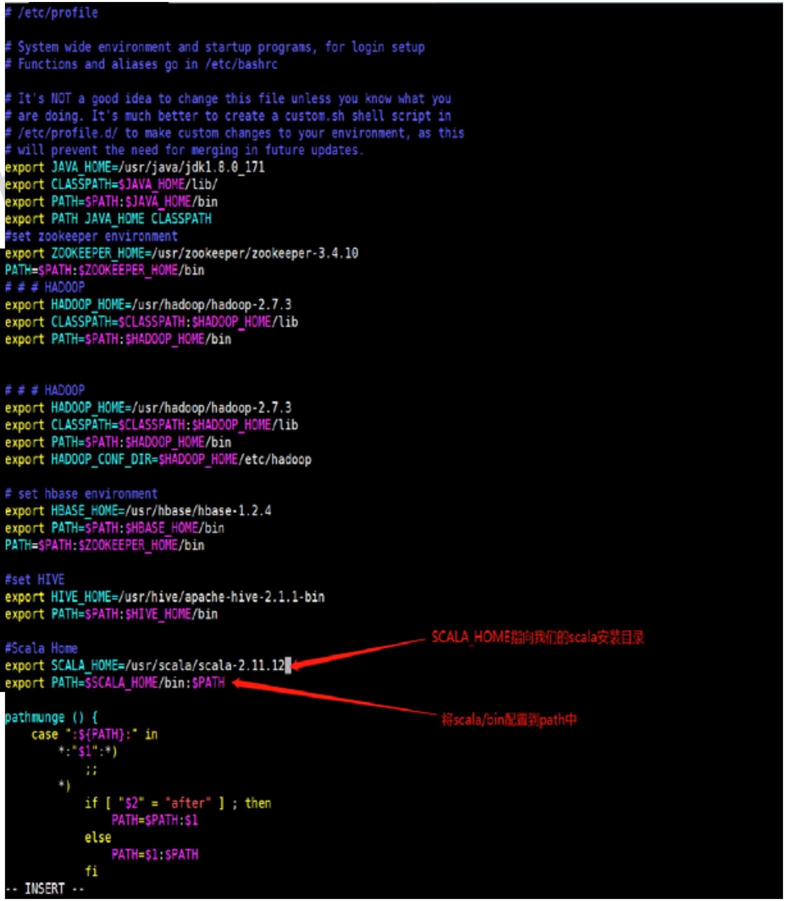


```
[root@master scala]# cd /usr/scala/scala-2.11.12/
[root@master scala-2.11.12]# ls
bin  doc  lib  man
[root@master scala-2.11.12]# pwd
/usr/scala/scala-2.11.12
[root@master scala-2.11.12]# vim /etc/profile
profile    profile.d/
[root@master scala-2.11.12]# vim /etc/profile
```

进入到scala工作路径

查看当前路径

使用Vim编辑环境变量文件



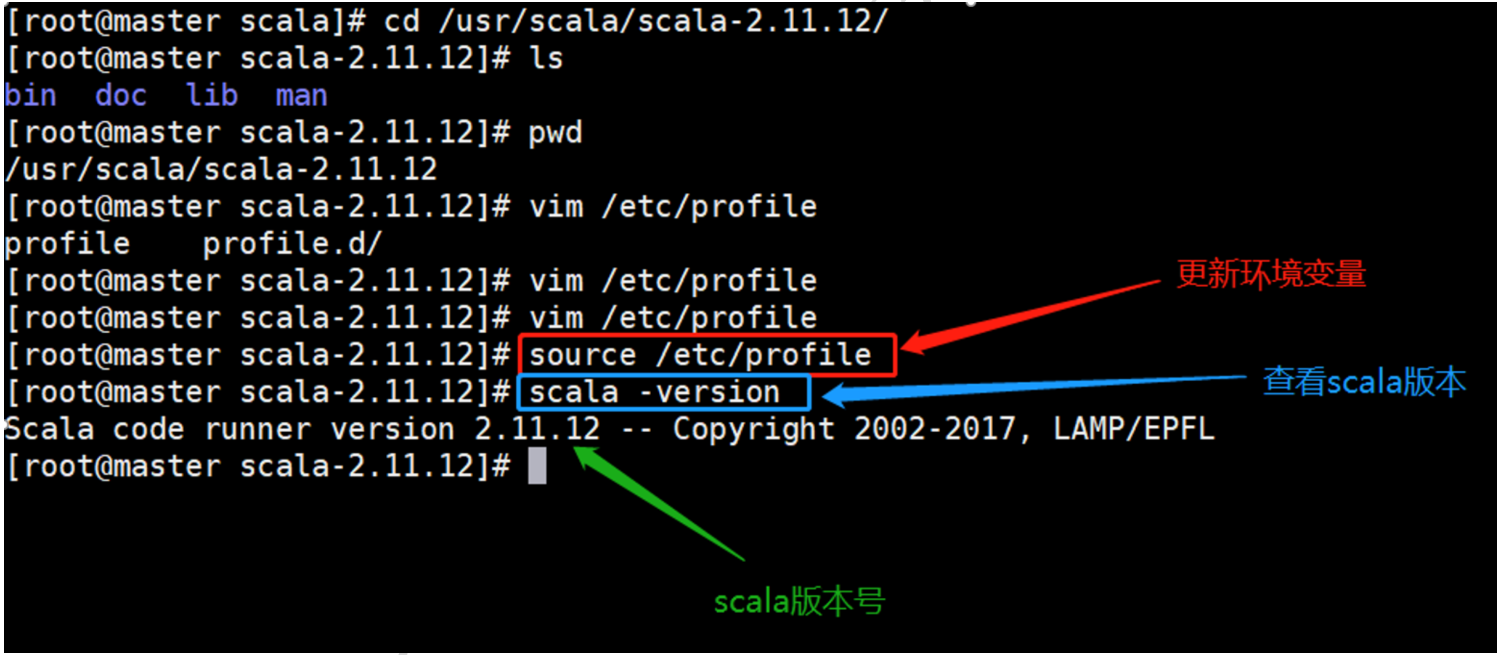
```
# /etc/profile

# System wide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc

# It's NOT a good idea to change this file unless you know what you
# are doing. It's much better to create a custom.sh shell script in
# /etc/profile.d/ to make custom changes to your environment, as this
# will prevent the need for merging in future updates.
export JAVA_HOME=/usr/java/jdk1.8.0_171
export CLASSPATH=$JAVA_HOME/lib/
export PATH=$PATH:$JAVA_HOME/bin
export PATH JAVA_HOME CLASSPATH
#set zookeeper environment
export ZOOKEEPER_HOME=/usr/zookeeper/zookeeper-3.4.10
PATH=$PATH:$ZOOKEEPER_HOME/bin
# # # HADOOP
export HADOOP_HOME=/usr/hadoop/hadoop-2.7.3
export CLASSPATH=$CLASSPATH:$HADOOP_HOME/lib
export PATH=$PATH:$HADOOP_HOME/bin


# # # HADOOP
export HADOOP_HOME=/usr/hadoop/hadoop-2.7.3
export CLASSPATH=$CLASSPATH:$HADOOP_HOME/lib
export PATH=$PATH:$HADOOP_HOME/bin
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop

# set hbase environment
export HBASE_HOME=/usr/hbase/hbase-1.2.4
export PATH=$PATH:$HBASE_HOME/bin
PATH=$PATH:$ZOOKEEPER_HOME/bin

#set HIVE
export HIVE_HOME=/usr/hive/apache-hive-2.1.1-bin
export PATH=$PATH:$HIVE_HOME/bin

#Scala Home
export SCALA_HOME=/usr/scala/scala-2.11.12
export PATH=$SCALA_HOME/bin:$PATH

pathmunge () {
    case ":${PATH}:" in
        *:"$1":*)
            ;;
        *)
            if [ "$2" = "after" ] ; then
                PATH=$PATH:$1
            else
                PATH=$1:$PATH
            fi
-- INSERT --
```

SCALA_HOME指向我们的scala安装目录

将scala/bin配置到path中

更新环境变量并查看版本号

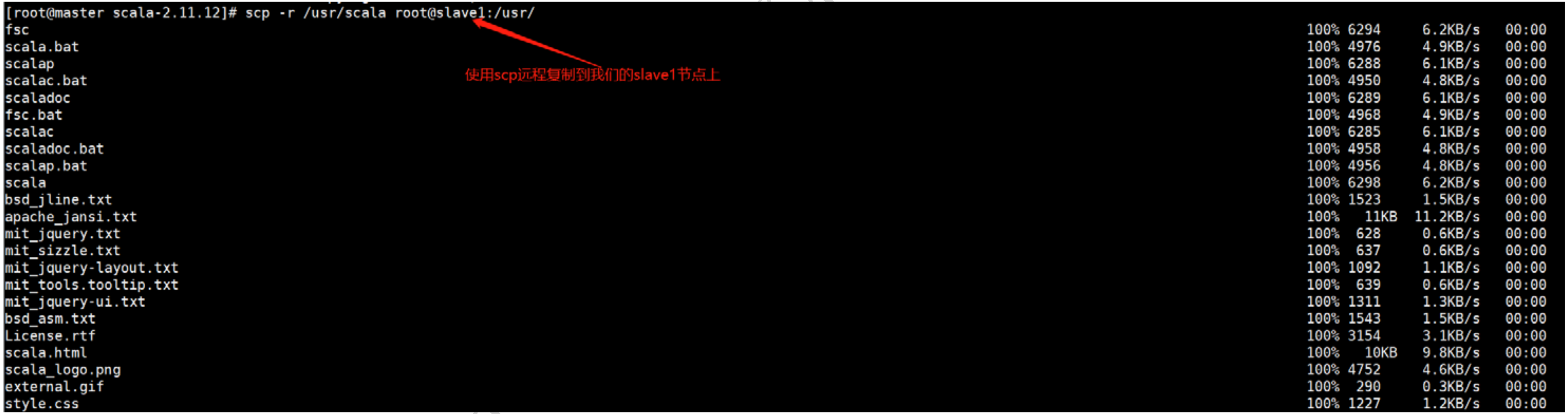
当我们配置好环境变量之后我们需要使用source 命令去更新我们的环境变量文件，最后我们使用`scala -version`查看我们的scala是否安装成功，具体操作如下图所示：



```
[root@master scala]# cd /usr/scala/scala-2.11.12/
[root@master scala-2.11.12]# ls
bin   doc   lib   man
[root@master scala-2.11.12]# pwd
/usr/scala/scala-2.11.12
[root@master scala-2.11.12]# vim /etc/profile
profile     profile.d/
[root@master scala-2.11.12]# vim /etc/profile
[root@master scala-2.11.12]# vim /etc/profile
[root@master scala-2.11.12]# source /etc/profile
[root@master scala-2.11.12]# scala -version
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
[root@master scala-2.11.12]# 
```

发送至所有子节点

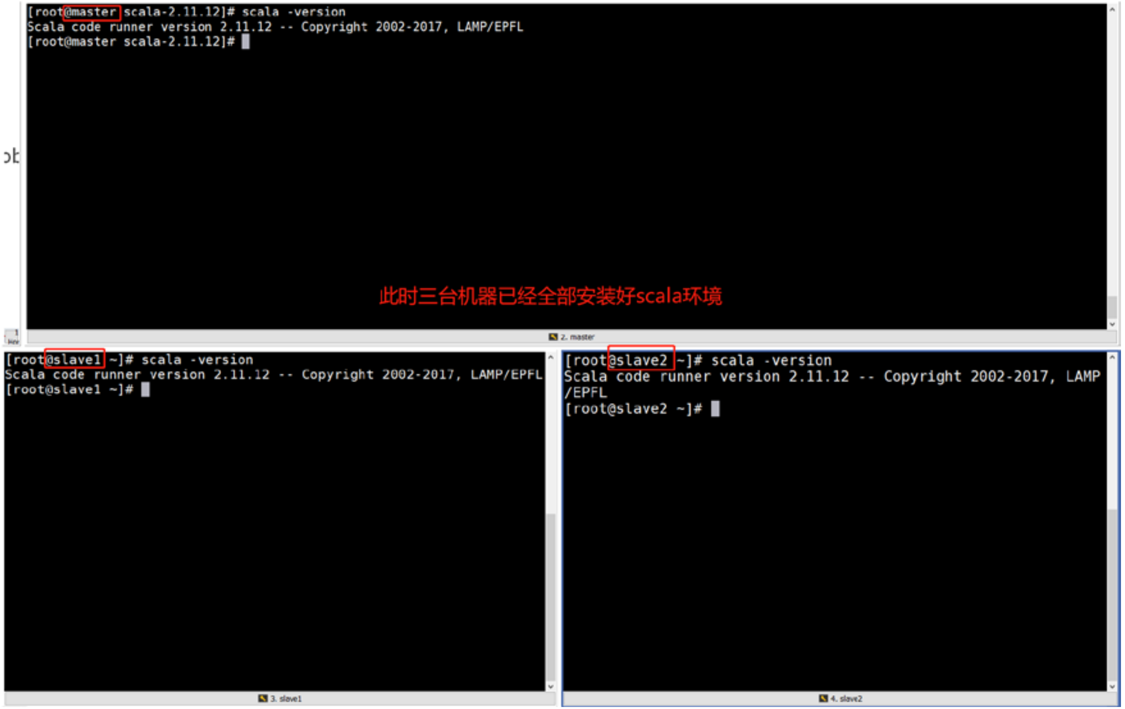
- **复制scala到子节点：**
  因为我们是集群环境，所以接下来我们需要将我们的scala环境发送到我们的其他子节点上，具体操作如下图所示：
- **命令：**`scp -r /usr/scala root@slave1:/usr/`
- **注：**图中只展示了复制到slave1中的操作，slave2的操作同理，请同学们自行操作。



```
[root@master scala-2.11.12]# scp -r /usr/scala root@slave1:/usr/
fsc                                                                                         100% 6294     6.2KB/s   00:00
scala.bat                                                                                   100% 4976     4.9KB/s   00:00
scalap                                                                                      100% 6288     6.1KB/s   00:00
scalac.bat                                                                                  100% 4950     4.8KB/s   00:00
scaladoc                                                                                    100% 6289     6.1KB/s   00:00
fsc.bat                                                                                     100% 4968     4.9KB/s   00:00
scalac                                                                                      100% 6285     6.1KB/s   00:00
scaladoc.bat                                                                                100% 4958     4.8KB/s   00:00
scalap.bat                                                                                  100% 4956     4.8KB/s   00:00
scala                                                                                       100% 6298     6.2KB/s   00:00
bsd_jline.txt                                                                               100% 1523     1.5KB/s   00:00
apache_jansi.txt                                                                            100%   11KB  11.2KB/s   00:00
mit_jquery.txt                                                                              100%  628     0.6KB/s   00:00
mit_sizzle.txt                                                                              100%  637     0.6KB/s   00:00
mit_jquery-layout.txt                                                                       100% 1092     1.1KB/s   00:00
mit_tools.tooltip.txt                                                                       100%  639     0.6KB/s   00:00
mit_jquery-ui.txt                                                                           100% 1311     1.3KB/s   00:00
bsd_asm.txt                                                                                 100% 1543     1.5KB/s   00:00
License.rtf                                                                                 100% 3154     3.1KB/s   00:00
scala.html                                                                                  100%   10KB   9.8KB/s   00:00
scala_logo.png                                                                              100% 4752     4.6KB/s   00:00
external.gif                                                                                100%  290     0.3KB/s   00:00
style.css                                                                                   100% 1227     1.2KB/s   00:00
```

所有节点安装成功

切换slave1和slave2节点去编写环境变量将scala环境变量填进去，然后更新环境变量，操作和master节点操作一样，这里就不赘述了。当环境变量配置成功后我们需要检测每个节点的scala环境是否安装成功，具体操作如下图所示：



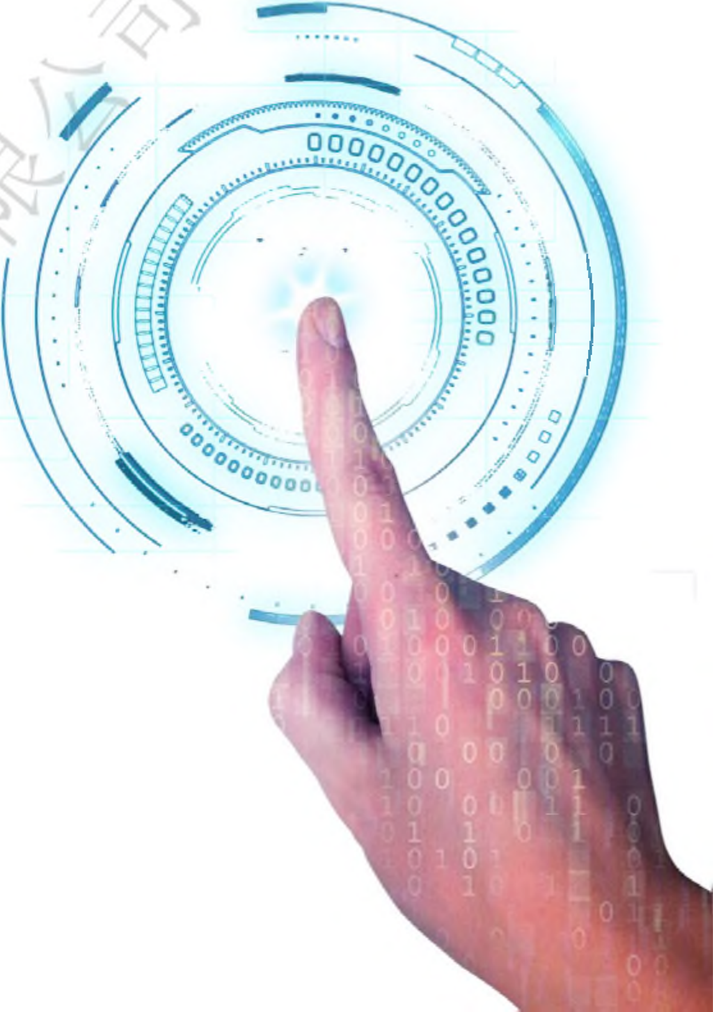
```
[root@master scala-2.11.12]# scala -version
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
[root@master scala-2.11.12]#
```

此时三台机器已经全部安装好scala环境

```
[root@slave1 ~]# scala -version
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
[root@slave1 ~]#
```

```
[root@slave2 ~]# scala -version
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP
/EPFL
[root@slave2 ~]#
```

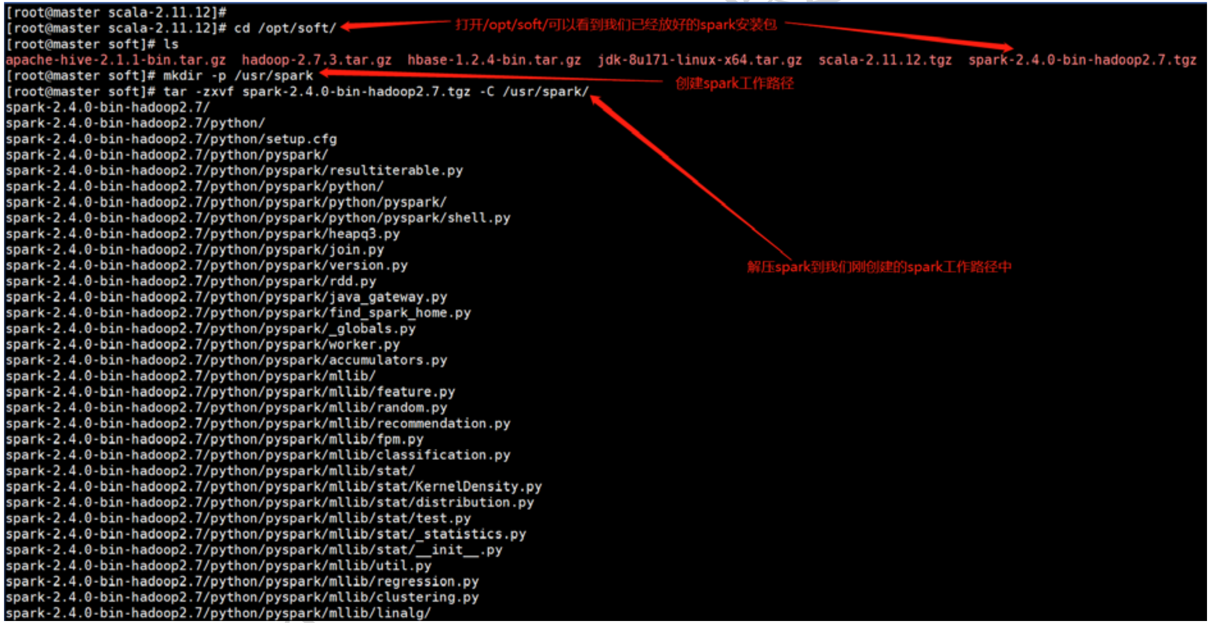
02 安装Spark



安装Spark

- 解压spark的tar包：

　　首先我们进入到本系统的`/opt/soft`路径下可以看到我们所提供的spark安装包，接下来我们在`/usr/`下创建spark文件夹，然后解压spark到我们所创建的spark工作路径中，具体 操作如下图所示：



```
[root@master scala-2.11.12]#
[root@master scala-2.11.12]# cd /opt/soft/
[root@master soft]# ls
apache-hive-2.1.1-bin.tar.gz  hadoop-2.7.3.tar.gz  hbase-1.2.4-bin.tar.gz  jdk-8u171-linux-x64.tar.gz  scala-2.11.12.tgz  spark-2.4.0-bin-hadoop2.7.tgz
[root@master soft]# mkdir -p /usr/spark
[root@master soft]# tar -zxvf spark-2.4.0-bin-hadoop2.7.tgz -C /usr/spark/
spark-2.4.0-bin-hadoop2.7/
spark-2.4.0-bin-hadoop2.7/python/
spark-2.4.0-bin-hadoop2.7/python/setup.cfg
spark-2.4.0-bin-hadoop2.7/python/pyspark/
spark-2.4.0-bin-hadoop2.7/python/pyspark/resultiterable.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/python/
spark-2.4.0-bin-hadoop2.7/python/pyspark/python/pyspark/
spark-2.4.0-bin-hadoop2.7/python/pyspark/python/pyspark/shell.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/heapq3.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/join.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/version.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/rdd.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/java_gateway.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/find_spark_home.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/_globals.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/worker.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/accumulators.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/feature.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/random.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/recommendation.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/fpm.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/classification.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/stat/
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/stat/KernelDensity.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/stat/distribution.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/stat/test.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/stat/_statistics.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/stat/__init__.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/util.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/regression.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/clustering.py
spark-2.4.0-bin-hadoop2.7/python/pyspark/mllib/linalg/
```

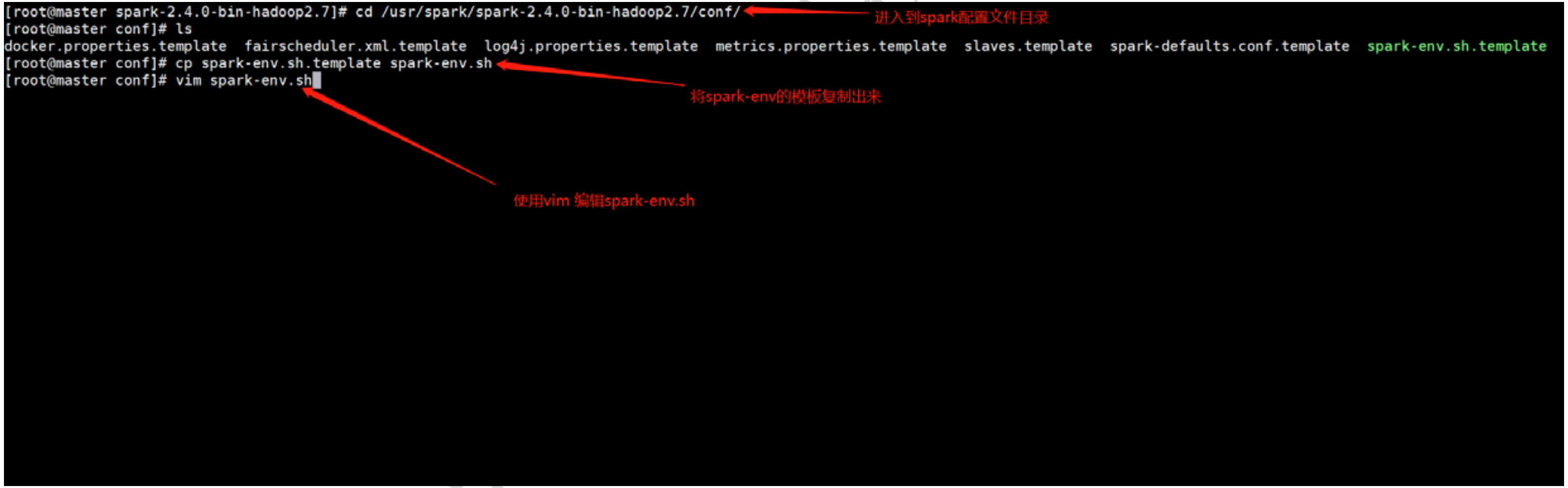
打开/opt/soft/可以看到我们已经放好的spark安装包

创建spark工作路径

解压spark到我们刚创建的spark工作路径中

复制spark-env.sh模板

我们需要将spark-env.sh.template复制为spark-env.sh，命令为：`cp spark-env.sh.template spark-env.sh`.当复制出spark-env.sh文件后我们可以使用vim进行编译，具体操作如下图所示：



```
[root@master spark-2.4.0-bin-hadoop2.7]# cd /usr/spark/spark-2.4.0-bin-hadoop2.7/conf/
[root@master conf]# ls
docker.properties.template  fairscheduler.xml.template  log4j.properties.template  metrics.properties.template  slaves.template  spark-defaults.conf.template  spark-env.sh.template
[root@master conf]# cp spark-env.sh.template spark-env.sh
[root@master conf]# vim spark-env.sh
```

进入到spark配置文件目录

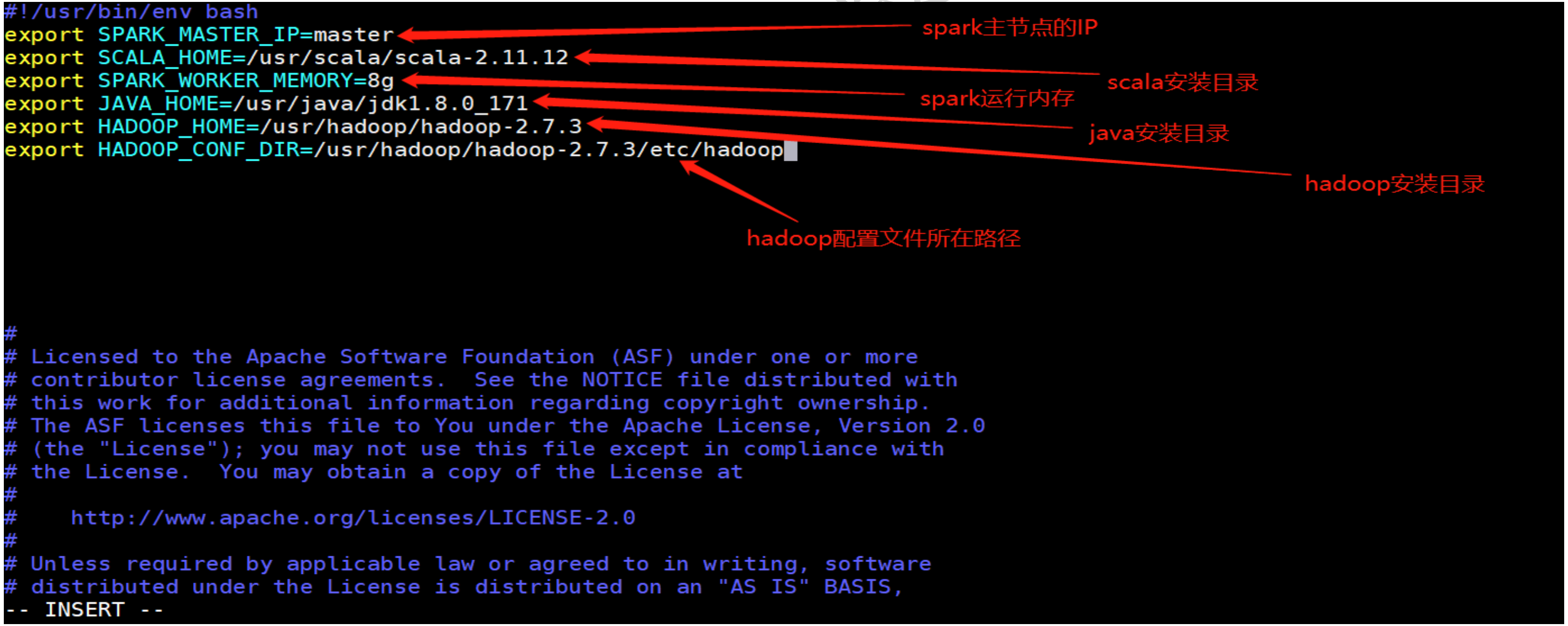
将spark-env的模板复制出来

使用vim 编辑spark-env.sh

配置spark-env.sh文件

添加以下内容，具体操作如下图所示：

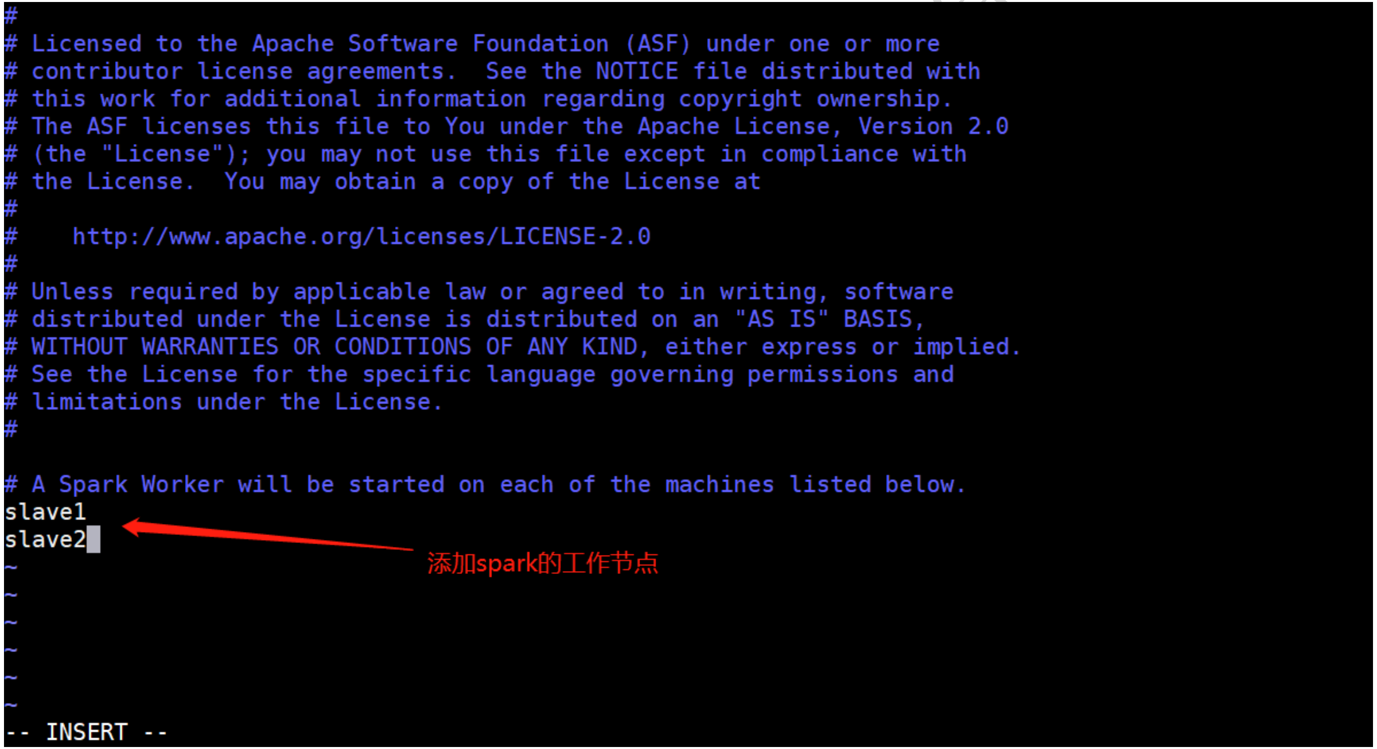
```
export SPARK_MASTER_IP=master
export SCALA_HOME=/usr/scala/scala-2.11.12
export SPARK_WORKER_MEMORY=8g
export JAVA_HOME=/usr/java/jdk1.8.0_171
export HADOOP_HOME=/usr/hadoop/hadoop-2.7.3
export HADOOP_CONF_DIR=/usr/hadoop/hadoop-2.7.3/etc/hadoop
```



```
#!/usr/bin/env bash
export SPARK_MASTER_IP=master
export SCALA_HOME=/usr/scala/scala-2.11.12
export SPARK_WORKER_MEMORY=8g
export JAVA_HOME=/usr/java/jdk1.8.0_171
export HADOOP_HOME=/usr/hadoop/hadoop-2.7.3
export HADOOP_CONF_DIR=/usr/hadoop/hadoop-2.7.3/etc/hadoop







#
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
-- INSERT --
```

配置spark从节点，修改slaves文件

- *命令：`cp slaves.template.template slaves`
  使用vim命令编辑 slaves，其内容如下图所示：



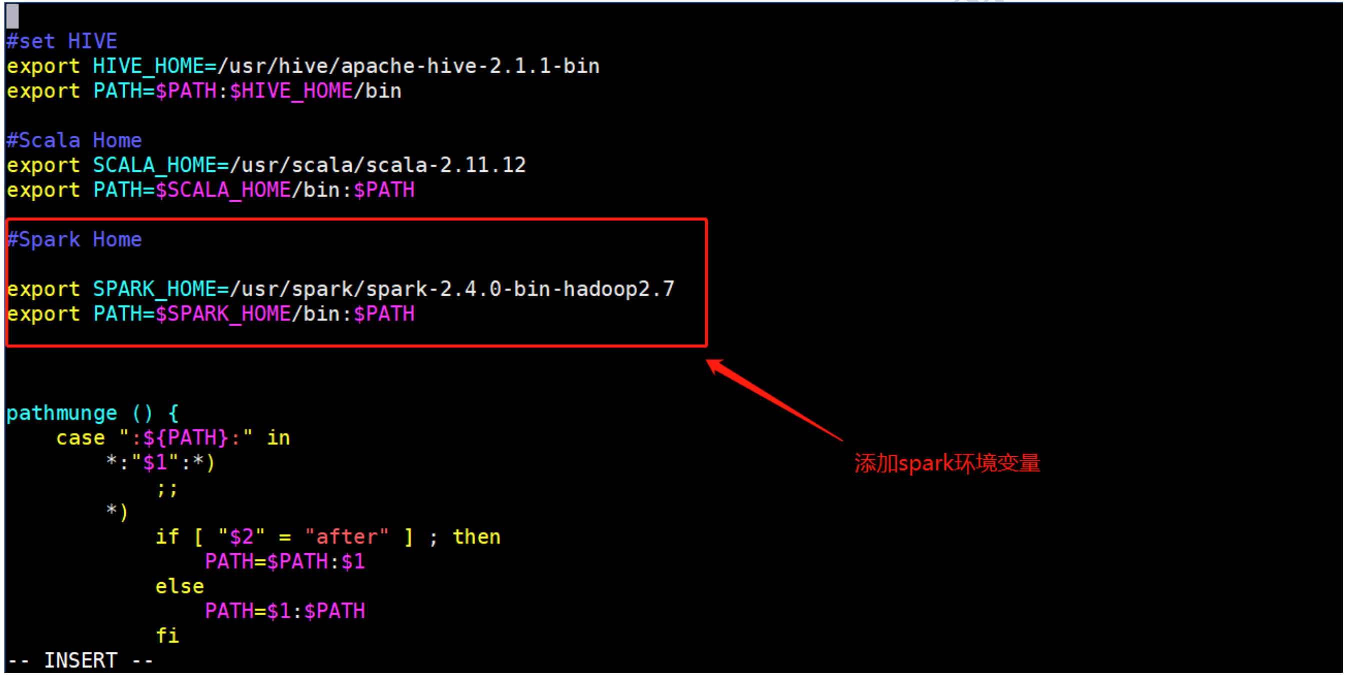
```
#
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# A Spark Worker will be started on each of the machines listed below.
slave1
slave2
~
~
~
~
~
~
-- INSERT --
```

配置spark环境变量

* 命令：`vim /etc/profile` 在其中添加如下内容：

```
export SPARK_HOME=/usr/spark/spark-2.4.0-bin-hadoop2.7
export PATH=$SPARK_HOME/bin:$PATH
```

* 使环境变量生效：`source /etc/profile`

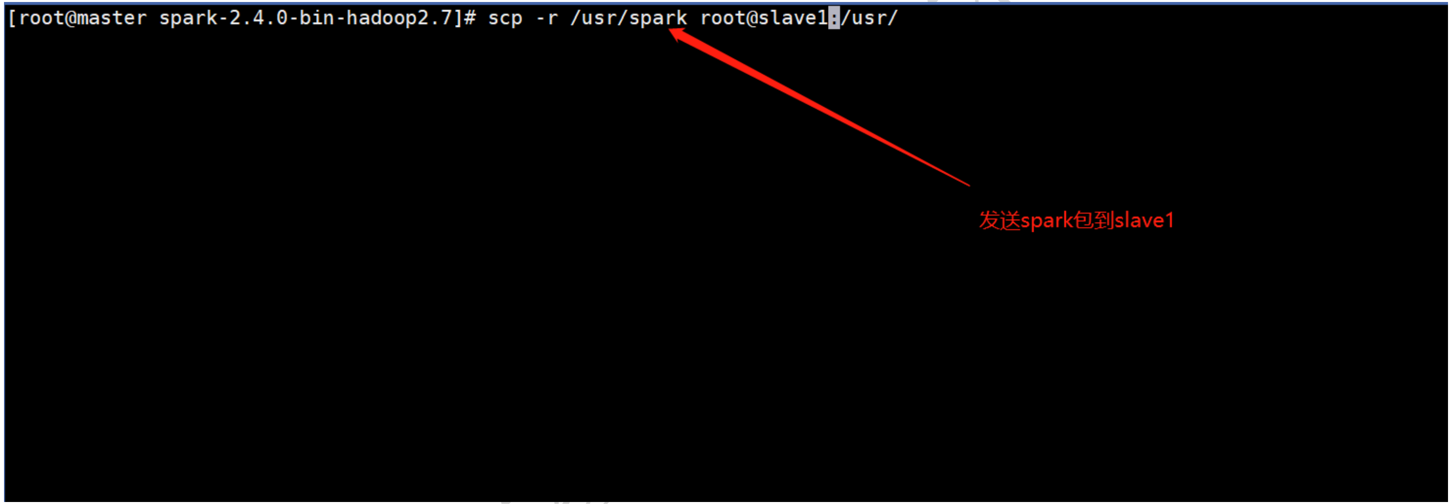


```

#set HIVE
export HIVE_HOME=/usr/hive/apache-hive-2.1.1-bin
export PATH=$PATH:$HIVE_HOME/bin

#Scala Home
export SCALA_HOME=/usr/scala/scala-2.11.12
export PATH=$SCALA_HOME/bin:$PATH

#Spark Home

export SPARK_HOME=/usr/spark/spark-2.4.0-bin-hadoop2.7
export PATH=$SPARK_HOME/bin:$PATH



pathmunge () {
    case ":${PATH}:" in
        *:"$1":*)
            ;;
        *)
            if [ "$2" = "after" ] ; then
                PATH=$PATH:$1
            else
                PATH=$1:$PATH
            fi
-- INSERT --
```

添加spark环境变量

发送配置好的spark安装包到子节点

* 接下来向所有子节点发送spark配置好的安装包，具体操作如下图所示：
* **注：** slave2同理，请同学们自己进行操作。
* **命令：** `scp -r /usr/spark root@slave1:/usr/`
* **命令：** `scp -r /usr/spark root@slave2:/usr/`

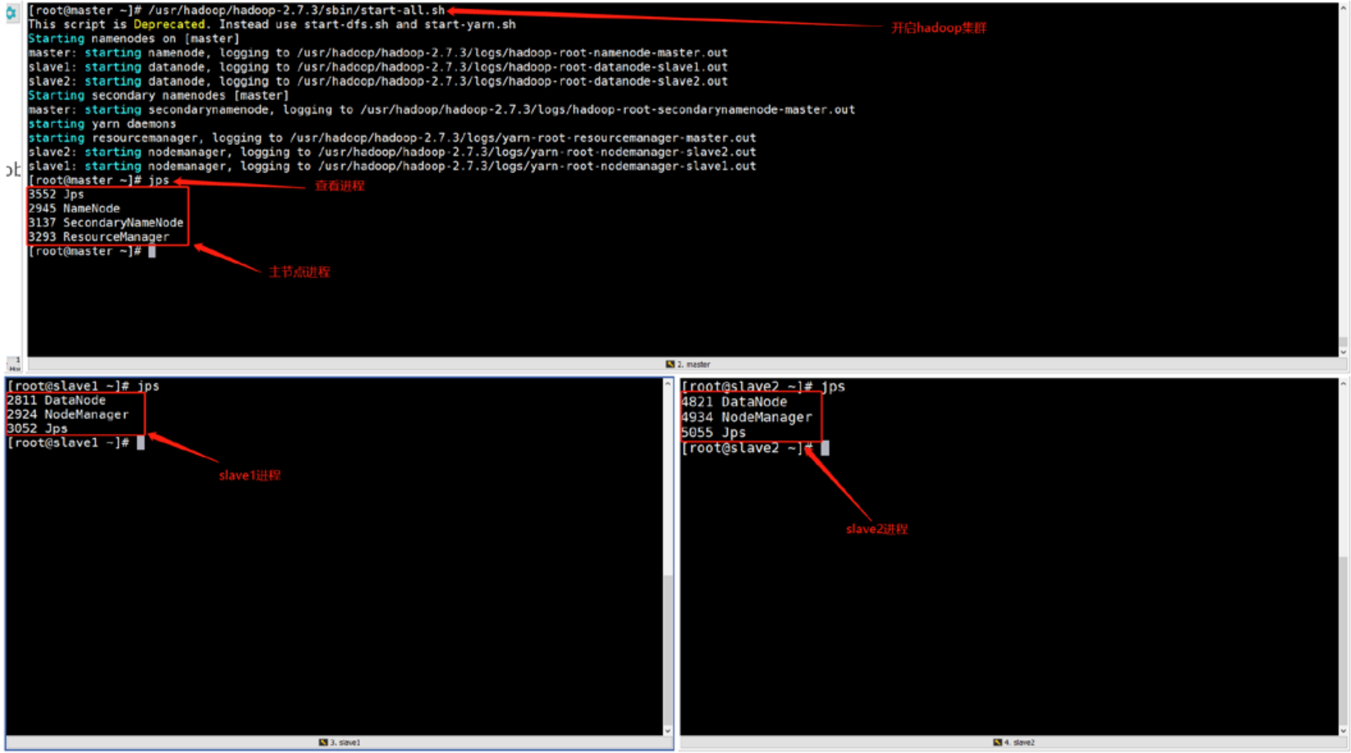


```
[root@master spark-2.4.0-bin-hadoop2.7]# scp -r /usr/spark root@slave1:/usr/
```

* **修改slave1和slave2的环境变量**，此步骤和修改master中spark环境变量相同，这里就不多介绍了，最后记得是环境变量生效。这时我们的spark环境就安装成功了。

测试spark环境

因为我们安装的是spark on yarn 模式，所有接下来我们需要开启hadoop环境，我们只需要在master节点上执行此**命令：**`/usr/hadoop/hadoop-2.7.3/sbin/start-all.sh`即可开启hadoop集群，具体操作如下图所示:



```
[root@master ~]# /usr/hadoop/hadoop-2.7.3/sbin/start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [master]
master: starting namenode, logging to /usr/hadoop/hadoop-2.7.3/logs/hadoop-root-namenode-master.out
slave1: starting datanode, logging to /usr/hadoop/hadoop-2.7.3/logs/hadoop-root-datanode-slave1.out
slave2: starting datanode, logging to /usr/hadoop/hadoop-2.7.3/logs/hadoop-root-datanode-slave2.out
Starting secondary namenodes [master]
master: starting secondarynamenode, logging to /usr/hadoop/hadoop-2.7.3/logs/hadoop-root-secondarynamenode-master.out
starting yarn daemons
starting resourcemanager, logging to /usr/hadoop/hadoop-2.7.3/logs/yarn-root-resourcemanager-master.out
slave2: starting nodemanager, logging to /usr/hadoop/hadoop-2.7.3/logs/yarn-root-nodemanager-slave2.out
slave1: starting nodemanager, logging to /usr/hadoop/hadoop-2.7.3/logs/yarn-root-nodemanager-slave1.out
[root@master ~]# jps
3552 Jps
2945 NameNode
3137 SecondaryNameNode
3293 ResourceManager
[root@master ~]#
```

开启hadoop集群

查看进程

主节点进程

```
[root@slave1 ~]# jps
2811 DataNode
2924 NodeManager
3052 Jps
[root@slave1 ~]#
```

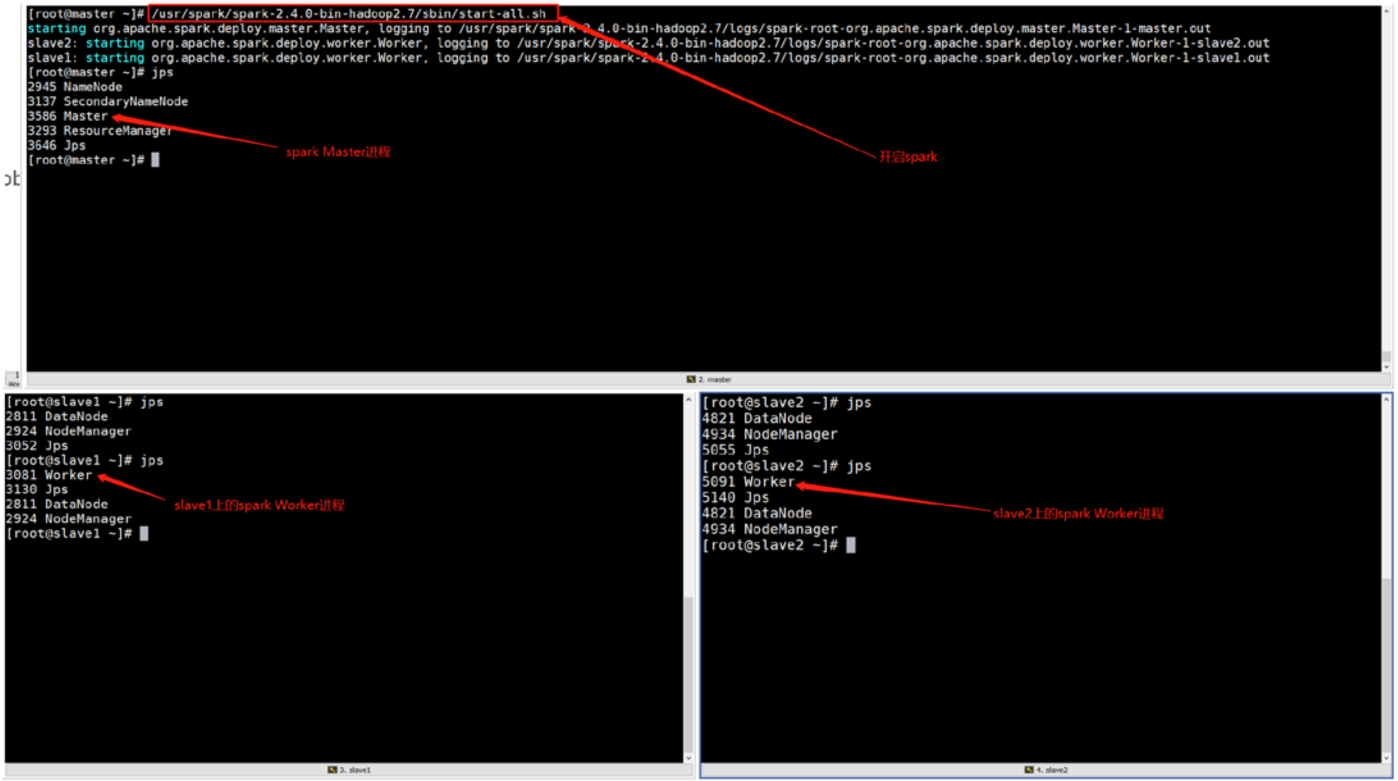
slave1进程

```
[root@slave2 ~]# jps
4821 DataNode
4934 NodeManager
5055 Jps
[root@slave2 ~]#
```

slave2进程

开启spark集群

我们只需要在master节点上执行此命令：`/usr/spark/spark-2.4.0-bin-hadoop2.7/sbin/start-all.sh`即可开启hadoop集群，具体操作如下图所示:



```
[root@master ~]# /usr/spark/spark-2.4.0-bin-hadoop2.7/sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /usr/spark/spark-2.4.0-bin-hadoop2.7/logs/spark-root-org.apache.spark.deploy.master.Master-1-master.out
slave2: starting org.apache.spark.deploy.worker.Worker, logging to /usr/spark/spark-2.4.0-bin-hadoop2.7/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-slave2.out
slave1: starting org.apache.spark.deploy.worker.Worker, logging to /usr/spark/spark-2.4.0-bin-hadoop2.7/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-slave1.out
[root@master ~]# jps
2945 NameNode
3137 SecondaryNameNode
3586 Master
3293 ResourceManager
3646 Jps
[root@master ~]# 
```

spark Master进程

开启spark

```
[root@slave1 ~]# jps
2811 DataNode
2924 NodeManager
3052 Jps
[root@slave1 ~]# jps
3081 Worker
3130 Jps
2811 DataNode
2924 NodeManager
[root@slave1 ~]# 
```

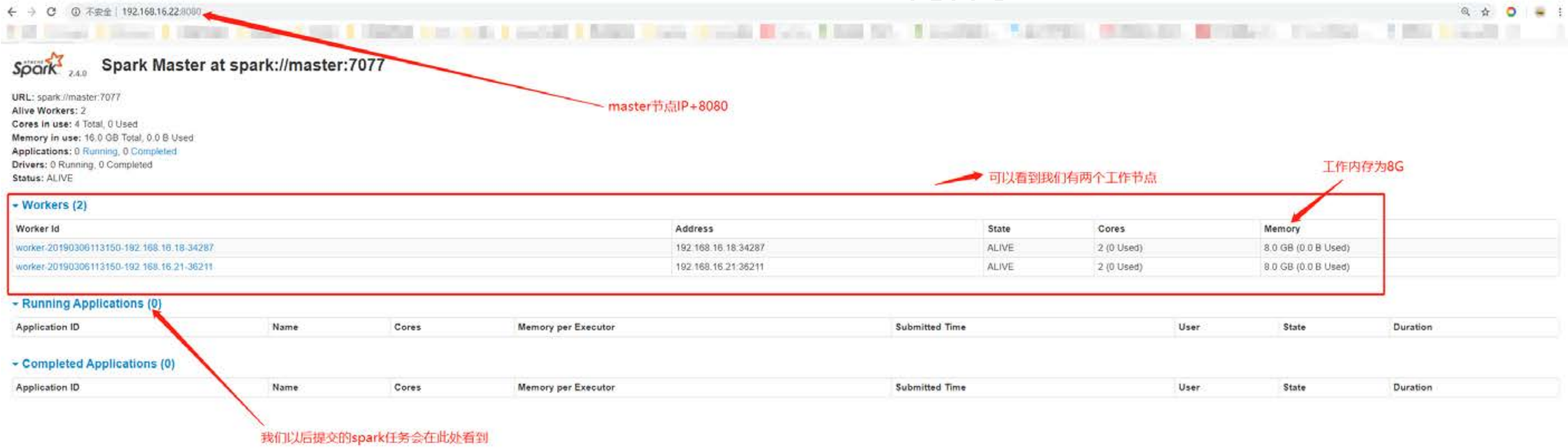
slave1上的spark Worker进程

```
[root@slave2 ~]# jps
4821 DataNode
4934 NodeManager
5055 Jps
[root@slave2 ~]# jps
5091 Worker
5140 Jps
4821 DataNode
4934 NodeManager
[root@slave2 ~]# 
```

slave2上的spark Worker进程

访问SparkWeb界面

我们可以在浏览器中输入我们master节点的IP地址，端口号为8080具体操作如下图所示：



192.168.16.22:8080

Spark Master at spark://master:7077

URL: spark://master:7077
Alive Workers: 2
Cores in use: 4 Total, 0 Used
Memory in use: 16.0 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory
worker-20190306113150-192.168.16.18-34287	192.168.16.18:34287	ALIVE	2 (0 Used)	8.0 GB (0.0 B Used)
worker-20190306113150-192.168.16.21-36211	192.168.16.21:36211	ALIVE	2 (0 Used)	8.0 GB (0.0 B Used)

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

master节点IP+8080

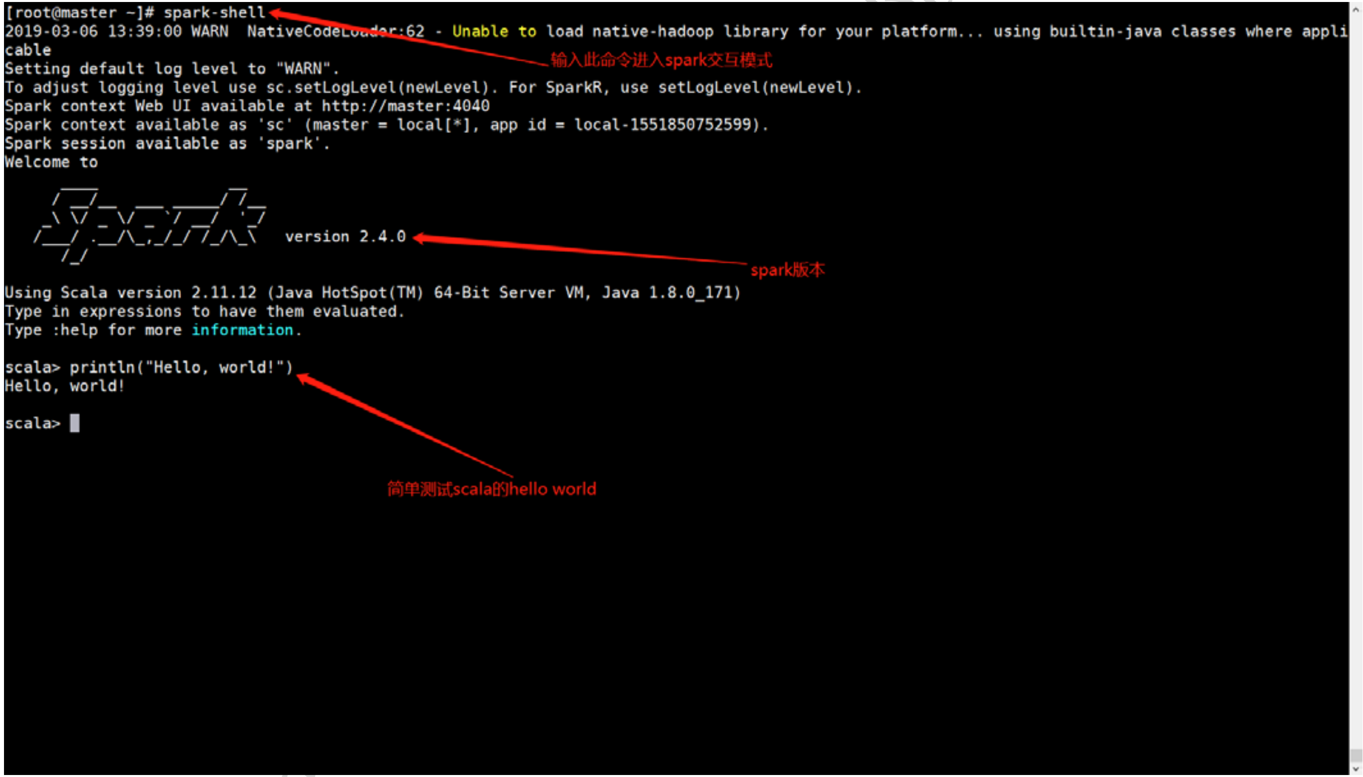
可以看到我们有两个工作节点

工作内存为8G

我们以后提交的spark任务会在此处看到

开启spark-shel

* 接下来我们开启我们的spark-shell以及pyspark进入到spark的交互模式：
* 首先spark-shell此时进入的是scala环境的spark交互模式，具体操作如下图所示：



```
[root@master ~]# spark-shell
2019-03-06 13:39:00 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where appli
cable
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://master:4040
Spark context available as 'sc' (master = local[*], app id = local-1551850752599).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.0
      /_/

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_171)
Type in expressions to have them evaluated.
Type :help for more information.

scala> println("Hello, world!")
Hello, world!

scala> 
```

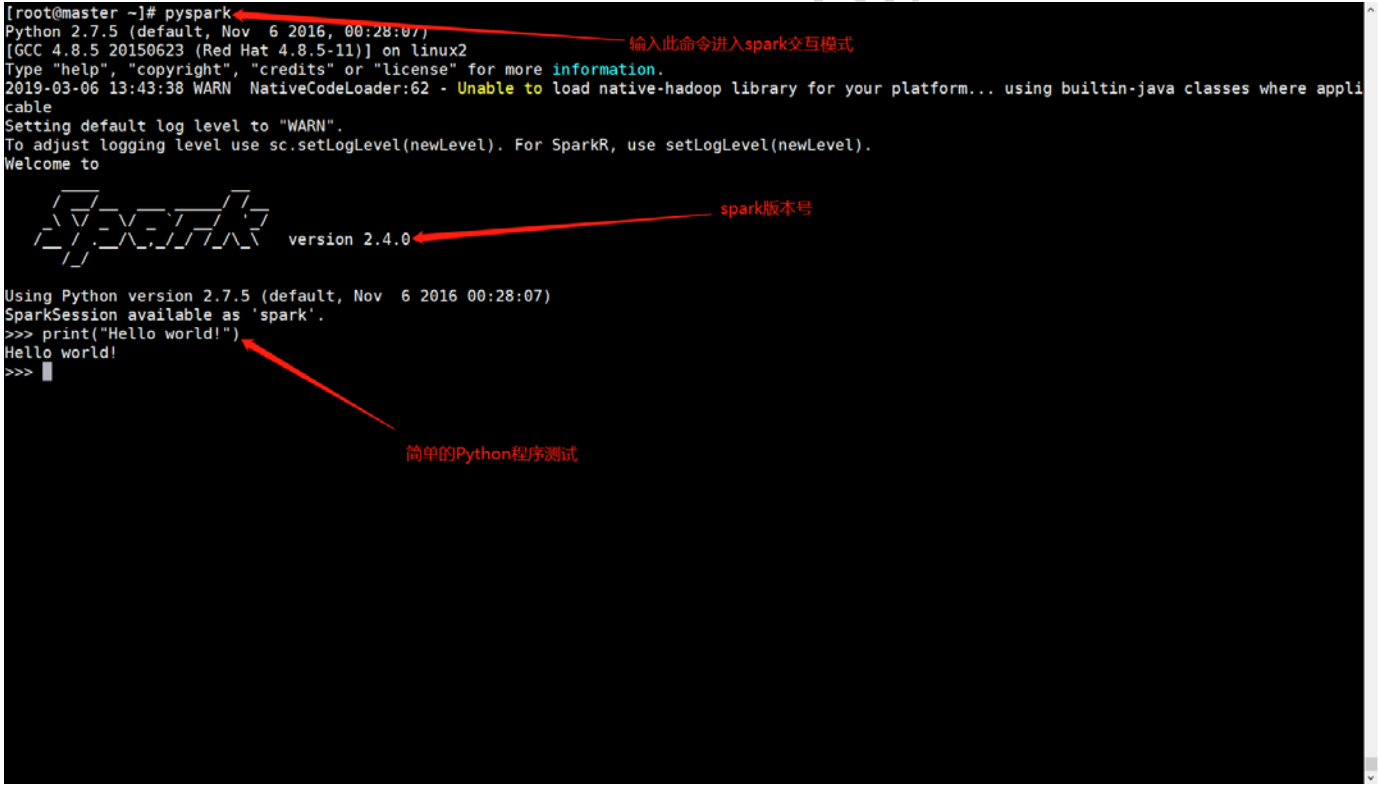
输入此命令进入spark交互模式

spark版本

简单测试scala的hello world

开启pyspark

接下来我们输入**命令**：`pyspatk`进入python环境下的spark交互模式，具体操作如下图所示：



```
[root@master ~]# pyspark
Python 2.7.5 (default, Nov  6 2016, 00:28:07)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
2019-03-06 13:43:38 WARN  NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where appli
cable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.0
      /_/

Using Python version 2.7.5 (default, Nov  6 2016 00:28:07)
SparkSession available as 'spark'.
>>> print("Hello world!")
Hello world!
>>> 
```

Thank you!