一、虚拟机

a) 环境版本

i. CentOS-7.0-1406-x86_64-GnomeLive.iso

about 云日志分析项目，由于在 Linux 系统中，centos 比较受企业，应该选择 centos 作为项目选择的操作系统，这里使用的是桌面版 centos

下载链接：

http://archive.kernel.org/centos-vault/7.0.1406/isos/x86_64/

CentOS-7.0-1406-x86_64-DVD.iso 标准安装版，一般下载这个就可以了

CentOS-7.0-1406-x86_64-NetInstall.iso 网络安装镜像

CentOS-7.0-1406-x86_64-Everything.iso 对完整版安装盘的软件进行补充，集成所有软件。

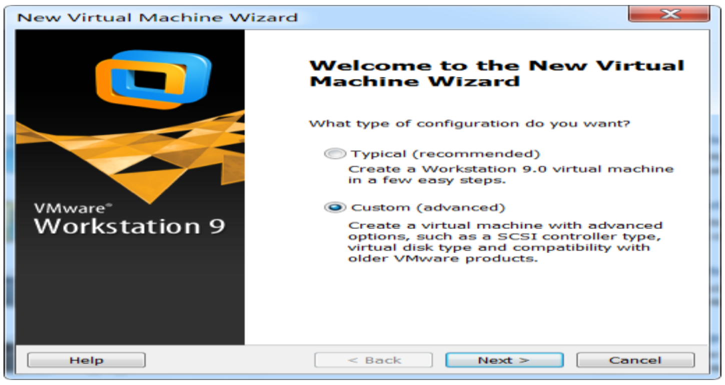
CentOS-7.0-1406-x86_64-GnomeLive.iso GNOME 桌面版

CentOS-7.0-1406-x86_64-KdeLive.iso KDE 桌面版

CentOS-7.0-1406-x86_64-livecd.iso 光盘上运行的系统，类拟于 winpe

ii. VMware_workstation_ful12

b) 虚拟机安装

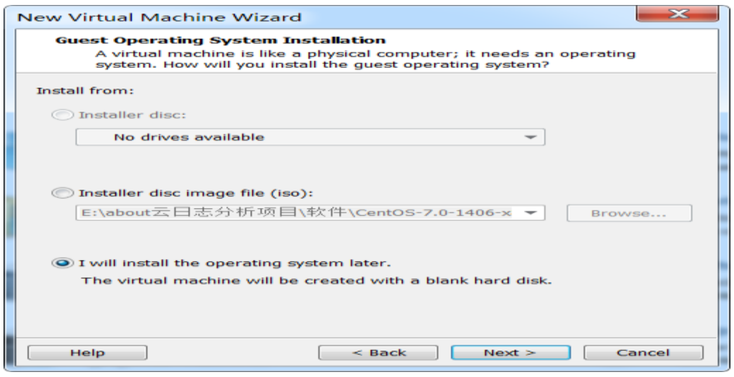

New Virtual Machine Wizard

Welcome to the New Virtual Machine Wizard

What type of configuration do you want?

( ) Typical (recommended)
Create a Workstation 9.0 virtual machine in a few easy steps.

[✓ Custom (advanced)]
Create a virtual machine with advanced options, such as a SCSI controller type, virtual disk type and compatibility with older VMware products.


New Virtual Machine Wizard

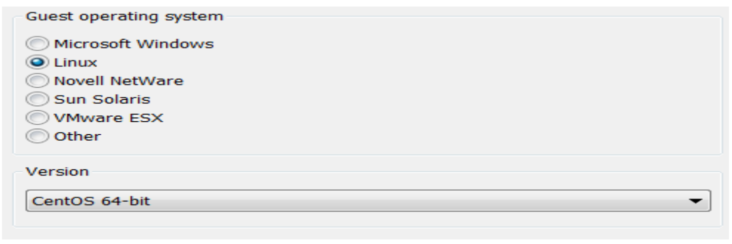
Guest Operating System Installation
A virtual machine is like a physical computer; it needs an operating system. How will you install the guest operating system?

Install from:

( ) Installer disc:
No drives available

( ) Installer disc image file (iso):
E:\about云日志分析项目\软件\CentOS-7.0-1406-x

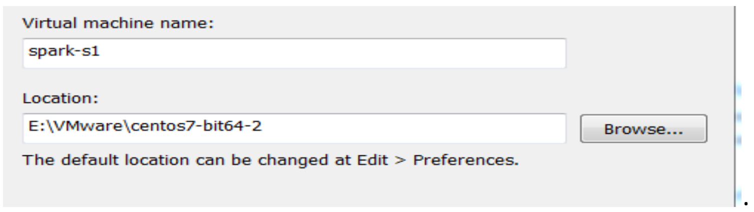
[✓ I will install the operating system later.]
The virtual machine will be created with a blank hard disk.


Guest operating system

( ) Microsoft Windows
[✓ Linux]
( ) Novell NetWare
( ) Sun Solaris
( ) VMware ESX
( ) Other

Version

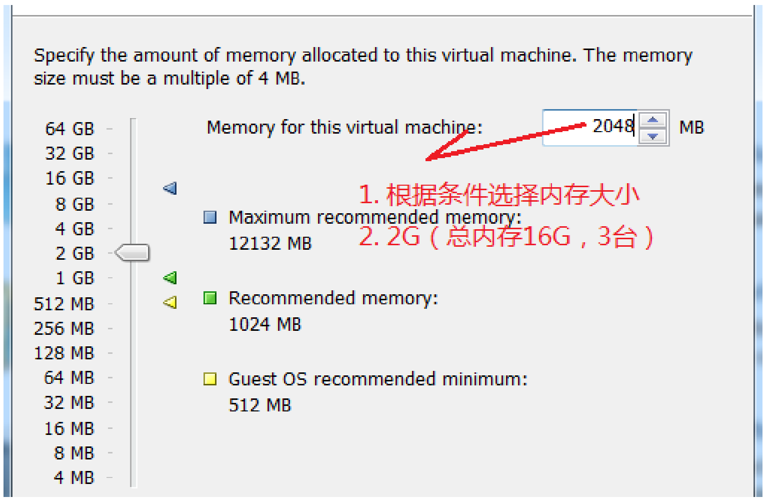
CentOS 64-bit



Virtual machine name:
spark-s1

Location:
E:\VMware\centos7-bit64-2

The default location can be changed at Edit > Preferences.



Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB.

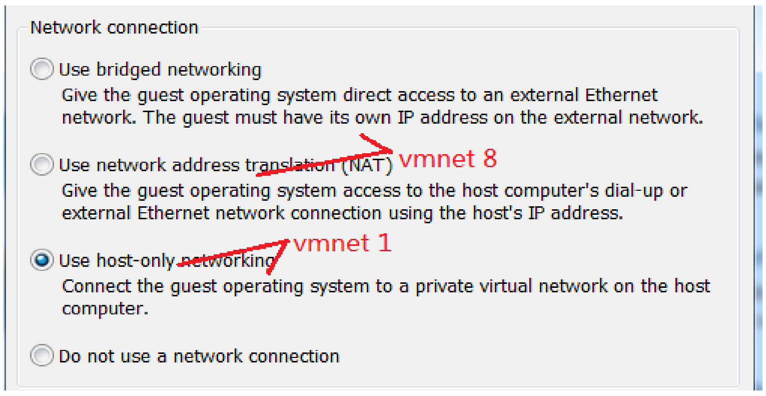
Memory for this virtual machine: 2048 MB

1. 根据条件选择内存大小
2. 2G（总内存16G，3台）

🟦 Maximum recommended memory:
12132 MB

🟩 Recommended memory:
1024 MB

🟨 Guest OS recommended minimum:
512 MB



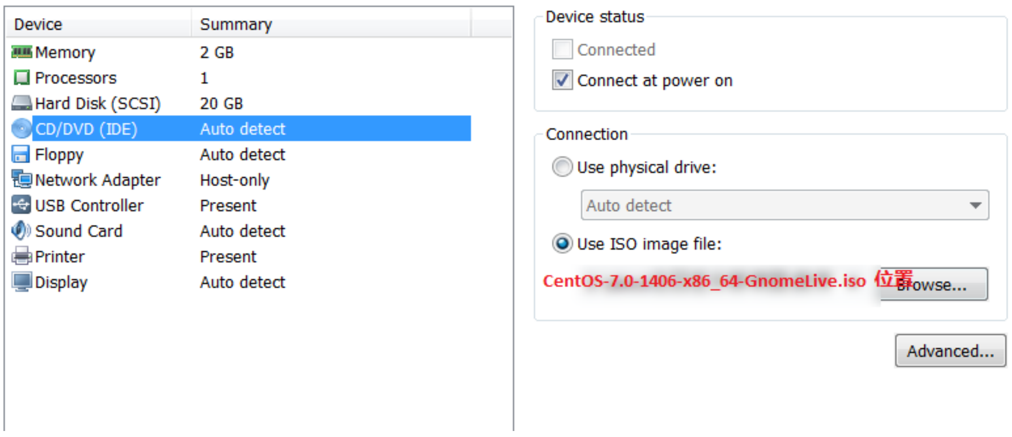
Network connection

○ Use bridged networking
Give the guest operating system direct access to an external Ethernet network. The guest must have its own IP address on the external network.

○ Use network address translation (NAT) vmnet 8
Give the guest operating system access to the host computer's dial-up or external Ethernet network connection using the host's IP address.

● Use host-only networking vmnet 1
Connect the guest operating system to a private virtual network on the host computer.

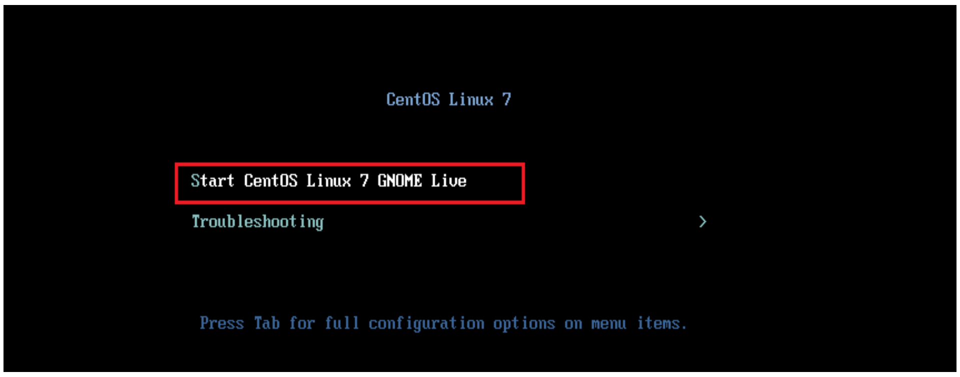
○ Do not use a network connection



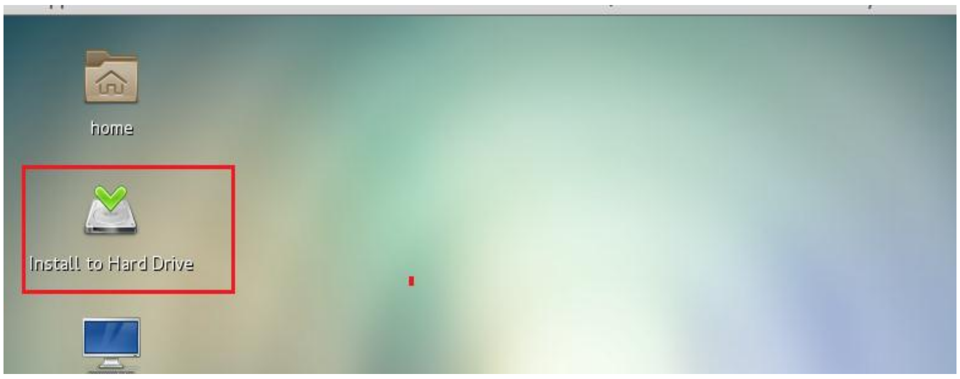
Device	Summary
Memory	2 GB
Processors	1
Hard Disk (SCSI)	20 GB
CD/DVD (IDE)	Auto detect
Floppy	Auto detect
Network Adapter	Host-only
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

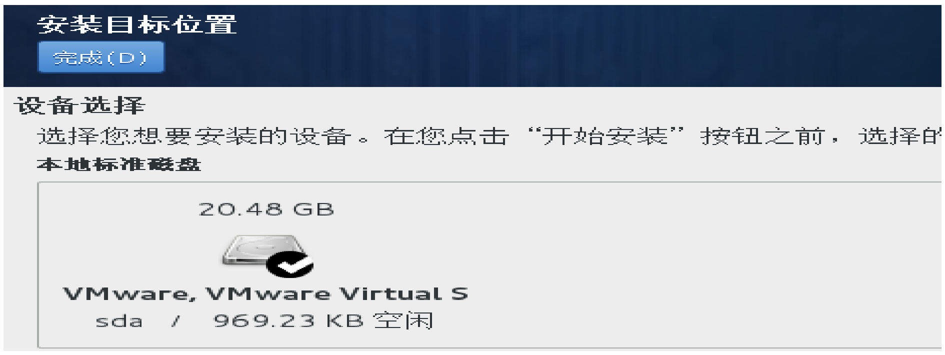
Device status
☐ Connected
☑ Connect at power on

Connection
○ Use physical drive:
Auto detect
● Use ISO image file:
CentOS-7.0-1406-x86_64-GnomeLive.iso 位置



```
CentOS Linux 7

Start CentOS Linux 7 GNOME Live
Troubleshooting                                  >

Press Tab for full configuration options on menu items.
```





安装目标位置

完成(D)

设备选择

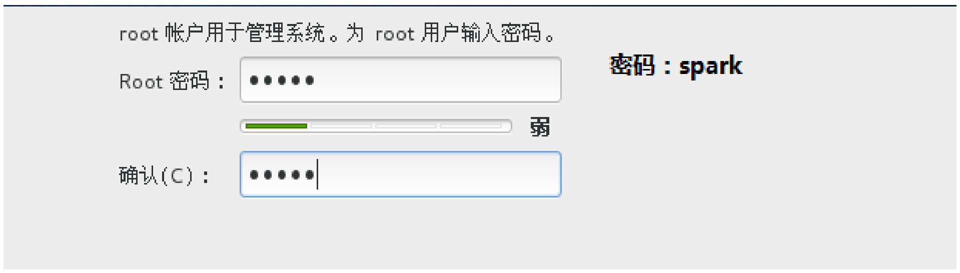
选择您想要安装的设备。在您点击"开始安装"按钮之前，选择的

本地标准磁盘

20.48 GB

VMware, VMware Virtual S

sda / 969.23 KB 空闲



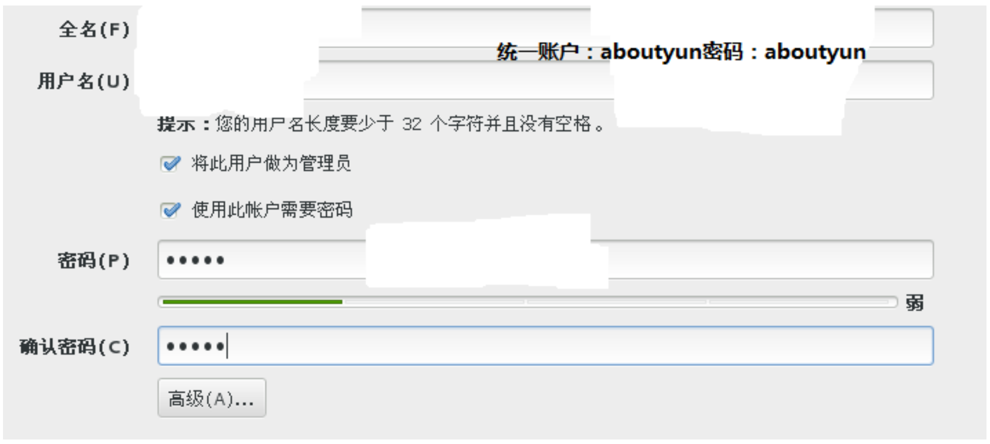
root 帐户用于管理系统。为 root 用户输入密码。

Root 密码： ●●●●●

弱

确认(C)： ●●●●●

密码：spark



全名(F) [redacted]

用户名(U) [redacted]

统一账户：aboutyun密码：aboutyun

提示：您的用户名长度要少于 32 个字符并且没有空格。

☑ 将此用户做为管理员

☑ 使用此帐户需要密码

密码(P) ●●●●●

弱

确认密码(C) ●●●●●

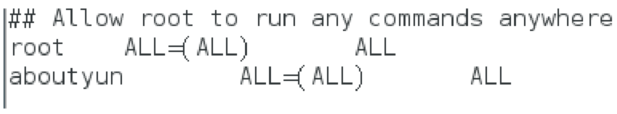
高级(A)...

c) 简单配置

i. 添加 sudo 权限

su

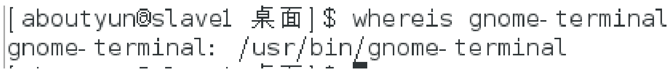
visudo



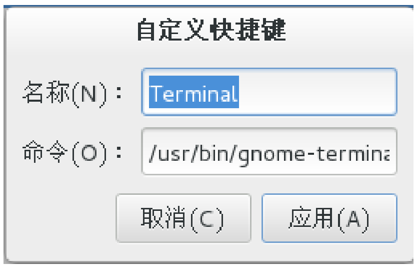
```
## Allow root to run any commands anywhere
root     ALL=(ALL)        ALL
aboutyun          ALL=(ALL)        ALL
```

ii. 设置快捷键

1. 系统工具->设置->键盘->快捷键->新建



```
[aboutyun@slave1 桌面]$ whereis gnome-terminal
gnome-terminal: /usr/bin/gnome-terminal
```



自定义快捷键

名称(N)： Terminal

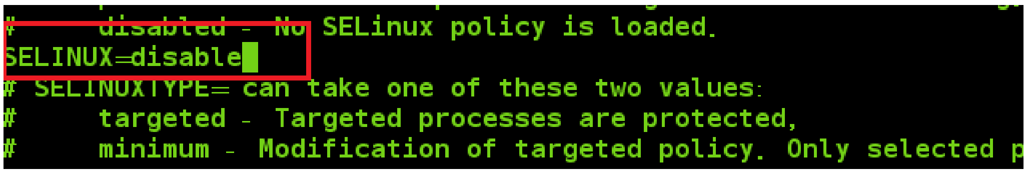
命令(O)： /usr/bin/gnome-termina

2. 设置自己喜欢的快捷键

iii. 关闭 SELINUX

1. vi /etc/sysconfig/selinux

2. 修改



```
#     disabled - No SELinux policy is loaded.
SELINUX=disable
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected p
```

iv. 关闭防火墙

1. Firewalld 简介

Centos7 中默认将原来的防火墙 iptables 升级为了 firewalld，firewalld 跟 iptables 比起来至少有两大好处：

a) 1、firewalld 可以动态修改单条规则，而不需要像 iptables 那样，在修改了规则后必须得全部刷新才可以生效；

b) 2、firewalld 在使用上要比 iptables 人性化很多，即使不明白“五张表五条链”而且对 TCP/IP 协议也不理解也可以实现大部分功能。

2. 命令

a) sudo systemctl status firewalld.service

b) sudo systemctl stop firewalld.service

c) sudo systemctl disable firewalld.service

v. ssh 开启

1. sudo systemctl start sshd.service

2. sudo systemctl status sshd.service

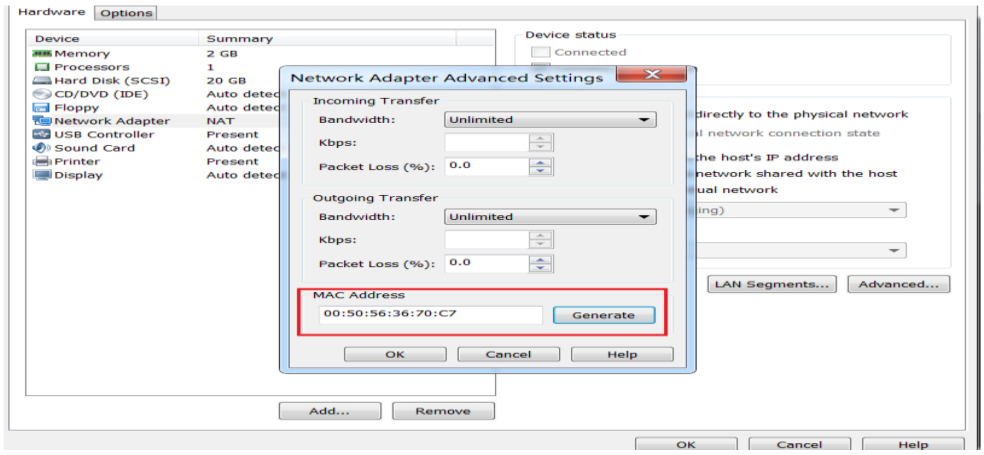
3. sudo systemctl enable sshd.service #开机启动 ssh

d) 克隆一台(也可以全部操作完成之后在克隆，但是需要修改 IP，主机名等)

i. 克隆一个虚拟机的内容远比重新搭建一个节点所需的时间快得多，然而克隆后需要更改一些常用配置，克隆后的虚拟机方能使用。下面仅描述一下需要更改的基本配置信息：

1.更改 ip 地址（虚拟机里面进行修改）;

2.更改 MAC 地址，（克隆后的虚拟机，右击-->设置-->网络适配器-->高级-->单机生成按钮）



Device	Summary
Memory	2 GB
Processors	1
Hard Disk (SCSI)	20 GB
CD/DVD (IDE)	Auto detec
Floppy	Auto detec
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detec
Printer	Present
Display	Auto detec

Network Adapter Advanced Settings

Incoming Transfer
Bandwidth: Unlimited
Kbps:
Packet Loss (%): 0.0

Outgoing Transfer
Bandwidth: Unlimited
Kbps:
Packet Loss (%): 0.0

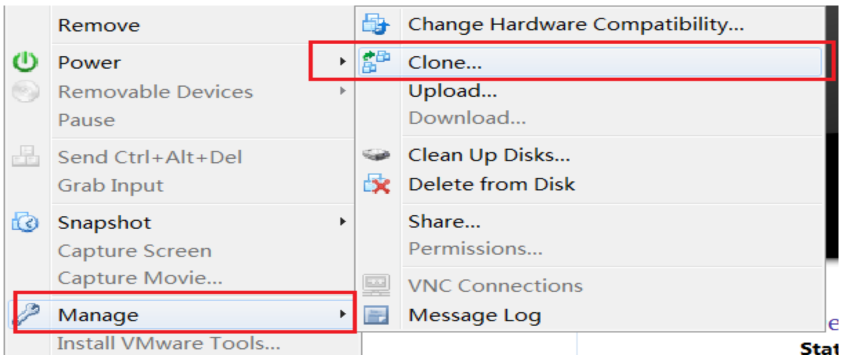
MAC Address
00:50:56:36:70:C7

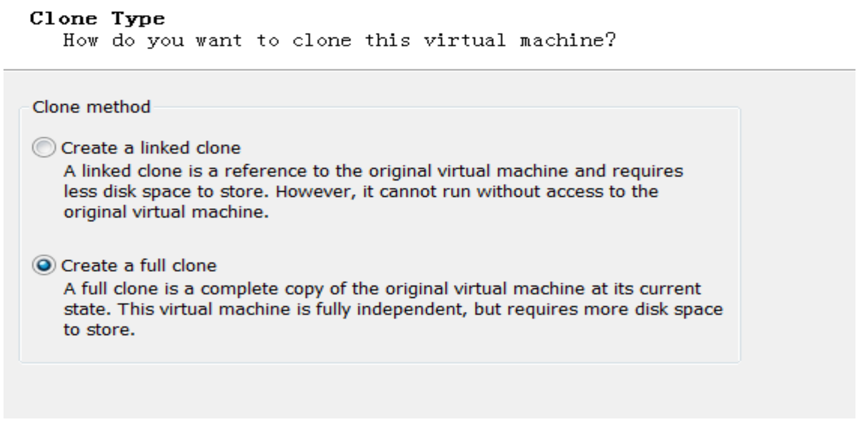
3.重启虚拟机即可与其他虚拟机进行通信了

提示与建议：

1.针对刚入门的童靴；建议在/etc/hosts (ubuntu 虚拟机)多加一些 ip hostname 的映射，空闲的映射也不会报错；

2.克隆后会导致软件新旧版本不一致，如 HBase 和Hadoop，需清理 HBase 中就版本中的数据，以及重新将 Hadoop 格式化。

ii. 

iii. 

Clone Type
How do you want to clone this virtual machine?

Clone method

( ) Create a linked clone
A linked clone is a reference to the original virtual machine and requires less disk space to store. However, it cannot run without access to the original virtual machine.

(●) Create a full clone
A full clone is a complete copy of the original virtual machine at its current state. This virtual machine is fully independent, but requires more disk space to store.

e) 集群设置

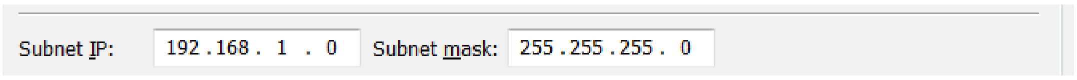
i. 设置主机名

1. sudo vi /etc/hostname

ii. 设置 ip（能 ping 通）

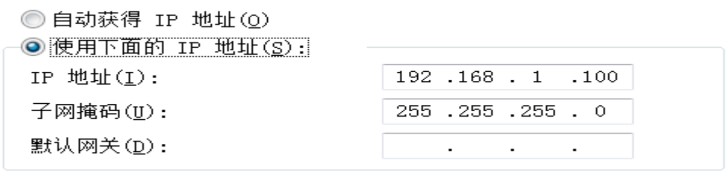
1. Host-only

a) 虚拟机设置网段



Subnet IP: 192 .168 . 1 . 0 Subnet mask: 255 .255 .255 . 0

b) Window 设置 Vmnet1 IPv4

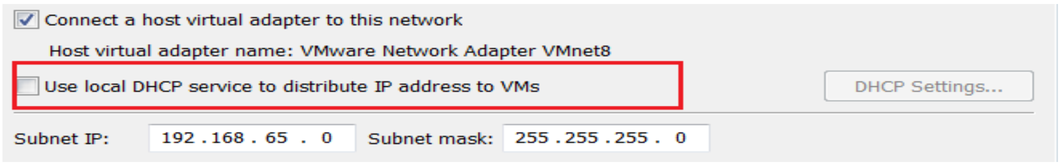


( ) 自动获得 IP 地址(O)
(●) 使用下面的 IP 地址(S):

IP 地址(I):	192 .168 . 1 .100
子网掩码(U):	255 .255 .255 . 0
默认网关(D):	. . .

c) 选中虚拟机 -> 右键 -> settings -> network adapter -> host only -> ok

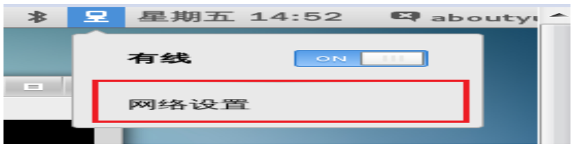
2. Nat（这里只需要注意禁止 DHCP,win7 网关,dns[114]）

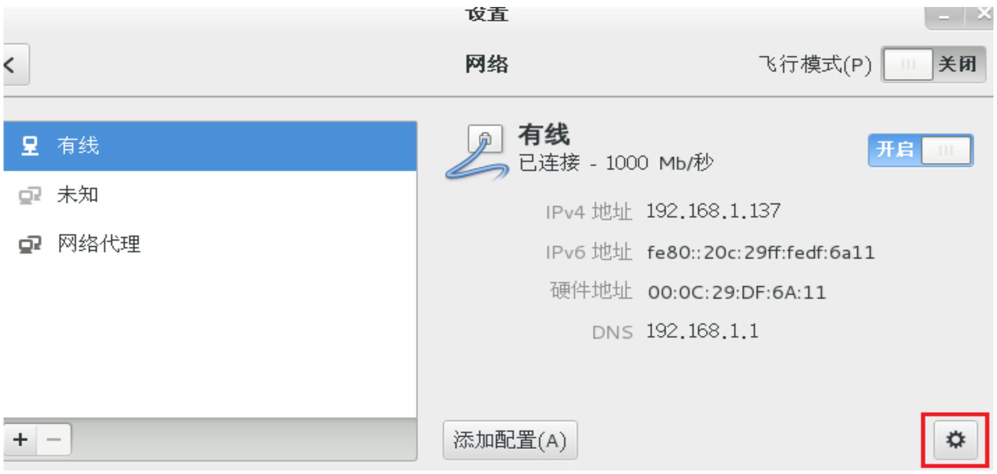


[✓] Connect a host virtual adapter to this network
Host virtual adapter name: VMware Network Adapter VMnet8
[ ] Use local DHCP service to distribute IP address to VMs

Subnet IP: 192 .168 . 65 . 0 Subnet mask: 255 .255 .255 . 0

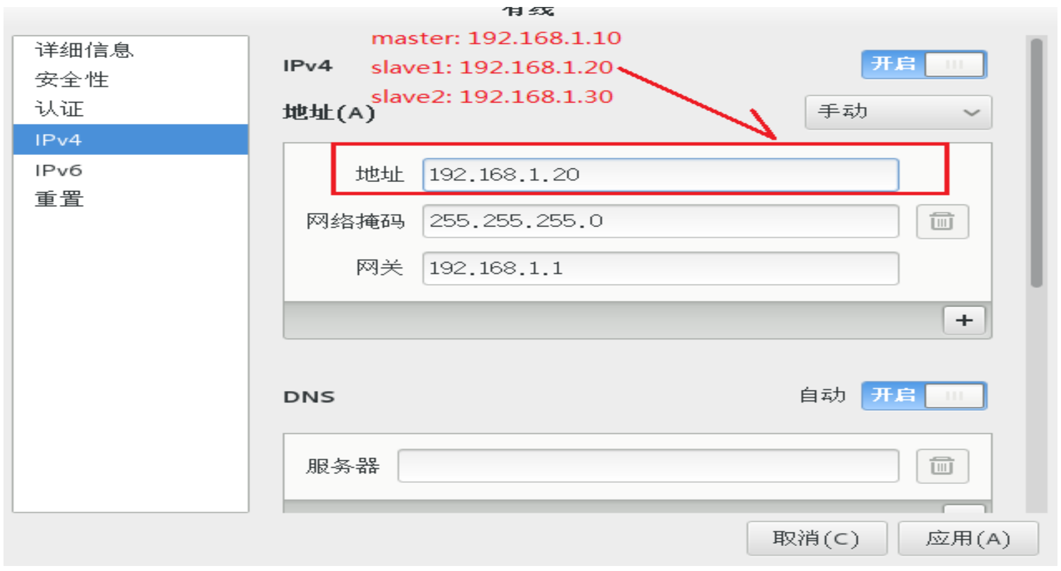
3. 设置 centos 7 ip





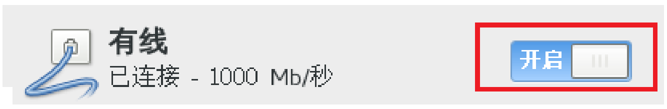
有线
已连接 - 1000 Mb/秒

IPv4 地址	192.168.1.137
IPv6 地址	fe80::20c:29ff:fedf:6a11
硬件地址	00:0C:29:DF:6A:11
DNS	192.168.1.1

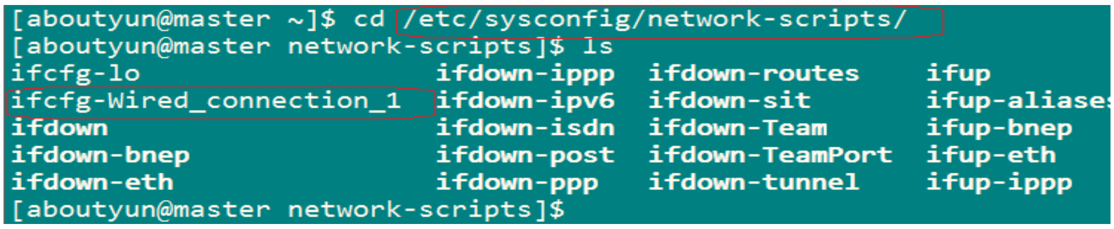


master: 192.168.1.10
slave1: 192.168.1.20
slave2: 192.168.1.30

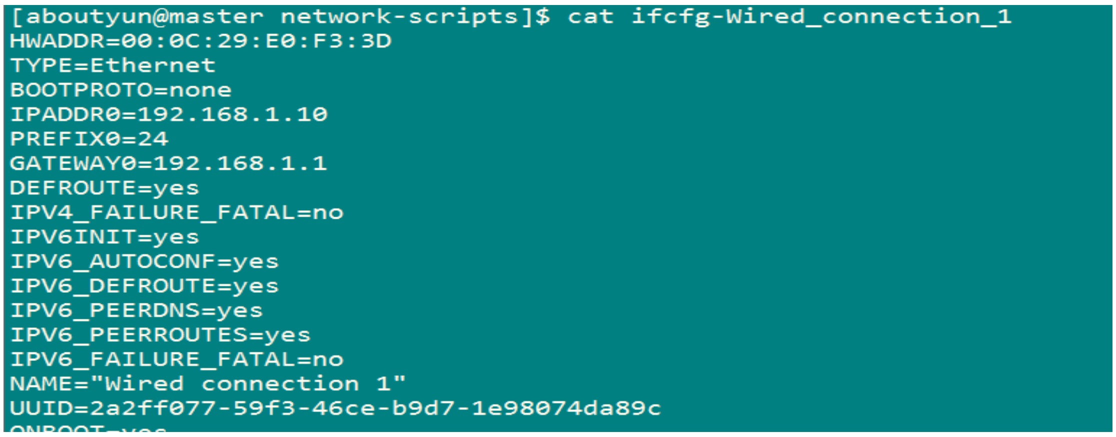
地址 192.168.1.20
网络掩码 255.255.255.0
网关 192.168.1.1



（重启）



```
[aboutyun@master ~]$ cd /etc/sysconfig/network-scripts/
[aboutyun@master network-scripts]$ ls
ifcfg-lo                    ifdown-ippp   ifdown-routes    ifup
ifcfg-Wired_connection_1    ifdown-ipv6   ifdown-sit       ifup-aliases
ifdown                      ifdown-isdn   ifdown-Team      ifup-bnep
ifdown-bnep                 ifdown-post   ifdown-TeamPort  ifup-eth
ifdown-eth                  ifdown-ppp    ifdown-tunnel    ifup-ippp
[aboutyun@master network-scripts]$
```



```
[aboutyun@master network-scripts]$ cat ifcfg-Wired_connection_1
HWADDR=00:0C:29:E0:F3:3D
TYPE=Ethernet
BOOTPROTO=none
IPADDR0=192.168.1.10
PREFIX0=24
GATEWAY0=192.168.1.1
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME="Wired connection 1"
UUID=2a2ff077-59f3-46ce-b9d7-1e98074da89c
```

iii. 修改 hosts 文件

1. *sudo vi /etc/hosts*

192.168.1.10	*master*
192.168.1.20	*slave1*
192.168.1.30	*slave2*

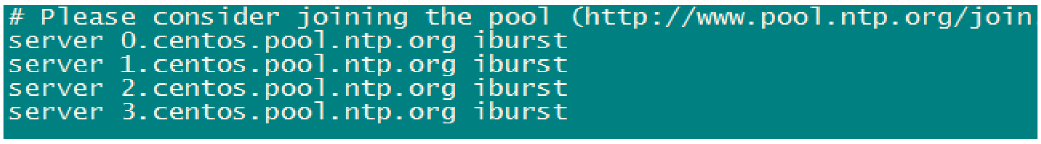
iv. NTP 服务（同步时间，集群必备）

1. *sudo vim /etc/ntp.conf*

ntp 设置方法：

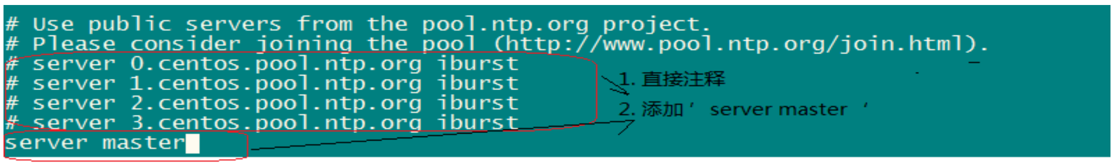
master 同步网络服务器，slave1 和 slave2 可同步 master 的时间

a) master：



```
# Please consider joining the pool (http://www.pool.ntp.org/join
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst
```

b) slave：



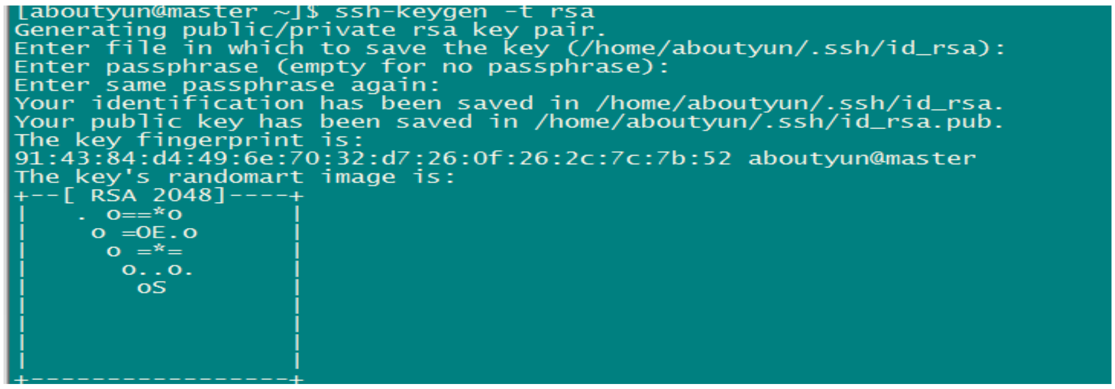
```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
# server 0.centos.pool.ntp.org iburst
# server 1.centos.pool.ntp.org iburst
# server 2.centos.pool.ntp.org iburst
# server 3.centos.pool.ntp.org iburst
server master
```

二、Hadoop 集群搭建

a) SSH 免密码登录

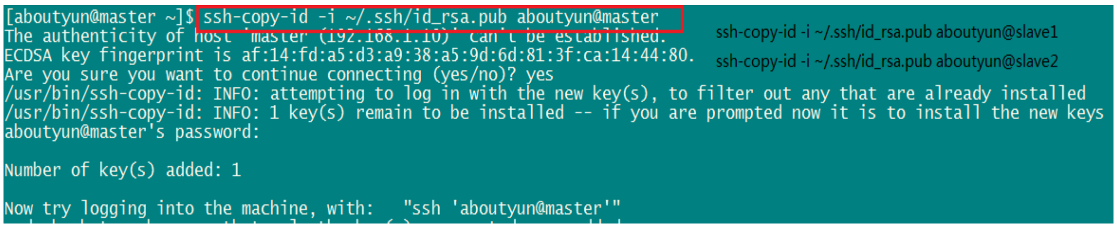
i. 需要实现在 master ssh 无密码登录本机、slave1 和 slave2。在 master 机器上，执行 ***ssh-keygen -t rsa***，然后一直回车，这样就生成了 aboutyun 用户在 master 上的公钥和秘钥。

ii.



```
[aboutyun@master ~]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aboutyun/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aboutyun/.ssh/id_rsa.
Your public key has been saved in /home/aboutyun/.ssh/id_rsa.pub.
The key fingerprint is:
91:43:84:d4:49:6e:70:32:d7:26:0f:26:2c:7c:7b:52 aboutyun@master
The key's randomart image is:
+--[ RSA 2048]----+
|  . o==*o        |
|   o =OE.o       |
|    o =*=        |
|     o..o.       |
|      oS         |
|                 |
|                 |
|                 |
|                 |
+-----------------+
```

ssh-copy-id -i ~/.ssh/id_rsa.pub aboutyun@slave1



```
[aboutyun@master ~]$ ssh-copy-id -i ~/.ssh/id_rsa.pub aboutyun@master
The authenticity of host 'master (192.168.1.10)' can't be established.
ECDSA key fingerprint is af:14:fd:a5:d3:a9:38:a5:9d:6d:81:3f:ca:14:44:80.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
aboutyun@master's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'aboutyun@master'"
```

ssh-copy-id -i ~/.ssh/id_rsa.pub aboutyun@slave1

ssh-copy-id -i ~/.ssh/id_rsa.pub aboutyun@slave2

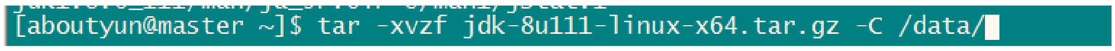
b) JDK

i. 创建安装目录

sudo mkdir /data

sudo chmod -R 777 /data

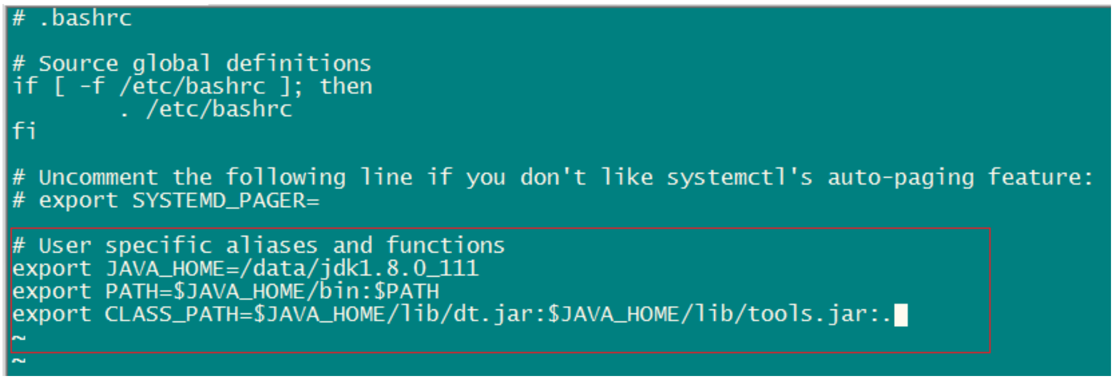
ii. 解压



```
[aboutyun@master ~]$ tar -xvzf jdk-8u111-linux-x64.tar.gz -C /data/
```

iii. 设置环境变量

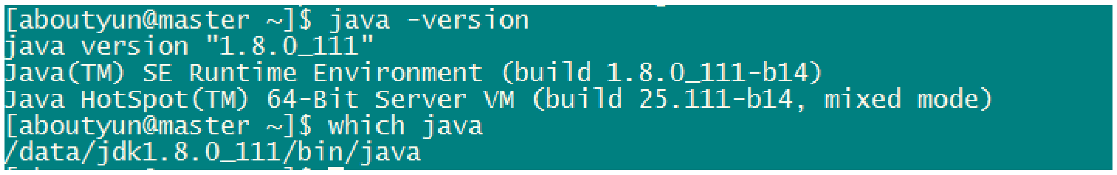
sudo vi ~/.bashrc



```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
export JAVA_HOME=/data/jdk1.8.0_111
export PATH=$JAVA_HOME/bin:$PATH
export CLASS_PATH=$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:.
~
~
```

iv. [aboutyun@master ~]$ *source ~/.bashrc*

v. 验证



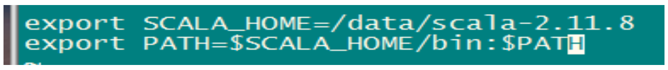
```
[aboutyun@master ~]$ java -version
java version "1.8.0_111"
Java(TM) SE Runtime Environment (build 1.8.0_111-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.111-b14, mixed mode)
[aboutyun@master ~]$ which java
/data/jdk1.8.0_111/bin/java
```

c) Scala

i. 解压

1. *tar -xvzf scala-2.11.8.tgz -C /data/*

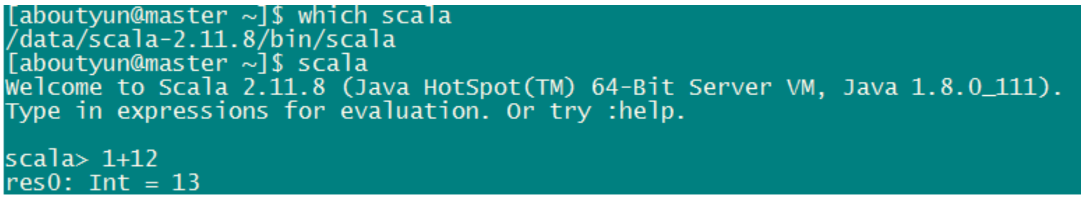
ii. 设置环境变量



```
export SCALA_HOME=/data/scala-2.11.8
export PATH=$SCALA_HOME/bin:$PATH
```

iii. *source ~/.bashrc*

iv. 验证



```
[aboutyun@master ~]$ which scala
/data/scala-2.11.8/bin/scala
[aboutyun@master ~]$ scala
Welcome to Scala 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_111).
Type in expressions for evaluation. Or try :help.

scala> 1+12
res0: Int = 13
```

d) Hadoop

i. 解压

1. *tar -xvzf hadoop-2.6.5.tar.gz -C /data/*

ii. 配置

${HADOOP_HOME}/etc/hadoop/hadoop-env.sh
${HADOOP_HOME}/etc/hadoop/yarn-env.sh
${HADOOP_HOME}/etc/hadoop/slaves
${HADOOP_HOME}/etc/hadoop/core-site.xml
${HADOOP_HOME}/etc/hadoop/hdfs-site.xml
${HADOOP_HOME}/etc/hadoop/mapred-site.xml
${HADOOP_HOME}/etc/hadoop/yarn-site.xml

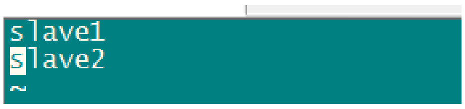
1. hadoop-env.sh
   a) 指定 JAVA_HOME

   export JAVA_HOME=/data/jdk1.8.0_111
2. yarn-env.sh

a) 同上

3. slaves

a) 将从节点加入



```
slave1
slave2
~
```

4. core-site.xml

```
<configuration>
	<property>
		<name>fs.defaultFS</name>
		<value>hdfs://master:8020</value>
	</property>

	<property>
		<name>hadoop.tmp.dir</name>
		<value>file:///home/aboutyun/hadoop/tmp</value>
		<description>Abase for other temporary directories.</description>
	</property>

	<property>
		<name>hadoop.proxyuser.aboutyun.hosts</name>
		<value>*</value>
		<description>abouyun 用户可以代理任意机器上的用户</description>
	</property>

	<property>
		<name>hadoop.proxyuser.aboutyun.groups</name>
		<value>*</value>
		<description>abouyun 用户代理任何组下的用户</description>
	</property>

	<property>
		<name>io.file.buffer.size</name>
		<value>131072</value>
	</property>
</configuration>
```

注意：需要创建 tmp 目录

5. hdfs-site.xml

```
<configuration>
        <property>
                <name>dfs.namenode.secondary.http-address</name>
                <value>master:9001</value>
        </property>

        <property>
                <name>dfs.namenode.name.dir</name>
                <value>file:///home/aboutyun/hadoop/namenode</value>
        </property>

        <property>
                <name>dfs.datanode.data.dir</name>
                <value>file:///home/aboutyun/hadoop/datanode</value>
        </property>

        <property>
                <name>dfs.replication</name>
                <value>3</value>
        </property>

        <property>
                <name>dfs.webhdfs.enabled</name>
                <value>true</value>
        </property>
</configuration>
```

注意：在本地创建 namenode，datanode 目录

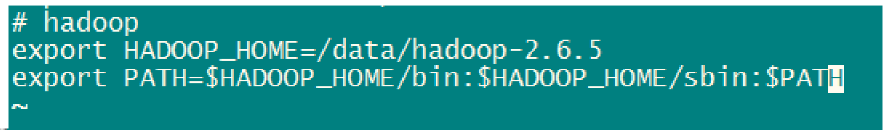
6. mapred-site.xml

```
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>

        <property>
                <name>mapreduce.jobhistory.address</name>
                <value>master:10020</value>
        </property>

        <property>
                <name>mapreduce.jobhistory.webapp.address</name>
                <value>master:19888</value>
        </property>
</configuration>
```

7. yarn-site.xml

```
<configuration>
	<property>
		<name>yarn.nodemanager.aux-services</name>
		<value>mapreduce_shuffle</value>
	</property>

	<property>
		<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
		<value>org.apache.hadoop.mapred.ShuffleHandler</value>
	</property>

	<property>
		<name>yarn.resourcemanager.address</name>
		<value>master:8032</value>
	</property>

	<property>
		<name>yarn.resourcemanager.scheduler.address</name>
		<value>master:8030</value>
	</property>

	<property>
		<name>yarn.resourcemanager.resource-tracker.address</name>
		<value>master:8031</value>
	</property>

	<property>
		<name>yarn.resourcemanager.admin.address</name>
		<value>master:8033</value>
	</property>

	<property>
		<name>yarn.resourcemanager.webapp.address</name>
		<value>master:8088</value>
	</property>
</configuration>
```

iii. 配置环境变量



```
# hadoop
export HADOOP_HOME=/data/hadoop-2.6.5
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
~
```

```
source .bashrc
```

iv. 复制到其他节点

1. 复制安装目录

a) *scp -r /data/hadoop-2.6.5/ /data/scala-2.11.8/ /data/jdk1.8.0_111/ aboutyun@slave1:~/*

scp -r /data/hadoop-2.6.5/ /data/scala-2.11.8/ /data/jdk1.8.0_111/ aboutyun@slave2:~/

b) 登录到 slave1 和 slave2

sudo mkdir /data

sudo chmod 777 /data

mv hadoop-2.6.5/ scala-2.11.8/ jdk1.8.0_111/ /data

2. 复制 hadoop 日志目录

[aboutyun@master ~]$ scp -r ~/hadoop aboutyun@slave1:~/

[aboutyun@master ~]$ scp -r ~/hadoop aboutyun@slave2:~/

3. 复制环境变量

[aboutyun@master ~]$ scp -r ~/hadoop aboutyun@slave1:~/

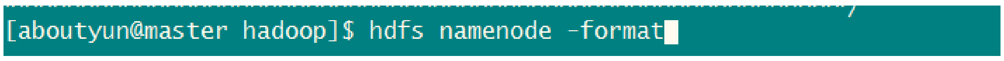
[aboutyun@master ~]$ scp -r ~/hadoop aboutyun@slave2:~/

登录 salve1 和 slave2

source .bashrc

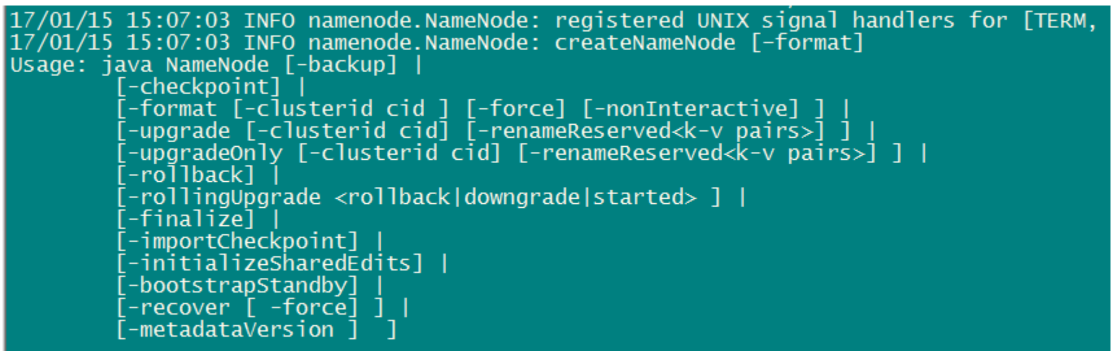
v. 验证

1. 格式化 hdfs



```
[aboutyun@master hadoop]$ hdfs namenode -format
```

问题：

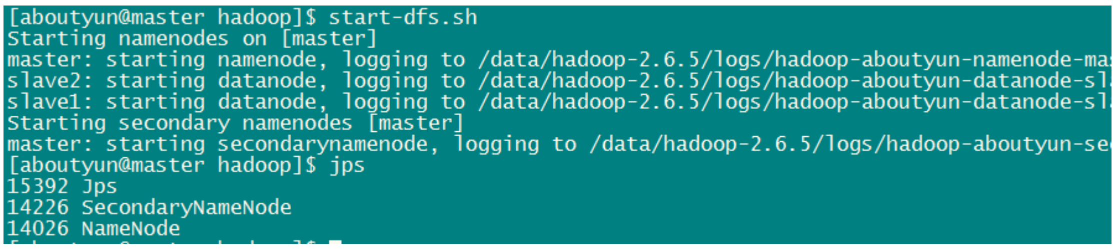


```
17/01/15 15:07:03 INFO namenode.NameNode: registered UNIX signal handlers for [TERM,
17/01/15 15:07:03 INFO namenode.NameNode: createNameNode [-format]
Usage: java NameNode [-backup] |
        [-checkpoint] |
        [-format [-clusterid cid ] [-force] [-nonInteractive] ] |
        [-upgrade [-clusterid cid] [-renameReserved<k-v pairs>] ] |
        [-upgradeOnly [-clusterid cid] [-renameReserved<k-v pairs>] ] |
        [-rollback] |
        [-rollingUpgrade <rollback|downgrade|started> ] |
        [-finalize] |
        [-importCheckpoint] |
        [-initializeSharedEdits] |
        [-bootstrapStandby] |
        [-recover [ -force] ] |
        [-metadataVersion ]  ]
```

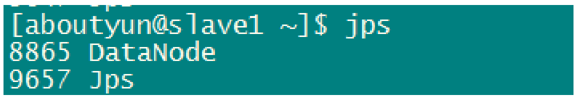
解决方法:

-format 的那个横线是中文的横线

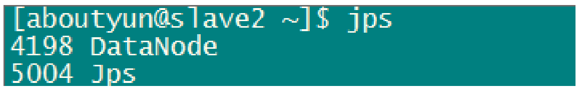
2. 启动 HDFS



```
[aboutyun@master hadoop]$ start-dfs.sh
Starting namenodes on [master]
master: starting namenode, logging to /data/hadoop-2.6.5/logs/hadoop-aboutyun-namenode-ma
slave2: starting datanode, logging to /data/hadoop-2.6.5/logs/hadoop-aboutyun-datanode-sl
slave1: starting datanode, logging to /data/hadoop-2.6.5/logs/hadoop-aboutyun-datanode-sl
Starting secondary namenodes [master]
master: starting secondarynamenode, logging to /data/hadoop-2.6.5/logs/hadoop-aboutyun-se
[aboutyun@master hadoop]$ jps
15392 Jps
14226 SecondaryNameNode
14026 NameNode
```

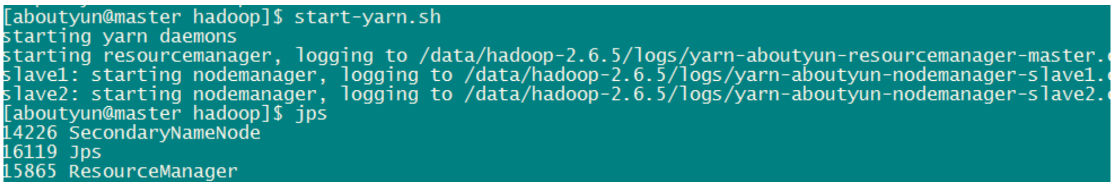


```
[aboutyun@slave1 ~]$ jps
8865 DataNode
9657 Jps
```

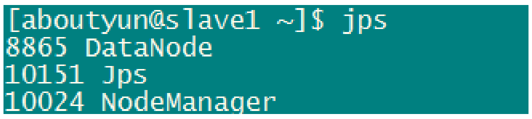


```
[aboutyun@slave2 ~]$ jps
4198 DataNode
5004 Jps
```

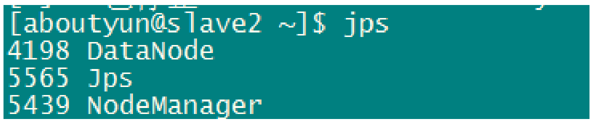
3. 启动 Yarn



```
[aboutyun@master hadoop]$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /data/hadoop-2.6.5/logs/yarn-aboutyun-resourcemanager-master.
slave1: starting nodemanager, logging to /data/hadoop-2.6.5/logs/yarn-aboutyun-nodemanager-slave1.
slave2: starting nodemanager, logging to /data/hadoop-2.6.5/logs/yarn-aboutyun-nodemanager-slave2.
[aboutyun@master hadoop]$ jps
14226 SecondaryNameNode
16119 Jps
15865 ResourceManager
```

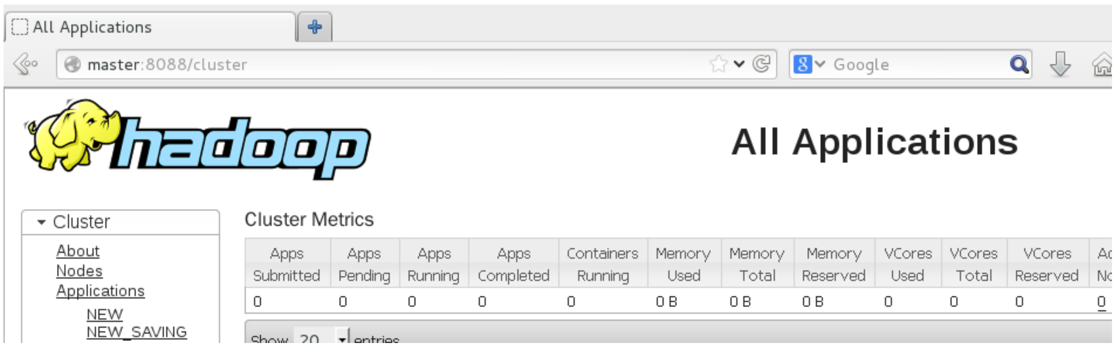


```
[aboutyun@slave1 ~]$ jps
8865 DataNode
10151 Jps
10024 NodeManager
```



```
[aboutyun@slave2 ~]$ jps
4198 DataNode
5565 Jps
5439 NodeManager
```

4. 访问 webUI



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
0	0	0	0	0	0 B	0 B	0 B	0	0	0

三、Spark 集群搭建

a) 解压

i. [aboutyun@master ~]$ tar -xvzf spark-1.6.3-bin-hadoop2.6.tgz -C /data/

b) 配置（需要复制 template）

${SPARK_HOME}/conf/spark-env.sh

${SPARK_HOME}/conf/slaves

${SPARK_HOME}/conf/spark-defaults.conf

i. spark-env.sh

```
JAVA_HOME=/data/jdk1.8.0_111
SCALA_HOME=/data/scala-2.11.8
SPARK_MASTER_IP=192.168.1.10
HADOOP_CONF_DIR=/data/hadoop-2.6.5/etc/hadoop
# shuffled 以及 RDD 的数据存放目录
SPARK_LOCAL_DIRS=/data/spark_data
# worker 端进程的工作目录
SPARK_WORKER_DIR=/data/spark_data/spark_works
```

创建目录：

```
[aboutyun@master conf]$ mkdir /data/spark_data
[aboutyun@master conf]$ mkdir /data/spark_data/spark_works
```

ii. slaves

```
master
```

slave1
slave2

iii. spark-defaults.conf

```
spark.master              spark://master:7077
spark.serializer              org.apache.spark.serializer.KryoSerializer
spark.eventLog.enabled             true
spark.eventLog.dir              file:///data/spark_data/history/event-log
spark.history.fs.logDirectory              file:///data/spark_data/history/spark-events
spark.eventLog.compress              true
```

创建目录：

```
[aboutyun@master conf]$ mkdir /data/spark_data/history
[aboutyun@master conf]$ mkdir /data/spark_data/history/event-log
[aboutyun@master conf]$ mkdir /data/spark_data/history/spark-events
```

c) 复制到其他节点

i. master

```
scp -r /data/spark* aboutyun@slave1:~/
scp -r /data/spark* aboutyun@slave2:~/
```

ii. slave1 和 slave2

```
mv ~/spark* /data
```

d) 设置环境变量

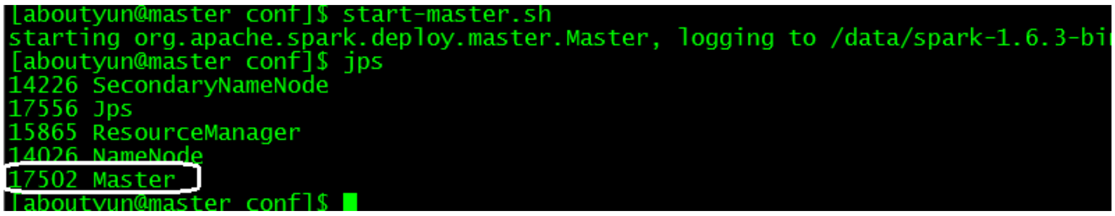
```
[aboutyun@master conf]$ vi ~/.bashrc
```

```
# spark
export SPARK_HOME=/data/spark-1.6.3-bin-hadoop2.6
export PATH=$SPARK_HOME/bin:$SPARK_HOME/sbin:$PATH
```

```
[aboutyun@master conf]$ source ~/.bashrc
```

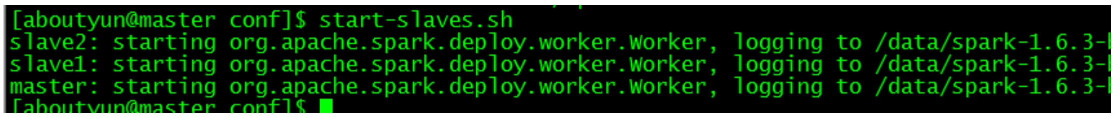
e) 验证

i. 启动 master

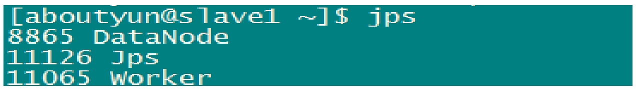


```
[aboutyun@master conf]$ start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /data/spark-1.6.3-bi
[aboutyun@master conf]$ jps
14226 SecondaryNameNode
17556 Jps
15865 ResourceManager
14026 NameNode
17502 Master
[aboutyun@master conf]$
```

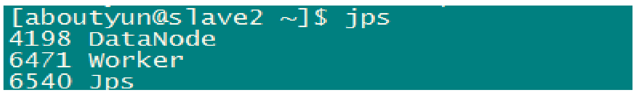
ii. 启动 slave



```
[aboutyun@master conf]$ start-slaves.sh
slave2: starting org.apache.spark.deploy.worker.Worker, logging to /data/spark-1.6.3-
slave1: starting org.apache.spark.deploy.worker.Worker, logging to /data/spark-1.6.3-
master: starting org.apache.spark.deploy.worker.Worker, logging to /data/spark-1.6.3-
[aboutyun@master conf]$
```

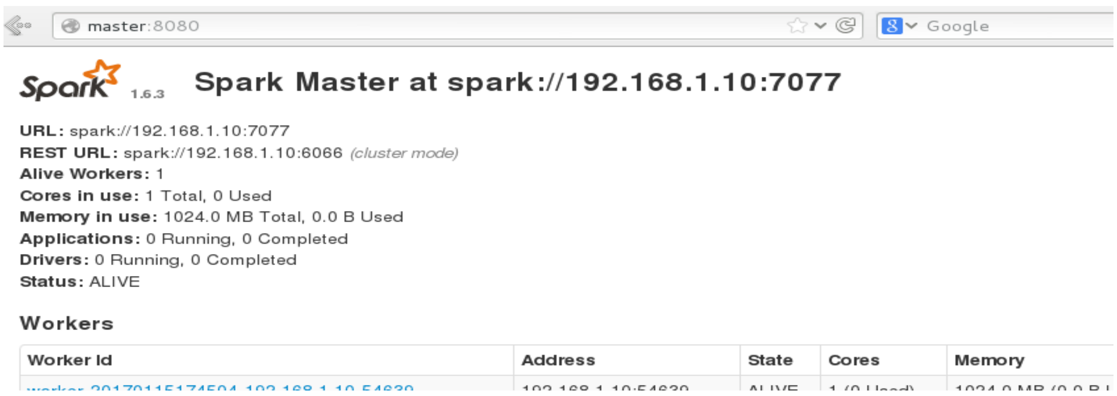


```
[aboutyun@slave1 ~]$ jps
8865 DataNode
11126 Jps
11065 Worker
```



```
[aboutyun@slave2 ~]$ jps
4198 DataNode
6471 Worker
6540 Jps
```

iii. 访问 WebUI



master:8080

Spark 1.6.3 **Spark Master at spark://192.168.1.10:7077**

URL: spark://192.168.1.10:7077
REST URL: spark://192.168.1.10:6066 *(cluster mode)*
Alive Workers: 1
Cores in use: 1 Total, 0 Used
Memory in use: 1024.0 MB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory

四、Hive 安装配置

a) centos 7 安装 mysql

hive 默认的元数据存在 derby 中，这样会存在很多弊端，所以我们用 MySQL

i. 下载 mysql 源

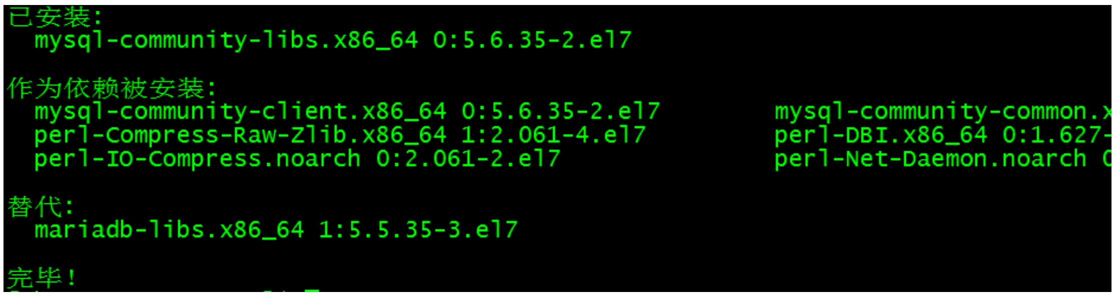
由于 cent os 7.0 使用 mariadb 作为默认的数据库，所以在 yum 源中并没有 mysql，需要下载

wget https://repo.mysql.com//mysql-community-release-el7-5.noarch.rpm

ii. 安装 mysql

sudo rpm -ivh mysql-community-release-el7-5.noarch.rpm

sudo yum -y install mysql-community-server



```
已安装:
  mysql-community-libs.x86_64 0:5.6.35-2.el7

作为依赖被安装:
  mysql-community-client.x86_64 0:5.6.35-2.el7        mysql-community-common.x
  perl-Compress-Raw-Zlib.x86_64 1:2.061-4.el7         perl-DBI.x86_64 0:1.627-
  perl-IO-Compress.noarch 0:2.061-2.el7               perl-Net-Daemon.noarch 0

替代:
  mariadb-libs.x86_64 1:5.5.35-3.el7

完毕!
```

iii. 初始化 mysql（**这一步必须要执行：否则之后登陆 mysql 时很有可能出错**）

sudo mysql_install_db --user=mysql

iv. 启动 mysql 服务

sudo systemctl start mysqld # 启动 mysql 服务

sudo systemctl enable mysqld # 开机自启 mysql

v. 修改 root 用户密码

mysqladmin -u root password '123' # 将 root 用户密码设置为 123

vi. 登录 mysql

mysql -uroot –p123 # 以 root 用户登录 mysql

vii. 创建 hiveUser 用户

create user 'hiveUser'@'%' identified by 'hive'; # 创建一个可以在任何机器上登录的 hive 用户，密码为 hive

viii. 赋予 hiveUser 用户权限(创建 create datebase '*hiveMetada'*)

grant all privileges on hiveMetada. to 'hiveUser'@'localhost' identified by 'hive';* # hiveUser

用户可以在任何机器上对 hiveMetada 库下的所有表进行任何操作

```
flush privileges; # 刷新权限
```

注意：在将 hive 元数据存入 mysql 中时，最好不要修改 mysql 的编码，否则在 hive 启动时会出现：Specified key was too long; max key length is 767 bytes 异常。在文章中保留是为了将这个问题记录下来。

b) 安装 Hive

i. 解压安装包

```
tar -xvzf apache-hive-1.2.1-bin.tar.gz -C /data/
mv /data/apache-hive-1.2.1-bin/ /data/hive-1.2.1/
```

ii. 添加 mysql 驱动

将 mysql 的 java 驱动 jar 包放入${hive_home}/lib 目录下

```
[aboutyun@master ~]$ cp mysql-connector-java-5.1.40-bin.jar /data/hive-1.2.1/lib/
[aboutyun@master ~]$ cd /data/hive-1.2.1/lib/
[aboutyun@master lib]$ chmod 664 mysql-connector-java-5.1.40-bin.jar
```

iii. 配置 Hive

修改${HIVE_HOME}/conf 目录下的配置文件，涉及到的配置文件有以下几个：

hive-env.sh

hive-site.xml

这两个文件从 template 文件拷贝得到

```
cp hive-env.sh.template hive-env.sh
cp hive-default.xml.template hive-site.xml
```

1. hive-env.sh

```
export HADOOP_HOME=/data/hadoop-2.6.5
export HIVE_HOME=/data/hive-1.2.1
export HIVE_CONF_DIR=/data/hive-1.2.1/conf
export HIVE_AUX_JARS_PATH=/data/hive-1.2.1/lib,/data/hive-1.2.1/hcatalog/share/hcatalog
export JAVA_HOME=/data/jdk1.8.0_111
export JAVA_LIBRARY_PATH=$HADOOP_HOME/lib/native
```

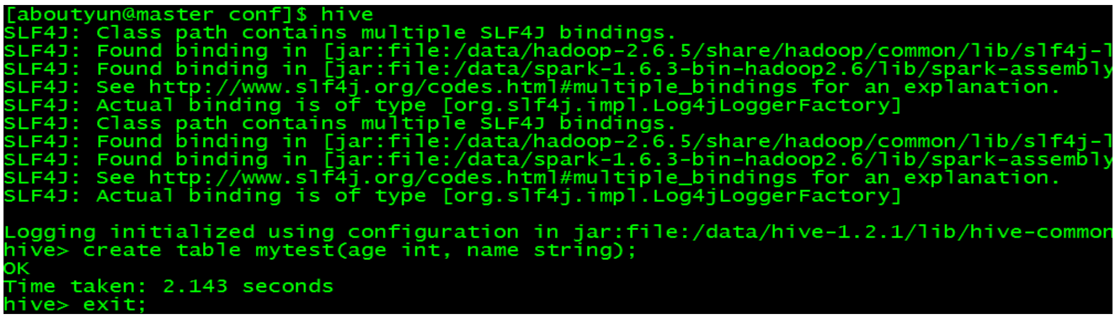
2. hive-site.xml

```
<configuration>
        <property>
                <name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://localhost:3306/hiveMetada?createDatabaseIfNotExist=true</value>
                <description>JDBC connect string for a JDBC metastore</description>
        </property>

        <property>
                <name>javax.jdo.option.ConnectionDriverName</name>
                <value>com.mysql.jdbc.Driver</value>
                <description>Driver class name for a JDBC metastore</description>
        </property>
        <property>
                <name>javax.jdo.option.ConnectionUserName</name>
                <value>hiveUser</value>
                <description>username to use against metastore database</description>
        </property>

        <property>
                <name>javax.jdo.option.ConnectionPassword</name>
                <value>hive</value>
                <description>password to use against metastore database</description>
        </property>

        <property>
                <name>hive.aux.jars.path</name>
                <value>file:///data/hive-1.2.1/lib/mysql-connector-java-5.1.40-bin.jar
                </value>
        </property>
</configuration>
```

iv. 添加环境变量

```
vi ~/.bashrc
# hive
export HIVE_HOME=/data/hive-1.2.1
export PATH=$HIVE_HOME/bin:$PATH
source ~/.bashrc
```

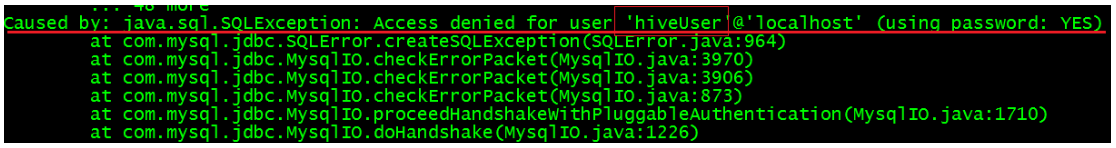
v. 启动验证



```
[aboutyun@master conf]$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/data/hadoop-2.6.5/share/hadoop/common/lib/slf4j-l
SLF4J: Found binding in [jar:file:/data/spark-1.6.3-bin-hadoop2.6/lib/spark-assembly
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/data/hadoop-2.6.5/share/hadoop/common/lib/slf4j-l
SLF4J: Found binding in [jar:file:/data/spark-1.6.3-bin-hadoop2.6/lib/spark-assembly
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/data/hive-1.2.1/lib/hive-common
hive> create table mytest(age int, name string);
OK
Time taken: 2.143 seconds
hive> exit;
```

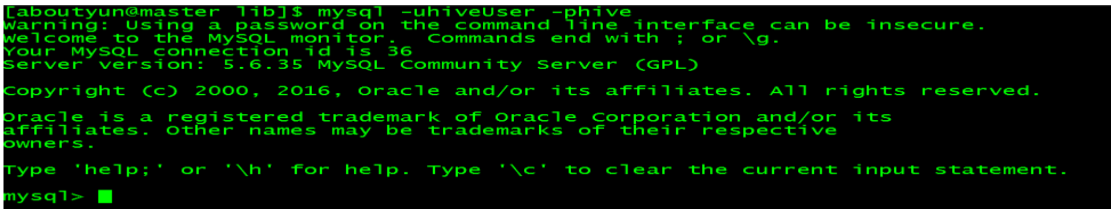
c) 问题与解决方法

i. mysql HiveUser 创建以及授权问题



```
        ... 48 more
Caused by: java.sql.SQLException: Access denied for user 'hiveUser'@'localhost' (using password: YES)
        at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:964)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3970)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3906)
        at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:873)
        at com.mysql.jdbc.MysqlIO.proceedHandshakeWithPluggableAuthentication(MysqlIO.java:1710)
        at com.mysql.jdbc.MysqlIO.doHandshake(MysqlIO.java:1226)
```

我的处理方式是，删除 hiveUser，重新注册、授权，在测试是否可以登录



```
[aboutyun@master lib]$ mysql -uhiveUser -phive
Warning: Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 36
Server version: 5.6.35 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, 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> 
```

问题就解决了

ii. 在 hive1.2 版本对应的是 hadoop2.6，Hadoop share 目录下存在老版本 jline

```
[ERROR] Terminal initialization failed; falling back to unsupported
java.lang.IncompatibleClassChangeError: Found class jline.Terminal, but interface was expected
	at jline.TerminalFactory.create(TerminalFactory.java:101)
	at jline.TerminalFactory.get(TerminalFactory.java:158)
	at jline.console.ConsoleReader.<init>(ConsoleReader.java:229)
	at jline.console.ConsoleReader.<init>(ConsoleReader.java:221)
	at jline.console.ConsoleReader.<init>(ConsoleReader.java:209)
	at org.apache.hadoop.hive.cli.CliDriver.setupConsoleReader(CliDriver.java:787)
	at org.apache.hadoop.hive.cli.CliDriver.executeDriver(CliDriver.java:721)
	at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:681)
	at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:621)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
	at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:498)
	at org.apache.hadoop.util.RunJar.run(RunJar.java:221)
	at org.apache.hadoop.util.RunJar.main(RunJar.java:136)

Exception in thread "main" java.lang.IncompatibleClassChangeError: Found class jline.Terminal,
but interface was expected
	at jline.console.ConsoleReader.<init>(ConsoleReader.java:230)
	at jline.console.ConsoleReader.<init>(ConsoleReader.java:221)
	at jline.console.ConsoleReader.<init>(ConsoleReader.java:209)
	at org.apache.hadoop.hive.cli.CliDriver.setupConsoleReader(CliDriver.java:787)
	at org.apache.hadoop.hive.cli.CliDriver.executeDriver(CliDriver.java:721)
	at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:681)
	at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:621)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
	at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:498)
	at org.apache.hadoop.util.RunJar.run(RunJar.java:221)
	at org.apache.hadoop.util.RunJar.main(RunJar.java:136)
```

删除--$HADOOP_HOME/share/hadoop/yarn/lib/jline-0.9.94.jar 即可

五、Kafka 集群搭建

a) zookeeper 安装

i. 解压

```
tar -xvzf zookeeper-3.4.6.tar.gz -C /data/
```

ii. 配置

${ZOOKEEPER_HOME}/conf/zoo.cfg

```
cp zoo_sample.cfg zoo.cfg

# 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=/data/zk_data
# the port at which the clients will connect
clientPort=2181

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

这儿解释下格式为 server.X=host:port1:port2 的意思，X 表示当前 host 所运行的服务的 zookeeper 服务的 id（在接下来填写 myid 时需要用到），port1 表示 zookeeper 中的 follower 连接到 leader 的端口号，port2 表示 leadership 时所用的端口号。注意：需要手动去创建 dataDir 所配置的/data/zk_data 目录（*mkdir -p /data/zk_data*）

iii. 填写 myid

在 zoo.cfg 配置文件中的 dataDir 目录（在这儿是/data/data_zk）下创建 myid 文件，文件内容为 zoo.cfg 中 master 所对应的 server.X

iv. 复制到其他节点

```
scp -r /data/zookeeper-3.4.6/ /data/zk_data   aboutyun@slave1:/data
scp -r /data/zookeeper-3.4.6/ /data/zk_data   aboutyun@slave2:/data

slave1：
[aboutyun@slave1 ~]$ echo “2” > /data/zk_data/myid
slave2：
[aboutyun@slave1 ~]$ echo “3” > /data/zk_data/myid
```

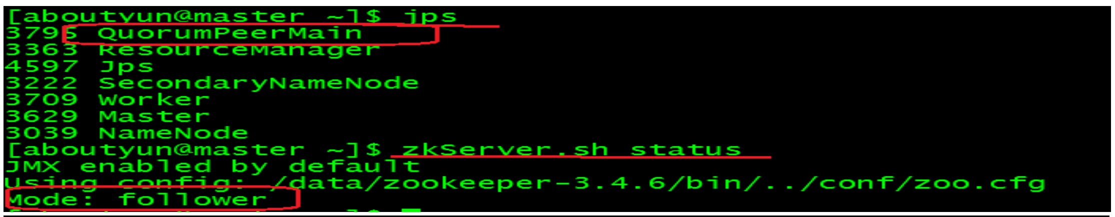
v. 添加到环境变量

分别登录到 master，slave1，slave2 将变量加入到~/.bashrc

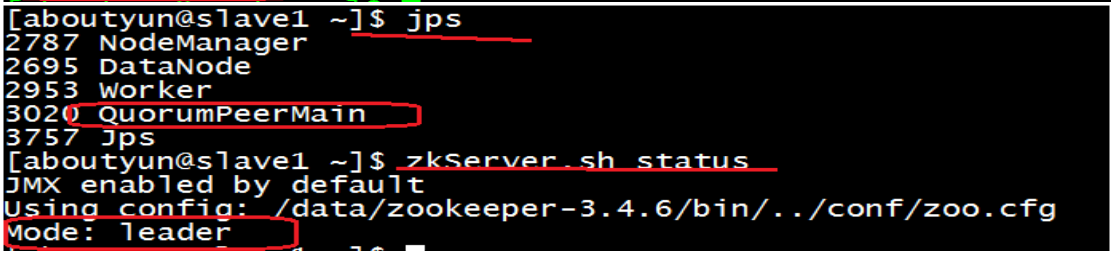
```
# zookeeper
export ZOOKEEPER_HOME=/data/zookeeper-3.4.6
export PATH=$ZOOKEEPER_HOME/bin:$PATH
source ~/.bashrc
```

vi. 启动验证

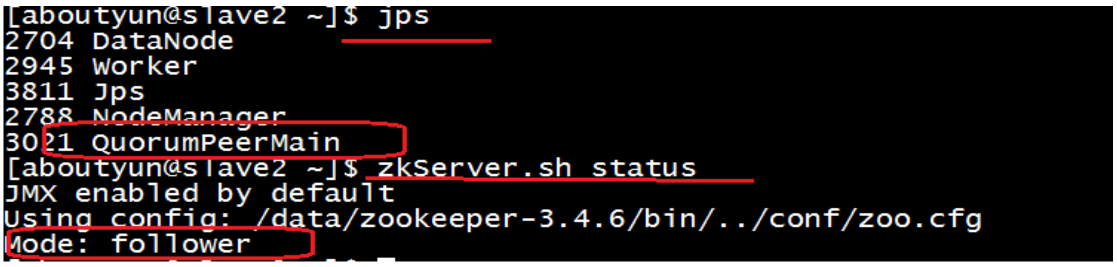
zkServer.sh start（三台机器）



```
[aboutyun@master ~]$ jps
3795 QuorumPeerMain
3363 ResourceManager
4597 Jps
3222 SecondaryNameNode
3709 Worker
3629 Master
3039 NameNode
[aboutyun@master ~]$ zkServer.sh status
JMX enabled by default
Using config: /data/zookeeper-3.4.6/bin/../conf/zoo.cfg
Mode: follower
```



```
[aboutyun@slave1 ~]$ jps
2787 NodeManager
2695 DataNode
2953 Worker
3020 QuorumPeerMain
3757 Jps
[aboutyun@slave1 ~]$ zkServer.sh status
JMX enabled by default
Using config: /data/zookeeper-3.4.6/bin/../conf/zoo.cfg
Mode: leader
```



```
[aboutyun@slave2 ~]$ jps
2704 DataNode
2945 Worker
3811 Jps
2788 NodeManager
3021 QuorumPeerMain
[aboutyun@slave2 ~]$ zkServer.sh status
JMX enabled by default
Using config: /data/zookeeper-3.4.6/bin/../conf/zoo.cfg
Mode: follower
```

vii. 问题与解决方法

安装 zookeeper 时候，可以查看进程启动，但是状态显示报错：Error contacting service. It is probably not running

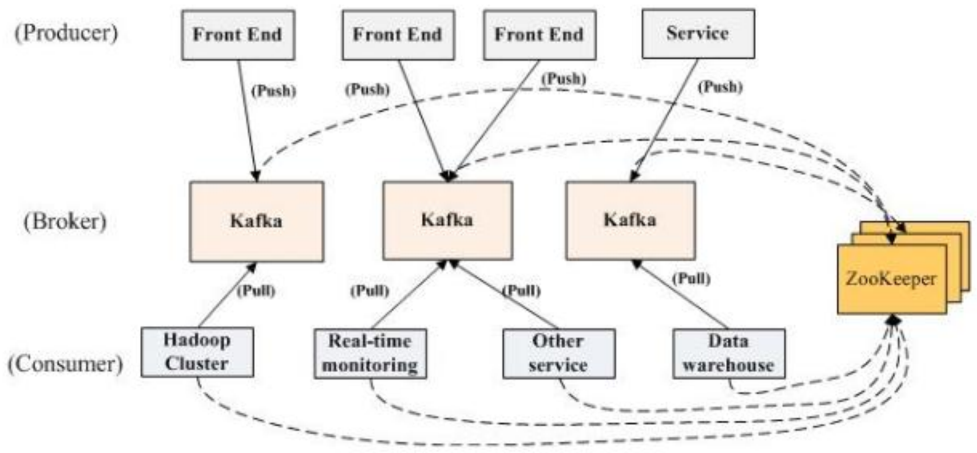
关闭防火墙

a) sudo systemctl status firewalld.service

b) sudo systemctl stop firewalld.service

c) sudo systemctl disable firewalld.service

b) kafka 集群搭建



i. 解压

```
tar -xvzf kafka_2.11-0.9.0.1.tgz -C /data/
```

ii. 配置

```
${KAFA_HOME}/config/server.properties
```

必须要配置的是这三个参数：broker.id、log.dirs、zookeeper.connect：
broker.id 表示当前 broker 的 id，要求是唯一的非负数
log.dirs 表示 kafka 日志的存放目录
zookeeper.connect 表示连接的 zookeeper 的地址

```
mkdir -p /data/kafka-logs
```

```
broker.id=0
log.dirs=/data/kafka-logs
zookeeper.connect=master:2181,slave1:2181,slave2:2181
```

iii. 复制到其他节点

```
scp -r /data/kafka_2.11-0.9.0.1/ /data/kafka-logs/ aboutyun@slave1:/data
scp -r /data/kafka_2.11-0.9.0.1/ /data/kafka-logs/ aboutyun@slave2:/data
```

在 slave1 机器上将 server.properties 配置文件的 broker.id 值改为 1 在 slave2 机器上将 server.properties 配置文件的 broker.id 值改为 2

iv. 添加环境变量

在 master，slave1，slave2 上的~/.bashrc 添加

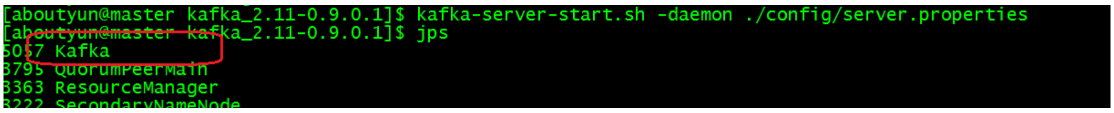
```
export KAFKA_HOME=/data/kafka_2.11-0.9.0.1
export PATH=$KAFKA_HOME/bin:$PATH
```

```
source ~/.bashrc
```

v. 启动验证

在 master，slave1，slave2 上分别执行 kafka 启动命令

```
cd $KAFKA_HOME
kafka-server-start.sh -daemon ./config/server.properties
```



```
[aboutyun@master kafka_2.11-0.9.0.1]$ kafka-server-start.sh -daemon ./config/server.properties
[aboutyun@master kafka_2.11-0.9.0.1]$ jps
5057 Kafka
3795 QuorumPeerMain
3363 ResourceManager
3222 SecondaryNameNode
```

如果每台机器上都成功启动了 kafka 这个进程，说明我们搭建成功。如果发现某台机器上没有 kafka 这个进程，可以将 kafka 的启动命令去掉参数-daemon（加上的话表示后台启动），这样可以直接在屏幕上看到错误信息

c) kafka 实例

i. 创建 topic

#创建一个有 3 个 partition、1 个副本的 test topic

kafka-topics.sh --zookeeper master:2181,slave1:2181,slave2:2181 --create --topic test --replication-factor 1 --partitions 3

ii. 创建 producer、创建 consumer

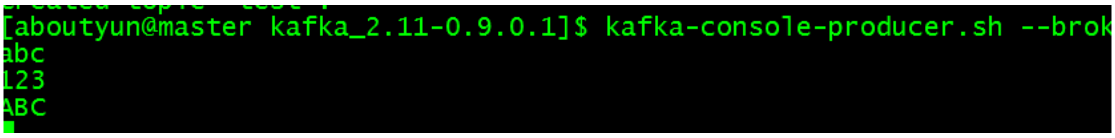
kafka-console-producer.sh --broker-list master:9092,slave1:9092,slave2:9092 --topic test

打开一个新窗口创建 consumer

kafka-console-consumer.sh --zookeeper master:2181,slave1:2181,slave2:2181 --topic test --from-beginning

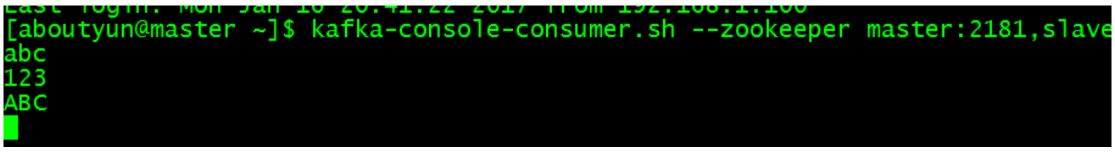
iii. producer 产生消息，consumer 接受消息并消费

1. 生产者（producer）



```
Created topic "test".
[aboutyun@master kafka_2.11-0.9.0.1]$ kafka-console-producer.sh --brok
abc
123
ABC
```

2. 消费者（consumer）



```
Last login: Mon Jan 16 20:41:22 2017 from 192.168.1.100
[aboutyun@master ~]$ kafka-console-consumer.sh --zookeeper master:2181,slave
abc
123
ABC
```

六、flume 集群搭建

组件名称	功能介绍
Agent代理	使用JVM 运行Flume。每台机器运行一个agent，但是可以在一个agent中包含多个sources和sinks。
Client客户端	生产数据，运行在一个独立的线程。
Source源	从Client收集数据，传递给Channel。
Sink接收器	从Channel收集数据，进行相关操作，运行在一个独立线程。
Channel通道	连接 sources 和 sinks ，这个有点像一个队列。
Events事件	传输的基本数据负载。

a) Flume 安装

i. 解压

tar -xvzf apache-flume-1.6.0-bin.tar.gz -C /data/

mv apache-flume-1.6.0-bin/ flume-1.6.0/

ii. 配置

${FLUME_HOME}/conf/ flume-env.sh

cp flume-env.sh.template flume-env.sh

修改 JAVA_HOME

export JAVA_HOME= /data/jdk1.8.0_111

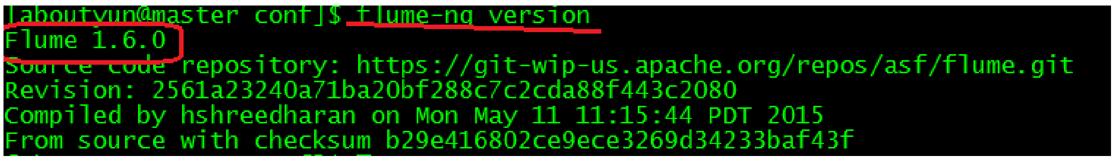
iii. 环境变量

echo -e "# flume\nexport FLUME_HOME=/data/flume-1.6.0\nexport PATH=\$FLUME_HOME/bin:\$PATH" >> ~/.bashrc

source ~/.bashrc

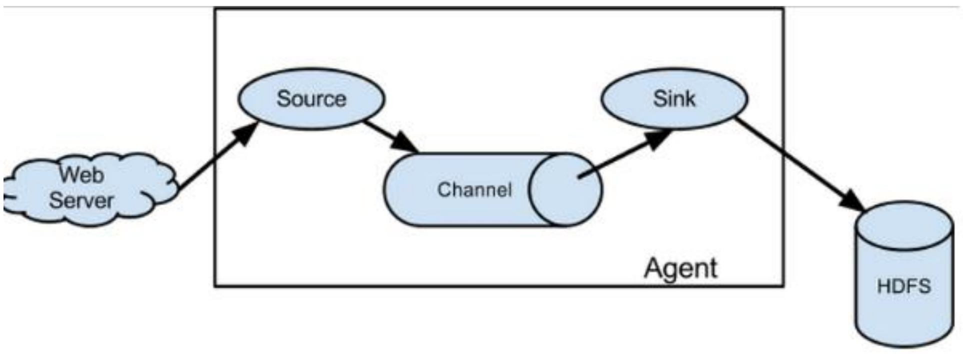
iv. 验证安装

flume-ng version



```
[aboutyun@master conf]$ flume-ng version
Flume 1.6.0
Source code repository: https://git-wip-us.apache.org/repos/asf/flume.git
Revision: 2561a23240a71ba20bf288c7c2cda88f443c2080
Compiled by hshreedharan on Mon May 11 11:15:44 PDT 2015
From source with checksum b29e416802ce9ece3269d34233baf43f
```

b) Flume 应用实例



i. 单节点的 agent

1. 添加配置文件- single_agent.conf

```
cd $FLUME_HOME/conf
vim single_agent.conf
```

添加内容

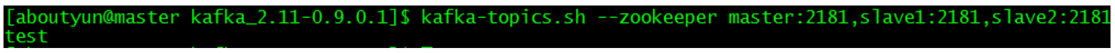
```
# agent 的名称为 a1
a1.sources = source1
a1.channels = channel1
a1.sinks = sink1

# set source
a1.sources.source1.type = spooldir
a1.sources.source1.spoolDir=/data/aboutyunlog
a1.sources.source1.fileHeader = false

# set sink
a1.sinks.sink1.type = org.apache.flume.sink.kafka.KafkaSink
# a1.sinks.sink1.kafka.bootstrap.servers = master:9092,slave1:9092,slave2:9092
a1.sinks.sink1.brokerList=master:9092,slave1:9092,slave2:9092
a1.sinks.sink1.topic= aboutyunlog

# set channel
a1.channels.channel1.type = file
a1.channels.channel1.checkpointDir = /data/flume_data/checkpoint
a1.channels.channel1.dataDirs= /data/flume_data/data
# bind
a1.sources.source1.channels = channel1
a1.sinks.sink1.channel = channel1
```

ii. 创建需要的目录

```
mkdir -p /data/aboutyunlog
mkdir -p /data/flume_data/checkpoint
mkdir -p /data/flume_data/data
```

iii. 查看 kafka 现有的 topic

kafka-topics.sh --zookeeper master:2181,slave1:2181,slave2:2181 –list



```
[aboutyun@master kafka_2.11-0.9.0.1]$ kafka-topics.sh --zookeeper master:2181,slave1:2181,slave2:2181
test
```

iv.　在 kafka 上创建名为 aboutyunlog 的 topic

kafka-topics.sh --zookeeper master:2181,slave1:2181,slave2:2181 --create --topic aboutyunlog --replication-factor 1 --partitions 3

v.　启动 flume(**terminal1**)

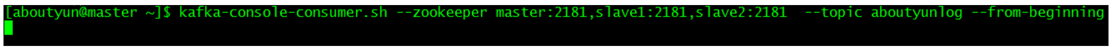
flume-ng agent --conf-file /data/flume-1.6.0/conf/single_agent.conf --name a1 -Dflume.root.logger=INFO,console

日志输出…

vi.　创建一个 kafka 的 consumer (**terminal2**)

kafka-console-consumer.sh --zookeeper master:2181,slave1:2181,slave2:2181 --topic aboutyunlog --from-beginning

这条命令的意思是说创建 aboutyunlog 这个 topic 下的消费者，消费时从最开始的一条信息开始消费。



```
[aboutyun@master ~]$ kafka-console-consumer.sh --zookeeper master:2181,slave1:2181,slave2:2181  --topic aboutyunlog --from-beginning
█
```

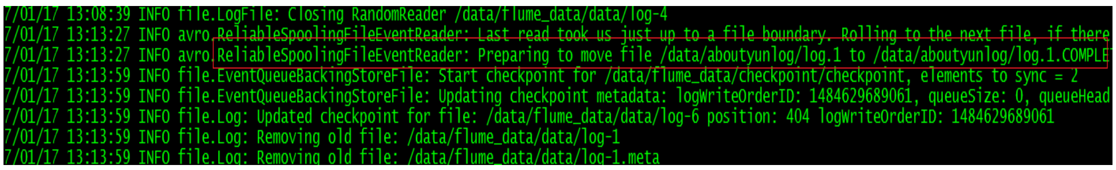
上图说明该消费者创建成功，由于本地/data/aboutyunlog 目录下没有新文件加入，造成 aboutyunlog 这个 topic 没有信息输入，所以消费者没有得到一条信息

vii.　添加文件到 flume source 目录(**terminal3**)

[aboutyun@master ~]$ echo -e "apache-flume-1.6.0-bin.tar.gz \napache-hive-1.2.1-bin.tar.gz \nhadoop-2.6.5.tar.gz \njdk-8u111-linux-x64.tar.gz" >> log.1

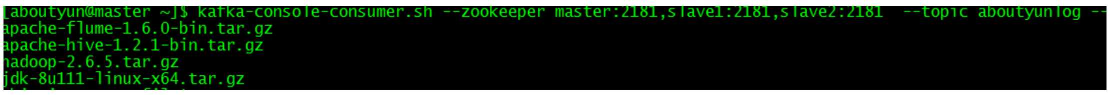
[aboutyun@master ~]$ mv log.1 /data/aboutyunlog/

将一个文件添加到 flume 的监控目录之后，flume 会将改文件重命名为 filename.COMPLETED。这一信息可以在启动 flume 的 shell 界面中看到



```
7/01/17 13:08:39 INFO file.LogFile: Closing RandomReader /data/flume_data/data/log-4
7/01/17 13:13:27 INFO avro.ReliableSpoolingFileEventReader: Last read took us just up to a file boundary. Rolling to the next file, if there
7/01/17 13:13:27 INFO avro.ReliableSpoolingFileEventReader: Preparing to move file /data/aboutyunlog/log.1 to /data/aboutyunlog/log.1.COMPLE
7/01/17 13:13:59 INFO file.EventQueueBackingStoreFile: Start checkpoint for /data/flume_data/checkpoint/checkpoint, elements to sync = 2
7/01/17 13:13:59 INFO file.EventQueueBackingStoreFile: Updating checkpoint metadata: logWriteOrderID: 1484629689061, queueSize: 0, queueHead
7/01/17 13:13:59 INFO file.Log: Updated checkpoint for file: /data/flume_data/data/log-6 position: 404 logWriteOrderID: 1484629689061
7/01/17 13:13:59 INFO file.Log: Removing old file: /data/flume_data/data/log-1
7/01/17 13:13:59 INFO file.Log: Removing old file: /data/flume_data/data/log-1.meta
```

viii.　查看 consumer



```
[aboutyun@master ~]$ kafka-console-consumer.sh --zookeeper master:2181,slave1:2181,slave2:2181  --topic aboutyunlog --
apache-flume-1.6.0-bin.tar.gz
apache-hive-1.2.1-bin.tar.gz
hadoop-2.6.5.tar.gz
jdk-8u111-linux-x64.tar.gz
```