Flink 消费 Kafka 数据

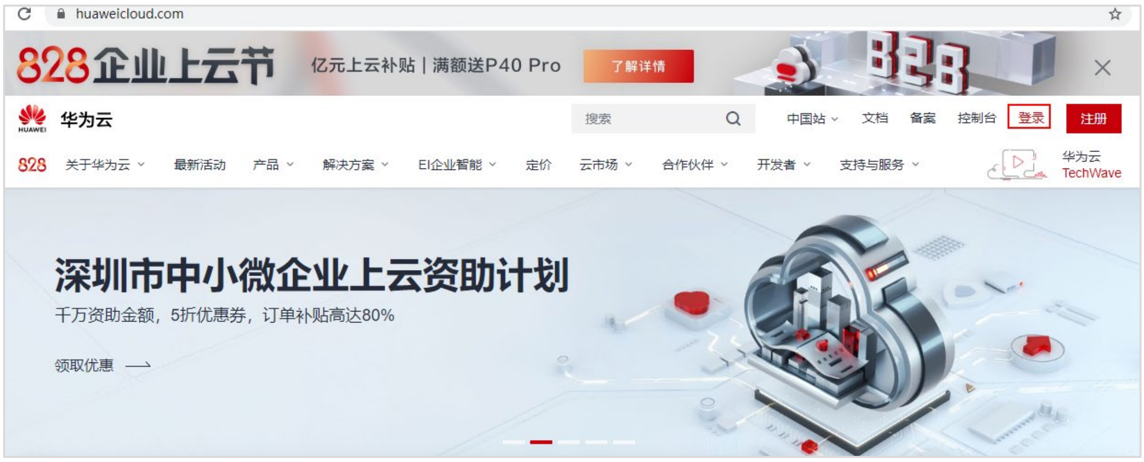
1. 实验环境

- 虚拟机数量：3
- 系统版本：Centos 7.6（鲲鹏 aarch64 架构）
- Hadoop 版本：hadoop-2.7.7
- JDK 版本：jdk-1.8.0
- Flink 版本：flink-1.8.0
- Zookeeper 版本：zookeeper-3.4.12
- Kafka 版本：kafka_2.10-0.8.2.1

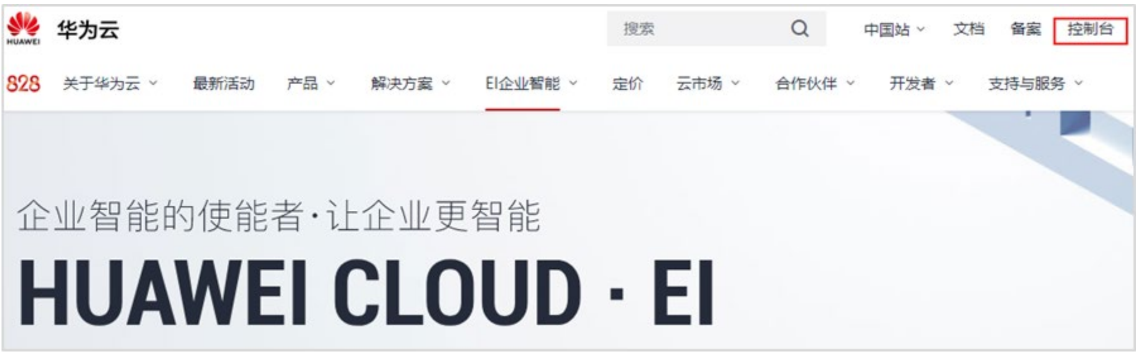
2. 资源购买

浏览器登录华为云

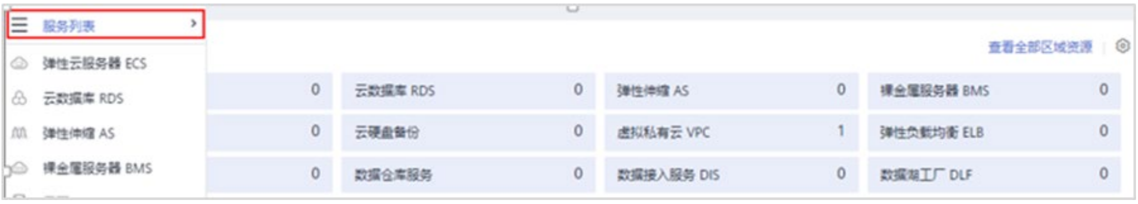
打开华为云地址：https://www.huaweicloud.com/，点击“登录”，输入用户名、密码，如下图：



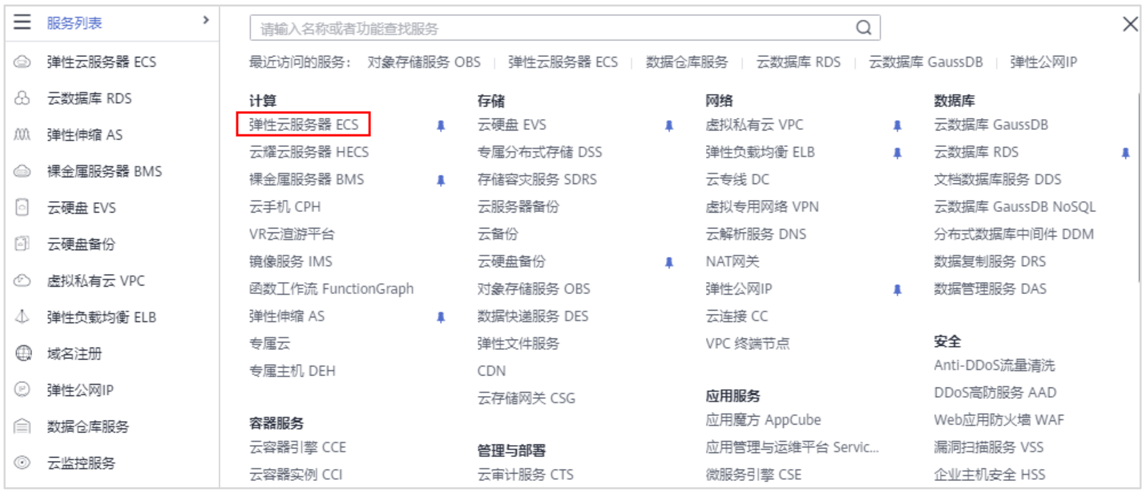
点击“控制台”，如下图：



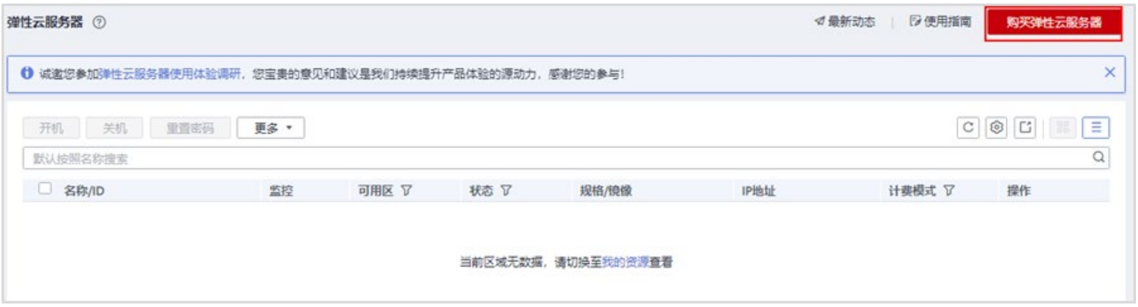
点击“服务列表”，如下图：



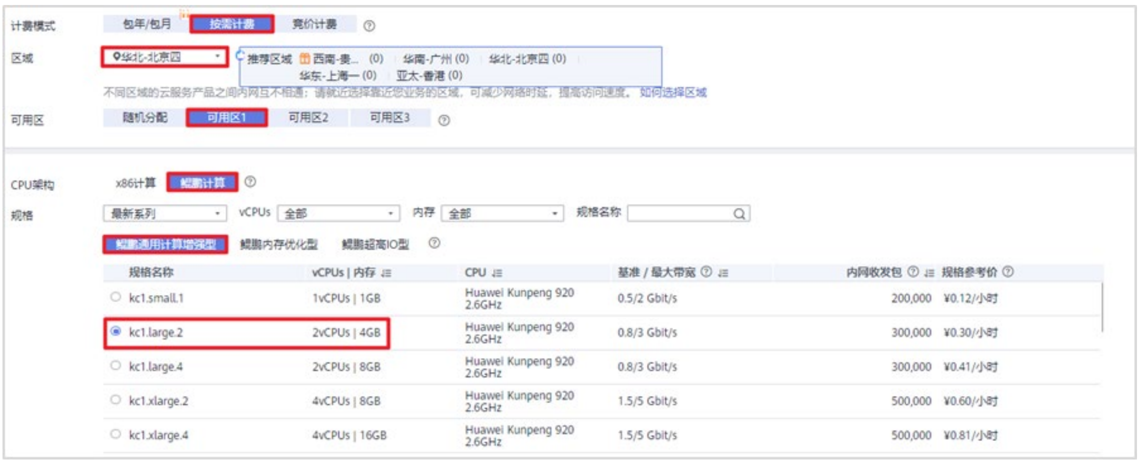
选择“弹性云服务器 ECS”，如下图：



选择“买弹性云服务器 ECS”，如下图：


弹性云服务器
最新动态
使用指南
购买弹性云服务器
诚邀您参加弹性云服务器使用体验调研，您宝贵的意见和建议是我们持续提升产品体验的源动力，感谢您的参与！
开机
关机
重置密码
更多
默认按照名称搜索
名称/ID
监控
可用区
状态
规格/镜像
IP地址
计费模式
操作
当前区域无数据，请切换至我的资源查看

选择“按需计费”，“可用区 2”，CPU 架构“鲲鹏计算”，选择“鲲鹏通用计算增强型”，2vCPUs|4GB，如下图：



计费模式 包年/包月 [✓ 按需计费] 竞价计费

区域 华北-北京四 推荐区域 西南-贵... (0) 华南-广州 (0) 华北-北京四 (0) 华东-上海一 (0) 亚太-香港 (0)

不同区域的云服务产品之间内网互不相通；请就近选择靠近您业务的区域，可减少网络时延，提高访问速度。 如何选择区域

可用区 随机分配 [✓ 可用区1] 可用区2 可用区3

CPU架构 x86计算 [✓ 鲲鹏计算]

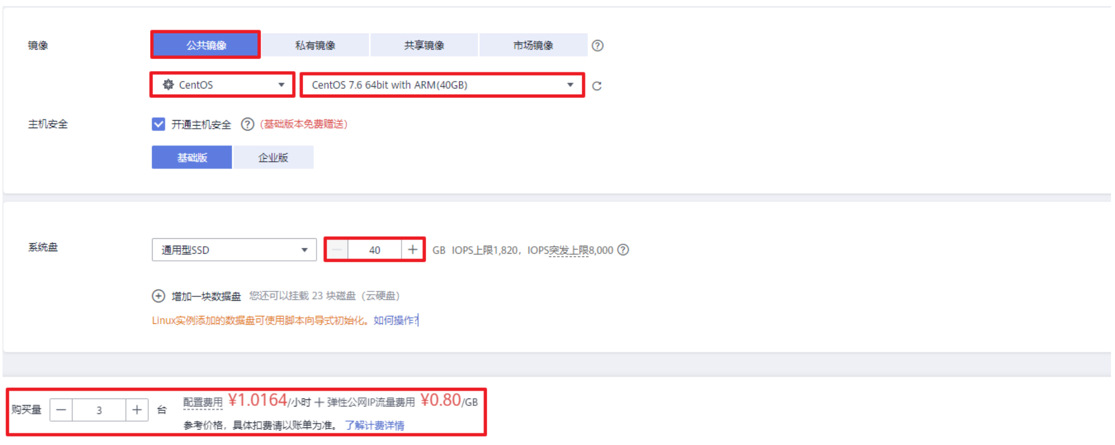
规格 最新系列 vCPUs 全部 内存 全部 规格名称

[✓ 鲲鹏通用计算增强型] 鲲鹏内存优化型 鲲鹏超高IO型

	规格名称	vCPUs | 内存	CPU	基准 / 最大带宽	内网收发包	规格参考价
○	kc1.small.1	1vCPUs | 1GB	Huawei Kunpeng 920 2.6GHz	0.5/2 Gbit/s	200,000	¥0.12/小时
●	kc1.large.2	2vCPUs | 4GB	Huawei Kunpeng 920 2.6GHz	0.8/3 Gbit/s	300,000	¥0.30/小时
○	kc1.large.4	2vCPUs | 8GB	Huawei Kunpeng 920 2.6GHz	0.8/3 Gbit/s	300,000	¥0.41/小时
○	kc1.xlarge.2	4vCPUs | 8GB	Huawei Kunpeng 920 2.6GHz	1.5/5 Gbit/s	500,000	¥0.60/小时
○	kc1.xlarge.4	4vCPUs | 16GB	Huawei Kunpeng 920 2.6GHz	1.5/5 Gbit/s	500,000	¥0.81/小时

配置操作系统和磁盘

选择“公共镜像”，CentOS7.6，系统盘建议配置 40GB，购买数量 3 台，点击“网络配置”，如下图：



镜像 [✓ 公共镜像] 私有镜像 共享镜像 市场镜像

CentOS CentOS 7.6 64bit with ARM(40GB)

主机安全 ☑ 开通主机安全 (基础版本免费赠送)

[✓ 基础版] 企业版

系统盘 通用型SSD 40 GB IOPS上限1,820，IOPS突发上限8,000

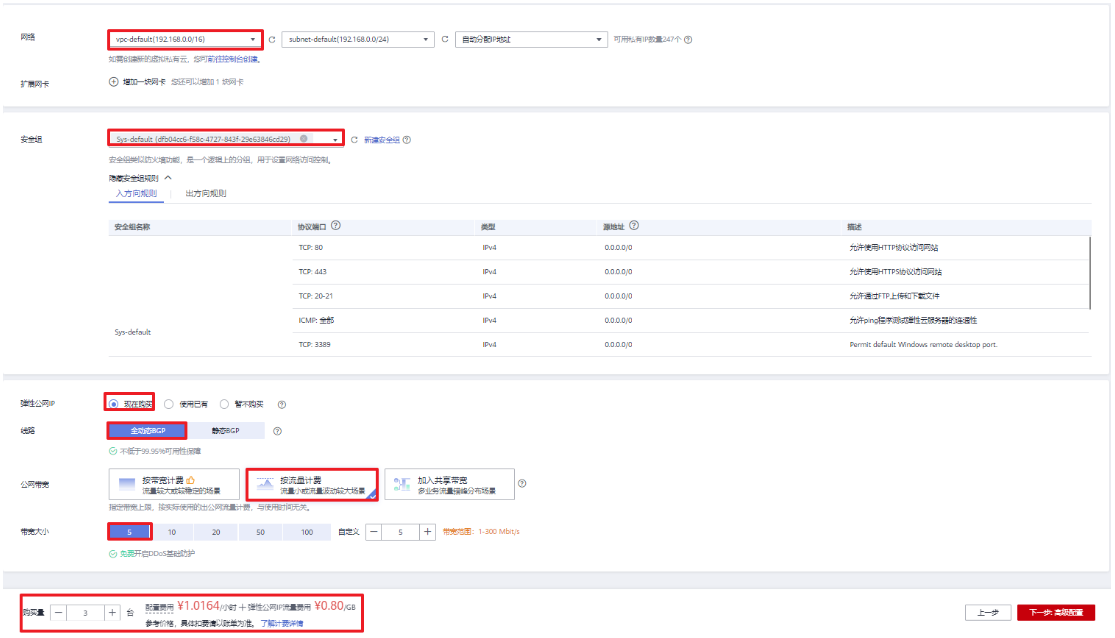
⊕ 增加一块数据盘 您还可以挂载 23 块磁盘（云硬盘）

Linux实例添加的数据盘可使用脚本向导式初始化。如何操作?

购买量 3 台 配置费用 ¥1.0164/小时 + 弹性公网IP流量费用 ¥0.80/GB

参考价格，具体扣费请以账单为准。 了解计费详情

配置网络，网络选择“vpc-default”，安全组选择“Sys-default”，“现在购买”，选择“全动态 BGP”，“按流量计算”，“5M”，点击“高级配置”，如下图：



网络 vpc-default(192.168.0.0/16) subnet-default(192.168.0.0/24) 自动分配IP地址 可用私有IP数量247个

如需创建新的虚拟私有云，您可前往控制台创建。

扩展网卡 ⊕ 增加一块网卡 您还可以增加 1 块网卡

安全组 Sys-default (dfb04cc6-f58c-4727-843f-29e63846cd29) 新建安全组

安全组类似防火墙功能，是一个逻辑上的分组，用于设置网络访问控制。

隐藏安全组规则

入方向规则 | 出方向规则

安全组名称	协议端口	类型	源地址	描述
Sys-default	TCP: 80	IPv4	0.0.0.0/0	允许使用HTTP协议访问网站
	TCP: 443	IPv4	0.0.0.0/0	允许使用HTTPS协议访问网站
	TCP: 20-21	IPv4	0.0.0.0/0	允许通过FTP上传和下载文件
	ICMP: 全部	IPv4	0.0.0.0/0	允许ping程序测试弹性云服务器的连通性
	TCP: 3389	IPv4	0.0.0.0/0	Permit default Windows remote desktop port.

弹性公网IP [✓ 现在购买] 使用已有 暂不购买

线路 [✓ 全动态BGP] 静态BGP

不低于99.95%可用性保障

公网带宽 按带宽计费 流量较大或较稳定的场景 [✓ 按流量计费 流量小或流量波动较大场景] 加入共享带宽 多业务流量错峰分布场景

指定带宽上限，按实际使用的出公网流量计费，与使用时间无关。

带宽大小 [✓ 5] 10 20 50 100 自定义 5 带宽范围：1-300 Mbit/s

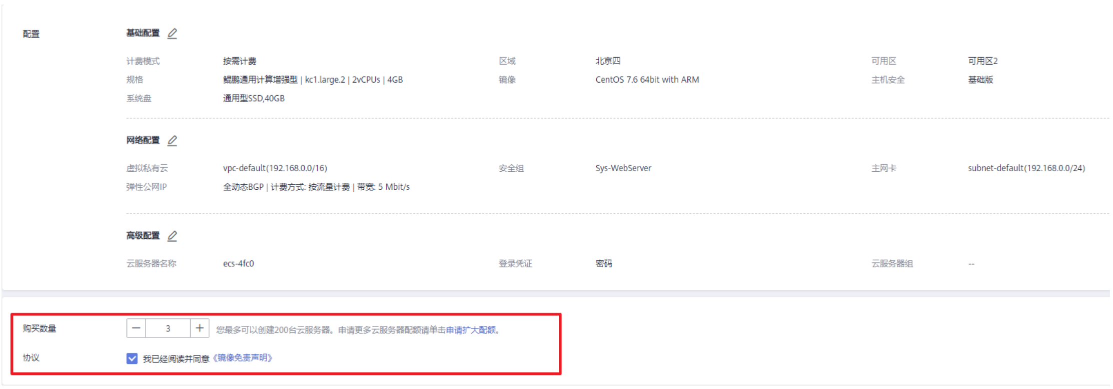
免费开启DDoS基础防护

购买量 3 台 配置费用 ¥1.0164/小时 + 弹性公网IP流量费用 ¥0.80/GB

参考价格，具体扣费请以账单为准。 了解计费详情

上一步 下一步: 高级配置

配置密码，自定义云服务器名称，自行设置 root 登录密码，云备份选择“暂不购买”，点击“确认配置”，点中“我已经阅读并同意”，点击“立即购买”，如下图：



配置

基础配置

计费模式	按需计费	区域	北京四	可用区	可用区2
规格	鲲鹏通用计算增强型 | kc1.large.2 | 2vCPUs | 4GB	镜像	CentOS 7.6 64bit with ARM	主机安全	基础版
系统盘	通用型SSD,40GB				

网络配置

虚拟私有云	vpc-default(192.168.0.0/16)	安全组	Sys-WebServer	主网卡	subnet-default(192.168.0.0/24)
弹性公网IP	全动态BGP | 计费方式: 按流量计费 | 带宽: 5 Mbit/s				

高级配置

云服务器名称	ecs-4fc0	登录凭证	密码	云服务器组	--

购买数量	3 您最多可以创建200台云服务器。申请更多云服务器配额请单击申请扩大配额。
协议	✓ 我已经阅读并同意《镜像免责声明》

注：本次需购买 3 个 ECS，每个 ECS 规格相同，其中 1 个主节点、2 个从节点。

点击“我已经阅读并同意”，可以点击“返回云服务器列表”，创建过程需要等待几分钟。

任务提交成功！

您的弹性云服务器已经开始创建。

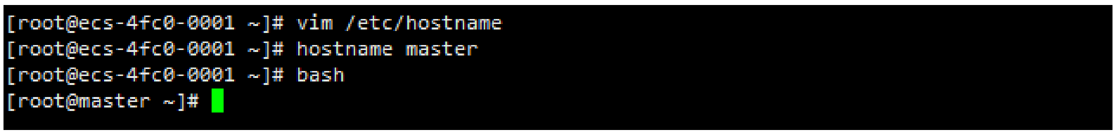
3. 开始实验

说明：若服务器已安装 JDK 和 Hadoop，3.1 可跳过

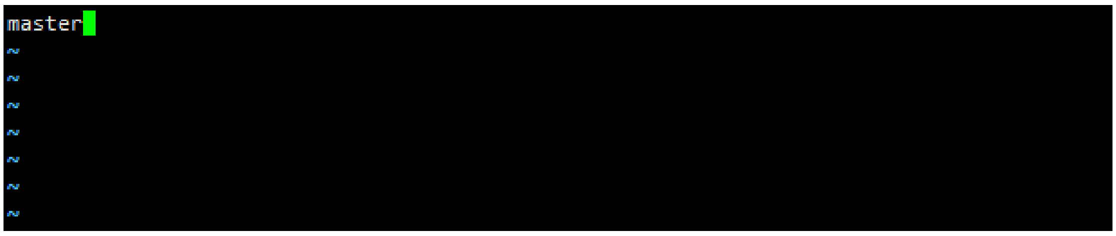
3.1 Hadoop 安装（参照普开 Hadoop 安装部署）

3.1.1 修改主机名（以 master 主机为例）

slave01，slave02 主机相同操作修改 hostname 即可



```
[root@ecs-4fc0-0001 ~]# vim /etc/hostname
[root@ecs-4fc0-0001 ~]# hostname master
[root@ecs-4fc0-0001 ~]# bash
[root@master ~]# 
```

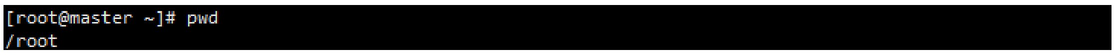


```
master
~
~
~
~
~
~
~
```

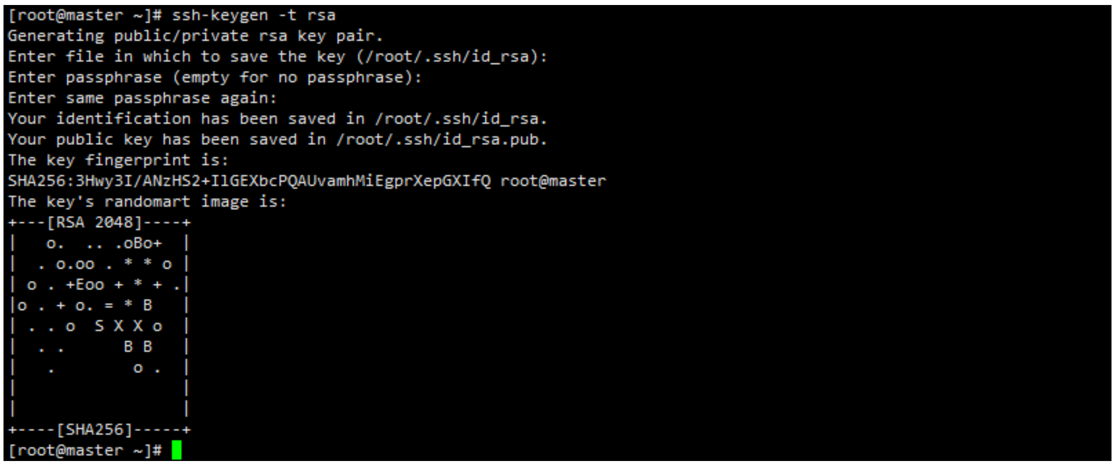
3.1.2 免密钥登录配置

3.1.2.1 在终端生成密钥，命令如下（一路按回车完成密钥生成）

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



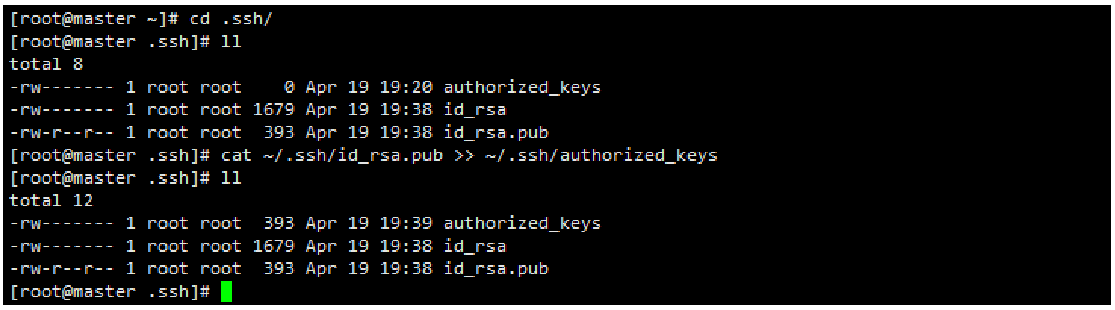
```
[root@master ~]# pwd
/root
```



```
[root@master ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:3Hwy3I/ANzHS2+IlGEXbcPQAUvamhMiEgprXepGXIfQ root@master
The key's randomart image is:
+---[RSA 2048]----+
|   o.  .. .oBo+  |
|  . o.oo . * * o |
| o . +Eoo + * + .|
|o . + o. = * B   |
| . . o  S X X o  |
|  . .      B B   |
|   .        o .  |
|                 |
|                 |
+----[SHA256]-----+
[root@master ~]# 
```

3.1.2.2 进行复制公钥文件

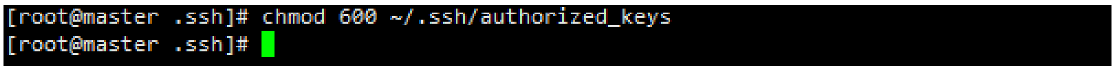
```
cd .ssh/
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```



```
[root@master ~]# cd .ssh/
[root@master .ssh]# ll
total 8
-rw------- 1 root root    0 Apr 19 19:20 authorized_keys
-rw------- 1 root root 1679 Apr 19 19:38 id_rsa
-rw-r--r-- 1 root root  393 Apr 19 19:38 id_rsa.pub
[root@master .ssh]# cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
[root@master .ssh]# ll
total 12
-rw------- 1 root root  393 Apr 19 19:39 authorized_keys
-rw------- 1 root root 1679 Apr 19 19:38 id_rsa
-rw-r--r-- 1 root root  393 Apr 19 19:38 id_rsa.pub
[root@master .ssh]# 
```

3.1.2.3 修改 authorized_keys 文件的权限，命令如下

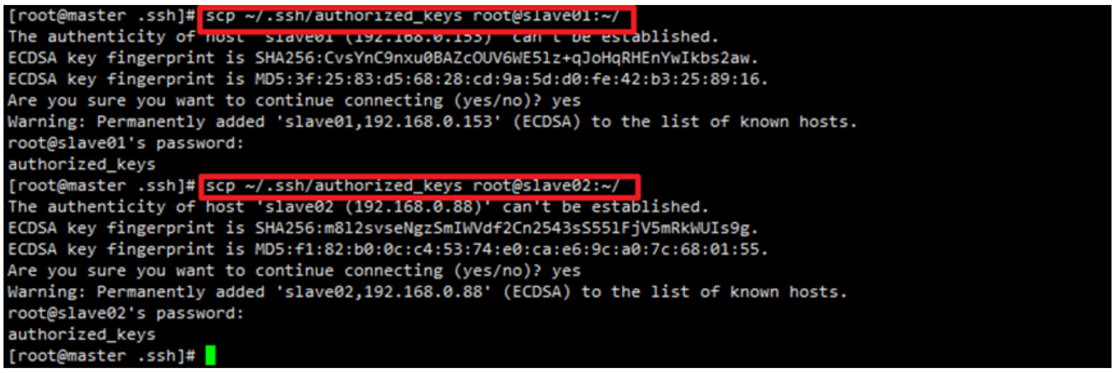
```
chmod 600 ~/.ssh/authorized_keys
```



```
[root@master .ssh]# chmod 600 ~/.ssh/authorized_keys
[root@master .ssh]# 
```

3.1.2.4 将 authorized_keys 文件复制到 slave01、slave02 节点，命令如下

```
scp ~/.ssh/authorized_keys root@slave01:~/
scp ~/.ssh/authorized_keys root@slave02:~/
```

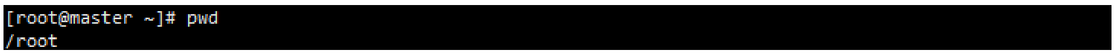


```
[root@master .ssh]# scp ~/.ssh/authorized_keys root@slave01:~/
The authenticity of host 'slave01 (192.168.0.153)' can't be established.
ECDSA key fingerprint is SHA256:CvsYnC9nxu0BAZcOUV6WE5lz+qJoHqRHEnYwIkbs2aw.
ECDSA key fingerprint is MD5:3f:25:83:d5:68:28:cd:9a:5d:d0:fe:42:b3:25:89:16.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave01,192.168.0.153' (ECDSA) to the list of known hosts.
root@slave01's password:
authorized_keys
[root@master .ssh]# scp ~/.ssh/authorized_keys root@slave02:~/
The authenticity of host 'slave02 (192.168.0.88)' can't be established.
ECDSA key fingerprint is SHA256:m812svseNgzSmIWVdf2Cn2543sS551FjV5mRkWUIs9g.
ECDSA key fingerprint is MD5:f1:82:b0:0c:c4:53:74:e0:ca:e6:9c:a0:7c:68:01:55.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave02,192.168.0.88' (ECDSA) to the list of known hosts.
root@slave02's password:
authorized_keys
[root@master .ssh]#
```

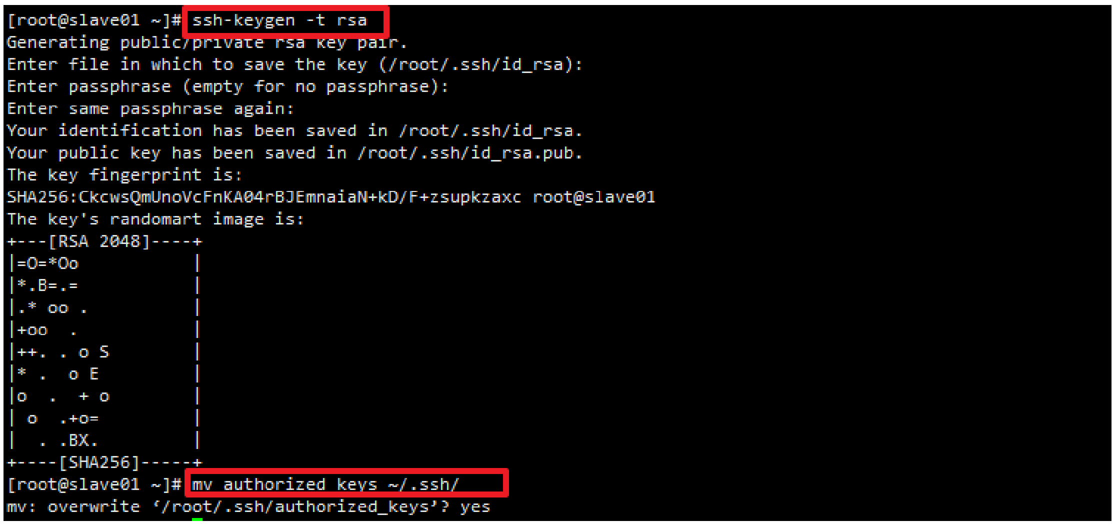
3.1.2.5 在 slave01、slave02 节点分别执行如下操作

ssh-keygen -t rsa(一直回车)

mv authorized_keys ~/.ssh/（输入 yes）

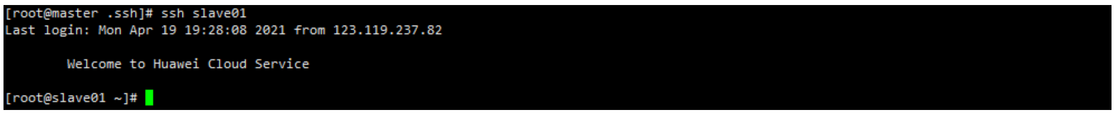


```
[root@master ~]# pwd
/root
```



```
[root@slave01 ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:CkcwsQmUnoVcFnKA04rBJEmnaiaN+kD/F+zsupkzaxc root@slave01
The key's randomart image is:
+---[RSA 2048]----+
|=O=*Oo           |
|*.B=.=           |
|.* oo .          |
|+oo  .           |
|++. . o S        |
|* .  o E         |
|o  .  + o        |
| o  .+o=         |
|  . .BX.         |
+----[SHA256]-----+
[root@slave01 ~]# mv authorized_keys ~/.ssh/
mv: overwrite ‘/root/.ssh/authorized_keys’? yes
```

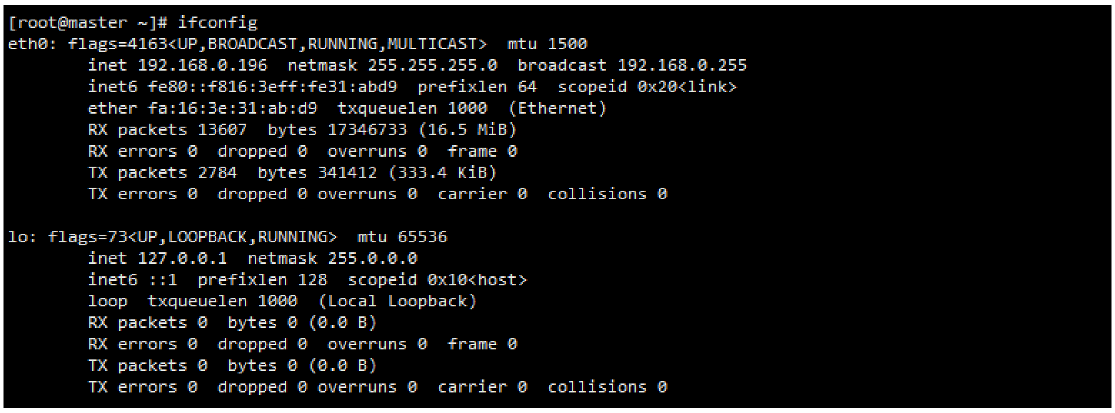
3.1.2.6 如果出现下图的内容表示免密钥配置成功



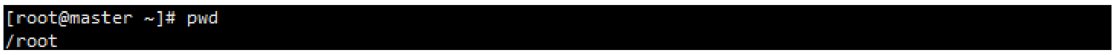
```
[root@master .ssh]# ssh slave01
Last login: Mon Apr 19 19:28:08 2021 from 123.119.237.82

        Welcome to Huawei Cloud Service

[root@slave01 ~]#
```

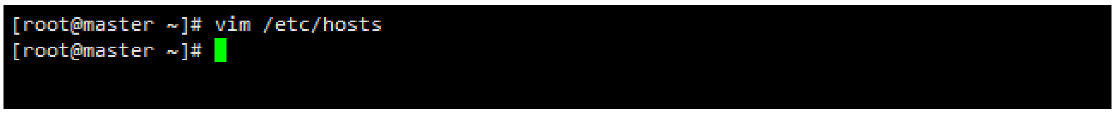
3.1.3 配置 hosts 列表

3.1.3.1 先分别在各服务器中运行 ifconfig 命令，获得当前节点的 ip 地址，如下图是 master 的 ip 地址


```
[root@master ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.196  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::f816:3eff:fe31:abd9  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:31:ab:d9  txqueuelen 1000  (Ethernet)
        RX packets 13607  bytes 17346733 (16.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2784  bytes 341412 (333.4 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 1000  (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
```

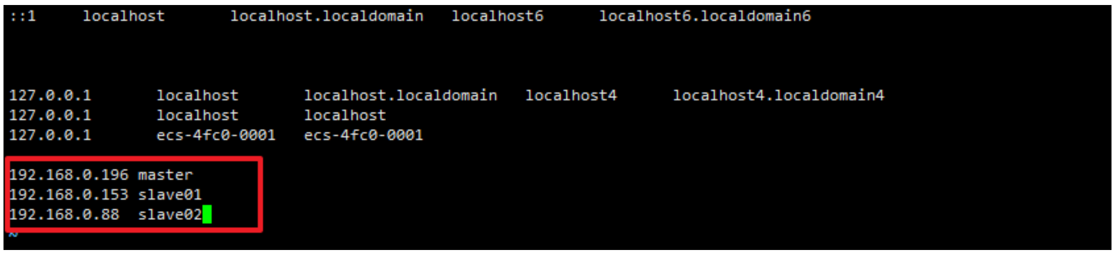
3.1.3.2 编辑主机名列表文件


```
[root@master ~]# pwd
/root
```

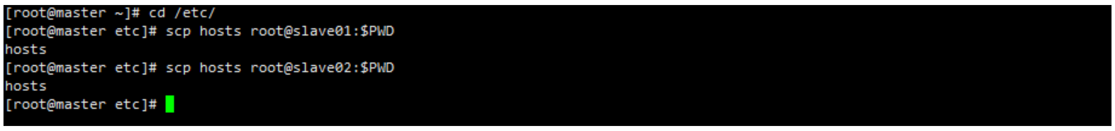

```
[root@master ~]# vim /etc/hosts
[root@master ~]# 
```

3.1.3.3 将下面三行添加到/etc/hosts 文件中，保存退出

注意：这里 master 节点对应 IP 地址是 192.168.0.196，slave01 对应的 IP 是 192.168.0.153，slave02 对应的 IP 是 192.168.0.88，而自己在做配置时，需要将 IP 地址改成自己的 master、slave01 和 slave02 对应的 IP 地址。

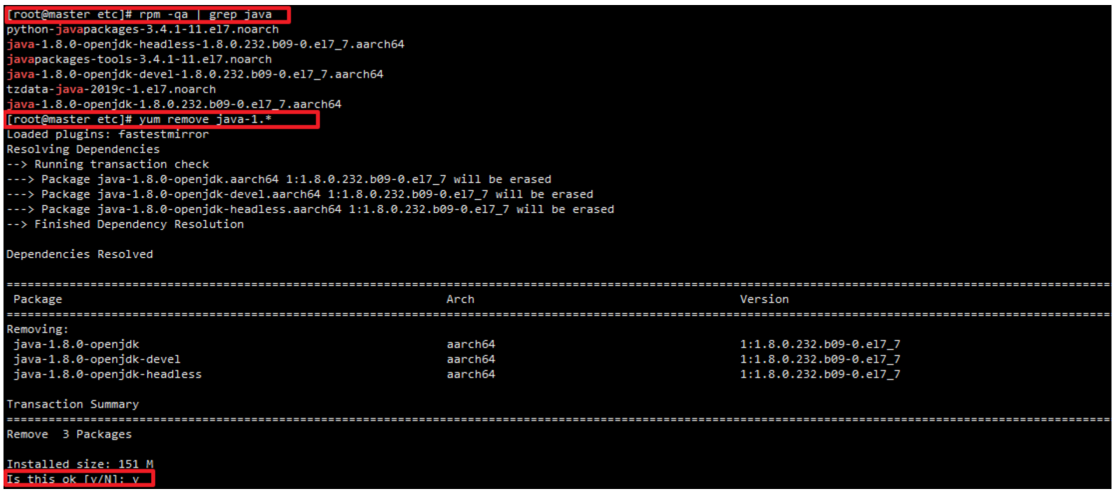

```
::1     localhost       localhost.localdomain   localhost6      localhost6.localdomain6


127.0.0.1       localhost       localhost.localdomain   localhost4      localhost4.localdomain4
127.0.0.1       localhost       localhost
127.0.0.1       ecs-4fc0-0001   ecs-4fc0-0001

192.168.0.196 master
192.168.0.153 slave01
192.168.0.88  slave02
```

3.1.3.4 将 hosts 文件分发到 slave01，slave02 节点


```
[root@master ~]# cd /etc/
[root@master etc]# scp hosts root@slave01:$PWD
hosts
[root@master etc]# scp hosts root@slave02:$PWD
hosts
[root@master etc]# 
```

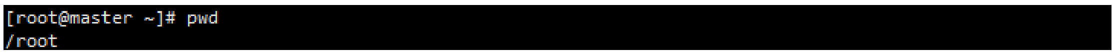
3.1.4 安装 JDK(在三台节点分别操作此步骤)

3.1.4.1 删除系统自带的 jdk(如若出现下图效果，说明系统自带 java，需要先卸载)

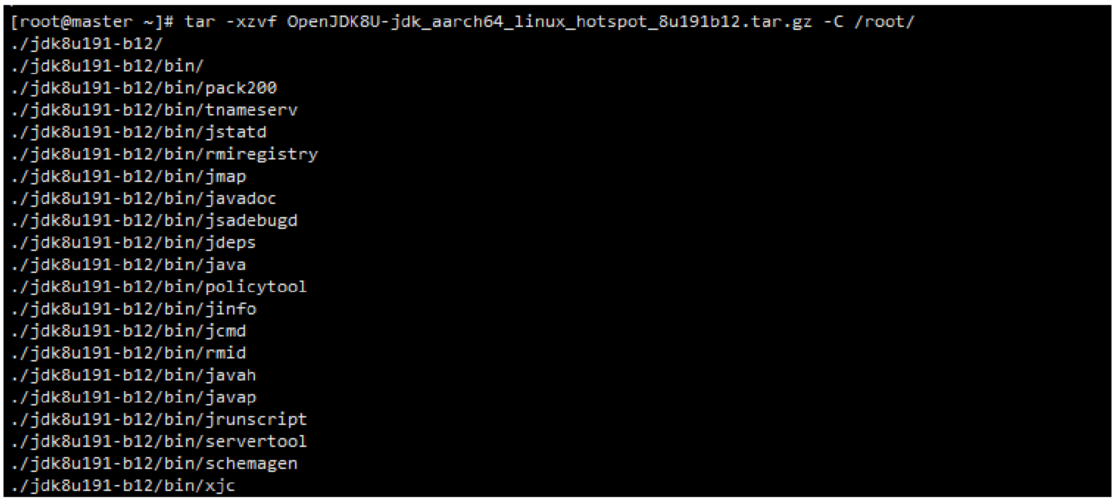


```
[root@master etc]# rpm -qa | grep java
python-javapackages-3.4.1-11.el7.noarch
java-1.8.0-openjdk-headless-1.8.0.232.b09-0.el7_7.aarch64
javapackages-tools-3.4.1-11.el7.noarch
java-1.8.0-openjdk-devel-1.8.0.232.b09-0.el7_7.aarch64
tzdata-java-2019c-1.el7.noarch
java-1.8.0-openjdk-1.8.0.232.b09-0.el7_7.aarch64
[root@master etc]# yum remove java-1.*
Loaded plugins: fastestmirror
Resolving Dependencies
--> Running transaction check
---> Package java-1.8.0-openjdk.aarch64 1:1.8.0.232.b09-0.el7_7 will be erased
---> Package java-1.8.0-openjdk-devel.aarch64 1:1.8.0.232.b09-0.el7_7 will be erased
---> Package java-1.8.0-openjdk-headless.aarch64 1:1.8.0.232.b09-0.el7_7 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

==============================================================================================================================================================
 Package                                                        Arch                                      Version
==============================================================================================================================================================
Removing:
 java-1.8.0-openjdk                                             aarch64                                   1:1.8.0.232.b09-0.el7_7
 java-1.8.0-openjdk-devel                                       aarch64                                   1:1.8.0.232.b09-0.el7_7
 java-1.8.0-openjdk-headless                                    aarch64                                   1:1.8.0.232.b09-0.el7_7

Transaction Summary
==============================================================================================================================================================
Remove  3 Packages

Installed size: 151 M
Is this ok [y/N]: y
```

3.1.4.2 解压 JDK

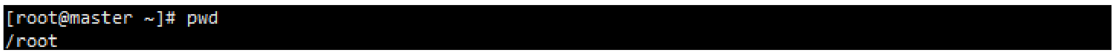


```
[root@master ~]# pwd
/root
```

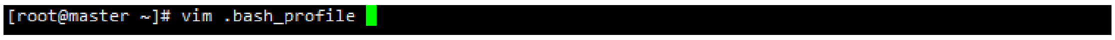


```
[root@master ~]# tar -xzvf OpenJDK8U-jdk_aarch64_linux_hotspot_8u191b12.tar.gz -C /root/
./jdk8u191-b12/
./jdk8u191-b12/bin/
./jdk8u191-b12/bin/pack200
./jdk8u191-b12/bin/tnameserv
./jdk8u191-b12/bin/jstatd
./jdk8u191-b12/bin/rmiregistry
./jdk8u191-b12/bin/jmap
./jdk8u191-b12/bin/javadoc
./jdk8u191-b12/bin/jsadebugd
./jdk8u191-b12/bin/jdeps
./jdk8u191-b12/bin/java
./jdk8u191-b12/bin/policytool
./jdk8u191-b12/bin/jinfo
./jdk8u191-b12/bin/jcmd
./jdk8u191-b12/bin/rmid
./jdk8u191-b12/bin/javah
./jdk8u191-b12/bin/javap
./jdk8u191-b12/bin/jrunscript
./jdk8u191-b12/bin/servertool
./jdk8u191-b12/bin/schemagen
./jdk8u191-b12/bin/xjc
```

3.1.4.3 配置用户环境变量（在三台节点分别操作此步骤）



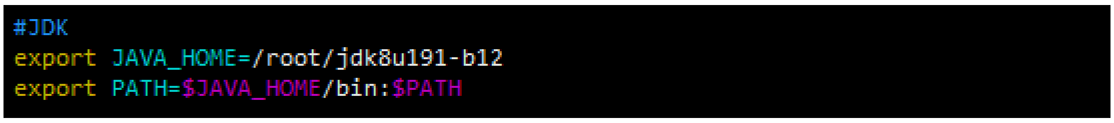
```
[root@master ~]# pwd
/root
```



```
[root@master ~]# vim .bash_profile
```

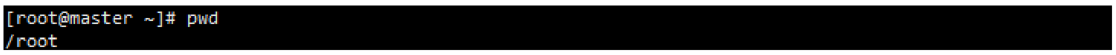
添加以下内容

```
#JDK
export JAVA_HOME=/root/jdk8u191-b12
export PATH=$JAVA_HOME/bin:$PATH
```

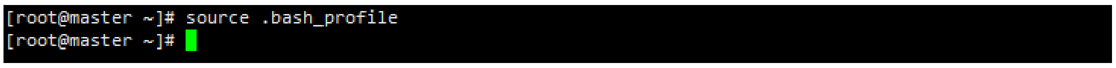


```
#JDK
export JAVA_HOME=/root/jdk8u191-b12
export PATH=$JAVA_HOME/bin:$PATH
```

使环境变量生效

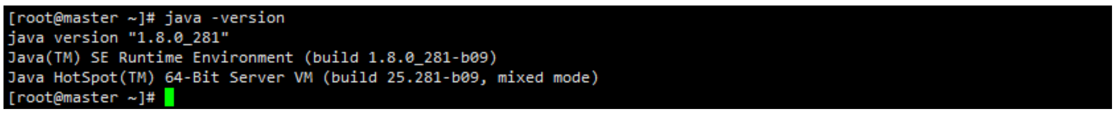


```
[root@master ~]# pwd
/root
```



```
[root@master ~]# source .bash_profile
[root@master ~]#
```

3.1.4.4 查看 java 是否配置成功

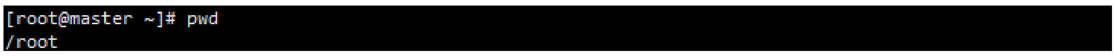


```
[root@master ~]# java -version
java version "1.8.0_281"
Java(TM) SE Runtime Environment (build 1.8.0_281-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.281-b09, mixed mode)
[root@master ~]#
```

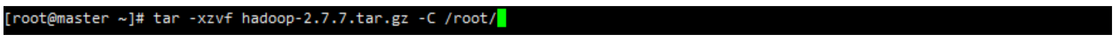
3.1.5 安装部署 Hadoop 集群

说明：每个节点上的 Hadoop 配置基本相同，在 master 节点操作，然后复制到 slave01、slave02 两个节点。

3.1.5.1 解压 Hadoop 文件



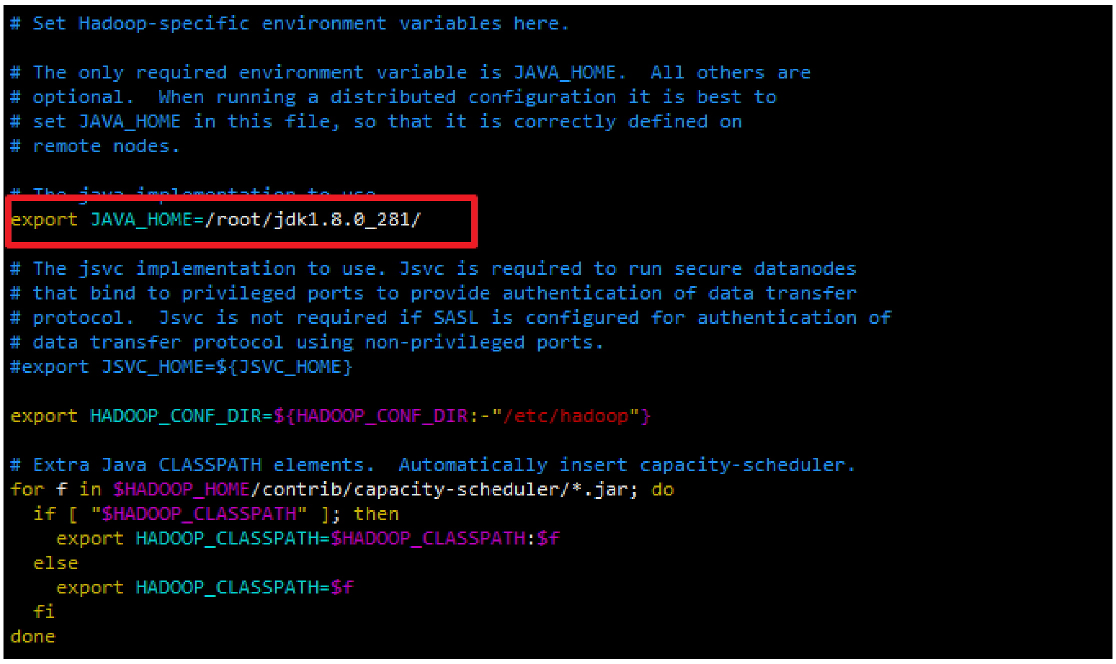
```
[root@master ~]# pwd
/root
```



```
[root@master ~]# tar -xzvf hadoop-2.7.7.tar.gz -C /root/
```

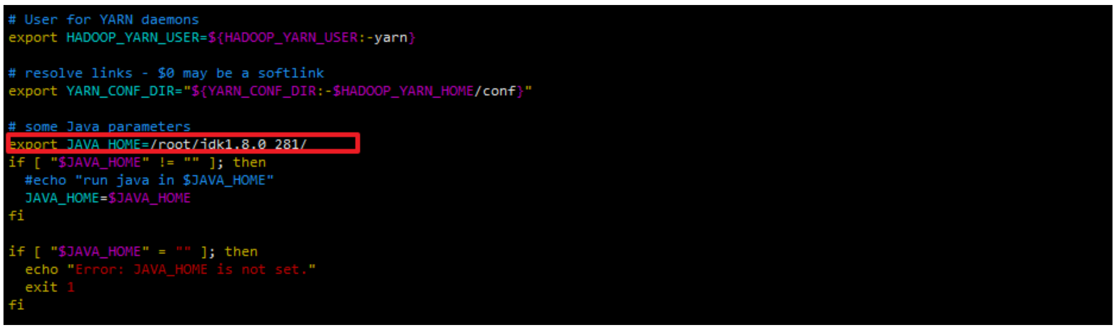
3.1.5.2 配置以下 7 个文件:

- hadoop-env.sh 文件



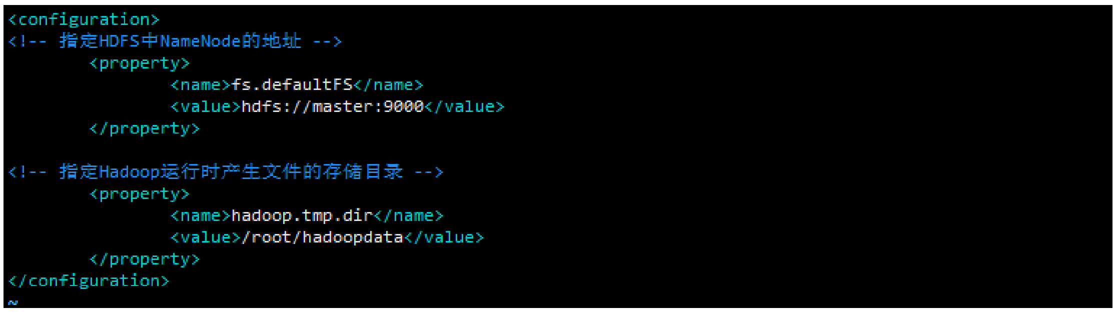
```
# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
export JAVA_HOME=/root/jdk1.8.0_281/

# The jsvc implementation to use. Jsvc is required to run secure datanodes
# 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
```

- yarn-env.sh 文件



```
# User for YARN daemons
export HADOOP_YARN_USER=${HADOOP_YARN_USER:-yarn}

# resolve links - $0 may be a softlink
export YARN_CONF_DIR="${YARN_CONF_DIR:-$HADOOP_YARN_HOME/conf}"

# some Java parameters
export JAVA_HOME=/root/jdk1.8.0_281/
if [ "$JAVA_HOME" != "" ]; then
  #echo "run java in $JAVA_HOME"
  JAVA_HOME=$JAVA_HOME
fi

if [ "$JAVA_HOME" = "" ]; then
  echo "Error: JAVA_HOME is not set."
  exit 1
fi
```

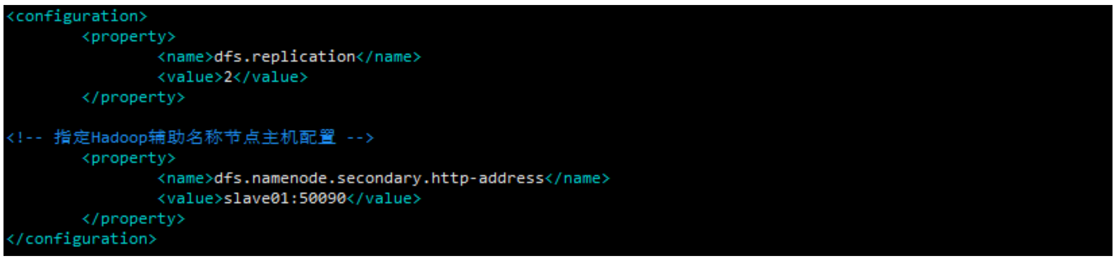
- core-site.xml 文件



```
<configuration>
<!-- 指定HDFS中NameNode的地址 -->
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://master:9000</value>
        </property>

<!-- 指定Hadoop运行时产生文件的存储目录 -->
        <property>
                <name>hadoop.tmp.dir</name>
                <value>/root/hadoopdata</value>
        </property>
</configuration>
~
```

```
    <configuration>
    <!-- 指定 HDFS 中 NameNode 的地址 -->
        <property>
            <name>fs.defaultFS</name>
                <value>hdfs://master:9000</value>
        </property>

    <!-- 指定 Hadoop 运行时产生文件的存储目录 -->
        <property>
            <name>hadoop.tmp.dir</name>
            <value>/root/hadoopdata</value>
        </property>
</configuration>
```

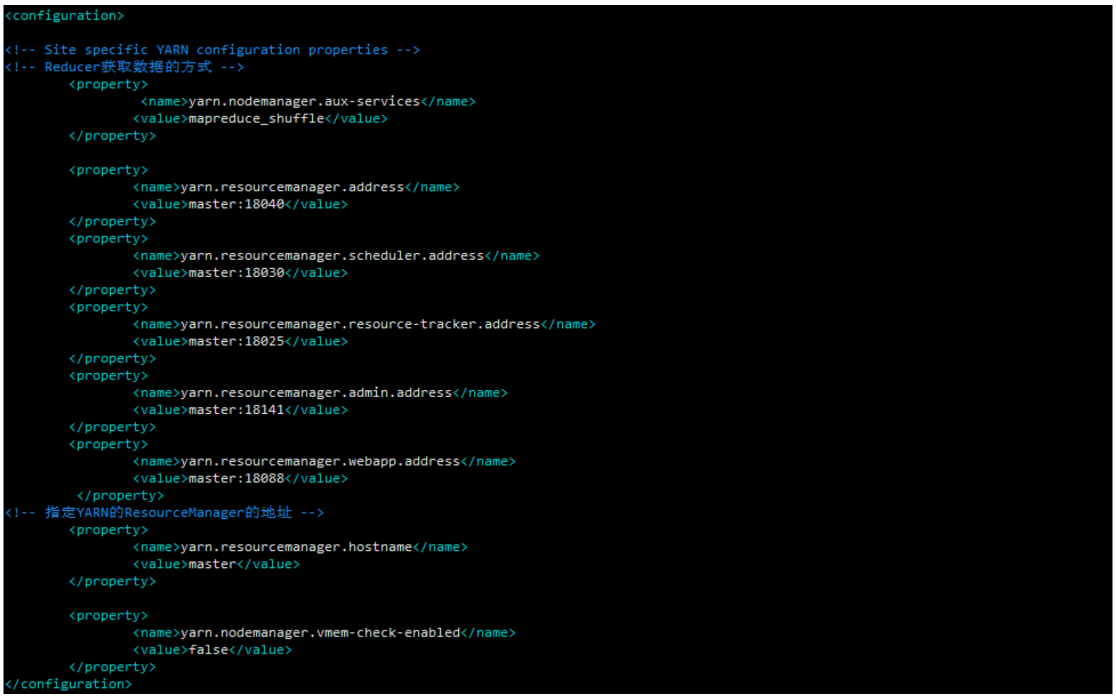
- hdfs-site.xml 文件



```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>2</value>
        </property>

<!-- 指定Hadoop辅助名称节点主机配置 -->
        <property>
                <name>dfs.namenode.secondary.http-address</name>
                <value>slave01:50090</value>
        </property>
</configuration>
```

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>2</value>
    </property>

<!-- 指定 Hadoop 辅助名称节点主机配置 -->
    <property>
            <name>dfs.namenode.secondary.http-
address</name>
            <value>slave01:50090</value>
    </property>
</configuration>
```

- yarn-site.xml 文件

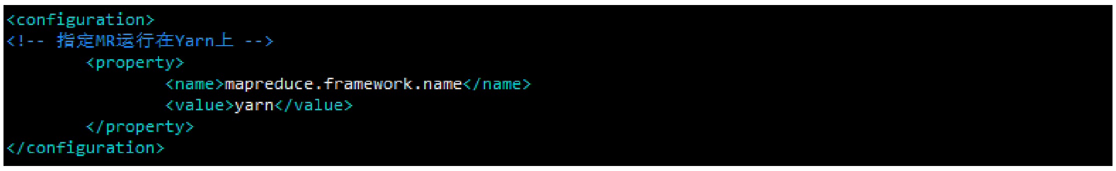


```
<configuration>

<!-- Site specific YARN configuration properties -->
<!-- Reducer获取数据的方式 -->
        <property>
                 <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>

        <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.resource-tracker.address</name>
                <value>master:18025</value>
        </property>
        <property>
                <name>yarn.resourcemanager.admin.address</name>
                <value>master:18141</value>
        </property>
        <property>
                <name>yarn.resourcemanager.webapp.address</name>
                <value>master:18088</value>
         </property>
<!-- 指定YARN的ResourceManager的地址 -->
        <property>
                <name>yarn.resourcemanager.hostname</name>
                <value>master</value>
        </property>

        <property>
                <name>yarn.nodemanager.vmem-check-enabled</name>
                <value>false</value>
        </property>
</configuration>
```

```
    <configuration>

    <!-- Site specific YARN configuration properties -->
    <!-- Reducer 获取数据的方式 -->
       <property>
            <name>yarn.nodemanager.aux-services</name>
              <value>mapreduce_shuffle</value>
       </property>

           <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.resource-
tracker.address</name>
              <value>master:18025</value>
           </property>
           <property>

    <name>yarn.resourcemanager.admin.address</name>
              <value>master:18141</value>
           </property>
           <property>

    <name>yarn.resourcemanager.webapp.address</name>
```

```
                <value>master:18088</value>
            </property>
    <!-- 指定 YARN 的 ResourceManager 的地址 -->
        <property>
            <name>yarn.resourcemanager.hostname</name>
            <value>master</value>
        </property>

        <property>
                <name>yarn.nodemanager.vmem-check-
enabled</name>
                <value>false</value>
        </property>
</configuration>
```

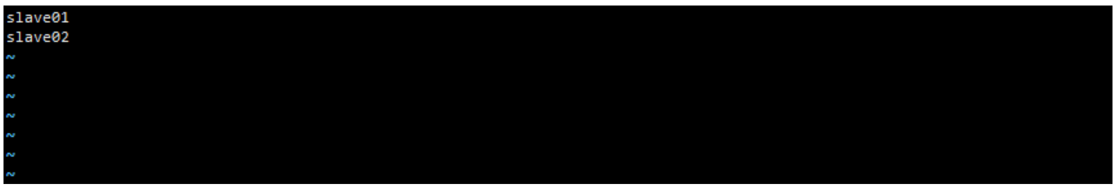
- mapred-site.xml 文件



```
<configuration>
<!-- 指定MR运行在Yarn上 -->
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
</configuration>
```

```
    <configuration>
    <!-- 指定 MR 运行在 Yarn 上 -->
        <property>
            <name>mapreduce.framework.name</name>
            <value>yarn</value>
        </property>
</configuration>
```

- slaves 文件

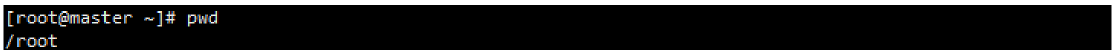


```
slave01
slave02
~
~
~
~
~
~
~
```

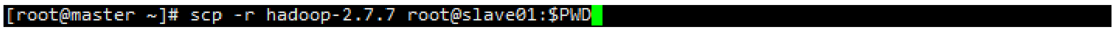
3.1.5.3 在 core-site.xml 指定了 Hadoop 运行时产生文件的存储目录

```
mkdir -p /root/hadoopdata
```

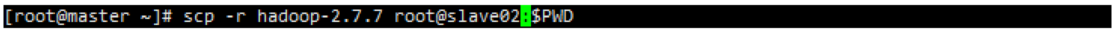
3.1.5.4 使用 scp 命令将 Hadoopslave01、slave02 上



```
[root@master ~]# pwd
/root
```

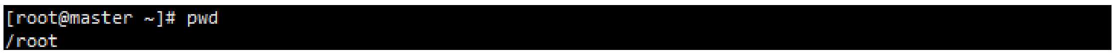


```
[root@master ~]# scp -r hadoop-2.7.7 root@slave01:$PWD
```

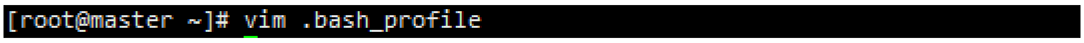


```
[root@master ~]# scp -r hadoop-2.7.7 root@slave02:$PWD
```

配置 Hadoop 环境变量（在三台节点分别操作此步骤）



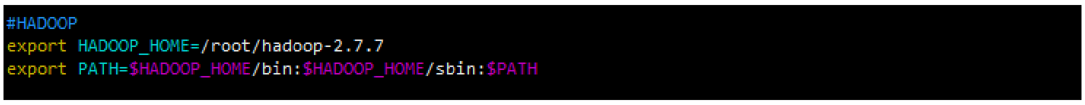
```
[root@master ~]# pwd
/root
```



```
[root@master ~]# vim .bash_profile
```

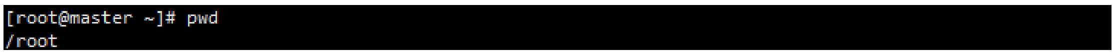
在.bash_profile 末尾添加如下内容

```
#HADOOP
export HADOOP_HOME=/root/hadoop-2.7.7
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
```

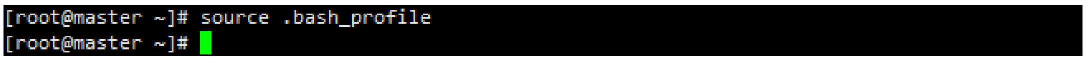


```
#HADOOP
export HADOOP_HOME=/root/hadoop-2.7.7
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
```

使环境变量生效



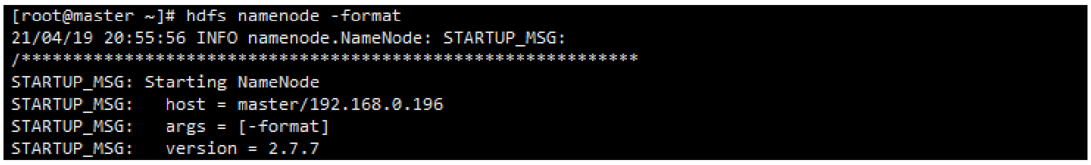
```
[root@master ~]# pwd
/root
```



```
[root@master ~]# source .bash_profile
[root@master ~]#
```

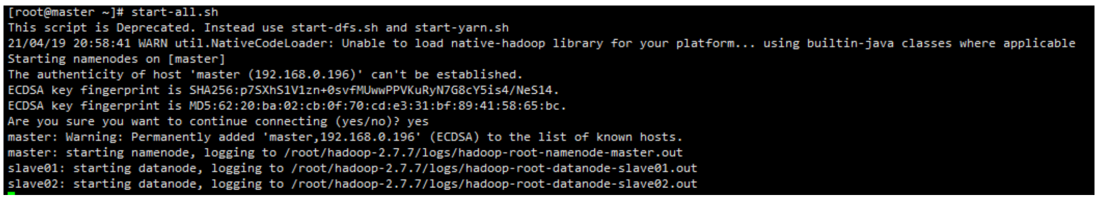
3.1.5.5 格式化 Hadoop 文件目录

```
hdfs namenode -format
```



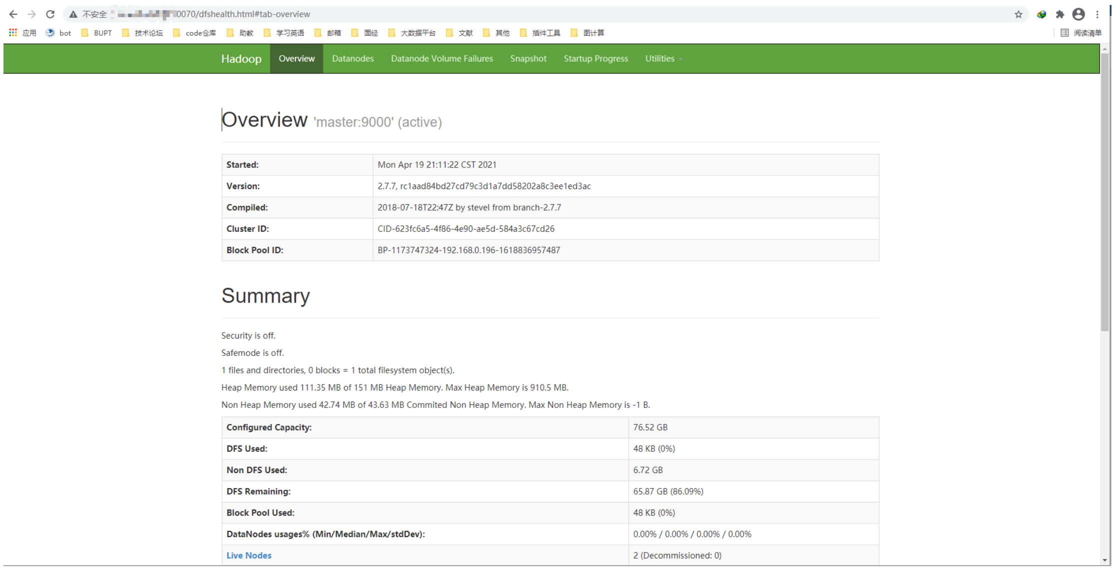
```
[root@master ~]# hdfs namenode -format
21/04/19 20:55:56 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = master/192.168.0.196
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.7.7
```

3.1.5.6 验证 Hadoop 是否安装成功



```
[root@master ~]# start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
21/04/19 20:58:41 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting namenodes on [master]
The authenticity of host 'master (192.168.0.196)' can't be established.
ECDSA key fingerprint is SHA256:p7SXhS1V1zn+0svfMUwwPPVKuRyN7G8cY5is4/NeS14.
ECDSA key fingerprint is MD5:62:20:ba:02:cb:0f:70:cd:e3:31:bf:89:41:58:65:bc.
Are you sure you want to continue connecting (yes/no)? yes
master: Warning: Permanently added 'master,192.168.0.196' (ECDSA) to the list of known hosts.
master: starting namenode, logging to /root/hadoop-2.7.7/logs/hadoop-root-namenode-master.out
slave01: starting datanode, logging to /root/hadoop-2.7.7/logs/hadoop-root-datanode-slave01.out
slave02: starting datanode, logging to /root/hadoop-2.7.7/logs/hadoop-root-datanode-slave02.out
```

http://masterIP 地址:50070/


不安全 [redacted]0070/dfshealth.html#tab-overview

应用 bot BUPT 技术论坛 code仓库 助教 学习英语 邮箱 面经 大数据平台 文献 其他 插件工具 图计算 阅读清单

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Overview 'master:9000' (active)

Started:	Mon Apr 19 21:11:22 CST 2021
Version:	2.7.7, rc1aad84bd27cd79c3d1a7dd58202a8c3ee1ed3ac
Compiled:	2018-07-18T22:47Z by stevel from branch-2.7.7
Cluster ID:	CID-623fc6a5-4f86-4e90-ae5d-584a3c67cd26
Block Pool ID:	BP-1173747324-192.168.0.196-1618836957487

Summary

Security is off.

Safemode is off.

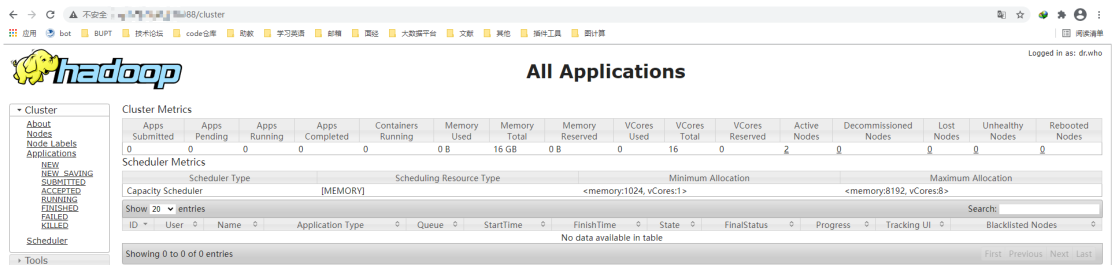
1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 111.35 MB of 151 MB Heap Memory. Max Heap Memory is 910.5 MB.

Non Heap Memory used 42.74 MB of 43.63 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	76.52 GB
DFS Used:	48 KB (0%)
Non DFS Used:	6.72 GB
DFS Remaining:	65.87 GB (86.09%)
Block Pool Used:	48 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	2 (Decommissioned: 0)

http://masterIP 地址:18088/


不安全 [redacted]88/cluster

应用 bot BUPT 技术论坛 code仓库 助教 学习英语 邮箱 面经 大数据平台 文献 其他 插件工具 图计算 阅读清单

Logged in as: dr.who

All Applications

Cluster
About
Nodes
Node Labels
Applications
NEW
NEW_SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
0	0	0	0	0	0 B	16 GB	0 B	0	16	0	2	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:8>

Show 20 entries　　Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blacklisted Nodes
No data available in table											

Showing 0 to 0 of 0 entries　　First Previous Next Last

3.2 Zookeeper 安装（参考普开 Zookeeper 安装部署）

3.2.1 解压 Zookeeper 压缩包

```
[root@master ~]# pwd
/root
```

```
[root@master ~]# tar -xzvf zookeeper-3.4.12.tar.gz -C /root/
```

3.2.2 使用复制命令生成配置文件

```
[root@master conf]# pwd
/root/zookeeper-3.4.12/conf
[root@master conf]# cp zoo_sample.cfg zoo.cfg
```

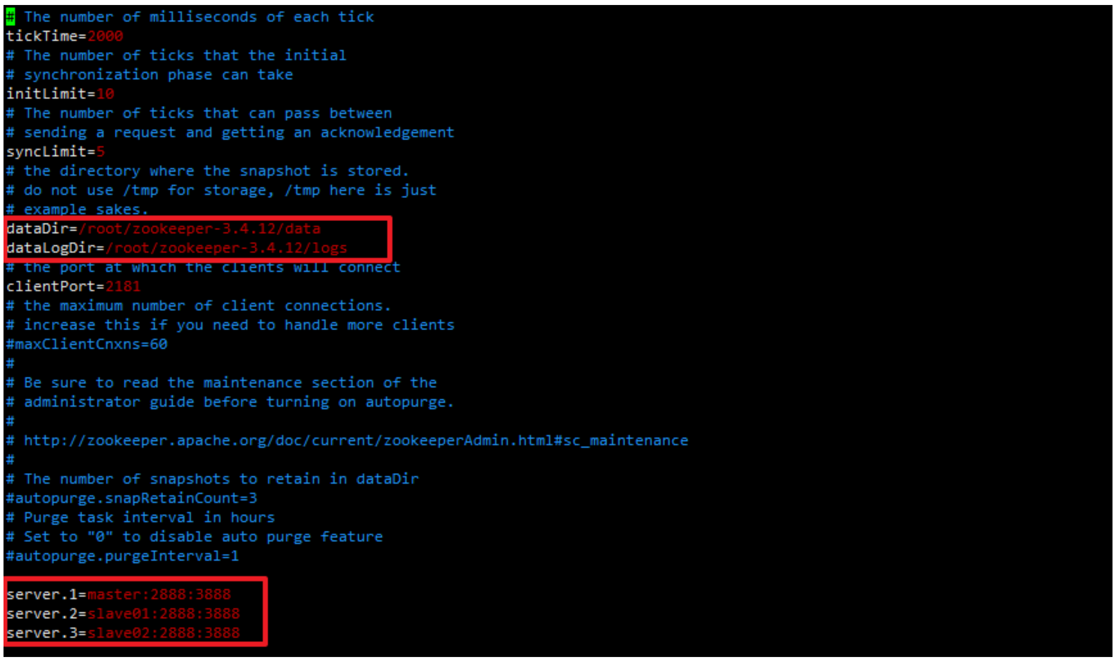
3.2.3 用 vim 编译器修改 zoo.cfg 配置文件

```
dataDir=/root/zookeeper-3.4.12/data
dataLogDir=/root/zookeeper-3.4.12/logs
```

server.1=master:2888:3888

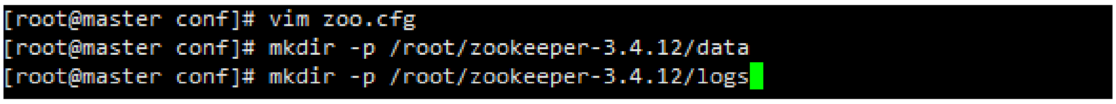
server.2=slave01:2888:3888

server.3=slave02:2888:3888


```
# 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=/root/zookeeper-3.4.12/data
dataLogDir=/root/zookeeper-3.4.12/logs
# 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

server.1=master:2888:3888
server.2=slave01:2888:3888
server.3=slave02:2888:3888
```

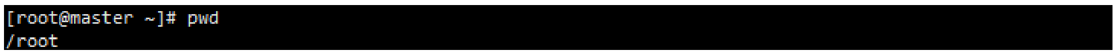
3.2.4 新建在 zoo.cfg 配置文件中 dataDir 和 dataLogDir 所指定的文件夹

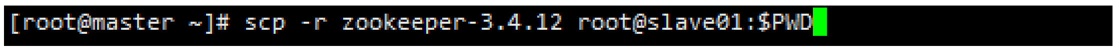
```
mkdir -p /root/zookeeper-3.4.12/data
mkdir -p /root/zookeeper-3.4.12/logs
```

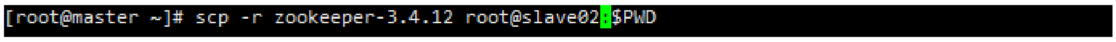

```
[root@master conf]# vim zoo.cfg
[root@master conf]# mkdir -p /root/zookeeper-3.4.12/data
[root@master conf]# mkdir -p /root/zookeeper-3.4.12/logs
```

3.2.5 将 zookeeper 文件夹远程复制到 slave01 和 slave02 节点上

```
scp -r zookeeper-3.4.12 root@slave01:$PWD
scp -r zookeeper-3.4.12 root@slave02:$PWD
```


```
[root@master ~]# pwd
/root
```


```
[root@master ~]# scp -r zookeeper-3.4.12 root@slave01:$PWD
```

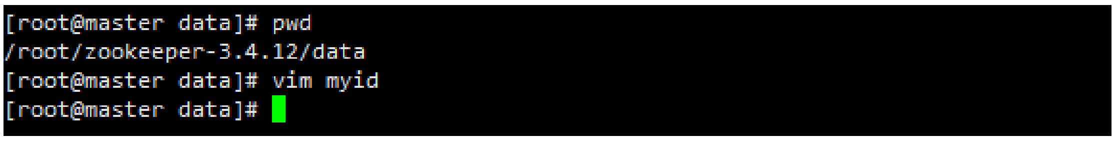

```
[root@master ~]# scp -r zookeeper-3.4.12 root@slave02:$PWD
```

3.2.6 zoo.cfg 的配置属性 dataDir=/root/zookeeper-3.4.12/data/，在此配置

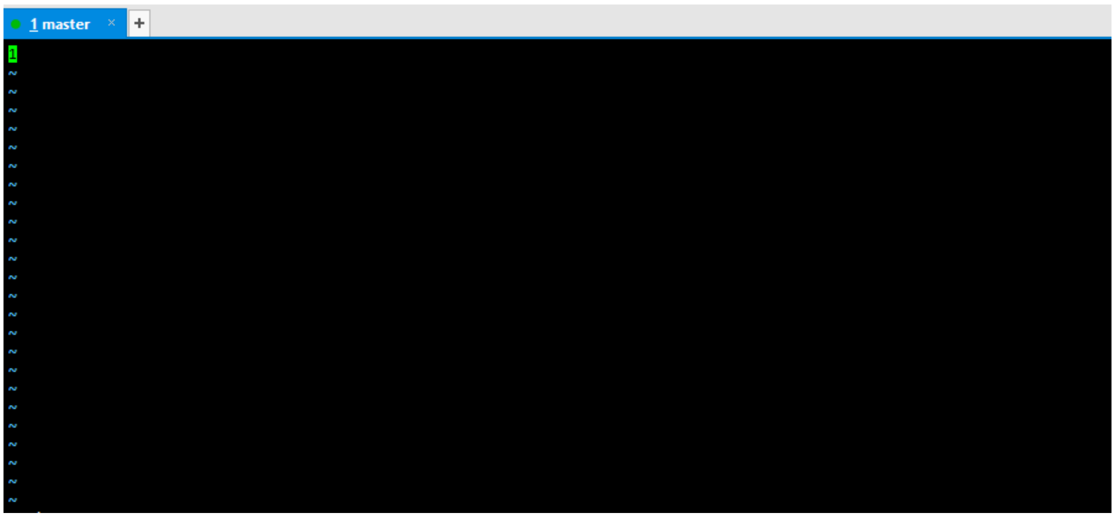
属性所指定文件夹下新建 myid 文件（三个节点都需要如下操作）

注意:在/root/zookeeper-3.4.12/data/目录下创建 myid 文件，文件中只包含一行，且内容为该节点对应上图中标识 1 的 server.id 中的 id 编号。

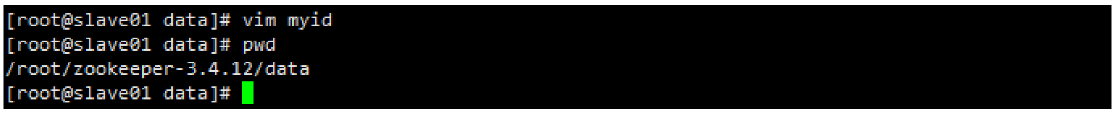
例如，master 和 slave01、slave02 分别对应的 myid 文件中的值是 1 和 2、3。



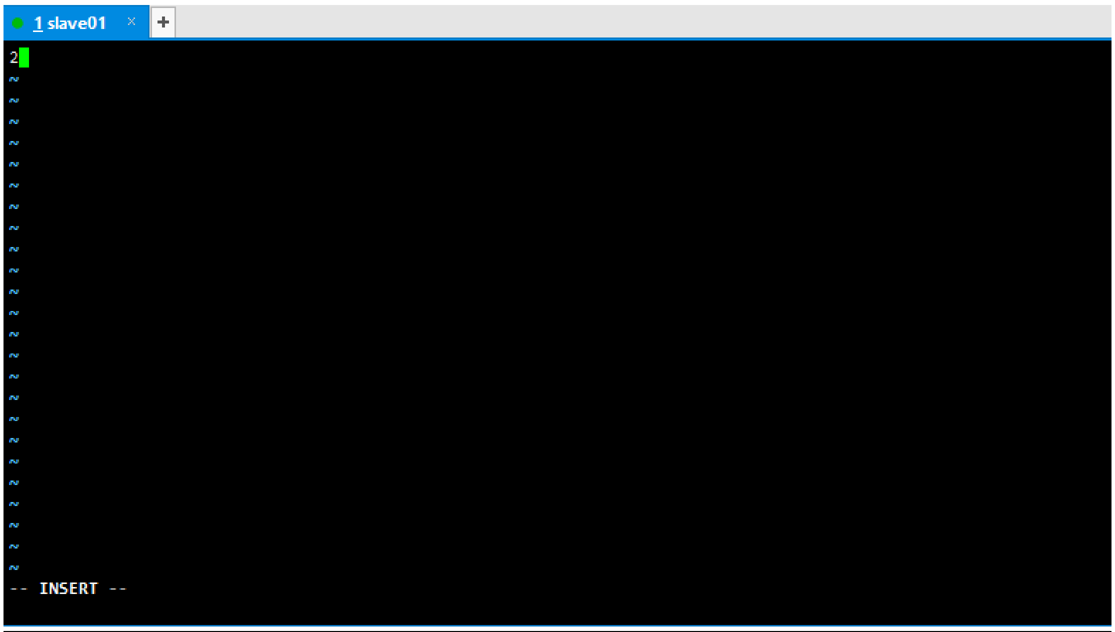
```
[root@master data]# pwd
/root/zookeeper-3.4.12/data
[root@master data]# vim myid
[root@master data]#
```



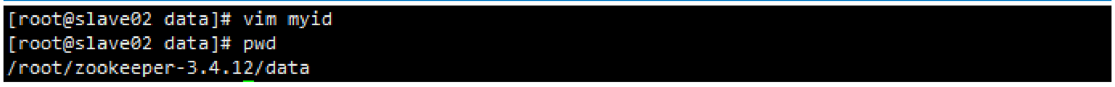
```
1
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```



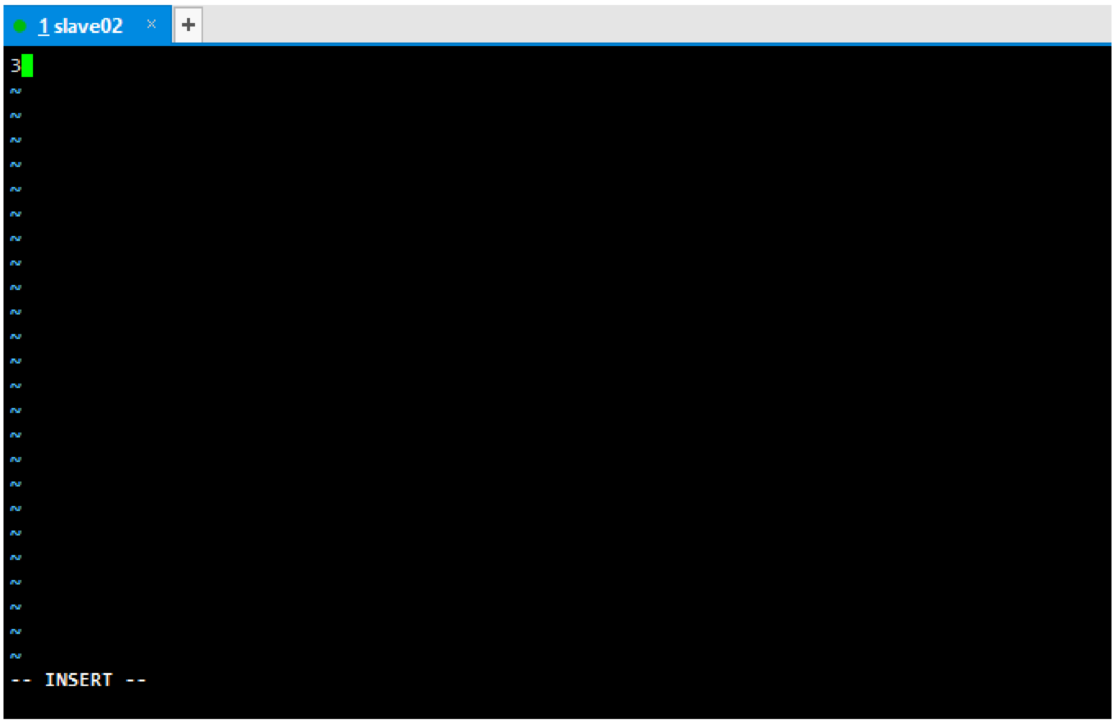
```
[root@slave01 data]# vim myid
[root@slave01 data]# pwd
/root/zookeeper-3.4.12/data
[root@slave01 data]#
```



```
2
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

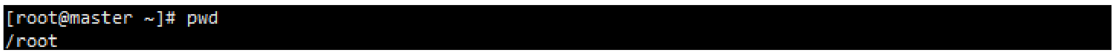


```
[root@slave02 data]# vim myid
[root@slave02 data]# pwd
/root/zookeeper-3.4.12/data
```

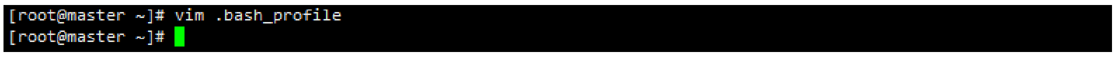


```
3
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

3.2.7 配置环境变量（在三台节点分别操作此步骤）

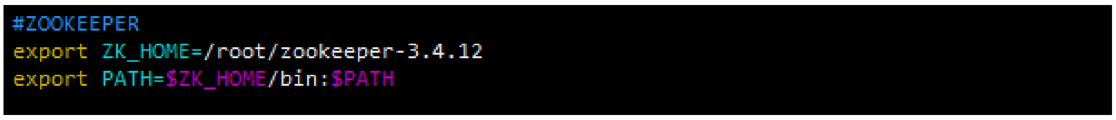


```
[root@master ~]# pwd
/root
```



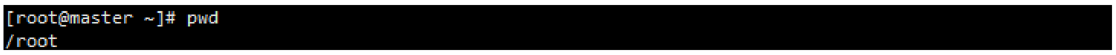
```
[root@master ~]# vim .bash_profile
[root@master ~]#
```

将下图内容添加到“.bash_profile”文件中

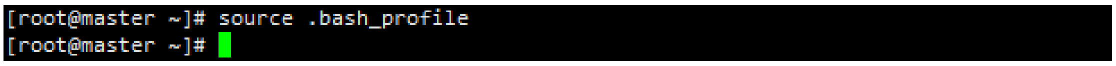


```
#ZOOKEEPER
export ZK_HOME=/root/zookeeper-3.4.12
export PATH=$ZK_HOME/bin:$PATH
```

让新添加的环境变量生效



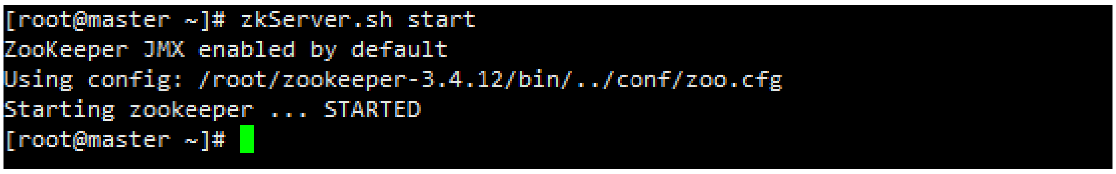
```
[root@master ~]# pwd
/root
```



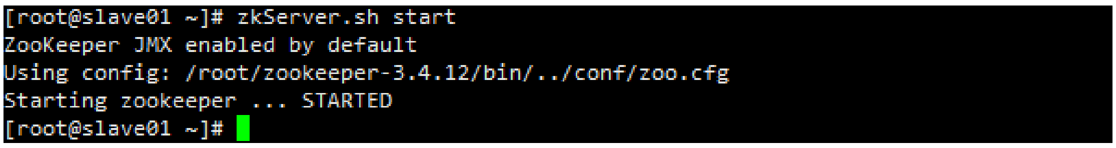
```
[root@master ~]# source .bash_profile
[root@master ~]#
```

3.2.8 验证 Zookeeper 集群是否安装成功

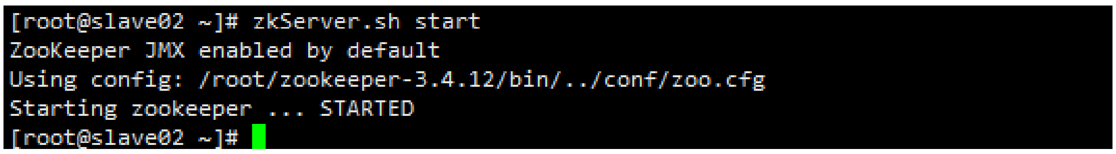
在三个节点分别运行如下命令



```
[root@master ~]# zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
[root@master ~]#
```

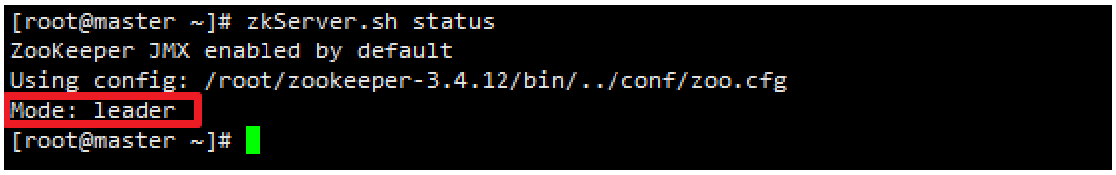


```
[root@slave01 ~]# zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
[root@slave01 ~]#
```

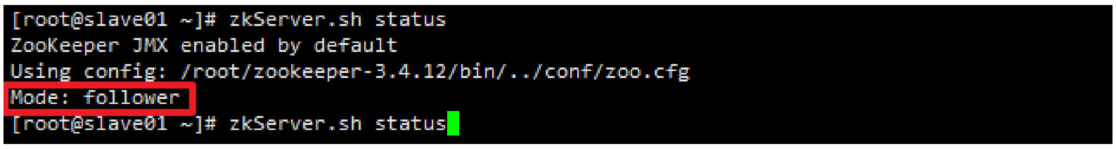


```
[root@slave02 ~]# zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
[root@slave02 ~]#
```

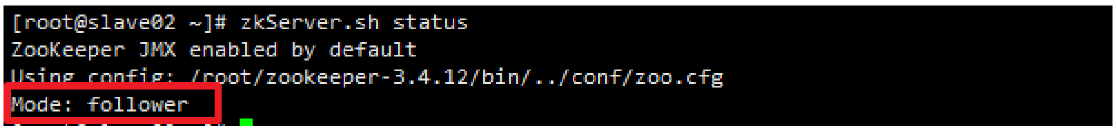
安装成功



```
[root@master ~]# zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Mode: leader
[root@master ~]#
```



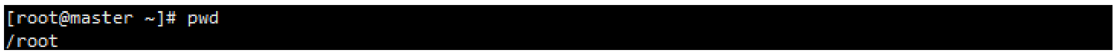
```
[root@slave01 ~]# zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Mode: follower
[root@slave01 ~]# zkServer.sh status
```



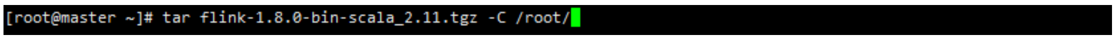
```
[root@slave02 ~]# zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Mode: follower
```

3.3 Flink 安装

3.3.1 解压 Flink 压缩包



```
[root@master ~]# pwd
/root
```



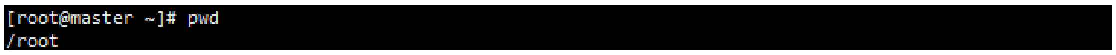
```
[root@master ~]# tar flink-1.8.0-bin-scala_2.11.tgz -C /root/
```

3.3.2 修改 flink-conf.yaml 文件

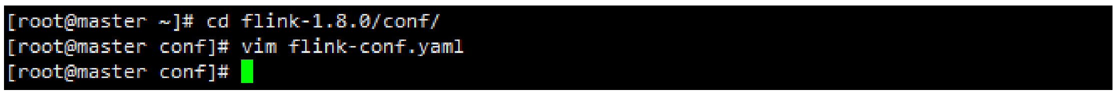
Jobmanager.rpc.address: JobManager 的 IP 地址

taskmanager.numberOfTaskSlots: 每个 TaskManager 提供的任务 Slots 数量大小，一般根据每台机器的可用 CPU 进行配置

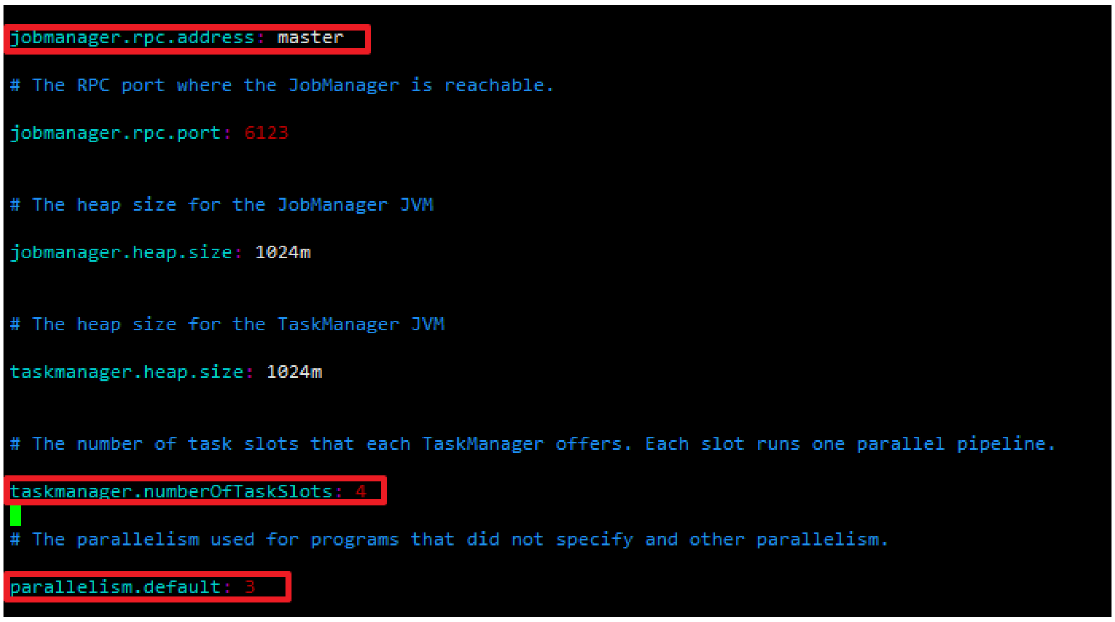
parallelism.default:程序默认并行计算的个数



```
[root@master ~]# pwd
/root
```

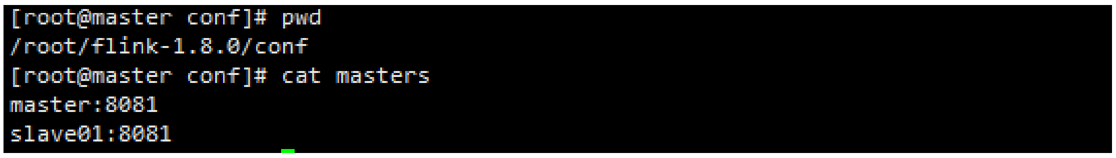


```
[root@master ~]# cd flink-1.8.0/conf/
[root@master conf]# vim flink-conf.yaml
[root@master conf]#
```



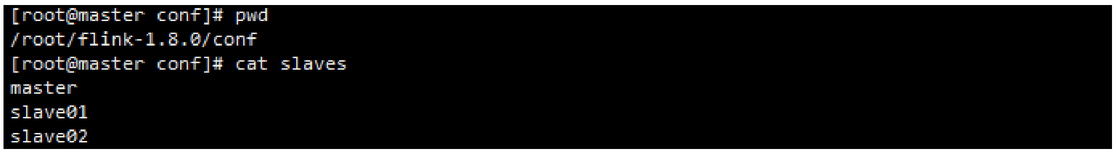
```
jobmanager.rpc.address: master

# The RPC port where the JobManager is reachable.

jobmanager.rpc.port: 6123


# The heap size for the JobManager JVM

jobmanager.heap.size: 1024m


# The heap size for the TaskManager JVM

taskmanager.heap.size: 1024m


# The number of task slots that each TaskManager offers. Each slot runs one parallel pipeline.

taskmanager.numberOfTaskSlots: 4

# The parallelism used for programs that did not specify and other parallelism.

parallelism.default: 3
```

3.3.3 修改 master 文件



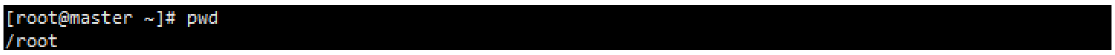
```
[root@master conf]# pwd
/root/flink-1.8.0/conf
[root@master conf]# cat masters
master:8081
slave01:8081
```

3.3.4 修改 slaves 文件

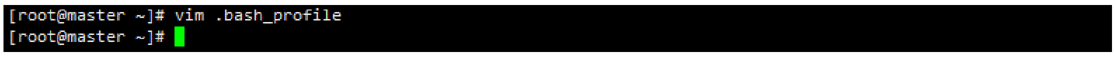


```
[root@master conf]# pwd
/root/flink-1.8.0/conf
[root@master conf]# cat slaves
master
slave01
slave02
```

3.3.5 配置环境变量（在三台节点分别操作此步骤）

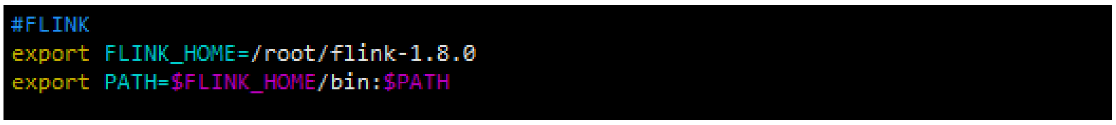


```
[root@master ~]# pwd
/root
```



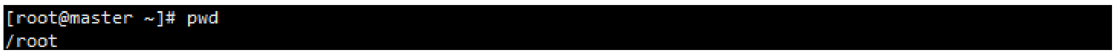
```
[root@master ~]# vim .bash_profile
[root@master ~]#
```

将下图内容添加到“.bash_profile”文件中

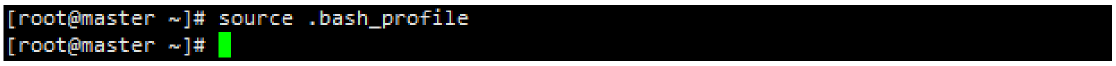


```
#FLINK
export FLINK_HOME=/root/flink-1.8.0
export PATH=$FLINK_HOME/bin:$PATH
```

让新添加的环境变量生效

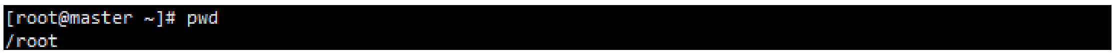


```
[root@master ~]# pwd
/root
```

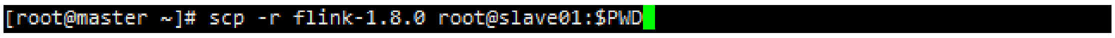


```
[root@master ~]# source .bash_profile
[root@master ~]#
```

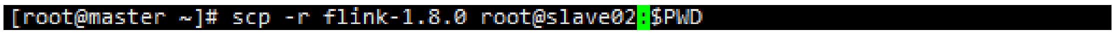
3.3.6 将 master 的 flink 安装包分发到其他节点



```
[root@master ~]# pwd
/root
```



```
[root@master ~]# scp -r flink-1.8.0 root@slave01:$PWD
```

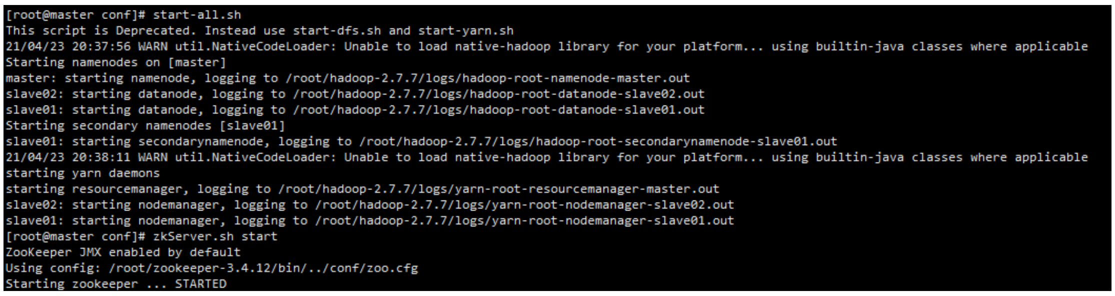


```
[root@master ~]# scp -r flink-1.8.0 root@slave02:$PWD
```

3.3.7 安装验证

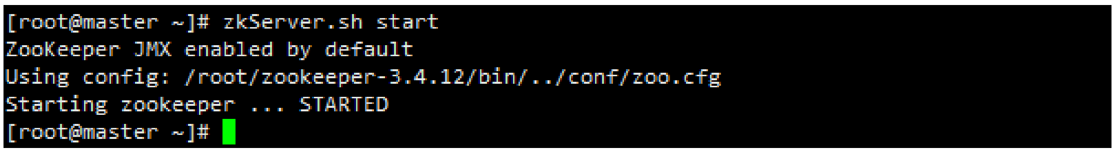
启动 flink 之前需要依次启动 hadoop、zookeeper

在 master 节点启动 hadoop

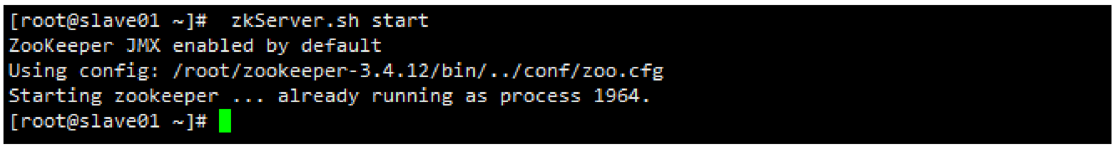


```
[root@master conf]# start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
21/04/23 20:37:56 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting namenodes on [master]
master: starting namenode, logging to /root/hadoop-2.7.7/logs/hadoop-root-namenode-master.out
slave02: starting datanode, logging to /root/hadoop-2.7.7/logs/hadoop-root-datanode-slave02.out
slave01: starting datanode, logging to /root/hadoop-2.7.7/logs/hadoop-root-datanode-slave01.out
Starting secondary namenodes [slave01]
slave01: starting secondarynamenode, logging to /root/hadoop-2.7.7/logs/hadoop-root-secondarynamenode-slave01.out
21/04/23 20:38:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
starting yarn daemons
starting resourcemanager, logging to /root/hadoop-2.7.7/logs/yarn-root-resourcemanager-master.out
slave02: starting nodemanager, logging to /root/hadoop-2.7.7/logs/yarn-root-nodemanager-slave02.out
slave01: starting nodemanager, logging to /root/hadoop-2.7.7/logs/yarn-root-nodemanager-slave01.out
[root@master conf]# zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
```

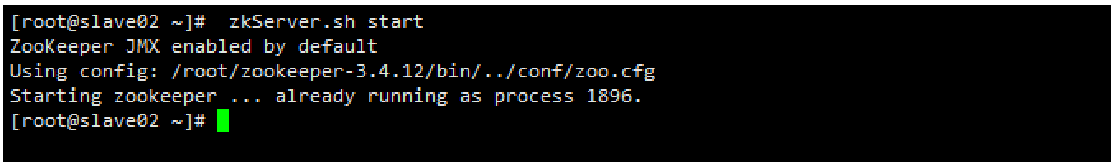
在三个节点分别输入以下命令启动 zookeeper



```
[root@master ~]# zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
[root@master ~]#
```



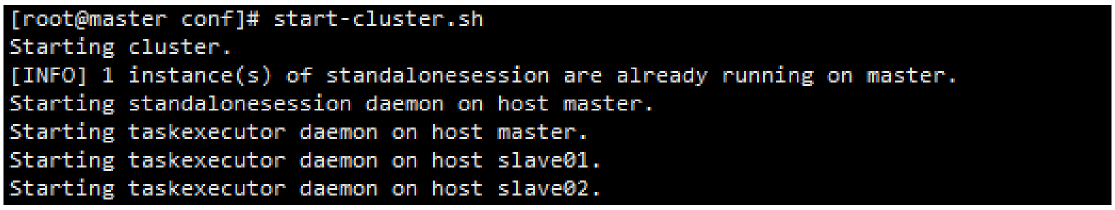
```
[root@slave01 ~]#  zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Starting zookeeper ... already running as process 1964.
[root@slave01 ~]#
```



```
[root@slave02 ~]#  zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Starting zookeeper ... already running as process 1896.
[root@slave02 ~]#
```

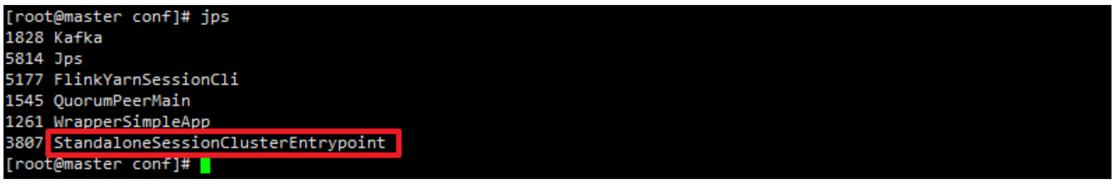
首次启动时需要在三个节点分别输入：start-cluster.sh

第二次启动只需要在 master 输入：start-cluster.sh

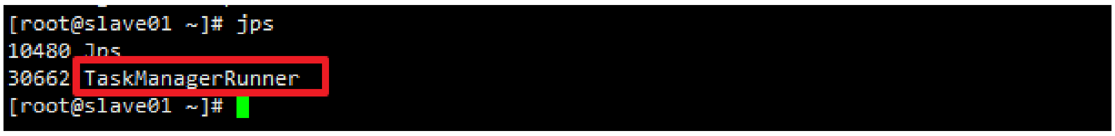


```
[root@master conf]# start-cluster.sh
Starting cluster.
[INFO] 1 instance(s) of standalonesession are already running on master.
Starting standalonesession daemon on host master.
Starting taskexecutor daemon on host master.
Starting taskexecutor daemon on host slave01.
Starting taskexecutor daemon on host slave02.
```

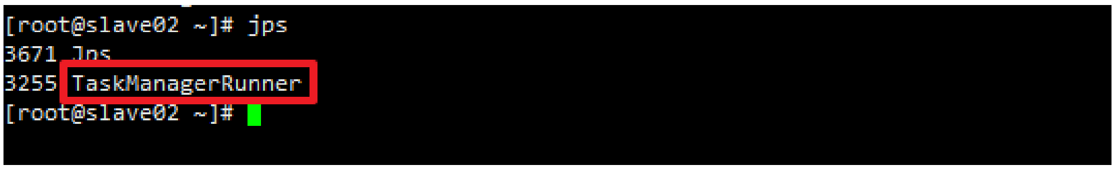
安装成功



```
[root@master conf]# jps
1828 Kafka
5814 Jps
5177 FlinkYarnSessionCli
1545 QuorumPeerMain
1261 WrapperSimpleApp
3807 StandaloneSessionClusterEntrypoint
[root@master conf]#
```



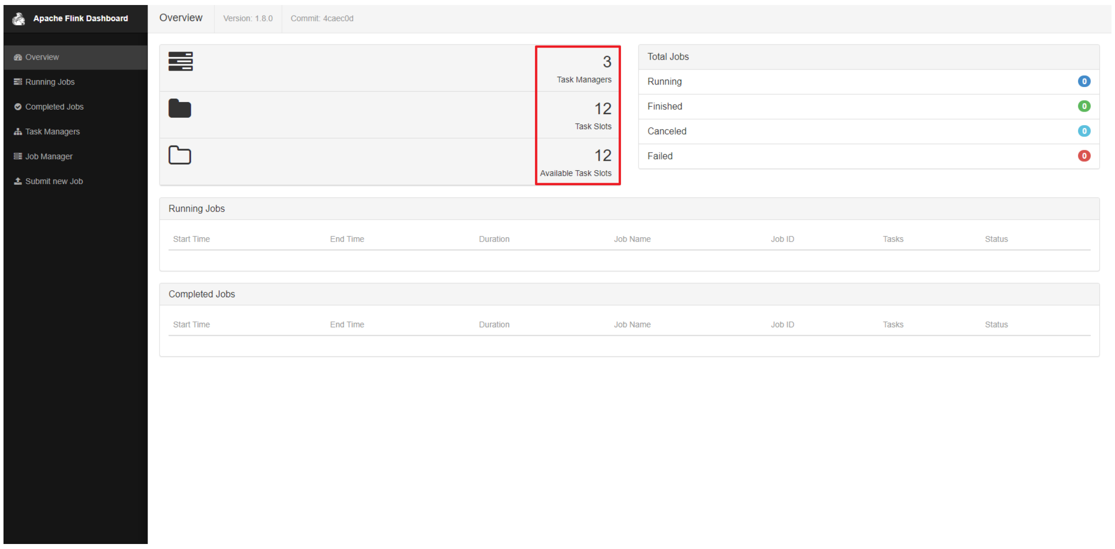
```
[root@slave01 ~]# jps
10480 Jps
30662 TaskManagerRunner
[root@slave01 ~]#
```



```
[root@slave02 ~]# jps
3671 Jps
3255 TaskManagerRunner
[root@slave02 ~]#
```

访问 http://masterIP:8081/

圈中地方为 0 则为安装失败



Apache Flink Dashboard

Overview   Version: 1.8.0   Commit: 4caec0d

- Overview
- Running Jobs
- Completed Jobs
- Task Managers
- Job Manager
- Submit new Job

3	Task Managers
12	Task Slots
12	Available Task Slots

Total Jobs	
Running	0
Finished	0
Canceled	0
Failed	0

Running Jobs

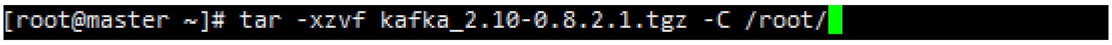
Start Time	End Time	Duration	Job Name	Job ID	Tasks	Status

Completed Jobs

Start Time	End Time	Duration	Job Name	Job ID	Tasks	Status

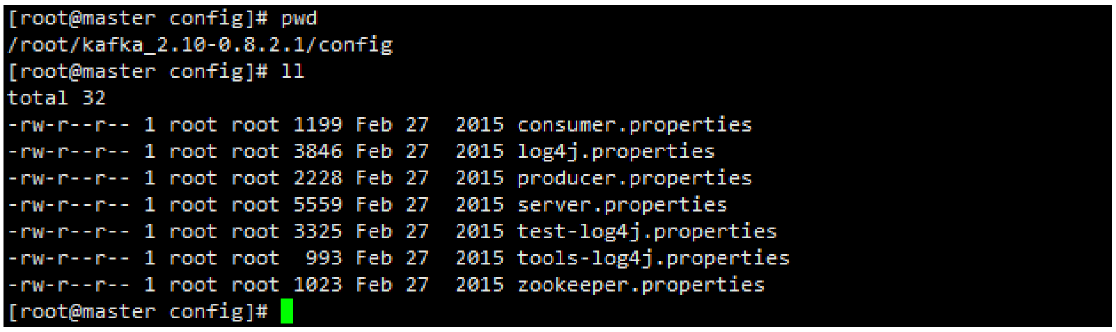
3.4 Kafka 安装（参考普开 Kafka 安装）

3.4.1 解压 Kafka 压缩包



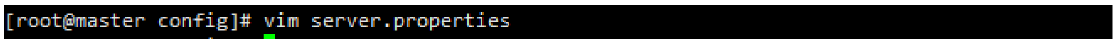
```
[root@master ~]# tar -xzvf kafka_2.10-0.8.2.1.tgz -C /root/
```

3.4.2 进入 config 文件夹



```
[root@master config]# pwd
/root/kafka_2.10-0.8.2.1/config
[root@master config]# ll
total 32
-rw-r--r-- 1 root root 1199 Feb 27  2015 consumer.properties
-rw-r--r-- 1 root root 3846 Feb 27  2015 log4j.properties
-rw-r--r-- 1 root root 2228 Feb 27  2015 producer.properties
-rw-r--r-- 1 root root 5559 Feb 27  2015 server.properties
-rw-r--r-- 1 root root 3325 Feb 27  2015 test-log4j.properties
-rw-r--r-- 1 root root  993 Feb 27  2015 tools-log4j.properties
-rw-r--r-- 1 root root 1023 Feb 27  2015 zookeeper.properties
[root@master config]#
```

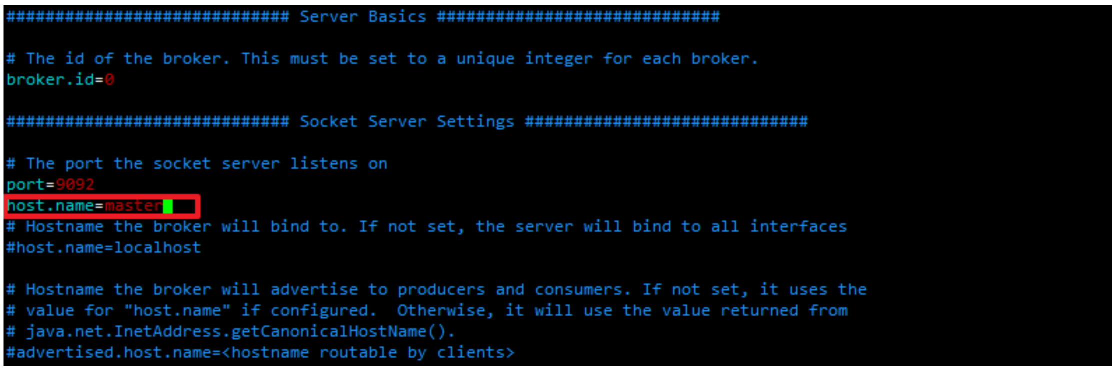
3.4.3 修改 server.properties 配置文件



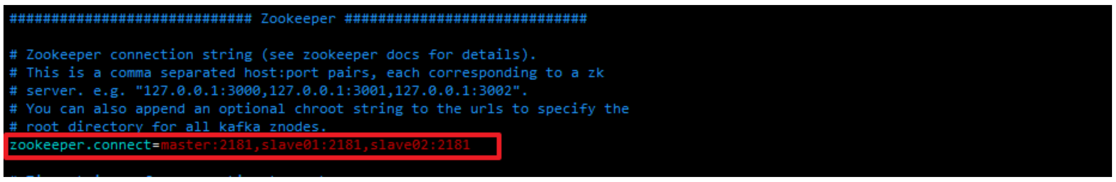
```
[root@master config]# vim server.properties
```

```
broker.id=0
host.name=master
zookeeper.connect=master:2181,slave01:2181,slave02:2181
```

master 做如下修改

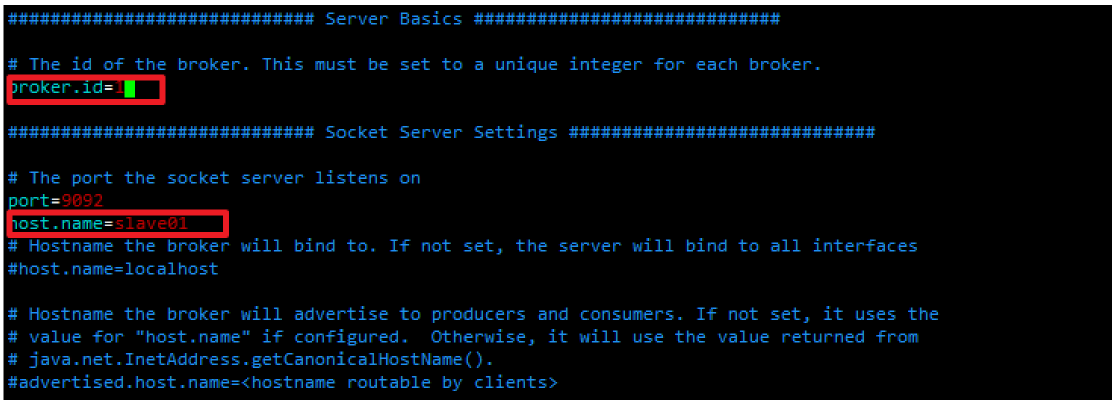


```
############################# Server Basics #############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=0

############################# Socket Server Settings #############################

# The port the socket server listens on
port=9092
host.name=master
# Hostname the broker will bind to. If not set, the server will bind to all interfaces
#host.name=localhost

# Hostname the broker will advertise to producers and consumers. If not set, it uses the
# value for "host.name" if configured.  Otherwise, it will use the value returned from
# java.net.InetAddress.getCanonicalHostName().
#advertised.host.name=<hostname routable by clients>
```

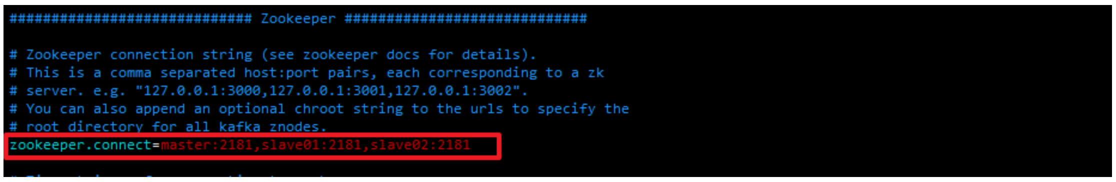


```
############################# Zookeeper #############################

# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
zookeeper.connect=master:2181,slave01:2181,slave02:2181
```

slave01 做如下修改

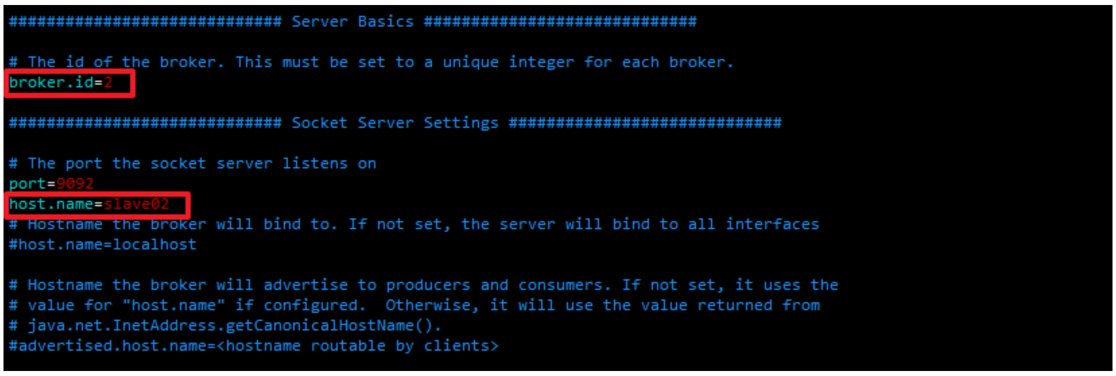


```
############################# Server Basics #############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=1

############################# Socket Server Settings #############################

# The port the socket server listens on
port=9092
host.name=slave01
# Hostname the broker will bind to. If not set, the server will bind to all interfaces
#host.name=localhost

# Hostname the broker will advertise to producers and consumers. If not set, it uses the
# value for "host.name" if configured.  Otherwise, it will use the value returned from
# java.net.InetAddress.getCanonicalHostName().
#advertised.host.name=<hostname routable by clients>
```

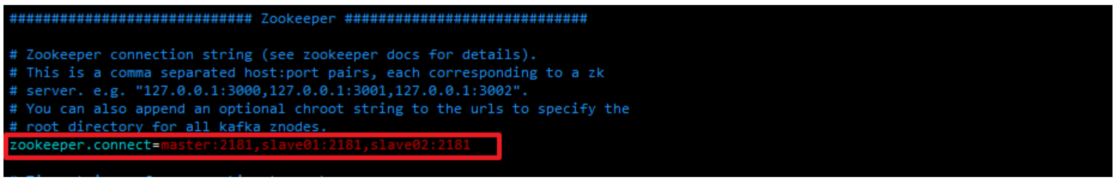


```
############################# Zookeeper #############################

# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
zookeeper.connect=master:2181,slave01:2181,slave02:2181
```

slave02 做如下修改

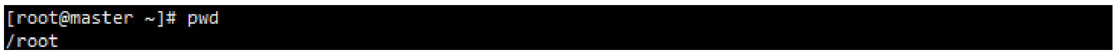


```
############################# Server Basics #############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=2

############################# Socket Server Settings #############################

# The port the socket server listens on
port=9092
host.name=slave02
# Hostname the broker will bind to. If not set, the server will bind to all interfaces
#host.name=localhost

# Hostname the broker will advertise to producers and consumers. If not set, it uses the
# value for "host.name" if configured.  Otherwise, it will use the value returned from
# java.net.InetAddress.getCanonicalHostName().
#advertised.host.name=<hostname routable by clients>
```

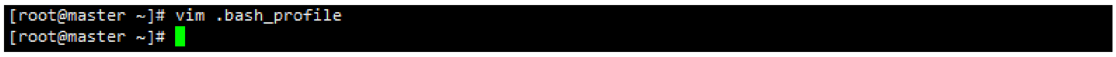


```
############################# Zookeeper #############################

# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
zookeeper.connect=master:2181,slave01:2181,slave02:2181
```

3.4.4 配置环境变量（在三台节点分别操作此步骤）

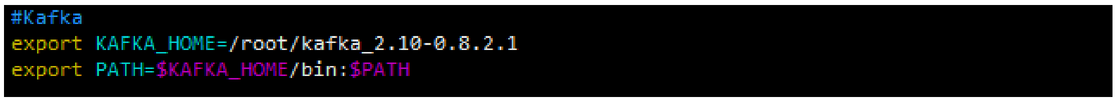


```
[root@master ~]# pwd
/root
```



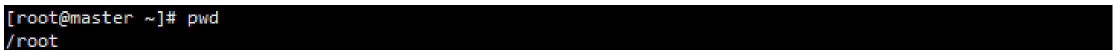
```
[root@master ~]# vim .bash_profile
[root@master ~]#
```

将下图内容添加到“.bash_profile”文件中

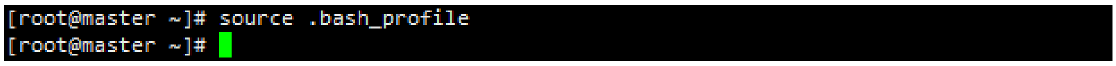


```
#Kafka
export KAFKA_HOME=/root/kafka_2.10-0.8.2.1
export PATH=$KAFKA_HOME/bin:$PATH
```

让新添加的环境变量生效

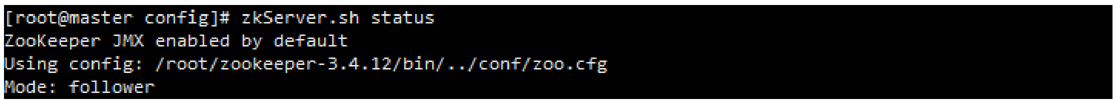


```
[root@master ~]# pwd
/root
```

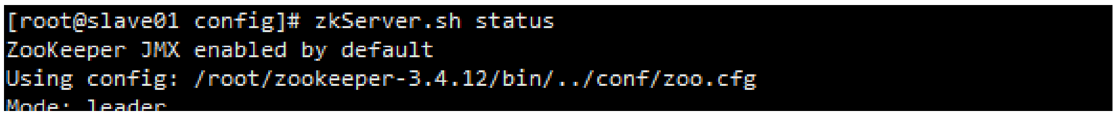


```
[root@master ~]# source .bash_profile
[root@master ~]#
```

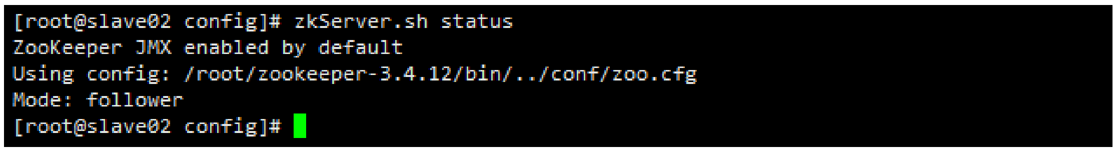
3.4.5 验证 Kafka 是否安装成功（需要提前启动 Zookeeper）



```
[root@master config]# zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Mode: follower
```

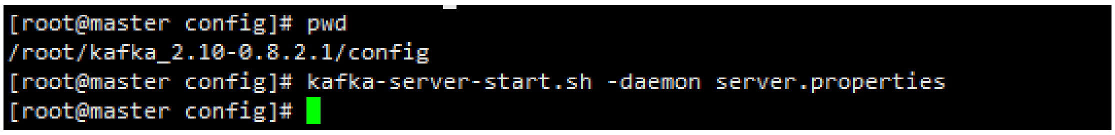


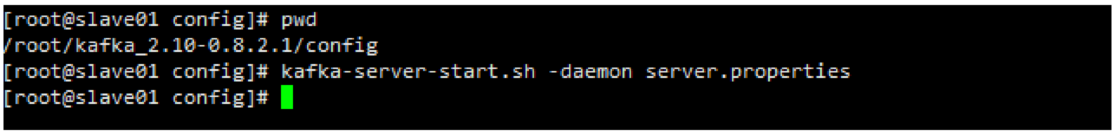
```
[root@slave01 config]# zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Mode: leader
```

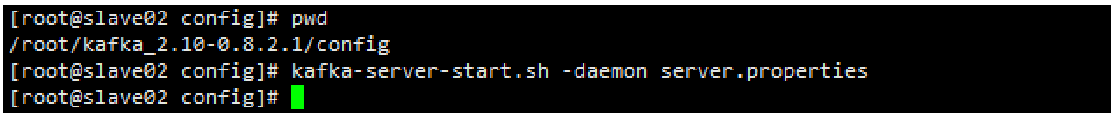


```
[root@slave02 config]# zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Mode: follower
[root@slave02 config]#
```

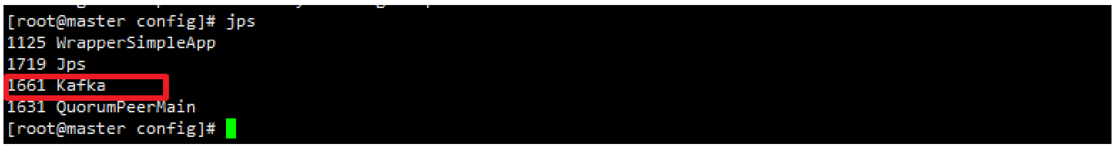
在 master 和 slave01、slave02 节点分别启动 Kafka

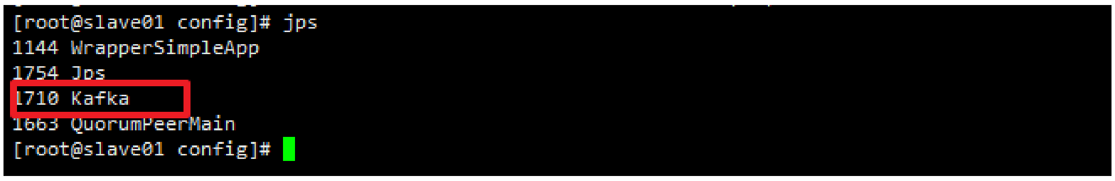

```
[root@master config]# pwd
/root/kafka_2.10-0.8.2.1/config
[root@master config]# kafka-server-start.sh -daemon server.properties
[root@master config]# 
```

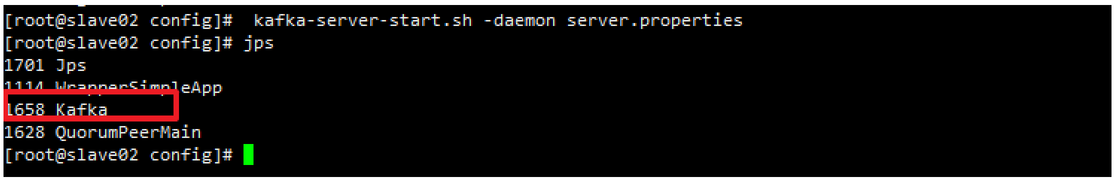

```
[root@slave01 config]# pwd
/root/kafka_2.10-0.8.2.1/config
[root@slave01 config]# kafka-server-start.sh -daemon server.properties
[root@slave01 config]# 
```


```
[root@slave02 config]# pwd
/root/kafka_2.10-0.8.2.1/config
[root@slave02 config]# kafka-server-start.sh -daemon server.properties
[root@slave02 config]# 
```

安装成功


```
[root@master config]# jps
1125 WrapperSimpleApp
1719 Jps
1661 Kafka
1631 QuorumPeerMain
[root@master config]# 
```


```
[root@slave01 config]# jps
1144 WrapperSimpleApp
1754 Jps
1710 Kafka
1663 QuorumPeerMain
[root@slave01 config]# 
```

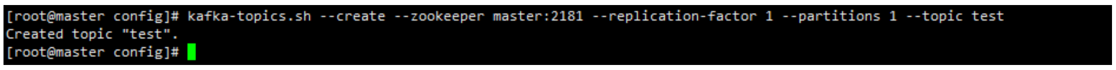

```
[root@slave02 config]#  kafka-server-start.sh -daemon server.properties
[root@slave02 config]# jps
1701 Jps
1114 WrapperSimpleApp
1658 Kafka
1628 QuorumPeerMain
[root@slave02 config]# 
```

3.4.6 对话测试

3.4.6.1 验证 Kafka 是否安装成功

3.4.6.2 在 master 创建一个名为 test 的主题 topic

3.4.6.3 kafka-topics.sh --create --zookeeper master:2181 --replication-factor 1 --partitions 1 --topic test


```
[root@master config]# kafka-topics.sh --create --zookeeper master:2181 --replication-factor 1 --partitions 1 --topic test
Created topic "test".
[root@master config]# 
```

create 表示创建一个 topic

zookeeper master:2181 表示连接 zookeeper 的服务和端口号

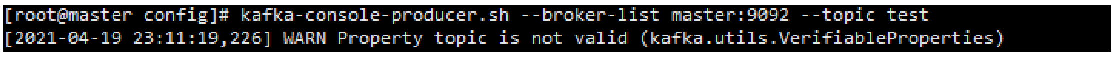
replication-factor 1 表示创建副本数量是 1

partitions 1 表示分区数量是 1

topic test 表示创建一个名为 test 的 topic 主题

3.4.6.4 在一个终端上启动一个生产者

3.4.6.5 kafka-console-producer.sh --broker-list master:9092 --topic test



```
[root@master config]# kafka-console-producer.sh --broker-list master:9092 --topic test
[2021-04-19 23:11:19,226] WARN Property topic is not valid (kafka.utils.VerifiableProperties)
```

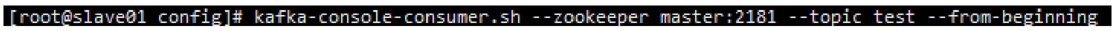
kafka-console-producer.sh 表示启动一个生产者

broker-list master:9092 表示 broker 服务列表中的 master 服务和端口号

topic test 表示向名为 test 的 topic 中生产数据

3.4.6.6 在 slave01 终端上启动一个消费者

3.4.6.7 kafka-console-consumer.sh --zookeeper master:2181 --topic test --from-beginning



```
[root@slave01 config]# kafka-console-consumer.sh --zookeeper master:2181 --topic test --from-beginning
```

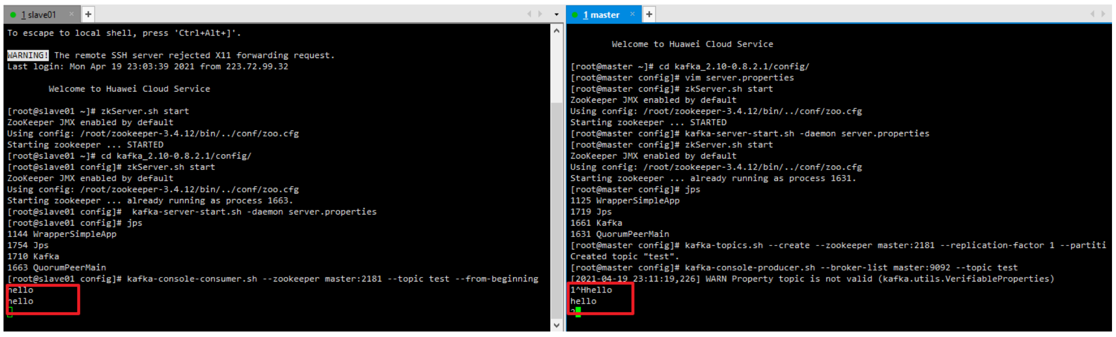
kafka-console-consumer.sh 表示启动一个消费者

zookeeper master:2181 表示连接 zookeeper 的服务和端口号

topic test 表示从名字为 test 的 topic 中消费数据

from-beginning 从最早的消息开始消费而不是从最新的消息开始

3.4.6.8 对话效果

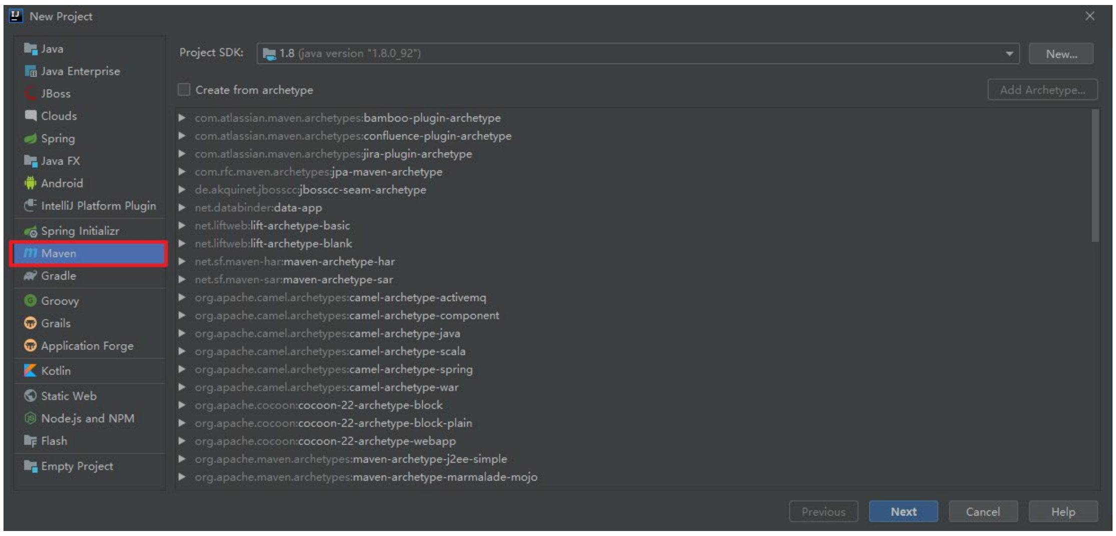


```
To escape to local shell, press 'Ctrl+Alt+]'.

WARNING! The remote SSH server rejected X11 forwarding request.
Last login: Mon Apr 19 23:03:39 2021 from 223.72.99.32

        Welcome to Huawei Cloud Service

[root@slave01 ~]# zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
[root@slave01 ~]# cd kafka_2.10-0.8.2.1/config/
[root@slave01 config]# zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Starting zookeeper ... already running as process 1663.
[root@slave01 config]#  kafka-server-start.sh -daemon server.properties
[root@slave01 config]# jps
1144 WrapperSimpleApp
1754 Jps
1710 Kafka
1663 QuorumPeerMain
[root@slave01 config]# kafka-console-consumer.sh --zookeeper master:2181 --topic test --from-beginning
hello
hello
```

```
        Welcome to Huawei Cloud Service

[root@master ~]# cd kafka_2.10-0.8.2.1/config/
[root@master config]# vim server.properties
[root@master config]# zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
[root@master config]# kafka-server-start.sh -daemon server.properties
[root@master config]# zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /root/zookeeper-3.4.12/bin/../conf/zoo.cfg
Starting zookeeper ... already running as process 1631.
[root@master config]# jps
1125 WrapperSimpleApp
1719 Jps
1661 Kafka
1631 QuorumPeerMain
[root@master config]# kafka-topics.sh --create --zookeeper master:2181 --replication-factor 1 --partiti
Created topic "test".
[root@master config]# kafka-console-producer.sh --broker-list master:9092 --topic test
[2021-04-19 23:11:19,226] WARN Property topic is not valid (kafka.utils.VerifiableProperties)
1^Hhello
hello
```

3.5 代码部分（IDEA 工具）

3.5.1 使用 IDEA 新建 maven 项目



Project SDK: 1.8 (java version "1.8.0_92")

Create from archetype

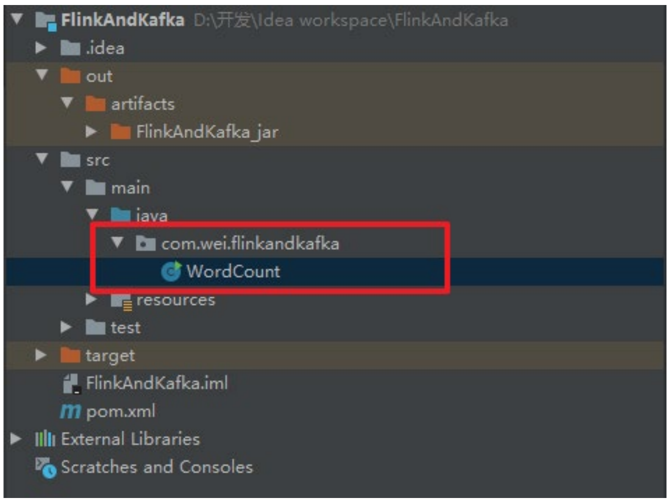
```
com.atlassian.maven.archetypes:bamboo-plugin-archetype
com.atlassian.maven.archetypes:confluence-plugin-archetype
com.atlassian.maven.archetypes:jira-plugin-archetype
com.rfc.maven.archetypes:jpa-maven-archetype
de.akquinet.jbosscc:jbosscc-seam-archetype
net.databinder:data-app
net.liftweb:lift-archetype-basic
net.liftweb:lift-archetype-blank
net.sf.maven-har:maven-archetype-har
net.sf.maven-sar:maven-archetype-sar
org.apache.camel.archetypes:camel-archetype-activemq
org.apache.camel.archetypes:camel-archetype-component
org.apache.camel.archetypes:camel-archetype-java
org.apache.camel.archetypes:camel-archetype-scala
org.apache.camel.archetypes:camel-archetype-spring
org.apache.camel.archetypes:camel-archetype-war
org.apache.cocoon:cocoon-22-archetype-block
org.apache.cocoon:cocoon-22-archetype-block-plain
org.apache.cocoon:cocoon-22-archetype-webapp
org.apache.maven.archetypes:maven-archetype-j2ee-simple
org.apache.maven.archetypes:maven-archetype-marmalade-mojo
```

3.5.2 pom.xml 文件依赖

```
        <dependencies>
            <dependency>
                <groupId>org.apache.flink</groupId>
                <artifactId>flink-clients_2.11</artifactId>
                <version>1.8.0</version>
            </dependency>
            <dependency>
                <groupId>org.apache.flink</groupId>
                <artifactId>flink-java</artifactId>
                <version>1.8.0</version>
            </dependency>
            <dependency>
                <groupId>org.apache.flink</groupId>
                <artifactId>flink-streaming-
    java_2.11</artifactId>
                <version>1.8.0</version>
            </dependency>
            <dependency>
                <groupId>org.apache.flink</groupId>
                <artifactId>flink-connector-kafka-
    0.8_2.11</artifactId>
```

```
                <version>1.8.0</version>
            </dependency>
            <dependency>
                <groupId>log4j</groupId>
                <artifactId>log4j</artifactId>
                <version>1.2.17</version>
            </dependency>
        </dependencies>
```

3.5.3 在项目文件新建 WordCount 类

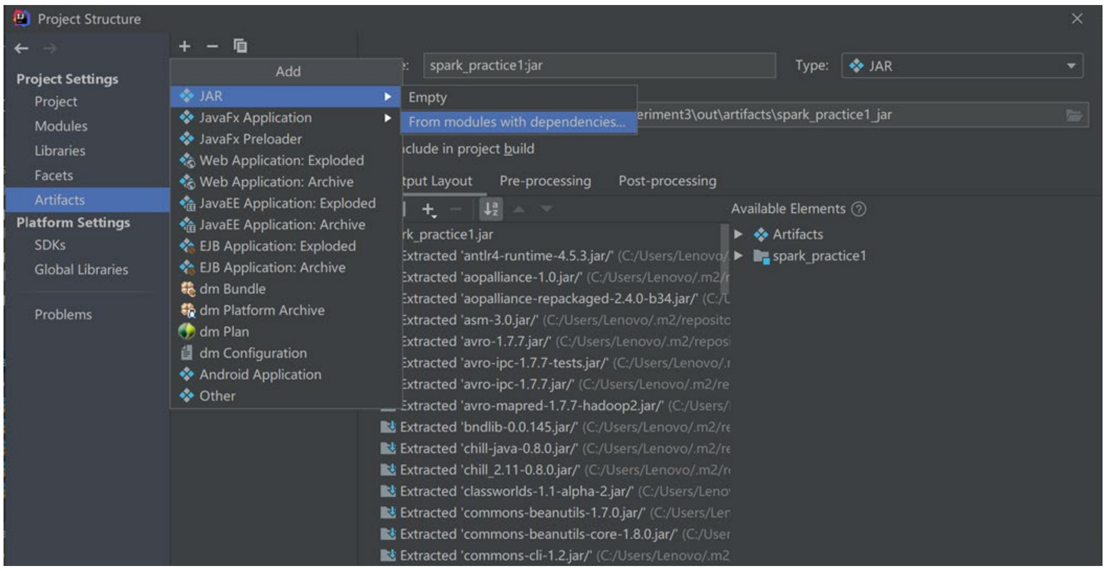

FlinkAndKafka D:\开发\Idea workspace\FlinkAndKafka
.idea
out
artifacts
FlinkAndKafka_jar
src
main
java
com.wei.flinkandkafka
WordCount
resources
test
target
FlinkAndKafka.iml
pom.xml
External Libraries
Scratches and Consoles

3.5.4 代码

```
        package com.wei.flinkandkafka;

        import
    org.apache.flink.api.common.functions.FlatMapFunction;
        import
    org.apache.flink.api.common.serialization.SimpleStringSchema;
        import org.apache.flink.api.java.tuple.Tuple2;
        import
    org.apache.flink.streaming.api.datastream.DataStream;
        import
    org.apache.flink.streaming.api.environment.StreamExecutionEnvi
    ronment;
        import
    org.apache.flink.streaming.connectors.kafka.FlinkKafkaConsumer
    08;
        import org.apache.flink.util.Collector;

        import java.util.Properties;

        /**
         * @Name: com.wei.flinkandkafka.WordCount
         * @Date: 2021/04/17
         * @Auther: weiwending
         * @Description:
         */

        public class WordCount {
            public static void main(String[] args) throws Exception
    {
                /*获取Flink运行环境*/
                StreamExecutionEnvironment          env          =
    StreamExecutionEnvironment.getExecutionEnvironment();

                /*配置Kafka连接属性*/
                Properties properties = new Properties();

                properties.setProperty("bootstrap.servers",
    "master:9092");
                properties.setProperty("zookeeper.connect",
    "master:2181");
                properties.setProperty("group.id", "1");

                FlinkKafkaConsumer08<String>   myconsumer   =   new
    FlinkKafkaConsumer08<>("test",    new    SimpleStringSchema(),
    properties);

                /*默认消费策略*/
                myconsumer.setStartFromGroupOffsets();

                DataStream<String>          dataStream             =
    env.addSource(myconsumer);

                DataStream<Tuple2<String,    Integer>>    result    =
    dataStream.flatMap(new MyFlatMapper()).keyBy(0).sum(1);

                result.print().setParallelism(3);

                env.execute();
            }

            public    static    class    MyFlatMapper    implements
    FlatMapFunction<String, Tuple2<String, Integer>> {

                @Override
                public        void        flatMap(String         s,
    Collector<Tuple2<String, Integer>> out) throws Exception {
                    /*按空格分词*/
                    String[] words = s.split(" ");
                    for (String word : words) {
                        out.collect(new Tuple2<>(word, 1));
                    }
                }
            }
    }
```

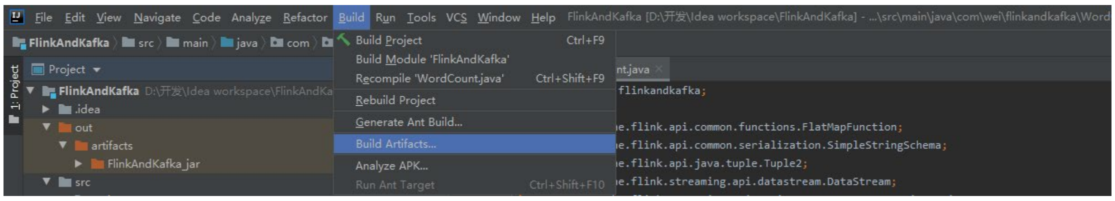
3.5.5 打包

3.5.5.1 点击 IDEA 的 File->Project Structure 中进行构建，点击 Artifacts，点击上方的加号，选择 JAR，From modules with dependencies，点击 OK 即可。

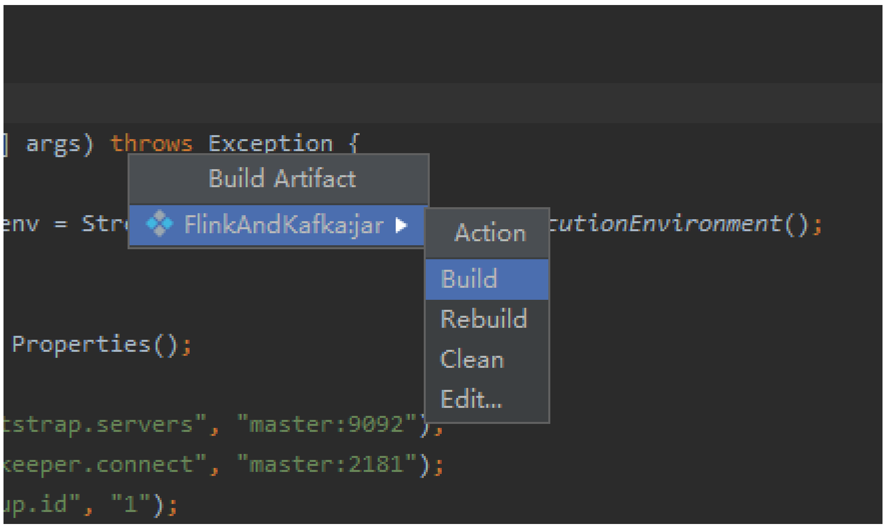


```
rk_practice1.jar
Extracted 'antlr4-runtime-4.5.3.jar/' (C:/Users/Lenovo/
Extracted 'aopalliance-1.0.jar/' (C:/Users/Lenovo/.m2/r
Extracted 'aopalliance-repackaged-2.4.0-b34.jar/' (C:/U
Extracted 'asm-3.0.jar/' (C:/Users/Lenovo/.m2/reposito
Extracted 'avro-1.7.7.jar/' (C:/Users/Lenovo/.m2/reposi
Extracted 'avro-ipc-1.7.7-tests.jar/' (C:/Users/Lenovo/.
Extracted 'avro-ipc-1.7.7.jar/' (C:/Users/Lenovo/.m2/re
Extracted 'avro-mapred-1.7.7-hadoop2.jar/' (C:/Users/
Extracted 'bndlib-0.0.145.jar/' (C:/Users/Lenovo/.m2/re
Extracted 'chill-java-0.8.0.jar/' (C:/Users/Lenovo/.m2/re
Extracted 'chill_2.11-0.8.0.jar/' (C:/Users/Lenovo/.m2/r
Extracted 'classworlds-1.1-alpha-2.jar/' (C:/Users/Leno
Extracted 'commons-beanutils-1.7.0.jar/' (C:/Users/Ler
Extracted 'commons-beanutils-core-1.8.0.jar/' (C:/User
Extracted 'commons-cli-1.2.jar/' (C:/Users/Lenovo/.m2
```

3.5.5.2 选择 Build->Build Artifacts->xxx.jar->Build


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
FlinkAndKafka [D:\开发\Idea workspace\FlinkAndKafka] - ...\src\main\java\com\wei\flinkandkafka\Word
FlinkAndKafka src main java com
Project
FlinkAndKafka D:\开发\Idea workspace\FlinkAndKa
.idea
out
artifacts
FlinkAndKafka_jar
src
Build Project Ctrl+F9
Build Module 'FlinkAndKafka'
Recompile 'WordCount.java' Ctrl+Shift+F9
Rebuild Project
Generate Ant Build...
Build Artifacts...
Analyze APK...
Run Ant Target Ctrl+Shift+F10

```
flinkandkafka;
e.flink.api.common.functions.FlatMapFunction;
e.flink.api.common.serialization.SimpleStringSchema;
e.flink.api.java.tuple.Tuple2;
e.flink.streaming.api.datastream.DataStream;
```



```
] args) throws Exception {
env = Str                              cutionEnvironment();
 Properties();
tstrap.servers", "master:9092");
keeper.connect", "master:2181");
up.id", "1");
```

3.5.5.3 然后上传 jar 包到 Flink 集群

3.5.6 运行

在集群依次启动 hadoop、zookeeper、kafka、flink

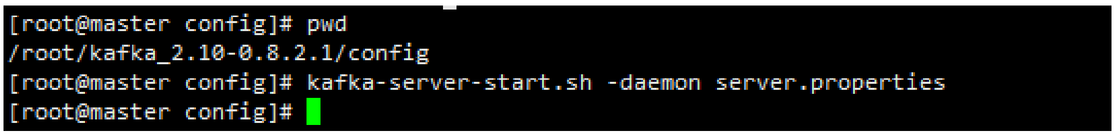
3.5.6.1 启动 hadoop，在 master 输入以下命令

```
start-all.sh
```

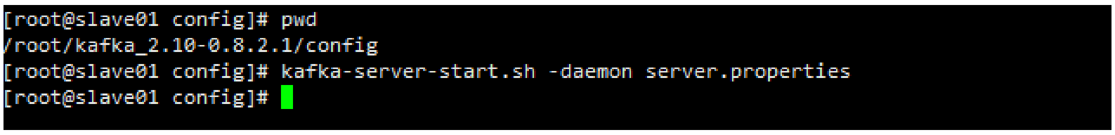
3.5.6.2 启动 zookeeper，在三个节点分别输入以下命令

```
zkServer.sh start
```

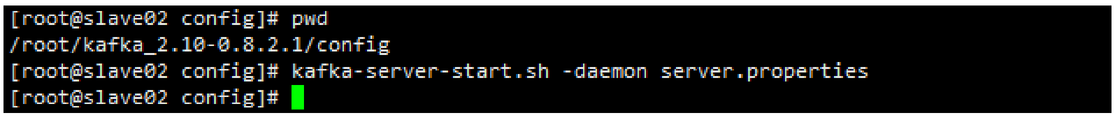
3.5.6.3 启动 kafka，在三个节点分别输入以下命令



```
[root@master config]# pwd
/root/kafka_2.10-0.8.2.1/config
[root@master config]# kafka-server-start.sh -daemon server.properties
[root@master config]#
```



```
[root@slave01 config]# pwd
/root/kafka_2.10-0.8.2.1/config
[root@slave01 config]# kafka-server-start.sh -daemon server.properties
[root@slave01 config]#
```



```
[root@slave02 config]# pwd
/root/kafka_2.10-0.8.2.1/config
[root@slave02 config]# kafka-server-start.sh -daemon server.properties
[root@slave02 config]#
```

3.5.6.4 创建一个 test topic(前面已经创建就无需创建)

```
kafka-topics.sh --create --zookeeper master:2181 --replication-factor 1 --partitions 1 --topic test
```

3.5.6.5 启动生成者

```
kafka-console-producer.sh --broker-list master:9092 --topic test
```

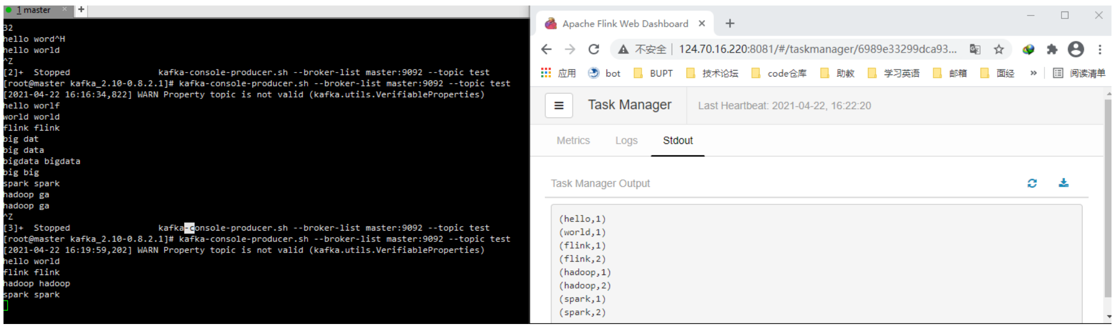
3.5.6.6 运行 jar 包

```
flink run -c com.wei.flinkandkafka.WordCount
/root/FlinkAndKafka.jar
```

3.5.6.7 访问 web 页面：http://主机 ip:8081/，点击页面 Task Managers->选择 taskmanager(master 节点)->stdout

在左侧终端不断输入数据，右端将不断有数据显示，如果没有更新，点击刷新按钮即可。

【实验结果截图】:



1 master
Apache Flink Web Dashboard
不安全 | 124.70.16.220:8081/#/taskmanager/6989e33299dca93...
应用 bot BUPT 技术论坛 code仓库 助教 学习英语 邮箱 面经 阅读清单
Task Manager
Last Heartbeat: 2021-04-22, 16:22:20
Metrics
Logs
Stdout
Task Manager Output

```
32
hello word^H
hello world
^Z
[2]+  Stopped                 kafka-console-producer.sh --broker-list master:9092 --topic test
[root@master kafka_2.10-0.8.2.1]# kafka-console-producer.sh --broker-list master:9092 --topic test
[2021-04-22 16:16:34,822] WARN Property topic is not valid (kafka.utils.VerifiableProperties)
hello worlf
world world
flink flink
big dat
big data
bigdata bigdata
big big
spark spark
hadoop ga
hadoop ga
^Z
[3]+  Stopped                 kafka-console-producer.sh --broker-list master:9092 --topic test
[root@master kafka_2.10-0.8.2.1]# kafka-console-producer.sh --broker-list master:9092 --topic test
[2021-04-22 16:19:59,202] WARN Property topic is not valid (kafka.utils.VerifiableProperties)
hello world
flink flink
hadoop hadoop
spark spark
```

```
(hello,1)
(world,1)
(flink,1)
(flink,2)
(hadoop,1)
(hadoop,2)
(spark,1)
(spark,2)
```