北京郵電大學

Beijing University of Posts and Telecommunications

华为云鲲鹏大数据基础实验体系 2
搭建 Hadoop 集群并实践 HDFS

编写负责人：鄂海红、朱一凡

参编人员：宋美娜、程祥、张忠宝、

王浩田、魏文定、刘钟允、汤子辰、罗子霄 等

华为云鲲鹏大数据基础实验体系 2：搭建 Hadoop 集群并实践 HDFS

1.1.1. 搭建 Hadoop 集群并实践 HDFS

1.1.1.1. 实验描述

在之前购买的华为云 ECS 服务器上，搭建 Hadoop 集群。并使用 idea 创建 maven 工程，完成 HDFS 文件读取实践。

1.1.1.2. 实验目的

- **学习搭建 Hadoop 集群；**
- **学习创建 maven 工程；**
- **掌握 HDFS 文件读写操作。**

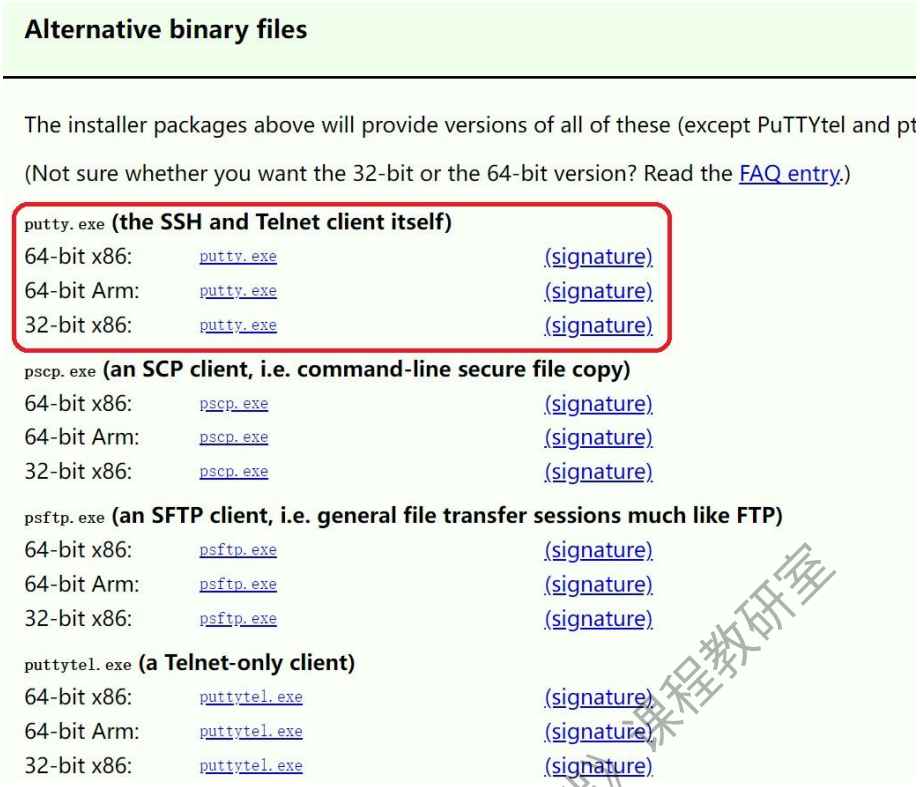
1.1.1.3. 实验步骤

1. 下载和安装远程登录传输服务器工具

Mac 同学跳过此步，进入下一步。Windows 同学建议使用远程传输工具，以防后续步骤遇到因为文件传输不完整，带来的安装报错。

下载 Putty 工具（也可以使用 Xshell 等工具）。

访问：https：//www.chiark.greenend.org.uk/~sgtatham/putty/latest.html 下载 putty.exe 并安装。



Alternative binary files

The installer packages above will provide versions of all of these (except PuTTYtel and pt

(Not sure whether you want the 32-bit or the 64-bit version? Read the FAQ entry.)

putty.exe (the SSH and Telnet client itself)		
64-bit x86:	putty.exe	(signature)
64-bit Arm:	putty.exe	(signature)
32-bit x86:	putty.exe	(signature)
pscp.exe (an SCP client, i.e. command-line secure file copy)		
64-bit x86:	pscp.exe	(signature)
64-bit Arm:	pscp.exe	(signature)
32-bit x86:	pscp.exe	(signature)
psftp.exe (an SFTP client, i.e. general file transfer sessions much like FTP)		
64-bit x86:	psftp.exe	(signature)
64-bit Arm:	psftp.exe	(signature)
32-bit x86:	psftp.exe	(signature)
puttytel.exe (a Telnet-only client)		
64-bit x86:	puttytel.exe	(signature)
64-bit Arm:	puttytel.exe	(signature)
32-bit x86:	puttytel.exe	(signature)

下载 WinSCP 工具。访问网址进行下载 https：//winscp.net/eng/docs/lang:chs

2. Hadoop 集群搭建

查看在第一章大数据实践 1 中创建完成的服务器的 IP。

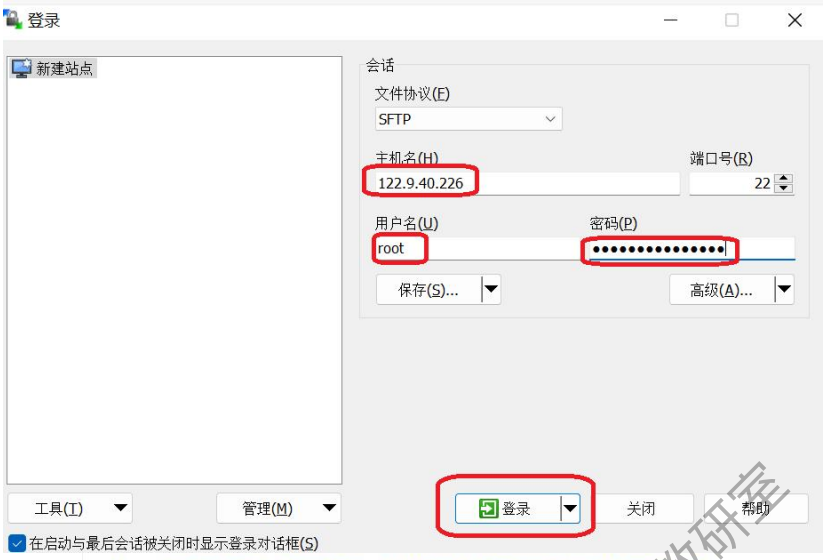
上传 Hadoop 安装包：Windows 同学使用刚刚下载的 WinSCP 进行传输，MAC 同学利用系统自带“终端”进行传输。

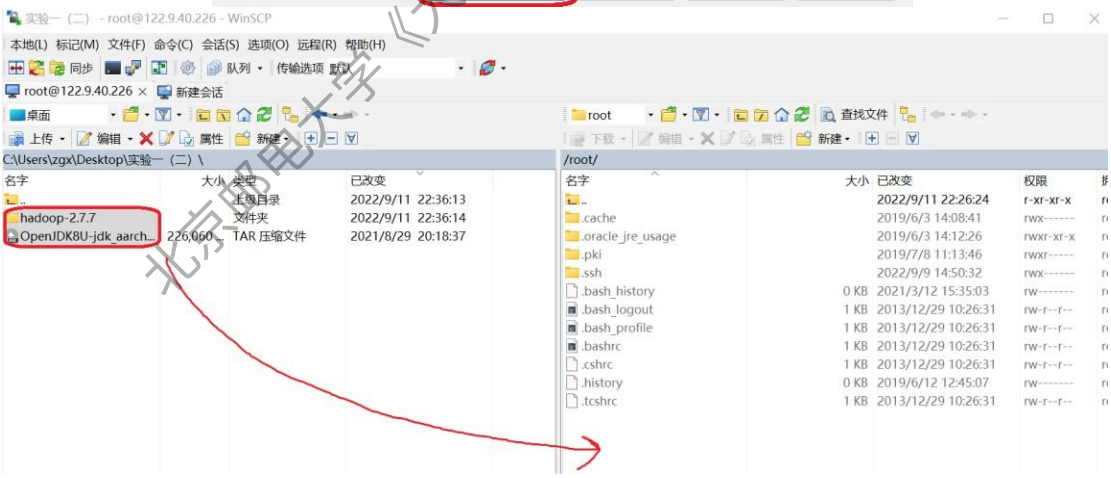
Windows 方法：

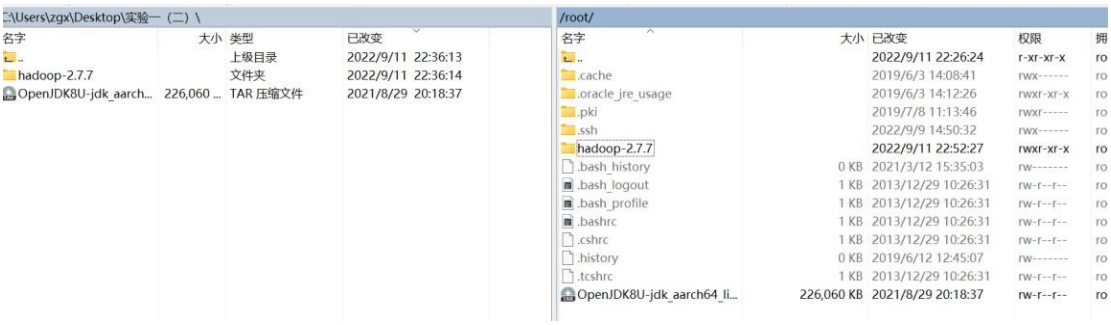
双击下载好的 WinSCP 图标 

，打开该软件。登陆页面中主机名填写刚刚查到的 ip，用户名填写 root，密码为刚刚创建服务器时设置的密码。填写完成后点击登录。

在左侧找到本实验的文件，然后拖拽到右侧（传输文件需要一定时间，请耐心等待）。







MAC 方法：打开终端，利用 scp 命令传输实验所用安装包，连接服务器时要求输入的密码为创建云服务器时设置的密码。

上传 Hadoop

scp -r ~/Desktop/hadoop-2.7.7 root@121.36.99.86:~/

```
(base) lxd@lvxiaodongdeMacBook-Pro-2 ~ % scp -r ~/Desktop/hadoop-2.7.7 root@121.36.99.86:~/
The authenticity of host '121.36.99.86 (121.36.99.86)' can't be established.
ECDSA key fingerprint is SHA256:DaQe1nfebk4XEp7ishh8ckEyw0666bzMKbc/mXncLCM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '121.36.99.86' (ECDSA) to the list of known hosts.
root@121.36.99.86's password:
Permission denied, please try again.
root@121.36.99.86's password:
hadoop.cmd                                    100% 8786   771.9KB/s   00:00
rcc                                           100% 1776   176.0KB/s   00:00
test-container-executor                       100%  200KB   1.8MB/s   00:00
mapred                                        100% 5953   524.6KB/s   00:00
yarn                                          100%   13KB 890.2KB/s   00:00
yarn.cmd                                      100%   11KB 882.3KB/s   00:00
hadoop                                        100% 6488   628.4KB/s   00:00
mapred.cmd                                    100% 6310   622.3KB/s   00:00
hdfs                                          100%   12KB   1.1MB/s   00:00
container-executor                            100%  161KB   2.0MB/s   00:00
hdfs.cmd                                      100% 7478   584.5KB/s   00:00
kms-config.sh                                 100% 5431   331.7KB/s   00:00
hadoop-config.sh                              100%   11KB 236.0KB/s   00:00
hdfs-config.cmd                               100% 1640   177.4KB/s   00:00
hadoop-config.cmd                             100% 8270   840.1KB/s   00:00
yarn-config.cmd                               100% 2131   220.6KB/s   00:00
mapred-config.cmd                             100% 1640   184.2KB/s   00:00
hdfs-config.sh                                100% 1427   147.1KB/s   00:00
yarn-config.sh                                100% 2134   222.7KB/s   00:00
mapred-config.sh                              100% 2223   200.2KB/s   00:00
httpfs-config.sh                              100% 5749   592.3KB/s   00:00
hdfs.h                                        100%   33KB   1.8MB/s   00:00
Pipes.hh                                      100% 6330   658.3KB/s   00:00
SerialUtils.hh                                100% 4514   492.2KB/s   00:00
TemplateFactory.hh                            100% 3319   331.6KB/s   00:00
StringUtils.hh                                100% 2441   291.6KB/s   00:00
yarn-daemon.sh                                100% 4295   446.3KB/s   00:00
yarn-daemons.sh                               100% 1353   146.2KB/s   00:00
distribute-exclude.sh                         100% 2752   216.8KB/s   00:00
stop-dfs.cmd                                  100% 1455    28.7KB/s   00:00
start-all.cmd                                 100% 1779   171.1KB/s   00:00
start-yarn.sh                                 100% 1347   128.6KB/s   00:00
start-all.sh                                  100% 1471    10.3KB/s   00:00
mr-jobhistory-daemon.sh                       100% 4080    27.9KB/s   00:00
hdfs-config.cmd                               100% 1640   174.7KB/s   00:00
stop-yarn.cmd                                 100% 1642   178.2KB/s   00:00
kms.sh                                        100% 3128   271.2KB/s   00:00
stop-dfs.sh                                   100% 3206   308.1KB/s   00:00
start-secure-dns.sh                           100% 1357   143.2KB/s   00:00
```

上传 OpenJDK：

scp -r ~/Desktop/OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9-0.26.0.tar root@121.36.99.86:~/

```
(base) lxd@lvxiaodongdeMacBook-Pro-2 ~ % scp -r ~/Desktop/OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_o
penj9-0.26.0.tar root@121.36.99.86:~/
root@121.36.99.86's password:
OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9-0.26.0.tar        100%  221MB   1.2MB/s   03:05
```

3. 配置服务器间免密访问

利用终端连接到上传安装包的服务器（Windows 可使用 putty，mac 用默认的终端即可），用 ssh 指令连接到服务器，ssh 的格式如下：

ssh user@ip

在本实验中 user 为 root，ip 为本节中第 2 步查到的服务器 ip。输入完指令后提示输入密码，密码为创建服务器时输入的密码，密码验证正确后登录到服务器。

```
C:\Users\zgx>ssh root@122.9.40.226
root@122.9.40.226's password:
Last failed login: Sun Sep 11 22:48:31 CST 2022 from 92.255.85.69 on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Sun Sep 11 22:28:05 2022 from 223.72.82.105

        Welcome to Huawei Cloud Service

[root@zgx-202[redacted]-0004 ~]#
```

关闭服务器上的防火墙

systemctl stop firewalld

systemctl disable firewalld

```
[root@ecs-[redacted]-0001 ~]# systemctl stop firewalld
[root@ecs-[redacted]-0001 ~]# systemctl disable firewalld
```

登录到创建的四个节点（服务器）上，分别执行如下 2 个命令：

（1）生成密钥

ssh-keygen -t rsa

提问框按默认连续回车即可，生成/root/.ssh/id_rsa.pub 文件。

```
[root@ecs-[redacted]-0001 ~]# 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:/YBIe/64ah+J8k3g3bmSoh5fKXp4qH2lPwRoREEIfHc root@ecs-[redacted]
The key's randomart image is:
+---[RSA 2048]----+
|o. ++.           |
| ..... E         |
```

（2）获得公钥

node1~node4 节点分别执行命令 cat /root/.ssh/id_rsa.pub 命令。

cat /root/.ssh/id_rsa.pub

```
[root@ecs-[redacted]-0001 ~]# cat /root/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC3CCayM/Qqk8QP8suWTZSsUEO3AUhvd[redacted]
Mph+A2eiDBzdzcTY7+HVf+oUNpUUt3aUdUQGDjSNFnvQxa7af7csHz3fju2kxwNvIxVRI[redacted]
gcPG0yqGVW1qr5srt6FV9YjG3Xx6[redacted]9GX1rMS+tT7epgM6uhRy1L30rOr[redacted]
[redacted]
fuQSRN1xBSnOQRpFi8nKmtGqhh6XHXjN8cIZHKAAe3LJ33RsViLluIVXoV7t root@ecs-[redacted]
```

将 4 个节点执行完 cat 指令后的内容复制汇总到一个新建文本中

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCg Untitled-1
1  ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCgawirqpbLAgKe/96rGAzHF5zXCAtqw0ynmweLAWexWz5EavyNoIiuLmPguRgP4Sc4w5
2  ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDdpFKkSAHhjH6c006xFLdZxNGK+2RRX58FljOHIsPuzdjVRSlADSTJWSffAclkFXIqJO
3  ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC+yn8t/1OqI36QqHamGtDJVxPW3eebN2nt5ZvKOn8mIxwp5rE3i+Mu7qHZiiL88VN+2h
4  ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCe2MGmsdScDkiPl+zRdqhfOqqIrCE4Rteexr6xUv5Mh89q2y4dZlLTnV/ez8Me/DJLWt
```

在每个节点上输入下列指令，然后将文本中的内容复制进去，从而将公钥分别复制到 node1、node2、node3、node4 的/root/.ssh/authorized_keys 中，

vim /root/.ssh/authorized_keys

```
root@zgx-20   root@zgx-20:   root@zgx-20:   root@zgx-20:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCgawirqpbLAgKe/96rGAzHF5zXCAtqw0ynmweLAWexWz5EavyN
oIiuLmPguRgP4Sc4w5gi4vpb6thkdZWpXVZ5QzLLJynCUtApOneRe2uoshPN4rRmOIqQFrboN10S/jQPCiMWFLJP
MQaDr4j+TWJJSPo7lRN7+natkYagfFyPUHeR5myf2DA91UuKK8qtevU5tWJWuU5Xtqaaq3K9oISTgcsb6mmVLgqK
vD4Yb1oNKanyqH/ZjypLFAQmNYquJw41iAF9RqpMKvb7N2222oNzSX5PdDK9AvCy3EYVKOhg2J99WU8RLpui5dZV
PVecA8ZjC117U9vfRyIWbL0YPHF7 root@zgx-20[redacted]-0004
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDdpFKkSAHhjH6c006xFLdZxNGK+2RRX58FljOHIsPuzdjVRSlA
DSTJWSffAclkFXIqJOe+d4/qzqyOkG0gwa7DMcCW5j0uFL82qmIThz7drL6wixx4dXzKp6KhWjwJijkKce5UDnwI
h5fBofEqc1+oPwd2HMxt4ftypje8ZHlX9YiwzYsYOEnU0c2LhN/UNxHo1rhpGcmNR420J2IoYsfBGQIiFEo1zc8i
uu+VIP/Sg+sHRQp0FUKt59MF9w7Up/X9LHWO3+g4DIx1Khxofo+D+iWP5oehfPh2VhWxyNGmWwnZPxfJyLw52Trk
48fssm7AQC5OrdKaQg7v1/VrQ8+x root@zgx-20[redacted]-0003
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC+yn8t/1OqI36QqHamGtDJVxPW3eebN2nt5ZvKOn8mIxwp5rE3
i+Mu7qHZiiL88VN+2hXr3YaoDgpNPmRvVbpCBSJXn3izbZ9X/P/BIf2Bw9CjUQK7rEWzhQR/BDJFeEJTcielNDN/
+ZcowAZx67Xtim07oYaPrh9ncWwsYO8dftxBgoCahu4D+qMt1xY8ks1fkNbEWs5/D6OooHtMZhBFpjaAeR9mzFzR
iaU/O3JX0U5L0trx1tHk78vcccEszmMXd+bknPHZWSqbCtDRacjSIse3PELFaZK9r0ad6LUk+s6BdI21Z0WVVa9u
9Gg6Br29txM3pJo+wx7KxyIIGVd/ root@zgx-202[redacted]-0002
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCe2MGmsdScDkiPl+zRdqhfOqqIrCE4Rteexr6xUv5Mh89q2y4d
ZlLTnV/ez8Me/DJLWtEBP9iMEJNjjmdqem0CavHDLhOxq92wowYa/GtRNCtwDvOazILrFzIHuz4EW8zwquudDuYe
jib3zpRhOEzw0u0FVUo2l/KIHzYtBNAMrDJj7XwjMgMehIJeqSHMjlrSBl1m3GfIQrnKrU730nFWP6MO+dAs8iyM
wfQtXePl0pEuZnyOzXhSHn6yERyAsPK3B8g95SrUMhxOsLYip7m6rfSu/cRQQSusE4I1j9DhfaTr6wgaw0PjG36V
8IinC0Sz5XYlTlLr4Fg+GOqq9sNl root@zgx-202[redacted]-0001
~
~
```

输入指令后，英文状态下按“i”键，进入输入模式，将文本中的全部内容复制进去后，按“Esc”键退出编辑模式，然后再英文状态下输入“:”（冒号），然后输入 wq，最后按回车完成编辑。（更多 vim 用法请参阅网上教程）

（3）查看内网 ip

在 node1~node4 节点分别执行命令 ifconfig，查看每个节点的内网 ip。

```
[root@zgx-202[redacted]-0004 logs]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.135  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::f816:3eff:fe72:c951  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:72:c9:51  txqueuelen 1000  (Ethernet)
        RX packets 624262  bytes 48225590 (45.9 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 936563  bytes 2781546594 (2.5 GiB)
        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 737  bytes 468454 (457.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 737  bytes 468454 (457.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

编辑 hosts 文件，加入 node1~node4 对应 IP 及 node 节点名。

vim /etc/hosts

格式如下

node1_ip node1
node2_ip node2
node3_ip node3
node4_ip node4

其中本节点的 ip 用刚刚查到的内网 ip，其他节点的 ip 用外网 ip（之前服务器管理页面显示的服务器的 ip）

在 node4 查询内网 ip 并修改 hosts 文件后，hosts 的截图如下所示

```
::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       zgx-202[redacted]-0004     zgx-202[redacted]-0004

122.9.46.217    node1
114.116.201.211 node2
114.116.195.153 node3
192.168.0.135   node4
```

（4）检测节点间是否能无密访问

所有节点加入后 IP 映射后，node1~node4 节点分别执行命令 ssh node1~node4，选择 yes 后，确保能够无密码跳转到目的节点。node1 节点无密码跳转到 node4 节点如下图，其余同理。

```
[root@zgx-202[redacted]-0001 ~]# ssh node4
The authenticity of host 'node4 (122.9.40.226)' can't be established.
ECDSA key fingerprint is SHA256:/A9wSPkH5tG7WnwUY9b8DP3r3yEHnvCgXE28TTIcWP0.
ECDSA key fingerprint is MD5:20:e8:ff:f6:99:55:be:d1:07:fe:27:58:12:ce:71:6a.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'node4,122.9.40.226' (ECDSA) to the list of known hosts.
Last login: Tue Sep 13 12:58:09 2022 from 114.116.195.153

        Welcome to Huawei Cloud Service

[root@zgx-202[redacted]-0004 ~]#
```

4. 安装 OpenJDK

登录到上传安装包的节点，执行如下命令，将 jdk 安装包拷贝到/usr/lib/jvm 目录下。

cp OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9-0.26.0.tar /usr/lib/jvm/

执行如下命令，用于该节点将安装包分发到剩余 3 个节点（注意：如下命令中斜体的 *node1*、*node2*、*node3*，需要根据自己的实际情况进行替换为自己需要分发的节点名）：

scp OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9-0.26.0.tar root@*node1*:/usr/lib/jvm/

scp OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9-0.26.0.tar root@*node2*:/usr/lib/jvm/

scp OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9-0.26.0.tar root@*node3*:/usr/lib/jvm/

```
[root@zgx-20[redacted]-0004 ~]# ls
hadoop-2.7.7  OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9-0.26.0.tar
[root@zgx-20[redacted]-0004 ~]# cp OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9-0.26.0.tar /usr/lib/jvm/
[root@zgx-20[redacted]-0004 ~]# scp OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9-0.26.0.tar root@node1:/usr/lib/jvm/
[root@zgx-20[redacted]-0004 ~]# scp OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9-0.26.0.tar root@node2:/usr/lib/jvm/
[root@zgx-20[redacted]-0004 ~]# scp OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9-0.26.0.tar root@node3:/usr/lib/jvm/
OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9-0.26.0.tar                          7%   17MB   2.6MB/s   01:18 ETA
```

在 node1~node4 四个节点分别执行命令

cd /usr/lib/jvm/

tar -vxf OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9- 0.26.0.tar

```
[root@zgx-26        -0001 ~]# cd /usr/lib/jvm/
[root@zgx-26        -0001 jvm]# tar -vxf OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9-0.26.0.tar
jdk8u292-b10/
jdk8u292-b10/THIRD_PARTY_README
jdk8u292-b10/sample/
jdk8u292-b10/sample/jmx/
jdk8u292-b10/sample/jmx/jmx-scandir/
jdk8u292-b10/sample/jmx/jmx-scandir/truststore
jdk8u292-b10/sample/jmx/jmx-scandir/manifest.mf
jdk8u292-b10/sample/jmx/jmx-scandir/build.xml
jdk8u292-b10/sample/jmx/jmx-scandir/build.properties
jdk8u292-b10/sample/jmx/jmx-scandir/logging.properties
jdk8u292-b10/sample/jmx/jmx-scandir/nbproject/
jdk8u292-b10/sample/jmx/jmx-scandir/nbproject/project.xml
jdk8u292-b10/sample/jmx/jmx-scandir/nbproject/netbeans-targets.xml
```

在 node1~node4 四个节点上编辑/etc/profile 增加如下的配置

vim /etc/profile

```
[root@ecs-        -0001 jvm]# vim /etc/profile
```

添加下面一行到文件末尾

export JAVA_HOME=/usr/lib/jvm/jdk8u292-b10

```
for i in /etc/profile.d/*.sh /etc/profile.
    if [ -r "$i" ]; then
        if [ "${-#*i}" != "$-" ]; then
            . "$i"
        else
            . "$i" >/dev/null
        fi
    fi
done

unset i
unset -f pathmunge

export JAVA_HOME=/usr/lib/jvm/jdk8u191-b12
```

让配置生效

source /etc/profile

然后在各个节点上确认 java 版本

java -version

```
[root@zgx-202        -0002 jvm]# vim /etc/profile
[root@zgx-202        -0002 jvm]# source /etc/profile
[root@zgx-202        -0002 jvm]# java -version
openjdk version "1.8.0_232"
OpenJDK Runtime Environment (build 1.8.0_232-b09)
OpenJDK 64-Bit Server VM (build 25.232-b09, mixed mode)
```

5. 安装 Hadoop：

登录上传安装包的节点，复制 hadoop 安装包到/home/modules 下

cp -r hadoop-2.7.7 /home/modules/

cd /home/modules/

```
[root@zgx-20[redacted]-0004 ~]# ls
hadoop-2.7.7  OpenJDK8U-jdk_aarch64_linux_openj9_8u292b10_openj9-0.26.0.tar
[root@zgx-20[redacted]-0004 ~]# cp -r hadoop-2.7.7 /home/modules/
[root@zgx-20[redacted]-0004 ~]# cd /home/modules/
[root@zgx-20[redacted]-0004 modules]# ls
bin  etc  hadoop-2.7.7  include  lib  libexec  LICENSE.txt  NOTICE.txt  README.txt  sbin  share
```

（1）配置 hadoop 环境变量

vim /home/modules/hadoop-2.7.7/etc/hadoop/hadoop-env.sh

在最后一行加入

export JAVA_HOME=/usr/lib/jvm/jdk8u292-b10

```
# A string representing this instance of hadoop. $USER by default.
export HADOOP_IDENT_STRING=$USER

export JAVA_HOME=/usr/lib/jvm/jdk8u292-b10
```

执行下列命令，配置 hadoop core-site.xml 文件：

vim /home/modules/hadoop-2.7.7/etc/hadoop/core-site.xml

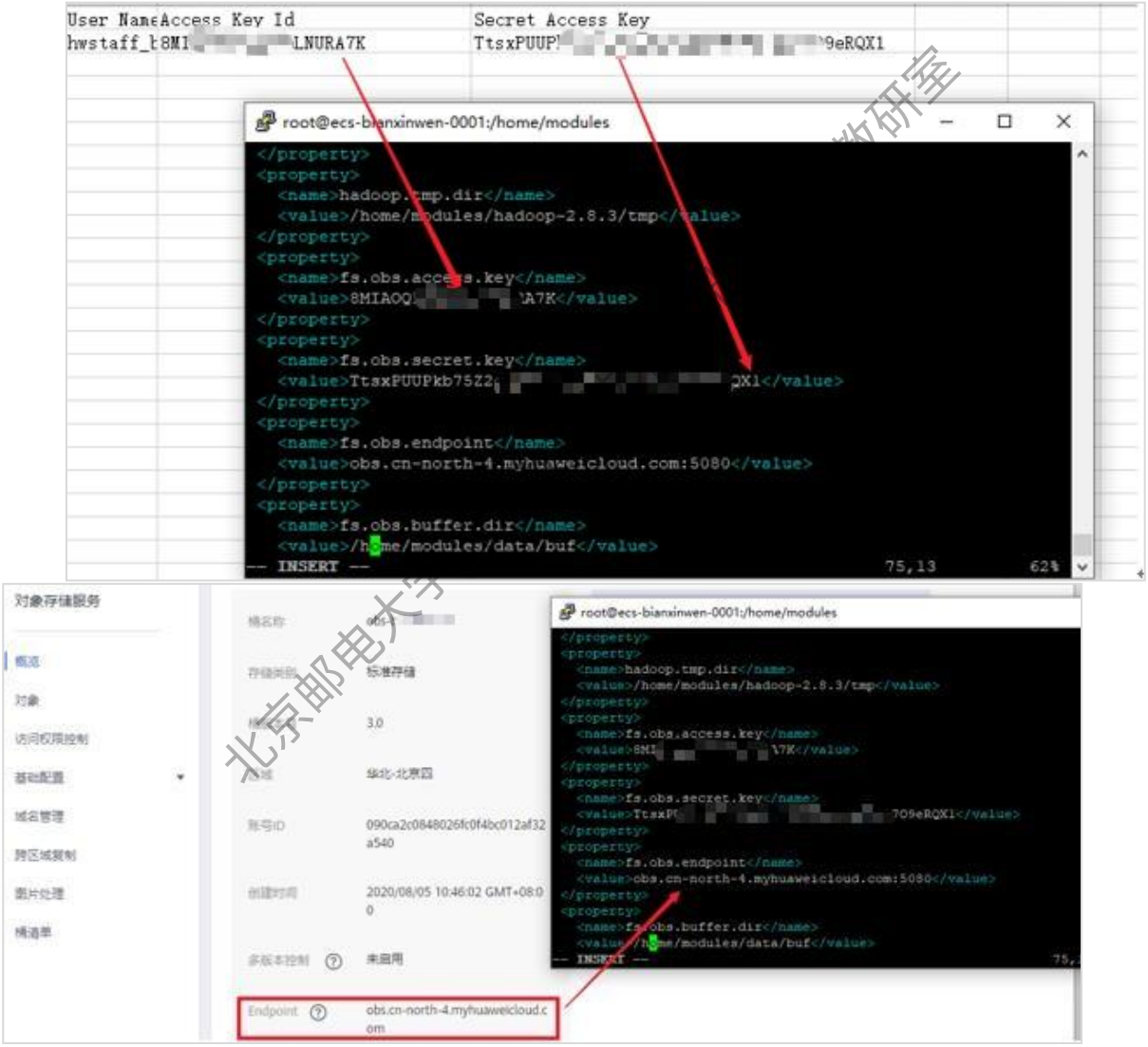
参数配置如下：

```
<configuration>
<property>
    <name>fs.obs.readahead.inputstream.enabled</name>
    <value>true</value>
  </property>
  <property>
    <name>fs.obs.buffer.max.range</name>
    <value>6291456</value>
  </property>
  <property>
    <name>fs.obs.buffer.part.size</name>
    <value>2097152</value>
  </property>
  <property>
    <name>fs.obs.threads.read.core</name>
    <value>500</value>
  </property>
  <property>
    <name>fs.obs.threads.read.max</name>
    <value>1000</value>
  </property>
  <property>
    <name>fs.obs.write.buffer.size</name>
    <value>8192</value>
  </property>
  <property>
    <name>fs.obs.read.buffer.size</name>
    <value>8192</value>
  </property>
  <property>
    <name>fs.obs.connection.maximum</name>
    <value>1000</value>
  </property>
  <property>
```

```
        <name>fs.defaultFS</name>
        <value>hdfs://node4:8020</value>
    </property>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/home/modules/hadoop-2.8.3/tmp</value>
     </property>
     <property>
        <name>fs.obs.access.key</name>
        <value>NVONVZGSZ2PPZS7PRCV3</value>
</property>
     <property>
        <name>fs.obs.secret.key</name>
        <value>MFKSvUrjDNQyklX29uSOQ7YDadvQRfaTy207AmLa</value>
     </property>
     <property>
        <name>fs.obs.endpoint</name>
        <value>obs.cn-north-4.myhuaweicloud.com</value>
     </property>
     <property>
        <name>fs.obs.buffer.dir</name>
        <value>/home/modules/data/buf</value>
     </property>
     <property>
        <name>fs.obs.impl</name>
        <value>org.apache.hadoop.fs.obs.OBSFileSystem</value>
     </property>
     <property>
        <name>fs.obs.connection.ssl.enabled</name>
        <value>false</value>
     </property>
     <property>
        <name>fs.obs.fast.upload</name>
        <value>true</value>
     </property>
     <property>
        <name>fs.obs.socket.send.buffer</name>
        <value>65536</value>
     </property>
     <property>
        <name>fs.obs.socket.recv.buffer</name>
        <value>65536</value>
     </property>
     <property>
        <name>fs.obs.max.total.tasks</name>
        <value>20</value>
     </property>
     <property>
        <name>fs.obs.threads.max</name>
    <value>20</value>
     </property>
    </configuration>
```

注：fs.defaultFS、fs.obs.access.key、fs.obs.secret.key、fs.obs.endpoint 需根据实际情况修改（后三者的具体值查阅上一实验保存在本地的文件）。

```
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://ecs-[illegible]-0001:8020</value>
</property>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/home/modules/hadoop-2.8.3/tmp</value>
</property>
```



修改后的文件部分截图如下

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

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

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

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

<configuration>
<property>
   <name>fs.obs.readahead.inputstream.enabled</name>
   <value>true</value>
 </property>
 <property>
   <name>fs.obs.buffer.max.range</name>
   <value>6291456</value>
 </property>
 <property>
   <name>fs.obs.buffer.part.size</name>
```

（2）配置 hdfs-site.xml

vim /home/modules/hadoop-2.7.7/etc/hadoop/hdfs-site.xml

参数配置如下：

```xml
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.namenode.secondary.http-address</name>
    <value>node4:50090</value>
  </property>
  <property>
    <name>dfs.namenode.secondary.https-address</name>
    <value>node4:50091</value>
  </property>
</configuration>
```

注意：node 名称使用自己所在的服务器名称。

```
<configuration>
 <property>
   <name>dfs.replication</name>
   <value>3</value>
 </property>
 <property>
   <name>dfs.namenode.secondary.http-address</name>
   <value>node4:50090</value>
 </property>
 <property>
   <name>dfs.namenode.secondary.https-address</name>
   <value>node4:50091</value>
 </property>
</configuration>
```

（3）配置 yarn-site.xml

vim /home/modules/hadoop-2.7.7/etc/hadoop/yarn-site.xml

参数配置如下：

```
<configuration>
<property>
  <name>yarn.nodemanager.local-dirs</name>
<value>/home/nm/localdir</value>
</property>
<property>
  <name>yarn.nodemanager.resource.memory-mb</name>
  <value>28672</value>
 </property>
  <property>
  <name>yarn.scheduler.minimum-allocation-mb</name>
  <value>3072</value>
 </property>
  <property>
  <name>yarn.scheduler.maximum-allocation-mb</name>
  <value>28672</value>
 </property>
  <property>
  <name>yarn.nodemanager.resource.cpu-vcores</name>
  <value>38</value>
 </property>
  <property>
  <name>yarn.scheduler.maximum-allocation-vcores</name>
  <value>38</value>
 </property>
 <property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
 </property>
 <property>
  <name>yarn.resourcemanager.hostname</name>
  <value>node4</value>
 </property>
 <property>
  <name>yarn.log-aggregation-enable</name>
  <value>true</value>
 </property>
```

```
  <property>
    <name>yarn.log-aggregation.retain-seconds</name>
    <value>106800</value>
  </property>
  <property>
      <name>yarn.nodemanager.vmem-check-enabled</name>
      <value>false</value>
      <description>Whether virtual memory limits will be enforced for containers</description>
  </property>
  <property>
      <name>yarn.nodemanager.vmem-pmem-ratio</name>
      <value>4</value>
      <description>Ratio between virtual memory to physical
memory when setting memory limits for
containers</description>
  </property>
  <property>
      <name>yarn.resourcemanager.scheduler.class</name>
    <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler</value>
    </property>
<property>
        <name>yarn.log.server.url</name>
        <value>http://node4:19888/jobhistory/logs</value>
</property>
</configuration>
```

注意：node4 替换为自己所在的节点名。

（4）配置 mapred-sit.xml

执行下列命令

```
cd /home/modules/hadoop-2.7.7/etc/hadoop/ mv
mapred-site.xml.template mapred-site.xml
vim /home/modules/hadoop-2.7.7/etc/hadoop/mapred-site.xml
```

参数配置如下：

```
<configuration>
  <property>
      <name>mapreduce.framework.name</name>
      <value>yarn</value>
  </property>
  <property>
      <name>mapreduce.jobhistory.address</name>
      <value>node4:10020</value>
  </property>
<property>
      <name>mapreduce.jobhistory.webapp.address</name>
      <value>node4:19888</value>
  </property>
  <property>
      <name>mapred.task.timeout</name>
```

```
        <value>1800000</value>
    </property>
</configuration>
```

注意：node4 为自己实际节点名。

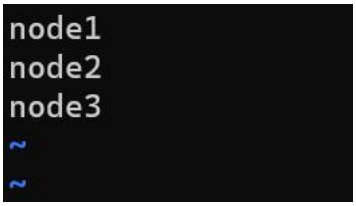
（5）配置 slaves

vim /home/modules/hadoop-2.7.7/etc/hadoop/slaves

编辑内容如下：

node1
node2
node3

#配置内容为其余三个节点的名，每行一个，共 3 行(删掉原有的部分)。



（6）分发 hadoop 包到其余节点

首先在其余 3 个节点上创建目标文件夹

mkdir /home/modules/

在安装包所在节点上，用下列命令分发 Hadoop 到其余节点（此处安装包在 node4 为例）。

#分发 Hadoop 到节点 1

scp -r /home/modules/hadoop-2.7.7 root@node1:/home/modules/

#分发 Hadoop 到节点 2

scp -r /home/modules/hadoop-2.7.7 root@node2:/home/modules/

#分发 Hadoop 到节点 3

scp -r /home/modules/hadoop-2.7.7 root@node3:/home/modules/

根据实际情况更改上面的分发的节点。

执行命令前确保其他节点的/home/modules 下没有 hadoop-2.7.7 文件夹，如有，用下面的指令删除：

rm -rf /home/modules/hadoop-2.7.7

（7）配置环境变量

node1~node4 四个节点下执行下列命令：

vim /etc/profile

添加如下 4 行：

export HADOOP_HOME=/home/modules/hadoop-2.7.7
export PATH=$JAVA_HOME/bin:$PATH
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
export HADOOP_CLASSPATH=/home/modules/hadoop-2.7.7/share/hadoop/tools/lib/*:$HADOOP_CLASSPATH

node1 节点添加如图所示，其余节点同理。

```
root@ecs-[redacted]-0001:~
fi

for i in /etc/profile.d/*.sh /etc/profile.d/sh.local ; do
    if [ -r "$i" ]; then
        if [ "${-#*i}" != "$-" ]; then
            . "$i"
        else
            . "$i" >/dev/null
        fi
    fi
done

unset i
unset -f pathmunge

export JAVA_HOME=/usr/lib/jvm/jdk8u191-b12

export HADOOP_HOME=/home/modules/hadoop-2.8.3
export PATH=$JAVA_HOME/bin:$PATH
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
export HADOOP_CLASSPATH=/home/modules/hadoop-2.8.3/share/hadoop/tools/lib/*:$HAD
OOP_CLASSPATH
```

node1~node4 四个节点下执行下列命令：

source /etc/profile

```
[root@ecs-[redacted]-0001 ~]# vim /etc/profile
[root@ecs-[redacted]-0001 ~]# source /etc/profile
[root@ecs-[redacted]-0002 ~]# vim /etc/profile
[root@ecs-[redacted]-0002 ~]# source /etc/profile
[root@ecs-[redacted]-0003 ~]# vim /etc/profile
[root@ecs-[redacted]-0003 ~]# source /etc/profile
[root@ecs-[redacted]-0004 ~]# vim /etc/profile
[root@ecs-[redacted]-0004 ~]# source /etc/profile
```

node1~node4 四个节点下执行下列命令：

chmod -R 777 /home/modules/hadoop-2.7.7

在上传安装包的节点执行下列命令。

hadoop namenode -format

启动 hadoop：start-all.sh

```
[root@ecs-62b7-0001 ~]# start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
21/09/16 15:44:55 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes
Starting namenodes on [ecs-62b7-0001]
ecs-62b7-0001: starting namenode, logging to /home/modules/hadoop-2.7.7/logs/hadoop-root-namenode-ecs-62b7-0001.out
localhost: starting datanode, logging to /home/modules/hadoop-2.7.7/logs/hadoop-root-datanode-ecs-62b7-0001.out
ecs-62b7-0003: starting datanode, logging to /home/modules/hadoop-2.7.7/logs/hadoop-root-datanode-ecs-62b7-0003.out
ecs-62b7-0004: starting datanode, logging to /home/modules/hadoop-2.7.7/logs/hadoop-root-datanode-ecs-62b7-0004.out
ecs-62b7-0002: starting datanode, logging to /home/modules/hadoop-2.7.7/logs/hadoop-root-datanode-ecs-62b7-0002.out
Starting secondary namenodes [ecs-62b7-0001]
ecs-62b7-0001: starting secondarynamenode, logging to /home/modules/hadoop-2.7.7/logs/hadoop-root-secondarynamenode-ecs-62b7-0001.
21/09/16 15:45:11 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes
starting yarn daemons
starting resourcemanager, logging to /home/modules/hadoop-2.7.7/logs/yarn-root-resourcemanager-ecs-62b7-0001.out
ecs-62b7-0003: starting nodemanager, logging to /home/modules/hadoop-2.7.7/logs/yarn-root-nodemanager-ecs-62b7-0003.out
localhost: starting nodemanager, logging to /home/modules/hadoop-2.7.7/logs/yarn-root-nodemanager-ecs-62b7-0001.out
ecs-62b7-0004: starting nodemanager, logging to /home/modules/hadoop-2.7.7/logs/yarn-root-nodemanager-ecs-62b7-0004.out
ecs-62b7-0002: starting nodemanager, logging to /home/modules/hadoop-2.7.7/logs/yarn-root-nodemanager-ecs-62b7-0002.out
```

输入 *jps* 若为一下截图，则为 hadoop 安装成功。（截图形成实验结果。）

```
[root@zgx-202        -0004 hadoop-2.7.7]# jps
19038 ResourceManager
18854 SecondaryNameNode
18636 NameNode
19309 Jps
```

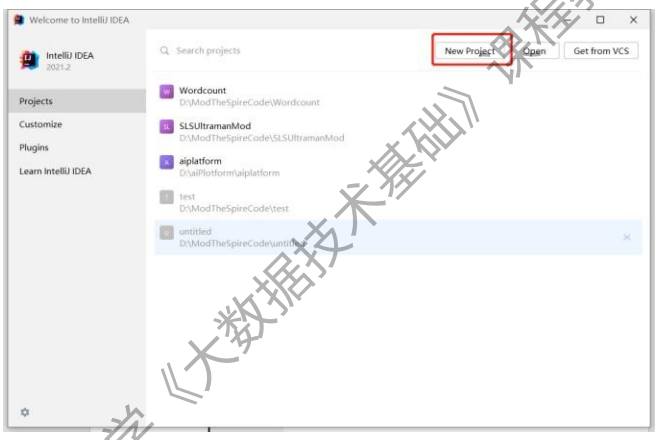
登录到子节点的服务器，输入 jps，若进程为下面的截图则启动成功。（截图形成实验结果。）

```
[root@zgx-202        -0001 ~]# jps
12711 NodeManager
12869 Jps
12601 DataNode
```

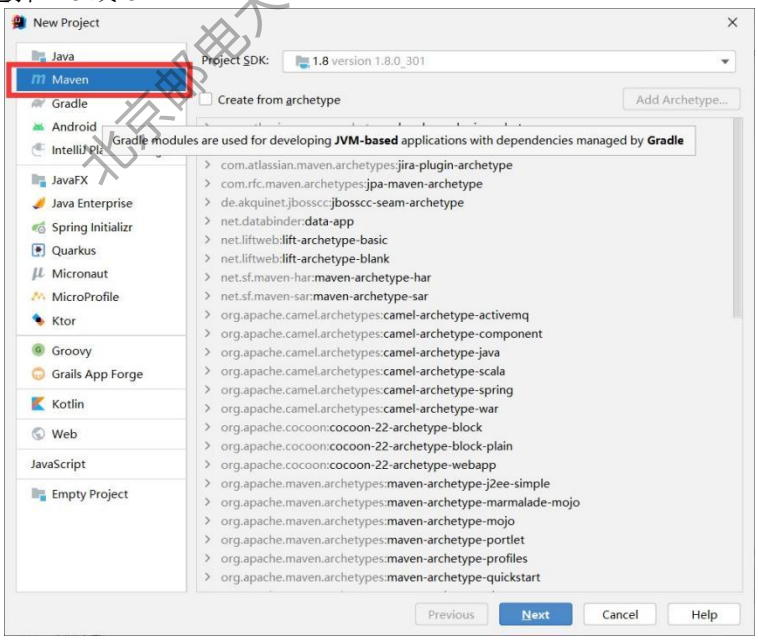
若缺少其中的进程，进入 hadoop-2.7.7/logs 文件夹，打开对应进程的 log 文件，查看失败原因、借助搜索引擎排查错误。

6. 创建 maven 工程

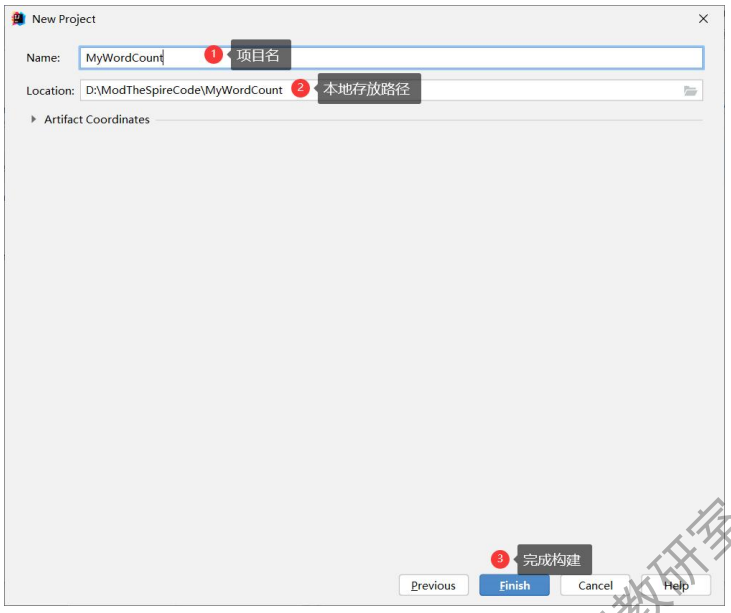
步骤 1：创建项目，打开 IDEA（IDEA 需要在自己电脑上安装），创建工程：



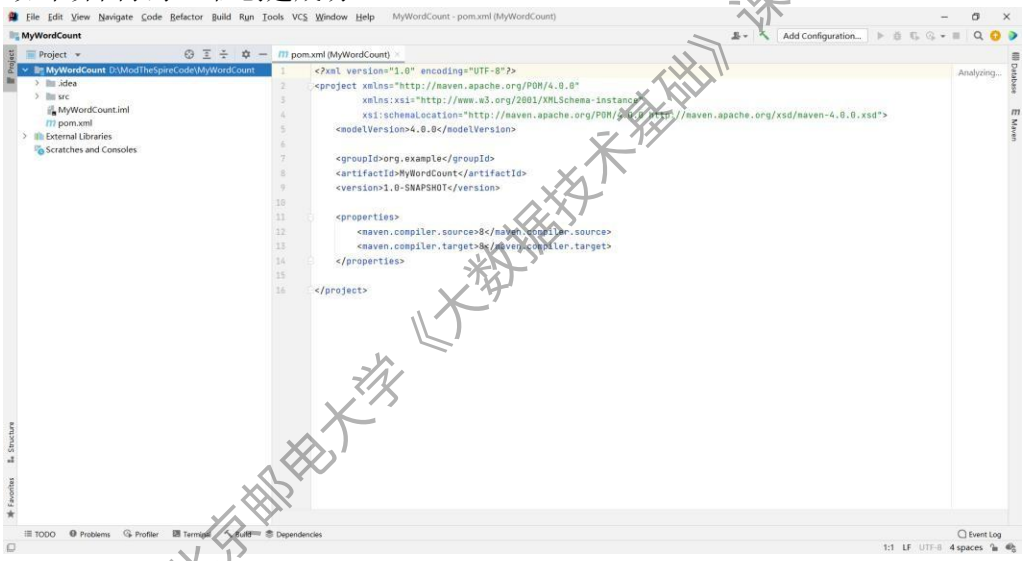
Java 版本选择 1.8 或 8



点击 New Project，然后点击选择 Maven 项目，点击 Next，之后输入 Name，选择 Location，并点击 Finish。



进入如下界面表示工程创建成功：



步骤 2：依赖设置：

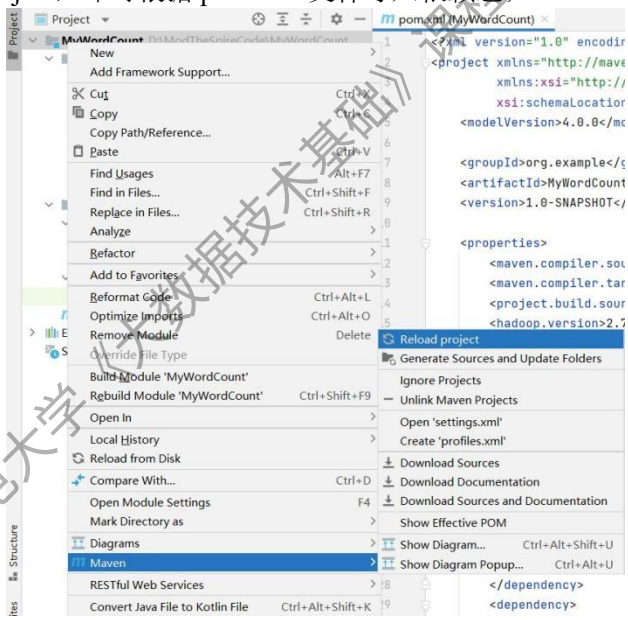
1）在 pom.xml 文件中找到 properties 配置项，新增 hadoop 版本号（此处对应 hadoop 安装版本）；

```
<properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <hadoop.version>2.7.7</hadoop.version>
</properties>
```

2）找到 dependency 配置项（若无则手动添加），添加如下图标红部分的配置，这部分是 hadoop 的依赖，${hadoop.version}表示上述配置的 hadoop.version 变量；

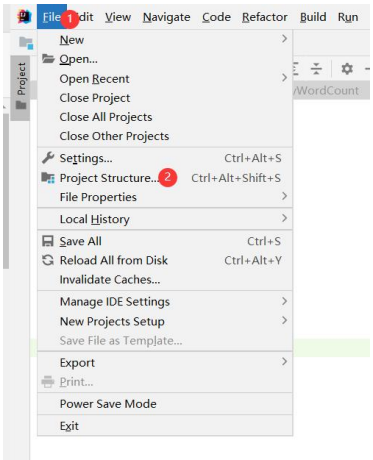
```
<properties>
  <!--1.8 或 8 都行-->
  <maven.compiler.source>8</maven.compiler.source>
  <maven.compiler.target>8</maven.compiler.target>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <hadoop.version>2.7.7</hadoop.version>
</properties>

<dependencies>
  <dependency>
    <!--需要在src/main/resources路径下配置log4jproperties文件(根据报错，百度配置)-->
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
  </dependency>
  <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-client</artifactId>
    <version>${hadoop.version}</version>
  </dependency>
  <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-common</artifactId>
    <version>${hadoop.version}</version>
  </dependency>
  <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-hdfs</artifactId>
    <version>${hadoop.version}</version>
  </dependency>
</dependencies>
```

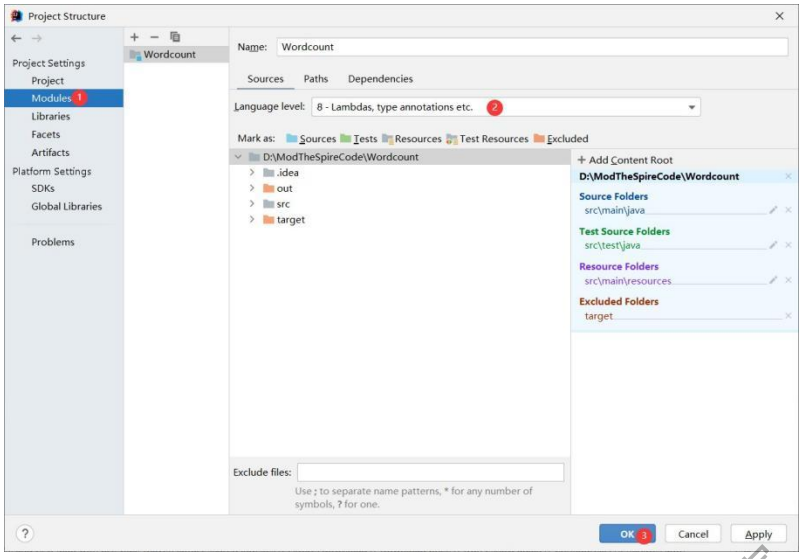
一般修改 pom.xml 文件后，会提示 enable auto-import，点击即可，如果没有提示，则可以右键点击工程名，依次选择 Maven—->Reload project，即可根据 pom.xml 文件导入依赖包；



步骤 3：设置语言环境：

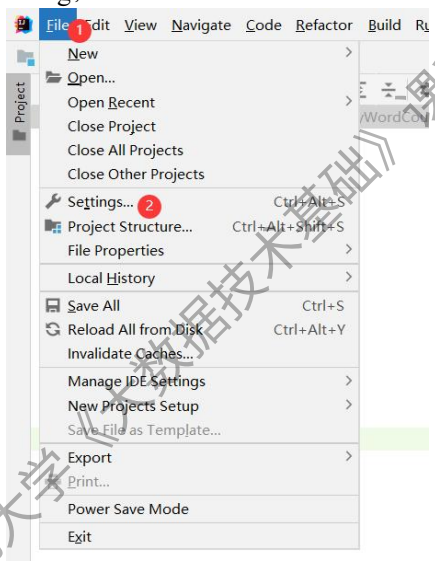
1)设置语言环境 language level，点击菜单栏中的 file，选择 Project Structure；弹出如下对话框，选择 Modules，选择 Language level 为 8，然后点击 Apply，点击 OK；



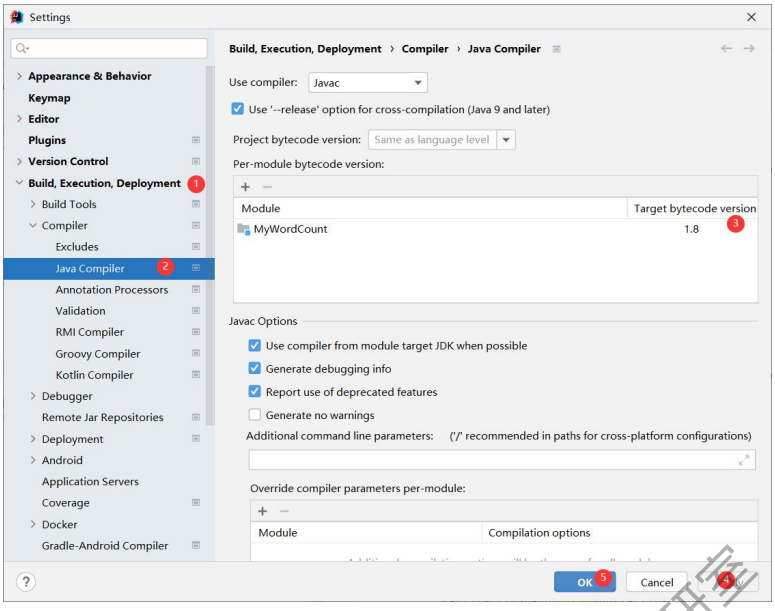


步骤 4：设置 java Compiler 环境：

1）点击菜单栏中的 file，选择Setting；



2）弹出如下对话框，依次选择 Build，Execution—->Compiler—->Java Compiler，设置图中的 Project bytecode version 为 1.8，设置图中的 Target bytecode version 为 1.8，然后依次点击 Apply 和 OK；

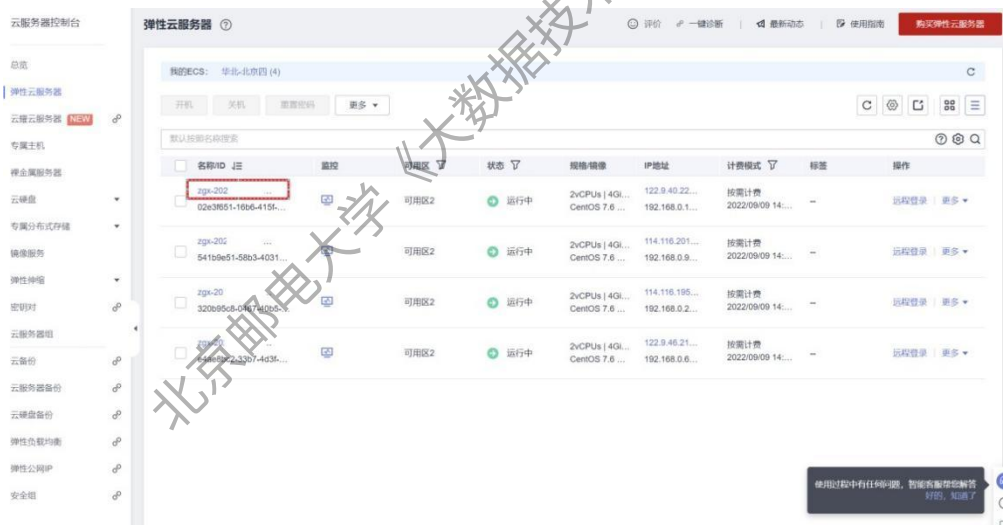


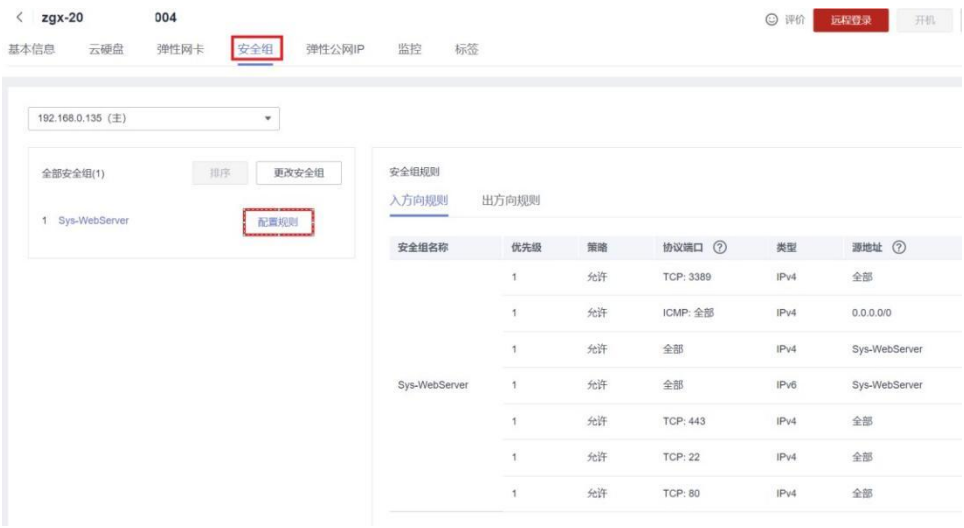
之后就可以开始程序编写。

7. java 实现 HDFS 文件读写

在华为云服务器管理处开放 8080 端口，点击主节点所在的服务器

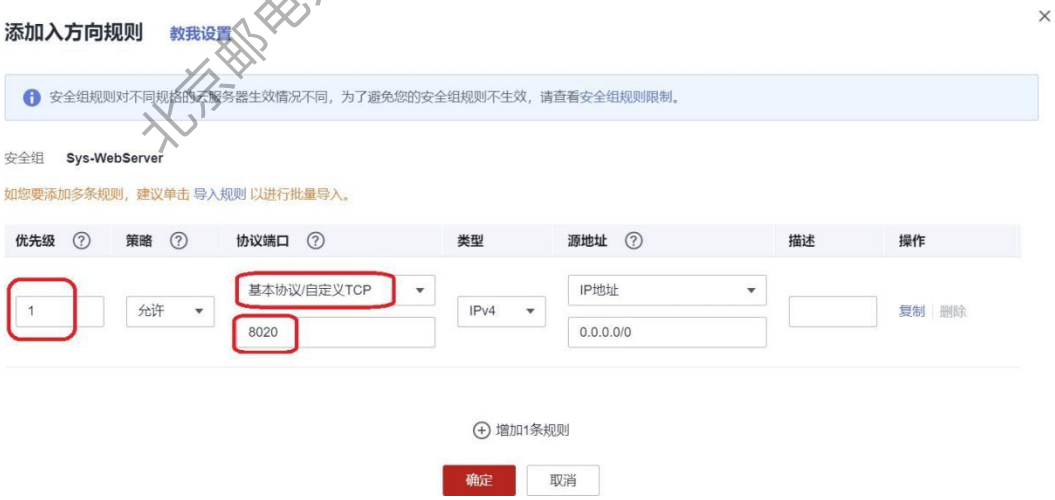
点击安全组，点击配置规则



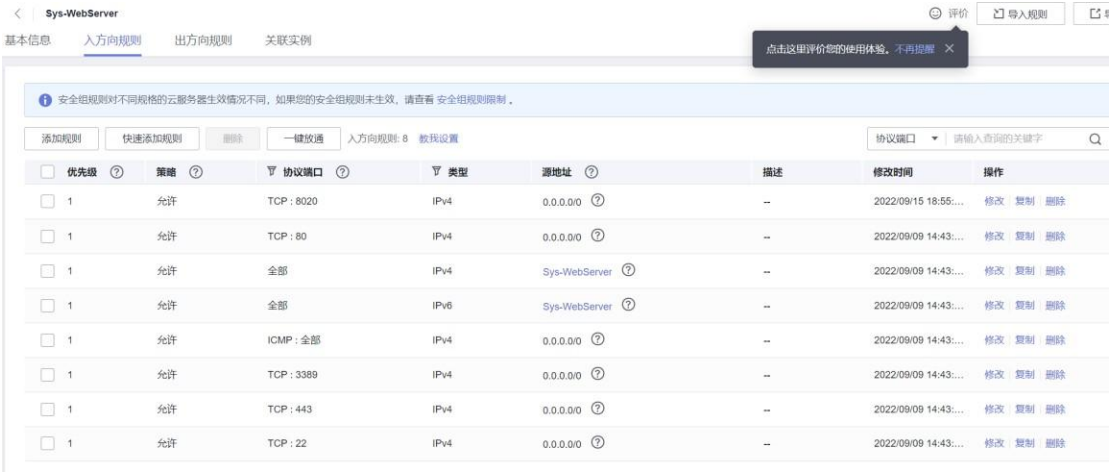


点击入方向规则，点击添加规则

按照下图填入后，点击确定

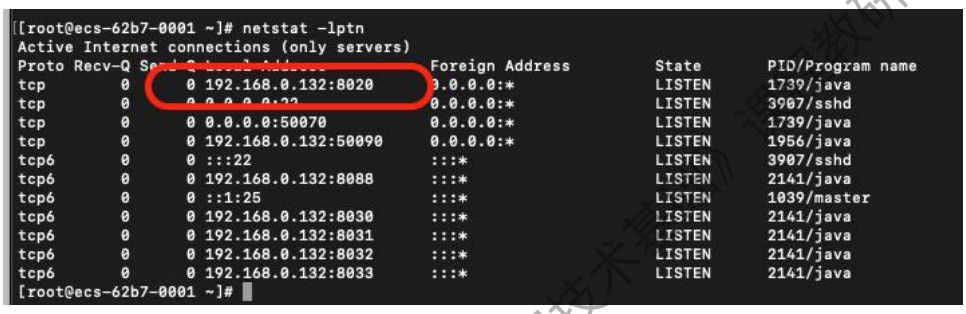


修改后的截图如下，则表示开放成功



首先确定 Hadoop 集群 8020 端口是否开放，连接服务器后输入：

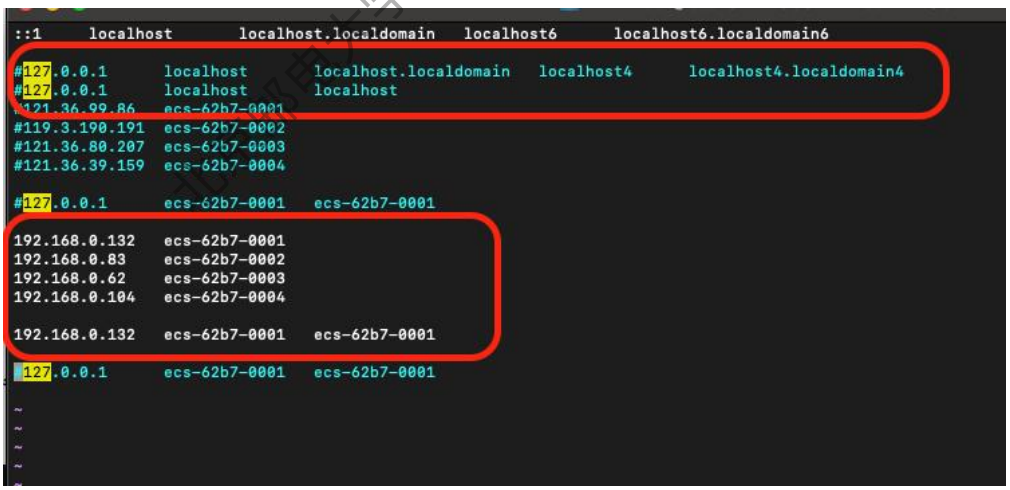
netstat -ltpn



确保 8020 端口监听的不是本地 IP（上图为正确情况，可跳过 hosts 文件修改步骤；若为 127.0.0.1:8020 则需要修改 hosts 文件）

修改 hosts 文件（四台服务器都需要操作），输入：

vim /etc/hosts



将 127.0.0.1 的部分注释掉，然后将四台服务器 ip 修改为局域网 ip（可在华为云上查看）然后设置电脑与服务器的 ssh 免密登陆：

打开终端，输入下面命令

ls ~/.ssh

如果存在 id_rsa 和 id_rsa.pub 文件，说明之前生成过密钥，无需操作；如果不存在上述两个文件，则命令行输入

ssh-keygen -t rsa

即可生成上述两个文件。

将公钥文件 id_rsa.pub 传送到服务器到~/.ssh 目录下

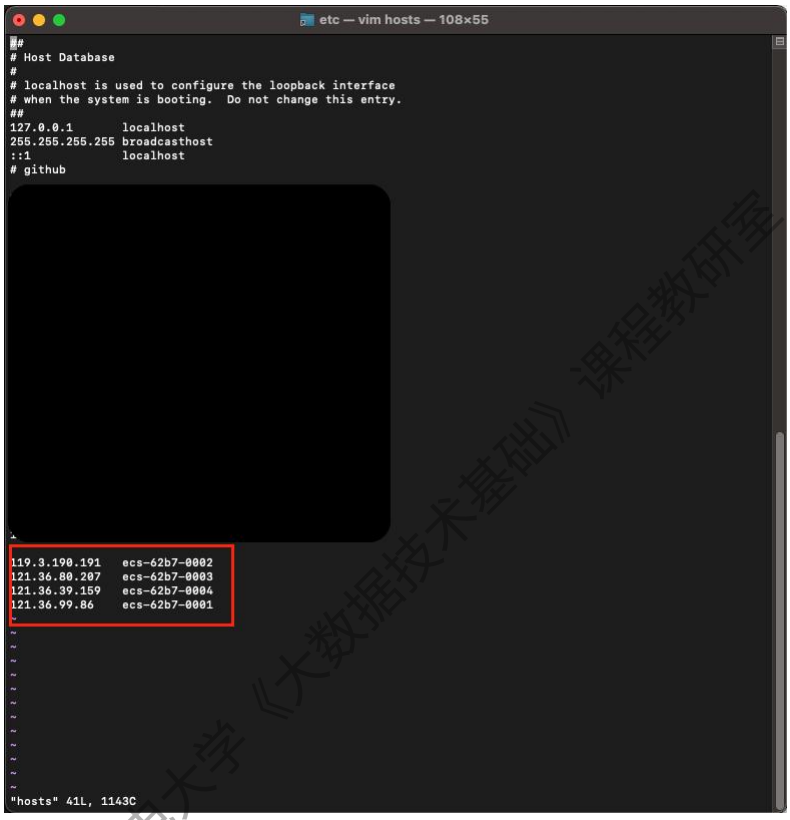
scp ~/.ssh/id_rsa.pub user-name@10.10.10.6:~/id_rsa.pub

服务器~/.ssh 目录已存在 authorized_keys，则将上传的 id_rsa.pub 添加到文件内容的后面确保本地电脑可以直接 ssh 连通服务器。

修改本地 hosts 文件，在本地终端输入：

vim /etc/hosts

添加四台服务器局域网 ip 以及服务器名称

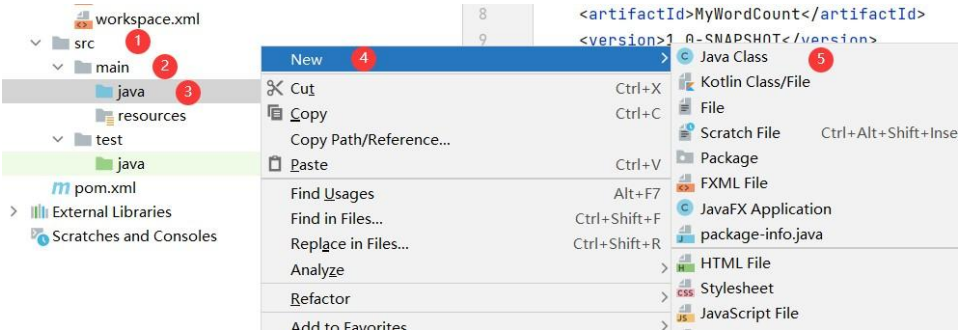


最后在主节点上启动 hadoop：

.start-dfs.sh start-yarn.sh

程序编写：

如下图依次打开 src—>main—>java，在 java 上点击右键，创建 Java Class；



2)弹出如下对话框，输入类名 ExeHDFS，点击ok

代码实现：

```
//查看HDFS文件
public void testView() throws IOException, URISyntaxException, InterruptedException {
    System.out.println("View file:");
    Configuration conf = new Configuration();
    conf.set("dfs.client.use.datanode.hostname", "true");
    conf.set("fs.defaultFS", "hdfs://121.36.99.86:8020");
    FileSystem hdfs = FileSystem.get(new URI( str: "hdfs://121.36.99.86"),conf, user: "root");
    Path path = new Path(hdfsPath);
    FileStatus[] list = hdfs.listStatus(path);
    if(list.length==0){
        System.out.println("HDFS is empty.");
    }else {
        for (FileStatus f : list) {
            System.out.printf("name: %s, folder: %s, size: %d\n", f.getPath(), f.isDirectory(), f.getLen());
        }
    }
}
```

查看 HDFS 文件系统：

上传本地文件到 HDFS：

```
//上传本地文件到HDFS
public void testUpload() throws IOException, URISyntaxException, InterruptedException {
    System.out.println("Upload file:");
    Configuration conf = new Configuration();
    conf.set("dfs.client.use.datanode.hostname", "true");
    conf.set("fs.defaultFS", "hdfs://121.36.99.86:8020");
    FileSystem hdfs = FileSystem.get(new URI( str: "hdfs://121.36.99.86"),conf, user: "root");
    InputStream in = new FileInputStream( name: "/Users/lxd/Desktop/upload.txt");
    OutputStream out = hdfs.create(new Path( pathString: hdfsPath+"upload_2021123456.txt"));
    IOUtils.copyBytes(in, out, conf);
    System.out.println("Upload successfully!");
}
```

```java
// 创建HDFS文件
public void testCreate() throws Exception {
    System.out.println("Write file:");
    Configuration conf = new Configuration();
    conf.set("dfs.client.use.datanode.hostname", "true");
    conf.set("fs.defaultFS", "hdfs://121.36.99.86:8020");
    //待写入文件内容
    //写入自己的姓名与学号信息
    byte[] buff = "Hello world! My name is lxd, my student id is 2021123456.".getBytes();
    //FileSystem 为 HDFS的API，通过此调用HDFS
    FileSystem hdfs = FileSystem.get(new URI( str: "hdfs://121.36.99.86"),conf, user: "root");
    //文件目标路径，应填写hdfs文件路径
    Path dst = new Path( pathString: hdfsPath + "lxd_2021123456.txt");
    FSDataOutputStream outputStream = null;
    try {
        //写入文件
        outputStream = hdfs.create(dst);
        outputStream.write(buff,  off: 0, buff.length);
    } catch (Exception e) {
        e.printStackTrace();

    } finally {
        if (outputStream != null) {
            outputStream.close();
        }
    }
    //检查文件写入情况
    FileStatus files[] = hdfs.listStatus(dst);
    for (FileStatus file : files) {
        //打印写入文件路径及名称
        System.out.println(file.getPath());
    }
}
```

HDFS 写入文件：

```java
//从HDFS下载文件到本地
public void testDownload() throws URISyntaxException, IOException, InterruptedException {
    System.out.println("Download file:");
    Configuration conf = new Configuration();
    conf.set("dfs.client.use.datanode.hostname", "true");
    conf.set("fs.defaultFS", "hdfs://121.36.99.86:8020");
    FileSystem hdfs = FileSystem.get(new URI( str: "hdfs://121.36.99.86"),conf, user: "root");
    InputStream in = hdfs.open(new Path( pathString: hdfsPath + "lxd_2021123456.txt"));
    OutputStream out = new FileOutputStream( name: "/Users/lxd/Desktop/download_2021123456.txt");
    IOUtils.copyBytes(in, out, conf);
    System.out.println("Download successfully!");
}
```

下载 HDFS 文件到本地：

参考代码

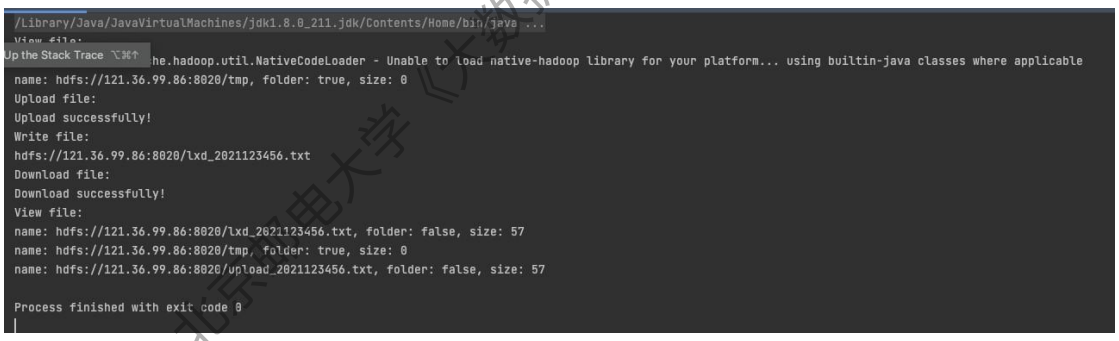
```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
```

```
import java.net.URI;
import java.net.URISyntaxException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IOUtils;

public class ExeHDFS { String hdfsPath = "/";

public static void main(String[] args) {
ExeHDFS testHDFS = new ExeHDFS();
try {
testHDFS.testView();
testHDFS.testUpload();
testHDFS.testCreate();
testHDFS.testDownload();
testHDFS.testView();
}
catch (Exception e) {
  e.printStackTrace();
}
}

// 查看 HDFS 文件系统
public void testView() throws IOException, URISyntaxException, InterruptedException
{ System.out.println("View file:");
Configuration conf = new Configuration(); conf.set("dfs.client.use.datanode.hostname", "true");
conf.set("fs.defaultFS", "hdfs://122.9.40.226:8020");
// TODO: 将 "node1ip" 修改为自己主节点的公网 ip 地址
FileSystem hdfs = FileSystem.get(new URI("hdfs://122.9.40.226"), conf, "root");
// TODO: 将 "node1ip" 修改为自己主节点的公网 ip 地址
Path path = new Path(hdfsPath); FileStatus[] list = hdfs.listStatus(path); if (list.length == 0) {
System.out.println("HDFS is empty.");
}
else {
for (FileStatus f : list) {
System.out.printf("name: %s,  folder:    %s,  size: %d\n",    f.getPath(), f.isDirectory(),
f.getLen());
}
}
}
```

```
// 上传本地文件到 HDFS
public void testUpload() throws IOException, URISyntaxException, InterruptedException
{ System.out.println("Upload file:");
Configuration conf = new Configuration();
conf.set("dfs.client.use.datanode.hostname", "true");
conf.set("fs.defaultFS", "hdfs://122.9.40.226:8020");
// TODO: 将 "node1ip" 修改为自己主节点的公网 ip 地址
FileSystem hdfs = FileSystem.get(new URI("hdfs://122.9.40.226"), conf, "root");
// TODO: 将 "node1ip" 修改为自己主节点的公网 ip 地址
InputStream in = new FileInputStream("./upload.txt");
// TODO: fix, 完善要上传的文件(upload.txt)的路径
OutputStream out = hdfs.create(new Path(hdfsPath + "upload_studentID.txt"));
// TODO:将 "studentID" 修改为自己的学号
IOUtils.copyBytes(in, out, conf); System.out.println("Upload successfully!");
}

// 创建 HDFS 文件
public void testCreate() throws Exception { System.out.println("Write file:"); Configuration conf
= new Configuration();
conf.set("dfs.client.use.datanode.hostname", "true");
conf.set("fs.defaultFS", "hdfs://122.9.40.226:8020");
 // TODO：将 "node1ip" 修改为自己主节点的公网 ip 地址
// 待写入文件内容
// 写入自己姓名与学号
byte[] buff = "Hello world! Myname is name, my student id is studentID.".getBytes();
// TODO: 完善姓名与学号
// FileSystem 为 HDFS 的 API，通过此调用 HDFS
FileSystem hdfs = FileSystem.get(new URI("hdfs://122.9.40.226"), conf, "root");
// TODO：将 "node1ip" 修改为自己主节点的公网 ip 地址
// 文件目标路径，应填写 hdfs 文件路径
Path dst = new Path(hdfsPath + "gby_studentID.txt");
// TODO：将 "studentID" 修改为自己的学号
FSDataOutputStream outputStream = null; try {
// 写入文件
outputStream = hdfs.create(dst); outputStream.write(buff, 0, buff.length);
} catch (Exception e) { e.printStackTrace();
} finally {
if (outputStream != null) { outputStream.close();
}
}
// 检查文件写入情况
FileStatus files[] = hdfs.listStatus(dst); for (FileStatus file : files) {
// 打印写入文件路径及名称
System.out.println(file.getPath());
```

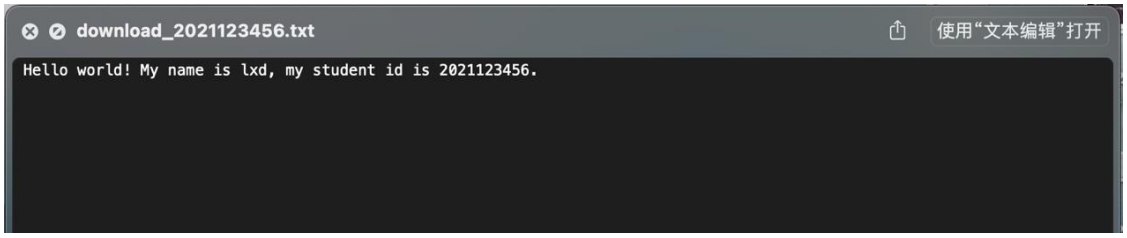
```
    }
    }

    // 从 HDFS 下载文件到本地
    public void testDownload() throws IOException, URISyntaxException, InterruptedException
    {
    System.out.println("Download file:"); Configuration conf = new Configuration();
conf.set("dfs.client.use.datanode.hostname", "true");
    conf.set("fs.defaultFS", "hdfs://122.9.40.226:8020");
    // TODO: 将 "node1ip" 修改为自己主节点的公网 ip 地址
    FileSystem hdfs = FileSystem.get(new URI("hdfs://122.9.40.226"), conf, "root");
    // TODO:  将 "node1ip" 修改为自己主节点的公网 ip 地址
    InputStream in = hdfs.open(new Path(hdfsPath + "gby_studentID.txt"));
    // TODO:  将"studentID"修改为自己的学号
    OutputStream out = new FileOutputStream("download_studentID.txt");
    // TODO:  fix，完善下载的文件(download_studentID.txt)的存放路径，就是放哪儿
    IOUtils.copyBytes(in, out, conf); System.out.println("Download successfully!");
    }
    }
```

最终输出格式：



注意：要包含自己的学号信息：

若 IDEA 的输出结果于上图不一致，可翻看控制台查看具体错误，上网查阅解决错误的方法



（从 HDFS 下载的文件）

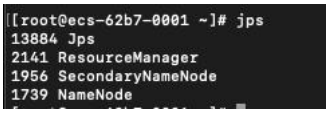
1.1.1.4. 实验结果与评分标准

实验结束后应得到：一个 Hadoop 集群，其中 1 个主节点，3 个子节点。一个 maven 工程。

完成 HDFS 文件读写实践。

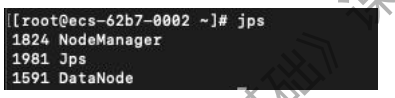
实验输出和应该完成的重点步骤应包含：

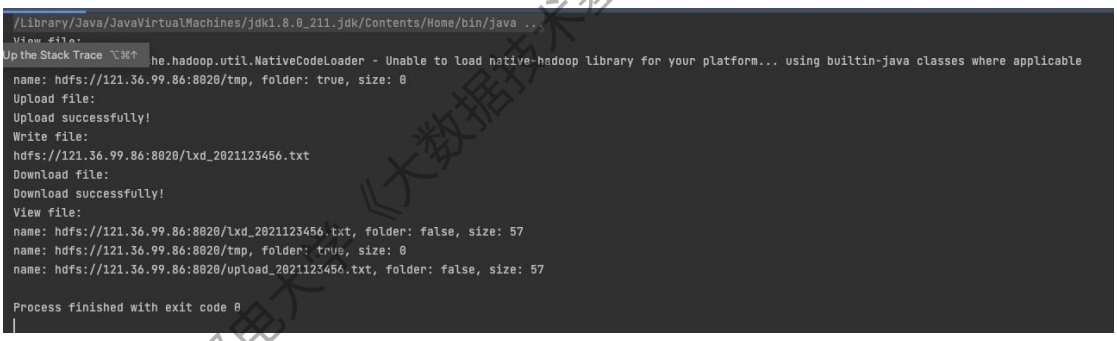
1) maven 打压缩包
2) 实验报告：



图一：启动 Hadoop 后，主节点输入 jps 后的输出，截图中显示学号，表示独立完成实验。

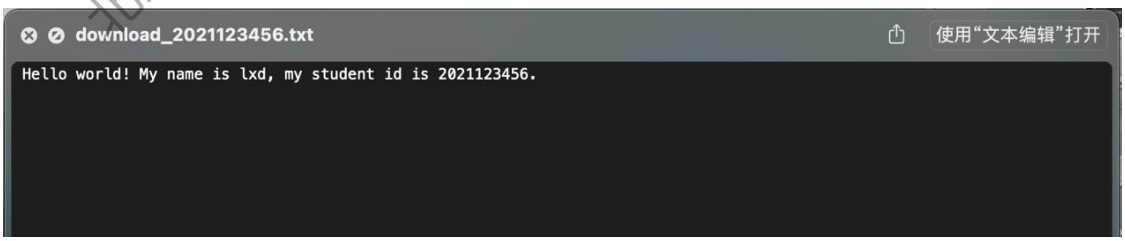
图二：启动 Hadoop 后，任意子节点输入 jps 后的输出，截图中显示学号，表示独立完成实验。





图三：java 代码运行结果（按要求包含学号信息，表示独立完成实验。）

图四：HDFS 下载文件截图（按要求包含学号信息，表示独立完成实验。）



实验报告应包含对截图的文字介绍，以证明理解截图含义。