

实验一：环境搭建

Design by W.H Huang | Direct by Prof Feng

1 实验目的

本次实验预估耗时较长，因此将给出所有详细步骤，如若不能及时完成可在课后完成。

通过本次实验，你应该完成以下部分：

- 组内合作完成 Hadoop & Spark 单机版环境搭建
- 组内合作完成 Hadoop & Spark 分布式环境搭建

最终需搭建相关详细环境如下：

- 操作系统：centOS 7.6.64
- 图形界面：GNOME
- 语言环境：python 3.6.8
- 相关软件：Hadoop 2.8.5、Spark 2.4.4

版本通常不严格要求一致，保持大版本一致一般即可，如python可使用 3.x 相关版本。

2 实验准备

2.0 计分说明

本次实验将详细介绍三种方式来搭建 Hadoop & Spark 分布式环境。在你正式开始选择实验前，**请认真阅读每个选择的利弊，酌情选择适合自己的方式。**

	最高分	优点	缺点	适合人群	备注
云服务器分布式	100	1.给分会更高一点	1.会出现更多的端口、网络、甚至病毒攻击问题	基础相对较好、动手能力较强、脾气好有耐心的同学	后续基于云服务器分布式的实验，按最高100计分
VM虚拟机分布式	90	1.实验更简单、出现各种奇怪的问题更少	1.给分相对较低一点	基础相对次好，和我一样经常暴躁debug导致rm -rf服务器的同学	后续基于虚拟机分布式的实验，按最高95计分
伪分布式	-	-	运行时易出现资源不足，该实验本教程已过时	-	-

特别的，考虑到大家 IP 是动态分配（DHCP），没有使用固定IP。使用第三种方式 **多台实际机器搭建（如，三个同学使用双系统）不方便。**

因此这里推荐大家使用前两种方式：云服务器分布式、VM虚拟机分布式进行环境搭建。

2.1 系统安装

无论是选择云服务器搭建还是VM虚拟机搭建的同学，

- 建议按照 `ex0--实验准备` 配置相应环境！
- 建议按照 `ex0--实验准备` 配置相应环境！
- 建议按照 `ex0--实验准备` 配置相应环境！

3 （2选1）云服务器分布式搭建

出于最简化演示目的，本次搭建将采用两台云服务器进行Hadoop+Spark 详细搭建记录。

☺ 如果小组成员>2，分布式搭建过程大同小异聪明如你应该知道怎么做。

首先记录下小组成员各自服务器的 内网IP&公网IP，例如我的：

主机名	内网IP	外网IP
master	172.30.0.7	129.28.154.240
slave01	172.16.0.4	134.175.210.3

3.1 Spark单机版搭建

⚠ 请注意，3.1.1 部分需在小组成员在各自云服务器上完成。3.1.2~3.1.4 小节只需在一台云服务器完成即可（作为master节点那台服务器）。

在进行Hadoop、Spark环境搭建前，我们需要进行一些准备工作。

3.1.1 准备工作

3.1.1 部分需在小组成员在各自云服务器上完成。

1 配置用户

该小节主要是创建 Hadoop 用户。

1. 创建用户

```
1 useradd -m hadoop -s /bin/bash
```

同时设置用户密码：（如 123456）

```
1 passwd hadoop
```

2. 配置权限

为了方便，给用户 hadoop 等同 root 权限：

```
1 visudo # 执行 visudo命令进入vim编辑
```

找到如下位置，添加红框那一行配置权限：

```
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)    ALL
hadoop  ALL=(ALL)    ALL
```

3. 切换用户

配置完成后，我们切换到hadoop用户下：

```
1 su hadoop # 注意，不要使用root用户，以下全部切换到hadoop用户下操作
```

⚠ 如非特殊说明，接下来所有命令都是Hadoop用户（不用使用root用户）下完成！⚠

2 配置SSH

为什么要配置ssh？

因为集群、单节点模式都需要用到 ssh登陆。同时每次登陆ssh都要输入密码是件蛮麻烦的事，我们可以通过生成公钥配置来面密码登陆。

1. 生成密钥

为了生成 ~/.ssh 目录，我们直接通过执行下面命令会直接生成

```
1 ssh localhost # 按提示输入yes，然后键入hadoop密码
```

然后开始生成密钥

```
1 | cd ~/.ssh/          # 切换目录到ssh下
2 | ssh-keygen -t rsa    # 生成密钥
```

生成密钥过程会有三个提示，不用管全部回车。

2. 授权

```
1 | cat id_rsa.pub >> authorized_keys # 加入授权
```

3. 修改权限

如果不修改文件 `authorized_keys` 权限为 `600`，会出现访问拒绝情况

```
1 | chmod 600 ./authorized_keys # 修改文件权限
```

4. 测试

```
1 | ssh localhost # ssh登陆
```

不用输入密码，直接登陆成功则说明配置正确。

```
[hadoop@VM_0_7_centos .ssh]$ ssh localhost
Last login: Thu Jan 23 17:05:19 2020 from 127.0.0.1
```

5. 新的风暴

- "倒霉的某个晚上，无论是MobaXterm，XShell 7，还是putty都无法进入服务器，差点破防了"

@[issue#30](#)，参考解决方案。

3 配置yum源

官方网站下载实在太慢，我们可以先配置一下阿里源来进行下载。

1. 切换到 yum 仓库

```
1 | cd /etc/yum.repos.d/
```

2. 备份下原repo文件

```
1 | sudo mv CentOS-Base.repo CentOS-Base.repo.backup
```

3. 下载阿里云repo文件

```
1 | sudo wget -O /etc/yum.repos.d/CentOS-7.repo
http://mirrors.aliyun.com/repo/Centos-7.repo
```

防止权限不足使用 `sudo` 命令。

4. 设置为默认repo文件

就是把阿里云repo文件名修改为 `CentOS-Base.repo`

```
1 | sudo mv CentOS-7.repo CentOS-Base.repo # 输入y
```

5. 生成缓存

```
1 yum clean all
2 yum makecache
```

4 配置Java环境

最开始下载的是 1.7 版本的JDK，后面出现的问题，重新下载 1.8 版本JDK。

hadoop2 基于 java 运行环境，所以我们要配置java 运行环境。

1. 安装JDK

执行下面命令，经过实际测试前面几分钟一直显示镜像错误不可用。它会进行自己尝试别的源，等待一会儿就可以下载成功了。

```
1 sudo yum install java-1.8.0-openjdk java-1.8.0-openjdk-devel
```

⚠ 此时默认安装位置是 /usr/lib/jvm/java-1.8.0-openjdk

其实，查找安装路径，可以通过以下命令：

```
1 rpm -ql java-1.8.0-openjdk-devel | grep '/bin/javac'
```

- rpm -ql <RPM包名> : 查询指定RPM包包含的文件
- grep <字符串> : 搜索包含指定字符的文件

2. 配置环境变量

```
1 vim ~/.bashrc # vim编辑配置文件
```

在文件最后面添加如下单独一行（指向JDK 的安装位置），并保存：

```
1 export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
```

```
# add chrome path
export PATH=$PATH:/opt/google/chrome
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk
```

最后是环境变量生效，执行：

```
1 source ~/.bashrc
```

3. 测试

```
1 echo $JAVA_HOME # 检验变量值
```

正常会输出 2. 环境变量JDK配置路径。

```
1 java -version
```

正确配置会输出java版本号。

5 安装python

CentOS自带python2版本过低，我们进行python3安装。

1. yum查找python3

查找仓库存在的python3安装包

```
1 yum list python3
```

```
python3.x86_64 3.6.8-10.el7 @base
```

2. yum 安装python3

```
1 sudo yum install python3.x86_64
```

如果最开始会显示没有，等一会自动切换阿里源就可以进行安装了，同时还会安装相关依赖。

如果依旧报错，@[issue#35](#)，请安装epel使用阿里云镜像。

```
1 yum install -y epel-release
2 sudo yum install python3
```

3.1.2 hadoop 安装

3.1.2~3.1.4 小节只需在一台云服务器完成即可（作为master节点那台服务器）。

本文使用 `wget` 命令来下载 `hadoop`：[了解更多wget](#)。

使用的是[北理工镜像站](#)，下载 `hadoop`：

⚠ hadoop在本教程是使用 `2.x` 版本，如果你使用的是 `3.x` 版本，请注意@[issue#29](#)：

- 下面配置slaves文件时，`3.x` 版本里已经不叫slaves，**更名为workers!**
- 即目录更改为：`/usr/local/hadoop/etc/hadoop/worker`

1. 下载

为防止证书验证出现的下载错误，加上 `--no-check-certificate`，相关讨论可见 [issue#1](#)

```
1 # 这里下载2.8.5版本，可能已失效，请去北理工镜像站，查看可下载的版本链接
2 # 建议下载版本低于3.0版本
3 sudo wget -O hadoop-2.8.5.tar.gz
  https://mirrors.cnnic.cn/apache/hadoop/common/hadoop-2.8.5/hadoop-
  2.8.5.tar.gz --no-check-certificate
```

- `wget -O <指定下载文件名> <下载地址>`

2. 解压

```
1 sudo tar -zxf hadoop-2.8.5.tar.gz -C /usr/local
```

把下载好的文件 `hadoop-2.8.5.tar.gz` 解压到 `/usr/local` 目录下

3. 修改文件

```

1 | cd /usr/local/    # 切换到解压目录下
2 | sudo mv ./hadoop-2.8.5/ ./hadoop    # 将加压缩的文件hadoop-2.8.5重命名为hadoop
3 | sudo chown -R hadoop:hadoop ./hadoop # 修改文件权限

```

4. 测试

```

1 | cd /usr/local/hadoop    # 切换到hadoop目录下
2 | ./bin/hadoop version    # 输出hadoop版本号

```

```

[hadoop@VM_0_7_centos hadoop]$ ./bin/hadoop version
Hadoop 2.8.5

```

3.1.3 spark安装

在前我们已经安装了 *hadoop*，现在我们来开始进行*spark* 安装。

这次下载根据官网推荐使用的清华源。

1. 下载

官网下载地址：[官网下载](#)

Download Apache Spark™

1. Choose a Spark release:
2. Choose a package type:
3. Download Spark: [spark-2.4.4-bin-without-hadoop.tgz](#)
4. Verify this release using the 2.4.4 [signatures](#), [checksums](#) and [project release KEYS](#).

- 这样选择的版本可以使用于大部分 *hadoop* 版本

点击上述链接，根据跳转的页面提示选择清华源下载：

注意，版本号可能发生变化，建议打开上述官网链接查看当前存在的版本。如我查看到只支持 2.4.7 版本（2020/09/17），那么需修改下面版本号：2.4.4-->2.4.7

```

1 | sudo wget -O spark-2.4.7-bin-without-hadoop.tgz
   http://mirrors.tuna.tsinghua.edu.cn/apache/spark/spark-2.4.7/spark-2.4.7-
   bin-without-hadoop.tgz    # 版本号发生变化，记得替换，下同

```

2. 解压

同前解压到 */usr/local* 目录下

```

1 | sudo tar -zxvf spark-2.4.7-bin-without-hadoop.tgz -C /usr/local

```

3. 设置权限

```

1 | cd /usr/local    # 切换到解压目录
2 | sudo mv ./spark-2.4.7-bin-without-hadoop ./spark    # 重命名解压文件
3 | sudo chown -R hadoop:hadoop ./spark    # 设置用户hadoop为目录spark所有者

```

4. 配置spark环境

先切换到 */usr/local/spark*，（为了防止没权限，下面用 *sudo*）

```
1 | cd /usr/local/spark
2 | cp ./conf/spark-env.sh.template ./conf/spark-env.sh
```

编辑 spark-env.sh 文件：

```
1 | vim ./conf/spark-env.sh
```

在第一行添加下面配置信息，使得Spark可以从Hadoop读取数据。

```
1 | export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
```

5. 配置环境变量

```
1 | vim ~/.bashrc
```

在 .bashrc 文件中添加如下内容：

```
1 | export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk # 之前配置的java环境变量
2 | export HADOOP_HOME=/usr/local/hadoop # hadoop安装位置
3 | export SPARK_HOME=/usr/local/spark
4 | export PYTHONPATH=$SPARK_HOME/python:$SPARK_HOME/python/lib/py4j-0.10.7-
src.zip:$PYTHONPATH
5 | export PYSARK_PYTHON=python3 # 设置pyspark运行的python版本
6 | export PATH=$HADOOP_HOME/bin:$SPARK_HOME/bin:$PATH
```

最后为了使得环境变量生效，执行：

```
1 | source ~/.bashrc
```

6. 测试是否运行成功

```
1 | cd /usr/local/spark
2 | bin/run-example SparkPi
```

执行会输出很多信息，也可以选择执行：

```
1 | bin/run-example SparkPi 2>&1 | grep "Pi is"
```

```
[hadoop@VM_0_7_centos spark]$ bin/run-example SparkPi 2>&1 | grep "Pi is"
Pi is roughly 3.1462557312786563
```

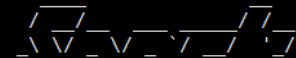
3.1.4 测试

1. 启动pyspark

```
1 | cd /usr/local/spark
2 | bin/pyspark
```



```
[hadoop@VM_0_7_centos spark]$ bin/pyspark
Python 3.6.8 (default, Aug  7 2019, 17:28:10)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] on linux
Type "help", "copyright", "credits" or "license" for more information.
20/01/23 17:33:19 WARN util.Utils: Your hostname, VM_0_7_centos resolves to a loopback add
ress: 127.0.0.1; using 172.30.0.7 instead (on interface eth0)
20/01/23 17:33:19 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another addre
ss
20/01/23 17:33:19 WARN util.NativeCodeLoader: Unable to load native-hadoop library for you
r platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel
).
Welcome to
```



```
version 2.4.4
```

2. 简单测试

```
1 >>> 8 * 2 + 5
```

使用 `exit()` 命令可退出。

3.2 Hadoop+Spark 分布式环境搭建

3.2.1 准备工作

1 修改主机名

两台服务器一台作为master，一台作为slave。为了以示区分，我们分别修改它们的主机名：

- 在master

```
1 | sudo vim /etc/hostname
```

编辑修改为: master

- 在 slave01

```
1 | sudo vim /etc/hostname
```

编辑修改为: slave01

最后使用命令 `sudo reboot` 重启，便会生效。

2 修改host

修改hosts目的：可以使用云服务器名字访问，而不直接使用IP地址

首先上自己的云服务器，记录下三台服务器的内网IP&公网IP

主机名	内网IP	外网IP
master	172.30.0.7	129.28.154.240
slave01	172.16.0.4	134.175.210.3

⚠ 警告，下面有个史前大坑。因为云服务器默认访问本身是用内网IP地址

- 在master上

```
1 | su hadoop
2 | sudo vim /etc/hosts
```

编辑hosts文件如下（以前的全部删除，改成下面这样）：

```
1 | 127.0.0.1 localhost
2 | 172.30.0.7 master      # master必须用内网IP
3 | 134.175.210.3 slave01  # slave01用外网IP
```

- 在 slave01 上

```
1 | su hadoop
2 | sudo vim /etc/hosts
```

```
1 | 127.0.0.1 localhost
2 | 129.28.154.240 master  # master必须用外网IP
3 | 172.16.0.4    slave01  # slave01用内网IP
```

3 SSH互相免密

在之前我们搭建Spark单机版环境时，我们配置ssh可以 *无密码* 本地连接：

```
1 | ssh localhost    # 保证两台服务器都可以本地无密码登陆
```

现在我们还要让 master主机免密码登陆slave01、slave02。因此我们要将master主机的 `id_rsa.pub` 分别传递给两台slave主机。

1. 在 master 上scp传递公钥

第一次传要输入slave01@hadoop用户密码，例如之前设置为123456

```
1 | scp ~/.ssh/id_rsa.pub hadoop@slave01:/home/hadoop/
```

2. 在slave01上加入验证

```
1 | ls /home/hadoop/    # 查看master传送过来的 id_rsa.pub文件
```

将master公钥加入免验证：

```
1 | cat /home/hadoop/id_rsa.pub >> ~/.ssh/authorized_keys
2 | rm /home/hadoop/id_rsa.pub
```

3. 测试

现在我们切换到master主机上，尝试能否免密登陆：

```
[hadoop@master ~]$ ssh slave01
Last failed login: Thu Jan 23 17:44:52 CST 2020 from 129.28.154.240 on ssh:notty
There was 1 failed login attempt since the last successful login.
Last login: Thu Jan 23 17:44:31 2020 from 127.0.0.1
```

验证可以免密登陆后切换回master主机

```
1 | ssh master    # 要输入master@hadoop用户密码
```

4. 新的风暴

- "SSH登录: WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!"

master免密登录slave结点, 但是ssh回master可能出现问题, 大致如下:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!    @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
```

解决方案:

- `vim ~/.ssh/known_hosts` 编辑配置文件
- 删除文件里与master的ip相关的那一项
- 重新登陆即可解决

3.2.2 Hadoop集群配置

原本我们需要同时在master和slave节点安装配置Hadoop集群, 但是我们也可以通过仅配置master节点Hadoop, 然后将整个配置好的Hadoop文件传递给各个子节点。

1 master节点配置

我们需要修改master主机上hadoop配置文件。

1. 切换目录

配置文件在 `/usr/local/hadoop/etc/hadoop` 目录下:

```
1 | cd /usr/local/hadoop/etc/hadoop
```

```
[hadoop@master hadoop]$ ll
total 160
-rw-r--r-- 1 hadoop hadoop 4942 Sep 10 2018 capacity-scheduler.xml
-rw-r--r-- 1 hadoop hadoop 1335 Sep 10 2018 configuration.xml
-rw-r--r-- 1 hadoop hadoop 318 Sep 10 2018 container-executor.cfg
-rw-r--r-- 1 hadoop hadoop 1085 Jan 23 17:51 core-site.xml
-rw-r--r-- 1 hadoop hadoop 3804 Sep 10 2018 hadoop-env.cmd
-rw-r--r-- 1 hadoop hadoop 4666 Sep 10 2018 hadoop-env.sh
```

2. 修改文件 `slaves`

△ 注意, 3.x 版本的hadoop已将slaves文件重命名workers!

master主机作为 `NameNode`, 而 slave01 作为 `DataNode`

```
1 | # `3.x` 版本的hadoop , vim workers
2 | vim slaves
```

修改如下:

```
1 | # `3.x` 版本的hadoop , 请删除localhost
2 | slave01
```

3. 修改文件 `core-site.xml`

```
1 | vim core-site.xml
```

```

1  <configuration>
2      <property>
3          <name>hadoop.tmp.dir</name>
4          <value>/usr/local/hadoop/tmp</value>
5          <description>Abase for other temporary directories.
6      </property>
7      <property>
8          <name>fs.defaultFS</name>
9          <value>hdfs://master:9000</value>
10     </property>
11 </configuration>

```

```

<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>file:/usr/local/hadoop/tmp</value>
    <description>Abase for other temporary directories.</description>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000</value>
  </property>
</configuration>

```

4. 修改 `hdfs-site.xml` :

```
1  vim hdfs-site.xml
```

```

1  <configuration>
2      <property>
3          <name>dfs.replication</name>
4          <value>3</value>
5      </property>
6      <property>
7          <name>mapred.job.tracker</name>
8          <value>master:9001</value>
9      </property>
10     <property>
11         <name>dfs.namenode.http-address</name>
12         <value>master:50070</value>
13     </property>
14 </configuration>

```

5. 修改 `mapred-site.xml.template`

⚠ 首先复制它产生一个新复制文件并命名为: `mapred-site.xml`

```
1  cp mapred-site.xml.template mapred-site.xml
```

然后修改文件 `vim mapred-site.xml` :

```

1  <configuration>
2      <property>
3          <name>mapreduce.framework.name</name>
4          <value>yarn</value>
5      </property>
6 </configuration>

```

6. 修改 yarn-site.xml

```
1 | vim yarn-site.xml
```

```
1 | <configuration>
2 |   <!-- Site specific YARN configuration properties -->
3 |   <property>
4 |     <name>yarn.nodemanager.aux-services</name>
5 |     <value>mapreduce_shuffle</value>
6 |   </property>
7 |   <property>
8 |     <name>yarn.resourcemanager.hostname</name>
9 |     <value>master</value>
10 |   </property>
11 | </configuration>
```

2 slave节点配置

⚠ 根据多次血泪经验：在slave节点上重复一遍master节点上的配置，而非通过传送文件。这种方式是不可行的，会导致意外之外的错误，即使你修改的一模一样！

🚩 方法1：通过scp将上述变动文件发送至slave（可以大幅度减少传送时间）

1. 传送已修改的配置文件

在master上节点上，使用如下命令将yarn-site.xml等发送到从机slave01上。

发送给其它从机，如slave02，同理。

```
1 | # on master
2 | scp /usr/local/hadoop/etc/hadoop/core-site.xml
   hadoop@slave01:/usr/local/hadoop/etc/hadoop/
3 | scp /usr/local/hadoop/etc/hadoop/hdfs-site.xml
   hadoop@slave01:/usr/local/hadoop/etc/hadoop/
4 | scp /usr/local/hadoop/etc/hadoop/mapred-site.xml
   hadoop@slave01:/usr/local/hadoop/etc/hadoop/
5 | scp /usr/local/hadoop/etc/hadoop/yarn-site.xml
   hadoop@slave01:/usr/local/hadoop/etc/hadoop/
```

2. 设置文件权限

```
1 | # on slave
2 | sudo chown -R hadoop /usr/local/hadoop
```

3. 检查文件变更

通过cat命令检查slave上的相关文件是否变更。

```
1 | # on slave
2 | cat /usr/local/hadoop/etc/hadoop/core-site.xml    # 确认文件是否传送正确
```

输出中含有上一步中修改后的信息，则确认正确。比如，core-site.xml 文件输出如下：

```

1 <!-- Put site-specific property overrides in this file. -->
2
3 <configuration>
4   <property>
5     <name>hadoop.tmp.dir</name>
6     <value>/usr/local/hadoop/tmp</value>
7     <description>Abase for other temporary directories.</description>
8   </property>
9   <property>
10    <name>fs.defaultFS</name>
11    <value>hdfs://master:9000</value>
12  </property>
13 </configuration>

```

🚩 方法2：压缩拷贝整个hadoop目录

- 在master节点上执行

```

1 cd /usr/local/
2 rm -rf ./hadoop/tmp      # 删除临时文件
3 rm -rf ./hadoop/logs/*  # 删除日志文件
4 # 压缩./hadoop文件，并重名为hadoop.master.tar.gz
5 tar -zcf ~/hadoop.master.tar.gz ./hadoop

```

将压缩好的文件传递给 slave01：

```

1 cd ~
2 scp ./hadoop.master.tar.gz slave01:/home/hadoop

```

😊 传递速度有点慢，大概要半小时。等待时间你可以先撰写部分实验报告，或者尝试浏览接下来实验步骤。

- 在slave01节点上
(如果有) 删除原有hadoop文件夹

```

1 sudo rm -rf /usr/local/hadoop/

```

解压传过来的文件到指定目录 `/usr/local`：

```

1 sudo tar -zxf /home/hadoop/hadoop.master.tar.gz -C /usr/local

```

设置解压出来的hadoop文件夹权限：

```

1 sudo chown -R hadoop /usr/local/hadoop

```

3 集群启动测试

1. master上启动集群

```

1 cd /usr/local/hadoop
2 bin/hdfs namenode -format # 注意，仅在第一次启动集群时使用该命令格式化！
3 sbin/start-all.sh

```

2. 测试

- 在master上

```
1 | jps
```

master节点出现以下4个进程则配置成功：

```
[hadoop@master hadoop]$ jps
13191 NameNode
13869 Jps
13598 ResourceManager
13438 SecondaryNameNode
```

- 在 slave01上

```
1 | jps
```

slave节点出现以下3个进程则配置成功：

```
[hadoop@slave01 ~]$ jps
5384 DataNode
5689 Jps
5549 NodeManager
```

4 新的风暴：问题解决

Q1: slave 节点没有 DataNode 进程 / master 节点没有 namenode 进程？

这个问题一般是由于在启动集群多次执行格式化命令：

```
1 | bin/hdfs namenode -format
```

导致 hodoop 目录下 tmp/dfs/name/current 文件下的 VERSION 中的 namespaceId 不一致。

首先我们 在master节点上 停止集群：

```
1 | cd /usr/local/hadoop # 切换到你的hadoop目录下
2 | sbin/stop-all.sh      # 关闭集群
```

- slave 节点删除 tmp

删除slave节点的临时 tmp 文件

```
1 | cd /usr/local          # 切换到hadoop目录
2 | rm -rf ./hadoop/tmp
```

删除 tmp 文件，如法炮制在 其它节点 进行一样的操作：

```
1 | rm -rf ./hadoop/tmp # 后面格式化会重新生成，大胆删除
```

- 在master 节点删除 tmp

```
1 | cd /usr/local
2 | rm -rf ./hadoop/tmp
```

- 重新启动集群

在master节点执行以下操作：

```
1 | cd /usr/local/hadoop
2 | bin/hdfs namenode -format # 重新格式化
3 | sbin/start-all.sh
```

- 验证

在 master 节点执行以下操作：

```
1 | cd ~
2 | jps
```

```
[root@master hadoop]# jps
30496 NameNode
30882 ResourceManager
30717 SecondaryNameNode
621 Master
32127 Jps
```

在子节点再次输入 jps 命令：

```
1 | cd ~
2 | jps
```

```
[root@slave02 ~]# jps
5794 Jps
5613 NodeManager
5503 DataNode
```

ok~

Q2：启动集群后发现，slave 节点没有 NodeManager 进程

```
[hadoop@slave01 local]$ jps
30673 DataNode
31172 Jps
```

少了nodemanager进程

△ 建议先尝试 Q1 方法，一般能解决大部分问题。

启动集群时可以知道，启动 slave01 节点 notemanager 进程相关日志在（最后不是 .out 是 .log）：

/usr/local/hadoop/logs/yarn-hadoop-nodemanager-slave01.log

```
[hadoop@master hadoop]$ sbin/start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [master]
master: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-namenode-master.out
slave01: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hadoop-datanode-slave01.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-secondary-namenode-master.out
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-resourcemanager-master.out
slave01: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-nodemanager-slave01.out
```

1. 查看日志

在 slave01 节点下

```
1 | vim /usr/local/hadoop/logs/yarn-hadoop-nodemanager-slave01.log
```

日志太多，我们在 命令模式 下，输入 :\$ ，直接跳到最后一行：


```

2020-01-31 18:12:33,973 INFO org.apache.hadoop.service.AbstractService: Service org.apache.hadoop.yarn.server.nodemanager.containermanager.localizer.ResourceLocalizationService failed in state STARTED
; cause: org.apache.hadoop.yarn.exceptions.YarnRuntimeException: java.net.BindException: Problem binding to [0.0.0.0:8040] java.net.BindException: Address already in use; For more details see: http://wiki.apache.org/hadoop/BindException
org.apache.hadoop.yarn.exceptions.YarnRuntimeException: java.net.BindException: Problem binding to [0.0.0.0:8040] java.net.BindException: Address already in use; For more details see: http://wiki.apache.org/hadoop/BindException
    at org.apache.hadoop.yarn.factories.impl.pb.RpcServerFactoryPBImpl.getServer(RpcServerFactoryPBImpl.java:138)

```

- 很显然，显示端口 8040 被占用

2. 查看谁占用 8040 端口

```
1 | netstat -tln | grep 8040
```

```

[hadoop@slave01 local]$ netstat -tln | grep 8040
tcp6      0      0 :::8040          :::*              LISTEN

```

果然 8040 端口已经被占用

3. 释放端口

```
1 | sudo lsof -i :8040 # 查询占用8040端口进程pid
```

```

[hadoop@slave01 local]$ lsof -i :8040
COMMAND  PID  USER  FD  TYPE  DEVICE  SIZE/OFF  NODE NAME
java    16961 hadoop 231u IPv6 38843230      0t0  TCP *:ampify (LISTEN)

```

杀死相应进程：

```
1 | sudo kill -9 16961
```

4. 测试

重新启动集群

```

1 | cd /usr/local/hadoop
2 | sbin/stop-all.sh
3 | sbin/start-all.sh

```

再次输入 jps 命令，发现 slave01 节点 NodeManager 进程已经出现！

```

[hadoop@slave01 local]$ jps
30673 DataNode
6419 Jps
6035 NodeManager

```

3.2.3 Spark集群配置

以下步骤都建立在是我们三台云服务器已经搭建好Spark单机版环境 & hadoop集群。

1 Spark配置

1. 切换配置目录

```
1 | cd /usr/local/spark/conf
```

2. 配置 slaves 文件

```
1 | cp slaves.template slaves # 先把模板文件复制重命名
```

开始编辑 vim slaves，将默认内容 localhost 替换为以下：

```
1 | slave01
```

3. 配置 spark-env.sh 文件

```
1 | cp spark-env.sh.template spark-env.sh
```

开始编辑，添加下面内容：

```
1 | vim spark-env.sh
```

```
1 | export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
2 | export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
3 | export SPARK_MASTER_IP=172.30.0.7 # 注意，使用的master内网IP!!
```

4. 复制Spark文件到各个slave节点

```
1 | cd /usr/local/
2 | tar -zcf ~/spark.master.tar.gz ./spark
3 | cd ~
4 | scp ./spark.master.tar.gz slave01:/home/hadoop
```

5. 节点替换文件

以下操作是在 slave节点上：

```
1 | sudo rm -rf /usr/local/spark/ # 删除节点原有Spark文件（如果有）
2 | sudo tar -zxvf /home/hadoop/spark.master.tar.gz -C /usr/local # 解压到
  local
3 | sudo chown -R hadoop /usr/local/spark # 设置spark文件权限拥有者是hadoop
```

2 启动Spark集群

在master主机上执行以下操作

1. 先启动hadoop集群

```
1 | cd /usr/local/hadoop/
2 | sbin/start-all.sh
```

2. 启动master节点

```
1 | cd /usr/local/spark/
2 | sbin/start-master.sh
```

master上运行 jps 命令可以看到：

```
[hadoop@master spark]$ jps
18160 Master
18224 Jps
13191 NameNode
13598 ResourceManager
13438 SecondaryNameNode
```

3. 启动所有slave节点

```
1 | sbin/start-slaves.sh
```

slave节点上运行 `jps` 命令可以看到：

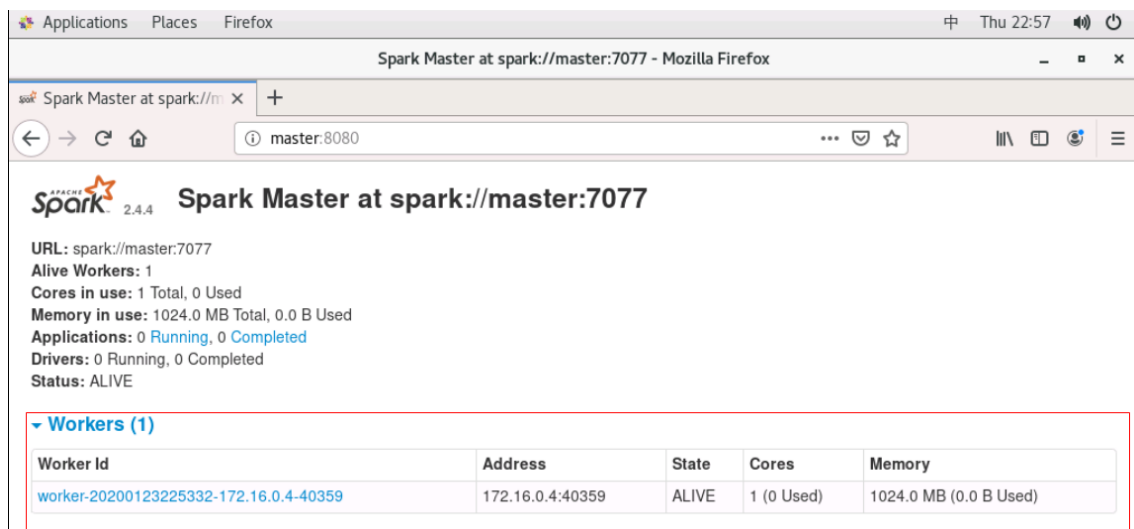
```
[hadoop@slave01 ~]$ jps
9527 Jps
5384 DataNode
5549 NodeManager
9471 Worker
```

4. web UI查看

现在你有两种方式查看：

1. 打开腾讯云控制台，选择 VNC 登陆服务器，在浏览器上输入：`master:8080`；
2. 在本地机器上，输入 `master公网IP:8080` 查看（可能需要内网穿透，不一定可行）。

如果出现下面界面则表示 *Hadoop+Spark* 分布式环境搭建成功！



3 问题解决

⚠ 如果前面一切正常，Web UI 却无法正常工作显示worker。

[ERROR#1] 查看slave节点相关 `spark` 日志发现报错：无法访问 `<master外网ip>:7070`，多次连接失败。

一般出现这个问题，那么则可能是：**ip、端口、防火墙**等问题。

【1. 端口问题】

相关的一些讨论也可参考：[issue#3 @trevery](#)

- 使用nmap工具测试

在slave节点测试master:7077端口是否被放通（master测试slave同理）：

```
1 | nmap -p 7077 master_ip
```

如果7077端口没有被放通：

- 本机防火墙放通指定端口

```
1 | sudo firewall-cmd --zone=public --add-port=7077/tcp --permanent
2 | sudo firewall-cmd --reload
```

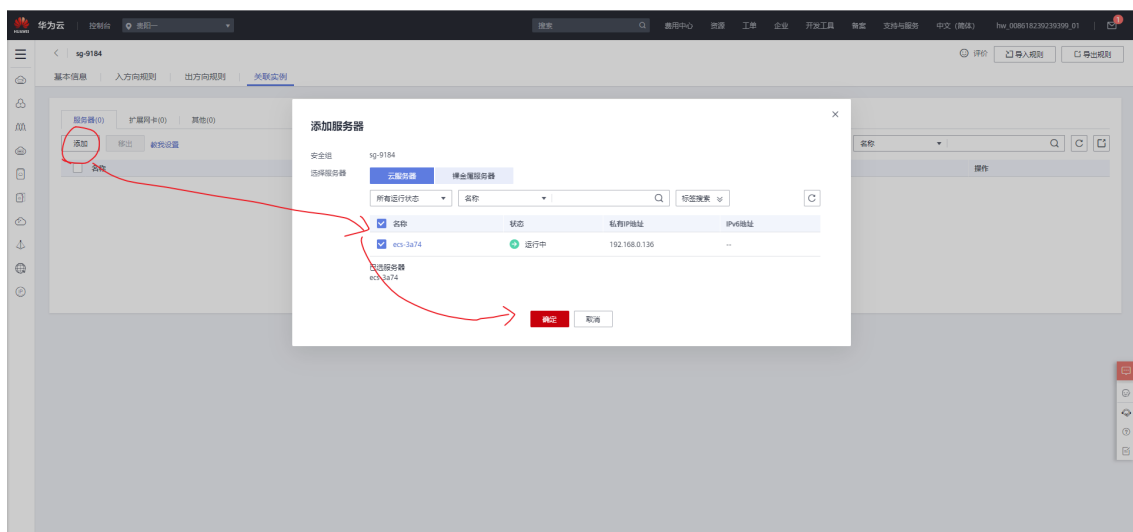
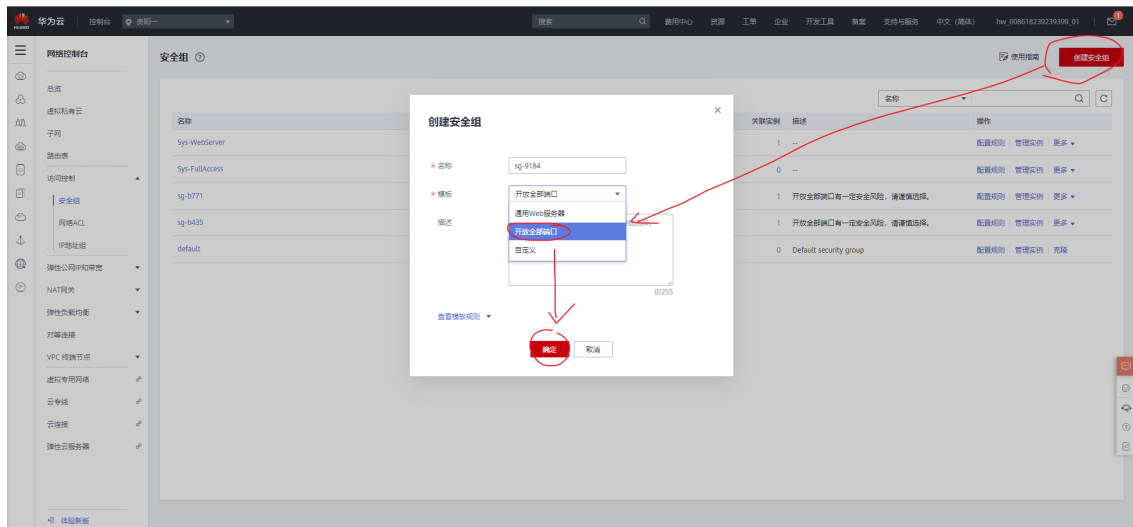
- 我们去云控制台上将端口放通，下详

- 华为云（其它云平台操作类似）放通端口

⚠ 这里为快速验证端口问题，选择**全部放通**，这是不安全的操作，易遭受攻击。建议确认是端口问题后，只放通7077端口。

- 其它: [腾讯云放通端口](#)

登陆控制台 (master) --> 创建安全组 (选择**放通所有端口**) --> 将**master**加入刚创建的安全组



- 重新启动集群

```
1 sbin/start-master.sh # 先启动master
2 sbin/start-slave.sh spark://master内网ip:7077 # 指定master内网ip启动slaves节点
```

【2.ip问题】

确定master的7077端口已被放通后, 那么极有可能是ip问题导致。

注意到日志中slave是以master的**内网IP**: 192.168.1.219, 进行连接的。这是因为, 我们在 `spark-env.sh` 文件设置了 `SPARK_MASTER_IP` 为内网ip。某些情况可能会导致错误, **因为slave无法通过内网ip去访问master服务器。**

所以, 我们进行如下修改:

```
1 vim spark-env.sh
```

修改 `SPARK_MASTER_IP=master`, slave中hosts文件记录了主机名master对应的**公网ip**, 这样slave就可以通过master公网ip去访问master, 从而避免错误。

```
1 export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
2 export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
3 export SPARK_MASTER_IP=master
```

【3.其它问题】

当然，这也可能是防火墙等问题导致，关闭防火墙可以解决。但一般是ip、端口设置错误导致。

```
1 //Disable firewall
2 systemctl disable firewalld
3 systemctl stop firewalld
4 systemctl status firewalld
5
6 //Enable firewall
7 systemctl enable firewalld
8 systemctl start firewalld
9 systemctl status firewalld
```

🌸🌸 聪明如你终于做到这步了，第一个实验完结，撒花 🌸🌸

4 (2选1) VM虚拟机分布式搭建

不同于云服务器搭建，VM虚拟机只需准备一台机器即可，我们搭建分布式大概流程如下：

1. 创建一台虚拟机
2. 在该虚拟机下进行单机版配置
3. 配置完成后，将该虚拟机文件复制N份，形成分布式环境

4.1 单机版搭建

在进行Hadoop、Spark环境搭建前，我们需要进行一些准备工作。

4.1.1 准备工作

1 配置用户

该小节主要是创建 `hadoop` 用户。

1. 创建用户

```
1 useradd -m hadoop -s /bin/bash
```

同时设置用户密码：（如 123456）

```
1 passwd hadoop
```

2. 配置权限

为了方便，给用户 `hadoop` 等同 `root` 权限：

```
1 visudo # 执行 visudo命令进入vim编辑
```

找到如下位置，添加红框那一行配置权限：

```
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)    ALL
hadoop  ALL=(ALL)    ALL
```

3. 切换用户

配置完成后，我们切换到hadoop用户下：

```
1 | su hadoop # 注意，不要使用root用户，以下全部切换到hadoop用户下操作
```

⚠ 如非特殊说明，接下来所有命令都是Hadoop用户（不用使用root用户）下完成！⚠

2 配置SSH

为什么要配置ssh？

因为集群、单节点模式都需要用到ssh登陆。同时每次登陆ssh都要输入密码是件蛮麻烦的事，我们可以通过生成公钥配置来面密码登陆。

1. 安装openssh及配置

CentOS7默认安装openssh, 6以及之前的版本需要自行安装。

```
1 | sudo yum install open-ssh
```

生成的密钥有后缀“用户@主机”，所以建议先设置主机名。

2. 生成密钥

再次强调，默认都是在hadoop用户下执行以下操作。

为了生成 ~/.ssh 目录，我们直接通过执行下面命令会直接生成

```
1 | ssh localhost # 按提示输入yes，然后键入hadoop密码
```

然后开始生成密钥

```
1 | cd ~/.ssh/ # 切换目录到ssh下
2 | ssh-keygen -t rsa # 生成密钥
```

生成密钥过程会有三个提示，不用管全部回车。

3. 授权

```
1 | cat id_rsa.pub >> authorized_keys # 加入授权
```

4. 修改权限

如果不修改文件 authorized_keys 权限为 600，会出现访问拒绝情况

```
1 | chmod 600 ./authorized_keys # 修改文件权限
```

5. 测试

```
1 | ssh localhost # ssh登陆
```

不用输入密码，直接登陆成功则说明配置正确。

```
[hadoop@VM_0_7_centos .ssh]$ ssh localhost
Last login: Thu Jan 23 17:05:19 2020 from 127.0.0.1
```

6. 新的风暴

- "倒霉的某个晚上，无论是MobaXterm，XShell 7，还是putty都无法进入服务器，差点破防了"

@[issue#30](#)，参考解决方案。

3 配置yum源

官方网站下载实在太慢，我们可以先配置一下阿里源来进行下载。

1. 切换到 yum 仓库

```
1 | cd /etc/yum.repos.d/
```

2. 备份下原repo文件

```
1 | sudo mv CentOS-Base.repo CentOS-Base.repo.backup
```

3. 下载阿里云repo文件

```
1 | sudo wget -O /etc/yum.repos.d/CentOS-7.repo
http://mirrors.aliyun.com/repo/Centos-7.repo
```

防止权限不足使用 `sudo` 命令。

4. 设置为默认repo文件

就是把阿里云repo文件名修改为 `CentOS-Base.repo`

```
1 | sudo mv CentOS-7.repo CentOS-Base.repo # 输入y
```

5. 生成缓存

```
1 | yum clean all
2 | yum makecache
```

4 配置Java环境

最开始下载的是 1.7 版本的JDK，后面出现的问题，重新下载 1.8 版本 JDK。

hadoop2 基于 *java* 运行环境，所以我们先要配置 *java* 运行环境。

1. 安装JDK

执行下面命令，经过实际测试前面几分钟一直显示镜像错误不可用。它会进行自己尝试别的源，等待一会儿就可以下载成功了。

```
1 | sudo yum install java-1.8.0-openjdk java-1.8.0-openjdk-devel
```

⚠ 此时默认安装位置是 `/usr/lib/jvm/java-1.8.0-openjdk`

其实，查找安装路径，可以通过以下命令：

```
1 | rpm -ql java-1.8.0-openjdk-devel | grep '/bin/javac'
```

- `rpm -ql <RPM包名>` : 查询指定RPM包包含的文件
- `grep <字符串>` : 搜索包含指定字符的文件

2. 配置环境变量

```
1 | vim ~/.bashrc # vim编辑配置文件
```

在文件最后面添加如下单独一行（指向JDK的安装位置），并保存：

```
1 | export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
```

```
# add chrome path
export PATH=$PATH:/opt/google/chrome
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk
```

最后是环境变量生效，执行：

```
1 | source ~/.bashrc
```

3. 测试

```
1 | echo $JAVA_HOME # 检验变量值
```

正常会输出 2. 环境变量JDK配置路径。

```
1 | java -version
```

正确配置会输出java版本号。

5 安装python

CentOS自带python2版本过低，我们进行python3安装。

1. yum查找python3

查找仓库存在的python3安装包

```
1 | yum list python3
```

```
python3.x86_64 3.6.8-10.el7 @base
```

2. yum 安装python3

```
1 | sudo yum install python3.x86_64
```

如果最开始会显示没有，等一会自动切换阿里源就可以进行安装了，同时还会安装相关依赖。

如果依旧报错，@[issue#35](#)，请安装epel使用阿里云镜像。

```
1 | yum install -y epel-release
2 | sudo yum install python3
```


6 更改hostname和hosts

建议结合ex0，虚拟机安装过程进行阅读理解下面设计原因。

在我们设计的集群中有三台主机，一台为master，两台slave。

我们设计这几台的ip地址为：

主机	IP地址
master	192.168.45.60
slave01	192.168.45.70
slave02	192.168.45.80

现在我们修改hostname文件和hosts文件，达到上述的设计。

1. 修改hostname

```
1 | sudo vi /etc/hostname
```

修改主机名为 `master`。目前只存在master主机，另外两台slave从机等待复制生成。再后续复制后，我们在修改从机的hostname文件。

2. 修改hosts文件

```
1 | sudo vi /etc/hosts
```

修改文件配置如下：

```
1 | 192.168.45.60 master
2 | 192.168.45.70 slave01
3 | 192.168.45.80 slave02
```

3. 重启生效

```
1 | sudo reboot
```

主机名更换表示hostname文件配置成功。

```
Last login: Tue Dec 1 00:24:47 2020 from 192.168.45.5
[hadoop@master ~]$
```

4.1.2 Hadoop安装

本文使用 `wget` 命令来下载 `hadoop`：[了解更多wget](#)。

使用的是[北理工镜像站](#)，下载 `hadoop`：

⚠️ ⚠️ hadoop在本教程是使用 `2.x` 版本，如果你使用的是 `3.x` 版本，请注意@[issue#29](#)：

- 下面配置**slaves文件**时，`3.x` 版本里已经不叫slaves，更名为**workers**！
- 即目录更改为：`/usr/local/hadoop/etc/hadoop/worker`

1. 下载

为防止证书验证出现的下载错误，加上 `--no-check-certificate`，相关讨论可见 [issue#1](#)

```
1 # 这里下载2.8.5版本，可能已失效，请去北理工镜像站，查看可下载的版本链接
2 # 建议下载版本低于3.0版本
3 sudo wget -O hadoop-2.8.5.tar.gz
https://mirrors.cnnic.cn/apache/hadoop/common/hadoop-2.8.5/hadoop-
2.8.5.tar.gz --no-check-certificate
```

◦ `wget -O <指定下载文件名> <下载地址>`

2. 解压

```
1 sudo tar -zxf hadoop-2.8.5.tar.gz -C /usr/local
```

把下载好的文件 `hadoop-2.8.5.tar.gz` 解压到 `/usr/local` 目录下

3. 修改文件

```
1 cd /usr/local/ # 切换到解压目录下
2 sudo mv ./hadoop-2.8.5/ ./hadoop # 将解压的文件hadoop-2.8.5重命名为
hadoop
3 sudo chown -R hadoop:hadoop ./hadoop # 修改文件权限
```

4. 测试

```
1 cd /usr/local/hadoop # 切换到hadoop目录下
2 ./bin/hadoop version # 输出hadoop版本号
```

```
[hadoop@VM_0_7_centos hadoop]$ ./bin/hadoop version
Hadoop 2.8.5
```

4.1.3 Spark安装

在前我们已经安装了 `hadoop`，现在我们来开始进行`spark` 安装。

这次下载根据官网推荐使用的清华源。

1. 下载

官网下载地址：[官网下载](#)

Download Apache Spark™

1. Choose a Spark release: `2.4.4 (Aug 30 2019)`
2. Choose a package type: `Pre-built with user-provided Apache Hadoop`
3. Download Spark: `spark-2.4.4-bin-without-hadoop.tgz`
4. Verify this release using the 2.4.4 [signatures](#), [checksums](#) and [project release KEYS](#).

◦ 这样选择的版本可以使用于大部分 `hadoop` 版本

点击上述链接，根据跳转的页面提示选择清华源下载：

注意，版本号可能发生变化，建议打开上述官网链接查看当前存在的版本。如我查看到只支持 `2.4.7` 版本（2020/09/17），那么需修改下面版本号：`2.4.4-->2.4.7`

```
1 sudo wget -O spark-2.4.7-bin-without-hadoop.tgz
http://mirrors.tuna.tsinghua.edu.cn/apache/spark/spark-2.4.7/spark-2.4.7-
bin-without-hadoop.tgz # 版本号发生变化，记得替换，下同
```

2. 解压

同前解压到 `/usr/local` 目录下

```
1 | sudo tar -zxf spark-2.4.7-bin-without-hadoop.tgz -C /usr/local
```

3. 设置权限

```
1 | cd /usr/local # 切换到解压目录
2 | sudo mv ./spark-2.4.7-bin-without-hadoop ./spark # 重命名解压文件
3 | sudo chown -R hadoop:hadoop ./spark # 设置用户hadoop为目录spark拥有者
```

4. 配置spark环境

先切换到 `/usr/local/spark` , (为了防止没权限, 下面用 `sudo`)

```
1 | cd /usr/local/spark
2 | cp ./conf/spark-env.sh.template ./conf/spark-env.sh
```

编辑 `spark-env.sh` 文件 :

```
1 | vim ./conf/spark-env.sh
```

在第一行添加下面配置信息, 使得Spark可以从Hadoop读取数据。

```
1 | export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
```

5. 配置环境变量

```
1 | vim ~/.bashrc
```

在 `.bashrc` 文件中添加如下内容:

```
1 | export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk # 之前配置的java环境变量
2 | export HADOOP_HOME=/usr/local/hadoop # hadoop安装位置
3 | export SPARK_HOME=/usr/local/spark
4 | export PYTHONPATH=$SPARK_HOME/python:$SPARK_HOME/python/lib/py4j-0.10.7-
src.zip:$PYTHONPATH
5 | export PYSARK_PYTHON=python3 # 设置pyspark运行的python版本
6 | export PATH=$HADOOP_HOME/bin:$SPARK_HOME/bin:$PATH
```

最后为了使得环境变量生效, 执行:

```
1 | source ~/.bashrc
```

6. 测试是否运行成功

```
1 | cd /usr/local/spark
2 | bin/run-example SparkPi
```

执行会输出很多信息, 也可以选择执行:

```
1 | bin/run-example SparkPi 2>&1 | grep "Pi is"
```

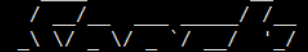
```
[hadoop@VM_0_7_centos spark]$ bin/run-example SparkPi 2>&1 | grep "Pi is"
Pi is roughly 3.1462557312786563
```

4.1.4 测试

1. 启动pyspark

```
1 cd /usr/local/spark
2 bin/pyspark
```

```
[hadoop@VM_0_7_centos spark]$ bin/pyspark
Python 3.6.8 (default, Aug  7 2019, 17:28:10)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] on linux
Type "help", "copyright", "credits" or "license()" for more information.
20/01/23 17:33:19 WARN util.Utils: Your hostname, VM_0_7_centos resolves to a loopback add
ress: 127.0.0.1; using 172.30.0.7 instead (on interface eth0)
20/01/23 17:33:19 WARN util.Utils: Set SPARK_LOCAL_IP if you need to bind to another addre
ss
20/01/23 17:33:19 WARN util.NativeCodeLoader: Unable to load native-hadoop library for you
r platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel
).
Welcome to
```



```
version 2.4.4
```

2. 简单测试

```
1 >>> 8 * 2 + 5
```

使用 `exit()` 命令可退出。

4.2 Hadoop+Spark 分布式环境搭建

4.2.1 Hadoop集群配置

VM虚拟机只需在一台虚拟机（master）上配置好，然后复制多份即可。

我们需要修改master主机上hadoop配置文件。

1. 切换目录

配置文件在 `/usr/local/hadoop/etc/hadoop` 目录下:

```
1 | cd /usr/local/hadoop/etc/hadoop
```

```
[hadoop@master hadoop]$ ll
total 160
-rw-r--r-- 1 hadoop hadoop 4942 Sep 10 2018 capacity-scheduler.xml
-rw-r--r-- 1 hadoop hadoop 1335 Sep 10 2018 configuration.xsl
-rw-r--r-- 1 hadoop hadoop 318 Sep 10 2018 container-executor.cfg
-rw-r--r-- 1 hadoop hadoop 1085 Jan 23 17:51 core-site.xml
-rw-r--r-- 1 hadoop hadoop 3804 Sep 10 2018 hadoop-env.cmd
-rw-r--r-- 1 hadoop hadoop 4666 Sep 10 2018 hadoop-env.sh
```

2. 修改文件 slaves

⚠ 注意, 3.x 版本的hadoop已将slaves文件重命名workers!

master主机作为 NameNode，而 slave01 作为 DataNode

```
1 # `3.x` 版本的hadoop , vim workers
2 vim slaves
```

修改如下：

```
1 # `3.x` 版本的hadoop , 请删除localhost
2 slave01
3 slave02
```

3. 修改文件 `core-site.xml`

```
1 vim core-site.xml
```

```
1 <configuration>
2   <property>
3     <name>hadoop.tmp.dir</name>
4     <value>/usr/local/hadoop/tmp</value>
5     <description>Abase for other temporary directories.
6   </description>
7   </property>
8   <property>
9     <name>fs.defaultFS</name>
10    <value>hdfs://master:9000</value>
11  </property>
12 </configuration>
```

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>file:/usr/local/hadoop/tmp</value>
    <description>Abase for other temporary directories.</description>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000</value>
  </property>
</configuration>
```

4. 修改 `hdfs-site.xml` :

```
1 vim hdfs-site.xml
```

```
1 <configuration>
2   <property>
3     <name>dfs.replication</name>
4     <value>3</value>
5   </property>
6   <property>
7     <name>mapred.job.tracker</name>
8     <value>master:9001</value>
9   </property>
10  <property>
11    <name>dfs.namenode.http-address</name>
12    <value>master:50070</value>
13  </property>
14 </configuration>
```

5. 修改 `mapred-site.xml.template`

⚠ 首先复制它产生一个新复制文件并命名为: `mapred-site.xml`

```
1 cp mapred-site.xml.template mapred-site.xml
```

然后修改文件 `vim mapred-site.xml`：

```
1 <configuration>
2   <property>
3     <name>mapreduce.framework.name</name>
4     <value>yarn</value>
5   </property>
6 </configuration>
```

6. 修改 `yarn-site.xml`

```
1 vim yarn-site.xml
```

```
1 <configuration>
2   <!-- Site specific YARN configuration properties -->
3   <property>
4     <name>yarn.nodemanager.aux-services</name>
5     <value>mapreduce_shuffle</value>
6   </property>
7   <property>
8     <name>yarn.resourcemanager.hostname</name>
9     <value>master</value>
10  </property>
11 </configuration>
```

4.2.2 Spark集群配置

1. 切换配置目录

```
1 cd /usr/local/spark/conf
```

2. 配置 `slaves` 文件

```
1 cp slaves.template slaves # 先把模板文件复制重命名
```

开始编辑 `vim slaves`，将默认内容 `localhost` 替换为以下：

```
1 slave01
2 slave02
```

3. 配置 `spark-env.sh` 文件

```
1 cp spark-env.sh.template spark-env.sh
```

开始编辑，添加下面内容：

```
1 vim spark-env.sh
```

```
1 export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
2 export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
3 export SPARK_MASTER_IP=192.168.45.60 #对于新版本SPARK_MASTER_HOST, 老版本
  SPARK_MASTER_IP
4 export SPARK_MASTER_PORT=7077
```

4.2.3 主从机复制

复制多份虚拟机只需复制母机的 .vmx 和 .vmdk 文件即可。

vmx文件是虚拟机系统的配置文件，vmdk则是虚拟磁盘文件。

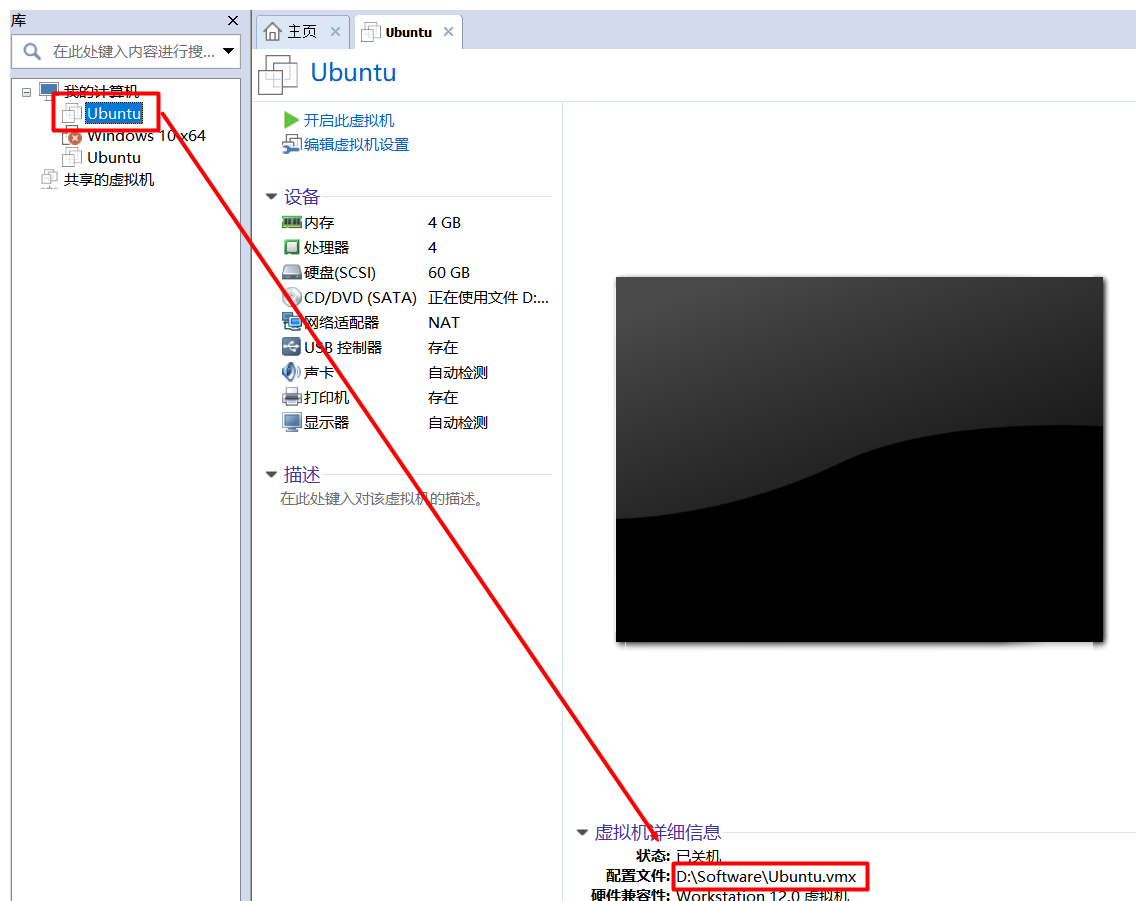
1. 新建文件夹

在合适目录建立文件夹：slave01,slave02。

2. 找到母机的 .vmx 和 .vmdk 文件

以管理员身份启动VMware--->点击我们的虚拟机，查看 .vmx 所在目录。 .vmdk 文件会出现在同目录下。

这里为Ubuntu，并非CentOS，但不影响操作。



3. 复制虚拟机

将 CentOS.vmx 和 CentOS.vmdk 复制到文件夹slave01,slave02中。

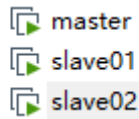
4. 打开复制的从虚拟机

用VMware：文件--->打开，分别选择slave01,slave02中的 CentOS.vmx 文件。

虚拟机即可自动识别新虚拟机。

5. VM中重命名虚拟机

我们在VMware中，右击虚拟机--->重命名虚拟机为如下：



6. 配置从机

以下操作分别在，从机slave01和slave02中进行。

◦ 更改IP地址

根据之前的4.1.1中的设计，修改从机的ip

```
1 # on slave01 or slave02
2 sudo vi /etc/sysconfig/network-scripts/ifcfg-ens33
```

- slave01的IPADDR=192.168.45.70

```
IPADDR=192.168.45.70
GATEWAY=192.168.45.1
NETMASK=255.255.255.0
DNS1=8.8.8.8
```

- slave02的IPADDR=192.168.45.80

```
IPADDR=192.168.45.80
GATEWAY=192.168.45.1
NETMASK=255.255.255.0
DNS1=8.8.8.8
```

◦ 更改主机名

```
1 # on slave01 or slave02
2 sudo vim /etc/hostname # 修改主机名（此处从主机分别为slave01或slave02）
3 reboot #重启生效
```

◦ 配置SSH免密登陆

无需配置。

因为三台虚拟机~/.ssh/直接复制的，私钥公钥都是相同的，故相互都可以直接登录。

4.2.4 集群测试及问题解决

1 Hadoop集群测试

1. master上启动集群

```
1 cd /usr/local/hadoop
2 bin/hdfs namenode -format # 注意，仅在第一次启动集群时使用该命令格式化！
3 sbin/start-all.sh
```

2. 测试

◦ 在master上

```
1 jps
```

master节点出现以下4个进程则配置成功：


```
[hadoop@master hadoop]$ jps
13191 NameNode
13869 Jps
13598 ResourceManager
13438 SecondaryNameNode
```

- 在 slave01 上

```
1 | jps
```

slave节点出现以下3个进程则配置成功：

```
[hadoop@slave01 ~]$ jps
5384 DataNode
5689 Jps
5549 NodeManager
```

3. 新的风暴：问题解决

Q1: slave 节点没有 DataNode 进程 / master 节点没有 namenode 进程？

这个问题一般是由于在启动集群多次执行格式化命令：

```
1 | bin/hdfs namenode -format
```

导致 hodoop 目录下 tmp/dfs/name/current 文件下的 VERSION 中的 namespaceId 不一致。

首先我们 在master 节点上 停止集群：

```
1 | cd /usr/local/hadoop # 切换到你的hadoop目录下
2 | sbin/stop-all.sh      # 关闭集群
```

- slave 节点删除 tmp

删除slave节点的临时 tmp 文件

```
1 | cd /usr/local # 切换到hadoop目录
2 | rm -rf ./hadoop/tmp
```

删除 tmp 文件，如法炮制在 其它节点 进行一样的操作：

```
1 | rm -rf ./hadoop/tmp # 后面格式化会重新生成，大胆删除
```

- 在master 节点删除 tmp

```
1 | cd /usr/local
2 | rm -rf ./hadoop/tmp
```

- 重新启动集群

在master节点执行以下操作：

```
1 | cd /usr/local/hadoop
2 | bin/hdfs namenode -format # 重新格式化
3 | sbin/start-all.sh
```

- 验证

在 master 节点执行以下操作：

```
1 | cd ~
2 | jps
```

```
[root@master hadoop]# jps
30496 NameNode
30882 ResourceManager
30717 SecondaryNameNode
621 Master
32127 Jps
```

在子节点再次输入 `jps` 命令：

```
1 | cd ~
2 | jps
```

```
[root@slave02 ~]# jps
5794 Jps
5613 NodeManager
5503 DataNode
```

ok~

Q2：启动集群后发现，`Slave` 节点没有 `NodeManager` 进程

```
[hadoop@slave01 local]$ jps
30673 DataNode
31172 Jps
```

少了nodemanager进程

△ 建议先尝试 Q1 方法，一般能解决大部分问题。

启动集群时可以知道，启动 `slave01` 节点 `notemanager` 进程相关日志在（最后不是 `.out` 是 `.log`）：

`/usr/local/hadoop/logs/yarn-hadoop-nodemanager-slave01.log`

```
[hadoop@master hadoop]$ sbin/start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [master]
master: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-namenode-master.out
slave01: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hadoop-datanode-slave01.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-secondarynamenode-master.out
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-resourcemanager-master.out
slave01: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-nodemanager-slave01.out
```

1. 查看日志

在 `slave01` 节点下

```
1 | vim /usr/local/hadoop/logs/yarn-hadoop-nodemanager-slave01.log
```

日志太多，我们在 命令模式 下，输入 `:$`，直接跳到最后一行：

```
2020-01-31 18:12:33,973 INFO org.apache.hadoop.service.AbstractService: Service org.apache.hadoop.yarn.server.nodemanager.containermanager.localizer.ResourceLocalizationService failed in state STARTED; cause: org.apache.hadoop.yarn.exceptions.YarnRuntimeException: java.net.BindException: Problem binding to [0.0.0.0:8040] java.net.BindException: Address already in use; For more details see: http://wiki.apache.org/hadoop/BindException
org.apache.hadoop.yarn.exceptions.YarnRuntimeException: java.net.BindException: Problem binding to [0.0.0.0:8040] java.net.BindException: Address already in use; For more details see: http://wiki.apache.org/hadoop/BindException
    at org.apache.hadoop.yarn.factories.impl.pb.RpcServerFactoryPBImpl.getServer(RpcServerFactoryPBImpl.java:138)
```

■ 很显然，显示端口 `8040` 被占用

2. 查看谁占用 `8040` 端口

```
1 | netstat -tln | grep 8040
```

```
[hadoop@slave01 local]$ netstat -tln | grep 8040
tcp6      0      0 :::8040      :::*          LISTEN
```

果然 8040 端口已经被占用

3. 释放端口

```
1 | sudo lsof -i :8040 # 查询占用8040端口进程pid
```

```
[hadoop@slave01 local]$ lsof -i :8040
COMMAND  PID  USER  FD  TYPE  DEVICE  SIZE/OFF  NODE  NAME
java     16961 hadoop 231u IPv6  38843230      0t0  TCP *:ampify (LISTEN)
```

杀死相应进程:

```
1 | sudo kill -9 16961
```

4. 测试

重新启动集群

```
1 | cd /usr/local/hadoop
2 | sbin/stop-all.sh
3 | sbin/start-all.sh
```

再次输入 jps 命令, 发现 slave01 节点 NodeManager 进程已经出现!

```
[hadoop@slave01 local]$ jps
30673 DataNode
6419 Jps
6035 NodeManager
```

2 Spark集群测试

在master主机上执行以下操作

1. 先启动hadoop集群

```
1 | cd /usr/local/hadoop/
2 | sbin/start-all.sh
```

2. 启动master节点

```
1 | cd /usr/local/spark/
2 | sbin/start-master.sh
```

master上运行 jps 命令可以看到:

```
[hadoop@master spark]$ jps
18160 Master
18224 Jps
13191 NameNode
13598 ResourceManager
13438 SecondaryNameNode
```

3. 启动所有slave节点

```
1 | sbin/start-slaves.sh
```

slave节点上运行 jps 命令可以看到:

```
[hadoop@slave01 ~]$ jps
9527 Jps
5384 DataNode
5549 NodeManager
9471 Worker
```

4. web UI查看

在浏览器输入：192.168.45.60:8080即可，其中192.168.45.60是master主机ip。

若想在虚拟机中浏览器查看，确保安装了GNOME桌面（4.1.1小节）。

5. 问题解决

⚠ 如果前面一切正常，Web UI 却无法正常工作显示worker，即workers数目为0。

一般出现这个问题，那么则可能是：**ip、端口、防火墙**等问题。对于虚拟机而言，则基本是**端口、防火墙**的问题。

【端口问题】

- 使用nmap工具测试

在slave节点测试master:7077端口是否被放通（master测试slave同理）：

```
1 | nmap -p 7077 master_ip
```

如果7077端口没有被放通：

- 本机防火墙放通指定端口

```
1 | sudo firewall-cmd --zone=public --add-port=7077/tcp --permanent
2 | sudo firewall-cmd --reload
```

- 重新启动集群

```
1 | sbin/start-master.sh
2 | sbin/start-slave.sh
```

【其它问题】

当然，这也可能是防火墙等问题导致，关闭防火墙可以解决。

```
1 | //Disable firewall
2 | systemctl disable firewalld
3 | systemctl stop firewalld
4 | systemctl status firewalld
5 |
6 | //Enable firewall
7 | systemctl enable firewalld
8 | systemctl start firewalld
9 | systemctl status firewalld
```

🎉🎉 聪明如你终于做到这步了，第一个实验完结，撒花 🎉🎉

5（已过时）伪分布式搭建

😊 选择伪分布式搭建的同学，**每一个组员**都需要在各自服务器上**独立完成**环境搭建。

5.1 Spark单机版搭建

在进行Hadoop、Spark环境搭建前，我们需要进行一些准备工作。

5.1.1 准备工作

1 配置用户

该小节主要是创建 Hadoop 用户。

1. 创建用户

```
1 useradd -m hadoop -s /bin/bash
```

同时设置用户密码：（如 123456）

```
1 passwd hadoop
```

2. 配置权限

为了方便，给用户 hadoop 等同 root 权限：

```
1 visudo # 执行 visudo命令进入vim编辑
```

找到如下位置，添加红框那一行配置权限：

```
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root    ALL=(ALL)    ALL
hadoop  ALL=(ALL)    ALL
```

3. 切换用户

配置完成好，我们切换到hadoop用户下：

```
1 su hadoop
```

⚠ 如非特殊说明，接下来所有命令都是Hadoop用户下完成！

2 配置SSH

为什么要配置ssh?

因为集群、单节点模式都需要用到 ssh登陆。同时每次登陆ssh都要输入密码是件蛮麻烦的事，我可以通过生成公钥配置来面密码登陆。

1. 生成密钥

为了生成 ~/.ssh 目录，我们直接通过执行下面命令会直接生成

```
1 ssh localhost # 按提示输入yes，然后键入hadoop密码
```

然后开始生成密钥

```
1 cd ~/.ssh/ # 切换目录到ssh下
2 ssh-keygen -t rsa # 生成密钥
```

生成密钥过程会有三个提示，不用管全部回车。

2. 授权

```
1 cat id_rsa.pub >> authorized_keys # 加入授权
```

3. 修改权限

如果不修改文件 `authorized_keys` 权限为 `600`，会出现访问拒绝情况

```
1 | chmod 600 ./authorized_keys # 修改文件权限
```

4. 测试

```
1 | ssh localhost # ssh登陆
```

不用输入密码，直接登陆成功则说明配置正确。

```
[hadoop@huang .ssh]$ ssh localhost
Last login: Fri Jan 24 11:16:06 2020 from 127.0.0.1
```

3 配置yum源

官方网站下载实在太慢，我们可以先配置一下阿里源来进行下载。

1. 切换到 yum 仓库

```
1 | cd /etc/yum.repos.d/
```

2. 备份下原repo文件

```
1 | sudo mv CentOS-Base.repo CentOS-Base.repo.backup
```

3. 下载阿里云repo文件

```
1 | sudo wget -O /etc/yum.repos.d/CentOS-7.repo
http://mirrors.aliyun.com/repo/Centos-7.repo
```

防止权限不足使用 `sudo` 命令。

4. 设置为默认repo文件

就是把阿里云repo文件名修改为 `CentOS-Base.repo`

```
1 | sudo mv CentOS-7.repo CentOS-Base.repo # 输入y
```

5. 生成缓存

```
1 | yum clean all
2 | yum makecache
```

4 配置Java环境

最开始下载的是 `1.7` 版本的JDK，后面出现的问题，重新下载 `1.8` 版本 JDK。

`hadoop2` 基于 `java` 运行环境，所以我们要先配置 `java` 运行环境。

1. 安装 JDK

执行下面命令，经过实际测试前面几分钟一直显示镜像错误不可用。它会进行自己尝试别的源，等待一会儿就可以下载成功了。

```
1 | sudo yum install java-1.8.0-openjdk java-1.8.0-openjdk-devel
```

⚠ 此时默认安装位置是 `/usr/lib/jvm/java-1.8.0-openjdk`

其实，查找安装路径，可以通过以下命令：

```
1 | rpm -ql java-1.8.0-openjdk-devel | grep '/bin/javac'
```

- `rpm -ql <RPM包名>`：查询指定RPM包包含的文件
- `grep <字符串>`：搜索包含指定字符的文件

2. 配置环境变量

```
1 | vim ~/.bashrc # vim编辑配置文件
```

在文件最后面添加如下单独一行（指向JDK的安装位置），并保存：

```
1 | export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
```

```
# add chrome path
export PATH=$PATH:/opt/google/chrome
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk
```

最后是环境变量生效，执行：

```
1 | source ~/.bashrc
```

3. 测试

```
1 | echo $JAVA_HOME # 检验变量值
```

正常会输出 `2. 环境变量JDK配置路径。`

```
1 | java -version
```

正确配置会输出java版本号。

5 安装python

CentOS自带python2版本过低，我们进行python3安装。

1. yum查找python3

查找仓库存在的python3安装包

```
1 | yum list python3
```

```
python3.x86_64 3.6.8-10.el7 @base
```

2. yum 安装python3

```
1 | sudo yum install python3.x86_64
```

如果最开始会显示没有，等一会自动切换阿里源就可以进行安装了，同时还会安装相关依赖。

5.1.2 hadoop 安装

本文使用 `wget` 命令来下载 `hadoop` : [了解更多wget](#)

使用的是[北理工镜像站](#) , 下载 `hadoop` :

Index of /apache/hadoop/common/hadoop-2.8.5

Name	Last modified	Size	Description
 Parent Directory	-	-	-
 hadoop-2.8.5-src.tar.gz	2018-09-18 23:47	34M	
 hadoop-2.8.5.tar.gz	2018-09-18 23:47	235M	

1. 下载

为防止证书验证出现的下载错误, 加上 `--no-check-certificate` , 相关讨论可见 [issue#1](#)

```
1 | sudo wget -O hadoop-2.8.5.tar.gz
   https://mirrors.cnnic.cn/apache/hadoop/common/hadoop-2.8.5/hadoop-
   2.8.5.tar.gz --no-check-certificate
```

o `wget -O <指定下载文件名> <下载地址>`

2. 解压

```
1 | sudo tar -zxf hadoop-2.8.5.tar.gz -C /usr/local
```

把下载好的文件 `hadoop-2.8.5.tar.gz` 解压到 `/usr/local` 目录下

3. 修改文件

```
1 | cd /usr/local/      # 切换到解压目录下
2 | sudo mv ./hadoop-2.8.5/ ./hadoop      # 将解压的文件hadoop-2.8.5重命名为
   hadoop
3 | sudo chown -R hadoop:hadoop ./hadoop  # 修改文件权限
```

4. 测试

```
1 | cd /usr/local/hadoop      # 切换到hadoop目录下
2 | ./bin/hadoop version      # 输出hadoop版本号
```

```
[hadoop@huang hadoop]$ ./bin/hadoop version
Hadoop 2.8.5
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r 0b8464d75227fcee2c6e7f2410377b3d53d3d5f8
```

5.1.3 spark安装

在前我们已经安装了 `hadoop` , 现在我们来开始进行`spark` 安装。

这次下载根据官网推荐使用的清华源。

1. 下载

官网下载地址: [官网下载](#)

Download Apache Spark™

1. Choose a Spark release:
2. Choose a package type:
3. Download Spark: [spark-2.4.4-bin-without-hadoop.tgz](#)
4. Verify this release using the 2.4.4 [signatures](#), [checksums](#) and [project release KEYS](#).

- 这样选择的版本可以使用于大部分 `hadoop` 版本

点击上述链接，根据跳转的页面提示选择清华源下载：

注意，版本号可能发生变化，建议打开上述官网链接查看当前存在的版本。如我查看到只支持 2.4.7 版本（2020/09/17），那么需修改下面版本号：2.4.4-->2.4.7

```
1 | sudo wget -O spark-2.4.7-bin-without-hadoop.tgz
   http://mirrors.tuna.tsinghua.edu.cn/apache/spark/spark-2.4.7/spark-2.4.7-
   bin-without-hadoop.tgz # 版本号发生变化记得替换，下同
```

2. 解压

同前解压到 `/usr/local` 目录下

```
1 | sudo tar -zxvf spark-2.4.7-bin-without-hadoop.tgz -C /usr/local
```

3. 设置权限

```
1 | cd /usr/local # 切换到解压目录
2 | sudo mv ./spark-2.4.7-bin-without-hadoop ./spark # 重命名解压文件
3 | sudo chown -R hadoop:hadoop ./spark # 设置用户hadoop为目录spark拥有者
```

4. 配置spark环境

先切换到 `/usr/local/spark`，（为了防止没权限，下面用 `sudo`）

```
1 | cd /usr/local/spark
2 | cp ./conf/spark-env.sh.template ./conf/spark-env.sh
```

编辑 `spark-env.sh` 文件：

```
1 | vim ./conf/spark-env.sh
```

在第一行添加下面配置信息，使得Spark可以从Hadoop读取数据。

```
1 | export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
```

5. 配置环境变量

```
1 | vim ~/.bashrc
```

在 `.bashrc` 文件中添加如下内容：

最后为了使得环境变量生效，执行：

6. 测试是否运行成功

执行会输出很多信息，也可以选择执行：

5.1.4 测试

1. 启动pyspark

```
[hadoop@huang spark]$ bin/pyspark
Python 3.6.8 (default, Aug 7 2019, 17:28:10)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-39)] on linux
Type "help", "copyright", "credits" or "license" for more information.
20/01/24 11:28:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platfo
rm... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to

      /_/_\_/__/_/_/_/_/_/_/_/_/_/_\_
     /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\_
    /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\_
   /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\_
  /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\_
 /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\_
/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\_

version 2.4.4
```

2. 简单测试

使用 `exit()` 命令可退出。

5.2 Hadoop+Spark 分布式环境搭建

5.2.1 Hadoop集群配置

Hadoop文件配置

我们需要修改hadoop配置文件。

1. 切换目录

配置文件在 `/usr/local/hadoop/etc/hadoop` 目录下：

```
1 | cd /usr/local/hadoop/etc/hadoop
```

```
[hadoop@master hadoop]$ ll
total 160
-rw-r--r-- 1 hadoop hadoop 4942 Sep 10 2018 capacity-scheduler.xml
-rw-r--r-- 1 hadoop hadoop 1335 Sep 10 2018 configuration.xml
-rw-r--r-- 1 hadoop hadoop 318 Sep 10 2018 container-executor.cfg
-rw-r--r-- 1 hadoop hadoop 1085 Jan 23 17:51 core-site.xml
-rw-r--r-- 1 hadoop hadoop 3804 Sep 10 2018 hadoop-env.cmd
-rw-r--r-- 1 hadoop hadoop 4666 Sep 10 2018 hadoop-env.sh
```

2. 修改文件 `core-site.xml`

```
1 | vim core-site.xml
```

```
1 | <configuration>
2 |     <property>
3 |         <name>hadoop.tmp.dir</name>
4 |         <value>/usr/local/hadoop/tmp</value>
5 |         <description>Abase for other temporary directories.
6 |     </description>
7 |     </property>
8 |     <property>
9 |         <name>fs.defaultFS</name>
10 |         <value>hdfs://0.0.0.0:9000</value>
11 |     </property>
12 | </configuration>
```

⚠ 实际测试必须要 `hdfs://0.0.0.0:9000` 才能使用 `hdfs` 服务。

⚠ 有可能依旧报错： `Error JAVA_HOME is not set and could not be found` -

- 配置 `hadoop-env.sh`

```
1 | cd /usr/local/hadoop/etc/hadoop
2 | vim hadoop-env.sh
```

配置 `JAVA_HOME` 路径如下：

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
```

3. 修改 `hdfs-site.xml` :

```
1 | vim hdfs-site.xml
```

```
1 | <configuration>
2 |     <property>
3 |         <name>dfs.replication</name>
4 |         <value>1</value>
5 |     </property>
```

```

6      <property>
7          <name>dfs.namenode.name.dir</name>
8          <value>file:/usr/local/hadoop/tmp/dfs/name</value>
9      </property>
10     <property>
11         <name>dfs.datanode.data.dir</name>
12         <value>file:/usr/local/hadoop/tmp/dfs/data</value>
13     </property>
14 </configuration>

```

集群启动测试

1. 启动集群

```

1  cd /usr/local/hadoop
2  bin/hdfs namenode -format # 注意，仅在第一次启动集群时使用该命令格式化！
3  sbin/start-all.sh

```

2. 测试

```
1 jps
```

出现以下6个进程则配置成功：

```

[hadoop@huang hadoop]$ jps
3024 DataNode
3184 SecondaryNameNode
3729 Jps
3331 ResourceManager
3431 NodeManager
2895 NameNode

```

5.2.2 Spark集群配置

Spark配置

1. 切换配置目录

```
1 cd /usr/local/spark/conf
```

2. 配置 spark-env.sh 文件

```
1 cp spark-env.sh.template spark-env.sh
```

开始编辑，添加下面内容：

```
1 vim spark-env.sh
```

```

1 export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
2 export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
3 export SPARK_MASTER_IP=master

```

启动Spark集群

执行以下操作

1. 先启动hadoop集群

```
1 | cd /usr/local/hadoop/  
2 | sbin/start-all.sh
```

2. 启动spark集群

```
1 | cd /usr/local/spark/  
2 | sbin/start-all.sh
```

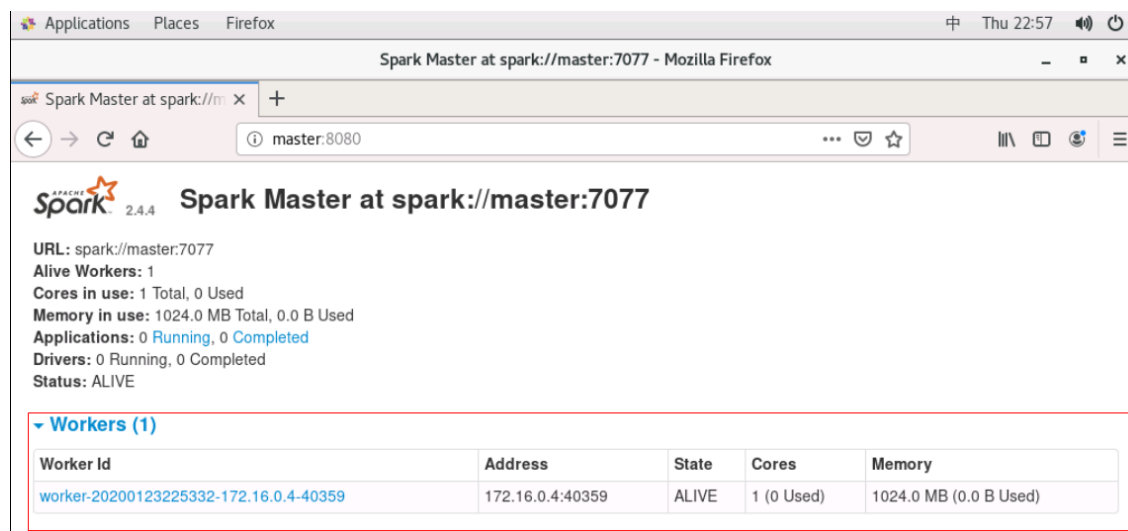
运行 `jps` 命令可以看到:

```
[root@huang ~]# jps  
3024 DataNode  
3184 SecondaryNameNode  
3331 ResourceManager  
3909 Worker  
23798 Jps  
3431 NodeManager  
3770 Master
```

3. web UI查看

打开腾讯云控制台, 选择 VNC 登陆服务器, 在浏览器上输入: `master:8080`。

如果出现下面界面则表示 *Hadoop+Spark* 分布式环境搭建成功!



🎉🎉 聪明如你终于做到这步了, 第一个实验完结, 撒花 🎉🎉

4. WebUI 显示不正常

⚠️ 如果前面一切正常, Web UI 却无法显示worker。

- 查看slave节点相关 `spark` 日志发现报错: 无法访问 `<master外网ip>:7070`, 多次连接失败。

- 关闭集群, 重启启动集群, 执行如下命令

```
1 | sbin/start-master.sh # 先启动master  
2 | sbin/start-slave.sh spark://<master内网ip>:7077 # 指定master内网  
   | ip启动slaves节点
```

- 如果依旧不行，考虑：登陆控制台 --> 创建安全组（选择**放通所有端口**） --> 将master加入刚创建的安全组
- 重新按第一步启动集群，一般都可以正常显示了

相关的一些讨论也可参考：[issue#3 @trevery](#)

6 实验总结

在本次实验中，我们进行了：

- Hadoop/Spark的单机版；
- Hadoop/Spark的分布式搭建。

实验过程中，相信你也遇上了不少问题。在开源项目下及时去查看 `issue` 是个很棒的行为，当然Google和Stackoverflow也是个不错的选择。

接下来的实验中，我们将通过几个有趣小项目，来进行大数据实践开发。它们设计之初，被设计的尽量精简，需要你能有所收获~