

## Hadoop on ubuntu 部署方法

环境:

ubuntu14.04 64 位

### 1、创建 hadoop 用户

注: **hadoop** 集群要求 **master** 和 **slave** 的用户名一致, 才可搭建, 因此, 或者在安装 **ubuntu** 系统新建用户的时候就把用户名设置为统一名称 (如: **nathychen**), 或者新建一个以统一名称命名的新用户。

创建新用户方法:

首先按 **ctrl+alt+t** 打开终端窗口, 输入如下命令创建新用户:

- **sudo useradd -m nathychen -s /bin/bash**

这条命令创建了可以登陆的 **nathychen** 用户, 并使用 **/bin/bash** 作为 **shell**。

**Ubuntu** 终端复制粘贴快捷键: 在 **Ubuntu** 终端窗口中, 复制粘贴的快捷键需要加上 **shift**, 即 **ctrl+shift+v**。

然后使用如下命令设置密码, 按照提示输入两次密码:

- **sudo passwd nathychen**

可为 **nathychen** 用户增加管理员权限, 方便部署:

- **sudo adduser nathychen sudo**

最后注销当前用户, 登录新创建的 **nathychen** 用户。

### 2、更新 apt 和 vim

#### 2.1、apt

**apt** 是一个用于自动从互联网的软件仓库中搜索、安装、升级、卸载软件或操作系统的 **linux** 命令。

用 **nathychen** 用户登录后, 先更新一下 **apt**, 后续使用 **apt** 安装软件, 如果没更新可能有一些软件安装不了。打开终端, 执行如下命令:

- **sudo apt-get update**

若出现如下“**Hash** 校验和不符”的提示, 可通过更改软件源来解决。若没有该问题, 则不需要更改。

```
W: 无法下载 bzip2:/var/lib/apt/lists/partial/cn.archive.ubuntu.com_ubuntu_dists_
trusty-updates_universe_i18n_Translation-en Hash 校验和不符
W: 无法下载 bzip2:/var/lib/apt/lists/partial/cn.archive.ubuntu.com_ubuntu_dists_
trusty-backports_universe_binary-amd64_Packages Hash 校验和不符
E: Some index files failed to download. They have been ignored, or old ones used
instead.
hadoop@star-lab:~$
```

## 2.2、vim

后续需要更改一些配置文件，推荐使用 **vim**。

**vim** 是一个在终端进行的文本编辑器。

首先下载 **vim**：

- **sudo apt-get install vim**

安装软件的时候需要确认，输入 **y** 即可。

下面简要介绍常用的 **vim** 方法：

**vim** 分为 **normal** 模式和 **insert** 模式，可以粗略地认为：**normal** 模式是进行浏览、删除的；**insert** 模式是进行文本写入的。换言之，**normal** 模式不可以写入文本。

(1) 当启动 **vim** 后，**vim** 在 **normal** 模式下。

(2) 按下键 **i**，进入 **insert** 模式。

(3) 此时可以像记事本一样输入文本。

(4) 按下键 **esc**，返回 **normal** 模式。

(5) 在 **normal** 模式下键入 **:wq**，显示在最低下一行，回车即保存退出

(：**w** 存盘，：**q** 退出)

详细教程见此：

<http://blog.csdn.net/niushuai666/article/details/7275406>

## 3、安装 SSH、配置 SSH 无密码登录

集群、单节点模式都需要用到 **SSH** 登录（类似于远程登录），**ubuntu** 默认已安装了 **SSH client**，此外还需要安装 **SSH server**：

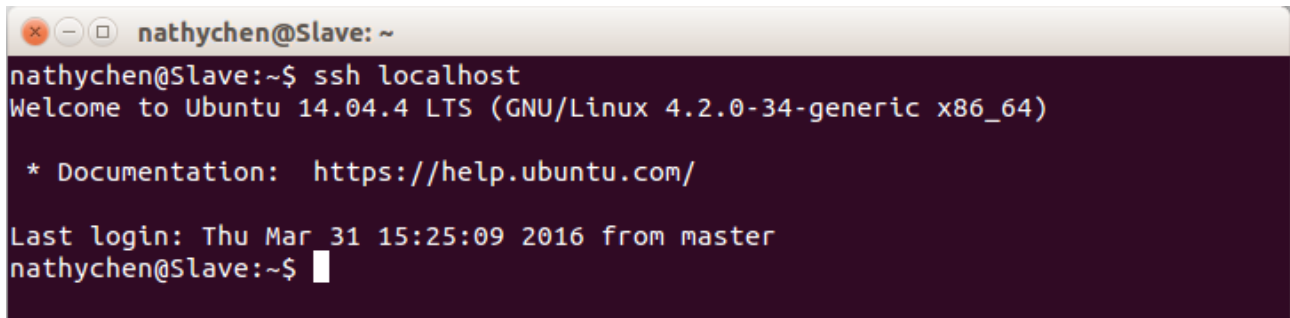
- **sudo apt-get install openssh-server**

安装后，可以使用如下命令登录本机：

- **ssh localhost**

此时会有如下提示（**ssh** 首次登录提示），输入 **yes**。然后按提示输入密码 **hadoop**，这样就登录到本机了。

登录成功是这样：

A terminal window titled 'nathychen@Slave: ~' showing the output of the 'ssh localhost' command. The output includes a welcome message for Ubuntu 14.04.4 LTS, documentation link, and login history.

```
nathychen@Slave:~$ ssh localhost
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-34-generic x86_64)

* Documentation:  https://help.ubuntu.com/

Last login: Thu Mar 31 15:25:09 2016 from master
nathychen@Slave:~$
```

接下来配置成 **ssh** 无密码登录：

首先退出刚才的 **ssh**，回到原先的终端窗口，然后利用 **ssh-keygen** 生成密钥，并将密钥加入到授权中：

- **exit**
- **cd ~/.ssh**
- **ssh-keygen -t rsa**                      #会有提示，都按回车就可以
- **cat ./id\_rsa.pub >> ./authorized\_keys**

此时再用 **ssh localhost** 命令，无需输入密码就可以直接登录了。

- **ssh localhost**

#### 4、安装 java 环境

直接安装 **OpenJDK7** 比较方便。

- **sudo apt-get install openjdk-7-jre**
- **sudo apt-get install openjdk-7-jdk**

安装好 **OpenJDK** 后，需要找到相应的安装路径，这个路径是用于配置 **JAVA\_HOME** 环境变量的。执行如下命令：

- **dpkg -L openjdk-7-jdk | grep '/bin/javac'**

该命令会输出一个路径，除去路径末尾的 “**/bin/javac**”，剩下的就是正确的路径了。如输出路径为 **/usr/lib/jvm/java-7-openjdk-amd64/bin/javac**，则我们需要的路径为 **/usr/lib/jvm/java-7-openjdk-amd64**。

接着配置 **JAVA\_HOME** 环境变量，为方便，我们在 **~/.bashrc** 中进行设置。

- **vim ~/.bashrc**

在文件最前面添加如下单独一行（注意 **=** 号前后不能有空格），将“**JDK 安装路径**”改为上述命令得到的路径，并保存：

- **export JAVA\_HOME=JDK 安装路径**

如下图所示（该文件原本可能不存在，内容为空，这不影响）：

```
hadoop@DBLab-XMU: ~  
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64  
  
# ~/.bashrc: executed by bash(1) for non-login shells.  
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)  
# for examples  
  
# If not running interactively, don't do anything  
case $- in  
    *i*) ;;  
    *)  
        ;;  
esac
```

接着还需要让该环境变量生效，执行如下代码：

- `source ~/.bashrc`

设置好后我们来检验一下是否设置正确：

- `echo $JAVA_HOME`
- `java -version` # (1)
- `$JAVA_HOME/bin/java -version` # (2)

如果设置正确的话，(1)和(2)输出的 `java` 版本信息应该是一样的，如下图所示：

```
hadoop@DBLab-XMU: ~  
hadoop@DBLab-XMU:~$ vim ~/.bashrc  
hadoop@DBLab-XMU:~$ source ~/.bashrc  
hadoop@DBLab-XMU:~$ echo $JAVA_HOME  
/usr/lib/jvm/java-7-openjdk-amd64  
hadoop@DBLab-XMU:~$ java -version  
java version "1.7.0_91"  
OpenJDK Runtime Environment (IcedTea 2.6.3) (7u91-2.6.3-0ubuntu0.14.04.1)  
OpenJDK 64-Bit Server VM (build 24.91-b01, mixed mode)  
hadoop@DBLab-XMU:~$ $JAVA_HOME/bin/java -version  
java version "1.7.0_91"  
OpenJDK Runtime Environment (IcedTea 2.6.3) (7u91-2.6.3-0ubuntu0.14.04.1)  
OpenJDK 64-Bit Server VM (build 24.91-b01, mixed mode)
```

这样，Hadoop 所需的 JAVA 运行环境就安装好了。

## 5、安装 Hadoop2

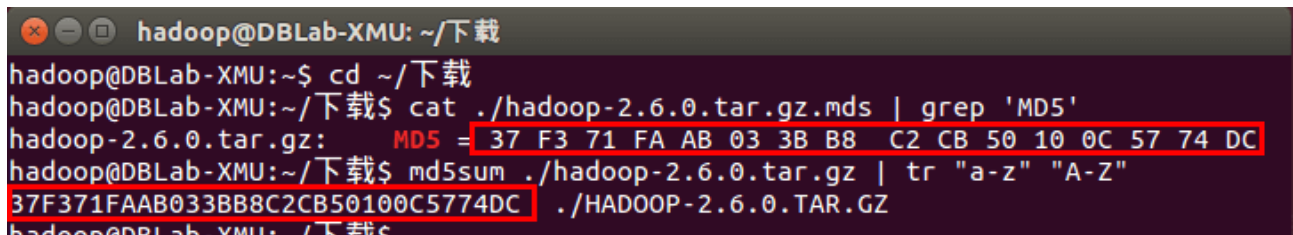
Hadoop 2 可以通过 <http://mirror.bit.edu.cn/apache/hadoop/common/> 或者 <http://mirrors.cnnic.cn/apache/hadoop/common/> 下载，本教程选择的是 2.6.0 版本，下载时请下载 `hadoop-2.x.y.tar.gz` 这个格式的文件，这是编译好的，另一个包含 `src` 的则是 Hadoop 源代码，需要进行编译才可使用。

下载时强烈建议也下载 `hadoop-2.x.y.tar.gz.md5` 这个文件，该文件包含了检验值可用于检查 `hadoop-2.x.y.tar.gz` 的完整性，否则若文件发生了损坏或下载不完整，Hadoop 将无法正常运行。

本文涉及的文件均通过浏览器下载，默认保存在“下载”目录中（若不是请自行更改 tar 命令的相应目录）。另外，如果你用的不是 2.6.0 版本，则将所有命令中出现的 2.6.0 更改为你所使用的版本。

- `cat ~/下载/hadoop-2.6.0.tar.gz.mds | grep 'MD5'`
- `md5sum ~/下载/hadoop-2.6.0.tar.gz | tr "a-z" "A-Z"`

若文件不完整则这两个值一般差别很大，可以简单对比下前几个字符跟后几个字符是否相等即可，如下图所示，如果两个值不一样，请务必重新下载。



```
hadoop@DBLab-XMU: ~/下载
hadoop@DBLab-XMU:~/下载$ cd ~/下载
hadoop@DBLab-XMU:~/下载$ cat ./hadoop-2.6.0.tar.gz.mds | grep 'MD5'
hadoop-2.6.0.tar.gz: MD5 = 37 F3 71 FA AB 03 3B B8 C2 CB 50 10 0C 57 74 DC
hadoop@DBLab-XMU:~/下载$ md5sum ./hadoop-2.6.0.tar.gz | tr "a-z" "A-Z"
37F371FAAB033BB8C2CB50100C5774DC ./HADOOP-2.6.0.TAR.GZ
```

我们选择将 Hadoop 安装至 /usr/local/ 中：

- `sudo tar -zxf ~/下载/hadoop-2.6.0.tar.gz -C /usr/local`  
#解压到/usr/local 中
- `cd /usr/local`
- `sudo mv ./hadoop-2.6.0/ ./hadoop` #将文件夹名改为 hadoop
- `sudo chown -R nathychen:nathychen ./hadoop`

Hadoop 解压后即可使用。输入如下命令来检查 Hadoop 是否可用，成功则会显示 Hadoop 版本信息：

- `cd /usr/local/hadoop`
- `./bin/hadoop version`

## 6、Hadoop 单机配置（非分布式）

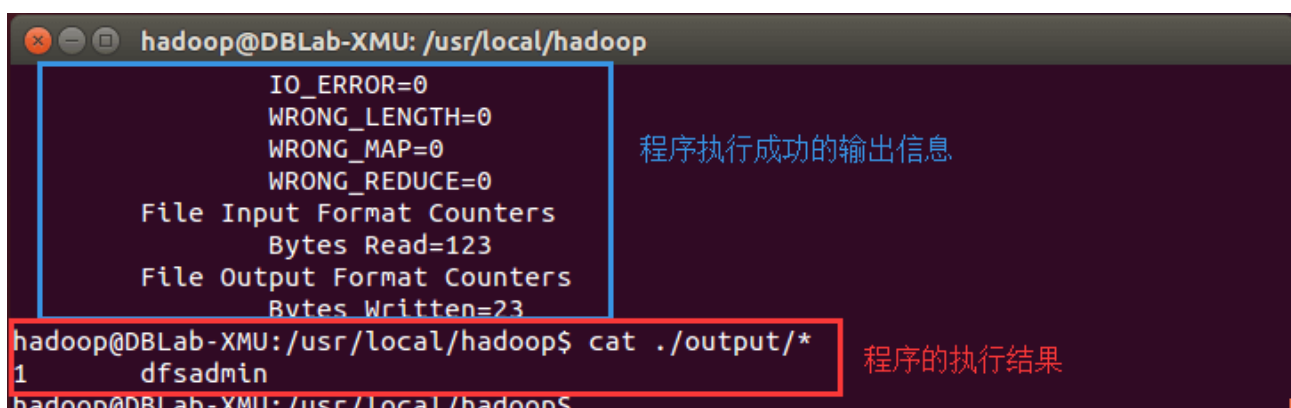
Hadoop 默认模式为非分布式模式，无需进行其他配置即可运行。非分布式即单 JAVA 进程，方便进行调试。

现在可以执行例子来感受一下 hadoop 的运行。Hadoop 附带了丰富的例子（运行 `./bin/hadoop jar ./share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar` 可以看到所有例子），包括 wordcount、terasort、join、grep 等。

在此选择运行 **grep** 例子，我们将 **input** 文件夹中的所有文件作为输入，筛选当中符合正则表达式 **dfs[a-z.]** 的单词并统计出现的次数，最后输出结果到 **output** 文件夹中。

- `cd /usr/local/hadoop`
- `mkdir ./input` #新建 input 文件夹
- `cp ./etc/hadoop/*.xml ./input` #将配置文件作为输入文件，拷贝至 input 文件夹
- `./bin/hadoop jar ./share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar grep ./input ./output 'dfs[a-z.]'`
- `cat ./output/*` #查看运行结果

执行成功后如下所示，输出了作业的相关信息，输出到结果是符合正则的单词 **dfsadmin** 出现了 1 次。



```
hadoop@DBLab-XMU: /usr/local/hadoop
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=123
File Output Format Counters
Bytes Written=23
hadoop@DBLab-XMU: /usr/local/hadoop$ cat ./output/*
1 dfsadmin
hadoop@DBLab-XMU: /usr/local/hadoop$
```

注意，**hadoop** 默认不会覆盖结果文件，因此再次运行上面实例会提示出错，需要先 将 **./output** 删除。

- `rm -r ./output`

## 7、Hadoop 伪分布式配置

**Hadoop** 可以在单节点上以伪分布式的方式运行，**Hadoop** 进程以分离的 **java** 进程来运行，节点既作为 **NameNode** 也作为 **DataNode**，同时，读取的是 **HDFS** 中的文件。

**Hadoop** 的配置文件位于 **/usr/local/hadoop/etc/hadoop** 中，伪分布式需要修改 2 个配置文件 **core-site.xml** 和 **hdfs-site.xml**。**Hadoop** 的配置文件是 **xml** 格式，每个配置以声明 **property** 的 **name** 和 **value** 的方式来实现。

修改配置文件 **core-site.xml**（通过 **gedit** 编辑会比较方便），将当中的

```
<configuration>
</configuration>
```

修改为下面配置：

```
<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://localhost:9000</value>
  </property>
</configuration>
```

同样的，修改配置文件 `hdfs-site.xml`：

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///usr/local/hadoop/tmp/dfs/name</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///usr/local/hadoop/tmp/dfs/data</value>
  </property>
</configuration>
```

配置完成后，执行 `NameNode` 的格式化：

- `./bin/hdfs namenode -format`



成功的话，会看到“successfully formatted”和“Exiting with status 0”的提示，若为“Exiting with status 1”则是出错。

```
nathychen@Slave: /usr/local/hadoop
16/04/01 09:27:42 INFO common.Storage: Storage directory /usr/local/hadoop/
fs/name has been successfully formatted.
16/04/01 09:27:42 INFO namenode.NNStorageRetentionManager: Going to retain
ges with txid >= 0
16/04/01 09:27:42 INFO util.ExitUtil: Exiting with status 0
16/04/01 09:27:42 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at Slave/192.168.10.46
*****/
```

注意：

在这一步时若提示 `Error: JAVA_HOME is not set and could not be found.` 的错误，则需要在文件 `./etc/hadoop/hadoop-env.sh` 中设置 `JAVA_HOME` 变量，即在该文件中找到：

```
export JAVA_HOME=${JAVA_HOME}
```

将这一行改为 `JAVA` 安装位置：

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

再重新尝试格式化即可。

接着开启 `NameNode` 和 `DataNode` 守护进程。

- `./sbin/start-ds.sh`

若出现如下 `SSH` 提示，输入 `yes` 即可。

```
hadoop@DBLab-XTU:/usr/local/hadoop$ sbin/start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hadoop-na
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hadoop-da
Starting secondary namenodes [0.0.0.0]
The authenticity of host '0.0.0.0 (0.0.0.0)' can't be established.
ECDSA key fingerprint is a9:28:e0:4e:89:40:a4:cd:75:8f:0b:8b:57:79:67:86.
Are you sure you want to continue connecting (yes/no)? yes
```

启动时可能会出现如下 `WARN` 提示：`WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable` `WARN` 提示可以忽略，并不会影响正常使用。

启动完成后，可以通过命令 `jps` 来判断是否成功启动，若成功启动则会列出如下进程：`“NameNode”`、`“DataNode”` 和 `“SecondaryNameNode”`（如果 `SecondaryNameNode` 没有启动，请运行 `sbin/stop-dfs.sh` 关闭进程，然后再次尝

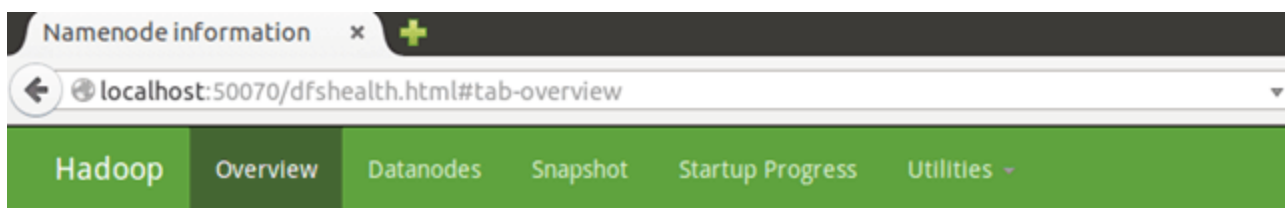


试启动尝试)。如果没有 **NameNode** 或 **DataNode**，那就是配置不成功，请仔细检查之前步骤，或通过查看启动日志排查原因。

- **jps**

```
hadoop@powerxing-M1:/usr/local/hadoop$ jps
7100 Jps
6867 SecondaryNameNode
6445 NameNode
6594 DataNode
```

成功启动后，可以访问 web 界面 <http://localhost:50070> 查看 **NameNode** 和 **Datanode** 信息，还可以在线查看 **HDFS** 中的文件。



## Overview 'localhost:9000' (active)

Started:	Thu Aug 07 10:33:16 CST 2014
Version:	2.4.1, r1604318
Compiled:	2014-06-21T05:43Z by jenkins from branch-2.4.1
Cluster ID:	CID-788afdca-c5d4-46b0-873f-68c7e843474c
Block Pool ID:	BP-1018774623-127.0.1.1-1407377625336

@给力星  
powerxing.com

## 8、运行 Hadoop 伪分布式实例

上面的单机模式，**grep** 例子读取的是本地数据，伪分布式读取的则是 **HDFS** 上的数据，要使用 **HDFS**，首先需要在 **HDFS** 中创建用户目录：

- `./bin/hdfs dfs -mkdir -p /user/hadoop`

接着将 `./etc/hadoop` 中的 `xml` 文件作为输入文件复制到分布式文件系统中，即将 `/usr/local/hadoop/etc/hadoop` 复制到分布式文件系统 `/user/hadoop/input` 中。我们使用的是 `hadoop` 用户，并且已创建相应的用户目录 `/user/hadoop`，因此在命令中就可以使用相对路径如 `input`，其对应的绝对路径就是 `/user/hadoop/input`：

- `./bin/hdfs ds -mkdir input`

- `./bin/hdfs dfs -put ./etc/hadoop/*.xml input`

复制完成后，可以通过如下命令查看文件列表：

- `./bin/hdfs dfs -ls input`

伪分布式运行 MapReduce 作业的方式跟单机模式相同，区别在于伪分布式读取的是 HDFS 的文件（可以将单机步骤中创建的本地 `input` 文件夹，输出结果 `output` 文件夹都删掉来验证这一点）。

- `./bin/hadoop jar ./share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar grep input output 'dfs[a-z.]+'`

查看运行结果的命令（查看的是位于 HDFS 中的输出结果）：

- `./bin/hdfs dfs -cat output/*`

结果如下，注意到刚才我们已经更改了配置文件，所以运行结果不同。

```
File Input Format Counters
  Bytes Read=219
File Output Format Counters
  Bytes Written=77
hadoop@DBLab-XMU:/usr/local/hadoop$ bin/hdfs dfs -cat output/*
1      dfsadmin
1      dfs.replication
1      dfs.namenode.name.dir
1      dfs.datanode.data.dir
hadoop@DBLab-XMU:/usr/local/hadoop$
```

我们也可以将运行结果取回到本地：

- `rm -r ./output` #先删除本地的 output 文件夹（如果存在）
- `./bin/hdfs dfs -get output ./output` #将 HDFS 上的 output 文件夹拷贝到本机
- `cat ./output/*`

Hadoop 运行程序时，输出目录不能存在，否则会提示错误  
“org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory hdfs://localhost:9000/user/hadoop/output already exists”，因此若要再次执行，需要执行如下命令删除 `output` 文件夹：

- `./bin/hds dfs -rm -r output` #删除 output 文件夹

若要关闭 Hadoop，则运行

- `./sbin/stop-dfs.sh`

注意：

下次启动 `hadoop` 时，无需进行 `NameNode` 的初始化，只需要运行 `./sbin/start-dfs.sh` 就可以。

## 9、启动 YARN

`Yarn` 是从 `MapReduce` 中分离出来的，负责资源管理与任务调度。`YARN` 运行于 `MapReduce` 之上，提供了高可用性、高扩展性。

上述通过 `./sbin/start-dfs.sh` 启动 `Hadoop`，仅仅是启动了 `MapReduce` 环境，我们可以启动 `YARN`，让 `YARN` 来负责资源管理与任务调度。

首先修改配置文件 `mapred-site.xml`，这边需要先进行重命名：

- `mv ./etc/hadoop/mapred-site.xml-template ./etc/hadoop/mapred-site.xml`

然后再进行编辑，同样使用 `gedit` 编辑会比较方便，可在文件夹里找到右键 `gedit` 打开，也可以通过下列命令打开：

- `gedit ./etc/hadoop/mapred-site.xml`

编辑配置文件如下：

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

接着修改配置文件 `yarn-site.xml`：

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

然后就可以启动 `YARN` 了（需要先执行过 `./sbin/start-dfs.sh`）：

- `./sbin/start-yarn.sh` #启动 YARN
- `./sbin/mr-jobhistory-daemon.sh start historyserver` #开启历史服务器，才能在 web 中查看任务运行情况

开启后通过 `jps` 查看，可以看到多了 `NodeManager` 和 `ResourceManager` 两个后台进程，如下图所示。

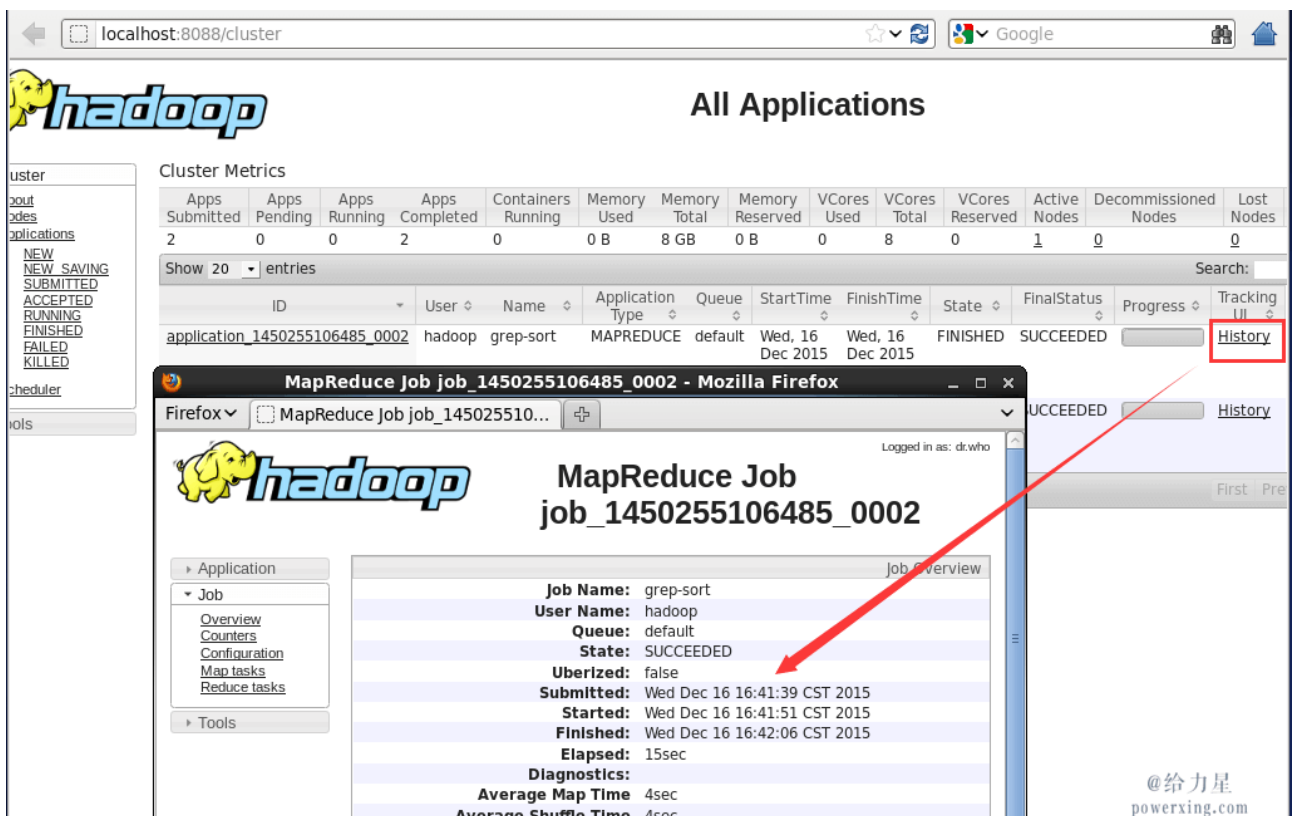
```
hadoop@DBLab-XMU: /usr/local/hadoop
hadoop@DBLab-XMU:/usr/local/hadoop$ ./sbin/start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-resource
manager-DBLab-XMU.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hadoop-n
odemanager-DBLab-XMU.out
hadoop@DBLab-XMU:/usr/local/hadoop$ jps
13519 NameNode
13829 SecondaryNameNode
15284 NodeManager
15317 Jps
13633 DataNode
15163 ResourceManager
```

启动YARN的输出信息

成功启动后多了 `NodeManager` 和 `ResourceManager`

@给力星  
powerxing.com

启动 `YARN` 之后，运行实例的方法还是一样的，仅仅是资源管理方式、任务调度不同。观察日志信息可以发现，不起用 `YARN` 时，是“`mapred.LocalJobRunner`”在跑任务，启用 `YARN` 之后，是“`mapred.YARNRUNNER`”在跑任务。启动 `YARN` 有个好处是可以通过 `web` 界面查看任务的运行情况：<http://localhost:8088/cluster>，如下图所示。



但是 `YARN` 主要是为集群提供更好的资源管理与任务调度，然而这在单机上体现不出价值，反而会使程序跑得稍慢些。因此在单机上是否开启 `YARN` 就看实际情况了。

注意：不启动 `YARN` 需重命名 `mapred-site.xml`

如果不想启动 YARN，务必把配置文件 `mapred-site.xml` 重命名，改成 `mapred-site.xml.template`，需要用时改回来就行。否则在该配置文件存在，而未开启 YARN 的情况下，运行程序会提示“`Retrying connect to server: 0.0.0.0/0.0.0.0:8032`”的错误，这也是为何该配置文件初始文件名为 `mapred-site.xml.template`。

同样的，关闭 YARN 的脚本如下：

- `./sbin/stop-yarn.sh`
- `./sbin/mr-jobhistory-daemon.sh stop historyserver`

至此，你已经掌握 Hadoop 的配置和基本使用了。

## 10、配置 PATH 环境变量

上面的教程中，我们都是先进入到 `/usr/local/hadoop` 目录中，再执行 `sbin/hadoop`，实际上等同于运行 `/usr/local/hadoop/sbin/hadoop`。我们可以将 Hadoop 命令的相关目录加入到 PATH 环境变量中，这样就可以直接通过 `start-dfs.sh` 开启 Hadoop，也可以直接通过 `hdfs` 访问 HDFS 的内容，方便平时的操作。

同样我们选择在 `~/.bashrc` 中进行设置：

- `vim ~/.bashrc`

在文件最前面加入如下单独一行：

```
export PATH=$PATH:/usr/local/hadoop/sbin:/usr/local/hadoop/bin
```

添加后执行如下命令使设置生效：

- `source ~/.bashrc`

下面开始搭建集群，即分布式模式：

环境：

本教程使用两个节点作为集群环境：一个做为 Master 节点，局域网 IP 为 192.168.10.45；另一个作为 Slave 节点，局域网 IP 为 192.168.10.46。

准备工作：

Hadoop 集群的安装配置大致为如下流程：

- (1) 选定一台机器作为 Master
- (2) 在 Master 节点上配置 hadoop 用户、安装 SSH server、安装 Java 环境
- (3) 在 Master 节点上安装 Hadoop，并完成配置
- (4) 在其他 Slave 节点上配置 hadoop 用户、安装 SSH server、安装 Java 环境

(5) 将 Master 节点上的/usr/local/hadoop 目录复制到其他 Slave 节点上

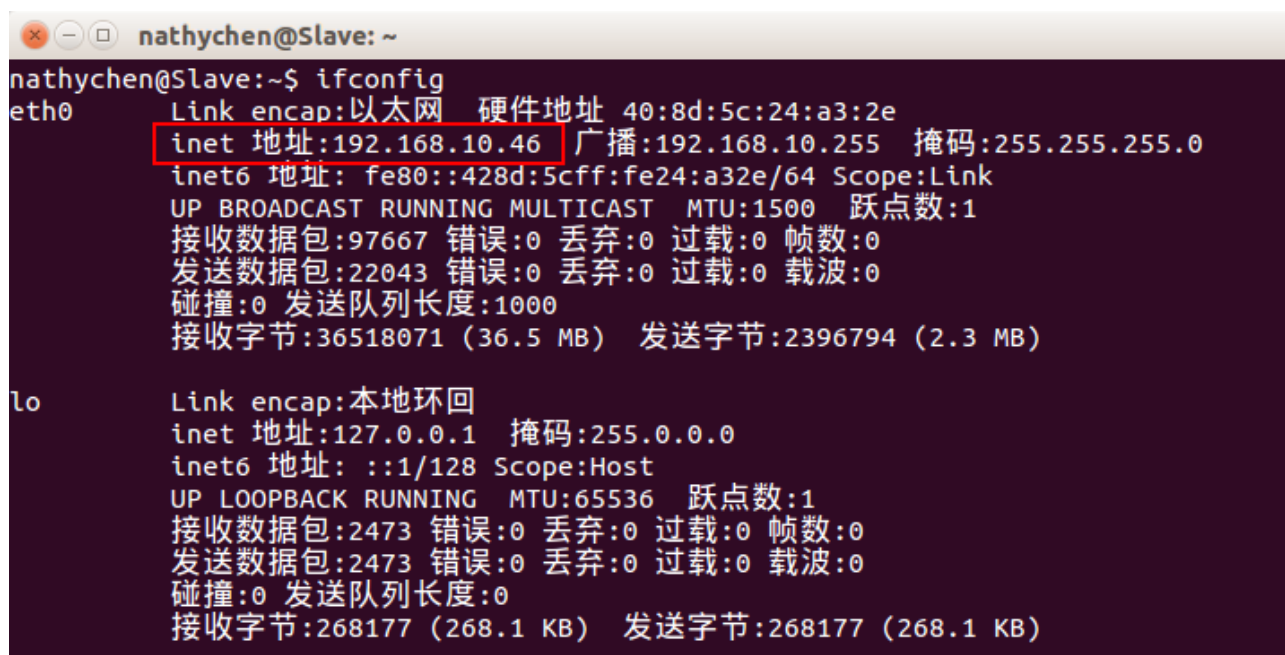
(6) 在 Master 节点上开启 Hadoop

配置 hadoop 用户、安装 SSH server、安装 Java 环境、安装 Hadoop 等在上文 1-10 中有详细介绍，在继续下一步配置前，请先完成上述流程的前 4 个步骤。

## 11、网络配置

假设集群所用的节点都位于同一个局域网。

Linux 中查看节点 IP 地址的命令为 ifconfig，即下图所示的 inet 地址：

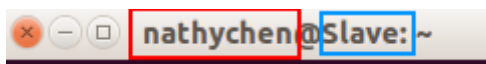


```
nathychen@Slave: ~  
nathychen@Slave:~$ ifconfig  
eth0      Link encap:以太网  硬件地址 40:8d:5c:24:a3:2e  
          inet 地址:192.168.10.46 广播:192.168.10.255 掩码:255.255.255.0  
          inet6 地址: fe80::428d:5cff:fe24:a32e/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  跃点数:1  
          接收数据包:97667 错误:0 丢弃:0 过载:0 帧数:0  
          发送数据包:22043 错误:0 丢弃:0 过载:0 载波:0  
          碰撞:0 发送队列长度:1000  
          接收字节:36518071 (36.5 MB)  发送字节:2396794 (2.3 MB)  
  
lo        Link encap:本地环回  
          inet 地址:127.0.0.1 掩码:255.0.0.0  
          inet6 地址: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:65536  跃点数:1  
          接收数据包:2473 错误:0 丢弃:0 过载:0 帧数:0  
          发送数据包:2473 错误:0 丢弃:0 过载:0 载波:0  
          碰撞:0 发送队列长度:0  
          接收字节:268177 (268.1 KB)  发送字节:268177 (268.1 KB)
```

首先在 Master 节点上完成准备工作，并关闭 Hadoop，再进行后续集群配置：

- /usr/local/hadoop/sbin/stop-dfs.sh

为了便于区分，可以修改各个节点的主机名。在终端顶端，红色为用户名，蓝色为主机名。



在 Ubuntu 中，我们在 Master 节点上执行如下命令修改主机名（即改为 Master，注意是区分大小写的）：

- sudo vim /etc/hostname

然后执行如下命令修改自己所用节点的 IP 映射：

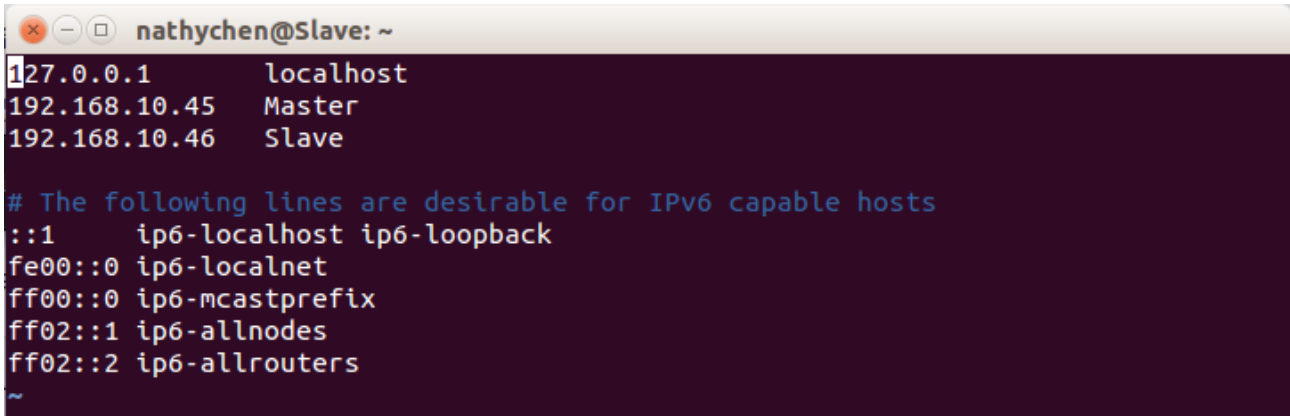
- sudo vim /etc/hosts

如本教程使用两个节点的名称与对应的 IP 关系如下：

192.168.10.45 Master

192.168.10.46 Slave

我们在 `/etc/hosts` 中将该映射关系填写上去即可，如下图所示（一般该文件中只有一个 `127.0.0.1`，其对应名为 `localhost`，如果有多余的应删除，特别是不能有“`127.0.0.1 Master`”这样的记录）：

A terminal window titled 'nathychen@Slave: ~' showing the contents of the /etc/hosts file. The first three lines are: 127.0.0.1 localhost, 192.168.10.45 Master, and 192.168.10.46 Slave. Below these are several lines for IPv6 addresses and their corresponding hostnames, including ::1, fe00::0, ff00::0, ff02::1, and ff02::2.

```
nathychen@Slave: ~
127.0.0.1      localhost
192.168.10.45  Master
192.168.10.46  Slave

# The following lines are desirable for IPv6 capable hosts
::1          ip6-localhost ip6-loopback
fe00::0      ip6-localnet
ff00::0      ip6-mcastprefix
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
~
```

修改完成后需要重启一下，重启后在终端中才会看到机器名的变化。

接下来的教程中请注意区分 **Master** 节点与 **Slave** 节点的操作。

注意：需要在所有节点上完成网络配置

如上面讲的是 **Master** 节点的配置，而在其他的 **Slave** 节点上，也要对 `/etc/hostname`（修改为 `Slave` 等）和 `/etc/hosts`（跟 **Master** 的配置一样）这两个文件进行修改！

配置好后需要在各个节点上执行如下命令，测试时否相互 **ping** 得通，如果 **ping** 不通，后面就无法顺利配置成功：

- `ping Master -c 3`      #只 ping3 次，否则要按 `ctrl+c` 中断
- `ping Slave -c 3`

**ping** 是可以从 **master** 到 **slave** 或 **slave** 到 **master** 双向的，如下图显示从 **slave** 到 **master** 的结果，**ping** 得通会显示 **time**：



```
nathychen@Slave: /usr/local/hadoop
nathychen@Slave:/usr/local/hadoop$ ping Master -c 3
PING Master (192.168.10.45) 56(84) bytes of data.
64 bytes from Master (192.168.10.45): icmp_seq=1 ttl=64 time=0.159 ms
64 bytes from Master (192.168.10.45): icmp_seq=2 ttl=64 time=0.151 ms
64 bytes from Master (192.168.10.45): icmp_seq=3 ttl=64 time=0.150 ms

--- Master ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.150/0.153/0.159/0.010 ms
nathychen@Slave:/usr/local/hadoop$
```

继续下一步配置前，请先完成所有节点的网络配置，修改过主机名的话需要重启才能生效。

## 12、SSH 无密码登录节点

这个操作是要让 Master 节点可以无密码 SSH 登录到各个 Slave 节点上。

首先生成 Master 节点的公钥，在 Master 节点的终端中执行（因为改过主机名，所以还需要删掉原有的再重新生成一次）：

- `cd ~/.ssh` #如果没有该目录，先执行一次 `ssh localhost`
- `rm ./id_rsa*` #删除之前生成的公钥（如果有）
- `ssh-keygen -t rsa` #一直回车就可以

让 Master 节点需能无密码 SSH 本机，在 Master 节点上执行：

- `cat ./id_rsa.pub >> ./authorized_keys`

完成后可执行 `ssh Master` 验证一下（可能需要输入 `yes`，成功后执行 `exit` 返回原来的终端）。

- `ssh Master`
- `exit`

接着在 Master 节点将上公钥传输到 Slave 节点：

- `scp ~/.ssh/id_rsa.pub nathychen@Slave:/home/nathychen/`

`scp` 是 `secure copy` 的简写，用于在 `linux` 下进行远程拷贝文件，类似于 `cp` 命令，不过 `cp` 只能在本机中拷贝。执行 `scp` 时会要求输入 Slave 上 `nathychen` 用户的密码，输入完成后会提示传输完毕，如下图所示：

```
nathychen@Master: ~/.ssh
nathychen@Master:~/.ssh$ scp ~/.ssh/id_rsa.pub nathychen@Slave:/home/nathychen/
nathychen@slave's password:
id_rsa.pub                                100% 398      0.4KB/s   00:00
nathychen@Master:~/.ssh$
```

接着在 Slave 节点上，将 ssh 公钥加入授权：

- `mkdir ~/.ssh` #如果不存在该文件夹需先创建，若已存在则忽略
- `cat ~/id_rsa.pub >> ~/.ssh/authorized_keys`
- `rm ~/id_rsa.pub` #用完就可以删掉了

如果有其他 Slave 节点，也要执行将 Master 公钥传输到 Slave 节点、在 Slave 节点上加入授权这两步。

这样，在 Master 节点上就可以无密码 SSH 到各个 Slave 节点了，可在 Master 节点上执行如下命令进行检验，如下图所示：

- `ssh Slave`

```
nathychen@Slave: ~
nathychen@Master:~/.ssh$ ssh Slave
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 4.2.0-34-generic x86_64)

* Documentation:  https://help.ubuntu.com/

Last login: Fri Apr  1 13:58:53 2016 from slave
nathychen@Slave:~$
```

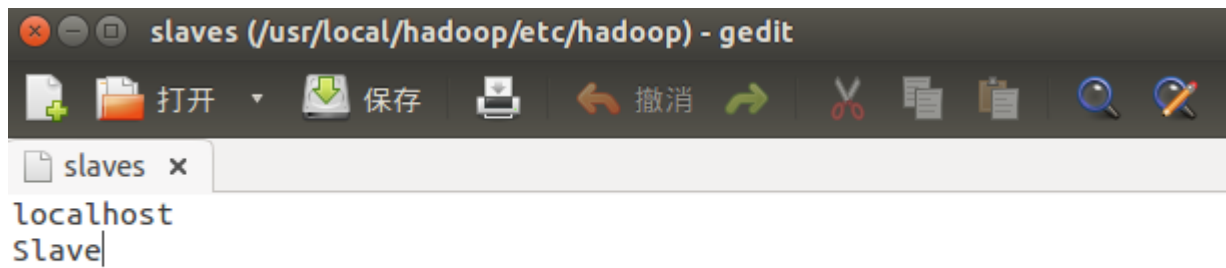
注意上图是在 Master 终端上执行的 ssh，ssh 登录后，终端标题以及命令符变为 Slave，此时执行的命令等同于在 Slave 节点上执行，可执行 `exit` 退回到原来的 Master 终端。

## 13、配置集群/分布式环境

集群/分布式模式需要修改 `/usr/local/hadoop/etc/hadoop` 中的 5 个配置文件，这里仅设置了正常启动所必需的设置项：`slaves`、`core-site.xml`、`hdfs-site.xml`、`mapred-site.xml`、`yarn-site.xml`。

13.1、文件 `slaves`，将作为 `DataNode` 的主机名写入该文件，每行一个，默认为 `localhost`，所以在伪分布式配置时，节点既作为 `NameNode` 也作为 `DataNode`。分布式配置可以保留 `localhost`，也可以删掉，让 Master 节点仅作为 `NameNode` 使用。

本教程让 Master 节点既作为 `NameNode` 也作为 `DataNode`，因此设置如下：



13.2、文件 `core-site.xml` 改为下面的配置:

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

13.3、文件 `hdfs-site.xml`, `dfs.replication` 一般设为 3,但我们只有一个 Slave 节点,所以 `dfs.replication` 的值还是设为 1:

```
<configuration>
  <property>
    <name>dfs.namenode.secondary.http-address</name>
    <value>Master:50090</value>
  </property>
  <property>
    <name>ds.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/usr/local/hadoop/tmp/dfs/name</value>
  </property>
```

```
<property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/usr/local/hadoop/tmp/dfs/data</value>
</property>
</configuration>
```

13.4、文件 `mapred-site.xml`（可能需要先重命名，默认文件名为 `mapred-site.xml.template`），然后配置修改如下：

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>mapreduce.jobhistory.address</name>
        <value>Master:10020</value>
    </property>
    <property>
        <name>mapreduce.jobhistory.webapp.address</name>
        <value>Master:19888</value>
    </property>
</configuration>
```

13.5、文件 `yarn-site.xml`：

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

配置好后，将 **Master** 上的 `/usr/local/Hadoop` 文件夹复制到各个节点上。因为之前有跑过伪分布式模式，建议在切换到集群模式前先删除之前的临时文件。在 **Master** 节点上执行：

- `cd /usr/local`
- `sudo rm -r ./hadoop/tmp` #删除 Hadoop 临时文件
- `sudo rm -r ./hadoop/logs/*` #删除日志文件
- `tar -zcf ~/hadoop.master.tar.gz ./hadoop` #先压缩再复制
- `cd ~`
- `scp ./hadoop.master.tar.gz Slave:/home/nathychen`

在 **Slave** 节点上执行：

- `sudo rm -r /usr/local/hadoop` #删掉旧的（如果存在）
- `sudo tar -zxf ~/hadoop.master.tar.gz -C /usr/local`
- `sudo chown -R hadoop /usr/local/hadoop`

同样，如果有其他 **Slave** 节点，也要执行将 `hadoop.master.tar.gz` 传输到 **Slave** 节点、在 **Slave** 节点解压文件的操作。

首次启动需要先在 **Master** 节点执行 **NameNode** 的格式化：

- `hdfs namenode -format` #首次运行需要执行初始化，之后不需要

接着可以启动 **hadoop** 了，启动需要在 **Master** 节点上进行：

- `start-dfs.sh`
- `start-yarn.sh`
- `mr-jobhistory-daemon.sh start historyserver`

通过命令 `jps` 可以查看各个节点所启动的进程。正确的话，在 **Master** 节点上可以看到 **NameNode**、**ResourceManager**、**SecondaryNameNode**、**JobHistoryServer** 进程，如下图所示：

```
nathychen@Master: ~  
nathychen@Master:~$ jps  
12425 Jps  
11370 NameNode  
11731 SecondaryNameNode  
11535 DataNode  
12383 JobHistoryServer  
12203 NodeManager  
11894 ResourceManager  
nathychen@Master:~$
```

在 Slave 节点可以看到 DataNode 和 NodeManager 进程，如下图所示：

```
nathychen@Slave: ~  
nathychen@Slave:~$ jps  
7115 NodeManager  
6983 DataNode  
7309 Jps  
nathychen@Slave:~$
```

缺少任意进程都表示出错。

另外还需要在 Master 节点上通过命令 `hdfs dfsadmin -report` 查看 DataNode 是否正常启动，如果 Live datanodes 不为 0，则说明集群启动成功。例如本例一共有 2 个 Datanodes：

- `hdfs dfsadmin -report`

```
nathychen@Master: ~  
nathychen@Master:~$ hdfs dfsadmin -report  
Configured Capacity: 735909183488 (685.37 GB)  
Present Capacity: 689360064512 (642.02 GB)  
DFS Remaining: 689359527936 (642.02 GB)  
DFS Used: 536576 (524 KB)  
DFS Used%: 0.00%  
Under replicated blocks: 0  
Blocks with corrupt replicas: 0  
Missing blocks: 0  
  
-----  
Live datanodes (2):  
  
Name: 192.168.10.45:50010 (Master)  
Hostname: Master  
Decommission Status : Normal  
Configured Capacity: 720295477248 (670.83 GB)  
DFS Used: 139264 (136 KB)  
Non DFS Used: 40482009088 (37.70 GB)  
DFS Remaining: 679813328896 (633.13 GB)  
DFS Used%: 0.00%  
DFS Remaining%: 94.38%  
Configured Cache Capacity: 0 (0 B)  
Cache Used: 0 (0 B)  
Cache Remaining: 0 (0 B)  
Cache Used%: 100.00%  
Cache Remaining%: 0.00%  
Xceivers: 1  
Last contact: Fri Apr 01 15:22:09 CST 2016  
  
Name: 192.168.10.46:50010 (Slave)  
Hostname: Slave  
Decommission Status : Normal  
Configured Capacity: 15613706240 (14.54 GB)  
DFS Used: 397312 (388 KB)  
Non DFS Used: 6067109888 (5.65 GB)  
DFS Remaining: 9546199040 (8.89 GB)  
DFS Used%: 0.00%  
DFS Remaining%: 61.14%  
Configured Cache Capacity: 0 (0 B)  
Cache Used: 0 (0 B)  
Cache Remaining: 0 (0 B)  
Cache Used%: 100.00%  
Cache Remaining%: 0.00%  
Xceivers: 1  
Last contact: Fri Apr 01 15:22:08 CST 2016  
  
nathychen@Master:~$
```

也可以通过 web 页面查看 DataNode 和 NameNode 的状态: <http://master:50070/>。

注意: 伪分布式、分布式配置切换时的注意事项

1、从分布式切换到伪分布式时, 不要忘记修改 `slaves` 配置文件;



2、在两者之间切换时，若遇到无法正常启动的情况，可以删除所涉及节点的临时文件家，这样虽然之前的数据会被删掉，但能保证集群正确启动。所以如果集群以前能启动，但后来启动不了，特别是 **DataNode** 无法启动，不妨试着删除所有节点（包括 **Slave** 节点）上的 `/usr/local/hadoop/tmp` 文件夹，再重新执行一次 `hdfs namenode -format`，再次启动试试。

#### 14、执行分布式实例

执行分布式实例过程与伪分布式模式一样，首先创建 **HDFS** 上的用户目录：

- `hdfs dfs -mkdir -p /user/hadoop`

将 `/usr/local/hadoop/etc/hadoop` 中的配置文件作为输入文件复制到分布式文件系统中：

- `hdfs dfs -mkdir input`
- `hdfs dfs -put /usr/local/hadoop/etc/hadoop/*.xml input`

通过查看 **DataNode** 的状态（占用大小有改变），输入文件确实复制到了 **DataNode** 中，如下图所示：

master:50070/dfshealth.html#tab-datanode

搜索

☆

📁

✓

↓

🏠

😊

☰

Hadoop

Overview

Datanodes

Snapshot

Startup Progress

Utilities

# Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
Slave (192.168.10.46:50010)	2	In Service	14.54 GB	388 KB	5.65 GB	8.89 GB	4	388 KB (0%)	0	2.6.0
Master (192.168.10.45:50010)	1	In Service	670.83 GB	136 KB	37.7 GB	633.13 GB	10	136 KB (0%)	0	2.6.0

Decomissioning

Node	Last contact	Under replicated blocks	Blocks with no live replicas	Under Replicated Blocks In files under construction
------	--------------	-------------------------	------------------------------	--

Hadoop, 2014.

Legacy UI

接着就可以运行 MapReduce 作业了：

- `hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar grep input output 'dfs[a-z.]+'`

运行输出信息与伪分布式类似，会显示 Job 的进度。

```
nathychen@Master: ~
16/04/01 15:42:59 INFO mapreduce.Job: The url to track the job: http://Master:8088/proxy/application_1459494982209_0001/
16/04/01 15:42:59 INFO mapreduce.Job: Running job: job_1459494982209_0001
16/04/01 15:43:04 INFO mapreduce.Job: Job job_1459494982209_0001 running in uber mode : false
16/04/01 15:43:04 INFO mapreduce.Job: map 0% reduce 0%
16/04/01 15:43:09 INFO mapreduce.Job: map 67% reduce 0%
16/04/01 15:43:11 INFO mapreduce.Job: map 89% reduce 0%
16/04/01 15:43:12 INFO mapreduce.Job: map 100% reduce 0%
16/04/01 15:43:14 INFO mapreduce.Job: map 100% reduce 100%
16/04/01 15:43:14 INFO mapreduce.Job: Job job_1459494982209_0001 completed successfully
```

同样可以通过 web 界面查看任务进度 <http://master:8088/cluster>，在 web 界面点击“Tracking UI” 这一列的 History 连接，可以看到任务的运行信息，如下图所示：

es	VCores	Active	Decommissioned	Lost	Unhealthy	Rebooted
l	Reserved	Nodes	Nodes	Nodes	Nodes	Nodes
	0	2	0	0	0	0
Search: <input type="text"/>						
FinishTime ▾	State ▾	FinalStatus ▾	Progress ▾	Tracking UI ▾		
Fri, 01 Apr 2016 07:43:13 GMT	FINISHED	SUCCEEDED	<div></div>	<a href="#">History</a>		
First Previous 1 Next Last						

执行完毕后的输出结果：

- `/usr/local/hadoop/bin/hdfs dfs -cat output/*`

```
nathychen@Master: ~
nathychen@Master:~$ /usr/local/hadoop/bin/hdfs dfs -cat output/*
1      dfsadmin
1      dfs.replication
1      dfs.namenode.secondary.http
1      dfs.namenode.name.dir
1      dfs.datanode.data.dir
nathychen@Master:~$
```

关闭 Hadoop 集群也是在 Master 节点上执行的：

- `stop-yarn.sh`
- `stop-dfs.sh`

- `mr-jobhistory-daemon.sh stop historyserver`

此外，同伪分布式一样，也可以不启动 YARN，但要记得改掉 `mapred-site.xml` 的文件名。

至此，你就掌握了 Hadoop 的集群搭建与基本使用了！！！！

END

附：

hadoop on linux 单机、伪分布安装方法：

<http://www.powerxing.com/install-hadoop/>

hadoop on linux 集群安装方法：<http://www.powerxing.com/install-hadoop-cluster/>

hadoop on windows 安装方法：

cygwin、java、ssh 安装方法：

<http://www.cnblogs.com/kinglau/archive/2013/08/20/3270160.html>

（hadoop 安装不要看此网站，会出错安装不上）

hadoop 配置、安装方法：<http://wiki.apache.org/hadoop/Hadoop2OnWindows>

（官网网址，好用！）