

# 基于 Docker 构建 Hadoop 平台

----- written by Mike.Wang 2022.07.15

## 0. 绪论

使用Docker搭建Hadoop技术平台，包括安装Docker、Java、Scala、Hadoop、Hbase、Spark。集群共有5台机器，主机名分别为 h01、h02、h03、h04、h05。其中 h01 为 master，其他的为 slave。

虚拟机配置：建议1盒2线程、8G内存、30G硬盘。最早配置4G内存，HBase和Spark运行异常。

- JDK 1.8
- Scala 2.11.12
- Hadoop 3.3.3
- Hbase 3.0.0
- Spark 3.3.0

## 1. Docker

### 1.1 Ubuntu 22.04 安装Docker

在 Ubuntu 下对 Docker 的操作都需要加上 `sudo`，如果已经是 root 账号了，则不需要。如果不加 `sudo`，Docker 相关命令会无法执行。

在 Ubuntu 下安装 Docker 的时候需在管理员的账号下操作。

```
mike@ubuntu2204:~$ wget -qO- https://get.docker.com/ | sh
```

安装完成之后，以 `sudo` 启动 Docker 服务。

```
mike@ubuntu2204:~$ sudo service docker start
```

显示 Docker 中所有正在运行的容器，由于 Docker 才安装，我们没有运行任何容器，所以显示结果如下所示。

```
mike@ubuntu2204:~$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
mike@ubuntu2204:~$
```

### 1.2 使用Docker

现在的 Docker 网络能够提供 DNS 解析功能，我们可以使用如下命令为接下来的 Hadoop 集群单独构建一个虚拟的网络。可以采用直通、桥接或macvlan方式，这里采用桥接模式，可以做到5台主机互联，并能访问宿主主机和网关，可以连接外网，便于在线下载程序资源。

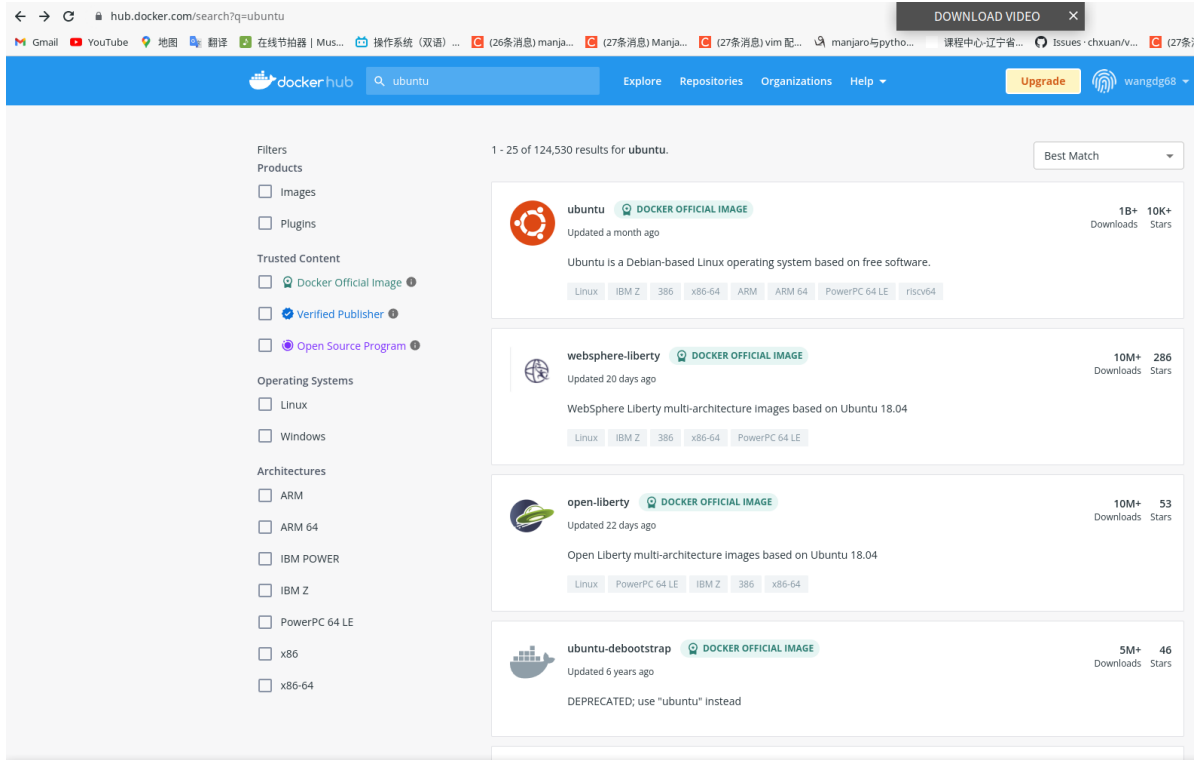
```
mike@ubuntu2204:~$ sudo docker network create --driver=bridge hadoop
```

以上命令创建了一个名为 `hadoop` 的虚拟桥接网络，该虚拟网络内部提供了自动的DNS解析服务。使用下面这个命令查看 Docker 中的网络，可以看到刚刚创建的名为 `hadoop` 的虚拟桥接网络。

```
mike@ubuntu2204:~$ sudo docker network ls
[sudo] password for mike:
NETWORK ID      NAME      DRIVER  SCOPE
3948edc3e8f3   bridge   bridge  local
337965dd9b1e   hadoop   bridge  local
cb8f2c453adc   host     host    local
fff4bd1c15ee   mynet    macvlan local
30e1132ad754   none     null    local
mike@ubuntu2204:~$
```

查找 ubuntu 容器

打开<https://hub.docker.com/>官网，搜索ubuntu，找到官方认证镜像，这里选取第一个



The screenshot shows the Docker Hub search results for 'ubuntu'. The search bar contains 'ubuntu' and the results are sorted by 'Best Match'. The first result is 'ubuntu', which is a Docker Official Image. It has 1B+ Downloads and 10K+ Stars. The description states: 'Ubuntu is a Debian-based Linux operating system based on free software.' Below the description, there are tags for 'Linux', 'IBM Z', '386', 'x86-64', 'ARM', 'ARM 64', 'PowerPC 64 LE', and 'riscv64'. The second result is 'websphere-liberty', which is also a Docker Official Image. It has 10M+ Downloads and 286 Stars. The description states: 'WebSphere Liberty multi-architecture images based on Ubuntu 18.04'. Below the description, there are tags for 'Linux', 'IBM Z', '386', 'x86-64', and 'PowerPC 64 LE'. The third result is 'open-liberty', which is also a Docker Official Image. It has 10M+ Downloads and 53 Stars. The description states: 'Open Liberty multi-architecture images based on Ubuntu 18.04'. Below the description, there are tags for 'Linux', 'PowerPC 64 LE', 'IBM Z', '386', and 'x86-64'. The fourth result is 'ubuntu-debootstrap', which is a Docker Official Image. It has 5M+ Downloads and 46 Stars. The description states: 'DEPRECATED, use "ubuntu" instead'. On the left side of the page, there are filters for Products, Trusted Content, Operating Systems, and Architectures.

点击第一个ubuntu，查找可选用的版本，这里选取22.04

## Quick reference

- **Maintained by:**  
Canonical and Tianon (Debian Developer)
- **Where to get help:**  
the Docker Community Forums, the Docker Community Slack, or Stack Overflow

## Supported tags and respective Dockerfile links

- 18.04 , bionic-20220531 , bionic
- 20.04 , focal-20220531 , focal
- 21.10 , impish-20220531 , impish
- 22.04 , jammy-20220531 , jammy , latest , rolling
- 22.10 , kinetic-20220602 , kinetic , devel
- 14.04 , trusty-20191217 , trusty
- 16.04 , xenial-20210804 , xenial

## Quick reference (cont.)

下载 ubuntu 22.04 版本的镜像文件

```
mike@ubuntu2204:~$ sudo docker pull ubuntu:22.04
```

查看已经下载的镜像

```
mike@ubuntu2204:~$ sudo docker images
[sudo] password for mike:
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
newuhadoop    latest   fe08b5527281  3 days ago    2.11GB
ubuntu        22.04    27941809078c  6 weeks ago   77.8MB
mike@ubuntu2204:~$
```

根据镜像启动一个容器，可以看出 shell 已经是容器的 shell 了，这里注意@后面的容器ID与上图镜像ID一致

```
mike@ubuntu2204:~$ sudo docker run -it ubuntu:22.04 /bin/bash
root@27941809078c:/#
```

输入 `exit` 可以退出容器，不过建议使用 `Ctrl + P + Q`，退出容器状态，但仍让容器处于后台运行状态。

```
mike@ubuntu2204:~$
```

查看本机上所有的容器

```

mike@ubuntu2204:~$ sudo docker ps -a
[sudo] password for mike:
CONTAINER ID   IMAGE          COMMAND          CREATED        STATUS        PORTS
NAMES
8016da5278ae   newuhadoop    "/bin/bash"     3 days ago    Up 2 days
h05
409c7e8aa2e9   newuhadoop    "/bin/bash"     3 days ago    Up 2 days
h04
0d8af236e1e7   newuhadoop    "/bin/bash"     3 days ago    Up 2 days
h03
72d62b7d4874   newuhadoop    "/bin/bash"     3 days ago    Up 2 days
h02
d4d3ca3bbb61   newuhadoop    "/bin/bash"     3 days ago    Up 2 days    0.0.0.0:8088-
>8088/tcp, :::8088->8088/tcp, 0.0.0.0:9870->9870/tcp, :::9870->9870/tcp   h01
mike@ubuntu2204:~$

```

此处会看到刚刚创建好的容器，并在后台运行。这里因为是后期制作的教程，为了节省内存，只保留了5台hadoop的容器，最原始的容器已经删除。

启动一个状态为退出的容器，最后一个参数为容器 ID

```
mike@ubuntu2204:~$ sudo docker start 27941809078c
```

进入一个容器

```
mike@ubuntu2204:~$ sudo docker attach 27941809078c
```

关闭一个容器

```
mike@ubuntu2204:~$ sudo docker stop 27941809078c
```

## 2. 安装集群

主要是安装JDK 1.8的环境，因为 Spark 要 Scala，Scala 要JDK 1.8，以及 Hadoop，以此来构建基础镜像。

### 2.1 安装Java 与 Scala

进入之前的 Ubuntu 容器

先更换 apt 的源

#### 2.1.1 修改 apt 源

备份源

```

root@27941809078c:/# cp /etc/apt/sources.list /etc/apt/sources_init.list
root@27941809078c:/#

```

先删除就源文件，这个时候没有 vim 工具..

```
root@27941809078c:/# rm /etc/apt/sources.list
```

复制以下命令，回车，即可一键切换到阿里云 ubuntu 22.04镜像：（此时已经是root权限，提示符为#）

```
bash -c "cat << EOF > /etc/apt/sources.list && apt update
deb http://mirrors.aliyun.com/ubuntu/ jammy main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ jammy main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ jammy-security main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ jammy-security main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ jammy-updates main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ jammy-updates main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ jammy-proposed main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ jammy-proposed main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ jammy-backports main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ jammy-backports main restricted
universe multiverse
EOF"
```

再使用 `apt update` / `apt upgrade` 来更新，`update`更列表，`upgrade`更新包

```
root@27941809078c:/# apt update
root@27941809078c:/# apt upgrade
```

## 2.1.2 安装 Java与 Scala

安装 jdk 1.8，直接输入命令

```
root@27941809078c:/# apt install openjdk-8-jdk
```

测试一下安装结果

```
root@27941809078c:/# java -version
openjdk version "1.8.0_312"
OpenJDK Runtime Environment (build 1.8.0_312-8u312-b07-0ubuntu1-b07)
OpenJDK 64-Bit Server VM (build 25.312-b07, mixed mode)
root@27941809078c:/#
```

接下来安装scala

```
root@27941809078c:/# apt install scala
```

测试一下安装结果

```
root@27941809078c:/# scala
Welcome to Scala 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_312).
Type in expressions for evaluation. Or try :help.

scala>
```

输入 :quit 退出scala

## 2.2 安装 Hadoop

- 在当前容器中将配置配好
- 导入出为镜像
- 以此镜像为基础创建五个容器，并赋予 hostname
- 进入 h01 容器，启动 Hadoop

### 2.2.1 安装 Vim 与 网络工具包

安装 vim，用来编辑文件

```
root@27941809078c:/# apt install vim
```

安装 net-tools、iputils-ping、iproute2 网络工具包，目的是为了使用 ping、ifconfig、ip、traceroute 等命令

```
root@27941809078c:/# apt install net-tools
root@27941809078c:/# apt install iputils-ping
root@27941809078c:/# apt install iproute2
```

### 2.2.2 安装 SSH

安装 SSH，并配置免密登录，由于后面的容器之间是由一个镜像启动的，就像同一个磨具出来的 5 把锁与钥匙，可以互相开锁。所以在当前容器里配置 SSH 自身免密登录就 OK 了。

安装 SSH 服务器端

```
root@27941809078c:/# apt install openssh-server
```

安装 SSH 的客户端

```
root@27941809078c:/# apt install openssh-client
```

进入当前用户的用户根目录

```
root@27941809078c:/# cd ~
root@27941809078c:~#
```

生成密钥，不用输入，一直回车就行，生成的密钥在当前用户根目录下的 `.ssh` 文件夹中。以 `.` 开头的文件与文件夹 `ls` 是隐藏的，需要 `ls -al` 才能查看。

```
root@27941809078c:~# ssh-keygen -t rsa -P ""
```

将公钥追加到 `authorized_keys` 文件中

```
root@27941809078c:~# cat .ssh/id_rsa.pub >> .ssh/authorized_keys
root@27941809078c:~#
```

## 启动 SSH 服务

```
root@27941809078c:~# service ssh start
* Starting OpenBSD Secure Shell server sshd

    [ OK ]
root@27941809078c:~#
```

## 免密登录自己

```
root@27941809078c:~# ssh 127.0.0.1
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-41-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sun Jul 17 08:26:15 2022 from 172.18.0.1
* Starting OpenBSD Secure Shell server sshd
root@27941809078c:~#
```

修改 `.bashrc` 文件，启动 shell 的时候，自动启动 SSH 服务

用 vim 打开 `.bashrc` 文件

```
root@27941809078c:~# vim ~/.bashrc
```

按一下 `i` 键，使得 vim 进入插入模式，此时终端的左下角会显示为 `-- INSERT --`，将光标移动到最后一行，添加一行（Caps + g 可直接到最后一行）

```
service ssh start
```

添加完的结果为，只显示最后几行

```
if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
#    . /etc/bash_completion
#fi
service ssh start
```

按一下 `Esc` 键，使得 vim 退出插入模式

再输入英文模式下的冒号 `:`，此时终端的左下方会有一个冒号 `:` 显示出来

再输入三个字符 `wq!`，这是一个组合命令

- `w` 是保存的意思
- `q` 是退出的意思
- `!` 是强制的意思

再输入回车，退出 vim。

此时，SSH 免密登录已经完全配置好。

## 2.2.3 安装 Hadoop

下载 Hadoop 的安装文件

```
root@27941809078c:~# wget https://mirrors.aliyun.com/apache/hadoop/common/hadoop-3.3.3/hadoop-3.3.3.tar.gz
```

解压到 `/usr/local` 目录下面并重命名文件夹

```
root@27941809078c:~# tar -zxvf hadoop-3.3.3.tar.gz -C /usr/local/
root@27941809078c:~# cd /usr/local/
root@27941809078c:/usr/local# mv hadoop-3.3.3 hadoop
root@27941809078c:/usr/local#
```

修改 `/etc/profile` 文件，添加一下环境变量到文件中

先用 vim 打开 `/etc/profile`

```
vim /etc/profile
```

追加以下内容

JAVA\_HOME 为 JDK 安装路径，使用 apt 安装就是这个，用 `update-alternatives --config java` 可查看

```
#java
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
export PATH=${JAVA_HOME}/bin:$PATH
#hadoop
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_CONF_DIR=$HADOOP_HOME
export HADOOP_LIBEXEC_DIR=$HADOOP_HOME/libexec
export JAVA_LIBRARY_PATH=$HADOOP_HOME/lib/native:$JAVA_LIBRARY_PATH
export HADOOP_CONF_DIR=$HADOOP_PREFIX/etc/hadoop
```



```
export HDFS_DATANODE_USER=root
export HDFS_DATANODE_SECURE_USER=root
export HDFS_SECONDARYNAMENODE_USER=root
export HDFS_NAMENODE_USER=root
export YARN_RESOURCEMANAGER_USER=root
export YARN_NODEMANAGER_USER=root
```

使环境变量生效

```
root@27941809078c:/usr/local# source /etc/profile
root@27941809078c:/usr/local#
```

在目录 `/usr/local/hadoop/etc/hadoop` 下，修改6个重要配置文件

修改 `hadoop-env.sh` 文件，在文件末尾添加一下信息

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HDFS_NAMENODE_USER=root
export HDFS_DATANODE_USER=root
export HDFS_SECONDARYNAMENODE_USER=root
export YARN_RESOURCEMANAGER_USER=root
export YARN_NODEMANAGER_USER=root
```

修改 `core-site.xml`，修改为

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://h01:9000</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/hadoop3/hadoop/tmp</value>
  </property>
</configuration>
```

修改 `hdfs-site.xml`，修改为

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>2</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/home/hadoop3/hadoop/hdfs/name</value>
  </property>
  <property>
    <name>dfs.namenode.data.dir</name>
    <value>/home/hadoop3/hadoop/hdfs/data</value>
  </property>
</configuration>
```

修改 `mapred-site.xml`，修改为

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>mapreduce.application.classpath</name>
    <value>
      /usr/local/hadoop/etc/hadoop,
      /usr/local/hadoop/share/hadoop/common/*,
      /usr/local/hadoop/share/hadoop/common/lib/*,
      /usr/local/hadoop/share/hadoop/hdfs/*,
      /usr/local/hadoop/share/hadoop/hdfs/lib/*,
      /usr/local/hadoop/share/hadoop/mapreduce/*,
      /usr/local/hadoop/share/hadoop/mapreduce/lib/*,
      /usr/local/hadoop/share/hadoop/yarn/*,
      /usr/local/hadoop/share/hadoop/yarn/lib/*
    </value>
  </property>
</configuration>
```

修改 yarn-site.xml, 修改为

```
<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>h01</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

修改 worker 为

```
h01
h02
h03
h04
h05
```

此时, hadoop已经配置好了

## 2.2.4 在 Docker 中启动集群

先将当前容器导出为镜像, 并查看当前镜像。使用 `ctrl + p + q`, 退出容器, 回到宿主机

```
mike@ubuntu2204:~$ sudo docker commit -m "hadoop" -a "hadoop" 27941809078c
newuhadoop
sha256:648d8e082a231919faeaa14e09f5ce369b20879544576c03ef94074daf978823
mike@ubuntu2204:~$ sudo docker images
[sudo] password for mike:
REPOSITORY   TAG           IMAGE ID       CREATED        SIZE
newuhadoop   latest       fe08b5527281  4 days ago    2.11GB
ubuntu       22.04        27941809078c  6 weeks ago   77.8MB
mike@ubuntu2204:~$
```

启动 5 个终端，分别执行这几个命令

第一条命令启动的是 `h01` 是做 master 节点的，所以暴露了端口，以供访问 web 页面

```
mike@ubuntu2204:~$ sudo docker run -it --network hadoop -h "h01" --name "h01" -p
9870:9870 -p 8088:8088 newuhadoop /bin/bash
* Starting OpenBSD Secure Shell server sshd

[ OK ]
root@h01:/#
```

其余的四条命令就是几乎一样的了，注意：启动容器后，使用 `ctrl + p + q` 退回到宿主机，之后再启动下一个容器

```
mike@ubuntu2204:~$ sudo docker run -it --network hadoop -h "h02" --name "h02"
newuhadoop /bin/bash
[sudo] password for mike:
* Starting OpenBSD Secure Shell server sshd

[ OK ]
root@h02:/#

mike@ubuntu2204:~$ sudo docker run -it --network hadoop -h "h03" --name "h03"
newuhadoop /bin/bash
[sudo] password for mike:
* Starting OpenBSD Secure Shell server sshd

[ OK ]
root@h03:/#

mike@ubuntu2204:~$ sudo docker run -it --network hadoop -h "h04" --name "h04"
newuhadoop /bin/bash
[sudo] password for mike:
* Starting OpenBSD Secure Shell server sshd

[ OK ]
root@h04:/#

mike@ubuntu2204:~$ sudo docker run -it --network hadoop -h "h05" --name "h05"
newuhadoop /bin/bash
[sudo] password for mike:
* Starting OpenBSD Secure Shell server sshd

[ OK ]
root@h05:/#
```

接下来，在 `h01` 主机中，启动 Hadoop 集群

先进行格式化操作，不格式化操作，hdfs会起不来

```
root@h01:/usr/local/hadoop/bin# ./hadoop namenode -format
```

进入 hadoop 的 sbin 目录

```
root@h01:/# cd /usr/local/hadoop/sbin/  
root@h01:/usr/local/hadoop/sbin#
```

启动 hadoop

```
root@h01:/usr/local/hadoop/sbin# ./start-all.sh  
Starting namenodes on [h01]  
h01: Warning: Permanently added 'h01,172.18.0.2' (ECDSA) to the list of known  
hosts.  
Starting datanodes  
h05: Warning: Permanently added 'h05,172.18.0.6' (ECDSA) to the list of known  
hosts.  
h02: Warning: Permanently added 'h02,172.18.0.3' (ECDSA) to the list of known  
hosts.  
h03: Warning: Permanently added 'h03,172.18.0.4' (ECDSA) to the list of known  
hosts.  
h04: Warning: Permanently added 'h04,172.18.0.5' (ECDSA) to the list of known  
hosts.  
h03: WARNING: /usr/local/hadoop/logs does not exist. Creating.  
h05: WARNING: /usr/local/hadoop/logs does not exist. Creating.  
h02: WARNING: /usr/local/hadoop/logs does not exist. Creating.  
h04: WARNING: /usr/local/hadoop/logs does not exist. Creating.  
Starting secondary namenodes [h01]  
Starting resourcemanager  
Starting nodemanagers  
root@h01:/usr/local/hadoop/sbin#
```

使用 `jps` 查看集群启动状态（这个状态不是固定不变的，随着应用不同而不同，但至少应该有3个）

```
root@h01:~# jps  
10017 HRegionServer  
10609 Master  
9778 HQuorumPeer  
8245 SecondaryNameNode  
8087 DataNode  
9881 HMaster  
41081 Jps  
10684 Worker  
7965 NameNode  
8477 ResourceManager  
8591 NodeManager  
root@h01:~#
```

使用命令 `./hdfs dfsadmin -report` 可查看分布式文件系统的状态

```
root@h01:/usr/local/hadoop/bin# ./hdfs dfsadmin -report  
Configured Capacity: 90810798080 (84.57 GB)
```

Present Capacity: 24106247929 (22.45 GB)  
DFS Remaining: 24097781497 (22.44 GB)  
DFS Used: 8466432 (8.07 MB)  
DFS Used%: 0.04%  
Replicated Blocks:  
    Under replicated blocks: 0  
    Blocks with corrupt replicas: 0  
    Missing blocks: 0  
    Missing blocks (with replication factor 1): 0  
    Low redundancy blocks with highest priority to recover: 0  
    Pending deletion blocks: 0  
Erasure Coded Block Groups:  
    Low redundancy block groups: 0  
    Block groups with corrupt internal blocks: 0  
    Missing block groups: 0  
    Low redundancy blocks with highest priority to recover: 0  
    Pending deletion blocks: 0

-----  
Live datanodes (5):

Name: 172.18.0.2:9866 (h01)  
Hostname: h01  
Decommission Status : Normal  
Configured Capacity: 18162159616 (16.91 GB)  
DFS Used: 2875392 (2.74 MB)  
Non DFS Used: 11887669248 (11.07 GB)  
DFS Remaining: 4712182185 (4.39 GB)  
DFS Used%: 0.02%  
DFS Remaining%: 25.95%  
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: 10  
Last contact: Wed Jul 20 04:55:01 GMT 2022  
Last Block Report: Tue Jul 19 23:36:54 GMT 2022  
Num of Blocks: 293

Name: 172.18.0.3:9866 (h02.hadoop)  
Hostname: h02  
Decommission Status : Normal  
Configured Capacity: 18162159616 (16.91 GB)  
DFS Used: 1396736 (1.33 MB)  
Non DFS Used: 11889147904 (11.07 GB)  
DFS Remaining: 4846399828 (4.51 GB)  
DFS Used%: 0.01%  
DFS Remaining%: 26.68%  
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: 8  
Last contact: Wed Jul 20 04:55:01 GMT 2022  
Last Block Report: Tue Jul 19 23:51:39 GMT 2022

Num of Blocks: 153

Name: 172.18.0.4:9866 (h03.hadoop)  
Hostname: h03  
Decommission Status : Normal  
Configured Capacity: 18162159616 (16.91 GB)  
DFS Used: 1323008 (1.26 MB)  
Non DFS Used: 11889221632 (11.07 GB)  
DFS Remaining: 5114835114 (4.76 GB)  
DFS Used%: 0.01%  
DFS Remaining%: 28.16%  
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: 4  
Last contact: Wed Jul 20 04:55:01 GMT 2022  
Last Block Report: Wed Jul 20 02:14:39 GMT 2022  
Num of Blocks: 151

Name: 172.18.0.5:9866 (h04.hadoop)  
Hostname: h04  
Decommission Status : Normal  
Configured Capacity: 18162159616 (16.91 GB)  
DFS Used: 1527808 (1.46 MB)  
Non DFS Used: 11889016832 (11.07 GB)  
DFS Remaining: 4712182185 (4.39 GB)  
DFS Used%: 0.01%  
DFS Remaining%: 25.95%  
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: 10  
Last contact: Wed Jul 20 04:55:01 GMT 2022  
Last Block Report: Wed Jul 20 00:42:09 GMT 2022  
Num of Blocks: 134

Name: 172.18.0.6:9866 (h05.hadoop)  
Hostname: h05  
Decommission Status : Normal  
Configured Capacity: 18162159616 (16.91 GB)  
DFS Used: 1343488 (1.28 MB)  
Non DFS Used: 11889201152 (11.07 GB)  
DFS Remaining: 4712182185 (4.39 GB)  
DFS Used%: 0.01%  
DFS Remaining%: 25.95%  
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: 10

```
Last contact: Wed Jul 20 04:55:01 GMT 2022
Last Block Report: Wed Jul 20 02:36:21 GMT 2022
Num of Blocks: 149
```

```
root@h01:/usr/local/hadoop/bin#
```

访问宿主机的 8088 与 9870 端口就可以看到监控信息了

至此，Hadoop 集群已经构建好了

## 2.2.5 运行内置WordCount例子

把 license 作为需要统计的文件

```
root@h01:/usr/local/hadoop# cat LICENSE.txt > file1.txt
root@h01:/usr/local/hadoop# ls
```

在 HDFS 中创建 input 文件夹

```
root@h01:/usr/local/hadoop/bin# ./hadoop fs -mkdir /input
root@h01:/usr/local/hadoop/bin#
```

上传 file1.txt 文件到 HDFS 中

```
root@h01:/usr/local/hadoop/bin# ./hadoop fs -put ../file1.txt /input
root@h01:/usr/local/hadoop/bin#
```

查看 HDFS 中 input 文件夹里的内容

```
root@h01:/usr/local/hadoop/bin# ./hadoop fs -ls /input
Found 1 items
-rw-r--r--  2 root supergroup      15217 2022-07-17 08:50 /input/file1.txt
root@h01:/usr/local/hadoop/bin#
```

运行wordcount 例子程序

```
root@h01:/usr/local/hadoop/bin# ./hadoop jar ../share/hadoop/mapreduce/hadoop-
mapreduce-examples-3.3.3.jar wordcount /input /output
```

输出如下:

```
root@h01:/usr/local/hadoop/bin# ./hadoop jar ../share/hadoop/mapreduce/hadoop-
mapreduce-examples-3.3.3.jar wordcount /input /output
2022-07-20 05:12:38,394 INFO client.DefaultNoHARMAFailoverProxyProvider:
Connecting to ResourceManager at h01/172.18.0.2:8032
2022-07-20 05:12:38,816 INFO mapreduce.JobResourceUploader: Disabling Erasure
Coding for path: /tmp/hadoop-yarn/staging/root/.staging/job_1658047711391_0002
2022-07-20 05:12:39,076 INFO input.FileInputFormat: Total input files to process
: 1
2022-07-20 05:12:39,198 INFO mapreduce.JobSubmitter: number of splits:1
2022-07-20 05:12:39,399 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1658047711391_0002
2022-07-20 05:12:39,399 INFO mapreduce.JobSubmitter: Executing with tokens: []
2022-07-20 05:12:39,674 INFO conf.Configuration: resource-types.xml not found
2022-07-20 05:12:39,674 INFO resource.ResourceUtils: Unable to find 'resource-
types.xml'.
2022-07-20 05:12:39,836 INFO impl.YarnClientImpl: Submitted application
application_1658047711391_0002
2022-07-20 05:12:39,880 INFO mapreduce.Job: The url to track the job:
http://h01:8088/proxy/application_1658047711391_0002/
2022-07-20 05:12:39,882 INFO mapreduce.Job: Running job: job_1658047711391_0002
2022-07-20 05:12:49,171 INFO mapreduce.Job: Job job_1658047711391_0002 running in
uber mode : false
2022-07-20 05:12:49,174 INFO mapreduce.Job:  map 0% reduce 0%
2022-07-20 05:12:54,285 INFO mapreduce.Job:  map 100% reduce 0%
2022-07-20 05:13:01,356 INFO mapreduce.Job:  map 100% reduce 100%
2022-07-20 05:13:02,391 INFO mapreduce.Job: Job job_1658047711391_0002 completed
successfully
2022-07-20 05:13:02,524 INFO mapreduce.Job: Counters: 54
    File System Counters
        FILE: Number of bytes read=12507
        FILE: Number of bytes written=577413
        FILE: Number of read operations=0
```



FILE: Number of large read operations=0  
FILE: Number of write operations=0  
HDFS: Number of bytes read=15313  
HDFS: Number of bytes written=9894  
HDFS: Number of read operations=8  
HDFS: Number of large read operations=0  
HDFS: Number of write operations=2  
HDFS: Number of bytes read erasure-coded=0

#### Job Counters

Launched map tasks=1  
Launched reduce tasks=1  
Data-local map tasks=1  
Total time spent by all maps in occupied slots (ms)=3141  
Total time spent by all reduces in occupied slots (ms)=3811  
Total time spent by all map tasks (ms)=3141  
Total time spent by all reduce tasks (ms)=3811  
Total vcore-milliseconds taken by all map tasks=3141  
Total vcore-milliseconds taken by all reduce tasks=3811  
Total megabyte-milliseconds taken by all map tasks=3216384  
Total megabyte-milliseconds taken by all reduce tasks=3902464

#### Map-Reduce Framework

Map input records=270  
Map output records=1672  
Map output bytes=20756  
Map output materialized bytes=12507  
Input split bytes=96  
Combine input records=1672  
Combine output records=657  
Reduce input groups=657  
Reduce shuffle bytes=12507  
Reduce input records=657  
Reduce output records=657  
Spilled Records=1314  
Shuffled Maps =1  
Failed Shuffles=0  
Merged Map outputs=1  
GC time elapsed (ms)=126  
CPU time spent (ms)=1110  
Physical memory (bytes) snapshot=474148864  
Virtual memory (bytes) snapshot=5063700480  
Total committed heap usage (bytes)=450887680  
Peak Map Physical memory (bytes)=288309248  
Peak Map Virtual memory (bytes)=2528395264  
Peak Reduce Physical memory (bytes)=185839616  
Peak Reduce Virtual memory (bytes)=2535305216

#### Shuffle Errors

BAD\_ID=0  
CONNECTION=0  
IO\_ERROR=0  
WRONG\_LENGTH=0  
WRONG\_MAP=0  
WRONG\_REDUCE=0

#### File Input Format Counters

Bytes Read=15217

#### File Output Format Counters

Bytes Written=9894

root@h01:/usr/local/hadoop/bin#

查看 HDFS 中的 /output 文件夹的内容

```
root@h01:/usr/local/hadoop/bin# ./hadoop fs -ls /output
Found 2 items
-rw-r--r--  2 root supergroup          0 2022-07-20 05:13 /output/_SUCCESS
-rw-r--r--  2 root supergroup    9894 2022-07-20 05:13 /output/part-r-00000
root@h01:/usr/local/hadoop/bin#
```

查看 `part-r-00000` 文件的内容

```
root@h01:/usr/local/hadoop/bin# ./hadoop fs -cat /output/part-r-00000
```

至此，hadoop部分已经结束

## 2.3 安装 Hbase

在 Hadoop 集群的基础上安装 Hbase

下载 Hbase 3.0.0

```
root@h01:~# wget https://mirrors.tuna.tsinghua.edu.cn/apache/hbase/3.0.0-alpha-3/hbase-3.0.0-alpha-3-bin.tar.gz
```

解压到 `/usr/local` 目录下面

```
root@h01:~# tar -zxvf hbase-3.0.0-bin.tar.gz -C /usr/local/
```

修改 `/etc/profile` 环境变量文件，添加 Hbase 的环境变量，追加下述代码

```
export HBASE_HOME=/usr/local/hbase-3.0.0
export PATH=$PATH:$HBASE_HOME/bin
```

使环境变量配置文件生效

```
root@h01:/usr/local# source /etc/profile
root@h01:/usr/local#
```

使用 `ssh h02` 可进入h02容器，修改profile文件如上。依次修改h03、h04、h05

即是每个容器都要在 `/etc/profile` 文件后追加那两行环境变量

在目录 `/usr/local/hbase-3.0.0/conf` 修改配置

修改 `hbase-env.sh`，追加

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HBASE_MANAGES_ZK=true
```

修改 `hbase-site.xml` 为

```
<configuration>
  <property>
    <name>hbase.rootdir</name>
```

```
        <value>hdfs://h01:9000/hbase</value>
    </property>
</property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
</property>
</property>
    <name>hbase.master</name>
    <value>h01:60000</value>
</property>
</property>
    <name>hbase.zookeeper.quorum</name>
    <value>h01,h02,h03,h04,h05</value>
</property>
</property>
    <name>hbase.zookeeper.property.dataDir</name>
    <value>/home/hadoop/zoodata</value>
</property>
</configuration>
```

修改 `regionservers` 文件为

```
h01
h02
h03
h04
h05
```

使用 `scp` 命令将配置好的 Hbase 复制到其他 4 个容器中

```
root@h01:~# scp -r /usr/local/hbase-3.0.0 root@h02:/usr/local/
root@h01:~# scp -r /usr/local/hbase-3.0.0 root@h03:/usr/local/
root@h01:~# scp -r /usr/local/hbase-3.0.0 root@h04:/usr/local/
root@h01:~# scp -r /usr/local/hbase-3.0.0 root@h05:/usr/local/
```

启动 Hbase

```
root@h01:/usr/local/hbase-3.0.0/bin# ./start-hbase.sh
h04: running zookeeper, logging to /usr/local/hbase-3.0.0/bin/./logs/hbase-root-zookeeper-h04.out
h02: running zookeeper, logging to /usr/local/hbase-3.0.0/bin/./logs/hbase-root-zookeeper-h02.out
h03: running zookeeper, logging to /usr/local/hbase-3.0.0/bin/./logs/hbase-root-zookeeper-h03.out
h05: running zookeeper, logging to /usr/local/hbase-3.0.0/bin/./logs/hbase-root-zookeeper-h05.out
h01: running zookeeper, logging to /usr/local/hbase-3.0.0/bin/./logs/hbase-root-zookeeper-h01.out
running master, logging to /usr/local/hbase-3.0.0/bin/./logs/hbase--master-h01.out
h05: running regionserver, logging to /usr/local/hbase-3.0.0/bin/./logs/hbase-root-regionserver-h05.out
h01: running regionserver, logging to /usr/local/hbase-3.0.0/bin/./logs/hbase-root-regionserver-h01.out
h04: running regionserver, logging to /usr/local/hbase-3.0.0/bin/./logs/hbase-root-regionserver-h04.out
h03: running regionserver, logging to /usr/local/hbase-3.0.0/bin/./logs/hbase-root-regionserver-h03.out
h02: running regionserver, logging to /usr/local/hbase-3.0.0/bin/./logs/hbase-root-regionserver-h02.out
root@h01:/usr/local/hbase-3.0.0/bin#
```

打开 Hbase 的 shell

```
root@h01:/usr/local/hbase-3.0.0/bin# ./hbase shell
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/book.html#shell
Version 3.0.0-alpha-3, rb3657484850f9fa9679f2186bf53e7df768f21c7, Wed Jun 15
07:56:54 UTC 2022
Took 0.0017 seconds

hbase:001:0>
```

hbase测试

创建表member

```
hbase:006:0> create 'member','id','address','info'
Created table member
Took 0.6838 seconds

=> Hbase::Table - member
hbase:007:0>
```

添加数据, 并查看表中数据

```
hbase:007:0> put 'member', 'debugo','id','11'
Took 0.1258 seconds

hbase:008:0> put 'member', 'debugo','info:age','27'
```

```

Took 0.0108 seconds

hbase:009:0> count 'member'
1 row(s)
Took 0.0499 seconds

=> 1
hbase:010:0> scan 'member'
ROW                                COLUMN+CELL

  debugo                            column=id:, timestamp=2022-07-
20T05:37:58.720, value=11

  debugo                            column=info:age, timestamp=2022-07-
20T05:38:11.302, value=27
1 row(s)
Took 0.0384 seconds

hbase:011:0>

```

## 2.4 安装 Spark

在 Hadoop 的基础上安装 Spark

下载 Spark 3.3.0

```

root@h01:~# wget https://mirrors.tuna.tsinghua.edu.cn/apache/spark/spark-
3.3.0/spark-3.3.0-bin-hadoop3.tgz

```

解压到 `/usr/local` 目录下

```

root@h01:~# tar -zxvf spark-3.3.0-bin-hadoop3.tgz -C /usr/local/

```

修改文件夹的名字

```

root@h01:~# cd /usr/local/
root@h01:/usr/local# mv spark-3.3.0-bin-hadoop3 spark-3.3.0

```

修改 `/etc/profile` 环境变量文件，添加 Hbase 的环境变量，追加下述代码

```

export SPARK_HOME=/usr/local/spark-3.3.0
export PATH=$PATH:$SPARK_HOME/bin

```

使环境变量配置文件生效

```

root@h01:/usr/local# source /etc/profile
root@h01:/usr/local#

```

使用 `ssh h02` 可进入其他四个容器，依次修改。

即是每个容器都要在 `/etc/profile` 文件后追加那两行环境变量

在目录 `/usr/local/spark-3.3.0/conf` 修改配置

修改文件名

```
root@h01:/usr/local/spark-3.3.0/conf# mv spark-env.sh.template spark-env.sh
root@h01:/usr/local/spark-3.3.0/conf#
```

修改 spark-env.sh, 追加

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
export SCALA_HOME=/usr/share/scala

export SPARK_MASTER_HOST=h01
export SPARK_MASTER_IP=h01
export SPARK_WORKER_MEMORY=4g
```

修改文件名

```
root@h01:/usr/local/spark-3.3.0/conf# mv slaves.template slaves
root@h01:/usr/local/spark-3.3.0/conf#
```

修改 slaves 如下

```
h01
h02
h03
h04
h05
```

使用 `scp` 命令将配置好的 Hbase 复制到其他 4 个容器中

```
root@h01:/usr/local# scp -r /usr/local/spark-3.3.0 root@h02:/usr/local/
root@h01:/usr/local# scp -r /usr/local/spark-3.3.0 root@h03:/usr/local/
root@h01:/usr/local# scp -r /usr/local/spark-3.3.0 root@h04:/usr/local/
root@h01:/usr/local# scp -r /usr/local/spark-3.3.0 root@h05:/usr/local/
```

启动 Spark

```
root@h01:/usr/local/spark-3.3.0/sbin# ./start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /usr/local/spark-
3.3.0/logs/spark--org.apache.spark.deploy.master.Master-1-h01.out
h03: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark-
3.3.0/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-h03.out
h02: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark-
3.3.0/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-h02.out
h04: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark-
3.3.0/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-h04.out
h05: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark-
3.3.0/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-h05.out
h01: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark-
3.3.0/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-h01.out
root@h01:/usr/local/spark-3.3.0/sbin#
```

## 3 其他

## 3.1 HDFS 重格式化问题

参考 [https://blog.csdn.net/gis\\_101/article/details/52821946](https://blog.csdn.net/gis_101/article/details/52821946)

- 重新格式化意味着集群的数据会被全部删除，格式化前需考虑数据备份或转移问题；
- 先删除主节点（即namenode节点），Hadoop的临时存储目录tmp、namenode存储永久性元数据目录dfs/name、Hadoop系统日志文件目录log 中的内容（注意是删除目录下的内容不是目录）；
- 删除所有数据节点(即datanode节点)，Hadoop的临时存储目录tmp、namenode存储永久性元数据目录dfs/name、Hadoop系统日志文件目录log 中的内容；
- 格式化一个新的分布式文件系统：

```
root@h01:/usr/local/hadoop/bin# ./hadoop namenode -format
```

注意事项:

- Hadoop的临时存储目录tmp（即core-site.xml配置文件中的hadoop.tmp.dir属性，默认值是/tmp/hadoop- $\{user.name\}$ ），如果没有配置hadoop.tmp.dir属性，那么hadoop格式化时将会在/tmp目录下创建一个目录，例如在cloud用户下安装配置hadoop，那么Hadoop的临时存储目录就位于/tmp/hadoop-cloud目录下
- Hadoop的namenode元数据目录（即hdfs-site.xml配置文件中的dfs.namenode.name.dir属性，默认值是 $\{hadoop.tmp.dir\}/dfs/name$ ），同样如果没有配置该属性，那么hadoop在格式化时将自行创建。必须注意的是在格式化前必须清楚所有子节点（即DataNode节点）dfs/name下的内容，否则在启动hadoop时子节点的守护进程会启动失败。这是由于，每一次format主节点namenode，dfs/name/current目录下的VERSION文件会产生新的clusterID、namespaceID。但是如果子节点的dfs/name/current仍存在，hadoop格式化时就不会重建该目录，因此形成子节点的clusterID、namespaceID与主节点（即namenode节点）的clusterID、namespaceID不一致。最终导致hadoop启动失败。