**实验 使用Docker搭建Hadoop分布式集群**

**声明：本文当仅供导引目的，很多的坑需要自行完成跨越**

1. **实验目的**

**自行补充完成**

1. **实验内容和要求**

**增加自己的sudo用户，用户名 s0205 , 要求都在自己用户下做**

**其他的文字：自行补充完成**

1. **实验环境**

**Linux + docker +？**

**4、背景知识概要**

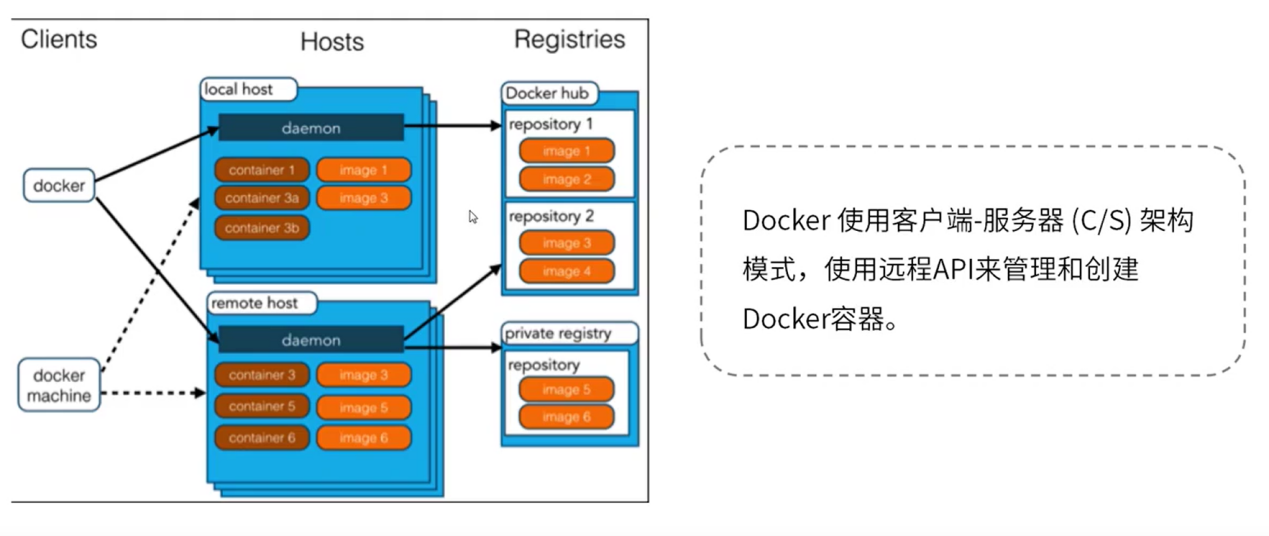
**更多的参考：**

[**https://www.runoob.com/docker/docker-tutorial.html**](https://www.runoob.com/docker/docker-tutorial.html)

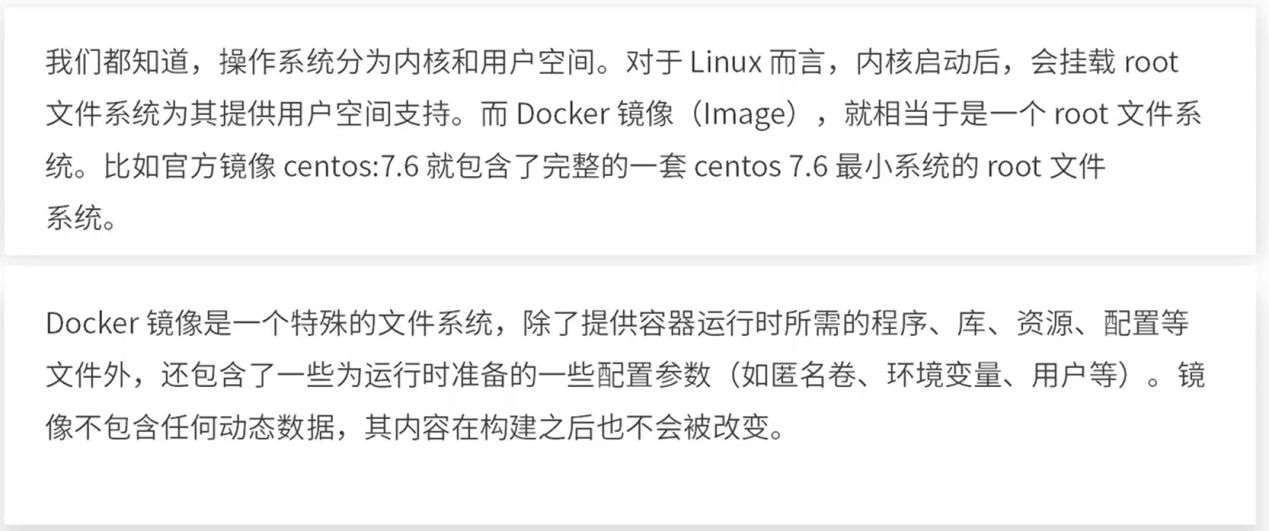
[**https://docs.docker.com/**](https://docs.docker.com/)

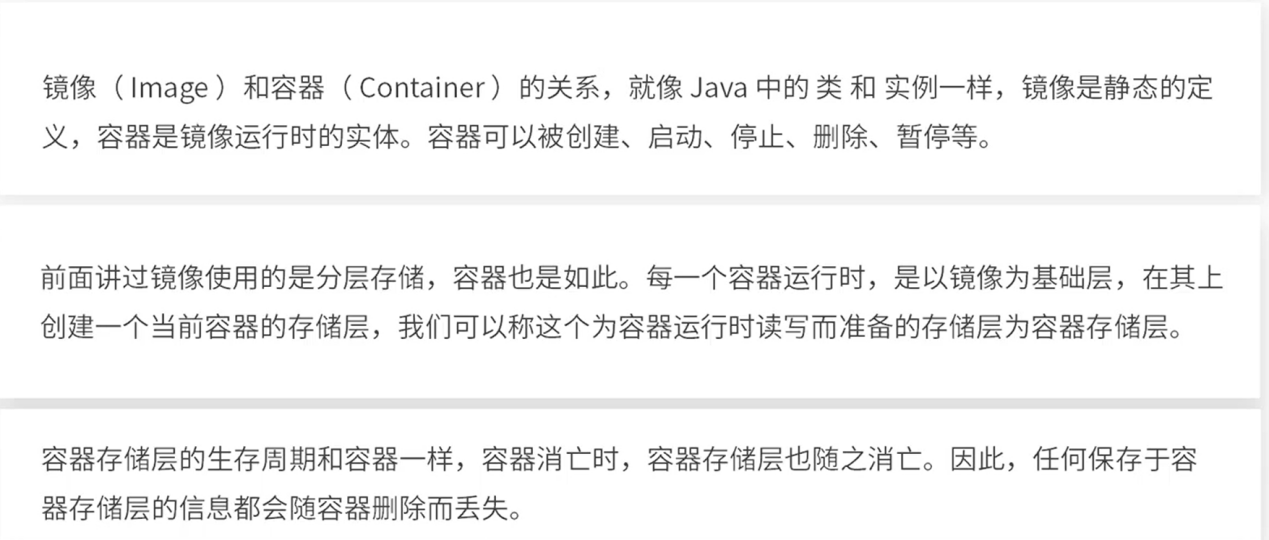
**…….很多很多**

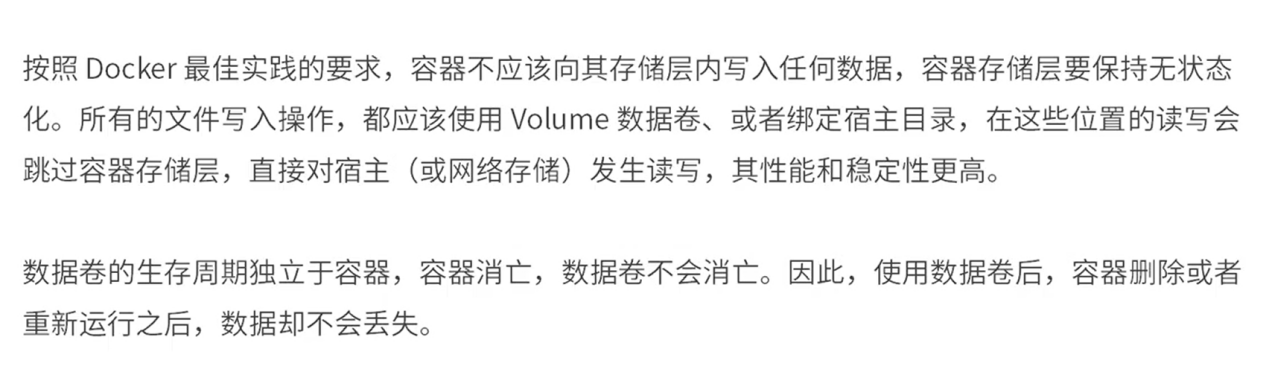




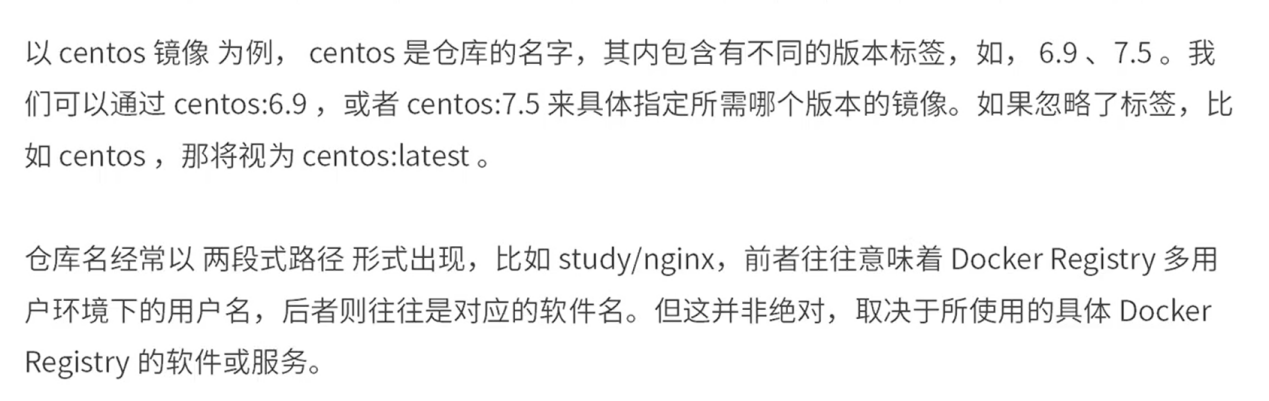




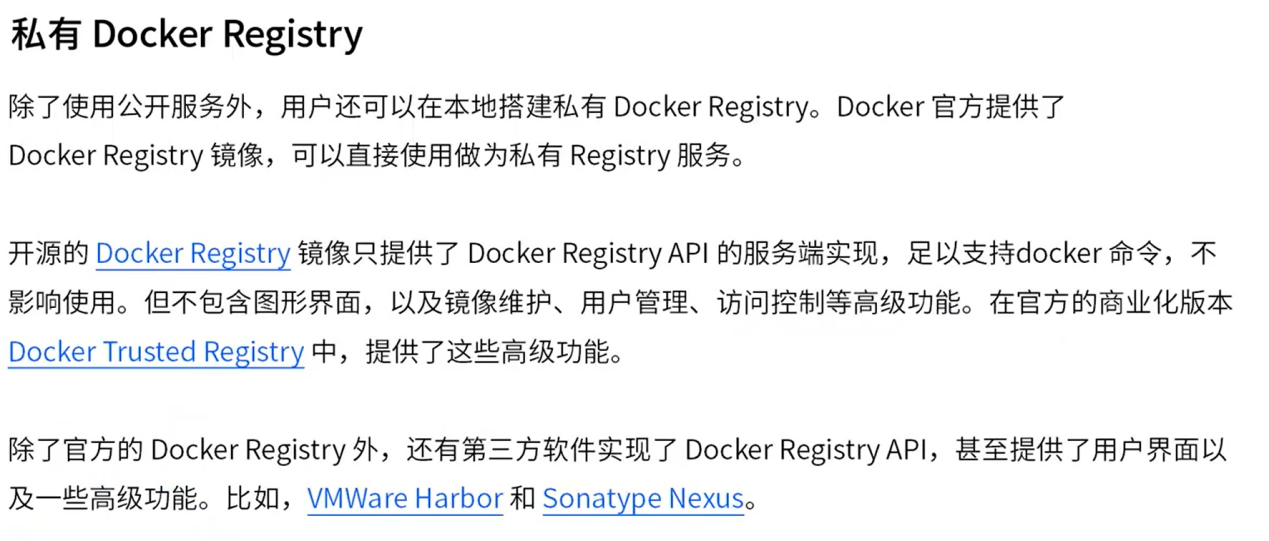














Hadoop 分布式集群环境搭建已经完成前提下，探索如何在Docker上搭建Hadoop集群；首先，我们需要在Ubuntu上安装Docker;

**安装Docker**

安装Docker，首先必须保证是64位Linux系统，其次内核版本必须大于3.10；我们可以用如下命令来检测Ubuntu内核版本:

1. uname -r

Shell 命令

比如笔者电脑执行结果如下：

4.4.0-53-generic

Shell 命令

Sudo apt-get update

Sudo apt-get install docker.io

等这个命令结束之后，Docker即安装成功；我们可以通过下面命令开启Docker服务 (2017年开始 docker 分为企业ee版和社区版ce，我们可以使用ce版，基本功能和io版差别不大)

1. sudo service docker start

Shell 命令

然后我们也可以运行Docker官方提供的hello-world程序检测Docker安装运行成功:

1. sudo docker run hello-world

Shell 命令

这个命令会输出一堆文字，其中有段文字是

Hello from Docker!

This message shows that your installation appears to be working correctly.

则表示安装成功!

docker默认是只有root才能执行Docker命令，因此我们还需要添加用户权限：  
创建docker用户组:

1. sudo groupadd docker

Shell 命令

添加当前用户到Docker用户组:

1. sudo usermod -aG docker $USER

Shell 命令

$USER表示当前用户名，例如，当前用户是hadoop，则把$USER替换成hadoop即可；然后注销，再次登录即可方便使用Docker了；

**在Docker安装Ubuntu系统**

安装好Docker之后，接下来就要在Docker上安装Ubuntu，其实和安装其他镜像一样，只需运行一个命令足矣，如下:

1. docker pull ubuntu

Shell 命令

docker pull命令表示从Docker hub上拉取Ubuntu镜像到本地；这时可以在终端运行以下命令查看是否安装成功,

1. docker images

Shell 命令

有如下输出则表示安装成功:

REPOSITORY TAG IMAGE ID CREATED SIZE

ubuntu latest 4ca3a192ff2a 11 days ago 128.2 MB

docker images表示列出Docker上所有的镜像；镜像也是一堆文件，我们需要在Docker上开启这Ubuntu系统；在启动Ubuntu镜像时，需要先在个人文件下创建一个目录，用于向Docker内部的Ubuntu系统传输文件；命令如下:

1. cd ~
2. mkdir build

Shell 命令

然后再在Docker上运行Ubuntu系统；

1. docker run -it -v /home/hadoop/build:/root/build --name ubuntu ubuntu

Shell 命令

这里解析下这个命令参数：  
\* docker run 表示运行一个镜像；  
\* -i表示开启交互式；-t表示分配一个tty，可以理解为一个控制台；因此-it可以理解为在当前终端上与docker内部的ubuntu系统交互；  
\* -v 表示docker内部的ubuntu系统/root/build目录与本地/home/hadoop/build共享；这可以很方便将本地文件上传到Docker内部的Ubuntu系统；  
\* –name ubuntu 表示Ubuntu镜像启动名称，如果没有指定，那么Docker将会随机分配一个名字；  
\* ubuntu 表示docker run启动的镜像文件；

**Ubuntu系统初始化**

现在系统进入到**容器虚机中，它是一个**很纯净的Ubuntu系统(thin ubuntu)，很多软件是没有安装的，所以我们需要先更新下Ubuntu系统的源以及安装一些必备的软件；

**更新系统软件源**

更新系统源命令如下:

#apt-get update

Shell 命令

**安装vim**

然后我们安装下经常会使用到的vim软件:

#apt-get install vim

Shell 命令

**安装sshd**

接着安装sshd，因为在开启分布式Hadoop时，需要用到ssh连接slave:

#apt-get install ssh

Shell 命令

然后运行如下脚本即可开启sshd服务器:

#/etc/init.d/ssh start

Shell 命令

但是这样的话，就需要每次在开启镜像时，都需要手动开启sshd服务，因此我们把这启动命令写进~/.bashrc文件，这样我们每次登录Ubuntu系统时，都能自动启动sshd服务;

#vim ~/.bashrc

Shell 命令

在该文件中最后一行添加如下内容：

/etc/init.d/ssh start

**配置sshd**

安装好sshd之后，我们需要配置ssh无密码连接本地sshd服务，如下命令:

#ssh-keygen -t rsa *#一直按回车键即可*

#cat id\_dsa.pub >> authorized\_keys

Shell 命令

执行完上述命令之后，即可无密码访问本地sshd服务；

**安装JDK**

因为Hadoop有用到Java，因此还需要安装JDK；直接输入以下命令来安装JDK:

#apt-get install default-jdk

Shell 命令

这个命令会安装比较多的库，可能耗时比较长；等这个命令运行结束之后，即安装成功；然后我们需要配置环境变量，打开~/.bashrc文件，在最后输入如下内容；注意：java-8-openjdk-amd64会因为jdk的在线版本的差异名称有所不同。

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64/

export PATH=$PATH:$JAVA\_HOME/bin

Shell 命令

接着执行如下命令使~/.bashrc生效即可;

#source ~/.bashrc

Shell 命令

**保存镜像文件**

我们在Docker内部的容器做的修改是不会自动保存到镜像库的，也就是说，我们把容器关闭，然后重新开启容器，则之前的设置会全部消失，因此我们需要保存当前的配置；为了达到复用配置信息，我们在每个步骤完成之后，都保存成一个**新的镜像**，然后开启保存的新镜像即可；切换到宿主机Linux中，首先我们要到这个网址注册一个账号<https://hub.docker.com/>；账号注册成功后，然后在终端输入以下信息：

$docker login

Shell 命令

然后会有如下提示信息，输出相应的用户名和密码即可:

Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.

Username: yourusername

Password:

Login Succeeded

登录之后，即可输入以下命令保存修改后的容器为一个新的镜像(参照linux的ps命令的含义)；

$docker ps

Shell 命令

查看当前ubuntu的镜像id，如下

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

fd1fc69d75a3 ubuntu "/bin/bash" About an hour ago Up About an hour ubuntu

然后保存当前镜像为ubuntu/jdkinstalled，表示jdk安装成功的ubuntu版本，命令如下:

1. docker commit fd1fc69d75a3 ubuntu/jdkinstalled

Shell 命令

输出如下:

sha256:05c9bc2b849359d029d417421d6968bdd239aad447ac6054c429b630378118aa

最后输出所有镜像查看是否保存成功:

1. docker images

Shell 命令

输出如下:

REPOSITORY TAG IMAGE ID CREATED SIZE

ubuntu/jdkinstalled latest 05c9bc2b8493 9 seconds ago 899.3 MB

ubuntu latest 4ca3a192ff2a 12 days ago 128.2 MB

以上命令意思如下:  
1. docker ps查看当前运行的容器信息，目前只运行一个ubuntu容器;  
2. docker commit保存fd1fc69d75a3(容器id)容器为一个新的镜像，镜像名称为ubuntu/jdkinstalled  
3. docker images查看当前docker所有镜像，可以看到我们新添加的镜像ubuntu/jdkinstalled

**安装Hadoop**

安装好JDK之后，接下来，我们来安装Hadoop；我们开启保存的那份镜像ubuntu/jdkinstalled：

1. docker run -it -v /home/hadoop/build:/root/build --name ubuntu-jdkinstalled ubuntu/jdkinstalled

Shell 命令

sudo apt-get install net-tools

我们可以用如下命令查看开启的容器:

docker run -it -v /home/hcc/build:/root/build –name ubuntu/hadoopinstalled ubuntu/hadoopinstalled

1. docker ps

Shell 命令

输出如下:

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

325a1c2f2f93 ubuntu/jdkinstalled "/bin/bash" 6 seconds ago Up 5 seconds ubuntu-jdkinstalled

ok，开启系统之后，我们把下载下来的Hadoop安装文件放到共享目录/home/hadoop/build下面，然后在Docker内部Ubuntu系统的/root/build目录即可获取到Hadoop安装文件；在Docker内部的Ubuntu系统安装Hadoop和本地安装一样 hadoop.apache.org

(<https://mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/hadoop-3.2.2/hadoop-3.2.2.tar.gz>)

http://res.aihyzh.com/大数据技术原理与应用3/02/hadoop-3.1.3.tar.gz

#cd /root/build

#tar -zxvf hadoop-3.1.4.tar.gz -C /usr/local

#mv /usr/local/hadoop-3.1.4 /usr/local/hadoop

Shell 命令

如果是单机版Hadoop，到这里已经安装完成了，可以运行如下命令测试下:

1. cd /usr/local/hadoop
2. ./bin/hadoop version

Shell 命令

输出如下:

root@0d1507361a9b:/usr/local/hadoop# ./bin/hadoop version

Hadoop 3.1.4

Source code repository https://github.com/apache/hadoop.git -r 1e877761e8dadd71effef30e592368f7fe66a61b

Compiled by gabota on 2020-07-21T08:05Z

Compiled with protoc 2.5.0

From source with checksum 38405c63945c88fdf7a6fe391494799b

This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-3.1.4.jar

**配置Hadoop集群**

接下来，我们来看下如何配置Hadoop集群；对一些文件的设置和之前教程一样，首先打开hadoop\_env.sh文件，修改JAVA\_HOME(*java-8-openjdk-amd64* 根据实际***安装的jdk主目录名***做相应的替换)

1. *#假设现在/usr/local/hadoop目录下*
2. vim etc/hadoop/hadoop-env.sh
3. *# 将export JAVA\_HOME=${JAVA\_HOME}替换成*
4. export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64/

Shell 命令

接着打开core-site.xml，输入一下内容:

<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>

然后再打开hdfs-site.xml输入以下内容:

<configuration>

<property>

<name>dfs.namenode.name.dir</name>

<value>file:/usr/local/hadoop/namenode\_dir</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>file:/usr/local/hadoop/datanode\_dir</value>

</property>

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

</configuration>

接下来修改mapred-site.xml(复制mapred-site.xml.template,再修改文件名)，输入以下内容:

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

最后修改yarn-site.xml文件，输入以下内容:

<configuration>

<!-- Site specific YARN configuration properties -->

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.resourcemanager.hostname</name>

<value>master</value>

</property>

</configuration>

到这里Hadoop集群配置就已经差不多了，我们先保存这个镜像，在其他终端输入如下命令:

Docker ps

0d1507361a9b

$docker commit a40b99f869ae ubuntu/hadoopinstalled

Shell 命令

ok，接下来，我们在宿主机的三个终端(terminal)上开启三个容器运行ubuntu/hadoopinstalled镜像，分别表示Hadoop集群中的master,slave01和slave02；

1. *# 第一个终端*

$docker run -it -h master --name master ubuntu/hadoopinstalled

1. *# 第二个终端*

$docker run -it -h slave01 --name slave01 ubuntu/hadoopinstalled

1. *# 第三个终端*

$docker run -it -h slave02 --name slave02 ubuntu/hadoopinstalled

Shell 命令

接着进入各个容器中，分别配置master,slave01和slave02的地址信息，这样他们才能找到彼此，分别打开/etc/hosts可以查看本机的ip和主机名信息,最后得到三个ip和主机地址信息如下:

172.18.0.2 master

172.18.0.3 slave01

172.18.0.4 slave02

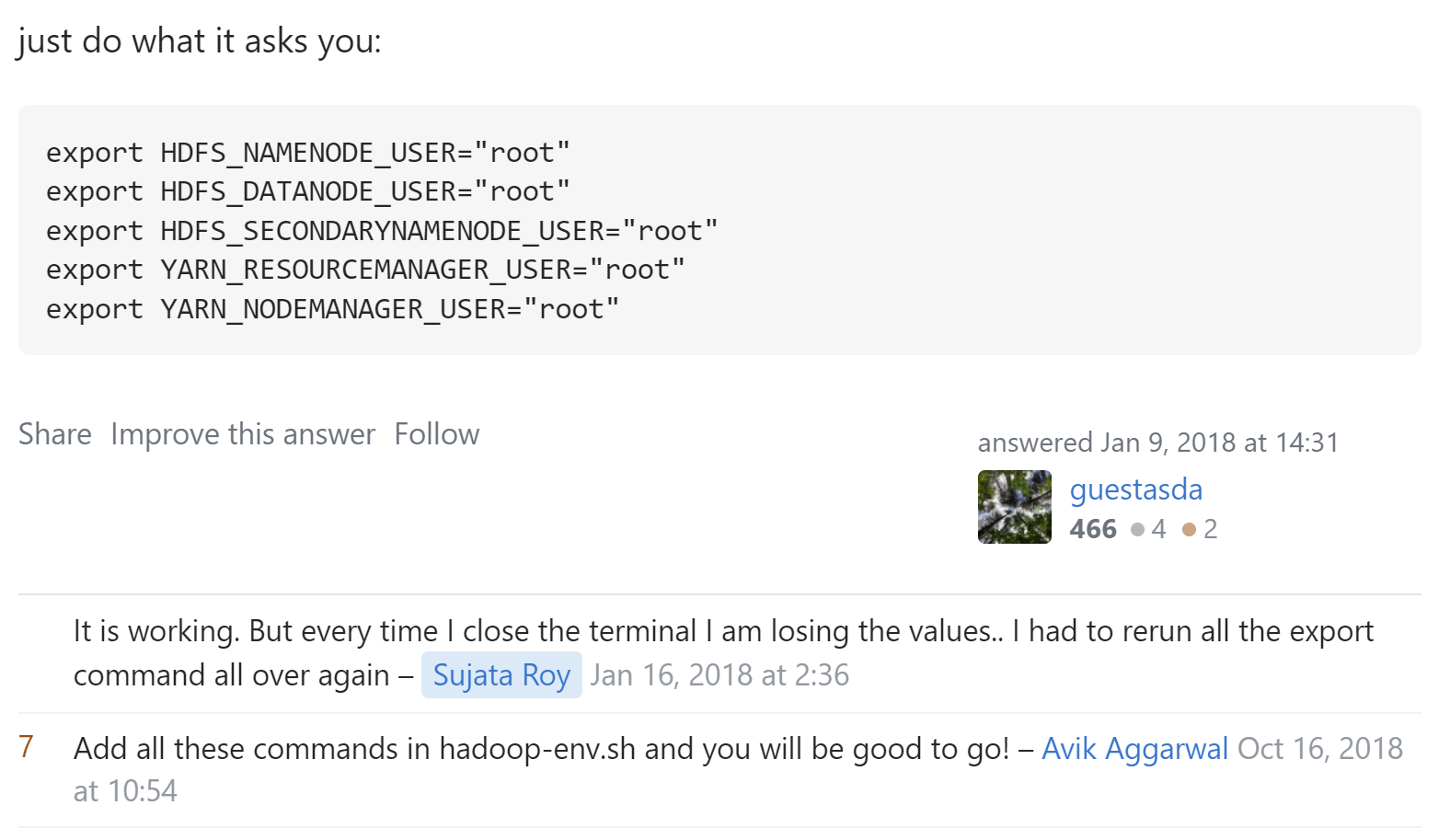
最后把上述三个地址信息分别复制到master,slave01和slave02的/etc/hosts即可，我们可以用如下命令来检测下是否master是否可以连上slave01和slave02(最简单的方式，从master节点容器中执行 scp /etc/hosts slave01:/etc/hosts；scp /etc/hosts slave02:/etc/hosts 大规模部署集群有价值)

ssh slave01

ssh slave02

到这里，我们还差最后一个配置就要完成hadoop集群配置了，打开master上的/usr/local/hadoop/etc/hadoop/workers文件，输入几个主机名(datanode/slave):

<https://stackoverflow.com/questions/48129029/hdfs-namenode-user-hdfs-datanode-user-hdfs-secondarynamenode-user-not-defined>



Ref: https://blog.csdn.net/wang725/article/details/83098685

1. vim /usr/local/Hadoop/etc/hadoop/workers
2. *# 将slaves文件已经被替换为workers*
3. *# 将localhost替换成两个slave的主机名*
4. slave01
5. slave02
6. master
7. *# master也参与作为nodename*

Shell 命令

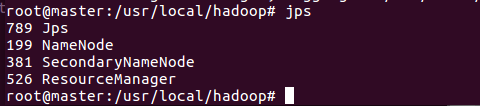
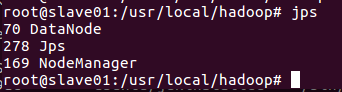
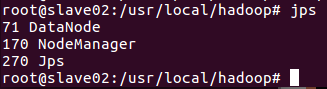
ok，Hadoop集群已经配置完成，我们来启动集群；

在master终端上，首先进入/usr/local/hadoop，然后运行如下命令:

1. cd /usr/local/hadoop
2. bin/hdfs namenode -format
3. sbin/start-all.sh

Shell 命令

这时Hadoop集群就已经开启，我们可以在master,slave01和slave02上分别运行命令jps查看运行结果;  
下面是运行结果图

**总体集群信息显示**

在master 上运行 hdfs dfsadmin -report

包含：

Live datanodes (3):

**运行Hadoop实例程序grep**

到目前为止，我们已经成功启动hadoop分布式集群，接下来，我们通过运行hadoop自带的grep实例来查看下如何在hadoop分布式集群运行程序；这里我们运行的实例是hadoop自带的grep

因为要用到hdfs，所以我们先在hdfs上创建一个目录:

1. ./bin/hdfs dfs -mkdir -p /user/hadoop/input

Shell 命令

然后将/usr/local/hadoop/etc/hadoop/目录下的所有文件拷贝到hdfs上的目录:

1. dfs -put ./etc/hadoop/\*.xml /user/hadoop/input

Shell 命令

然后通过ls命令查看下是否正确将文件上传到hdfs下:

1. dfs -ls /user/hadoop/input

Shell 命令

输出如下:

Found 9 items

-rw-r--r-- 3 root supergroup 4436 2016-12-26 07:40 /user/hadoop/input/capacity-scheduler.xml

-rw-r--r-- 3 root supergroup 1090 2016-12-26 07:40 /user/hadoop/input/core-site.xml

-rw-r--r-- 3 root supergroup 9683 2016-12-26 07:40 /user/hadoop/input/hadoop-policy.xml

-rw-r--r-- 3 root supergroup 1133 2016-12-26 07:40 /user/hadoop/input/hdfs-site.xml

-rw-r--r-- 3 root supergroup 620 2016-12-26 07:40 /user/hadoop/input/httpfs-site.xml

-rw-r--r-- 3 root supergroup 3518 2016-12-26 07:40 /user/hadoop/input/kms-acls.xml

-rw-r--r-- 3 root supergroup 5511 2016-12-26 07:40 /user/hadoop/input/kms-site.xml

-rw-r--r-- 3 root supergroup 866 2016-12-26 07:40 /user/hadoop/input/mapred-site.xml

-rw-r--r-- 3 root supergroup 947 2016-12-26 07:40 /user/hadoop/input/yarn-site.xml

接下来，通过运行下面命令执行实例程序:

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

Shell 命令

过一会，等这个程序运行结束之后，就可以在hdfs上的output目录下查看到运行结果:

1. ./bin/hdfs dfs -cat output/\*
2. 1 dfsadmin
3. 1 dfs.replication
4. 1 dfs.namenode.name.dir
5. 1 dfs.datanode.data.dir

Shell 命令

hdfs文件上的output目录下，输出程序正确的执行结果，hadoop分布式集群顺利执行grep程序;