# 大数据笔记

## 利用Docker搭建大数据处理集群

## 1.1安装docker 1.9.1

|  |
| --- |
| $ curl -sSL -O https://get.docker.com/builds/Linux/x86\_64/docker-1.9.1 && chmod +x docker-1.9.1 && mv docker-1.9.1 /usr/local/bin/docker  $ cp /usr/local/bin/docker /usr/bin/docker  $ service docker start |

## 1.2在Docker上创建Hadoop镜像.

### 1.2.1从Docker Hub官网仓库上获取centos镜像库

|  |
| --- |
| docker pull centos #查看镜像库  docker images |

### 1.2.2生成带有SSH功能的centos的镜像文件

为了后面配置各节点之间的SSH免密码登陆，需要在pull下的centos镜像库种安装SSH, 这里利用 Dockerfile 文件来创建镜像

|  |
| --- |
| cd /usr/local  # 创建一个存放带ssh的centos镜像Dockerfile文件的目录  mkdir -p DockerImagesFiles/centos7.shh  #创建带ssh的centos的Dockerfile 文件  vi Dockerfile  # Dockerfile文件内容  #基于centos镜像库创建  FROM centos  MAINTAINER dys  #安装ssh  RUN yum install -y openssh-server sudo  RUN sed -i 's/UsePAM yes/UsePAM no/g' /etc/ssh/sshd\_config  RUN yum install -y openssh-clients  #配置root名  RUN echo "root:123456" | chpasswd  RUN echo "root ALL=(ALL) ALL" >> /etc/sudoers  #生成ssh key  RUN ssh-keygen -t dsa -f /etc/ssh/ssh\_host\_dsa\_key  RUN ssh-keygen -t rsa -f /etc/ssh/ssh\_host\_rsa\_key  #配置sshd服务  RUN mkdir /var/run/sshd  EXPOSE 22  CMD ["/usr/sbin/sshd", "-D"] |



### 1.2.3 根据上面的Dockerfile生成centos7-ssh镜像

|  |
| --- |
| docker build -t="centos7-ssh" .  #执行完成后，查看已安装的镜像库  docker images |

### 1.2.4 生成Hadoop镜像库文件

在构建Hadoop镜像库的Dockerfile所在目录下，上传已经下载的 jdk-8u101-linux-x64.tar.gz, hadoop-2.7.3.tar.gz,scala-2.11.8.tgz,spark-2.0.1-bin-hadoop2.7.tgz。

注意：这里要提前在Dockerfile文件配置环境变量，如果镜像库构建完成后，在

容器中配置环境变量是不起作用的。

|  |
| --- |
| cd /usr/local  # 创建一个存放hadoop镜像Dockerfile文件的目录  mkdir -p DockerImagesFiles/hadoop  #创建带ssh的centos的Dockerfile 文件  vi Dockerfile  # Dockerfile文件内容  #基于centos7-ssh构建  FROM centos7-ssh  #安装java  ADD jdk-8u101-linux-x64.tar.gz /usr/local/  RUN mv /usr/local/jdk1.8.0\_101 /usr/local/jdk1.8  #配置JAVA环境变量  ENV JAVA\_HOME /usr/local/jdk1.8  ENV PATH $JAVA\_HOME/bin:$PATH  #安装hadoop  ADD hadoop-2.7.3.tar.gz /usr/local  RUN mv /usr/local/hadoop-2.7.3 /usr/local/hadoop  #配置hadoop环境变量  ENV HADOOP\_HOME /usr/local/hadoop  ENV PATH $HADOOP\_HOME/bin:$PATH  #安装scala 注意Spark2.0.1对于Scala的版本要求是2.11.x  ADD scala-2.11.8.tgz /usr/local  RUN mv /usr/local/scala-2.11.8 /usr/local/scala2.11.8  #配置scala环境变量  ENV SCALA\_HOME /usr/local/scala  ENV PATH $SCALA\_HOME/bin:$PATH  #安装spark  ADD spark-2.0.1-bin-hadoop2.7.tgz /usr/local  RUN mv /usr/local/spark-2.0.1-bin-hadoop2.7 /usr/local/spark2.0.1  #配置spark环境变量  ENV SPARK\_HOME /usr/local/spark  ENV PATH $SPARK\_HOME/bin:$PATH  #创建hdfs账号  RUN useradd hdfs  RUN echo "hdfs:12345678" | chpasswd  RUN yum install -y which sudo |



### 1.2.5 根据上面的Dockerfile构建Hadoop镜像库

|  |
| --- |
| docker build -t="hadoop" .  #执行完成后，查看已安装的镜像库  docker images |

### 1.2.6 生成三节点Hadoop容器集群

1.2.6.1首先规划一下节点的主机名称,IP地址(局域网内构建docker镜像时，自动分配172.17.0.1/16网段的IP)和端口号

master 172.17.0.2

slave01 172.17.0.3

slave02 172.17.0.4

1.2.6.2下面在Hadoop镜像上创建三个容器，做为集群的节点

#创建master容器，50070和8088，8080是用来在浏览器中访问hadoop yarn spark #WEB界面，这里分别映射到物理机的50070和8088，8900端口。

|  |
| --- |
| #重点注意：容器启动后，映射比较麻烦，最好在这里映射。  docker run -d -P -p 50070:50070 -p 8088:8088 -p 8080:8900 --name master -h master --add-host slave01:172.17.0.3 --add-host slave02:172.17.0.4 hadoop  #创建slave01容器,在容器host文件，添加hostname，并配置其他节点主机名称和IP地址  docker run -d -P --name slave01 -h slave01 --add-host master:172.17.0.2 --add-host slave02:172.17.0.4 hadoop  #创建slave02容器  docker run -d -P --name slave02 -h slave02 --add-host master:172.17.0.2 --add-host slave01:172.17.0.3 hadoop |

1.2.6.3 查看已创建的容器并更改hadoop和spark2.0.1目录所属用户

|  |
| --- |
| #查看已创建的容器  docker ps -a  #更改hadoop和spark2.0.1目录所属用户  chown -R hdfs:hdfs /usr/local/hadoop  chown -R hdfs:hdfs /usr/local/spark2.0.1 |

## 在Docker上配置三节点Hdfs集群

### 2.1首先规划一下节点的主机名称,IP地址

(局域网内构建docker镜像时，自动分配172.17.0.1/16网段的IP)和端口号

*master* 172.17.0.2   
*slave01* 172.17.0.3   
*slave02* 172.17.0.4

1.2.6.2下面在Hadoop镜像上创建三个容器，做为集群的节点

|  |
| --- |
| #清理不在运行的容器  docker ps -a | grep Exit | cut -d ' ' -f 1 | xargs docker rm  #创建master容器，50070和8088，8080是用来在浏览器中访问hadoop yarn spark #WEB界面，这里分别映射到物理机的50070和8088，8900端口。  #重点注意：容器启动后，映射比较麻烦，最好在这里映射。  docker run -d -P -p 50070:50070 -p 8088:8088 -p 8080:8900 --name master -h master --net docker0 --ip 172.17.0.2 --add-host slave01:172.17.0.3 --add-host slave02:172.17.0.4 hadoop  #创建slave01容器,在容器host文件，添加hostname，并配置其他节点主机名称和IP地址  docker run -d -P --name slave01 -h slave01 --net=bridge --ip=172.17.0.3 --add-host master:172.17.0.2 --add-host slave02:172.17.0.4 hadoop  #创建slave02容器  docker run -d -P --name slave02 -h slave02 --net docker0 --ip 172.17.0.4 --add-host master:172.17.0.2 --add-host slave01:172.17.0.3 hadoop  docker run -itd --name hadoop0 --hostname hadoop0 --net docker0 --ip 192.168.10.30 --add-host hadoop1:192.168.10.31 --add-host hadoop2:192.168.10.32 -d -P -p 50070:50070 -p 8088:8088 hadoop:master  docker save hadoop | bzip2 | ssh root@192.168.0.101 "cat | docker load" |

创建自定义网卡（--net只能使用用户定义的网络）

|  |
| --- |
| [root@localhost hadoop]# docker network create --subnet=172.18.0.0/16 mynetwork  a8d74c267d2b45e9889f652a03ef9d36d4e328335206d3ee11a22aa12ee7c954  [root@localhost hadoop]# docker network ls  NETWORK ID NAME DRIVER SCOPE  4299ec47dd84 bridge bridge local  5471f37d0391 host host local  a8d74c267d2b mynetwork bridge local  e62da77b2055 none null local |

查看已创建的容器并更改hadoop和spark2.0.1目录所属用户

|  |
| --- |
| [root@localhost hadoop]# docker exec -it master /bin/bash  [root@master /]# chown -R hdfs:hdfs /usr/local/hadoop  [root@master /]# chown -R hdfs:hdfs /usr/local/spark2.0.1  [root@master /]# read escape sequence  [root@localhost hadoop]# docker exec -it slave01 /bin/bash  [root@slave01 /]# chown -R hdfs:hdfs /usr/local/hadoop  [root@slave01 /]# chown -R hdfs:hdfs /usr/local/spark2.0.1  [root@slave01 /]# read escape sequence  [root@localhost hadoop]# docker exec -it slave02 /bin/bash  [root@slave02 /]# chown -R hdfs:hdfs /usr/local/hadoop  [root@slave02 /]# chown -R hdfs:hdfs /usr/local/spark2.0.1  [root@slave02 /]# read escape sequence |

### 2.2配置hdfs账号容器各节点间的SSH免密码登陆

分别进入master,slave01,slave02三个容器节点，执行下面命令

|  |
| --- |
| #切换到hdfs账号  su hdfs  #生成hdfs账号的key，执行后会有多个输入提示，不用输入任何内容，全部直接回车即可  ssh-keygen  #拷贝到其他节点  ssh-copy-id -i /home/hdfs/.ssh/id\_rsa -p 22 hdfs@master  ssh-copy-id -i /home/hdfs/.ssh/id\_rsa -p 22 hdfs@slave01  ssh-copy-id -i /home/hdfs/.ssh/id\_rsa -p 22 hdfs@slave02  #验证是否设置成功  ssh slave01 |