|  |  |  |
| --- | --- | --- |
|  | | |
|  | | |
| Docker课程设计 | | |
| 课程名称: | Docker课程设计 |  |
| 学生姓名: | 张文斌 |  |
| 专业班级: | 计算机16-1 |  |
|  | | |
| 2019年6月26日 | | |

目录

[1 标题1 1](#_Toc12469495)

[1.1 标题2 1](#_Toc12469496)

[1.1.1 标题3 1](#_Toc12469497)

[2 标题1 1](#_Toc12469498)

[2.1 标题2 1](#_Toc12469499)

# Docker环境搭建

所有课程设计内容均在一个虚拟机中的docker环境中设计完成

## 虚拟机

CentOS-7-x86\_64-DVD-1708.iso

CentOS Linux release 7.6.1810 (Core)

Linux localhost.localdomain 3.10.0-693.el7.x86\_64 #1 SMP Tue Aug 22 21:09:27 UTC 2017 x86\_64 x86\_64 x86\_64 GNU/Linux

## Docker version

Client:

Version: 17.05.0-ce

API version: 1.29

Go version: go1.7.5

Git commit: 89658be

Built: Thu May 4 22:06:25 2017

OS/Arch: linux/amd64

Server:

Version: 17.05.0-ce

API version: 1.29 (minimum version 1.12)

Go version: go1.7.5

Git commit: 89658be

Built: Thu May 4 22:06:25 2017

OS/Arch: linux/amd64

Experimental: false

## jdk

java version "1.8.0\_91"

Java(TM) SE Runtime Environment (build 1.8.0\_91-b14)

Java HotSpot(TM) 64-Bit Server VM (build 25.91-b14, mixed mode)

# 部署Hadoop HA 和spark集群

最终需要实现使用Dockerfile和docker-compose一键部署，为保证配置文件完全有效，需要先在容器中手动配置完成，然后导出配置文件供接下来使用。

## 在容器中配置Hadoop HA和spark集群

### 统一环境

所需各软件包版本如下：

* hadoop-2.7.3.tar.gz
* jdk-8u91-linux-x64.rpm
* scala-2.12.8.tgz
* spark-2.4.3-bin-hadoop2.7.tgz
* zookeeper-3.4.6.tar.gz

### Hadoop配置文件

#### core-site.xml

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://mycluster</value>

</property>

<property>

<name>ha.zookeeper.quorum</name>

<value>hadoop-master:2181,hadoop-slave1:2181,hadoop-slave2:2181</value>

</property>

<property>

<name>fs.trash.interval</name>

<value>2</value>

</property>

<property>

<name>fs.trash.checkpoint.interval</name>

<value>1</value>

</property>

<property>

<name>hadoop.http.staticuser.user</name>

<value>root</value>

</property>

</configuration>

#### hdfs-site.xml

<configuration>

<property>

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

<value>file:///root/hdfs/namenode</value>

<description>NameNode directory for namespace and transaction logs storage.</description>

</property>

<property>

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

<value>file:///root/hdfs/datanode</value>

<description>DataNode directory</description>

</property>

<property>

<name>dfs.replication</name>

<value>2</value>

</property>

<property>

<name>dfs.nameservices</name>

<value>mycluster</value>

</property>

<property>

<name>dfs.ha.namenodes.mycluster</name>

<value>nn1,nn2</value>

</property>

<property>

<name>dfs.namenode.rpc-address.mycluster.nn1</name>

<value>hadoop-master:9000</value>

</property>

<property>

<name>dfs.namenode.rpc-address.mycluster.nn2</name>

<value>hadoop-slave1:9000</value>

</property>

<property>

<name>dfs.namenode.http-address.mycluster.nn1</name>

<value>hadoop-master:50070</value>

</property>

<property>

<name>dfs.namenode.http-address.mycluster.nn2</name>

<value>hadoop-slave1:50070</value>

</property>

<property>

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

<value>qjournal://hadoop-master:8485;hadoop-slave1:8485;hadoop-slave2:8485/mycluster</value>

</property>

<property>

<name>dfs.client.failover.proxy.provider.mycluster</name>

<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>

</property>

<property>

<name>dfs.ha.fencing.methods</name>

<value>sshfence</value>

</property>

<property>

<name>dfs.ha.fencing.ssh.private-key-files</name>

<value>/root/.ssh/id\_rsa</value>

</property>

<property>

<name>dfs.journalnode.edits.dir</name>

<value>/root/hdfs/jnode\_edits\_dir</value>

</property>

<property>

<name>dfs.permissions.enable.mycluster</name>

<value>false</value>

</property>

<property>

<name>dfs.ha.automatic-failover.enabled</name>

<value>true</value>

</property>

</configuration>

#### yarn-site.xml

<configuration>

<property>

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

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce\_shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<property>

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

<value>hadoop-master</value>

</property>

<property>

<name>yarn.resourcemanager.ha.enabled</name>

<value>true</value>

</property>

<property>

<name>yarn.resourcemanager.cluster-id</name>

<value>ycluster</value>

</property>

<property>

<name>yarn.resourcemanager.ha.rm-ids</name>

<value>rm1,rm2</value>

</property>

<property>

<name>yarn.resourcemanager.hostname.rm1</name>

<value>hadoop-master</value>

</property>

<property>

<name>yarn.resourcemanager.hostname.rm2</name>

<value>hadoop-slave1</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address.rm1</name>

<value>hadoop-master:8088</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address.rm2</name>

<value>hadoop-slave1:8088</value>

</property>

<property>

<name>yarn.resourcemanager.zk-address</name>

<value>hadoop-slave1:2181,hadoop-slave2:2181,hadoop-master:2181</value>

</property>

<property>

<name>yarn.resourcemanager.recovery.enabled</name>

<value>true</value>

</property>

<property>

<name>yarn.resourcemanager.store.class</name>

<value>org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore</value>

</property>

#### hadoop-env.sh

#修改$JAVA\_HOME

export JAVA\_HOME=/usr/java/jdk1.8.0\_91

#### mapred-site.xml

<configuration>

<property>

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

<value>yarn</value>

</property>

</configuration>

#### slaves

hadoop-slave1

hadoop-slave2

### zookeeper配置文件

#### zoo.cfg

dataDir=/usr/local/zookeeper-3.4.6/data

dataLogDir=/usr/local/zookeeper-3.4.6/logs

server.1=hadoop-master:2888:3888

server.2=hadoop-slave1:2888:3888

server.3=hadoop-slave2:2888:3888

### spark配置文件

#### spark-env.sh

export JAVA\_HOME=/usr/java/jdk1.8.0\_91

export HADOOP\_CONF\_DIR=/usr/local/hadoop-2.7.3/etc/hadoop

export SCALA\_HOME=/usr/local/scala-2.12.8

export SPARK\_MASTER\_HOST=hadoop-master

export SAPRk\_WORKER\_MEMORY=1g

#### slaves

hadoop-master

hadoop-slave1

hadoop-slave2

## 一键部署Hadoop HA和spark集群

### 编写Dockerfile

FROM centos

MAINTAINER Harper Cheung <harper.cheung@qq.com>

WORKDIR /root

COPY file/\* /file/

RUN rpm -ivh /file/jdk-8u91-linux-x64.rpm

RUN tar -zxvf /file/hadoop-2.7.3.tar.gz -C /usr/local

RUN tar -zxvf /file/zookeeper-3.4.6.tar.gz -C /usr/local

RUN tar -xvf /file/scala-2.12.8.tgz -C /usr/local

RUN tar -xvf /file/spark-2.4.3-bin-hadoop2.7.tgz -C /usr/local

ENV JAVA\_HOME=/usr/java/jdk1.8.0\_91

ENV CLASSPATH=.:$JAVA\_HOME/jre/lib/rt.jar:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

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

ENV HADOOP\_HOME=/usr/local/hadoop-2.7.3

ENV HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/native

ENV HADOOP\_OPTS="-Djava.library.path=$HADOOP\_HOME/lib/native"

ENV PATH=$PATH:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin

ENV SCALA\_HOME=/usr/local/scala-2.12.8

ENV PATH=$PATH:$SCALA\_HOME/bin

ENV SPARK\_HOME=/usr/local/spark-2.4.3-bin-hadoop2.7

ENV PATH=$PATH:$SPARK\_HOME/bin:$SPARK\_HOME/sbin

ENV ZOOKEEPER\_HOME=/usr/local/zookeeper-3.4.6

ENV PATH=$PATH:$ZOOKEEPER\_HOME/bin

RUN mkdir -p /root/hdfs/namenode && \

mkdir -p /root/hdfs/datanode && \

mkdir $HADOOP\_HOME/logs && \

mkdir $ZOOKEEPER\_HOME/data && \

mkdir $ZOOKEEPER\_HOME/logs

RUN yum update -y && \

yum install -y openssh-server openssh-clients which psmisc

RUN ssh-keygen -t rsa -f ~/.ssh/id\_rsa -P '' && \

cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

RUN mv /file/ssh\_config /etc/ssh/

ADD file/hadoop/hadoop-env.sh $HADOOP\_HOME/etc/hadoop

ADD file/hadoop/hdfs-site.xml $HADOOP\_HOME/etc/hadoop

ADD file/hadoop/core-site.xml $HADOOP\_HOME/etc/hadoop

ADD file/hadoop/mapred-site.xml $HADOOP\_HOME/etc/hadoop

ADD file/hadoop/yarn-site.xml $HADOOP\_HOME/etc/hadoop

ADD file/hadoop/slaves $HADOOP\_HOME/etc/hadoop

ADD file/zookeeper/zoo.cfg $ZOOKEEPER\_HOME/conf

#ADD file/zookeeper/slave2/myid $ZOOKEEPER\_HOME/data

ADD file/spark/slaves $SPARK\_HOME/conf

ADD file/spark/spark-env.sh $SPARK\_HOME/conf

RUN mv $SPARK\_HOME/sbin/start-all.sh $SPARK\_HOME/sbin/start-spark-all.sh

RUN mv $SPARK\_HOME/sbin/stop-all.sh $SPARK\_HOME/sbin/stop-spark-all.sh

RUN chmod +x /file/start.sh && \

chmod +x $HADOOP\_HOME/sbin/start-dfs.sh && \

chmod +x $HADOOP\_HOME/sbin/start-yarn.sh

CMD [ "sh", "-c", "systemctl start sshd; bash"]

### 编写docker-compose.yml

---

services:

hadoop-master:

command:

- /bin/sh

- "-c"

- |

echo 1 >>/usr/local/zookeeper-3.4.6/data/myid

systemctl start sshd

zkServer.sh start

/usr/sbin/init

container\_name: hadoop-master

depends\_on:

- hadoop-slave1

- hadoop-slave2

environment:

container: docker

hostname: hadoop-master

image: "harper/hadoop-cluster:8.0"

network\_mode: hadoop

ports:

- "8088:8088"

- "8080:8080"

- "50070:50070"

privileged: true

volumes:

- "/sys/fs/cgroup:/sys/fs/cgroup"

hadoop-slave1:

command:

- /bin/sh

- "-c"

- |

echo 2 >>/usr/local/zookeeper-3.4.6/data/myid

systemctl start sshd

zkServer.sh start

/usr/sbin/init

container\_name: hadoop-slave1

environment:

container: docker

hostname: hadoop-slave1

image: "harper/hadoop-cluster:8.0"

network\_mode: hadoop

privileged: true

volumes:

- "/sys/fs/cgroup:/sys/fs/cgroup"

hadoop-slave2:

command:

- /bin/sh

- "-c"

- |

echo 3 >>/usr/local/zookeeper-3.4.6/data/myid

systemctl start sshd

zkServer.sh start

/usr/sbin/init

container\_name: hadoop-slave2

environment:

container: docker

hostname: hadoop-slave2

image: "harper/hadoop-cluster:8.0"

network\_mode: hadoop

privileged: true

volumes:

- "/sys/fs/cgroup:/sys/fs/cgroup"

version: "3.4"

# nginx+tomcat+mysql

要求实现：

1. MySQL读写分离和高可用
2. tomcat做负载均衡分布
3. nginx做负载均衡和反向代理
4. 要求用AB测试法（apache的开源软件），进行压力测试，必须测出当前环境最大的峰值以及最短响应时间，形成测试报告

## MySQL主从复制

### 编写docker-compose.yml

---

mysql-master:

container\_name: mysql-master

environment:

- MYSQL\_ROOT\_PASSWORD=root

hostname: msyql-master

image: "mysql:5.7"

ports:

- "3306:3306"

volumes:

- "/root/harper-mysql/mysqlData/master/cnf:/etc/mysql/conf.d"

- "/root/harper-mysql/mysqlData/master/data:/var/lib/mysql"

mysql-slave:

container\_name: mysql-slave

environment:

- MYSQL\_ROOT\_PASSWORD=root

hostname: msyql-slave

image: "mysql:5.7"

ports:

- "3307:3306"

volumes:

- "/root/harper-mysql/mysqlData/slave/cnf:/etc/mysql/conf.d"

- "/root/harper-mysql/mysqlData/slave/data:/var/lib/mysql"

### 配置文件mysql.conf

[mysqld]

pid-file = /var/run/mysqld/mysqld.pid

socket = /var/run/mysqld/mysqld.sock

datadir = /var/lib/mysql

symbolic-links=0

character-set-server = utf8

innodb\_print\_all\_deadlocks = 1

max\_connections = 2000

max\_connect\_errors = 6000

open\_files\_limit = 65535

table\_open\_cache = 128

max\_allowed\_packet = 4M

binlog\_cache\_size = 1M

max\_heap\_table\_size = 8M

tmp\_table\_size = 16M

read\_buffer\_size = 2M

read\_rnd\_buffer\_size = 8M

sort\_buffer\_size = 8M

join\_buffer\_size = 28M

key\_buffer\_size = 4M

thread\_cache\_size = 8

query\_cache\_type = 1

query\_cache\_size = 8M

query\_cache\_limit = 2M

ft\_min\_word\_len = 4

log-bin = mysql-bin

server-id = 1

binlog\_format = mixed

performance\_schema = 0

explicit\_defaults\_for\_timestamp

interactive\_timeout = 28800

wait\_timeout = 28800

sql\_mode=NO\_ENGINE\_SUBSTITUTION,NO\_AUTO\_CREATE\_USER,STRICT\_TRANS\_TABLES

[mysqldump]

quick

max\_allowed\_packet = 16M

[myisamchk]

key\_buffer\_size = 8M

sort\_buffer\_size = 8M

read\_buffer = 4M

write\_buffer = 4M

## nginx+tomcat实现反向代理和负载均衡

### 编写tomcat的Dockerfile

FROM centos

MAINTAINER Harper Cheung <harper.cheung@qq.com>

WORKDIR /usr/local

ADD apache-tomcat-8.5.43.tar.gz /usr/local

ADD jdk-8u101-linux-x64.tar.gz /usr/local/

COPY server.xml /usr/local/apache-tomcat-8.5.43/conf/server.xml

ENV JAVA\_HOME /usr/local/jdk1.8.0\_101

ENV CLASSPATH $JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

ENV CATALINA\_HOME /usr/local/apache-tomcat-8.5.43

ENV PATH $PATH:$JAVA\_HOME/bin:$JAVA\_HOME/jre/bin:$CATALINA\_HOME/lib:$CATALINA\_HOME/bin

EXPOSE 8080

ENTRYPOINT ["/usr/local/apache-tomcat-8.5.43/bin/catalina.sh", "run"]

### nginx配置文件nginx.conf

user nginx;

worker\_processes 1;

error\_log /var/log/nginx/error.log warn;

pid /var/run/nginx.pid;

events {

worker\_connections 1024;

}

http {

server {

listen 80;

location / {

proxy\_pass http://blance;

}

}

upstream blance{

server 192.168.235.146:8081;

server 192.168.235.146:8082;

server 192.168.235.146:8083;

}

include /etc/nginx/mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

keepalive\_timeout 65;

include /etc/nginx/conf.d/\*.conf;

}

### 编写docker-compose.yml

---

nginx:

container\_name: nginx

hostname: nginx

image: nginx

ports:

- "82:80"

valumes:

- "/root/harper-nginx-tomcat/nginx/nginx:conf:/etc/nginx/nginx.conf"

tomcat1:

container\_name: tomcat1

hostname: tomcat1

image: harper/tomcat:3.0

ports:

- "8081:8080"

volumes:

- "/root/harper-nginx-tomcat/tomcat1/:/usr/local/tomcat/webapps/ROOT/"

tomcat2:

container\_name: tomcat2

hostname: tomcat2

image: harper/tomcat:3.0

ports:

- "8082:8080"

volumes:

- "/root/harper-nginx-tomcat/tomcat2/:/usr/local/tomcat/webapps/ROOT/"

tomcat3:

container\_name: tomcat3

hostname: tomcat3

image: harper/tomcat:3.0

ports:

- "8083:8080"

volumes:

- "/root/harper-nginx-tomcat/tomcat2/:/usr/local/tomcat/webapps/ROOT/"

# 心得体会

在做课设之前就下定决心，一定要拿到优秀。当得知课程设计题目要求的时候感觉到了压力，但是仍然有信心去完成要求，我就是要去研究它，搞定它。而且我的目标非常明确，就是要就业，所以技术方面一定得努力努力再努力。

拿到题目之后紧接着就投入到紧张的工作中，开头还算顺利，之前有认真去学习docker和hadoop，在容器中部署Hadoop的HA和spark集群时没有遇到太大的困难，但是当要去写Dockerfile和docker-compose.yml进行一键部署时遇到了难题。

首先我去详细的学习了Dockerfile和docker-compose.yml和相关知识，然后自己尝试进行编写，运行的时候会出现很多错误，只能是一条一条的去修改指令，每尝试一次都要创建一次镜像和容器，大量的重复工作，大量重复的指令，而且电脑不太给力，非常的卡，是不是就提示CUP卡顿多少多少秒。为了提高效率，又简单学习了Linxu脚本的编写，还好都是一些重复性的命令，没有太多的逻辑问题，脚本较好编写。这一段工作持续了接近两天，两天的时间完成Dockerfile和docker-compose.yml的编写。但是又遇到一个严峻的问题，namenode的格式化放到Dockerfile里面是没法完成的，必须在容器启动时执行。这时候一共考虑了这么几套解决办法：

1. 写在Dockerfile的CMD命令中
2. 写在docker-compose.yml的command命令中
3. 编写一个脚本，运行docker-compose之后直接执行这部分命令
4. 将这部分命令写成脚本，从Dockerfile中ADD到容器，然后去容器中手动运行此脚本

CMD和command中的命令每次重启容器都会运行，所以我们不能将格式化的命令放在那里面，所以只剩3和4两个选项，原理上来说都是可行的，我也更倾向于用第三个方法，凡是最终失败了，没有查出原因，所以最终使用了第四种方法。

由于第一部分的题目占用了太多的时间，留给第二部分题目的时间不多了，最终到答辩还有部分功能没来得及去做，也比较遗憾。第二部分首先入手解决了负载均衡和反向代理的问题，使用一个nginx加三个tomcat完成部署，这部分操作虽然不是太难，但是在实际开发中是非常有用的，很大提高了项目的性能及可用性。MySQL的读写分离和高可用部分更是实用，较之前极大提高了数据库的读写性能，能够应对更大的数据压力。具体在实现方面了解到可以使用两主两从的结构，同时配合mycat和proxy来完成。而这些都需要基于MySQL的主从复制，目前我只是实现了主从复制这一部分，剩下的虽然没能在答辩之前完成，但是我想利用暑假的时间在去解决完善，因为这个课程设计的内容真的太有用的，无论是对个人的技术还是以后的工作都有极大的帮助。