主要策略：

宿主机为Ubuntu18.04操作系统，安装docker 5.18.09，将宿主机的操作系统制作成docker基础镜像，之后使用自制的基础镜像在docker中启动6个容器，分配固定IP，再在3个容器中配置webServer集群。

编号	静态IP	容器名称
1	172.20.0.11	HAProxy
2	172.20.0.12	tomcat1
3	172.20.0.13	tomcat2
4	172.20.0.14	amoeba
5	172.20.0.15	mysql1
6	172.20.0.16	mysql2
7	172.20.0.17	tomcat3
8	172.20.0.18	mysql3

一、配置负责负载均衡的节点HAProxy

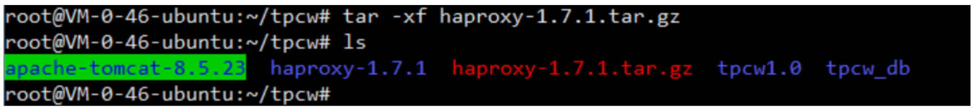
1.下载最新haproxy安装包：haproxy-1.5.8.tar.gz

最好不要安装最新版本，因为很多步骤错误没有解决方法。

下载链接：https://www.haproxy.org/download/1.5/src/haproxy-1.5.8.tar.gz

2.上传到Linux的haproxy用户根目录下，并解压：

tar -zxvf haproxy-1.5.8.tar.gz



```
root@VM-0-46-ubuntu:~/tpcw# tar -xf haproxy-1.7.1.tar.gz
root@VM-0-46-ubuntu:~/tpcw# ls
apache-tomcat-8.5.23  haproxy-1.7.1  haproxy-1.7.1.tar.gz  tpcw1.0  tpcw_db
root@VM-0-46-ubuntu:~/tpcw#
```

创建目录/home/tank/haproxy

```
mkdir /home/tank/haproxy
```

3.编译安装

```
cd haproxy-1.5.8
make  TARGET=linux26 ARCH=x86_64 PREFIX=/home/haproxy/haproxy
```

#将haproxy安装到/home/tank/haproxy ,TARGET是指定内核版本

```
make install PREFIX=/home/tank/haproxy
```

进入/home/tank/haproxy目录创建/home/tank/haproxy/conf目录，复制配置examples的haproxy.cfg文件到conf目录中(没有则创建一个)

```
cp  /home/haproxy/haproxy-1.5.8/examples/haproxy.cfg  /home/haproxy/haproxy/conf/
```

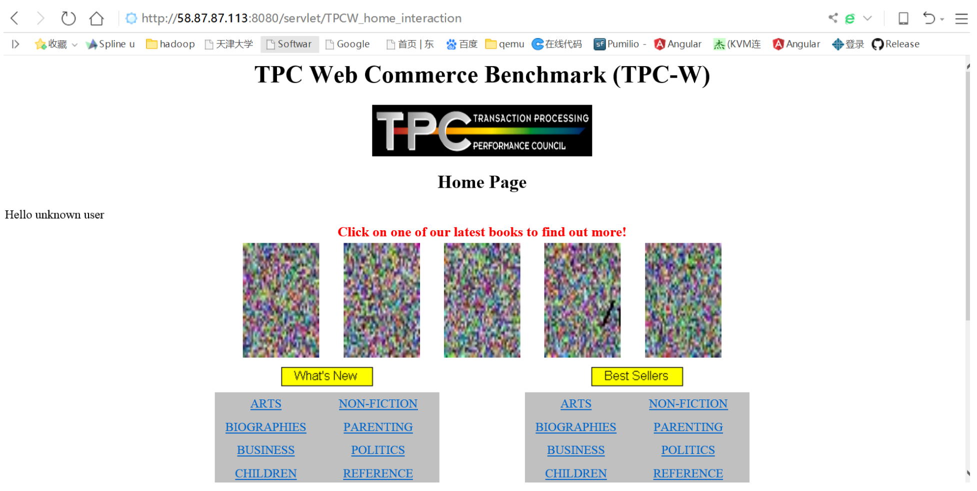
4.修改配置haproxy.cfg

```
###########全局配置#########
global
 daemon
 nbproc 1
 pidfile /home/tank/haproxy/conf/haproxy.pid #haproxy 进程PID文件
```

```
  ulimit-n 819200
  maxconn 4096
########默认配置############
defaults
  mode http
  retries 2
  option redispatch
  option abortonclose
  maxconn 4096
  timeout connect 5000ms
  timeout client 30000ms
  timeout server 30000ms
  balance roundrobin
 ########统计页面配置########
 listen admin_stats
  bind 0.0.0.0:8080
  mode http
  option httplog
  log 127.0.0.1 local0 err
  maxconn 10
  stats refresh 30s
  stats uri /stats
  stats realm XingCloud\ Haproxy
  stats auth admin:admin
  stats auth Frank:Frank
  stats hide-version
  stats admin if TRUE
 listen server_haproxy
  bind 0.0.0.0:8080
  mode tcp
  server web1 172.20.0.12:8080 weight 1
  server web2 172.20.0.13:8080 weight 1
```

6.启动服务

```
   /home/tank/haproxy/sbin/haproxy -f /home/tank/haproxy/conf/haproxy.cfg
```



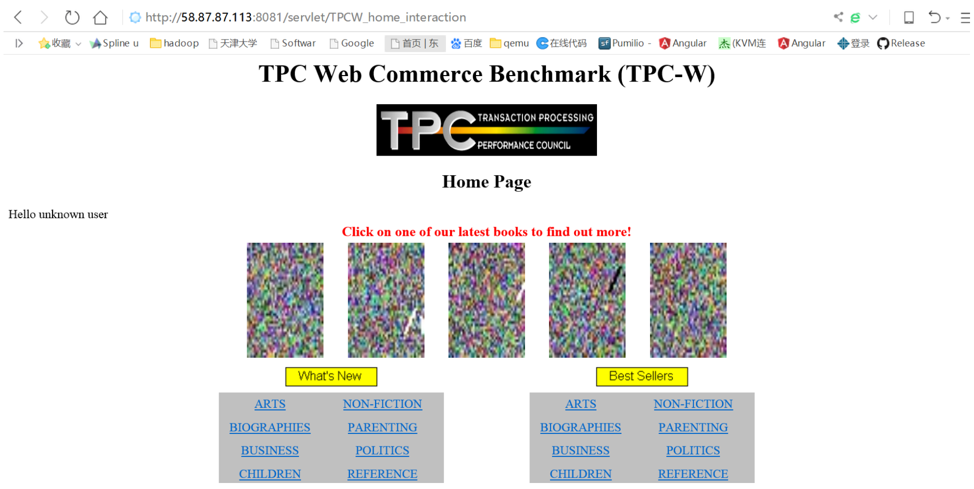
TPC Web Commerce Benchmark (TPC-W)

Home Page

Hello unknown user

Click on one of our latest books to find out more!

通过haproxy的8081端口也可以访问



http://58.87.87.113:8081/servlet/TPCW_home_interaction

TPC Web Commerce Benchmark (TPC-W)

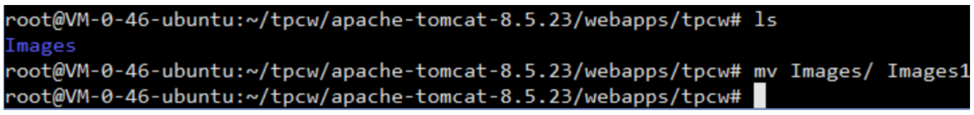
Home Page

Hello unknown user

Click on one of our latest books to find out more!

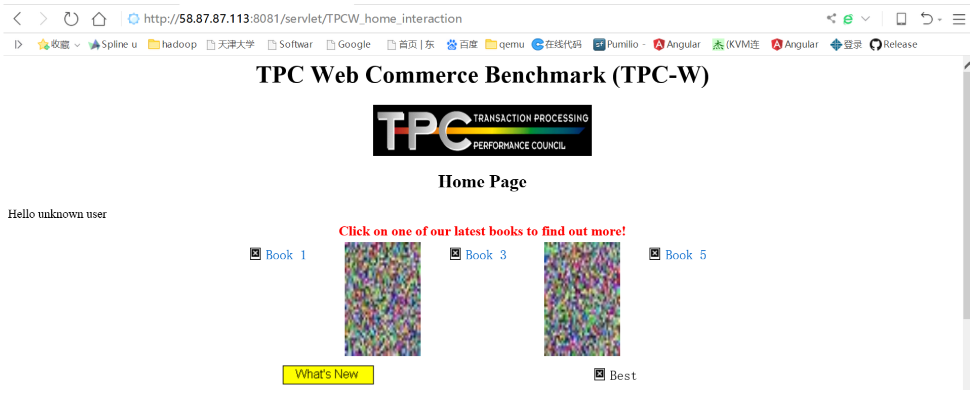
What's New		Best Sellers	
ARTS	NON-FICTION	ARTS	NON-FICTION
BIOGRAPHIES	PARENTING	BIOGRAPHIES	PARENTING
BUSINESS	POLITICS	BUSINESS	POLITICS
CHILDREN	REFERENCE	CHILDREN	REFERENCE

删掉其中一台tomcat上的图片目录，我们观察变化



```
root@VM-0-46-ubuntu:~/tpcw/apache-tomcat-8.5.23/webapps/tpcw# ls
Images
root@VM-0-46-ubuntu:~/tpcw/apache-tomcat-8.5.23/webapps/tpcw# mv Images/ Images1
root@VM-0-46-ubuntu:~/tpcw/apache-tomcat-8.5.23/webapps/tpcw# 
```

可以看到 其中部分图片显示不出来了，代表确实请求转发给了两台tomcat服务器，而且分发下去的是随机请求（缺失的图片 位置不一样）



http://58.87.87.113:8081/servlet/TPCW_home_interaction

TPC Web Commerce Benchmark (TPC-W)

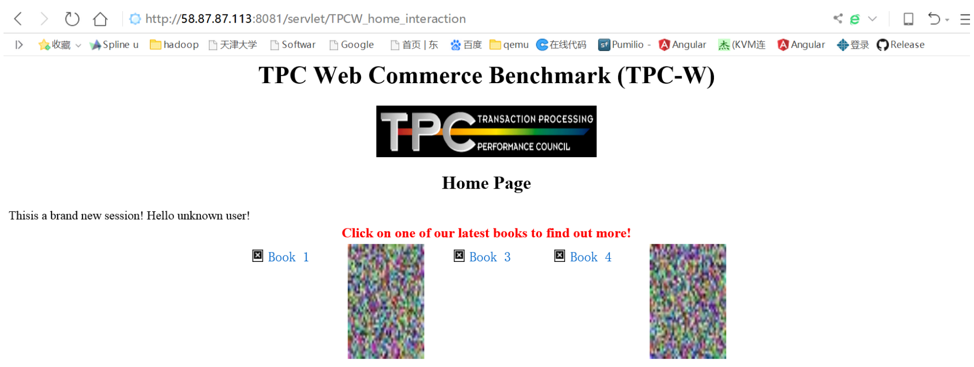
Home Page

Hello unknown user

Click on one of our latest books to find out more!

Book 1 Book 3 Book 5

What's New Best



http://58.87.87.113:8081/servlet/TPCW_home_interaction

TPC Web Commerce Benchmark (TPC-W)

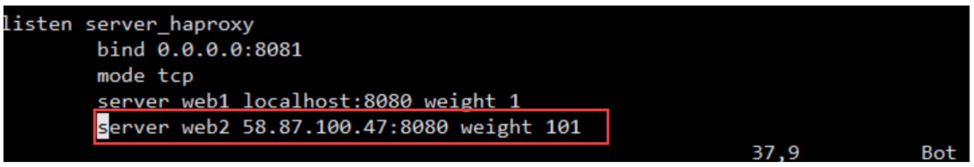
Home Page

Thisis a brand new session! Hello unknown user!

Click on one of our latest books to find out more!

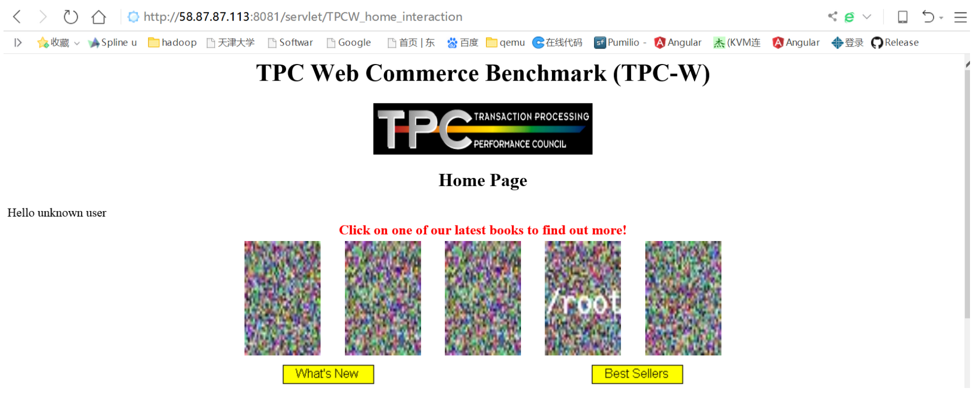
Book 1 Book 3 Book 4

之后我们调节权重，本地的tomcat里删除了图片文件夹，访问本地的时候会缺失图片，那么我们把远程tomcat的权重调大，看看效果



```
listen server_haproxy
        bind 0.0.0.0:8081
        mode tcp
        server web1 localhost:8080 weight 1
        server web2 58.87.100.47:8080 weight 101
                                                            37,9          Bot
```

可以看到，请求转发给远程服务器的频率大大增加了，很少出现过图片缺失的现象，



http://58.87.87.113:8081/servlet/TPCW_home_interaction

TPC Web Commerce Benchmark (TPC-W)

Home Page

Hello unknown user

Click on one of our latest books to find out more!

What's New　　Best Sellers

重启服务：

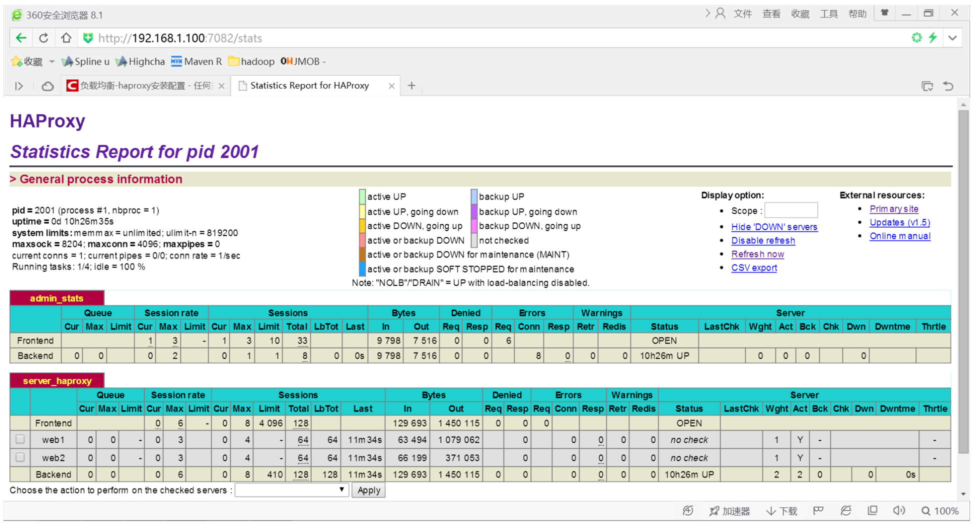
/home/tank/haproxy/sbin/haproxy -f /home/tank/haproxy/conf/haproxy.cfg -st `cat /home/tank/haproxy/conf/haproxy.pid`

停止服务：

killall haproxy

7.打开监控页面（启动容器的时候添加了端口映射，宿主机的7082映射到了HAProxy）

http://192.168.1.100:7082/stats



http://192.168.1.100:7082/stats

Statistics Report for HAProxy

HAProxy

Statistics Report for pid 2001

> General process information

pid = 2001 (process #1, nbproc = 1)
uptime = 0d 10h26m35s
system limits: memmax = unlimited; ulimit-n = 819200
maxsock = 8204; **maxconn =** 4096; **maxpipes =** 0
current conns = 1; current pipes = 0/0; conn rate = 1/sec
Running tasks: 1/4; idle = 100 %

active UP	backup UP
active UP, going down	backup UP, going down
active DOWN, going up	backup DOWN, going up
active or backup DOWN	not checked
active or backup DOWN for maintenance (MAINT)	
active or backup SOFT STOPPED for maintenance	

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Disable refresh
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.5)
- Online manual

admin_stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				1	3	-	1	3	10	33			9 798	7 516	0	0	6					OPEN								
Backend	0	0		0	2		0	1	1	8	0	0s	9 798	7 516	0	0		8	0	0	0	10h26m UP		0	0	0		0		

server_haproxy

		Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server							
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
	Frontend				0	6	-	0	8	4 096	128			129 693	1 450 115	0	0	0					OPEN								
☐	web1	0	0	-	0	3		0	4	-	64	64	11m34s	63 494	1 079 062		0		0	0	0	0	*no check*		1	Y	-				-
☐	web2	0	0	-	0	3		0	4	-	64	64	11m34s	66 199	371 053		0		0	0	0	0	*no check*		1	Y	-				-
	Backend	0	0		0	6		0	8	410	128	128	11m34s	129 693	1 450 115	0	0		0	0	0	0	10h26m UP		2	2	0		0	0s	

Choose the action to perform on the checked servers : Apply

二、配置tomcat1和tomcat2节点

1.安装java运行环境

网盘链接：

链接：https://pan.baidu.com/s/1IpfCiGdIj3HWS-9M3_YfLA
提取码：bync
复制这段内容后打开百度网盘手机App，操作更方便哦--来自百度网盘超级会员V3的分享

1.1解压缩文件

```
tar -zxvf jdk1.8.0_162.tar.gz -C /usr/local/java/
```

1.2.向/etc/profile文件中追加下面内容：

export JAVA_HOME=/usr/local/java/jdk1.8.0_162

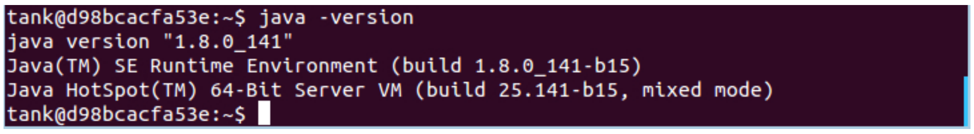
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
export PATH=$PATH:${JAVA_HOME}/bin

1.3.让文件生效

```
source /etc/profile
```

1.4.验证java成功安装

```
java -version
```



```
tank@d98bcacfa53e:~$ java -version
java version "1.8.0_141"
Java(TM) SE Runtime Environment (build 1.8.0_141-b15)
Java HotSpot(TM) 64-Bit Server VM (build 25.141-b15, mixed mode)
tank@d98bcacfa53e:~$
```

2.安装tomcat

到tomcat官网下载9.0.44版本的tomcat安装包

https://tomcat.apache.org/

进入下载好的Tomcat压缩包地址路径，解压Tomcat至/usr/local/目录中。

```
tar -zxvf apache-tomcat-9.0.44.tar.gz -C /usr/local/
cd /usr/local/apache-tomcat-9.0.44
```

进入Tomcat安装目录。命令启动（默认绿色后缀为.sh的便是Linux的可执行脚本）

```
cd /usr/local/apache-tomcat-9.0.43/bin
./startup.sh //开启
./shutdown.sh //关闭
```

说明：在window系统中启动脚本是.bat文件，在Linux系统中使用的是.sh文件。执行格式为： ./脚本
注意：如果.sh文件显示为灰色，且无法执行，则是因为权限不足，使用命令给脚本文件增加执行权限。

```
chmod +x *.sh     #给所有脚本文件增加执行权限
```

https://blog.csdn.net/wangyonglin1123/article/details/50986524/ tomcat调优

下载并配置JDBC MySQL驱动
http://dev.mysql.com/downloads/connector/j/
解压后复制mysql-connector-java-5.1.13-bin.jar到此路径下（目录不存在则自己创建）：/usr/local/apache-tomcat-6.0.26/webapps/servlet/WEB-INF/lib
2.环境变量设置：

```
vim /etc/profile
```

根据自己安装软件的路径在/etc/profile文件末尾添加：
export JAVA_HOME=/usr/local/java/jdk1.8.0_162
export CATALINA_HOME=/usr/local/apache-tomcat-9.0.44
export PATH=$JAVA_HOME/bin:$CATALINA_HOME/bin:$PATH
export
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$CATALINA_HOME/lib/servlet-

api.jar: $CATALINA_HOME/webapps/servlet/WEB-INF/lib/mysql-connector-java-5.1.13-bin.jar$CLASSPATH

让环境变量生效

```
source /etc/profile
```

三、配置amoeba节点

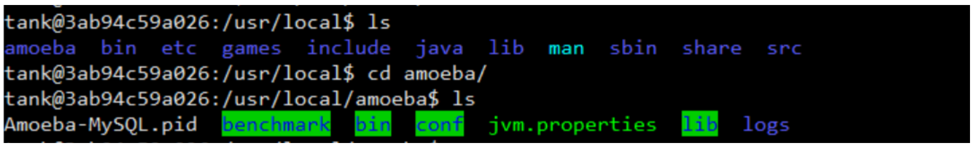
1.安装java环境

略

2、安装Amoeba

下载链接：https://jaist.dl.sourceforge.net/project/amoeba/Amoeba%20for%20mysql/3.x/amoeba-mysql-3.0.5-RC-distribution.zip

下载的是amoeba-mysql-3.0.5-RC-distribution.zip。Amoeba安装非常简单，直接解压即可使用，这里将Amoeba解压到/usr/local/amoeba目录下，这样就安装完成了



```
tank@3ab94c59a026:/usr/local$ ls
amoeba  bin  etc  games  include  java  lib  man  sbin  share  src
tank@3ab94c59a026:/usr/local$ cd amoeba/
tank@3ab94c59a026:/usr/local/amoeba$ ls
Amoeba-MySQL.pid  benchmark  bin  conf  jvm.properties  lib  logs
```

3. 配置Amoeba

Amoeba的配置文件在本环境下位于/usr/local/amoeba/conf目录下。配置文件比较多，但是仅仅使用读写分离功能，只需配置两个文件即可，分别是dbServers.xml和amoeba.xml
如果需要配置ip访问控制，还需要修改access_list.conf文件，下面首先介绍dbServers.xml
cat conf/dbServers.xml

```
<?xml version="1.0" encoding="gbk"?>

<!DOCTYPE amoeba:dbServers SYSTEM "dbserver.dtd">
<amoeba:dbServers xmlns:amoeba="http://amoeba.meidusa.com/">

 <!--
 Each dbServer needs to be configured into a Pool,
 If you need to configure multiple dbServer with load balancing that can be simplified by the following configuration:
 add attribute with name virtual = "true" in dbServer, but the configuration does not allow the element with name factoryConfig
  such as 'multiPool' dbServer
  -->
  <dbServer name="abstractServer" abstractive="true">
  <factoryConfig class="com.meidusa.amoeba.mysql.net.MysqlServerConnectionFactory">
  <property name="connectionManager">${defaultManager}</property>
  <property name="sendBufferSize">64</property>
  <property name="receiveBufferSize">128</property>
  <!-- mysql port -->
  <property name="port">3306</property>    #设置Amoeba要连接的mysql数据库的端口，默认是3306
  <!-- mysql schema -->
  <property name="schema">tpcw</property>    #设置缺省的数据库，当连接amoeba时，操作表必须显式的指定数据库名，即采用dbname.tablename的方式，不支持 use dbname指定缺省库，因为操作会调度到各个后端dbserver
```

```

<!-- mysql user -->
<property name="user">root</property>    #设置amoeba连接后端数据库服务器的账号和密码，因此需要在所有后端数据库上创建该用户，并授权amoeba服务器可连接

<property name="password">root</property>
</factoryConfig>
<poolConfig class="com.meidusa.toolkit.common.poolable.PoolableObjectPool">
<property name="maxActive">500</property>    #最大连接数，默认500
<property name="maxIdle">500</property>        #最大空闲连接数
<property name="minIdle">1</property>        #最新空闲连接数
<property name="minEvictableIdleTimeMillis">600000</property>
<property name="timeBetweenEvictionRunsMillis">600000</property>
<property name="testOnBorrow">true</property>
<property name="testOnReturn">true</property>
<property name="testWhileIdle">true</property>
</poolConfig>
</dbServer>
<dbServer name="mysql1" parent="abstractServer">    #设置一个后端可写的dbServer，这里定义为writedb，这个名字可以任意命名，后面还会用到
<factoryConfig>
<!-- mysql ip -->
<property name="ipAddress">172.20.0.15</property> #设置后端可写dbserver
</factoryConfig>
</dbServer>
<dbServer name="mysql2" parent="abstractServer">    #设置后端可读dbserver
<factoryConfig>
<!-- mysql ip -->
<property name="ipAddress">172.20.0.16</property>
</factoryConfig>
</dbServer>
<dbServer name="defaultPool" virtual="true">    #设置定义一个虚拟的dbserver，实际上相当于一个dbserver组，这里将可读的数据库ip统一放到一个组中，将这个组的名字命名为myslave
<poolConfig class="com.meidusa.amoeba.server.MultipleServerPool">
<!-- Load balancing strategy: 1=ROUNDROBIN , 2=WEIGHTBASED , 3=HA-->
<property name="loadbalance">1</property>    #选择调度算法，1表示复制均衡，2表示权重，3表示HA， 这里选择1
<!-- Separated by commas,such as: server1,server2,server1 -->
<property name="poolNames">mysql1</property>    #myslave组成员
<property name="poolNames">mysql2</property>    #myslave组成员
</poolConfig>
</dbServer>
</amoeba:dbServers>
```

下面首先介绍amoeba.xml

cat conf/amoeba.xml

```
<?xml version="1.0" encoding="gbk"?>

<!DOCTYPE amoeba:configuration SYSTEM "amoeba.dtd">
<amoeba:configuration xmlns:amoeba="http://amoeba.meidusa.com/">

 <proxy>
```

```

<!-- service class must implements com.meidusa.amoeba.service.Service -->
<service name="Amoeba for Mysql" class="com.meidusa.amoeba.mysql.server.MySQLService">
 <!-- port -->
 <property name="port">8066</property>       #设置amoeba监听的端口，默认是8066

 <!-- bind ipAddress -->       #下面配置监听的接口，如果不设置，默认监听所以的IP
 <!--
 <property name="ipAddress">127.0.0.1</property>
 -->

 <property name="connectionFactory">
 <bean class="com.meidusa.amoeba.mysql.net.MysqlClientConnectionFactory">
 <property name="sendBufferSize">128</property>
 <property name="receiveBufferSize">64</property>
 </bean>
 </property>

 <property name="authenticateProvider">
 <bean class="com.meidusa.amoeba.mysql.server.MysqlClientAuthenticator">


# 提供客户端连接amoeba时需要使用这里设定的账号 (这里的账号密码和amoeba连接后端数据库服务器的密码无关)

 <property name="user">tank</property>


 <property name="password">tank</property>

 <property name="filter">
 <bean class="com.meidusa.toolkit.net.authenticate.server.IPAccessController">
 <property name="ipFile">${amoeba.home}/conf/access_list.conf</property>
 </bean>
 </property>
 </bean>
 </property>

 </service>

 <runtime class="com.meidusa.amoeba.mysql.context.MysqlRuntimeContext">

 <!-- proxy server client process thread size -->
 <property name="executeThreadSize">128</property>

 <!-- per connection cache prepared statement size -->
 <property name="statementCacheSize">500</property>

 <!-- default charset -->
 <property name="serverCharset">utf8</property>

 <!-- query timeout( default: 60 second , TimeUnit:second) -->
 <property name="queryTimeout">60</property>
 </runtime>
```

```

</proxy>

<!--
Each ConnectionManager will start as thread
manager responsible for the Connection IO read , Death Detection
-->
<connectionManagerList>
<connectionManager name="defaultManager" class="com.meidusa.toolkit.net.MultiConnectionManagerWr
apper">
<property name="subManagerClassName">com.meidusa.toolkit.net.AuthingableConnectionManager</prope
rty>
</connectionManager>
</connectionManagerList>

<!-- default using file loader -->
<dbServerLoader class="com.meidusa.amoeba.context.DBServerConfigFileLoader">
<property name="configFile">${amoeba.home}/conf/dbServers.xml</property>
</dbServerLoader>

<queryRouter class="com.meidusa.amoeba.mysql.parser.MysqlQueryRouter">
<property name="ruleLoader">
<bean class="com.meidusa.amoeba.route.TableRuleFileLoader">
<property name="ruleFile">${amoeba.home}/conf/rule.xml</property>
<property name="functionFile">${amoeba.home}/conf/ruleFunctionMap.xml</property>
</bean>

</property>
<property name="sqlFunctionFile">${amoeba.home}/conf/functionMap.xml</property>
<property name="LRUMapSize">1500</property>
<property name="defaultPool">defaultPool</property>    #设置amoeba默认的池，这里设置为writedb

<property name="needParse">true</property>
</queryRouter>
</amoeba:configuration>
```

4.修改jvm

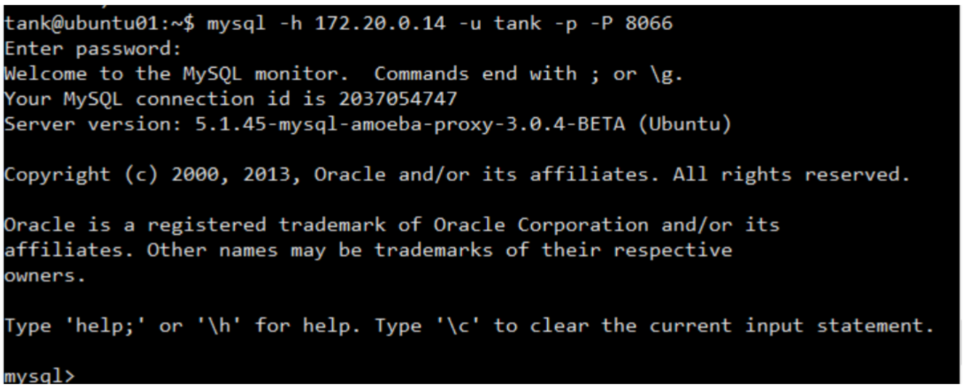
vi /usr/local/amoeba/jvm.properties

原为：JVM_OPTIONS="-server -Xms256m -Xmx1024m -Xss196k -XX:PermSize=16m -XX:MaxPermSize=96m"

改成：JVM_OPTIONS="-server -Xms1024m -Xmx1024m -Xss256k -XX:PermSize=16m -XX:MaxPermSize=96m"

5.启动amoeba

/usr/local/amoeba/bin/launcher



```
tank@ubuntu01:~$ mysql -h 172.20.0.14 -u tank -p -P 8066
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2037054747
Server version: 5.1.45-mysql-amoeba-proxy-3.0.4-BETA (Ubuntu)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

四、配置mysql1和mysql2节点

1.安装mysql环境

$ sudo apt-get install mysql-server

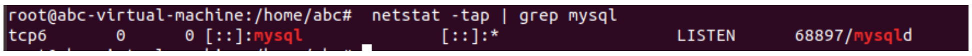
$ sudo apt isntall mysql-client

$ sudo apt install libmysqlclient-dev

检查是否安装成功

$ netstat -tap | grep mysql

若安装成功会有如下输出



```
root@abc-virtual-machine:/home/abc#  netstat -tap | grep mysql
tcp6       0      0 [::]:mysql            [::]:*                  LISTEN      68897/mysqld
```

2.设置远程访问

编辑MySQL的配置文件

$ vim /etc/mysql/mysql.conf.d/mysqld.cnf

把下面的内容注释

bind-address   = 127.0.0.1

以root权限进入MySQL命令行，执行开启权限命令，本示例中MySQL中用户与密码皆为root

grant all on *.* to root@'%' identified by 'root' with grant option;

flush privileges;

重启MySQL

$ sudo /etc/init.d/mysql restart

五、编译javaweb工程，生成benchmark文件

1.下载TPC-W（Java版）

http://pharm.ece.wisc.edu/tpcw/tpcw.tar.gz

解压缩后就是一个文件夹tpcw1.0

2.修改部分源码：

2.1修改tpcw1.0\populate\populate_images

```
#!/usr/local/bin/perl  ------> #!/usr/bin/perl
$DEST_DIR="/local_home/cain/Images";  ------> $DEST_DIR="/usr/local/apache-tomcat-6.0.26/webapps/t
pcw/Images";
```

2.2修改tpcw1.0\populate\TPCW_Populate.java

```
private static final String driverName = "com.mysql.jdbc.Driver";//"COM.ibm.db2.jdbc.app.DB2Drive
r";
private static final String dbName = "jdbc:mysql://localhost:3306/tpcw2";//"jdbc:db2:tpcw2";
[java] view plain copy
PreparedStatement statement = con.prepareStatement
        ("INSERT INTO address(ADDR_ID,ADDR_STREET1,ADDR_STREET2,ADDR_CITY,ADDR_STATE,ADDR_ZIP,ADDR
_CO_ID) VALUES (?, ?, ?, ?, ?, ?, ?)");
//mysql是安装在linux上的，所以有大小写之分，解决com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorExcept
ion: Table 'tpcw2.ADDRESS'
[java] view plain copy
try {
        Class.forName(driverName);
         con = DriverManager.getConnection(dbName, "root", "");//(dbName);
          con.setAutoCommit(false);//解决java.sql.SQLException: Can't call commit when autocommit=
true
```

2.3修改tpcw1.0\servlets\TPCW_Database.java

```
[java] view plain copy
 static String driverName = "com.mysql.jdbc.Driver";//"COM.ibm.db2.jdbc.app.DB2Driver";
 static String jdbcPath = "jdbc:mysql://localhost:3306/tpcw2";//"jdbc:db2:tpcw2";
[java] view plain copy
private static final boolean use_connection_pool = false; //true;
[java] view plain copy
```

```
try {
         Class.forName(driverName).newInstance();//Class.forName(driverName);
        // Class.forName("postgresql.Driver");

        // Create URL for specifying a DBMS
        Connection con;
        while(true) {
        try {
            //   con = DriverManager.getConnection("jdbc:postgresql://eli.ece.wisc.edu/tpcw", "mil
o", "");
             con = DriverManager.getConnection(jdbcPath, "root", "root");//con = DriverManager.get
Connection(jdbcPath);
             break;
```

2.4修改tpcw1.0\servlets\TPCW_Util.java

```
public static final String SESSION_ID="jsessionid="; //"$sessionid{1}quot;;
```

2.5修改tpcw1.0\rbe\RBE.java

```
public static String www1 = "http://localhost:8080/";//"http://ironsides.cs.wisc.edu:8001/";
```

```

  new StrStrPattern(";jsessionid="); //(";$sessionid{1}quot;);

public static final String field_sessionID = ";jsessionid="; //";$sessionid{1}quot;;

```

3.在MySQL中创建数据库tpcw

# service mysqld start

启动失败 Failed to connect to socket /com/ubuntu/upstart: Connection refused

chown -R mysql:mysql /var/lib/mysql

# mysql

> CREATE DATABASE tpcw;

5.开始安装TPC-W：

mkdir -p ${CATALINA_HOME}/webapps/tpcw/Images

mkdir -p ${CATALINA_HOME}/webapps/servlet/WEB-INF/classes

6.在数据库tpcw中生成数据

cd populate

javac TPCW_Populate.java

java TPCW_Populate

cp TPCW_Populate.class ${CATALINA_HOME}/webapps/servlet/WEB-INF/classes

7.生成并部署图片

cd ../ImgGen/ImgFiles

make

cd ..

cd ../populate

perl populate_images

cp ../images/* ${CATALINA_HOME}/webapps/tpcw/Images

8.编译并部署servlets

cd ../servlets

javac *.java

cp *.class ${CATALINA_HOME}/webapps/servlet/WEB-INF/classes

vi ${CATALINA_HOME}/webapps/servlet/WEB-INF/web.xml

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>

<web-app version="2.5"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_
2_5.xsd">

```

```xml
<display-name>TPC-W</display-name>
 <description>
    TPC-W Java Implementation
  </description>

  <servlet>
    <servlet-name>TPCW_home_interaction</servlet-name>
    <servlet-class>TPCW_home_interaction</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>TPCW_shopping_cart_interaction</servlet-name>
    <servlet-class>TPCW_shopping_cart_interaction</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>TPCW_order_inquiry_servlet</servlet-name>
    <servlet-class>TPCW_order_inquiry_servlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>TPCW_order_display_servlet</servlet-name>
    <servlet-class>TPCW_order_display_servlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>TPCW_search_request_servlet</servlet-name>
    <servlet-class>TPCW_search_request_servlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>TPCW_execute_search</servlet-name>
    <servlet-class>TPCW_execute_search</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>TPCW_new_products_servlet</servlet-name>
    <servlet-class>TPCW_new_products_servlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>TPCW_best_sellers_servlet</servlet-name>
    <servlet-class>TPCW_best_sellers_servlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>TPCW_product_detail_servlet</servlet-name>
    <servlet-class>TPCW_product_detail_servlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>TPCW_customer_registration_servlet</servlet-name>
    <servlet-class>TPCW_customer_registration_servlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>TPCW_buy_request_servlet</servlet-name>
    <servlet-class>TPCW_buy_request_servlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>TPCW_buy_confirm_servlet</servlet-name>
    <servlet-class>TPCW_buy_confirm_servlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>TPCW_admin_request_servlet</servlet-name>
    <servlet-class>TPCW_admin_request_servlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>TPCW_admin_response_servlet</servlet-name>
    <servlet-class>TPCW_admin_response_servlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>TPCW_home_interaction</servlet-name>
    <url-pattern>/TPCW_home_interaction</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TPCW_shopping_cart_interaction</servlet-name>
    <url-pattern>/TPCW_shopping_cart_interaction</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TPCW_order_inquiry_servlet</servlet-name>
    <url-pattern>/TPCW_order_inquiry_servlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TPCW_order_display_servlet</servlet-name>
    <url-pattern>/TPCW_order_display_servlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TPCW_search_request_servlet</servlet-name>
    <url-pattern>/TPCW_search_request_servlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TPCW_execute_search</servlet-name>
    <url-pattern>/TPCW_execute_search</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TPCW_new_products_servlet</servlet-name>
    <url-pattern>/TPCW_new_products_servlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TPCW_best_sellers_servlet</servlet-name>
    <url-pattern>/TPCW_best_sellers_servlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TPCW_product_detail_servlet</servlet-name>
    <url-pattern>/TPCW_product_detail_servlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TPCW_customer_registration_servlet</servlet-name>
    <url-pattern>/TPCW_customer_registration_servlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TPCW_buy_request_servlet</servlet-name>
    <url-pattern>/TPCW_buy_request_servlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TPCW_buy_confirm_servlet</servlet-name>
    <url-pattern>/TPCW_buy_confirm_servlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TPCW_admin_request_servlet</servlet-name>
    <url-pattern>/TPCW_admin_request_servlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>TPCW_admin_response_servlet</servlet-name>
    <url-pattern>/TPCW_admin_response_servlet</url-pattern>
  </servlet-mapping>

</web-app>
```

9.编译RBE

cd rbe

cd util

javac *.java

cd ../args

mkdir -p rbe/util

cp ../util/*.class rbe/util/

mkdir rbe/args

javac *.java

cp *.class rbe/args/

mv rbe ../

cd ..

javac *.java

(注：由于直接编译会出现错误，所以这里要修改部分函数名

tpcw1.0/rbe/util/Debug.java中

public class Debug { public static void assert(boolean assertCond, String message) ...

把函数名assert改成你自己想要的名字, 如assert1

接着以下三个文件中所有出现的Debug.assert中的assert都改成你自己修改函数名, 如Debug.assert1

tpcw1.0/rbe/EB.java

tpcw1.0/rbe/util/CharSetStrPattern.java

tpcw1.0/rbe/util/Histogram.java)

...

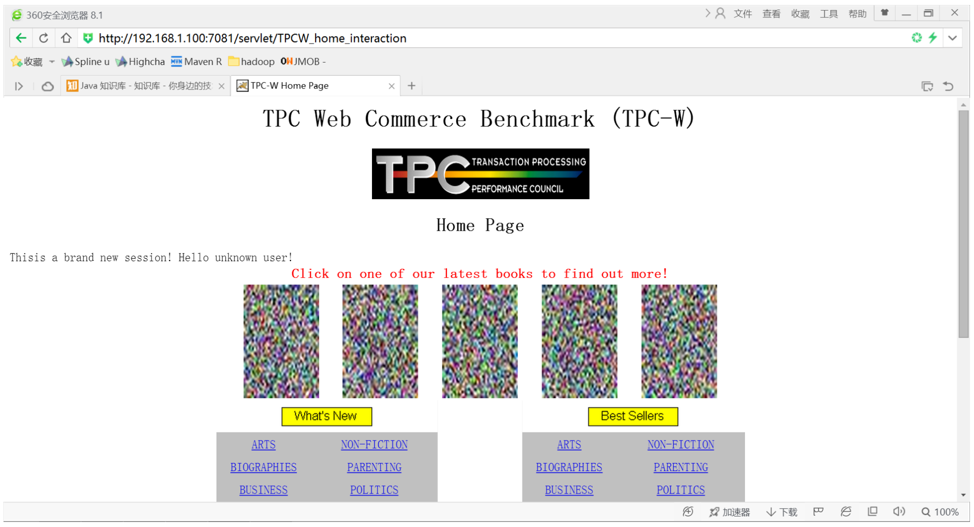
也可以在eclipse中建立工程，编译文件

10.测试TPC-W

启动tomcat1,tomcat2

./startup.sh

访问http://192.168.1.100:7081/servlet/TPCW_home_interaction页面测试是否安装成功



TPC Web Commerce Benchmark (TPC-W)

Home Page

Thisis a brand new session! Hello unknown user!

Click on one of our latest books to find out more!

What's New

ARTS	NON-FICTION
BIOGRAPHIES	PARENTING
BUSINESS	POLITICS

Best Sellers

ARTS	NON-FICTION
BIOGRAPHIES	PARENTING
BUSINESS	POLITICS

六、安装docker

```
lyz@ubuntu:~$ sudo su
[sudo] password for lyz:
root@ubuntu:/home/lyz# apt-get update

root@ubuntu:/home/lyz# apt-get install \
> apt-transport-https \
> ca-certificates \
> curl \
> gnupg \
> lsb-release
```

E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)

E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?

root@ubuntu:/home/lyz# apt-get install apt-transport-https ca-certificates curl gnupg lsb-release

E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)

E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?

```
root@ubuntu:/home/lyz# sudo rm /var/cache/apt/archives/lock
root@ubuntu:/home/lyz# sudo rm /var/lib/dpkg/lock
root@ubuntu:/home/lyz# apt-get install apt-transport-https ca-certificates curl gnupg lsb-release

root@ubuntu:/home/lyz# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmo
r -o /usr/share/keyrings/docker-archive-keyring.gpg
root@ubuntu:/home/lyz# echo \
> "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.dock
er.com/linux/ubuntu \
> $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
root@ubuntu:/home/lyz# apt-get update
root@ubuntu:/home/lyz# apt-get install docker-ce docker-ce-cli containerd.io

```

```
root@ubuntu:/home/lyz# apt-cache madison docker-ce
```

```
 docker-ce | 5:20.10.6~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:20.10.5~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:20.10.4~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:20.10.3~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:20.10.2~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:20.10.1~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:20.10.0~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.15~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.14~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.13~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.12~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.11~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.10~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.9~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.8~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.7~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.6~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.5~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.4~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.3~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.2~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
 docker-ce | 5:19.03.1~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
```

docker-ce | 5:19.03.0~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 5:18.09.9~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 5:18.09.8~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 5:18.09.7~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 5:18.09.6~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 5:18.09.5~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 5:18.09.4~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 5:18.09.3~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 5:18.09.2~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 5:18.09.1~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 5:18.09.0~3-0~ubuntu-bionic | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 18.06.3~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 18.06.2~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 18.06.1~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 18.06.0~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
docker-ce | 18.03.1~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages

```
root@ubuntu:/home/lyz# apt-get install docker-ce=5:18.09.0~3-0~ubuntu-bionic docker-ce-cli=5:18.09.0~3-0~ubuntu-bionic containerd.io

```

```
root@ubuntu:/home/lyz# docker run hello-world
```

```
root@ubuntu:/home/lyz# vim /etc/docker/daemon.json
```

在里面添加一句话配置docker 加速

```
{
  "registry-mirrors": ["https://registry.docker-cn.com"]
}
```

```
root@ubuntu:/home/lyz# systemctl daemon-reload
root@ubuntu:/home/lyz# systemctl restart docker
```

注：配置过程中的一些docker命令

172.20.0.11	HAProxy
172.20.0.12	tomcat1
172.20.0.13	tomcat2
172.20.0.14	amoeba
172.20.0.15	mysql1
172.20.0.16	mysql2
172.20.0.17	tomcat3
172.20.0.18	mysql3

docker network create --subnet=172.20.0.0/16 webserver_network

docker run -itd --name HAProxy **--net webserver_network --ip 172.20.0.11 -p 7080:80 -p 7081:8080 -p 7082:1080 -p 7083:5222 ubuntu-self /bin/bash**

docker run -itd --name tomcat1 **--net webserver_network --ip 172.20.0.12 ubuntu-self /bin/bash**

docker run -itd --name tomcat2 **--net webserver_network --ip 172.20.0.13 ubuntu-self /bin/bash**

docker run -itd --name amoeba **--net webserver_network --ip 172.20.0.14 ubuntu-self /bin/bash**

docker run -itd --name mysql1 **--net webserver_network --ip 172.20.0.15 ubuntu-self /bin/bash**

docker run -itd --name mysql2 **--net webserver_network --ip 172.20.0.16 ubuntu-self /bin/bash**

docker run -itd --name tomcat3 **--net webserver_network --ip 172.20.0.17 -p 7084:8080 ubuntu-self-tomcat /bin/bash**

docker run -itd --name mysql3 **--net webserver_network --ip 172.20.0.18 -p 7085:3306 ubuntu-self-mysql /bin/bash**

apt-get install openssh-server

service ssh start

http://blog.csdn.net/zhu_tianwei/article/details/411173233

http://192.168.1.128:1:8080/servlet/TPCW_home_interaction

docker update Tomcat1 Tomcat2 Tomcat3 Tomcat4 Tomcat5 Tomcat6 Tomcat7 Tomcat8 Tomcat9 Tomcat10 Tomcat11 Tomcat12 Haproxy BeTask1 --cpuset-cpus=0-9,40-49