目 录

1 运行环境需求和开发语言

2 用户类型

2.1 管理员用户 2

2.2 普通用户 2

3 系统结构

3.1 存储结构 3

3.1.1 login\_image 3

3.1.2 login\_workspace 3

3.1.3 manager\_image 4

3.1.4 manager\_workspace 4

3.1.5 user\_image 5

3.1.6 user\_workspace 5

3.1.7 sql\_workspace 6

3.2 系统架构 6

3.2.1 四个镜像 6

3.2.2 四类容器 6

3.3 容器运行流程图 8

4 系统设计

4.1 系统环境的安装与配置 9

4.1.1安装虚拟机 9

4.1.2安装docker并配镜像加速器 9

4.1.3开启远程管理接口 9

4.2 登录界面 9

4.3 注册界面 10

4.4 用户主界面 10

4.5 镜像管理 11

4.5.1 镜像列表 11

4.5.2 Remove 11

4.5.3 Remove All 12

4.5.4 Create A Container 13

4.6 容器管理 14

4.6.1 容器列表 14

4.6.2 Start 14

4.6.3 Restart 16

4.6.4 Stop 18

4.6.5 Remove 20

4.6.6 Remove All 22

总结

附录

1 运行环境需求和开发语言

本系统在任何服务器或者虚拟机上都可以使用，只需要装有Docker服务即可。开发本系统的计算机配置如下所示。

处理器：酷睿i5 。

内存：8G 。

操作系统：Windows 10。

开发平台：VMware 9.0.1。

虚拟机操作系统：CentOS 7.6.1810。

Docker版本：18.09.6。

开发语言：html、php、js、css。

2 用户类型

云平台系统具有两种用户类型，分别为管理员用户及普通用户，管理员用户负责对镜像及容器进行管理，普通用户具有使用容器的权限。

2.1 管理员用户

Manager用户负责管理镜像与容器，具有删除镜像，创建容器，启动容器，重启容器，停止容器，删除一个已停止的容器，删除全部停止的容器的权限。并且可以通过SSH远程登陆。如图2.1所示。

2.2 普通用户

普通用户可以是老师或者学生，通过注册拥有自己的一个容器，并且能够在容器中操作。该类用户同样可以通过SSH服务远程登录。如图2.1所示。

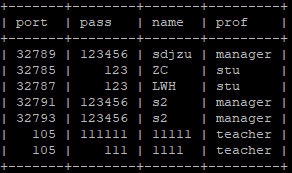


图2.1 docker\_user数据表

3 系统结构

本节从存储结构、系统架构及系统运行流程图三个方面对系统结构进行介绍。

3.1 存储结构

系统的存储结构体现了系统文件在虚拟机中的分布及存储情况。文件关键代码补充在附录中。

3.1.1 login\_image

该文件夹存储登录部分的镜像文件，以构造login镜像，包含Dockerfile、run.sh。关键代码见附录1。

(1) Dockerfile

负责构造login镜像，镜像以Centos镜像为基础，容器工作在my\_httpd目录下，安装了httpd镜像、php镜像以及php\_mysqlnd镜像，并且配置了连接数据库的环境变量，容器暴露80端口，并且以CMD命令启动run.sh脚本。

(2) run.sh

启动脚本包含容器启动时所要执行的命令，负责启动httpd服务并且防止容器休眠。

3.1.2 login\_workspace

该文件夹存储登录部分的前台与后台文件，是在创建login容器时所要挂载的卷。包含index.php、login\_check.php、loginstyle.css、register.html、register\_check.php、registerstyle.css、showing.php。关键代码见附录2。

(1) index.html

构造登录部分的界面，负责获取用户的账号及密码，以及账户的注册。

(2) login\_check.php

接收index.html的账号及密码，然后与数据库容器连接，查询并校验账号密码的正确性，若正确则跳转进入账号对应的容器主界面，否则返回登录界面。

(3) loginstyle.css

设计index.html的界面，各组件的位置及大小等问题。

(4)register.html

提供注册界面，供用户注册使用，以获取用户的密码，令用户确认密码及输入验证码。

(5) register\_check.php

从register.html获取密码、确认密码及验证码进行校验。验证正确后为用户分配账号（主机为容器分配的端口号）及创建映射相应端口号的容器，并将账户及密码存入数据库容器；若验证失败，则返回注册界面，用户重新输入信息。

(6) registerstyle.css

设计register.html的界面，各组件的位置及大小等问题。

(7) showing.php

负责生成注册界面的验证码，以防止恶意的网络攻击。

(8) title\_logo\_g.png

这是小组自行设计的“山东建筑大学云计算平台”logo，显示在登录页面、注册页面。

3.1.3 manager\_image

该文件夹存储管理员容器的镜像文件，以构造manager镜像，包含Dockerfile、run.sh文件。关键代码见附录3。

(1) Dockerfile

负责构造manager镜像，镜像以Centos镜像为基础，容器工作在manager目录下，安装了httpd镜像、php镜像、mysql镜像、php\_mysqlnd镜像以及openssh-server镜像，并且配置了连接数据库的环境变量，以备后期开发使用，容器暴露22及80端口，并且以CMD命令启动run.sh脚本。

(2) run.sh

启动脚本包含容器启动时所要执行的命令，负责启动SSH服务及httpd服务并且防止容器休眠。

3.1.4 manager\_workspace

该文件夹存储管理员用户的前台与后台文件，是在创建manager容器时所要挂载的卷。包含index.html、image.php、containers.php、opt\_con.php、title\_logo\_g.png、js与css文件夹。关键代码见附录4。

(1) index.html

管理员用户主界面，管理员登录成功后由登录界面跳转到此界面，界面包含image.php及containers.php的链接。

(2) image.php

镜像管理界面，镜像列表显示docker所有的镜像，并提供Remove、Remove all、Create A Container按钮以供使用相关功能。

(3) containers.php

容器管理界面，容器列表显示docker所有的容器，并提供Start、Restart、Stop、Remove、Remove all按钮以供使用相关功能。

(4) opt\_con.php

响应镜像管理界面及容器管理界面操作的后台文件。获取操作类型及相关对象，通过docker远程接口实现相关操作。

(5) title\_logo\_g.png

这是小组自行设计的“山东建筑大学云计算平台”logo，显示在index.html主页面。

(6) js与css文件夹

存储界面设计及界面动作等相关文件。

3.1.5 user\_image

该文件夹存储普通用户的镜像文件，包含Dockerfile、run.sh文件。

(1) Dockerfile

负责构造manager镜像，镜像以Centos镜像为基础，容器工作在user目录下，安装了httpd镜像、php镜像以及openssh-server镜像，并且配置了连接数据库的环境变量，以备后期开发使用，容器暴露22及80端口，并且以CMD命令启动run.sh脚本。

(2) run.sh

启动脚本包含容器启动时所要执行的命令，负责启动SSH服务及httpd服务并且防止容器休眠。

3.1.6 user\_workspace

该文件夹存储普通用户的前台与后台文件，是在创建user容器时所要挂载的卷。包含index.html文件，通过创建exec实例将用户命令传到容器中执行。

index.html普通用户主界面，用户登录成功后由登录界面跳转到此界面，在此界面获取用户命令并执行。

3.1.7 sql\_workspace

存储所有用户的账号及密码，供用户登录时的查询校验及新用户数据的存储。

3.2 系统架构

本节从系统的构建角度对系统进行介绍，以展现系统的整体性。

3.2.1 四个镜像

本系统包含centos、httpd、php、php\_mysqlnd、mysql、openssh-server六个基础镜像。由这六个基础镜像通过Dockerfile构造本系统的login\_image、manager\_image、user\_image三个镜像。而最终系统以四个镜像呈现。如图3.1所示。

(1) login\_image

centos、httpd、php、php\_mysqlnd、openssh-server

centos、httpd、php、php\_mysqlnd

(2) manager\_image

centos、httpd、php、php\_mysqlnd、openssh-server

(3) user\_image

mysql

(4) mysql\_image

图3.1 四个镜像

3.2.2 四类容器

由3.2.1节中的四个镜像挂载相应数据卷以构建四类容器，其中manager与user类容器可以通过镜像建立多个容器。这四类容器构成整个云平台系统。如图3.2所示。

Login

Manager

User

Mysql

云系统

图3.2 四类容器

(1)Login容器创建命令：



图 3.3 创建login容器

(2)Manager容器创建命令：



图 3.4 创建Manager容器

(3)User容器创建命令：



图 3.5 创建User容器

(4)Mysql容器创建命令：



图 3.6 创建Mysql容器

3.3 容器运行流程图

……

Login

Mysql

User

Manager

开始

终止

……

图3.7 容器运行流程图

4 系统设计

本节将对系统实现进行详细的介绍，包含开启Docker远程接口、登录、注册、用户主界面、镜像管理、容器管理六个小节。

4.1 系统环境的安装与配置

为了能够远程访问Docker服务器，在Docker中开启远程访问接口，由此便可在其他设备远程访问Docker实现操作命令。

4.1.1安装虚拟机

首先下载并安装VMware虚拟机，然后在虚拟机上安装centos 7.6操作系统，设置虚拟机网络模式为桥接模式，方便在其它设备上访问。

4.1.2安装docker并配镜像加速器

通过老师给出的docker安装课件，在centos 7.6操作系统上安装最新版本docker，在安装完成后，配置阿里云镜像加速器，方便后续拉取镜像。

4.1.3开启远程管理接口

在装有docker的centos操作系统上，切换到/usr/lib/systemd/system/目录，利用vi命令，打开docker.service文件，并添加ExecStart内容：-H 0.0.0.0:5555表示允许任何IP通过5555端口访问docker daemon。



图4.1 开启远程管理接口

依次输入systemctl daemon-reload和systemctl restart docker.service命令，重新启动docker，加载配置文件。

输入netstat -lntp | grep dockerd命令，检查是否开启成功，出现下图所示结果表示开启成功。



图4.2 检查接口

4.2 登录界面

页面代码保存在index.html中，负责获取账号及密码，输入后点击“登录”按钮跳转到login\_check.php文件。login\_check.php连接数据库容器“mysql”对账号、密码进行验证，验证无误后跳转到用户主页并进入到用户容器。若无密码，则点击“没有账号？点我注册”跳转到注册页面进行注册。



图4.3 用户登录

4.3 注册界面

页面代码保存在register.html中，负责获取注册信息，点击“提交”按钮，跳转到register\_check.php中，对用户输入进行验证，验证无误后将数据保存在数据库中。



图4.4 用户注册

4.4 用户主界面

页面代码保存在manager\_workspace文件夹下的index.html中。

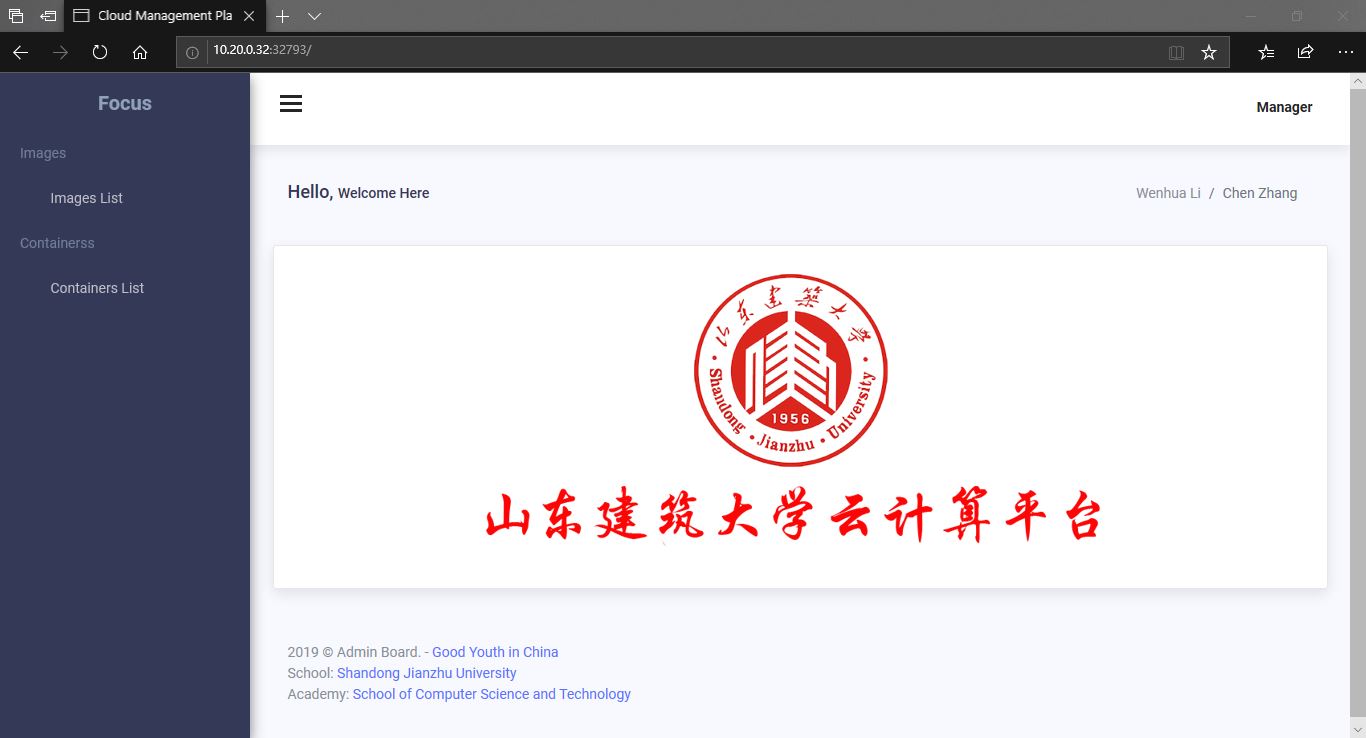


图4.5 用户主界面

4.5 镜像管理

点击左侧导航栏进入“Images List”页面，该页面代码保存在images.php文件中，负责显示docker服务中的镜像列表，并为管理员提供镜像管理功能。在页面中包含 “Remove”、“Remove All”、“Create A Container”三个镜像管理功能按钮。

4.5.1 镜像列表

调用getAllImages()，返回一个json文件，然后利用json\_decode()将json文件解析为PHP数组，该数组维度较高，要注意分析其层次关系，将多维数组内的Id,Created,Size, RepoTags,分别提取出来并存储在新的数组IdArray，CreatedArray，SizeArray，RepoTagsArray中，供HTML界面显示使用。关键代码见附录4、(1)、2)。

4.5.2 Remove

点击“Remove”按钮，程序调用prompt()函数弹出输入窗口，用户输入所要“Remove”的镜像ID，点击“确定”按钮，程序通过“window.location.href='opt\_con.php?opt=remove&imageid='+imageid;”调用opt\_con.php文件，并利用GET方式传入操作类型“opt”和镜像ID“imageid”两个参数。opt\_con.php文件执行run($opt,$ id)函数，根据“opt”参数选择对应的功能函数“removeImage($id)”完成镜像删除操作。操作成功后，跳转回images.php并刷新容器列表。如图4.6所示。

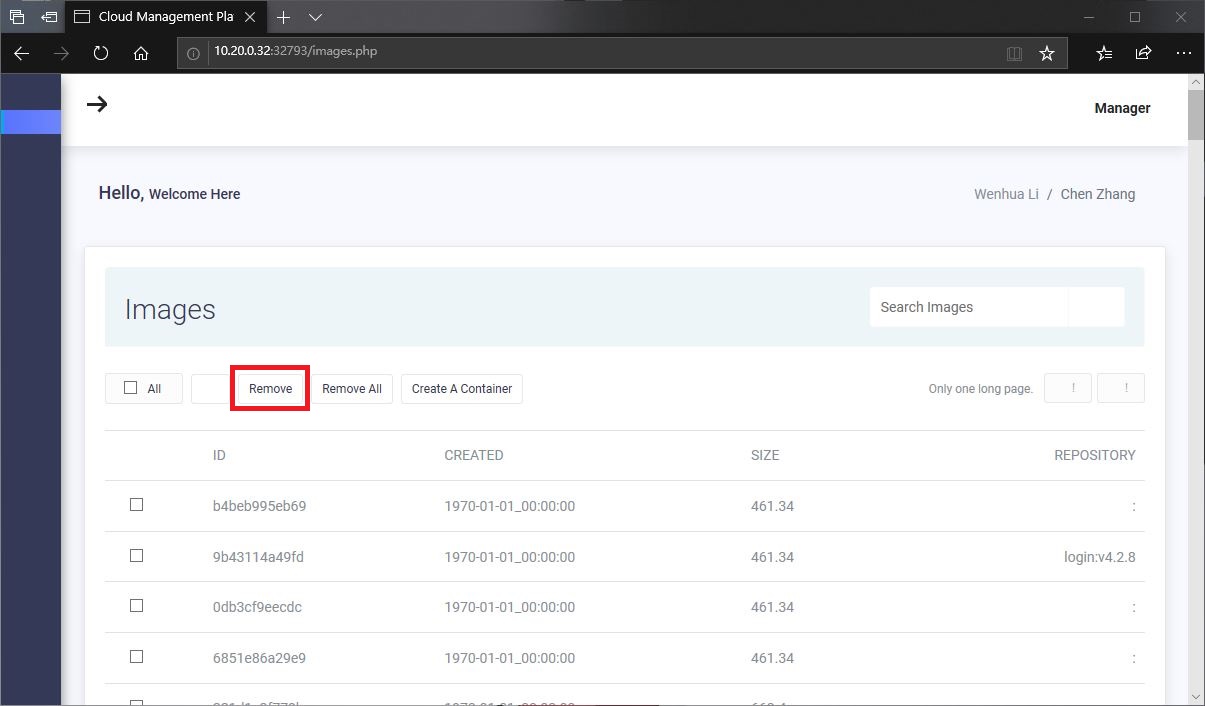


图4.6 Remove

4.5.3 Remove All

点击“Remove All”按钮，程序调用prompt()函数弹出输入窗口，用户无需输入，点击“确定”按钮，程序通过“window.location.href='opt\_con.php?opt=removeall&imageid='+imageid;”调用opt\_con.php文件，并利用GET方式传入操作类型“opt”和镜像ID“imageid”两个参数。opt\_con.php文件执行run($opt,$ id)函数，根据“opt”参数选择对应的功能函数“removeallImage($id)”完成镜像删除操作。操作成功后，跳转回images.php并刷新容器列表。如图4.7所示。

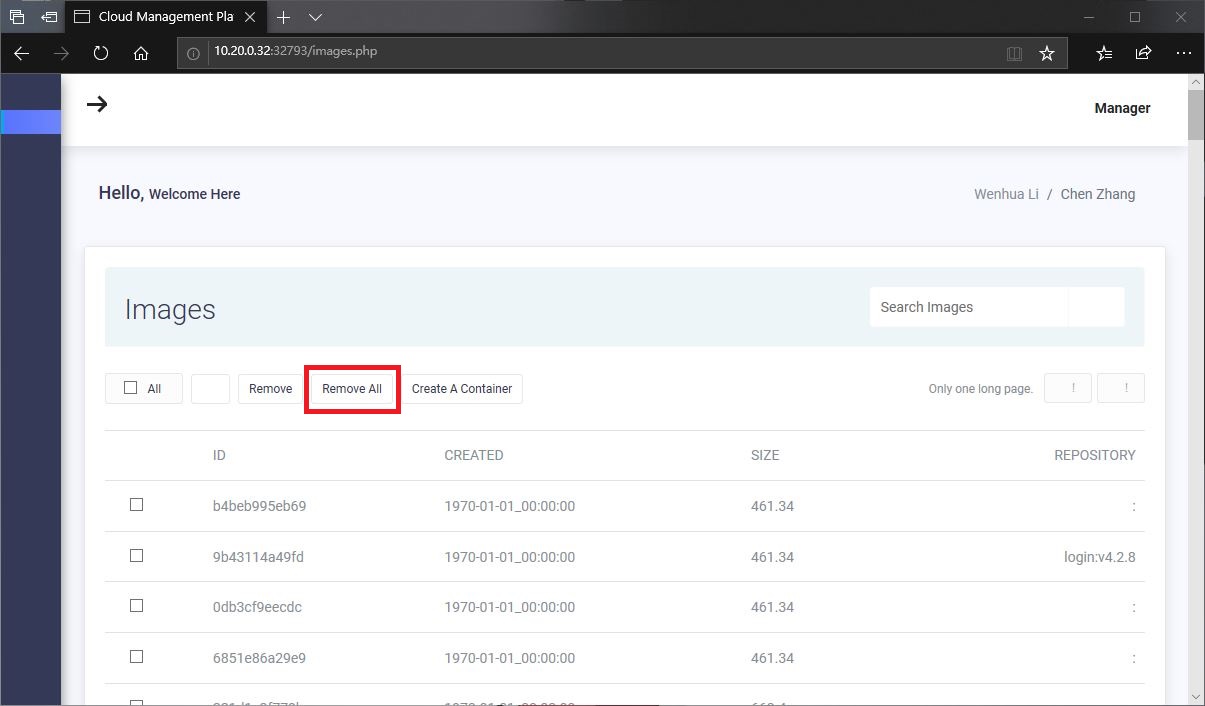


图4.7 Remove All

4.5.4 Create A Container

点击“Create A Container”按钮，程序调用prompt()函数弹出输入窗口，用户输入所要“Create A Container”的镜像ID，点击“确定”按钮，程序通过“window.location.href='opt\_con.php?opt=create&imageid='+imageid;”调用opt\_con.php文件，并利用GET方式传入操作类型“opt”和镜像ID“imageid”两个参数。opt\_con.php文件执行run($opt,$ id)函数，根据“opt”参数选择对应的功能函数“createImage($id)”完成容器创建操作。操作成功后，跳转回images.php并刷新容器列表。如图4.8所示。

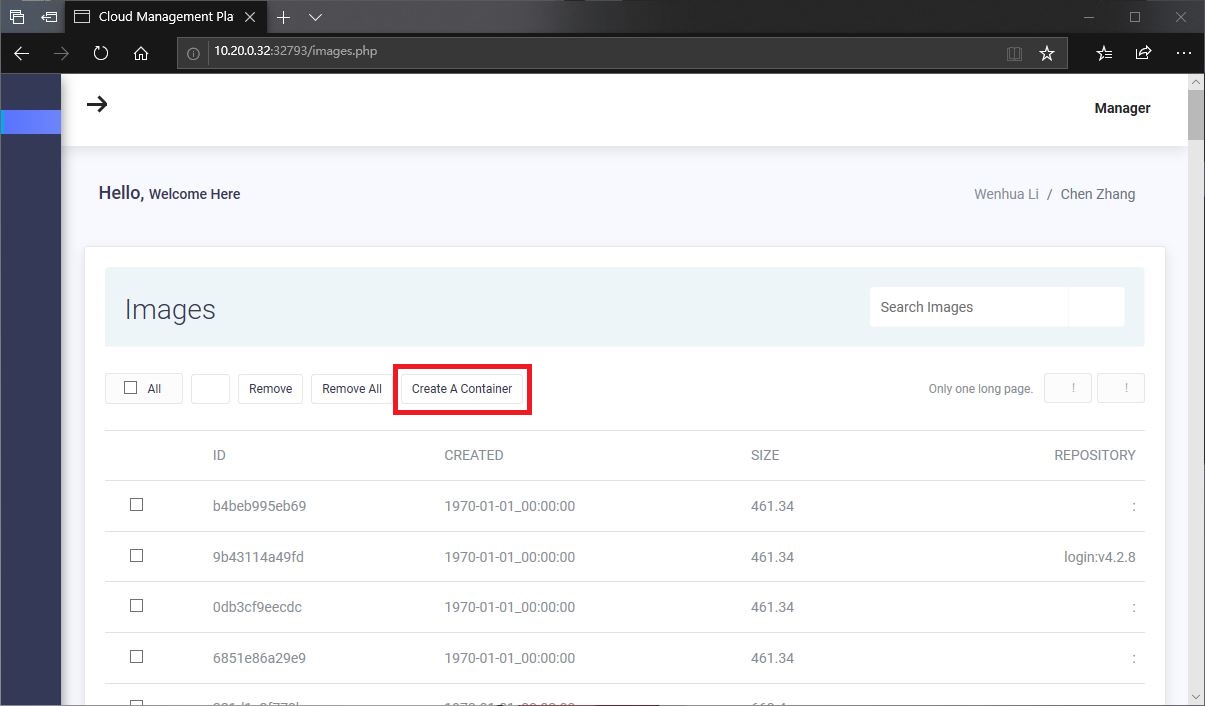


图4.8 Create A Container

4.6 容器管理

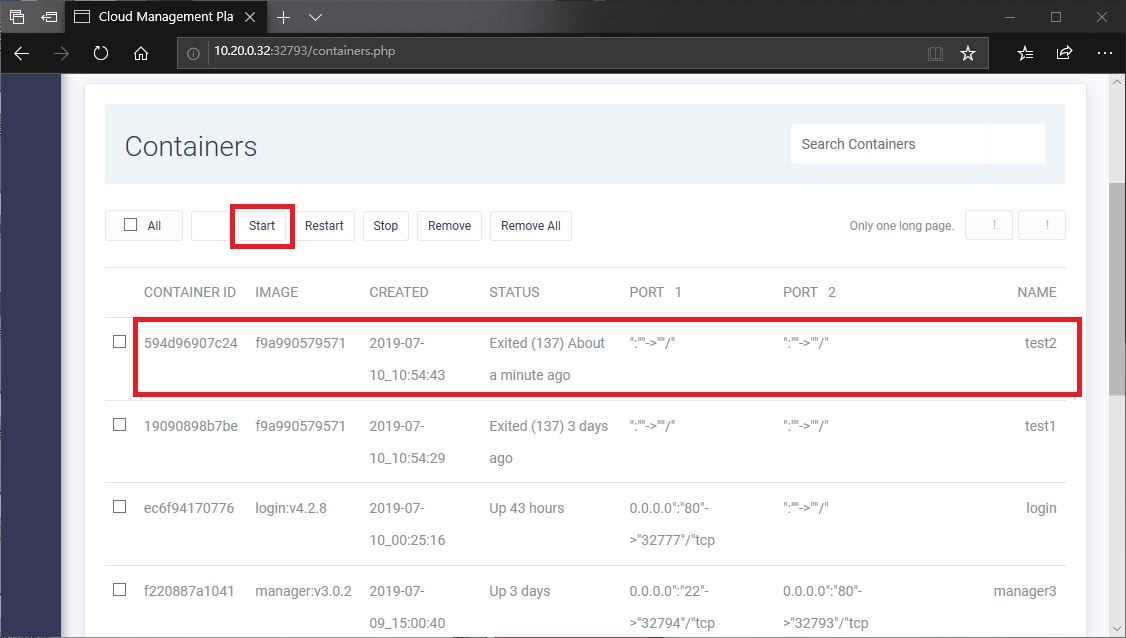
点击左侧导航栏进入“Containers List”页面，该页面代码保存在containers.php文件中，负责显示docker服务中的容器列表，并为管理员提供容器管理功能。在页面中包含“Start”、“Restart”、“Stop”、“Remove”、“Remove All”五个容器管理功能按钮。

4.6.1 容器列表

调用getAllContainers()，该函数返回的PHP数组，维度更高，分析难度更大，将其中的Id，Image，Command，Created，Status，Ports，Names分别提取出来并存储在新的数组IdArry，ImageArray，CommandArray，CreatedArray，StatusArray，PortsArray，nameArray中，供HTML界面显示使用。关键代码见附录4、(2)、2)。

4.6.2 Start

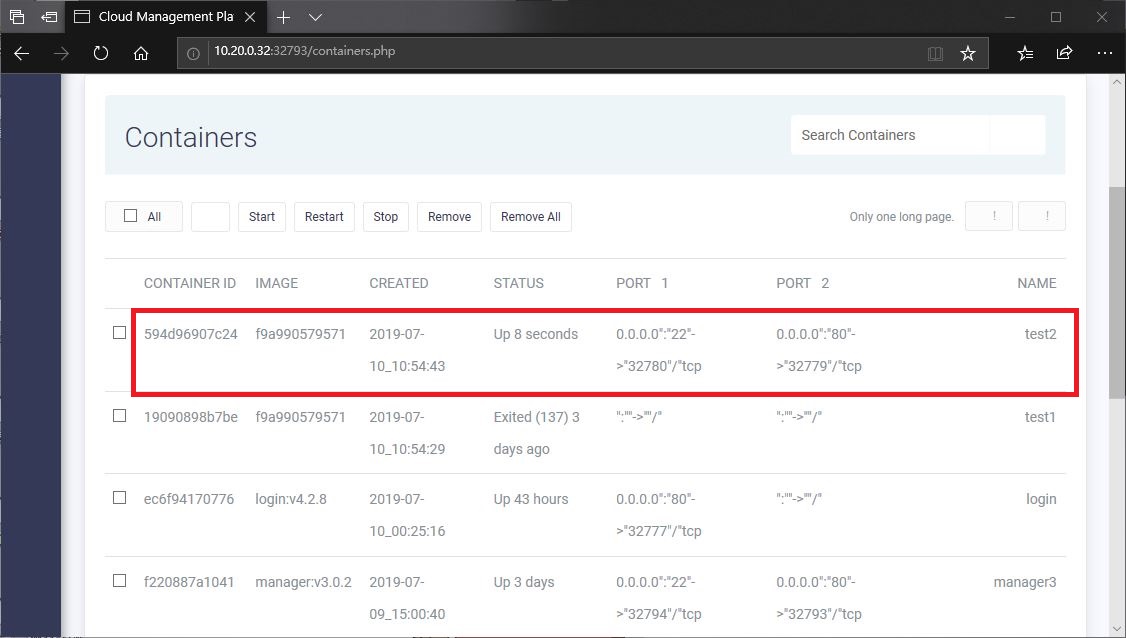
点击“Start”按钮，程序调用prompt()函数弹出输入窗口，用户输入所要“Start”的容器ID，点击“确定”按钮，程序通过“window.location.href='opt\_con.php?opt=start&containerid='+containerid;”调用opt\_con.php文件，并利用GET方式传入操作类型“opt”和容器ID“containerid”两个参数。opt\_con.php文件执行run($opt,$containerid)函数，根据“opt”参数选择对应的功能函数“startContainer($containerid)”完成容器启动操作。操作成功后，跳转回containers.php并刷新容器列表。如图4.9所示。



(a) 点击“Start”按钮



(b) 输入容器ID

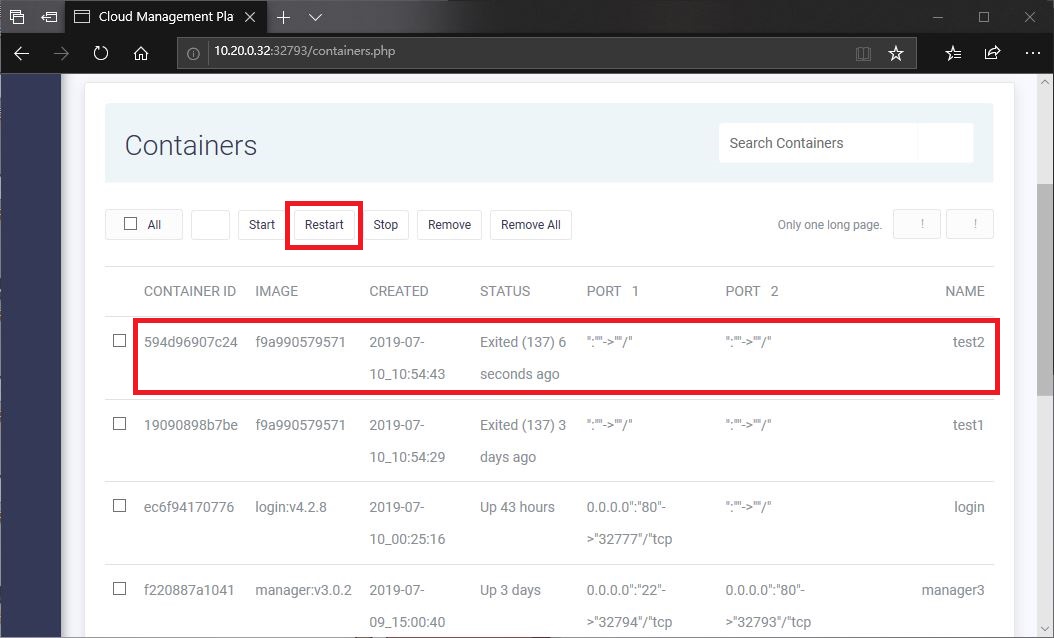


(c) 查看结果

图4.9 “Start”操作

4.6.3 Restart

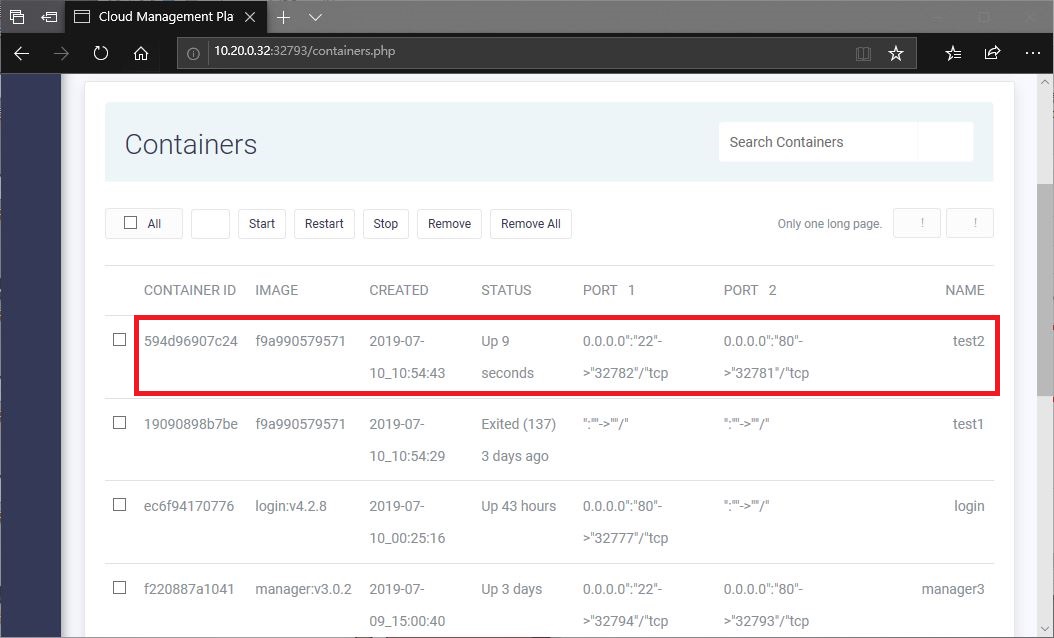
点击“Restart”按钮，程序调用prompt()函数弹出输入窗口，用户输入所要“Restart”的容器ID，点击“确定”按钮，程序通过“window.location.href='opt\_con.php?opt=restart&containerid='+containerid;”调用opt\_con.php文件，并利用GET方式传入操作类型“opt”和容器ID“containerid”两个参数。opt\_con.php文件执行run($opt,$containerid)函数，根据“opt”参数选择对应的功能函数“restartContainer($containerid)”完成容器重启动操作。操作成功后，跳转回containers.php并刷新容器列表。如图4.10所示。



(a) 点击“Restart”按钮



(b) 输入容器ID

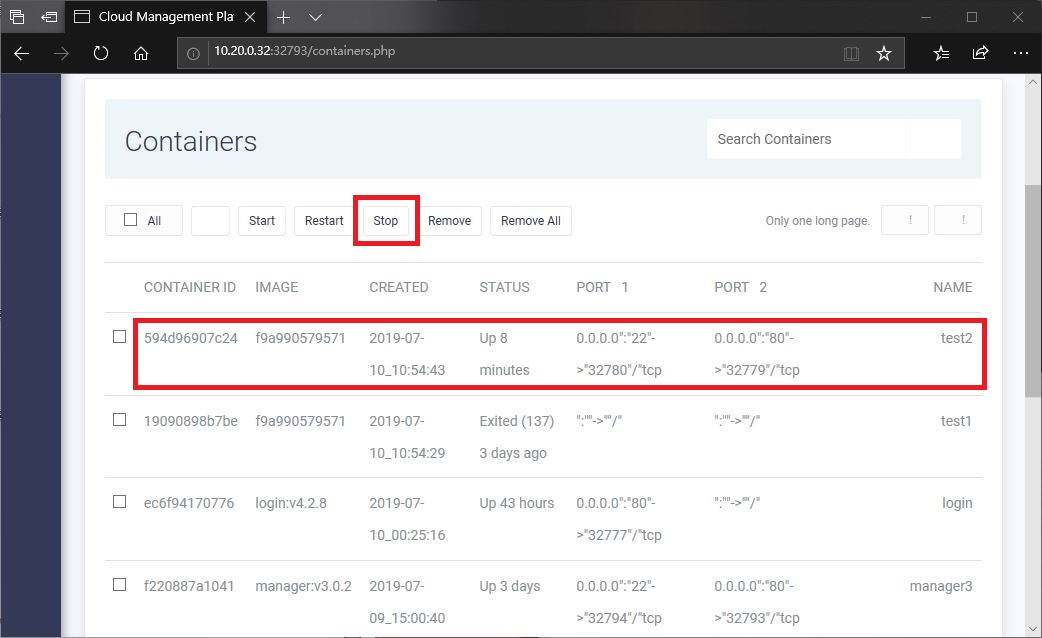


(c) 查看结果

图4.10 “Restart”操作

4.6.4 Stop

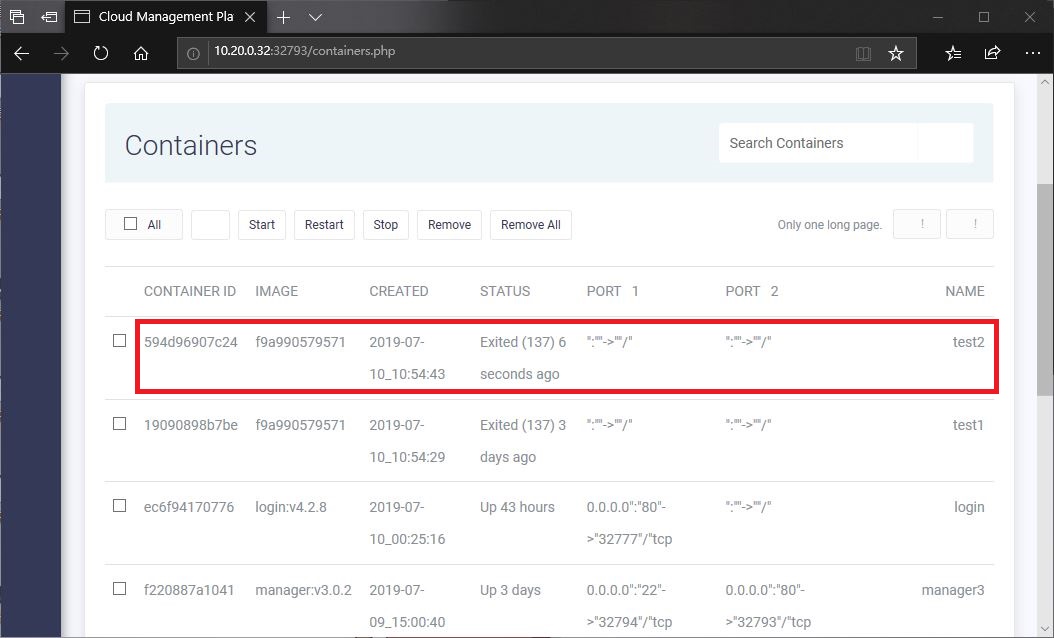
点击“Stop”按钮，程序调用prompt()函数弹出输入窗口，用户输入所要“Stop”的容器ID，点击“确定”按钮，程序通过“window.location.href='opt\_con.php?opt=stop&containerid='+containerid;”调用opt\_con.php文件，并利用GET方式传入操作类型“opt”和容器ID“containerid”两个参数。opt\_con.php文件执行run($opt,$containerid)函数，根据“opt”参数选择对应的功能函数“restartContainer($containerid)”完成容器停止操作。操作成功后，跳转回containers.php并刷新容器列表。如图4.11所示。



(a) 点击“Stop”按钮



(b) 输入容器ID

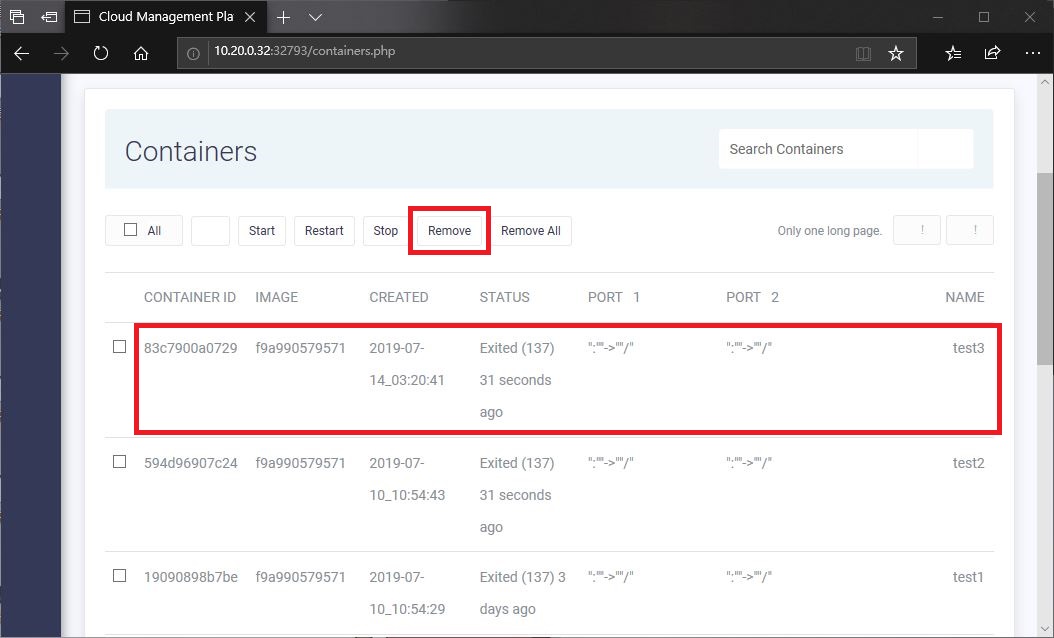


(c) 查看结果

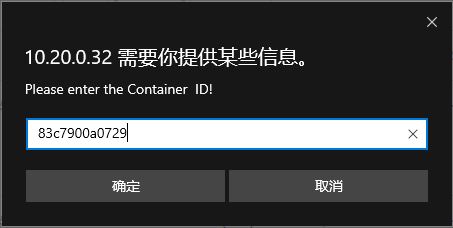
图4.11 “Stop”操作

4.6.5 Remove

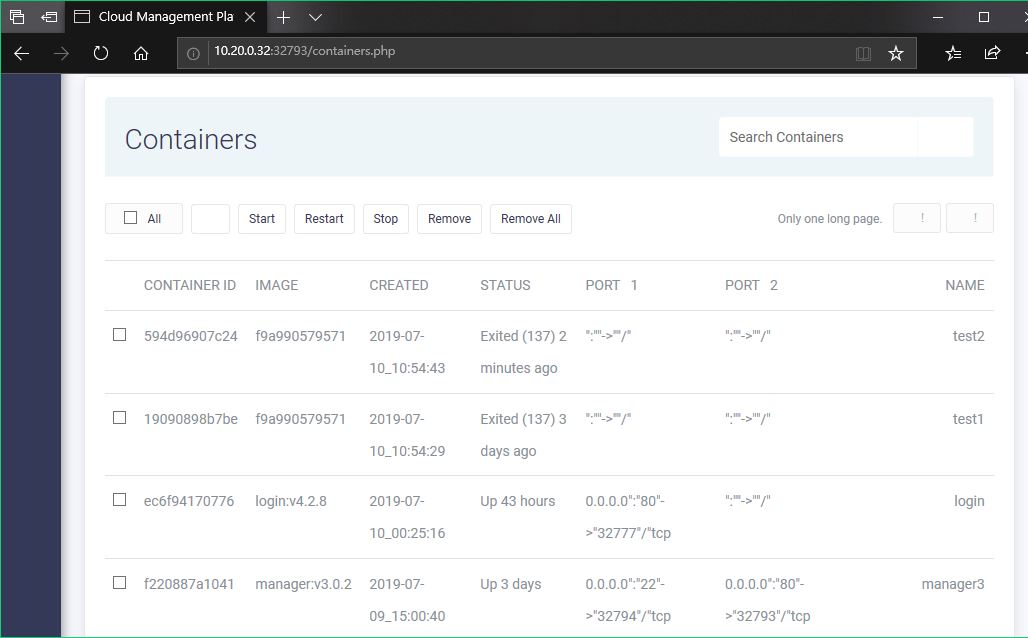
点击“Remove”按钮，程序调用prompt()函数弹出输入窗口，用户输入所要“Remove”的容器ID，点击“确定”按钮，程序通过“window.location.href='opt\_con.php?opt=remove&containerid='+containerid;”调用opt\_con.php文件，并利用GET方式传入操作类型“opt”和容器ID“containerid”两个参数。opt\_con.php文件执行run($opt,$containerid)函数，根据“opt”参数选择对应的功能函数“restartContainer($containerid)”完成容器删除操作。操作成功后，跳转回containers.php并刷新容器列表。如图4.12所示。



(a) 点击“Remove”按钮



(b) 输入容器ID

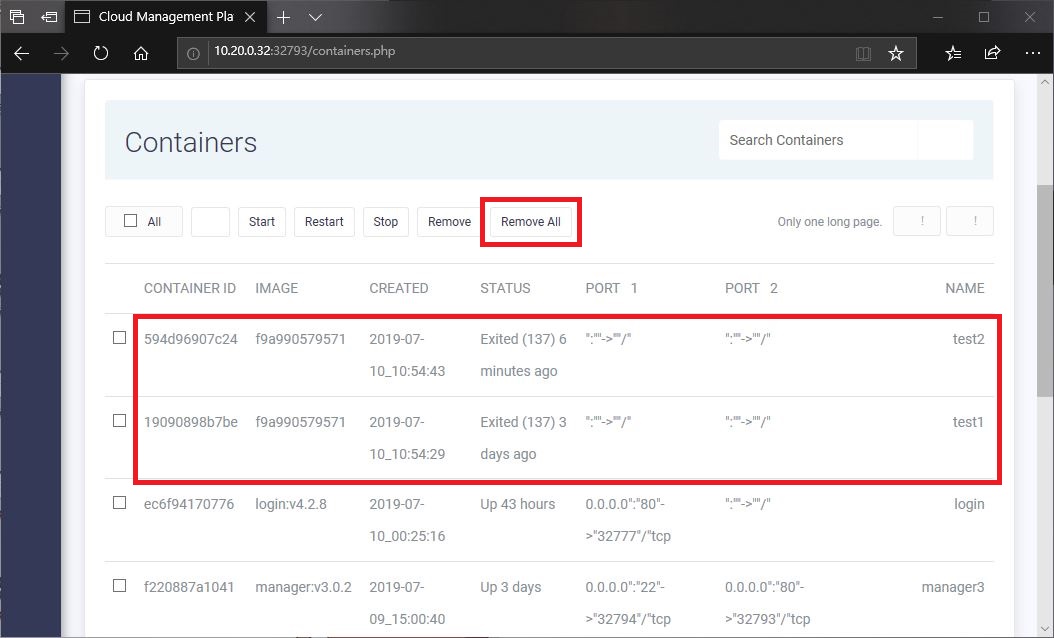


(c) 查看结果

图4.12 “Remove”操作

4.6.6 Remove All

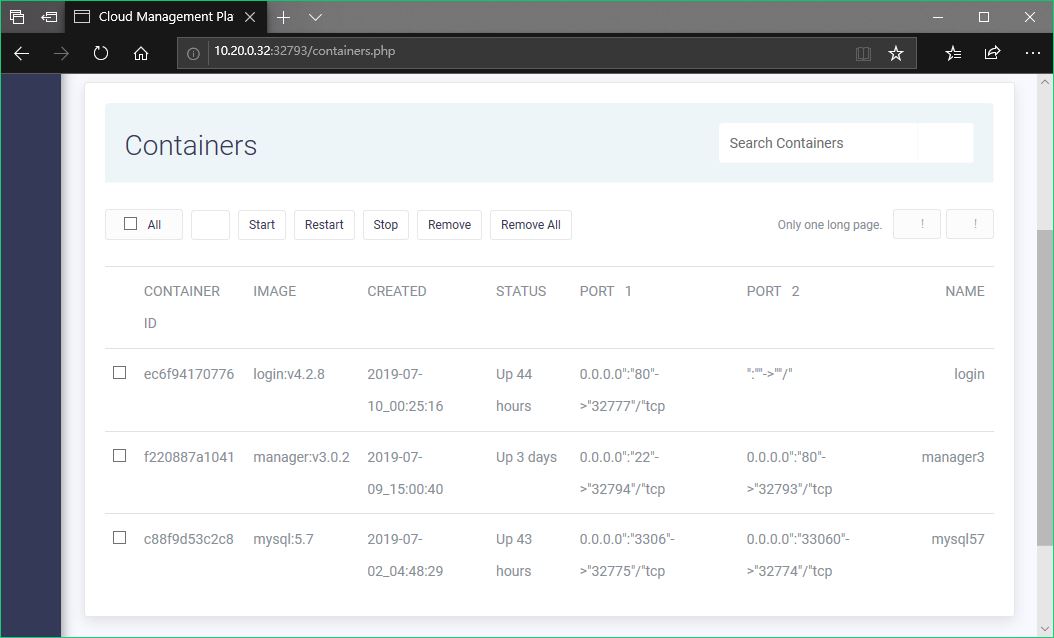
点击“Remove All”按钮，程序调用prompt()函数弹出输入窗口，用户无需输入，点击“确定”按钮，程序通过“window.location.href='opt\_con.php?opt= removeall&containerid='+containerid;”调用opt\_con.php文件，并利用GET方式传入操作类型“opt”和容器ID“containerid”两个参数。opt\_con.php文件执行run($opt,$containerid)函数，根据“opt”参数选择对应的功能函数“restartContainer($containerid)”完成删除全部已停止容器操作。操作成功后，跳转回containers.php并刷新容器列表。如图4.13所示。



(a) 点击“Remove All”按钮



(b) 默认容器ID输入框



(c) 查看结果图

4.13 “Remove All”操作

总结

在此次云计算系统搭建的过程中，我与李文华对于docker技术有了更加系统的认识与理解，docker使得服务更加灵活和轻量化。在此过程中，我们安装虚拟机并设置其网络模式为桥接模式，安装并启动docker服务，搭建login、manager、user、mysql镜像并创建相应容器，为每个容器挂载相应的数据卷，配置容器及数据文件形成云计算系统。

下载并安装VMware虚拟机并安装centos7.6操作系统，为方便通过浏览器访问，设置网卡驱动为开机自启，并设置IP为静态IP。安装最新版本的docker服务，配置并开启docker远程访问接口。

在系统中，以“login\_image”文件夹构建登录镜像，并挂载“login\_workspace”文件夹创建登录容器，使得所有用户通过该登录容器进入自己的容器。以mysql官方镜像为基础，并挂载“sql\_workspace”文件夹创建数据库容器，保存云计算系统的所有数据，登录容器负责查询数据库以验证账号及密码，无误后用户方可进入各自的容器。以“manager\_image”文件夹构建管理员镜像，并挂载“manager\_workspace”文件夹创建管理员容器，使得管理员通过自己的容器能够管理docker中的所有镜像和容器。以“user\_image”文件夹构建普通用户镜像，并挂载“user\_workspace”文件夹创建普通用户容器，使得普通用户能够借助浏览器通过exec实例在容器内执行命令。

我们的系统仍有不足之处，还需要继续完善。比如：系统未启用“会话控制”服务，由登录容器页面跳转到管理员容器页面或者普通用户容器页面时，通过 “IP:PORT”进行跳转，未创建Session对象，这使得任意用户可以通过“IP:PORT”直接进入容器而无需验证；镜像的部分管理功能尚未完善，仍存在不足；容器的管理未与数据库结合，没有在容器列表展示出容器的用户信息。

感谢李老师不厌其烦的指导，每次上课都给我们讲的很仔细，带领我们亲身实践，探索docker的各种功能。在做系统的过程中，每次看到老师的课件和自己的笔记，都会使编程思路和整体框架更加清晰，都会感觉自己还可以继续往下做。谢谢老师！

附录

系统部分文件的关键代码在附录中体现，以下是部分文件的关键代码。

1、login\_image文件夹

(1)Dockerfile文件：

#基础镜像centos

FROM 9f38484d220f

#作者信息

MAINTAINER Wenhua and Chen

#工作目录

WORKDIR /my\_httpd/

#安装httpd和php。

RUN yum -y install httpd php

#安装mysql客户端(用来和远程mysql服务端通信)和php-mysqlnd(用于php和mysql的沟通驱动)；

RUN yum -y install php-mysqlnd

#定义环境变量，下面的ip地址就是刚才创建mysql容器的ip地址，

#用于 php\_mysqli 函数的MYSQL\_ADDR

ENV MYSQL\_ADDR 172.17.0.3

ENV MYSQL\_USER root

ENV MYSQL\_PASS 123456

ENV MYSQL\_DATABASE docker\_user

#定义启动服务必需的环境变量TERM，并定义支持中文；

ENV TERM linux

ENV LC\_ALL en\_US.UTF-8

#添加启动脚本

ADD run.sh /my\_httpd/run.sh

RUN chmod u+x /my\_httpd/run.sh

#导出apache的服务端口80

EXPOSE 80

CMD /my\_httpd/run.sh

(2)run.sh文件：

#!/bin/bash

#httpd

/usr/sbin/httpd

#防止脚本结束

while true;do sleep 1000;done

2、login\_workspace文件夹

(1)login\_check.php文件：

<?php

$port = $\_POST['sj\_land\_id'];

$pass = $\_POST['sj\_land\_code'];

if($port == ""){

echo"<script type='text/javascript'>alert('请填写用户名');location='index.html';</script>";}

elseif($pass == ""){

echo"<script type='text/javascript'>alert('请填写密码');location='index.html';</script>";}

else{

// 创建连接

$conn = mysqli\_connect(getenv("MYSQL\_ADDR"),getenv("MYSQL\_USER"),getenv("MYSQL\_PASS"),getenv("MYSQL\_DATABASE"));

// Check connection

if (!$conn){

die("连接失败: " . mysqli\_connect\_error());}

mysqli\_query("SET NAMES utf8");

$sql = "SELECT \* FROM port\_pass Where port = $port and pass = $pass ";

$result = mysqli\_query($conn, $sql);

$colum = mysqli\_fetch\_assoc($result);

if(($colum['port'] == $port) && ($colum['pass'] == $pass)){

echo"<script type='text/javascript'>alert('登陆成功');location='http://10.20.0.12:$port';</script>";}

else{

echo"<script type='text/javascript'>alert('密码错误');location='index.html';</script>";

}}

?>

(2) register\_check.php文件：

<?php

$name=$\_POST['sj\_regis\_id'];

$password=$\_POST['sj\_regis\_code'];

$pwd\_again=$\_POST['sj\_regis\_codeconfirm'];

$code=$\_POST['sj\_regis\_check'];

if($name==""|| $password==""){

echo"<script type='text/javascript'>alert('用户名或者密码不能为空');location='register.html';</script>";}

else{

if($password!=$pwd\_again){

echo"<script type='text/javascript'>alert('两次输入的密码不一致,请重新输入！');location='register.html';</script>";}

else if ($code!=0){

echo"<script type='text/javascript'>alert('验证码错误！');location='register.html';</script>";}

else{

// 创建连接

#MYSQL\_DATABASE 加进docker

$conn = mysqli\_connect(getenv("MYSQL\_ADDR"),getenv("MYSQL\_USER"),getenv("MYSQL\_PASS"),getenv("MYSQL\_DATABASE"));

// Check connection

if (!$conn) {

die("连接失败: " . mysqli\_connect\_error());}

mysqli\_query("SET NAMES utf8");

$sql="insert into port\_pass values('105','$password','$name','teacher')";

$result = mysqli\_query($conn, $sql);

if(!$result){

echo"<script type='text/javascript'>alert('注册不成功！');location='register.html';</script>";}

else{

echo"<script type='text/javascript'>alert('注册成功！');location='index.html';</script>";}

mysqli\_close($conn); }

}

?>

(3) title\_logo\_g.png:



3、manager\_image文件夹

(1)Dockerfile文件:

#基础镜像centos

FROM 9f38484d220f

#作者信息

MAINTAINER Wenhua and Chen

#工作目录

WORKDIR /manager/

#安装httpd和php。

RUN yum -y install httpd php

#安装mysql客户端(用来和远程mysql服务端通信)和php-mysqlnd(用于php和mysql的沟通驱动)；

RUN yum -y install mysql php-mysqlnd

#定义环境变量，下面的ip地址就是刚才创建mysql容器的ip地址，

ENV MYSQL\_ADDR 172.17.0.3

ENV MYSQL\_USER root

ENV MYSQL\_PASS 123456

ENV MYSQL\_DATABASE docker\_user

#定义启动服务必需的环境变量TERM，并定义支持中文；

ENV TERM linux

ENV LC\_ALL en\_US.UTF-8

#注意：mysql8字符集已改为utf8mb4格式，创建容器时要改为utf-8，否则浏览器不识别。

#导出apache的服务端口80

EXPOSE 80

# 安装openssh-server

RUN yum -y install openssh-server

RUN mkdir /var/run/sshd

RUN ssh-keygen -t rsa -f /etc/ssh/ssh\_host\_rsa\_key

RUN ssh-keygen -t dsa -f /etc/ssh/ssh\_host\_dsa\_key

# 指定root密码

RUN /bin/echo 'root:123456'|chpasswd

RUN /bin/sed -i 's/.\*session.\*required.\*pam\_loginuid.so.\*/session optional pam\_loginuid.so/g' /etc/pam.d/sshd

RUN /bin/echo -e "LANG=\"en\_US.UTF-8\"" > /etc/default/local

EXPOSE 22

#添加启动脚本

ADD run.sh /manager/run.sh

RUN chmod u+x /manager/run.sh

CMD /manager/run.sh

(2)run.sh

#!/bin/bash

/usr/sbin/sshd -D &

httpd

#防止脚本结束

while true;do sleep 1000;done

4、manager\_workspace文件夹

(1)images\_php文件部分重要代码：

1)按钮代码：

<div class="btn-group">

<a data-original-title="Refresh" data-placement="top" data-toggle="dropdown" href="#" class="btn mini tooltips"><i class=" fa fa-refresh"></i></a>

</div>

<div class="btn-group">

<script type="text/javascript">

var str=prompt("Please enter the base image !","ID");

if(str) {

alert("Input ："+ str) }}

</script>

<!--<input type="submit" name="Su1" onclick="diag()" /> -->

<a data-toggle="dropdown" href="#" class="btn mini blue" type="submit" onclick="diag()">Remove</a>

</div>

<div class="btn-group">

<script type="text/javascript">

function diag(){

var str=prompt("Please enter the images !","ID");

if(str) {

alert("Input ："+ str) }}

</script>

<a data-toggle="dropdown" href="#" class="btn mini blue" type="submit" onclick="diag()">Remove All</a>

</div>

<div class="btn-group">

<script type="text/javascript">

function diag(){

var str=prompt("Please enter the base image !","ID");

if(str){

alert("Input ："+ str)}}

</script>

<!--<input type="submit" name="Su1" onclick="diag()" /> -->

<a data-toggle="dropdown" href="#" class="btn mini blue" type="submit" onclick="diag()">Create A Container</a>

</div>

2)镜像列表代码

function getAllImages(){

$res=getDockerAPI(host."images/json?all=1");

$array=json\_decode($res,true);

$IdArray;

$Created;

$Date;

$SizeArray;

$RepoTagsArray;

for($i=0;$i<count($array);$i++){

$IdArray[$i]=substr($array[$i]['Id'], 8,12) ;

$Created[$i] = $array[$i]['Created'];

$Date[$i]=date('Y-m-d\_H:i:s',$CreatedArray[$i]);

$SizeArray[$i]=round($array[$i]['Size']/1000/1000,2);

$RepoTagsArray[$i]=$array[$i]['RepoTags'][0];}

for($i=0;$i<count($array);$i++){

echo ' <tr class=""><td class="inbox-small-cells"><input type="checkbox" class="mail-checkbox"></td>

<td class="inbox-small-cells"><i class="ti-star"></i></td>

<td class="view-message dont-show">' . $IdArray[$i] .

'<td class="view-message dont-show">' . $Date[$i] .

'</td><td class="view-message">'. $SizeArray[$i] .

'</td><td class="view-message inbox-small-cells"></td><td class="view-message text-right">'. $RepoTagsArray[$i].

'</td> </tr> ';}

(2)containers\_php文件部分重要代码：

1)按钮代码：

<div class="btn-group">

<script type="text/javascript">

function Start(){

var containerid=prompt("Please enter the Container ID !","ID");

window.location.href='opt\_con.php?opt=start&containerid=' + containerid; }</script>

<a data-toggle="dropdown" href="#" class="btn mini blue" type="submit" onclick="Start()">Start</a></div>

<div class="btn-group">

<script type="text/javascript">

function Restart(){

var containerid=prompt("Please enter the Container ID!","ID");

window.location.href='opt\_con.php?opt=restart&containerid=' + containerid; }</script>

<a data-toggle="dropdown" href="#" class="btn mini blue" type="submit" onclick="Restart()">Restart</a></div>

<div class="btn-group"><script type="text/javascript">

function Stop(){

var containerid=prompt("Please enter the Container ID!","ID");

window.location.href='opt\_con.php?opt=stop&containerid=' + containerid;}</script>

<a data-toggle="dropdown" href="#" class="btn mini blue" type="submit" onclick="Stop()">Stop</a></div>

<div class="btn-group"><script type="text/javascript">

function Remove(){

var containerid=prompt("Please enter the Container ID!","ID");

window.location.href='opt\_con.php?opt=remove&containerid=' + containerid; }</script>

<a data-toggle="dropdown" href="#" class="btn mini blue" type="submit" onclick="Remove()">Remove</a></div>

<div class="btn-group">

<script type="text/javascript">

function Remove\_all(){

var containerid=prompt("Please enter the Containers ID!","ID"); window.location.href='opt\_con.php?opt=removeall&containerid=' + containerid; }</script>

<a data-toggle="dropdown" href="#" class="btn mini blue" type="submit" onclick="Remove\_all()">Remove All</a></div>

2)容器列表代码

function getAllContainers(){

$res=getDockerAPI(host."containers/json?all=1");

$array=json\_decode($res,true);

$IdArray; $ImageArray; $CommandArray; $CreatedArray; $Date; $StatusArray;

$PortsArray; $SubPortsArray; $SubSubPortsArray; $nameArray;

for($i=0;$i<count($array);$i++){

$IdArray[$i]=substr($array[$i]['Id'], 0,12) ;

$ImageArray[$i]=$array[$i]['Image'];

$CommandArray[$i]=$array[$i]['Command'];

$CreatedArray[$i]=$array[$i]['Created'];

$Date[$i] = date('Y-m-d\_H:i:s',$CreatedArray[$i]);

$StatusArray[$i]=$array[$i]['Status'];

$PortsArray[$i]=array($array[$i]['Ports']);

$SubPortsArray[$i]=$PortsArray[$i][0];

$SubSubPortsArray0[$i]=$SubPortsArray[$i][0];

$SubSubPortsArray1[$i]=$SubPortsArray[$i][1];

$NameArray[$i]=substr($array[$i]['Names'][0], 1);}

for($i=0;$i<count($array);$i++){

echo ' <tr class="">

<td class="inbox-small-cells"><input type="checkbox" class="mail-checkbox"></td>

<td class="view-message dont-show">' . $IdArray[$i] .

'<td class="view-message dont-show">' . $ImageArray[$i] .

'<td class="view-message dont-show">' . $Date[$i] .

'<td class="view-message dont-show">' . $StatusArray[$i] .

'</td><td class="view-message">'. $SubSubPortsArray0[$i]['IP'] .'":"'. $SubSubPortsArray0[$i]['PrivatePort'] . '"->"' . $SubSubPortsArray0[$i]['PublicPort']. '"/"' .$SubSubPortsArray0[$i]['Type'].

'</td><td class="view-message">'. $SubSubPortsArray1[$i]['IP'] .'":"'. $SubSubPortsArray1[$i]['PrivatePort'] . '"->"' . $SubSubPortsArray1[$i]['PublicPort']. '"/"' .$SubSubPortsArray1[$i]['Type'].

'</td><td class="view-message text-right">'. $NameArray[$i].

'</td></tr> ';}}

(3)opt\_con.php文件部分重要代码：

<?php $opt = $\_GET['opt'];

$containerid = $\_GET['containerid'];

define("host","http://10.20.0.12:5555/");

function postDockerAPI($url,$post\_param=false,$isjson=false,$d){

$ch = curl\_init();

if($isjson) $header[] = "Content-type:application/json"; //狠狠重要

else $header[] = "Content-type:text/plain;Charset=Utf-8";

curl\_setopt($ch, CURLOPT\_HEADER, 1);

curl\_setopt($ch, CURLOPT\_FOLLOWLOCATION, 1);

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);

curl\_setopt($ch, CURLOPT\_TIMEOUT, 5000);

if ($d == true) curl\_setopt($ch, CURLOPT\_CUSTOMREQUEST, "DELETE");

else curl\_setopt($ch, CURLOPT\_POST, 1);

if($post\_param)

curl\_setopt($ch, CURLOPT\_POSTFIELDS, $post\_param);

curl\_setopt($ch, CURLOPT\_SSL\_VERIFYPEER, FALSE);

curl\_setopt($ch, CURLOPT\_SSL\_VERIFYHOST, FALSE);

curl\_setopt($ch, CURLOPT\_HTTPHEADER, $header);

curl\_setopt($ch, CURLOPT\_URL, $url);

$res = curl\_exec($ch);}

function startContainer($containerid) {

$params='{"HostConfig":{ "Binds":null,"PortBindings":{},"Privileged":false}}';

postDockerAPI(host."containers/".$containerid."/start",$params,true,false); }

function restartContainer($containerid) {

$params='{"HostConfig":{ "Binds":null,"PortBindings":{},"Privileged":false}}';

postDockerAPI(host."containers/".$containerid."/restart",$params,true,false); }

function stopContainer($containerid) {

$params='{"HostConfig":{ "Binds":null,"PortBindings":{},"Privileged":false}}';

postDockerAPI(host."containers/".$containerid."/stop",$params,true,false); }

function deleteContainer($containerid) {

$params='{"HostConfig":{ "Binds":null,"PortBindings":{},"Privileged":false}}';

postDockerAPI(host."containers/".$containerid,$params,true,true); }

//清空已停止容器

function prunestopContainer(){

$params='{"HostConfig":{ "Binds":null,"PortBindings":{},"Privileged":false}}';

postDockerAPI(host."containers"."/prune",$params,true); }

//应位于opt\_images,以images中镜像为基础创建容器

function createContainer(){

$params="{".'"Image":"httpd",'.' "Cmd":['.' "usr/sbin/sshd -D" '.']'."}";

postDockerAPI(host."containers/create",$params,true,false);}

function run($opt,$containerid){

switch ($opt){

case "start":

startContainer($containerid) ;break;

case "restart":

restartContainer($containerid) ;break;

case "stop":

stopContainer($containerid) ;break;

case "remove":

deleteContainer($containerid) ;break;

case "removeall":

prunestopContainer() ;break;

case "create":

createContainer() ;break;}}

run($opt,$containerid);

?>