山东大学计算机科学与技术学院云计算技术课程实验报告

- **学号** : 201900161104　　**姓名**:张浩男　　**班级**:2019级4班
- **实验题目** : 实现在云服务器上的负载均衡
- **实验学时** : 6　　**实验日期**:2022/05/09

实验目的

在云端用docker部署nginx和tomcat实现负载均衡

具体包括：

1. 要求部署一台Nginx和三台Tomcat服务器
2. Ngnix需要实现三种策略：
    1. 轮询；
    2. 权重，三台服务器的权重为1，3，5；
    3. IP Hash。

最终实现的效果是，本地电脑通过网页访问云服务器时，网页能够体现Ngnix三种策略的结果。

实验环境

硬件环境

联网的计算机一台

软件环境

华为云

XShell

Docker

Ngnix

Tomcat

实验内容与步骤

配置一台云服务器

这里选择通过华为云配置云服务器。

1. **基础配置**

    这里选择的内容如下:

- 计费模式 包年包月
- 区域 华北-北京四
- 可用区 可用区1
- CPU架构 x86计算
- 规格

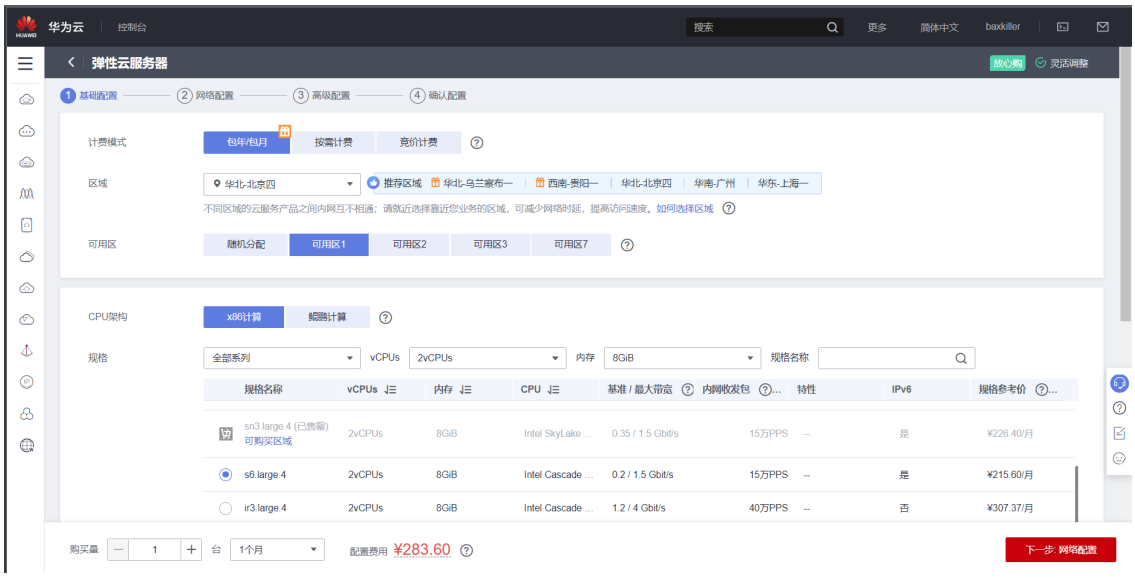
```
s6.large.4  2vCPUs  8GiB    IntelCascade 0.2/1.5Gits/s  15万PPS  IPV6
是
```

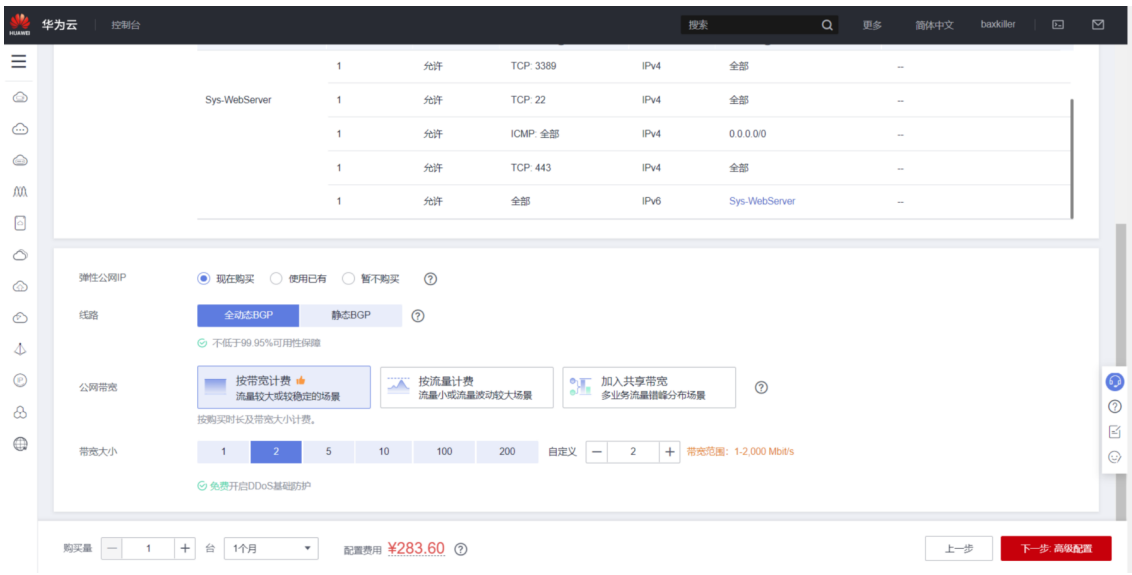
- 镜像

  公共镜像 : CentOS7.5 64bit(40GB)

- 系统盘 高IO40GiB

完整内容如下 :





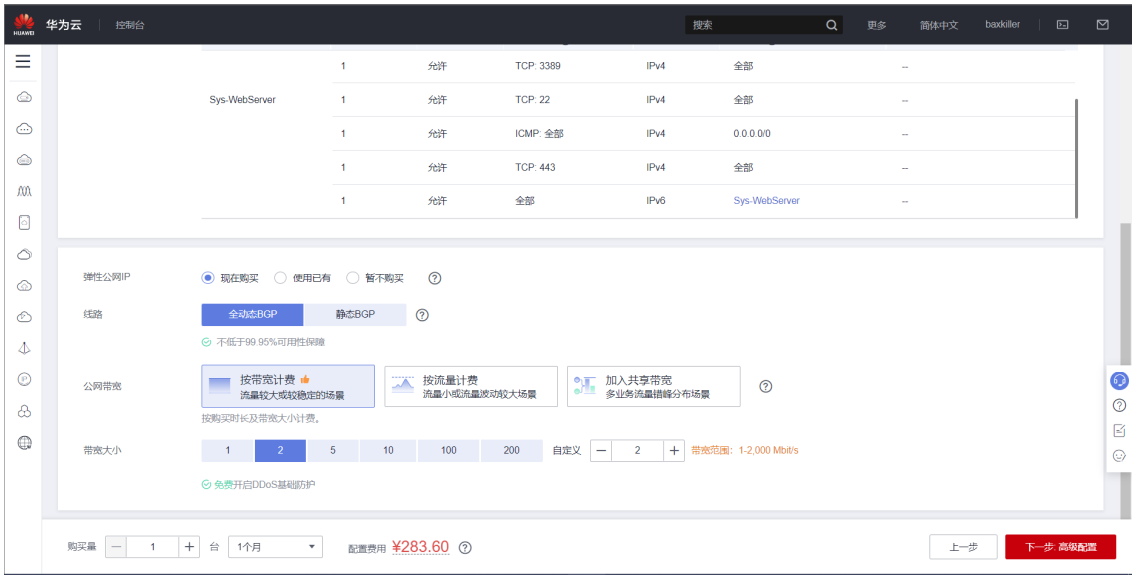
2. 网络配置

虚拟私有云选择`subnet-default`

弹性公网IP可以选择现在购买.全动态BGP

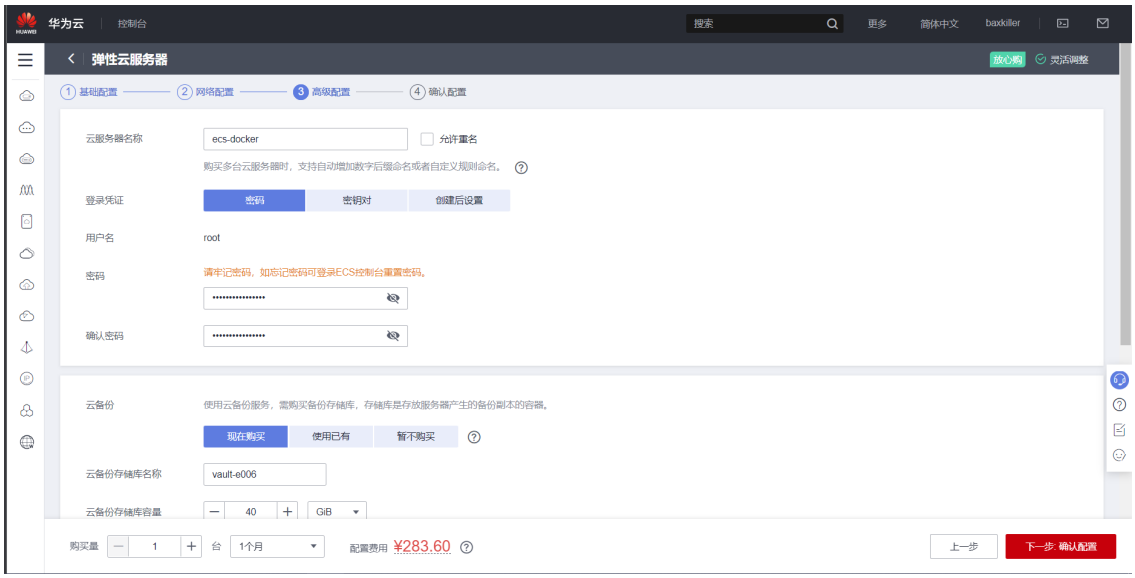
公网带宽选择按照带宽计费

带宽大小2,如下图所示:

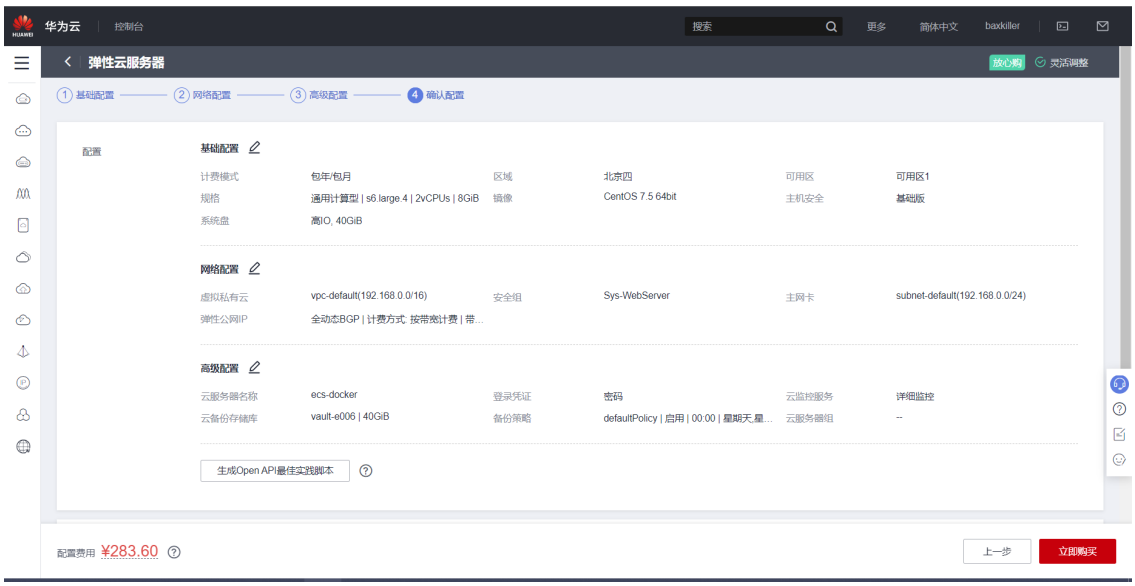


3. 高级配置

在此处设置云服务器名称,登陆凭证为密码

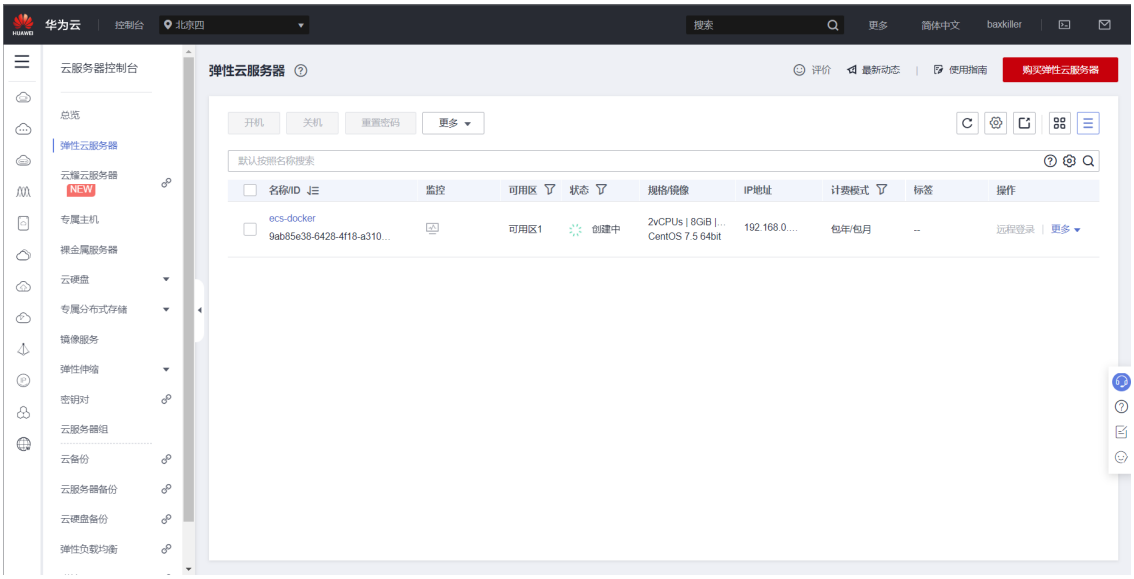


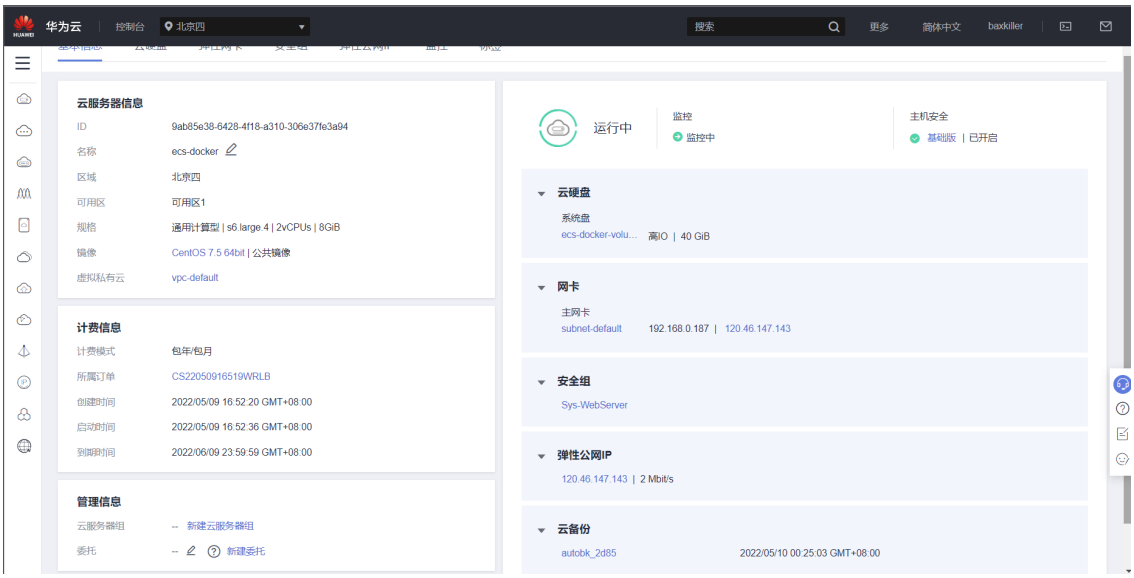
4. 确认配置 此处可以看到先前配置的所有内容,如下所示:



5. 购买

   购买完成后的界面应当如下所示:





服务器的链接与环境配置

服务器的链接这里选择使用XShell进行连接.

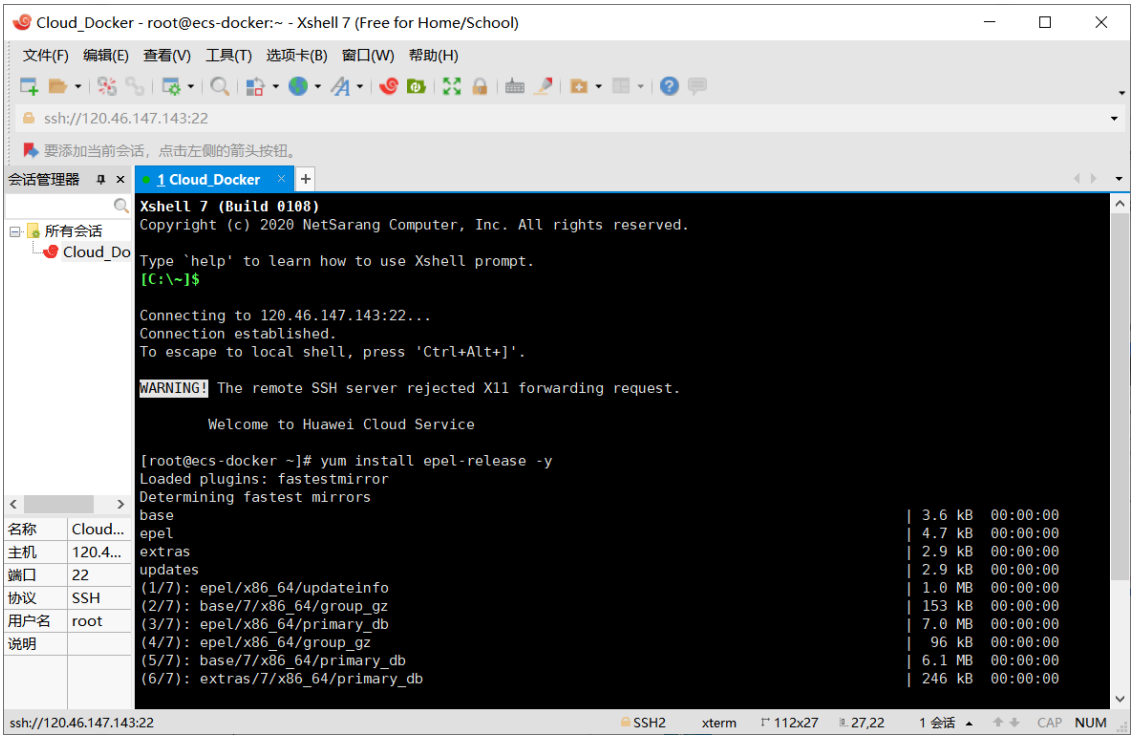
链接

通过新建,输入服务器的公网IP,然后输入用户名和密码,即可成功链接.

安装Docker

1. 添加yml源

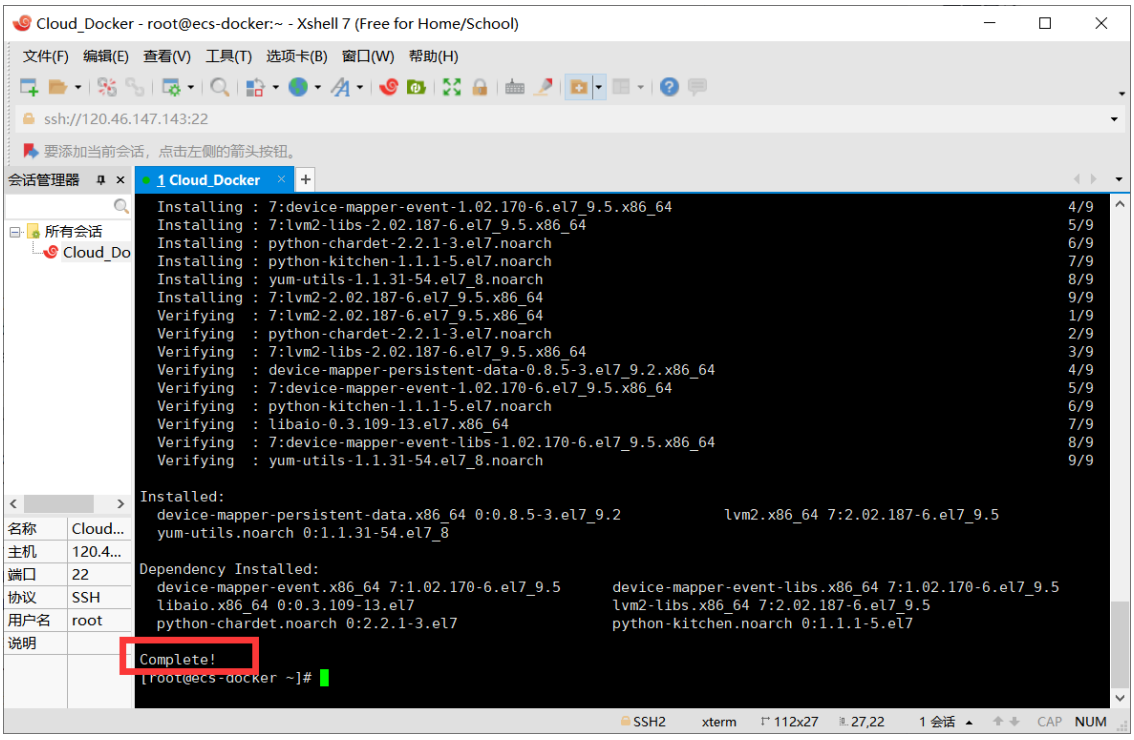
```
yum install epel-release -y
yum clean all
```



2. 安装uum-util

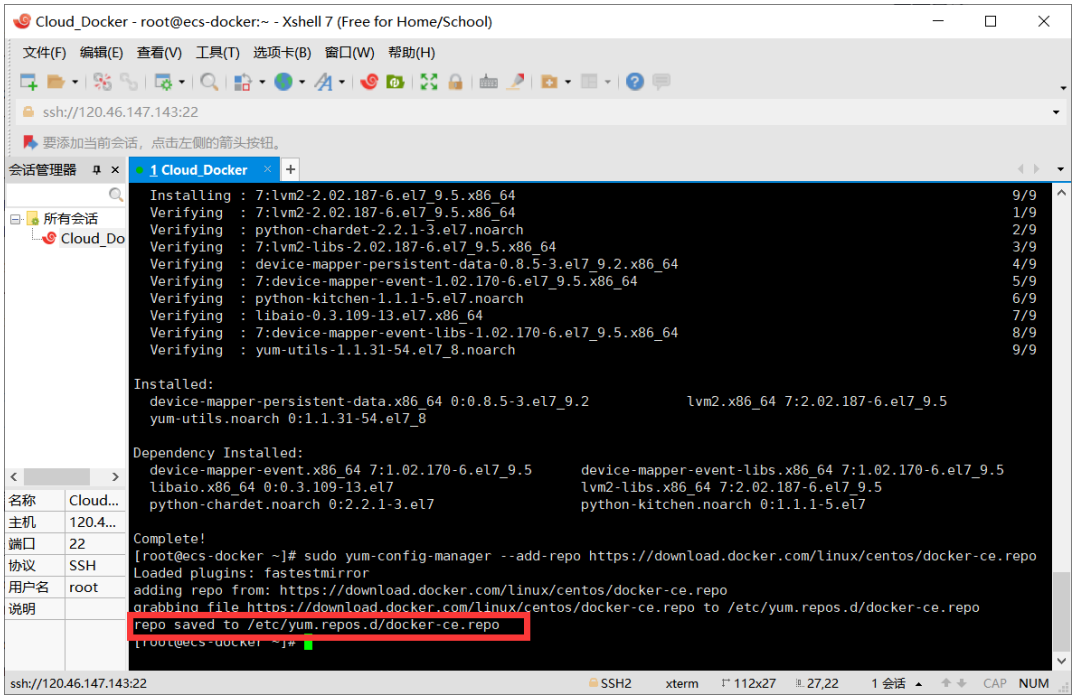
```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

安装成功如下:



3. 设置docker yum源

```
sudo yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
```

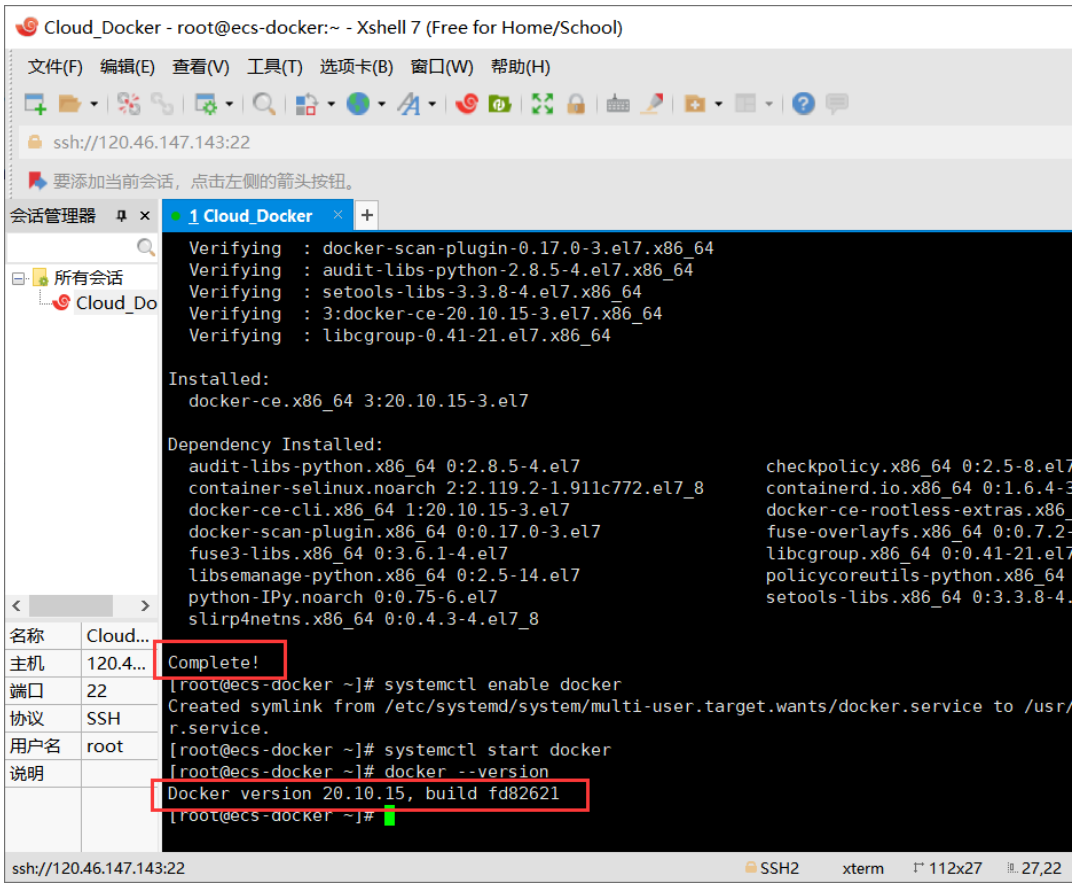


4. 安装并运行Docker。

```
sudo yum install docker-ce
systemctl enable docker
systemctl start docker
```

检查安装结果

```
docker --version
```

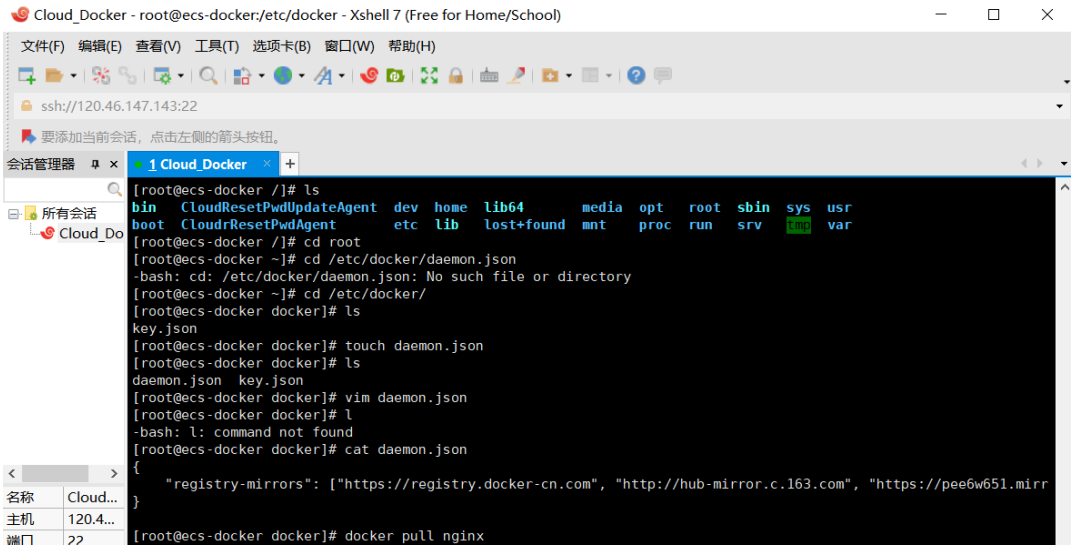


5. 配置docker镜像仓库(编辑对应json文件)

   路径: `/etc/docker/daemon.json`

   内容:

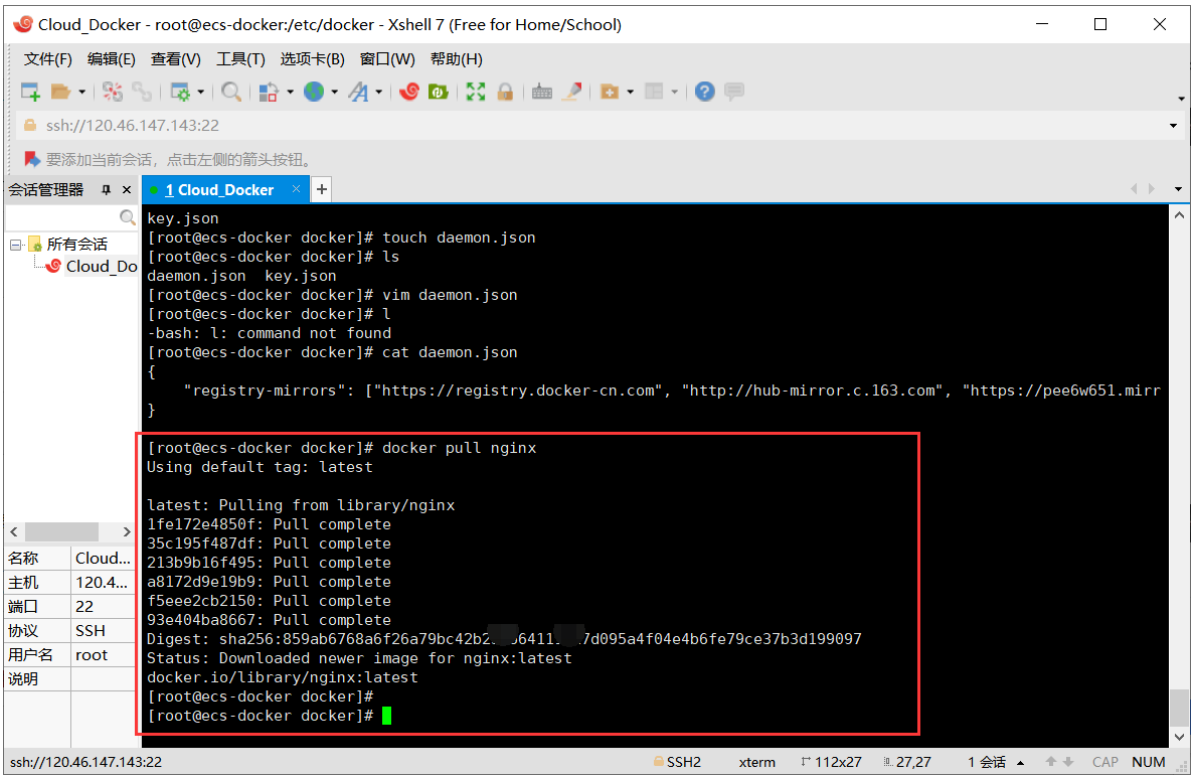
```
{
    "registry-mirrors":
    ["https://registry.docker-cn.com",
     "http://hub-mirror.c.163.com",
     "https://pee6w651.mirror.aliyuncs.com"]
}
```

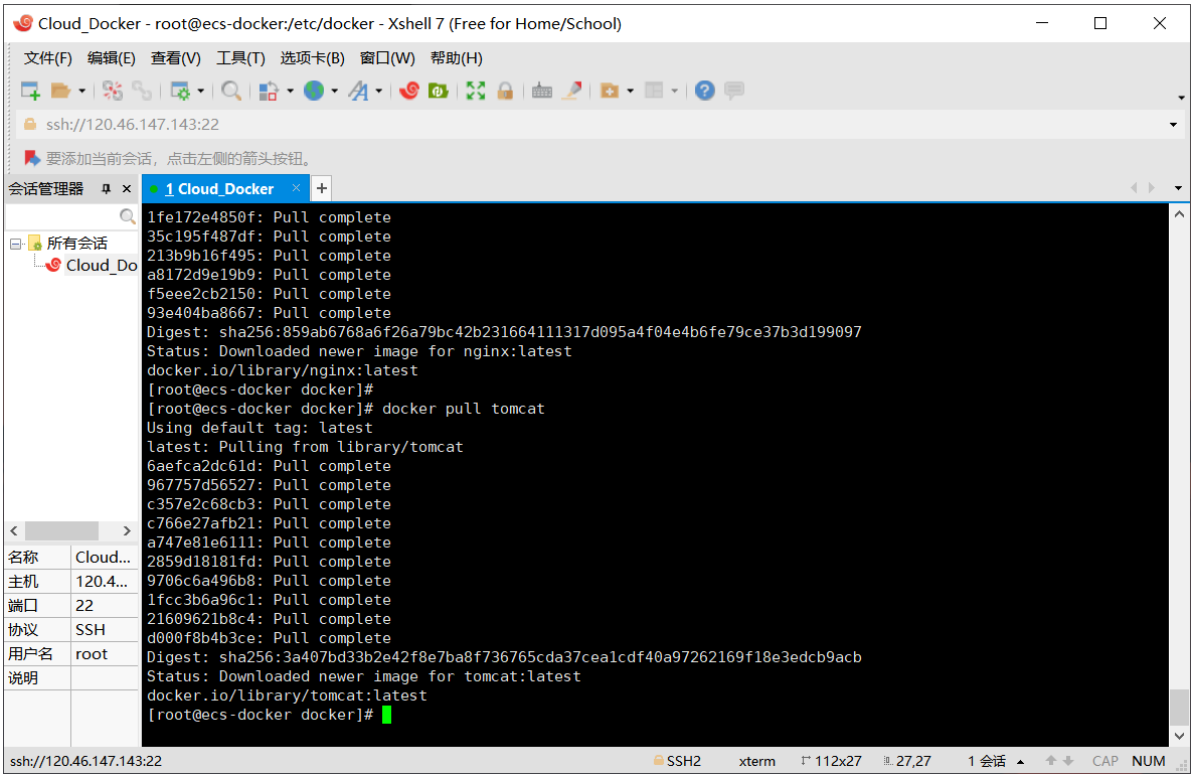


配置nginx与tomcat服务器

1. 下载nginx和tomcat的镜像

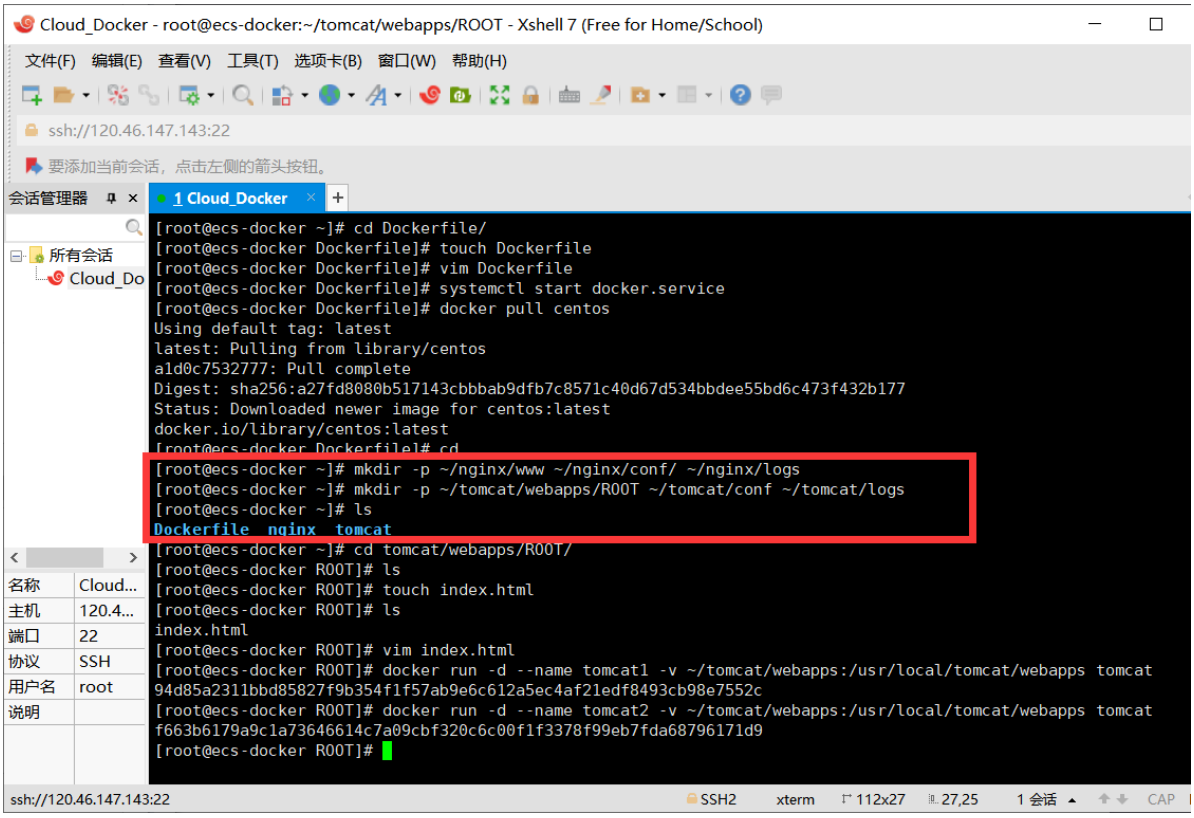
```
docker pull nginx
docker pull tomcat
```

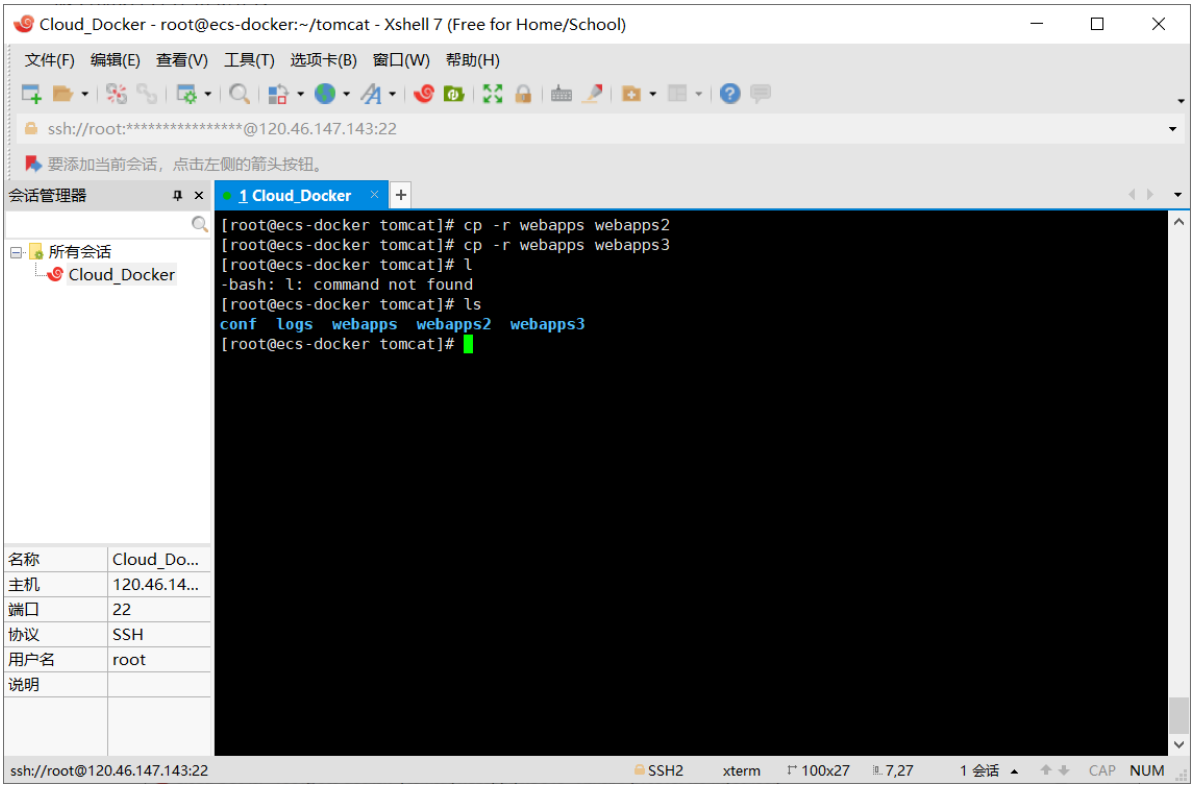




2. 创建nginx和tomcat本地目录 稍后挂载到docker容器上

   这里为了对三个tomcat对应的服务器主页面进行区分,在原有`webapps`文件的基础上,再创建两个`webapps`文件,用于将不同的tomcat服务器进行各自的对应.

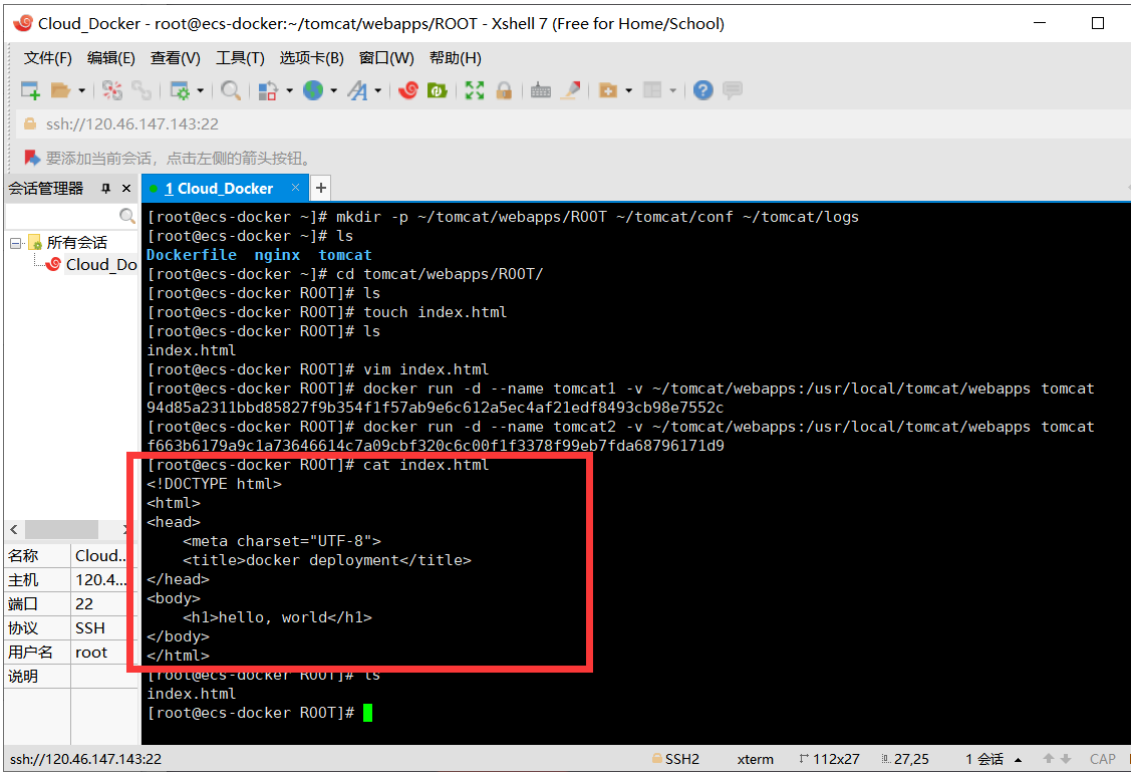




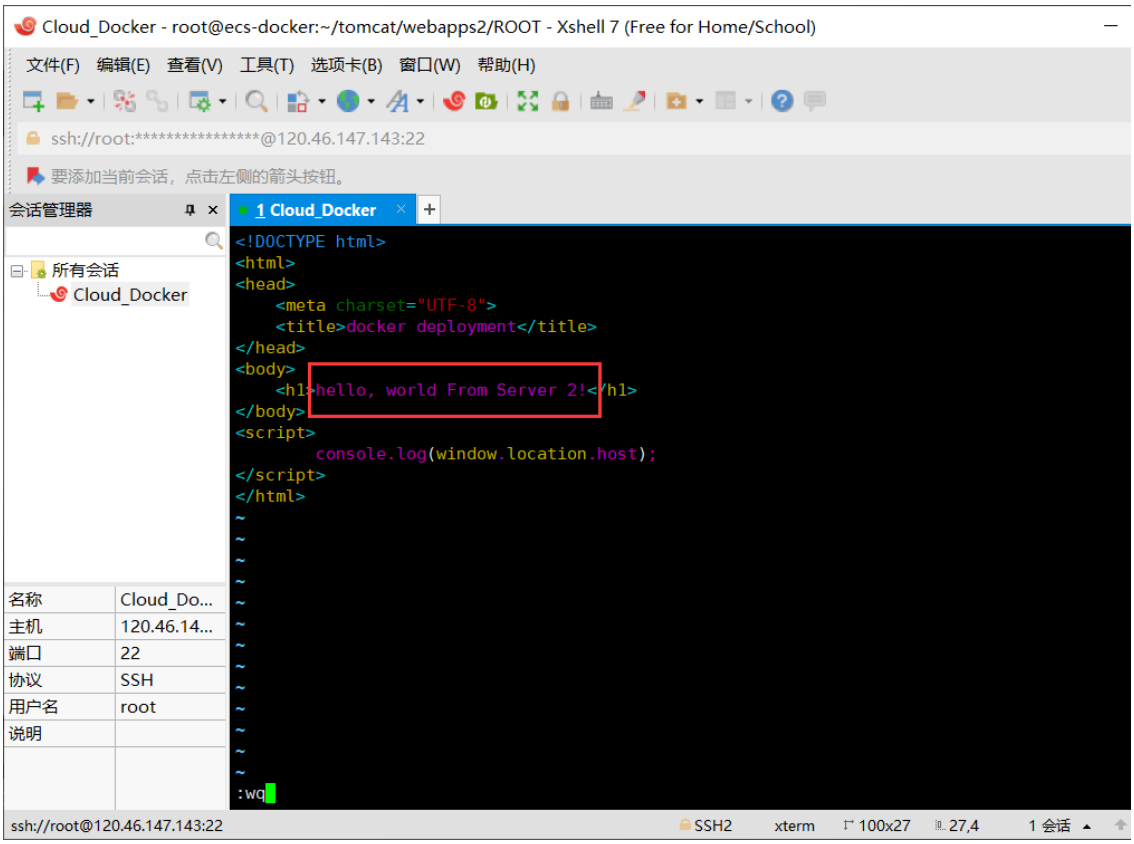
3. 编写`index.html`

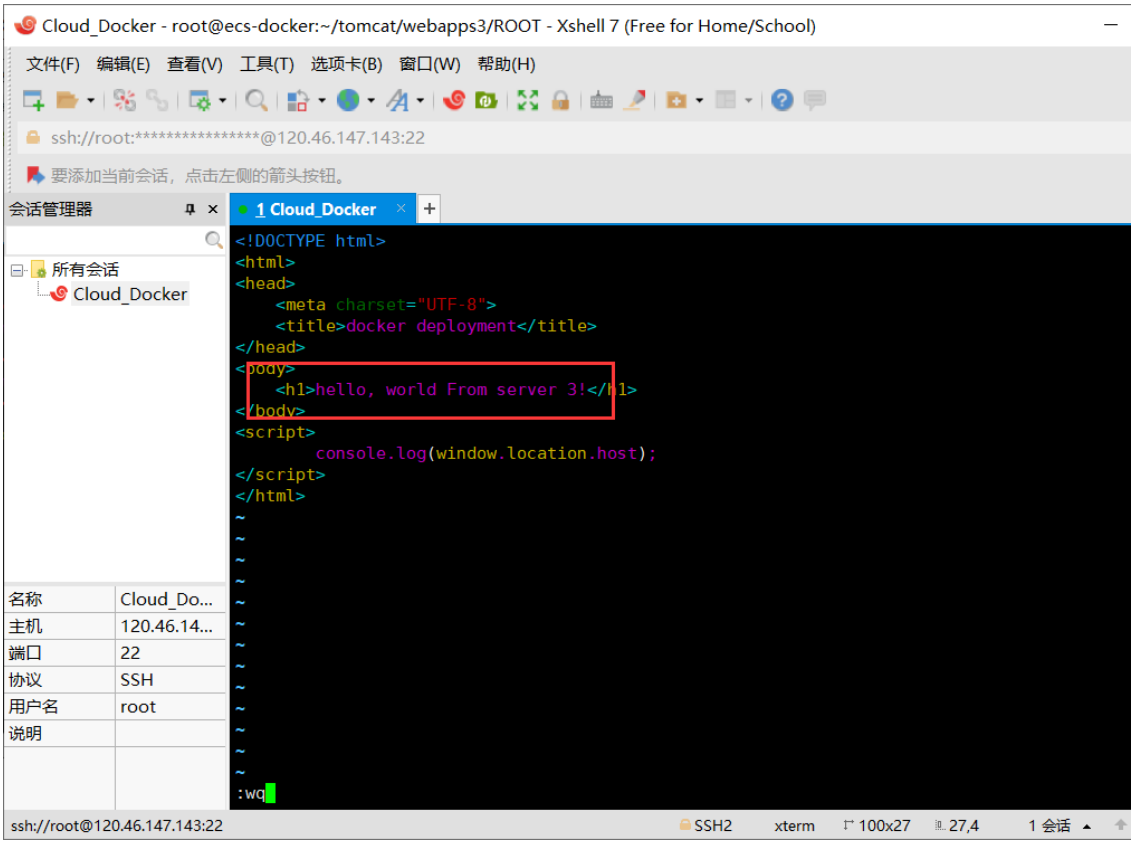
   内容如下:

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>docker deployment</title>
</head>
<body>
    <h1>hello, world</h1>
</body>
</html>
```



将该文件分别复制到另外两个 `webapps` 文件夹,并进行一定的修改以显现差异:



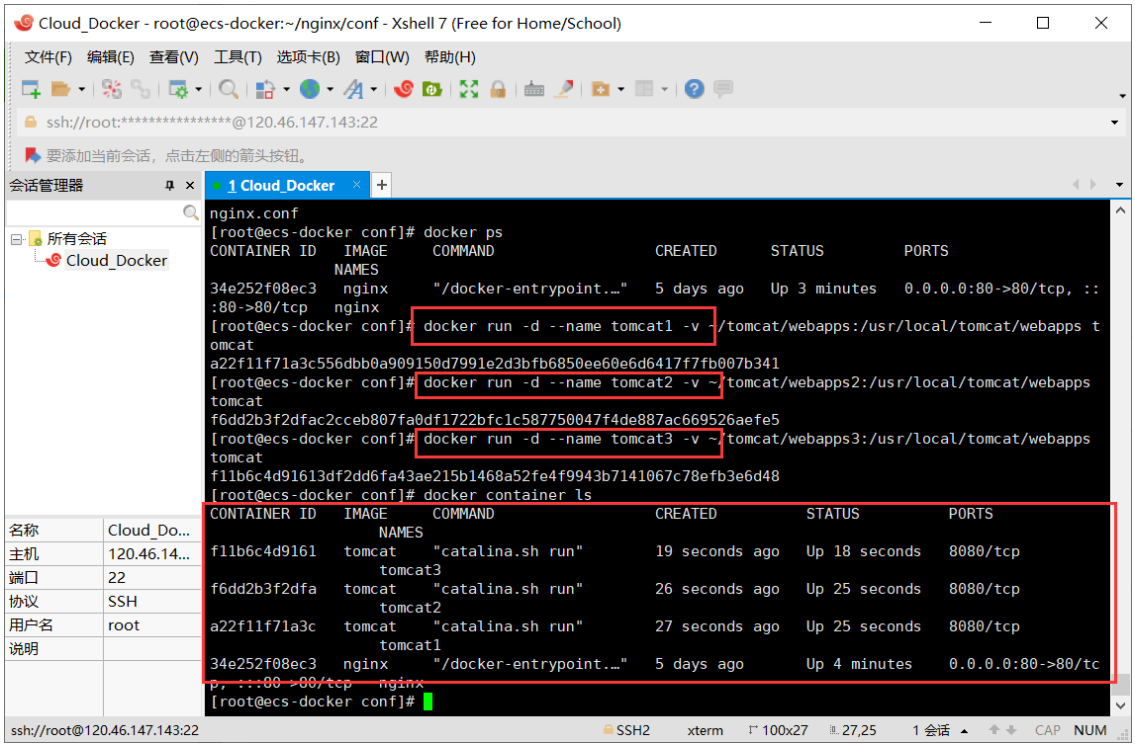


4. 启动三个tomcat容器

   注意这里的三个tomcat挂载的目录需要是不同的!

```
1 docker run -d --name tomcat1 -v
  ~/tomcat/webapps:/usr/local/tomcat/webapps tomcat
2 docker run -d --name tomcat2 -v
  ~/tomcat/webapps2:/usr/local/tomcat/webapps tomcat
3 docker run -d --name tomcat3 -v
  ~/tomcat/webapps3:/usr/local/tomcat/webapps tomcat
```

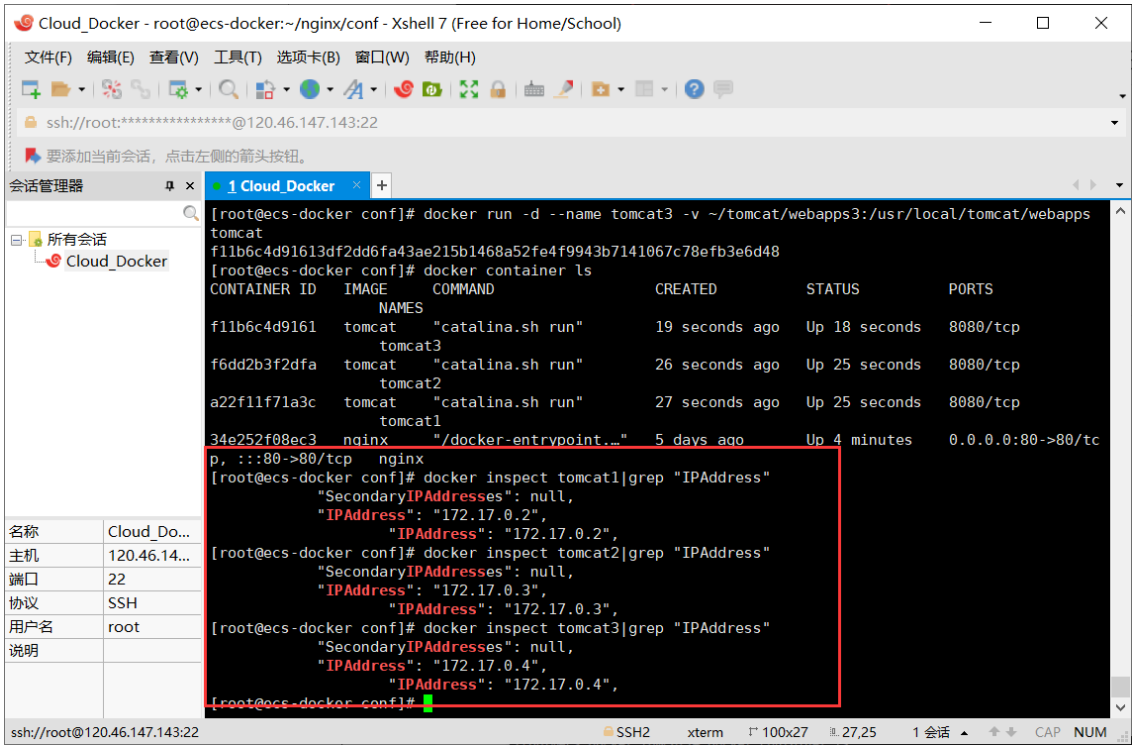
   启动完成后如下所示:



5. 获取tomcat容器IP，获取到的IP将配置到nginx的配置文件中：

   IP获取:

```
docker inspect tomcat1|grep "IPAddress"
docker inspect tomcat2|grep "IPAddress"
docker inspect tomcat3|grep "IPAddress"
```



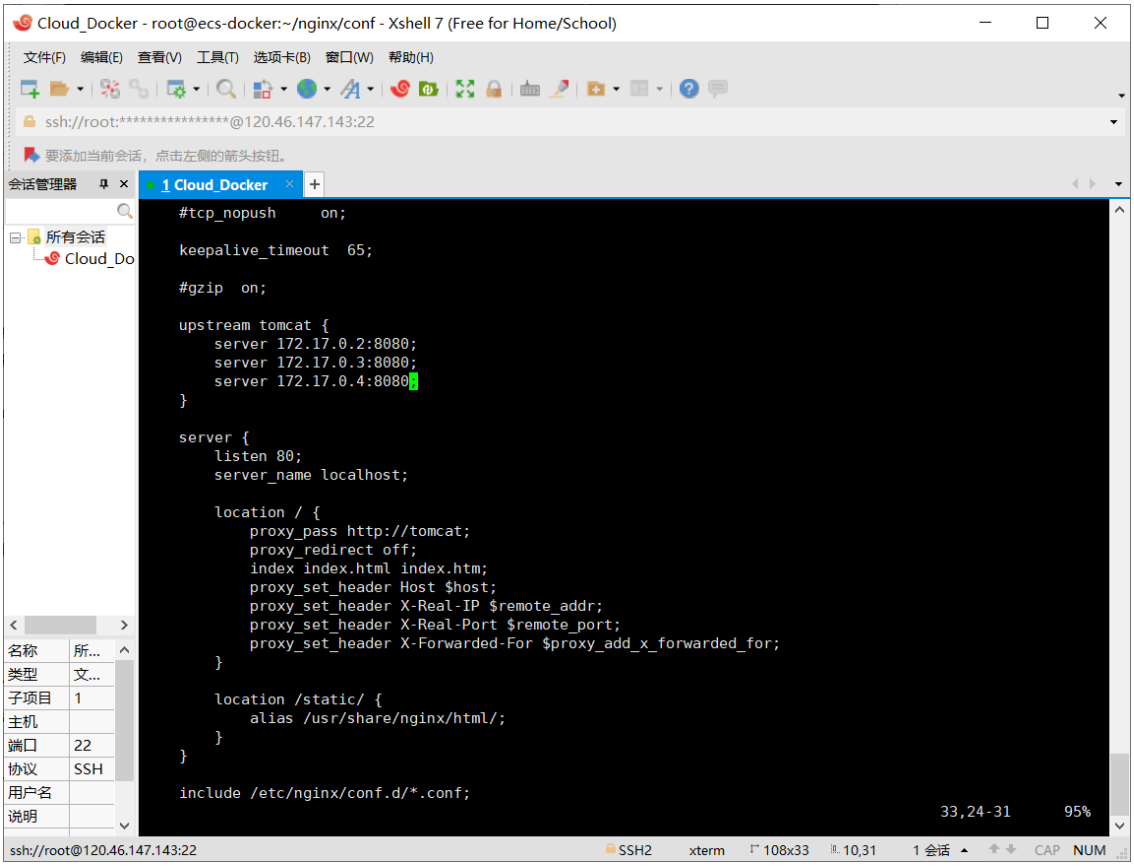
6. 配置相关文件

   路径: nginx/conf/nginx.conf

   nginx配置文件:

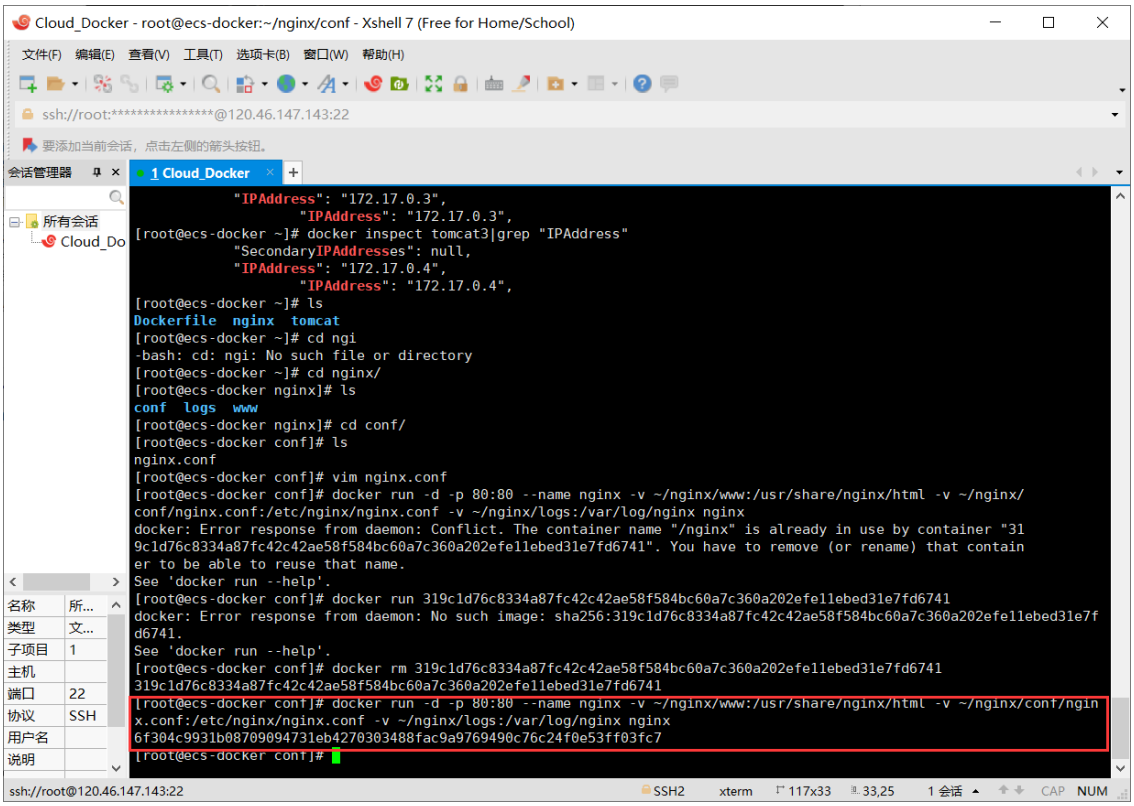
```
user  nginx;
worker_processes  1;
```

```
3
4  error_log  /var/log/nginx/error.log warn;
5  pid        /var/run/nginx.pid;
6
7
8  events {
9      worker_connections  1024;
10 }
11
12
13 http {
14     include       /etc/nginx/mime.types;
15     default_type  application/octet-stream;
16
17     log_format  main  '$remote_addr - $remote_user [$time_local]
   "$request" '
18                       '$status $body_bytes_sent "$http_referer" '
19                       '"$http_user_agent" "$http_x_forwarded_for"
   "$upstream_addr"';
20
21     access_log  /var/log/nginx/access.log  main;
22
23     sendfile        on;
24     #tcp_nopush     on;
25
26     keepalive_timeout  65;
27
28     #gzip  on;
29
30     upstream tomcat {
31         server 172.17.0.2:8080;
32         server 172.17.0.3:8080;
33         server 172.17.0.4:8080;
34     }
35
36     server {
37         listen 80;
38         server_name localhost;
39
40         location / {
41             proxy_pass http://tomcat;
42             proxy_redirect off;
43             index index.html index.htm;
44             proxy_set_header Host $host;
45             proxy_set_header X-Real-IP $remote_addr;
46             proxy_set_header X-Real-Port $remote_port;
47             proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
48         }
49
50         location /static/ {
51             alias /usr/share/nginx/html/;
52         }
53     }
54
55     include /etc/nginx/conf.d/*.conf;
56 }
```



7. 启动nginx：

```
docker run -d -p 80:80 --name nginx -v ~/nginx/www:/usr/share/nginx/html -v ~/nginx/conf/nginx.conf:/etc/nginx/nginx.conf -v ~/nginx/logs:/var/log/nginx nginx
```



nginx三种策略

轮询

- **说明**

  由于nginx的配置文件中,如果没添加额外的内容,默认的策略就是轮询,因此这里不再展示对应配置文件的内容(与上面提到的配置文件是相同的)

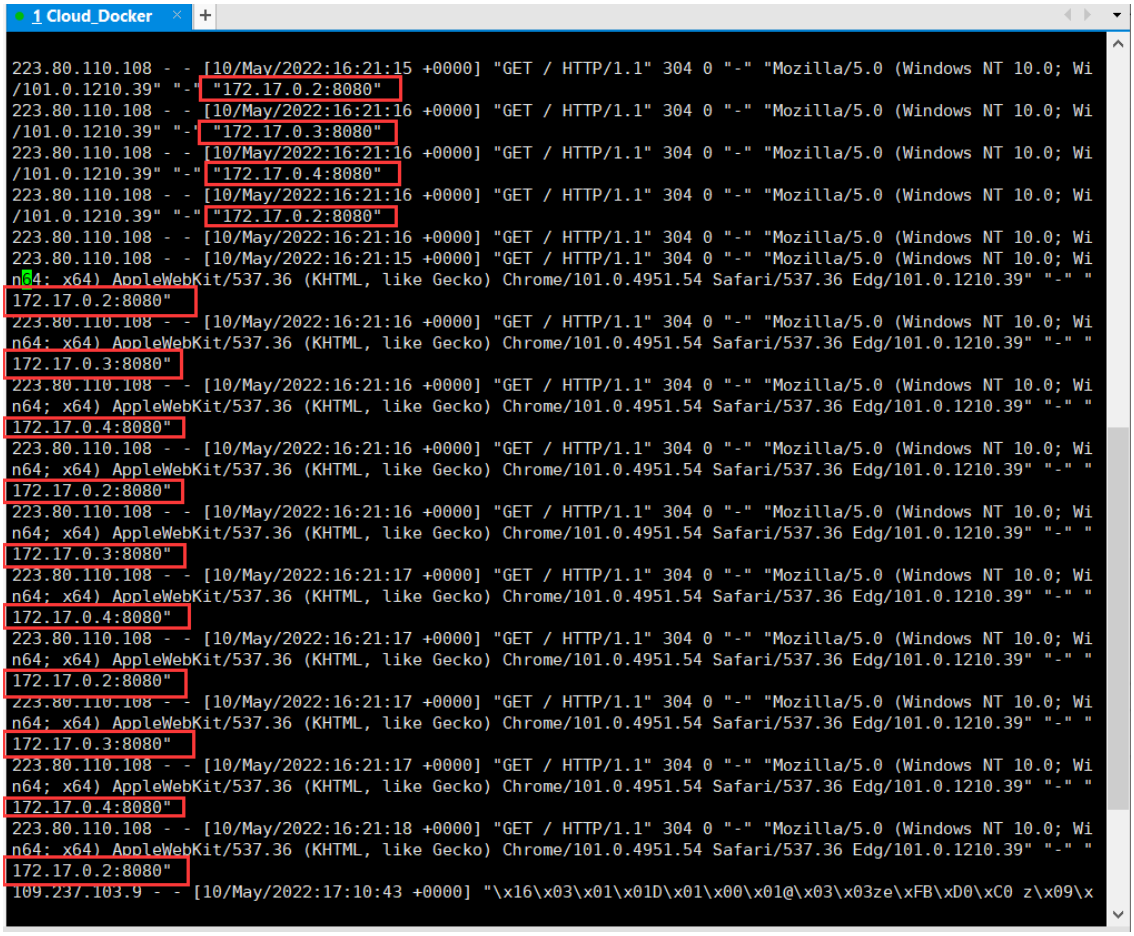
- **测试**

  尝试多次访问服务器地址:[docker deployment]

  查看nginx日志: `access.log` 文件

  根据下图,不难看出,在多次访问网站时,网站反馈得到的具体访问处理服务器是平均分配且顺序轮询的.

  即第一次访问对应于设置中给定的第一个服务器地址`172.17.0.2:8000`,第二次访问对应于设置中给定的第二个服务器地址`172.17.0.3:8000`,第三次依次类推...



反应到网页上,即有以下的结果:

以下三个页面**依次轮流**出现:

权重

- **修改**

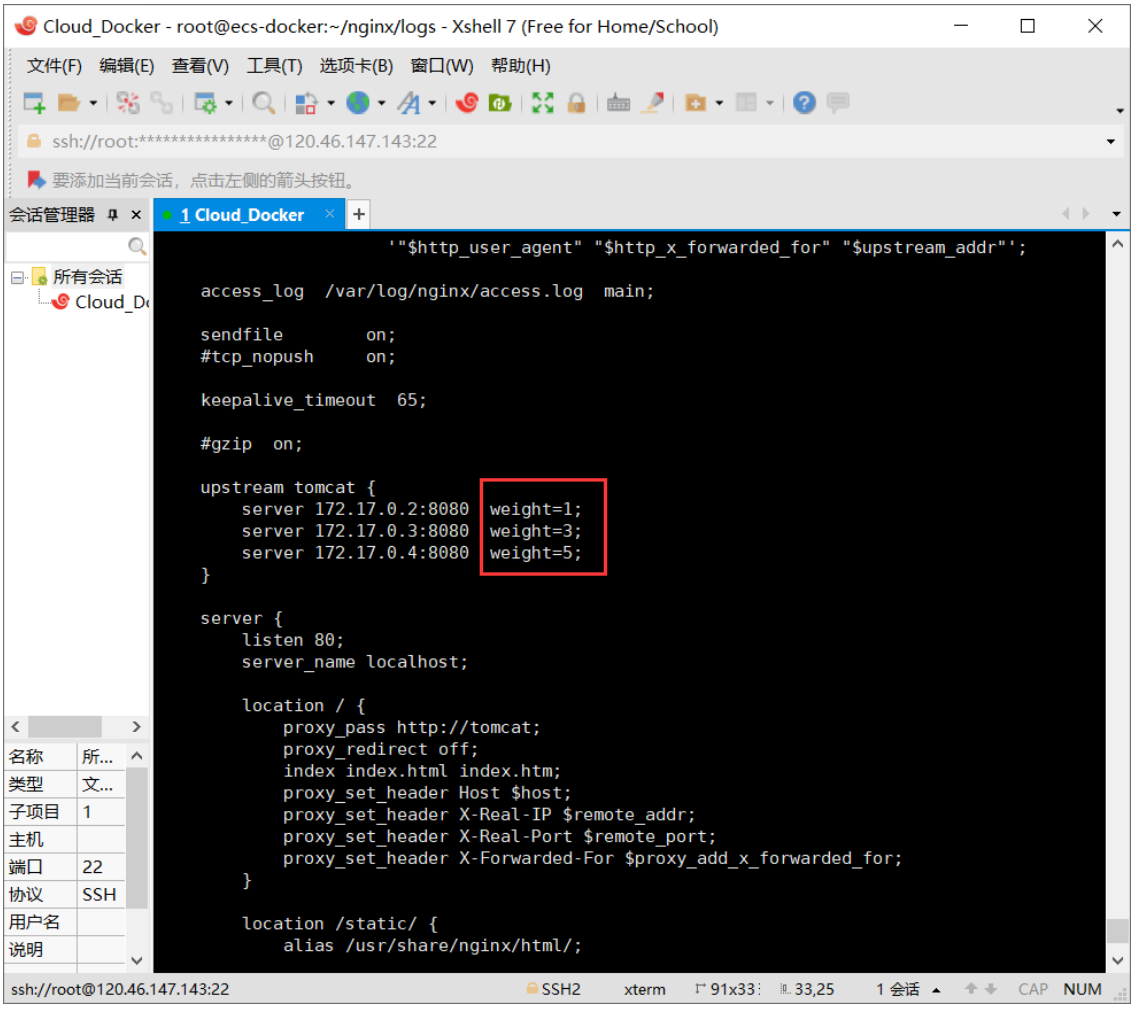
  按照实验指导书说明,设定三台服务器对应的权重信息为`1`,`2`,`3`

  修改完成后的`~/nginx/conf/nginx.json`内部代码内容如下所示:

```
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}

http {
    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" "$upstream_addr"';

```

```
    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    upstream tomcat {
        server 172.17.0.2:8080  weight=1;
        server 172.17.0.3:8080  weight=3;
        server 172.17.0.4:8080  weight=5;
    }

    server {
        listen 80;
        server_name localhost;

        location / {
            proxy_pass http://tomcat;
            proxy_redirect off;
            index index.html index.htm;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Real-Port $remote_port;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }

        location /static/ {
            alias /usr/share/nginx/html/;
        }
    }

    include /etc/nginx/conf.d/*.conf;
}
```

与前面的默认相比,新增了如下内容:



- **重启nginx**

  输入以下命令重启nginx

```
docker restart nginx
```

  同时,为了便于观察更改后的请求分配情况,这里将原本的请求日志清空:

```
rm access.log
touch access.log
```

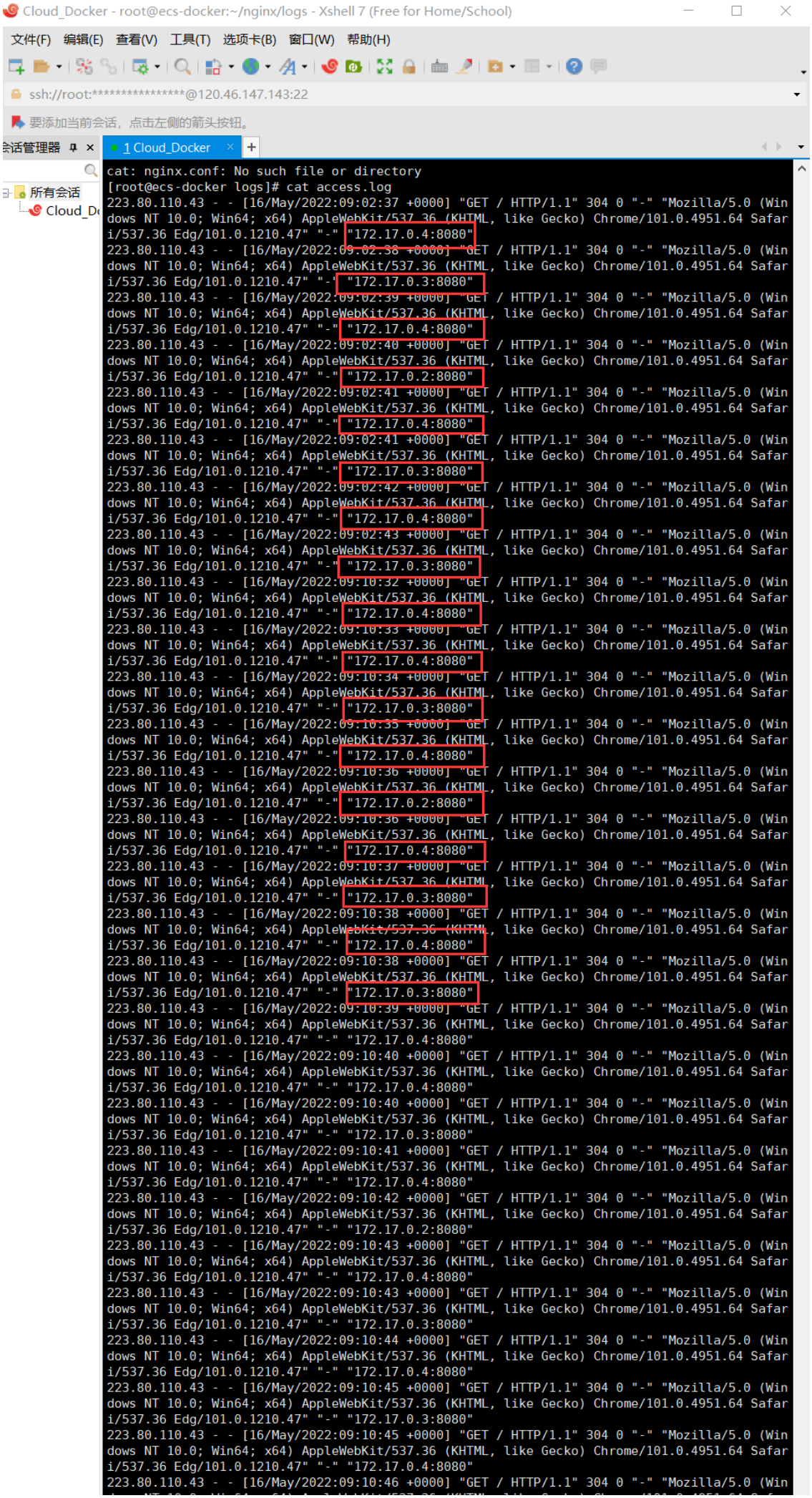
- **测试**

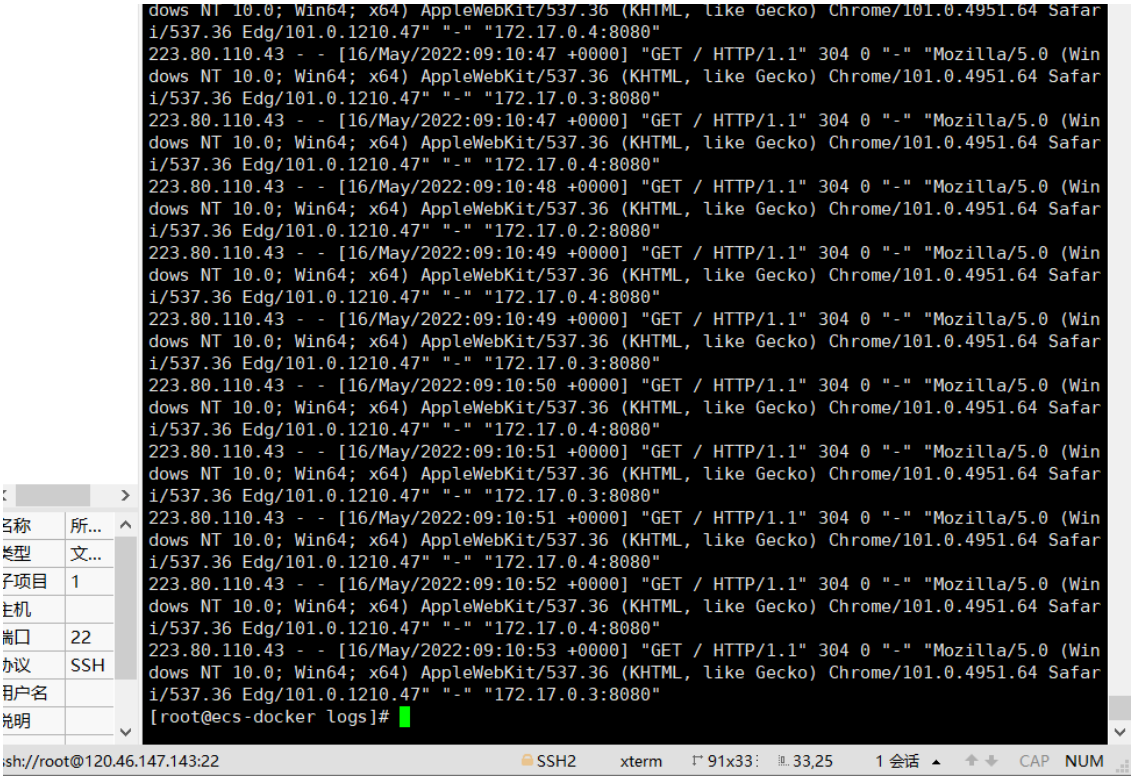
  在本地物理机多次访问服务器(35次左右)

  访问过程中,页面的具体变化表现为:

  三个页面随机出现,但是就概率而言，每个页面出现的概率是不同的。权重越大的服务器对应页面出现的概率越高。

  查看对应的日志文件`~/nginx/logs/access.log`,如下图所示.


Cloud_Docker - root@ecs-docker:~/nginx/logs - Xshell 7 (Free for Home/School)
文件(F) 编辑(E) 查看(V) 工具(T) 选项卡(B) 窗口(W) 帮助(H)
ssh://root:**********@120.46.147.143:22
要添加当前会话，点击左侧的箭头按钮。
会话管理器
所有会话
Cloud_D
1 Cloud_Docker
cat: nginx.conf: No such file or directory
[root@ecs-docker logs]# cat access.log
223.80.110.43 - - [16/May/2022:09:02:37 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:02:38 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.3:8080"
223.80.110.43 - - [16/May/2022:09:02:39 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:02:40 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.2:8080"
223.80.110.43 - - [16/May/2022:09:02:41 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:02:41 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.3:8080"
223.80.110.43 - - [16/May/2022:09:02:42 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:02:43 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.3:8080"
223.80.110.43 - - [16/May/2022:09:10:32 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:10:33 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:10:34 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.3:8080"
223.80.110.43 - - [16/May/2022:09:10:35 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:10:36 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.2:8080"
223.80.110.43 - - [16/May/2022:09:10:36 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:10:37 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.3:8080"
223.80.110.43 - - [16/May/2022:09:10:38 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:10:38 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.3:8080"
223.80.110.43 - - [16/May/2022:09:10:39 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:10:40 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:10:40 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.3:8080"
223.80.110.43 - - [16/May/2022:09:10:41 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:10:42 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.2:8080"
223.80.110.43 - - [16/May/2022:09:10:43 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:10:43 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.3:8080"
223.80.110.43 - - [16/May/2022:09:10:44 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:10:45 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.3:8080"
223.80.110.43 - - [16/May/2022:09:10:45 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win
dows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safar
i/537.36 Edg/101.0.1210.47" "-" "172.17.0.4:8080"
223.80.110.43 - - [16/May/2022:09:10:46 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Win



根据上图中表现出来的日志内容,可以统计得到每个服务器对应的访问次数如下所示:

IP地址	访问次数
172.17.0.4	21
172.17.0.3	12
172.17.0.2	4

可以确定:

- 权重越高分配到需要处理的请求越多。
- 处理的请求个数与对应的权重值成正比

  即:

  $4:12:21 \approx 1:3:5$

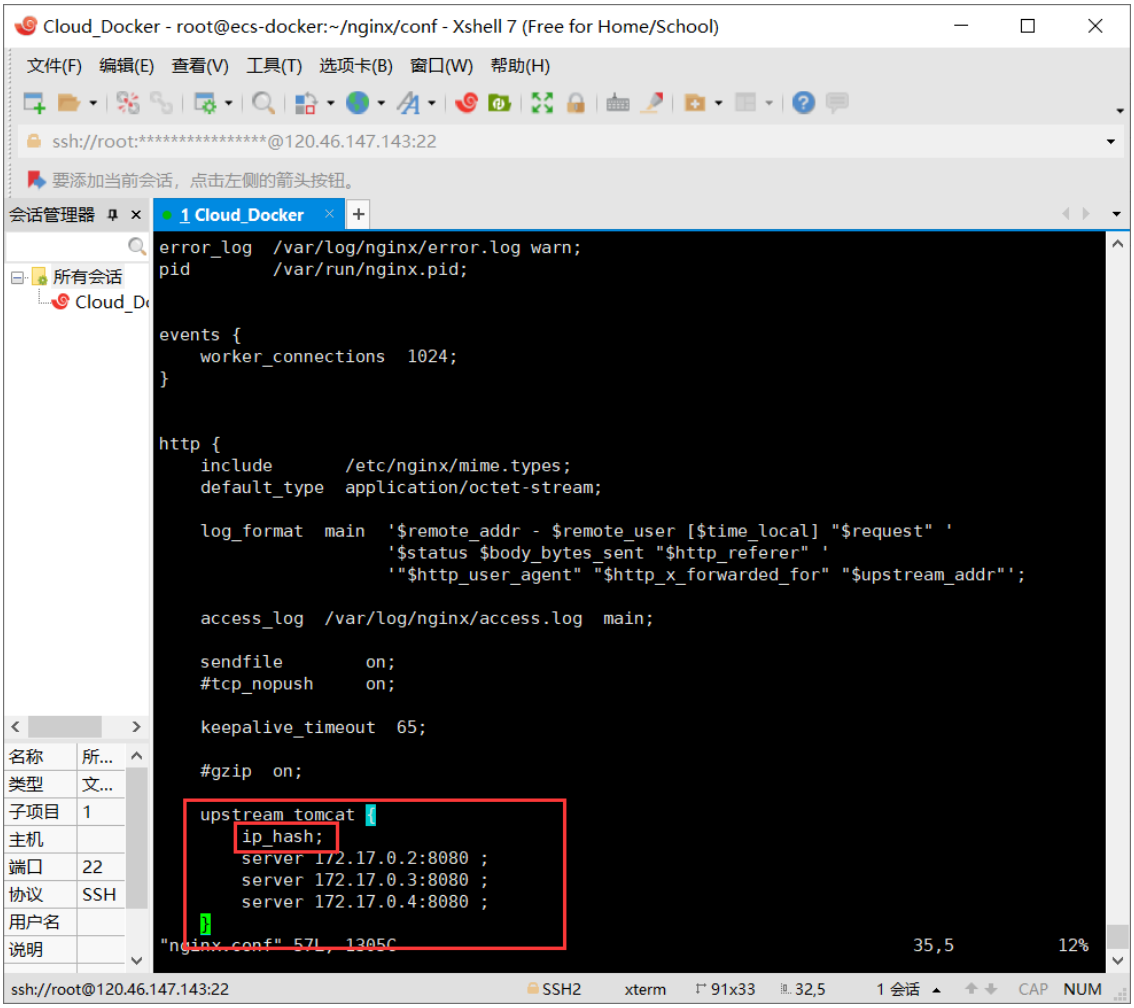
IP Hash

- **配置修改**

  这个修改相对容易,只需要在对应配置项中添加一行代码即可:

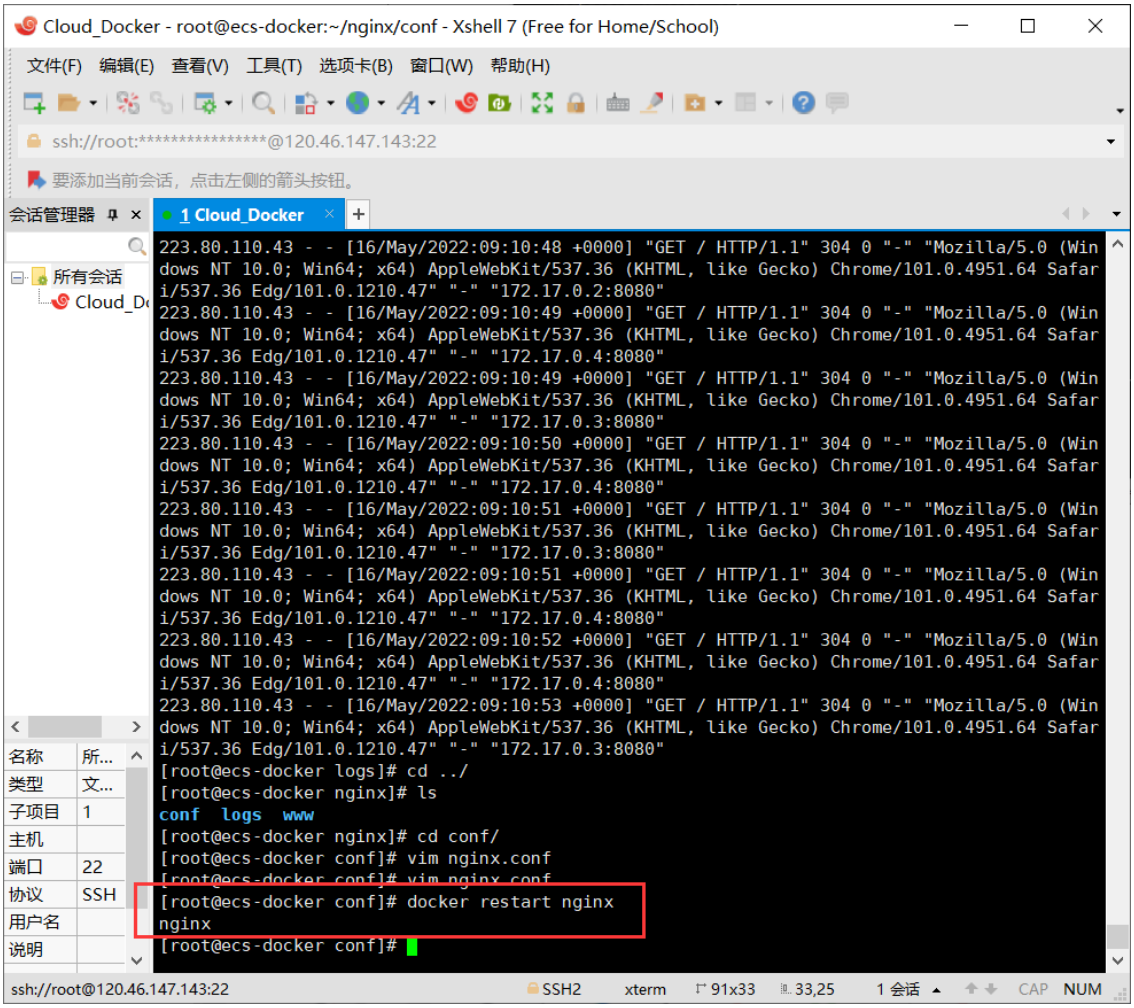
```
1  ip_hash;
```

修改后的配置文件基本与轮询的配置文件相同,不同内容如下:



- **重启nginx**

  该部分内容与上面相同,不过多赘述.



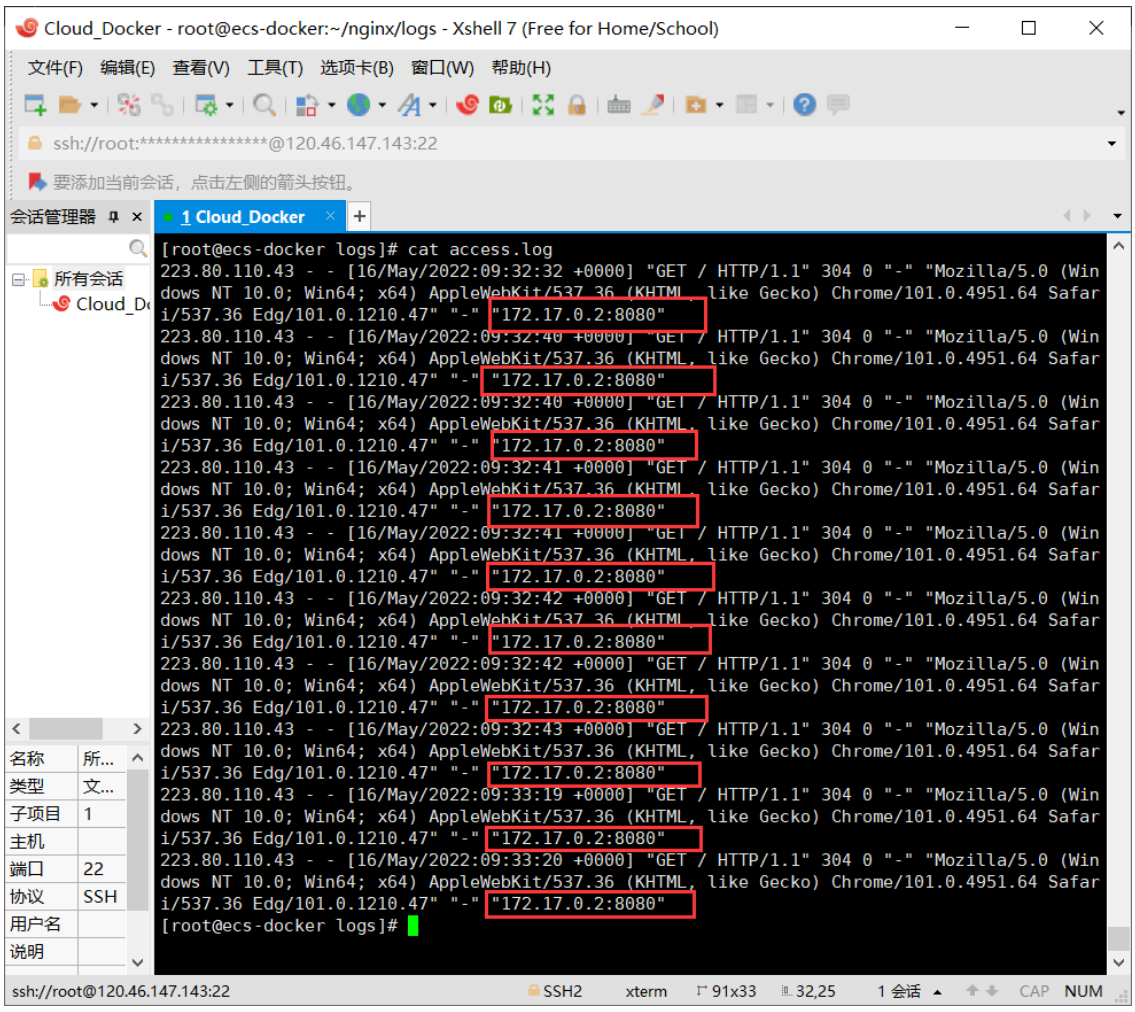
日志文件清空

- 测试

  从网页来看，每次访问时，如果主机确定，IP地址不发生变化，那么每次访问到的页面都是同一个页面。

  如下所示:

从当前主机以网页访问服务器几次,查看对应的日志文件如下所示:



由上图不难看出,当物理主机IP地址不改变时,其发送的请求对应的处理服务器是一开始就已经确定了的.后续的多次访问

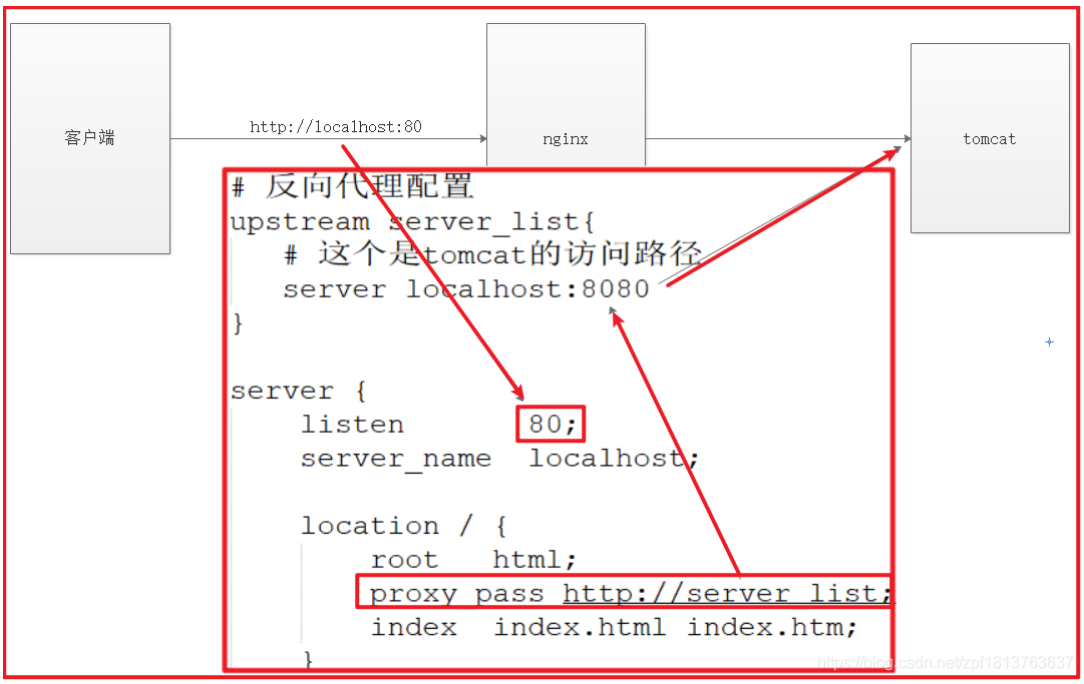
结论分析与体会

关于负载均衡与Nginx

所谓负载均衡就是：就是把大量的请求按照我们指定的方式均衡的分配给集群中的每台服务器，从而不会产生集群中大量请求只请求某一台服务器，从而使该服务器宕机的情况。

如果想要实现负载均衡,就需要在此之前实现反向代理。这里的反向代理是指，程序发送请求到某个域名，默认该请求会被Nginx获取，类似于DNS解析，Nginx会根据配置，将发送的请求转发到特定的服务器中。

实现反向代理的关键部分在于对Nginx配置文件的设置。上述过程反映到Nginx表现为下图：



其中配置文件中定义好的`upstream server_list{}`内部存放的是当前获取到访问请求后，可以转发到的服务器地址，服务器地址之间以`;`作为分隔符。

如果在对应服务器地址后添加属性`weight: x`代表将会执行权重分配的策略。

在整个列表的第一行添加`ip_hash`代表将会执行IP_HASH的分配策略。

三种负载均衡策略

轮询策略

定义：

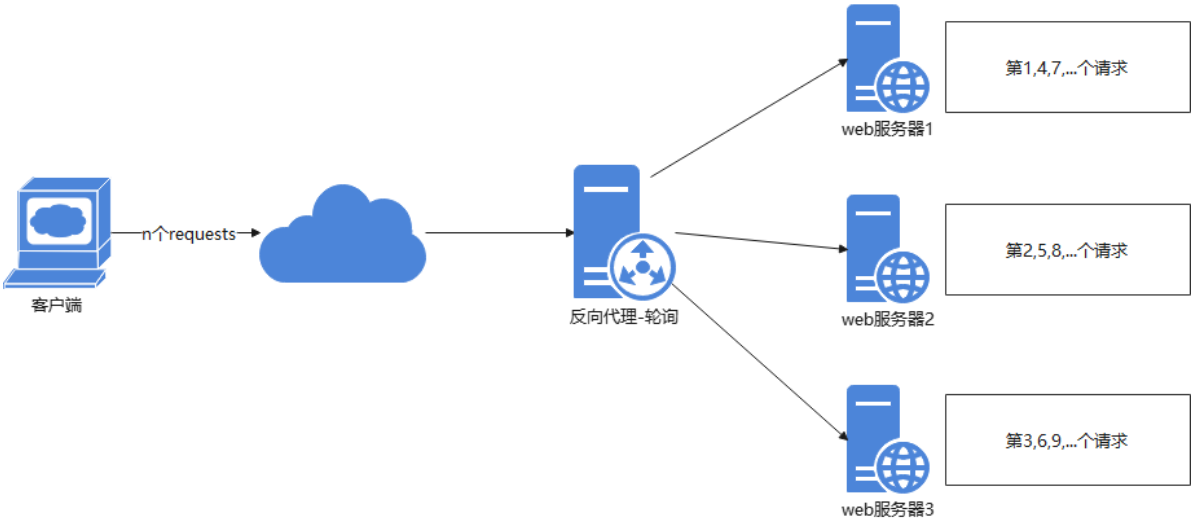
最基本的配置方法，它是upstream模块默认的负载均衡默认策略。每个请求会按时间顺序逐一分配到不同的后端服务器。含有以下参数：

属性	作用
fail_timeout	与max_fails结合使用。
max_fails	设置在fail_timeout参数设置的时间内最大失败次数，如果在这个时间内，所有针对该服务器的请求都失败了，那么认为该服务器会被认为是停机了，
fail_time	服务器会被认为停机的时间长度,默认为10s。
backup	标记该服务器为备用服务器。当主服务器停止时，请求会被发送到它这里。
down	标记服务器永久停机了。

注意点：

- 在轮询中，如果服务器down掉了，会自动剔除该服务器。
- 缺省配置就是轮询策略。
- 此策略适合服务器配置相当，无状态且短平快的服务使用。

示意图：



权重策略

定义：

权重方式，在轮询策略的基础上指定轮询的几率。

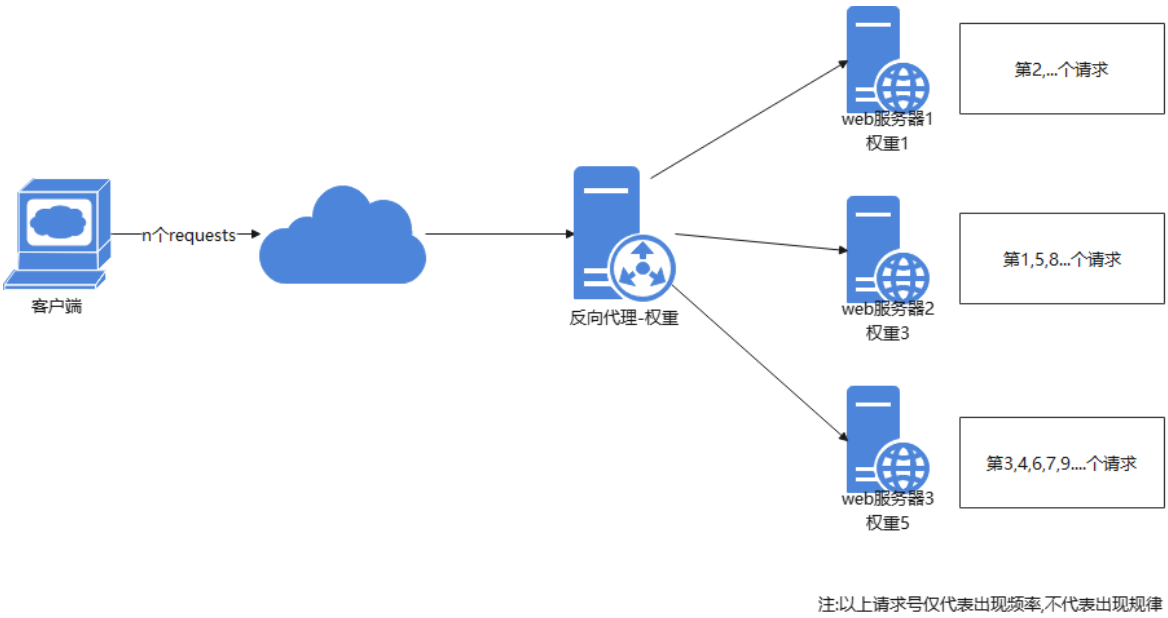
weight参数用于指定轮询几率，weight的默认值为1；

weight的数值与访问比率成正比，比如weight为2的服务器被访问的几率为weight为1的服务器的两倍。

注意点：

- 权重越高分配到需要处理的请求越多。
- 此策略可以与least_conn和ip_hash结合使用。
- 此策略比较适合服务器的**硬件配置差别比较大**的情况。硬件配置更高，可将其权重设置更大。

示意图：



IP_HASH策略

定义：

指定负载均衡器按照基于客户端IP的分配方式，这个方法确保了相同的客户端的请求一直发送到相同的服务器，以保证session会话。

每个访客都固定访问一个后端服务器，可以解决session不能跨服务器的问题。

注意点：

- 在nginx版本1.3.1之前，不能在ip_hash中使用权重（weight）。
- ip_hash不能与backup同时使用。
- 此策略适合有状态服务，比如session。
- 当有服务器需要剔除，必须手动down掉。

示意图：