

第1章 Nginx反向代理

1.什么是代理

- 1 现实生活中代理的场景：
- 2 房东
- 3 订票平台

3.Nginx反向代理介绍

- 1 反向代理是指将用户的请求向后面的服务器继续请求，并将后端服务器返回的数据返回给用户。
- 2 nginx除了可以做为高性能web服务以外，还可以提供反向代理和负载均衡的功能。

4.Nginx反向代理参数解释

官方网址：

- 1 https://nginx.org/en/docs/http/nginx_http_proxy_module.html#proxy_pass

参数解释：

- 1 `proxy_set_header Host $http_host;` #1b服务器将用户访问网站的HOST信息传递后端的web服务器
- 2 `proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;` #将用户的真实IP传递给后端的web服务器
- 3 `proxy_connect_timeout 30;` #代理与后端服务器连接超时时间(代理连接超时)
- 4 `proxy_send_timeout 60;` #后端服务器数据回传给nginx代理超时时间
- 5 `proxy_read_timeout 60;` #代理等待后端服务器的响应时间
- 6 `proxy_buffering on;` #把后端返回的内容先放到缓冲区当中，然后再返回给客户端,边收边传,不是全部接收完再传给客户端
- 7 `proxy_buffer_size 32k;` #设置nginx代理保存用户头信息的缓冲区大小
- 8 `proxy_buffers 4 128k;` #proxy_buffers缓冲区

5.Nginx反响代理配置文件优化

nginx的反向代理参数较多，如果每个server标签都写的话比较麻烦，所以我们可以将反向代理的配置写入文件中，以后server调用时只需要使用include引用即可。

- 1 `[root@1b-5 ~]# cat /etc/nginx/proxy_params`
- 2 `proxy_set_header Host $http_host;`
- 3 `proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;`
- 4 `proxy_connect_timeout 30;`
- 5 `proxy_send_timeout 60;`
- 6 `proxy_read_timeout 60;`
- 7 `proxy_buffering on;`
- 8 `proxy_buffer_size 32k;`
- 9 `proxy_buffers 4 128k;`

6.Nginx反向代理小试身手

6.1 实验需求

- 1 访问1b-5的80端口代理到web-7的8080端口

6.2 web-7的Nginx配置

```
1 cat > /etc/nginx/conf.d/web.conf<<EOF
2 server {
3     listen 8080;
4     server_name www.oldboyedu.com;
5     location / {
6         root /code;
7         index index.html;
8     }
9 }
10 EOF
11 mkdir /code -p
12 echo "$(hostname)" >/code/index.html
13 nginx -t
14 systemctl restart nginx
```

6.3 1b-5的Nginx配置

```
1 cat > /etc/nginx/conf.d/proxy.conf <<EOF
2 server {
3     listen 80;
4     server_name www.oldboyedu.com;
5     location / {
6         proxy_pass http://172.16.1.7:8080;
7         include proxy_params;
8     }
9 }
10 EOF
11 nginx -t
12 systemctl restart nginx
```

6.4 测试访问

- 1 curl 172.16.1.5

第2章 Nginx负载均衡

1.什么是负载均衡

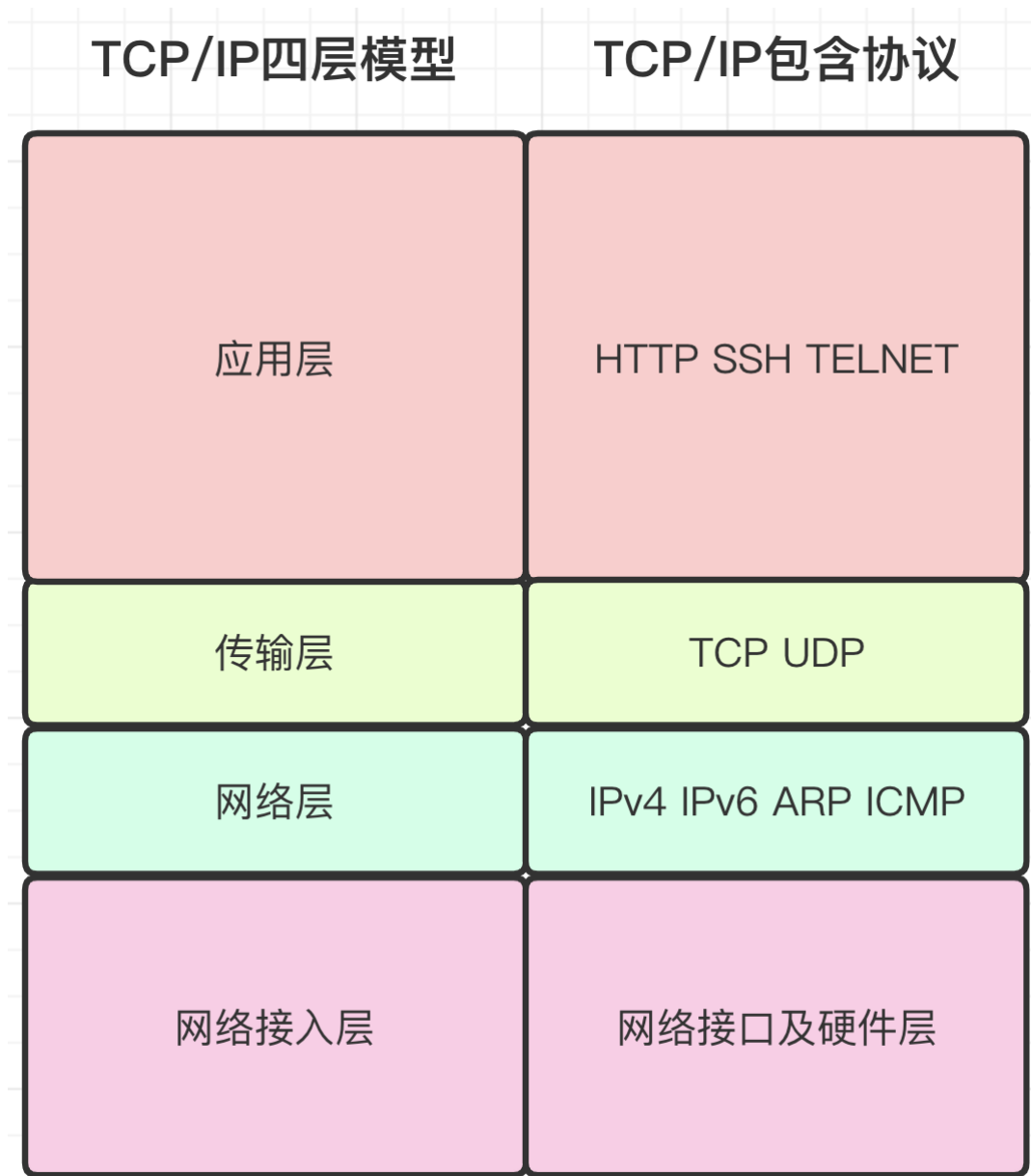
- 1 我们的web服务器直接面向用户，往往要承载大量并发请求，单台服务器难以负荷
- 2 我使用多台WEB服务器组成集群，前端使用Nginx负载均衡，将请求分散的打到我们的后端服务器集群中实现负载的分发。
- 3 那么会大大提升系统的吞吐率、请求性能、高容灾

2.负载均衡和反向代理的关系

- 1 因为反向代理多台机器，所以可以达到负载均衡的效果

3.负载均衡配置场景

3.1 TCP/IP四层模型



3.2 四层负载均衡

- 1 所谓四层负载均衡指的是OSI七层模型中的传输层
- 2 Nginx已经能支持TCP/IP的控制，所以只需要对客户端的请求进行TCP/IP协议的包转发就可以实现负载均衡
- 3 它的好处是性能非常快、只需要底层进行应用处理，而不需要进行一些复杂的逻辑

3.3 七层负载均衡

- 1 七层负载均衡它是在应用层，那么它可以完成很多应用方面的协议请求
- 2 比如我们说的http应用的负载均衡，它可以实现http信息的改写、头信息的改写、安全应用规则控制、URL匹配规则控制、以及转发,rewrite等等的规则
- 3 所以在应用层的服务里面，我们可以做的内容就更多，那么Nginx则是一个典型的七层负载均衡

3.4 四层与七层负载均衡小结

- 1 四层负载均衡数据包在底层就进行了分发，而七层负载均衡数据包则是在最顶层进行分发
- 2 由此可以看出，七层负载均衡效率没有四层负载均衡高。
- 3 但七层负载均衡更贴近于服务
- 4 如：`http` 协议就是七层协议，我们可以用Nginx可以作会话保持，URL路径规则匹配、`head`头改写等等
- 5 这些是四层负载均衡无法实现的

4.负载均衡配置语法

官方地址：

- 1 https://nginx.org/en/docs/http/nginx_http_upstream_module.html

配置语法：

```
1 upstream www_pools {                #定义后端服务器组名称
2     server 172.16.1.7;                #后端服务器地址
3     server 172.16.1.8;                #后端服务器地址
4 }
5
6 server {
7     listen 80;
8     server_name www.oldboyedu.com;
9     location / {
10         proxy_pass http://www_pools;    #如果匹配上就跳转到www_pools地址池
11         include proxy_params;          #包含反向代理参数
12     }
13 }
```

5.七层负载均衡小试身手

5.1 项目需求

- 1 访问`www.oldboyedu.com` 轮训跳转到`172.16.1.7`和`172.16.1.8`两台机器

5.2 主机目录规划

- 1 `lb-51 10.0.0.5`
- 2 `web-7 172.16.1.7`
- 3 `web-8 172.16.1.8`

5.3 web服务器nginx配置文件

```
1 cat >/etc/nginx/conf.d/www.conf <<EOF
2 server {
3     server_name www.oldboyedu.com;
4     listen 80;
5     root /code/;
6     index www.html;
7 }
8 EOF
9 nginx -t
10 systemctl restart nginx
```

5.4 web服务器生成测试页面

```
1 echo "${hostname} www" > /code/www.html
```

5.5 web服务器测试访问

```
1 curl -H 'Host:www.oldboyedu.com' 127.0.0.1
```

5.6 lb服务器nginx配置

```
1 upstream www_pools{
2     server 172.16.1.7;
3     server 172.16.1.8;
4 }
5
6 server {
7     listen 80;
8     server_name www.oldboyedu.com;
9     location / {
10         proxy_pass http://www_pools;
11         include proxy_params;
12     }
13 }
14 nginx -t
15 systemctl restart nginx
```

5.7 lb服务器访问测试

```
1 curl -H 'Host:www.oldzhang.com' 10.0.0.5
```

6.四层负载均衡小试身手

6.1 项目需求

- 1 1b-51和1b-52上分别安装了redis和mysql数据库
- 2 2.访问1b服务器的3306和6379可以代理到后端对应的数据库
- 3 3.ssh访问1b服务器的8000端口，代理到m-61的22端口上

6.2 lb-5服务器的nginx配置文件

```
1 cat > /etc/nginx/conf.d/redis.conf << 'EOF'
2 stream {
3     upstream redis_server {
4         server 172.16.1.51:6379 max_fails=3 fail_timeout=30s;
5         server 172.16.1.52:6379 max_fails=3 fail_timeout=30s;
6     }
7
8     upstream mysql_server {
9         server 172.16.1.51:3306 max_fails=3 fail_timeout=30s;
10        server 172.16.1.52:3306 max_fails=3 fail_timeout=30s;
11    }
12
13    server {
14        listen 10.0.0.7:6379;
15        proxy_pass redis_server;
16        include proxy_params;
17    }
18
19    server {
20        listen 10.0.0.7:3306;
21        proxy_pass mysql_server;
22        include proxy_params;
23    }
24
25    server {
26        listen 10.0.0.7:3306;
27        proxy_pass 172.16.1.61:22;
28        include proxy_params;
29    }
30 }
31 EOF
32 nginx -t
33 systemctl restart nginx
```

6.3 访问并测试

```
1 redis-cli -h 10.0.0.5
2 mysql -uroot -p123 -h10.0.0.5
```

第3章 负载均衡调度算法实验

1.负载均衡调度算法介绍

调度算法	概述
rr 轮询	按时间顺序逐一分配到不同的后端服务器
weight 加权轮询	按照权重分配请求，权重越大，分配到的几率越高
ip_hash 基于ip轮询	按照访问IP的hash结果分配，同一个IP固定访问同一个后端服务器
url_hash 基于url轮询	按照访问URL的hash结果分配，同一个URL固定访问同一个后端服务器
least_conn 最少连接数	按照后端服务器的连接数分配，谁连接数少就连谁

2.负载均衡调度算法-weight实验

```
1 upstream www_pools {
2     server 172.16.1.7 weight=1;
3     server 172.16.1.8 weight=2;
4 }
```

3.负载均衡调度算法-ip_hash实验

```
1 upstream www_pools {
2     ip_hash;
3     server 172.16.1.7 ;
4     server 172.16.1.8 ;
5 }
```

4.负载均衡调度算法-url_hash实验

```
1 upstream www_pools {
2     hash $request_uri;
3     server 172.16.1.7 ;
4     server 172.16.1.8 ;
5 }
```

5.测试命令

```
1 for i in {1..100};do curl -s -H "host:www.oldboyedu.com" 127.0.0.1;done |grep
web-8|wc -l
```

第4章 负载均衡配置参数实验

1.负载均衡配置参数解释

调度算法	概述
backup	配置了backup参数的服务器，只有同一组的其他服务器都不能访问了，最后才会使用
max_fails	允许请求失败的次数
down	配置了down参数的服务器不参与负载均衡
fail_timeout	经过max_fails失败后服务暂停时间
max_conns	限制最大的接收连接数

2.负载均衡配置参数-down实验

```

1 upstream www_pools {
2     server 172.16.1.7 down;
3     server 172.16.1.8 ;
4 }
```

3.负载均衡配置参数-backup实验

```

1 upstream www_pools {
2     server 172.16.1.7 backup;
3     server 172.16.1.8 ;
4 }
```

4.负载均衡配置参数-max_files和fail_tomeout实验

```

1 upstream www_pools {
2     server 172.16.1.7 weight=1 fail_timeout=5s max_fails=3;
3     server 172.16.1.8 weight=2 fail_timeout=5s max_fails=3;
4 }
```

第5章 Nginx反向代理实战

1.根据客户端类型转发

需求：

- 1 如果用户是iphone就跳转到iphone页面
- 2 如果用户是安卓就跳转到安卓页面
- 3 如果用户是pc就跳转到pc页面
- 4 如果用户是IE就返回403

web服务器nginx配置：

```

1 cat >/etc/nginx/conf.d/sj.conf <<EOF
2 server {
3     listen 8080;
4     server_name sj.oldboyedu.com;
5     location / {
```



```

6         root /code/android;
7         index index.html;
8     }
9 }
10 server {
11     listen 8081;
12     server_name sj.oldboyedu.com;
13     location / {
14         root /code/iphone;
15         index index.html;
16     }
17 }
18 server {
19     listen 8082;
20     server_name sj.oldboyedu.com;
21     location / {
22         root /code/pc;
23         index index.html;
24     }
25 }
26 EOF

```

生成测试页面：

```

1 mkdir -p /code/{android,iphone,pc}
2 echo "$(hostname) PC" > /code/pc/index.html
3 echo "$(hostname) Iphone" > /code/iphone/index.html
4 echo "$(hostname) Android" > /code/android/index.html
5 nginx -t
6 systemctl restart nginx

```

lb服务器配置nginx：

```

1 vim /etc/nginx/conf.d/sj.conf
2 upstream android {
3     server 172.16.1.8:8080;
4 }
5 upstream iphone {
6     server 172.16.1.7:8081;
7 }
8 upstream pc {
9     server 172.16.1.7:8082;
10    server 172.16.1.8:8082;
11 }
12
13 server {
14     listen 80;
15     server_name sj.oldboyedu.com;
16     location / {
17         #默认跳转至 pc 站点
18         proxy_pass http://pc;
19         include proxy_params;
20
21         #如果客户端是 Iphone 则跳转到 iphone 的资源池
22         if ($http_user_agent ~* "Iphone") {
23             proxy_pass http://iphone;

```

```

24     }
25
26     #如果客户端是 Android 则跳转到 android 的资源池
27     if ($http_user_agent ~* "Android"){
28         proxy_pass http://android;
29     }
30
31     #如果客户端是 IE 浏览器，则返回 403 错误。
32     if ($http_user_agent ~* "msie"){
33         return 403;
34     }
35 }
36 }

```

检查并重启nginx

```

1 nginx -t
2 systemctl restart nginx

```

测试访问

```

1 echo "10.0.0.5 sj.oldboy.com" >> /etc/hosts
2 curl sj.oldboy.com
3 curl -A "iphone" sj.oldboyedu.com
4 curl -A "android" sj.oldboyedu.com
5 curl -A "msie" sj.oldboyedu.com

```

2.根据文件类型转发

需求:

```

1 访问图片格式就跳转到web-7
2 访问其他地址就跳转到web-8

```

web服务器配置

```

1 cat >/etc/nginx/conf.d/www.conf <<EOF
2 server {
3     listen 80;
4     server_name www.oldboyedu.com;
5     location / {
6         root /code;
7         index index.html;
8     }
9 }
10 EOF

```

生成测试页面

```

1 echo "$(hostname) www" > /code/index.html
2 cd /code/ && wget -O sun.jpg
  http://pic.51yuansu.com/pic3/cover/02/27/64/59c008e1c7954_610.jpg

```

lb服务器nginx配置:

```
1 cat >/etc/nginx/conf.d/jpg.conf<<EOF
2 upstream static {
3     server 172.16.1.7;
4 }
5
6 upstream default {
7     server 172.16.1.8;
8 }
9
10 server {
11     listen 80;
12     server_name www.oldboyedu.com;
13     location / {
14         proxy_pass http://default;
15         include proxy_params;
16     }
17
18     location ~ .*.(gif|jpg|jpeg|png|bmp|swf|css|js)$ {
19         proxy_pass http://static;
20         include proxy_params;
21     }
22 }
23 EOF
```

访问测试:

```
1 curl www.oldboyedu.com
2 curl www.oldboyedu.com/sun.jpg
```

第6章 作业

配置lb服务器反向代理负载均衡wordpress和zh网站

实现关闭任意一台web服务器，用户仍然可以正常访问另一台