第1章 Nginx反向代理

1.什么是代理

```
现实生活中代理的场景:
房东
订票平台
```

3.Nginx反向代理介绍

```
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。
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:

```
https://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_pass
```

参数解释:

```
proxy_set_header Host $http_host;        #lb服务器将用户访问网站的HOST信息传递后后端的web服务器
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
proxy_connect_timeout 30;          #代理与后端服务器连接超时时间(代理连接超时)
proxy_send_timeout 60;             #后端服务器数据回传给nginx代理超时时间
proxy_read_timeout 60;             #代理等待后端服务器的响应时间
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
proxy_buffer_size 32k;             #设置nginx代理保存用户头信息的缓冲区大小
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。

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

6.Nginx反向代理小试身手

6.1 实验需求

```
访问lb-5的80端口代理到web-7的8080端口
```

6.2 web-7的Nginx配置

```
cat > /etc/nginx/conf.d/web.conf<<EOF
server {
    listen 8080;
    server_name www.oldboyedu.com;
    location / {
        root /code;
        index index.html;
    }
}
EOF
mkdir /code -p
echo "$(hostname)" >/code/index.html
nginx -t
systemctl restart nginx
```

6.3 lb-5的Nginx配置

```
cat > /etc/nginx/conf.d/proxy.conf <<EOF
server {
    listen 80;
    server_name www.oldboyedu.com;
    location / {
        proxy_pass http://172.16.1.7:8080;
        include proxy_params;
    }
}
EOF
nginx -t
systemctl restart nginx
```

6.4 测试访问

```
curl 172.16.1.5
```

第2章 Nginx负载均衡

1.什么是负载均衡

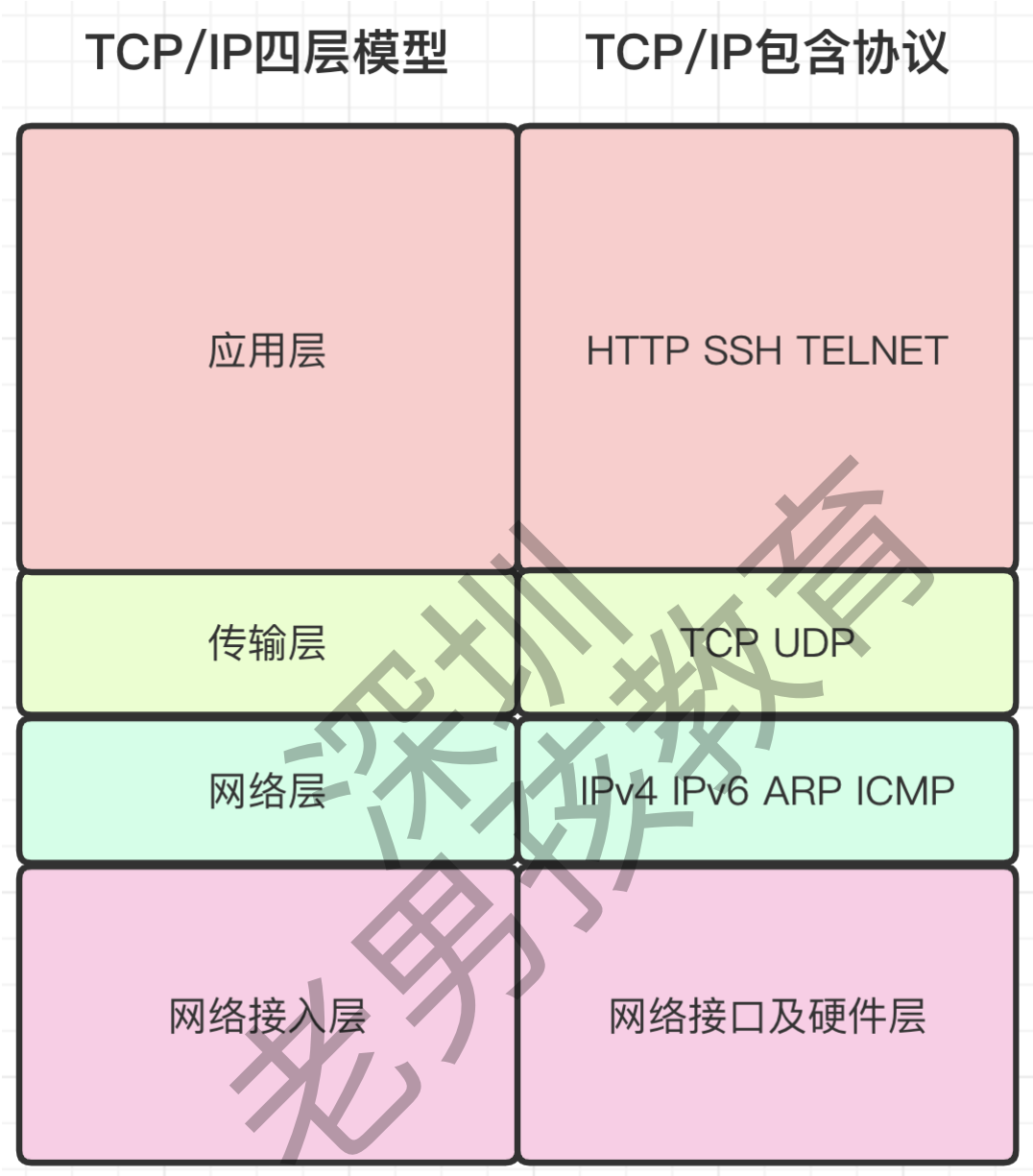
```
我们的Web服务器直接面向用户，往往要承载大量并发请求，单台服务器难以负荷
我使用多台WEB服务器组成集群，前端使用Nginx负载均衡，将请求分散的打到我们的后端服务器集群中实现负载的分发。
那么会大大提升系统的吞吐率、请求性能、高容灾
```

2.负载均衡和反向代理的关系

```
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

```
所谓四层负载均衡指的是OSI七层模型中的传输层
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
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

```
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
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
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

```
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
由此可以看出，七层负载均衡效率没有四负载均衡高。
但七层负载均衡更贴近于服务
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
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：

```
https://nginx.org/en/docs/http/ngx_http_upstream_module.html
```

配置语法：

```
upstream www_pools {                    #定义后端服务器组名称
    server 172.16.1.7;                #后端服务器地址
    server I72.16.1.8;                #后端服务器地址
}

server {
    listen 80;
    server_name www.oldboyedu.com;
    location / {
        proxy_pass http://www_pools;          #如果匹配上就跳转到www_pools地址池
        include proxy_params;                 #包含反向代理参数
    }
}
```

5.七层负载均衡小试身手

5.1 项目需求

```
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

```
lb-5l 10.0.0.5
web-7   172.16.1.7
web-8   172.16.1.8
```

5.3 web服务器nginx配置文件

```
cat >/etc/nginx/conf.d/www.conf <<EOF
server {
    server_name www.oldboyedu.com;
    listen 80;
    root /code/;
    index www.html;
}
EOF
nginx -t
systemctl restart nginx
```

5.4 web服务器生成测试页面

```
echo "$(hostname) www" > /code/www.html
```

5.5 web服务器测试访问

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

5.6 lb服务器nginx配置

```
upstream www_pools{
        server 172.16.1.7;
        server 172.16.1.8;
    }

server {
    listen 80;
    server_name www.oldboyedu.com;
    location / {
        proxy_pass http://www_pools;
        include proxy_params;
    }
}
nginx -t
systemctl restart nginx
```

5.7 lb服务器访问测试

```
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.lb-51和lb-52上分别安装了redis和mysql数据库
2.访问lb服务器的3306和6379可以代理到后端对应的数据库
3.ssh访问lb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

```
cat > /etc/nginx/conf.d/redis.conf << 'EOF'
stream {
    upstream redis_server {
        server 172.16.1.51:6379 max_fails=3 fail_timeout=30s;
        server 172.16.1.52:6379 max_fails=3 fail_timeout=30s;
    }

    upstream mysql_server {
        server 172.16.1.51:3306 max_fails=3 fail_timeout=30s;
        server 172.16.1.52:3306 max_fails=3 fail_timeout=30s;
    }

    server {
        listen 10.0.0.7:6379;
        proxy_pass redis_server;
        include proxy_params;
    }

    server {
        listen 10.0.0.7:3306;
        proxy_pass mysql_server;
        include proxy_params;
    }

    server {
        listen 10.0.0.7:3306;
        proxy_pass 172.16.1.61:22;
        include proxy_params;
    }
}
EOF
nginx -t
systemctl restart nginx
```

6.3 访问并测试

```
redis-cli -h 10.0.0.5
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

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

3.负载均衡调度算法-ip_hash实验

```
upstream www_pools {
   ip_hash;
   server 172.16.1.7 ;
   server 172.16.1.8 ;
}
```

4.负载均衡调度算法-url_hash实验

```
upstream www_pools {
   hash   $request_uri;
   server 172.16.1.7 ;
   server 172.16.1.8 ;
}
```

5.测试命令

```
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_failes	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

```
upstream www_pools {
   server 172.16.1.7 down;
   server 172.16.1.8 ;
}
```

3.负载均衡配置参数-backup实验

```
upstream www_pools {
   server 172.16.1.7 backup;
   server 172.16.1.8 ;
}
```

4.负载均衡配置参数-max_files和fail_tomeout实验

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

第5章 Nginx反向代理实战

1.根据客户端类型转发

需求：

```
如果用户是iphone就跳转到iphone页面
如果用户是安卓就跳转到安卓页面
如果用户是pc就跳转到pc页面
如果用户是IE就返回403
```

web服务器nginx配置：

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

```
        root /code/android;
        index index.html;
    }
}
server {
    listen 8081;
    server_name sj.oldboyedu.com;
    location / {
        root /code/iphone;
        index index.html;
    }
}
server {
    listen 8082;
    server_name sj.oldboyedu.com;
    location / {
        root /code/pc;
        index index.html;
    }
}
EOF
```

生成测试页面：

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

lb服务器配置nginx：

```
vim /etc/nginx/conf.d/sj.conf
upstream android {
    server 172.16.1.8:8080;
}
upstream iphone {
    server 172.16.1.7:8081;
}
upstream pc {
    server 172.16.1.7:8082;
    server 172.16.1.8:8082;
}

server {
    listen 80;
    server_name sj.oldboyedu.com;
    location / {
        #默认跳转至 pc 站点
        proxy_pass http://pc;
        include proxy_params;

        #如果客户端是 Iphone 则跳转到 iphone 的资源池
        if ($http_user_agent ~* "Iphone") {
            proxy_pass http://iphone;
```

```
        }

        #如果客户端是 Android 则跳转到 android 的资源池
        if ($http_user_agent ~* "Android"){
            proxy_pass http://android;
        }

        #如果客户端是 IE 浏览器，则返回 403 错误。
        if ($http_user_agent ~* "msie"){
            return 403;
        }
    }
}
```

检查并重启nginx

```
nginx -t
systemctl restart nginx
```

测试访问

```
echo "10.0.0.5 sj.oldboy.com" >> /etc/hosts
curl sj.oldboy.com
curl -A "iphone" sj.oldboyedu.com
curl -A "android" sj.oldboyedu.com
curl -A "msie" sj.oldboyedu.com
```

2.根据文件类型转发

需求:

```
访问图片格式就跳转到web-7
访问其他地址就跳转到web-8
```

web服务器配置

```
cat >/etc/nginx/conf.d/www.conf <<EOF
server {
    listen 80;
    server_name www.oldboyedu.com;
    location / {
        root /code;
        index index.html;
    }
}
EOF
```

生成测试页面

```
echo "$(hostname) www" > /code/index.html
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:

```
cat >/etc/nginx/conf.d/jpg.conf<<EOF
upstream static {
    server 172.16.1.7;
}

upstream default {
    server 172.16.1.8;
}

server {
    listen 80;
    server_name www.oldboyedu.com;
    location / {
        proxy_pass http://default;
        include proxy_params;
    }

    location ~ .*.(gif|jpg|jpeg|png|bmp|swf|css|js)$ {
        proxy_pass http://static;
        include proxy_params;
    }
}
EOF
```

访问测试：

```
curl www.oldboyedu.com
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