**案例1：Nginx反向代理**

**用nginx做调度器的作用：1，负载均衡（它会轮寻每台后端服务器）**

**2，web高可用**

**步骤1：部署后端web服务器**

**[root@web1 ~]# yum -y install httpd**

**[root@web1 ~]# systemctl restart httpd**

**[root@web1 ~]# systemctl enabel httpd**

**[root@web1 ~]# firewall-cmd --set-default-zone=trusted**

**[root@web1 ~]# echo "aaaa" > /var/www/html/index.html**

**[root@web2 ~]# yum -y install httpd**

**[root@web2 ~]# systemctl restart httpd**

**[root@web2 ~]# systemctl enabel httpd**

**[root@web2 ~]# firewall-cmd --set-default-zone=trusted**

**[root@web2 ~]# echo "bbbb" > /var/www/html/index.html**

**步骤2：修改nginx服务器配置（使用默认的主配置文件，用备份覆盖之前做的）**

**[root@proxy conf]# cp nginx.conf.default nginx.conf**

**[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf**

**#gzip on;**

**upstream webserver {**

**server 192.168.2.100:80;**

**server 192.168.2.200:80;**

**}**

**server {**

**listen 80;**

**server\_name localhost;**

**location / {**

**proxy\_pass <http://webserver;> #把原来的删掉，写入这一行**

**}**

**步骤3：重启nginx服务**

**[root@proxy ~]# nginx**

**[root@proxy ~]# nginx -s reload**

**步骤4：客户端测试**

**[root@client ~]# curl http://192.168.4.5**

**192.168.2.100**

**[root@client ~]# curl http://192.168.4.5**

**192.168.2.200**

**[root@client ~]# curl http://192.168.4.5**

**192.168.2.100**

**实验二：配置upstream服务器集群池属性**

**设置失败次数、超时时间、权重**

**[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf**

**#gzip on;**

**upstream webserver {**

**server 192.168.2.100 weight=1 max\_fails=1 fail\_timeout=10;**

**server 192.168.2.200 weight=2 max\_fails=2 fail\_timeout=10;**

**server 192.168.2.101 down;**

**#服务器权重(weight)，默认为1；**

**#max\_fails设置最大失败次数；**

**#fail\_timeout设置失败超时时间，单位为秒，10妙内不再测试;**

**#down 标记服务器已关机，不参与集群调度；**

**}**

**server {**

**listen 80;**

**server\_name localhost;**

**location / {**

**proxy\_pass <http://webserver;> #把原来的删掉，写入这一行**

**}**

**重启nginx服务**

**[root@proxy ~]# nginx -s reload**

**用客户端测试**

1. **先关闭web1的httpd服务**

**[root@web1 ~]# systemctl stop httpd**

1. **在客户端测试**

**[root@client ~]# curl http://192.168.4.5**

**192.168.2.200**

**[root@client ~]# curl http://192.168.4.5**

**192.168.2.200**

1. **再开启web1的httpd服务**

**[root@web1 ~]# systemctl start httpd**

1. **在客户端测试**

**[root@client ~]# curl http://192.168.4.5**

**192.168.2.100**

**[root@client ~]# curl http://192.168.4.5**

**192.168.2.200**

**[root@client ~]# curl http://192.168.4.5**

**192.168.2.200**

**实验三，配置upstream服务器集群的调度算法**

**步骤1，设置相同客户端访问相同web服务器**

**[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf**

**#gzip on;**

**upstream webserver {**

**Ip\_hash;**

**server 192.168.2.100 weight=1 max\_fails=1 fail\_timeout=10;**

**server 192.168.2.200 weight=2 max\_fails=2 fail\_timeout=10;**

**}**

**server {**

**listen 80;**

**server\_name localhost;**

**location / {**

**proxy\_pass <http://webserver;> #把原来的删掉，写入这一行**

**}**

**步骤2，重启nginx服务**

**[root@proxy ~]# nginx -s reload**

**步骤3，客户端测试**

**[root@client ~]# curl http://192.168.4.5**

**192.168.2.200**

**[root@client ~]# curl http://192.168.4.5**

**192.168.2.200**

**案例2：Nginx的TCP/UDP调度器**

**步骤一，部署Nginx服务器，并开启4层代理模块。**

**[root@proxy ~]# nginx -s stop**

**[root@proxy ~]# rm -rf /usr/local/nginx/ #删除老的nginx**

**[root@proxy ~]# cd nginx-1.12.2/**

**[root@proxy nginx-1.12.2]# ./configure \**

**> --user=nginx \**

**> --group=nginx \**

**> --with-http\_ssl\_module \ #开启ssl加密功能**

**> --with-stream #开启4层反向代理功能**

**[root@proxy nginx-1.12.2]# make && make install**

**修改配置文件（全部手工写入，位置必须是在http上面）**

**[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf**

**stream {**

**upstream sshs {**

**server 192.168.2.100:22;**

**server 192.168.2.200:22;**

**}**

**server {**

**listen 12345;**

**proxy\_pass sshs;**

**}**

**}**

**http {**

**......**

**}**

**步骤二，起服务，并查看端口**

**[root@proxy ~]# nginx**

**[root@proxy ~]# ss -ntualp | grep 12345**

**tcp LISTEN 0 128 \*:12345 \*:\* users:(("nginx",pid=11118,fd=6),("nginx",pid=11117,fd=6))**

**步骤三，客户端验证**

**[root@client ~]# ssh -p 12345 192.168.4.5**

**The authenticity of host '[192.168.4.5]:12345 ([192.168.4.5]:12345)' can't be established.**

**ECDSA key fingerprint is SHA256:59Byua15gv4TAVEm7YutMtmgXfxGTYbzSfO84kICu0E.**

**ECDSA key fingerprint is MD5:7e:6e:95:ed:84:fe:5b:8e:26:94:73:9b:c3:4b:90:cb.**

**Are you sure you want to continue connecting (yes/no)? yes**

**Warning: Permanently added '[192.168.4.5]:12345' (ECDSA) to the list of known hosts.**

**root@192.168.4.5's password: #这边的密码，输入的是为web1、web2的；**

**Last login: Fri Dec 7 15:28:49 2018 from 192.168.2.254**

**[root@web1 ~]#**

**登出，再次验证。发现会变化。**

**[root@web1 ~]# exit**

**[root@client ~]# ssh -p 12345 192.168.4.5**

**root@192.168.4.5's password: #这边的密码，输入的是为web1、web2的；**

**Last login: Fri Dec 7 15:27:50 2018 from 192.168.2.254**

**[root@web2 ~]#**

**案例三，Nginx常见问题处理**

**1，自定义报错页面**

1. **优化前，客户端访问一个不存在的页面，会提示404文件未找到**

**[root@client ~]# firefox http://192.168.4.5/test.html**

1. **修改Nginx配置文件，自定义报错页面**

**[root@proxy ~]# nginx**

**[root@proxy ~]# ss -ntulp | grep 80**

**tcp LISTEN 0 128 \*:80 \*:\* users:(("nginx",pid=1552,fd=7),("nginx",pid=1551,fd=7))**

**[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf**

**##找到下面这一行，去掉注释并修改**

**error\_page 404 /40x.html;**

**自主生成错误页面**

**[root@proxy ~]# vim /usr/local/nginx/html/40x.html**

**Oops,No no NO page...**

**重启nginx服务**

**[root@proxy ~]# nginx -s reload**

1. **客户端测试**

**#出现的页面内容，是我们自己定义的内容**

**[root@client ~]# firefox http://192.168.4.5/test.html**

**2，提高并发量（解决“Too many open files”提示问题）**

**#先测试出问题，如下一条**

**[root@proxy lnmp\_soft]# ab -c 2000 -n 2000 http://192.168.4.5/**

**socket: Too many open files (24)**

**[root@proxy lnmp\_soft]# vim /usr/local/nginx/conf/nginx.conf**

**worker\_processes 1; #数值与cpu内核数量相同就可。**

**events {**

**worker\_connections 1024; } #修改最大并发量数值**

**[root@proxy ~]# ulimit -a #查看所有的用户限制信息（-a:所有）**

**open files (-n) 1024**

**[root@proxy lnmp\_soft]# ulimit -Hn 100000 #硬限制，只有root可以修改**

**[root@proxy lnmp\_soft]# ulimit -Sn 100000 #软限制，普通用户可以修改（最大不能超过硬限制）；这两条命令在重启后就会重置为默认值1024！**

**[root@proxy ~]# vim /etc/security/limits.conf**

**#<domain> <type> <item> <value>**

**\* soft nofile 100000 #去掉注释**

**\* hard nofile 100000**

**再次查看结果(需退出ssh，再重新连接)**

**[root@proxy ~]# ulimit -a**

**open files (-n) 100000**

**3, 414报错（解决客户端访问头部信息过长的问题）**

**[root@proxy lnmp\_soft]# sh buffer.sh**

**414 Request-URI Too Large**

**[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf**

**http {**

**server\_tokens off;**

**client\_header\_buffer\_size 1K; #默认请求包头信息的缓存大小**

**large\_client\_header\_buffers 4 1M; #大请求包头部信息的缓存个数与容量，大小可改**

**4， 开启gzip压缩功能，提高数据传输效率**

**[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf**

**#接着前面的操作，写入下面的内容(仅对文档压缩，不对多媒体压缩)**

**gzip on;**

**gzip\_min\_length 1000;**

**gzip\_comp\_level 4;**

**gzip\_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss text/javascript;**

**5， 如何让客户端浏览器缓存数据(仅对静态数据)**

**[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf**

**server {**

**listen 80;**

**server\_name localhost;**

**location ~\* \.(jpg|png|gif)$ { #添加这3行内容**

**expires 30d;**

**}**

**6，如何查看服务器状态信息（非常重要的功能）**

**重新安装nginx**

**[root@proxy ~]# cd nginx-1.12.2/**

**[root@proxy nginx-1.12.2]# ./configure \**

**> --prefix=/usr/local/nginx \**

**> --user=nginx \**

**> --group=nginx \**

**> --with-http\_ssl\_module \**

**> --with-stream \**

**> --with-http\_stub\_status\_module**

**#如果没记住这个模块名，可以用下面的命令去查找**

**[root@proxy nginx-1.12.2]# ./configure --help | grep status**

**--with-http\_stub\_status\_module**

**[root@proxy nginx-1.12.2]# make && make install**

**启服务，并检测是否成功**

**[root@proxy nginx-1.12.2]# ln -s /usr/local/nginx/sbin/nginx /usr/bin/**

**[root@proxy nginx-1.12.2]# nginx**

**[root@proxy nginx-1.12.2]# ss -nutlp | grep 80**

**tcp LISTEN 0 128 \*:80 \*:\* users:(("nginx",pid=4887,fd=6),("nginx",pid=4886,fd=6))**

**修改配置文件，并重启nginx服务**

**[root@proxy nginx-1.12.2]# vim /usr/local/nginx/conf/nginx.conf**

**server {**

**listen 80;**

**server\_name localhost;**

**location /status { #只要在server里添加即可，/后面可写任意，**

**stub\_status on; #只要自己记住写的内容即可。**

**#allow IP地址； #这两行，在实际工作中再写**

**#deny IP地址；**

**}**

**[root@proxy nginx-1.12.2]# nginx -s reload**

**客户端测试**

**#刚才再配置文件里的新增的location /后面写的什么内容，这边就写什么**

**[root@client ~]# firefox <http://192.168.4.5/status>**

**Active connections: 1**

**server accepts handled requests**

**1 1 1**

**Reading: 0 Writing: 1 Waiting: 0**

**#解释：**

**Active connections：当前活动的连接数量。**

**Accepts：已经接受客户端的连接总数量。**

**Handled：已经处理客户端的连接总数量。**

**（一般与accepts一致，除非服务器限制了连接数量）。**

**Requests：客户端发送的请求数量。**

**Reading：当前服务器正在读取客户端请求头的数量。**

**Writing：当前服务器正在写响应信息的数量。**

**Waiting：当前多少客户端在等待服务器的响应。**

**7，日志切割**

**日志文件越来越大怎么办？单个文件10G? 如何切割？（非常常见的面试题）**

**步骤：1. 把旧的日志重命名**

**2. kill USR1 PID(nginx的进程PID号)**

**1）手动执行**

**#/usr/local/nginx/logs/nginx.pid文件中存放的是nginx的进程PID号。**

**[root@proxy ~]# cd /usr/local/nginx/logs/**

**[root@proxy logs]# mv access.log access2.log**

**[root@proxy logs]# kill -USR1 $(cat /usr/local/nginx/logs/nginx.pid)**

1. **自动完成（每周5的3点3分自动执行脚本完成日志切割工作）**

**[root@proxy ~]# cd /usr/local/nginx**

**[root@proxy nginx]# vim logbak.sh**

**#!/bin/bash**

**date=`date +%Y%m%d`**

**logpath=/usr/local/nginx/logs**

**mv $logpath/access.log $logpath/access-$date.log**

**mv $logpath/error.log $logpath/error-$date.log**

**kill -USR1 $(cat $logpath.nginx.pid)**

**[root@proxy nginx]# crontab -e**

**03 03 \* \* 5 /usr/local/nginx/logbak.sh**

**8，对页面进行压缩处理 (教学环境无效果，了解)**

**[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf**

**gzip on; #去掉注释“#”，并添加下面3行内容**

**gzip\_min\_length 1000; #小于1000字节的不压缩**

**gzip\_comp\_level 4; #压缩比率**

**gzip\_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss text/javascript;**

**#对特定文件压缩，类型参考mime.types**

**server {**

**9，服务器内存缓存 (教学环境下无效果，了解)**

**如果需要处理大量静态文件，可以将文件缓存再内存，下次访问会更快**

**[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf**

**http {**

**open\_file\_cache max=2000 inactive=20s;**

**open\_file\_cache\_valid 60s;**

**open\_file\_cache\_min\_uses 5;**

**open\_file\_cache\_errors off;**

**#设置服务器最大缓存2000个文件句柄，关闭20秒内无请求的文件句柄**

**#文件句柄的有效时间是60秒，60秒后过期**

**#只有访问次数超过5次会被缓存**