实验一：配置nginx，实现代理服务器功能

1. 准备好3台虚拟机，proxy web1 web2

yum -y install vim

yum -y install net-tools

yum -y install bash-completion

1. 为web1与web2安装httpd服务并启动

yum -y install httpd

systemctl restart httpd

1. 为web1与web2分别创建测试页面

[root@web1 ~]# echo web\_1 > /var/www/html/index.html

[root@web2 ~]# echo web\_2 > /var/www/html/index.html

之后使用proxy分别测试两个页面

[root@proxy nginx]# curl 192.168.2.100

web\_1

[root@proxy nginx]# curl 192.168.2.200

web\_2

1. 配置nginx服务器实现代理功能

修改proxy主机的nginx配置文件，按下图填写

upstream是定义集群，后面的webserver是集群名称可以自定义，最下方的proxy\_pass是调用集群，http后面写之前定义的集群名称



sbin/nginx -s reload 写完配置后重置服务

然后使用浏览器测试访问192.168.2.5可以看到web1与web2页面轮询显示

实验二：优化集群

1，解决用户重复登陆的问题

upstream webserver {

ip\_hash; 相同客户机访问相同服务器

server 192.168.2.100:80;

server 192.168.2.200:80;

}

sbin/nginx -s reload 写完配置后重置服务

再次使用浏览器测试发现不再轮询访问

2，健康检查

max\_fails可以定义检测失败多少次就不再轮询该主机

fail\_timeout可以定义多少秒检测主机的间隔时间

upstream webserver {

#ip\_hash;

server 192.168.2.100:80 max\_fails=2 fail\_timeout=20;

server 192.168.2.200:80;

}

3，修改权重,weight是权重，可以决定集群中主机被轮询的比率，性能较强的主机可以提高权重值

upstream webserver {

#ip\_hash;

server 192.168.2.100:80 weight=1 max\_fails=2 fail\_timeout=20;

server 192.168.2.200:80 weight=2;

}

sbin/nginx -s reload 写完配置后重置服务

再次使用浏览器测试发现轮询web2主机2次之后才会轮询web1主机

nginx功能 ： web服务器，web代理(将自身作为调度器指挥后台集群中的web业务主机)，四层代理(可以实现非web业务的集群)

实验三：使用nginx实现四层代理

1，删除现有nginx

sbin/nginx -s stop 停服务

rm -rf /usr/local/nginx/ 删除

2，回到nginx源码包目录，从新编译

--with-stream 开启四层代理的模块

--with-http\_stub\_status\_module 可以查看nginx网站后台数据的模块

./configure --with-http\_ssl\_module --with-stream --with-http\_stub\_status\_module

make

make install

/usr/local/nginx/sbin/nginx -V 检测

./configure --help | grep stub 可以帮助搜索对应的模块名称

3，修改配置，添加集群，调用集群

stream { 在第16行开始添加，包括以下内容

upstream backend { 定义集群

server 192.168.2.100:22; 集群中的主机以及业务端口号

server 192.168.2.200:22;

}

server { 调用集群

listen 12345; 监听端口号，可以自定义

proxy\_pass backend; 调用后台集群的名称

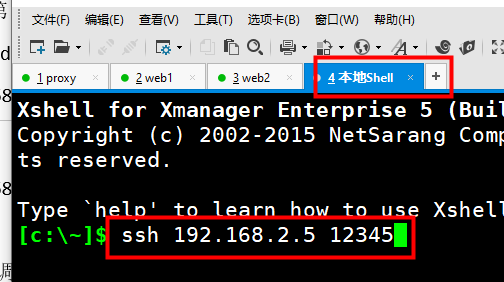
}

}

配置完后，开启服务

netstat -ntulp | grep nginx 检测看12345端口是否开启

之后使用xshell开启新窗口ssh 192.168.2.5 12345链接集群



注意要把web1与web2的密码设置成一致的，并且登录时输入该密码，而不是输入proxy的

实验四：nginx问题处理

1，404页面报错问题

修改proxy的nginx主配置文件，在第58左右修改，当用户访问了不存在的页面就显示test.jpg图片

error\_page 404 /test.jpg;

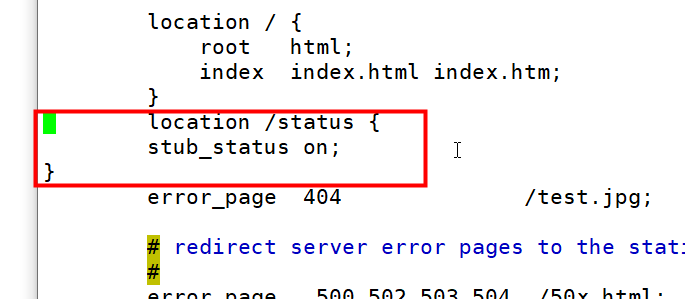
之后重置服务

在找个图片放入/usr/local/nginx/html中

然后使用火狐随意访问一个不存在的页面，比如192.168.2.5/xxx

2，开启nginx状态信息

在配置文件中添加以下内容



重置nginx服务

之后使用火狐浏览器打开192.168.2.5/status

Active connections: 1 访问用户数量

server accepts handled requests 这里的accepts代表接受客户端连接请求的数量，handled代表已经连接完毕的客户端数量，requests代表客户端发送的请求 ，分别可以用下面3个数值表示

10 10 12

Reading: 0 服务器正在读取客户请求的数量 Writing: 1 服务器正在回应给客户机信息的数量 Waiting: 0 有多少客户在等待服务器回应

3， 优化nginx的并发

ab 网站压力测试工具 来自httpd-tools软件包

ab -c 1000 -n 1000 <http://192.168.2.5/> 使用压力测试工具ab模拟1000个人，发出1000次访问请求给服务器

打开nginx主配置文件

worker\_processes 2; nginx开启的子程序数量，通常与cpu核心数相当

worker\_connections 50000; 每个子程序支持的并发访问量

sbin/nginx -s reload 重置服务

ulimit -n 查询linux服务器对文件打开数量的限制，默认1024

ulimit -Hn 100000 修改限制，扩大到10万

ulimit -Sn 100000

ulimit -n 通过上述命令修改后再次查询

ab -c 3000 -n 3000 <http://192.168.2.5/> 再次测试，发现nginx以及可以支持更多并发

[root@proxy ~]# vim /etc/security/limits.conf 如果打算永久保留打开文件数量的限制效果，可以通过此文件，填写如下内容，配置完毕后需要重启服务器，\*是可以让任何用户对此配置生效,soft是软件限制，hard是硬件限制，都要放开，nofile指的是打开文件数量的配置

\* soft nofile 100000

\* hard nofile 100000