operation day 03

================================================

**地址重写相关选项:**

last 不再读其他rewrite

break 不再读其他语句

redirect 临时重定向 状态码 302

permanent 永久重定向 状态码 301

**测试一: 重定向**

cd /usr/local/nginx

echo "nginx A~~" > html/a.html //准备素材

echo "nginx B~~" > html/b.html

echo "nginx C~~" > html/c.html

cd conf

cp nginx.conf.default nginx.conf //把配置文件恢复默认

cd ..

vim conf/nginx.conf //修改配置

rewrite /a.html /b.html redirect; //在42行添加

然后开启或者重加载nginx服务

rewrite /a.html /b.html permanent; //再次修改配置

重加载nginx服务

curl 192.168.2.5/a.html //此时看到的页面也是b，说明redirect与permanent效果在客户机看来是一样的，但是状态码不一样，对于搜索引擎来说更关心301的

**测试二：**

**last 不再读其他rewrite**

打开主配置文件，在42行修改

rewrite /a.html /b.html last; //如果没有last,看a页面会得到c页面

的内容,加了last之后,就不会再读后面的rewrite了

rewrite /b.html /c.html;

sbin/nginx -s reload //重加载nginx服务

使用火狐访问192.168.2.5/a.html 看到的是b页面

**break 不再读其他语句**

**按下列方式修改配置文件,将默认的location中加入rewrite语句,然后**

**再创建一个新的location也加入rewrite语句**

vim conf/nginx.conf //打开配置文件，修改为以下状态，首先

在默认的location中添加rewrite，再新创建一个location也添加

rewrite语句。

location / { //此处为默认的

rewrite /a.html /b.html break; //如果这里的选项依然用last,则仅仅会阻止

该location之内的rewrite语句,换成break就可以阻止之后的所有rewrite语句

root html;

index index.html index.htm;

}

location /b.html { //这里是新添加的

rewrite /b.html /c.html;

}

-------------------------------------------------

**用nginx实现网站代理功能 (组建网站集群)**

**一台服务器的能力是有限的，如果客户访问量比较大，可以利用nginx的代理功能组建集群，集群中的服务器越多集群整体性能就越强**

集群

proxy(代理服务器) 192.168.2.5

web1 192.168.2.100

web2 192.168.2.200

**1，集群中服务器的准备工作，这里使用web1**

yum -y install vim

yum -y install bash-completion //tab键补全软件包

yum -y install net-tools //网络工具软件包，支持ifconfig等命令

yum -y install httpd

echo "httpd-web1~~" > /var/www/html/index.html //这里

如果是web2的话，要改成"httpd-web2~~"

systemctl restart httpd

systemctl stop firewalld

**之后web2的配置与web1基本一致**

**之后到proxy主机测试:**

[root@proxy nginx]# curl 192.168.2.100

httpd\_web1~~~

[root@proxy nginx]# curl 192.168.2.200

httpd\_web2~~~

**2，在proxy主机添加创建集群配置**

**打开nginx配置文件，可以删除原有的地址重写的配置**

首先是34~37行

upstream web { //创建nginx集群，名称是web

server 192.168.2.100:80; //这里是集群中的服务器ip与端口

server 192.168.2.200:80; //第二台集群主机

}

然后在47行

47 proxy\_pass http://web; //调用web集群

sbin/nginx -s reload //重加载nginx服务

使用火狐访问192.168.2.5 不断刷新，看到的页面内容是web1与web2之间切换

或者使用curl 192.168.2.5

**集群的优化**

1. **任务量分配**

server 192.168.2.200:80 weight=2; //设置权重，值越大，分配

的工作量越多

sbin/nginx -s reload //重加载nginx服务

1. **健康检查功能**

server 192.168.2.200:80 max\_fails=2 fail\_timeout=30; //检测2次如

果失败，则认为集群中的服务器故障,认为集群中的服务器故障之

后等待30秒才会再次链接

sbin/nginx -s reload //重加载nginx服务

1. **相同客户机访问相同服务器**

upstream web {

ip\_hash; //相同客户机访问相同服务器，让一个客户机访问集群时锁定

一个后台服务器，避免重复登陆的问题

server 192.168.2.100:80;

server 192.168.2.200:80 max\_fails=2 fail\_timeout=30;

}

sbin/nginx -s reload //重加载配置

再次访问192.168.2.5，就不会再轮询访问后台集群主机了,只会咬住一台

1. **集群主机需要维护时**

server 192.168.2.100:80 down; //加down标记，使集群服务器暂时

不参与任务轮询

sbin/nginx -s reload //重加载配置

再次访问192.168.2.5，再也不会出现web1

------------------------------------------------------------------------------------

**使用nginx实现其他业务的集群**

**通常情况下nginx是搭建网站的工具，还可以组建网站集群，但如果后端的集群服务器跑的不是网站业务，就可以利用--with-stream模块创建非网站业务的集群。--with-http\_stub\_status\_module是之后实验需要的模块，可以查看网站内部数据。**

1. **准备工作**

sbin/nginx -s stop //停止nginx

cd /root/lnmp\_soft/nginx-1.17.6

rm -rf /usr/local/nginx

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

make

make install

[root@proxy nginx-1.17.6]# cd /usr/local/nginx/

[root@proxy nginx]# sbin/nginx -V //查询nginx版本与安装的模块

nginx version: nginx/1.17.6

built by gcc 4.8.5 20150623 (Red Hat 4.8.5-28) (GCC)

configure arguments: --with-stream --with-http\_stub\_status\_module

**2，创建集群**

打开nginx主配置文件，在16行左右(http上面)，添加以下内容

stream { //创建新业务

upstream backend { //创建名叫backend的集群

server 192.168.2.100:22; //集群中的主机使用22端口对外提供服务

server 192.168.2.200:22;

}

server {

listen 12345; //监听端口号

proxy\_pass backend; //调用集群

}

}

写完后开启nginx服务，如果服务已经开启，需要重新加载配置

ssh 192.168.2.5 -p 12345 //在proxy主机使用12345端口号连接代理服务器，可以登录web1或者web2，之后使用exit退出回到proxy

[root@proxy nginx]# rm -rf ~/.ssh/known\_hosts //先删除远程连接的主机记录，之

后每次连接测试都先删除该文件

ssh 192.168.2.5 -p 12345 //再次连接就会轮询到另一台集群主机比如web2

此处使用自己连接自己测试，如果使用client主机就远程连接192.168.4.5

------------------------------------------------------

**ss命令可以查看系统中启动的端口信息，该命令常用选项如下：**

-a显示所有端口的信息

-n以数字格式显示端口号

-t显示TCP连接的端口

-u显示UDP连接的端口

-l显示服务正在监听的端口信息，如httpd启动后，会一直监听80端口

-p显示监听端口的服务名称是什么（也就是程序名称）

**注意：在RHEL7系统中可以使用ss命令替代netstat命令，功能一样，选**

**项一样。**

------------------------------------------------------------------

**nginx 常见问题处理**

**1，404报错**

**客户访问网站时，如果看到了不存在的页面会有404报错的英文提示，这种**

**提示很不友好，可以通过自定义页面改善用户体验**

首先修改配置文件 大概58~59行附近

error\_page 404 /test.jpg; //如果客户访问了不存在的页面

就显示test.jpg的内容

保存退出

找一张图片，内容随意，比如用中文标注"抱歉！您访问的页面不存在" 然后保

存成test.jpg格式，然后拷贝到proxy主机的/usr/local/nginx/html目录下

重新加载nginx配置

使用浏览器虽随意访问不存在的页面192.168.2.5/XXXX.html 就可以看到之前

那张图片的内容

1. **查看网站后台数据(重要功能！)**

该实验需要模块 ： --with-http\_stub\_status\_module

修改配置文件，在 "error\_page 404" 那一行的上面添加：

location /status { //当用户输入的地址后面跟了/status之后

stub\_status on; //开启网站后台状态信息查看功能

allow 192.168.2.5; //仅仅允许2.5查看

deny all; //拒绝其他所有主机

}

sbin/nginx -s reload //重加载配置

<http://192.168.2.5/status>

页面中各个字段的含义：

Active connections：当前活动的连接数量（当前有多少用户访问该

网站）。

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。

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。