静态网站 用户访问环境无论是否变化，网站内容不会发生变化

动态网站 用户访问环境发生变化时，网站的内容可能也会发生变化

LNMP

linux+nginx+mariadb+php

实验一：部署LNMP环境

rm -rf /usr/local/nginx/ 删除了nginx所有数据

netstat -ntulp | grep :80 发现程序忘记关闭了

/usr/local/nginx/sbin/nginx -s stop 再次关闭报错，关闭不了nginx，因为程序已经删除

kill 30184 可以使用kill命令让nginx的程序退出，后面的30184是随机的nginx的进程号，通过上面netstat可以查询到

killall nginx 使用killall命令结束所有关于nginx的进程，而无需进程号，但是killal命令可能需要提前安装

yum -y install psmisc-22.20-15.el7.x86\_64 安装支持killall命令的软件包

1，安装nginx

yum -y install gcc pcre-devel openssl-devel

./configure --with-http\_ssl\_module

make

make install

yum -y install mariadb mariadb-server mariadb-devel

yum -y install php php-mysql php-fpm

内容：shell脚本 静态页面(html) 动态页面

工具：bash 浏览器 php

2，开启相关服务

[root@proxy nginx]# sbin/nginx 开启nginx

systemctl restart mariadb 开启数据库

systemctl enable mariadb

netstat -ntulp | grep 3306

systemctl restart php-fpm 开启动态页面后台解析服务

systemctl enable php-fpm

netstat -ntulp | grep 9000

3，实现nginx动静分离

nginx默认只支持静态页面

用户---nginx---静态页面 ----用户

如果需要nginx支持动态页面

用户---nginx---动态页面---php-fpm---fastCGI---翻译好的页面---nginx---用户

php-fpm 是进程管理器，可以管理多个fastCGI进程

fastCGI 公共网关接口，可以关联web服务（比如nginx）与动态网站解释器（比如php），每次nginx需要翻译动态页面了，就找fastCGI，一个服务器可以开启多个fastCGI进程，提高翻译效率

vim /etc/php-fpm.d/www.conf 此处配置为了解内容，目前无需修改

listen = 127.0.0.1:9000 第12行可以看到监听端口等信息

pm.max\_children = 50 第70行可以定义最多开启的fastCGI进程数量

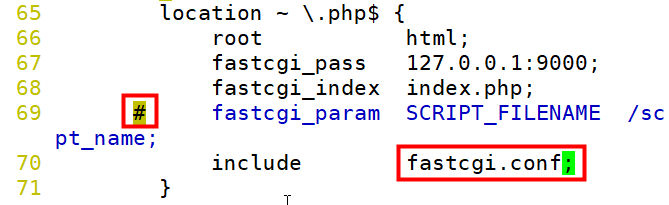
pm.start\_servers = 4 第75行可以定义无论是否需要，直接开启的fastCGI进程数量

cd /root/lnmp\_soft/php\_scripts

cp test.php /usr/local/nginx/html/ 拷贝动态测试页面到nginx网页目录下

修改nginx主配置文件，按下图

如果访问的是以.php结尾的页面，就转交给后台的9000端口(php-fpm)进行动态页面解析



/usr/local/nginx/sbin/nginx -s reload 再重置服务

然后使用浏览器访问192.168.2.5/test.php 可以看到动态网站的最终内容

4，测试

cd /root/lnmp\_soft/php\_scripts

cp mysql.php /usr/local/nginx/html/ 拷贝与数据库有关的动态测试页面到nginx

访问192.168.2.5/mysql.php

mysql 进入数据库

create user dc@localhost identified by '123'; 创建一个账户

quit

再次访问192.168.2.5/mysql.php 测试可以看到数据库新账户信息

排错：防火墙、php-pfm服务(9000端口)、nginx主配置文件、数据库服务是否开启，账户认证是否正确

实验二：地址重写

访问 [www.abcd1234xyz.com](http://www.abcd1234xyz.com) 时

看到的是 [www.abc.com](http://www.abc.com)

或者

访问 a.html 页面

看到的是 b.html

地址重写语法结构

rewrite regex replacement flag

rewrite 旧地址 新地址 [选项]

注意：此句必须写在虚拟机里面。

1，基本功能测试

cd /usr/local/nginx/conf

cp nginx.conf.default nginx.conf 还原nginx配置文件为默认

vim /nginx.conf 修改配置文件

rewrite /a.html /b.html; 添加地址重写，让看a页面时显示b页面的内容

cd ..

echo proxy\_a > html/a.html 创建a页面

echo proxy\_b > html/b.html 创建b页面

sbin/nginx -s reload 重置服务

curl 192.168.2.5/a.html 测试效果

2，重定向测试

再修改主配置文件，添加redirect选项

vim conf/nginx.conf

rewrite ^/a.html$ /b.html redirect;

sbin/nginx -s reload 重置服务

使用火狐浏览器测试192.168.2.5/a.html

3，测试不同网站间的跳转

再次修改nginx主配置文件

rewrite ^/ <http://www.tmooc.cn>; 无论访问目前网站的什么内容，都会跳到新网站

sbin/nginx -s reload 重置服务

使用火狐访问192.168.2.5/a.html可以跳转到<http://www.tmooc.cn>

4，测试不同网站的跳转，并且可以保留访问的页面

rewrite ^/(.\*) <http://www.tmooc.cn/$1>;

其中小括号表示保留（复制），后面的$1表示粘贴

sbin/nginx -s reload 重置服务

使用火狐访问192.168.2.5/a.html可以跳转到<http://www.tmooc.cn/a.html>

5，测试用户不同环境得到不同页面

页面准备2套

1套给火狐浏览器，另1套给curl

[www.taobao.com](http://www.taobao.com)

$http\_user\_agent 是nginx的内置变量，里面包含了用户的浏览器等环境信息

首先修改主配置文件，可以将之前的rewrite语句删除，再原位置添加以下内容,~是可以使用正则表达式来匹配被搜索内容，\*是忽略大小写

if ($http\_user\_agent ~\* firefox ) { 如果用户浏览器是火狐

rewrite ^(.\*)$ /firefox/$1; 就跳转到火狐专用页面

}

sbin/nginx -s reload 重置服务

mkdir html/firefox 创建火狐专用页面的目录

echo firefox\_web > html/firefox/a.html 创建火狐专用页面

最后测试使用 curl 192.168.2.5/a.html 与火狐访问该页面会得到不同结果

实验三：地址重写的选项

1，last选项

打开nginx主配置文件，将之前的地址重写语句删除，在原位置替换为以下内容，last可以实现不再读取后面rewrite语句的效果

rewrite /a.html /b.html last; 不加last，用户访问a页面看到的是c页面，加了的话只能看到b页面

rewrite /b.html /c.html;

sbin/nginx -s reload 重置nginx服务

curl 192.168.2.5/a.html 访问a页面看到的是c页面，因为rewrite语句可以持续匹配并重写

2，break选项

可以再次修改配置文件，将rewrite分别写到2个不同的location语句中（location语句可以实现如果用户访问的是什么页面，就执行后面大括号里面的任务），可以发现last语句失效，但如果改为break则可以在执行第一个rewrite语句后不再读取后面的语句。

location / { 这里是默认的location

rewrite /a.html /b.html break; 这里的last改成break

root html;

index index.html index.htm;

}

location /b.html { 这里创建的新location，如果用户访问的是b页面，就去执行大括号中任务

rewrite /b.html /c.html;

}