6_Operation02LNMPnginx+fastcgiNGINX 高级技术

一 案例：部署 LNMP 环境(动态网页环境)

静态网页:pdf,doc,mp3,mp4,txt,html...

动态网页:shell,php,java,python,perl,ruby...

静态网页处理流程:用户->nginx[/usr/.../html/静态文档]

动态网页处理流程:用户->nginx[/usr/.../html/编程语言编写的执行文档]

LNMP:主流的企业网站平台,Linux+Nginx+MySQL\MariaDB+PHP\Perl\Python

1.1 问题

安装部署 Nginx、MariaDB、PHP 环境

安装部署 Nginx、MariaDB、PHP、PHP-FPM；

启动 Nginx、MariaDB、FPM 服务；

并测试 LNMP 是否工作正常。

1.2 方案

LNMP（Linux、Nginx、MySQL、PHP）

在系统中，源码安装 Nginx，使用 RPM 包安装 MariaDB、PHP、PHP-FPM 软件。

操作过程中需要安装的软件列表如下：

nginx

mariadb、mariadb-server、mariadb-devel　　#客户端,服务端,依赖包

php、php-fpm、php-mysql #解释器,服务,扩展包(用于和 mysql 连接)

备注：mariadb（数据库客户端软件）、mariadb-server（数据库服务器软件）、

mariadb-devel（其他客户端软件的依赖包）、php（解释器）、php-fpm（进程管理器服务）、php-mysql（PHP 的数据库扩展包）。

1.3 步骤

1.3.1 安装软件(proxy 主机)

1）使用 yum 安装基础依赖包

```
~]# yum -y install gcc openssl-devel pcre-devel
```

2）源码安装 Nginx（如果前面课程中已经安装 Nginx，则忽略这一步）

```
~]# useradd -s /sbin/nologin  nginx
~]# tar -xvf nginx-1.12.2.tar.gz
~]# cd nginx-1.12.2
nginx-1.12.2]# ./configure   \
> --user=nginx   --group=nginx \
> --with-http_ssl_module
~]# make && make install
```

3）安装 MariaDB

Mariadb 在新版 RHEL7 光盘中包含有该软件,配置 yum 源后可以直接使用 yum 安装：

```
~]# yum -y install mariadb mariadb-server mariadb-devel
```

4）php 和 php-fpm

```
~]# yum -y  install  php   php-mysql
~]# yum -y  install php-fpm
```

1.3.2 启动服务

1）启动 Nginx 服务:80 端口（如果已经启动 nginx，则可以忽略这一步）

这里需要注意的是，如果服务器上已经启动了其他监听 80 端口的服务软件（如 httpd），则需要先关闭该服务，否则会出现冲突。

```
~]# systemctl stop httpd  #如果该服务存在则关闭该服务
~]# /usr/local/nginx/sbin/nginx  #启动 Nginx 服务
~]# netstat -utnlp | grep :80    #端口检查
```

2）启动 MySQL 服务:3306 端口

```
~]# systemctl start  mariadb     #启动服务器
~]# systemctl status mariadb     #查看服务状态
~]# systemctl enable mariadb     #设置开机启动
~]# netstat -utnlp | grep :3306 #端口检查
```

3）启动 PHP-FPM 服务:9000 端口

```
~]# systemctl start php-fpm      #启动服务
~]# systemctl status php-fpm     #查看服务状态
~]# systemctl enable php-fpm     #设置开机启动
~]# netstat -utnlp | grep :9000 #端口检查
```

1.3.3 client 主机测试 firefox 192.168.4.5,失败,L N M P 之间无关联,需要修改配置文件进行关联,实现对 PHP 页面的支持.

二 案例 2：构建 LNMP 平台

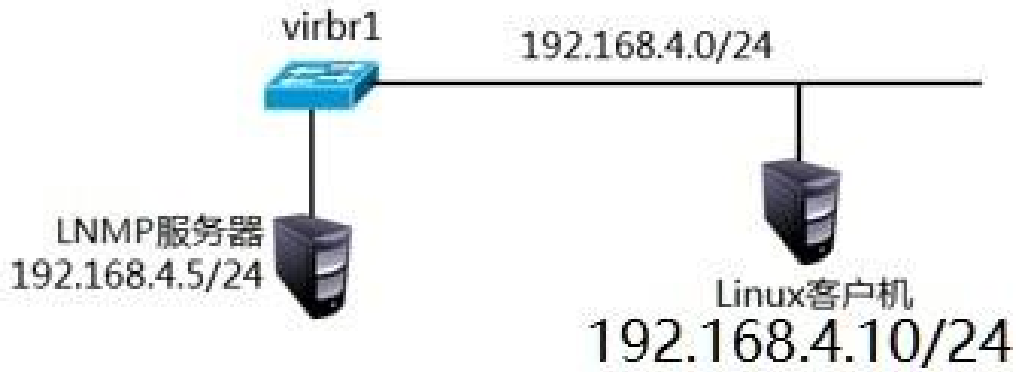
2.1 问题

沿用练习一，通过调整 Nginx 服务端配置，实现以下目标(对动态页面的支持)：

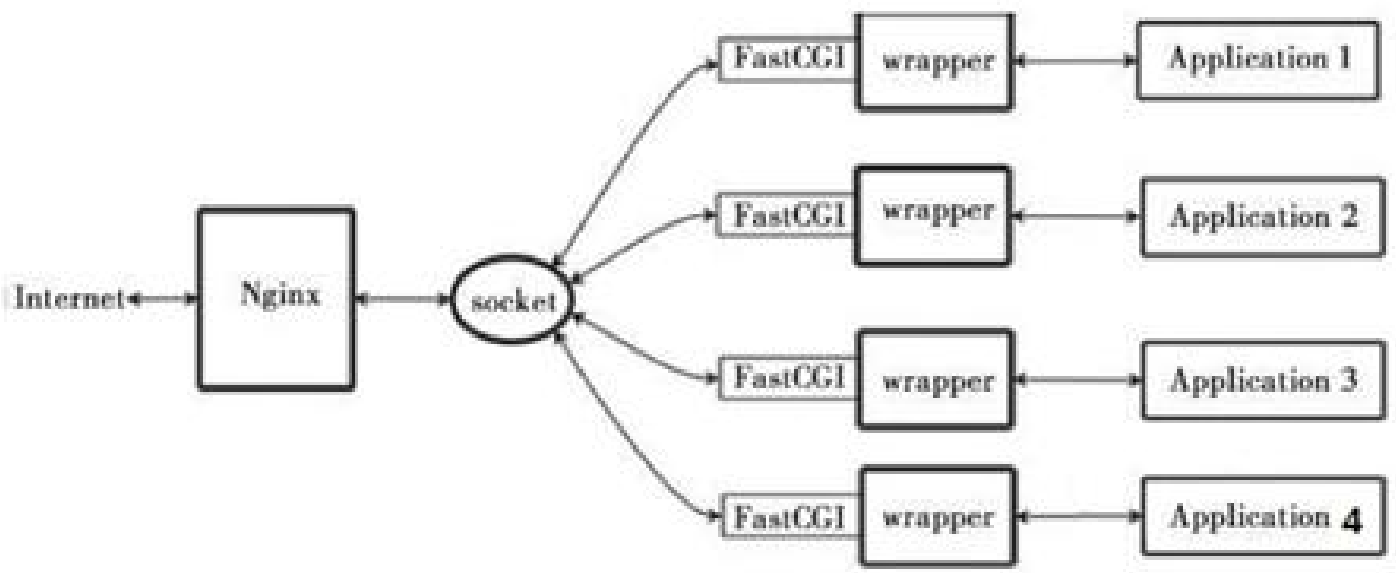
配置 Fast-CGI 支持 PHP 网页\创建 PHP 测试页面，测试使用 PHP 连接数据库的效果

2.2 方案

使用 2 台 RHEL7 虚拟机，其中一台作为 LNMP 服务器（192.168.4.5）、另外一台作为测试用的 Linux 客户机（192.168.4.10），如图所示。



Nginx 结合 FastCGI 技术即可支持 PHP 页面架构，如图所示。



FastCGI工作原理（续1）

Tedu.cn 达内教育

知识讲解

- 工作流程
  - 1. Web Server启动时载入FastCGI进程管理器
  - 2. FastCGI进程管理器初始化，启动多个解释器进程
  - 3. 当客户端请求到达Web Server时，FastCGI进程管理器选择并连接到一个解释器
  - 4. FastCGI子进程完成处理后返回结果，将标准输出和错误信息从同一连接返回Web Server

FastCGI缺点

Tedu.cn 达内教育

知识讲解

- 内存消耗大
  - 因为是多进程，所以比CGI多线程消耗更多的服务器内存，PHP-CGI解释器每进程消耗7至25兆内存，将这个数字乘以50或100就是很大的内存数
  - Nginx+PHP(FastCGI)服务器在3万并发连接下
    开10个Nginx进程消耗150M内存（10*15M）
    开64个php-cgi进程消耗1280M内存（20M*64）

因此本案例，需要延续练习一的实验内容，通过修改 Nginx 及 php-fpm 配置文件实现对 PHP 页面的支持。

FastCGI 支持 PHP,C/C++,Java,Perl,Python,Ruby 等

注意，FastCGI 的内存消耗问题，一个 PHP-FPM 解释器将消耗约 25M 的内存。

2.3 步骤

2.3.1 php-fpm 配置文件

1）查看 php-fpm 配置文件（实验中不需要修改该文件）

```
etc]# vim /etc/php-fpm.d/www.conf
[www]
listen = 127.0.0.1:9000          #PHP 端口号
pm.max_children = 32             #最大进程数量
pm.start_servers = 15            #最小进程数量
pm.min_spare_servers = 5         #最少需要几个空闲着的进程
pm.max_spare_servers = 32        #最多允许几个进程处于空闲状态
```

2.3.2 修改 Nginx 配置文件并启动服务

nginx

如果用户访问静态,则直接返回

如果用户访问动态,则转发给端口 9000,执行脚本后返回

location 可以匹配用户的地址栏,从/(网页根目录)开始

location / 能匹配一切,最后匹配;location 可理解为”匹配或 if”

详细流程:

用户地址栏 http://192.168.4.5/a.jpg->找文件->返回

用户地址栏 http://192.168.4.5/b.php->匹配->找文件->转发->执行->返回

~]# **vim /usr/local/nginx/conf/nginx.conf**

```
location / {
            root  html;  #网页根目录,表示网页在目录 html 下
            index index.php index.html index.htm;
#设置默认首页，当用户在浏览器地址栏中只写域名或 IP，不说访问什么页面时，服务器会把三个页面从左到右按顺序返回给用户
        }
 location ~ \.php$  { #~号代表正则匹配包含,匹配以.php 结尾的,\表示转义
            root           html;  #声明.php 文件在网页根目录下
            fastcgi_pass   127.0.0.1:9000;
            #将请求转发给本机 9000 端口，调用 PHP 解释器
            fastcgi_index  index.php;
            #fastcgi_param..._script_name;  #此行注释掉
            include        fastcgi.conf;        #加载其他配置文件
        }
```

~]# /usr/local/nginx/sbin/nginx -s reload #nginx 启动状态下重载

2.3.3 创建 PHP 页面，测试 LNMP 架构能否解析 PHP 页面

1）创建 PHP 测试页面 1，可以参考 lnmp_soft/php_scripts/test.php：

```
~]# vim /usr/local/nginx/html/test.php
<?php
$i="This is a test Page";
echo $i;
?>
```

2）创建 PHP 测试页面,连接并查询 MariaDB 数据库。

可以参考 lnmp_soft/php_scripts/mysql.php:

```
~]# vim /usr/local/nginx/html/mysql.php
<?php
$mysqli = new mysqli('localhost','root','密码','mysql');
#注意：root 为 mysql 数据库的账户名称，密码需要修改为实际 mysql 密码，无密码则留空即可
#localhost 是数据库的域名或 IP，mysql 是数据库的名称
if (mysqli_connect_errno()){
    die('Unable to connect!'). mysqli_connect_error();
}     #底色表示 1 段代码,下同
$sql = "select * from user";
$result = $mysqli->query($sql);
while($row = $result->fetch_array()){
```

```
    printf("Host:%s",$row[0]);
    printf("</br>");
    printf("Name:%s",$row[1]);
    printf("</br>");
}
?>
```

3）客户端使用浏览器访问服务器 PHP 首页文档，检验是否成功：

```
[root@client ~]# firefox http://192.168.4.5/test.php
[root@client ~]# firefox http://192.168.4.5/mysql.php
```

4）LNMP 常见问题日志

Nginx 的默认访问日志文件为/usr/local/nginx/logs/access.log

Nginx 的默认错误日志文件为/usr/local/nginx/logs/error.log

PHP 的默认错误日志文件为/var/log/php-fpm/www-error.log

如果动态网站访问失败，可用参考错误日志，查找错误信息。

三 案例 3：地址重写

3.1 问题

沿用练习二，通过调整 Nginx 服务端配置，实现以下目标：

所有访问 a.html 的请求，重定向到 b.html;

所有访问 192.168.4.5 的请求重定向至 www.tmooc.cn；

所有访问 192.168.4.5/下面子页面，重定向至 www.tmooc.cn/下相同的页面；

实现 firefox 与 curl 访问相同页面文件，返回不同的内容。

3.2 方案

关于 Nginx 服务器的地址重写，主要用到的配置参数是 rewrite：

```
rewrite regex replacement flag
```

rewrite 旧地址 新地址 [选项] # /表示网页根目录,旧地址支持正则表达式

3.3 步骤

3.3.1 修改配置文件(访问 a.html 重定向到 b.html)

1）修改 Nginx 配置文件：

```
~]# vim /usr/local/nginx/conf/nginx.conf
server {
      listen       80;
      server_name  localhost;      #测试时要注意添加地址重写时的域名
      rewrite /a.html  /b.html;    #在 server_name 下添加
location / {
      root   html;
      index  index.html index.htm;
      }
}
~]# echo "BB" > /usr/local/nginx/html/b.html
```

2）重新加载配置文件

```
~]# /usr/local/nginx/sbin/nginx  -s  reload
```

3）客户端测试

```
[root@client ~]# firefox  http://192.168.4.5/a.html
```

3.3.2 访问 a.html 重定向到 b.html（跳转地址栏）

1）修改 Nginx 服务配置：

```
~]# vim /usr/local/nginx/conf/nginx.conf
server {
      listen       80;
      server_name  localhost;
      rewrite /a.html  /b.html  redirect;
      #选项 redirect 的功能是在浏览器地址栏显示重写后的地址
location / {
      root   html;
      index  index.html index.htm;
      }
}
```

2）重新加载配置文件

```
~]# /usr/local/nginx/sbin/nginx  -s  reload
```

3）客户端测试（仔细观察浏览器地址栏的变化）

```
[root@client ~]# firefox  http://192.168.4.5/a.html
```

3.3.3 修改配置文件(访问 192.168.4.5 的请求重定向至 www.tmooc.cn)

1) 修改 Nginx 配置文件

```
~]# vim /usr/local/nginx/conf/nginx.conf
server {
      listen       80;
      server_name  localhost;
      rewrite ^/  http://www.tmooc.cn/;
      # ^/,正则表达式,将网页根目录下的所有页面全部重写为 www.tmooc.cn
      # rewrite /a.html  /b.html  redirect;  #此行可注释掉
      location / {
      root   html;
      index  index.html index.htm;
      }
}
```

2）重新加载配置文件

```
~]# /usr/local/nginx/sbin/nginx  -s  reload
```

#请先确保 nginx 是启动状态，否则运行该命令会报错,报错信息如下：

3）客户端测试（真实机测试，真实机才可以连接 tmooc）

```
[root@room9pc01 ~]# firefox  http://192.168.4.5
```

3.3.4 修改配置文件(访问 192.168.4.5/下面子页面，重定向至 www.tmooc.cn/

下相同的页面)

1) 修改 Nginx 配置文件

```
~]# vim /usr/local/nginx/conf/nginx.conf
server {
      listen      80;
      server_name  localhost;
      rewrite ^/(.*)$  http://www.tmooc.cn/$1;
      # $1 表示粘贴前面的(.*)
location / {
      root   html;
      index  index.html index.htm;
      }
}
```

2）重新加载配置文件

```
~]# /usr/local/nginx/sbin/nginx  -s  reload
```

3）客户端测试（真实机测试，真实机才可以连接 tmooc）

```
[root@room9pc01 ~]# firefox  http://192.168.4.5
[root@room9pc01 ~]# firefox  http://192.168.4.5/test
```

3.3.5 修改配置文件(实现 curl 和火狐访问相同链接返回的页面不同),类似手机和电脑访问同一个网站同一个页面,显示的页面效果不一样

1) 创建网页目录以及对应的页面文件：

```
~]# echo "I am Normal page" > /usr/local/nginx/html/test.html
~]# mkdir  -p  /usr/local/nginx/html/firefox/
~]# echo "firefox page" > /usr/local/nginx/html/firefox/test.html
```

2) 修改 Nginx 服务配置

```
~]# vim /usr/local/nginx/conf/nginx.conf
server {
      listen       80;
      server_name  localhost;
location / {
      root   html;
      index  index.html index.htm;
      }
      #这里，~(包含)符号代表正则匹配，*符号代表不区分大小写
if ($http_user_agent ~* firefox) {  #识别客户端 firefox 浏览器
rewrite ^/(.*)$  /firefox/$1;
                                    }
}
```

3）重新加载配置文件

```
~]# /usr/local/nginx/sbin/nginx  -s  reload
```

4）客户端测试

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

[root@client ~]# curl -u tom:123456 http://192.168.4.5/test.html
```

5）地址重写格式【总结】

rewrite 旧地址 新地址 [选项];旧地址支持正则表达式

选项

last	不再读其他 rewrite
break	不再读其他语句，结束请求
redirect	临时重定向
permament	永久重定向