Nginx服务

Day01

## **一 案例1:搭建Nginx服务器**

安装nginx-1.10.3版本时,需要使用如下参数

--with-http\_ssl\_module:提供SSL加密功能

--user:指定用户

--group:指定组

1.1步骤

步骤一:构建Nginx软件包

[1]使用源码包安装Nginx安装软件包

]#yum -y install gcc pcre-devel openssl-devel //安装依赖包

]#useradd -s /sbin/nologin nginx //防止nginx进入系统

]#tar -xf nginx-1.10.3.tar.gz

]#cd nginx-1.10.3 //进入nginx-1.10.3目录

nginx-1.10.3]# ./configure \

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

>--user=nginx //指定用户

>--group=nginx //指定组

>--with-http\_ssl\_module //开启SSL加密功能

]#make && make install //编译安装

[2]nignx命令用法

]#/usr/local/nginx/sbin/nginx //启动服务

]#/usr/local/nginx/sbin/nginx -s stop //关闭服务

]#/usr/local/nginx/sbin/nginx -s reload //重新加载配合文件

]#/usr/local/nginx/sbin/nginx -V //查看软件版本信息

]#ln -s /usr/local/nginx/sbin/nginx /sbin //创建软链接方便以后使用

netstat命令可以查看系统启动的端口信息,该命令的选项如下:

-a:显示所有端口信息

-n:以数字格式显示端口号

-t:显示TCP连接的端口

-u:显示UDP连接端口

-l:显示服务正在坚挺的端口信息

-p:显示监听端口服务的名称

]#netstat -anptu | grep nginx //nginx服务默认通过TCP 80端口监听客户端请求

步骤二:升级nginx

[1]编译新版本nginx软件

]#tar -zxvf nginx-1.12.2.tar.gz

]#cd nginx-1.12.0.tar.gz

]#./configure \

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

>--user=nginx

>--group=nginx

>--with-http\_ssl\_moduel

]#make

[2]备份老版本的nginx主程序,并使用编译好的新版本nginx替换老版本

]#mv /usr/local/nginx/sbin/nginx \

>/usr/local/nginx/sbin/nginxold

]#cp objs/nginx /usr/local/nginx/sbin/ //拷贝新版本

]#make upgrade #或者使用killall ngix杀死进程后启动nginx

步骤三:客户端范文测试

[1]分别使用firefox浏览器和curl测试页面

]#firefox <http://192.168.4.5>

]#curl <http://192.168.4.5>

**二 案例2:用户认证**

2.1问题

沿用案例1.通过调整nginx服务端配置,实现以下目标

[1]访问web页面需要进行用户认证

[2]用户名为:tom,密码为:123456

2.2 方案

通过Nginx实现web页面的认证,需要修改Nginx配置文件,在配置文件中添加auth语句实现用户认证.最后使用htpasswd命令创建用户及密码即可

2.3步骤

实现此案例需要按照如下步骤进行

步骤一:修改Nginx配置文件

[1]修改/usr/local/nginx/conf/nginx.conf

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

server {

listen 80;

server\_name localhost;

auth\_basic “input pass”;

auth\_basic\_user\_file “/usr/local/nginx/pass”;

location / {

root html;

index index.html index.htm;

}

}

[2]生成密码文件创建用户及密码

使用htpasswd命令创建账户文件,需要确保系统中已经安装了http-tools.

]#yum -y install httpd-tools

]#htpasswd -c /usr/local/nginx/pass tom //创建密码文件(-c只在第一使用)

New password: //输入密码

Re-type new password:

Adding passwd for user tom

]#htpasswd /usr/local/nginx/pass jerry //再次添加用户不适用 -c选项

New password:

Re-type new password:

Adding passwd for user jerry

]#cat /usr/local/nginx/pass

[3]重新加载配置文件

]#/usr/local/nginx/sbin/nginx -s relaod

**案例3:基于域名的虚拟主机**

3.1问题

沿用练习二,配置基于域名的虚拟主机,实现以下目标:

1. 实现两个基于域名的虚拟主机,域名分别为www.a.com和www.b.com
2. 对域名为www.a.com的站点进行用户认证,用户名称为tom,密码为123456

3.2方案

修改Nginx配置文件,添加Server容器实现虚拟主机功能,对于需要进行用户认证的虚拟主机添加auth认证语句.

虚拟主机一般分为:基于域名,基于IP和基于端口的虚拟主机

3.3步骤

步骤一:修改配置文件

[1]修改Nginx服务配置,添加相关虚拟主机配置如下:

server {

listen 80; //端口

server\_name [www.a.com;](http://www.a.com;) //域名

auth\_basic "input passwd"; //认证提示符

auth\_basic\_user\_file "/usr/local/nginx/pass"; //认证密码文件

location / {

root html; //指定网站根路径

index index.html index.htm;

}

}

server {

listen 80; //端口

server\_name [www.b.com;](http://www.b.com;) //域名

location / {

root www; //指定网站路径

index index.html index.htm;

}

}

[2]创建网站根目录以及对应首页文件

]#mkdir /us r/local/nginx/www

]#echo “WWW” >/usr/loacl/nginx/www/index.html

[3]重新加载配置

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

步骤二:客户端测试

[1]修改客户端主机192.168.4.10的/etc/hosts文件,进行域名解析.

192.168.4.5 [www.a.com](http://www.a.com) [www.b.com](http://www.b.com)

[2]登录192.168.4.10客户端主机记性测试

注意:请先关闭真实机的firefox,再SSH-X远程连接调用虚拟机的firefox

]#firefox <http://www.a.com>

]#firefox <http://www.b.com>

步骤三:扩展课外实验:其他类型的虚拟主机

[1]基于端口的虚拟主机

server{

listen 8080;

server\_name web1.example.com

}

server{

listen 8000;

server\_name web1.example.com

}

[2]基于IP的虚拟主机

server{

listen 192.168.0.1:80;

server\_name web1.example.com

}

server{

listen 192.168.0.2:80;

server\_name web1.example.com

}

**案例4:SSL虚拟主机**

4.1问题

沿用练习三,配置基于加密网站的虚拟机,实现以下目标:

1. 域名为www.c.com
2. 该站点通过hosts访问
3. 通过私钥,证书对该点所有数据加密

4.2方案

源码安装Nginx时必须使用--with\_ssl\_module参数,启用加密模块,对于需要进行SSL加密处理的站点添加ssl相关指令(设置网站需要的私钥和证书)

加密算法一般分为对称算法,非对称算法,信息摘要

对称算法有:AES,DES,主要应用在单机数据加密

非对称算法有:RSA,DSA,主要应用在网络数据加密

信息摘要:MD5,sha256,主要应用在数据完整性校验

4.3步骤

步骤一:配置SSL虚拟主机

[1]生成私钥证书

]#cd /usr/local/nginx/conf

]#openssl genrsa > cert.key

]#openssl req-new-x509-key cert.key > cert.pem

[2]修改Nginx配置文件,设置加密网站的虚拟主机

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

server{

listen 443ssl;

server\_name [www.c.com;](http://www.c.com;)

ssl\_certificate cert.pem; #这里是证书文件

ssl\_certificate\_key cert.key; #这里是私钥文件

ssl\_session\_cache shared:SSL:1m;

ssl\_session\_timeout 5m;

ssl\_ciphers HIGH:!aNULL:!MD5;

ssl\_prefer\_server\_ciphers on;

location / {

root html;

index index.html index.htm;

}

}

[3]重新加载配置

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

步骤二:客户验证

[1]修改客户端主机192.168.4.10的/etc/hosts文件,进行域名解析

]#vim /etc/hosts

192.168.4.5 [www.c.com](http://www.c.com) [www.a.com](http://www.a.com) [www.b.com](http://www.b.com)

[2]登录1922..168.4.10客户端主机进行测试

]#firefox <https://www.c.com>

Day02

**案例1:部署LNMP环境**

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,MariaDBm,FRM

\*并测试LNMP是否工作正常

1.2方案

LNMP(Linux,Nginx,Mysql.PHP)

在RHEL7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

php、php-fpm、php-mysql

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

实现此案例需要按照以下步骤进行.

步骤一:安装软件

[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

]#useradd -s /bin/nologin nginx

]#tar -xf nginx-1.12.2.tar.gz

]#cd nginx-1.12.2

]#./configure \

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

>--user=nginx --group=nginx \

>--with-http\_sssl\_module

]#make && make install

[3]安装MariaDB

]#yun -y install mariadb mariadb-server mariadb-devel

[4]php和php-fpm

]#yum -y install php php-mysql

]#yum -y install php php-mysql

]#yum -y install php-frm

步骤二:启动服务

[1]启动Nginx服务（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。

]#systamctl stop httpd //如果该服务存在则关闭该服务

]#/usr/local/nginx/sbin/nginx //启动nginx

]#netatat atunp | grep :80

[2]启动Mysql服务

]#systemctl start mariadb //启动服务

]#systemctl status mariadb //查看服务状态

]#systemctl enable mariadb //设置开机自启

[3]启动PHP-FRM服务

]#systemctl start php-fpm

]#sysemctl status php-fpm

]#systemc enable php-fpm

**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:

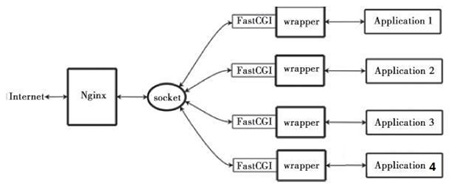
1. 配置Fast-CGI支持PHP页面

2.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,其中一台为LNMP服务器(192.168.4.5),另一台作为测试用的Linux客户机(192.168.4.10)

Nginx结合FastCGI技术即可支持PHP页面架构,如图:



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。

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

步骤一:php-fpm配置文件

[1]查看php-fpm配置文件

]#vim /etc/php-fpm.d/www.conf

[www]

listen=1270.0.01:9000 //PHP端口号

pm.max\_shildren=32 //最大进程数

pm.start\_server=15 //最小进程数

pm.min\_space\_server=5 //最少需要几个空闲的进程

pm.max\_space\_server=32 //最多允许几个进程处于空闲状态

步骤二:修改Nginx配置文件并启动服务

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

location/{

root html;

index index.php index.html index.htm

#设置默认首页为index.html,当用户在浏览器中只写域名或IP,不说访问哪个页面时,服务器会默认将首页返回给用户

}

location ~ \.php${

root html;

fastcgi\_pass 127.0.0.1:9000; #将请求转发给本机9000端口，PHP解释器

fastcgi\_index index.php;

#fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi.conf; #加载其他配置文件

}

]# /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：

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

步骤三: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 tast 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();

}

$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:

]#firefox <http://192.168.4.5/test.php>

]#firefox <http://192.168.4.5/mysql.php>

在mysql创建用户grant all on \*.\* to ‘用户’@’主机’identified by ‘密码’;

[4]LNMP常见问题

Nginx的默认访问日志为:/usr/local/nginx/logs/access.log

Nginx的默认错误日志为:/usr/local/nginx/logs/error.log

PHP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,可以参考错误日志,查看错误信息.

**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:

[1]所有访问a.html的请求,重定向到b.html;

[2]所有访问192.168.4.5/下面子页面,重定向至www.tmooc.cn;

[3]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

[4]实现firefox与curl访问相同的页面文件,返回不同内容.

3.2 方案

关于Nginx服务器的重写,主要用到的配置参数是rewriter

\*rewrite regex replacement flag

\*rewrite 旧地址 新地址[选项]

3.3步骤

步骤一:修改配置文件(访问a.html重定向到b.html)

[1]修改Nginx配置文件:

]#/var/local/nginx/conf/nginx.conf

server{

listen 80;

server\_name locslhost;

rewrite /a.html /b.html //重定向

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]客户测试

]#firefox http://192.168.4.5/a.html

步骤二:访问a.html重定向到b.html(跳转地址栏)

[1]修改Nginx服务:

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

server{

listen 80;

server\_name localhost;

rewrite /a.html /b.html;

location / {

root html;

index index.html index.htm

}

}

[2]重新加载配置文件

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

#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:

#[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory)

[3]客户端测试

]#firefox <http://192.168.4.5/a.html>

步骤三:修改配置文件(访问192.168..4.5的请求重定向www.tmooc.cn)

[1]修改Nginx服务配置

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

server{

listen 80;

server\_name localhost

rewrite ^/ <http://www.tmooc.cn/;>

location / {

root html;

index index.html index.htm;

#rewrite /a.html /b.html redirect;

}

}

[2]加载配置文件

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

[3]客户端测试(真机测试)

]#firefox <http://192.168.4.5>

步骤四:修改配置文件(访问192.168.4.5/下面的子页,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;

location / {

root html;

index index.html index.htm;

}

}

[2]重新加载文件

]#/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)

]#firefox <http://192.168.4.5>

]#firefox <http://192.168.4.5/test>

步骤五:修改配置文件(实现curl火狐访问相同链接返回的页面不同)

[1]创建网页目录以及对应的页面文件

]#echo “I am Nomal 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;

localhost / {

root html;

index index.html index.html;

}

#这里,~符号代表正则匹配,\*符号代表不区分大小写

if($http\_user\_agent ~\* firefix){ //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

]#firefox <http://192.168.4.5/test.html>

]#curl <http://192.168.4.5/test.html>

[5]地址重写格式

rewrite 旧地址 新地址 [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

Day03

**案例1:Nginx反向代理**

**1.1问题**

使用Nginx实现Web反向代理功能,实现如下功能:

\*后端Web服务两台,可以使用httpd实现

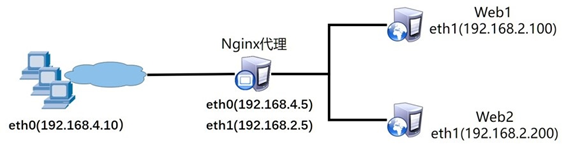
\*Nginx采用轮询的方式调用后端Web服务器

\*两台Web服务器的权重要求设置不同的值

\*最大失败次数为1,失败超时时间为30秒

**1.2 方案**

使用四台RHET7虚拟机,其中一台作为Nginx代理服务器,该服务器需要配置两块网卡IP地址分别为192.168.4.5和192.168.2.5，两台Web服务器IP地址分别为192.168.2.100和192.168.2.200。



**1.3 步骤**

实现此案例需要按照如下步骤进行.

**步骤一:部署实施后端Web服务器**

[1]部署后端Web1服务器

后端Web服务器可以简单使用yum方式安装httpd实现Web服务，为了可以看出后端服务器的不同，可以将两台后端服务器的首页文档内容设置为不同的内容。

]#yum -y install httpd

]#echo “192.168.2.100” > /var/www/html/index.html

]#systemctl restart httpd

]#firewall-cmd --set-default-zone=trusted

]#setentorce 0

[2]部署后端Web2服务器

]#yum -y install httpd

]#echo “192.168.2.200” > /var/www/html/index.html

]#systemctl restart httpd

]#firewall-cmd --set-default-zone=trusted

]#setentorce 0

**步骤二:配置Nginx服务器,添加服务池,实现反向代理功能**

[1]修改/usr/local/nginx/conf/nginx.conf配置文件

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

http{

#使用upstream定义后端服务器集群,集群名称任意(如:webserver)

#使用server定义集群中的具体服务器端口

upstream webserver{

server 192.168.2.100:80;

server 192.168.2.200:80;

}

server{

listen 80;

server\_name localhost;

location / {

#通过proxy-pass将用户的请求妆发给webserver集群

proxy\_pass http://webserver;

}

}

[2]重新加载配置

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

[3]客户端使用浏览器访问代理服务器测试轮询效果

]#curl <http://192.168.4.5> //使用该命令多次查看访问效果

]#curl <http://192.168.4.5> //使用该命令多次查看访问效果

**步骤三:配置upstream服务器集群池属性**

[1]设置失败次数,超时时间,权重

weight可以设置后台服务器的权重，max\_fails可以设置后台服务器的失败次数，fail\_timeout可以设置后台服务器的失败超时时间。

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

http{

upstream webserver{

server 192.168.2.100 weight=1 max\_fails=1 fail\_timeout=30;

server 192.168.2.200 weight=2 max\_fails=2 fail\_timeout=30;

server 192.168.2.101 down;

}

#weight设置服务器权重值，默认值为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

#down标记服务器已关机，不参与集群调度

server{

listen 80;

server\_name localhost;

location / {

#通过proxy-pass将用户的请求妆发给webserver集群

proxy\_pass http://webserver;

}

}

[2]重新加载配置

]#/usr/loca/nginx/sbin/nginx -s reload

[3]关闭一台后端服务器(如web1)

]#systemctl stop httpd

[4]客户端使用浏览器访问代理服务器测试轮询效果

]#curl <http://192.168.4.5>

[5]再次启动后端服务器的httpd(如web1)

]#systemctl start httpd

[6]客户端再次使用浏览器访问代理服务器测试轮询效果

]#curl <http://192.168.4.5>

**步骤三:配置uostream服务器集群的调度算法**

[1]设置相同客户端访问相同web服务

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

http{

upstream webserver{

#通过ip\_hash设置调度规则为:相同客户端访问相同服务器

ip\_hash;

server 192.168.2.100 weight=1 max\_fails=2 fail\_timeout=10;

server 192.168.2.200 weight=1 max\_fails=2 fail\_timeout=10;

}

server{

listen 80;

server\_name [www.tarena.com;](http://www.tarena.com;)

location / {

proxy\_pass http://webserver;

}

}

[2]重新加载配置

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

[3]客户端使用浏览器访问代理服务器测试轮询效果

]#curl <http://192.168.4.5>

**案例2:Nginx的TCP/UDP调度器**

**2.1问题**

使用Nginx实现TCP/UDP调度器功能,实现如下功能:

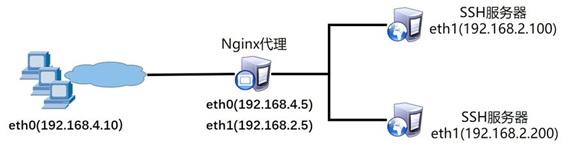
\*后端SSH服务器两台

\*Nginx编译安装时需要使用--with-stream,开启ngx\_stream\_core\_module模块

\*Nginx采用轮询的方式调用后端SSH服务器.

**2.2方案**

使用4台RHEL7虚拟机，其中一台作为Nginx代理服务器，该服务器需要配置两块网卡，IP地址分别为192.168.4.5和192.168.2.5，两台SSH服务器IP地址分别为192.168.2.100和192.168.2.200。客户端测试主机IP地址为192.168.4.10。如图-2所示。



**2.3步骤**

**步骤一：部署支持4层TCP/UDP代理的Nginx服务器**

[1]部署Nginx服务器

编译安装必要使用--with-stream参数开启四层代理模块

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

]#tar -xf nginx-1.12.2.tar.gz

]#cd nginx-1.12.2

]#./configure \

>--with-http\_ssl\_module //开启SSL加密功能

>--with-stream //开启四层反向代理功能

]#make && make install

**步骤二：配置Nginx服务器,添加服务器池,实现TCP/UDP;反向代理功能**

[1]修改/usr/local/nginx/conf/nginx.conf配置文件

]#vim /usr/local/ngnx/conf/nginx.conf

stream{

upstream bakend{

server 192.168.2.100:22; //后端SSH服务器的IP端口

server 192.168.2.200:22;

}

server{

listen 12345; //Nginx监听端口

proxy\_connect\_timeout 1s; //连接超时时间,可选配置

proxt\_timeout 3s;

proxy\_pass bakend;

}

}

http{

......

}

[2]重新加载配置

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

[3]客户端使用访问代理服务器测试轮询效果

]#ssh 192.168.4.5 -p 12345 //使用该命令多次访问查看效果

**案例3:Nginx常见问题处理**

**3.1问题**

本案例要求对Nginx服务器进行适当优化，解决如下问题，以提升服务器的处理性能：

\*如何自定义返回给客户端的404错误页面

\*如何查看服务器状态信息

\*如果客户端访问服务器提示“Too many open files”如何解决

\*如何解决客户端访问头部信息过长的问题

\*如何让客户端浏览器缓存数据

\*日志切割

\*开启gzip压缩功能，提高数据传输效率

\*开启文件缓存功能

\*然后客户机访问此Web服务器验证效果：

\*使用ab压力测试软件测试并发量

\*编写测试脚本生成长头部信息的访问请求

\*客户端访问不存在的页面，测试404错误页面是否重定向

**3.2步骤**

**步骤一:自定义报错页面**

[1]优化前,客户端使用浏览器访问不存在的页面,会提示404文件未找到

]#firefox://http192.168.4.5/xxxx //访问一个不存在的页面

[2]修改Nginx配置文件,自定义报错页面

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

....

charset utf-8; //仅需要中文时需要该选项,可选项

error\_page 404/404.html; //自定义错误页面

....

}#vim /usr/local/nginx/html/404.html //生成错误页面

]#nginx -s reload

#请先确保nginx服务处于启动状态,否则运行该命令会报错,

[3]优化后,客户端使用浏览器访问不存在的页面,会提示自己定义的404.htm页面

]#firefox <http://192.168.4.5/xxxx> //访问一个不存在的页面

[4]常见http状态码



**步骤二:如何查看服务器状态信息(非常重要的功能)**

[1]编译安装时使用--with-http\_stub\_status\_module开启状态页模块

]#tar -xf nginx-1.12.2.tar.gz

]#cd nginx-1.12.2

]#./configure \

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

>--with-stream \ //开启TCP/UDP代理模块

>--with-http\_stub\_status\_module //开启status状态页面

]#make && make install

[2]启动Nginx服务并查看监听端口状态

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。

]#/usr/local/nginx/sbin/

]#netstat -anptu | grep nginx

tcp 0 0 0.0.0:80 0.0.0.0:\* LIS TEN 10441/nginx

]#ss -anptu | grep nginx

[3]修改Nginx配置文件,定义状态页面

]#cat /usr/local/nginx/conf/nginx.conf

......

location / status{

stub\_status on;

allow IP

deny IP

}

.....

]#nginx

[4]优化后,查看状态页面信息

]#curl <http://192.168.4.5/status>

Active connections: 1

server accepts handled requests

10 10 3

Reading: 0 Writing: 1 Waiting: 0

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。

**步骤三;优化Nginx并发量**

[1]优化前使用ab高并发测试

]#ab -n 2000 -c 2000 http://192.168.4.5/

Benchmarking 192.168.4.5 (be patient)

socket: Too many open files (24)                //提示打开文件数量过多

[2]修改Nginx配置文件,增加并发量

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

....

worker\_process 2; //与cpu核心 数保持一至

events{

worker\_connection 65535; //每个worker最大并发连接数

}

....

]#nginx -s reload

[3]优化linux内核(最大文件数量)

[root@proxy ~]# ulimit -a                        //查看所有属性值

[root@proxy ~]# ulimit -Hn 100000                //设置硬限制（临时规则）

[root@proxy ~]# ulimit -Sn 100000                //设置软限制（临时规则）

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

    .. ..

\* soft nofile 100000

\* hard nofile 100000

#该配置文件分4列，分别如下：

#用户或组 硬限制或软限制 需要限制的项目 限制的值

[4]优化测试服务器并发量(因为客户端没调内核参数,所以在proxy测试)

]#ab -n 2000 /c 2000 http”//192..168.4.5

**步骤四：优化Nginx数据包头缓存**

[1]优化前,使用脚本测试长头部请求是否能获得响应

[root@proxy ~]# cat lnmp\_soft/buffer.sh

#!/bin/bash

URL=http://192.168.4.5/index.html?

for i in {1..5000}

do

    URL=${URL}v$i=$i

done

curl $URL                                //经过5000次循环后，生成一个长的URL地址栏

[root@proxy ~]# ./buffer.sh

.. ..

<center><h1>414 Request-URI Too Large</h1></center>        //提示头部信息过大

[2]修改Nginx配置文件,增加数据包头部缓存大小

]#vim /usr/loacl/nginx/conf/nginx.conf

.. ..

http {

client\_header\_buffer\_size 1k;        //默认请求包头信息的缓存

large\_client\_header\_buffers 4 4k;        //大请求包头部信息的缓存个数与容量

.. ..

}

[root@proxy ~]# nginx -s reload

[3]优化后,使用脚本测试长头部请求是否能获得响应

[root@proxy ~]#cat cat buffer.sh

#!/bin/bash

URL=http://192.168.4.5/index.html?

for i in {1..5000}

do

    URL=${URL}v$i=$i

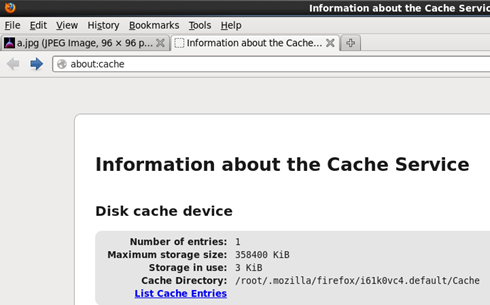
done

curl $URL

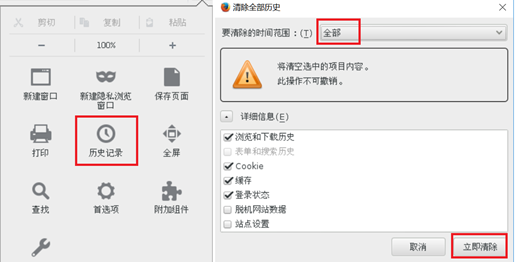
[root@proxy ~]# ./buffer.sh

**步骤五：浏览器本地缓存静态数据**

[1]使用firefox浏览器查看缓存

以Firefox浏览器为例，在Firefox地址栏内输入about:cache将显示Firefox浏览器的缓存信息，如图-3所示，点击List Cache Entries可以查看详细信息。

[2]清空firefox本地缓存数据,如图



[3]修改Nginx配置文件,定义对静态页面的缓存时间

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

server{

listen 80;

server\_name localhost;

location / {

root html;

index index.html index.htm;

}

location ~\* \.(jpg|jpej|png|css|js|ico|xml)${

expires 30d; //定义客户段缓存时间为30天

}

}

]#cp /usr/local/share/backgrounds/day.jpg /usr/local/nginx/html

]#nginx -s reload

[4]优化后,使用Firefox浏览器访问图片,再次查看缓存信息

]#firefox http:192.168.4.5/day.jpg

在firefox地址栏内输入about:cache，查看本地缓存数据，查看是否有图片以及过期时间是否正确。

**步骤六:日志切割**

日志我嫩见越来越大怎么办?单个文件10G?如何切割?(常见面试题)

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

备注:/usr/local/nginx/logs/nginx.pid文件中存放的是nginx的进程PID.

]#mv access.log ceesee2.log

]#kill -USR1 $(cat /usr/local/nginx.lods/nginx.pid)

[2]自动完成

每周五的03点0.3分自动执行脚本完成日志切割工作.

]#vim /usr/local/nginx/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)

]#crontab -e

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

**步骤七修改nginx配置文件**

]#cat /usr/local/nginx/conf/nginx.conf

http{

....

gzip on; //开启压缩

gzip\_min\_length 1000; //小文件不压缩

gzip\_comp\_level 4; //压缩比率

gzip\_type 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

....

}

步骤八:服务器内存缓存

[1]如果需要处理大容量静态文件,可以将文件缓存在内存,下次访问会更快

http{

open\_file\_cache max=200 inactive=20s;

open\_file\_cache\_valid 60s;

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

open\_file\_cache\_errors off;

//设置服务器组大缓存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秒后过期

//只有访问次数超过5次会被缓存

}

day04

**案例1:PHP的本地session信息**

**1.1问题**

通过Nginx调度器负责后端两台web服务器,实现以下目标

1. 部署Nginx为前台调度服务器
2. 调度算法设置为轮询
3. 后端为两台LNMP服务器
4. 部署测试页面,查看PHP本地的session信息

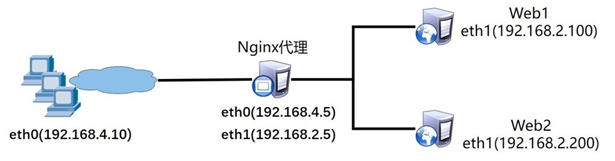
1.2方案

概念:

Session:存储在服务器端,存储用户名,登录状态等信息.

Cookie:y:由服务器下发给客户端,保存在客户端的一个文件里.

存储的内容主要包括:Session



**1.3步骤**

实现此案例需要按如下步骤进行.

步骤一:部署后端LNMP服务器相关软件

注意:以下部署LNMP服务器的操作,需要在两台后端服务器做相同的操作,下面我么一web2服务器(192.168.2.200)为例,对web1服务器进行相同的操作即可

[1]使用yum安装基础依赖

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

[2]源码安装Nginx

]#tar -xf nginx-1.12.2.tar.gz

]#cd nginx-1.12.2.tar.gz

]#./configure \

>--with-http\_ssl\_module

]#make && make install

[3]安装mariadb数据库

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

[4]安装PHP

]#yum -y install php php-mysql

]#yum -y install php-fpm

[4]修改Nginx配置文件

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

location \ {

root html;

index index.php index.html index.htm

}

location ~ /.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

#fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fast.conf

}

**步骤二:启动LNMP服务器相关服务**

[1]启动Nginx服务

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,苟泽会出现冲突

]#systemctl stop httpd

]#/usr/local/nginx/sbin/nginx

]#netstat -utnlp |grep :80

[2]启动MySQL服务

]#sustemctl start mariadb

]#systemctl status mariadb

[3]启动php-fpm服务

]#systemctl start php-fpm

]#systemctl status php-fpm

**步骤三:部署前段Nginx调度服务器**

[1]使用远吗安装nginx软件(如果nginx软件包已安装可以忽略此步骤)

[root@proxy ~]# yum -y install gcc pcre-devel openssl-devel

[root@proxy ~]# tar -xf nginx-1.12.2.tar.gz

[root@proxy ~]# cd nginx-1.12.2

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

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

[2]修改Nginx配置文件,实现代理服务器

Nginx配置文件中，通过upstream定义后端服务器地址池，默认调度策略为轮询，使用proxy\_pass调用upstream定义的服务器地址池：

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

.. ..

upstream webs {

server 192.168.2.100:80;

server 192.168.2.200:80;

}

server {

listen 80;

server\_name localhost;

location / {

             proxy\_pass http://webs;

root html;

index index.php index.html index.htm;

}

}

[3]重新加载配置文件

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

**步骤四:部署测试页**

[1] 部署测试页面(web1服务器)

测试页面可以参考lnmp\_s**oft/php\_scripts/php-**memcached-demo.tar.gz。

[root@web1 ~]# cd lnmp\_soft/php\_scripts/

[root@web1 php\_scripts]# tar -xf php-memcached-demo.tar.gz

[root@web1 php\_scripts]# cd php-memcached-demo

[root@web1 php-memcached-demo]# cp -r \* /usr/local/nginx/html/

[2]浏览器直接访问后端服务器的测试页面(web1服务)

[root@web1 ~]# firefox http://192.168.2.100            //填写账户信息

[root@web1 ~]# cd /var/lib/php/session/            //查看服务器本地的Session信息

[root@web1 ~]# ls

sess\_ahilcq9bguot0vqsjtd84k7244                        //注意这里的ID是随机的

[root@web1 ~]# cat sess\_ahilcq9bguot0vqsjtd84k7244

注意：可用修改index.php和home.php两个文件的内容，添加页面颜色属性，以区别后端两台不同的服务器:<body bgcolor=blue>。

[3]部署测试页面(web2)

测试页面可以参考lnmp\_soft/php\_scripts/php-memcached-demo.tar.gz。

[root@web2 ~]# cd lnmp\_soft/php\_scripts/

[root@web2 php\_scripts]# tar -xf php-memcached-demo.tar.gz

[root@web2 php\_scripts]# cd php-memcached-demo

[root@web2 php-memcached-demo]# cp -a \* /usr/local/nginx/html/

[4]浏览器直接访问后端服务器的测试页面

[root@web2 ~]# firefox http://192.168.2.100         //填写账户信息

[root@web2 ~]# cd /var/lib/php/session/            //查看服务器本地的Session信息

[root@web2 ~]# ls

sess\_qqek1tmel07br8f63d6v9ch401                        //注意这里的ID是随机的

[root@web2 ~]# cat sess\_qqek1tmel07br8f63d6v9ch401

注意：可用修改index.php和home.php两个文件的内容，添加页面颜色属性，以区别后端两台不同的服务器:<body bgcolor=green>。

[5]浏览器访问前段调度测试(不同后端服务器session不一致).

推荐使用goole浏览器

root@client ~]# google-chrome http://192.168.4.5

//填写注册信息后，刷新，还需要再次注册，说明两台计算机使用的是本地Session

//第二台主机并不知道你再第一台主机已经登录，第一台主机的登录信息也没有传递给第二台主机

案例2:构建memcached服务

2.1问题

本案例要求先快速搭建号一台memcached服务器,并对memcached进行简单的增删改查操作:

\*安装memcached软件,并启动服务

\*使用telnet测试memcache服务

\*对memcached进行增删改查等操作

**2.2方案**

使用1台RHEL7虚拟机包含有memcached服务器(192.168.4.5)

在RHEL7系统光盘中包含有memcached,因此需要提前配置yum源,即可直接使用yum安装,客户端测试时需要提前安装telnet远程工具.

验证时需要客户端主机安装telnet，远程memcached来验证服务器的功能：

add name 0 180 10 //变量不存在则添加

set name 0 180 10 //添加或替换变量

replace name 0 180 10 //替换

get name //读取变量

append name 0 180 10 //向变量中追加数据

delete name //删除变量

flush\_all //清空所有

提示：0表示不压缩，180为数据缓存时间，10为需要存储的数据字节数量。

**2.3步骤**

实现此案例要按照如下步骤进行.

**步骤一:构建memcached服务**

[1]使用yum安装软件包memcached

]#yum -y install memcached

]#rpm -qa memcached

[2]memcached配置文件(查看即可,不需要修改)

[root@proxy ~]# vim /usr/lib/systemd/system/memcached.service

ExecStart=/usr/bin/memcached -u $USER -p $PORT -m $CACHESIZE -c $MAXCONN $OPTIONS

[root@proxy ~]# vim /etc/sysconfig/memcached

PORT="11211"

USER="memcached"

MAXCONN="1024"

CACHESIZE="64"

OPTIONS=""

[3]启动服务并查看网络连接状态验证是否开启成功:

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，使用ss命令可以替代netstat，功能与选项一样。

[root@proxy ~]# systemctl start memcached

[root@proxy ~]# systemctl status memcached

[root@proxy ~]# netstat -anptu | grep memcached

tcp    0    0 0.0.0.0:11211        0.0.0.0:\*        LISTEN        2839/memcached

tcp    0    0 :::11211            :::\*                LISTEN        2839/memcached

udp    0    0 0.0.0.0:11211        0.0.0.0:\*                    2839/memcached

udp    0    0 :::11211            :::\*                            2839/memcached

[root@proxy ~]# setenforce 0

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

**步骤二:使用telnet访问,emecached服务**

[1使用yum安装telnet

]#yum -y install telnet

[2]使用telnet连接服务器测试memecached服务器功能,包括增删改查操作

[root@proxy ~]# telnet 192.168.4.5 11211

Trying 192.168.4.5...

……

##提示：0表示不压缩，180为数据缓存时间，3为需要存储的数据字节数量。

set name 0 180 3                //定义变量，变量名称为name

plj                            //输入变量的值，值为plj

STORED

get name                        //获取变量的值

VALUE name 0 3             //输出结果

plj

END

##提示：0表示不压缩，180为数据缓存时间，3为需要存储的数据字节数量。

add myname 0 180 10            //新建，myname不存在则添加，存在则报错

set myname 0 180 10            //添加或替换变量

replace myname 0 180 10        //替换，如果myname不存在则报错

get myname                    //读取变量

append myname 0 180 10        //向变量中追加数据

delete myname                    //删除变量

flush\_all                        //清空所有

quit                            //退出登录

案例3:LNMP+memcached

**3.1问题**

沿用练习一和练习二，部署LNMP+memcached网站平台,通过PHP页面实现对memcached服务器的数据操作，实现以下目标：

\*为PHP安装memcache扩展

\*创建PHP页面，并编写PHP代码，实现对memcached的数据操作

**3.2方案**

如果希望使用PHP来操作memcached，注意必须要为PHP安装memcache扩展（php-pecl-memcache），否则PHP无法解析连接memcached的指令。客户端测试时需要提前安装telnet远程工具

**3.3步骤**

**步骤一：创建PHP页面，使用PHP语言测试memcached服务**

[1]部署测试页面

创建PHP首页文档/usr/local/nginx/html/mem.php，测试页面可以参考lnmp\_soft/php\_scripts/mem.php。

注意：192.168.2.5是memcached数据库

[root@web1 ~]# vim /usr/local/nginx/html/mem.php

<?php

$memcache=new Memcache;                //创建memcache对象

$memcache->connect('192.168.2.5',11211) or die ('could not connect!!');

$memcache->set('key','test');          //定义变量

$get\_values=$memcache->get('key');     //获取变量值

echo $get\_values;

?>

[2]客户端测试

客户端使用浏览器访问服务器PHP首页文档，检验对memcached的操作是否成功：

]#firefox://192.168.2.200/mem.php

注意：这里因为没有给PHP安装扩展包，默认PHP无法连接memcached数据库，需要给PHP安装扩展模块才可以连接memcached数据库。

[3]为PHP添加memcache扩展

[root@web1 ~]# yum -y install php-pecl-memcache

[root@web1 ~]# systemctl restart php-fpm

[4]客户端再次测试

]#firefox <http://192.168.2.100/mem.php>

**案例4:PHP实现session共享**

**4.1问题**

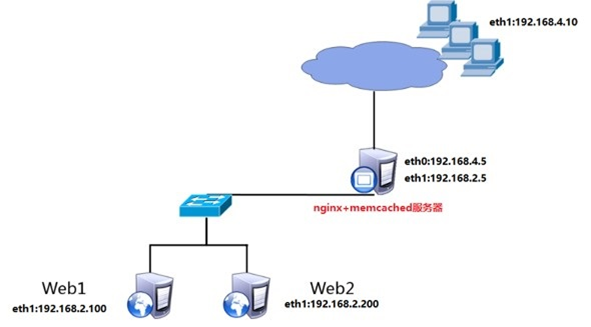
沿用练习三，通过修改PHP-FPM配置文件，实现session会话共享：

配置PHP使用memcached服务器共享Session信息

客户端访问两台不同的后端Web服务器时，Session 信息一致

**4.2 方案**

在练习三拓扑的基础上，Nginx服务器除了承担调度器外，还需要担任memcached数据库的角色，并在两台后端LNMP服务器上实现PHP的session会话共享。拓扑结构如图-4所示。



### **4.3 步骤**

实现此案例需要按照如下步骤进行。

步骤一：在后端LNMP服务器上部署Session共享

注意：这些操作在两台后端Web服务器上均需要执行，以下操作以Web1（192.168.2.100）服务器为例。

[1]为PHP添加memcache扩展

注意，因为后端两台web服务器(web1,web2)都需要连接memcached数据库，所以两台主机都需要安装PHP扩展模块(下面也web1为例)。

]#yum -y install php-pecl-memcache

[2]修改PHP-FPM配置文件,并重启服务

注意:因为后端两台web服务器(web1,web2)都需要修改配置文件(下面也web1为例)

[root@web1 ~]# vim /etc/php-fpm.d/www.conf            //修改该配置文件的两个参数

//文件的最后2行

修改前效果如下:

php\_value[session.save\_handler] = files

php\_value[session.save\_path] = /var/lib/php/session

//原始文件，默认定义Sessoin会话信息本地计算机（默认在/var/lib/php/session）

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

修改后效果如下:

php\_value[session.save\_handler] = memcache

php\_value[session.save\_path] = "tcp://192.168.2.5:11211"

//定义Session信息存储在公共的memcached服务器上，主机参数中为memcache（没有d）

//通过path参数定义公共的memcached服务器在哪（服务器的IP和端口）

[root@web1 ~]# systemctl restart php-fpm

步骤三:客户端测试

客户端使用浏览器访问两台不同的Web服务器。

操作步骤参考练习一，最终可以获得相关的Session ID信息