Nginx安装

源码包安装nginx软件包

1)

yum -y install gcc pcre-devel openssl-devel

\*gcc pcre-devel openssl-devel 3个依赖包

2)

\*\*useradd -s /sbin/nologin nginx

权限针对用户(rwx)

\*谁启动了程序若被攻击及拥有了启动该程序的用户权限

\*root执行命令 nginx启动便降级成nginx 降权 (类似suid 效果 提权)

\*默认安装会生成nobody用户

3)

tar XXX 解包

cd XXX 进入

./configure 运行脚本

\*默认配置 (也可以自己编辑相关配置)

> --prefix=/usr/local/nginx \ //指定安装路径

> --user=nginx \ //指定用户

> --group=nginx \ //指定组

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

> --with-stream //开启4层反向代理功能

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

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

make && make install 编译并安装

4)

nginx命令的用法

/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 //重新加载配置文件(服务器开启的情况下)

/usr/local/nginx/sbin/nginx -V //查看软件信息

ln -s /usr/local/nginx/sbin/nginx /sbin/ //方便后期使用

5)

netstat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：

查端口信息

netstat -ntulp |grep :80 nginx

netstat -ntulp |grep :3306 mariadb

netstat -ntulp |grep :11211 memcached

netstat -ntulp |grep :9000 php-fpm

netstat -ntulp |grep :8005 tomcat 关闭

netstat -ntulp |grep :8009 tomcat 高效端口

netstat -ntulp |grep :8080 tomcat 默认

6)

设置防火墙与SELinux

systemctl stop firewalld

setenforce 0

#######################################################################################################

升级Nginx服务器

1）编译新版本nginx软件

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

[root@proxy ~]# cd nginx-1.12.2

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

> --prefix=/usr/local/nginx \ //指定安装路径

> --user=nginx \ //指定用户

> --group=nginx \ //指定组

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

> --with-stream //开启4层反向代理功能

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

[root@proxy nginx-1.12.2]# make

\*\*注意只到此步骤. 若运行make install 则会直接覆盖 原版本之后

2) 备份老的nginx主程序，并使用编译好的新版本nginx替换老版本

[root@proxy nginx-1.12.2]# mv /usr/local/nginx/sbin/nginx \

>/usr/local/nginx/sbin/nginxold

[root@proxy nginx-1.12.2]# cp objs/nginx /usr/local/nginx/sbin/ //拷贝新版本

[root@proxy nginx-1.12.2]# make upgrade //升级

make upgrade 与 killall nginx 效果相同

\*\*\*但是 make upgrade 必须在解包生成目录下

[root@proxy nginx-1.12.2]# make upgrade

\*\*此步为平滑升级最后一步

########################################################################################

用户认证

1）修改/usr/local/nginx/conf/nginx.conf

server {

listen 80;

server\_name localhost;

auth\_basic "Input Password:"; //认证提示符信息

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

location / {

root html;

index index.html index.htm;

}

}

\*\* auth\_basic 认证提示符信息 auth\_basic\_user\_file 认证的密码文件

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httpd-tools

# yum -y install httpd-tools

# htpasswd -c /usr/local/nginx/pass tom //创建密码文件

3)添加密码认证用户

# htpasswd -c /usr/local/nginx/pass jerry //首次添加用户

# htpasswd /usr/local/nginx/pass harry //追加用户，不使用-c选项

4)重新加载配置

/usr/local/nginx/sbin/nginx -s reload //重新加载配置文件

#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)

###############################################################################################

基于域名的虚拟主机

server {

listen 80; //端口

server\_name www.a.com; //域名

auth\_basic "Input Password:"; //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; //域名 \*\*\*\*\*\*\*\*\*\*\*\*

location / {

root www; //指定网站根路径

index index.html index.htm;

}

}

#################################################################################

基于端口的虚拟主机

server {

listen 8080; //端口\*\*\*\*\*\*\*\*\*\*\*\*

server\_name web1.example.com; //域名

......

}

server {

listen 8000; //端口

server\_name web1.example.com; //域名

.......

}

####################################################################################

基于IP的虚拟主机

server {

listen 192.168.0.1:80; //IP地址与端口 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

server\_name web1.example.com; //域名

... ...

}

server {

listen 192.168.0.2:80; //IP地址与端口

server\_name web1.example.com;

... ...

}

#########################################################################################

基于加密网站的虚拟主机(SSL虚拟主机)

1)

生成私钥与证书

# cd /usr/local/nginx/conf

# openssl genrsa > cert.key //生成私钥

# openssl req -new -x509 -key cert.key > cert.pem //生成证书

2)

修改Nginx配置文件，设置加密网站的虚拟主机

打开注释即可

server {

listen 443 ssl;

server\_name 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;

}

}

#########################################################################################

部署LNMP环境

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。

php-fpm配置文件 不需要修改该文件 (需要了解各个参数意义)

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

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.php，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，服务器会把默认首页index.php返回给用户

}

#只需要打开注释即可 需要修改一处

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; \*\*此处注释

include fastcgi.conf; \*\*\*此处需要修改 #加载其他配置文件

}

#########################################################################################

地址重写

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];

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

修改配置文件(访问a.html重定向到b.html)

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; \*\*添加处 (地址栏不跳转 仍然显示a.html)

location / {

root html;

index index.html index.htm;

}

}

访问a.html重定向到b.html（跳转地址栏）

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

.. ..

server {

listen 80;

server\_name localhost;

rewrite /a.html /b.html redirect;

location / {

root html;

index index.html index.htm;

}

}

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

# 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;

}

}

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)

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

.. ..

server {

listen 80;

server\_name localhost;

rewrite ^/(.\*)$ http://www.tmooc.cn/$1; \*\*\* $1的内容与( )内的内容是一致的 与sed s替换中 () \1 作用一致

location / {

root html;

index index.html index.htm;

}

}

######################################################################################

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)

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

.. ..

server {

listen 80;

server\_name localhost;

location / {

root html;

index index.html index.htm;

}

#这里，~符号代表正则匹配，\*符号代表不区分大小写

#$http\_user\_agent 此参数为确定浏览器类型 配置文件中有 可以抄

if ($http\_user\_agent ~\* firefox) { //识别客户端firefox浏览器

rewrite ^(.\*)$ /firefox/$1;

}

}

########################################################################################

部署实施后端Web服务器

配置Nginx服务器，添加服务器池，实现反向代理功能

\*\*仅应用http 7层代理 默认安装nginx已经打开 upstream需要写在http服务内的上方

\*\*

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和端口

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将用户的请求转发给webserver集群

proxy\_pass http://webserver;

}

}

配置upstream服务器集群池属性

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

配置upstream服务器集群的调度算法

\*\* 默认调度为轮循 ip\_hash为唯一匹配

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=2 max\_fails=2 fail\_timeout=10;

}

#######################################################################################

部署支持4层TCP/UDP代理的Nginx服务器

部署nginx服务器

编译安装必须要使用--with-stream参数开启4层代理模块

\*\*4层代理 必须是 stream 并且写在http外面不为7层应用模块

stream {

upstream backend {

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

server 192.168.2.200:22;

}

server {

listen 12345; //Nginx监听的端口

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

proxy\_timeout 3s;

proxy\_pass backend;

}

}

http {

.. ..

}

#######################################################################################

如何查看服务器状态信息（非常重要的功能）

修改Nginx配置文件，定义状态页面

location /status {

stub\_status on;

#allow IP地址;

#deny IP地址;

}

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。

优化Nginx并发量

测试并发量的工具

ab -c 并发数 -n 总请求数 url

http\_loca -p 并发数 -s 测试时间 url

webbench -c 并发数 -t 测试时间 url

seige -c 并发数 -r 重复次数 url

修改Nginx配置文件，增加并发量

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

.. ..

worker\_processes 2; //与CPU核心数量一致

events {

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

}

优化Linux内核参数（最大文件数量）

# ulimit -a //查看所有属性值

# ulimit -Hn 100000 //设置硬限制（临时规则）

# ulimit -Sn 100000 //设置软限制（临时规则）

# vim /etc/security/limits.conf //修改配置文件(永久规则)

.. ..

\* 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

修改Nginx配置文件，增加数据包头部缓存大小

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

.. ..

}

修改Nginx配置文件，定义对静态页面的缓存时间

server {

listen 80;

server\_name localhost;

location / {

root html;

index index.html index.htm;

}

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

expires 30d; //定义客户端缓存时间为30天

}

}

日志切割

1. 把旧的日志重命名

2. kill USR1 PID(nginx的进程PID号)

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

# mv access.log access2.log //重命名 保留原文件

# kill -USR1 $(cat /usr/local/nginx/logs/nginx.pid) //通过pid切割

kill 操作并不是全是杀死 kill -USR1 = kill -10 这个选项

对页面进行压缩处理

[root@proxy ~]# 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 压缩比例越高 时间越慢

gzip\_types 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

.. ..

}

\*\*\* jpg|jpeg|gif|png|css|js|ico|xml 类型在 /usr/local/nginx/conf/mime.types 中可查看

服务器内存缓存

http {

open\_file\_cache max=2000 inactive=20s;

open\_file\_cache\_valid 60s;

open\_file\_cache\_min\_uses 5;

open\_file\_cache\_errors off;

//设置服务器最大缓存2000个文件句柄，关闭20秒内无请求的文件句柄 最多缓存2000个文件 若在缓存中文件20秒内未被访问则移出缓存区

//文件句柄的有效时间是60秒，60秒后过期 在缓存区中的文件60秒后会被清除

//只有访问次数超过5次会被缓存

}

#####################################################################################

PHP的本地Session信息

概念：

Session：存储在服务器端，保存用户名、登陆状态等信息。

Cookies：由服务器下发给客户端，保存在客户端的一个文件里。

保存的内容主要包括：SessionID。

/var/lib/php/session/ //查看服务器本地的Session信息

每访问一次网页生成一个session ID

#####################################################################################

构建memcached服务

memcached 是高性能的分布式缓存服务器

将信息存入内存中 与 mysql 不同 mysql 存入硬盘中

内存调用明显 快 与mysql

memcached 中常用命令 add set replace get append delete flush\_all

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。

1）使用yum安装软件包memcached

yum -y install 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 // /usr/lib/systemd/system 中存放所有使用systemctl控制的服务

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)使用telnet连接服务器测试memcached服务器功能，包括增、删、改、查等操作

# 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

#########################################################################################

LNMP+memcached

yum -y install php-pecl-memcache //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。

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 //没有d

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

//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）

#########################################################################3

JAVA SE 标准版

JAVA EE 企业版

JAVA ME 移动版

部署Tomcat服务器软件

1)

yum -y install java-1.8.0-openjdk //安装JDK

JDK是sun针对java开发者推出的java语言的软件开发包

jdk 是整个java的核心

--包括了java运行环境

--java工具(编译 排错 打包等工具)

--java基础的类库

2)

安装Tomcat（apache-tomcat-8.0.30.tar.gz软件包，在lnmp\_soft中有提供）

# tar -xf apache-tomcat-8.0.30.tar.gz

# mv apache-tomcat-8.0.30 /usr/local/tomcat

# ls /usr/local/tomcat

bin/ //主程序目录

lib/ //库文件目录

logs/ //日志目录

temp/ //临时目录

work/ //自动编译目录jsp代码转换servlet

conf/ //配置文件目录

webapps/ //页面目录

如果检查端口时，8005端口启动非常慢，默认tomcat启动需要从/dev/random读取大量的随机数据，默认该设备生成随机数据的速度很慢，可用使用下面的命令用urandom替换random

# mv /dev/random /dev/random.bak

# ln -s /dev/urandom /dev/random

或者

# yum install rng-tools

# systemctl start rngd

# systemctl enable rngd

均为随机生成RANDOM

3)

Tomcat配置文件

/usr/local/tomcat/webapps/ROOT/test.jsp // 网页根目录

##################################################################################################

修改server.xml配置文件

\*\*\* 严格区分大小写

\*\*\* 有头有尾

<Server>

<Service>

<Connector port=8080 />

<Connector port=8009 />

<Engine name="Catalina" defaultHost="localhost">

<Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true"> //

</Host>

<Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

</Host>

#################################################################################

git依赖ssh服务 (客户端 读 写)

git clone root@ip:/目录

git服务 (客户端 只 读)

git clone git://ip

以上都需要客户端都需要安装git服务

(客户端 只 读)

git 服务器(代码 配置 文档) http://

http不需要客户端安装git服务

非交互生成秘钥

ssh-keygen -f /root/.ssh/id\_rsa -N '' //-f 定义秘钥存放位置 -N 定义密码

###########################################################################################

内部私有地址

A 10.0.0.0~10.255.255.255

B 172.16.0.0~172.31.255.255

C 192.168.0.0~192.168.255.255

###################################################################################

NTP

\*\*\* 层数不要大于15层 可以等于

####################################################################################

lvs

NAT模式

优点:效率很高 编写简单

缺点:不带健康检查 适合小规模集群规模

\*\*\* 配置完eth0:0 后不生效 因为 network 与 NetworkManager 共同在管理网卡 关闭NetworkManager