# Nginx

**By Brother.Lee 20171211 Shanghai**

# 一.Nginx环境搭建

## 1.安装wget

yum install wget

## 2.安装gcc

yum install gcc-c++

## 3.安装pcre

yum -y install pcre pcre-devel

## 4.依赖的解压包

yum -y install zlib zlib-devel

## 5.安装openssl

yum install -y openssl openssl-devel

## 6.wget下载1.12.1是稳定版

wget -c http://nginx.org/download/nginx-1.12.1.tar.gz

## 7.解压

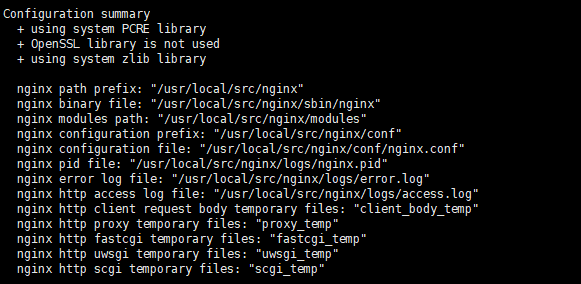
tar –xvf nginx-1.12.1.tar.gz

解压后最好不要修改名称(mv nginx-1.12.1 nginx)

## 8.进行configure配置不报错

在有comfigure的目录下执行下面命令(注意:这里的结尾nginx不要和你的目录名字相同了!!!)

./configure --prefix=/usr/local/nginx



此后会将sbin文件等目录放在这个文件夹下/usr/local/nginx

## 9.编译安装

make && make install

## 10.启动nginx

Cd /usr/local/src/nginx下:看到如下目录

Conf配置文件,html网页文件,logs日志文件,sbin主要二进制程序

启动

./nginx

重启

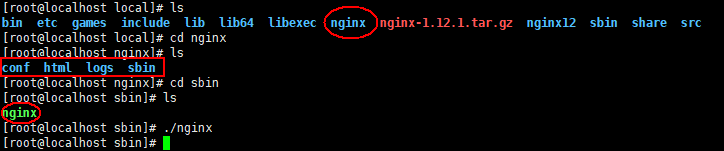
./nginx –s reload

关闭

./nginx –s quit

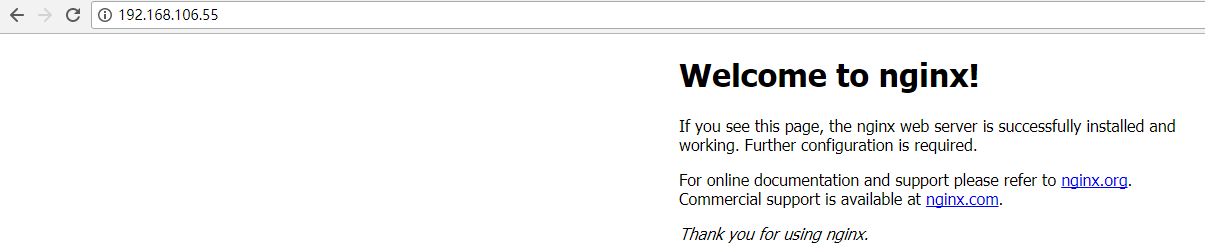
./nginx –s stop

./nginx -s quit:此方式停止步骤是待nginx进程处理任务完毕进行停止。  
./nginx -s stop:此方式相当于先查出nginx进程id再使用kill命令强制杀掉进程。

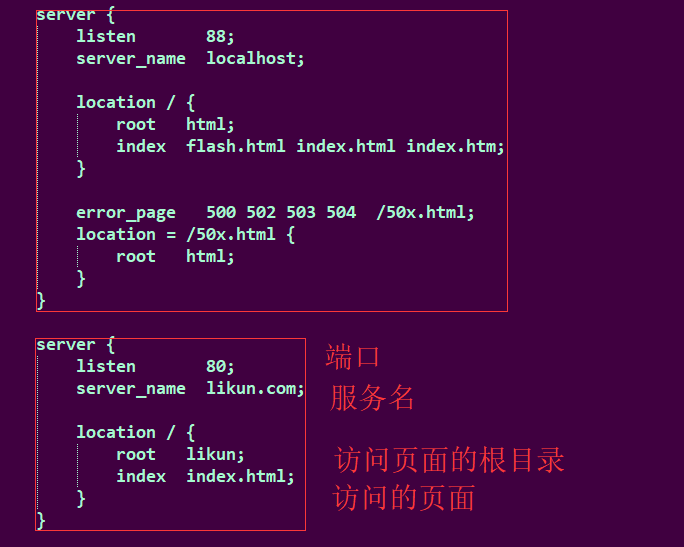


启动后登陆网页

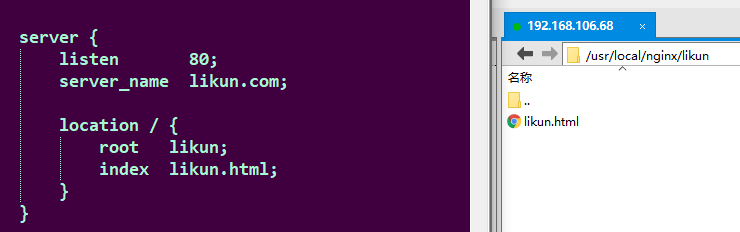
192.168.106.55:80(80也可以不敲)



# 二.nginx配置虚拟主机



12.设置自己的域名网站



在windows中的hosts下配置一下域名

C:\Windows\System32\drivers\etc\hosts

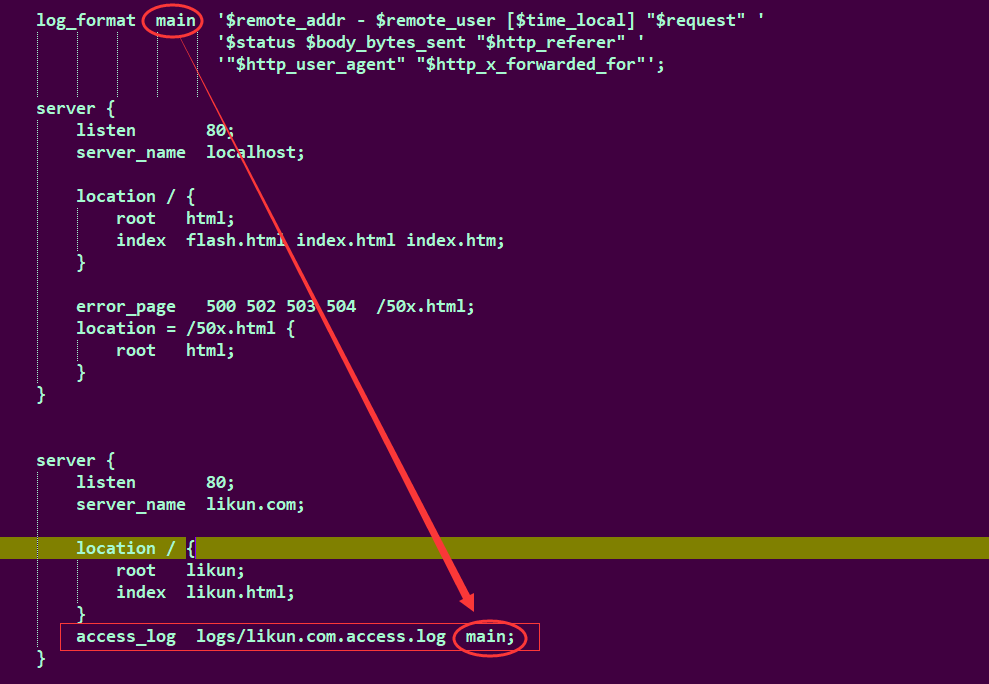


这样就可以访问自己的网站啦:



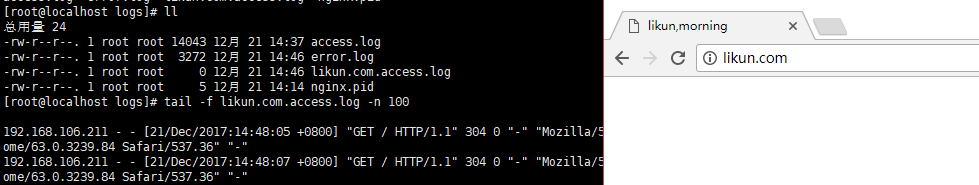
# 三.nginx日志文件

先打开日志文件,文件夹/日志名称/日志格式



配置完成后,reload后会在logs日志文件下生成你配置的log文件.

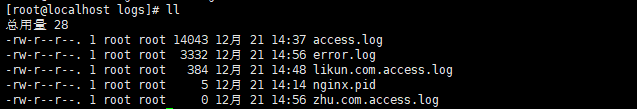
刷新页面两次,生成的日志信息就保存在这个log下.



日志互相独立



互相记录各自的日志信息,互不影响.



# 四.nginx日志文件切分

第一步:分析如何去实现日志切分,编写shell脚本

log.sh

**#!/bin/sh**BASE\_DIR=/usr/local/nginx  
BASE\_FILE\_NAME=likun.com.access.log  
  
CURRENT\_PATH=$BASE\_DIR/logs  
BAK\_PATH=$BASE\_DIR/datalogs  
  
CURRENT\_FILE=$CURRENT\_PATH/$BASE\_FILE\_NAME  
BAK\_TIME=*`*/bin/date -d yesterday +%Y%m%d%H%M*`*BAK\_FILE=$BAK\_PATH/$BAK\_TIME-$BASE\_FILE\_NAME  
echo $BAK\_FILE  
  
$BASE\_DIR/sbin/nginx -s stop  
  
mv $CURRENT\_FILE $BAK\_FILE  
  
$BASE\_DIR/sbin/nginx

第二步:对脚本赋权限chmod 777 (**log.sh并存放在sbin目录下**)

第三步:在nginx目录下mkdir datalogs

第四步:定时任务对脚本进行调度:在log.sh目录下执行

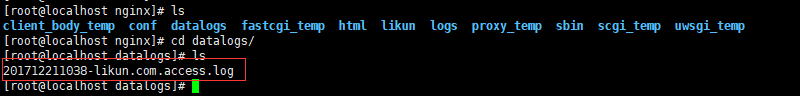
crontab -e

输入

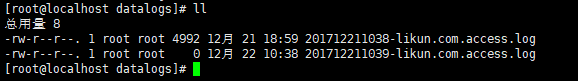
\*/1 \* \* \* \* sh /usr/local/nginx/sbin/log.sh

每分钟执行一下log.sh脚本

验证:



会安装时间每分钟产生一个日志文件!



删除crontab

crontab –e 清除掉里面的内容,便不会再产生日志文件.

# 五.nginx\_location配置

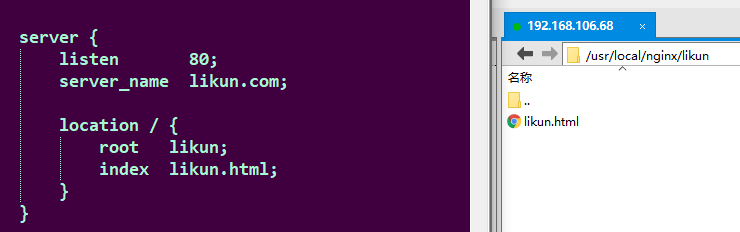
location语法:表示uri方式定位

基础语法有三种:

location = pattern{}精准匹配

location pattern{}一般匹配

location ~ pattern{}正则匹配



# 六.nginx反向代理

* nginx反向代理proxy与负载均衡upstream
* 配置反向代理proxy:proxy\_pass url地址
* 配置负载均衡upstream:upstream
* 官方配置;http://nginx.org/en/docs
* 注意:反向代理之后获取客户端ip地址为nginx服务器地址,这里需要nginx进行forward,设置真实的ip地址:

设置客户端真实ip地址

proxy\_set\_header X-real-ip $remote\_addr

# 附录: nginx配置文件说明

#user nobody;  
  
#开启进程数 <=CPU数   
worker\_processes 1;  
  
#错误日志保存位置  
#error\_log logs/error.log;  
#error\_log logs/error.log notice;  
#error\_log logs/error.log info;  
  
#进程号保存文件  
#pid logs/nginx.pid;  
  
#每个进程最大连接数（最大连接=连接数x进程数）每个worker允许同时产生多少个链接，默认1024  
events {  
 worker\_connections 1024;  
}  
  
  
http {  
 #文件扩展名与文件类型映射表  
 include mime.types;  
 #默认文件类型  
 default\_type application/octet-stream;  
  
 #日志文件输出格式 这个位置相于全局设置  
 log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '  
 '$status $body\_bytes\_sent "$http\_referer" '  
 '"$http\_user\_agent" "$http\_x\_forwarded\_for"';  
  
 #请求日志保存位置  
 #access\_log logs/access.log main;  
   
 #打开发送文件  
 sendfile on;  
 #tcp\_nopush on;  
  
 #keepalive\_timeout 0;  
 #连接超时时间  
 keepalive\_timeout 65;  
  
 #打开gzip压缩  
 #gzip on;  
   
 #设定请求缓冲  
 #client\_header\_buffer\_size 1k;  
 #large\_client\_header\_buffers 4 4k;  
   
 #设定负载均衡的服务器列表  
 #upstream myproject {  
 #weigth参数表示权值，权值越高被分配到的几率越大  
 #max\_fails 当有#max\_fails个请求失败，就表示后端的服务器不可用，默认为1，将其设置为0可以关闭检查  
 #fail\_timeout 在以后的#fail\_timeout时间内nginx不会再把请求发往已检查出标记为不可用的服务器  
 #}  
   
 #webapp  
 #upstream myapp {   
 # server 192.168.1.171:8080 weight=1 max\_fails=2 fail\_timeout=30s;   
 # server 192.168.1.172:8080 weight=1 max\_fails=2 fail\_timeout=30s;   
 #}   
  
 #配置虚拟主机，基于域名、ip和端口  
 server {  
 #监听端口  
 listen 80;  
 #监听域名  
 server\_name localhost;  
  
 #charset koi8-r;  
   
 #nginx访问日志放在logs/host.access.log下，并且使用main格式（还可以自定义格式）  
 #access\_log logs/host.access.log main;  
  
 #返回的相应文件地址  
 location / {  
 #设置客户端真实ip地址  
 #proxy\_set\_header X-real-ip $remote\_addr;   
 #负载均衡反向代理  
 #proxy\_pass http://myapp;  
   
 #返回根路径地址（相对路径:相对于/usr/local/nginx/）  
 root html;  
 #默认访问文件  
 index index.html index.htm;  
 }  
  
 #配置反向代理tomcat服务器：拦截.jsp结尾的请求转向到tomcat  
 #location ~ \.jsp$ {  
 # proxy\_pass http://192.168.1.171:8080;  
 #}   
   
 #error\_page 404 /404.html;  
 # redirect server error pages to the static page /50x.html  
 #  
   
 #错误页面及其返回地址  
 error\_page 500 502 503 504 /50x.html;  
 location = /50x.html {  
 root html;  
 }  
  
 # proxy the PHP scripts to Apache listening on 127.0.0.1:80  
 #  
 #location ~ \.php$ {  
 # proxy\_pass http://127.0.0.1;  
 #}  
  
 # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000  
 #  
 #location ~ \.php$ {  
 # root html;  
 # fastcgi\_pass 127.0.0.1:9000;  
 # fastcgi\_index index.php;  
 # fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name;  
 # include fastcgi\_params;  
 #}  
  
 # deny access to .htaccess files, if Apache's document root  
 # concurs with nginx's one  
 #  
 #location ~ /\.ht {  
 # deny all;  
 #}  
 }  
   
 #虚拟主机配置：  
 server {  
 listen 1234;  
 server\_name bhz.com;  
 location / {  
 #正则表达式匹配uri方式：在/usr/local/nginx/bhz.com下 建立一个test123.html 然后使用正则匹配  
 #location ~ test {  
 ## 重写语法：if return （条件 = ~ ~\*）  
 #if ($remote\_addr = 192.168.1.200) {  
 # return 401;  
 #}   
   
 #if ($http\_user\_agent ~\* firefox) {  
 # rewrite ^.\*$ /firefox.html;  
 # break;  
 #}   
   
 root bhz.com;  
 index index.html;  
 }  
   
 #location /goods {  
 # rewrite "goods-(\d{1,5})\.html" /goods-ctrl.html;  
 # root bhz.com;  
 # index index.html;  
 #}  
   
 #配置访问日志  
 access\_log logs/bhz.com.access.log main;  
 }  
   
  
  
 # another virtual host using mix of IP-, name-, and port-based configuration  
 #  
 #server {  
 # listen 8000;  
 # listen somename:8080;  
 # server\_name somename alias another.alias;  
  
 # location / {  
 # root html;  
 # index index.html index.htm;  
 # }  
 #}  
  
  
 # HTTPS server  
 #  
 #server {  
 # listen 443 ssl;  
 # server\_name localhost;  
  
 # ssl\_certificate cert.pem;  
 # ssl\_certificate\_key cert.key;  
  
 # 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;  
 # }  
 #}  
  
}