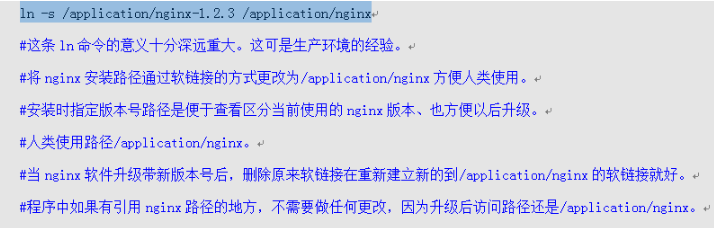
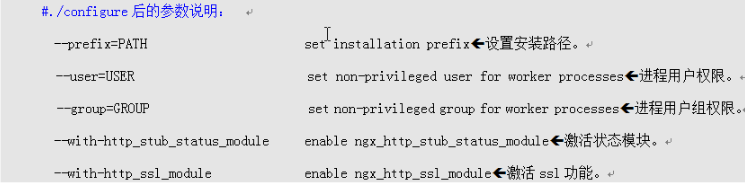
# Nginx

Nginx工作模式是一个master进程+N个worker进程

Prce是为了使nginx支持http Rewire模块

添加nginx用户





Nginx重要目录

## 增加一个nginx实例

# Location写法

location = / {

# 精确匹配 / ，主机名后面不能带任何字符串

[ configuration A ]

}

location / {

# 因为所有的地址都以 / 开头，所以这条规则将匹配到所有请求

# 但是正则和最长字符串会优先匹配

[ configuration B ]

}

location /documents/ {

# 匹配任何以 /documents/ 开头的地址，匹配符合以后，还要继续往下搜索

# 只有后面的正则表达式没有匹配到时，这一条才会采用这一条

[ configuration C ]

}

location ~ /documents/Abc {

# 匹配任何以 /documents/ 开头的地址，匹配符合以后，还要继续往下搜索

# 只有后面的正则表达式没有匹配到时，这一条才会采用这一条

[ configuration CC ]

}

location ^~ /images/ {

# 匹配任何以 /images/ 开头的地址，匹配符合以后，停止往下搜索正则，采用这一条。

[ configuration D ]

}

location ~\* \.(gif|jpg|jpeg)$ {

# 匹配所有以 gif,jpg或jpeg 结尾的请求

# 然而，所有请求 /images/ 下的图片会被 config D 处理，因为 ^~ 到达不了这一条正则

[ configuration E ]

}

location /images/ {

# 字符匹配到 /images/，继续往下，会发现 ^~ 存在

[ configuration F ]

}

location /images/abc {

# 最长字符匹配到 /images/abc，继续往下，会发现 ^~ 存在

# F与G的放置顺序是没有关系的

[ configuration G ]

}

location ~ /images/abc/ {

# 只有去掉 config D 才有效：先最长匹配 config G 开头的地址，继续往下搜索，匹配到这一条正则，采用

[ configuration H ]

}

location ~\* /js/.\*/\.js

* 已=开头表示精确匹配  
  如 A 中只匹配根目录结尾的请求，后面不能带任何字符串。
* ^~ 开头表示uri以某个常规字符串开头，不是正则匹配
* ~ 开头表示区分大小写的正则匹配;
* ~\* 开头表示不区分大小写的正则匹配
* / 通用匹配, 如果没有其它匹配,任何请求都会匹配到

顺序 no优先级：  
(location =) > (location 完整路径) > (location ^~ 路径) > (location ~,~\* 正则顺序) > (location 部分起始路径) > (/)

上面的匹配结果  
按照上面的location写法，以下的匹配示例成立：

* / -> config A  
  精确完全匹配，即使/index.html也匹配不了
* /downloads/download.html -> config B  
  匹配B以后，往下没有任何匹配，采用B
* /images/1.gif -> configuration D  
  匹配到F，往下匹配到D，停止往下
* /images/abc/def -> config D  
  最长匹配到G，往下匹配D，停止往下  
  你可以看到 任何以/images/开头的都会匹配到D并停止，FG写在这里是没有任何意义的，H是永远轮不到的，这里只是为了说明匹配顺序
* /documents/document.html -> config C  
  匹配到C，往下没有任何匹配，采用C
* /documents/1.jpg -> configuration E  
  匹配到C，往下正则匹配到E
* /documents/Abc.jpg -> config CC  
  最长匹配到C，往下正则顺序匹配到CC，不会往下到E

所以实际使用中，个人觉得至少有三个匹配规则定义，如下：

#直接匹配网站根，通过域名访问网站首页比较频繁，使用这个会加速处理，官网如是说。

#这里是直接转发给后端应用服务器了，也可以是一个静态首页

# 第一个必选规则

location = / {

proxy\_pass http://tomcat:8080/index

}

# 第二个必选规则是处理静态文件请求，这是nginx作为http服务器的强项

# 有两种配置模式，目录匹配或后缀匹配,任选其一或搭配使用

location ^~ /static/ {

root /webroot/static/;

}

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

root /webroot/res/;

}

#第三个规则就是通用规则，用来转发动态请求到后端应用服务器

#非静态文件请求就默认是动态请求，自己根据实际把握

#毕竟目前的一些框架的流行，带.php,.jsp后缀的情况很少了

location / {

proxy\_pass http://tomcat:8080/

}

# Rewrite规则

rewrite功能就是，使用nginx提供的全局变量或自己设置的变量，结合正则表达式和标志位实现url重写以及重定向。rewrite只能放在server{},location{},if{}中，并且只能对域名后边的除去传递的参数外的字符串起作用，例如 http://seanlook.com/a/we/index.php?id=1&u=str 只对/a/we/index.php重写。语法rewrite regex replacement [flag];

如果相对域名或参数字符串起作用，可以使用全局变量匹配，也可以使用proxy\_pass反向代理。

表明看rewrite和location功能有点像，都能实现跳转，主要区别在于rewrite是在同一域名内更改获取资源的路径，而location是对一类路径做控制访问或反向代理，可以proxy\_pass到其他机器。很多情况下rewrite也会写在location里，它们的执行顺序是：

1. 执行server块的rewrite指令
2. 执行location匹配
3. 执行选定的location中的rewrite指令

如果其中某步URI被重写，则重新循环执行1-3，直到找到真实存在的文件；循环超过10次，则返回500 Internal Server Error错误。

**flag标志位**

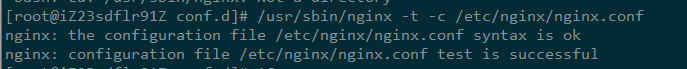
* last : 相当于Apache的[L]标记，表示完成rewrite
* break : 停止执行当前虚拟主机的后续rewrite指令集
* redirect : 返回302临时重定向，地址栏会显示跳转后的地址
* permanent : 返回301永久重定向，地址栏会显示跳转后的地址

因为301和302不能简单的只返回状态码，还必须有重定向的URL，这就是return指令无法返回301,302的原因了。这里 last 和 break 区别有点难以理解：

1. last一般写在server和if中，而break一般使用在location中
2. last不终止重写后的url匹配，即新的url会再从server走一遍匹配流程，而break终止重写后的匹配
3. break和last都能组织继续执行后面的rewrite指令

# 经验

/usr/sbin/nginx -t -c /etc/nginx/nginx.conf 查看nginx配置文件语法是否正确



配置文件模板

#user nobody;

worker\_processes 1;

#error\_log logs/error.log;

#error\_log logs/error.log notice;

#error\_log logs/error.log info;

#pid logs/nginx.pid;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

#log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

# '$status $body\_bytes\_sent "$http\_referer" '

# '"$http\_user\_agent" "$http\_x\_forwarded\_for"';

#access\_log logs/access.log main;

sendfile on;

#tcp\_nopush on;

#keepalive\_timeout 0;

keepalive\_timeout 65;

#gzip on;

upstream myweb.com {

ip\_hash;

server 192.168.254.10:8081;

server 192.168.254.9:8081;

}

server {

listen 192.168.254.100:80;

server\_name www.myweb.com;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

root /usr/local/tomcat-1/webapps/ROOT;

index index.jsp index.html index.htm;

proxy\_pass http://myweb.com; #转向tomcat处理

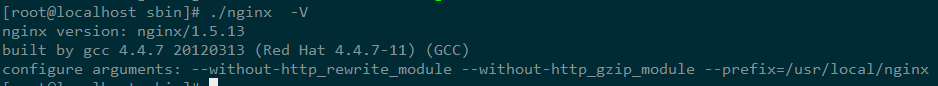
}

}

}

# 添加新模块

原已经安装好的nginx，现在需要添加一个未被编译安装的模块:  
  
nginx -V 可以查看原来编译时都带了哪些参数



添加的参数:   
--with-http\_geoip\_module

步骤1、使用参数重新配置

./configure --with-http\_geoip\_module --without-http\_rewrite\_module --without-http\_gzip\_module --prefix=/usr/local/nginx



2. 编译:  
make  
#不要make install，否则就是覆盖安装

3. 替换nginx二进制文件:  
cp /usr/local/nginx/sbin/nginx /usr/local/nginx/sbin/nginx.bak  
cp ./objs/nginx /usr/local/nginx/sbin/

# 基于城市IP地址进行转发

http://www.bubuko.com/infodetail-1424732.html

安装 MaxMind 的 GeoIP 库  
MaxMind 提供了免费的 IP 地域数据库（GeoIP.dat），不过这个数据库文件是二进制的，需要用 GeoIP 库来读取，所以除了要下载 GeoIP.dat 文件外（见下一步），还需要安装能读取这个文件的库。

# wget <http://geolite.maxmind.com/download/geoip/api/c/GeoIP.tar.gz>  
# tar -zxvf GeoIP.tar.gz  
# cd GeoIP-1.4.6  
# ./configure  
# make; make install  
刚才安装的库自动安装到 /usr/local/lib 下，所以这个目录需要加到动态链接配置里面以便运行相关程序的时候能自动绑定到这个 GeoIP 库：

# echo '/usr/local/lib' > /etc/ld.so.conf.d/geoip.conf  
# ldconfig

下载 IP 数据库  
MaxMind 提供了免费的 IP 地域数据库，这个数据库是二进制的，不能用文本编辑器打开，需要上面的 GeoIP 库来读取：

# wget <http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat.gz>  
# gunzip GeoIP.dat.gz

配置 Nginx  
最后是配置 nginx，在相关地方加上如下的配置就可以了：

# vi /etc/nginx/nginx.conf  
...  
geoip\_country /home/vpsee/GeoIP.dat;

fastcgi\_param GEOIP\_COUNTRY\_CODE $geoip\_country\_code;  
fastcgi\_param GEOIP\_COUNTRY\_CODE3 $geoip\_country\_code3;  
fastcgi\_param GEOIP\_COUNTRY\_NAME $geoip\_country\_name;  
...

 if ($geoip\_country\_code = CN) {  
    root /home/vpsee/cn/;  
 }

这样，当来自中国的 IP 访问网站后就自动访问到预定的 /home/vpsee/cn 页面。关于 Nginx + GeoIP 还有很多有用的用法，比如做个简单的 CDN，来自中国的访问自动解析到国内服务器、来自美国的访问自动转向到美国服务器等。MaxMind 还提供了全球各个城市的 IP 信息，还可以下载城市 IP 数据库来针对不同城市做处理。

# 基于用户特征进行转发

https://github.com/CNSRE/ABTestingGateway