

## 第1章 目录索引-下载服务

- 0.官方地址
- 1.应用场景
- 2.参数说明
- 3.配置文件
- 4.创建测试数据
- 5.访问页面

## 第2章 状态监控

- 0.官方地址
- 1.状态字段解释
- 2.配置文件
- 3.访问测试

## 第3章 Nginx 基于IP的访问控制

- 0.官方地址
- 1.配置语法
- 2.配置案例1: 拒绝windows访问www域名
- 3.windows测试访问
- 4.使用虚拟机测试访问
- 5.配置案例2: 只允许windows访问, 其他全部拒绝
- 6.windows测试访问
- 7.使用虚拟机测试访问

## 第4章 Nginx 基于用户认证的访问控制

- 0.官方配置
- 1.配置语法
- 2.配置文件
- 3.访问测试

## 第5章 Nginx 请求限制

- 0.官方地址
- 1.配置语法
- 2.配置文件
- 3.访问测试
- 4.查看访问日志
- 5.查看错误日志
- 6.为什么限制请求的效果更好

## 第6章 Nginx 内置变量

- 1.Nginx内置变量介绍
- 2.Nginx内置变量

## 第7章 Nginx 添加第三方模块

- 1.第三方模块介绍
- 2.添加第三方模块实战 - echo模块

## 第8章 Nginx location匹配

- 0.官方网址
- 1.location语法介绍
- 2.location语法优先级
- 3.配置location匹配规则实战
- 4.测试location匹配规则

## 第9章 Nginx rewrite跳转

- 0.官方地址
- 1.ngx\_http\_rewrite\_module模块介绍
- 2.if指令
  - 2.1 官方文档:
  - 2.2 指令说明:
  - 2.3 指令语法:

- 2.4 匹配规则:
- 2.5 创建测试配置文件:
- 2.6 测试方法:
- 3.return指令
  - 3.1 官方文档
  - 3.2 指令说明
  - 3.3 指令语法
  - 3.4 实验配置
  - 3.5 测试方法
- 4.set指令
  - 4.1 官方文档
  - 4.2 指令说明
  - 4.3 指令语法
  - 4.4 实验配置
  - 4.5 测试方法
- 5.break指令
  - 5.1 官方文档
  - 5.2 指令说明
  - 5.3 指令语法
  - 5.4 实验配置
  - 5.5 实验结果
- 6.rewrite指令--重点掌握
  - 6.1 官方文档
  - 6.2 指令说明
  - 6.3 指令语法
  - 6.4 flag标识符说明
  - 6.4 redirect和permanent实验
    - 6.4.1 permanent 301 永久跳转 实验
    - 6.4.2 redirect 302 临时跳转 实验
  - 6.5 break和last实验
    - 6.5.1 last实验
    - 6.5.2 break实验
  - 6.6 生产跳转场景
    - 6.6.1 URL跳转
    - 6.6.2 http跳转https
    - 6.6.3 URL拼接跳转
    - 6.6.4 带参数跳转

## 第10章 制作内网YUM仓库

- 1.为什么需要私有YUM仓库
- 2.需要的软件
- 3.配置nginx索引模块
- 4.安装createrepo
- 5.准备软件仓库目录并下载需要的软件
- 6.生成yum元数据
- 7.客户端生成本地源
- 8.客户端测试安装
- 9.更新软件包的操作步骤:

# 第1章 目录索引-下载服务

## 0.官方地址

# 1.应用场景

使用nginx作为简易的文件下载服务器

## 2.参数说明

参数解释:

```
1 Syntax: autoindex on | off;
2 Default: autoindex off;
3 Context: http, server, location
4
5 # autoindex 常用参数
6 autoindex_exact_size off;
7 默认为 on, 显示出文件的确切大小, 单位是 bytes。
8 修改为 off, 显示出文件的大概大小, 单位是 KB 或者 MB 或者 GB。
9
10 autoindex_localtime on;
11 默认为 off, 显示的文件时间为 GMT 时间。
12 修改为 on, 显示的文件时间为文件的服务器时间。
13
14 charset utf-8,gbk;
15 默认中文目录乱码, 添加上解决乱码
```

## 3.配置文件

```
1 cat >/etc/nginx/conf.d/download.conf <<EOF
2 server {
3     listen 80;
4     server_name download.oldzhang.com;
5     location / {
6         root /usr/share/nginx/html/download;
7         charset utf-8,gbk;
8         autoindex on;
9         autoindex_localtime on;
10        autoindex_exact_size off;
11    }
12 }
13 EOF
```

## 4.创建测试数据

```
1 touch /usr/share/nginx/html/download/{1..10}.txt
```

## 5.访问页面

download.oldzhang.com

# Index of /

<a href="#">../</a>		
<a href="#">1.txt</a>	30-Jul-2019 18:03	0
<a href="#">10.txt</a>	30-Jul-2019 18:03	0
<a href="#">2.txt</a>	30-Jul-2019 18:03	0
<a href="#">3.txt</a>	30-Jul-2019 18:03	0
<a href="#">4.txt</a>	30-Jul-2019 18:03	0
<a href="#">5.txt</a>	30-Jul-2019 18:03	0
<a href="#">6.txt</a>	30-Jul-2019 18:03	0
<a href="#">7.txt</a>	30-Jul-2019 18:03	0
<a href="#">8.txt</a>	30-Jul-2019 18:03	0
<a href="#">9.txt</a>	30-Jul-2019 18:03	0

## 第2章 状态监控

### 0.官方地址

1 | [http://nginx.org/en/docs/http/nginx\\_http\\_stub\\_status\\_module.html](http://nginx.org/en/docs/http/nginx_http_stub_status_module.html)

### 1.状态字段解释

```
1 Active connections # 当前活动的连接数
2 accepts # 当前的总连接数 TCP
3 handled # 成功的连接数 TCP
4 requests # 总的 http 请求数
5 Reading # 请求
6 Writing # 响应
7 waiting # 等待的请求数, 开启了 keepalive
8 # 注意, 一次 TCP 的连接, 可以发起多次 http 的请求, 如下参数可配置进行验证
9 keepalive_timeout 0; # 类似于关闭长连接
10 keepalive_timeout 65; # 65s 没有活动则断开连接
```

### 2.配置文件

```
1 cat > /etc/nginx/conf.d/status.conf <<EOF
2 server {
3     listen 80;
4     server_name status.oldzhang.com;
5     stub_status on;
6     access_log off;
7 }
8 EOF
```

### 3.访问测试

```
1 [root@web01 ~]# curl status.oldboy.com
2 Active connections: 1
3 server accepts handled requests
4 4 4 6
5 Reading: 0 Writing: 1 Waiting: 0
```

## 第3章 Nginx 基于IP的访问控制

### 0.官方地址

```
1 http://nginx.org/en/docs/http/nginx_http_access_module.html
```

### 1.配置语法

```
1 #允许配置语法
2 Syntax: allow address | CIDR | unix: | all;
3 Default: -
4 Context: http, server, location, limit_except
5
6 #拒绝配置语法
7 Syntax: deny address | CIDR | unix: | all;
8 Default: -
9 Context: http, server, location, limit_except\
```

### 2.配置案例1: 拒绝windows访问www域名

```
1 cat > /etc/nginx/conf.d/01-www.conf <<EOF
2 server {
3     listen      80;
4     server_name www.oldzhang.com;
5     location / {
6         root    /usr/share/nginx/html/www;
7         index   index.html index.htm;
8         deny 10.0.0.1;
9         allow all;
10    }
11 }
12 EOF
```

### 3.windows测试访问

[www.oldboy.com](http://www.oldboy.com)



403 Forbidden

nginx/1.16.0

### 4.使用虚拟机测试访问

```
1 [root@web01 ~]# curl www.oldzhang.com
2 www
```

## 5.配置案例2: 只允许windows访问, 其他全部拒绝

```
1 cat >/etc/nginx/conf.d/01-www.conf<<EOF
2 server {
3     listen      80;
4     server_name www.oldzhang.com;
5     location / {
6         root    /usr/share/nginx/html/www;
7         index   index.html index.htm;
8         allow 10.0.0.1;
9         deny all;
10    }
11 }
12 EOF
```

## 6.windows测试访问

[www.oldboy.com](http://www.oldboy.com)



WWW

## 7.使用虚拟机测试访问

```
1 [root@web01 ~]# curl www.oldzhang.com
2 <html>
3 <head><title>403 Forbidden</title></head>
4 <body>
5 <center><h1>403 Forbidden</h1></center>
6 <hr><center>nginx/1.16.0</center>
7 </body>
8 </html>
```

# 第4章 Nginx 基于用户认证的访问控制

## 0.官方配置

```
1 http://nginx.org/en/docs/http/nginx_http_auth_basic_module.html
```

## 1.配置语法

```
1 #访问提示字符串
2 Syntax: auth_basic string| off;
3 Default: auth_basic off;
4 Context: http, server, location, limit_except
5
6 #账户密码文件
7 Syntax: auth_basic_user_file file;
8 Default: -
9 Context: http, server, location, limit_except
```

## 2.配置文件

需要安装 httpd-tools, 该包中携带了 htpasswd 命令

```
1 [root@web01 ~]# yum install httpd-tools -y
```

创建新的密码文件, -c 创建新文件 -b 允许命令行输入密码

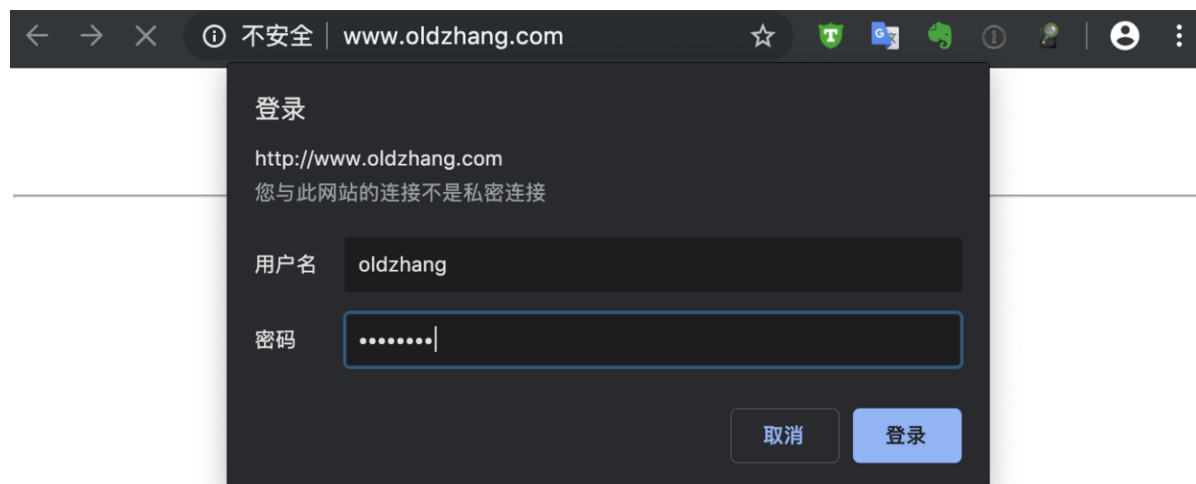
```
1 [root@web01 ~]# htpasswd -b -c /etc/nginx/auth_conf oldzhang oldzhang
2 Adding password for user oldzhang
```

nginx 配置调用

```
1 cat >/etc/nginx/conf.d/01-www.conf <EOF
2 server {
3     listen      80;
4     server_name www.oldzhang.com;
5     location / {
6         auth_basic "Auth access Blog Input your Passwd!";
7         auth_basic_user_file auth_conf;
8         root     /usr/share/nginx/html/www;
9         index    index.html index.htm;
10    }
11 }
12 EOF
```

## 3.访问测试

[www.oldzhang.com](http://www.oldzhang.com)



# 第5章 Nginx 请求限制

## 0.官方地址

1 | [http://nginx.org/en/docs/http/nginx\\_http\\_limit\\_req\\_module.html](http://nginx.org/en/docs/http/nginx_http_limit_req_module.html)

## 1.配置语法

```
1 #模块名 ngx_http_limit_req_module
2 Syntax: limit_req_zone key zone=name:size rate=rate;
3 Default: -
4 Context: http
5
6 #引用限速模块
7 Syntax: limit_conn zone number [burst=number] [nodelay];
8 Default: -
9 Context: http, server, location
```

参数解释:

```
1 定义一条规则
2 limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;
3 limit_req_zone $binary_remote_addr zone=two:10m rate=2r/s;
4
5 limit_req_zone                                #引用限速模块
6 $binary_remote_addr                            #判定条件，每个请求的IP
7 zone=one:10m                                   #定义一个zone名称
8 rate=1r/s;                                     #限制速度，1秒1次
9
10 引用一条限速规则
11 limit_req zone=two burst=5 nodelay;
12
13 limit_req                                     #引用限速规则语法
14 zone=one                                     #引用哪一条规则
15 burst=5                                     #令牌桶，允许排队的数量
16 nodelay;                                     #如果不希望在请求被限制时延迟过多的请求，则应使用参
    数nodelay
```

## 2.配置文件

```
1 cat >/etc/nginx/conf.d/01-www.conf <<EOF
2 limit_req_zone $binary_remote_addr zone=req_zone:10m rate=1r/s;
3 server {
4     listen      80;
5     server_name www.oldzhang.com;
6     limit_req zone=req_zone burst=3 nodelay;
7     access_log /var/log/nginx/www.access.log main;
8     location / {
9         root /usr/share/nginx/html/www;
10        index index.html index.htm;
11    }
12 }
13 EOF
```



## 3.访问测试

ab压测

```
1 yum install httpd-tools -y
2 ab -n 20 -c 2 http://www.oldzhang.com/
```

for循环

```
1 一秒1次
2 for i in {1..10000};do curl -I 10.0.0.7;sleep 1;done
3
4 一秒2次
5 for i in {1..10000};do curl -I 10.0.0.7;sleep 0.5;done
6
7 一秒5次
8 for i in {1..10000};do curl -I 10.0.0.7;sleep 0.2;done
```

## 4.查看访问日志

```
1 tail -f /var/log/nginx/www.access.log
```

## 5.查看错误日志

```
1 tail -f /var/log/nginx/error.log
```

## 6.为什么限制请求的效果更好

我们先来回顾一下 http 协议的连接与请求，首先 HTTP 是建立在 TCP 基础之上，在完成 HTTP 请求需要先建立 TCP 三次握手（称为 TCP 连接），在连接的基础上在完成 HTTP 的请求。

所以多个 HTTP 请求可以建立在一次 TCP 连接之上，那么我们对请求的精度限制，当然比对一个连接的限制会更加的有效，因为同一时刻只允许一个 TCP 连接进入，但是同一时刻多个 HTTP 请求可以通过一个 TCP 连接进入。所以针对 HTTP 的请求限制才是比较优的解决方案。

# 第6章 Nginx 内置变量

## 1.Nginx内置变量介绍

变量介绍：

```
1 nginx包含了很多内置的变量，这些变量可以用来定义日志的格式或者跳转时作为判断条件。
```

官方地址：

```
1 http://nginx.org/en/docs/varindex.html
```

## 2.Nginx内置变量

```

1  $args                #存放URL中的参数，例如：
    http://www.ldbboyedu.com/index.php?id=526195417&partener=search 则args变量结果
    为：id=526195417&partener=search
2  $document_root       #当前请求资源的根目录，例如：/code/blog/
3  $document_uri        #只返回用户请求中的URI，不包含参数，例如：
    http://www.ldbboyedu.com/index.php
4  $remote_addr         #记录客户端 IP 地址，注意，是公网IP
5  $remote_user         #记录经过用户认证模块认证后的客户端用户名
6  $host                #请求的host名称
7  $request             #记录请求的方法以及请求的 http 协议
8  $request_method      #请求资源的方法，例如GET/PUT/POST
9  $scheme              #请求的协议，例如http或https
10 $status              #记录请求状态码(用于定位错误信息)
11 $server_addr         #Nginx服务器的IP地址
12 $server_name         #Nginx服务器的主机名
13 $server_port         #Nginx服务器的端口号
14 $body_bytes_sent     #发送给客户端的资源字节数，不包括响应头的大小
15 $bytes_sent          #发送给客户端的总字节数
16 $http_referer        #记录从哪个页面链接访问过来的
17 $http_user_agent     #记录客户端浏览器相关信息
18 $http_x_forwarded_for #记录客户端 IP 地址
19 $request_length       #请求的长度（包括请求行， 请求头和请求正文）
20 $request_filename    #请求的资源在系统中的路径，例如：
    /code/blog/index.html
21 $request_time         #请求花费的时间，单位为秒，精度毫秒
22 # 注:如果 Nginx 位于负载均衡器，nginx 反向代理之后，web 服务器无法直接获取到客 户端
    真实的 IP 地址。
23 # $remote_addr 获取的是反向代理的 IP 地址。反向代理服务器在转发请求的 http 头信息
    中，
24 # 增加 X-Forwarded-For 信息，用来记录客户端 IP 地址和客户端请求的服务器地址。

```

## 第7章 Nginx 添加第三方模块

### 1.第三方模块介绍

1 | nginx除了核心模块和官方的模块以外，还有很多第三方的模块，需要注意的是：添加新模块需要重新编译安装源码包

### 2.添加第三方模块实战 - echo模块

模块开源地址：

1 | <https://github.com/openresty/echo-nginx-module>

编译安装步骤：

```

1  yum install openssl-devel pcre-devel gcc -y
2  groupadd www -g 1000
3  useradd www -s /sbin/nologin -M -u 1000 -g 1000
4  mkdir /data/soft -p
5  cd /data/soft/
6  wget http://nginx.org/download/nginx-1.19.0.tar.gz
7  tar zxvf nginx-1.19.0.tar.gz
8  cd nginx-1.19.0/

```

```

9  ./configure \
10 --user=www \
11 --group=www \
12 --prefix=/opt/nginx-1.19.0 \
13 --with-http_stub_status_module \
14 --with-http_ssl_module \
15 --with-pcre
16 make && make install
17 cd /opt/
18 ln -s /opt/nginx-1.19.0 nginx
19 cp /opt/nginx/sbin/nginx /usr/sbin/
20 nginx -v
21 mkdir -p /opt/nginx/pid
22 mkdir -p /opt/nginx/conf.d
23 cat > /opt/nginx/conf/nginx.conf << 'EOF'
24 user www;
25 worker_processes 1;
26
27 error_log logs/error.log warn;
28 pid pid/nginx.pid;
29
30 events {
31     worker_connections 1024;
32 }
33
34 http {
35     include mime.types;
36     default_type application/octet-stream;
37
38     log_format main '$remote_addr - $remote_user [$time_local] "$request" '
39                     '$status $body_bytes_sent "$http_referer" '
40                     '"$http_user_agent" "$http_x_forwarded_for"';
41
42     access_log logs/access.log main;
43     sendfile on;
44     keepalive_timeout 65;
45     include /opt/nginx/conf.d/*.conf;
46 }
47 EOF
48 nginx -t

```

添加第三方模块:

```

1  cd /data/soft/
2  yum install git -y
3  git clone https://github.com/openresty/echo-nginx-module.git
4  ll -d /data/soft/echo-nginx-module/
5  cd /data/soft/nginx-1.19.0
6  ./configure \
7  --user=www \
8  --group=www \
9  --prefix=/opt/nginx-1.19.0 \
10 --with-http_stub_status_module \
11 --with-http_ssl_module \
12 --with-pcre \
13 --add-module=/data/soft/echo-nginx-module
14 make && make install

```

```
15 \cp /opt/nginx/sbin/nginx /usr/sbin/  
16 nginx -v
```

创建子配置文件:

```
1 cat > /opt/nginx/conf.d/echo.conf << 'EOF'  
2 server {  
3     listen      80;  
4     server_name www.echo.com;  
5     location / {  
6         echo "day day up!";  
7         echo $remote_addr;  
8         echo $http_user_agent;  
9     }  
10 }  
11 EOF  
12 nginx -t  
13 nginx -s reload
```

验证结果:

```
1 [root@web-7 ~]# curl 127.0.0.1  
2 day day up!  
3 127.0.0.1  
4 curl/7.29.0
```

## 第8章 Nginx location匹配

使用 Nginx location 可以控制访问网站的路径, 但一个 server 可以有多个 location 配置, 多个 location 的优先级该如何区分

### 0. 官方网址

```
1 http://nginx.org/en/docs/http/nginx_http_core_module.html#location
```

### 1. location 语法介绍

```
1 Syntax: location [ = | ~ | ~* | ^~ ] uri { ... }  
2 location @name { ... }  
3 Default: -  
4 Context: server, location
```

### 2. location 语法优先级

匹配符	匹配规则	优先级
=	精确匹配	1
^~	以某个字符串开头	2
~	区分大小写的正则匹配	3
~*	不区分大小写的正则匹配	4
!~	区分大小写不匹配的正则	5
!~*	不区分大小写不匹配的正则	6
/	通用匹配，任何请求都会匹配到	7

### 3.配置location匹配规则实战

```
1 cat >/etc/nginx/conf.d/01-www.conf << 'EOF'
2 server {
3     listen      80;
4     server_name www.oldzhang.com;
5     root        /usr/share/nginx/html/www;
6     location / {
7         return 200 "location /\n";
8     }
9     location = / {
10        return 200 "location = \n";
11    }
12
13    location /documents/ {
14        return 200 "location /documents/ \n";
15    }
16    location ^~ /images/ {
17        return 200 "location ^~ /images/ \n";
18    }
19    location ~* \.(gif|jpg|jpeg)$ {
20        return 200 "location ~* \.(gif|jpg|jpeg) \n";
21    }
22    access_log off;
23 }
24 EOF
```

### 4.测试location匹配规则

```
1 #精确匹配=/
2 [root@web01 ~]# curl www.oldzhang.com
3 location =
4
5 #没有满足的请求，所以匹配了/
6 [root@web01 ~]# curl www.oldzhang.com/oldzhang.html
7 location /
8
9 #匹配了/documents
10 [root@web01 ~]# curl www.oldzhang.com/documents/oldboy.html
11 location /documents/
```

```
12
13 #没有满足的条件, 匹配/
14 [root@web01 ~]# curl www.oldzhang.com/oldboy/documents/oldboy.html
15 location /
16
17 #正则匹配了文件名
18 [root@web01 ~]# curl www.oldzhang.com/oldboy.jpg
19 location ~* \.(gif|jpg|jpeg)
20
21 #~*匹配正则不区分大小写优先于/documents
22 [root@web01 ~]# curl www.oldzhang.com/documents/oldboy.jpg
23 location ~* \.(gif|jpg|jpeg)
24
25 #^~优先匹配于~*
26 [root@web01 ~]# curl www.oldzhang.com/images/oldboy.jpg
27 location ^~ /images/
```

## 第9章 Nginx rewrite跳转

### 0.官方地址

```
1 | https://nginx.org/en/docs/http/nginx\_http\_rewrite\_module.html
```

### 1.ngx\_http\_rewrite\_module模块介绍

```
1 | 在实际工作中, 我们经常会遇到需要修改用户的URL的请求, 例如如果用户访问http那就强制跳转到
2 | https,
3 | 或者判断用户浏览器类型, 并跳转到不同的站点。
4 | 在Nginx中, 跳转使用的是ngx_http_rewrite_module模块
5 |
6 | ngx_http_rewrite_module 模块包含了多个指令, 分别为:
7 | break          #中断配置
8 | if              #请求判断
9 | set             #设置变量
10 | return          #返回状态或URL
11 | rewrite         #对用户请求的URL或URI进行重写
```

### 2.if指令

#### 2.1 官方文档:

```
1 | http://nginx.org/en/docs/http/nginx\_http\_rewrite\_module.html#if
```

#### 2.2 指令说明:

```
1 | if用于条件判断, 并且可以根据判断的结果执行不同的配置, if可以配置在server或location中。
```

## 2.3 指令语法:

```
1  if (匹配条件) {
2      执行的动作
3  }
```

## 2.4 匹配规则:

```
1  =          #比较变量和字符串是否相等, 相等则为true, 不相等则为false
2  !=         #比较变量和字符串是否不相等, 不相等则为true, 不相等则为false
3
4  ~          #区分大小写的正则匹配, 匹配上则为true, 不匹配则为false
5  !~         #区分大小写的正则匹配, 不匹配则为true, 匹配则为false
6
7  ~*         #不区分大小写的正则匹配, 匹配上则为true, 不匹配则为false
8  !~*        #不区分大小写的正则匹配, 不匹配则为true, 匹配则为false
9
10 -f和!-f    #判断请求的文件是否存在
11 -d和!-d    #判断请求的目录是否存在
12 -e和!-e    #判断请求的文件/目录/软链接是否存在
13 -x和!-x    #判断请求的文件是否具有可执行权限
```

## 2.5 创建测试配置文件:

```
1  cat >/opt/nginx/conf.d/1f.conf << 'EOF'
2  server {
3      listen      8080;
4      server_name www.01dzhang.com;
5      root        /usr/share/nginx/html/www;
6
7      location /mall {
8          #客户端类型完全匹配iphone
9          if ($http_user_agent = iphone) {
10             echo "agent is iphone";
11         }
12
13         #客户端类型区分大小写匹配
14         if ($http_user_agent ~ Android) {
15             echo "agent is Android";
16         }
17
18         #客户端类型不区分大小写匹配
19         if ($http_user_agent ~* Chrome) {
20             echo "agent is Chrome";
21         }
22
23         #如果不是GET则输出
24         if ($request_method != GET){
25             echo "method is not GET";
26         }
27
28         #如果是IE浏览器, 直接返回404
29         if ($http_user_agent ~* IE){
30             return 404;
31         }
32     }
```

```

32
33     #都没匹配上则执行下面语句
34     echo "not match";
35     echo "agent ==> $http_user_agent";
36     echo "request ==> $request_method";
37 }
38 }
39 EOF

```

## 2.6 测试方法:

### 1.默认访问

```

1 [root@web-7 ~]# curl 127.0.0.1:8080/mall
2 not match
3 agent ==> curl/7.29.0
4 request ==> GET

```

### 2.测试完全匹配=

```

1 [root@web-7 ~]# curl -A "iphone" 127.0.0.1:8080/mall
2 agent is iphone
3
4 [root@web-7 ~]# curl -A "iphoneeee" 127.0.0.1:8080/mall
5 not match
6 agent ==> iphoneeee
7 request ==> GET
8
9 [root@web-7 ~]# curl -A "Iphone" 127.0.0.1:8080/mall
10 not match
11 agent ==> Iphone
12 request ==> GET

```

### 3.测试区分大小写匹配~

```

1 [root@web-7 ~]# curl -A "Android" 127.0.0.1:8080/mall
2 agent is Android
3
4 [root@web-7 ~]# curl -A "android" 127.0.0.1:8080/mall
5 not match
6 agent ==> android
7 request ==> GET
8
9 [root@web-7 ~]# curl -A "Androidd" 127.0.0.1:8080/mall
10 agent is Android
11
12 [root@web-7 ~]# curl -A "AAndroidD" 127.0.0.1:8080/mall
13 agent is Android
14
15 [root@web-7 ~]# curl -A "AndXroid" 127.0.0.1:8080/mall
16 not match
17 agent ==> AndXroid
18 request ==> GET

```

### 4.测试不区分大小写~\*



```
1 [root@web-7 ~]# curl -A "Chrome" 127.0.0.1:8080/mall
2 agent is Chrome
3
4 [root@web-7 ~]# curl -A "chrome" 127.0.0.1:8080/mall
5 agent is Chrome
6
7 [root@web-7 ~]# curl -A "ChRoMe" 127.0.0.1:8080/mall
8 agent is Chrome
9
10 [root@web-7 ~]# curl -A "AaChRoMe" 127.0.0.1:8080/mall
11 agent is Chrome
12
13 [root@web-7 ~]# curl -A "AaChRoMeDd" 127.0.0.1:8080/mall
14 agent is Chrome
```

## 3.return指令

### 3.1 官方文档

```
1 http://nginx.org/en/docs/http/nginx_http_rewrite_module.html#return
```

### 3.2 指令说明

1. return的作用是当匹配上之后直接返回响应状态码或者重写URL或者返回状态码的同时显示文本。
2. return后的指令不会在执行。

### 3.3 指令语法

```
1 reutrn可以作用于 server, location, if
2 return code [text];
3 return code URL;
4 return URL;
```

### 3.4 实验配置

```
1 server {
2     listen      8090;
3     server_name www.oldzhang.com;
4     root        /usr/share/nginx/html/www;
5
6     location / {
7         echo "this is index";
8     }
9
10    location /mall {
11        #客户端类型完全匹配iphone
12        if ($http_user_agent = iphone) {
13            return 200 "agent is iphone \n";
14        }
15
16        #客户端类型区分大小写匹配
17        if ($http_user_agent ~ Android) {
18            return 200 "agent is Android \n";
```

```

19     }
20
21     #客户端类型不区分大小写匹配
22     if ($http_user_agent ~* Chrome) {
23         return 200 "agent is Chrome \n";
24     }
25
26     #如果是IE浏览器，直接301跳转到/dog
27     if ($http_user_agent ~* IE){
28         return 301 http://www.oldzhang.com:8090/dog;
29     }
30
31     #都没匹配上则跳转到首页
32     return 301 http://www.oldzhang.com:8090/;
33 }
34
35 location /dog {
36     return 444 "wang wang! \n";
37 }
38 }

```

## 3.5 测试方法

### 1.不指定客户端跳转到首页

```

1 [root@web-7 ~]# curl -L http://127.0.0.1:8090/mall
2 this is index

```

### 2.指定客户端跳转

```

1 [root@web-7 ~]# curl -L -A "iphone" http://127.0.0.1:8090/mall
2 agent is iphone
3
4 [root@web-7 ~]# curl -L -A "Chrome" http://127.0.0.1:8090/mall
5 agent is Chrome

```

### 3.指定客户端为IE，跳转到另一个页面

```

1 [root@web-7 ~]# curl -L -A "IE" http://127.0.0.1:8090/mall
2 wang wang!

```

## 4.set指令

### 4.1 官方文档

```

1 https://nginx.org/en/docs/http/nginx_http_rewrite_module.html#set

```

### 4.2 指令说明

```

1 为指定变量设置一个值。该值可以包含文本，变量及其组合

```

## 4.3 指令语法

- 1 set可以作用于 server, location, if
- 2 set 变量名 变量值;

## 4.4 实验配置

```
1 server {
2     listen      8000;
3     server_name  www.oldzhang.com;
4     root        /usr/share/nginx/html/www;
5     set $nb_agent $http_user_agent;
6
7     location /{
8         if ($nb_agent ~* IE){
9             return 301 http://www.oldzhang.com:8090/dog;
10        }
11        return 200 "this is index \n";
12    }
13
14    location /dog {
15        return 444 "wang wang! \n";
16    }
17 }
```

## 4.5 测试方法

1.默认访问会跳转到首页

```
1 [root@web-7 ~]# curl 127.0.0.1:8000/
2 this is index
```

2.指定客户端类型为IE会跳转到dog

```
1 [root@web-7 ~]# curl -L -A "IE" 127.0.0.1:8000/
2 wang wang!
```

## 5.break指令

### 5.1 官方文档

```
1
```

### 5.2 指令说明

- 1 终止break所处的作用域的配置，执行完break后，其所在的区域后面的ngx\_http\_rewrite\_module模块的指令不在执行。
- 2 注意，只是不执行ngx\_http\_rewrite\_module模块的指令，其他指令不受影响

## 5.3 指令语法

```
1 break;  
2 可以作用于: server, location, if
```

## 5.4 实验配置

```
1 cat > /opt/nginx/conf.d/www.conf << 'EOF'  
2 server {  
3     listen      80;  
4     server_name www.oldboyedu.com;  
5     location / {  
6         set $name oldboyedu;  
7         echo $name;  
8         break;  
9         set $name szoldboy;  
10        echo $name;  
11    }  
12 }  
13 EOF
```

## 5.5 实验结果

```
1 [root@web-7 ~]# curl www.oldboyedu.com  
2 oldboyedu  
3 oldboyedu
```

## 6. rewrite指令--重点掌握

### 6.1 官方文档

```
1 https://nginx.org/en/docs/http/ngx\_http\_rewrite\_module.html#rewrite
```

### 6.2 指令说明

1. rewrite指令可以基于用户请求的URI通过正则表达式的匹配来进行改写。
2. rewrite指令可以存在多条，并且按照次序依次执行
3. rewrite指令可以根据flag标志符对指令进一步处理
4. 如果替换字符串以“http://”、“https://”或“\$scheme”开头，则处理将停止，并将重定向返回到客户端。

### 6.3 指令语法

```
1 rewrite regex replacement [flag];  
2 rewrite 匹配规则 重写指令 [标志符];  
3  
4 rewrite指令可以作用于server, location, if
```

## 6.4 flag标识符说明

flag作用说明：

1	<b>redirect</b>	返回302临时重定向，浏览器地址会显示跳转后的URL地址，浏览器不会缓存DNS记录
2	<b>permanent</b>	返回301临时重定向，浏览器地址会显示跳转后的URL地址，浏览器会缓存DNS记录
3		
4	<b>last</b>	本条规则匹配完成后，继续向下匹配新的location URI规则
5	<b>break</b>	本条规则匹配完成即终止，不再匹配后面的任何规则

使用场景区分：

1. 使用**redirect**和**permanent**参数，浏览器会显示跳转后的URL，服务端将改写后的URL返回给客户端，由客户端重新发起新的请求。
2. 使用**last**和**break**，浏览器显示的还是原来的URL，由服务器内部完成跳转。

last和break区别：

1. **last**标记在本条规则匹配完成后，对其所在的**server**标签重新发起修改后的URL请求，再次匹配**location**。
2. **break**标记则会在本条规则匹配完成后，终止匹配，不会在匹配后面的规则。

## 6.4 redirect和permanent实验

### 6.4.1 permanent 301 永久跳转 实验

permanent 301跳转需求：

1. 用户访问 `www.oldboy.com` 跳转到 `blog.oldboy.com`

permanent 301配置：

```
1 cat > /opt/nginx/conf.d/www.conf << 'EOF'
2 server {
3     listen      80;
4     server_name  www.oldboyedu.com;
5     location / {
6         root     /code/www;
7         index    index.html;
8         rewrite  / http://blog.oldboyedu.com permanent;
9     }
10 }
11 EOF
12
13 cat > /opt/nginx/conf.d/blog.conf << 'EOF'
14 server {
15     listen      80;
16     server_name  blog.oldboyedu.com;
17     location / {
18         root     /code/blog;
19         index    index.html;
20     }
21 }
```

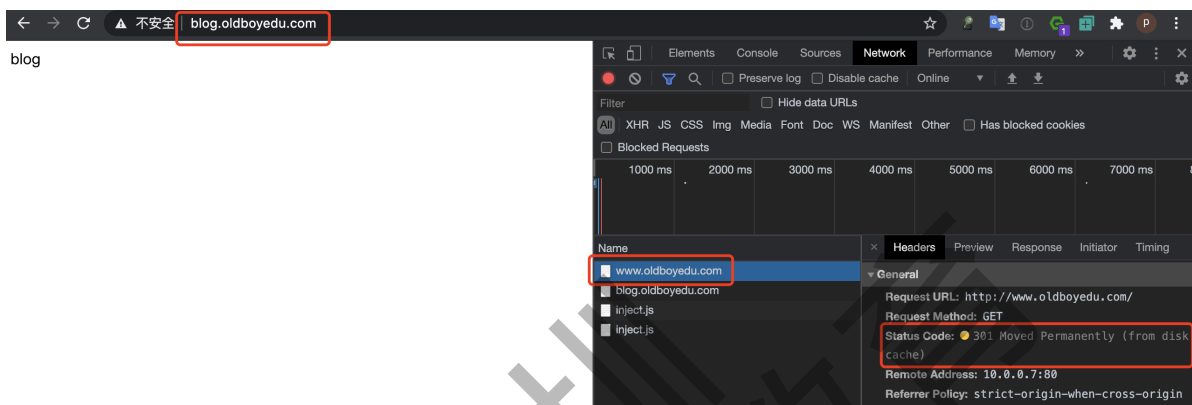
```

22 EOF
23
24 mkdir /code/{www,blog} -p
25 echo www > /code/www/index.html
26 echo blog > /code/blog/index.html
27
28 nginx -t
29 nginx -s reload

```

permanent 301测试:

- 1 浏览输入 `www.oldboyedu.com` 后会跳转到 `blog.oldboyedu.com`
- 2 headers头部信息显示from disk cache



#### 6.4.2 redirect 302 临时跳转 实验

redirect 302跳转需求:

- 1 用户访问`www.o1dboy.com` 跳转到 `b1og.o1dboy.com`

redirect 302配置:

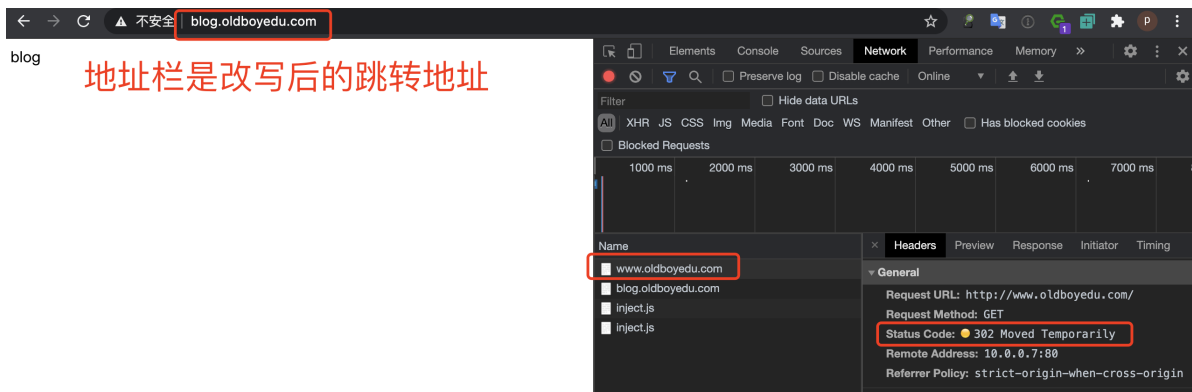
```

1 cat > /opt/nginx/conf.d/www.conf << 'EOF'
2 server {
3     listen      80;
4     server_name www.o1dboyedu.com;
5     location / {
6         root    /code/www;
7         index  index.html;
8         rewrite / http://b1og.o1dboyedu.com redirect;
9     }
10 }
11 EOF

```

redirect 302测试:

- 1 浏览输入 `www.oldboyedu.com` 后会跳转到 `blog.oldboyedu.com`
- 2 headers头部信息没有 from disk cache , 因为302跳转浏览器不缓存DNS记录



地址栏是改写后的跳转地址

## 6.5 break和last实验

### 6.5.1 last实验

last 实验需求:

- 1 访问AAA 跳转到BBB 然后再跳转到CCC

last 配置文件:

```
1 cat > /opt/nginx/conf.d/www.conf << 'EOF'
2 server {
3     listen      80;
4     server_name www.oldboyedu.com;
5     location /CCC {
6         return 200 "CCCC \n";
7     }
8
9     location /BBB {
10        rewrite ^/BBB/(.*)$ /CCC/$1 last;
11    }
12
13    location /AAA {
14        rewrite ^/AAA/(.*)$ /BBB/$1 last;
15    }
16 }
17 EOF
```

last 测试

```
1 [root@web-7 ~]# curl -L http://www.oldboyedu.com/AAA/index.html
2 CCCC
```

### 6.5.2 break实验

break 实验需求:

- 1 访问AAA后直接访问跳转后的资源, 不再匹配BBB和CCC的location

break 配置文件:

```
1 cat > /opt/nginx/conf.d/www.conf << 'EOF'
2 server {
```

```

3      listen      80;
4      server_name www.oldboyedu.com;
5      location /CCC {
6          return 200 "CCCC \n";
7      }
8
9      location /BBB {
10         rewrite ^/BBB/(.*)$ /CCC/$1 last;
11     }
12
13     location /AAA {
14         root /code/www;
15         index index.html;
16         rewrite ^/AAA/(.*)$ /BBB/$1 break;
17     }
18 }
19 EOF

```

创建测试文件:

```

1 [root@web-7 ~]# mkdir /code/www/AAA
2 [root@web-7 ~]# mkdir /code/www/BBB
3 [root@web-7 ~]# mkdir /code/www/CCC
4 [root@web-7 ~]# echo AAA > /code/www/AAA/index.html
5 [root@web-7 ~]# echo BBB > /code/www/BBB/index.html
6 [root@web-7 ~]# echo CCC > /code/www/CCC/index.html
7 [root@web-7 ~]# tree /code/www/
8 /code/www/
9 |— AAA
10 |   └─ index.html
11 |— BBB
12 |   └─ index.html
13 |— CCC
14 |   └─ index.html
15 └─ index.html

```

测试访问效果:

```

1 [root@web-7 ~]# curl -L http://www.oldboyedu.com/AAA/index.html
2 BBB

```

## 6.6 生产跳转场景

### 6.6.1 URL跳转

需求:

```

1 用户访问 www.oldboyedu.com/blog 跳转到 blog.oldboyedu.com/

```

实验配置:

```

1 cat > /opt/nginx/conf.d/www.conf << 'EOF'
2 server {
3     listen      80;
4     server_name www.oldboyedu.com;

```



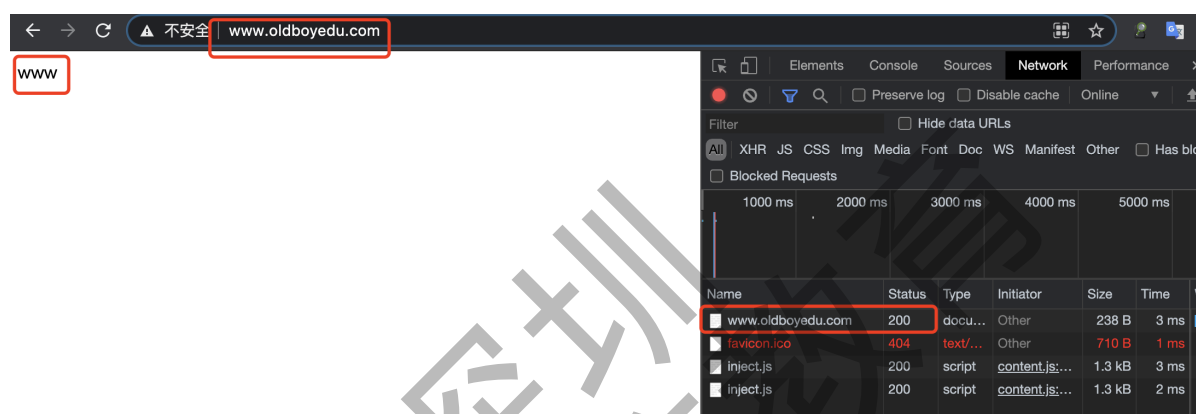
```

5     location / {
6         root /code/www;
7         index index.html;
8     }
9
10    location /blog/ {
11        rewrite ^/blog/(.*)$ http://blog.oldboyedu.com/$1 redirect;
12    }
13 }
14 EOF
15
16 nginx -t
17 nginx -s reload

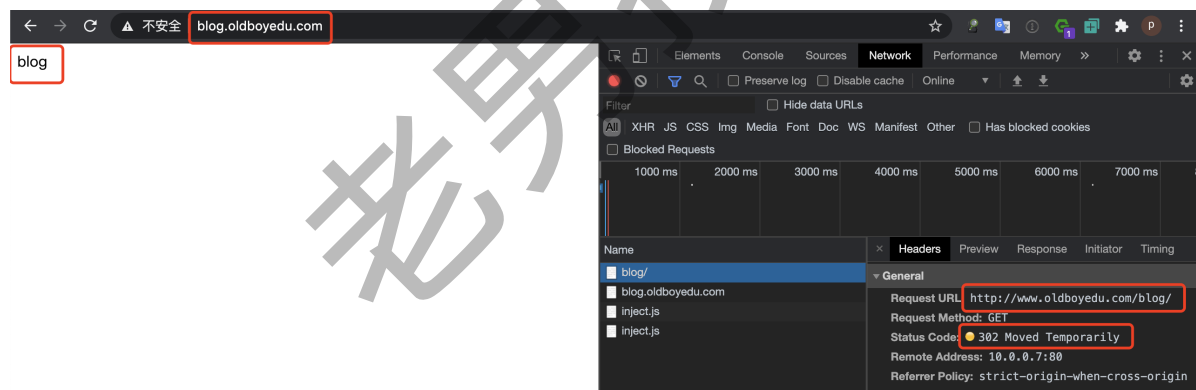
```

跳转结果:

正常访问由 location 提供服务



访问blog会被跳转到blog.oldboyedu.com



## 6.6.2 http跳转https

需求:

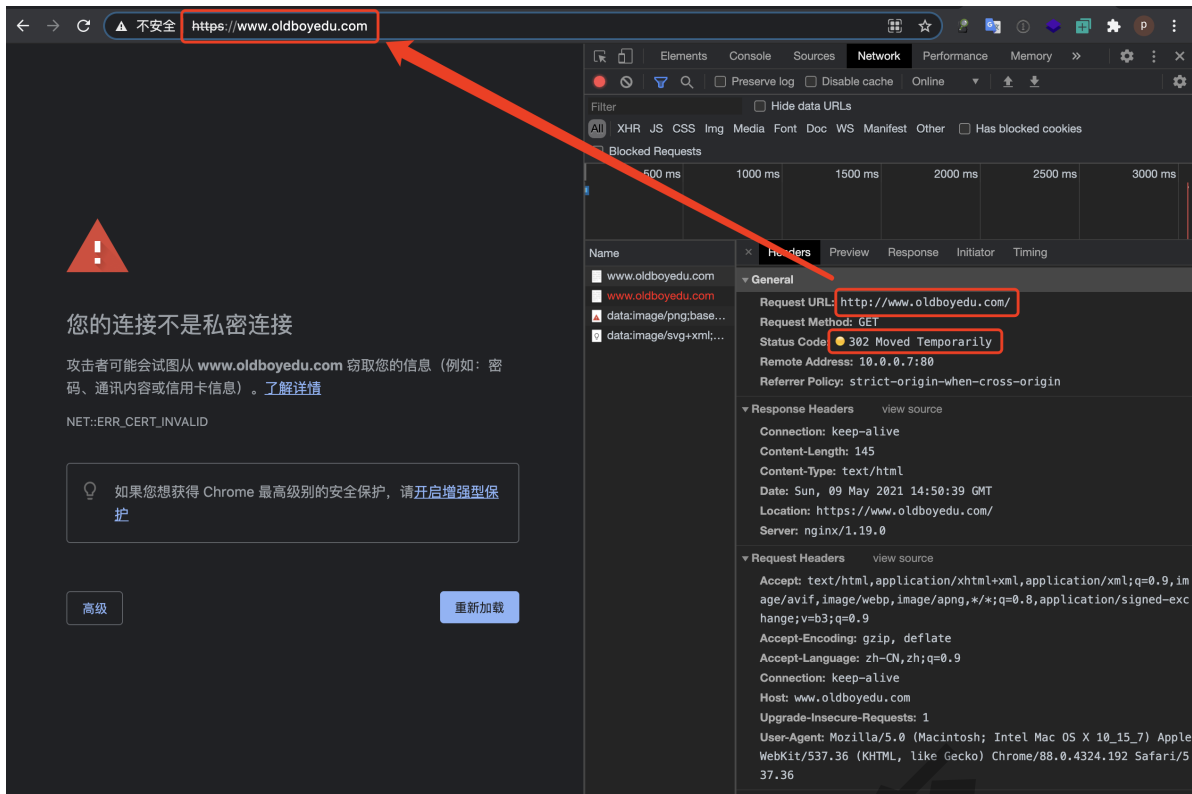
- 1 用户访问 `http://www.oldboyedu.com`
- 2 跳转到 `https://www.oldboyedu.com`

配置:

- 1 #生成证书并拷贝到指定位置
- 2 `mkdir /opt/nginx/ssl_key`
- 3 `cd /opt/nginx/ssl_key`

```
4 openssl genrsa -idea -out server.key 2048
5 输入密码:
6
7 openssl req -days 36500 -x509 -sha256 -nodes -newkey rsa:2048 -keyout
  server.key -out server.crt
8 -----
9 CN
10 SZ
11 SZ
12 linux6
13 SA
14 linux6
15 linux6@qq.com
16 -----
17
18 #检查证书文件
19 [root@web-7 ~]# ll /opt/nginx/ssl_key/
20 总用量 8
21 -rw-r--r-- 1 root root 1367 5月  9 22:47 server.crt
22 -rw-r--r-- 1 root root 1708 5月  9 22:47 server.key
23
24 #创建配置文件
25 cat > /opt/nginx/conf.d/www.conf << 'EOF'
26 server {
27     listen 80;
28     server_name www.oldboyedu.com;
29     rewrite ^(.*) https://$server_name$1 redirect;
30 }
31
32 server {
33     listen 443 ssl;
34     server_name www.oldboyedu.com;
35     ssl_certificate /opt/nginx/ssl_key/server.crt;
36     ssl_certificate_key /opt/nginx/ssl_key/server.key;
37     location / {
38         root /code/www;
39         index index.html;
40     }
41 }
42 EOF
43
44 #测试并重启
45 nginx -t
46 nginx -s stop
47 nginx
```

访问测试:



### 6.6.3 URL拼接跳转

需求:

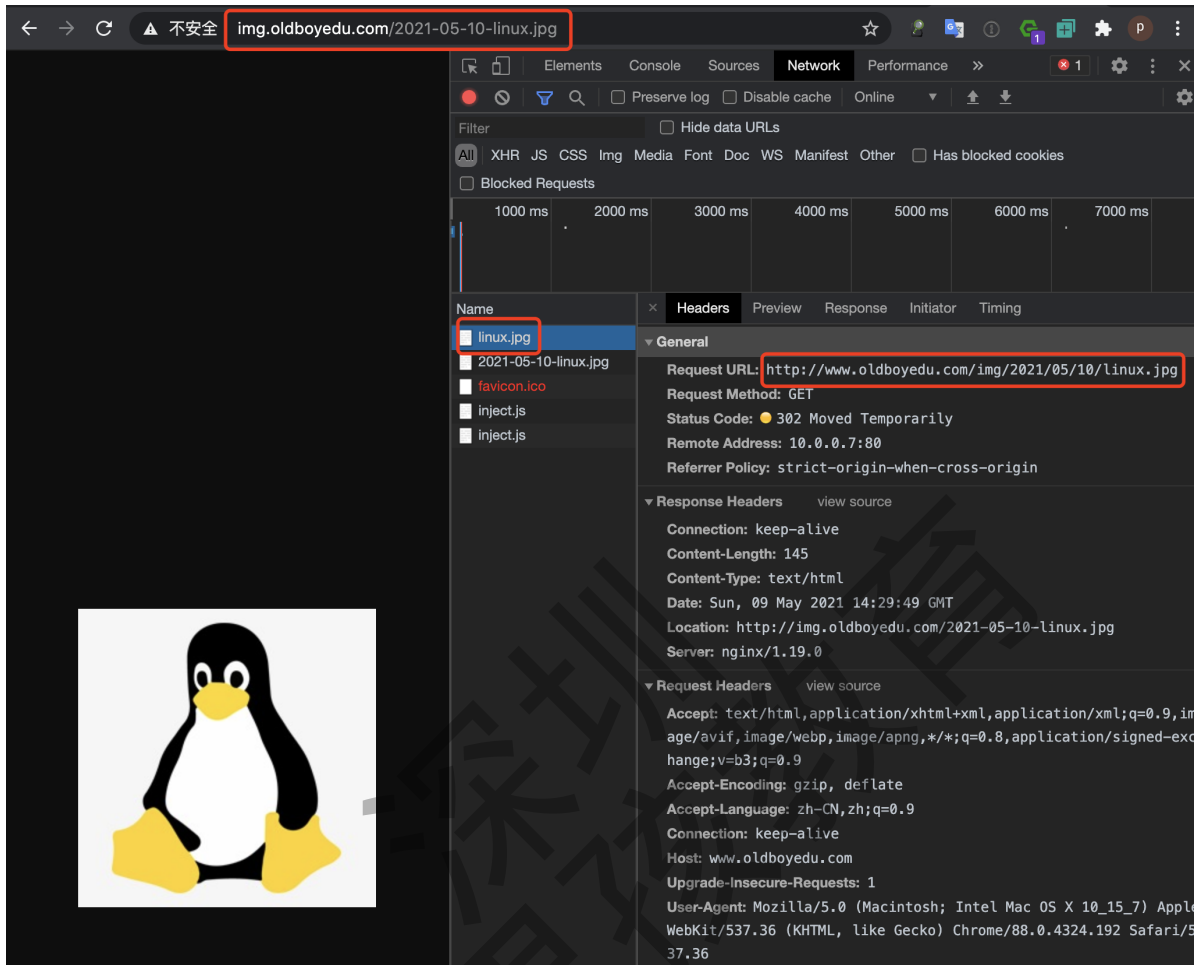
- 1 用户访问 `www.oldboyedu.com/img/2021/05/10/linux.jpg`
- 2 跳转到 `img.oldboyedu.com/2021-05-10-linux.jpg`

配置:

```
1 cat > /opt/nginx/conf.d/www.conf << 'EOF'
2 server {
3     listen      80;
4     server_name www.oldboyedu.com;
5     location / {
6         root /code/www;
7         index index.html;
8     }
9
10    location /img/ {
11        rewrite ^/img/(\d+)/(\d+)/(\d+)/(.*)\.jpg$
12        http://img.oldboyedu.com/$1-$2-$3-$4.jpg redirect;
13    }
14 }
15 EOF
16
17 cat > /opt/nginx/conf.d/img.conf << 'EOF'
18 server {
19     listen      80;
20     server_name img.oldboyedu.com;
21     location / {
22         root /code/img;
23         index index.html;
24     }
25 }
```

```
25 EOF
26
27 nginx -t
28 nginx -s reload
```

测试效果:



#### 6.6.4 带参数跳转

此类跳转比较特殊，有两种需求

需求1:

- 1 访问 `www.oldboyedu.com/img/2021/05/10/linux.jpg`
- 2 跳转到 `img.oldboyedu.com/index.php?name=2021-05-10-linux.jpg`

需求2:

- 1 访问 `www.oldboyedu.com/index.php?name=2021-05-10-linux.jpg`
- 2 跳转到 `img.oldboyedu.com/img/2021/05/10/linux.jpg`

其中需求2比较难实现。

需求1配置:

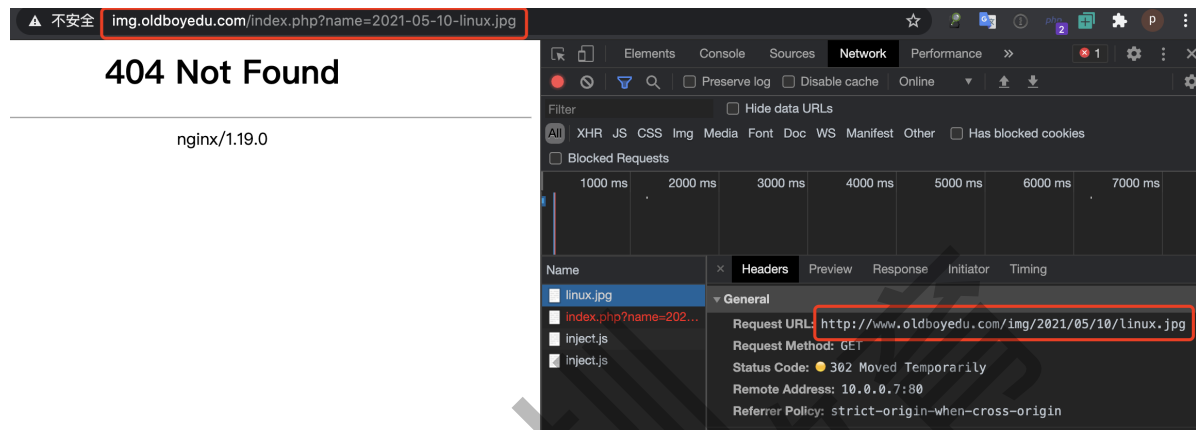
```
1 cat > /opt/nginx/conf.d/www.conf << 'EOF'
2 server {
3     listen      80;
4     server_name www.oldboyedu.com;
```

```

5     location / {
6         root /code/www;
7         index index.html;
8     }
9
10    location /img/ {
11        rewrite ^/img/(\d+)/(\d+)/(\d+)/(.*)\.jpg$
http://img.oldboyedu.com/index.php?name=$1-$2-$3-$4.jpg redirect;
12    }
13 }
14 EOF

```

测试效果:



需求2配置:

1

## 第10章 制作内网YUM仓库

### 1.为什么需要私有YUM仓库

1. 下载速度慢
2. 需要有外网
3. 有些Base源和epel源软件没有，需要单独创建下载源

### 2.需要的软件

- 1 createrepo
- 2 nginx

### 3.配置nginx索引模块

```

1 cat >/etc/nginx/conf.d/index.conf <<EOF
2 server {
3     listen      80;
4     server_name yum.mysun.com;
5     location / {
6         autoindex on;
7         autoindex_exact_size off;
8         autoindex_localtime on;

```

```
9         autoindex_format html;  
10        charset utf-8,gbk;  
11        root    /data/yum;  
12        index   index.html index.htm;  
13    }  
14 }  
15 EOF
```

## 4.安装createrepo

```
1 yum install createrepo -y
```

## 5.准备软件仓库目录并下载需要的软件

```
1 yum install --downloadonly --downloadaddir=/data/yum nginx screen vim tree -y
```

## 6.生成yum元数据

```
1 createrepo /data/yum
```

## 7.客户端生成本地源

```
1 cat >/etc/yum.repos.d/local.repo <<EOF  
2 [local]  
3 name=local  
4 enable=1  
5 gpgcheck=0  
6 baseurl=http://10.0.0.61
```

## 8.客户端测试安装

```
1 yum makecache  
2 yum search nginx  
3 yum install nginx
```

## 9.更新软件包的操作步骤:

第一种方法：真实下载

1.打开yum缓存

```
1 [root@web01 /data/yum]# grep "keepcache" /etc/yum.conf  
2 keepcache=1
```

2.清空原来的缓存

```
1 yum clean all
```

3.下载软件

```
1 yum remove php-mysql-5.4 php php-fpm php-common
2 rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-
  7.noarch.rpm
3 rpm -Uvh https://mirror.webtatic.com/yum/el7/webtatic-release.rpm
4 yum install php71w php71w-cli php71w-common php71w-devel php71w-embedded
  php71w-gd php71w-mcrypt php71w-mbstring php71w-pdo php71w-xml php71w-fpm
  php71w-mysqlnd php71w-opcache -y
```

4.移动已经缓存下来的rpm包到yum仓库目录

```
1 find /var/cache/yum/ -type f -name "*.rpm"|xargs mv -t /data/yum/
```

5.生成新的yum元数据

```
1 createrepo --update /data/yum/
```

第二种方法：只下载不安装

```
1 yum install --downloadonly --downloadaddir=/data/yum php71w php71w-cli php71w-
  common php71w-devel php71w-embedded php71w-gd php71w-mcrypt php71w-mbstring
  php71w-pdo php71w-xml php71w-fpm php71w-mysqlnd php71w-opcache
```