第1章 目录索引

0.官方地址

```
http://nginx.org/en/docs/http/ngx_http_autoindex_module.html
```

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：

```
Syntax: autoindex on | off;
Default: autoindex off;
Context: http, server, location

# autoindex 常用参数
autoindex_exact_size off;
默认为 on， 显示出文件的确切大小，单位是 bytes。
修改为 off，显示出文件的大概大小，单位是 kB 或者 MB 或者 GB。

autoindex_localtime on;
默认为 off，显示的文件时间为 GMT 时间。
修改为 on， 显示的文件时间为文件的服务器时间。

charset utf-8,gbk;
默认中文目录乱码，添加上解决乱码
```

3.配置文件

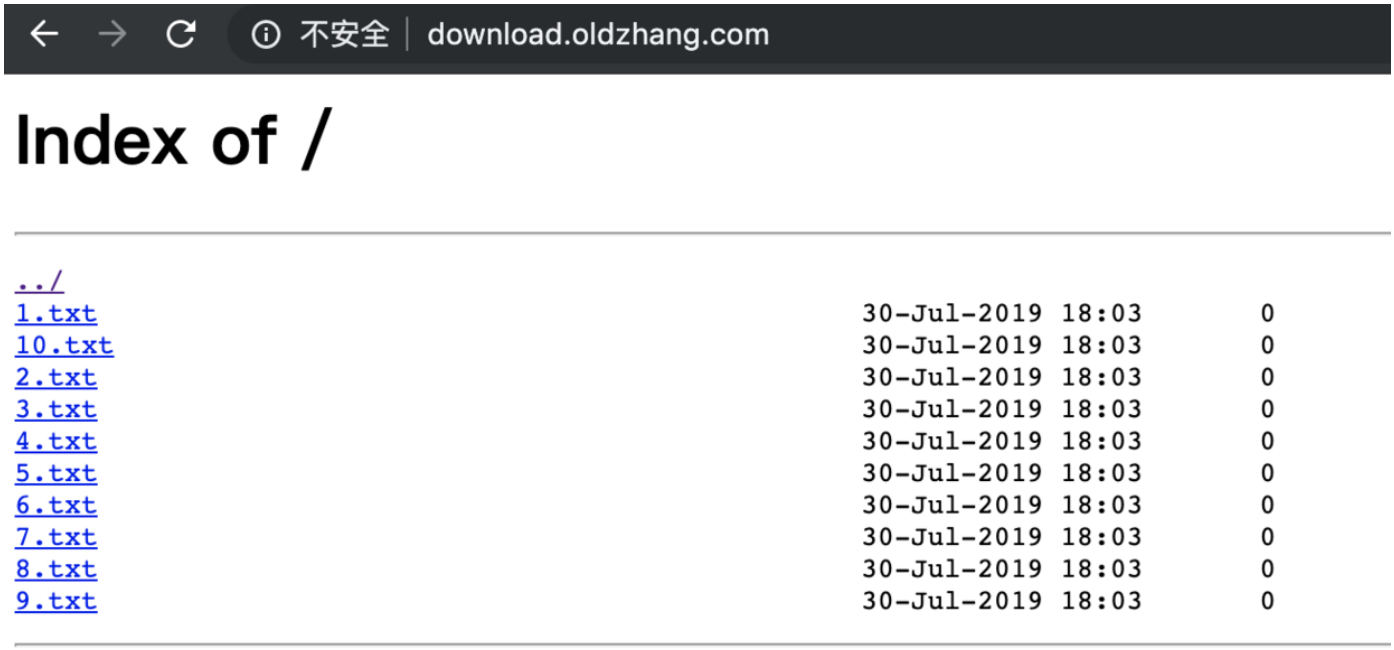
```
cat >/etc/nginx/conf.d/download.conf <<EOF
server {
    listen 80;
    server_name download.oldzhang.com;
    location / {
        root /usr/share/nginx/html/download;
        charset utf-8,gbk;
        autoindex on;
        autoindex_localtime on;
        autoindex_exact_size off;
    }
}
EOF
```

4.创建测试数据

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

5.访问页面

download.oldzhang.com



第2章 状态监控

0.官方地址

```
http://nginx.org/en/docs/http/ngx_http_stub_status_module.html
```

1.状态字段解释

```
Active connections # 当前活动的连接数
accepts # 当前的总连接数 TCP
handled # 成功的连接数 TCP
requests # 总的 http 请求数
Reading # 请求
Writing # 响应
Waiting # 等待的请求数，开启了 keepalive
# 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
keepalive_timeout 0; # 类似于关闭长连接
keepalive_timeout 65; # 65s 没有活动则断开连接
```

2.配置文件

```
cat > /etc/nginx/conf.d/status.conf <<EOF
server {
   listen 80;
   server_name  status.oldzhang.com;
   stub_status on;
   access_log off;
}
EOF
```

3.访问测试

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

第3章 基于IP的访问控制

0.官方地址

```
http://nginx.org/en/docs/http/ngx_http_access_module.html
```

1.配置语法

```
#允许配置语法
Syntax: allow address | CIDR | unix: | all;
Default: —
Context: http, server, location, limit_except

#拒绝配置语法
Syntax: deny address | CIDR | unix: | all;
Default: —
Context: http, server, location, limit_except\
```

2.配置案例1: 拒绝windows访问www域名

```
cat > /etc/nginx/conf.d/01-www.conf <<EOF
server   {
    listen       80;
    server_name  www.oldzhang.com;
    location / {
        root   /usr/share/nginx/html/www;
        index  index.html index.htm;
        deny 10.0.0.1;
        allow all;
    }
}
EOF
```

3.windows测试访问

www.oldboy.com

4.使用虚拟机测试访问

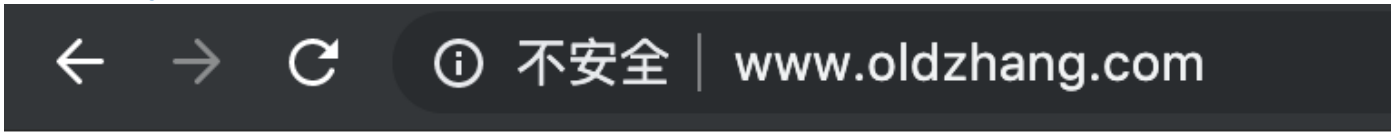
```
[root@web01 ~]# curl www.oldzhang.com
www
```

5.配置案例2: 只允许windows访问，其他全部拒绝

```
cat >/etc/nginx/conf.d/01-www.conf<<EOF
server   {
    listen       80;
    server_name  www.oldzhang.com;
    location / {
        root   /usr/share/nginx/html/www;
        index  index.html index.htm;
        allow 10.0.0.1;
        deny all;
    }
}
EOF
```

6.windows测试访问

www.oldboy.com



www

7.使用虚拟机测试访问

```
[root@web01 ~]# curl www.oldzhang.com
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.16.0</center>
</body>
</html>
```

第4章 基于用户认证的访问控制

0.官方配置

```
http://nginx.org/en/docs/http/ngx_http_auth_basic_module.html
```

1.配置语法

```
#访问提示字符串
Syntax: auth_basic string| off;
Default: auth_basic off;
Context: http, server, location, limit_except

#账户密码文件
Syntax: auth_basic_user_file file;
Default: -
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

```
[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

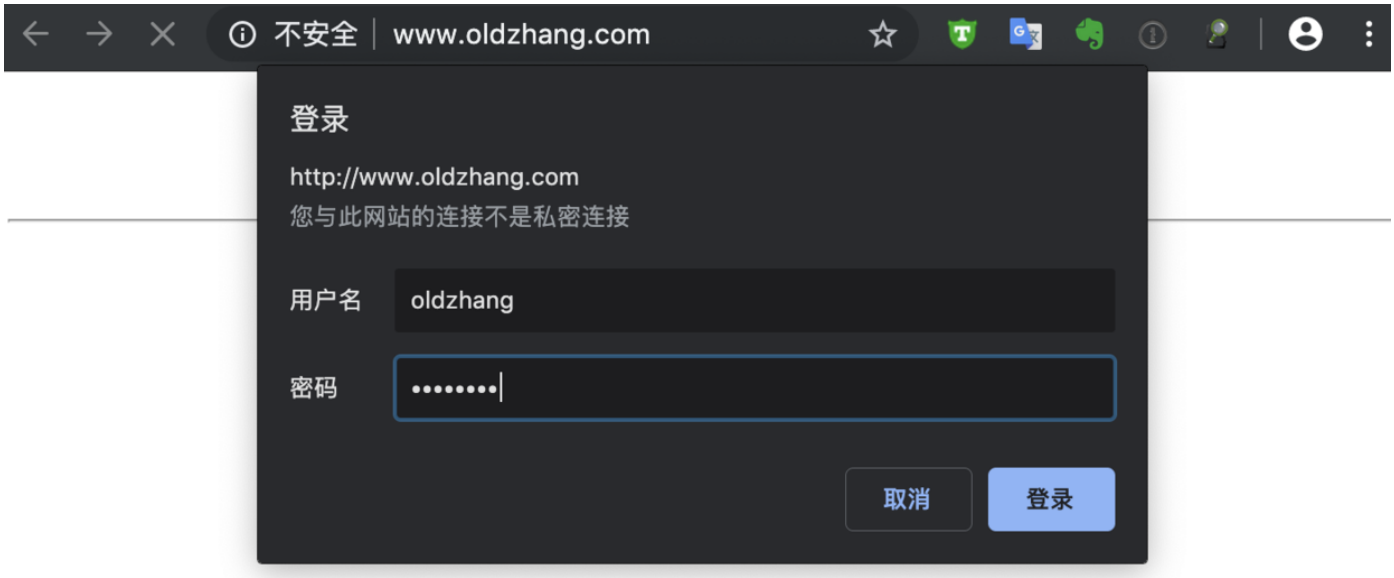
```
[root@web01 ~]# htpasswd -b -c /etc/nginx/auth_conf oldzhang oldzhang
Adding password for user oldzhang
```

nginx 配置调用

```
cat >/etc/nginx/conf.d/01-www.conf <EOF
server   {
    listen       80;
    server_name  www.oldzhang.com;
    location / {
        auth_basic "Auth access Blog Input your Passwd!";
        auth_basic_user_file auth_conf;
        root   /usr/share/nginx/html/www;
        index  index.html index.htm;
    }
}
EOF
```

3.访问测试

www.oldzhang.com



第5章 请求限制

0.官方地址

```
http://nginx.org/en/docs/http/ngx_http_limit_req_module.html
```

1.配置语法

```
#模块名 ngx_http_limit_req_module
Syntax: limit_req_zone key zone=name:size rate=rate;
Default: —
Context: http

#引用限速模块
Syntax: limit_conn zone number [burst=number] [nodelay];
Default: —
Context: http, server, location
```

参数解释：

```
定义一条规则
limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;

limit_req_zone         #引用限速模块
$binary_remote_addr    #判定条件，每个请求的IP
zone=one:10m           #定义一个zone名称
rate=1r/s;             #限制速度，1秒1次

引用一条限速规则
limit_req zone=two burst=5 nodelay;

limit_req            #引用限速规则语法
zone=one             #引用哪一条规则
burst=5              #令牌桶，允许排队的数量
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

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

3.访问测试

ab压测

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

for循环

```
一秒1次
for i in {1..10000};do curl -I 10.0.0.7;sleep 1;done

一秒2次
for i in {1..10000};do curl -I 10.0.0.7;sleep 0.5;done

一秒5次
for i in {1..10000};do curl -I 10.0.0.7;sleep 0.2;done
```

4.查看访问日志

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

5.查看错误日志

```
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 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

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

1.location语法介绍

```
Syntax: location [ = | ~ | ~* | ^~ ] uri { ... }
location @name { ... }
Default:  —
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

```
cat >/etc/nginx/conf.d/01-www.conf <<EOF
server {
    listen       80;
    server_name  www.oldzhang.com;
    root   /usr/share/nginx/html/www;
    location / {
```

```
        return 200  "location / \n";
    }
    location = / {
        return 200 "location = \n";
    }

    location /documents/ {
        return 200 "location /documents/ \n";
    }
    location ^~ /images/ {
        return 200 "location ^~ /images/ \n";

    }
    location ~* \.(gif|jpg|jpeg)$ {
        return 200 "location ~* \.(gif|jpg|jpeg) \n";
    }
    access_log off;
}
EOF
```

4.测试location匹配规则

```
#精确匹配=/
[root@web01 ~]# curl www.oldzhang.com
location =

#没有满足的请求，所以匹配了/
[root@web01 ~]# curl www.oldzhang.com/oldzhang.html
location /

#匹配了/documents
[root@web01 ~]# curl www.oldzhang.com/documents/oldboy.html
location /documents/

#没有满足的条件，匹配/
[root@web01 ~]# curl www.oldzhang.com/oldboy/documents/oldboy.html
location /

#正则匹配了文件名
[root@web01 ~]# curl www.oldzhang.com/oldboy.jpg
location ~* \.(gif|jpg|jpeg)

#~*匹配正则不区分大小写优先于/documents
[root@web01 ~]# curl www.oldzhang.com/documents/oldboy.jpg
location ~* \.(gif|jpg|jpeg)

#^~优先匹配于~*
[root@web01 ~]# curl www.oldzhang.com/images/oldboy.jpg
```

```
location ^~ /images/
```

第7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

```
createrepo
nginx
```

3.配置nginx索引模块

```
cat >/etc/nginx/conf.d/index.conf <<EOF
server {
    listen       80;
    server_name  yum.mysun.com;
    location / {
        autoindex on;
        autoindex_exact_size off;
        autoindex_localtime on;
        autoindex_format html;
        charset utf-8,gbk;
        root   /data/yum;
        index  index.html index.htm;
    }
}
EOF
```

4.安装createrepo

```
yum install createrepo -y
```

5.准备软件仓库目录并下载需要的软件

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

6.生成yum元数据

```
createrepo /data/yum
```

7.客户端生成本地源

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

8.客户端测试安装

```
yum makecache
yum search nginx
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

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

2.清空原来的缓存

```
yum clean all
```

3.下载软件

```
yum remove php-mysql-5.4 php php-fpm php-common
rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
rpm -Uvh https://mirror.webtatic.com/yum/el7/webtatic-release.rpm
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

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

5.生成新的yum元数据

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

第二种方法：只下载不安装

```
yum install --downloadonly --downloaddir=/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
```