****

**老男孩教育教学核心思想6重：重目标、重思路、重方法、重实践、重习惯、重总结**

**学无止境，老男孩教育成就你人生的起点！**

目 录

[第1章 nginx 1](#_Toc514245704)

[1.1 web网站服务介绍 1](#_Toc514245705)

[1.1.1 常用来提供静态Web服务的软件 1](#_Toc514245706)

[1.1.2 1常用来提供动态服务的软件 1](#_Toc514245707)

[第2章 第2章 nginx网站服务软件 1](#_Toc514245708)

[2.1 2.1 Nginx是什么 1](#_Toc514245709)

[2.2 2.2 Nginx的特色及优点 2](#_Toc514245710)

[2.2.1 2.2.1 Nginx的重要特性 2](#_Toc514245711)

[2.2.2 2.2.2 Nginx HTTP服务器的特色及优点（重要） 3](#_Toc514245712)

[2.3 2.3 Nginx软件的主要企业功能应用 3](#_Toc514245713)

[2.3.1 2.3.1 作为Web服务软件 3](#_Toc514245714)

[2.3.2 2.3.2 反向代理或负载均衡服务 3](#_Toc514245715)

[2.3.3 2.3.3 前端业务数据缓存服务 3](#_Toc514245716)

[2.4 2.4 为什么Nginx总体性能比Apache高 4](#_Toc514245717)

[第3章 第3章 部署nginx软件（编译安装） 4](#_Toc514245718)

[3.1 3.1 第一个里程碑：进行nginx软件下载 4](#_Toc514245719)

[3.2 3.2 第二个里程碑：安装依赖软件 5](#_Toc514245720)

[3.3 3.3 第三个里程碑：解压软件，创建虚拟用户，进行编译安装（三步曲） 5](#_Toc514245721)

[3.3.1 3.3.1 解压软件，创建虚拟用户 5](#_Toc514245722)

[3.3.2 3.3.2 编译第一步：进行配置 6](#_Toc514245723)

[3.3.3 3.3.3 编译第二步：编译（翻译） 7](#_Toc514245724)

[3.3.4 3.3.4 编译第三步：编译安装 7](#_Toc514245725)

[3.4 3.4 第四个里程碑：创建程序目录软链接文件 8](#_Toc514245726)

[3.5 3.5 第五个里程碑：网站服务启动成功 8](#_Toc514245727)

[3.6 3.6 第六个里程碑：检查测试 8](#_Toc514245728)

[第4章 第4章 部署nginx软件可能会遇到的问题 9](#_Toc514245729)

[第5章 第5章 nginx软件程序目录结构 10](#_Toc514245730)

[5.1 5.1 nginx软件程序目录结构 10](#_Toc514245731)

[5.2 5.2 配置文件目录说明 11](#_Toc514245732)

[5.3 5.3 精简nginx配置文件信息 12](#_Toc514245733)

[5.4 5.4 nginx服务命令参数说明 15](#_Toc514245734)

[5.5 5.5 实践配置编写配置文件 16](#_Toc514245735)

# nginx

## ****web网站服务介绍****

### ****常用来提供静态web服务的软件****

静态web就是直接访问的web服务器上面的内容，通过访问web服务器里面的文件实现的，静态web服务里面的内容是固定不变的，不会连接数据库的。

Apache：这是中小型Web服务的主流，Web服务器中的老大哥。

Nginx：大型网站Web服务的主流，曾经Web服务器中的初生牛犊，现已长大。Nginx的分支Tengine（http://tengine.taobao.org/）目前也在飞速发展。

Lighttpd：这是一个不温不火的优秀Web软件，社区不活跃，静态解析效率很高。在Nginx流行前，它是大并发静态业务的首选，国内百度贴吧、豆瓣等众多网站都有Lighttpd奋斗的身影。

### ****常用来提供动态web服务的软件****

动态web就是通过访问服务器上面的内容然后来调取数据库里面的内容。不同的人，可能访问的内容也不一样。

PHP（FastCGI）：大中小型网站都会使用，动态网页语言PHP程序的解析容器。它可配合Apache解析动态程序，不过，这里的PHP不是FastCGI守护进程模式，而是mod\_php5.so（module）。也可配合Nginx解析动态程序，此时的PHP常用FastCGI守护进程模式提供服务。(.php)

Tomcat：中小企业动态Web服务主流，互联网Java容器主流（如jsp、do）。（中小型规模）

Resin：大型动态Web服务主流，互联网Java容器主流（如jsp、do）。（大型公司）

IIS（Internet information services）：微软windows下的Web服务软件（如asp、aspx）

## ****nginx网站服务软件****

### ****Nginx是什么****

Nginx是一款类似于Apache的web服务软件， Nginx（“engine x”）是一个开源的(c语言)，支持高性能、高并发的WWW服务器和代理服务软件。

Nginx因具有高并发（特别是静态资源）、占用系统资源少等特性，且功能丰富逐渐流行起来。

### ****Nginx的特色及优点****

#### ****Nginx的重要特性****

可针对静态资源高速高并发访问及缓存

可使用反向代理加速，并且可进行数据缓存

具有简单负载均衡、节点健康检查和容错功能

支持远程FastCGI服务的缓存加速

支持FastCGI、Uwsgi、SCGI、Memcached Servers的加速和缓存

支持SSL、TLS、SNI。

具有模块化的架构：过滤器包括gzip压缩、ranges支持、chunked响应、XSLT、SSI及图像缩放等功能。在SSI过滤器中，一个包含多个SSI的页面，如果经由FastCGI或反向代理处理，可被并行处理。

#### ****Nginx HTTP服务器的特色及优点（重要-面试需要）****

支持高并发：能支持几万并发连接（特别是静态小文件业务环境）

资源消耗少：在3万并发连接下，开启10个Nginx线程消耗的内存不到200MB（ab webbenh）

可以做HTTP反向代理及加速缓存、即负载均衡功能，内置对RS节点服务器健康检查功能，这相当于专业的Haproxy软件或LVS的功能。

具备Squid等专业缓存软件等的缓存功能。

支持异步网络I/O事件模型epoll（Linux 2.6+）（apache select）

### ****Nginx软件的主要企业功能应用****

#### ****作为Web服务软件****

Nginx是一个支持高性能、高并发的Web服务软件，它具有很多优秀的特性，作为Web服务器，与Apache相比，Nginx能够支持更多的并发连接访问，但占用的资源却更少，效率更高，在功能上也强大了很多，几乎不逊于Apache。

#### ****反向代理或负载均衡服务****

在反向代理或负载均衡服务方面，Nginx可以作为Web服务，PHP等动态服务及Memcached缓存的代理服务器，它具有类似专业代理软件（如Haproxy）的功能，同时也是一个优秀的邮件代理服务软件（最早开发这个产品的目的之一就是作为邮件代理服务）。Nginx的代理功能在逐渐增强。

#### ****前端业务数据缓存服务****

在web缓存服务方面，Nginx可通过自身的proxy\_cache模块实现类Squid等专业缓存软件的功能。

Nginx的这三大功能（web服务、反向代理或负载均衡服务、前端业务数据缓存服务）是国内使用Nginx的主要场景，特别是前两个。

## 2.4 ****为什么Nginx总体性能比Apache高****

Nginx使用最新的epoll（Linux 2.6内核）和Kqueue（freebsd）异步网络I/O模型，

而Apache使用的是传统的select模型，处理大量连接的读写时，Apache所采用的select网络I/O模型比较低效。（select 遍历查找信息，epoll 利用提供的信息去注册表里面去查找）

目前Linux先能够承受高并发访问的Squid、Memcached软件采用的都是epoll模型。

#### **Apache select 和Nginx epoll的技术对比**

|  |  |  |
| --- | --- | --- |
| **指标** | **select** | **epoll** |
| 性能 | 随着连接数的增加性能急剧下降。处理成千上万并发连接数，性能很差 | 随着连接数的增加，性能基本上没有下降。处理成千上万并发连接时，性能很好 |
| 连接数 | 连接数有限制，处理的最大连接数不超过1024，如果要处理的连接数超过1024个，则需要修改FD\_SETSIZE宏，并重新编译 | 连接数无限制 |
| 内在处理机制 | 线性轮询 | 回调callback |
| 开发复杂性 | 低 | 中 |

## ****如何正确选择web服务器****

建议如下：

1. 静态业务：若是高并发场景，尽量采用nginx或lighttpd,二者首选nginx
2. 动态业务：理论上采用nginx和apaceh均可，建议选择nginx,为了避免相同业务的服务软件多样化，增加额外的维护成本，动态业务可以由nginx兼做前段代理，再根据页面元素类型或者目录，转发到后端相应的服务器处理进程---首选 tomcat
3. 既有动态业务，又有静态业务采用nginx

备注说明：如果并发不是很大，有对apache很熟悉，采用apache也是可以的。

# ****部署nginx软件（编译安装）****

## nginx软件部署安装的几种方法

1. yum安装软件 ---rpm install -y nginx
2. rpm安装 -- rpm -avh ...
3. 编译安装

## nginx安装配置文件

查看配置文件的内容

/server/tools/nginx-1.14.0/configure --help

重要配置参数总结

--prefix=PATH set installation prefix

设置软件安装到的目录,并且指定的不需要创建出来

具体用法 --prefix=/application/nginx-1.14

--user=USER set non-privileged user for worker processes

对worker进程设置一个非特权的用户（no-privileged 非特权）

--group=GROUP set non-privileged group for worker processes

对 worker进程设置一个非特权的的用户组

--with-http\_ssl\_module enable ngx\_http\_ssl\_module (可以实现https功能) enable 能够

--with-http\_stub\_status\_module enable ngx\_http\_stub\_status\_module

能够监控服务的运行状态

## ****wget命令****

wget - The non-interactive network downloader.

非交互式的网络下载器

用法 wget 下载地址

## ****nginx软件部署安装****

### ****第一个里程碑：进行nginx软件程序包下载****

#### 创建存放软件包的目录

mkdir -p /server/tools

#### 在nginx官方网站去下载

cd /server/tools

wget <http://nginx.org/download/nginx-1.14.0.tar.gz>

具体操作如下：输入网址<http://nginx.org/>



只选择稳定的版本进行安装。 另外工作中安装nignx的时候要选当前时间，半年前的的稳定版本的。这样如果报错的话，会有人沟通解决。

#### 解压缩下载的压缩包

tar xf nginx-1.14.0.tar.gz

### 第二个里程碑：安装依赖软件（解决软件依赖问题）

yum install openssl-devel pcre-devel -y

说明，如果安装系统没有安装兼容程序库、开发工具的，请安装一下基础依赖包

yum install gcc gcc-c++ automake autoconf -y

-------------------------------------------------------------------------------------

解释1：实际上这个软件缺少的依赖包有 openssi openssi-devel 和 pcre pcre-devel 这四个包，但是安装了 openssi-devel就相当于安装了 openssi 和openssi-devel 安装了pcre-devel 相当于安装了 pcre 和pcre-devel

只要解决依赖的问题，就要安装 devel软件包

解释2：

openssl-devel 为了让nginx服务实现https访问的功能

pcre-devel ----兼容per语言的正则表达式

怎么知道需要解决那些依赖的软件呢？就是在编译安装的环节，会报错，然后根据报错再次安装即可。



### 第三个里程碑：创建worker进程的管理用户 www

useradd -s /sbin/nologin -M www

----------------下面代表创建成功----------------

[root@web01 tools]# grep www /etc/passwd

www:x:501:501::/home/www:/sbin/nologin

[root@web01 tools]#

可以 用 echo $? 检查是否执行成功

### 第四个里程碑：进行软件的编译安装

#### 进行软件的配置

cd /server/tools/nginx-1.14.0

./configure --prefix=/application/nginx-1.14 --user=www --group=www --with-http\_ssl\_module --with-http\_stub\_status\_module

------------------------可以用下面全路径也可以------------------------

/server/tools/nginx-1.14.0/configure --prefix=/application/nginx-1.14 --user=www --group=www --with-http\_ssl\_module ----with-http\_stub\_status\_module

可以 用 echo $? 检查是否执行成功

#### 进行软件的编译（既将各个语言编写的代码翻译成系统可以识别的二进制信息）

make

---既在命令行里面输入make命令即可 --可以 用 echo $? 检查是否执行成功

#### 进行软件的安装（将软件最终安装到系统中）

make install

----既在命令行里面输入 make install即可

可以 用 echo $? 检查是否执行成功

### 第五个里程碑：创建程序的软连接

ln -s /application/nginx-1.14/ /application/nginx

-----检查----------

[root@web01 application]# ls -l

total 4

lrwxrwxrwx 1 root root 24 May 16 11:05 nginx -> /application/nginx-1.14/

drwxr-xr-x 11 root root 4096 May 16 19:15 nginx-1.14

[root@web01 application]#

可以 用 echo $? 检查是否执行成功

为什么创建软连接

在实际生产环境中，一个程序目录部署好之后，会被其他开发程序或者脚本程序所调用，但是我们的软件会经常更新版本，所以如果一旦你把这个目录写了版本号的话，那么这个版本就会在程序里面出现，当软件更新版本之后呢，为了更好的区分，所以软件的目录也会对应的更新，这个时候程序就灭有办法调用这个对应的目录了，如果要调用的话，就需要修改程序代码了，那么就会很麻烦。所以这个时候创建一个软连接文件，那么就让开发一直访问这个软连接就可以了，就算软件更新了，只用更新源文件即可。

代码程序1：nginx\_info1="/application/nginx/"

代码程序2：nginx\_info2="/application/nginx/"

代码程序3: nginx\_info3="/application/nginx/"

### 第六个里程碑：启动nginx服务

root@web01 sbin]# /application/nginx/sbin/nginx --启动nginx服务

[root@web01 sbin]# ps -ef|grep nginx ----查看是否启动成功

root 15596 1 0 19:45 ? 00:00:00 nginx: master process /application/nginx/sbin/nginx

www 15597 15596 0 19:45 ? 00:00:00 nginx: worker process

root 15599 12942 0 19:45 pts/0 00:00:00 grep --color=auto nginx

[root@web01 sbin]#

### 第七个里程碑检查测试

第一种方法： curl 10.0.0.7 （相当于ping 的功能 并且比ping的功能还多。 1.检查连接外网的时候是否畅通2，如果畅通就会显示http协议请求和相应的具体信息，既http协议 index首页的具体信息）

[root@web01 sbin]# curl 10.0.0.7 ----这里说明一下 web01这个时候相当于nginx服务器，那么测试的时候我们用其他主机进行测试即可。 用 其他主机作为客户端curl 10.0.0.7即可。

<!DOCTYPE html>

<html>

<head>

<title>Welcome to nginx!</title>

<style>

body {

width: 35em;

margin: 0 auto;

font-family: Tahoma, Verdana, Arial, sans-serif;

}

</style>

</head>

<body>

<h1>Welcome to nginx!</h1>

<p>If you see this page, the nginx web server is successfully installed and

working. Further configuration is required.</p>

<p>For online documentation and support please refer to

<a href="http://nginx.org/">nginx.org</a>.<br/>

Commercial support is available at

<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>

</body>

</html>

[root@web01 sbin]#

第二种方法：在浏览器里面输入主机的外网ip地址，会显示登录的界面



## ****部署nginx软件可能会遇到的问题****

### 1.****依赖包软件没有安装****

### 2.****启动Nginx时如下报错“nginx:[emerg]getpwnam(“nginx”)failed”****

**解答：这是因为没有对应的Nginx服务用户，执行useradd nginx -s /sbin/nologin -M创建Nginx用户即可。**

### ****如何查看Nginx编译时的参数？****

**解答：可采用如下命令查看：/application/nginx/sbin/nginx -V**

[root@web02 application]# /application/nginx/sbin/nginx -V

nginx version: nginx/1.12.2

built by gcc 4.4.7 20120313 (Red Hat 4.4.7-18) (GCC)

built with OpenSSL 1.0.1e-fips 11 Feb 2013

TLS SNI support enabled

configure arguments: --prefix=/application/nginx-1.12.2 --user=www --group=www --with-http\_ssl\_module --with-http\_stub\_status\_module

# nginx软件应用

## nginx软件程序编译之后的重要目录

[root@web01 nginx]# pwd

/application/nginx

[root@web01 nginx]# ll

drwxr-xr-x 2 root root 4096 May 16 10:57 conf

drwxr-xr-x 2 root root 4096 May 16 10:57 html

drwxr-xr-x 2 root root 4096 May 16 19:45 logs

drwxr-xr-x 2 root root 4096 May 16 19:12 sbin

--------------解释--------------------

### conf目录

drwxr-xr-x 2 root root 4096 May 16 10:57 conf

所有nginx相关的配置文件保存目录

[root@web01 sbin]# pwd --查看当前路径

/application/nginx/sbin

[root@web01 sbin]# cd ../conf ---返回上一级目录，并且进入conf目录

[root@web01 conf]# ll

total 68

-rw-r--r-- 1 root root 1077 May 16 10:57 fastcgi.conf

-rw-r--r-- 1 root root 1077 May 16 19:12 fastcgi.conf.default

-rw-r--r-- 1 root root 1007 May 16 10:57 fastcgi\_params

-rw-r--r-- 1 root root 1007 May 16 19:12 fastcgi\_params.default

-rw-r--r-- 1 root root 2837 May 16 19:12 koi-utf

-rw-r--r-- 1 root root 2223 May 16 19:12 koi-win

-rw-r--r-- 1 root root 5170 May 16 10:57 mime.types

-rw-r--r-- 1 root root 5170 May 16 19:12 mime.types.default

-rw-r--r-- 1 root root 2656 May 16 10:57 nginx.conf -**-nginx程序的主配置文件**

-rw-r--r-- 1 root root 2656 May 16 19:12 nginx.conf.default 默认的nginx程序的主配置文件，相当于备份文件

-rw-r--r-- 1 root root 636 May 16 10:57 scgi\_params

-rw-r--r-- 1 root root 636 May 16 19:12 scgi\_params.default

-rw-r--r-- 1 root root 664 May 16 10:57 uwsgi\_params

-rw-r--r-- 1 root root 664 May 16 19:12 uwsgi\_params.default

-rw-r--r-- 1 root root 3610 May 16 19:12 win-utf

[root@web01 conf]#

#### nginx程序主配置文件 --nginx.conf查看/精简

1.查看nginx程序的主配置文件

[root@web01 conf]# pwd

/application/nginx/conf

[root@web01 conf]# cat nginx.conf

#user nobody;

.

.

.

}

[root@web01 conf]#

2．精简nginx主配置文件的文件信息

[root@web01 conf]# egrep -v "#|^$" nginx.conf.default >nginx.conf 精简nginx主配置文件的信息

[root@web01 conf]# cat nginx.conf --查看

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name localhost;

location / {

root html;

index index.html index.htm;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

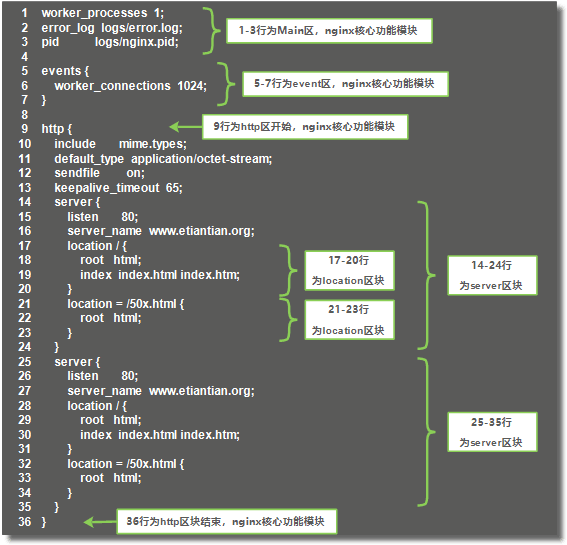
}

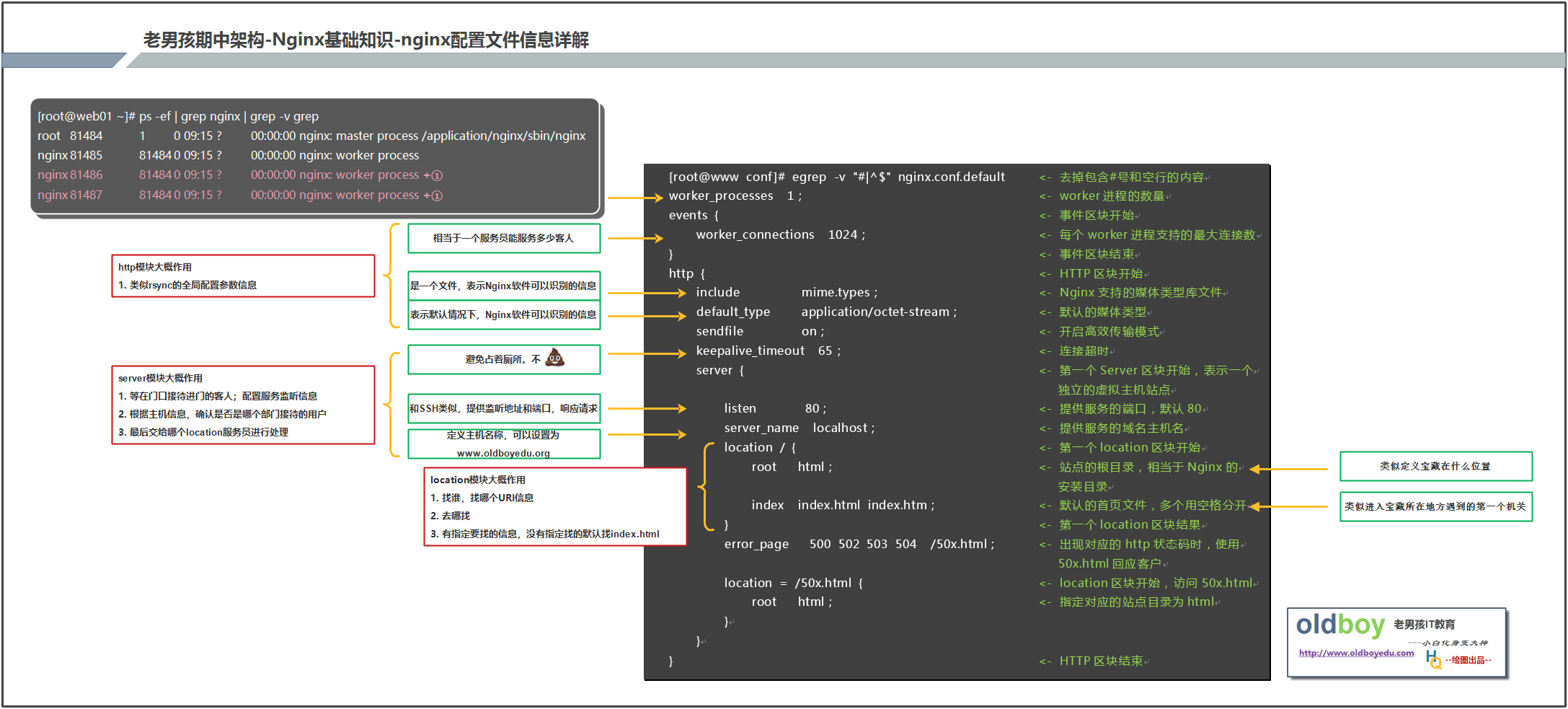
}

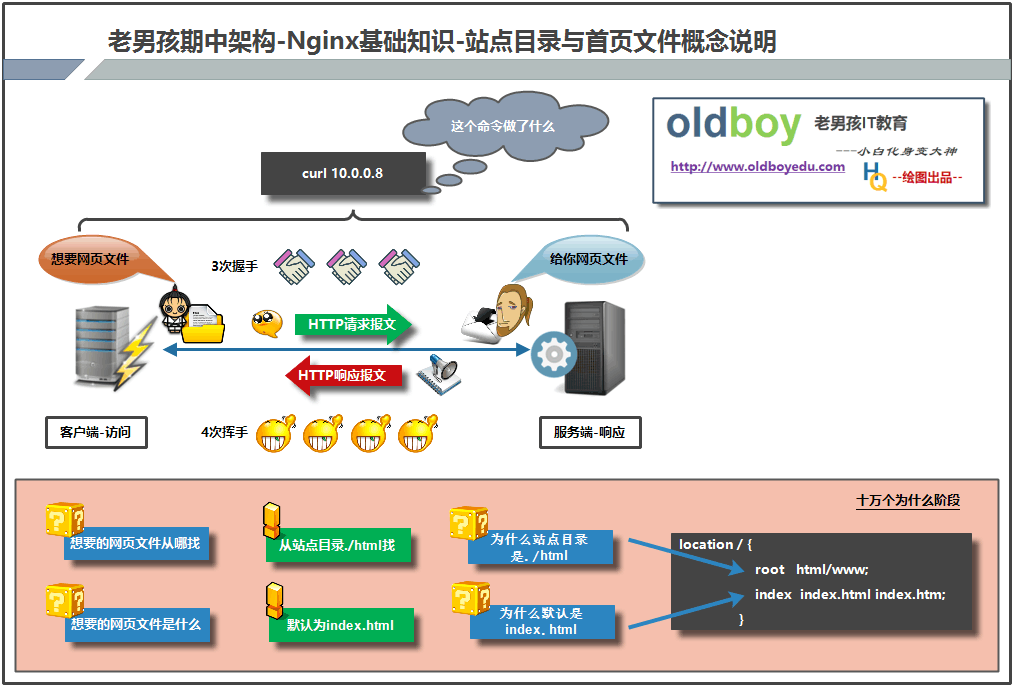
}

[root@web01 conf]#

#### nginx程序主配置文件详解







上面文件太大，如果看的时候可以下载下来看

[root@web01 conf]# cat -n nginx.conf

1 worker\_processes 1; --worker的进程数量

解释：root@web01 nginx-1.14.0]# ps -ef|grep nginx

root 15596 1 0 19:45 ? 00:00:00 nginx: master process /application/nginx/sbin/nginx

www 15852 15596 0 21:42 ? 00:00:00 nginx: worker process

root 15963 15602 0 22:57 pts/2 00:00:00 grep --color=auto nginx

[root@web01 nginx-1.14.0]#

这里设置的worker\_processes 的数值是1，那么在ps -ef里面查询的就是1，如果设置的是2 那么在ps -ef 里面查询的就是2.

2 events { --事件区块的开始

3 worker\_connections 1024; --- 代表一个 worker进程可以同时处理1024个并发数（连接数）

4 } ---事件区块的结束

5 http { ----http区块的开始

6 include mime.types; ---支持的媒体类型库文件 代表使用这个主程序文件的时候调用 mine.types这个文件。 mine.type这个文件和nginx这个程序文件在同一个目录里面

7 default\_type application/octet-stream; --默认媒体类型

8 sendfile on; --开启高效传输方式

9 keepalive\_timeout 65; --连接超时

http区块相当于rsync全局配置参数

10 server { ------server区块的开始 表示一个独立虚拟主机站点 server模块的作用，配置服务监听信息，根据主机信息，确认接待哪一个用户，最后交给location服务员处理

11 listen 80; ---提供服务的端口，默认是80

12 server\_name localhost; --提供服务域名的主机名 可以修改到对应的web服务主机名称

或者修改的时候，server\_name [www.shiyanbo.org](http://www.shiyanbo.org) www是主机名 shiyanbo.org是万网地址

13 location / { ---第一个location区块开始

14 root html; **---站点的根目录，相当于nginx的安装目录**

15 index index.html index.htm; **默认的首页文件，多个用空格分开**

16 } ---第一个location区块结束

17 error\_page 500 502 503 504 /50x.html; 出现对应http状态码的时候，使用/50x.html恢复客户

18 location = /50x.html { --loication区块的开始 location木块大致作用，找谁，找哪一个url，去哪里找，找到指定的信息，找不到则找 index.html

19 root html; --- 指定站点目录html

20 }

21 }

22 } -----------------------http区块结束

[root@web01 conf]#

原则：这里面的每一行是以分号为结束的，大括号是成对出现的。

检查是否是成对的大括号 第一种方法 鼠标放到一个大括号，如果是成对的大括号，另外一个大括号也会闪烁，或者是选中一个大括号，时候shift5 --% 会来回挑来挑去

### html目录

drwxr-xr-x 2 root root 4096 May 16 10:57 html

web服务器的站点目录

### 日志目录

drwxr-xr-x 2 root root 4096 May 16 19:45 logs

日志目录

### sbin目录

4.drwxr-xr-x 2 root root 4096 May 16 19:12 sbin

程序命令的保存目录

[root@web01 sbin]# /application/nginx/sbin/nginx -h

nginx version: nginx/1.14.0

Usage: nginx [-?hvVtTq] [-s signal] [-c filename] [-p prefix] [-g directives]

Options:

-?,-h : this help

帮助

-v : show version and exit

显示软件版本并且推出

-V : show version and configure options then exit

显示版本信息和配置参数并且退出

-t : test configuration and exit

配置文件的语法检查参数（类似于ansible语法检查功能）

-T : test configuration, dump it and exit

配置文件语法检查，并且将配置文件的信息显示出来

-q : suppress non-error messages during configuration testing

只显示错误的信息，在测试的时候

-s signal : send signal to a master process: stop, quit, reopen, reload

发送状态到master主进程 stop(关闭) reload 重启

nginx -s stop (关闭) nginx -s reload 重启

-p prefix : set prefix path (default: /application/nginx-1.14/)

-c filename : set configuration file (default: conf/nginx.conf)

-g directives : set global directives out of configuration file

#### 查看之前已经配置好服务器上的nginx编译安装的配置

/application/nginx/sbin/nginx -V

其中 /application/nginx是自己定义的软连接目录 sbin/nginx 是编译安装之后生成的目录。

[root@web01 nginx]# /application/nginx/sbin/nginx -V

nginx version: nginx/1.14.0

built by gcc 4.4.7 20120313 (Red Hat 4.4.7-18) (GCC)

built with OpenSSL 1.0.1e-fips 11 Feb 2013

TLS SNI support enabled

configure arguments: --prefix=/application/nginx-1.14 --user=www --group=www --with-http\_ssl\_module --with-http\_stub\_status\_module

[root@web01 nginx]#

#### nginx命令几个重要的参数

1. nginx -V 显示nginx服务编译安装时候的配置
2. nginx -t 在修改完nginx主程序配置文件之后可以测试一下是否修改成功
3. nginx -s stop 代表关闭 nginx 进程
4. nginx -s reload 代表的是平滑重启

这里需要注意的是：在修改虚拟主机配置文件里面的监听的地方的时候，如果定义或者修改了 以ip地址为监听的时候，这个时候不能平滑重启了，一定是 nginx -s top 然后 nginx

另外如果把监听的ip修改了之后，变成其他的端口或者什么的 这个时候代表的是修改到了ip的监听，所以这个时候也需要nginx -s top 然后 nginx

#### master进程和worker进程说明

[root@web01 nginx]# ps -ef |grep nginx

root 15596 1 0 19:45 ? 00:00:00 nginx: master process /application/nginx/sbin/nginx

www 15597 15596 0 19:45 ? 00:00:00 nginx: worker process

说明 nginx：master process 是主进程 ，nginx:worker process 副进程 只要master process 进程还存在，那么nginx这个进程就会一直，存在的。无论你杀死多少次 worker process 进程 nginx进程都会一直存在。

#### 设置nginx命令环境变量（替代全路径输出）

分析：目前在执行nginx相关的命令的时候，需要/application/nginx/sbin/nginx 然后后面在跟其他相关的参数

因为这个命令没有配置在 环境里面。所以就么有办法实现输出一个nginx 就可以执行，所以现在思路就是把这个命令设置在环境变量的配置文件里面然后source /etc/profile 即可

[root@web01 sbin]# vim /etc/profile 编辑这个文件

# /etc/profile

# System wide environment and startup programs, for login setup

# Functions and aliases go in /etc/bashrc

export PATH="$PATH:/application/nginx/sbin/" **--注意这个地方** export PATH= 正常定义环境的变量的格式

"$PATH:/application/nginx/sbin/" 正常情况下用的是单引号，这里用的是双引号，因为$PATH：需要被识别

因为是需要把/application/nginx/sbin 这个命令加入到环境变量里面所以用了一个$PATH 这样就不会改变原来的变量的基础上增加内容。

"/etc/profile" 84L, 2053C written

[root@web01 sbin]#

[root@web01 sbin]# source /etc/profile --使修改的配置文件生效

[root@web01 sbin]# nginx -V --测试

nginx version: nginx/1.14.0

built by gcc 4.4.7 20120313 (Red Hat 4.4.7-18) (GCC)

built with OpenSSL 1.0.1e-fips 11 Feb 2013

TLS SNI support enabled

configure arguments: --prefix=/application/nginx-1.14 --user=www --group=www --with-http\_ssl\_module --with-http\_stub\_status\_module

[root@web01 sbin]#

## 利用nginx软件搭建一个简单的网站

### 第一个里程碑：编写nginx配置文件

这里声明一下，无论怎么搭建一个简单的网站，前面的步骤部署的还是一样的，nginx软件包的下载，安装依赖软件，创建worker进程，对软件进行编译安装，创建程序的软连接，启动nginx服务，检查测试。

在以上没有问题之后，这个时候应该做的是 精简nginx主配置文件的内容。之后是下面的内容

--------------------------------------------------------

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

root html/www;

index oldboy.html index.htm;

}

}

（2）检查配置文件是否修改正确

nginx -t

--------------------------------说明-----------------------

说明 oldboy.html 对应下面的 oldboy.html文件的内容。 html/www 这个站点目录 看是否有，没有需要创建

server\_name [www.etiantian.org](http://www.etiantian.org) 这个需要在本地主机..\etc\hosts 文件 和Linux /etc/hosts 文件配置

**这里面的配置文件，修改的时候需要注意细节，1.括号是成对出现的（当鼠标移动到其中一个的时候会闪烁，或者是shift 5 ==% 也可以） 2.另外内容后面是冒号分割。**

### 第二个里程碑：根据配置文件创建站点目录和首页文件和配置主机名

#### 将本地主机的\etc\hosts 文件和linux主机上面的/etc/hosts文件进行修改

本地主机配置

1. 本地主机上hosts文件进行配置 我的电脑--  C:\Windows\System32\drivers\etc\ --找到hosts文件进行修改 如下

# Copyright (c) 1993-2009 Microsoft Corp.

#

# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.

#

# This file contains the mappings of IP addresses to host names. Each

# entry should be kept on an individual line. The IP address should

# be placed in the first column followed by the corresponding host name.

# The IP address and the host name should be separated by at least one

# space.

#

# Additionally, comments (such as these) may be inserted on individual

# lines or following the machine name denoted by a '#' symbol.

#

# For example:

#

# 102.54.94.97 rhino.acme.com # source server

# 38.25.63.10 x.acme.com # x client host

# localhost name resolution is handled within DNS itself.

# 127.0.0.1 localhost

# ::1 localhost

10.0.0.7 www.etiantian.org

保存退出

Linux主机配置

1. vim /etc/hosts

172.16.1.7 web01 [www.etiantian.org](http://www.etiantian.org)

保存退出

#### 创建站点目录

mkdir /application/nginx/html/www -p

#### 配置首页oldboy.html文件

cd /application/nginx/html/www

vim oldboy.html

<html>

<meta charset="utf-8">

<head>

<title>朱超最爱</title>

</head>

<body>

哈哈哈哈哈哈

<table border=1>

<tr> <td>01</td> <td>宫下华奈</td> </tr>

<tr> <td>02</td> <td>水野朝阳</td> </tr>

<tr> <td>03</td> <td>波野结衣</td> </tr>

</table>

<a href="http://www.qiuxia63.com/lunli/">

<img src="xixi.jpg" />

</a>

</body>

</html>

[root@web01 www]#

--------------------------------html语言解析---------------------------------

<html> --html 开始

<meta charset="utf-8"> --代表字符集

<head> --代表头信息

<title>上海01期最牛</title>

</head> --代表头信息

<body> ---body部分开始

上海01期最牛

<table border=1> ---表

<tr> <td>01</td> <td>oldboy</td> </tr> tr 表示行 td 表示列

<tr> <td>02</td> <td>oldgirl</td> </tr>

<tr> <td>03</td> <td>olddog</td> </tr>

</table>

<a href="http://blog.oldboyedu.com"> --表示一个超连接 **这个可以修改**

<img src="oldboy.jpg" /> ---代表通过图片连接过去 **这个也可以修改，就是把图片放到站点目录下面**

</a>

</body> ---body部分结束

</html> -----------html结束

### 第三个里程碑：重启nginx服务

nginx -s reload

-----------------------------------------------

只有nginx主配置文件修改了，需要重启nginx服务

站点目录下面内容发生了变更，不需要重启nginx服务

### 第三个里程碑检查测试

1. ping 本地主机（检查本地域名和主机通不通）

[f:\~]$ ping www.etiantian.org

正在 Ping www.etiantian.org [10.0.0.7] 具有 32 字节的数据:

来自 10.0.0.7 的回复: 字节=32 时间=25ms TTL=64

来自 10.0.0.7 的回复: 字节=32 时间<1ms TTL=64

10.0.0.7 的 Ping 统计信息:

数据包: 已发送 = 2，已接收 = 2，丢失 = 0 (0% 丢失)，

往返行程的估计时间(以毫秒为单位):

最短 = 0ms，最长 = 25ms，平均 = 12ms

1. curl [www.etiantian.org](http://www.etiantian.org)

[root@web01 ~]# curl www.etiantian.org

10.0.0.7 [www.etiantian.org](http://www.etiantian.org)

# nginx虚拟主机配置实战

## 虚拟主机的概念和类型

### 虚拟主机的概念

所谓虚拟主机，在web服务器里面就是一个独立的网站站点，这个站点对应独立的域名（也可以是IP或者端口），具有独立的程序及资源目录，可以独立地对外提供服务共用户访问

这个独立的站点在配置里面由一定格式的标签段标记的，对于apace软件来说，一个虚拟主机的标签段，通常包含在<virtualhost></virtualhost>内。而nginx软件则使用一个server{}标签来标识一个虚拟主机，一个web服务里面可以有多个虚拟主机标签对。既可以同时支持多个虚拟主机站点。

总结 1. 虚拟主机--网站站点

2. nginx 里面的虚拟主机在server区块里面定义

server {

listen 80; --可以修改

server\_name localhost; --可以修改不同的域名

root html; ---网站对应的站点目录

index index.html index.htm ---网站默认的首页文件

}

-----------------------------------------------------------------------------------

server { ------server区块的开始 表示一个独立虚拟主机站点 server模块的作用，配置服务监听信息，根据主机信息，确认接待哪一个用户，最后交给location服务员处理

listen 80; ---提供服务的端口，默认是80

server\_name localhost; --提供服务域名的主机名 可以修改到对应的web服务主机名称

或者修改的时候，server\_name [www.shiyanbo.org](http://www.shiyanbo.org) www是主机名 shiyanbo.org是万网地址

root html; **---站点的目录**

index index.html index.htm; **默认的首页文件，多个用空格分开**

}

1. 一个web服务器里面可以创建多个sever{} 标签对，也就是说可以创建多个独立网站的站点，也就是说可以创建多个虚拟主机，在创建的时候，建议把这个虚拟主机的配置文件汇总放在同一个目里面，而这个目录和nginx.conf在同一个目录下面

### 虚拟主机类型

虚拟主机配置类型分为三种：1.基于域名的方式进行配置2.基于端口的方式进行配置3.基于IP地址方式进行配置

### 基于域名方式进行配置（重要）

#### 第一个里程碑：申请域名信息

这里还是要声明一下：进行第一步之前的时候还是要部署好nginx软件的服务，也就是nginx的七个步骤，加上一步精简nginx主配置文件

第一步：百度搜索 万网 第二步：转到域名注册阿里云<https://wanwang.aliyun.com/?utm_medium=text&utm_source=bdbrandww&utm_campaign=bdbrand&utm_content=se_103066>

第三步：输入自己想要注册的域名看是否注册，然后找一个没有注册进行注册。然后按照上面要求一步一步进行注册 知道成功

#### 第二个里程碑：编写配置文件并测试

(1)配置文件编写第一步

[root@web01 conf]# pwd

/application/nginx/conf

[root@web01 conf]# cat nginx.conf

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name [www.etiantian.org](http://www.etiantian.org);

root html/www;

index index.html index.htm;

}

server {

listen 80;

server\_name bbs.etiantian.org;

root html/bbs;

index index.html index.htm;

}

server {

listen 80;

server\_name blog.etiantian.org;

root html/blog;

index index.html index.htm;

}

}

[root@web01 conf]#

（2）测试

nginx -t

-------------------------说明--------------------

关于这个配置文件修改的地方

server {

listen 80;

server\_name [www.etiantian.org](http://www.etiantian.org); ---这个需要在本地主机hosts和Linux的hosts文件里面对应修改

root html/www; --需要在静态网页目录html下面创建这个目录

index index.html index.htm; ---需要在www目录下面创建 index.html这个文件

}

---------------上面的解释同上---------------

server {

listen 80;

server\_name bbs.etiantian.org;

root html/bbs;

index index.html index.htm;

}

server {

listen 80;

server\_name blog.etiantian.org;

root html/blog;

index index.html index.htm;

}

#### 第三个里程碑：根据配置文件，进行DNS解析，创建目录，创建虚拟主机首页文件

（1）A本地主机上hosts文件进行配置 我的电脑--  C:\Windows\System32\drivers\etc\ --找到hosts文件进行修改

10.0.0.7 [www.etiantian.org](http://www.etiantian.org) bbs.etiantian.org blog.etiantian.org

（1）B 在Linux虚拟主机上/etc/hosts 文件上进行修改

[root@web01 conf]# vim /etc/hosts

10.0.0.7 www.etiantian.org blog.etiantian.org bbs.etiantian.org

(2)mkdir -p /application/nginx/html/{www,bbs,blog}

(3)for name in www bbs blog;do echo "10.0.0.7 $name.ettian.org" >/application/nginx/html/$name/index.html;done

第三步说明：这里的意思就是 在三个虚拟主机上对应的站点目录下面分别创建index.html 文件，然后往里面追加内容

这里不会for循环的话，可以分别创建文件，然后分别编辑html文件即可（这里的html文件的名称和前面的 index index.html index.htm 正好这对上）另外这里的虚拟主机的首页html文件太简单了，正常的是html标签语言的。

有head body等信息的。

#### 第四个里程碑：启动nginx服务

nginx

-------------------------------------------------

如果nginx服务已经启动了，那么就是 nginx -s reload 如果修改了ip监听 那么就是 nginx -stop

然后 nginx

#### 第五个里程碑：进行测试检查

1.在本地cmd里面 分别屏这三个域名

2.在linux虚拟机上其他不同主机上分别 curl 域名

3. 在浏览器上分别输入这三个域名即可

### 基于端口的方式进行配置

#### 第一个里程碑：修改配置文件并测试

这里还是要声明一下：进行第一步之前的时候还是要部署好nginx软件的服务，也就是nginx的七个步骤，加上一步精简nginx主配置文件

[root@web01 conf]# pwd

/application/nginx/conf

（1）修改配置文件

[root@web01 conf]# cat nginx.conf

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

root html/www;

index index.html index.htm;

}

server {

listen 8800;

server\_name bbs.etiantian.org;

root html/bbs;

index index.html index.htm;

}

server {

listen 80;

server\_name blog.etiantian.org;

root html/blog;

index index.html index.htm;

}

}

[root@web01 conf]#

（2）nginx -t

----------------解释----------------------

server {

listen 80;

server\_name [www.etiantian.org](http://www.etiantian.org); --修改了域名，那么需要去本地主机和linux主机上的hosts文件修改

root html/www; ---需要到html目录下面创建 www目录

index index.html index.htm; 需要在www目录下面配置 index.html

}

server {

listen 8800; **--修改了监听端口，在连接测试的时候需要加上：8800**

server\_name bbs.etiantian.org; --同上

root html/bbs; --同上

index index.html index.htm; --同上

}

server {

listen 80;

server\_name blog.etiantian.org; --同上

root html/blog; --同上

index index.html index.htm; --同上

}

#### 第二个里程碑：根据配置文件，进行DNS解析，创建目录，创建虚拟主机首页文件

（1）A本地主机上hosts文件进行配置 我的电脑--  C:\Windows\System32\drivers\etc\ --找到hosts文件进行修改

10.0.0.7 [www.etiantian.org](http://www.etiantian.org) bbs.etiantian.org blog.etiantian.org

（1）B 在Linux虚拟主机上/etc/hosts 文件上进行修改

[root@web01 conf]# vim /etc/hosts

10.0.0.7 www.etiantian.org blog.etiantian.org bbs.etiantian.org

(2)mkdir -p /application/nginx/html/{www,bbs,blog}

(3)for name in www bbs blog;do echo "10.0.0.7 $name.ettian.org" >/application/nginx/html/$name/index.html;done

第三步说明：这里的意思就是 在三个虚拟主机上对应的站点目录下面分别创建index.html 文件，然后往里面追加内容

这里不会for循环的话，可以分别创建文件，然后分别编辑html文件即可（这里的html文件的名称和前面的 index index.html index.htm 正好这对上）另外这里的虚拟主机的首页html文件太简单了，正常的是html标签语言的。

有head body等信息的。

#### 第三个里程碑：启动nginx服务

nginx

-------------------------------------------------

如果nginx服务已经启动了，那么就是 nginx -s reload 如果修改了ip监听 那么就是 nginx -stop

然后 nginx

#### 第四个里程碑：进行测试检查

1.在本地cmd里面 分别屏这三个域名

2.在linux虚拟机上其他不同主机上分别 curl 域名

3. 在浏览器上分别输入这三个域名即可 这里说明一下在连接bbs虚拟主机的时候 需要 bbs.etiantian.org:8800

### 用户访问nginx网站服务底层的逻辑过程（重要）

1. 通过DNS解析获取IP地址，利用IP地址向服务端访问
2. 服务端接收到请求的数据包，数据包解封装的过程

（1）三层：识别目标IP地址信息和服务监听地址信息匹配了

（2）四层：识别目标端口信息

（3）上层：识别HTTP协议信息，识别请求的host信息

当访问的host主机域名信息不存在的时候，默认显示配置文件中第一个虚拟主机网站信息

总结：这里核心的思想是 当用户访问nginx网站服务的时候，首先通过域名去配置文件里面找对应的端口，找到对应的端口然后在找对用的域名信息，如果找不到对应的域名信息的话，那么他就会在找到对应的端口的几个域名里面显示最上面的一个域名。

举例说明：以下是三个虚拟主机的配置信息

server {

listen 80;

server\_name www.etiantian.org;

root html/www;

index index.html index.htm;

}

server {

listen 8800;

server\_name bbs.etiantian.org;

root html/bbs;

index index.html index.htm;

}

server {

listen 80;

server\_name blog.etiantian.org;

root html/blog;

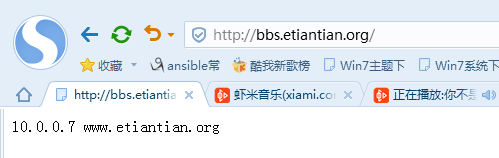
index index.html index.htm;

}

例子1：

一访问bbs主机，但是端口是默认的80 ，首先通过端口他匹配到 www虚拟主机和blog虚拟主机。

但是匹配域名的时候发现没有bbs这个虚拟主机，所以就找到第一个80端口的主机 wwww虚拟主机



例子2：

首先访问的是 8800端口的主机，找到bbs虚拟主机，然后找www主机发现端口不匹配，所以这个时候就找到了 bbs虚拟主机



### 基于IP地址方式进行配置

重点强调：nginx服务配置文件中只要涉及到IP地址的修改，都要重启nginx服务，而不是平滑重启

具体是nginx -s stop 然后nginx重启

基于ip地址的配置和前面端口配置一样的，就是在server{} 这个模块里面需要把其他两个虚拟主机的信息注释掉，不然可能会报错。

1.修改配置文件

server {

listen 10.0.0.7:80;

server\_name www.etiantian.org;

root html/www;

index index.html index.htm;

}

重启nginx服务。

1. 重启nginx服务

重点强调：nginx服务配置文件中只要涉及到IP地址的修改，都要重启nginx服务，而不是平滑重启

## nginx问题汇总说明

01. 当访问网站配置文件与解析都没有问题时候，显示的页面与真正正确页面不同？

原因：浏览器有缓存

解决：1. 清除缓存记录信息

2. 打开谷歌浏览器的无痕模式（如果还有缓存问题ctrl+F5强制刷新）

02. 配置所有服务文件时，一定要注意规范

03. 排错思路过程：

第一步：利用无痕模式进行测试（避免缓存问题）

第二步：域名有没有解析正确

ping 域名信息

第三步：在Linux上进行一下测试

# curl www.etiantian.org

curl: (7) couldn't connect to host

第四步：检查服务有没有启动正确

netstat -lntup|grep 80

第五步：防火墙服务有没有阻止

第六步：检查配置文件是否正确

第七步：试着重启一下nginx服务

04. 访问网站的时候，没有配置禁止指定用户访问，但是显示出403错误？

403错误出现的原因：

1）禁止用户访问

2）当站点目录下面没有指定的首页文件的时候，访问网站会报403错误

## 企业当中规范化编写nginx配置

在企业中，因为虚拟主机很多，如果把每一个虚拟主机的信息都配置在一个配置文件，就会显得虚拟主机信息特别长，并且如果配置文件出错了也不好排查，所以这个时候我们的建议是每一个虚拟主机都单独生成一个单独的配置文件，然后用主进程调用这些配置文件即可。

### 第一个里程碑：编写nginx主配置文件

这里还是要声明一下：进行第一步之前的时候还是要部署好nginx软件的服务，也就是nginx的七个步骤，加上一步精简nginx主配置文件

c[root@web01 conf]# pwd

/application/nginx/conf

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

include extra/www/www.conf;

include extra/other/\*;

}

------------------------------------------------------------------------------------------

[root@web01 conf]# pwd

/application/nginx/conf

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

include extra/www/www.conf; 这个需要在/application/nginx/conf创建extra/www目录和[www.conf](http://www.conf)文件

include extra/other/\*; 这个需要在/application/nginx/conf创建extra/other目录和bbs.conf文件和blog.conf文件

并且一个每一个conf文件都是server{} 区块里面的内容

-----------------------------------配置说明--------------------

### 第二个里程碑：创建对应的目录和文件

#### 第一：先创建/application/nginx/conf/extra/www目录和[www.conf](http://www.conf)文件

A1

mkdir -p /application/nginx/conf/extra/www

A1-www

[root@web01 conf]# cd /application/nginx/conf/extra/www

[root@web01 www]# vim [www.conf](http://www.conf)

server {

listen 80;

server\_name www.etiantian.org;

root html/www;

index index.html index.htm;

}

#### 第二：创建/application/nginx/conf/extra/other目录和下面的配置文件

A1 创建目录

mkdir -p /application/nginx/conf/extra/other

A2 创建配置文件

A2-bbs

[root@web01 other]# vim bbs.conf

server {

listen 80;

server\_name bbs.etiantian.org;

root html/bbs;

index index.html index.htm;

}

A2-blog

[root@web01 other]# vim blog.conf

server {

listen 80;

server\_name blog.etiantian.org;

root html/blog;

index index.html index.htm;

}

------------------配置说明----------------------------------------------------

[root@web01 www]# vim [www.conf](http://www.conf)

server {

listen 80;

server\_name [www.etiantian.org](http://www.etiantian.org); ---需要在本地和linux hosts文件里面配置这个域名

root html/www; ---需要在html里面创建www目录并且 配置index.html文件

index index.html index.htm; --需要编写index.html文件

}

说明剩下的 bbs主机和blog主机都需要 配置hosts文件以及创建目录和编写index.html文件

### 第三个里程碑：配置DNS解析，创建站点目录以及创建html文件

1）A本地主机上hosts文件进行配置 我的电脑--  C:\Windows\System32\drivers\etc\ --找到hosts文件进行修改

10.0.0.7 [www.etiantian.org](http://www.etiantian.org) bbs.etiantian.org blog.etiantian.org

（1）B 在Linux虚拟主机上/etc/hosts 文件上进行修改

[root@web01 conf]# vim /etc/hosts

10.0.0.7 www.etiantian.org blog.etiantian.org bbs.etiantian.org

(2)mkdir -p /application/nginx/html/{www,bbs,blog}

(3)for name in www bbs blog;do echo "10.0.0.7 $name.ettian.org" >/application/nginx/html/$name/index.html;done

第三步说明：这里的意思就是 在三个虚拟主机上对应的站点目录下面分别创建index.html 文件，然后往里面追加内容

这里不会for循环的话，可以分别创建文件，然后分别编辑html文件即可（这里的html文件的名称和前面的 index index.html index.htm 正好这对上）另外这里的虚拟主机的首页html文件太简单了，正常的是html标签语言的。

有head body等信息的。

### 第四个里程碑：启动nginx服务

（1）nginx -t 测试语法

（2）nginx -s reload 平缓加载

-------------------------------------------------

如果nginx服务已经启动了，那么就是 nginx -s reload 如果修改了ip监听 那么就是 nginx -stop

然后 nginx

### 第五个里程碑：进行测试检查

1.在本地cmd里面 分别屏这三个域名

2.在linux虚拟机上其他不同主机上分别 curl 域名

3. 在浏览器上分别输入这三个域名即可

## 老师讲的配置过程

企业当中规范化编写nginx配置

每一个虚拟主机都生成一个单独的配置文件

第一个里程：创建一个虚拟主机配置文件保存目录

[root@web01 conf]# pwd

/application/nginx/conf

[root@web01 conf]# mkdir extra

第二个里程: 生成虚拟主机配置文件

/application/nginx/conf

[root@web01 conf]# sed -n '10,15p' nginx.conf >extra/www.conf

[root@web01 conf]# sed -n '16,21p' nginx.conf >extra/bbs.conf

[root@web01 conf]# sed -n '22,27p' nginx.conf >extra/blog.conf

[root@web01 conf]# cat extra/\*

server {

listen 80;

server\_name bbs.etiantian.org;

root html/bbs;

index index.html index.htm;

}

server {

listen 80;

server\_name blog.etiantian.org;

root html/blog;

index index.html index.htm;

}

server {

listen 80;

server\_name www.etiantian.org;

root html/www;

index index.html index.htm;

}

说明：虚拟主机文件配置完毕，要将nginx中的虚拟主机配置信息删除

第三个里程：编写nginx主配置文件，加载虚拟主机配置文件

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

include extra/www.conf;

include extra/\*;

}

第四个里程：重启nginx服务

nginx -s reload

## 配置虚拟主机别名（主要用于测试）

在生产场景中，如果程序鞥更新，如果公司的域名太长，来回测试会很麻烦，这个时候我们可以设置一个别名，测试的时候输入别名就可以了。

server {

listen 80;

server\_name www.etiantian.org www.et.org;

root html/www;

index index.html index.htm;

}

nginx -t

nginx -s reload

然后在本地和linux上面做好解析，然后重启测试即可

## nginx状态模块页面配置

--with-http\_stub\_status\_module **通过这个模块可以统计用户的的访问网站的信息。**

root@web01 other]# nginx -V

nginx version: nginx/1.14.0

built by gcc 4.4.7 20120313 (Red Hat 4.4.7-18) (GCC)

built with OpenSSL 1.0.1e-fips 11 Feb 2013

TLS SNI support enabled

configure arguments: --prefix=/application/nginx-1.14 --user=www --group=www --with-http\_ssl\_module --with-http\_stub\_status\_module ---监控nginx的运行状态以及统计用户的访问信息

[root@web01 other]#

### nginx状态模块配置步骤

#### 第一个里程碑：编写一个status虚拟主机配置文件

A1 切换路径

[root@web01 conf]# pwd

/application/nginx/conf/extra

A2 编辑文件

[root@web01 extra]# vim status.conf

server {

listen 80;

server\_name status.etiantian.org;

stub\_status;

#root html/www;

#index index.html index.htm;

}

--------------------解释备注-----------------

server {

listen 80; --默认监听端口先不设置

server\_name status.etiantian.org; --配置一个域名来监听统计，那么本地和linux hosts文件就需要配置

stub\_status; ---这个指令超找文档看到的

#root html/www; --不需要外网访问，所以不需要站点目录

#index index.html index.htm; --不需要外网访问，就是为了监听统计

}

#### 第二个里程碑：根据status配置文件修改本地hosts文件和Linux hosts文件

（1）A本地主机上hosts文件进行配置 我的电脑--  C:\Windows\System32\drivers\etc\ --找到hosts文件进行修改

10.0.0.7 [www.etiantian.org](http://www.etiantian.org) bbs.etiantian.org blog.etiantian.org **status.etiantian.org**

（1）B 在Linux虚拟主机上/etc/hosts 文件上进行修改

[root@web01 conf]# vim /etc/hosts

10.0.0.7 www.etiantian.org blog.etiantian.org bbs.etiantian.org **status.etiantian.org**

#### 第三个里程碑：修改主配置文件，调用status虚拟主机配置文件

（1）[root@web01 conf]# vim nginx.conf

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

include extra/www/www.conf;

include extra/other/\*;

include extra/status.conf;

}

~

~ （2）nginx -t 检查语法

#### 第四个历程平滑重启nginx服务

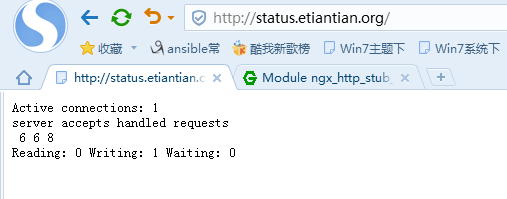
[root@web01 conf]# nginx -s reload

[root@web01 conf]#

#### 第五个里程碑：检查测试

1. ping **status.etiantian.org**
2. **在linux上面 curl status.etiantian.org**
3. **在浏览器上面 status.etiantian.org**

### 关于 status虚拟主机相关参数介绍



Active connections --当前客户端连接的数量，包含waite的连接数量（简单理解就是当前有多少客户和我服务端建立连接，还包含还有有多少等待的客户即将和我进行链接）**并发连接数就是查看的，具体看文件里面最大并发连接数量是 worker process 2\*worker connection 1024=2048** （计算较多举例说明）

The current number of active client connections including Waiting connections.

accepts --接受客户端连接的总数量 （简单理解就是只要有客户连接一次机会数量增加一次，比如通一个客户刷新一次，那么会增加一次） 6

The total number of accepted client connections.

handled --处理连接的总数量，通常这个参数值和accept值相同，除非已经达到一些资源限制（具体就是相应的恢复的总连接个数，也是累加的）6 所以这两个数值，不要差距太大，差距太大，就要调节这个并发的数值了

请求的数值也不要太大，如果太大的话，那么就是负载太高了

The total number of handled connections. Generally, the parameter value is the same as accepts unless some resource limits have been reached (for example, the worker\_connections limit).

requests --客户端请求的总数 （）客户所有请求的总数，只代表客户端请求的总数，对端有没有接受不知道（）

The total number of client requests.

Reading --当前nginx正在读取请求头的连接数量 当前连接读取报文的数量，一般并发不高的都是0

The current number of connections where nginx is reading the request header.

Writing --当前nginx将相应写回客户机的连接数量 （当前正在相应客户端的数量）

The current number of connections where nginx is writing the response back to the client.

Waiting --当前空闲客户端连接等待请求的数量 如果并发的话，就会等待 尽量不要等太多

The current number of idle client connections waiting for a request.

## 怎么和其他主机建立连接，让其他主机进行访问

模拟环境

找到局域地址：控制面板\网络和 Internet\网络连接--本地连接-192.168.11.134

虚拟机-编辑-虚拟网络编辑器-NAT模式-NAT设置-端口转发-添加-



9000 是让其他局域网里面的小伙伴访问的端口

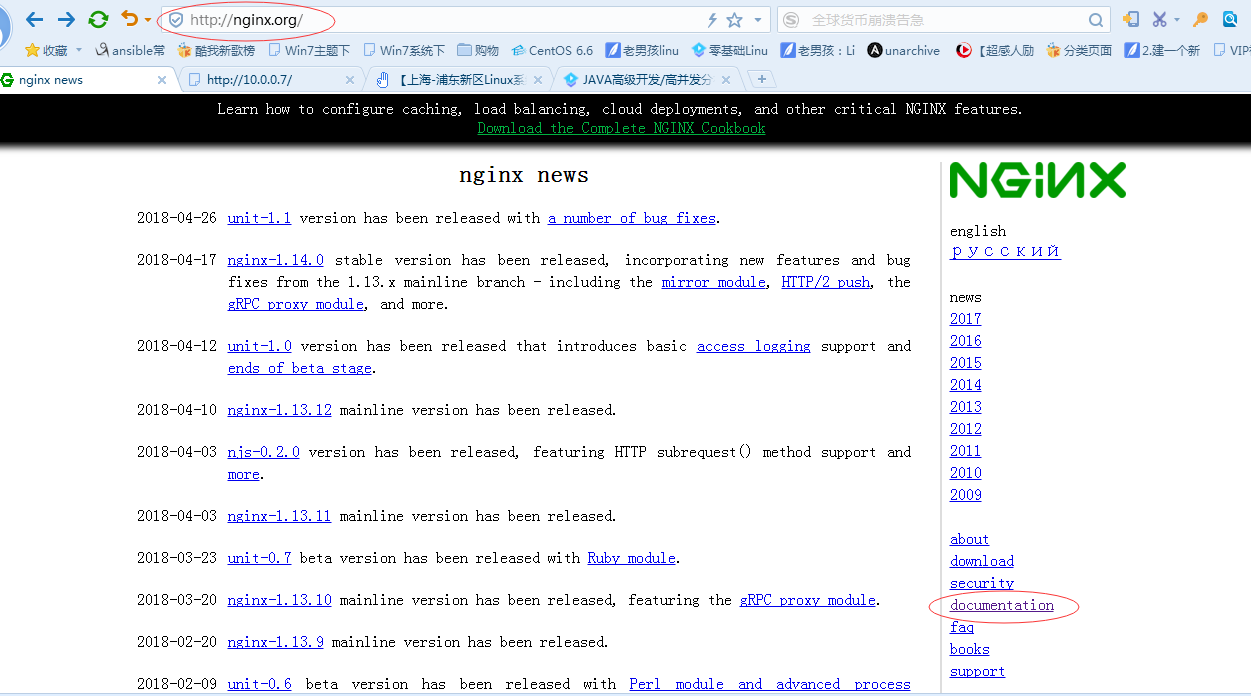
10.0.0.7 你的服务器地址

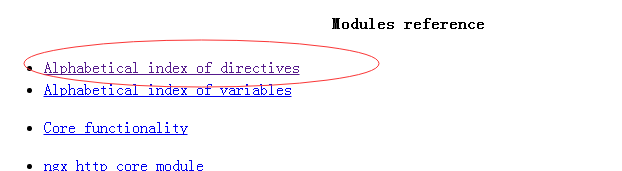
80 你的服务器配置文件默认的端口

确认

这个时候发送 192.168.11.134:9000这样大家就可以访问了。

## nginx官方文档参考查找方法





在这里面输入的都是指令：什么是指令呢？--标记黄色的都是指令 （ctrl +f）

server {

listen 80;

server\_name status.etiantian.org;

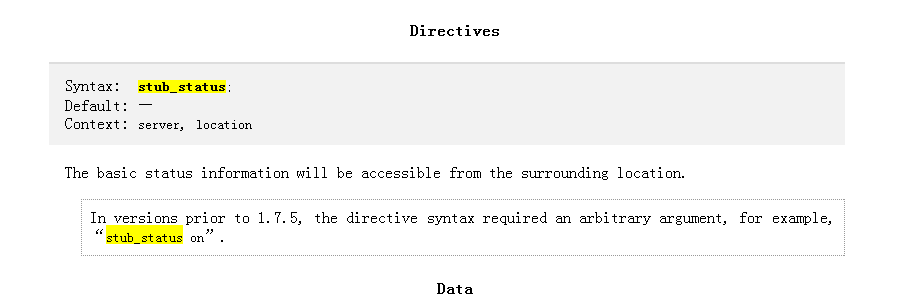
stub\_status;

#root html/www;

#index index.html index.htm;

}





## nginx服务日志的功能配置

运维人员基本的素质是可以能够看懂日志，当服务器出现问题的时候，可以通过日志来识别问题，出来问题

### nginx日志默认存放的地方

[root@web01 logs]# pwd

/application/nginx/logs

[root@web01 logs]# ll

total 36

-rw-r--r-- 1 root root 16801 May 18 01:38 access.log

-rw-r--r-- 1 root root 6646 May 18 11:27 error.log

-rw-r--r-- 1 root root 5 May 18 09:14 nginx.pid

[root@web01 logs]#

### access.log

-rw-r--r-- 1 root root 16801 May 18 01:38 access.log --nginx的访问日志

1.在日志配置文件没有进行配置的时候，nginx程序也会默认生成日志信息

#### 访问日志配置

备注：访问日志有自己的格式也有自己的存储路径，所以下面就是看看怎么配置的格式，实际工作中根据需要就行相关的修改

首先去nginx.conf原始主配置(defalut)文件里面查看，然后把原来的格式写下来，然后把路径更改一下即可。

**第一步：查看访问日志默认的配置格式**

[root@web01 ~]# cd /application/nginx/conf/

[root@web01 conf]# cat nginx.conf.default

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;

**第二步：去官方文档查看acees日志的格式存放在配置文件哪一个地方**

nginx.org- documentation-Modules reference- Alphabetical index of directives-然后搜索。 使用的格式以及存放的地方

下面是官方文档：

Example Configuration --配置例子

log\_format compression '$remote\_addr - $remote\_user [$time\_local] '

'"$request" $status $bytes\_sent '

'"$http\_referer" "$http\_user\_agent" "$gzip\_ratio"';

access\_log /spool/logs/nginx-access.log compression buffer=32k;

Directives

Syntax: access\_log path [format [buffer=size] [gzip[=level]] [flush=time] [if=condition]];

access\_log off;

Default:

access\_log logs/access.log combined;

Context: http, server, location, if in location, limit\_except

这里代表aceess配置内容放的地方环境，这里面的server http location区块都可以

另外：具体access的具体配置参考ngixn主配置文件里面的配置即可

**第三步：修改配置文件**

[root@web01 conf]# vim nginx.conf

worker\_processes 2;

error\_log logs/oldboy\_error.log error;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

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/oldboy\_access.log main;

keepalive\_timeout 65;

include extra/www/www.conf;

include extra/other/\*;

include extra/status.conf;

}

---------------------------备注说明--------------------

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/oldboy\_access.log main;

log\_format 日志的格式 main 格式的名称 后面一堆$就是具体的访问日志怎么定义的

access\_log 代表的是生成访问日志的指令

logs/oldboy\_access.log 代表的是 访问日志生成的路径是/application/nginx/logs/ oldboy\_access.log 这里面的logs/oldboy\_access.log 是相对路径。 另外默认的路径是/application/nginx/logs/access.log 可以不用修改

main 代表调用访问日志的格式

**第四步：检查语法重启nginx服务**

[root@web01 conf]# nginx -t

nginx: the configuration file /application/nginx-1.14/conf/nginx.conf syntax is ok

nginx: configuration file /application/nginx-1.14/conf/nginx.conf test is successful

[root@web01 conf]# nginx -s reload

[root@web01 conf]#

**第五步：测试新生成的访问日志文件**

[root@web01 logs]# pwd

/application/nginx/logs

[root@web01 logs]# ll --发现一旦重启nginx服务之后，这个时候就自动生成了新命名的访问日志文件，因为还没有访问，所以里面内容是0 如果不修改这个访问日志的文件，那么默认访问记录还是在acccess.log这个配置文件里面。

total 36

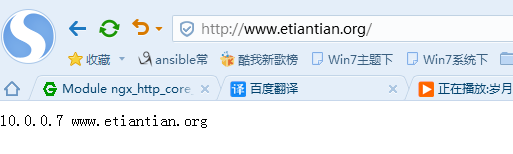
-rw-r--r-- 1 root root 16801 May 18 01:38 access.log

-rw-r--r-- 1 root root 6706 May 18 17:30 error.log

-rw-r--r-- 1 root root 5 May 18 09:14 nginx.pid

-rw-r--r-- 1 root root 0 May 18 17:30 oldboy\_access.log

[root@web01 logs]#



再次查看

[root@web01 logs]# ll

total 40

-rw-r--r-- 1 root root 16801 May 18 01:38 access.log

-rw-r--r-- 1 root root 6706 May 18 17:30 error.log

-rw-r--r-- 1 root root 5 May 18 09:14 nginx.pid

-rw-r--r-- 1 root root 611 May 18 17:37 oldboy\_access.log

**访问日志记录每一个参数具体的解释**

[root@web01 logs]# cat oldboy\_access.log

（1）日志记录

10.0.0.1 - - [18/May/2018:17:37:46 +0800] "GET / HTTP/1.1" **200 99** "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.221 Safari/537.36 SE 2.X MetaSr 1.0" "-"

10.0.0.1 - - [18/May/2018:17:37:46 +0800] "GET / HTTP/1.1" **304 0** "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.221 Safari/537.36 SE 2.X MetaSr 1.0" "-"

（2）日志格式

'$remote\_addr - $remote\_user [$time\_local] "$request" '

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

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

1. 对应说明

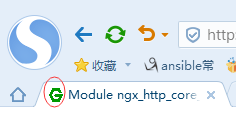
10.0.0.1 --$ remote\_user 代表的访问用户源公网IP地址（字面理解就是：远程用户）

- -

- $remote\_user 代表访问用户的认证信息

[18/May/2018:17:37:46 +0800] [$time\_local] 代表访问的时间信息

"GET / HTTP/1.1" "$request" 代表的是访问的时间信息。 另外在还会出现一种情况就是"GET /favicon.ico HTTP/1.1" 代表的就是下面截图显示的地方



怎么实现呢？ 把一个图片命名为favicon.ico 然后放在和index.html同一个目录里面就好了。

200 99 $status 代表显示状态码信息 这里解释一下，200 表示请求成功。 99代表的访问网站首页的字节数量既index.html 另外下面有一个 304 0 代表客户端有缓存信息，所以返回的字节就是0

"-" $body\_bytes\_sent 服务端相应客户端数据流量大小

"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.221 Safari/537.36 SE 2.X MetaSr 1.0" "$http\_referer" 这里面会跟一个域名地址，代表是谁盗取了你的网站的流量。盗链信息

"-" "$http\_user\_agent" 用户代理 简单理解就是：访问网站用户客户端软件信息

空 "$http\_x\_forwarded\_for" 代表的是 先不管 反向代理

### error.log

-rw-r--r-- 1 root root 6646 May 18 11:27 error.log ---nginx的错误日志

#### 错误的日志系统的配置

备注：错误日志有自己的格式也有自己的存储路径，所以下面就是看看怎么配置的格式，实际工作中根据需要就行相关的修改

首先去nginx.conf原始主配置(defalut)文件里面查看，然后把原来的格式写下来，然后把路径更改一下即可

**第一步：查看错误日志默认的配置格式**

[root@web01 ~]# cd /application/nginx/conf/

[root@web01 conf]# cat nginx.conf.default

[root@web01 conf]# cat nginx.conf.default

#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;

-------------备注说明-------------

#error\_log logs/error.log; 第一种错误日志的格式

#error\_log logs/error.log notice; 第二种错误日志的格式

#error\_log logs/error.log info; 第三种错误日志的格式

**第二步：去官方文档查看错误日志的格式存放在配置文件哪一个地方**

nginx.org- documentation-Modules reference- Alphabetical index of directives-然后搜索。 使用的格式以及存放的地方

下面是官方文档：

Syntax: error\_log file [level]; 命令的使用格式是：error\_log 文件 错误级别

Default:

error\_log logs/error.log error;

Context: main, http, mail, stream, server, location

可以放的模块下面有 main http mail server location 默认是 main模块下面

**第三步把错误日志配置放到配置文件里面**

[root@web01 conf]# vim nginx.conf

worker\_processes 2;

error\_log logs/oldboy\_error.log error;

events {

worker\_connections 1024;

}

--------------------解释备注------------

error\_log logs/oldboy\_error.log error;

error\_log 错误日志的命令

logs/oldboy\_error.log 代表的是 错误日志生成的路径是/application/nginx/logs/ oldboy\_error.log 这里面的logs/oldboy=\_error.log 是相对路径。 另外默认的路径是/application/nginx/logs/error.log 可以不用修改

error 错误的级别

**第四步：测试语法并重启nginx服务**

[root@web01 conf]# nginx -t

nginx: the configuration file /application/nginx-1.14/conf/nginx.conf syntax is ok

nginx: configuration file /application/nginx-1.14/conf/nginx.conf test is successful

[root@web01 conf]# nginx -s stop

[root@web01 conf]# nginx

[root@web01 conf]#

**第五步：检查测试**

[root@web01 logs]# pwd

/application/nginx/logs

[root@web01 logs]# ll

total 40

-rw-r--r-- 1 root root 16801 May 18 01:38 access.log

-rw-r--r-- 1 root root 6766 May 18 21:29 error.log

-rw-r--r-- 1 root root 5 May 18 21:29 nginx.pid

-rw-r--r-- 1 root root 0 May 18 17:30 oldboy\_error.log 配置文件修改之后就生成了，因为灭有错误，所以暂时就没有信息。

[root@web01 logs]#

**模拟错误**

[root@web01 conf]# nginx

nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)

nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)

nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)

nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)

nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)

^C

[root@web01 conf]#

[root@web01 logs]# ll

total 44

-rw-r--r-- 1 root root 16801 May 18 01:38 access.log

-rw-r--r-- 1 root root 6766 May 18 21:29 error.log

-rw-r--r-- 1 root root 5 May 18 21:29 nginx.pid

-rw-r--r-- 1 root root 1017 May 18 18:34 oldboy\_access.log

-rw-r--r-- 1 root root 465 May 18 21:33 oldboy\_error.log

[root@web01 logs]#

#### 关于错误日志的补充说明

错误日志的，默认情况下不指定也没有关系，因为nginx很少有错误日志的，但有时出现错误的时候是有必要记录一下错误日志的，方便我们排查问题

error\_log级别分为 debug,info, nitice,warn,error,crit 默认为 crit

crit 在日志名后边定义格式如下：

error\_log path crit;

crit 记录的日志最少，而 debug记录的日志最多

如果nginx遇到一些问题，比如502比较频繁出现，但是看默认的error\_log并没有看到有意义的信息，那么就可以调一下错误的日志级别，当你调成error级别时候，错误日志记录的内容会更加丰富。

这些级别是由低到高依次增加的

debug 相当于调试，把正确的错误的都显示出来。

info 相当于echo 执行过程中只要输出内容了，就会被记录

notice 通知信息，不是操作完的结果信息，是一些通知的信息

warn 警告，代表出现一些问题了，但是不是很严重

error 错误 一般出现错了就要关注了

crit 严重的错误

### nginx.pid

-rw-r--r-- 1 root root 5 May 18 09:14 nginx.pid --nginx的pid文件，有这个文件，nginx服务启动了，这个文件里面放的是nginx的主进程的pid 具体如下：

[root@web01 logs]# cat nginx.pid

1435

[root@web01 logs]# ps -ef |grep 1435

uid PID PPID C STIME TTY CMD

root 1435 1 0 11:21 ? 00:00:00 nginx: master process nginx

www 1971 1435 0 13:34 ? 00:00:00 nginx: worker process

www 1972 1435 0 13:34 ? 00:00:00 nginx: worker process

root 2141 2095 0 15:32 pts/5 00:00:00 grep --color=auto 1435

[root@web01 logs]#

#### kill -9 nginx进行出现的问题

正常在上面杀进程的时候，用kill 进程号 就可以把nginx的主进程杀掉，同时nginx服务的pid文件也会消失，相当于正常的关闭nginx的服务。

如果用kill -9来杀死进程的时候，如果这个时候啥的是主进程，那么这个时候只是把主进程master杀死了，woker进程和pid文件还在，所以这个时候你在重新启动nginx服务进程的时候，就会显示没有办法启动，所以以后在杀进程的时候，不建议用kill -9 杀进程，不然可能就会出错。

所以当我们在启动nginx服务的时候，一直启动失败的话，那么我们可以去检查一下nginx服务的pid文件以及worker进程是否存在

## 日志的切割

在我们工作中，如果网站访问量大的话，那么访问日志就会生成一个巨大的文件，所以这个时候我们就需要对日志进行切割，不然如果一个日志大于50兆的时候，根本就打不开。

### 利用shell脚本的形式进行日志的切割（配合定时任务）

#### 第一个里程碑：编写shell脚本

[root@web01 scripts]# vim nginx\_log.sh

#!/bin/bash

# cron cut log for nginx

#第一个里程碑：编写shell脚本

#进行日志切割

mv /application/nginx/logs/oldboy\_access.log /var/log/nginx\_access\_$(date +%F).log

#进行平滑重启nginx服务,生成新的日志信息

/application/nginx/sbin/nginx -s reload 一定要全路径，不然那在定时任务里面就会报错

#### 第二个里程碑：在命令行测试脚本

[root@web01 scripts]# /bin/sh nginx\_log.sh

[root@web01 scripts]#

[root@web01 log]# pwd

/var/log

[root@web01 log]# ll nginx\_access\_2018-05-18.log

-rw-r--r-- 1 root root 814 May 18 22:44 nginx\_access\_2018-05-18.log

[root@web01 log]#

[root@web01 logs]# pwd

/application/nginx/logs

[root@web01 logs]# ll

total 40

-rw-r--r-- 1 root root 16801 May 18 01:38 access.log

-rw-r--r-- 1 root root 7006 May 18 22:44 error.log

-rw-r--r-- 1 root root 5 May 18 21:29 nginx.pid

-rw-r--r-- 1 root root 0 May 18 22:44 oldboy\_access.log

-rw-r--r-- 1 root root 465 May 18 21:33 oldboy\_error.log

[root@web01 logs]#

#### 第三个里程碑：编写定时任务并且测试

[root@web01 log]# pwd

/var/log

[root@web01 log]# ll nginx\_access\_2018-05-18.log

-rw-r--r-- 1 root root 814 May 18 22:44 nginx\_access\_2018-05-18.log

[root@web01 log]# crontab -e

# crontab cut nginx log

\*/1 \* \* \* \* /bin/sh /server/scripts/nginx\_log.sh &>/dev/null

[root@web01 logs]# ll

total 44

-rw-r--r-- 1 root root 16801 May 18 01:38 access.log

-rw-r--r-- 1 root root 8686 May 18 23:41 error.log

-rw-r--r-- 1 root root 5 May 18 21:29 nginx.pid

-rw-r--r-- 1 root root 0 May 18 23:41 oldboy\_access.log

-rw-r--r-- 1 root root 465 May 18 21:33 oldboy\_error.log

[root@web01 logs]# pwd

/application/nginx/logs

[root@web01 logs]#

[root@web01 log]# ll

total 6132

-rw-------. 1 root root 2368 Mar 21 17:46 anaconda.ifcfg.log

-rw-r--r-- 1 root root 0 May 18 23:47 nginx\_access\_2018-05-18.log

#### 第四个里程碑：根据定时要求更改定时任务时间

# crontab cut nginx log

00 01 \* \* \* /bin/sh /server/scripts/nginx\_log.sh &>/dev/null

### 利用系统切割日志功能进行切割lograte

首先我们的定时任务，系统日志，我们多没有对它做定时备份，但是却可以自动切割，这就是系统自带的切割日志功能实现的，那么这个实现系统日志自动切割服务的配置文件在哪里呢？

/etc/logrotate.conf ---切割日志程序配置文件

/etc/logrotate.d/ ----保存特殊的配置文件的目录

其中这这个目录下面有有一个定时任务的系统配置文件

[root@web01 logrotate.d]# cat syslog

/var/log/cron --定时任务

/var/log/maillog --邮件日志

/var/log/messages --用户默认的日志

/var/log/secure ---用户登录信息的日志

/var/log/spooler

下面这些代表的是对日志切割的命令

{

sharedscripts

postrotate --代表日志切割完成后做下面的操作（重启服务）

/bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true

endscript

}

[root@web01 logrotate.d]#

扩展：如何利用logrotate服务程序实现nginx的访问日志的切割

## 扩展教学：利用location模块实现对用户访问权限的限制

1. 搭建好一台nginx的web 服务器，配置好内网网卡地址
2. web 服务的网站域名为 [www.etiantian.org](http://www.etiantian.org) 站点目录为html/www
3. 要求内网用户可以访问网站 <http://www.etiantian.org/AV> 资源信息
4. 要求外网用户禁止访问网站<http://www.etiantian.org/AV> 资源信息

解决问题：下面location的具体说明回了，这个很简单

第一里程：需要进行一个访问资源的定位

location /AV {

}

第二里程：需要完成访问策略的配置（白名单 黑名单）

host\_allow host\_deny

allow 172.16.1.0/24;

deny 10.0.0.0/24;

第三里程：将第一和第二里程进行结合

location /AV {

allow 172.16.1.0/24;

deny 10.0.0.0/24;

}

## nginx中location区块说明

01 = 精确匹配 =/AV 也就是说输出来的uri必须是/AV

02 ^~ 不做正则表达式的检查/实现优先匹配

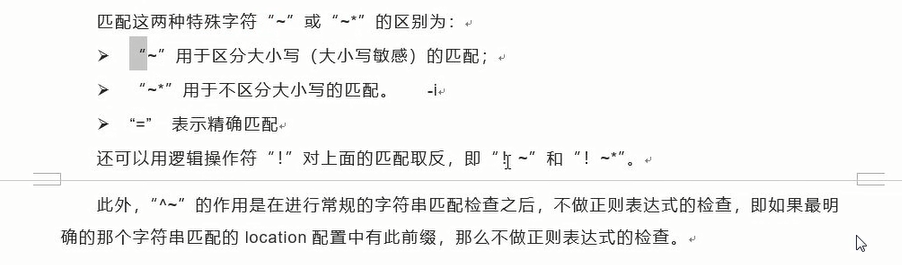
03 ~ 模糊匹配（区分大小写）--正则

03 ~\* 模糊匹配（不区分大小写）---正则

03 !~ 正则取反

04 /dir/ 根据目录进行匹配

05 / 多有信息都不匹配时，默认匹配 location /



### nginx在匹配的时候经典错误说明

需要说明的是 下面标记红色的地方。 location /test02 ----> /test02 是一个uri 是域名后面加的目录或者是文件。 和域名本身默认的存放的路径没有关系。所以下面 root html/test02 代表的是默认的路径，如果要访问

[www.etiantian.org/test](http://www.etiantian.org/test) 那么实际上他需要访问的是 html/test02/test02 下面的内容。所以一定要有 test02/test02 这个目录 。另外最后访问的一定是一个文件，一定是需要一个文件支撑的，不然一样会报错的。

所以 html/test02/test02 下面需要有一个文件

配置1

[root@web01 html]# pwd

/application/nginx/html

[root@web01 html]# tree test02/

test02/

└── test02

└── test02.html

1 directory, 1 file

[root@web01 html]# cat test02/test02/test02.html

lai da wo ya haha

[root@web01 html]#

配置文件

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

location /test02 {

root html/test02;

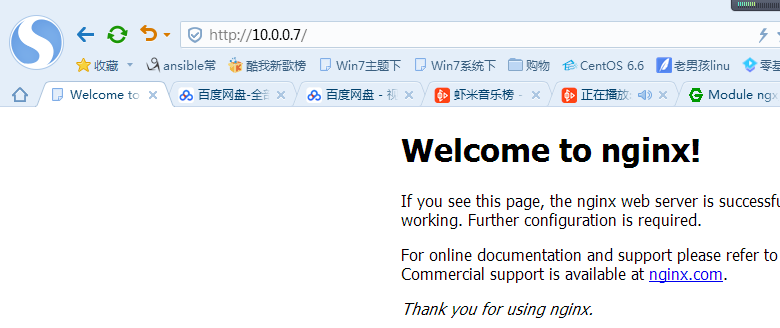
index test02.html;

}

}

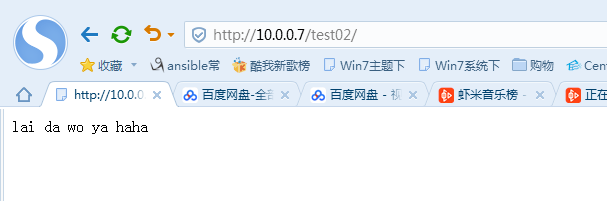
}

测试1：



原因分析： 当你输入一个www.etiantian.org的时候这个时候发现在 location /test02 里面没有匹配到，那么就会转到默认的配置文件下面的 index.html，所以这个时候就是 welcome to gninx

测试2



原因分析：

当你输入 [www.etiantian.org/test02 那么就匹配到了 location /test02 默认目录下面test02目录下面的文件 test02.html](http://www.etiantian.org/test02%20那么就匹配到了%20location%20/test02%20默认目录下面test02目录下面的文件%20test02.html) 文件，所以就会出现上图的结果。

### 普通与精确匹配的案例说明

主要讲的 精准匹配和普通匹配的区别

原则是那配置文件里面location后面的 / 或者其他的匹配 输入网页的 url 可以把普通匹配看做一个字符串，但是是一个一段一段的字符串。

不是拿网页的url来匹配配置文件里面的location

比如网页上输入的是 [www.etiantina.org](http://www.etiantina.org) 那么他的uri 是 / 如果这个时候 配置文件里面的 location 是 location /index.html 那么明显 配置文件里面的uri 小于网站的uri没有办法匹配

#### 案例1

案例1：

配置文件：说明其他路径和相关的目录和文件都准备好了。具体看配置文件说明定位

配置1：

[root@web01 www]# pwd

/application/nginx/html/www

[root@web01 www]# cat index.htm index.html

10.0.0.7 www.etiantian.org htm

10.0.0.7 www.etiantian.org

[root@web01 www]#

配置2：

[root@web01 test01]# pwd

/opt/test01

[root@web01 test01]# cat index.htm index.html

456

123

[root@web01 test01]#

-----------------------------------------------------------------------------

配置文件：

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

location /{

root html/www;

index index.html index.htm;

}

location =/{

root /opt/test01;

index index.html index.htm;

}

}

}

------------------------------------------------------------

测试结果：



原因分析：

首先当你输入[www.etianitan.org的时候，立马就会精准匹配location=/](http://www.etianitan.org的时候，立马就会精准匹配location=/)这个uri 但是 /是一个目录。没有办法给你匹配内容，所以这个时候他就会进入精准匹配目录下面的第一个索引文件 index.html文件，这个时候他就会内部做第二次跳转，变成访问的是[www.etiantian.org/index.html](http://www.etiantian.org/index.html) （index.html是uri）但是这个时候精准的/就不能匹配了，（精准匹配就是等号后面跟什么然后就会匹配什么）那么他就会继续找，然后就找到了 /index.html 所以最后结果就是10.0.0.7 www.etiantian.org

/只是一个目录最终需要引导到文件上 因为目录没有办法相应你的请求

#### 案例2

配置1：

[root@web01 www]# pwd

/application/nginx/html/www

[root@web01 www]# cat index.htm index.html

10.0.0.7 www.etiantian.org htm

10.0.0.7 www.etiantian.org

[root@web01 www]#

配置2：

[root@web01 test01]# pwd

/opt/test01

[root@web01 test01]# cat index.htm index.html

456

123

[root@web01 test01]#

-----------------------------------------------------------------------------

配置文件：

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

location /{

root html/www;

index index.html index.htm;

}

location =/index.html{

root /opt/test01;

index index.html index.htm;

}

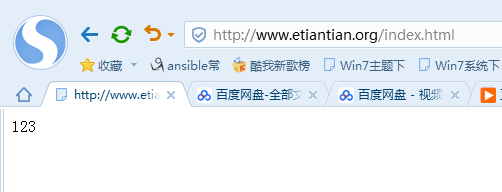
}

测试：1



原因解释：先通过/匹配到 location / 这个配置文件，然后访问到对应目录下面的 index.html 所以匹配的结果就是 10.0.0.7 [www.etiantian.org](http://www.etiantian.org)

测试2：



uri 是 /index.html 所以直接完全匹配到精准匹配的配置文件，所以这个时候就会显示 精准匹配下面的 index.html的结果123

#### 案例3

配置1：

[root@web01 www]# pwd

/application/nginx/html/www

[root@web01 www]# cat index.htm index.html

10.0.0.7 www.etiantian.org htm

10.0.0.7 www.etiantian.org

[root@web01 www]#

配置2：

[root@web01 test01]# pwd

/opt/test01

[root@web01 test01]# cat test01.htm test01.html

test01-htm

test01-html

[root@web01 test01]#

配置文件：

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

location /{

root html/www;

index index.html index.htm;

}

location =/ {

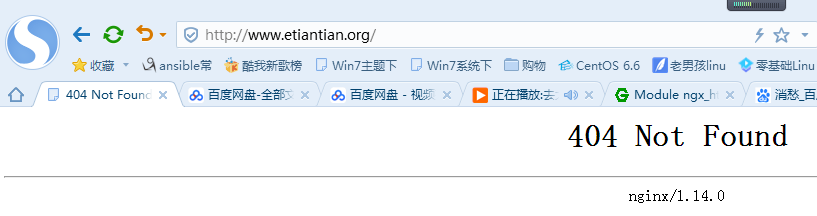
root /opt/test01;

index test01.html test01.htm;

}

}

测试1：



原因解释： 首先通过/ 找到精准匹配的 location =/ 然后 在找到他下面的 test01.html 然后进行第二次访问，这个时候访问的是 [www.etiantian.org/test01.html](http://www.etiantian.org/test01.html) 但是这个时候精准匹配就不能匹配了，那么就去不是精准匹配的 location / 匹配不到 test01.html这个文件，所以就报错404找不到了。

#### 案例4

其他配置文件都存在

配置文件

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

location /index.html{

root html/www;

index index.html index.htm;

}

# location / {

# index index.html;

# }

location =/test01.html {

root /opt/test01;

index test01.html test01.htm;

}

}

}

测试结果：



当你输入[www.etiantian.org](http://www.etiantian.org)的时候，那么就会找到默认 的index.html M那么就是 [www.etianitan.org](http://www.etianitan.org)

所以这个时候就会访问 location /index.html 所以就是 10.0.0.7 wwww.etiantian.org

#### 案例5

配置1

[root@web01 test01]# pwd

/opt/test01

[root@web01 test01]# cat test01.htm test01.html

test01-htm

test01-html

[root@web01 test01]#

配置2

[root@web01 www]# pwd

/application/nginx/html/www

[root@web01 www]# cat index.htm index.html

10.0.0.7 www.etiantian.org htm

10.0.0.7 www.etiantian.org

[root@web01 www]#

配置3

如下面配置用的还是配置1相关路径下的文件。 最红索引找的是文件，但是必须是前面root命名的路径，不然也找不到。

location / {

root /opt/test01;

index test01.html;

}

配置文件：

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

location /index.html{

root html/www;

index index.html index.htm;

}

location / {

root /opt/test01;

index test01.html;

}

location =/test01.html {

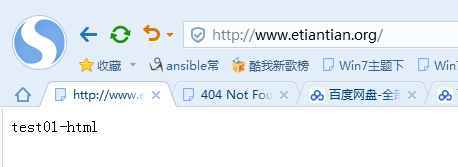
root /opt/test01;

index test01.html test01.htm;

}

}

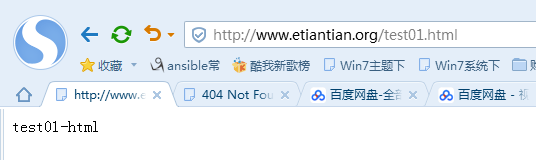
测试：



老样子 www.etiantian.org--->www.etiantian.org/test01.html(对应路径下面应该存在这个文件)------- ->location=/test01.html --->test01.html.

下面两个比较简单，就不解释了





#### 案例6

配置1

[root@web01 www]# pwd

/application/nginx/html/www

[root@web01 www]# cat index.htm index.html

10.0.0.7 www.etiantian.org htm

10.0.0.7 www.etiantian.org

[root@web01 www]#

配置2

[root@web01 test01]# pwd

/opt/test01

[root@web01 test01]# cat test01.htm test01.html

test01-htm

test01-html

[root@web01 test01]#

配置3说明

root /html/www 这个路径存在 但是这个路径下面没有index test01.html

location / {

root /html/www;

index test01.html;

}

配置文件：

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

location /index.html{

root html/www;

index index.html index.htm;

}

location / {

root /html/www;

index test01.html;

}

location =/test01.html {

root /opt/test01;

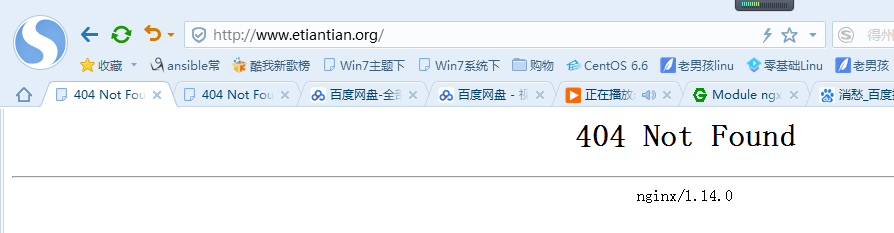
index test01.html test01.htm;

}

}

}

测试：



原因说明 当访问 / 的时候 会自动挑战其对应目录下面的test01.html文件 但是这个路径下面没有这个文件，所以就是 404 not found

#### 案例7

配置1：

[root@web01 www]# cat index.htm index.html

10.0.0.7 www.etiantian.org htm

10.0.0.7 www.etiantian.org

[root@web01 www]#

配置2：

[root@web01 test01]# pwd

/opt/test01

[root@web01 test01]# cat test01.htm test01.html

test01-htm

test01-html

[root@web01 test01]#

配置3：

location / {

index index.html; 没有路径会找到默认路径下面的，index.html（welcome nginx）

}

配置文件

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

location /index.html{

root html/www;

index index.html index.htm;

}

location / {

index index.html;

}

location =/test01.html {

root /opt/test01;

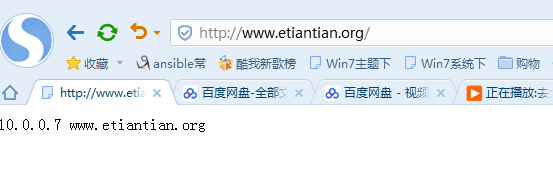
index test01.html test01.htm;

}

}

}

测试



原因说明：首先这个文件先找的 [www.etiantian](http://www.etiantian) 对应的路径下面的 index.html文件,但是没有路径，那么就会找到默认路径下面的 index.html(既welcome nginx)然后用 [www.etianitan.org/index.html](http://www.etianitan.org/index.html) --->location /index.html 这个文件 所以就会显示他对应路径下面的文件 10.0.0.7 [www.etiantian.org](http://www.etiantian.org)

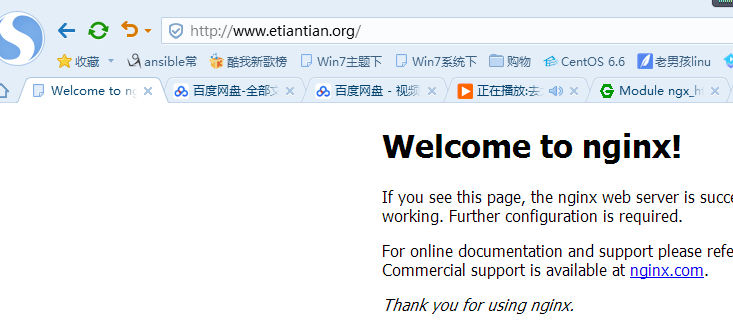
具体的可以自己就只留下一个

location / {

index index.html;

}

那么就会出现下面的结果



#### 案例8

配置1：

[root@web01 www]# pwd

/application/nginx/html/www

[root@web01 www]# cat index.htm index.html

10.0.0.7 www.etiantian.org htm

10.0.0.7 www.etiantian.org

[root@web01 www]#

配置2：

[root@web01 test01]# pwd

/opt/test01

[root@web01 test01]# cat index.htm index.html

456

123

[root@web01 test01]#

配置三

location / {

index index.htm; 默认的路径就是 html下面的文件

}

[root@web01 html]# pwd

/application/nginx/html

[root@web01 html]# ll

total 24

-rw-r--r-- 1 root root 537 May 16 10:57 50x.html

drwxr-xr-x 2 root root 4096 May 17 09:47 bbs

drwxr-xr-x 2 root root 4096 May 17 09:48 blog

-rw-r--r-- 1 root root 19 May 19 21:55 index.htm

-rw-r--r-- 1 root root 612 May 16 10:57 index.html

drwxr-xr-x 3 root root 4096 May 19 18:09 www

[root@web01 html]# cat index.htm

welcomr nginx hhah

[root@web01 html]#

配置文件：

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

location /index.html{

root html/www;

index index.html index.htm;

}

location / {

index index.htm;

}

location =/index.html {

root /opt/test01;

index index.html index.htm;

}

}

}

测试：



原因解释：

首先访问 location / 然后跳转 [www.etiantian.org/index.htm](http://www.etiantian.org/index.htm) 然后发现 location =/index.html 不匹配

发现 locaton /index.html 也不匹配，所以就会返回 默认的 index.htm 的内容 welcomr nginx hhah

### 普通URI与正则匹配的URI的区别

#### 案例1

配置1：

[root@web01 www]# cat index.htm index.html

10.0.0.7 www.etiantian.org htm

10.0.0.7 www.etiantian.org

[root@web01 www]#

配置2：

[root@web01 image]# pwd

/opt/image

[root@web01 image]# tree

.

├── index.htm

├── index.html

└── nainai

└── nainai.html

1 directory, 3 files

[root@web01 image]#

配置文件

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

location / {

root html/www;

index index.html index.htm;

}

location ~\* /image/ {

root /opt/;

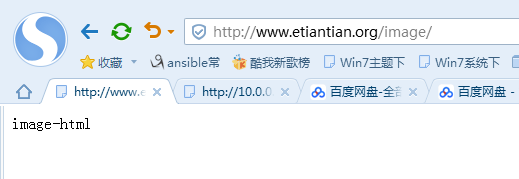
index index.html index.htm;

}

}

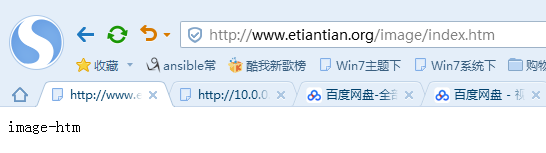
}

测试1：



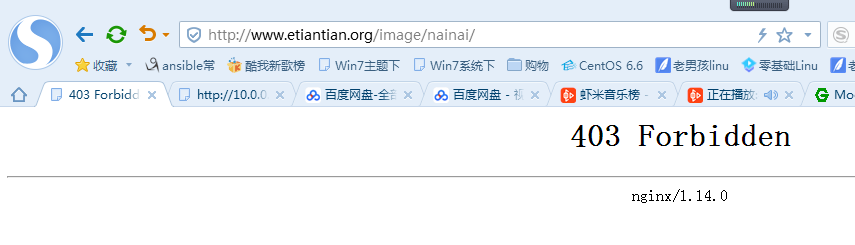
原因分析：首先当在浏览器输入 [www.etiantian.org/image/](http://www.etiantian.org/image/) 的时候，首先匹配的是 location / 这一个普通的配置。但是普通匹配一旦匹配完之后不会停止，除非是精确匹配（=）,所以这个时候就会进入正则匹配，正则匹配 ~image 只要查找的uri有 image就可以匹配上，这个时候正则就会覆盖掉之前普通匹配的内容，所以这个时候就会显示image目录下面的内容，所以这个时候结果就是 默认的index.html ---image-html

测试2：



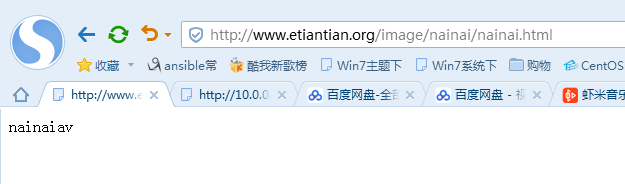
原因大致一样，到了正则把这个覆盖掉，然后找到index.htm 然后停止匹配。结果返回 image-htm

测试3：



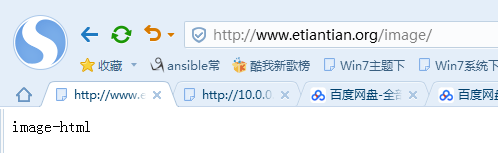
基本上前面一样，到了 image之后，匹配到了之后就会停止匹配。这个时候找到image.nainai可以找到是一个目录，但是这个时候是一个目录，不是内容就会报错，所以就需要下面的样子一致匹配具体的文件。

测试4：



所以结论 普通如果是 /image 那么结果还是正则的 /image的结果，配置文件我就不该了

看结果



#### 案例2

配置1：

[root@web01 foo]# pwd

/opt/foo

[root@web01 foo]# ll

total 8

-rw-r--r-- 1 root root 7 May 20 00:54 foo.html

-rw-r--r-- 1 root root 7 May 20 00:55 index.html

[root@web01 foo]# cat index.html

foo123

[root@web01 foo]#

配置其他的不变，参考上面的

配置文件

orker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

location /image {

root html/www;

index index.html index.htm;

}

location /foo/ {

root /opt/;

index index.html index.htm;

}

location ~\* /image/ {

root /opt/;

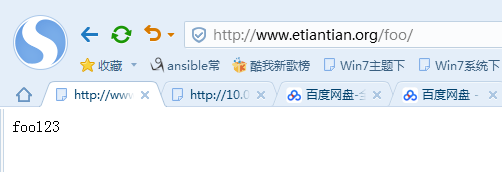
index index.html index.htm;

}

}

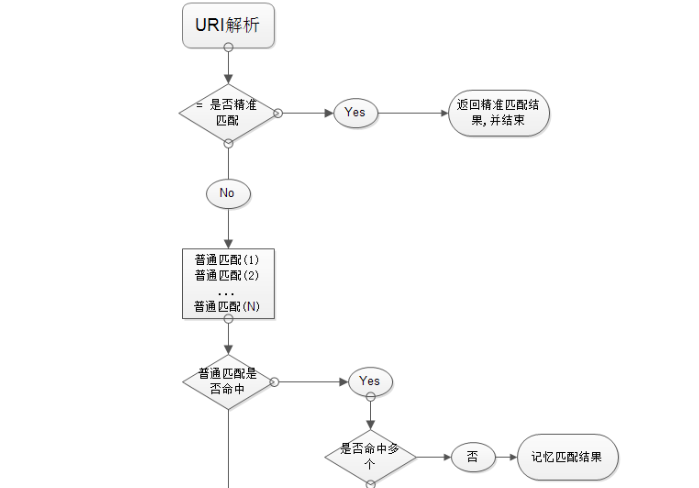
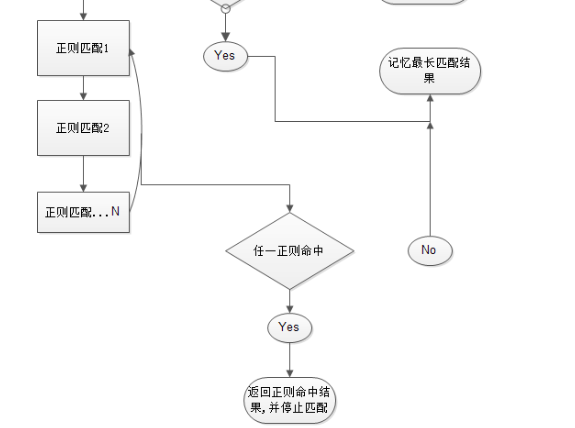
}

测试



说明：首先 [www.etiantian.org/fool](http://www.etiantian.org/fool) 的时候，匹配的只有 location / 和location /fool 在普通匹配的时候采取最长匹配原则，所以这个时候就匹配到了 /foo 下面的内容 foo123

### location定位整个运行流程

、

具体流程是这样子的：

第一步，先看是否有精准匹配（所谓的精准匹配意思就是输入到浏览器的uri和location =/uri 完全一模一样）

如果有看是否匹配，如果匹配精准的uri正确，那么就会停止匹配，返回精准匹配的结果。

第二步，如果没有精准匹配（没有=）或者是精准匹配的uri和输入浏览器的uri不一样，那么就进入普通的匹配环节。

第三步增加，当进入普通匹配的时候，这个时候需要查看是否设置了 ^~ （不做正则表达式检查，优先匹配）如果结果匹配了 ^~的结果，那么就会停止匹配，返回这个不匹配正则lcoation的结果

第四步：进入普通匹配的时候 按照最长匹配原则（既如果个时候输入到浏览器的uri是 oldboy/oldgrl/AV/ngg, 那么这个时候 普通的location 指定的uri 分别是 location / location /oldboy location /oldboy/oldgrl. 以及 location/oldboy/oldgril/A 那么匹配的结果是 /location/oldboy/oldgrl 对应返回的结果（这个结果只是暂时保存））来进行匹配，匹配出来的结果，暂时保存，然后去正则匹配的location区域查找，（正则location的区域是按照写的顺序，从上到下依次匹配）一旦正则匹配到了指定的结果，那么就不会往下再次匹配了，就会覆盖普通匹配的结果，返回正则第一个匹配的结果。

### 老师讲课资料统一测试

配置文件：其他环境都有

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

location / {

return 401;

}

location = / {

return 402;

}

location /AV/ {

return 403;

}

location ^~ /images/ {

return 301;

}

location ~\* \.（gif|jpg|jpeg）$ {

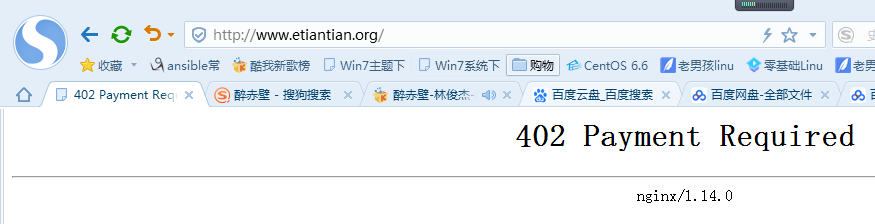
return 500;

}

}

}

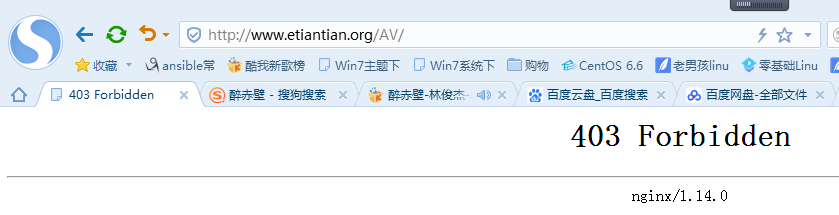
测试1：



原因解释：首先输入[www.etianian.org默认的uri](http://www.etianian.org默认的uri)是 / 那么这个时候，那个时候首先就会被 location =/匹配到。 然后没有路径，也没有文件，直接来一个return 402，那么直接就会返回402了，如果是一个index.html文件的话。那么他机会变成 [www.etiantian.org/index.html](http://www.etiantian.org/index.html) 继续去寻找了、

这里return 是rewrite 语法里面的特殊字段，类似于break. 如果是break直接就终止，这里return 直接返回 retrun后面的数值。

测试2：



这里面还是老样子 直接uri是/AV/那么就会最大原则，找打 location /AV/ 这个uri。但是这个时候直接来一个 return 403 所以就把你阻断了。所以就返回了403

测试3：



当输入这么多uri的时候唯一匹配的就是 location / 这个时候因为是目录，那么他就会继续往他对应下面去匹配对应的文件，但是半路被return 拦截了，所以就会返回 401

### 官方文档解释

官方文档解释

REFER:  <http://wiki.nginx.org/NginxHttpCoreModule#location>

location

syntax: *location [=|~|~\*|^~|@] /uri/ { ... }*

default: *no*

context: *server*

This directive allows different configurations depending on the URI.

（译者注：1 、different configurations depending on the URI 说的就是语法格式：location [=|~|~\*|^~|@] /uri/ { ... } ，依据不同的前缀“= ”，“^~ ”，“~ ”，“~\* ”和不带任何前缀的（因为[A] 表示可选，可以不要的），表达不同的含义, 简单的说尽管location 的/uri/ 配置一样，但前缀不一样，表达的是不同的指令含义。2 、查询字符串不在URI范围内。例如：/films.htm?fid=123 的URI 是/films.htm 。）

It can be configured using both literal strings and regular expressions. To use regular expressions, you must use a prefix:

"~" for case sensitive matching

"~\*" for case insensitive matching

译文：上文讲到location /uri/ 可通过使用不同的前缀，表达不同的含义。对这些不同前缀，分下类，就2 大类：正则location ，英文说法是location using regular expressions 和普通location ，英文说法是location using literal strings 。那么其中“~ ”和“~\* ”前缀表示正则location ，“~ ”区分大小写，“~\* ”不区分大小写；其他前缀（包括：“=”，“^~ ”和“@ ”）和无任何前缀的都属于普通location 。

To determine which *location* directive matches a particular query, the literal strings are checked first.

译文：对于一个特定的 HTTP 请求（ a particular query ）， nginx 应该匹配哪个 location 块的指令呢（注意：我们在 nginx.conf 配置文件里面一般会定义多个 location 的）？匹配 规则是：先匹配普通location （再匹配正则表达式）。注意：官方文档这句话就明确说了，先普通location ，而不是有些同学的误区“先匹配正则location ”。

Literal strings match the beginning portion of the query - the most specific match will be used.

前面说了“普通location ”与“正则location ”之间的匹配规则是：先匹配普通location ，再匹配正则location 。那么，“普通location ”内部（普通location 与普通location ）是如何匹配的呢？简单的说：最大前缀匹配。原文：1、match the beginning portion of the query （说的是匹配URI 的前缀部分beginning portion ）； 2 、the most specific match will be used （因为location 不是“严格匹配”，而是“前缀匹配”，就会产生一个HTTP 请求，可以“前缀匹配”到多个普通location ，例如：location /prefix/mid/ {} 和location /prefix/ {} ，对于HTTP 请求/prefix/mid/t.html ，前缀匹配的话两个location 都满足，选哪个？原则是：the most specific match ，于是选的是location /prefix/mid/ {} ）。

Afterwards, regular expressions are checked in the order defined in the configuration file. The first regular expression to match the query will stop the search.

这段话说了两层意思，第一层是：“Afterwards, regular expressions are checked ”, 意思是普通location 先匹配，而且选择了最大前缀匹配后，不能就停止后面的匹配，最大前缀匹配只是一个临时的结果，nginx 还需要继续检查正则location （但至于最终才能普通location 的最大前缀匹配，还是正则location 的匹配，截止当前的内容还没讲，但后面会讲）。第二层是“regular expressions are checked in the order defined in the configuration file. The first regular expression to match the query will stop the search. ”，意思是说“正则location ”与“正则location”内部的匹配规则是：按照正则location 在配置文件中的物理顺序（编辑顺序）匹配的（这句话就说明location 并不是一定跟顺序无关，只是普通location 与顺序无关，正则location 还是与顺序有关的），并且只要匹配到一条正则location ，就不再考虑后面的（这与“普通location ”与“正则location ”之间的规则不一样，“普通location ”与“正则location ”之间的规则是：选择出“普通location ”的最大前缀匹配结果后，还需要继续搜索正则location ）。

If no regular expression matches are found, the result from the literal string search is used.

这句话回答了“普通location ”的最大前缀匹配结果与继续搜索的“正则location ”匹配结果的决策关系。如果继续搜索的“正则location ”也有匹配上的，那么“正则location ”覆盖 “普通location ”的最大前缀匹配（因为有这个覆盖关系，所以造成有些同学以为正则location 先于普通location 执行的错误理解）；但是如果“正则location ”没有能匹配上，那么就用“普通location ”的最大前缀匹配结果。

For case insensitive operating systems, like Mac OS X or Windows with Cygwin, literal string matching is done in a case insensitive way (0.7.7). However, comparison is limited to single-byte locale's only.

Regular expression may contain captures (0.7.40), which can then be used in other directives.

It is possible to disable regular expression checks after literal string matching by using "^~" prefix.If the most specific match literal location has this prefix: regular expressions aren't checked.

通常的规则是，匹配完了“普通location ”指令，还需要继续匹配“正则location ”，但是你也可以告诉Nginx ：匹配到了“普通location ”后，不再需要继续匹配“正则location ”了，要做到这一点只要在“普通location ”前面加上“^~ ”符号（^ 表示“非”，~ 表示“正则”，字符意思是：不要继续匹配正则）。

By using the "=" prefix we define the exact match between request URI and location. When matched search stops immediately. E.g., if the request "/" occurs frequently, using "location = /" will speed up processing of this request a bit as search will stop after first comparison.

除了上文的“^~ ”可以阻止继续搜索正则location 外，你还可以加“= ”。那么如果“^~ ”和“= ”都能阻止继续搜索正则location 的话，那它们之间有什么区别呢？区别很简单，共同点是它们都能阻止继续搜索正则location ，不同点是“^~ ”依然遵守“最大前缀”匹配规则，然而“= ”不是“最大前缀”，而是必须是严格匹配（exact match ）。

这里顺便讲下“location / {} ”和“location = / {} ”的区别，“location / {} ”遵守普通location 的最大前缀匹配，由于任何URI 都必然以“/ ”根开头，所以对于一个URI ，如果有更specific 的匹配，那自然是选这个更specific 的，如果没有，“/ ”一定能为这个URI 垫背（至少能匹配到“/ ”），也就是说“location / {} ”有点默认配置的味道，其他更specific的配置能覆盖overwrite 这个默认配置（这也是为什么我们总能看到location / {} 这个配置的一个很重要的原因）。而“location = / {} ”遵守的是“严格精确匹配exact match ”，也就是只能匹配 http://host:port/ 请求，同时会禁止继续搜索正则location 。因此如果我们只想对“GET / ”请求配置作用指令，那么我们可以选“location = / {} ”这样能减少正则location 的搜索，因此效率比“location / {}” 高（注：前提是我们的目的仅仅只想对“GET / ”起作用）。

On exact match with literal location without "=" or "^~" prefixes search is also immediately terminated.

前面我们说了，普通location 匹配完后，还会继续匹配正则location ；但是nginx 允许你阻止这种行为，方法很简单，只需要在普通location 前加“^~ ”或“= ”。但其实还有一种“隐含”的方式来阻止正则location 的搜索，这种隐含的方式就是：当“最大前缀”匹配恰好就是一个“严格精确（exact match ）”匹配，照样会停止后面的搜索。原文字面意思是：只要遇到“精确匹配exact match ”，即使普通location 没有带“= ”或“^~ ”前缀，也一样会终止后面的匹配。

先举例解释下，后面例题会用实践告诉大家。假设当前配置是：location /exact/match/test.html { 配置指令块1}，location /prefix/ { 配置指令块2} 和 location ~ \.html$ { 配置指令块3} ，如果我们请求 GET /prefix/index.html ，则会被匹配到指令块3 ，因为普通location /prefix/ 依据最大匹配原则能匹配当前请求，但是会被后面的正则location 覆盖；当请求GET /exact/match/test.html ，会匹配到指令块1 ，因为这个是普通location 的exact match ，会禁止继续搜索正则location 。

To summarize, the order in which directives are checked is as follows:

Directives with the "=" prefix that match the query exactly. If found, searching stops.

All remaining directives with conventional strings. If this match used the "^~" prefix, searching stops.

Regular expressions, in the order they are defined in the configuration file.

If #3 yielded a match, that result is used. Otherwise, the match from #2 is used.

这个顺序没必要再过多解释了。但我想用自己的话概括下上面的意思“正则 location 匹配让步普通location 的严格精确匹配结果；但覆盖普通 location 的最大前缀匹配结果”。

It is important to know that nginx does the comparison against decoded URIs. For example, if you wish to match "/images/%20/test", then you must use "/images/ /test" to determine the location.

在浏览器上显示的URL 一般都会进行URLEncode ，例如“空格”会被编码为%20 ，但是Nginx 的URL 的匹配都是针对URLDecode 之后的。也就是说，如果你要匹配“/images/%20/test ”，你写location 的时候匹配目标应该是：“/images/ /test ”。

Example:

[location](http://wiki.nginx.org/NginxHttpCoreModule#location)  = / {

  # matches the query / only.

  [ configuration A ]

}

[location](http://wiki.nginx.org/NginxHttpCoreModule#location)  / {

  # matches any query, since all queries begin with /, but regular

  # expressions and any longer conventional blocks will be

  # matched first.

  [ configuration B ]

}

[location](http://wiki.nginx.org/NginxHttpCoreModule#location)^~ /images/ {

    # matches any query beginning with /images/ and halts searching,

  # so regular expressions will not be checked.

  [ configuration C ]

}

[location](http://wiki.nginx.org/NginxHttpCoreModule#location)~\* \.(gif|jpg|jpeg)$ {

  # matches any request ending in gif, jpg, or jpeg. However, all

  # requests to the /images/ directory will be handled by

  # Configuration C.

  [ configuration D ]

}

上述这4 个location 的配置，没什么好解释的，唯一需要说明的是location / {[configuration B]} ，原文的注释严格来说是错误的，但我相信原文作者是了解规则的，只是文字描述上简化了下，但这个简化容易给读者造成“误解：先检查正则location ，再检查普通location ”。原文：“matches any query, since all queries begin with /, butregular expressions and any longer conventional blocks will be matched first. ”大意是说：“location / {} 能够匹配所有HTTP 请求，因为任何HTTP 请求都必然是以‘/ ’开始的（这半句没有错误）。但是，正则location 和其他任何比‘/ ’更长的普通location （location / {} 是普通location 里面最短的，因此其他任何普通location 都会比它更长，当然location = / {} 和 location ^~ / {} 是一样长的）会优先匹配（matched first ）。” 原文作者说“ but regular expressions will be matched first. ”应该只是想说正则 location 会覆盖这里的 location / {} ，但依然是普通location / {} 先于正则 location 匹配，接着再正则 location 匹配；但其他更长的普通 location （ any longer conventional blocks ）的确会先于 location / {} 匹配。

Example requests:

/ -> configuration A

/documents/document.html -> configuration B

/images/1.gif -> configuration C

/documents/1.jpg -> configuration D

Note that you could define these 4 configurations in any order and the results would remain the same.

需要提醒下：这里说“in any order ”和“… remain the same ”是因为上面只有一个正则location 。文章前面已经说了正则location 的匹配是跟编辑顺序有关系的。

While nested locations are allowed by the configuration file parser, their use is discouraged and may produce unexpected results.

实际上 nginx 的配置文件解析程序是允许 location 嵌套定义的（ location / { location /uri/ {} } ）。但是我们平时却很少看见这样的配置，那是因为 nginx 官方并不建议大家这么做，因为这样会导致很多意想不到的后果。

The prefix "@" specifies a named location. Such locations are not used during normal processing of requests, they are intended only to process internally redirected requests (see [error\_page](http://wiki.nginx.org/HttpCoreModule#error_page),[try\_files](http://wiki.nginx.org/HttpCoreModule#try_files)).

文章开始说了location 的语法中，可以有“= ”，“^~ ”，“~ ”和“~\* ”前缀，或者干脆没有任何前缀，还有“@ ”前缀，但是后面的分析我们始终没有谈到“@ ”前缀。文章最后点内容，介绍了“＠”的用途：“@ ”是用来定义“Named Location ”的（你可以理解为独立于“普通location （location using literal strings ）”和“正则location （location using regular expressions ）”之外的第三种类型），这种“Named Location ”不是用来处理普通的HTTP 请求的，它是专门用来处理“内部重定向（internally redirected ）”请求的。注意：这里说的“内部重定向（internally r

edirected ）”或许说成“forward ”会好点，以为内internally redirected 是不需要跟浏览器交互的，纯粹是服务端的一个转发行为。

## rewrite

将一个域名的地址进行重写。

通过rewrite的重写实现访问域名的跳转

可以实现网站页面信息的伪静态处理

[www.odlboy.ocm/baoming?oldboy&etiantian.js](http://www.odlboy.ocm/baoming?oldboy&etiantian.js)

[www.oldboy.com/baoming-oldboy\_etiantian.js](http://www.oldboy.com/baoming-oldboy_etiantian.js)

Syntax: rewrite regex replacement [flag]; ---rewrite 使用格式

rewrite 正则表达式（per语言） 替换的内容 标签的信息

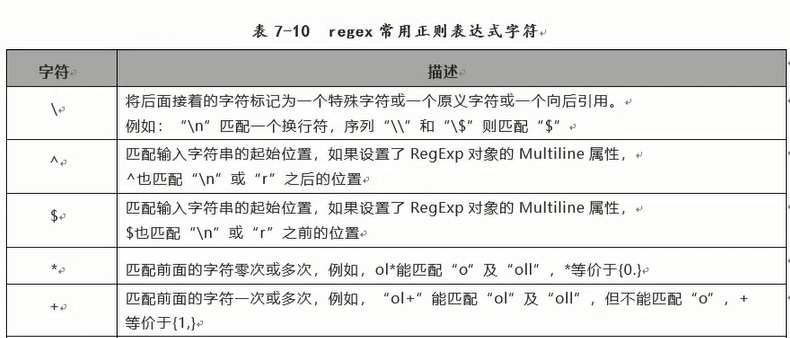
Default: —

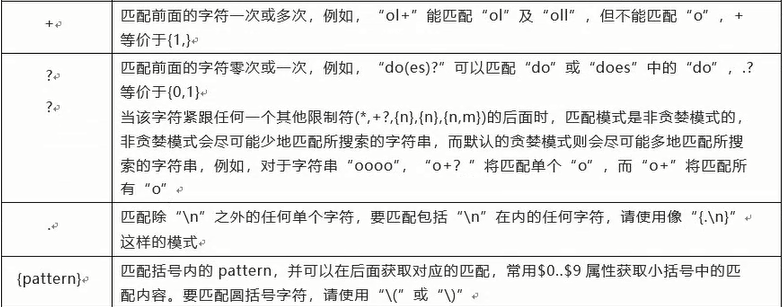
Context: server, location, if rewrite 可以用的模块

### rewrite语法标签分类

|  |  |
| --- | --- |
| flag标记符号 | 说明 |
| last | 本条规则匹配完成后，继续向下匹配新的location URL规则 |
| break | 本条规则匹配完成之后，不在匹配后面的任何股则 |
| redirect | 返回302临时重定向，浏览器地址栏会显示跳转后的URL |
| permanent | 返回301永久重定向，浏览器地址栏会显示跳转后的URL地址 |

### rewrite 语法正则表达式常用字符





### 实现输入etiantian.org域名的时候自动会跳转到[www.etiantian.org--rewrite、](http://www.etiantian.org--rewrite、)

**注意事项1.域名解析2.解析的目录和文件存在**

#### 书写rewrite功能跳转代码的书写

rewrite ^/(.\*) http://[www.etiantian.org/$1](http://www.etiantian.org/$1) permanent;

解释： ^/ uri前面的的内容都会进行跳转成为 [www.etiantian.org](http://www.etiantian.org)

(.\*) 代表保留 uri的内容

http://[www.etiantian.org](http://www.etiantian.org) 代表输入到网站的域名之后，然后跳转成为的域名,这里备注一定是浏览器跳转的全路径，必须是 http:// 开头 切记。

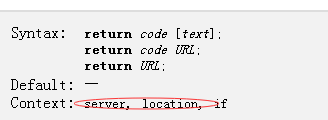
/$1 反向引用 意思是将uri原封不动的输出

permanent 代表永久跳转，并且返回跳转之后的uri

; 代表的是分割符号。

#### 编写配置文件

1. 查看官方文档，查看跳转信息存放的地方



1. 编写配置文件

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;`

keepalive\_timeout 65;

**server {**

**listen 80;**

**server\_name etiantian.org;**

**rewrite ^/(.\*) http://www.etiantian.org/$1 permanent;**

**}**

server {

listen 80;

server\_name www.etiantian.org;

location / {

root html/www;

index index.html index.htm;

}

}

}

-------------备注解释--------------

**server {**

**listen 80;**

**server\_name etiantian.org;**

**rewrite ^/(.\*) http://www.etiantian.org/$1 permanent;**

**}**

**备注：这里面如果把rewrite这条命令放到下面server区块里面的话，那么他就会被变成一个死循环，也就是说当你在命令行里面输入 etianitan.org的时候，这个时候他就会被自动跳转到** [**www.etiantian.org**](http://www.etiantian.org) **这个域名，（简单理解就是如果单纯的输入的 域名后面就跟一个/的话，那么还是需要以文件支撑的，这个时候正好有跳转功能（内部跳转），所以优先找到这个 rewrite重写了，然后和之前一样进行第二次的访问）这个时候就会以** [**www.etiantian.org**](http://www.etiantian.org)**去访问。 然后就会是一个死循环，但是单独拿出来组成一个server区块就不会了。**

**所以单独拿出来的时候，一个server 区块， listen server\_name rewrite 就可以了，后面就常规操作。**

server {

listen 80;

server\_name www.etiantian.org;

location / {

root html/www;

index index.html index.htm;

}

}

3.检查语法

[root@web01 ~]# nginx -t

nginx: the configuration file /application/nginx-1.14/conf/nginx.conf syntax is ok

nginx: configuration file /application/nginx-1.14/conf/nginx.conf test is successful

[root@web01 ~]#

#### 平滑重启

[root@web01 ~]# nginx -s reload

[root@web01 ~]#

#### 测试



测试成功

在linux里面测试

-L (location 位置定位的意思 -v verbose 显示详细信息的意思) 所以- Lv就是显示详细的定位信息的意思。

[root@web01 conf]# curl -Lv etiantian.org --命令

\* About to connect() to etiantian.org port 80 (#0) --域名解析过程

\* Trying 172.16.1.7... connected --域名解析过程

\* Connected to etiantian.org (172.16.1.7) port 80 (#0) --域名解析过程

> GET / HTTP/1.1 --get方法请求行 http协议版本号是 1.1 长连接

> User-Agent: curl/7.19.7 (x86\_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.21 Basic ECC zlib/1.2.3 libidn/1.18 libssh2/1.4.2 ---请求头信息 ：用户代理代表的是用户是以什么身份访问的

> Host: etiantian.org ---请求头信息：请求的主机域名是 etiantian.org

> Accept: \*/\* ----- 接受的...

> 空行

因为get方法没有报文主体，所以就是直接到相应报文

< HTTP/1.1 301 Moved Permanently --相应报文的起始行， http/1.1 代表的是版本号 301 代表永久跳转， permanently 代表的是nginx语法的标签 ）（他代表的意思就是返回301永久跳转））

< Server: nginx/1.14.0 -- 相应报文的相应头信息 代表的是你访问的服务器是用的nginx 1.14版本的服务

< Date: Sun, 20 May 2018 12:28:20 GMT -----相应的时间

< Content-Type: text/html ------ content-type 用户浏览器加载数据的格式 ，所以这里的意思就是web服务器加载数据格式是 text/html格式

< Content-Length: 185 代表浏览器加载数据的长度是 185

< Connection: keep-alive --代表保持长连接的意思

< Location: <http://www.etiantian.org/> ---跳转后的域名地址是 http://www.etiantian.org

< ------------空行

\* Ignoring the response-body 进行第二次DNS的解析

\* Connection #0 to host etiantian.org left intact

\* Issue another request to this URL: 'http://www.etiantian.org/'

\* About to connect() to www.etiantian.org port 80 (#1)

\* Trying 172.16.1.7... connected

\* Connected to www.etiantian.org (172.16.1.7) port 80 (#1) 进行第二次DNS的解析

> GET / HTTP/1.1 --请求行 （get 方法 版本）

> User-Agent: curl/7.19.7 (x86\_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.21 Basic ECC zlib/1.2.3 libidn/1.18 libssh2/1.4.2 （请求头 用户代理,就是代表用户用什么东西来访问客户端）

> Host: [www.etiantian.org](http://www.etiantian.org) ---这次用户是以 [www.etiantian.org](http://www.etiantian.org)的域名来访问服务我

> Accept: \*/\*

> 空行

因为get方法没有报文主体，所以就是直接到相应报文

< HTTP/1.1 200 OK -----相应的起始行 200代表相应成功，服务器成功返回网页

< Server: nginx/1.14.0 相应头（服务器的一些信息）

< Date: Sun, 20 May 2018 12:28:20 GMT 相应的时间

< Content-Type: text/html ------ content-type 用户浏览器加载数据的格式 ，所以这里的意思就是web服务器加载数据格式是 text/html格式

< Content-Length: 27 代表浏览器加载数据的长度是27

< Last-Modified: Thu, 17 May 2018 01:46:43 GMT --时间

< Connection: keep-alive --服务器保持长连接

< ETag: "5afcdf03-1b"

< Accept-Ranges: bytes 接受的范围是字节

< --空格

10.0.0.7 [www.etiantian.org](http://www.etiantian.org) 相应的主体内容

\* Connection #1 to host www.etiantian.org left intact --断开连接

\* Closing connection #0 --断开连接

\* Closing connection #1 --断开连接

[root@web01 conf]#

[root@web01 conf]#

### 实现输入etiantian.org域名时自动跳转 [www.etiantian.org](http://www.etiantian.org)

**注意事项1.域名解析2.解析的目录和文件存在**

#### 书写if条件判断的代码

if ($host ~\* etiantian.org) {

**rewrite ^/(.\*) http://www.etiantian.org/$1 permanent;**

}

--------------备注说明---------------

1. if 判断具体格式见后面格式 简单理解就是：if 里面的条件成立了，那么就会执行大括号里面放的内容
2. $host 就是主机名，域名的意思
3. ~\* 代表模糊匹配的意思，不区分大小写
4. etiantian.org 代表如果你在浏览器里面输入的etiantian.org 那么就会跳转
5. rewrite 命令
6. ^/ 代表匹配域名
7. （.\*）正则表达式 代表匹配域名后面的所有东西
8. [**http://www.etiantian.org/**](http://www.etiantian.org/) **代表即将要跳转的域名**
9. **$1 后项引用，代表取出签名（.\*）的内容**
10. **permanent 永久跳转**
11. **; 分割符号**

#### 编写配置文件

**配置文件编写**

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

location / {

root html/www;

index index.html index.htm;

if ( $host ~\* ^etiantian.org$) {

rewrite ^/(.\*) http://www.etiantian.org/$1 permanent;

}

}

}

}

**语法检查**

[root@web01 ~]# nginx -t

nginx: the configuration file /application/nginx-1.14/conf/nginx.conf syntax is ok

nginx: configuration file /application/nginx-1.14/conf/nginx.conf test is successful

[root@web01 ~]#

#### 平滑重启

[root@web01 ~]# nginx -s reload

[root@web01 ~]#

#### 测试检查



测试成功

### 重写中用到的指令

重写中用到的指令

if (条件) {} 设定条件，再进行重写

set #设置变量

return 返回状态码

break#跳出rewrite

rewrite #重写

### rewrit重写的if条件

基本格式

if (条件) {

重写模式

}

注意：if 和条件中间必须有空格

条件有哪些？

1. = 来判断相等，用于字符串比较 2. ~ 用正则来匹配（属于模糊匹配，区分大小写） 3.~\* 不区分大小写模糊匹配的正则 4. -f -d -e 判断是否是文件 是否是目录 是否存在

### 练习题目

1. 实现rewrite跳转功能，用户访问[www.taobao.com](http://www.taobao.com)自动跳转为 [www.etiantian.org](http://www.etiantian.org)

第一个脚本如下：域名和首页站点的目录和文件都需要存在

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.etiantian.org;

location / {

root html/www;

index index.html index.htm;

if ( $host ~\* ^www.taobao.com$) {

rewrite ^/(.\*) http://www.etiantian.org/$1 permanent;

}

}

}

}

第二个脚本如下：

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.taobao.com;

rewrite ^/(.\*) http://www.etiantian.org/$1 permanent;

}

server {

listen 80;

server\_name www.etiantian.org;

location / {

root html/www;

index index.html index.htm;

}

}

}

实现rewrite跳转功能，

用户访问www.taobao.com/av/oldboy.html ---> www.etiantian.org/AV/oldboy.html

用户访问www.taobao.com/bv/oldboy.html 不做任何跳转，直接访问

第一问题的脚本：

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.taobao.com;

location /AV/ {

rewrite ^/(.\*) http://www.etiantian.org/$1 permanent;

}

root html/taobao;

index index.html;

}

server {

listen 80;

server\_name www.etiantian.org;

location ~\* /AV/ {

root html/www;

index index.html;

}

}

}

第二问题的脚本：

worker\_processes 2;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server {

listen 80;

server\_name www.taobao.com;

location !~\*/bv/ {

rewrite ^/(.\*) http://www.etiantian.org/$1 permanent;

}

root html/taobao;

index index.html;

}

server {

listen 80;

server\_name www.etiantian.org;

location / {

root html/www;

index index.html;

}

}

}

## nginx的访问认证

在实际工作中，企业要求我们为网站设置访问账号和密码的权限，这样操作后，只用拥有账号和密码的用户才可以访问网站，这种使用账号和密码才可以访问的网站功能主要应用于企业内部人员。

具体配置如下：

Authentication 简写 auth 认证的意思

auth\_basic "web auth"; --代表的是认证的一个描述说明

auth\_basic\_user\_file conf/htpasswd; --htppasswd生成密文的信息

### nginx访问认证实战

#### 第一个里程碑编写配置文件

[root@web01 other]# pwd

/application/nginx/conf/extra/other

[root@web01 other]# vim bbs.conf

server {

listen 80;

server\_name bbs.etiantian.org;

root html/bbs;

index index.html index.htm;

auth\_basic "web auth";

auth\_basic\_user\_file /application/nginx/conf/htpasswd;

}

#### 第二个里程：实现密码文件种的密码信息显示为密文

第一步：

yum install -y httpd-tools

第二步：

htpasswd -bc /application/nginx/conf/htpasswd oldboy 123456

第三步

chmod 400 /application/nginx/conf/htpasswd

第四步

chown www /application/nginx/conf/htpasswd

--------解释备注：

第一步就是安装生成密码文件的软件包

第二步就是利用htpasswd命令 执行生成密码的文件 然后后面跟上用户名和密码

第三步

给密码文件授权一般授权的是600，这里授权400 也可以授权600

第四步：

将密码文件授权nginx的虚拟用户www去管理

#### 第三个里程碑：重启nginx服务，进行访问测试

nginx -t

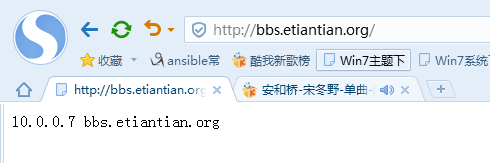
nginx -s reload

[root@web01 conf]# htpasswd -bc /application/nginx/conf/htpasswd oldboy 123456

Adding password for user oldboy

[root@web01 conf]# curl -u oldboy:123456 bbs.etiantian.org

10.0.0.7 bbs.etiantian.org



#### 密码认证原理

客户端访问的时候最终访问的是web服务器，所以用户在访问web服务器之前所以就需要转变为www用户，另外需要说明用户i名字和密码，所以就是为什么要授权www 权限600

