

概要:

1. Nginx 简介
2. Nginx 架构说明
3. Nginx 基础配置与使用

一、Nginx 简介与安装

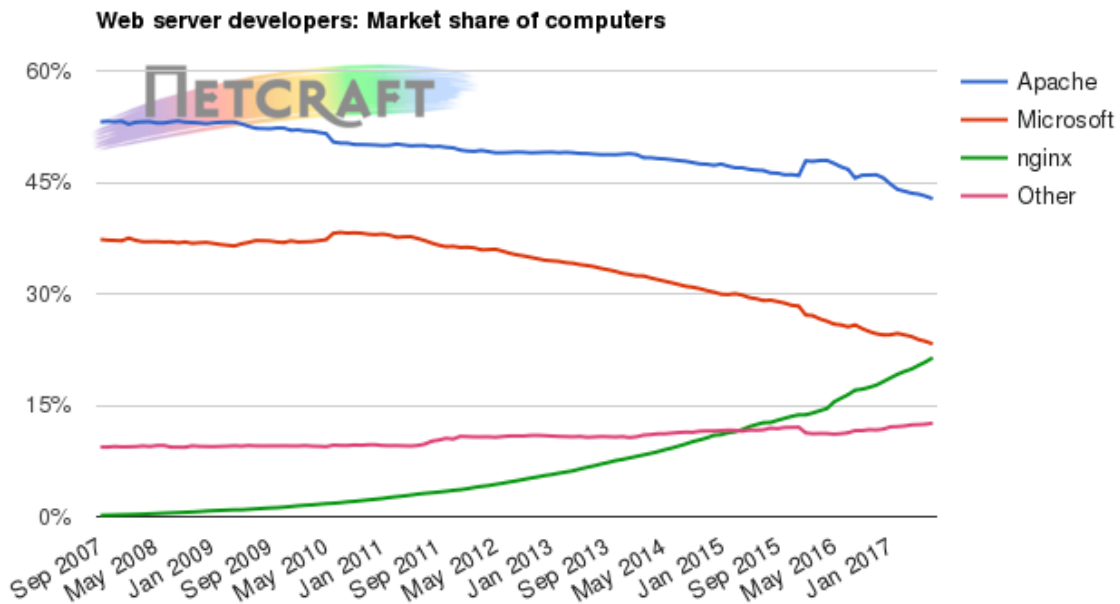
知识点:

1. Nginx 简介
2. Nginx 编译与安装

1、Nginx简介:

Nginx是一个高性能WEB服务器，除它之外Apache、Tomcat、Jetty、IIS，它们都是Web服务器，或者叫做WWW（World Wide Web）服务器，相应地也都具备Web服务器的基本功能。Nginx 相对其它WEB服务有什么优势呢？

1. Tomcat、Jetty 面向java语言，先天就是重量级的WEB服务器，其性能与Nginx没有可比性。
2. IIS只能在Windows操作系统上运行。Windows作为服务器在稳定性与其他一些性能上都不如类UNIX操作系统，因此，在需要高性能Web服务器的场合下IIS并不占优。
3. Apache的发展时期很长，而且是目前毫无争议的世界第一大Web服务器，其有许多优点，如稳定、开源、跨平台等，但它出现的时间太长了，在它兴起的年代，互联网的产业规模远远比不上今天，所以它被设计成了一个重量级的、不支持高并发的Web服务器。在Apache服务器上，如果有数以万计的并发HTTP请求同时访问，就会导致服务器上消耗大量内存，操作系统内核对成百上千的Apache进程做进程间切换也会消耗大量CPU资源，并导致HTTP请求的平均响应速度降低，这些都决定了Apache不可能成为高性能Web服务器，这也促使了Lighttpd和Nginx的出现。下图可以看出07年到17 年强劲增长势头。



Developer	June 2017	Percent	July 2017	Percent	Change
Apache	2,813,543	43.13%	2,821,008	42.78%	-0.35
Microsoft	1,538,763	23.59%	1,532,351	23.24%	-0.35
nginx	1,358,510	20.83%	1,410,658	21.39%	0.57

2、编译与安装

安装环境准备：

(1) linux 内核2.6及以上版本:

只有2.6之后才支持epool，在此之前使用select或pool多路复用的IO模型，无法解决高并发压力的问题。通过命令uname -a 即可查看。epool的性能强劲。

```
#查看 linux 内核
uname -a
```

(2) GCC编译器

GCC (GNU Compiler Collection) 可用来编译C语言程序。Nginx不会直接提供二进制可执行程序,只能下载源码进行编译。

(3) PCRE库

PCRE (Perl Compatible Regular Expressions, Perl兼容正则表达式) 是由Philip Hazel开发的函数库，目前为很多软件所使用，该库支持正则表达式。

(4) zlib库

zlib库用于对HTTP包的内容做gzip格式的压缩，如果我们在nginx.conf里配置了gzip on，并指定对于某些类型 (content-type) 的HTTP响应使用gzip来进行压缩以减少网络传输量。

(5) OpenSSL开发库

如果我们的服务器不只是要支持HTTP，还需要在更安全的SSL协议上传输HTTP，那么就需要拥有OpenSSL了。另外，如果我们想使用MD5、SHA1等散列函数，那么也需要安装它。上面几个库都是Nginx 基础功能所必需的，为简单起见我们可以通过yum 命令统一安装。

```
#yum 安装nginx 环境
yum -y install make zlib zlib-devel gcc-c++ libtool openssl openssl-devel pcre
pcre-devel
```

源码获取:

nginx 下载页: <http://nginx.org/en/download.html>。

```
# 下载nginx 最新稳定版本
wget http://nginx.org/download/nginx-1.14.0.tar.gz
#解压
tar -zxvf nginx-1.14.0.tar.gz
```

需要注意的时候, 官网有三个版本: 主线版本, 稳定版本, 历史版本, 部署的时候要用稳定版本。学习中使用1.14版本。

最简单的安装:

```
# 全部采用默认安装
./configure
make && make install
```

执行完成之后 nginx 运行文件 就会被安装在 /usr/local/nginx 下。

基于参数构建, 加入默认模块之外的一些模块。

```
./configure --prefix=/usr/local/nginx --with-http_stub_status_module --with-
http_ssl_module --with-debug
```

控制命令:

```
#默认方式启动:
./sbin/nginx
#指定配置文件启动
./sbin/nginx -c /tmp/nginx.conf
#指定nginx程序目录启动
./sbin/nginx -p /usr/local/nginx/

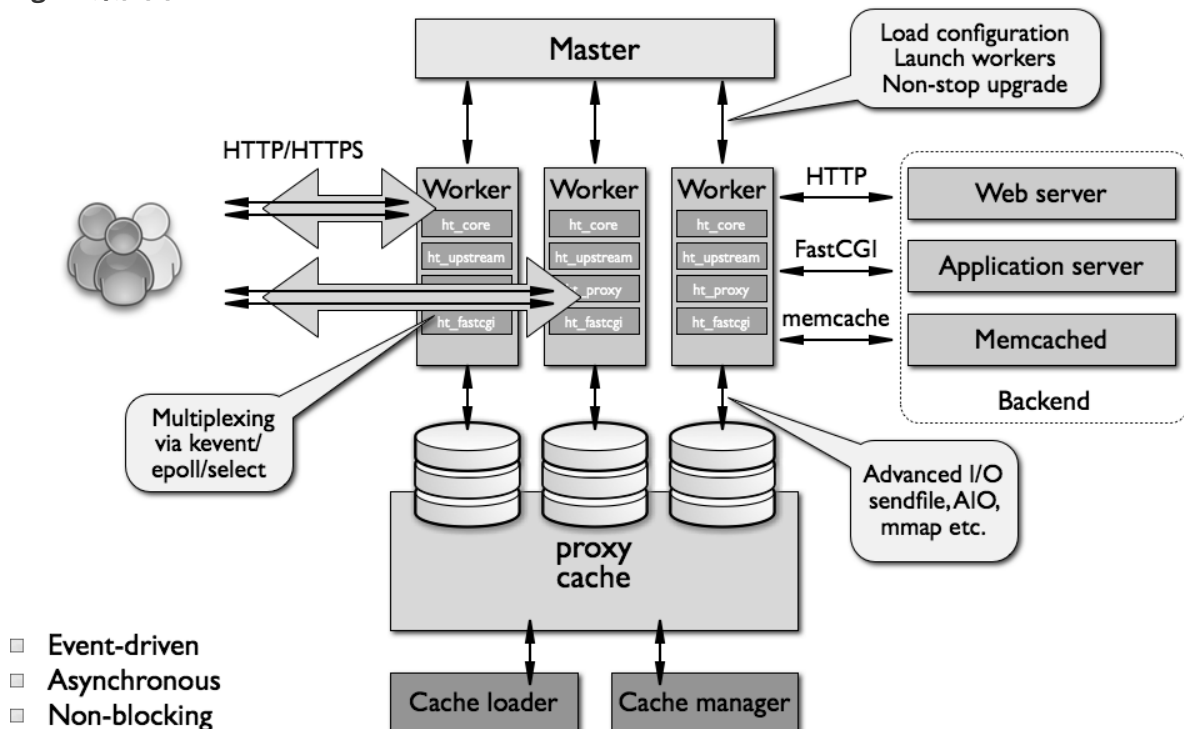
#快速停止
./sbin/nginx -s stop
#优雅停止
./sbin/nginx -s quit

# 检测配置文件是否符合语法
./sbin/nginx -t
# 热装载配置文件, 不重启nginx来热更新
./sbin/nginx -s reload
# 重新打开日志文件, 是根据文件名去打开, 因为nginx拿到的是句柄, 你在使用过程中改变文件名, 然后新建日志文件, 不
# 会改变nginx写入日志的位置。
./sbin/nginx -s reopen
```

启动之后如何测试: curl 127.0.0.1 返回界面就说明没问题

二、Nginx 架构说明

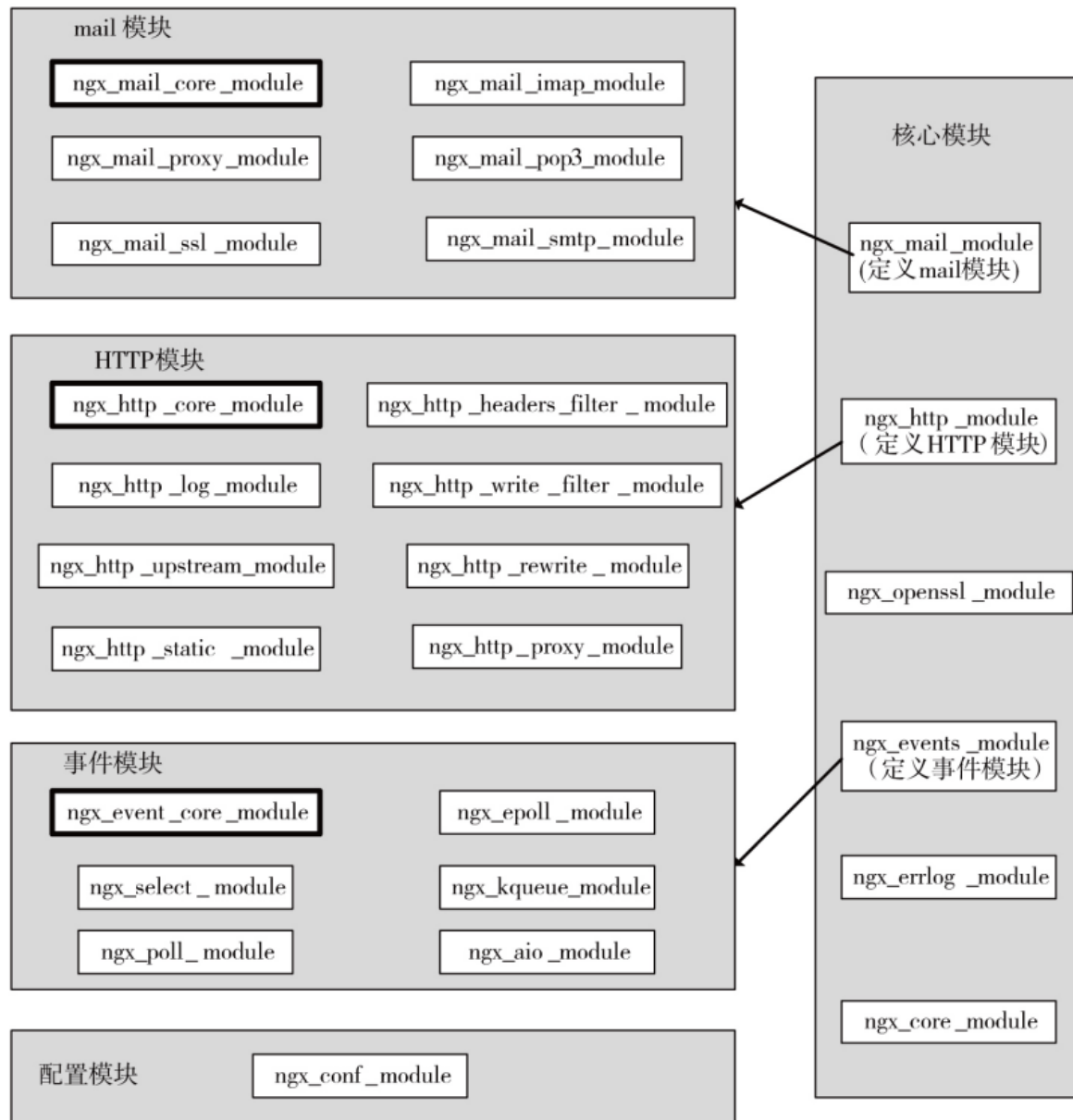
Nginx 架构图:



架构说明:

- 1) nginx启动时, 会生成两种类型的进程, 一个是主进程 (Master), 一个 (windows版本的目前只有一个) 和多个工作进程 (Worker)。主进程并不处理网络请求, 主要负责调度工作进程, 也就是图示的三项: 加载配置、启动工作进程及非停升级。所以, nginx启动以后, 查看操作系统的进程列表, 我们就能看到至少有两个nginx进程。
- 2) 服务器实际处理网络请求及响应的是工作进程 (worker), 在类unix系统上, nginx可以配置多个worker, 而每个worker进程都可以同时处理数以千计的网络请求。一般来说几个cpu就几个worker, 但是不绝对。worker进程专门用于建立连接。
- 3) 模块化设计。nginx的worker, 包括核心和功能性模块, 核心模块负责维持一个运行循环 (run-loop), 执行网络请求处理的不同阶段的模块功能, 如网络读写、存储读写、内容传输、外出过滤, 以及将请求发往上游服务器等。而其代码的模块化设计, 也使得我们可以根据需要对功能模块进行适当的选择和修改, 编译成具有特定功能的服务器。
- 4) 事件驱动、异步及非阻塞, 可以说是nginx得以获得高并发、高性能的关键因素, 同时也得益于对Linux、Solaris及类BSD等操作系统内核中事件通知及I/O性能增强功能的采用, 如kqueue、epoll及event ports。
- 5) 反向代理, 基于http和fastcgi实现反向代理, sendfile实现零拷贝。基于异步非阻塞的模型来实现。

Nginx 核心模块：



三、Nginx 配置与使用

知识点

1. 配置文件语法格式
2. 配置第一个静态WEB服务
3. 配置案例
4. 动静分离实现
5. 防盗链
6. 多域名站点
7. 下载限速
8. IP 黑名单
9. 基于user-agent分流
10. 日志配置

1、配置文件的语法格式：

先来看一个简单的nginx 配置

```

worker_processes 1; //几个工作进程
events {
    worker_connections 1024; //一个工作进程处理多少链接
}
http {
    include mime.types; # 引入这个文件
    default_type application/octet-stream;
    sendfile on; # 启用sendfile实现零拷贝，直接从内核拷贝到网卡，跳过拷贝到应用
    keepalive_timeout 65; # 长连接的保活时间
    server { # 代表虚拟机
        listen 80; //监听端口
        server_name localhost; //配置域名
        location / { # 请求的地址
            root html;
            index index.html index.htm;
        }
        location /nginx_status {
            stub_status on;
            access_log off;
        }
    }
}

```

上述配置中的events、http、server、location、upstream等属于**配置项块**。而worker_processes、worker_connections、include、listen 属于配置项块中的属性。 /nginx_status 属于配置块的特定参数参数。其中server块嵌套于http块，其可以直接继承访问Http块当中的参数。

配置块	名称开头用大口号包裹其对应属性
属性	基于空格切分属性名与属性值，属性值可能有多个项 都以空格进行切分 如： access_log logs/host.access.log main
参数	其配置在 块名称与大括号间，其值如果有多个也是通过空格进行拆

注意 如果配置项值中包括语法符号，比如空格符，那么需要使用单引号或双引号括住配置项值，否则Nginx会报语法错误。例如：

```

log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                '$status $body_bytes_sent "$http_referer" '
                '"$http_user_agent" "$http_x_forwarded_for"';

```

2、配置第一个静态WEB服务

基础站点演示：

- ☐ 创建站点目录 `mkdir -p /usr/www/luban`

```

[root@localhost usr]# mkdir -p www/luban
[root@localhost usr]# echo 'luban is good man' > www/luban/luban.html

```

- ☐ 编写静态文件，新建一个server块，注意监听的端口号可以不一样，只要server_name唯一就好了。

```

#log_format main '$remote_addr - $remote_user [$time_local] "$request" '
#
# '$status $body_bytes_sent "$http_referer" '
# '$http_user_agent' "$http_x_forwarded_for"';

#access_log logs/access.log main;

sendfile on;
#tcp_nopush on;

#keepalive_timeout 0;
keepalive_timeout 65;

#gzip on;
server {
    listen 80;
    server_name www.luban.com *.luban.com luban.*;
    location / {
        root /usr/www/luban/;
    }
}

server {
    listen 80;
    server_name localhost;

    #charset koi8-r;

    #access_log logs/host.access.log main;
}

```

也可以把root拿出去，然后配置index界面（就是默认页面，没有任何路径）。

```

#gzip on;
server {
    listen 80;
    server_name www.luban.com *.luban.com luban.*;
    root /usr/www/luban/;
    location / {
        index luban.html;
    }
}

```

- ☐ 配置 nginx.conf
 - ☐ 配置server
 - ☐ 配置location
 - ☐ 配置user 是 root

```

user root;
worker_processes 1;

```

- ☐ 检测语法是不是正确的。

```

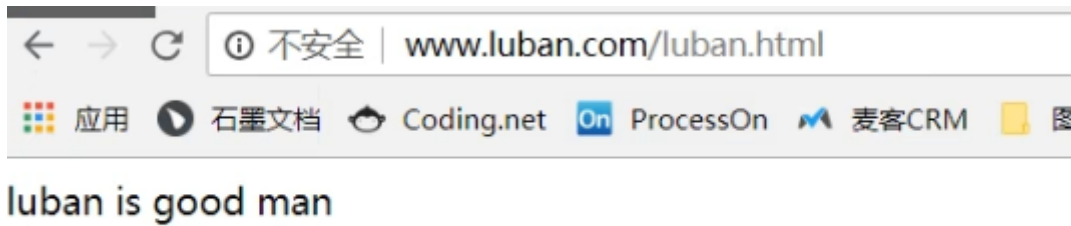
[root@localhost nginx]# ./sbin/nginx -t
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful

```

- ☐ 配置hosts，如果是开发的时候

192.168.0.147 www.luban.com
192.168.0.147 good.man.luban.com

测试:



基本配置介绍说明:

(1) 监听端口

语法: listen address:

默认: listen 80;

配置块: server

(2) 主机名称

语法: server_name name[.....];

默认: server_name "";

配置块: server

server_name后可以跟多个主机名称, 如server_name www.testweb.com、download.testweb.com;。支持通配符与正则

(3) location

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

配置块: server

1. / 基于uri目录匹配
2. =表示把URI作为字符串, 以便与参数中的uri做完全匹配。
3. ~表示正则匹配URI时是字母大小写敏感的。
4. ~*表示正则匹配URI时忽略字母大小写问题。
5. ^~表示正则匹配URI时只需要其前半部分与uri参数匹配即可。

动静分离演示: 静态文件与动态请求分开

- ☐ 创建静态站点
- ☐ 配置 location /static
- ☐ 配置 ~*.(gif|png|css|js)\$

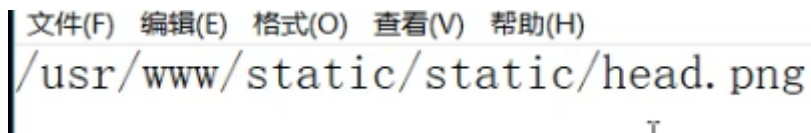
基于目录动静分离


```

server {
    listen 80;
    server_name *.luban.com;
    root /usr/www/luban;
    location / {
        index luban.html;
    }
    location /static {
        alias /usr/www/static; # 这里面不能写root，因为如果写root的话，就是root目录/static
    }
}

```

如果写成root会变成：



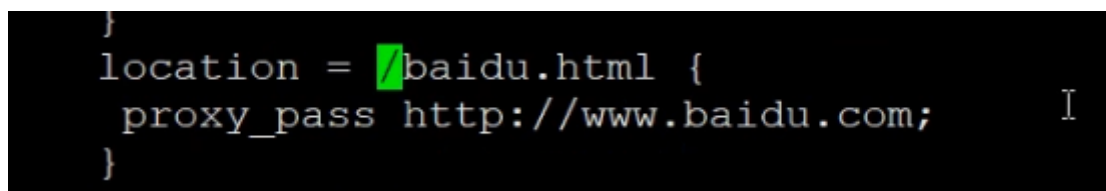
基于正则动静分离，这里配置root就是ok的。也针对一些不以static开头的路径。

```

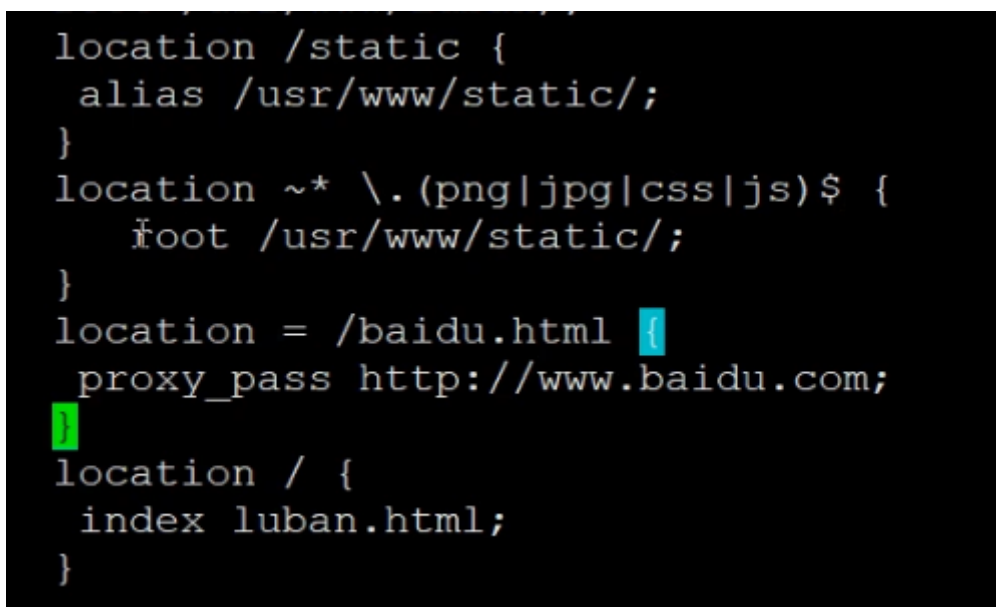
location ~* \.(gif|jpg|png|css|js)$ {
    root /usr/www/static;
}

```

测试定向至百度：



注意：location匹配顺序



```

location /static {
    alias /usr/www/static/;
}
location ~* \.(png|jpg|css|js)$ {
    root /usr/www/static/;
}
location = /baidu.html {
    proxy_pass http://www.baidu.com;
}
location / {
    index luban.html;
}

```

1. 正则表达式
2. /static
3. /

几个server并排写，谁在前面，先匹配谁。可以通过配置默认来说明谁在前面：

```
server {  
    listen      80 default;  
    server_name localhost;  
}
```

防盗链配置演示：

```
# 加入至指定location 即可实现  
valid_referers none blocked *.luban.com;  
if ($invalid_referer) {  
    return 403;  
}
```

下载限速：只会限制这个location中的下载速度

```
location /download {  
    limit_rate 1m; # 下载速度不超过1024k/s  
    limit_rate_after 30m; # 30m以后限速，也就是小文件不限速  
}
```

```
location /download {  
    alias /usr/www/download/;  
}
```

创建IP黑名单

```
# 创建黑名单文件  
echo 'deny 192.168.0.132;' >> black.ip  
#http 配置块中引入 黑名单文件  
include black.ip;
```

```
[root@localhost conf]# echo 'deny 192.168.0.132;' >> ip.black  
[root@localhost conf]# echo 'deny 192.168.0.0/11^C >>ip.black  
[root@localhost conf]#
```

屏蔽一个网段。如上。

```
http {  
    include mime.types;  
    include ip.black;
```

3、日志配置：

日志格式：

```
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;
#基于域名打印日志
access_log logs/$host.access.log main;
```

error日志的设置

语法: error_log /path/file level;

默认: error_log logs/error.log error;

level是日志的输出级别, 取值范围是debug、info、notice、warn、error、crit、alert、emerg,

针对指定的客户端输出debug级别的日志

语法: debug_connection[IP|CIDR]

events {

debug_connection 192.168.0.147;

debug_connection 10.224.57.0/200;

}

nginx.conf