# 从零开始学nginx

mp.weixin.qq.com/s/6KLS6u1iaLaqUhWHZNXuRw

击上方“**一口Linux**”，选择“**星标公众号**”

## 干货福利，第一时间送达！

## 1. nginx简介

nginx（发音同engine x）是一款轻量级的Web服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器，并在一个BSD-like协议下发行。

nginx由俄罗斯的程序设计师Igor Sysoev所开发，最初供俄国大型的入口网站及搜寻引擎Rambler使用。

第一个公开版本0.1.0发布于2004年10月4日。其将源代码以类BSD许可证的形式发布，因它的稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名。2011年6月1日，nginx 1.0.4发布。

nginx的特点是占有内存少，并发能力强，事实上nginx的并发能力确实在同类型的网页服务器中表现较好，中国大陆使用nginx网站用户有：百度、京东、新浪、网易、腾讯、淘宝等。

## 2. nginx的特性与优点

## 2.1 nginx的特性

nginx是一个很牛的高性能Web和反向代理服务器，它具有很多非常优越的特性：

在高连接并发的情况下，nginx是Apache服务器不错的替代品，能够支持高达50000个并发连接数的响应

使用epoll and kqueue作为开发模型

nginx作为负载均衡服务器：nginx既可在内部直接支持和PHP程序对外进行服务，也可支持作为HTTP代理服务器对外进行服务

nginx采用C进行编写，不论系统资源开销还是CPU使用效率都比Perlbal要好很多

## 2.2 nginx的优点

- 高并发连接：官方测试能够支撑5万并发连接，在实际生产环境中跑到2-3万并发连接数
- 内存消耗少：在3万并发连接下，开启的10个nginx进程才消耗150M内存（15M*10=150M）
- 配置文件非常简单：风格跟程序一样通俗易懂
- 成本低廉：nginx为开源软件，可以免费使用。而购买F5 BIG-IP、NetScaler等硬件负载均衡交换机则需要十多万至几十万人民币
- 支持Rewrite重写规则：能够根据域名、URL的不同，将HTTP请求分到不同的后端服务器群组
- 内置的健康检查功能：如果Nginx Proxy后端的某台Web服务器宕机了，不会影响前端访问
- 节省带宽：支持GZIP压缩，可以添加浏览器本地缓存的Header头
- 稳定性高：用于反向代理，宕机的概率微乎其微
- 模块化设计：模块可以动态编译
- 外围支持好：文档全，二次开发和模块较多
- 支持热部署：可以不停机重载配置文件
- 支持事件驱动、AIO（AsyncIO，异步IO）、mmap（Memory Map，内存映射）等性能优化

## 3. nginx的功能及应用类别

## 3.1 nginx的基本功能

1. 静态资源的web服务器，能缓存打开的文件描述符
2. http、smtp、pop3协议的反向代理服务器
3. 缓存加速、负载均衡
4. 支持FastCGI（fpm，LNMP），uWSGI（Python）等
5. 模块化（非DSO机制），过滤器zip、SSI及图像的大小调整
6. 支持SSL

## 3.2 nginx的扩展功能

- 基于名称和IP的虚拟主机
- 支持keepalive
- 支持平滑升级
- 定制访问日志、支持使用日志缓冲区提高日志存储性能
- 支持URL重写
- 支持路径别名
- 支持基于IP及用户的访问控制
- 支持速率限制，支持并发数限制

## 3.3 nginx的应用类别

- 使用nginx结合FastCGI运行PHP、JSP、Perl等程序
- 使用nginx作反向代理、负载均衡、规则过滤
- 使用nginx运行静态HTML网页、图片
- nginx与其他新技术的结合应用

# 4. nginx的模块与工作原理

nginx由内核和模块组成。其中，内核的设计非常微小和简洁，完成的工作也非常简单，仅仅通过查找配置文件将客户端请求映射到一个location block（location是nginx配置中的一个指令，用于URL匹配），而在这个location中所配置的每个指令将会启动不同的模块去完成相应的工作。

## 4.1 nginx的模块分类

nginx的模块从结构上分为核心模块、基础模块和第三方模块

1. HTTP模块、EVENT模块和MAIL模块等属于核心模块
2. HTTP Access模块、HTTP FastCGI模块、HTTP Proxy模块和HTTP Rewrite模块属于基本模块
3. HTTP Upstream模块、Request Hash模块、Notice模块和HTTP Access Key模块属于第三方模块

用户根据自己的需要开发的模块都属于第三方模块。正是有了如此多模块的支撑，nginx的功能才会如此强大

nginx模块从功能上分为三类，分别是：

1. Handlers（处理器模块）。此类模块直接处理请求，并进行输出内容和修改headers信息等操作。handlers处理器模块一般只能有一个
2. Filters（过滤器模块）。此类模块主要对其他处理器模块输出的内容进行修改操作，最后由nginx输出
3. Proxies（代理器模块）。就是nginx的HTTP Upstream之类的模块，这些模块主要与后端一些服务比如fastcgi等操作交互，实现服务代理和负载均衡等功能

nginx模块分为：核心模块、事件模块、标准Http模块、可选Http模块、邮件模块、第三方模块和补丁等

nginx基本模块：所谓基本模块，指的是nginx默认的功能模块，它们提供的指令，允许你使用定义nginx基本功能的变量，在编译时不能被禁用，包括：核心模块：基本功能和指令，如进程管理和安全。常见的核心模块指令，大部分是放置在配置文件的顶部事件模块：在Nginx内配置网络使用的能力。常见的events（事件）模块指令，大部分是放置在配置文件的顶部配置模块：提供包含机制

具体的指令，请参考nginx的官方文档

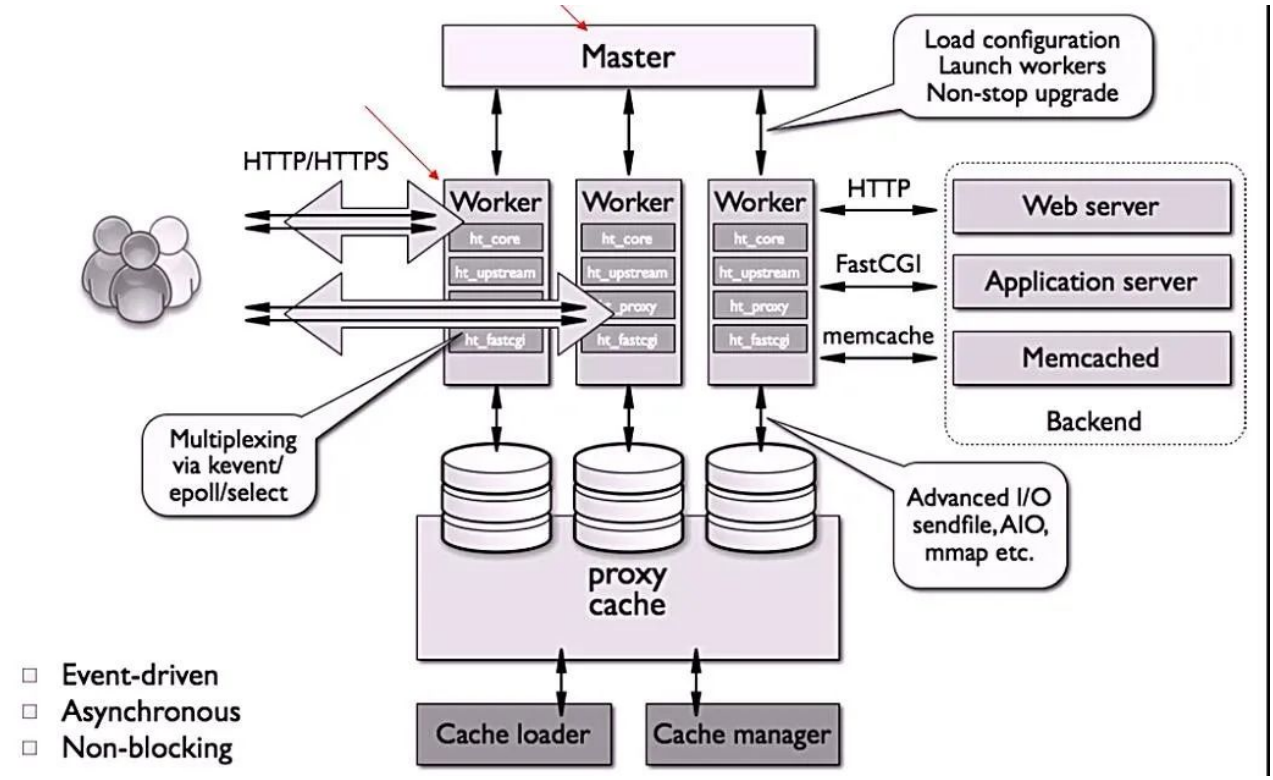
## 4.2 nginx的工作原理

nginx的模块直接被编译进nginx，因此属于静态编译方式。

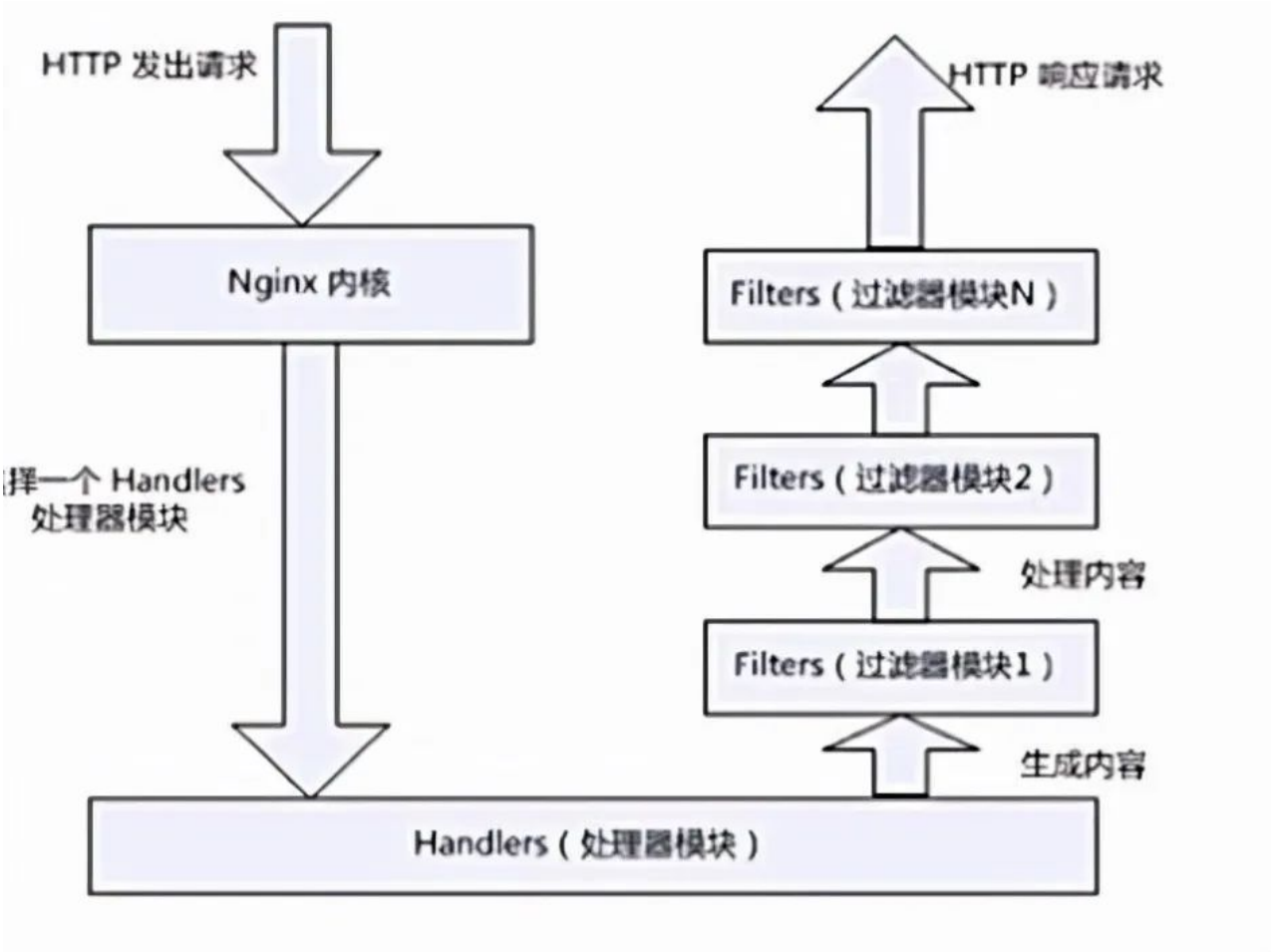
启动nginx后，nginx的模块被自动加载，与Apache不一样，首先将模块编译为一个so文件，然后在配置文件中指定是否进行加载。

在解析配置文件时，nginx的每个模块都有可能去处理某个请求，但是同一个处理请求只能由一个模块来完成。

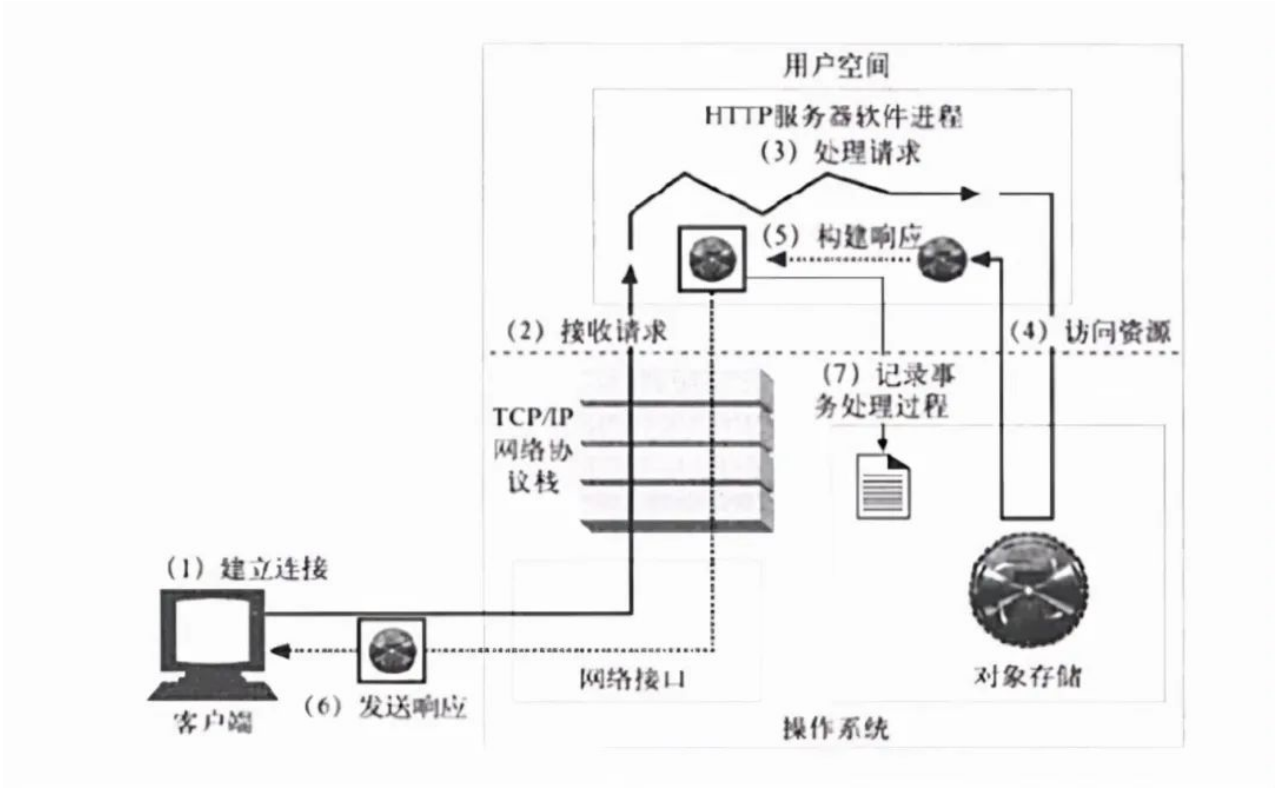
nginx的进程架构：
启动nginx时，会启动一个Master进程，这个进程不处理任何客户端的请求，主要用来产生worker线程，一个worker线程用来处理n个request。




下图展示了nginx模块一次常规的HTTP请求和响应的过程



下图展示了基本的WEB服务请求步骤




## 5. nginx的安装与配置

## 5.1 nginx的安装

```
//创建系统用户nginx[root@localhost ~]# useradd -r -M -s /sbin/nologin nginx//安装依赖环境[root@localhost ~]# yum -y install pcre-devel openssl openssl-devel gd-devel gcc gcc-c++安装过程略....[root@localhost ~]# yum -y groups mark install 'Development Tools'Loaded plugins: product-id, search-disabled-repos, subscription-managerThis system is not registered with an entitlement server. You can use subscription-manager to register.There is no installed groups file.Maybe run: yum groups mark convert (see man yum)Marked install: Development Tools//创建日志存放目录[root@localhost ~]# mkdir -p /var/log/nginx[root@localhost ~]# chown -R nginx.nginx /var/log/nginx//下载nginx[root@localhost ~]# cd /usr/src/[root@localhost src]# wget http://nginx.org/download/nginx-1.12.0.tar.gz--2018-08-20 11:19:09-- http://nginx.org/download/nginx-1.12.0.tar.gzResolving nginx.org (nginx.org)... 95.211.80.227, 206.251.255.63, 2606:7100:1:69::3f, ...Connecting to nginx.org (nginx.org)|95.211.80.227|:80... connected.HTTP request sent, awaiting response... 200 OKLength: 980831 (958K) [application/octet-stream]Saving to: ‘nginx-1.12.0.tar.gz’100% [======================================================>] 980,831 15.9KB/s in 43s2018-08-20 11:19:52 (22.3 KB/s) - ‘nginx-1.12.0.tar.gz’ saved [980831/980831]//编译安装[root@localhost src]# lsdebug kernels nginx-1.12.0.tar.gz[root@localhost src]# tar xf nginx-1.12.0.tar.gz[root@localhost src]# cd nginx-1.12.0[root@localhost nginx-1.12.0]# ./configure \--prefix=/usr/local/nginx \--user=nginx \--group=nginx \--with-debug \--with-http_ssl_module \--with-http_realip_module \--with-http_image_filter_module \--with-http_gunzip_module \--with-http_gzip_static_module \--with-http_stub_status_module \--http-log-path=/var/log/nginx/access.log \--error-log-path=/var/log/nginx/error.log[root@localhost nginx-1.12.0]# make -j $(grep 'processor' /proc/cpuinfo | wc -l) && make install安装过程略....
```

## 5.2 nginx安装后配置

```
//配置环境变量[root@localhost ~]# echo 'export PATH=/usr/local/nginx/sbin:$PATH' > /etc/profile.d/nginx.sh[root@localhost ~]# . /etc/profile.d/nginx.sh//服务控制方式，使用nginx命令-t //检查配置文件语法-v //输出nginx的版本-c //指定配置文件的路径-s //发送服务控制信号，可选值有{stop|quit|reopen|reload}//启动nginx[root@localhost ~]# nginx[root@localhost ~]# ss -antlState Recv-Q Send-Q Local Address:Port Peer Address:PortLISTEN 0 128 *:80 *:*LISTEN 0 128 *:22 *:*LISTEN 0 100 127.0.0.1:25 *:*LISTEN 0 128 :::22 :::*LISTEN 0 100 ::1:25 :::*
```

## 6 nginx的配置文件详解

主配置文件：

```
/usr/local/nginx/conf/nginx.conf
```

默认启动nginx时，使用的配置文件是：安装路径/conf/nginx.conf文件

可以在启动nginx时通过-c选项来指定要读取的配置文件

nginx常见的配置文件及其作用

| 配置文件 | 作用 |
| --- | --- |
| nginx.conf | nginx的基本配置文件 |
| mime.types | MIME类型关联的扩展文件 |
| fastcgi.conf | 与fastcgi相关的配置 |
| proxy.conf | 与proxy相关的配置 |
| sites.conf | 配置nginx提供的网站，包括虚拟主机 |

## 6.1 nginx.conf配置详解

nginx.conf的内容分为以下几段：

- main配置段：全局配置段。其中main配置段中可能包含event配置段
- event {}：定义event模型工作特性
- http {}：定义http协议相关的配置

配置指令：要以分号结尾，语法格式如下：

```
derective value1 [value2 ...];
```

支持使用变量：

内置变量：模块会提供内建变量定义

自定义变量：set var_name value

## 6.2 用于调试、定位问题的配置参数

daemon {on|off}; //是否以守护进程方式运行nginx，调试时应设置为offmaster_process {on|off}; //是否以master/worker模型来运行nginx，调试时可以设置为offerror_log 位置 级别; //配置错误日志

error_log里的位置和级别能有以下可选项：

| 位置 | 级别 |
| --- | --- |
| | |

| | |
|---|---|
| file<br>stderr<br>syslog:server=address[,parameter=value]<br>memory:size | debug：若要使用debug级别，需要在编译nginx时使用--with-debug选项<br>info<br>notice<br>warn<br>error<br>crit<br>alert<br>emerg |

## 6.3 正常运行必备的配置参数

```
user USERNAME [GROUPNAME]; //指定运行worker进程的用户和组pid /path/to/pid_file; //指定nginx守护进程的pid文件worker_rlimit_nofile number; //设置所有worker进程最大可以打开的文件数，默认为1024worker_rlimit_core size; //指明所有worker进程所能够使用的总体的最大核心文件大小，保持默认即可
```

## 6.4 优化性能的配置参数

worker_processes n; //启动n个worker进程，这里的n为了避免上下文切换，通常设置为cpu总核心数-1或等于总核心数

worker_cpu_affinity cpumask ...; //将进程绑定到某cpu中，避免频繁刷新缓存//cpumask：使用8位二进制表示cpu核心，如：

0000 0001 //第一颗cpu核心

0000 0010 //第二颗cpu核心

0000 0100 //第三颗cpu核心

0000 1000 //第四颗cpu核心

0001 0000 //第五颗cpu核心

0010 0000 //第六颗cpu核心

0100 0000 //第七颗cpu核心

1000 0000 //第八颗cpu核心

timer_resolution interval; //计时器解析度。降低此值，可减少gettimeofday()系统调用的次数

worker_priority number; //指明worker进程的nice值

## 6.5 事件相关的配置：event{}段中的配置参数

accept_mutex {off|on}; //master调度用户请求至各worker进程时使用的负载均衡锁；on表示能让多个worker轮流地、序列化地去响应新请求

lock_file file; //accept_mutex用到的互斥锁锁文件路径use [epoll | rtsig | select | poll]; //指明使用的事件模型，建议让nginx自行选择

worker_connections #; //每个进程能够接受的最大连接数

## 6.6 网络连接相关的配置参数

keepalive_timeout number; //长连接的超时时长，默认为65s

keepalive_requests number; //在一个长连接上所能够允许请求的最大资源数

keepalive_disable [msie6|safari|none]; //为指定类型的UserAgent禁用长连接

tcp_nodelay on|off; //是否对长连接使用TCP_NODELAY选项，为了提升用户体验，通常设为on

client_header_timeout number; //读取http请求报文首部的超时时长

client_body_timeout number; //读取http请求报文body部分的超时时长

send_timeout number; //发送响应报文的超时时长

## 6.7 fastcgi的相关配置参数

LNMP：php要启用fpm模型
配置示例如下：

```
location ~ \.php$ {root html;fastcgi_pass 127.0.0.1:9000; //定义反向代理
fastcgi_index index.php;fastcgi_param SCRIPT_FILENAME
/scripts$fastcgi_script_name;include fastcgi_params;}
```

## 6.8 常需要进行调整的参数

```
worker_processesworker_connectionsworker_cpu_affinityworker_priority
```

## 6.9 nginx作为web服务器时使用的配置：http{}段的配置参数

http{...}：配置http相关，由ngx_http_core_module模块引入。nginx的HTTP配置主要包括四个区块，结构如下：

```
http {//协议级别include mime.types;default_type application/octet-stream;keepalive_timeout 65;gzip on;upstream {//负载均衡配置...}server {//服务器级别，每个server类似于httpd中的一个<VirtualHost>listen 80;server_name localhost;location / {//请求级别，类似于httpd中的<Location>，用于定义URL与本地文件系统的映射关系root html;index index.html index.htm;}}}
```

http{}段配置指令：
server {}：定义一个虚拟主机，示例如下：

```
server {listen 80;server_name www.idfsoft.com;root "/vhosts/web";}
```

listen：指定监听的地址和端口listen address[:port];listen port;server_name NAME [...];后面可跟多个主机，名称可使用正则表达式或通配符当有多个server时，匹配顺序如下：先做精确匹配检查左侧通配符匹配检查，如*.idfsoft.com右侧通配符匹配检查，如mail.*正则表达式匹配检查，如~^.*\.idfsoft\.com$default_serverroot path; 设置资源路径映射，用于指明请求的URL所对应的资源所在的文件系统上的起始路径alias path; 用于location配置段，定义路径别名index file; 默认主页面index index.php index.html;error_page code [...] [=code] URI | @name 根据http响应状态码来指明特用的错误页面，例如 error_page 404 /404_customed.html[=code]：以指定的响应码进行响应，而不是默认的原来的响应，默认表示以新资源的响应码为其响应码，例如 error_page 404 =200 /404_customed.htmllog_format 定义日志格式log_format main '$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;//注意：此处可用变量为nginx各模块内建变量location区段，通过指定模式来与客户端请求的URI相匹配//功能：允许根据用户请求的URI来匹配定义的各location，匹配到时，此请求将被相应的location配置块中的配置所处理，例如做访问控制等功能//语法：location [ 修饰符 ] pattern {......}

常用修饰符说明：

| 修饰符 | 功能 |
| --- | --- |
| = | 精确匹配 |
| ~ | 正则表达式模式匹配，区分大小写 |
| ~* | 正则表达式模式匹配，不区分大小写 |
| ^~ | 前缀匹配，类似于无修饰符的行为，也是以指定模块开始，不同的是，如果模式匹配，那么就停止搜索其他模式了，不支持正则表达式 |

@ 定义命名location区段，这些区段客户端不能访问，只可以由内部产生的请求来访问，如try_files或error_page等

没有修饰符表示必须以指定模式开始，如：

```
server {server_name www.idfsoft.com;location /abc {......}}
```

那么如下内容就可正确匹配：

```
http://www.idfsoft.com/abchttp://www.idfsoft.com/abc?
p1=11&p2=22http://www.idfsoft.com/abc/
```

=：表示必须与指定的模式精确匹配，如：

```
server {server_name www.idfsoft.com;location = /abc {......}}
```

那么如下内容就可正确匹配：

```
http://www.idfsoft.com/abchttp://www.idfsoft.com/abc?p1=11&p2=22
```

如下内容则无法匹配：

```
http://www.idfsoft.com/abc/http://www.idfsoft.com/abc/abcde
```

~：表示指定的正则表达式要区分大小写，如：

```
server {server_name www.idfsoft.com;location ~ ^/abc$ {......}}
```

那么如下内容就可正确匹配：

```
http://www.idfsoft.com/abchttp://www.idfsoft.com/abc?p1=11&p2=22
```

如下内容则无法匹配：

```
http://www.idfsoft.com/abc/http://www.idfsoft.com/ABChttp://www.idfsoft.com/abcde
```

~*：表示指定的正则表达式不区分大小写，如：

```
server {server_name www.idfsoft.com;location ~* ^/abc$ {......}}
```

那么如下内容就可正确匹配：

```
http://www.idfsoft.com/abchttp://www.idfsoft.com/abc?
p1=11&p2=22http://www.idfsoft.com/ABC
```

如下内容则无法匹配：

```
http://www.idfsoft.com/abc/http://www.idfsoft.com/abcde
```

~：类似于无修饰符的行为，也是以指定模式开始，不同的是，如果模式匹配，则停止搜索其他模式

查找顺序和优先级：由高到底依次为

- 带有=的精确匹配优先
- 正则表达式按照他们在配置文件中定义的顺序
- 带有^~修饰符的，开头匹配
- 带有~或~*修饰符的，如果正则表达式与URI匹配
- 没有修饰符的精确匹配

优先级次序如下：

( location = 路径 ) --> ( location ^~ 路径 ) --> ( location ~ 正则 ) --> ( location ~* 正则 ) --> ( location 路径 )

## 6.10 访问控制

用于location段
allow：设定允许哪台或哪些主机访问，多个参数间用空格隔开
deny：设定禁止哪台或哪些主机访问，多个参数间用空格隔开
示例：

```
allow 192.168.1.1/32 172.16.0.0/16;deny all;
```

## 6.11 基于用户认证

```
auth_basic "欢迎信息";auth_basic_user_file "/path/to/user_auth_file"
```

user_auth_file内容格式为：

username:password

这里的密码为加密后的密码串，建议用htpasswd来创建此文件：

htpasswd -c -m /path/to/.user_auth_file USERNAME

## 6.12 https配置

生成私钥，生成证书签署请求并获得证书，然后在nginx.conf中配置如下内容：

```
server {listen 443 ssl;server_name www.idfsoft.com;ssl_certificate
/etc/nginx/ssl/nginx.crt;ssl_certificate_key
/etc/nginx/ssl/nginx.key;ssl_session_cache shared:SSL:1m;ssl_session_timeout
5m;ssl_ciphers HIGH:!aNULL:!MD5;ssl_prefer_server_ciphers on;location / {root
html;index index.html index.htm;}}
```

## 6.13 开启状态界面

开启status：

```
location /status {stub_status {on | off};allow 172.16.0.0/16;deny all;}
```

访问状态页面的方式：http://server_ip/status

状态页面信息详解：

| 状态码 | 表示的意义 |
| --- | --- |
| Active connections 2 | 当前所有处于打开状态的连接数 |

| | |
|---|---|
| accepts | 总共处理了多少个连接 |
| handled | 成功创建多少握手 |
| requests | 总共处理了多少个请求 |
| Reading | nginx读取到客户端的Header信息数，表示正处于接收请求状态的连接数 |
| Writing | nginx返回给客户端的Header信息数，表示请求已经接收完成，且正处于处理请求或发送响应的过程中的连接数 |
| Waiting | 开启keep-alive的情况下，这个值等于active - (reading + writing)，意思就是Nginx已处理完正在等候下一次请求指令的驻留连接 |

## 6.14 rewrite

语法：`rewrite regex replacement flag;`，如：`rewrite ^/images/(.*\.jpg)$ /imgs/$1 break;`此处的`$1`用于引用`(.*.jpg)`匹配到的内容，又如：`rewrite ^/bbs/(.*)$ http://www.idfsoft.com/index.html redirect;`

如上例所示，replacement可以是某个路径，也可以是某个URL

常见的flag

| flag | 作用 |
|---|---|
| last | 基本上都用这个flag，表示当前的匹配结束，继续下一个匹配，最多匹配10个到20个<br>一旦此rewrite规则重写完成后，就不再被后面其它的rewrite规则进行处理<br>而是由UserAgent重新对重写后的URL再一次发起请求，并从头开始执行类似的过程 |
| break | 中止Rewrite，不再继续匹配<br>一旦此rewrite规则重写完成后，由UserAgent对新的URL重新发起请求，且不再会被当前location内的任何rewrite规则所检查 |
| redirect | 以临时重定向的HTTP状态302返回新的URL |

| | |
|---|---|
| permanent | 以永久重定向的HTTP状态301返回新的URL |

rewrite模块的作用是用来执行URL重定向。这个机制有利于去掉恶意访问的url，也有利于搜索引擎优化（SEO）

nginx使用的语法源于Perl兼容正则表达式（PCRE）库，基本语法如下：

| 标识符 | 意义 |
|---|---|
| ^ | 必须以^后的实体开头 |
| $ | 必须以$前的实体结尾 |
| . | 匹配任意字符 |
| [] | 匹配指定字符集内的任意字符 |
| [^] | 匹配任何不包括在指定字符集内的任意字符串 |
| \| | 匹配 \| 之前或之后的实体 |
| () | 分组，组成一组用于匹配的实体，通常会有 \| 来协助 |

捕获子表达式，可以捕获放在()之间的任何文本，比如：

^(hello|sir)$ //字符串为“hi sir”捕获的结果：$1=hi$2=sir

//这些被捕获的数据，在后面就可以当变量一样使用了

## 6.15 if

语法：if (condition) {...}

应用场景：

server段location段

常见的condition

变量名（变量值为空串，或者以“0”开始，则为false，其它的均为true）以变量为操作数构成的比较表达式（可使用=，!=类似的比较操作符进行测试）正则表达式的模式匹配操作~：区分大小写的模式匹配检查~*：不区分大小写的模式匹配检查!~和!~*：对上面两种测试取反测试指定路径为文件的可能性（-f，!-f)测试指定路径为目录的可能性（-d，!-d）测试文件的存在性（-e，!-e）检查文件是否有执行权限（-x，!-x）

## 6.15.1 基于浏览器实现分离案例

```
if ($http_user_agent ~ Firefox) {rewrite ^(.*)$ /firefox/$1 break;}if
($http_user_agent ~ MSIE) {rewrite ^(.*)$ /msie/$1 break;}if ($http_user_agent ~
Chrome) {rewrite ^(.*)$ /chrome/$1 break;}
```

## 6.15.2 防盗链案例

```
location ~* \.(jpg|gif|jpeg|png)$ {valid_referers none blocked www.idfsoft.com;if
($invalid_referer) {rewrite ^/ http://www.idfsoft.com/403.html;}}
```

## 6.16 反向代理与负载均衡

nginx通常被用作后端服务器的反向代理，这样就可以很方便的实现动静分离以及负载均衡，从而大大提高服务器的处理能力。

nginx实现动静分离，其实就是在反向代理的时候，如果是静态资源，就直接从nginx发布的路径去读取，而不需要从后台服务器获取了。

但是要注意，这种情况下需要保证后端跟前端的程序保持一致，可以使用Rsync做服务端自动同步或者使用NFS、MFS分布式共享存储。

Http Proxy模块，功能很多，最常用的是proxy_pass和proxy_cache

如果要使用proxy_cache，需要集成第三方的ngx_cache_purge模块，用来清除指定的URL缓存。这个集成需要在安装nginx的时候去做，如：
./configure --add-module=../ngx_cache_purge-1.0 .....

nginx通过upstream模块来实现简单的负载均衡，upstream需要定义在http段内

在upstream段内，定义一个服务器列表，默认的方式是轮询，如果要确定同一个访问者发出的请求总是由同一个后端服务器来处理，可以设置ip_hash，如：

```
upstream idfsoft.com {ip_hash;server 127.0.0.1:9080 weight=5;server 127.0.0.1:8080
weight=5;server 127.0.0.1:1111;}
```

注意：这个方法本质还是轮询，而且由于客户端的ip可能是不断变化的，比如动态ip，代理，翻墙等，因此ip_hash并不能完全保证同一个客户端总是由同一个服务器来处理。

定义好upstream后，需要在server段内添加如下内容：

```
server {  location / {    proxy_pass http://idfsoft.com;  }}
```

end

**一口Linux**

**关注，回复【1024】海量Linux资料赠送**


**一口Linux**
写点代码，写点人生！

203篇原创内容

公众号

**精彩文章合集**

文章推荐

☞【专辑】ARM
☞【专辑】粉丝问答
☞【专辑】所有原创
☞【专辑】linux入门
☞【专辑】计算机网络
☞【专辑】Linux驱动
☞【干货】嵌入式驱动工程师学习路线

☞【干货】Linux嵌入式所有知识点-思维导图

点击“**阅读原文**”查看更多分享，欢迎**点分享、收藏、点赞、在看**