# 主模块的配置选项

user www-data;

#设置进程以什么用户运行，编译安装时没有指定则默认为nobody账户

pid /run/nginx.pid;

#设置pid文件路径，可以使用kill命令发送相关信号

worker\_processes auto;

#设置工作进程数，一个工作进程为一个单线程，在CPU密集型环境中，可以设置为CPU核数

worker\_rlimit\_nofile 100000;

#worker\_rlimit\_nofile 更改worker进程的最大打开文件数限制。如果没设置的话，这个值为操作系统的限制。设置后你的操作系统和Nginx可以处理比“ulimit -a”更多的文件，所以把这个值设高，这样nginx就不会有“too many open files”问题了。

# event模块配置选项

event模块主要控制nginx处理连接的方式

events {

worker\_connections 768; #一个worker进程同时打开的最大连接数

#worker\_connections和worker\_processes可以计算你的理论最大链接数：worker\_connections\*worker\_processes

use epoll;

#如果在configure时指定的不止一个事件模型，可以通过use告诉nginx要使用哪一个模型：seletc、poll、kqueue、epoll、rtsig、/dev/poll、eventport等，设置用于复用客户端线程的轮询方法。

Ps: 如果你使用Linux 2.6+，你应该使用epoll。如果你使用\*BSD，你应该使用kqueue。如果你不知道Nginx该使用哪种轮询方法的话，它会选择一个最适合你操作系统的

multi\_accept on;

#指示nginx收到一个新连接通知后接受尽可能多的连接。

}

# http模块里面主要是对http服务器相关属性进行设置

## nginx配置文件主要是Http字段，一个Http字段里面可以有多个字段，一个server字段代表一个虚拟主机。一个server里面可以有多个字段，每个location代表一个URI资源。

http {

# Basic Settings 基础设置

sendfile on;

#sendfile拷贝文件在内核态完成，更加高效

tcp\_nopush on;

#指示nginx在一个数据包里发送所有头文件，而不是一个接一个的发送

tcp\_nodelay on;

#nginx不要缓存数据，而是一段一段的发送——当需要及时发送数据时，就应该给应用设置这个属性，这样发送一小块数据信息时就不能立即得到返回值。

keepalive\_timeout 65;

#可以设置两个值，第一个表示客户端与服务器长连接的超时时间，超过这个时间，服务器将关闭连接。第二个值指定的应答头中keep-alive中timeout的值，让浏览器知道什么时候关闭连接。

types\_hash\_max\_size 2048;

server\_tokens off;

#不会让nginx执行的速度更快，但它可以关闭在错误页面中的nginx版本数字，不在错误页面以及响应头中显示Nginx版本号。这样对于安全性是有好处的。

# server\_names\_hash\_bucket\_size 64;

# server\_name\_in\_redirect off;

include /etc/nginx/mime.types;

#可以用include指令包含一些其他文件，支持通配符，可以使用绝对路径，也可以使用相对路径，相对路径以nginx.conf为根据

default\_type application/octet-stream; # 设置默认的MIME类型

# SSL Settings SSL设置

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2; # Dropping SSLv3, ref: POODLE

ssl\_prefer\_server\_ciphers on;

# Logging Settings 日志记录

access\_log /var/log/nginx/access.log;

#指定访问日志的路径和格式，缓冲区大小 默认combined格式。语法: access\_log path [format [buffer=size [flush=time]]]; #[]内为可选项

error\_log /var/log/nginx/error.log; # 指定nginx错误日志文件的位置，禁止错误日志使用error\_log /dev/null，可存在main，http，server字段

# Gzip Settings gzip设置

#开启gzip压缩

gzip on;

gzip\_disable "msie6";

#gzip\_disable为指定的客户端禁用gzip功能。我们设置成IE6或者更低版本以使我们的方案能够广泛兼容。

# gzip\_vary on;

# gzip\_proxied any;

#允许或者禁止压缩基于请求和响应的响应流。any将会压缩所有的请求。

# gzip\_comp\_level 6;

#设置数据的压缩等级。这个等级可以是1-9之间的任意数值，9是最慢但是压缩比最大的。

# gzip\_buffers 16 8k;

# gzip\_http\_version 1.1;

# gzip\_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/javascript;

#设置需要压缩的数据格式。

#在三次握手时，发送给客户端应答后的超时时间，目前还没进入连接状态，只完成了两次握手，如果在规定时间没收到应答包，nginx将关闭链接

send\_timeout 30

# Virtual Host Configs

include /etc/nginx/conf.d/\*.conf;

include /etc/nginx/sites-enabled/\*;

#包含sites-enabled下的所有文件

# File upload

client\_max\_body\_size 12M;

# server模块嵌在http模块中，主要用来配置虚拟主机

server {

#指定server字段中可以被访问到的ip地址及端口

listen 80;

#将http请求的主机头与server中的server\_name参数进行匹配，并找出第一个结果，如果没有server\_name参数匹配上，则第一个出现listen的server将被匹配，多域名用空格分割

server\_name [www.nginx.com](http://www.nginx.com);

#这个指令是应答头中的content-type字段使用指定的编码集，off表示不在应答头中添加content-type信息

charset off;

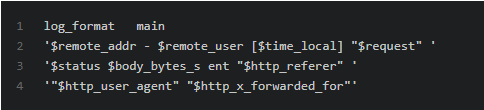
#PS：服务端根据请求头（headers）中的 Content-Type 字段来获知请求中的消息主体是用何种方式编码，再对请求主体进行解析。

#指定www.nginx.com域名的访问日志路径及格式

#nginx日志记录有两条指令：

* + 1. log\_format：用来设置日志格式；
    2. access\_log：用来指定日志文件的存放路径、格式（把定义的log\_format 跟在后面）和缓存大小；如果不想启用日志则access\_log off ;

示例：



具体可设置的参数有：



用了log\_format 指令设置了日志格式之后，需要用access\_log指令指定日志文件的存放路径；

语法：

access\_log path(存放路径) format (自定义日志名称)

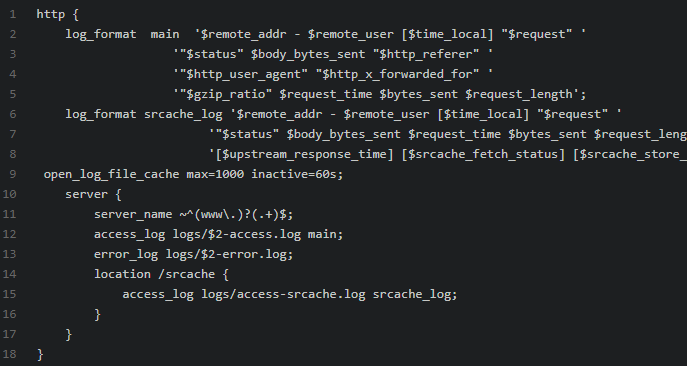
示例:

access\_log logs/host.access.log main;

#如果在url中没有指定文件，则设置一个默认主页，可以设置多个文件，空格分开，可以在http、server、location中设置

PS：一般log\_format在全局设置，可以设置多个。access\_log 可以在全局设置，但往往是定义在虚拟主机（server）中的location中。

如：



index index.php index.htm;

#根据URL的不同需求进行配置，可以使用字符串和正则匹配，最确切的匹配被使用，搜索到第一个后会停止

# ~\* 不区分大小写；~ 区分大小写；^~ 禁止在字符串匹配后检查正则；= 在URL和location之间精确匹配，匹配完成后不做额外搜索。

location /i/ {

#请求到达后的文件根目录，在请求中root会把location匹配的值加到root指定的值后面，请求/i/a.php，则会是/html/i/a.php响应

root html;

#在location中设置index

index index.html index.htm;

}

## root path在server中位置：

在server中：表示全局的，对所有的location都有意义。

在location：局部的资源定义

location /a/ {

root /var/www/;

}

location /就代表location里面的root，若是location里的root未定义，就从server里面继承！

如：/a/1.html = /var/www/a/1.html

#为错误代码指定相应的错误界面，可以用在http、server、location字段中。

error\_page 404 /404.html;

# 从定向服务器错误页面到静态页面/50x.html

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

#精确匹配50x.html，真实响应是/html/50x.html

location = /50x.html {

root html;

}

# proxy the PHP scripts to Apache listening on 127.0.0.1:80

location ~ \.php$ {

proxy\_pass http://127.0.0.1;

}

#配置php脚本传至fastcgi

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

#/scripts是php脚本所在的目录

fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name;

include fastcgi\_params;

}

#拒绝访问.htaccess文件

location ~ /\.ht {

deny all;

}

}

}

#mail {

# # See sample authentication script at:

# # http://wiki.nginx.org/ImapAuthenticateWithApachePhpScript

#

# # auth\_http localhost/auth.php;

# # pop3\_capabilities "TOP" "USER";

# # imap\_capabilities "IMAP4rev1" "UIDPLUS";

#

# server {

# listen localhost:110;

# protocol pop3;

# proxy on;

# }

#

# server {

# listen localhost:143;

# protocol imap;

# proxy on;

# }

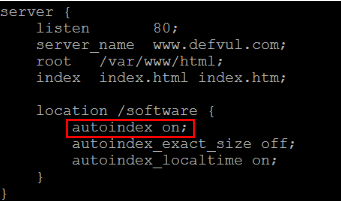
#}

## 安全加固

1）默认安装nginx是已经关闭目录浏览功能的（autoindex off;）

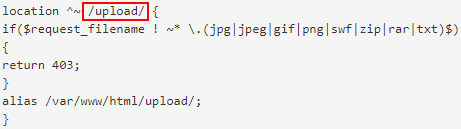
由于配置不当可能被打开（autoindex on;），如果业务需要目录浏览，可以针对特定目录进行设置。

仅对网站软件下载目录开放目录浏览功能。



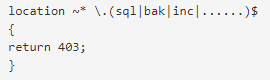
2）限制上传目录执行脚本

对upload目录进行限制，仅允许访问jpg等常见上传附件文件（根据业务需求自行修改允许的文件类型）。



3）备份文件处理

禁止对服务器下sql、bak等敏感文件进行访问（根据业务需求自行修改允许的文件类型）。



4）不在错误页面以及响应头中显示Nginx版本号

在server段中配置：

server\_tokens off;

5）重定向错误页面

将错误页面重定向到指定的错误页面（根据业务自行调整），避免显示nginx缺省的错误页面。

在server段中配置：

error\_page 400 403 404 500 501 502 503 504 505 <http://www.defvul.com/error.html>