目录

[Nginx配置文件说明 2](#_Toc99461965)

[一、 main模块 3](#_Toc99461966)

[二、 event模块 3](#_Toc99461967)

[三、 http模块 3](#_Toc99461968)

[1. 基础配置 3](#_Toc99461969)

[2. 日志配置 4](#_Toc99461970)

[3. SSL证书配置 5](#_Toc99461971)

[4. 压缩配置 5](#_Toc99461972)

[5. 文件缓存配置 5](#_Toc99461973)

[四、 sever模块 6](#_Toc99461974)

[五、 location模块 6](#_Toc99461975)

[1. 基本配置 7](#_Toc99461976)

[2. 反向代理配置 7](#_Toc99461977)

[3. uwsgi配置 7](#_Toc99461978)

[六、 负载均衡模块（upstream） 7](#_Toc99461979)

# Nginx配置文件说明

在项目使用中，使用最多的三个核心功能是静态服务器、反向代理和负载均衡。

这三个不同的功能的使用，都跟Nginx的配置密切相关，Nginx服务器的配置信息主要集中在"nginx.conf"这个配置文件中，并且所有的可配置选项大致分为以下几个部分.

main # 全局配置

events { # 工作模式配置

}

http { # http设置

....

server { # 服务器主机配置（虚拟主机、反向代理等）

....

location { # 路由配置（虚拟目录等）

....

}

location path {

....

}

location otherpath {

....

}

}

server {

....

location {

....

}

}

upstream name { # 负载均衡配置

....

}

}

## main模块

**user：** 用来指定nginx worker进程运行用户以及用户组，默认nobody账号运行

**worker\_processes：**  指定nginx要开启的子进程数量，运行过程中监控每个进程消耗内存(一般几M~几十M不等)根据实际情况进行调整，通常数量是CPU内核数量的整数倍

**error\_log：**  定义错误日志文件的位置及输出级别【debug / info / notice / warn / error / crit】

**pid：**  用来指定进程id的存储文件的位置

**worker\_rlimit\_nofile：**  用于指定一个进程可以打开最多文件数量的描述

...

## event模块

**worker\_connections：** 指定最大可以同时接收的连接数量，这里一定要注意，最大连接数量是和worker processes共同决定的。

**multi\_accept：**  配置指定nginx在收到一个新连接通知后尽可能多的接受更多的连接

**use epoll：**  配置指定了线程轮询的方法，如果是linux2.6+，使用epoll，如果是BSD如Mac请使用Kqueue

...

## http模块

作为web服务器，http模块是nginx最核心的一个模块，配置项也是比较多的，项目中会设置到很多的实际业务场景，需要根据硬件信息进行适当的配置。

### 基础配置

sendfile on：配置on让sendfile发挥作用，将文件的回写过程交给数据缓冲去去完成，而不是放在应用中完成，这样的话在性能提升有有好处

tcp\_nopush on：让nginx在一个数据包中发送所有的头文件，而不是一个一个单独发

**tcp\_nodelay on：**让nginx不要缓存数据，而是一段一段发送，如果数据的传输有实时性的要求的话可以配置它，发送完一小段数据就立刻能得到返回值，但是不要滥用哦

**keepalive\_timeout 10：**给客户端分配连接超时时间，服务器会在这个时间过后关闭连接。一般设置时间较短，可以让nginx工作持续性更好

**client\_header\_timeout 10：**设置请求头的超时时间

**client\_body\_timeout 10:**设置请求体的超时时间

**send\_timeout 10：**指定客户端响应超时时间，如果客户端两次操作间隔超过这个时间，服务器就会关闭这个链接

**limit\_conn\_zone $binary\_remote\_addr zone=addr:5m ：**设置用于保存各种key的共享内存的参数，

**limit\_conn addr 100:** 给定的key设置最大连接数

**server\_tokens：**虽然不会让nginx执行速度更快，但是可以在错误页面关闭nginx版本提示，对于网站安全性的提升有好处哦

**include /etc/nginx/mime.types：**指定在当前文件中包含另一个文件的指令

**default\_type application/octet-stream：**指定默认处理的文件类型可以是二进制

**type\_hash\_max\_size 2048：**混淆数据，影响三列冲突率，值越大消耗内存越多，散列key冲突率会降低，检索速度更快；值越小key，占用内存较少，冲突率越高，检索速度变慢

### 日志配置

**access\_log logs/access.log：**设置存储访问记录的日志

**error\_log logs/error.log：**设置存储记录错误发生的日志

### SSL证书配置

**ssl\_protocols：**指令用于启动特定的加密协议，nginx在1.1.13和1.0.12版本后默认是ssl\_protocols SSLv3 TLSv1 TLSv1.1 TLSv1.2，TLSv1.1与TLSv1.2要确保OpenSSL >= 1.0.1 ，SSLv3 现在还有很多地方在用但有不少被攻击的漏洞。

**ssl prefer server ciphers：**设置协商加密算法时，优先使用我们服务端的加密套件，而不是客户端浏览器的加密套件

### 压缩配置

**gzip：** 是告诉nginx采用gzip压缩的形式发送数据。这将会减少我们发送的数据量。

**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\_static：** 告诉nginx在压缩资源之前，先查找是否有预先gzip处理过的资源。这要求你预先压缩你的文件（在这个例子中被注释掉了），从而允许你使用最高压缩比，这样nginx就不用再压缩这些文件了（想要更详尽的gzip\_static的信息，请点击这里）。

**gzip\_proxied：** 允许或者禁止压缩基于请求和响应的响应流。我们设置为any，意味着将会压缩所有的请求。

**gzip\_min\_length：** 设置对数据启用压缩的最少字节数。如果一个请求小于1000字节，我们最好不要压缩它，因为压缩这些小的数据会降低处理此请求的所有进程的速度。

**gzip\_comp\_level：** 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。我们设置为4，这是一个比较折中的设置。

**gzip\_type：** 设置需要压缩的数据格式。上面例子中已经有一些了，你也可以再添加更多的格式。

### 文件缓存配置

**open\_file\_cache：** 打开缓存的同时也指定了缓存最大数目，以及缓存的时间。我们可以设置一个相对高的最大时间，这样我们可以在它们不活动超过20秒后清除掉。

**open\_file\_cache\_valid：** 在open\_file\_cache中指定检测正确信息的间隔时间。

**open\_file\_cache\_min\_uses：** 定义了open\_file\_cache中指令参数不活动时间期间里最小的文件数。

**open\_file\_cache\_errors：** 指定了当搜索一个文件时是否缓存错误信息，也包括再次给配置中添加文件。我们也包括了服务器模块，这些是在不同文件中定义的。如果你的服务器模块不在这些位置，你就得修改这一行来指定正确的位置。

## sever模块

srever模块配置是http模块中的一个子模块，用来定义一个虚拟访问主机，也就是一个虚拟服务器的配置信息。

server {

listen 80;

server\_name localhost 192.168.1.100;

charset utf-8;

access\_log logs/access.log;

error\_log logs/error.log;

......

}

**server：**一个虚拟主机的配置，一个http中可以配置多个server

**server\_name：**用来指定ip地址或者域名，多个配置之间用空格分隔

**charset：**用于设置www/路径中配置的网页的默认编码格式

**access\_log：**用于指定该虚拟主机服务器中的访问记录日志存放路径

**error\_log：**用于指定该虚拟主机服务器中访问错误日志的存放路径

## location模块

location模块是Nginx配置中出现最多的一个配置，主要用于配置路由访问信息。

在路由访问信息配置中关联到反向代理、负载均衡等等各项功能，所以location模块也是一个非常重要的配置模块。

### 基本配置

location / {

root /nginx/www;

index index.php index.html index.htm;

}

**location /：**表示匹配访问根目录

**root：**用于指定访问根目录时，访问虚拟主机的web目录

**index：**在不指定访问具体资源时，默认展示的资源文件列表

### 反向代理配置

通过反向代理代理服务器访问模式，通过proxy\_set配置让客户端访问透明化。

location / {

proxy\_pass http://localhost:8888;

proxy\_set\_header X-real-ip $remote\_addr;

proxy\_set\_header Host $http\_host;

}

### uwsgi配置

location / {

include uwsgi\_params;

uwsgi\_pass localhost:8888

}

## 负载均衡模块（upstream）

upstream模块主要负责负载均衡的配置，通过默认的轮询调度方式来分发请求到后端服务器。简单的配置方式如下。

upstream name {

ip\_hash;

server 192.168.1.100:8000;

server 192.168.1.100:8001 down;

server 192.168.1.100:8002 max\_fails=3;

server 192.168.1.100:8003 fail\_timeout=20s;

server 192.168.1.100:8004 max\_fails=3 fail\_timeout=20s;

}

**ip\_hash：**指定请求调度算法，默认是weight权重轮询调度，可以指定

**server host:port：**分发服务器的列表配置

**-- down：**表示该主机暂停服务

**-- max\_fails：**表示失败最大次数，超过失败最大次数暂停服务

**-- fail\_timeout：**表示如果请求受理失败，暂停指定的时间之后重新发起请求

Nginx主要配置

静态Http服务器配置

首先，Nginx是一个HTTP服务器，可以将服务器上的静态文件（如HTML、图片）通过HTTP协议展现给客户端。

配置：

server {

listen 80; 　　# 端口

server\_name localhost 192.168.1.100; 　　# 域名

location / {　　　　　　　　　　　　　# 代表这是项目根目录

root /usr/share/nginx/www; 　 # 虚拟目录

}

}

反向代理服务器配置

什么是反向代理？

客户端本来可以直接通过HTTP协议访问某网站应用服务器，如果网站管理员在中间加上一个Nginx，客户端请求Nginx，Nginx请求应用服务器，然后将结果返回给客户端，此时Nginx就是反向代理服务器。

反向代理配置：

server {

listen 80;

location / {

proxy\_pass http://192.168.0.112:8080; 　　# 应用服务器HTTP地址

}

}

既然服务器可以直接HTTP访问，为什么要在中间加上一个反向代理，不是多此一举吗？反向代理有什么作用？继续往下看，下面的负载均衡、虚拟主机，都基于反向代理实现，当然反向代理的功能也不仅仅是这些。

负载均衡配置

当网站访问量非常大，也摊上事儿了。因为网站越来越慢，一台服务器已经不够用了。于是将相同的应用部署在多台服务器上，将大量用户的请求分配给多台机器处理。同时带来的好处是，其中一台服务器万一挂了，只要还有其他服务器正常运行，就不会影响用户使用。Nginx可以通过反向代理来实现负载均衡。

负载均衡配置：

upstream myapp {

server 192.168.0.111:8080; 　　# 应用服务器1

server 192.168.0.112:8080; 　　# 应用服务器2

}

server {

listen 80;

location / {

proxy\_pass http://myweb;

}

}

虚拟主机配置

有的网站访问量大，需要负载均衡。然而并不是所有网站都如此出色，有的网站，由于访问量太小，需要节省成本，将多个网站部署在同一台服务器上。

例如将www.aaa.com和www.bbb.com两个网站部署在同一台服务器上，两个域名解析到同一个IP地址，但是用户通过两个域名却可以打开两个完全不同的网站，互相不影响，就像访问两个服务器一样，所以叫两个虚拟主机。

虚拟主机配置：

server {

listen 80 default\_server;

server\_name \_;

return 444; 　　# 过滤其他域名的请求，返回444状态码

}

server {

listen 80;

server\_name www.aaa.com; 　　# www.aaa.com域名

location / {

proxy\_pass http://localhost:8080; 　　# 对应端口号8080

}

}

server {

listen 80;

server\_name www.bbb.com; 　　# www.bbb.com域名

location / {

proxy\_pass http://localhost:8081; 　　# 对应端口号8081

}

}

在服务器8080和8081分别开了一个应用，客户端通过不同的域名访问，根据server\_name可以反向代理到对应的应用服务器。

虚拟主机的原理是通过HTTP请求头中的Host是否匹配server\_name来实现的，有兴趣的同学可以研究一下HTTP协议。

另外，server\_name配置还可以过滤有人恶意将某些域名指向你的主机服务器。

Nginx开机启动(方法1，适用CentOS7，systemctl管理服务)

CentOS7系统服务脚本目录

用户（user）：用户登录后才能运行的程序，存在用户（user）。

/usr/lib/systemd/user

系统（system）：如需要开机没有登陆情况下就能运行的程序，存在系统服务（system）里。

/usr/lib/systemd/system

编写service脚本

服务文件名以.service结尾：

vim /usr/lib/systemd/system/nginx.service

编写脚本内容（固定格式）：

[Unit]

Description=nginx

After=network.target

[Service]

Type=forking

PIDFile=/usr/install/nginx/logs/nginx.pid

ExecStart=/usr/install/nginx/sbin/nginx

ExecReload=/usr/install/nginx/sbin/nginx -s reload

ExecStop=/usr/install/nginx/sbin/nginx -s stop

PrivateTmp=true

[Install]

WantedBy=multi-user.target

以上路径必须均为绝对路径！！而ExecStart、ExecReload、ExecStop的值也可以是"/etc/init.d"下的自定义的sh脚本文件的绝对路径，我就是采用这种方式来实现 uWSGI 开机启动的：

/etc/init.d目录下的uWSGI服务启动脚本文件"uwsgi-start.sh"：

#!/bin/sh

/pyvenv/bin/uwsgi --ini /pyvenv/src/eduonline/uwsgi.ini;

/etc/init.d目录下的uWSGI服务重启脚本文件"uwsgi-restart.sh"：

#!/bin/sh

/pyvenv/bin/uwsgi --restart /pyvenv/src/eduonline/uwsgi.pid;

/etc/init.d目录下的uWSGI服务停止脚本文件"uwsgi-stop.sh"：

#!/bin/sh

/pyvenv/bin/uwsgi --stop /pyvenv/src/eduonline/uwsgi.pid;

注意：sh脚本中同样采用绝对路径！！保存后，赋予可读可执行权限。然后编写service脚本文件。

设置开机启动（强大的CentOS服务管理工具systemctl）

systemctl enable nginx.service #".service"可省略

# 附其他命令：

systemctl start nginx.service # 启动

systemctl restart nginx.service # 重启，服务会中止一会儿

systemctl reload nginx.service　　　 # 重载服务配置文件，类似于重启，但服务不会中止

systemctl stop nginx.service # 停止

systemctl disable nginx.service # 关闭开机启动

如果提示"Failed to execute operation: Access denied"，输入"systemctl daemon-reexec"可解决。

Nginx开机启动(方法2，适用CentOS7以下)

首先，在Linux系统的"/etc/init.d/"目录下创建nginx脚本文件，使用如下命令：

touch nginx # 创建

vim nginx # 编辑

在脚本中添加如下命令：

View Code

保存脚本文件后对所有用户追加可执行权限：

chmod a+x /etc/init.d/nginx

先将nginx服务加入chkconfig管理列表：

chkconfig --add /etc/init.d/nginx

设置终端模式开机启动：

chkconfig nginx on

————————————————

版权声明：本文为CSDN博主「一个有梦想的混子」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/qq\_33454884/article/details/89212702