Nginx笔记

本⽂档为GeekHour的《30分钟Nginx⼊⻔教程》的配套笔记⽂档，转载请注明出处。 视频地址 ↓

[[Bilibili]](https://space.bilibili.com/102438649)

[[YouTube]](https://www.youtube.com/channel/UCaHGGhmThVehDUapGxl7RVQ)

本⽂档所有内容均来⾃于Nginx官⽹，仅供学习使⽤。

[nginx admin guide](https://docs.nginx.com/nginx/admin-guide)

[nginx documentation](https://nginx.org/en/docs/)

**1.** **Nginx**简介

Nginx是⽬前最流⾏的Web服务器，

最开始是由⼀个叫做igor的俄罗斯的程序员开发的，

2019年3⽉11⽇被美国的F5公司以6.7亿美元的价格收购，

现在Nginx是F5公司旗下的⼀款产品了。

**2.** **Nginx**的版本

Nginx开源版本主要分为两种，⼀种是稳定版，⼀种是主线版。

 主线版( mainline)：主线版是最新的版本，功能会⽐较多，会包含⼀些正在开发中的体

验性模块功能，但是也可能会有⼀些新的bug。

 稳定版(Stable)：稳定版是经过⻓时间测试的版本，不会有太多的bug，也不会包含⼀

些新的功能。

**3.** **Nginx**安装**(**预编译⼆进制包**)**

这种⽅式⽐通过源码编译安装的⽅式要简单快捷得多，只需要输⼊⼀条install命令就可以了。 不同Linux发⾏版的安装⽅式略有不同，下⾯分别介绍⼀下。

**3.1** **CentOS/RHEL/Oracle** **Linux/AlmaLinux/Rocky** **Linux** **repository.**

CentOS系Linux发⾏版可以使⽤yum来安装。

# 1. 安装EPEL仓库

sudo yum install epel - release

# 2. 更新repo

sudo yum update

# 3. 安装nginx

sudo yum install nginx

# 4. 验证安装

sudo nginx -V

也可以通过Nginx的官⽅仓库来安装，这样可以保证安装的是最新的版本。

1. 安装前置依赖

sudo yum install yum -utils

2. 添加nginx仓库

sudo vi /etc/yum . repos .d/nginx . repo

3. 添加以下内容

[nginx -stable]

name=nginx stable repo

baseurl=http://nginx.org/packages/centos/$releasever/$basearch/ gpgcheck=1

enabled=1

gpgkey=https://nginx.org/keys/nginx\_signing.key

module\_hotfixes=true

[nginx -mainline]

name=nginx mainline repo

baseurl=http://nginx.org/packages/mainline/centos/$releasever/$basearch/ gpgcheck=1

enabled=0

gpgkey=https://nginx.org/keys/nginx\_signing.key

module\_hotfixes=true

上⾯的 stable 和 mainline 就分别表示稳定版和主线版，可以根据⾃⼰的需要来选择。

4. 更新repo

sudo yum update

5. 安装nginx

sudo yum install nginx

6. 验证安装

除了使⽤ sudo nginx -V 之外，还可以使⽤下⾯的⽅式来验证：

# 启动Nginx

sudo nginx

curl -I 127.0.0.1

如果能够看到类似下⾯的输出，就表示安装成功了：

HTTP/1 .1 200 OK

Server: nginx/1 .25 .1

**3.2** **Debian/Ubuntu** **repository**

Debian 、Ubuntu系列的Linux发⾏版可以使⽤apt来安装。

# 1. 更新仓库信息

sudo apt -get update

# 2. 安装nginx

sudo apt -get install nginx

# 3. 验证安装

sudo nginx -V

同样也可以从Nginx官⽅仓库来安装。

# 1. 安装前置依赖

sudo apt install curl gnupg2 ca -certificates lsb - release debian -archive - keyring

# 2. 导⼊官⽅Nginx签名密钥

curl https://nginx .org/keys/nginx\_signing .key | gpg --dearmor \

| sudo tee /usr/share/keyrings/nginx -archive -keyring .gpg >/dev/null

# 3. 验证下载的⽂件中包含了正确的密钥

gpg --dry - run --quiet --no -keyring --import --import -options import -show /usr/share/keyrings/nginx -archive -keyring .gpg

# 4. 设置稳定版或者主线版的Nginx包

# 稳定版

echo "deb [signed -by=/usr/share/keyrings/nginx -archive -keyring .gpg] \ <http://nginx.org/packages/debian> `lsb\_ release -cs` nginx" \

| sudo tee /etc/apt/sources .list .d/nginx .list

# 主线版

echo "deb [signed -by=/usr/share/keyrings/nginx -archive -keyring .gpg] \

<http://nginx.org/packages/mainline/debian> `lsb\_ release -cs` nginx" \ | sudo tee /etc/apt/sources .list .d/nginx .list

# 5. 设置仓库优先级，优先使⽤Nginx官⽅仓库

echo -e "Package: \*\nPin: origin nginx .org\nPin: release o=nginx\nPin - Priority: 900\n" \

| sudo tee /etc/apt/preferences .d/99nginx

# 6. 安装nginx

sudo apt update

sudo apt install nginx

# 7. 验证安装

sudo nginx

curl -I 127.0.0.1

**3.3** 从源码编译安装

从源码编译安装的⽅式可以让我们⾃定义Nginx的安装⽬录、模块等，但是安装过程⽐较繁 琐，需要安装⼀些依赖库。

**3.3.1** 安装**PCRE**库

PCRE是Perl Compatible Regular Expressions的缩写，是⼀个Perl库，包括perl兼容的正则表

达式库。

wget github .com/PCRE2Project/pcre2/releases/download/pcre2 -10 .42/pcre2 -

10 .42 .tar .gz

tar -zxf pcre2 -10 .42 .tar .gz

cd pcre2 -10 .42

./configure

**3.3.2** 安装**zlib**库

zlib是⼀个数据压缩库，⽤于Nginx的gzip模块。

wget http://zlib .net/zlib -1 .2 .13 .tar .gz

tar -zxf zlib -1 .2 .13 .tar .gz

cd zlib -1 .2 .13

./configure

make

sudo make install

**3.3.3** 安装**OpenSSL**库

OpenSSL是⼀个强⼤的安全套接字层密码库，⽤于Nginx的SSL模块。

wget http://www .openssl .org/source/openssl -1 .1 .1t .tar .gz

tar -zxf openssl -1 .1 .1t .tar .gz

cd openssl -1 .1 .1t

./Configure darwin64 -x86\_64 -cc --prefix=/usr

make

sudo make install

**3.3.4** 下载**Nginx**源码

下载主线版的Nginx源码：

wget https://nginx .org/download/nginx -1 .23 .4 .tar .gz

tar zxf nginx -1 .23 .4 .tar .gz

cd nginx -1 .23 .4

下载稳定版的Nginx源码：

wget https://nginx .org/download/nginx -1 .24 .0 .tar .gz

tar zxf nginx -1 .24 .0 .tar .gz

cd nginx -1 .24 .0

**3.3.5** 配置编译选项

编译选项可以通过 ./configure --help 来查看。

下⾯是⼀个官⽹的例⼦：

./configure

--sbin -path=/usr/local/nginx/nginx

--conf -path=/usr/local/nginx/nginx .conf

--pid -path=/usr/local/nginx/nginx .pid

--with -pcre= . ./pcre2 -10 .42

--with -zlib= . ./zlib -1 .2 .13

--with -http\_ssl\_module

--with -stream

--with -mail=dynamic

--add -module=/usr/build/nginx - rtmp -module

--add -dynamic -module=/usr/build/3party\_module

|  |  |
| --- | --- |
| 参数(**Parameter**) | 说明(**Description**) |
| --prefix= | 指定安装⽬录 |
| --sbin-path= | 指定Nginx可执⾏⽂件 |
| --conf-path= | 指定配置⽂件位置 |
| --pid-path= | 指定pid⽂件位置 |
| --error-log-path= | 指定错误⽇志⽂件 |
| --http-log-path= | 指定HTTP⽇志⽂件 |
| -- user= | 指定运⾏Nginx的⽤户 |
| --group= | 指定运⾏Nginx的组 |
| --with-pcre= | 指定PCRE库的位置 |
| --with-pcre-jit | 开启PCRE的JIT (Just-in-time compilation)⽀持 |
| --with-zlib= | 指定zlib库的位置 |

**4.** **Nginx**的配置⽂件

Nginx的配置⽂件是 nginx .conf ，⼀般位于 /etc/nginx/nginx .conf 。

可以使⽤ nginx -t 来查看配置⽂件的位置和检查配置⽂件是否正确。

**4.1** 配置⽂件的结构

Nginx的配置⽂件是由⼀系列的指令组成的，每个指令都是由⼀个指令名和⼀个或者多个参数 组成的。

指令和参数之间使⽤空格来分隔，指令以分号 ; 结尾，参数可以使⽤单引号或者双引号来包 裹。

配置⽂件分为以下⼏个部分：

# 全局块

worker\_processes 1;

events {

# events块

}

http {

# http块

server {

# server块

location / {

# location块

}

}

}

4.1.1 全局块

全局块是配置⽂件的第⼀个块，也是配置⽂件的主体部分，主要⽤来设置⼀些影响Nginx服务 器整体运⾏的配置指令，主要包括配置运⾏Nginx服务器的⽤户(组)、允许⽣成的worker process数、进程PID存放路径、⽇志存放路径和类型以及配置⽂件引⼊等。

# 指定运⾏Nginx服务器的⽤户，只能在全局块配置

# 将user指令注释掉，或者配置成nobody的话所有⽤户都可以运⾏

# user [user] [group]

# user nobody nobody;

user nginx;

# 指定⽣成的worker进程的数量，也可使⽤⾃动模式，只能在全局块配置

worker\_processes 1;

# 错误⽇志存放路径和类型

error\_log /var/log/nginx/error .log warn;

# 进程PID存放路径

pid /var/run/nginx .pid;

4.1.2 events块

events {

# 指定使⽤哪种⽹络IO模型，只能在events块中进⾏配置

# use epoll

# 每个worker process允许的最⼤连接数

worker\_connections 1024;

}

4.1.3 http块

http块是配置⽂件的主要部分，包括http全局块和server块。

http {

# nginx 可以使⽤ include指令引⼊其他配置⽂件

include /etc/nginx/mime .types;

# 默认类型，如果请求的URL没有包含⽂件类型，会使⽤默认类型

default\_type application/octet -stream; # 默认类型

# 开启⾼效⽂件传输模式

sendfile on;

# 连接超时时间

keepalive\_timeout 65;

# access\_log ⽇志存放路径和类型

# 格式为： access\_log <path> [format [buffer=size] [gzip[=level]] [flush=time] [if=condition]];

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

# 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"' ;

# 设置sendfile最⼤传输⽚段⼤⼩，默认为0，表示不限制

# sendfile\_max\_chunk 1m;

# 每个连接的请求次数

# keepalive\_ requests 100;

# keepalive超时时间

keepalive\_timeout 65;

# 开启gzip压缩

# gzip on;

# 开启gzip压缩的最⼩⽂件⼤⼩

# gzip\_min\_length 1k;

# gzip压缩级别， 1 -9，级别越⾼压缩率越⾼，但是消耗CPU资源也越多

# gzip\_comp\_level 2;

# gzip压缩⽂件类型

# gzip\_types text/plain application/javascript application/x - javascript text/css application/xml text/javascript application/x -httpd - php image/jpeg image/gif image/png;

# upstream指令⽤于定义⼀组服务器，⼀般⽤来配置反向代理和负载均衡

upstream www .example .com {

# ip\_hash指令⽤于设置负载均衡的⽅式， ip\_hash表示使⽤客户端的IP进⾏hash， 这样可以保证同⼀个客户端的请求每次都会分配到同⼀个服务器，解决了session共享的问题

ip\_hash;

# weight ⽤于设置权重，权重越⾼被分配到的⼏率越⼤

server 192 .168 .50 .11:80 weight=3;

server 192 .168 .50 .12:80;

server 192 .168 .50 .13:80;

}

server {

# 参考server块的配置

}

}

4.1.4 server块

server块是配置虚拟主机的，⼀个http块可以包含多个server块，每个server块就是⼀个虚拟主

机。

server {

# 监听IP和端⼝

# listen的格式为：

# listen [ip]:port [default\_server] [ssl] [http2] [spdy] [proxy\_protocol] [setfib=number] [fastopen=number] [backlog=number];

# listen指令⾮常灵活，可以指定多个IP和端⼝，也可以使⽤通配符

# 下⾯是⼏个实际的例⼦：

# listen 127 .0 .0 .1:80; # 监听来⾃ 127.0.0.1的80端⼝的请求

# listen 80; # 监听来⾃所有IP的80端⼝的请求

# listen \*:80; # 监听来⾃所有IP的80端⼝的请求，同上

# listen 127 .0 .0 .1; # 监听来⾃来⾃ 127.0.0.1的80端⼝，默认端⼝为80

listen

80;

# server\_name ⽤来指定虚拟主机的域名，可以使⽤精确匹配、通配符匹配和正则匹配等

⽅式

# server\_name example .org www .example .org; # 精确匹配

# server\_name \* .example .org; # 通配符匹配

# server\_name ~^www\d+\ .example\ .net$; # 正则匹配

server\_name localhost;

# location块⽤来配置请求的路由，⼀个server块可以包含多个location块，每个

location块就是⼀个请求路由

# location块的格式是：

# location [=|~ |~\* |^~] /uri/ { ... }

# = 表示精确匹配，只有完全匹配上才能⽣效

# ~ 表示区分⼤⼩写的正则匹配

# ~\* 表示不区分⼤⼩写的正则匹配

# ^~ 表示普通字符匹配，如果匹配成功，则不再匹配其他location

# /uri/ 表示请求的URI，可以是字符串，也可以是正则表达式

# { ... } 表示location块的配置内容

location / {

# root指令⽤于指定请求的根⽬录，可以是绝对路径，也可以是相对路径 root /usr/share/nginx/html; # 根⽬录

# index指令⽤于指定默认⽂件，如果请求的是⽬录，则会在⽬录下查找默认⽂件 index index .html index .htm; # 默认⽂件

}

# 下⾯是⼀些location的示例：

location =

root

index }

/ { # 精确匹配请求

/usr/share/nginx/html;

index .html index .htm;

location

root

} location

root

} location

root

} location

root

}

^~ /images/ { # 匹配以/images/开头的请求

/usr/share/nginx/html;

~\* \ .(gif|jpg|jpeg)$ { # 匹配以gif 、jpg或者jpeg结尾的请求

/usr/share/nginx/html;

!~ \ .(gif|jpg|jpeg)$ { # 不匹配以gif 、jpg或者jpeg结尾的请求 /usr/share/nginx/html;

!~\* \ .(gif|jpg|jpeg)$ { # 不匹配以gif 、jpg或者jpeg结尾的请求 /usr/share/nginx/html;

# error\_page ⽤于指定错误⻚⾯，可以指定多个，按照优先级从⾼到低依次查找 error\_page 500 502 503 504 /50x .html; # 错误⻚⾯

location = /50x .html {

root /usr/share/nginx/html;

}

}

**5.** **Nginx**的常⽤命令

|  |  |
| --- | --- |
| nginx | # 启动Nginx |
| nginx -c filename | # 指定配置⽂件 |
| nginx -V | # 查看Nginx的版本和编译参数等信息 |
| nginx -t | # 检查配置⽂件是否正确，也可⽤来定位配置⽂件的位置 |
| nginx -s quit | # 优雅停⽌Nginx |
| nginx -s stop | # 快速停⽌Nginx |
| nginx -s reload | # 重新加载配置⽂件 |
| nginx -s reopen | # 重新打开⽇志⽂件 |

**6.** **Nginx**的常⽤模块

|  |  |
| --- | --- |
| 模块名(**Module** **Name**) | 描述(**Description**) |
| http\_access\_module | 接受或者拒绝特定的客户端请求 |
| http\_auth\_basic\_module | HTTP基本认证，使⽤⽤户名和密码来限制对资源的 访问 |
| http\_autoindex\_module | ⾃动索引，⽤于显示⽬录列表 |
| http\_browser\_module | 从 User -Agent 请求头中获取和识别客户端浏览器 |
| http\_charset\_module | 添加特定的字符集到 Content -Type 响应头中 |
| http\_empty\_gif\_module | 返回⼀个1像素的透明GIF图⽚ |
| http\_fastcgi\_module | FastCGI⽀持 |
| http\_geo\_module | 从IP地址中获取地理位置信息 |
| http\_gzip\_module | Gzip压缩⽀持 |
| http\_limit\_conn\_module | 限制并发连接数 |
| http\_limit\_req\_module | 限制请求速率 |
| http\_map\_module | 从变量中获取值 |
| http\_memcached\_module | Memcached⽀持 |
| http\_proxy\_module | 反向代理⽀持 |

|  |  |
| --- | --- |
| http\_referer\_module | 防盗链 |
| http\_rewrite\_module | URL重写 |
| http\_scgi\_module | 转发请求到SCGI服务器 |
| http\_ssi\_module | 处理和⽀持SSI ( Server Side Includes) |
| http\_split\_clients\_module | 根据客户端IP地址或者其他变量将客户端分配到组  中，⼀般⽤于A/B测试 |
| http\_upstream\_hash\_module | 启⽤⼀致性哈希负载均衡 |
| http\_upstream\_ip\_hash\_module | 启⽤IP哈希负载均衡 |
| http\_upstream\_keepalive\_module | 启⽤⻓连接负载均衡 |
| http\_upstream\_least\_conn\_module | 启⽤最少连接负载均衡 |
| http\_upstream\_zone\_module | 启⽤共享内存负载均衡 |
| http\_userid\_module | 为客户端设置⼀个唯⼀的ID ( UID 、cookie) |
| http\_uwsgi\_module | 转发请求到uWSGI服务器，⼀般⽤于Python应⽤ |