

Nginx基础01：安装和基本使用

文章分类: *Server*; 标签: *Nginx*; 作者: *Hackyle*;

更新时间: *Thu Jan 12 17:47:00 CST 2023*

1. **CentOS环境的安装**
1. 安装前的准备

2. 通过Nginx源码

1. ./configure参数

2. 启动Nginx

3. 安装目录结构

4. 卸载

5. 与systemctl整合

1. 添加到环境变量

3. 通过yum安装

1. 启动Nginx

2. 源码安装与yum安装的区别
2. **控制Nginx**
1. 获取Nginx的PID

2. 信号控制

3. 命令控制

4. 平滑升级

背景

- Nginx是一个高性能的Web服务器，几乎所有的Web服务都需要使用Nginx。
- 关于Nginx的功能特性这里不再赘述，让我们从0开始，**了解Nginx的基本用法，学习它在Web服务中都有哪些应用。**

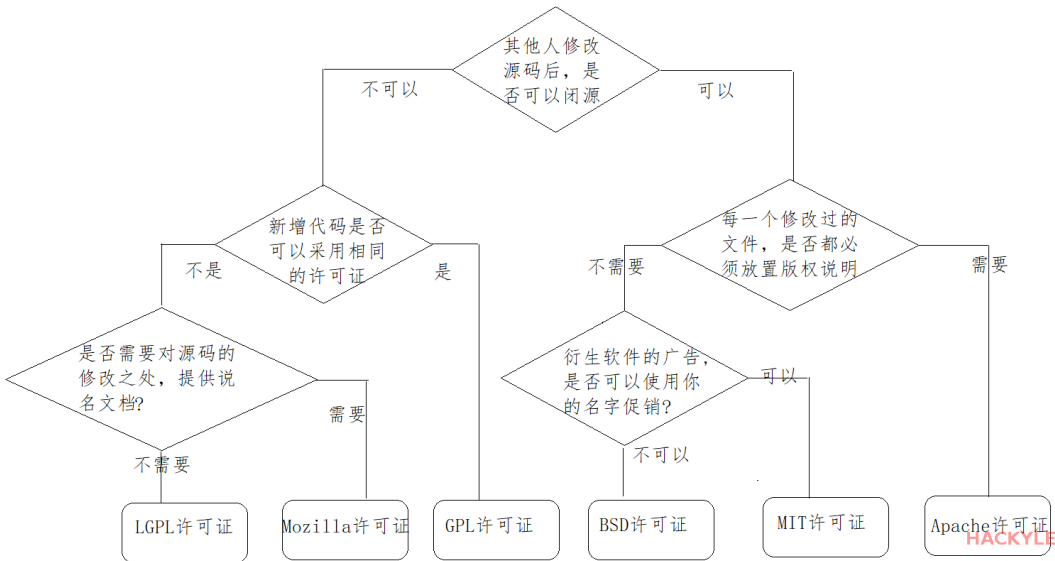
本文主要介绍Nginx的**安装**以及基础的**控制命令**。

内容导航

- [CentOS环境的安装](#)
 - [安装前的准备](#)
 - [通过Nginx源码](#)
 - [./configure参数](#)
 - [启动Nginx](#)
 - [安装目录结构](#)
 - [卸载](#)
 - [与systemctl整合](#)
 - [添加到环境变量](#)
 - [通过yum安装](#)
 - [启动Nginx](#)
 - [源码安装与yum安装的区别](#)
- [控制Nginx](#)
 - [获取Nginx的PID](#)
 - [信号控制](#)
 - [命令控制](#)
 - [平滑升级](#)

Nginx (“engine x”)

- 是一个具有高性能的【HTTP】和【反向代理】的【WEB服务器】，同时也是一个邮件服务器（支持POP3/SMTP/IMAP协议）
- 是由伊戈尔赛索耶夫(俄罗斯人)使用C语言编写的，Nginx的第一个版本是2004年10月4号发布的0.1.0版本
- 基于BSD许可证开源



安装包下载

- 最新版下载: <http://nginx.org/en/download.html>

- 历史版本: <http://nginx.org/download/>

Windows环境的安装

- 下载数据包, 解压后直接使用, 双击exe启动
- 在浏览器输出: <http://localhost>, 即可看到欢迎页面

CentOS环境的安装

为了获得较好的学习环境, 我们应该考虑安装在一个**纯净的虚拟机**或者**云服务器**上。同时, 为了避免CentOS防火墙拦截端口上的数据, 我们考虑将其关闭

关闭防火墙

- `systemctl stop firewalld` 关闭运行的防火墙, 系统重新启动后, 防火墙将重新打开
- `systemctl disable firewalld` 永久关闭防火墙, 系统重新启动后, 防火墙依然关闭
- `systemctl status firewalld` 查看防火墙状态

防火墙控制

- 查询防火墙中指定的端口是否开放: `firewall-cmd --query-port=9001/tcp`
- 开放一个指定的端口: `firewall-cmd --permanent --add-port=9002/tcp` #参数"--permanent"表示永久开放, 不加表示临时开放
- 批量添加开发端口: `firewall-cmd --permanent --add-port=9001-9003/tcp`
- 如何移除一个指定的端口: `firewall-cmd --permanent --remove-port=9003/tcp`
- 重新加载: `firewall-cmd --reload`

安装前的准备

停用selinux

1. selinux(security-enhanced linux), 美国安全局对于强制访问控制的实现, 在6内核以后的版本中, selinux已经成功内核中的一部分。
2. 查看selinux: `sestatus`
3. 为了方便, 直接关闭: `vim /etc/selinux/config`
4. 重启系统

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
#SELINUX=enforcing
SELINUX=disabled
# SELINUXTYPE= can take one of three values:
#   targeted - Targeted processes are protected.
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

安装GCC

- 原因: Nginx是使用C语言编写的程序
- 检查: `gcc --version`
- 安装: `yum install -y gcc`

安装PCRE

- 原因: Nginx在编译过程中需要使用到PCRE库 (perl Compatible Regular Expressoin 兼容正则表达式库), 因为在Nginx的Rewrite模块和http核心模块都会使用到PCRE正则表达式语法。
- 安装: `yum install -y pcre pcre-devel`
- 检查: `rpm -qa pcre pcre-devel`

zlib库

- 原因: 提供了开发人员的压缩算法, 在Nginx的各个模块中需要使用gzip压缩, 所以我們也需要提前安装其库及源代码zlib和zlib-devel
- 安装: `yum install -y zlib zlib-devel`
- 检查: `rpm -qa zlib zlib-devel`

OpenSSL

- 是一个开放源代码的软件库包, 应用程序可以使用这个包进行安全通信, 并且避免被窃听。
- SSL:Secure Sockets Layer安全套接协议的缩写, 可以在Internet上提供秘密性传输, 其目标是保证两个应用间通信的保密性和可靠性。

- 原因：在Nginx中，如果服务器需要提供安全网页时就需要用到OpenSSL库，所以我们需要对OpenSSL的库文件及它的开发安装包进行一个安装。
- 安装：yum install -y openssl openssl-devel
- 检查：rpm -qa openssl openssl-devel
- 全部安装：yum install -y gcc pcre pcre-devel zlib zlib-devel openssl openssl-devel

通过Nginx源码

下载：wget http://nginx.org/download/nginx-1.22.0.tar.gz

解压：tar -xzf nginx-1.22.0.tar.gz

进入nginx目录后，执行：./configure

编译：make

安装：make install

查看Nginx的默认安装位置：whereis nginx

```
[root@localhost nginx-1.22.0]# whereis nginx
nginx: /usr/local/nginx
```

Nginx安装包目录说明

- auto:存放的是编译相关的脚本
- CHANGES:版本变更记录
- ru:俄罗斯文的版本变更记录
- conf:nginx默认的配置文件的
- configure:nginx软件的自动脚本程序,是一个比较重要的文件，作用如下：
 - 检测环境及根据环境检测结果生成C代码
 - 生成编译代码需要的Makefile文件
- contrib:存放的是几个特殊的脚本文件，其中README中对脚本有着详细的说明
- html:存放的是Nginx自带的两个html页面，访问Nginx的首页和错误页面
- LICENSE:许可证的相关描述文件
- man:nginx的man手册
- README:Nginx的阅读指南
- src:Nginx的源代码

./configure参数

configure参数的作用：用来生成 Makefile，为下一步的**编译做准备**，可以通过在 configure 后加上参数来对**安装进行控制**

查看所有“ ./configure” 时的额外参数：./configure --help

参数类别

- 基础配置
 - --prefix=PATH 指定安装根目录，默认是/usr/local/nginx。此设置会更改其他配置目录的相对路径
 - --sbin-path=PATH 可执行文件的路径，默认为<prefix>/sbin/nginx
 - --modules-path=PATH 模块存储路径
 - --conf-path=PATH 配置文件的路径，默认为<prefix>/conf/nginx.conf
 - --pid-path=PATH pid文件的存放路径，默认存放在<prefix>/logs/nginx.pid，是一个存放nginx的master进程ID的纯文本文件，刚安装的时候不会生成，nginx启动的时候会自动生成
 - --lock-path=PATH 指向Nginx锁文件的存放路径,默认值为<prefix>/logs/nginx.lock
 - --with-ld-opt 加入第三方链接时需要的参数。编译之后nginx最终的可执行二进制文件是由编译后的目标文件和一些第三方的库链接生成的。如果想要将某个库链接到nginx中，就需要指定--with-ld-opt=目标库名-目标库路径
 - --with-debug 将nginx需要打印debug调试级别日志的代码编译进nginx，这样才可以通过修改配置文件将调试日志打印出来，便于定位服务问题
- 日志文件
 - --http-log-path=PATH 指向访问日志文件的路径,默认值为<prefix>/logs/access.log
 - --error-log-path=PATH 错误日志文件存储位置，默认值为<prefix>/logs/error.log
- 加载额外的模块
 - --with-select_module
- 排除额外的模块
 - --without-stream_limit_conn_module

实例：./configure --prefix=/usr/local/nginx \

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

```
--modules-path=/usr/local/nginx/modules \
--conf-path=/usr/local/nginx/conf/nginx.conf \
--error-log-path=/usr/local/nginx/logs/error.log \
--http-log-path=/usr/local/nginx/logs/access.log \
--pid-path=/usr/local/nginx/logs/nginx.pid \
--lock-path=/usr/local/nginx/logs/nginx.lock
```

启动Nginx

通过 “whereis nginx” 获得Nginx的目录（默认/usr/local/nginx）并进入

进入sbin，执行以下命令即可启动：./nginx

进入浏览器访问：<http://localhost>

安装目录结构

通过 “whereis nginx” 获得Nginx的目录（默认/usr/local/nginx）并进入，执行tree命令

```
[root@localhost nginx]# tree
.
├── conf
│   ├── fastcgi.conf
│   ├── fastcgi.conf.default
│   ├── fastcgi_params
│   ├── fastcgi_params.default
│   ├── koi-utf
│   ├── koi-win
│   ├── mime.types
│   ├── mime.types.default
│   ├── nginx.conf
│   ├── nginx.conf.default
│   ├── scgi_params
│   ├── scgi_params.default
│   ├── uwsgi_params
│   ├── uwsgi_params.default
│   └── win-utf
├── html
│   ├── 50x.html
│   └── index.html
├── logs
├── sbin
│   └── nginx
└──
```

4 directories, 18 files

conf: nginx所有配置文件目录

- CGI
 - CGI(Common Gateway Interface)通用网关【接口】，主要解决的问题是从客户端发送一个请求和数据，服务端获取到请求和数据后可以调用调用CGI【程序】处理及相应结果给客户端的一种标准规范。
 - conf: fastcgi相关配置文件
 - conf.default: fastcgi.conf的备份文件
 - fastcgi_params: fastcgi的参数文件
 - default: fastcgi的参数备份文件
 - scgi_params: scgi的参数文件
 - default: scgi的参数备份文件
 - uwsgi_params: uwsgi的参数文件
 - default: uwsgi的参数备份文件
- **mime.types**: 记录的是HTTP协议中的**Content-Type**的值和文件后缀名的对应关系
- mime.types.default: mime.types的备份文件
- **nginx.conf**: 这个是Nginx的**核心配置文件**，这个文件非常重要，也是我们即将要学习的重点
- nginx.conf.default: nginx.conf的备份文件

- 编码：koi-utf、koi-win、win-utf这三个文件都是与编码转换映射相关的配置文件，用来将一种编码转换成另一种编码

html: 存放nginx自带的两个静态的html页面

- html:访问失败后的失败页面
- html:成功访问的默认首页

logs: 记录入门的文件，当nginx服务器启动后，这里面会有 access.log error.log 和nginx.pid三个文件出现。

sbin:是存放执行程序文件nginx，用来控制Nginx的启动和停止等相关的命令。

卸载

卸载（进入安装目录，通过“whereis nginx” 获取）

- 需要将nginx的进程关闭：./nginx -s stop
- 将安装的nginx进行删除：rm -rf /usr/local/nginx
- 回到之前执行make的目录，将安装包之前编译的环境清除掉：make clean

与systemctl整合

基于源码安装的nginx，默认情况下是不能通过systemctl操作的

只能进入到其安装目录下的sbin子目录中的nginx二进制文件控制

整合systemctl

- 在`/usr/lib/systemd/system`目录下添加service文件
- 编辑文件内容：

```

1  [Unit]
2  Description=nginx web service
3  Documentation=http://nginx.org/en/docs/
4  After=network.target
5
6  [Service]
7  Type=forking
8  PIDFile=/usr/local/nginx/logs/nginx.pid
9  ExecStartPre=/usr/local/nginx/sbin/nginx -t -c /usr/local/nginx/conf/nginx.conf
10 ExecStart=/usr/local/nginx/sbin/nginx
11 ExecReload=/usr/local/nginx/sbin/nginx -s reload
12 ExecStop=/usr/local/nginx/sbin/nginx -s stop
13 PrivateTmp=true
14
15 [Install]
16 WantedBy=default.target

```

- 添加完成后如果权限有问题需要进行权限设置：chmod 755 /usr/lib/systemd/system/nginx.service
- 使用系统命令来操作nginx
 - 启动: systemctl start nginx
 - 停止: systemctl stop nginx
 - 重启: systemctl restart nginx
 - 重新加载配置文件: systemctl reload nginx
 - 查看nginx状态: systemctl status nginx
 - 开机启动: systemctl enable nginx

添加到环境变量

背景：每次执行nginx的二进制指令时，都要切换到其安装目录下的sbin目录，很麻烦

解决方案：将其添加到环境变量中，让Linux自己去寻找指令所在位置

- 在/etc/profile文件汇总追加：export PATH=\$PATH:/usr/local/nginx/sbin
- 使之立即生效：source /etc/profile
- 测试：nginx -v

通过yum安装

安装工具: yum install -y yum-utils

添加yum: vim /etc/yum.repos.d/nginx.repo

```
1  [nginx-stable]
2  name=nginx stable repo
3  baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
4  gpgcheck=1
5  enabled=1
6  gpgkey=https://nginx.org/keys/nginx_signing.key
7  module_hotfixes=true
8
9  [nginx-mainline]
10 name=nginx mainline repo
11 baseurl=http://nginx.org/packages/mainline/centos/$releasever/$basearch/
12 gpgcheck=1
13 enabled=0
14 gpgkey=https://nginx.org/keys/nginx_signing.key
15 module_hotfixes=true
```

查看是否安装: yum list | grep nginx

安装: yum install -y nginx

查看安装: whereis nginx

启动Nginx

查看nginx的安装位置: whereis nginx

编辑Nginx配置文件: %nginx安装目录%/nginx.conf, 添加一个虚拟主机:

```
14 http
15 include      /etc/nginx/mime.types;
16 default_type application/octet-stream;
17
18 log_format   main '$remote_addr - $remote_user [$time_local] "$request" '
19                  '$status $body_bytes_sent "$http_referer" '
20                  '"$http_user_agent" "$http_x_forwarded_for"';
21
22 access_log   /var/log/nginx/access.log main;
23
24 sendfile     on;
25 #tcp_nopush  on;
26
27 keepalive_timeout 65;
28
29 #gzip        on;
30
31 include      /etc/nginx/conf.d/*.conf;
32
33 server {
34     listen 8000;
35
36     location / {
37         root /usr/share/nginx/html;
38         index index.html;
39     }
40 }
41
```

HACKYLE

访问: <http://localhost:8000>, 获取Nginx欢迎主页

源码安装与yum安装的区别

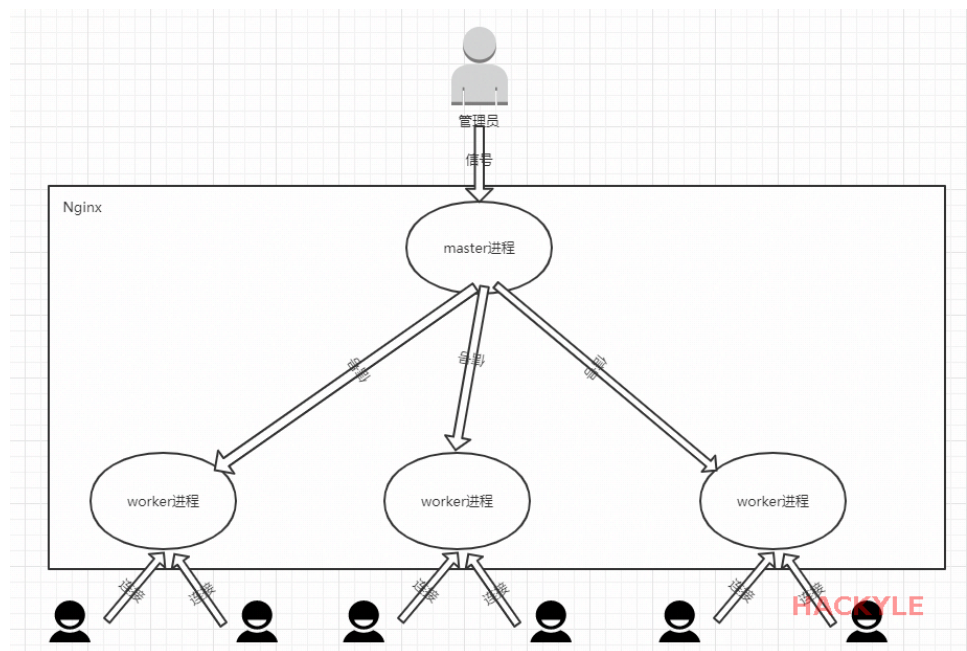
- 源码安装: 所有数据都放在一个文件夹内 (默认/usr/local/nginx)
- yum安装: 将数据分散在操作系统的不同位置, 符合Linux数据分离的思想 (日志文件放在/var/log/, 配置文件放在/etc)


```
[root@HackyleCent-0 ~]# nginx -V
nginx version: nginx/1.20.2
built by gcc 4.8.5 20150623 (Red Hat 4.8.5-44) (GCC)
built with OpenSSL 1.0.2k-fips 26 Jan 2017
TLS SNI support enabled
configure arguments: --prefix=/etc/nginx --sbin-path=/usr/sbin/nginx --modules-path=/usr/lib64/nginx/modules --conf-path=/etc/nginx/nginx.conf --error-log-path=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --pid-path=/var/run/nginx.pid --lock-path=/var/run/nginx.lock --http-client-body-temp-path=/var/cache/nginx/client_temp --http-proxy-temp-path=/var/cache/nginx/proxy_temp --http-fastcgi-temp-path=/var/cache/nginx/fastcgi_temp --http-uwsgi-temp-path=/var/cache/nginx/uwsgi_temp --http-scgi-temp-path=/var/cache/nginx/scgi_temp --user=nginx --group=nginx --with-compat --with-file-aio --with-threads --with-http_addition_module --with-http_auth_request_module --with-http_dav_module --with-http_flv_module --with-http_gunzip_module --with-http_gzip_static_module --with-http_mp4_module --with-http_random_index_module --with-http_realip_module --with-http_secure_link_module --with-http_slice_module --with-http_ssl_module --with-http_stub_status_module --with-http_sub_module --with-http_v2_module --with-mail --with-mail_ssl_module --with-stream --with-stream_realip_module --with-stream_ssl_module --with-stream_ssl_preread_module --with-cc-opt='-O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protector-strong --param=ssp-buffer-size=4 -grecord-gcc-switches -m64 -mtune=generic -fPIC' --with-ld-opt='-Wl,-z,relro -Wl,-z,now -pie'
[root@HackyleCent-0 ~]#
```

控制Nginx

Nginx默认采用的是多进程的方式来工作的

- Nginx后台进程中包含一个master进程和多个worker进程，master进程主要用来管理worker进程，包含接收外界的信息，并将接收到的信号发送给各个worker进程，监控worker进程的状态，当worker进程出现异常退出后，会自动重新启动新的worker进程。
- worker进程则是专门用来处理用户请求的，各个worker进程之间是平等的并且相互独立，处理请求的机会也是一样的。



查看Nginx开启的进程：ps -ef | grep "nginx"

```
[root@localhost ~]# ps -ef | grep "nginx"
root      17771    1    0 05:31 ?        00:00:00 nginx: master process ./nginx
nobody    17773    17771    0 05:31 ?        00:00:00 nginx: worker process
root      18125    17927    0 05:41 pts/1    00:00:00 grep --color=auto nginx
[root@localhost ~]#
```

控制Nginx，实际上是在操作master进程：打开、关闭、重启

获取Nginx的PID

获取进程号

- ps aux | grep nginx
- ps -ef | grep nginx
- 注意：由于前文中已经说过了，控制Nginx主要是操作master进程，所以带有“master process”的进程PID才是Nginx的进程PID

```
[root@Hackyle log]# ps aux | grep nginx
root      1011  0.0  0.1  41112 1304 ?        Ss   2022   0:00 nginx: master process /usr/sbin/nginx
nginx     1012  0.0  0.5  41832 4832 ?        S    2022   0:14 nginx: worker process
root     28270  0.0  0.1  112812  980 pts/0    R+   16:09   0:00 grep --color=auto nginx
[root@Hackyle log]# ps -ef | grep nginx
root      1011      1  0  2022 ?        00:00:00 nginx: master process /usr/sbin/nginx
nginx     1012    1011  0  2022 ?        00:00:14 nginx: worker process
root     28408 25944  0 16:09 pts/0    00:00:00 grep --color=auto nginx
[root@Hackyle log]#
```

第二列就是进程的PID

通过Nginx的PID文件获取master的进程号

- 使用find命令全局查找pid文件：**find / -name nginx.pid**
 - /表示从根目录下开始查找
 - -name后面指定要查找的文件名
- 将反引号内命令的执行结果作为另一条命令的参数：**cat `find / -name nginx.pid`**
 - 注意是反引号、飘号，ESC按键下的那个按键
 - 含义是：将飘号内的命令find / -name nginx.pid的执行结果，作为cat的参数

```
[root@Hackyle log]# find / -name nginx.pid
/run/nginx.pid
[root@Hackyle log]# cat /run/nginx.pid
1011
[root@Hackyle log]# cat `find / -name nginx.pid`
1011
[root@Hackyle log]#
```

Linux中将上一条命令的结果作为下一条命令的参数

- 使用**xargs**：查找Nginx的PID文件，然后将查找结果交给cat：find / -name nginx.pid | xargs cat
- 使用**反斜杠、飘号**：cat `find / -name nginx.pid`
- 使用**find exec命令**
 - find / -name nginx.pid -exec cat {} \;
 - 这里的{}和\是成对使用的。将find的查找结果作为参数
- 使用**\$()**
 - cat \$(find / -name nginx.pid)
 - 将\$()的执行结果作为cat的参数

```
[root@Hackyle log]# find / -name nginx.pid | xargs cat
5262
[root@Hackyle log]# cat `find / -name nginx.pid`
5262
[root@Hackyle log]# find / -name nginx.pid -exec cat {} \;
5262
[root@Hackyle log]# cat $(find / -name nginx.pid)
5262
[root@Hackyle log]#
```

信号控制

控制语法：kill -信号类型 PID进程号

信号类型	作用
TERM/INT	立即关闭整个服务
QUIT	"优雅"地关闭整个服务 master进程会控制所有的work进程不再接收新的请求，等所有请求处理完后，再把进程都关掉。
HUP	重读配置文件并使用服务对新配置项生效
USR1	重新建立日志文件，可以用来进行日志切割
USR2	平滑升级到最新版的nginx
WINCH	所有子进程不在接收处理新连接，相当于给work进程发送QUIT指令

发送**TERM/INT**信号给master进程，会将Nginx服务立即关闭。

```
kill -TERM PID
```

```
kill -TERM $(cat `find / -name nginx.pid`)
```

发送**QUIT**信号给master进程，master进程会控制所有的work进程不再接收新的请求，等所有请求处理完后，再把进程都关闭掉。

```
kill -QUIT PID
```

```
kill -QUIT $(cat `find / -name nginx.pid`)
```

发送**HUP**信号给master进程，master进程会把控制旧的work进程不再接收新的请求，等处理完请求后将旧的work进程关闭掉，然后根据nginx的配置文件重新启动新的work进程

```
kill -HUP PID / kill -HUP $(cat `find / -name nginx.pid`)
```

发送**USR1**信号给master进程，告诉Nginx重新开启日志文件

```
kill -USR1 PID / kill -USR1 $(cat `find / -name nginx.pid`)
```

发送**USR2**信号给master进程，告诉master进程要平滑升级

1. 这个时候，会重新开启对应的master进程和work进程，整个系统中将会有两个master进程，并且新的master进程的PID会被记录在/run/nginx.pid。通过命令“find / -name nginx.pid”获取pid文件的位置
2. 而之前的旧的master进程PID会被记录在/run/nginx.pid.oldbin文件中，接着再次发送QUIT信号给旧的master进程，让其处理完请求后再进行关闭

```
kill -USR2 PID / kill -USR2 $(cat `find / -name nginx.pid`)
```

```
kill -QUIT PID / kill -QUIT $(cat `find / -name nginx.pid.oldbin`)
```

发送**WINCH**信号给master进程，让master进程控制不让所有的work进程在接收新的请求了，请求处理完后关闭work进程。注意master进程不会被关闭掉

```
kill -WINCH PID / kill -WINCH $(cat `find / -name nginx.pid`)
```

命令控制

通过Nginx安装目录下的sbin下的可执行文件nginx来进行Nginx状态的控制

通过“whereis nginx”获得Nginx的目录（默认/usr/local/nginx）并进入sbin，执行以下命令即可启动：./nginx

./nginx -h 查看帮助信息

./nginx -s 信号类型

- stop[快速关闭，类似于TERM/INT信号的作用]
- quit[优雅的关闭，类似于QUIT信号的作用]
- reopen[重新打开日志文件类似于USR1信号的作用]
- reload[类似于HUP信号的作用]

./nginx -t: 测试nginx的配置文件语法是否正确并退出

./nginx -T: 测试nginx的配置文件语法是否正确并列出用到的配置文件信息然后退出

./nginx -p: prefix, 指定Nginx的prefix路径, (默认为: /usr/local/nginx/)

./nginx -c: filename, 指定Nginx的配置文件路径, (默认为: conf/nginx.conf)

./nginx -g: 用来补充Nginx配置文件, 向Nginx服务指定启动时应用全局的配置

Examples

- 修改了配置文件后进行正确性检查: ./nginx -t
- 对指定的配置文件进行正确性检查: ./nginx -c 配置文件路径 -t

平滑升级

不关闭Nginx的情况下进行升级

需求：将Nginx-1.14.2升级到Nginx-1.16.1，要求Nginx不能中断提供服务。

主要思想

- Stage1：编译高版本，获取二进制文件
 - 将高版本的Nginx像安装一样，进行参数配置和编译，但不执行安装
 - 编译后数据包文件夹中的objs目录下有nginx二进制文件
- Stage2：将高版本的nginx二进制文件替换原来低版本的文件
- Stage3：发送**USR2**信号给master进程，告诉master进程要平滑升级
 - 这个时候，会重新开启对应的master进程和work进程，整个系统中将会有两个master进程，并且新的master进程的PID会被记录在/run/nginx.pid
 - 而之前的旧的master进程PID会被记录在/run/nginx.pid.oldbin文件中，接着再次发送QUIT信号给旧的master进程，让其处理完请求后再进行关闭
 - kill -USR2 PID / kill -USR2 `cat /run/nginx.pid`
 - kill -QUIT PID / kill -QUIT `cat /run/nginx.pid.oldbin`

升级前准备：将Nginx1.16.1进行参数配置和编译，不需要进行安装。

1. 进入数据包目录
2. ./configure
3. make

方案一：使用Nginx服务信号进行升级

- 第一步:将14.2版本的sbin目录下的nginx进行备份
 - cd /usr/local/nginx/sbin
 - mv nginx nginxold
- 第二步:将16.1安装目录编译后的objs目录下的nginx文件，拷贝到原来/usr/local/nginx/sbin目录下
 - cd ~/nginx/core/nginx-1.16.1/objs
 - cp nginx /usr/local/nginx/sbin
- 第三步:发送信号USR2给Nginx的14.2版本对应的master进程: kill -USR2 `cat /run/nginx.pid`
- 第四步:发送信号QUIT给Nginx的14.2版本对应的master进程: kill -QUIT `more /run/nginx.pid.oldbin`

方案二:使用Nginx安装目录的make命令

- 第一步:将14.2版本的sbin目录下的nginx进行备份
 - cd /usr/local/nginx/sbin
 - mv nginx nginxold
- 第二步:将16.1安装目录编译后的objs目录下的nginx文件，拷贝到原来/usr/local/nginx/sbin目录下
 - cd ~/nginx/core/nginx-1.16.1/objs
 - cp nginx /usr/local/nginx/sbin
- 第三步:进入到安装目录，执行make upgrade
- 第四步:查看是否更新成功: ./nginx -v

版权声明：非明确标注皆为原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上本文链接及此声明。
原文链接：<https://blog.hackyle.com/article/server/hello-nginx>


留下你的评论

Name:

Email:


Link:

File Edit View Format Tools Table Help



Input comment, please

p

0 words 

SUBMIT

RESET

© Copy Right: 2022 HACKYLE. All Rights Reserved

Designed and Created by HACKYLE SHAW

备案号：浙ICP备20001706号-2