### nginx操作手册

**前言：**

①蓝色字体表示命令行命令，正式执行时不要复制命令前面的 #号，#号只是提示符

②#绿色字体表示注释，有时注释太多就不用绿色表示了

③本文档以centos7为基础做操作命令示例；默认是在centos7上操作的，后来有新增centos8的一些特性命令（会指明是★centos8的，没有特别指明的则为centos7的）

④注意：本文档的所有操作请先在测环境进行实践，请不要直接在真实的服务器中操作！

**版权声明**：

本文档以开源的形式发布，所有条款如下：

1. 无担保：作者不保证文档内容的准确无误，亦不承担由于使用此文档所导致的任何后果

2. 自由使用：任何人可以出于任何目的而自由地 阅读/链接/打印/转载/引用/再创作 此文档，无需任何附加条件

若您 阅读/链接/打印/转载/引用/再创作 本文档，则说明接受以上2个条款。

作者：李茂福

更新日期：2023-07-26

**★第1章、yum安装Nginx**

**★添加nginx的yum源**

Centos7自带的yum repo仓库没有nginx安装包，所以要添加nginx的yum repo源文件，

先去官网看看repo的文件名称：http://nginx.org/packages/centos/7/noarch/RPMS/

找nginx-release-centos-7-xxx.rpm的文件，并复制链接，再下载到centos7系统里，最后在centos7系统里安装此rpm包

# rpm -ivh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.ngx.noarch.rpm

#然后在 /etc/yum.repos.d/目录下多了一个nginx.repo仓库源文件；如果安装不了这个rpm，我们也可以直接创建nginx.repo文件，内容如下：

[nginx-stable]

name=nginx-stable-repo

baseurl=http://nginx.org/packages/centos/7/$basearch/

gpgcheck=1

enable=1

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

# yum search nginx --showduplicates #搜索时显示版本号

# yum install nginx #默认安装最新版本

#yum install nginx-1.18.0-1.el7.ngx.x86\_64 #也可指定目标版本

# systemctl enable nginx

# nginx -v #查看nginx版本

程序路径：/usr/sbin/nginx

主配置文件：/etc/nginx/nginx.conf

**小结**：

使用yum安装的nginx默认是做成了系统服务，可以直接用systemctl命令操作，所以不建议再直接调用nginx命令了。使用yum安装的nginx主程序路径和源码安装的路径不一样，主配置文件所在地方也不一样。

**★第2章、yum安装Openresty**

**#下载openresty的yum源文件**

# cd /etc/yum.repos.d/

# wget https://openresty.org/package/centos/openresty.repo

#如果下载失败，也可手动创建这个openresty.repo文件，内容如下：

[openresty]

name=openresty-repo

baseurl=https://openresty.org/package/centos/$releasever/$basearch

skip\_if\_unavailable=False

gpgcheck=1

repo\_gpgcheck=0

gpgkey=https://openresty.org/package/pubkey.gpg

enabled=1

enabled\_metadata=1

# yum install openrsty

# yum install openrsty-resty

# yum install openrsty-opm

**#也可安装指定的版本：**

具体版本名称可在https://openresty.org/package/centos/7/x86\_64查看，先卸载刚刚安装的版本，再安装指定版本

# yum install openresty-1.17.8.2-1.el7.x86\_64

# yum install openresty-resty-1.17.8.2-1.el7.x86\_64

# yum install openresty-opm-1.17.8.2-1.el7.x86\_64

#whereis openrsty

**#启动openresty**

使用yum安装的程序建议只用systemctl命令去操作

# systemctl enable openresty

# systemctl start openresty

程序路径为 /usr/bin/openresty是一个链接，指向 /usr/local/openresty/nginx/sbin/nginx

主配置文件 /usr/local/openresty/nginx/conf/nginx.conf

使用yum安装的openresty自动做成了服务，服务文件在：

/usr/lib/systemd/system/openresty.service

**#openresty启动失败，缺少libssl.so.1.1**

因为我们的centos7系统可能已经安装了openssl，所以在使用yum安装openresty时，它就不下载openssl依赖了，但系统之前安装的openssl版本较低，（不含libssl.so.1.1）而openresty要求的ssl版本较高，所以启动失败。解决办法为 安装openssl11-libs

而centos7自带的yum仓库源没有openssl11-libs软件包，所以要添加epel的repo

# yum install epel-release #添加epel的源

# vi /etc/yum.repos.d/epel.repo #也可手动创建epel.repo文件，内容如下：

[epel]

name=epel-repo

metalink=https://mirrors.fedoraproject.org/metalink?repo=epel-7&arch=$basearch

failovermethod=priority

enabled=1

gpgcheck=1

gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7

# yum search openssl11-libs

# yum install openssl11-libs

#systemctl restart openresty #重启openresty

**★第3章、centos7源码安装Nginx**

**#首先在Centos7上安装编译器和依赖环境**

# yum install gcc-c++ gcc pcre pcre-devel zlib zlib-devel openssl openssl-devel

#gcc/gcc-c++是编译器，nginx的http模块要用pcre来解析正则表达式，nginx要用zlib对http内容进行压缩/解压缩，nginx还要用到ssl模块（openssl）

**#下载nginx源代码并编译安装**

在https://nginx.org/en/download.html 查看各个版本的下载地址，选择目标版本，复制下载地址，然后在centos7上下载（或者在其他地方下载再复制到centos7系统里）

# wget https://nginx.org/download/nginx-1.18.xx.tar.gz #下载源代码压缩包

# tar -xf nginx-1.18.xx.tar.gz #解压源码包

# cd nginx-1.18.xx #进入解压目录

# ./configure --prefix=/usr/local/nginx \

--with-http\_stub\_status\_module \

--with-http\_ssl\_module \

--with-http\_gzip\_static\_module \

--with-stream \

--with-stream\_ssl\_module

#make && make install #编译并安装

# whereis nginx #查看nginx程序在哪个目录

# cd /usr/local/nginx/sbin/ #nginx程序文件默认是在这个目录里

# ls -l

# cat /usr/local/nginx/conf/nginx.conf #主配置文件

**3.nginx的启动及重启**

# cd /usr/local/nginx/sbin/ #进入程序目录下

# ./nginx -t #检查配置语法

# ./nginx #启动nginx程序

# ./nginx -s stop #停止运行，快速停止，不保存相关信息

# ./nginx -s quit #停止运行，正常退出，保存相关信息

# ./nginx -s reload #重新加载配置，不中断服务

#重启服务，可以先退出再启动：

# ./nginx -s quit

# ./nginx

之所以要在程序名前加上点和斜杠 ./ 是因为linux默认不会在当前目录下去找程序，所以 ./ 表示当前目录，可将nginx的程序所在目录加入Path环境变量里，以后就不用加./

**#把nginx做成系统服务开机自启**

使用源码安装的nginx默认没有做成系统的服务，可以把nginx程序配置成随开机自启，创建一个名为nginx.service的文件，内容如下：

[Unit]

Description=nginx web server

After=network-online.target remote-fs.target nss-lookup.target

Wants=network-online.target

[Service]

Type=forking

PIDFile=/var/run/nginx.pid

ExecStart=/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf

ExecReload=/bin/kill -s HUP $MAINPID

ExecStop=/bin/kill -s TERM $MAINPID

[Install]

WantedBy=multi-user.target

保存，并将文件复制到/etc/systemd/system/目录下

# cp nginx.service /etc/systemd/system/nginx.service

# systemctl daemon-reload

然后就可以使用systemctl命令对nginx服务进行操作了，比如

# systemctl enable nginx #设置为开机自启

# systemctl start/restart/reload nginx #启动，重启，重加载配置

**注意**：将nginx做成服务后，就推荐只用systemctl命令对其进行操作（启动，重启等），而不要再使用直接调用Nginx程序，不然2者会有冲突，即把nginx设置成随开机自启和做成系统服务（以上第4、5步骤）只能选一个，只能用一个。

不做成系统服务时，直接调用./nginx命令，做成系统服务时，只用systemctl命令去操作。

**★第4章、centos7源码安装Openresty**

**#首先在Centos7上安装编译器和依赖环境**

# yum install gcc-c++ gcc pcre pcre-devel zlib zlib-devel openssl \

openssl-devel readline-devel curl perl

**#下载Openresty源代码并安装**

在https://openresty.org/en/download.html 查看各个版本的下载地址，选择目标版本，复制下载地址，然后在centos7上下载（或者在其他地方下载再复制到centos7系统里）

# wget -c https://openresty.org/download/openresty-1.17.8.1.tar.gz #下载源码包

# tar -xf openresty-1.17.8.1.tar.gz

# cd openresty-1.17.8.1

# ./configure --prefix=/usr/local/openresty \

--with-luajit \

--with-http\_stub\_status\_module \

--with-http\_ssl\_module \

--with-http\_gzip\_static\_module \

--with-stream \

--with-stream\_ssl\_module \

--with-http\_realip\_module

# gmake && gmake install && gmake clean #使用gmake编译并安装

主程序名也是nginx，位于 /usr/local/openresty/nginx/sbin/目录下

然后把这个nginx文件做了个链接，/usr/local/openresty/bin/openresty

主配置文件为 /usr/local/openresty/nginx/conf/nginx.conf

**#Openresty的启动及重启**

# cd /usr/local/nginx/sbin/ #进入程序目录下

# ./nginx -t #检查配置语法

# ./nginx #启动nginx程序

# ./nginx -s stop #停止运行，快速停止，不保存相关信息

# ./nginx -s quit #停止运行，正常退出，保存相关信息

# ./nginx -s reload #重新加载配置，不中断服务

#重启服务，可以先退出再启动：

# ./nginx -s quit

# ./nginx

#或者在/usr/local/openresty/bin/目录下

# ./openresty

#其他的操作可参考nginx的

若做成系统服务，则不要混用./nginx命令和 systemctl命令

\***编译安装可能要用到的模块**

# ./configure --with-pcre-jit \

--with-steam \

--with-stream\_ssl\_module \

--with-stream\_ssl\_preread\_module \

--with-http\_v2\_module \

--with-http\_stub\_status\_module \

--with-http\_realip\_module \

--with-http\_addition\_module \

--with-http\_auth\_request\_module \

--with-http\_secure\_link\_module \

--with-http\_random\_index\_module \

--with-http\_gzip\_static\_module \

--with-http\_sub\_module \

--with-http\_dav\_module \

--with-http\_flv\_module \

--with-http\_mp4\_module \

--with-http\_gunzip\_module \

--with-http\_ssl\_module \

--with-threads \

--with-compat

根据实际需求去安装相应的模块，实在不确定的，可以都加上

**★第5章、4种安装方式的异同**

**1.程序文件路径**

|  |  |
| --- | --- |
| nginx源码安装 | /usr/local/nginx/sbin/nginx |
| nginx yum安装 | /usr/sbin/nginx |
| openresty源码安装 | /usr/local/openresty/nginx/sbin/nginx并做了个链接：  /usr/local/openresty/bin/openresty |
| openresty yum安装 | /usr/local/openresty/nginx/sbin/nginx并做了个链接：  /usr/bin/openresty |

**2.配置文件路径**

|  |  |
| --- | --- |
| nginx源码安装 | /usr/local/nginx/conf/nginx.conf |
| nginx yum安装 | /etc/nginx/nginx.conf |
| openresty源码安装 | /usr/local/openresty/nginx/conf/nginx.conf |
| openresty yum安装 | /usr/local/openresty/nginx/conf/nginx.conf |

**3.是否做成系统服务**（可用systemctl管理）

|  |  |
| --- | --- |
| nginx源码安装 | 否 |
| nginx yum安装 | 是，服务文件/usr/lib/systemd/system/nginx.service |
| openresty源码安装 | 否 |
| openresty yum安装 | 是，服务文件/usr/lib/systemd/system/openresty.service |

**4.相同点，主程序名都是nginx**

当直接使用./nginx命令与systemctl xx命令同时操作时，会有冲突，发生错误时，可先杀掉nginx进程，再统一使用一个命令去操作

# pkill nginx

直接调用./nginx程序和systemctl去操作nginx服务 这2种方式不要混用。因为假如刚刚使用systemctl start nginx命令启动服务，结果又用 ./nginx命令去做其他操作，这时systemctl并不知道，所以会出错。用systemctl status nginx看就会得到错误的状态。

**★第6章、基础web服务端配置**

yum安装的nginx主配置文件为：/etc/nginx/nginx.conf

**配置讲解：**

# cat /etc/nginx/nginx.conf

user nginx;

worker\_processes auto; #工作进程数，可手动指定与cpu核数相同

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

pid /var/run/nginx.pid;

events {

worker\_connections 1024; #最大连接数

}

http **{**

include /etc/nginx/mime.types; #文件扩展名与文件类型映射表

default\_type application/octet-stream; #默认文件类型，字节流

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

# log\_format定义一个名为main的日志格式

access\_log /var/log/nginx/access.log main; #访问日志文件及使用的日志格式

sendfile on; #允许sendfile方式传输文件，默认为off

#tcp\_nopush on;

keepalive\_timeout 65; #连接超时时间，默认为75s

#gzip on;

include /etc/nginx/conf.d/\*.conf; #加载此目录下的其他\*.conf配置文件

**}**

**★主配置最后一行include会加载/etc/nginx/conf.d/default.conf配置，内容如下：**

# cat /etc/nginx/conf.d/default.conf | grep -v '#' #里面是一个默认站点

server **{** #server这段是加载到 http {} 模块里的

listen 80; #监听80端口

server\_name localhost; #http的server\_name

location / { #针对 / 这个路径的配置

root /usr/share/nginx/html; #指定此url所在根目录

index index.html index.htm; #默认index页面文件

}

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

location = /50x.html { #针对/50x.html这个url的配置

root /usr/share/nginx/html;

}

**}**

**★第7章、tcp/udp反向代理配置**

**nginx做TCP反向代理示例：**

stream **{** #该模块与 http{}同级，不在http{}里

upstream tcp\_xx\_serv {

#tcp不能指定调度算法，默认weight，若后端有故障可主动移除

server 10.99.1.61:22 weight=1 max\_fails=2 fail\_timeout=8; #后端服务器

server 10.99.1.62:22 weight=1 max\_fails=2 fail\_timeout=8;

} #6秒内共失败2次则此后端不可达（每次失败超时是由

server { # proxy\_connect\_timeout决定的）

listen 2222; #监听本服务器上的2222/tcp

proxy\_pass tcp\_xx\_serv; #反向代理到名为tcp\_xx\_serv的upstream

proxy\_connect\_timeout 4; #默认60，最大75秒，本代理与后端服务的连接超时

}

**}**

**nginx做UDP反向代理示例：**

stream { #该模块与 http{}同级，不在http{}里

upstream udp\_xx\_serv {

#udp不能指定调度算法，默认weight，若后端有故障也不能主动移除

server 10.99.1.61:522; #后端服务器

server 10.99.1.62:522;

}

server {

listen 5222 udp; #监听本服务器上的5222/udp

proxy\_pass udp\_xx\_serv; #反向代理到名为udp\_xx\_serv的upstream

}

}

**★第8章、http(s)反向代理配置**

**https反向代理示例：**

upstream lb\_xxx { #位于http{}模块里

ip\_hash; #调度算法，其他有的 weight, fair, url\_hash（需加载相应模块支持）

server 10.99.1.61:480 weight=1 max\_fails=2 fail\_timeout=8; #后端服务器1

server 10.99.1.61:480 weight=1 max\_fails=2 fail\_timeout=8; #后端服务器2

}

server **{** #位于http{}模块里

listen 8822 ssl; #监听8822 ssl端口，防火墙要放通8822/tcp端口

server\_name vtest.xxx.com; #域名

ssl\_certificate /etc/test/vtest.xxx.com.crt; #证书文件

ssl\_certificate\_key /etc/test/vtest.xxx.com.key; #key文件

ssl\_session\_cache shared:SSL:10m;

ssl\_session\_timeout 10m; #客户端可重用ssl会话的时间，默认5分钟

keepalive\_timeout 75 75; #保持http连接的超时（第2个值75可选，表示返回给客户的http报头添加keepalive字段，这2个值可不一致，一般客户端浏览器不受第2个值的影响）决定服务器是否关闭http连接的是第一个值

client\_header\_timeout 60; #2次GET/POST请求时间间隔不能超过此时间，秒

client\_body\_timeout 60; #否则返回request time out (408)

send\_timeout 60; #客户端与本服务器在60秒内至少读一次

ssl\_ciphers ECDHE-RSA-AES128-GCM-SHA256:HIGH:!aNULL:!MD5;

ssl\_prefer\_server\_ciphers on;

location /xxx { #将此/xxx 路径做反向代理

proxy\_redirect off;

proxy\_pass https://lb\_xxx; #指定后端upstream，协议可为http,https

proxy\_http\_version 1.1;

proxy\_connect\_timeout 4; #本代理与后端服务的连接超时，默认60，最大75秒

proxy\_read\_timeout 60; #本服务与后端在指定时间内至少读一次，秒

proxy\_send\_timeout 60; #本服务与后端在指定时间内至少发一次

proxy\_ssl\_trusted\_certificate /etc/test/back-server-ca.crt; #验证后端证书

proxy\_ssl\_verify on;

proxy\_ssl\_verify\_depth 1;

proxy\_ssl\_session\_reuse on;

proxy\_ssl\_server\_name on;

proxy\_set\_header Host $host; #如果前后端域名相同则设置为原报文的host

#proxy\_set\_header Host *server.xxx.com*; #前后端域名不同则需改host字段

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_intercept\_errors on;

}

location / {

root /test; #服务器的/test目录作为 vtest.xxx.com站点的根目录

index index.html;

}

**}**