前言

对于项目里面**只是使用代理等常用功能**，**在线安装**即可，如需**制定化模块**，则推荐**编译安装**

> PS：本文不仅仅包含Nginx相关的知识点，还包含了逆天学习方法（对待新事物的处理）

官方网站： https://nginx.org/

> Github： https://github.com/nginx/nginx

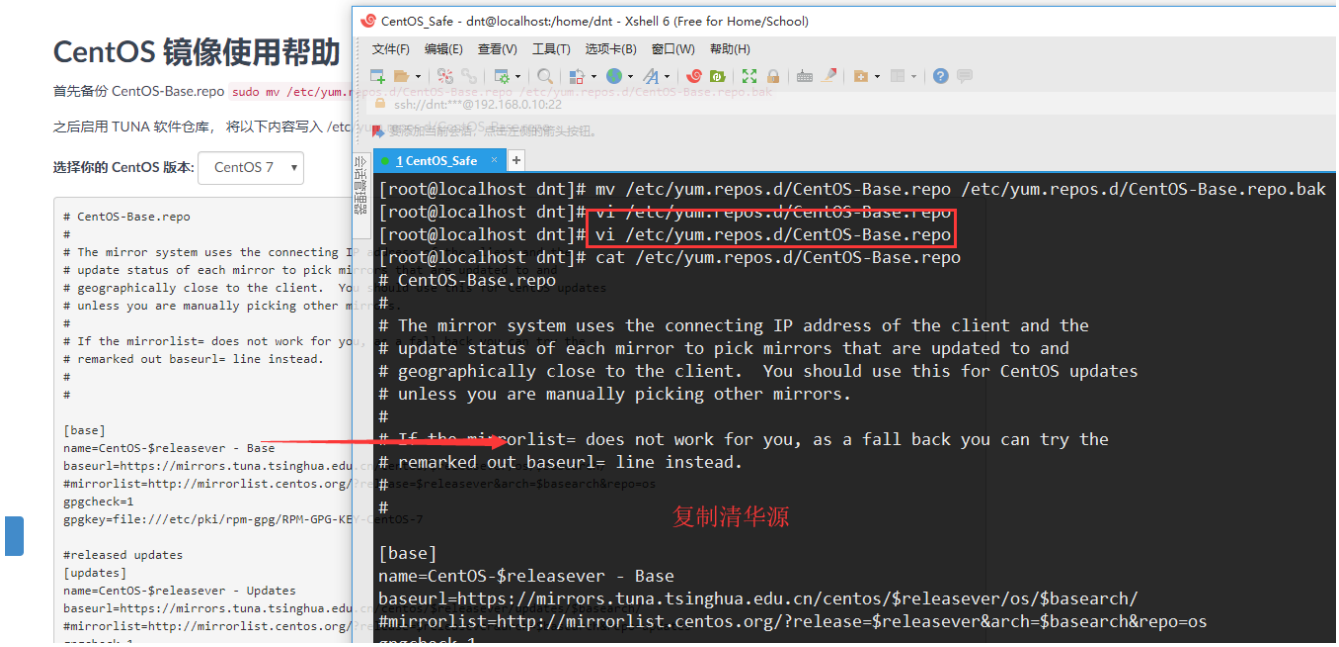
Nginx书籍：

> 1. Nginx Cookbook 中文版 https://huliuqing.gitbooks.io/complete-nginx-cookbook-zh/content/
> 2. Nginx官方中文文档 https://docshome.gitbooks.io/nginx-docs/content/
> 3. Nginx入门教程 https://xuexb.github.io/learn-nginx/
> 4. 淘宝Nginx文档 http://tengine.taobao.org/book/

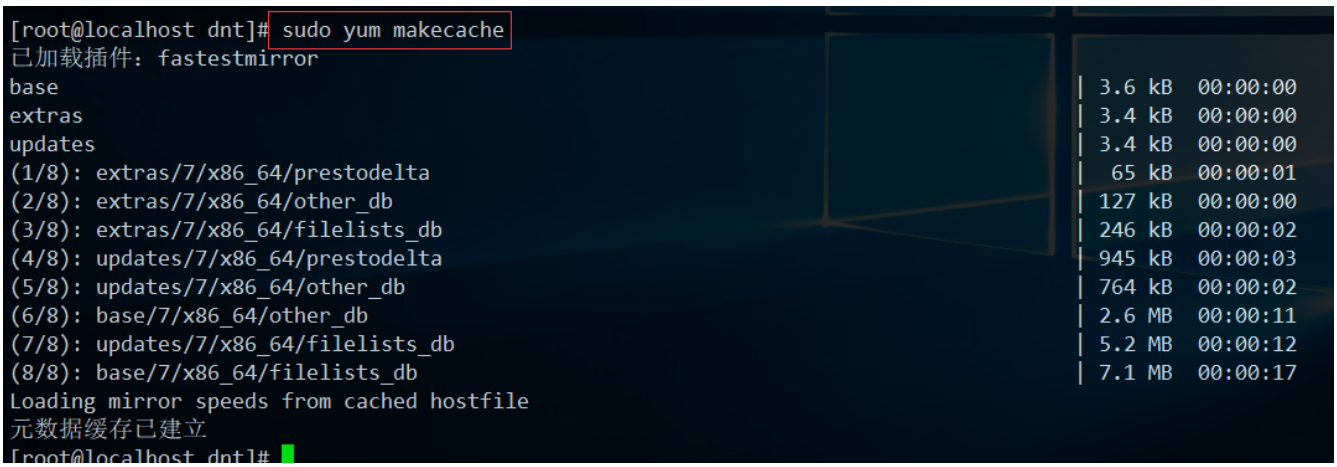
1.在线安装

1.1.修改yum源地址

清华源： https://mirrors.tuna.tsinghua.edu.cn/help/centos/



更新软件包缓存： `yum makecache`



1.2.在线安装Nginx

在线安装比较简单，参考官方文档即可： https://nginx.org/en/linux_packages.html

> PS：线上选 `stable` 的就行了，记得把 `$releasever` 改成你的版本号，eg： `7`

RHEL/CentOS

Install the prerequisites:

```
sudo yum install yum-utils
```

To set up the yum repository, create the file named `/etc/yum.repos.d/nginx.repo` with the following contents:

改成你的版本，eg： 7

```
[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
```

```
[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
```

By default, the repository for stable nginx packages is used. If you would like to use mainline nginx packages, run the following command:

```
sudo yum-config-manager --enable nginx-mainline
```

To install nginx, run the following command:

```
sudo yum install nginx
```

When prompted to accept the GPG key, verify that the fingerprint matches `573B FD6B 3D8F BC64 1079 A6AB ABF5 BD82 7BD9 BF62`, and if so, accept it.

安装图示：

```
[root@localhost dnt]# vi /etc/yum.repos.d/nginx.repo
[root@localhost dnt]# cat /etc/yum.repos.d/nginx.repo
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/centos/7/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
[root@localhost dnt]# yum install nginx -y
已加载插件：fastestmirror
base                                                     | 3.6 kB  00:00:00
extras                                                   | 3.4 kB  00:00:00
Loading mirror speeds from cached hostfile
 * base: mirrors.nju.edu.cn
 * extras: mirrors.163.com
 * updates: mirrors.163.com
nginx-stable/x86_64/primary_db                           |  46 kB  00:00:02
正在解决依赖关系
--> 正在检查事务
---> 软件包 nginx.x86_64.1.1.16.0-1.el7.ngx 将被 安装
--> 解决依赖关系完成
```

```
# 创建nginx的yum
vi /etc/yum.repos.d/nginx.repo

# 内容如下:
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/centos/7/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key

# 在线安装
yum install nginx -y
```

1.3.端口放行

放行80端口： `firewall-cmd --zone=public --add-port=80/tcp --permanent`

> PS：规则生效： `firewall-cmd --reload`

```
[root@localhost dnt]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: active (running) since 六 2019-08-03 18:50:17 CST; 17h ago
 Main PID: 868 (firewalld)
   CGroup: /system.slice/firewalld.service
           └─868 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid

8月 03 18:50:14 localhost.localdomain systemd[1]: Starting firewalld - dynamic firewall daemon...
8月 03 18:50:17 localhost.localdomain systemd[1]: Started firewalld - dynamic firewall daemon.
[root@localhost dnt]# firewall-cmd --zone=public --add-port=80/tcp --permanent
success
[root@localhost dnt]# firewall-cmd --reload
success
[root@localhost dnt]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:15:5d:00:07:01 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.10/24 brd 192.168.0.255 scope global eth0
```

防火墙状态

开放80短口

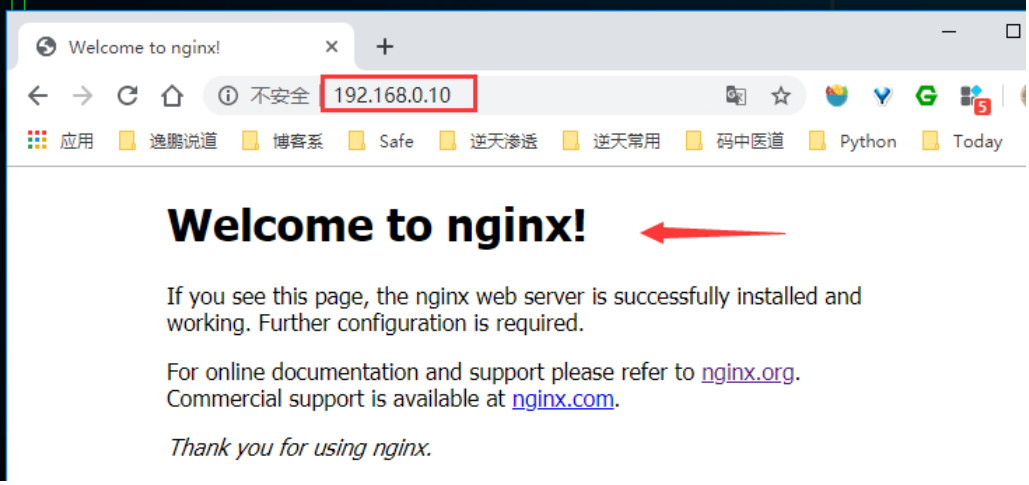
重新加载防火墙规则

1.4.验证安装

```
[root@localhost dnt]# systemctl enable nginx
Created symlink from /etc/systemd/system/multi-user.target.wants/nginx.service to /usr/lib/syst
nx.service.
[root@localhost dnt]# systemctl start nginx
[root@localhost dnt]# ps -aux |grep nginx
root      24599  0.0  0.0  46392   964 ?        Ss   12:15   0:00 nginx: master process /usr/sb
tc/nginx/nginx.conf
nginx     24600  0.0  0.1  46800  1912 ?        S    12:15   0:00 nginx: worker process
root      24602  0.0  0.0 112664   964 pts/4    S+   12:15   0:00 grep --color=auto nginx
[root@localhost dnt]#
```

设置开机启动

启动nginx



2.知识拓展

2.1.编译参数

离线安装可以参考在线安装的配置： `nginx -V` ：**编译参数** （ `nginx -v` ： 查看版本）

```
----------------------------------------------------------------------
  验证中      : 1:nginx-1.16.0-1.el7.ngx.x86_64                          1/1

已安装:
  nginx.x86_64 1:1.16.0-1.el7.ngx

完毕！
[root@localhost dnt]# nginx -v
nginx version: nginx/1.16.0
[root@localhost dnt]# nginx -V
nginx version: nginx/1.16.0
built by gcc 4.8.5 20150623 (Red Hat 4.8.5-36) (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/ac
cess.log --pid-path=/var/run/nginx.pid --lock-path=/var/run/nginx.lock --http-client-body-temp-path=/var/cach
e/nginx/client_temp --http-proxy-temp-path=/var/cache/nginx/proxy_temp --http-fastcgi-temp-path=/var/cache/ng
inx/fastcgi_temp --http-uwsgi-temp-path=/var/cache/nginx/uwsgi_temp --http-scgi-temp-path=/var/cache/nginx/sc
gi_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 --w
ith-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_rea
lip_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@localhost dnt]#
```

▶ 编译参数详解（点我展开）

```
# 1.编译选项
## Nginx的安装主目录
--prefix=/etc/nginx \
## Nginx的执行文件路径
--sbin-path=/usr/sbin/nginx \
## Nginx的模块目录
--modules-path=/usr/lib64/nginx/modules \
## Nginx的配置文件路径
--conf-path=/etc/nginx/nginx.conf \
## Nginx的错误日志路径
--error-log-path=/var/log/nginx/error.log \
## Nginx的访问日志
--http-log-path=/var/log/nginx/access.log \
## Nginx的pid文件路径
--pid-path=/var/run/nginx.pid \
## Nginx的lock路径
--lock-path=/var/run/nginx.lock \

# 2.编译选项（执行对应模块时Nginx缓存文件的存放地址）
--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 \

# 3.设置Nginx权限组（虽然root权限安装，但可以指定nginx的运行权限）
--user=nginx \
```

```
--group=nginx \

# 4.优化
## 启用gzip压缩模块（常用）
--with-http_gzip_static_module \
--with-http_gunzip_module \
# 文件使用aio异步操作
--with-file-aio \

## C系列优化
--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' \
## 设置附加的参数，链接系统库
--with-ld-opt='-Wl,-z,relro -Wl,-z,now -pie' \
# HTTP内容替换
--with-http_sub_module \

# 其他优化选项 or 模块
--with-compat \
--with-threads \
--with-http_addition_module \
--with-http_auth_request_module \
--with-http_dav_module \
--with-http_flv_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_v2_module \
--with-mail \
--with-mail_ssl_module \
--with-stream \
--with-stream_realip_module \
--with-stream_ssl_module \
--with-stream_ssl_preread_module \
```

2.2.安装目录

在线安装的包都可以通过：`rpm -ql xxx` 查看安装到哪些目录

▶ 安装目录详解（点我展开）

```
[root@localhost dnt]# rpm -ql nginx

# Nginx使用用logrotate服务对日志进行切割的配置文件（eg：按天切割）
/etc/logrotate.d/nginx

# Nginx的核心目录
```

```
/etc/nginx
# 主要配置文件，Nginx启动的时候会读取
/etc/nginx/nginx.conf
/etc/nginx/conf.d
# nginx.conf没变更久读default.conf（默认Server加载的文件）
/etc/nginx/conf.d/default.conf

# Nginx对Python的wsgi配置
/etc/nginx/uwsgi_params
# fastcgi配置
/etc/nginx/fastcgi_params
# scgi配置
/etc/nginx/scgi_params

# Nginx缓存目录
/var/cache/nginx

# Nginx日志目录
/var/log/nginx

# Nginx默认网站存放的路径
/usr/share/nginx/html
/usr/share/nginx/html/50x.html
/usr/share/nginx/html/index.html

# 设置http的Content-Type与扩展名对应关系的配置文件
/etc/nginx/mime.types

# Nginx模块所在目录
/usr/lib64/nginx/modules
/etc/nginx/modules

# 二进制执行文件
/usr/sbin/nginx
/usr/sbin/nginx-debug

# 编码转换的映射文件
/etc/nginx/koi-utf
/etc/nginx/koi-win
/etc/nginx/win-utf

# 配置CentOS守护进程对Nginx的管理方式
/usr/lib/systemd/system/nginx-debug.service
/usr/lib/systemd/system/nginx.service
/etc/sysconfig/nginx
/etc/sysconfig/nginx-debug

# Nginx的文档
/usr/share/doc/nginx-1.16.0
/usr/share/doc/nginx-1.16.0/COPYRIGHT
/usr/share/man/man8/nginx.8.gz

# Nginx检测更新命令
```

```
/usr/libexec/initscripts/legacy-actions/nginx
/usr/libexec/initscripts/legacy-actions/nginx/check-reload
/usr/libexec/initscripts/legacy-actions/nginx/upgrade
```

2.3.默认配置

配置语法检查： `nginx -t -c /etc/nginx/nginx.conf`

> PS：不重启的方式加载配置： `Nginx -s reload -c /etc/nginx/nginx.conf`

全局以及服务级别的配置：

参数	说明
`user`	使用用户来运行nginx
`worker_processes`	工作进程数
`error_log`	nginx的错误日记
pid	nginx启动时的pid

events相关配置：

参数	说明
`worker_connections`	每个进程的最大连接数
`use`	工作进程数

常用中间件配置：

```
http {
    ......
    server {
        listen          80;             # 端口号
        server_name     localhost;      # 域名
        # 路径访问控制（默认访问路径，eg: / ==> 根目录）
        location / {
            root /usr/share/nginx/html; # 网站根目录
            index index.html index.htm index.py; # 首页配置
        }

        error_page 500 502 503 504 /50x.html; # 错误页面（可以自定义添404页面，error_page 404
/404.html;...）
        # 访问xxx/50x.html的时候去指定目录找
        location = /50x.html {
            root /usr/share/nginx/html; # 错误页面所在路径
        }
    }
    # 一个server配置一个虚拟 or 独立的站点（通过listen和server_name来区别多个server）
```

```
    server {
        ......
    }
}
```

2.4.systemctl配置

nginx:（等会编译安装的时候可以参考）

```
[root@localhost dnt]# cat /usr/lib/systemd/system/nginx.service
[Unit]
Description=nginx - high performance web server
Documentation=http://nginx.org/en/docs/
After=network-online.target remote-fs.target nss-lookup.target
Wants=network-online.target

[Service]
Type=forking
PIDFile=/var/run/nginx.pid
ExecStart=/usr/sbin/nginx -c /etc/nginx/nginx.conf
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID

[Install]
WantedBy=multi-user.target
```

nginx-debug：

```
[root@localhost dnt]# cat /usr/lib/systemd/system/nginx-debug.service
[Unit]
Description=nginx - high performance web server
Documentation=http://nginx.org/en/docs/
After=network-online.target remote-fs.target nss-lookup.target
Wants=network-online.target

[Service]
Type=forking
PIDFile=/var/run/nginx.pid
ExecStart=/usr/sbin/nginx-debug -c /etc/nginx/nginx.conf
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID

[Install]
WantedBy=multi-user.target
```

3.编译安装

3.1.安装编译环境

一步到位：`yum install gcc-c++ pcre pcre-devel zlib zlib-devel openssl openssl-devel -y`

```
[root@localhost dnt]# yum install gcc-c++ pcre pcre-devel zlib zlib-devel openssl openssl-devel -y
已加载插件：fastestmirror
Loading mirror speeds from cached hostfile
正在解决依赖关系
--> 正在检查事务
---> 软件包 gcc-c++.x86_64.0.4.8.5-4.el7 将被 升级
---> 软件包 gcc-c++.x86_64.0.4.8.5-36.el7_6.2 将被 更新
--> 正在处理依赖关系 libstdc++-devel = 4.8.5-36.el7_6.2，它被软件包 gcc-c++-4.8.5-36.el7_6.2.x86_64 需
--> 正在处理依赖关系 libstdc++ = 4.8.5-36.el7_6.2，它被软件包 gcc-c++-4.8.5-36.el7_6.2.x86_64 需要
--> 正在处理依赖关系 gcc = 4.8.5-36.el7_6.2，它被软件包 gcc-c++-4.8.5-36.el7_6.2.x86_64 需要
---> 软件包 openssl.x86_64.1.1.0.1e-42.el7.9 将被 升级
---> 软件包 openssl.x86_64.1.1.0.2k-16.el7_6.1 将被 更新
--> 正在处理依赖关系 openssl-libs(x86-64) = 1:1.0.2k-16.el7_6.1，它被软件包 1:openssl-1.0.2k-16.el7_6.1
4 需要
--> 正在处理依赖关系 libcrypto.so.10(OPENSSL_1.0.2)(64bit)，它被软件包 1:openssl-1.0.2k-16.el7_6.1.x86_
要
---> 软件包 openssl-devel.x86_64.1.1.0.2k-16.el7_6.1 将被 安装
--> 正在处理依赖关系 krb5-devel(x86-64)，它被软件包 1:openssl-devel-1.0.2k-16.el7_6.1.x86_64 需要
---> 软件包 pcre.x86_64.0.8.32-15.el7 将被 升级
---> 软件包 pcre.x86_64.0.8.32-17.el7 将被 更新
```

简单拆分解析一下：

1. Nginx使用 `C/C++` 编写的，安装一下依赖：`yum install gcc-c++ -y`
2. Nginx需要使用PCRE来进行正则解析：`yum install pcre pcre-devel -y`
3. 现在服务器和浏览器一般都是使用gzip：`yum install -y zlib zlib-devel -y`
4. 让Nginx支持https：`yum install openssl openssl-devel -y`

3.2.Nginx编译安装

3.2.1.下载解压

先编译安装一下，后面说lua模块的时候再重新编译下就行了

下载：`curl -o nginx.tar.gz http://nginx.org/download/nginx-1.16.0.tar.gz`

解压：`tar -zxvf nginx.tar.gz`

3.2.2.配置编译参数

参考前面说的在线版Nginx来设置编译参数的配置：

> PS：`nginx -V`

切换到nginx的解压目录：`cd nginx-1.16.0` 然后执行下面命令

> PS：root权限编译哦~

```
./configure --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'
```

3.2.3.进行编译安装

接着编译安装： `make && make install`

> PS：提速： `make -j 4 && make install`

```
Configuration summary
  + using threads
  + using system PCRE library
  + using system OpenSSL library
  + using system zlib library

  nginx path prefix: "/etc/nginx"
  nginx binary file: "/usr/sbin/nginx"
  nginx modules path: "/usr/lib64/nginx/modules"
  nginx configuration prefix: "/etc/nginx"
  nginx configuration file: "/etc/nginx/nginx.conf"
  nginx pid file: "/var/run/nginx.pid"
  nginx error log file: "/var/log/nginx/error.log"
  nginx http access log file: "/var/log/nginx/access.log"
  nginx http client request body temporary files: "/var/cache/nginx/client_temp"
  nginx http proxy temporary files: "/var/cache/nginx/proxy_temp"
  nginx http fastcgi temporary files: "/var/cache/nginx/fastcgi_temp"
  nginx http uwsgi temporary files: "/var/cache/nginx/uwsgi_temp"
  nginx http scgi temporary files: "/var/cache/nginx/scgi_temp"

[root@localhost nginx-1.16.0]# make && make install
make -f objs/Makefile
make[1]: 进入目录“/home/dnt/nginx-1.16.0”
cc -c -I/usr/local/LuaJIT/include/luajit-2.0  -pipe  -O -W -Wall -Wpointer-arith -W
r -g -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protector-stro
=4 -grecord-gcc-switches -m64 -mtune=generic -fPIC -DNDK_SET_VAR -I src/core -I src
s -I src/os/unix -I /etc/nginx/modules/ngx_devel_kit-0.3.1/objs -I objs/addon/ndk -
ginx-module-0.10.15/src/api -I objs \
        -o objs/src/core/nginx.o \
```

编译安装

3.2.4.配置systemctl

利用systemctl添加自定义系统服务

```
[root@localhost dnt]# vi /usr/lib/systemd/nginx.service
[root@localhost dnt]# cat /usr/lib/systemd/nginx.service
[Unit]
Description=nginx - high performance web server
Documentation=http://nginx.org/en/docs/
After=network-online.target remote-fs.target nss-lookup.target
Wants=network-online.target

[Service]
Type=forking
PIDFile=/var/run/nginx.pid
ExecStart=/usr/sbin/nginx -c /etc/nginx/nginx.conf
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID

[Install]
WantedBy=multi-user.target
[root@localhost dnt]# vi /usr/lib/systemd/nginx-debug.service
[root@localhost dnt]# cat /usr/lib/systemd/nginx-debug.service
[Unit]
Description=nginx - high performance web server
Documentation=http://nginx.org/en/docs/
After=network-online.target remote-fs.target nss-lookup.target
Wants=network-online.target

[Service]
Type=forking
PIDFile=/var/run/nginx.pid
ExecStart=/usr/sbin/nginx-debug -c /etc/nginx/nginx.conf
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID

[Install]
WantedBy=multi-user.target
```

```
# vi /usr/lib/systemd/system/nginx.service
[Unit]
Description=nginx - high performance web server
Documentation=http://nginx.org/en/docs/
After=network-online.target remote-fs.target nss-lookup.target
Wants=network-online.target
```

```
[Service]
Type=forking
PIDFile=/var/run/nginx.pid
ExecStart=/usr/sbin/nginx -c /etc/nginx/nginx.conf
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID

[Install]
WantedBy=multi-user.target
```

PS：如果不生效可以重载下systemctl： `systemctl daemon-reload`

3.2.5.端口放行

放行80端口： `firewall-cmd --zone=public --add-port=80/tcp --permanent`

> PS：规则生效： `firewall-cmd --reload`

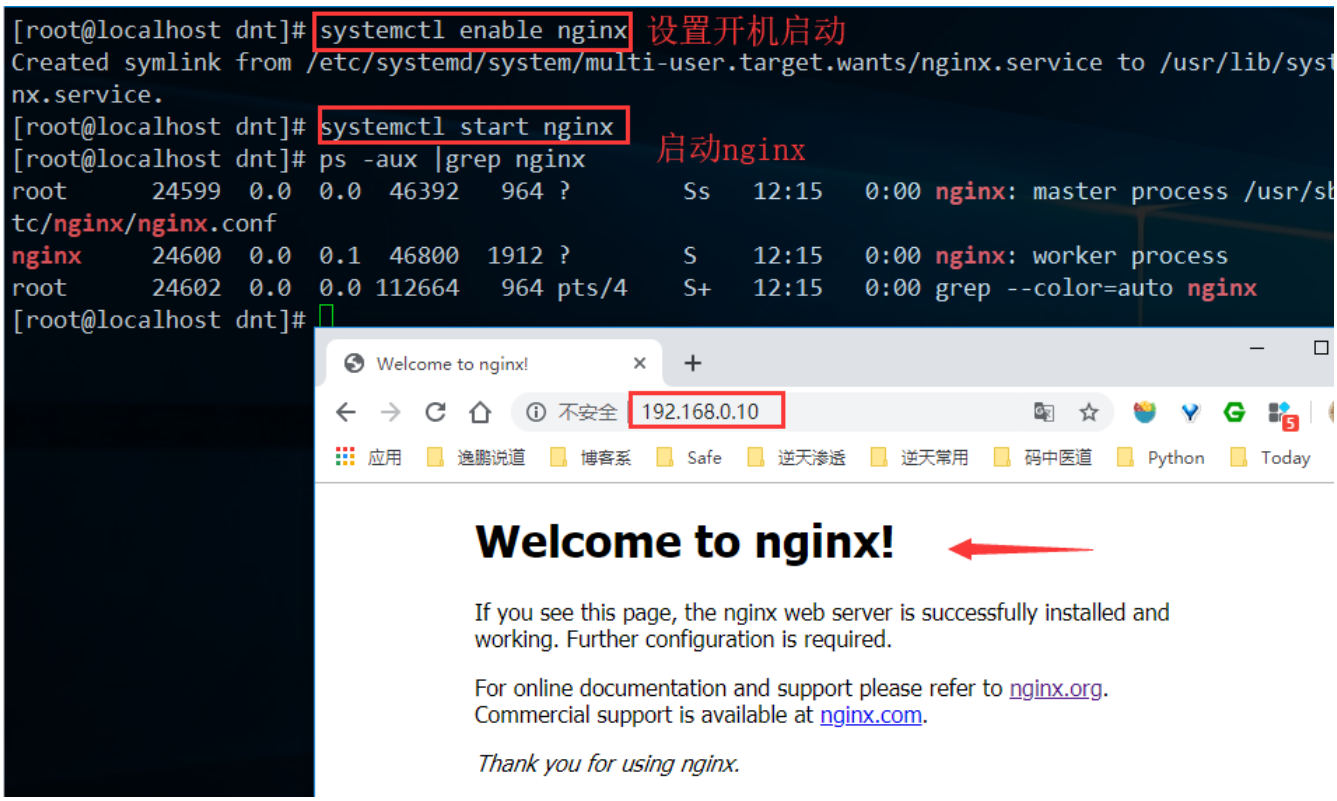
```
[root@localhost dnt]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: active (running) since 六 2019-08-03 18:50:17 CST; 17h ago
 Main PID: 868 (firewalld)
   CGroup: /system.slice/firewalld.service
           └─868 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid

8月 03 18:50:14 localhost.localdomain systemd[1]: Starting firewalld - dynamic firewall daemon...
8月 03 18:50:17 localhost.localdomain systemd[1]: Started firewalld - dynamic firewall daemon.
[root@localhost dnt]# firewall-cmd --zone=public --add-port=80/tcp --permanent
success
[root@localhost dnt]# firewall-cmd --reload
success
[root@localhost dnt]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:15:5d:00:07:01 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.10/24 brd 192.168.0.255 scope global eth0
```

防火墙状态

开放80短口

重新加载防火墙规则

3.2.6.验证



运行的时候如果出现 `nginx: [emerg] getpwnam("nginx") failed` 的错误可以参考我写这篇文章：https://www.cnblogs.com/dotnetcrazy/p/11304783.html

> PS：核心：`useradd -s /sbin/nologin -M nginx`

3.3.编译安装Lua模块

大体思路

默认是不支持Lua的，所以需要自己编译安装下

> PS：记得安装下Lua库：`yum install lua lua-devel -y`

主要就3步走：

1. 安装**Lua即时编译器**：`LuaJIT`
   - 目前最新：http://luajit.org/download/LuaJIT-2.0.5.tar.gz
2. 安装Nginx模块：`ngx_devel_kit` and `lua-nginx-module`
   1. ngx_devel_kit：https://github.com/simplresty/ngx_devel_kit/archive/v0.3.1.tar.gz
   2. lua-nginx-module：https://github.com/openresty/lua-nginx-module/archive/v0.10.15.tar.gz
3. 重新编译Nginx：复制在线安装的**编译参数**（`nginx -V`）然后**添加两个参数**
   1. `--add-module=../ngx_devel_kit-0.3.1`
   2. `--add-module=../lua-nginx-module-0.10.15`

3.3.1.编译安装luajit并导入环境变量

解压缩

```
[dnt@localhost ~]$ ls
LuaJIT-2.0.5.tar.gz              nginx-1.16.0         ngx_devel_kit-0.3.1.tar.gz
lua-nginx-module-0.10.15.tar.gz  nginx-1.16.0.tar.gz
[dnt@localhost ~]$ tar -zxf lua-nginx-module-0.10.15.tar.gz
[dnt@localhost ~]$ tar -zxf ngx_devel_kit-0.3.1.tar.gz
[dnt@localhost ~]$ tar -zxf LuaJIT-2.0.5.tar.gz
[dnt@localhost ~]$ ls
LuaJIT-2.0.5         lua-nginx-module-0.10.15         nginx-1.16.0         ngx_devel_kit-0.3.1
LuaJIT-2.0.5.tar.gz  lua-nginx-module-0.10.15.tar.gz  nginx-1.16.0.tar.gz  ngx_devel_kit-0.3.1.tar.gz
[dnt@localhost ~]$
```

```
# 编译安装
make install PREFIX=/usr/local/LuaJIT
# 导入环境变量
export LUAJIT_LIB=/usr/local/LuaJIT/lib
export LUAJIT_INC=/usr/local/LuaJIT/include/luajit-2.0
```

```
[root@localhost LuaJIT-2.0.5]# make install PREFIX=/usr/local/LuaJIT
==== Installing LuaJIT 2.0.5 to /usr/local/LuaJIT ====
mkdir -p /usr/local/LuaJIT/bin /usr/local/LuaJIT/lib /usr/local/LuaJIT/include/luajit-2.0
hare/man/man1 /usr/local/LuaJIT/lib/pkgconfig /usr/local/LuaJIT/share/luajit-2.0.5/jit /us
e/lua/5.1 /usr/local/LuaJIT/lib/lua/5.1
cd src && install -m 0755 luajit /usr/local/LuaJIT/bin/luajit-2.0.5
cd src && test -f libluajit.a && install -m 0644 libluajit.a /usr/local/LuaJIT/lib/libluaj
rm -f /usr/local/LuaJIT/bin/luajit /usr/local/LuaJIT/lib/libluajit-5.1.so.2.0.5 /usr/local
it-5.1.so /usr/local/LuaJIT/lib/libluajit-5.1.so.2
cd src && test -f libluajit.so && \
  install -m 0755 libluajit.so /usr/local/LuaJIT/lib/libluajit-5.1.so.2.0.5 && \
  ldconfig -n /usr/local/LuaJIT/lib && \
  ln -sf libluajit-5.1.so.2.0.5 /usr/local/LuaJIT/lib/libluajit-5.1.so && \
  ln -sf libluajit-5.1.so.2.0.5 /usr/local/LuaJIT/lib/libluajit-5.1.so.2 || :
cd etc && install -m 0644 luajit.1 /usr/local/LuaJIT/share/man/man1
cd etc && sed -e "s|^prefix=.*|prefix=/usr/local/LuaJIT|" -e "s|^multilib=.*|multilib=lib|
t.pc.tmp && \
  install -m 0644 luajit.pc.tmp /usr/local/LuaJIT/lib/pkgconfig/luajit.pc && \
  rm -f luajit.pc.tmp
cd src && install -m 0644 lua.h lualib.h lauxlib.h luaconf.h lua.hpp luajit.h /usr/local/L
t-2.0
cd src/jit && install -m 0644 bc.lua v.lua dump.lua dis_x86.lua dis_x64.lua dis_arm.lua di
lua dis_mipsel.lua bcsave.lua vmdef.lua /usr/local/LuaJIT/share/luajit-2.0.5/jit
ln -sf luajit-2.0.5 /usr/local/LuaJIT/bin/luajit
==== Successfully installed LuaJIT 2.0.5 to /usr/local/LuaJIT ====
[root@localhost LuaJIT-2.0.5]# export LUAJIT_LIB=/usr/local/LuaJIT/lib
[root@localhost LuaJIT-2.0.5]# export LUAJIT_INC=/usr/local/LuaJIT/include/luajit-2.0
[root@localhost LuaJIT-2.0.5]#
```

1. 编译安装

2. 导入环境变量

3.3.2.配置nginx的编译参数

```
[root@localhost LuaJIT-2.0.5]# export LUAJIT_LIB=/usr/local/LuaJIT/lib
[root@localhost LuaJIT-2.0.5]# export LUAJIT_INC=/usr/local/LuaJIT/include/luajit-2.0
[root@localhost LuaJIT-2.0.5]# cd ../nginx-1.16.0
[root@localhost nginx-1.16.0]# ls
auto  CHANGES  CHANGES.ru  conf  configure  contrib  html  LICENSE  Makefile  man  objs  README  src
[root@localhost nginx-1.16.0]# ./configure --prefix=/etc/nginx --sbin-path=/usr/sbin/nginx --modules
r/lib64/nginx/modules --conf-path=/etc/nginx/nginx.conf --error-log-path=/var/log/nginx/error.log --
path=/var/log/nginx/access.log --pid-path=/var/run/nginx.pid --lock-path=/var/run/nginx.lock --http-
dy-temp-path=/var/cache/nginx/client_temp --http-proxy-temp-path=/var/cache/nginx/proxy_temp --http-
emp-path=/var/cache/nginx/fastcgi_temp --http-uwsgi-temp-path=/var/cache/nginx/uwsgi_temp --http-scg
th=/var/cache/nginx/scgi_temp --user=nginx --group=nginx --with-compat --with-file-aio --with-thread
http_addition_module --with-http_auth_request_module --with-http_dav_module --with-http_flv_module -
p_gunzip_module --with-http_gzip_static_module --with-http_mp4_module --with-http_random_index_modul
http_realip_module --with-http_secure_link_module --with-http_slice_module --with-http_ssl_module --
_stub_status_module --with-http_sub_module --with-http_v2_module --with-mail --with-mail_ssl_module
ream --with-stream_realip_module --with-stream_ssl_module --with-stream_ssl_preread_module --with-cc
 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protector-strong --param=ssp-buffer-siz
ord-gcc-switches -m64 -mtune=generic -fPIC' --with-ld-opt='-Wl,-z,relro -Wl,-z,now -pie' --add-modul
devel_kit-0.3.1 --add-module=../lua-nginx-module-0.10.15
checking for OS
 + Linux 3.10.0-327.el7.x86_64 x86_64
checking for C compiler ... found
 + using GNU C compiler
 + gcc version: 4.8.5 20150623 (Red Hat 4.8.5-36) (GCC)
```

就多了两个模块的参数，其他和在线的一样

完整参数附录：

```
./configure --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' --add-
module=../ngx_devel_kit-0.3.1 --add-module=../lua-nginx-module-0.10.15
```

3.3.3.重新编译安装nginx

编译安装： `make && make install`

```
Configuration summary
  + using threads
  + using system PCRE library
  + using system OpenSSL library
  + using system zlib library

  nginx path prefix: "/etc/nginx"
  nginx binary file: "/usr/sbin/nginx"
  nginx modules path: "/usr/lib64/nginx/modules"
  nginx configuration prefix: "/etc/nginx"
  nginx configuration file: "/etc/nginx/nginx.conf"
  nginx pid file: "/var/run/nginx.pid"
  nginx error log file: "/var/log/nginx/error.log"
  nginx http access log file: "/var/log/nginx/access.log"
  nginx http client request body temporary files: "/var/cache/nginx/client_temp"
  nginx http proxy temporary files: "/var/cache/nginx/proxy_temp"
  nginx http fastcgi temporary files: "/var/cache/nginx/fastcgi_temp"
  nginx http uwsgi temporary files: "/var/cache/nginx/uwsgi_temp"
  nginx http scgi temporary files: "/var/cache/nginx/scgi_temp"

[root@localhost nginx-1.16.0]# make && make install
make -f objs/Makefile
make[1]: 进入目录“/home/dnt/nginx-1.16.0”
cc -c -I/usr/local/LuaJIT/include/luajit-2.0  -pipe  -O -W -Wall -Wpointer-arith -W
r -g -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protector-stro
=4 -grecord-gcc-switches -m64 -mtune=generic -fPIC -DNDK_SET_VAR -I src/core -I src
s -I src/os/unix -I /etc/nginx/modules/ngx_devel_kit-0.3.1/objs -I objs/addon/ndk -
ginx-module-0.10.15/src/api -I objs \
        -o objs/src/core/nginx.o \
```

编译安装

3.3.4.共享lua动态库

加载lua库到ld.so.conf文件

```
echo "/usr/local/LuaJIT/lib" >> /etc/ld.so.conf
```

```
[root@localhost dnt]# cat /etc/ld.so.conf
include ld.so.conf.d/*.conf
[root@localhost dnt]# echo "/usr/local/LuaJIT/lib" >> /etc/ld.so.conf
[root@localhost dnt]# cat /etc/ld.so.conf
include ld.so.conf.d/*.conf
/usr/local/LuaJIT/lib
```

执行 `ldconfig` 让动态函式库加载到缓存中

```
[root@localhost nginx-1.16.0]# ldconfig
[root@localhost nginx-1.16.0]# systemctl start nginx
[root@localhost nginx-1.16.0]# ps -aux |grep nginx
root      13969  0.0  0.0  54636  1112 ?        Ss   20:09   0:00 nginx: master process /usr/sbin/nginx
tc/nginx/nginx.conf
nginx     13970  0.0  0.1  55032  2080 ?        S    20:09   0:00 nginx: worker process
root      13973  0.0  0.0 112664   964 pts/0    S+   20:10   0:00 grep --color=auto nginx
[root@localhost nginx-1.16.0]#
```

3.3.5.验证Lua模块

验证下Lua是否已经可用：

在nginx.config的server节点下添加：

> PS：`vi /etc/nginx/nginx.conf`

```
# 测试Nginx的Lua
location /hello {
    default_type 'text/plain';
    content_by_lua 'ngx.say("欢迎访问逸鹏说道公众号~")';
}
```

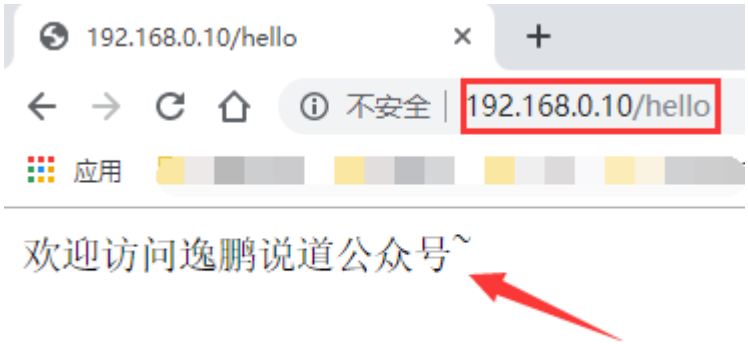
```
server {
    listen       80;
    server_name  localhost;
    charset utf-8; # 默认编码为utf-8

    location / {
        root   html;
        index  index.html index.htm;
    }
    ...
    # 测试Nginx的Lua（添加这一段）
    location /hello {
        default_type 'text/plain';
        content_by_lua 'ngx.say("欢迎访问逸鹏说道公众号~")';
    }
    ...
}
```

检查配置：`nginx -t -c /etc/nginx/nginx.conf`

> PS：配置生效：`nginx -s reload -c /etc/nginx/nginx.conf`

```
[root@localhost dnt]# vi /etc/nginx/nginx.conf
[root@localhost dnt]# nginx -t -c /etc/nginx/nginx.conf
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@localhost dnt]# nginx -s reload -c /etc/nginx/nginx.conf
```

看看效果：



扩展：你可以试试获取ip哦~

```
# 获取客户端ip
location /myip {
    default_type 'text/plain';
    content_by_lua '
        clientIP = ngx.req.get_headers()["x_forwarded_for"]
        ngx.say("IP:",clientIP)
    ';
}
```

4.Nginx+Lua搭建WAF防火墙

市面上比较常用一块开源项目：`ngx_lua_waf`

> https://github.com/loveshell/ngx_lua_waf

1. 拦截Cookie类型工具
2. 拦截异常post请求
3. 拦截CC洪水攻击
4. 拦截URL
5. 拦截arg（提交的参数）

4.1.环境

clone代码并移动到nginx的waf目录下

```
[root@localhost dnt]# ls
[root@localhost dnt]# git clone https://github.com/loveshell/ngx_lua_waf.git
正克隆到 'ngx_lua_waf ...
remote: Enumerating objects: 538, done.
remote: Total 538 (delta 0), reused 0 (delta 0), pack-reused 538
接收对象中: 100% (538/538), 82.48 KiB | 84.00 KiB/s, done.
处理 delta 中: 100% (298/298), done.
[root@localhost dnt]# ls
ngx_lua_waf
[root@localhost dnt]# cd ngx_lua_waf/
[root@localhost ngx_lua_waf]# ls
config.lua  init.lua  install.sh  README.md  wafconf  waf.lua
[root@localhost ngx_lua_waf]# ls wafconf/
args  cookie  post  url  user-agent  whiteurl
[root@localhost ngx_lua_waf]# mkdir /etc/nginx/waf
[root@localhost ngx_lua_waf]# mv * /etc/nginx/waf/
[root@localhost ngx_lua_waf]# ls
[root@localhost ngx_lua_waf]# ls /etc/nginx/waf/
config.lua  init.lua  install.sh  README.md  wafconf  waf.lua
```

克隆一份代码

各种防护规则，可以自己添加

移动到nginx的waf目录下

简单说下里面的规则分别有啥用：

1. args里面的规则get参数进行过滤的
2. url是只在get请求url过滤的规则
3. post是只在post请求过滤的规则
4. whitelist是白名单，里面的url匹配到不做过滤
5. user-agent是对user-agent的过滤规则

4.2.配置

修改必要配置

```
[root@localhost ngx_lua_waf]# cd /etc/nginx/waf/
[root@localhost waf]# vi config.lua
[root@localhost waf]# cat config.lua
RulePath = "/etc/nginx/waf/wafconf/"
attacklog = "on"
logdir = "/var/log/nginx/hack/"
UrlDeny="on"
Redirect="on"
CookieMatch="on"
postMatch="on"
whiteModule="on"
black_fileExt={"php","jsp","asp","aspx","py"}
ipWhitelist={"127.0.0.1"}
ipBlocklist={"1.0.0.1"}
CCDeny="on"
CCrate="100/60"
html=[[
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<meta http-equiv="Content-Type" content="text/html;
<title>网站防火墙</title>
<style>
p {
        line-height:20px;
}
ul{ list-style-type:none;}
li{ list-style-type:none;}
</style>
</head>
```

1. 设置规则所在目录

2. 设置记录日志的存放路径

3. 不允许上传的文件（自定义）

4. cc防护

详细说明我引用一下我的上篇文章：

参数简单说明下：**红色字体部分需要修改**

```
bryan@bryan-pc:~$ sudo vi /etc/nginx/waf/config.lua
bryan@bryan-pc:~$ cat /etc/nginx/waf/config.lua
RulePath = "/etc/nginx/waf/wafconf/"
attacklog = "on"
logdir = "/var/log/nginx/hack"
UrlDeny="on"
Redirect="on"
CookieMatch="on"
postMatch="on"
whiteModule="on"
black_fileExt={"php","jsp","py","asp","aspx"}
ipWhitelist={"127.0.0.1"}
ipBlocklist={"1.0.0.1"}
CCDeny="off"
CCrate="100/60"
html=[[
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>网站防火墙</title>
<style>
p {
        line-height:20px;
}
```

1. 设置一下规则在的目录
记录日志
2. 设置记录日志存放地址（不存在可以创建一下）
匹配url
匹配cookie
匹配post
开启白名单
不允许上传的后缀（自己添加）
白名单列表
黑名单列表
on就打开cc防护
访问频率：60s内访问100次
拦截后显示的html（可自定义）

`nginx.config` 的 `http` 下添加如下内容：

```
http {
    include       mime.types;
    default_type  application/octet-stream;
    server_tokens off;
    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';
    # access_log  logs/access.log  main;

    # Lua配置
    lua_package_path "/etc/nginx/waf/?.lua";
    lua_shared_dict limit 10m;
    init_by_lua_file /etc/nginx/waf/init.lua;
    access_by_lua_file /etc/nginx/waf/waf.lua;

    sendfile        on;
    #tcp_nopush     on;
```

```
lua_package_path "/etc/nginx/waf/?.lua";
lua_shared_dict limit 10m;
init_by_lua_file /etc/nginx/waf/init.lua;
access_by_lua_file /etc/nginx/waf/waf.lua;
```

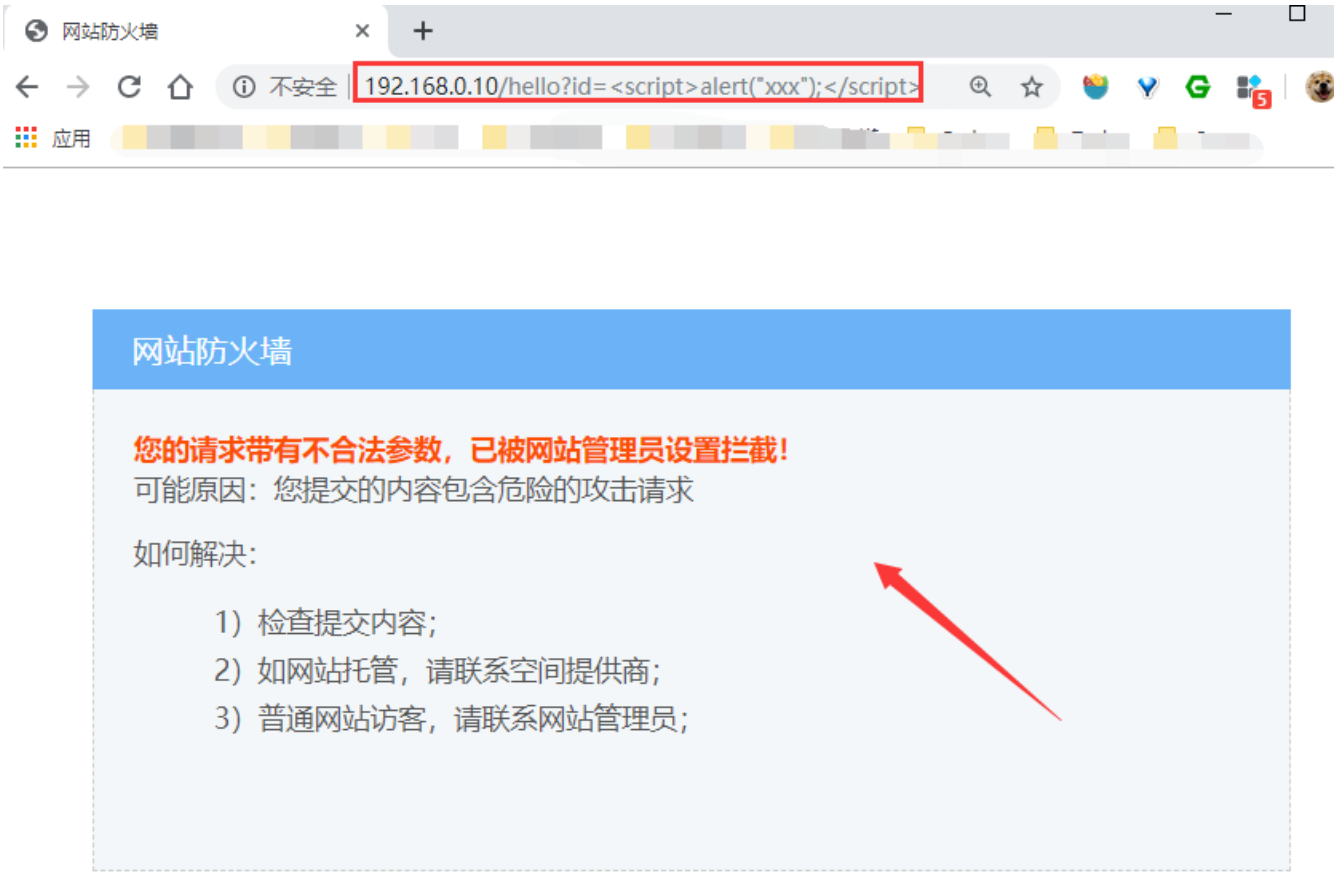
4.3.生效

配置语法检查：`nginx -t -c /etc/nginx/nginx.conf`

> PS：不重启的方式加载配置：`Nginx -s reload -c /etc/nginx/nginx.conf`

```
[root@localhost ~]# nginx -t -c /etc/nginx/nginx.conf
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@localhost ~]# nginx -s reload -c /etc/nginx/nginx.conf
[root@localhost ~]#
```

4.4.简单验证

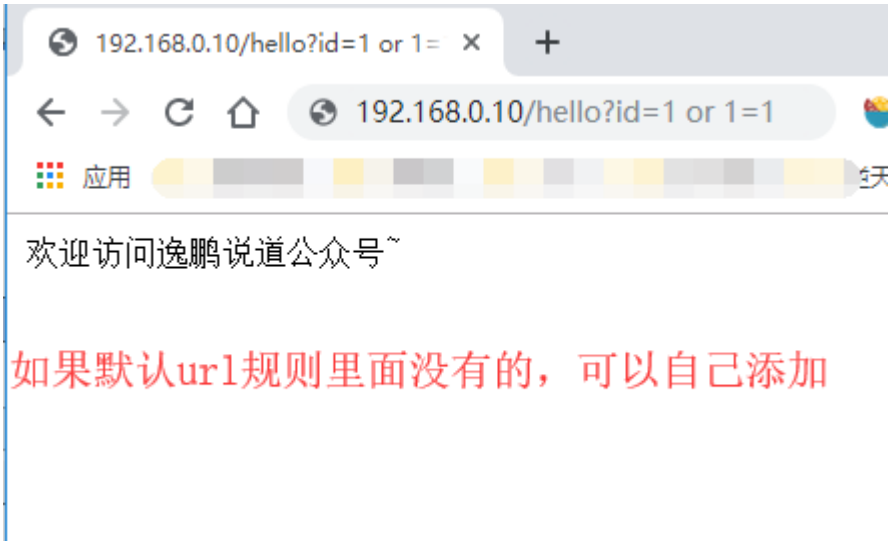


PS：其实绕过很简单，看看他默认规则即可，这款WAF的强大之处在于轻量级，而且规则可以自定化

> 过滤规则在wafconf下，可根据需求自行调整，每条规则需换行,或者用|分割

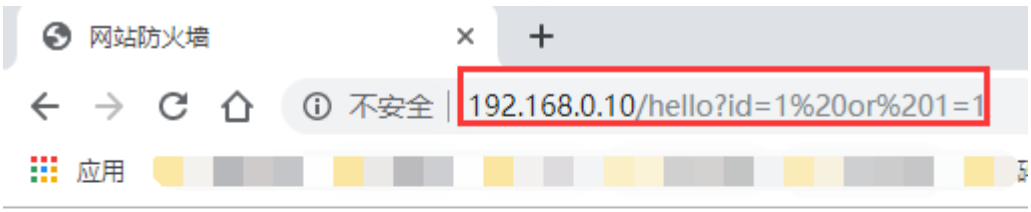
举个例子：`http://192.168.0.10/hello?id=1 or 1=1`

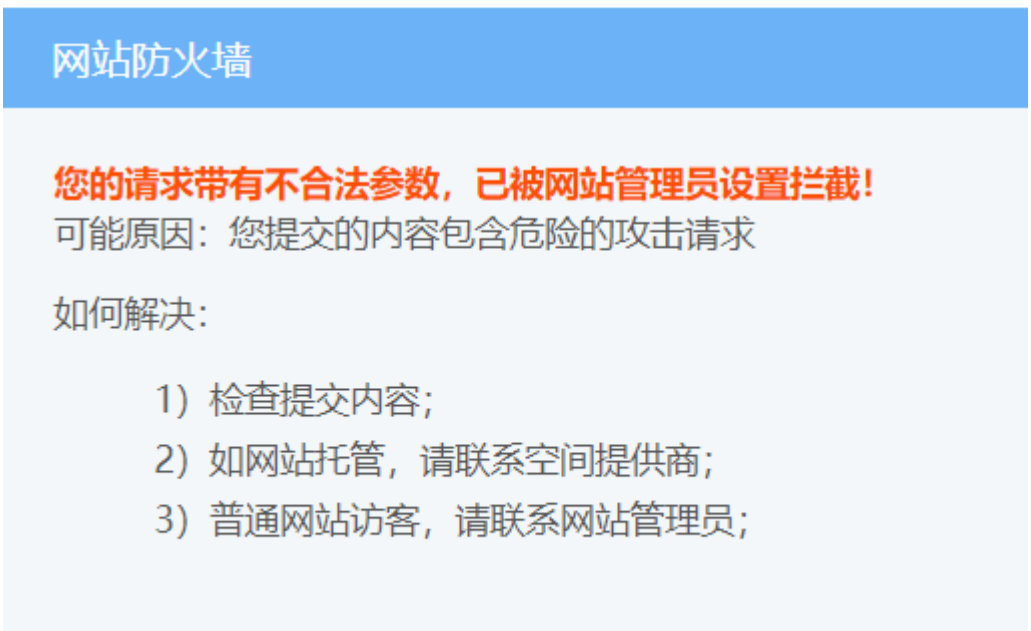
> PS：默认规则没有这点的防护



那么我们可以在args规则中添加比如 `\sor\s+` ，然后 `nginx -s reload` 一下就行了

> PS：如果是从post进行注入，或者cookie中转注入，那么在对应规则里面添加就行，我这边只是演示下防火墙被绕过该怎么解决~（多看看日志）





4.5.CC验证

留个课后作业：**使用ab来测试下nginx+lua的waf对cc的防御效果**

提示：可以使用 `ab -n 2000 -c 200 http://192.168.0.10` 来简单测试

> PS：测试前curl **http://192.168.0.10/hello** 看看返回内容，测试后再curl看看返回内容

扩展：隐藏Nginx版本信息

防止被黑客进行针对性渗透，隐藏下版本信息

> PS：其他配置今天就不详细讲解了，下次讲Nginx的时候会说的

原来：

404 Not Found

---

nginx/1.16.0

配置下： `vi /etc/nginx/nginx.conf`

> http下添加： `server_tokens off;`

```
[root@localhost dnt]# vi /etc/nginx/nginx.conf
[root@localhost dnt]# cat /etc/nginx/nginx.conf

#user  nobody;
worker_processes  1;

#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;

#pid        logs/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       mime.types;
    default_type  application/octet-stream;
    server_tokens off;
    #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
    #                  '$status $body_bytes_sent "$http_referer" '
    #                  '"$http_user_agent" "$http_x_forwarded_for"';

    #access_log  logs/access.log  main;
```

检查下语法：`nginx -t`

不重启的方式加载配置文件：`nginx -s reload`

```
[root@localhost dnt]# nginx -t -c /etc/nginx/nginx.conf
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@localhost dnt]# nginx -s reload -c /etc/nginx/nginx.conf
```

现在效果：

404 Not Found

---

nginx