链滴

Nginx Configuration file

作者：xfell

原文链接：https://ld246.com/article/1548953933188

来源网站：链滴

Install Nginx

```
yum -y install gcc* pcre-devel zlib-devel openssl-devel
useradd -M -s /sbin/nologin nginx   # nginx 运行用户
tar -zxf nginx*
./configure --prefix=/usr/local/nginx
    --user=nginx  --group=nginx --with-http_stub_status_module --with-http_realip_module -
with-http_addition_module
    --with-http_sub_module --with-http_dav_module --with-http_flv_module --with-http_mp4
module
    --with-http_gunzip_module --with-http_gzip_static_module --with-http_random_index_mo
ule --with-http_secure_link_module
    --with-http_stub_status_module --with-http_auth_request_module --with-pcre --with-mail
-with-mail_ssl_module --with-http_spdy_module

    make
    make  install
    chkconfig nginx on
```

Start Scripts Path：/etc/init.d/

```
#!/bin/bash
#chkconfig: 2345 99 20
PROG="/usr/local/nginx/sbin/nginx"
PIDF="/usr/local/nginx/logs/nginx.pid"
PROG_FPM="/usr/local/sbin/php-fpm"
PIDF_FPM="/usr/local/php5/var/run/php-fpm.pid"
case "$1" in
 start)
   $PROG
   $PROG_FPM
   echo " Nginx service start success."
   ;;
 stop)
   kill -s QUIT $(cat $PIDF)
   kill -s QUIT $(cat $PIDF_FPM)
   echo " Nginx service stop success."
   ;;
 restart)
   $0 stop
   $0 start
   ;;
 reload)
   kill -s HUP $(cat $PIDF)
   kill -s HUP $(cat $PIDF_FPM)
   ;;
 *)
   echo " Usage:$0 (start:stop:restart:reload)"
   exit 1
esac
```

Document Descripton

```
user    nginx    nginx   #指定 Nginx 运行时的用户与组
worker_cpu_affinity 0001 0010 0100 1000;
#亲和力配置，让不同的进程使用不同的 CPU 核心，4核心4线程时可以使用以上的配置。
2核心使用01  10；2核心4线程为 01 10 01 10；
8核心8线程使用  00000001 00000010 00000100 00001000 00010000  00100000   01000000 10
00000；
worker_processes  4;  #进程数，即处理器请求的进程，可设置为 CPU 的核心数或者  CPU 核心数
2倍
error_log  logs/error.log  warn;
#全局错误日志定义类型，格式有[ debug|info|notice|warn|error|crit]，一定要设置在warn以上的
别
pid      logs/nginx.pid;
#把进程号记录到文件之中
worker_rlimit_nofile 51200;
#系统打开文件的最大数，可用 cat /proc/sys/fs/file-max 查看，基本使用百分之60左右
events {
    use epoll;
#采用网络 IO 模型，有select 与 epoll 可选，在大并发的情况下最优使用 epoll
    worker_connections  12800;
# #nginx最大连接数=worker连接数*worker进程数
}
http {
    server_tokens off;
    #隐藏 Nginx 的版本号信息
    include       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
    server_names_hash_bucket_size 128;
    #服务器域名 hash 表大小
    server_names_hash_max_size 512;
    #域名服务器的最大 hash 表大小
    client_header_buffer_size 4k;
    #客户端请求头部的缓冲区大小，这个可以根据你的系统分页大小来设置，一般一个请求的头部大
不会超过1k，不过由于一般系统分页都要大于1k，所以这里设置为分页大小。分页大小可以用命令get
onf PAGESIZE取得。
    large_client_header_buffers 8 128k;
    #设定客户请求头缓存
    client_max_body_size 8m;
    #设定通过nginx上传文件的大小
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
    #设定日志 main 的格式
    access_log  logs/access.log  main;
    #全局访问日志
    sendfile        on;
    #开启高效文件传输模式，sendfile指令指定nginx是否调用sendfile函数来输出文件，对于普通应
 用设为 on，如果用来进行下载等应用磁盘IO重负载应用，可设置为off，以平衡磁盘与网络I/O处理
度，降低系统的负载。注意：如果图片显示不正常把这个改成off
    autoindex   off;
    #开启目录列表访问，适合下载服务器，默认关闭
    tcp_nopush     on;
    #防止网络阻塞
```

原文链接：Nginx Configuration file

```
    tcp_nodelay    on;
    #防止网络阻塞
    keepalive_timeout  120;
    #长连接超时时间，单位是秒

    fastcgi_connect_timeout 300;
    fastcgi_send_timeout 300;
    fastcgi_read_timeout 300;
    fastcgi_buffer_size 64k;
    fastcgi_buffers 4 64k;
    fastcgi_busy_buffers_size 128k;
    fastcgi_temp_file_write_size 128k;
   # Fasecgi相关参数是为了改善网站的性能，减少资源占用，提高访问速度

#gzip模块设置
    #开启 gzip 压缩输出
    gzip  on;
    #最小压缩文件大小
    gzip_min_length 1k;
    #压缩缓存区
    gzip_buffers 4 16k;
    #压缩版本（默认为1.1，如果前端是squid 2.5 采用1.0）
    gzip_http_version 1.1;
    #压缩等级
    gzip_comp_level 2;
    #压缩类型，默认包含textml，所以下面就不用再写了，写上去也不有问题，但是会有一个warn
    gzip_types  text/plain application/x-javascript text/css application/xml;   # 安装目录下minme
types 里定义了压缩类型
    #vary header支持。该选项可以让前端的缓存服务器缓存经过GZIP压缩的页面，例如用Squid缓存
   经过Nginx压缩的数据。
    gzip_vary on;
```

Example

```
# 反向代理
location / {
  proxy_pass http://192.168.1.3:80;   # 通过proxy_pass将client的数据请求转发给后端
}

# 负载调度
upstream qxfell_server {
        ip_hash;      #调度算法，默认 rr 轮训，hash常用语解决session共享的问题
        server 192.168.1.240:80    weight  1;
        server 192.168.1.241:80    weight  1;
        server 192.168.1.242:80    weight  1 backup;
        #backup 表示机器处于热备状态，weight代表权重，权重越高代表使用越多
     }

# 在生产环境中需要设置一些信息，这样日志中能获取到真正的访问IP
proxy_redirect off;
proxy_set_header X-Real-IP $remote_addr;    #后端的Web服务器可以通过X-Forwarded-For获
用户真实IP
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

原文链接：Nginx Configuration file

```
#以下是一些反向代理的配置，可选。
proxy_set_header Host $host;
client_max_body_size 10m; #允许客户端请求的最大单文件字节数
client_body_buffer_size 128k; #缓冲区代理缓冲用户端请求的最大字节数，
proxy_connect_timeout 90; #nginx跟后端服务器连接超时时间(代理连接超时)
proxy_send_timeout 90; #后端服务器数据回传时间(代理发送超时)
proxy_read_timeout 90; #连接成功后，后端服务器响应时间(代理接收超时)
proxy_buffer_size 4k; #设置代理服务器（nginx）保存用户头信息的缓冲区大小
proxy_buffers 4 32k; #proxy_buffers缓冲区，网页平均在32k以下的设置
proxy_busy_buffers_size 64k; #高负荷下缓冲大小（proxy_buffers*2）
proxy_temp_file_write_size 64k;
#设定缓存文件夹大小，大于这个值，将从upstream服务器传s

# 自带状态统计(监控会用到)
htpasswd  -c /usr/local/nginx/conf/confpasswd username #利用apache自带的htpasswd生成
码进行登录验证
location /NginxStatus {
      stub_status on;
      access_log on;
      auth_basic "NginxStatus";
      auth_basic_user_file confpasswd;
}

# 虚拟主机
# 在 nginx 配置文件中的 http{} 中间的每一个 server{} 区域都相当于一个虚拟主机，
# 即创建多个虚拟主机即添加多个 server{} 区域即可。
# 注意的是每台虚拟主机之间如果使用相同端口必须通过不同域名区分，如果使用相同域名必须通过
同端口区分，也就是说必须保持独立性
# 例:
   server {
      listen       80;
      server_name  www.qxfell.com;
      #charset koi8-r;
      #access_log  logs/host.access.log  main;
      location / {
      proxy_set_header Host $host;
            proxy_set_header X-Real-IP      $remote_addr;
            proxy_set_header X-Forwarded-For
            $proxy_add_x_forwarded_for;
            proxy_pass  http://qxfell.com;
            expires 1d;
      }
      error_page   500 502 503 504  /50x.html;
      l。0ocation = /50x.html {
         root   html;
      }
   }

# https配置
server {
   listen 80;
   server_name   www.qxfell.com;
   rewrite ^(.*)$ https://$host$1 permanent;   # 用户访问网址http重定向到https  还有另外几种
式 整理好一起更新
```

原文链接：Nginx Configuration file

```
}

server
  {
      listen 443;
      server_name www.qxfell.com qxfell.com;
      ssl                  on;
      ssl_certificate      /etc/nginx/cert/214234223453245.pem;  # 证书文件
      ssl_certificate_key  /etc/nginx/cert/346456456645663.key;
      ssl_session_timeout  5m;
      ssl_protocols  TLSv1 TLSv1.1 TLSv1.2;
      ssl_ciphers  HIGH:!RC4:!MD5:!aNULL:!eNULL:!NULL:!DH:!EDH:!EXP:+MEDIUM;
      ssl_prefer_server_ciphers   on;
      access_log /var/log/nginx/www.qxfell.log main;

     location ~ \.php$ {
        fastcgi_pass   127.0.0.1:9000;
        fastcgi_index  index.php;
        fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
        include        fastcgi_params;
     }
```

Cut Logfile Scripts

```
#!/bin/bash
logfile=/logs/nginx/
logbk=nginx_log/backup/
logtime=`date +%y%m%d`
if [ ! -d $logbk ];then
    mkdir -p $logbk
fi
   cd $logfile
   tar zcvfP ${logtime}.tar.gz *.log >>/dev/null && mv ${logtime}.tar.gz ${logbk} >>/dev/null
   find $logfile -name "*.log" -exec rm {} \; >>/dev/null
   kill -USR1 $(cat /usr/local/server/nginx/logs/nginx.pid )
if [ $? -eq 0 ];then
    echo "${logtime} is good" >>/tmp/cut_nginx.log
else
     echo "${logtime} is bad" >> /tmp/cut_nginx.log
 fi
```