```
#全局配置
#user  nobody;
#程序数，与 CPU 核心数量一致
worker_processes  1;

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

#pid        logs/nginx.pid;

#每个程序的最大并发连接数 动：1000  静：10000
events {
    worker_connections  1024;
}

#Nginx 的 TCP/UDP 调度器,编译安装必须要使用--with-stream 参数开启 4 层代理模块
#stream {
#upstream ssh {
#server 201.1.2.100:22;  #定义后端 ssh 服务器的 ip 和端口
#server 201.1.2.200:22;
#}
#server {
#listen 1235;     #nginx 监听的端口
#proxy_pass ssh;
#proxy_connect_timeout 1s;
#proxy_timeout 3s;
#}
#}

#http 配置
http {
    include       mime.types;
    default_type  application/octet-stream;

    #access_log  logs/access.log  main;
    sendfile        on;
    #tcp_nopush     on;
    #keepalive_timeout  0;
    keepalive_timeout  65;

    #优化 Nginx 数据包头缓存
    #client_header_buffer_size    1k;       #默认请求包头信息的缓存
    #large_client_header_buffers  4 4k;     #大请求包头部信息的缓存个数与容量
```

```
    #服务器内存缓存
    #open_file_cache    max=2000   inactive=20s;
    #open_file_cache_valid      60s;
    #open_file_cache_min_uses 5;
    #open_file_cache_errors    off;
    #设置服务器最大缓存 2000 个文件句柄，关闭 20 秒内无请求的文件句柄
    #文件句柄的有效时间是 60 秒，60 秒后过期
    #只有访问次数超过 5 次会被缓存，关闭缓存错误报错


    #对页面进行压缩处理
    #gzip   on;  #开启压缩
    #gzip_min_length 1000;                #小文件不压缩（字节）
    #gzip_comp_level 4;                   #压缩比率（0-9）
    #gzip_types  text/plain  text/css  application/json  application/x-javascript
text/xml application/xml application/xml+rss text/javascript;
    #对特定文件压缩，类型参考 mime.types

    #log_format      main      '$remote_addr  -  $remote_user  [$time_local]
"$request" '
    #  日志格式    主要       客户端 ip       客户端用户名    本地时间          请求
    #              '$status $body_bytes_sent "$http_referer" '
    #              状态（码） 网页大小（流量） 从别处跳转的（百度--tmooc）
    #              '"$http_user_agent" "$http_x_forwarded_for"';
    #                代理 http              【查：和代理服务器有关（无值或 ip）】

    #反向代理
    #upstream web {
    # ip_hash;
    # server 201.1.2.100:80 weight=2;
    # server 201.1.2.200:80 max_fails=1 fail_timeout=30;
    # server 201.1.3.3:80 down;
    # }

    server {
        listen          80;
        server_name   www.a.com;

        #rewrite ^/ http://www.tmooc.cn/;   #地址重写-跳转首页网页
        #rewrite ^/(.*)$ http://www.tmooc.cn/$1; #地址重写-跳转对应网页
        #charset koi8-r;

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

```
 location / {
#反向代理，通过 proxy_pass 将用户的请求转发给 web 集群
#       proxy_pass http://web;
        root     html;
        index    index.html index.htm;
   #rewrite /a.aa /test.php;    #地址重写-跳转地址栏
}

   #地址重写，实现相同链接不同返回页面
   #if ($http_user_agent ~* firefox){       #~表示比较匹配符  *不区分大小写
                                            #识别客户端 firefox 浏览器
   # rewrite    ^(.*)$    /firefox/$1;
   # }

    #查看服务器状态信息
    #编译安装时使用--with-http_stub_status_module 开启状态页面模块
    #location /abc {
      #stub_status on;
      #allow IP 地址;
      #deny IP 地址;
    #}

 #定义对客户端静态页面的缓存时间
 #location ~* \.(jpg|jpeg|gif|png|css|js|ico|xml)$ {
 #expires           30d;              #定义客户端缓存时间为 30 天
 #}

 #自定义 404 错误页面
  #error_page   404                  /404.html;

 # redirect server error pages to the static page /50x.html

 error_page     500 502 503 504    /50x.html;
 location = /50x.html {
      root     html;
 }

 # proxy the PHP scripts to Apache listening on 127.0.0.1:80
 #
 #location ~ \.php$ {
 #       proxy_pass     http://127.0.0.1;
 #}
```

```
    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #动静分离,将 include 改成 include    fastcgi.conf;
    #location ~ \.php$ {
    #       root           html;
    #       fastcgi_pass   127.0.0.1:9000;
    #       fastcgi_index  index.php;
    #       include        fastcgi.conf;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #       deny   all;
    #}
}


# another virtual host using mix of IP-, name-, and port-based configuration
#基于域名的虚拟主机：打开配置文件 server 容器及改默认网页
server {
    listen          80;
    server_name         www.b.com;      #虚拟主机域名

  #用户认证
  #auth_basic "input password:";                  #认证提示符
  #auth_basic_user_file "/usr/local/nginx/pass"; #认证密码文件及存放路径

  #   server_name   somename   alias   another.alias;
  #基于域名的虚拟主机:创建网站根目录/usr/local/nginx/www 及首页文件
    location / {
        root    www;
        index   index.html index.htm;
    }
}


# HTTPS server
#SSL 虚拟主机,源码安装 Nginx 时必须使用--with-http_ssl_module 参数
#server {
#       listen          443 ssl;
#       server_name   www.c.com;

#       ssl_certificate         cert.pem;   #证书文件
```

```
    #       ssl_certificate_key    cert.key;   #私钥文件

    #       ssl_session_cache      shared:SSL:1m;
    #       ssl_session_timeout    5m;

    #       ssl_ciphers    HIGH:!aNULL:!MD5;
    #       ssl_prefer_server_ciphers    on;

    #       location / {
    #           root     ccc;
    #           index    index.html index.htm;
    #       }
    #}

}

#添加模块：
#加密的 http---源码安装 Nginx 时必须使用--with-http_ssl_module 参数
#反向代理，TCP/UDP 调度--编译安装必须要使用--with-stream 参数（开启 4 层代理模块）
#查看服务器状态--编译安装时使用--with-http_stub_status_module 开启状态页面模块

#查看服务器的状态
#编译安装时使用--with-http_stub_status_module 开启状态页面模块
#nginx 的状态页面，Active connections：当前活动的连接数量。
#Accepts：已经接受客户端的连接总数量。
#Handled：已经处理客户端的连接总数量。
#（一般与 accepts 一致，除非服务器限制了连接数量）。
#Requests：客户端发送的请求数量。
#Reading：当前服务器正在读取客户端请求头的数量。
#Writing：当前服务器正在写响应信息的数量。
#Waiting：当前多少客户端在等待服务器的响应

#日志切割
#命令行操作：创建一个日志切割脚本，结合周期性计划任务，实现定时日志切割
#       #!/bin/bash
#       date=`date +%Y%m%d`
#       logpath=/usr/local/nginx/logs
#       mv $logpath/access.log $logpath/access-$date.log
#       mv $logpath/error.log $logpath/error-$date.log
#       kill -USR1 $(cat $logpath/nginx.pid)

#优化并发量：1.改全局配置
#                   2.优化 Linux 内核参数
```

#ulimit -a　　#查看所有属性值
#ulimit -Hn 100000　#设置硬限制　ulimit -Sn 100000　#设置软限制（临时规则）
#vim /etc/security/limits.conf（添加）
#（1）*　soft　nofile　100000　　（2）*　hard　nofile　100000

#反向代理，TCP/UDP 调度
#编译安装必须要使用--with-stream 参数（开启 4 层代理模块）
#配置 Nginx 服务器，添加服务器池
#使用 upstream 定义后端服务器集群，集群名称任意
#使用 server 定义集群中的具体服务器和端口
#配置 upstream 服务器集群的调度算法
#通过 ip_hash 设置调度规则为：相同客户端访问相同服务器
#配置 upstream 服务器集群池属性
#weight 设置服务器权重值，默认值为 1
#max_fails 设置最大失败次数
#fail_timeout 设置失败超时时间，单位为秒
#down 标记服务器已关机，不参与集群调度

#部署 LNMP 环境:yum 下载 mariadb、mariadb-server、mariadb-devel php、php-mysql
#　　php-fpm-5.4.16-42.el7.x86_64.rpm　源码包下载：nginx
(防止 httpd 干扰)启动服务：systemctl start　mariadb
systemctl start php-fpm　/usr/local/nginx/sbin/nginx
#构建 LNMP 平台：打开配置文件动静分离 location(改成这样：include　fastcgi.conf;)

#部署 LNMP+memcached 环境：1.yum 下载 memcached　php-pecl-memcache
2.启动
#查看服务器本地的 Session 信息　ls /var/lib/php/session/
#在后端每台 LNMP 服务器上部署 Session 共享
#修改 PHP-FPM 配置文件：　/etc/php-fpm.d/www.conf
#文件的最后两行：php_value[session.save_handler] = memcache
#　　php_value[session.save_path] = "tcp://192.168.2.5:11211"
#　定义 Session 信息存储在公共的 memcached 服务器上，主机参数中为 memcache（没有 d）
#　通过 path 参数定义公共的 memcached 服务器在哪（服务器的 IP 和端口）
#下载 telnet（测试 memcached 服务器功能）　telnet　192.168.4.5　11211
#telnet ip 端口（set a 0 180 3　get--add -replace -append -delete
#　　flush_all 清空所有 stats 查看状态 quit）

#地址重写，能将客户端输入地址重定向到别的地址或网页，也能实现不同浏览器返回不同页面
#格式：rewrite 旧地址 新地址 [选项]；
#选项：last 不再读其他 rewrite　break 不再读其他语句，结束请求
#　redirect 临时重定向　permament 永久重定向

#nginx 命令用法
#启动服务：nginx
#关闭服务：nginx -s stop
#重新加载：nginx -s reload
#查看版本信息：nginx -V
#查看启动端口信息：netstat -anpltu 或者 ss -anpltu
#-a 显示所有端口的信息 -n 以数字格式显示端口号
#-t 显示 TCP 连接的端口 -u 显示 UDP 连接的端口
#-l 显示服务正在监听的端口信息，如 httpd 启动后，会一直监听 80 端口
#-p 显示监听端口的服务名称是什么

#源码 nginx 安装:1.下载依赖包 gcc openssl-devel pcre-devel zlib-devel
2.创建执行用户 3.tar 解压 4../configure 系统检查和配置
5.编译并安装 make && make install 6.创建软连接 ln -s

#平滑升级:1.tar 解压 2.cd 到对应目录 3../configure 系统检查和配置 3.编译（make）
#4.备份旧的，并将新版本拷贝到安装主程序目录下（cp objs/nginx 旧安装目录）
#5.关闭服务，再重新启动 6.查看版本
#源路径备份：mv /usr/local/nginx/sbin/nginx{,.bak}

#用户认证：改配置文件及下载 httpd-tools,再创建认证用户和认证密码文件
##htpasswd -c /usr/local/nginx/pass tom #创建认证密码文件加选项-c
##htpasswd /usr/local/nginx/pass jerry
##cat /usr/local/nginx/pass #查看

#SSL 虚拟主机:打开配置文件的 ssl 虚拟主机
#源码安装 Nginx 时必须使用--with-http_ssl_module 参数
#cd /usr/local/nginx/conf
openssl genrsa > cert.key #生成私钥
#openssl req -new -x509 -key cert.key > cert.pem #生成证书
#测试 firefox https://www.c.com