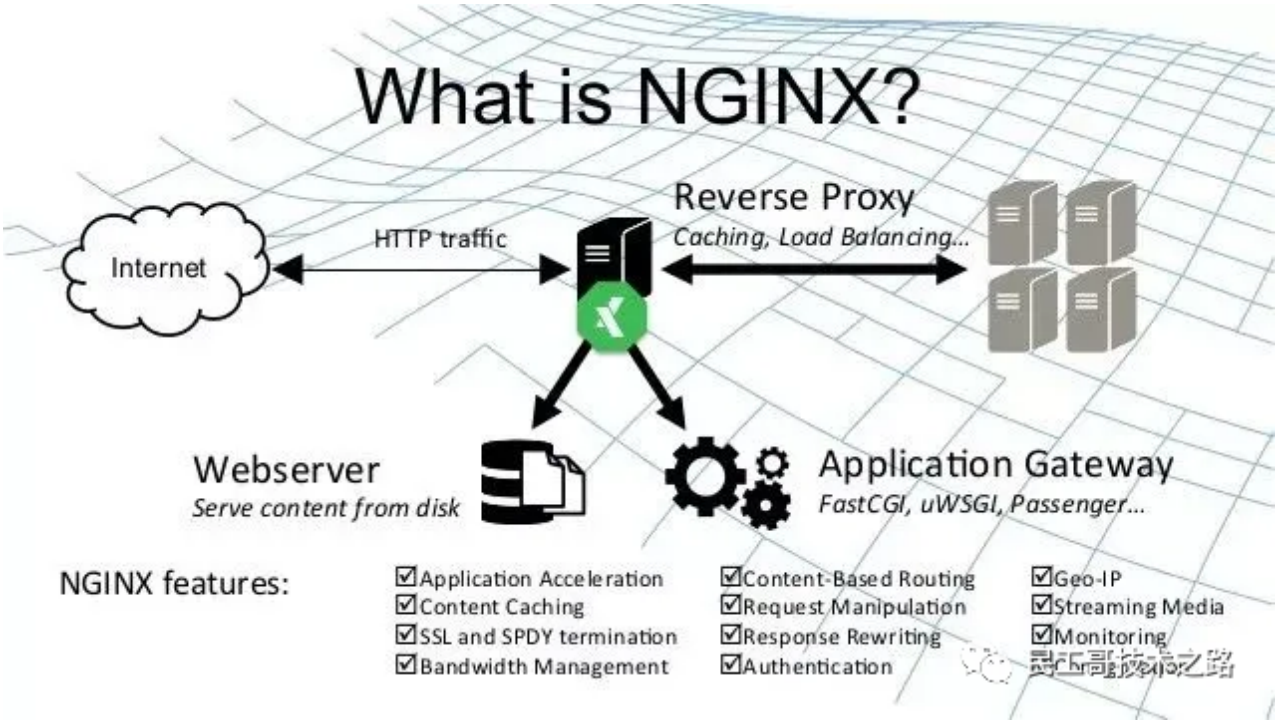
Nginx 常用配置汇总！从入门到干活足矣

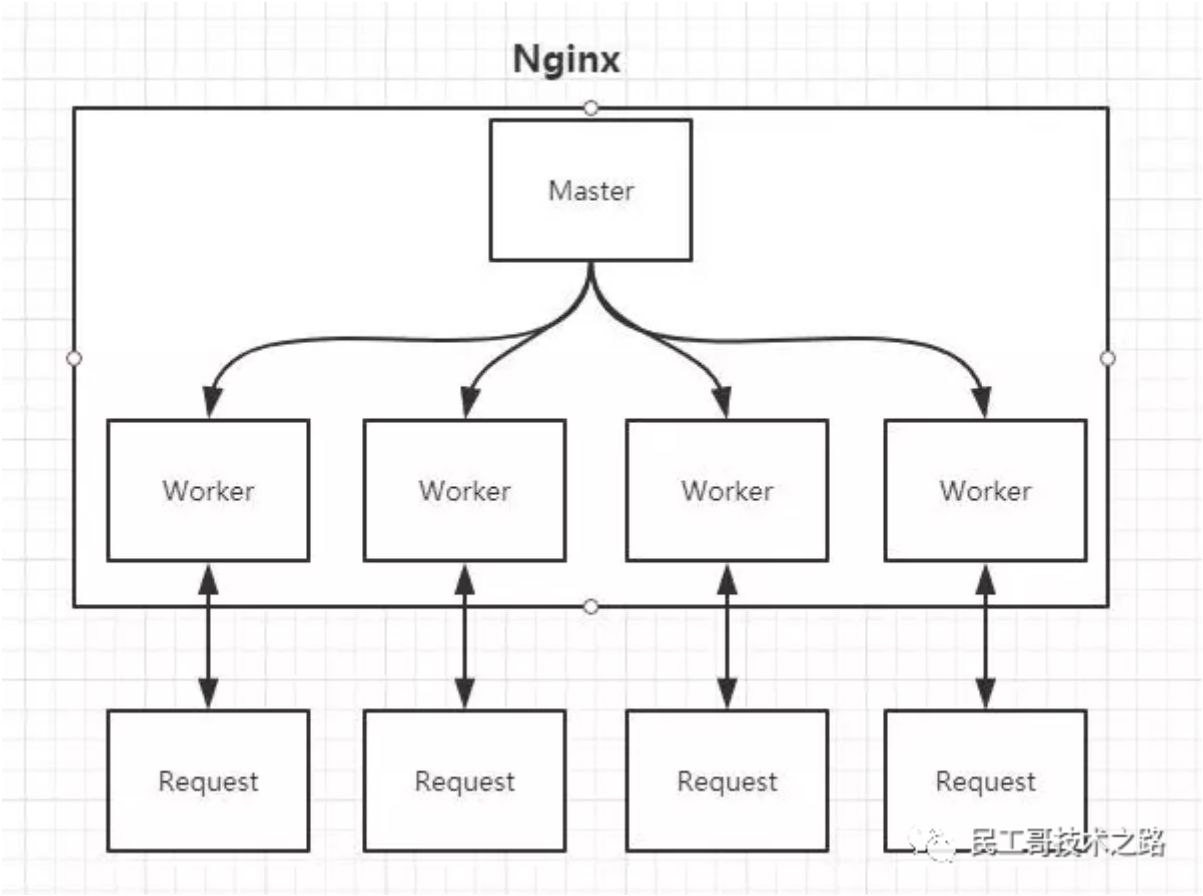
mp.weixin.qq.com/s/Je1prqCxGnw1_J5-Mkm40w

众所周知，Nginx 是 Apache服务不错的替代品。其特点是占有内存少，并发能力强，事实上 Nginx 的并发能力在同类型的网页服务器中表现较好，因此国内知名大厂例如：淘宝，京东，百度，新浪，网易，腾讯等等都在使用Nginx网站。



Nginx简介

Nginx 是开源、高性能、高可靠的 Web 和反向代理服务器，而且支持热部署，同时也提供了 IMAP/POP3/SMTP 服务，可以不间断运行，提供热更新功能。占用内存少、并发能力强，最重要的是，Nginx 是免费的并可以商业化，配置使用都比较简单。



Nginx 特点

- 高并发、高性能
- 模块化架构使得它的扩展性非常好
- 异步非阻塞的事件驱动模型这点和 Node.js 相似
- 无需重启可不间断运行
- 热部署、平滑升级
- 完全开源，生态好

Nginx 最重要的几个使用场景：

- 静态资源服务
- 反向代理服务，包括缓存、负载均衡等
- API 服务，OpenResty

所以，今天民工哥就给大家整理一份 Nginx的常用配置清单，供大家学习与生产配置参考使用。主要包括以下三个方面：

- 基础配置
- 高级配置
- 安全配置

基础配置

去掉不用的 Nginx 模块

```
./configure --without-module1 --without-module2 --without-module3
例如：
./configure --without-http_dav_module --withouthttp_spdy_module
#注意事项：配置指令是由模块提供的。确保你禁用的模块不包含你需要使用的指令！在决定禁用模块之前，应该检查Nginx文档中每个模块可用的指令列表。
```

Nginx 版本的平滑升级与回滚

1分钟搞定 Nginx 版本的平滑升级与回滚

进程相关的配置

```
worker_processes 8;
#Nginx 进程数，建议按照CPU数目来指定，一般为它的倍数 (如,2个四核的CPU计为8)。

worker_rlimit_nofile 65535;
#一个Nginx 进程打开的最多文件描述符数目

worker_connections 65535;
#每个进程允许的最多连接数
```

监听端口

```
server {
        listen       80;   #监听端口
        server_name  www.mingongge.com;  #域名信息
        location / {
            root   /www/www;   #网站根目录
            index  index.html index.htm;  #默认首页类型
            deny 192.168.2.11;   #禁止访问的ip地址，可以为all
            allow 192.168.3.44；#允许访问的ip地址，可以为all
        }
        }
```

小技巧补充：域名匹配的四种写法

```
精确匹配：server_name www.mingongge.com ;
左侧通配：server_name *.mingongge.com ;
右侧统配：server_name www.mingongge.* ;
正则匹配：server_name ~^www\.mingongge\.*$ ;

匹配优先级：精确匹配 > 左侧通配符匹配 > 右侧通配符匹配 > 正则表达式匹配
```

配置 Nginx 状态页面

```
[root@proxy ~]# cat /usr/local/nginx/conf/nginx.conf
… …

location /NginxStatus {
      stub_status           on;
      access_log            on;
      auth_basic            "NginxStatus";
      auth_basic_user_file  conf/htpasswd;
        }
… …
[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload
```

Nginx 日志（访问与错误日志管理）

```
error_log  /var/log/nginx/error.log warn;
#配置错误日志的级别及存储目录

events {
    worker_connections  1024;
}
http {
..................
    log_format  main '$remote_addr - $remote_user [$time_local] "$request" '
                     '$status $body_bytes_sent "$http_referer" '
                     '"$http_user_agent" "$http_x_forwarded_for"';
    #配置日志的模式
    access_log  /var/log/nginx/access.log main;
    #配置访问日志存储目录
}
```

以上配置只是Nginx自身关于日志的基本配置，在实际生产环境中，我们需要收集日志、分析日志，才定更好的去定位问题，推荐给大家：超强干货！通过filebeat、logstash、rsyslog 几种方式采集 nginx 日志

http 相关的配置

```
http {
    sendfile  on                    #高效传输文件的模式 一定要开启
    keepalive_timeout   65          #客户端服务端请求超时时间

 }
```

静态资源配置

```
server {
listen 80;
server_name mingongge.com;
location /static {
  root /wwww/web/web_static_site;
  }
}
```

也可以使用下面的方法

```
location /image {
 alias /web/nginx/static/image/;
}
注意:使用alias末尾一定要添加/,并且它只能位于location中
```

反向代理

比如生产环境（同一台服务中）有不同的项目，这个就比较实用了，用反向代理去做请示转发。

```
http {
.............
    upstream product_server{
        127.0.0.1:8081;
    }

    upstream admin_server{
        127.0.0.1:8082;
    }

    upstream test_server{
        127.0.0.1:8083;
    }

server {



#默认指向product的server
  location / {
      proxy_pass http://product_server;
      }

  location /product/{
      proxy_pass http://product_server;
     }

  location /admin/ {
      proxy_pass http://admin_server;
     }

  location /test/ {
      proxy_pass http://test_server;
      }
    }
}
```

更多关于 Nginx 实践：location 路径匹配

负载均衡

```
upstream server_pools {
  server 192.168.1.11:8880   weight=5;
  server 192.168.1.12:9990   weight=1;
  server 192.168.1.13:8989   weight=6;
  #weigth参数表示权值，权值越高被分配到的几率越大
}
server {
  listen 80;
  server_name mingongge.com;
  location / {
  proxy_pass http://server_pools;
   }
}
```

代理相关的其它配置

```
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;      #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
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

proxy_set_header Host $host;
proxy_set_header X-Forwarder-For $remote_addr;  #获取客户端真实IP
```

高级配置

重定向配置

```
location / {
 return 404; #直接返回状态码
}
location / {
 return 404 "pages not found"; #返回状态码 + 一段文本
}
location / {
 return 302 /blog ; #返回状态码 + 重定向地址
}
location / {
 return https://www.mingongge.com ; #返回重定向地址
}
```

示例如下

```
server {
listen 80;
server_name www.mingongge.com;
return 301 http://mingongge.com$request_uri;
}
server {
listen 80;
server_name www.mingongge.com;
location /cn-url {
   return 301 http://mingongge.com.cn;
   }
}
```

```
server{
  listen 80;
  server_name mingongge.com; # 要在本地hosts文件进行配置
  root html;
  location /search {
   rewrite ^/(.*) https://www.mingongge.com redirect;
  }


  location /images {
   rewrite /images/(.*) /pics/$1;
  }


  location /pics {
   rewrite /pics/(.*) /photos/$1;
  }


  location /photos {

    }
}
```

设置缓冲区容量上限

这样的设置可以阻止缓冲区溢出攻击（同样是Server模块）

```
client_body_buffer_size 1k;
client_header_buffer_size 1k;
client_max_body_size 1k;
large_client_header_buffers 2 1k;
#设置后，不管多少HTTP请求都不会使服务器系统的缓冲区溢出了
```

限制最大连接数

在http模块内server模块外配置limit_conn_zone，配置连接的IP,在http，server或location模块配置limit_conn，能配置IP的最大连接数。

```
limit_conn_zone $binary_remote_addr zone=addr:5m;
limit_conn addr 1;
```

Gzip压缩

```
gzip_types

#压缩的文件类型
 text/plain text/css
 application/json
 application/x-javascript
 text/xml application/xml
 application/xml+rss
 text/javascript

gzip on;
#采用gzip压缩的形式发送数据

gzip_disable "msie6"
#为指定的客户端禁用gzip功能

gzip_static;
#压缩前查找是否有预先gzip处理过的资源

gzip_proxied any;
#允许或者禁止压缩基于请求和响应的响应流

gzip_min_length  1000;
#设置对数据启用压缩的最少字节数

gzip_comp_level 6;
#设置数据的压缩等级
```

缓存配置

```
open_file_cache
#指定缓存最大数目以及缓存的时间

open_file_cache_valid
#在open_file_cache中指定检测正确信息的间隔时间

open_file_cache_min_uses
#定义了open_file_cache中指令参数不活动时间期间里最小的文件数

open_file_cache_errors
#指定了当搜索一个文件时是否缓存错误信息

location ~ .*\.(gif|jpg|jpeg|png|bmp|swf)$
#指定缓存文件的类型

        {
        expires 3650d;
           #指定缓存时间
        }
        location ~ .*\.(js|css)?$
        {
        expires 3d;
        }
```

SSL 证书配及跳转HTTPS配置

```
server {
      listen 192.168.1.250:443 ssl;
      server_tokens off;
      server_name mingonggex.com www.mingonggex.com;
      root /var/www/mingonggex.com/public_html;
      ssl_certificate /etc/nginx/sites-enabled/certs/mingongge.crt;
      ssl_certificate_key /etc/nginx/sites-enabled/certs/mingongge.key;
      ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
}

# Permanent Redirect for HTTP to HTTPS
server
{
listen 80;
server_name mingongge.com;
https://$server_name$request_uri;
}
```

流量镜像功能

```
location / {
    mirror /mirror;
    proxy_pass http://backend;
}

location = /mirror {
    internal;
    proxy_pass http://test_backend$request_uri;
}
```

限流功能

流量限制配置两个主要的指令，`limit_req_zone` 和 `limit_req`

```
limit_req_zone $binary_remote_addr zone=mylimit:10m rate=10r/s;

server {
    location /login/ {
        limit_req zone=mylimit;

        proxy_pass http://my_upstream;
    }
}
```

更多、更详细的限流配置请参考：葵花宝典！一文搞定 Nginx 限流配置

Nginx常用的内置变量

变量名	功能
$host	请求信息中的 Host，如果请求中没有 Host 行，则等于设置的服务器名
$request_method	客户端请求类型，如 GET、POST
$remote_addr	客户端的 IP 地址
$args	请求中的参数
$content_length	请求头中的 Content-length 字段
$http_user_agent	客户端 agent 信息
$http_cookie	客户端 cookie 信息
$remote_addr	客户端的 IP 地址
$remote_port	客户端的端口
$server_protocol	请求使用的协议，如 HTTP/1.0、HTTP/1.1\
$server_addr	服务器地址
$server_name	服务器名称
$server_port	服务器的端口号

安全配置

禁用server_tokens项

server_tokens在打开的情况下会使404页面显示Nginx的当前版本号。这样做显然不安全，因为黑客会利用此信息尝试相应Nginx版本的漏洞。只需要在nginx.conf中http模块设置server_tokens off即可，例如：

```
server {
    listen 192.168.1.250:80;
    Server_tokens off;
    server_name mingongge.com www.mingongge.com;
    access_log /var/www/logs/mingongge.access.log;
    error_log /var/www/logs/mingonggex.error.log error;
    root /var/www/mingongge.com/public_html;
    index index.html index.htm;
}
#重启Nginx后生效：
```

禁止非法的HTTP User Agents

User Agent是HTTP协议中对浏览器的一种标识，禁止非法的User Agent可以阻止爬虫和扫描器的一些请求，防止这些请求大量消耗Nginx服务器资源。

为了更好的维护，最好创建一个文件，包含不期望的user agent列表例如/etc/nginx/blockuseragents.rules包含如下内容：

```
map $http_user_agent $blockedagent {
    default 0;
    ~*malicious 1;
    ~*bot 1;
    ~*backdoor 1;
    ~*crawler 1;
    ~*bandit 1;
}
```

然后将如下语句放入配置文件的server模块内

```
include /etc/nginx/blockuseragents.rules;
并加入if语句设置阻止后进入的页面：
```

阻止图片外链

```
location /img/ {
      valid_referers none blocked 192.168.1.250;
        if ($invalid_referer) {
          return 403;
      }
}
```

封杀恶意访问

挺带劲！通过 Nginx 来实现封杀恶意访问

禁掉不需要的 HTTP 方法

一些web站点和应用，可以只支持GET、POST和HEAD方法。在配置文件中的 serve r模块加入如下方法可以阻止一些欺骗攻击

```
if ($request_method !~ ^(GET|HEAD|POST)$) {
    return 444;
}
```

禁止 SSL 并且只打开 TLS

尽量避免使用SSL，要用TLS替代，以下配置可以放在Server模块内

```
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
```

通过这一系列的配置之后，相信你的Nginx服务器足够应付实际生产需求了。

也欢迎大家积极留言补充这份常用配置清单，以便它更完整、更完善。