Nginx服务器

一、 Nginx服务器简介

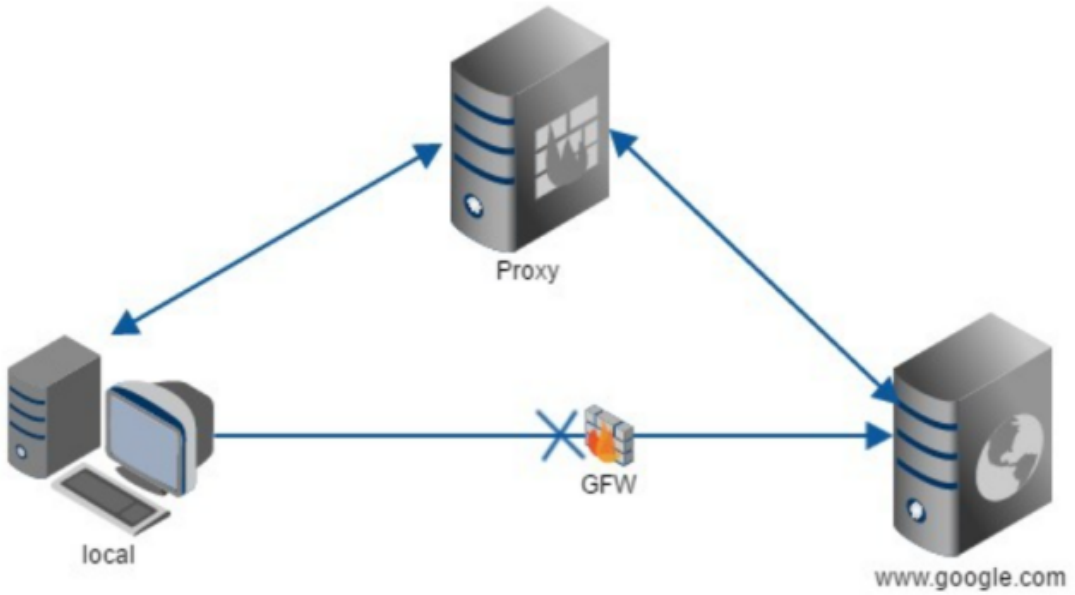
Nginx 是一个高性能的HTTP和反向代理web服务器，同时也提供了IMAP/POP3/SMTP服务。Nginx是由伊戈尔·赛索耶夫为俄罗斯访问量第二的Rambler.ru站点（俄文：Рамблер）开发的，第一个公开版本0.1.0发布于2004年10月4日。

其将源代码以类BSD许可证的形式发布，因它的稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名。2011年6月1日，nginx 1.0.4发布。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，中国大陆使用nginx网站用户有：百度、京东、新浪、网易、腾讯、淘宝等。

二、 正向代理和反向代理

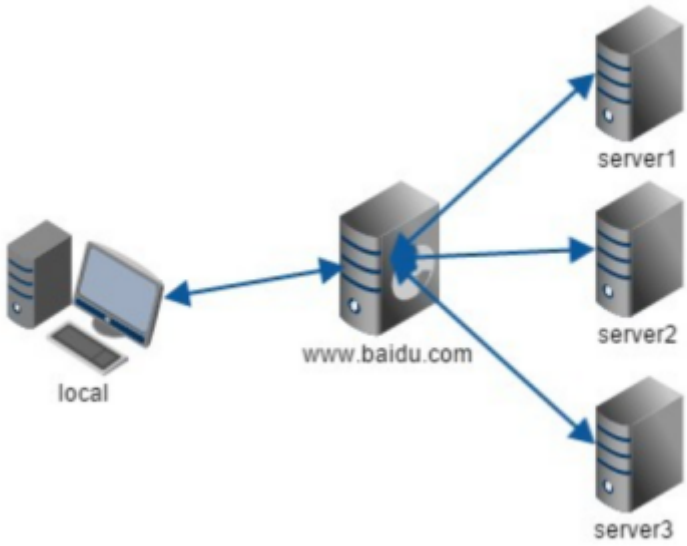
1、 正向代理

我们常说的代理也就是只正向代理，正向代理的过程，**它隐藏了真实的请求客户端**，服务端不知道真实的客户端是谁，客户端请求的服务都被代理服务器代替来请求，某些科学上网工具扮演的就是典型的正向代理角色。用浏览器访问http://www.google.com时，被残忍的block，于是你可以在国外搭建一台代理服务器，让代理帮你去请求google.com，代理把请求返回的相应结构再返回给你。



2、 反向代理

反向代理隐藏了真实的服务端，当我们请求www.baidu.com的时候，背后可能有成千上万台服务器为我们服务，但具体是哪一台，你不知道，也不需要知道，你只需要知道反向代理服务器是谁就好了，www.baidu.com就是我们的反向代理服务器，反向代理服务器会帮我们把请求转发到真实的服务器那里去。Nginx就是性能非常好的反向代理服务器，用来做负载均衡。



三、 CentOS 7 中 Nginx 的安装

1、安装epel

```
yum -y install epel-release
```

2、查看yum源中的nignx的信息

```
yum info nginx
```

3、开始安装nginx

```
yum install nginx
```

4、启动 nginx

```
nginx
```

5、 Nginx相关命令

```
nginx -s reopen #重启Nginx

nginx -s reload #重新加载Nginx配置文件，然后以优雅的方式重启Nginx

nginx -s stop #强制停止Nginx服务

nginx -s quit #优雅地停止Nginx服务（即处理完所有请求后再停止服务）

nginx -t #检测配置文件是否有语法错误，然后退出

nginx -v #显示版本信息并退出

killall nginx #杀死所有nginx进程
```

四、 Nginx配置文件详解（/etc/nginx/nginx.conf)

```
...              #全局块

events {         #events块
   ...
}

http      #http块
{
    ...   #http全局块
    server        #server块
    {
        ...       #server全局块
        location [PATTERN]   #location块
        {
            ...
        }
        location [PATTERN]
        {
            ...
        }
    }
    server
    {
      ...
    }
    ...     #http全局块
}
```

Nginx配置文件主要分为三大块：全局块，events块，http块

全局块：从配置文件开始到events块开始之前的内容，都属于全局块，配置影响nginx全局的指令。一般有运行nginx服务器的用户组，允许生成的工作进程数，错误日志存放路径， nginx进程pid存放路径 ，配置文件引入等。

events块：配置影响nginx服务器或与用户的网络连接。例如： worker_connections 1024表示单个工作进程可以同时建立1024个外部连接。

http块：该模块是Nginx服务器配置中的重要部分，代理、缓存和日志定义等绝大多数的功能和第三方模块的配置都可以放在这个模块中。需要注意的是：http块可以包括http全局块、server块。

（1）http全局块：http全局块配置的指令包括文件引入、MIME-TYPE定义、日志自定义、连接超时时间、单链接请求数上限等。

（2）server块：这块和虚拟主机有密切关系，虚拟主机从用户角度看，和一台独立的硬件主机是完全一样，该技术的产生是为了节省互联网服务器硬件成本。每个http块可以包括多个server块，而每个server块就相当于一个虚拟主机。而每个server块也分为全局server块，以及可以同时包含多个location块。

（3）全局server块：最常见的配置是本虚拟机主机的监听配置和本虚拟主机的名称或IP配置

（4）location块：一个server块可以配置多个location块，这块的主要作用是基于nginx服务器接收到的请求字符串，对特定的请求进行处理，如地址定向、数据缓存和应答控制等功能，还有许多第三方模块的配置也在这里进行。

server块：这块和虚拟主机有密切关系，虚拟主机从用户角度看，和一台独立的硬件主机是完全一样，该技术的产生是为了节省互联网服务器硬件成本。

每个http块可以包括多个server块，而每个server块就相当于一个虚拟主机。

每个server块也分为全局server块，以及可以同时包含多个location块。

```
user nginx;      #运行Nginx的用户
worker_processes auto;  #允许生成的worker进程数，auto表示自动分配，一般等于服务器核数
error_log /var/log/nginx/error.log;  #错误日志存放路径
pid /run/nginx.pid;  #Nginx进程pid存放路径
include /usr/share/nginx/modules/*.conf;  # 配置文件引入

events {
    worker_connections 1024;  #单个工作进程可以同时建立的外部连接数量
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"'; #日志格式

    access_log  /var/log/nginx/access.log  main;  #访问日志存放位置以及使用的日志格式

    sendfile            on;     #提高文件传输速度
    tcp_nopush          on;     #启用后数据包会累积一下再一起传输，可以提高一些传输效率。
    tcp_nodelay         on;     #小的数据包不等待直接传输。默认为on。看上去是和
tcp_nopush相反的功能，但是两边都为 on 时 nginx 也可以平衡这两个功能的使用。
    keepalive_timeout   65;  #长连接超时时间
    types_hash_max_size 4096;  #该参数指定了存储MIME type与文件扩展名的散列的最大大小

    include             /etc/nginx/mime.types; # 文件扩展名与文件类型映射表
    default_type        application/octet-stream;  #如果没对应文件的扩展名，就用
default_type定义的处理方式。

    include /etc/nginx/conf.d/*.conf;

    server {
        listen       80;  #监听端口
        listen       [::]:80;  #ipv6监听端口
        server_name  _;  #用于配置基于名称的虚拟主机
        root         /usr/share/nginx/html; #网站根目录
        server_tokens off;  #关闭服务器令牌信息，配置文件默认没有
        limit_rate 1024000; #限制响应给客户端的传输速率，单位是字节默认值0表示无限制，可以
使用一个大文件用来实验效果
        autoindex off;  #是否启用目录索引

        include /etc/nginx/default.d/*.conf;

        error_page 404 /404.html; #定义了当服务器返回 404 错误时显示的页面
        location = /404.html {  #指定了当请求的路径为 /404.html 时，如何处理该请求。
        }
        error_page 500 502 503 504 /50x.html;  #定义了当服务器返回 500、502、503 或
504 错误时，应显示的页面
        location = /50x.html {  #指定了当请求的路径为 /50x.html 时，如何处理该请求
```

```
        }
    }
}
```

日志格式

配置项	说明
$remote_addr	记录客户端的ip地址
$remote_user	用来记录客户端用户名称
$time_local	用来记录访问时间与时区
$request	用来记录请求的url与http协议
$status	用来记录请求状态；成功是200
$body_bytes_sent	记录发送给客户端文件主体内容大小
$http_referer	用来记录从那个页面链接访问过来的
$http_user_agent	记录客户端浏览器的相关信息

Nginx访问控制

```
示例1：禁止所有人访问 /test/index.html 文件
location = /test/index.html {
deny all;
}

示例2：仅拒绝 192.168.0.100 访问 /test/index.html 文件
location = /test/index.html {
deny 192.168.0.100;
allow all;
}

示例3：仅允许 192.168.0.100 访问 /test/index.html 文件
location = /test/index.html {
allow 192.168.0.100;
deny all;
}

示例4：仅允许特定用户访问 /test/index.html 文件

#安装httpd-tools
yum install httpd-tools -y

#创建认证用户
htpasswd -c /var/passwd test

#编辑配置文件
#访问 /test/index.html文件时需要登录认证
location = /test/index.html {
auth_basic "Please input your username and password";
```

```
auth_basic_user_file /var/passwd;
}

示例5：禁止对于特定目录的所有访问，注意给目录设置时要把目录后面的 / 加上，否则会出现意料之外的匹配，如给 /test/ 目录设置时如果没写最后的 / 那么他会匹配到 /test、/test1、/testuser 等等的以 /test 开头的内容。
location /test/ {
deny all;
}
```

注意

对目录设置访问权限时要去掉 location 后面的 =，有 = 表示进行精确匹配，只有访问的路径完全等于 location 后面的内容时才会匹配到。

去掉 = 表示进行前缀匹配，匹配那些以指定的内容开头的 URI，可以看下面的案例。

如果同时存在精确匹配和前缀匹配，那么精确匹配的优先级高于前缀匹配。

示例

```
#执行下列命令新建两个目录
cd /usr/share/nginx/html/
mkdir test test1
#执行下列命令新建两个文件
echo "This is test" > test/index.html;echo "This is test1" > test1/index.html
#在Nginx配置中添加如下内容
location /test/ {
deny all;
}
location = /test/index.html {
allow all;
}
```

访问路径	访问结果
http://NginxIP/test/	403
http://NginxIP/test/index.html	This is test

说明如果同时存在精确匹配和前缀匹配时，精确匹配优先。

设置虚拟主机

```
通过端口设置
server {
    listen 81;
    server_name _;
    root /usr/share/nginx/html/a;
}
```

通过域名设置

```
server {
    listen 80;
    server_name www.a.com;
    root /usr/share/nginx/html/a;
}
```