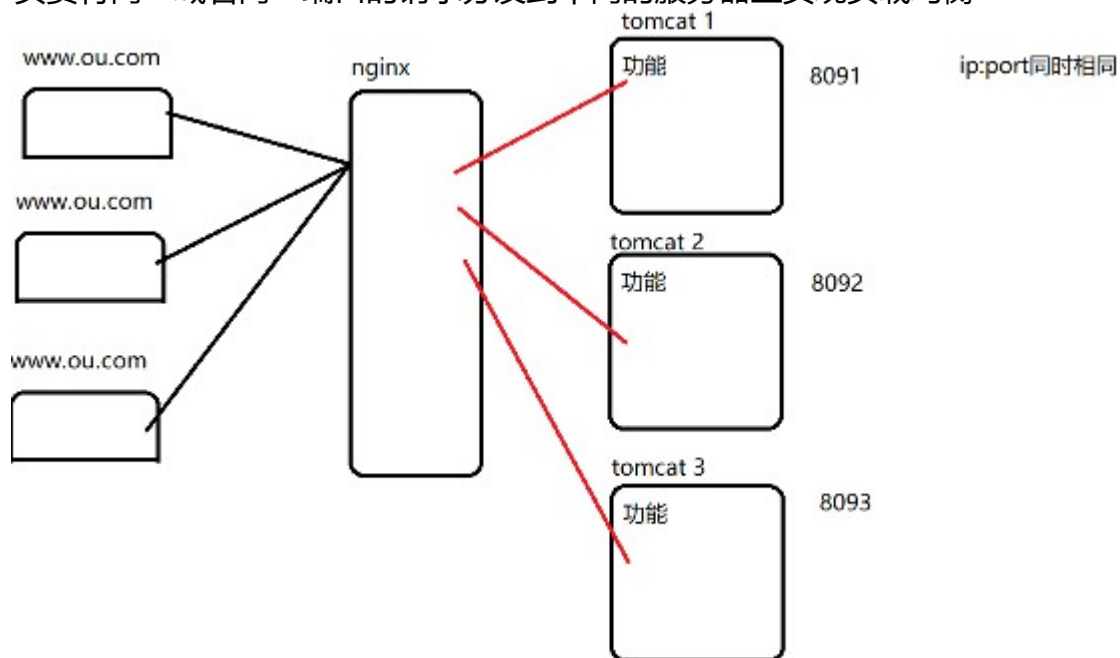


一 Nginx 简介

nginx 是一个高性能(理论值单台服务器上限可以达到 10 万)的 http 代理服务器技术, 可以实现 http 请求的处理.

1 负载均衡

假如一个服务器能同时处理 500 个请求, 那么要达到 5000 并发请求就需要 10 个服务器, nginx 负责将同一域名同一端口的请求分发到不同的服务器上实现负载均衡



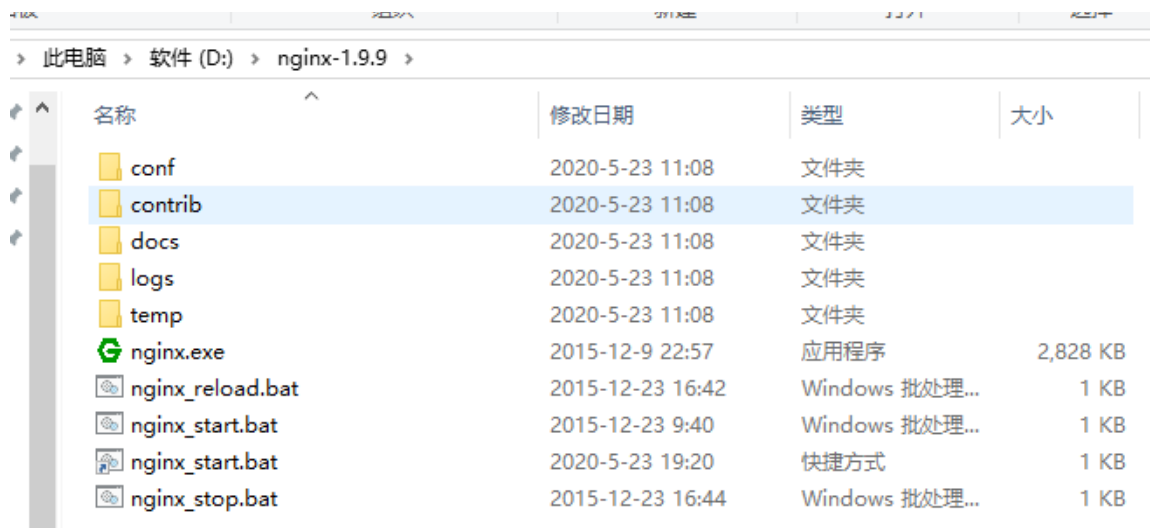
2 动静分离

所有静态文件, `.html`、`.js`、`.img` 等交给 nginx 管理, 客户端访问静态文件是非常迅速的, 动态资源(代码功能)交给后端服务器处理

二 Nginx 的安装

1 下载 Nginx 绿色版安装包

2 解压(无中文无空格路径)



名称	修改日期	类型	大小
conf	2020-5-23 11:08	文件夹	
contrib	2020-5-23 11:08	文件夹	
docs	2020-5-23 11:08	文件夹	
logs	2020-5-23 11:08	文件夹	
temp	2020-5-23 11:08	文件夹	
nginx.exe	2015-12-9 22:57	应用程序	2,828 KB
nginx_reload.bat	2015-12-23 16:42	Windows 批处理...	1 KB
nginx_start.bat	2015-12-23 9:40	Windows 批处理...	1 KB
nginx_start.bat	2020-5-23 19:20	快捷方式	1 KB
nginx_stop.bat	2015-12-23 16:44	Windows 批处理...	1 KB

3 目录介绍

conf:核心配置文件 nginx.conf 所在的目录, 和进程, 线程占用配置的配置文件所在目录

logs: access.log 通过 nginx 访问的记录, error.log 运行启动时的错误日志

nginx_start.bat: 启动 nginx 的 bat 文件

nginx_stop.bat: 停止 nginx 的 bat 文件

nginx_reload.bat: 重新加载配置文件

4 常用命令

验证配置是否正确: `nginx -t`

查看 Nginx 的版本号: `nginx -V`

启动 Nginx: `start nginx`

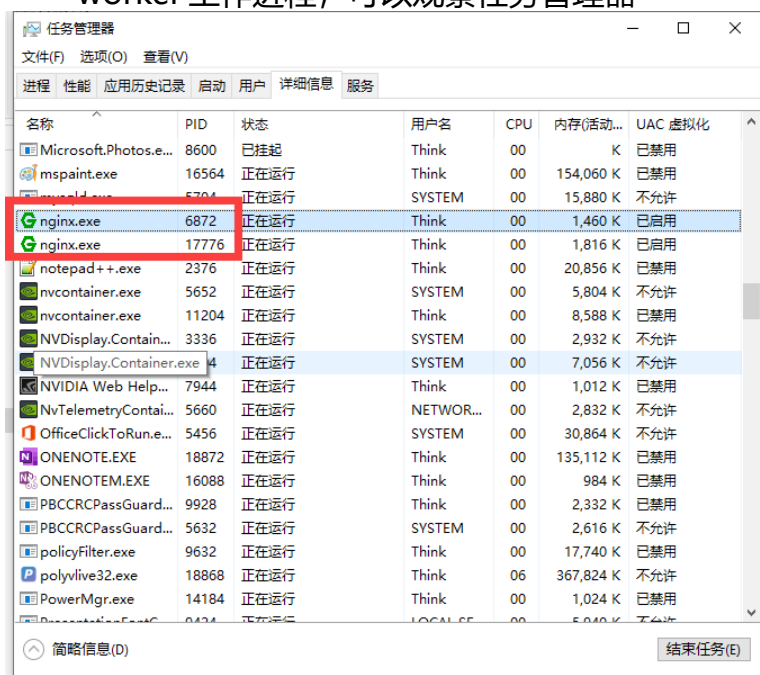
快速停止或关闭 Nginx: `nginx -s stop`

正常停止或关闭 Nginx: `nginx -s quit`

配置文件修改重载命令: `nginx -s reload`

- 检查 nginx 是否正常运行

- 默认配置中，nginx 需要启动 2 个进程，一个是 master 管理员，一个是 worker 工作进程，可以观察任务管理器



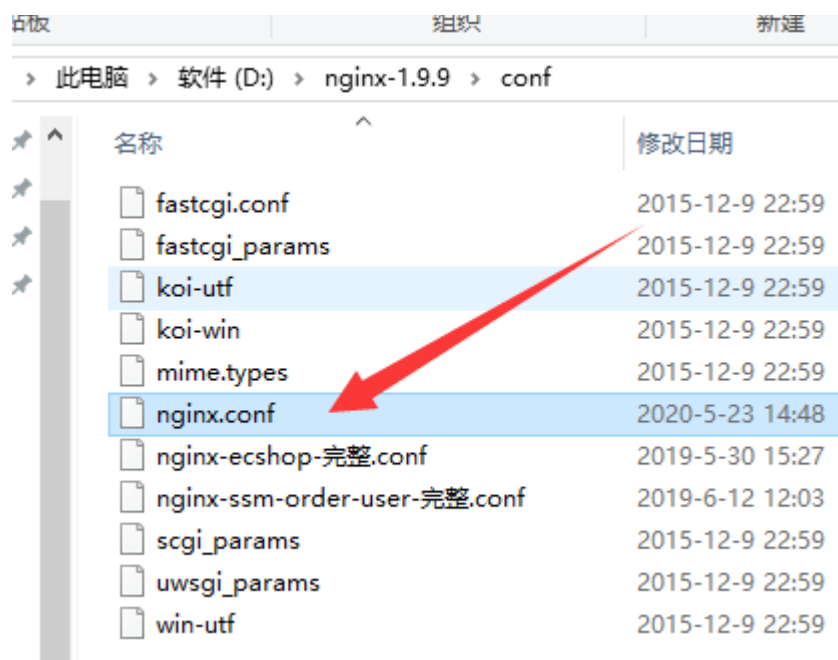
三 Nginx 入门案例

1 配置虚拟服务器

Nginx 作为代理，可以在核心配置文件中配置多个虚拟服务器(http 服务器) 接收到达 Nginx 服务器的 http 请求，然后根据配置逻辑实现监听判，，满足条件的请求将由有这个虚拟服务器处理。

在 `http{`
`}` 标签内配置

1.A 打开配置文件



1.B 配置服务器(server)

```
http {
    include    mime.types;
    default_type application/octet-stream;
    #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;
    sendfile    on;
    #tcp_nopush  on;
    #keepalive_timeout 0;
    keepalive_timeout 65;
    #gzip on;

    server {
        listen 80;
        server_name shuxs.nat300.top;
        location / {
            proxy_pass http://127.0.0.1:8080;
        }
    }
}
```

2 参数含义

2.A server:

在 http 的内容中可以配置多个 server，每个 server 都是一个可以接收 http 请求的虚拟服务器，和能接受

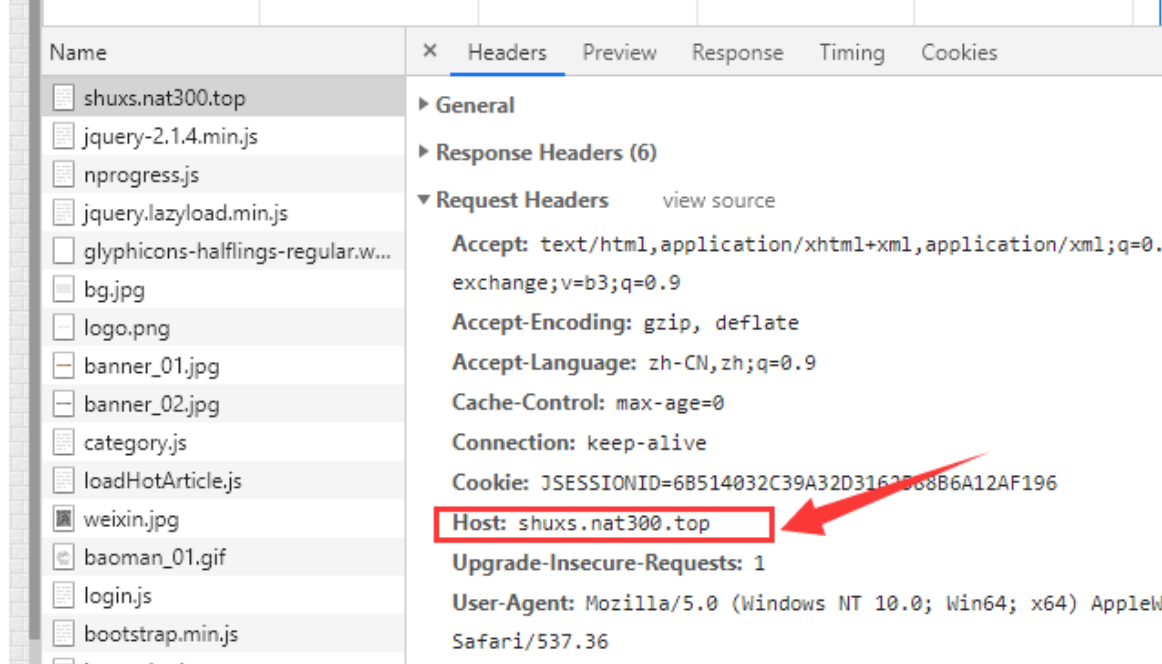
http 请求的真实服务器是一样

2.B listen

表示该 server 在 nginx 中运行时监听的 nginx 服务器上所有该端口的访问

2.C server_name:

listen 发现请求后，判断请求中的 **host 头(所以不能有 http://)**，匹配则进入该 server.不同的 2 个 server 配置中，监听端口和域名名称不能相同



2.D location /

/ 是一个通配符，后面有详细介绍
负责计算请求 uri 路径，判断和规则匹配与否，一旦匹配进入到 location 内部，不匹配则不处理.
www.ou.com/user/query/point location /表示所有路径以/开始的都能匹配上---统配表示方式，优先级最低

2.E proxy_pass

代理路径，一旦进入 location，可以通过这个关键字转向后端服务器(必须是带 http 的完整 URL 路径).

3 location 的匹配规则和优先级

在同一个 server 的内容里，可以同时存在多个 location，根据请求中携带 uri 地址结构不同值不同，分到不同 location 来出不同的请求，设计 location pattern 有哪些内容，多个 location 同时匹配成功时，

优先级如何判断.

3.A 匹配规则

精确匹配	location =/image	去掉域名和端口 uri 地址必须完全等于/image
有修饰的字符串前缀匹配	location ^~/image	uri 地址以/image 开始,可以表示多级
无修饰的字符串前缀匹配	location /image	uri 地址以 /image 开始,不能表示多级 /image/haha
正则匹配区分大小写	location ~.png\$	uri 以.png 结尾,区分大小写
正则匹配不区分大小写	location ~*.png\$	uri 地址以.png/.pNg/.pnG/等结尾
通配	location /	只要满足 server 中端口和域名都能匹配到这个/

3.B 优先级

- 上表中，优先级从上往下优先级越低
- 相互包含关系的优先级确定
 - 包含关系：例如 location ^~/image (以/image 开始) 包含 location =/image,
 - 有修饰的字符串前缀匹配如果有包含关系,以最大匹配长度的优先级最高
 - 正则表达式有包含关系,以配置在上面的正则优先级更高

3.C 练习(伪代码)

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

location /	{return 200}
location /image	{return 201}
location =/image	{return 202}
location ~^/image	{return 203}
location ~^/image/test	{return 204}
location ~.png\$	{return 205}
location ~.(png gif jpg)\$	{return 206}

}

http://www.nginx.com/user/query	200
http://www.nginx.com/image	202
http://www.nginx.com/image/test/haha	204
http://www.nginx.com/user/query.png	205

4 注意事项

4.A 分号结尾

4.B 参数与值空格分隔

4.C 满足匹配规则后，会将匹配的字符串截掉，后面的信息会加到转发的 URL 上

```
listen 80;
server_name www.ou.com;
location /user {
    proxy_pass http://127.0.0.1:8763/zuul-user/user;
    add_header 'Access-Control-Allow-Origin' '*';
    add_header 'Access-Control-Allow-Credentials' 'true';
}
```

假如用户访问 www.ou.com/user/test01

这是匹配，然后会将访问的域名去掉，location 的 `/user` 去掉，剩下 `/test01`

最后将 test01 拼接到 proxy_pass 后面： <http://127.0.0.1:8763/zuul-user/user/test01>

4.D 静态文件 url 拼接时不会去掉匹配的字符串

```
server {
    listen 80;
    server_name image.jt.com;
    location /upload {
        root img;
    }
}
```

访问地址：<http://image.jt.com/uplod/> 这是匹配到 `/upload`，然后将域名替换为相对路径 `img/upload/`

注意与 `proxy_pass` 的区别

5 访问流程

域名服务器已经建立 `shuxsh.nat300.top` 与本机 IP 地址的映射关系

浏览器访问 <http://shuxs.nat300.top/>，默认 80 端口，请求头携带 `host:shuxsh.nat300.top`

Nginx 监听到有请求

判断 `server_name` 与请求头 `host` 相当，并且端口相同

“ / ” location 与所有匹配

最后将请求转发到 `proxy_pass` 的 URL

四 负载均衡

在 nginx 可以通过虚拟的集群域名,来实现一个后端集群的统一管理的 list 列表,通过不同方式的负载均衡计算可以实现负载均衡的访问.

1 轮询配置

1.A 定义服务器列表(http 标签内)

```
#集群列表
upstream myserverlist{
    server 127.0.0.1:8080;
    server 127.0.0.1:8081;
    server 127.0.0.1:8082;
}
```

1.B 监听域名端口,转发到服务器

这种配置,在使用 `myserverlist` 这个自定义虚拟域名访问时,对 list 中数据 server 进行依次访问--轮询

在 server 中的 location 里 proxy_pass 不要在指向一个具体的服务器,指向这个虚拟域名 `myserverlist`

```
#集群列表
upstream myserverlist{
    server 127.0.0.1:8080;
    server 127.0.0.1:8081;
    server 127.0.0.1:8082;
}

server {
    listen 80;
    server_name shuxs.nat300.top;
    location / {
        proxy_pass http://myserverlist/;
    }
}
```


2 权重配置

轮询作为依次访问的逻辑,一种物理平均的负载均衡计算,如果后端运行服务器集群每个节点性能不同,应该给高性能的服务器分配更多的请求,给低性能的服务器分配少的请求,可以使用权重(权衡比重),例如:8091(2),8092(3),8093(5).

权重的配置,需要在轮训的 list 服务器列表 upstream 里配置 2 个关键字
weight=数字 >= 1 整数, 表示当前 server 分配的负载均衡比重

down: 当前 server 不在进行访问 比重为 0.有的时候一个集群中某些节点宕机之后无法短时间恢复,可以使用 down 禁止对其进行访问

```
#集群列表
upstream myserverlist{
    server 127.0.0.1:8080 weight=1;
    server 127.0.0.1:8081 down;
    server 127.0.0.1:8082 weight=5;
}
```

3 集群 session 共享问题(ip_hash)

SESSION 保存在服务器上, 当第一次访问产生 SESSION 后第二次访问负载均衡到另一台服务器, 此时 SESSION 丢失

3.A 场景:

用户在 SERVER1 登录成功在 SESSION 中保存相关数据, 然后转首页, Nginx 负载均衡到 SERVER2, 却提示未登录.

3.B 解决方法

3.B.a 集群共享 session:

- 通过集群节点之间的有效代码逻辑,使得每一个节点的所有 session 对象在其他所有节点都存在,SESSION 同步?
 - 缺点:造成每个服务器分配大量资源处理冗余 session 对象

3.B.b 代理 ip_hash 计算(nignx 支持):

- 只要客户端的 ip 地址不发生变化,对应负载均衡的计算就只能访问同一个节点
 - 缺点:将客户端强耦合到一个后端服务器上,一旦服务器出现问题不能访问,这些客户端将被动的暂停对系统使用
- 计算逻辑
 - 本质上是一种 hash 取余的计算逻辑

- 对客户端 ip 地址进行 hash 整数计算,得到 hash 值 ,假设 3 个负载均衡的节点,用 hash 值对 3 进行%取余计算,得到[0,1,2] 绑定 list 中下标, ip 不变绑定下标不变,对应 server 不会发生变化.

3.B.c ★ 采用第三方的存储容器(redis):

- session 不存储在服务器上,使用第三方存储容器管理 session 数据.
 - 缺点:占用更多的网络带宽资源.redis 稳定,性能高是个优秀的选择

五 动静分离

```
server {
    listen 80;
    server_name shuxs.nat300.top;
    #静态资源
    location ~*^(/css|docx|fonts|html|images|js|register|swf) {
        root stati/myblog;
    }
    #动态资源
    location / {
        proxy_pass http://myserverlist;
    }
}
```

root 可以使用绝对路径, 相当路径表示当前 nginx 的根目录
index index.html 指定首页

```
server {
    listen 80;
    server_name www.ou.com;
    location /user {
        proxy_pass http://127.0.0.1:8091/user;    }
    location /order {
        proxy_pass http://127.0.0.1:8092/order;
    }
    location /{
        root easymall;
        index 123.html;
    }
}
```

当访问 www.ou.com 则转到 123.html