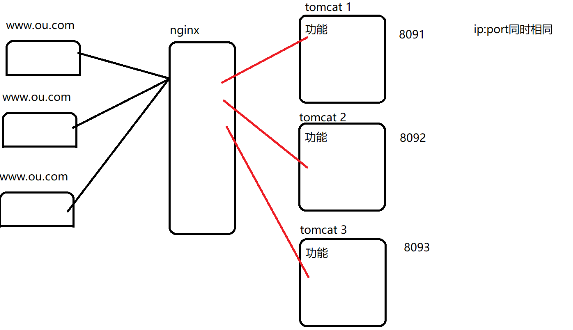
# Nginx简介

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.

### 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



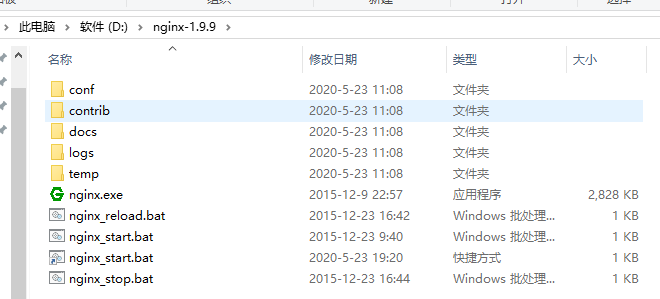
### 动静分离

所有静态文件，.html、.js、.img等交给nignx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

# Nginx的安装

### 下载Nginx绿色版安装包

### 解压(无中文无空格路径)



### 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

### 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

配置文件修改重装载命令：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

计算机生成了可选文字:
文仁〔F）远项〔0苷(V)
进程性能应甲历史记录启
名称
Microsoft.Photos.e
0
mspalnt.exe
ng•nx.exe
not巳d++X巳
过nvcontaner.exe
过nvcontaner.exe
过NVDisplay℃onta•nm
PID
85閬
16564
6872
2375
5652
11204
彐彐36
NVDisplay.Container.exe
0NVIDIA、V巳5Help..
过NvTelemetryContai...
刂OfficeClickToRun.e
《的0〕NENOTEM.EXE
PBCCRCPassGuardm
PBCCRCPassGuardm
policyFilter.exe
0四爿32艹
(EpowerMgr.exe
简峪启息(D)
7g44
5650
5456
18872
16088
"28
5632
g532
18858
14184
用户详继1一息
状态
已多起
正在运行
正在运行
正在运行
正在运行
正在运行
正在运行
正在运行
正在运行
正在运行
正在运行
正在运行
正在运行
正在运行
正在运行
正在运行
户名
Think
Think
SYSTEM
Think
Think
Think
SYSTEM
Think
SYSTEM
SYSTEM
Think
NETWORm
SYSTEM
Think
Think
Think
SYSTEM
Think
Think
Think
CPU
05
内存〔活一
154050K
1，880K
1460K
1816K
20856K
，804K
588K
2围32K
7囤56K
1012K
2。832K
30854K
135112K
g84K
2囤32K
216K
17。740K
367。824K
1囤24K
LIAC拟化
不允许
已启用
已禁
不允许
已禁
不允许
不允许
不允许
不允许
已禁
已禁
不允许
不允许
结束任务旧

# Nginx入门案例

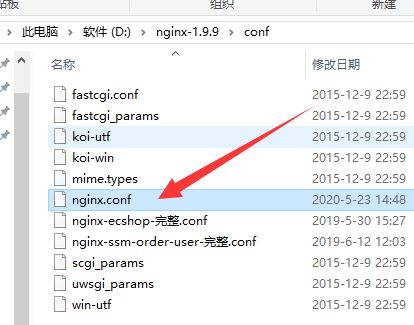
### 配置虚拟服务器

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

#### 打开配置文件



#### 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;  }  }  } |

### 参数含义

#### server:

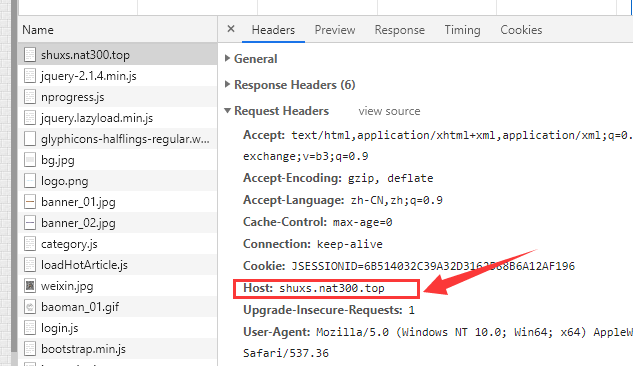
在http的内容中可以配置多个server，每个server都是一个可以接收http请求的虚拟服务器，和能接受http请求的真实服务器是一样

#### 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

#### 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



#### 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

#### 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).

### 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.

#### 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/ |

#### 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,以配置在上面的的正则优先级更高

#### 练习(伪代码)

server{

listen 80;

server\_name [www.nginx.com](http://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 |

### 注意事项

#### 分号结尾

#### 参数与值空格分隔

#### 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](http://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

#### 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

### 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](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

# 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.

### 轮询配置

#### 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;  } |

#### 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/;  }  } |

### 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;  } |

### 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

#### 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.

#### 解决方法

##### 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

##### 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.

##### 重要 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

# 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](http://www.ou.com)则转到123.html