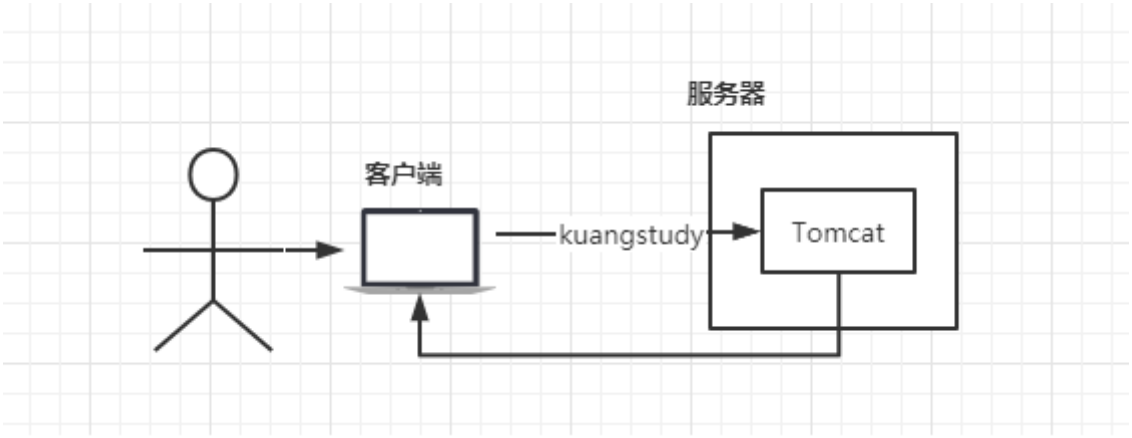
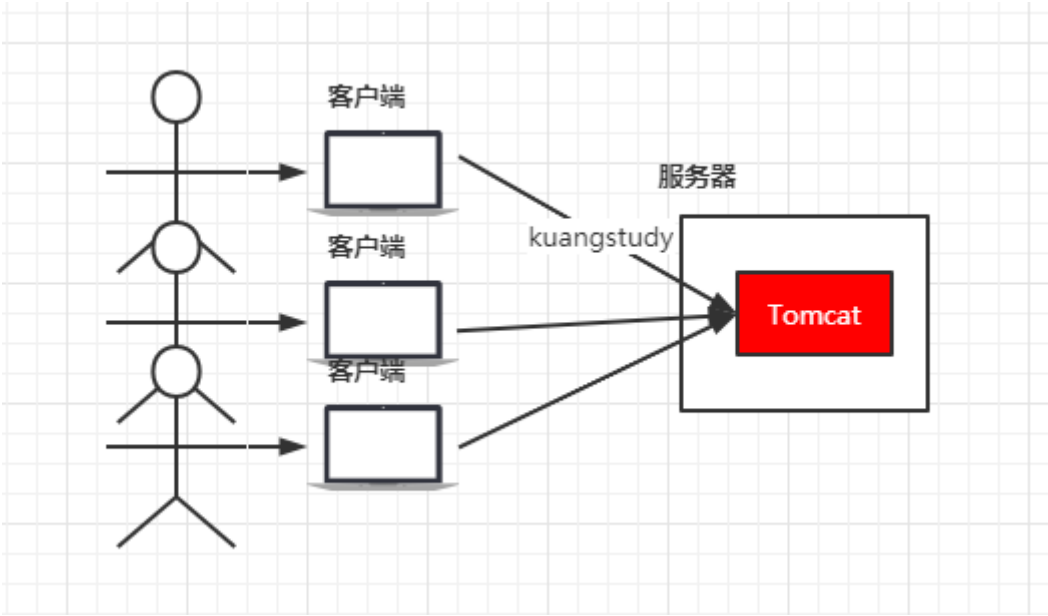
Nginx简介

公司产品出现瓶颈?

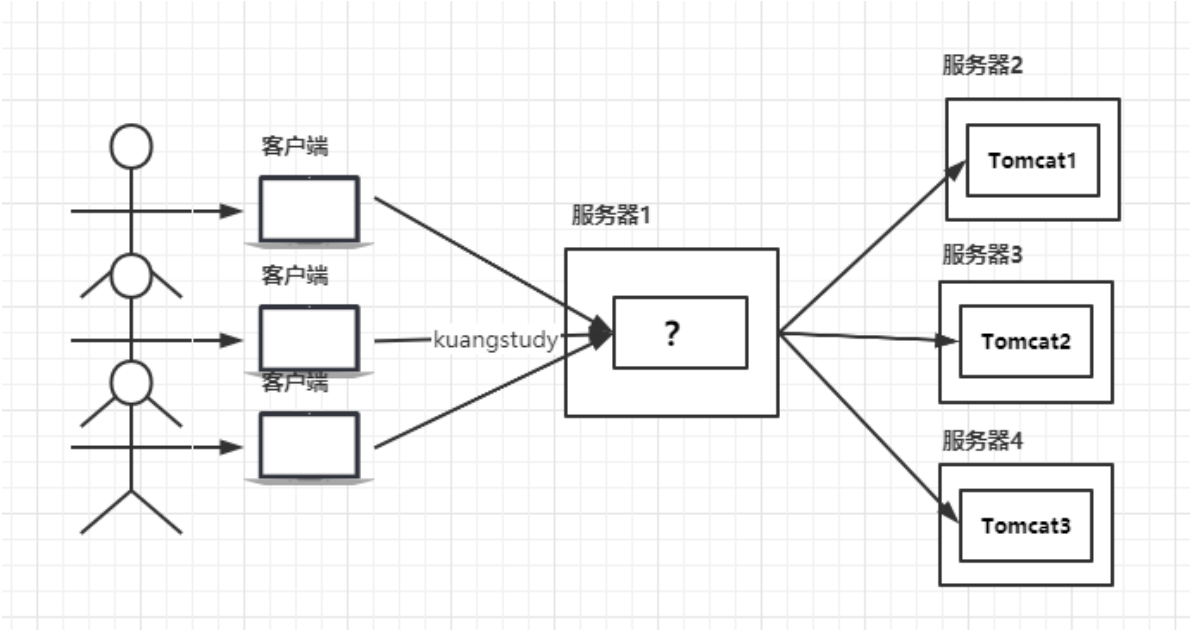
我们公司项目刚刚上线的时候，并发量小，用户使用的少，所以在低并发的情况下，一个jar包启动应用就够了，然后内部tomcat返回内容给用户。



但是慢慢的，使用我们平台的用户越来越多了，并发量慢慢增大了，这时候一台服务器满足不了我们的需求了。



于是我们横向扩展，又增加了服务器。这个时候几个项目启动在不同的服务器上，用户要访问，就需要增加一个代理服务器了，通过代理服务器来帮我们转发和处理请求。



我们希望这个代理服务器可以帮助我们接收用户的请求，然后将用户的请求按照规则帮我们转发到不同的服务器节点之上。这个过程用户是无感知的，用户并不知道是哪个服务器返回的结果，我们还希望他可以按照服务器的性能提供不同的权重选择。保证最佳体验！所以我们使用了Nginx。

什么是Nginx

Nginx (engine x) 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。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。在全球活跃的网站中有12.18%的使用比率，大约为2220万个网站。

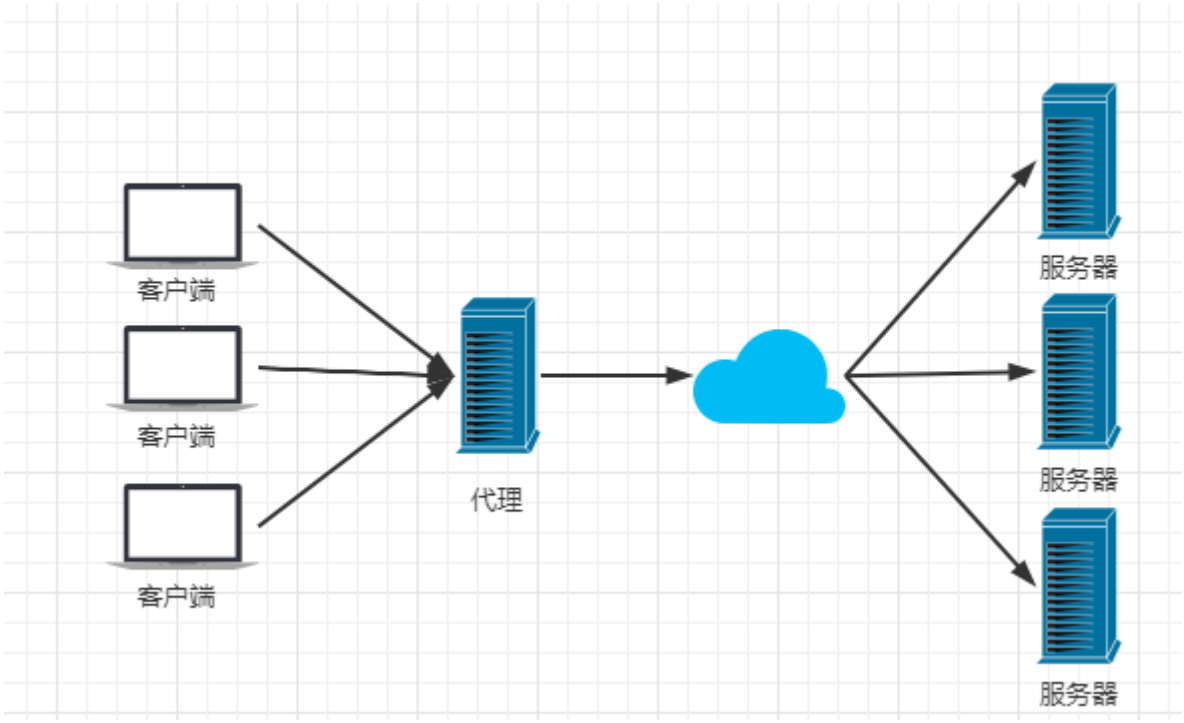
Nginx 是一个安装非常的简单、配置文件非常简洁（还能够支持perl语法）、Bug非常少的服务。Nginx 启动特别容易，并且几乎可以做到7*24不间断运行，即使运行数个月也不需要重新启动。你还能够不间断服务的情况下进行软件版本的升级。

Nginx代码完全用C语言从头写成。官方数据测试表明能够支持高达 50,000 个并发连接数的响应。

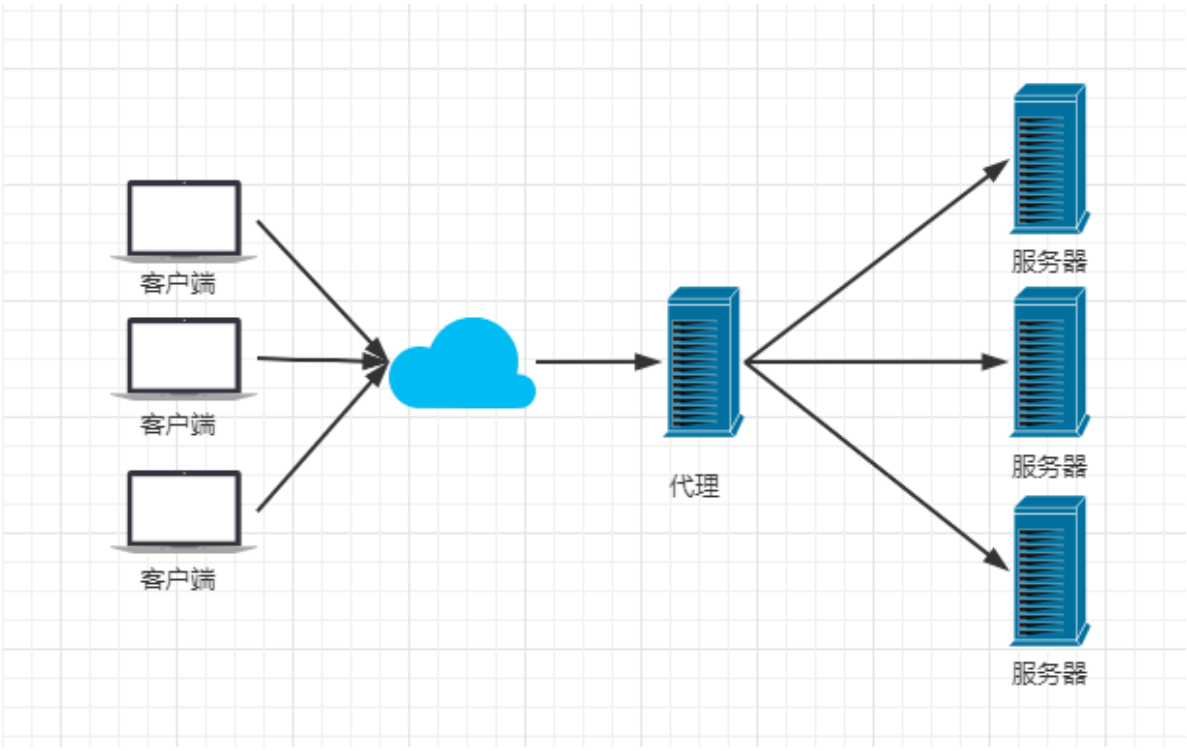
Nginx作用?

> Http代理，**反向代理**：作为web服务器最常用的功能之一，尤其是反向代理。

正向代理

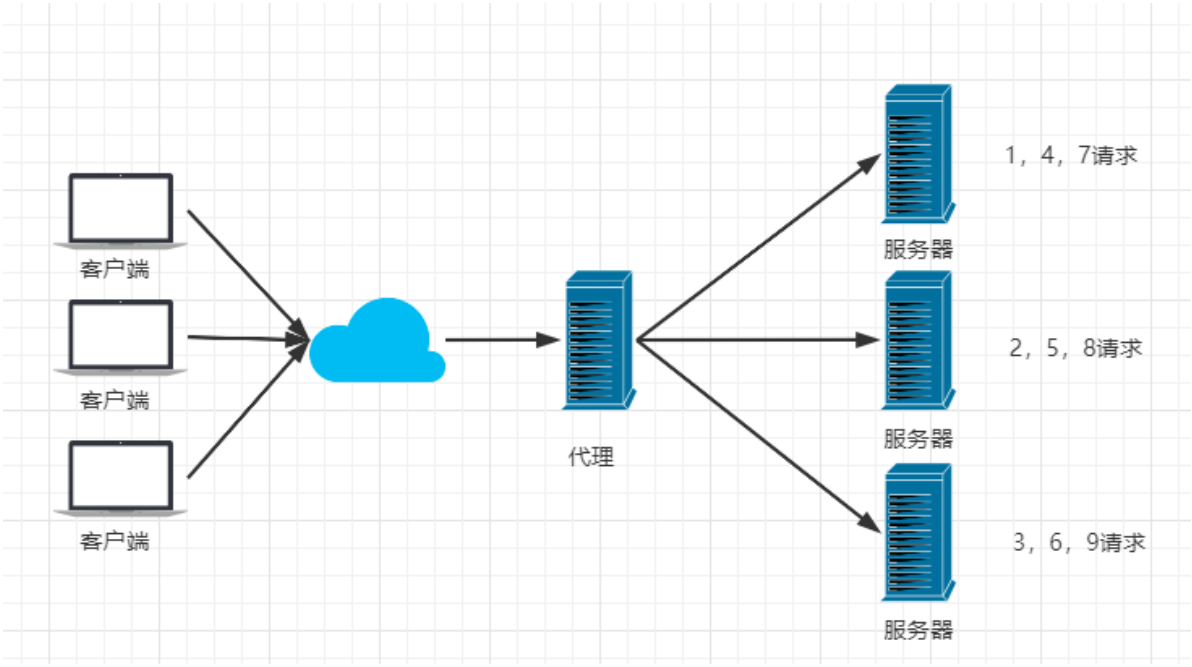


反向代理

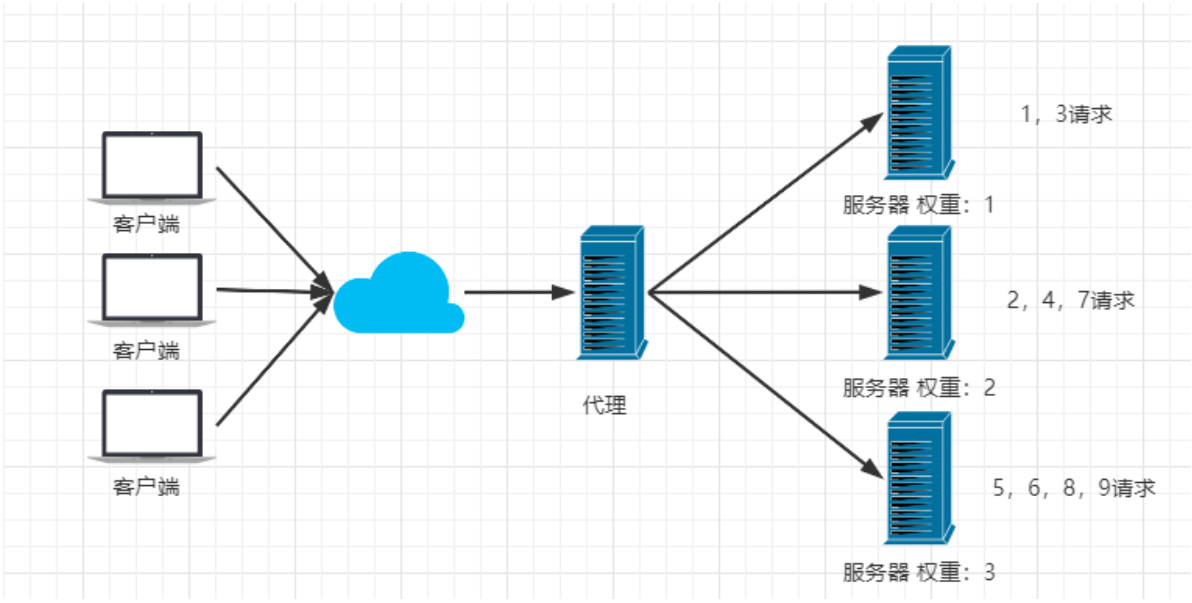


> Nginx提供的**负载均衡**策略有2种：内置策略和扩展策略。内置策略为轮询，加权轮询，Ip hash。扩展策略，就天马行空，只有你想不到的没有他做不到的。

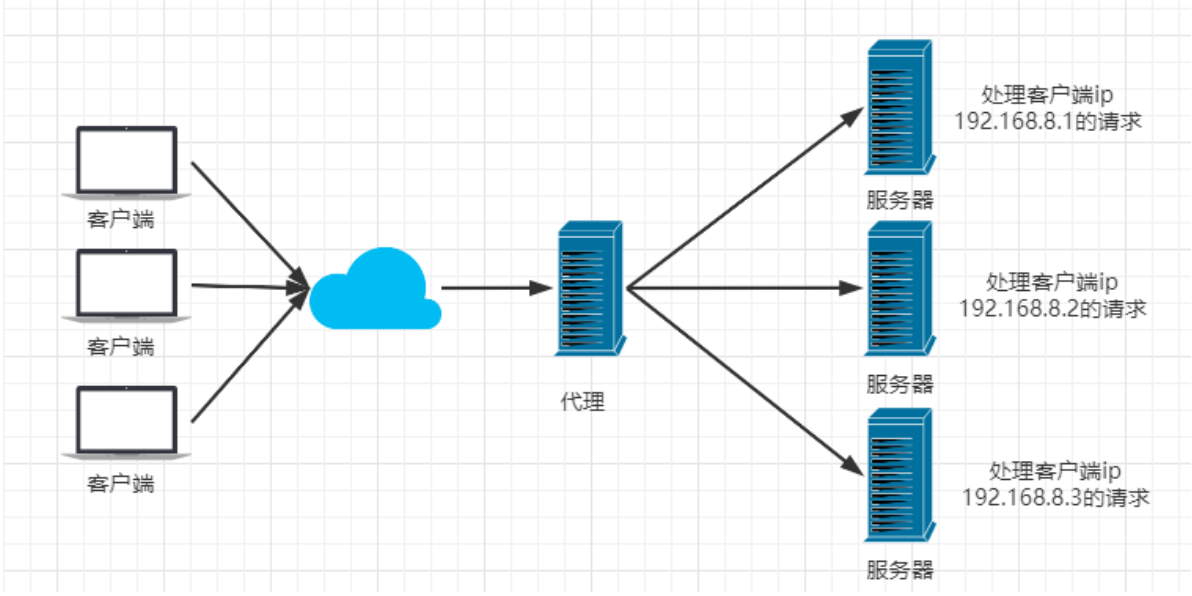
轮询



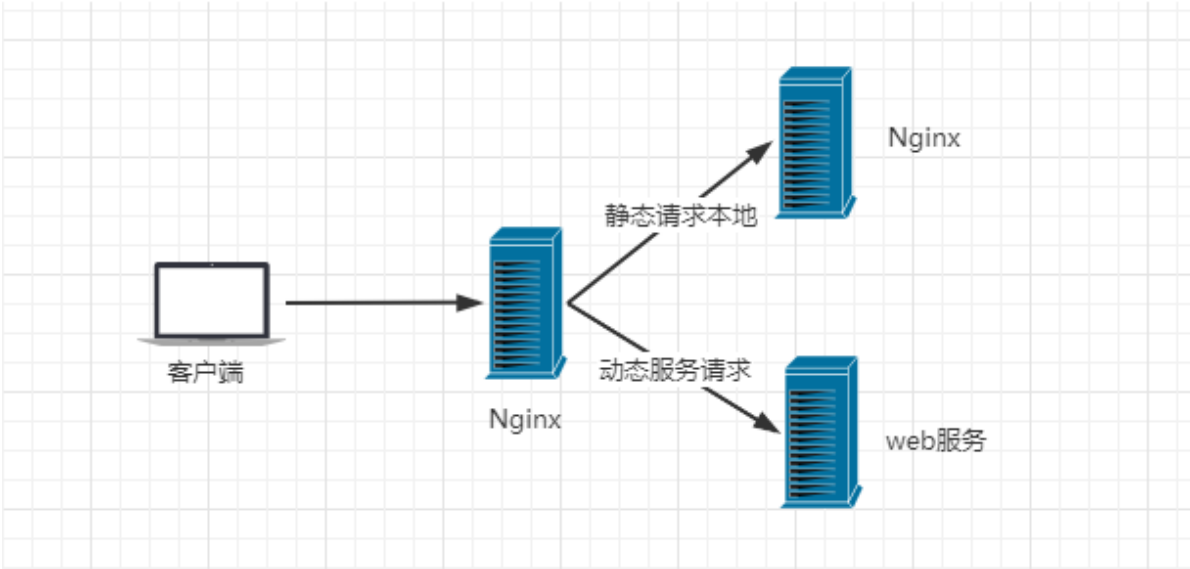
加权轮询



iphash对客户端请求的ip进行hash操作，然后根据hash结果将同一个客户端ip的请求分发给同一台服务器进行处理，可以解决session不共享的问题。



> **动静分离**，在我们的软件开发中，有些请求是需要后台处理的，有些请求是不需要经过后台处理的（如：css、html、jpg、js等等文件），这些不需要经过后台处理的文件称为静态文件。让动态网站里的动态网页根据一定规则把不变的资源和经常变的资源区分开来，动静资源做好了拆分以后，我们就可以根据静态资源的特点将其做缓存操作。提高资源响应的速度。



目前，通过使用Nginx大大提高了我们网站的响应速度，优化了用户体验，让网站的健壮性更上一层楼！

Nginx安装

Windows下安装

1、下载nginx

http://nginx.org/en/download.html 下载稳定版本。
下载后解压

2、启动nginx

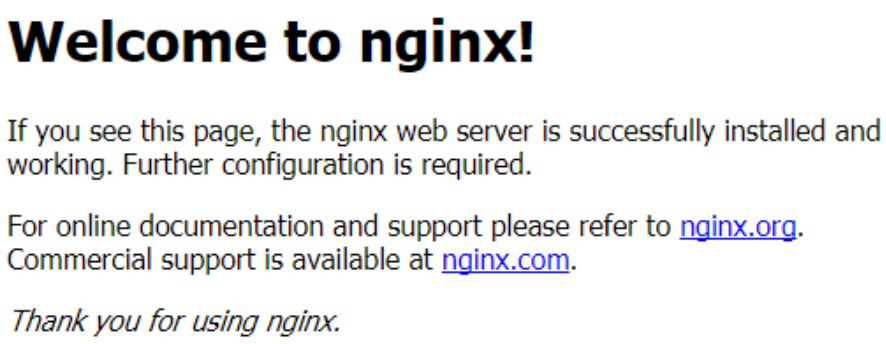
有很多种方法启动nginx

(1)直接双击nginx.exe，双击后一个黑色的弹窗一闪而过

(2)打开cmd命令窗口，切换到nginx解压目录下，输入命令 `nginx.exe` ，回车即可

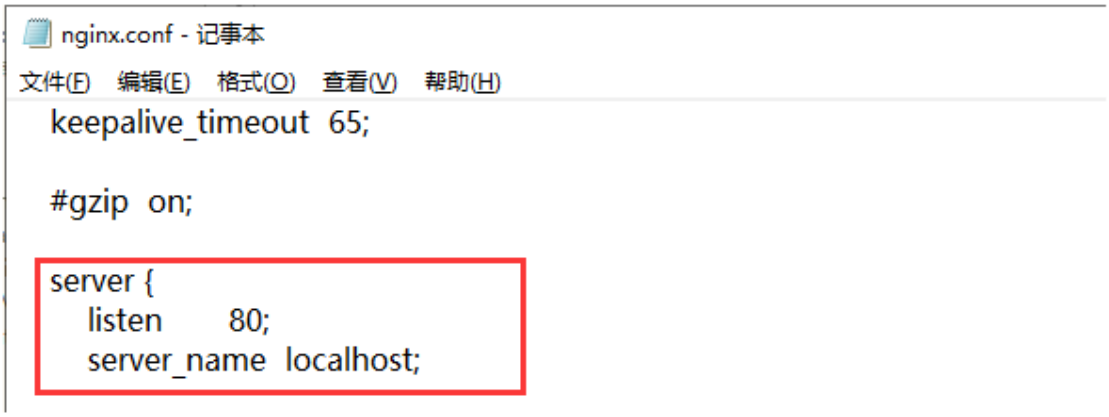
3、检查nginx是否启动成功

直接在浏览器地址栏输入网址 http://localhost:80 回车，出现以下页面说明启动成功！



4、配置监听

nginx的配置文件是conf目录下的nginx.conf，默认配置的nginx监听的端口为80，如果80端口被占用可以修改为未被占用的端口即可。



当我们修改了nginx的配置文件nginx.conf 时，不需要关闭nginx后重新启动nginx，只需要执行命令 `nginx -s reload` 即可让改动生效

5、关闭nginx

如果使用cmd命令窗口启动nginx， 关闭cmd窗口是不能结束nginx进程的，可使用两种方法关闭nginx

(1)输入nginx命令 `nginx -s stop`(快速停止nginx) 或 `nginx -s quit`(完整有序的停止nginx)

(2)使用taskkill `taskkill /f /t /im nginx.exe`

```
taskkill是用来终止进程的,
/f是强制终止 .
/t终止指定的进程和任何由此启动的子进程。
/im示指定的进程名称 .
```

Linux下安装

1、安装gcc

安装 nginx 需要先将官网下载的源码进行编译，编译依赖 gcc 环境，如果没有 gcc 环境，则需要安装:

```
yum install gcc-c++
```

2、PCRE pcre-devel 安装

PCRE(Perl Compatible Regular Expressions) 是一个Perl库，包括 perl 兼容的正则表达式库。nginx 的 http 模块使用 pcre 来解析正则表达式，所以需要在 linux 上安装 pcre 库，pcre-devel 是使用 pcre 开发的一个二次开发库。nginx也需要此库。命令:

```
yum install -y pcre pcre-devel
```

3、zlib 安装

zlib 库提供了很多种压缩和解压缩的方式， nginx 使用 zlib 对 http 包的内容进行 gzip ，所以需要在 Centos 上安装 zlib 库。

```
yum install -y zlib zlib-devel
```

4、OpenSSL 安装

OpenSSL 是一个强大的安全套接字层密码库，囊括主要的密码算法、常用的密钥和证书封装管理功能及 SSL 协议，并提供丰富的应用程序供测试或其它目的使用。

nginx 不仅支持 http 协议，还支持 https（即在ssl协议上传输http），所以需要在 Centos 安装 OpenSSL 库。

```
yum install -y openssl openssl-devel
```

5、下载安装包

手动下载.tar.gz安装包，地址：https://nginx.org/en/download.html

6、解压

```
tar -zxvf nginx-1.18.0.tar.gz
cd nginx-1.18.0
```

7、配置

使用默认配置，在nginx根目录下执行

```
./configure
make
make install
```

查找安装路径：`whereis nginx`

Docker安装

```
docker run \
    --name nginx \
    -p 80:80 \
    -v /home/zpb/data/nginx/nginx.conf:/etc/nginx/nginx.conf:ro \
    -v /home/zpb/data/nginx/html:/usr/share/nginx/html:ro \
    -d \
    nginx:stable
```

default.conf

```
server {
    listen       80;
    listen  [::]:80;
    server_name  localhost;

    #access_log  /var/log/nginx/host.access.log  main;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }

    #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   /usr/share/nginx/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
    #
    #location ~ \.php$ {
    #    root           html;
    #    fastcgi_pass   127.0.0.1:9000;
    #    fastcgi_index  index.php;
    #    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
    #    include        fastcgi_params;
    #}

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

Nginx常用命令

```
cd /usr/local/nginx/sbin/
./nginx             # 启动
./nginx -s stop     # 停止
./nginx -s quit     # 安全退出
./nginx -s reload   # 重新加载配置文件
ps aux|grep nginx   # 查看nginx进程
```

相关命令:

```
# 开启
service firewalld start
# 重启
service firewalld restart
# 关闭
service firewalld stop
# 查看防火墙规则
firewall-cmd --list-all
# 查询端口是否开放
firewall-cmd --query-port=8080/tcp
# 开放80端口
firewall-cmd --permanent --add-port=80/tcp
# 移除端口
firewall-cmd --permanent --remove-port=8080/tcp
#重启防火墙(修改配置后要重启防火墙)
firewall-cmd --reload
# 参数解释
1、firwall-cmd：是Linux提供的操作firewall的一个工具;
2、--permanent：表示设置为持久;
3、--add-port：标识添加的端口;
```

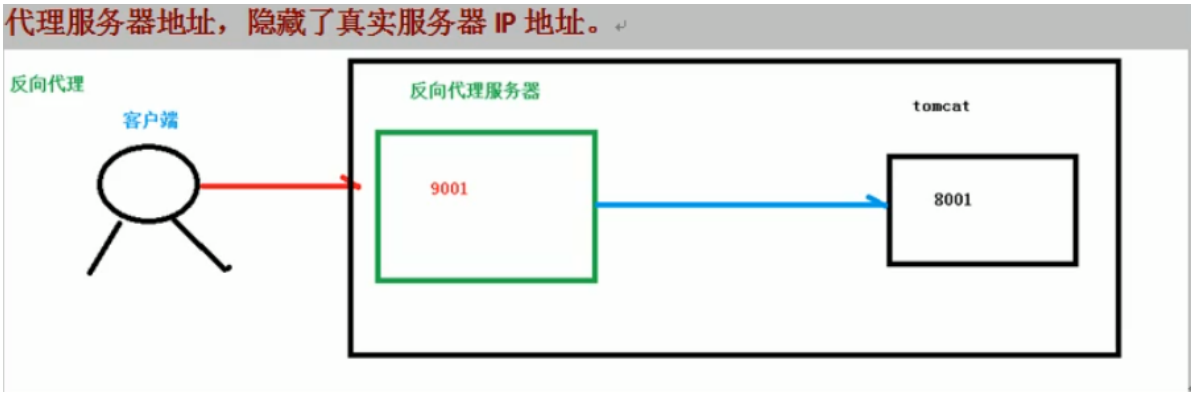
Nginx配置

nginx.conf

- 全局块
- event块
- http块
    - server

Nginx核心

反向代理



基本

```
http {
    ...
    server {
        listen       80;
        server_name  192.168.0.101;

        location / {
            root         /usr/share/nginx/html;
            proxy_pass   http://192.168.0.101:8090;
            index        index.html index.htm;
        }
    }
}
```

路由转发

```
http {
    server {
        listen 80;
        server_name  192.168.0.101;

        location ~ /edu/ {
            proxy_pass   http://192.168.0.101:8090;
        }

        location ~ /vod/ {
            proxy_pass   http://192.168.0.101:8091;
        }
    }
}
```

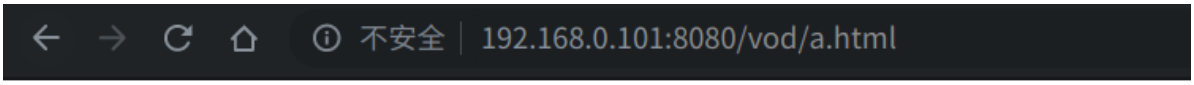
1、= ：用于不含正则表达式的 uri 前，要求请求字符串与 uri 严格匹配，如果匹配成功，就停止继续向下搜索并立即处理该请求。

2、~：用于表示 uri 包含正则表达式，并且区分大小写。

3、~*：用于表示 uri 包含正则表达式，并且不区分大小写。

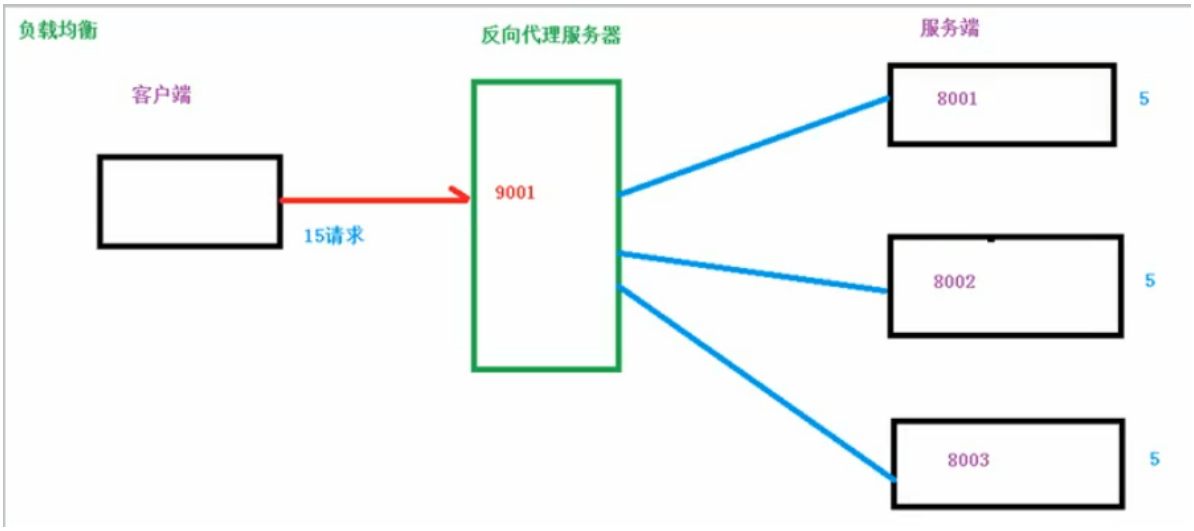
4、^~：用于不含正则表达式的 uri 前，要求 Nginx 服务器找到标识 uri 和请求字符串匹配度最高的 location 后，立即使用此 location 处理请求，而不再使用 location 块中的正则 uri 和请求字符串做匹配。

注意：如果 uri 包含正则表达式，则必须要有 ~ 或者 ~* 标识。



vod

负载均衡



基本

```
http {
    upstream myserver {
        #ip_hash;
        server 192.168.0.101:8090 weight=1;
        server 192.168.0.101:8091 weight=1;
    }
    server {
        location / {
            proxy_pass http://myserver;
            #proxy_connect_timeout 10;
        }
    }
}
```

策略

- 轮询（默认）

每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器 down 掉，能自动剔除。

- 第二种 weight

weight 代表权重默认为 1,权重越高被分配的客户端越多

```
upstream server_pool {
    server 192.168.5.21 weight=1;
    server 192.168.5.22 weight=1;
}
```

- 第三种 ip_hash

每个请求按访问 ip 的 hash 结果分配，这样每个访客固定访问一个后端服务器。

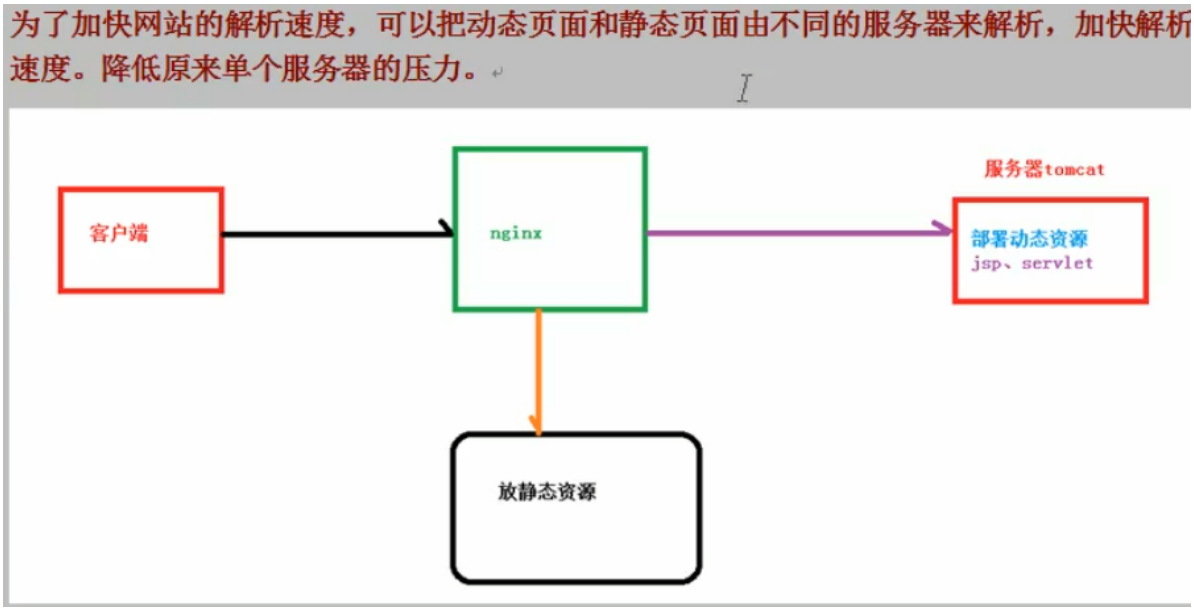
```
upstream server_pool {
    ip_hash;
    server 192.168.5.21;
    server 192.168.5.22;
}
```

- fair（第三方）

按后端服务器的响应时间来分配请求，响应时间短的优先分配。

```
upstream server_pool {
    server 192.168.5.21;
    server 192.168.5.22;
    fair;
}
```

动静分离



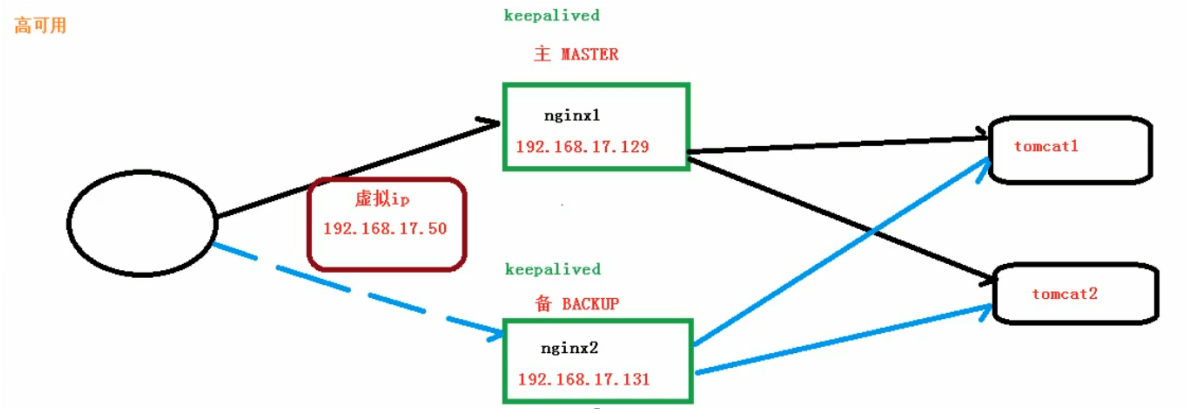
基本

```
http {
    server {
        listen 80;
        server_name 192.168.0.102;

        location /www/ {
            root /usr/share/nginx/html/;
        }

        location /images/ {
            root /usr/share/nginx/html/;
            autoindex on; #列出目录
        }
    }
}
```

高可用



> keepalived