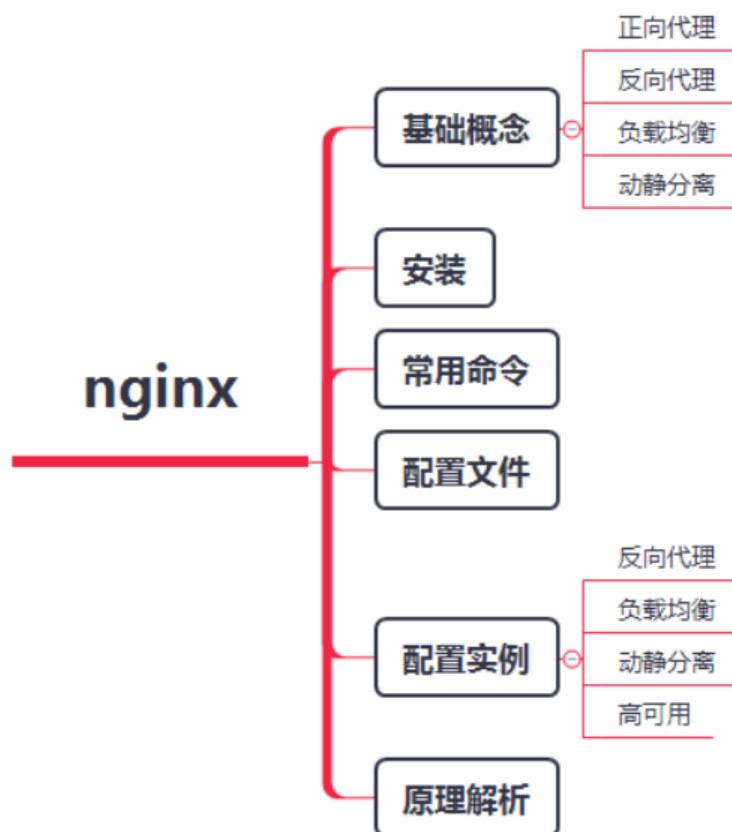


NGINX

Nginx知识网结构图



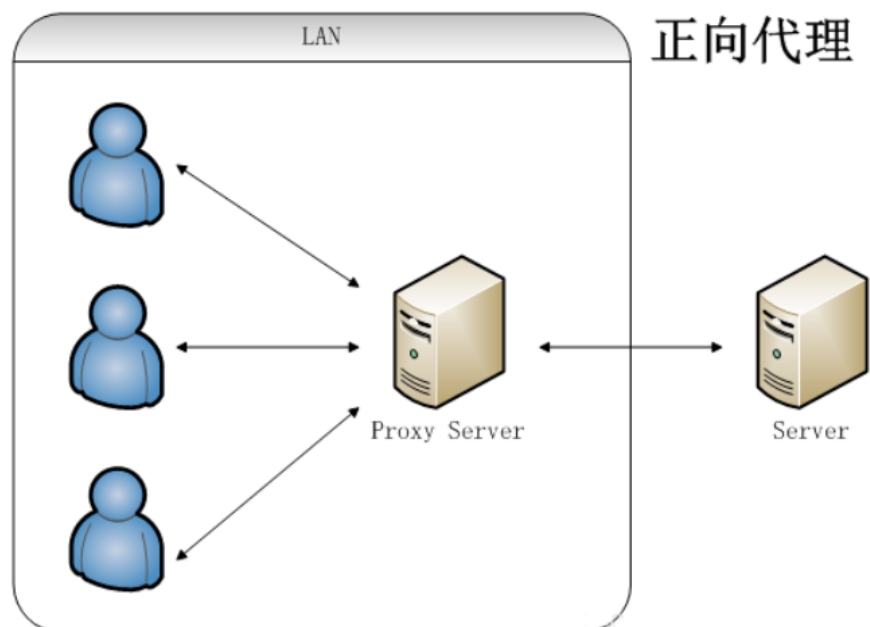
Nginx是一个高性能的HTTP和反向代理服务器，特点是占用内存少，并发能力强，事实上nginx的并发能力确实在同类型的网页服务器中表现较好。

Nginx专为性能优化而开发，性能是其最重要的要求，十分注重效率，有报告Nginx能支持高达50000个并发连接数。

基础概念

正向代理

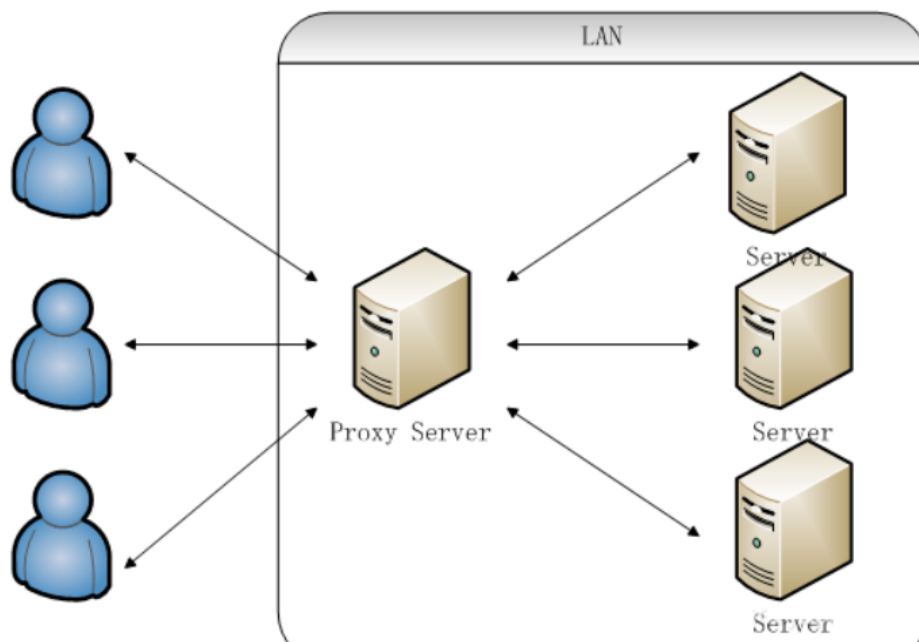
局域网中的电脑用户想要直接访问网络是不可行的，只能通过代理服务器来访问，这种代理服务就被称为正向代理。



反向代理

客户端无法感知代理，因为客户端访问网络不需要配置，只要把请求发送到反向代理服务器，由反向代理服务器去选择目标服务器获取数据，然后再返回到客户端，此时反向代理服务器和目标服务器对外就是一个服务器，暴露的是代理服务器地址，隐藏了真实服务器IP地址。

反向代理



负载均衡

客户端发送多个请求到服务器，服务器处理请求，有一些可能要与数据库进行狡猾，服务器处理完毕之后，再将结果返回给客户端。

普通请求和响应过程：

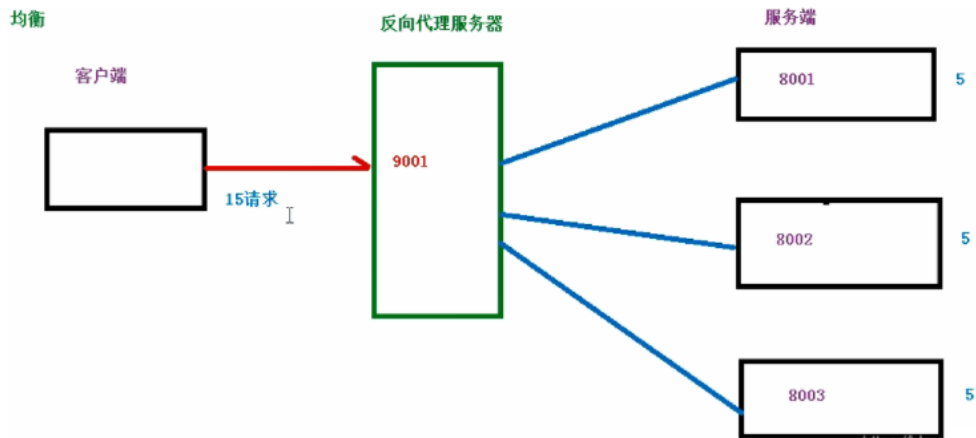


但是随着信息数量增长，访问量和数据量飞速增长，普通架构无法满足现在的需求。

我们首先想到的是升级服务器配置，可以由于摩尔定律的日益失效，单纯从硬件提升性能已经逐渐不可取了，怎么解决这种需求呢？

我们可以增加服务器的数量，构建集群，将请求分发到各个服务器上，将原来请求集中到单个服务器的情况改为请求分发到多个服务器，也就是我们说的负载均衡。

图解负载均衡：

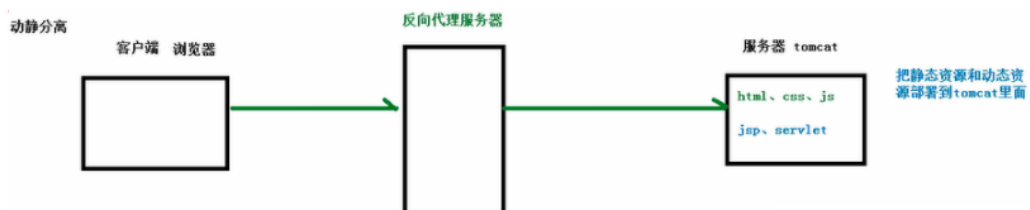


假设有15个请求发送到代理服务器，那么由代理服务器根据服务器数量，平均分配，每个服务器处理5个请求，这个过程就叫做负载均衡。

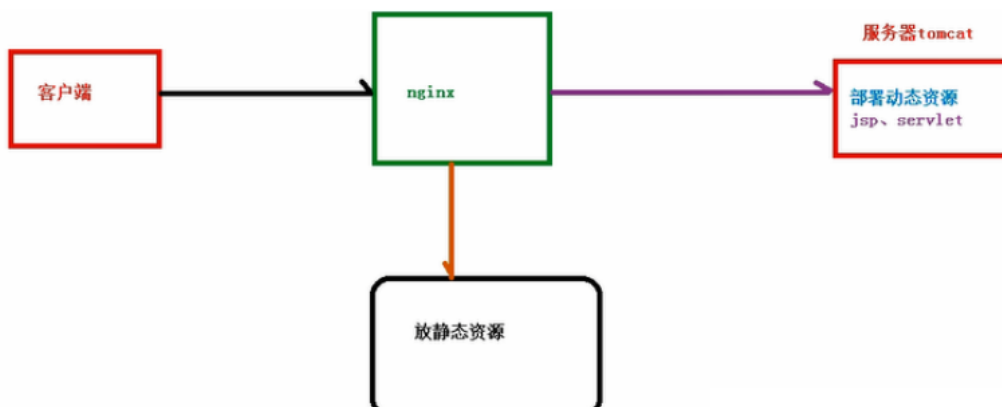
动静分离

为了加快网站的解析速度，可以把动态页面和静态页面交给不同的服务器来解析，加快解析的速度，降低由单个服务器的压力。

动静分离之前的状态L:



动静分离之后:





Nginx如何在linux安装

第一种：Linux系统

CentOS 7 64位

下载以下安装包，用Xftp放入Linux系统。

 nginx-1.12.2.tar.gz	2019/8/27 10:41	360压缩	959 KB
 pcre-8.37.tar.gz	2019/8/27 10:41	360压缩	1,994 KB

第一步：安装pcre依赖

解压压缩文件，进入解压之后的目录执行./configure，然后执行make && make install。

```
ln -sf pcre_get_substring_list.3 /usr/local/share/man/man3/pcre32_get_substring_list.3
ln -sf pcre_jit_exec.3 /usr/local/share/man/man3/pcre32_jit_exec.3
ln -sf pcre_jit_stack_alloc.3 /usr/local/share/man/man3/pcre32_jit_stack_alloc.3
ln -sf pcre_jit_stack_free.3 /usr/local/share/man/man3/pcre32_jit_stack_free.3
ln -sf pcre_maketables.3 /usr/local/share/man/man3/pcre32_maketables.3
ln -sf pcre_pattern_to_host_byte_order.3 /usr/local/share/man/man3/pcre32_pattern_to_host_byte_order.3
ln -sf pcre_refcount.3 /usr/local/share/man/man3/pcre32_refcount.3
ln -sf pcre_study.3 /usr/local/share/man/man3/pcre32_study.3
ln -sf pcre_utf32_to_host_byte_order.3 /usr/local/share/man/man3/pcre32_utf32_to_host_byte_order.3
ln -sf pcre_version.3 /usr/local/share/man/man3/pcre32_version.3
make[3]: 离开目录"/root/pcre-8.37"
make[2]: 离开目录"/root/pcre-8.37"
make[1]: 离开目录"/root/pcre-8.37"
```

查看是否安装成功：

```
[root@localhost pcre-8.37]# pcre-config --version
```

```
[root@localhost pcre-8.37]# pcre-config --version
```

```
[root@localhost pcre-8.37]# pcre-config --version
8.37
```

第二步：安装其他依赖

```
[root@localhost pcre-8.37]# yum -y make zlib zlib-devel gcc-c++ libtool openssl
openssl-devel
```

```

已安装:
  libtool.x86_64 0:2.4.2-22.el7_3

作为依赖被安装:
  autoconf.noarch 0:2.69-11.el7      automake.noarch 0:1.13.4-3.el7 m4.x86_64 0:1.4.16-10.el7 perl-Test-Harness.noarch 0:3.28-3.el7
  perl-Thread-Queue.noarch 0:3.02-2.el7

更新完毕:
  make.x86_64 1:3.82-24.el7

完毕!
[root@localhost ~]# cd

```

第三步：安装Nginx

解压Nginx，进入Nginx目录，执行./configure：

```

+ using system zlib library

nginx path prefix: "/usr/local/nginx"
nginx binary file: "/usr/local/nginx/sbin/nginx"
nginx modules path: "/usr/local/nginx/modules"
nginx configuration prefix: "/usr/local/nginx/conf"
nginx configuration file: "/usr/local/nginx/conf/nginx.conf"
nginx pid file: "/usr/local/nginx/logs/nginx.pid"
nginx error log file: "/usr/local/nginx/logs/error.log"
nginx http access log file: "/usr/local/nginx/logs/access.log"
nginx http client request body temporary files: "client_body_temp"
nginx http proxy temporary files: "proxy_temp"
nginx http fastcgi temporary files: "fastcgi_temp"
nginx http uwsgi temporary files: "uwsgi_temp"
nginx http scgi temporary files: "scgi_temp"

```

<https://blog.csdn.net/yujing1314>

执行make && make install：

```

cp conf/nginx.conf '/usr/local/nginx/conf/nginx.conf.default'
test -d '/usr/local/nginx/logs' \
|| mkdir -p '/usr/local/nginx/logs'
test -d '/usr/local/nginx/logs' \
|| mkdir -p '/usr/local/nginx/logs'
test -d '/usr/local/nginx/html' \
|| cp -R html '/usr/local/nginx'
test -d '/usr/local/nginx/logs' \
|| mkdir -p '/usr/local/nginx/logs'
make[1]: 离开目录"/root/nginx-1.12.2"

```

去sbin文件夹下启动Nginx。

```
cd /usr/local/nginx/sbin
```

```
[root@localhost sbin]# ./nginx
```

检查是否启动成功。

```
[root@localhost sbin]# ps -ef|grep nginx
```

```
[root@localhost sbin]# ps -ef|grep nginx
root      27213      1  0 15:51 ?        00:00:00 nginx: master process ./nginx
nobody    27214    27213  0 15:51 ?        00:00:00 nginx: worker process
root      27298    12200  0 15:59 pts/1    00:00:00 grep --color=auto nginx
```

第二种

Linux系统

Red Hat Enterprise Linux Server release 6.5 (Santiago)

提前需要准备的：

Nginx源码：<http://nginx.org/en/download.html>

yum安装教程：<https://blog.csdn.net/yujing1314/article/details/97237644>

gcc-c++：

```
[root@localhost ~]# yum install gcc-c++
```

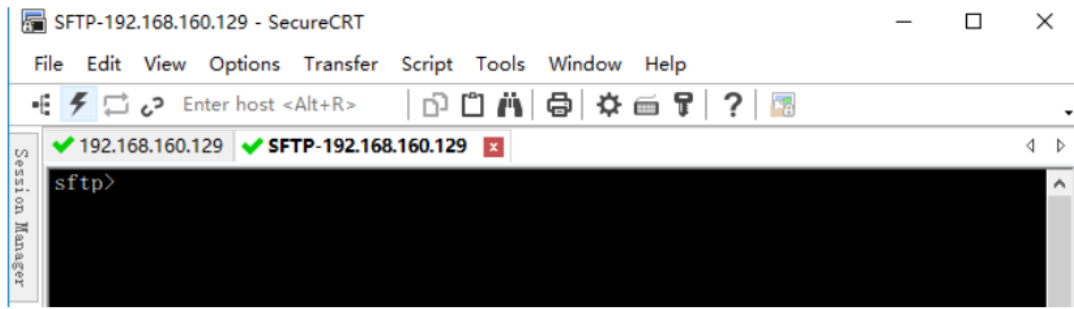
第三方开发包：

```
[root@localhost ~]# yum install -y pcre pcre-devel
[root@localhost ~]# yum install -y zlib zlib-devel
[root@localhost ~]# yum install -y openssl openssl-devel
```

安装步骤：

第一步：把Nginx的源码包上传到Linux系统。

我使用的SecureCRT的sftp文件传输，直接把文件拖进去就OK了。



第二步：解压缩

```
[root@localhost ~]tar zxf nginx-1.8.0.tar.gz
```

第三步：使用configure命令创建—makeFile文件

```
./configure
-prefix=/usr/local/nginx
-pid-path=/var/run/nginx/nginx.pid
-lock-path=/var/lock/nginx.lock
-error-log-path=/var/log/nginx/error.log
-http-log-path=/var/log/nginx/access.log
-with-http_gzip_static_module
-http-client-body-temp-path=/var/temp/nginx/client
-http-proxy-temp-path=/var/temp/nginx/proxy
-http-fastcgi-temp-path=/var/temp/nginx/fastcgi
-http-uwsgi-temp-path=/var/temp/nginx/uwsgi
-http-scgi-temp-path=/var/temp/nginx/scgi
```

第四步：上一步可能会报错，因为缺少temp文件，如下创建即可

```
[root@localhost sbin]# mkdir /var/temp/nginx/client -p
```

第五步：make

直接输入make。

第六步：make install

直接输入make install。

开启Nginx：

```
[root@localhost sbin]# ./nginx
```

如何查看进程[root@bogon stefan]# ps aux|grep nginx:


```
root@bogon stefan]# ps aux|grep nginx
root      44531  0.0  0.0  24316   792 ?        Ss   Jul24   0:00 nginx: master process ./nginx
nobody    44532  0.0  0.0  24744  1616 ?        S    Jul24   0:00 nginx: worker process
root      45072  0.0  0.0  103260   852 pts/1    S+   00:59   0:00 grep nginx
root@bogon stefan]#
```

关闭Nginx:

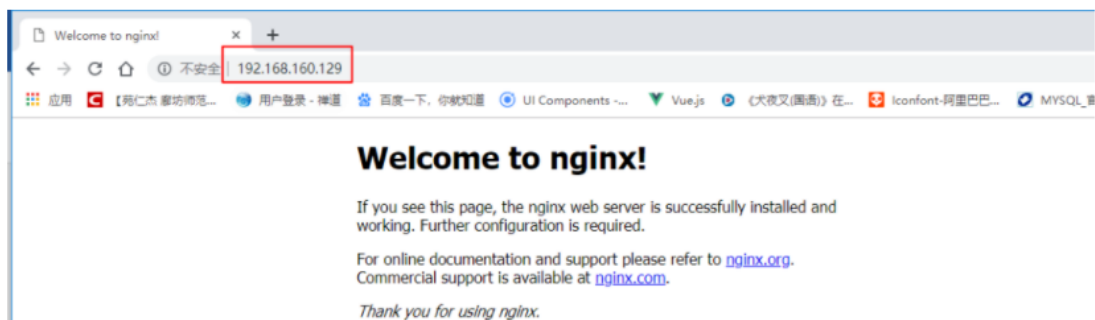
```
[root@localhost sbin]# ./nginx -s stop
```

推荐使用:

```
[root@localhost sbin]# ./nginx -s quit
```

测试

输入你虚拟机的IP，如下图就成功了。



如果测试失败，注意查看虚拟机防火墙是否关闭

Nginx常用命令

查看版本:

```
./nginx -v
```

启动:

```
./nginx
```

关闭（有两种方式，推荐使用 `./nginx -s quit`）：

```
./nginx -s stop  
./nginx -s quit
```

重新加载Nginx配置：

```
./nginx -s reload
```

Nginx的配置文件

配置文件分三部分组成。

1、全局块

从配置文件开始到events块之间，主要是设置一些影响nginx服务器整体运行的配置指令。

并发处理服务的配置，值越大，可以支持的并发处理量越多，但是会受到硬件、软件等设备的制约。

```
1  
2 #user  nobody;  
3 worker_processes 1;  
4
```

2、events块

影响nginx服务器与用户的网络连接，常用的设置包括是否开启对多workprocess下的网络连接进行序列化，是否允许同时接收多个网络连接等等。

支持的最大连接数：

```
12 events {  
13     worker_connections 1024;  
14 }
```

3、http块

诸如反向代理和负载均衡都在此配置。

location指令说明：

该语法用来匹配url，语法如下：

```
location[ = | ~ | ~* | ^~ ] url {  
  
}
```

- =：用于不含正则表达式的url前，要求字符串与url严格匹配，匹配成功就停止向下搜索并处理请求
- ~：用于表示url包含正则表达式，并且区分大小写
- ~*：用于表示url包含正则表达式，并且不区分大小写
- ^~：用于不含正则表达式的url前，要求Ngin服务器找到表示url和字符串匹配度最高的location后，立即使用此location处理请求，而不再匹配
- 如果有url包含正则表达式，不需要有~开头标识

反向代理实战

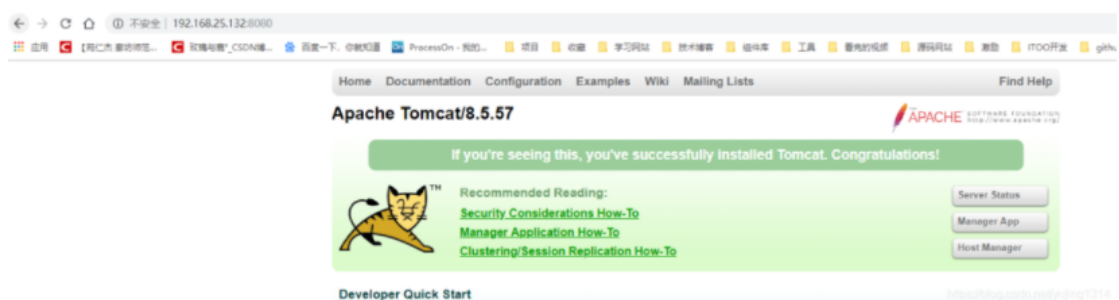
配置反向代理

目的：在浏览器地址栏输入地址www.123.com跳转Linux系统Tomcat主页面。

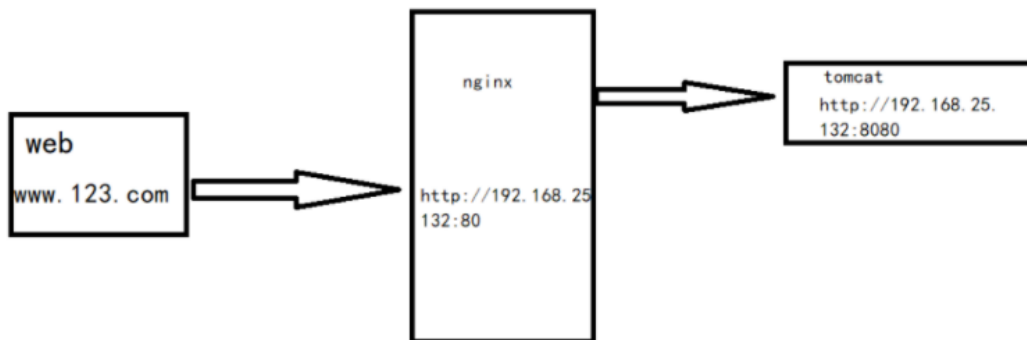
具体实现：

先配置Tomcat：因为比较简单，此处不再赘叙。

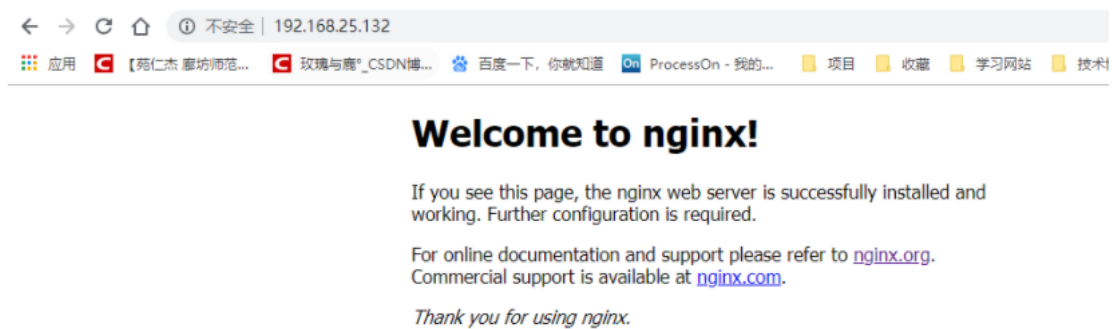
并在Windows访问：



具体流程：



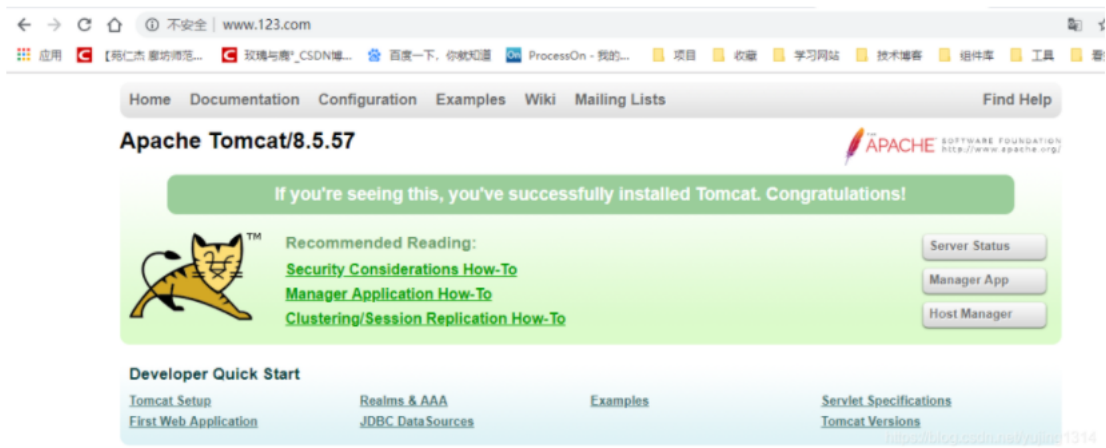
修改之前:



配置:

```
server {  
    listen      80;  
    server_name 192.168.25.132;  
  
    #charset koi8-r;  
  
    #access_log logs/host.access.log main;  
  
    location / {  
        root    html;  
        proxy_pass http://127.0.0.1:8080;  
        index   index.html index.htm;  
    }  
}
```

再次访问:



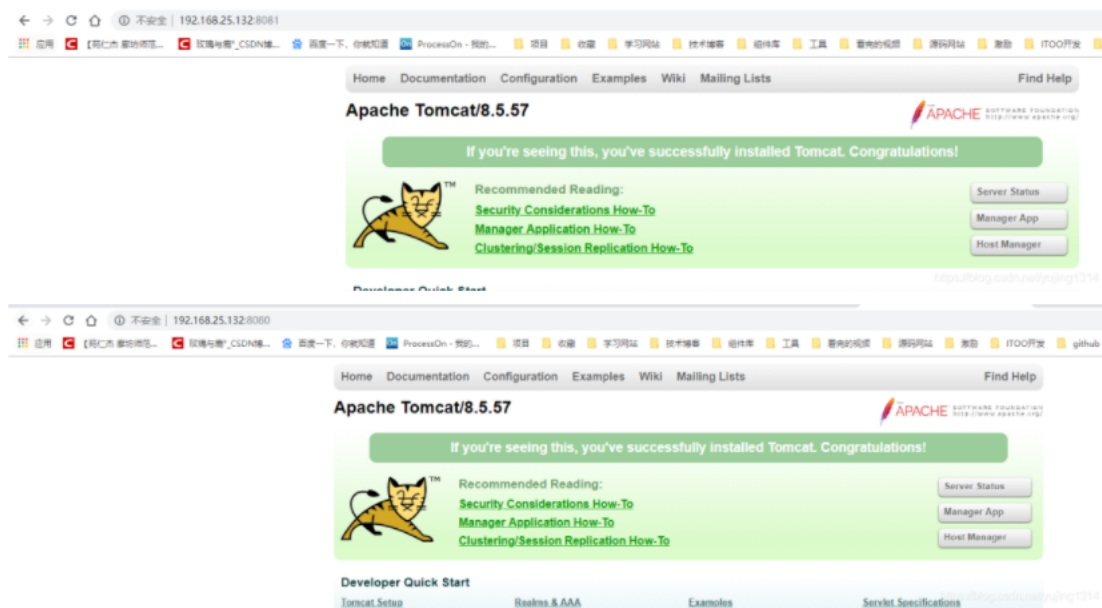
反向代理2

目标:

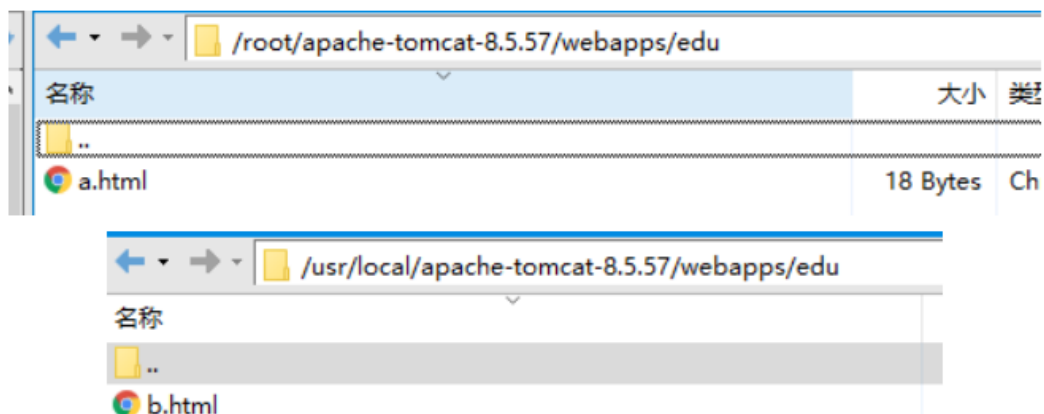
- 访问<http://192.168.25.132:9001/edu/>直接跳转到192.168.25.132:8080
- 访问<http://192.168.25.132:9001/vod/>直接跳转到192.168.25.132:8081

准备:

配置两个Tomcat, 端口分别为8080和8081, 都可以访问, 端口修改配置文件即可。



新建文件内容分别添加8080!!! 和8081!!!



响应如下:



8080!!!!!!



8081!!!!!!

具体配置:

```

35     server {
36         listen      9001;
37         server_name 192.168.25.132;
38
39         #charset koi8-r;
40
41         #access_log  logs/host.access.log  main;
42
43         location /edu/ {
44             root      html;
45             proxy_pass http://127.0.0.1:8080;
46         }
47         location /vod/ {
48             root      html;
49             proxy_pass http://127.0.0.1:8081;
50         }

```

重新加载Nginx:

```
./nginx -s reload
```

访问:



实现了同一个端口代理, 通过edu和vod路径的切换显示不同的页面。

反向代理小结

第一个例子: 浏览器访问www.123.com, 由host文件解析出服务器IP地址192.168.25.132 www.123.com, 然后默认访问80端口, 而通过Nginx监听80端口代理到本地的8080端口上, 从而实现了访问www.123.com, 最终转发到tomcat 8080上去。

第二个例子：访问<http://192.168.25.132:9001/edu/>直接跳转到192.168.25.132:8080，访问<http://192.168.25.132:9001/vod/>直接跳转到192.168.25.132:8081，实际上就是通过Nginx监听9001端口，然后通过正则表达式选择转发到8080还是8081的Tomcat上去。

负载均衡实战

修改nginx.conf:

```
34     upstream myserver{
35         server 192.168.25.132:8080;
36         server 192.168.25.132:8081;
37     }

80 server {
81     listen      80;
82     server_name 192.168.25.132;
83
84     #charset koi8-r;
85
86     #access_log logs/host.access.log main;
87
88     location / {
89         proxy_pass http://myserver;
90         root      html;
91         index     index.html index.htm;
92     }
```

重启Nginx:

```
./nginx -s reload
```

在8081的tomcat的webapps文件夹下新建edu文件夹和a.html文件，填写内容为8081！！！！

在地址栏回车，就会分发到不同的Tomcat服务器上。



负载均衡方式:

1、轮询（默认）

2、weight，代表权，权越高优先级越高

```
34     upstream myserver{
35         server 192.168.25.132:8080 weight=1;
36         server 192.168.25.132:8081 weight=2;
37     }
```

3、fair，按后端服务器的响应时间来分配请求，相应时间短的优先分配

```
34     upstream myserver{
35         server 192.168.25.132:8080;
36         server 192.168.25.132:8081;
37         fair;
38     }
```

4、ip_hash，每个请求按照访问IP的hash结果分配，这样每一个访客固定的访问一个后端服务器，可以解决session 的问题

```
34     upstream myserver{
35         ip_hash;
36         server 192.168.25.132:8080;
37         server 192.168.25.132:8081;
38     }
39
```

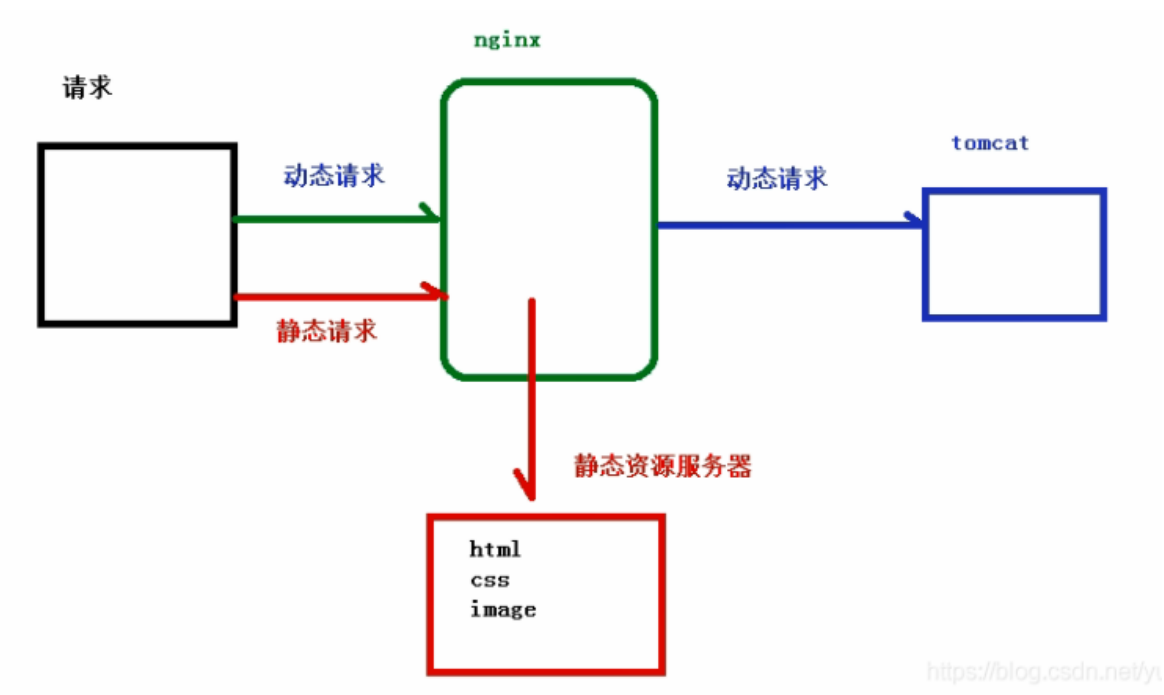
动静分离实战

什么是动静分离

把动态请求和静态请求分开，不是讲动态页面和静态页面物理分离，可以理解为Nginx处理静态页面，Tomcat处理动态页面。

动静分离大致分为两种：一、纯粹将静态文件独立成单独域名放在独立的服务器上，也是目前主流方案；二、将动态跟静态文件混合在一起发布，通过Nginx分开。

动静分离图析：



实战准备

准备静态文件：



配置Nginx：

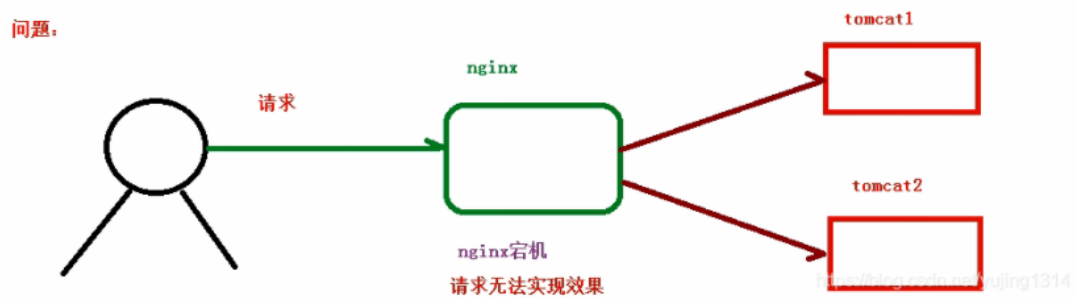
```

34     server {
35         listen      80;
36         server_name 192.168.25.132;
37
38         #charset koi8-r;
39
40         #access_log logs/host.access.log main;
41
42         location /www/ {
43             root    /data/;
44             index   index.html index.htm;
45         }
46         location /image/ {
47             root    /data/; root后面的路径就是从/根目录开始的
48             autoindex on;
49         }

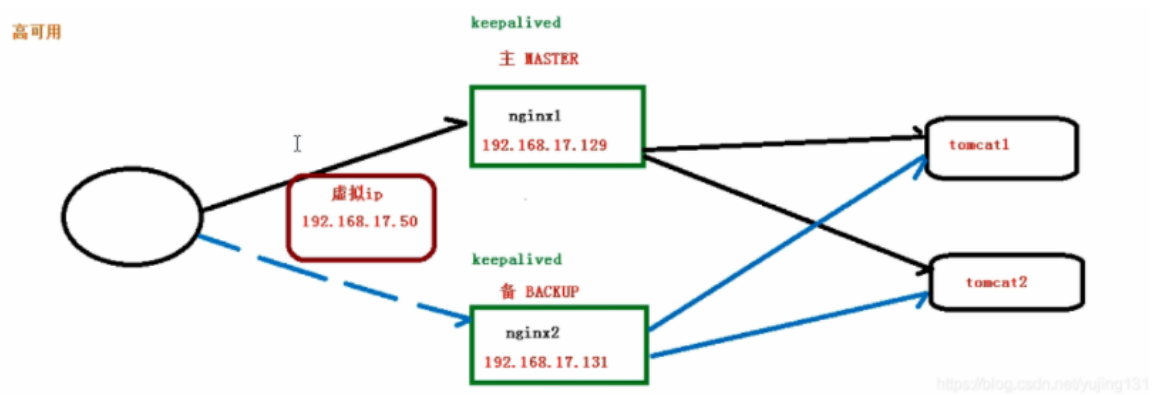
```

Nginx高可用

如果Nginx出现问题：



解决办法：



前期准备：

- 两台Nginx服务器
- 安装Keepalived
- 虚拟IP

安装Keepalived

```
[root@192 usr]# yum install keepalived -y
[root@192 usr]# rpm -q -a keepalived
keepalived-1.3.5-16.el7.x86_64
```

修改配置文件:

```
[root@192 keepalived]# cd /etc/keepalived
[root@192 keepalived]# vi keepalived.conf
```

分别将如下配置文件复制粘贴, 覆盖掉keepalived.conf。

虚拟IP为192.168.25.50。

对应主机IP需要修改的是:

```
smtp_server 192.168.25.147 (主) smtp_server 192.168.25.147 (备)
state MASTER (主) state BACKUP (备)
```

```
global_defs {
    notification_email {
        acassen@firewall.loc
        failover@firewall.loc
        sysadmin@firewall.loc
    }
    notification_email_from Alexandre.Cassen@firewall.loc
    smtp_server 192.168.25.147
    smtp_connect_timeout 30
    router_id LVS_DEVEL # 访问的主机地址
}

vrrp_script chk_nginx {
    script "/usr/local/src/nginx_check.sh" # 检测文件的地址
    interval 2 # 检测脚本执行的间隔
    weight 2 # 权重
}

vrrp_instance VI_1 {
    state BACKUP # 主机MASTER、备机BACKUP
    interface ens33 # 网卡
    virtual_router_id 51 # 同一组需一致
    priority 90 # 访问优先级, 主机值较大, 备机较小
```

```

advert_int 1
authentication {
    auth_type PASS
    auth_pass 1111
}
virtual_ipaddress {
    192.168.25.50 # 虚拟ip
}
}

```

启动:

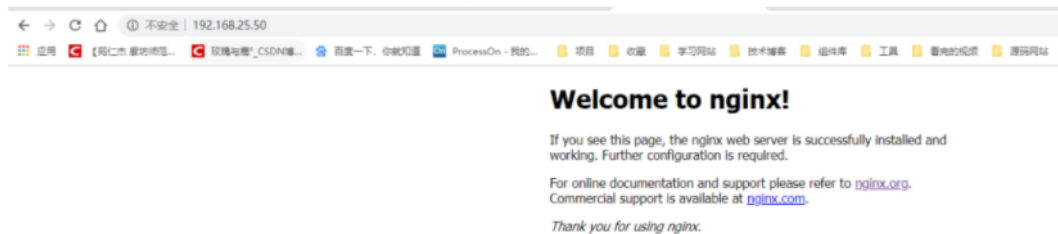
```
[root@192 sbin]# systemctl start keepalived.service
```

```

[root@192 sbin]# ps -ef | grep nginx
root      14532      1  0 07:47 ?        00:00:00 nginx: master process ./nginx
nobody    14533    14532  0 07:47 ?        00:00:00 nginx: worker process
root      27762    15124  0 08:53 pts/0    00:00:00 grep --color=auto nginx
[root@192 sbin]# ps -ef | grep keepalived
root      27764    15124  0 08:53 pts/0    00:00:00 grep --color=auto keepalived
[root@192 sbin]# systemctl start keepalived.service
[root@192 sbin]# ps -ef | grep keepalived
root      27772      1  0 08:53 ?        00:00:00 /usr/sbin/keepalived -D
root      27773    27772  0 08:53 ?        00:00:00 /usr/sbin/keepalived -D
root      27774    27772  0 08:53 ?        00:00:00 /usr/sbin/keepalived -D
root      27795    15124  0 08:54 pts/0    00:00:00 grep --color=auto keepalived

```

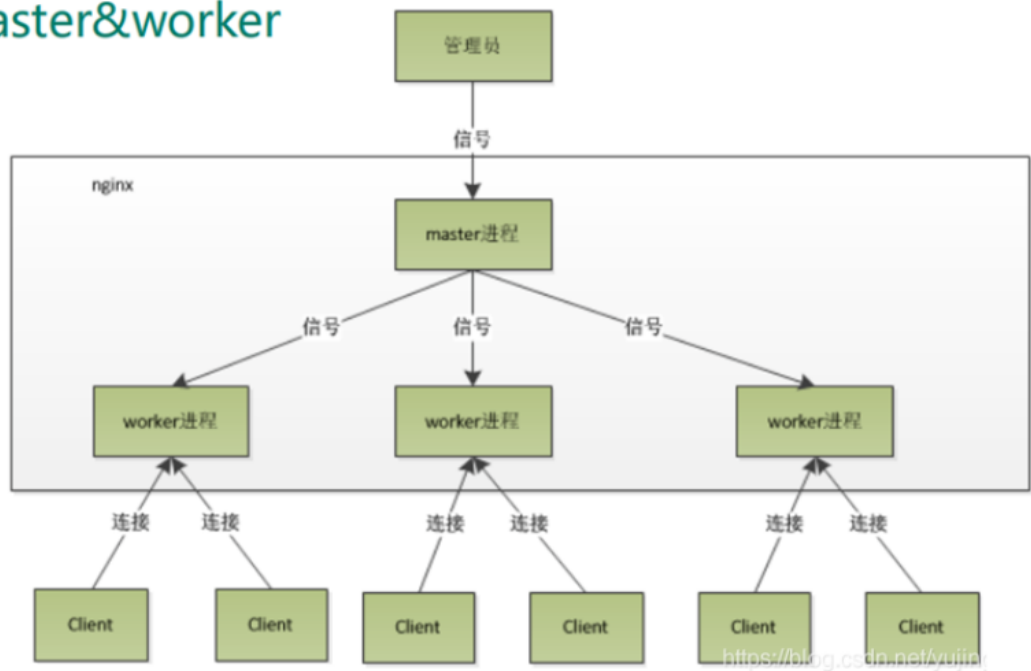
访问虚拟IP成功。



关闭主机147的Nginx和Keepalived，发现仍然可以访问。

原理解析

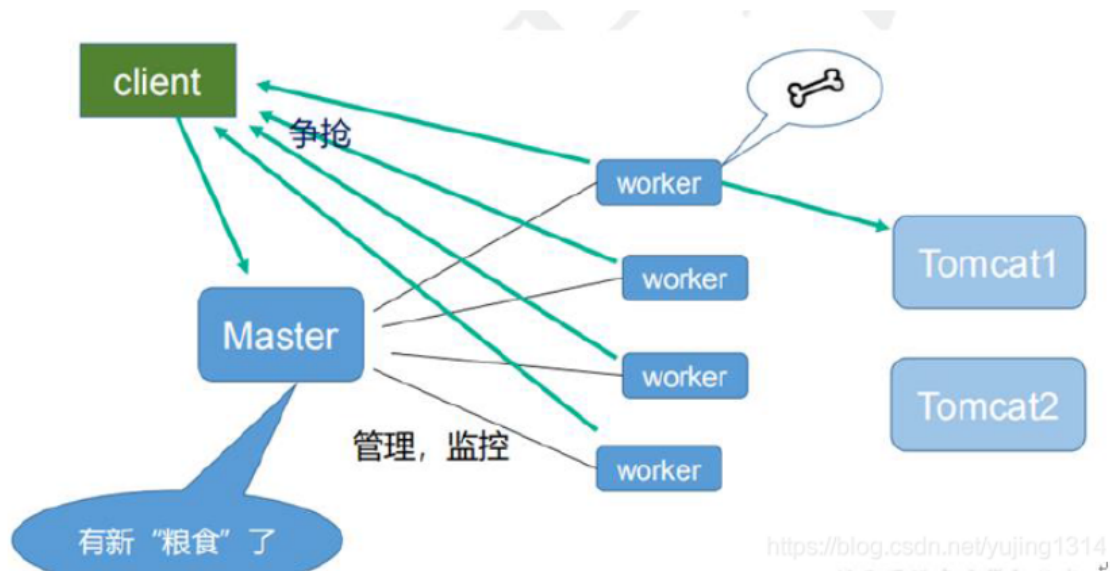
master&worker



如下图，就是启动了一个Master，一个Worker，Master是管理员，Worker是具体工作的进程。

```
[root@192 sbin]# ps -ef|grep nginx
root      28052      1  0 09:37 ?        00:00:00 nginx: master process ./nginx
nobody    28053    28052  0 09:37 ?        00:00:00 nginx: worker process
root      28055    15124  0 09:37 pts/0    00:00:00 grep --color=auto nginx
```

Worker如何工作：



小结

- Worker数应该和CPU数相等
- 一个Master多个Worker可以使用热部署，同时Worker是独立的，一个挂了不会影响其他的。

原文链接: <https://blog.csdn.net/yujing1314/article/details/107000737>