



课堂笔记

内容介绍

nginx

1、nginx 简介

- (1) 什么是 nginx 和可以做什么事情
- (2) 正向代理
- (3) 反向代理
- (4) 动静分离

2、Nginx 的安装

- (1) 在 linux 系统中安装 nginx

3、Nginx 的常用命令和配置文件

4、Nginx 配置实例 1 反向代理

5、Nginx 配置实例 2 负载均衡

6、Nginx 配置实例 3 动静分离

7、Nginx 的高可用集群

- (1) nginx 配置主从模式
- (2) nginx 配置双主模式

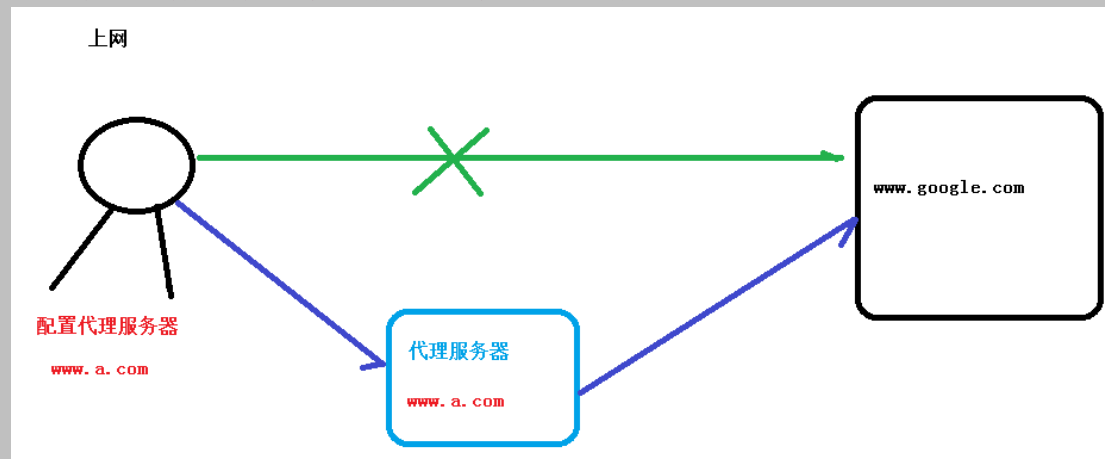
Nginx 的简介

1、什么是 nginx

Nginx 是高性能的 HTTP 和反向代理的服务器，处理高并发能力是十分强大的，能经受高负载的考验，有报告表明能支持高达 50,000 个并发连接数。

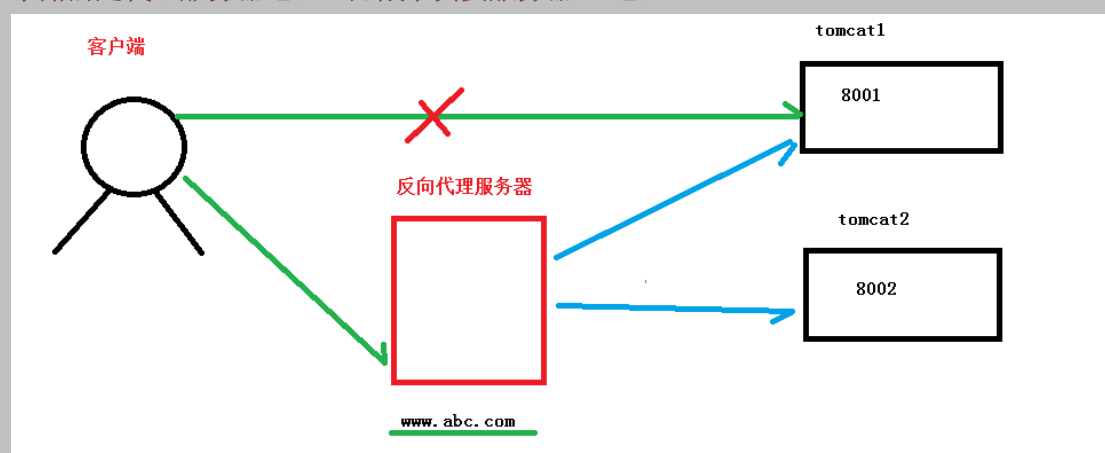
2、正向代理

- (1) 需要在客户端配置代理服务器进行指定网站访问



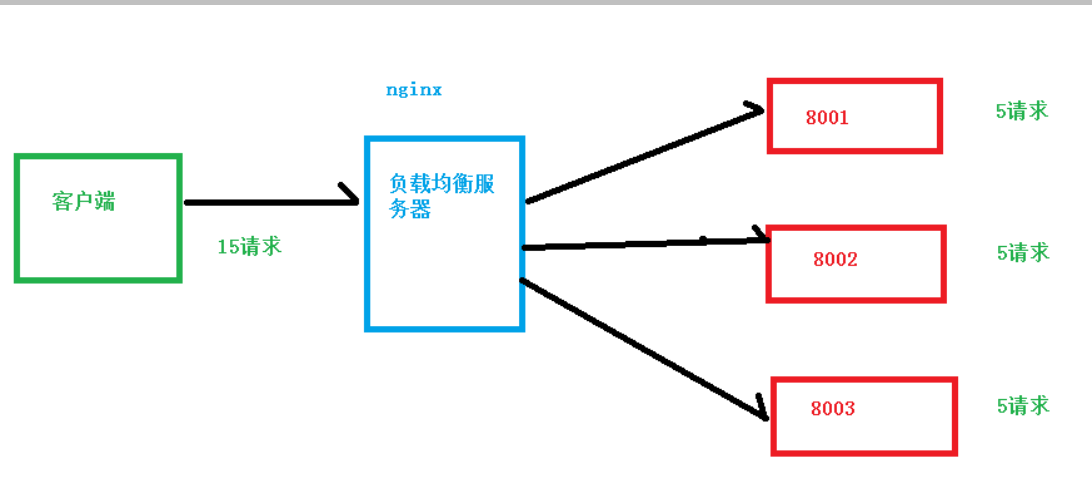
3、反向代理

暴露的是代理服务器地址，隐藏了真实服务器 IP 地址。

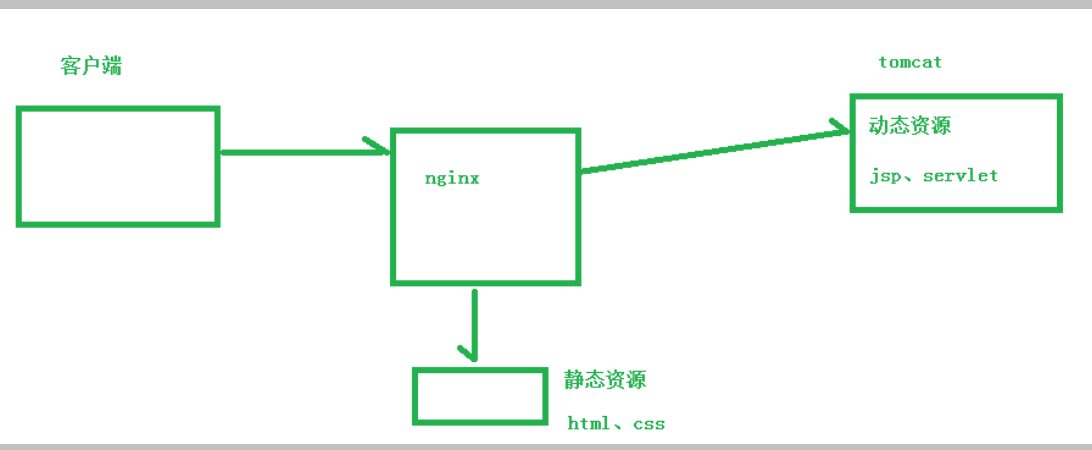


4、负载均衡

增加服务器的数量，然后将请求分发到各个服务器上，将原先请求集中到单个服务器上的情况改为将请求分发到多个服务器上，将负载分发到不同的服务器，也就是我们所说的负载均衡



5、动静分离



Nginx 的安装

1、准备工作

- (1) 打开虚拟机，使用远程连接工具连接 linux 操作系统
- (2) 到 nginx 官网下载软件

<http://nginx.org/>

Check out our latest release with easier dynamic module integration, additional TCP/UDP load-balancing features, enhancements to nginxScript, support for GeoIP2, and more. [Explore R11](#)

nginx news

2016-12-13 [nginx-1.11.7](#) mainline version has been released.

2016-11-15 [nginx-1.11.6](#) mainline version has been released.

2016-10-18 [nginx-1.10.2](#) stable version has been released.

2016-10-11 [nginx-1.11.5](#) mainline version has been released.

2016-09-13 [nginx-1.11.4](#) mainline version has been released.

2016-07-26 [nginx-1.11.3](#) mainline version has been released.



english
[русский](#)

news
[2015](#)
[2014](#)
[2013](#)
[2012](#)
[2011](#)
[2010](#)

2、开始进行 nginx 安装

(1) 安装 pcre 依赖

第一步 联网下载 pcre 压缩文件依赖

wget <http://downloads.sourceforge.net/project/pcre/pcre/8.37/pcre-8.37.tar.gz>

```
[root@localhost src]# ls
debug  kernels  pcre-8.37.tar.gz
```

第二步 解压压缩文件

使用命令 `tar -xvf pcre-8.37.tar.gz`

第三步 `./configure` 完成后, 回到 pcre 目录下执行 `make`, 最后执行 `make install`

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

(2) 安装 openssl、zlib、gcc 依赖

`yum -y install make zlib zlib-devel gcc-c++ libtool openssl openssl-devel`

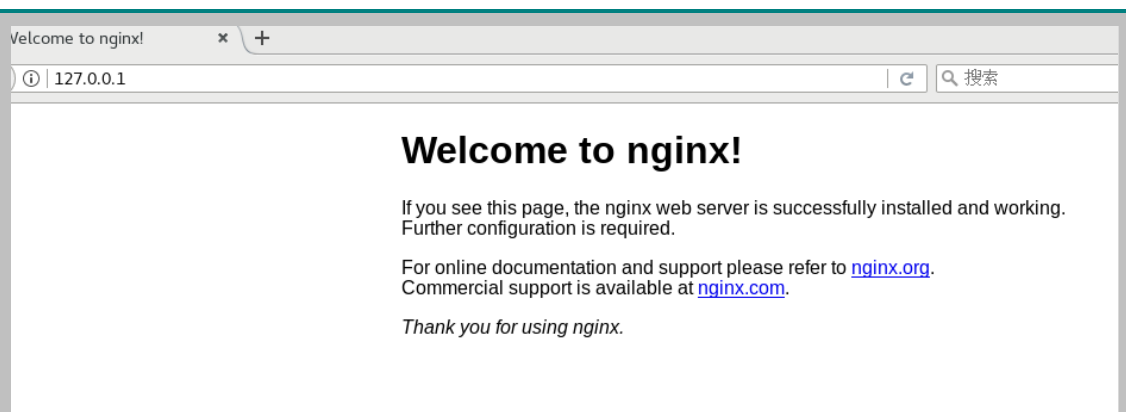
(3) 安装 nginx

* 使用命令解压

* `./configure`

* `make && make install`

进入目录 `/usr/local/nginx/sbin/nginx` 启动服务



在 windows 系统中访问 linux 中 nginx，默认不能访问的，因为防火墙问题

(1) 关闭防火墙

(2) 开放访问的端口号，80 端口

查看开放的端口号

firewall-cmd --list-all

设置开放的端口号

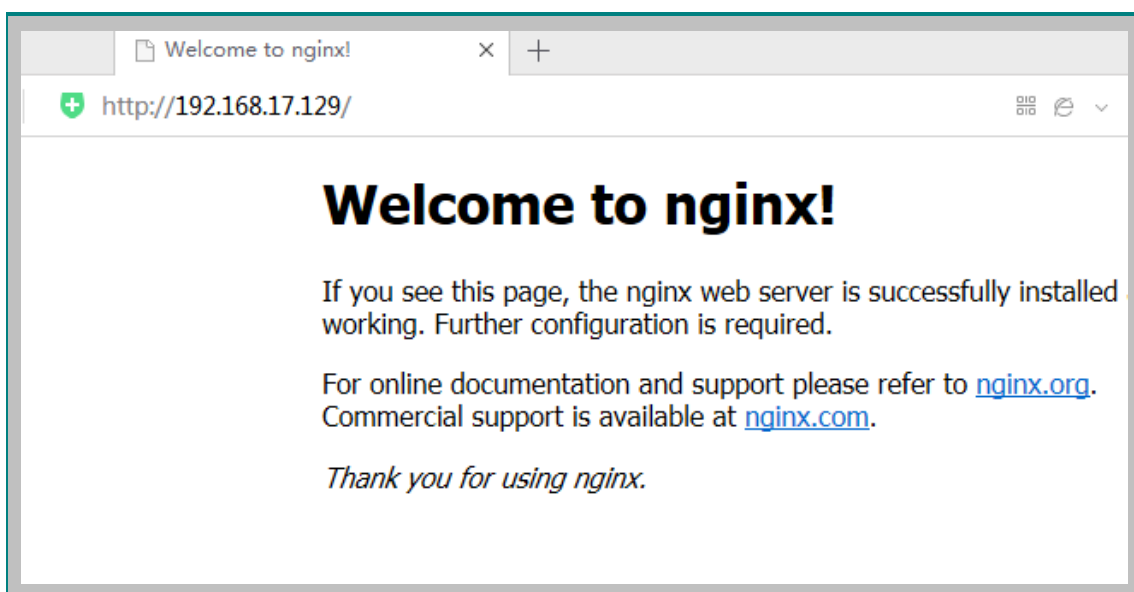
firewall-cmd --add-service=http --permanent

firewall-cmd --add-port=80/tcp --permanent

重启防火墙

firewall-cmd --reload

```
[root@localhost /]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens33
  sources:
  services: ssh dhcpv6-client http
  ports: 80/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```



Nginx 的常用的命令

进入 nginx 目录中

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

1、查看 nginx 版本号

```
./nginx -v
```

```
root@localhost sbin]# ./nginx -v
nginx version: nginx/1.12.2
```

2、启动 nginx

```
./nginx
```

3、停止 nginx

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

4、重新加载 nginx

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

Nginx 的配置文件

1、nginx 配置文件位置

6

更多 Java - 大数据 - 前端 - python 人工智能资料下载，可访问百度：尚硅谷官网

```
cd /usr/local/nginx/conf/nginx.conf
```

```
root@localhost nginx]# cd conf
root@localhost conf]# ls
fastcgi.conf      fastcgi_params    koi-utf  mime.types    nginx.conf  scgi_params  uwsgi_params  win-utf
fastcgi.conf.default fastcgi_params.default koi-win  mime.types.default nginx.conf.default scgi_params.default uwsgi_params.default
```

2、配置文件中的内容

包含三部分内容

(1) 全局块：配置服务器整体运行的配置指令

比如 `worker_processes 1`; 处理并发数的配置

(2) events 块：影响 Nginx 服务器与用户的网络连接

比如 `worker_connections 1024`; 支持的最大连接数为 1024

(3) http 块

还包含两部分：

http 全局块

server 块

Nginx 配置实例-反向代理实例 1

1、实现效果

(1) 打开浏览器，在浏览器地址栏输入地址 www.123.com，跳转到 liunx 系统 tomcat 主页面中

2、准备工作

(1) 在 liunx 系统安装 tomcat，使用默认端口 8080

* tomcat 安装文件放到 liunx 系统中，解压

* 进入 tomcat 的 bin 目录中，`./startup.sh` 启动 tomcat 服务器

(2) 对外开放访问的端口

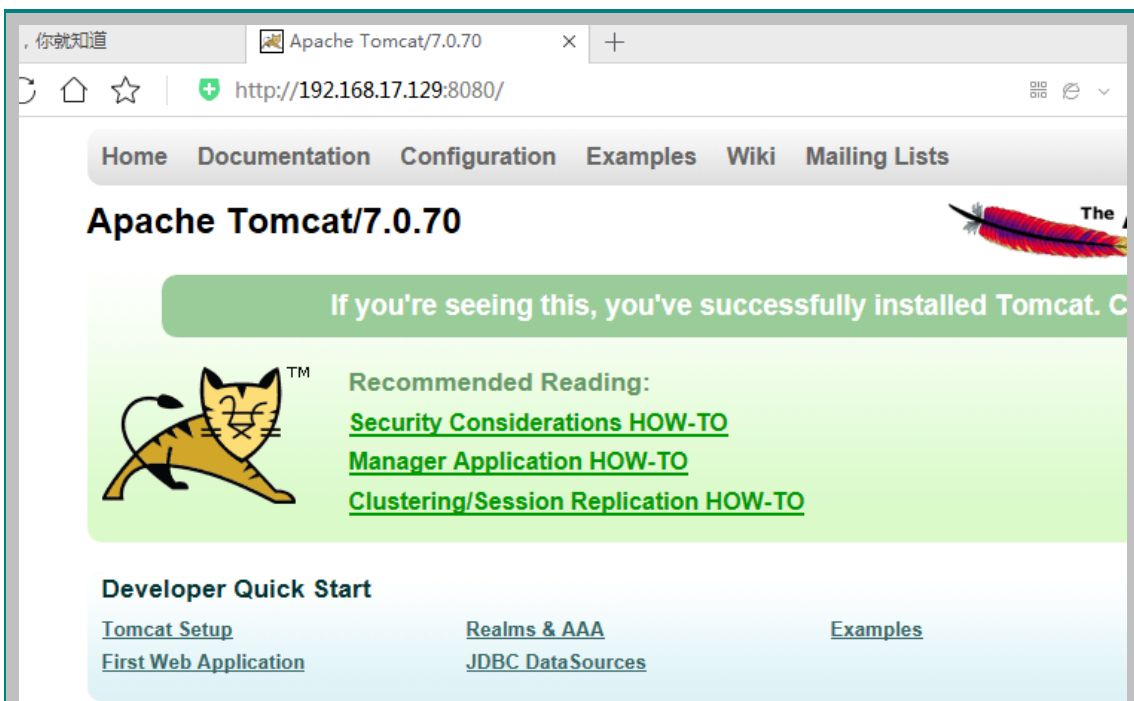
`firewall-cmd --add-port=8080/tcp --permanent`

`firewall-cmd --reload`

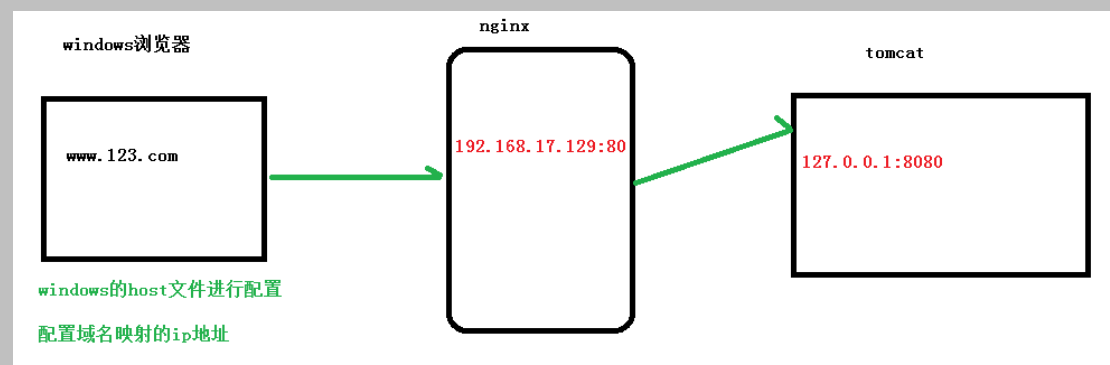
查看已经开放的端口号

`firewall-cmd --list-all`

(3) 在 windows 系统中通过浏览器访问 tomcat 服务器

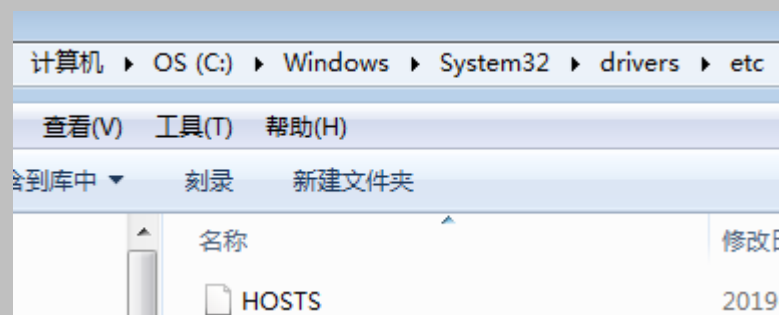


3、访问过程的分析



4、具体配置

第一步 在 windows 系统的 host 文件进行域名和 ip 对应关系的配置



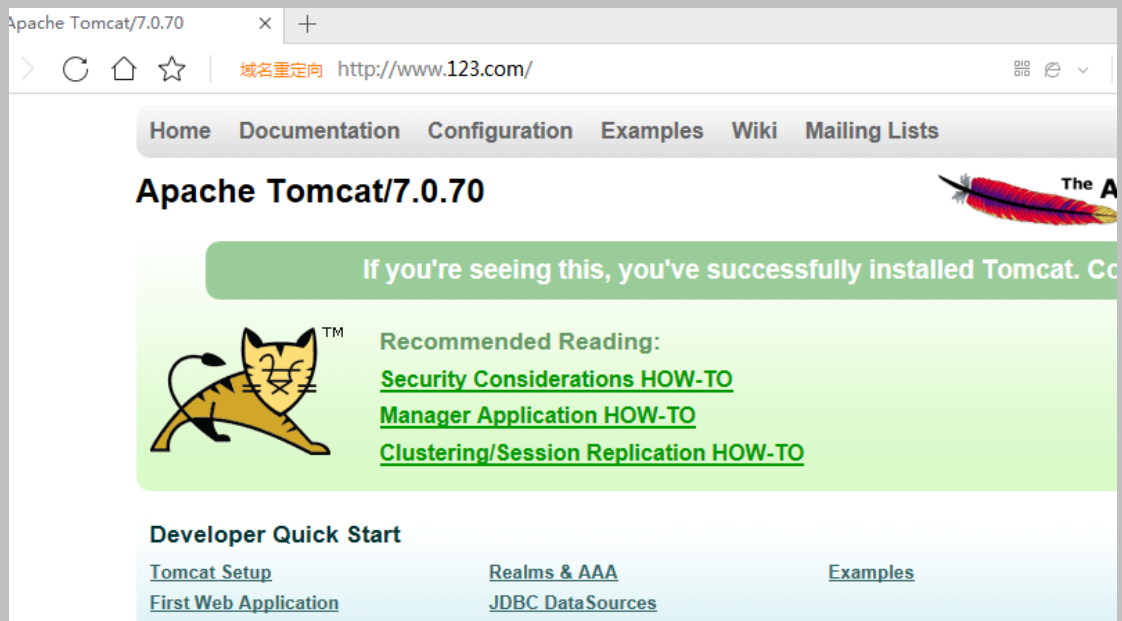
(1) 添加内容在 host 文件中

192.168.17.129 www.123.com

第二步 在 nginx 进行请求转发的配置（反向代理配置）

```
server {  
    listen      80;  
    server_name 192.168.17.129;  
  
    #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;  
    }  
}
```

5、最终测试



Nginx 配置实例-反向代理实例 2

1、实现效果

使用 nginx 反向代理，根据访问的路径跳转到不同端口的服务中
nginx 监听端口为 9001，

访问 <http://192.168.17.129:9001/edu/> 直接跳转到 127.0.0.1:8080
访问 <http://192.168.17.129:9001/vod/> 直接跳转到 127.0.0.1:8081

2、准备工作

- (1) 准备两个 tomcat 服务器，一个 8080 端口，一个 8081 端口
- (2) 创建文件夹和测试页面

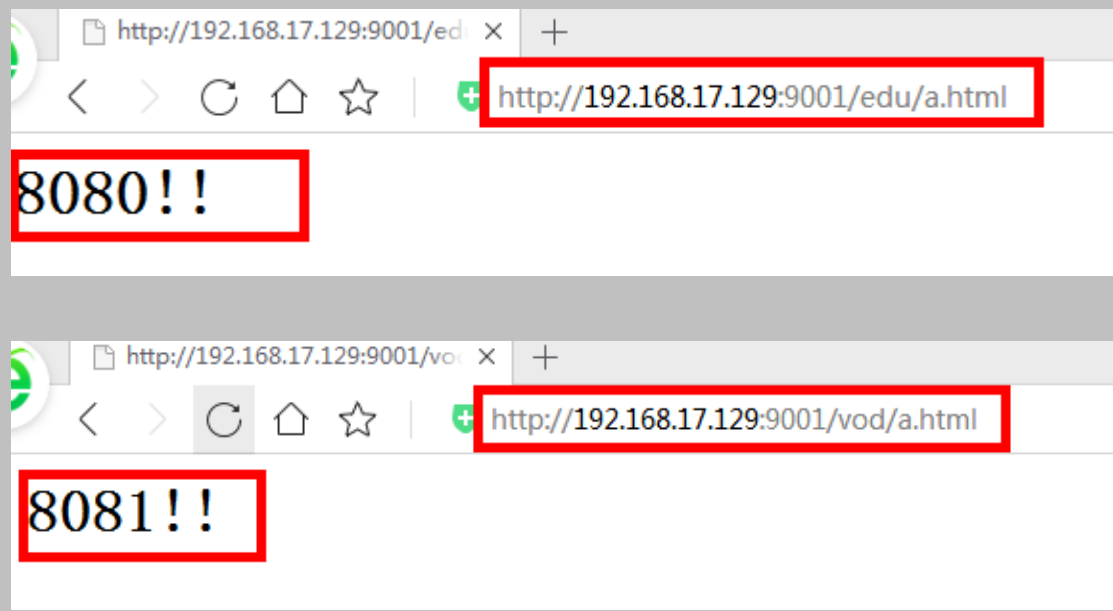
3、具体配置

- (1) 找到 nginx 配置文件，进行反向代理配置

```
server {  
    listen      9001;  
    server_name 192.168.17.129;  
  
    location ~ /edu/ {  
        proxy_pass http://127.0.0.1:8080;  
    }  
  
    location ~ /vod/ {  
        proxy_pass http://127.0.0.1:8081;  
    }  
}
```

- (2) 开放对外访问的端口号 9001 ~~8080 8081~~

4、最终测试



Nginx 配置实例-负载均衡

1、实现效果

(1) 浏览器地址栏输入地址 `http://192.168.17.129/edu/a.html` 负载均衡效果, 平均 8080 和 8081 端口中

2、准备工作

(1) 准备两台 tomcat 服务器, 一台 8080, 一台 8081

(2) 在两台 tomcat 里面 webapps 目录中, 创建名称是 edu 文件夹, 在 edu 文件夹中创建页面 a.html, 用于测试

3、在 nginx 的配置文件中负载均衡的配置

```
upstream myserver {  
    server 192.168.17.129:8080;  
    server 192.168.17.129:8081;  
}
```

```
server {  
    listen      80;  
    server_name 192.168.17.129;  
  
    #charset koi8-r;  
  
    #access_log logs/host.access.log main;  
  
    location / {  
        proxy_pass http://myserver;  
        root      html;  
        index     index.html index.htm;  
    }  
}
```

4、nginx 分配服务器策略

第一种 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

第三种 ip_hash

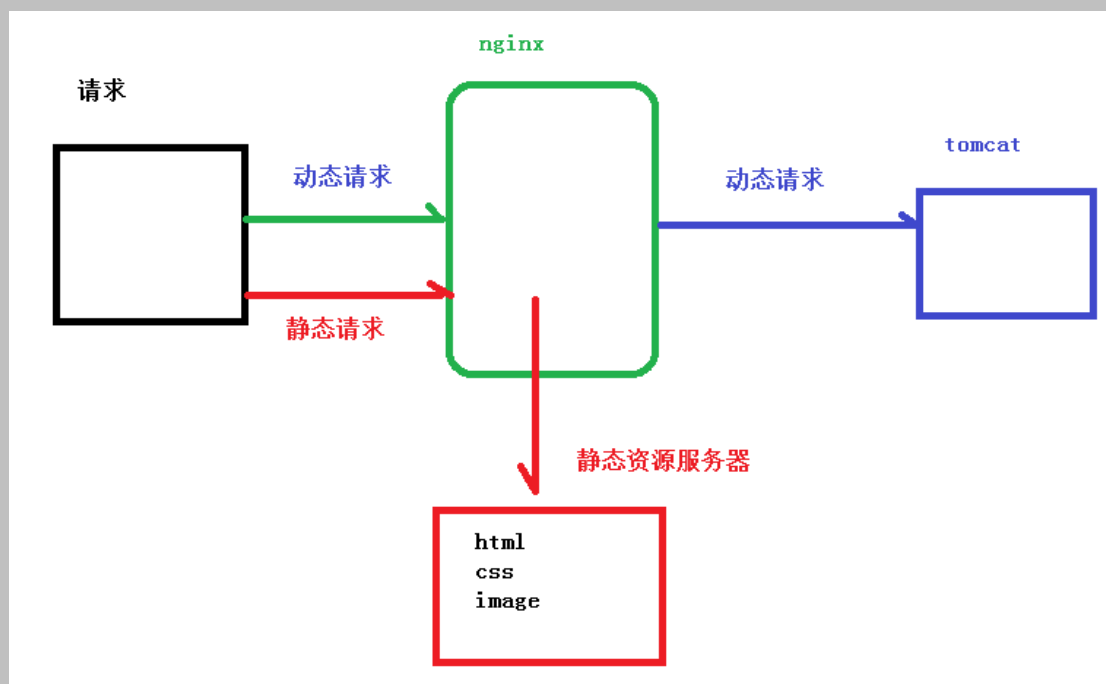
每个请求按访问 ip 的 hash 结果分配, 这样每个访客固定访问一个后端服务器

第四种 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。

Nginx 配置实例-动静分离

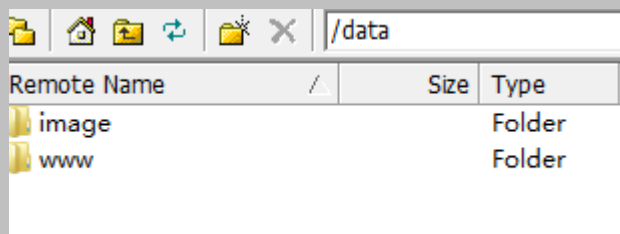
1、什么是动静分离



通过 `location` 指定不同的后缀名实现不同的请求转发。通过 `expires` 参数设置，可以使浏览器缓存过期时间，减少与服务器之前的请求和流量。具体 `Expires` 定义：是给一个资源设定一个过期时间，也就是说无需去服务端验证，直接通过浏览器自身确认是否过期即可，所以不会产生额外的流量。此种方法非常适合不经常变动的资源。（如果经常更新的文件，不建议使用 `Expires` 来缓存），我这里设置 `3d`，表示在这 3 天之内访问这个 URL，发送一个请求，比对服务器该文件最后更新时间没有变化，则不会从服务器抓取，返回状态码 `304`，如果有修改，则直接从服务器重新下载，返回状态码 `200`。

2、准备工作

（1）在 `liunx` 系统中准备静态资源，用于进行访问



3、具体配置

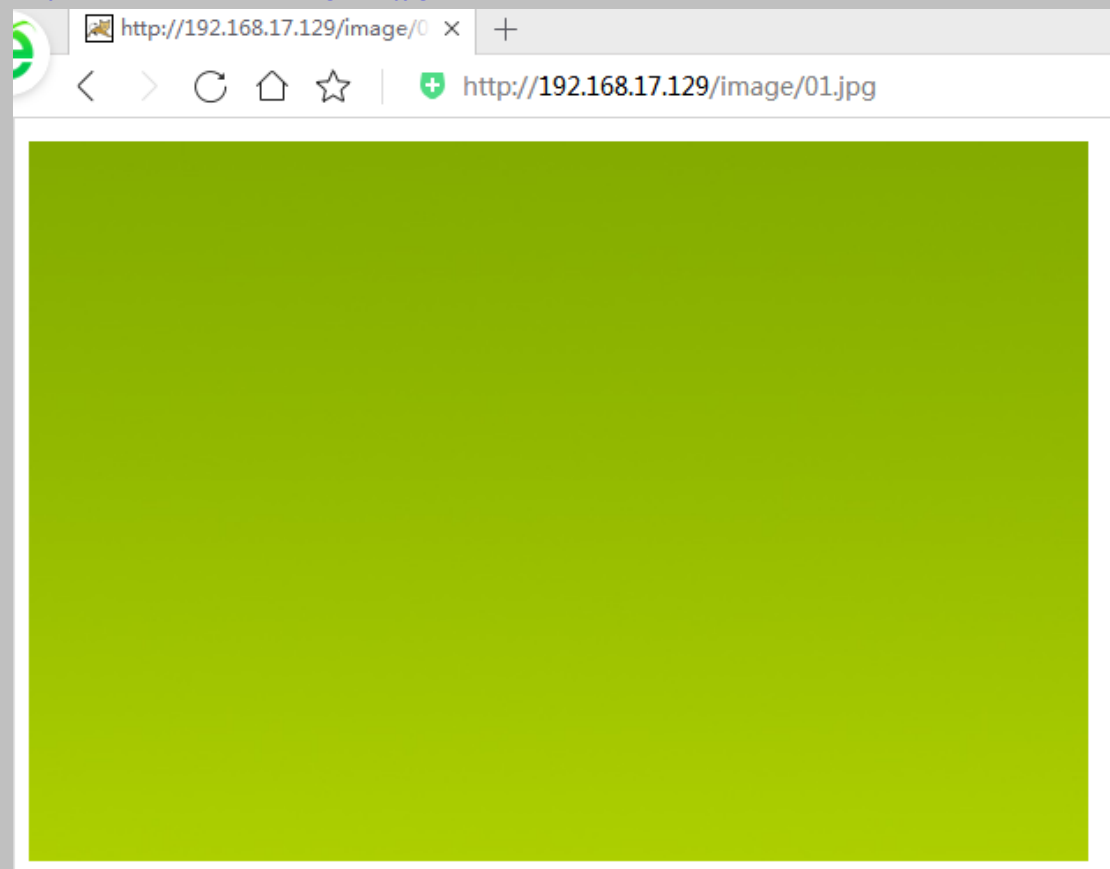
（1）在 `nginx` 配置文件中配置

```
server {  
    listen      80;  
    server_name 192.168.17.129;  
  
    #charset koi8-r;  
  
    #access_log logs/host.access.log main;  
  
    location /www/ {  
        root   /data/;  
        index  index.html index.htm;  
    }  
  
    location /image/ {  
        root   /data/;  
        autoindex on;  
    }  
}
```

4、最终测试

(1) 浏览器中输入地址

<http://192.168.17.129/image/01.jpg>

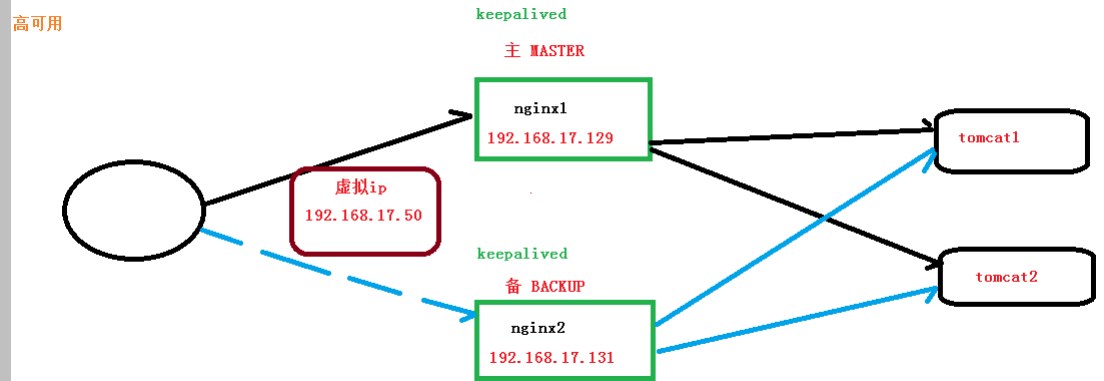


* 因为配置文件 `autoindex on`



Nginx 配置高可用的集群

1、什么是 nginx 高可用



- (1) 需要两台 nginx 服务器
- (2) 需要 keepalived
- (3) 需要虚拟 ip

2、配置高可用的准备工作

- (1) 需要两台服务器 192.168.17.129 和 192.168.17.131
- (2) 在两台服务器安装 nginx
- (3) 在两台服务器安装 keepalived

3、在两台服务器安装 keepalived

- (1) 使用 yum 命令进行安装

`yum install keepalived -y`

```
[root@localhost usr]# yum install keepalived -y
```

- (2) 安装之后，在 `etc` 里面生成目录 `keepalived`，有文件 `keepalived.conf`

4、完成高可用配置（主从配置）

- (1) 修改 `/etc/keepalived/keepalived.conf` 配置文件

```
global_defs {
    notification_email {
        acassen@firewall.loc
        failover@firewall.loc
        sysadmin@firewall.loc
    }
    notification_email_from Alexandre.Cassen@firewall.loc
    smtp_server 192.168.17.129
    smtp_connect_timeout 30
    router_id LVS_DEVEL
}

vrrp_script chk_http_port {

    script "/usr/local/src/nginx_check.sh"

    interval 2      #（检测脚本执行的间隔）

    weight 2

}

vrrp_instance VI_1 {
    state BACKUP    # 备份服务器上将 MASTER 改为 BACKUP
    interface ens33 //网卡
    virtual_router_id 51  # 主、备机的 virtual_router_id 必须相同
    priority 90        # 主、备机取不同的优先级，主机值较大，备份机值较小
    advert_int 1
```

```
authentication {  
    auth_type PASS  
    auth_pass 1111  
}  
virtual_ipaddress {  
    192.168.17.50 // VRRP H 虚拟地址  
}  
}
```

(2) 在/usr/local/src 添加检测脚本

```
#!/bin/bash  
A=`ps -C nginx --no-header |wc -l`  
if [ $A -eq 0 ];then  
    /usr/local/nginx/sbin/nginx  
    sleep 2  
    if [ `ps -C nginx --no-header |wc -l` -eq 0 ];then  
        killall keepalived  
    fi  
fi
```

(3) 把两台服务器上 nginx 和 keepalived 启动

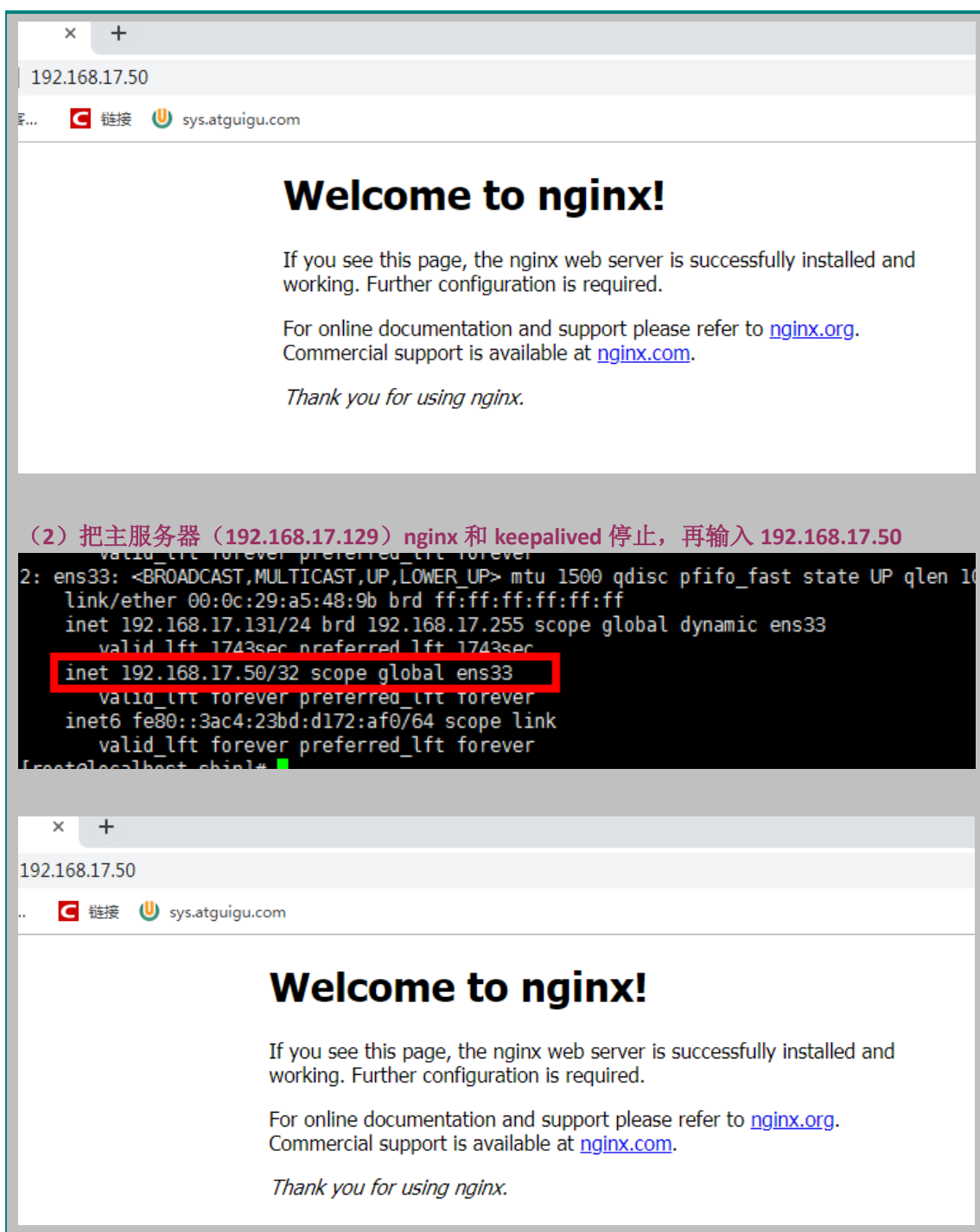
启动 nginx: ./nginx

启动 keepalived: systemctl start keepalived.service

5、最终测试

(1) 在浏览器地址栏输入 虚拟 ip 地址 192.168.17.50

```
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fa  
    link/ether 00:0c:29:17:b0:34 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.17.129/24 brd 192.168.17.255 scope global dynamic  
        valid_lft 1265000s preferred_lft 1265000s  
    inet 192.168.17.50/32 scope global ens33  
        valid_lft forever preferred_lft forever  
    inet6 fe80::58cb:67a2:182f:b4b0/64 scope link  
        valid_lft forever preferred_lft forever
```

x +

192.168.17.50

链接 sys.atguigu.com

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

(2) 把主服务器 (192.168.17.129) nginx 和 keepalived 停止, 再输入 192.168.17.50

```
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:a5:48:9b brd ff:ff:ff:ff:ff:ff
    inet 192.168.17.131/24 brd 192.168.17.255 scope global dynamic ens33
        valid_lft 1743sec preferred_lft 1743sec
    inet 192.168.17.50/32 scope global ens33
        valid_lft forever preferred_lft forever
    inet6 fe80::3ac4:23bd:d172:af0/64 scope link
        valid_lft forever preferred_lft forever
```

x +

192.168.17.50

链接 sys.atguigu.com

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

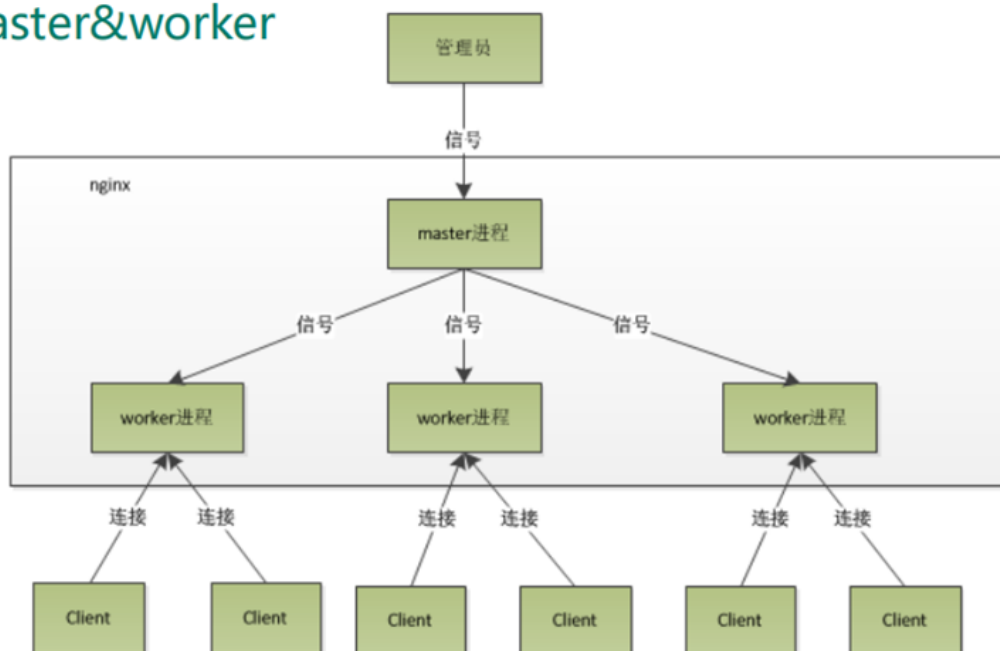
Thank you for using nginx.

Nginx 的原理

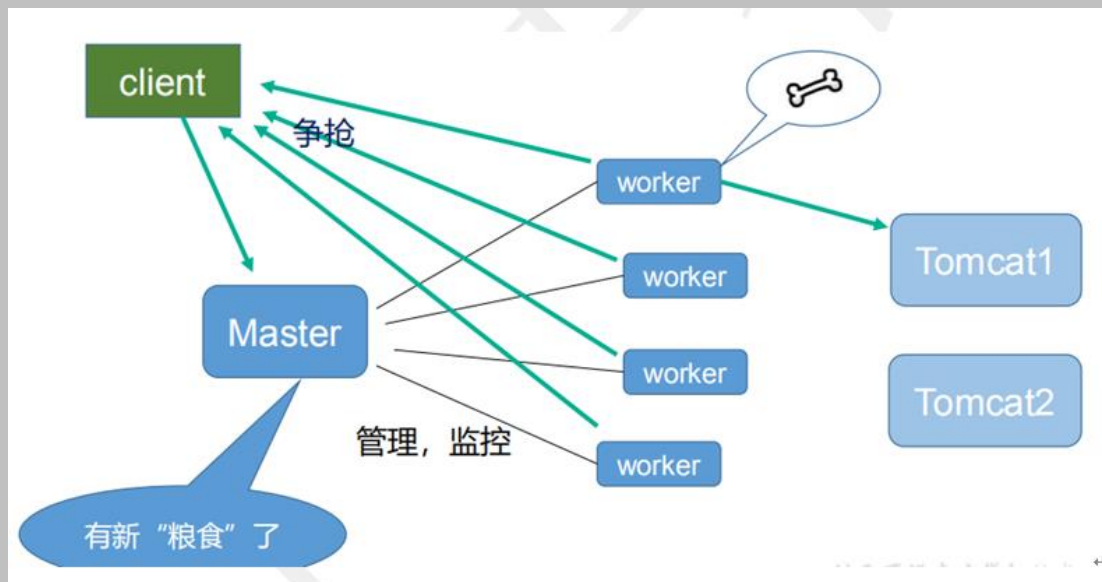
1、mater 和 worker

```
root      3094      1  0 13:12 ?        00:00:00 nginx: master process ./nginx
nobody    3095    3094  0 13:12 ?        00:00:00 nginx: worker process
```

master&worker



2、worker 如何进行工作的



3、一个 master 和多个 woker 有好处

- (1) 可以使用 `nginx -s reload` 热部署，利用 nginx 进行热部署操作
- (2) 每个 worker 是独立的进程，如果有其中的一个 worker 出现问题，其他 worker 独立的，继续进行争抢，实现请求过程，不会造成服务中断

4、设置多少个 woker 合适

worker 数和服务器的 cpu 数相等是最为适宜的

5、连接数 worker_connection

第一个：发送请求，占用了 woker 的几个连接数？

答案：2 或者 4 个

第二个：nginx 有一个 master，有四个 woker，每个 woker 支持最大的连接数 1024，支持的最大并发数是多少？

- 普通的静态访问最大并发数是： $\text{worker_connections} * \text{worker_processes} / 2$,
- 而如果是 HTTP 作为反向代理来说，最大并发数量应该是 $\text{worker_connections} * \text{worker_processes} / 4$ 。