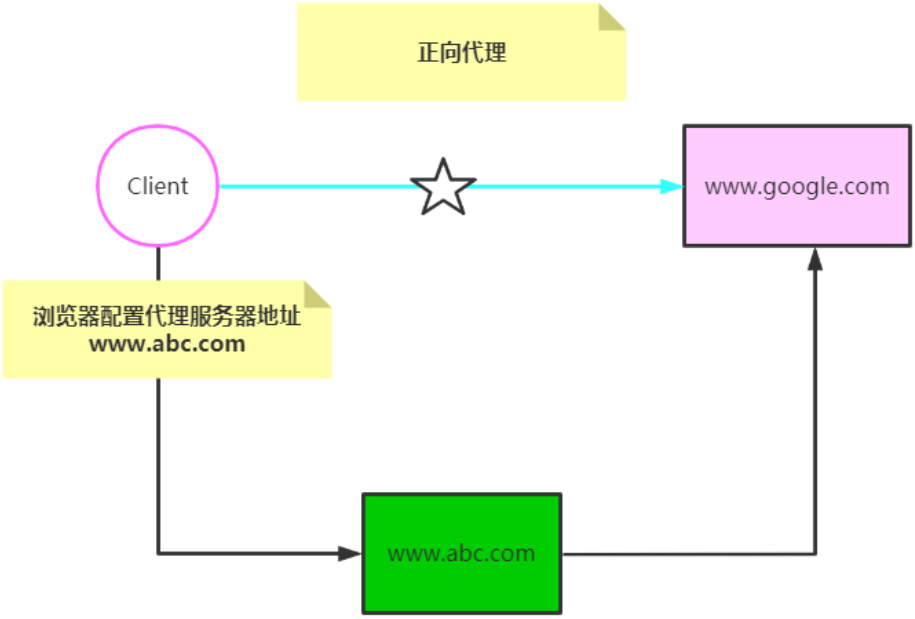
一、Nginx 简介

1、什么是NGINX

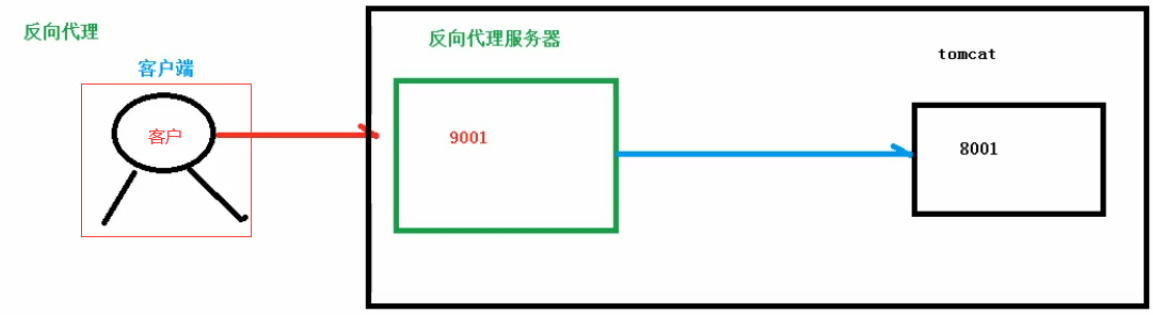
```
Nginx ("engine x")是一个高性能的HTTP和反向代理服务器，特点是占有内存少，并发能
力强，事实上nginx的并发能力确实在同类型的网页服务器中表现较好
Nginx专为性能优化而开发，性能是其最重要的考量，实现上非常注重效率，能经受高负载
的考验，有报告表明能支持高达50000个并发连接数。
```

2、反向代理

a. 正向代理 在客户端(浏览器)配置代理服务器，通过代理服务器进行互联网访问



b. 反向代理 反向代理，其实客户端对代理是无感知的，因为客户端不需要任何配置就可以访问，我们只需要将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后,在返回给客户端，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。

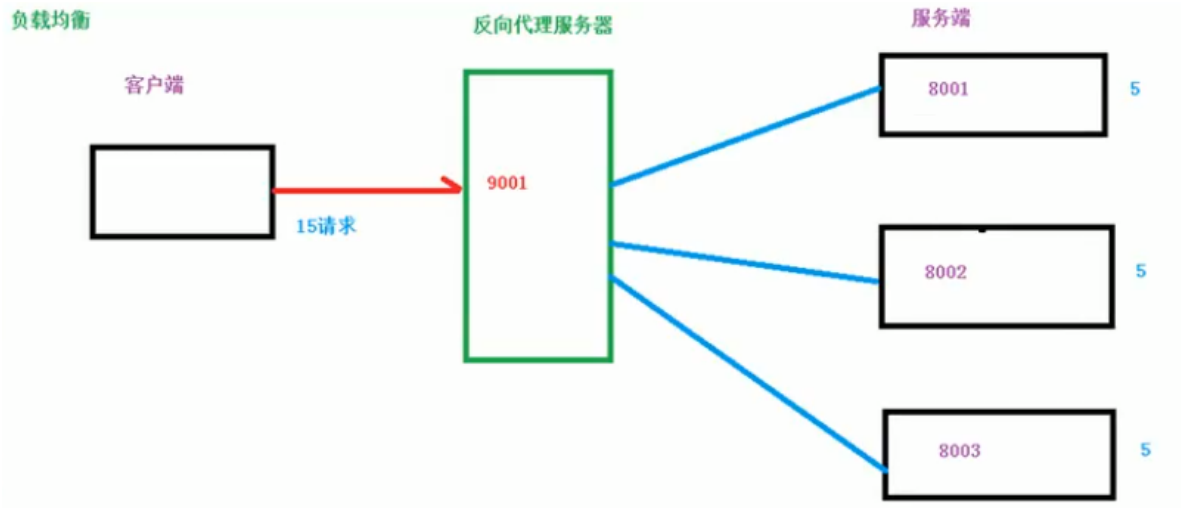


参考链接：

反向代理为何叫反向代理？ - 刘志军的回答 - 知乎 https://www.zhihu.com/question/24723688/answer/128105528)

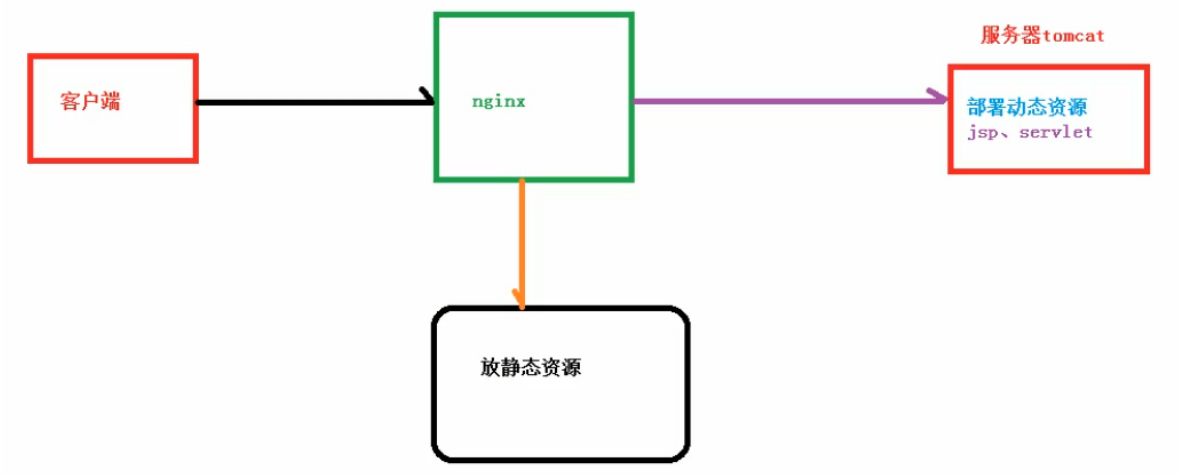
3、负载均衡

单个服务器解决不了，我们增加服务器的数量，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



4、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由不同的服务器来解析，加快解析速 度。降低原来单个服务器的压力。



二、Nginx 安装

下面的操作是以Centos7为例

1. 使用远程连接工具连接Centos7操作系统 2. 安装nginx相关依赖

```
gcc
pcre
openssl
zlib
```

① 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(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 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 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
```

3. 安装Nginx

① 下载nginx，两种方式

> a. 直接下载`.tar.gz`安装包，地址：https://nginx.org/en/download.html
>
> b. **使用`wget`命令下载（推荐）**。确保系统已经安装了wget，如果没有安装，执行 yum install wget 安装。

```
$ wget -c https://nginx.org/download/nginx-1.19.0.tar.gz
```

② 依然是直接命令：

```
$ tar -zxvf nginx-1.19.0.tar.gz
$ cd nginx-1.19.0
```

③ 配置：

其实在 nginx-1.12.0 版本中你就不需要去配置相关东西，默认就可以了。当然，如果你要自己配置目录也是可以的。 1.使用默认配置

```
$ ./configure
```

2.自定义配置（不推荐）

```
$ ./configure \
--prefix=/usr/local/nginx \
--conf-path=/usr/local/nginx/conf/nginx.conf \
--pid-path=/usr/local/nginx/conf/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
```

> 注：将临时文件目录指定为/var/temp/nginx，需要在/var下创建temp及nginx目录

④ 编辑安装

```
$ make && make install
```

查看版本号(`使用nginx操作命令前提条件:必须进入nginx的目录/usr/local/nginx/sbin`.)

```
$ ./nginx -v
```

查找安装路径:

```
$ whereis nginx
```

⑤ 启动，停止nginx

```
$ cd /usr/local/nginx/sbin/
$ ./nginx
$ ./nginx -s stop
$ ./nginx -s quit
$ ./nginx -s reload
```

查询nginx进程:

```
$ ps aux|grep nginx
```

由于linux默认没有开放80端口，此处需要开启端口

```
添加：firewall-cmd --zone=public --add-port=80/tcp --permanent    （--
permanent永久生效，没有此参数重启后失效）重新载入：firewall-cmd --reload
查看：firewall-cmd --zone=public --query-port=80/tcp删除：firewall-cmd --
zone=public --remove-port=80/tcp --permanent
```

成功后在浏览器访问可以看到下面的页面

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.

案列1

1. 实现效果，在浏览器中输入 www.123.com 后跳转到Linux上Tomcat的主页
2. 准备工作：

安装tomcat，修改server.xml文件设置端口号8088，防火墙开放该端口号

```
添加端口号：
firewall-cmd --zone=public --add-port=8088/tcp --permanent
重新加载：
firewall-cmd --reload
查看防火墙开发的所有端口
firewall-cmd --list-all
```

3. 在window本地host中配置www.123.com
4. 配置linux上面的nginx配置文件

```
server {
        listen       80;
        server_name  192.168.138.3;

        #charset koi8-r;

        #access_log  logs/host.access.log  main;

        location / {
            root   html;
            proxy_pass   http://127.0.0.1:8088;
            index  index.html index.htm;
        }

```

案例2

实现在浏览器输入192.xx.xx.xx/edu跳转到tomcathttp://127.0.0.1:8088

输入192.xx.xx.xx/org跳转到tomcathttp://127.0.0.1:8089

1. 准备两个tomcat，修改端口号
2. 分别在2个tomcat中创建不同文件夹和页面

```
drwxr-x---. 15 root root 4096 11月 15 09:45 docs
drwxr-x---.  6 root root   83 11月 15 09:45 examples
drwxr-x---.  5 root root   87 11月 15 09:45 host-manager
drwxr-x---.  6 root root  114 11月 15 09:45 manager
drwxr-xr-x.  2 root root   24 11月 15 10:04 org
drwxr-x---.  3 root root  283 11月 15 09:45 ROOT
[root@localhost webapps]#

```

nginx配置

```
  server {
          listen       9001;
          server_name  192.168.138.3;

          location ~ /edu/ {
              proxy_pass   http://127.0.0.1:8088;
          }

          location ~ /org/ {
              proxy_pass   http://127.0.0.1:8089;
        }
      }
```

添加端口号:

```
firewall-cmd --zone=public --add-port=9001/tcp --permanent
重新加载:
firewall-cmd --reload
查看防火墙开发的所有端口
firewall-cmd --list-all
```

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。

案例3 负载均衡

配置

```
#配置负载均衡
      upstream myserver {
          server  192.168.138.3:8088;
          server  192.168.138.3:8089;
      }

      server {
          listen       80;
          server_name       192.168.138.3;

```

```
        location / {
            proxy_pass   http://myserver;
            index  index.html index.htm;
        }
    }

```

访问：http://192.168.138.3/edu/index.html 测试

负载分配策略

在linux下有Nginx、LVS、 Haproxy 等等服务可以提供负载均衡服务，而且Nginx提供了几种分配方式(策略):。

- **1、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

- **2、weight** weight代表权重默认为1,权重越高被分配的客户端越多。 指定轮询几率，weight和访问比率成正比，用于后端服务器性能不均的情况。例如:

```
upstream server pool{
  ip_ hash
  server 192.168.5.21:80 weight=3;
  server 192.168.5.22:80 weight=2;
}
```

- **3、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,可以解决 session 的问题。例如:

```
upstream server pool{
  ip_ hash;
  server 192.168.5.21:80;
  server 192.168.5.22:80;
}
```

- **4、fair (第三方)**（能者多劳） 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 192.168.5.21:80;
    server 192.168.5.22:80;
    fair;
}
```

案例4 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)，如果设置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
>
> /data/image 图片文件夹
>
> /data/www html文件夹

3、具体配置

```
location /www/ {
    root /data/;
    index index.html index.htm;
}

location /image/ {
    root /data/;
    autoindex on;
}

```

4、实际测试

```
http://192.168.138.3/www/index.html
http://192.168.138.3/image/a.png
```

浏览器响应头

```
Request URL: http://192.168.138.3/www/
Request Method: GET
Status Code: 304 Not Modified
Remote Address: 192.168.138.3:80
Referrer Policy: strict-origin-when-cross-origin
```

六、Nginx配置高可用集群

1、什么是nginx高可用

2、配置高可用的准备工作

> (1) 需要两台服务器192.168.138.xx 和192.168.138.xx (2) 在两台服务器安装nginx. (3) 在两合服务器安装keepalived.

3、在两台服务器安装keepalived 使用yum命令进行安装

```
$ yum install keepalived
$ rpm -q -a keepalived    #查看是否已经安装上
```

默认安装路径: /etc/keepalived

安装之后，在etc里面生成目录keepalived, 有配置文件keepalived.conf

4、完成高可用配置(主从配置)

（1）修改keepalived的配置文件`keepalived.conf`为：

```
global_defs {
    notification_email {
      acassen@firewall.loc
      failover@firewall.loc
      sysadmin@firewall.loc
    }
    notification_email_from Alexandre.Cassen@firewall.loc
    smtp_ server 192.168.138.3
    smtp_connect_timeout 30
    router_id LVS_DEVEL # LVS_DEVEL这字段在/etc/hosts文件中看；通过它访问到主机
}

vrrp_script chk_http_ port {
    script "/usr/local/src/nginx_check.sh"
    interval 2   # (检测脚本执行的间隔)2s
    weight 2  #权重，如果这个脚本检测为真，服务器权重+2
}

vrrp_instance VI_1 {
    state MASTER   # 备份服务器上将MASTER 改为BACKUP
    interface ens33 //网卡名称
    virtual_router_id 51 # 主、备机的virtual_router_id必须相同
    priority 100   #主、备机取不同的优先级，主机值较大，备份机值较小
    advert_int 1    #每隔1s发送一次心跳
    authentication {    # 校验方式，  类型是密码，密码1111
        auth type PASS
        auth pass 1111
    }
    virtual_ipaddress { # 虚拟ip
        192.168.138.8 // VRRP H虚拟ip地址
    }
}
```

（2）在路径/usr/local/src/ 下新建检测脚本 nginx_check.sh

nginx_check.sh

```
#! /bin/bash
A=`ps -C nginx -no-header | wc - 1`
if [ $A -eq 0];then
    /usr/local/nginx/sbin/nginx
    sleep 2
    if [`ps -C nginx --no-header| wc -1` -eq 0 ];then
        killall keepalived
    fi
fi
```

(3) 把两台服务器上nginx和keepalived启动

```
$ systemctl start keepalived.service        #keepalived启动
$ ps -ef I grep keepalived      #查看keepalived是否启动
```

5、最终测试

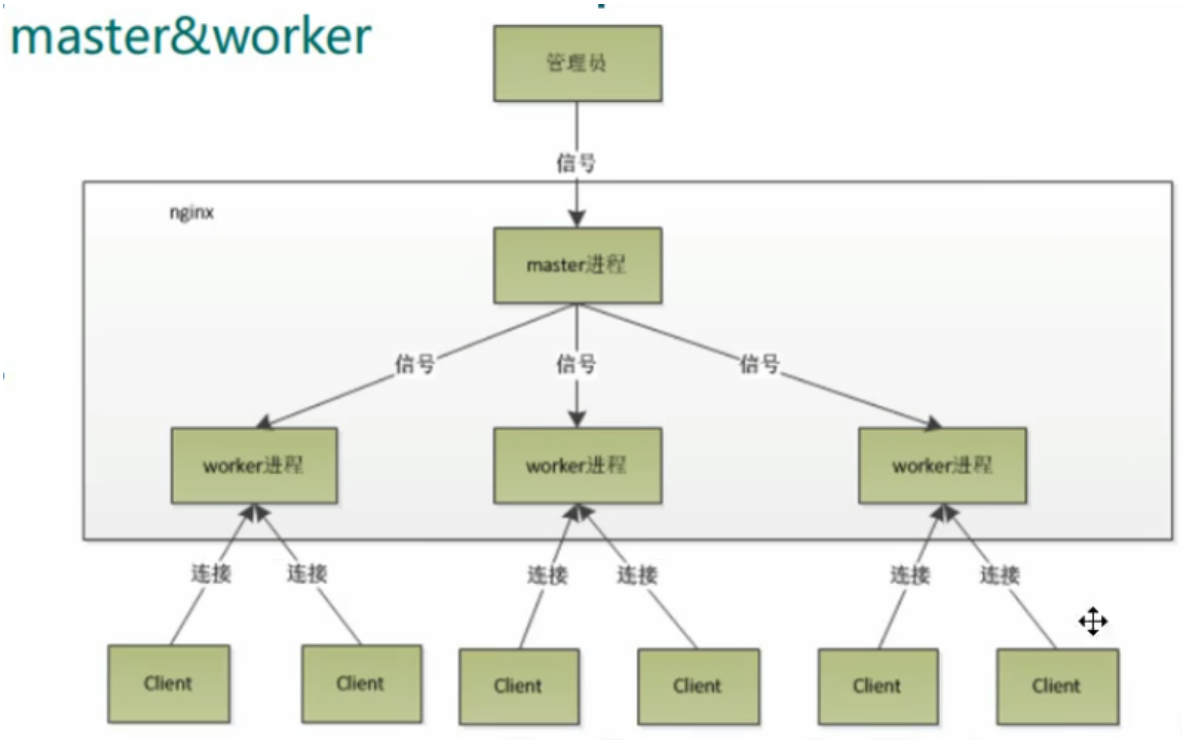
(1) 在浏览器地址栏输入虚拟ip地址192.168.17.50

(2) 把主服务器(192.168.138.3) nginx和keealived停止，再输入192.168.138.8

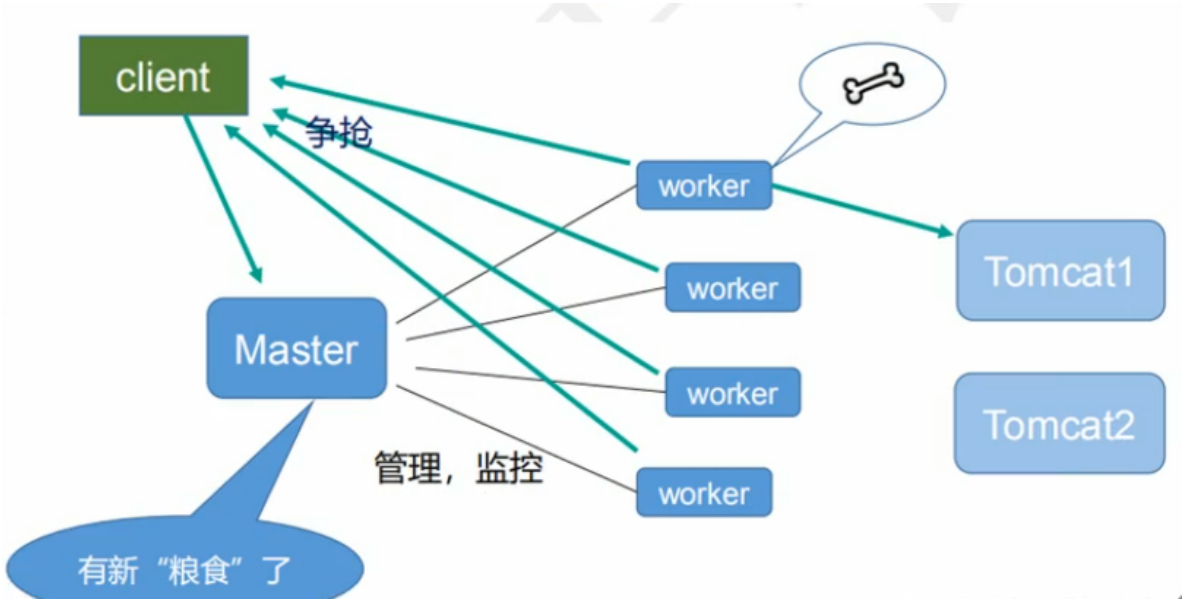
```
$ systemctl stop keepalived.service  #keepalived停止
```

七、Nginx原理解析

1、master和worker



2、worker如何进行工作的



3、一个master和多个woker的好处 (1) 可以使用nginx -s reload热部署。

首先，对于每个worker进程来说，独立的进程，不需要加锁，所以省掉了锁带来的开销，同时在编程以及问题查找时，也会方便很多。其次,采用独立的进程，可以让互相之间不会影响，一个进程退出后，其它进程还在工作，服务不会中断，master进程则很快启动新的worker进程。当然，worker进程的异常退出，肯定是程序有bug了，异常退出，会导致当前worker.上的所有请求失败，不过不会影响到所有请求，所以降低了风险。

4、设置多少个woker合适

```
Nginx同redis类似都采用了io多路复用机制，每个worker都是一个独立的进程， 但每个进程里只有一个主线程，通过异步非阻塞的方式来处理请求，即使是 千上万个请求也不在话下。每个worker的线程可以把一个cpu的性能发挥到极致。所以worker数和服务器的cpu数相等是最为适宜的。设少了会浪费cpu,设多了会造成cpu频繁切换上下文带来的损耗。
# 设置worker数量
worker.processes 4

# work绑定cpu(4work绑定4cpu)
worker_cpu_affinity 0001 0010 0100 1000

# work绑定cpu (4work绑定8cpu中的4个)
worker_cpu_affinity 0000001 00000010 00000100 00001000
```

5、连接数worker_ connection

```
这个值是表示每个worker进程所能建立连接的最大值，所以，一个nginx 能建立的最大连接数，应该是worker.connections * worker processes。当然，这里说的是最大连接数，对于HTTP 请求本地资源来说，能够支持的最大并发数量是worker.connections * worker processes,如果是支持http1.1的浏览器每次访问要占两个连接，所以普通的静态访问最大并发数是: worker.connections * 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worker.connections * worker_proceses/4. 因为作为反向代理服务器，每个并发会建立与客户端的连接和与后端服务的连接，会占用两个连接.
```

> 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? 答案：普通的静态访问最大并发数是: worker connections * 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worker connections * worker processes/4