nginx

Nginx的基本概念

1 介绍

nginx是一个高性能的HTTP和反向代码服务器

特点：

- 占有内存少
- 并发能力强

nginx专为性能优化而开发，能经受高负载的考验，最高可以支持50000个并发连接数

2 反向代理

1. 正向代理

```
用户上网的正向代理过程
    用户在局域网内是不能直接访问www.google.com的
    需要在客户端（浏览器）配置代理服务器的转发下才能访问

    用户  ---> 代理 ---> www.google.com
```

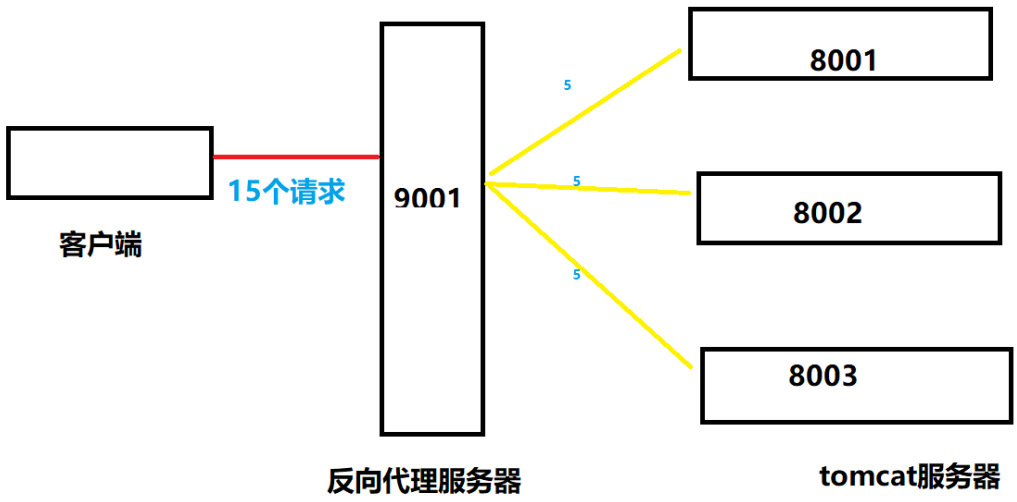
2. 反向代理

```
其实客户端对代理是没有任何感知的，因为客户端不需要任何的配置就可以访问，我们只需要将请求发送到反向代理服务器，再由反向代理服务器去选择目标服务器获取数据之后，在返回到客户端，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，真实的服务器ip地址被隐藏了

    用户访问的实际是8001，经过了9001的转发，到8001，获取数据到客户端
    用户  ---> 反向代理服务器（9001） ---> tomcat（8001）
```

负载均衡

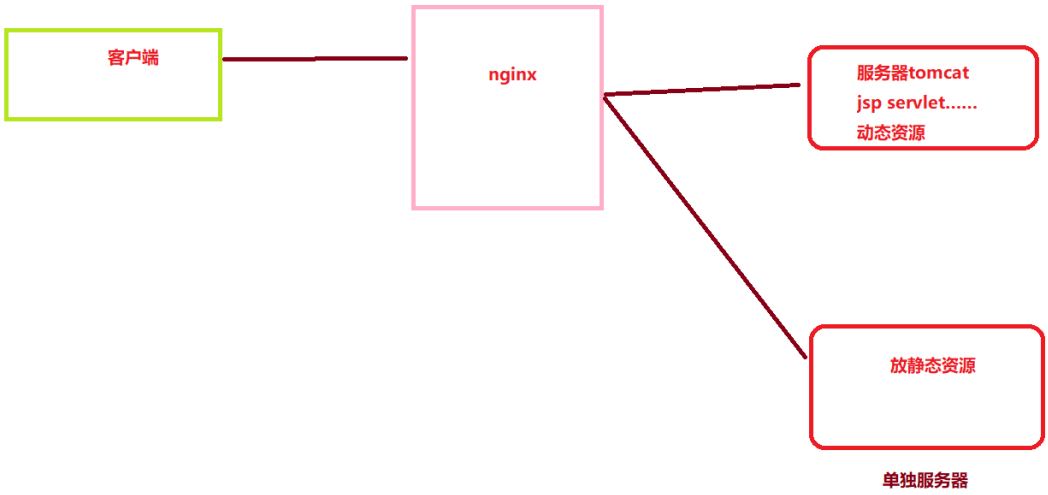
客户端发送多个请求到数据库，服务器处理请求的时候，可能要和数据库进行操作，服务器处理完毕后，再将结果返回到客户端

```
            (请求)
      --->   服务器   ---> 数据库
用户
      <---   服务器   <--- 数据库
            (响应)

这种架构模式适合于早期单一的系统，并发请求相对较少的前提下，成本较低
随着信息量，访问量和数据量的不断增长，以及系统业务的复杂程度增加，这种架构模式会造成服务器相应客户端的请求日益缓慢，并发量特别的的时候，还容易造成服务器直接崩溃，这是因为服务器性能的瓶颈造成的问题。

解决方式：负载均衡

增加服务器的数量，将请求发送到各个服务器，将原先的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，就是所说的负载均衡

```



动静分离

```
静态和动态资源分开进行部署

通过nginx可以做到
```



nginx的安装、常用命令和配置文件

在linux中安装nginx，

首先需要安装nginx需要的一些依赖

- pcre-8.37.tar.gz (也可以安装libpcre++0v5)
  上传到linux系统中，解压压缩文件
  进入解压之后的目录执行`./configure` 检查配置
  使用make && make install （编译并安装）
  安装之后使用pcre-config --version

```
root@ubuntu:/home/yuluo/war/pcre-8.45# pcre-config --version
8.45
```

- openssl-1.0.1t.tar.gz
- zlib.1.2.8.tar.gz
- nginx-1.21.6.tar.gz

```
解压缩进入目录 (tar --xvf)
执行.configure
执行make && make install

进入/usr/local/nginx/sbin
执行./nginx
启动nginx
然后ps查看nginx的进程信息
```

```
root@ubuntu:/usr/local/nginx/sbin# ps -ef | grep nginx
root       21721    1540  0 20:23 ?        00:00:00 nginx: master process ./nginx
nobody     21722   21721  0 20:23 ?        00:00:00 nginx: worker process
root       21772   14507  0 20:25 pts/0    00:00:00 grep --color=auto nginx
```

默认监听**80**端口

访问nginx：

在地址栏输入 http://ip 地址:80

nginx的常用操作命令

必须进入/usr/local/nginx/sbin目录中去

- 查看版本号 `./nginx -v`
- 启动nginx `./nginx`
- 关闭nginx `./nginx -s stop`
- 重加载 `./nginx -s reload`

nginx的配置文件

nginx的配置文件在**nginx**下的 `/usr/local/nginx/conf`

nginx的配置由三部分组成

1. 全局块：从配置文件开始到events块之间的内容，主要会设置一些影响nginx服务器整体运行的配置指令

```
比如：worker_process：值越大，可以支持的并发处理量越高
```

2. events：涉及的指令主要影响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 对nginx的性能影响较大，在实际中应该灵活配置
```

3. http块：配置最频繁的部分，代理缓存和日志定义等绝大多数功能和第三方模块的配置都在这里进行 **http**块包含**http**全局块，**server**块

   1. http块：指令包含文件引入，MIME-TYPE定义，日志自定义，连接超时，单链接请求数上限
   2. server块：和虚拟主机有密切关系，虚拟主机从用户角度看，和一台独立的硬件主机时完全一样的，该技术的出现是为了节省互联网服务器硬件成本

      包含全局**server**块和**location**块

      1. 全局server块：最常见的配置是本虚拟主机的监听配置和本虚拟主机的名称或者ip配置
      2. 一个server块可以配置多个locatiob块：对特定的请求进行处理，对虚拟主机名称之外字符串进行匹配。地址定

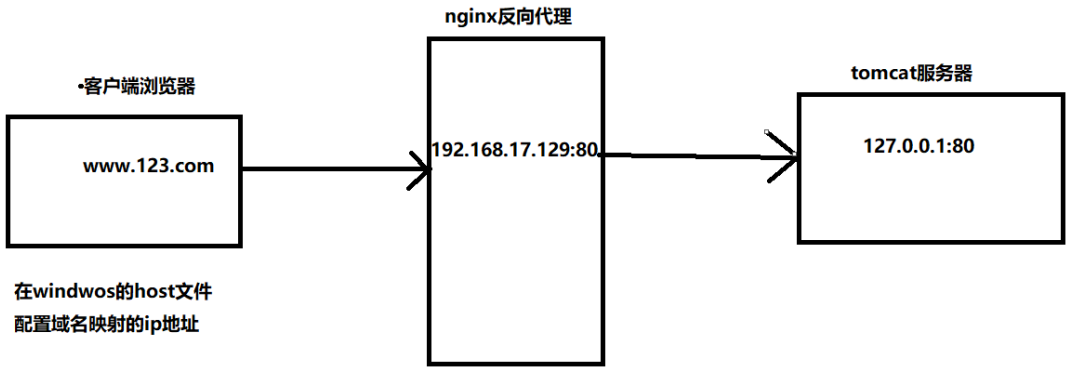
向，数据缓存……

nginx的配置实例 1-反向代理

反向代理实例1

实现效果：

打开浏览器，在浏览器输入地址 www.123.com ,跳转到linux系统中tomcat的主页面



具体实现

1. 在windows的host文件中添加ip地址和域名之间的映射关系

   路径：`c:windows/system32/drivrs/etc/HOSTS`

```
ip地址              域名
192.168.17.129   www.123.com
```

2. 在nginx中配置请求转发

```
server {
    listen: 80;
    server_name: 192.168.17.129;

    location / {
        root html
        proxy_pass  http://127.0.0.1:8080  转发的路径
        index index.html index.htm;
    }
}
```

访问**192.168.17.129**的时候nginx会转发到 **http://127.0.0.1:8080**

反向代理实例2

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**的监听端口为**9001**

访问 http://127.0.0.1:9001/edu 直接跳转到127.0.0.1:8080

访问 http://127.0.0.1:9001/vod 直接跳转到127.0.0.1:8082

具体实现

1. 修改nginx的配置文件

```
server {
    listen 9001;   监听9001端口
    server_name  192.168.17.129;

    location ~ /edu/ {   采用正则匹配的形式 路径中含有edu跳转到此路径
        proxy_pass  http://127.0.0.1:8080;
    }

    location ~ /vod/ {
        proxy_pass  http://127.0.0.1:8081;
    }
}
```

2. location中符号的说明:

1. =：用户不含正则表达式的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改请求
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后让请求字符串做匹配
5. 注意：如果uri包含正则表达式，则必须要有~或者 ~* 字样

nginx的配置实例2-负载均衡

实现效果

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

具体实现

```
http {

    upstream myserver {    起一个服务器名字
        server  192.168.17.129:8080;    加入服务器列表
        server  192.168.17.129:8081;
    }

    server {
        listen          50;
        server_name     192.168.17.129;
        location / {
            proxy_pass  http://myserver;    转发到myserver
        }
    }
}
```

浏览器访问 http://192.168.17.129/edu/a.html 时，就会被平均分配到8080和8081两个端口中去

nginx分配服务器的策略

1. 轮询（默认）：每个请求按照时间顺序逐一分配到不同的后端服务器中去，如果后端服务器down掉，能自动剔除
2. weight：weight表示权重，默认为1，权重越高被分配的客户端越多

```
    upstream myserver {    表示分配到8080的请求比8081多一倍
        server  192.168.17.129:8080  weight=10;
        server  192.168.17.129:8081  weight=5;
    }
```

3. ip哈希：每个请求访问的ip由hash结果分配，这样每个访问者可固定访问一个后端服务器，可以解决session共享的问题。

```
    upstream myserver {
        ip_hash;    表示用ip_hash的方式分配服务器
        server  192.168.17.129:8080;
        server  192.168.17.129:8081;
    }
```

4. fair：安装后端服务器的响应时间去分配，谁的响应时间短，先给谁分配

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

nginx的配置实例3-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设置参数，可以使浏览器缓存过期时间，减少与服务器之间的请求和流量。

具体的expires设置，是给一个资源设定一个过期时间，也就是说不需要去服务器验证，直接通过浏览器自身确认是否是过期文件，所以不会产生额外流量，此种方法非常适合不经常变动的资源。（如果经常变动的资源不建议使用expires来做缓存）

配置

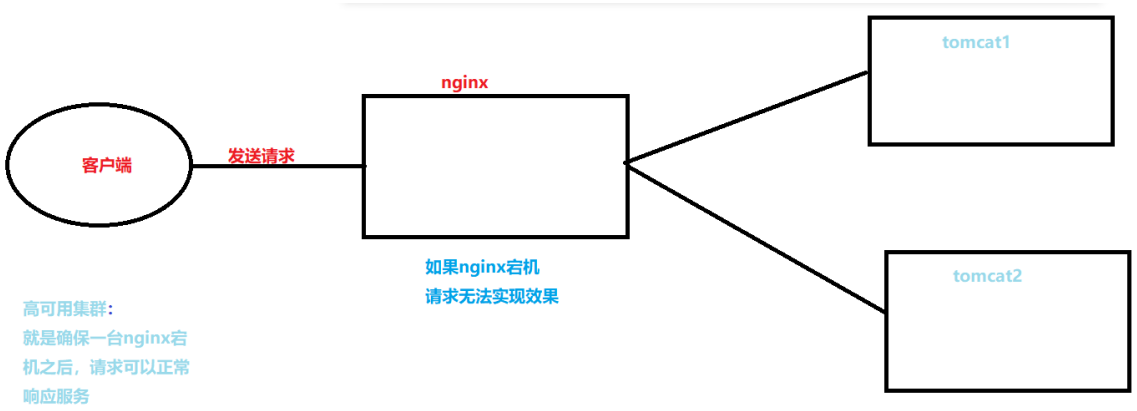
```
server {
    listen          80;
```

```
    servr_name      192.168.17.129;

    location /www {   配置一个访问的路径规则
        root    /data/;      设置资源位置
        index   index.html  index.htm;
    }

    location /image {
        root    /data/;
        autoindex on;        表示列出当前文件夹中的内容
    }
}
```

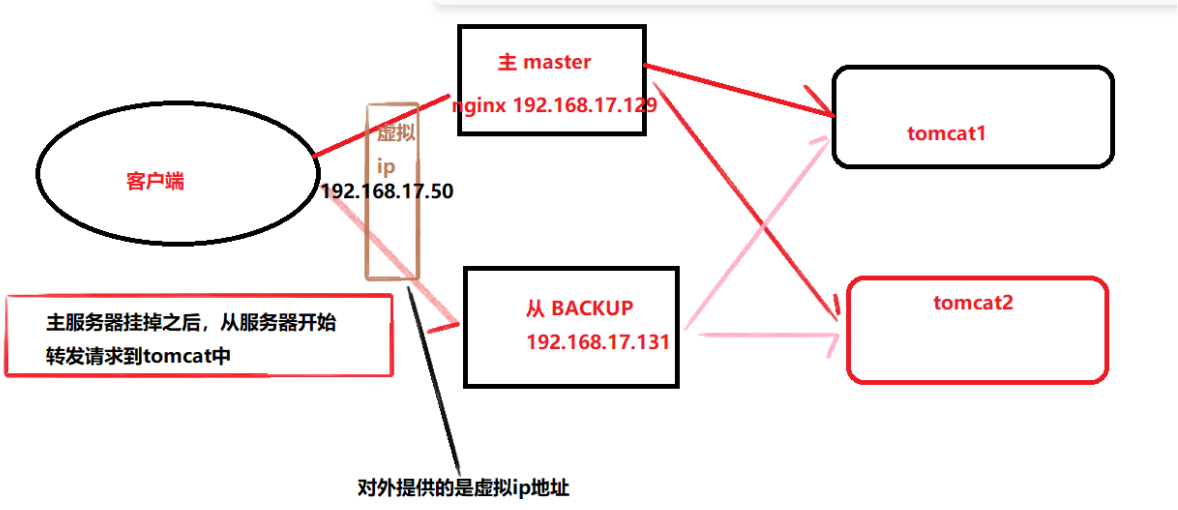
浏览器输入 http://192.168.17.129/image/ 和 http://192.168.17.129/www/a.html

nginx配置高可用集群

介绍



高可用的实现方式：



准备工作

1. 安装keepalived 监控nginx是否宕机的软件 安装位置`/etc/keepalived`

配置文件

1. 修改keepalived的配置文件
2. 在/usr/local/src/下面keepalived添加检测nginx的脚本

```
# 启动keepalived
systemctl keepalived service
```