

Nginx笔记

0. 阅读须知

本笔记来源：

- 博客：<https://heydingjeee.github.io/otherLanguage/Nginx%E5%A6%E4%B9%A0.html#%E5%89%8D%E6%8F%90%E5%87%86%E5%A4%87>
- 尚硅谷 nginx 课件
- 个人笔记以及修改 by 十一

1. 介绍

- Nginx开源版 <http://nginx.org/en/>

官方原始的Nginx版本

- Nginx plus商业版

开箱即用，集成了大量功能

- Open Resty <https://openresty.org/cn/>

OpenResty是一个基于 **Nginx与 Lua 的高性能 Web 平台**，其内部集成了大量精良的 Lua 库、第三方模块以及大多数的依赖项。**更适用于需要大量二次开发的场景，有极强的扩展性**

- Tengine <https://tengine.taobao.org/>

由淘宝网发起的Web服务器项目。它在**Nginx (opens new window)**的基础上，针对大访问量网站的需求，添加了很多高级功能和特性。Tengine的性能和稳定性已经在大型的网站如**淘宝网 (opens new window)**，**天猫商城 (opens new window)**等得到了很好的检验。相比于Open Resty，扩展性不够强，但是能够满足绝大多数使用场景

2. 安装

解压Nginx包，并安装

```
tar -zxvf nginx-1.21.6.tar.gz 解压到当前目录  
cd nginx-1.21.6 进入解压后的文件夹  
ls 文件夹中的文件: auto      CHANGES.ru  configure  html      man  
      src  CHANGES  conf           contrib    LICENSE  README
```

安装依赖库

```
# 安装C编译器  
yum install -y gcc  
  
# 安装pcre库  
yum install -y pcre pcre-devel  
  
# 安装zlib  
yum install -y zlib zlib-devel
```

```
# 使用prefix选项指定安装的目录  
./configure --prefix=/usr/local/nginx  
make  
make install
```

启动

```
cd /usr/local/nginx/sbin  
  
ls 里面是一个nginx的可执行文件  
  
. ./nginx 启动这个可执行
```

关闭防火墙

```
systemctl stop firewalld
```

补充Nginx命令

```
./nginx -s stop 快速停止  
./nginx -s quit 完成已接受的请求后，停止  
./nginx -s reload 重新加载配置  
./nginx -t 检查nginx配置是否正确
```

查看nginx状态

```
ps -ef|grep nginx
```

启动时：

```
[root@bogon ~]# ps -ef |grep nginx
root      1310      1  0 20:12 ?
nobody    1311  1310  0 20:12 ?
root      1492  1476  0 20:29 tty1      00:00:00 grep --color=auto nginx
```

停止时：

```
[root@bogon sbin]# ps -ef |grep nginx
root      1496  1476  0 20:30 tty1      00:00:00 grep --color=auto nginx
```

注册系统服务

通过系统服务的方式启动nginx

```
vim /usr/lib/systemd/system/nginx.service
```

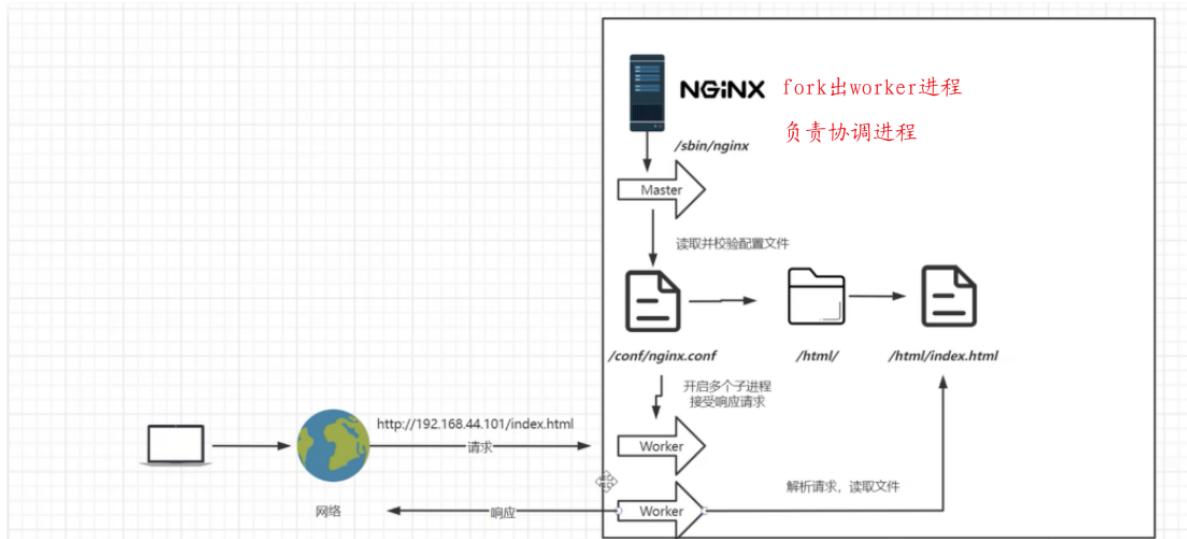
```
[Unit]
Description=nginx
After=network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
PIDFile=/usr/local/nginx/logs/nginx.pid
ExecStartPre=/usr/local/nginx/sbin/nginx -t -c
/usr/local/nginx/conf/nginx.conf
ExecStart=/usr/local/nginx/sbin/nginx -c
/usr/local/nginx/conf/nginx.conf
ExecReload=/usr/local/nginx/sbin/nginx -s reload
ExecStop=/usr/local/nginx/sbin/nginx -s stop
ExecQuit=/usr/local/nginx/sbin/nginx -s quit
PrivateTmp=true

[Install]
WantedBy=multi-user.target 多用户
```

```
# 重新加载系统服务
systemctl daemon-reload
# 启动服务
systemctl start nginx.service
# 开机启动
systemctl enable nginx.service
```

3. 基本运行原理



4. Nginx配置使用场景

4.1 Nginx目录

```
conf 配置文件
  | -nginx.conf 主配置文件
  | -其他配置文件，都被引入到了nginx.conf
html 静态页面（首页和错误页）
logs
  | -access.log 访问日志（记录用户访问的详细信息，有可能占很大空间，把磁盘写满了）
  | -error.log 错误日志
  | -nginx.pid 进程号
sbin
  | -nginx 执行文件
*_temp 运行时，生成临时文件
```

4.2 简化版的nginx.conf

```
worker_processes 1; # 默认为1，表示开启一个业务进程

events {
    worker_connections 1024; # 单个业务进程可接受连接数
}

http {
```

```
include          mime.types; # include是引入关键字，这里引入了
mime.types这个配置文件（同在conf目录下，mime.types是用来定义，请求返回
的content-type，也就是告诉浏览器返回的是什么类型的文件）
default_type   application/octet-stream; # 如果mime类型没匹配
上，默认使用二进制流的方式传输。

sendfile        on; # 高效网络传输，也就是数据0拷贝。
keepalive_timeout 65; # 长链接超时时间

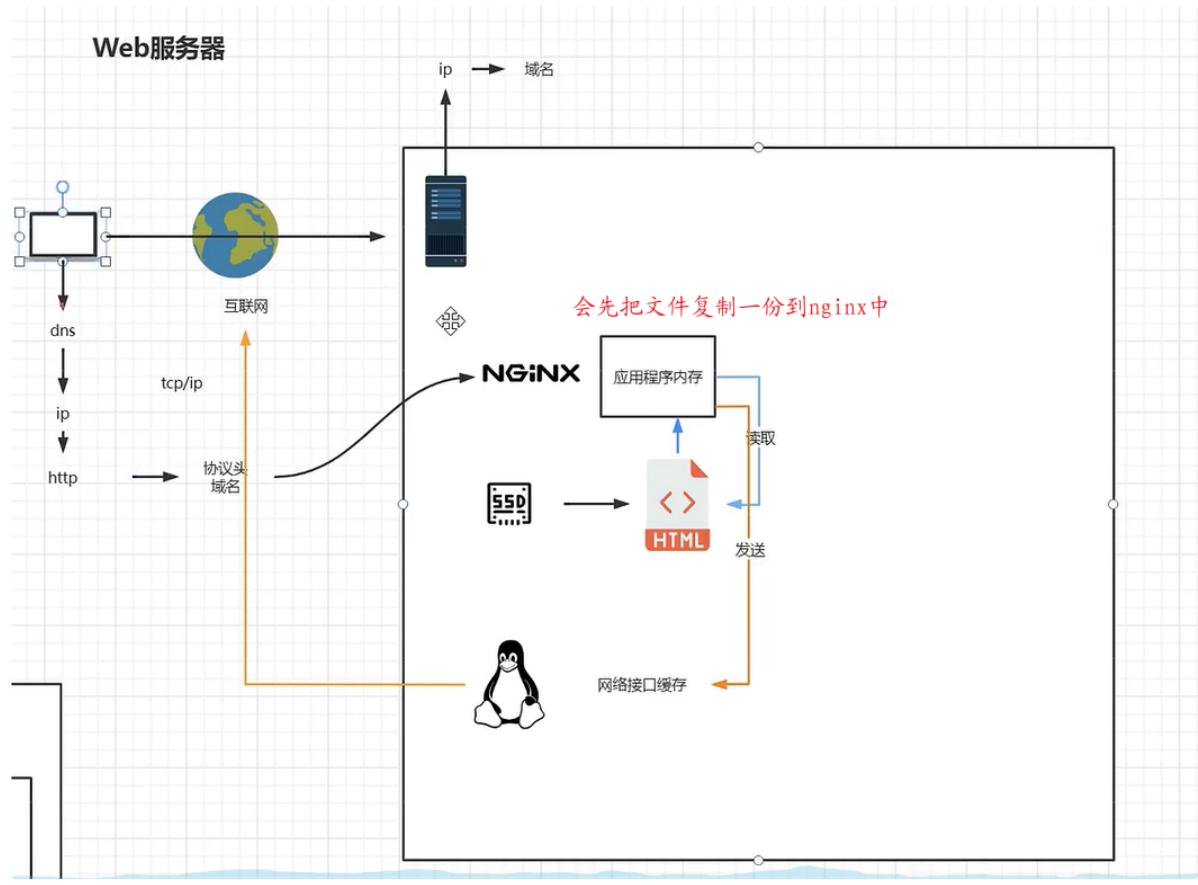
# 一个nginx可以启用多个server（虚拟主机 vhost）
server {
    listen        80; # 监听80端口
    server_name   localhost; # 接收的域名、主机名

    # 匹配路径
    location / {
        root      html; # 文件根目录
        index   index.html index.htm; # 默认页名称
    }

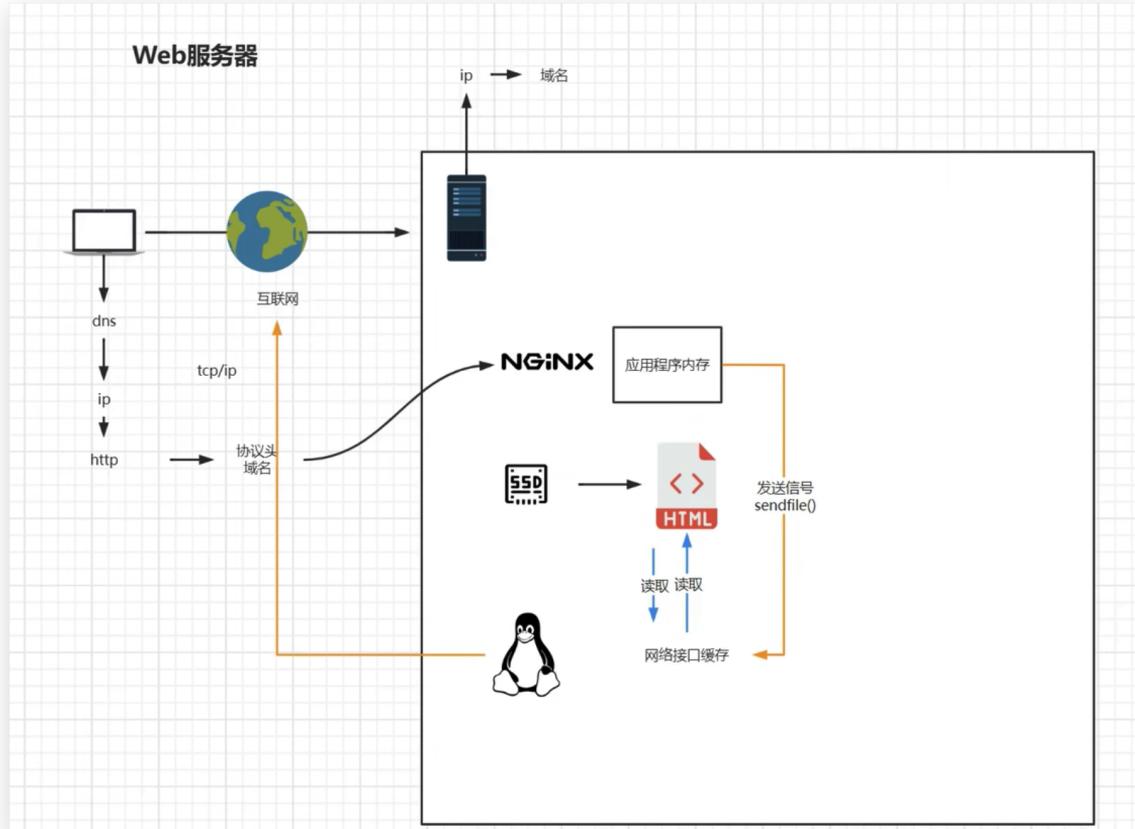
    error_page   500 502 503 504   /50x.html; # 报错编码对应
    页面
    location = /50x.html {
        root      html;
    }

}
}
```

不使用 sendfile



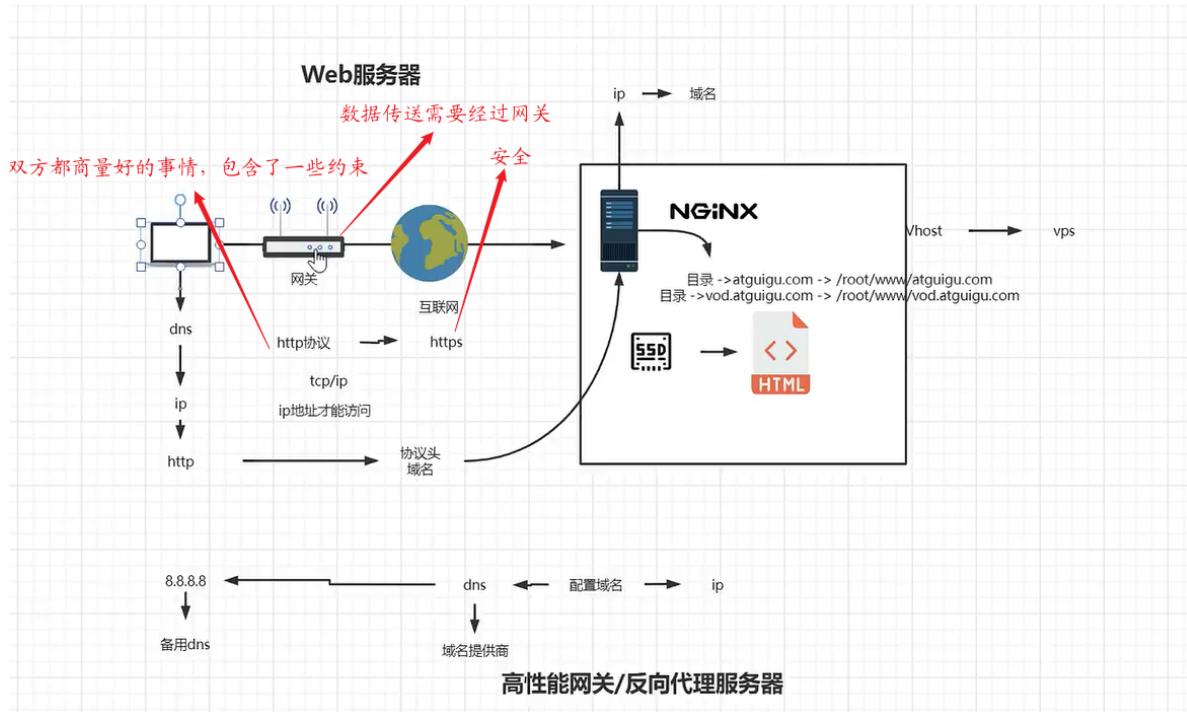
打开sendfile，用户请求的数据不用再加载到nginx的内存中，而是直接发送



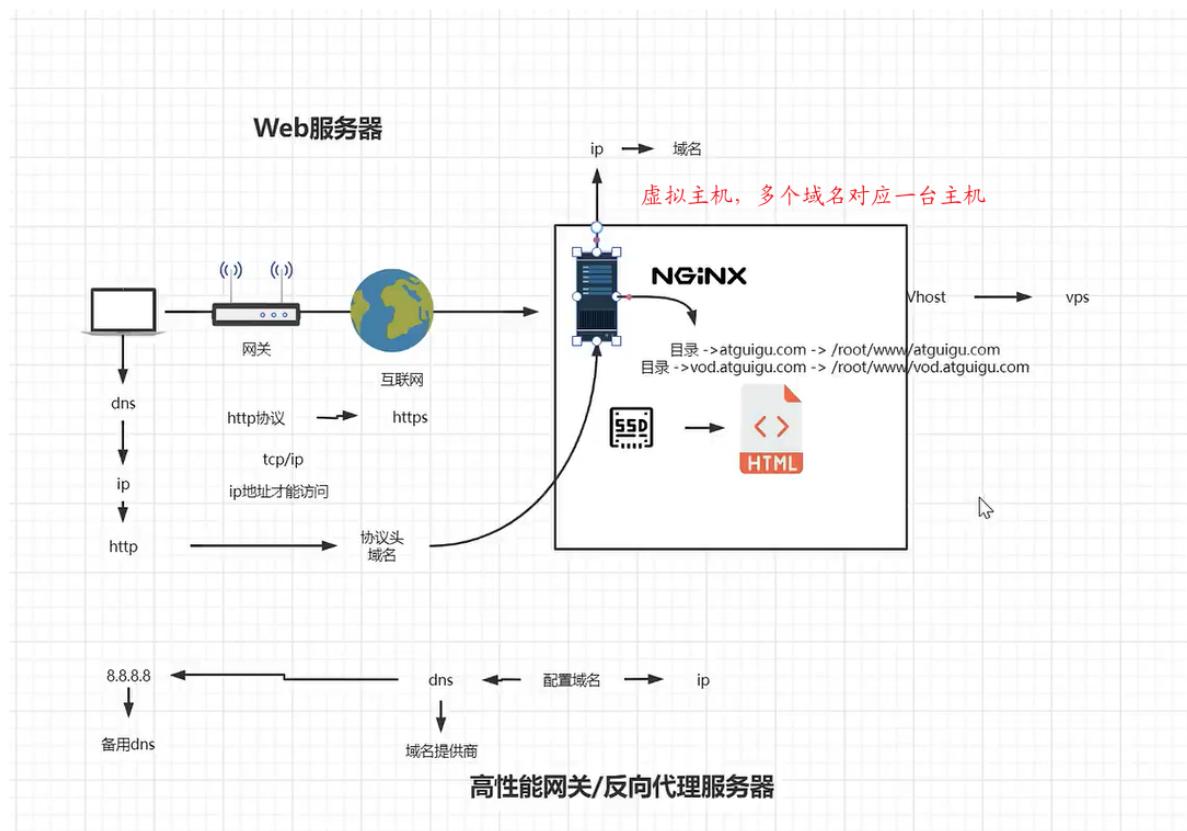
4.3 虚拟主机

原本一台服务器只能对应一个站点，通过虚拟主机技术可以虚拟化成多个站点同时对外提供服务

浏览器、Nginx与http协议



虚拟主机原理



修改nginx配置文件后，记得重新加载nginx

```
systemctl reload nginx
```

```
http {

    ...

    server {
        listen      80;
        server_name www.tt110617.xyz;

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

        error_page   500 502 503 504   /50x.html;
        location = /50x.html {
            root  html;
        }
    }

    server {
        listen      88;
        server_name vod.tt110617.xyz;

        location /www/vod {
            root  html;
            index index.html index.htm;
        }

        error_page   500 502 503 504   /50x.html;
        location = /50x.html {
            root  html;
        }
    }
}
```

1. servername匹配规则

我们需要注意的是servername匹配分先后顺序，写在前面的匹配上就不会继续往下匹配了。

完整匹配

我们可以在同一servername中匹配多个域名

```
server_name vod.mmban.com www1.mmban.com;
```

通配符匹配

```
server_name *.mmban.com
```

通配符结束匹配

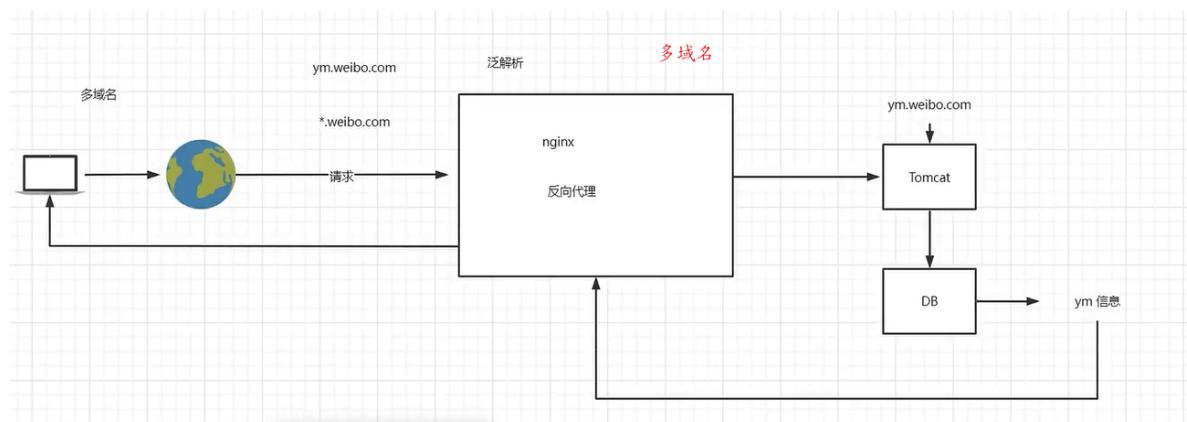
```
server_name vod.*;
```

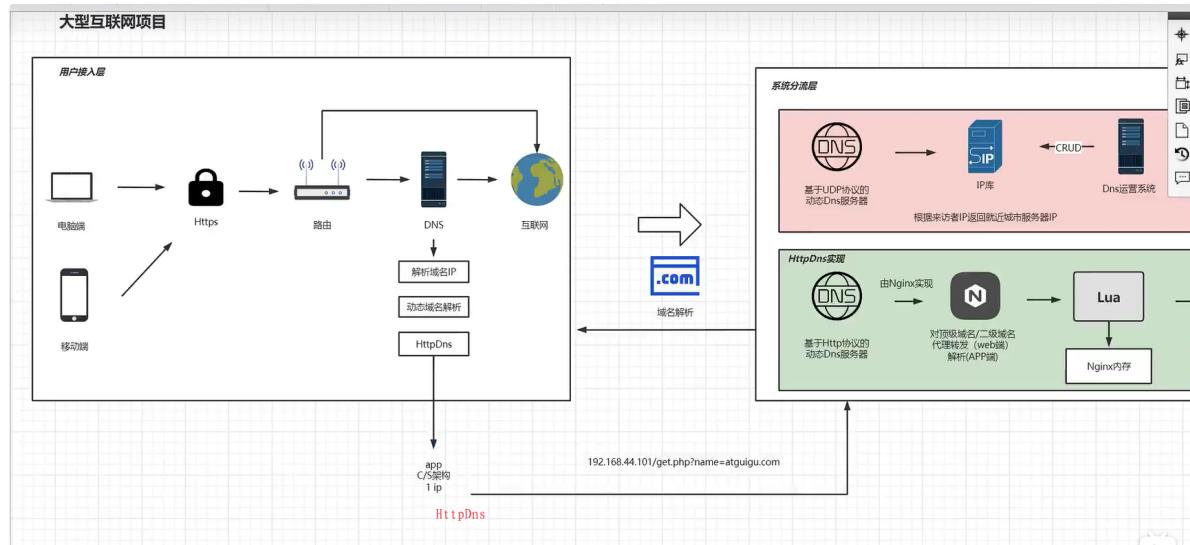
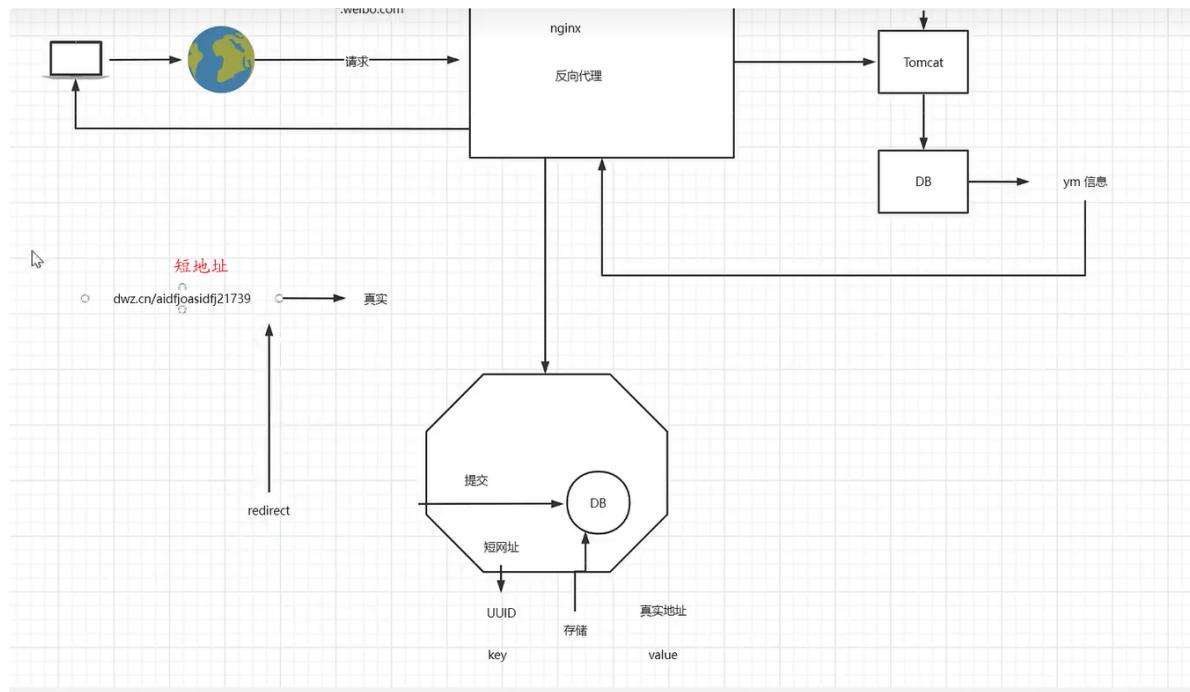
正则匹配

```
server_name ~^ [0-9]+ \.mmban \.com$;
```

2. 域名解析相关企业项目实战技术架构

- 多用户二级域名
- 短网址
- httpdns

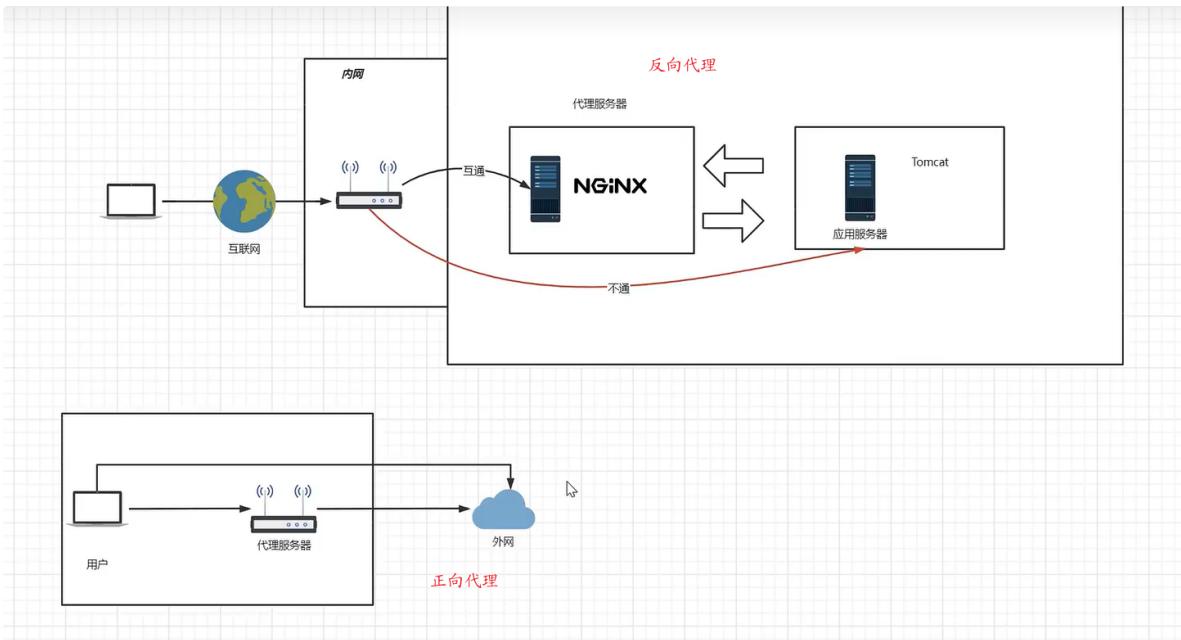




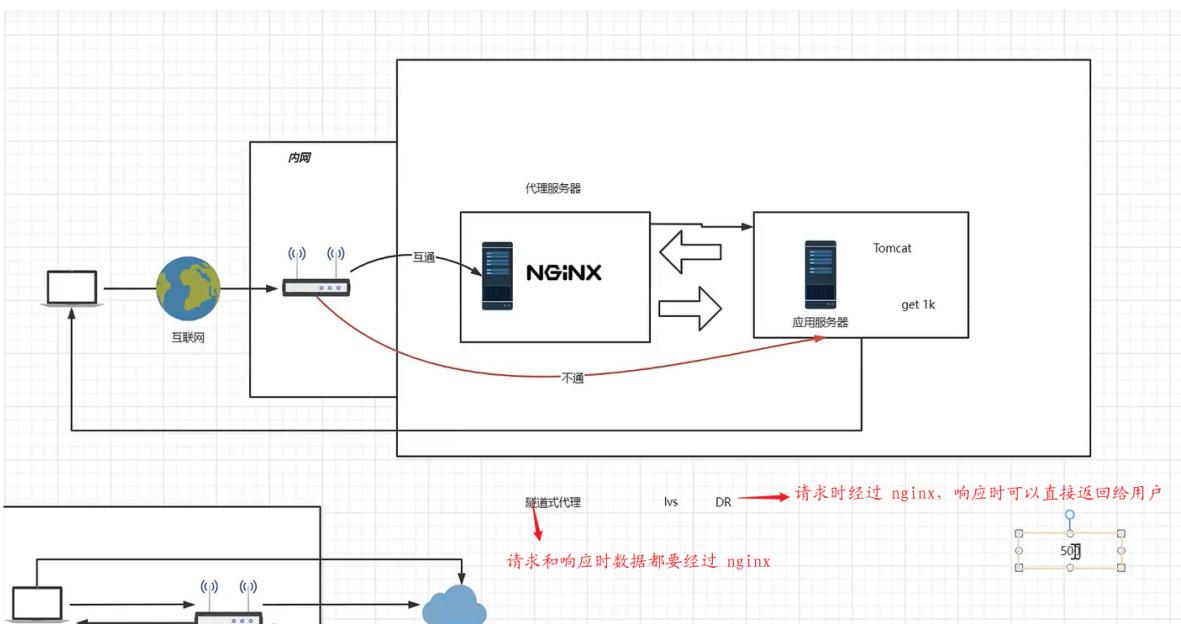
5. 反向代理与负载均衡

5.1 网关、代理与反向代理

反向代理和正向代理

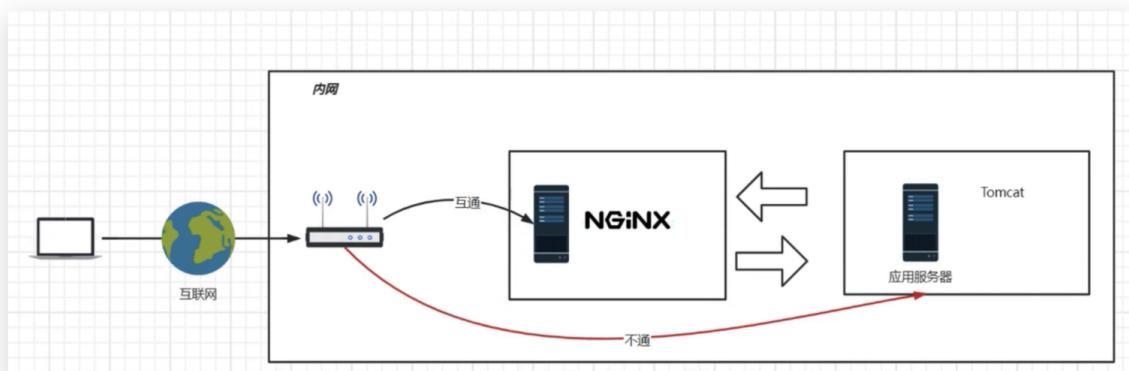


隧道式代理 和 DR



5.2 反向代理

反向代理：这种代理方式叫做，隧道代理。有性能瓶颈，因为所有的数据都经过Nginx，所以Nginx服务器的性能至关重要



nginx.conf配置文件

- 启用proxy_pass, root和index字段就会失效
- proxy_pass后的地址必须写完整 http://xxx; , 不支持https
- 当访问localhost时 (Nginx服务器) , proxy_pass
http://www.atguigu.com; 是请求转发, 浏览器地址还是原来的地址
- 当访问localhost时 (Nginx服务器) , proxy_pass http://atguigu.com;
是重定向, 响应码是 302 , 浏览器地址是 新地址

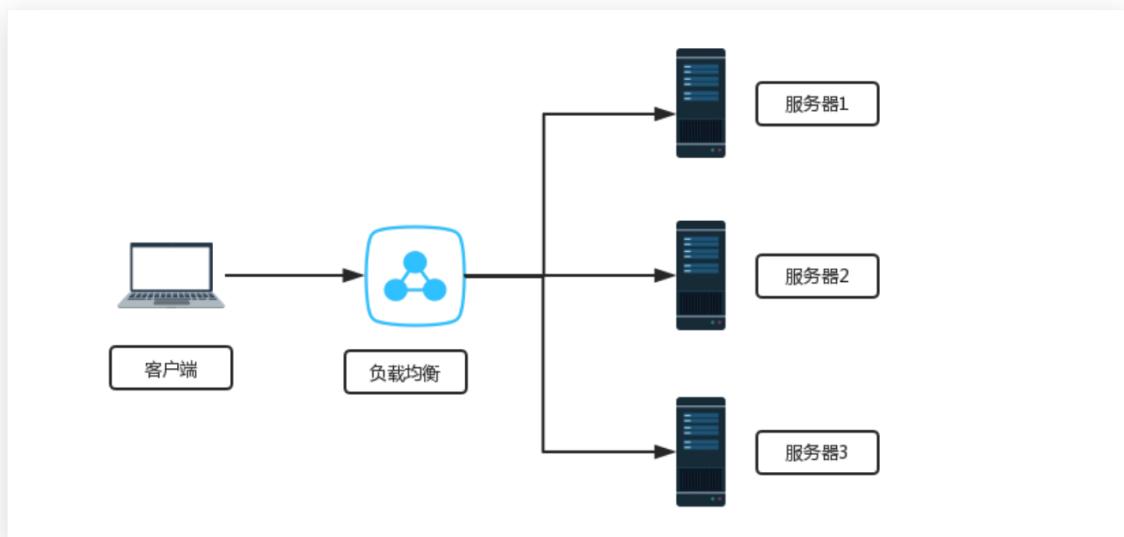
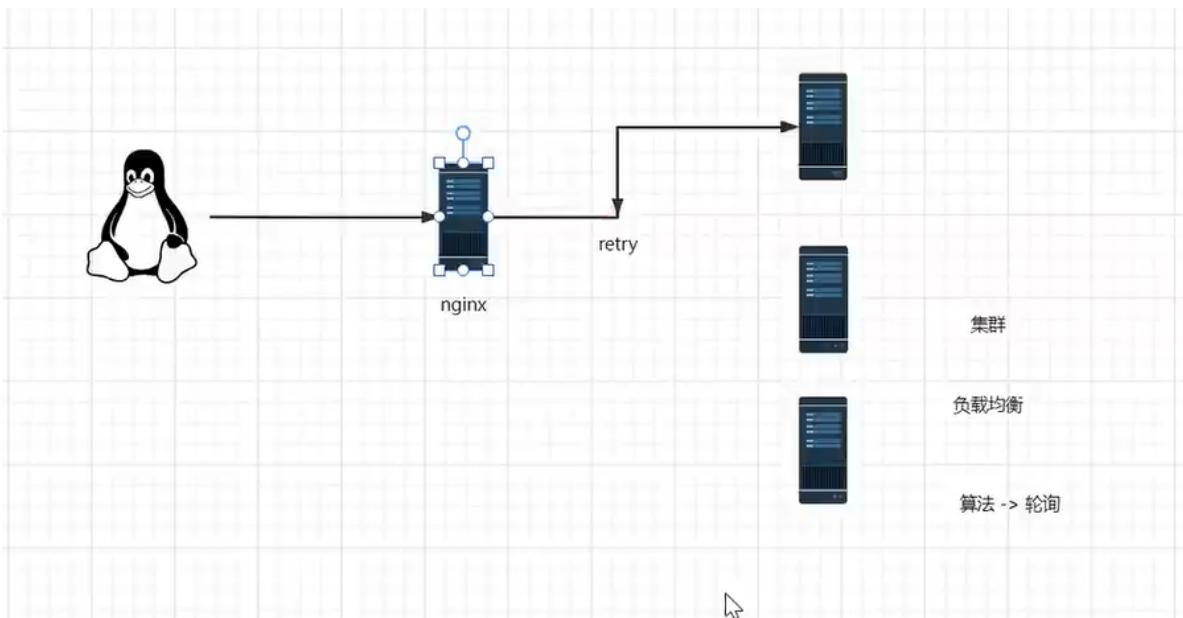
```
http{
    server {
        listen      80;
        server_name localhost;

        location / {
            proxy_pass http://xxx;
            # 启用proxy_pass, root和index字段就会失效
            # root      html/test;
            # index    index.html index.htm;
        }

        error_page   500 502 503 504   /50x.html;
        location = /50x.html {
            root      html;
        }
    }
}
```

5.3 负载均衡策略

负载均衡: 把请求, 按照一定算法规则, 分配给多台业务服务器 (即使其中一个坏了/维护升级, 还有其他服务器可以继续提供服务)



使用upstream定义一组地址【在server字段下】

访问localhost，访问都会代理到192.168.174.133:80和192.168.174.134:80这两个地址之一，每次访问这两个地址轮着切换 轮询

```
http{
    upstream httpds{
        server 192.168.174.133:80;  # 如果是80端口，可以省略不写
        server 192.168.174.134:80;
    }
    server {
        ...
        location / {
            proxy_pass http://httpds;
        }
    }
}
```

```
    ...
}
}
```

轮询

默认情况下使用轮询方式，逐一转发，这种方式适用于无状态请求。

weight(权重)

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。

```
upstream httpds{
    server 192.168.174.133:80 weight=10;
    server 192.168.174.134:80 weight=80;
}
```

关闭

```
upstream httpds{
    server 192.168.174.133:80 weight=10 down;
    server 192.168.174.134:80 weight=80;
}
```

备用机

如果 **192.168.174.133:80** 出现故障，无法提供服务，就用使用**backup**的这个机器

```
upstream httpds{
    server 192.168.174.133:80 weight=10;
    server 192.168.174.134:80 weight=80 backup;
}
```

ip_hash

根据客户端的ip地址转发同一台服务器。

least_conn

最少连接访问

url_hash

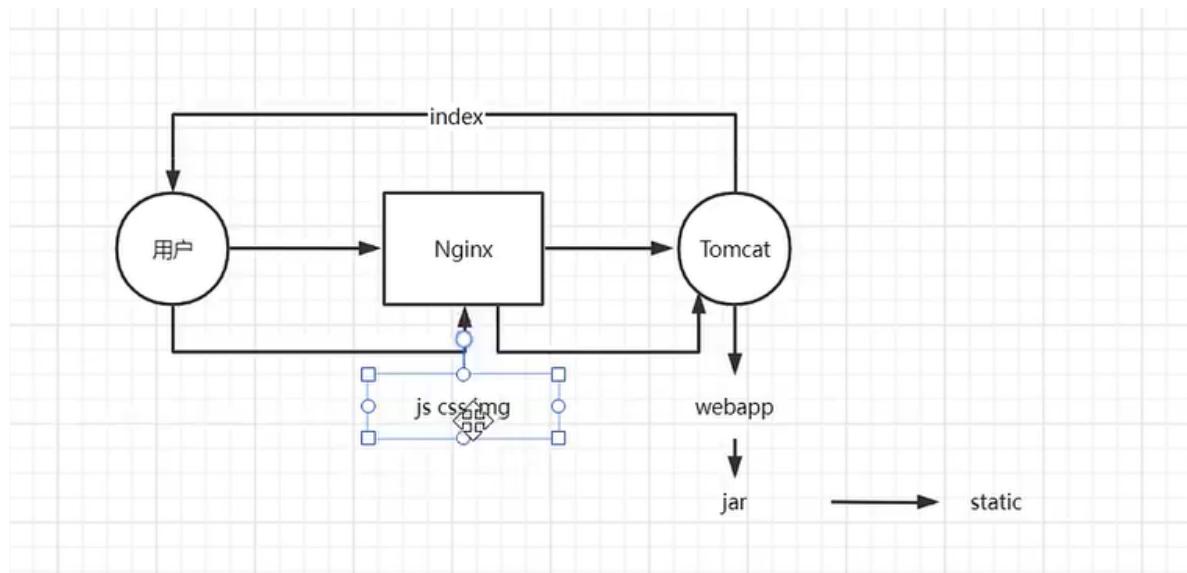
根据用户访问的url定向转发请求

fair

根据后端服务器响应时间转发请求

5.4 动静分离

当用户请求时，动态请求分配到Tomcat业务服务器，静态资源请求放在Nginx服务器中



例子：

- 如果请求的资源地址是 `location /`, `/` 的优先级比较低, 如果下面的 `location` 没匹配到, 就会走 `http://xxx` 这个地址的机器
- 如果请求的资源地址是 `location/css/*`, 就会被匹配到nginx的html目录下的css文件夹中 (我们把css静态资源放在这个位置)

```
server {  
    listen      80;  
    server_name localhost;  
  
    # /的优先级比较低, 如果下面的location没匹配到, 就会走  
    # http://127.0.0.1:8080 这个地址的机器  
    location / {  
        proxy_pass http://127.0.0.1:8080;  
    }  
  
    # root指的是html, location/css指的是root下的css, 所以就是  
    # html/css  
    location /css {
```

```
root html;
index index.html index.htm;
}

error_page 500 502 503 504 /50x.html;
location = /50x.html {
    root html;
}
}
```

使用正则

```
location ~*/(js|css|img){
    root html;
    index index.html index.htm;
}
```

5.5 URL重写

rewrite是URL重写的关键指令，根据regex（正则表达式）部分内容，重定向到replacement，结尾是flag标记。

rewrite <regex> <replacement> [flag];
关键字 正则 替代内容 flag标记

正则：匹配正则表达式语句进行规则匹配

替代内容：将正则匹配的内容替换成replacement

flag标记说明：

last # 本条规则匹配完成后，继续向下匹配新的location URI规则

break # 本条规则匹配完成即终止，不再匹配后面的任何规则

redirect # 返回302临时重定向，浏览器地址会显示跳转后的URL地址

permanent # 返回301永久重定向，浏览器地址栏会显示跳转后的URL地址

浏览器地址栏访问 **xxx/123.html** 实际上是访问 **xxx/index.jsp?pageNum=123**

```
server {
    listen 80;
    server_name localhost;

    location / {
        rewrite ^/([0-9]+).html$ /index.jsp?pageNum=$1 break;
    }
}
```

```
proxy_pass http://xxx;  
}  
  
error_page 500 502 503 504 /50x.html;  
location = /50x.html {  
    root html;  
}  
}
```

5.6 网关服务器



上图中，应用服务器，不能直接被外网访问到，**只能通过Nginx服务器进行访问（使用 proxy_pass）**，这时候这台Nginx服务器就成为了网关服务器（承担入口的功能）

所以，我们启动应用服务器的防火墙，设置其只能接受这台Nginx服务器的请求

开启防火墙

```
systemctl start firewalld
```

添加rich规则

```
# 这里的192.168.13.20是网关服务器地址  
firewall-cmd --permanent --add-rich-rule="rule family="ipv4"  
source address="192.168.13.20" port protocol="tcp" port="8080"  
accept"
```

移除rich规则

```
firewall-cmd --permanent --remove-rich-rule="rule family="ipv4"  
source address="192.168.13.20" port port="8080" protocol="tcp"  
accept"
```

重启

移除和添加规则都要重启才能生效

```
firewall-cmd --reload
```

查看所有规则

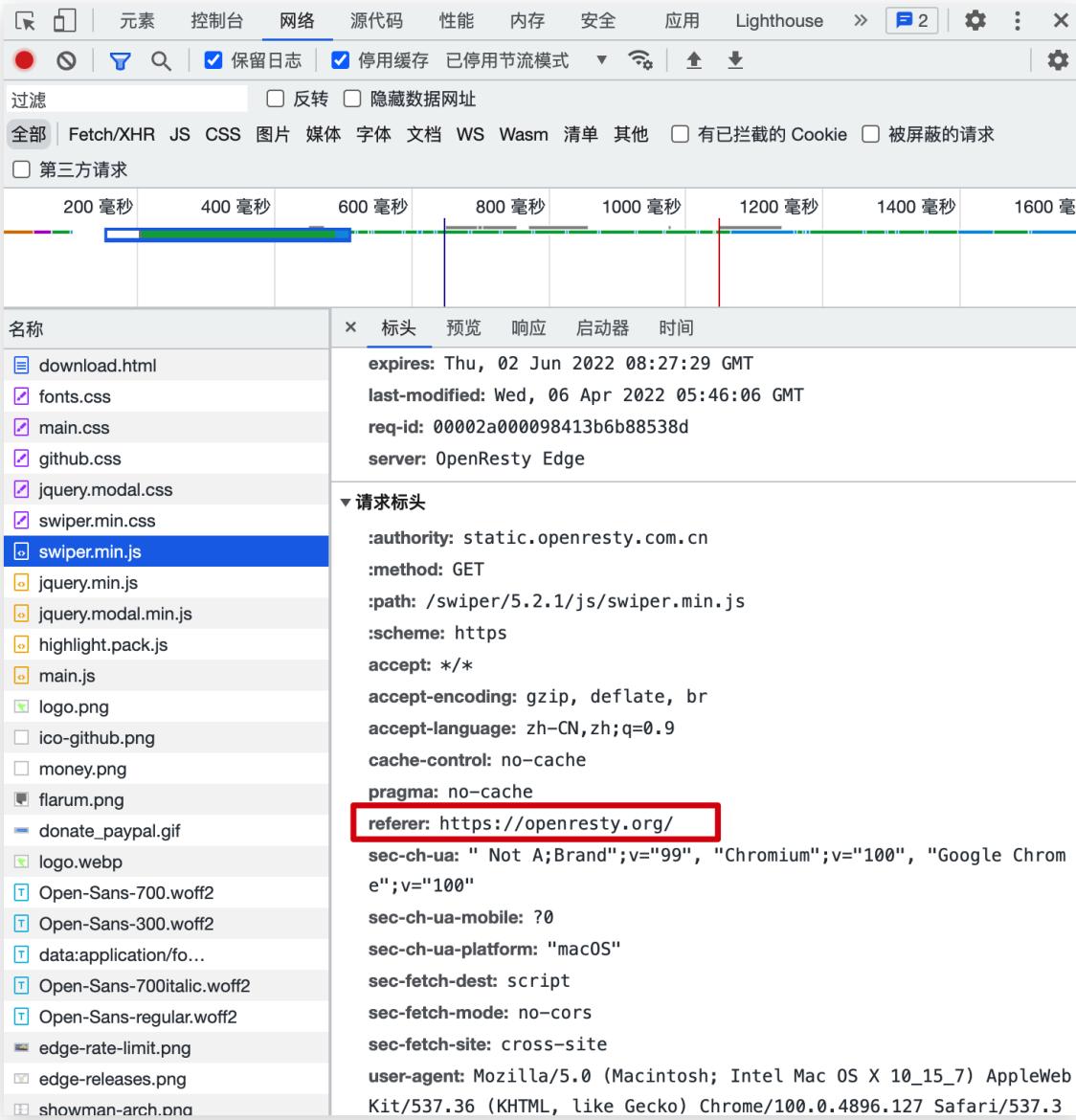
```
# 所有开启的规则  
firewall-cmd --list-all
```

网关配置

```
upstream httpds {  
    server 192.168.44.102 weight=8 down;  
    server 192.168.44.103:8080 weight=2;  
    server 192.168.44.104:8080 weight=1 backup;  
}  
location / {  
    rewrite ^/([0-9]+).html$ /index.jsp?pageNum=$1 redirect;  
    proxy_pass http://httpds ;  
}
```

6. 防盗链

当我们请求到一个页面后，这个页面一般会再去请求其中的静态资源，这时候请求头中，会有一个refer字段，表示当前这个请求的来源，我们可以限制指定来源的请求才返回，否则就不返回，这样可以节省资源



防盗链配置

```
valid_referers none | blocked | server_names | strings ....;
```

- `none`, 检测 `Referer` 头域不存在的情况。
- `blocked`, 检测 `Referer` 头域的值被防火墙或者代理服务器删除或伪装的情况。这种情况该头域的值不以“`http://`”或“`https://`”开头。
- `server_names`, 设置一个或多个 URL , 检测 `Referer` 头域的值是否是这些 URL 中的某一个。

注意: `if ($invalid_referer)` 中if后有个空格, 不写就会报错

```
nginx: [emerg] unknown directive "if($invalid_referer)" in
/usr/local/nginx/conf/nginx.conf:27
```

例子：这里设置nginx服务器中的img目录下的图片必须refer为 192.168.174.133 才能访问

```
server {
    listen      80;
    server_name localhost;

    location / {
        proxy_pass http://xxx;
    }

    location /img{
        valid_referers 192.168.174.133;
        if ($invalid_referer){ # 无效的
            return 403; # 返回状态码403
        }
        root html;
        index index.html index.htm;
    }

    error_page   500 502 503 504   /50x.html;
    location = /50x.html {
        root   html;
    }
}
```

如果引用这张图片的页面且refer并没有被设置，图片无法加载出来

如果直接访问图片地址，因为没有refer字段指向来源，会直接显示Nginx的页面

403 Forbidden

nginx/1.21.6

使用curl测试

```
curl -I http://192.168.44.101/img/logo.png
```

带引用

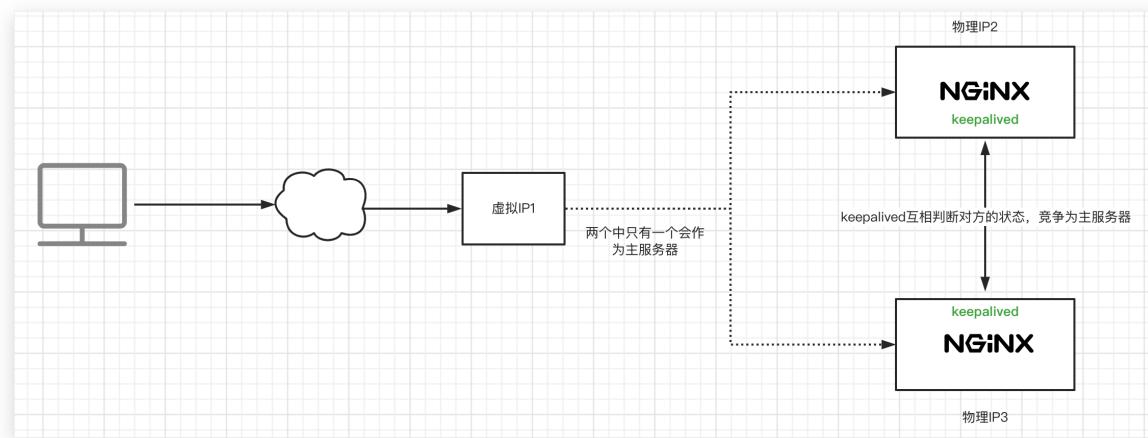
```
curl -e "http://baidu.com" -I
http://192.168.44.101/img/logo.png
```

设置盗链图片

将提示图片放在html/img/x.png，访问设置防盗链图片时，就返回这x.png张图

```
location /img{  
    valid_referers 192.168.174/133;  
    if ($invalid_referer){ # 无效的  
        rewrite ^/ /img/x.png break;  
    }  
    root html;  
    index index.html index.htm;  
}
```

7. 高可用场景-keepalived



用户访问时，访问的是一个虚拟IP，keepalived会选定一个主服务器使用这个虚拟IP，当服务宕机后，虚拟IP会移动到新的 master 节点

每台机器上的keepalived会相互通信，根据其他机器上的keepalived进程是否存在，判断服务器状态，如果默认的Master停止了，就会在剩下的Backup机器中，竞选出一台Nginx服务器作为Master

安装keepalived

```
yum install -y keepalived
```

修改keepalived配置

- 配置文件在 `/etc/keepalived/keepalived.conf`
- `vrrp_instance`、`authentication`、`virtual_router_id`、`virtual_ipaddress` 这几个一样的机器，才算是同一个组里。这个组才会选出一个作为Master机器

这里我们设置两台机器，分别下载好keepalived，然后进行配置

机器一：

```
! Configuration File for keepalived

global_defs {
    router_id lb1  # 名字与其他配置了keepalive的机器不重复就行
}

vrrp_instance heydingjie { # vrrp实例名可以随意取
    state MASTER # 只能有一个默认的Master, 其他写BACKUP
    interface ens33 # ip addr查看下网卡名, 默认时ens33
    virtual_router_id 51
    priority 100 # 多台安装了keepalived的机器竞争成为Master的优先级
    advert_int 1 # 通信时间
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.200.16 # 虚拟IP
    }
}
```

机器二：

```
! Configuration File for keepalived

global_defs {
    router_id lb2
}

vrrp_instance heydingjie {
    state BACKUP # 只能有一个默认的Master, 其他写BACKUP
    interface ens33
    virtual_router_id 51
    priority 50
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.200.16 # 虚拟IP
    }
}
```

```
}
```

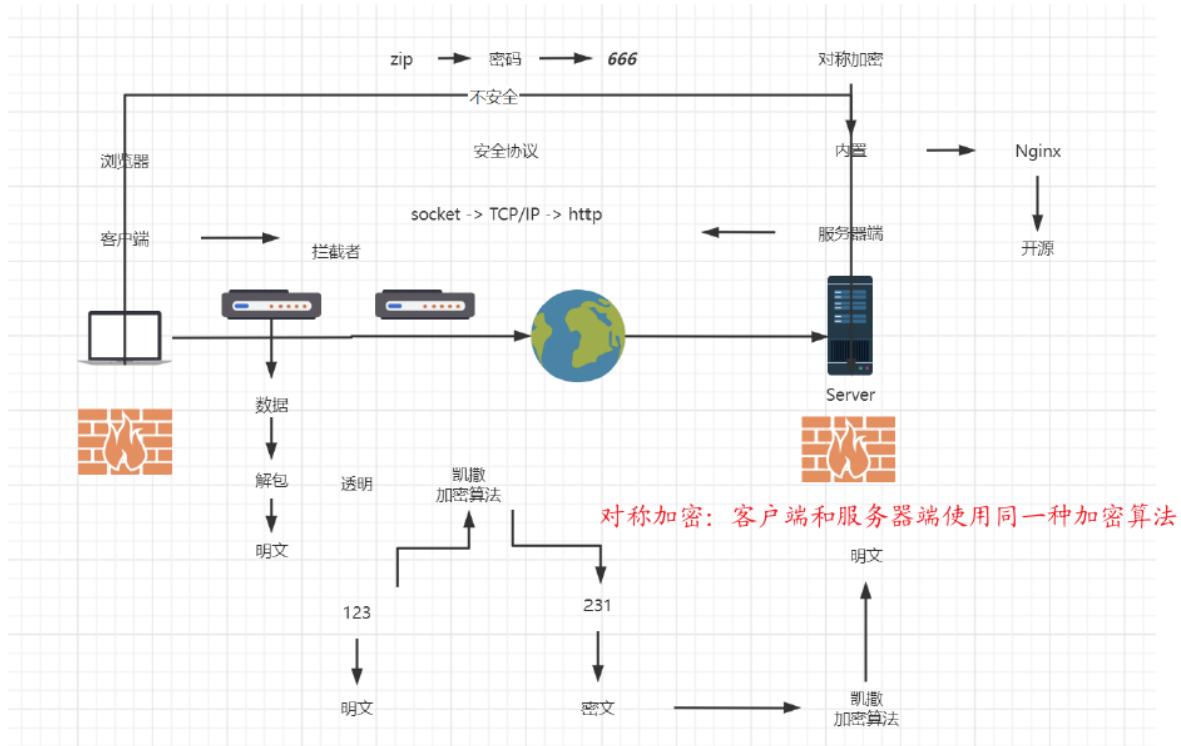
通过命令 `ip addr` 查看机器一的ip信息，可以看到虚拟IP

```
[root@bogon keepalived]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:f6:32:04 brd ff:ff:ff:ff:ff:ff
    inet 192.168.174.135/24 brd 192.168.174.255 scope global noprefixroute dynamic ens33
        valid_lft 1738sec preferred_lft 1738sec
    inet 192.168.200.16/32 scope global ens33
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fed6:3204/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

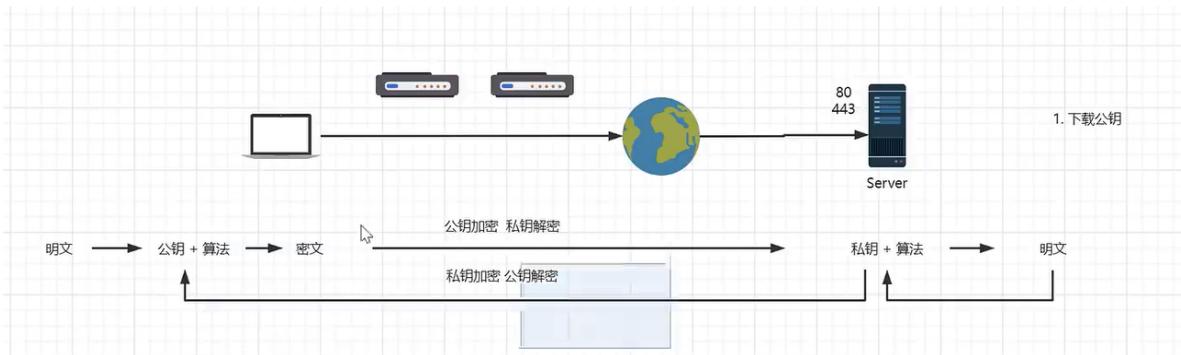
8. 配置证书

8.1 不安全的http协议

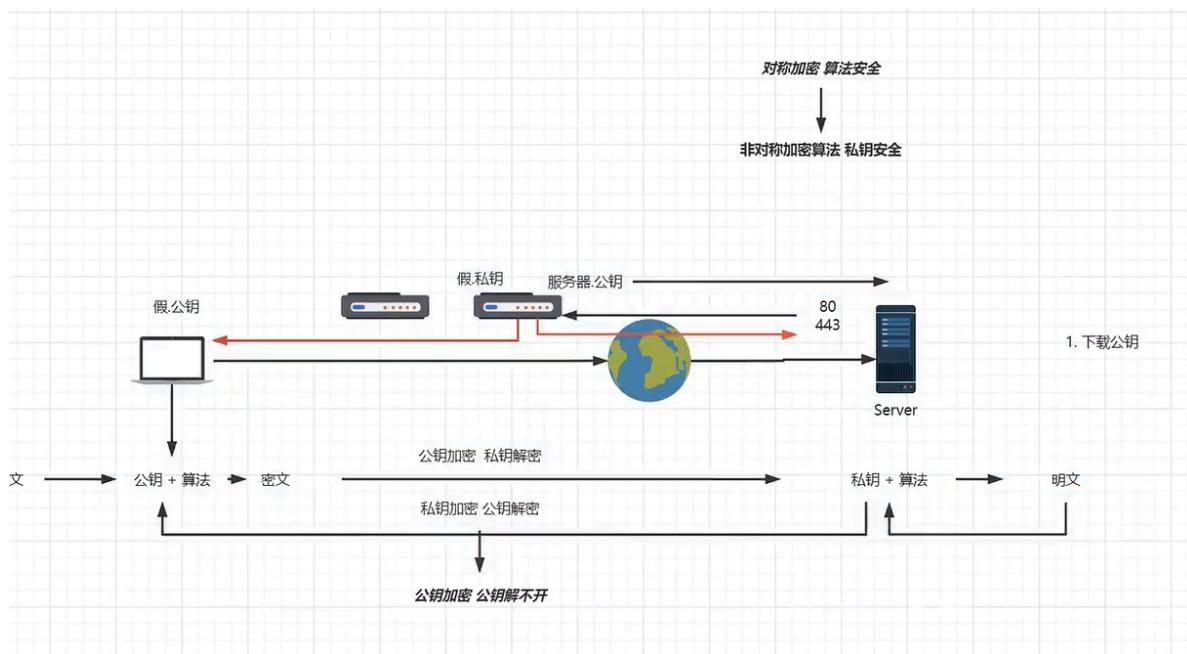
对称算法



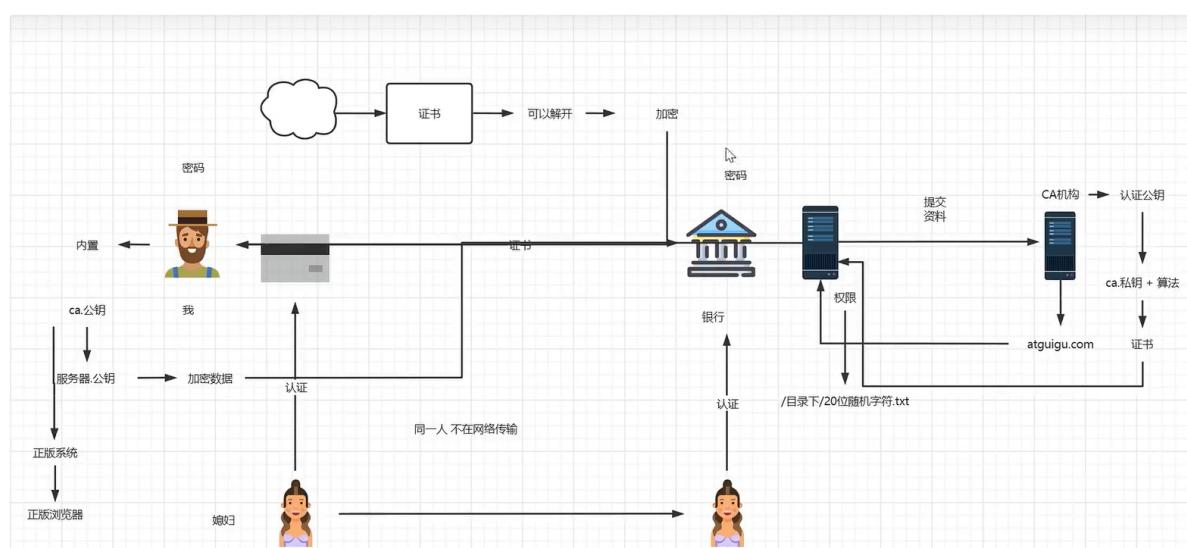
8.2 非对称算法



非对称加密算法同样不安全



解决：CA机构对服务器公钥进行非对称加密（CA.私钥 + 算法），生成证书，下发表证书，黑客可以解开证书，但是没法用CA.私钥再次进行加密，保证了安全性。我们的操作系统内置了CA.公钥，**可以解开证书，得到服务器公钥**，这样就**保证了服务器公钥的安全性**



购买服务器—>购买域名，并解析到这个主机—>购买证书，绑定到域名上，并且把证书文件安装到服务器，并在Nginx上配置好

这时候，这个域名就可以使用https进行访问了（<https://xxxx>），浏览器上会有一个小锁



上面的流程我比较熟悉了，就直接跳过了，这里直接写申请到证书后的Nginx配置部分

下载证书文件

A screenshot of the Alibaba Cloud SSL Certificate Management Service interface. The left sidebar shows 'SSL 证书' selected. The main area displays a table of certificates. One row is highlighted, showing details: 'cert-7330124', 'DigiCert 免费版 SSL', 'RSA', '已签发', 'hedaodao.ltd', 'www.hedaodao.ltd', '1年', '2023-05-03'. To the right of the table, there are '部署' (Deploy), '续费' (Renew), and '下载' (Download) buttons, with a red arrow pointing to the 'Download' button.

证书下载

X

请根据您的服务器类型选择证书下载：

服务器类型	操作
Tomcat	帮助 下载
Apache	帮助 下载
Nginx	帮助 下载
IIS	帮助 下载
JKS	帮助 下载
其他	下载
根证书下载	下载

网站代理HTTPS服务

不用安装证书，不用纠结各种安全套件的选择，不用担心私钥泄露，网站代理HTTPS服务帮助 [立即使用](#) 您解决网站HTTPS问题。



下载后，解压压缩包，可以看到两个文件，一个是 `xxx.key` (私钥) 和 `xxx.pem` (证书)

配置

将两个文件上传到Nginx目录中，记得放置的位置。我这里直接放在nginx.conf配置文件所在的目录 (`/user/local/nginx/conf`)，所以写的都是相对路径

```
server {
    listen 443 ss1;
    # 这里是证书路径
    ss1 certificate xxx.pem;
    # 这里是私钥路径
    ss1_certificate_key xxx.key
}
```