Nginx 学习笔记

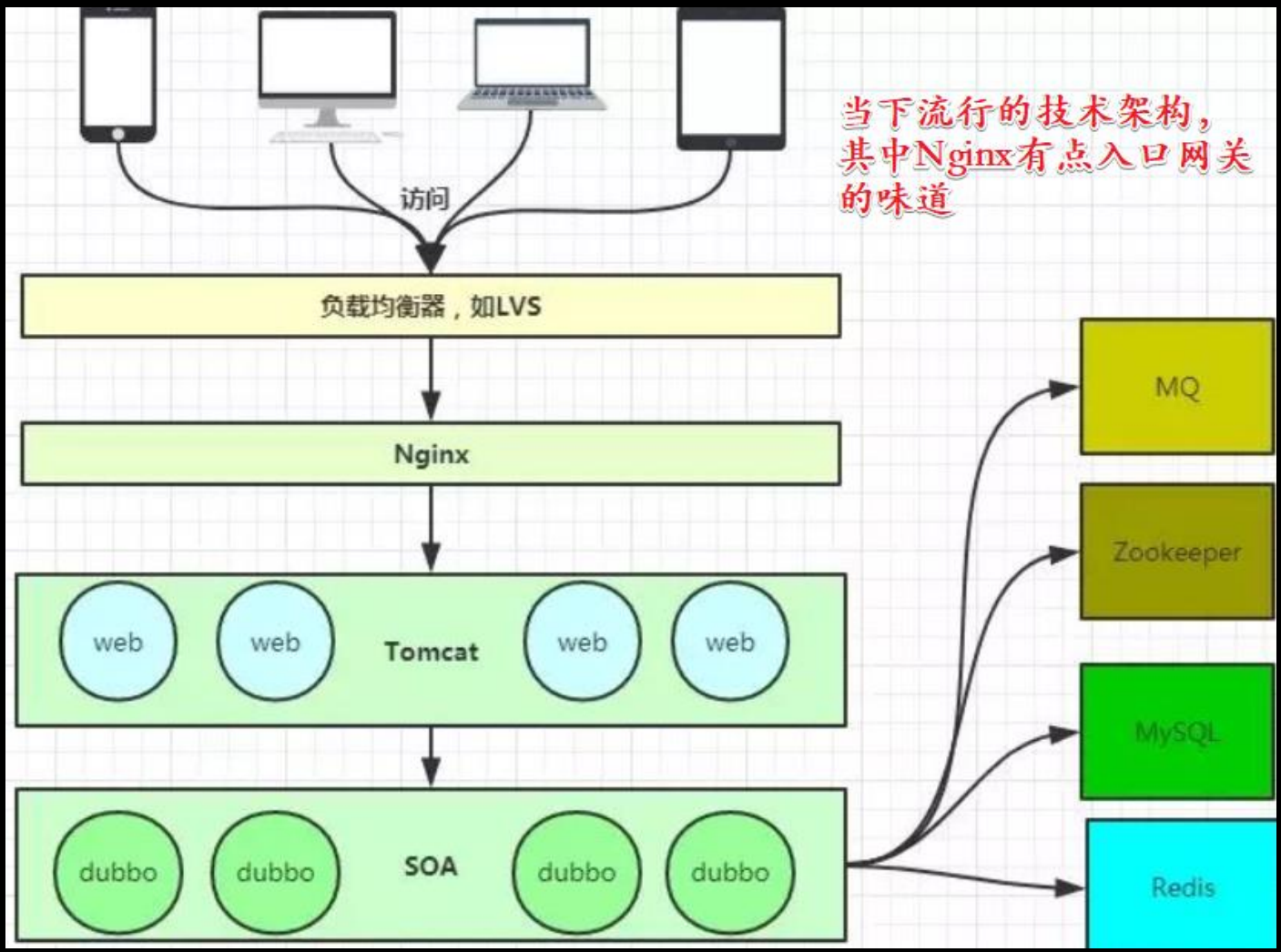
深入浅出 Nginx

https://zhuanlan.zhihu.com/p/34943332
https://juejin.cn/post/6844904129987526663

Nginx 是什么

Nginx 是一款**轻量级**的 Web 服务器、反向代理服务器，由于它内存占用少、启动极快、并发能力强，应用广泛。

Nginx 是一款轻量级的 **HTTP 服务器**，采用**事件驱动**的**异步非阻塞**处理方式框架，这让其具有极好的 IO 性能，时常用于服务端的反向代理和负载均衡

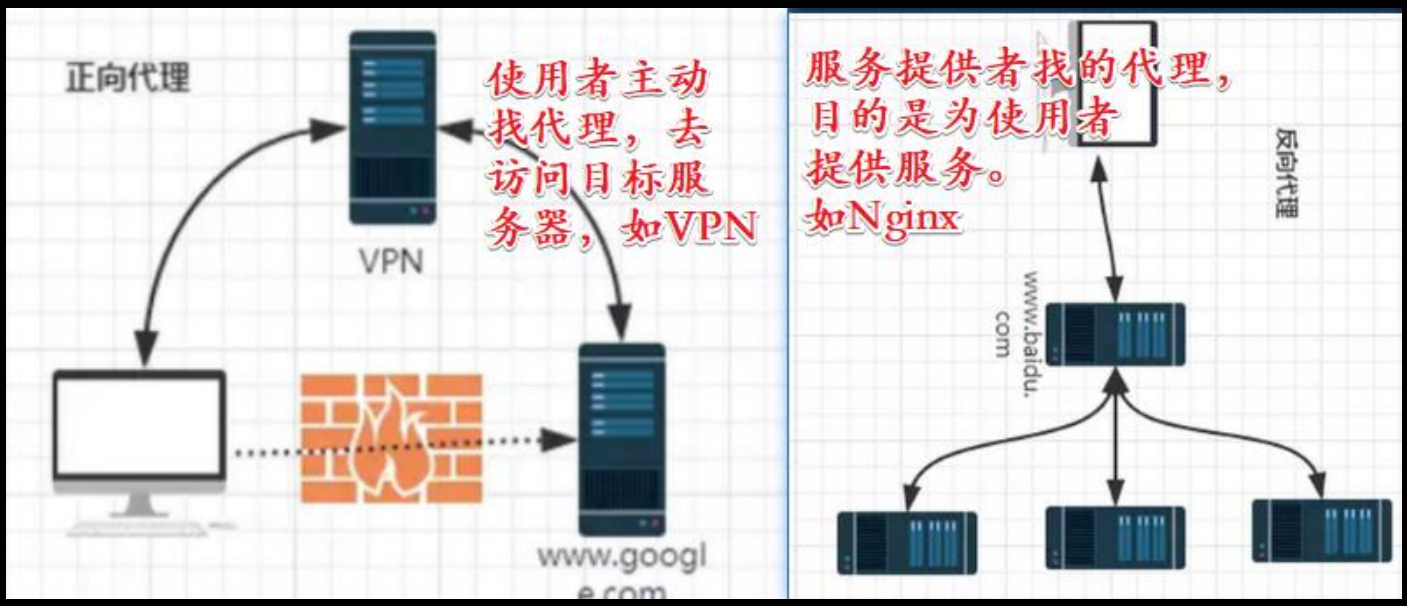


1. web 服务器：**负责处理和响应用户请求**，一般也称为 http 服务器，如 Apache、IIS、Nginx
2. 应用服务器：存放和运行系统程序的服务器，负责处理程序中的业务逻辑，如 Tomcat、Weblogic、Jboss（现在大多数应用服务器也包含了 web 服务器的功能）

总之，Nginx 是：

- 一种轻量级的 web 服务器
- 设计思想是事件驱动的异步非阻塞处理（类 node.js）
- 占用内存少、启动速度快、并发能力强
- 使用 C 语言开发
- 扩展性好，第三方插件非常多
- 在互联网项目中广泛应用

反向代理 vs. 正向代理



为什么要学习 Nginx?

Nginx 是全球排名前三的服务器
世界上约有三分之一的网址采用了 Nginx，在大型网站的架构中，Nginx 被普遍使用
Nginx 安装简单，配置简洁，作用却无可替代。Nginx 是运维和后端的必修课，也是前端进阶的必修课。
因为掌握了 Nginx，能让前端站得更高，更好的设计系统架构，更好的选择问题的解决方案，更好的服务业务开发。

安装

Mac: `brew install nginx`

修改配置

经常要用到的几个文件路径：

```
/usr/local/etc/nginx/nginx.conf （nginx 配置文件路径）
/usr/local/var/www （nginx 服务器默认的根目录）
/usr/local/Cellar/nginx/1.17.9 （nginx 的安装路径）
/usr/local/var/log/nginx/error.log (nginx 默认的日志路径)
```

nginx 默认配置文件简介：

```
# 首尾配置暂时忽略
server {
        # 当 nginx 接到请求后，会匹配其配置中的 service 模块
        # 匹配方法就是将请求携带的 host 和 port 去跟配置中的 server_name 和 listen 相匹配
        listen       8080;
        server_name  localhost; # 定义当前虚拟主机（站点）匹配请求的主机名

        location / {
            root   html; # Nginx 默认值
            # 设定 Nginx 服务器返回的文档名
            index  index.html index.htm; # 先找根目录下的 index.html，如果没有再找 index.htm
        }
}
# 首尾配置暂时忽略
```

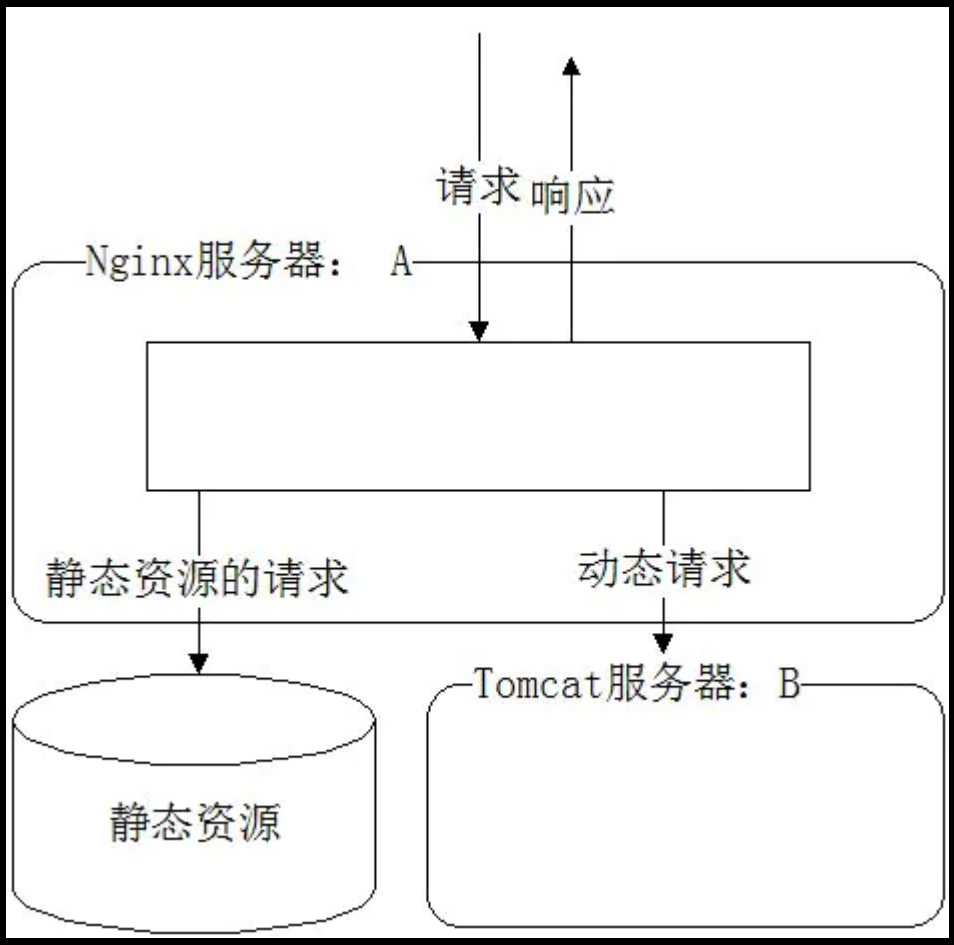
server{ } 其实是包含在 http{ } 内部的。每一个 server{ } 是一个虚拟主机（站点）。

当一个请求叫做 localhost:8080 请求 nginx 服务器时，该请求就会被匹配进该代码块的 server{ } 中执行。

Nginx 有哪些应用？

主要有 4 大应用。

1. 动静分离



动静分离其实就是 Nginx 服务器将接收到的请求分为**动态请求和静态请求**。
静态请求直接从 nginx 服务器所设定的根目录路径去取对应的资源，动态请求转发给真实的后台（前面所说的应用服务器，如图中的 Tomcat）去处理。

```
server {
        listen       8080;
        server_name  localhost;

        location / {
            root   html; # Nginx 默认值
            index  index.html index.htm;
        }

        # 静态化配置，所有静态请求都转发给 nginx 处理，存放目录为 my-project
        location ~ .*\.(html|htm|gif|jpg|jpeg|bmp|png|ico|js|css)$ {
            root /usr/local/var/www/my-project; # 静态请求所代理到的根目录
        }

        # 动态请求匹配到 path 为'node'的就转发到 8002 端口处理
        location /node/ {
            proxy_pass http://localhost:8002; # 充当服务代理
        }
}
```

访问静态资源 nginx 服务器会返回 my-project 里面的文件，如获取 index.html

2. 反向代理
服务提供者找的代理，使用者从代理这里获取服务，但是具体从哪里拿到的服务并不知道。

反向代理的作用：
a) 保障应用**服务器的安全**（增加一层代理，可以屏蔽危险攻击，更方便的控制权限）
b) 实现**负载均衡**

c) **实现跨域（号称是最简单的跨域方式）**

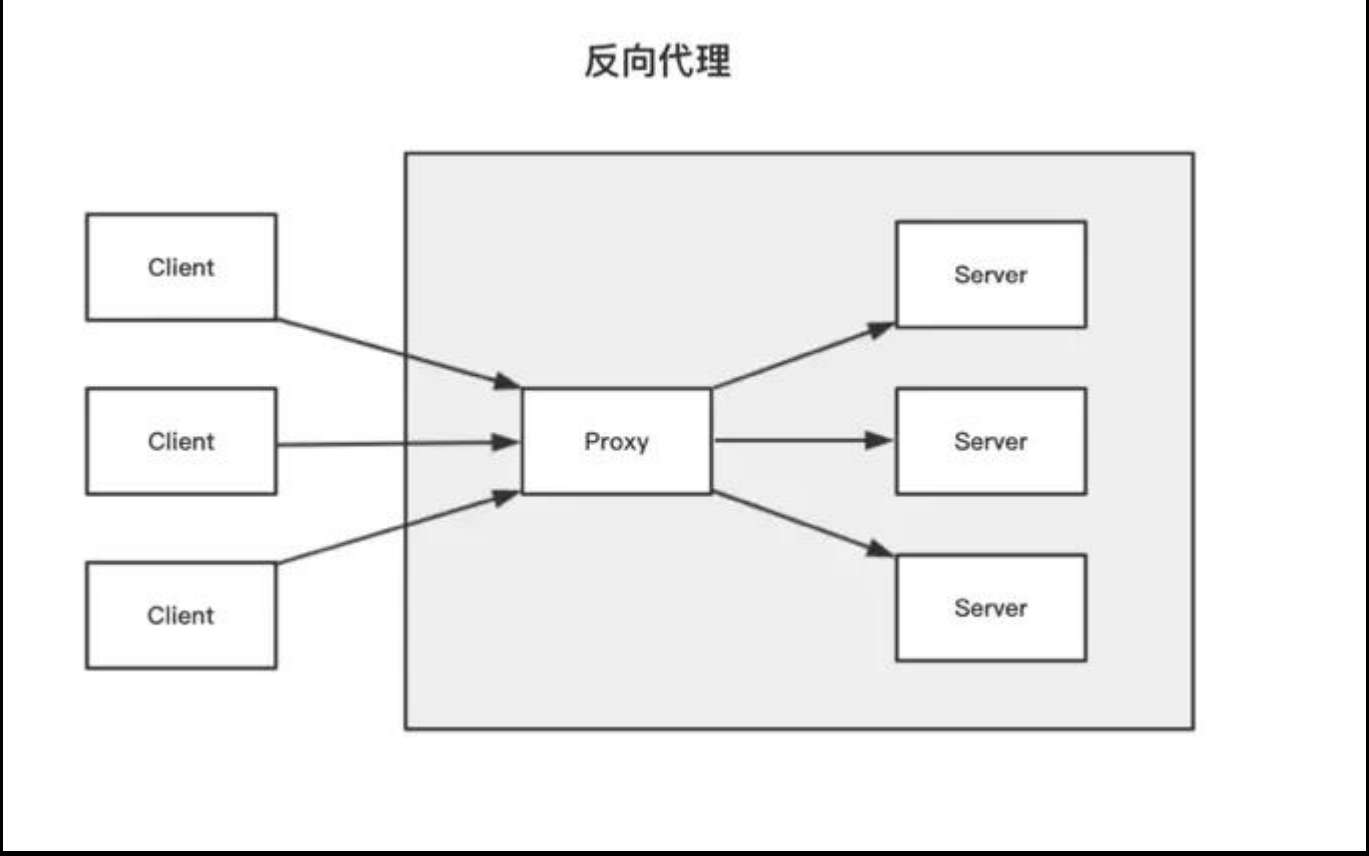
配置一个简单的反向代理：

```
server {
        listen       8080;
        server_name  localhost;

        location / {
            root   html; # Nginx 默认值
            index  index.html index.htm;
        }

        proxy_pass http://localhost:8000; # 反向代理配置，请求会被转发到 8000 端口
}
```

现实中反向代理多数是用在负载均衡中：



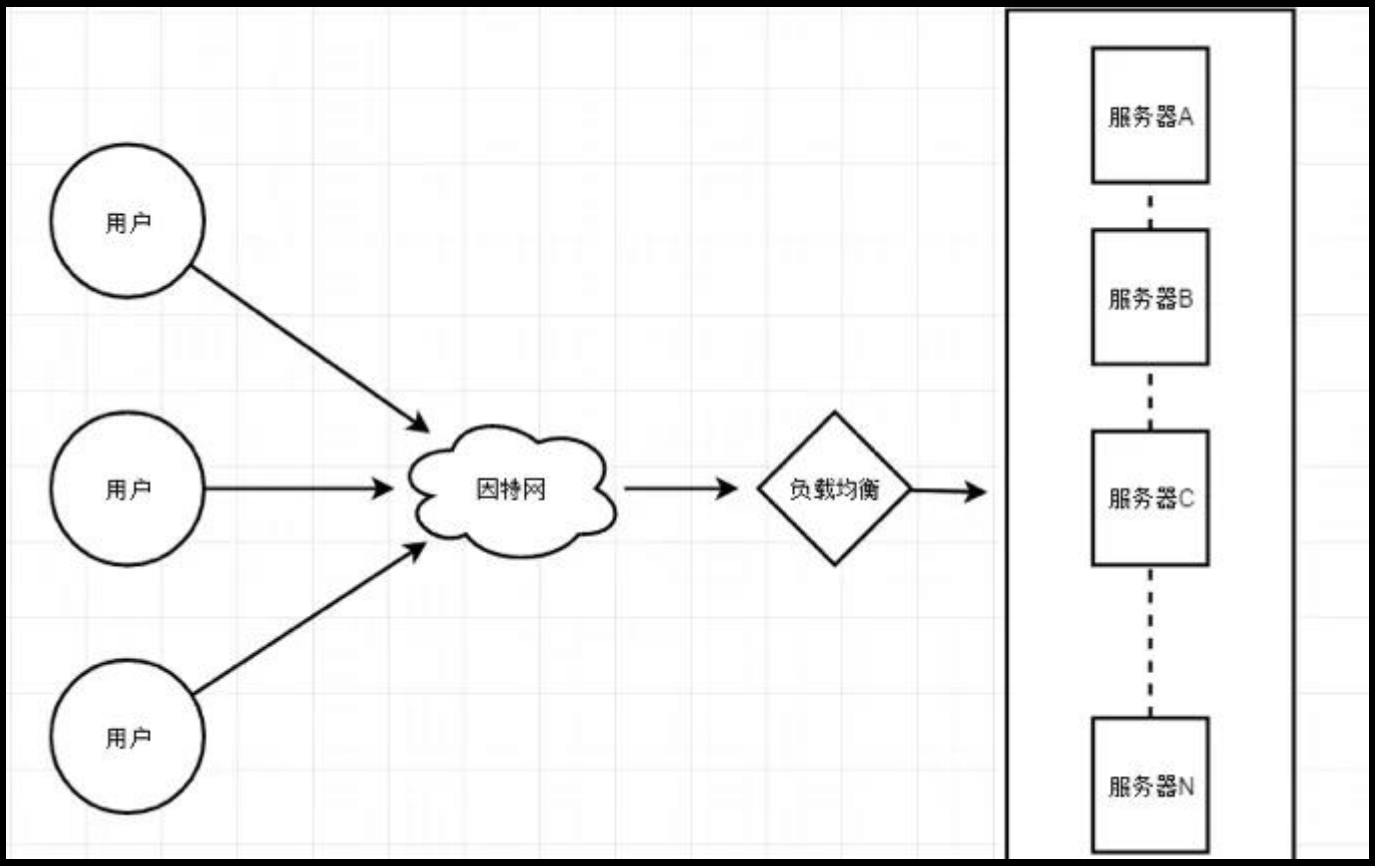
nginx 就是充当图中的 proxy。左边的 3 个 client 在请求时向 nginx 获取内容，是感受不到 3 台 server 存在的。proxy 就充当了 3 个 server 的反向代理。

反向代理应用十分广泛，**CDN 服务**就是反向代理经典的应用场景之一。除此之外，反向代理也是实现负载均衡的基础，很多大公司的架构都应用到了反向代理。

3. 负载均衡

随着业务的不断增长和用户的不断增多，一台服务已经满足不了系统要求了。这个时候就出现了服务器 集群

在服务器集群中，Nginx 可以将接收到的客户端请求“均匀地”（严格讲并不一定均匀，可以通过设置权重）分配到这个集群中所有的服务器上。这个就叫做负载均衡。



负载均衡的作用

a) 分摊服务器集群压力

b) 保证客户端访问的稳定性

除此之外，Nginx 还带有健康检查（服务器心跳检查）功能，会定期轮询向集群里的所有服务器发送健康检查请求，来检查集群中是否有服务器处于异常状态。

配置负载均衡：

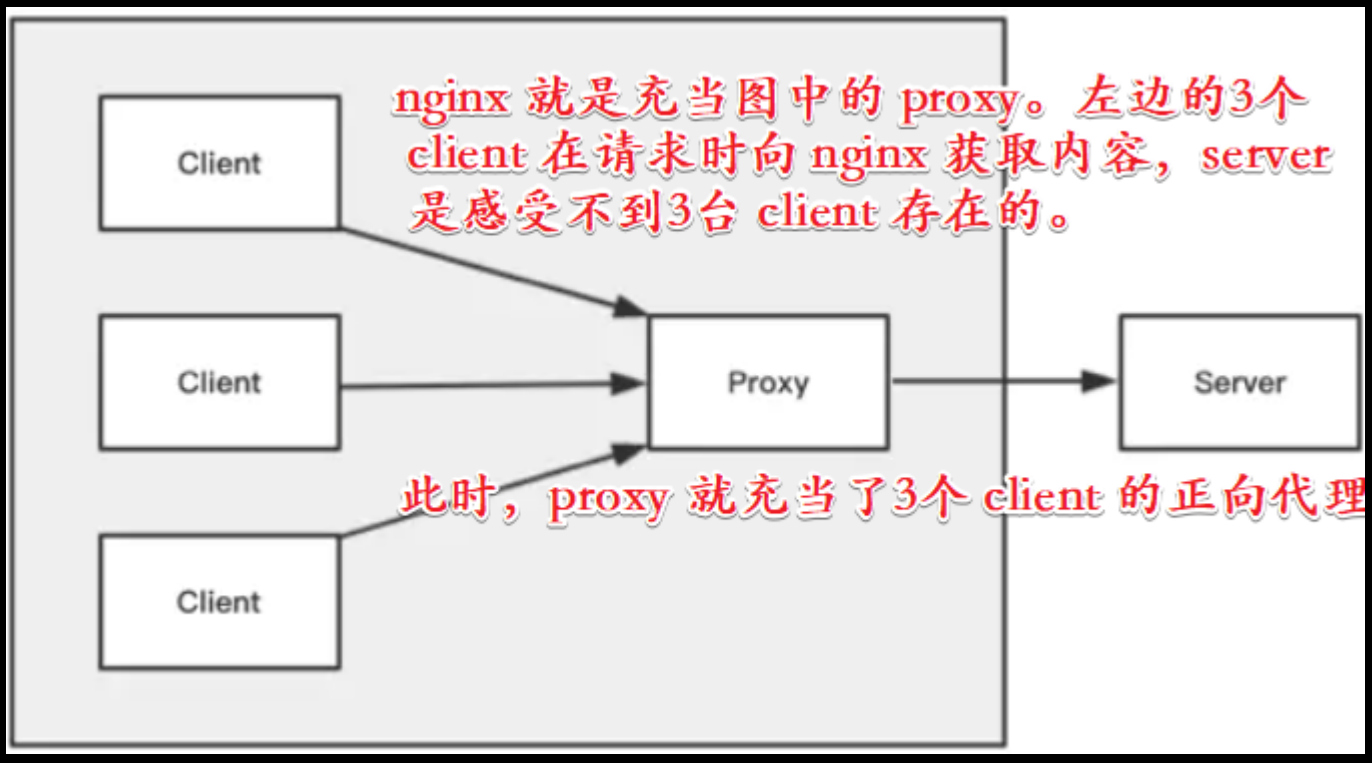
```
    # 负载均衡：设置 domain
upstream domain {
    server localhost:8000;
    server localhost:8001;
}
server {
        listen       8080;
        server_name  localhost;

        location / {
            # root   html; # Nginx 默认值
            # index  index.html index.htm;

            proxy_pass http://domain; # 负载均衡配置，请求会被平均分配到 8000 和 8001 端口
            proxy_set_header Host $host:$server_port;
        }
}
```

4. 正向代理

和反向代理相反，相当于代购，店主并不知道是谁买的



Vpn：正向代理

CND

内容分发网络（Content Delivery Network，**CDN**）是建立并覆盖在承载网上，由不同区域的服务器组成的分布式网络。**将源站资源缓存到全国各地的边缘服务器，供用户就近获取，降低源站压力。**

安装

安装依赖：

1. yum install gcc
2. yum install pcre-devel
3. yum install zlib zlib-devel
4. yum install openssl openssl-devel
5. //一键安装上面四个依赖
   yum -y install gcc zlib zlib-devel pcre-devel openssl openssl-devel

下载 nginx 的 tar 包

```
cd /usr/local
mkdir nginx
cd nginx
//下载 tar 包
wget http://nginx.org/download/nginx-1.19.3.tar.gz
tar -xvf nginx-1.13.7.tar.gz
```

安装 nginx

```
//进入 nginx 目录
cd /usr/local/nginx
//执行命令
./configure
//执行 make 命令
make
//执行 make install 命令
make install
```

Nginx 常用命令

```
//测试配置文件
安装路径下的/nginx/sbin/nginx -t
复制代码
//启动命令
安装路径下的/nginx/sbin/nginx
//停止命令
安装路径下的/nginx/sbin/nginx -s stop
或者 : nginx -s quit
//重启命令
安装路径下的/nginx/sbin/nginx -s reload
```

//平滑重启

```
kill -HUP Nginx 主进程号
```

配置防火墙

```
//打开防火墙文件
sudo vim /etc/sysconfig/iptables
//新增行  开放 80 端口
-A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
//保存退

//重启防火墙
sudo service iptables restart
```

```
Nginx 启动
//进入 nginx 安装目录
cd sbin
sudo ./nginx
测试访问
http://ip 地址
```

Ubuntu 下可以直接使用 apt-get install nginx
为什么 CentOS 需要使用 yum 编译安装

Ubuntu的	CentOS的
基于Debian	基于RHEL
经常更新	几乎没有更新
没有cPanel支持（有替代品）	支持cPanel / WHM
更大的用户和开发人员社区	较小的用户和开发人员社区
以教程和免费指南的形式提供更多帮助	提供的帮助较少
对于过去使用过Ubuntu桌面的初学者来说，更容易学习	由于RHEL发布的着名桌面发行版并不多，因此难以为初学者学习
使用apt-get包管理器安装的.deb包	使用yum包管理器安装的.rpm包

Ubuntu:
apt-get install nginx
service nginx start
启动后输入 IP 即可查看 http://42.192.203.208/ （端口默认是 80）

配置:
/etc/nginx/nginx.conf

Linux 中服务器软件为什么需要编译安装

不是因为 编译安装性能好，而是因为:
1. 软件在编译期间需要配置：比如 Linux，需要在编译的时候指定包含哪些 module，PHP、Apache 也是一样。
   同样的是数据库，mysql 通过编译安装，因为要定制存储引擎，比如是否支持 innodb，而 sqlite

都是直接下载二进制文件使用。
2. 软件需要统一安装路径，每个 team 都会有自己的安装目录约定，有的喜欢/opt/有的喜欢/usr/local/，编译安装可以方便的指定这些路径(configure -prefix=xxx)
3. 需要最新的版本

https://www.nginx.cn/doc/

常用命令

start nginx: 启动 nginx
nginx -s stop: 关闭 nginx
nginx -s reload: 重新加载配置
nginx -s reopen: 重新打开
nginx -t: 检测配置文件是否正常

更改配置不生效

我们对 nginx 的配置主要写在 nginx.conf 文件里，这个目录下还有 **conf.d 和 sites-enabled** 两个文件夹，**里面为默认的配置文件**。相应的，在配置 nginx，编辑 nginx.conf 文件时，**需要把这两行注释掉，否则 nginx.conf 不会生效。**

```
include /etc/nginx/conf.d/*.conf;
include /etc/nginx/sites-enabled/*;
```

配置的时候，末尾的逗号不能少!!!

Nginx 配置 SSL 证书

```
        server{
                listen  9999 ssl; #SSL 访问端口号
                server_name  edwinxu.xyz; #填写绑定证书的域名
                ssl_certificate /home/ubuntu/EdwinXu/ssl/nginx/1_edwinxu.xyz_bundle.crt; #证书文件名称
                ssl_certificate_key /home/ubuntu/EdwinXu/ssl/nginx/2_edwinxu.xyz.key; #私钥文件名称
                ssl_session_timeout 5m;
                ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
                ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:HIGH:!aNULL:!MD5:!RC4:!DHE;
                ssl_prefer_server_ciphers on;

                location / {
                        index index.html;
                        root /home/ubuntu/EdwinXu/email-sender/frontend;
```

```
                try_files $uri $uri/ /index.html;
        }
    }
```

Nginx 反向代理:HTTPS 到 HTTP

做了一个Web项目，前后端分离，前端需要使用HTTPS访问。
部署的时候发现，前端HTTPS项目是不能发起HTTP的ajax到后端的，怎么办?
最直接的就是把后端也部署到支持HTTPS的Web服务器上，比如Tomcat，可以配置SSL。不过这样实在太麻烦，而且我们一般使用的是java内嵌的tomcat。

这时候可以考虑Nginx反向代理。

前端ajax不能使用HTTP访问，那就使用HTTPS呗，使用Nginx反向代理，把这个HTTPS代理到HTTP不就行了!

所以，解决方案是:

1. 前端部署到Nginx上，配置HTTPS。比如通过 https://frontend.com 访问。前端 Ajax也使用HTTPS访问后端，比如 https://ajax.com。
2. 部署后端: java -jar， HTTP方式，比如通过 http://backend.com 访问
3. Nginx设置反向代理: 把 https://ajax.com 代理到 http://backend.com。

这样前端HTTPS和后端HTTP就能通信了!

```
    # 这是 NG 直接运行的 email-sender 前端， https 方式
    server{
            listen  9999 ssl;
            server_name  edwinxu.xyz;
            ssl_certificate /home/ubuntu/EdwinXu/ssl/nginx/1_edwinxu.xyz_bundle.crt;
            ssl_certificate_key /home/ubuntu/EdwinXu/ssl/nginx/2_edwinxu.xyz.key;
            ssl_session_timeout 5m;
            ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
            ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:HIGH:!aNULL:!MD5:!RC4:!DHE;
            ssl_prefer_server_ciphers on;

            location / {
                    index index.html;
                    root /home/ubuntu/EdwinXu/email-sender/frontend;
                    try_files $uri $uri/ /index.html;


            }
    }


    #反向代理，把 ajax https 请求代理到 HTTP 后端
    server{
            listen  7777  ssl;
            server_name  edwinxu.xyz;
            ssl_certificate /home/ubuntu/EdwinXu/ssl/nginx/1_edwinxu.xyz_bundle.crt;
            ssl_certificate_key /home/ubuntu/EdwinXu/ssl/nginx/2_edwinxu.xyz.key;
            ssl_session_timeout 5m;
            ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
            ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:HIGH:!aNULL:!MD5:!RC4:!DHE;
```

```
        ssl_prefer_server_ciphers on;

        location / {
                # 加了这个就是反向代理，会把所有请求都转发给下面这个服务
                proxy_pass http://edwinxu.xyz:8888;

        }
    }
```

vue-router+nginx 非根路径配置方法

https://blog.csdn.net/m0_37960566/article/details/105931227