Nginx 原理探究

作者：jyl

原文链接：https://ld246.com/article/1621930014906

来源网站：链滴

关于ng:

优点简直太多了，我觉得高并发和热部署、负载均衡、动静分离再加上完全开源免费就足以让开发者涎了。(占用内存还少、配置还简单。。)

我今年25岁，做了5年web开发，力推ng。只能说这个WEB+反向代理服务器完美顺应了如今的开发势。(前后端分离)

关于正向代理、反向代理:

正向代理我们日常就用的很多，因为他非常直观透明，比如使用burp、翻墙挂代理等等

举个例子，现在国外的同事开发完一个项目让我访问地址为http://1.2.3.4/index.html，然后我从国根本访问不到，国外同事说

需要挂一个代理服务器才能访问到，于是我就从internet的网络连接设置成了先去连接可以访问到这国外地址的代理地址，向代理

地址发送请求http://1.2.3.4/index.html的指令，(通过代理携带指令请求)代理作为可访问到的转发器发请求，同时返回访问到的数据信息

所以正向代理:客户端、服务器端、代理端都是透明可见的。。。但是对于服务端来说是不透明的，服端是不知道是客户端还是代理请求来的。

反向代理就是，你要访问http://5.6.7.8/，然后一下就访问到了，所以你就默认认为自己可以访问到个地址，

其实内部可能是你的同事已经通过ng事先将http://5.6.7.8/指向到了其他地址http://8.7.6.5/，最终是通过http://5.6.7.8/

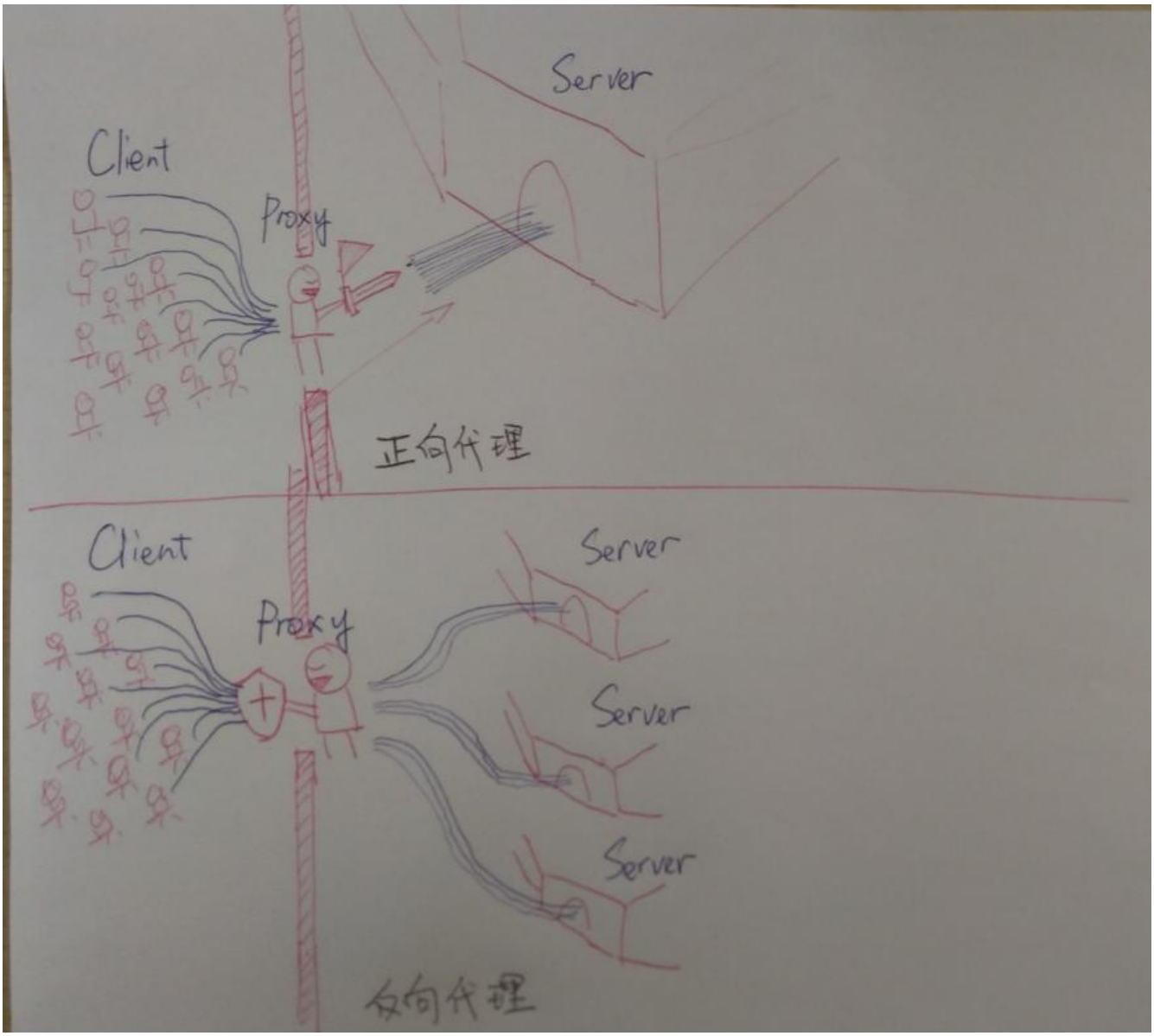
访问到了http://8.7.6.5/，归结起来也是通过一种代理转发的方式去请求，但是你还一直觉得自己可访问到的是http://5.6.7.8/，中间的请求操作是不透明的。

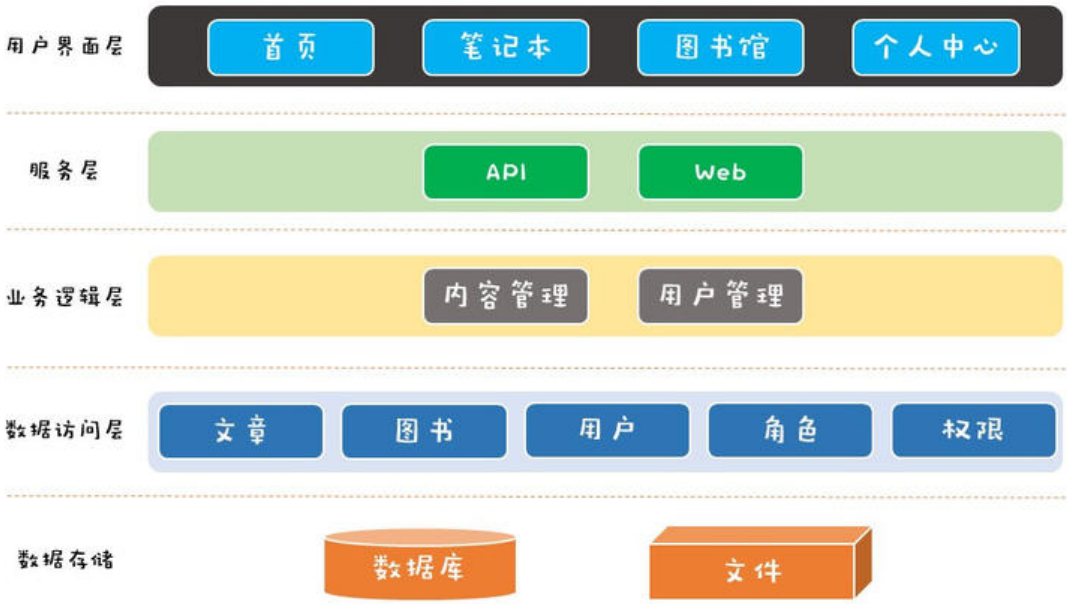
这个例子举的不清晰，类似于一个网络代理请求访问，实际的业务逻辑里我们一般是不会这么举例的用ng做反向纯粹是为了隐藏目标服务地址。

你访问了一个网站，你以为它是“谷弟弟”，但其实它是“谷姐”，“谷姐”知道你其实是想找她弟，就取回“谷弟弟”的内容给你看。作为用户的你，是不知道有这个过程的，这么做是为了保护服务器不暴露服务器的真实地址。

所以反向代理:客户端可见，你以为服务器端可见，其实你访问的服务端就是代理地址，可见的是代理服务端并不可见。

对于客户端来说是不透明的，客户端是不知道是服务端还是代理请求来的

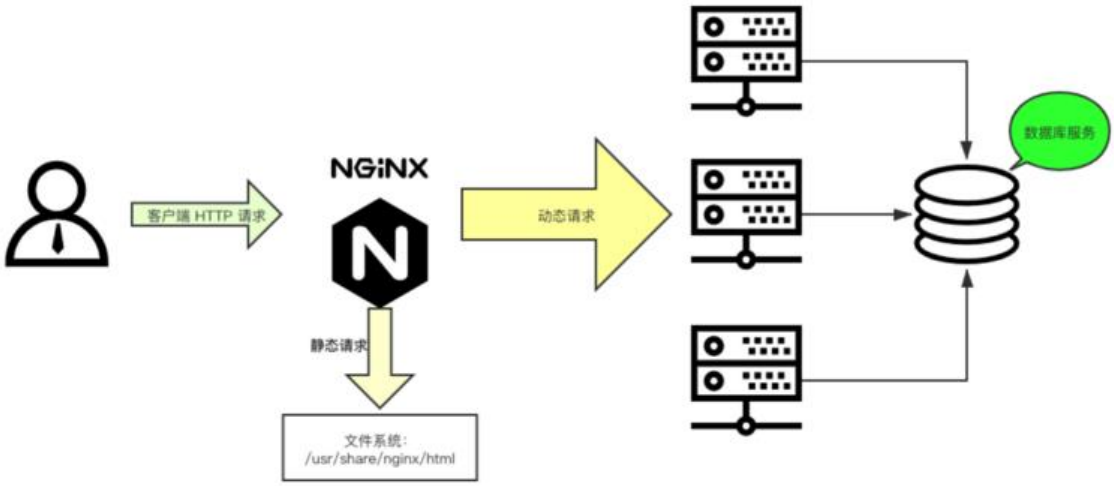




这三张图是抄以前做前后端白卷项目**Evan-Nightly**大佬的图。分别是业务图解正反向代理、流程图技术栈。针对此项目，ng就可以有很好的应用场景。(技术栈这张好像没传上来，算了，可以去大佬客翻翻)

动静分离:说白了就是你打你的，我打我的，前后端分离里，启动一个前端项目作为静态资源访问，后项目作为动态资源访问。说白了就是在ng上代理挂载俩项目

动静分离是指在 web 服务器架构中，将静态页面与动态页面或者静态内容接口和动态内容接口分开不同系统访问的架构设计方法，进而提示整个服务的访问性和可维护性。



一般来说，都需要将动态资源和静态资源分开，由于 Nginx 的高并发和静态资源缓存等特性，经常将静态资源部署在 Nginx 上。如果请求的是静态资源，直接到静态资源目录获取资源，如果是动态资源的请求，则利用反向代理的原理，把请求转发给对应后台应用去处理，从而实现动静分离。

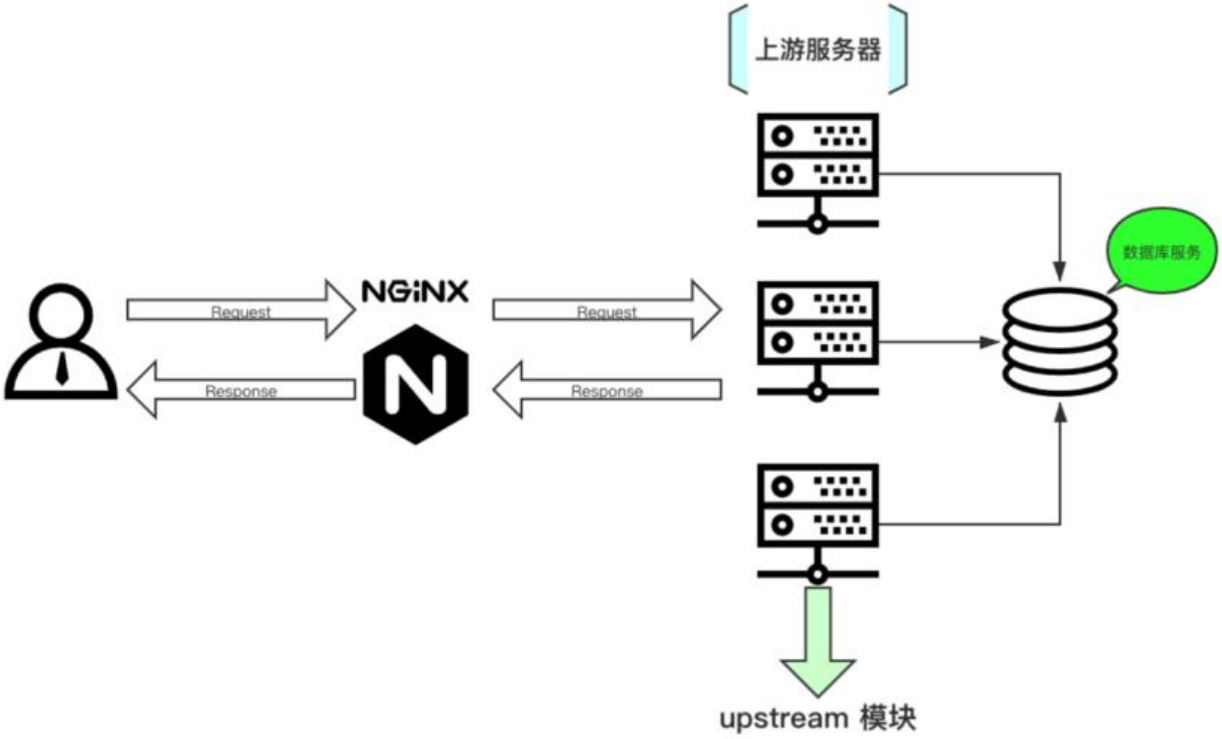
使用前后端分离后，可以很大程度提升静态资源的访问速度，即使动态服务不可用，静态资源的访问也不会受到影响。

负载均衡:从开发角度看，负载均衡是一种轮询机制，目的是减轻服务器压力，我不太清楚用ng做负均衡的好处，平常都是在springcloud代码

实现。网上搜了一下:Nginx适合于服务器端实现负载均衡 比如Tomcat ， Ribbon适合与在微服务中RC远程调用实现本地服务负载均衡，比如

Dubbo、SpringCloud中都是采用本地负载均衡。总结一下，单应用springboot项目挂ng负载，sprigcloud就用自己的ribbon做负载

总结下ng负载均衡:可以理解为是相同的后端程序用不同端口启动，用ng配置后访问后端会采用轮询制访问，其原理和springcloud ribbon差不多。

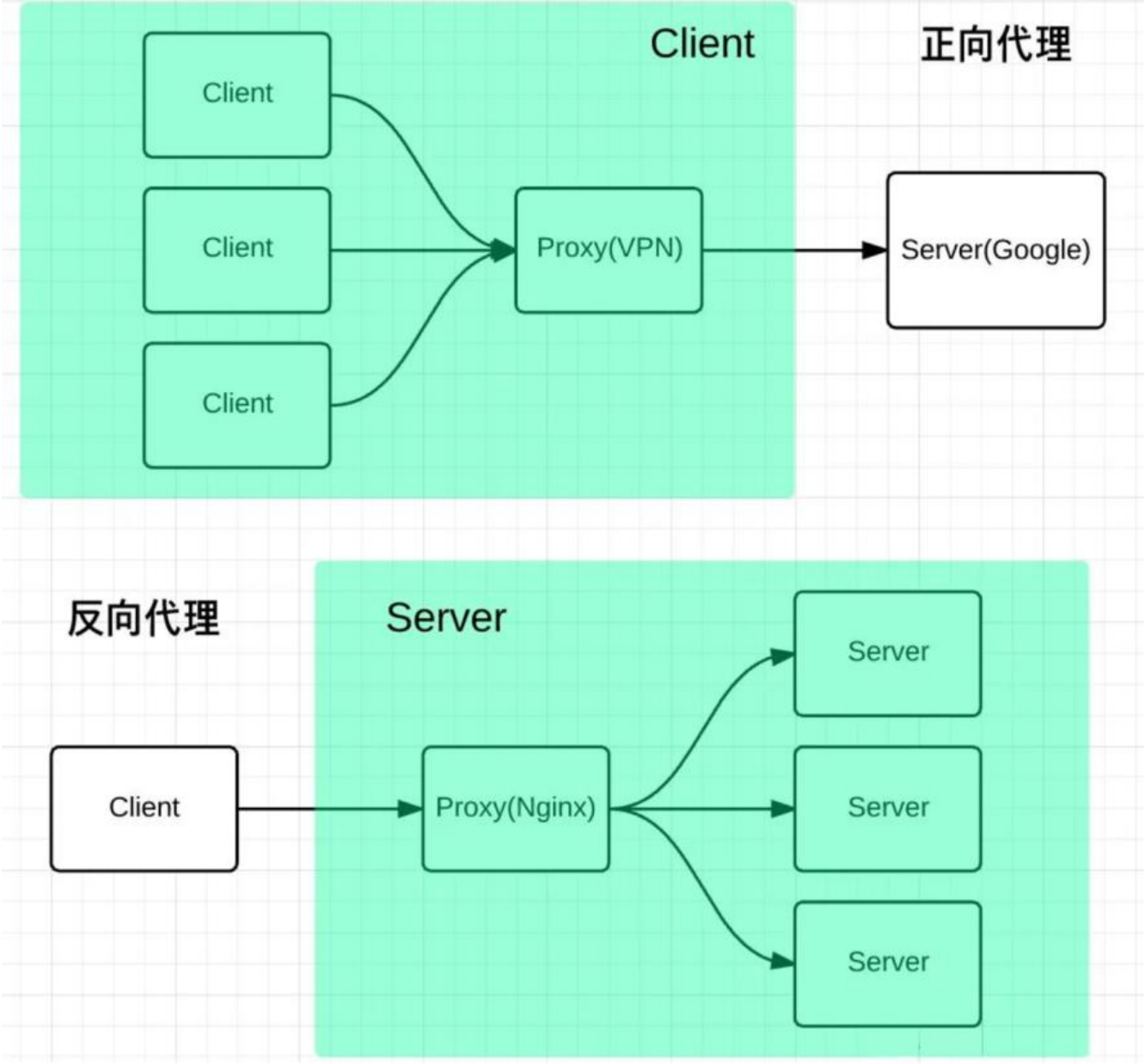


ngnix跨域:

照csdn配置即可。

补一张正反向代理透明关系图

原文链接：Nginx 原理探究



其实这几张图我在上面都有做笔记，摆这就是备忘一下。

4月份总结的了，明天会深入的去研究下负载均衡(SLB)模拟实现一个宕机场景。

参考文章:万字总结，带你全面系统的认识 Nginx

侵转删yum