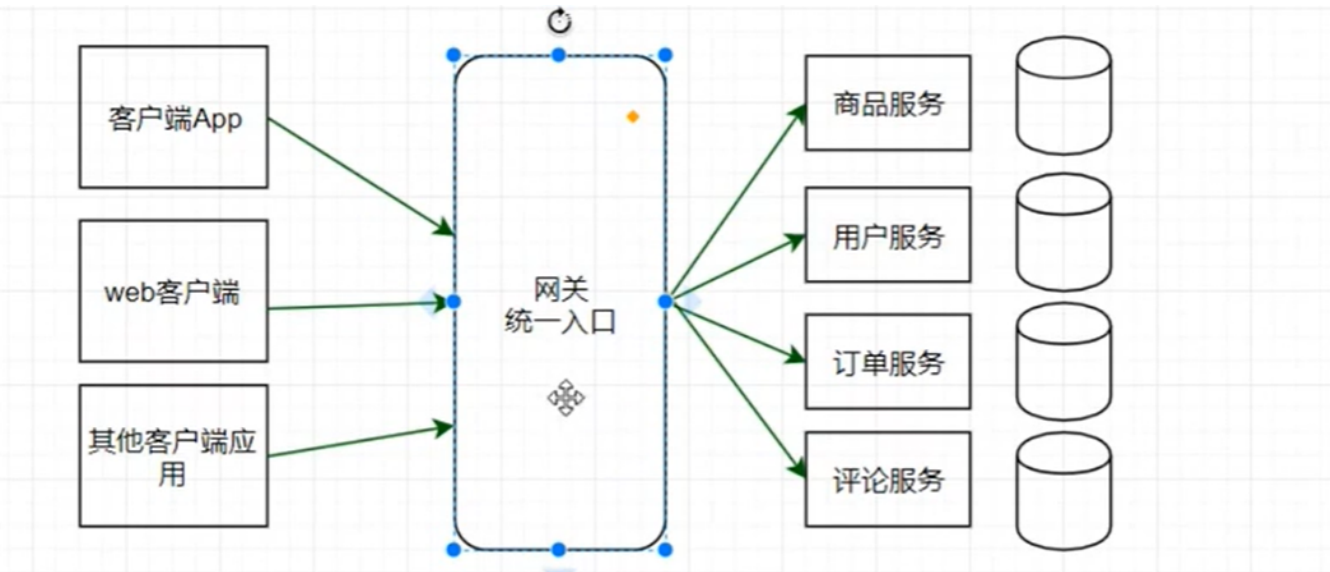
初识API网关

api网关：

API 网关将各系统对外暴露的服务聚合起来，所有要调用这些服务的系统都需要通过 API 网关进行访问，基于这种方式网关可以对 API 进行统一管控，例如：认证、鉴权、流量控制、协议转换、监控等等。

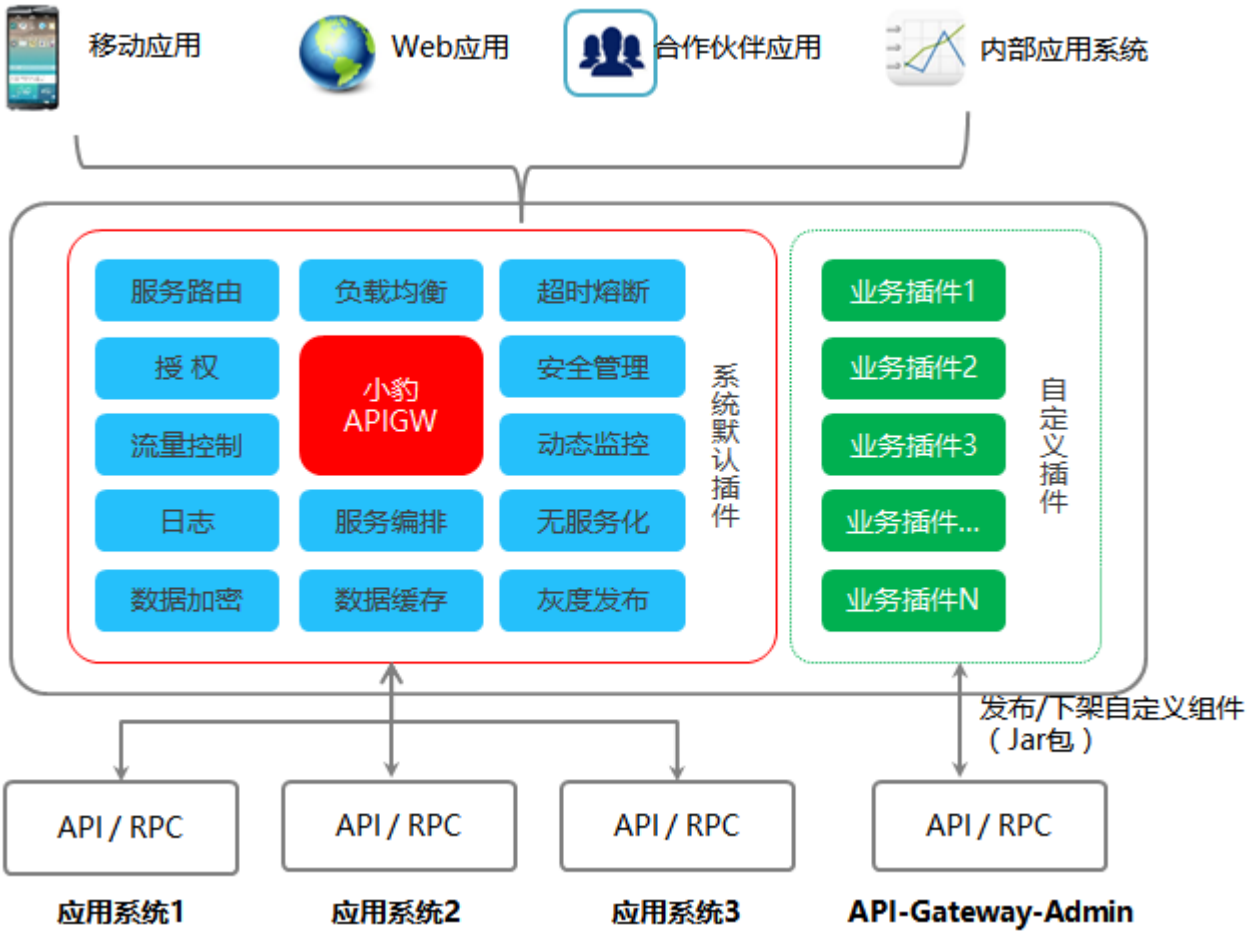
作用：

- 减轻客户端的负重，不需要知道商品服务、用户服务、订单服务等服务的地址，其实就是将客户端和服务端进行更深层次的解耦，汇聚服务，统一服务入口
- 安全：实现微服务访问流量计算，基于流量计算分析进行限流，可以定义多种限流规则
- 缓存：数据缓存
- 日志：日志记录
- 监控：记录请求响应数据，api耗时分析，性能分析
- 重试：异常重试
- 熔断：降级



API 网关的流行得益于近几年微服务架构的兴起，原本一个庞大的业务系统被拆分成许多粒度更小的系统进行独立部署和维护，这种模式势必会带来更多的跨系统交互，企业 API 的规模也会成倍增加，API 网关（或者微服务网关）就逐渐成为了微服务架构的标配组件。

如下是我们整理的 API 网关的几种典型应用场景：



现有框架

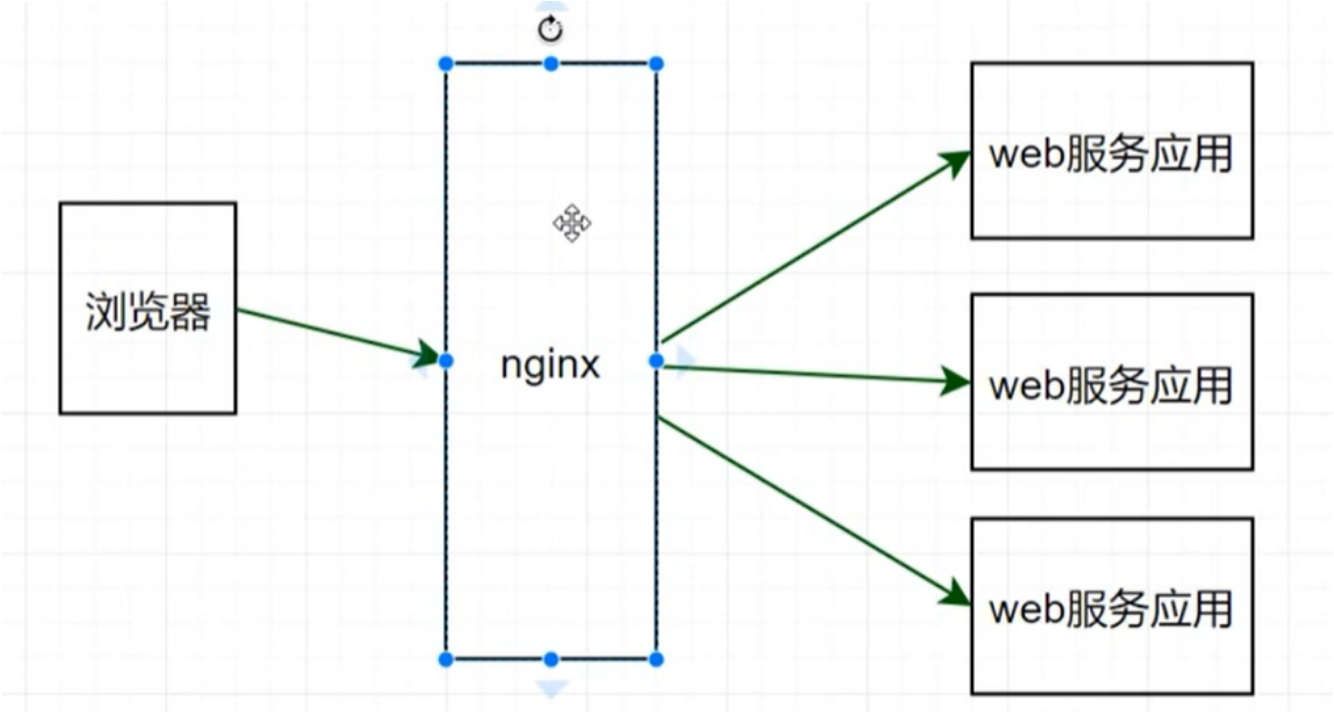
Tyk：Tyk是开放源码的api网关，它是快速、可扩展和现代的。Tyk提供了一个api管理平台，其中包括api网关、api分析、开发人员门户和api管理面板。Tyk是基于Go实现的网关服务。

Kong：Kong是一个可扩展的开放原码API Layer（也称位api网关/api中间件）。kong在任何RESTful api的前面运行，通过插件扩展，它提供了超越核心平台的额外功能和服务。

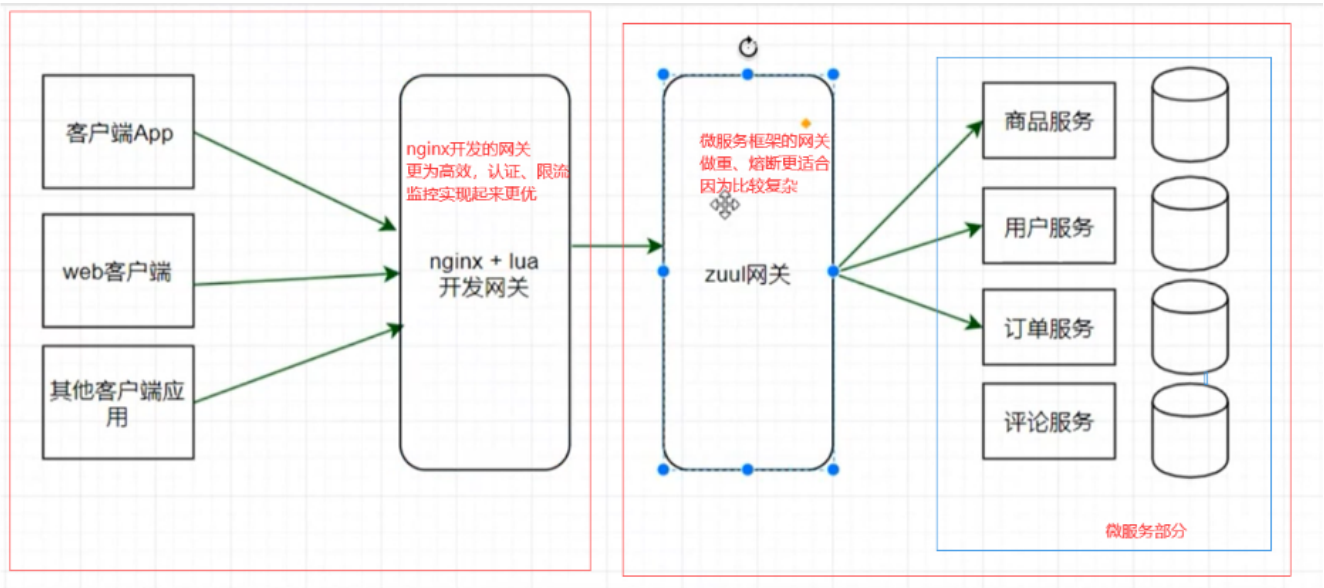
类似的框架还有：Orange（和Kong类似，基于openResty的网关，是由国人开发的）、 apiaxle（基于Nodejs），zuul（spring clond）等

Web应用部署

在我们web应用部署的时候有没有一个框架很类似于api网关？ 那就是Nginx，浏览器返回nginx的虚拟主机，nginx反向代理后端服务，所以看上去很类似一个网关。Kong、Orange 都是基于（nginx + lua）技术网关，lua脚本技术，常用于游戏开发，可直接运行。openresty相当于升级版的nginx，lua类库。



我们将基于nginx + lua脚本开发一个网关：



Nginx的下载和安装

1、Nginx下载： nginx-1.13.0.tar.gz 下载到 /opt/softwares/

```
wget -P /opt/softwares http://nginx.org/download/nginx-1.13.0.tar.gz
```

2、Nginx解压安装

```
tar -zxvf nginx-1.13.0.tar.gz -C ./
```

3、预先安装

```
yum -y install gcc gcc-c++ ncurses-devel perl pcre-devel zlib zlib-devel
```

4、Nginx编译

```
./configure --prefix=/usr/local/nginx
```

5、安装

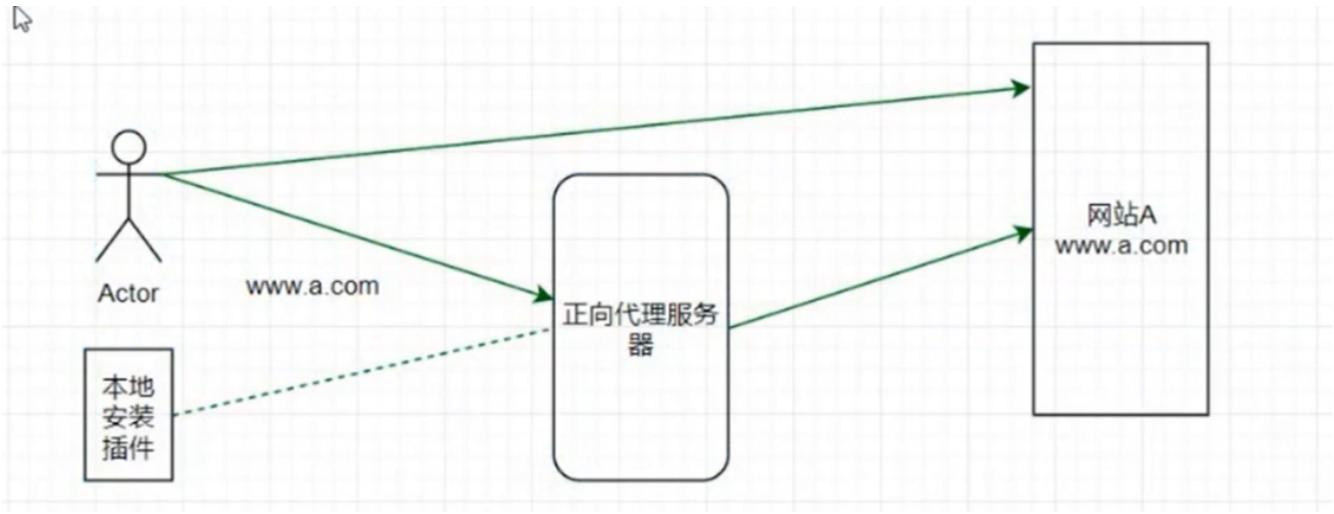
```
make & make install
```

6、查看安装路径

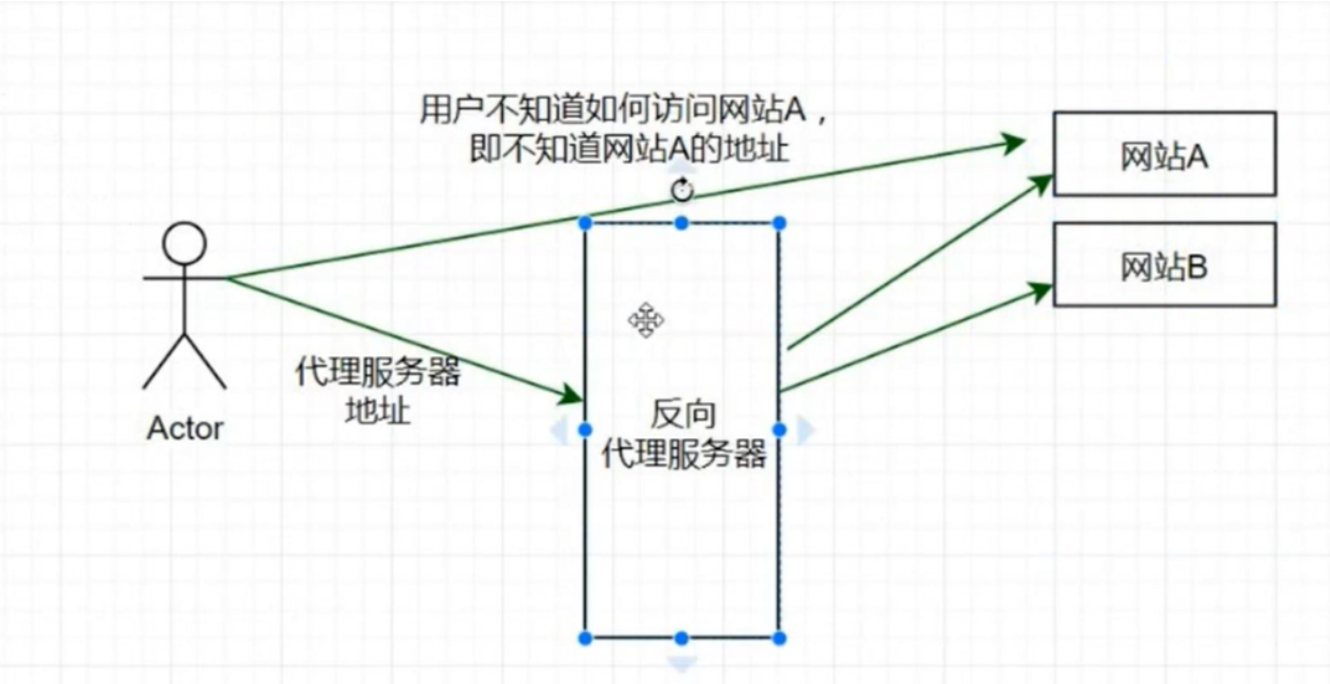
```
cd /usr/local/nginx

./nginx  启动nginx
./nginx -v  检测版本
./nginx -s quit  关闭nginx
```

正向代理和反向代理

正向代理：用户直接访问网站有时候是访问不了的，需要使用代理服务器。（类似于你想上课期间找学生，你可以把东西给班主任，班主任给学生）



反向代理： 用户需要访问一些服务应用，但对方不想把服务应用地址暴露给用户，这样可以确保安全，那用户如果访问那?可以通过反向代理服务器，用户只需要知道反向代理服务器的地址就可以（类似于你要去一个班找一个学生，但是你只知道这个班的班主任）



正向/反向代理的：

1、正向代理需要在用户的电脑上配置正向代理的服务器；而反向代理不需要，因为用户访问的就是反向代理服务器
2、正向代理的应用场景是 用户是知道目标服务器的地址的，而反向代理的应用场景是用户本来就不知道目标服务器地址
3、反向代理极大的保护了应用的安全性，而且此结构可以很好的搭建负载均衡