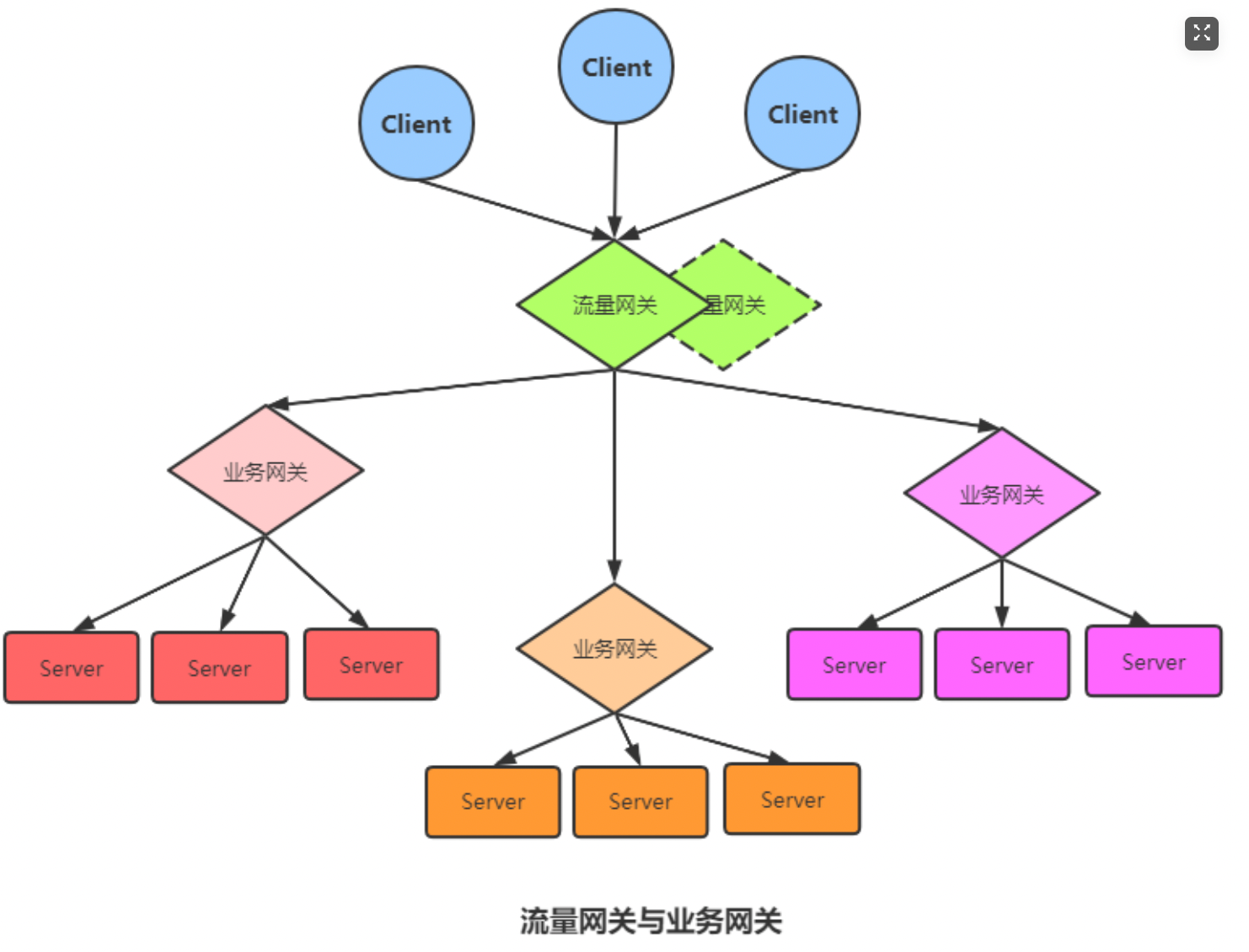
**网关技术选型**

# 网关使用场景

通常下网关分为流量网关与业务网关，流量网关负责接入所有的流量，将流量分发给不同的系统，在具体的业务接入之前还有一层业务网关。

流量网关提供全局性的、与后端业务应用无关的策略，例如全局流量监控、日志记录、全局限流、黑白名单、接入请求到业务层的LB层等，比如Kong就是典型的流量网关。

业务网关一般是跟在流量网关之后部署的，提供单个业务域级别与业务非强相关的一些公共策略，如服务治理、身份认证、权限控制、熔断限流、服务降级、协议转换等功能，目前较成熟流行的开源业务网关有SpringCloud Gateway、zuul1和zuul2。



从上边这张图来看，流量网关负责南北向流量调度及安全防护，业务网关负责东西向流量调度及服务治理。故流量网关又称南北向网关，业务网关又称为东西向网关。

# 常见网关介绍

目前常见的开源网关按照语言分类有如下几类：

Nginx+lua: OpenRestry、Kong、APISIX、Oragne、Abtesting gateway等

Java: Zuul/Zuul2 、SpringCloud Gateway、Kaazing KWG、Dromara soul等

Go: Janus、fagongzi、Grpc-gateway

Donet: Ocelot

NodeJS: Express Gateway 、Micro Gateway

按照使用量、成熟度来划分，主流的有以下几个:

OpenRestry、Kong、APISIX、Zuul1\Zuul2、SpringCloud Gateway以及阿里的Higress等。

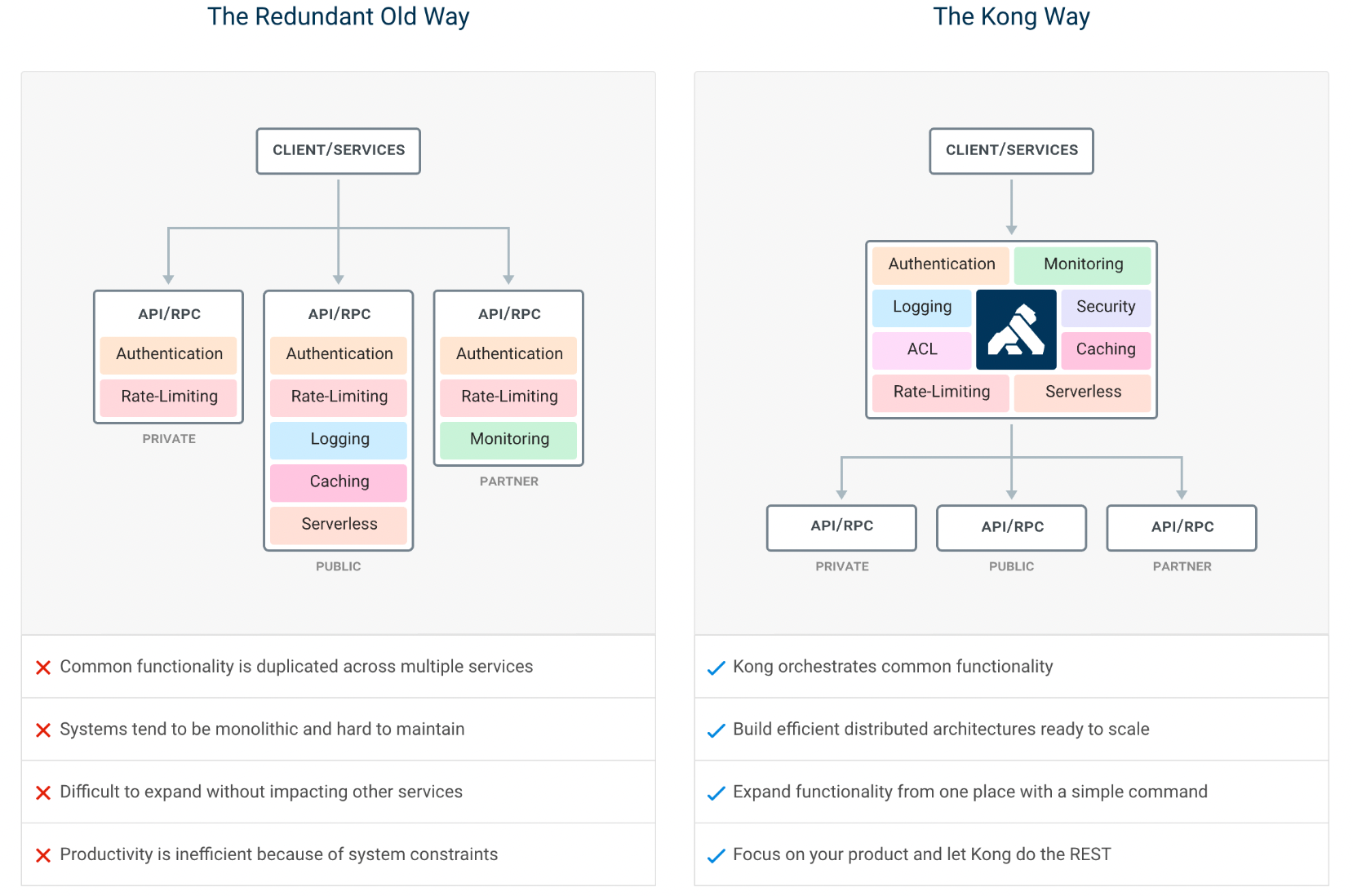
## 2.1 OpenRestry

OpenResty是一个流量网关，OpenResty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。用å于方便地搭建能够处理超高并发、扩展性极高的动态 Web 应用、Web 服务和动态网关。

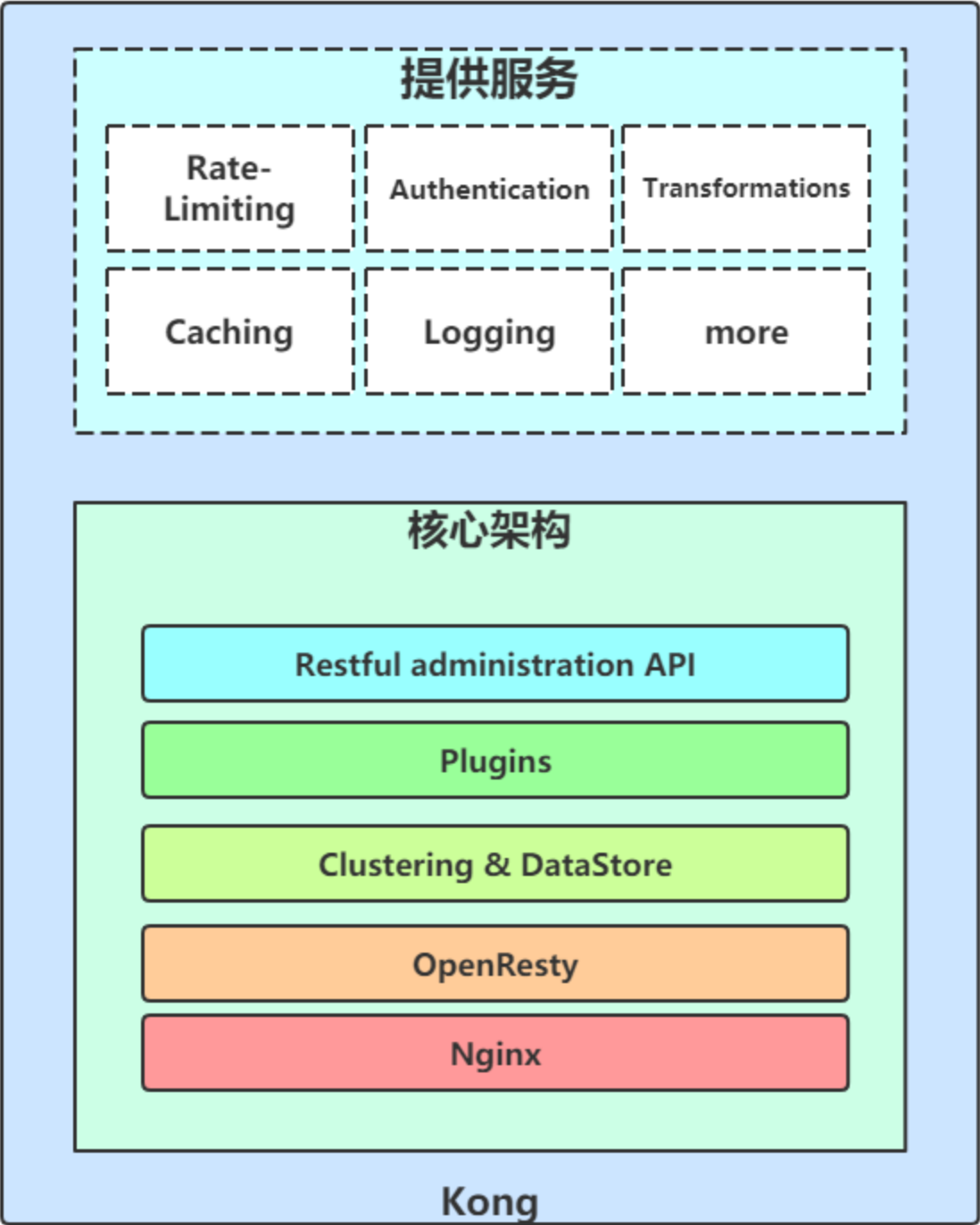
## **2.2 Kong**

Kong基于OpenResty开发，也是流量层网关, 是一个云原生、快速、可扩展、分布式的Api 网关。继承了OpenResty的高性能、易扩展性等特点。Kong通过简单的增加机器节点，可以很容易的水平扩展。同时功能插件化，可通过插件来扩展其能力。而且在任何基础架构上都可以运行。具有以下特性：

* 提供了多样化的认证层来保护Api。
* 可对出入流量进行管制。
* 提供了可视化的流量检查、监视分析Api。
* 能够及时的转换请求和相应。
* 提供log解决方案
* 可通过api调用Serverless 函数。



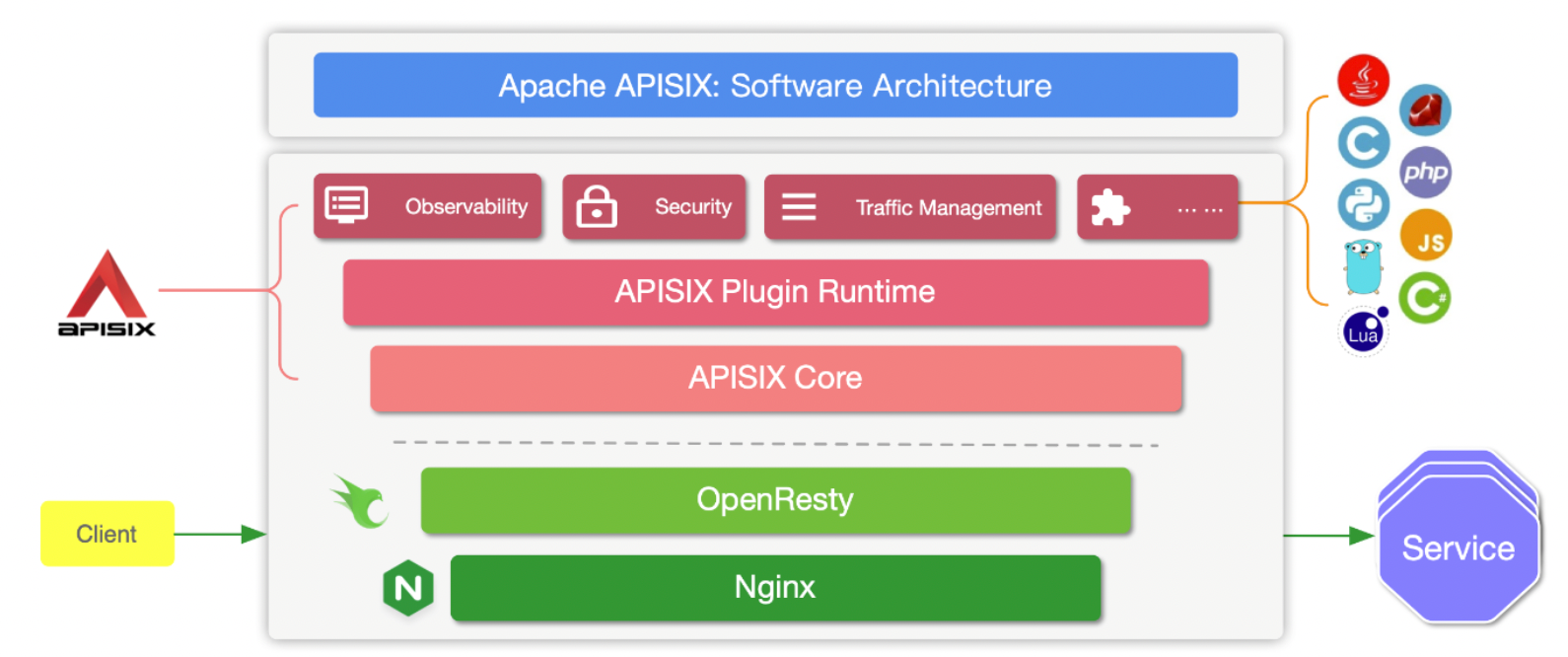
Kong内部架构：



## 2.3 APISIX

APISIX 是一个微服务API网关，具有高性能、可扩展性等优点。它基于 nginx（openresty）和 Lua 实现功能，借鉴了Kong的思路，将Kong底层的关系型数据库（Postgres）替换成了NoSQL型的 etcd，这使得 APISIX 相较于 Kong 在性能上有了很大提升，在启用各类插件的情况下，Apache APISIX 的性能是 Kong 的 10 倍，且Apisix是100%开源的，它的功能和Kong收费版的功能相当。

APISIX架构图:

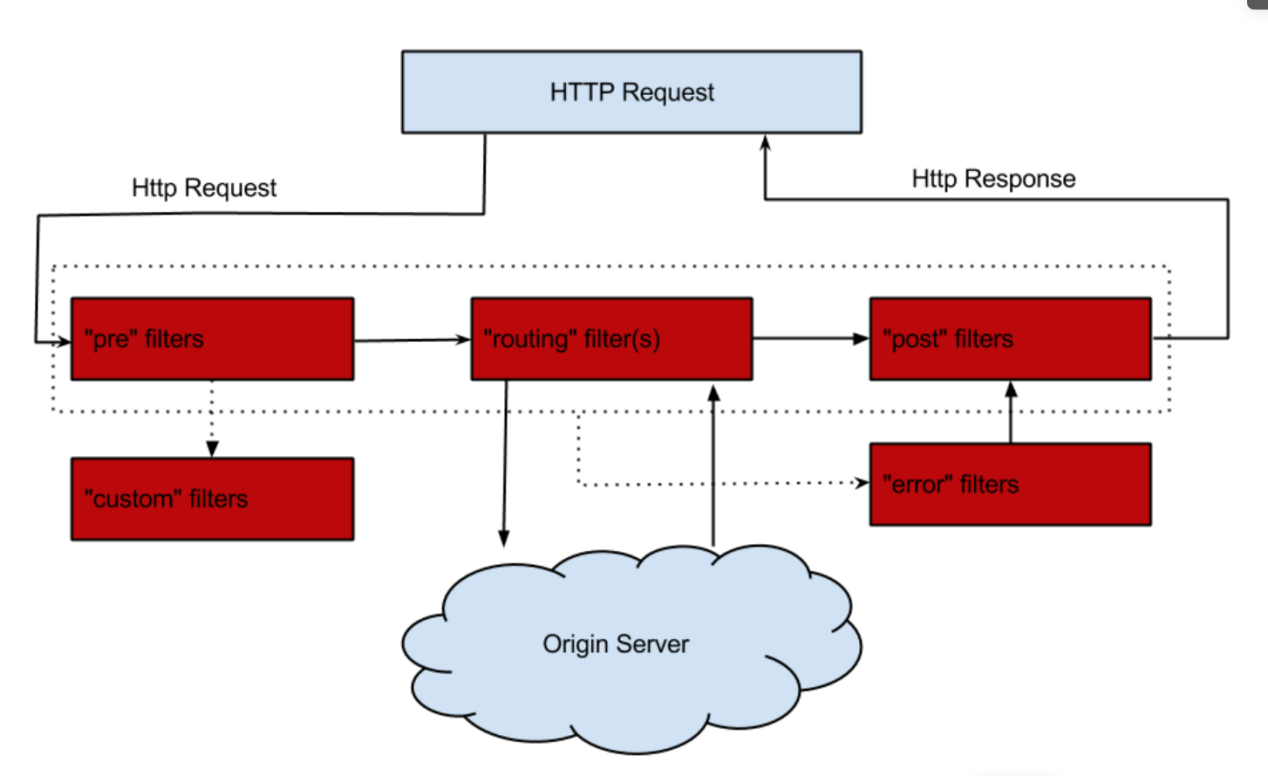


## 2.4 zuul1

zuul1是netflix开源的，使用了一系列不同类型的过滤器，被构建来支持动态路由、监视、弹性和安全性，使我们能够快速灵活地将功能应用到服务中。Zuul采用同步阻塞架构，依赖多线程来支持吞吐量的增长，性能较低。Zuul使用了一系列不同类型的过滤器，使我们能够快速灵活地将功能应用到服务中。

zuul1主要提供了pre、routing、post、error四种类型的过滤器作为zuul1处理的核心逻辑。

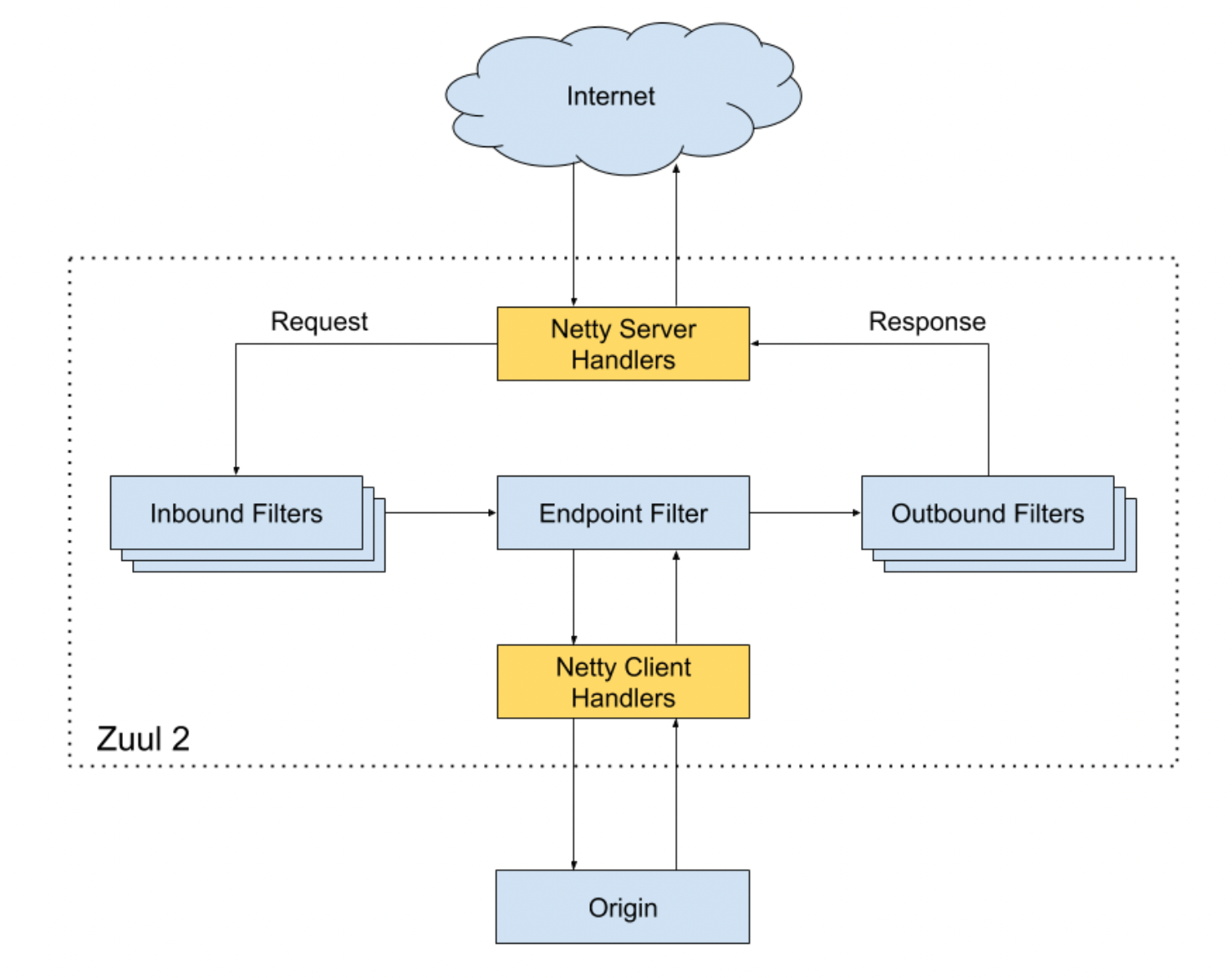
zuul1请求的生命周期:



## 2.5 zuul2

zuul2是netflix对通用API网关zuul的转型。zuul原本是采用阻塞架构，转型后叫zuul2,采用异步非阻塞架构。zuul2与zuul1在架构方面主要区别在于zuul2运行在异步非阻塞的框架上，比如netty，zuul1只能通过依赖线程来支持吞吐量的增长。

zuul2架构图：



上图是zuul2的架构图，和zuul1没有本质区别，主要有以下区别：

前端使用netty server代替servlet，目的是支持前端异步。后端使用netty client代替传统的http client，目的是支持后端异步。

过滤器换了一下名字，用Inbound Filters代替Pre-routing Filters，用Endpoint Filter代替Routing Filter，用Outbound Filters代替Post-routing Filters。

## 2.6 springcloud gateway

SpringCloud Gateway 是 Spring Cloud 的一个全新项目，该项目是基于 Spring 5.0，Spring Boot 2.0 和 Project Reactor 等技术开发的网关，它旨在为微服务架构提供一种简单有效的统一的 API 路由管理方式。

SpringCloud Gateway 作为 Spring Cloud 生态系统中的网关，目标是替代 Zuul，在Spring Cloud 2.0以上版本中，没有对新版本的Zuul 2.0以上最新高性能版本进行集成，仍然还是使用的Zuul 2.0之前的非Reactor模式的老版本。而为了提升网关的性能，SpringCloud Gateway是基于WebFlux框架实现的，而WebFlux框架底层则使用了高性能的Reactor模式通信框架Netty。

Spring Cloud Gateway 的目标，不仅提供统一的路由方式，并且基于 Filter 链的方式提供了网关基本的功能，例如：安全，监控/指标，和限流。

**Spring Cloud Gateway 底层使用了高性能的通信框架Netty**。

SpringCloud官方，对SpringCloud Gateway 特征介绍如下：

（1）基于 Spring Framework 5，Project Reactor 和 Spring Boot 2.0

（2）集成 Hystrix 断路器

（3）集成 Spring Cloud DiscoveryClient

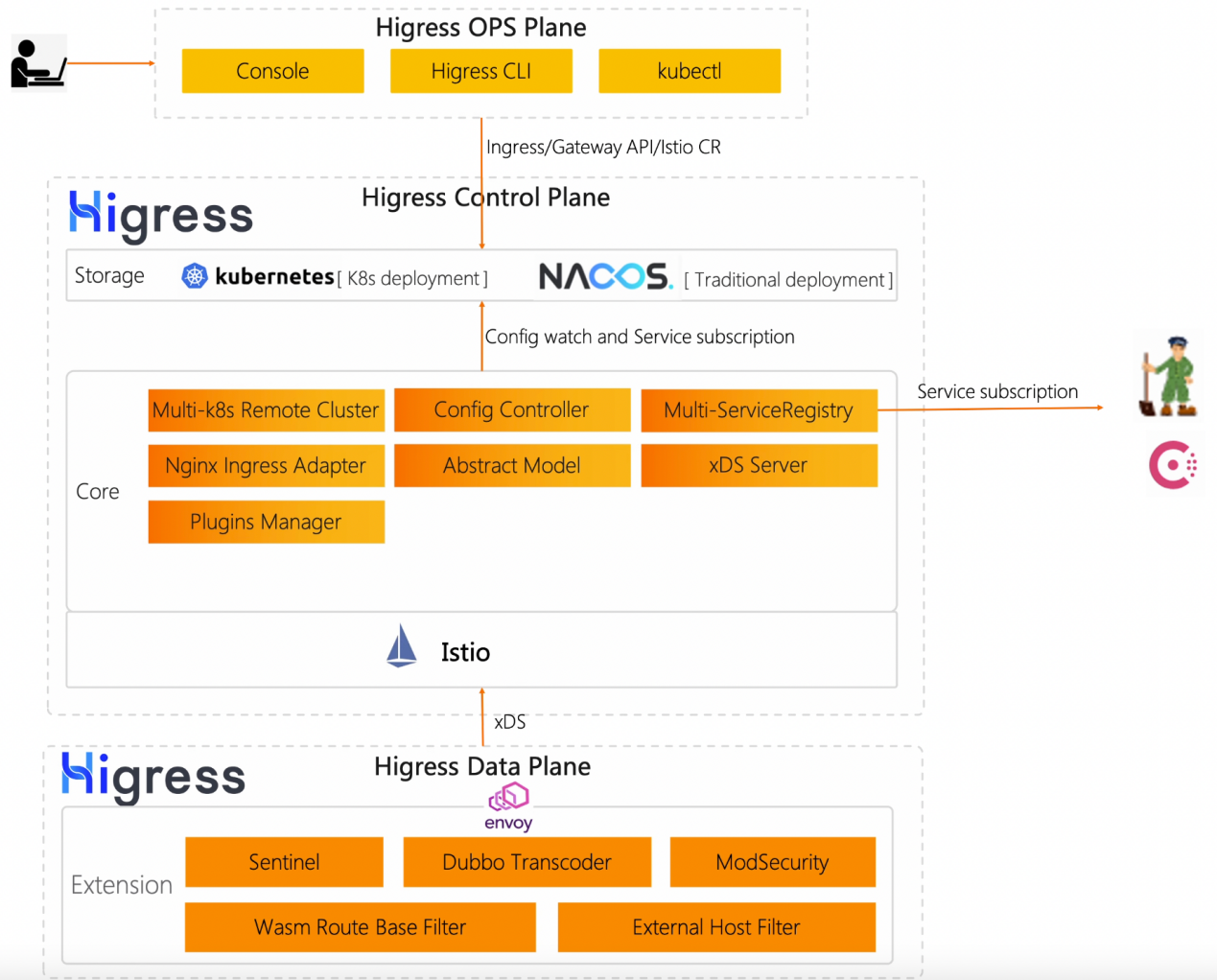
（4）Predicates 和 Filters 作用于特定路由，易于编写的 Predicates 和 Filters

（5）具备一些网关的高级功能：动态路由、限流、路径重写

从以上的特征来说，和Zuul的特征差别不大。SpringCloud Gateway和Zuul主要的区别，还是在底层的通信框架上。

## **2.7 Higress**

Higress是基于阿里内部的Envoy Gateway实践沉淀、以开源Istio + Envoy为核心构建的下一代云原生网关，实现了流量网关 + 微服务网关 + 安全网关三合一的高集成能力，深度集成Dubbo、Nacos、Sentinel等微服务技术栈，能够帮助用户极大的降低网关的部署及运维成本且能力不打折；在标准上全面支持Ingress与Gateway API，积极拥抱云原生下的标准API规范；同时，Higress Controller也支持Nginx Ingress平滑迁移，帮助用户零成本快速迁移到Higress。



Higress定位:

在虚拟化时期的微服务架构下，业务通常采用流量网关 + 微服务网关的两层架构，流量网关负责南北向流量调度和安全防护，微服务网关负责东西向流量调度和服务治理，而在容器和 K8s 主导的云原生时代，Ingress 成为 K8s 生态的网关标准，赋予了网关新的使命，使得流量网关 + 微服务网关合二为一成为可能。

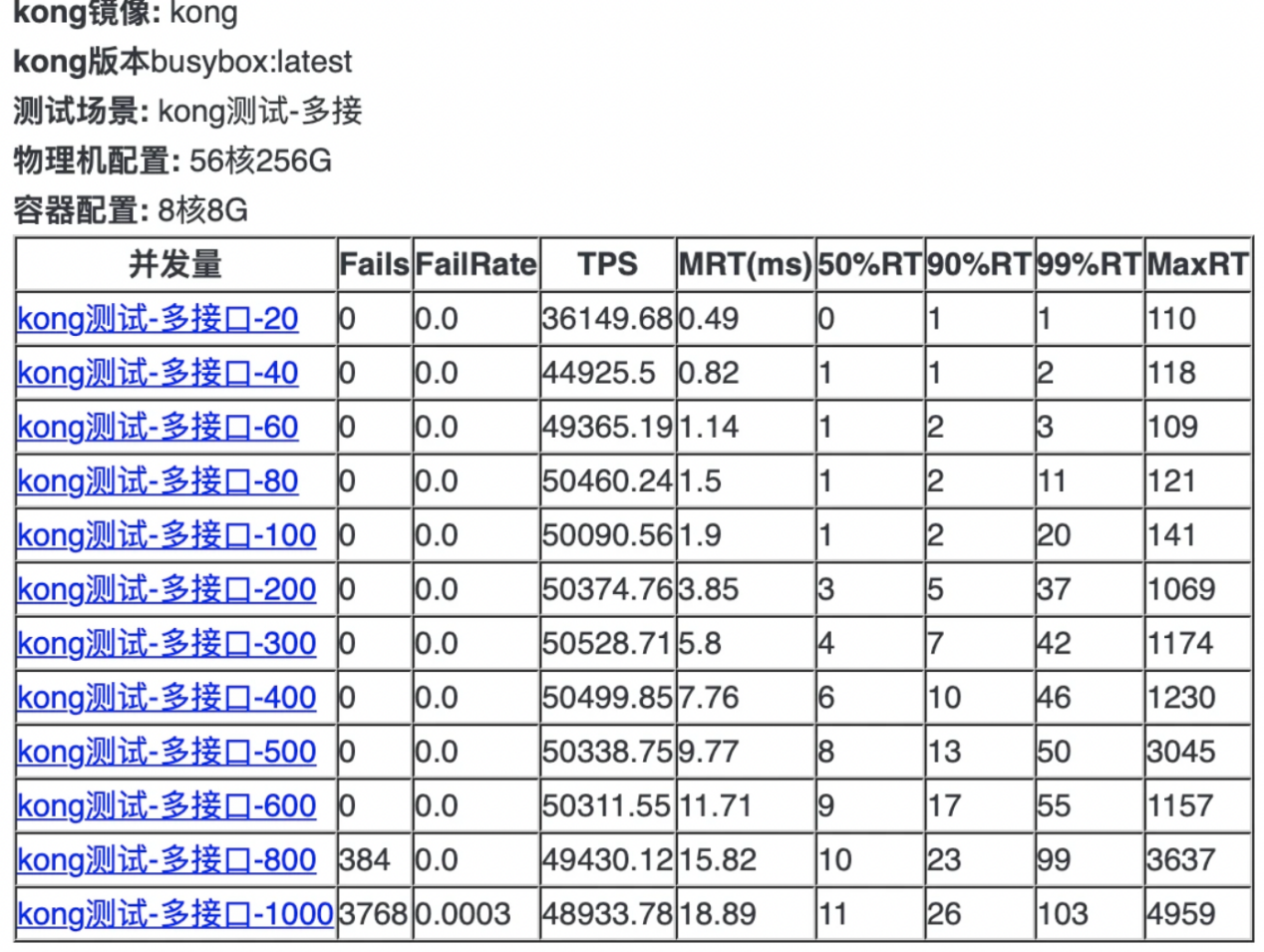
作为面向南北向的公网网关，使用Waf防护异常流量是很常规的需求，而且随着互联网环境变得越来越复杂，用户对防护的诉求是持续增强的，常规做法是将流量先接入Waf安全网关，过滤后再将流量转发给流量网关，最后到达微服务网关；Higress希望通过内置Waf模块，使得用户的请求链接只经过Higress就可以同时完成Waf防护、流量分发、微服务治理，既可以提升链路RT，也可以降低网关的运维复杂度。因此Higress实现了流量网关 + 微服务网关 + 安全网关三合一的高集成能力。

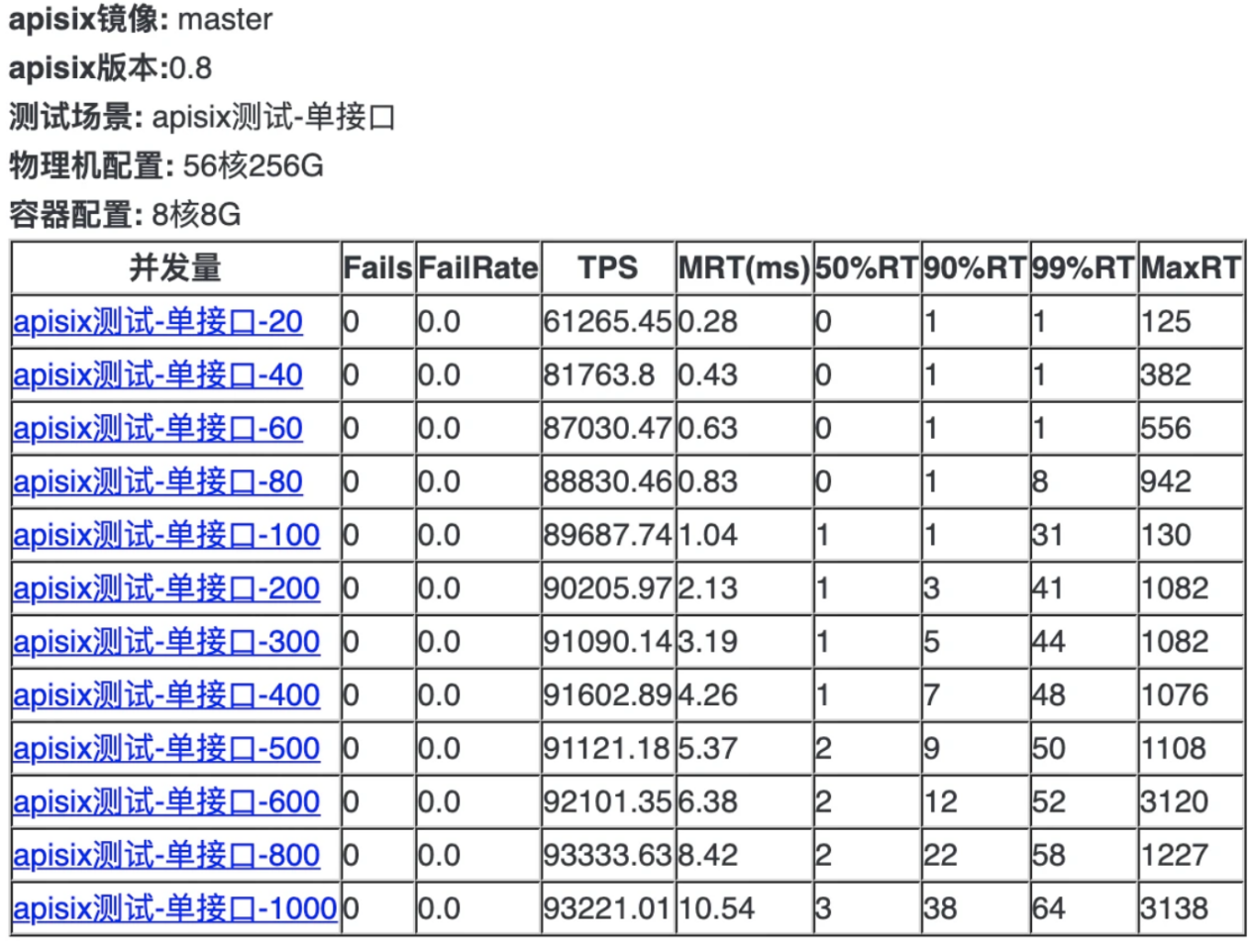
# 主要网关对比

APISIX与Kong的功能对比:

|  |  |  |
| --- | --- | --- |
| 功能 | APISIX | KONG |
| 反向代理和路由 | 支持 | 支持 |
| 负载均衡 | 支持 | 支持 |
| 身份验证和授权 | 支持 | 支持 |
| IP列表白名单/黑名单 | 支持 | 支持 |
| 限速和流控 | 支持 | 支持 |
| 请求变形 | 支持 | 支持 |
| 版本控制 | 支持 | 支持 |
| 断路器 | 支持 | 支持 |
| 多协议支持 | 支持 | 支持 |
| 缓存 | 支持 | 支持 |
| 数据库存储 | etcd | Postgres/Cassandra |

APISIX与Kong的性能对比:





几种业务网关的对比:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 网关 | 限流 | 鉴权 | 监控 | 易用性 | 可维护性 | 成熟度 | 性能 |
| zuul1 | filter中实现 | filter中实现 | filter中实现 | 简单易用 | 较差 | 开源时间长，资料较多 | 基于阻塞IO，性能较差 |
| zuul2 | 可以通过配置文件配置集群限流和单服务器限流亦可通过filter实现限流扩展 | filter中实现 | filter中实现 | 参考资料较少 | 较差 | 资料较少 | 基于netty异步非阻塞IO性能较为客观 |
| springcloud gateway | 可以通过IP，用户，集群限流，提供了相应的接口进行扩展 | 普通鉴权、auth2.0 | Gateway Metrics Filter | 简单易用 | spring系列可扩展强，易配置 可维护性好 | spring社区成熟，但gateway资源较少 | 基于netty异步IO性能与Zuul2近似 |

# 网关选型建议

基于以上数据进行对比:

1. 常见的流量网关主要有OpenRestry、Kong、APISIX以及阿里的Higress等。

OpenRestry、Kong、APISIX都是以ngnix+lua实现，Higress是以开源istio+envoy为核心构建的下一代云原生网关。在性能方面APISIX要比OpenRestry以及Kong有显著提升。

**如果考虑开源的流量网关可以选择APISIX或者Kong，**

**如果倾向于一些商业化团队支持可以采用阿里的Higress/APISIX商业版。**

1. 常见的业务网关主要有zuul1、zuul2、springCloud gateway以及阿里的Higress，前三者都是开源的，在性能方面zuul2和Spring Cloud Gateway相比于zuul1来说有显著提升，但是编程和阅读相对于zuul1来说较为有难度。Spring Cloud Gateway相比于zuul2来说整体开源支持较为好。目前在各大银行的实践中，Spring Cloud Gateway也是使用较多的API网关（中国银行、邮储银行、兴业银行等）。

**对于业务网关建议选择Spring Cloud Gateway**，相对于上层流量网关来说业务网关一般更与业务有耦合度，一般的开源网关提供相应的扩展点即可满足业务方面的定制化需求，一般不需要商业化的业务网关。