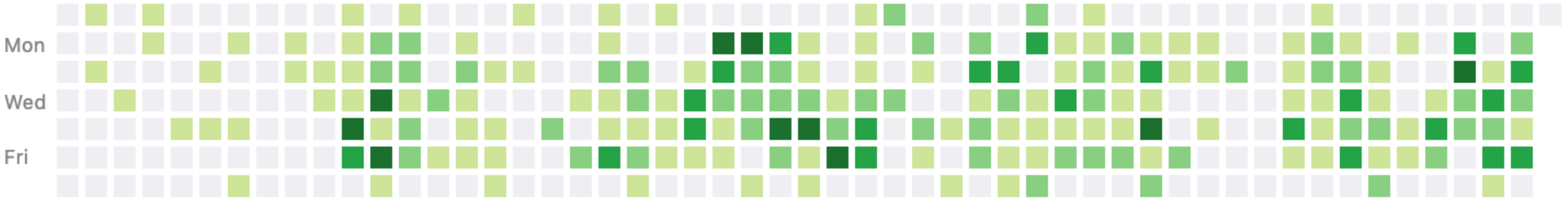
API 网关的选型测试和持续集成

温铭 https://github.com/moonming



关于我

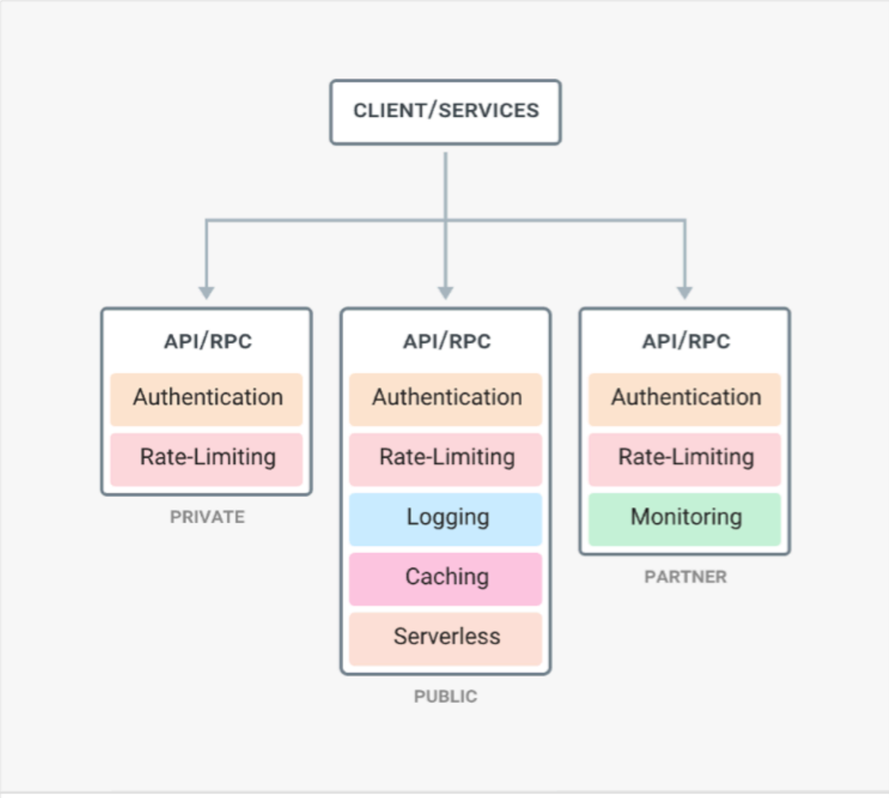
- OpenResty 软件基金会发起人、主席
- 《OpenResty 从入门到实战》专栏作者
- 曾在 OpenResty Inc. 担任合伙人
- 曾在奇虎 360 担任架构师，开源委员会发起人、委员

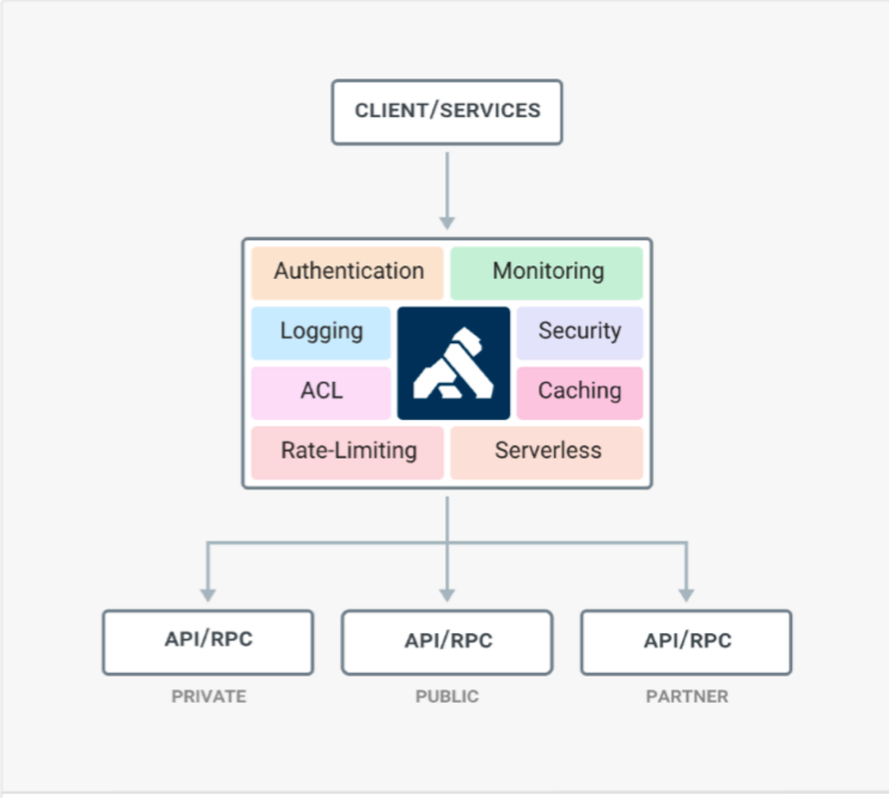
OpenResty 软件基金会

- 注册在香港的慈善组织：免税、接受政府监管
- 锤子科技：100 万人民币
- fred 吴：从去年 12 月起，持续每个月捐款 1000 人民币
- 感谢资助过 OpenResty 大会的每家公司：
- 奇虎 360、腾讯、bearychat、蓝汛、七牛、Coding、云帆加速、IT 大咖说、网易云、又拍云、Kong

SOFTWARE FOUNDATION

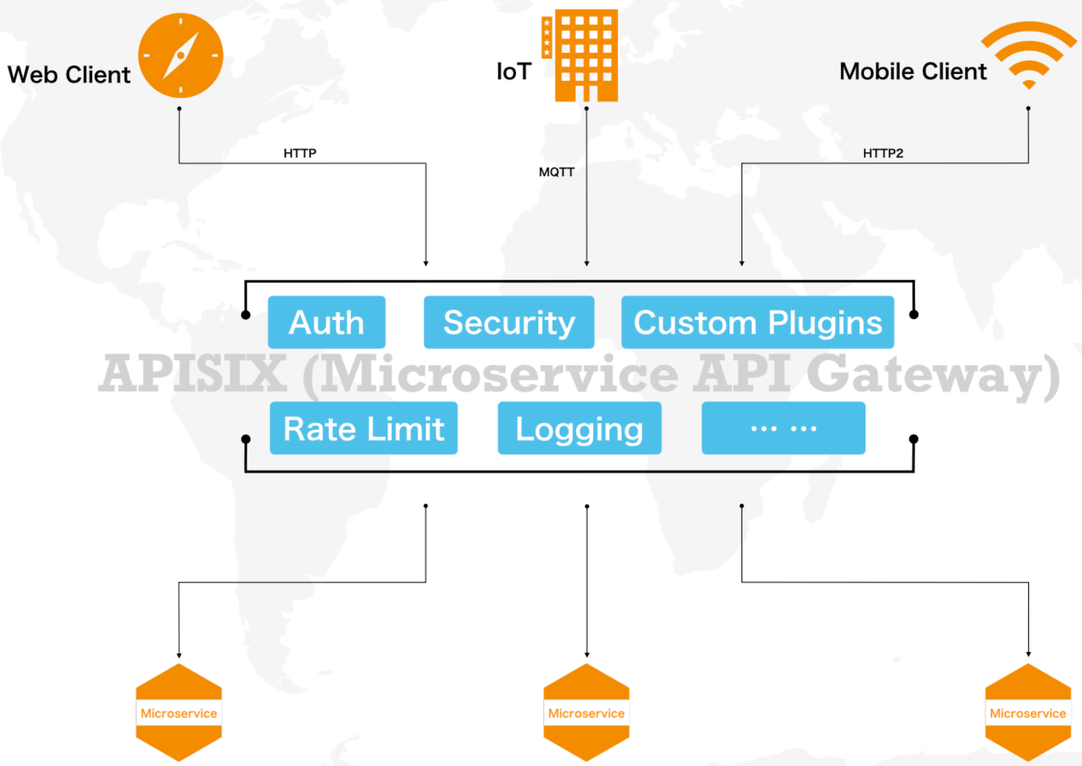
什么是 API 网关?





OPENRESTY
SOFTWARE FOUNDATION

API 网关的位置

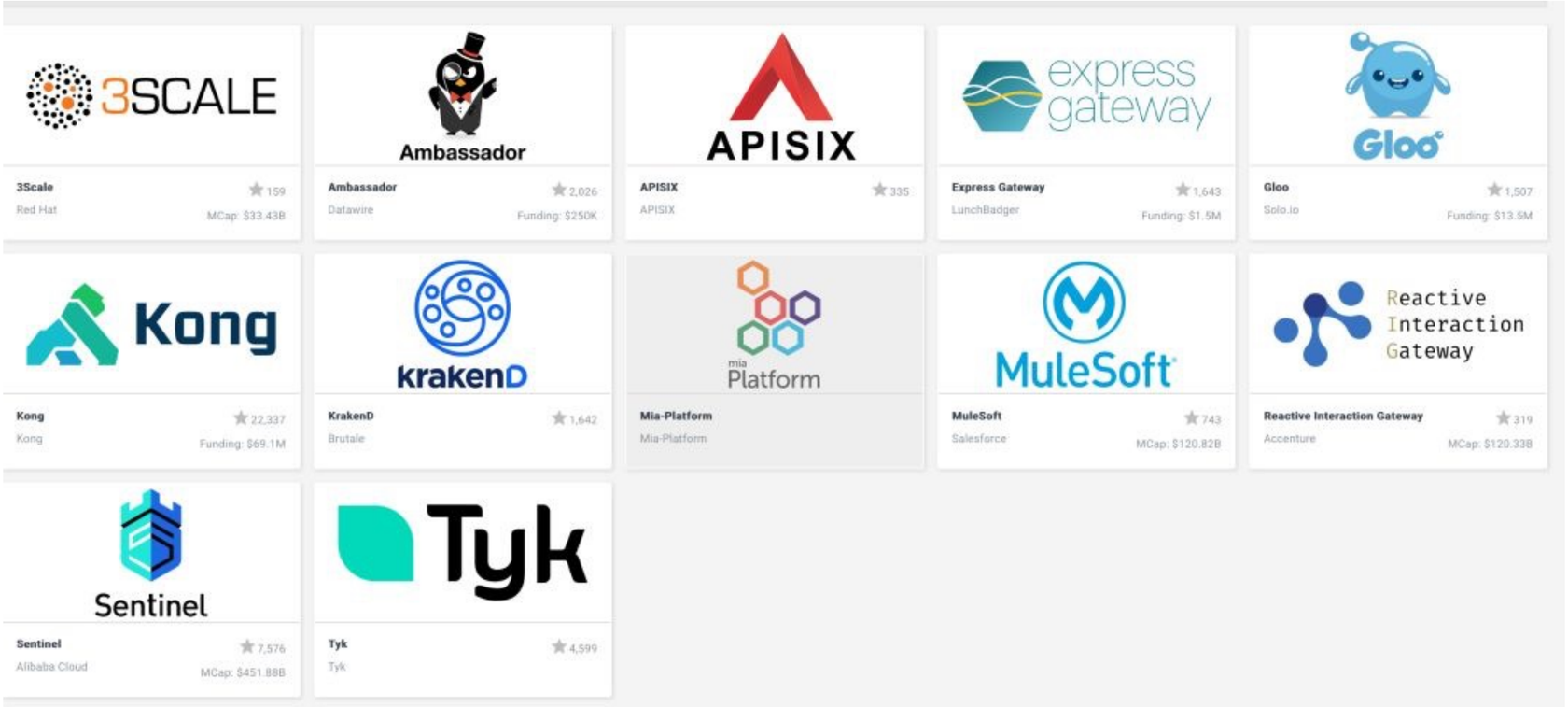


选型

了解行业：Gartner 报告

OPENRESTY
SOFTWARE FOUNDATION

CNCF 全景图

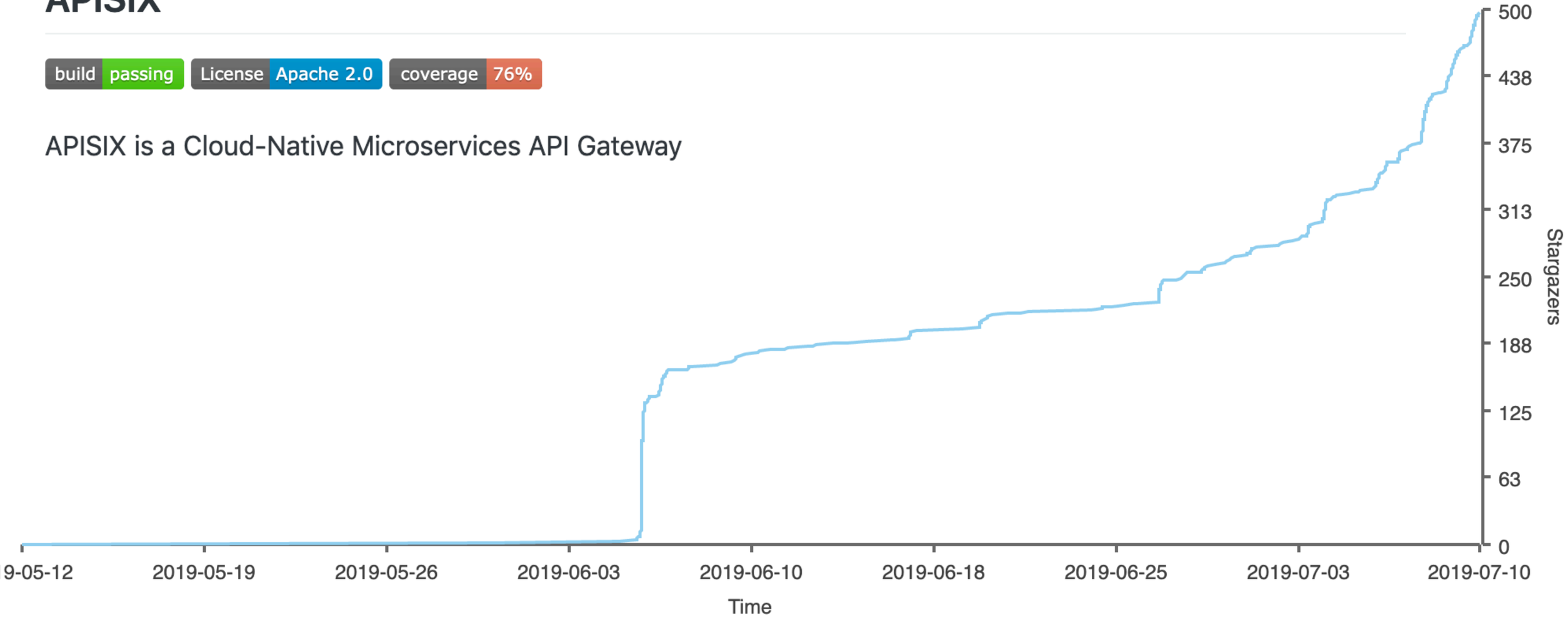


OPENRESTY
SOFTWARE FOUNDATION

APISIX

build passing | License Apache 2.0 | coverage 76%

APISIX is a Cloud-Native Microservices API Gateway



OPENRESTY
SOFTWARE FOUNDATION

产品选型

产品/特性	优势	劣势	二次开发难度
apigee	谷歌背书、 全生命周期	性能差、不开源	不支持
Kong	开源社区活跃、 产品思路清晰	代码量大、封装多、 依赖关系型数据库、 底层架构跟不上趋势	支持，难度大
APISIX	基于 etcd、代码易 读、插件热加载	开源时间短	支持，难度小

如何做网关的技术选型?

SOFTWARE FOUNDATION

API 网关的核心组件

- 路由：遍历、哈希、前缀树，以及他们的组合
- 插件：允许自定义、PDK（插件开发包）、热加载
- schema：参数校验、前后端配合
- 存储：关系型数据库、键值数据库、etcd

APISIX 的选型原则

- APISIX：云原生、高性能、开源的 API 网关
- 云原生友好
- 开发者第一：架构简洁、方便扩展、代码易读、部署简单
- 极致性能：时间复杂度、FFI 和 Lua 代码

APISIX 的选型

- 路由：lua-resty-r3，FFI
- 插件：灵感来自 Kong，大幅度简化编写难度，热加载
- schema：rapidjson， json schema
- 存储：etcd， lua-resty-etcd

测试

测试驱动开发

- 开源项目的立足之本
- 没有专职 QA
- 提交功能代码的同时，必须有测试案例
- 代码覆盖率不能低于 70%

OpenResty 是怎么做的

- test::nginx：用 perl 编写的 DSL，专门用于 nginx 和 OpenResty 的单元测试
- OpenResty 和 lua-resty-* 模块都带有完整的测试案例集
- PR 跑单元测试
- 发版本跑完整回归测试、内存泄漏测试、fuzzing 测试

SOFTWARE FOUNDATION

APISIX 的实践

- 开发即测试
- 单元测试完全基于 test::nginx
- 代码风格检测：luackeck 和 lua-relang
- 代码覆盖率：luacov

- 合并 PR 的前提是跑完上述三种所有测试
- 发版本前跑性能测试，分析火焰图
- 定期跑 fuzzing 测试

test::nginx 上手难度大

- 并非只使用 Lua 编写
- 对 test::nginx、perl、OpenResty 不熟悉的话，很可能写不出来测试案例
- 一直是代码贡献者的拦路虎

OPENRESTY
SOFTWARE FOUNDATION

持续集成

OpenResty 的做法

- 不能自动化的测试，终将变成没有测试
- 花费在测试和 CI 上的工具和时间都是值得的
- travis CI
- 回归测试、打包、发布版本和发行包还依赖人的参与

APISIX 的做法

- travis CI
- coveralls.io
- docker 自动化打包（WIP）

OPENRESTY
SOFTWARE FOUNDATION

COVERAGE	Δ	FILE
100.0		...ravis/build/iresty/apisix/lua/apisix/core/json.lua
100.0		...is/build/iresty/apisix/lua/apisix/core/request.lua
100.0		...ome/travis/build/iresty/apisix/lua/apisix/core.lua
96.15		...travis/build/iresty/apisix/lua/apisix/core/ctx.lua
94.87		...s/build/iresty/apisix/lua/apisix/admin/plugins.lua
92.77		...vis/build/iresty/apisix/lua/apisix/core/schema.lua
92.31		...ravis/build/iresty/apisix/lua/apisix/core/http.lua
91.79		/home/travis/build/iresty/apisix/lua/apisix.lua
91.3		...travis/build/iresty/apisix/lua/apisix/core/log.lua
90.91		...travis/build/iresty/apisix/lua/apisix/consumer.lua

OPENRESTY
SOFTWARE FOUNDATION

打包很重要

- 打包工程师
- 辛苦活
- 心细、有耐心才行
- 原生的方式，还是 fpm
- 虚拟机还是 docker
- 手工还是自动化

SOFTWARE FOUNDATION

总结

- 资源少，不一定是坏事儿
- APISIX 的选型、测试和 CI 都是找“取巧”和自动化的方式
- 这三者很重要，比性能重要

Q&A

OPENRESTY

SOFTWARE FOUNDATION