前后端分离中 API 接口与数据 Mock 的思考与应用

吕伟

QCon

2016.10.20~22
上海·宝华万豪酒店

全球软件开发大会2016

[上海站]

购票热线：010-64738142
会务咨询：qcon@cn.infoq.com
赞助咨询：sponsor@cn.infoq.com
议题提交：speakers@cn.infoq.com
在线咨询（QQ）：1173834688

团·购·享·受·更·多·优·惠

7折 优惠（截至06月21日）
现在报名，立省2040元/张

关于我

- 现任职于美团大众基础终端组
- 在美团大众负责过考试系统的开发
- 曾负责百度音乐 FM，主站 webapp 的开发
- 在 Node 工程工具和自动化方面有较丰富的经验

联调的进化

模版时代

模版传递数据

项目高度耦合
项目间数据复用度低

前后端分离时代

串行开发时代

前端必须等待 API
开发完成后才能开发
开发者闲置率高

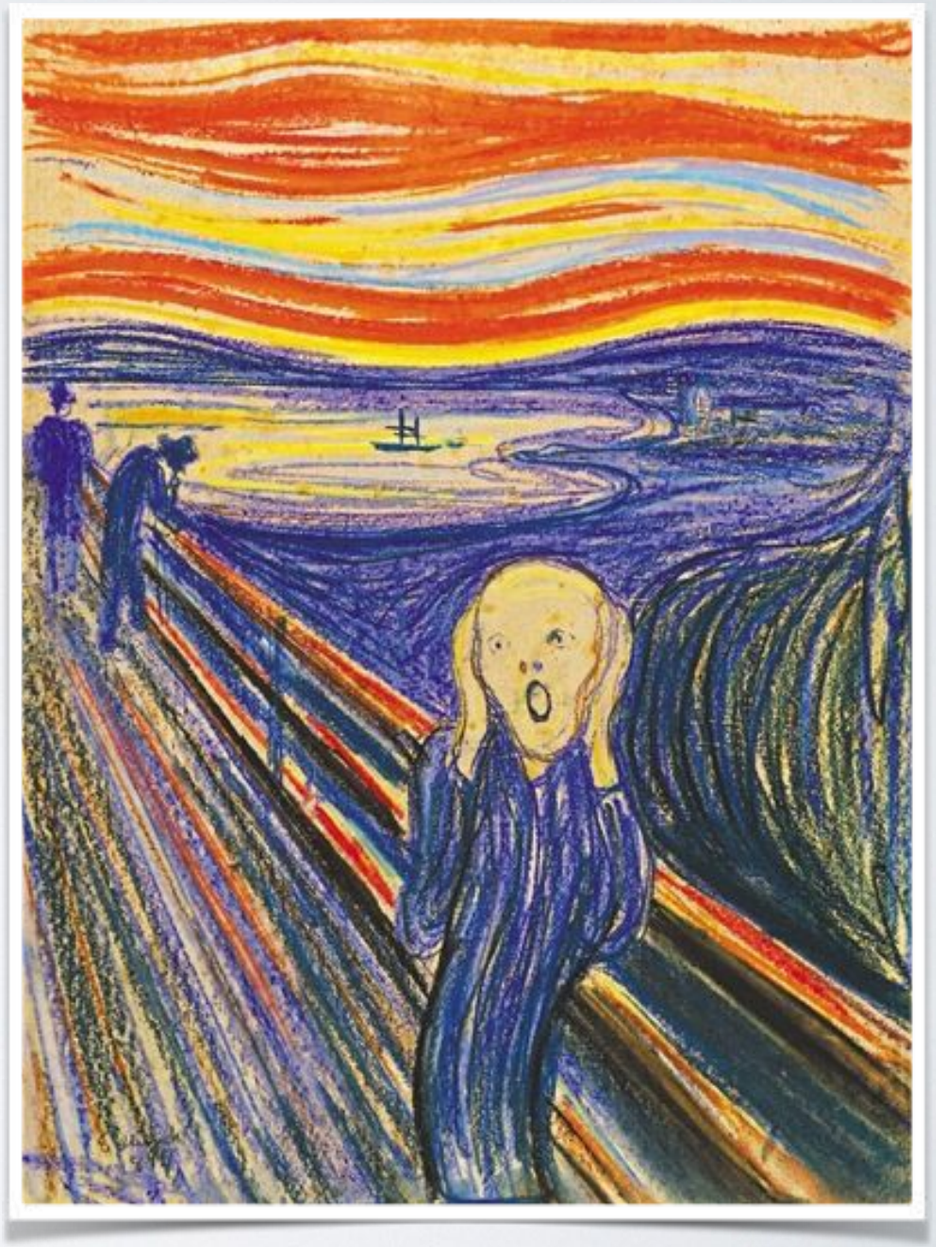
单项目并行开发时代

多项目间假数据复用率低
无法统一控制流程

多项目持续
并行开发时代

前后端并行开发时代的声音

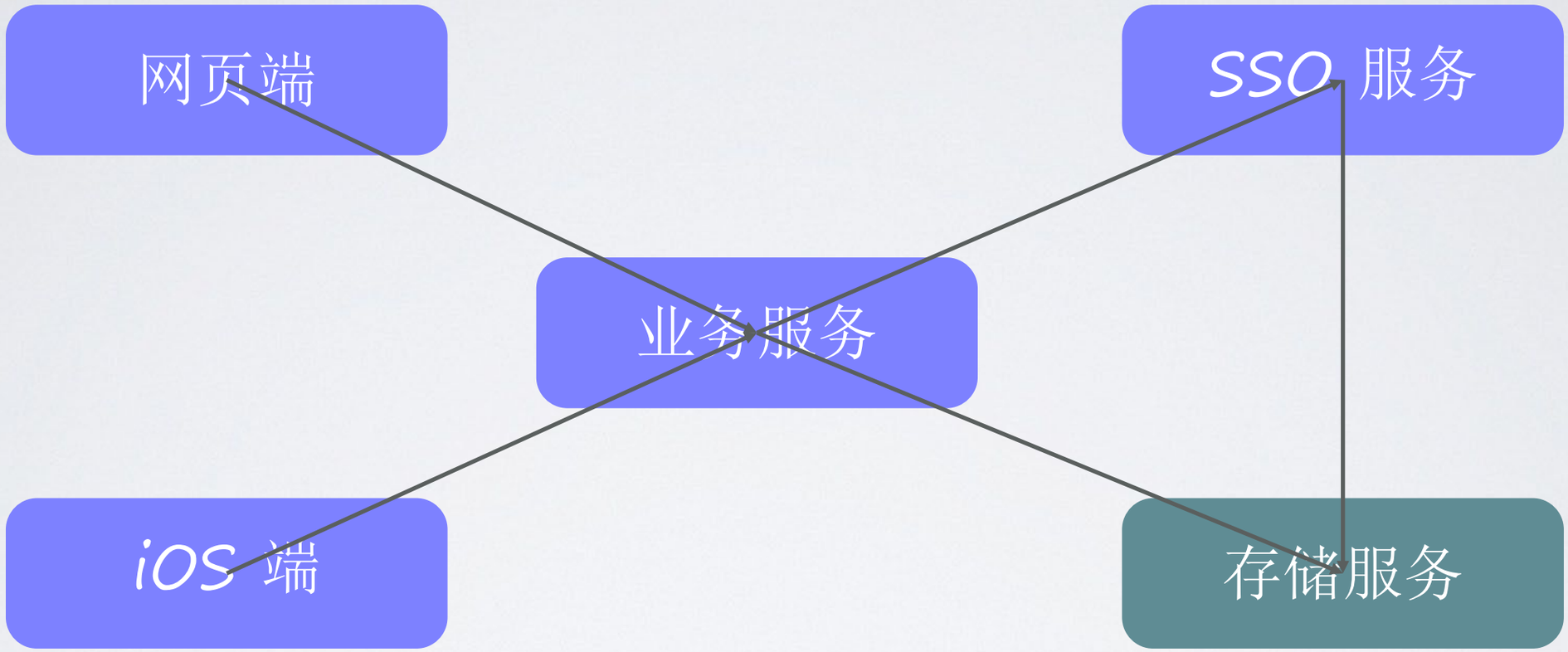
- 约定好的 API 又变了，半个月之后我才知道这件事
- 这么多 mock 方案我选哪个？
- 公司这个服务的 API 上哪查文档？
- 文档写起来好费时
- 写文档就够烦了，测试就免了吧



联调面临的现状

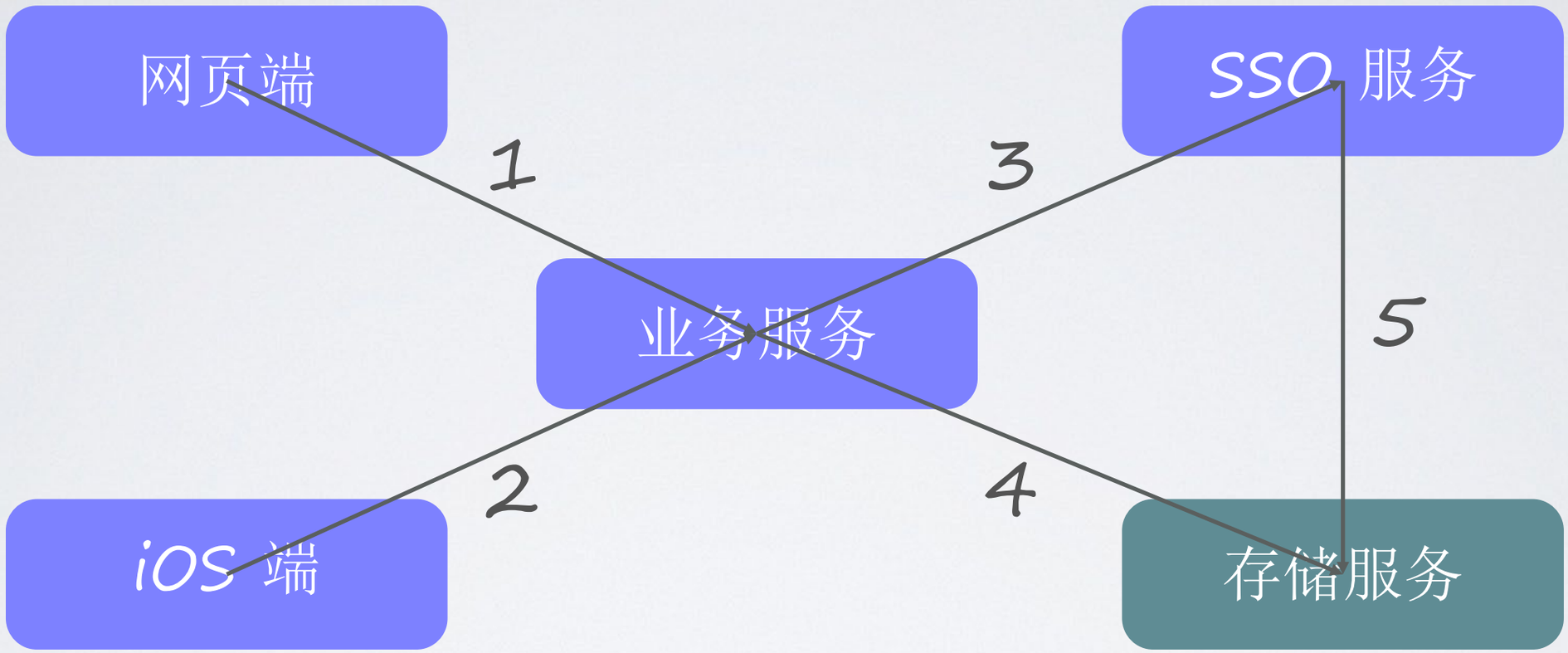
- 大量项目选择前后端完全分离
- microservices 大势所向
- 广义上后端通常也是作为其他服务的前端

广义的前端



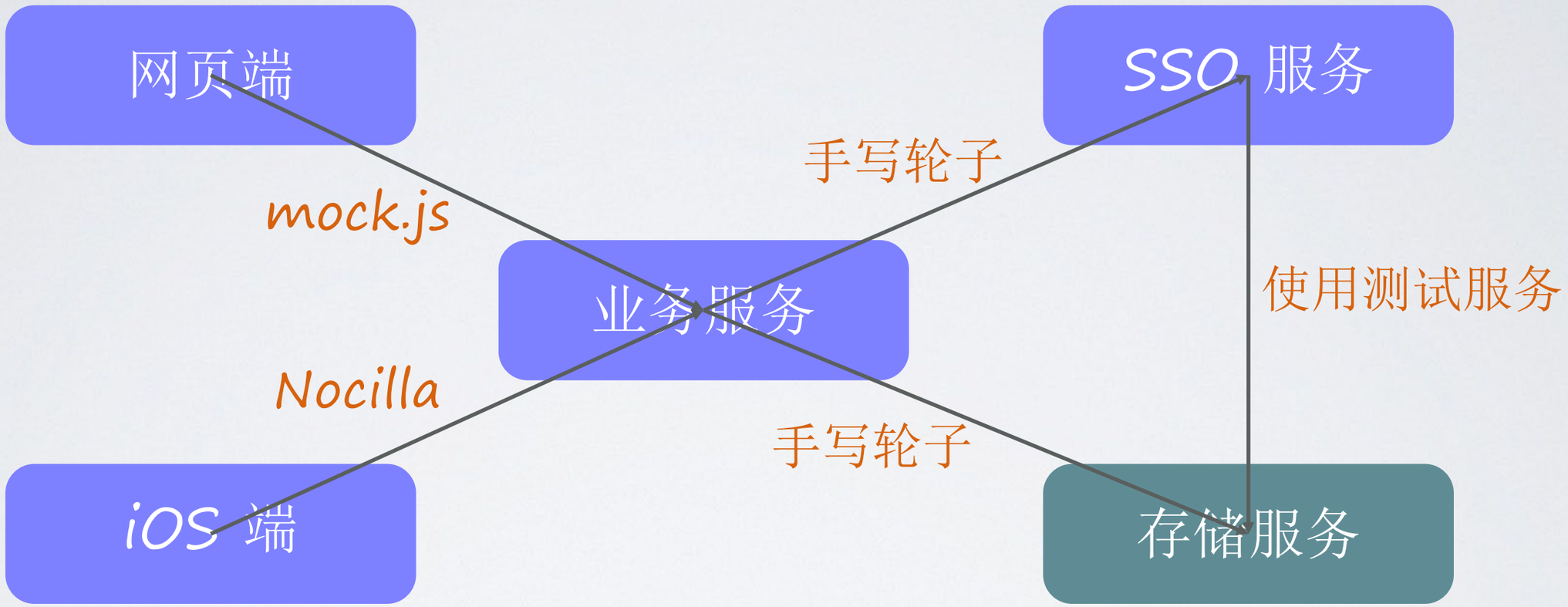
这里只有“存储服务”不作为广义前端

广义的前端都可能需要 mock



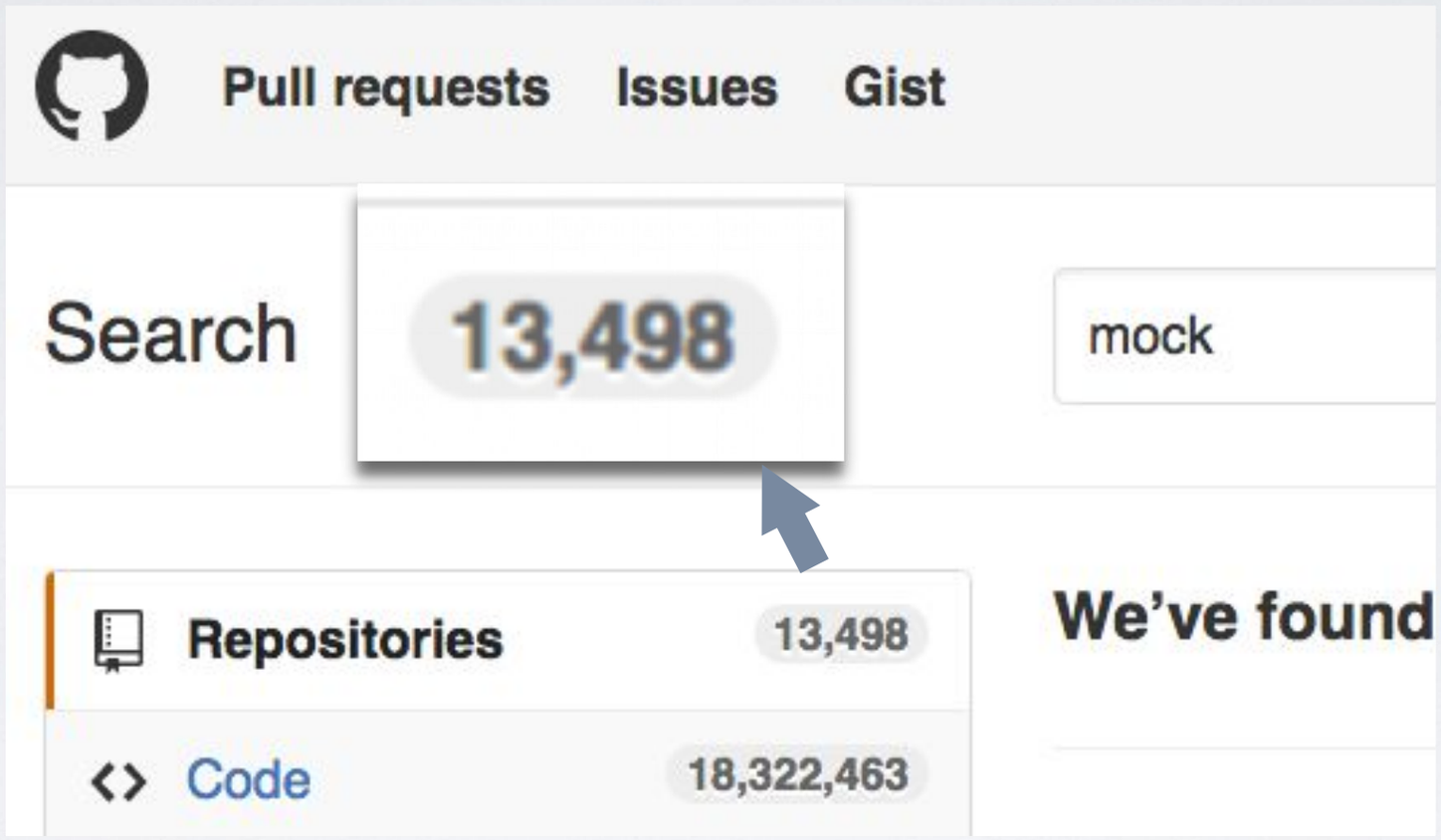
1，2，3，4，5 在开发初期都可能需要 mock

这是个充满轮子的世界



网页端、iOS 端几乎无法复用 API 假数据

轮子真的非常之多



比如 Github 上的 mock 相关仓库就有 1.3w 个之多

我不想造轮子，能不能用三方的方案？

	免费	开源	语言中立	流程控制	自动单元测试	沙盒化	文档生成	文档共享	多协议支持	侵入用户代码
mock.js	o	o	x	x	x	o	x	x	o	o

总结现有轮子的常见问题

1. 各工具设计语言不中立，难以共享定义

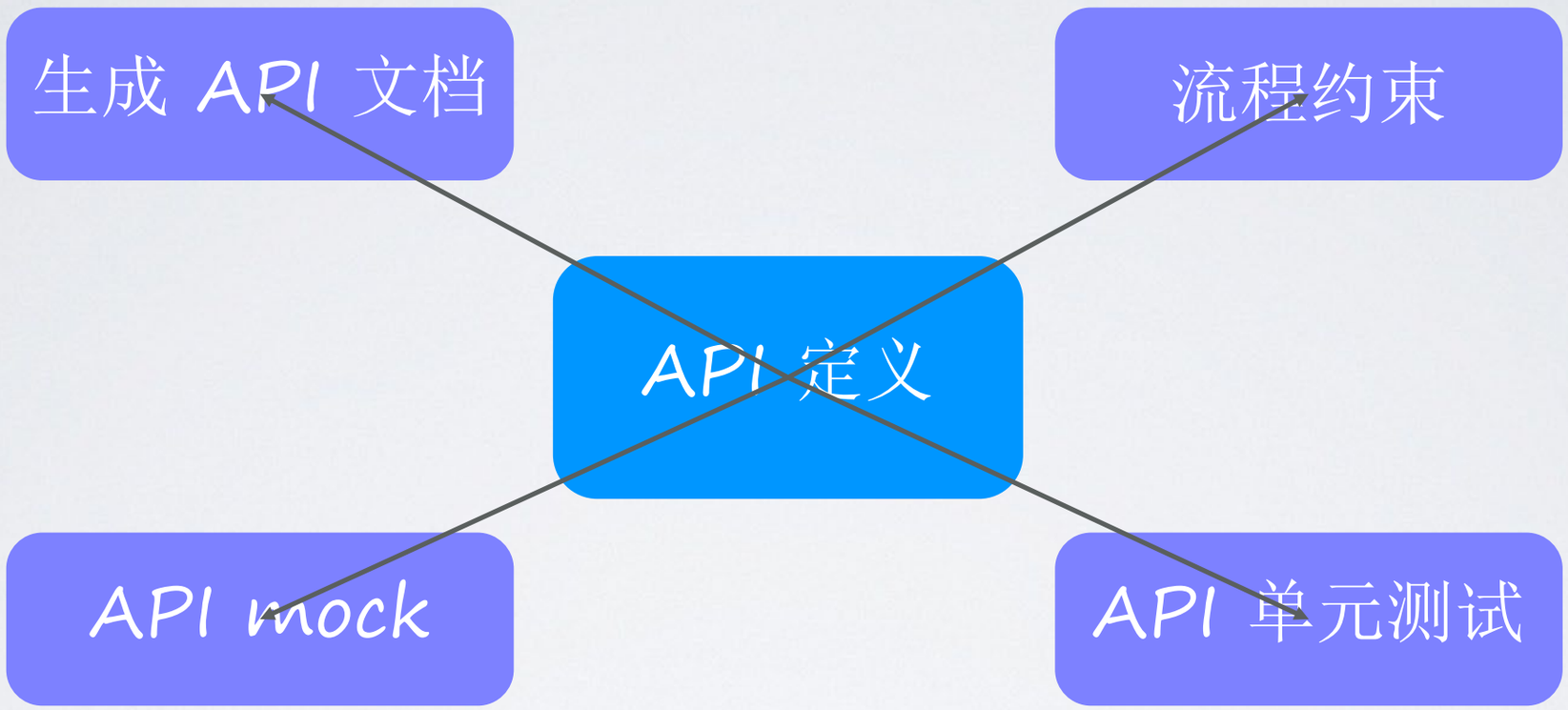
2. 大多难以做到沙盒化

3. 难以确保 API 的可用性

如何解决这些问题

1.定义一次 API 而可以处处复用

API 定义中心化



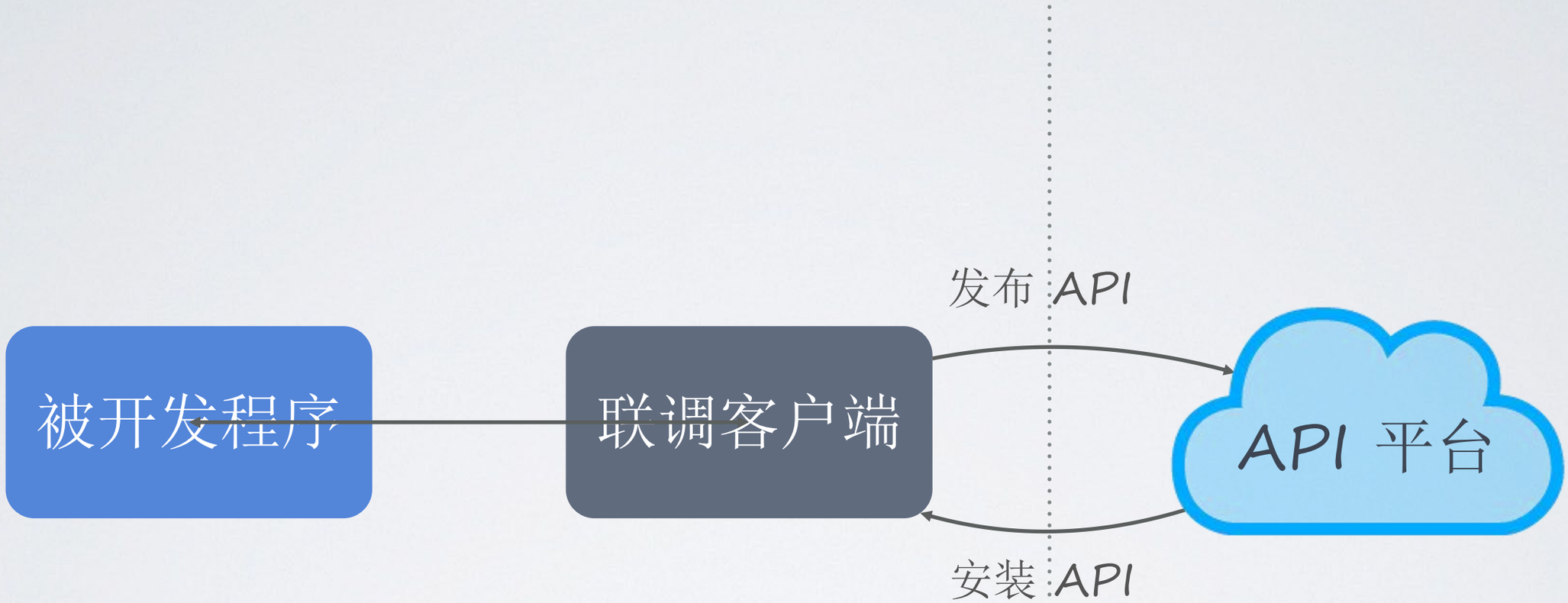
只需要定义一次 API 即可自动完成 API 文档生成， mock 假数据，单元测试 API，以及流程约束

2. 沙盒化

沙盒常常意味着 CS 架构

- 类似于 *github* 和 *git* 的关系
- 在封闭环境里也能持续测试
- 平台仅用于共享，开发时可以脱离平台

开发去中心化（沙盒化）

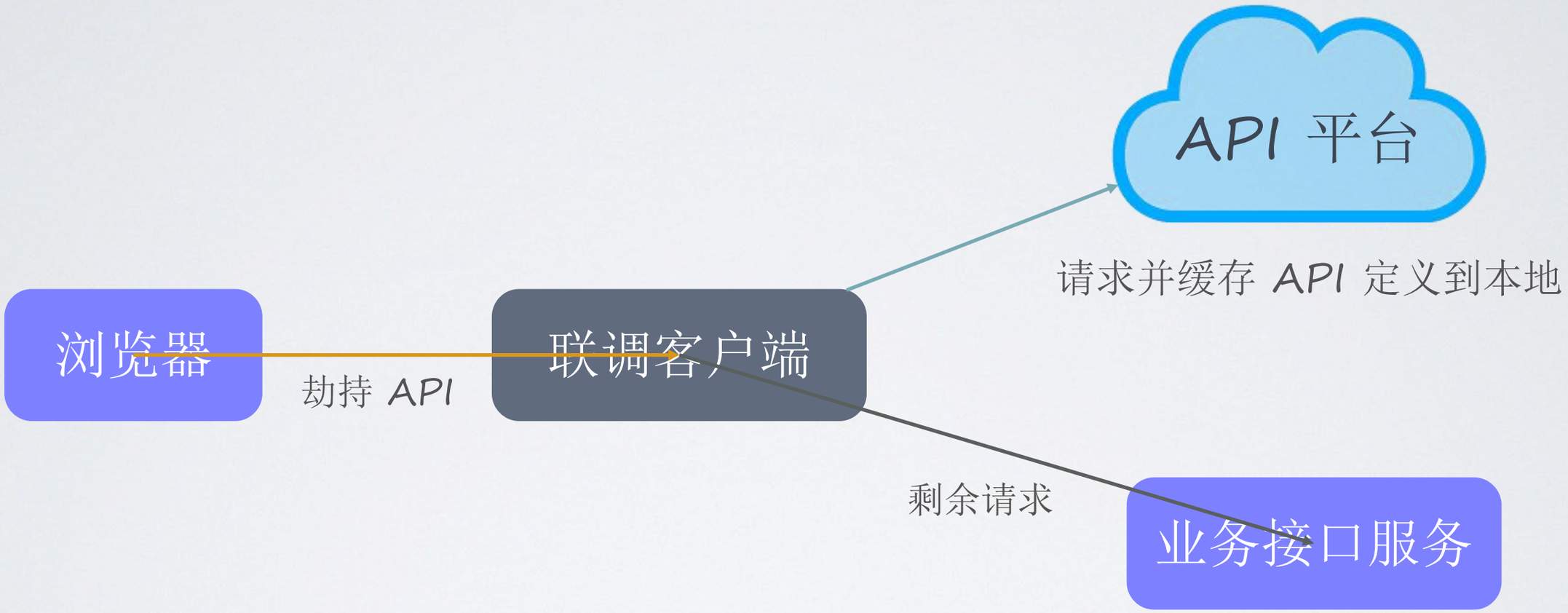


一般 mock 开发模式示意

代理流量

代码注入

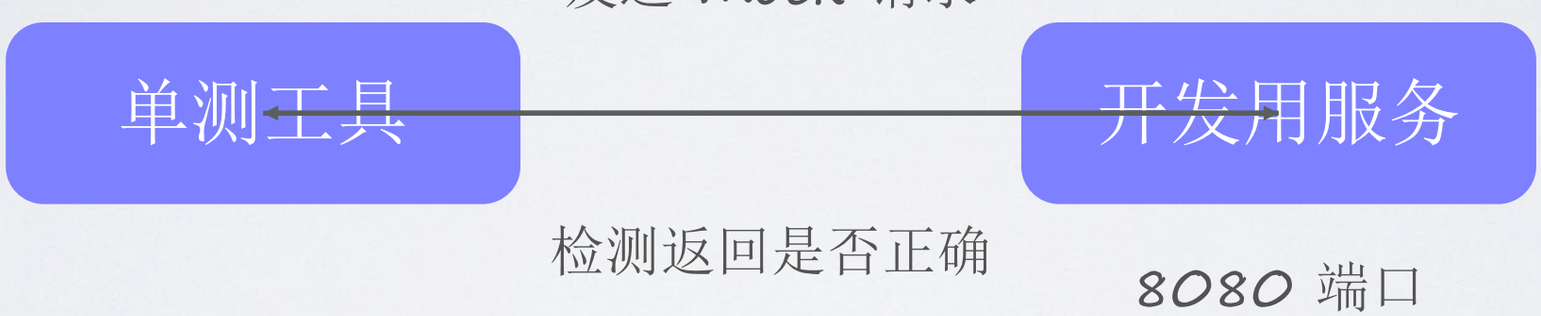
mock 接入



类似MIMT，客户端的接入完全透明，且语言中立

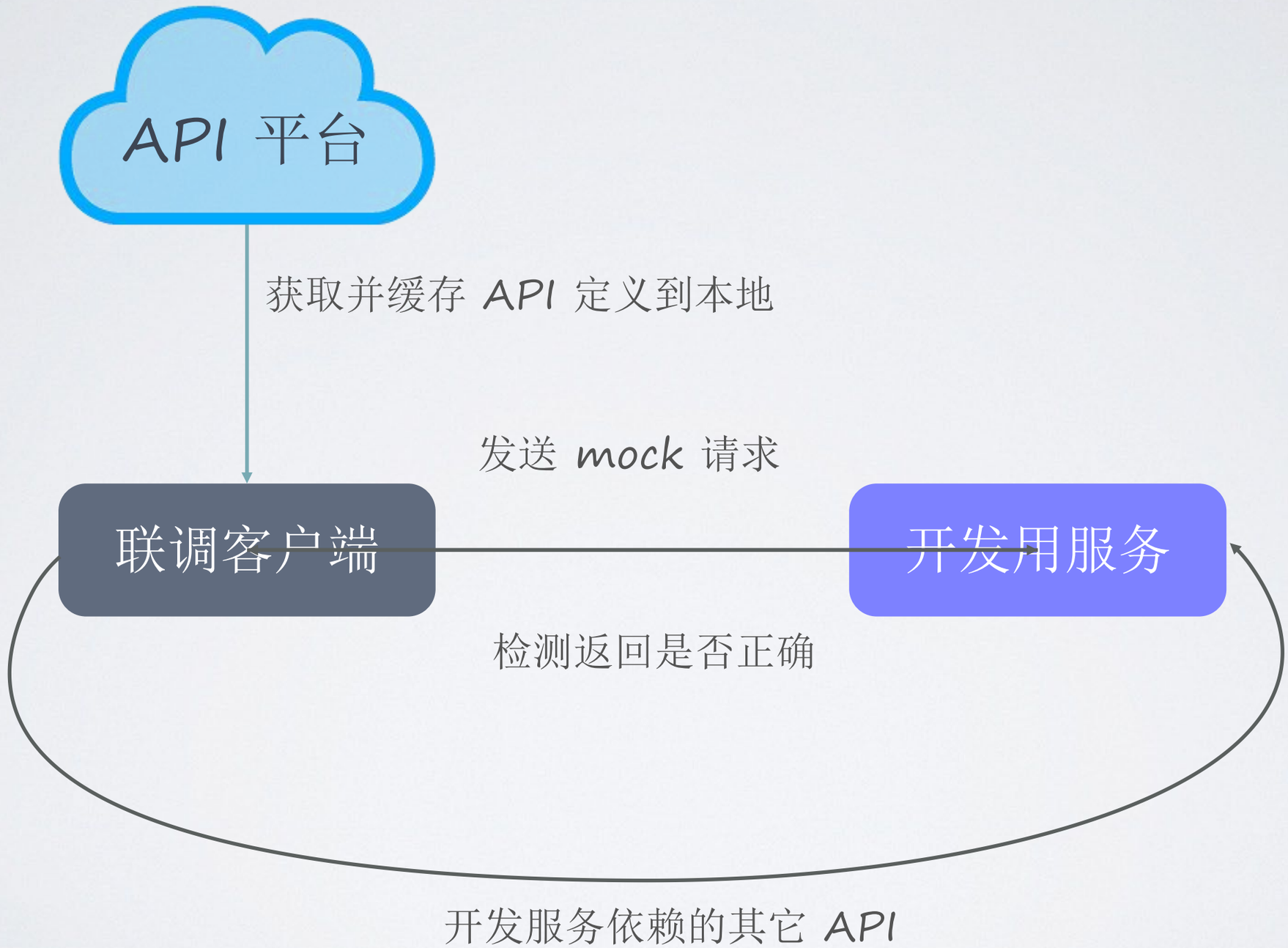
如何确保 API 的可用性

一般 RD 的 API 开发流程



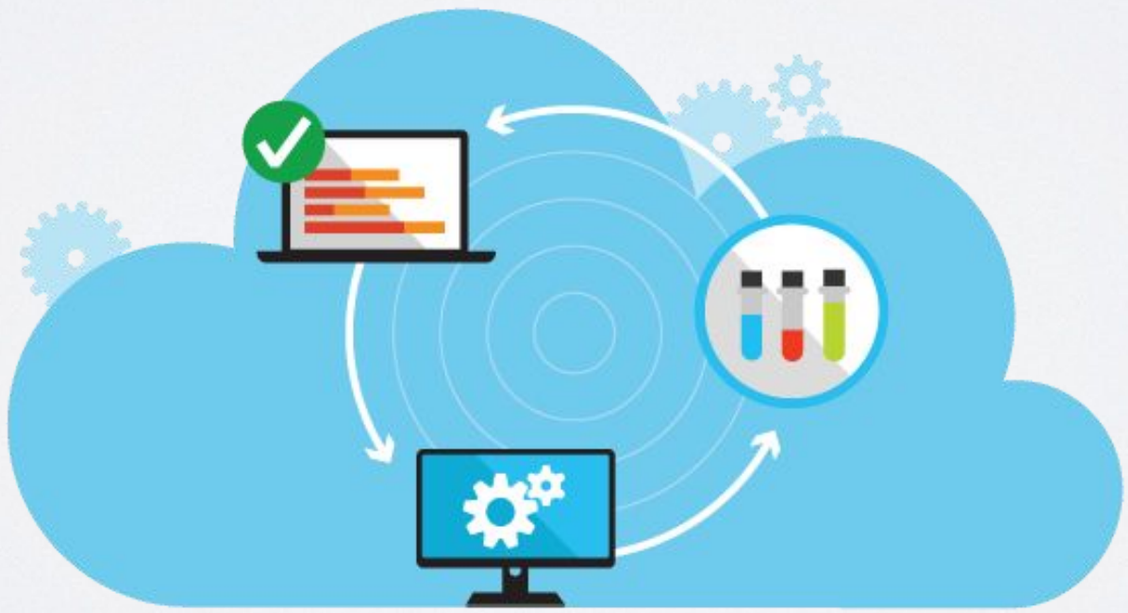
需要费时间自己写测试，甚至费时去调试测试代码本身

API 测试接入


API 平台
获取并缓存 API 定义到本地
发送 mock 请求
联调客户端
开发用服务
检测返回是否正确
开发服务依赖的其它 API

将 API 测试加入到 CI 过程

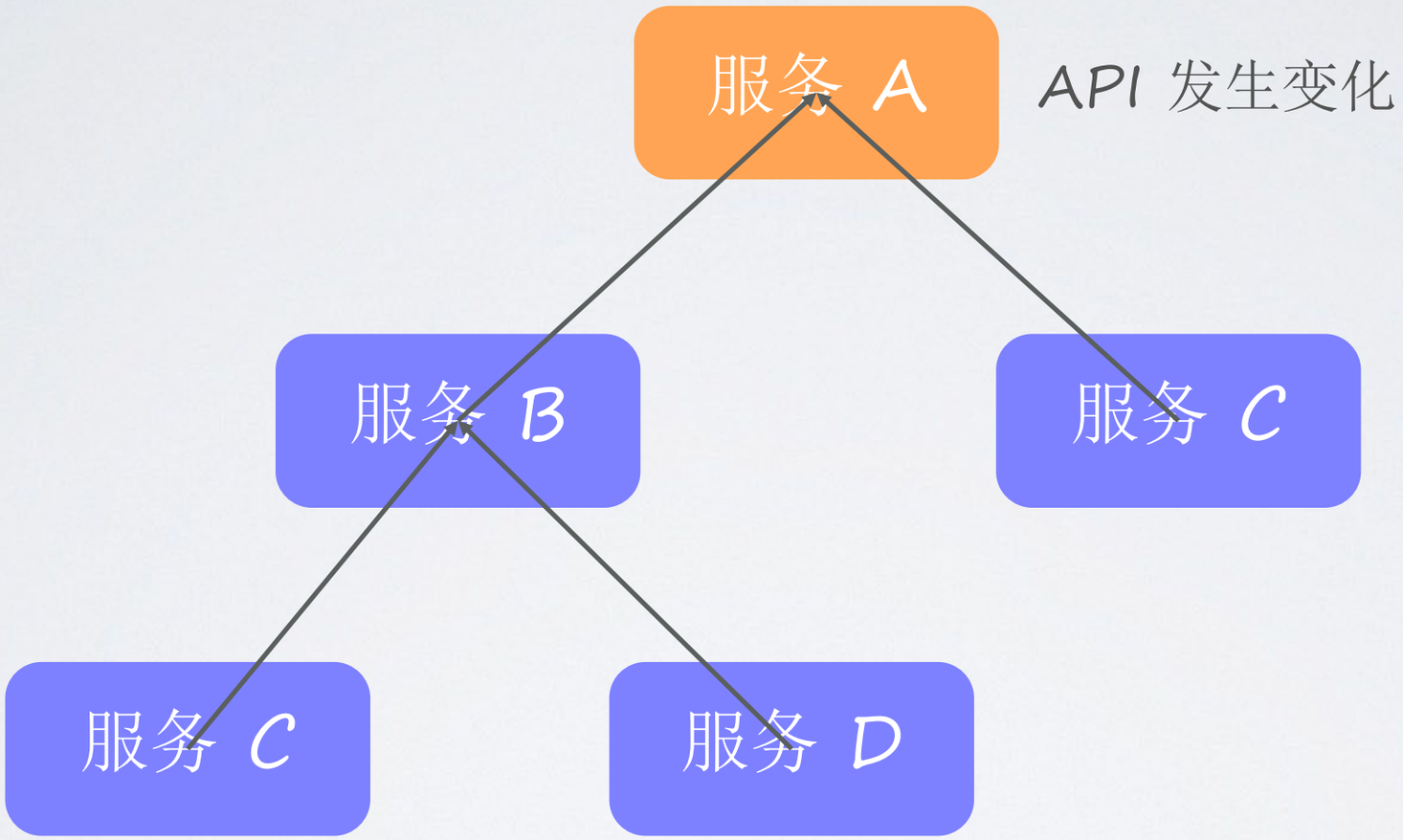
- 通过把 API 测试过程加入到 CI 流程，保证去掉 mock 后前端能无缝对接上后端



流程控制

- API 变更的递归通知
- 项目成员间的 Approve 和 Reject 机制

API 依赖树锁定



递归的检测 API 变化所带来的影响

提升项目间协作效率

- 共享 API 的定义
- 通过定义我们可以轻松生成文档
- 持续优化 API 相关的垂直搜索

应用场景

美团云

犀牛·云盘

美团外卖
美团外卖 送啥都快

猫眼电影

Vane 联调平台


Vane
搜索
项目列表
查找API
文档
lvwei04
登出
Vane
只需要用 markdown 定义一次 API 即可自动生成 API 文档， mock 假数据，单元测试 API，以及流程控制

ScreenFlow ファイル 編集 マーク 挿入 フォント アクション 表示 ウインドウ ヘルプ 100%

frontend — ys@ys-rmbp — -zsh — 122×31

..list/frontend ..list/frontend

$

举一个简单开发场景示例

一些常被问的问题

道理我都懂 那 API 如何定义呢？

adoc (api documentation)

```
1
2  ## @url /items/{id}{?limit}
3
4  我们可以写任意的 markdown 来说明这个 API
5
6  ## @case
7
8  ### @response
9
10 - @body Hello World!
```

markdown 超集

示例

路径 /items

我们可以写任意的 markdown 来说明这个 API

用例

返回

- Hello World!

渲染成文档

RPC Thrift

````
1
2  - @request
3
4    请求符合下面的定义：
5
6    ```thrift
7      service Calculator extends shared.SharedService {
8
9        void ping(),
10
11       i32 add(1:i32 num1, 2:i32 num2),
12
13       oneway void zip()
14
15     }
16   ```
17
````

利用 code block 的语言声明语法
如 RPC 几乎可以根据实际情况随意扩展

有状态的 API 请求如何 mock ？

- 每个 API 是有多个 case 定义的
- 可以选中任一的 case 组成一个 scenario

我们并没有止步于此

- API 线上监测
- 服务调用将逐步 RPC 化
- 从 API 调用的源头入手，我们将从 RPC 库开始尝试

通讯核心 nisper

- 一个非图灵完备的 lisp-0 语言
- 较 RPC 更为抽象灵活的一种思路
- 基于权限的语言，初始不含任何功能
- 项目地址：

问答

谢谢