腾讯在线教育大前端架构演进之路

曹海歌

腾讯 前端高级工程师

曹海歌

- 16年进入腾讯QQ 浏览器
- 17年加入在线教育部，负责企鹅辅导前端架构设计
- IMWeb 团队成员
- Keywords: 全栈 Nodejs、多端、性能优化

目录

1、腾讯在线教育部业务矩阵及技术图谱

腾讯在线教育部业务产品矩阵

产品覆盖了各年龄层

在 线 教 育 部

ABCMouse

企鹅辅导

腾讯课堂

腾讯在线教育大前端技术图谱

APP

小程序

h5、PC

服务端

存储

2019

腾讯在线教育大前端面临的挑战

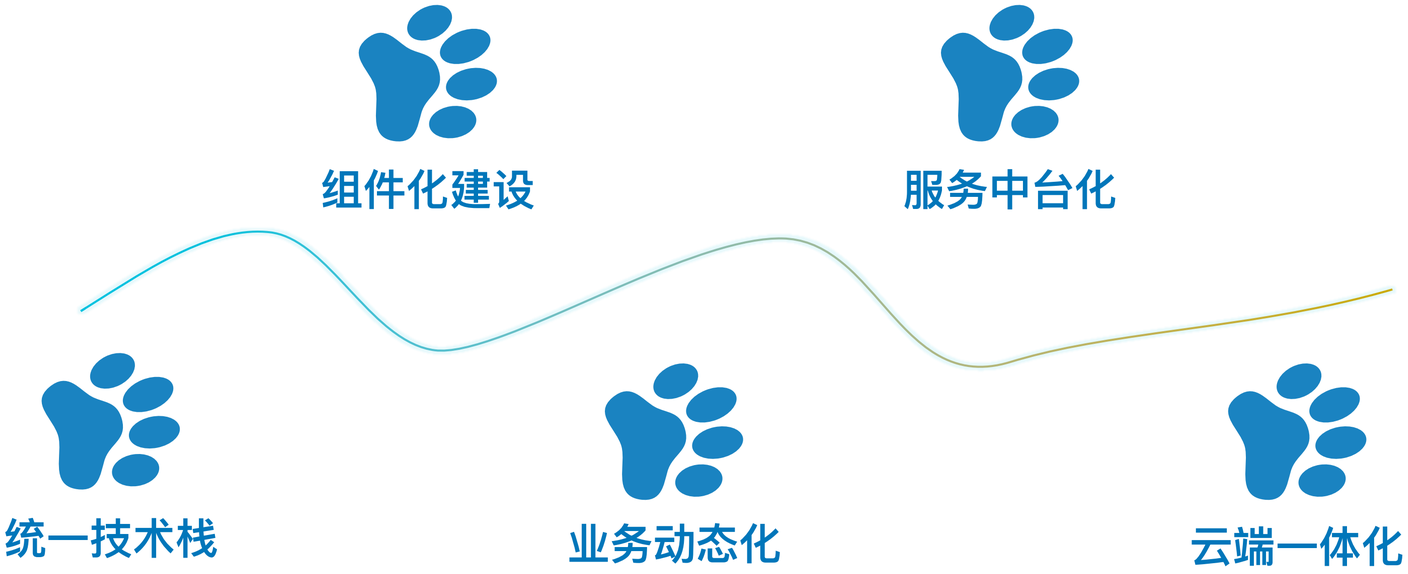
1、业务增长速度快、产品迭代周期短、项目质量要求高

2、大前端团队人数，前端加客户端，增长到了70多人

3、部门内三个业务并行发展，重复建设的能力越来越多

2、在线教育大前端的架构演进历程

腾讯在线教育大前端技术演进历程



架构之初的设计思考

如何避免功能
重复建设?

代码如何
更好的复用?

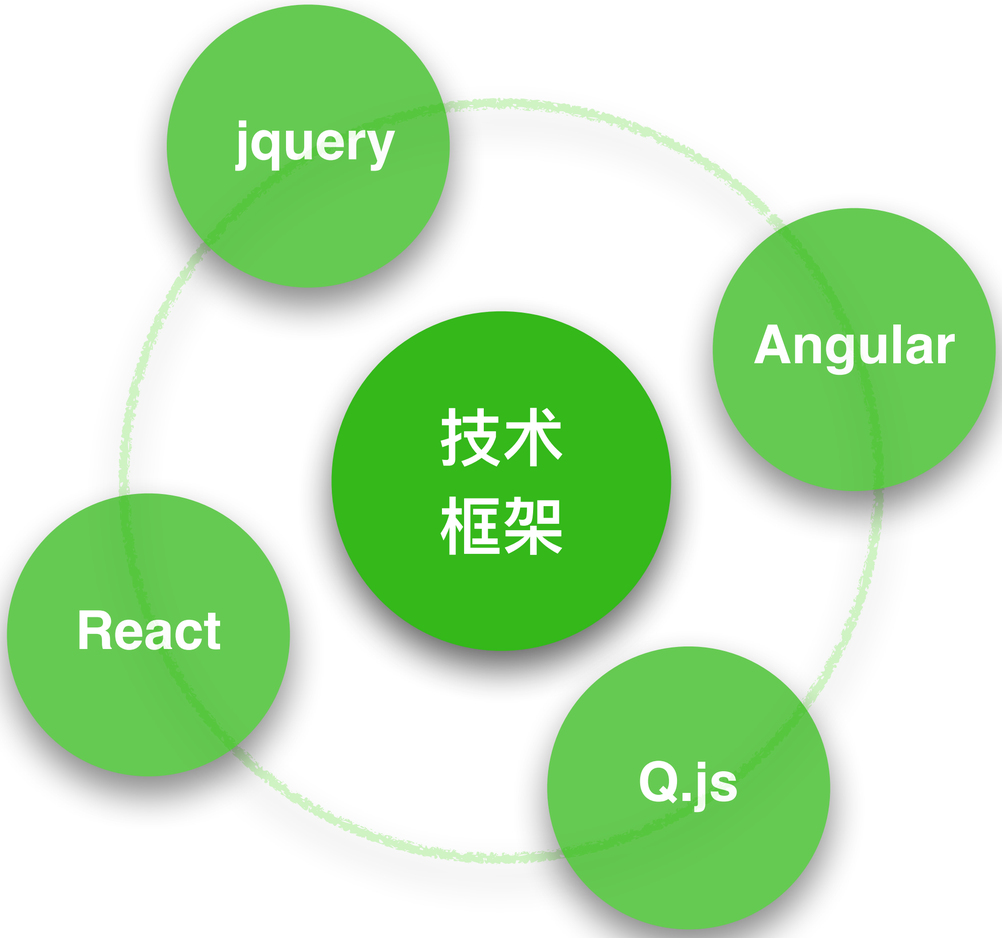
业务如何能
快速的上线?

怎么实现
可插拔设计?

答案：组件化

在线教育部成立之初技术现状

统一前端技术栈

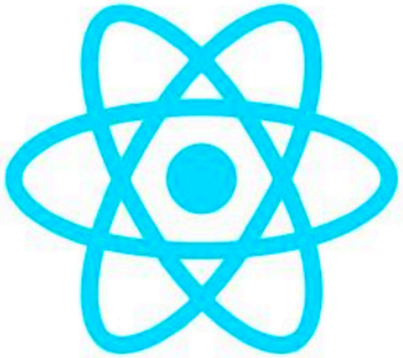


项目框架不同，构建也不同

存在自建模块的加载机制

前端统一技术栈

技术栈收归，是组件化落地的前提



React

Webpack

业务组件化落地

减少重复开发，一定程度提升了开发效率

随着业务发展，产品新的诉求

解决时效性问题

产品：首页给我上个活动推荐卡片

开发：可以，预期是下个月，跟 APP 版本提测

产品：我下周一，就要上线！

开发：做不到啊，苹果审核需要时间 ……

开发的思考

APP 动态化

1、能不能不发版本就解决问题?

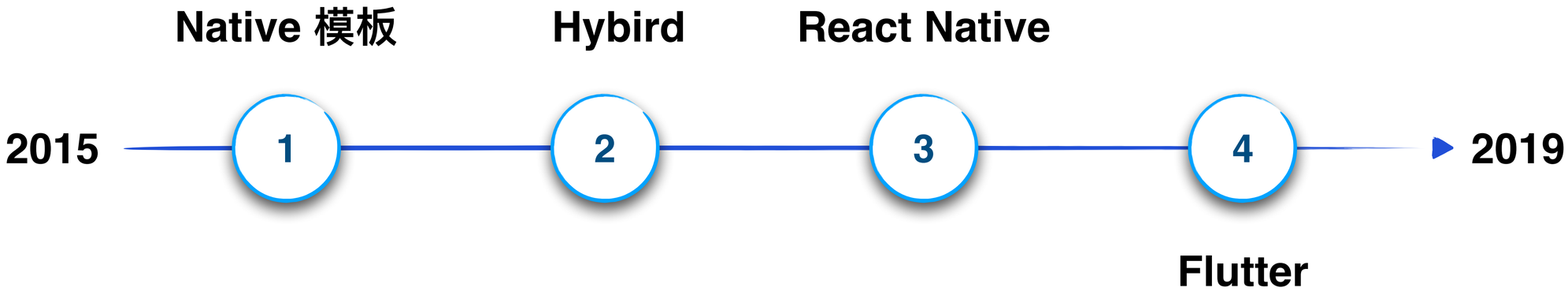
2、业务代码如何能实时热更新?

3、能不能不需要提前预埋逻辑?

4、安卓 和 IOS 能否一套代码?

2.1 在线教育大前端的服务中台化建设

在线教育大前端的动态化架构时间线



Native 模板

可以满足基本动态化诉求

模板

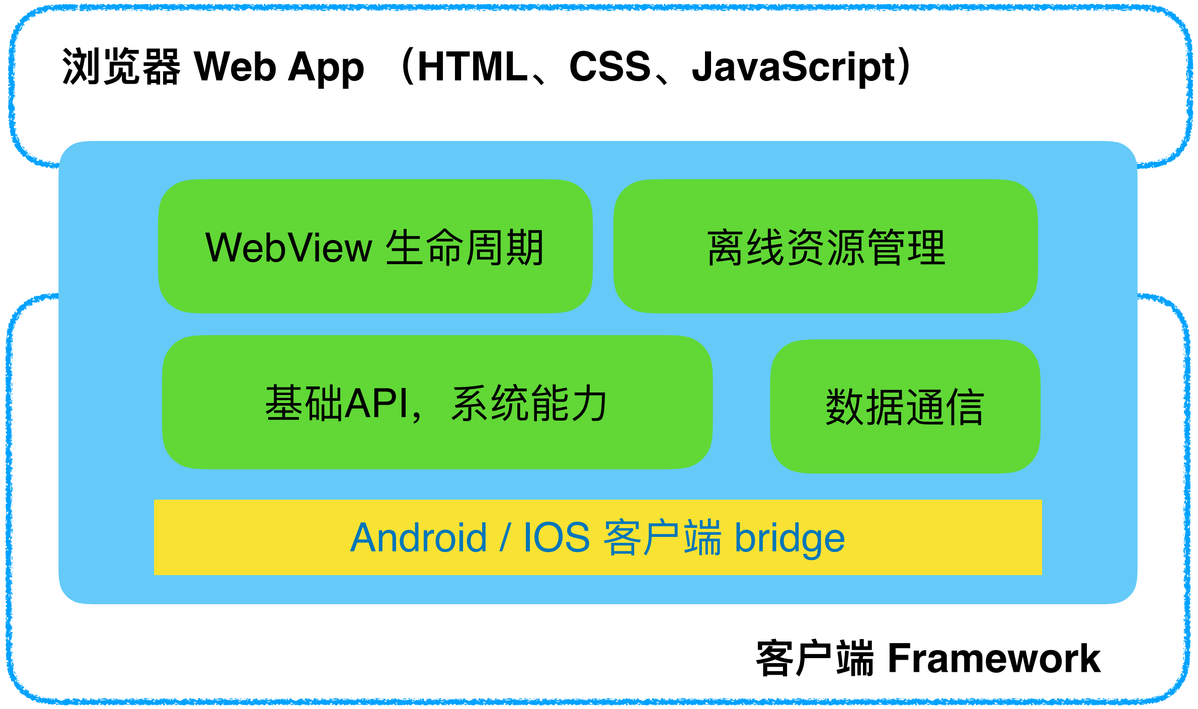
Native UI 绘制

- ✓ 优点：可以配置化更新
- ✗ 缺点：需要预埋，扩展性不够

遇见突发事件，无法解决根本问题

Hybird 离线包

满足业务动态化，提升页面加载速度

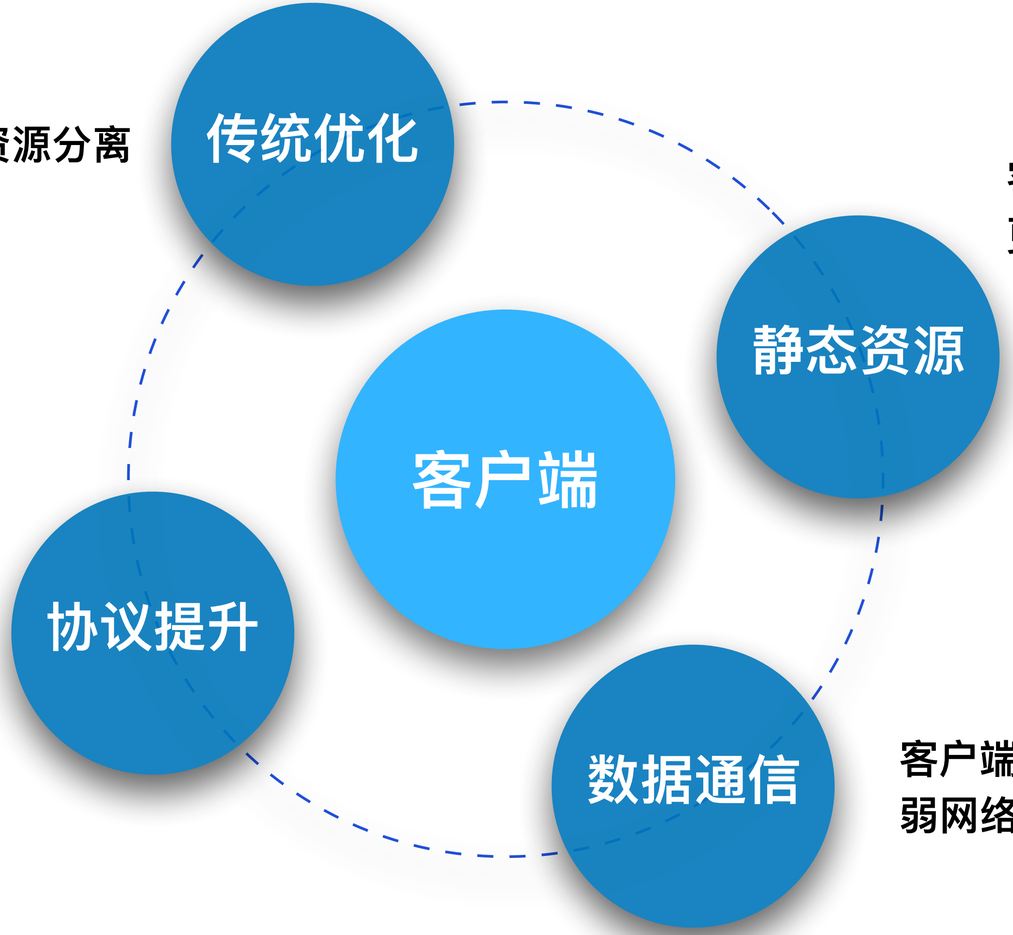


Web UI 绘制

- 优点：动态更新，开发和维护成本低
- 缺点：跟 Native 比，存在性能问题

Hybird 优化方案

资源压缩、Tree Shaking、资源分离



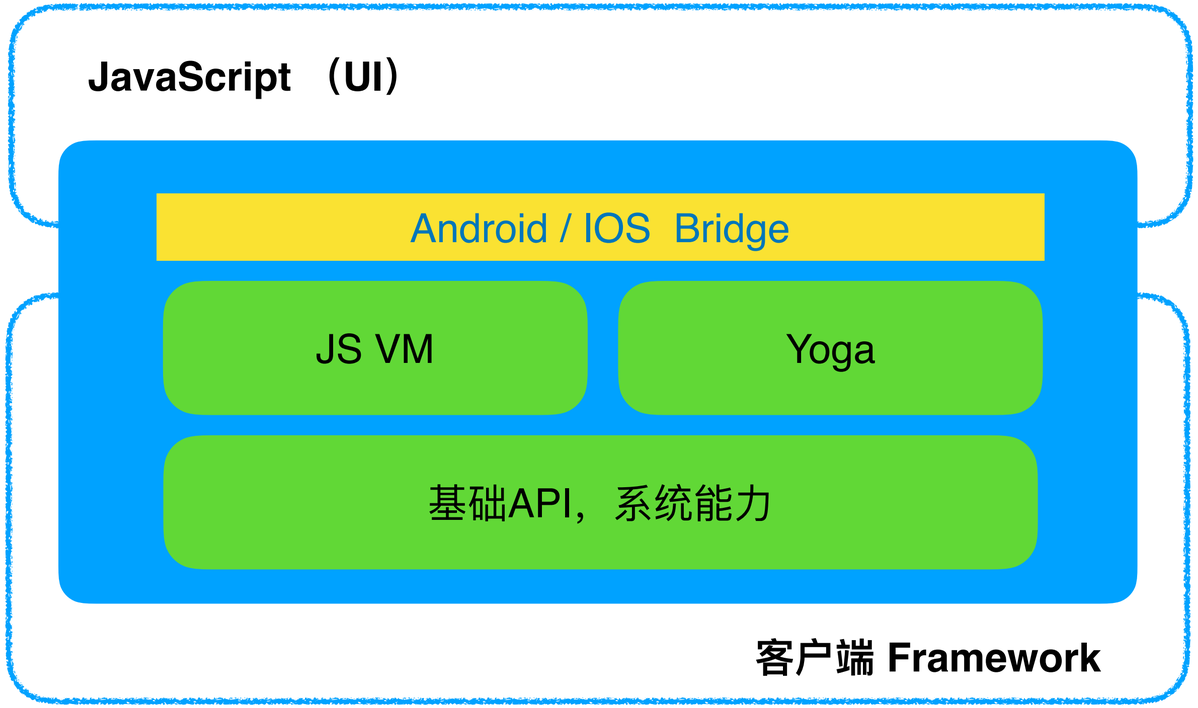
客户端离线包
更新策略，使用 diff 进行最小更新

传输协议：HTTPS 到 RPC
数据协议：JSON 到 PB、JCE

客户端预加载，减少网络耗时
弱网络情况，客户端协议重试

React Native

解决 APP 首屏问题



Native UI 绘制

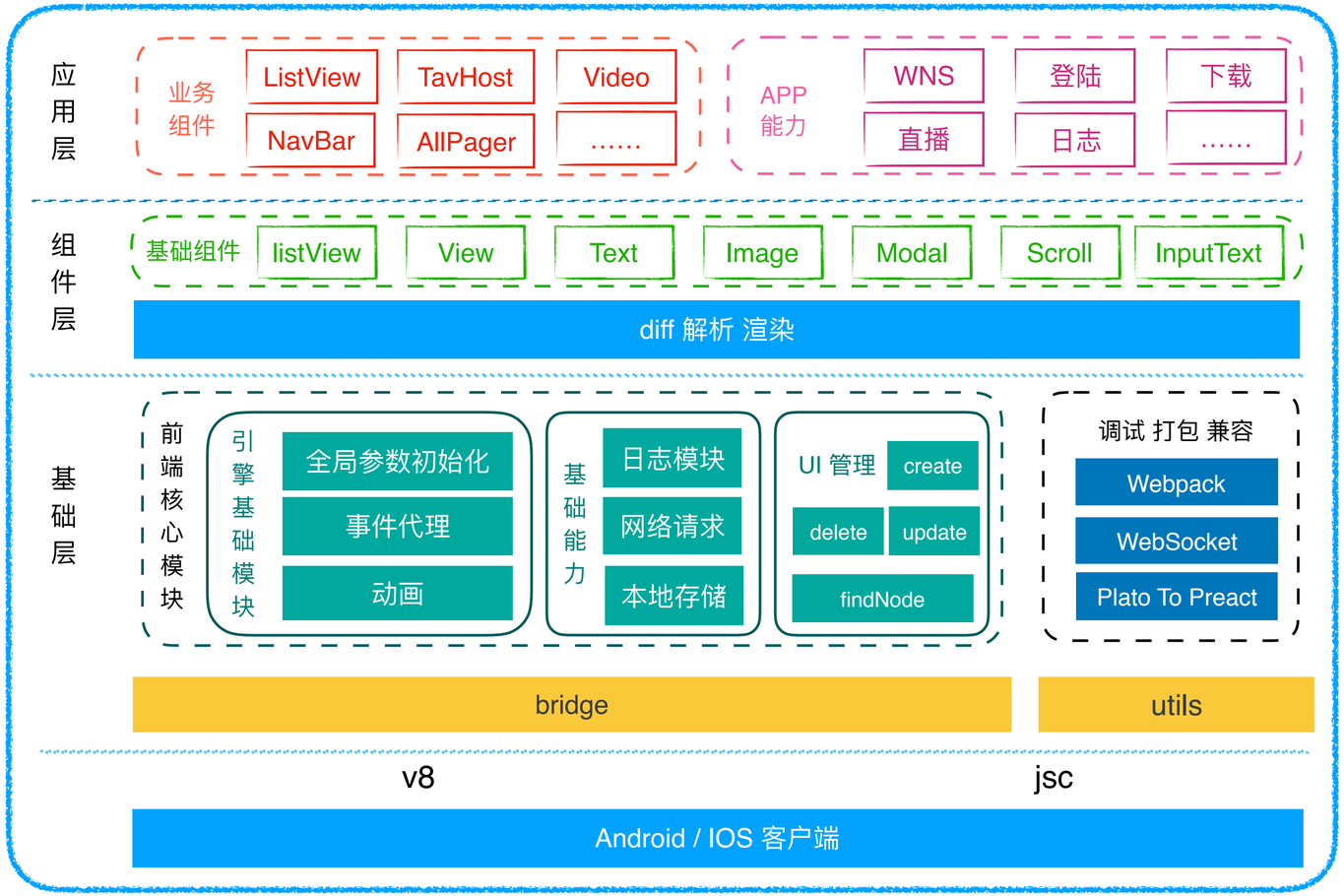
✔ 优点：保持 JS 开发，渲染性能提升

✘ 缺点：不同平台有差异，需要兼容处理
crash 率略高

17年出现了 license 问题

Plato 探索

自研 RN 框架



设计

分层更为合理 — 减少耦合

各层接口清晰 — 灵活性高

每层最小可用 — 渐进增强

优化

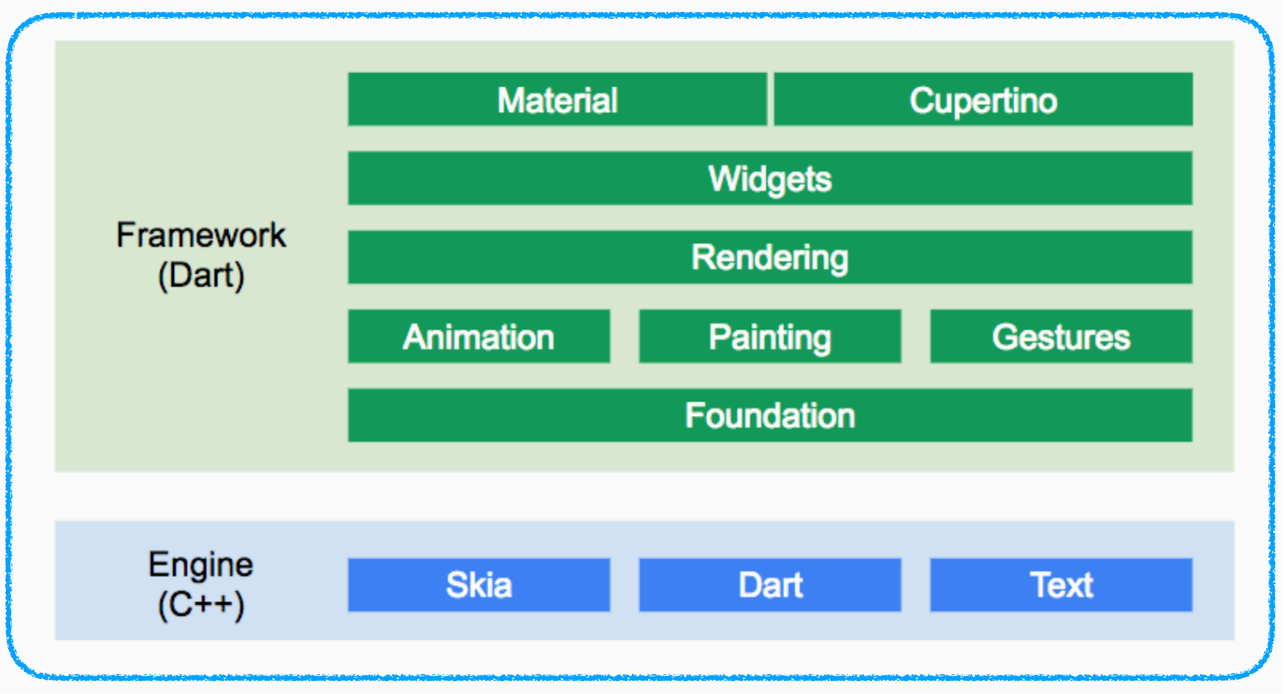
复杂动画，定制化后交给 Native 绘制

扩展性好，支持 vue、preact

裁剪事件监听，只抛出绑定的事件

Flutter

降低 Crash 率，提升 UI 一致性



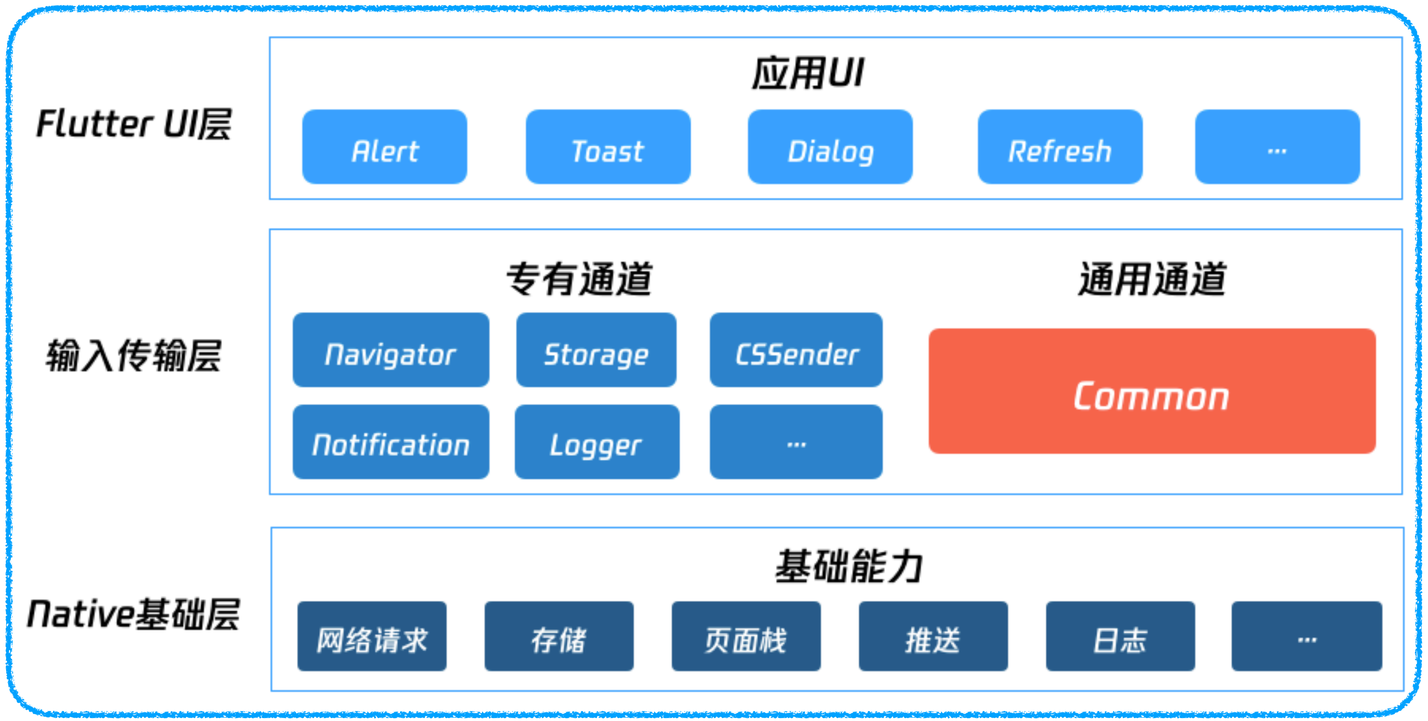
跨平台引擎 UI 绘制

✔ 优点：性能媲美 Native，且多端渲染一致

✘ 缺点：不支持动态化，目前生态不如 RN
前端编写 dart，脱离前端 JS 生态

MJFlutter 实践

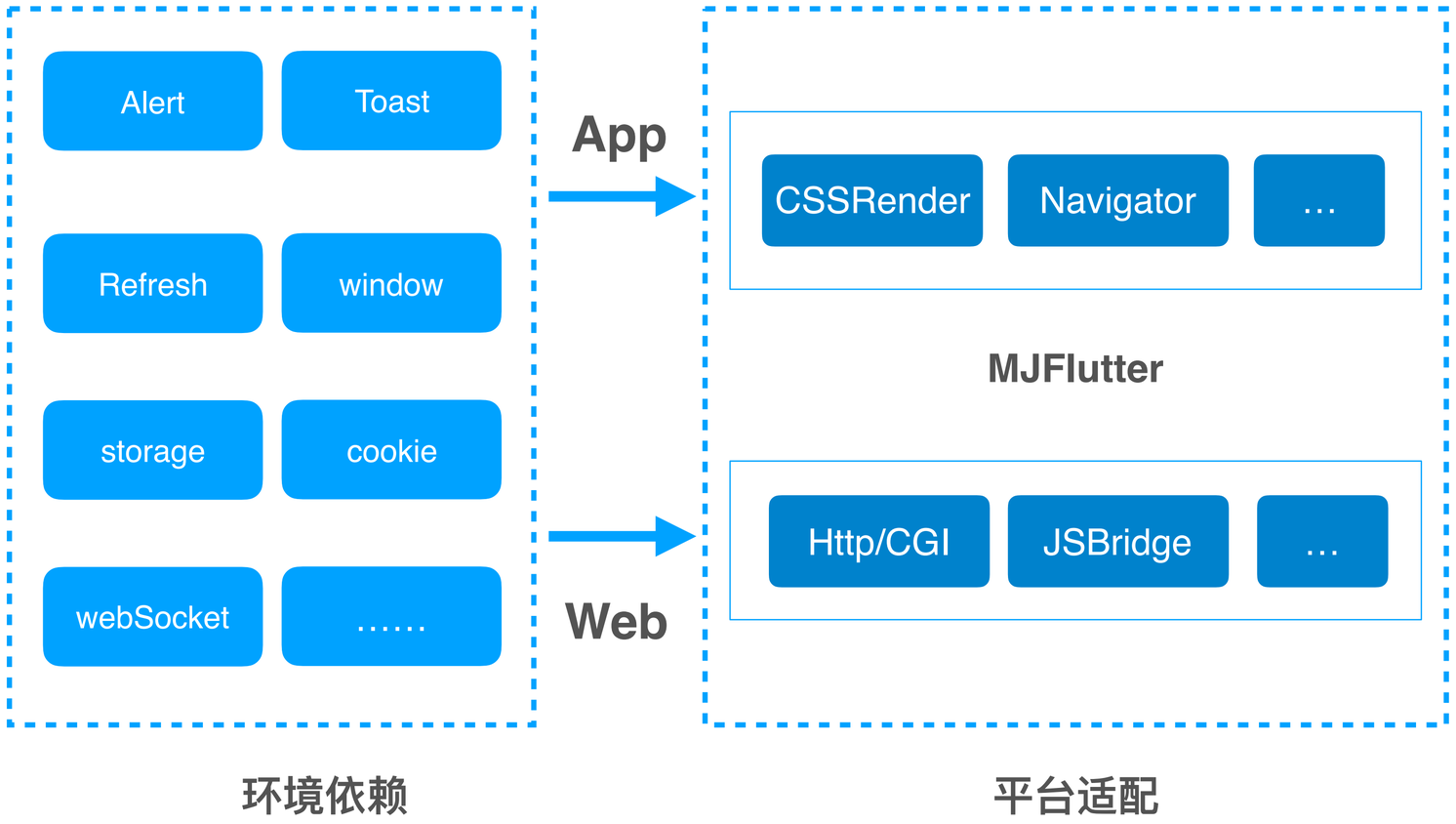
企鹅辅导 落地 Flutter



Flutter 官方不支持动态化

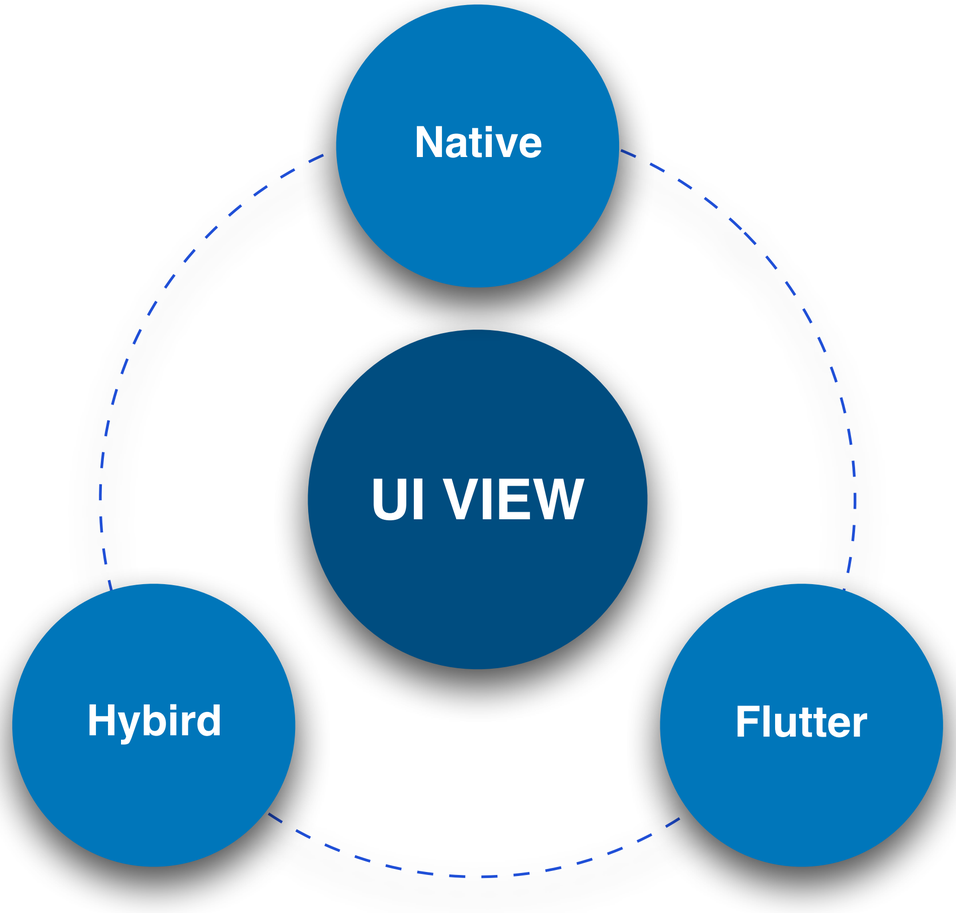
Flutter to Web 探索

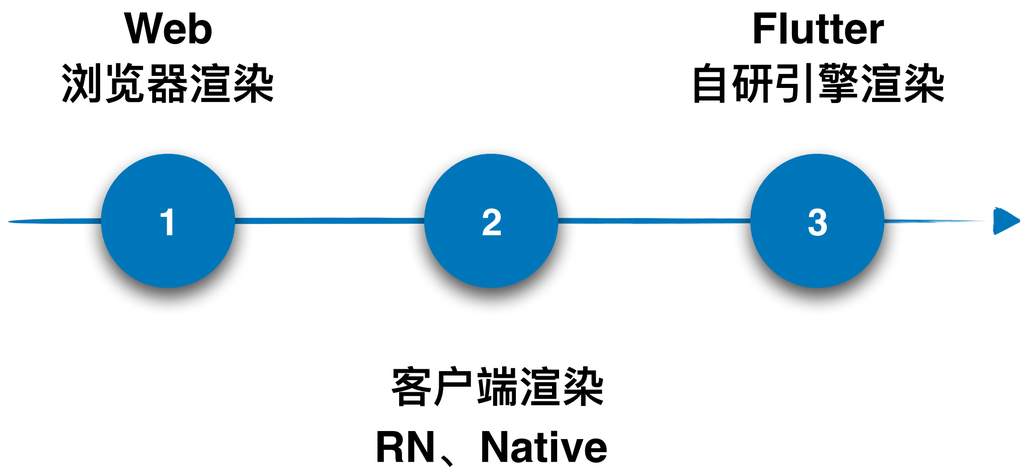
不适合追求性能的业务，适合非 C 端场景



腾讯在线教育动态化演进回顾

本质是渲染方式的变革





业务快速发展，如何提升产品质量

核心页面秒开

产品：核心页面打开很慢啊?

开发：是吗？我看一下上报的页面数据。

产品：是的，尤其是端外，很慢！

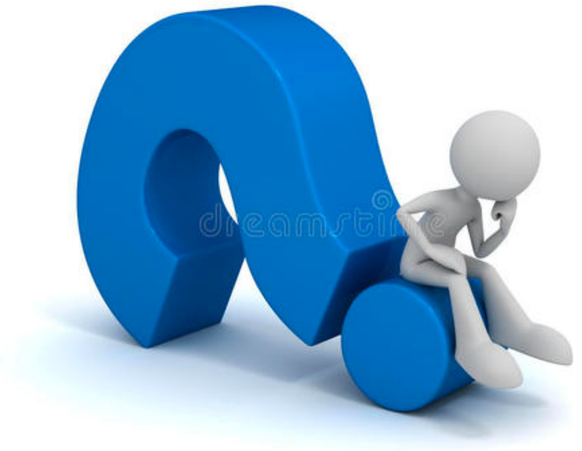
开发：嗯，我们优化一下。

开发的思考

Node 服务端直出

1、在非离线包的场景下，怎样提升页面打开速度?

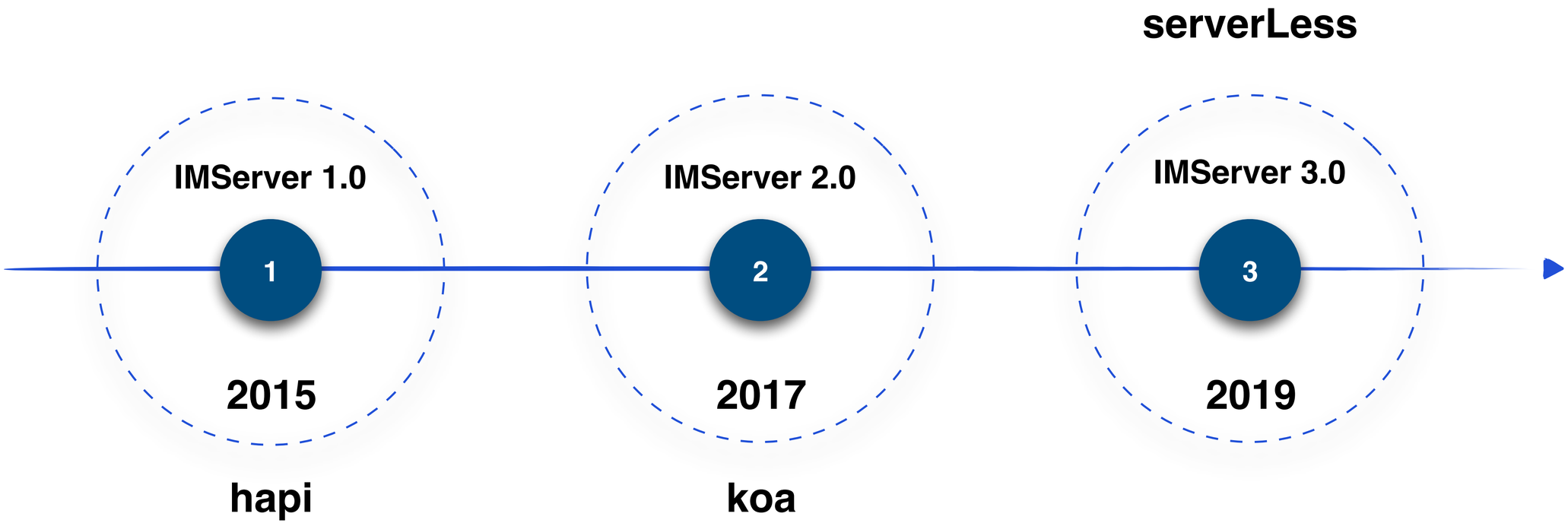
2、在 APP 内有离线包的情况下，为什么页面打开会慢?



基于 Nodejs 衍生的服务端体系建设

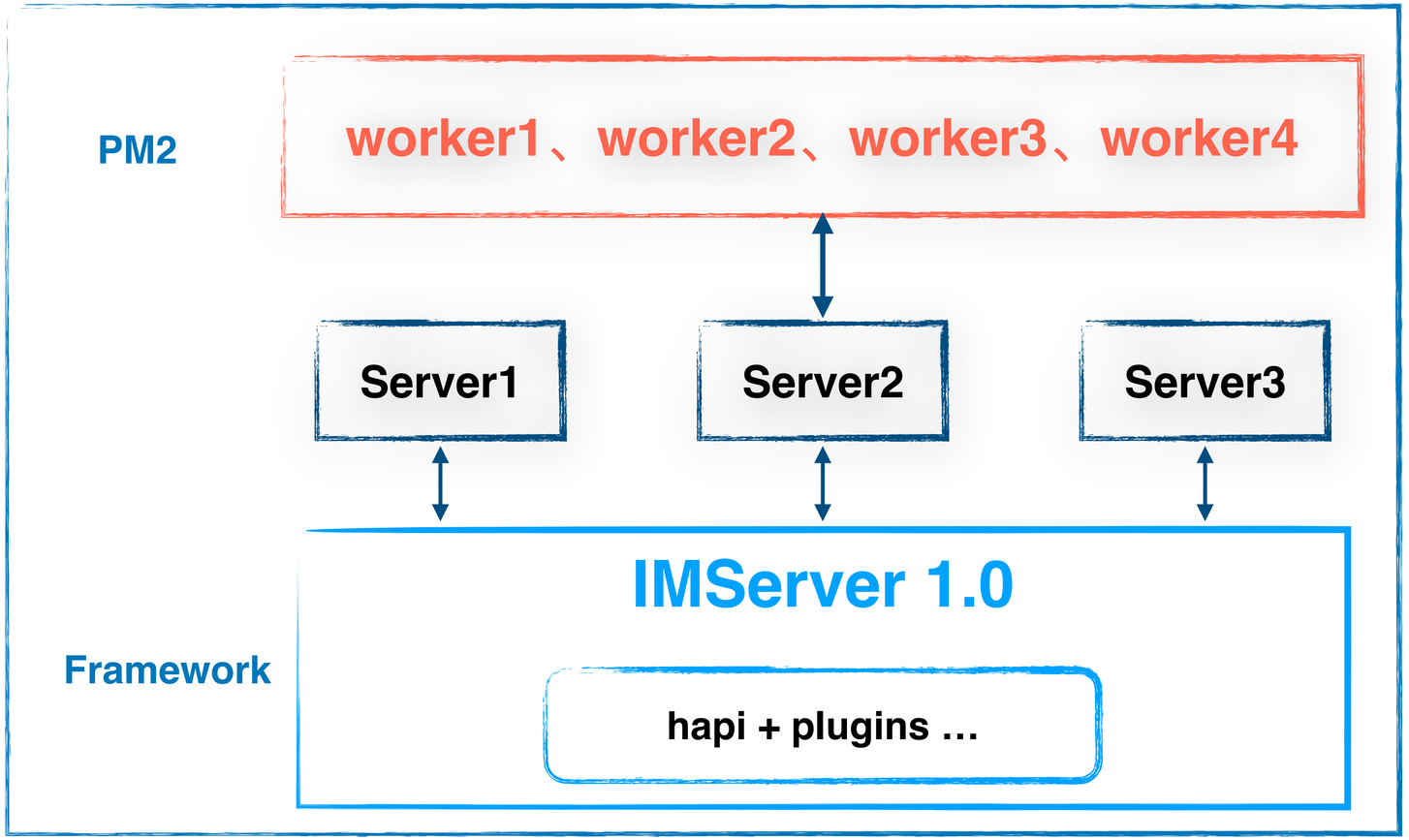
2.2 在线教育大前端的服务中台化建设

在线教育团队 Node 服务的基石 IMServer



IMServer 1.0 建设

拓展了前端的边界



业务目标

替代 Java VM 方案

优化团队研发模式，提升开发效率

IMServer 1.0 问题

IMServer 架构升级

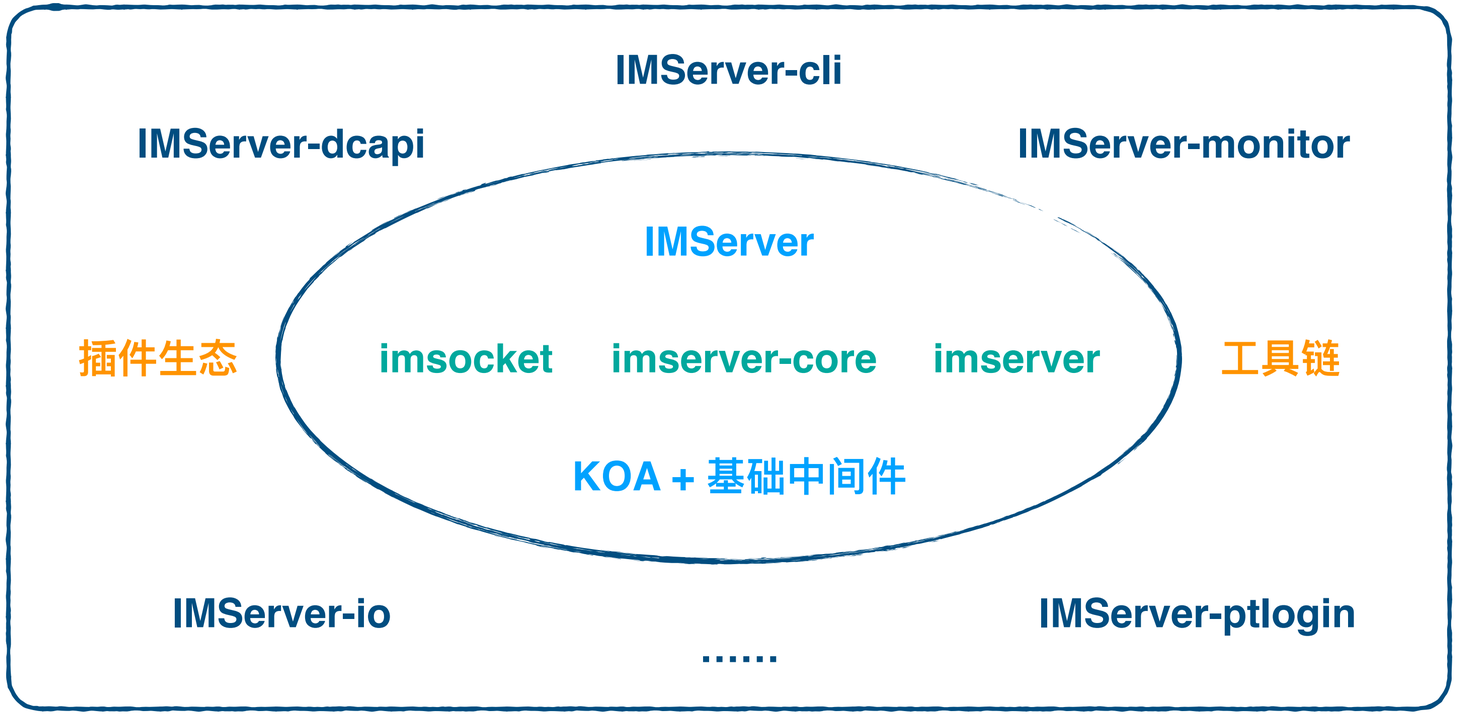
共用一个 Framework 耦合严重

一个业务出了问题，影响其它业务

缺少合理的团队代码开发规范

IMServer 2.0 升级改造

约定大于规范



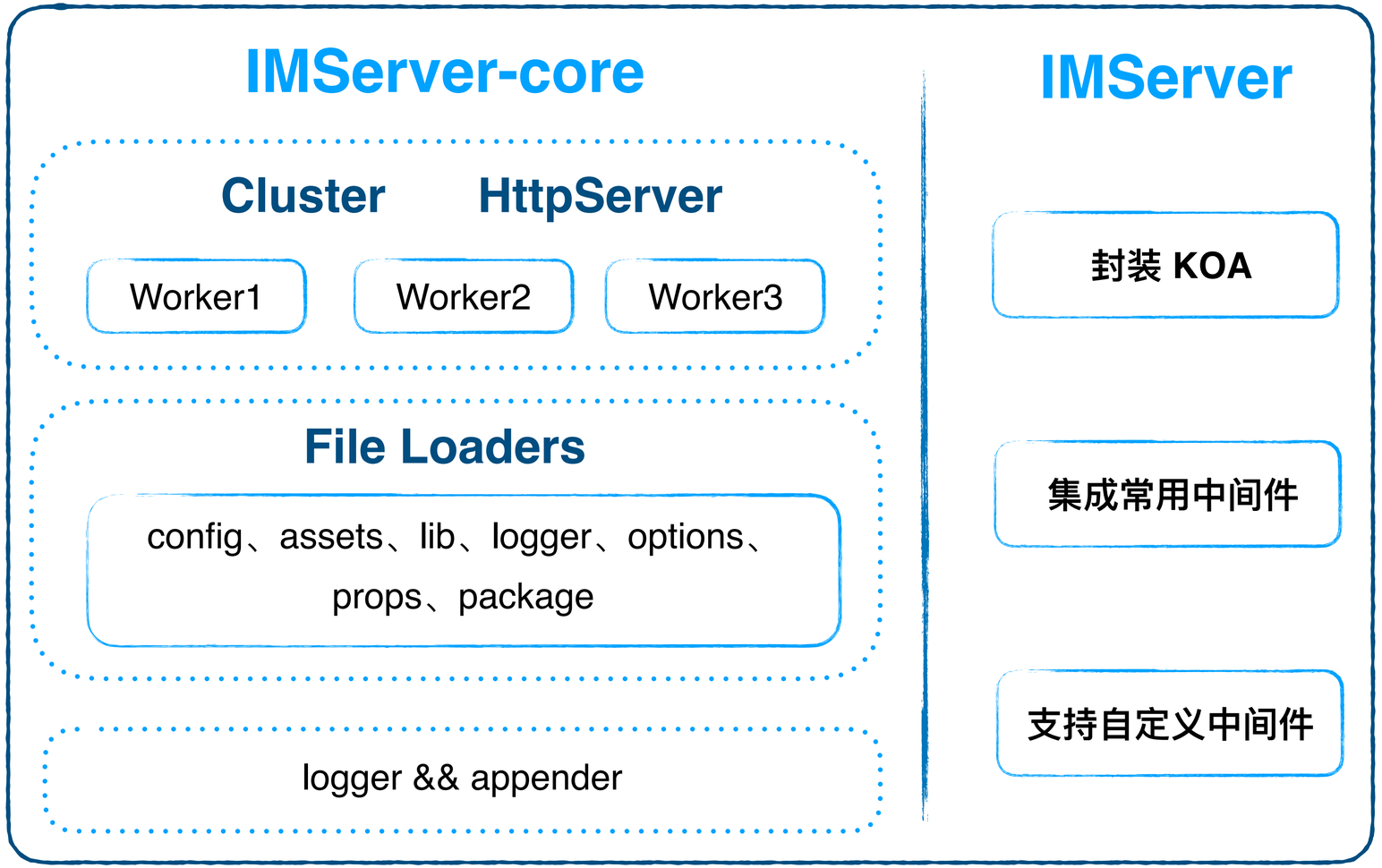
设计思想

灵活可扩展的微内核架构

业务采用插件的形式嵌入

IMServer 2.0 架构图

组合大于继承



功能模块

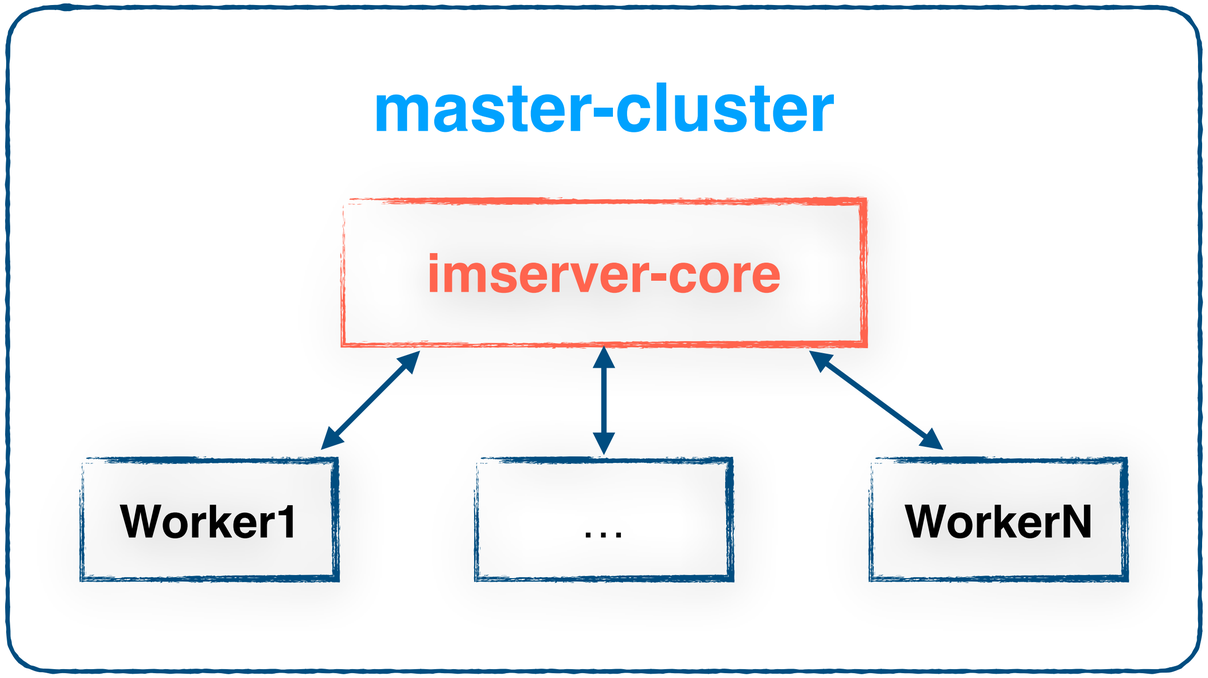
进程管理：PM2 —> IMServer

原子框架：hapi —> Koa2

加载策略：目录规范、模块引用

IMServer 2.0 进程管理

业务隔离部署



进程管理

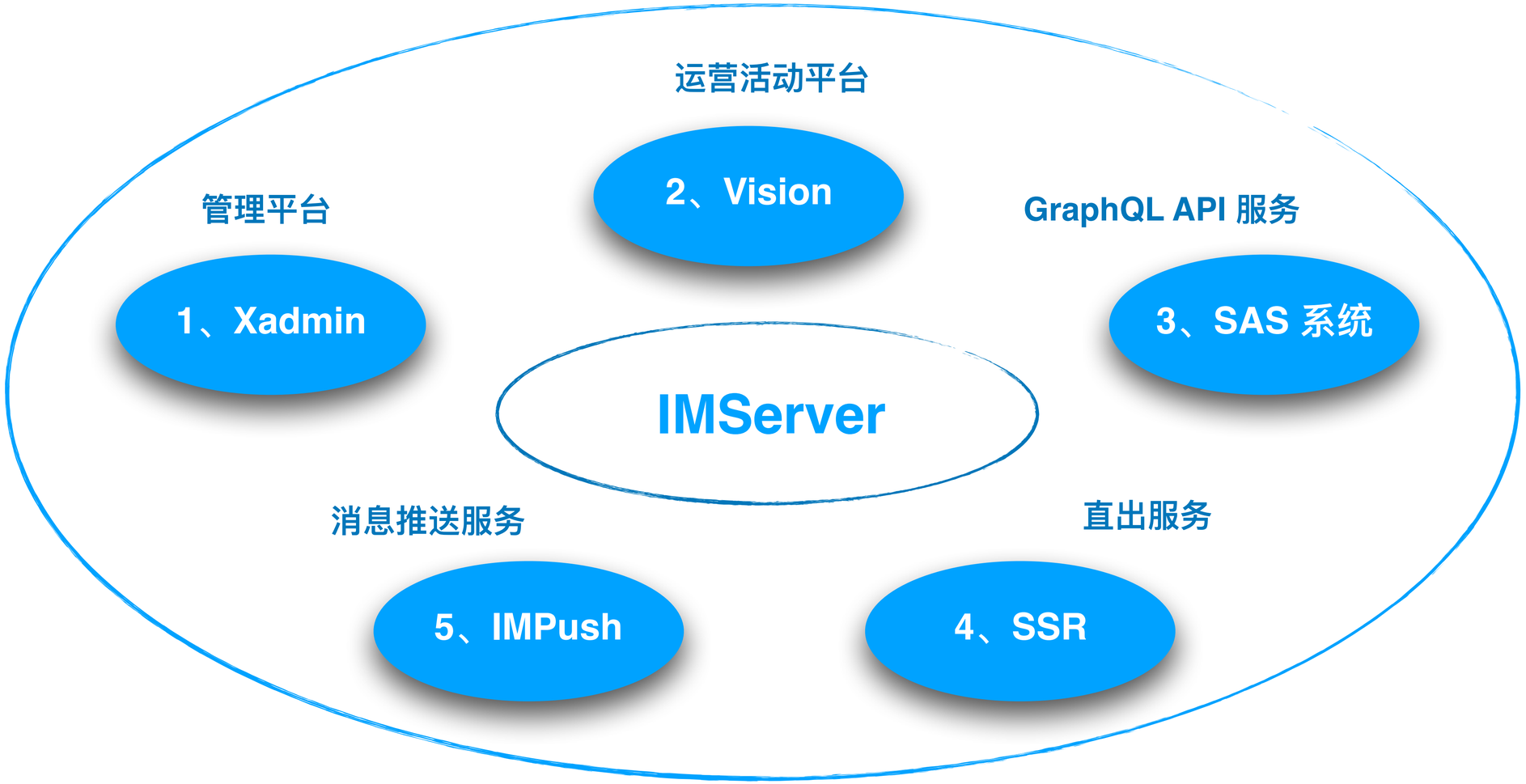
摒弃了全局共用的 Framework

imserver-core 定位是守护进程

处理进程的心跳检测、保活、重启、异常捕获

IMServer 2.0 落地业务系统

业务服务化中台



前端开发者面临的困境

serverless 恰逢其时的出现了

流量预估　　域名申请　　操作系统知识

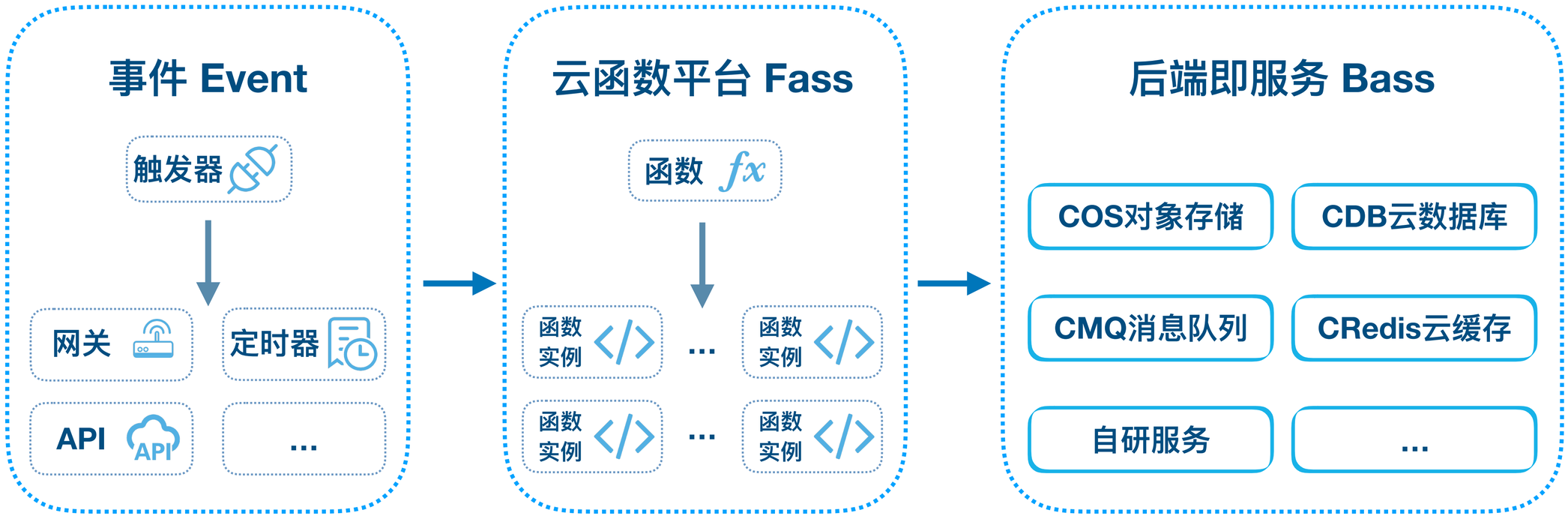
资源申请　　资源利用率　　告警和监控

备份容灾　　内存泄露问题　　机器扩容　　环境搭建

服务上云

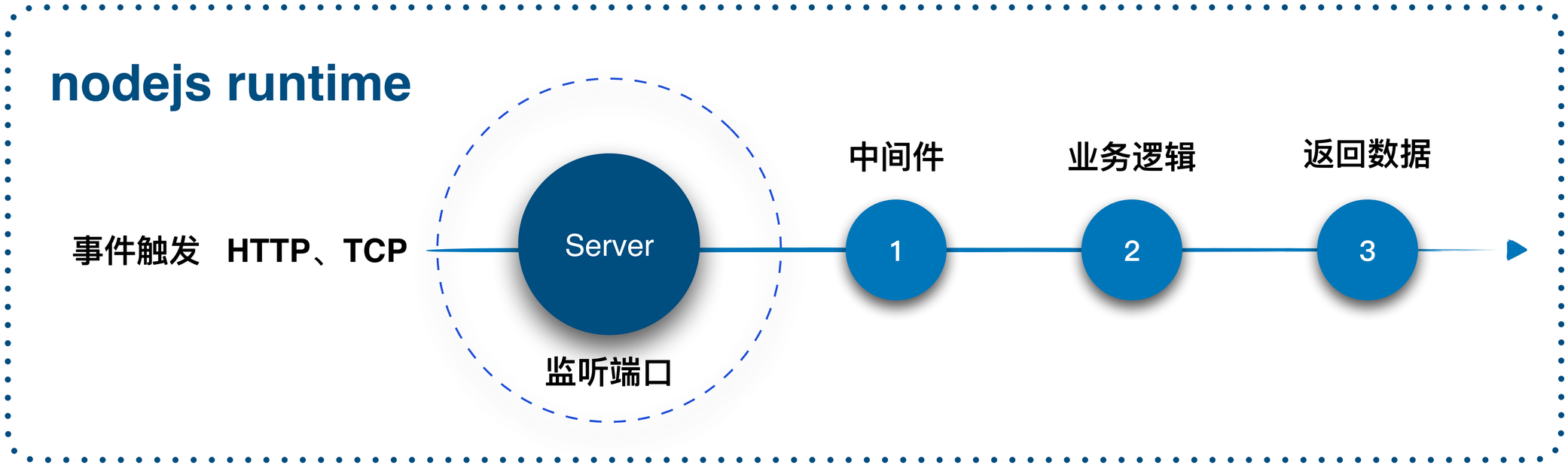
Serverless 是什么?

让前端远离运维，无服务的设计



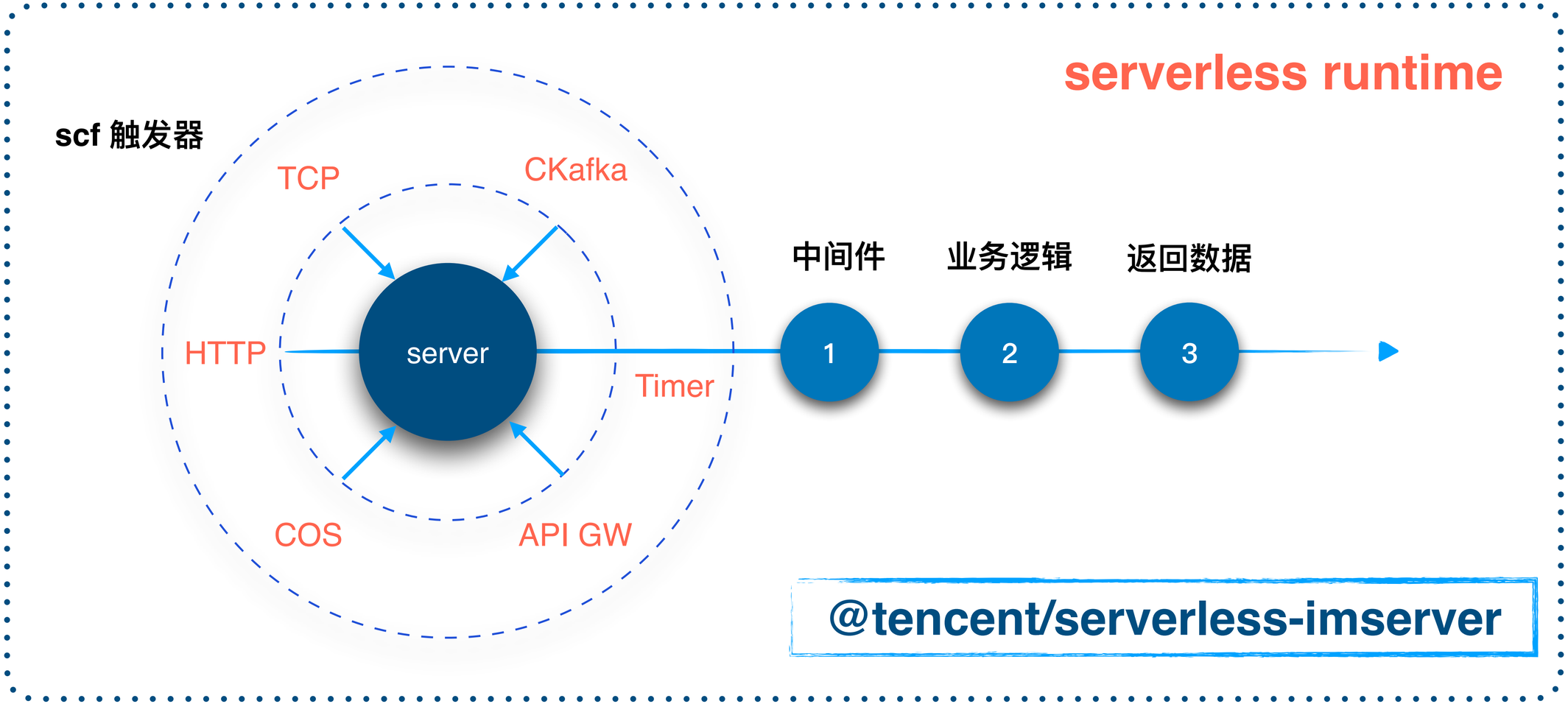
事件驱动　无状态　弹性扩容　无运维

传统 Node 服务事件处理

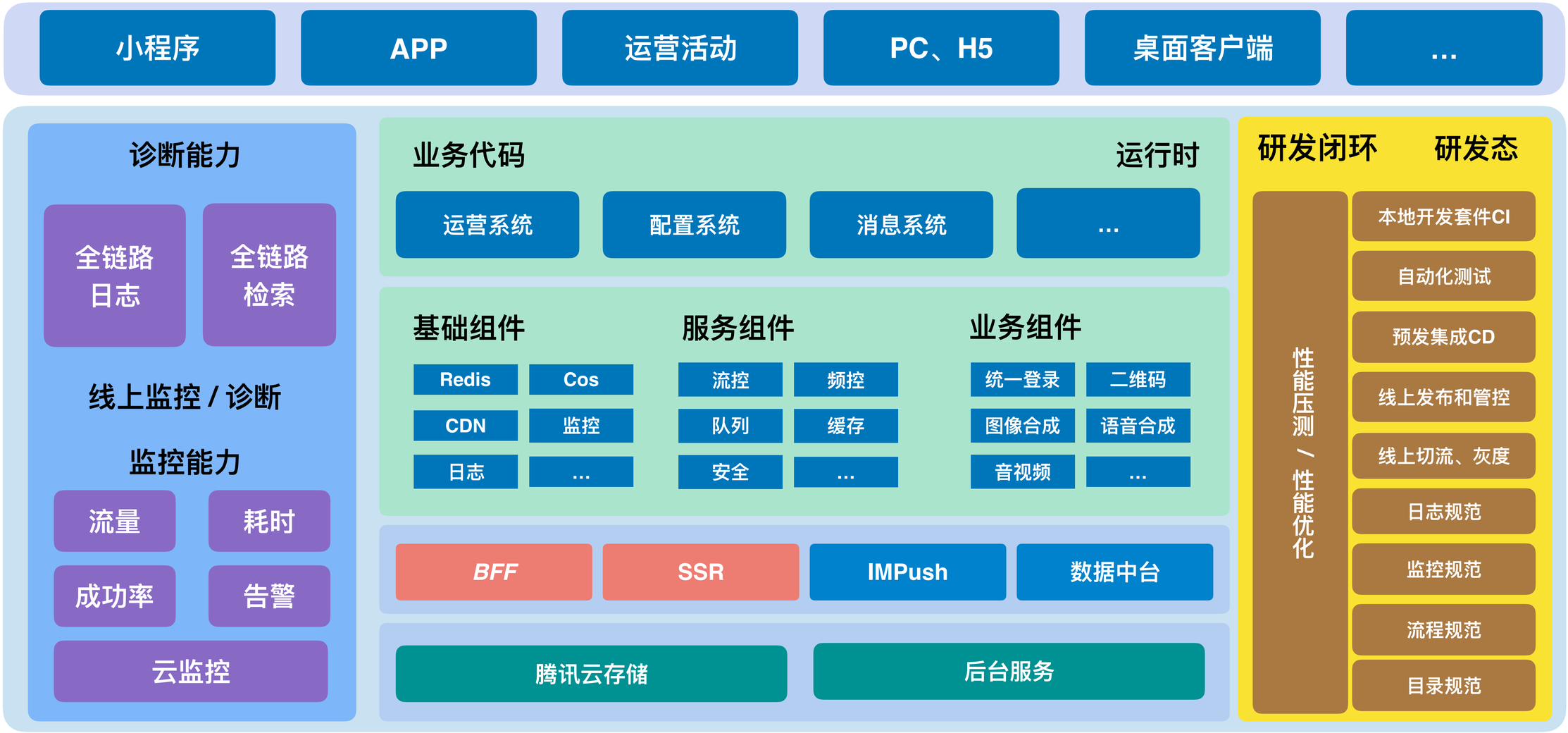


IMServer3.0 如何适配 Serverless

保证老服务的顺利迁移

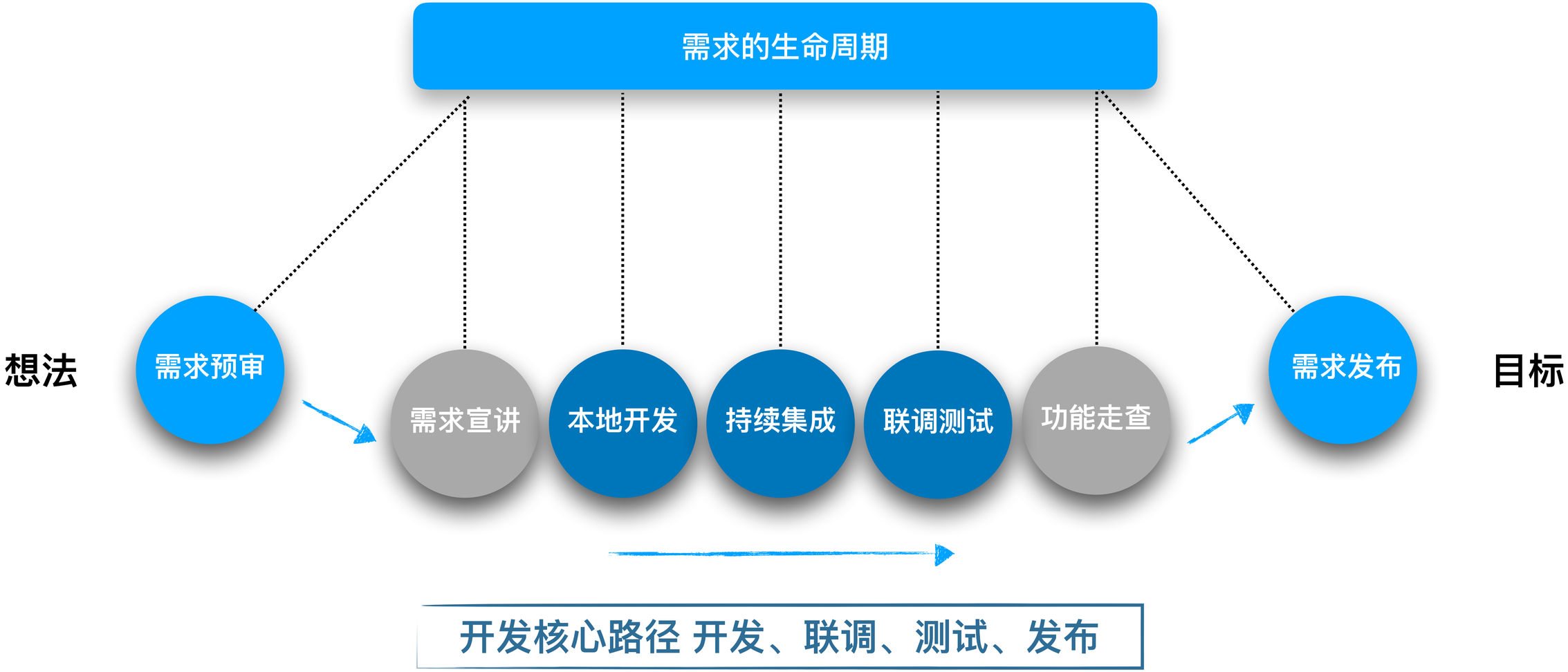


在线教育大前端服务端架构全景图



2.3 在线教育大前端研发效率工具演进

一个功能从想法到落地的过程



CLI 开发工具

开发者核心诉求

1、解决繁琐的 WebPack 配置问题，并且要保持工具扩展性，满足各类前端场景。

2、通用能力沉淀（主域重试、页面数据上报、单元测试），完成代码闭环。

3、统计代码规范（目录、组件、esLint、commit），降低合作成本。

4、统一前端工作流（创建、开发、发布），标准化研发模式。

减少重复轮子的建设，就能提升轮子的质量

CLI 工具演示

解决前端开发的效率，让开发聚焦于业务


```
Options:
  -V, --version          output the version number
  --nochecklatest        不检测最新版本
  --debug                输出构建调试信息
  -h, --help             output usage information

Commands:
  create [dir] [type]    初始化项目
  page|p                 新建页面
  component|c            新建组件
  sb-init                初始化 storybook 配置
  sb [options]           开启storybook测试组件
  build [options]        构建生产包
  dev [options]          启动开发者模式
  test [options]         运行 jest 测试
  clean                  清理缓存文件和构建结果文件
  check                  检测代码是否合并主干
  init [type]            给项目添加额外能力
```

友好的交互命令行

前端开发效率 ≠ 需求开发效率

前端只是软件开发体系的一部分

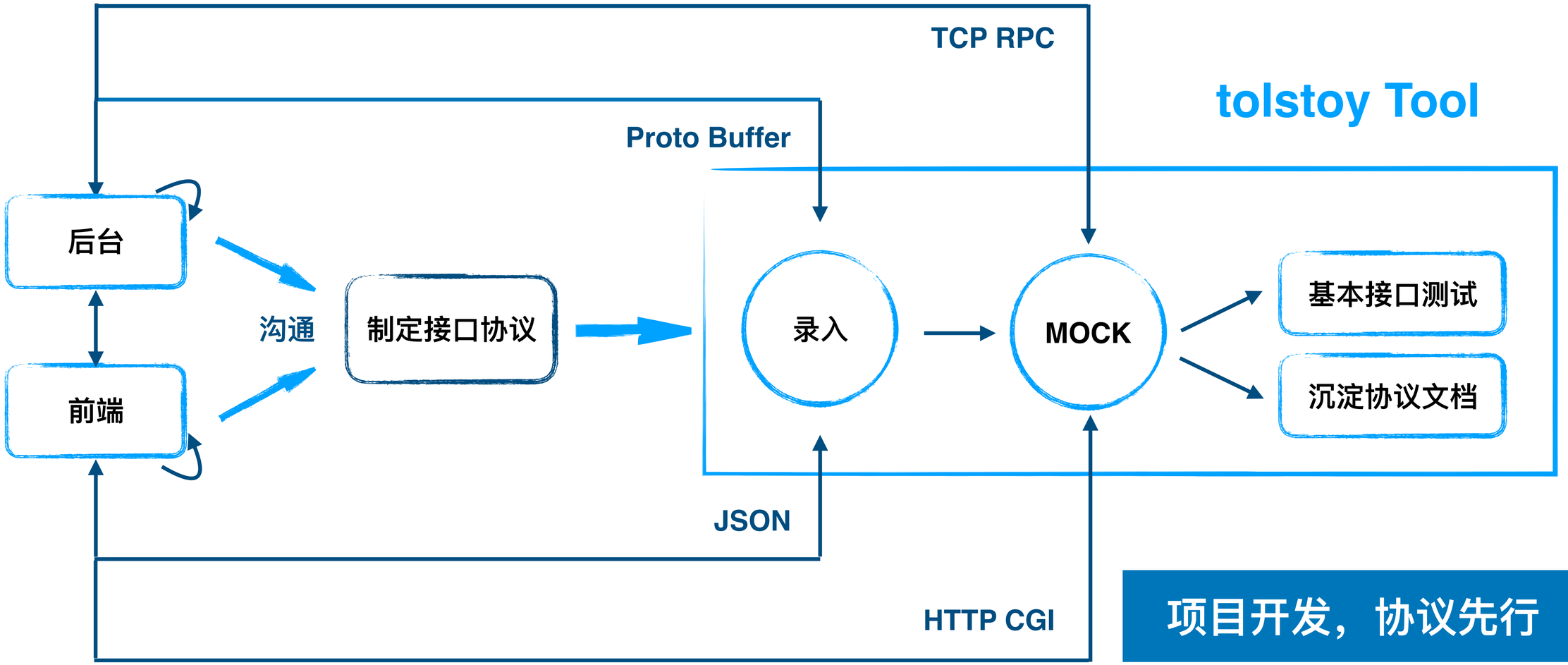
接口文档更新不及时，协议对不上

业务祖传接口，靠读代码理解含义

面向协议开发，提升联调效率

业务联调工具架构

核心能力

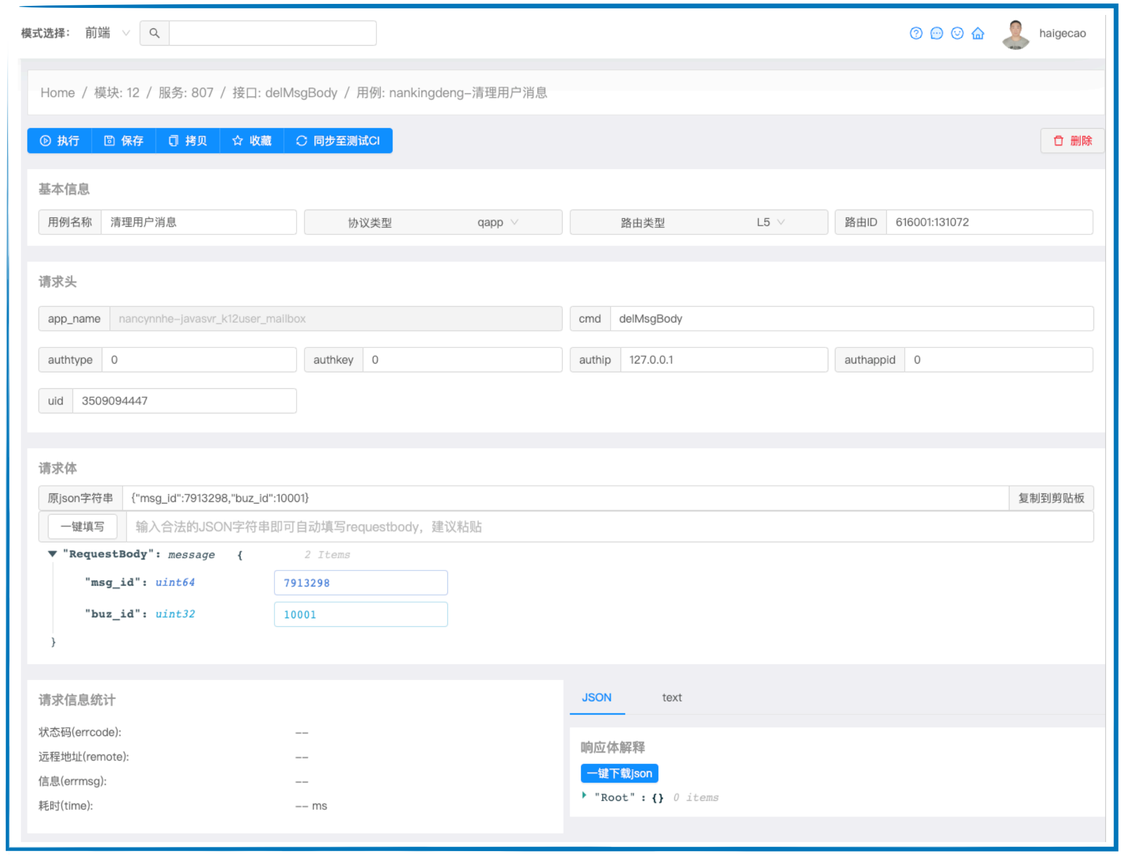


项目开发，协议先行

tolstoy 平台展示

优化团队联调效率

协议制定 以及 Mock



后台接口测试

优化调试效率 ≠ 省略业务调试

调试是软件开发的必经之路

多个需求同时联调，环境相互覆盖

没有稳定环境，页面报错，接口不稳定

测试违反单一原则，逻辑相互影响

代码夹带调试，可能导致发布问题

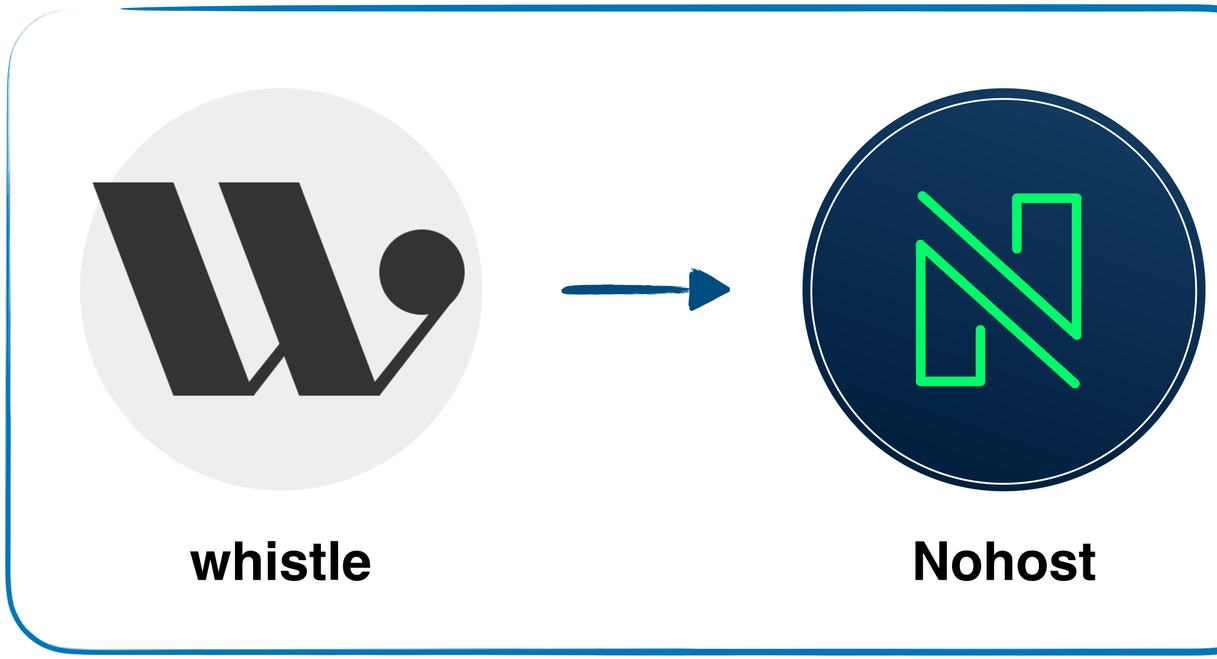
解决环境
冲突问题

布置多套环境成本过高

解决环境冲突 & 保证环境稳定

自研环境隔离工具

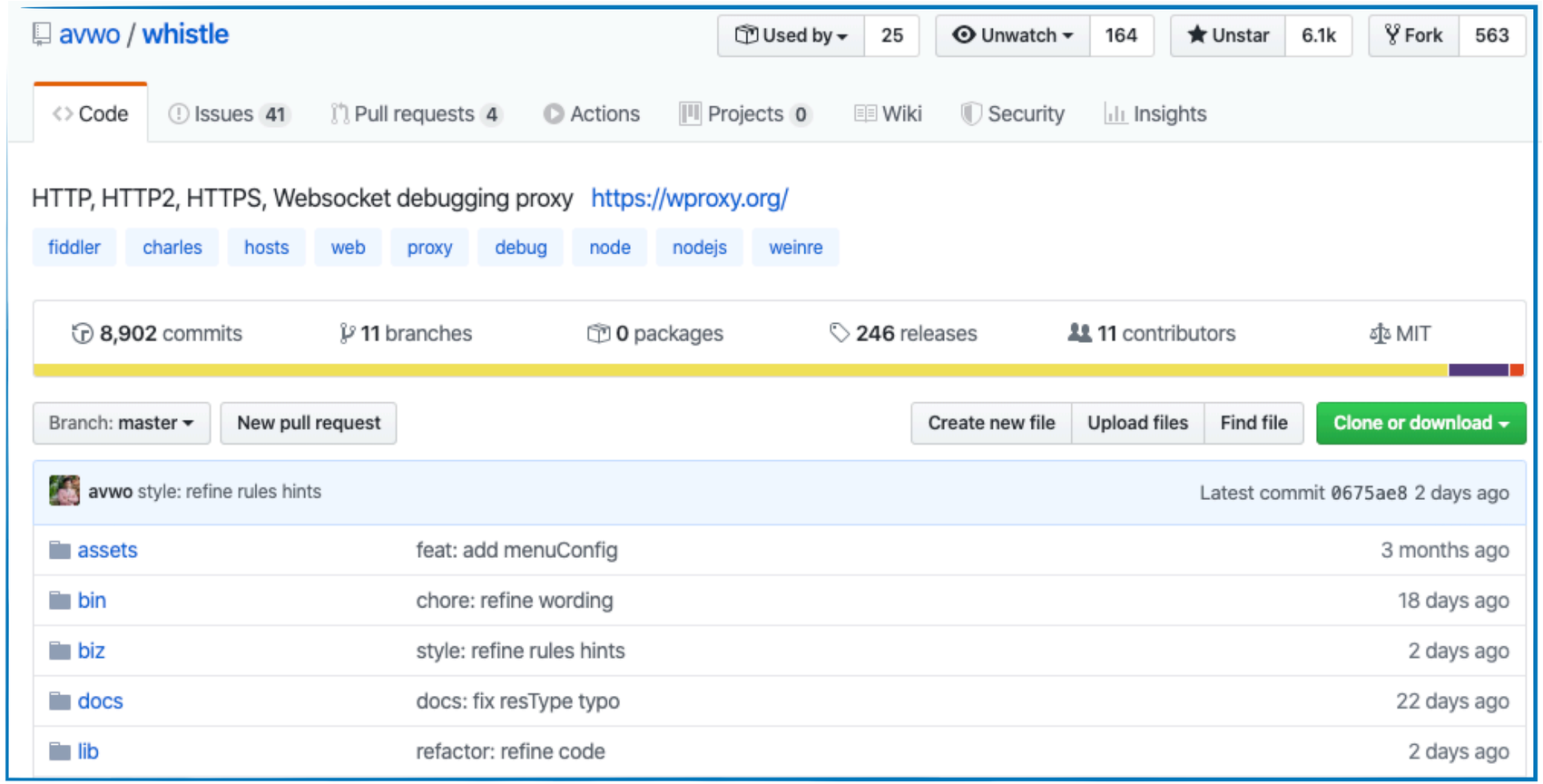
加机器维护成本过高



whistle 是什么?

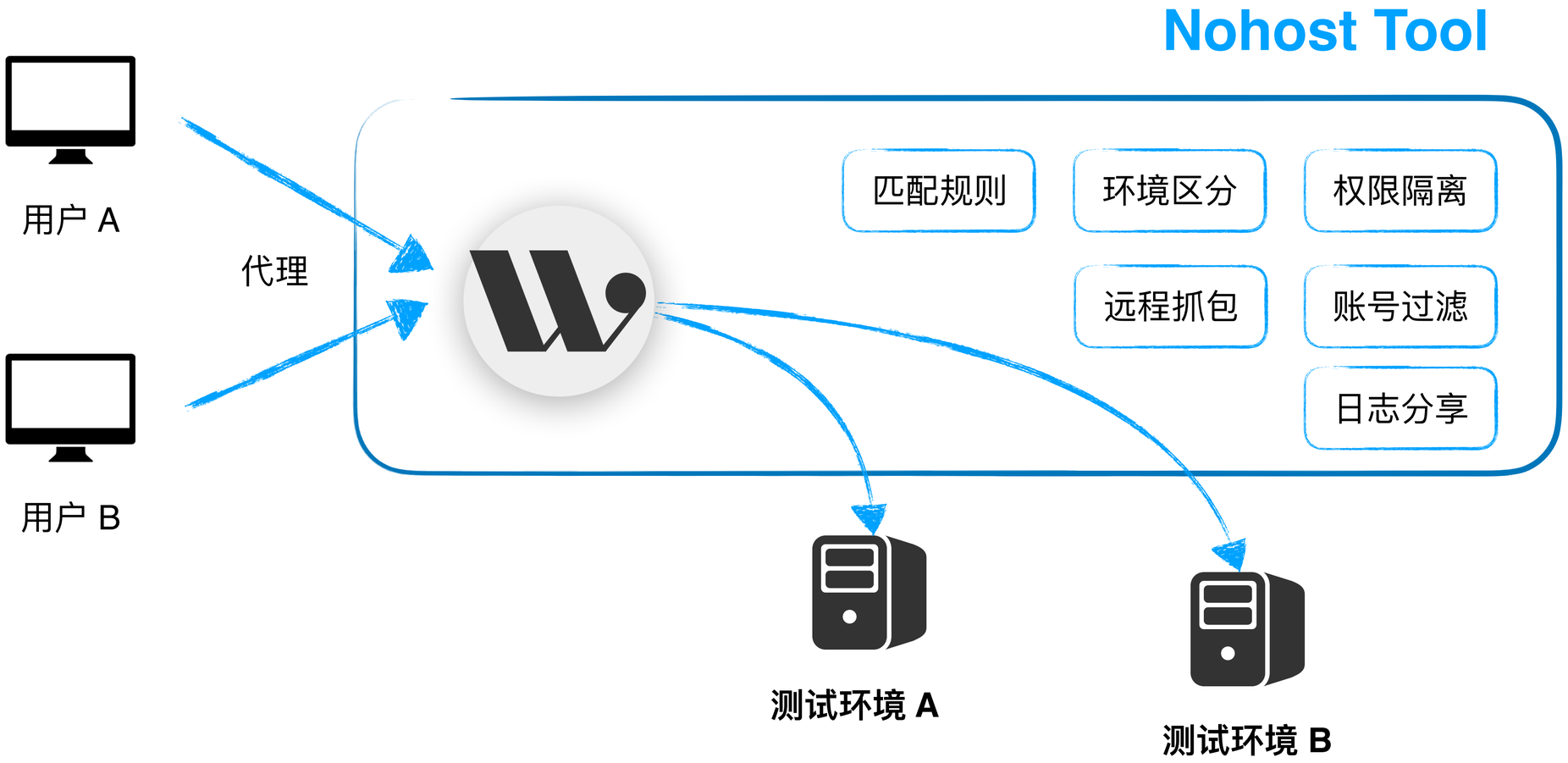
跨平台、可定制、免费 fiddler

IMWeb aven



Nohost 是什么?

可定制的云端 fiddler



Nohost 工具展示

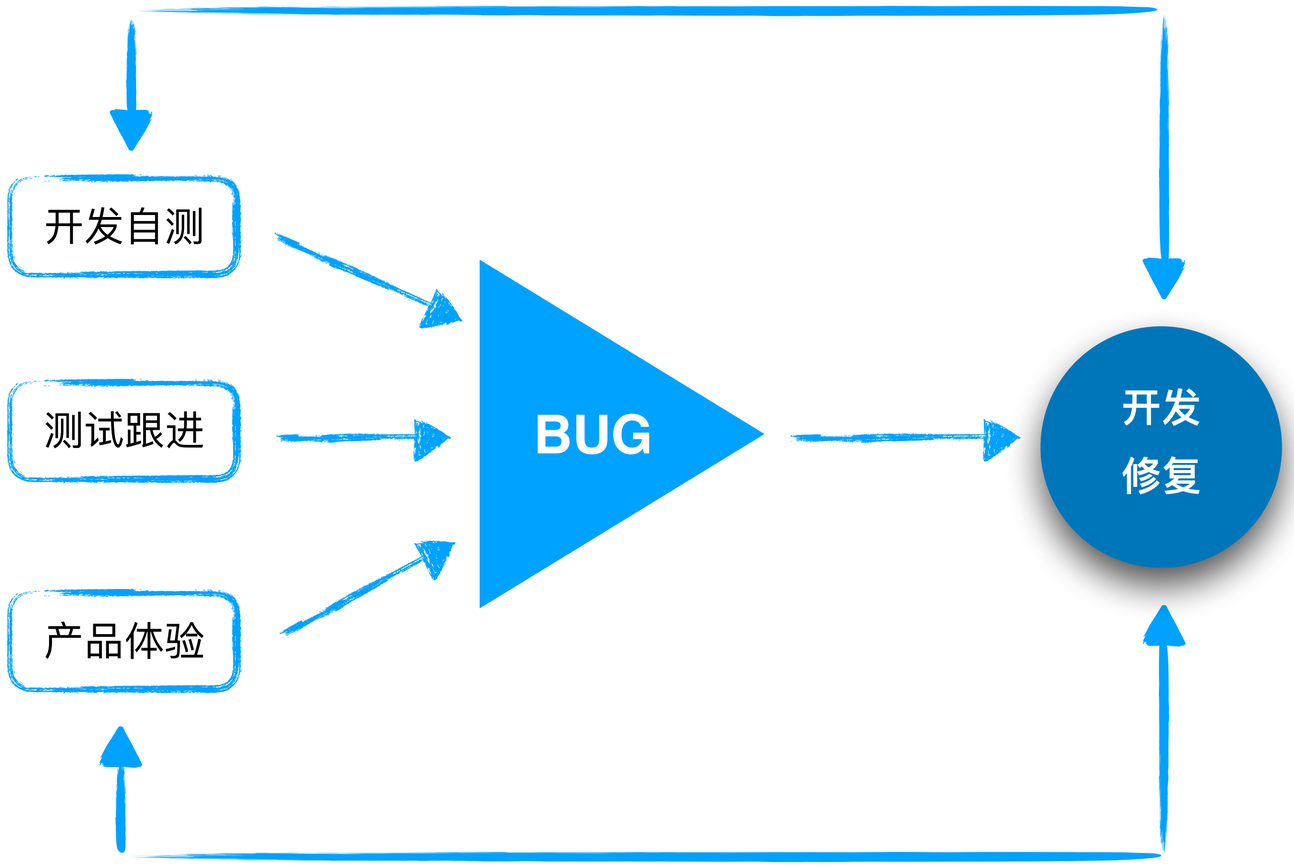
解决开发调试、测试环节的体验问题

桌面端

移动端

开发完成进入测试阶段

通过工具，优化效率



使用 TAPD， BUG 单管理闭环

设定聊天机器人，自动化反馈

测试完成后的发布阶段

如何规范发布流程?

发布覆盖

发布效率低

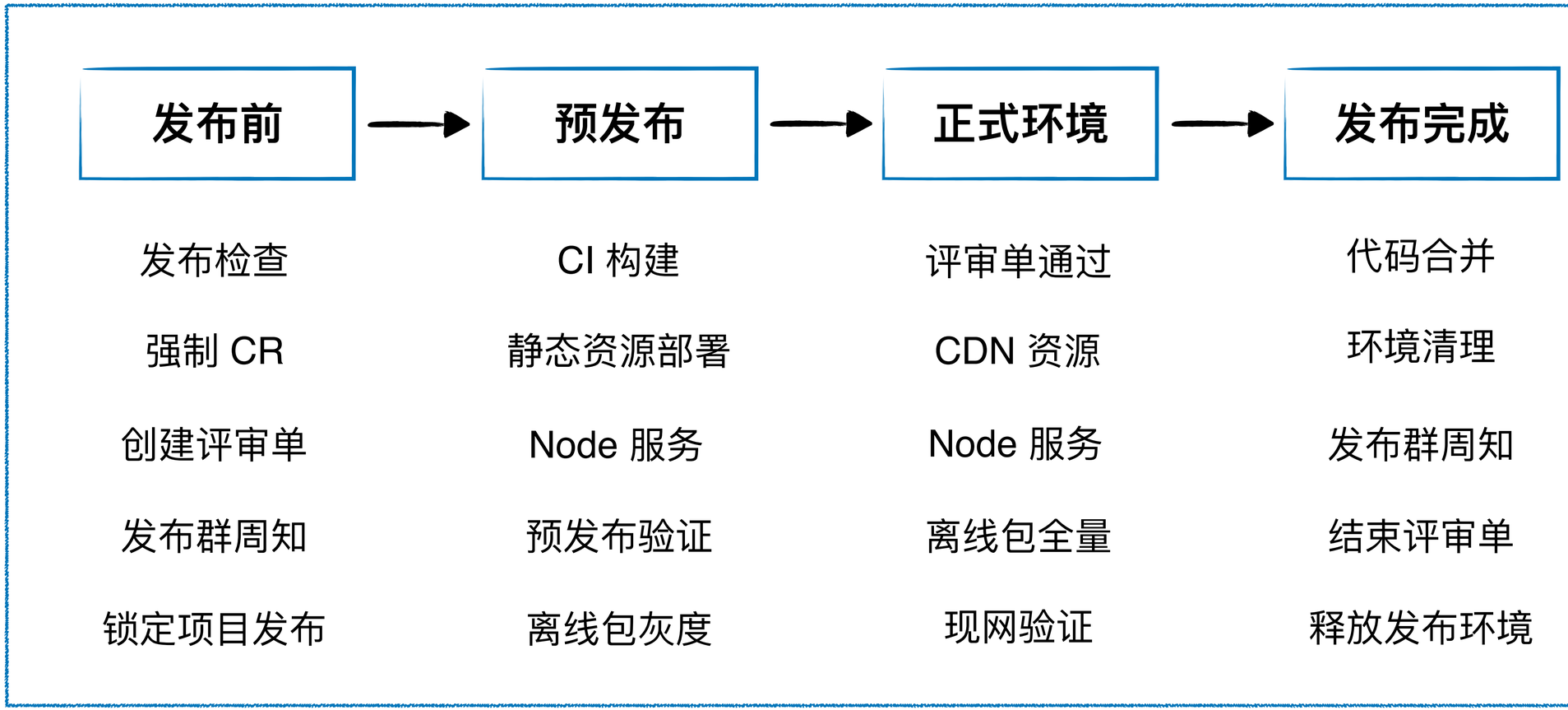
发布冲突

发布核心诉求：简单、高效、稳定

落地 GUI 工具 Thanos

规范发布流程 & 自动化流水线

发布周期



Thanos 工作台功能展示

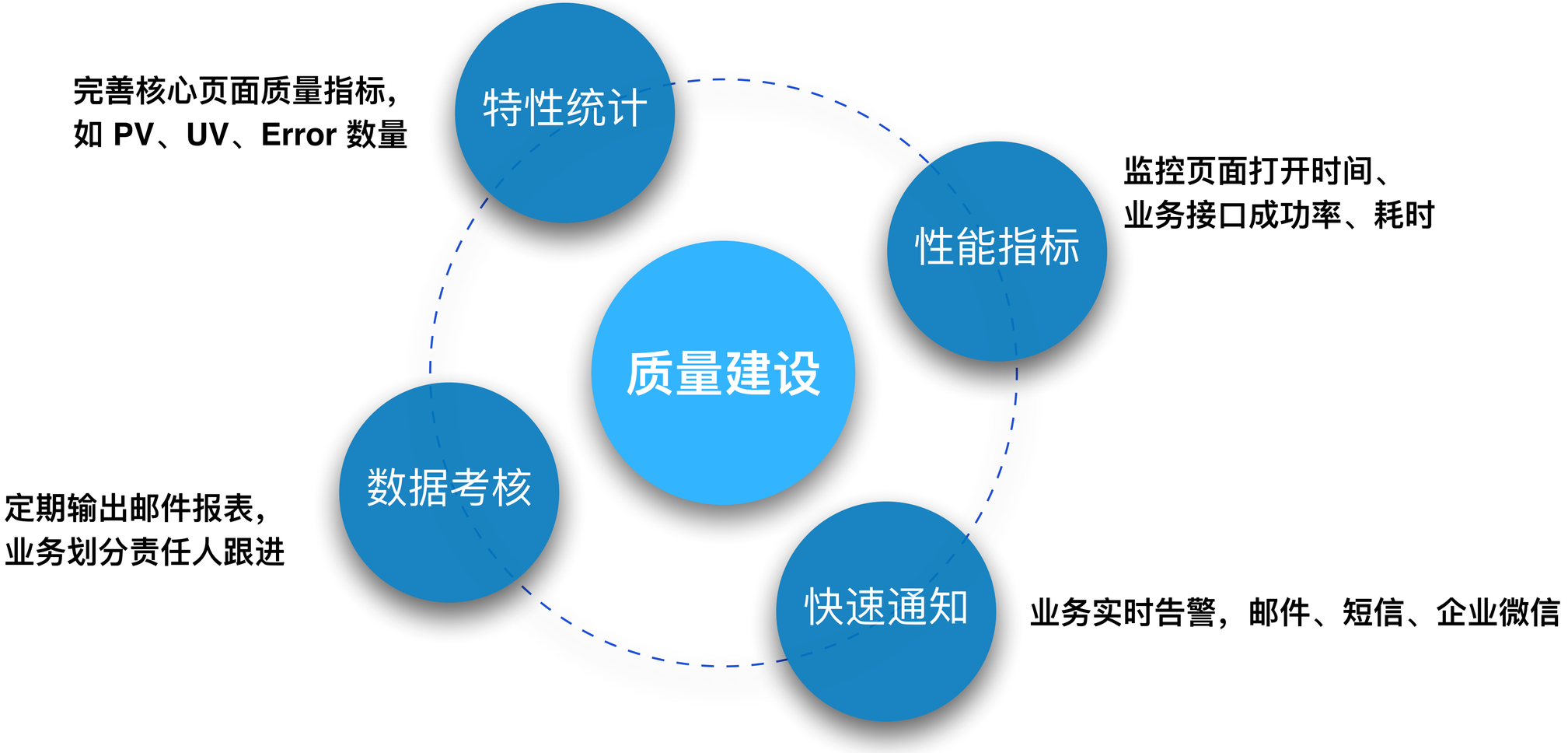
优化发布效率，降低发布风险

发布准备

发布完成

提升开发效率 ≠ 解决产品质量

监控体系建设



如何定位和解决问题

最大限度的还原问题

接入 Sentry

查看并展示错误信息，对移动端支持友好

全链路日志

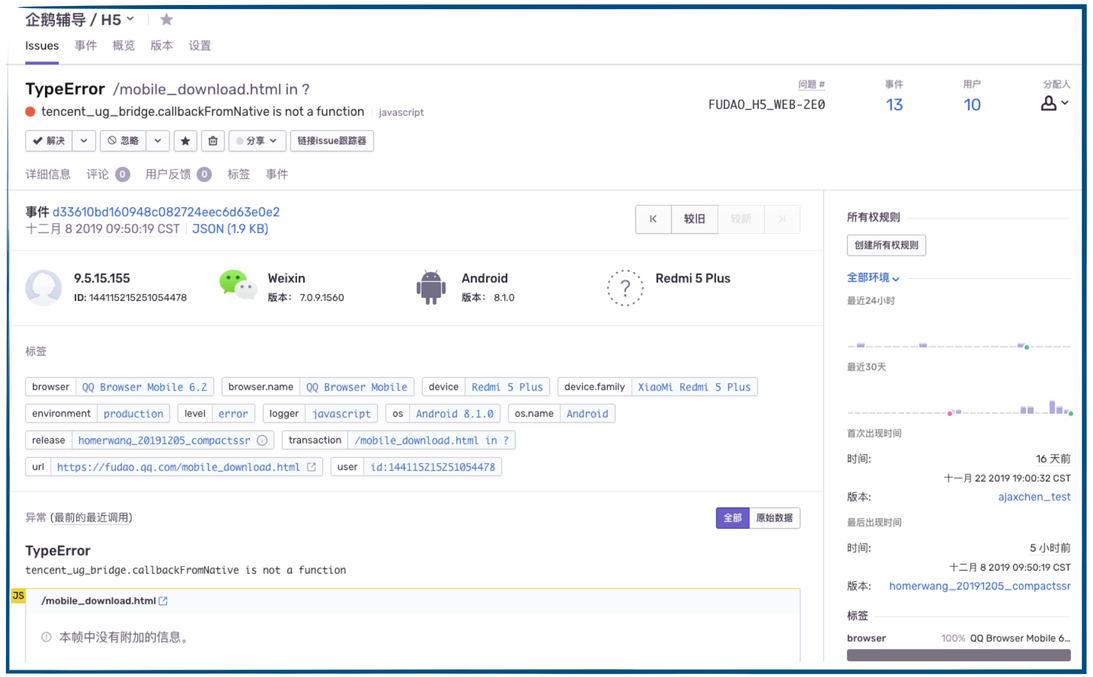
基于 ELK 存储，通过 requestId 完成从前到后的日志查询能力

离线日志建设

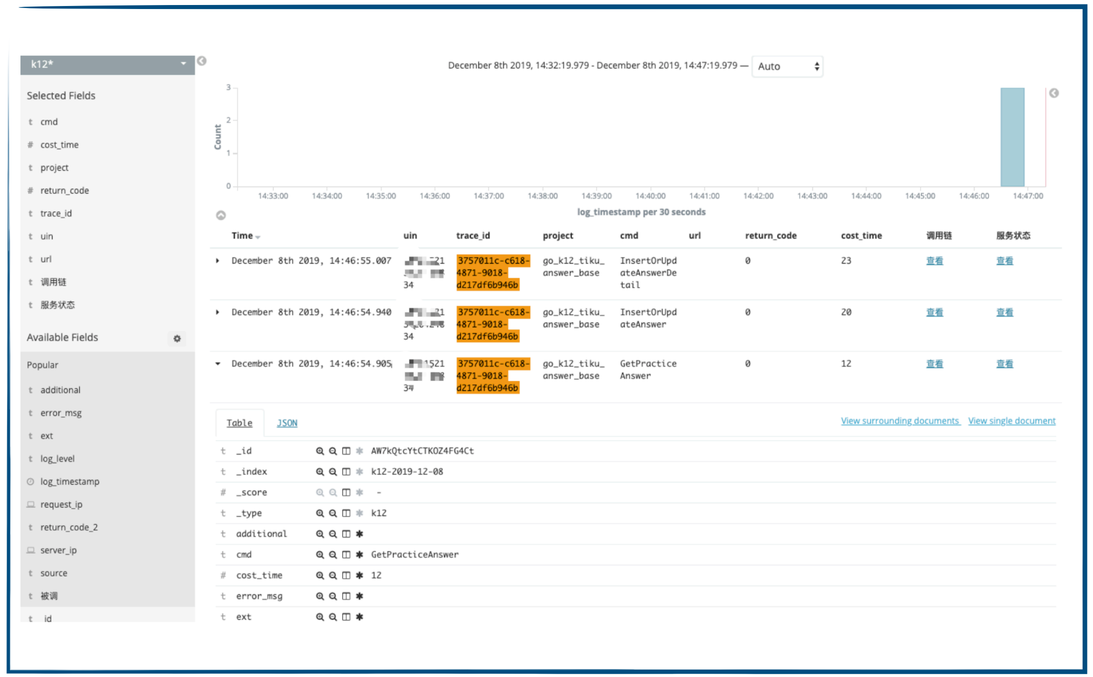
通过 APP 和 localStorage 进行大文件用户日志上传

定位问题工具展示

提升了开发定位问题的效率



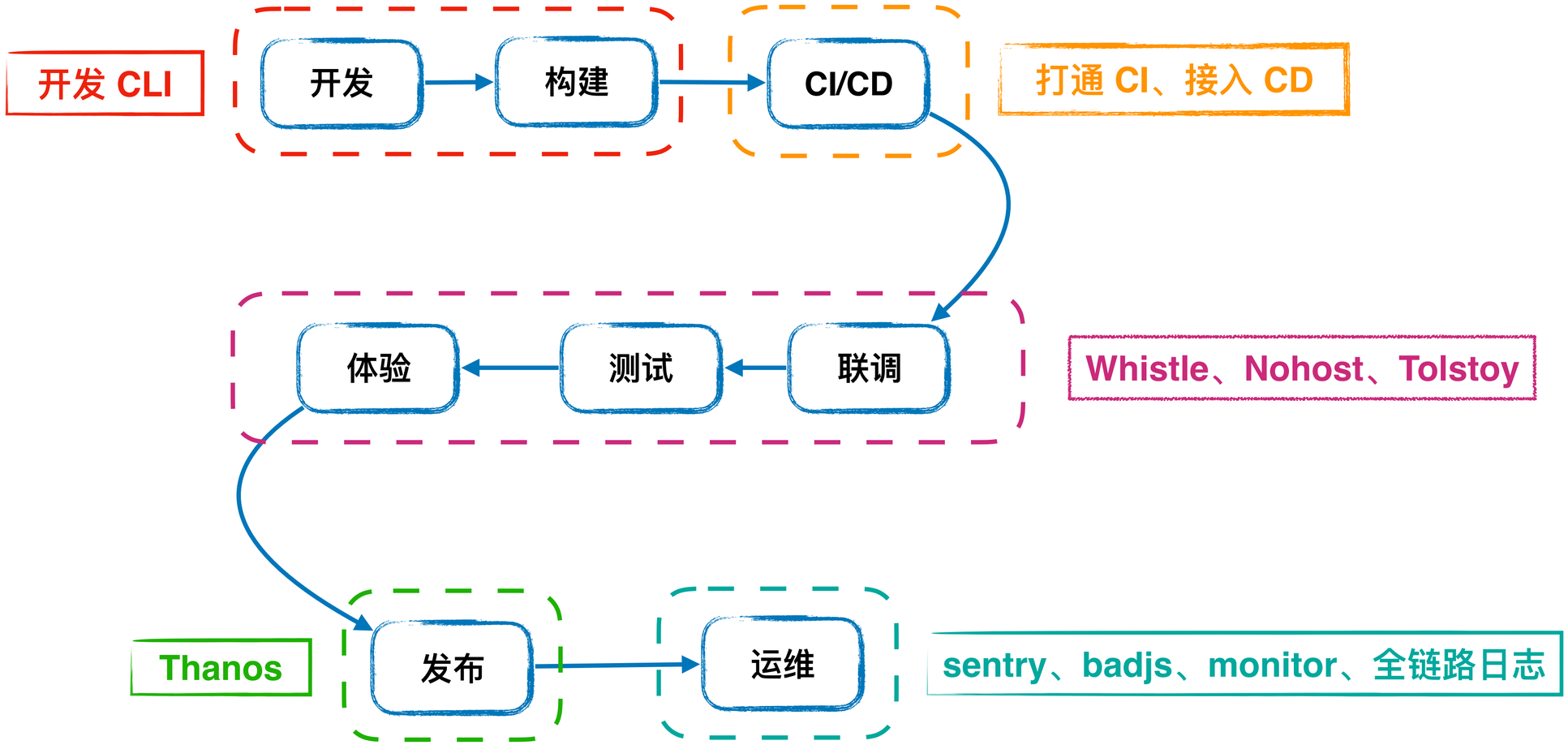
Sentry



全链路日志

总结开发周期的核心路径

研发流水线闭环

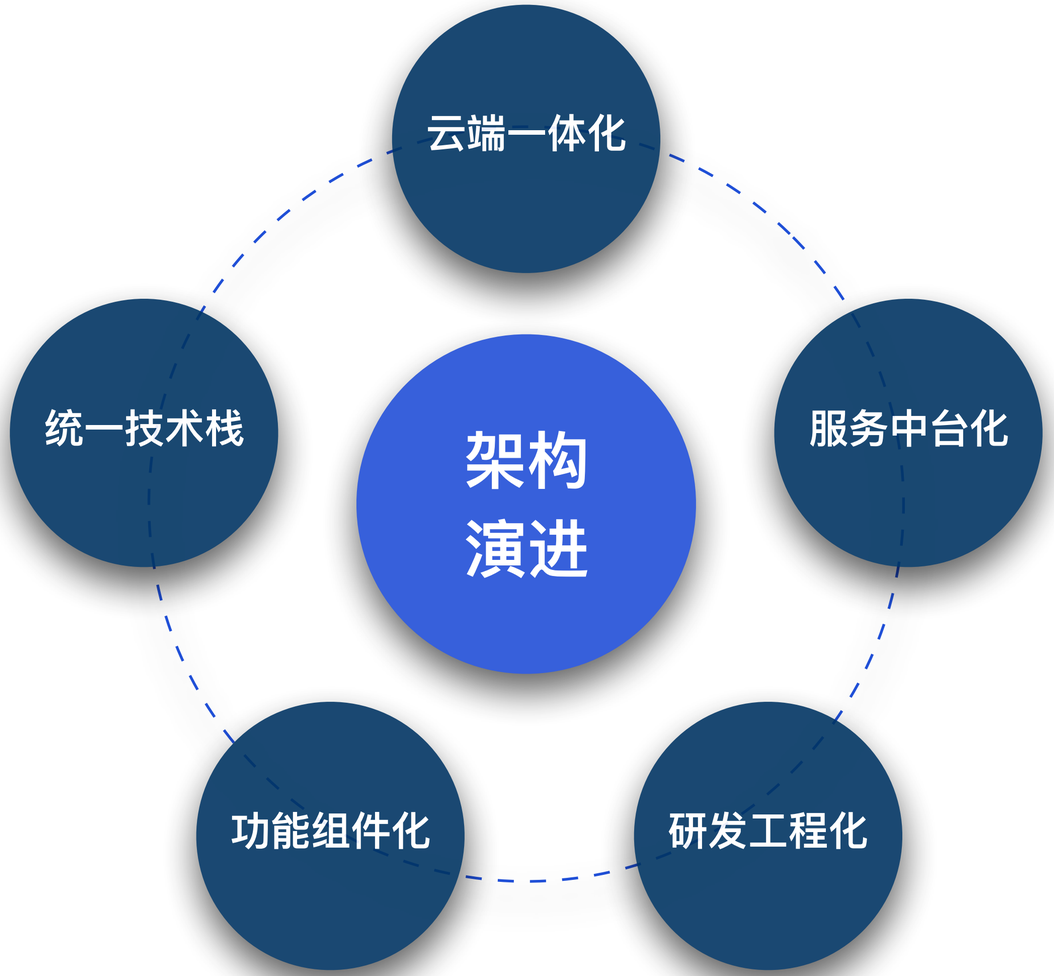


解决效率和提升质量的方法论

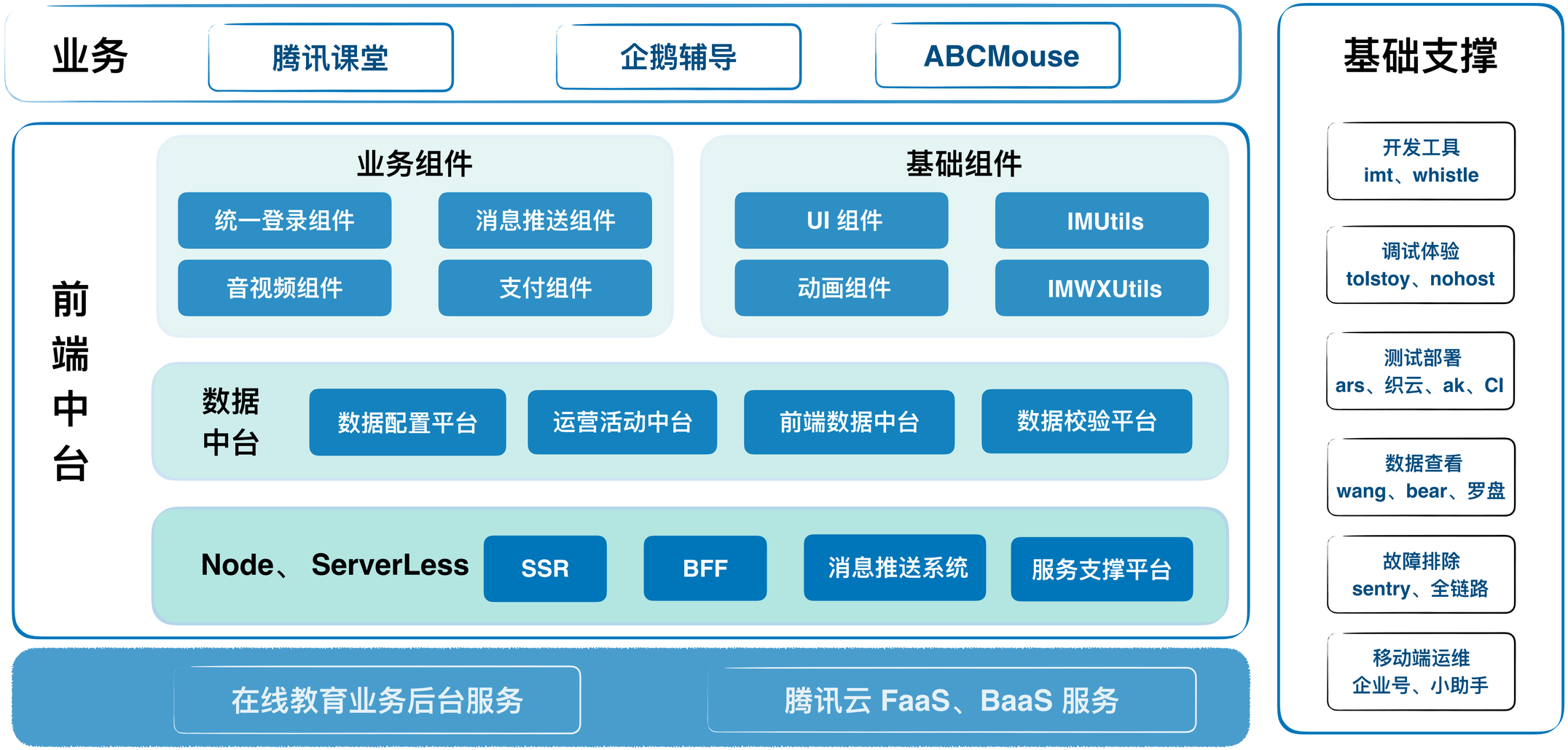
目标：帮助开发者提升开发幸福感

3、总结与展望

腾讯在线教育大前端架构演进思路

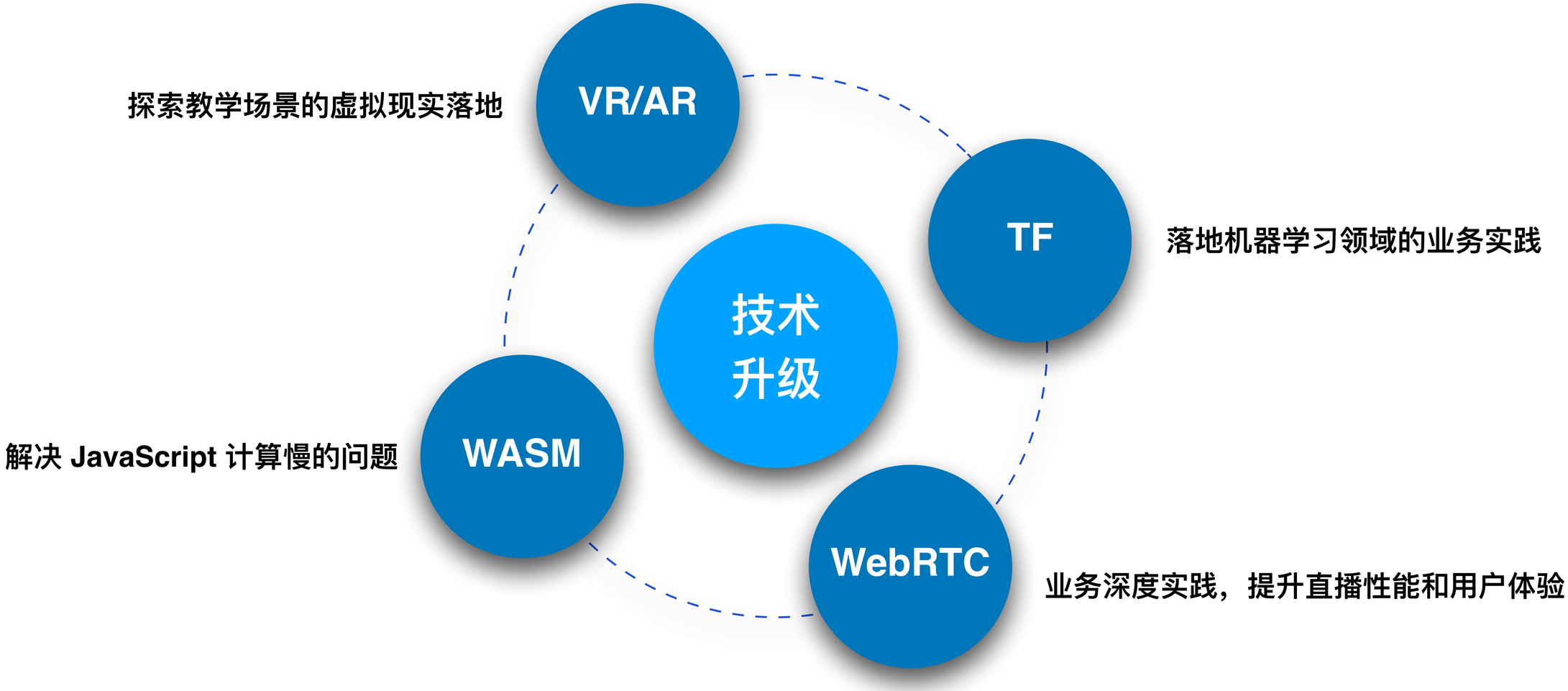


在线教育大前端中台架构全景图



未来的技术展望

新技术方案的探索和落地



回归本质

1、技术栈统一和组件化的落地是一个团队正式成熟的标志。

2、制定合理的规范和建设有效的工具是优化团队效率的有效手段。

3、基于业务的技术探索才能够让技术本身更有价值。

极客时间

一起领「敲代码神器」

Git 快捷口令超大鼠标垫

常看常记，运指如飞，不卡壳

扫码免费领取

仅限200份，先到先得

THANKS

GMTC
全球大前端技术大会

Q&A