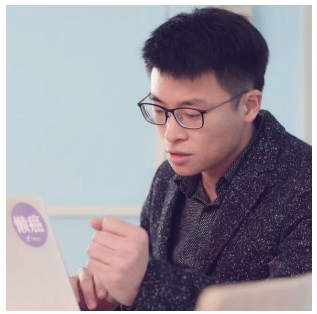
msup

字节跳动 Web 基建 Serverless 探索与实践

讲师简介



许骏宇
字节跳动 Web Infra 成员

毕业于杭州电子科技大学，曾就职于网易，现作为字节跳动 **Web Infra** 成员。

多年基础建设研发经验，对 **Web 基础架构、研发生态、服务发布运维体系、Serverless** 等方向有多年实战经验。

现阶段关注 Serverless 架构为 Web 场景带来的研发效率提升，字节跳动 Web 基建中的 Serverless 体系。

字节跳动终端技术架构 - Web Infra Team

APM Platform	Test Infra	低代码搭建
跨端解决方案	Web 开发引擎	移动中台
Web IDE	Code SandBox	体验性能
一站式研发平台	Serverless	Node.js 架构

Web Infra 公众号

讲在前面

Serverless 是什么？
（当下的主流认知角度）

参考：CNCF Serverless 白皮书

讲在前面

Web Infra 团队早期在字节内进行 FaaS + BaaS 的一些落地尝试后，发现：

主推优势：

- 开发简单
- 部署简单
- 运维简单 (自动扩容)

推广阻力：

- 历史工程迁移低效
- 大型项目是否适用 (大量微服务)

所以，我们重新从 **实际场景** 出发，「探索 Serverless 理念的**工程化落地**」。

今天主要内容

- 字节跳动 Web 基础设施服务的现状和挑战
- Serverless 化建设的思路和目标
- Web Infra 的 Serverless 平台技术实现方案
- Serverless 落地案例分享与未来展望

一、字节跳动 Web 基础设施服务的现状和挑战

（为什么探索 Serverless 的工程落地）

一、字节跳动 Web 基础设施服务的现状和挑战

在开始之前，带大家一起回顾一下：在迎来 **Serverless** 化变迁之前，字节跳动 **Web** 基础设施服务的模式变迁：

自建机房	IaaS:Infrastructure as a Service	CaaS: Container as a Service	PaaS: Platform as a Service	?
自建物理机，连入了 IP 公网	利用虚拟机技术，为开发者们开辟出一个个 1c 的虚拟机	虚拟机会宕机（硬盘写满，系统重启），为降低运维成本，利用 Docker & Kubernetes 技术，快速部署容器	计算资源平台化，平台操作完成资源（内存、CPU）申请。服务数量达到了数以万计	?

总体趋势：虚拟化、无服务器化、免运维化（Serverless）

一、字节跳动 Web 基础设施服务的现状和挑战

现阶段，Web 开发人员在服务研发领域（**常用 Node.js**）的的开发、部署、运维知识体系

一、字节跳动 Web 基础设施服务的现状和挑战

msup

汇总来自业务 Web 开发同学的槽点反馈

Node.js 研发环节槽点	发布环节槽点		运维环节槽点
需要在代码中主动引入 Node.js 性能监控分析组件	发布较慢：5min - 30min	缺乏与构建流程的联动	需要开发者提前申请好服务配额
构建运行命令无法通过 npm scripts 配置	申请服务资源效率较低	申请预览环境效率较低	缺乏扩缩容服务资源长期低水位

1) 对于开发者而言，整体的研发、迭代、运维仍需要考虑较多的 **服务器资源相关** 的事项：环境、实例数量；

2) 所以，我们可以围绕降低开发者的服务器资源感知，使开发者可以 **快速将代码转换成服务**，以实现 **降本、增效**展开优化；

综上，我们希望通过 **Serverless** 化改造，来改善已有的 Web 开发人员的 **服务研发模式**。

二、Serverless 化建设的思路和目标

二、Serverless 化建设的思路和目标

1) 存量项目：提升 已有项目 的构建、发布、运维体验

2) 增量项目：基于 Serverless 理念探索业务开发的最佳实践

明确做 Serverless 化建设的思路和目标

三、Web Infra 的 Serverless 平台技术实现方案

二、Serverless 化建设的思路和目标

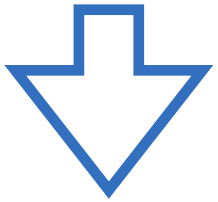
关于 Serverless，我们做了一些云计算厂商的调研，我们会发现云厂商对于 Serverless 有两种形态：
1. FaaS & BaaS；2. 服务 Serverless 托管；

	Google Cloud Platform	Microsoft Azure	AWS
FaaS & BaaS	• Google Cloud Functions：函数无服务器开发。 • Firebase：集成一些轻量级的存储服务，对客户端 / 前端研发提供相对简单的服务开发、运维体验。	• Azure Functions，来自微软公有云的 Serverless 实现。	• AWS Lambda，最早被大众所认可的 Serverless 实现
服务 Serverless 托管	• Google App Engine	• Create a Node.js web app in Azure	• AWS Web Services

三、Web Infra 的 Serverless 平台技术实现方案

1. 存量项目：提升 **已有项目** 的构建、发布、运维体验

2. 增量项目：基于 Serverless 理念探索业务开发的最佳实践



1. 服务 Serverless 托管

2. FaaS + BaaS 研发模式升级

3. JS Worker 边缘高效扩容

三、Web Infra 的 Serverless 平台技术实现方案

3.1 面向存量项目的解决方案

- 需求：提升 已有项目 的构建、发布、运维体验；
- 分析：固有的直接基于 kubernetes 的发布、运维模式有提升空间；
- 解决：搭建 Serverless 平台，实现 **服务 Serverless 托管**基础能力；

三、Web Infra 的 Serverless 平台技术实现方案

支持服务的 0 - 1 冷启
（Scaling to zero）

2) 冷启足够快: Warming Pool, 最小实例, bundle

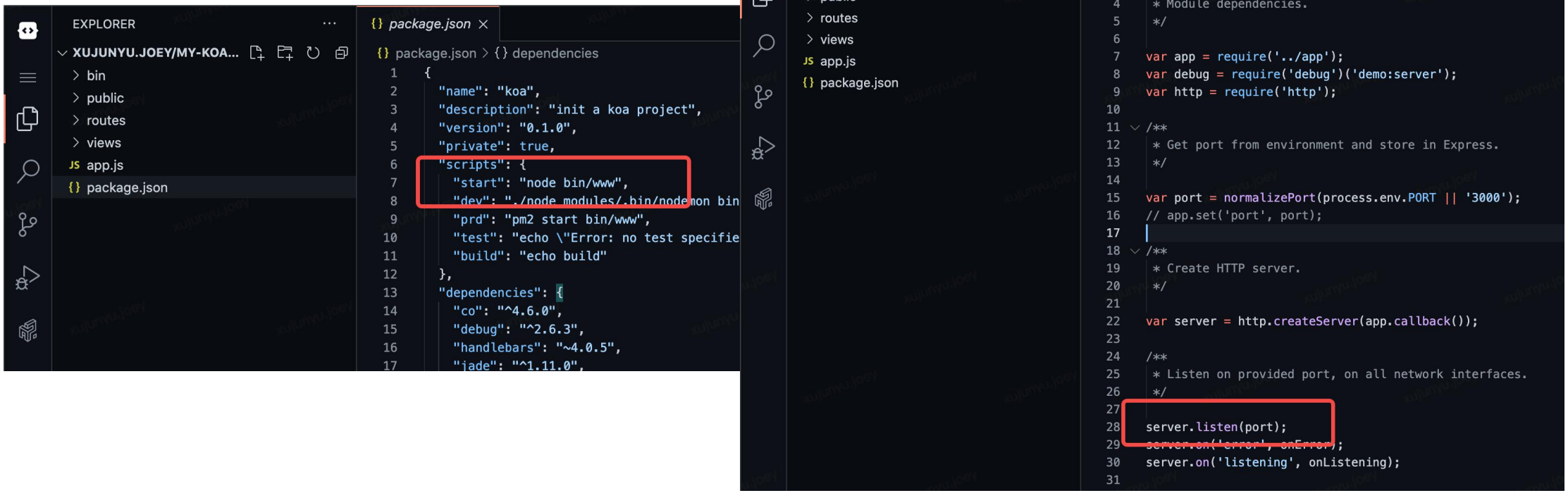
3) 支持运行时扩容

4) 开发者体验建设

服务 Serverless 托管核心特性

三、Web Infra 的 Serverless 平台技术实现方案

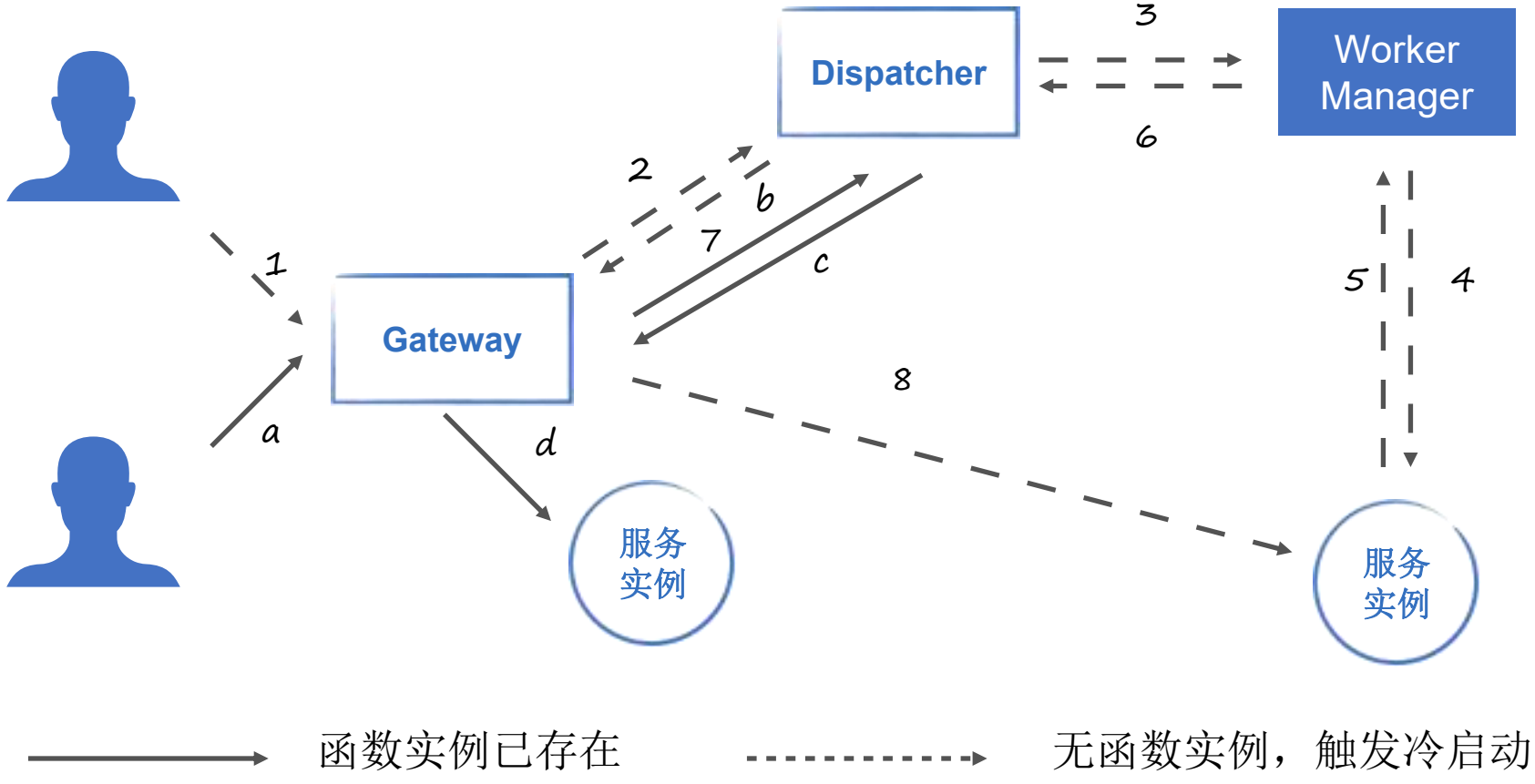
支持**服务**的 0 - 1 冷启，服务定义：
1. npm start scripts: node app.js
2. app.js 中存在端口监听



核心特性一：服务的 0 - 1 冷启

三、Web Infra 的 Serverless 平台技术实现方案

服务的 0 - 1 冷启策略



核心特性一：服务的 0 - 1 冷启

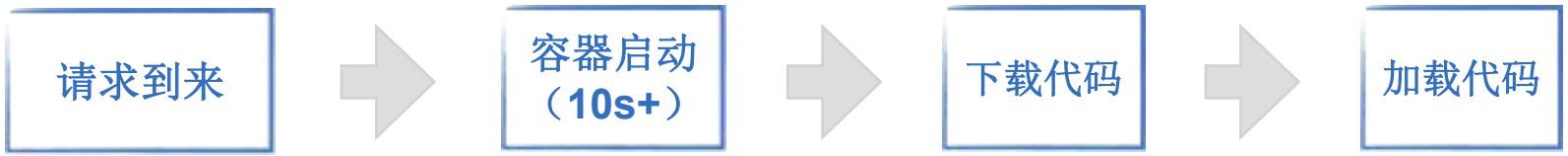
三、Web Infra 的 Serverless 平台技术实现方案

服务冷启动慢怎么办？

1）Pod 池化

2）预留实例

3）代码 bundle



核心特性二：冷启动优化

三、Web Infra 的 Serverless 平台技术实现方案

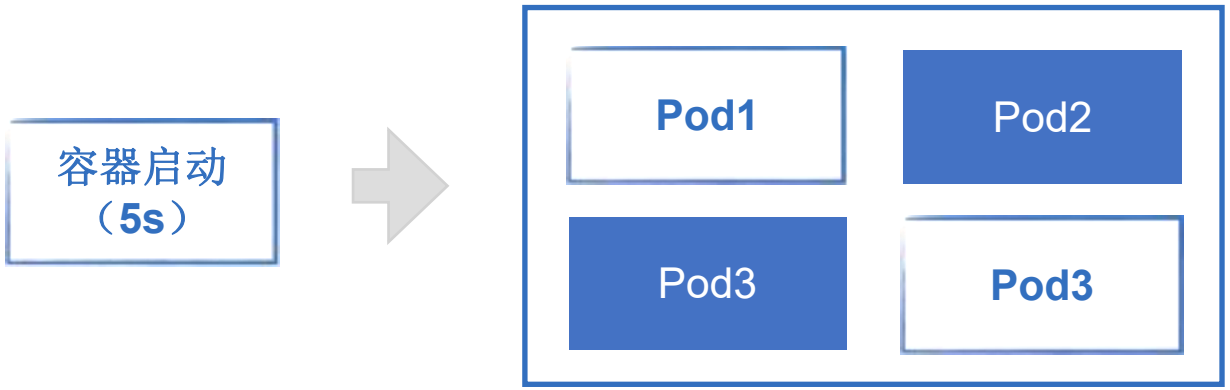
服务冷启动慢怎么办？

1）Pod 池化

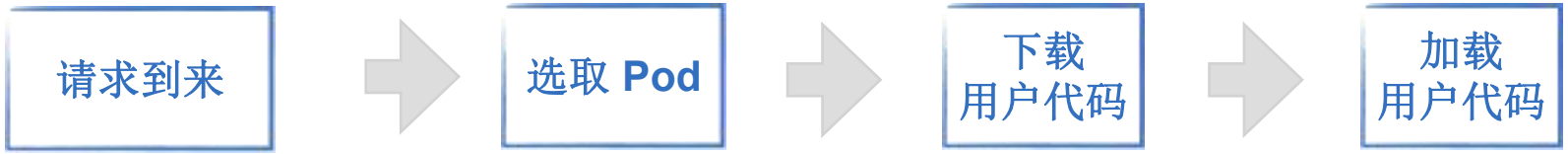
2）预留实例

3）代码 bundle

启动容器提前待命



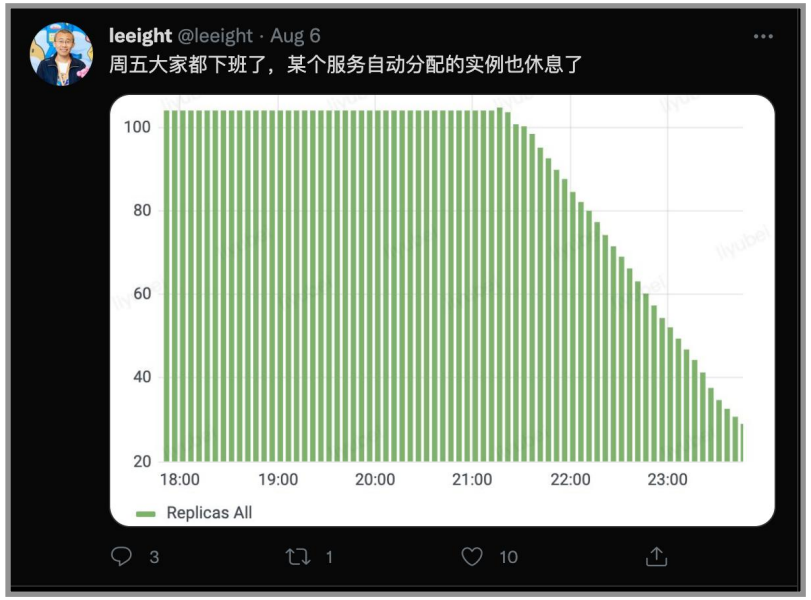
请求到来时使用容器



核心特性二：冷启动优化

三、Web Infra 的 Serverless 平台技术实现方案

如何提升冷启动效率？
1）Pod 池化
2）预留实例
3）代码 bundle



优势：规避冷启，提升首次访问体验
缺陷：资源利用率降低

核心特性二：冷启动优化

三、Web Infra 的 Serverless 平台技术实现方案

如何提升冷启动效率？
1）Pod 池化
2）预留实例
3）代码 bundle

冷启动耗时：3s

冷启动耗时：300ms

核心特性二：冷启动优化

三、Web Infra 的 Serverless 平台技术实现方案

实例变化模型效果呈现：1) 网关实时流量；2) 系统资源情况；

核心特性三：自动扩容

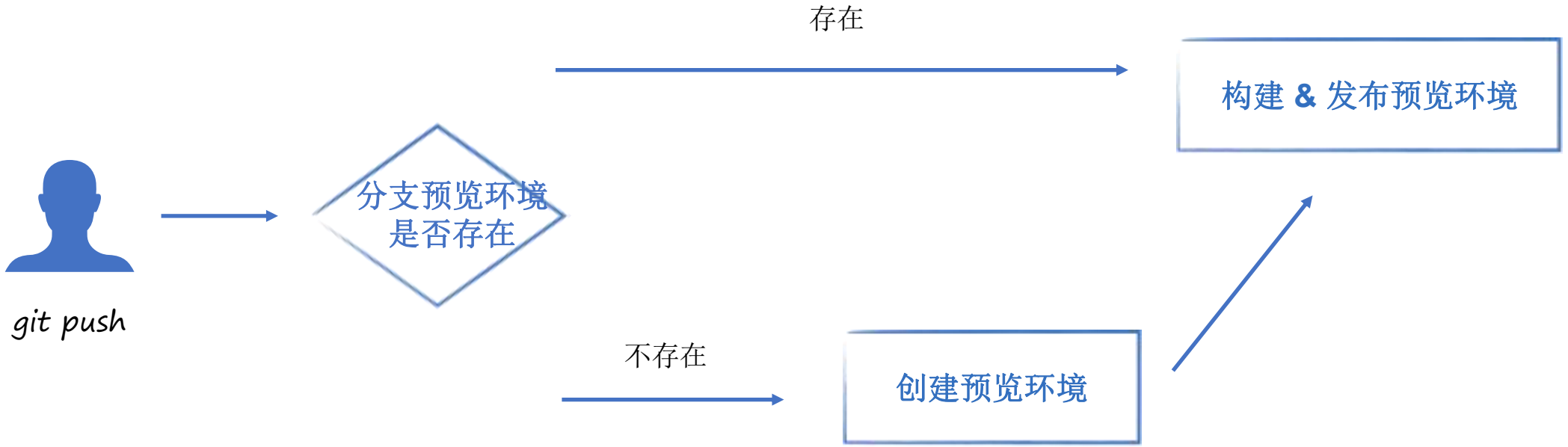
三、Web Infra 的 Serverless 平台技术实现方案

关键点：

1) GIT 驱动，预览环境的无缝打通

2) 一键开启 Node.js Inspector

3) 快速创建、云端研发 …

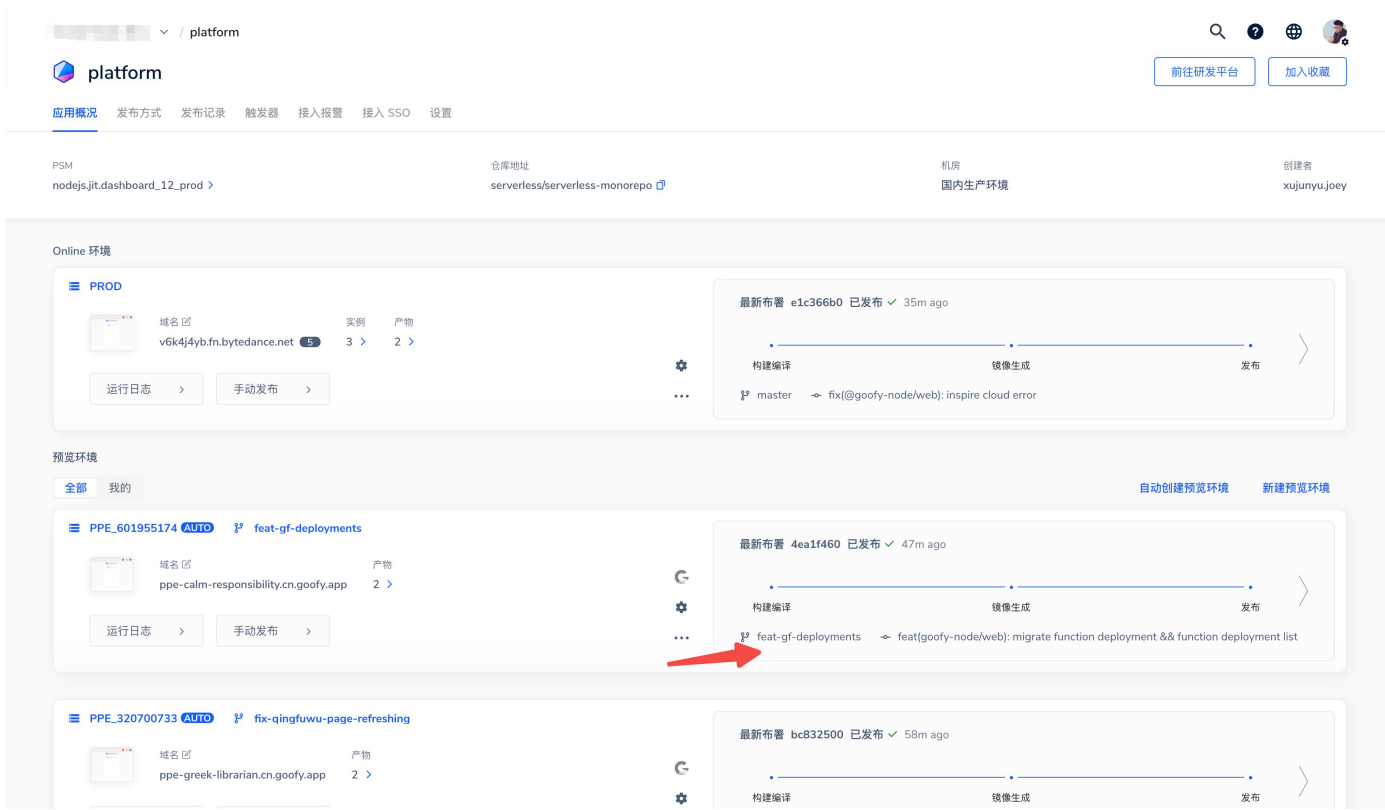


核心特性四：开发者体验建设

三、Web Infra 的 Serverless 平台技术实现方案

关键点：

1) **GIT 驱动，预览环境的无缝打通**
2) 一键开启 Node.js Inspector
3) 快速创建、云端研发 …

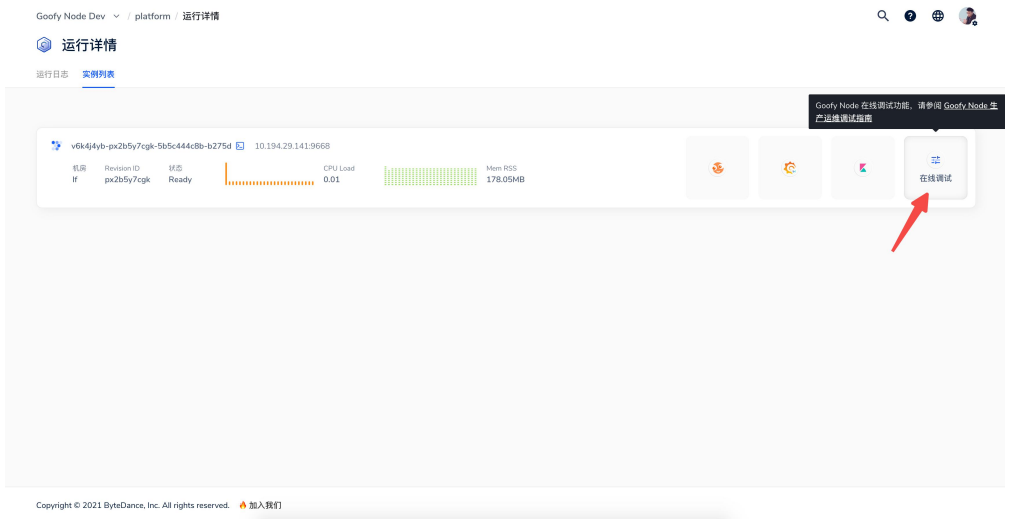


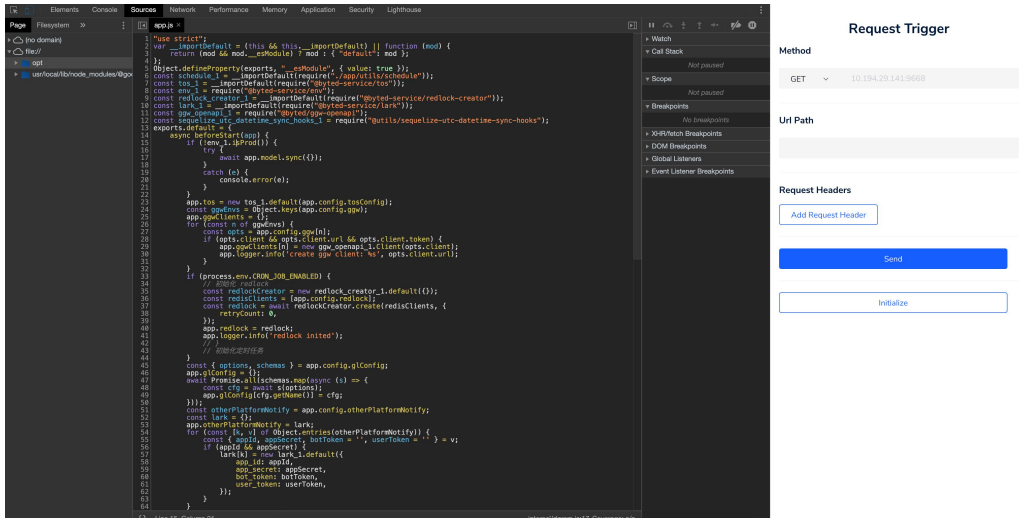
核心特性四：开发者体验建设

三、Web Infra 的 Serverless 平台技术实现方案

关键点：

1) GIT 驱动，预览环境的无缝打通
2) **一键开启 Node.js Inspector**
3) 快速创建、云端研发 …





核心特性四：开发者体验建设

三、Web Infra 的 Serverless 平台技术实现方案

存量项目的 Serverless 化的整体效果：

	服务 Serverless 托管方案收益
服务发布	1）发布效率大大提升，15min -> 1min（流程自定义）； 2）支持构建源码后发布的能力； 3）发布前可带有离线环境启动检查能力；
扩容 & 运维	1）扩容效率提升，遇到平常的峰值流量可较好适应；
服务申请	1）随时创建服务，大部分情况下无需再走审批流程（大流量除外）
创建特性预览环境	1）随时创建特性预览环境，无需走审批流程

3.2 面向增量项目的解决方案

- 需求：基于 Serverless 理念探索业务开发的最佳实践
- 分析：3.1 提供的方案只是针对存量场景的优化，我们是否可以面向未来设计更好、体验更加的开发、部署、运维模式；
- 解决：1) FaaS + BaaS 研发模式升级; 2) JavaScript Worker 落地轻量级极速扩容场景；

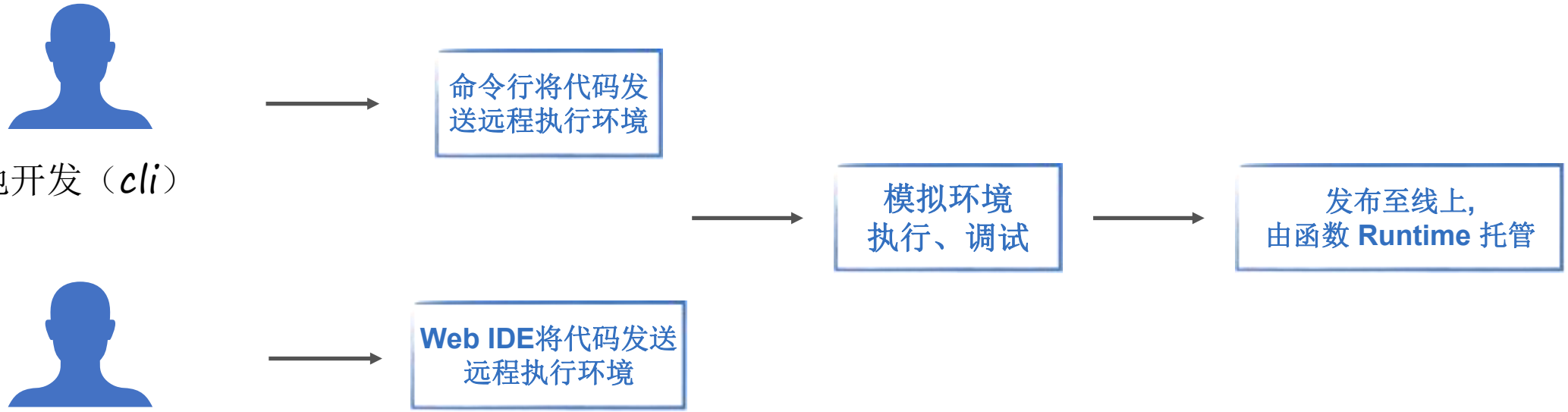
三、Web Infra 的 Serverless 平台技术实现方案

Serverless 优先的研发模式

1) FaaS 研发模式

2) BaaS 研发模式

3) JavaScript Worker 极限扩容

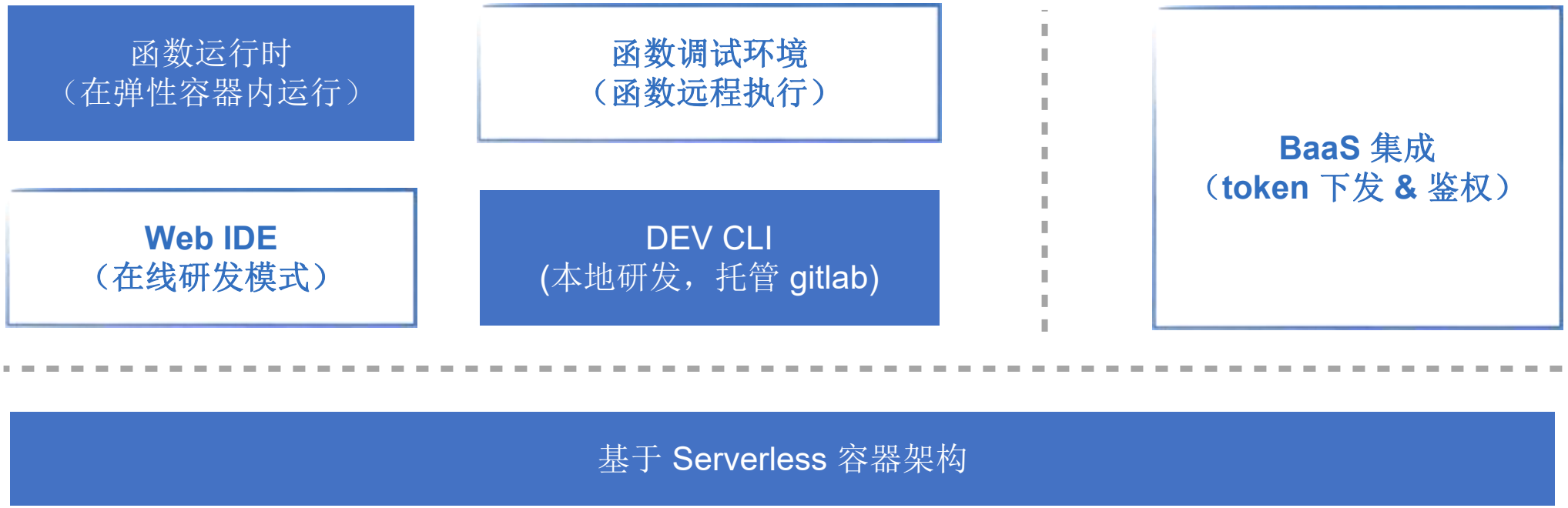


FaaS 主体用户流程

三、Web Infra 的 Serverless 平台技术实现方案

Serverless 优先的研发模式

1) **FaaS 研发模式**
2) BaaS 研发模式
3) JavaScript Worker 极速扩容



FaaS 基建主体组成

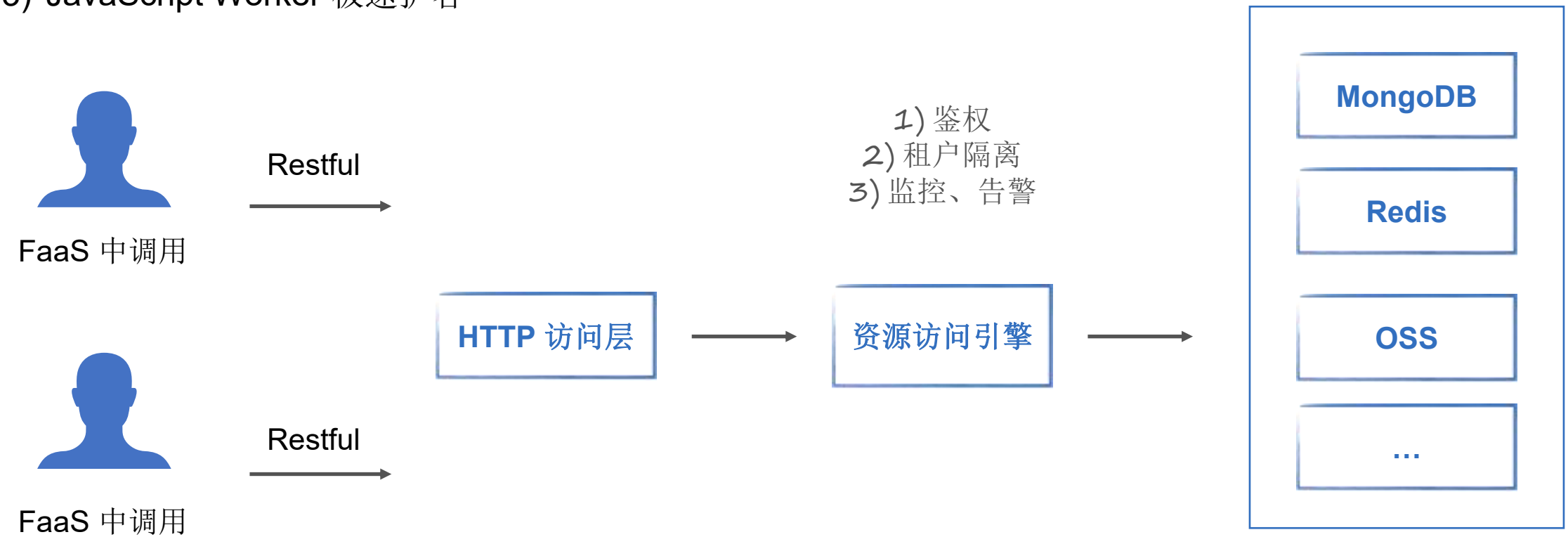
三、Web Infra 的 Serverless 平台技术实现方案

Serverless 优先的研发模式
1) FaaS 研发模式
2) BaaS 研发模式
3) JavaScript Worker 极速扩容

BaaS 能力覆盖范围

三、Web Infra 的 Serverless 平台技术实现方案

Serverless 优先的研发模式
1) FaaS 研发模式
2) BaaS 研发模式
3) JavaScript Worker 极速扩容



BaaS 运行时主体路径

三、Web Infra 的 Serverless 平台技术实现方案

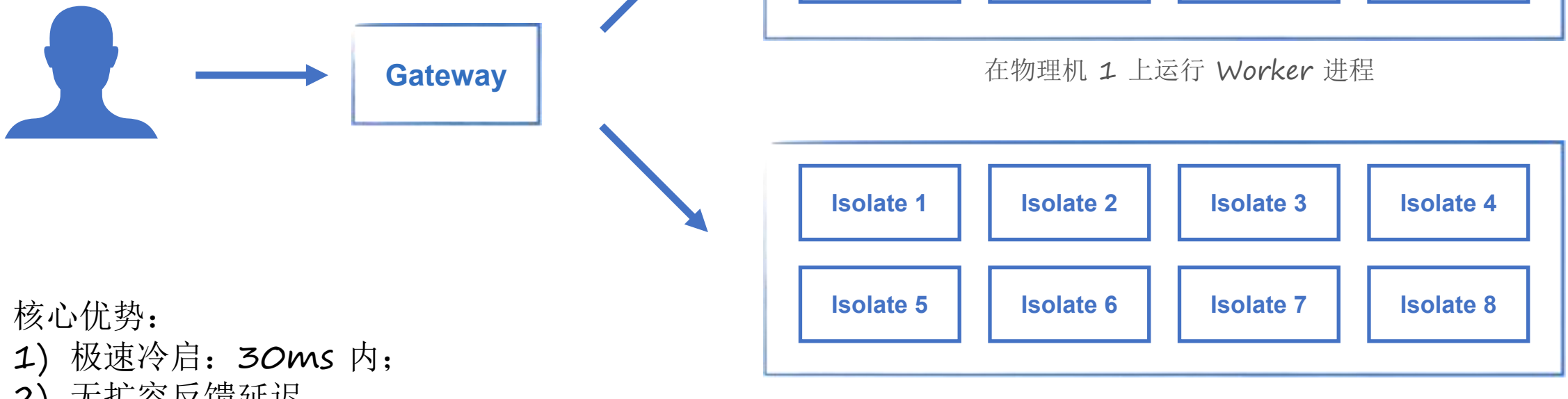
msup

FaaS + BaaS 方案核心收益

	FaaS + BaaS 方案核心收益
研发 / 调试效率	1）所见即所得，改代码即可调试调用； 2）开发者屏蔽资源细节，不再关心：函数托管到哪里，后端资源容量；
运维相关	1）以函数作为业务单位，单个接口的冷启动较小，规避冷启动问题； 2）支持对业务内不同函数设置不同的实例套餐；
场景相关	1）比较适合 RPC BFF 层场景，开发者创建模版工程后，由用户简单改改即可创建一个接口；

三、Web Infra 的 Serverless 平台技术实现方案

Serverless 优先的研发模式
1) FaaS 研发模式
2) BaaS 研发模式
3) **JavaScript Worker 极速扩容**



核心优势：
1) 极速冷启：30ms 内；
2) 无扩容反馈延迟

四、Serverless 落地案例分享与未来展望

四、Serverless 落地案例分享与未来展望

合作中的业务团队

四、Serverless 落地案例分享与未来展望

818（电商）

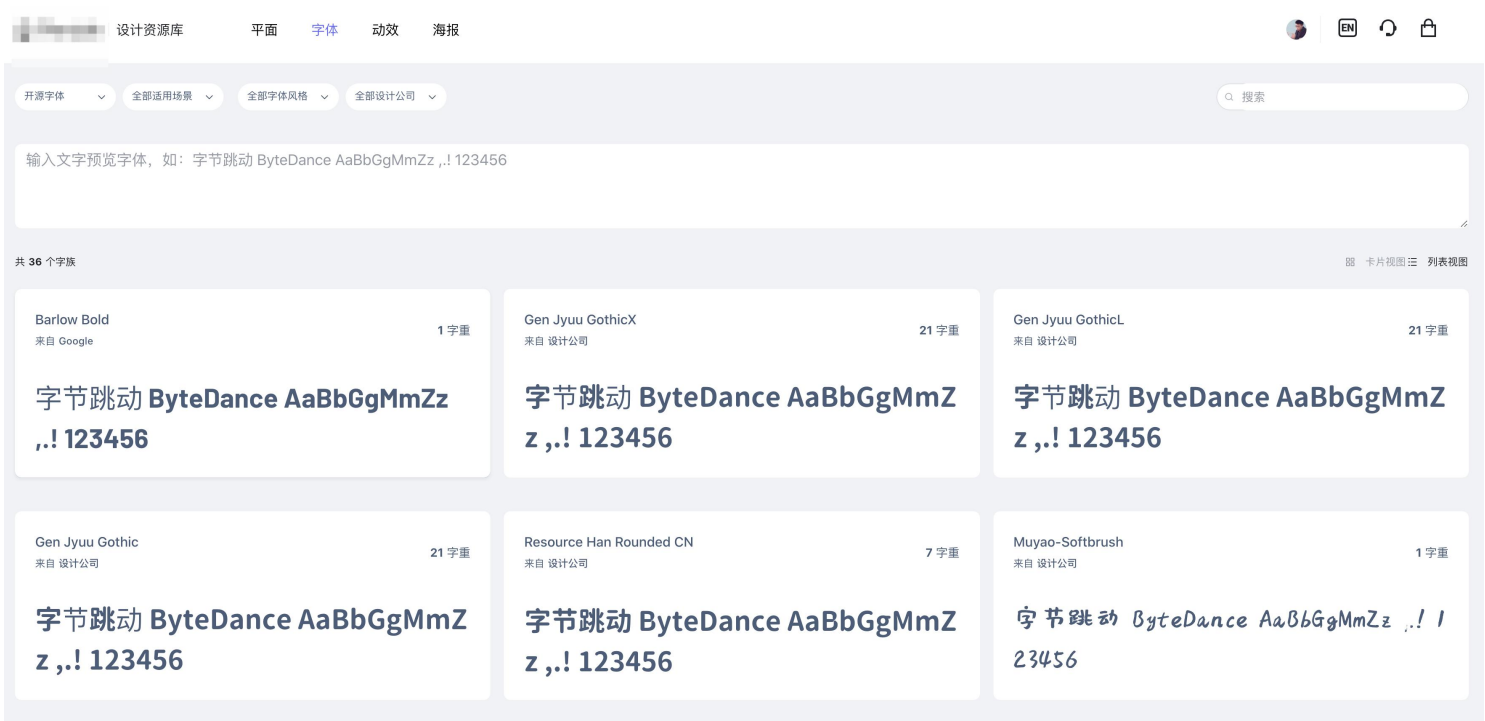
直播间（直播）

武林闲侠（游戏）

Bytedance NPM 包管理

服务 Serverless 托管落地案例（2w + 服务，含测试）

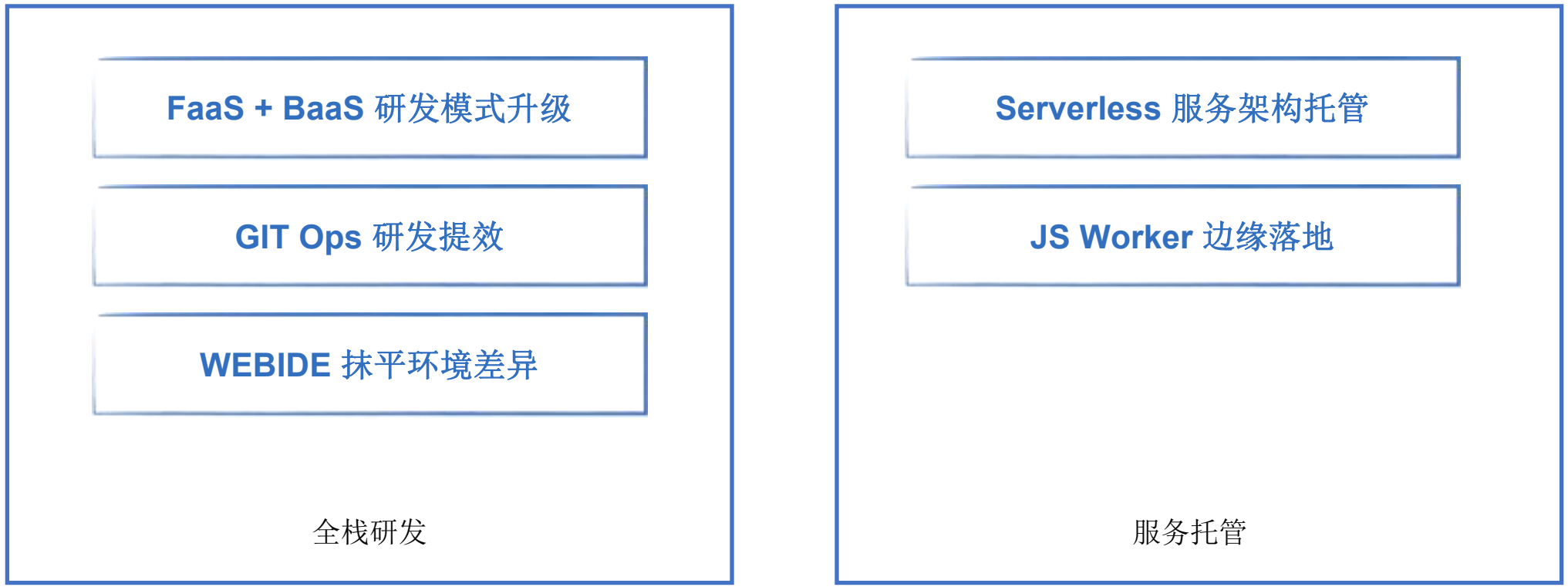
四、Serverless 落地案例分享与未来展望



JavaScript Worker 落地案例：M 站（头条）

FaaS + BaaS 落地案例：字体下发业务（字体业务）

四、Serverless 落地案例分享与未来展望



未来展望

总结

1) 字节跳动 Web 基础设施服务的现状和挑战（为什么要做 Serverless）
2) Serverless 化建设的思路和目标: 适配存量和增量场景
3) 具体的 Serverless 化建设思路: 服务 Serverless 托管，FaaS & BaaS，JS Worker 等大致思路
4) Serverless 落地案例分享与未来展望

谢谢观看

加入我们

关注 Web Infra 公众号

关注msup公众号

获取更多AI落地实践

麦思博(msup)有限公司是一家面向技术型企业的培训咨询机构，携手2000余位中外客座导师，服务于技术团队的能力提升、软件工程效能和产品创新迭代，超过3000余家企业续约学习，是科技领域占有率第1的客座导师品牌，msup以整合全球领先经验实践为己任，为中国产业快速发展提供智库。