

云原生微服务下一站

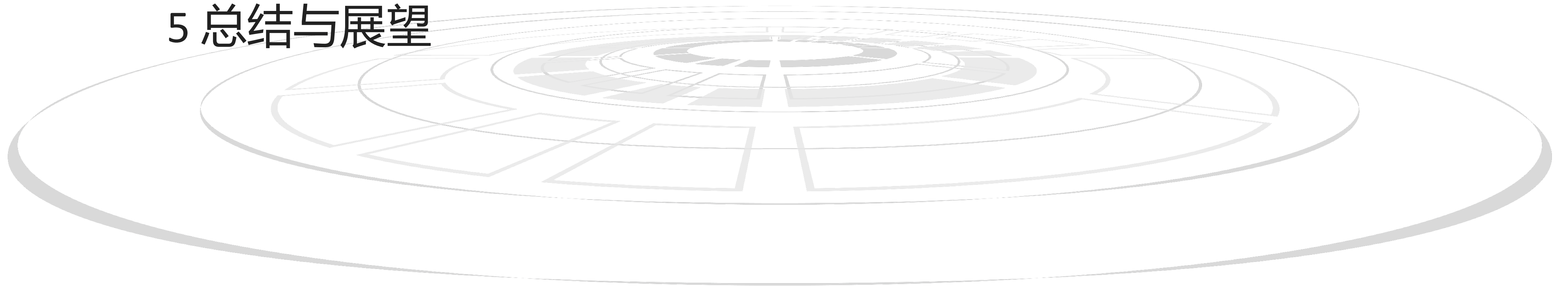
蚂蚁 SOFAServerless 新架构的探索与实践

赵真灵 / 蚂蚁中间件应用服务



目录

- 1 蚂蚁微服务背景
- 2 SOFAServerless 的解法与效果
- 3 SOFAServerless 的运维调度平台介绍
- 4 蚂蚁的实践经验与案例
- 5 总结与展望



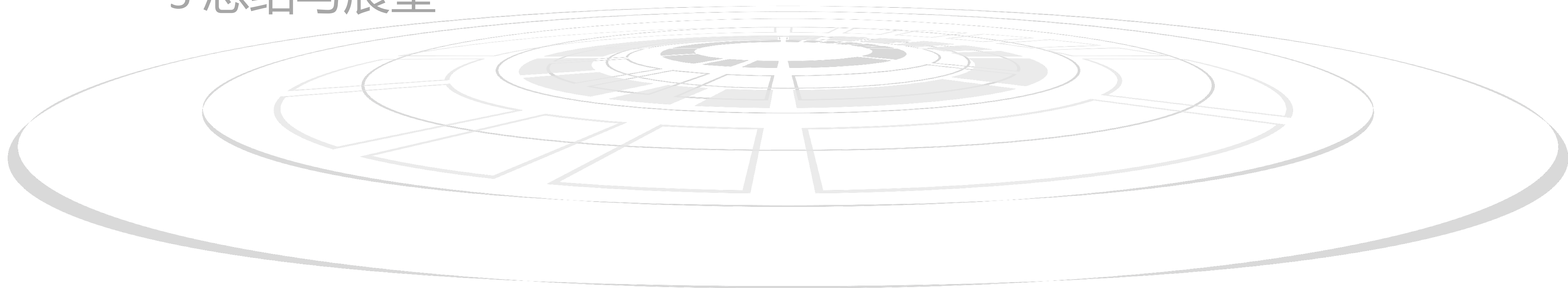
1 蚂蚁微服务背景

2 SOFAServerless 的解法与效果

3 SOFAServerless 的运维调度平台介绍

4 蚂蚁的实践经验与案例

5 总结与展望



蚂蚁微服务背景

发展年代久

15年 +

规模庞大

15W + 服务

全面容器化
K8S / PaaS

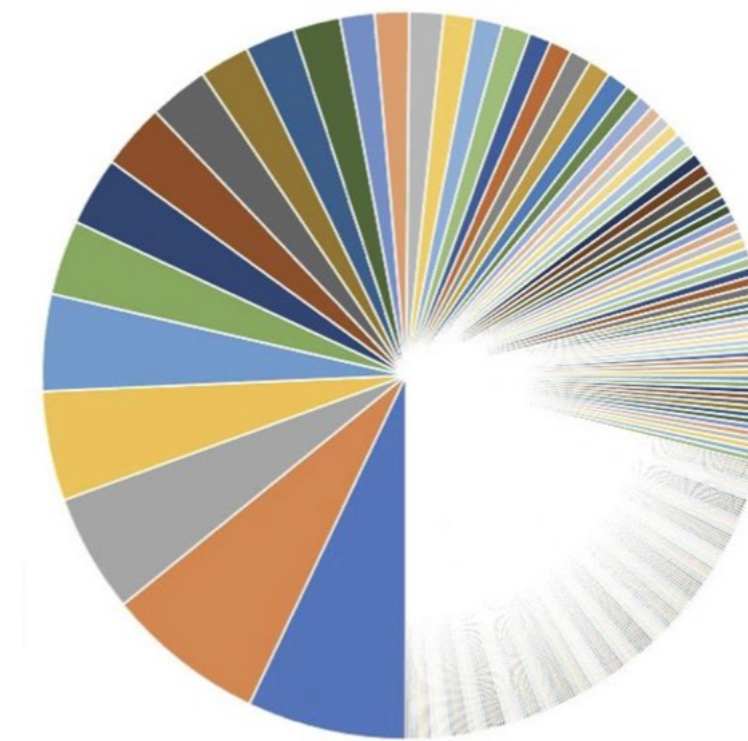
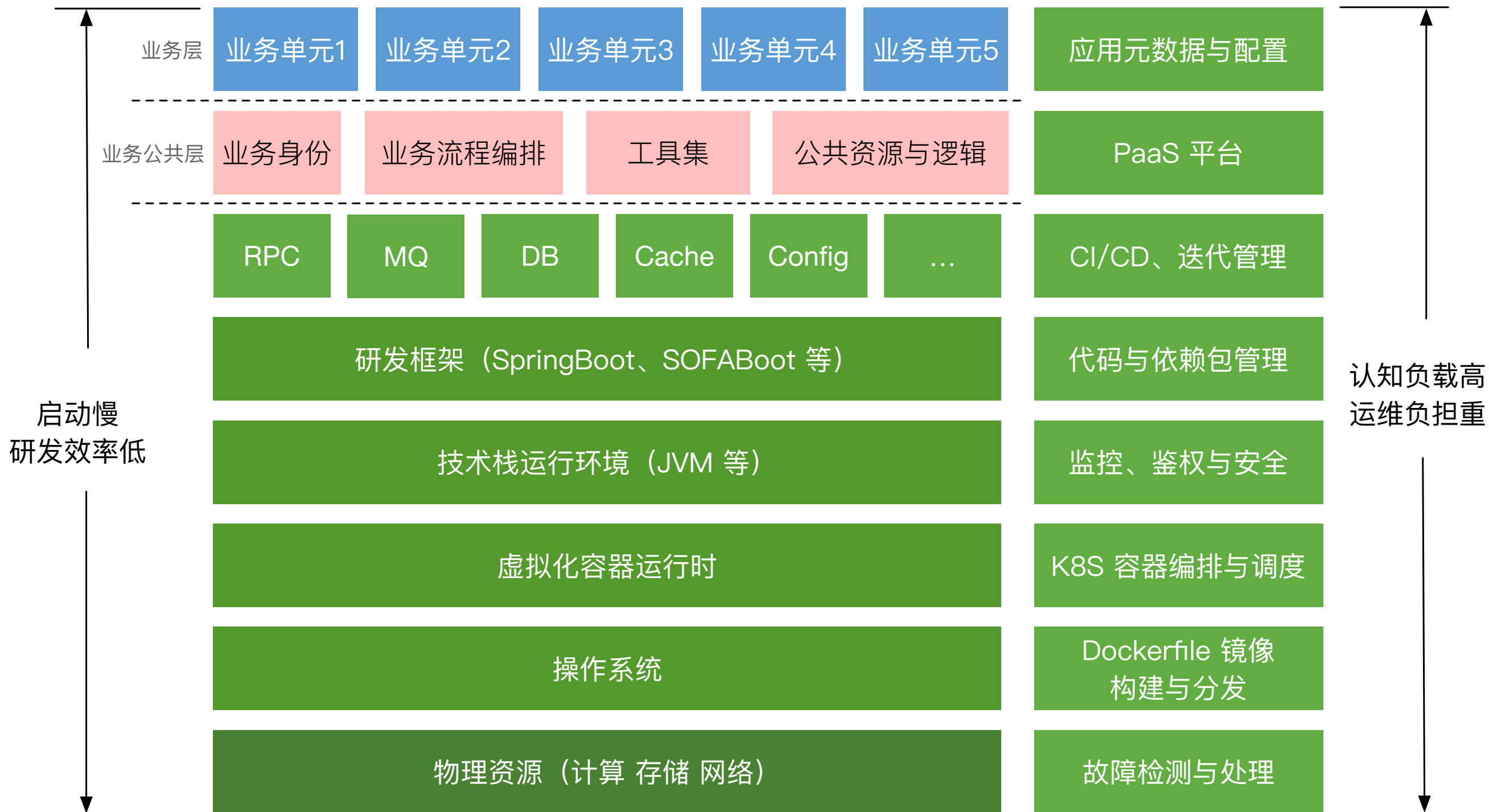
Java 语言为主/
多语言
NodeJs, Python, Golang



资源与效率

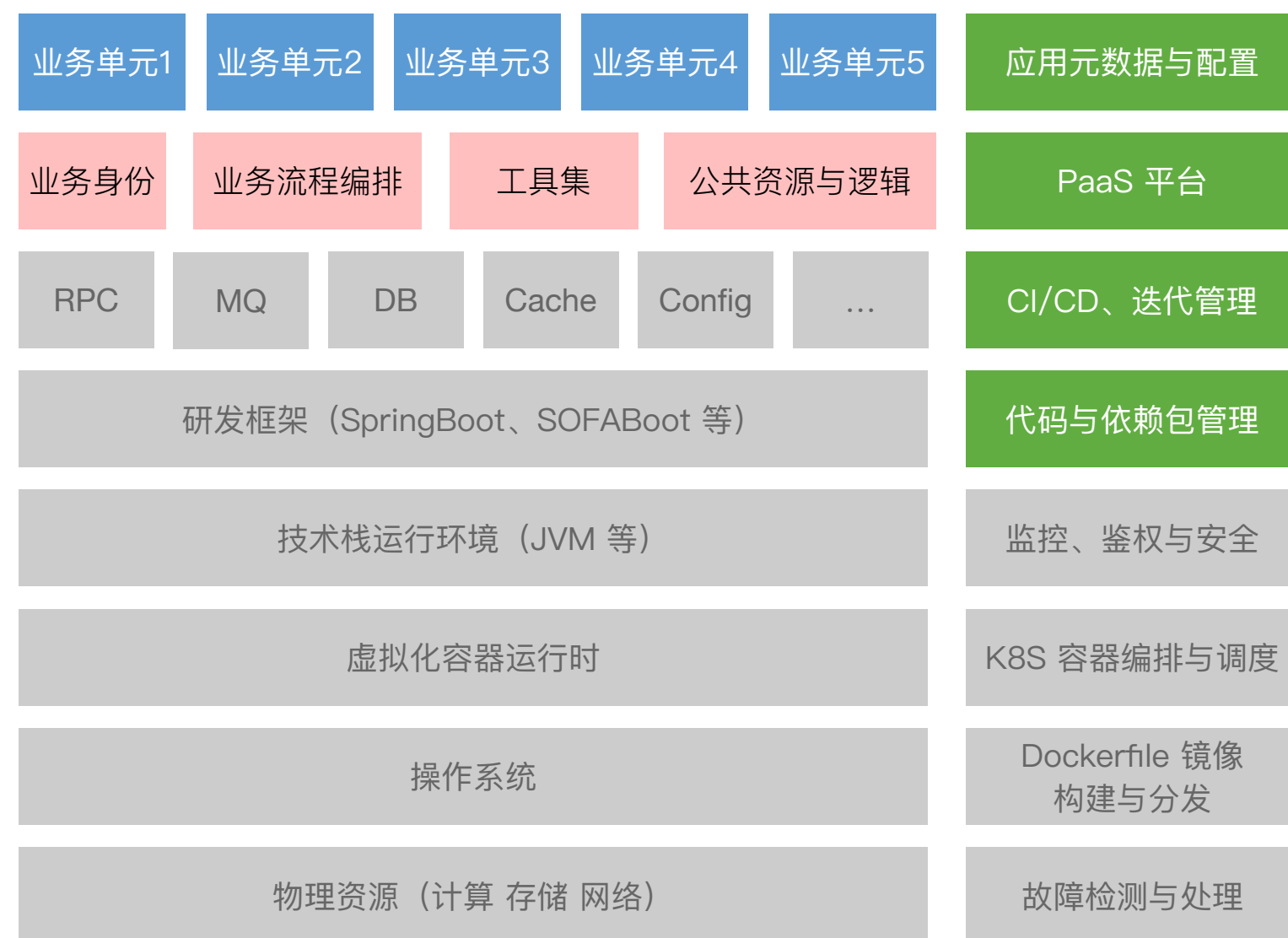
效率低：认知负荷高，运维负担重

- 业务开发者需要感知复杂基础设施，异常多
- 框架与中间件升级维护成本高、周期长
- 部署上线慢，特别是大应用（构建启动慢、机器多）



异常多，业务同学心声：不出问题还好，一出问题，就相当于新学一门技术

正在尝试的解法



- 服务网格
- 应用运行时
- 平台工程

业务还想要的



• 业务公共部分也是基础设施

蚂蚁微服务研发痛点- 协作与资源成本高

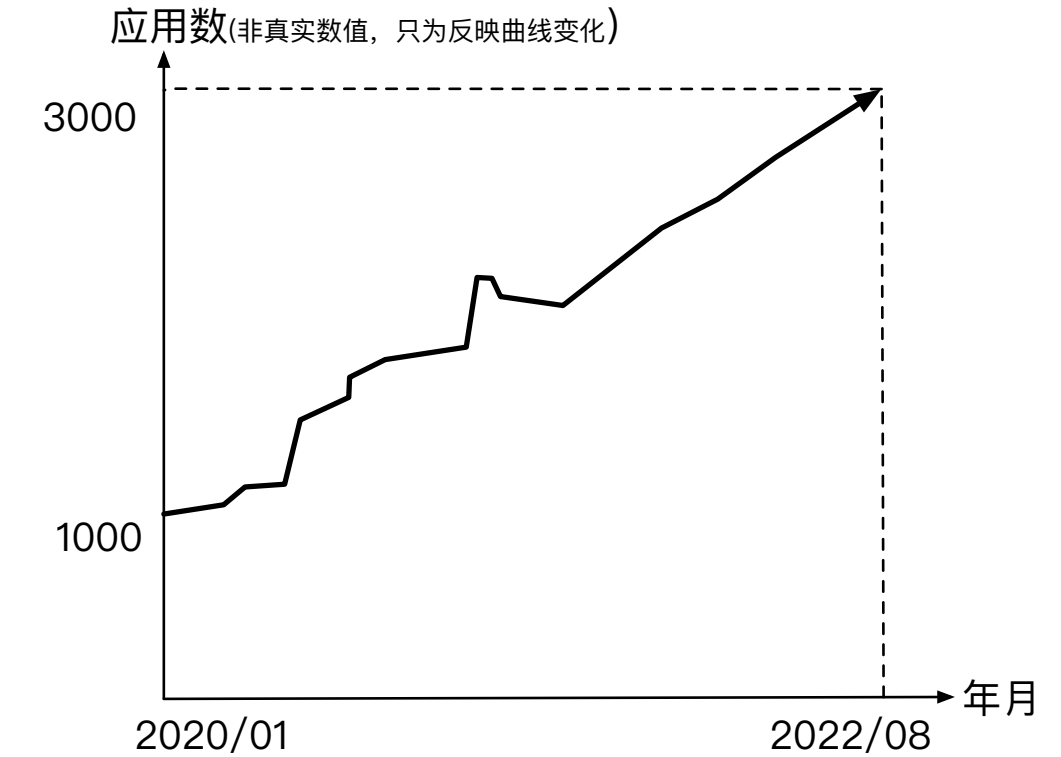
多人协作阻塞，变更影响面大，风险高

- 大应用过大



资源成本与长期维护成本高

- 小应用过多



问题必然性

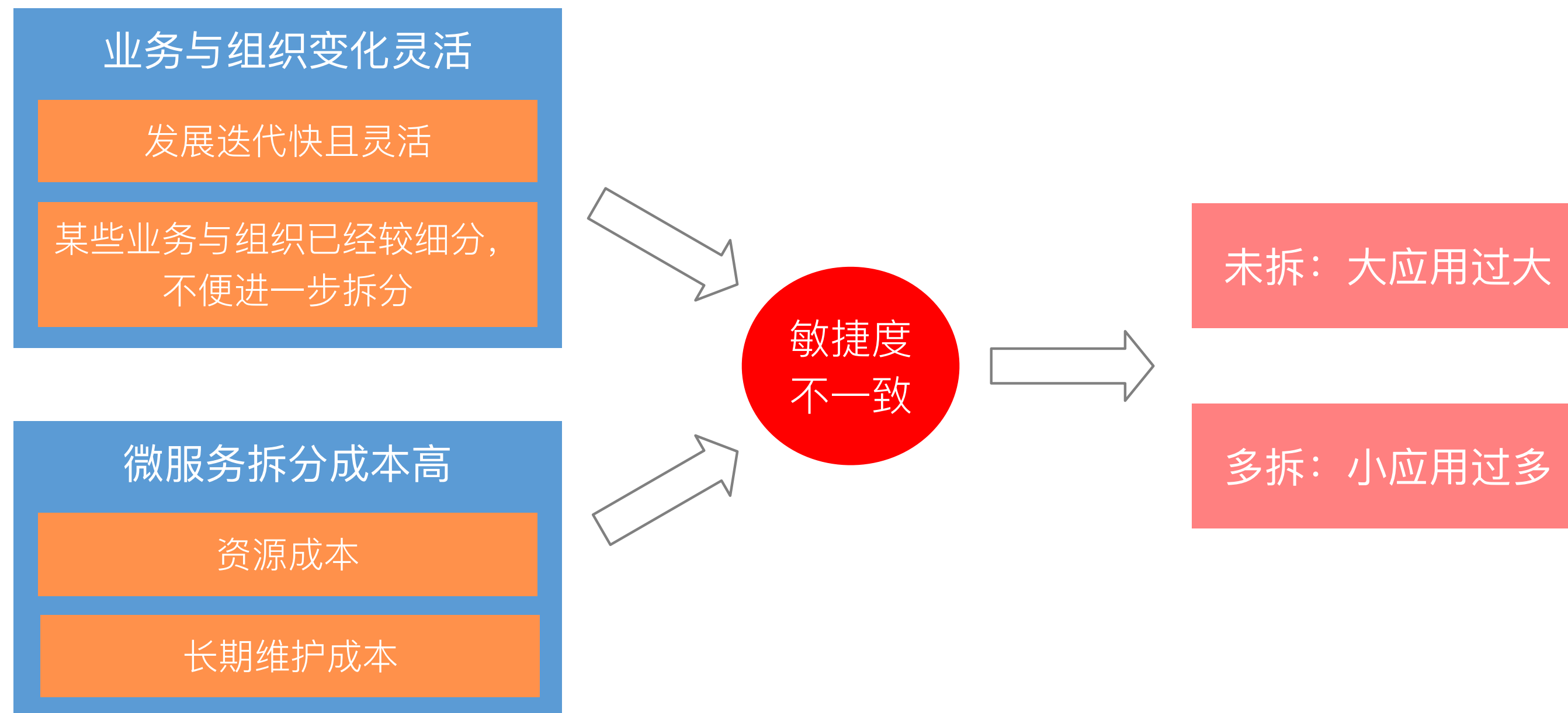
- 微服务系统也是生态，参考28定律，极少数的大应用占有大量的流量

大应用多大算大？

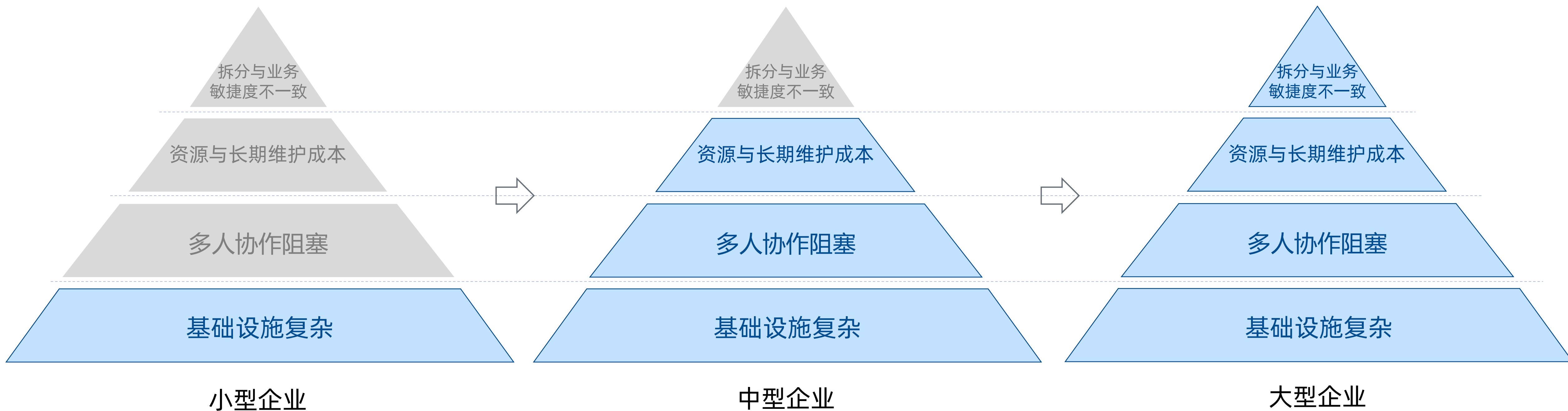
小应用多少个算多？

如何合理拆分微服务是个老大难的问题

- 微服务的拆分与业务和组织发展敏捷度不一致



行业微服务现状



现有尝试解法

- 服务网格、应用运行时 dapr、layotto, 平台工程
- Spring modulith, Google ServiceWeaver

现有尝试解法的问题

- 从业务角度, 只屏蔽部分基础设施, 未屏蔽业务公共部分
- 存量应用接入改造成本高
- 初期发展中, 解决其中部分问题

1 蚂蚁微服务背景

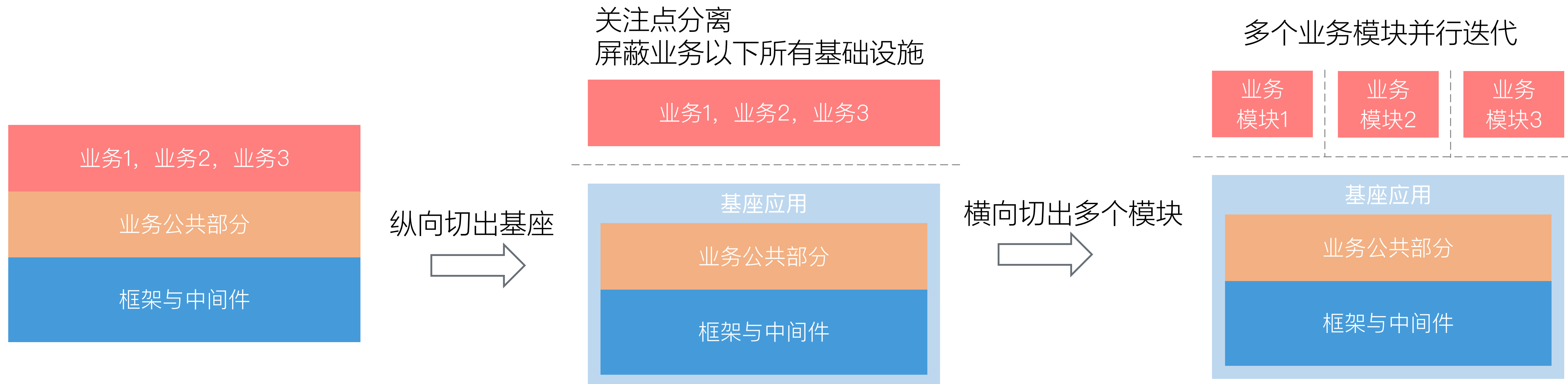
2 SOFAServerless 的解法与效果

3 SOFAServerless 的运维调度平台介绍

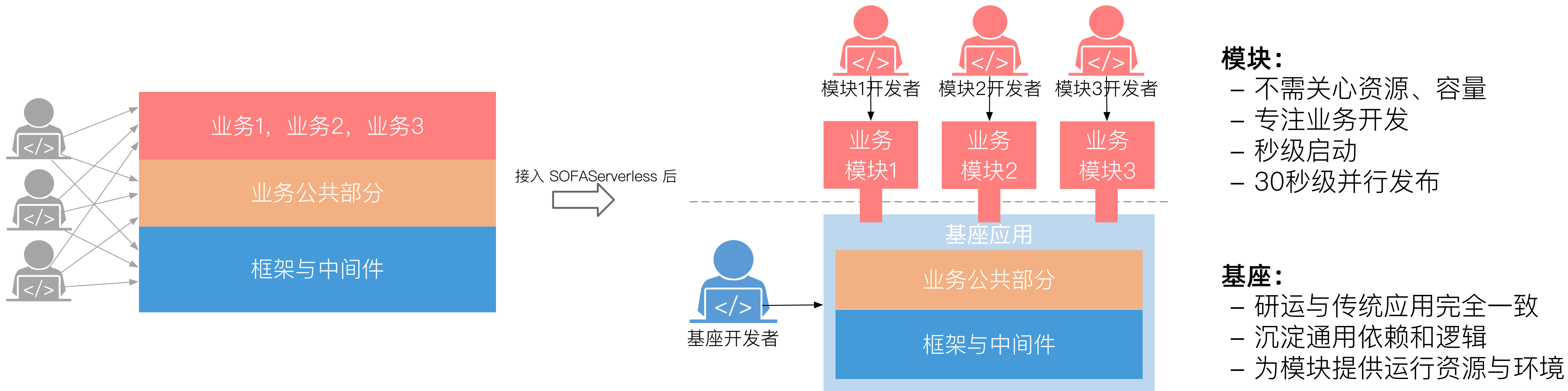
4 蚂蚁的实践经验与案例

5 总结与展望

SOFA Serverless 的解法



SOFA Serverless 的解法



对应用同时进行纵向和横向拆分，纵向拆分成基座和模块，横向拆分成多个模块，让业务专注于业务本身

那么如何拆分的呢？

多模块间的隔离与共享

模块是什么

- SpringBoot 打包生成的 jar
- 一个 ClassLoader + 一个 SpringContext
- 热部署（不重启机器，可选）



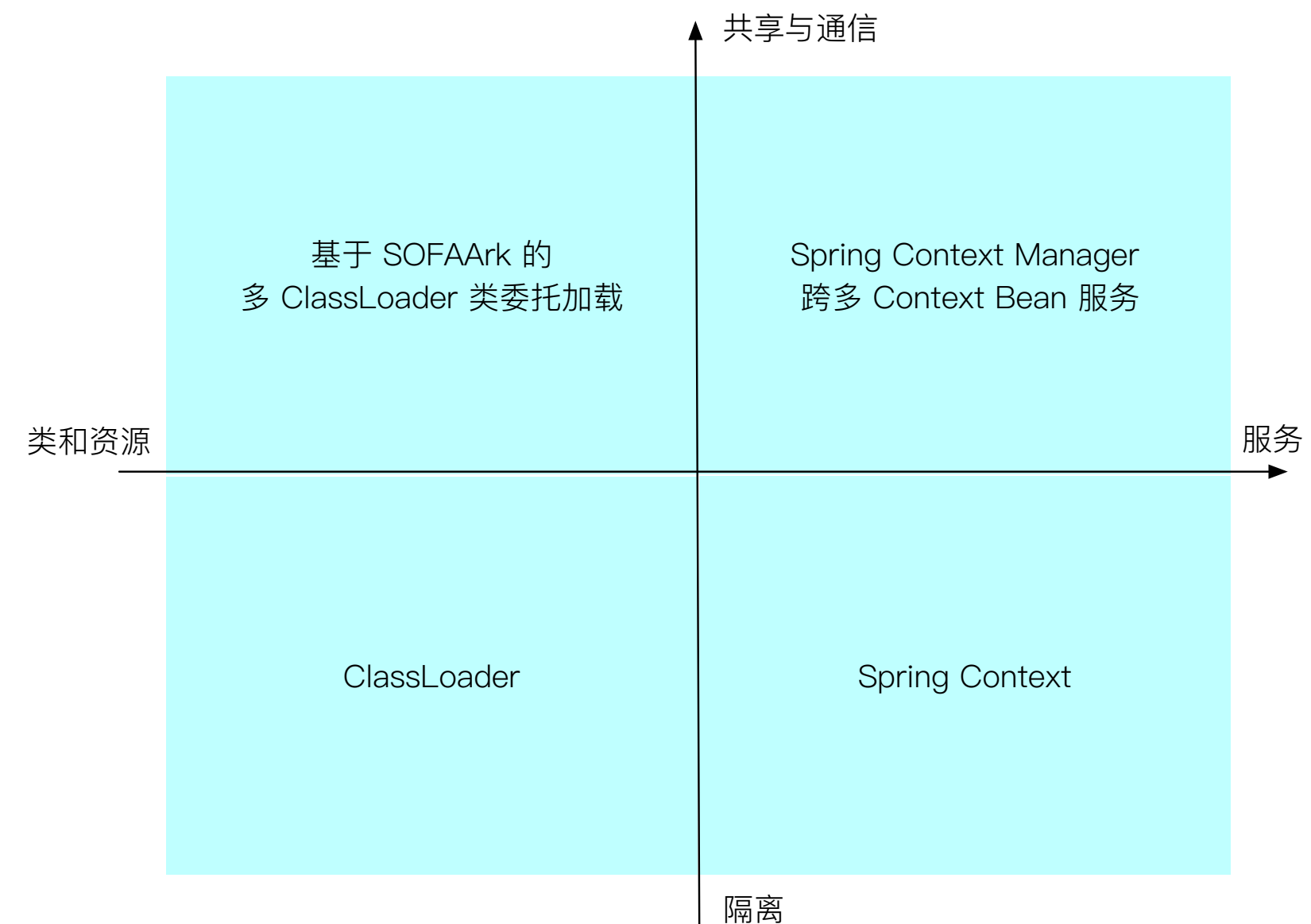
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter</artifactId>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  <scope>provided</scope>
</dependency>
```

模块间隔离

- SpringBoot 隔离 Bean
- ClassLoader 隔离配置和代码

模块间共享

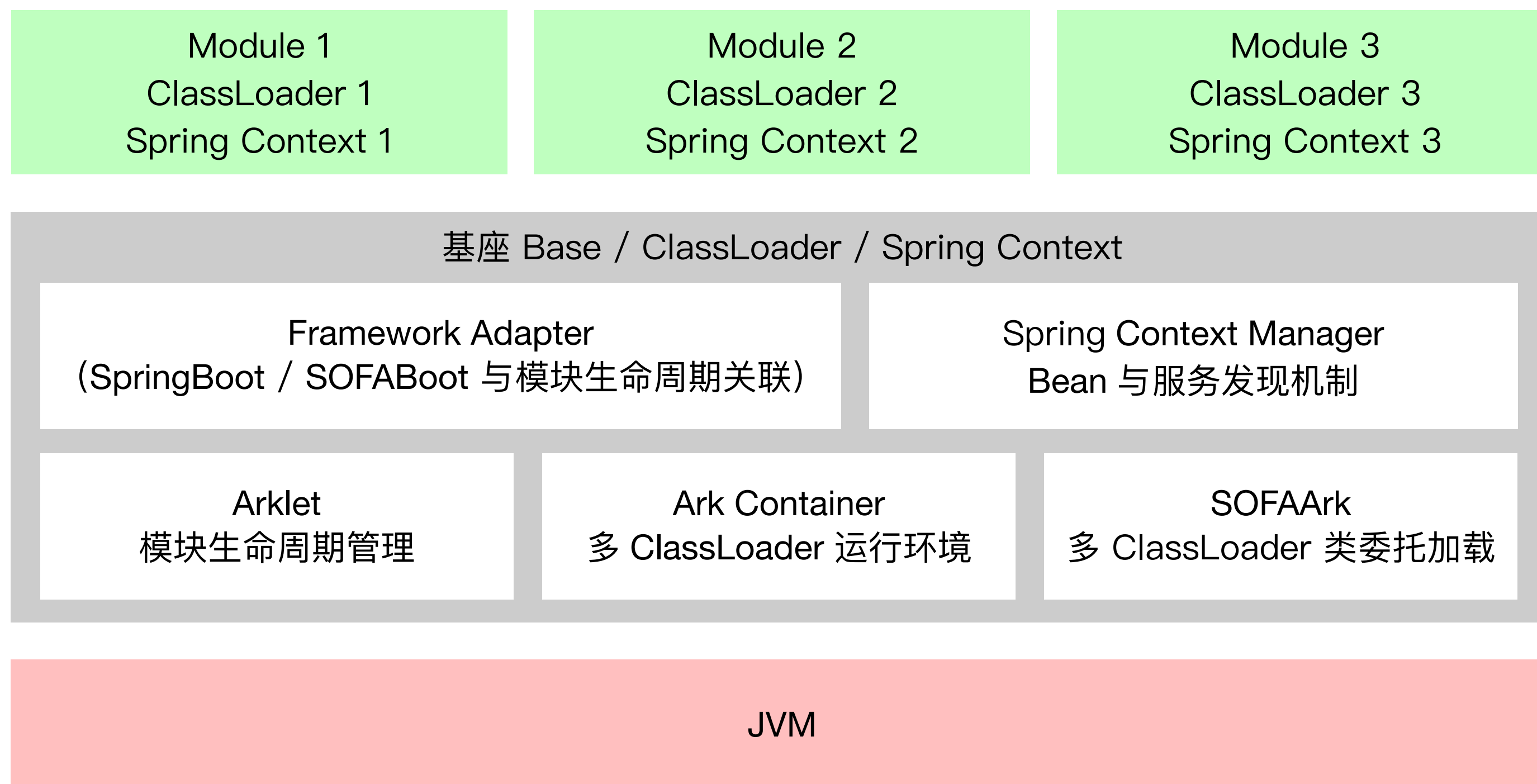
- SpringContext Manager
- 模块默认复用基座类



多模块间的隔离与共享

多模块的运行环境

- 模块运维与生命周期管理：Arklet
- 模块的运行容器：ArkContainer
- 框架适配器
- 多模块的隔离与共享机制
 - 类委托加载：SOFAArk
 - Bean 与服务的发现



多模块间的隔离与共享

	隔离		共享	
	类与资源	bean与服务	类与资源	bean与服务
OSGI	ClassLoader	SpringBoot	定义模块间依赖关系，源模块与目标模块定义导入导出列表	定义模块间依赖关系，源模块与目标模块定义导入导出列表
JPMS	JVM 内置模块级别访问控制	JVM 内置模块级别访问控制	定义模块间依赖关系，源模块与目标模块定义导入导出列表	定义模块间依赖关系，源模块与目标模块定义导入导出列表
Spring Modulith	inner 的子 package, inner 内不允许外部模块访问, 并通过 ArchUnit 的 verify 来做校验, 未通过则构建失败。	inner 包, 通过单元测试 verify 来校验	模块api	模块api与事件
SOFAArk	ClassLoader	SpringBoot	源模块默认委托给基座 (pom 依赖 scope 设置成 provided)	默认打通跨模块bean与服务发现

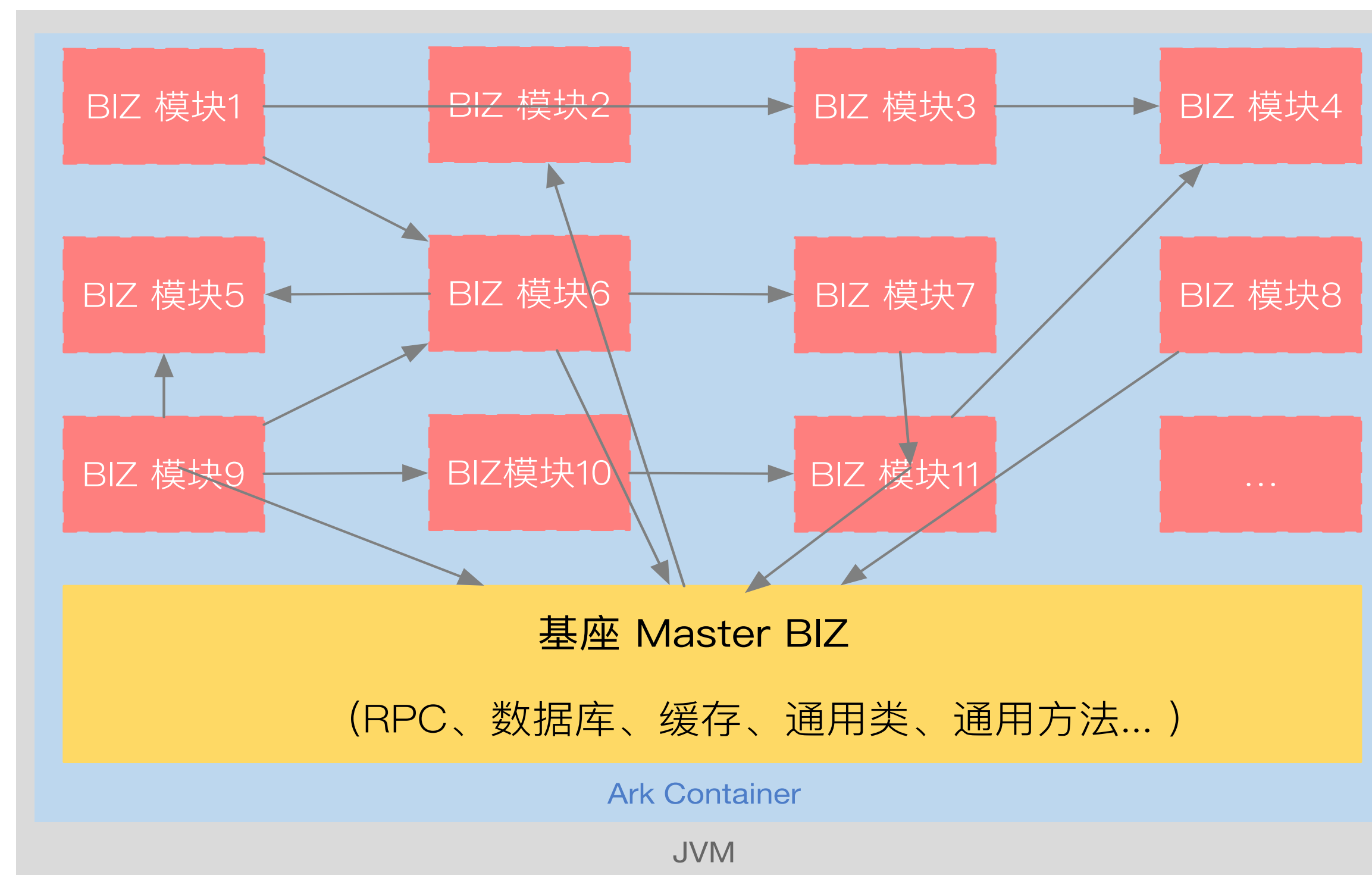
原生使用方式，低成本接入

- 模块也是原生 SpringBoot 或 SOFABoot
- 默认的共享机制，业务代码低侵入
- 同一套代码既可以独立部署，也可以作为模块合并部署

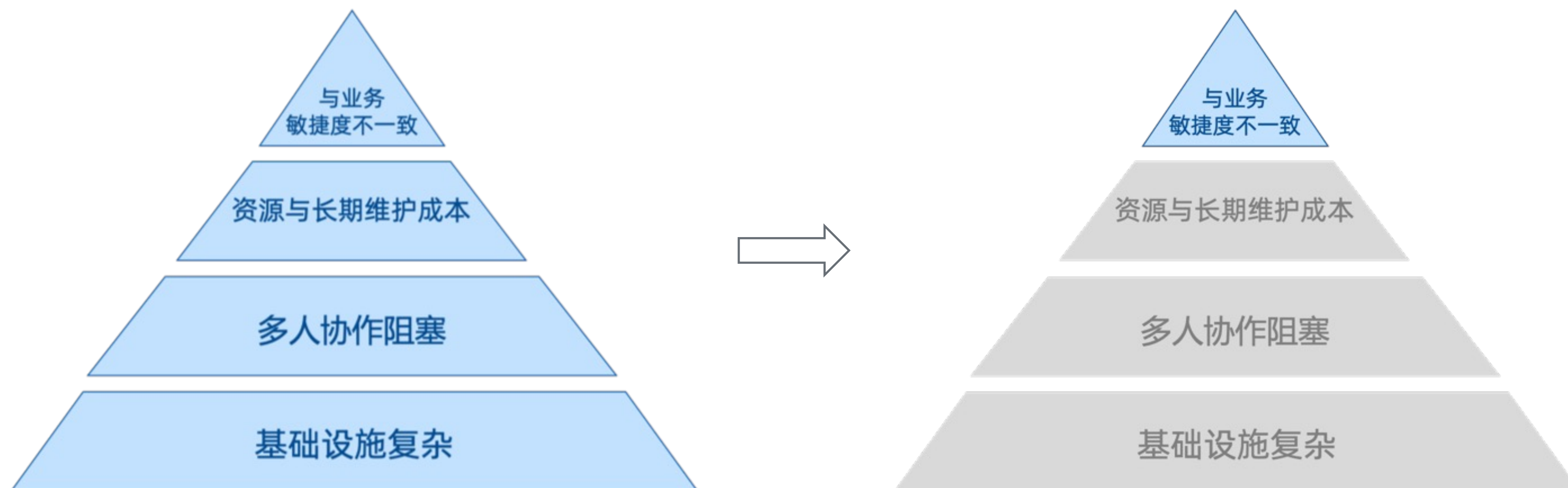
多模块间的隔离与共享

多模块间通信

- 基座与模块通信无序列化开销
- 模块与模块通信有序列化开销



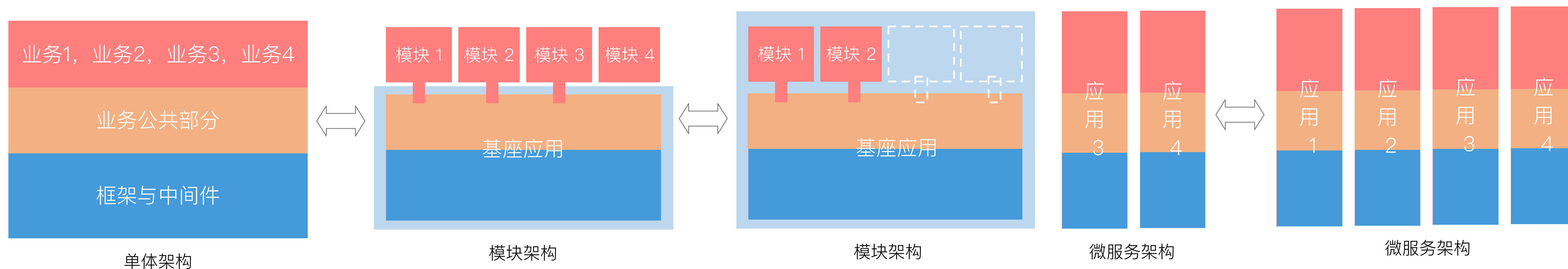
SOFA Serverless 的解法



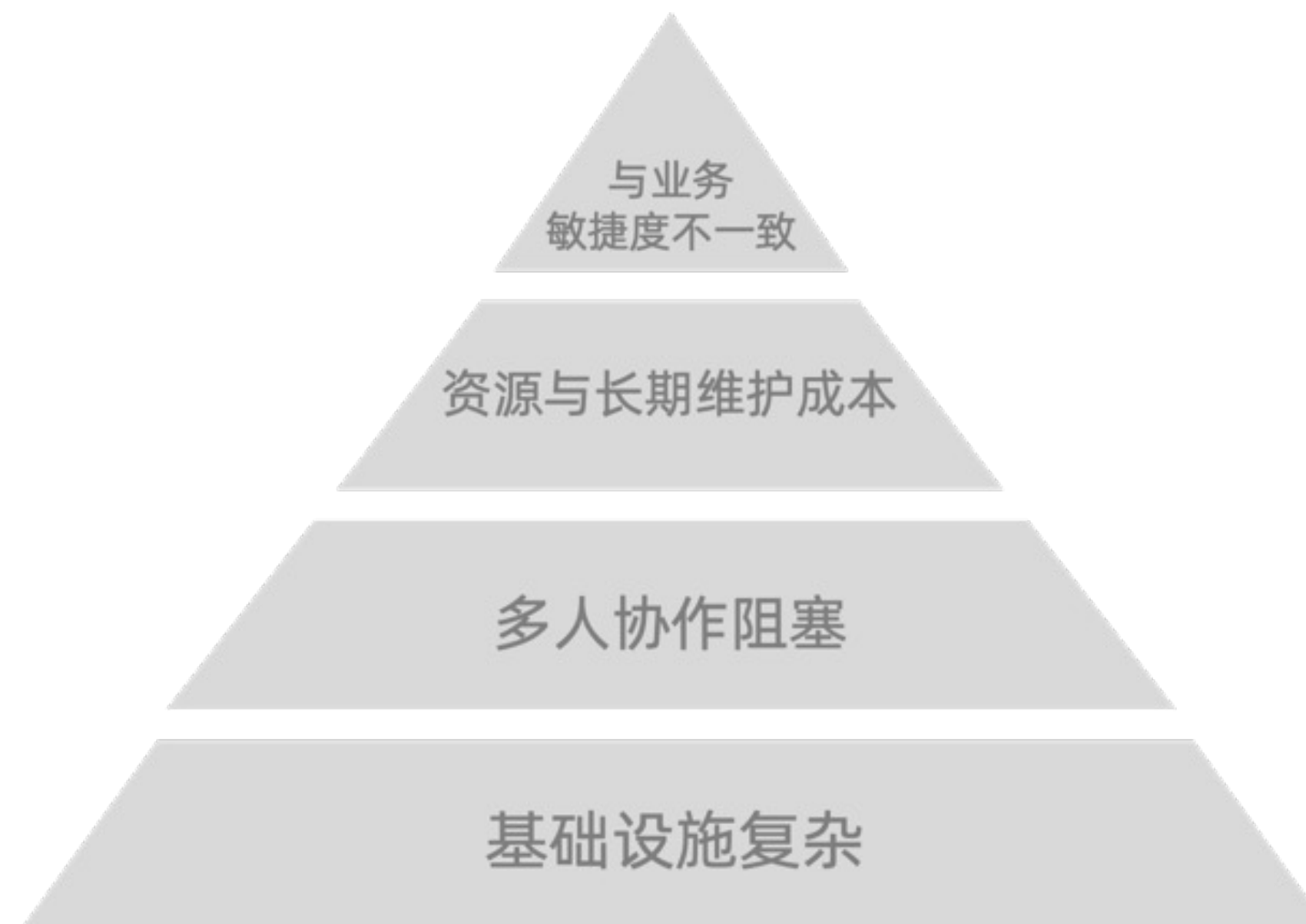
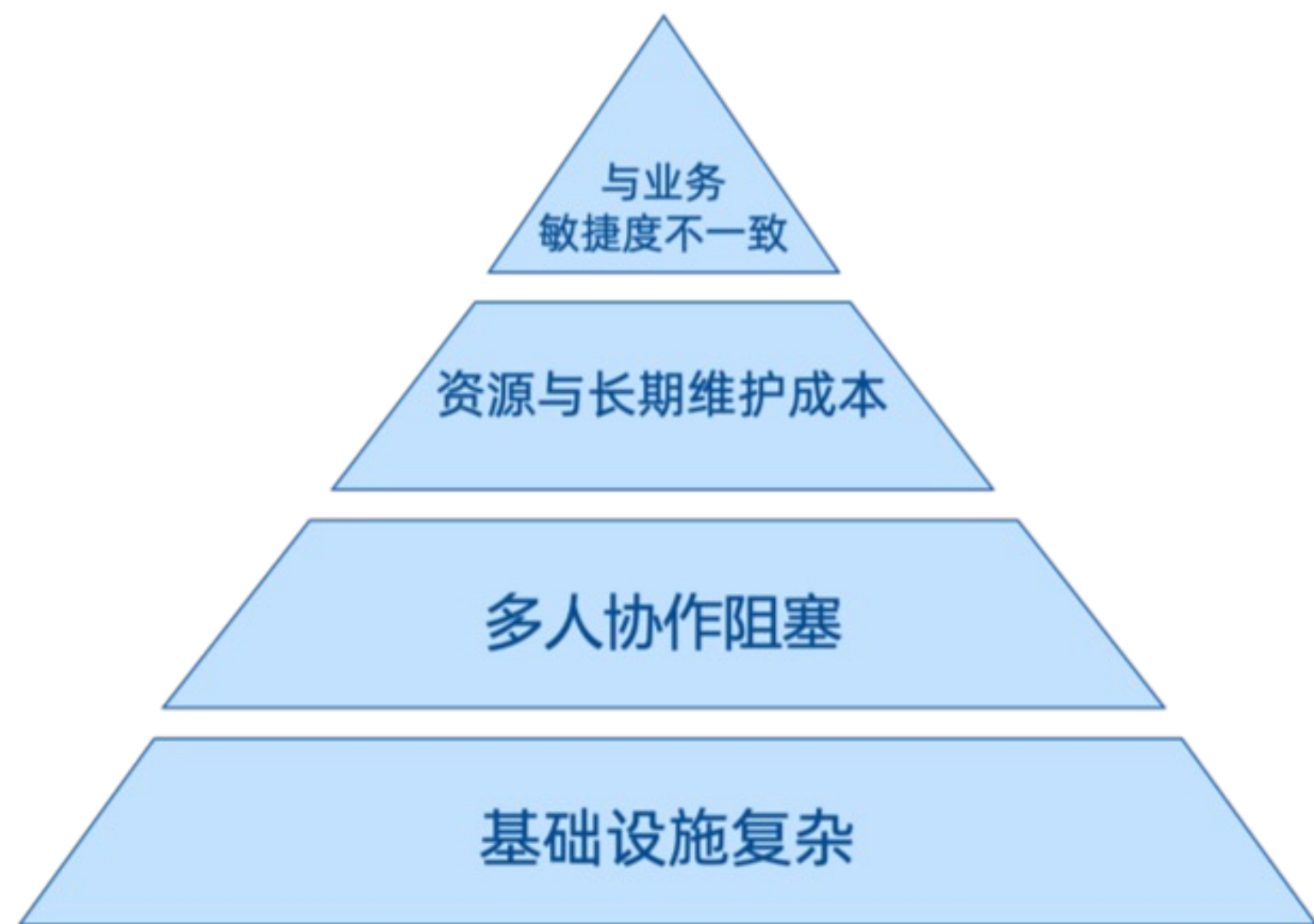
降低微服拆分成本

资源成本

- 单体架构与微服务架构之间增加模块架构
- 单体应用低成本拆分成模块应用
- 模块应用可低成本演进成微服务，也可回退回单体应用



SOFA Serverless 的解法



SOFA Serverless 的效果

- 只感知业务本身，低认知负载，秒级启动
- 并行迭代无阻塞

- 模块可独立部署成进程，也可合并部署成模块
- 部署粒度与变更风险面小，只涉及模块代码和对应机器



- 模块不占额外机器，只占极少业务自身的cpu和内存
- 弹性调度粒度小，资源密度高

- 存量应用可低成本改造成或拆分成模块
- 模块可低成本演进成微服务也可回退回单体

	构建耗时	构建产物大小	启动类个数	内存消耗	部署耗时（包括镜像拉取与切挂流）
普通应用模式	265 秒	1, 385 MB	10, 530 个	337 MB	141 秒
模块应用模式	27秒	0.02 MB	512 个	17 MB	4秒
对比效果	1/10	1/7万	1/20	1/20	1/35

1 蚂蚁微服务背景

2 SOFAServerless 的解法与效果

3 SOFAServerless 的运维与调度平台介绍

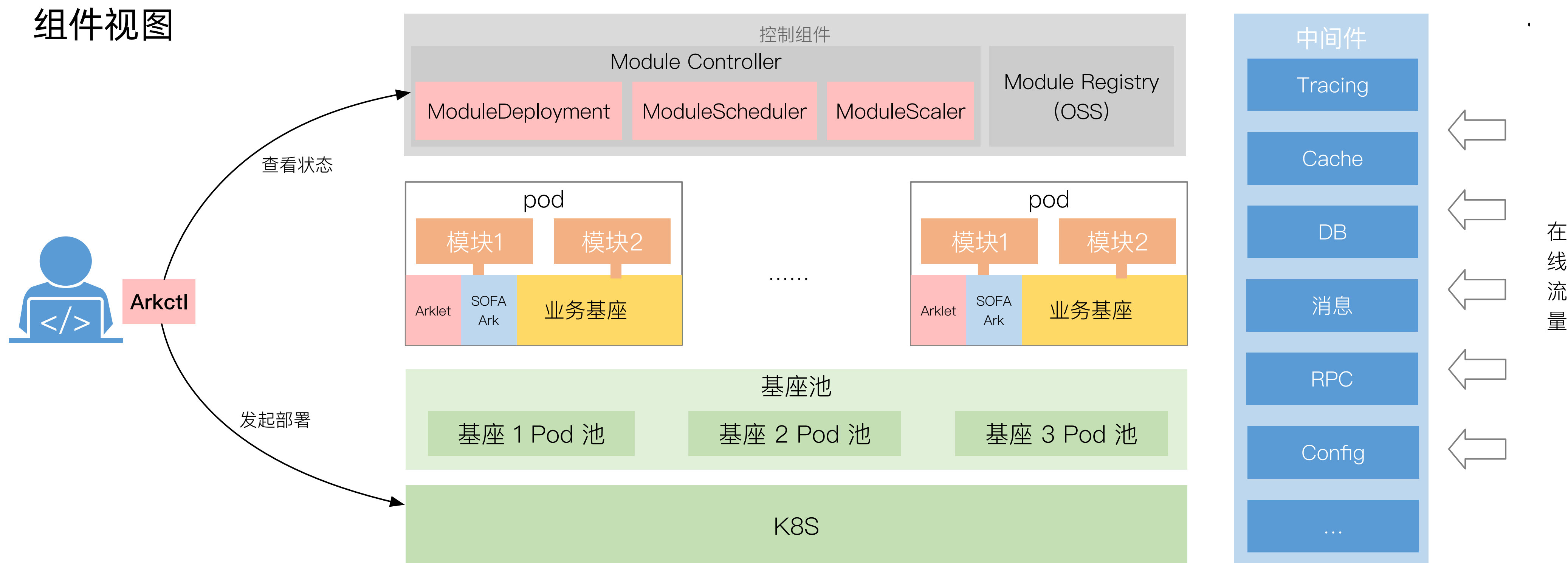
4 蚂蚁的实践经验与案例

5 总结与展望

研发框架与平台



研发框架与平台



SOFAArk

- 类隔离与热部署的多模块框架

Arklet

- 运维管道，模块生命周期管理，多模块运行环境

Arkctl

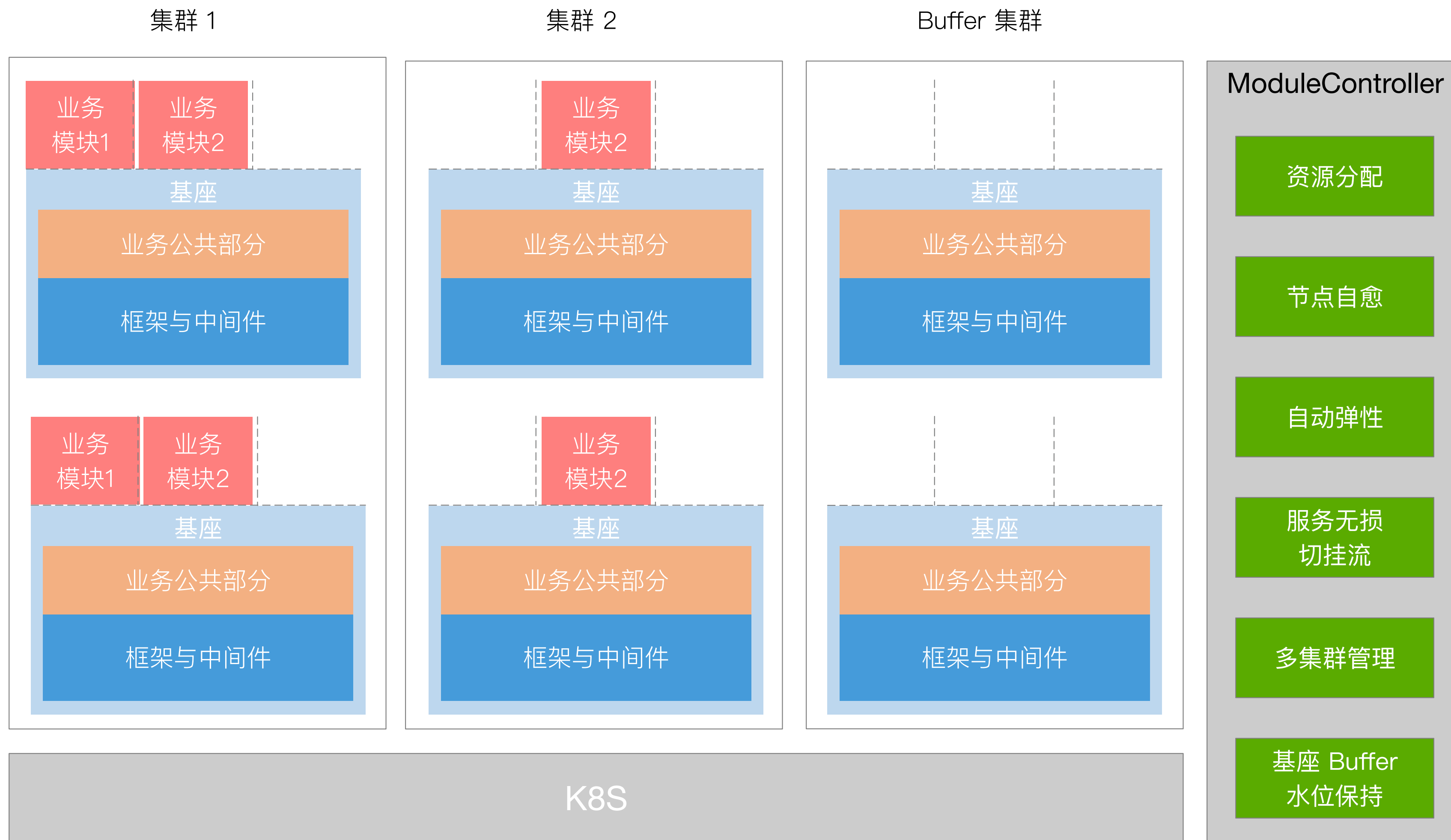
- 研发工具，模块初始化，快速部署验证

ModuleController

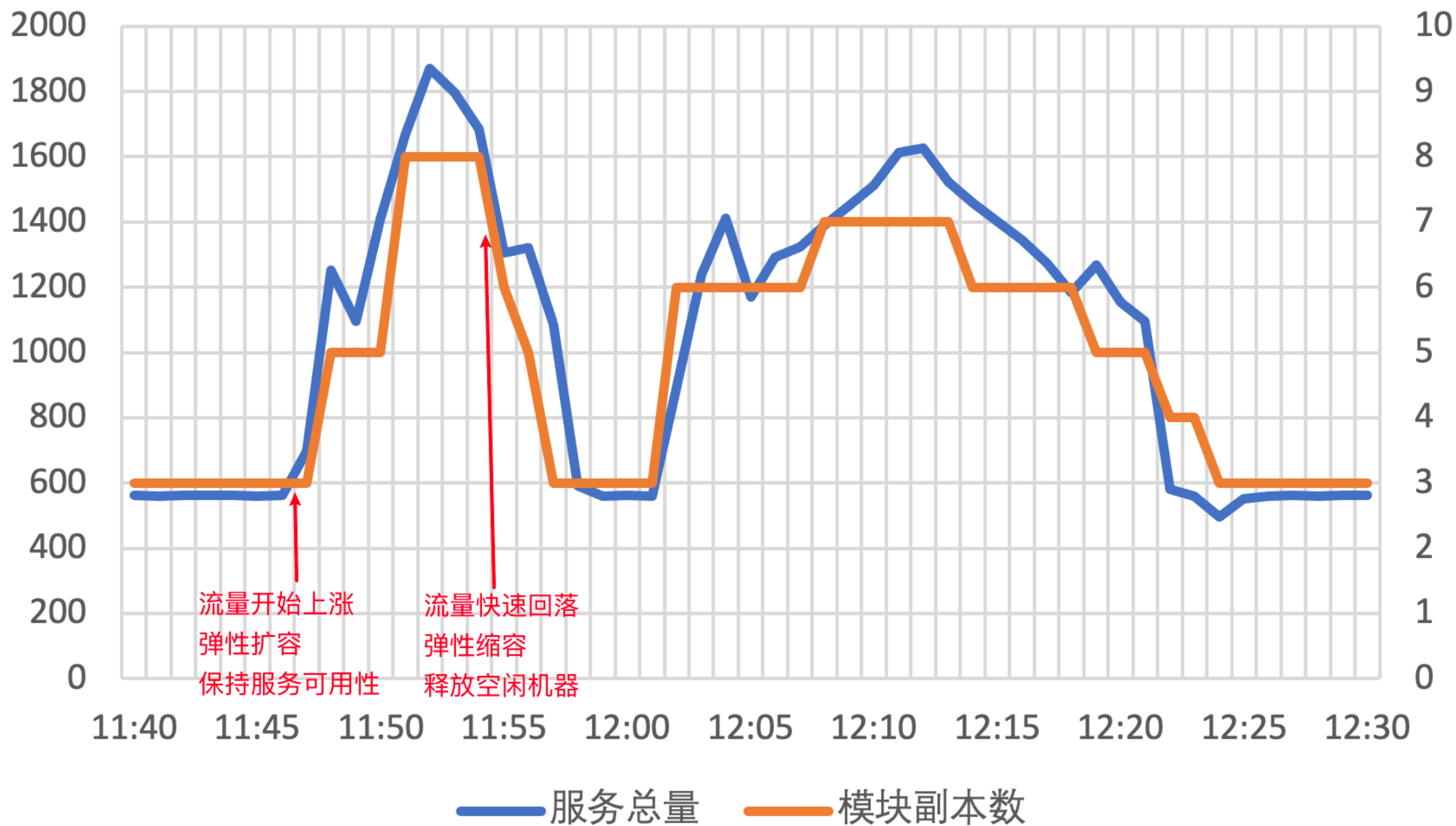
- **ModuleDeployment**
 - 模块发布运维
- **ModuleScheduler**
 - 模块调度
- **ModuleScaler**
 - 模块伸缩

研发框架与平台 - 弹性调度

调度与伸缩



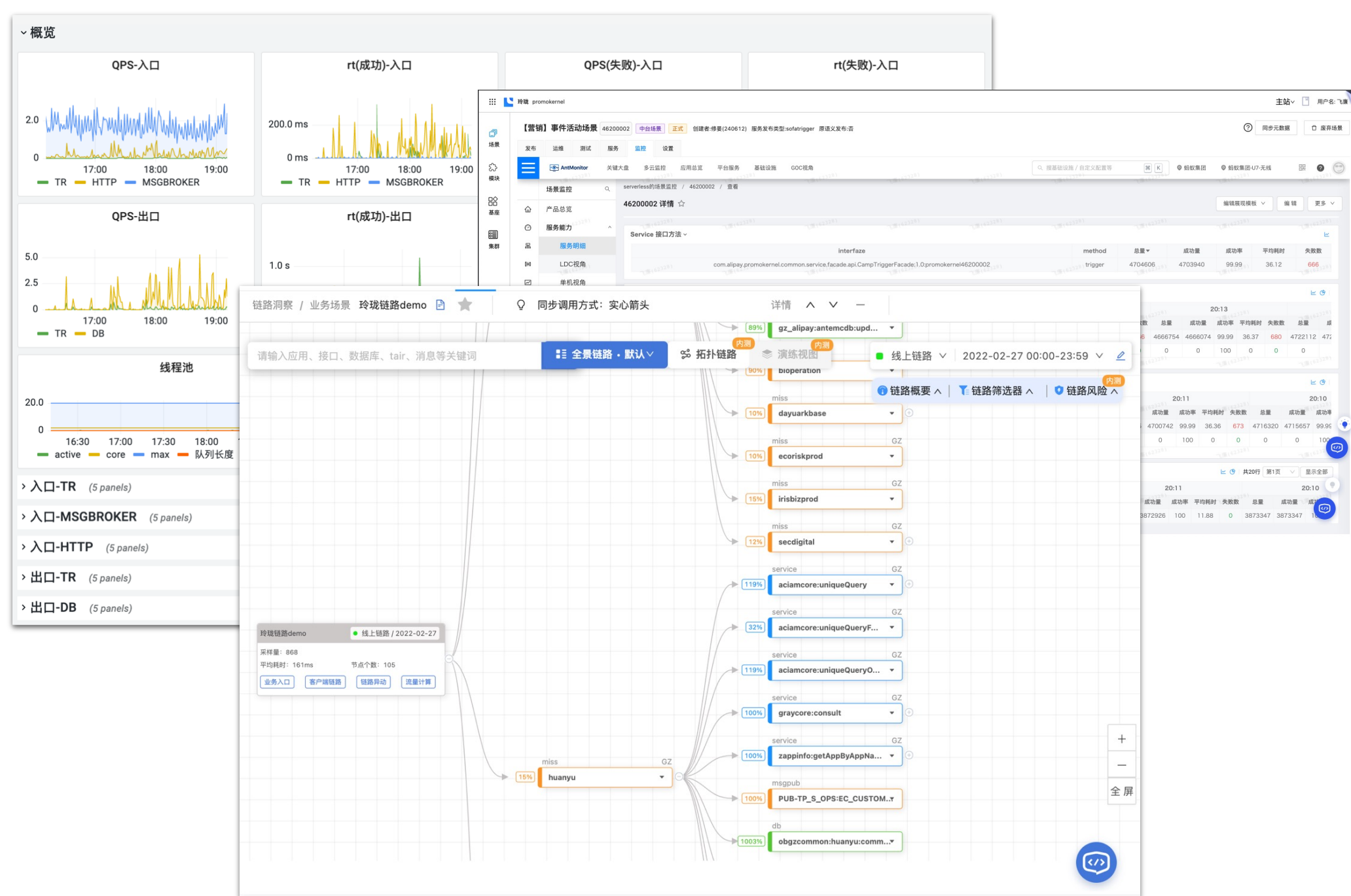
调度与伸缩



研发框架与平台 - 可靠性

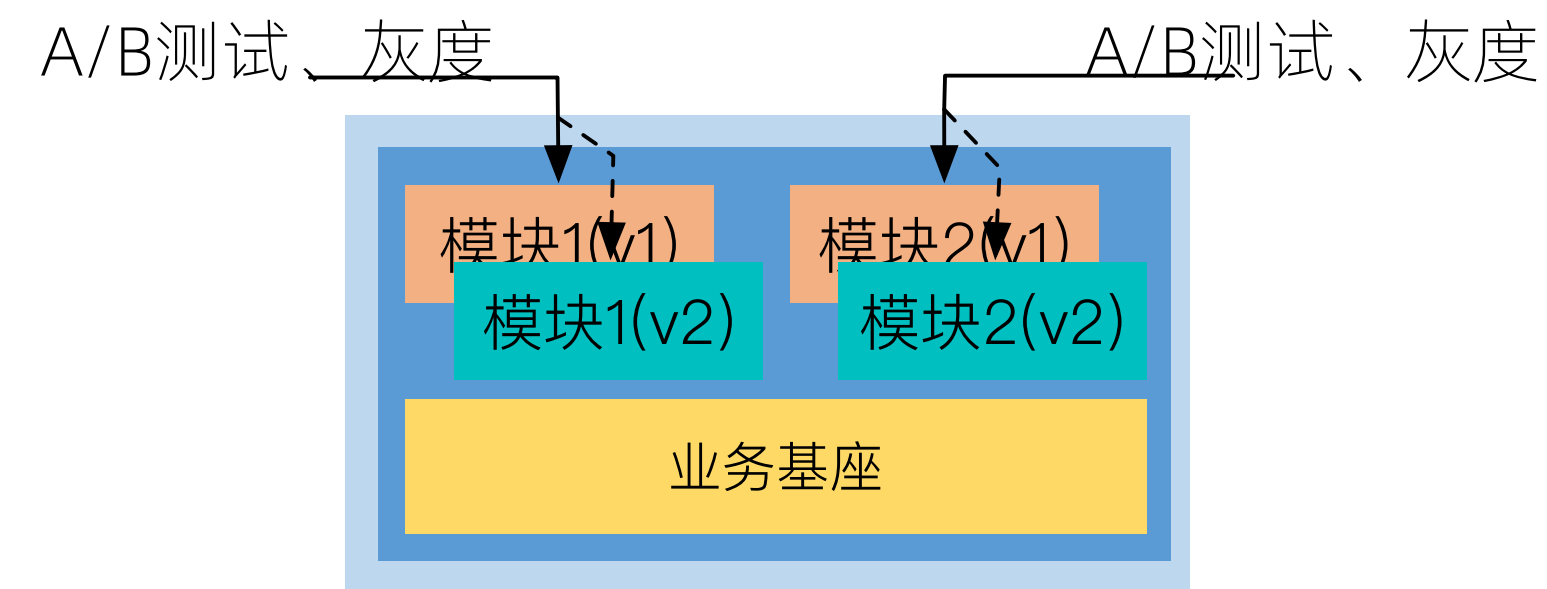
可观测性

- 模块粒度健康检查
- 服务监控、日志监控、Trace、排障



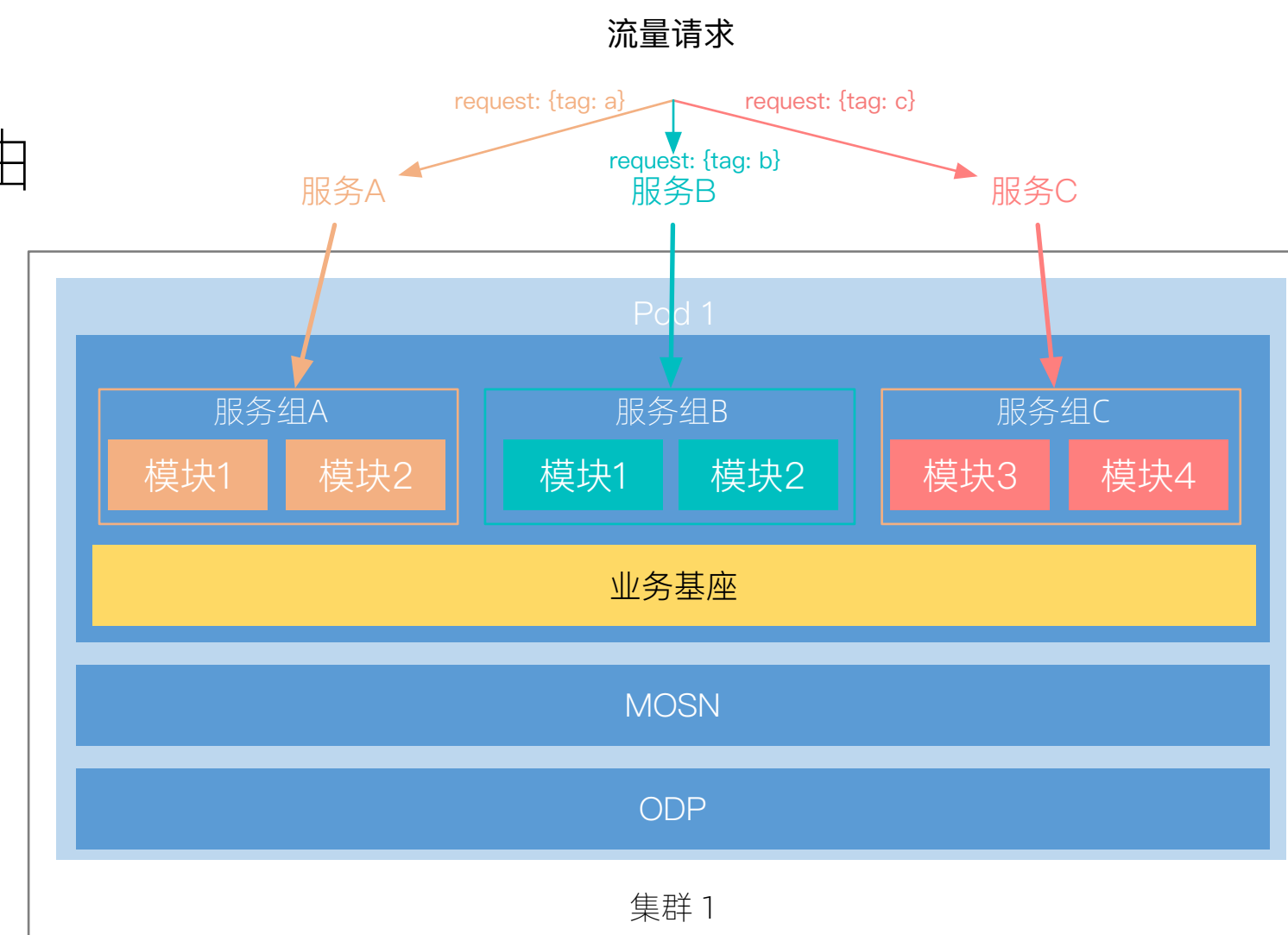
AB 测试/灰度

- 模块多版本, 快速AB测试/灰度/回滚



流量隔离

- 单集群模块间流量隔离
- 根据请求参数精细化路由

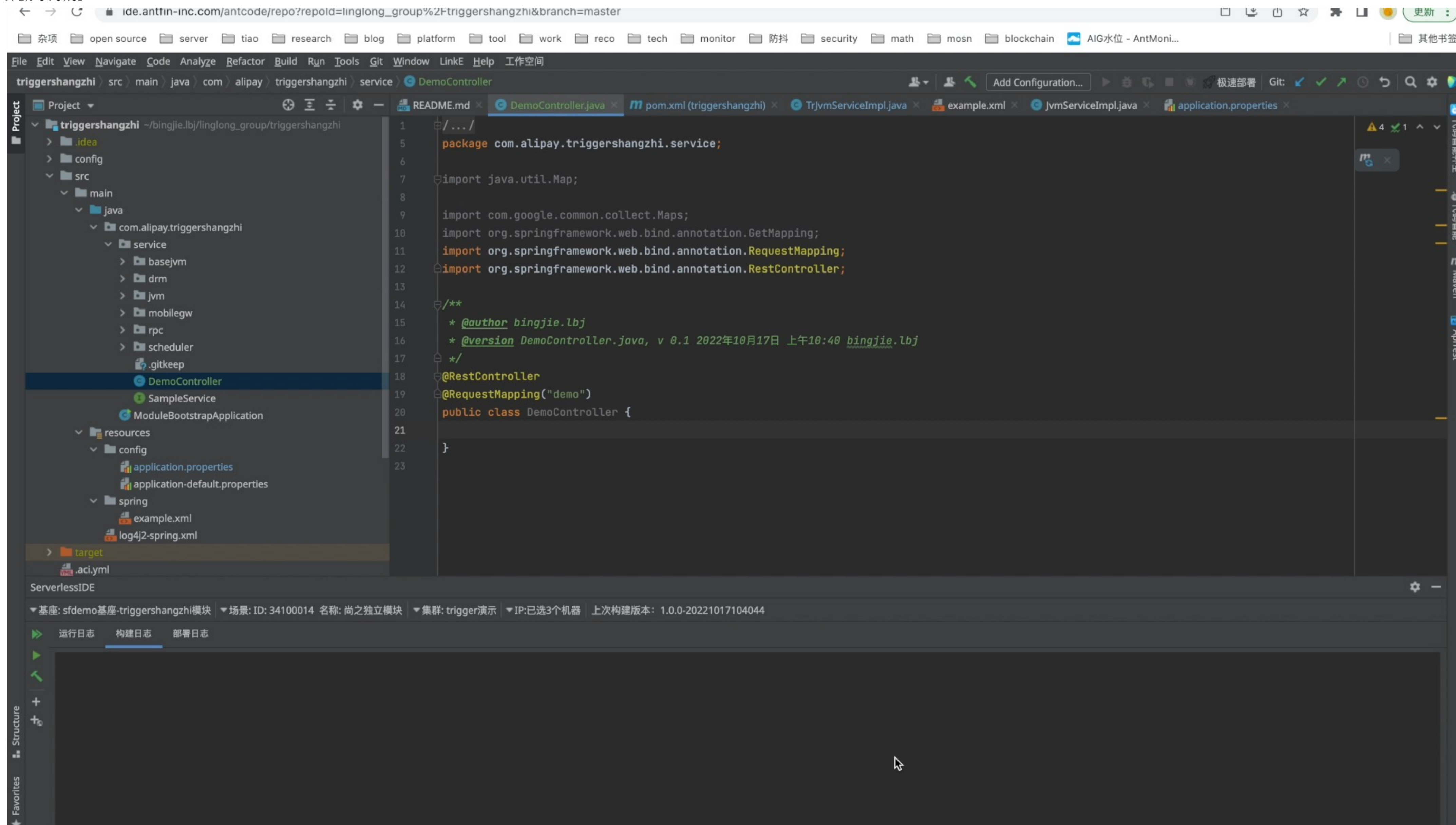


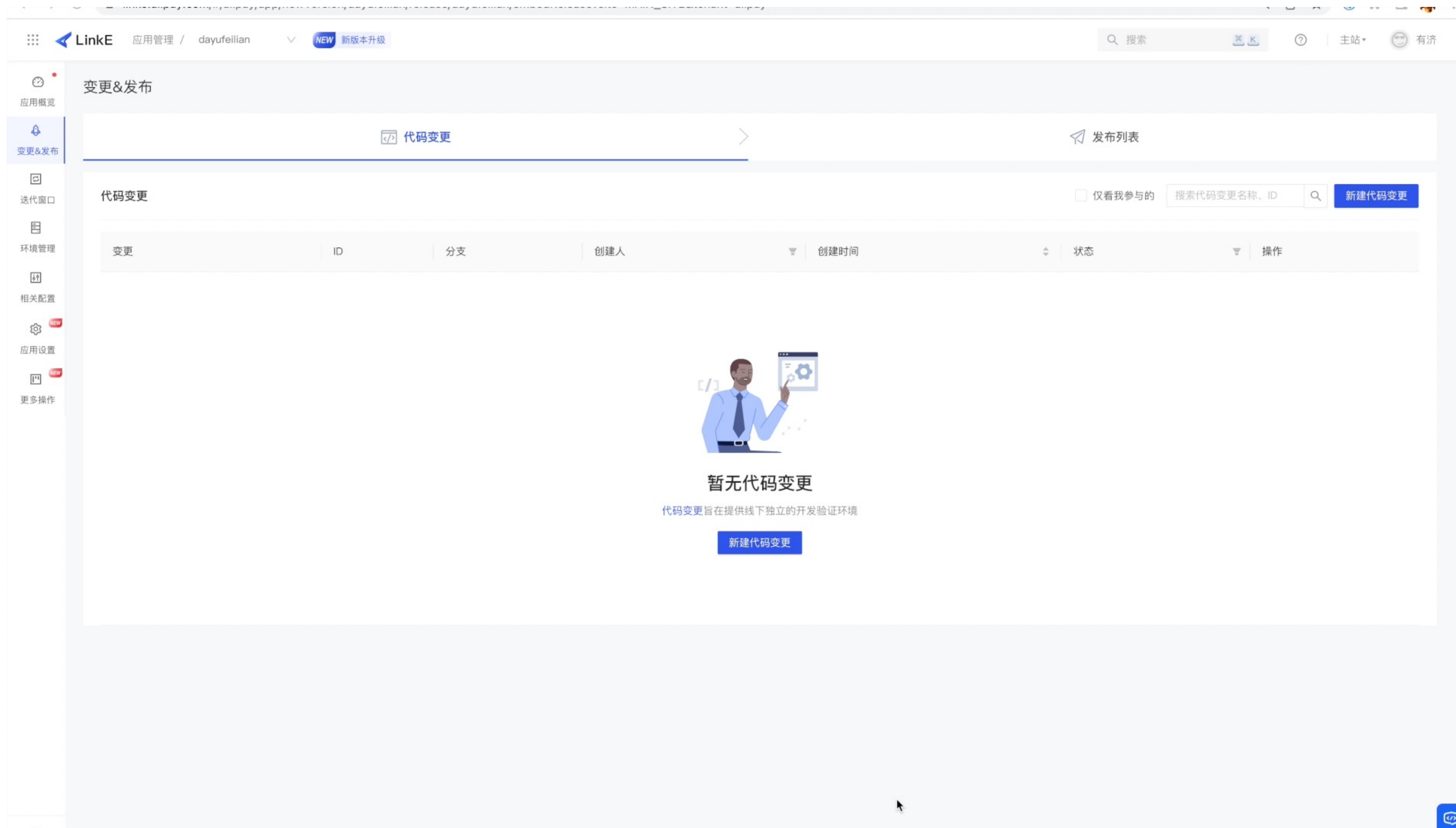
体验极速人生：模块创建

The screenshot displays the LinkE dashboard with a sidebar on the left containing navigation items like '工作台', '我的收藏', '我的工作项', '迭代', '应用', '发布窗口', '服务器', '流程中心', '架构域', '代码仓库', and '应用架构'. The main content area is divided into several sections:

- 工作台** (Dashboard): Includes tabs for '最近参与的迭代', '与我相关的迭代', '最近参与的发布 (新模式)', and '新建迭代'. It lists recent iterations with details like 'spi支持多版本', 'arkserverless 4.3.0', '测试', '迁移九州', and '迁移九州'.
- 我执行的** (My Executed): Shows a summary of tasks with counts: '合并请求 18', '审批 0', '需求 30', '缺陷 85', '任务 0'. Below is a list of tasks with status (e.g., '待评审', '有冲突') and details.
- 最近触发的流水线** (Recently Triggered Pipelines): A section that currently shows '你还没有触发的流水线' (You haven't triggered any pipelines).
- 常用功能** (Common Functions): A grid of utility buttons including '服务自助', '代码搜索', '代码权限申请', '联调环境', '服务器操作记录', and '链路-链路分析'.
- 公告** (Announcements): A notice regarding the greyed-out interface for the LinkE dashboard/iteration details page.
- 为你推荐** (Recommended for you): A section for 'LinkE推送' (LinkE Push) and '本周热点' (This Week's Hotspots) with article titles and view counts.

体验极速人生：模块开发





1 蚂蚁微服务背景

2 SOFAServerless 的解法与效果

3 SOFAServerless 的运维与调度平台介绍

4 蚂蚁的实践经验与案例

5 总结与展望

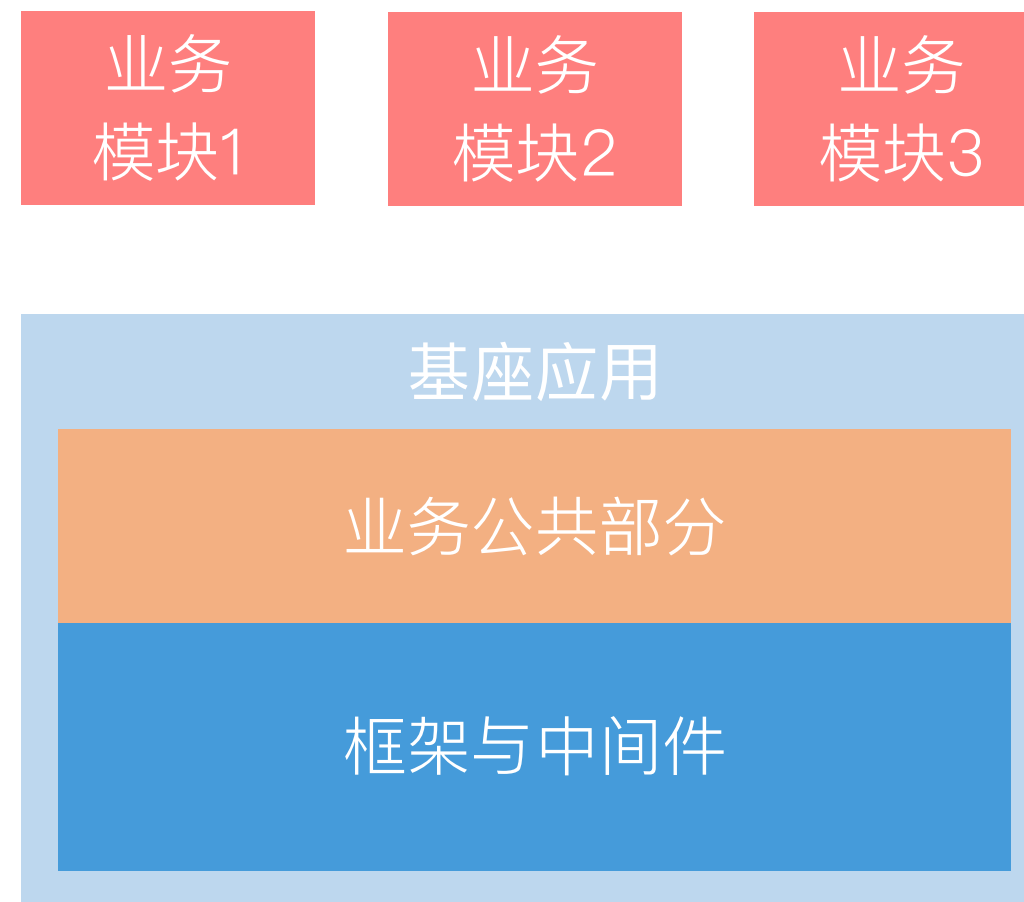
该模式存在的挑战

单进程里多应用

- 中间件与二三方包 static 变量导致多应用间互相干扰

模块应用动态卸载

- 中间件与二三方包未考虑动态卸载，存在部分资源未清理



问题的发现

- 兼容性评测与报告
- 运行异常自动诊断



问题的治理

- 中间件与二三方包治理
- 事件扩展机制，允许自定义清理逻辑
- 指定版本路由，流量只进入最新版本



问题的防御

- 代码扫描与准入
- 最佳实践

案例 — 合并部署



价值： 蚂蚁长尾应用多且每个机房至少要部署 2 台机器，CPU 使用率仅 10%。使用合并部署多个应用可以合并到同一个基座上，基座由各业务域专人负责维护，从而极大降低了开发者的运维成本和资源成本。



收益：

- 1、极致的**资源成本降低**，蚂蚁数管BU将 100 个传统应用合并到 10 个基座，每年节省 500 万成本。
- 2、小应用可以不走很重的应用申请上线流程，**不需要申请机器**，直接部署到通用基座，帮助业务快速创新。

案例 — 通用基座



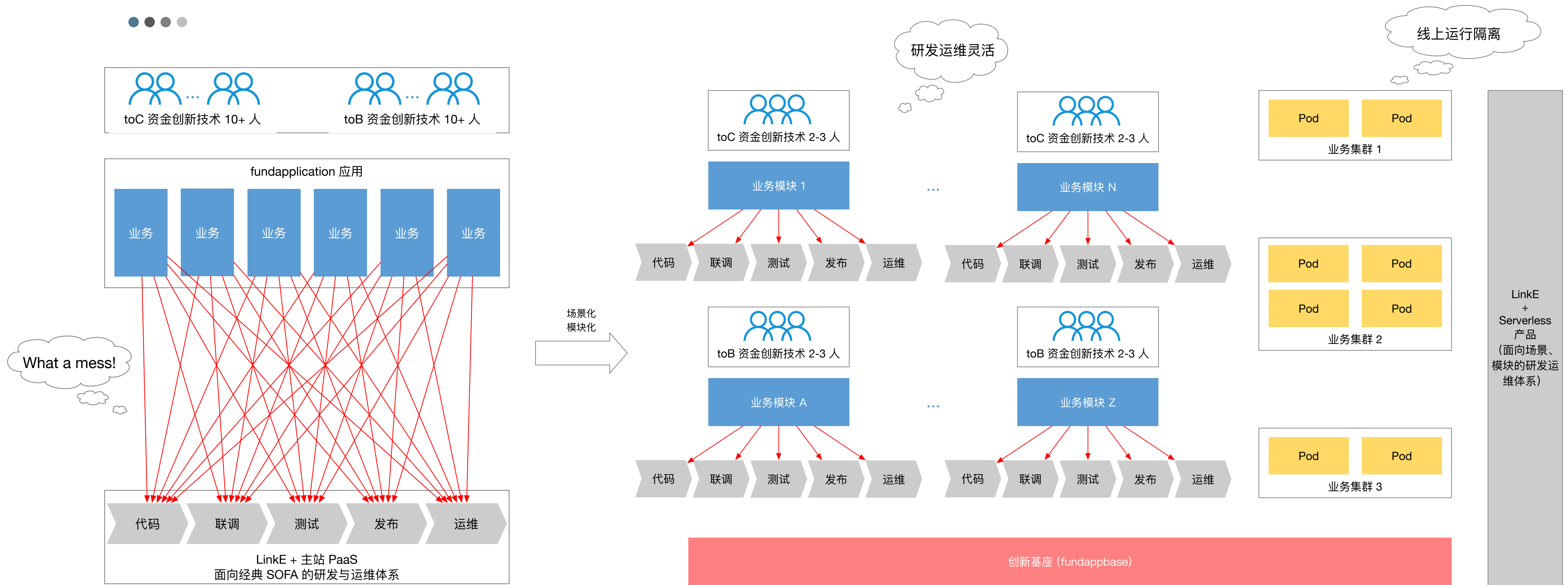
价值：在蚂蚁集团，各种 SDK 的升级打扰、构建发布慢是痛点问题。借助 SOFAServerless 通用基座模式，蚂蚁集团帮助部分应用实现了**基础设施微感升级**，同时应用的构建与发布速度也从 600 秒减少到了 90 秒。

镜像化构建和发布运维慢、中间件、二方包、三方包升级负担重是蚂蚁在线应用的研发痛点。



业务应用会被构建成 FatJar 包，在发布业务应用新版本时，调度器会选择一台没有安装模块的空基座将 FatJar 热部署到基座，装有模块的老基座 Pod 会被异步的替换成新的空基座 Pod。

案例一 资金业务 (轻应用模式)



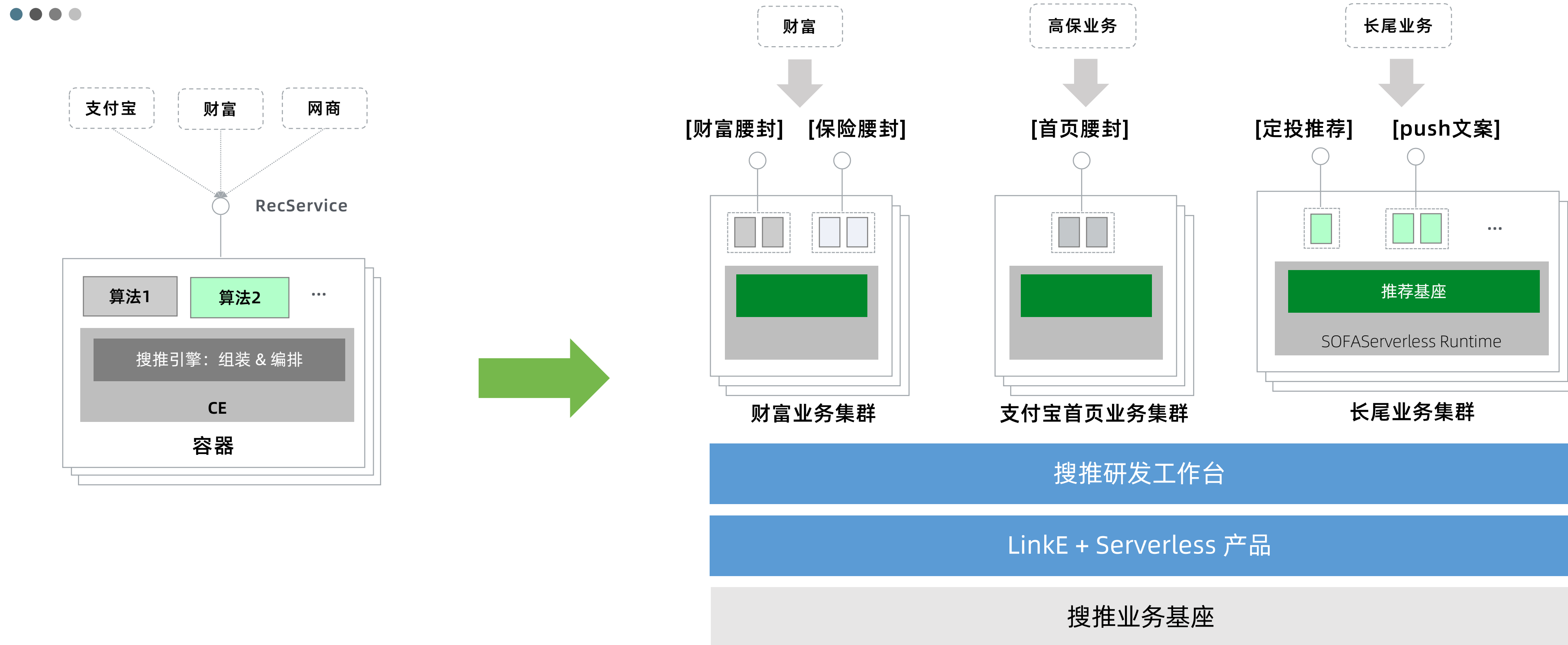
主要矛盾

创新效率低下，发布 10min+，周级别 迭代
多团队共建协作成本高，互相抢占现象严重
流量不隔离，无法支持业务高保，故障风险高

Serverless 架构红利

创新大幅提效，发布 10min => 13s，周级别 -> 1周3次 * x 迭代
多人敏捷迭代，模块独立开发运维互不影响，资源成本下降
隔离流量和资源从而实现故障隔离

案例一 搜推业务 (中台模式)



- 纯计算中台型业务，线上研发
- 期望小时级上线快速试错
- 一个大应用，线上搜推场景未隔离，互相影响

- 总计已接入 **1000+** 模块，划分了 **60+** 业务集群实现了隔离部署
- 模块开发者 **600+**，基座维护者 20 人
- 研发过程完全独立，平均上线耗时 < 1 天

SOFA Serverless 荣获 中国信息通信研究院 评选的 2022 年度云原生技术创新奖



1 蚂蚁微服务背景

2 SOFAServerless 的解法与效果

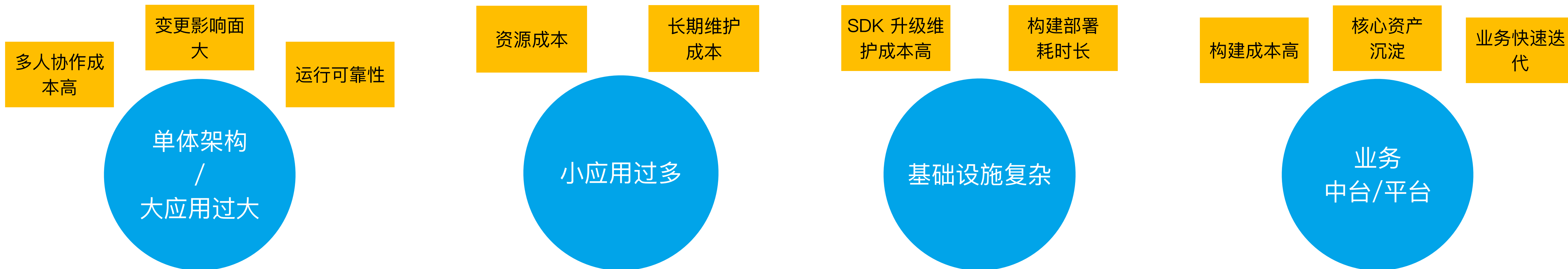
3 SOFAServerless 的运维与调度平台介绍

4 蚂蚁的实践经验与案例

5 总结与展望

总结

适用场景



让普通应用低成本享受到 Serverless 收益的研发框架与平台

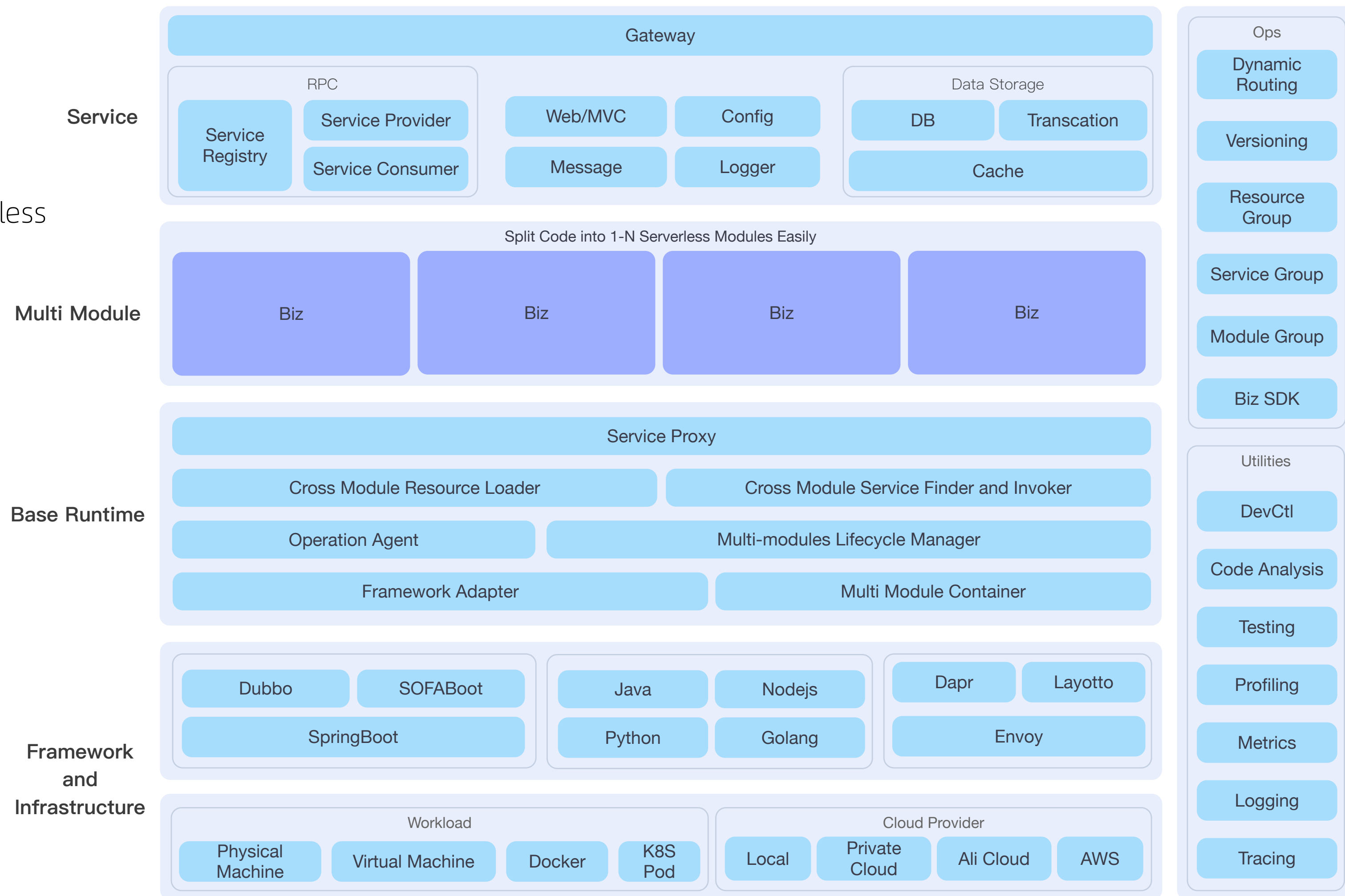
展望 - 开源与愿景

开源

- 能力开源：框架，研发工具，运维调度平台
- 官网：<https://sofaserverless.netlify.app/>
- Github：<https://github.com/sofastack/sofa-serverless>

长期目标与愿景

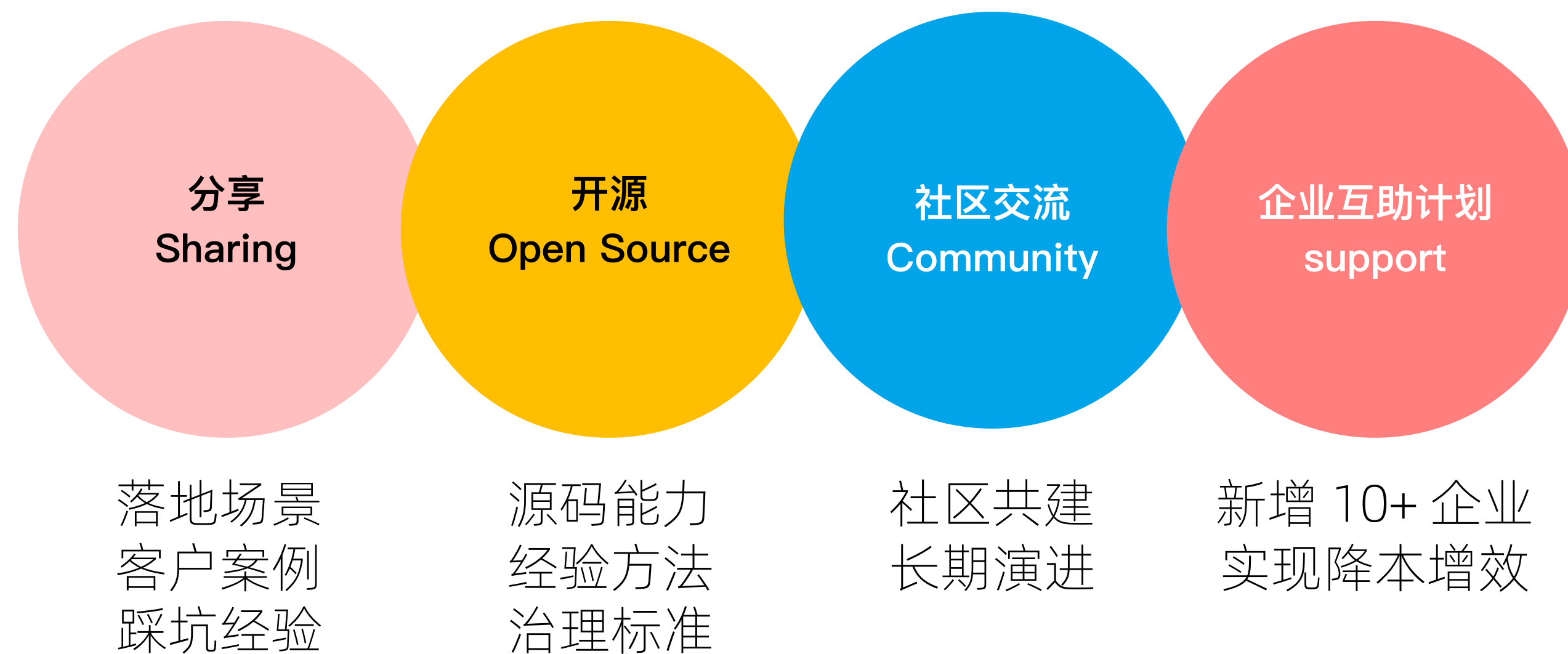
- Speed as you need
- Pay as you need
- Deploy as you need
- Evolution as you need
- 支持多语言，多框架
- 1000 家企业



关键时间计划

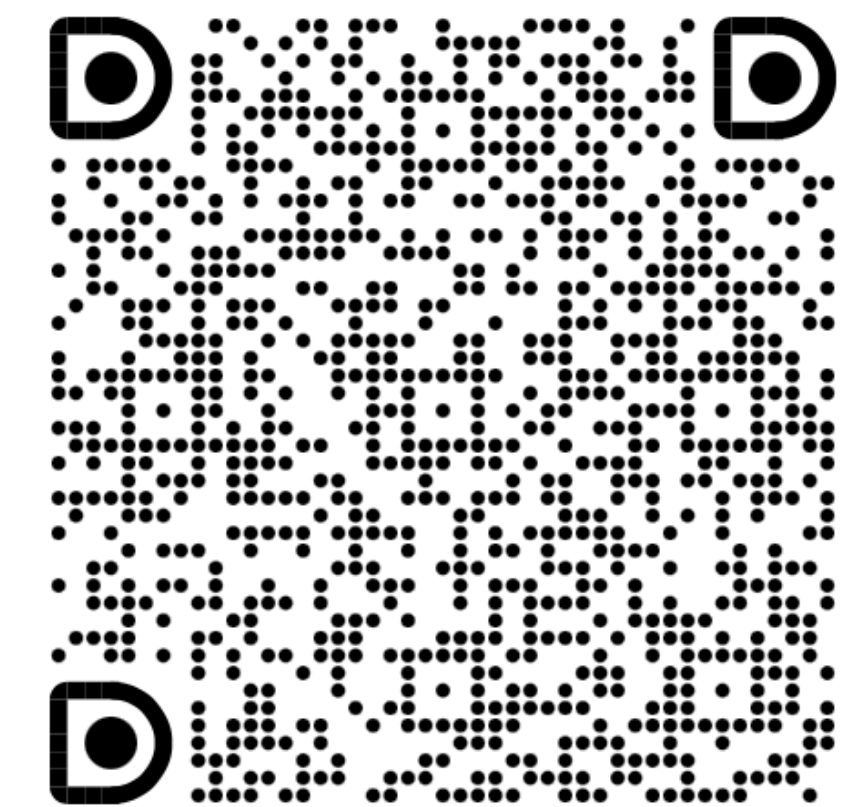
当前：SOFAArk 模块化组件已开源 4 年，ModuleController、Arklet、RuntimeAdaptor 12 月 1.0 版本即将重磅发布！

可统计到的有 15+ 外部企业投产，兼容 25+ 常用中间件。



微信交流群

该二维码7天内有效，过期后重新进入将更新



钉钉交流群

THE END

欢迎一同探索微服务新架构，致力降本增效



• 微信企业沟通交流群



• 开发者沟通协作群