

提升平台&中台项目 研发效能

字节跳动 +Ajqcl r Gq_qlrsarspc - 石延龙

QCon+ 案例研习社



扫码学习大厂案例

学习前沿案例，向行业领先迈进

40个

热门专题

—
行业专家把关内容筹备，
助你快速掌握最新技术发展趋势

200个

实战案例

—
了解大厂前沿实战案例，
为 200 个真问题找到最优解

40场

直播答疑

—
40 位技术大咖，每周分享最新
技术认知，互动答疑

365天

持续学习

—
视频结合配套 PPT
畅学 365 天

个人简介：石延龙



大纲

0	背景概述	<ul style="list-style-type: none">● 字节团队：平台 & 中台 & 基础● Client Infra：技术架构概览● 解决方案：三个重点
1	组件平台	<ul style="list-style-type: none">● 背景与挑战● 方案与效果● 技术实现
2	DevOps	<ul style="list-style-type: none">● 背景与挑战● 方案与效果● 技术实现
3	应用框架	<ul style="list-style-type: none">● 背景与挑战● 方案与效果● 技术实现

字节团队：平台&中台&基础

业务团队

- i 专有业务，通用技术

业务中台

- i 垂直业务，专有技术

平台架构

- i 平台应用的架构、工具、流程

基础技术

- i 通用场景的工具、流程、系统



ByteDance-Client Infrastructure

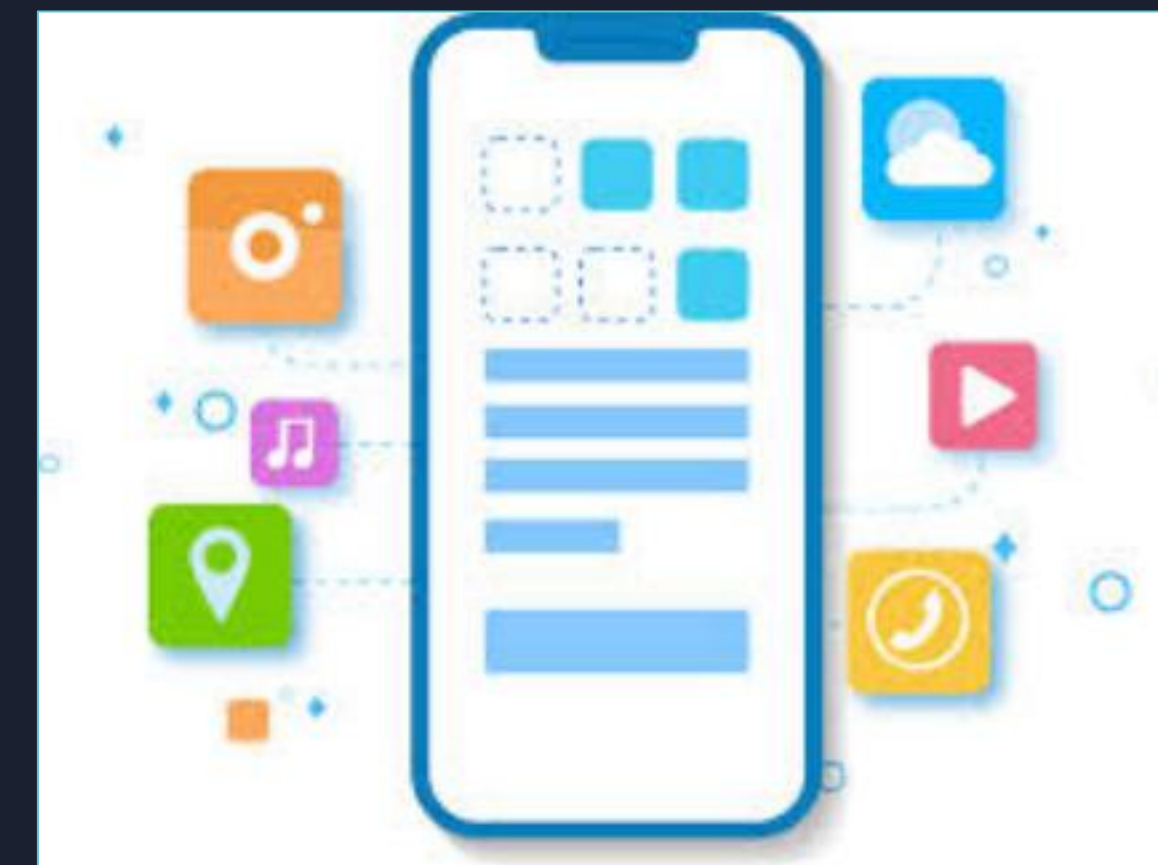
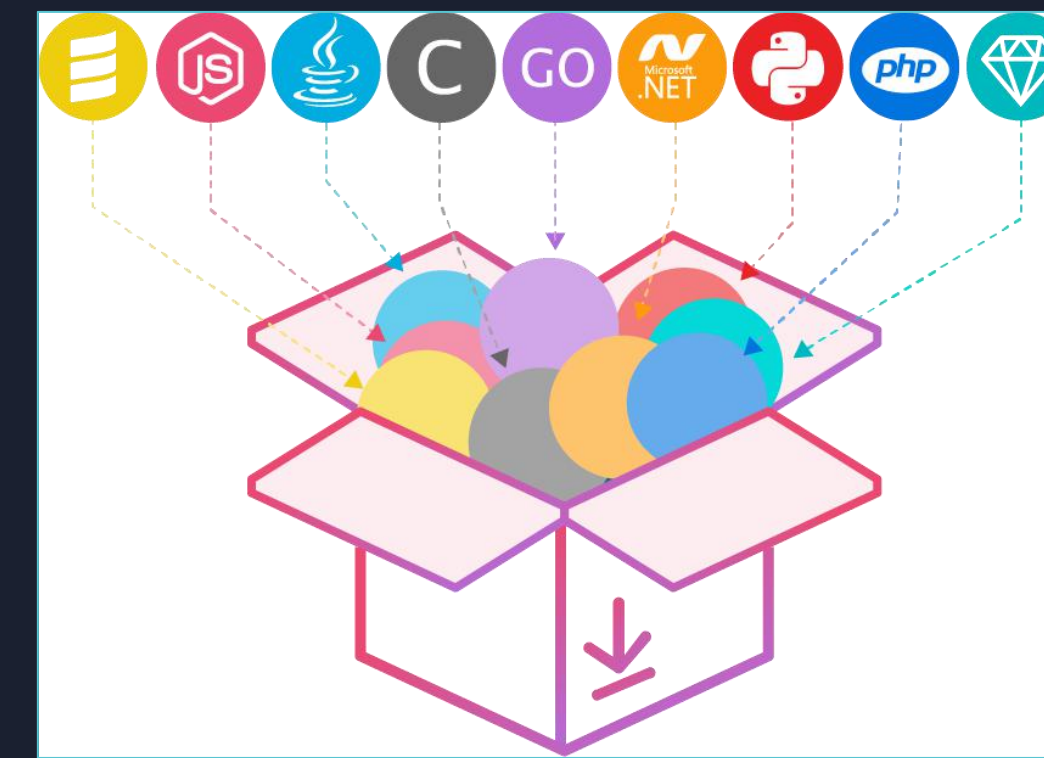
Ajçl r Gç_是为
@wrcB_l ac全公司
大前端提供基础架
构服务的部门，负
责建设通用的终端
基础设施，提升字
节各产品的研发效
率和质量。



愿景：提供行业领先的终端技术,助力字节跳动的业务高效、高质发展。
使命：成为全球顶尖的终端技术平台，引领终端技术发展。

解决方案：重点

- 组件平台：信息化的资产库
 - 基础信息、使用指引、管控能力
- Bct Mnq：可编排的流水线
 - 基础流、多仓流、多宿主流
- 应用框架：可定制的组件集
 - 按需集成、一键启动、自助体验



大纲

0	背景概述	<ul style="list-style-type: none">● 字节团队：平台 & 中台● Client Infra：技术架构● 重点方案：简介
1	组件平台	<ul style="list-style-type: none">● 查找● 使用● 管控
2	DevOps	<ul style="list-style-type: none">● 背景与挑战● 方案与效果● 技术实现
3	应用框架	<ul style="list-style-type: none">● 背景与挑战● 方案与效果● 技术实现

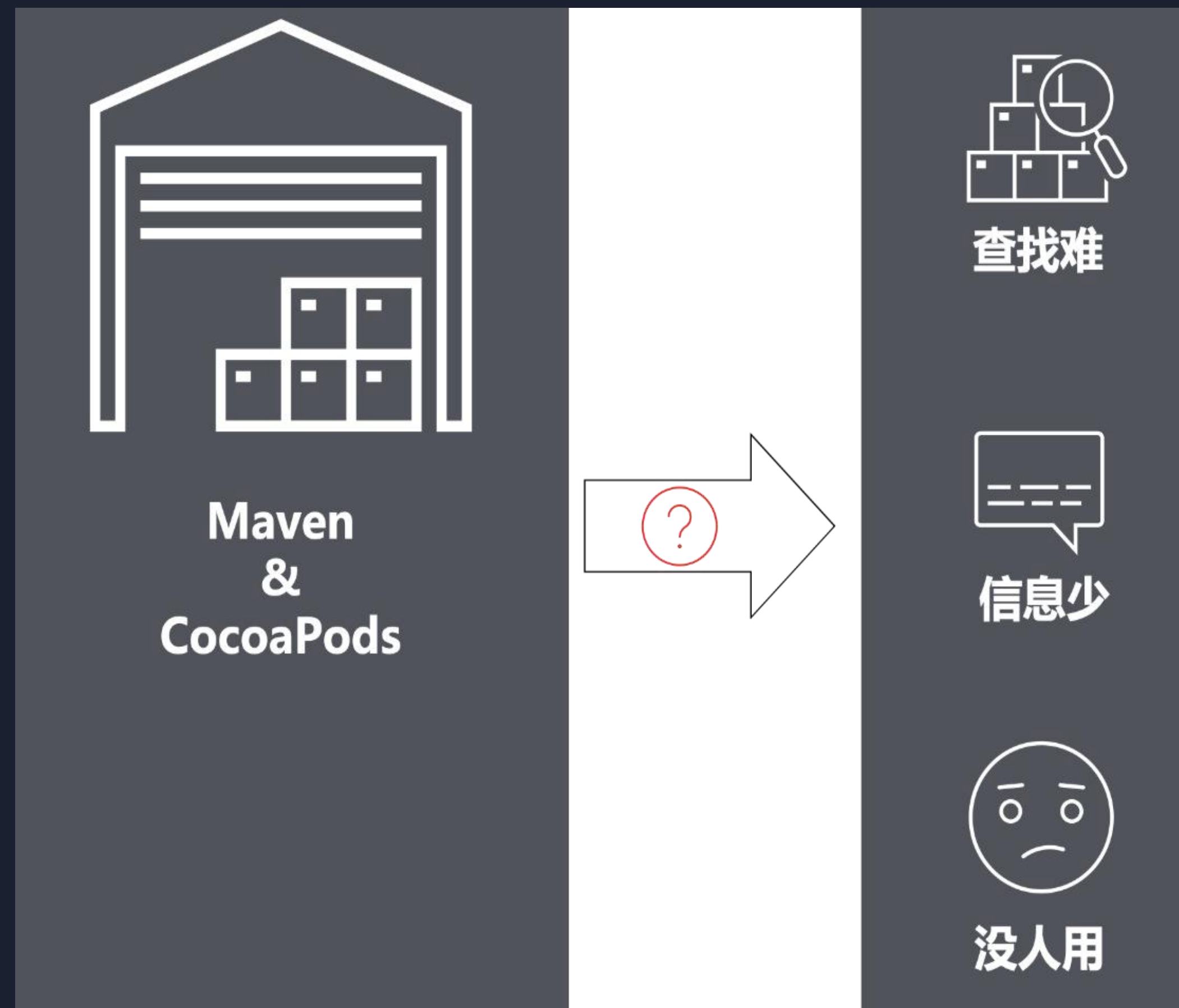
组件：背景&挑战

背景

- 平台 → 组件化
- 中台 → 模块化

挑战：混乱无主的基础组件

- 索引困难
- 信息缺失
- 复用度低



组件平台：方案概览

基础信息

- 开发者将信息注册到平台
- 接入指引、开发信息

使用统计

- 统计各种使用信息
- 应用统计、用户评分

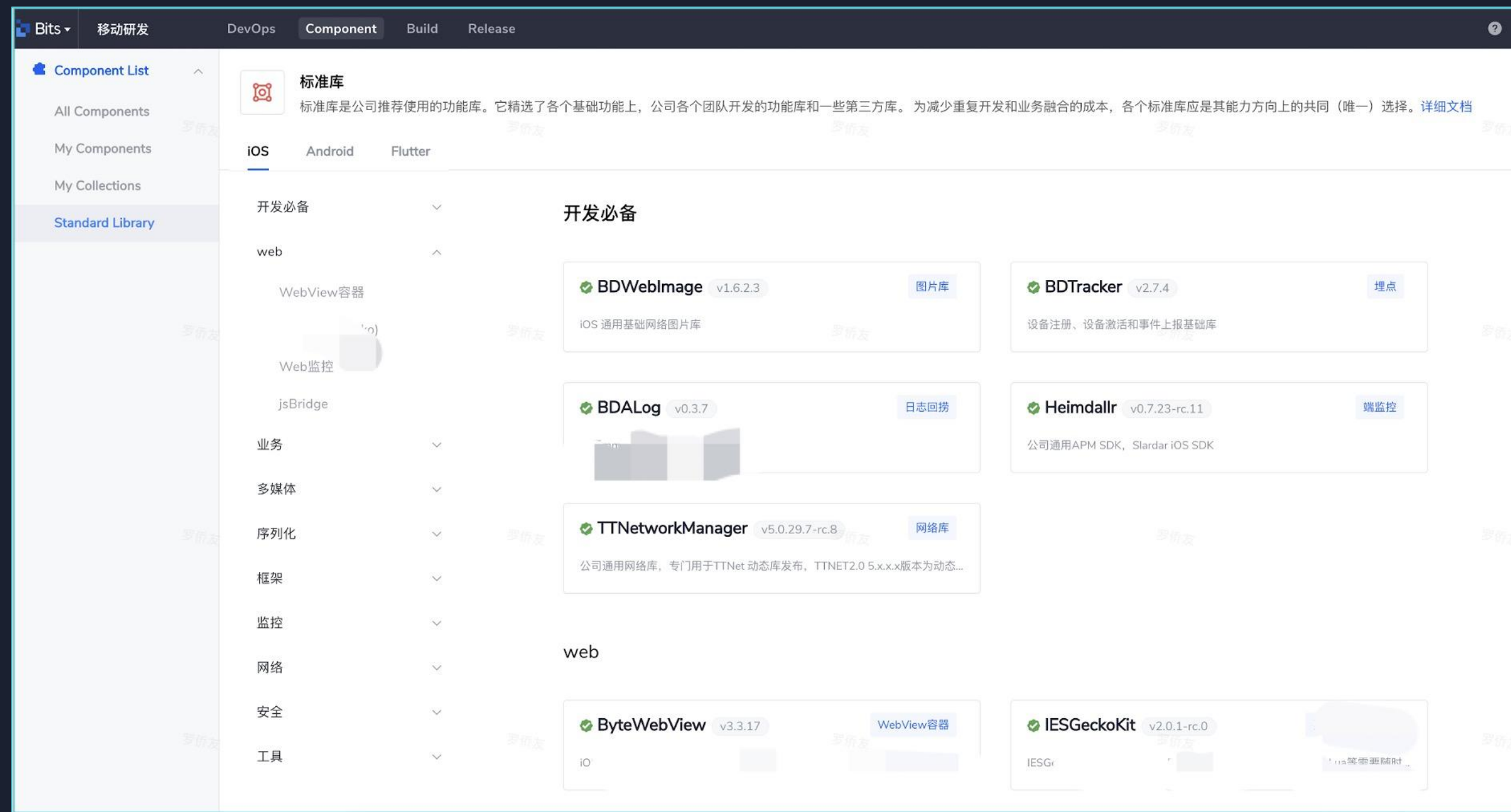
管控能力

- 支持设置组件状态，通知给用户
- 推荐、过期、废弃、禁用

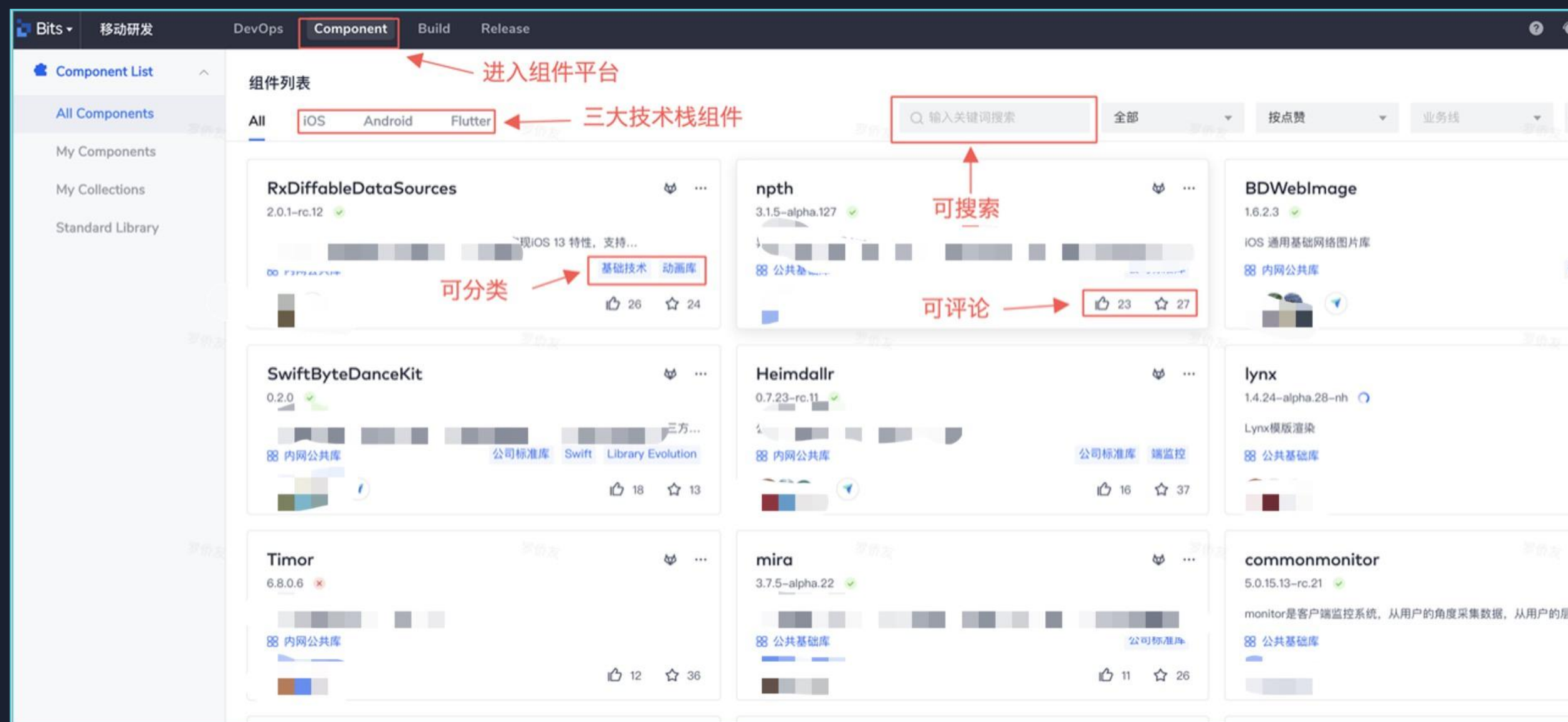


实现I：易查找

- 标准组件：一目了然
- 精选自各团队最常见的基础库
- 覆盖各个基础功能的基础库



- 通用组件：按需搜索
- 按平台、功能来分类
- 按点赞、评论来排序
- 按关键字搜索



实现II：易使用

- i 接入指引：信息完整
- i 文档指引) 操作指引
- i 版本记录：历史清晰
- i 版本号) Af_l ecJørr
- i 使用统计：统计全面
- i 应用统计) 用户反馈

The image displays three screenshots of the Bits component management interface. The top screenshot shows the details for the **BDTracker** component (version 2.10.0-alpha.2), including its introduction and features. The middle screenshot shows the details for the **BDWebImage** component (version 1.6.2.3), highlighting its version history and update logs. The bottom screenshot shows the **BDWebImage** component's version history table, which lists the application name, component version, and the time of installation. A red arrow points to the table with the text "显示哪些app接入了, 当前使用的版本号是多少".

应用名称	组件版本	时间
	1.0.11-rc.0	2021-01-14 22:56:31
	1.0.13	2021-01-07 17:23:01
	1.0.14-rc.0	2021-01-12 20:21:59
	1.0.20.1	2021-01-13 21:05:06
	1.0.21	2021-01-08 20:27:24
	1.0.21.1-bugfix	2021-01-13 19:52:29
	1.0.23	2021-01-11 17:52:57
	1.0.24	2021-01-06 18:55:02
	1.0.24	2021-01-12 22:24:35
	1.0.24	2021-01-13 16:33:28
	1.0.24	2021-01-14 18:40:35
	1.0.24.3-bugfix	2021-01-11 16:29:24
	1.0.24.3-bugfix	2021-01-14 14:41:49
	1.0.24.51	2021-01-15 11:26:37
	1.0.3	2021-01-13 15:21:59
	1.0.9	2021-01-13 21:49:00
	1.2.6	2021-01-04 18:06:20

实现III：可控

依赖分析

直接依赖) 间接依赖

组件状态

推荐、过期、废弃

组件通知

安全管控通知

The screenshot displays the BDWebImage dependency management interface. At the top, it shows the component name 'BDWebImage' and its version '1.8.18.2'. Below this, there is a table of dependencies with columns for 'Component Name', 'Dependency Version', and 'Subspec'. The table lists several dependencies, including '1.4.8.1-binary', '1.8.18.2', and '1.1.0.1-binary'. A red box highlights the 'Dependency Version' column. Below the table, there is a 'Label Settings' dialog box with radio buttons for '普通版本' (Selected), '推荐版本', '稳定版本', '过期版本', and '废弃版本'. Another red box highlights these radio buttons. Below the dialog box, there is a 'Reason' field with a text input containing '这个版本有安全风险, 请升级到5.1.1版' and a checkbox for '创建Lark群并通知所有使用方' which is checked. The interface also shows a '返回 bdtracker' button and a '升级成功' status message.

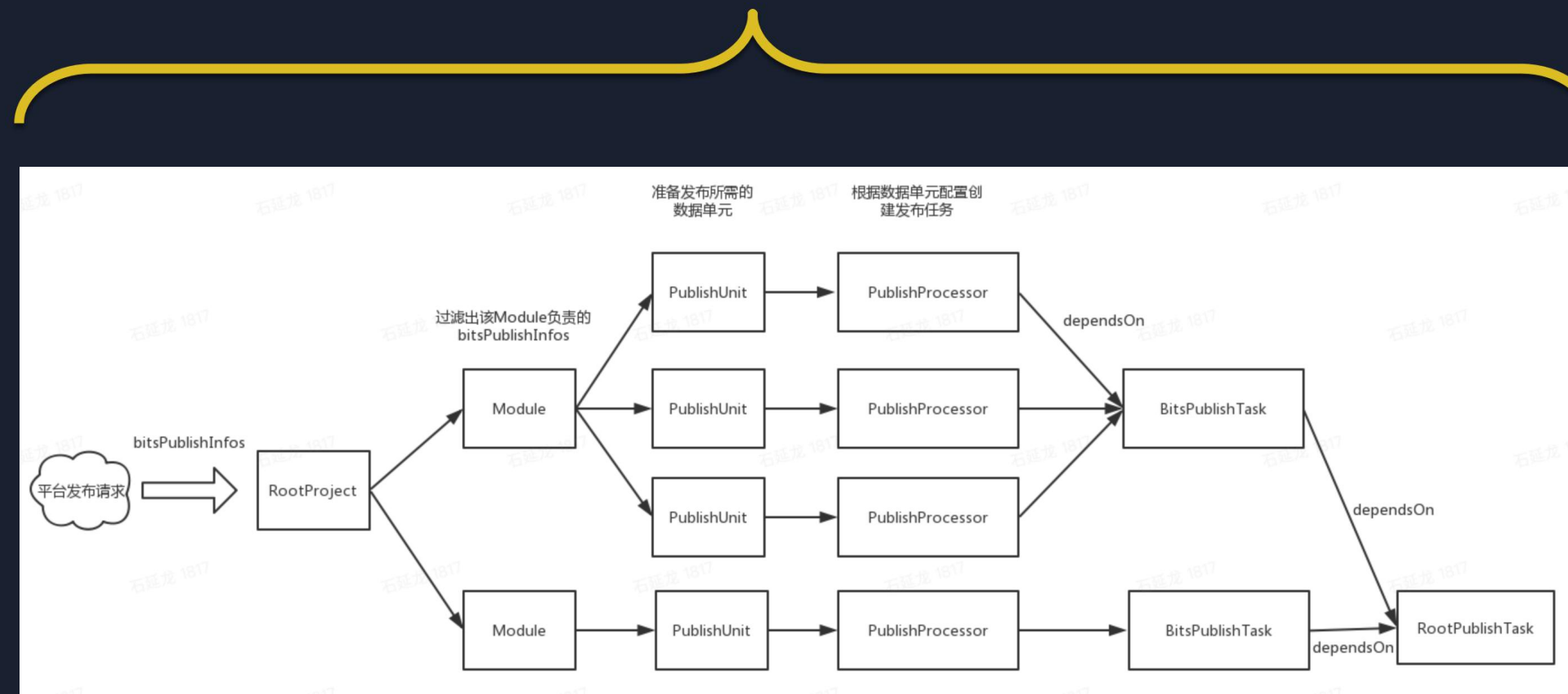
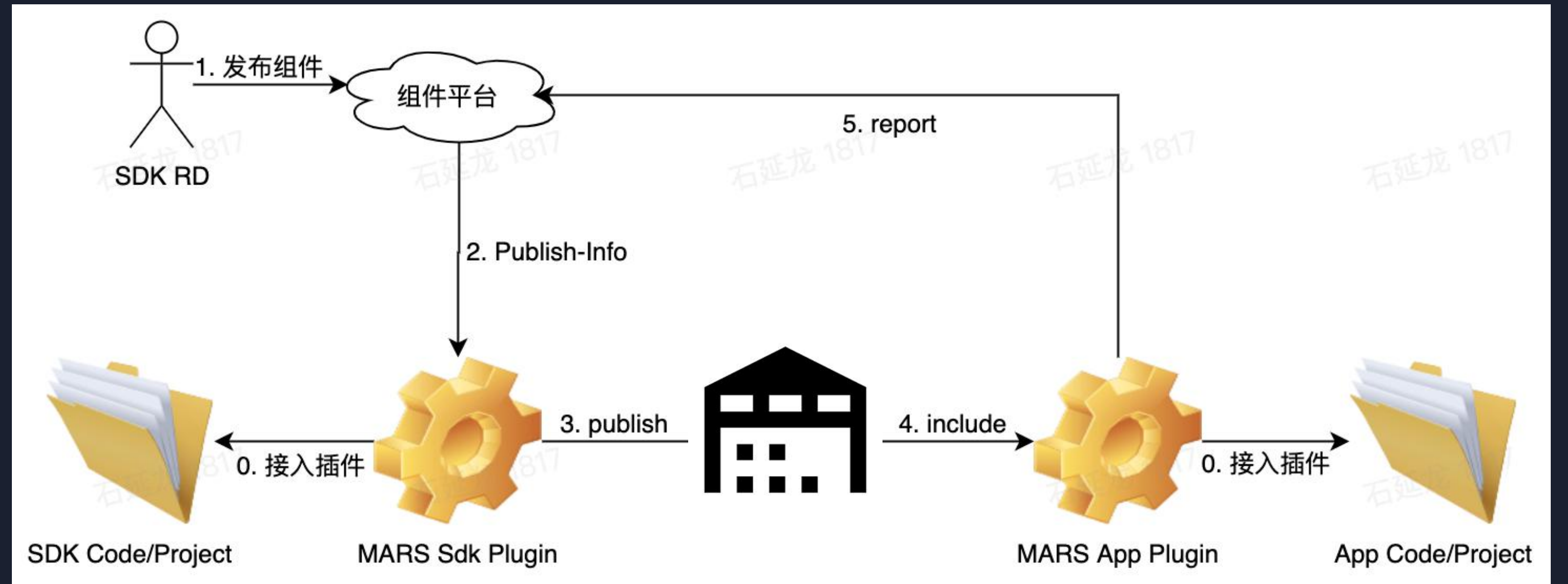
组件平台：技术实现

方案：统一的组件流水线

- 入口，收拢到平台
- 发布，统一发布插件
- 使用，统一引用插件

亮点：扩展功能

难点：？



- 扩展功能
- 1. 管理原生组件(so/a+h)
 - 2. 支持ASAN(构建&使用)
 - 3. 源码追溯→透明中心
 - 4. 统一仓库（加速与鉴权）
 - ...

组件平台：总结

效果&字节

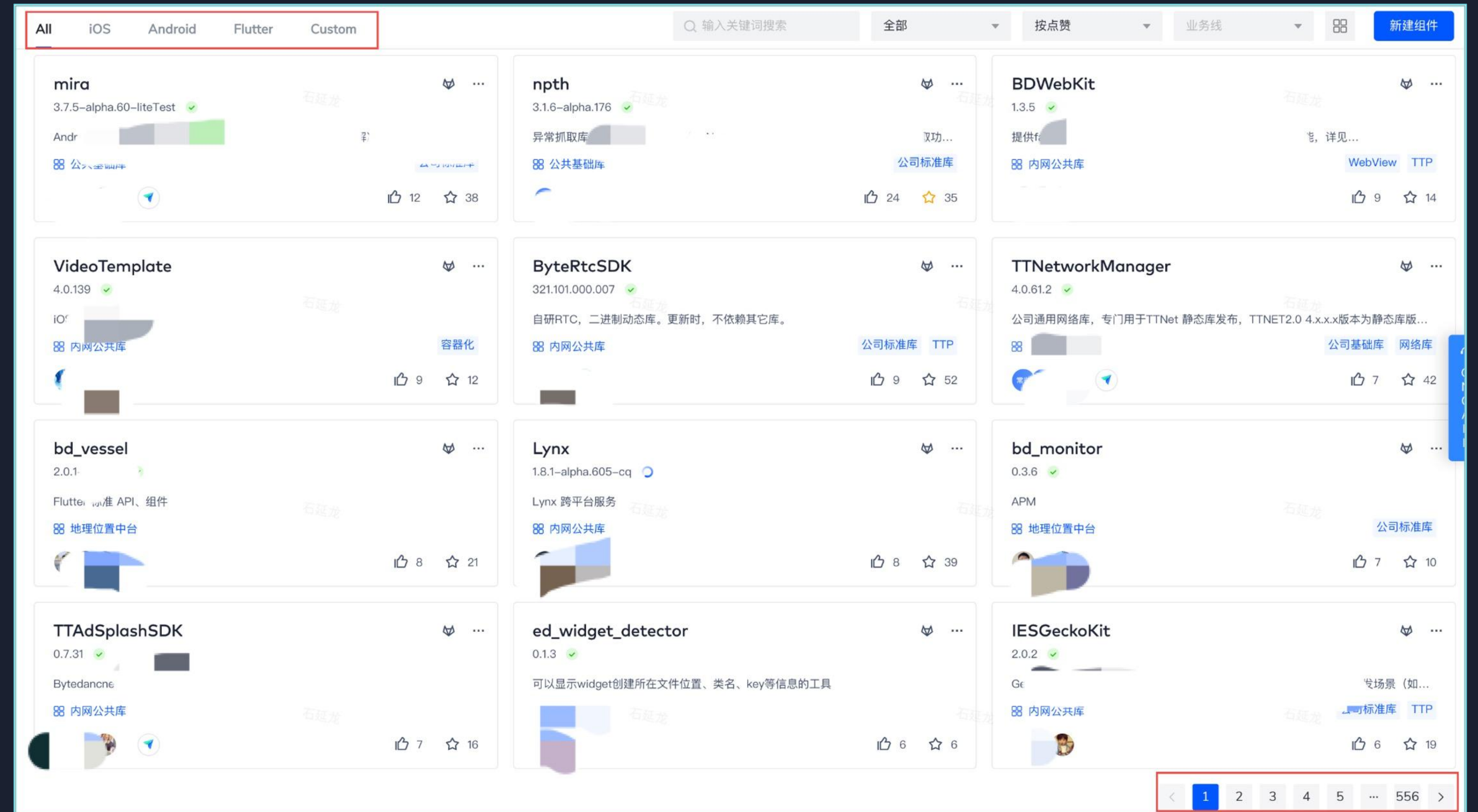
gMQ: 23..)

?l bprp: 54..)

Djsrrcp: 23.)

体验

[火山引擎 # cK?PQ](#)



大纲

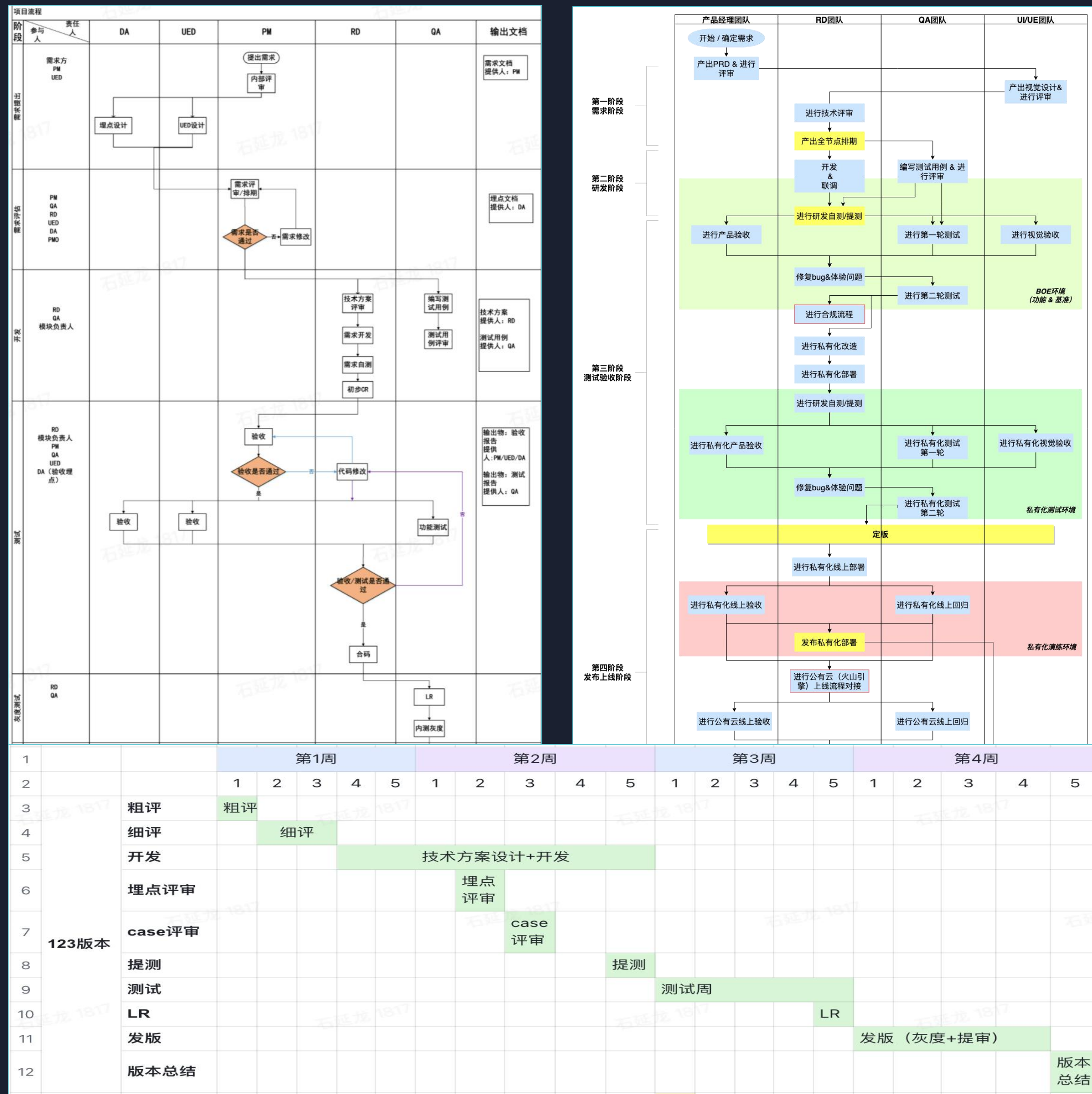
0	背景概述	<ul style="list-style-type: none">● 字节团队：平台 & 中台● Client Infra：技术架构● 重点方案：简介
1	组件平台	<ul style="list-style-type: none">● 背景与挑战● 方案与效果● 技术实现
2	DevOps	<ul style="list-style-type: none">● 基础流● 多仓流● 多宿主流
3	应用框架	<ul style="list-style-type: none">● 背景与挑战● 方案与效果● 技术实现

流程：背景&挑战

背景：无

挑战：复杂多样的研发流程

- 流程长
- 分支多
- 时间久



DevOps: 方案概览

三层产品架构&'自下而上':

原子服务

- 规范化封装的最小功能性任务
- 全流程、多平台

workflow

- 一系列原子服务串-并联组成的任务流
- 自由编排、按需配置

模版

- 字节跳动在研发模式的沉淀
- 基础流、多仓流、多宿主流



DevOps: 基础流

开发流

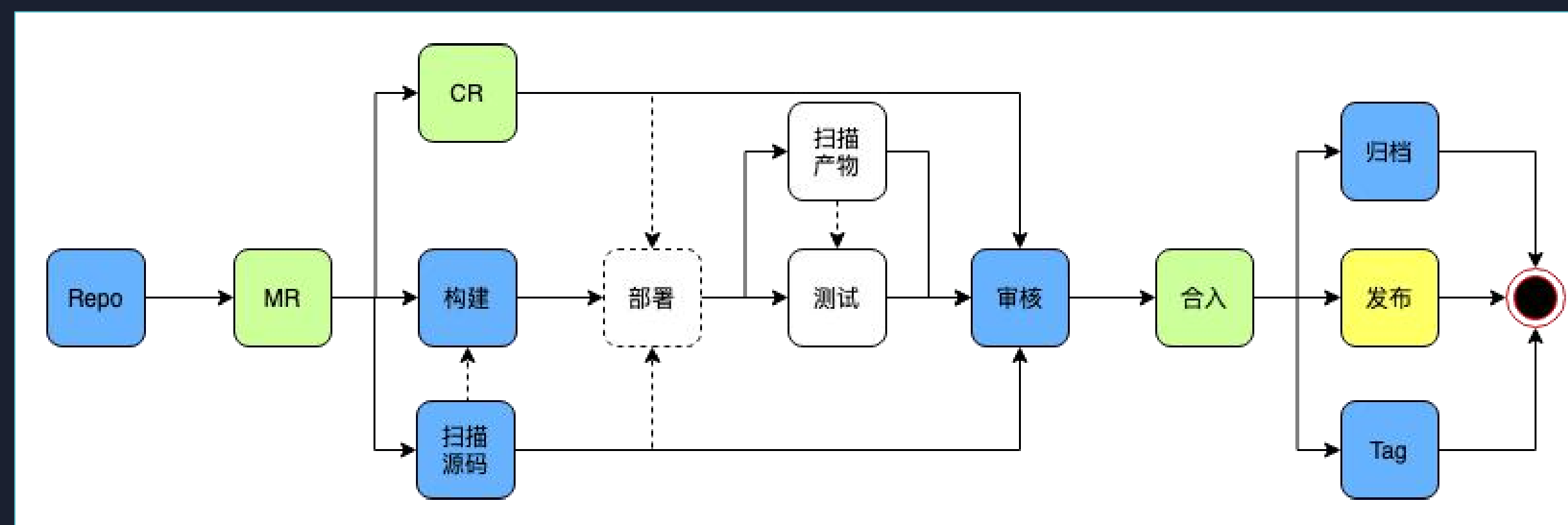
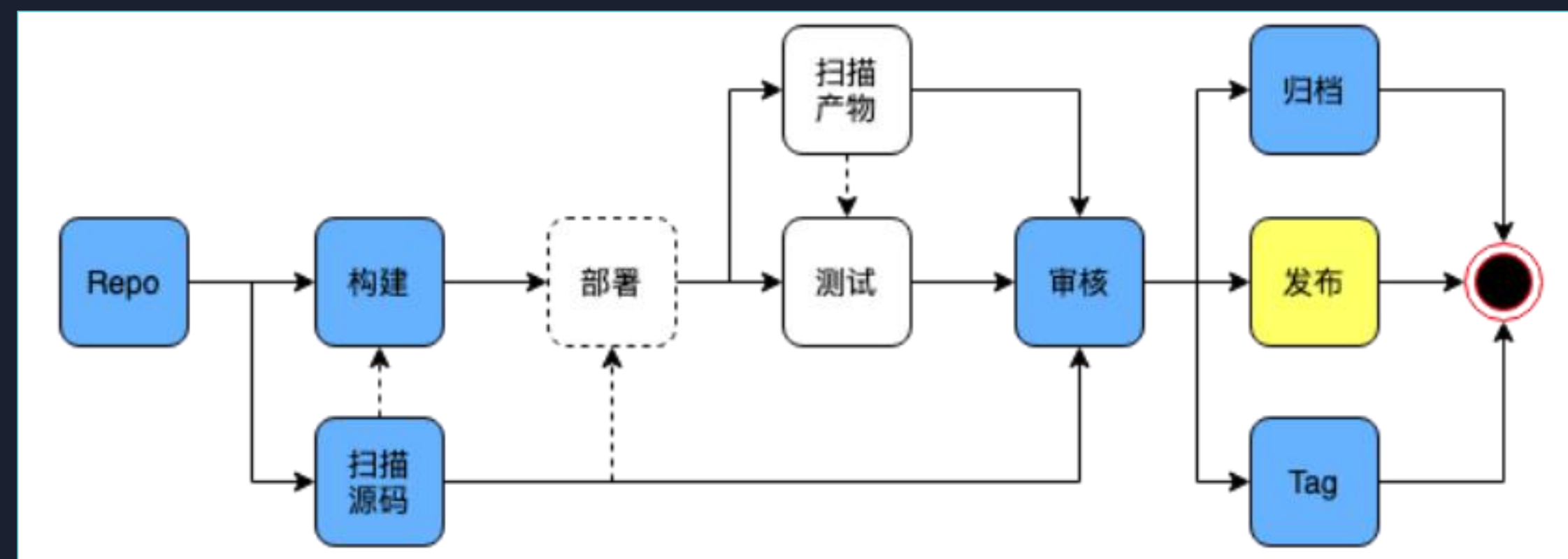
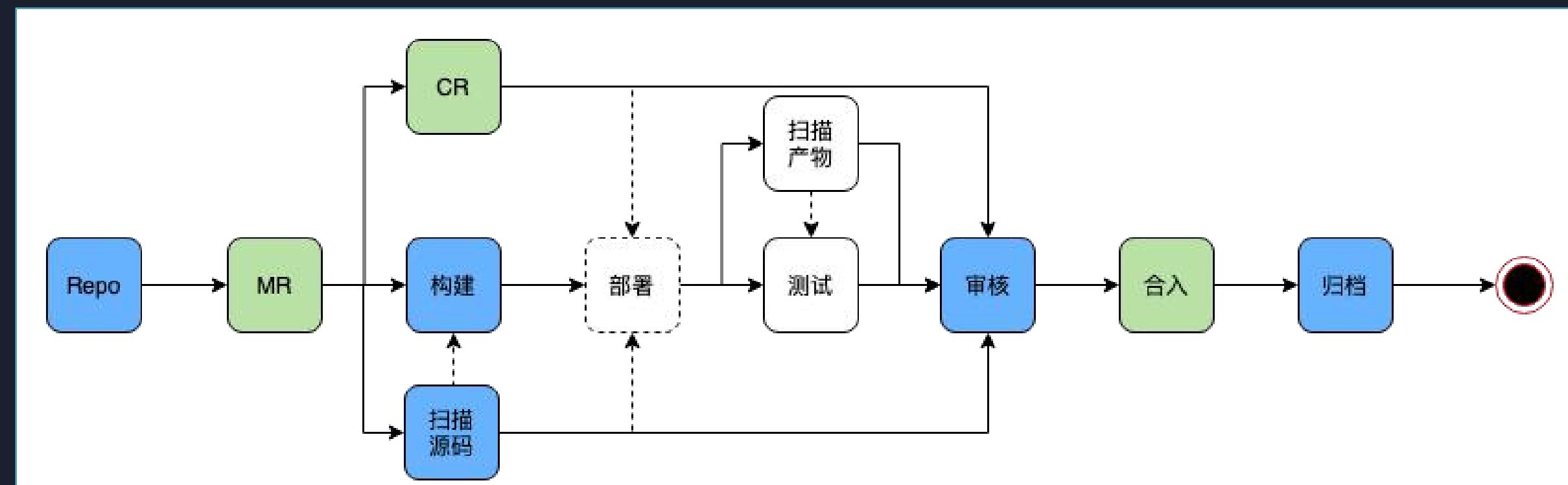
开发新功能时，拉出 `develop` 分支，进行持续的开发、测试，最终合入主干。

发布流

发布新版本时，拉出 `production-ready` 分支，进行持续的集成测试，最终灰度发布。

修复流

紧急修复外网 `issue` 时，从对应版本的 `production-ready` 分支独立拉出 `hotfix` 分支，进行快速的开发、测试，直接对外灰度发布，并将 `hotfix` 合入主分支。



DevOps: 多仓流I

应用场景

- 模块化、组件化的平台项目
- 同时修改组件、模块，和主工程

背景问题

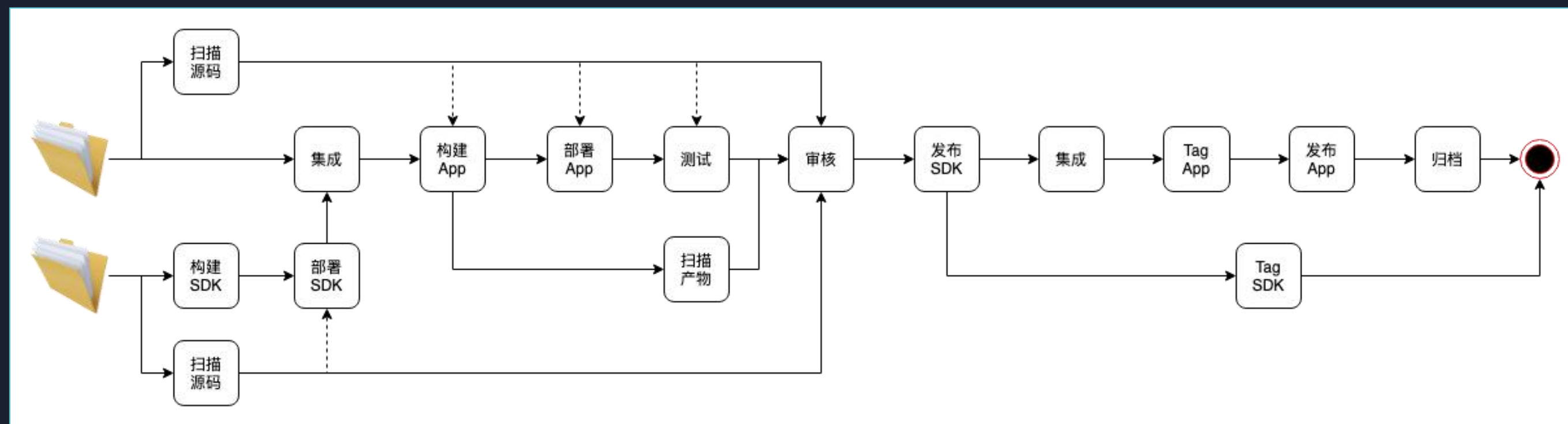
- 流程复杂
- 操作繁琐

```
uf gc& se] dcc' y
dmp&qbi g qbi ]jgr' y
  bct &qbi '
  ns` jgrf &qbi '
  g rcep_rc& nn* qbi '
  {
  bct & nn'
  {
```

DevOps: 多仓流II

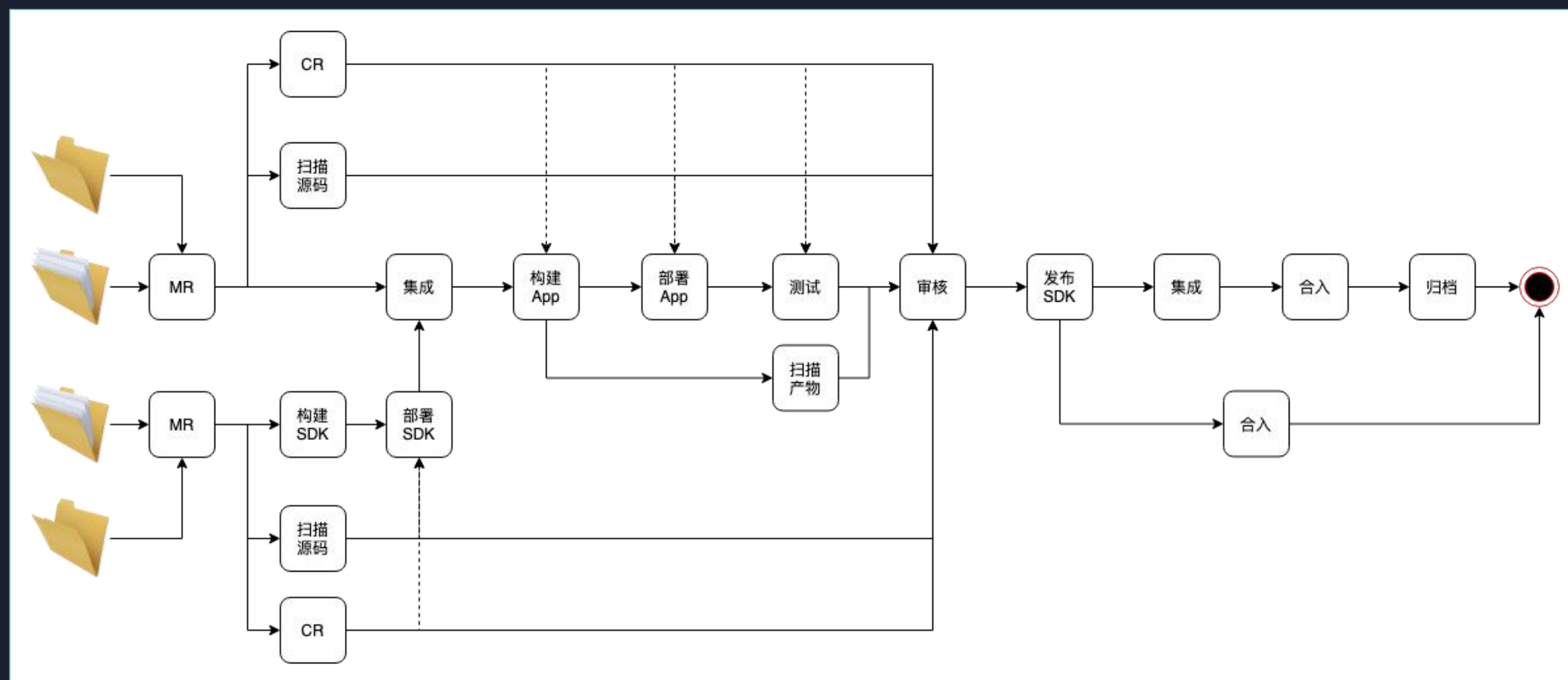
方案

- 为QBI 和?nn, 分别创建基础流
- 以“集成”原子服务, 串联QBI 和?nn



场景

- 多仓单分支
- 多仓多分支



DevOps: 多宿主流I

应用场景

- 中台开发新版
- 发布到多个宿主? nn

背景问题

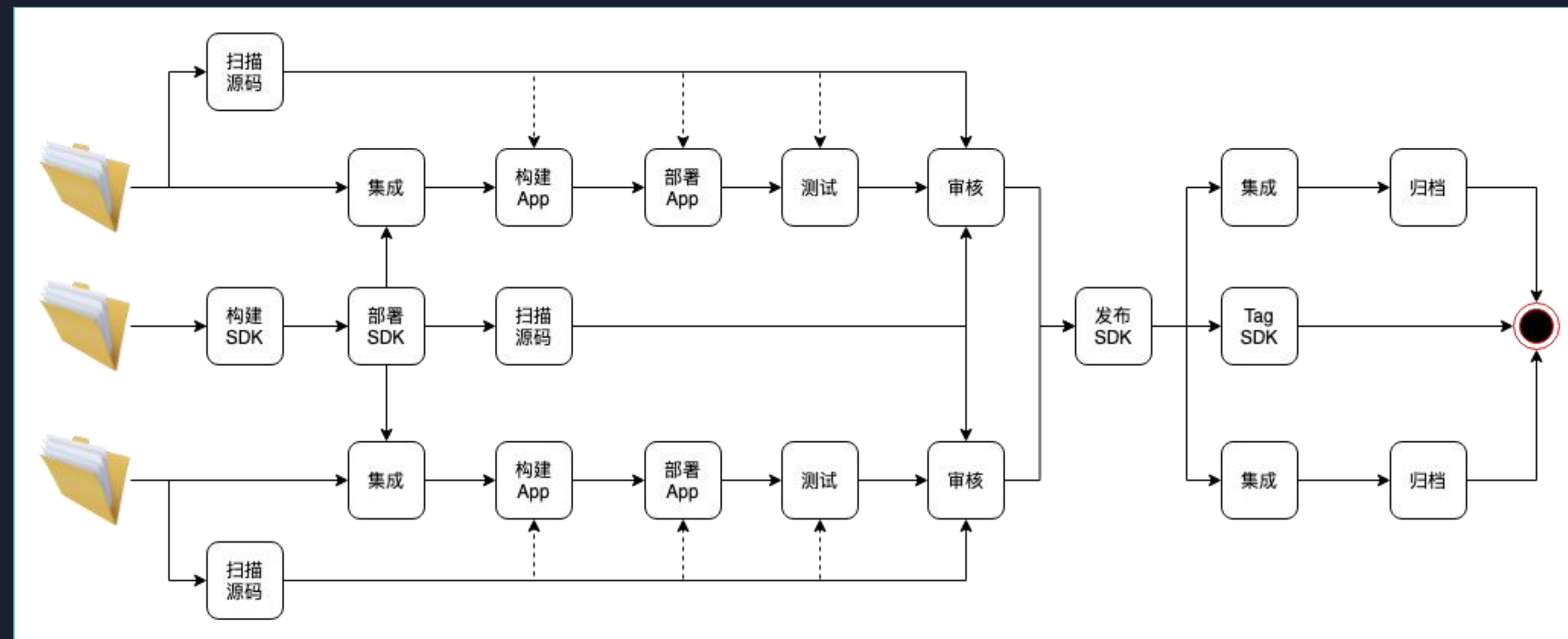
- 流程复杂
- 操作繁琐

```
uf jc& se] dcc' y
  bct & nbsjc'
  ns` jgf & nbsjc'
  dmp& nn dj f mqrq' y
    dj rcep_rc& nn* k nbsjc'
  bct & nn'
  {
  {
```

DevOps: 多宿主主流II

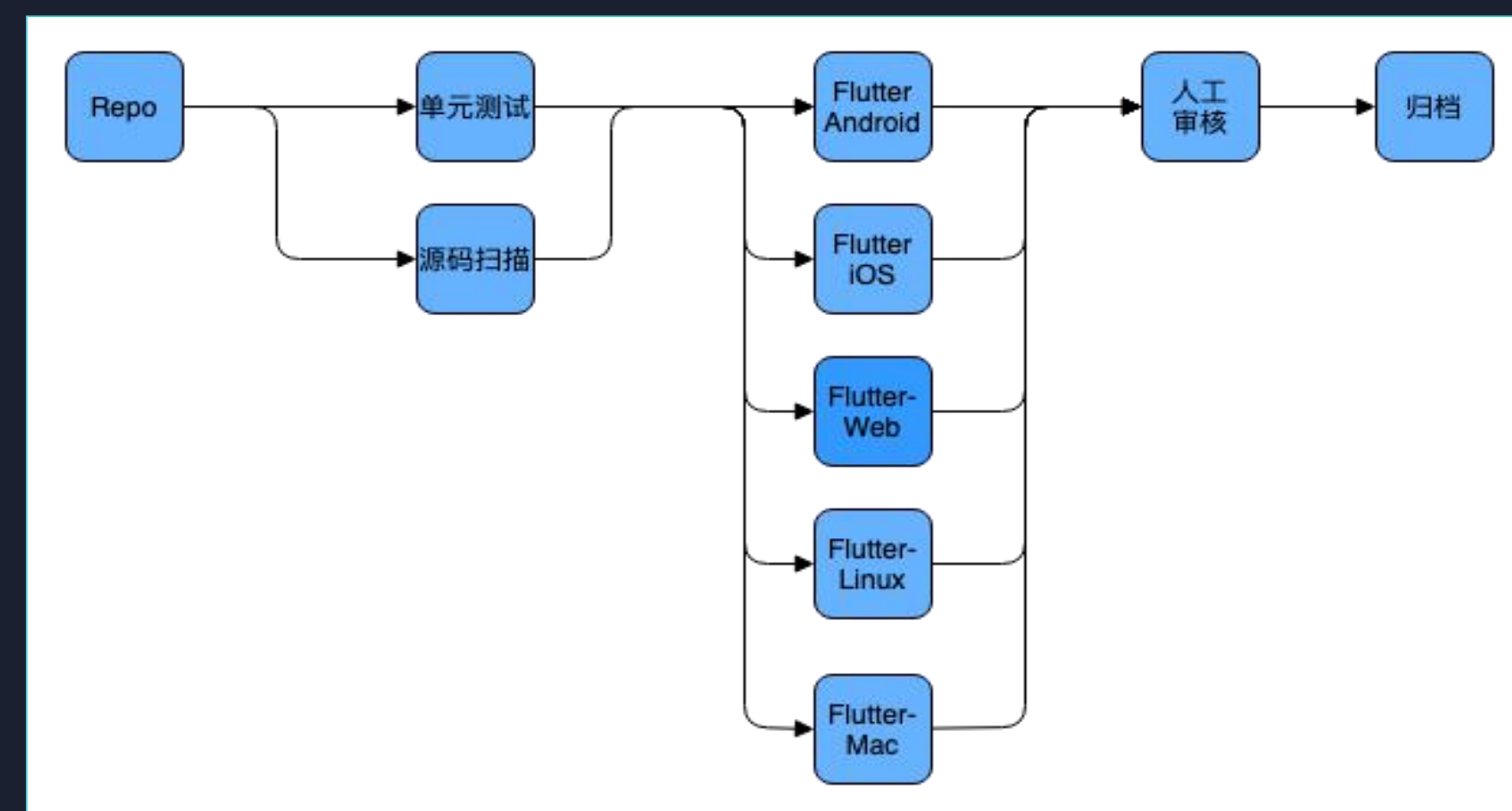
方案

- 为K mbsjc和? nn, 分别创建基础流
- 以“集成”原子服务, 串联QBI \$? nn
- 引入多个? nn的基础流



场景

- 中台多宿主
- Djsrrcp多端



DevOps: “集成”方案

方案: 规范的依赖管理

1, 统一的依赖配置

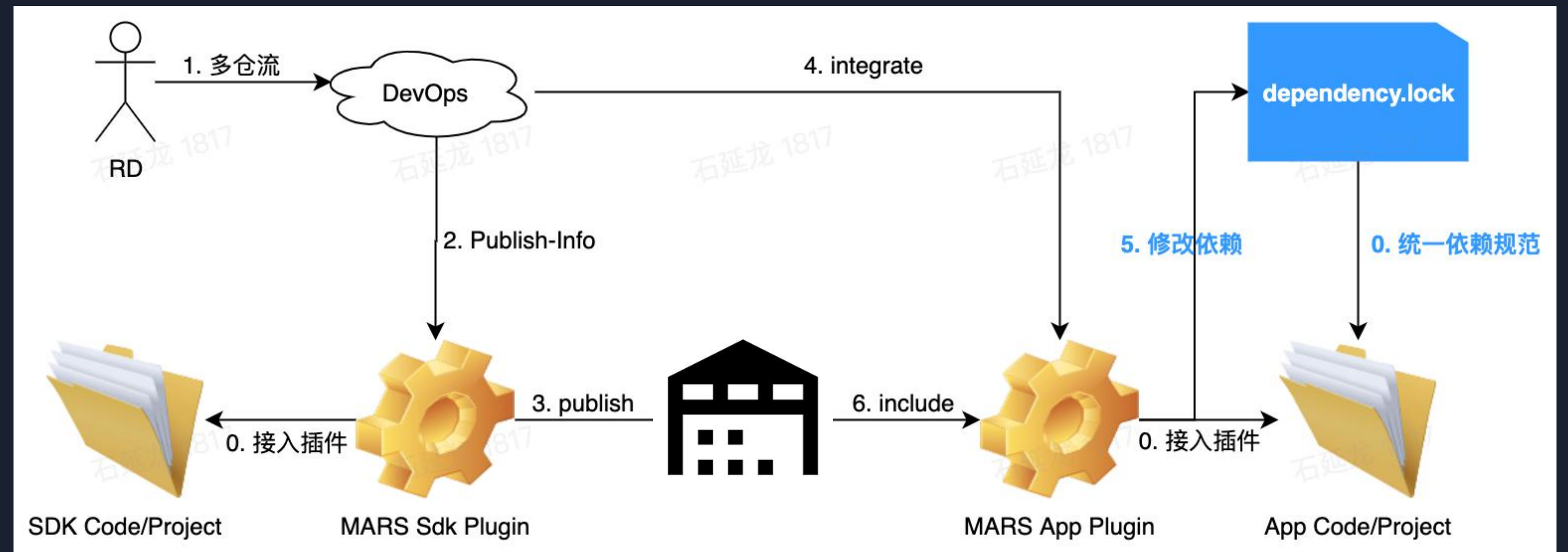
0, 统一的组件发布流

亮点: 本地多仓研发

难点: ?

亮点: 本地多仓研发
1. 源码 → 组件 → 加速构建
2. 组件 → 源码 → 方便开发
3. 自动切换 → 单仓效果

更多信息: 字节MBox



DevOps: 总结



效果

1, 基础流: 常规开发

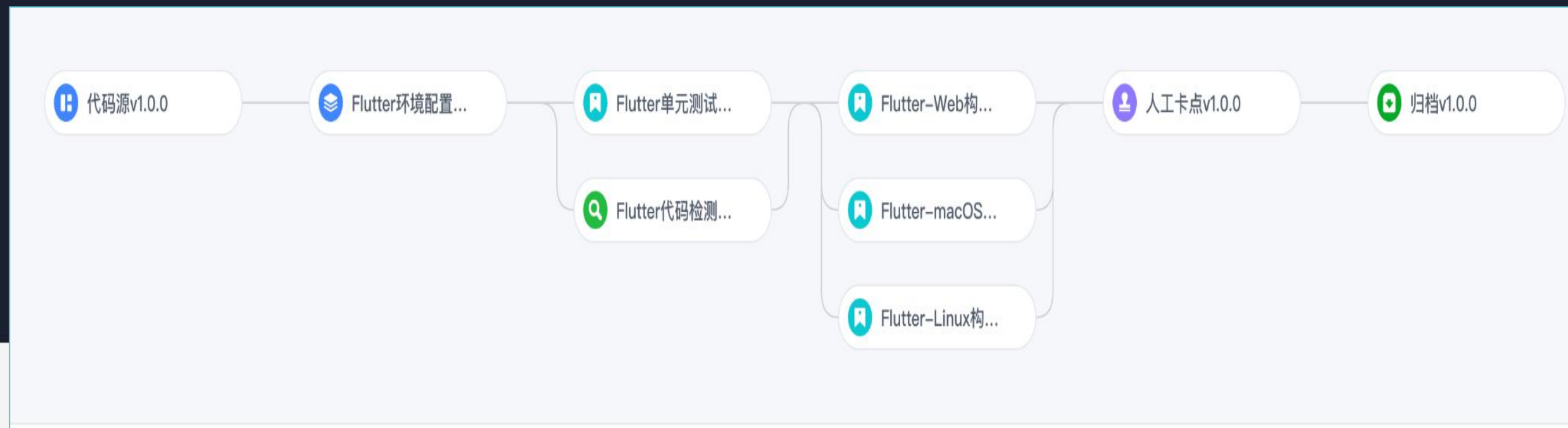
0, 多端流: Djsrrcp一项目多平台

1, 多仓\$多宿主: 平台\$中台



体验

1, [火山引擎# cK?PQ](#)



大纲

0	背景概述	<ul style="list-style-type: none">● 字节团队：平台 & 中台● Client Infra：技术架构● 重点方案：简介
1	组件平台	<ul style="list-style-type: none">● 背景与挑战● 方案与效果● 技术实现
2	DevOps	<ul style="list-style-type: none">● 背景与挑战● 方案与效果● 技术实现
3	应用框架	<ul style="list-style-type: none">● 按需集成● 一键启动● 自助体验

框架：背景&挑战



背景

- ?nn优 → ?nn多
- 通用需求多 → 框架要求高

挑战：庞大笨重的工程架构

- 迭代慢
- 编译久
- 性能差



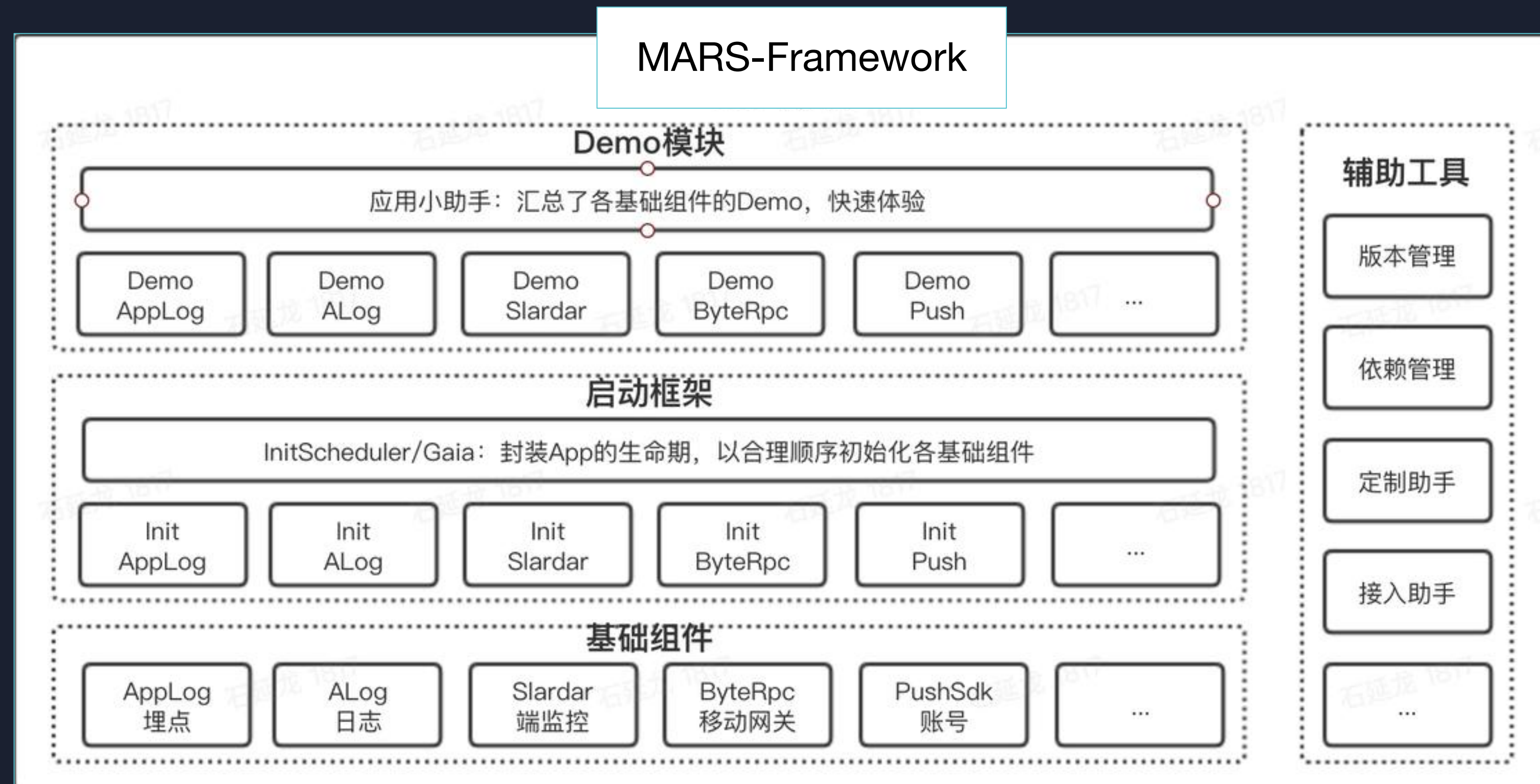
应用框架：方案概览

功能目标

- i 按需集成
- i 一键启动
- i 自助体验

预期效果

- i 加速新项目的启动
- i 汇聚并优化框架技术



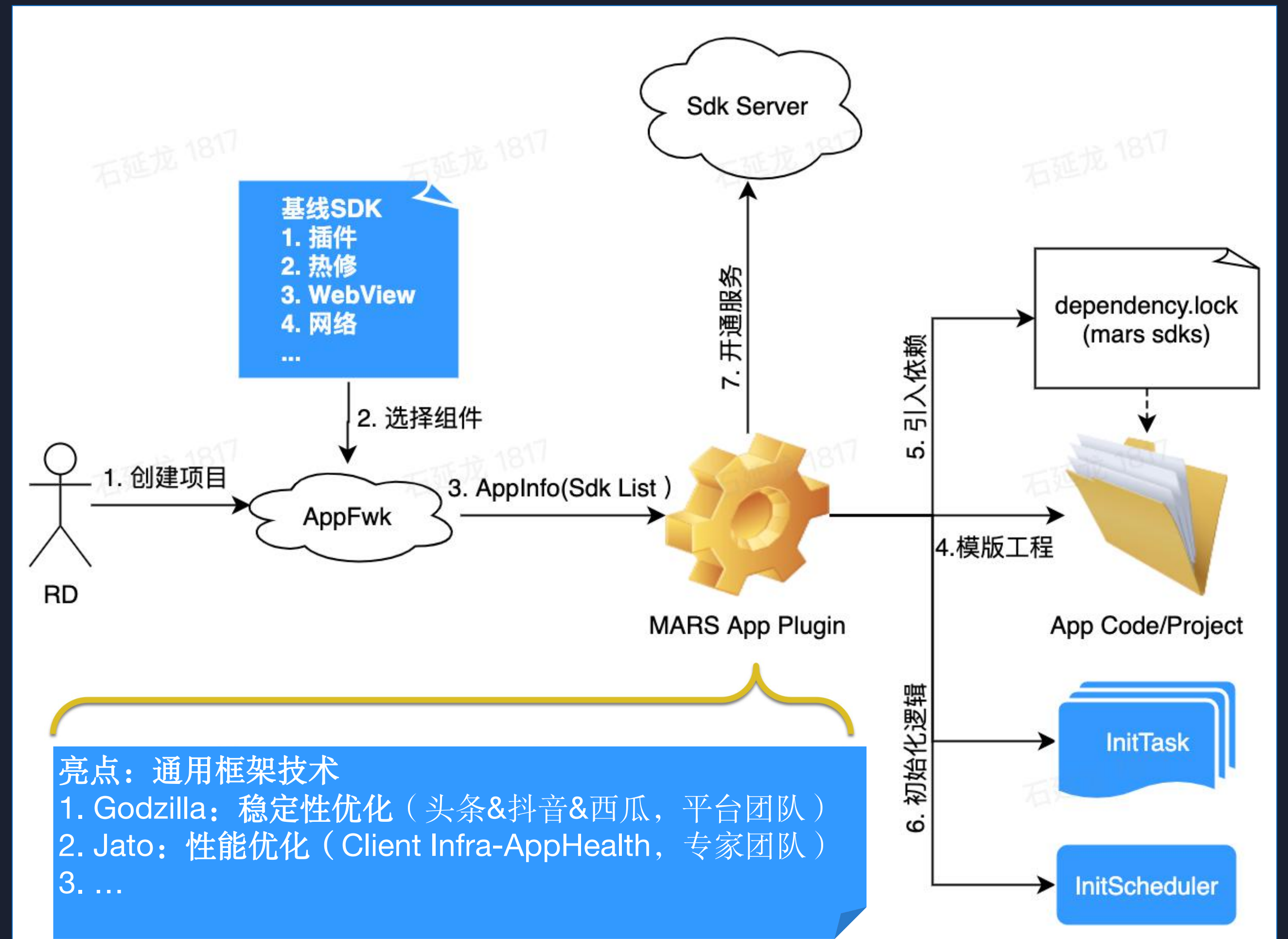
应用框架：技术方案

方案：可编排的启动框架

- 1. 生成时按需选择
- 2. 构建时服务发现
- 3. 运行时灵活调度
- 4. 后台联通配置

亮点：通用框架技术

难点：？



应用框架：总结

效果

创建新项目：3 B_w+ / F msp

体验

[火山引擎 # cK?PQ](#)



总结

平台-中台项目的挑战

- ❖ 混乱无主的基础组件
 - ❖ 索引困难、信息缺失、版本冲突
- ❖ 复杂多样的研发流程
 - ❖ 分支多、标准差、各不相同
- ❖ 庞大笨重的工程架构
 - ❖ 编译久、迭代慢、性能差

字节+Alibaba+Google的方案

- ❖ 组件平台：信息化的资产库
 - ❖ 易查找、易使用、可管控
- ❖ Bct Mnq：可编排的流水线
 - ❖ 基础流、多仓流、多宿主流
- ❖ 应用框架：可定制的组件集
 - ❖ 按需集成、一键启动、自助体验

火山引擎

www.volcengine.com

智能激发增长

veMARS

www.volcengine.com/product/mars

助力研发升级



cmyk

(字节/Client Infra/石延龙)

为一线互联网公司核心技术 人员提供优质内容

☑ TGO专访

☑ 技术干货

☑ 每周精要

☑ 行业趋势



关注 InfoQ 公众号

THANKS

Life feeds on negative entropy.