TLC 2019 大前端 信息流

腾讯LIVE开发者大会
TENCENT LIVE CONFERENCE

微保小程序的开发与架构实践

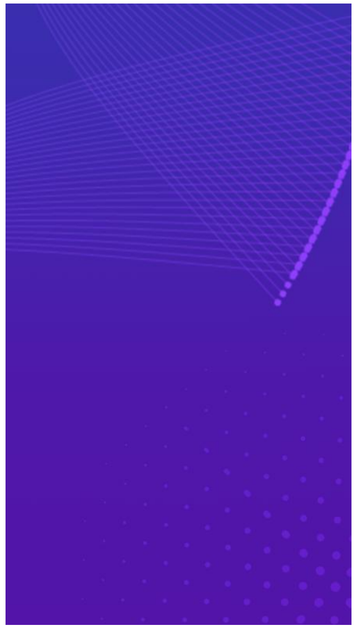
Brook Zhao from 微保

个人介绍

赵小溪

- 微保前端架构师
- 擅长小程序、WEB平台的基础架构搭建、工程化的规划与实施

内容大纲



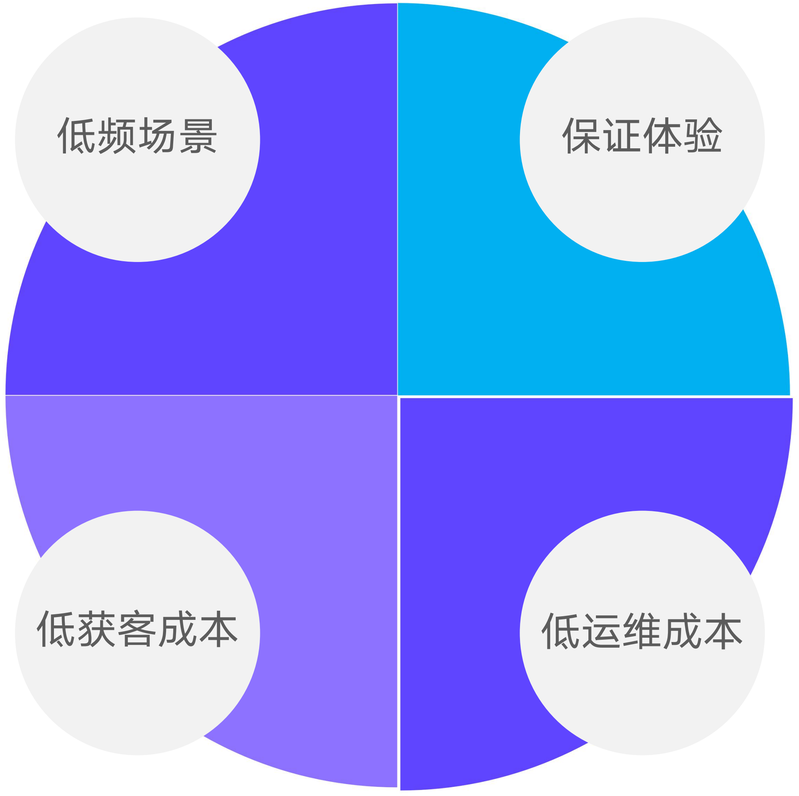
关于微保

微信钱包九宫格——保险服务

微保小程序遇到的挑战

为什么选择小程序平台？

保险的购买及售后(查询、理赔)是典型的低频操作



使用体验优于Web平台

分享方便，轻量

无繁琐的下载动作

不需要经过各大应用市场

不需要额外的服务器资源及分包、离线缓存等相关功能定制

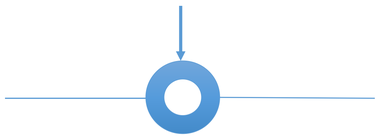
我们的目标

- 体积小
- 加载速度快
- 迭代速度快
- 开发流程清晰可控
- 支撑足够多的业务
- 出色的交互体验

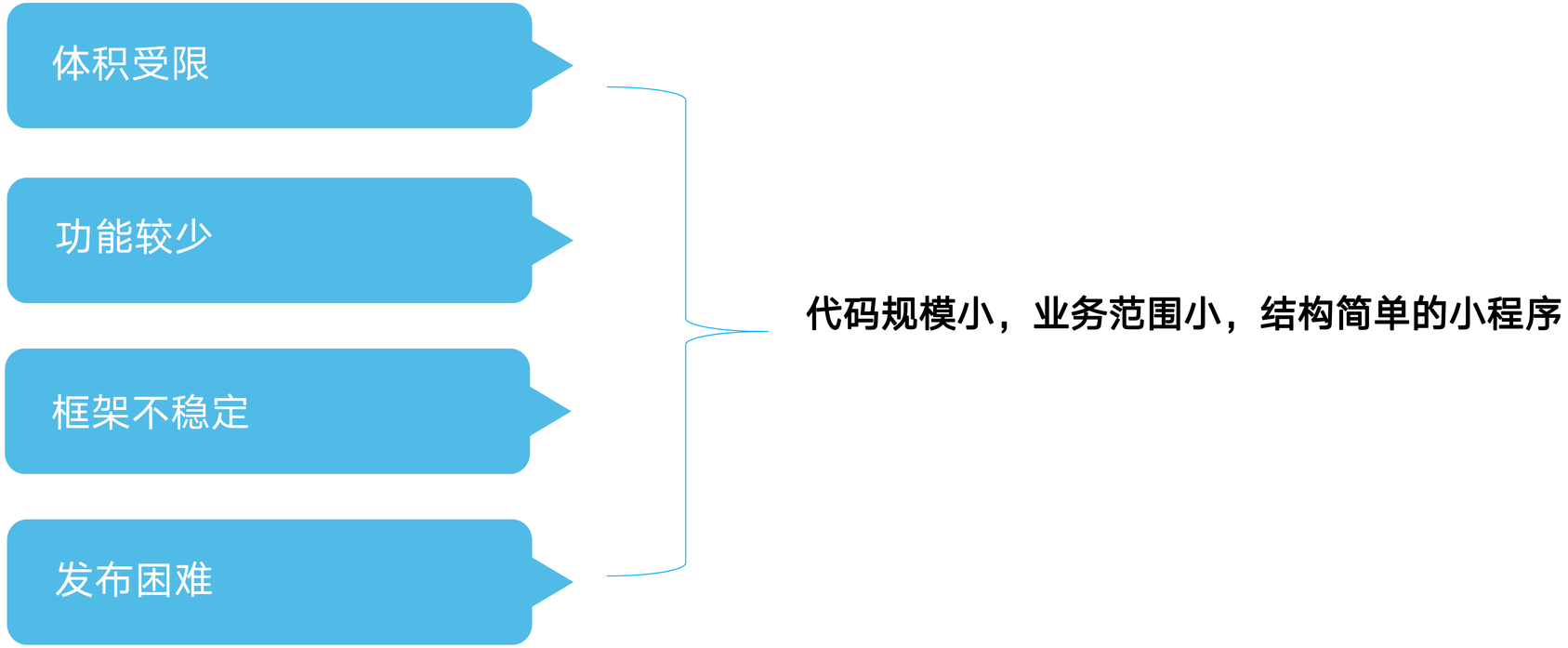
一切从这里开始

在这个时间点，我们能做出怎样的小程序？

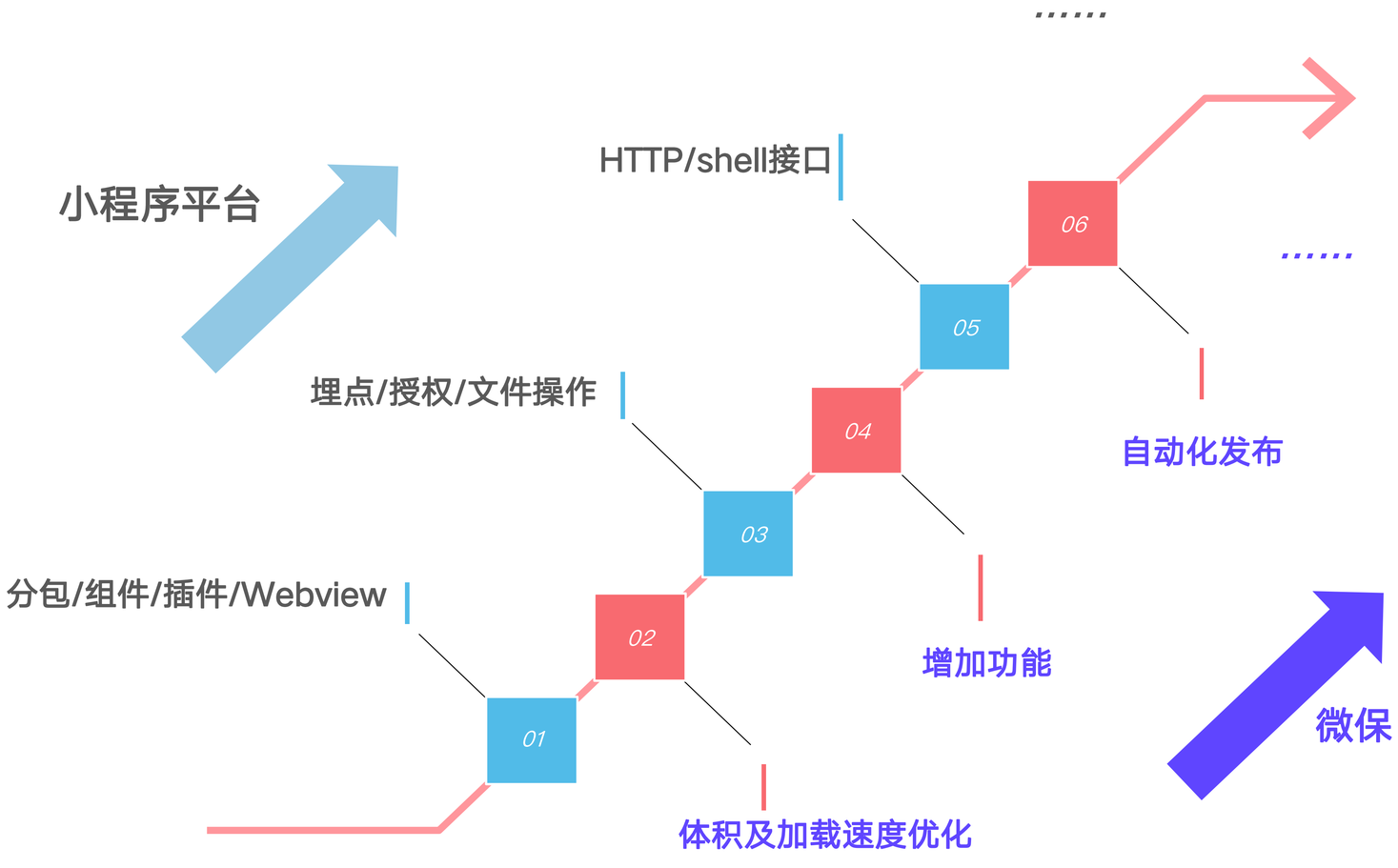
2017.4



早期的挑战



和小程序一起成长


.....
小程序平台
HTTP/shell接口
06
.....
05
埋点/授权/文件操作
04
自动化发布
03
分包/组件/插件/Webview
02
增加功能
01
微保
体积及加载速度优化

大规模小程序仍需要的

平台架构

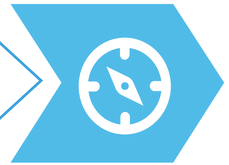
工程化

开发管理

微保小程序的技术架构

保险产品有什么特点（相对实物产品）？

保险，卖的是文案



展示文案的关键信息吸引用户

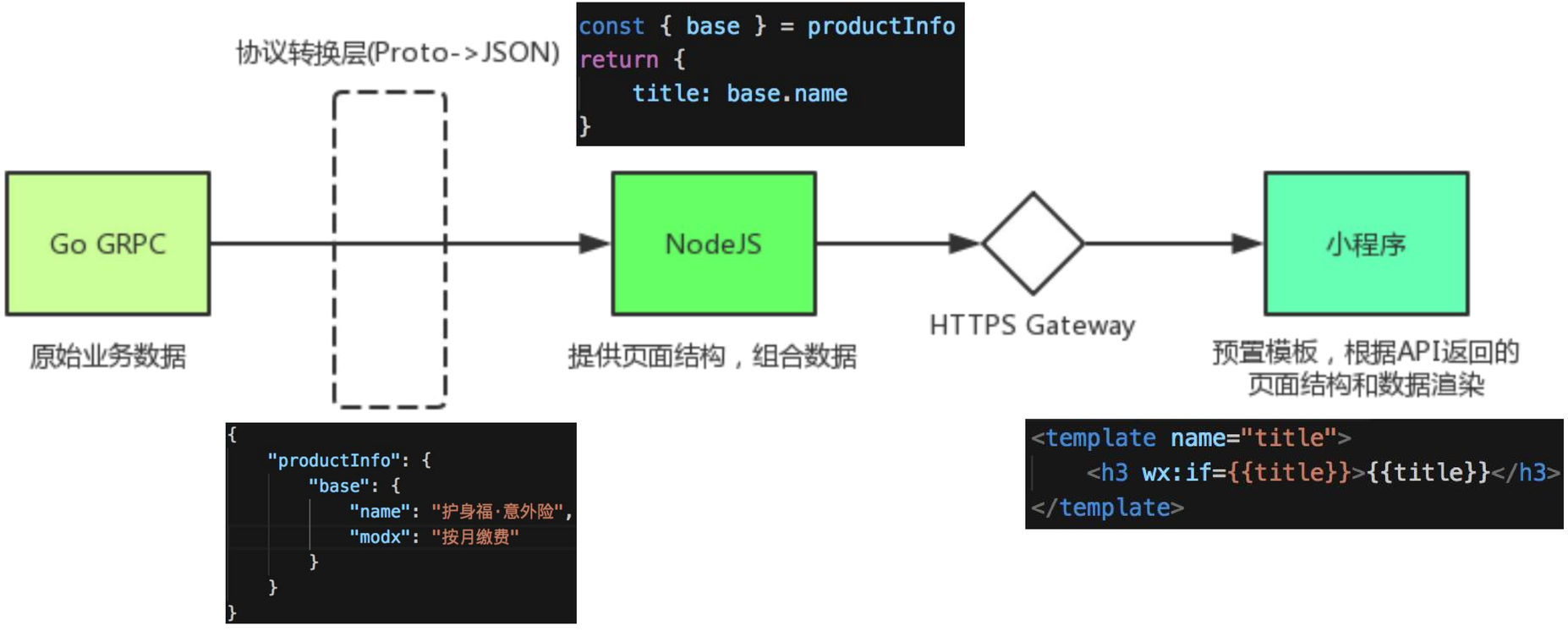
前端架构向文案维护倾斜

技术架构重点之一：满足业务的灵活性要求

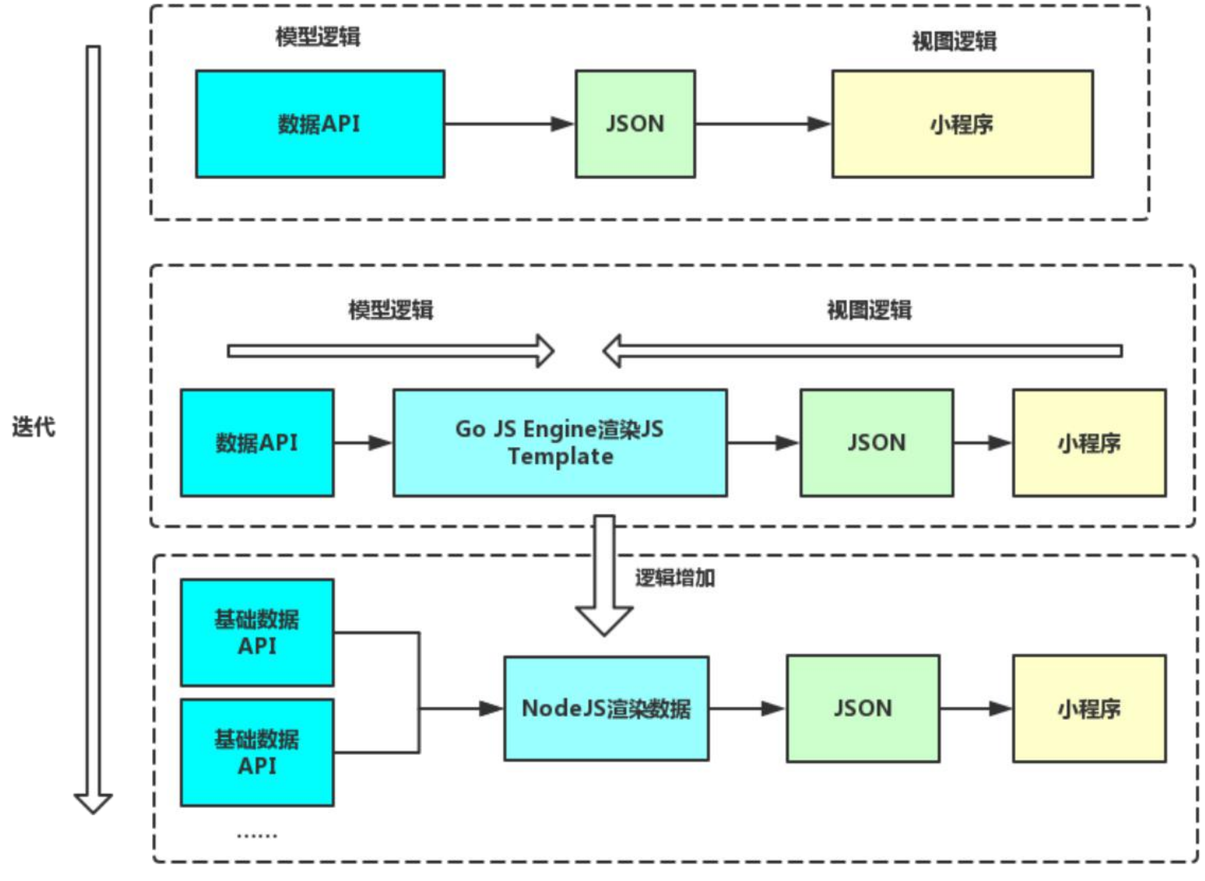
微保小程序的特点

- 模板化
- 组件化
- 页面框架场景化
- 文案可配置化

模板化


协议转换层(Proto->JSON)
const { base } = productInfo
return {
title: base.name
}
Go GRPC
NodeJS
HTTPS Gateway
小程序
原始业务数据
提供页面结构，组合数据
预置模板，根据API返回的
页面结构和数据渲染
{
"productInfo": {
"base": {
"name": "护身福·意外险",
"modx": "按月缴费"
}
}
}
<template name="title">
<h3 wx:if={{title}}>{{title}}</h3>
</template>

模板化API渲染层的演进



组件化

采用小程序平台提供的组件方案

```
Component({
    behaviors: [/** behaviors */],

    properties: {
        /** props */
    },
    data: {
        /** data */
    },
    methods: {
        /** methods */
    }
});
```

页面框架场景化

投保主流程	小程序 + 子包
广告投放	独立子包
外部合作跳转	插件
信息类运营页	WebView

文案可配置化

自研的配置平台(inspired by strapi)

强模式

proto文件直接生成配置页面，数据存储过程中与使用GRPC平台的服务达到数据类型安全

结构化数据

数组及map任意嵌套，可轻易转成JSON格式供HTTP请求调用

版本化数据管控

所有数据的发布基于版本，最大程度保证发布以及回滚的及时性

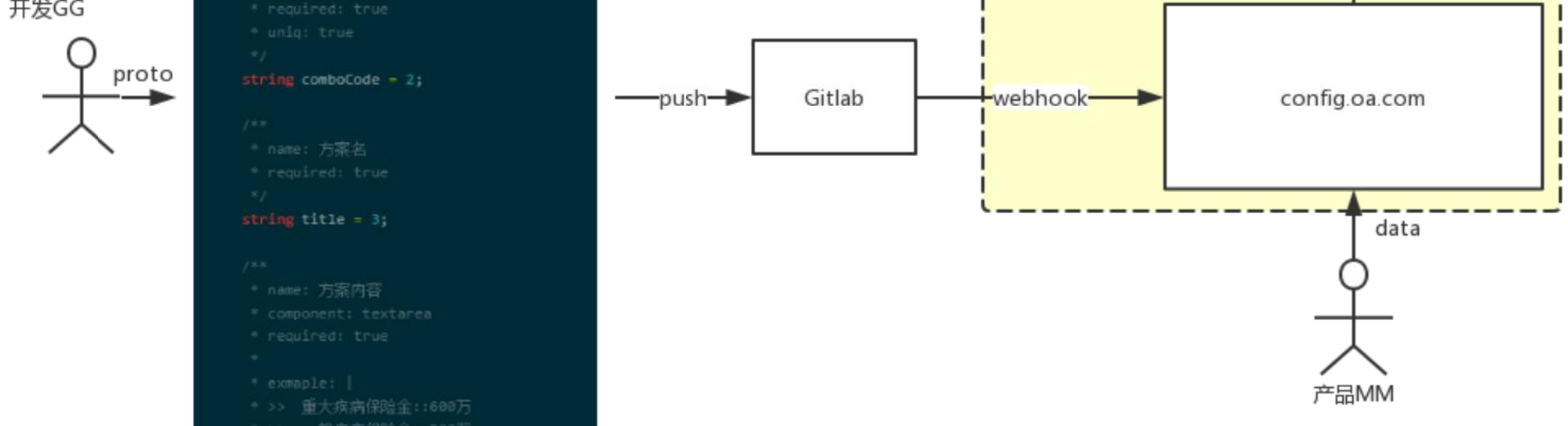
配置平台架构


```
/**
 * name: 保险方案信息
 */
message InsureCombo {
    /**
     * name: 产品代码
     * required: true
     * index: true
     */
    string productCode = 1;

    /**
     * name: 方案代码
     * required: true
     * uniq: true
     */
    string comboCode = 2;

    /**
     * name: 方案名
     * required: true
     */
    string title = 3;

    /**
     * name: 方案内容
     * component: textarea
     * required: true
     *
     * exmaple: |
     * >>  重大疾病保险金::600万
     * >>  一般疾病保险金::300万
     */
    string text = 4;
```



技术架构重点二：小程序体积及加载速度控制

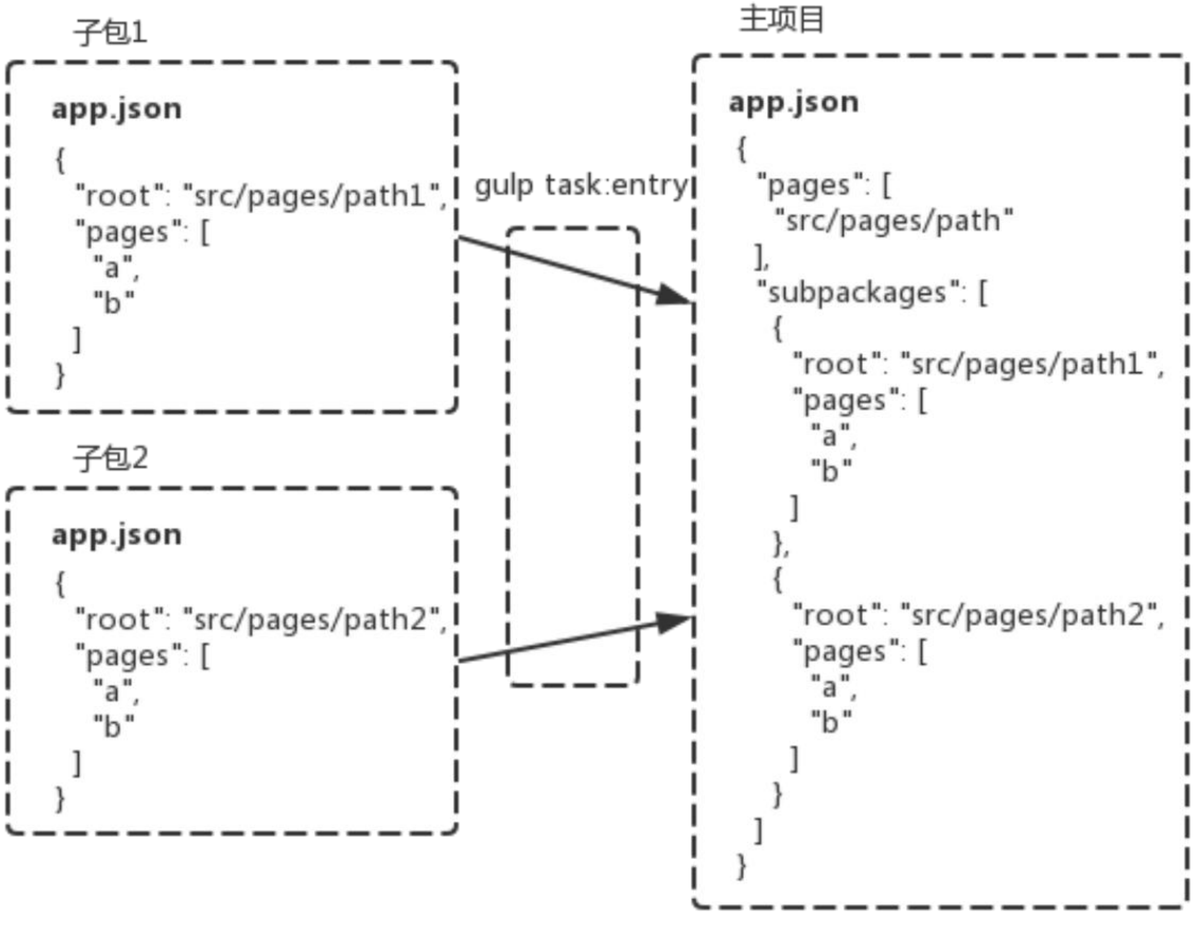
减小体积的方式

小程序分包

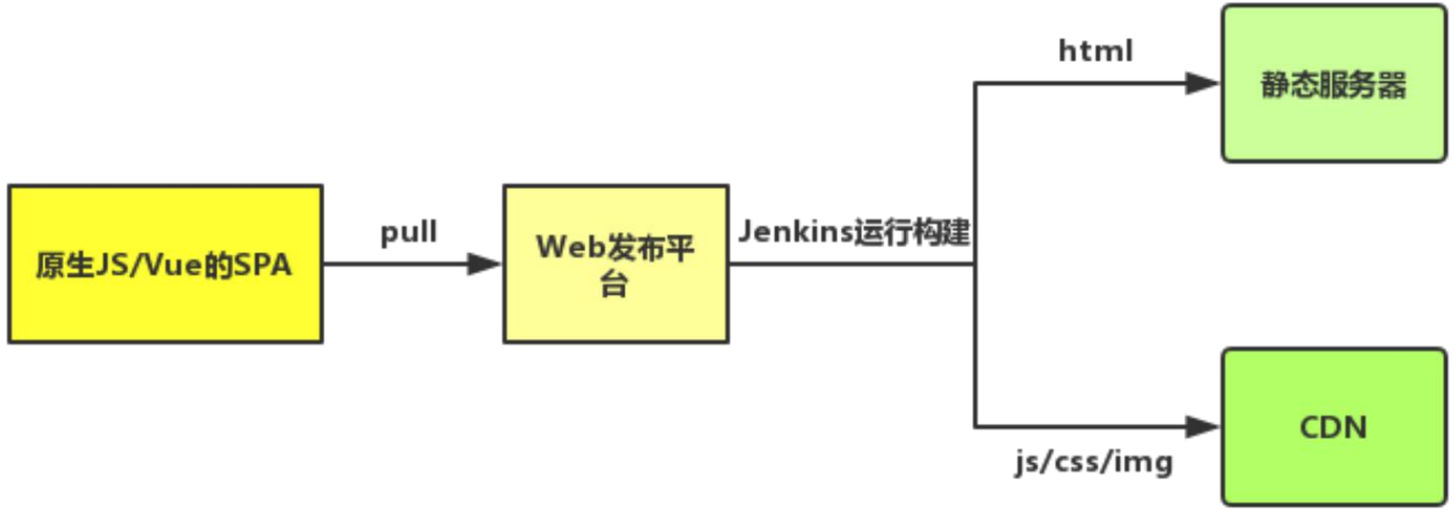
最大限度压缩主包大小，最大限度利用总包大小上限，上传前预压缩

拆分app.json到各分包，在发布时构建合并

生成app.json


子包1
app.json
{
"root": "src/pages/path1",
"pages": [
"a",
"b"
]
}
子包2
app.json
{
"root": "src/pages/path2",
"pages": [
"a",
"b"
]
}
gulp task:entry
主项目
app.json
{
"pages": [
"src/pages/path"
],
"subpackages": [
{
"root": "src/pages/path1",
"pages": [
"a",
"b"
]
},
{
"root": "src/pages/path2",
"pages": [
"a",
"b"
]
}
]
}

运营类的WebView



插件/新小程序

独立发布，且不占用主小程序包体空间，独立开发即可

问题1： 为什么我们没有使用Taro, mpvue等小程序同构框架？

- 并没有很大的多端复用需求
- 小程序并非向前兼容，而框架的维护很难及时跟进新特性的更新
- 小程序特性重度使用，经常触发edge case甚至一些特有问题

问题2： 为什么使用Gulp，不用Webpack？

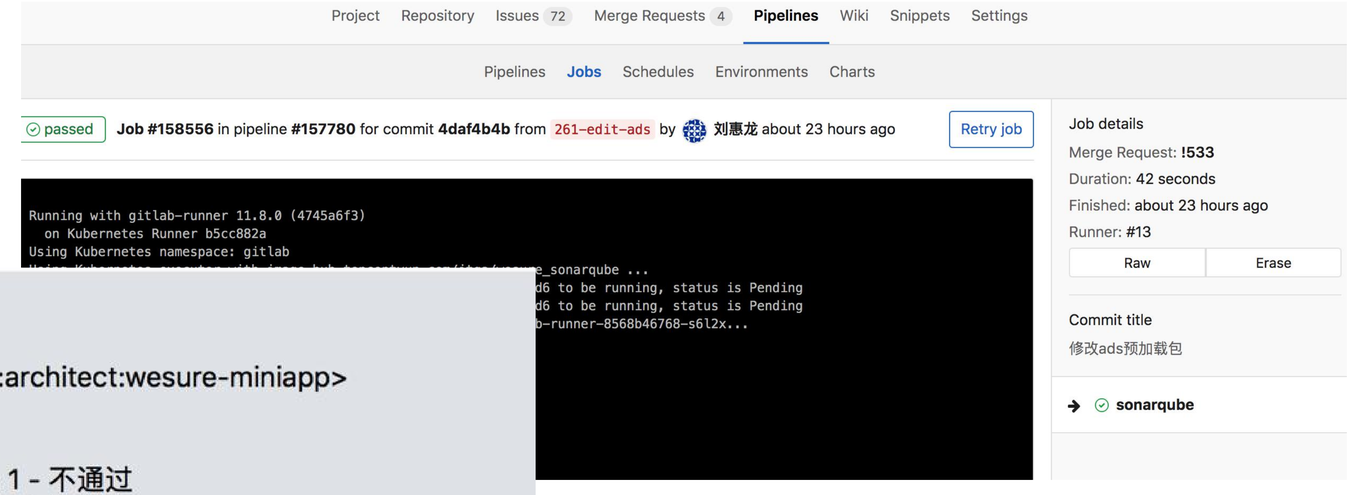
Gulp更适合小程序:

- 构建要做的任务其实很少
- Gulp基于task机制，更简单易用

技术架构重点三：CI的实现

代码检查

Gitlab集成sonar



您提交的代码存在质量隐患，请修正:

项目: AT-web/wesure-miniapp <wesure:architect:wesure-miniapp>

[bug]:blocker 水位线要求:==0 实际数量: 1 - 不通过
[bug]:major 水位线要求:==0 实际数量: 3 - 不通过

导致这次bug的人员(自动忽略非wesure邮箱的): [u'[redacted]@wesure.cn', u'[redacted]@tencent.com', u'[redacted]@wesure.cn', u'[redacted]@wesure.cn']

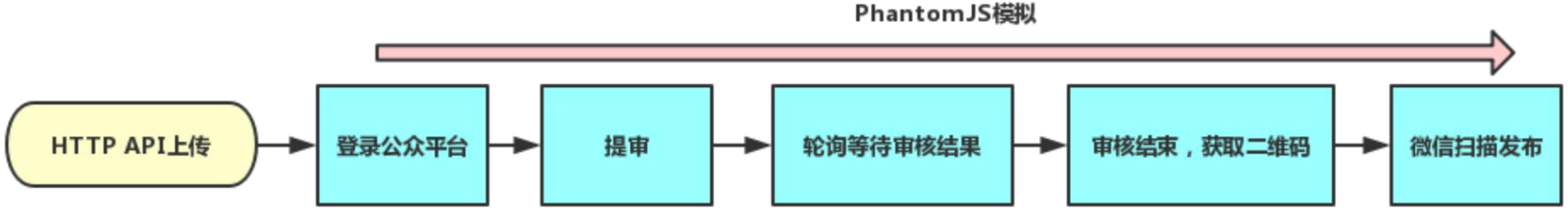
通知抄送: ['[redacted]@wesure.cn', '[redacted]@wesure.cn', '[redacted]@wesure.cn', '[redacted]@wesure.cn']

点击查看(复制到浏览器打开): https://sonar.oa.com/dashboard?id=wesure:architect:wesure-miniapp

自动化上传


使用专用发布账号维持登录态
启动定时上传任务
7月4日 下午 12:16
[小程序发布系统]: 微保小程序UAT2.20.11已上传，待审核, 发布描述: Auto publish.
运行在Windows服务器上
使用开发者工具HTTP接口

自动化提审


PhantomJS模拟
HTTP API上传
登录公众平台
提审
轮询等待审核结果
审核结束，获取二维码
微信扫描发布

为什么自动化发布没有使用Jenkins而采用自研服务？

- 自研平台更容易控制长发布流程中的异常
- 自研平台有更好的交互体验
- 自研平台和Jenkins其实开发成本很接近

小程序开发及项目管理流程的优化

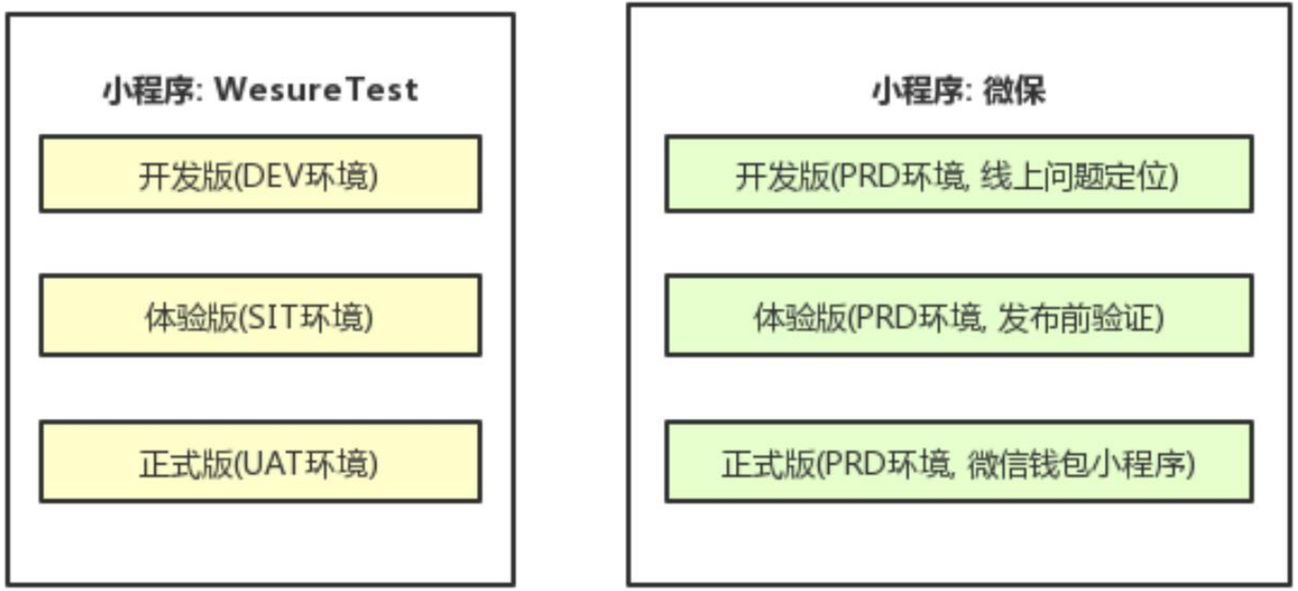
四个环境

DEV: 开发环境

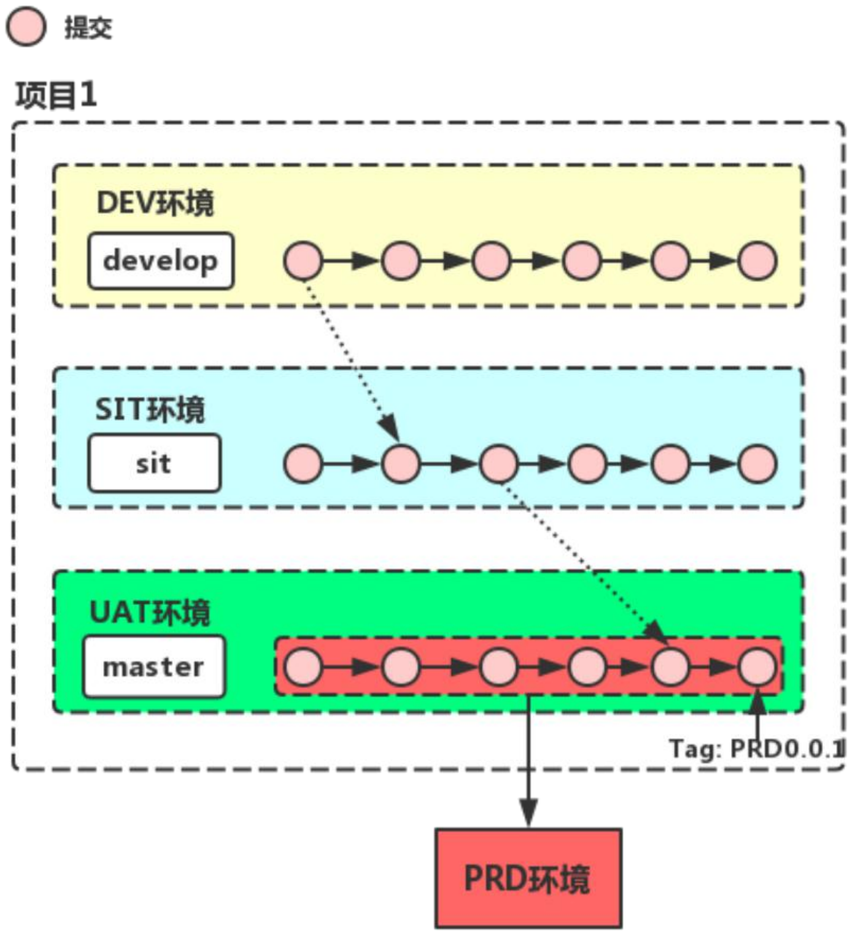
SIT: 集成测试环境

UAT: 体验环境

PRD: 生产环境



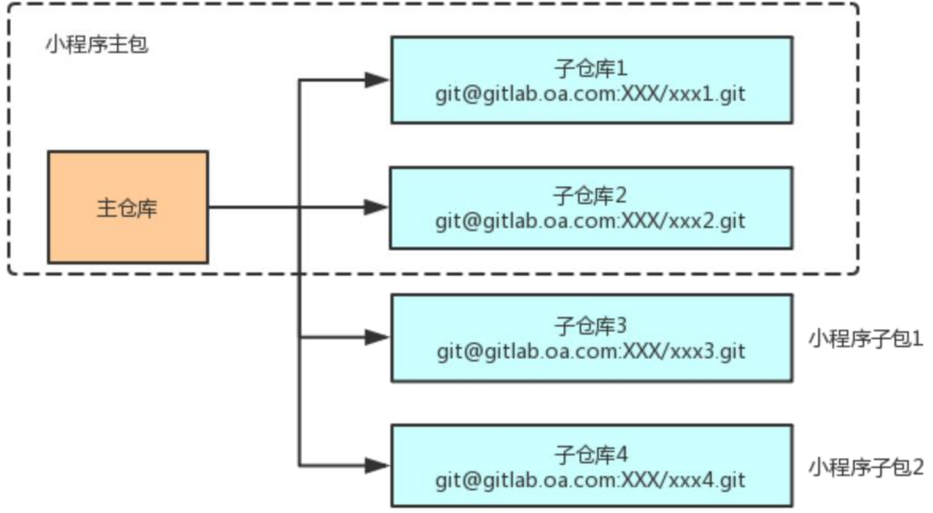
环境和分支的对应


提交
项目1
DEV环境
develop
SIT环境
sit
UAT环境
master
Tag: PRD0.0.1
PRD环境

仓库管理

1 使用自定义Gulp task配合Git做父子仓库管理

2 做分包及Git子仓库的逻辑映射



为什么没有使用Lerna或者Git submodule?

- Lerna:

子目录下大量配置文件，这些文件会被打进小程序包，需要较多的额外处理

- Git submodule:

会在父仓库产生关联子仓库的版本管理文件，在父仓库内产生大量无效commit

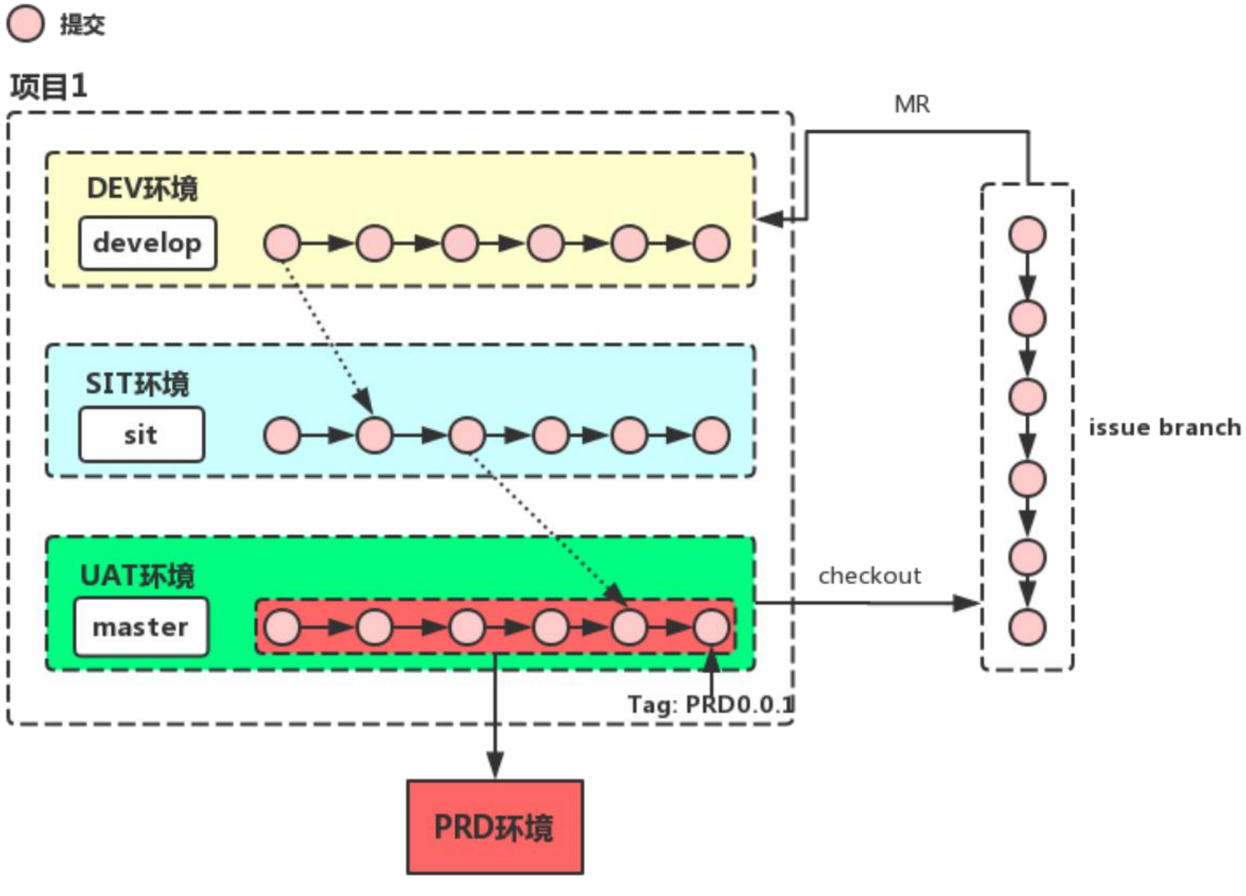
GitHub Flow

主（master）分支上的任何内容都要保证是可部署的。

特性开发

- 新建issue
- 从master分支拉出特性分支，分支号与issue关联
- 业务特性开发，完成后Push代码到Gitlab
- 提Merge Request到目标分支

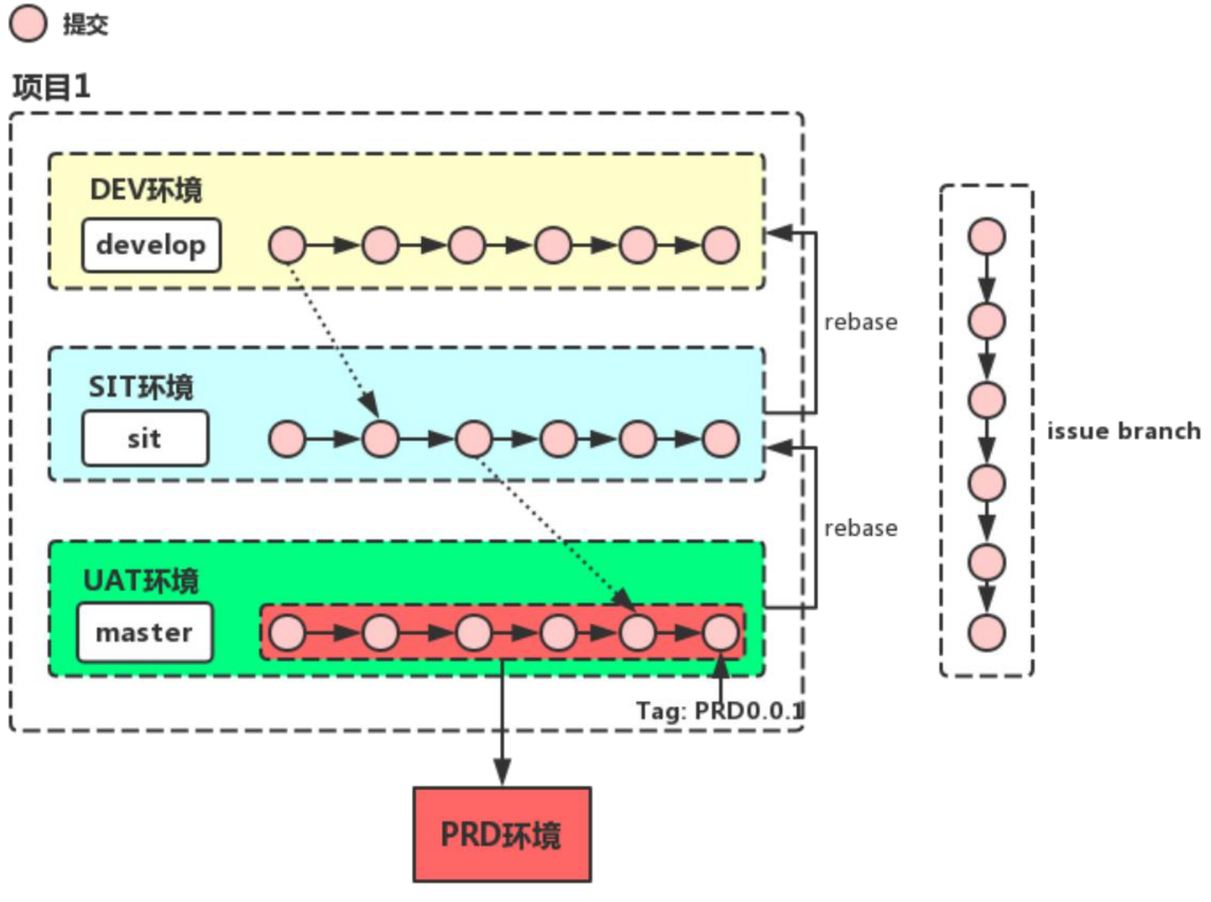
特性开发


提交
项目1
MR
DEV环境
develop
SIT环境
sit
UAT环境
master
checkout
issue branch
Tag: PRD0.0.1
PRD环境

特性开发

- 发布完成后，由发布系统(定时任务)自动打Tag
- 由发布系统处理分支的Git rebase操作

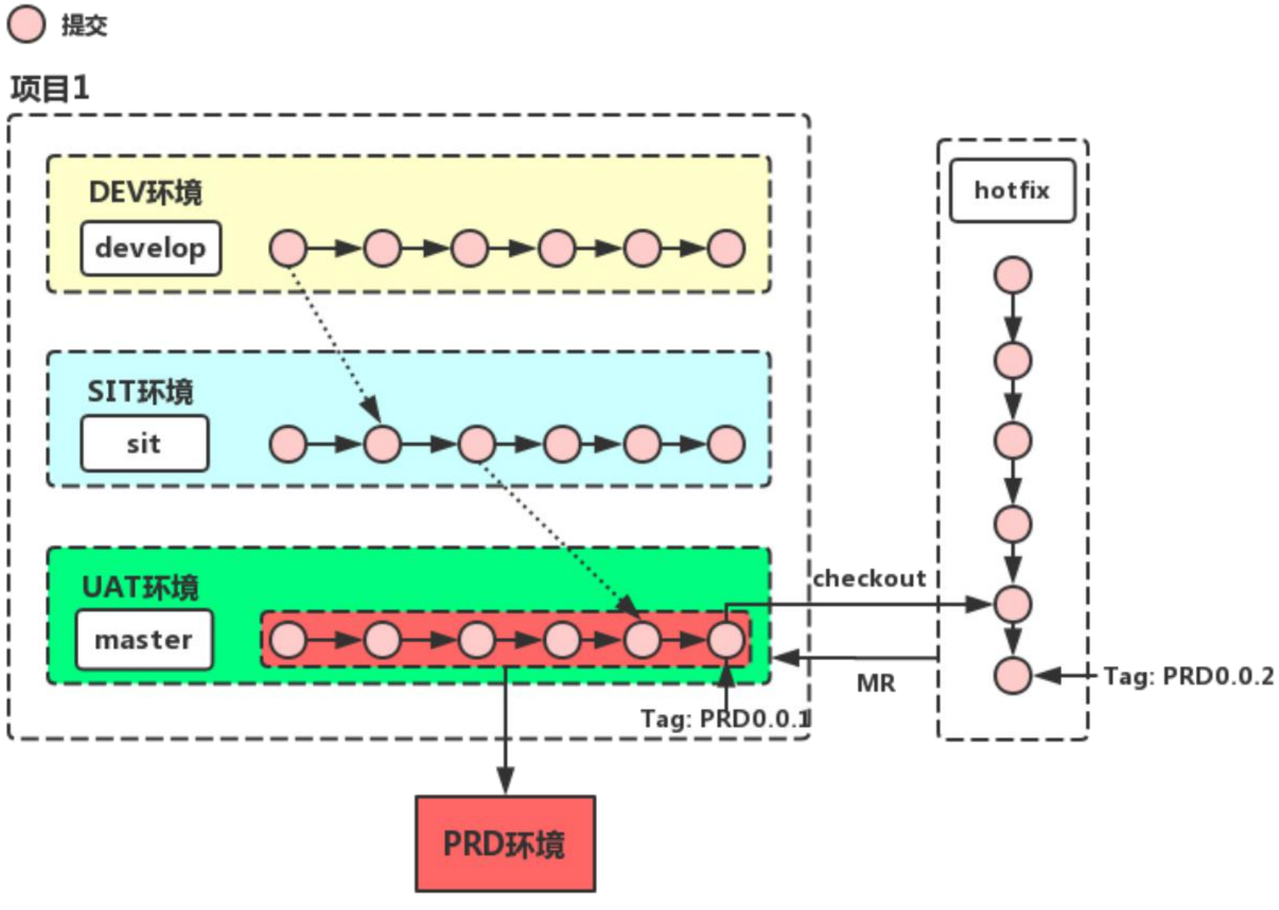
特性开发


提交
项目1
DEV环境
develop
SIT环境
sit
UAT环境
master
rebase
rebase
issue branch
Tag: PRD0.0.1
PRD环境

紧急发布

- 从当前PRD Tag拉出hotfix分支
- 修复，提MR，合并
- 上传后，在小程序的体验版中验证
- 验证通过，发布。将发布内容向前合并

紧急发布

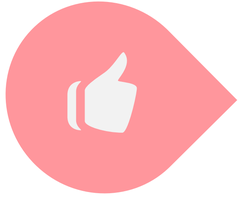

提交
项目1
DEV环境
develop
SIT环境
sit
UAT环境
master
hotfix
checkout
MR
Tag: PRD0.0.1
Tag: PRD0.0.2
PRD环境

通过以上，我们做到了……

双周常规迭代与单周紧急迭代并行

50+有交互差异、体验出色的保险产品维护

线上的非框架结构性问题可在5分钟内处理完毕



一年内线上0阻断型Bug

我们期望的未来

自动化测试

更强大的WebView

向前兼容的架构设计、由大版本发布breaking changes

开放更多引流方式

Thank you for your attention!

By Brook Zhao