

如何构建后现代前端工程化开发体系

# 自我介绍

- 李成熙 (heyli)
- 腾讯AlloyTeam
- webpack中文社区负责人
- docschina文档社区运营负责人
- Github: <https://github.com/lcxfs1991>

# 收获到什么

- 前端开发关注哪些目标
- 如何利用前端开发体系完成这些目标
- 对初学者：了解到整个前端开发流程
- 对进阶者：了解如何设计前端开发的体系

# 一个小经历



## 小团队开发窘境

- 1 – 10人
- 流程简陋
- 没基础设施

产出少，速度慢，质量差，成本高

# 我们关注什么

我们的目标

# 多快好省

提高开发效率  
提升页面性能



产出多，速度快，质量好，成本低

# 标准背后

是制度与体系的保障



# 开发体系是什么

百度百科：指若干有关事物或思想意识互相联系而构成的一个整体。

我的理解：完整性、协同性



# 前端开发体系及其系统构成



# 小团队如何搭建开发体系

成长为架构师的必经之路

# 开发环境

环节一：脚手架与命令行、组件化、接口联调

# 技术选型

## 框架

react  
vue  
jQuery

## 页面 复杂度

redux/vuex  
router

## 页面场景

长列表  
富文本  
滑动手势  
自适应

## 构建工具

gulp  
webpack

## 脚手架与命令行

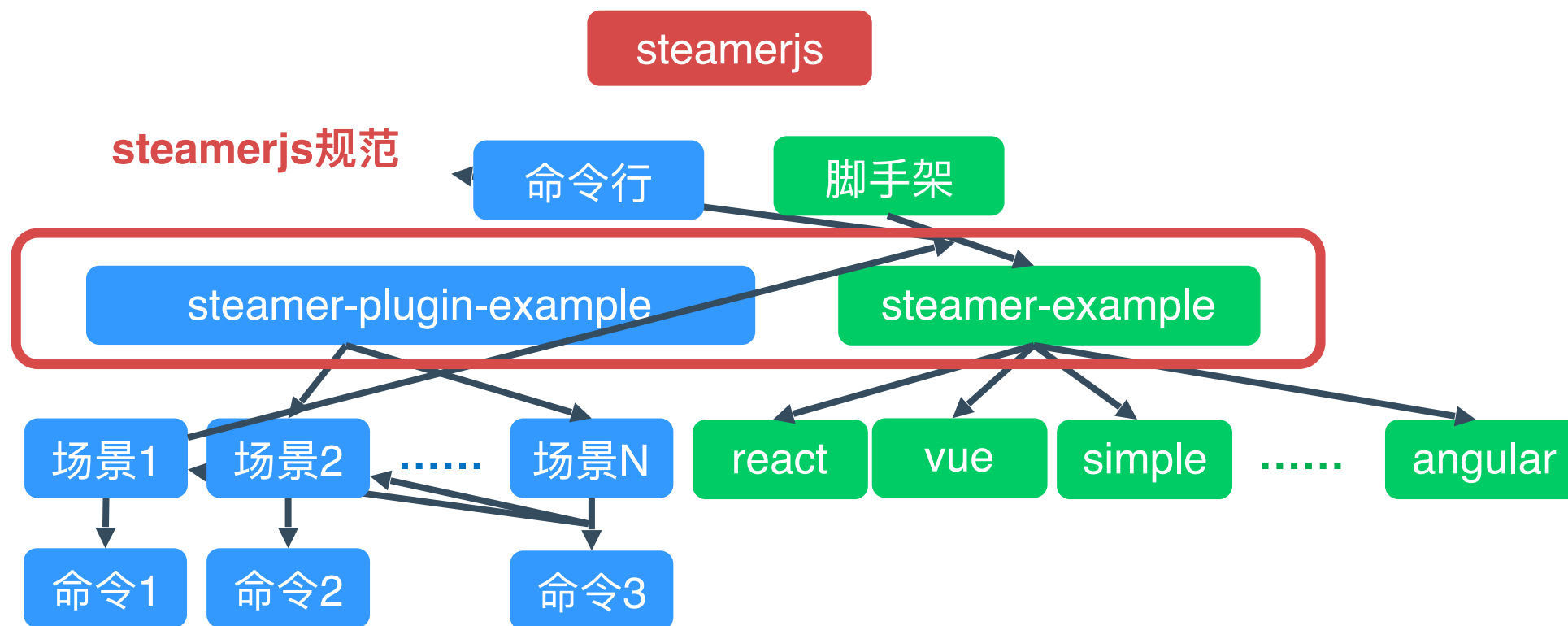
- create-react-app? vue-cli?
- 快速启动
- 构建逻辑复杂难理解
- 构建性能一般
- 两者体验完全不同

# 统一脚手架与命令行





# 脚手架与命令行



# 脚手架搭建考量的5大要素

- 快速部署与更新
- 团队约定
- 快速配置
- 页面性能
- 构建性能

# 1. 脚手架-快速部署与更新

- 即装即用的插件式机制

```
npm i -g steamerjs
```

```
npm i -g steamer-plugin-kit
```

```
npm i -g steamer-react
```

```
steamer kit
```



# 1. 脚手架-快速部署与更新

## 一键生成项目

```
Lis-MacBook-Pro:test root# steamer kit
? which starterkit do you like: (Use arrow keys)
  Local installed Starter Kits:
> simple: alloyteam frameworkless starterkit
  Other official Starter Kits:
  react: alloyteam react starterkit
  vue: alloyteam vue starterkit
  simple-component: alloyteam frameworkless component development starterkit
  react-component: alloyteam react component development starterkit
  vue-component: alloyteam vue component development starterkit
```

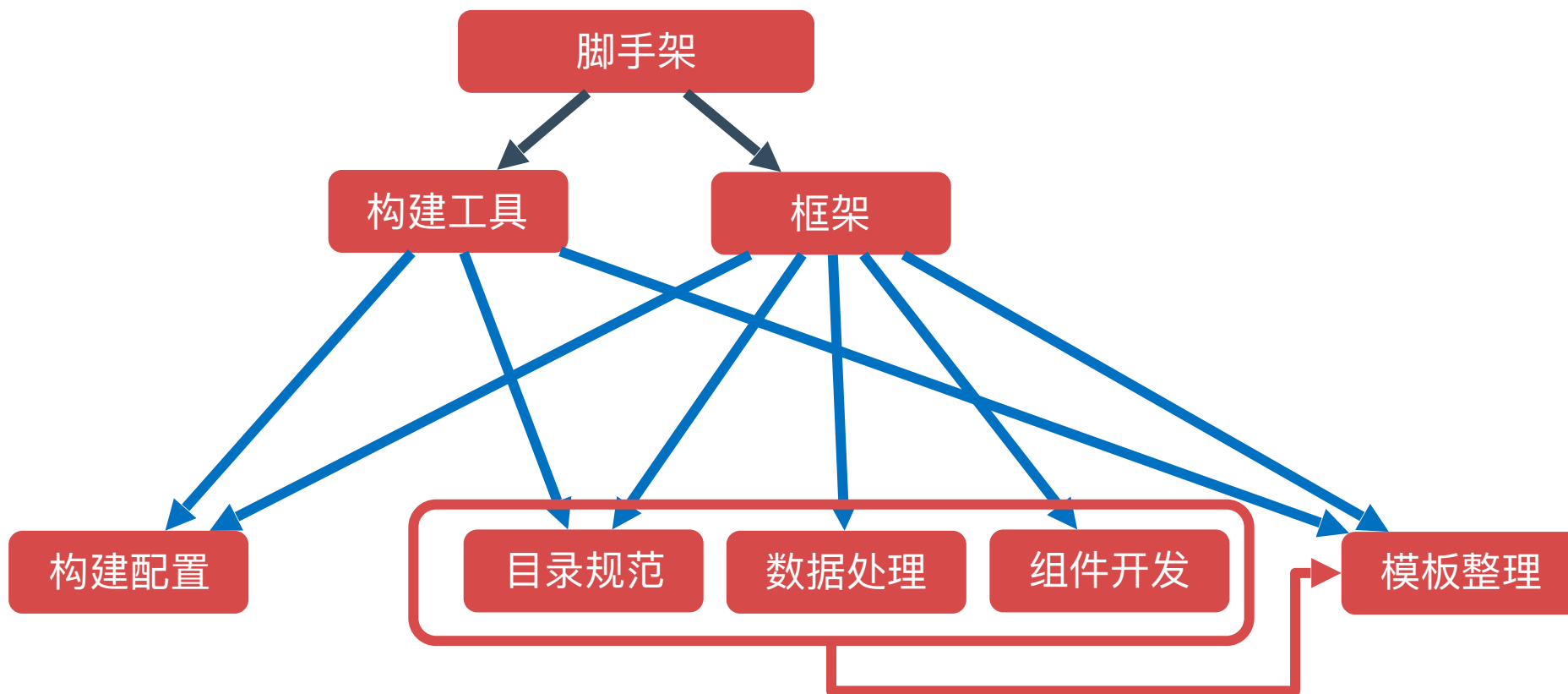
## 一键基于模板生成页面

```
Lis-MacBook-Pro:react-test root# steamer kit -t
? which template do you like: (Use arrow keys)
> list
  preact-list
  react-mobx
  simple
  spa
  ts
```

## 一键更新脚手架

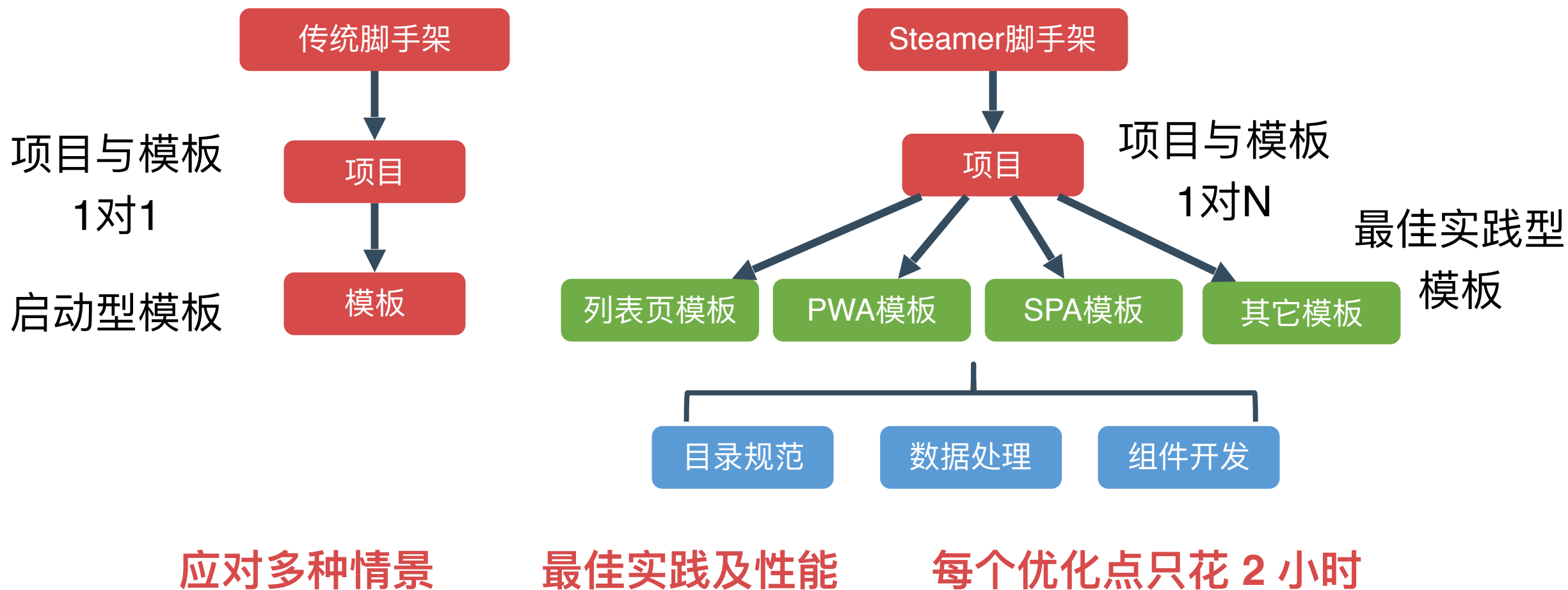
```
Lis-MacBook-Pro:react-test root# steamer kit -u
steamer-react update success
```

## 2. 脚手架-团队约定



## 2. 脚手架-团队约定

- 模板积淀



### 3. 脚手架-快速配置

- 快速配置

1. 压缩
2. 合图 (not retina & retina)
3. 样式 (css, sass, less, stylus)
4. 模板 (ejs, pug, handlebars)
5. Sourcemap (cheap, inline-cheap, ...)

查文档



自助配置



### 3. 脚手架-快速配置

- 优化初始node\_modules包

分拆依赖

1. Feature 特性依赖
2. Template 模板依赖

185.9MB 56% → 125.5MB



## 4. 脚手架-构建性能

- 优化性能
  - 只构建要修改的页面
  - 并行构建
  - 外链体积较大类库
  - 预编译稳定部份
  - 缓存
  - 减少路径搜索时间
  - 摒弃webpack-stream

### webpack Performance: The Comprehensive Guide #15

 Open lcxf1991 opened this issue on 26 Oct 2016 · 8 comments



lcxf1991 commented on 26 Oct 2016 • edited

Owner + 👤 ✎

Shared from

#### Introduction

For beginners, webpack building performance is sometimes annoying. However, this will not stop webpack from becoming the best web building tool in the world. Once you master the following skills you can soon boost the building performance and save your team a lot of time.



# webpack

## 4. 脚手架-构建性能

10个大型React页面，代码丑化 66秒 => 20秒

**生产环境全量构建  
30% - 40%提升**

10个大型React页面，5秒 => 0.4秒

**开发环境增量构建  
90%以提升**

## 5. 脚手架-页面性能

- 从构建层面做好该做的优化，保证页面性能达标（md5, 合图，压缩等）

## 5. 脚手架-页面性能

- 同构直出方案



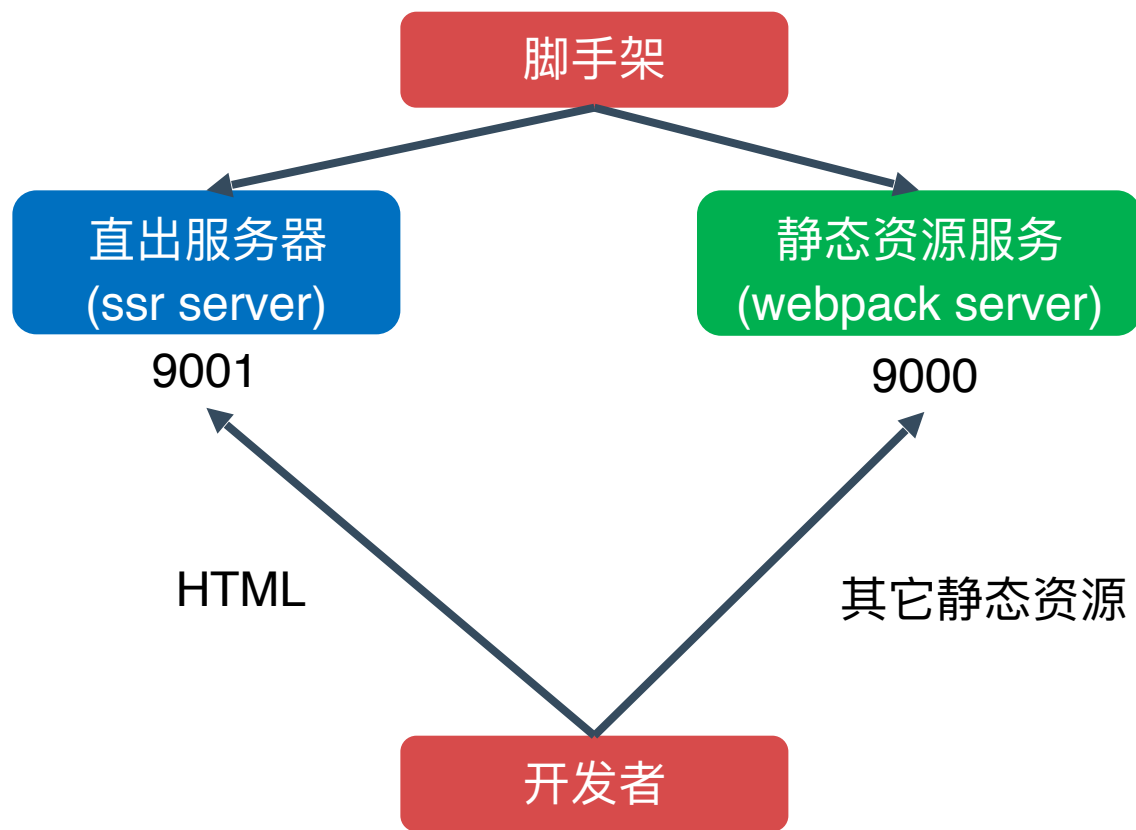
- 高度集成的同构直出框架

## 5. 脚手架-页面性能

- 服务器成本的考虑
  - 有的页面不需要直出
  - 容错的考虑
- 
- `steamer-react/steamer-vue & steamer-koa`

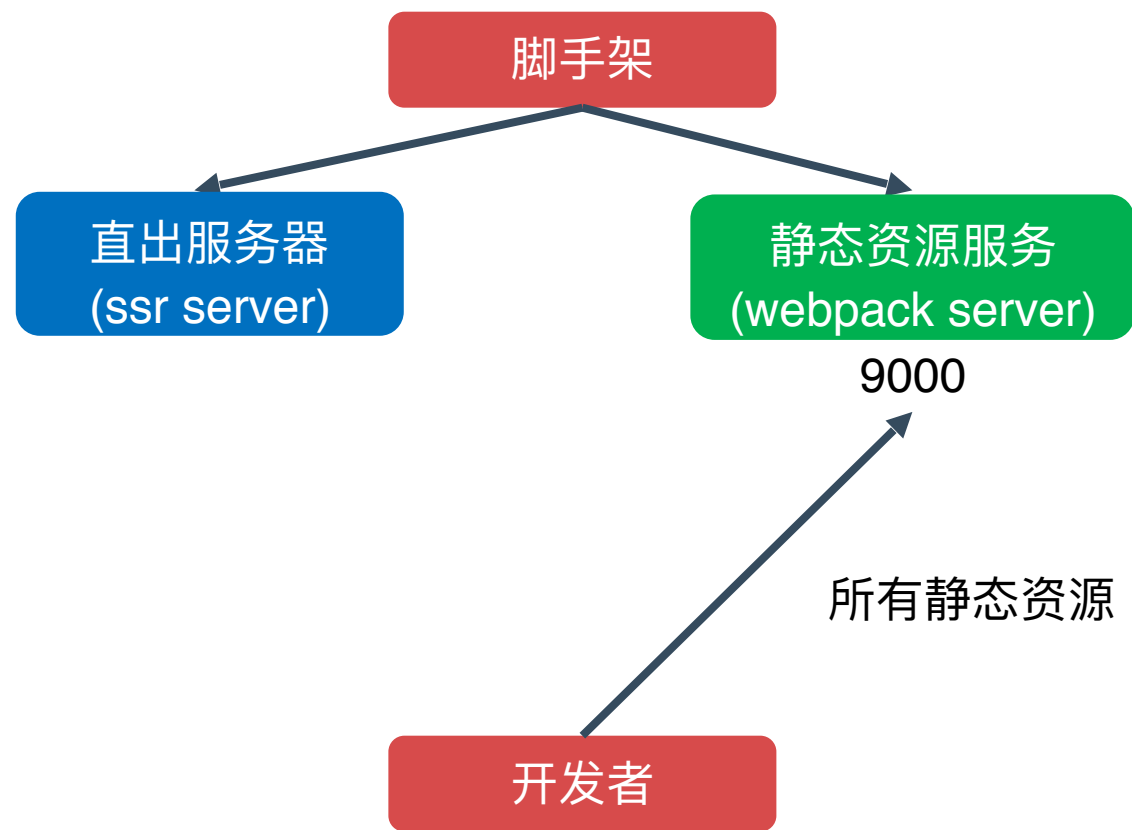
## 5. 脚手架-页面性能

- 整合直出框架



统一构建

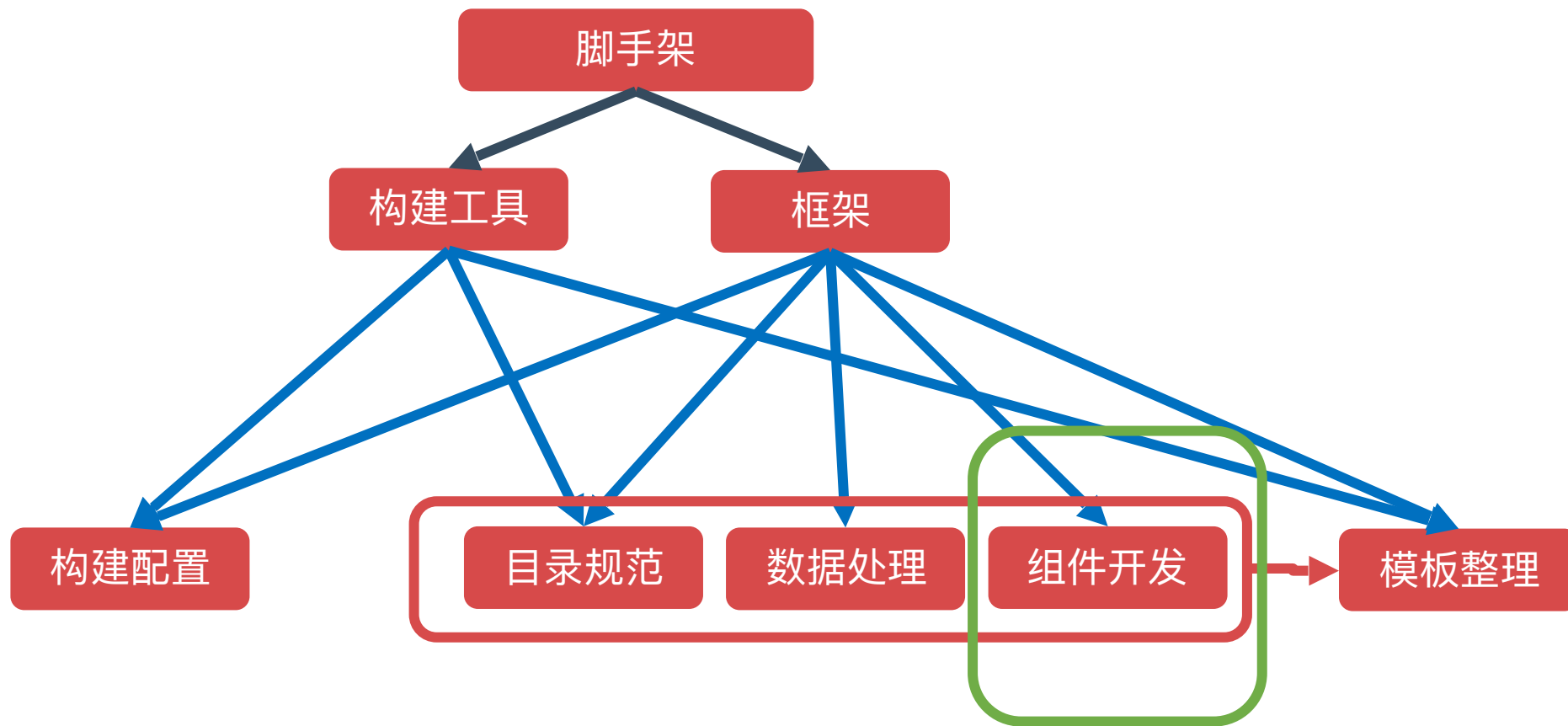
并行开发



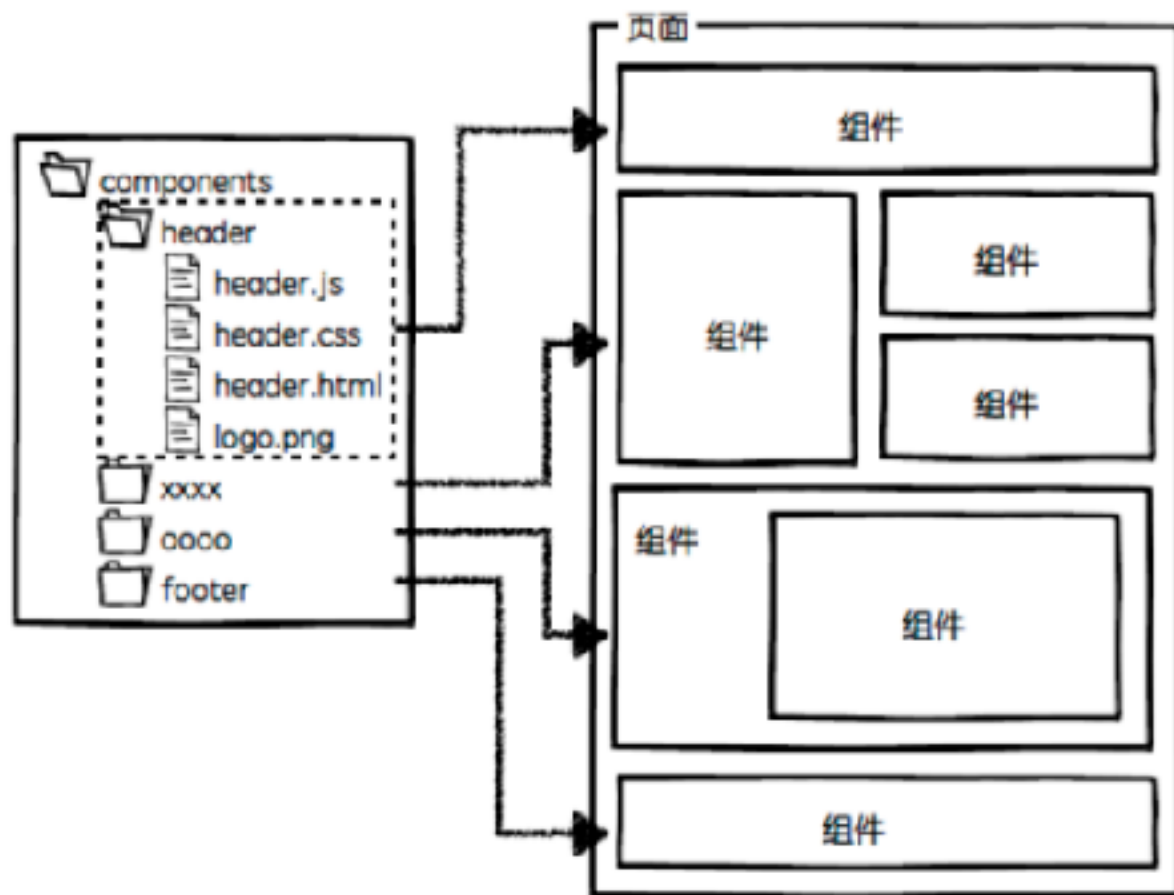
0天时间，全民直出



# 组件化



# 组件化





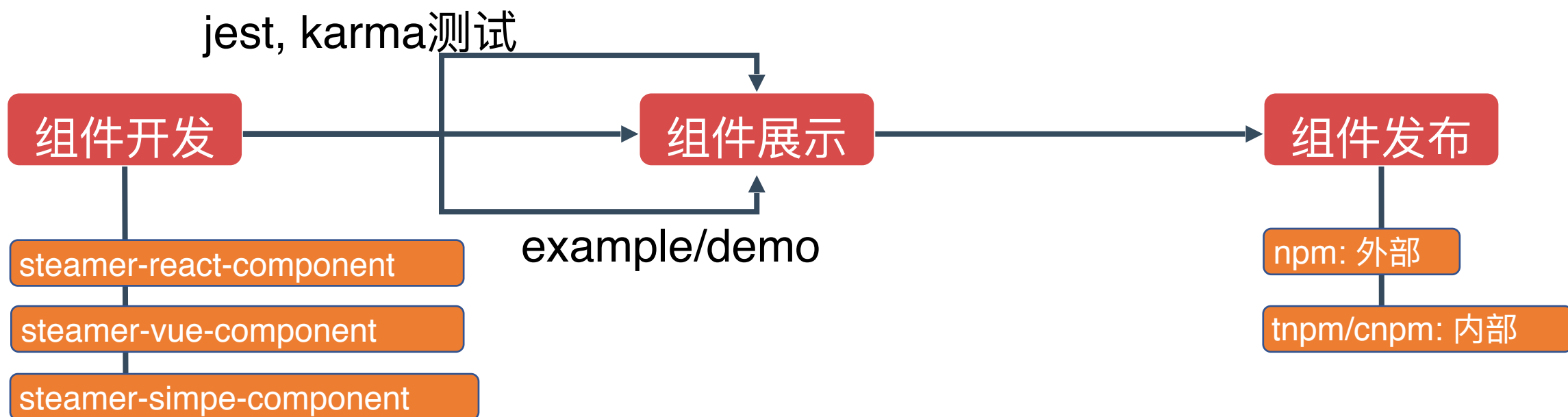
# 组件化

## 项目内组件开发及使用



# 组件化

## 跨项目组件开发及使用



目录，命令，测试，打包方式等

规范 自动 专业

# 组件化

## 组件库开发及使用



- 更复杂的脚手架
- 整体引入 + 单个组件引用
- 国际化

# 接口联调

- 需求： 定义数据结构 + 假数据测试 + 记录接口文档
- 推荐： postman
- 线上服务： [easy-mock.com](https://easy-mock.com)
- 本地服务： steamer-plugin-mock

# 开发环境与生产环境衔接

## 环节二

# 数据上报与错误监控

- 难点：数据并发量 + 数据处理 + 服务器运维
- 整合：SDK
- 取代方案：
  - 产品：百度统计 or 腾讯统计
  - 性能：wetest
  - 报错：sentry.io

# 持续集成

- 开源项目：Github + Travis-CI
- 私有项目：Bitbucket + Pipeline or Gitlab + Pipeline

	公有仓库	私有仓库	持续集成
Github + Travis	免费	收费	免费
Bitbucket + Pipeline	免费	免费	免费
Gitlab + Pipeline	免费	免费	收费
Gitlab Community	免费	免费	免费

# 持续集成





# 持续集成 - 代码规范

- 更推荐的做法 IDE 提醒 + Pre-Commit 检查
- JavaScript (ESLint) + CSS (StyleLint)
- eslint-config-alloy

# 持续集成 – 测试

- 组件测试 (UI组件 & 逻辑组件)
- 业务测试 (UI)
- 推荐: 持续集成 (多Node版本) + 兼容性测试 (Chrome, Firefox等)



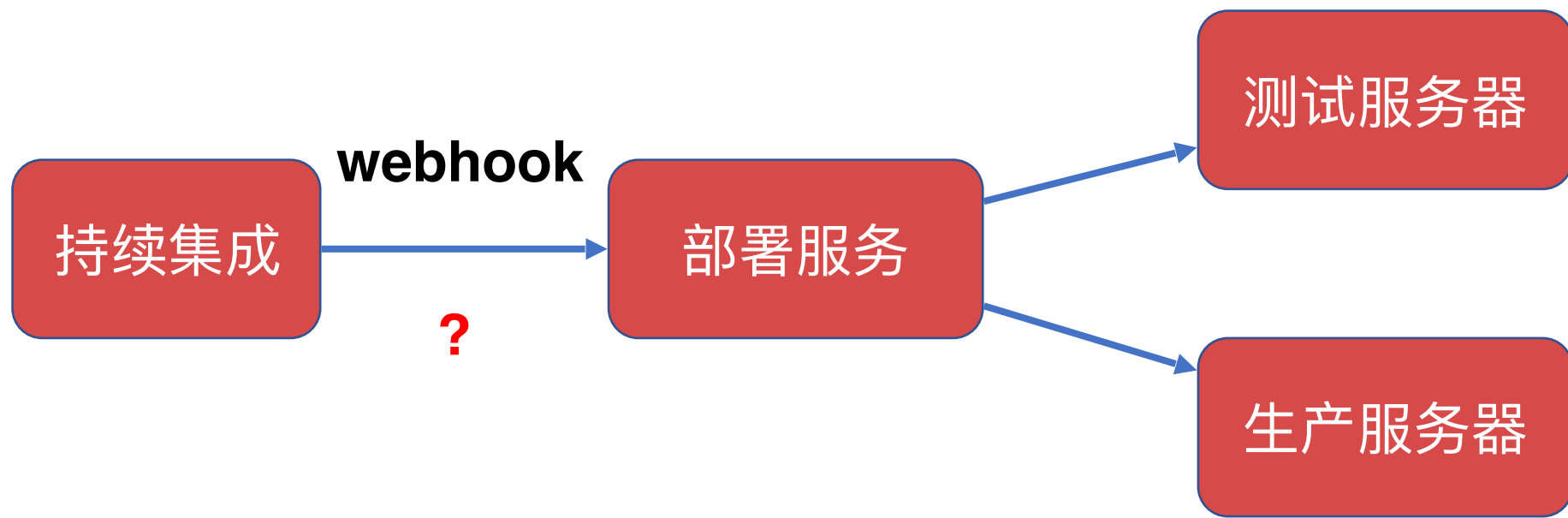
李CHENGXI · 6月前



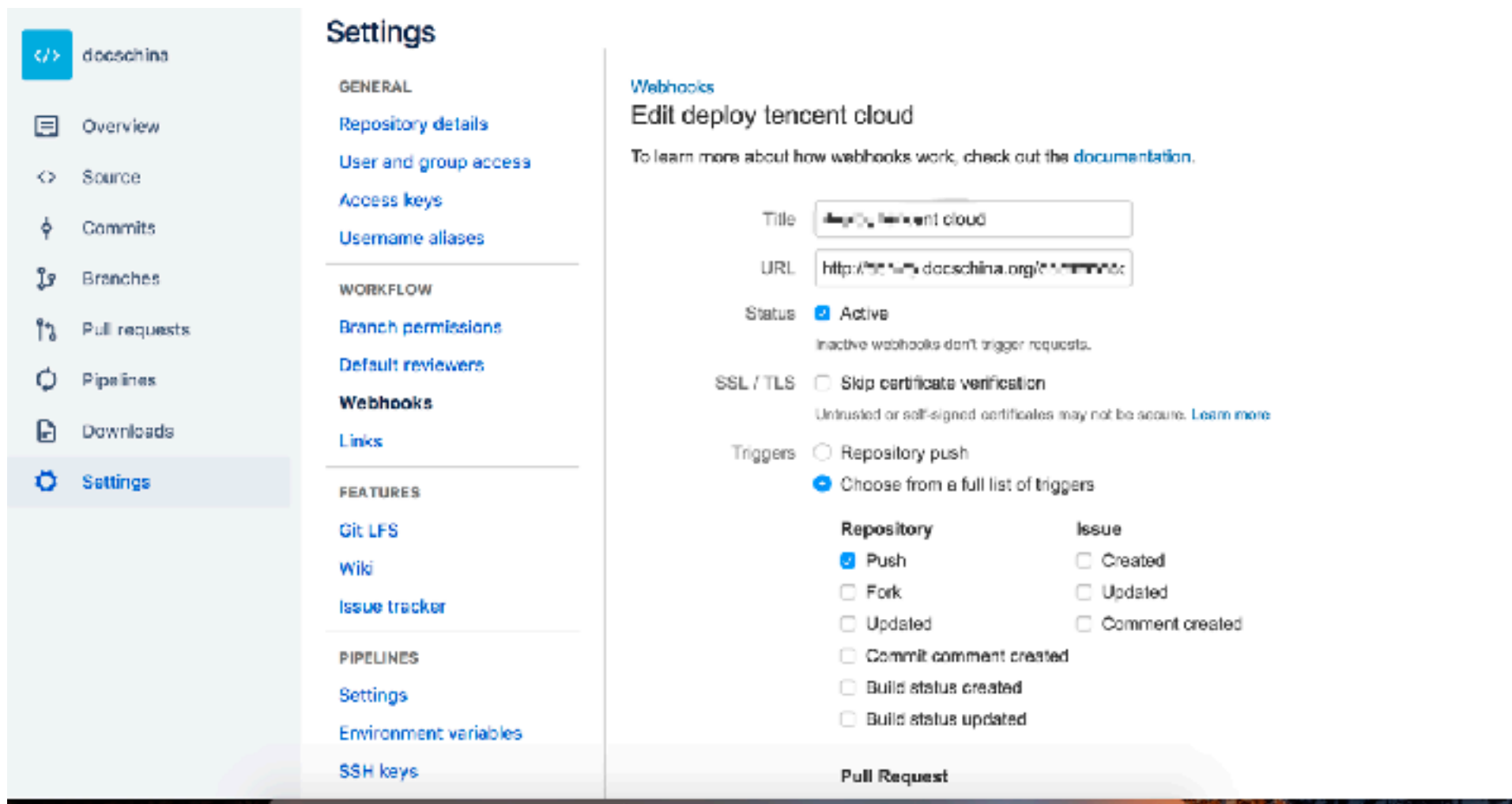
## 从工程化角度讨论如何快速构建可靠React组件

[原文链接](#) 前言 React 的开发也已经有2年时间了, 先从QQ的家校群, 转成做

# 持续集成 - （部署）衔接



# 持续集成 - （部署） 衔接



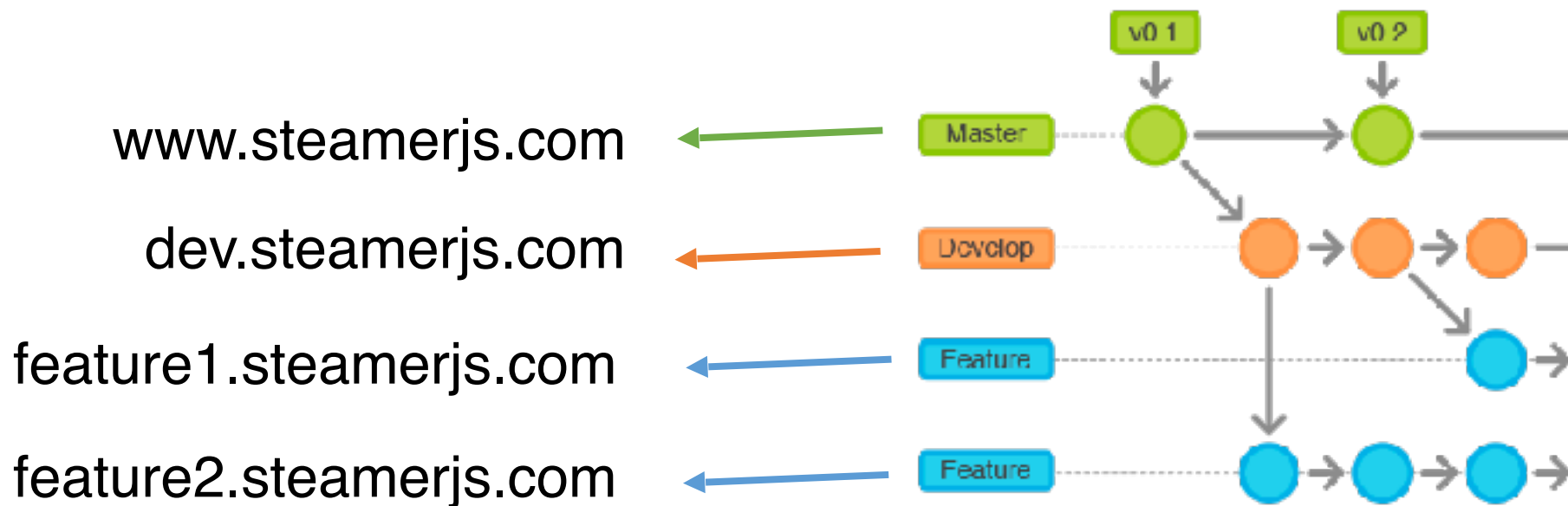
The screenshot displays the GitHub repository settings for a user named 'doeschina'. The left sidebar contains navigation links: Overview, Source, Commits, Branches, Pull requests, Pipelines, Downloads, and Settings (which is highlighted). The main content area is titled 'Settings' and is divided into several sections: GENERAL (Repository details, User and group access, Access keys, Username aliases), WORKFLOW (Branch permissions, Default reviewers), Webhooks (with a 'Links' sub-link), FEATURES (Git LFS, Wiki, Issue tracker), PIPELINES (Settings, Environment variables, SSH keys), and a partially visible 'Pull Request' section at the bottom.

The 'Webhooks' section is expanded, showing the configuration for a webhook titled 'deploy to tencent cloud'. The URL is set to 'http://gitlab.doeschina.org:8080/'. The status is 'Active'. Under the 'Triggers' section, 'Choose from a full list of triggers' is selected, revealing a grid of options:

Repository	Issue
<input checked="" type="checkbox"/> Push	<input type="checkbox"/> Created
<input type="checkbox"/> Fork	<input type="checkbox"/> Updated
<input type="checkbox"/> Updated	<input type="checkbox"/> Comment created
<input type="checkbox"/> Commit comment created	
<input type="checkbox"/> Build status created	
<input type="checkbox"/> Build status updated	

# 测试环境部署

- 多域名单测试环境

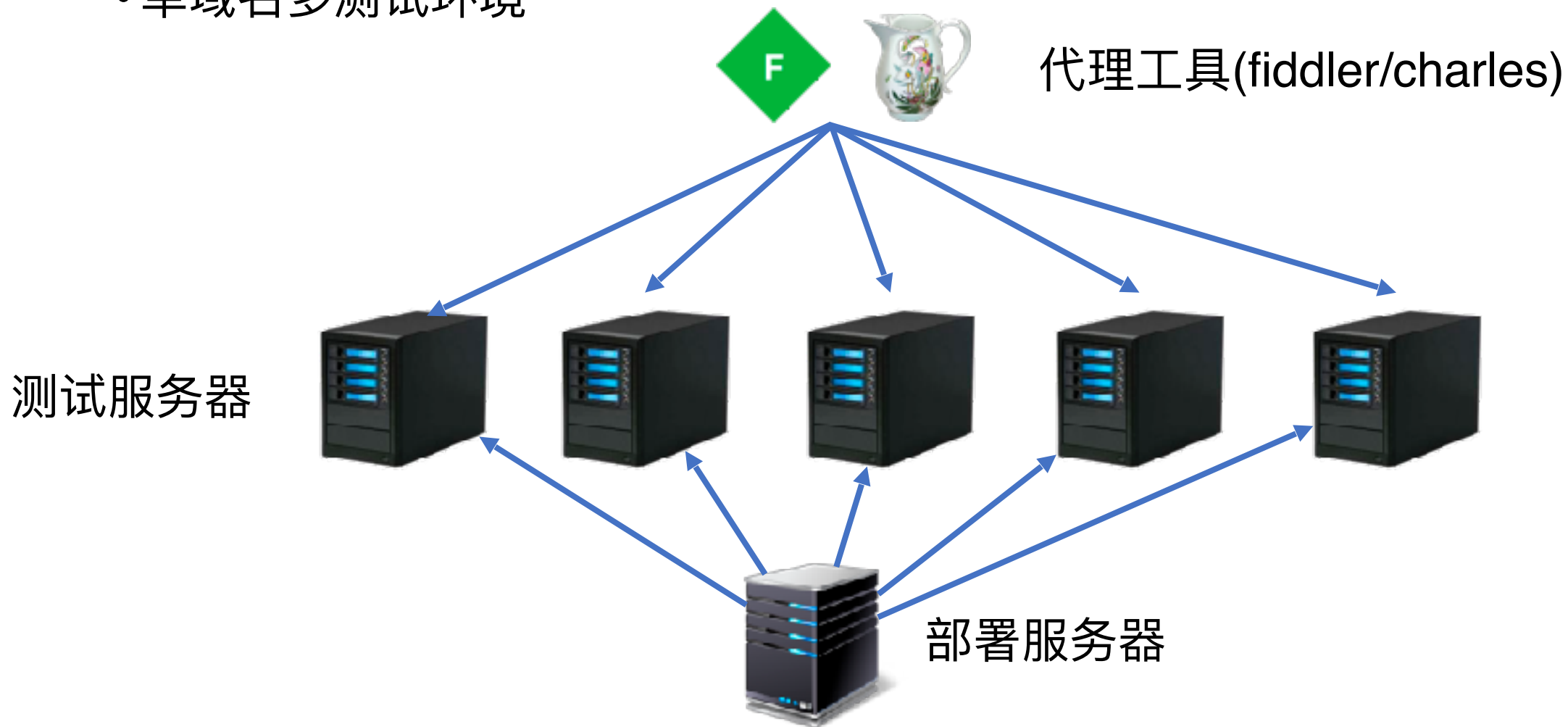


下发Nginx配置



# 测试环境部署

- 单域名多测试环境



# 测试环境部署

测试部署方式	优势
多域名测试环境	<ol style="list-style-type: none"><li>1. 对于纯Web站，不需要代理工具，测试便利</li><li>2. 节省服务器</li></ol>
单域名多测试环境	<ol style="list-style-type: none"><li>1. 对于APP内嵌页面，可以方便测试</li><li>2. 各测试环境独立，互不干扰</li></ol>

# 生产环境

## 环节三



# 发布上线

## • 印记中文

印记中文

最权威的中文开发文档

文档分类

Node.js

构建工具

工具库


Web框架

代码质量


css/JS框架

翻译指南

翻译群组




Node.js

 nodejs

基于V8引擎的JS运行环境


翻译

构建工具

 webpack


前端资源打包工具

翻译 文集 群组 社区


 rollup

新一代JavaScript模块打包器

翻译


 gulp

基于流的自动化构建工具


 grunt

JavaScript世界的构建工具

工具库


 jQuery


高效精简强大的工具库


 Zepto


轻量级的工具库

Web框架

 koa koa

 express

 Vue

 React

# 发布上线

- 印记中文

1	节点地区	CPU	内存	CDN	硬盘	预算
2	广州	4	8GB	2mbps	30GB	5000
3	香港	2	2GB	2mbps	20GB	2000
4	香港	4	8GB	8mbps	40GB	8000
5						

¥ 15000

# 发布上线

- 印记中文

brianlin(林楠) 2017-08-25 13:35:24

通过cvm来部署静态文档成本不划算的

heyli(李成熙) 2017-08-25 13:35:27

如果后续接他们的动态，要拆分出来

brianlin(林楠) 2017-08-25 13:35:42

静态放cos+cdn

动态走cvm

**¥ 1300**

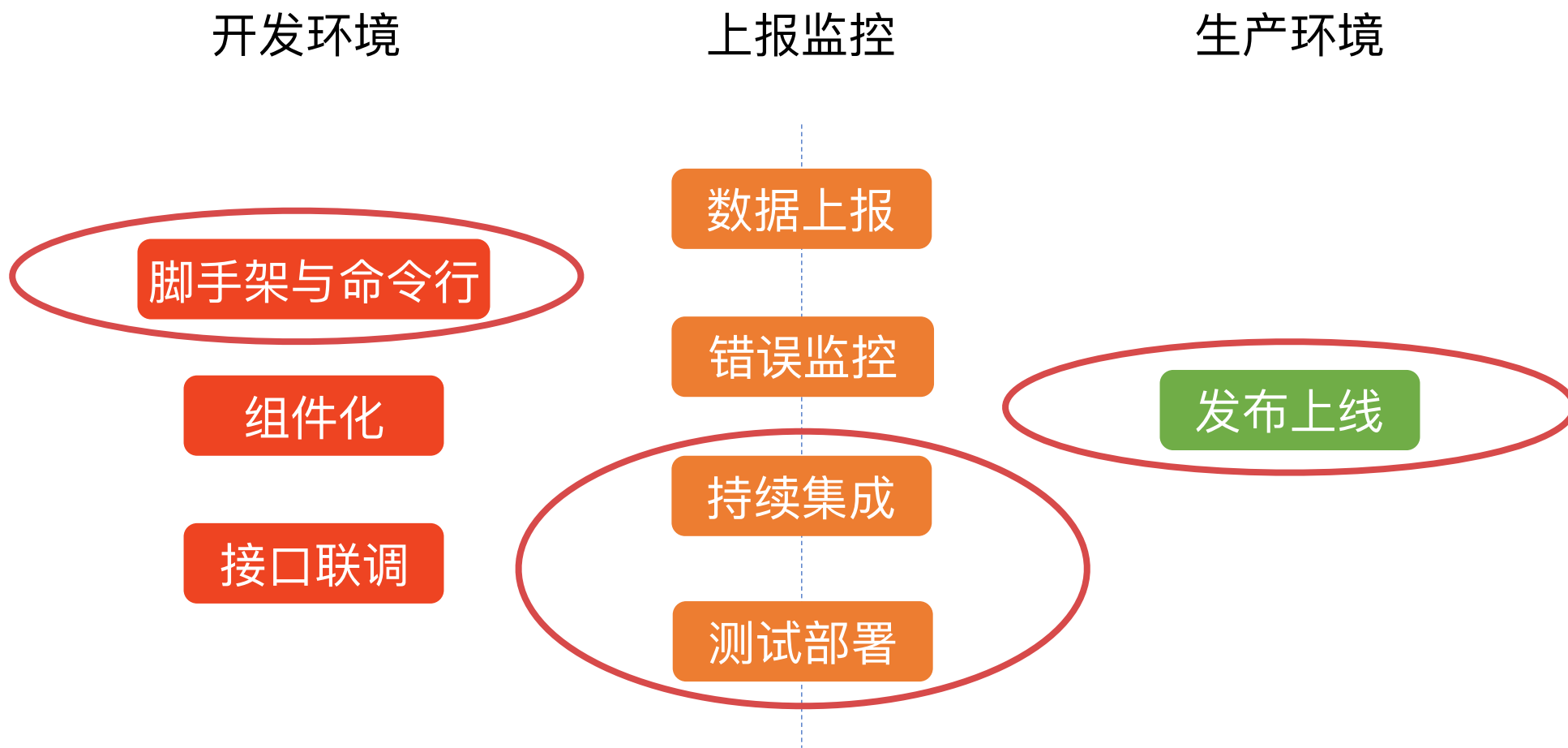
# 发布上线

技术	优势	发布方式
CDN加速 + 对象存储 (COS)	静态资源 (节省30% - 40%的带宽成本) 大文件	Webhook + Git + COS上传脚本
云虚拟服务器 (CVM)	数据 后台服务 部署更灵活	Webhook + Git + 后置 脚本 (重启)

# 小结

最后的最后

# 前端开发体系及其系统构成





如有谬误，恳请斧正