

第十期

杭州 NodeParty × Rokid

技术分享



基于 Docker 容器的多平台统一打包服务

万宁邦

丁香园前端技术专家

自我介绍

万宁邦

2018 年加入丁香园 - 技术产品事业部 - 前端平台架构组

全栈开发，目前致力于丁香园跨端方案的演进，和 Node 基础设施的建设



万宁邦 Nicolas

浙江 杭州



扫一扫上面的二维码图案，加我微信

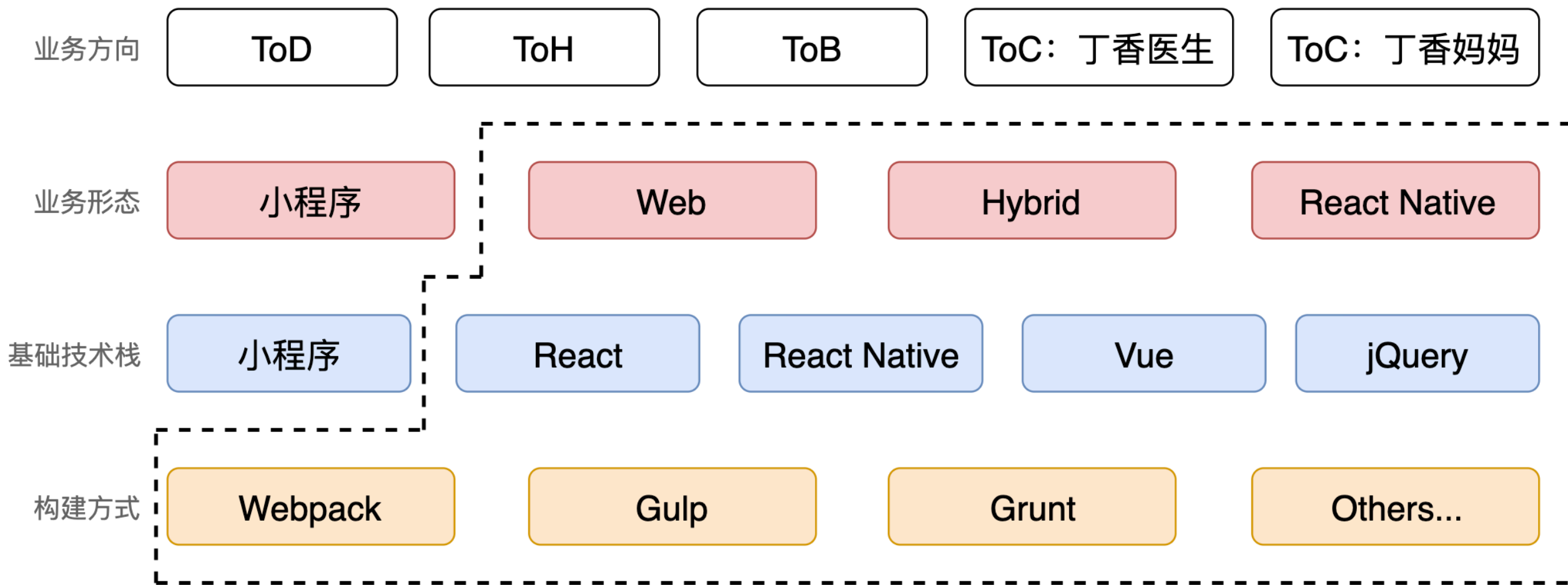
目录

- ① 发现问题
- ② 解决问题
- ③ 未来展望

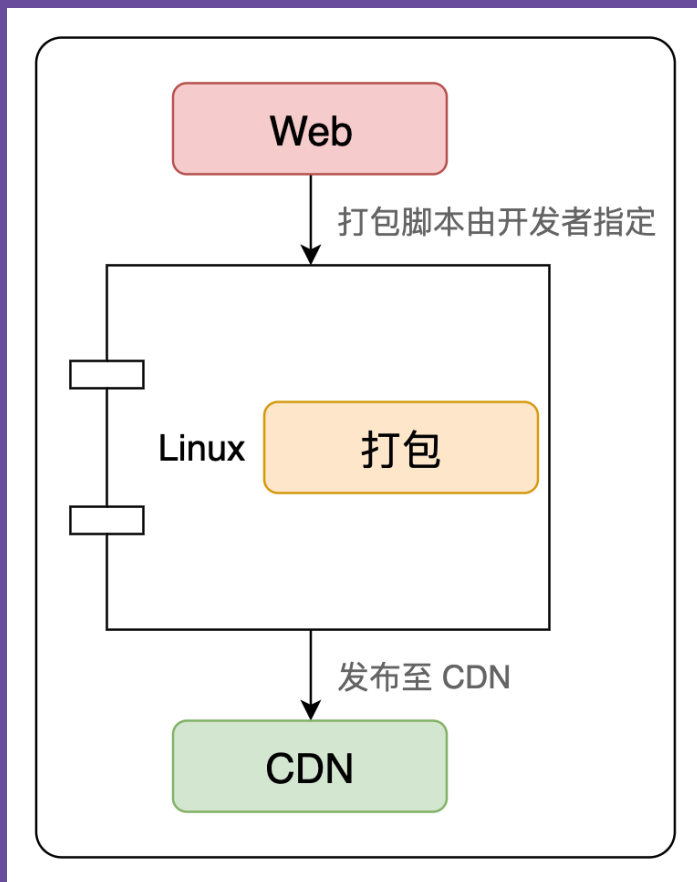


发现问题

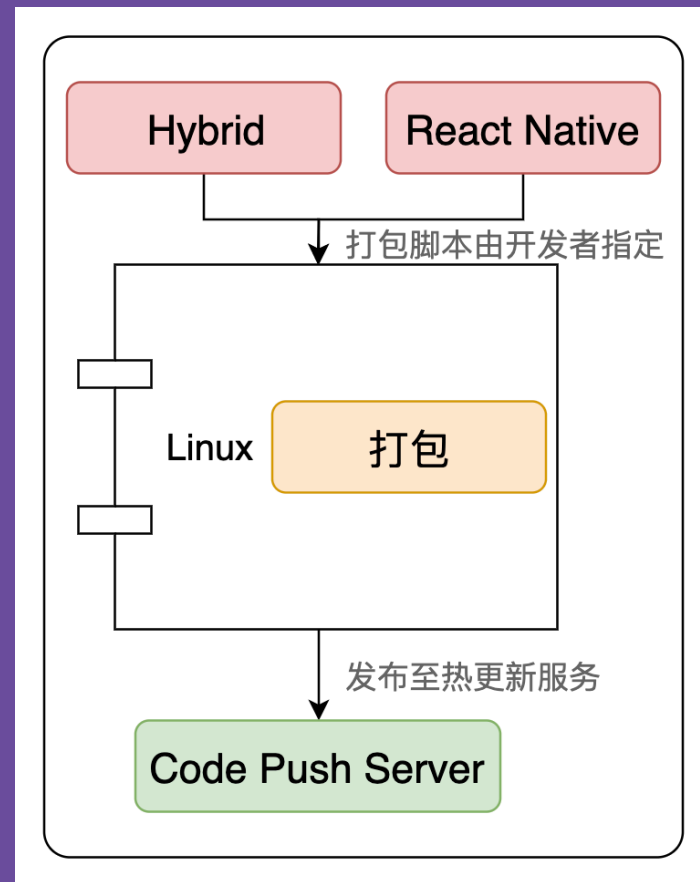
丁香园的前端业务形态



多发布系统并存



Assets 发布系统



Code Push 发布系统

应用打包存在和潜在的问题



安全性



副作用



资源占用

开发者的心声

打包又失败了，就不能稳定一点吗？

2

解决问题

开发者的需求

打包产物 = \mathcal{F} (项目, 构建脚本)

将打包过程放在 Docker 容器中

- 良好的隔离性
- 快速的启动效率
- 用完即停即删



打包过程的高可用性？

- 打包任务低频但连续
- 公司体量越来越大，同一时刻的打包任务数增多
- 打包任务不需要排队
- 容器快启快停，用完即删，能否做到高可用？

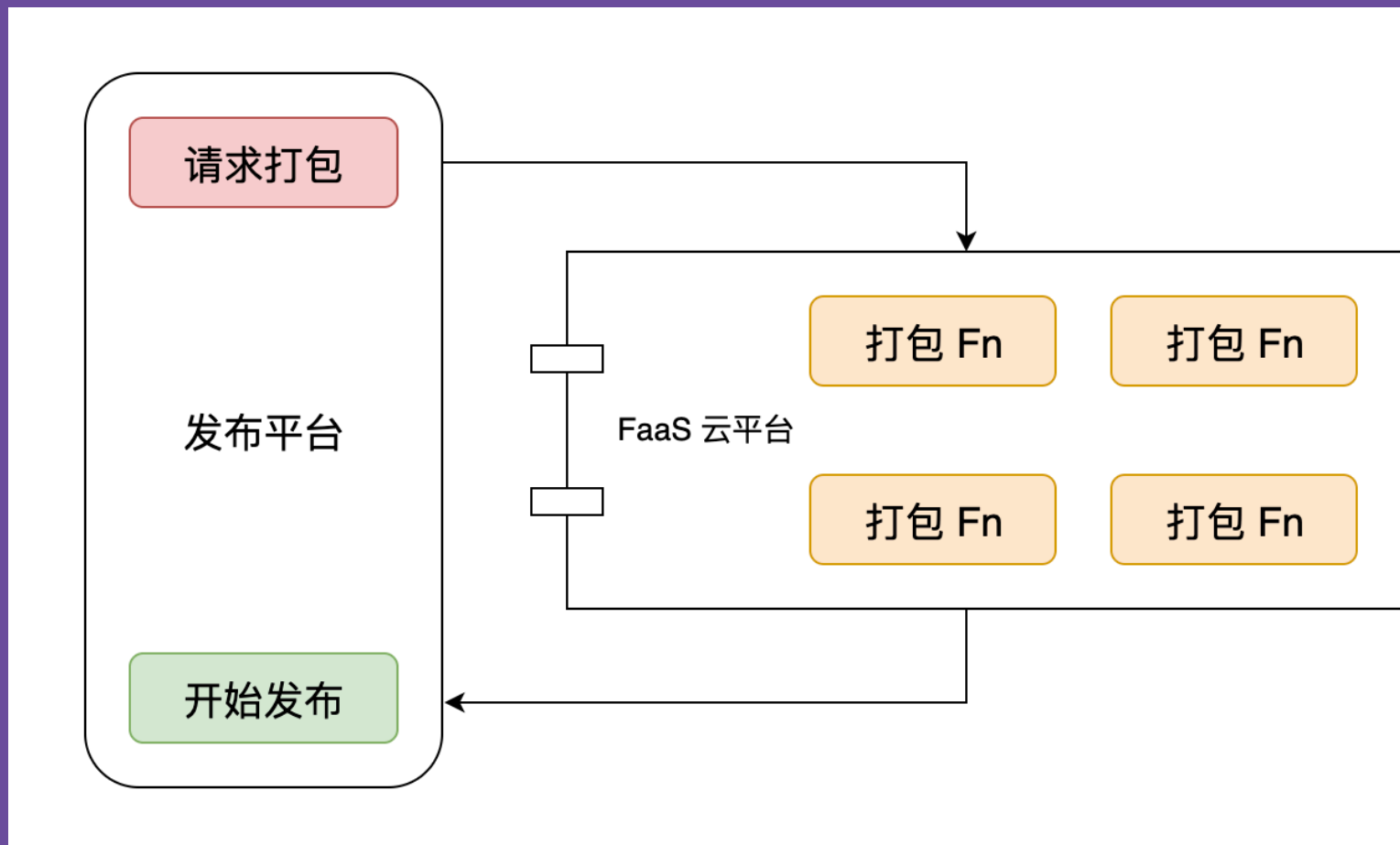
Serverless 或许是最好的解决方案

What is serverless computing?

Serverless computing refers to the concept of building and running applications that **do not require server management**. It describes a finer-grained deployment model **where applications, bundled as one or more functions**, are uploaded to a platform and then **executed, scaled, and billed in response to the exact demand needed at the moment**.

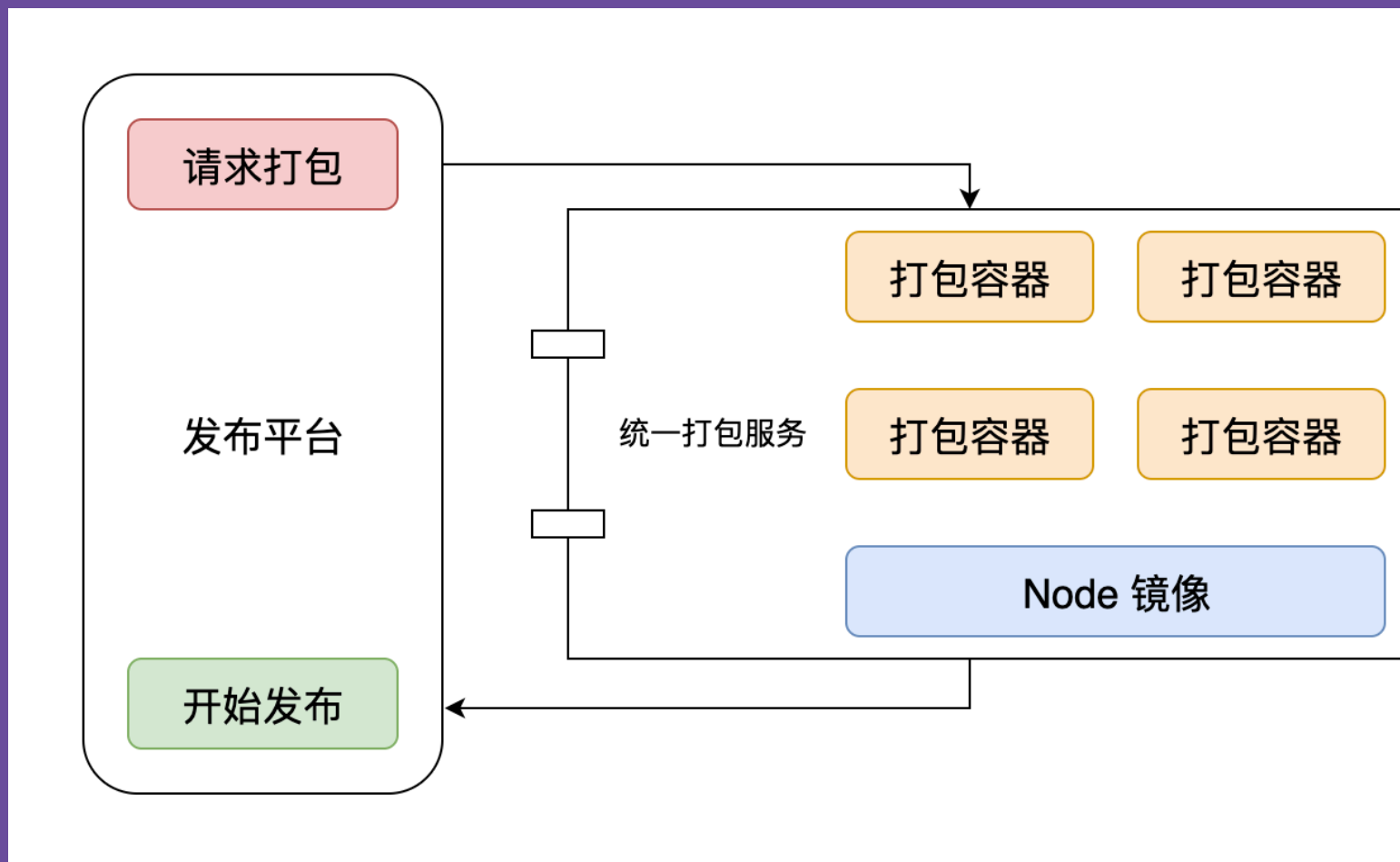
Serverless 或许是最好的解决方案

- Node 是首选语言
- 可以用阿里云函数计算等云平台
- 被运维大佬否决
- 自建 Serverless ?



回到 Docker 方案

- 服务仍然部署在 Linux 机器
- 需要限制容器资源



Docker 容器资源限制



内存限制



CPU 限制

Node or Docker Bug ?

- OS : `os.cpus().length` 实际上是宿主机的 CPU 数
- Node modules : 大部分皆使用该 API 来决定进程的 fork 数
- 因此决定不做 CPU 限制
- 详情见 Node Issue [#28762](#) & [#28855](#)

Dockerfile

```
1 FROM node:10.15.3
2
3 # 安装 yarn 命令
4 RUN npm config set registry https://registry.npm.taobao.org/ && \
5   npm i -g yarn
6
7 # 使用 apt 代理提高更新速度
8 RUN echo 'Acquire::http {Proxy "http://deb.host.dxy:3142"; };' \
9   > /etc/apt/apt.conf.d/01proxy
10
11 # 安装 zip 命令
12 RUN apt-get update && \
13   apt-get install -y zip
14
15 # 初始化 www-data 用户并授权 /var/www & /tmp/cache 目录
16 RUN usermod --shell /bin/sh --uid 1001 www-data && \
17   mkdir -p /var/www /tmp/cache && \
18   chown www-data:www-data /var/www /tmp/cache
19
20 # 拷贝 pack 命令行工具
21 COPY ./bin/pack.sh /usr/local/bin/pack
22
23 # 切换至 www-data 用户
24 USER www-data
25
26 # 配置 npm registry 并设置 npm 为非生产环境
27 RUN npm config set registry https://registry.npm.taobao.org/ && \
28   npm config set spin false
29
30 # 指定容器运行脚本
31 ENTRYPOINT ["/usr/local/bin/pack"]
```

容器启动命令

```
46     child = spawn('docker', [  
47         'container',  
48         'run',  
49         '--rm',  
50         `--memory=${memory}`,  
51         '--name',  
52         name, // 指定容器名  
53         '-v',  
54         `${name}:/tmp/cache`, // 指定 node_modules tar 包缓存目录  
55         '-v',  
56         `${packageDir}:/tmp/pack`,  
57         '-v',  
58         `${join(sshKeyDir, 'known_hosts')}/var/www/.ssh/known_hosts`,  
59         '-v',  
60         `${privateSshKeyPath}/var/www/.ssh/id_rsa`,  
61         '-i',  
62         'dxy-pack-service',  
63         id,  
64         gitUrl,  
65         gitRef,  
66         npmScript,  
67         distDir,  
68         npmClient,  
69     ]);
```

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100
```

打包脚本

- 拉取代码
- 安装依赖
- 运行构建
- 压缩打包产物

打包服务工程搭建

- Egg
- TypeScript

```
  "scripts": {
    "start": "egg-scripts start --daemon --sticky",
    "stop": "egg-scripts stop --title=egg-server",
    "dev": "egg-bin dev --sticky --port 7000",
    "debug": "egg-bin debug --sticky --port 7000",
    "ci": "npm run lint && npm run cov",
    "lint": "npm run lint:ts && npm run lint:es",
    "lint:ts": "tslint --project . -c tslint.json",
    "lint:es": "eslint . --ext .js",
    "lint:md": "eslint . --ext .md",
    "lint-fix": "npm run lint-fix:ts && npm run lint-fix:es",
    "lint-fix:ts": "tslint --project . -c tslint.json --fix",
    "lint-fix:es": "eslint . --ext .js --fix",
    "lint-fix:md": "eslint . --ext .md --fix",
    "test": "egg-bin test",
    "cov": "egg-bin cov",
    "tsc": "ets && tsc -p tsconfig.json",
    "clean": "ets clean",
    "autod": "autod"
  },
  "husky": {
    "hooks": {
      "pre-commit": "lint-staged",
      "commit-msg": "commitlint -E HUSKY_GIT_PARAMS"
    },
    "lint-staged": {
      "*.ts": [
        "npm run lint-fix:ts",
        "git add"
      ],
      "*.js": [
        "npm run lint-fix:es",
        "git add"
      ],
      "*.md": [
        "npm run lint-fix:md",
        "git add"
      ]
    }
  }
}
```

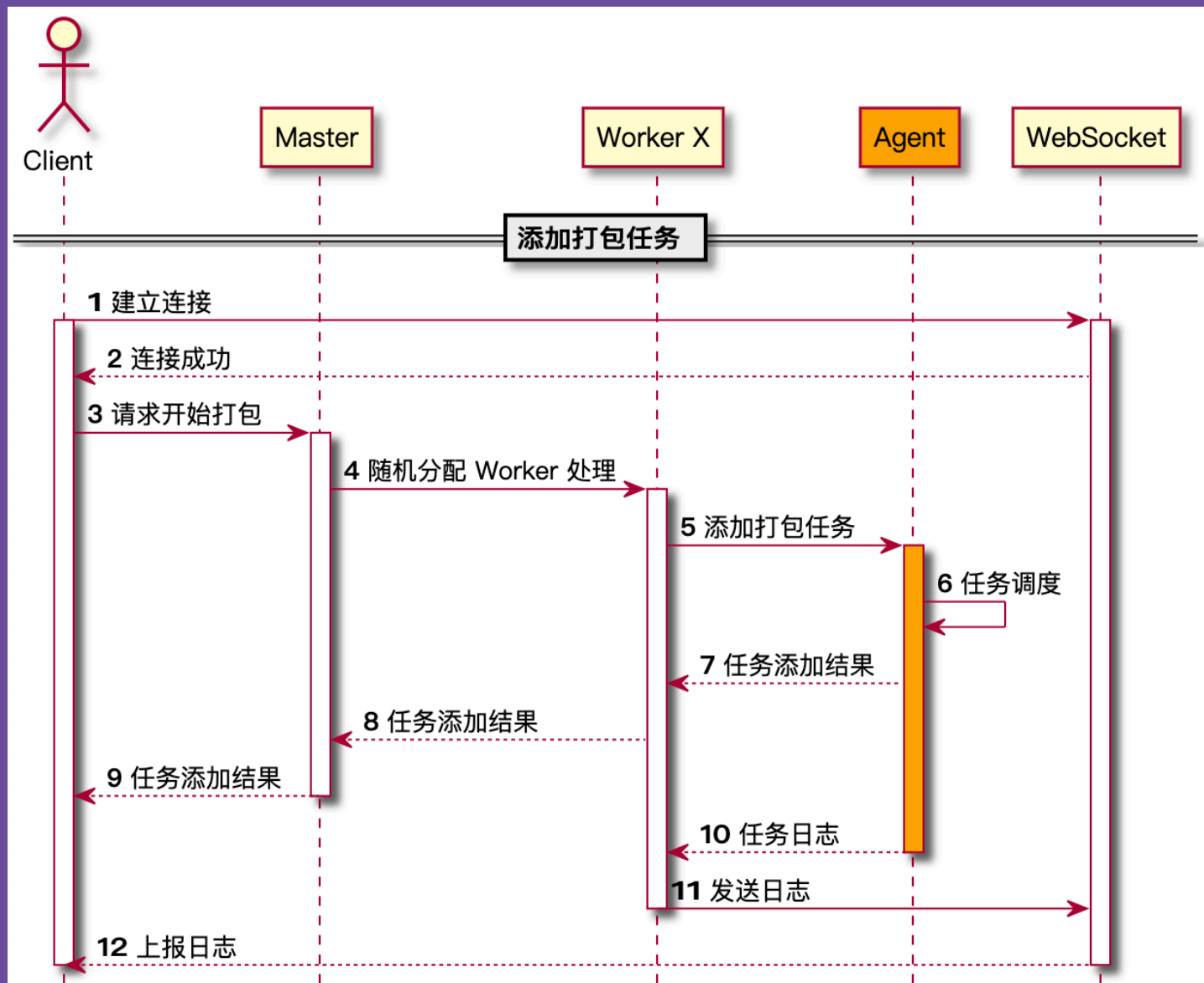
装饰器的运用

```
4  /**
5   * Controller 类的方法修饰器 -- 请求校验, 规则详见
6   * https://github.com/yiminghe/async-validator
7   */
8  export function validateRequest(
9    { query, params, body }: {
10      query?: object;
11      params?: object;
12      body?: object;
13    },
14  ) {
15    return function (_target: any, _propertyKey: string | symbol, descriptor: PropertyDescriptor) { ...
39    };
40  }
41
42  /**
43   * Service 类的方法修饰器 -- 函数异常捕获
44   */
45  export function catchError(code: number = 500, message: string = '服务出错') {
46    return function (_target: any, _propertyKey: string | symbol, descriptor: PropertyDescriptor) { ...
59    };
60  }
61
```

装饰器的运用

```
26  /**
27  /** 添加打包任务
28  /**
29  @validateRequest({
30  body: {
31    platform: [
32      { type: 'string', required: true, whitespace: true },
33      { type: 'enum', enum: PLATFORMS },
34    ],
35    app: { type: 'string', required: true, whitespace: true },
36    env: { type: 'string', required: true, whitespace: true },
37    gitUrl: { type: 'string', required: true, whitespace: true },
38    gitRef: { type: 'string', required: true, whitespace: true },
39    npmScript: { type: 'string', required: true, whitespace: true },
40    distDir: { type: 'string', required: true, whitespace: true },
41    npmClient: [
42      { type: 'string', whitespace: true },
43      { type: 'enum', enum: NPM_CLIENTS },
44    ],
45  },
46  })
47  public async create() { ...
65  }
```


请求打包时序图



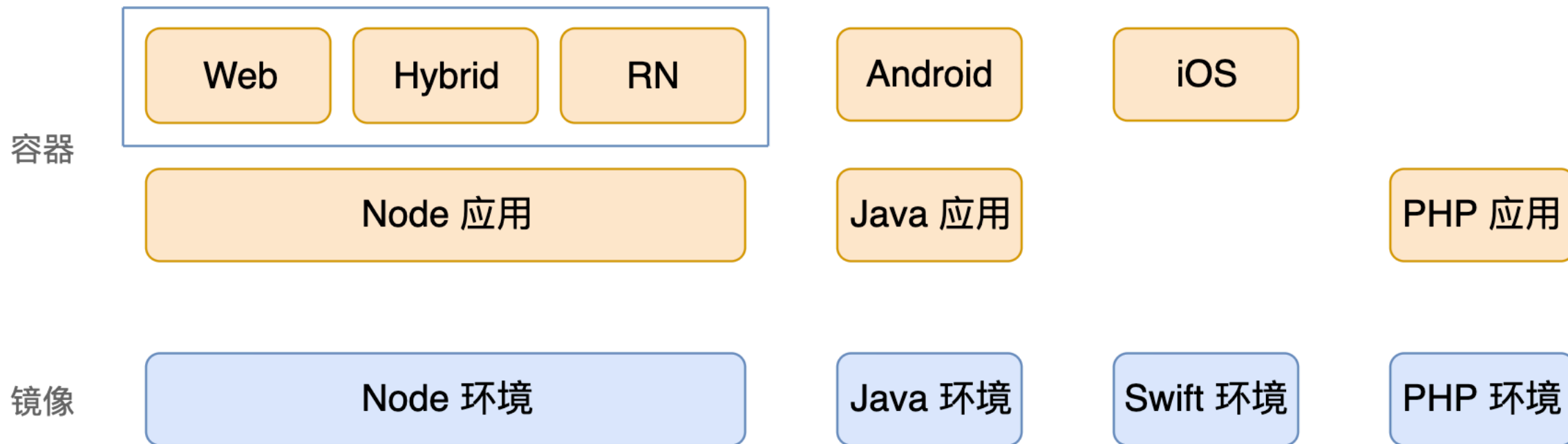
性能优化

- 安装依赖
 - Lock 文件使用国内镜像源
 - 打包容器数据卷缓存 node_modules , 速度提升 60% 左右
- 打包配置
 - 设置 parallel 模式
 - 限制并行任务数

3

未来展望

丁香园统一打包服务



丁香园的技术挑战

- 跨端方案
 - 工程体系
 - 逻辑分离
 - Bridge 跨平台兼容
- Node 建设
 - BFF
 - Microservices
 - Serverless

欢迎发送简历至 wanningbang@dxy.cn

招资深前端、前端专家



Q & A

Thanks