



有赞前端的统一打包方案

一张梦轩
@有赞前端



Superman + Build

Superman



Superman



A Global Command Based
On Webpack

superman init

So, How to use

```
linuxixuan in ~/tmp/superman-demo
$ superman init
? 项目名称? 如: www superman-demo
? js文件入口? src
? js本地打包文件路径? local/js
? js上传文件路径? build/js
? css文件入口? src/sass
? css本地打包文件路径? local/css
? css上传文件路径? build/css
? version文件格式? php
? js version文件的路径? ./version.php
? css version文件的路径? ./version_css.php
? CDN上传路径? /superman-demo/build
? shims配置文件的路径 (兼容requirejs配置)?
? paths配置文件的路径 (兼容requirejs配置)?
? 通用 (全局) 模块配置文件路径?
? 是否需要使用自动扫描Entry (最终的入口配置为项目entry和扫描的
? 项目的webpack配置文件路径?
superman环境配置成功!
可以通过直接编辑 superman.json来更改打包流程
conflict superman.json
? Overwrite superman.json? (Ynaxdh) █
```

superman dev

```
linuxixuan in ~/tmp/superman-demo
$ superman dev
开始打包: 开发模式
自动扫描入口文件...
打包业务逻辑, 打包文件:
{ app: [ './src/main.js' ] }
Hash: 4fc1d87f1d2fd9139470
Time: 370ms
  Asset      Size  Chunks  Chunk Names
  app.js    1.5B   0  [emitted]  app
```

先聊聊很久以前...

```
gulp
├── config
│   ├── env.js
│   ├── qiniu_deploy.js
│   ├── setting.js
│   ├── upyun_deploy.js
│   └── worker_require.js
├── daxue.js
├── fuwu.js
├── hash.js
├── intro.js
├── intro_new.js
├── iron_static.js
├── less.js
├── open.js
├── pinjian.js
├── qiniu.js
├── sass.js
├── shop.js
├── transpiling_jsx.js
├── uglify.js
├── upyun.js
├── www.js
├── gulpfile.js [error opening dir]
└── webpack
    ├── commonSetting.js
    ├── gulpfile.js
    ├── hash.js
    ├── npm-debug.log
    ├── webpack.config.js
    └── webpack_trans_util
        ├── amd_trans.js
        └── require_define_trans.js
Makefile [error opening dir]
```

基于 gulp 任务流

每个任务抽成一个文件

随着时间推移，问题开始凸显出来...

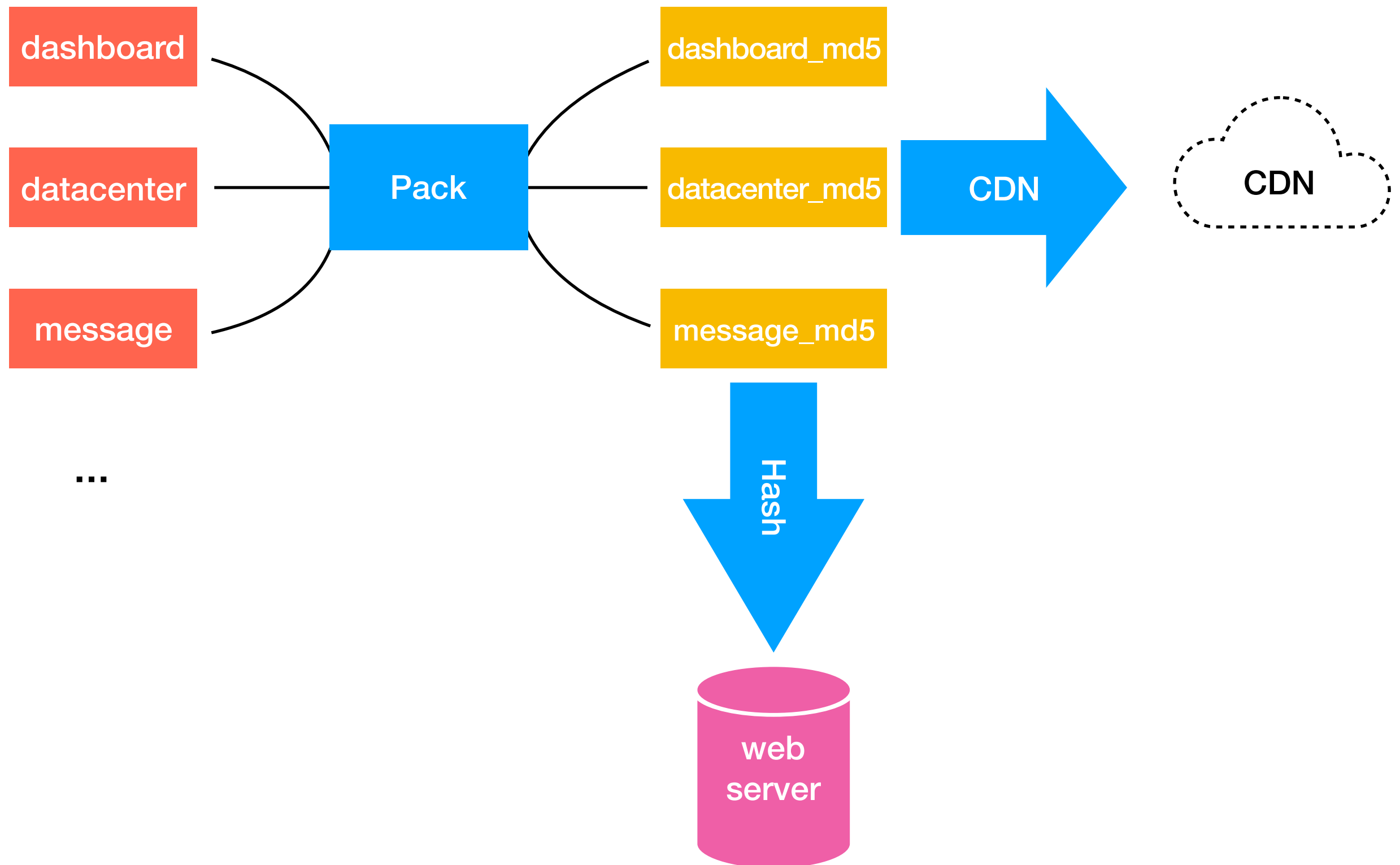
不够灵活

难以维护

不够通用

我们决定重写打包流程

打包流程



CDN

安全

自定义路径 VS 自动生成路径

暴露CDN能力在更多场景

不同格式文件上传不同域名

同时上传多个CDN灾备

CDN

使用方式

受众

上传指定文件

superman cdn path/to/filename

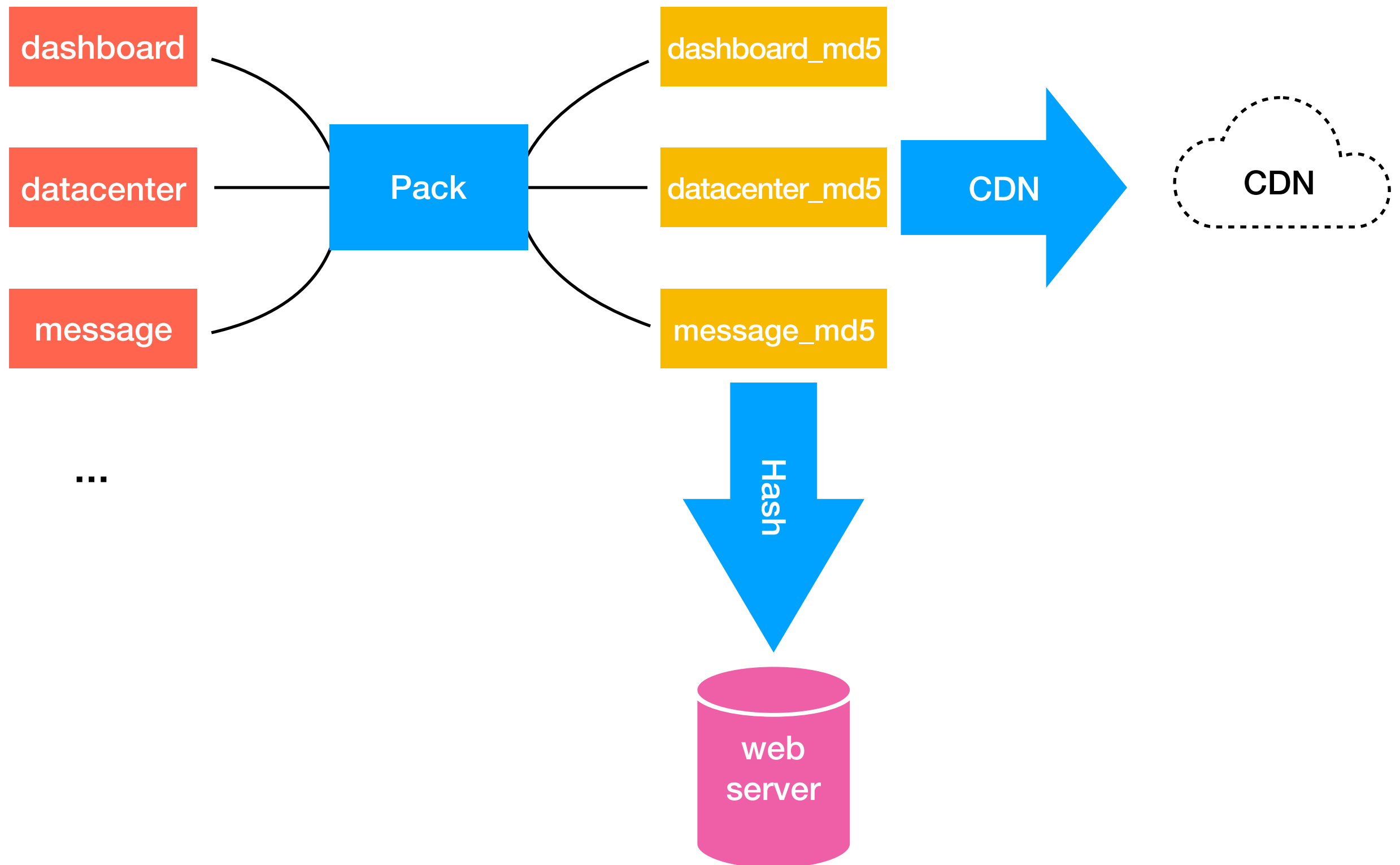
产品、运营...

上传项目中某一类文件

```
"cdn": {  
  "basic": {  
    "src": [  
      "build/**/*.js",  
      "build/**/*.css"  
    ],  
    "dist": "/v2/wsc/build"  
  },  
  "image": {  
    "src": [  
      "src/**/*.img/**/*.*"br/>    ],  
    "dist": "/v2/wsc/img"  
  }  
},
```

RD

打包的流程



Hash

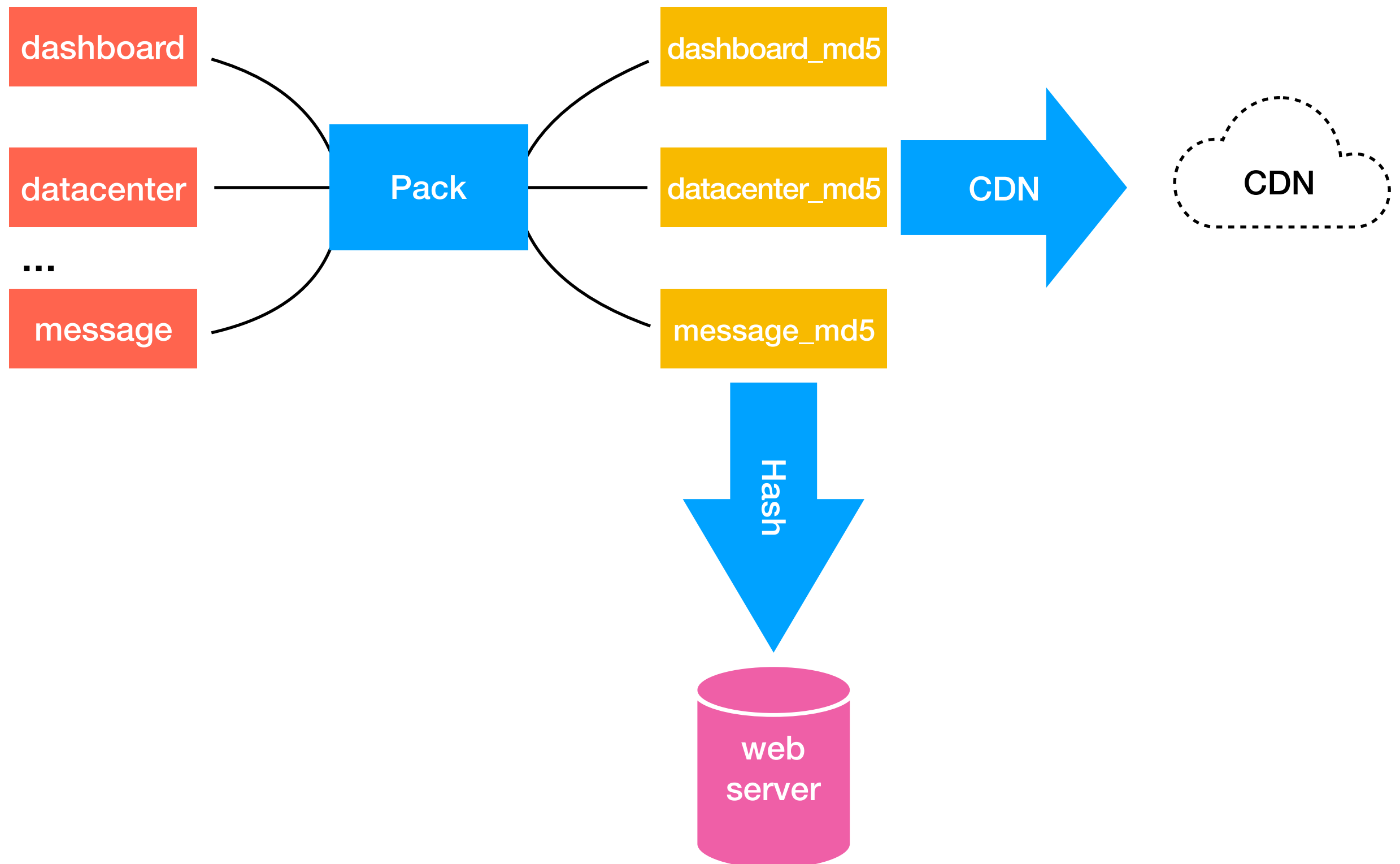
一个有赞特色功能

Hash

一个有赞特色功能

```
"hash": [  
  {  
    "src": "build/js/**/*.js",  
    "type": "php",  
    "dest": "../xxx/resource/config/version_wsc.php",  
    "recordTpl": "\"<%= name %>\" => \"<%= 'wsc/' + path %>.<%= extension %>\""  
  },  
  {  
    "src": "build/css/**/*.css",  
    "type": "php",  
    "dest": "../xxx/resource/config/version_wsc_css.php",  
    "recordTpl": "\"<%= name %>\" => \"<%= 'wsc/' + path %>.<%= extension %>\""  
  }  
],
```


打包的流程



Pack

gulp + r.js

require.path

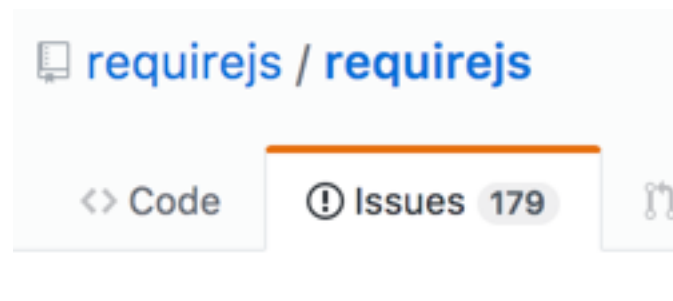
old fashion libs

作为最后一个完成的功能一定是有原因的...
global module

require.shim

require.loader

Pack



 **Jamund Ferguson** : moving requirejs to webpack
xjamundx

Pack

- `r.js => webpack`
- `requirejs.paths => webpack.alias`
- `requirejs.exclude => webpack.external`
- `requirejs.shims =>`
 - `webpack.importLoader`
 - `webpack.exposeLoader`

Pack

项目依赖太多

打包流程不同

为什么要实现通用打包工具？

代码风格各异

技术栈有差别

Pack

- 收敛依赖 (e.g. webpack, babel)

- 项目瘦身

- 统一风格

满足90%的需求

- js: babel-config

- css: sass or postcss

- 统一打包的行为

- 让所有人享受到各种优化

Pack

```
"build": {  
  "extra": {  
    "appendEntry": true  
  },  
  "js": {  
    "src": "src",  
    "local": "local/js",  
    "build": "build/js"  
  },  
  "css": {  
    "src": "src/sass",  
    "local": "local/css",  
    "build": "build/css"  
  }  
}
```

Pack

```
"build": {
  "extra": {
    "shims": "shim.js",
    "paths": "alias.js",
    "commonModule": "commonModule.js",
    "esVersion": 6,
    "appendEntry": true,
    "globalEntry": "global.js",
    "babelrc": "superman_babelrc"
  },
  "confPath": "webpack.conf.superman.js",
  "js": {
    "src": "src/pages",
    "local": "local/js",
    "build": "build/js"
  },
  "css": {
    "src": "src/styles",
    "local": "local/css",
    "build": "build/css"
  }
}
```


“这是一场统一和灵活的战争”

Demo

Demo??不存在的

Status of Superman

Status of Superman

Done

Status of Superman

Done

- extract css: 自动删除多余的js output file
- postcss-plugins: css url 自动注入cdn域名...
- 性能优化:
 - common chunk——global.js
 - ddl——base.js(common module)
 - happy-pack——并行打包, 缓存babel file
 - parallel-uglify——并行压缩
- hotload: 无需改源码自动hotload (废弃😓)
- 模块分析: 展示项目中每个模块的体积及占比

```
linixuan in ~/repos/store on hotfix/wap
$ superman analyze
开始分析模块...
通用模块配置文件: /Users/linixuan/repos/superman/.superman/retail.commonModule.json
zentr: 1.88 MB (44.8%)
lodash: 222.94 KB (29.3%)
  ↳self: 877.95 KB (79.7%)
react-dom: 535.36 KB (21.8%)
react-dnd-html5-backend: 106.11 KB (4.32%)
  ↳lodash: 64.98 KB (61.2%)
  ↳self: 41.13 KB (38.8%)
react: 303.13 KB (4.20%)
dnd-core: 101.83 KB (4.15%)
  ↳lodash: 52.26 KB (51.3%)
  ↳redux: 9.06 KB (8.89%)
  ↳self: 40.58 KB (39.8%)
react-router: 88.12 KB (3.50%)
react-dnd: 69.51 KB (2.83%)
  ↳lodash: 6.23 KB (8.97%)
HOISTED=react-static: 1.95 KB (2.81%)
  ↳self: 61.33 KB (88.2%)
```

Status of Superman

Doing

Status of Superman

Doing

- Webpack3: unpublish
- 增量打包: 自动识别哪些entry需要打包@beta
- Sass2Postcss

Status of Superman

ToDo

Status of Superman

ToDo

- Sea of postcss plugins
- Lifecycle of superman
- Open source
- Docker



Superman + Build

Build



A Server For Packing Code

Why?


Build



- 统一打包环境
- 更高的机器性能（集群）
- 前端上线变成原子操作

Build



 Youzan.com

我的分页

有赞人巴卡

Code Review

构建平台

创建者


项目

分支

命令

状态

时间信息




headquarter-portal

master

release

● 构建成功

耗时44秒




iron-front

release/201707281522

www_pre

● 构建成功

耗时292秒




retail-php

master

release

● 构建成功

耗时54秒




iron-front

release/201707281522

pre

● 构建成功

耗时554秒




retail-php

hotfix/refunds_list

release

● 构建成功

耗时60秒




retail-php

hotfix/refunds_list

release

● 构建成功

耗时88秒




retail-php

feature/pam_u2

release

● 构建成功

耗时14秒




iron-front

feature/delivery_unity_0717

www_pre

● 构建成功

耗时414秒



retail-php

hotfix/add-alert


release

● 构建成功

耗时109秒

< 1 2 3 >

添加任务

 提示：请根据目标项目的构建需要选择合适的构建命令，请充分利用“再次构建”功能。

机器集群状态

服务器 0

待分配

空闲

服务器 1

待分配

空闲

节点信息

System Information

Hostnames: qabb-r1buffer31

Version: CentOS release 6.8 (Final)

Architectures: x86_64

IP: 10.215.20.31

Yarn Version: 0.27.5

Node Version: v6.10.0

Superman Version: 0.6.7

Cores: 24

Memory used: 9748952 / 65920884

Active WorkerFarm processes: 23

Active HappyPeck processes: 0

Active Superman processes: 1

Build



- Easy Access
 - Based on Make
- Performance
- Pack History
- Combination With CI

Thanks