前端 MonoRepo 实践

lerna + yarn workspaces

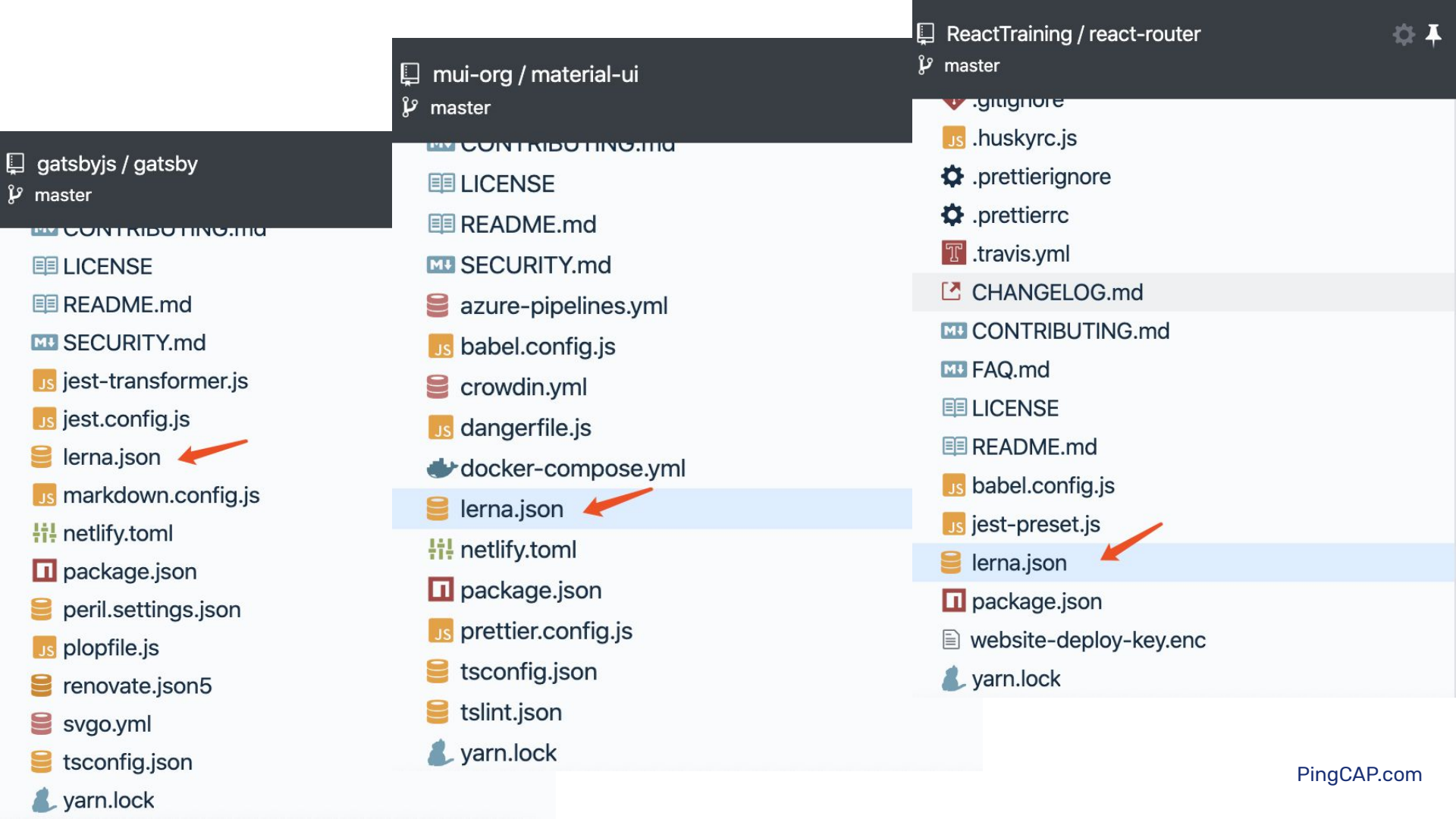
PingCAP TiDB

Presented by Baoling Huang

2020/04/16

PingCAP

lerna.json ?

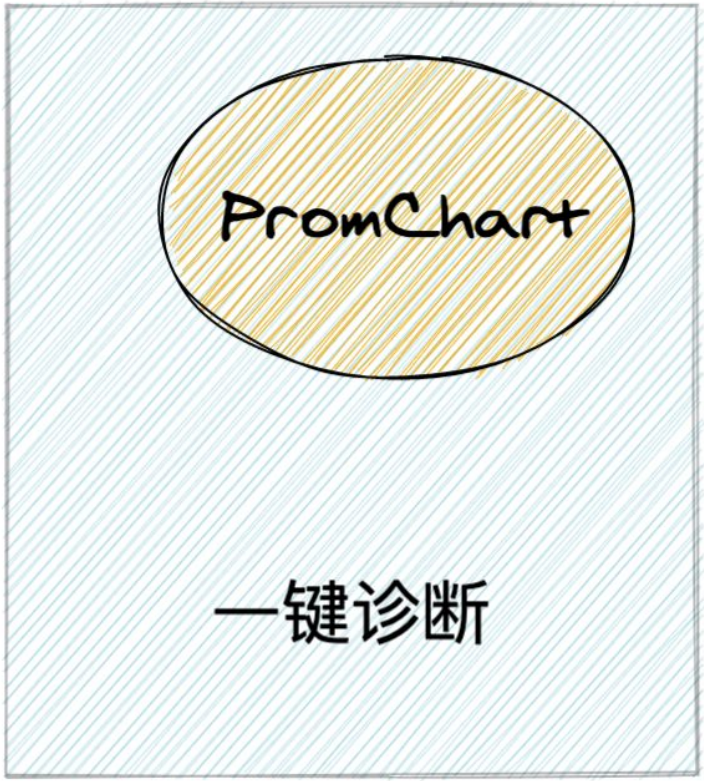

gatsbyjs / gatsby
master
LICENSE
README.md
SECURITY.md
jest-transformer.js
jest.config.js
lerna.json
markdown.config.js
netlify.toml
package.json
peril.settings.json
plopfile.js
renovate.json5
svgo.yml
tsconfig.json
yarn.lock
mui-org / material-ui
master
LICENSE
README.md
SECURITY.md
azure-pipelines.yml
babel.config.js
crowdin.yml
dangerfile.js
docker-compose.yml
lerna.json
netlify.toml
package.json
prettier.config.js
tsconfig.json
tslint.json
yarn.lock
ReactTraining / react-router
master
.huskyrc.js
.prettierignore
.prettierrc
.travis.yml
CHANGELOG.md
CONTRIBUTING.md
FAQ.md
LICENSE
README.md
babel.config.js
jest-preset.js
lerna.json
package.json
website-deploy-key.enc
yarn.lock
PingCAP.com

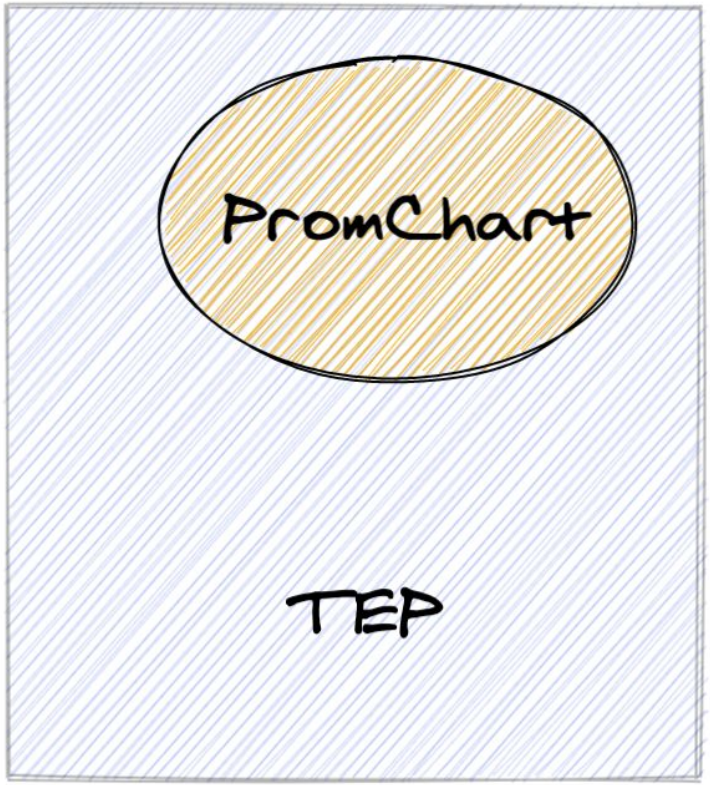
Lerna

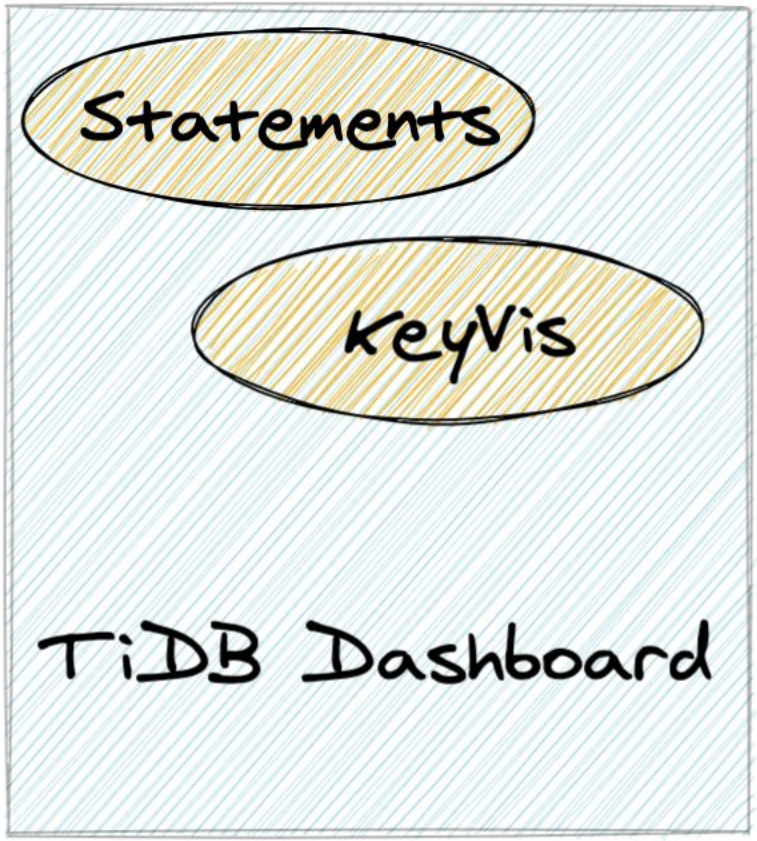
A tool for managing JavaScript projects with multiple packages.

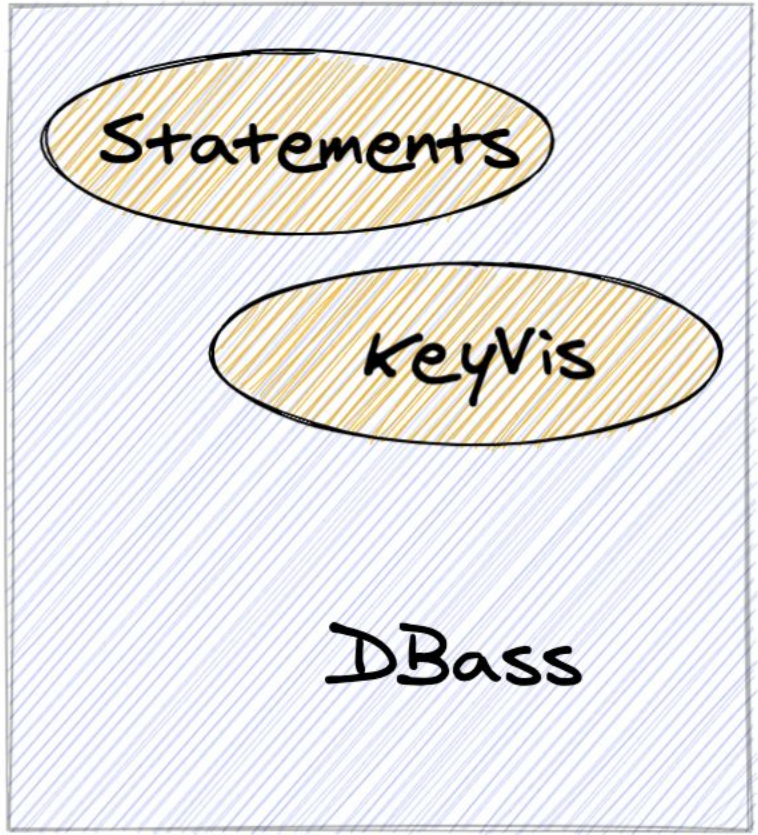
PingCAP

需求？(场景)


PromChart
一键诊断


PromChart
TEP


Statements
KeyVis
TiDB Dashboard


Statements
KeyVis
DBass

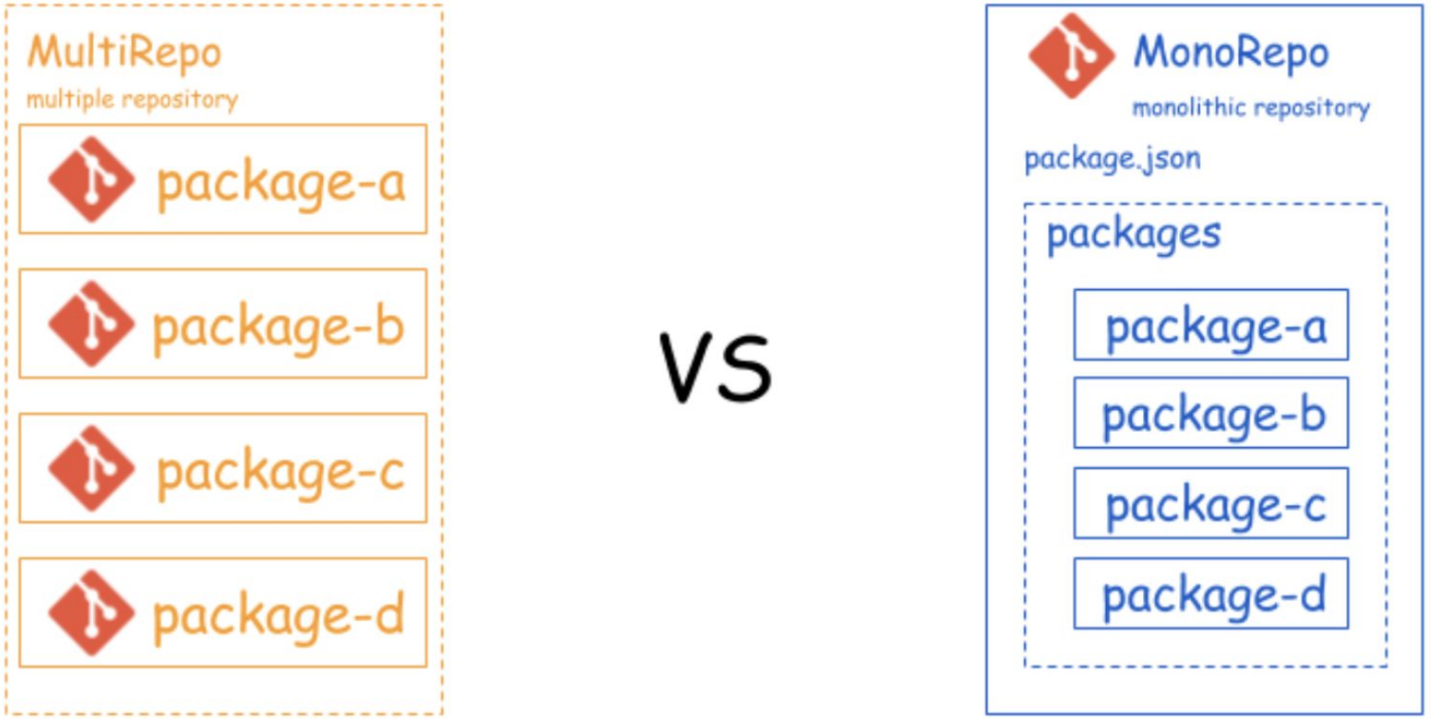
如何在多个项目之间复用相同的功能模块？

- 方法一：拷贝代码
- 优点：简单快速，而且灵活 (方便自己改代码)
- 缺点：显而易见，可维护性差
- 在公司内部可以使用

如何在多个项目之间复用相同的功能模块？

- 方法二：将需要复用的功能模块抽取成可独立分发的库
- 优点：
  - 使用方便: npm install / yarn add
  - 好维护，库更新直接 update 就行
- 缺点：需要单独构建，分发，稍微有点麻烦
- 面临的新问题：如何方便地管理，测试，分发这些库

MulitRepo Vs MonoRepo



MultiRepo

- 每个独立模块都是一个单独的 git repo
- 优点：个人觉得相比 MonoRepo 没啥突出的优点
- 缺点：
  - 如果模块多，则要创建的仓库太多
  - 不方便测试，每个模块需要在每个 repo 创建自己的测试 app (比如加个 example 目录啥的)，要么更新代码后 publish 到 npm registry，再在独立 app 中测试，流程太长 (另一种办法是在本地 npm link，没试过...)
  - 如果多个模块间有依赖就更麻烦了，比如 pkg-c 依赖 pkg-b, pkg-b 依赖 pkg-a，如果 pkg-a 更新了，那就得手动更新并发布 pkg-b, pkg-c ...

MonoRepo

- 每个独立模块和主 APP 在一个 git repo 中
- 每个独立模块放置在 packages 目录下的单独目录中

```
├── packages
|   ├── pkg1
|   |   ├── package.json
|   ├── pkg2
|   |   ├── package.json
├── package.json
```

一些疑问？

- 虽然这些模块是和主 APP 放在一个 Repo 里了，那模块不也是还要 publish 到 npm registry，主 APP 才能使用它们吗？(非也)
- 如果模块之间有依赖，我修改了某个模块后，不也还是需要手动修改其它依赖它的模块的 package.json 吗？(非也)
- 每个模块下都会有一个 node_modules 吧，那项目目录体积有点恐怖哦 (非也)

优点

- 一个仓库维护多个模块，代码集中
- 方便版本管理和依赖管理，模块之间的引用，调试
- 统一发布，生成 CHANGELOG 等
- 所有依赖放置在一个 node_modules 中，减少磁盘空间
- ...

PingCAP

实现

背后的功臣：lerna & yarn workspaces

- 两者关系
  - 功能上较多重合
  - 先有 lerna 后有 yarn workspaces
  - lerna 为第三方实现，yarn workspaces 为原生实现
  - lerna 支持和 npm 使用，yarn workspaces 只支持 yarn
  - yarn 官方推荐：用 yarn 来处理依赖问题，用 lerna 来处理发布问题

配置 lerna.json

```
{
  "packages": ["packages/*"],
  "version": "independent",
  "npmClient": "yarn",
  "useWorkspaces": true
}
```

配置：root package.json

```
{
  "workspaces": ["packages/*"],
  "private": true,
  "devDependencies": {
    "lerna": "^3.20.2",
    ...
  }
  ...
}
```

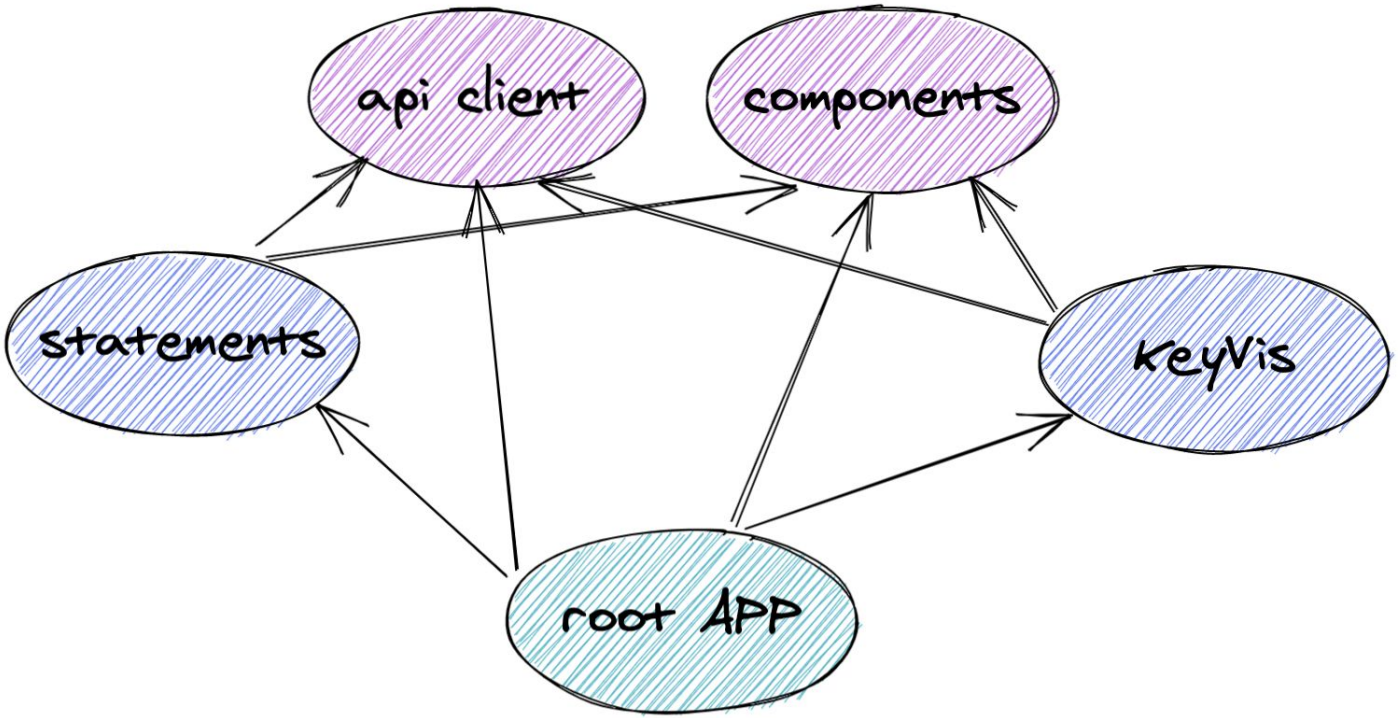
yarn workspaces 管理依赖

- 给某个 package 安装依赖
  - yarn workspace packageB add antd
  - yarn workspace packageB add packageA@version
- 给所有的 packages 安装依赖
  - yarn workspaces run add lodash
- 给 root 安装依赖: 一般的公用的开发工具都是安装在 root 里
  - yarn add -W -D typescript

leran 管理打包，发布

- 按拓扑顺序进行 build (yarn workspaces 不支持)
  - lerna run --stream --sort build
- 开发模式下并行 watch 所有 packages
  - lerna run --parallel watch
- 升级版本：所有依赖同时更新
  - lerna version
- 发布
  - lerna publish

TiDB-Dashboard 的 packages 拓扑



package 是怎么打包的？

- Rollup.js
- root APP 是 webpack 打包

参考

- lerna+yarn workspace+monorepo 项目的最佳实践
- Workspaces in Yarn
- Why Lerna and Yarn Workspaces is a Perfect Match for Building Mono-Repos – A Close Look at Features and Performance
- 基于 lerna 和 yarn workspace 的 monorepo 工作流

PingCAP

Thank You !