# Native Modules 发布到NPM

文档地址：<https://reactnative.dev/docs/native-modules-setup>

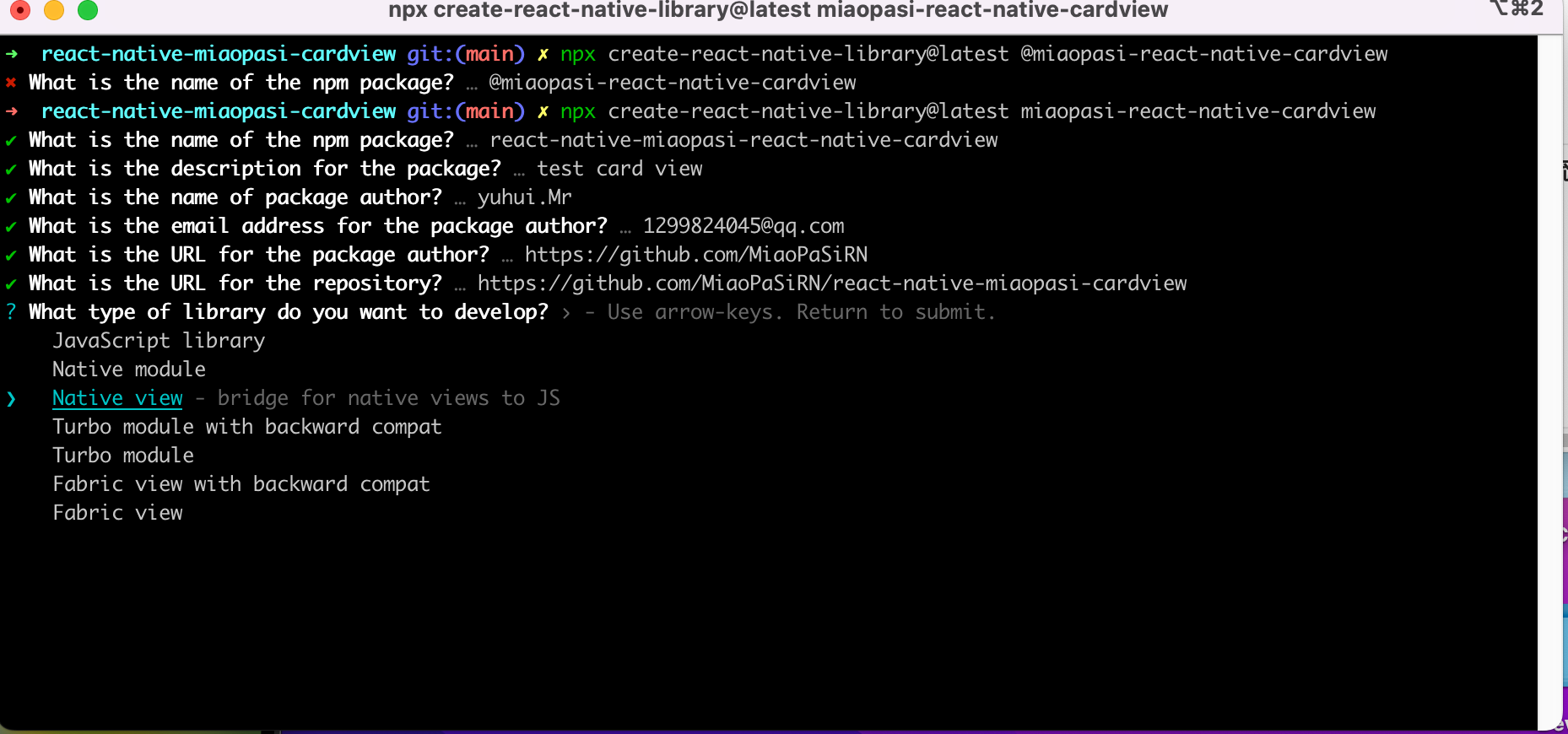
1. 全局安装 create-react-native-library 模块

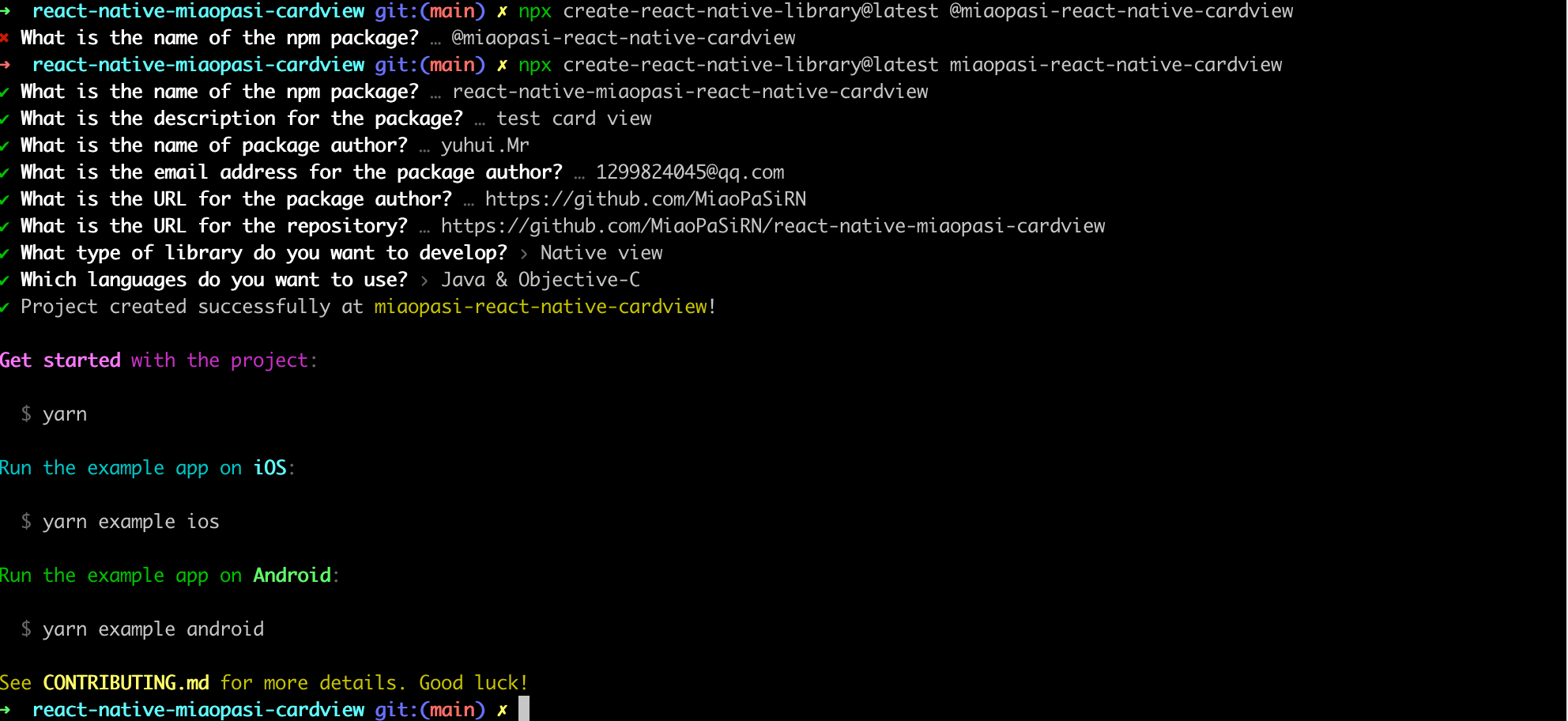
$ yarn global add create-react-native-library

1. 创建插件工程

$ npx create-react-native-library@latest react-native-awesome-module

示例：  
npx create-react-native-library@latest cardview





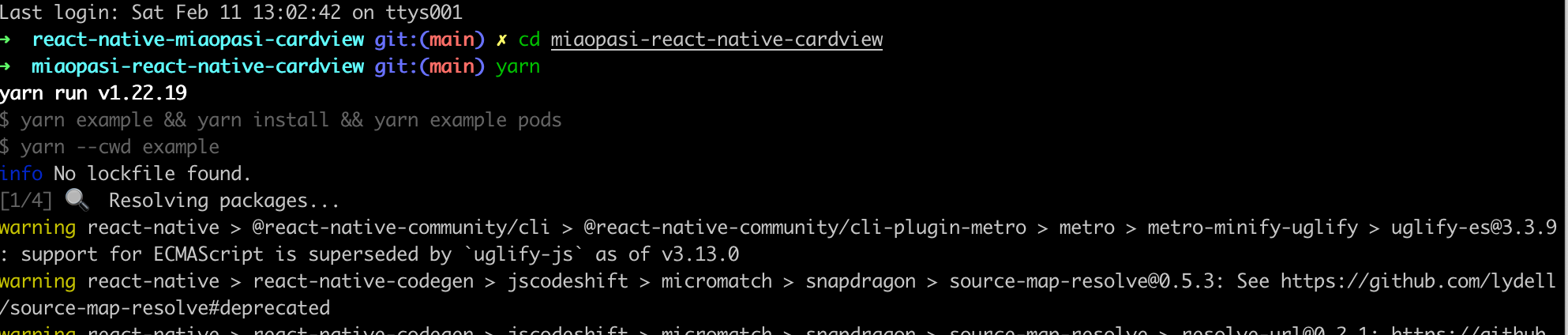
1. 修改名字

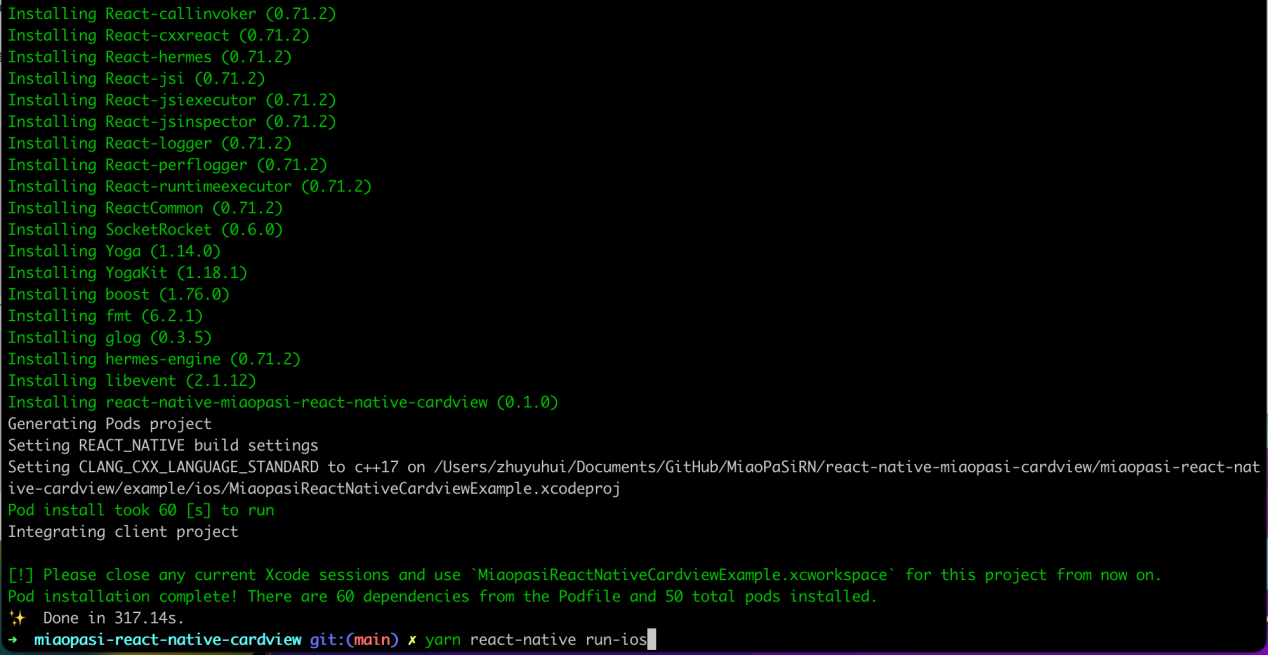
$ mv cardview miaopasi-rn-cardview

有人可能会说，楼主为什么不直接生成react-native-cardview的项目，而要先生成cardview再重命名。其实这是一个小技巧，因为利用react-native-create-library生产的项目，一些跟组件相关的名称或者类会默认加上react-native或者RN前缀。  
例如，如果你的初始项目名是react-native-card-view，那么package.json中定义的组件名将是react-native-react-native-card-view，android模块中定义的相关类会是RNReactNativeCardviewModule.java，这显然比较丑啊。

1. 安装相关依赖

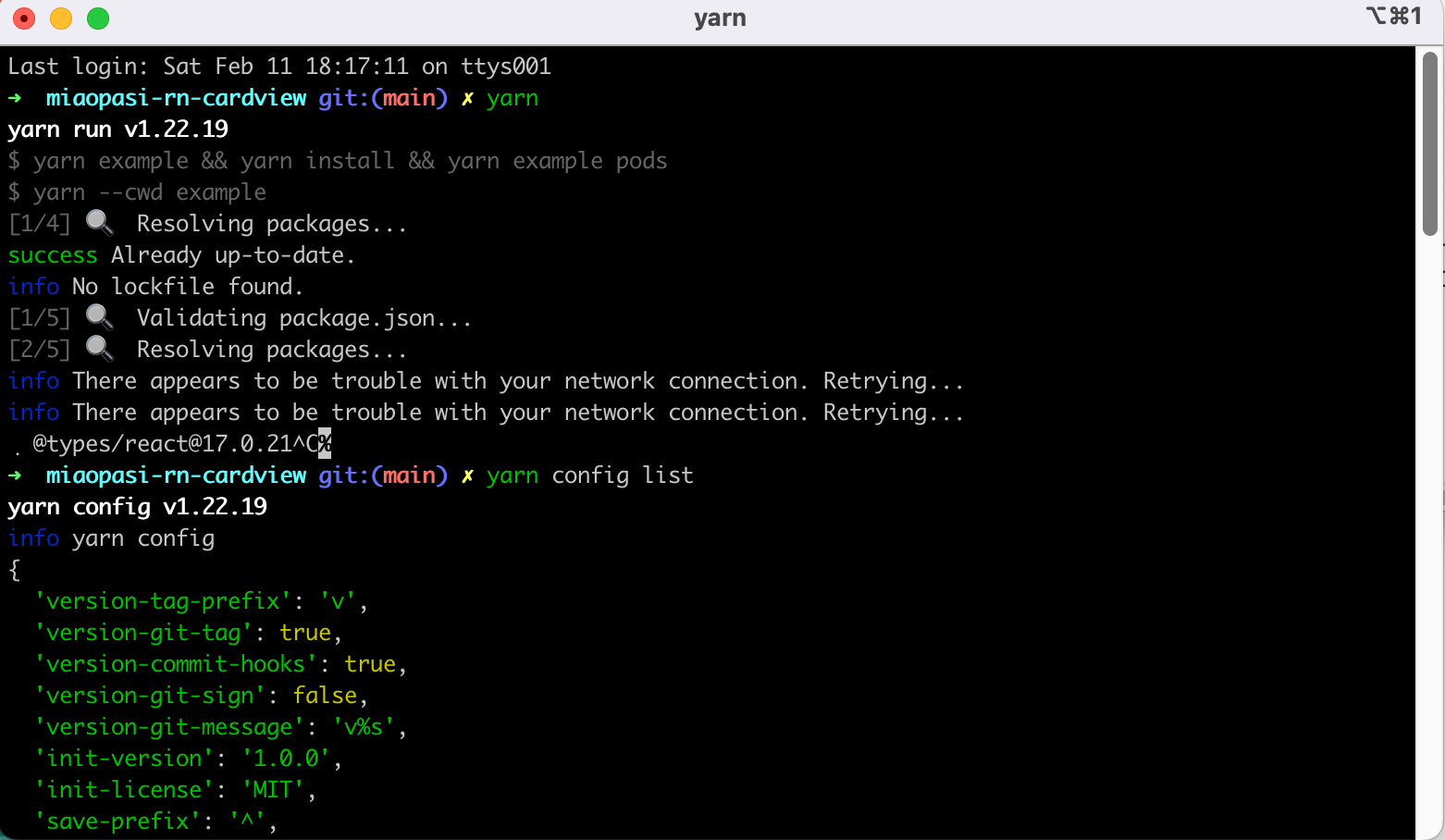
$ yarn





## 报错

1. 使用yarn命令报错：

info There appears to be trouble with your network connection. Retrying...  


解决方案：

出现这种问题分为不同环境，且大概率是代理问题；

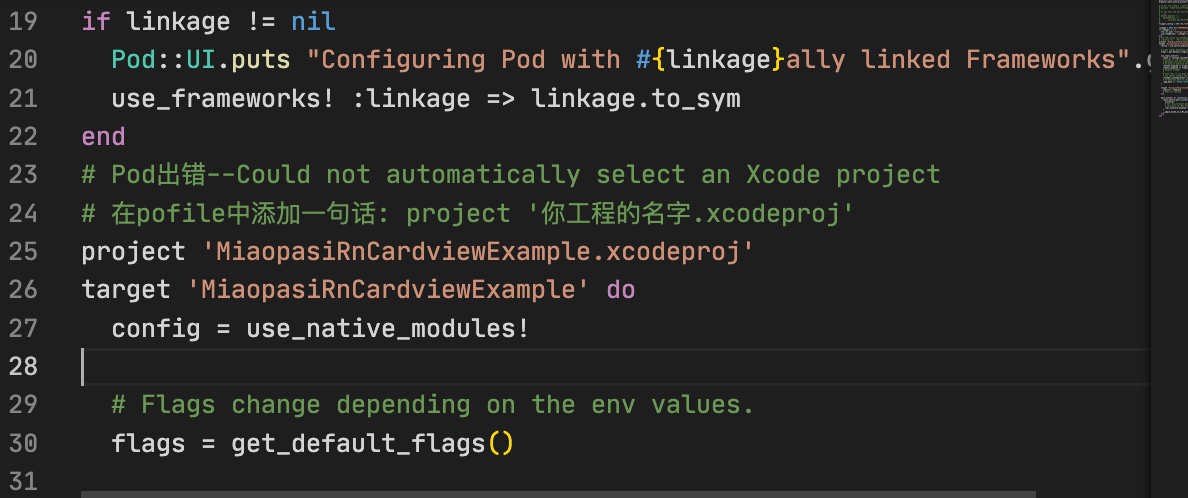
# 调整为taobao镜像源

yarn config set registry https://registry.npm.taobao.org

Pod出错--Could not automatically select an Xcode project

# Pod出错--Could not automatically select an Xcode project

# 在pofile中添加一句话: project '你工程的名字.xcodeproj'



## 组件发布

链接：<https://juejin.cn/post/6844903670694461454>

开发好组件之后，想在其他的项目(或者提供给其他人安装使用)中通过npm install的方式安装你的组件，那么你的组件必须发布到npm registry中。  
简单来说，npm registry就相当于一个包注册管理中心。它管理着全世界的开发者们发布上来的各种插件，同时开发者们可以通过npm install的方式安装所需要的插件。

npm官方registry为：[registry.npmjs.org/](https://link.juejin.cn?target=http://registry.npmjs.org/" \o "http://registry.npmjs.org/" \t "/Users/zhuyuhui/Documents\\x/_blank)

国内速度较快的为：[registry.npm.taobao.org/](https://link.juejin.cn?target=https://registry.npm.taobao.org/" \o "https://registry.npm.taobao.org/" \t "/Users/zhuyuhui/Documents\\x/_blank)

#### 查看

你可以查看当前使用的registry:

$ npm config get registry

#### 切换

当然也可以通过命令切换当前使用的npm registry

# 全局切换$ npm config set registry http://registry.npmjs.org/

有时候你可能只想在执行某些npm命令时临时切换，这个时候，可以使用--registry来指定临时切换的registry，比如在npm发布

$ npm publish --registry http://registry.npmjs.org/

就可以临时指定，当然，在命令执行结束之后，registry仍然会恢复到原来的registry

#### 创建/登陆npm registry账户

要发布组件到npm registry，你必须要是npm registry的注册用户，通过：

$ npm adduser

来新增一个用户，或者你已经在[官网](https://link.juejin.cn?target=https://www.npmjs.com/" \o "https://www.npmjs.com/" \t "/Users/zhuyuhui/Documents\\x/_blank)注册了一个用户，可以通过：

$ npm login复制代码

来登陆npm registry账户。

利用以下两种方式来确认你是否创建/登陆成功npm registry

命令$ npm whoami确认本地是否成功登陆认证成功

在线打开 [npmjs.com/~username](https://link.juejin.cn?target=https://npmjs.com/~username" \o "https://npmjs.com/~username" \t "/Users/zhuyuhui/Documents\\x/_blank) 查看是否创建账户成功

npm adduser 报错e409 Conflict

npm login或者npm adduser时 输入完账号密码邮箱后 提示E409 Conflict 报错 一般因为淘宝镜像的原因

解决：找到 .npmrc 一般在文件C:\Users\Administrator 中

或C:\Program Files\nodejs\node\_modules\npm

将里面的

registry=http://registry.npm.taobao.org

改成

registry=http://registry.npmjs.org/

即可

npm adduser --registry http://registry.npmjs.org/

npm notice Beginning October 4, 2021, all connections to the npm registry - including for package installation - must use TLS 1.2 or higher. You are currently using plaintext http to connect.

sssss

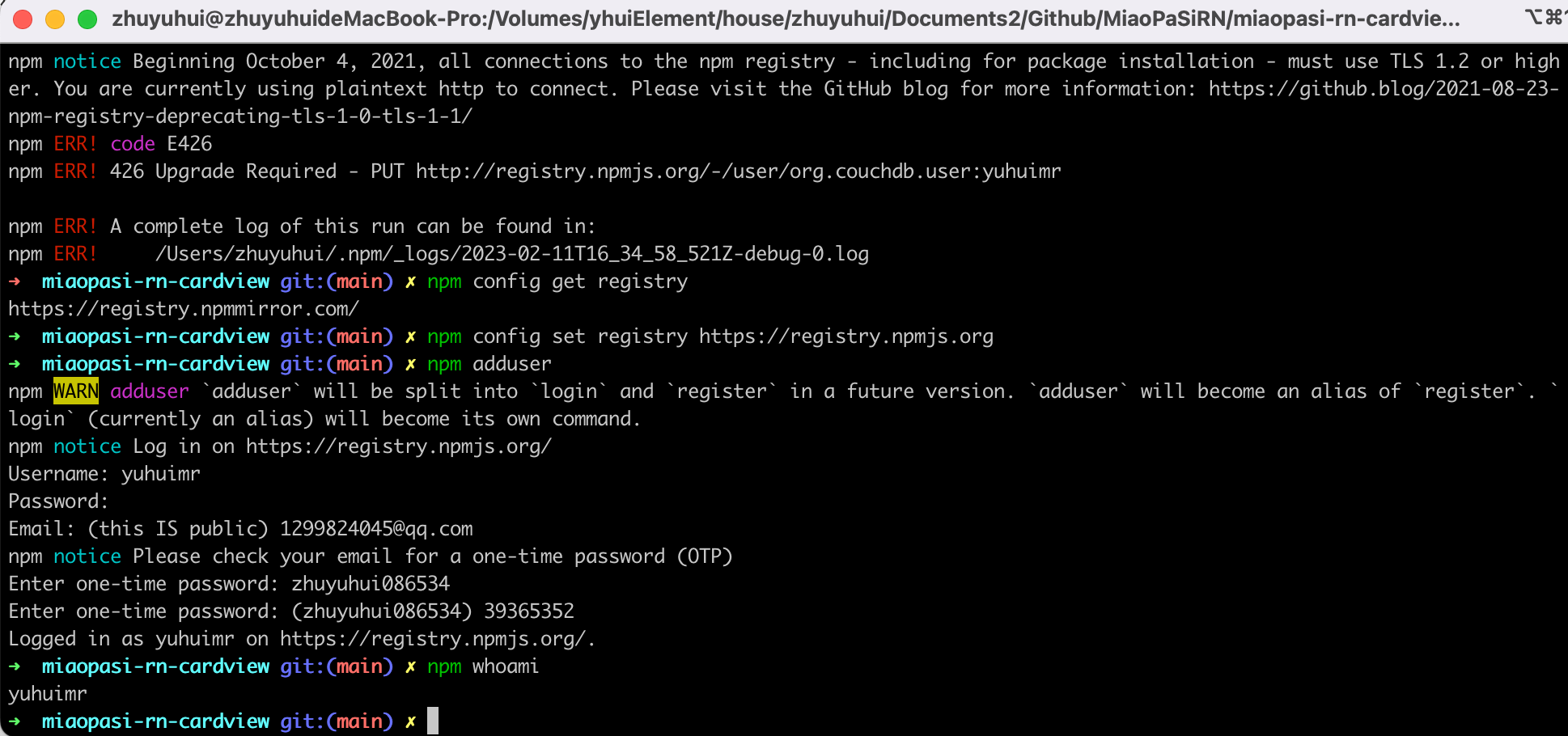
“从2021 10月4日开始，与npm注册表的所有连接（包括用于安装软件包的连接）必须使用TLS 1.2或更高版本。您当前正在使用明文http连接”

1、查看本地npm是不是官方的

npm config get registry

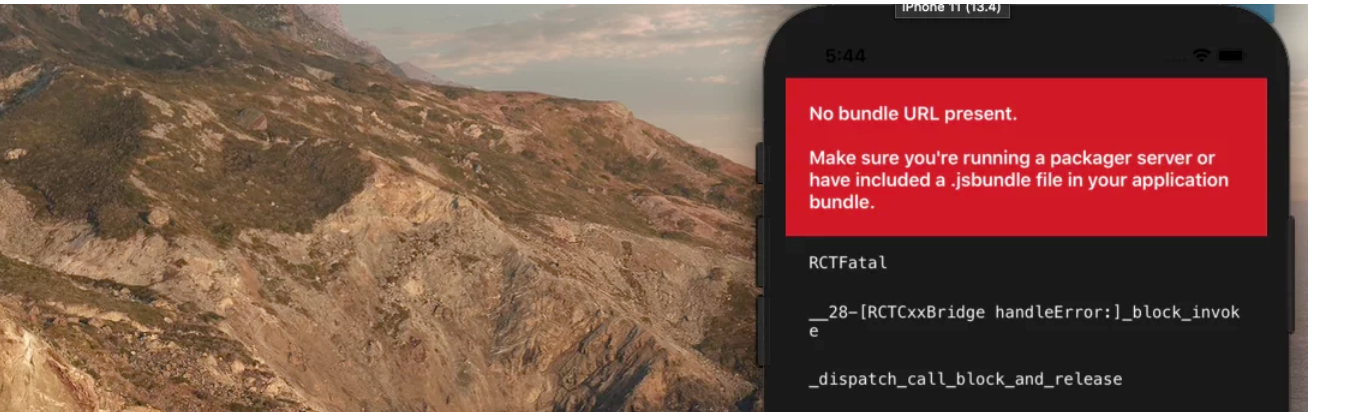
2、设置npm官方源

npm config set registry https://registry.npmjs.org



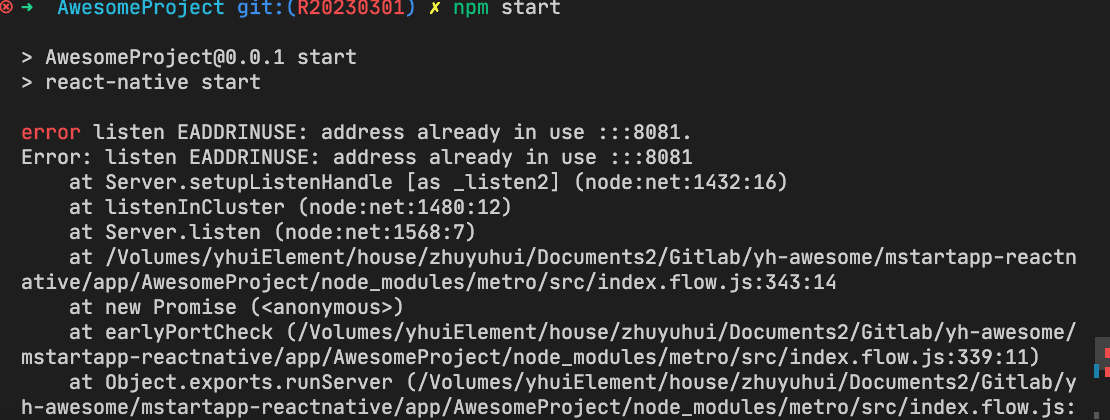
No bundle URL present.

Make sure you're running a package server or have included a .jsbundle file in your application bundle.



问题原因：8081端口被占用。

尝试运行命令：npm start

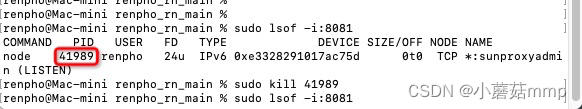


解决方案：

1、查看被占用进程的pid：

sudo lsof -i:8081

1



2、鲨死这个进程：

sudo kill 41989

3、再次check一下：

sudo lsof -i:8081

#### 首次发布

第一次发布的话，直接执行命令：

$ npm publish

就搞定了，可以在线查看确认是否发布成功。访问链接(<package>是你发布的npm package名)：  
[www.npmjs.com/package/<pa…](https://link.juejin.cn?target=https://www.npmjs.com/package/YourComponentName" \o "https://www.npmjs.com/package/YourComponentName" \t "/Users/zhuyuhui/Documents\\x/_blank)  
看看是否已经有内容了，有内容说明发布成功了。

#### 更新发布

如果不是首次发布，需要执行两个命令

$ npm version <update\_type>$ npm publish

$ npm version命令是用来自动更新版本号，update\_type取值有patch ``minor ``major。那么在什么场景应该选择什么update\_type呢？看下表

|  |  |  |  |
| --- | --- | --- | --- |
| update\_type | 场景 | 版本号规则 | 举例 |
| - | 首次发布 | 版本号1.0.0 | 1.0.0 |
| path | 修复bug、微小改动时 | 从版本号第3位开始增量变动 | 1.0.0 -> 1.0.1 |
| minor | 上线新功能，并且对当前版本已有功能模块不影响时 | 从版本号第2位开始增量变动 | 1.0.3 -> 1.1.3 |
| major | 上线多个新功能模块，并且对当前版本已有功能会有影响时 | 从版本号第1位开始增量变动 | 1.0.3 -> 2.0.0 |

注意  
  
如果首次发布版本号不是1.0.0的话，那么用$ npm version <update\_type>  
来更新会报错，因为你没有按照它约定的版本规则来，这个时候，你可以手动修改package.json中的version字段为符合约定规则的版本号，然后直接执行$ npm publish就可以，然后下次再增量更新的时候，就可以直接使用$ npm version <update\_type>的方式来自动更新版本号了