<https://github.com/xtx1130/blog/issues/8>

travis-ci持续集成和coveralls.io覆盖度测试报告上传使用详解 #8

**Open**

[**xtx1130**](https://github.com/xtx1130) opened this issue on 20 Nov 2017 · 3 comments

**Comments**

窗体顶端

Assignees

No one assigned

窗体底端

窗体顶端

Labels

[**nodejs**](https://github.com/xtx1130/blog/labels/nodejs)[**框架**](https://github.com/xtx1130/blog/labels/%E6%A1%86%E6%9E%B6)

窗体底端

窗体顶端

Projects

None yet

窗体底端

窗体顶端

Milestone

No milestone

窗体底端

Notifications

窗体顶端

 Subscribe

窗体底端

You’re not receiving notifications from this thread.

2 participants

[@xtx1130](https://github.com/xtx1130)

[@For-me](https://github.com/For-me)

[](https://github.com/xtx1130)

Owner

[**xtx1130**](https://github.com/xtx1130)**commented**[**on 20 Nov 2017**](https://github.com/xtx1130/blog/issues/8#issue-275227238)**•**

**edited**

|  |
| --- |
| 跟大家分享一下如何使用travis-ci和coveralls.io分别实现持续集成和覆盖度测试报告上传  **背景（npm找包指北）**  npm上包的储备量实在是太大了，如何快速找到你心仪的而且高质量的包呢？   * 不知道大家是不是注意过，很多开源的项目，都拥有下面这两个角标： [issue8-1](https://github.com/xtx1130/blog/raw/master/images/issue8/issue8-1.png) 一般拥有这两个角标的，包的质量都不会特别差。在我的找包经验中，一般都会先看这两个角标，如果有覆盖度测试的角标，说明他的测试用例比较全面，入手了解如何使用一个包的各种API，最快速的方式就是阅读测试用例；如果有持续集成角标，说明他的包是可以跑通的，从安装到使用的流程上是没有任何问题的。 * 再有一个看包的质量如何的方式就是看他的下载量：如果月下载量大于1k的话，说明这个包有人长期使用，稳定性应该不错，可以一式。 * 还有一个辨别包的质量好坏的标准就是看这个包的核心贡献者，国外的比如node CTC成员以及tj大神等等，国内的比如说蚂蚁金服、淘宝、天猫的前端/node团队成员、75团、fex、Alloy Team等（人员太多就不一一列出了，话说十月份腾讯好像在git上追回了一部分开源代码版权？其中还有livepool）。   **travis-ci简介和使用**  **简介**  travis-ci是一款开源的实现持续集成的工具。因为开源而且空间公用，所以比较适合个人开发者或者较小的项目（像node这种的都会有自己搭建的jekins，集成版本也会有很多版本，而且每天都会有nightly-build）。  **使用**  这个使用起来也是非常方便的，你只需要在你的文件目录的根目录下加上.travis.yml文件，然后上传到git上就好了，tarvis-ci可以同步你github上的项目。接下来跟大家详细说一下常用的配置：  sudo: false//sudo 最好设置成false，这样travis会对你的项目根据提交的集成时间顺延进行集成，而且官方不鼓励sudo  language: node\_js //语言设置，前端来说一般都是node咯  node\_js:  - "8"//集成时候用到的nodejs版本，如果是多个请继续在下面加 - "9"，ps:不同版本会分开集成  install: npm install//在环境构建的时候执行的指令，一般都是npm install吧，装npm包  script: npm test // 这是在执行的时候运行的命令，执行一遍单测保证程序的可用性，如果有问题会直接exit 1退出  os:  - linux //操作系统环境  after\_script: npm run cov//在命令执行完之后运行的指令，在这里run cov是利用coveralls.io和jest实现覆盖度测试并把报告上传到coveralls.io  env:  global://设置集成环境的全局变量  - NODE\_ENV=travis  下面给大家看一下 npm test 和npm run cov的指令分别是什么：  "scripts": {  "test": "./node\_modules/.bin/cross-env NODE\_ENV=travis jest --forceExit",  "cov": "./node\_modules/.bin/cross-env NODE\_ENV=travis npm run test -- --coverage && cat ./coverage/lcov.info | ./node\_modules/coveralls/bin/coveralls.js",  "debug": "nodemon ./index.js"  }   * 我比较青睐的单测框架是fb官方出品的jest，安装使用都很方便，只需要安装jest，你想要的功能全都有~而且jest很贴心的想到了--ForceExit 参数，如果你在单测的时候忘了srever.close了，没关系，jest保证你在持续集成的时候exit 0，无错误。 * cov 就比较简单了，在run测试用例的时候透传 --coverage参数就可以，这样就能生成覆盖度报告，然后运行coveralls的bin上传到coveralls.io。 触发持续集成所需要的操作可以在traavis-ci网站的setting中进行设置，而且也可以手动触发，亦可以设置cron job： [issue8-2](https://github.com/xtx1130/blog/raw/master/images/issue8/issue8-2.png) 我这个项目设置的cron job就是按天进行集成，它会每天定期根据.travis.yml进行一次集成。   **coveralls.io简介和使用**  **简介**  coveralls.io 是一款开源的上传覆盖度测试报告的工具，并根据上传的报告形成coverage角标，你在上面也可以清晰地看到每次提交的报告的对比，以及每次提交的覆盖率的增长率。 [issue8-3](https://github.com/xtx1130/blog/raw/master/images/issue8/issue8-3.png) 根据这张图片可以看到，我提交了三次，覆盖度增长到了百分之百。  **使用**  这个覆盖度测试报告使用起来也很简单，刚才的代码中也有使用方式了：  ./node\_modules/.bin/cross-env NODE\_ENV=travis npm run test -- --coverage && cat ./coverage/lcov.info | ./node\_modules/coveralls/bin/coveralls.js  在npm install comveralls --save-dev 之后，它会提供给你一个cli工具，通过调用这个工具，把lcov.info上传给coveralls.io，形成最终的测试报告。覆盖度测试一般用的包是istanbul，jest也不例外。它内部集成了istanbul，所以在需要忽略一些分支的时候，请使用istanbul的注释规则进行忽略。还有如果在单独使用istanbul进行覆盖度测试的时候，有一些es6语法不被支持，请先用babel转义或者利用babel-istanbul来进行覆盖度测试。 |