持续集成教程 1 通识科普

2017, Jun 1 by Tesla Ice Zhang

本教程系列将以 Travis CI 为主，我也不知道以后会不会讲 AppVeyor，我也不知道以后会不会讲 Circle CI 和 CodeShip。这篇文章你可以把它当成一个索引，我给出了使用 Travis 需要阅读的内容，读者可以根据自己的需求选择阅读文档的特定部分。

CI 能做啥

- 能帮你在云端自动编译项目
- 每次你推送代码就会触发编译
- 可以保留编译生成的目标文件
- 自动上传 release
- 编译失败发邮件提醒你
- 编译失败发 Slack 消息提醒你

等等功能（这些都是最基本的）

混开源社区的 friends 喜欢使用一些现有的 CI 服务，比如 Travis, AppVeyor, Circle CI, CodeShip 等， 公司企业喜欢自己写 CI 自己

用，因为这本来就是个高度定制的东西，要是你能提供高度定制的环境（比如装好了依赖的服务器）当然做 CI 就超级简单了。但是我们是混开源社区的 friends，所以没有这种操作，首选当然是 Travis CI。

理由： Linux + 自动部署

getting started

我才懒得写， RTFM

流程大概就是先注册 Travis 账号，然后在仓库里面写个配置文件 `.travis.yml` ，然后在 Travis 网站上选这个项目， 然后就可以等 build 了。 Travis 提供了一个 build 的 badge，可以放在 README 里面装逼（雾

可以让 Travis 自动测试，这样可以在别人发 pull request 的时候先通过 CI 的检验， CI 上编译测试通过了再 merge， 明显有安全感的多。

关于自动 release

就是每次 push 之后它不是会帮你编译吗，你可以让它把目标文件(artifact)上传到 release 里面。

这里推荐先搭建本地的 Travis 环境（就是 Ruby 环境 + `gem install travis` ）然后让它自己配置。

这样不会泄露你的 GitHub Token。教程见：

> RTFM

注意

一定要记得加上

```
on:
  tags: true
```

否则你每次推送都会收到一个 artifact 。加上之后就只有在创建 release 的时候才会上传。
于是就不需要在 release 的时候再上传目标文件了。
还可以本地 debug build ，然后让 CI 跑 release build ，不需要编译二遍。

Travis CI 的缺点

- 触发慢
- 触发慢
- 触发慢
- 触发慢

每次 push algo4j 后都能感受到 Travis CI 满满的恶意， 一般得等个一分钟才能看到它开始编译（这是因为需要 `sudo` ）。
不过它编译本身速度还是可以的，还有 CI 上下载各种源啊，clone 各种仓库啊也都是快到没朋友。
我部署了 Flutter 应用就是这样的，我先让 CI clone Flutter ，再编译 Flutter ，再下载 Android SDK ，再安装谷歌支持库， 再用 Flutter 编译我的项目，整个过程不到 4 分钟。
这个问题在 AppVeyor上得到了很好的解决， AppVeyor 非常非常快(触发和编译都快)，但是它是 Windows 的，而且各种预装的依赖略有些迷醉。 我自己也在用，而且非常舒服，这里就先不说了。

Tweet this

Top

创建一个 issue 以申请评论

Create an issue to apply for commentary

协议/License