Open Source Tips

By Eddie Jaoude

Table of Contents

Abstract

This book are tips of DOs & DONT of Open Source software.

Preface

Open Source is really dominating the industry. With so many projects out there from well known Organisations like Facebook, Twitter, NetFlix etc to passionate individuals around the world. How do you share your work & how do you decide who's work to use with so much choice.

As with technology that keep on changing, this book will need continually updating, suggestions & pull requests are welcome. Currently on version v0.1.34.master.

Source Repository is on GitHub https://github.com/eddiejaoude/book-open-source-tips

Introduction

With so many popular Open Source projects & many people contributing to Open Source, it would be beneficial to capture the pros & cons. Open Source not only benefits the community but also the authors as they get feedback on their project from a technical & lower level, more exposure & marketing, testing & bug fixing etc.

TIP	This book will continually be updated, please check the version 0.1.34.master. Any question please contact the author Eddie Jaoude on https://twitter.com/eddiejaoude.

DOs

Recommended...

Readme file

Documentation is usually left to last. At least have a `README.md` with some basic information, for example QuickStart.

Contribution file

One of the main benefits of an Open Source project & its community is Contributions. Lower the barrier to entry with a `CONTRUBTION.md` file in the root of your Open Source project. Read more about GitHub Contribution file

Code of Conduct

Your community needs to feel safe, diverse & included. Make sure you have a Code of Conduct for your project & community. Read more about Open Code of Conducts

GitHub template files

Issue & Pull Request templates really help keeping the project consistent & reminds people not to leave out certain useful information.

Commits

Commits should contain a single item, then your commit message will be easier to write & describe what you have done. There are many benefits to this:

- Looking back through the history will be clear & easy to understand
  - if you want to find something
  - undo / remove some work
- Automate the changelog generation as part of your build for tag & package etc

Little & often

Steady projects not only look more stable, but are generally more successful & are better for your health. Trying doing a little every week.

Plan ahead

Create or check tasks today ready for tomorrow. Two benefits of this, allowing you tomorrow to immediately hit the ground running & to digest the tasks overnight as you might make some final

tweaks.

Respond to Issues & Pull Requests

Always respond to Issues & Pull Requests in a timely fashion, ideally 48hours (even if it is with a comment of, I will take a look next week).

Pull Requests

If you spent the time doing the work, make sure you add a description to your your Pull Request to make it easier for the Review to review your work. Raise an Issue first so a plan of action can be discussed before you begin the work.

Reviews

Even if it is only you on the project, try to raise Pull Requests & get a friend to review it. This is so valuable.

Automated tests & Continuous Integration

Automate everything! This helps lower the barrier to entry & increase repeatability.

- Automated tests have many benefits & give confidence in the state & quality of the application:
  - Unit tests are great for design & architecture of functionality
  - Integration tests are great for the touch points
  - End-to-end tests are great for full appliaction testing & simulate the user
- Run the automated tests on Continuous Integration (CI) (for example TravisCI)
- When autonated tests are successful, deploy the application aka Continuous Delivery (CD)

Note: This include database schema migrations etc

Get a Prototype to your users quickly

Feedback as soon as possible. Quick & dirty prototype in front of some of your users will give you instant feedback & direction. Remember to make it clear it is a prototype.

Work at your optimal time

People work better at different times of the day & night. Work at your most efficient & optimal time. Even if that is 11pm at night or 4am, try utilise your most productive time.

Never enough time

We all have the same 24 hours in a day available to us. Its what you do with it that counts. Try to

find a small amount of time per day, even 10 minutes when you are on the toilet - yes you heard me right "on the toilet". Multi tasking in that situation is possible, but trying to work while watching TV is very unproductive.

Make it open source (public) from day 1

I do NOT understand people who say "I will make it public when it is finished". First of all, it will never be finished, secondly you then do not gain all the benefits from Open Source.

DONTs

NOT recommended…

Big bang projects

Dont try to complete the project in a manic weekend, then ignore it for the next year. This is not good for your health nor does it look good for your project from the community's view.

God commits

God commits or big bang commits are useless & confusing. These are terrible for you & the community for so many reasons.

God Pull Requests

Similar to "god commits", god pull requests are a bad idea. Reviewing god or big bang pull requests are not only annoying & painful, but leave room for skim reviewing & therefore mistakes. A pull request should be a single feature. No one ever complained that a Pull Request was too small.