Open Source Tips

By Eddie Jaoude

Table of Contents

Abstract

This book contains some common DOs & DON'Ts for Open Source software.

Preface

The Open Source community is thriving. Each day the number of Open Source projects grows, as does the army of contributors that maintain them. While this is exciting for the industry, it can be daunting as a developer new to the community. This book aims to provide some tips for newcomers to help them avoid the pitfalls of Open Source development and learn from the community's collective wisdom.

As the ancient proverb goes, *Time and tide [and technology] wait for no man*. And to the best of our ability, neither will this book. Remember to check the version number for updates! We're currently on v0.1.64.master.

We would love your help in keeping this book updated. Your comments, suggestions and pull requests are most welcome. You can find the repository on GitHub: https://github.com/eddiejaoude/book-open-source-tips.

If you have any questions, please contact the author, Eddie Jaoude on https://twitter.com/eddiejaoude.

Introduction

Open Source is dominating the software industry. Its champions include well known organisations like Facebook, Twitter and Netflix, but more significantly, an army of passionate individual developers around the world. Their efforts have impacted almost every part of computer science, culminating in millions of open source projects, with billions of lines of code!

While this abundant ecosystem has been of huge benefit to the whole industry, it can also make it difficult for newcomers to know where to start. If you're a newcomer, you might be faced with questions such as *"How can I contribute to the Open Source community?"* Or, *"How do I choose between so many competing projects?"*. The following DOs and DON'Ts aim to address some of those basic questions, plus some pointers for aspiring Open Source developers.

Let's dive right in.

DOs

Readme file

Documentation is usually left to last. Start every project with at least a `README.md` with some basic information, for example a QuickStart guide, if you change features or functionality, try at least update this with your commits.

Code block

Use syntax highlighting within code blocks in documentation to make it easier to read.

Syntax highlighting

You can add an optional language identifier to enable syntax highlighting in your fenced code block.

For example, to syntax highlight Ruby code:

```
```ruby
require 'redcarpet'
markdown = Redcarpet.new("Hello World!")
puts markdown.to_html
```
```

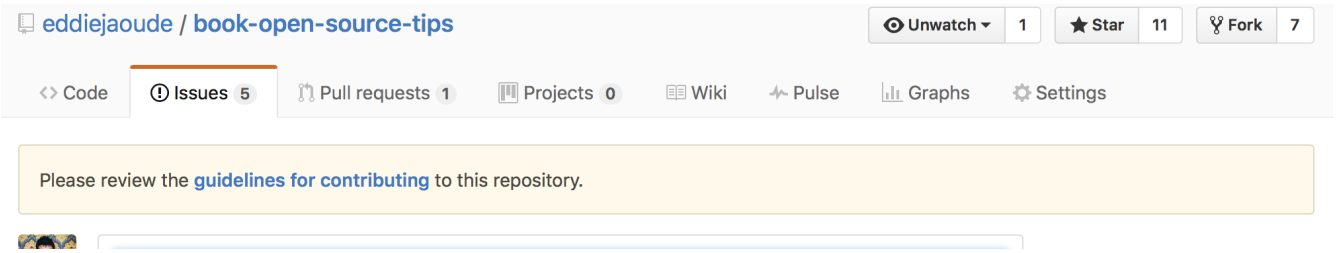
```ruby
require 'redcarpet'
markdown = Redcarpet.new("Hello World!")
puts markdown.to_html
```

GitHub syntax highlightingGitHub syntax highlighting

Contribution file

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

> Oftentimes open source projects place a CONTRIBUTING file in the root directory. It explains how a participant should do things like format code, test fixes, and submit patches. Here is a fine example from puppet and another one from factory_girl_rails. From a maintainer's point of view, the document succinctly communicates how best to collaborate. And for a contributor, one quick check of this file verifies their submission follows the maintainer's guidelines.
>
> — GitHub, GitHub Contributing Guidelines



GitHub Contributing GuidelinesGitHub Contributing Guidelines

Code of Conduct

Your community needs to feel safe, diverse & inclusive. Make sure you have a Code of Conduct for your project & community. Read more about Contributor Covenant - A Code of Conduct for Open Source Projects

> Open Source has always been a foundation of the Internet, and with the advent of social open source networks this is more true than ever. But free, libre, and open source projects suffer from a startling lack of diversity, with dramatically low participation by women, people of color, and other marginalized populations.
>
> — Contributor Covenant, Brief overview of the problem

> An easy way to begin addressing this problem is to be overt in our openness, welcoming all people to contribute, and pledging in return to value them as human beings and to foster an atmosphere of kindness, cooperation, and understanding.
>
> — Contributor Covenant, Brief overview of the solution

GitHub template files

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

Licensing

It is not required to select a license, however it is very useful to do so. GitHub have created an informational website to help you choose a license

Git commit

Git commits should be small and atomic, be it a single change or small feature. This will make your commit messages easier to write and and changes will be grouped logically. 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 and 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. There are two benefits of this, allowing you tomorrow to immediately hit the ground running and to digest the tasks overnight as you might make some final tweaks.

Issue / Task

Keep these small. Tackling a large piece of work is always daunting and more difficult to find the time. The smaller the tasks, the more likely it is to be done and the lower risk it will be. But remember the task still needs to provide value to the project.

Include diagrams, screenshots, anything visual to help describe the issue & changes.

Always respond - Issue and Pull Request

Always respond to Issue and Pull Request in a timely fashion, ideally within 24 hours (even if it is with a comment acknowledging you have at least read the issue & will respond fully in 3 days). This manages expectations for when the contributor can expect a full response.

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 reviewer to digest your work. Raise an Issue first so a plan of action can be

discussed before you begin the work and to remind yourself of the goals set out in that task.

Pull Requests should be linked to the original Issue it is trying to solve. This can be done with using # followed by the **Issue No**, e.g. #123. The Issue description will contain the information before, therefore the Pull Request description should contain the information after the changes. Include visual material too, for example diagrams & screenshots.

Review

Even if it is only you on the project, try to raise Pull Requests & get a friend to review it. This approach is invaluable as a second pair of eyes often picks up oversights.

Automation - Tests, Continuous Integration (CI), Continuous Deployment (CD)

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 application testing & simulate the user
- It should be very easy to setup up a local environment & run all these tests
- Run the automated tests on Continuous Integration (CI) (for example TravisCI) after someone has pushed their code changes to a feature branch
- When automated tests are successful, deploy the application aka Continuous Delivery (CD)

Note: This includes database schema migrations, asset building and anything that is required by the end product

Prototyping - Get a prototype to your users quickly

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

Optimal time - work when at your best

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

Not enough time

We all have the same 24 hours in a day available to us. It's 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.

Codebase improvements - Leave the codebase better than you found it

Many code repositories (mostly Closed Source ones) go from bad to worse. On the other hand, Open Source projects tend to do the complete opposite due to it being in the public eye. Even the smallest of improvements add up and really help - **no improvement is too small**.

DON’Ts

Big bang project

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 commit

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

God Pull Request

Similar to "god commit", god pull request are a bad idea. Reviewing god or big bang pull request 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.

CV Driven Development

Most of us have heard of Test Driven Development (TDD). Do not fall in to the trap of CV Driven Development (CVDD). It is good to push your skills with new technology but do a little at a time to reduce the risk. Do not try every new technology you want to learn on the same project, this is very high risk, with the mostly likely outcome of little result & therefore frustration. Approximately 10% per of new technology per project is a good & safe amount.