
Open Source Support
Andrew Hooker

Open Source Support

Andrew Hooker

This book is for sale at http://leanpub.com/os-support

This version was published on 2014-11-12

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Tweet This Book!

Please help Andrew Hooker by spreading the word about this book on Twitter!

The suggested hashtag for this book is #os-support.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=#os-support

Contents

About the Author

Andrew Hooker is a support engineer for Spree Commerce. He was a user of the project, and became actively involved in the open source support community before finally being hired on staff. He has contributed to a variety of open source projects, and communicates frequently with a number of other project maintainers for ruby projects.

About the Book

This book is intended to be a resource for anybody who is making code freely available on the internet. I do my best to lay out resources you can use to do free support, as well as ways to make money from it.

About the Writing Style

I'm a big fan of the Minimal Viable Product principle. This means that I've written initial content for all of my chapters, and I am offering that to you at this time. I have my own roadmap, but I hope to get feedback on what readers think I should improve, so that I can support you in the best way possible. I will do my best to include detailed information in the Changenotes that gets sent by Leanpub when the book is updated, so that you've got an idea how things are being improved.

Teaching a Programmer to Code Recursive Fish

Getting users to file quality issues

Getting people to complain about issues in your project is easy, assuming you have people using your project at all. Users will, as a default behavior, provide the least information possible, as providing anything additional takes work. As someone who is supporting open source, it is in your best interest to migrate your users from this default behavior to one which is more helpful to you.

> There is no right or wrong way to do this. The nature of the project will determine a lot of how you handle support, and how much time and effort you can invest into bringing new contributors into the fold. Remember, every active contributor can also be helpful in encouraging this process.

Here are some simple things that you can do to start to change the mindsets of your users:

- Have a well defined contributors guide. Your contributors guide should include a section on how to file an issue. Some things you might want to require:
  - A comprehensive list of steps to reproduce the issue.
  - A description of expected behavior vs actually behavior.
  - Programming Language Version
  - Software and Dependency Versions
  - Any error messages
  - Steps to Reproduce
  - Tested issue against latest version of your project
  - Submit a failing regression test
- Close issues that don't meet the requirements you have provided with a link to the contributors guide
- Respond quickly, thanking them for providing a proper issue, and give them an estimated timeline in which you will look at it
- Review incoming communication as it comes in; sometimes a few minutes on an issue when it comes in can save lots of time later.

> *"My workflow has been altered recently to triage all new emails that come into my inbox. That way, I provide more immediate feedback on the little niggling issues that people file rather than having them linger forever and ever."* - Ryan Bigg[a], Community Manager at Spree Commerce

[a] https://github.com/radar

Getting users to fix their own issue

Getting your users to submit a high quality issue is a good first step, and will reduce the work you have to do; however wouldn't it be great if you didn't have to investigate the issue at all? The best time to get somebody involved contributing to your project is to help them fix their own problem. Once they see how easy you are (hopefully) making it to contribute, hopefully they'll contribute again in the future without help! There are some things that you may have to do in order to get people to submit their first open source pull request:

- Prioritize first time contributors. This may seem counter-intuitive, but there have been a number of projects that I've contributed to that I only sent one pull request because they took too long to respond to it. When somebody goes to the effort to send you code, you should respond to that effort the best you can.
- Answer questions that aren't strictly relevant to your project; yes, there are better resources to teach them how to create a git branch, but if you force them to read a long manual[1] they're likely to find something better to do with their time. Feel free to provide the reference material, but giving them simple instructions to create their branch, or write tests, or the most elegant way to solve a simple problem will make them more likely to complete the fix.
- Put policies into place about what is accepted and what isn't.

[1] http://git-scm.com/docs/git-branch

- Provide a good venue to discuss feature additions.
- Offer "office hours" or slots for pair programming to work on issues. You'll meet new people, and get an issue fixed at the same time.
- Provide quick feedback, no matter what it is. Odds are it falls into one of three categories:
  - There's some reason that there's no way their pull request will ever be merged (feature that you don't want, impossible without adding an unresonable dependency). Thank them for the effort, give them a real reason why you can't use it.
  - The pull request needs improvement before it can be merged (no tests, style conflicts, bugs, etc.). Again, thank them for their contribution; point them in the direction of what needs to be done to get it ready. If the change is simple, consider fixing it for them, so that your project benefits from the fix right away, and they benefit from seeing their work recognized. Make sure you told them what you did to fix it, so that they can hopefully do it on their own the next time.
  - The pull request is ready to go as is. Merge it! Thank them! Recognize them however you can! I usually share on twitter when I've helped somebody get their first Spree Commerce commit. When we do a gem release, we typically try and mention anybody in the release notes who had contributed for the first time in that release.

Getting users to keep coming back

You've gotten your user to file a high quality issue report, and then submit a pull request to fix it. If you can get them to keep doing this, time after time, you might have a potential core contributor, who can shoulder some of the support workload. So how do you get them to keep coming back?

> *"Review pull requests in a timely manner. I know I'll not come back if my pull request doesn't even get a maintainer comment. I had a homebrew formula sit in the queue for months without comment. I finally gave up and vowed not to write another."* - Robert Rouse, Open Source Contributor[a]

[a] http://twitter.com/rerouse