How open source works (or doesn't work)

A Technically book review?!

Justin
Aug 10, 2020

This week's issue is going to be a little different. I'm going to dive deeper into a concept we've already covered - open source software - and talk about a new book I've had the pleasure of reading. It's from Stripe Press, it's called *Working In Public*, and it's all about how open source software gets built and maintained.

Working in Public's author is Nadia Eghbal, a researcher / writer / smart person who has spent time working in the belly of the open source beast itself, GitHub. She's uniquely qualified to write about the topic, and brings a lot of interesting data to the table. You can get *Working in Public* on Amazon here.

The unlikelihood of open source

When we covered open source a few months ago, the focus was mostly on what it is:

> *Open source technically means that an app's code is publicly available, but more generally refers to the movement of communities of developers building transparent, open codebases free for everyone to use.*

We also saw that *tons and tons* of popular software is open source - I don't know of a single startup on the planet that's not using open source libraries to power their apps. Some of the biggest, most popular pieces of developer tooling are completely free and open:

- Programming languages: Python, Rust, etc.
- Web: Node.js, Express, Django, Requests, etc.
- Frameworks: React, Vue, Angular, etc.
- Machine Learning: Tensorflow, PyTorch, Keras, etc.

So if most of the websites you visit and applications you use depend on packages like these, you'd imagine that there are a bunch of really smart developers working full time on them, and getting paid pretty well. But you'd be wrong - it's actually quite the opposite. In fact, most open source software is built and maintained by developers **in their free time, for free**.

> 🔍 **Deeper Look**🔍
>
> These days, a lot of open source software actually comes from the corporate side - e.g. React was built and open sourced by Facebook, Tensorflow by Google, etc. This messes with the model a little bit, as maintainers are often employed by these companies.

> 🔍 **Deeper Look**🔍

If you're thinking "well that doesn't make sense" then you're right - the dissonance between how open source is *built* vs. how it's *used* creates a lot of tension, and some interesting community dynamics. How did we get here? Why is there such a disconnect between how important open source software is, and the models that exist to produce it?

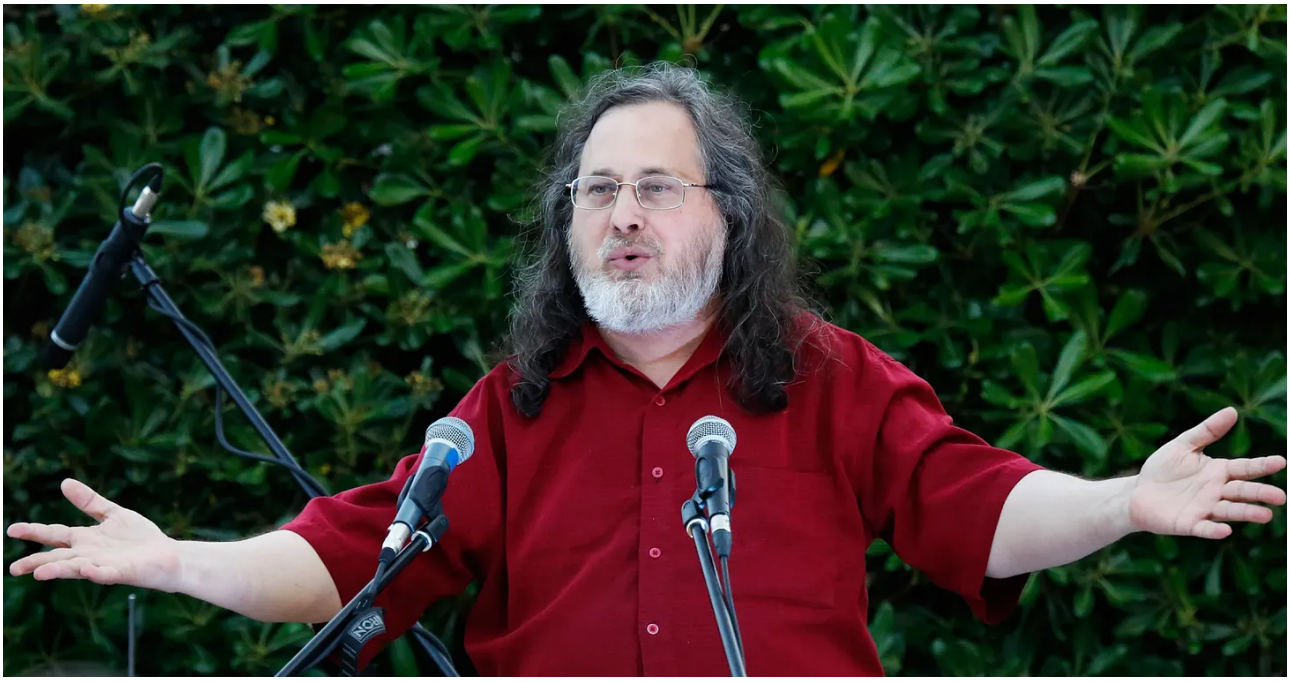
Open source is really the hacker movement

In *Working in Public*, Nadia argues that the key to understanding all of this is open source's history. Open source is really a brainchild of the **hacker movement**, which emphasizes software as a tool for freedom and equality:

> *The term "hacker" was popularized by author Steven Levy, who memorably captured a portrait of the 1980s hacker generation in the book Hackers: Heroes of the Computer Revolution. Levy profiles a number of well-known programmers of the time, including Bill Gates, Steve Jobs, Steve Wozniak, and Richard Stallman. He suggests that hackers believe in sharing, openness, and decentralization, which he calls the "hacker ethic."*
>
> *(p. 23)*

If you haven't heard of Richard Stallman before (affectionately referred to as RMS), he's quite the character - he authored the popular GNU operating system,

and is generally kind of like an activist (?) supporting free software and decentralization.



This is why open source kind of just *happens* without a ton of economic incentive - developers are building because they *want to*, and because they want their software to be free. The engineers working on React aren't exactly OG hackers, but in theory, they share a lot of the principles that drove the free software movement.

A lot of open source really is driven by these larger than life characters, many of who struggle with the burden of maintaining massive projects (more on that below). Linus Torvalds, the creator of the Linux operating system and Git, is a notorious asshole on the forums. Even Stallman had to step down from his post at MIT for questionable comments at the end of last year.

Maintaining open source is hard AF

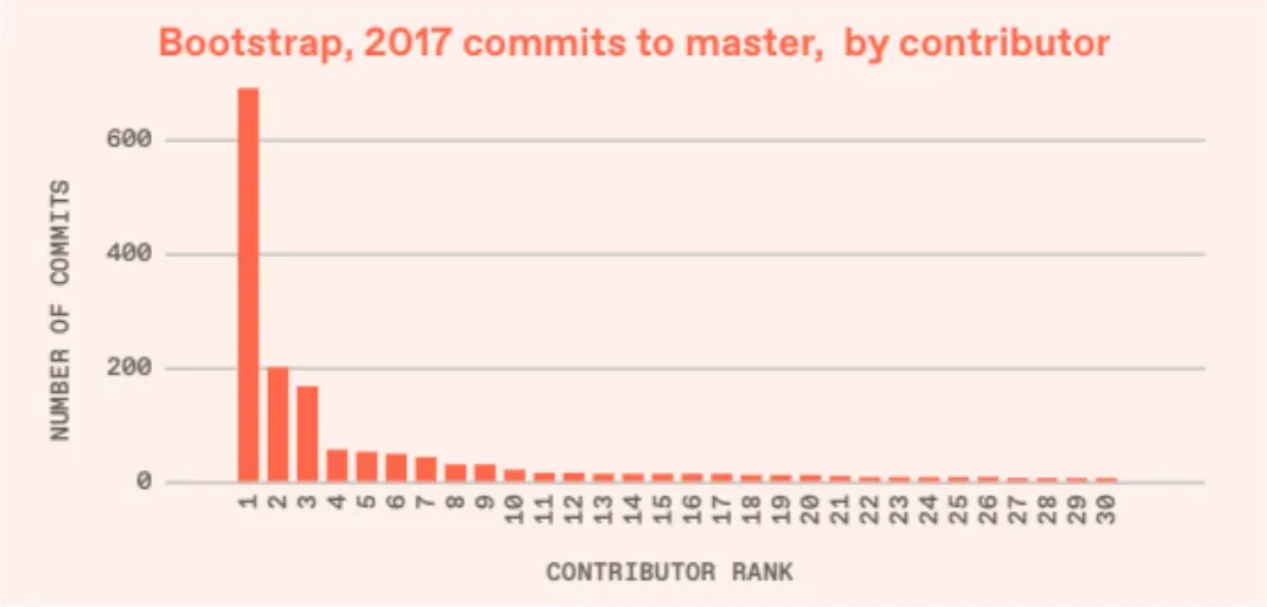
As you can imagine, having millions of developers use a project that's run and maintained by only a few people doesn't always work very well.

> 🧠 **Jog your memory** 🧠
>
> Open source software typically has a leadership group of *maintainers*. They're developers that have been with the project for a while, and run the

> show - they decide what needs to get worked on, merge in suggested changes, and most importantly, write a ton of code themselves.
>
> 🧠 **Jog your memory** 🧠

In theory, the idea of open source is that *anyone* can contribute. Literally, if you wanted, you could waltz over to the React repo on Github, write some code, get it reviewed, and have someone merge it in. And this does indeed happen - even though React was built and maintained by Facebook, random unrelated developers do make contributions. But in *Working in Public*, Nadia shows that this is *not* the norm.

The reality is that for most open source projects - especially popular ones - the majority of work gets done by a few very focused developers, usually the ones who set up the project in the first place. A good example is Bootstrap, a popular framework for styling sites – the overwhelming majority of contributions to the codebase come from a few developers.



(Working in Public, p. 10)

The top developers commit a *ton*, and the rest of the community - much less so. This pattern holds true for almost all of the open source projects that have gained meaningful traction, and it puts a **massive burden** on these maintainers to hold up the projects themselves. And remember: they're usually not getting paid.

Actually contributing isn't the only thing that takes up maintainers' time: they also need to deal with all of the contributions that other, less experienced developers make. Now that most open source projects are on GitHub, it's really easy for any n00b (myself included) to suggest changes, however weak, to a codebase - and the maintainers of a project need to deal with the overhead of reviewing, improving, and merging those changes. A lot of the time, novice developers don't take the time to test or explain their changes either.

Nadia sums up the problem:

> *Taken together, it's the perfect storm of drive-by users – most well intentioned, others not so much – flooding the channels of single maintainers, using a platform that's designed to encourage this behavior, and shielded by social norms that explicitly discourage pushing back on inbound requests.*
>
> *(p. 41)*

TL;DR: being an open source maintainer is hard, and getting harder.

Where open source goes from here

Open source is getting more and more important, but harder and harder to maintain - at some point, things are going to stop working. The conclusion of *Working in Public* explores a few ways that platforms like GitHub might be able to improve things via filtering mechanisms, but the whole book leaves you wondering if this is really a solvable problem or not.

In the meanwhile, if you find yourself using an open source library a lot, try contributing to the docs, helping promote it, or donating (if they accept donations). And if you want to contribute code, put yourself in the shoes of the maintainer who's going to need to review your suggestions.

I've only covered a few chapters from *Working in Public*, but Nadia explores a lot of other cool stuff in the book:

- Where open source fits in with existing economic theory, and the "tragedy of the commons" concept

- The anatomy of an open source project: more detail on maintainers, contributors, and different project types
- Measuring the economic value and production costs of open source, plus the role of money in the ecosystem

You can purchase *Working in Public* on Amazon here – it's quite good, and I do not get any sort of referral fee. So you can trust me. With your life.

Comments

Write a comment...
