|  |
| --- |
| layout: post |
| title: Open Source (Almost) Everything |

# {{ page.title }}

22 Nov 2011 - San Francisco

When Chris and I first started working on GitHub in late 2007, we split the work into two parts. Chris worked on the Rails app and I worked on Grit, the first ever Git bindings for Ruby. After six months of development, Grit had become complete enough to power GitHub during our public launch of the site and we were faced with an interesting question:

Should we open source Grit or keep it proprietary?

Keeping it private would provide a higher hurdle for competing Ruby-based Git hosting sites, giving us an advantage. Open sourcing it would mean thousands of people worldwide could use it to build interesting Git tools, creating an even more vibrant Git ecosystem.

After a small amount of debate we decided to open source Grit. I don't recall the specifics of the conversation but that decision nearly four years ago has led to what I think is one of our most important core values: open source (almost) everything.

## Why is it awesome to open source (almost) everything?

If you do it right, open sourcing code is **great advertising** for you and your company. At GitHub we like to talk publicly about libraries and systems we've written that are still closed but destined to become open source. This technique has several advantages. It helps determine what to open source and how much care we should put into a launch. We recently open sourced Hubot, our chat bot, to widespread delight. Within two days it had 500 watchers on GitHub and 409 upvotes on Hacker News. This translates into goodwill for GitHub and more superfans than ever before.

If your code is popular enough to attract outside contributions, you will have created a **force multiplier** that helps you get more work done faster and cheaper. More users means more use cases being explored which means more robust code. Our very own [resque](https://github.com/defunkt/resque) has been improved by 115 different individuals outside the company, with hundreds more providing 3rd-party plugins that extend resque's functionality. Every bug fix and feature that you merge is time saved and customer frustration avoided.

Smart people like to hang out with other smart people. Smart developers like to hang out with smart code. When you open source useful code, you **attract talent**. Every time a talented developer cracks open the code to one of your projects, you win. I've had many great conversations at tech conferences about my open source code. Some of these encounters have led to ideas that directly resulted in better solutions to problems I was having with my projects. In an industry with such a huge range of creativity and productivity between developers, the right eyeballs on your code can make a big difference.

If you're hiring, **the best technical interview possible** is the one you don't have to do because the candidate is already kicking ass on one of your open source projects. Once technical excellence has been established in this way, all that remains is to verify cultural fit and convince that person to come work for you. If they're passionate about the open source code they've been writing, and you're the kind of company that cares about well-crafted code (which clearly you are), that should be simple! We hired [Vicent Martí](https://github.com/tanoku) after we saw him doing stellar work on [libgit2](https://github.com/libgit2/libgit2), a project we're spearheading at GitHub to extract core Git functionality into a standalone C library. No technical interview was necessary, Vicent had already proven his skills via open source.

Once you've hired all those great people through their contributions, dedication to open source code is an amazingly effective way to **retain that talent**. Let's face it, great developers can take their pick of jobs right now. These same developers know the value of coding in the open and will want to build up a portfolio of projects they can show off to their friends and potential future employers. That's right, a paradox! In order to keep a killer developer happy, you have to help them become more attractive to other employers. But that's ok, because that's exactly the kind of developer you want to have working for you. So relax and let them work on open source or they'll go somewhere else where they can.

When I start a new project, I assume it will eventually be open sourced (even if it's unlikely). This mindset leads to **effortless modularization**. If you think about how other people outside your company might use your code, you become much less likely to bake in proprietary configuration details or tightly coupled interfaces. This, in turn, leads to cleaner, more maintainable code. Even internal code should pretend to be open source code.

Have you ever written an amazing library or tool at one job and then left to join another company only to rewrite that code or remain miserable in its absence? I have, and it sucks. By getting code out in the public we can drastically **reduce duplication of effort**. Less duplication means more work towards things that matter.

Lastly, **it's the right thing to do**. It's almost impossible to do anything these days without directly or indirectly executing huge amounts of open source code. If you use the internet, you're using open source. That code represents millions of man-hours of time that has been spent and then given away so that everyone may benefit. We all enjoy the benefits of open source software, and I believe we are all morally obligated to give back to that community. If software is an ocean, then open source is the rising tide that raises all ships.

## Ok, then what shouldn't I open source?

That's easy. Don't open source anything that represents core business value.

Here are some examples of what we don't open source and why:

* Core GitHub Rails app (easier to sell when closed)
* The Jobs Sinatra app (specially crafted integration with github.com)

Here are some examples of things we do open source and why:

* Grit (general purpose Git bindings, useful for building many tools)
* Ernie (general purpose BERT-RPC server)
* Resque (general purpose job processing)
* Jekyll (general purpose static site generator)
* Gollum (general purpose wiki app)
* Hubot (general purpose chat bot)
* Charlock\_Holmes (general purpose character encoding detection)
* Albino (general purpose syntax highlighting)
* Linguist (general purpose filetype detection)

Notice that everything we keep closed has specific business value that could be compromised by giving it away to our competitors. Everything we open is a general purpose tool that can be used by all kinds of people and companies to build all kinds of things.

## What is the One True License?

I prefer the MIT license and almost everything we open source at GitHub carries this license. I love this license for several reasons:

* It's short. Anyone can read this license and understand exactly what it means without wasting a bunch of money consulting high-octane lawyers.
* Enough protection is offered to be relatively sure you won't sue me if something goes wrong when you use my code.
* Everyone understands the legal implications of the MIT license. Weird licenses like the WTFPL and the Beer license pretend to be the "ultimate in free licenses" but utterly fail at this goal. These fringe licenses are too vague and unenforceable to be acceptable for use in some companies. On the other side, the GPL is too restrictive and dogmatic to be usable in many cases. I want everyone to benefit from my code. Everyone. That's what Open should mean, and that's what Free should mean.

## Rad, how do I get started?

Easy, just flip that switch on your GitHub repository from private to public and tell the world about your software via your blog, Twitter, Hacker News, and over beers at your local pub. Then sit back, relax, and enjoy being part of something big.

[Discuss this post on Hacker News](http://news.ycombinator.com/item?id=3267432)