

Licensing: How to open-source your work

Jeffrey Fisher Skylar

Introduction

Presenters

What this presentation is

We intend that after this presentation you:

- ▶ Know what permissions popular licenses grant you
- ▶ Have a good idea of what licenses you may want to use for your projects
- ▶ (maybe) Have some understanding of the issues if you want to make money from open-source work

What this presentation is **not**

We will avoid promoting a particular choice. However, we may mention pros and cons of particular choices. Also, this presentation is focused on free and open-source licenses.

We are not lawyers. We focus on well-understood truths, or point out when something is not well-understood. We cite our sources. If you want custom license terms or to use less well-understood licenses, you should probably consult a lawyer.

Terminology

Free and Open Source Software (FOSS)

The free software movement and the open-source software movement are separate.

Most licenses that fit the free software definition also fit the open-source software definition, and vice versa. (“Categories of Free and Nonfree Software,” n.d.)

Copyright, works, licenses, oh my!

No License

No License

Just stick it online, right?

No license = All rights reserved

When you make a creative work (which includes code), the work is under exclusive copyright by default. Unless you include a license that specifies otherwise, nobody else can copy, distribute, or modify your work without being at risk of take-downs, shake-downs, or litigation. Once the work has other contributors (each a copyright holder), “nobody” starts including you. (choosealicense)

- ▶ If you are the only user, this is probably fine
- ▶ If you want other people to reuse and share your work, this is probably not what you want: pick a license to specify your terms more clearly

Permissive licenses

I want it simple and permissive

Permissive licenses let people do almost anything they want with your project, like making and distributing closed source versions. (choosealicense)

- ▶ Includes public domain (there is no copyright owner)
- ▶ Easier to use and distribute
- ▶ Harder to retain control and share improvements

Examples

- ▶ MIT, BSD, Apache
- ▶ Chromium, Apache Web Server

Copyleft licenses

I care about sharing improvements

Copyleft licenses lets people do almost anything they want with your project, /except/ distributing closed source versions. (choosealicense)

- ▶ Changes must redistributed with a copyleft license and with source code
- ▶ Harder to use and distribute
- ▶ Easier to retain control and share improvements

Examples

- ▶ GPL, AGPL, LGPL (variants for linking code and sharing it over server applications)
- ▶ Linux, Bash, Emacs

Licenses for non-software works

Creative Commons

- ▶ CC0 = public domain
- ▶ CC-BY = attribution (like MIT)
- ▶ CC-BY-SA = share alike (like GPL)

Hardware

TODO

Working with others and in a community

Making a new project

- ▶ Your license conveys to the community how you intend your project to be used and shared

Contributing to an existing project

- ▶ Review the license
- ▶ Check for contributor agreements:
 - ▶ GNU Emacs: Contributing to Emacs itself or the official package repository (ELPA) requires you to assign your copyright to the FSF.
 - ▶ Developer Certificate of Origin (DCO): Contributing to the Linux kernel requires signing this. In brief, the certificate verifies that the user wrote the open-source changes and allows the project to use them in a way consistent with the license.
<https://developercertificate.org/>
 - ▶ Contributor License Agreement (CLA): Can be very restrictive. CLAs typically allow the owner of the project (typically a foundation or a company) to relicense your patches. CLAs are controversial because if the company makes the project proprietary, you no longer own the open-source code that you contributed. The company does. This has spoiled the relationship between a few communities already.

Dependencies and using other people's code

- ▶ License compatability: your code should respect the licenses of code you depend on
- ▶ Rule of thumb:
 - ▶ copyleft code can only be reused in copyleft code
 - ▶ permissive code can be reused almost anywhere
 - ▶ proprietary code may have restrictions (frequently noncommercial)

You can check the licenses of your code's dependencies for various languages:

- ▶ Rust
- ▶ Python
- ▶ Javascript

Non-FOSS licenses

Non-FOSS licenses

Just because you can look at the code doesn't mean it's FOSS

Proprietary licenses

- ▶ Leaked/reverse-assembled proprietary code is still proprietary

Source-available licenses

An important requirement of FOSS is that it can be used for commercial purposes.

- ▶ Business Source License (“permissive”)
 - ▶ Releases are source available with a promise to open-source it later
 - ▶ QT, MariaDB, Codon
- ▶ Server Side Public License (“copyleft”)
 - ▶ Copyleft extends much farther than the AGPL
 - ▶ Withdrawn from consideration to the Open-Source Initiative: restricts the right to make use of the program for any purpose

Ethical-source license

An important requirement of FOSS is that it can be used for any purpose.

Ethical source licenses have provisions that restrict uses for ethical purposes.

- ▶ JSON license: “The Software shall be used for Good, not Evil.”
 - ▶ But what counts as an “ethical” usage?
 - ▶ Maybe unenforceable

Practical and economic considerations

Practical and economic considerations

- ▶ Whatever license you pick, consider the community
- ▶ If you need a legally-contentious or custom license, consult a lawyer
- ▶ Now you know the basics of software licensing

Bibliography

Bibliography

“Categories of Free and Nonfree Software.” n.d. Free Software Foundation. <https://www.gnu.org/philosophy/categories.html>.