How to Properly Open Source Software

Lessons Learned from the Linux Foundation

Hart Montgomery

CTO, Linux Foundation Decentralized Trust

Talk Outline

Open Source Background

- Why open source?
- The Linux Foundation, or why you should care about my opinion

Open Source Table Stakes

- Open source software licensing
- IP protections for software
- Basic best security practices

Open Governance and Communities

- Open source governance
- Building an open community

I'm not a lawyer, this is not legal advice, and opinions are my own.

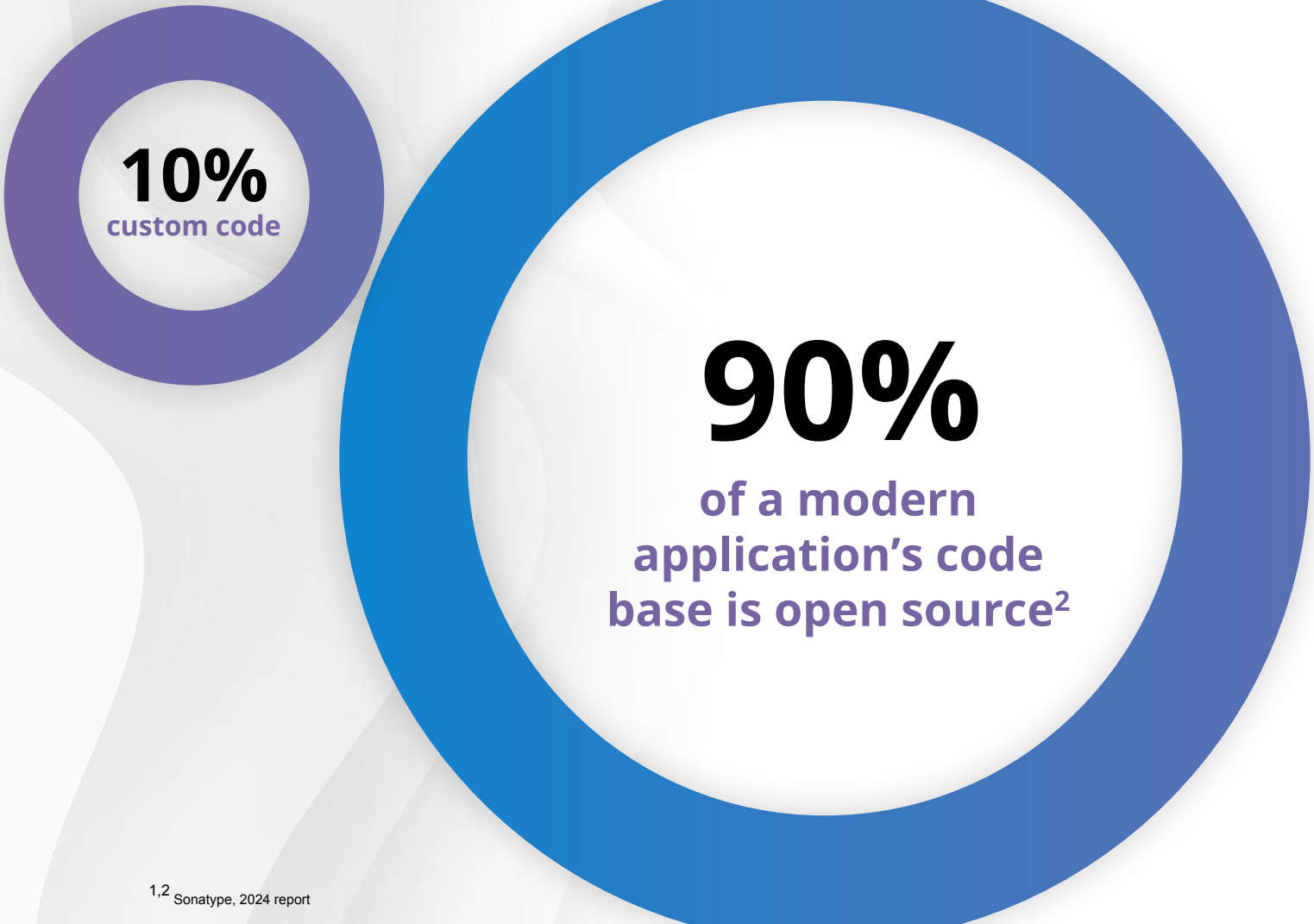
Why Do We Open Source Code?

So others can **examine**, **verify**, and **learn** from our code:

- Verifying the software allows **outside people to trust it**.
- Open sourcing has many benefits for things like **software security**, where others can report bugs and vulnerabilities.

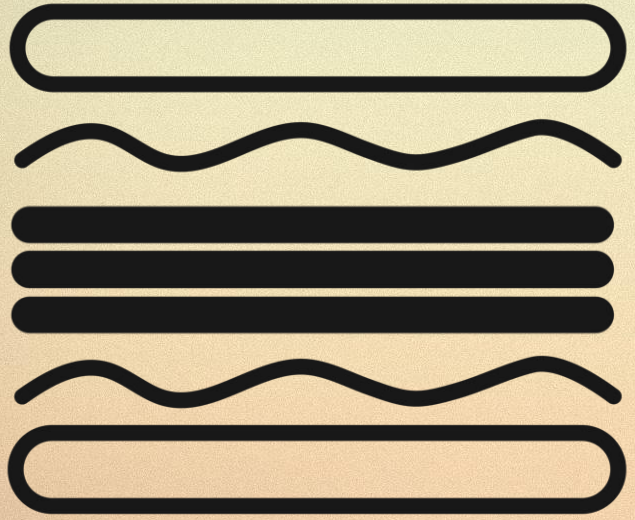
So others can **use** and even **contribute to** our code:

- There are many **economic efficiencies** of open source: working together in the open, we can **build core technology more efficiently**, **even if we are still competing**!
- A large ecosystem around a piece of open source code ensures that **even if one organization goes away**, **development continues**.



[1,2] Sonatype, 2024 report

Building a Modern Platform or Application



Use Open Source Libraries to Solve Problems

Open Source Code (~70%)

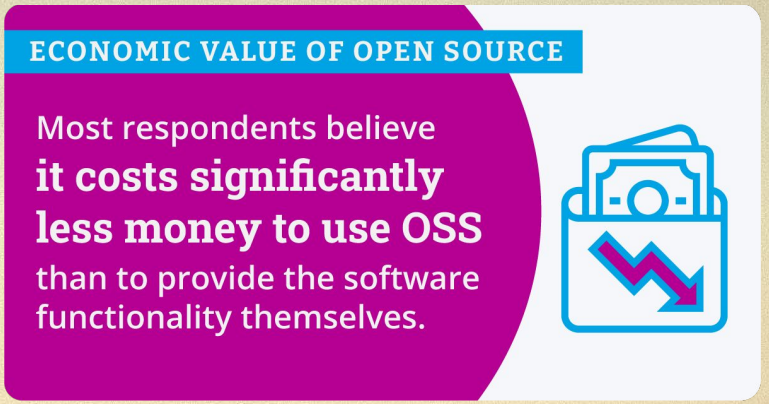
Write Custom Code

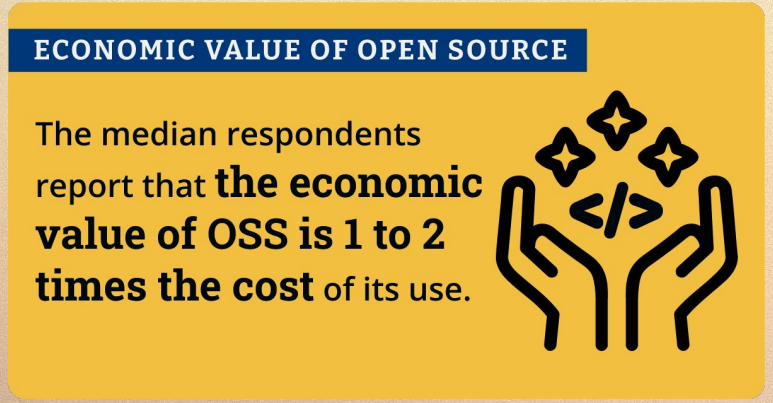
Custom Code (~10%)

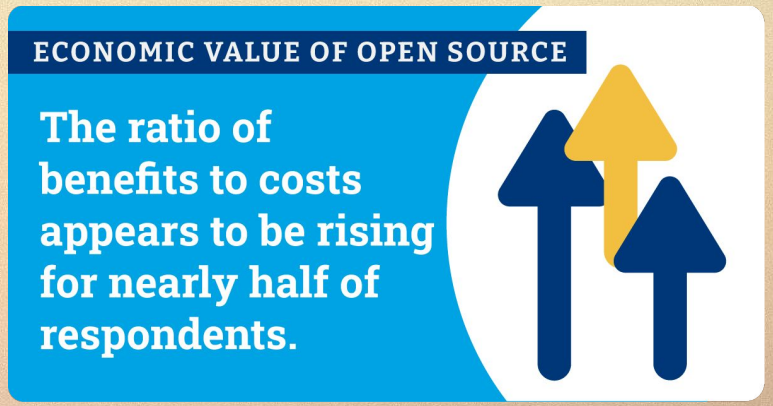
Choose a Framework

Open Source Code (~20%)

Economic Value of Open Source







Source: Economic Value of Open Source, Global Spotlight 2023

Open source is most efficient when multiple companies, entities or people **collaborate to build software that they all need** in the open.

This is just **decentralized development!**

The Linux Foundation solves **decentralized development** for open source code**.**

When multiple companies, entities, or individuals want to collaborate on open source software but don’t trust one single party to own the code, they turn to the Linux Foundation.

THE LINUX FOUNDATION

The Linux Fo **development.**

When multiple com want to collaborate on open source softwa party to own the code, th tion.

THE LINUX F

We are behind some of the most critical projects in the world

Across our foundations: hundreds of projects, lines of code in the billions

11.7M
Lines of Code Added Weekly

9.4M
Lines of Code Removed Weekly

208,660
Contributing Developers

12,435
Contributing Companies

16,618
Repositories

9.1M
Commits

960,610
Pull Requests

1.3M
Builds Monitored

859,150
Logged Issues

5.2B
Container Downloads

1.6M
Group Chat Messages

3.6M
Group Email Messages

4,532
Scanned Repositories

282,342
Open Vulnerabilities

62,379
Recommended Fixes

36,163
Vulnerabilities Fixed

31,496
CLA Contributors

26,998
Community Meetings

Core Ethereum

besu

Tools for Ethereum

Hyperledger WEB3J

PALADIN

Hyperledger FIREFLY

Hyperledger CACTI

Ethereum-Associated

Hiero

Hyperledger SOLANG

LF DECENTRALIZED TRUST

Open Source Software: Table Stakes

What is Open Source Software?

Open source software (OSS) is a type of computer software in which source code is released under a **license** in which the copyright holder grants users the rights to study, change, and distribute the software to anyone and for any purpose. Open source software may be developed in a collaborative public manner.

License is a legal document that records and formally expresses a set of legally enforceable acts, processes, or contractual duties, obligations, or rights related to the software and/or source code.

Open Source Licensing: Three Types

Business (e.g. BSL)

Very restrictive software licenses that may require payments to a developing company for use.

Copyleft (e.g. GPL)

Modifying the software requires you to contribute back any “derivative works”.

Permissive (e.g. Apache)

You can use and modify the software in essentially any way that you like.

Business Source License (BSL)

Technically a source-available license, **not an open source license**. The source code is **public but use is limited to certain users**.

After a certain amount of time the **code converts to a standard open source license**.

Viewed as a **compromise between proprietary software and open source**. Treat software under this license as **proprietary**, but with the extra added benefit that you (and others) can review the source code.

Notable examples: BUSL
Notable Projects: MariaDB MaxScale, Hashicorp Terraform, **Arbitrum Nitro**

Copyleft License

Open source licenses that **require users to make available all derivative works**. In other words, if you modify the code and use it in something, you typically have to make the source of that "something" available for free.

Very hard to use commercially–often you must release to the public code that you wish to keep private!

Notable Examples: GPL, MPL, LGPL
Notable Projects: Linux Kernel, Mozilla Firefox, **Geth**

"GNU is not in the public domain. Everyone will be permitted to modify and redistribute GNU, but no distributor will be allowed to restrict its further redistribution. That is to say, proprietary modifications will not be allowed. I want to make sure that all versions of GNU remain free."

–Richard Stallman, GNU Manifesto

Copyleft goal: increase contributions by legally requiring them.

Unfortunate reality: people just won't use your code if there are any viable alternatives due to the potentially cumbersome legal requirements.

It is VERY hard to relicense to a permissive license.

"GNU is not in the public domain. Everyone will be permitted to modify and redistribute GNU, but no distributor will be allowed to restrict its further redistribution. That is to say, proprietary modifications will not be allowed. I want to make sure that all versions of GNU remain free."

–Richard Stallman, GNU Manifesto

Permissive License

An open source license that lets you **modify and use the code freely**. Essentially **“do whatever you want”** with few extra restrictions.

Most commercially used open source code and most Linux Foundation codebases are licensed with a permissive license. It’s **by far the easiest kind of code to use**.

We recommend using code with a permissive license **if you want others to use your code** because it is by far the easiest to use without any legal repercussions.

Notable examples: Apache 2.0, MIT License
Notable projects: Chromium, Kubernetes, most LF projects, **Besu**

Common Licensing Pitfalls

markanddraw

Be Careful with Code You Use (LGPL Dependency Traps)!

When building a "combined work" with LGPL code in addition to other code: Section 4.d requires you to either LGPL license a "Minimal Corresponding Source" version of your code under LGPL, or

> "Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version."

This requirement generally implies that you can't have compile-time dependencies for LGPL code.

Be Careful with Licenses that Don't Have IP Protections!

"[Company XYZ] open-source products uses the BSD-3-clear license, which is a common, highly permissive open source license. However, this license does not explicitly cover patents, which is why we added an exemption that gives you the right to use our code (including the patents contained in it) for free for research, evaluation and prototyping purposes, as well as for your personal projects. The exemption does not grant any rights for commercial purposes. Therefore, if you want to use [Product] in a commercial product, you will need to purchase a separate commercial license by emailing us at [Company XYZ email]."

This company is actually very transparent! If you use code from less ethical and transparent people, you may be rolling the dice on a lawsuit!

Pay close attention to the licenses and licensing requirements of the open source projects that you use and to which you potentially contribute!

Recommendation: **use Apache 2.0**, which is a permissive license and has an explicit patent grants, for open source projects. If you want to make a proprietary codebase visible to others, a BSL license may be suitable.

Other Legal/IP and Copyright Protections

Problem: What if Contributors Add IP-Protected Code?

Contributors to open source software may unintentionally or maliciously **add code that has some form of IP protection** (like patents) to your open source project.

SCO–Linux disputes

From Wikipedia, the free encyclopedia

In a series of legal disputes between SCO Group and Linux vendors and users, SCO alleged that its license agreements with IBM meant that source code IBM wrote and donated to be incorporated into Linux was added in violation of SCO's contractual rights. Members of the Linux community disagreed with SCO's claims; IBM, Novell, and Red Hat filed claims against SCO.

On August 10, 2007, a federal district court judge in *SCO v. Novell* ruled on summary judgment that Novell, not the SCO Group, was the rightful owner of the copyrights covering the Unix operating system. The court also ruled that "SCO is obligated to recognize Novell's waiver of SCO's claims against IBM and Sequent". After the ruling, Novell announced they had no interest in suing people over Unix and stated "We don't believe there is Unix in Linux".[1][2][3][4] The final district court ruling, on November 20, 2008, affirmed the summary judgment, and added interest payments and a constructive trust.[5]

SCO–Linux disputes

Overview

Timeline · SCO–SGI code dispute of 2003 · SCOsource

Litigation

SCO v. IBM · *SCO v. AutoZone* · *SCO v. DaimlerChrysler* · *SCO v. Novell* · *Red Hat v. SCO*

Companies involved

SCO Group · IBM · Novell

Individuals involved

Ralph Yarro III · Pamela Jones · Darl McBride

Other

Project Monterey · United Linux · *USL v. BSDi* · Groklaw · Xinuos

V · T · E

On August 24, 2009, the U.S. Court of Appeals for the Tenth Circuit partially reversed the district court judgment. The appeals court remanded back to trial on the issues of copyright ownership and Novell's contractual waiver rights. The court upheld the $2,547,817 award granted to Novell for the 2003 Sun agreement.[6]

On March 30, 2010, following a jury trial, Novell, and not The SCO Group, was unanimously found to be the owner of the UNIX and UnixWare copyrights.[7] The SCO Group, through bankruptcy trustee Edward Cahn, decided to continue the lawsuit against IBM for causing a decline in SCO revenues.[8]

On March 1, 2016, SCO's lawsuit against IBM was dismissed with prejudice; SCO filed an appeal later that month.[9]

Protect Yourself from IP Trolls!

We recommend that your project either use the **Developer Certificate of Origin (DCO)** or a **Contributor License Agreement (CLA)**.

It is important to have appropriate legal frameworks around your codebase so that users of the code cannot get frivolously sued! **This is especially important if you allow contributions from people outside of a single company**.

Developer Certificate of Origin
Version 1.1

1 Letterman Drive
Suite D4700
San Francisco, CA, 94129

Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I have the right to submit it under the open source license indicated in the file; or

(b) The contribution is based upon previous work that, to the best of my knowledge, is covered under an appropriate open source license and I have the right under that license to submit that work with modifications, whether created in whole or in part by me, under the same open source license (unless I am permitted to submit under a different license), as indicated in the file; or

(c) The contribution was provided directly to me by some other person who certified (a), (b) or (c) and I have not modified it.

(d) I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project or the open source license(s) involved.

Developer Certificate of Origin
Version 1.1

1 Letterman Drive
Suite D4700
San Francisco, CA, 94129

Developer's Cer

By making a con

(a) The contribut ource license indicated in the

(b) The contribut propriate open source license and I have the right under that license to submit that work with modifications, whether created in whole or in part by me, under the same open source license (unless I am permitted to submit under a different license), as indicated in the file; or

(c) The contribution was provided directly to me by some other person who certified (a), (b) or (c) and I have not modified it.

(d) I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project or the open source license(s) involved.

That's it!

DCO is very easy to set up on Github using commits signed with the "-s" flag.

Be sure to have legal protections for your code! If you don't, **people will (justifiably) be less likely to use your code!**

We require this for all LF projects, and should should too for your OSS!

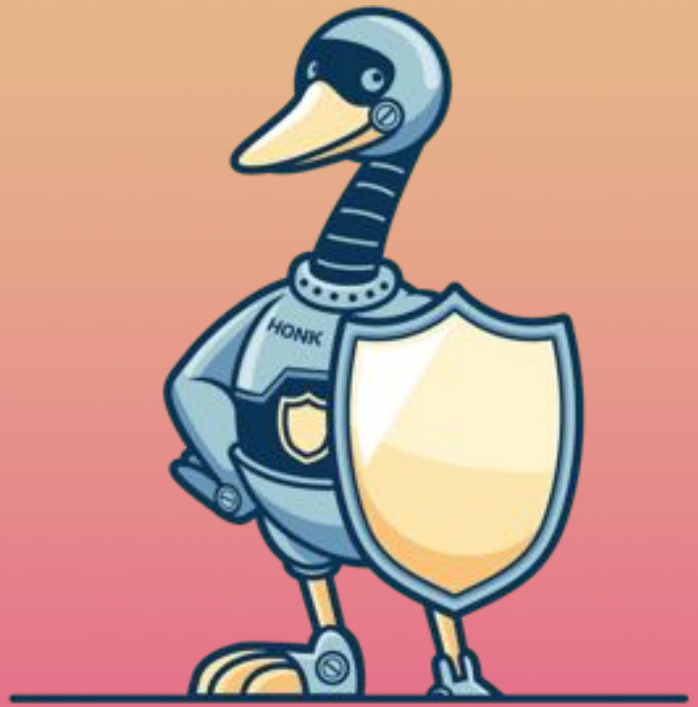
We like to use the **DCO at the Linux Foundation** because we find it has less contributor friction than a CLA, but a CLA (e.g. the Apache CLA) is a perfectly adequate solution too!

Handling Security

Placing a priority on security is essential for any open source project!

Some things that are different for open source:

- **Vulnerability disclosures and a security reporting pipeline** are very important and different from closed-source software.
- A **software bill of materials (SBOM)** is essential for making sure your software is secure when building with and in open source.
- **Authenticating your software and artifacts** is important for users.



OpenSSF

OPEN SOURCE SECURITY FOUNDATION

The Open Source Security Foundation (OpenSSF) is a community of software developers, security engineers, and more under the Linux Foundation who are working together to secure open source software for the greater public good.

The OpenSSF has answers and guidelines for all of the things you need to be doing for basic open source software security. Check it out for more details!

Security Vulnerability Disclosures and Pipelines

An advantage of open source software is that community members can find and report bugs. **You want to make this as easy as possible for them**.
Contributor friction for bug reporting may cause you to miss a big bug!

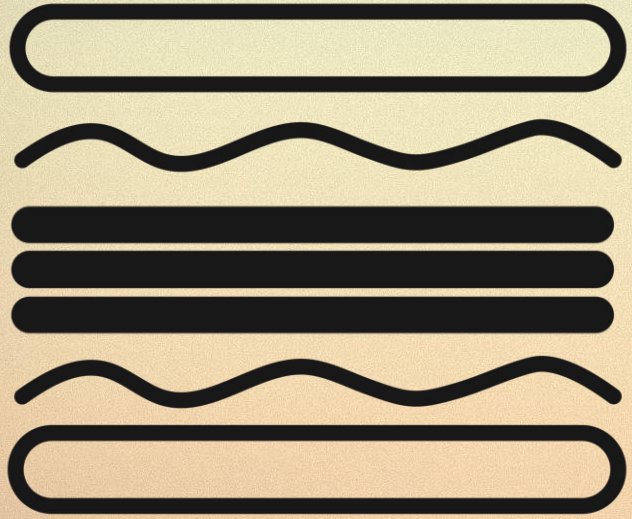
Form a security response team that can quickly respond to and handle bugs.

It doesn't matter too much the exact method you prefer for reporting, as long as it is **easy to find and follow**.

Recently in the Linux Foundation, many of our projects have been using the **Github tooling for security bugs**.

On the other hand, Besu has 7 reporting channels!

Recall: Building a Modern Platform or Application



Use Open Source Libraries to Solve Problems

Open Source Code (~70%)

Write Custom Code

Custom Code (~10%)

Choose a Framework

Open Source Code (~20%)

Software Supply Chain: A Big Attack Vector!

Software Bill of Materials (SBOM)

Keep track of all of the code you pulled in to your OSS project. If there are bugs you need to fix them and update quickly!

Carefully examine the code you do use to make sure it's **well-maintained**.

Well-run open source projects are **careful with the dependencies they use**!

There are a lot of good tools to help with this!

Software and Artifact Authentication

If you build a popular open source project, **people will attempt to impersonate you**!

This **constantly happens for Linux Foundation projects** (e.g. people attempt to create fake npm packages that look like those of official LF projects).

There are **plenty of tools** that you can use to sign and authenticate code; **use them!**

Building an Open Community

What is Open Source Software?

Open source software (OSS) is a type of computer software in which source code is released under a **license** in which the copyright holder grants users the rights to study, change, and distribute the software to anyone and for any purpose. Open source software may be developed in a collaborative public manner.

What is Open Source Software?

Open source software (OSS) is a type of computer software in which source code is released under a **license** in which the copyright holder grants users the rights to study, change, and distribute the software to anyone and for any purpose. Open source software may be developed in a collaborative public manner.

Collaborative Public Manner is important here because the freedoms provided by the license allow for **large and diverse communities to form** around popular open source software, helping drive innovation and **allowing companies to both contribute to and reap the benefits of these software projects**. Linux is a key example of one such community, but there are many others like the Apache Web Server, Kubernetes and OpenStack.

What Do We Mean by ...?

Open Source

A public repository of code that anyone can use, evaluate, contribute to and build upon for any purpose.

Open Development

There is a community actively building open source code in the open.

Open Governance

Procedures and roles for the community, how decisions and priorities are made for the project, and the roadmap are openly defined and managed.

Public Demonstration

The code is open source, but it is just simply publishing the code publicly. It is updated arbitrarily.

There is not necessarily a roadmap, and the main purpose of open-sourcing is to allow others to view, verify, and potentially use code that has been developed.

Open Product

A company or entity open sources a piece of software that constitutes a product.

It doesn't allow outside contributions, but keeps a roadmap, regular releases, and follows best security practices.

Benevolent Dictator

An organization or individual open sources their code and allows contributions from external contributors. Ultimate authority on merging code is always the "benevolent" org.

Contributors may be anyone, but maintainers are only from the "benevolent" organization and there's no clear path to influence.

Openly Governed

Code with transparent documentation and decentralized meritocratic governance Hosted under a third party non-profit entity which stewards IP, trademark, etc.

Anyone can join and contribute - inclusive clear, public process for leadership with no organizational or payment restrictions for maintainership or other roles.

Public Demonstration

Benefits

- Third parties can audit your code and see what you are doing. You can show off!
- There are well-known security advantages to being open source
- Others can benefit from your code or architectural decisions made in your code.

Drawbacks

- If you are doing a bad job, people will know–reputational risk.
    - But if you're trying to build a production system, you shouldn't be embarrassed to let others see your code.
- Outsiders cannot contribute.
- People are unlikely to use your code since they will effectively have to maintain it by themselves.

Open Product

Benefits

- All of the benefits of “public demonstration”
- People are more likely to use your code since they can see it is well supported.
- You maintain complete control over your project

Drawbacks

- Outsiders still cannot contribute.
- Because outsiders cannot affect change through contribution, they may be less inclined to use your software.
- This model requires a **high level of trust in the company**: what if the company changes direction and abandons the software, or goes away?

Benevolent Dictator

Benefits

- All of the non-control benefits of an open product
- Outsiders can contribute, which can mean less work for you
- Makes more folks likely to use your code, since they can fix/deal with their own pain points
- You ultimately retain control over the direction of the project.

Drawbacks

- Because you still retain control, competitors in the space or **those who don't trust your company/entity are unlikely to participate.**
- Still harder to get contributions and use than a fully openly governed project.

Openly Governed

Benefits

- All of the non-control benefits of a benevolent dictator project
- Companies/entities are strongly incentivized to contribute, as they can also play a role in governance proportional to their contributions
- A diverse group of contributors means that potential users can easily be convinced of the long-term stability and viability of the project, increasing overall use.

Drawbacks

- It's a community project now–not just "your" project.
- Other groups that contribute should receive governance powers proportional to their contributions, which may erode your control of the project.

Carefully choose projects to use or contribute (to) based on which style of governance is utilized by the project community! Whatever governance you choose, **document it clearly**!

If you want to grow an open community, we **recommend openly governed projects** but sometimes circumstances may make this impossible for your applications. Choose Wisely!

Best Practices for Growing a Community

Open Governance Do-ocracy
The most successful projects that have stood the test of time have neutral, open governance models where those who do the work make the decisions

Commercial Support Ecosystem
Encouraging commercial engagement in the project leads to jobs, faster adoption, new contributions and features that address new use cases.

A Neutral Home
It is difficult to attract investment from potential competitors if one entity owns or controls the codebase.

Transparent, Documented Governance
Companies and individuals looking to contribute want to know exactly how they can participate and don't want to be surprised.

“This seems hard.
What should I do?”

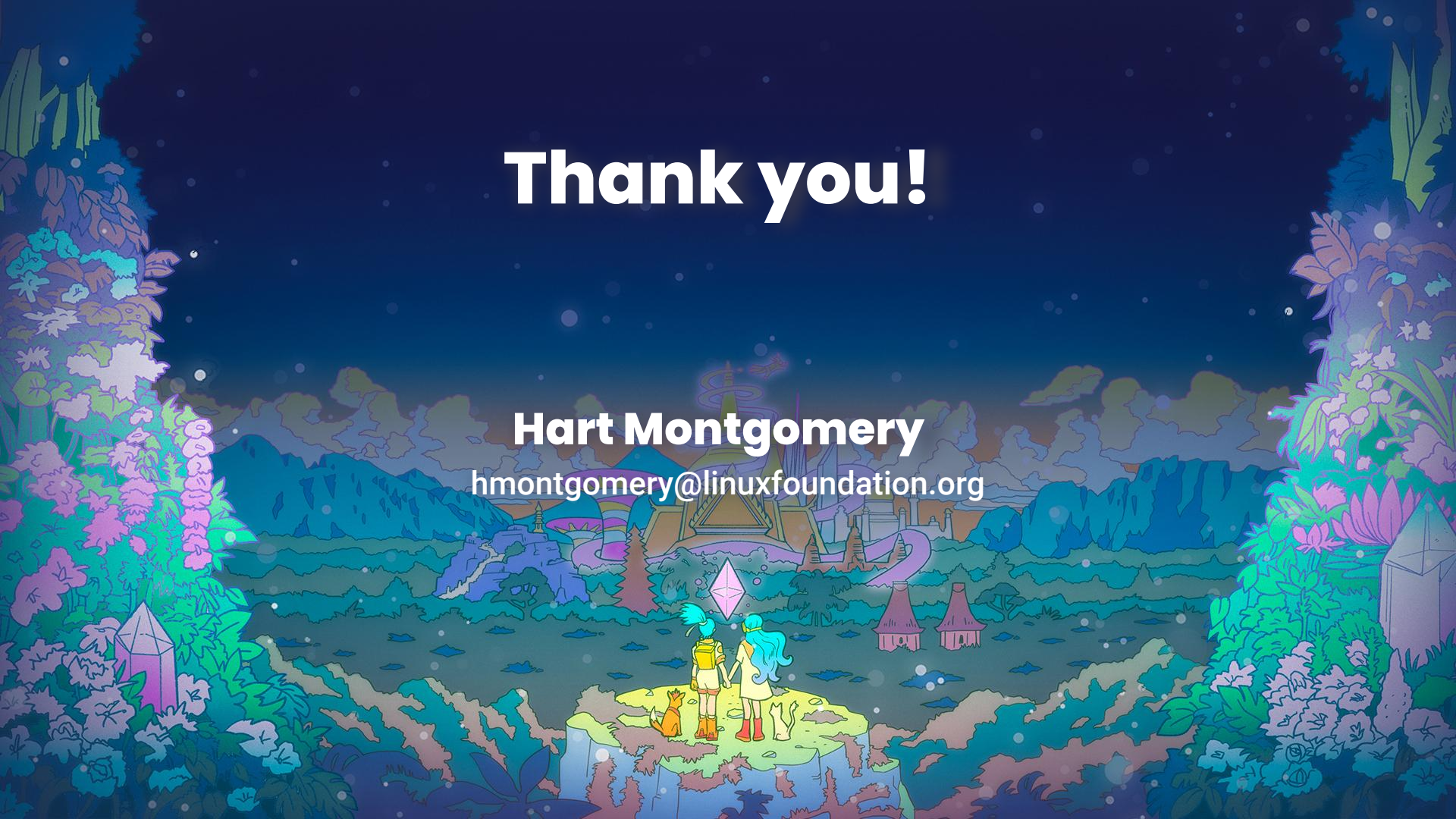
Use the LF (or another OSS foundation) as a home!

The LF serves as the **neutral organization to host the code** - eases worry that one organization will dominate or can terminate the project. We are trusted by most companies in technology.

It's hard to put together an open source community from scratch (legal & operational)! We have staff that are **experienced in managing and building communities**.

We have many shared resources across the Linux Foundation. **The same legal structure that protects the kernel and kubernetes can protect your project.**

Contributing code or a project is **completely free and open to anyone** (but it does require technical approval). Come find us at LF Decentralized Trust if you have questions!


Thank you!
Hart Montgomery
hmontgomery@linuxfoundation.org