Guide to open sourcing proprietary code

Ibrahim Haddad, Ph.D.
VP Strategic Programs, Linux Foundation
Executive Director, LF AI Foundation

This deck was contributed by Ibrahim Haddad to LF Energy and is licensed under CC BY 4.0.


@IbrahimAtLinux


linkedin.com/in/ibrahimhaddad/


github.com/ibrahimhaddad


IbrahimAtLinux.com


Ibrahim@Linux.com

LFENERGY

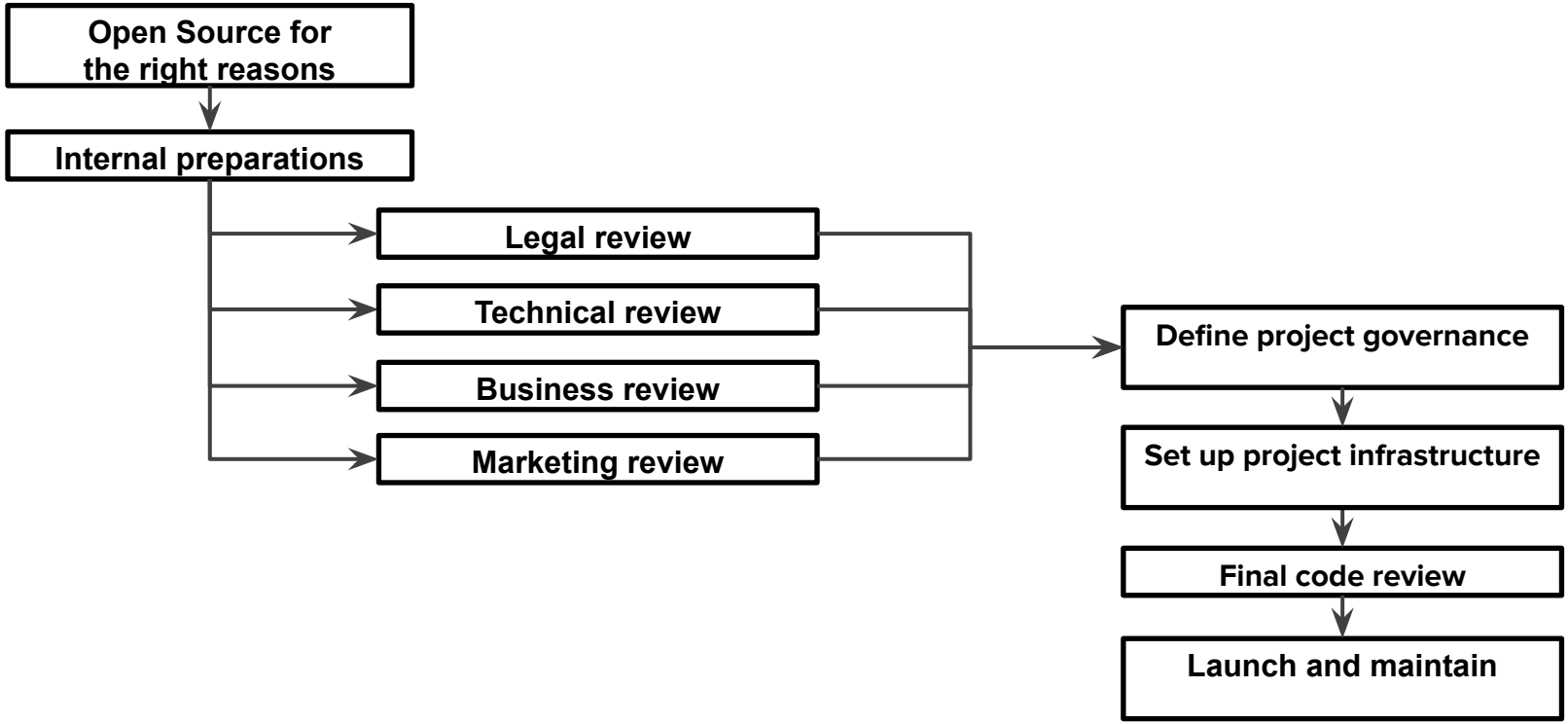
Basic assumptions

- Have completed competitive analysis
  - Arrived at conclusion that cannot adapt existing OSS project
  - Decide to proceed with open sourcing

- Employees are trained to work with open source development model

- Have funding and resource commitment to drive project as open source

High level overview of the process



Open source for the right reasons

Questions to ask yourself

- Is it possible to join an existing open source project?
- Can we launch and maintain a project using the open source model?
- Will other companies join the project from the start?
- Is there enough external interest to spontaneously form a community?
- What constitutes success? How do we measure it?
- Are we willing to finish what we start?

Why start a new open source project?

- More effort to participate in existing solution than start your own
  - Functionality
  - Cost
  - Strategic implications, such as undercut competition or establish an ecosystem around your company

- Strategic reasons
  - Establish a new de-facto standard
  - Onramp for other products or services you sell
  - Solve an industry problem in a way that attracts other contributors
  - Build an ecosystem that complements your company's mission

Whole stack or partial stack?

- Determine what parts of your product are sources of strategic advantage, and what simply supports those parts

- Non-strategic parts are often great candidates
  - Chances are good that other companies feel the same way, and might help you maintain it

Examples of good reasons to create an open source project from your proprietary technology

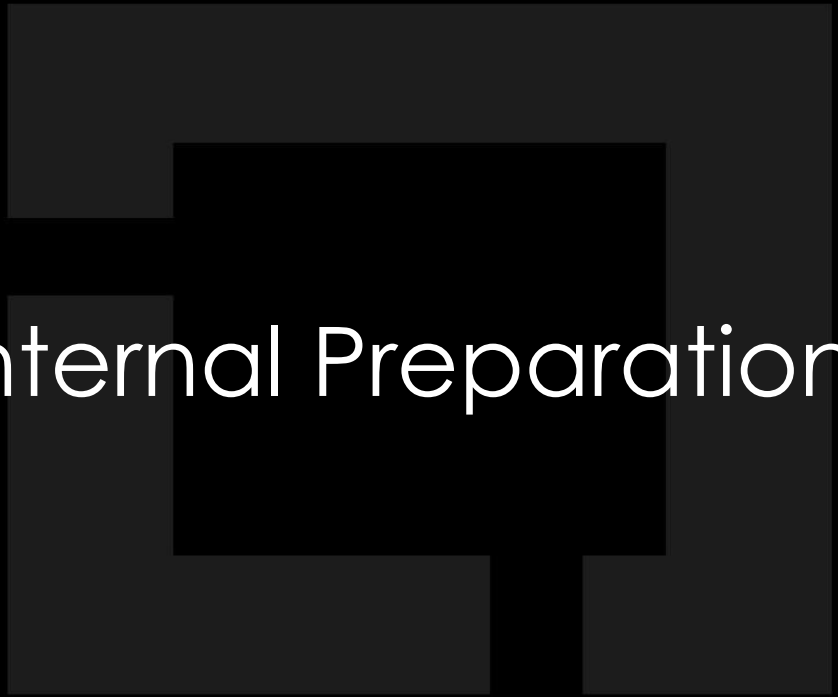
- Provide a reference implementation to a standard
  - Faster acceptance of a standard
- Ensure that critical software remains viable
  - Build a broader user and contributor base
- Ensure that new features are implemented in an existing project
  - Accelerate feature development
- Take control of your own destiny
  - Community-driven alternative to lock-in

Examples of good reasons to create an open source project from your proprietary technology

- Undercut the competition: Share development costs for non-strategic function
- Commoditize a Market: Reduce prices of non-strategic components
- Partner with others, engage customers: Strengthen relationships with common goal
- Drive demand by building an ecosystem: Grow symbiotic ecosystem for your product
- Offer customers ability to self-support: Ability to adapt your code without waiting on you

Examples of bad reasons

- Getting rid of obsolete software
- Leverage free engineering from the open source community when you aren't willing to invest
- As a positive marketing spin on an end-of-life announcement



Internal Preparations

Identify an executive project sponsor

- **Technical and Community**
  - Continually reaffirm commitment to the project
  - Foster a culture of merit-based recognition
  - Ensure neutrality
  - Follow open source methods and processes
- **Financial**
  - Internal: Fund enough development to get project going
    - Must have executive sponsor
    - To get others to believe in the project, you must show you do too
  - External: Commit to funding IT resources and sysadmins
    - Required as long as the project is active
    - May be shared among other co-sponsors
    - Be cautious who has access to the infrastructure

Train your employees

- Train your employees and managers
  - Open source development methods and processes
  - Working with the open source community
  - Your company's open source policies and compliance rules
  - Integrating open source software within your software development model
- Follow open source practices internally
  - Context switching between development practices is difficult
- Encourage community thinking
  - Welcome help in crafting a more general solution
  - Leave partisan feelings at the door, you will be working alongside competitors

Train your managers

- Traditional performance metrics may no longer apply
  - Only results count, not whose code is used
  - Influencing an outcome is just as valid a solution as writing code
- Manage your developers, not their maintainers
  - You can't control the open source process
  - Never start a dispute with someone who has a blog or Twitter account
- There is a learning curve for your employees and the community
  - Take the time to learn. There are no shortcuts
- Don't assume you know the answers
  - Whoever is closest to the community is probably the expert on what will work
- Building influence and respect takes time
  - Each new community member must build their own

Legal, technical, business and marketing reviews

Legal review

- Prior to open sourcing your proprietary technology, your counsel may perform the following legal due diligence
  - Patent scrub to identify all IP being made available
  - Decide upon license under which to release the code (adopt an OSI-approved license)
  - Write standard copyright notices to be made available in source code
  - Decide on whether you want to have a contributor agreement for new open source project (CLA vs DCO)
  - Decide on trademark use, requirements, and process
  - Run a source code scanning tool to ensure clean bill of materials
  - Etc.

Legal review: Considerations for submitted code

- A contributor agreement assigns copyright on contributed code
  - Ensure project can use all contributed code in perpetuity
  - Streamlines major decisions, like adding a new license to the code
- Some factors to consider:
  - Is there any situation in which your license might change?
  - Will participants accept a contributor agreement?
- Linux kernel equivalent is the Developer Certificate of Origin (DCO)

Technical review

- **Prepare the source code**
  - Document functions
  - Ensure coding style is consistent
  - Remove internal comments, references to other internal code, etc.
  - Remove any code not part of open sourcing plan
  - Remove critical dependencies on non-public components
- **Add license and copyright notices**
  - Add copyright notices and license information in source code files
  - Add license text as a file in the root directory

Technical review

- **Prepare the code for new external users**
  - Ensure it compiles and builds on target platforms
  - Provide documentation and use case examples
  - Fully document all APIs
  - Etc.

Business review

- Ensure there is a corporate champion for the project
- Ensure there is a commitment for resources
- Ensure there is a commitment for funding the open source project infrastructure

Marketing review

- Design project logo, color scheme, website, collateral, etc.
- Formalize branding guidelines
- Register social media accounts for the project (Twitter, Facebook, LinkedIn, etc.)
- Register domain names, including .org, .net, and .com

Project governance and processes

Establish a governance model

What is governance?

- The structure around a project that enables decisions on:
  - Participation guidelines and requirements
  - Architectural changes
  - Nominating maintainers
  - Final arbiter on disputes
  - Suspension of participants
  - Etc.
- Often similar to a board of directors
  - Typically represents mix of project contributors

Governance decisions

- Project should define and communicate, at minimum, processes for submitting code, requesting feature ideas, submitting bug reports, and release management
- A multi-company project may need a formal governance
  - Decide on how governance is structured, who can participate, for how long, etc.
- Example roles of a steering group
  - Secure funding for the project infrastructure
  - Steer the project to advance its market and technical success
  - Conflict resolution
  - Project level decisions
  - Sets the direction, tone, and vision of the project
  - Approve project roadmap prepared by a technical leadership group

Recommendations for formal governance

- Governing body should represent various participating entities
- Democratic system
- Clear decision-making process
- Clear path to resolve disputes
- Flexibility to adapt to changing project needs
- Clear means to add new or replace members

Project maintainer

- A formal position of leadership within the project
- Responsibilities include
  - Setting the criteria for accepted / rejected code
  - Reviewing submitted code / accept and reject based on predefined rules
  - Tracking dependency issues
  - Notifying developers of source code changes that may affect their packages
  - Managing source code security issues
  - Working closely with team developing the source code
  - Working closely with QA team testing the source code
  - Dealing with reported bugs in a timely manner
  - Preparing binaries – packages of the source code

Project infrastructure

Set up project infrastructure

Essential components

- Wiki
  - User contributed documentation
  - How-tos
  - Works-for-me platform testing results
  - Team collaboration
- Source code repository system
- Mailing lists
- IRC/Slack/Glitter/other
- Bug and feature tracking

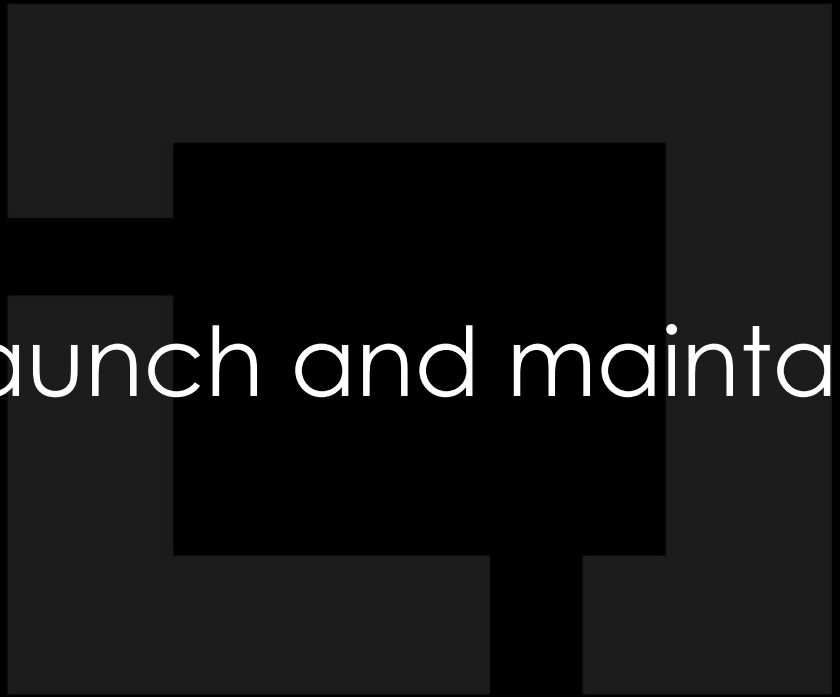
Nice to have

- Web site
- Milestone and release tracking
- Forums
- Branding
  - Logo, style guide, graphics, official colors or fonts
- Automated build and test system

Final code review

The final code review

- Ensures all requirements identified by the legal, technical, business, and marketing reviews are completely met
- Examples:
  - License, attribution, and copyright texts are all complete and in place
  - Source code scanner reports clean bill of materials
  - Comments are sanitized of casual or unrelated language
  - Source code does not inadvertently reveal internal projects
  - Source code is sufficiently complete that it builds
  - Source code builds using publicly available tools
  - File and function names reflect the project's name, if it has changed
  - MAINTAINERS file is up to date, if used
  - Etc.



Launch and maintain

Prior to launch

- Build critical mass before launching
  - Provide preview to customers and partners so they can begin to work with the code
  - Lobby for launch-day participants among your existing business partners
- Train your employees to follow open source methods and processes
- Ensure that all project infrastructure is running, secure, and scalable
- Ensure internal developers join and continually monitor IRC/Slack channels, mailing lists, etc.
- Upload the code

Be a good open source citizen

- Have conversations and make decisions in the open
  - Builds goodwill, but also reduces overhead in documenting decisions
  - Streamlines onboarding process for new participants
  - Archives ensure continuity if participants change
- Listen to the community
  - They know what they are doing, particularly on integration and testing
  - Encourage generalized implementations that extend what you need, particularly if someone else volunteers
- Initially allocate resources as though you will be the only company doing the work
  - Set goals, and make sure they have resources to get accomplished
  - This builds momentum until the leveraged development model takes effect

After the launch

- Work on building a developer community
  - Is it easy to find and join as an outsider?
  - Does the community have the documentation they need, and a means to update it?
  - Is the process for accepting community code working?
- Follow open source development model & practices
- Continue supporting the project to grow its community of users and contributors

Executive Summary

1. Analysis

A. Determine valid reasons to release proprietary code as a new open source project
B. Evaluate alternatives
C. Analyze the ecosystem for existing open source projects
D. Explore potential partners

2. Legal

A. General legal review
B. Confirm ownership of all code planned for release under an open source license
C. IP review
D. Decide on open source license to adopt
E. Advise engineering on applying the license to the source code
F. Develop a contribution agreement to govern how the project will manage contributions form the community - CLA vs DCO

3. Secure Funding

A. Determine funding requirements for the project for the next 18-24 months
B. Ensure internal support to provide the funding curing the critical early phases of the project

4. Branding

A. Create project logo and visual assets
B. Create GitHub organization and repos
C. Register domains and set up redirects
D. Register external accounts (Twitter, Facebook, LinkedIn, etc.)
E. Create web site

5. Prepare the Source Code

A. Update source code to match project branding, if it will change
B. Update copyrights on all source code
C. Add a license notice to all source code in the header
D. Add SPDX license identifier to all source code files
E. Add a license file in the root directory of the component that provides the full license text
F. Prepare detailed instructions on resolving dependencies, compiling, installing, and running the code
G. Ensure there are no dependencies on proprietary build systems or code that will remain private

6. Recruit Partners

A. Approach the business partners who will benefit the most from the project for public support on launch day and to participate in the project
B. Secure commitments from key partners to encourage their employees to participate in the projects and have some basic commitments
C. Privately approach related open source projects to ensure they are prepared for the announcement, and communicate how the new project will work with them
D. Anticipate conflicts where existing projects misinterpret the launch as competition, and diffuse them
E. Ensure business partners have early access to project source code
F. Work with partners to establish a joint value proposition and reference stack for shared customers

7. Establish Project Governance and Processes

A. Processes
   a. Feature requests
   b. Release management
   c. Submitting and reviewing contributions
   d. Etc.

B. Governance
   a. Determine initial maintainers
   b. Provide information on how to become a committer, a maintainer
   c. Explore need to have a steering committee, a small group of people who oversee the overall project
   d. Etc.

8. Establish Project's Infrastructure

A. Code repository / GitHub org
B. Bug tracking
C. Communication channels (mailing lists, glitter, slack, irc, etc.)
D. Wiki
E. Hardware
F. Server for collaboration infrastructure
G. Build system
H. Etc.

9. Announce and Launch Project

A. Prepare the Announcement
   a. Pre-brief launch partners
   b. Ensure that all project infrastructure is running, secure, and scalable
   c. Subscribe key project personnel to project mailing lists
   d. Ensure internal developers join and continually monitor IRC or Slack

B. Press and analyst relations
   a. Establish launch strategy and timeline
   b. Draft press release, get sign off from all involved parties
   c. Identify spokesperson and media contact
   d. Create internal FAQ

C. Announce & Launch the Project
   a. Release source code
   b. Publish a road map, even if it is preliminary
   c. Follow the open source development model

Guide to open sourcing proprietary code

Ibrahim Haddad, Ph.D.
VP Strategic Programs, Linux Foundation
Executive Director, LF AI Foundation