» The Linux Foundation

Upstreaming: Strengthening Open Source Development

January 2012

By Ibrahim Haddad (Ph.D.) and Brian Warner, The Linux Foundation

A Publication By The Linux Foundation
http://www.linuxfoundation.org

Upstreaming is a term used to describe the process of contributing in-house source code modifications back to an open source project, with the goal of having them accepted and distributed in future project releases. This paper discusses the process of upstreaming, the benefits to all parties involved (companies, projects, open source ecosystem), and highlights some best practices to follow.

Introduction

Upstreaming is a fundamental aspect of the open source development process. The term is derived from a river analogy, where water-borne goods float downstream and benefit those who are there to receive them. In open source, this is often used to describe the process whereby an individual or a company modifies the source code of an open source project to fit a specific need, and then contributes their modifications back to the project.

Open source software is being increasingly integrated into commercial products, driven by technical merit, ability to accelerate development, or to improve time to market. As a result, many companies are also modifying open source software to meet custom requirements. Development organizations that have not yet fully embraced corporate open source participation may maintain separate, internal branches of these projects, because it can be an easy way to get started when consuming open source software.

While separate in-house development trees can work for one-off product development or proofs-of-concept, long term maintenance can quickly become costly. This is particularly true if open source software will be updated throughout the lifecycle of the product, as custom code will need to be adapted to changes in the mainline open source project. Because developers in open source projects are likely unaware of modifications being made to their released code behind closed doors, there is no guarantee that routine development will not break internal derivatives of the code.

To fully realize the benefits of consuming open source software, many companies are choosing to follow an "upstream first" philosophy. This means that internal modifications to open source software are submitted back to the open source project to be evaluated for acceptance into the main development tree.

This article describes a typical process for upstreaming internal code, discusses the benefits of upstreaming, and offers practical guidance and best practices. It is one of several articles published by The Linux Foundation on open source development. A full list of publications is available from http://www.linuxfoundation.org/publications.

The Upstream Process

Upstreaming is the process of contributing independently or internally developed source code back to the originating open source project, with the goal of having it integrated into the main open source project development tree. While each open source project has its own set of processes and acceptance criteria, following the generic process illustrated in Figure 1 can improve the likelihood that your contribution will evaluated and possibly accepted into an open source project's main development tree.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

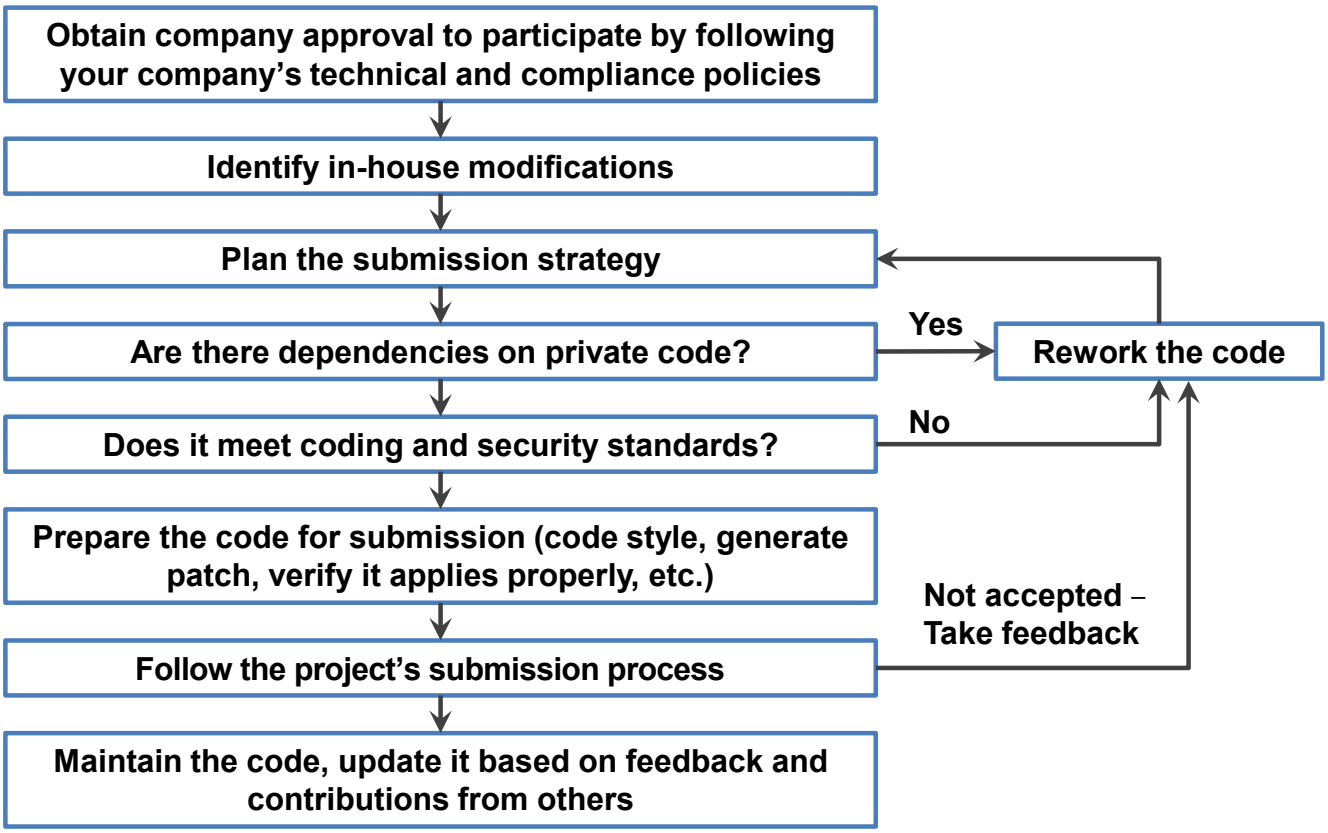

Figure 1: Typical process for identifying and upstreaming in-house code.

The process is begun within a development organization by obtaining internal approval to contribute modifications back to the project by following the company's open source policies and procedures. Next, identify the in-house modifications being made to open source software that can be submitted back to the project, and determine which ones are suitable candidates for upstreaming. When appropriate code has been identified, plan a submission strategy, particularly if the source code is a large submission, is complex, or has a major impact on the project. The goal is to avoid overwhelming the maintainer of the project and the project participants with a large submission that is difficult to understand, test and/or integrate.

Next, ensure there are no dependencies upon private, internal code. If there are, these must be resolved prior to submitting the code to the project. Also, be sure that the modifications are not inadvertently creating any security risks or breaking other code.

Prepare the code for submission, ensuring that it uses the project's coding and patch style, and verify that it applies cleanly against the project's source tree. Separate the source code into the smallest logical parts possible, as with any open source contribution, and then follow the project's submission process.

Submission typically begins with a discussion on the project mailing. Propose the contributions, ask for recommendations, respond to questions, and evaluate the likelihood of acceptance from the responses. If integration will be complex, communicate with the maintainer over the project mailing list and ask for advice on how to proceed. Once the code has been accepted, continue to maintain it within the project, and work with others to improve the functionality.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

It is important to remember that each project may differ in how code is submitted, tested, and accepted. If the process is not clear, ask for help and monitor the project mailing lists for other companies doing the same thing, and proceed accordingly.

Benefits of Upstreaming

Having code accepted upstream provides several benefits to the open source project, the companies submitting code to upstream projects, and to the open source ecosystem in general. Below is a list of advantages of upstreaming.

Less code to maintain in-house

Once in-house code becomes part of the main source code tree, the effort needed to maintain the remaining code can drop significantly, because the internal codebase is smaller. In addition, the code that has been upstreamed will evolve with the project, reducing the amount of effort required to maintain a parallel development tree.

More contributors

When code is accepted upstream, it becomes a visible part of the project. This enables other developers to contribute to it, submit new features, expand on existing functionality, and test it.

Increased code quality through peer review

Source code submitted to an open source project typically receives significant peer review as part of the normal submission process. This results in a feedback cycle leading to improvement of source code, and ultimately higher quality code that will be used in commercial products.

Faster integration and testing

Upstreaming simplifies and accelerates the process of integrating new or updated open source software. When maintaining a separate development tree (Figure 2), the integration cycle can be delayed by unanticipated integration, testing, and debugging required to maintain internal code. If changes in the upstream project have broken the internal code, it is the responsibility of internal developers to fix and verify any breakages before the product can ship. Because upstream code becomes visible to the rest of the project, the likelihood of these unforeseen breakages decreases.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

THE LINUX FOUNDATION

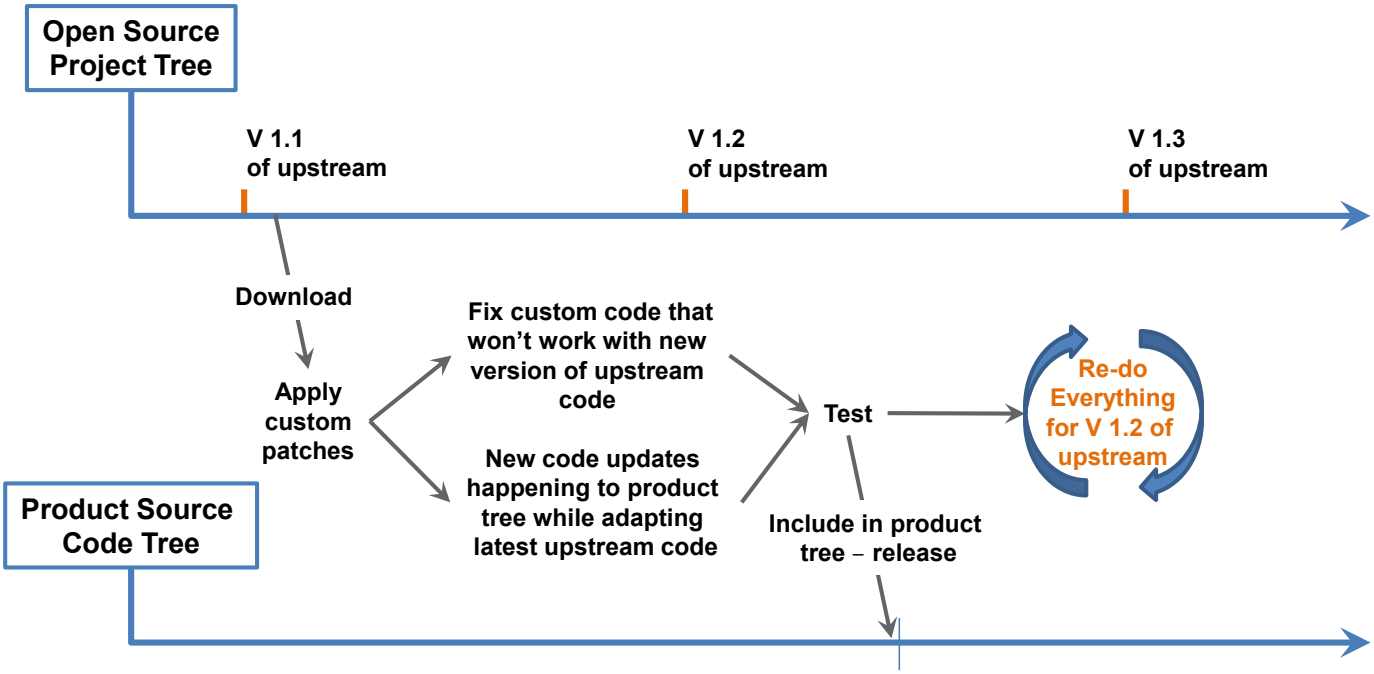

Figure 2: High level view of a typical development process: : In-house adaptation of private code can drive longer development cycles

Upstreaming can help avoid these breaks at the source, because upstream features are tested in the mainline project, and any issues should be found and fixed before a project release. In addition, a smaller amount of code will need to be maintained internally, shortening development, backporting, and testing time (Figure 3). In this way, reducing the amount of separately maintained custom code can substantially alleviate re-integration efforts.

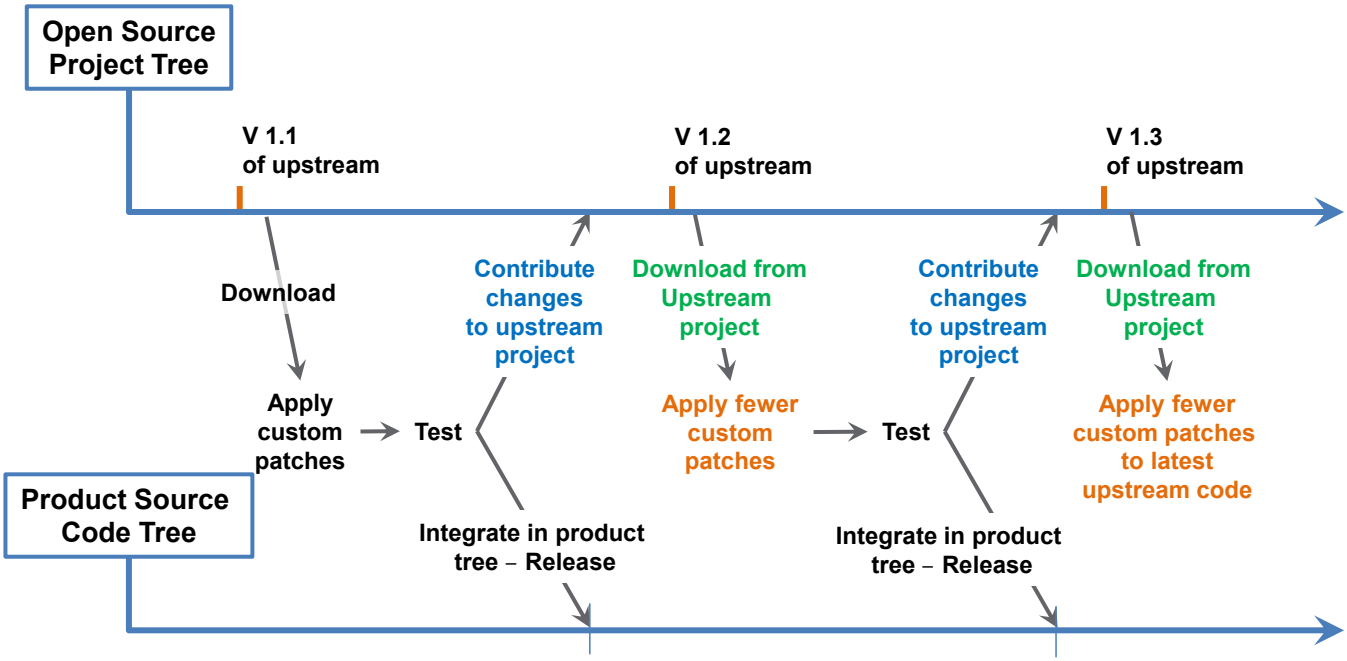

Figure 3: Similar development process with upstreaming to reduce in-house adaptation, resulting in shorter development cycles and faster time to market

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

Influence the project's direction

For companies relying upon open source software to build a product, upstreaming code is an effective way to provide technical leadership within a project, as it can guide the direction of the project and ensure it remains viable. Interaction with external project participants can also increase the likelihood that others will be aware of the company's needs, and they may be more likely to help implement new features and functions if those are of interest to them.

Reduce costs of achieving compliance

License compliance is a critical aspect of any open source consumption strategy. Upstreaming reduces the amount of internal source code that must be tracked independently from the parent open source project.

For more information on open source compliance, visit The Linux Foundation Open Compliance Program at http://www.linuxfoundation.org/programs/legal/compliance. The program offers a number of resources to help companies efficiently achieve compliance.

Decrease supply chain risks

Upstream contributions do more than reduce development and integration efforts and costs for the submitter. They can also be an effective means of ensuring stability in a company's open source supply chain.

Incorporating external software in a product requires a careful focus upon the stability of the software supply chain. Upstream contributions can offset risks from project direction or viability, because code contributions can provide a positive influence on the direction of the project. By providing guidance through source code submissions, a company can ensure that future versions of the open source software will continue to provide value to their development process.

Strengthening the project

Upstream contributions help provide stability to the open source project because they are a clear signal that the project is useful and important. Strong support from contributing companies tends to attract other participants, further increasing the durability of the project.

Upstreaming Best Practices

There are many ways to contribute changes upstream to an open source project. The following are recommended as starting points.

Design and implement code with upstreaming in mind

Not all modifications are suitable for contribution back to a project. For example, code with proprietary dependencies that can't be freely resolved, code with security or stability issues, or code that does not build using standard open source tools are all likely to be problematic when submitting.

Functional enhancements that improve the original source code to make it more stable, better performing, or more flexible tend to be better candidates. They can often be contributed without risking sources of competitive advantage, and are likely to have broader appeal beyond a company's own developers. This is ideal for the project, as it enhances the feature set, but also for the submitter, as it increases the likelihood that others will help maintain the contributed code.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

Ensure the contribution is useful to others

Project maintainers often look at the user-base for a contributed feature. Code that improves the functionality of the project for a broad base of users is generally favored over code with limited applicability.

Stay involved in upstream development

Maintaining upstream code is just like maintaining any other submission to an open source project. While others can contribute changes and fixes, they are not obligated to do so. Plan to keep staff engaged in open development, especially during the upstream transition. It can take time to build an external body of contributors.

Provide documentation

Document the code extensively, and provide use cases and proofs of concept, particularly if the new feature is complex. If the goal is to recruit external developers to further offset in-house development costs, make it easy for them to understand how the feature is used.

Upstream for the right reasons

Upstreaming is not a code retirement strategy, unless the company is already engaged with an external community that is willing to maintain it. Unmaintained code is usually removed from a project when it becomes clear nobody is willing to fix problems.

Listen to feedback, and act upon it

When submitting in-house code upstream, it is important to be responsive to feedback and questions, as other developers and maintainers may not be familiar with the motivations for making these changes. Be prepared for and respond to questions, and explain the background of the feature.

Candid feedback and peer review are fundamental to the open source development model. If a maintainer requests changes to the submission to make it more compatible with the parent project, evaluate and implement the feedback. Following the lead of the project maintainer may require modifications to the submission, but will increase the likelihood that other project participants will help maintain the code in the future.

Follow proper coding style

Many projects have standard guidelines for how source code should be formatted and submitted. Some projects also have very specific guidance on how code should be structured. It is important to ensure that upstream contributions match these guidelines, because non-adherence can lead to a rejection from the project maintainer.

Follow the processes set by the project

Most open source projects have detailed guidelines for the patch submission and review process. These processes are generally taken very seriously, and submissions to the project should follow them closely. This topic is covered in greater detail in another Linux Foundation publication, Understanding Open Source Development.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

THE LINUX FOUNDATION

Conclusion

This paper discussed the concept of upstreaming, a fundamental aspect of the open source development process. Upstreaming is the process of submitting in-house modifications to open source software back to the project for inclusion in the main development tree. This process can substantially reduce costs by minimizing internally maintained code. By following best practices and working within the processes of the project, upstreaming can result in lower maintenance costs, faster feature development, and better products.

This is the third paper in a series from The Linux Foundation that aims to provide practical discussions and best practices around open source development. Earlier papers included Understanding Open Source Development and Establishing an Open Source Software Strategy. For more papers on open source development, please visit http://www.linuxfoundation.org/publications.

Linux Foundation Resources

Linux Training

The Linux Foundation offers two training courses to enable organizations effectively work with open source developers:

- **LF 205: How to Participate in the Linux Community:** Working with the kernel development community is not particularly hard, but it does require an understanding of how that community works. This course is intended to bring attendees up to speed quickly on how kernel development is done and how to be a part of the process with a minimum of pain and frustration.
- **LF 271: Practical Guide to Open Source Development:** This course prepares organizations to maximize their effectiveness and shorten the time to value when participating in open source development projects. This course builds upon years of best practices and extensive experience in commercial participation in open source projects to help organizations approach the open development model in a structured and methodical manner, maximizing the likelihood of success. The course provides extensive examples from the Linux kernel community, and includes specific best practices for working with upstream.

Linux Foundation Labs

If you have a collaborative software project you need hosted at a neutral party, the Linux Foundation may be able to help. The Linux Foundation assists companies and communities by hosting collaborative software projects. The Linux Foundation provides three main services to Lab projects:

- The technical, operational and legal infrastructure so that project leaders can focus on technological innovation.
- Guidance and consulting on open source best practices gleaned from the two decades of experience of Linux and the ability to collaborate and network with the large and growing Linux Foundation community.
- By providing these services to companies and developers, the Linux Foundation provides a much needed framework for advancing and accelerating technology that allows project hosts to focus on innovation.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

There are two main criteria that must be met in order for the Linux Foundation to host a lab project:

- Use of open source governance best practices including license and contribution agreement choices in keeping with the ideals of Linux
- Project must either use Linux or have the potential to enhance the Linux ecosystem

If you have a project that may fit this criteria, please contact us:

http://www.linuxfoundation.org/labs.

Open Compliance Program

The Linux Foundation's Open Compliance Program was established to boost adoption of Linux and other open source by making license compliance ever-easier to achieve, to increase awareness and understanding of open source compliance responsibilities, and to make available free resources that can help companies establish their compliance programs. The program offers comprehensive training, compliance educational materials (white papers, compliance blog, webinars), compliance tools, an online compliance community (FOSSBazaar), a best practices checklist, a rapid alert directory of company compliance officers, and SPDX™, a standard to help companies uniformly tag and report software used in their products.

Events

The Linux Foundation produces a number of technical events around the world that provide a venue to bring together developers to solve problems in a real-time environment.

Publications

The Linux Foundation produces a wide range of publications that are available for free download. These publications are divided into three categories: Open Source Compliance, Workgroups (such as Tizen, OpenMAMA, LSB, SPDX, FOSSology, etc.) and Community. The Linux Foundation publications are available from http://www.linuxfoundation.org/publications.

About the Authors

Ibrahim Haddad, Ph.D., is the Director of Technology and Alliances at The Linux Foundation and Contributing Editor for the Linux Journal.

Brian Warner is Operations Manager at the Linux Foundation.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

The Linux Foundation promotes, protects, and advances Linux by providing unified resources and services needed for open source to successfully compete with closed platforms.

To learn more about The Linux Foundation, or any of our other initiatives please visit us at http://www.linuxfoundation.org/.