THE LINUX FOUNDATION

Enterprise Open Source: A Practical Introduction

By The Linux Foundation

www.linuxfoundation.org

If your company is involved in software engineering, it is very likely you already use open source software in your products or services; if so, you must have an open source strategy to ensure you are making the best use of open source software while protecting yourself from potential risks and liabilities. There has been an influx of new companies and industries into the open source ecosystem over many years, and each comes with a unique set of obstacles to overcome while implementing an effective open source strategy. Unfortunately, many companies fall into the trap of reacting to the open source ecosystem rather than adopting a proactive strategy that fully embraces open source engineering.

This book provides a practical approach to establishing an open source strategy by outlining the actions your enterprise can take to accelerate its open source efforts. The information provided here is based on the experience of hundreds of companies spanning more than two decades of professional, enterprise open source. This ebook will be most beneficial to software engineering executives, development managers, compliance experts, and senior engineers involved in enterprise open source activities.

Table of Contents

Introduction

The availability of enterprise grade open source software is changing the way organizations develop and deliver products. The combination of a transparent development community and access to public source code enables organizations to think differently about how they procure, implement, test, deploy, and maintain software. This has the potential to offer a wealth of benefits, including reduced development costs, faster product development, higher code quality standards, and more.

To establish open source software as a major driving force for software development, your company needs to develop business-level objectives and fully identify any constraints faced for the use of open source software. The goal is to establish consensus and communicate business rationale behind new policies. This book will help you develop a strategy that transforms your efforts from a defensive approach that reacts to open source software to offensive market leadership that is fueled by strong open source engineering.

Why Open Source?

The first step in this process is asking yourself what you want to get out of an open source engineering effort, projects, or industry initiatives. There are quite a few ways to benefit from the use and development of open source software, and there is no one-size-fits-all plan. Your strategy will be custom fit for your organization, so it is important to consider all aspects of your company that could benefit from the development and execution of an open source strategy.

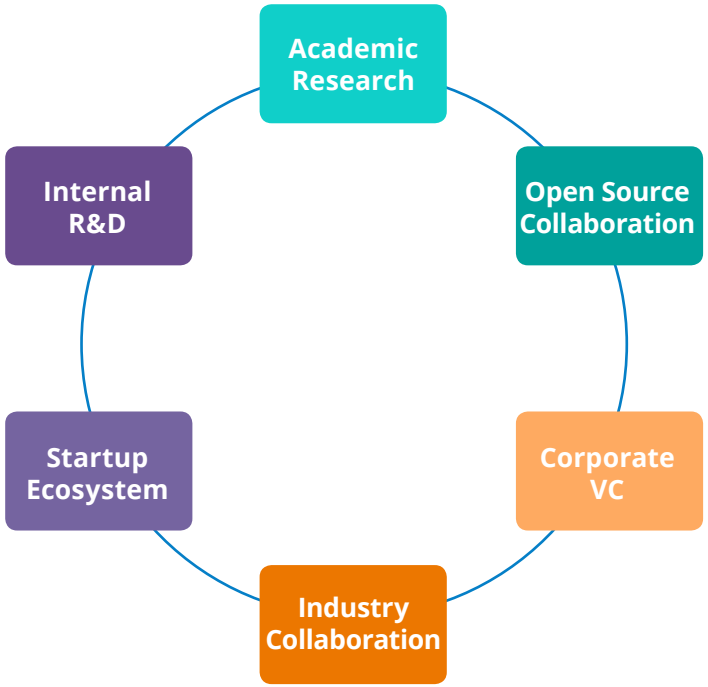

Figure 1: Open source is an important element among many other that help an enterprise accelerate its innovation

Another Tool in Your Tool Box

For starters, open source software allows shared development and lowers R&D cost by enabling you to reap the benefit of billions of dollars of open source software that can be harnessed to create better products and services. Open source software helps accelerate product development and enables faster time to market especially when products needs and requirements are aligned with upstream open source projects. Achieving such an alignment is a necessary aspect of an open source strategy.

Open source software development can also enable you to drive industry leadership by providing strong influence on the technologies used in products. This can help commoditize competing products and services as open source replaces critical components of innovation ecosystems. Additionally, participating in open source development gives you an edge in the talent war because organizations with strong open source R&D attract top software talent.

Software, Software, Software

Much like how "location, location, location" is the foundation for value in real estate, software has become the defining value factor in virtually every industry. To illustrate this concept, think about the various phones you might find at your local electronics store. They all come with amazing hardware, incredible displays, customizable memory and storage capacities, waterproof or water-resistant features, etc. The real difference is seen when you interact with them: it is software that makes them unique and this is the tipping point that shifts a consumer from buying one to another.

On a related note, looking at vertical software stacks across most industries, we find the penetration of open source to be astounding, ranging anywhere from 20 percent to 85 percent or more. No matter what industry you are in or what product or software you develop, you likely have a high reliance on open source software.

Adaptability to Various Business Models

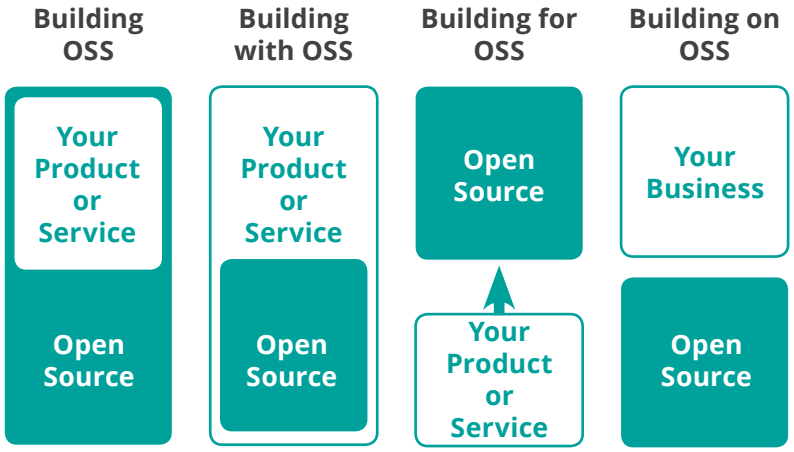

Figure 2: Various ways of using open source software to support your specific business model

If you put specific license requirements aside, open source software supports a variety of business models as illustrated in Figure 2. The figure shows the various ways you can use open source software.

These use cases are proven and have remained true for some time now, but their separation is mostly for illustration purposes. Hybrids of these models also exist, and these depend on a company's specific products or services strategy; it is common to find a company that uses multiple modules for different products or service offering.

Building open source

This is the most basic, yet most challenging model; it involves creating and commercializing open source software for direct return. With this strategy, your

company focuses primarily on building open source software and then provides value through expert services and products. Red Hat is perhaps the best example of a company that utilizes this model with an incredible amount of success.

Building with open source

This model gives you the ability to create proprietary software or services that work with or on top of open source. With this strategy, your company builds with open source software or relies on open source software to provide basic low-level libraries and components. This is a widely used model, and it is hard to find a software product that does not incorporate open source software.

Building for open source

Historically, this model has entailed creating software for providing it as open source and adding value-add services for revenue. Following this approach, a company builds a product or service with the goal to make it open source and build a business around it. Another example would be companies that create source code for the purpose of open sourcing it.

Building on open source

With this strategy, your company builds a product on top of open source software, where open source provides the foundation and you provide the higher parts of the stack where the actual commercial value resides. In this model, proprietary software or services have strong dependencies upon OSS and almost any new business today will heavily depend on this development model and the open source ecosystem.

Product Dependency

Organizations can rarely build a product without using open source. Virtually everything we build relies on open source in some way or another, and this also applies to enterprises that use software as part of their commercial offering. If you have valuable products that rely on open source, would you want to turn your back on billions of dollars worth of R&D? A planned involvement in strategic open source projects can prove immensely helpful for the company's bottom line in terms of leveraging external R&D and speeding up time to market.

More Focused, Faster Path to Innovation

Open source software is often used at lower levels of the software stack because these are the areas with the most in common between organizations. Better use of these low-level components allows you to focus your

own resources on differentiating at higher levels in the software stack and improve upon the unique value you provide to your consumers. This is a fundamental business advantage that open source software enables.

To examine this concept, consider the following: do people buy your products because of the specific software libraries you use in them? Probably not. Instead, they buy things based on the end user experience, which is more often defined by the features and improvements your product offers over your competitors. Freeing yourself from building low-level components frees up valuable resources to create value in the places customers care about the most.

Product Enablement

There are two ways open source software can enable better product development (Figure 3):

1. **Direct enablement:** wherever there is a direct link between internal activities related to open source development and and the products or services the company creates.
2. **Indirect enablement:** where the efforts spent on open source activities are not directly related to the products or services the company creates. This impact can still be tracked via specific actions that are taken as part of your open source activities.

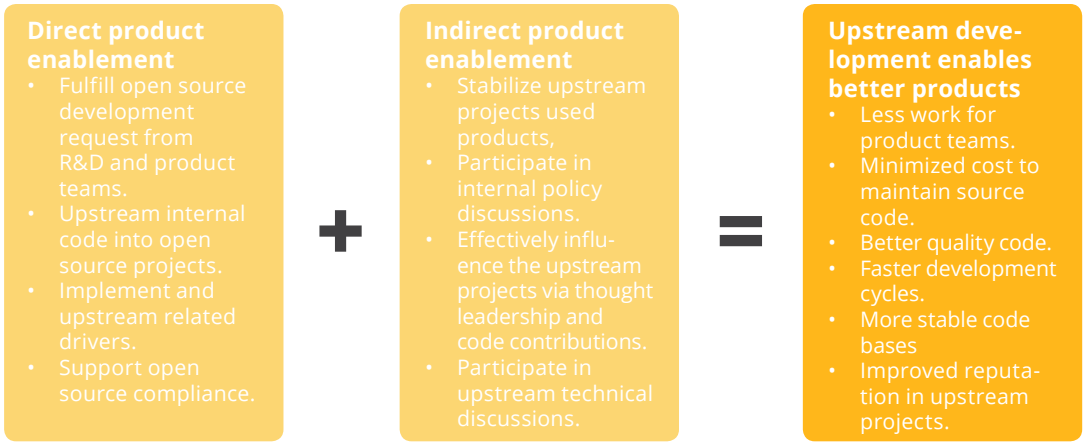

Figure 3: Open source product enablement

Direct product enablement

An open source program can directly impact an organization's open source code development by contributing code related to products and services. A few ways open source teams can directly improve product development include:

1. **Fulfill open source development requests from R&D and product teams.** Product teams typically have their own developers who contribute to open source projects but, they typically have less freedom because they are tied to product development. Often, open source developers embedded in a product team have a hard time striking a balance between upstream responsibilities (as a committer or maintainer) and their role in product development. Therefore, an open source group can receive requests, such as "we need feature X implemented in open source

project Y," and the engineering team will deliver the code for the product to the open source project.

2. **Upstream internal code into open source projects.** Contributing source code is the best way to gain influence in open source projects and build a positive reputation for the contributing company. One major goal of upstream development is to minimize the technical debt with respect to open source components used in products and services. In other words, companies look to minimize the delta between the open source branch and the internal branch. If upstream contributions do not happen, the product team will be stuck with large code bases that are uncoordinated with upstream, and they will spend much time back-porting updates to their out-of-sync fork instead of advancing the product.
3. **Compliance support.** Program managers can also provide assistance to resolve compliance issues and support the compliance team with the open source compliance inquiries they receive.

Indirect product enablement

An open source program's impact can go far beyond contributing code to various open source projects. From public relations and marketing, to legal support, developer training, and more, an open source program enables development in myriad ways:

1. **Stabilize upstream code used in products.** Open source offices with dedicated engineering resources can help stabilize the code of any open source projects a company relies on by finding, fixing, and testing bugs. This improves the code's overall value for all the project's users, including your company.
2. **Participate in internal policy discussions and decisions.** Open source engineering has a different set of requirements than traditional proprietary software development. Open source team representatives should be present at internal policy discussions to ensure these policies continue to enable open source development.
3. **Influence the upstream projects via thought leadership and code contributions.** The only way to influence an open source project is through direct participation and code contributions. If you want to provide leadership by influencing the direction of developing or maturing technology, you will need to have engineers that contribute the necessary code to do so.
4. **External technical discussions.** Open source engineers can influence communities through participation in technical discussions; they do this by being active on mailing lists and IRC channels to participate in discussions and stay informed on the latest project updates. Larger projects with formal governance structures sometimes also offer the ability to sit on technical steering committees for well-qualified individuals.
5. **Internal technical discussions.** Internally, open source developers can participate in policy and architecture discussions to ensure the organization's decisions match the direction of a specific project community. Open source developers should be present for any strategy discussions related to long-term planning for products that rely on open source code.

Open Source R&D Is an Innovation Enabler

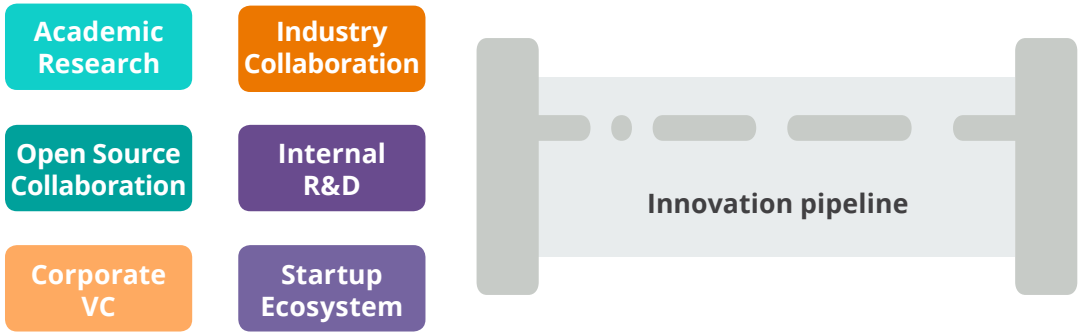

Figure 4: Open source is one of many innovation enablers

Open source R&D is one of many efforts that contribute to a company's overall innovation pipeline (Figure 4). It can be integrated with many practices including academic research, corporate VC, industry collaboration, internal R&D, and startup ecosystems to compound the effects each offers. Open source R&D works best when its impact is shared amongst multiple business initiatives.

Lessons Learned from Two Decades of Enterprise Open Source Experience

December of 2000 marked a major milestone in the history of open source when IBM pledged to spend $1 billion on Linux R&D. IBM was a true pioneer in the enterprise world, betting on Linux and open source when very few companies were doing so, especially not at that scale. IBM had to learn a lot about working with open source software and the various communities they were involved in. This was the starting point for open source adoption in the enterprise and other companies have since followed, first by the dozens, then by the hundreds.

Thousands of companies are still entering the open source ecosystem and wanting to be part of it today, as it becomes the new normal of software development. The question is: How can you minimize the enterprise learning curve and speed up the process of getting it right? In the following sections, we'll explore several lessons learned from nearly two decades of enterprise experience with open source software.

It's worth mentioning that in addition to implementing these practices, you'll need to lead a cultural shift from traditional software development practices to a more open and collaborative mindset. Internal company dynamics need to be favorable to open source efforts. As an open source leader inside your organization, you will face challenges in terms of funding resources, justifying ROI, getting upstream focus, and so forth. These challenges often require a major shift in mindset and a lot of education up the chain. We will explore some of those challenges in a later section.

Identify Reliance on Open Source Software

The first step toward improving your company's open source engagement is identifying where the company relies on open source software. The level of reliance and the level of importance of these open source components to the product portfolio, companies have several options to chose from. One option is to focus on software used by many business units. Another is to focus on software that poses more compliance risk than others (e.g., mobile apps and embedded hardware may pose more compliance issues than your datacenter code.) Such approaches will allow you to show a return on investment across multiple business units or across high risk areas and increases your chances for more funding and support.

Focus your contributions on upstream projects that directly benefit the company's strategy and products. It's easy to get carried away hopping between different interesting projects; in an enterprise setting where open source engineering is considered a cost center, you should focus on projects that support product development.

It's beneficial to do a yearly review of the company's product portfolio in an effort to be involved in open source projects that are commonly used across as many products as possible. This list can then be prioritized based on several factors to focus efforts on the top projects given limited resources. A methodology that drives your priorities is a great way and a forcing function to remain focused on what is important, justifiable, and fundable.

Identify Open Source Skills Portfolio and Hire from the Community

Once you have identified the most important ways in which your company relies on open source, you can begin to identify where you need to grow internal talent to match the required open source skill set. It takes considerable time to grow internal open source expertise, and hiring key developers is a critical step that allows your organization to quickly gain skills, recognition and mentorship capabilities.

Two or three people are a great start toward making a noticeable impact in a large project such as the Linux kernel, attracting further hires, and allowing sufficient resources to mentor existing junior developers.

The goal is to find people who have enough peer recognition to be influential in the community; there are typically four pillars to consider when hiring: technical domain expertise, open source methodology and experience, working practices and alignment between corporate interests and the candidates interests. It is very hard to motivate a senior open source developer when their personal interests and skillsets do not meet with corporate interests in a given project. A Linux memory management expert may not be interested in working on file systems, a corporate priority. Therefore, finding a match in interests is critical for a long-lasting relationship.

Develop Your Open Source Strategy

There are many questions to answer when determining your open source strategy, and they should be answered early in the process.

An open source strategy should address four key requirements (Figure 5):

- The open source projects it aims to address,
- The respective open source project community it aims to engage with,
- The internal enterprise open source governance, and
- The internal enterprise culture and whether or not it will be enabler of open source efforts.

Figure 5: The four poles of open source strategy

The following sections will cover the three primary questions you will need to go over before you create your open source strategy.

Although you can choose from several strategy objectives, some objectives are common for most companies that use and develop open source software:

- Reduce development costs.
- Improve the quality and flexibility of products.
- Achieve a faster time to market for products.
- Increase engineering capacity through community engagement.
- Broaden and deepen developer community commitment to your open source efforts.

Common benefits seen by customers of companies that use and contribute to open source software include:

- Lower cost products and applications.
- Higher quality and more reliable products and applications.
- New products, capabilities and applications delivered to the market sooner.

How can an open source strategy help you achieve overall corporate objectives?

The first question you answer will identify where open source software fits within your overall corporate objectives. Companies can benefit from open source use and contribution in many areas, and your strategy can be tailored to fit your specific needs. This determination will enable you to focus on the areas where open source can best benefit your company. The objectives you set can include any of the following:

- Set a long-term, high-quality roadmap to take leadership position within the product ecosystem.
- Reduce the cost and complexity of building and maintaining products or services.
- Build a differentiated position that targets higher profit margins.
- Commoditize competing products or services by building open source alternatives.
- Improve overall quality of products and services.
- Increase external visibility and brand recognition through public open source involvement.

How can it help you achieve your IP strategy?

Open source licensing is different from proprietary licensing and your company must account for how

differences in licenses affect your ability to benefit from the use and development of open source software.

The objectives you set can include any of the following:

- Determine a licensing strategy that best allows the company to benefit from external involvement while enabling better proprietary products.
- Mitigate intellectual property risk by ensuring compliance with open source software used in products and services.
- Enable greater differentiation in proprietary intellectual property by improving core open source components.

How can it help you grab opportunities that are otherwise unattainable?

Open source also offers some unique opportunities that are only be obtainable through an open source strategy. Common objectives that fall under this category include:

- Provide market leadership by focusing R&D investment to improve key open source technologies that are complementary to differentiated capabilities in products.
- Defend existing market positions by supporting key open source initiatives and consortia, selectively releasing proprietary capabilities as open source to disrupt competitors or competing markets, and use open source to level the technology playing field.
- Drive cost of goods sold improvements by incorporating readily available open source commoditized capabilities and market accelerators in products. The cost per product delivered will decline over time in as a result.

Identify Current and Target Position on the Open Source Strategy Ladder

There are four primary strategies to choose from when it comes to open source software: consumption, participation, contribution and leadership. Each strategy requires you to be successful at the previous strategy, and how far your company goes up this ladder is entirely up to you.

Figure 6 illustrates these four primary open source strategies/position and you can notice that they overlap as you transition from one position into another. Typically, the early stages, consumption and participants, are engineering driven when engineers start using various open source components based on their technical merits to speed up development and have limited participation in select projects, either to join the conversation or make small contributions. With time, this usage becomes known to higher levels in the company, and when it gains tractions, such involvement become business driven based on determined strategy.

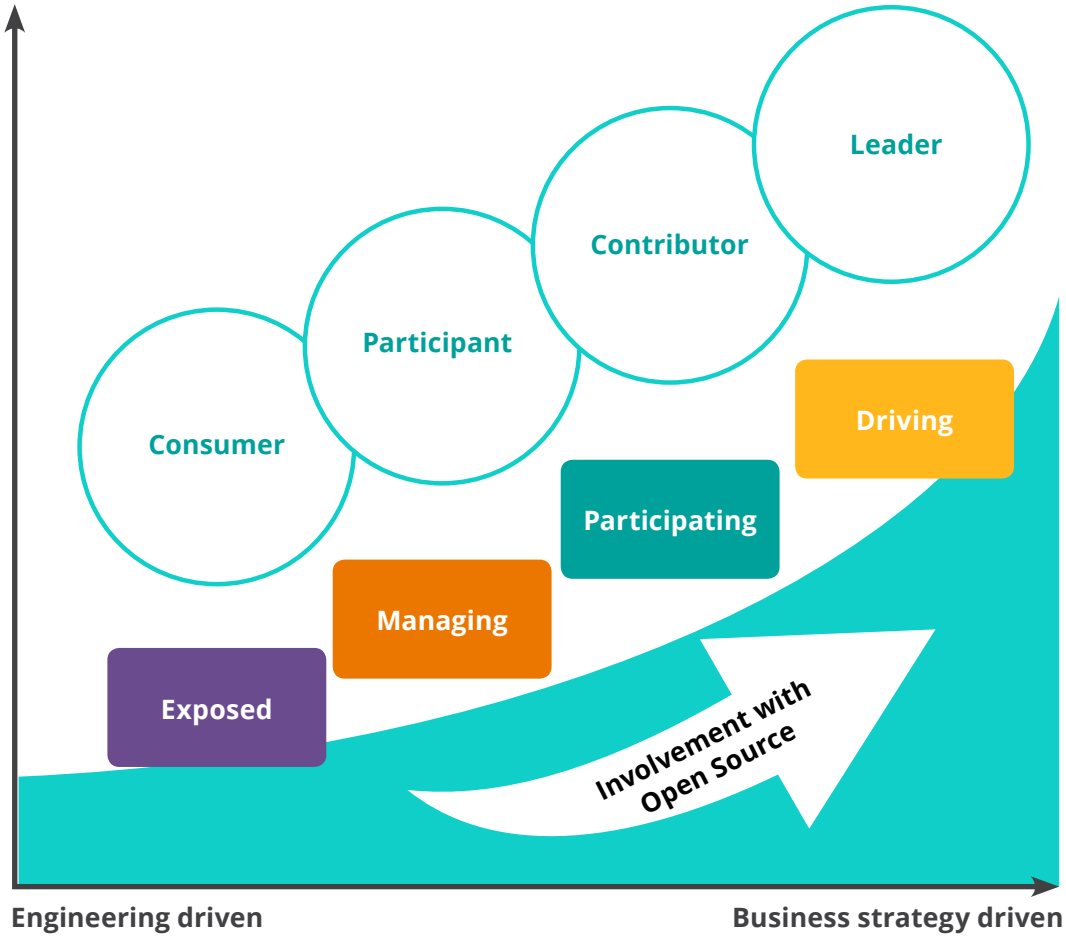

Figure 6: Stages of open source involvement

Some companies achieve their goals simply as consumers and are content to stay at that level, while other companies have different ambitions and want to push higher in the rank to attain a certain leadership position. There is a good chance you are already at one of these levels of the ladder, so it is important to identify both your current position in the ladder as well as your targeted position and chart a path to move from the current position to the target position.

Strategy: Open Source Consumption

The most common starting point for organizations is as an open source software user in their commercial products. Aggressively consuming open source components will increase your ability to differentiate and reduce overall time and cost to deliver commercial products. Here are the necessary components of the open source consumption strategy:

- Use a strategic classification scheme to guide decisions on what open source software to consume.
- Ensure the company meets all obligations of its use of open source software.
- Deploy automated workflow software for evaluating/approving open source usage.
- Establish an Open Source Review Board (OSRB) immediately to serve as a clearinghouse for all Open Source activities.
- Create incremental investment in headcount and infrastructure in engineering, product management, and legal to manage a complex mix of closed source / open source software.

Strategy: Open Source Participation

Once your company is successfully using open source software in products or services, you can expand your strategy to participate in the open source community. Unless you have already hired experienced developers from the community, you will first need to engage more closely with the community to increase your visibility and to begin attracting the talent you need. Here are the necessary components of the open source participation strategy:

- Monitor community communication platforms like chat servers, mailing lists, forums, and websites to stay informed about project developments.
- Attend relevant conferences and meetups to establish a relationship with the community.
- Sponsor project events and foundations to improve visibility within the community.
- Educate developers on how to participate in and contribute to open source projects

Strategy: Open Source Contribution

Once you are ready to build on your company's participation and begin contributing code to an open source project, you need to selectively engage with targeted projects and communities to drive you company's needs. Contributing to strategic open source projects can help your organization gain additional value as code contributions can help shape future features in the project that meet a company's needs

Here are the necessary components of the open source contribution strategy:

- Hire a staff director to lead open source strategy and manage the OSRB.
- Hire contributors and committers to key open source communities that are critical to your products.
- Deploy open source collaboration tools to support open source usage and contributions.
- Add open source developer resources.
- Incrementally invest in engineering, product management, and legal to engage with existing external communities.

Strategy: Open Source Leadership

The final step in the open source strategy ladder is leadership. This scenario builds on all of the prior scenarios to capitalize on emerging trends in technology to establish a leadership position. Leadership roles in existing open source communities are earned by establishing trust with the project members and by maintaining a high level of continuous contribution to the project.

This scenario requires significant investment in targeted open source communities and consortia to establish leadership agenda. It will also require incremental investment primarily in engineering, product management, and legal to establish leadership in external communities and industry consortia. Here are the necessary components of the open source leadership strategy:

- Increase engagement with targeted open source communities.
- Selectively engage with open standards to drive the company's needs.
- Engage with open source foundations.
- Establish an open source project, organization, or foundation.
- Incrementally invest in engineering, product management, and legal to engage with existing external communities.

Transitioning

As a tech company, you are evaluating, using, and deploying open source software already. You are likely participating and maybe even contributing to projects too. Ideally, your open source program is guiding these efforts, abating risks and leveraging your participation to benefit your strategy. Figure 7 illustrates the stages of participation with an organization. Whereas the earlier stages take place with or without guidance,

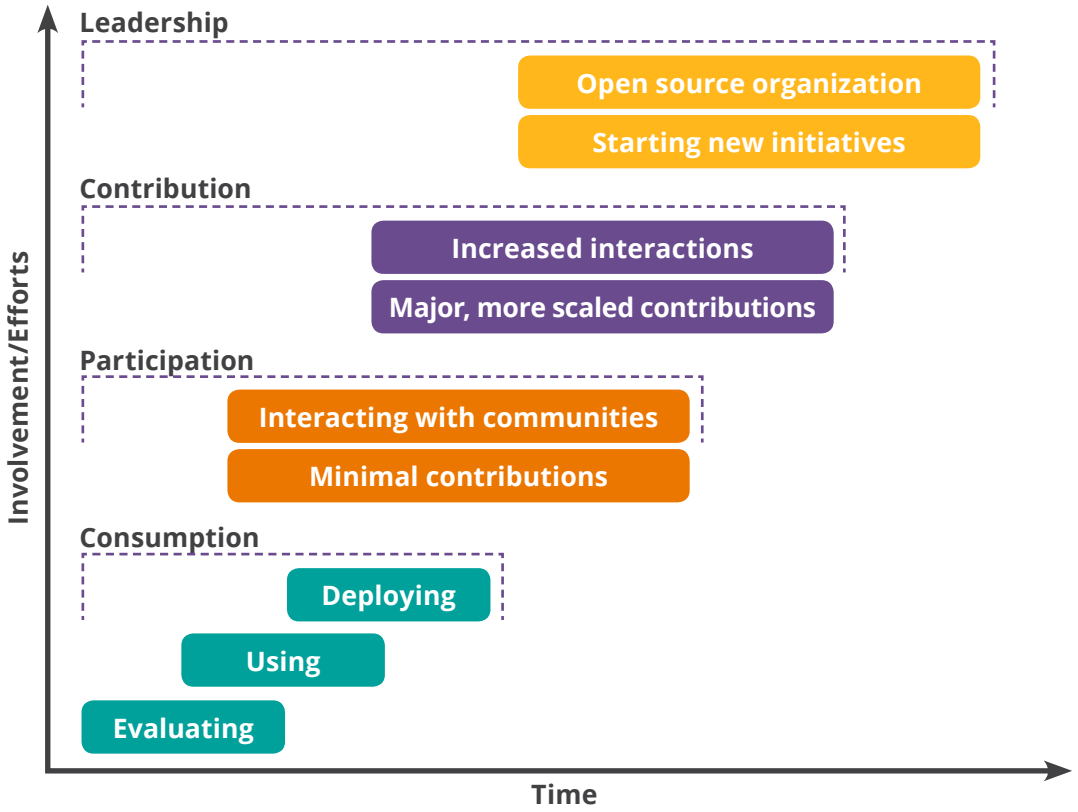

Figure 7: How to advance from one open source strategy to another by increasing efforts and activities.

achieving success at the leadership stage can only take place with an open source program in place.

Figure 7 illustrates the four key strategies and the major activities within each one. The purpose is to highlight the gradual proliferation of open source and the actions an enterprise can take to accelerate adoption and mastery of contributions.

Implement Open Source Infrastructure

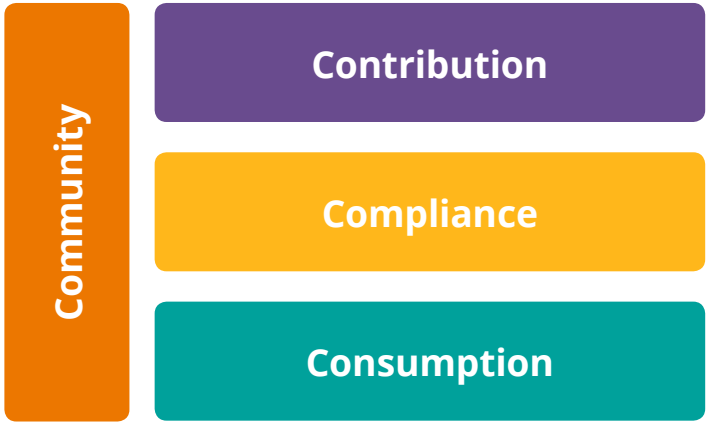

Figure 8: Core elements in an open source infrastructure

Once you have identified your company's open source strategy, you need to build infrastructure to support your open source engineering efforts. Figure 8 shows the four key pillars your infrastructure needs to support: community engagement, open source contribution, open source compliance, and open source usage.

The community is unique within these pillars because it involves all interactions between the company and the specific open source projects that the company is involved in from a consumption, compliance, and contribution aspect.

Open Source Consumption and Compliance Infrastructure

Consumption and compliance are closely related, especially in an enterprise setting where consumption usually equates using open source in a product or part of a service. Therefore, these core elements are often represented together, at least from an infrastructure perspective as any commercial use of open source software requires compliance with the corresponding licenses.

Figure 9 illustrates the various components required in building an infrastructure to support consumption and compliance.

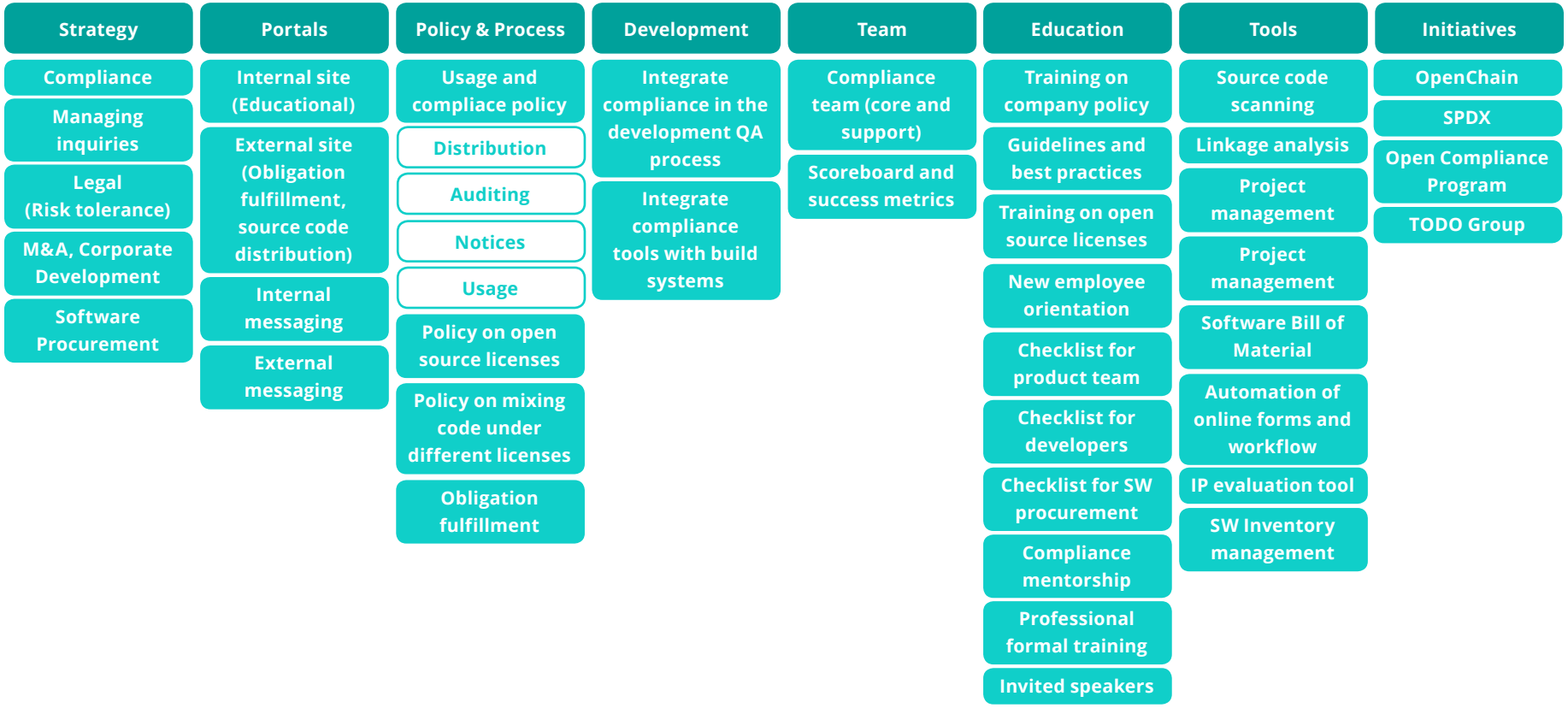

Figure 9: Open source consumption and compliance infrastructure

Framing the strategy

Open source compliance should include a framework for strategies related to:

- Legal compliance and risk tolerance,
- M&A and corporate development,
- Software procurement, and
- Managing compliance inquiries.

Defining the processes

The open source program office defines the processes and policies for how your company will handle code distribution, audits, notices, and usage. Additionally, the program office will publish internal policies and provide training highlighting which licenses are acceptable for which kinds of uses, how licenses can be combined and how your company will enforce compliance.

Establishing the team

Establish dedicated teams to ensure compliance. Agree upon their success metrics.

Getting the tools

Consider investing in certain source code scanning tools, as they help ensure compliance by automating the auditing.

Integrating into the process

Integrate open source compliance directly into the development and QA processes. Ideally, you will integrate compliance tools directly into the build systems.

Communicating the plan

You will need two websites to support your open source program. Publish your internal open source policies and processes on your intranet site. This is where you explain open source compliance to employees and guide them to your internal services to check on licenses, obligations, and approvals. You will also want an external website where you publish information promoting your open source projects, announcing news, and potentially potentially publishing compliance reports, notices, and source code.

Assisting other stakeholders

Ensure that everyone who deals with software procurement, development, distribution, and hiring is aware of open source compliance. Establish a process where they can easily raise questions and get answers.

Getting involved with the community

Certain foundations focus specifically on open source procurement, distribution, and compliance. It can be valuable to be involved in these foundations and have a strategy for what the company will achieve from involvement. In this specific case, The Linux Foundation hosts several initiatives that can support and enable your consumption and compliance activities. Please refer to the references sections for pointers to these initiatives.

Open Source Contribution Infrastructure

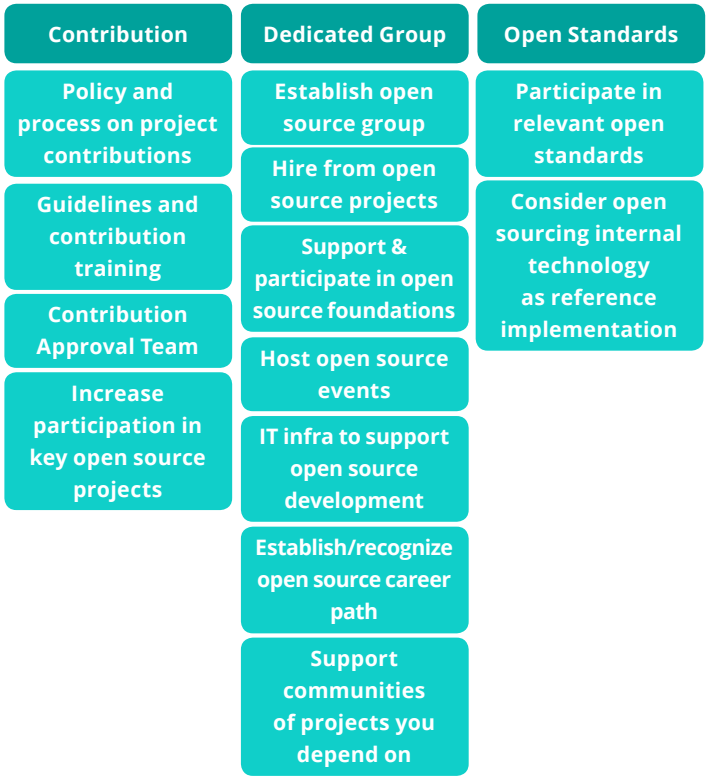

Figure 10: Open source contribution infrastructure

When you are ready to expand into contributing to open source projects (based on your internal priority map), you will need an infrastructure to support these contribution efforts and set a framework that will help you enforce company policy related to what can and cannot be contributed to any given project. This will also help avoid ad hoc contribution practices.

An open source contribution infrastructure has three core elements (Figure 10):

- Contribution building blocks that include a policy and process, a team that is responsible for vetting the contributions going to open source projects, and general guidelines.
- An open source group dedicated to open source participation, and involvement in relevant projects
- Participation in open standards relevant to the development activities or open source projects you are involved in.

Contribution enabling elements

Contribution support is the most critical component of open source contribution infrastructure, and includes:

- A contribution policy and process that express the company's policy and how it is executed via the contribution process.
- Guidelines to follow when contributing to open source projects.
- Training to ensure development teams are aware of the company policy and process with respect to contributions.
- Contribution approval team that is responsible for vetting all contributions.
- Priorities highlighting the areas where contributions are mostly needed in support for product development efforts.

Dedicated open source team

A team dedicated to upstream contributions can be extremely valuable and should be hired from the

community to support open source foundations, build IT infrastructure that is modeled after the open source community, and add open source metrics to developer HR career paths.

Involvement with open standards

Participation in relevant open standards bodies and trade organizations is often a critical component of the open source contribution infrastructure as you need you need people who are responsible for keeping the company up to date on any changes to relevant standards.

Join The Linux Foundation Compliance Initiatives

The Linux Foundation hosts several compliance initiatives that aim to improve compliance with licenses of free and open source software. Two prominent projects are OpenChain and the Software Package Data eXchange® (SPDX®). It is highly recommended to participate in these initiatives in support of your open source compliance practices.

OpenChain

The OpenChain Project helps to identify and share the core components of a high quality Free and Open Source Software (FOSS) compliance program. OpenChain builds trust in open source by making things simpler, more efficient and more consistent. It is the industry standard for managing Open Source compliance across the supply chain.

The OpenChain Project builds trust in open source by making open source license compliance simpler and more consistent. OpenChain consists of three core elements:

- The **OpenChain Specification** defines a core set of requirements every quality compliance program must satisfy.
- The **OpenChain Curriculum** provides the educational foundation for open source processes and solutions, whilst meeting a key requirement of the OpenChain Specification.
- The **OpenChain Conformance** allows organizations to display their adherence to these requirements. The result is that open source license compliance becomes more predictable, understandable and efficient for participants of the software supply chain.

Software Package Data eXchange®

The Software Package Data Exchange® specification is a standard format for communicating the components, licenses and copyrights associated with software packages.

The SPDX standard helps facilitate compliance with free and open source software licenses by standardizing the way license information is shared across the software supply chain.

SPDX reduces redundant work by providing a common format for companies and communities to share important data about software licenses and copyrights, thereby streamlining and improving compliance.

Hire or Promote a Leader for the Open Source Team

To ensure strong leadership of your open source engineering effort it's important to hire someone with a deep understanding of the methodology behind open source engineering. There are a number of traits this individual should possess:

- A strong engineering background,
- A solid understanding of common open source licenses and obligations
- Knowledge of industry practices
- Knowledge and experience in establishing corporate-wide policies and processes
- Technical knowledge related to the company's products
- Historical perspective on open source
- Knowledge of community consensus and practices
- Contacts in the key open source project communities
- Contacts in open source organizations

The TODO Group has published a template job specification for this role that you can customize to your specific needs; it is available from **http://todogroup.org/blog/sample-job-req/.**

Formalize an Open Source Career Path

Create an open source developer track in your human resources (HR) system so people hired as open source developers have a good sense of how their career will progress within the company versus other non-open source developers. Additionally, you should adjust performance-based bonuses to include goals related to open source development work. The metrics by which the performance of proprietary or closed source developers are measured are often different from those of open source developers.

At some companies, there is a clear distinction between open source and non-open source developers. However, in many companies, the line is much more fluid depending on organizational structure and open source strategy. In reality, all modern developers have to work with open source, there are no closed source developers. Rather, sometimes their code stays inside the company and sometimes it is published (contributed to a third party or published as a new project). Your HR track and incentives should reflect your organization's unique structure and approach to open source.

Finally, allow a work from home policy for open source developers, regardless of the general corporate policy related to this. Lately, we have witnessed a reverse in work from home policies across companies where many have either banned or created strict limitations to working from home. In the open source world, a work from home policy is almost mandatory because open source experts are located all over the planet, and this policy is often the only way to hire them. There are operational benefits to a flexible work policy as well.

Create or Outsource Open Source Training

Education is an essential building block in an open source program office, and it falls into two categories: technical training to expand open source technical knowledge and compliance training to ensure that employees possess a good understanding of policies governing the use of open source software.

The goal of providing this training is to raise awareness of open source policies and strategies and to build a common understanding around the issues and facts of open source licensing as well as the business and legal risks of incorporating open source software in products and/or software portfolios. Training also serves as a venue to publicize and promote compliance policies and processes within the organization and to foster a culture of compliance.

It is impossible for any company to hire all the senior and most expert developers in a given domain. This concept applies to the Linux kernel and any other prominent open source project. Therefore, you must have a way for your company to increase the competence of its developers in a given technical domain. In addition to technical training, you will also need training to teach the open source development model and the basic concepts of open source legal compliance.

Sample training courses include:

- Technical training that cover specific areas of the open source software. This is usually presented by maintainers or senior developers to grow internal open source expertise; this expertise is vital to pass on given how challenging it is to hire expert developers from open source communities.
- Open source development methodology course that teaches staff new to open source how open source development works and how to best get engaged.
- Open source compliance course that teaches staff the basics of compliance principles and open source licensing. This should also be used to inform them of your company's policy and process.

The OpenChain curriculum is a resource that is available to educate individuals involved with the software procurement process about open source compliance. In addition, The Linux Foundation offers a free **"Compliance Basics for Developers"** training course to educate developers on how open source compliance relates to their work.

Create Meaningful Metrics to Track Progress

Once you start implementing open source best practices, you will need proper metrics to drive the desired development behavior. However, the traditional metrics often used in product organizations do not apply in the context of open source development.

For example, tracking the number of changesets or lines of code can be a good metric for open source development impact, but you might have multiple instances of desired functionality being implemented upstream because your open source developers lobby for support from the community. In this case, the number of changesets or lines of code doesn't matter nearly as much as the technical leadership the team members provide to get code upstream and reduce the company's downstream maintenance efforts. Therefore, the metrics you track should account for both activities.

To start, create an internal system to keep track of developer contributions and impact. Metrics can include upstream development, support to product teams, knowledge transfer (mentoring, training), visibility (publications, talks), launching new open source projects, and establishing internal collaboration projects with other teams or groups.

There are several toolkits that help track source code contributions; for instance, The Linux Foundation uses a tool called gitdm, which produces the data reported in the annual **Linux Kernel Development Report.** This can be used to track both individual developers as well as the overall team performance. Individual developers can be tracked for the number of patches they submit, the patch acceptance rate (patches submitted divided by patches accepted), and the type of patch (e.g., a new feature, enhancement of existing functionality, bug fix, documentation, etc.). Other tools like GrimoireLab can also be used to chart and visualize the metrics you want to track.

The rest of this section will cover some of the metrics to consider tracking.

Number of Patches Submitted or Committed

The number of patches that are submitted or committed to an open source repository is the most basic metric to track and can be used to gather a general idea of activity in a project. It includes information about who authored the code, which project it was submitted to, and when it was committed. It should never be used on its own; rather it should either be used along with the other metrics below, or to identify starting points for further qualitative analysis.

Type of Patch

There is a variety of reasons for why code might be submitted to an open source project and there is a different type of value in each of them. Depending on what sort of problem you need to solve with a project, you can incentivize certain types of code, such as:

- Bug fixes
- Improvements to existing features
- Implementation of new minor features
- Implementation of new major features
- Contribution to test cases and test code
- Contribution to documentation

Patch Acceptance Rate

Just because you write code for an open source project does not mean you are guaranteed to get it accepted all the way through the peer review process; sometimes it requires multiple revisions over an extended period. The ratio of patches committed to a project vs. the number of patches accepted to the code base is a useful metric to track how successful your engineers are at getting code accepted the first time.

Patches Committed as Part of Collaboration Projects

If your open source engineers are submitting code directly on behalf of other teams, it can be useful to track their impact. This metric allows you to track the upstream work that results directly from internal collaborations with other teams, or collaborations with external organizations like universities.

Visibility Metrics

One of the benefits of open source engineering is soft influence that can be gained by being a leader within a community. This is established when employees improve their own visibility within their respective open source communities, and you can use metrics to track the number of publications, blog posts, conference presentations, media mentions, and more to encourage open source engineers to spend time improving their open source leadership.

New Projects and Initiatives

If one of your goals is to create new open source software that fills unmet needs, then you can track the number of new projects launched to incentivize teams to create them.

Contributions to Open Standards

In some cases, a direct link exists between a standard effort and an open source implementation that provides a reference implementation to the standard. In these cases, it can be beneficial to track employee contributions to the standard.

Establish Relationships with Open Source Foundations

Open source foundations are a great resource to expand your impact on an open source community. The best place to start is with the foundations that host initiatives that are relevant to your products or interests. Many companies find it worthwhile to get involved with well-known, established foundations such as The Linux Foundation, Mozilla Foundation, or Apache Software Foundation. If you are concerned primarily with the legal aspects, you can get involved with organizations such as the **Software Freedom Law Center** or the **Open Invention Network**. The primary goal is to identify opportunities that arise from getting involved with an open source organization that supports the ecosystem your company relies on.

Establish Plans to Release Proprietary Source Code Under an Open Source License

Many enterprises find checklists and templates useful for guiding code releases under an open source license; these minimize the time necessary to figure out what needs to be done. Granted, all cases are different in terms of what is being released as open source and for what purpose it is being released, but the mechanics are the same. Additionally, every time you follow these tools, it is a good idea to update and add any notes that will help improve the process in the future.

The Linux Foundation's guide on **Starting an Open Source Project** was created to help enterprises who

are already well versed in open source learn what they need to know to start their own open source projects. The guide goes through the process, from deciding on what to open source, to budget and legal considerations, and more. It also offers a sample checklist that covers all major tasks that need to be completed from the conceptual phase all the way to the launch and maintenance phase of the project. This **checklist** offers a great example of a guide your company can use so various teams have access to knowledge that improves the overall understanding of what's involved with the open sourcing process.

Encourage Internal Collaborations

Internal collaboration with other business units that use the same open source projects in their products is a great way to expand the impact of an open source engineering team. These collaborations can take one or more of many forms, including:

- Deliver training to product developers
- Run a workshop on a specific topic or problem
- Develop new functionality
- Troubleshoot and resolve issues and bugs
- Upstream existing code for which they have no resources to do
- Help get them off an old fork and onto a mainline version

These internal collaborations serve many purposes, but two are particularly important:

- They present incredible visibility opportunities for the open source team with other organizations or teams within your company.
- They allow you to become internal experts on open source, opening new opportunities such as becoming the upstream partner to R&D and product teams

Internal collaborations are a challenge, especially in large companies. Open source makes that challenge a little less difficult given the open nature of the software and the fact that being aligned with upstream has huge benefits; other teams are typically eager to collaborate to make that happen.

Provide a Flexible IT Infrastructure

Provide a flexible IT infrastructure that allows open source developers to communicate and work with the open source projects with minimal challenges. Additionally, set up internal IT infrastructure that

matches the tools used externally to help bridge the gap between internal teams and the open source community. Much of your infrastructure will naturally evolve along with your organization's open source culture, but it is important to be aware of the necessity and plan for its implementation.

There are three primary domains of IT services that are used in open source development: knowledge sharing (wikis, collaborative editing platforms, and public websites), communication and problem solving (mailing lists, forums, and real-time chat), and code development and distribution (code repositories and bug tracking platforms). Some or all of these tools will need to be made available internally to properly support open source development. These open source practices typically require an IT infrastructure that is free from many of the standard limiting IT policies, so there is a chance this might conflict with existing company-wide IT policies. If so, it is vital to resolve these conflicts and allow open source developers to use the tools they are familiar with.

Host Open Source Events

Support your developers to attend and participate in open source conferences and events, including local community meetups, hackathons, and summits. Such participation helps them connect at a personal level with their peers, build relationships, have face-to-face social interactions, and participate in technical discussions that guide the project's direction.

If your developers have work that others might be interested in, help these developers prepare content to present. Finally, you can also sponsor events, both big and small, to increase external visibility within the project's community. These events are also a great venue to look for talent!

A good place to start is to consider **Linux Foundation events** and identify the ones that are closely related to the technologies you're involved in. You can either send your own developers that are looking to expand their knowledge or influence or send management representatives to identify important individuals and recruit new talent.

Collaborate with Universities on Open Source R&D Projects

Many universities and schools are eager to work with companies who offer learning opportunities for their students, because it provides the opportunity to get real world software development experience. This

relationship is often beneficial to the companies involved because it can be a great way to develop and attract new talent in existing open source communities. This is particularly useful for projects with a shortage of experienced developers. You can't hire all the smart people in the world, so you'll need to find a way to tap into new knowledge and influence favorable outcomes in external projects.

Explore Inner Source Practices for Internal Projects

Inner source is a term that describes the process of taking the lessons learned from open source development methodology and applying them to the development methodology used inside the company. The goal is to incubate the same values in the enterprise as those you get out of a collaborative development model and methodology. Inner source methodology enables new employee-to-employee relationships that are cross-functional and touch on multiple product domains. This same relationship will exist between your employees and members from the open source community, which also often include other employees from your company

Why Incorporate Open Source Principles?

Why incorporate open source principles for your internal organization? For starters, they work and particularly at large scale. These processes have enabled the success of some of the largest software projects in the world, and they are proven methods adopted by many Fortune 500 companies.

Adopting innersource methodology often brings a wealth of benefits:

- Faster release cadence
- Improved source code quality
- Increased innovation
- Increased internal information sharing
- Reduced development costs
- Increased internal collaboration
- Increased moral retention
- Increased internal communication

Doing this also prepares your company to work effectively with external open source communities because it allows your employees to interact with both other employees and external community members without context switching. Incoming employees will also be able to adapt more quickly because they are often already familiar with this development model. Finally, your partners are probably already adopting many of these development practices, so your own adoption can improve your integration with the commercial ecosystem.

What Does this Mean in Practice?

To make this a reality, you need to start opening source code bases to all employees in the company, and anyone should be allowed to use them, contribute back to them, or fork them as needed. Ideally, you will have a team that is responsible for code stewardship and that makes sure code submissions are reviewed and applied. Companies that use inner source methods typically use mailing lists, chat rooms, and ticketing systems that are fully open to anyone at the company for discussing changes and asking questions. Please check the references for more information on how to implement inner source practices.

Important Open Source Workflow Practices for Enterprises

There are several essential steps to implementing proper inner source practices:

- Visibility
- Forking
- Pull/Merge Requests
- Peer Review
- Release Early, Release Often
- Testing
- Continuous Integration
- Documentation
- Issue Tracking

The remainder of this section will cover each of these concepts.

Visibility

All internal software projects should be visible to all employees in the company by default, and communication channels should be as open as possible. This enables cross-pollination between teams and provides an opportunity for employees to feel a shared sense of responsibility. It also facilitates ad-hoc communications and relationship building between people in your organization.

Occasionally projects require extra protections because of sensitive information related to their development, and these can be kept private to select groups. However, these should be the exception rather than the rule.

Forking

Everyone who can see internal code should be allowed to create a copy (fork) where they can make changes freely; these forks should then be made visible to everyone for them to do the same. This encourages greater understanding and better integration of your software stack across the various teams within your organization and results in beneficial cross-pollination.

Pull/Merge requests

People outside the project should be allowed to suggest changes to the project itself. You should encourage employees to welcome participation from teams outside their own and consider their contributions.

Peer review

Peer review is an essential component of open source development because is reduces variations in style, and ensures code keeps up with the project's quality standards. Code should always be reviewed by people with a strong understanding of the code base prior to being committed. These reviewers should also provide guidance to the submitters to refine their style over time and increase the likelihood of acceptance.

Release early and often

More frequent releases with fewer features and regular update releases are important for improving code development efficiency because it makes release integration and testing much easier. Additionally, it makes it easier for bugs and regressions to be identified and fixed. You should create a plan for a release hierarchy for things like nightly, weekly, milestone, release, long term, etc.

Testing

Unit and integration tests should be incorporated directly into the software development process so changes can be made with less fear of creating new problems. Potential issues should be identified and addressed early to avoid accumulating technical debt.

Continuous integration

Your software development process should implement continuous integration so that every proposed change is automatically tested with the results shown in the changes themselves.

Documentation

All software projects should include a README that describes what the software does, why it's important, how to run it, and how to develop it. Good documentation is often the difference between a successful project and a failed project.

Issue tracker

Issue trackers for all internal projects should be open to everyone in the organization to submit a bug, request a feature, or ask a question. This benefits the projects themselves by increasing the amount of feedback they receive. It also benefits other teams by expanding the expertise that is available to them.

Join the TODO Group

The TODO group is a collection of tech companies who collaborate on the policies, practices, and pragmatics of running an open source program office. Their collaboration is managed as a community project under the Linux Foundation, and they are a resource to companies who are just starting to get their open source programs established. The TODO group publishes guides for open source program offices, and its members frequently present at open source conferences on best practices. Visit todogroup.org to learn more and reach out to **info@todogroup.org** to find out how to join.

Update M&A Practices

If your company is considering a merger or is the target of an acquisition, it should structure its compliance program to offer the necessary level of disclosure and provide representations. Company policies regarding merger and acquisition transactions need to be updated to account for open source software.

Corporate development must mandate that source code be evaluated from a compliance perspective prior to any merger or acquisition to avoid surprises that might derail discussions or affect the company's valuation. For the acquiring company, comprehensive code evaluation assures accurate valuation of software assets and mitigates the risk of unanticipated licensing issues undermining future value. Additionally, the acquiring company may include provisions in the purchase agreement requiring the disclosure of open source that is subject to the transaction. Diligence practices should be updated to require open source disclosure and include guidance regarding the review of any disclosed open source and licenses.

Update Outsourced Development Agreements

Agreements relating to outsourced development of software should also be updated to reflect open source compliance procedures and to ensure that other provisions of these agreements (such as representations and warranties) are broad enough to cover the risks posed by open source. Corporate Development must mandate that all source code received from outsourced development centers goes through the compliance process to discover all open source being used and to ensure proper actions are being taken to fulfill license obligations.

Challenges You Will Face

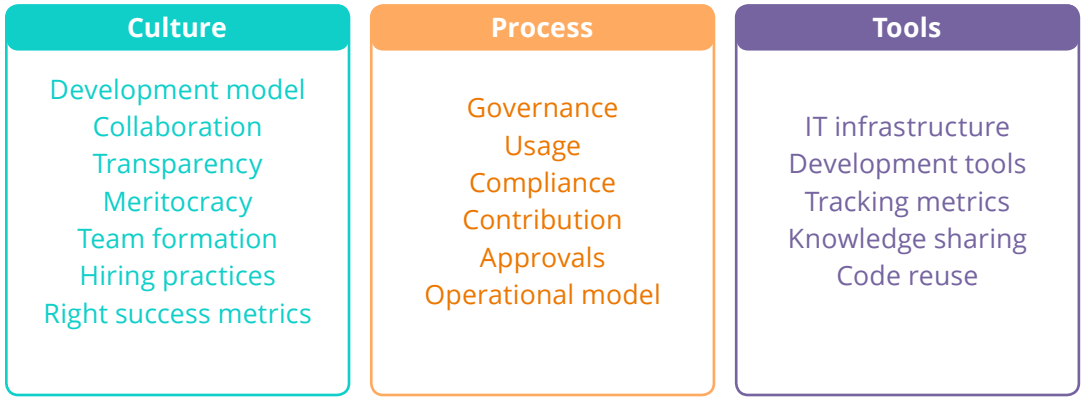

Figure 11: The three major challenge areas for open source engineering

There are three general areas where open source programs typically face significant challenges: culture, processes, and tools (Figure 11). Within each of these categories are several elements that you will need to address to fit the open source model.

Culture

Cultural challenges often stem from the fact that there is a gap between traditional software development practices and the requirements of open source development. You can bridge this gap by hiring open source experts and having them train other groups that aren't familiar with the open source development model. These experts can provide guidance to help

- Create internal processes that follow the open source development practices of release early and often, and peer review.
- Improve transparency between departments to encourage more cross-functional collaboration.
- Form your engineering teams around the ideals of meritocracy.
- Establish proper success metrics to encourage open source and cross-department contributions.

Processes

Open source development is dynamic, moves very quickly, and has special requirements for compliance. Companies that do not adapt their internal processes to meet this type of development can easily be left behind. Developers need to be enabled to contribute code upstream quickly and any internal code policies need to be modified to allow this.

First, it is vital to have a team in charge of maintaining proper open source compliance to avoid legal problems. You'll also need to have a simple internal approval model for open source use and contributions. You might have to move from highly complex and cumbersome policies to a more simple approach for receiving, reviewing, and approving source code contributions. It is a function of balance between all parties involved: legal, engineering, and open source. The compromise should support a dedicated open source team that has a blanket approval to contribute to a number of open source projects. This does not need to be the case for all teams, and you can use different levels of approval depending on the nature of the code being contributed (e.g. simple bug fixes, code to improve existing functionality, code that offers new functionality, or starting a new project).

Tools

The final major challenge area are is building tools that are compatible with the open source development model from the start; you need to create a setup that fulfills the needs of the open source program office and also meets corporate IT guidelines. The biggest challenge here is created by the fact that open source engineers deal primarily with collaboration and communication tools that are available to the public; communication and code they submit to these platforms is not directly related to internal IP and should be treated as such.

Open source engineers need greater flexibility to communicate with external participants via email, chat, and code development platforms, and their IT tools need to facilitate this. For example, emails to an open source project should never have anything attached to them that claims the content as intellectual property of the company it was sent from, and communication with public mailing lists from company accounts should be allowed without obstruction.

Additionally, a large portion of open source development occurs on Linux, and your engineers will need devices that support the development distribution of their choice because the IT environment you create should allow developers to join the team without requiring any major changes to the way they work. Ensure that all engineers who use Linux are able to access all vital internal tools and resources either on Linux, or via a separate compatible device.

Finally, since open source development happens all over the world, your tools will also need to support fully-distributed teams work in remote offices or their own home. If your company is located primarily at a single location, this will require you to enable tools that allow remote workers to connect to internal business resources through a VPN or similar technology. You will also need to evaluate your IT policies for things like helpdesk support to ensure you have secure methods to resolve IT issues for remote employees. For example, remote workers will not typically be able to visit an IT helpdesk in person when they have issues with internal resources, so you might need to establish an alternative.

Conclusion

It is no secret, open source is eating the software world; you can either watch the show or be a part of it. Mastery of open source requires a strong strategy that encompasses open source consumption, participation, contribution, and leadership, and each of these requires their own incremental effort and investment into improving open source engineering:

- **Consumption -** Establish internal infrastructure that enables proper open source practices and incorporates open source policies, processes, checklists, and training.
- **Participation -** Begin engaging with the open source community on communication platforms and at events. Sponsor projects and organizations that are important to open source software you rely on for your products.
- **Contribution -** Hire or train developers that focus specifically on open source contributions and deploy the necessary tools to support internal open source engineering.
- **Leadership -** Increase engagement with open source communities, open standards bodies, and foundations. Launch new open source initiatives and increase your visibility in open source communities.

Open source mastery is no secret; if you follow the steps and principles offered in this book, you should find yourself well on your way. Welcome to enterprise open source.

References

OpenChain
https://www.openchainproject.org

SPDX
https://spdx.org/

Open Compliance Program
https://compliance.linuxfoundation.org/

TODO
http://todogroup.org/

Open Source Compliance in the Enterprise
https://www.linuxfoundation.org/publications/open-source-compliance-enterprise/

Linux Foundation Enterprise Guides
https://www.linuxfoundation.org/resources/open-source-guides/

Practical GPL Compliance
https://www.linuxfoundation.org/publications/practical-gpl-compliance-download-this-free-guide-today/

Open Source Audits in Merger and Acquisition Transactions
https://www.linuxfoundation.org/resources/open-source-audits-merger-acquisition-transactions/

Recruiting Open Source Developers
A guide published by The Linux Foundation.

OpenChain Curriculum
Available as a free download, it is a great resource to educate individuals involved with the software procurement process about open source compliance.

Linux Foundation Free Compliance Training For Developers
The Linux Foundation Compliance Basics for Developers course is a great, free resource to educate developers on how open source compliance relates to their work.

Software Package Data eXchange®
https://spdx.org/about

Innersource: A Guide to the What, Why, and How
A practical guide on innersource is and how it can be applied to your company by Jono Bacon

The Linux Foundation promotes, protects and standardizes 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 our other initiatives please visit us at **www.linuxfoundation.org**