THE LINUX FOUNDATION | Research

In partnership with:

LF AI & DATA

# A Guide to Enterprise Open Source

## Developing and Executing Open Source Software Strategy

**May 2022**

Ibrahim Haddad, Ph.D.
Executive Director, *LF AI & Data Foundation*

With a foreword by David Marr, Vice President,
Legal Counsel, *Qualcomm Technologies*

# Contents


Foreword ..... 3

Infographic ..... 4

Abstract ..... 5

Introduction ..... 6

Leveraging Open Source Software ..... 7

Preparing the Enterprise for Open Source ..... 9

- Identify Reliance on Open Source Software ..... 9
- Identify Open Source Skills Portfolio ..... 10
- Consider Joining the TODO Group ..... 10

Developing an Open Source Strategy ..... 11

- Identify Enterprise Objectives ..... 11
- Identify Your Stage of Involvement ..... 13

Setting Up Your Infrastructure for Implementation ..... 16

- Start with Consumption and Compliance ..... 16
- Implement Core Compliance Practices ..... 17
- Provide Structure for Contributions ..... 19
- Update M&A Practices ..... 20
- Participate in Compliance Initiatives ..... 20

Setting Up Your Talent for Success ..... 21

- Name a Leader for the Team ..... 21
- Formalize an Open Source Career Path ..... 21
- Offer Open Source Training ..... 21
- Encourage Internal Collaborations ..... 22
- Provide Flexible IT Services ..... 22
- Track Progress via Meaningful Metrics ..... 23
- Apply Open Source Methods to Internal Projects ..... 24
- Raise Developer Visibility ..... 25

Challenges ..... 26

- Culture ..... 26
- Processes ..... 27
- Tools ..... 27
- Continuity ..... 27
- Education ..... 28

Closing Remarks ..... 29

Linux Foundation Resources ..... 30

About the Author ..... 31

Disclaimer ..... 31

# Foreword

In the early days of open source, there was ad hoc, improvised, license compliance at best. In some places there was no attention paid to compliance at all. That is not an indictment of any particular folks at the time because the real-world consequences of poor compliance practices were at first indeterminate. As with most new areas of practice, we matured in fits and starts, with flame wars on message boards, web pages dedicated to "Halls of Shame"—to list those who were known to have failed to adhere to license terms—and stories of maintainers threatening to bring lawsuits. And then those very same threatened lawsuits came, at first by the "pilgrims" who sought nothing more than a commitment to license obligations, minus the costs of bringing the court case. But that was immediately followed by some who saw license compliance litigation as a lucrative revenue stream, which was widely perceived as more mercenary than community-minded, and which then was actively discouraged by numerous open source community leaders.

Ibrahim's work in the area of open source license compliance is informed by all his years working in open source. He is one of those who you first hear of, before you meet. For my part I first heard of him during a conversation at a coffee shop on the Upper West Side of Manhattan in 2005. I was speaking with Professor Eben Moglen from the Columbia Law School, co-author of the General Public License version 3, who was politely indulging my complaints of the then-dearth of talent in the still relatively new area of open source license compliance—in particular the lack of those who had the needed skills that blended technical fluency and license know-how. As Ibrahim came highly recommended, I sought him out at his next conference presentation. Over the years I found in him a kindred colleague. Ibrahim is one who cares very much about not just mastering a field such as open source license compliance, but also transferring his knowledge to others.

In the following pages, you will gain an understanding of how broadly open source impacts our digital world today, and how it drives most of the conveniences of modern society. Enabling those outcomes are the legions of programmers who write and contribute to community code projects which are in turn consumed by product companies. And within those product companies an arcane blend of software tooling, code management, and legal disciplines coalesce to form what we call an Open Source Program Office to manage the menagerie of open source licenses and their myriad associated obligations in the context of software engineering processes. Ibrahim's book delightfully makes straightforward all that is complex in this space.

**David Marr**, *Vice President, Legal Counsel, Qualcomm Technologies*

## OPEN SOURCE OPPORTUNITITES

In vertical software stacks across industries, open source penetration ranges from **20 TO 85 PERCENT** of the overall software used.

## OPEN SOURCE OPPORTUNITITES

**OSS HAS BECOME A FOUNDATION** for new commercial products and services, and is essential to many organizations' software development workflow.

## OPEN SOURCE OPPORTUNITIES

**OSS ADAPTS** to different business models, regardless of the industry vertical in question.

## OPEN SOURCE OPPORTUNITITES

OSS can improve enterprise product development and **IMPROVE ENTERPRISE TALENT AND SKILLS DEVELOPMENT.**

## OPEN SOURCE OPPORTUNITITES

**THROUGH OPEN SOURCE PARTICIPATION,** enterprise developers can build on the work of others, experiment with new features, and focus more of their efforts on differentiation.

## OPEN SOURCE OPPORTUNITITES

Participation in open source projects positions organizations to leverage external R&D, spot opportunities to commercialize discoveries, and increase enterprise speed to market.

## PREPARING FOR OPEN SOURCE

**IDENTIFY RELIANCE ON OPEN SOURCE SOFTWARE,** clarify needed open source skills, and join organizations such as the TODO Group.

## OSS STRATEGY DEVELOPMENT

**Identify your enterprise's open source objectives, strategy, opportunities, and current stage of involvement.**

## ENTERPRISE INFRASTRUCTURE

Set up your organization to implement open source, from compliance to consumption to contribution.

## NURTURE OPEN SOURCE TALENT

**IDENTIFY CAREER PATHS** for open source talent and leadership development.

## ENTERPRISE CHALLENGES

Organizations face significant challenges in open source in five specific areas: **culture, processes, tools, continuity, and education.**

## OPEN SOURCE SUCCESS FACTORS

**MASTERY OF OPEN SOURCE REQUIRES A STRATEGY** that encompasses open source consumption, participation, contribution, and leadership.

Copyright © 2022 The Linux Foundation | May 2022
This report is licensed under the Creative Commons Attribution-NoDerivatives 4.0 International Public License

# Abstract

Open source software (OSS) has transformed our world and become the backbone of our digital economy and the foundation of our digital world. From the internet and the mobile apps we use daily to the operating systems and programming languages we use to build the future, OSS has played a vital role. It is the lifeblood of the technology industry. Today, OSS powers the digital economy and enables scientific and technological breakthroughs that improve our lives. It's in our phones, our cars, our airplanes, our homes, our businesses, and our governments. But just over two decades ago, few people had ever heard of OSS, and its use was limited to a small group of dedicated enthusiasts.

Organizations involved in building products or services involving software, regardless of their specific industry or sector, are likely to adopt OSS and contribute to open source projects deemed critical to their products and services. Organizations are creating open source program offices (OSPOs) to manage their open source activities from their adoption of OSS and compliance with applicable licenses to their participation in open standards and foundations.

Over many years, new industries and thousands of organizations have entered the open source ecosystem. In the early days, some organizations leapt into OSS without a proper strategy and an execution plan; they did not emerge as winners. Others took a deliberative approach that embraced OSS methodology and engineering practices; they came out as leaders for open source activities in their industries or verticals. To guide their ongoing use of OSS and their engagement with the open source ecosystem, they have developed open source strategies appropriate for their organizational constraints and industry challenges.

This guide offers a practical and systematic approach to establishing an open source strategy, developing an implementation plan, and accelerating an organization's open source efforts.

# Introduction

In December 2000, IBM announced its landmark investment of $1 billion ($1.58 billion today) in Linux and open source software. This commitment raised Linux's profile significantly as an alternative operating system. Other companies started evaluating Linux and figuring out which OSS components were worthy of corporate funding and adoption.

The availability of enterprise-grade OSS is changing how organizations create, develop, deliver, and maintain products and services. The global development community, an open governance model, access to publicly available source code, and the adoption of approved open source licenses have had a cumulative effect on the enterprise mindset. Organizational leaders now think differently about how they procure, implement, test, deploy, and maintain software. This transformation can come with several benefits—lower development costs, faster product development, higher quality of code, and more.

The first step of an organization's journey in open source is to understand how it would like to benefit from an open source engineering effort, involvement in OS projects, and collaboration with industry players, universities, and OS foundations. While no two organizations are exactly alike in their uses of and benefits from OSS, all successful OSS implementations have two elements in common: a strong OSS strategy, and a clear plan of execution.

Generally speaking, we define an open source strategy as a concise, high-level document that maps the organization's business objectives to open source software use and management directives. It is the reference document for establishing an agreement on future open source policies and processes. Leaders must consider all aspects of their organization that could benefit from an open source strategy. For example, many organizations use it as a mandate for implementing open source best practices and procedures.

To develop and implement an OSS strategy, an organization must also articulate a set of business-level objectives and identify all constraints to adopting OSS and contributing to projects integrated into their products and services. This guide will help your leadership team to craft a strategy right for your organization—one that your people can live into—that transforms your approach to OSS development and engineering from defensive to offensive.

# Leveraging Open Source Software

Open source software has become essential to modern society, from the devices we carry to the systems controlling our critical infrastructure. This section discusses the six strategic objectives that organizations can achieve, depending on how they adopt OSS, incorporate it into their products and services, and participate in OSS projects and communities.

**1. Software leadership.** Just as location is the foundation for value in real estate, software has become the defining factor of value in every industry. The world of software is changing. Once a niche technology of fringe startups, OSS is now a mainstream technology in Fortune 500 companies.[1] In vertical software stacks across industries, open source penetration ranges from 20 percent to 85 percent of the overall software used. No matter what industry you are in or what product or software you develop, you likely rely heavily on OSS. This significant adoption of OSS has revolutionized the technology landscape. But it has also required a shift in software leadership, from a closed model of proprietary software to an open model of OSS and from leading its usage to leading communities of developers who build, improve, and refine it.

**2. Enterprise dependency on open source.** Open source software has become a foundation for new commercial products and services, and its development model is essential to many organizations' software development workflow. An enterprise can rarely build a product without using OSS or adopt a product that does not draw from OSS code bases.

**3. Business model adaptability.** A business model is the architecture for creating value through a product or service. Putting specific license requirements aside, we can adapt open source software to various business models, making OSS suitable for nearly any industry across an industry's software supply chain. **TABLE 1** shows the various ways an enterprise can use OSS to implement different business models.

**TABLE 1**

## Different open source business models

| | BUILDING OPEN SOURCE | BUILDING *WITH* OPEN SOURCE | BUILDING *FOR* OPEN SOURCE | BUILDING *ON* OPEN SOURCE |
|---|---|---|---|---|
| **WHAT A COMPANY CREATES** | OSS | Basic low-level OSS libraries and components | OSS | OSS product or service |
| **WHAT A CUSTOMER PAYS FOR** | Expert services and products | Proprietary software or services on top of open source | Integration and services | Higher layers of software stack |

1 Megan Barnett, "How Corporate America Went Open Source," *Fortune*, Fortune Media IP Ltd., 16. Aug. 2010; and Raju Shahi, "Commercial Companies Built on Top of an Open-Source World," *Hackernoon*, Artmap Inc., 9 June 2020.

**4. Product improvement.** Open source software can improve enterprise product development (**FIGURE 1)** in two ways:

- ***Directly*** when internal activities enhance an enterprise's open source development and its own products or services by (1) fulfilling internal R&D and product teams' requests through open source projects, (2) contributing internal code to open source projects, thereby reducing the difference between the open source branch of code and the internal branch, and (3) helping to resolve compliance issues and address inquiries.
- ***Indirectly*** where an enterprise's open source activities improve its own talent's skills and enhance OSS by (1) stabilizing upstream code used in its products, (2) participating in internal policy discussions and decisions to make sure they support open source development, (3) influencing upstream projects via thought leadership and consistent code contribution, (4) participating in external technical discussions to influence open source communities, and (5) participating in internal technical discussions to align the enterprise's direction with that of a specific open source project community.

**5. Focused/faster innovation.** As the proverb goes, "many hands make light work." In open source projects, many hands also make faster work. Collaborators across organizations can make changes and revisions to their shared code more quickly. As a consequence, enterprise developers can (1) build on the work of others, (2) experiment with new features and optimizations much faster within the community than in-house, leading to breakthroughs in technology; (3) focus their efforts on differentiation in higher stack levels and improve their unique value to their consumers. In doing so, they contribute to an organization's overall innovation pipeline. Small startups can bring new ideas to market faster, and

**FIGURE 1**

How Open Source Enhances Enterprise Products

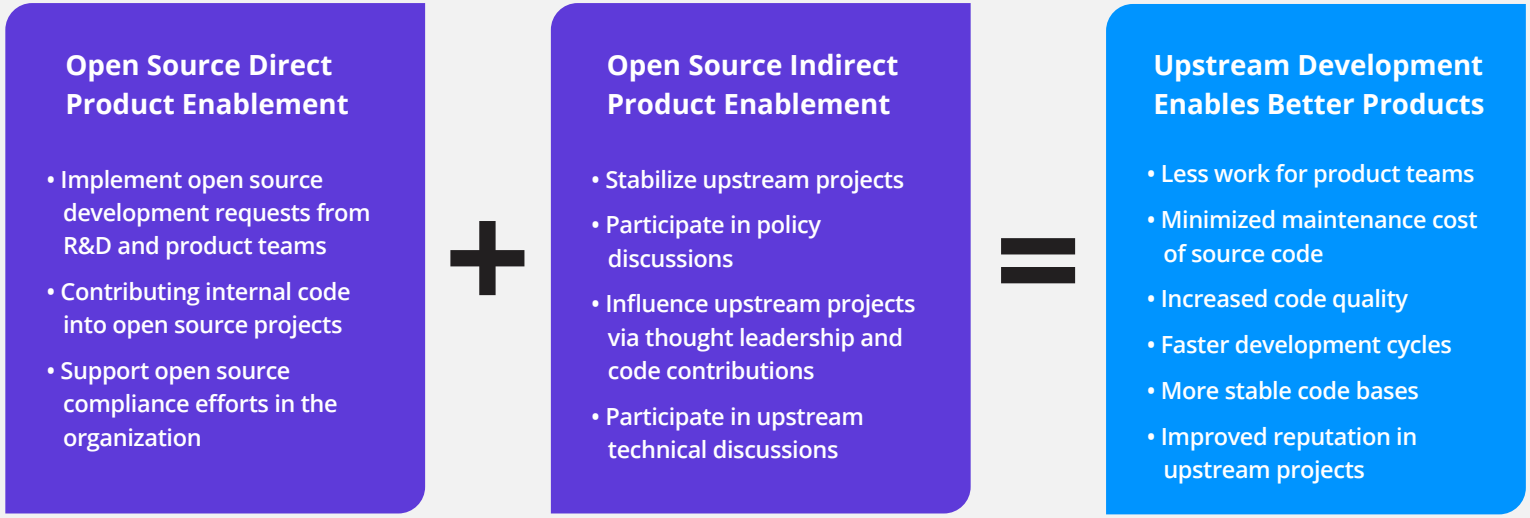
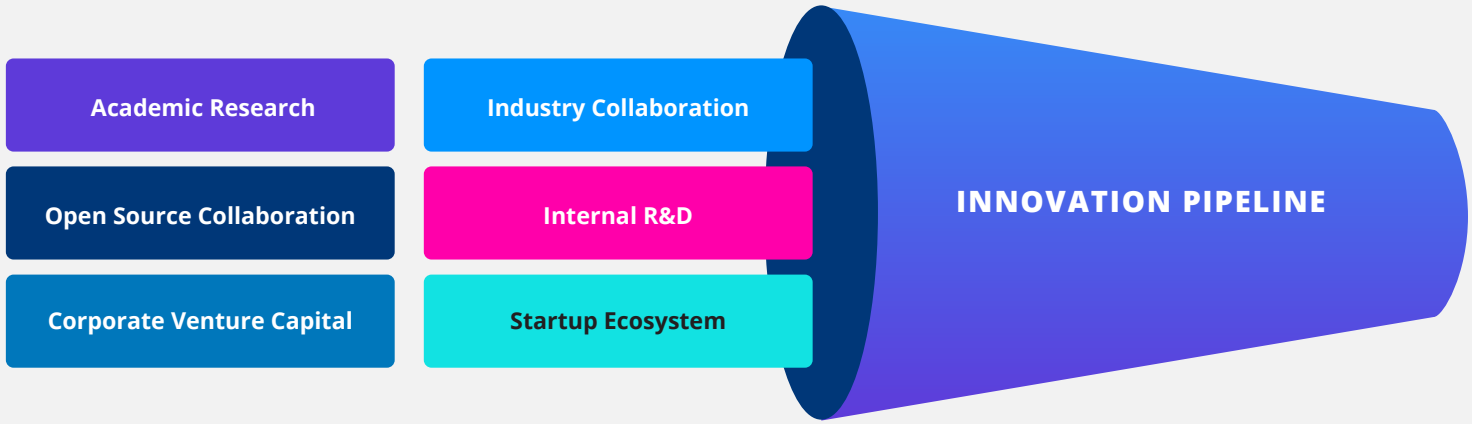


FIGURE 2

Open Source Accelerates Product Innovation




larger companies can experiment with new ideas without the risk of losing control over their intellectual property.

6. **Open source R&D helps drive enterprise innovation.** Participation in open source projects positions organizations to leverage external research and development (R&D), spot opportunities to commercialize discoveries, and increase enterprise speed to market with these breakthrough products and services. Shared access to open source R&D drives this type of innovation across business units. It is one of many innovation accelerators such as academic research, corporate venture capital, industry collaboration, internal R&D, and startup ecosystems.

# Preparing the Enterprise for Open Source

IBM's investment was its starting point for enterprise open source adoption in 2000. Since then, thousands of companies have entered the open source ecosystem (Figure 3). All have had much to learn about working with OSS and its respective communities; and through their participation, the open source model has become the new normal for software development.

How can companies now joining the OSS community minimize their enterprise learning curve and accelerate their returns on consumption and participation? The following sections explore several lessons learned from over two decades of enterprise experience with OSS.

## Identify Reliance on Open Source Software

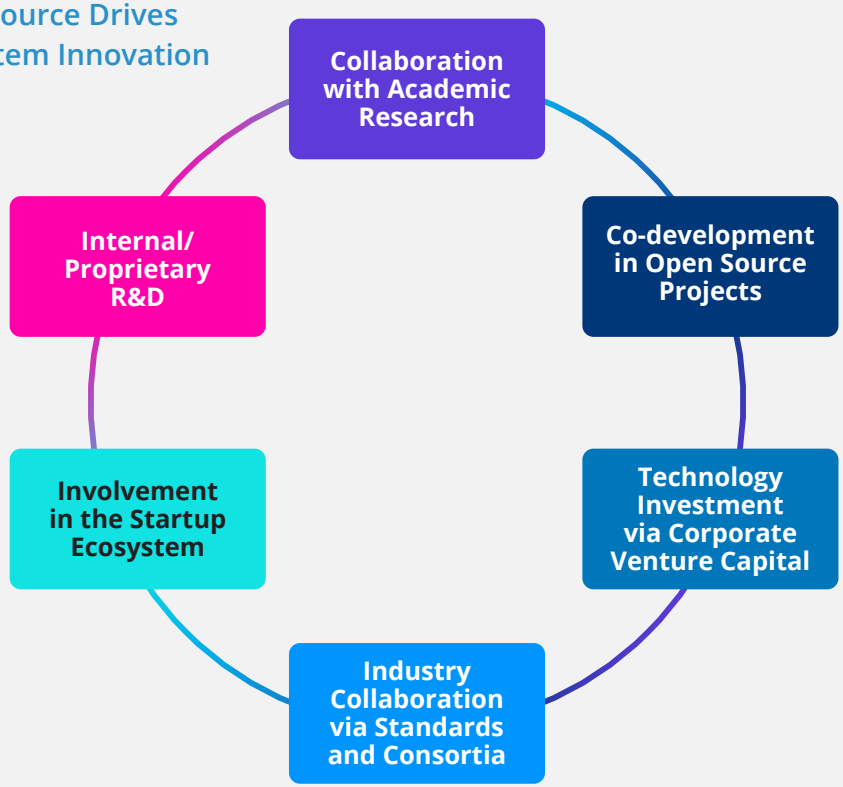
The first step toward improving your organization's open source engagement is identifying where the organization relies on OSS. With increased reliance on OSS, companies have several options to explore. One option is to focus on the software that many business units use. A second option is to focus on software that poses greater compliance risk than others. For example, mobile apps and embedded hardware may present more risk than your datacenter code. Pursuing these options, you can show a return on investment across multiple business units or high-risk areas and make a case for more funding and support.

Another option is to focus your contributions on upstream projects that directly benefit your organization's strategy and products rather than hop among different interesting projects. If your enterprise considers open source engineering a cost center, then focus on projects that support product development.

Conduct a yearly review of the organization's product portfolio. Determine which products draw from which open source projects, and then prioritize those projects that support the most products. This approach puts limited software engineering resources to maximum use.

FIGURE 3

Open Source Drives Ecosystem Innovation




## Identify Open Source Skills Portfolio

Once you have identified where your organization relies on open source, you can begin to determine what kind of expertise you need and whether you can develop it internally or access it externally. By hiring open source developers from the OSS community, your organization gains skills, recognition, and mentorship capabilities.

Two or three new hires can have a noticeable impact on a large project such as the Linux kernel. They can also attract other hires and mentor junior developers. The goal is to find people who have enough peer recognition and influence in the community.

Consider these four pillars when hiring: technical domain expertise, open source methodology and experience, working practices, and alignment between corporate interests and the candidate's interests. Motivating senior open source developers is easier when their interests and skillsets align with corporate interests in a given project. For example, a Linux memory management expert may not enjoy working on file systems. Finding this kind of match is critical for a long-lasting relationship.

## Consider Joining the TODO Group

The TODO group is a collection of tech companies that collaborate on running an open source program office's policies, practices, and pragmatics. Their collaboration is managed as a community project under the Linux Foundation, and they are a resource to companies that 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. To join, reach out to info@todogroup.org.

# Developing an Open Source Strategy

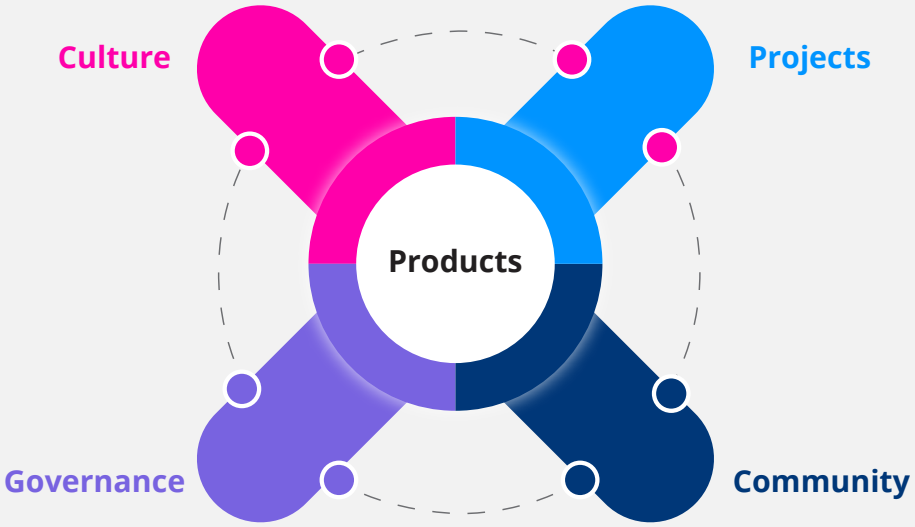
Open source software is the most impactful way to deliver software at scale. The ability to harness the power of an open ecosystem, in which anyone can contribute, to solve complex problems is unique. It has the potential to be more impactful than closed source alternatives. However, the full power of open source is only realized when a robust open source strategy leads it.

There are many questions to answer when developing an open source strategy. Organizations should aim to address these questions early in the process. An open source strategy should address four essential requirements (**FIGURE 4**):

1. The open source projects in which the enterprise seeks to participate.
2. The respective open source project communities the enterprise wants to engage.
3. The effectiveness of enterprise open source governance.
4. The openness of enterprise culture and whether it will welcome or reject open source efforts.

**FIGURE 4**

The Four Poles of Open Source Strategy




## Identify Enterprise Objectives

This section cover the three primary questions an organization needs to address before creating its open source strategy.

### CORPORATE OBJECTIVES

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

The first question you answer will identify where OSS fits within your overall corporate objectives. Companies can benefit from open source use and contribution in many areas. The open source strategy often addresses specific needs. Once you identify those needs, you can focus on the areas where open source can best benefit your organization. Your objectives could include any of the following:

- Create a long-term, high-quality road map to take a 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.

## INTELLECTUAL PROPERTY STRATEGY

How can an open source strategy help your organization achieve its intellectual property (IP) strategy?

Open source software can be an effective way for companies to build products and services that meet their needs without investing heavily in coding and design. It is often the precursor to a commercial product, where the enterprise uses the code to begin developing a product or service that it will eventually market and sell. OSS is also an excellent way for companies to achieve their IP strategy. Open source licensing differs from proprietary licensing, and so organizations must account for how differences in licenses affect their ability to benefit from the use and development of OSS. Business objectives could include any of the following:

- Determining a licensing strategy that best allows the organization to benefit from external involvement and still improve proprietary products.
- Mitigating IP risk by complying with licenses of OSS used in products and services.
- Differentiating proprietary IP significantly by improving core open source components.
- Releasing proprietary source code under an open source license.

Many enterprises find checklists and templates helpful in guiding code releases under an open source license. Every case is different, but the mechanics of releasing open source code are similar. The Linux Foundation has created a guide, "Starting an Open Source Project," to walk organizations through the process, from choice of project, budgeting and legal considerations, to project launch and maintenance. It offers a sample checklist as an example of what your organization can create so that various teams understand what's involved with the open sourcing process.

Open source licensing differs from proprietary licensing, and so organizations must account for how differences in licenses affect their ability to benefit from the use and development of OSS.

## OPPORTUNITIES THROUGH OPEN SOURCE

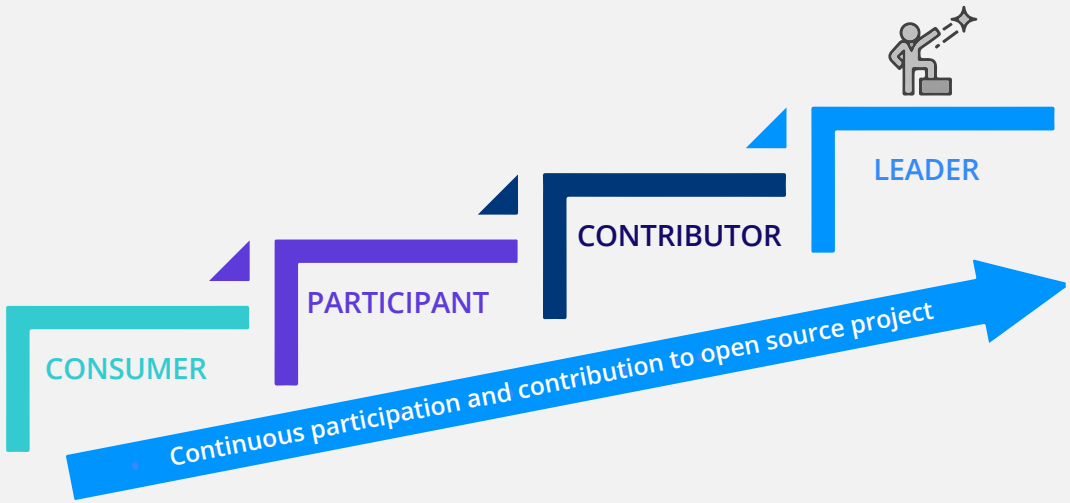
How can an open source strategy help you grab otherwise unattainable opportunities?

Open source also offers unique opportunities that an enterprise can obtain only through an open source strategy. Common objectives under this category include:

- Achieving market leadership by focusing R&D investment on improving critical open source technologies complementary to differentiated product capabilities.
- Defending existing market positions by supporting key open source initiatives and consortia.
- Releasing select proprietary capabilities as open source to disrupt competitors or competing markets.
- Using open source to level the technology playing field.
- Incorporating readily available open source commoditized capabilities and market accelerators in products to reduce the cost of goods sold and the price per product delivered over time.

FIGURE 5

Stages of an Enterprise Open Source Involvement




## Identify Your Stage of Involvement

Organizations move through four primary OSS strategies: consumption of OSS, participation, contribution, and leadership. Each strategy requires an enterprise to succeed at the previous strategy, and how far an organization advances depends entirely on the enterprise. **FIGURE 5** illustrates these four primary strategies. Notice that these strategies overlap as an enterprise transitions from one stage to another.

Engineering drives the early strategies of consumption and participation. Engineers use various open source components for their technical merits to speed up development, but they participate little in projects that maintain these components. Over time, higher levels of the organization learn about the value of this OSS usage. As OSS gains traction, business needs begin to drive such OSS involvement, and OSS efforts contribute to a determined business strategy.

Some companies achieve their goals as consumers. Other companies see strategic advantages in other stages of involvement. They sometimes set up an OSPO to oversee strategic planning and execution through these stages. This section helps to determine where your enterprise is, where you want to be, and how to get there.

### CONSUMPTION

The common starting point for organizations is adopting OSS. Voraciously consuming open source components will increase your ability to differentiate products and services and reduce your overall time and costs in delivering those products and services. Here are the action items essential to this strategy:

- Set up an open source review board (OSRB) to serve as a clearinghouse for all open source activities, including license compliance.
- Use a strategic classification scheme to guide decisions on what OSS to consume.
- Inventory the licenses of the OSS in use to determine whether the enterprise is complying with all license obligations.

- Deploy automated workflow software for evaluating/approving open source usage.
- Create a plan for incremental investment in headcount and infrastructure in engineering, product management, and legal to manage a complex mix of closed source and open source software.

The highest form of open source strategy is leadership. Open source leaders earn their strategic positions by establishing trust with project members and maintaining a high level of continuous contribution. Leading organizations can capitalize on emerging trends in technology.

## PARTICIPATION

Once your organization is successfully using OSS in products or services, you can expand your strategy to participate in the open source community. Unless you have already hired experienced developers, you may need to engage more closely with the community, increase your visibility, and begin attracting the talent you need. Here are the action items essential to participation:

- Monitor community communication platforms like chat servers, mailing lists, forums, and websites to keep on top of project developments.
- Attend relevant conferences and meetups to establish relationships within the community.
- Sponsor project events and foundations to improve enterprise visibility.
- Educate developers on how to participate in and contribute to open source projects.

## CONTRIBUTION

Once your enterprise is realizing the benefits of participating regularly in the community, you can assess the advantages of contributing code to projects and communities. Code contributors help to shape future features, and so contributing source code to those open source projects critical to your business objectives is the best way to influence those projects and build a positive reputation. Here are the action items essential to this strategy:

- Hire a staff director to lead open source strategy and manage the OSPO.
- Hire contributors and committers to open source communities vital to your products and services.
- Deploy open source collaboration tools to support open source usage and contributions.
- Add open source developer resources.
- Invest incrementally in engineering, product management, and legal resources to engage with external communities.

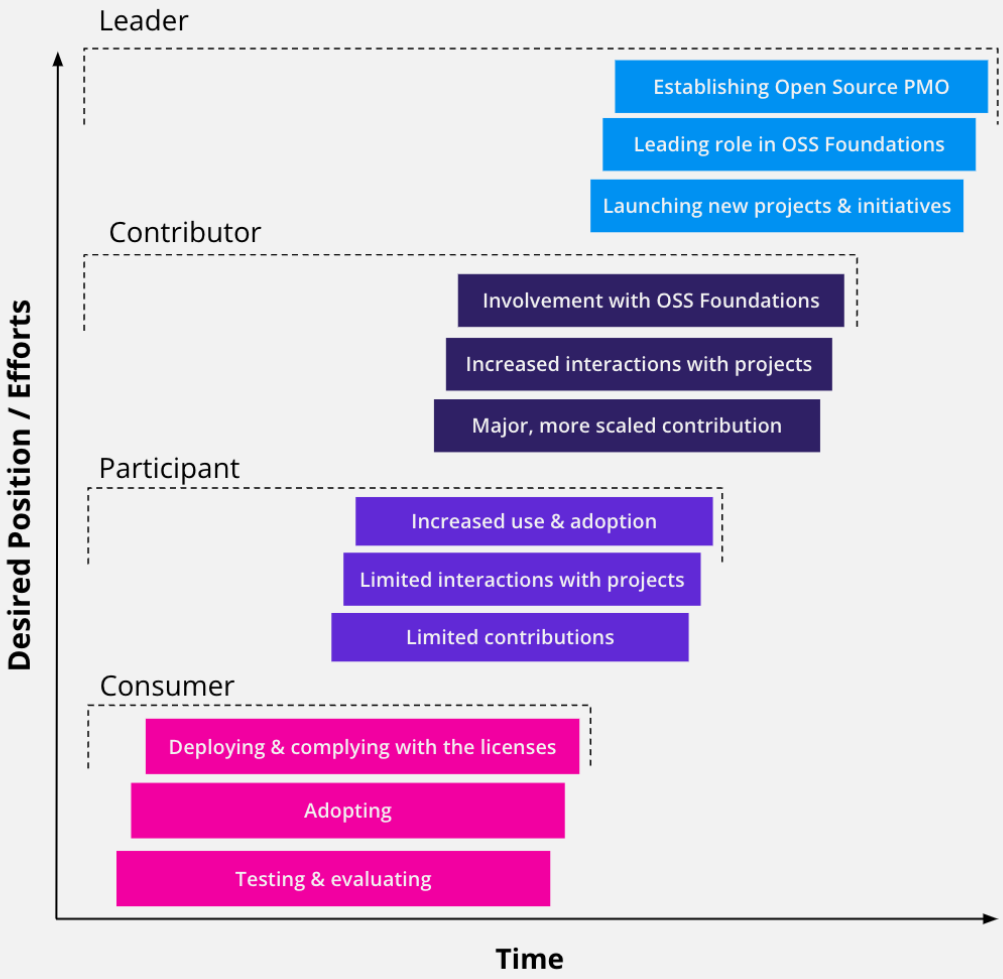
## LEADERSHIP

The highest form of open source strategy is leadership. Open source leaders earn their strategic positions by establishing trust with project members and maintaining a high level of continuous contribution. Leading organizations can capitalize on emerging trends in technology. To establish the leadership agenda, this strategy requires significant investment in targeted open source communities and consortia. Here are the action items essential to maintain leadership:

- Invest incrementally in engineering, product management, and legal resources to external communities and industry consortia.
- Increase engagement with targeted open source communities.

**FIGURE 6**

## Advancing Your Strategy by Increasing Efforts




- Engage selectively with open standards to drive the organization's needs and requirements.
- Engage with open source foundations.
- Establish an open source project, organization, or foundation.

As a technology-driven organization, your enterprise is already evaluating, using, and deploying OSS. You are likely participating and maybe contributing to projects. Ideally, your open source team guides these efforts, decreases risks, and leverages your participation to benefit your strategy.

### TRANSITIONING

As a technology-driven organization, your enterprise is already evaluating, using, and deploying OSS. You are likely participating and maybe contributing to projects. Ideally, your open source team guides these efforts, decreases risks, and leverages your participation to benefit your strategy. **FIGURE 6** illustrates the stages of involvement. Whereas the earlier stages take place with or without guidance, achieving success at the leadership stage can only take place with an open source program in place.

**FIGURE 6** also 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.

# Setting Up Your Infrastructure for Implementation

Once you have developed your organization's open source strategy, you need to build infrastructure to support your open source engineering efforts. **FIGURE 7** shows the four pillars of this infrastructure: community engagement, open source contribution, open source compliance, and open source usage.

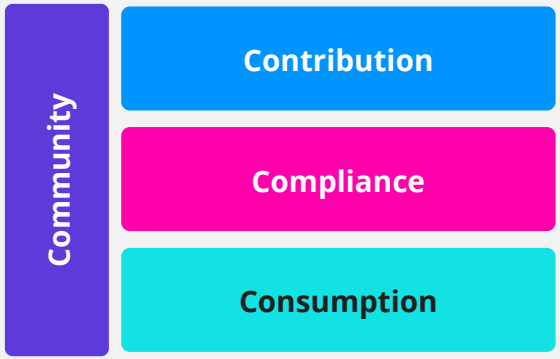
The infrastructure supports all interactions—consuming, complying with, and contributing code—within the community between the organization and its open source projects.

## Start with Consumption and Compliance

Consumption and compliance are closely related, especially in an enterprise setting, where consumption usually means using open source in a product or part of a service. Therefore, we often represent these core elements together, at least from an infrastructure perspective, as any commercial use of OSS requires compliance with the corresponding licenses. **FIGURE 8** itemizes the components.

**FIGURE 7**

Core Elements in an Enterprise Open Source Infrastructure




1. **Frame the strategy.** Open source compliance should include a framework for legal compliance and risk tolerance strategies, mergers and acquisitions (M&A) and corporate development, software procurement, and managing compliance inquiries.
2. **Define the processes.** The OSPO defines the processes and policies for how your organization will handle code distribution, audits, notices, and usage. Additionally, the program office will publish internal policies, implement processes to enforce adherence with compliance guidelines, and provide training on open source licenses.
3. **Communicate the plan.** We recommend setting up two websites to support your open source program—one for internal communications and another for external ones. Organizations publish their open source policies and processes on their intranet, where they can explain open source compliance and guide their staff to available services. Organizations also use their public-facing website to publish information on their open source projects, share announcements, and potentially post compliance reports, notices, and source code.
4. **Establish the teams.** Establish teams dedicated to compliance. Train them on their role and responsibilities. Agree upon their success metrics.
5. **Assist other stakeholders.** Educate everyone who deals with software procurement, development, distribution, and hiring on open source compliance. Establish a process whereby staff can raise questions and get answers.
6. **Get the tools.** Consider investing in software composition analysis tools that automate the discovery and auditing process. Such tools will assist the compliance team and other stakeholders.

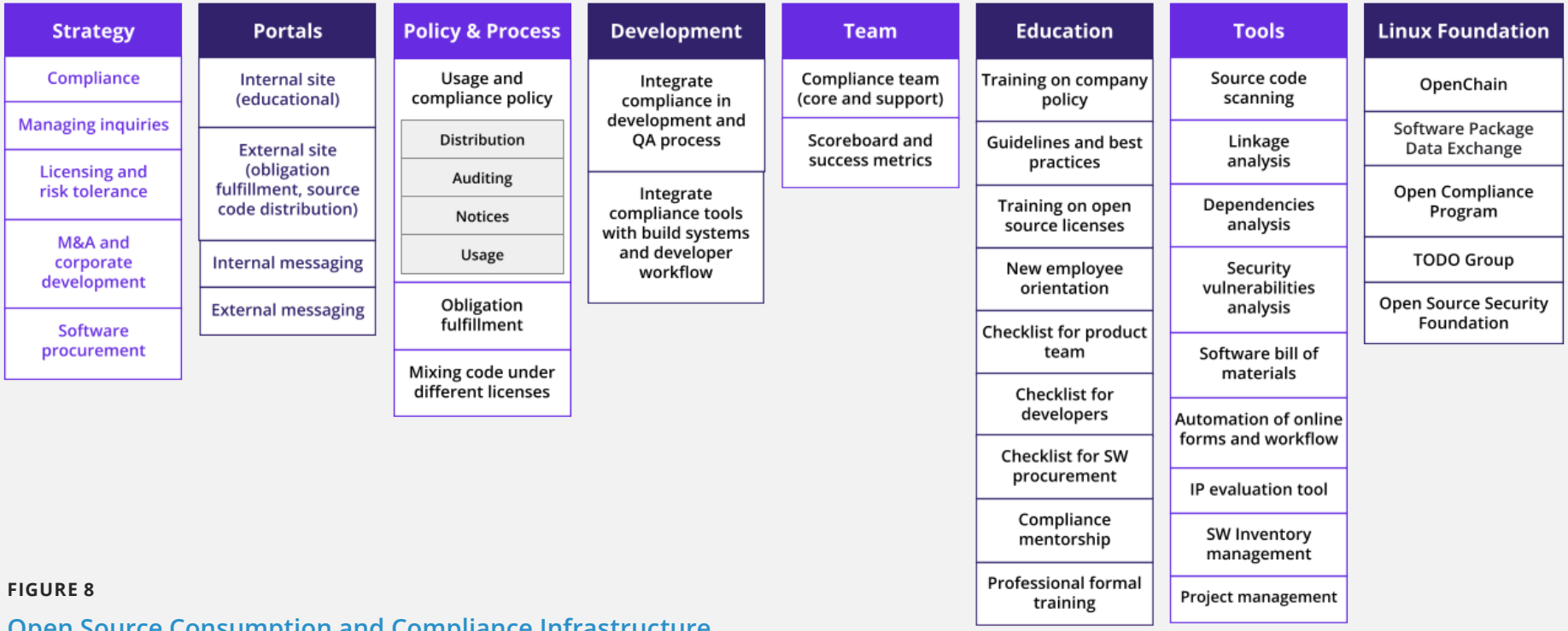



**FIGURE 8**

Open Source Consumption and Compliance Infrastructure

7. **Integrate compliance 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.

8. **Engage the community**. Certain foundations focus specifically on open source procurement, distribution, and compliance. For instance, the Linux Foundation hosts several initiatives that support and enable open source compliance activities. Developing a plan for participating in such foundations can benefit overall enterprise business objectives. Please refer to the Linux Foundation Resources section for links to these initiatives.

## Implement Core Compliance Practices

Every organization must have policies, guidelines, and processes for OSS. Therefore, some of these suggestions may not precisely fit your current compliance framework, but you can still use them as ideas for improvements.

1. **Identify all source code**. Software enters an organization via three primary sources: source code introduced by an organization's developers, source code coming via third-party commercial software providers, and open source channels such as GitHub. We recommend that all source code integrated into products or services go through the compliance process. Through the process, the organization

will identify the origin and license of the source code and set up a plan to meet the obligations of all applicable licenses.

2. **Establish a recurring scanning model.** Organizations adopt various methods to manage approvals to use open source. Given the massive adoption of OSS, these approvals have tended to be as efficient as possible. Some organizations require developers to request formal authorization to use open source code and only integrate it with the product repo once they receive the approval. In some cases, developers, in the rush of getting work done, may not issue a compliance ticket to request authorization to use the needed open source code. Therefore, it is critical to have an additional checkpoint or a method that captures unapproved open source code entering the software stack. You can address this situation by running regular full scans for the entire software stack every x weeks and identifying components that don't have a corresponding compliance ticket. A new compliance ticket will be created per component and pushed via the normal compliance verification process.
3. **Verify compliance case by case.** Approving the use of OSS in one case does not necessarily serve for all cases. Compliance issues may arise depending on the context of using a specific component and how it interacts with other components licensed under different licenses (open source or proprietary). We recommended that each time developers modify a previously approved open source component or plan to use a previously approved component, a new scan is run. The result will be a new bill of material, a confirmation of licenses, and an approval to use the open source code in the current context.
4. **Verify licenses when upgrading versions of open source components.** License changes in open source components may occur between major version upgrades. When developers upgrade open source components, we recommend verifying the license. If there is a change in the license, a new compliance ticket may need to be created requesting approval to use the latest version of the open source component licensed now under a different license.
5. **Resolve compliance issues.** When a source code scanner examines the code base, it may flag possible issues according to the rules and policies configured in the SCA tool. Once a compliance engineer has confirmed the accuracy of these flags, we recommend working with developers to resolve the issue, running a scan of the resolution, and generating a new bill of material from the updated source code.
6. **Save all licensing information.** We recommend saving all related licensing information such as COPYING, README, or LICENSE files in the repository in preparation for a legal review. Many organizations require developers to attach such files to the compliance ticket of the specific open source component so that ticket reviewers have all the information they need to evaluate the compliance status.
7. **Maintain a record of discussion.** Following the previous practice (saving licensing information), we also recommend maintaining a summary of discussions in the compliance ticket that lead to the approval or rejection of a specific open source component. Such documentation can prove very useful when attempting to determine the basis on which approval for a particular software component was granted and how possible issues were resolved.
8. **Provide a written offer.** Specific open source licenses require that projects notify recipients of the software of various license information and provide access to the source code. We recommend using clear language and including all OSS used in the product or service. Organizations often put the written offer and open source licensing information on the product itself, in the product documentation, and on a website. The location varies depending on the product or service.
9. **Manage modifications to open source software.** We recommend capturing all modifications to open source code in the revision history (aka *changelog file*). Mark changes to the

source code as such when redistributing modified code, per the license in effect. Some companies elect to provide the original open source code along with the company's contributed patch file so that the company's modifications are identifiable and separate from the original open source package.

10. **Keep track of retired components.** In some instances, developers may stop using a previously approved open source component. We recommend that developers re-open the corresponding compliance ticket and update it to reflect that they have retired the component with details about the build number and specific use information. This action will trigger an update of the open source license information given to the product or service users.

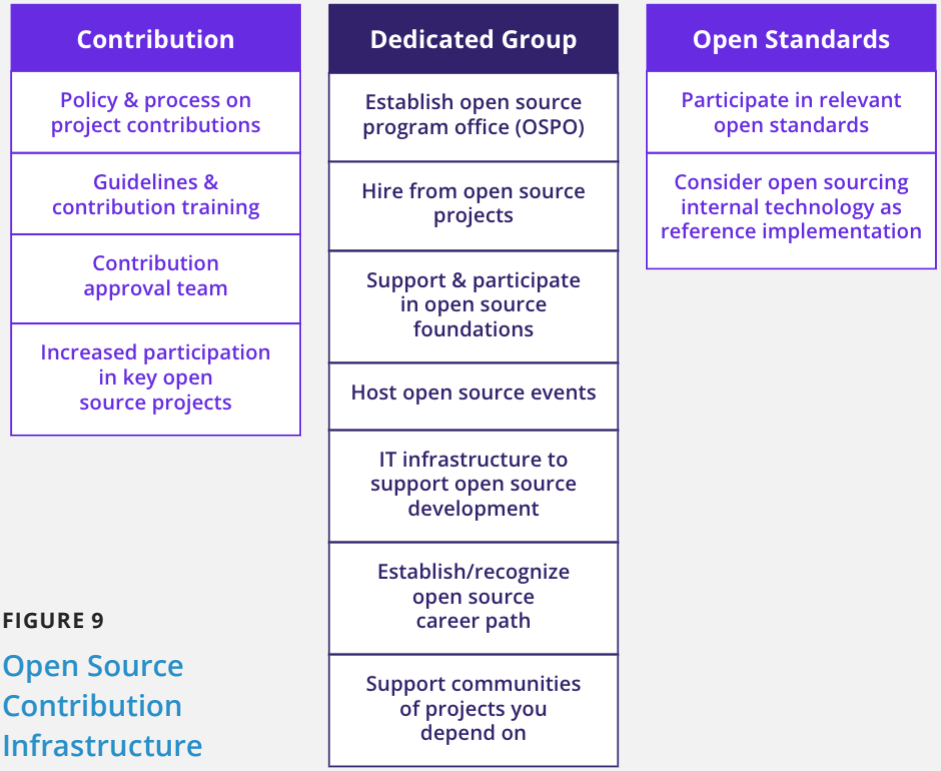



**FIGURE 9**

Open Source Contribution Infrastructure

11. **Avoid copy/paste.** Developers must avoid copying and pasting open source code into proprietary or third-party source code (or vice versa) without prior documented approval. Such actions can have severe implications for license compliance.

12. **Abstain from mixing source code with different licenses.** Some open source licenses are incompatible with others, and some are incompatible with proprietary licenses. Therefore, we recommend seeking guidance from your legal counsel and receiving approval to mix source code published under different licenses *prior to mixing such code.*

13. **Preserve original license information.** We highly recommend preserving existing copyrights, attribution, and licensing information in any open source component.

14. **Update outsourcing agreements.** Revise software outsourcing agreements to reflect adherence with open source compliance practices and mandate all incoming source code to go through the compliance process.

## Provide Structure for Contributions

When you expand your efforts into contributing to open source projects, you need infrastructure and a framework to support these efforts, enforce your contribution policy, and discourage ad hoc contribution practices. An open source contribution infrastructure has three elements **(FIGURE 9)**.

- **Support for contribution.** Contribution support is critical and includes a contribution policy and process; a guideline for contributing to open source projects; training for development teams so that they understand the contribution policy and process; a group responsible for approving requests to contribute; and a clear plan for contributing to open source projects that support the product development.

- **Dedicated open source group.** A group dedicated to upstream contributions can be valuable. The team, sometimes called OSPO, would support ongoing collaborations with open source projects and foundations and help build the organization's open source consumption, compliance, and compliance infrastructure.
- **Participation in open standards**. Participating in relevant open standards bodies and trade organizations is critical for the open source contribution infrastructure. Assign people responsibility for keeping the organization up to date on changes to relevant standards.

## Update M&A Practices

Update policies and guidelines regarding M&A or other corporate transactions to account for OSS. If your organization is considering a merger or acquisition, then it should structure its compliance program to offer the necessary level of disclosure and representations. Corporate development must mandate that the compliance team evaluate source code before any merger or acquisition to avoid surprises that might derail discussions or affect the estimated valuation. Comprehensive code evaluation assures accurate valuation of software assets for the acquiring company and mitigates the risk of unanticipated licensing issues undermining future value. Organizations must also update their due diligence practices requiring the disclosure of open source assets and provide guidance to their staff on open source licenses. In the purchase agreement, an acquiring company may also include provisions that require open source disclosure subject to the transaction.

## Participate in Compliance Initiatives

The Linux Foundation hosts several compliance initiatives that aim to improve compliance with free and OSS licenses. OpenChain and the Software Package Data eXchange (SPDX) are two significant Linux Foundation Open Compliance Program projects. We highly recommend organizations participate in these initiatives to support their open source compliance efforts.

### OPENCHAIN

The OpenChain Project is a standard for a quality open source compliance program. The project builds trust in open source by making open source license compliance more straightforward and more consistent. OpenChain consists of three core elements:

1. The OpenChain Specification defines a core set of requirements every quality compliance program must satisfy.
2. The OpenChain Curriculum provides the educational foundation for open source processes and solutions and meets a vital requirement of the OpenChain Specification.
3. The OpenChain Conformance allows organizations to display their adherence to these requirements. As a result, open source license compliance becomes more predictable, understandable, and efficient for participants of the software supply chain.

Most importantly, OpenChain is a growing community of people who are excellent resources for organizations starting their journey in open source compliance.

### SOFTWARE PACKAGE DATA EXCHANGE

The software ecosystem has become ever more complex and interconnected. A company cannot easily get an accurate and up-to-date view of its software assets and licenses. This complexity also hinders companies in identifying and seizing opportunities to reduce their software costs. Software Package Data eXchange (SPDX) addresses this complexity.

SPDX is a standardized format for capturing, storing, and sharing data about software packages, such as OSS packages. A community-driven project hosted at the Linux Foundation, SPDX helps software developers and engineers capture and share vital metadata about software packages, licenses, and dependencies.

SPDX also helps facilitate compliance with free and OSS licenses by standardizing how license information is shared across the software supply chain. It reduces redundancy by providing a standard format for companies and communities to share vital data about software licenses and copyrights, thereby streamlining and improving compliance.

# Setting Up Your Talent for Success

## Name a Leader for the Team

Hire or promote someone with a deep understanding of the methodology behind open source development and leadership traits to drive your open source effort. This individual should possess several traits:

- A strong engineering background.
- Contacts in open source organizations.
- A solid understanding of open source licenses.
- Knowledge of industry best practices.
- Knowledge and experience in establishing corporate-wide policies and processes.
- Technical knowledge related to the organization's products and services.
- Historical perspective on open source.
- Knowledge of how various technical projects communities operate.

The TODO Group has published a template job specification for this role that you can customize to your needs: https://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 that people hired as open source developers see their career potential within the organization rather than elsewhere. Organizations should also adjust performance-based bonuses to include open source development work goals. The performance metrics for proprietary/closed source developers often differ from those for open source developers.

In reality, all modern developers must work with open source; there are no closed source developers. Instead, sometimes their code stays inside the organization, and sometimes they publish it, perhaps by contributing it to a third party or posting it as a new project. The HR career track and incentives should reflect each organization's structure and approach to open source.

Finally, allow a work-from-home policy for open source developers, regardless of the general corporate policy. Lately, we have witnessed a reversal in work-from-home policies across companies; many have even banned or strictly limited 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.

## Offer Open Source Training

Education is an essential building block in an open source program office. It falls into two categories: technical training to expand open source technical knowledge and compliance training to ensure that employees understand policies governing the use of OSS.

This training aims to raise awareness of open source policies and strategies and build a shared understanding around the issues and facts of open source licensing and the business and legal risks of incorporating OSS in products and software portfolios. Training also serves as a venue to publicize and promote compliance policies and processes and foster a compliance culture.

No organization can 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 increase the competence of your developers in a given technical domain and educate them on the open source development

model and the basic concepts of open source legal compliance. Sample training courses include:

- Technical training on core open source technologies.
- Open source development methodology.
- Open source compliance with licenses.

Organizations can avail themselves of other resources such as the OpenChain curriculum and the Linux Foundation free "Compliance Basics for Developers" training course.

Internal collaborations are challenging, especially in large companies. Open source aligns employee incentives with the enormous benefits of upstream partnerships; teams collaborate to realize those benefits.

## Encourage Internal Collaborations

Fostering internal collaboration among business units that use the same open source projects in their products can increase the impact of an open source engineering team. These collaborations can take several forms, such as:

- Delivering training to product developers.
- Running workshops on specific topics or problems.
- Developing new functionalities.
- Troubleshooting and resolving issues and bugs.
- Upstreaming existing code for which the enterprise has no resources to support.
- Helping the enterprise to transition from an old fork to a mainline version.

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

- They increase visibility of the open source team in project communities or in groups within your organization.
- They cultivate internal expertise on open source, preparing the enterprise to become an upstream partner to R&D and product teams.

Internal collaborations are challenging, especially in large companies. Open source aligns employee incentives with the enormous benefits of upstream partnerships; teams collaborate to realize those benefits.

## Provide Flexible IT Services

Developers use three primary domains of IT services in open source development. Your open source developers should have access to these internally:

- Knowledge sharing (wikis, collaborative editing platforms, and public websites).
- Communication and problem solving (mailing lists, forums, Slack and similar real-time chat).
- Code development and distribution (code repositories and bug tracking platforms).

Set up flexible IT services so that open source developers may communicate and collaborate on open source projects with minimal friction. These services typically require an IT infrastructure free of limiting IT policies, and so this suggestion might conflict with existing IT policies. Resolve these conflicts and allow open source developers to use the tools they need.

Make sure your enterprise tools match the tools used externally. Much of your service infrastructure will evolve organically with your organization's open source culture, but if you understand its necessity, then you can plan for it and guide its implementation.

## Track Progress via Meaningful Metrics

Once an organization starts implementing open source best practices, it will need proper metrics to drive the desired development behavior. However, the traditional metrics of product organizations do not transfer and apply in open source development. For example, you might track the number of changesets or accepted lines of code to gauge the impact of your open source development impact, but you would miss have multiple instances of desired functionality implemented upstream because your open source developers lobbied effectively for support from the community. In this case, the team members' technical leadership upstream reduced the organization's downstream maintenance efforts. Therefore, the metrics you track should account for both types of activities.

To start, create an internal system that tracks developer contributions and impact. Metrics can include upstream development, support to product teams, knowledge transfer (i.e., mentoring, training, and sharing through wikis, mailing lists, forums, Slack, etc.), visibility (i.e., publications, talks), launch of new open source projects and internal collaborations with other teams or groups. Here are some of the primary metrics to consider.

1. **Number of patches submitted and committed.** The number of patches submitted and committed is the most basic metric to track. It should give you an idea of the activity in a project. It includes information about who authored the code, its submitted project, and the date of commit acceptance. Combine this metric with the other metrics below or identify data worth further qualitative analysis.

2. **Type of patch.** We can classify the types of patches, or code submission, into six categories:
   - Bug fixes.
   - Improvements to existing features.
   - Implementation of new minor features.
   - Implementation of new significant features.
   - Contribution to test cases and test code.
   - Contribution to documentation.

3. **Patch acceptance rate.** Just because you write code for an open source project does not mean the community will accept it during the peer review process. Sometimes, you must revise it many times over an extended period. The ratio of patches committed to a project versus the number of patches accepted to the code base is a valuable metric to track how successfully your engineers are getting code accepted the first time.

4. **Patches committed as part of collaborative projects**. If your open source engineers submit code directly on behalf of other teams—collaborations with other internal groups or external organizations such as universities—then you might want to track the impact of these submissions on upstream work.

5. **Visibility.** Through open source engineering, your enterprise can influence a project's direction within the community. Developers earn their peers' respect by continuously making quality contributions to a project and its open source community. To measure developer visibility, consider tracking the number of publications, blog posts, conference presentations, and media mentions, for instance, and encourage developers to spend time improving their open source leadership.

6. **New projects and initiatives.** If one of your enterprise goals is to create new projects that fill unmet needs, then you can track the number of new projects launched and incentivize teams to create them.

7. **Contributions to open standards.** Sometimes, an open source project provides a reference implementation to an open standard. In such cases, you can track employee contributions to the open standard.

## Apply Open Source Methods to Internal Projects

Inner sourcing is the application of open source methodologies to development projects inside the organization. The goal is to incubate the same capabilities within the enterprise as those incubated within the open source community, and to foster new employee-to-employee relationships that are cross-functional and touch on multiple product domains. Open source principles work well on large-scale projects distributed across an enterprise. Many Fortune 500 companies have adopted them externally and internally for the same reasons: faster releases, improved quality, increased innovation, increased communications and information sharing, reduced costs, increased and more effective collaboration, and increased employee morale and retention.

Inner sourcing prepares your organization to work effectively with external open source communities. It encourages your employees to interact with colleagues elsewhere and with external community members without switching contexts. New employees familiar with this development model may integrate more quickly into your workflows. Finally, your business partners are probably already using many of these development practices, and so your adoption can strengthen your integration with the commercial ecosystem.

### INNER SOURCING IN PRACTICE

Inner sourcing means opening source code bases to all employees and allowing anyone to use them, contribute to them, or fork them as needed. Establish a team to steward the code and review and apply code submissions. Companies using inner source methods use the same mailing lists, chat rooms, and ticketing systems that are fully open to anyone at the organization for discussing changes and asking questions. Please check the references for more information on implementing inner source practices.

### ESSENTIAL OPEN SOURCE PRINCIPLES

There are several essential principles to implementing proper inner source practices. This section covers each of these.

1. **Visibility**. All internal software projects should be visible to all employees by default, and communication channels should be as open as possible. This visibility 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 and individuals. However, these should be the exception rather than the rule.
2. **Forking.** Allow anyone who can see internal code to create a copy (fork) and make changes freely, as long as they make these forks visible to everyone. This approach encourages greater understanding and better integration of the software stack across the various teams within the organization and cross-pollinates ideas.
3. **Pull/merge requests.** Allow people outside the project to suggest and submit changes. Encourage employees to welcome input from teams outside their own and take their contributions seriously.
4. **Peer review.** Peer review reduces variations in style, prompts useful conversations, and preserves the project's quality standards. People with a strong understanding of the code base should always review code before submitters commit it. These peer reviewers should also provide feedback to submitters so that the latter refine their style over time and increase the likelihood of acceptance.
5. **Release early and often.** More frequent releases with fewer features and regular update releases are essential

for improving code development efficiency. The limited scope makes bugs and regressions easier to identify and fix. Create a plan for a release hierarchy such as nightly, weekly, milestone, long term, and so forth.

6. **Testing.** Incorporate unit and integration tests directly into the software development process to make changes without fear of creating new problems. Identify and address potential issues early to avoid accumulating technical debt.
7. **Continuous integration.** Software development processes should implement continuous integration to test every proposed change automatically.
8. **Documentation.** All software projects should include documentation describing what the software does, why it matters, how to run it, and how to participate in its development process.
9. **Issue tracker**. Everyone should be able to submit a bug, request a feature, or ask questions. Issue trackers for all internal projects should be open to these submissions. This openness increases the amount of feedback they receive. It also expands the available expertise to other teams.

## Raise Developer Visibility

### HOST AND ATTEND OPEN SOURCE EVENTS

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

If your developers have work that others might be interested in, then help these developers prepare presentations. Consider sponsoring big and small events to increase visibility within the project's community. These events are also great venues for finding talent!

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

Also check out Linux Foundation events and identify the events related to your interests. Encourage your developers to expand or share their knowledge or send management representatives to identify prospective hires and recruit new talent.

### COLLABORATE WITH UNIVERSITIES ON R&D PROJECTS

Many universities and schools are eager to work with companies that offer learning opportunities for their students because it provides the chance 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. Organizations can't hire all the intelligent people globally, so they will need to find a way to tap into new knowledge and influence favorable outcomes in external projects.

# Challenges

Open source software has become the default choice of many organizations, with the software powering some of the most high-profile websites and services on the Internet. While the concept of OSS is simple enough, how organizations use and adopt it has evolved significantly over the years. While some organizations still use OSS without knowing it, others have made significant effort to adopt OSS fully, as they would adopt proprietary software. In some cases, they have moved away from the proprietary software that their organizations had used for years and toward OSS that they can modify and customize. Despite the advantages, organizations often struggle to adopt OSS and move forward with their digital transformation.

There are generally five areas where organizations typically face significant challenges: culture, processes, tools, continuity, and education (**FIGURE 10**). Within each of these challenges, organizations must address several issues to support their adoption of the open source methodology and practices.

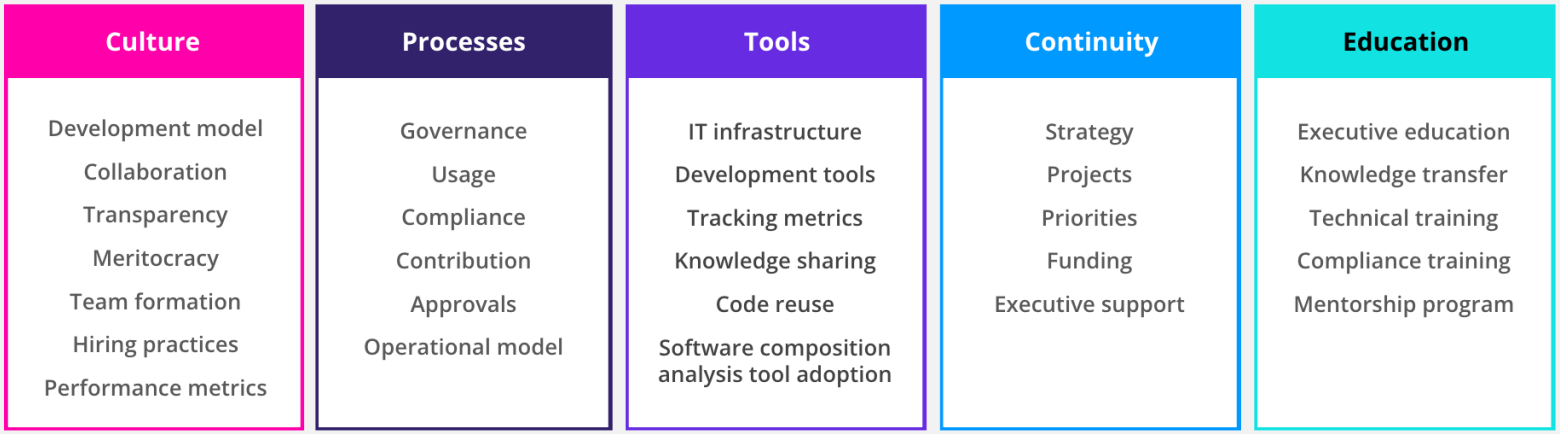
## Culture

Cultural challenges often stem from the gap between traditional software development practices and the requirements of open source development. To bridge this gap, hire open source experts and ask them to train other groups unfamiliar 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 engineering teams around the ideals of meritocracy.
- Establish proper success metrics to encourage open source and cross-department contributions.

**FIGURE 10**

The Five Challenge Areas for Open Source Engineering



## Processes

Open source development is dynamic, moves very quickly, and has unique requirements for compliance. Software-driven industries will leave behind those companies that do not adapt their internal processes to support this type of development. Developers must be able to contribute code upstream quickly, and so the enterprise must modify any internal code policies that hinder such development.

- Put a team in charge of maintaining open source compliance to avoid legal problems and set up a simple internal approval model for open source use and contributions.
- Move from highly complex and cumbersome policies to a more straightforward approach for receiving, reviewing, and approving source code contributions.
- Balance the interests of legal, engineering, and open source, but give the dedicated open source team blanket approval to contribute to many open source projects.
- Use different levels of approval depending on the nature of the code contributed (e.g., code to fix simple bugs, code to improve existing functionality, code to effect new functionality, or code to seed a new project).

## Tools

The IT environment you create should allow developers to join a team without requiring any significant changes to how they work. The tools must support the open source development model, fulfill the needs of the OSPO, and meet corporate IT guidelines. Open source engineers need greater flexibility to communicate with external participants via email, chat, and code development platforms, and their IT tools must facilitate this communication. For example, emails to an open source project should never include attachments that claim the content as IP of the email sender's company.

Allow communication with public mailing lists from company accounts without obstruction. Give engineers devices that support the development distribution of their choice. Make sure all open source developers can access all vital internal tools and resources on Linux or via a separate compatible device. Support fully distributed teams working in remote locations so that they can connect to internal business resources through a virtual private network or similar technology. Evaluate your IT policies for help desk support, with secure methods for resolving IT issues for remote employees.

## Continuity

For some organizations, continuity brings to mind a long, boring document that nobody reads. When it comes to OSS, continuity is an ongoing challenge as the organization adapts to changes in its business, its business strategy, and its industry. In practical terms, we can break continuity into three categories:

- **Continuity of the open source strategy:** Informing current and future employees of the ever-evolving open source strategy, with updates on new developments and changes as they happen.
- **Continuity of projects and priorities:** Ensuring continued involvement in open source projects and initiatives to make use of any momentum that preceded a period of disruption or changes in the organizational environment.
- **Continuity of executive support and funding:** Ensuring continued financial and executive support and ensuring adequate resources to support the open source program. Executive sponsor is critical to continuity, communicating the value of the open source efforts and expectations across the organization to encourage successful adoption, implementation, and contribution to open source projects.

## Education

Open source software is an integral part of the software landscape, with significant benefits for users and the ecosystem. But to realize these benefits, organizations must overcome knowledge deficits through education and training:

- **Executive training:** These courses help executives and managers understand and articulate the basic concepts for building effective open source practices. Such courses often cover techniques for building effective processes and strategies for consuming OSS, creating new open source projects, contributing to projects, and driving software leadership in the open ecosystem.
- **Compliance training:** With the adoption of OSS comes the responsibility to respect and fulfill the IP obligations of applicable open source licenses. To that end, organizations provide employee training on the basics of OSS, open source licenses, how copyright works, and the organization's policies and processes.
- **Mentorship program:** To increase open source knowledge and technical skills, organizations set up mentoring programs where a senior open source developer guides a junior developer in a structured, often one-to-one relationship. The goal is to transfer knowledge and train mentees on how to work effectively with open source projects and, in the process, increase their technical competencies in their specific domain.
- **Technical training:** Technical training expands the technical knowledge base of staff. It addresses weaknesses and upskills employees to do new and different tasks. The open source training industry is thriving because of the high demand for open source skills and training on the latest open source technologies.

# Closing Remarks

It is no secret that "open source is eating the software world."[2] Mastery of open source requires a strategy that encompasses open source consumption, participation, contribution, and leadership. Each of these activities requires an incremental effort and investment in 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 events. Sponsor projects and organizations are essential to OSS you rely on for your products.
- **Contribution:** Hire or train developers to focus 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 open source foundations. Launch new open source initiatives and increase your visibility in open source communities.

The path to open source mastery is no secret. If you follow the suggestions throughout this guide, then you should find yourself well on your way.

Welcome to enterprise open source.

2 Michael J. Skok, quoted by Adrian Bridgwater, "Open Source Is Eating the Software World," *ComputerWeekly.com*, TechTarget Inc., 18 April 2013.

# Linux Foundation Resources

## Publications

- Open Source Compliance in the Enterprise, 2nd Edition
- Open Source Audits in Merger and Acquisition Transactions
- Technical Debt and Open Source Development: A Discussion Toward a Better Understanding of Technical Debt and How Open Source Development Helps Mitigate It
- Practical GPL Compliance
- Assessment of Open Source Practices as Part of M&A Transaction Due Diligence
- Improving Open Source Development Impact
- Using Open Source
- Is the Open Source Development Model Right for Your Organization?
- Establishing an Open Source Software Strategy: Key Considerations and Tactical Recommendations
- Open Sourcing Proprietary Technology Made Simple: A High-Level Road Map for Open Sourcing Your Code
- Upstreaming: Strengthening Open Source Development

## Programs

- Open Compliance Program
- TODO Group

## Projects

- OpenChain
- Software Package Data eXchange

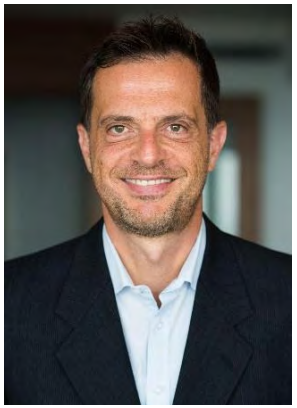
## Guides

- Linux Foundation Enterprise Guides
- Recruiting Open Source Developers
- OpenChain Curriculum

## Free Training

- Linux Foundation Free Compliance Training For Developers

# About the Author



Dr. Ibrahim Haddad is Vice President of Strategic Programs at the Linux Foundation, where he focuses on facilitating a vendor-neutral environment for advancing the open source platform and empowering generations of open source innovators. Haddad leads the LF AI & Data Foundation, providing a trusted hub for developers to code, manage, and scale open-source AI and data projects. His work, and the work of the Foundation as a whole, supports companies, developers, and the open source community in identifying and contributing to the technical projects that address industry challenges for the benefit of all participants. Before the Linux Foundation, he served as Vice President of R&D and Head of the Open Source Division at Samsung Electronics. Throughout his career, Haddad held technology and portfolio management roles at Ericsson Research, the Open Source Development Labs, Motorola, Palm, Hewlett-Packard, and the Linux Foundation. He graduated with Honors from Concordia University (Montréal, Canada) with a Ph.D. in Computer Science.

# Disclaimer

This report is provided "as is." The Linux Foundation and its authors, contributors, and sponsors expressly disclaim any warranties (express, implied, or otherwise), including implied warranties of merchantability, non-infringement, fitness for a particular purpose, or title, related to this report. In no event will the Linux Foundation and its authors, contributors, and sponsors be liable to any other party for lost profits or any form of indirect, special, incidental, or consequential damages of any character from any causes of action of any kind with respect to this report, whether based on breach of contract, tort (including negligence), or otherwise, and whether or not they have been advised of the possibility of such damage. Sponsorship of the creation of this report does not constitute an endorsement of its findings by any of its sponsors.

twitter.com/linuxfoundation

facebook.com/TheLinuxFoundation

linkedin.com/company/the-linux-foundation

youtube.com/user/TheLinuxFoundation

LF AI & DATA

**Part of the Linux Foundation, LF AI & Data supports open source innovation in artificial intelligence, machine learning, deep learning, and data. LF AI & Data was established to support a sustainable open source AI ecosystem that makes it easy to create AI and Data products and services using open source technologies. We foster collaboration under a neutral environment with an open governance model to support the harmonization and acceleration of open source technical projects.**

THE LINUX FOUNDATION | Research

Founded in 2021, Linux Foundation Research explores the growing scale of open source collaboration, providing insight into emerging technology trends, best practices, and the global impact of open source projects. Through leveraging project databases and networks, and a commitment to best practices in quantitative and qualitative methodologies, Linux Foundation Research is creating the go-to library for open source insights for the benefit of organizations the world over.

May 2022

Copyright © 2022 The Linux Foundation

This report is licensed under the Creative Commons Attribution-NoDerivatives 4.0 International Public License.

To reference the work, please cite as follows: Ibrahim Haddad, "A Guide to Enterprise Open Source," foreword by David Marr, May, 2022.