

# A Road Map to Improve the Effectiveness and Impact of Enterprise Open Source Development

**February 2023**

Ibrahim Haddad, Ph.D.  
Vice President, Strategic Programs (AI & Data)

Foreword by Jessica Murillo  
VP and Delivery Practice Leader, IBM



# Contents

<b>Foreword</b> .....	<b>3</b>	<b>Recommendations and lessons learned</b> .....	<b>12</b>
<b>Introduction</b> .....	<b>5</b>	Be patient.....	12
<b>Hire developers from the project’s community</b> .....	<b>7</b>	Embrace a flexible IT infrastructure .....	12
<b>Support and allocate time for upstream contributions</b> .....	<b>7</b>	Adopt proper success metrics .....	12
<b>Create a mentorship program</b> .....	<b>8</b>	Use a lightweight approval process .....	12
<b>Offer training</b> .....	<b>9</b>	Share information.....	12
<b>Participate in and host open source events</b> .....	<b>9</b>	Make strategic contributions .....	13
<b>Provide a flexible IT infrastructure</b> .....	<b>9</b>	Partner with product teams.....	13
<b>Track developer code contributions</b> .....	<b>10</b>	Grow open source talent.....	14
<b>Identify focus areas with a broad impact</b> .....	<b>10</b>	<b>Conclusion</b> .....	<b>15</b>
<b>Foster internal collaboration</b> .....	<b>10</b>	<b>Acknowledgments</b> .....	<b>17</b>
<b>Implement inner sourcing practices</b> .....	<b>11</b>	<b>Feedback</b> .....	<b>17</b>
		<b>Linux Foundation resources</b> .....	<b>17</b>
		<b>About the author</b> .....	<b>18</b>

# Foreword

“In real open source, you have the right to control your own destiny.”

– LINUS TORVALDS, CREATOR OF THE LINUX KERNEL

A lot has changed in the past 20 years since technology companies, like IBM, began their open source journey. In the first 10 years, enterprises started by contributing to open source projects to help fill their needs; they made strategic investments in technology, collaboration and communities and built an entirely new ecosystem. In the next 10 years, we saw the emergence of hyperscale cloud providers and Fortune 500 companies that shifted from being passive consumers to proactive participants in open source communities. This heightened collaboration spurred even faster innovation.

We have learned that companies who only participate in open source on an ad-hoc basis cannot achieve long-term success. The key is for companies to take a more structured, enterprise approach, putting open source at the core of their technology strategy. To truly benefit from the open source community model, each contributor is responsible for making the necessary investments in those communities. This includes providing open source developers from your company with the proper tooling, training, and mentoring to become strong community contributors and grow into leaders. It means we need to work together to solve not only the problems that scratch our own itch, but by broadening the scope of influence and focusing our

time and talent to improve the code base and remediate issues in open source software if they arise. That is what it means to be a good member of an open source community.

This document provides an overview and step by step guide for companies to engage in open source development, no matter where you are on your journey.

**JESSICA MURILLO**

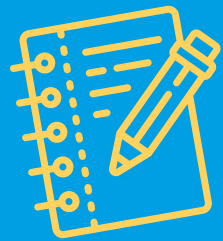
**VP and Delivery Practice Leader, IBM**

Be patient and seek out influential peers when **growing your domain expertise, open source methodology, and working practices.**



Practice and encourage **an open and collaborative mindset** when implementing open source infrastructure.

Adopt IT infrastructure that is **flexible and supportive** of open source development.



Track success through **specifically designed metrics for an open source environment.**



Follow a **lightweight and tailored approach** to source code contribution approvals.

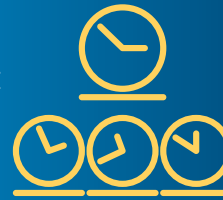


**Share information across divisions and foster internal collaborations** for successful implementation of innersource practices.

**Contribute strategically to projects** that are commonly used across products and services to remain essential, justifiable, and fundable.



Allocate time for open source developers to meet upstream responsibilities, **especially if they are maintainers.**



Partner with product teams on **upstream code development** that helps reduce their technical debt.



Develop open source talent internally, and **encourage involvement** in open source from developers across the organization.

**Create a mentorship program** to support the growth of junior developers and increase the quality and quantity of code accepted in open source projects.



Participate in and host open source events **to build developer networks, participate in technical discussions, and increase visibility.**

## Introduction

Corporate participation in open source has reached an all-time high and continues to grow as organizations realize the value of consuming and contributing to open source projects (**FIGURE 1**). In addition, the nature of corporate (also called enterprise) participation continues to evolve as organizations increasingly discover that open sourcing proprietary technologies can create new sources of value and more robust product ecosystems.

Enterprise open source development has challenges, which we discussed in detail in [“A Deep Dive into Open Source Program Offices: Structure, Roles, Responsibilities, and Challenges.”](#)

The enterprise open source journey is challenging (**FIGURE 2**), and an organization needs to address this to build its open source leadership. If the organization has a clear plan to implement

internal practices and address those known challenges, the journey becomes easier. For instance, the Linux Kernel is the largest collaborative software project in the world, and getting involved in the development process can be overwhelming. If you are one of the organizations that rely on the Linux Kernel for their products and services, investing time and resources into improving your internal development abilities, contributions process, and syncing your development with the upstream project can pay off immensely in the long run.

Fortunately, since so many organizations and individuals have been successful at contributing to the Linux Kernel, there is a clear path to improve your own Linux Kernel contributions and aim for a leadership role.

### FIGURE 1 OPEN SOURCE STRATEGIC IMPACT



- Accelerates the development of open solutions
- Provides an implementation to an open standard



- Commoditizes a market
- Reduces the process of nonstrategic software assets
- Provides an implementation to an open standard
- Shares development costs



- Drives demand by building an ecosystem for products and services



- Partners with others
- Engages customers
- Strengthens relationships with common goals

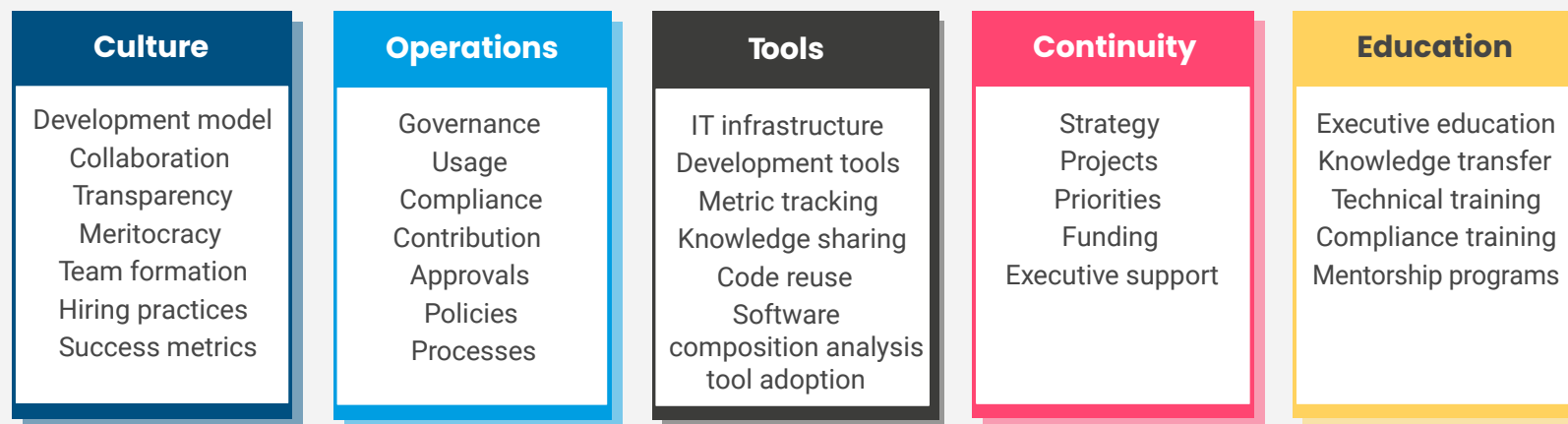
Several factors drive and motivate participation in open source projects:

- Reducing the amount of work needed from product teams
- Minimizing the cost to maintain source code and internal software branches
- Improving code quality
- Supporting faster development cycles
- Producing more stable code to serve as the base for products
- Improving the organization's reputation in critical open source communities

Organizations often upstream modifications to open source projects, which is a fundamental aspect of the open source methodology. Following this approach, enterprise developers submit internal changes to the open source project for evaluation for acceptance into the main development tree. This process achieves several technical and nontechnical benefits for the enterprise due to such contributions (see **FIGURE 3**).

This report covers several practices enterprises can adopt to help grow their footprint in open source projects.

**FIGURE 2**  
**CHALLENGES ENTERPRISES FACE AS PART OF**  
**INSTITUTIONALIZING OPEN SOURCE DEVELOPMENT PRACTICES**



## Hire developers from the project's community

This critical step allows your organization to gain skills and recognition immediately. Hiring two or three people is a great start toward making a noticeable impact on a large project, such as the Linux Kernel, attracting further hires and allowing enough resources to mentor existing junior developers.

It is crucial to align corporate interests with individual interests. It will be hard to motivate a senior open source developer to work on a given project when their interests do not match those of their employers. For instance, a memory management expert may not be interested in working on file systems; therefore, finding a match in interests is critical.

## Support and allocate time for upstream contributions

The core principle for hiring open source developers is to support an organization's open source strategy, development, and upstream activities; however, in most cases, there is the expectation that open source developers will need to be available to support product teams due to their expertise and influence in their respective open source projects. It is also common for product teams to exercise their influence in an attempt to hijack the time of the open source developers by having them work on product development as much as possible. If this happens, many open source developers will head to the door, seeking new opportunities that allows them to work on their upstream project before an organization realizes what just happened.

FIGURE 3

### BENEFITS OF UPSTREAMING CODE



Lower maintenance efforts for internally managed code, i.e., minimizes technical debt.



Upstreamed code becomes visible to others and receives peer review and feedback, leading to improvements.



Upstream contributions provide stability to the project. They send a signal that the project is useful and important, which helps attract new contributors.



Builds a positive relationship between the contributing organization and the project community.



Upstreaming code is an effective way to provide technical leadership and influence the project.



Upstreaming contributes to easier compliance and improved security due to centralizing code in upstream repos.



Upstream contributions are an effective means of ensuring stability in a company's software supply chain.



Helps organizations recruit talent from projects and retain their own developers by engaging them with the open source innovation engine.

Therefore, creating and maintaining a separation of upstream work and product work is essential. In other words, a followed practice is to provide open source developers with guaranteed time to meet their upstream aspirations and responsibilities, especially if they are maintainers. For junior developers, or other internal developers using open source in product components, such interactions with the upstream community will increase their language, communication, and technical skills. In the absence of such an upstream time guarantee, it is easy for these team members to become an extension of product teams, resulting in their upstream focus drying up in favor of product development.

## Create a mentorship program

Set up a mentorship program where senior, experienced open source developers mentor junior, less experienced developers. Typically, a mentorship program runs for three to six months, during which the mentor supervises the mentee's work, assigns tasks, and ensures proper results. The mentor also conducts code reviews and provides feedback on anything the mentee produces before the mentee pushes the code to the upstream project.

This exercise aims to increase the number of developers contributing code to the upstream project and improve individual effectiveness by increasing the quality of code and the percentage of code accepted into the upstream project. In general, four to five mentees should work with a given mentor, and, ideally, they should work in the same area as the mentor to make code reviews more efficient.

## Formalize open source human resources tracking & performance metrics

Mature open source organizations almost always have an open source developer track in their HR system. So, individuals hired as open source developers have a good sense of how their careers will progress within the organization. Additionally, organizations often need to adjust their performance-based bonuses and metrics to include goals related to open source development work. Closed source developers' performance metrics are often different from those of open source developers. For example, if an open source developer advocates for the implementation of a given feature, successfully gathers interest, and volunteers to write the code, how would they be rated, especially if they may not have written a single line of code?

Finally, organizations should allow a work-from-home (WFH) policy for open source developers regardless of the general corporate policy. During COVID-19, we witnessed organizations institute WFH policies to allow employees to be productive while under quarantine. It was a fascinating experiment for WFH policies, as organizations continued to operate, innovate, and produce, even though most of their employees worked from home. A WFH policy is almost mandatory in the open source world because open source developers are located worldwide, making hiring and retaining them easier.



## Offer training

It is only possible for organizations to hire some of the senior and most expert developers. They are always looking for ways to increase the competence of their developers in a given technical domain; therefore, in addition to specialized training, organizations need to offer training on the open source development model and the basic concepts of open source legal compliance.

Sample training courses include:

- An open source development methodology course that teaches staff new to open source how open source development works and how to get best engaged with the project community
- An open source compliance course that teaches staff the basics of compliance principles and open source licensing. The course often includes modules covering the organization's policy and process.

The Linux Foundation offers several technical training courses specific to open source technologies and several nontechnical courses, such as this free online open source compliance training for developers.

## Participate in and host open source events

Mature open source organizations support and encourage their developers to host, attend, and participate in open source conferences and events, including local community meetups, hackathons, and summits. Such participation helps open source developers connect personally with their peers, build relationships, and participate in technical discussions that guide the direction of the respective open source projects. As an organization that uses and adopts open source software, it is highly recommended to facilitate for your open source developers the process of attending and presenting at open source events. You can also sponsor big and small events to increase external visibility within the open source global community or simply target events tailored for specific open source projects. As a bonus benefit, such events are also great venues to look for talent.

## Provide a flexible IT infrastructure

Provide a flexible IT infrastructure that allows open source developers to communicate and work with the open source and Linux Kernel communities without any challenges. Additionally, set up an internal IT infrastructure that matches the tools used externally to help bridge the gap between internal teams and the Kernel community or any other open source project community for that purpose.

Open source development uses three primary domains of IT services: 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). Making some or all of these tools available internally properly supports open source development. However, this might conflict with existing organization-wide IT policies. If so, it is vital to resolve these conflicts and allow open source developers to use familiar tools.

## Track developer code contributions

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

With this data, you can compare contributions from various internal development teams to identify where source code contributions are coming from.

For instance, you can use these metrics to compare your performance to other organizations involved in the Kernel ecosystem. This approach helps better inform you about the overall developer ecosystem for the project. In addition, these metrics provide a much better idea of your strengths and weaknesses and can help inform your overall development strategy.

## Identify focus areas with a broad impact

Contribute to and focus on areas that benefit more than one business unit or more than one product. This contribution model, driven by the criticality of software components, allows you to provide value and show return on investment across multiple business units, increasing your chances for more funding and support.

## Foster internal collaboration

Create collaboration projects with other business units that use the specific open source projects in their products. These collaborations can take one or more of many forms:

- Deliver training to their 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 lack resources.
- Help get them off an old fork and onto a mainline version.

These collaborations aim to help the product teams understand their needs and fulfill their product goals via open source enablement.

# Implement inner sourcing practices

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 in the open source community and to foster new employee-to-employee relationships that are cross-functional and touch on multiple product domains.

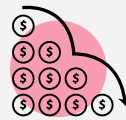
Open source principles work well on large-scale projects distributed across an enterprise. Many Fortune 500 organizations have adopted them externally and internally for the same reasons: faster releases, improved quality, increased innovation and communications, information sharing, reduced costs, greater and more effective collaboration, and increased employee morale and retention.

Inner sourcing prepares organizations to work effectively with external open source communities. It encourages employees to interact with colleagues elsewhere and with external community members without switching contexts. In addition, new employees familiar with this development model may integrate more quickly into established workflows. Finally, business partners are probably already using many of these development practices, so when an organization adopts inner sourcing practices, it is also strengthening its integration with the commercial ecosystem.

**FIGURE 4**  
**BENEFITS OF ADOPTING INNERSOURCE PRACTICES IN THE ENTERPRISE**



Releases cadency faster



Reduces costs of development



Improves source code quality



Increases internal collaboration



Increases motivation



Increases morale, retention



Increases internal information sharing



Increases internal communication

# Recommendations and lessons learned

## Be patient

It takes considerable time to grow internal open source expertise. The goal from an enterprise perspective is to find people with enough peer recognition to be influential in the community. There are typically three pillars to this: domain expertise, open source methodology, and working practices.

## Shift to a more collaborative environment

Internal organization dynamics must be favorable to open source efforts. Implementing these practices requires a shift from traditional software development practices to a more open and collaborative mindset. As an open source leader inside your organization, you will face several challenges in funding resources, justifying ROI, getting upstream focus, etc. These often require a major shift in mindset and a lot of education up the command chain.

## Embrace a flexible IT infrastructure

These open source practices require an IT infrastructure free from many limiting IT policies and a computing environment that supports open source development.

## Adopt proper success metrics

Proper open source metrics drive the desired development behavior. Unfortunately, the traditional metrics often used in product organizations only apply in the context of open source development. For example, we have had multiple instances of the upstream implementation of desired functionality because of OSG developers that lobby for support from the community.

In this case, the number of changesets or lines of code does not matter as much, as the technical leadership team members provide to get code upstream and reduce our downstream maintenance efforts. The metrics we track account for things like this.

## Use a lightweight approval process

Organizations have transitioned from highly complex and cumbersome policies to a more straightforward approach for receiving, reviewing, and approving source code contributions. Dedicated open source teams often receive blanket approval to contribute to open source projects. This is not the case for other groups, which need different approval levels depending on the nature of the contributed code (e.g., simple bug fixes, code to improve existing functionality, code that offers new functionality, or starting a new project). This is a function of the balance between all parties involved: legal, engineering, and open source.

## Share information

The organization must share information and priorities across different divisions. To illustrate this, assume you are on an open source team and request to support the implementation of a driver, but you cannot access the hardware manual and instructions. This situation sounds a bit like playing darts with the lights off; therefore, information sharing is critical to successful internal collaborations between the open source teams and everyone else.

## Make strategic contributions

Focus your contributions on upstream projects that would directly benefit the organization's strategy and products. In open source development, it is easy to get carried away by hopping between different exciting projects. However, in an enterprise setting where the open source group is a cost center, your driving force should be to focus on open source projects that support product development. Open source teams often perform a yearly review of the product portfolio they support and focus their involvement on open source projects commonly used across as many products as possible. Such a methodology drives priorities and is a great way to remain focused on what's essential, justifiable, and fundable.

## Partner with product teams

Be the upstream partner for product teams; they often feel like they are working inside a pressure cooker, especially in a consumer electronics environment. They often seem understaffed, need more critical resources to support parallel upstream development, and are under constant pressure for feature delivery within tight schedules. In such an environment, it is easy to overlook the benefit of upstreaming in favor of short-term time savings, which can, unfortunately, lead to technical debt that has a higher cost in the long term. Open source teams can help by being a partner that focuses on delivering the necessary code upstream, reducing this technical debt.

FIGURE 5

## RECOMMENDED PRACTICES FOR CONTRIBUTING TO OPEN SOURCE PROJECTS



Design & implement with upstreaming in mind to increase the likelihood of patch acceptance.



Ensure the contribution improves or introduces functionality that is useful for a broad base of users.



Stay involved in upstream development post merging with the upstream project.



Document the code to make it easier to understand and to lower the barrier for new contributors.



Upstream for the right reasons. Upstreaming is not a code retirement strategy.



Listen to feedback, and act upon it—rework the code based on the peer review process.



Follow proper coding style, and secure code guidelines.



Follow the processes set by the project for submitting code, new features, etc.

## Grow open source talent

Grow open source talent in specific technology areas relevant to your products. Hiring a few resources from outside the organization is easy, but this approach has several limitations. The alternative approach is to convert your existing developers into open source contributors via training on the technical domain and open source methodology. You can then pair these developers with a mentor to further expand their skills. Encourage developers outside the open source team to learn from and contribute to the open source community. We provide as much help as we can with upstream code contributions. Still, we need more resources and sometimes need a deeper understanding of products that might be necessary to identify where we can adequately upstream code. Better involvement in the open source community from teams outside our own allows us to get more critical code upstream and improves our ability to interact with the community.

## Conclusion

You must earn open source leadership, but you can lose it through a lack of participation. Regular, ongoing participation and contribution are the only ways to ensure your organization maintains open source leadership.

Hopefully, this paper makes the task of improving your enterprise's open source practices more manageable. Following some of the recommended practices will go a long way toward developing internal open source expertise. You can leverage this expertise to improve your products and services and reduce the cost of code maintenance. Many organizations have had considerable success through the use of these strategies.

**FIGURE 6**

### MASTERING OPEN SOURCE SOFTWARE

Master open source software requires you to mastering the three critical Cs:

#### Consumption

Establish internal infrastructure to enable proper practices for open source software consumption: policy, process, checklists, and training.

#### Compliance

Enable open source compliance practices within your development process to ensure proper fulfillment of open source license obligations once products ship.

#### Contribution

Enable your developers to engage within open source projects via a policy and a lightweight process and access to legal support. Provide training on open source development models and best practices.

## FIGURE 7

### TOP 10 TIPS FOR MASTERING: OPEN SOURCE CONSUMPTION

How can you build a healthy environment for open source consumption within your organization? And how can you get ready for the next phase (i.e., becoming a contributor)?

- 1 Establish a policy and process to guide open source usage.
- 2 Set up a team to oversee approvals for all open source usage.
- 3 Understand your open source product strategy and core values.
- 4 Provide the enabling IT infrastructure and tooling.
- 5 Setup an open source license compliance program.
- 6 Offer training to your staff and manager.
- 7 Track everything, measure, improve, and communicate.
- 8 Adopt open source practices for your internal development.
- 9 Identify incoming open source code through your software suppliers.
- 10 Identify key open source projects, and start contributing to them.

## FIGURE 8

### ELEVEN TIPS FOR MASTERING: OPEN SOURCE CONTRIBUTIONS

How can you build a healthy environment for open source contributions within your organization?

- 1 Establish a policy and process to guide open source contributions.
- 2 Set up a team to oversee approvals for all open source contributions.
- 3 Focus contributions in the areas that will enable your technologies.
- 4 Provide the needed IT infrastructure and tooling for contributors.
- 5 Offer training to your staff on contribution best practices.
- 6 Track contributions, measure impact, improve, and communicate.
- 7 Establish a mentorship program to train less experienced developers.
- 8 Provide contributions guidelines, How-To's, Do's and Don'ts.
- 9 Make open source legal support accessible to developers.
- 10 Hire from the open source communities you value the most.
- 11 Always follow community processes / practices of specific projects.



## Acknowledgments

The author would like to express his sincere appreciation to his Linux Foundation colleagues Hilary Carter, Jason Perlow, Melissa Schmidt, Jessica Murillo and Barry Hall for their valuable reviews and feedback. This report has benefited immensely from their experiences, reviews, and contributions.

## Feedback

The author apologizes in advance for any spelling errors or possible errors and is grateful to receive corrections and suggestions for improvements.

## Linux Foundation resources

- **E-book:** [A Deep Dive into Open Source Program Offices: Structure, Roles, Responsibilities, and Challenges](#)
- **E-book:** [A Guide to Enterprise Open Source](#)
- **E-book:** [Open Source Compliance in the Enterprise](#)
- **E-book:** [Open Source Audits in Merger and Acquisition Transactions](#)
- [Linux Foundation Open Source Best Practices for the Enterprise Guides](#)
- [Linux Foundation Open Source Compliance Program](#)
- [TODO Group](#)
- [The Software Package Data Exchange®](#)
- [Linux Foundation Training & Certification](#)
- [Linux Foundation Events](#)



## About the author

Dr. Ibrahim Haddad is the vice president of strategic programs at the Linux Foundation. He focuses on facilitating a vendor-neutral environment for advancing the open source AI platform. Haddad leads the Linux Foundation AI & Data Foundation and the PyTorch Foundation. His work and the work of both foundations support companies, developers, and the open source community in identifying and contributing to the technological projects that address industry and technology challenges for the benefit of all participants. Throughout his career, Haddad held technology and portfolio management roles at Ericsson Research, the Open Source Development Labs, Motorola, Palm, Hewlett-Packard, Samsung Research, and the Linux Foundation. He graduated with honors from Concordia University (Montréal, Canada) with a Ph.D. in computer science. He is fluent in Arabic, English, and French.

 [@ibrahimhaddad](#)

 [@IbrahimAtLinux](#)


 [IbrahimAtLinux.com](#)

**Latest fun project:** [Tux NFT Club](#)

 [twitter.com/linuxfoundation](#)

 [facebook.com/TheLinuxFoundation](#)

 [linkedin.com/company/the-linux-foundation](#)

 [youtube.com/user/TheLinuxFoundation](#)



Founded in 2021, [Linux Foundation Research](#) explores the growing scale of open source collaboration and provides 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.



Copyright © 2023 [The Linux Foundation](#)

This report is licensed under the Creative Commons Attribution-No Derivatives 4.0 International Public License.

To reference the work, please cite as follows: Ibrahim Haddad, Ph.D., "A Road Map to Improve the Effectiveness and Impact of Enterprise Open Source Development," Foreword by Jessica Murillo, The Linux Foundation, February 2023.