



# SUSTAINABILITY GUIDE FOR FLOSS TOOLS

# PURPOSE OF THE GUIDE

---

The "Sustainability Guide for FLOSS Tools" is a straightforward and brief guide made for developers who work with Free/Libre/Open Source Software (FLOSS). It gives useful tips and ideas on how developers can secure the enduring success of their tools.

This guide identifies key areas that are essential for the sustainability of FLOSS tools. It brings together knowledge and advice from different areas to give a full picture of what sustainability means in the context of FLOSS and outlines the necessary steps to address these areas, guiding developers towards a variety of reliable online resources that offer more in-depth information. This is particularly important as the landscape of free and open source software continues to change and grow.

The guide does not just help with immediate needs but also adds to the broader discussion about keeping FLOSS projects healthy and viable. By providing developers with necessary knowledge and tools, this guide aims to play a key role in supporting a strong and thriving free and open source community.

This guide was developed as part of Internews' two-year initiative to support six Free/Libre/Open Source Software (FLOSS) tool teams in their efforts to develop and implement sustainability action plans. Building on the lessons learned from this project, desk research was used to compile the information and resources provided throughout this report, using credible sources from experts in the field. Due to the rapidly developing nature of the FLOSS environment, this report prioritizes recently published materials, and older sources are referenced only where they remain applicable.

This report was written by Amira Galal, Cecilia Mwende Maundu, Joey Salazar, and Skyler Sallick of the Internews Internet Freedom and Resilience team and designed by Skyler Sallick.

January 2024

This content is available as a [CC-BY 4.0 International license](https://creativecommons.org/licenses/by/4.0/).

# Table of Contents

<b>Purpose of the Guide.....</b>	<b>1</b>
<b>Table of Contents.....</b>	<b>2</b>
<b>Forward .....</b>	<b>4</b>
<b>Section 1: Defining FLOSS Sustainability .....</b>	<b>5</b>
Sustainability in FLOSS Explained.....	5
The Importance of Longevity, Support, and Community in FLOSS.....	5
<b>Section 2: Long-Term Planning.....</b>	<b>6</b>
Project Roadmaps.....	6
Suggested Approaches.....	6
Resources.....	7
Suggested Approaches.....	7
Resources.....	7
Backward Compatibility Considerations.....	7
Suggested Approaches.....	7
Resources.....	8
<b>Section 3: Governance and Leadership.....</b>	<b>8</b>
Suggested Approaches .....	8
Defining Roles and Responsibilities.....	8
Transition Planning for Key Members.....	9
Decision-Making Processes and Conflict Resolution .....	9
<b>Section 4: Marketing and Visibility.....</b>	<b>9</b>
Why Does Marketing and Visibility Matter?.....	9
Suggested Approaches .....	10
Online Presence and Content .....	10
Community Building .....	10
Networking and Partnerships .....	11
Resources.....	12
<b>Section 5: Financial Sustainability .....</b>	<b>12</b>
Fundraising Strategies .....	12
Suggested Approaches.....	12
Resources.....	14
Potential Funding Sources.....	14



Suggested Approaches.....	15
Resources.....	16
Financial Management.....	16
Suggested Approaches.....	16
Resources.....	17
<b>Section 6: Community Building and Engagement.....</b>	<b>17</b>
Building an Inclusive and Diverse Community.....	17
Suggested Approaches.....	17
Engaging with the Community.....	18
Suggested Approaches.....	18
Hosting Events and Webinars.....	18
Developer Forums.....	18
Documentation for New Contributors.....	19
Resources.....	20
<b>Section 7: Technical Sustainability .....</b>	<b>20</b>
Licensing and Compliance .....	20
Suggested Approaches.....	21
Resources.....	21
Security Concerns .....	21
Suggested Approaches.....	22
Resources.....	22
Quality of Code .....	22
Suggested Approaches.....	23
Resources.....	23
Code Auditing.....	23
Suggested Approaches.....	23
Resources.....	23
<b>Afterword: Embracing Sustainability in FLOSS .....</b>	<b>24</b>



# FORWARD

---



In a relentlessly digital world, open source software is a testament to the incredible power of collaboration, inclusivity, and shared knowledge on a global scale. There are countless examples of how free, libre, and open source development has spearheaded innovation and, at times, provided solutions to problems when commercial applications could not. The success of projects like Linux in the realm of operating systems showcases how open source can offer not only free alternatives but also transparent and highly customizable solutions for users. Similarly, Mozilla Firefox revitalized browser competition, prioritizing user privacy and offering a faster, more secure browsing experience – giving rise to Tor Browser. And OpenSSL has played a crucial role in enhancing online security by providing a widely adopted solution for secure communication on the Internet.

Amid today's ever-increasing push across industries to innovate and digitize seemingly all aspects of our lives and identities, there is a renewed urgency for open source software to rise to the occasion– not just solving global challenges, but also defending people's interests: from safeguarding digital privacy, freedom of expression, and access to information, to giving rise to decentralized financial systems and emerging currencies to name just a few. We are seeing, in real time, the rise of new transformative technologies such as AI, alongside alarming geo-political events that expose the vulnerability of users to exponentially more powerful providers in an overly centralized digital ecosystem. At a moment's notice, corporate and government interests can change, and open source software is the last line of defence offering alternatives that give power back to users. Open source projects have an incredible track record of tackling complex issues with collective expertise.

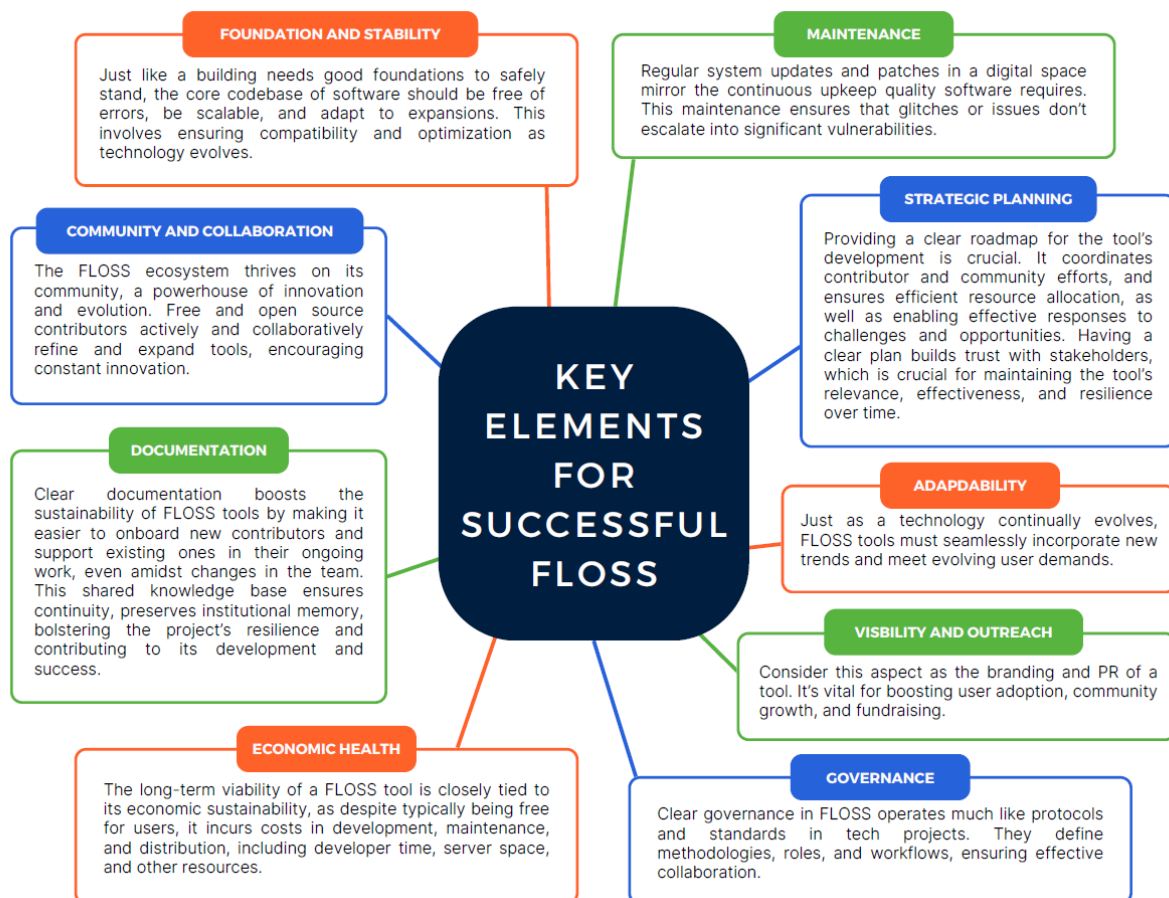
At the Tor Project, we know from first-hand experience that better digital tools are built with a collective vision, effort, and power. We are also looking back on more than two decades of maturation as an organization. This Sustainability Guide for FLOSS Tools manages to condense a lot of this kind of practical experience into a concise and easy-to-digest document. Laying out critical organizational elements to sustaining open source tech ecosystems and operations, it serves both as a practical-oriented cheat sheet for developers and maintainers, and as an advocacy tool for contributors and community members. We are certain that this guide will contribute to supporting future innovation in the FLOSS community and sustain and grow the impact of this technology in people's lives across the globe.

## The Tor Project

# Section 1: Defining FLOSS Sustainability

## Sustainability in FLOSS Explained

In the complex and diverse landscape of Free/Libre/Open Source Software (FLOSS), grasping the long-term viability of each FLOSS tool is crucial for both the tool's sustainability and the ongoing stability of the broader community. Each tool, be it an expansive data storage solution or a complex network configuration, plays a pivotal role in the ecosystem. Their sustainability ensures that these tools not only meet current needs but can also adapt and evolve with emerging technological trends. As many organizations and individual developers rely on these FLOSS tools, their long-term viability and adaptability become essential for continuous innovation, reliability, and overall progression of the free and open source movement.



## The Importance of Longevity, Support, and Community in FLOSS

Free/Libre/Open-Source Software (FLOSS) thrives on the principles of collective collaboration and shared value. Three pillars are central to its success: longevity, support, and community.

- **Longevity:** Beyond just its temporal existence, longevity is about staying relevant, adaptable, and non-obsolete in a fast-evolving tech world. This involves ensuring ongoing value, open access, and rights to use, modify, and distribute software across generations.



- **Support:** Support in FLOSS extends beyond technical help to a continuous exchange of knowledge and expertise. This collaboration fosters innovation and keeps the ecosystem vibrant, benefiting both experienced developers and newcomers.
- **Community:** The FLOSS community is more than a user group; it is the movement's core. It promotes transparency, participatory development, and democratic values, countering monopolistic trends in tech by prioritizing innovation and ethics over profit.

FLOSS is not just about open access to software. It is a vibrant mix of long-lasting relevance (longevity), collaborative problem-solving (support), and a strong, active community. These pillars not only underpin the software but also uphold the philosophy that underpins the FLOSS movement, highlighting the importance of *freedom, active collaboration, and collective advancement* in tech.

## Section 2: Long-Term Planning

Sustaining a FLOSS project over time requires careful planning and management. By setting clear roadmaps, managing deprecation and version control effectively, and considering backward compatibility, project maintainers can ensure a robust and enduring project. The key to sustainability lies in balancing progress with stability and community engagement with clear, transparent communication.

### Project Roadmaps

A project roadmap is crucial to FLOSS sustainability as it provides a clear, strategic direction, ensuring that contributors and users are aligned towards common goals and milestones. It also helps in managing expectations, fostering trust and engagement within the community, which is vital for the long-term vitality and growth of the project.

#### *Suggested Approaches*

- **Vision and Goals:** Define the long-term vision and specific goals for your FLOSS project. This helps in aligning the team and the community towards a common objective.
- **Milestones and Timelines:** Break down the project into manageable milestones with realistic timelines. This provides a structured approach and helps in tracking progress.
- **Community Involvement:** Engage with the community regularly. Gather feedback and incorporate it into the roadmap. This ensures the project remains relevant and community driven.
- **Transparency:** Maintain transparency in the roadmap development process. Publicly share updates and changes to keep the community informed and involved.



## Resources

- [The Open Source Way Guidebook](#): A manual on managing open source projects, focusing on vision, governance, and leadership.
- [Setting Open Source Strategies](#): An article on developing and measuring open source strategies, addressing a project's purpose, goals, and desired end state.

## Deprecation Strategies and Version Control

Deprecation strategies and version control are essential to FLOSS sustainability as they ensure smooth transitions between updates, maintaining system integrity and user trust. These practices also facilitate clear communication about changes and future plans, which is key to managing community expectations and fostering a stable development environment.

### *Suggested Approaches*

- **Clear Communication**: Announce deprecation plans well in advance. Provide detailed explanations and timelines to prepare users for changes.
- **Versioning Protocol**: Adopt a consistent versioning system (like Semantic Versioning) to signal changes, especially breaking changes, to users.
- **Upgrade Path**: Offer a clear and manageable upgrade path for users. Provide detailed documentation and support to ease the transition.
- **Archiving Old Versions**: Ensure that older versions remain accessible for users unable to upgrade immediately. This includes maintaining documentation for these versions.

## Resources

- [Open Source Series: Version Management](#): An article that examines software versioning, especially semantic versioning (SemVer), and its importance in open-source software, including managing breaking changes.
- [What Organizations Need to Know When Deprecating APIs](#): A blog post that outlines best practices for API deprecation, focusing on clear communication, transition periods, versioning, and user-impact minimization.

## Backward Compatibility Considerations

Prioritizing backward compatibility is key. By ensuring that new updates seamlessly integrate with existing functionalities, this strategy reduces user disruption. This approach not only retains a dedicated user base but also fosters continuous community engagement, both of which are essential for the enduring growth and health of the project.

### *Suggested Approaches*

- **Compatibility Goals**: Define your project's backward compatibility goals. Decide how many previous versions you intend to support.





- **Testing:** Implement rigorous testing for new releases to ensure they do not break backward compatibility. Automated testing can be particularly effective here.
- **Documentation:** Clearly document any backward-incompatible changes. Provide examples and guidance on how to adapt to these changes.
- **Gradual Transition:** Where possible, make backward-incompatible changes gradually. Introduce warnings in earlier versions before fully deprecating features.

## Resources

- [Ensuring backward compatibility in your applications](#): An article that discusses strategies for backward compatibility in software, emphasizing functionality maintenance and methods like coding, feature flags, and versioning.
- [How can you ensure backward compatibility in developing new software?](#): This LinkedIn collaborative article provides a step-by-step guide on ensuring backward compatibility in software development, focusing on key strategies for consistent functionality.

## Section 3: Governance and Leadership

Good [project governance and leadership](#) are crucial for ensuring project stability and sustainability, far beyond mere bureaucratic formalities. This involves clearly defining roles and responsibilities, planning for smooth transitions, and establishing effective decision-making processes. Additionally, having mechanisms in place for constructive conflict resolution is vital in maintaining a healthy community, underlining the importance of strong governance and leadership in any project's success.

### Suggested Approaches

#### *Defining Roles and Responsibilities*

Establishing clear roles and responsibilities is akin to creating an orchestra where every member knows their part. It is essential for maintaining order and accountability within your FLOSS community.

- **Maintainers:** These are the architects of your project's vision. Define the roles and responsibilities of maintainers, who oversee the codebase and contributions. They guide the project's direction, ensure quality control, and lead discussions on critical decisions.
- **Contributors:** Contributors form the lifeblood of your project. Outline the expectations and responsibilities of contributors and how they can get involved. Encourage them to provide code, documentation, and bug fixes. Make it clear how they can submit contributions and collaborate with maintainers.
- **Community Managers:** If applicable, describe the roles and responsibilities of community managers who engage with users and contributors. These individuals often



serve as bridges between the community and project leadership. They facilitate communication, organize events, and ensure community health.

### ***Transition Planning for Key Members***

In the fast-paced world of FLOSS, change is inevitable. Preparation for the [departure of key members](#) is essential to maintain project continuity.

- **[Knowledge Transfer](#)**: Document important project knowledge and processes to ensure a smooth transition. This includes technical documentation, decision logs, and guidelines.
- **[Succession Planning](#)**: Identify potential successors for key roles within the project. Encourage contributors to step up and take on leadership positions when the need arises. Nurturing future leaders ensures project resilience.

### ***Decision-Making Processes and Conflict Resolution***

Clearly defined decision-making processes and conflict-resolution mechanisms are the gears that keep your FLOSS community running smoothly.

- **[Decision-Making](#)**: Document how decisions are made within the project, whether through consensus, voting, or other methods. Transparency is key to community trust. When contributors understand how decisions are reached, they are more likely to participate actively.
- **[Conflict Resolution](#)**: Outline how conflicts within the community or project are resolved in a fair and constructive manner. Conflicts can arise from differences in opinion, technical disagreements, or personal issues. A structured resolution process ensures disputes are managed professionally.

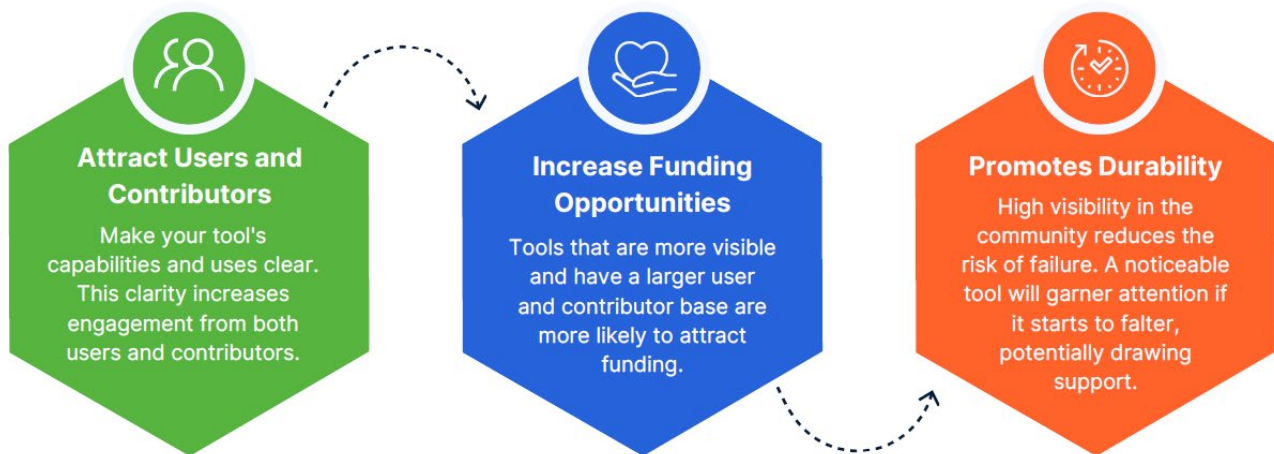
## **Section 4: Marketing and Visibility**

For sustainable success, open source teams must prioritize marketing and brand visibility. Despite challenges such as limited time, resources, or marketing expertise, understanding the value of these efforts is key to tool longevity. As the open source market rapidly expands, it becomes increasingly challenging to ensure that your tool stands out. Improved visibility not only attracts more users, contributors, and potential donors, but also facilitates meaningful growth and opportunities for funding.

### **Why Does Marketing and Visibility Matter?**

In the context of the FLOSS ecosystem, capturing the attention and resources of users, contributors, and donors is crucial for the success and sustainability of a tool. Here is a breakdown of its key benefits:





Building and sustaining a successful FLOSS tool requires deliberate team effort to create and implement a robust marketing and visibility strategy.

## Suggested Approaches

### *Online Presence and Content*

Enhancing your tool's online presence with distinct messaging and branding greatly increases its visibility, drawing in users, contributors, and donors. Consider these strategies to effectively market and enhance your tool's visibility:

- **Messaging and Branding**: Create messaging and branding that helps distinguish why users, contributors, and donors should invest their limited time and resources into your tool rather than your competitors'.
- **Website and Search Engine Optimization**: Use your brand identity to create clear messaging that highlights what makes your tool unique. Your tool should not only be easily recognizable, but it should also be easy to find online.
- **Social Media**: Grow your tool's community by promoting key pieces of information on social media platforms. Create a narrative around your tool that draws individuals to your website through your brand identity and clear messaging.
- **Content Marketing**: Target technical and non-technical audiences with your marketing strategy by highlighting a mix of case studies, technical updates, testimonials, and documentation. A key part of your tool's community may not relate to technical content, so it is crucial to use a multitude of methods for sharing what makes your tool unique.

### *Community Building*

Engaging with the community is a critical aspect of sustaining FLOSS projects and should be a central feature of any marketing and visibility strategy. An active community helps ensure that its members remain involved, passionate, and committed to the success of the project and the wider FLOSS ecosystem. Find more information and in-depth suggested approaches for community building in [Section 6: Community Building and Engagement](#).

- **[Engagement and Support](#)**: Use your marketing and visibility strategy to create pathways for inclusive and diverse community engagement and support.
- **[Transparency and Open Development](#)**: Regularly sharing updates on the project's progress, upcoming features, and decision-making keeps the community informed and engaged. It is essential to give users and the community insight into your tool's development. Prioritize this in your marketing and visibility strategy, as the sustainability of an open source project depends on community involvement.
- **[User Empowerment](#)**: Involve users in your marketing and visibility efforts. Engage the community by conducting polls on preferred documentation types, requesting testimonials, and inviting them to participate in campaigns that highlight the tool's unique value from their perspective. A successful community is not just about attracting contributors; it is about fostering a space where they feel valued and motivated to contribute.

## ***Networking and Partnerships***

Community presence is crucial for FLOSS sustainability. Embedding your tool within the FLOSS community through networking and partnerships opens doors for other key aspects of sustainability such as financial sustainability. The more embedded your tool is within the open source community, the more durable your tool will be in the long run.

- **[Collaborations](#)**: Connect with other open source projects through participation in developer forums. Contributing to other projects and actively participating in the open source community will boost your tool's reputation and allow for greater visibility of your work. Explore [Section 6: Community Building and Engagement](#) for more information about developer forums.
- **[Conference Speaking and Participation](#)**: Increase your tool's visibility to potential users, contributors, and donors by actively participating in conferences and events— both in-person and online. Use your tool's logo and branding at these gatherings to enhance recognition and spark discussions about your tool's role in the broader FLOSS ecosystem.
- **[Showcasing Case Studies](#)**: Highlight success stories of users who have benefitted from your tool. If necessary, keep these users anonymous and describe their use-cases generally, especially if there are security concerns. These case studies demonstrate to potential users, contributors, and donors the relevance and effectiveness of your tool in serving important communities.
- **[Funding and Sponsorship](#)**: Utilizing networks and partnerships through marketing and visibility can lead to funding opportunities. Donors want to provide funds to tools that are visible within the community. The more visible your tool is, the more likely it is to receive funding and/or sponsorship opportunities. Visit [Section 5: Financial Sustainability](#) for more information about accessing funding opportunities.



## Resources

Explore a curated collection of resources that can help you build a marketing plan and grow your visibility.

- [How To Market an Open-Source Project](#): A Linux Foundation blog post on how (and why) to market your open source project.
- [Why Open-Source Needs Marketing \(Even Though Developers Hate It\)](#): An article in Forbes with 5 key lessons learned about marketing open source projects.
- [Harnessing the Power of Content: Open-Source Software Promotion Strategies](#): A blog post from ClearVoice that breaks down the crucial first steps to developing a successful marketing strategy for open source tools.
- [Finding Users for Your Project](#): A guide within GitHub's Open Source Guides that explains how to use a marketing and visibility strategy to find new users for your tool.
- [Marketing Open-Source Projects](#): A TODO guide from The Linux Foundation that addresses the challenges of marketing open source projects and provides helpful information about how to address those challenges in a successful marketing strategy.
- [Marketing for maintainers: Promote your project to users and contributors](#): A GitHub panel interview where experts answer questions from the open source community about how to promote open source tools to users and contributors.

## Section 5: Financial Sustainability

The long-term success of a FLOSS tool hinges on its economic sustainability. Although these tools are typically free for users, they require funds for development, maintenance, and distribution. This includes costs like developer labor, server space, and other resources. FLOSS projects often struggle with securing maintenance funding, affording market rates for developers, and managing general business development. Balancing financial stability is challenging, as traditional for-profit funding models clash with the values of the open source community. Therefore, open source tools must explore alternative financial models, which presents its own set of challenges.

### Fundraising Strategies

Challenges often arise when larger opportunities require substantial funds due to their extensive requirements. Addressing this involves enhancing skills in financial management, building personal connections with donors, and diversifying funding sources. Additionally, it can be beneficial to provide with several ways to contribute.

### Suggested Approaches

Having a straightforward and well-defined fundraising strategy is crucial. Building personal relationships with donors, offering diverse methods for donations, and securing varied funding sources are key to ensuring your tool's financial sustainability.



## Cultivating Relationships with Donors

Building personal relationships with donors can make your tool more noticeable and appealing in competitive funding environments. Additionally, these relationships foster trust and understanding between your tool's team and the donors, enhancing collaboration and support.

- **[Get to know the donor](#)**: Investigate the organization's history, including past grants, staff, board members, and consult with former grant recipients. This research helps you understand the donor's objectives and how your tool aligns with their mission.
- **[Develop key materials](#)**: Develop a short "elevator pitch," a detailed pitch deck, and a one-page summary about your tool. These materials provide potential donors with quick, accessible information. Having these ready before establishing relationships is crucial, as donors often request such information to better understand your tool.
- **[Utilize your community for donor introductions](#)**: Connect with donors through mutual connections or introduce yourself at professional events such as conferences or workshops.

## Diversifying Your Fundraising Portfolio

Challenges exist in securing stable funding through non-revenue sources, as grants typically offer short-term support and crowdfunding can be inconsistent. However, employing multiple funding strategies simultaneously can contribute to a more sustainable long-term financial plan.

- Assess the effectiveness of existing funding streams, analyzing their stability and long-term viability.
- Identify areas within current funding sources that can be optimized or expanded for better financial support.
- Explore potential untapped funding avenues that align with your project's goals and values.

## Providing Multiple Pathways for Contribution

Offering a variety of ways for donors to contribute is key to maximizing their engagement and support. By catering to the diverse preferences and capacities of donors, you can enhance their involvement and investment in your project. Here are three ways to provide [multiple pathways for contribution](#):

- Offer different donation levels and options, including one-time gifts, recurring donations, and tiered support levels, to accommodate different financial capacities.
- Create opportunities for non-monetary contributions, such as volunteering time, expertise, or resources, appealing to those who prefer to offer support beyond financial means.



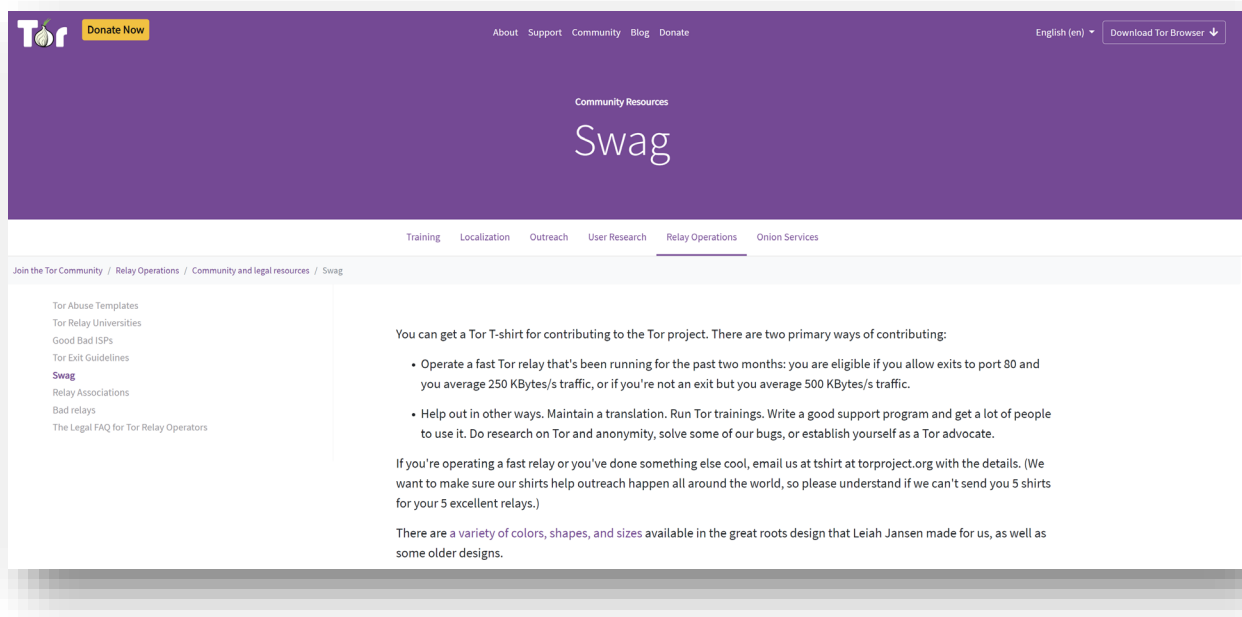


Figure 1: Tor [incentives](#) non-monetary contributions by offering merchandise in exchange for participation as opposed to sale, allowing for multiple pathways of support.

- Integrate innovative funding methods, like gift-matching programs or crowdfunding campaigns, to engage a broader donor base and maximize the impact of each contribution.

## Resources

- [Connecting with Funders](#): A resource from Code for Science & Society on how to connect with funders. The resource provides information on how to ask funders for assistance by preparing a concise pitch and deciphering which funder is right for you.
- [Fundraising Guide for Nonprofits](#): A guide from the Nonprofit Finance Fund that aggregates a series of resources on fundraising for nonprofit organizations. The included resources provide helpful information on how to build a non-revenue fundraising strategy that supports your project's financial goals.
- [How Do I Ask for Funding?](#): A resource from Code for Science & Society on how to ask for funding. The resource outlines key considerations to explore prior to requesting funding and provides information on how to make a specific ask.
- [Financial Sustainability](#): A resource from Code for Science & Society that provides an overview of and general basics for open source financial sustainability.

## Potential Funding Sources

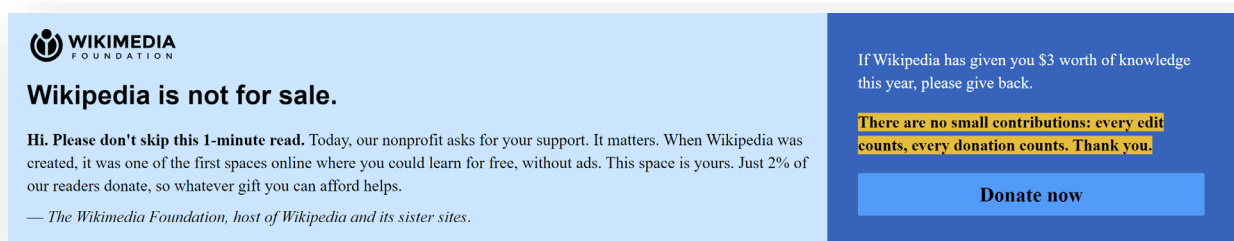
You can secure funding for your FLOSS project through various avenues. Consider applying for grants and fellowships designed for open source initiatives, seeking corporate sponsorships from businesses that value your project, and accepting user donations via platforms like Patreon or



Open Collective. Additionally, you can generate funds by offering professional support, selling branded merchandise, and forming partnerships with educational institutions for projects with academic relevance.

### ***Suggested Approaches***

- **Philanthropic Funds**: Explore different funding opportunities available through private foundations and public charities.
- **Public Funds**: Look into funding from federal or government sources, which could include core program support or specific solicitations from government agencies.
- **Peer-to-Peer**: Establish clear, [specific goals to guide your campaign](#) and use incentives like merchandise, event access, or donation matching to attract and engage new supporters, enhancing the success of your fundraising efforts.
- **Corporate Opportunities**: Tap into the expanding realm of corporate philanthropy, leveraging the growth in businesses looking to support social causes and projects, which can provide substantial funding and partnership opportunities for your initiatives.



The banner features the Wikimedia Foundation logo on the left. The main text reads: "Wikipedia is not for sale. Hi. Please don't skip this 1-minute read. Today, our nonprofit asks for your support. It matters. When Wikipedia was created, it was one of the first spaces online where you could learn for free, without ads. This space is yours. Just 2% of our readers donate, so whatever gift you can afford helps. — The Wikimedia Foundation, host of Wikipedia and its sister sites." On the right, a blue box contains the text: "If Wikipedia has given you \$3 worth of knowledge this year, please give back. There are no small contributions: every edit counts, every donation counts. Thank you." Below this is a "Donate now" button.

Figure 2: Wikipedia utilizes brand values to fuel peer-to-peer fundraising campaign advertisements.



The store page is titled "Welcome to the Wikipedia Store". Below the title, it says: "Where everything related to the world of Wikimedia, the free knowledge movement, comes to life in the shape of apparel, home goods, stationery, and more." A "Browse Collections" button is located on the left. On the right, there is a collection of merchandise: a black mug with a white logo, a pair of white socks with a black pattern, a black t-shirt with "I support wikipedia" text, a black bucket hat with "[edit]" text, a black backpack with the Wikipedia logo, and a small black and white patterned ball.

Figure 3: Wikipedia [incorporates](#) brand values into their merchandise-based fundraising efforts.



## Resources

- [Nonprofit Grant Writing: How to Secure Grants for Your Cause](#): A blog post that compiles a list of effective grant writing tips and provides additional resources for the grant writing process.
- [What are funders looking for?](#): A resource from Code for Science & Society to help you understand how federal funders and private philanthropy organizations make funding decisions. This resource provides key insights into what different types of donors are looking for in funding recipients.
- [Funding Resources](#): A list of resources from Code for Science & Society on finding funding, creating budgets, how to manage a budget, and working with volunteers— each of which are crucial to your tool's financial sustainability.
- [A Guide to Funding](#): A resource from Code for Science & Society that provides a list of fundraising terms that are important to become familiar with as you seek funding opportunities within the nonprofit community.
- [Why Do People Give to Charity? The Psychology of Storytelling and Its Impact on Nonprofit Fundraising](#): A blog post that details how to utilize storytelling to encourage financial contributions from funders and through crowdfunding campaigns.

## Financial Management

Funding for FLOSS tools often focuses mostly on software development, leaving limited resources for other critical aspects of long-term sustainability. Finding time to focus on financial management can be challenging with numerous immediate concerns. However, effectively managing finances is crucial. Not only does it ensure the efficient use of funds, but it also makes the tool more attractive to donors by demonstrating responsible and strategic financial stewardship.

### *Suggested Approaches*

- **Revenue Models**: Developing a clear and detailed revenue model is essential for effectively handling various funding sources. This approach allows for strategic planning and allocation of resources across different funding streams, ensuring the financial stability and sustainability for your project.
- **Budgeting**: Efficient budgeting processes are crucial, as many donors require financial or organizational budget statements during funding applications. Implementing financial tracking tools, budget templates, and other administrative tools can strengthen your financial reporting and administration, illustrating to donors that their investments will be effectively utilized and aligned with your budget structure.
- **Demonstrate Impact**: Creating an annual report that transparently details your use of funds over the past year is crucial. This not only showcases visible outcomes and progress but also reinforces trust and accountability with donors and stakeholders.



- **[Fiscal Sponsors](#)**: Consider partnering with a fiscal sponsor, an organization that manages donations for you, typically for a portion of the donation. Examples include Software Freedom Conservancy, Apache Foundation, Eclipse Foundation, Linux Foundation, and Open Collective, all of which act as fiscal sponsors for open source projects.
- **[Computerized Accounting](#)**: Consider the use of computerized accounting systems to simplify financial management; many of these systems are available as free and open source options.
- **[Financial Policies](#)**: Ensure efficient use of funds by developing key policies. This includes regular financial reporting, vendor disbursement procedures, travel and procurement policies, payroll management, and safeguards against fraud and mismanagement. Regular independent financial audits should also be conducted.

## *Resources*

- **[Financial Management](#)**: A comprehensive list of basic financial policies, financial literacy resources, and pointers about financial management for nonprofit organizations from the National Council of Nonprofits.
- **[Nonprofit Financial Management Webinars](#)**: A series of financial management webinars from the Nonprofit Finance Fund.
- **[5 Crucial Topics to Address in Any Nonprofit Organization's Financial Policy](#)**: A guide that provide information on key components that your organization's financial policies should address.

## **Section 6: Community Building and Engagement**

A [strong, engaged community](#) is vital for the sustainability of an open source software project. It not only aids in development but also creates a sense of ownership and support. Effective communication, community management, and a welcoming environment are crucial for building and sustaining this community. Recognizing contributors and valuing diverse opinions encourage active participation, which is essential for innovation, workload sharing, and ensuring the project's longevity. This ongoing effort benefits the project, its users, and the open source movement.

### **Building an Inclusive and Diverse Community**

A thriving community is inclusive and diverse, welcoming contributors from various backgrounds. It's not just about attracting contributors; it's about creating an environment where they feel valued and empowered to contribute.

### *Suggested Approaches*

- **[Diversity Outreach](#)**: Actively reach out to underrepresented groups and encourage their participation. Consider partnerships with organizations that promote diversity in tech.



- **Code of Conduct**: Implement a clear code of conduct that sets expectations for respectful behaviour. It should make clear that your community is committed to being a safe and welcoming space for all.
- **Mentorship Programs**: Establish mentorship programs to help newcomers get involved and grow within the community. Seasoned contributors can guide and support new members.
- **Inclusive Language**: Promote the use of inclusive language in project communications. Simple changes in language can make a big difference in fostering inclusivity.

Access more community-building resources [HERE](#).

## Engaging with the Community

Engaging with the community is a critical aspect of sustaining FLOSS projects. Once community networks are in place, it's important to keep them dynamic and relevant. An active community helps ensure that its members stay involved, passionate, and committed to the success of the project and the wider FLOSS ecosystem.

### *Suggested Approaches*

Regular and clear communication is key. This can be through newsletters, social media, forums, or regular get-togethers. Updates about the project's progress, upcoming features, and decision-making processes help keep the community informed and involved.

### *Hosting Events and Webinars*

Engaging with your community through events and webinars can significantly boost participation and strengthen relationships. It is an opportunity to connect with your community on a personal level.

- **Hackathons and Contribution Sprints**: Organize events where contributors can work together on specific tasks or features. These events are not only productive but also foster a sense of camaraderie.
- **Webinars and Workshops**: Conduct webinars and workshops to share knowledge and best practices. This is an excellent way to provide in-depth information and answer questions in real-time.
- **Community Gatherings**: Arrange physical or virtual meet-ups and conferences to connect contributors and users. These gatherings provide an excellent forum for networking and collaboration.

When attending events organized by other organizations and teams, take the time to reflect on what you did and did not like about it and most importantly whether it worked for the target audience.

### *Developer Forums*

Engaging with developer forums can be a great way to get answers to your technical questions, share knowledge, and connect with the community. Here are some examples:



- [\*\*Stack Overflow\*\*](#) is a popular Q&A site for developers to ask and answer technical queries, ideal for code troubleshooting.
- [\*\*GitHub Discussions\*\*](#) hosts discussion boards for open source projects, facilitating project-specific conversations and support.
- [\*\*Dev Community\*\*](#) is a network of over a million developers for sharing, collaborating, and accessing resources like events, jobs, mentorship, and more.
- [\*\*GitLab\*\*](#) is a web-based platform that offers source code repository management.
- [\*\*Bitbucket\*\*](#) is a source code repository hosting service, preferred for its strong version control and code management features.

Maintaining strong communication with end-users is key. This can be effectively achieved through user forums and chat channels. These platforms not only facilitate discussions, troubleshooting, and knowledge sharing but also help users support each other and foster a sense of community belonging.

- [\*\*Privacy-Respecting Analytics\*\*](#): Employ analytics tools that prioritize user privacy to gather important usage data. Transparency in how data is collected, adherence to privacy laws, and providing an opt-out option are essential. This data should be used to improve the user experience and inform development decisions, ensuring that user needs are at the forefront.
- [\*\*Feedback Channels\*\*](#): It's important to provide easily accessible pathways for users to report bugs, request features, and offer suggestions. Responsiveness to this feedback is crucial, as it shows users that their input is valued and taken seriously.
- [\*\*User Surveys\*\*](#): Conducting regular surveys is a great way to gather user feedback and insights. These surveys can highlight areas of improvement, desired features, and overall user satisfaction, guiding future developments.

### ***Documentation for New Contributors***

Beginners often [struggle with documentation](#), but it's a crucial aspect of open source projects. Well-structured and comprehensive documentation is not just nice to have; it is the cornerstone of attracting and retaining contributors. It is the first resource newcomers turn to when they want to understand your project.

- **Documentation Structure**: Organize documentation into clear sections, making it easy to navigate. A well-structured documentation hub ensures that users and contributors can quickly find the information they need.
- **Tutorials and Guides**: Create step-by-step tutorials and guides to help contributors get started.
- **Examples and Use Cases**: Provide practical examples and use cases to illustrate how the project works. Concrete examples help contributors grasp concepts faster.



- **Documentation Maintenance:** Regularly update and maintain documentation to keep it current. Outdated documentation can lead to frustration and confusion.

## Resources

- [Starting an Open Source Project](#): This guide provides tips on writing open source documentation. It covers everything from creating README files to documenting your project's code and processes.
- [How to Contribute to Open Source Projects - A Beginner's Guide](#): This step-by-step guide explains how to contribute to open source, including the documentation process, even for those who are non-coders.
- [How to write effective documentation for your open source project](#): This guide explains how documentation quality can make a difference in whether users or contributors try your project or pass it by. It covers how to write *about your code* to convince other developers to follow your project.

Access more documentation resources [HERE](#).

## Section 7: Technical Sustainability

In this section we delve into technical challenges such as Licensing and Compliance Issues, Security Concerns, and Code Auditing, highlighting their direct impact on a project's sustainability. A clear understanding and proactive management of these aspects are crucial, as they influence not only the immediate functionality and security of the software but also its ability to adapt, grow, and remain relevant over time. Through explanations, best practices, and real-world examples, this section aims to empower developers with the knowledge and strategies needed to navigate these challenges, contributing to the robustness and endurance of their open source projects.

### Licensing and Compliance

Licensing and compliance are very important to the sustainability of FLOSS projects, as they provide clear rules for how the software can be used, modified, and shared. A well-chosen license attracts more contributors and users, encouraging an innovative and reliable community around the software. This not only enhances the software's quality and functionality but also increases its appeal, potentially leading to more funding opportunities and economic stability. Clear licensing terms also build trust among users and developers, ensuring a transparent and reliable environment for collaboration.

Meanwhile, compliance helps to avoid costly and reputation-damaging legal disputes related to intellectual property rights. By observing licensing terms and regulations, FLOSS projects can maintain their legal integrity, protecting themselves from potential legal challenges and ensuring smooth operation. This legal stability is crucial for the project's credibility, attracting more users and contributors, and long-term economic viability. Licensing and compliance are not just boring



legal formalities; they are strategic practices that underpin the sustainability of FLOSS projects, ensuring their growth, innovation, and longevity.

### ***Suggested Approaches***

To make sure tools are properly licensed and in compliance with terms and conditions, it is important to understand the different licenses out there. Take time to research and compare various licenses, paying attention to their specific constraints and permissions. The [Free Software Foundation \(FSF\)](#) is a valuable resource, providing extensive information on a wide range of open source licenses. Make sure that the chosen license aligns with your project's goals, clearly defining what is and is not allowed. Use [online tools](#) to find a license that matches your project's objectives. Establish [solid compliance procedures](#) within your team, ensuring everyone observes the licensing terms and related regulations. This includes creating clear documentation on your licensing policy and including a detailed licensing file in your project.

There are several tools and resources available that can help maintain license compliance. These include:

- **[License Management Software](#)**: These tools efficiently manage and monitor software licenses and intellectual property, providing usage reports and compliance alerts.
- **[Compliance Consultants](#)**: Compliance consultants assist in ensuring license compliance, helping with agreement understanding, system implementation, audits, and provide training.
- **Legal Counsel**: Regulatory and compliance lawyers can advise on licensing agreements, manage non-compliance implications, assist negotiations, and address related legal issues (this should be specific to the country where your tool is registered).

When choosing the right tools for your organization, consider the size of your organization, the complexity of your licensing agreements, and the resources available to you.

### ***Resources***

- [FOSSA](#) is a license compliance tool for FLOSS projects that scans and audits licenses and provides remediation suggestions and policy enforcement.
- [License Finder](#) detects the licenses of FLOSS projects and their dependencies. A range of online resources can also be used for checking on [patents](#) and [copyright](#) for different types of content.

## **Security Concerns**

Secure coding practices are essential for FLOSS tools, influencing their widespread adoption and supporting their longevity. As trust plays a central role in the FLOSS community, tools that emphasize security and reliability are more likely to attract users, contributors, and vital financial support in the form of donations or sponsorships.

Ensuring the tool's integrity, availability, and confidentiality bolsters a tool's reputation, fostering both community growth and financial sustainability. Given that many FLOSS projects operate on



tight budgets, security breaches can severely hinder their ability to thrive. Thus, adopting secure coding practices is not just about technical robustness; it is a strategic move towards the project's continued relevance, growth, and financial viability.

### ***Suggested Approaches***

FLOSS tools should create [strong security frameworks](#) that, when used effectively, can both address current vulnerabilities and bolster the team's future capacity to secure their systems. Outlined are several recommended practices for achieving this:

#### **Regular Audits and Secure Coding**

- [Frequent Security Checks](#): Regularly audit the system to identify and mitigate vulnerabilities, tailoring the audit frequency to the complexity of the system and sensitivity of the data.
- [Secure Coding Standards](#): Follow industry-standard security protocols, including data encryption to protect sensitive information and input validation to prevent common vulnerabilities.

#### **Proactive and Educative Approaches**

- [Security Response Plans](#): Develop clear procedures for managing reported vulnerabilities, ensuring prompt validation, impact assessment, and implementation of fixes.
- [Security Training](#): Provide comprehensive training to the team on [security best practices](#) and responsible data handling to foster a culture of security awareness.

#### **External Validation**

- [External Security Auditors](#): Bring in third-party security experts for unbiased assessments, providing an additional layer of validation and help uncovering potential oversights in your security strategy.

### ***Resources***

Explore the [National Vulnerability Database](#) to find current information on security vulnerabilities, including their impacts, solutions, and [metrics](#) for a specific vulnerability.

## **Quality of Code**

Quality code is critical to a software's performance, scalability, and ease of maintenance. Poor quality code poses many challenges, including making it more difficult to onboard new contributors and is unattractive to the developer community. This leads to longer development cycles due to the need for more frequent code reviews and bug fixes, ultimately compromising the reliability of the software. A software's reliability is crucial, as it directly influences user satisfaction and trust in the software, impacting the long-term success and viability of the project.





## *Suggested Approaches*

- **Establish Clear Coding Guidelines:** Develop clear coding standards, such as naming and indentation conventions, as well as commenting guidelines, while also advocating for peer reviews and pair programming to internalize best practices.
- **Form a Dedicated Quality Team:** Create a team specifically tasked with maintaining and enhancing code quality.
- **Enhance Skills and Automation:** Invest in training for code quality and design, build automated testing to detect bugs, and adopt tools for consistent peer reviews.
- **Refine Review Processes:** Develop a multi-tiered review strategy for varying code complexities and host workshops to improve review skills and tool usage.

## *Resources*

Use [Crucible](#) for peer code reviews, enabling team members to collaboratively examine and improve each other's code for enhanced quality and performance.

## Code Auditing

Effective [code audits](#) are crucial for error detection, code quality, and compliance, yet they can be a significant challenge for many teams, due to funding and/or capacity restrictions. Nevertheless, neglecting regular auditing risks errors, inconsistency, and security vulnerabilities, so thinking of it as a key to the tool's sustainability is vital. It is important to coordinate a robust audit process. Weak processes might miss critical issues, leading to long-term maintenance challenges, reduced code efficiency, and increased vulnerability to malicious attacks.

## *Suggested Approaches*

- **Set Audit Objectives:** Outline the audit's focus, highlighting key concerns and all relevant software, hardware, and related technologies, while referencing past audits to inform a thorough and phased approach.
- **Promote Audit Collaboration:** Encourage a transparent, cooperative environment for audits, inviting participation and input from both internal and external stakeholders for impactful integration of results.
- **Build Audit Capacity:** Strengthen internal capacity by providing trainings on audit methods, adopting automated tools, formalizing review protocols, and seeking broad input for ongoing refinement.

## *Resources*

- [Open Source Software Audits: Why, Why, and How to Conduct an Audit](#): Consult Open Logic's guide for essential guidance on the timing, preparation, and best practices for open source software audits.





- [CodeSonar](#): Use CodeSonar to conduct static analysis on your source code or binaries, identifying quality and security flaws.
- [CodeScene](#) offers static analysis capabilities for detecting security and quality defects in your code and binaries.

## Afterword: Embracing Sustainability in FLOSS

As we conclude the "Sustainability Guide for FLOSS Tools," we celebrate our shared commitment to the long-term success of Free/Libre/Open-Source Software (FLOSS). This guide, short yet comprehensive, marks a step forward in our collective journey towards a sustainable open source future.

In the rapidly evolving landscape of FLOSS, sustainability is not just a buzzword; it's a necessary ethos. This guide serves as a primer, providing you with essential insights and directing you to more in-depth resources, understanding the time constraints and practical needs of developers. We've traversed various domains, from technical advancements to community engagement, emphasizing the multifaceted nature of sustainability in the FLOSS ecosystem.

Remember, this guide is a beginning, not an end. The journey towards sustainability is ongoing, and as developers and contributors, you play a crucial role in shaping a thriving open-source community. By embracing the principles of this guide, you contribute not only to your projects' success but also to the larger narrative of open source as a bastion of innovation and collaboration.

In closing, we extend our heartfelt gratitude for your dedication and enthusiasm. Your commitment to FLOSS is what drives this community forward. Let this guide be a beacon, guiding you towards sustainable practices that ensure the vitality and success of your projects in the open-source world.

