WHITEPAPER

Building and Maintaining a Successful Open Source Management Strategy

FOSSA

TL;DR

- Respecting the terms of the licenses that govern the software used by your organization is just as important when the licenses are open source as when they are proprietary.
- Developers choose to use open source code to accelerate development, but manual open source management counters the time-saving advantages that drew them to open source in the first place.
- In an environment where software is deployed continuously, one- time checks for code quality, compliance, security and other factors within open source need to be managed strategically.
- Best practices in open source management include:
  - Automation
  - Real-time auditing and tracking
  - Integration with the software development lifecycle
  - Buy-in from across the organization – including developers, legal, and the C-Suite.

To say that open source software is everywhere is an understatement. Approximately ninety percent of organizations report using open source today[1], due in no small part to the competitive advantage[2] that open source confers. Indeed, open source has become so pervasive that many companies — from small startups to flourish- ing enterprises — cannot even track where all of their open source code and dependencies are.

This state of affairs is remarkable for two reasons. First, for open source enthusiasts, it underscores how tremendously successful open source has become. The open source ecosystem has evolved since the 1990s from a set of niche projects with limited corporate backing, like the Linux kernel and Mozilla, into one that now counts companies such as Uber and Twitter as major open source contributors. This growth signifies a level of success for open source that was virtually unimaginable even just a decade ago, when scholars such as Steven Weber first took serious note of the increasing influence of open source beyond specific developer communities.[3]

1 Den Bleyker, Marlene. “As Open-Source Adoption Skyrockets in Enterprise, Linux Addresses Ease of Use.” SiliconANGLE, 2 June 2017, siliconangle.com/2017/06/02/ enterprise-open-source-adoption-skyrockets-linux-addresses-ease-use-guestoftheweek-devnetcreate/.

2 Rashid, Fahmida Y. “Open-Source Software Gives Competitive Advantage: Gartner Survey.” EWEEK, 19 May 2019, www.eweek.com/servers/open-source-software-gives-competitive-advantage-gartner-survey.

3 Weber, Steve. The Success of Open Source. Harvard University Press, 2005.

At the same time, however, the pervasiveness of open source code has also given rise to a set of new challenges.

With so many companies dependent on open source code in one way or another, managing open source code has become a daunting task. That's especially true given that those companies in some cases do not even realize where all of their open source dependencies exist.

In order to leverage open source successfully and in a way that minimizes risk, companies must ensure that they remain compliant with open source licenses, stay ahead of security vulnerabilities in third-party open source code, maintain the quality of source code supplied by external groups and much more.

Companies must also ensure that any open source code that they develop themselves — either from scratch or by building upon third-party open source components — is properly licensed and managed. This entails choosing the right license for the code, tracking where the code is used and integrating third-party contributions into the code properly.

"Merely identifying where open source is being used is tremendously difficult because a company's software stack cannot be broken neatly into open source and closed source components."

In these types of environments, merely identifying where open source is being used is tremendously diffcult because

a company's software stack cannot be broken neatly into open source and closed source components. As a result, effectively managing all of the responsibilities associated with open source software may seem impossible.

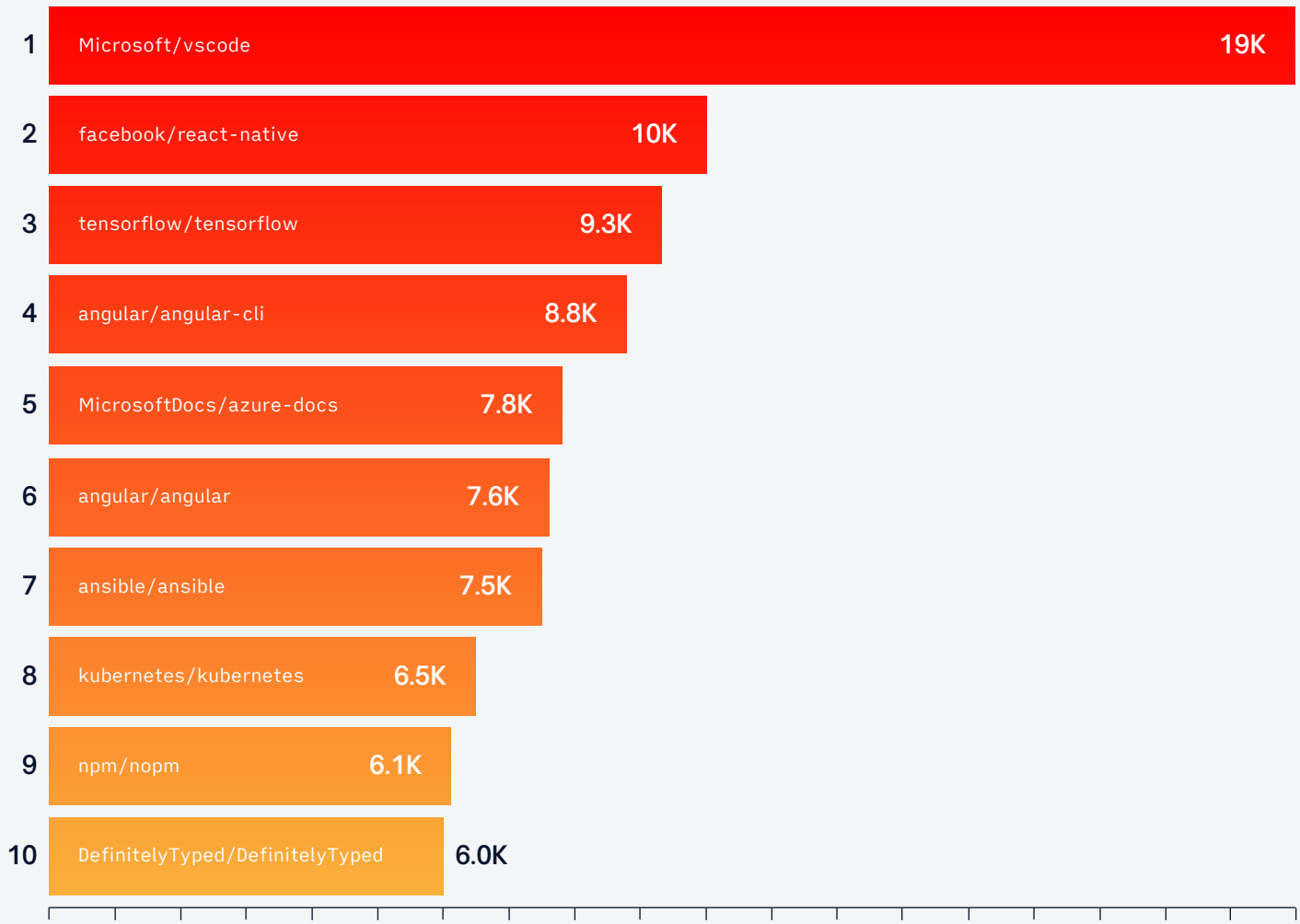
Fortunately, it is not. While open source management is indisputably more diffcult today than it was in the past, it is possible to overcome the hurdles in order to leverage open source successfully and effectively. In order to run a successful open source program it is critical to consider the challenges arise when using open source at scale, and understand the due diligence required to be a responsible member of the open source community while mitigating risk.

These challenges were relatively easy to meet when the typical open source strategy entailed depending on just a handful of distinct open source tools. In the past, companies might have relied on Linux to power some of their servers, and used a tool like Samba to integrate those servers with Windows environments. In that case, identifying which open source projects the company used, and managing the licensing, security, and code quality associated with those projects, was simple and straightforward.

Today, however, the typical use case for open source software looks very different. Instead of depending on open source for specific and discrete needs, companies now rely on open source to help accelerate product development, integrating open source code in complex ways within or alongside proprietary software platforms. Even a simple website that uses WordPress to manage content may rely on a complex mix of open source WordPress code and proprietary WordPress plugins.

Top Open Source Projects

Open Source Software (OSS, or sometimes defined as FOSS – free and open source software) is software with openly shared source code that others are free to contribute to and use as long as it's in accordance with the Open Source License. The top open source projects have over 19,000 contributors from across different organizations.



Source: "Projects | The State of the Octoverse." The State of the Octoverse, Github, Inc., octoverse.github.com/projects.

Open Source Management Challenges

The extent to which the typical company relies on open source software today, as well as the complex ways in which open source is interwoven with closed source code, gives rise to a number of distinct management challenges.

License Compliance

Respecting the terms of the licenses that govern the software used by your organization is just as important when the licenses are open source as when they are proprietary.

The consequences of non-compliance are real. Although most open source licenses are maintained by nonprofit or community groups rather than corporations with strong legal arms, those groups tend to be aggressive about enforcing licensing agreements. The GNU project, the oldest and perhaps most influential project within the free and open source software space, actively solicits information about violations of its various licenses[1]. And while major lawsuits involving open source licensing violations were rare in the early days of the open source software movement, that is no longer the case; we've entered into an age where litigation

"Violations of the GNU License." GNU Project - Free Software Foundation, Free Software Foundation's Licensing and Compliance Lab, 15 Dec. 2018, www.gnu.org/licenses/gpl-violation.en.html.

TL;DR

- Respecting the terms of the licenses that govern the software used by your organization is just as important when the licenses are open source as when they are proprietary.
- Developers choose to use open source code in order to save time in fast- paced workflows, but this can also create risk from a quality-control standpoint.
- When relying on a large volume of open source modules within your software stack, manual open source management practice becomes a serious time-sink for engineers and counters the time-saving advantages that drew them to open source in the first place.
- In an environment where software is deployed continuously, one-time checks for code quality, compliance, security and other factors within open source components does not suffice; those checks need to take place on a continual basis.

related to open source code can put as much as $100 million at stake[2].

Beyond the direct financial costs from litigation, companies that violate open source licenses may also suffer seriously damaged reputations. This could lead to a loss of customers who believe in the importance of open source, as well as potential employees who might decline working for an organization known for failing to uphold open source licensing requirements. Companies also risk losing ownership of their intellectual property in the event that it includes open source that was not used properly, which could result in ownership being assigned to another party.

2 “The $100 Million Court Case for Open Source License Compliance.” WhiteSource, White-Source Software, 21 Mar. 2019, resources.whitesourcesoftware.com/blog-whitesource/the-100-million-case-for-open-source-license-compliance.

Remaining compliant with Open Source Licenses is challenging for three main reasons:

1 **Understanding the implications of each license attached to open source code you either have implemented or want to implement.**

There are over 80 open source licenses currently available. Many are frequently updated and even more are added on a regular basis. Keeping track of the various requirements of

each license is hard enough for a full-time lawyer, let alone developers or business managers who simply want to use open source code.

2 **Open source code is often embedded within other applications, possibly without any licensing information included alongside the code.**

As a result, it can be diffcult even to know which open source licenses apply to the software that a company is using unless engineers perform tedious, time-consuming reviews of source code. This manual process often misses licenses, especially those embedded in unknown dependencies your open source code may rely on. Not only are these reviews often inaccurate, such reviews are a poor use of engineers' time. They can also lead to significant cost-ineffciency for the company, since burdening highly paid engineers with this work prevents them from spending their time on revenue-generating initiatives.

3 **In many cases the employees who choose to use an open source program or module are not well qualified to interpret and ensure compliance with its license.**

Developers, whose primary objective is to build high quality products, are usually the ones who decide which code to use. Most lack the legal expertise to analyze complex licensing requirements. Accidental licensing violations can easily occur as a result — and even accidental misuse can create serious liability.

Security Vulnerabilities

Like any other kind of software, open source is subject to oversights (or, in some cases, deliber- ate insertion of vulnerabilities) that can make it insecure. When a vulnerability is discovered in an open source program, the discoverer may or may not report it to the project's maintainers, who may or may not publicize it and may or may not choose to fix it themselves.

What this means is that identifying vulnerabilities within the open source code companies use and fixing them quickly can be challenging. This is especially true in cases where open source code is embedded within larger applications.

> **"Without the ability to track which open source components you use in real time, and discover problems associated with them, you undercut your ability to solve easy-to-fix issues before they turn into major problems."**

When your company depends on an array of open source tools and modules from a variety of sources spread throughout its software stacks and infrastructure, determining which vulnerabilities affect your code (and finding and applying patches for them) becomes tremendously diffcult.

To illustrate why real-time security tracking and auditing matters, consider what happens when a vulnerability such as Heartbleed is discovered in an open source tool that you depend on. If you fail to patch it quickly, you are at risk of ending up in the position of a company like Equifax, whose

latest headline-making breach was the result of a failure to address known security problems in open source software components[1]. Without the ability to track which open source components you use in real time, and discover problems associated with them, you undercut your ability to solve easy-to-fix issues before they turn into major problems.

Code Quality

The ability to share and reuse code is one of the characteristics that makes open source so powerful. However, it can also make open source risky from a quality-control standpoint. When you depend on code written by third parties, it is diffcult to enforce your organization's standards of code quality, which refers to how neatly and securely code is written.

Complicating this challenge is the fact that developers (understandably) choose to use third-party open source code in order to save time in fast-paced workflows, even if the code is of lower quality. An open source module written by someone else may solve a problem that your developers don't have time to solve themselves, so they use the module but introduce low-quality code.

Similarly, the quality of open source code can degrade over time. The maintainers of a given project could change, and code quality can suffer if less experienced or less devoted developers take over. Open source projects that your company depends on may also be abandoned altogether by their upstream maintainers, leaving it up to you to keep the code up-to-date and in conformance with current best practices.

---

1 Tung, Liam. "Open Source's Big Weak Spot? Flawed Libraries Lurking in Key Apps." ZDNet, 23 Nov. 2017, www.zdnet.com/article/open-sources-big-weak-spot-flawed-libraries-lurking-in-key-apps/.

“When you depend on code written by third parties, it is difficult to enforce your organization’s standards of code quality.”

Scale

The challenges of managing open source code can quickly undercut your organization’s ability to scale its IT processes and pace of development. When you begin relying on dozens or hundreds of open source modules within your software stack, a manual open source management practice becomes a serious time-sink for engineers who have to invest their limited time in tracking open source components within the code base. Eventually, it will completely stop your organization’s ability to continue evolving its technology solutions with the help of open source.

Scheduling

In most cases, the open source code that you adopt is unlikely to be developed according to the same schedule or pace as the rest of your applications. Open source projects set their own development roadmaps, and your organization typically has no ability to control when a third-party project will release its next major software update, fix a security vulnerability or add a new feature that you require. In some cases, an open source project may miss its release deadlines entirely.

Despite this unpredictability, organizations that use open source code must reconcile their own software delivery schedules with those of the open source projects that they depend on. Doing so is especially challenging given the

demands for continuous deployment of software updates that companies face today, which are crucial for remaining competitive in many industries. In an environment where software is deployed continuously, one-time checks for code quality, compliance, security and other factors within open source components does not suffice; those checks need to take place on a continual basis.

“In an environment where software is deployed continuously, on-time-checks for code quality, compliance, security and other factors within open source components does not suffice.”

Programming Language Diversity

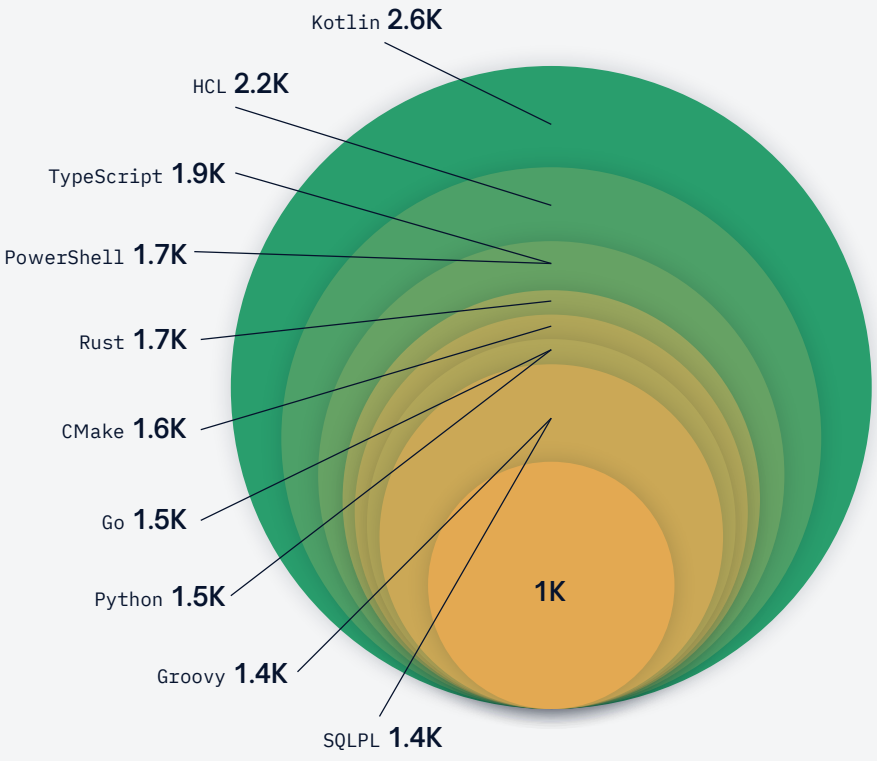
337 distinct programming languages are represented among the open source projects hosted by GitHub alone[1]. While developers might welcome the ease with which the open source model lets them write code in the language of their choice, the diversity of programming languages creates a risk in cases where your organiza- tion reuses code written in a language that your in-house engineers do not know well, and therefore will have difficulty to maintain. This practice limits the ability of your engineers to ensure that the code they use meets quality and security requirements and integrates optimally with the rest of your technology stack.

1 “The State of the Octoverse.” The State of the Octoverse, Github, Inc., 2019, www.octoverse.github.com/.

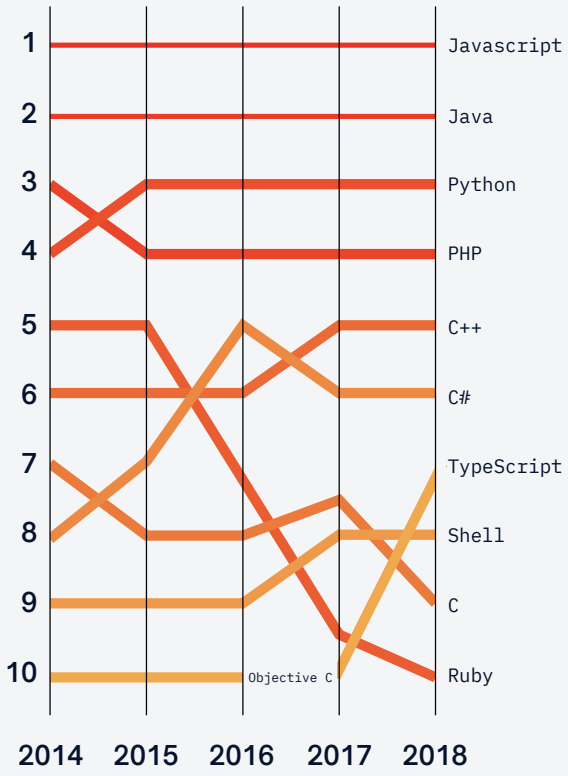
Top Languages and Fastest Growing Languages

These are both the most used open source languages (over the past 5 years), and the fastest growing of the 337 distinct programming languages represented among the open source projects hosted by GitHub.

Top Languages Over Time



Fastest-Growing Languages



Source: “Projects | The State of the Octoverse.” Github, Inc., octoverse.github.com/projects.

TL;DR

- Open source challenges are manageable, and avoiding open source could actually put your company at a competitive disadvantage.
- Best practices in open source management include:
  - Automation
  - Real-time auditing and tracking
  - Integration with the software development lifecycle
  - Buy-in from across the organization – including developers, legal, and the C-Suite.

Best Practices for Managing Open Source Software

Each of the issues described above presents a serious challenge for virtually any organization that uses open source code in a serious way today. In fact, these challenges may appear so intimidating that you might conclude that the best approach is to avoid open source code entirely.

That strategy is rarely the right solution. Choosing not to take advantage of the hundreds of thousands of open source software tools available today would likely put your company at a competitive disadvantage. Moreover, avoiding open source may well prove impossible in practice, since your employees might use open source code without even realizing it. An action as simple as deploying a WordPress plugin on a company website, or downloading an app on a company-owned phone, means that your company is using open source.

Key Goals of an Effective Open Source Management Strategy

1. Achieve continuous visibility into and awareness of open source code anywhere within their organization.
2. Avoid licensing compliance problems related to open source code.
3. Minimize the amount of time that engineers spend tracking and managing open source code manually.
4. React quickly to security vulnerabilities within open source components that they depend on.
5. Align the code quality of open source dependencies with their own standards.

Essential Steps for Achieving your Open Source Compliance Goals

Automation

A major challenge across several aspects of open source management is lengthy and error-prone manual processes. In order to counter this automating key workflows should form the basis for any open source code management strategy. Although identifying open source code within your codebases and reviewing associated requirements

manually may work on a small scale, managing open source on a large scale requires tools and processes that allow you to automatically track which open source modules you are using, what the licensing and security requirements associated with them are, and how those requirements change over time.

Best practices entail deploying an open source management tool that integrates with whichever tools your organization uses to build and deploy software. The tool should be able to scan your code automatically and continuously to identify and track open source comments, whether they are developed by you or a third party. It should generate reports to notify you of licensing, compliance, quality or security issues within that code. And it should be flexible enough to work with any type of programming language in the rapidly evolving open source ecosystem.

“Automation not only minimizes expenditure of manual time and effort — which in turn facilitates scaling — but also helps to ensure consistency across teams and projects with regard to the way in which they manage open source code.”

In order to be used most effectively, of course, automation in open source management should be balanced with clear escalation paths that allow for manual intervention when necessary. For example, your open source management tool should automatically flag issues and notify the legal team when open source licenses violate in house policies, but you need to have a clear company process to resolve those issues.

By embracing automated processes involved in managing open source software, organizations can take advantage of market opportunity and stay innovative, while mitigating risks.

Real-Time Auditing and Tracking

The introduction of agile development has accelerated the speed at which software changes. Open source codebases are constantly updated, making it more difficult to address problems with code performance, security and compliance. Real-time auditing allows you to re-evaluate the current state of your open source compliance and security. With real-time auditing, your codebase is scanned at every deploy. Issues and policy violations are identified immediately for streamlined resolution.

When you monitor your codebases in real-time for open source code, you not only keep an up-to-date database of which open source modules you are using, but you also position yourself to react instantly to security vulnerabilities, feature updates or other important changes to the open source projects on which you depend.

> **“Keeping pace with continuous delivery schedules requires baking open source compliance and management directly into software delivery processes.”**

Integrate With Your Software Development Lifecycle

Keeping pace with continuous delivery schedules requires baking open source compliance and management directly into software delivery processes. In other words, tracking and

auditing of open source code can't be disconnected from the rest of your software delivery pipeline; they need to become an integral part of it.

The only way to achieve this goal is to deploy open source manage- ment tools that can integrate with the rest of your software deliv- ery and deployment tool set and perform tasks such as Common Vulnerability Exposures/Enumeration (CVE) scanning and license auditing in real-time as code rolls from development to testing to release.

Buy-In

Keeping pace with continuous delivery s open source software requires achieving the support of a variety of stakeholders. They range from technical employees (like developers), to legal experts or managers who focus squarely on business outcomes.

It is critical to communicate the importance of proper open source code management to each of these groups, as well as empower them with the tools they need to do their part in managing open source code properly.

Conclusion

Today, the vast majority of organizations are taking advantage of open source software. In order to ensure that the benefits of open source are not outweighed by risks such as licensing compliance problems, poor code quality, or security vulnerabilities, organizations must manage their open source code responsibly.

Effective management entails the use of automated tools to keep track of which open source codebases and licenses your organization uses, real-time monitoring and auditing of security vulnerabilities and licensing compliance efforts, and the integration of open source management into the rest of your software delivery pipeline.

About FOSSA

FOSSA can help to achieve all of these best practices. By providing automated, real-time licensing and vulnerability management for open source code no matter where it exists within your software stack, FOSSA helps organizations minimize the risk and maximize the benefit of open source. Request a demo to learn more, or import FOSSA from GitHub to start analyzing your open source dependencies today.

fossa.com