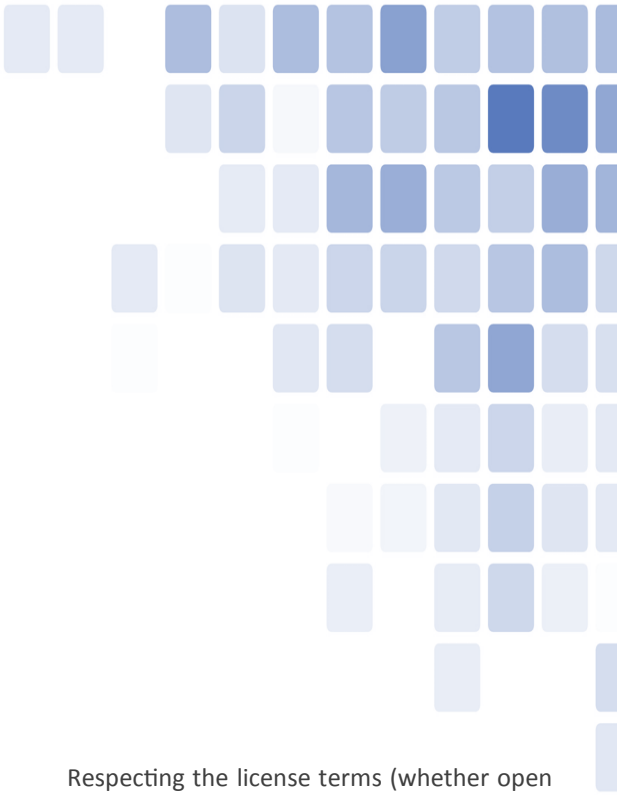


TRINITI

Building and Maintaining a Successful Open Source Management Strategy

It would be an understatement to say that open source software is everywhere. Today, more than 85% of organizations report using open source - from startups to large enterprises. While this incredible growth signifies a completely different level of success, at the same time, extensiveness of open source has also arisen new set of challenges.

With so many companies dependent on open source, its manageability has become an intimidating task. Organizations in some cases do not even realize where all their open source dependencies exist.

APPROXIMATELY

90%

OF ORGANIZATIONS REPORT USING OPEN SOURCE TODAY

In order to leverage open source successfully and minimize risk, companies must remain compliant with open source licenses, check on security vulnerabilities in third-party open source code, maintain the source code quality supplied by external groups and much more.

Companies must also ensure that any open source code that they develop, either from scratch or by building upon third-party open source components — is properly licensed and managed. While on one hand open source manageability is difficult than the past, it is possible to overcome the challenges effectively to leverage open source successfully.

Simply identifying where the open source is being used is highly difficult because an organization's software stack cannot be easily broken into open source and closed source components.

Today, the use cases for open source software looks different. It is not limited to specific or discrete needs but the companies are relying on it for accelerated product development, thus integrating open source code heavily with proprietary software.

Respecting the license terms (whether open source or proprietary) that governs the software used by your organization is important.

• Developers choose to use open source code to accelerate the development process, but manual open source management counters the time-saving advantages that made them use open source in the first place

• In an environment where software is continuously deployed, a one time check for code quality, security, compliance, and other factors within open source need to be strategically managed

• Best open source management practices include:

a. **Automation**

b. **Real-time auditing and tracking**

c. **Integration with the software development lifecycle**

d. **Buy-in from across the organization – including stakeholders, developers and legal**

Open Source Management Challenges

1. Licensing Compliance

Respecting the license terms (whether open source or proprietary) that governs the software used by your organization is important.

The consequences of non-compliance are real. Although most open source licenses are maintained by nonprofit or community groups, they tend to be enforce licensing agreements. Also, we have entered into an age where litigation related to non-compliance of open source can cause millions at stake, which was not the case in the early days of the open source software movement

Beyond the direct financial costs from litigation, organizations violating the licenses may also suffer seriously damaged reputations. This could lead to loss of customers who believe in the importance of open source, as well as potential employees who might decline to work for an organization known for failing to uphold open source licensing requirements.

2. Security Vulnerabilities

Like any other software, open source is subject to oversights that can make it insecure. When susceptibility is discovered in an open source program, the discoverer may or may not report it to the maintainers, who may or may not publicize it and may or may not choose to fix it themselves. Which means that identifying vulnerabilities within the open source code and fixing them can be challenging. This is especially true in cases where open source code is embedded within larger applications. Organizations today depend on various open source tools from multiple sources spread throughout its software stacks and infrastructure. Thus, determining which vulnerabilities affect your code (and finding and applying patches for them) becomes tremendously difficult. Without real-time ability to track which open source components you use, and discover problems associated with them, you undercut your ability to solve easy-to-fix issues before they turn into major problems.

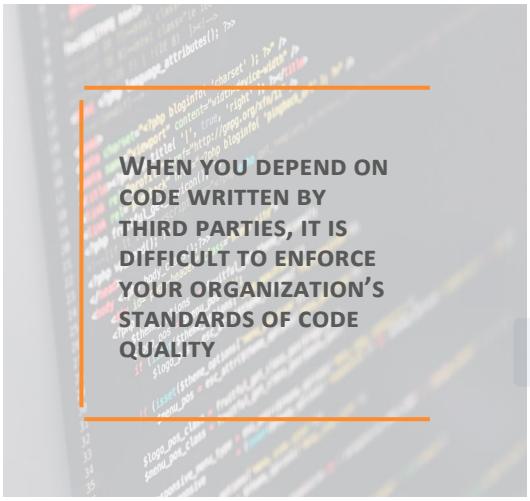
WITHOUT REAL TIME ABILITY TO TRACK WHICH OPEN SOURCE COMPONENTS YOU USE, AND DISCOVER PROBLEMS ASSOCIATED WITH THEM, YOU UNDERCUT YOUR ABILITY TO SOLVE EASY-TO-FIX ISSUES BEFORE THEY TURN INTO MAJOR PROBLEMS.

3. Code Quality

One of the most powerful characteristics of open source is the ability to reuse and share code. However, it can also make open source risky from a quality-control perspective. When you depend on code written by third parties, it is difficult to enforce your organization's standards of code quality.

Complicating this challenge is the fact that developers (understandably) choose to use third-party open source code to save time in fast-paced workflows, even if the code is not up to the mark in terms of quality. An open source module written by a third party 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 code quality can also degrade over time. The maintainers of a given project could change, and code quality can suffer if less skilled or less dedicated 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 updated and in conformance with current best practices.



4. Scale

The challenges of managing an open source environment can quickly wane your organization's capability to scale its development speed and IT processes. When you reply on multiple open source modules within your software stack, a manual management practice becomes a serious time-issue for developers who have to invest their time in tracking open source components within the codebase. Eventually, it might halt your organization's ability to continue evolving its technology solutions with the help of open source.

5. Scheduling

In most cases, the open source code that you plan to adopt doesn't really get developed according to the speed of the rest of your applications. Usually, most of the open source projects set their own roadmaps and you wont really have an ability to control the release of next software update, fix a security vulnerability or add a new feature as per your need. In some cases, an open source project may entirely miss its release deadlines.

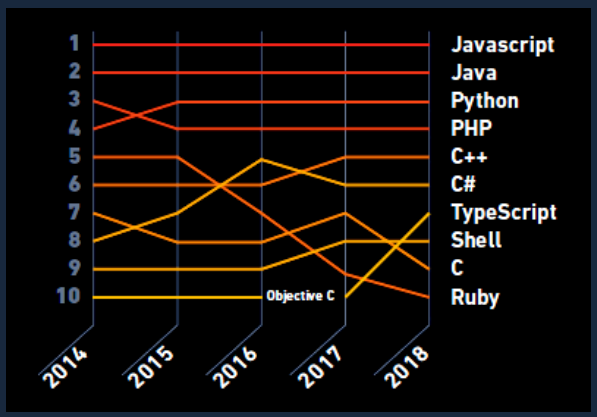
Despite this irregularity, organizations must unify their software delivery schedules with those of the open source projects that they depend on. Doing this would be challenging considering the demands for continuous deployment of software updates, which are important for remaining competitive. In an environment where software is deployed continuously, one-time checks for code quality, compliance, security and other factors within open source components is not sufficient; those checks need to happen continuously.

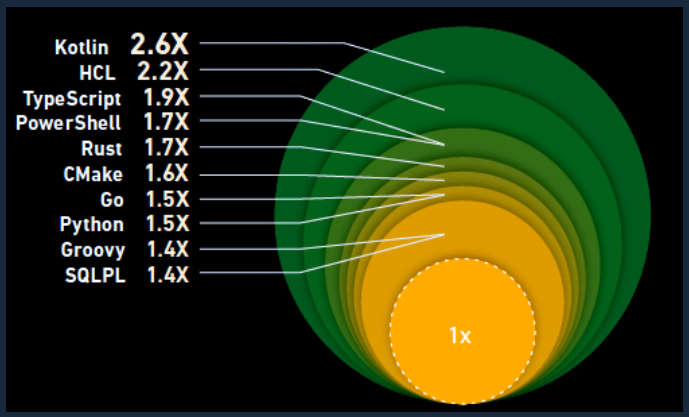
6. Programming Language Diversity

While developers might appreciate the ease provided by open source environment to write code in the language of their choice, this diversity of programming languages also creates a risk where your organization reuses code written in a language that your in-house engineers do not know well, and therefore will have difficulty to maintain. This limits the ability of your team to ensure that the code used by them meets quality and security requirements, integrating ideally with the rest of your technology stack.

>> TOP LANGUAGES AND FASTEST GROWING LANGUAGES

These are both the most used open source languages (over the past 5 years), and the fastest growing programming languages represented among the open source projects





TOP LANGUAGES OVER TIME

FASTEST GROWING LANGUAGES

***Source*—"Projects | The State of the Octoverse."** https://octoverse.github.com/projects

TRINITI

Best Practices for Managing Open Source Software

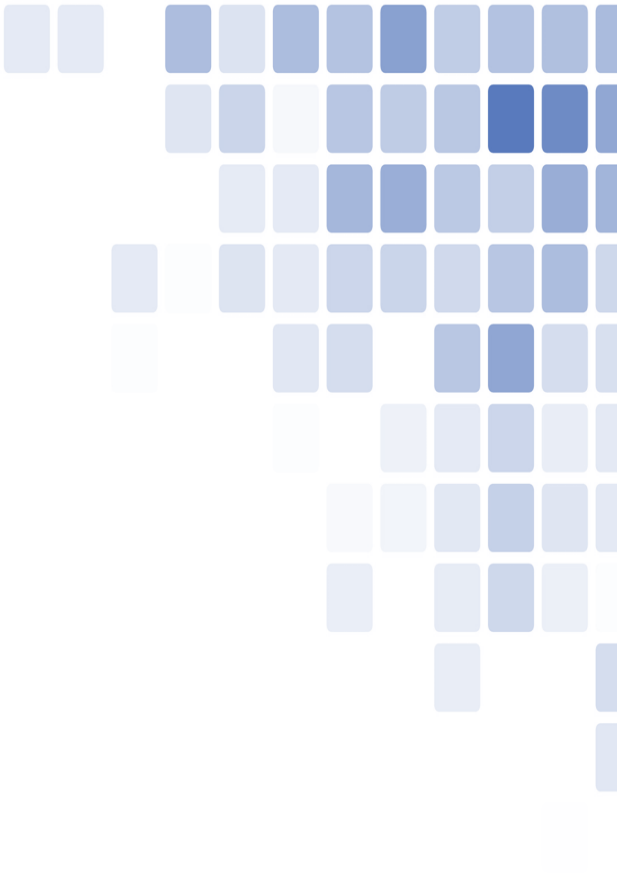
Each of the issues described previously pose a serious challenge for virtually any organization that uses open source code in a big way. In fact, these challenges may appear so intimidating that you might conclude on avoiding open source code entirely.

That strategy is rarely the right approach. Choosing not to take advantage of the hundreds of open source software tools available today would likely put your company at a competitive disadvantage. Moreover, it might seem impossible to completely avoid open source 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.

OPEN SOURCE CHALLENGES ARE MANAGEABLE, AND AVOIDING OPEN SOURCE COULD ACTUALLY PUT YOUR COMPANY AT A COMPETITIVE DISADVANTAGE

KEY GOALS OF AN EFFECTIVE OPEN SOURCE MANAGEMENT STRATEGY

- ⇒ To achieve continuous visibility and awareness of open source code throughout the organization
- ⇒ To avoid licensing compliance issues related to open source code
- ⇒ To minimize the total time spent on manual tracking and management of open source code
- ⇒ To quickly react to the security vulnerabilities within open source components
- ⇒ Align the code quality of open source dependencies with their own standards



TRINITI

Essential Steps for Achieving Open Source Compliance Goals

1. Automation

A major challenge across several aspects of open source management is lengthy and error-prone manual processes. To counter it, automation of key workflows should form the basis for any open source code management strategy. Although identifying open source code and reviewing associated requirements manually might work well on a small scale, managing open source on a large scale requires tools and processes that automatically track open source modules being used, licensing and security requirements associated with them and requirement changes over time.

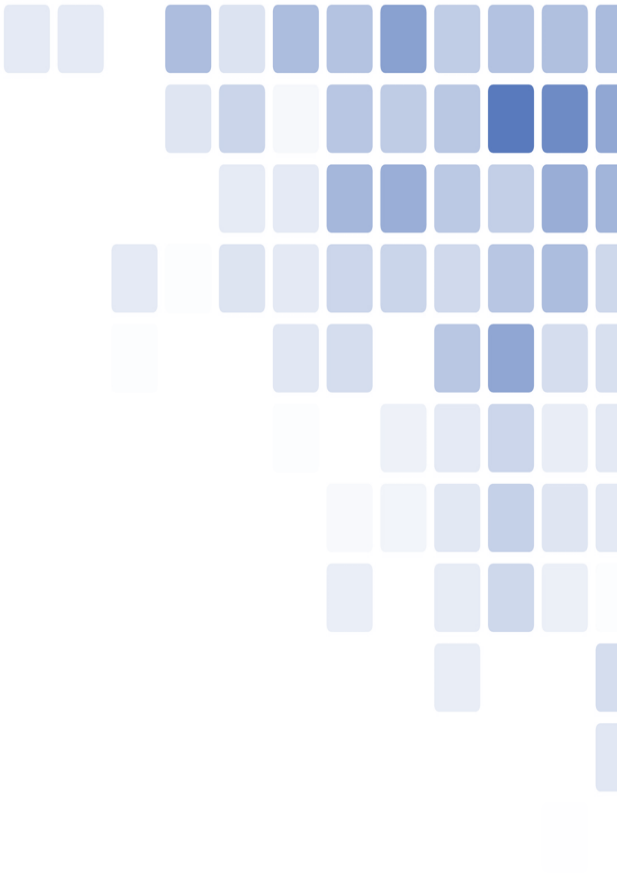
Best practices demand to deploy an open source management tool that integrates with any/every tool your organization uses to build and deploy software. The tool should scan your code continuously scan and automatically identify and track open source comments, whether they are developed by you or a third party. It should be flexible enough to work with any type of programming language in the rapidly evolving open source ecosystem.

AUTOMATION NOT ONLY MINIMIZES EXPENSES FOR MANUAL TIME AND EFFORT, BUT ALSO HELPS TO ENSURE CONSISTENCY ACROSS TEAMS AND PROJECTS WITH REGARDS TO THE WAY IN WHICH THEY MANAGE OPEN SOURCE CODE

Automation not only minimizes expenses for 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.

For effective use, 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.



2. Real time Audit and Tracking

Agile development has accelerated the pace 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. It scans your codebase at every deployment, immediately identifying issues and policy validation 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.

3. Integrate with 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 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 management tools that can integrate with the rest of your software delivery and deployment toolset and perform scanning and license auditing in real-time as code rolls from development to testing to release.

KEEPING PACE WITH CONTINUOUS DELIVERY SCHEDULES REQUIRES "BAKING" OPEN SOURCE COMPLIANCE AND MANAGEMENT DIRECTLY INTO SOFTWARE DELIVERY PROCESSES

4. Buy-In

Properly managing open source software requires the support of stakeholders. They range from technical employees (like developers) to legal experts or managers who focus 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, 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 Triniti

Triniti provides enterprise solutions to its clients, as Triniti firmly believes that these ERP implementations can streamline their core business processes. We build enterprise-class products in the areas of Master Data Management, Application Testing and Objects Migration, Project Management, Business Intelligence, and Reporting. To build a unified enterprise-wide IT platform, we have expertise in integrating Oracle EBS with other heterogeneous enterprise applications spanning product life cycle, manufacturing execution, customer relationship, human capital, and enterprise financial planning and reporting.

Triniti with its prominent expertise and experience in implementing open source based ERP systems help organizations in defining and integrating solutions in-line with their business goals across industry verticals. Since 2016, we have been designing and building an end to end ERP solutions. iDempiere by Triniti has been successfully implemented for customers across Manufacturing, Information technology, Pharmaceuticals and so on.

www.triniti.com

For our open source offerings visit—
www.erp4cloud.com