The Tidelift guide to managing open source

BEST PRACTICES FOR SOFTWARE DEVELOPMENT TEAMS SEEKING TO OPTIMIZE THEIR USE OF OPEN SOURCE COMPONENTS

February 2019

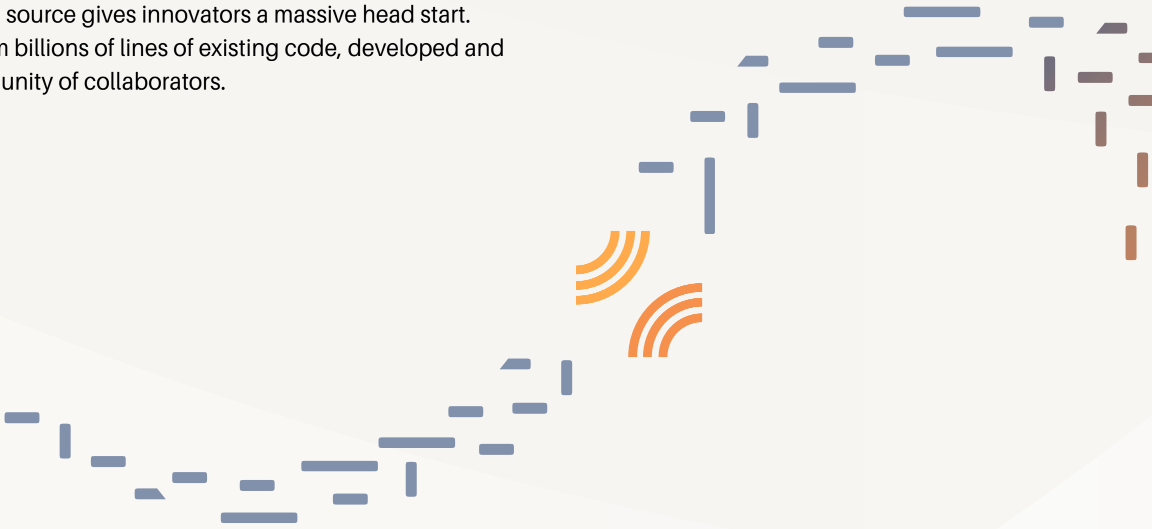
TIDELIFT

INTRODUCTION

Open source is everywhere

Today, open source forms the foundation of most professional software projects. Tidelift's recent study of professional developers revealed that on average, 92% of their projects contain open source components. And 68% reported that **all** of their projects contain open source components.

And it's no wonder. Open source gives innovators a massive head start. Developers can work from billions of lines of existing code, developed and shared by an open community of collaborators.



"As open source components lower costs, improve quality and speed software development processes, they've quickly become the building blocks of the modern software supply chain."
—Heather Meeker, Wired Insights

We mean, it is—quite literally—everywhere

Many people know open source in the context of standalone systems-level software like Linux, or applications like the Apache web server or MongoDB database. What many organizations don't realize is that the applications their in-house teams develop are also primarily built with open source libraries and components.

It's very likely that at least some—if not most—of the underlying code in your applications comes from open source projects.

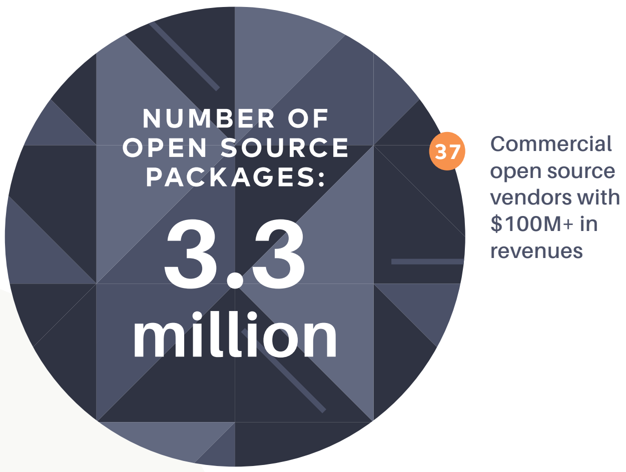
The pervasiveness of open source benefits professional developers in many ways. But reliance on open source components also raises some unique challenges. Do you have easy visibility into the security, licensing, and maintenance of these essential building blocks? Is your team able to find the time to ensure the open source components you are using are properly maintained?

These aren't necessarily new questions. Commercial open source services from companies like Red Hat, Cloudera, and MongoDB have been around for decades. Importantly, these subscriptions don't cover the vast majority of the open source libraries your business critically depends on.

Moving up the stack to the application layer is when the open source equation starts to get interesting. Here, developers select from over 3.3 million open source components from 37 different language-specific package managers.

It's clear that the "distro" model that worked for Linux and other system-level open source projects simply cannot scale to cover this massive breadth of open source software at the application level.

THE LIMITED REACH OF COMMERCIAL OPEN SOURCE SERVICES



Sources: $100M+ Revenue Commercial Open Source Software Company Index and Libraries.io

The hidden costs of open source

Because there are not commercial offerings available for most application-level open source software, professional development teams are left holding the bag on on maintenance, including integration, security, and license vetting. Making matters worse, only a small percentage of businesses approach these tasks systematically. Our recent survey tells us that only 9% of development teams have a formal process in place for introducing new dependencies. And almost a quarter of development teams make these decisions at the individual developer level.

This exposes teams to both big risks and major productivity drains. Free software starts to look a lot less free when you take the following costs into account:

Cost #1:

KEEPING UP

Development teams are stretched to the limit trying to keep up with constantly evolving open source dependencies. Valuable time is wasted assembling, integrating, and testing components to ensure they play well together.

Cost #2:

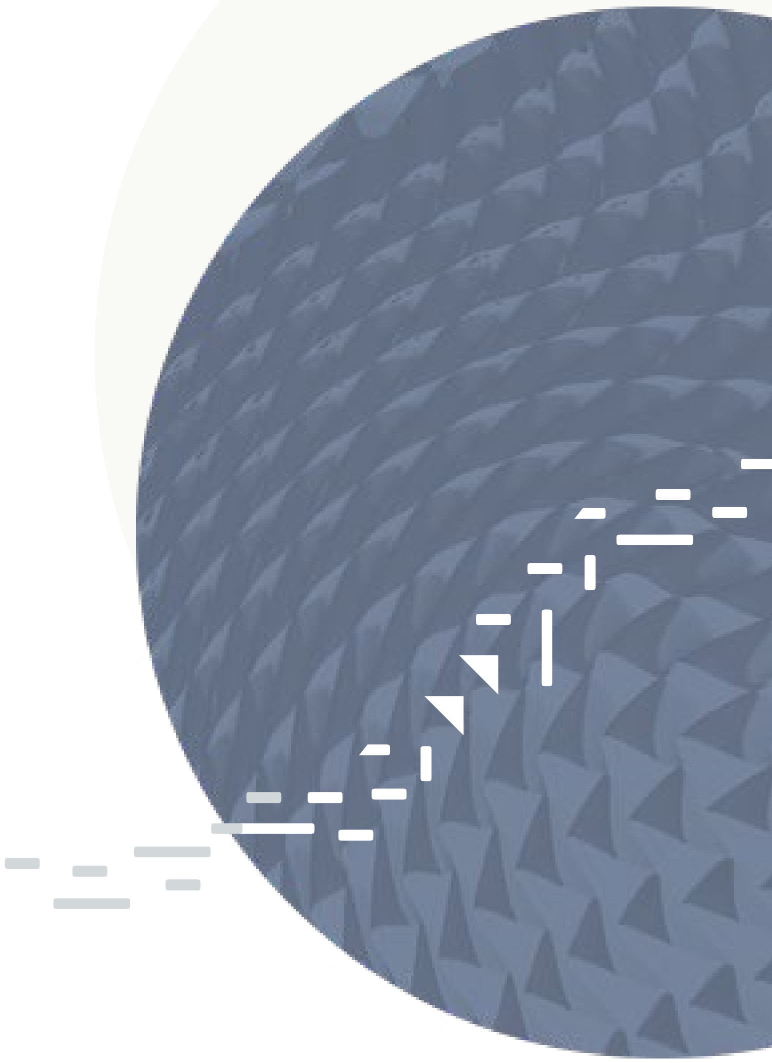
KEEPING IT SECURE

Development teams also take on the significant burden of policing the security status of the hundreds of individual components they integrate, or accept the risk of breaches like the one that impacted Equifax (and there are countless similar horror stories).

Cost #3:

KEEPING COMPLIANT

All of these great open source components come with a confusing array of different types of open source licenses. It's sometimes difficult to know if your organization's usage is in compliance with the license terms—not to mention how well the licenses for different components mesh with each other.

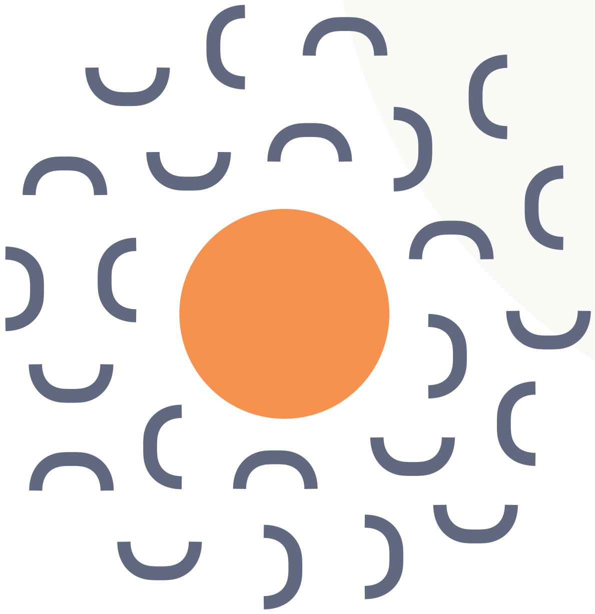


Getting your team back to shipping code

All of this raises a very simple question... what more could professional development teams achieve if they weren't spending so much time on these tasks above?

What could they achieve if they had a better way to ensure open source components worked well together? If they didn't have to spend the time proactively—or worse, reactively—addressing security issues. If they didn't have to worry about licensing issues, or whether software they want to use is even properly licensed at all?

Over the next few pages, we'll share a complete vision for a new way of thinking about using open source software in a professional development environment. This methodology can significantly impact the results an organization sees from its investments in open source.



WHAT YOU'LL LEARN IN THIS GUIDE

Our goals for this guide:

- Explore the issues professional development teams face when building building apps with open source software

- Show how Tidelift makes it easy to implement a three-step approach to proactively managing your organization's open source use through:

IDENTIFICATION

Helping you understand your current dependencies and the underlying maintenance, security, and licensing issues

INCIDENT PREVENTION

Helping you address issues before they start

RESOLUTION

Putting in place a managed open source program to protect your organization over the long term

Understanding open source software risk areas and opportunities

What is package management?

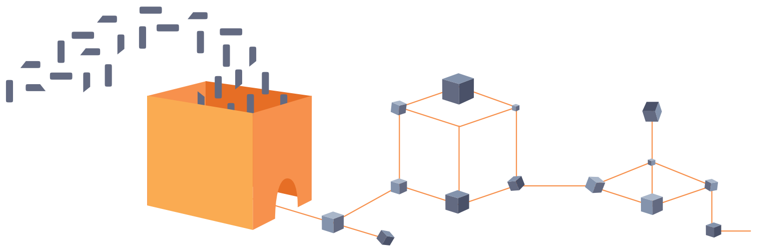
Open source software components or libraries are managed by tools called package managers. There is typically a distinct package manager for each programming language ecosystem or “stack.”

For example, the JavaScript/Node.js ecosystem mostly uses the npm package manager, the Java ecosystem predominantly uses Maven, and the Python ecosystem generally uses the Python Package Index (PyPI).

Open source software development typically happens on hosted source code collaboration platforms like GitHub, GitLab, and Bitbucket. Then it is assembled into package releases that go into the public package manager repositories.
To save time and make it easier to ensure the software is installed and runs properly, most commercial users of open source acquire their software from the package manager repositories, not directly from these code collaboration platforms.

WHAT IS A PACKAGE MANAGER?

A package manager is a set of software tools that automates the process of installing, upgrading, configuring, and removing computer programs in a consistent manner. Package managers help ensure everything is in place for software to run properly.



HOW DOES TIDELIFT HELP?

- ➜ Tidelift gives you a comprehensive view of these package managers and packages, across virtually every programming language and open source community.
- ➜ Tidelift has built the most complete index of open source packages in the world (with licensing and source information for over 3.3 million packages and counting!). We’ve done it by indexing more than 37 package manager ecosystems as well as GitHub, GitLab, and Bitbucket.

Think of it as a search engine for open source software. A way to find almost every available open source package and understand how they all relate to each other. Which brings us to...

Understanding dependencies in software projects

Software applications have always been built by assembling many small pieces of software, generally delivered as software packages—and alternatively called components or libraries.

When a component is added to your software application, your application depends on it to perform some function, and the component is now a *"dependency of"* your application. Each individual component may have many dependencies of its own, each of which may have many dependencies, and so on. The full set of software components forms a dependency tree. The full set of dependencies for a particular software project or application, including dependencies of dependencies and so on, are known as its transitive dependencies.

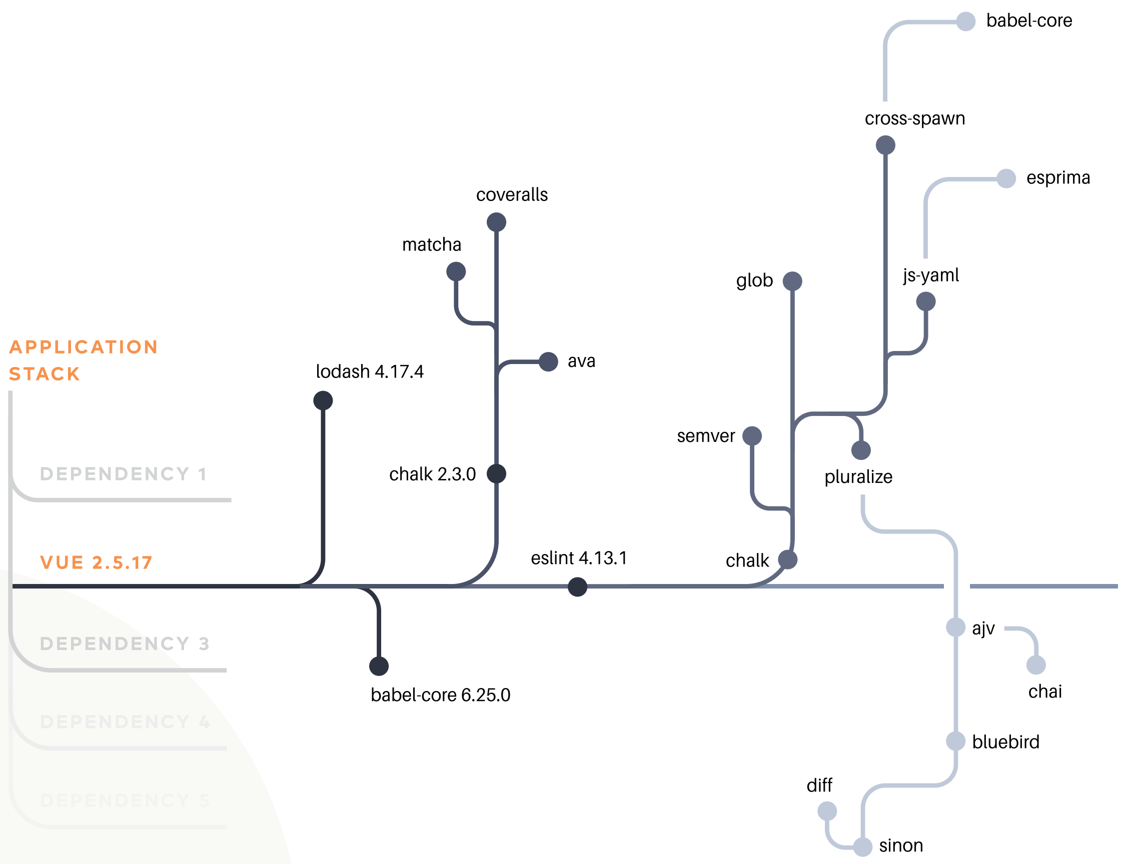
HOW MANY DEPENDENCIES?

Direct dependencies are those you've explicitly chosen to rely on; but direct dependencies in turn pull in other packages, called transitive dependencies.

A typical JavaScript application, for example, might have less than ten direct dependencies, but around a thousand transitive dependencies.

These transitive dependencies often haven't been reviewed by anyone on the application team. The quality of your application's dependency tree can mean the difference between an app that's a brick house or one that's a house of cards.

For example, the dependency tree for the JavaScript frontend package Vue is as follows:



HOW DOES TIDELIFT HELP?

Tidelift tracks and analyzes more than 500 million open source package dependency relationships.

Using this data, Tidelift connects to your software development process to understand the exact packages you use. We then pay the maintainers of those packages to ensure they are professionally and proactively maintained.

What's unique about Tidelift?

WE PAY THE MAINTAINERS.

Spoiler alert—today, maintaining most open source software projects does not pay well, unless that means a job at Google, Facebook, or other similar companies that support open source work. The impact of this for professional development teams is that most maintainers lack a key incentive they need to keep their packages well-maintained and secure.

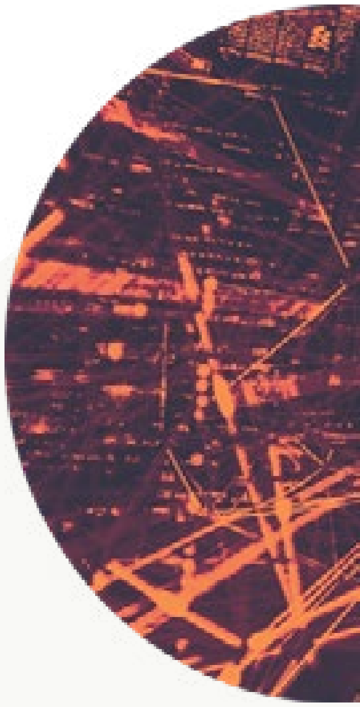
Tidelift is on a mission to change this, for the sake of both open source maintainers and professional development teams who incorporate the software components they create into their business applications.

We believe maintainers of popular open source projects both deserve to be paid and that these same maintainers are the best people to professionally maintain their projects.

Tidelift pays open source maintainers to stand behind their projects, providing necessary maintenance and updates—including for security and licensing issues—as part of the Tidelift Subscription.

With Tidelift, the days of scrambling to replace or update a key dependency when maintainers vanish or can't spend time on commercially important concerns are over.

Open source software is amazing, but it's still software



Each open source dependency that your application uses represents a distinct piece of software with its own license, release lifecycle, and security and maintenance practices.

It just takes one major issue in an open source package anywhere in the dependency tree for your application to cause a security breach that risks exposing customer data. Or a maintenance issue might result in a customer-facing systems outage. Or an issue with an open source project license might lead to an expensive and time-consuming intellectual property infringement claim against your organization.

The simple fact is that very few development teams have the time to inspect the thousands of open source projects their apps depend on for these kinds of issues. But not doing so—and not having an efficient way to do so—means some key changes may occur without you knowing, such as:

- A lead maintainer may lose interest, get a full-time job, or for some other reason stop working on an important dependency
- A dependency may change its license, or start using a transitive dependency with a problematic license
- A security vulnerability may emerge
- An update can break an API you need
- Maintainers may change the project roadmap in a way that impacts you

HOW DOES TIDELIFT HELP?

The Tidelift Subscription scales to monitor, fix, and proactively improve the multitudes of open source packages you work with every day. In the following section, we'll dive into issues around security, maintenance, and licensing—and demonstrate how Tidelift can help.

RISK AREA

Security

Open source software components, like all software, are susceptible to security vulnerabilities caused by programming oversights, or even intentional sabotage. Government-sponsored projects like Common Vulnerabilities and Exposures (CVE®) compile lists of common identifiers for publicly known cyber security vulnerabilities, including those in open source software.

However, very few commercial organizations have robust and effective processes in place to ensure that all of their software is continually assessed for known vulnerabilities. With the sheer number of open source packages out there, it's nearly impossible for developers to keep track of the security risks each may pose on their own.

One painfully familiar example is the Equifax exploit of 2017. While it's easy to point fingers at Equifax, it's also worth asking whether your own organization has the tools and processes in place to track vulnerabilities in all of your applications and their open source dependencies. As we've already seen, the number of dependencies in a single application can easily run into the hundreds.

"Equifax has confirmed that attackers entered its system in mid-May through a web-application vulnerability that had a patch available in March. In other words, the credit-reporting giant had more than two months to take precautions that would have defended the personal data of 143 million people from being exposed. It didn't."

—Wired Magazine, "Equifax Has No Excuse", September 2017

"Patching the security hole was labor intensive and difficult, in part because it involved downloading an updated version of Struts and then using it to rebuild all apps that used older, buggy Struts versions. Some websites may depend on dozens or even hundreds of such apps, which may be scattered across dozens of servers on multiple continents. Once rebuilt, the apps must be extensively tested before going into production to ensure they don't break key functions on the site."

—Ars Technica, Failure to patch two-month-old bug led to massive Equifax breach, September 2017

HOW DOES TIDELIFT HELP?

We work with and compensate open source maintainers to handle security issues responsibly. Specifically, Tidelift:

- Takes proactive steps to avoid future vulnerabilities and handle any new vulnerabilities
- Tells you about any vulnerabilities that affect your dependencies
- Provides information about past vulnerabilities and the affected version ranges

RISK AREA

Maintenance

At its essence, software maintenance means regularly adding features and fixing bugs. As an open source package gains popularity, the number of user requests for new features and fixes grows. Often, this flood of pull requests overtakes a part-time maintainer's ability to keep up.

Lapses in package maintenance often becomes a burden on the professional developers trying to use the packages.

The open source packages these professional developers use are updated and patched with varying frequency, sometimes requiring changes that break compatibility with other parts of the application stack. These updates and patches require a ripple of changes across the stack to keep everything running smoothly.

Bit rot and accumulating technical debt

Bit rot happens to all software when the dependencies and tooling required to build, test, and deploy it change over time. Eventually—when the software needs to be changed or redeployed—it cannot be returned to a functioning state because of conflicts with the changing ecosystem around it.

Software doesn't exist in a vacuum. Applications are built on top of hundreds, even thousands, of different pieces from open source frameworks and libraries. They're written in a range of programming languages, run on a variety of operating systems, and deployed to a vast array of hardware.

All of those components are updated and patched with varying frequency, sometimes requiring changes that break compatibility with other parts of your application stack. These updates and patches require a ripple of changes across the stack to keep everything running smoothly.

Sometimes you're in control of when those changes are applied—the version of the programming language you use, for example. Sometimes you're not so lucky. Even small updates can force you to make large, breaking changes across the application, just to keep things functional and secure. This affects projects that are under active development and those that have been quietly running away on a server for years, seemingly without issue.

WHAT IS BIT ROT?

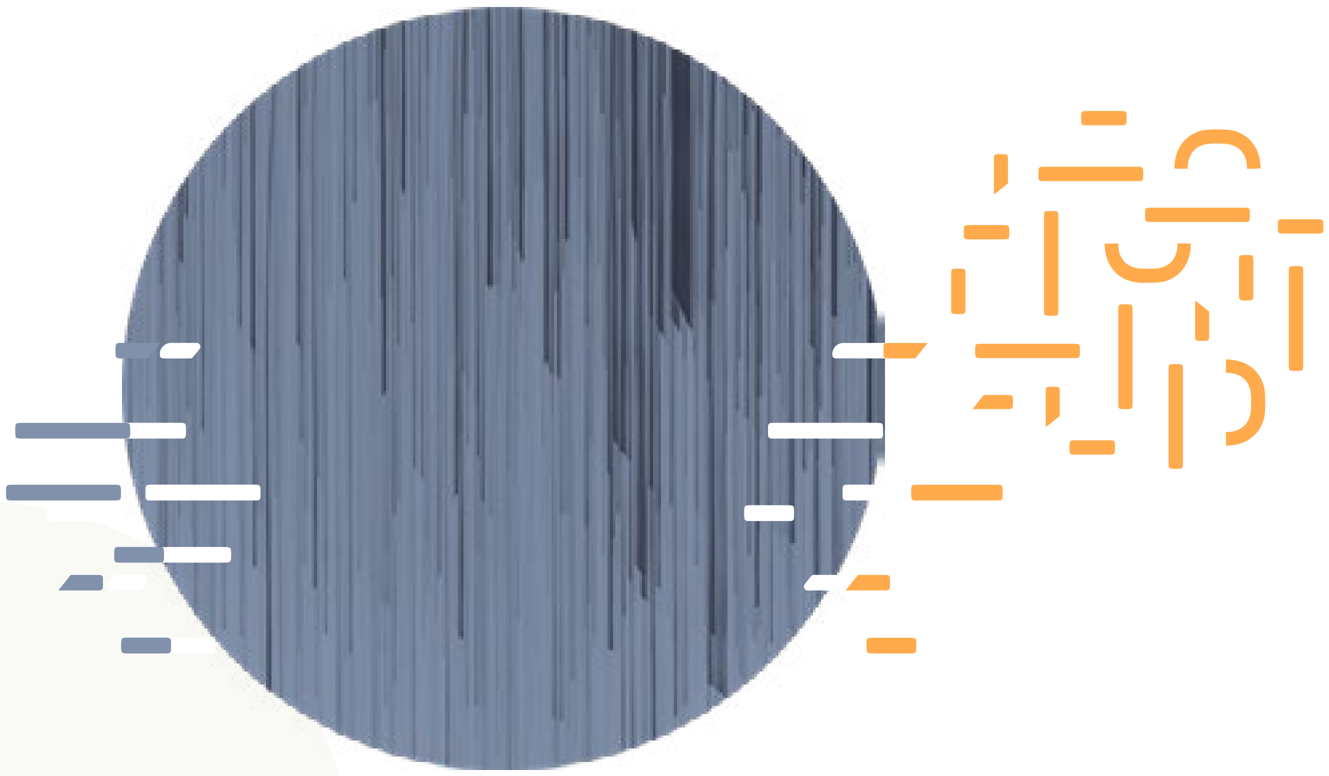
Bit rot is the deterioration of software performance over time, caused by changes to the dependencies and tooling required to build, test, and deploy it. Left unchecked, bit rot eventually makes the software prone to regular failures, poor performance, or otherwise becoming unusable.

Bit rot is often a death by a thousand cuts; each piece of software you depend upon can be susceptible to any number of changes that can cause your application to break. Some of the most regular causes are things like:

- Security releases that disable/change insecure interfaces
- Bug fix releases that inadvertently cause API changes
- Old versions being end-of-lifed and no longer tested for compatibility
- Incompatible breaking changes in major releases
- Conflicts within the dependency tree of your application
- Unrepeatable installation steps stopping you from reproducing a working environment (also known as onceability)
- Third-party or remote APIs changing or becoming unavailable without prior warning

Bit rot often goes hand in hand with the accumulation of technical debt. Pressure to ship product can result in short-term design and coding compromises. In the book *Refactoring: Improving the Design of Existing Code*, author Martin Fowler summarizes technical debt as follows:

"If you can get today's work done today, but you do it in such a way that you can't possibly get tomorrow's work done tomorrow, then you lose."

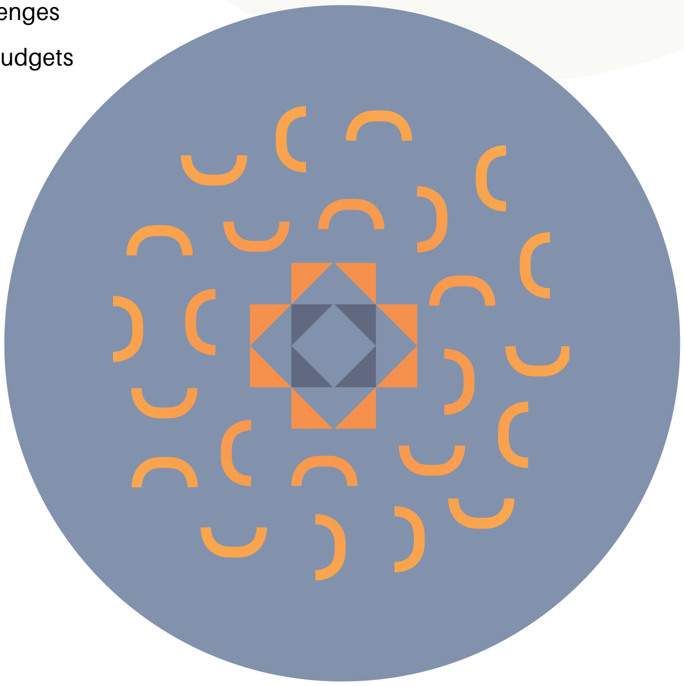


Too many such compromises result in technical debt. The more technical debt you accumulate, the more your application will feel the effects of bit rot.

The business impact of maintenance issues like bit rot and accumulating technical debt is that software teams waste incalculable amounts of time chasing and addressing software component compatibility issues (and they may not always be able to fix these issues at all), instead of solving their actual business problems .

The resulting business impacts are quiet but painfully real:

- Reduced developer productivity
- Expensive software team salaries wasted on trivial but time-consuming issues
- Developer frustration and resulting employee retention challenges
- Software projects that take longer to complete and overrun budgets



HOW DOES TIDELIFT HELP?

We work with and compensate open source maintainers to professionally maintain their software and make promises about its future dependability. Specifically, Tidelift:

- Works with open source maintainers to ensure they understand your issues and is your advocate to help get them fixed.
- Compensates maintainers for actively maintaining their packages according to a uniform set of best practices—looking at issues that are coming in, thinking about how to further develop the package over time, and generally keeping development on track
- Analyzes your dependencies to ensure maintenance issues are flagged so you can make changes if necessary
- Provides release notes for the projects you use through a "what's new in your dependencies" activity feed
- Helps you understand individual package stability assurances and branching schemes

Who are the open source maintainers?

MEET EVAN YOU.

Evan You didn't plan for his popular JavaScript framework, Vue.js, to become a full-time job. It just happened.

Evan came up with the idea for Vue after graduate school, when he worked at Google Creative Lab.

"We used Angular 1 in some projects," Evan said. "And there are parts of Angular 1 that I liked and there are parts that I felt I just don't really need. Vue started as an experimentation to extract some parts—[specifically, for] building a web application."

"We are trying to bring those good bits from a more monolithic complete full-stack framework," Evan said, "but we want to make those accessible to people who are building for simpler use cases."

Vue was a side project for two and a half years, from late 2013 to early 2016. Evan was working full-time and had to take three weeks vacation to release 1.0. Around that time, Vue got picked up by the Laravel community, when creator Taylor Otwell started using it.

"[When Otwell] started using Vue, the whole Laravel community went on board with him," Evan said. "We saw a spike in usage. That was around the time when I thought, 'Maybe there is something bigger than a side project in this.'"

When work at his day job started heading in a direction he wasn't interested in, Evan decided to quit and work on Vue full time. For maintainers like Evan, Tidelift provides a steady stream of income so they can make their open source projects even better.

RISK AREA

Licensing

Software licensing and intellectual property law are key "technologies" that enable open source software to exist in the first place. Every piece of open source software guarantees you the ability to use it, but also has at least some minimal restrictions on how it can be used, and possibly attribution requirements as well. Around 10% of packages have even stronger restrictions, usually called "copyleft."

However, especially in large software projects with many dependencies, open source license compliance can become a major complicating factor.

WHAT IS A COPYLEFT LICENSE?

"Copyleft" or "reciprocal" licenses require sharing of modifications under certain conditions. Examples include the well-known GNU General Public License and a spectrum of others, including the "network" Affero GPL (whose conditions may be triggered by use in services) and a variety of "weak" copylefts like the Eclipse and Mozilla licenses (whose conditions generally require sharing of fewer classes of changes).

"Open source and licenses can be a massive headache even for companies with massive resources at their disposal, given so many different licenses can be incompatible with each other. Just trying to get accurate and complete license information for anything can be a time-consuming headache especially on a large scale. At the same time, companies also need to better understand the issues around complying with open source licenses."

—The New Stack, SPDX Could Help Organizations Better Manage Their Thickets of Open Source Licenses

Often, individual software developers are tasked with making critical decisions about the software licenses of their open source components—a complex and nuanced area of expertise for which they are not trained.

The resulting business impacts can be painful, and include:

- Exposure to litigation by third parties over breach of license or copyright
- Expensive and unscheduled costs when components must be removed late in the software development process to remediate licensing issues
- Increases in customer service and support costs and decreased development speed associated with changing a product that is already deployed
- Delay or outright cancellation of financings, mergers and acquisitions, initial public offerings, and other corporate transactions
- A legal injunction against continuing to distribute a product while it has an infringement claim outstanding

"Compliance is also a bit of a headache for companies that are not adept at dealing with open source. Ensuring that your company distributes the source and complies fully with the GPL, including making sure that if the GPL is combined with other works that it is license-compatible, can require a fair amount of attention."

— CIO Magazine, How Open Source Licenses Affect Your Business and Your Developers

HOW DOES TIDELIFT HELP?

We work with and compensate open source maintainers to help subscribers avoid intellectual property and licensing issues. Specifically, Tidelift:

- Ensures all packages that are part of the Tidelift Subscription operate under an approved open source license
- Cleans up license metadata, ensuring that it is consistent and accurate for each package
- Fixes license violations
- Helps you understand if you have dependencies with restrictive licenses that may cause issues in typical professional development use cases
- Gets assurances from key maintainers that they did not violate the rights of others, and that you'll have a window to remedy inadvertent mistakes

Assess your organization’s current use of open source

Once you understand the issues involved with using open source in a professional development environment, the next step is to get a comprehensive view of your organization’s open source usage by mapping your dependency trees. While this was once a near-Herculean task, it is now much easier thanks to Tidelift. Here’s how to get started:

Get an open source assessment

With support for Javascript, Java, Python, PHP, and 20 more languages and package managers, our open source software assessment will give you a unified view of all open source components your organization is already using.

The Tidelift open source assessment uncovers important information about your stack, including:

- ➜ Deprecated and unmaintained packages
- ➜ Missing licenses
- ➜ Security vulnerabilities
- ➜ Direct and transitive dependency trees

To schedule yours, visit: https://tidelift.com/subscription/request-an-ossa

Developing a plan

Now that you have a comprehensive list of the open source components your organization uses, and a clearer view of any current issues, it's time to come up with a plan to address them.

One path forward is to start trying to remediate issues by upgrading individual packages, contributing your own pull requests to upstream open source packages, or removing problematic dependencies from your application and choosing other packages, or implementing similar functionality from scratch yourself. Then, keep an eye on the dashboard and rinse, repeat.

But at Tidelift, we provide a more comprehensive—and scalable—model. When you purchase the Tidelift Subscription, you can rely on expert maintainers partnered with Tidelift to do that work for you, in a professional manner—fixing issues that are visible today, and proactively avoiding and resolving issues that come up in the future. We recommend thinking about the Tidelift Subscription as a way to begin implementing a comprehensive, organization-wide open source management plan.

At Tidelift, we're working to provide a more comprehensive—and scalable— model.

Implement a professional open source management plan with Tidelift

Most organizations have a software vendor management program to protect them from risk and ensure their software is properly maintained. Often enforced by procurement or legal, vendor management programs ensure that software used within the organization is dependable.

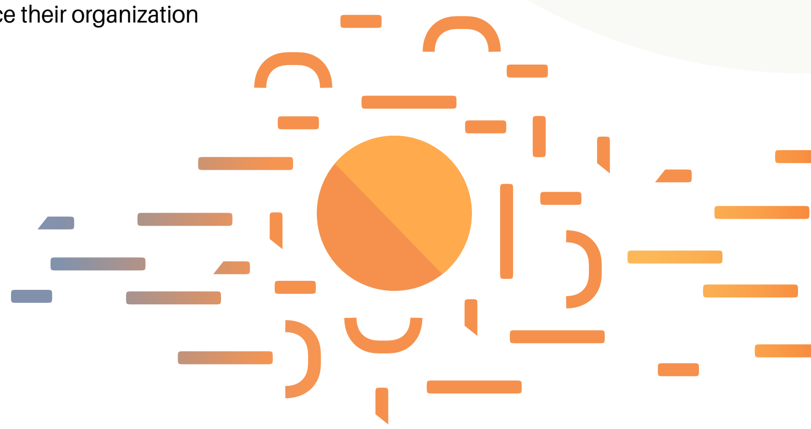
A vendor management program requires software vendors to provide representations about issues that are discovered or may arise in the future—and put in place formal means to correct them.

But who provides representations for open source software?

The emergence of millions of high-quality open source components has outpaced organizations' ability to keep track of their use of this software. Like all software, open source components require maintenance and ideally would include additional representations about the security, dependability, and intellectual property ownership of the software.

Historically there has typically been no one to provide these assurances for the vast majority of open source components that professional development teams use daily.

This is a missed opportunity that saddles professional development teams with long-term maintenance and upkeep headaches, as well as vulnerability to lots of bad outcomes. With over 3.3 million open source packages out there, the last thing any organization wants to do is boil the ocean. Rather, professional development teams should focus just on the the open source their organization depends on.



HOW DOES TIDELIFT HELP?

The Tidelift Subscription makes it simple for organizations to easily implement a professional approach to managing open source software, making it more secure and more dependable.

- ➜ We bring together the maintainers of the exact components you are already using, and continue to adapt as you add and remove components over time.
- ➜ We work with maintainers to establish professional-grade maintenance, security, and licensing standards and processes as part of the Tidelift Subscription.
- ➜ Through the Tidelift Subscription, we provide your organization with the same representations of dependability you would expect of any other software vendor.
- ➜ Your team engages with one vendor—saving time and money. And you get broad coverage for your open source dependencies.

Who are the open source maintainers?

MEET OLIVIER TASSINARI.

Olivier discovered Material-UI in early 2015 when he was a student.

Material-UI was about two months old at that point. Olivier chose it hoping it could save him some time while building his app. He also thought he could make some contributions to a new library.

The creators of Material-UI soon became overwhelmed by the popularity, so they asked Olivier to take over handling pull requests.

Once Olivier graduated from school, he had to decide whether to focus solely on work, or continue contributing to Material-UI at the same rate he had as a student.

One interesting dynamic of Olivier working a full-time job while moonlighting as a core maintainer came to light when he started using Material-UI on a project at his day job.

“I started having real professional needs from the project,” Olivier said. “I see all the pain points and what professional users will expect. This is one more reason why I am so dedicated to the project.”

One pain point is people helping on a project for a few months, then moving to something else. He thinks that’s where money can help things.

It’s important, Olivier reckons, that those who created the project stick around. Projects are built around the creator’s vision. When ownership shifts, the vision shifts—and that shift is “not always for the best, it’s a risk with an unknown outcome,” he said.

Tidelift is making it economically rewarding for project creators—and maintainers like Olivier—to continue investing their time in their open source software projects, so that professional users can rely on them well into the future.

Incorporate the Tidelift Subscription into your software development process

With the Tidelift Subscription in place, professional software developers can now get back to spending time on building their apps instead of wrangling dependencies.

We're continuing to add coverage for more parts of the open source landscape all the time. Powered by the Tidelift Subscription, professional developers can monitor their open source dependencies, and receive alerts when the subscription adds coverage for packages they use.

It’s time to stop wasting time wrangling open source dependencies and accelerate application development

Open source has fundamentally changed software development for the better. It allows us to increase development velocity and improve our world along the way.

Our mission at Tidelift is to make open source work better—for everyone. If this approach to using and managing open source software sounds right for your business and you’d like to learn more:

- ➜ Contact us to learn more about the Tidelift Subscription, to sign up for an open source assessment, or to request a demo
- ➜ Visit our website to learn more and sign up for updates