The 2019 Tidelift managed open source survey results

EIGHT KEY FINDINGS ILLUSTRATING HOW TO MAKE OPEN SOURCE WORK EVEN BETTER FOR DEVELOPERS

September 2019

TIDELIFT

INTRODUCTION

In June of 2019, Tidelift and The New Stack jointly fielded a survey of professional software developers. Almost 400 people responded with thoughts about how they use open source software today, what holds them back, and what tools and strategies would help them use it even more effectively.

Software developers and engineers account for 51% of respondents, with DevOps pros an additional 12%. Company size was well represented, with 34% of respondents working at a company with up to 50 employees and 25% at companies with more than 1,000 employees.

In particular, with this survey we were interested in learning how a managed open source strategy might help developers reclaim time, speed up development, and reduce risk. In this report, we'll share eight of the most interesting findings that illustrate ways to make open source work even better for developers.

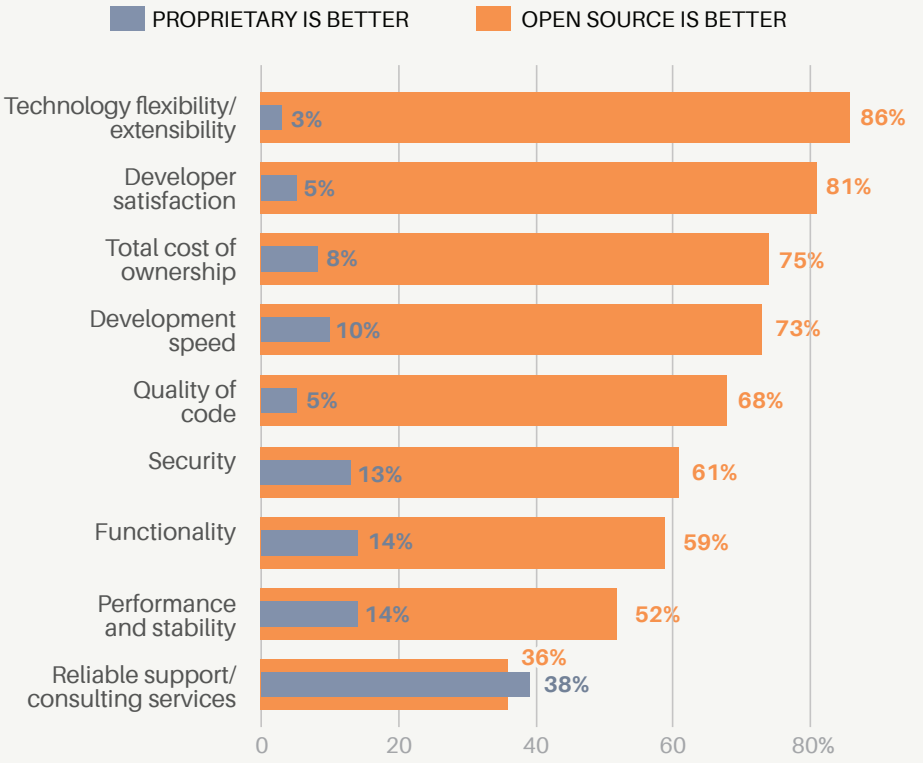
FINDING #1

Despite its overwhelming advantages, open source adoption is inhibited by concerns about the availability of reliable support

In a previous survey, we learned just how pervasively open source software is used by professional developers today, with 92% of applications using open source components. Open source has surely become the default stack for modern developers.

In this survey, developers compared open source and proprietary software in a number of different areas. The results roundly confirm why developers prefer open source. In every area except one—reliable support and consulting services—open source came out ahead of proprietary software.

Comparing open source and proprietary software



BASED ON 376 RESPONSES

According to respondents, the number one benefit of open source is technology flexibility and extensibility, with an astounding 86% agreeing that open source is better. This is followed closely by developer satisfaction, where 81% say that open source is their choice.

Open source also is preferred by significant margins when it comes to total cost of ownership (75% for open source to 8% for proprietary), development speed (73% for open source to 10% for proprietary), quality of code (68% for open source to 5% for proprietary), security (61% for open source to 13% for proprietary), and functionality (59% for open source to 14% for proprietary).

On performance and stability, the results are more mixed. While only 14% see proprietary as better (compared to 52% for open source), a full 30% view the two as about the same.

The only area where proprietary software is seen as slightly better is in the availability of reliable support and consulting services. Nearly 40% of respondents report proprietary software as stronger in this category, versus 36% for open source.

Given the vast advantages of open source in all other areas the survey explored, closing this "support gap" by creating open source support and consulting services on par with the technology's perceived quality represents the last hurdle before open source completely eclipses proprietary software.

Fortunately, efforts like the movement toward managed open source are narrowing the gap by combining the best elements of the proprietary model—like support and maintenance under a contractual service level agreement—with all of the existing benefits of open source. The industry is making progress, but these results show there is still more work to do!

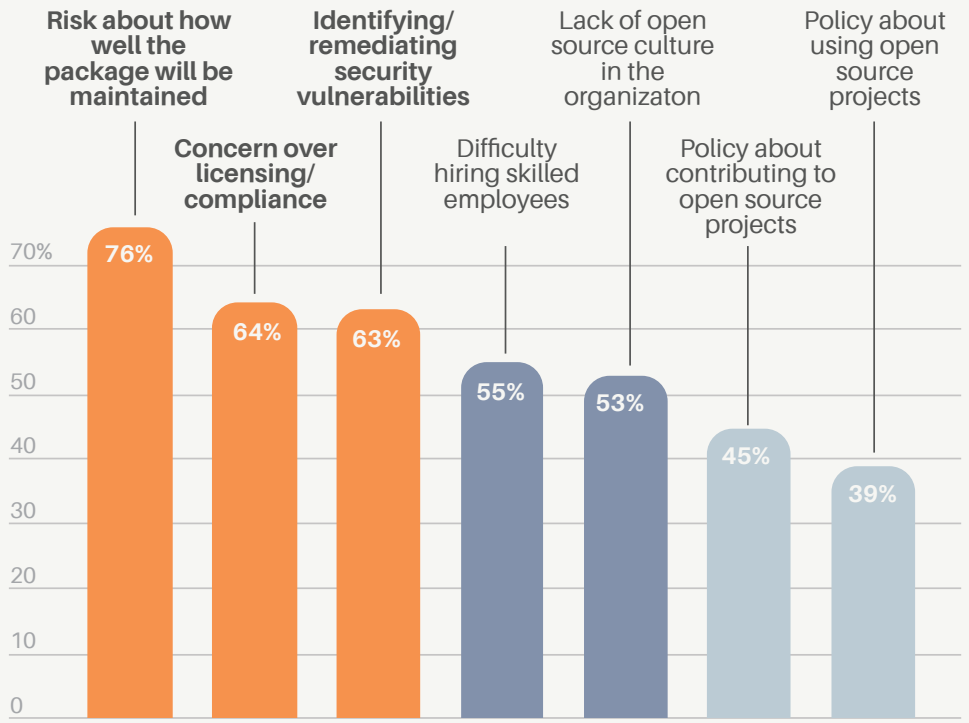
The only area where proprietary software is seen as slightly better is in the availability of reliable support and consulting services.

FINDING #2

Developers' biggest concern with open source is risk regarding how well projects will be maintained into the future

We asked developers to what degree a set of commonly described issues prevents them from expanding the use of open source in their organizations. Of the issues we highlighted, the most pressing fall into the categories of maintenance, security, culture, and licensing.

Obstacles to increasing use of open source



BASED ON 320 RESPONSES

Perhaps the clearest finding from this question is that 46% of respondents indicated risk about how well packages will be maintained into the future is either a major or a moderate obstacle, with another 30% indicating that it is a minor obstacle. Only 18% said that this is not an obstacle for them at all.

The next most pressing issue stopping developers from expanding their use of open source is concern about identifying and remediating security vulnerabilities. More than one-third, or 37% of respondents, indicated that this is either a major or moderate obstacle, with another 25% considering it a minor obstacle.

The findings regarding issues with licensing and compliance match the results from our previous surveys, where some organizations view this as a critical obstacle, while others see it as minor. This issue comes in second to maintenance risk as an identified obstacle, but 30% of respondents only consider it a minor obstacle, so just not as urgent as many as the other issues highlighted above.

The most mixed response is around the issue of open source culture in the organization. Many people—43% of respondents—do not see this as an issue at all, but 53% think it's an obstacle to some degree. This finding shows that many organizations have embraced an open source-friendly culture and are reaping the benefits, while others are left behind.

The findings from this question highlight the opportunity for open source to develop new strategies like managed open source to reduce maintenance, security, and licensing risk. Companies such as Red Hat, Elastic, and Cloudera recognized long ago that the way to help businesses deploy open source technologies successfully was to make promises about the future that were worth paying for in these three areas. Making these same assurances available not just for operating systems and data stores—but for the vast array of open source components used by application developers—represents one of the most promising opportunities for commercial open source.

46% of respondents indicated risk about how well packages will be maintained into the future is either a major or a moderate obstacle, with another 30% indicating that it is a minor obstacle.

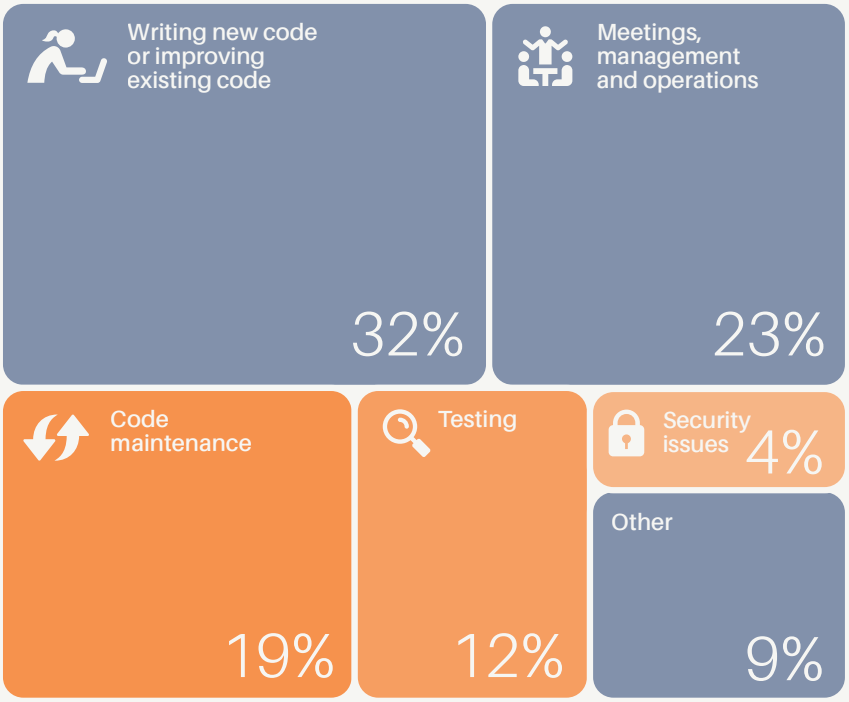
FINDING #3

Developers spend more time maintaining, testing, and securing existing code than they do writing or improving code

We wanted to use this survey to get a detailed view of how developers spend their time.

We gave them six categories in which to bucket their time and asked them to estimate the percentage of their work invested in each category.

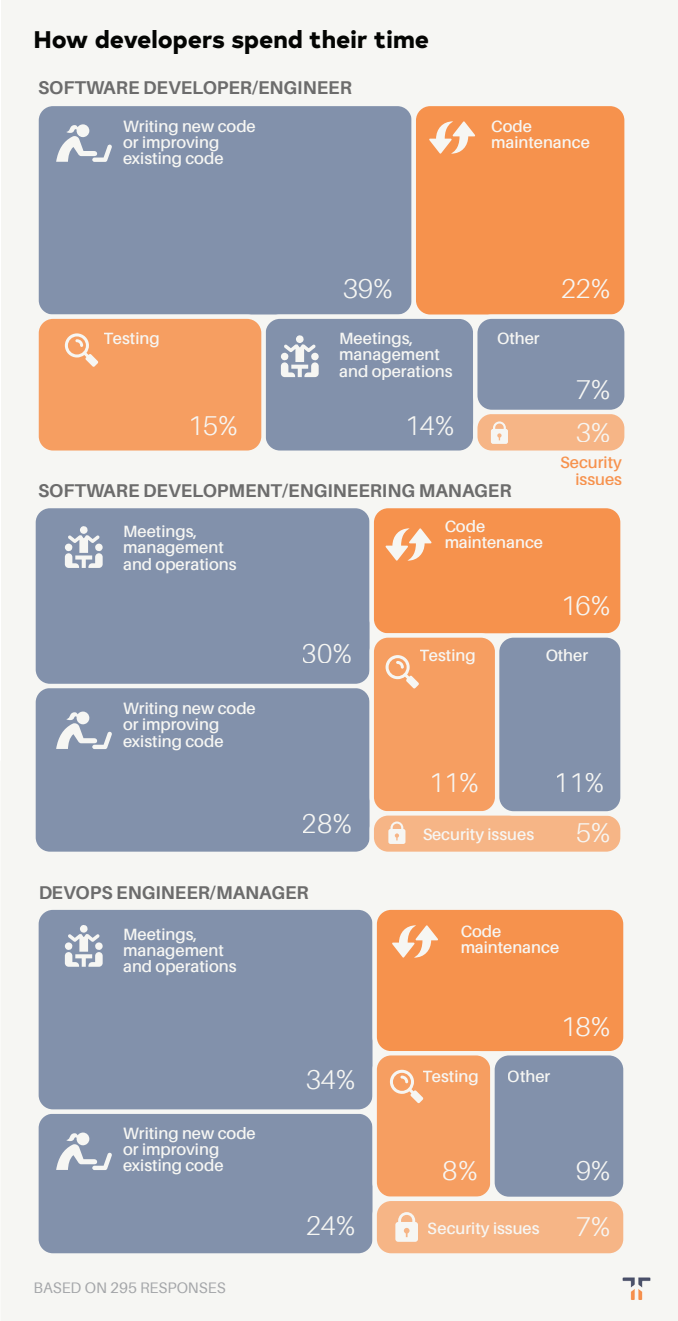
How developers spend their time



BASED ON 295 RESPONSES

While this might not be surprising to developers, it is perhaps disheartening to see that respondents spend less than one-third of their time writing new code or improving existing code (32%). Respondents spend 35% of their time managing code, including code maintenance (19%), testing (12%), and responding to security issues (4%). Another 23% is spent in meetings and on management and operational tasks.

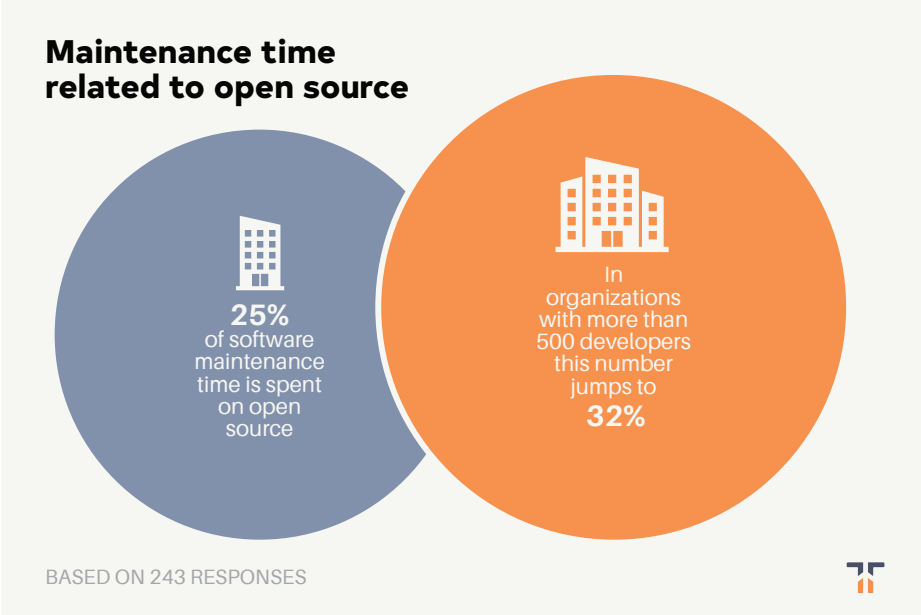
Breaking the data down by job description gives us an even clearer view. Software developers spend 22% of their time just doing code maintenance. They also spend a higher percentage of their time writing new code or improving existing code (39%) and a much lower percentage of their time on operational tasks and in meetings (14%).



Respondents spend 35% of their time managing code, including code maintenance (19%), testing (12%), and responding to security issues (4%).

Not surprisingly, people who manage software developers spend twice as much time in meetings as do the people they supervise. DevOps engineers and managers spend even more of their time in meetings (34%), partly because they are facilitating communication between different teams. They also spend twice as much time (7%) responding to security issues, which will be of no surprise to those familiar with the DevSecOps trend.

We also asked respondents to share the percentage of the time they spend on code maintenance related to their open source dependencies. The answer is right on target with the results in a previous survey (25%). But once we look at the data by number of developers in the organization, it presents an even starker picture. In organizations with over 500 developers, the percentage of time devoted to maintenance activities rises to 32%, which might be due to maintenance issues becoming more complex as the codebase and applications get larger.

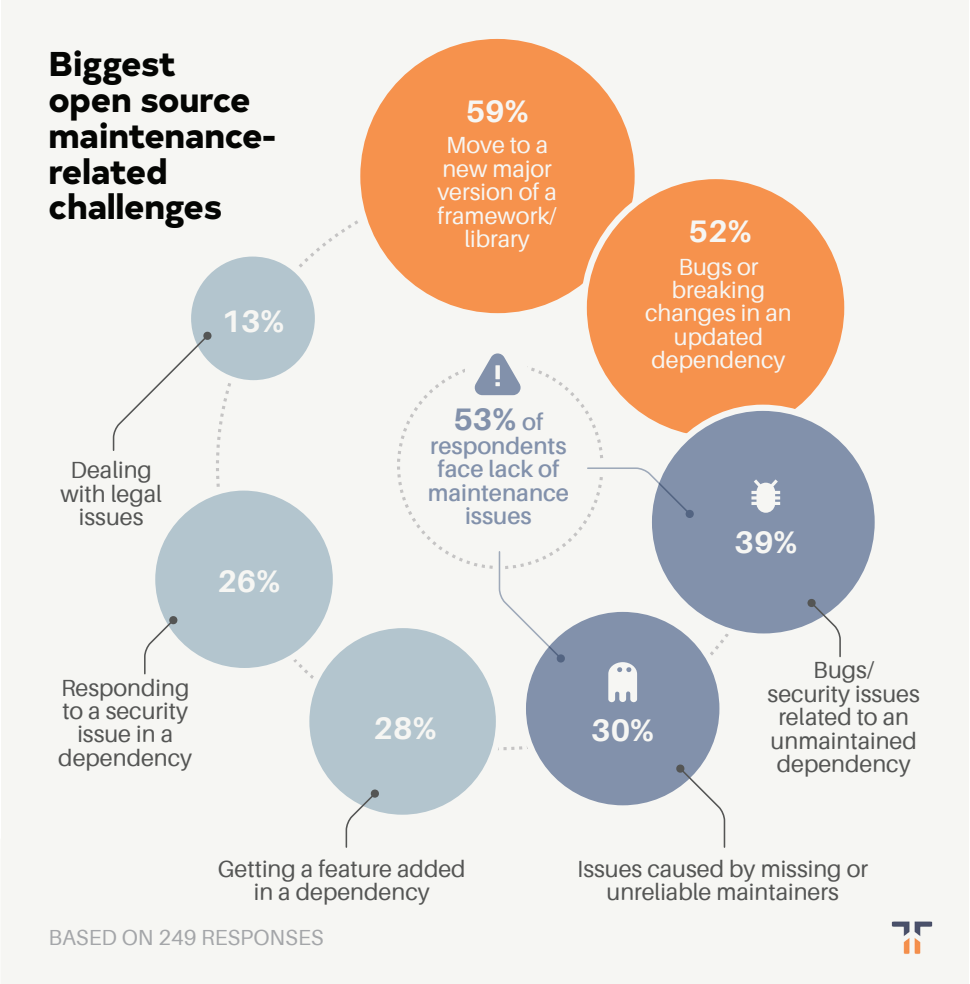


This data makes one thing very clear: there is a huge opportunity for organizations to find new ways to increase the percentage of time their developers spend writing code. What more can be done to make developers more efficient so they can spend less time on activities like code maintenance?

FINDING #4

Time-consuming code maintenance activities create challenges, often due to issues associated with poorly maintained open source dependencies

The survey shows that code maintenance activities like refactoring, debugging, rewriting functionality, and fixing technical debt take up about one-fifth of respondents' work week. We wanted to break this time down a bit so we could better understand exactly what sorts of maintenance activities caused the greatest challenge for developers. For this question, respondents could select all of the maintenance challenges that applied to their organization.



The number one maintenance challenge, cited by almost 60% of respondents, is moving to a new version of an open source library or framework. This is followed closely by adapting to bugs or breaking changes in an updated dependency, which 52% of respondents cited. But unlike moving to a new major version, this maintenance challenge can sometimes have even more urgency, since it isn't always an activity that developers can anticipate.

After these first two common challenges, there is a bit of a dropoff before the next two, which both relate to either unmaintained open source packages or unreliable maintainers. Looking at the data more closely, if we analyze how many people selected one of these two choices, more than half (53%) of respondents report dealing with at least one of these issues related to unmaintained open source code.

Less common issues include getting a feature added in a dependency (faced by 28% of respondents), responding to a security issue in a dependency (26%, meaning it isn't the biggest draw on time, but probably critical when it does happen), and providing the legal team with licensing information or responding to legal-related questions (13%). Legal challenges are much more common in companies with more than 1,000 employees, with double the percentage of respondents (27%) highlighting this as an issue.

So the maintenance issues that steal time from developers—time that they'd probably rather spend writing or improving code—usually stem from either moving to a new major version of a framework, adapting to bugs or breaking changes caused by an updated dependency (including the dreaded dependency hell!), and dealing with issues related to an unmaintained or under-maintained dependency.

It begs the question: Could we better solve some of these issues for developers by taking the maintenance tasks off of their plate? And at the same time create the incentives that lead to a world with fewer unmaintained or under-maintained dependencies causing security issues and other headaches?

The number one maintenance challenge, cited by almost 60% of respondents, is moving to a new version of an open source library or framework.

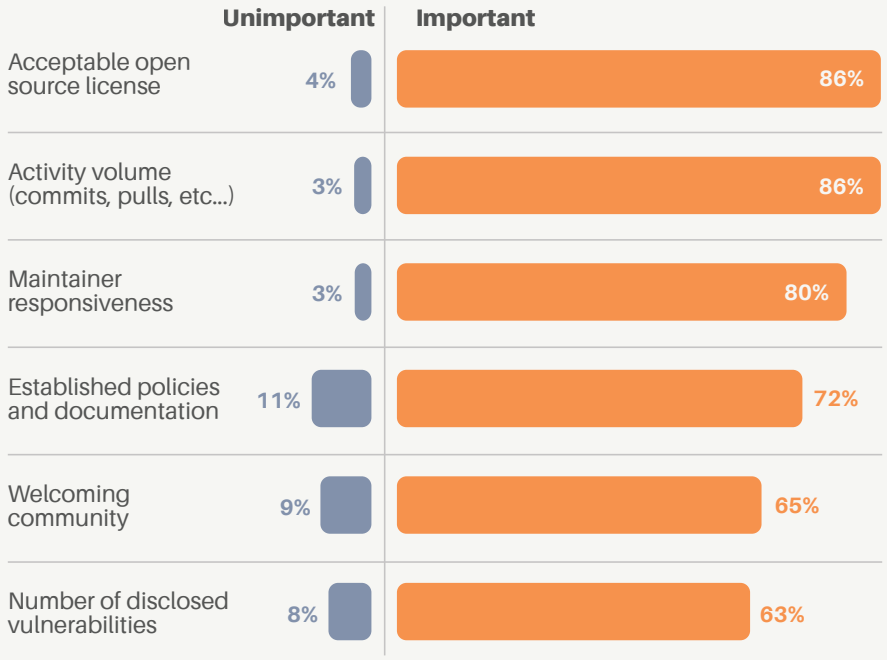
FINDING #5

Project activity, licensing, and maintainer responsiveness are key factors in choosing open source projects

Given the maintenance headaches respondents described in earlier findings, one obvious way to avoid them is to make good package choices in the first place. In the next part of our survey, we asked developers to tell us more about how they make decisions regarding which open source packages to use.

The first question on the subject asked how important some key project characteristics are when developers select packages.

Key factors when choosing open source packages



BASED ON 320 RESPONSES

When it comes to choosing packages, licensing is the most crucial issue: 61% of respondents said having an acceptable software license is extremely important. An additional 25% report licensing is somewhat important, for a total of 86% rating open source licensing as either extremely or somewhat important. Only 4% of respondents don't see this as an important issue.

This is particularly meaningful for companies with more than 1,000 employees, where 78% of respondents say having an acceptable open source license is extremely important. These findings clearly show that there are some "dealbreaker" licenses out there that most users attempt to avoid.

While activity (e.g., recent and volume of issues, commits, and pull requests) tied with licensing in overall importance (86%), it had a lower percentage rating extremely important (43%) than licensing.

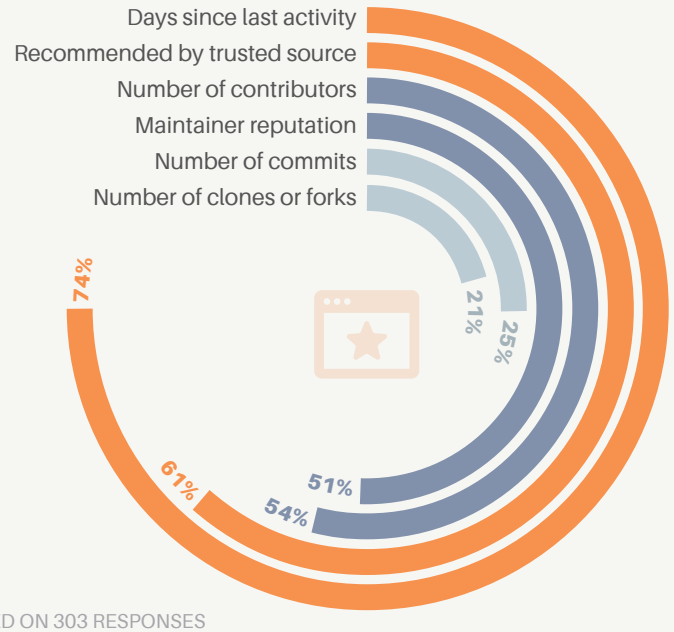
Also important when choosing open source packages is maintainer responsiveness, which 80% of respondents view as either extremely important or somewhat important when selecting an open source package to use. This is followed by established policies and documentation (e.g., code of conduct, contributing guide), with 72% of respondents rating this characteristic as important.

The last two items in order of importance were having a welcoming community (65% rated as important) and number of disclosed vulnerabilities (63%). While neither of these ranked as high as the other options, it is interesting to note that almost two-thirds of respondents still view these as key factors to look at when choosing open source projects.

We wanted to dive deeper into how developers analyze project activity. So for those who rated this as a key characteristic, we followed up by asking them which of the following activity metrics are most important to them.

When it comes to choosing packages, licensing is the most crucial issue: 61% of respondents said having an acceptable software license is extremely important. An additional 25% report licensing is somewhat important, for a total of 86% rating open source licensing as either extremely or somewhat important.

Most important metrics for evaluating open source projects

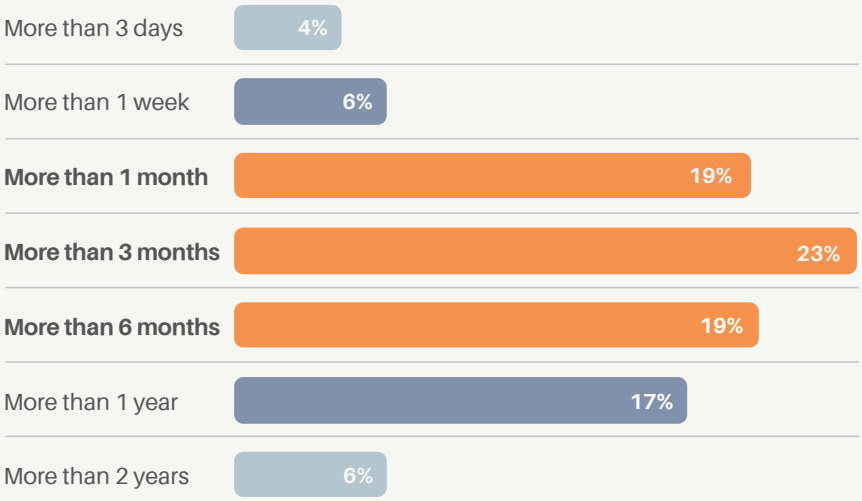


BASED ON 303 RESPONSES

Overall, 74% consider the number of days since last activity when deciding to use an open source project. While this is a low bar for respondents to think a project is in an active state, it does quickly eliminate many projects.

Another followup question asked what length of time since the last activity (e.g., commit, issue) would concern them when evaluating the health of an open source project. The results plotted in an even distribution curve, with the largest contingent of respondents falling into the "more than three months" category.

Length of inactivity considered concerning when evaluating an open source project



BASED ON 302 RESPONSES

Other important factors respondents consider when evaluating an open source project include being recommended by someone they respect (61%), number of contributors (54%), and the maintainer(s) having a good reputation (51%).

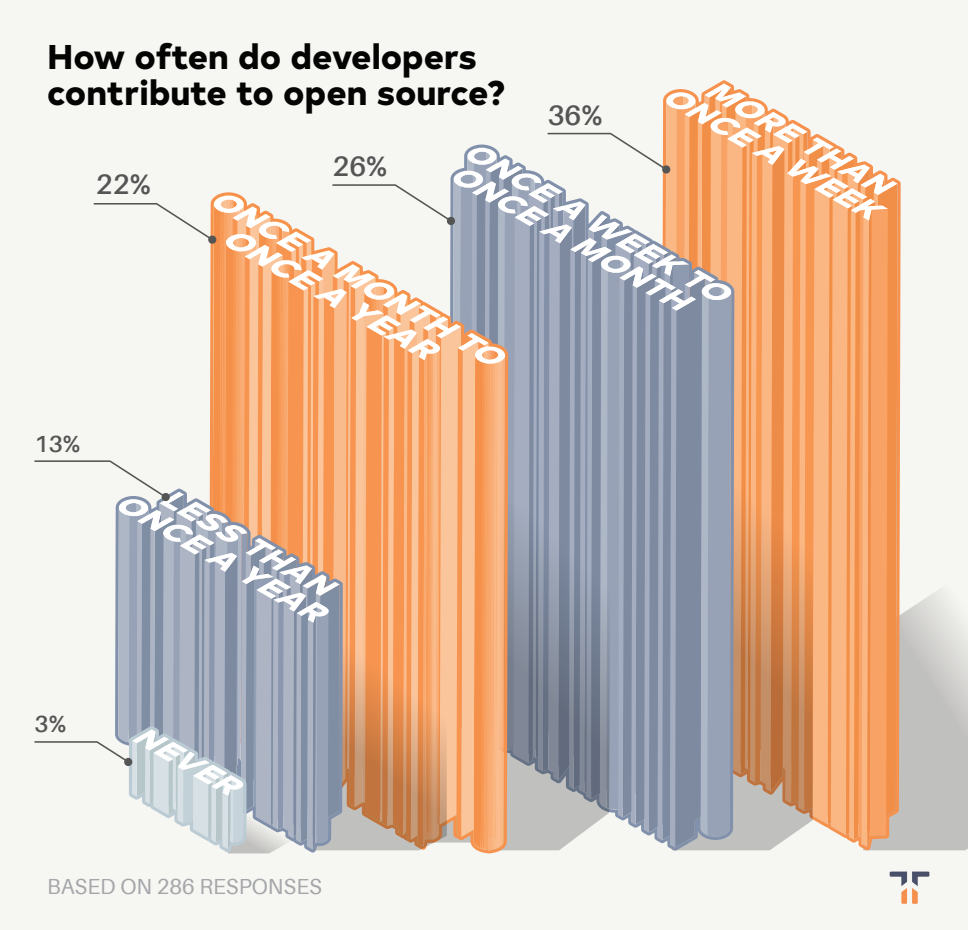
Overall, these data points show that developers put a lot of thought into how they choose their open source packages, knowing that good choices will help them avoid maintenance, security, and licensing headaches down the road.

FINDING #6

Most developers contribute to open source projects, and many would do even more if fairly compensated for their work

We also wanted to learn how developers contribute to open source projects themselves. First, we asked respondents on average how much time they devote to open source contributions.

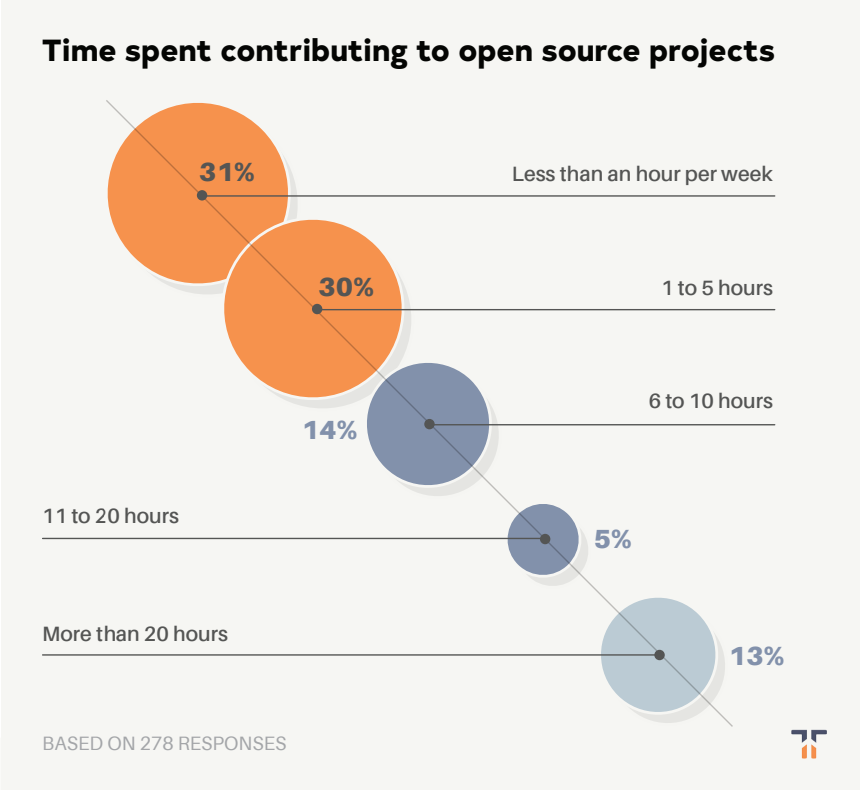
We were surprised to learn that most respondents (84%) view themselves as active contributors, with only 13% saying that they contribute to an open source project less than once a year and just 3% saying they never contribute to an open source project.



This high percentage is the result of sample bias—our initial qualifying question asked whether respondents use open source software to build applications at work, so all respondents are open source users already.

We still believe it is an interesting and useful finding just within the subset of developers who use open source to build applications at work. It shows that there isn't a rigid line between users and creators of open source, but instead perhaps more of a continuum, where most developers are both creators and users, with some falling closer to one end or the other of the user-creator spectrum.

We next asked how much time they contribute to open source on a weekly basis. As you might expect, results vary widely. The average respondent spends over six hours a week contributing, although the majority spend less time than that.



There isn't a rigid line between users and creators of open source, but instead perhaps more of a continuum, where most developers are both creators and users, with some falling closer to one end or the other of the user-creator spectrum.

Several other questions delve into this subject for more detail. Four-fifths of respondents would spend more time contributing outside of their day job if they were fairly compensated for their work, with 25% estimating they would spend an additional 20+ hours per week. Clearly there is an appetite among contributors to work more on open source if there was a better way for them to be compensated.

When asked roughly what percentage of their time contributing to open source was done as part of their job, the average response was 40%. People who contribute less than that amount reported that they would be more likely to spend a higher percentage of their time outside of their jobs participating in open source projects if they were fairly compensated.

This shows that developers who contribute to open source, without it being a significant part of their day job, might step up their open source work if the right incentives were in place. We estimate that the hours contributed per week to open source would increase by over 40% if respondents were fairly compensated for their non-job-related efforts.

What kind of conclusions can we draw from this data? Right now, many of the people writing and maintaining open source components are doing this work without being compensated. Yet an astounding 84% of developers using open source in their professional work consider themselves to be active open source contributors, at least to the level of more than one contribution a year—and fully one-third contribute code more than once a week. If there was a better, more consistent way to pay open source contributors for the work they do, would more of these developers become project creators and maintainers?

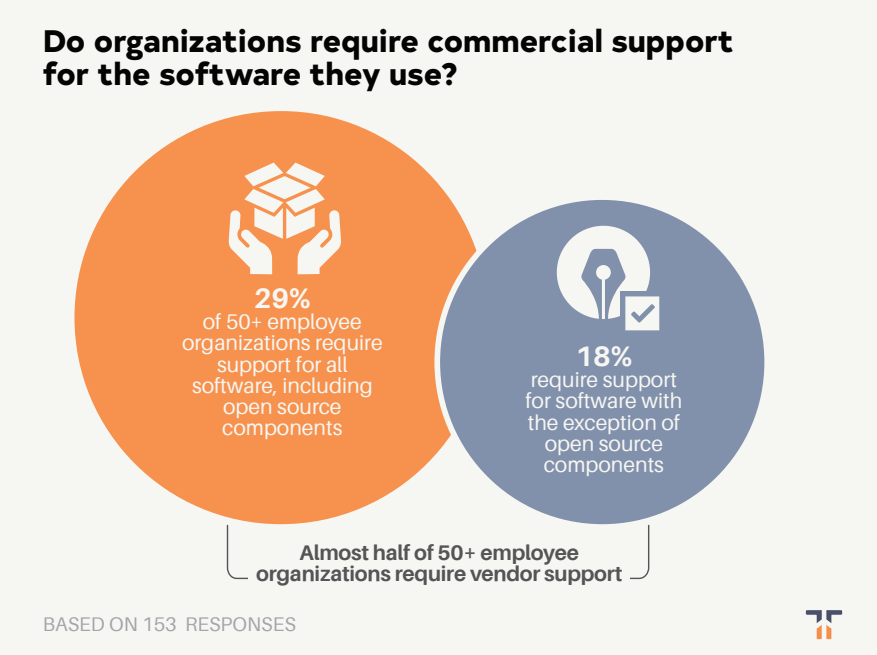
We believe so. Fortunately this is a problem that the managed open source model is well positioned to solve—putting the proper incentives in place for creators and maintainers of open source software to do their best work—and maybe even attracting a new class of creators and maintainers to open source in the process!

FINDING #7

Almost half of organizations have policies requiring commercial support for the software they use

It's the age-old managerial objection to using open source: "But who are we going to call when something breaks?" While in the early days of open source, this was a common refrain with the suit-and-tie crowd. Now, thanks to companies like Red Hat, Elastic, and Cloudera, which provide enterprise-class support and assurances for open source, this objection has become much less common with some heavily used open source projects.

Yet, for the vast majority of open source components used to build applications, there is no commercial organization backing them up. We wanted to learn more about how larger organizations handle this conundrum. Are they required to have a vendor backing up all of their open source dependencies? Or do they self-support and take on the risk themselves?



Like other open source policies we asked about, many respondents are unsure if their organization requires commercial support for software components. A full 24% have no idea whether their organization mandates commercial support for any type of component used in a production environment.

When we look at companies with more than 50 employees, 29% have policies in place requiring commercial support for all software components, including those that are open source. An additional 18% have policies in place requiring commercial support for any components that are not open source. This means that almost one-half of organizations with more than 50 employees require some sort of commercial support option for the software they use.

This raises a question: Why the gap between commercial support of open source and proprietary components? Why would an organization that needs commercial support for software components exclude open source?

The likely answer is that it is because developers don't have to go through a traditional procurement process to download freely available open source components, skipping their legal and procurement departments entirely by installing a package at the command line. Where procurement doesn't have a mechanism to pay, they cannot limit usage.

Is this OK? At some point this lack of oversight might come back to haunt these organizations, as it has for companies like Equifax.

One other possible explanation is that these companies aren't yet aware of any option that will give them support and assurances for the open source components that aren't backed by large commercial open source vendors. Even if they have a checkbook in hand, they don't know who to pay.

Fortunately, a managed open source strategy can help address this problem. With managed open source, an organization can pay one vendor to get enterprise-grade coverage for the vast array of open source packages they use, across common ecosystems like JavaScript, Python, Java, Ruby, and more. As more organizations that require support for their software understand the benefits of managed open source, hopefully this "but who are we going to call when something breaks?" issue for all open source components finally becomes a thing of the past.

Almost one-half of organizations with more than 50 employees require some sort of commercial support option for the software they use.

FINDING #8

The key benefits of a managed open source subscription involve maintenance, security, and licensing

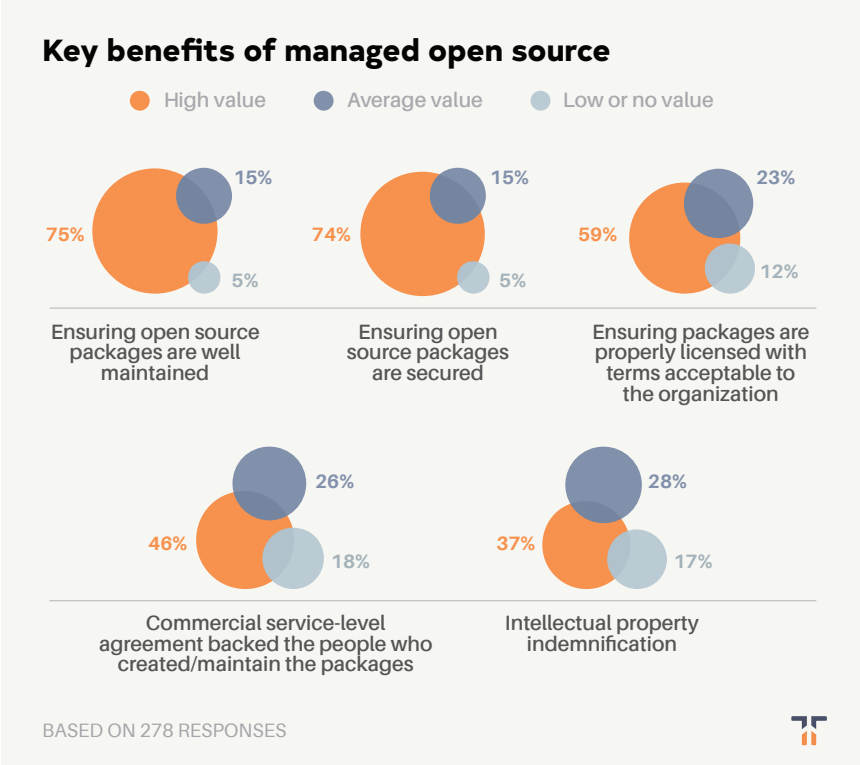
In our previous findings, we've highlighted some of the issues developers face when using open source today. Developers love using open source, and would like to use it even more. But several important things stand in the way.

We learned that the biggest concern they have with open source is the availability of reliable support. The biggest reason why they don't use even more open source today is because of concerns about how well components are going to be maintained into the future. We learned about how maintenance challenges are sucking up their valuable code-writing time. And we learned about the havoc that unmaintained or undermaintained and unacceptably licensed components can wreak in an organization.

So how might developers tackle some of these challenges so they can take advantage of the many benefits of open source, and use it even more effectively in their organizations?

One answer is by employing a managed open source strategy. Managed open source provides a way to help organizations better manage all of the open source software they use, ensuring it is up to date, secure, and well maintained, while providing standard commercial assurances like support under a service-level agreement and intellectual property indemnification (you can learn more about the approach to managed open source Tidelift takes here).

We explained managed open source as a concept to our survey respondents in order to understand which of the key benefits of a managed open source approach were most appealing to them.



While all of the benefits were seen as worthwhile by respondents, the most valuable should come as no surprise because they directly address many of the challenges developers face when using open source components.

The most valuable benefit of a managed open source subscription is that it helps ensure that open source packages are well maintained into the future. Three-quarters of respondents agreed that this benefit is either high value or extremely high value.

This is followed closely by ensuring that open source packages are secure, with 74% rating this benefit as either of extremely high value or high value. This is also the benefit with the most ratings for extremely high value (38%).

The most valuable benefit of a managed open source subscription is that it helps ensure that open source packages are well maintained into the future. Three-quarters of respondents agreed that this benefit is either high value or extremely high value.

The third most valuable benefit of a managed open source subscription is around ensuring components have a license that is acceptable to the organization, with 59% of respondents reporting this as either extremely high value or high value.

Companies with more than 1,000 employees are much more likely to think all elements of a managed open source subscription are valuable, but security stands out with 56% reporting that it is extremely valuable.

The evidence from this survey is clear. Developers are eager to make even better use of open source components. But there are critical issues around maintenance, security, and licensing that stand in the way and need to be addressed.

A managed open source strategy can solve many of these issues, making it easier than ever for developers to expand their use of open source, improve their confidence in the open source they use, and get time back to focus on the important work that really drives their business.

ABOUT THIS SURVEY

The 2019 Tidelift managed open source survey was conducted from June 24 through July 7, 2019. Participants were contacted via Tidelift and The New Stack email lists and social media. We screened respondents to make sure they use open source to build applications at work, and the full survey sample was 369 respondents.

Thanks to The New Stack, and in particular Head of Research Lawrence Hecht, for helping analyze and produce these findings.

ABOUT TIDELIFT

Tidelift makes open source work better—for everyone

Through the Tidelift Subscription and in direct partnership with maintainers, Tidelift is a single source for proactively managed open source components and professional assurances around those components. Tidelift makes it possible for open source projects to thrive, so we can all create even more incredible software, even faster.

ABOUT THE TIDELIFT SUBSCRIPTION

A managed open source subscription backed by creators and maintainers

The Tidelift Subscription manages your dependencies for you, delivering all of the capabilities you expect from commercial-grade software, for the full breadth of open source you use.

- **We provide the tools** you need to continuously catalog and understand the open source software that your application depends on.
- **We partner with and pay the open source community maintainers** of the exact packages you use, to ensure they meet the standards you require.
- **We address issues proactively,** not only scanning for new security, licensing, and maintenance issues, but also working with our participating open source maintainers to resolve them on your behalf.
- **We help you measure and improve your open source dependencies' health** —which improves your app's health—and give you a short list of high-impact steps your team can take to improve them even more.
- **We add commercial assurances that don't come for free** with open source packages, like intellectual property indemnification and support under a service level agreement. You expect these guarantees from proprietary software, and you should get them when using open source as well.

Request a demo and learn more: tidelift.com/subscription