RogueWave SOFTWARE

Open Source Support

7 questions to select, deploy, and maintain open source software effectively

AVOID BUSINESS-CRIPPLING DOWNTIME AND SECURITY VULNERABILITIES

A question that often comes up in enterprise software organizations is: What happens when open source software (OSS) doesn't work? Whether it's confusion building out a server stack or panic during a system outage, the best that IT leaders and developers can hope for is that they've planned and prepared just enough to get over this hurdle and realize that they must prepare for the next.

The challenges with selecting, deploying, and maintaining open source software today is vastly different when compared to just a few years ago. Back then, it was a matter of choosing one of a limited number of packages that solved a particular problem, learning about it, and getting it to work in a known environment. Today, it takes a team of architects, developers, administrators, security professionals, and more to work through everything from IT policy and security to debugging across a complex stack and dealing with on-site vs. SaaS/PaaS deployments. That's why job candidates today list as many skills as possible, and why employers dig deep to find that perfect candidate with enough knowledge about all the packages deployed in their system.

For organizations committed to adopting open source, here are seven key questions to think about when determining how effective current strategy is against common hurdles.

Who will support your implementation?

An IHS Markit survey of 400 medium to large-sized organizations revealed that enterprises see an average of five downtime events per month, costing between $1M and $60M per year, depending on the size of the business. Coupled with performance lags and scalability issues, it's challenging to ensure users get the best possible experience and prevent them from clicking away.

When an issue occurs, whether in development or production, teams usually rely on these remediation methods:

1. Self support
2. Publisher support
3. Community support

All these methods rely on the hope that the right skills are available and that they're dedicated to your problem until it's solved. Minimizing the time between problem detection and resolution is critical, yet all methods require the expertise to understand your entire stack and environment — how much time is spent getting up to speed versus implementing a fix? A recent ClusterHQ survey provides a hint:

"43 percent of application developers spend between 10 to 25 percent of their time debugging errors discovered in production, rather than developing new features."

For example, if the problem is a misconfiguration between ActiveMQ, a PHP application, and network security policy, who within your organization or the community can solve it within a reasonable timeframe? Many questions posted to community forums go weeks, months, or even years without response. According to a survey conducted by Open Systems Media and Rogue Wave Software, 54 percent of developers choose OSS packages based on the availability of technical support, so it's an important topic being considered.

Here's a real example from our open source support team. We have a customer in the document management space who was experiencing latency and heavy resource utilization with ActiveMQ. The person who had initially created the ActiveMQ implementation had left the company and the remaining IT staff was left without the necessary skills to diagnose the problem. We analyzed the situation and found that the original implementation was not built for scale and that the company had grown considerably since it was built. We recommended and helped implement a more scalable solution and the problems have yet to resurface.

Does your technical staff have the knowledge to support the software you've selected?

Self support, or relying on your team to solve issues, is a common approach yet the very reasons that make it popular are also why it's often ineffective. There's a perception that having direct control over people and tools makes problem resolution efficient but the reality is that the opposite is true. What happens when the skills or experience aren't there? What happens when expertise walks out the door? Moreover, should valuable developer time be spent debugging and fixing issues?

We recently analyzed a large sample of support customer incidents and categorized them into three root-cause scenarios to determine which ones occurred most often, as shown in Table 1.

Root-cause scenario	Description	Percentage of total issues
Environmental	Issue caused by the underlying infrastructure or related applications running in the landscape.	41%
Configuration	Issue caused by the improper configuration of the software.	39%
Other	Issue caused by another reason.	20%

Table 1: Analysis of support customer incidents

These results suggest that 80 percent of the issues our customers encounter are due to either lack of knowledge about how an application will run in a particular environment or misunderstanding of proper configuration. It's essential to have a plan that covers the lack of knowledge for crucial areas of your system. Whether it's for function or for performance, putting the resources in place to deal with unexpected issues involves three steps:

1. Identify critical areas of the system that require fast turnaround for problems
2. Identify gaps between those areas and your current skill sets
3. Implement remediation strategies (hiring new skills, obtaining commercial support, etc.)

How will you monitor the implementation?

Pick a package, install and configure it, then keep it running. That's the general lifecycle for provisioning open source, but what about optimizing for load, resources, and performance? It's often easier to find the skills for deployment, less so for continuous monitoring and optimization.

We have a customer in the high-frequency trading space who built a platform for automated trading using only open source technologies. The company was experiencing bottlenecks somewhere in the infrastructure but were unable to pinpoint exactly where the slowdowns were occurring. By installing the open source Zabbix monitoring platform in their environment, we were able to pinpoint several areas of slowness. We focused on scaling those parts of the system and improved the speed by several orders of magnitude. The customer has continued to use Zabbix as their enterprise monitoring solution, subsequently adding functionality for threshold alerts and resource utilization digest reports to the platform.

How will you stay ahead of community updates?

For the set of packages deployed in your system right now, are you aware of the latest versions and what they contain, and the most current best practices? Unless there's a person (or persons) dedicated to monitoring updates from the publisher and the community, it's nearly impossible to keep up. At best, this means it's nearly impossible to squeeze the most out of your deployment stack. At worst, your deployment is at serious risk of failure or vulnerability to software exploits.

An example from early 2016 was when "one programmer broke the internet" when the creator of kik, an open source JavaScript package that helps set up project templates, removed a minor package called left-pad from the NPM repository. This made the package inaccessible to thousands of applications around the world, and the downstream effect was that many organizations, including Facebook, Netflix, and Spotify, were affected until NPM republished the package.

This type of deeply embedded dependency isn't easy to discover, let alone fix. While the removal of left-pad made global headlines, most problems are subtle and much less popularized.

How quickly can you respond to critical security announcements?

Keeping on top of technical issues is one thing, how do you keep your open source packages secure? The time needed to monitor security updates can be substantial, and the expertise needed to select and deploy patches is typically hard to find.

A customer in the IT managed services space came to us after a breach of their intelligent platform management interface (IPMI) led to the installation of a rootkit on one of their production machines. The bad actors had installed Bitcoin mining software and were actively using the infected machine for this purpose. The customer, like many others whom we regularly work with, had chosen to disable the SELinux framework on their machines. Even though the IPMI would still be compromised, had SELinux been enabled, the hackers would not have been able to exploit the machine and install a rootkit. We instructed the customer on the proper use of SELinux using tools like audit2allow and helped them successfully "setenforce 1", greatly decreasing the probability of future exploitation.

Another way to stay informed is the OpenUpdate newsletter from Rogue Wave Software. This weekly digest provides open source news and security updates for the top mission-critical packages used by enterprises. We have staff dedicated to researching this information weekly — time that you shouldn't have to spend on your own. You can see an example and sign up for free here.

How will you guarantee comprehensive testing of your implementation?

In 2016, automated deployment is king. Organizations that can successfully implement continuous integration (CI) and continuous delivery (CD) will have a competitive edge, by producing high-quality releases fast, without fear of missing release dates or introducing problems into their production code. A key component of CI/CD is automated unit testing and open source technologies like Spock, JUnit, and Jenkins can help your business succeed in these areas. Automated deployment is laying the groundwork for containers and microservices but the key to that future lies in comprehensive testing of your software.

Amazon developers deploy their code to production every 11.7 seconds, a feat that would not be possible without solutions for automated testing. Like most disruptive technologies, the sophisticated frameworks that make this possible are community-driven, open source products.

It takes more than just technology to make these processes work, however. Moving to CI/CD requires changes in culture and process, and the best way to fast-track those solutions for your business is to seek out proven thought leaders who understand both the technical and human aspects of creating an automated software development lifecycle in an enterprise.

How do you deal with unexpected problems?

This last section summarizes the questions above to help determine whether your current strategy is effective or not.

Q: How do you deal with downtime in production systems?

1. Our team handles it
2. We go to the publisher
3. We ask the community

In all cases, consider whether the expertise exists and whether the support comes from a team dedicated to fixing the problem as fast as possible.

Q: What do you do when a problem occurs outside your team's areas of expertise?

1. Our team researches it and comes up with a solution
2. We go to the publisher
3. We ask the community

The same considerations above apply but also think about this: If the problem is the result of the interactions between packages and the system's environment, it's far more difficult to find the right expertise.

Q: What do you do when the community doesn't respond in a timely manner?

1. Our team researches it and comes up with a solution
2. We hire short-term contract developers with the necessary expertise
3. We wait for the community

For this scenario, is it better to have the situation out of your control and react to it or proactively implement a strategy outside of these options to contain it?

Q: How do you deal with release delays due to configuration, integration, or deployment issues?

1. We work overtime to meet the deadline
2. We hire short-term contract developers with the necessary expertise
3. We delay the release

Each of these options introduces unplanned costs, so consider a different option that lets you plan ahead with a cost effective mitigation solution.

Q: What do you do upon discovery that a system is vulnerable to a security attack?

1. Research the vulnerability and develop, test, and deploy a fix
2. Remove the suspect package and deploy an alternative
3. Wait until an attack occurs

Here, it's important to balance the likelihood of attack against the costs to implement the fix. Either way, you need the expertise to know which strategy will work.

Try enterprise-grade open source support

With Rogue Wave Open Source Support, you get much more than fast bug fixes, you get around-the-clock access to Tier 3/4 open source architects ready to support, consult, and educate your team to solve issues across your entire software stack and development lifecycle. Our vendor-neutral, unbiased support covers hundreds of open source software packages used in mission-critical production environments and includes:

- Guaranteed service level agreements (SLAs) with all support contracts
- 12x5 Silver Support or 24x7 Gold Support with around-the-clock coverage
- Access to Tier 3/4 open source architects and a dedicated CentOS development team
- The confidence to troubleshoot production issues, optimize performance, and complete system upgrades
- The experience to assess projects, perform architecture and security reviews, hold on-site classroom training, and create technical white papers
- Available support for CentOS Linux with a dedicated development team providing patches and hotfixes hosted on our own CentOS repository

See how Rogue Wave Open Source Support can help your team at roguewave.com.