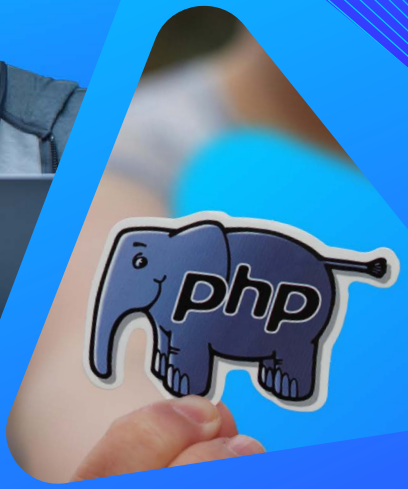
perforce
Zend



REPORT

2025 PHP Landscape Report

An in-depth examination of ongoing trends and evolutions in the PHP ecosystem.

Contents

Foreword

This year, 2025, PHP celebrates it's 30th year.

Matthew Weier O'Phinney
Senior Product Manager
Perforce Zend and OpenLogic

This is a tremendous landmark for the language. At the time that PHP first emerged, the web was still in its infancy, and many languages and commercial technologies were vying for the position of most-favored language for web applications. While many of these have disappeared or shifted to specialize in other contexts, PHP has largely retained a prominent position. Its design is uniquely suited for how the web works in practice, and it can serve anything from small sites with a handful of pages all the way to behemoth applications managed by the largest players in every industry.

The PHP ecosystem is continuously evolving. Around 15 years ago, PHP developers identified that a predictable release and support schedule leads to less uncertainty for users. We've seen this result in fewer organizations relying on end of life (EOL) versions, ensuring that PHP applications can access the latest features and performance improvements. That said, nearly two out of five respondents in this year's survey indicate they still deploy at least one EOL version, so there's plenty of work to go.

Considering PHP's maturity and consistently growing list of features, PHP developers have the world at their fingertips. They can integrate with just about any database on the planet, make both simple and sophisticated web service requests, and integrate with every emerging technology – whether it's key-value storage, search engines like Elastic, or real-time message brokers like Kafka. With that maturity, we're also seeing fewer new developers in the language and a larger pool of seasoned developers. This is a double-edged sword as we have experienced talent, but the fact that the pool isn't growing to keep up with demand means organizations are often struggling to hire. (We're hoping that our relaunch of the Zend Certified PHP Engineer exam will help raise visibility of qualified developers!)

With more sophisticated PHP applications also come new needs. Because of the critical nature of many PHP applications, security of the application and the entire stack becomes important to prevent data and privacy breaches. With many businesses operating at a global scale, performance at the edges and over prolonged hours is critical to ensure availability – after all, an application is only as good as it is available, and understanding the health of your application is more important than ever to ensure smooth customer interactions. While PHP is uniquely suited to this with its shared-nothing architecture, it still takes specialized understanding to build a scalable application, with the knowledge needed to operate PHP applications having grown immensely.

As always, I find our PHP Landscape Report data incredibly useful in understanding the health and needs of the PHP ecosystem. I'm thrilled that 30 years in, it is as vibrant as ever!

Sincerely,

Matthew Weier O'Phinney
Senior Product Manager | Perforce Zend and OpenLogic

About the Survey

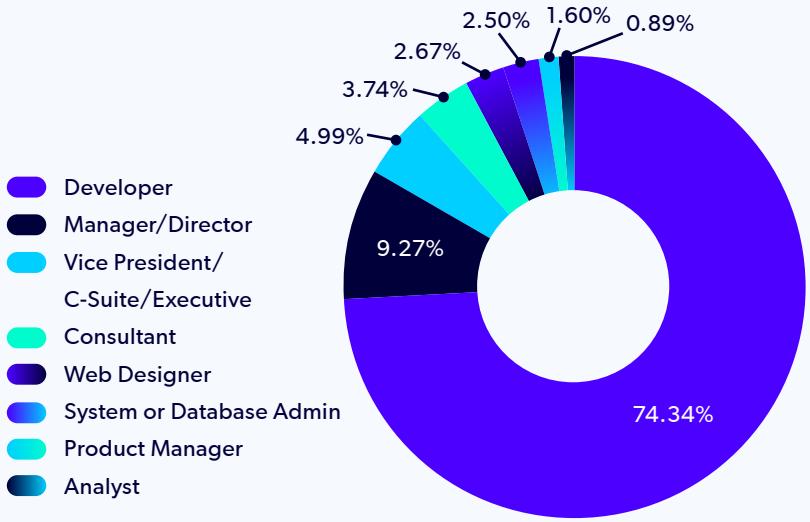
The Perforce 2025 PHP Landscape Report is based on the results of an anonymous survey of self-identified PHP users and administrators. It was conducted between October and December of 2024, and the survey received a total of 561 valid responses.

Respondent Firmographics

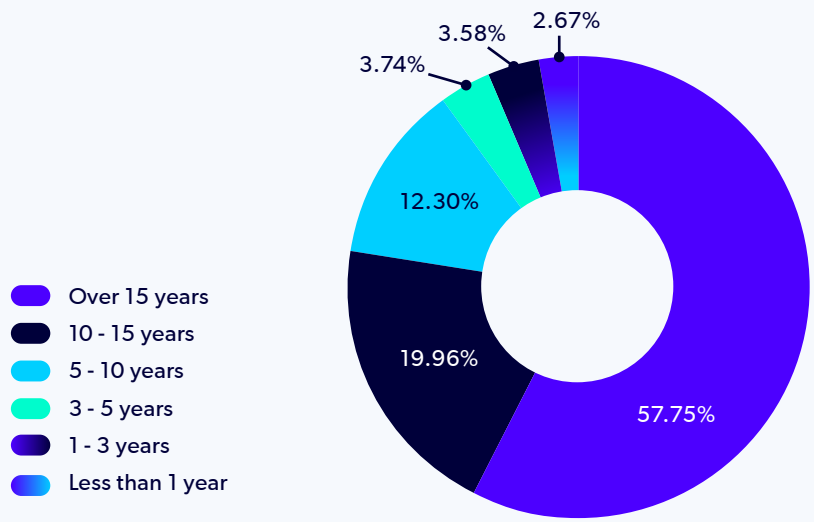
The survey's first question asked respondents to disclose their job title. As we have seen in previous years, the majority of participants were in developer roles, representing almost 75% of our surveyed population. Manager/Director roles came in second at 9.27%, and C-Suite roles represented 4.99%. All other roles represented 11.40% of survey participants

In a new question this year, we asked respondents to share how long they have been using PHP. Approximately 90% answered that they have been using PHP for 5 years or more, with 57.75% stating that they have been using PHP for over 15 years.

WHICH JOB TITLE BEST MATCHES YOUR CURRENT ROLE?



HOW LONG HAVE YOU BEEN USING PHP?



Development Team Size

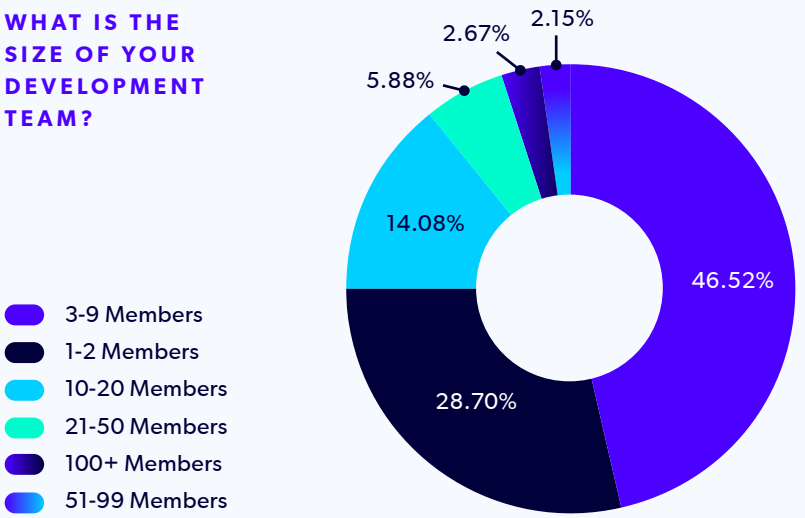
Next, we asked about the size of participant development teams. Over 75% of respondents reported development teams with 9 members or less, with 3-9 members being the most common size at 46.52%. This follows with our survey population from previous years.

Company Size

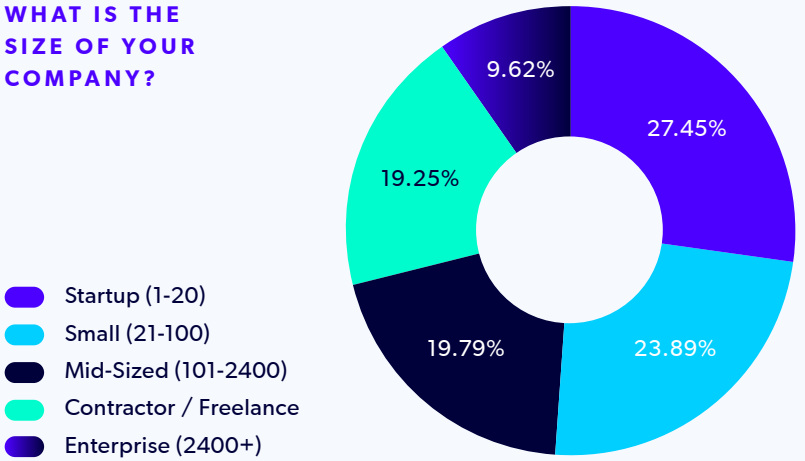
The next question in the survey asked participants to share their company size. As with previous years, the responses were distributed fairly evenly, with the most common answer being startups at 27.45%, followed by small companies at 23.89%, mid-sized companies at 19.79%, contractors or freelancers at 19.25%, and enterprise companies at 9.62%.

Most of our surveyed PHP developer teams have 9 or fewer members and tend to work for companies with fewer than 100 employees.

WHAT IS THE SIZE OF YOUR DEVELOPMENT TEAM?



WHAT IS THE SIZE OF YOUR COMPANY?

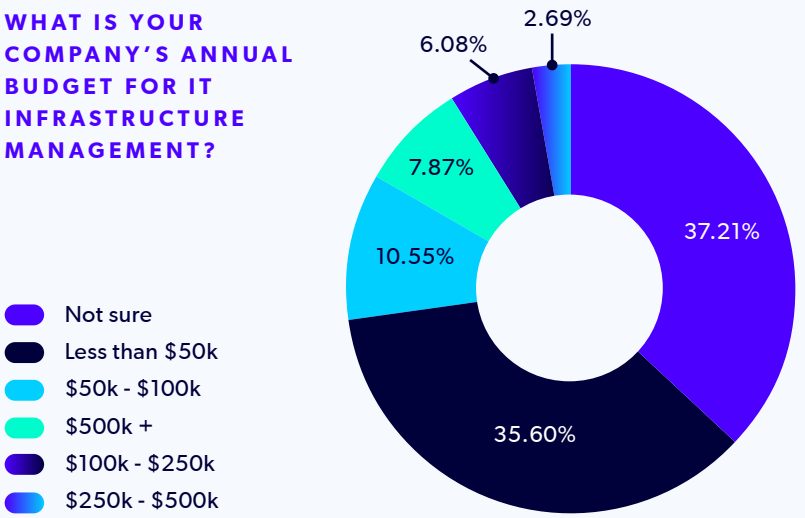


Annual IT Budget for Infrastructure Management

In a new question for the 2025 survey, we asked participants about their annual budget for infrastructure management. 37.21% were not sure or preferred not to answer, which followed with the large number of developers represented in the survey. However, 35.60% of participants had an annual budget of less than $50,000 per year, followed by 10.55% with an annual budget between $50,000 – $100,000 and 7.87% with an annual budget over $500,000

Out of all reported budgets, over half of surveyed PHP developer teams had an annual IT budget of $50,000 or less.

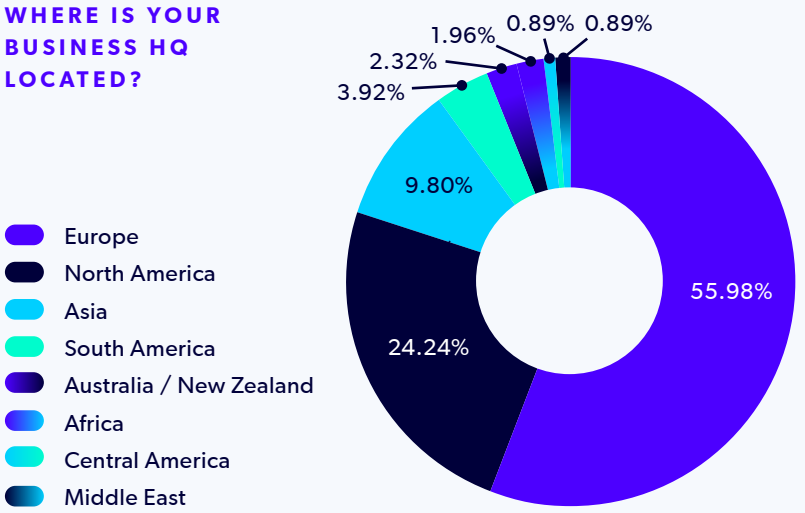
WHAT IS YOUR COMPANY'S ANNUAL BUDGET FOR IT INFRASTRUCTURE MANAGEMENT?



Region

We then moved on to asking participants to identify where their global headquarters were located. Over half were headquartered in Europe, with 24.24% headquartered in North America and 9.80% in Asia. South America, Australia and New Zealand, Africa, Central America, and the Middle East each represented less than 5% of participants, respectively.

WHERE IS YOUR BUSINESS HQ LOCATED?



Industry

The largest portion of survey participants were in either the Software as a Service or Technology industries, representing 25.85% and 17.29% of responses respectively. Among participants who selected "Other," common responses included property management, real estate, tourism or entertainment, and marketing.

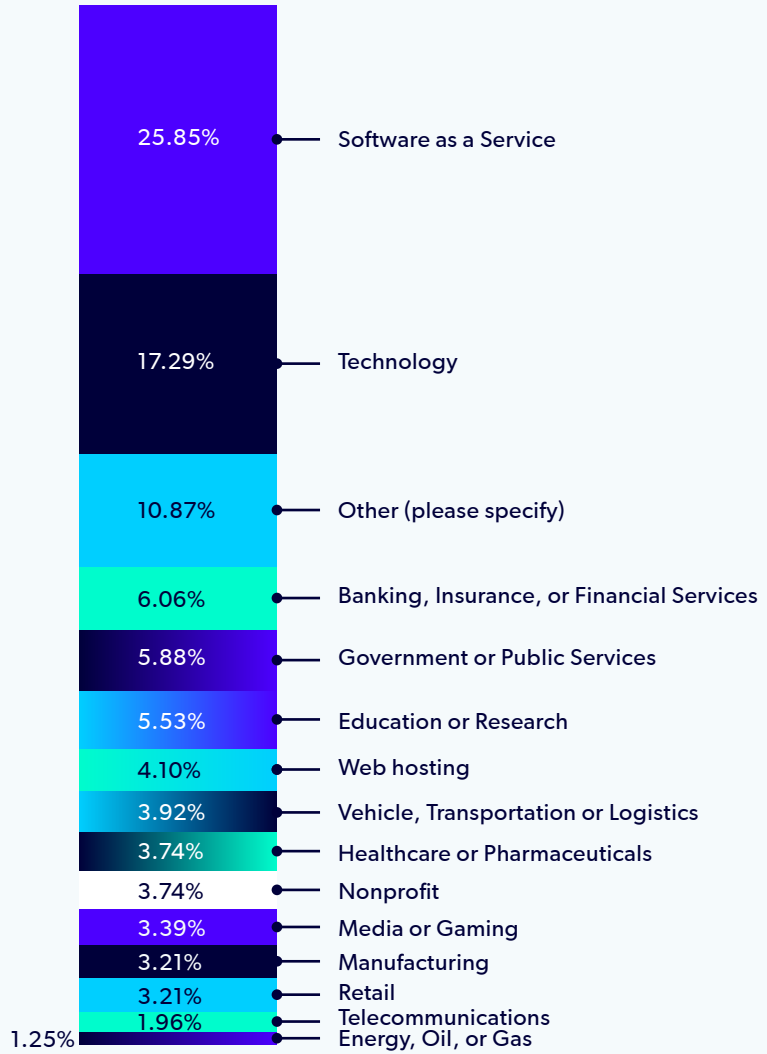
Key Takeaways

Looking at the firmographic profile for the 2025 PHP Landscape Report, we are pleased to find an overall balanced population. As with previous years, our participants tended to be in developer roles, which we feel is logical as developers usually have the most involved role in managing and building PHP applications.

We did find that our survey population skews toward a more experienced skill set, with many participants having over a decade of PHP experience. However, as PHP is now 30 years old, we feel it makes sense that many professionals will have been using it for an extended time.

Given our sample size and the even distribution throughout company sizes, developer team sizes, and other areas, we feel that this report is representative of the greater PHP landscape, and it gives accurate insight into new trends and the evolving state of PHP development and management.

IN WHICH INDUSTRY DOES YOUR COMPANY BELONG?

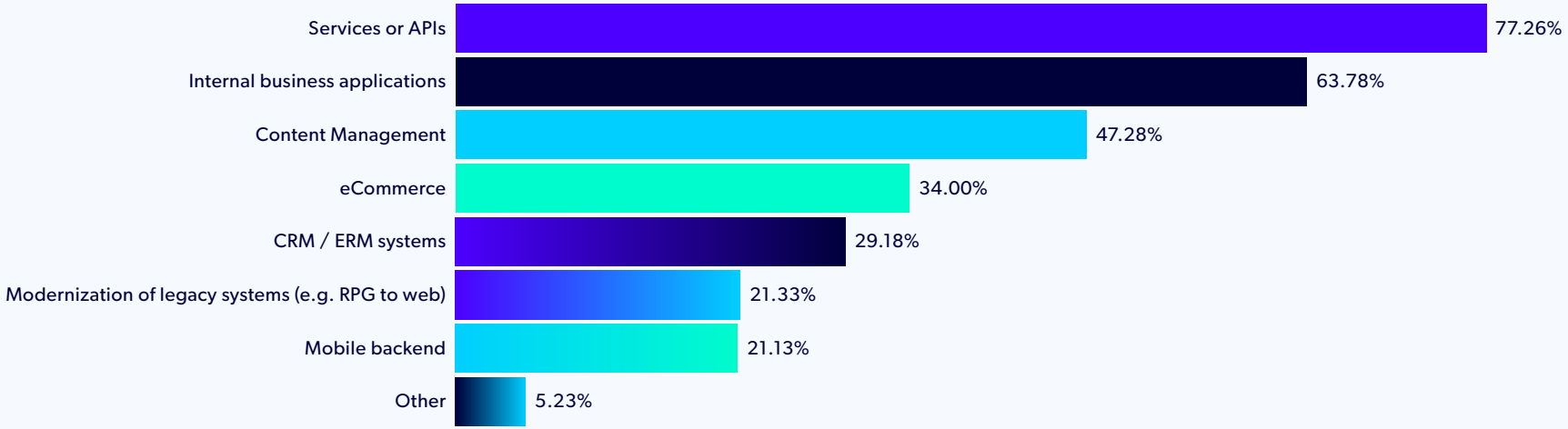


PHP Application Development Trends

Each year, we ask questions surrounding application development trends, allowing us to track ongoing evolutions in the PHP ecosystem. This includes information surrounding the types of PHP applications being developed, which technologies are being used or integrated with (including container and orchestration technologies), and how applications are deployed.

PHP Application Types

We began this section by asking teams about the types of applications they are building or deploying, allowing respondents to select multiple options. As with our findings in previous years, Services or APIs (77.26%), Internal Business Applications (63.78%), and Content Management (47.28%) were the top three PHP application types. Ecommerce (34.00%) and CRM/ERM Systems (29.18%) rounded out the top five.



PHP Application Integration Trends

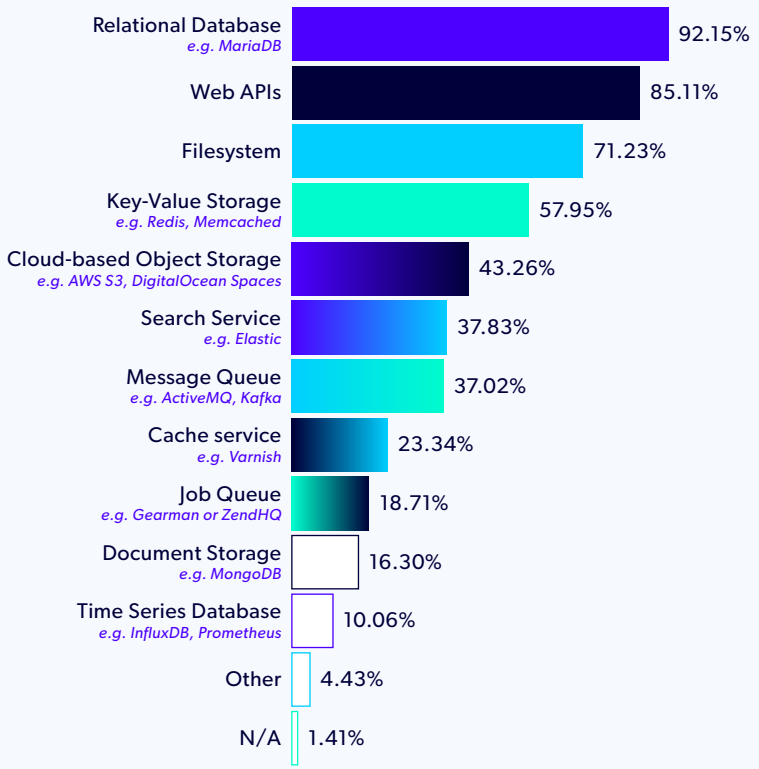
Next, we asked respondents to report which systems their PHP applications integrate with, allowing participants to select multiple options as applicable. Over 90% identified integration with Relational Databases, followed by 85.11% with Web APIs, and 71.23% with Filesystems.

While this tracks closely with our 2024 findings, we saw an increase in usage across most integration options. For instance, Time Series Databases, Message Queues, Search Services, and Key-Value Storage all saw increases compared to our 2024 report. Usage of Job Queues remained consistent year over year, while Document Storage integrations dropped from 19.83% to 16.30%.

Key Takeaways

PHP is well-known as the "glue language for the web." Its rich ecosystem of extensions provides connectivity to just about any database technology, and the expansive set of tooling for performing web requests means that it can pull and push data anywhere.

These capabilities are mirrored in what PHP powers, and the fact that APIs are so prevalent amongst respondents is no accident. Additionally, because browsers are ubiquitous and provide a generally consistent experience across operating systems, many businesses choose to build internal applications as web applications — and the fact that PHP can connect with just about anything means these applications can provide business critical integrations.



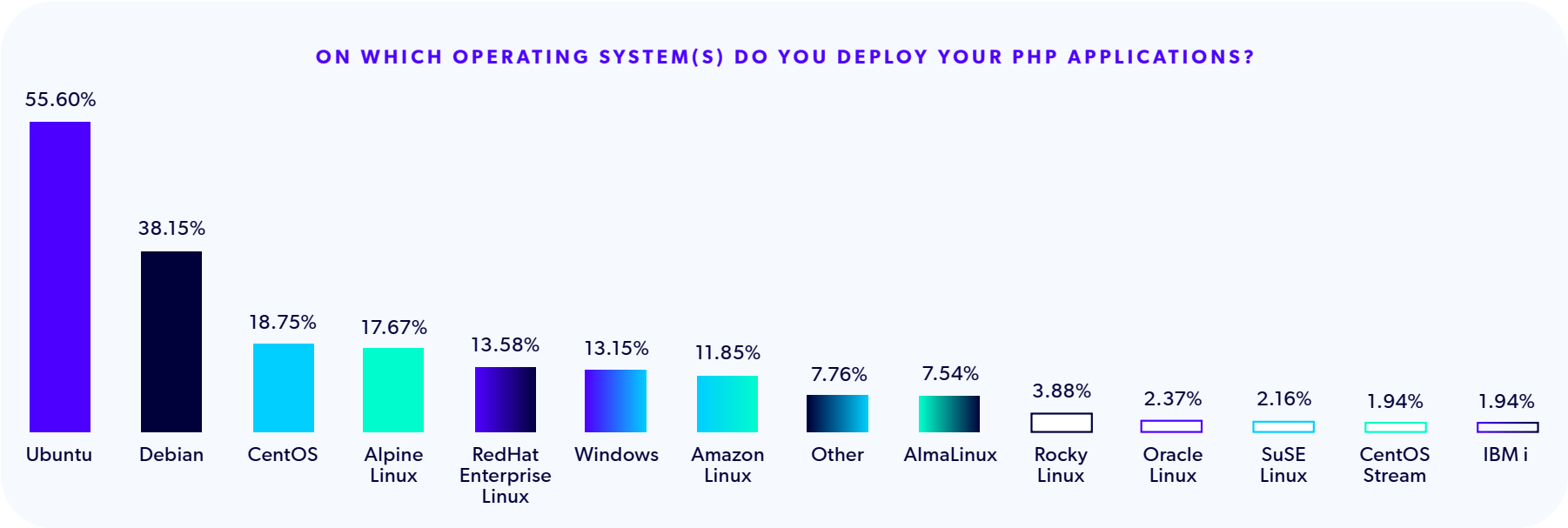
Operating System Usage Trends

We then asked participants to identify which operating systems (OS) they deploy their PHP applications on, with the ability to select multiple options as needed. As we've seen in previous years, Ubuntu (55.60% of applications) and Debian (38.15%) continue to be the most popular OS options for PHP teams, followed by CentOS (18.75%), Alpine Linux (17.67%), and RedHat Enterprise Linux (13.58%).

Key Takeaways

We were heartened to see a steep drop in CentOS usage this year (24.41% of applications in 2024 vs. 18.75% in 2025), as the final CentOS version, version 7, reached end of life in June 2024. The big surprise this year, however, was an increase in Alpine Linux usage at roughly the same rate as CentOS (12.11% in 2024 vs. 17.67% in 2025).

Interestingly, we did not see significant increases in containerization (see page 13). This suggests that the PHP users who are using containers are increasingly moving to Alpine Linux for their container operating systems, instead of using more mainstream operating systems.



Deployment Trends

We next asked participants to share where they are deploying their PHP applications, allowing them to select multiple answers as applicable. 55.67% of applications were deployed on-premises, 33.62% on Amazon Web Services, 29.98% on Other, and 12.42% on Digital Ocean. Most write-in responses were specific to web hosting providers, with Hostinger and Hetzner as the top write-in options.

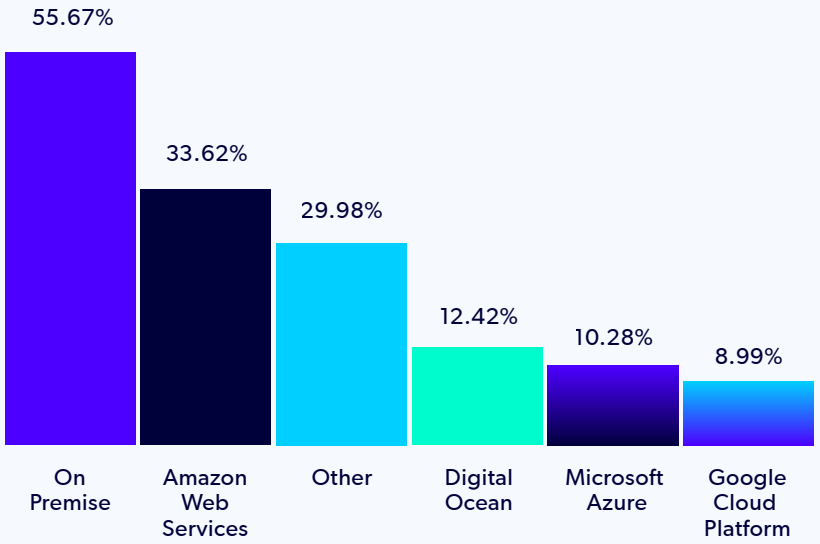
Smaller companies with less than 100 employees were more likely to adopt Digital Ocean compared to their larger counterparts, at 10.22% usage vs. 3.40%. Larger companies with over 100 employees, however, were more likely to deploy on Amazon Web Services than smaller companies, at 27.67% vs. 20.04%.

Key Takeaways

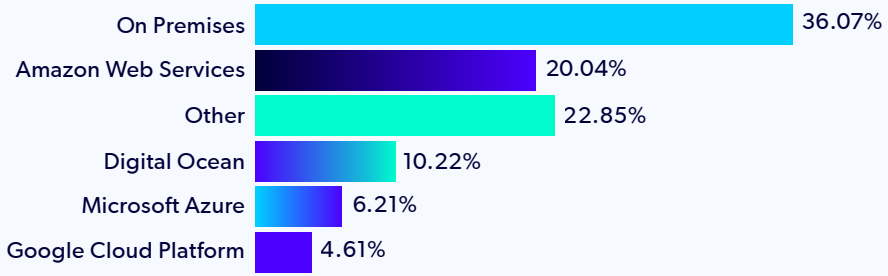
The results this year continue trends we observed in 2024, with an increasing number of organizations bringing their web applications on-premises, repatriating them from public clouds. We've observed price increases across all clouds, which may be a contributing factor, with other factors including data privacy and regulatory compliance.

Fortunately, ease of deployment has always been a strong suit for PHP, making these migrations relatively easy for businesses to accomplish.

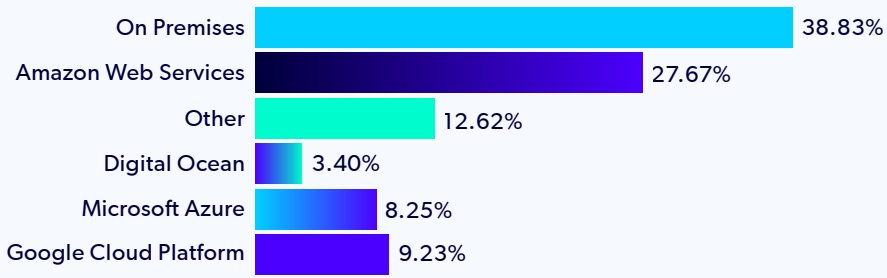
WHERE ARE YOU DEPLOYING YOUR PHP APPLICATIONS?



COMPANIES WITH 1 - 100 EMPLOYEES



COMPANIES OVER 100 EMPLOYEES



Web Server Trends

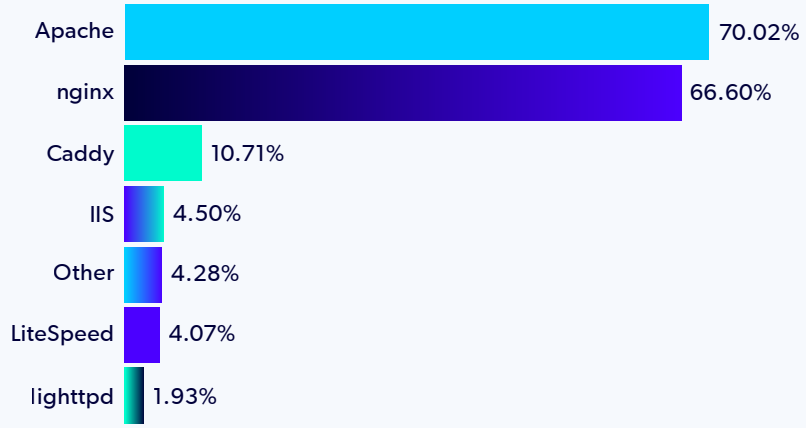
Next, we asked our participants to share which web servers they use for their PHP applications, including the option to select multiple servers. Apache and nginx led the pack with 70.02% and 66.60% of respondents, respectively. Caddy rose from fifth place in 2024 to third in 2025 at 10.71%.

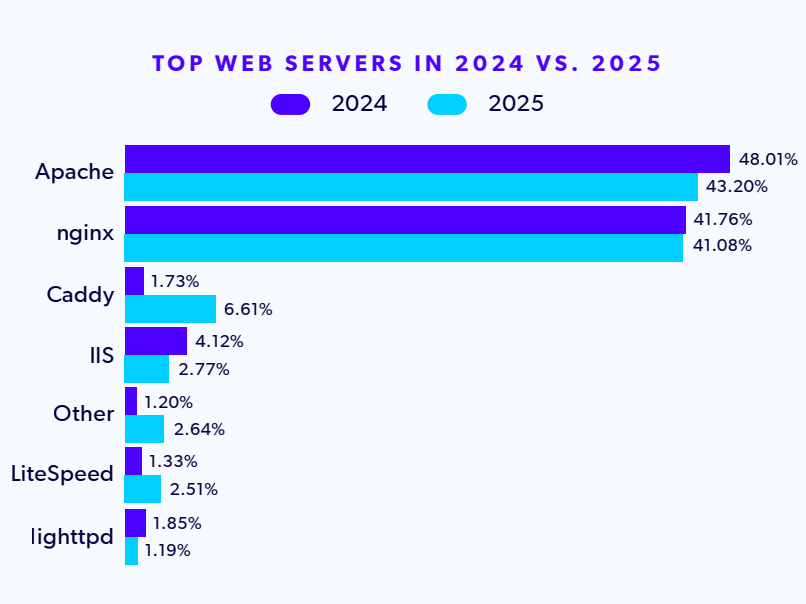
Considering respondents' top web servers for PHP applications year over year, most options have stayed relatively consistent. Caddy, again, has seen a significant jump in usage, with FrankenPHP (an application server built on the Caddy web server) being a repeated write-in option. While Apache and nginx are still the top selection, we did see a slight decline in their usage compared to our 2024 results.

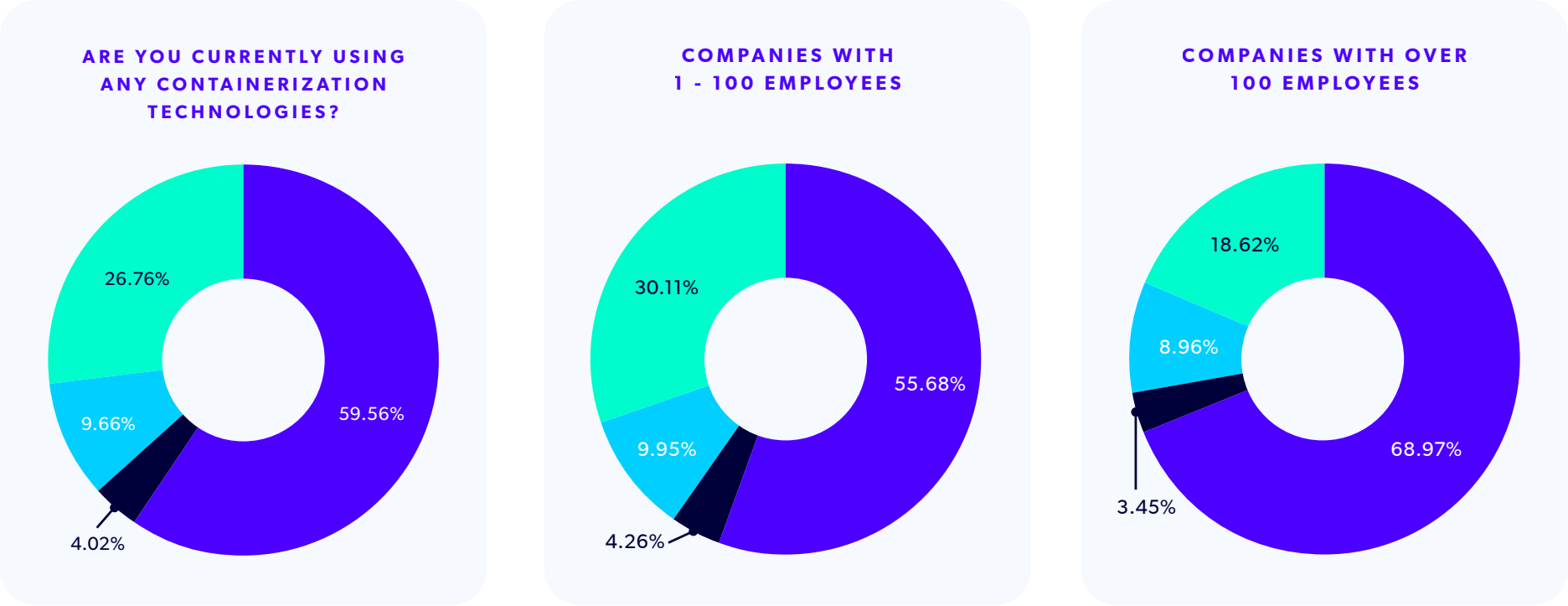
Apache, nginx, ISS, and lighttpd all saw small drops in usage, while Caddy and LiteSpeed saw an increase in adoption.

Key Takeaways

Apache and nginx are clearly here to stay. The numerous tutorials demonstrating usage with PHP, plus their rich feature sets, make them easy choices for organizations. Caddy's rise in popularity is definitely notable; we expect that its turnkey Let's Encrypt integrations, simplified FastCGI and reverse proxy configuration, and ability to quickly and seamlessly proxy for S3-compatible storage make it a strong choice for many new to serving web applications.







Container Trends

Next, we asked participants to tell us about their usage of container technologies. 59.56% reported that they are currently using container technologies, and 9.66% are planning to use container technologies within the next 12 months. 4.02% are currently using containers, but planning to scale back usage in the next 12 months, and 26.76% are not using containers and have no plans to use them in the future.

We found that larger companies were much more likely to currently use containers compared to their smaller counterparts at 68.97% vs. 55.68% of responses. This aligns closely with our data from our 2024 survey, showing container adoption rates remaining relatively consistent year over year.

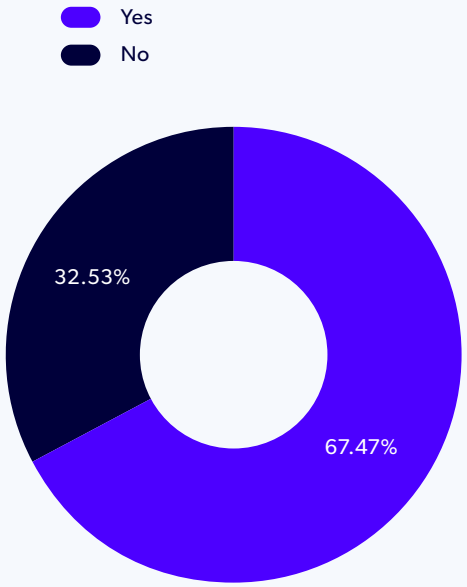
Community Container Usage

In a new question, we asked participants who noted they were currently using container technology if they were using community container images. 67.47% responded that yes, they were currently using container images offered by the community.

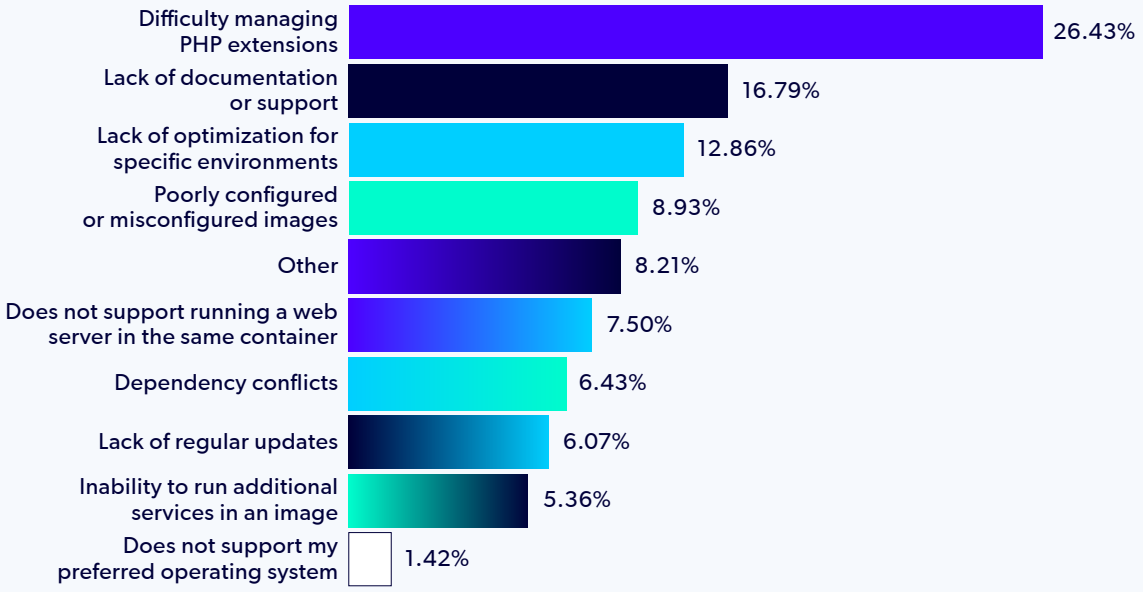
We then asked about which issues, if any, these PHP professionals experienced while using community container images. 26.43% of participants cited Difficulty Managing Extensions as the top issue with using community container images, followed by Lack of Documentation or Support (16.79%) and Lack of Optimization for Specific Environments (12.86%).

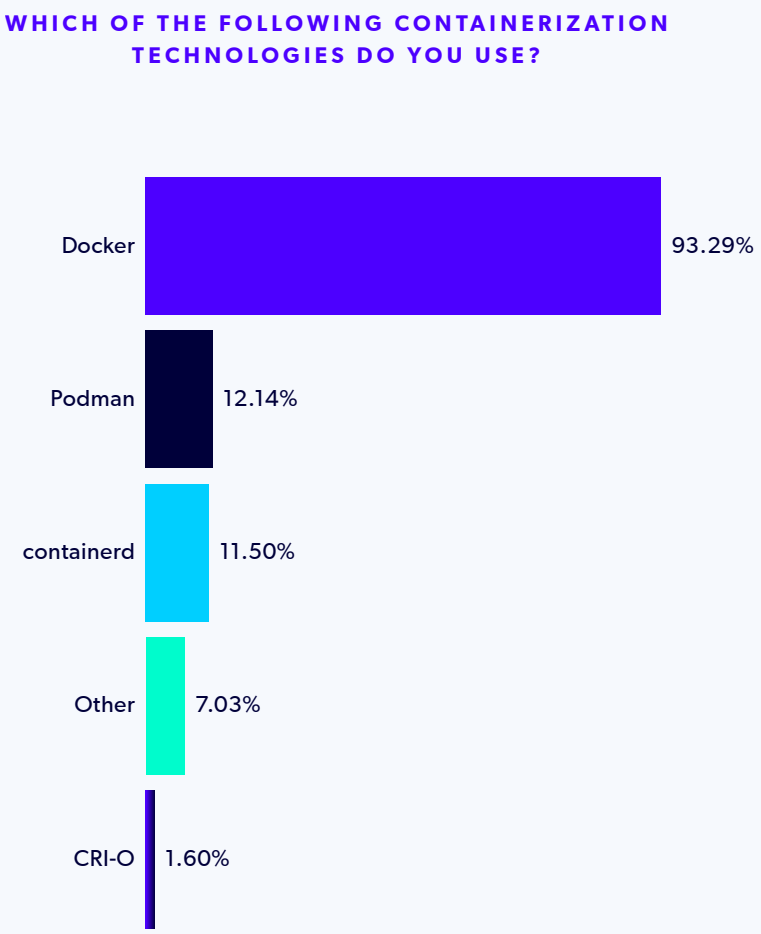
The most common write-in answers were slow build times and community container images failing to keep pace with the PHP release cycle.

DO YOU USE CONTAINER IMAGES OFFERED BY THE PHP COMMUNITY?



WHAT ARE THE ISSUES YOU'VE EXPERIENCED USING PHP COMMUNITY CONTAINER IMAGES?





Top Container Technologies

We then asked participants about which container technologies they utilize, allowing for the selection of multiple options as needed. Docker was the most used at 93.29% of respondents, followed by Podman (12.14%), containerd (11.50%), Other (7.03%), and CRI-O (1.60%). Repeated write-in options included LXC, K8, and Rancher.

The most notable deviation from our 2024 findings is the increase in usage of both Podman and containerd. Podman rose from 8.07% of 2024 respondents to 12.14% in 2025, and containerd moved from 5.47% to 11.50% - more than doubling in usage year over year.

Key Takeaways

Containers provide a way to create reproducible PHP environments. This can ensure that you have the same environment in development as in production, removing the “works on my machine” excuse when something breaks.

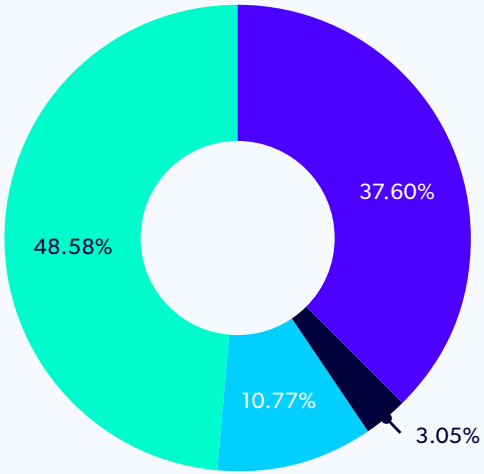
However, containers have a steep learning curve, which smaller organizations often cannot accommodate. Additionally, the difficulties developers experience with the community images can make adoption frustrating for many. Fortunately, creating your own images using patterns established from provisioning virtual machines is relatively easy, and a number of organizations, including Perforce Zend, are stepping up to provide simpler alternatives.

Orchestration Trends

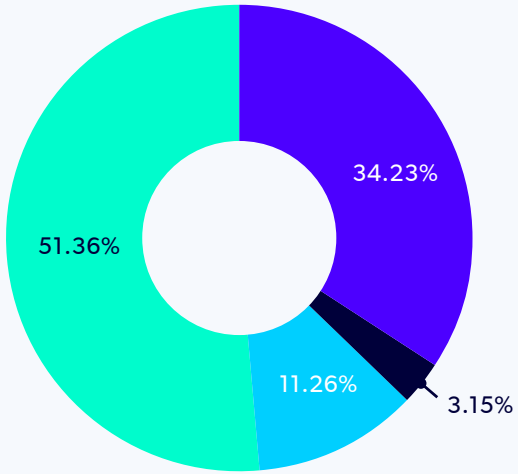
Next, we asked our survey respondents to tell us about their orchestration technology usage. 37.60% of participants said that they were currently using orchestration in their PHP applications, with an additional 10.77% planning to use orchestration technologies within the next 12 months. 48.58% currently do not nor plan to use orchestration technology, with 3.05% planning to scale usage back in the next 12 months.

Comparing our findings by developer team size, we found that developer teams with 20 members or more were far more likely to utilize orchestration technology (68.75%) compared to those with fewer than 20 members (34.23%).

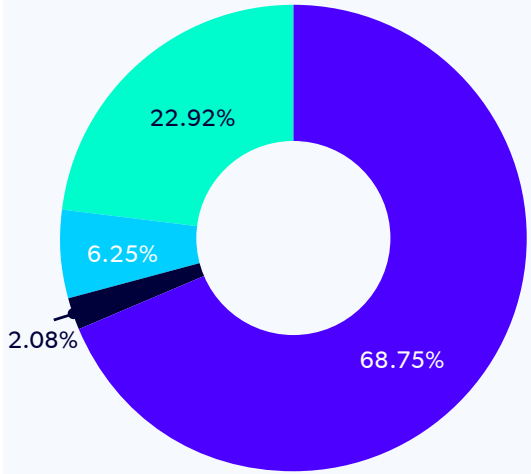
ARE YOU CURRENTLY USING ANY ORCHESTRATION TECHNOLOGIES?



TEAMS WITH 1-20 MEMBERS



TEAMS WITH 20+ MEMBERS



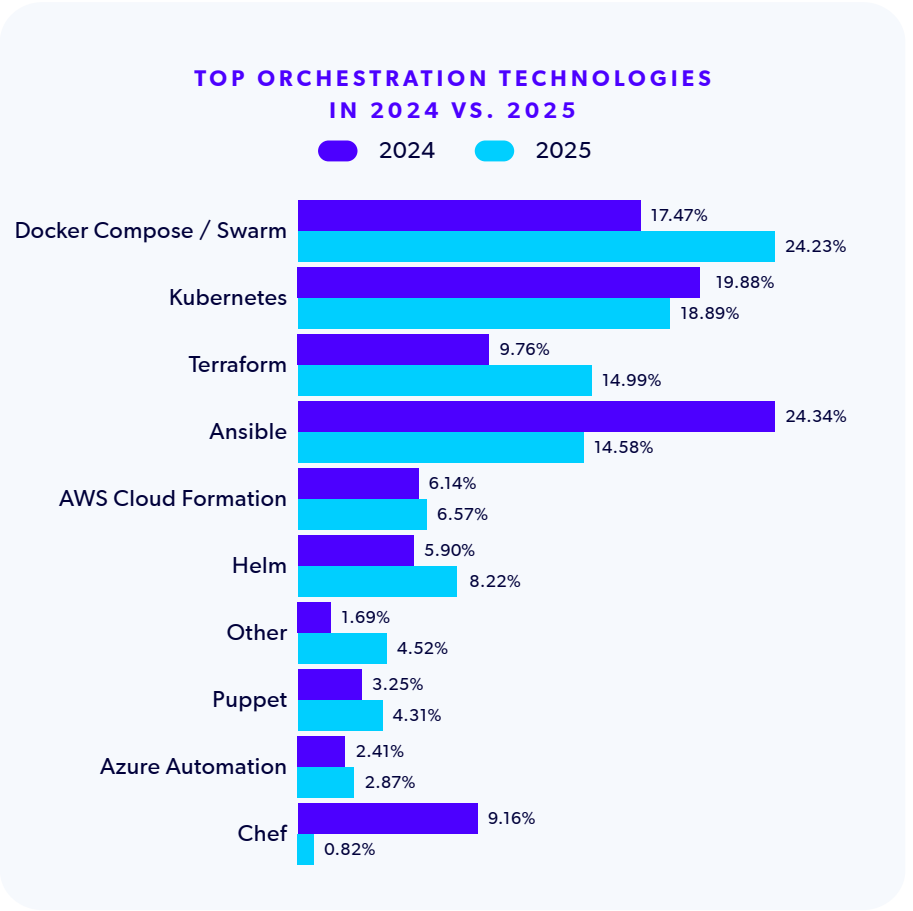
- Yes
- Yes, but we are scaling back usage in the next 12 months
- No, but we are planning to use them within the next 12 months
- No, and we currently have no plans to use them

Top Orchestration Technologies

Next, we asked participants to identify which orchestration technologies they were using in their applications.

Compared to 2024, we found that Docker Compose/Swarm had outpaced Kubernetes as the top choice for PHP professionals. However, both saw a change in usage, with Docker Compose/Swarm rising from 17.47% of 2024 participants to 24.23%, and Kubernetes falling slightly from 19.88% to 18.89%.

Other orchestration technologies likewise saw a rise in adoption in 2025, including Puppet, Azure Automation, and Terraform. Terraform is particularly noteworthy, with 14.99% of respondents reporting usage, slightly outpacing Ansible at 14.58% of respondents.



Key Takeaways

PHP has a "shared nothing" architecture, meaning after each request is handled, it discards all state before processing the next request. This architecture enables horizontal scaling, where instead of adding more capacity to your application server, you throw more servers into the mix and use a load balancer to route to the least busy instances.

While this makes scaling simpler, it also poses another problem: how do you orchestrate the application? You now need a load balancer, multiple PHP nodes, likely some sort of distributed session storage such as Redis or Valkey, and, of course, any other services you consume.

It is the need to scale horizontally that has driven adoption of both containers and orchestration tooling in the PHP ecosystem, and this is reflected in the fact that the leading orchestration tooling choices are Docker Compose/Swarm and Kubernetes. Compose and Swarm are easier technologies to learn and master, and since many applications can run on a single physical instance, they are often sufficient — they are in many ways ideal for client agencies that need to run many tenants with smaller bandwidth requirements.

We were surprised to see Terraform outpace Ansible this year. Ansible has had steady growth in past years, largely due to having the backing of RedHat and IBM. That said, the immense platform agnosticism of Terraform, and the ease with which you can provision machines for deploying k8s clusters, makes it a good fit for those using containers as well.

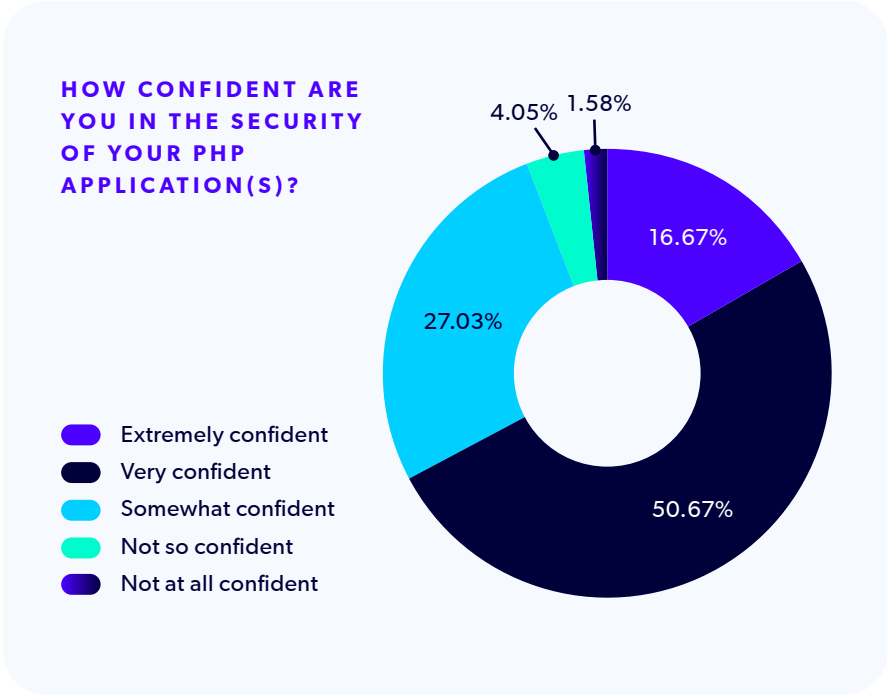
PHP Security and Compliance Trends

Our next section takes an in-depth look at upcoming and ongoing trends surrounding PHP security and compliance. With security consistently ranked as a top concern among PHP professionals year after year, it's essential to track the overall confidence levels surrounding PHP security and to explore the tactics teams are currently using to keep critical applications secure.

Security Trends

We began this section by asking participants to share their confidence level in the security of their PHP applications. 16.67% selected "Extremely Confident," 50.67% selected "Very Confident," 27.03% selected "Somewhat Confident," 4.05% selected "Not So Confident," and 1.58% selected "Not At All Confident."

With nearly 70% of survey participants ranking their confidence level at "Very Confident" or "Extremely Confident," it's clear that PHP is generally regarded as a secure choice of coding language.



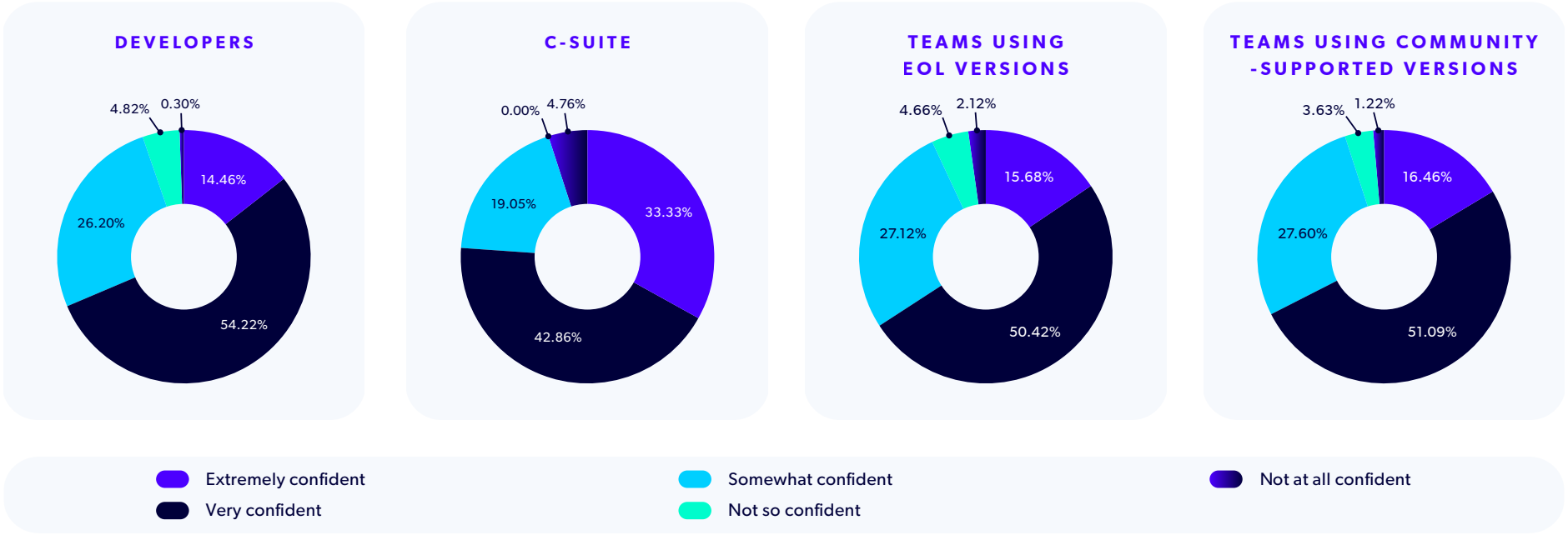
33.33% of respondents in C-suite roles were “Extremely Confident” in their PHP security compared to 14.46% of developers. However, developers were more likely to be “Very Confident” in their PHP security compared to their executive counterparts, at 54.22% vs. 42.86% of respective respondents.

Interestingly, teams using EOL PHP versions had very similar confidence levels to teams using PHP versions still supported by the community. Where 67.55% of teams deploying PHP 8.1, 8.2, or 8.3 reported feeling “Very Confident” or “Extremely Confident” in their PHP security, 66.10% of teams deploying EOL PHP reported the same.

Key Takeaways

PHP’s security posture has evolved tremendously over the decades, giving its users greater confidence in the language. However, we’ve also seen a number of high-profile CVEs in the past few years, including parameter injection vulnerabilities in the CGI binary, remote code execution vectors when spawning external programs on Windows, and XML External Entity vulnerabilities that can lead to information disclosure.

As such, seeing high confidence in PHP security from teams using EOL PHP feels misplaced; the only mitigations are to update to a supported PHP version or to use a commercial LTS version. Regardless, PHP is earning a solid reputation as a secure language, particularly when teams keep on top of language updates.



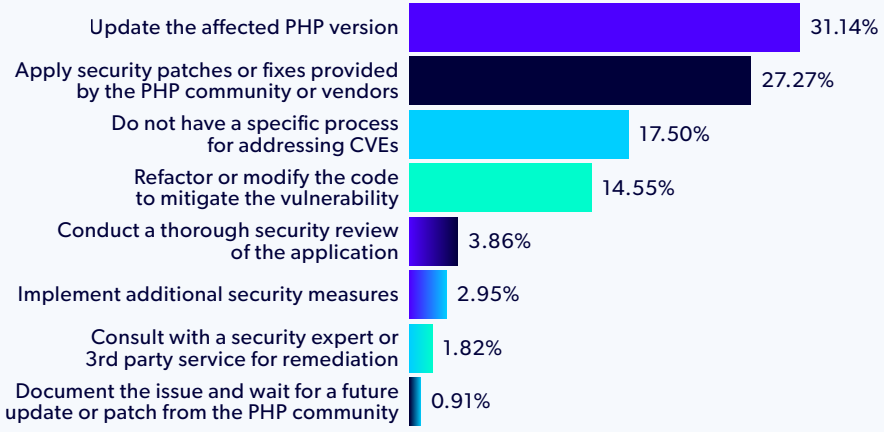
Tactical Security Trends

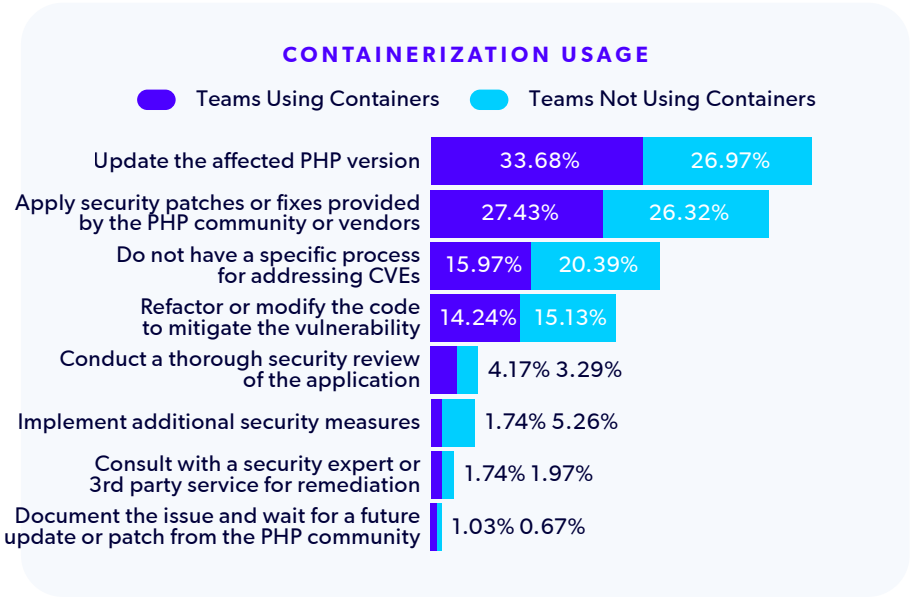
Next, we asked teams to rate which security measures were the most important in the development and maintenance of their PHP applications. Respondents were asked to rate each option on a scale of 1 – 5, with 1 indicating the tactic is not important, and 5 indicating the tactic is very important.

We found that the top tactic was a tie between Regularly Applying Security Patches and Updates in Application Dependencies, and Implementing and Enforcing Secure Coding Practices, with weighted averages of 4.37. These tactics were followed closely by Implementing Strong Authentication and Access Controls, with a weighted average of 4.34.

Segmenting by PHP version, we found that teams deploying EOL PHP were more likely to give greater importance to Implementing and Enforcing Secure Coding Practices over Regularly Applying Security Patches and Updates in Application Dependencies. Additionally, teams using EOL PHP in their applications were less likely to give importance to Regularly Reviewing and Updating Security Policies compared to those using community-supported versions, with respective weighted averages of 3.28 vs. 3.41.

WHICH OF THE FOLLOWING SECURITY MEASURES ARE MOST IMPORTANT IN THE DEVELOPMENT AND MAINTENANCE OF YOUR PHP APPLICATIONS?





In a new addition to this survey, we asked participants how they address identified issues and CVEs in their PHP applications. The top tactics were to Update the Affected PHP Version at 31.14% of participants, and Apply Security Patches or Fixes Provided by the PHP Community or Vendors at 27.27%. The top three was rounded out by teams who Do Not Have a Specific Process for Addressing CVEs at 17.50%.

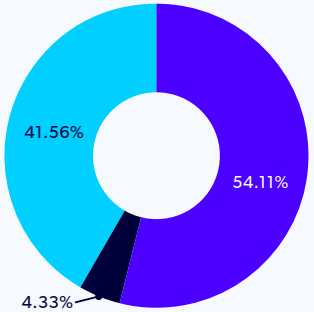
We then examined the differences in how teams address issues and CVEs by usage of container technologies. We found that teams using containers were more likely to update the affected PHP version than those who don't use containers (33.68% vs. 26.97%). Additionally, teams not using containers were more likely to not have a specific process for addressing CVEs in place compared to those using containers (20.39% vs. 15.97%).

Key Takeaways

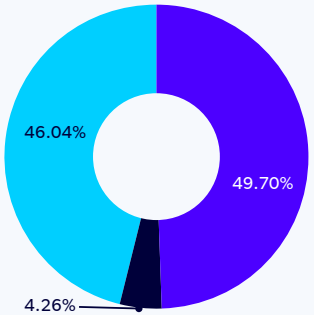
As noted in the previous section, the best approach to staying secure is to keep your PHP version updated. It is heartening to find that for those who are on EOL versions of PHP that LTS PHP versions are a common remedy, and that they turn to secure coding practices to further enhance their application security.

While we have previously in this report recommended containers as a way to create reproducible application environments, we note that they also bring an important security benefit: by making it easier to test newer versions of PHP, containers provide a path for more confidence in applying updates.

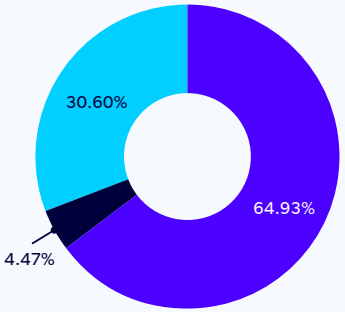
DO YOUR PHP APPLICATIONS HAVE REGULATORY OR INDUSTRY COMPLIANCE REQUIREMENTS?



COMPANIES WITH 1 - 100 EMPLOYEES



COMPANIES WITH OVER 100 EMPLOYEES



Yes
No, but we will within the next 12 months
No, and we currently have no forecast for them

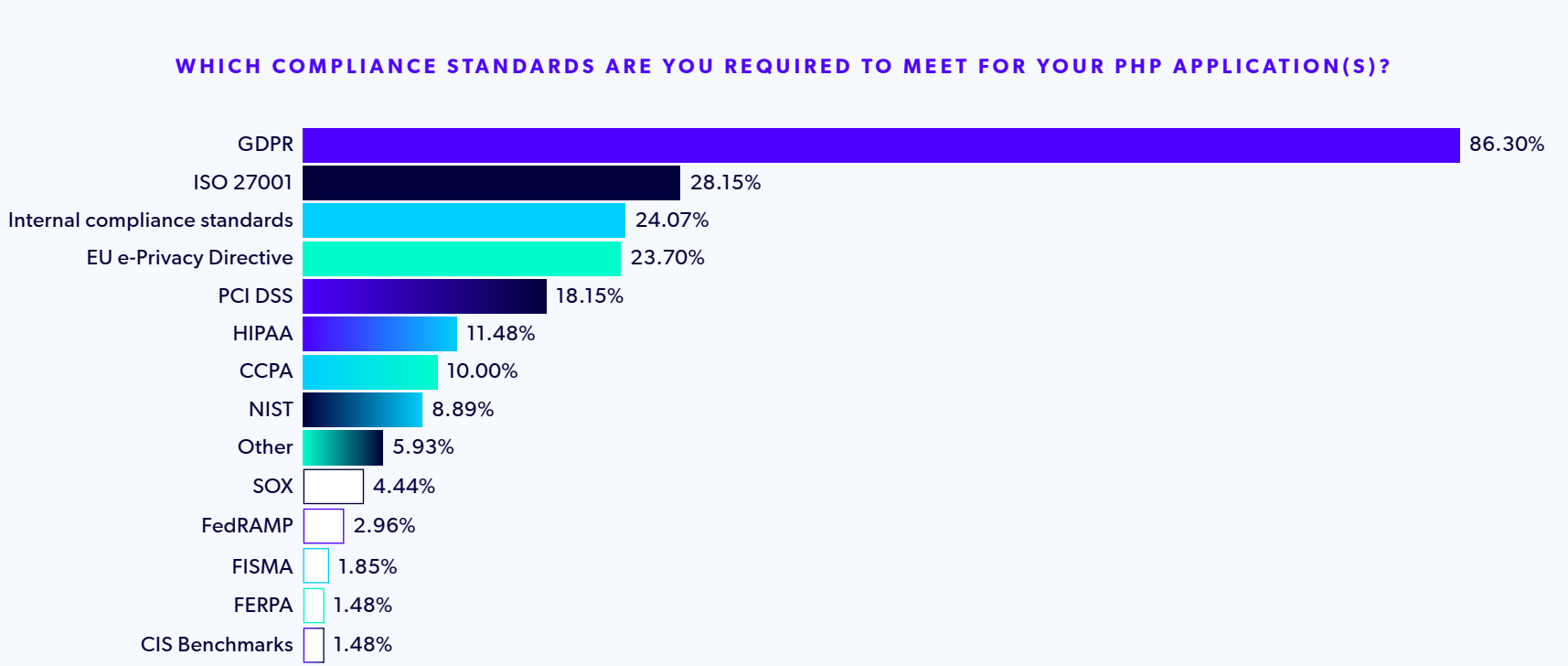
Compliance Requirements and Trends

Next, we asked participants if their PHP applications had any regulatory or industry compliance requirements. 54.11% said yes, 4.33% do not currently but will within the next 12 months, and 41.56% said no and did not have a forecast for them.

Comparing our results to our 2024 report, a similar number of teams have regulatory or industry compliance requirements. However, there has been an increase in teams who do not have nor forecast any compliance requirements compared to last year (45.89% vs. 38.37%). A smaller percentage of teams (4.33% vs. 7.17%) likewise do not currently have compliance requirements but anticipate them in the next 12 months.

Looking at compliance requirements by company size, we found that companies with over 100 employees were more likely to have regulatory or compliance standards compared to companies with under 100 employees. (64.93% vs. 49.70%).

Companies over 100 employees were 15% more likely to have PHP apps with compliance requirements compared to companies under 100 employees.



Compliance Standards

Next, we asked those who answered that they needed to meet compliance requirements to share which standards their PHP applications must meet, including the option to select multiple answers as applicable.

GDPR was the clear frontrunner with 86.30% of respondents. It was followed by ISO 27001 (28.15%), internal compliance standards (24.07%), EU e-Privacy Directive (23.70%), PCI DSS (18.15%), and HIPAA (11.48%). All other compliance regulations represented 10% or less of participants, respectively.

The need to meet GDPR standards has risen steeply, with 86.30% of teams requiring GDPR compliance in 2025 compared to 69.68% in 2024.

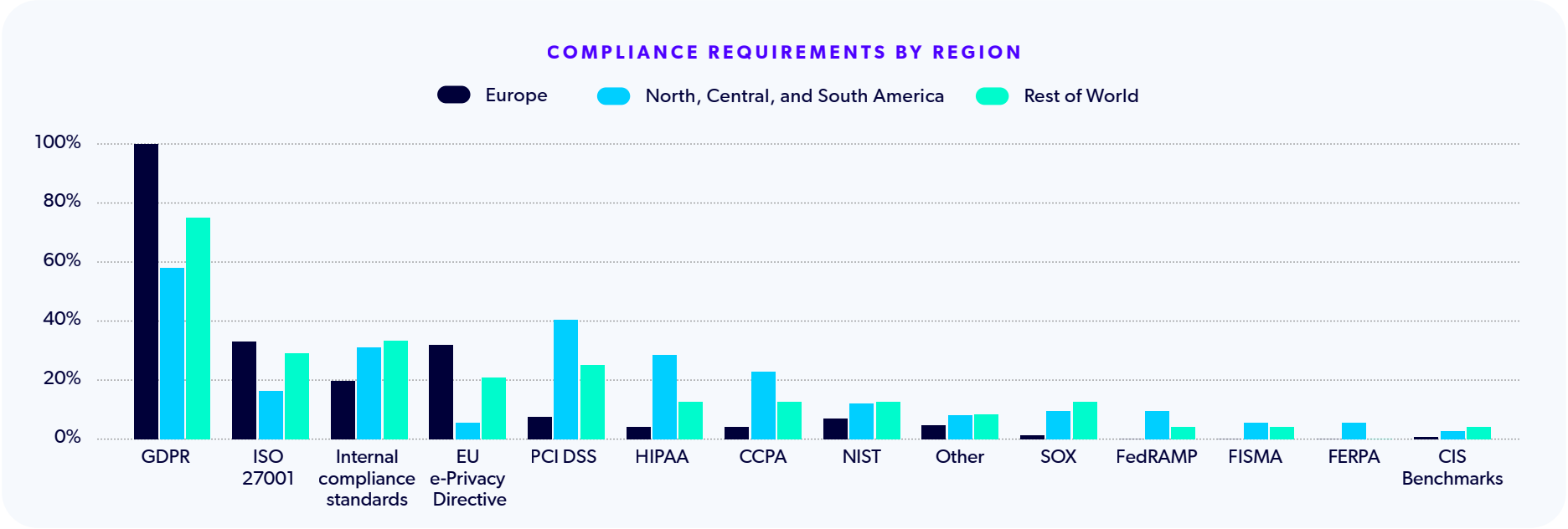
Looking at our findings compared to 2024, we found that internal compliance standards have become more common, rising from 19.68% of teams to 24.07%. Additionally, the EU e-Privacy Directive has become more prevalent in the ecosystem, with 23.70% of participants requiring compliance compared to 2024's 16.13%.

Looking at compliance requirements by region, we were not surprised to find that 100% of our European participants are required to meet GDPR standards. GDPR's prevalence was echoed elsewhere, with 58.11% of participants in North, South, and Central America and 75.00% of participants throughout Asia, Middle East, Africa, and Australia/New Zealand indicating it as their most common compliance requirement.

Key Takeaways

PHP is an increasingly global language, and most developer teams find themselves working with a progressively more complex network of compliance requirements. Government-mandated compliance standards – such as GDPR, ISO 27001, and EU e-Privacy Directives – still set the curve. However, due to the increasing amount of internal compliance standards, we can surmise that companies are taking matters into their own hands by creating internal standards that match their exact regulatory needs and business goals.

Compliance with regulations and standards is never optional. Teams should proactively work to meet all necessary requirements and maintain compliancy within their PHP applications, particularly when those applications handle sensitive customer data.



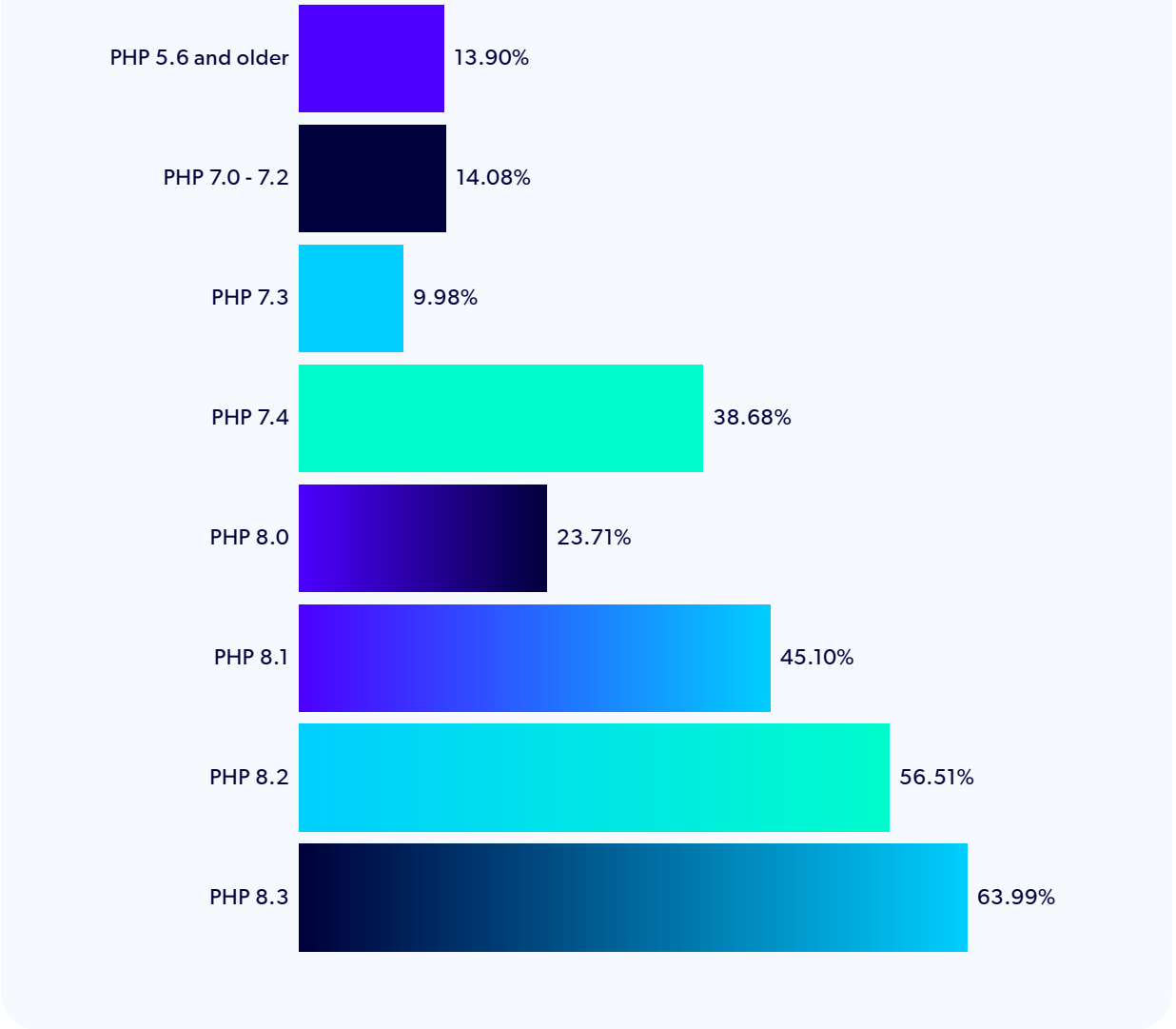
PHP Version Adoption and Migration Trends

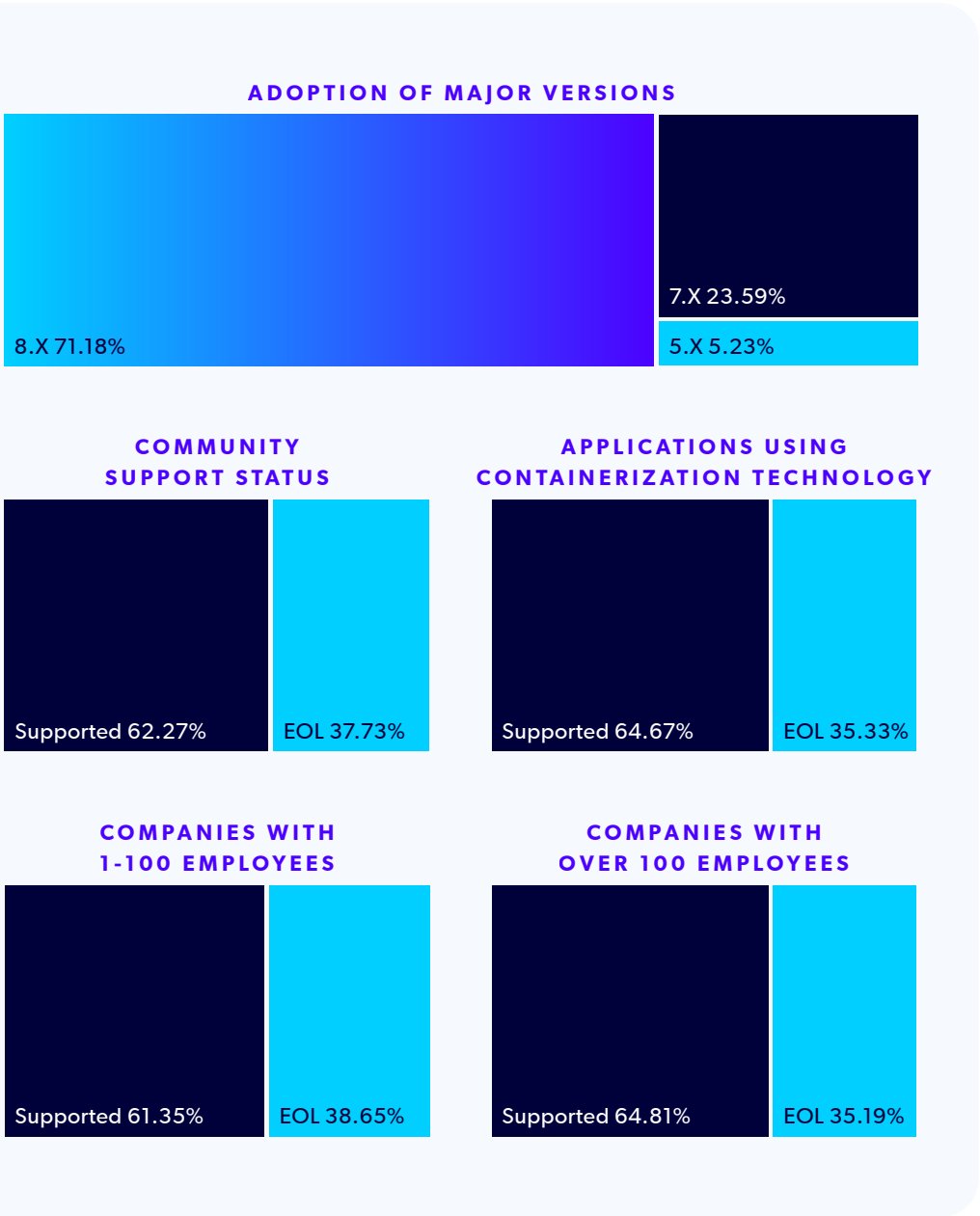
PHP sets an aggressive pace – releasing a new major or minor version every year and making it essential to track ongoing and shifting trends surrounding PHP version adoption and migration. This is particularly important in our 2025 report, as 2024 saw a few significant changes to the PHP release cycle, and we expanded our questions to fully understand the impact of these shifts.

PHP Version Adoption

We began this section by asking our participants to share which PHP versions their applications use, with the option to select multiple versions as applicable. On average, we found that teams were using 2.66 different PHP versions. This was higher than our 2024 survey, which found teams used an average of 2.43 different PHP versions, but aligned closely with our 2023 results, where participants used an average of 2.65 versions.

The most popular PHP version was PHP 8.3, at 63.99% of participants. This was followed by PHP 8.2 (56.51%), PHP 8.1 (45.10%), and PHP 7.4 (38.68%).





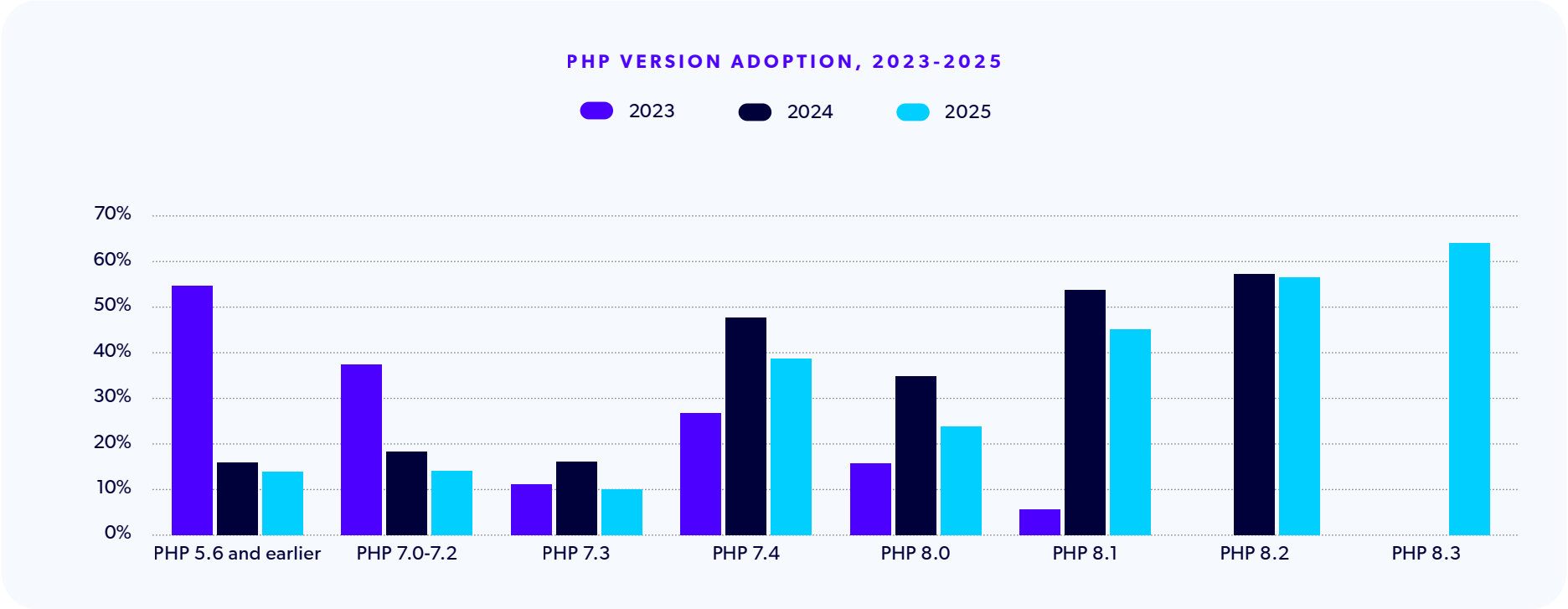
Looking at PHP adoption by major version, we found that 71.18% of participants were deploying PHP 8.X versions. This was followed by PHP 7.X at 23.59%, and PHP 5.X or older at 5.23%.

We saw a clear trend of teams using PHP versions currently supported by the community, with 37.73% of respondents deploying EOL PHP versions. This is a considerable drop compared to our 2024 findings, where 54.55% of teams were deploying EOL PHP in their applications, as well as our 2023 report with 61% using EOL PHP.

Nearly 38% of PHP teams are deploying EOL PHP versions in 2025.

This trend continued when looking at which versions teams using container technology were deploying, with 64.67% of teams using containers also using PHP versions currently supported by the community. In comparison, 60.67% of teams not using container technology were deploying supported PHP versions.

Segmenting by company size, we found that smaller companies with under 100 employees were more likely to deploy EOL PHP versions compared to their larger counterparts, at 38.65% of respondents. Larger companies with over 100 employees were less likely to deploy EOL PHP at 35.19% of participants.



Key Takeaways

The trends reported in the survey mirror our observations of our own download trends at Zend. We've seen a significant decrease in LTS usage in the past six months, and as reflected in the survey, PHP 8.3 has become the preferred version for users. Some might attribute this to a change in the PHP lifecycle policy enacted this past year: instead of two years of active support followed by one year of security-only support, each version now receives two years of active support and two years of security-only support. However, those changes happened too late to affect this year's survey, as the survey ran before the annual feature release.

Our suspicion is that many of the hurdles of jumping from PHP 7 to PHP 8 are handled in the libraries and frameworks PHP application developers consume, cushioning them from the impact of the upgrade. Within PHP 8 itself, the majority of the deprecations and BC breaks since 8.1 have had limited impact on most applications, making it easier to adopt new releases.

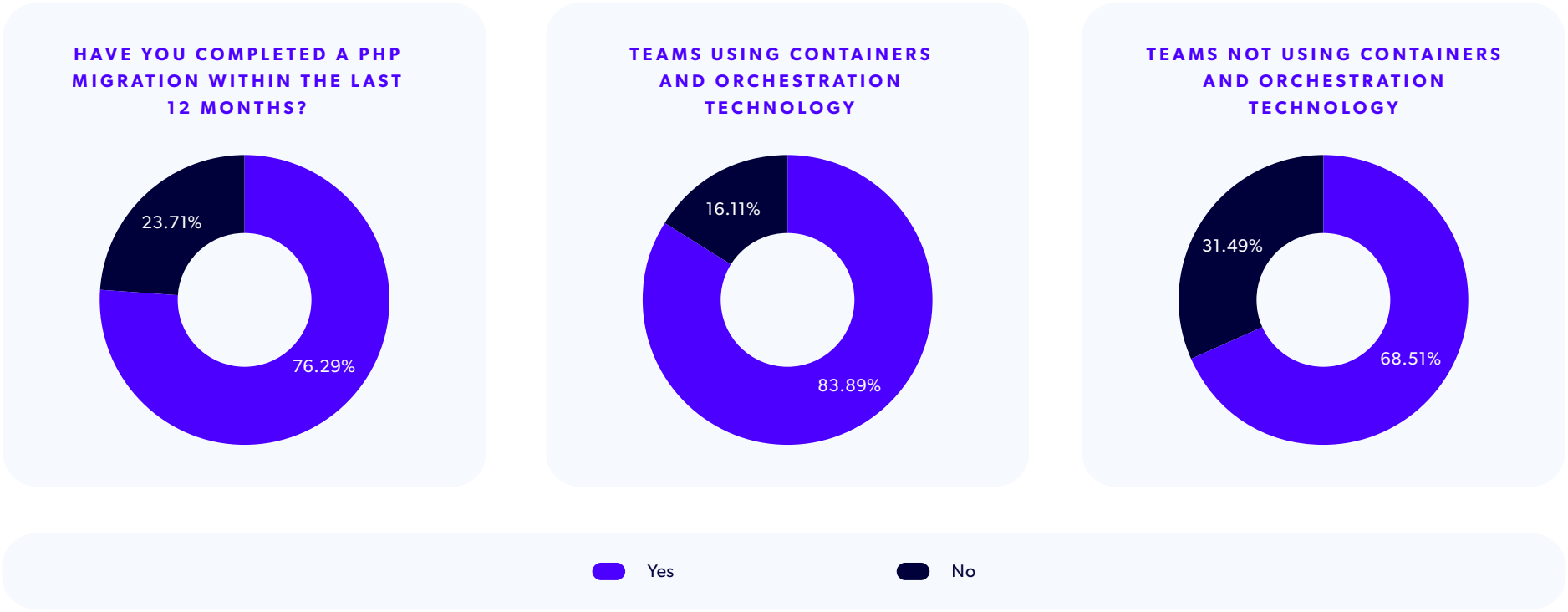
That said, over 37% using EOL versions is still a large number – it's more than one out of three. With the bulk of respondents on EOL versions using PHP 7.4, it's clear that there are still hurdles when migrating to PHP 8 two years after 7.4 reached its end of life.

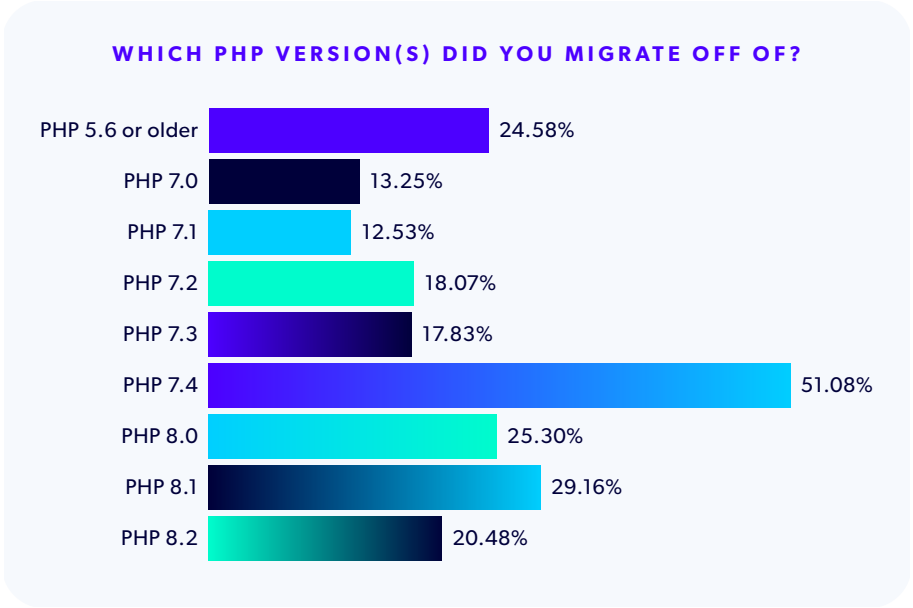
PHP Upgrade and Migration Trends

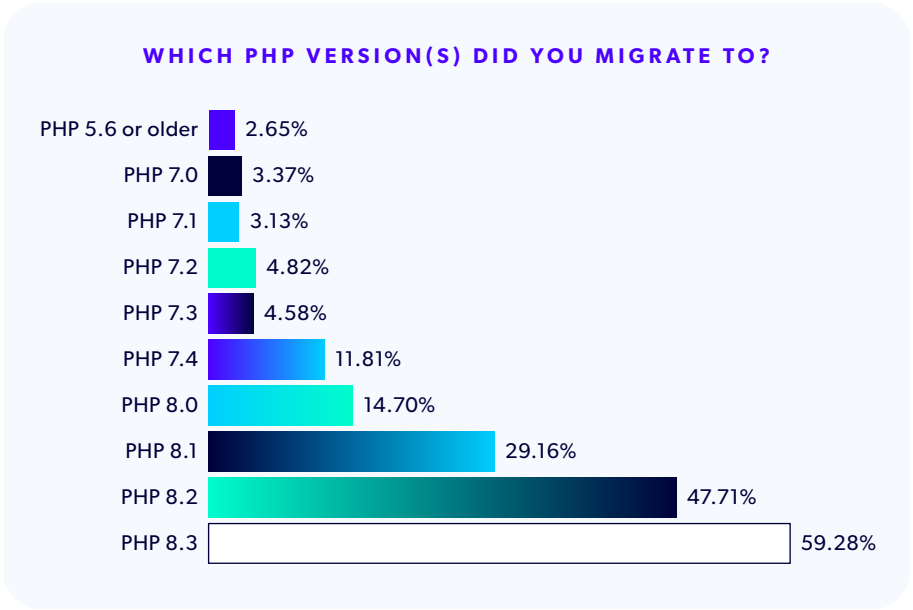
Next, we asked questions surrounding PHP upgrade and migration trends. As the PHP lifecycle changed in 2024, adding an additional year of security support for the community and extending the lifespan of PHP versions through December 31 of their final year of support, we were particularly interested in trends and changes in this area. Overall, we found that the extended release cycle had little noticeable impact on actual migrations or upgrades, with most participants feeling neutral or positive about the changes.

As with previous years, we began this section by asking teams if they had completed a PHP migration within the last 12 months, with 76.29% answering yes. This number rose significantly when looking at teams currently using container and/or orchestration technology, with 83.89% of participants having completed a migration in the past 12 months. Only 68.51% of teams not using container technologies completed a migration in the past 12 months.

Overall, teams were more likely to have completed a PHP migration in the past 12 months compared to 2024, at 76.29% vs. 72.28%. We believe this speaks to the overall improvements in the language and ease of migrations to PHP 8.X versions compared to previous iterations.







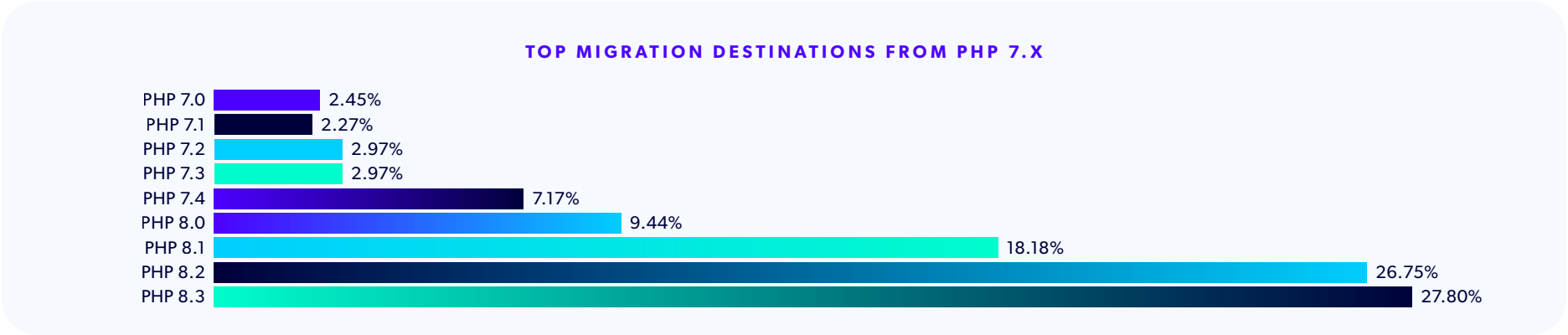
Migration Trends by PHP Version

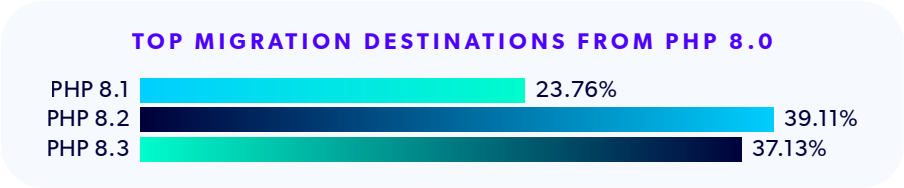
Next, we asked teams that had completed a migration in the past 12 months which PHP versions they migrated off of, and which PHP versions they migrated to – both with the option to select multiple versions as applicable.

The top PHP version participants migrated off of was PHP 7.4 at 51.08%. This was followed by PHP 8.1 at 29.16%, PHP 8.0 at 25.30%, PHP 5.6 or older at 24.58%, and PHP 8.2 at 20.48%.

The top PHP version teams migrated to was PHP 8.3, the most current version at the time of the survey, at 59.28%. It was followed by PHP 8.2 at 47.71%, PHP 8.1 at 29.16%, PHP 8.0 at 14.70%, and PHP 7.4 at 11.81%.

PHP 8.0 outpaced PHP 7.4 as a migration destination, with 14.07% of teams migrating to PHP 8.0 vs. 11.81% of teams migrating to PHP 7.4.





For teams migrating from PHP 7.X versions, PHP 8.3 was the top destination with 27.80% of respondents, followed by PHP 8.2 (26.75%), PHP 8.1 (18.18%), and PHP 8.0 (9.44%). For teams migrating from PHP 8.0, PHP 8.2 moved to the top destination at 39.11% of teams, followed by PHP 8.3 at 37.13% and PHP 8.1 at 23.76%. Teams migrating from PHP 8.1 saw a closer split, with 53.18% migrating to PHP 8.3 and 46.82% migrating to PHP 8.2.

Key Takeaways

When on older PHP versions, a common pattern is to upgrade first to PHP 7.4, and then to a more recent, supported PHP 8 version. The fact that PHP 7.4 was a destination for less than one out of eight respondents performing migrations, and that every PHP 8 version had higher adoption rates, tells us that PHP application developers are ready and eager to adopt the latest releases.

We were very surprised, however, to see that fully one out of four migrations came off of PHP 5.6 or older, considering that PHP 5.6 reached its end of life nine years ago. Clearly, many respondents have been running PHP applications for quite some time!

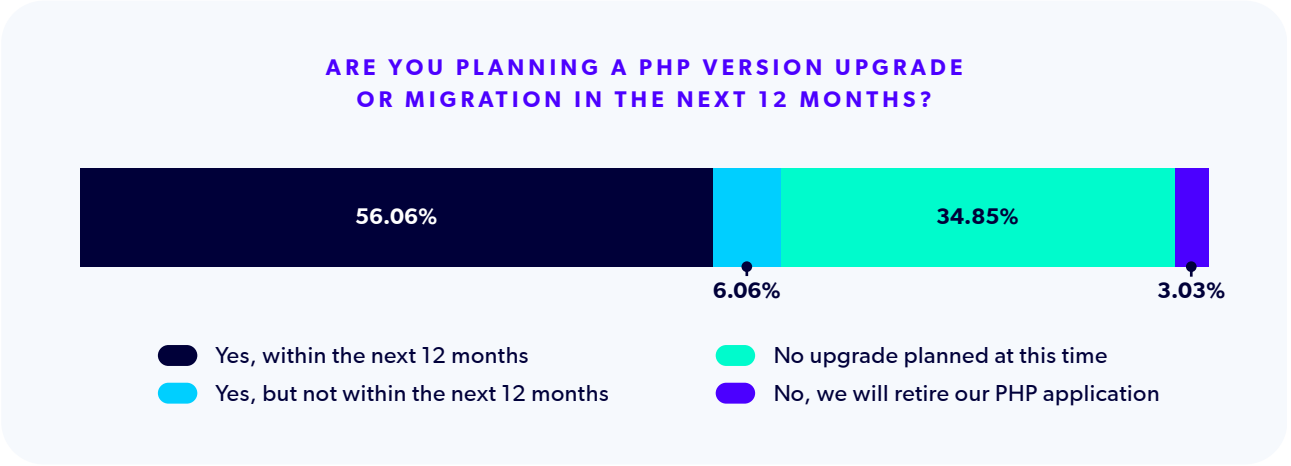
We also want to point out again the importance of containers in the PHP ecosystem: teams using containers were far more successful in completing migrations in the past 12 months. The ability to quickly test and validate applications against new versions clearly has an impact on success.

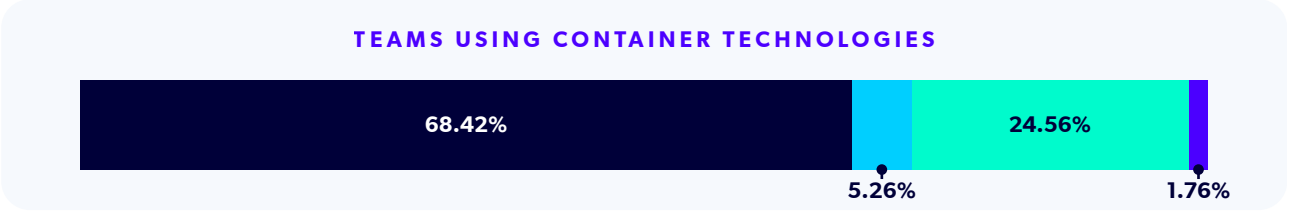
Forecasting 2025 PHP Upgrade and Migration Trends

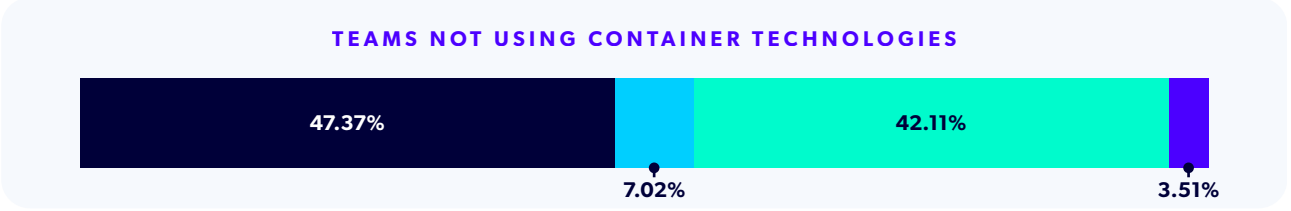
In our next question, we asked our respondents if they were planning a PHP version upgrade or migration within the next 12 months. 56.06% answered yes, they were planning an upgrade or migration within the next 12 months, with an additional 6.06% of teams answering yes, they were planning an upgrade or migration but not within the next 12 months. 34.85% had no upgrade planned at this time, and 3.03% planned to retire their PHP applications.

Segmenting our data, we found that teams using container technologies were more likely to have a migration planned in the next 12 months, with 68.42% of teams vs. 47.37% respectively. However, teams not using container technologies were slightly more likely to have an upgrade planned, but not in the next 12 months (7.02% vs. 5.26%).

Finally, as PHP 8.1 reaches end of life on December 31, 2025, we looked at migration and upgrade plans for teams currently deploying PHP 8.1. 60.42% had an upgrade planned in the next 12 months, with an additional 6.25% planning an upgrade, but not within the next 12 months.







Migration Pain Points

In our final migration and upgrade question, we asked participants about the most time-consuming components of their last PHP upgrade. Of identified pain points, 38.21% identified Testing, followed closely by 35.57% choosing Refactoring. At a significantly smaller percentage, 9.15% selected Infrastructure Provisioning, followed by Planning (8.74%), Other (4.47%), and Compliance Renewals (3.86%). Repeated write-in answers included dependency management, unmaintained or un-updated libraries, debugging, and addressing deprecations between PHP versions.

When segmenting by company size, we found that smaller companies with less than 100 employees were more likely to identify Testing as the top pain point with 39.07% of participants, followed by Refactoring at 34.99%. For larger companies, Refactoring was a significantly prominent pain point at 45.39% of respondents, followed by Testing at 24.91%.

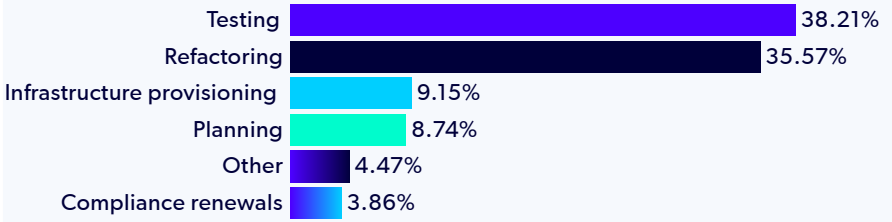
Key Takeaways

The best way to ensure success when performing a PHP migration is to have a robust test suite. Unfortunately, it's often hard to identify migration pain points in test suites: is a new error due to application logic, or a backwards incompatible change introduced in the language? Worse, if its due to a language change, is there a clear path to fixing the code to accommodate the change, or will you need to refactor substantially?

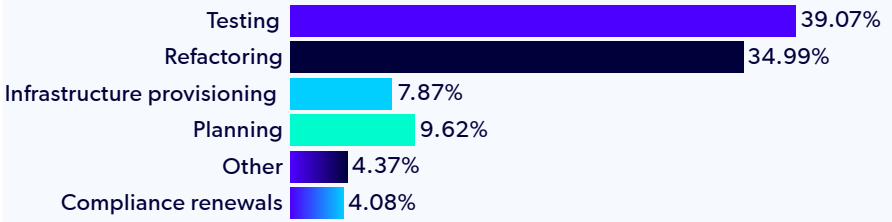
An interesting aspect when migrating to a new version is whether or not to take advantage of new language features. Many of the new language features in PHP 8 allow developers to write more robust code — but doing so will often require substantial changes. As such, many organizations may choose to leave code as-is, to prevent introduction of new issues.

With this in mind, we were a bit surprised to see Refactoring account for as much pain as it did, as this tells us that developers are willing to risk that trade-off due to the longer term benefits the refactor will provide.

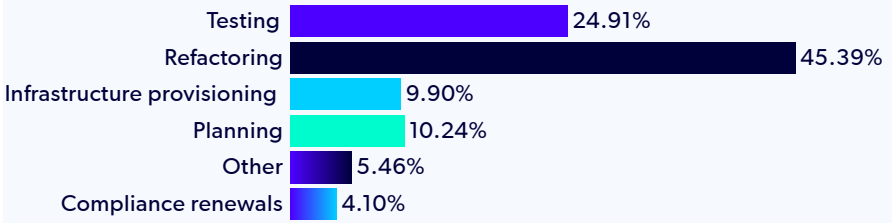
WHAT WAS THE MOST TIME-CONSUMING COMPONENTS OF YOUR LAST PHP UPGRADE?



COMPANIES WITH 1 - 100 EMPLOYEES



COMPANIES WITH OVER 100 EMPLOYEES



Development Priorities

In this section, we looked at the evolving development priorities for PHP teams. For our 2025 survey, we expanded this section to include new questions examining administrative priorities alongside development and maintenance priorities, allowing us to build a fuller picture of the PHP landscape.

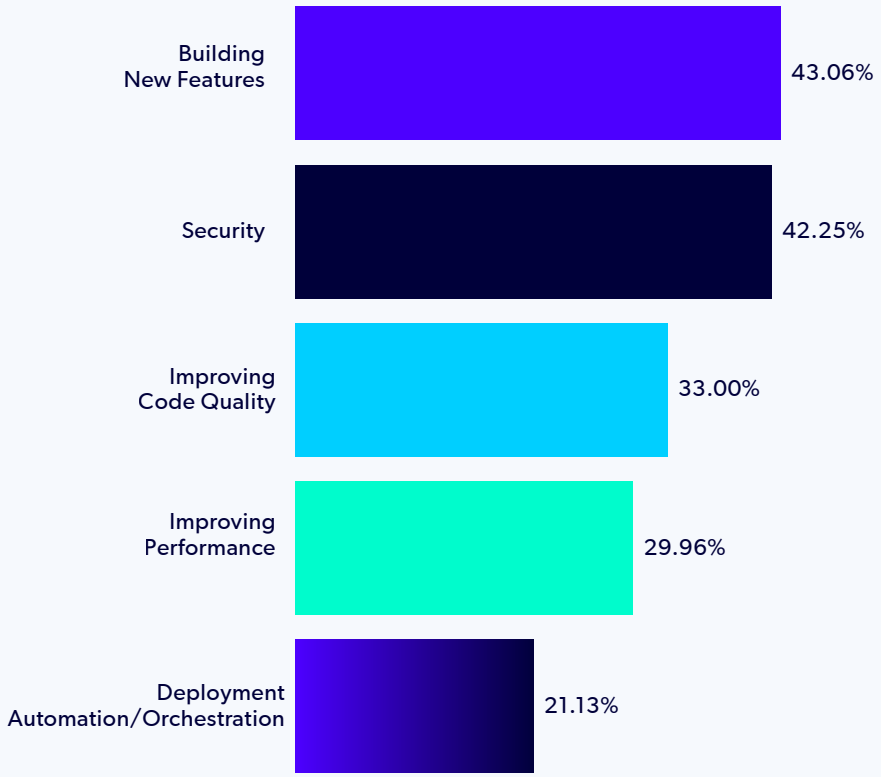
Top Development Priorities

In our first question of the section, we asked teams about their top development priorities in 2025. 43.06% answered that their biggest priority was Building New Features, followed by Security (42.25%), Improving Code Quality (33.00%), Improving Performance (29.96%), and Deployment Automation/ Orchestration (21.13%).

Building New Features has been selected as the top development priority for PHP teams for 4 years and counting.

These results echo our 2024 findings, though Improving Code Quality did rise in priority, moving from 22.36% of respondents to 33.00% in 2025. Additionally, Improving Performance rose from 18.78% of participants to nearly 30%.

WHAT ARE THE TOP DEVELOPMENT PRIORITIES FOR YOUR TEAM IN 2025?

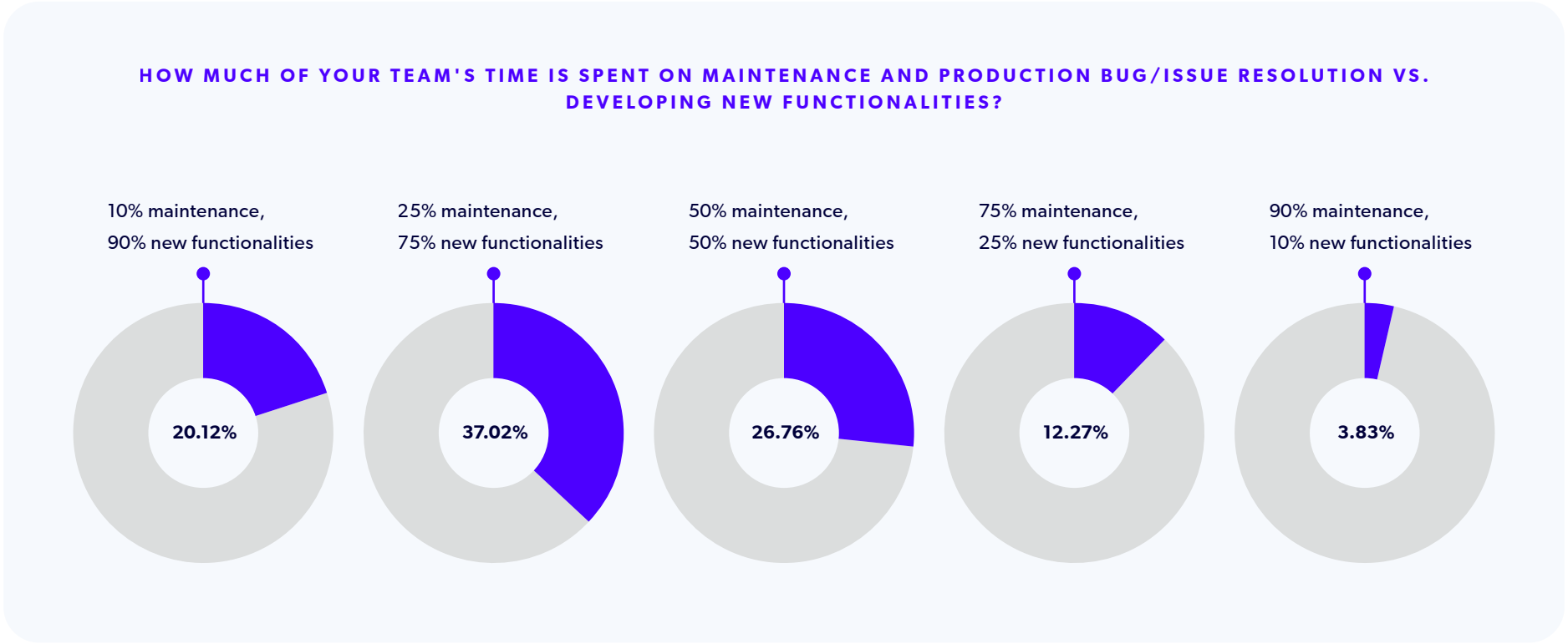


Feature Development vs. Maintenance and Administration

Next, we asked participants to estimate how much of their team's time is spent on maintenance and production bug/issue resolution vs. developing new functionalities.

The top selected response was "25% maintenance, 75% new functionalities" at 37.02% of respondents. This was followed by "50% maintenance, 50% new functionalities" at 26.76%, "10% maintenance, 90% new functionalities" at 20.12%, "75% maintenance, 25% new functionalities" at 12.27%, and "90% maintenance, 10% new functionalities" at 3.83%.

While we saw a similar order of results in 2024, the percentages have shifted in 2025. A higher percentage of teams (20.12% in 2025 vs. 14.77% in 2024) split their time between 10% maintenance, 90% new functionalities. However, we also saw an increase in teams (12.27% vs. 9.07%) splitting their time between 75% maintenance, 25% new functionality.



In a new question, we asked participants how many hours per week on average their team spends in the administration of their PHP applications. Nearly half of respondents reported spending less than 5 hours per week on PHP application administration, followed by 5-10 hours (26.25%), more than 30 hours (9.79%), 11 to 20 hours (8.83%), and teams who outsource their administration (2.14%).

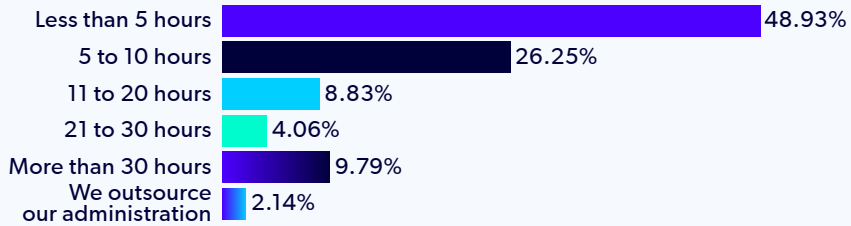
Looking at hours per week spent on administration by company size, we found that smaller companies with under 100 employees were more likely to spend less than 5 hours per week than larger companies with over 100 employees, at 50.66% of participants vs. 44.44% respectively.

51.08% of surveyed PHP teams spend 10+ hours per week on the administration of PHP applications.

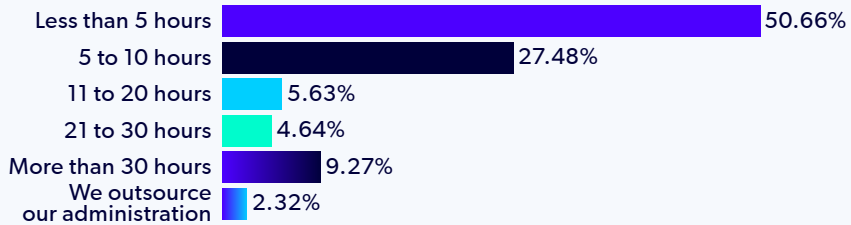
Key Takeaways

PHP is a fantastic choice for delivering business value, allowing developers to keep companies productive while engaging their own customers. For teams spending a significant amount of time on administration – while being asked to prioritize new feature development – outsourcing administration can help meet demands without hiring additional developers. With macro conditions in the development space forcing companies to do more with less, we expect to see more teams outsourcing PHP application administration in the coming year.

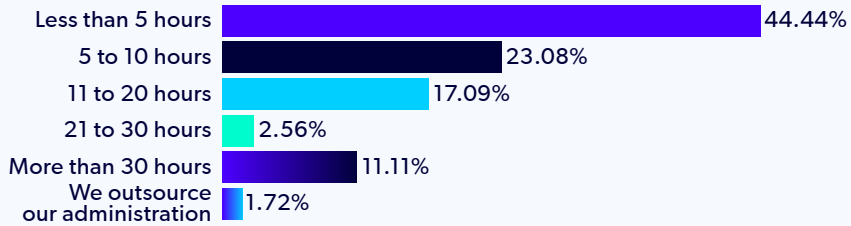
ON AVERAGE, HOW MANY HOURS PER WEEK DOES YOUR TEAM SPEND IN THE ADMINISTRATION OF YOUR PHP APPLICATIONS?



COMPANIES WITH 1 - 100 EMPLOYEES



COMPANIES WITH OVER 100 EMPLOYEES



Identifying and Addressing Development Issues

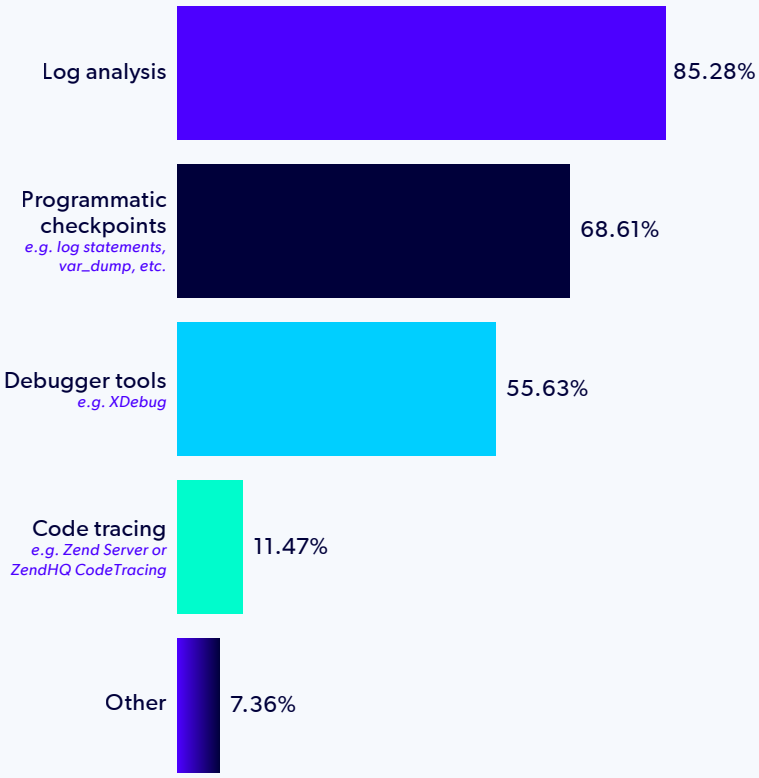
In another expanded section, we asked participants how they identify and resolve production issues in their PHP applications. The top method was through Log Analysis with 85.28% of teams, followed by Programmatic Checkpoints (68.61%), Debugger Tools (55.63%), Code Tracing (11.47%), and Other (7.36%). Repeated write-in options included ongoing testing, custom alerting systems, and use of tools such as Sentry or Prometheus.

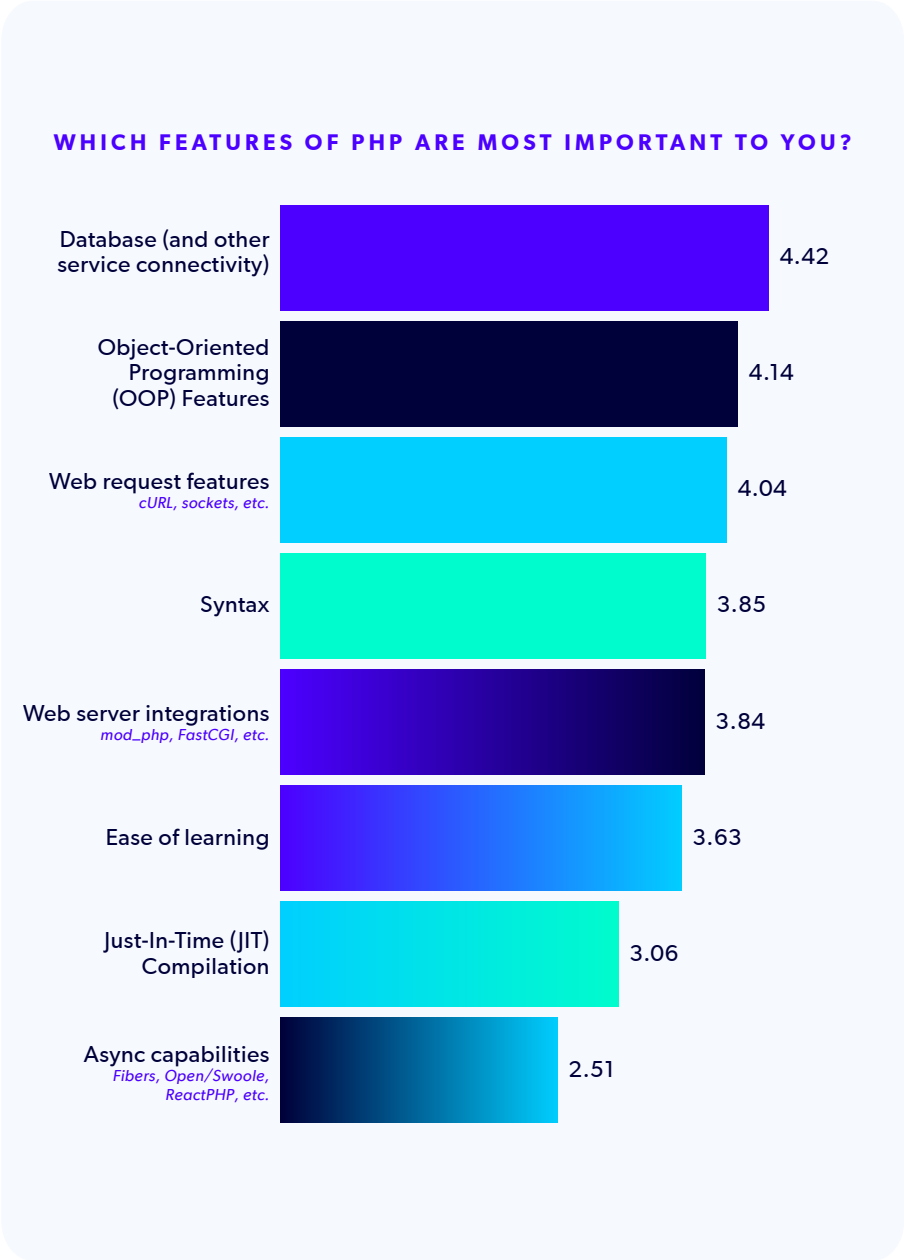
Key Takeaways

Because PHP is used for powering web applications, finding the root cause of production issues can be difficult to trace. Logs can provide pointers, but they often will not contain full context for whatever triggered the log. Additionally, understanding what to log and how to aggregate and interpret log results can require specialized skills.

On the flip side, dedicated debuggers can give this information, but can degrade performance significantly, making them a poor fit for production systems.

We expect that with the rising sophistication of PHP applications, PHP users will start to realize immense value from dedicated tools for tracking production issues.





The State of PHP in 2025

In the final section of our 2025 survey, we asked participants questions exploring the importance of certain PHP features, challenges teams face working with PHP, and observations on the current state of the language.

Most Important PHP Features

We began by asking participants to rank PHP features on a scale of 1 – 5, with 1 being the least important and 5 being the most important.

We found that Database and Other Service Connectivity was rated as the most important feature of PHP at a weighted average of 4.42. Object-Oriented Programming came in second at 4.14, and Web Request Features rounded out the top three at 4.04. Syntax and Web Server Integration nearly tied at 3.85 and 3.84 respectively, followed by Ease of Learning (3.63), Just-In-Time (JIT) Compilation (3.06), and Async Capabilities (2.51).

Key Takeaways

As in previous years, PHP's ability to connect with other databases, services, and systems is an important feature for users of the language. While performance features such as JIT and async programming are often touted as marquee features of new PHP releases, they are the features that are ultimately least important to most PHP users.

Biggest PHP Challenges

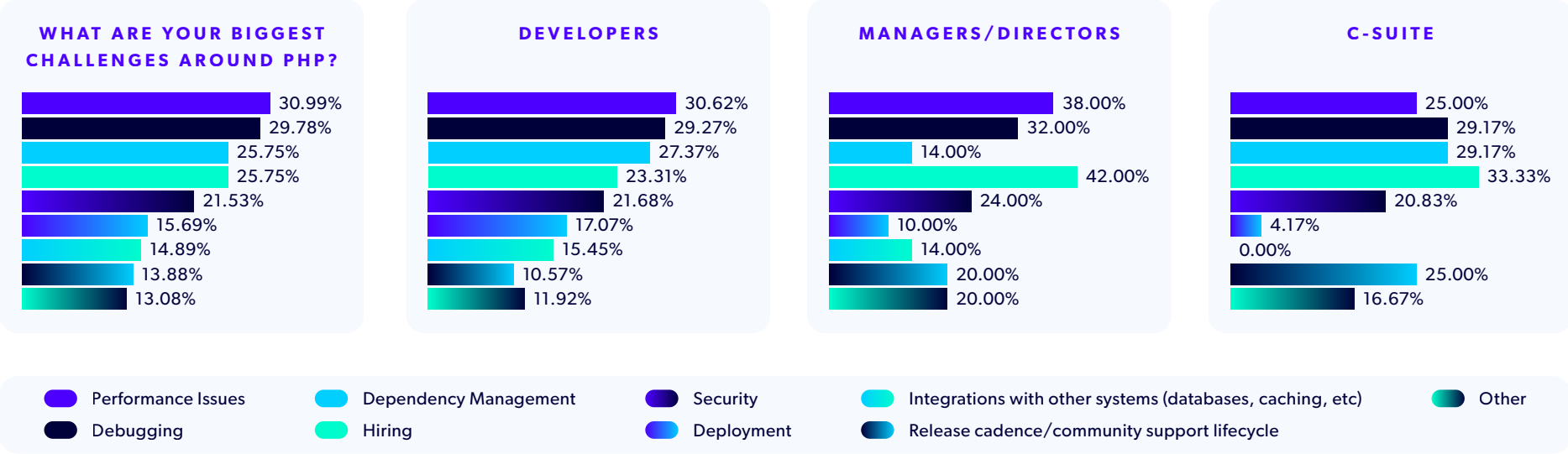
We next asked our survey participants to identify the biggest challenges they face while working with PHP, with the option to select multiple choices as needed. 30.99% of respondents chose Performance Issues as the top challenge of working with PHP, followed closely by Debugging at 29.78%, with Dependency Management and Hiring tying for third at 25.75% each.

While the top two challenges remain consistent with our 2024 findings, Hiring has grown as a challenge among participants, moving up from fifth position to third this year. When examining the data by job title, we found that Hiring is the top challenge for PHP professionals in Manager/Director and C-Suite roles, whereas Developers are more likely to follow the broad data trends.

Key Takeaways

Nearly 61% of participants agree that Performance Issues and Debugging are top challenges around PHP. With PHP developer teams tending to be small and operating on tight budgets, and with those teams spending significantly more time developing new features vs. maintaining or administrating their applications, we are unsurprised to see these as top issues. Scaling PHP often requires specialized skillsets, as does root cause analysis of production issues.

Additionally, we are interested in the continued rise of Hiring as a challenge in PHP, as we first noted its rise in our 2024 report. Considering the fact that we saw the average PHP experience of respondents at over 10 years, one question we have is whether there are not enough new PHP developers in the market as a pool to hire from. We hope that efforts initiated by The PHP Foundation this coming year will help create new talent in the industry.



The State of PHP

In the final question of the 2025 PHP Landscape Survey, we asked our participants to rank their level of agreement with four statements about PHP on a scale of 1 – 5, with 1 signifying no agreement and 5 signifying total agreement.

We found that most of the surveyed teams strongly agreed with the following statements: I enjoy working with PHP; If I were designing a new web application, I would use PHP; PHP is easy to learn. The majority of participants disagreed with the statement that PHP is evolving too slowly.

Compared to our 2024 findings, the sentiments surrounding these four statements have stayed consistent year over year, with very little deviation in the weighted averages.

Key Takeaways

30 years in, PHP is a vibrant language that developers like working with. Efforts made by the language developers in the past 10 to 15 years have led to steady improvements, which PHP users appreciate and tend to adopt at an increasing pace. With web applications becoming the de facto standard for both customer-facing and internal business applications, it's an ideal choice for organizations that want to deliver features quickly, securely, and at scale.

About Zend

Perforce Zend helps organizations use enterprise PHP to build innovative web and mobile solutions and modernize existing applications. Used by multiple Fortune 100 companies, our proven enterprise PHP offerings include secure, fully-supported PHP runtimes, software infrastructure, tools, professional services, and enterprise long-term support for PHP 7.2, 7.3, 7.4, and 8.0. For more information, visit zend.com today.

Learn More About Secure and Supported PHP Runtimes

Discover ZendPHP ▶

zend.com/products/zendphp-enterprise

Enterprise-Ready Professional PHP Services

Explore Professional Services ▶

zend.com/services

About Perforce

The best-run DevOps teams in the world choose Perforce. The Perforce suite is purpose-built to develop, build, and maintain high-stakes applications. Companies can finally manage complexity, achieve speed without compromise, and run their DevOps toolchains with full integrity. With a global footprint spanning more than 80 countries and including over 75% of the Fortune 100, Perforce is trusted by the world's leading brands to deliver solutions to even the toughest challenges. Accelerate technology delivery with no shortcuts.

Learn More ▶

perforce.com/support