zend
by Perforce

2022 PHP Landscape Report

Top PHP Technologies, Strategies, and Trends

Introduction

EXECUTIVE SUMMARY

What is the current state of PHP, and where is it heading? This report looks at the results of an industry survey of PHP development professionals and covers hot topics in PHP development, PHP versions, development and orchestration technologies, as well as async programming.

FOREWORD

To the reader,

It's been a big year for PHP!

It's been the first full year with a new major version of the language since 2015. PHP 8.0 offered increased performance, a Just-In-Time (JIT) compiler, and a plethora of new features to ease common development patterns as well as strengthen runtime validations. In November 2021, the first new minor release in the series (8.1) dropped, adding first-class support for async runtimes at the language level via Fibers, and more type system improvements including enums, read-only properties, and intersection types. The language is maturing at an incredible pace!

On top of that, we saw the formation of The PHP Foundation. This new foundation aims to provide funding for the development of the language by providing grants and employment for PHP internals developers. One specific goal is to help train the next generation of PHP internals developers, ensuring the language's continuity and evolution.

As this report demonstrates, businesses are building business-critical web applications and APIs with PHP. PHP's ease of use, ease of deployment, and easily accomplished scalability make it a natural fit for these use cases. APIs, in particular, allow them to integrate with other languages and systems, and other businesses. Add to that newer features such as async capabilities and JIT, and you have the means for massive scale and new computational use cases such as Machine Learning. PHP is not only a good choice today, but for the future.

That said, these capabilities do not exist in a vacuum. To achieve scale, and to integrate with complex systems, testing and deployment methodologies need to become more sophisticated. As with the rest of the software industry, containerization and orchestration tooling become increasingly important to automate these processes with confidence. Interestingly, as you'll see from our survey results, we're seeing larger companies adopting these practices at a faster rate than the rest of the industry.

This is an exciting time for PHP developers, and I'm looking forward to what the future has in store!

Enjoy the report,

Matthew Weier O'Phinney

Table of Contents

About the Survey

The Zend 2022 PHP Landscape Report is based on the results of an anonymous survey conducted between the months of October and December of 2021.

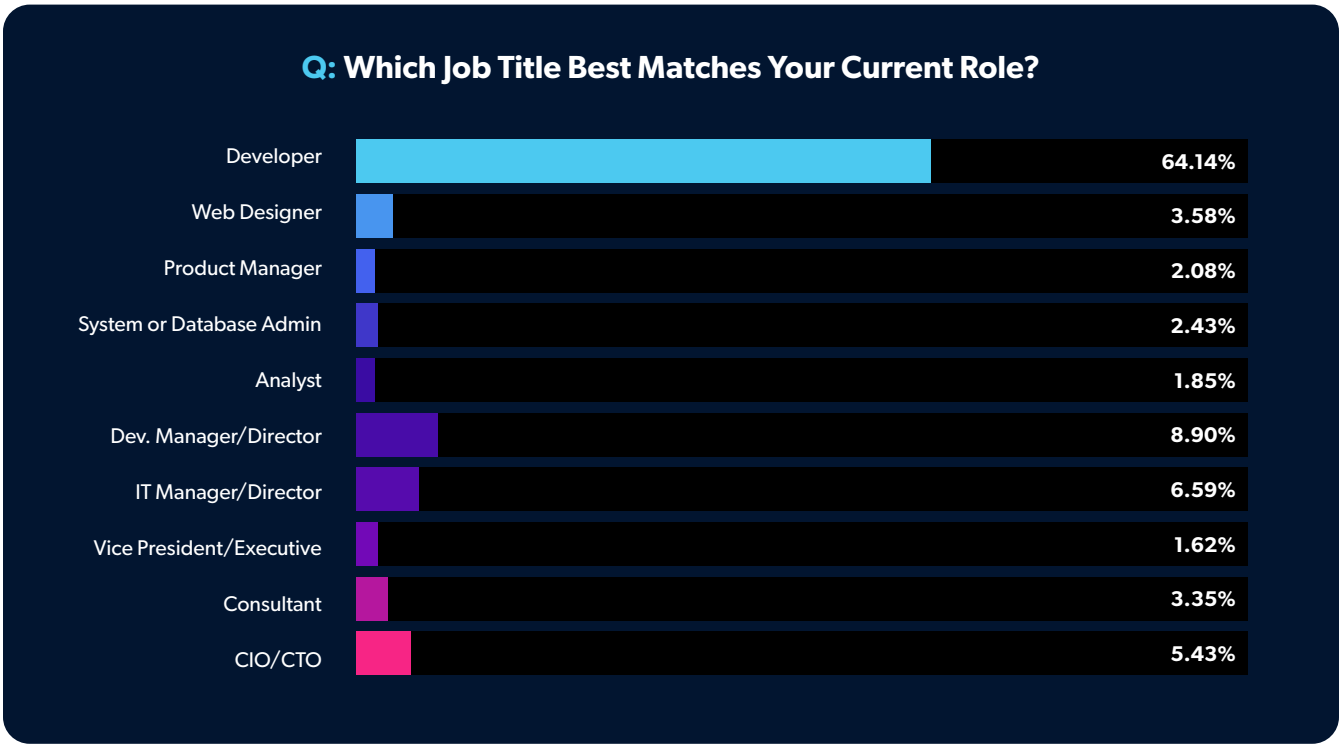
The survey, which was promoted via social media and email, focused primarily on how PHP development teams are working with the language, their priorities in PHP development, versions they use and plans for upgrades, and the technologies they use for development, orchestration, and async programming. The survey received a total of 865 responses.

REPORTING & ANALYSIS METHODOLOGY

To help increase report readability, we rounded percentage values to the nearest full percentage point in our analysis. Year on year comparisons, where applicable, used data from the Zend 2021 PHP Landscape Report.

RESPONDENT DEMOGRAPHICS

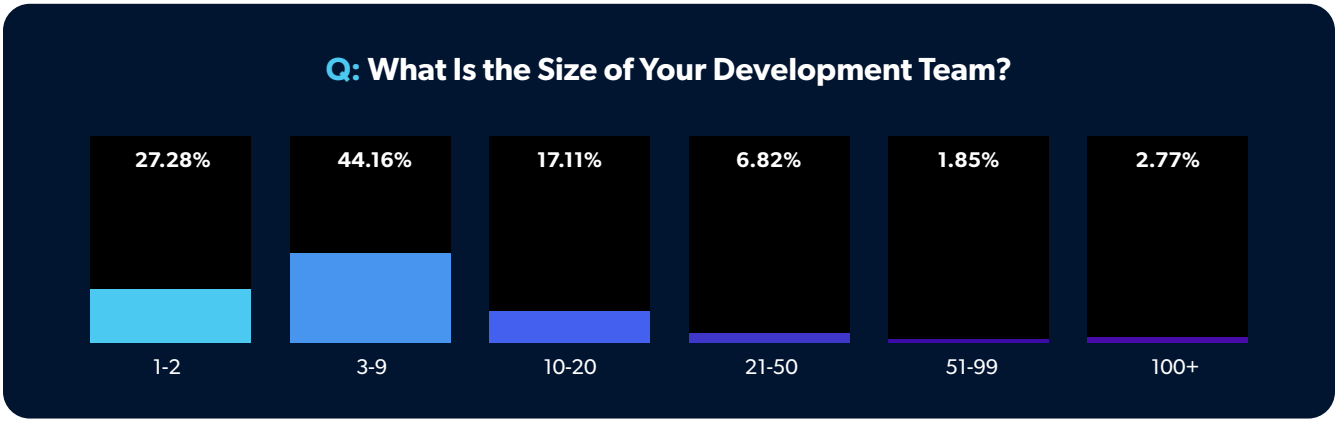
Respondents were primarily developers, who represented 64% of respondents. There were also a fair amount of decision makers among the respondents, with Development Manager/Director comprising 9%, IT Manager at 7%, and CIO/CTO at 6%.



DEVELOPMENT TEAM SIZE

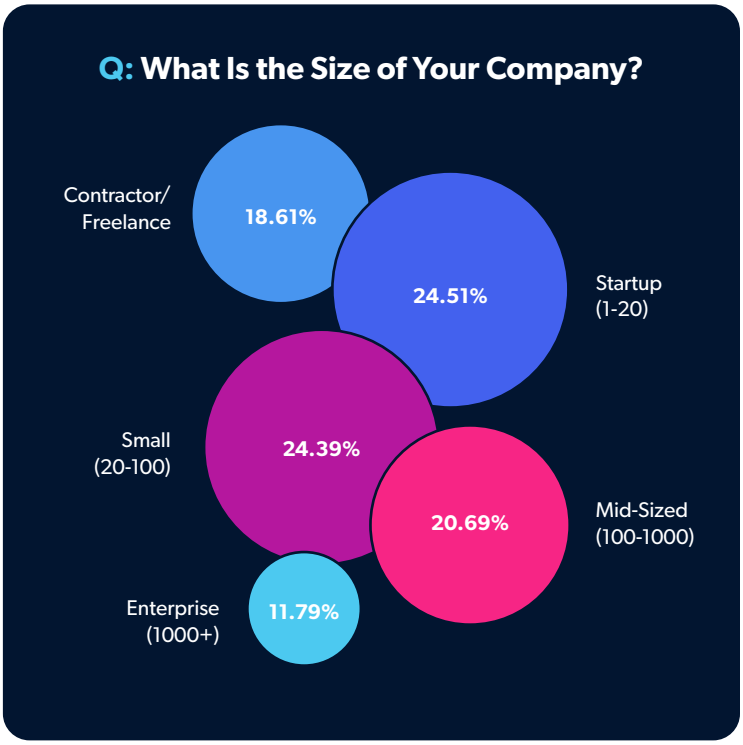
In our next question, we asked respondents to share the size of their development team. The largest segment of respondents, at 44%, reported working with a development team size of 3-9 members. This is not surprising; if a team size exceeds that upper parameter, you typically have to break the team into smaller units to achieve parallel progress on your initiatives.

The next largest segment worked on smaller teams, with 27% reporting a team size of 1-2 developers. Those working on development teams of 10-20 developers represented 17% of respondents.



COMPANY SIZE

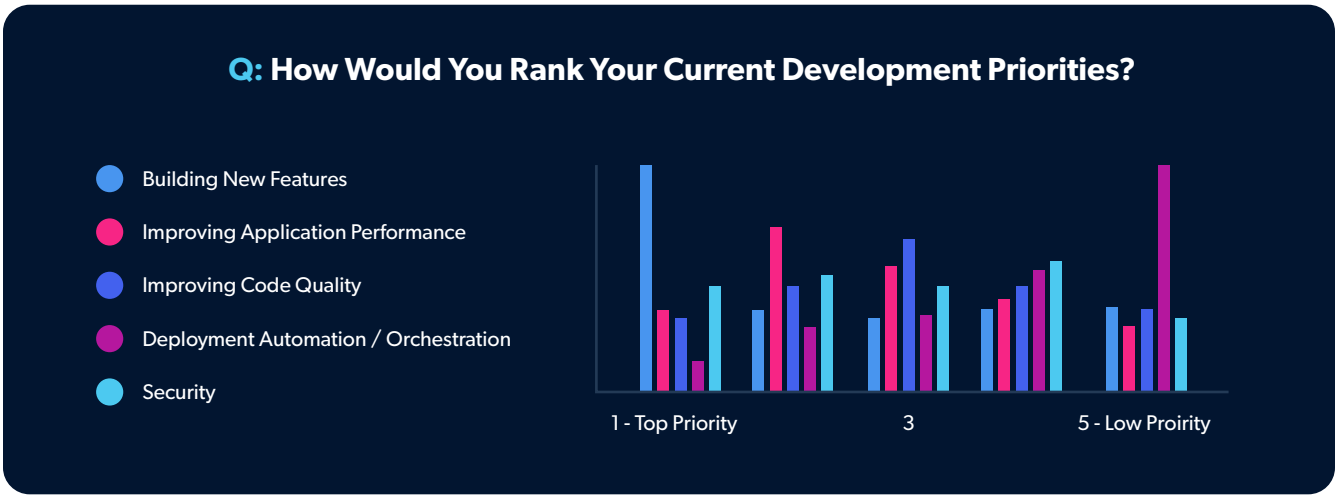
In this question, we asked respondents to share the size of their company. Our survey found a relatively even distribution across the top four answer choices, with 25% working for companies with between 1-20 employees, 24% working for companies with 20-100 employees, 20% working for mid-sized companies, and 19% working for startups. This distribution provides a general overview of what companies of all sizes are doing in the PHP ecosystem.



PHP Development Priorities

In this question, we asked respondents to rank their current development priorities, with choices that included building new features, improving application performance, improving code quality, deployment automation, and security.

Our survey found that the majority of development teams (46%) prioritized building new features on their current project. Security was another top priority, with 19% of teams reporting it as their top priority for ongoing development. Improving application performance (15%) and improving code quality (14%) followed close behind.



Deployment automation and orchestration has consistently been ranked as the lowest priority in each year of this survey. However, it is really tied into the other priorities. Code quality can't be improved without test automation. This is also true when building out new features — they cannot be deployed with confidence without having automation pipelines in place. Applying security patches means being able to test the application, which includes an automation pipeline. For many, automation is set up once, then forgotten and becomes a low priority — but it is still baked into all the other priorities listed.

WHAT ARE TEAMS DEVELOPING WITH PHP?

In this question, we asked respondents to pick which type of application or system they are currently developing with PHP. Respondents were allowed to select more than one answer choice.

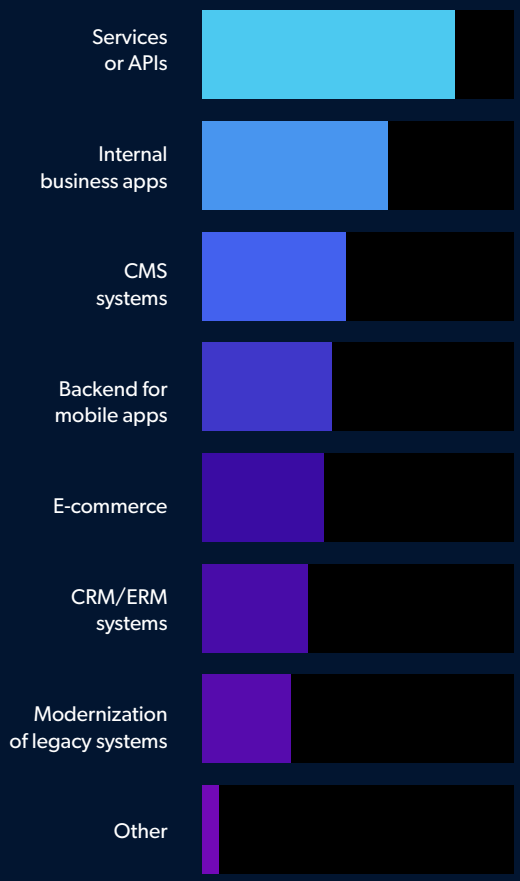
Our survey found 81% of respondents using PHP for services or APIs, nearly doubling last year's 41%. Second on the list was internal business applications with 59%, followed by CMS systems at 46%.

While API development continues to be the leading use case for PHP, the key statistic is the usage of PHP for internal business applications.

PHP has clearly become a part of the enterprise ecosystem, with organizations realizing those applications can be built in PHP instead of other languages. PHP isn't just being used for CMS or CRM systems anymore — the bulk of PHP use today is for business-critical applications.

This belies that PHP is dead or nobody wants to teach PHP anymore. PHP is clearly driving business.

Q: What Are You Using PHP For? (Select All That Apply.)



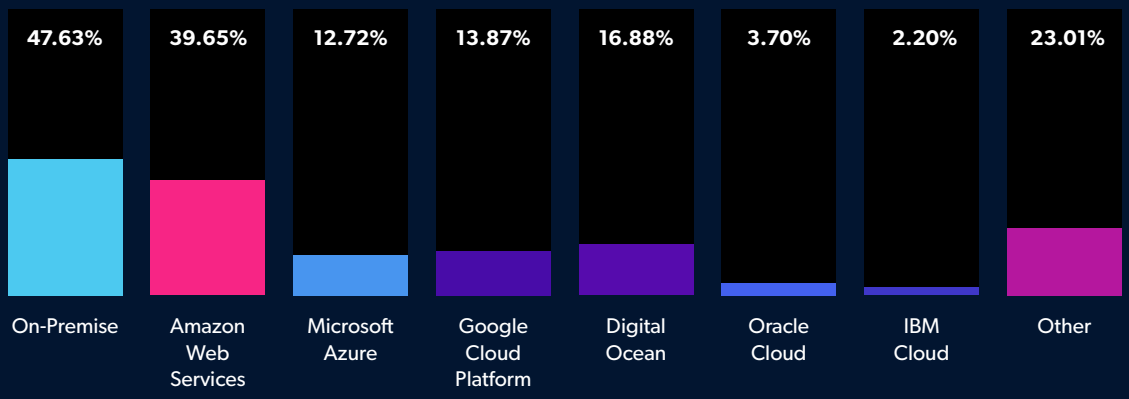
WHERE ARE TEAMS DEPLOYING THEIR PHP APPLICATIONS?

The next question asked where teams are deploying their applications.

Nearly half (48%) report on-premise. This number is likely even higher, as many write-in responses indicated "self-hosted," which would fit under the label of on-premise.

Amazon Web Services followed with 40%, trailed by Digital Ocean with 17%, Google Cloud Platform with 14%, and Microsoft Azure with 13%.

Q: Where Are You Deploying Your PHP Applications? (Select All That Apply.)

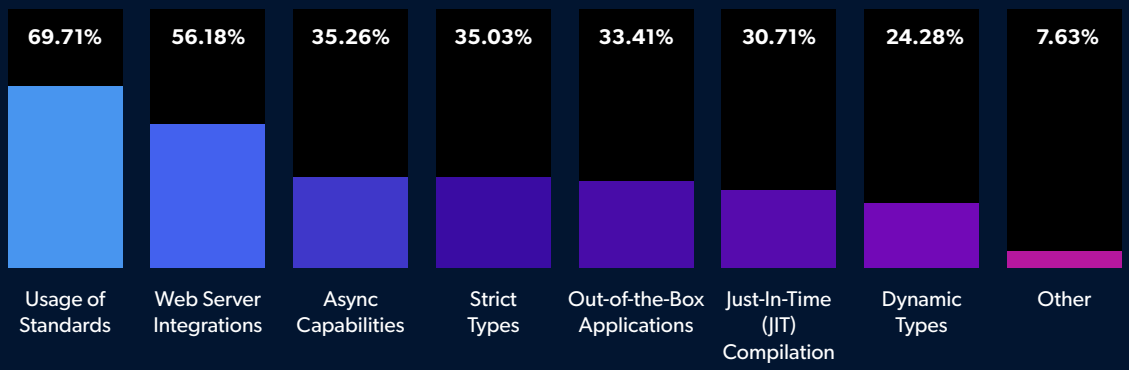


MOST VALUABLE PHP FEATURES

In this question, we asked what makes PHP valuable for new projects and/or refactoring old projects. Respondents were able to select more than one answer choice.

Usage of standards (such as PHP Standards Recommendations) was the most popular response at 70%. Web server integrations followed at 56%, which makes sense – it is far easier to set up PHP with a web server than any other language. This is a key piece to keeping developers in the PHP ecosystem, particularly around web and APIs.

Q: What PHP Features Make PHP a Valuable Candidate for New Projects and/or Refactoring Existing Projects? (Select All That Apply.)



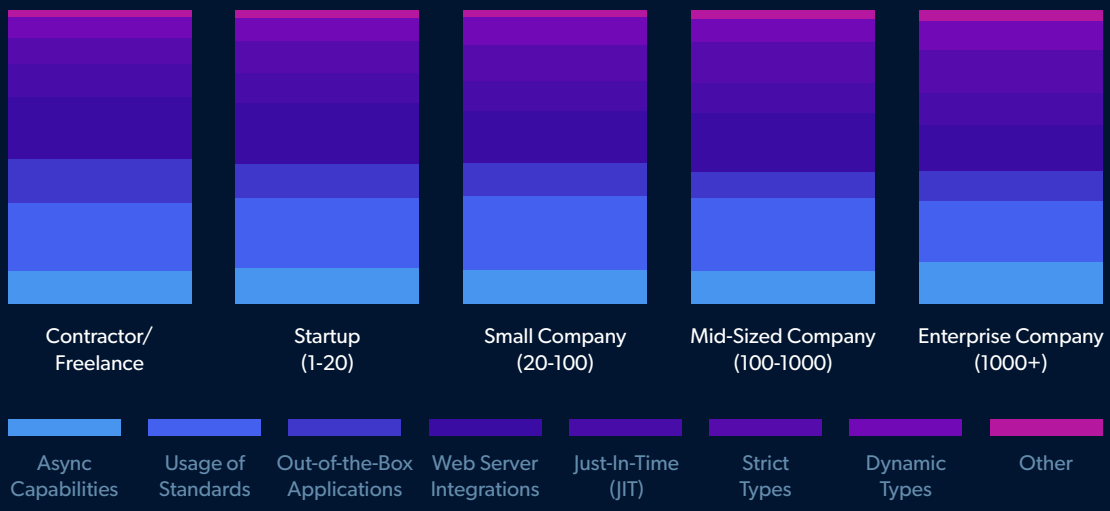
Strict types and async capabilities came in tied for third at 35% each. Async is not quite mainstream, but clearly many are thinking of it as a valuable feature that PHP is offering.

The fact that strict types had such a high response is interesting because PHP has historically been a dynamically-typed language; the ability to use strict types was only added in PHP 7. As developers are building applications with longer lifespans, being able to opt-in to strict types provides more capabilities for static analysis, and ultimately more confidence that your code is doing the "right thing," allowing you to build applications that are both easier to maintain and with greater security.

There is an interesting shift in the language, from being primarily dynamically typed to where strict types are helping people build better applications.

Additionally, we noted two other trends when breaking down responses by organization size: the number of organizations using out-of-the-box applications was inversely proportional to organization size (i.e., smaller organizations reported using out-of-the-box applications at a higher rate than larger organizations), and the adoption of async capabilities was directly proportional to organization size (i.e., larger organizations are adopting async at a higher rate). This suggests that custom development, microservices, and scaling become more prevalent as organization size increases.

Q: What PHP Features Make PHP a Valuable Candidate for New Projects and/or Refactoring Existing Projects? (Select All That Apply.)

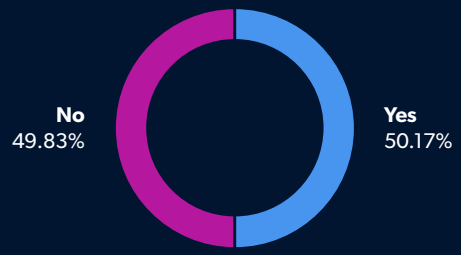


COMPLIANCE REQUIREMENTS

In this question, we asked respondents whether or not their PHP applications have regulatory or industry compliance requirements.

With the percentages rounded to the nearest whole number, our survey was a perfect split with half of respondents reporting no compliance requirements, and half reporting compliance requirements. This is in agreement with last year's survey results.

Q: Do Your PHP Applications Have Regulatory or Industry Compliance Requirements (e.g., SOX, PCI, HIPPA, GDPR)?

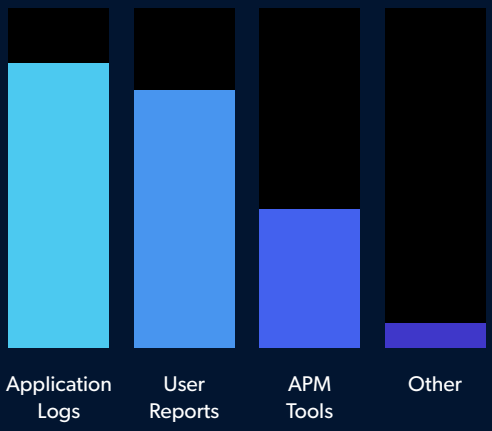


SOLVING ISSUES IN PRODUCTION

Our next question asked respondents to weigh in with how they solve issues in their production applications. Respondents were able to select multiple answer choices.

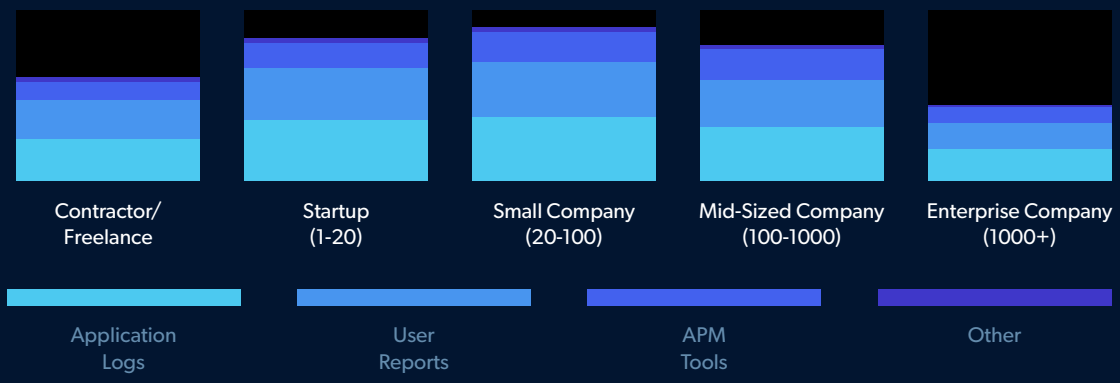
When looking at the data as a whole, the majority of respondents reported using application logs (85%) and user reports (76%) as their keys to finding production issues. APM tools round out the list at 41%. When segmenting the data by company size, application logs and APM tool usage is more prevalent as organization size increases.

Q: How Do You Discover Issues in Your Production Application? (Select All That Apply.)



User reports are the worst channel to discover issues in your production application, as it indicates that it is affecting the customer *now*. Having monitoring tools like those in Zend Server can prove to be invaluable, as they send monitoring alerts exactly when issues arise. One question we have is if monitoring tools may be priced out of reach of smaller organizations, or if the ecosystem needs more education on the benefits.

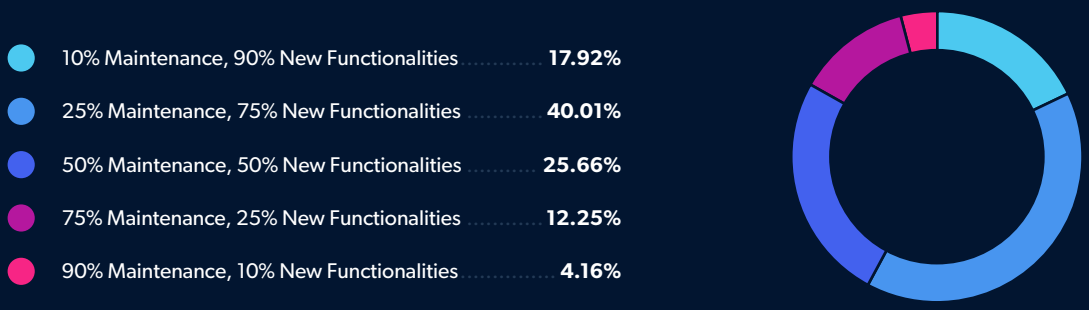
Q: How Do You Discover Issues in Your Production Application? (Select All That Apply.)



MAINTENANCE VS. FEATURE DEVELOPMENT

Our next question looked at how much time developers are spending on application maintenance and production bug fixes compared to how much time they spend on developing new functionalities in their application.

Q: How Much of Your Team's Time Is Spent on Maintenance and Production Bug/Issue Resolution vs. Developing New Functionalities? (Select Closest Matching Answer)

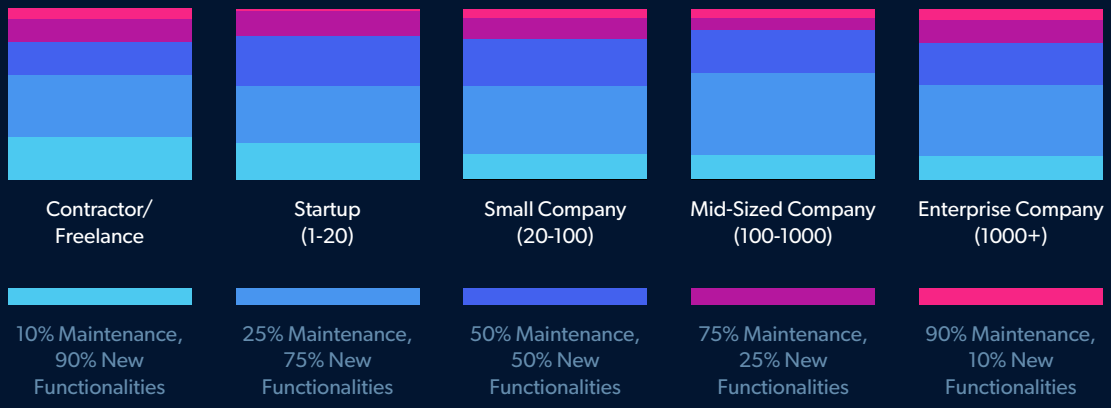


Our survey found that most respondents (40%) dedicate 25% of their time on maintenance and bug fixes, while the remaining 75% of their time is spent developing new functionalities.

Altogether, 83% of respondents reported spending at least 50% of their time on developing new functionalities, unchanged from last year's survey. Larger companies tend to spend more time on maintenance than new features, but only by a slim margin.

When considering organization size, there is one outlier: contractors and freelancers report more time spent on new functionality than maintenance across the board. This suggests that these service providers are being contracted primarily for implementing new functionality, versus maintenance and upgrades.

Q: How Much of Your Team's Time Is Spent on Maintenance and Production Bug/Issue Resolution vs. Developing New Functionalities? (Select Closest Matching Answer)



With a majority of respondents indicating they are building business critical applications, and yet also reporting that application logs and user reports are their primary mechanisms for issue discovery, the ability to balance building new features and performing maintenance tasks is critical. We recommend investment in CI/CD pipelines, and offer services to help you implement them.

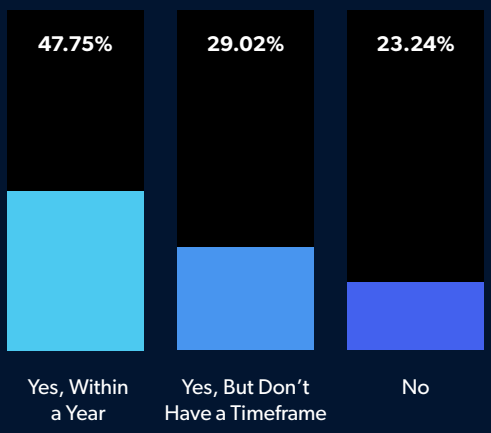
CONTAINERIZATION

Next, we asked respondents to share their status with regards to containerization technology adoption.

A combined 77% said they were planning on using containerization technologies, which is up from 65% last year. Of that 77%, 47% reported plans to use containerization technologies within the year, up from 41% last year.

Only 23% reported no current plans to use containerization technologies, down from 35% last year.

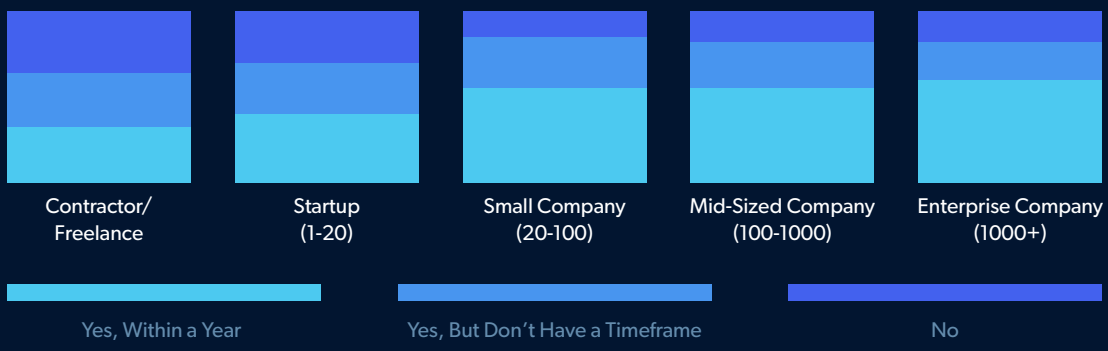
Q: Are You Currently Using, Or Planning to Use, Any Containerization Technologies?



When breaking this down by company size, we observe an interesting trend: within the more general software application industry, enterprises and mid-sized businesses are generally adopting containerization and orchestration practices at a slower rate.

Within the PHP ecosystem, however, these trends are reversed. Considering that web-based applications typically have many integrations and a need for autoscaling, our data demonstrates that organizations with larger web applications are adopting these trends faster out of necessity. Additionally, our sister company, OpenLogic, notes in their 2022 State of Open Source Report that with developer shortages, containerization and orchestration allows companies to automate processes as a way of extending the impact of their existing DevOps teams.

Q: Are You Currently Using, Or Planning to Use, Any Containerization Technologies?



PHP Version and Upgrade Statistics

For many years, PHP versions were released "when ready," with no formal process.

Leading into the PHP 7 release cycle, the PHP internals developers adopted a policy whereby each minor release (including new major "dot zero" releases) would receive 2 years of active support (bugfixes and security patches) and an additional year of security-only support, for a total lifetime of 3 years per version.

This policy has allowed the language to advance quickly, in large part because no more than three versions are being maintained at any given time. The flip side is that language users can only assume that the PHP version they deploy will be supported for up to three years from the time they first deploy.

While operating systems offer long term support for software they ship with their distributions, most only support a single PHP version per LTS release. Further, when bumping to a new LTS version of the operating system, developers are also faced with a daunting PHP migration.

As such, we like to ask what PHP versions our respondents use to get an idea of how quickly (or slowly!) the ecosystem is adapting to PHP release schedules.

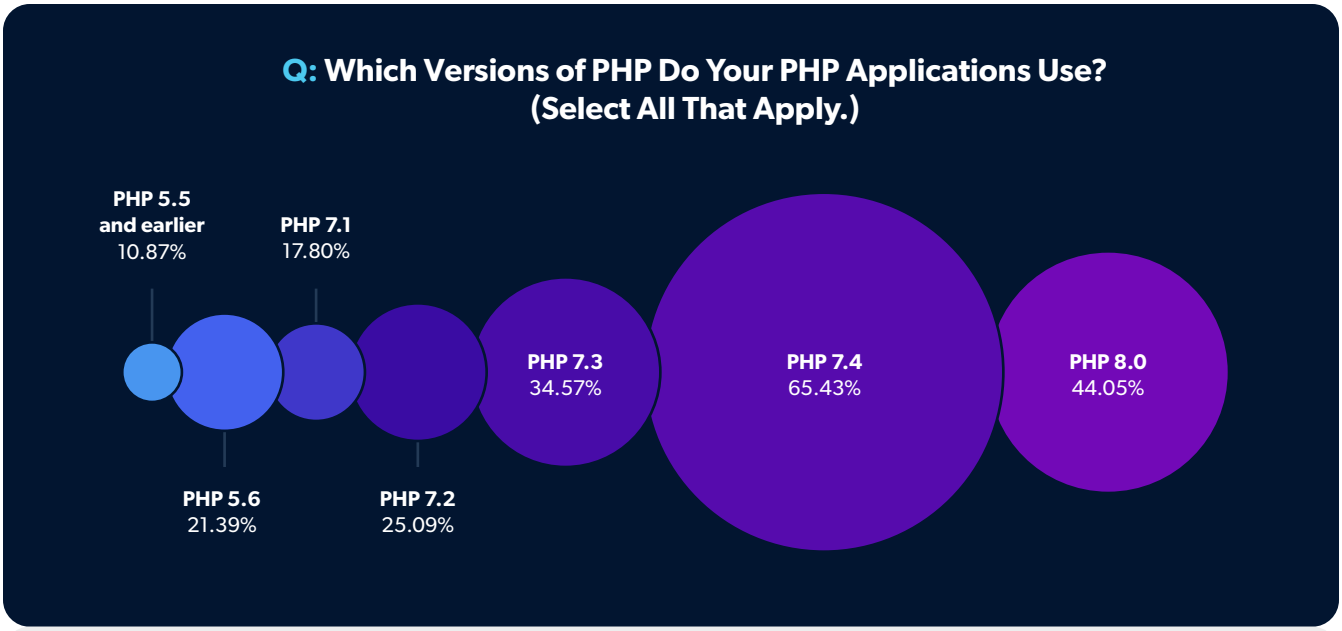
MOST USED PHP VERSIONS

This question asked respondents to share the PHP version they use on their current application. Respondents were able to select multiple answer choices. Summing the responses, it is apparent that many are using multiple versions of PHP.

Not surprisingly, most respondents were using currently supported PHP versions. Nearly two-thirds report using PHP 7.4, which is the last PHP 7 version. It is also the latest version running on many current LTS operating systems.

What is surprising is 44% responded with PHP 8.0, which may be a bit aspirational as it doesn't reflect what we're seeing reported elsewhere in the greater PHP ecosystem.

It is interesting that there are 11% using PHP 5.5 and 21% using PHP 5.6, when there have been two major versions of PHP delivered since then.



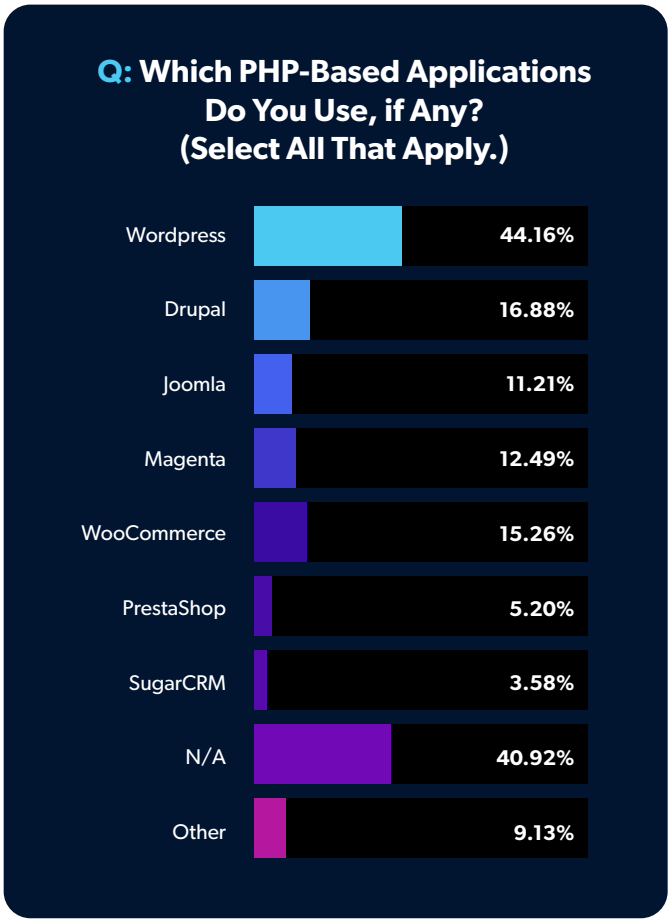
It is not easy to jump from PHP 5 to PHP 8.0. Being able to first move into PHP 7 releases and then gradually into PHP 8.0 will be important. This shift can be facilitated by purchasing commercial support for these versions (like ZendPHP).

MOST USED PHP-BASED APPLICATIONS

The next question asked which PHP-based applications were in use.

As WordPress is the heavyweight of the ecosystem, and a heavyweight of the web in general, we are not surprised to see this receiving the most responses with 44%. Drupal followed with 17%, an increase over prior years and an indication it is gaining more traction with larger organizations.

WooCommerce came in third with 15%. This makes sense as it is part of the WordPress ecosystem as a plug-in for e-commerce. We note that its adoption trends are inversely proportional to organization size. As a free option, this can be a way for organizations to reduce operational costs.

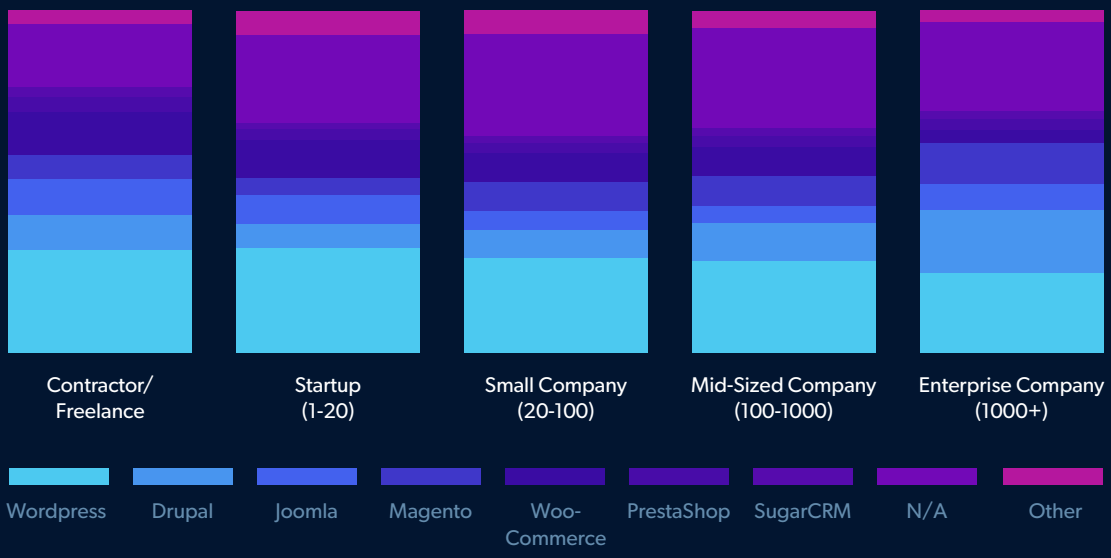


More than 40% of respondents reported that they are not using any of these PHP-based applications. This indicates that a huge part of the PHP ecosystem is creating bespoke applications and APIs.

This is wonderful to see. It's not just about having the ecosystem applications out there – it's also about having a general purpose language that developers feel confident with and can use to build business value.

It is interesting to see Magento only receive 12% of responses. When breaking down responses by organization size, however, we note that Magento usage is higher in larger organizations. Given Magento's acquisition by Adobe in 2018, this may reflect a change in market recognized by Adobe which it is now targeting more specifically.

Q: Which PHP-Based Applications Do You Use, If Any? (Select All That Apply.)



PHP VERSION UPGRADE PLANNING

Our next question looked at whether or not teams were planning a PHP version upgrade within the next year.

Our survey found that 80% of respondents were planning a PHP version upgrade within the next year, an increase from last year's 73%. The remaining 20% are abstaining from an upgrade.

Q: Are You Planning a PHP Version Upgrade Within the Next Year?

With the 3-year lifespan of PHP versions, organizations increasingly find themselves in situations where they need to upgrade their PHP version. This can be to take advantage of new features, or to ensure they continue receiving security patches. Practically speaking, performing an upgrade requires organizational buy-in and a significant investment in planning, testing, and refactoring, and generally involves not only developers but QA teams, infrastructure teams, and even financial departments.

PHP UPGRADE HURDLES

The next question looked at the biggest time commitments to PHP upgrades by asking respondents to select the most-time consuming component of their last upgrade.

Q: What Was the Most Time-Consuming Component of Your Last PHP Upgrade?

Testing	37.23%
Refactoring	35.95%
Compliance Renewals	4.97%
Planning	10.52%
Infrastructure Provisioning	11.33%

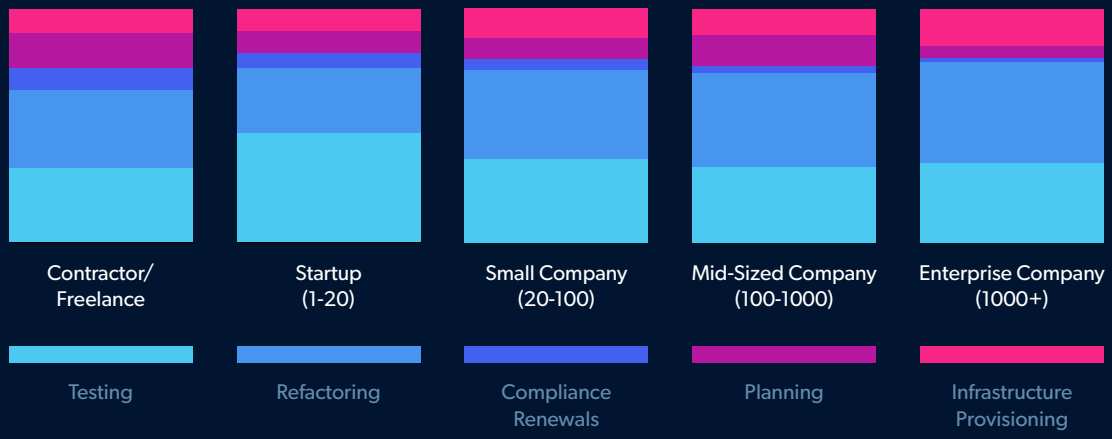
Similar to last year, testing and refactoring are relatively equal at the top with 37% and 36%, respectively.

Infrastructure provisioning and planning both followed with 11% of the responses.

Some might wonder why testing and refactoring are so high. With the PHP language advancing at the rate it does, new deprecations are added with each release, and new features introduced that can save processing cycles, simplify and strengthen code, and make future maintenance easier.

The former leads to testing cycles to ensure that code is updated to remove deprecated functionality, and the latter leads to refactoring of code to adopt the new features. These often take more time than initially estimated. When accounting for organizational size, we note that refactoring and infrastructure provisioning costs impact larger organizations at a higher rate.

Q: What Was the Most Time-Consuming Component of Your Last PHP Upgrade?



Development Technologies

For this portion of the survey, we asked respondents to share their most-used PHP development technologies.

Specifically, we asked respondents to share their currently adopted framework, web server, and methodologies.

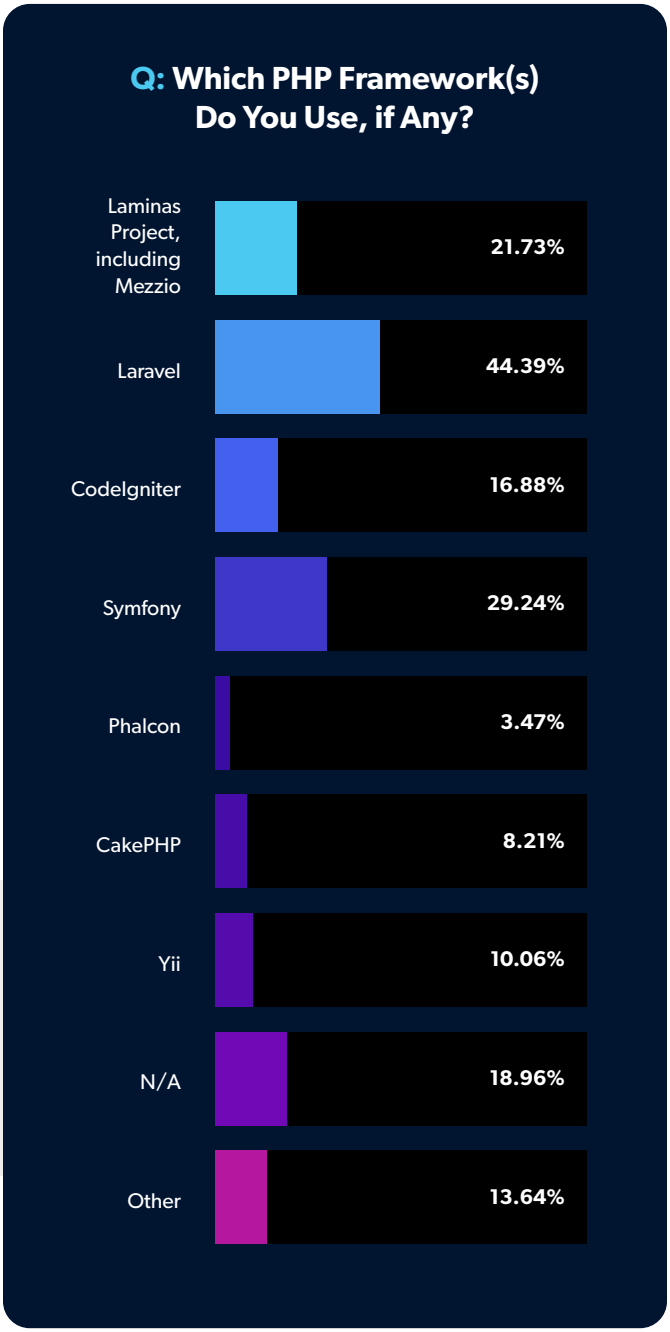
MOST POPULAR PHP FRAMEWORK

We first asked respondents to share which PHP framework they currently use. Answer choices included Zend Framework/Laminas, Laravel, CodeIgniter, Symfony, Phalcon, CakePHP, Yii, other, or not applicable.

Our survey found that most were either using Laravel (44%), Symfony (30%), or Zend Framework/Laminas (22%).

CodeIgniter was 4th-most used with 17%. This is a surprising result as it is considered more of a legacy framework. However, it has a strong community that has been slowly resurrecting it and updating it for recent PHP versions.

A caveat to these numbers: survey popularity does not reflect actual download rates. Regardless, the results are interesting in terms of seeing what is capturing developer attention.



MOST POPULAR PHP WEB SERVER

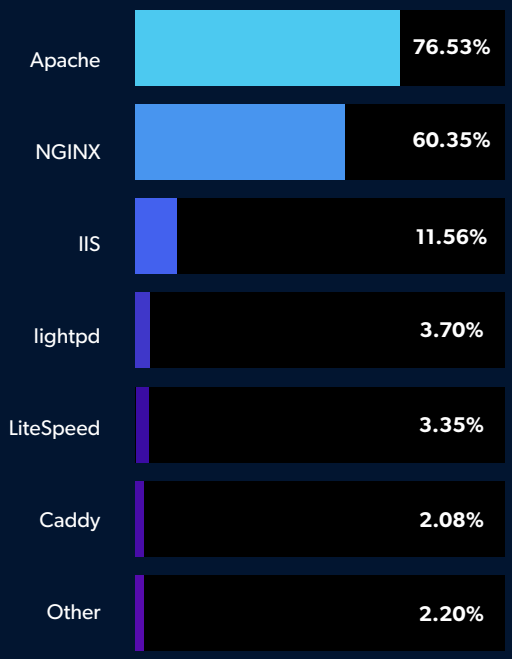
Our next question asked respondents to share the web server they use for their application. Respondents were able to select multiple choices, if applicable.

Last year, Apache received 64% of responses, which has risen to 77% this year.

What is surprising this year is the significant jump for NGINX. With the caveat that last year's survey only allowed for selecting one answer choice, usage of NGINX rose from 28% last year to 60% this year. This massive jump in NGINX is particularly driven by containerized usage of PHP. NGINX provides the easiest way to use PHP in containers.

With FastCGI quickly becoming the preferred way to integrate language runtimes with web servers, particularly due to its ability to run on separate servers or in containers, the web server ecosystem is starting to shift away from Apache as the default option. PHP has a strong offering with its FastCGI Process Manager (php-fpm).

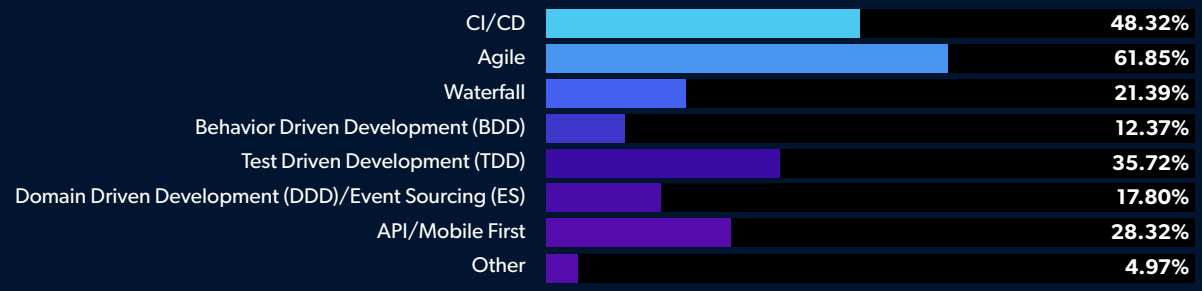
Q: Which Web Server(s) Do You Use For Your PHP Applications? (Select All That Apply.)



MOST POPULAR DEVELOPMENT METHODOLOGIES

The next question asks what development methodologies or practices are in use by PHP development teams. Respondents were able to select multiple answer choices.

Q: What Development Methodologies/Practices Does Your PHP Development Team Use? (Select All That Apply.)



Agile led the way with 62% of respondents, followed by CI/CD with 48%. Third-most popular was test driven development (TDD) with 36%.

CI/CD practices are a foundation for Agile, BDD, and TDD practices. Orchestration and automation of PHP testing and deployment will contribute to organizational success.

Orchestration Technologies

Trends and forecasts from industry analysts such as Gartner and Forrester have indicated that orchestration is trending upward at a dramatic rate the last few years, and will continue to do so for the foreseeable future.

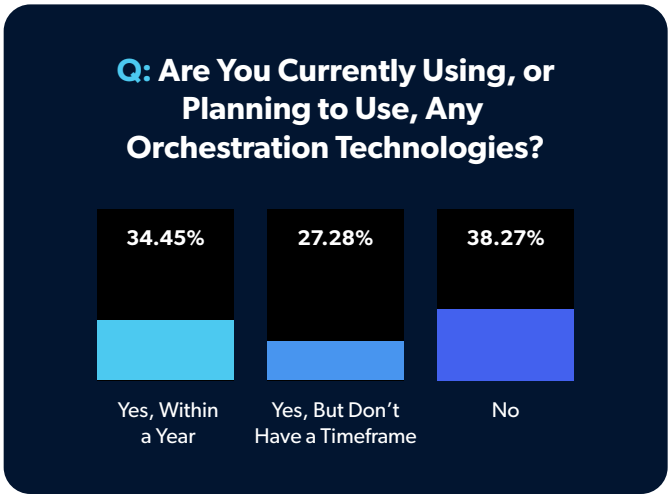
We have been curious to see if this is true within the PHP ecosystem as well, as the technologies could have a huge impact on repeatability and scalability of PHP application deployments.

USAGE OF ORCHESTRATION TECHNOLOGIES

Our first question asked if respondents were using, or planning to use, orchestration technologies.

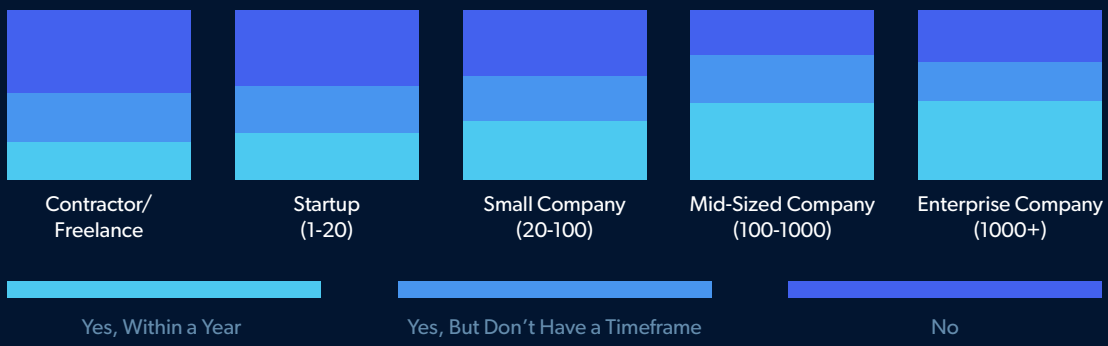
Overall, we are seeing close to two-thirds using or planning to use orchestration technologies, with 34% responding yes (within a year) and 27% responding yes (with no timeframe).

38% indicate they are not using nor plan to use such technologies.



When breaking this down by company size, we see that as the company size trends upward, so do the number reporting current or future adoption of orchestration technologies and practices. We can likely extrapolate that these trends are due to size and complexity of applications.

Q: Are You Currently Using, or Planning to Use, Any Orchestration Technologies?



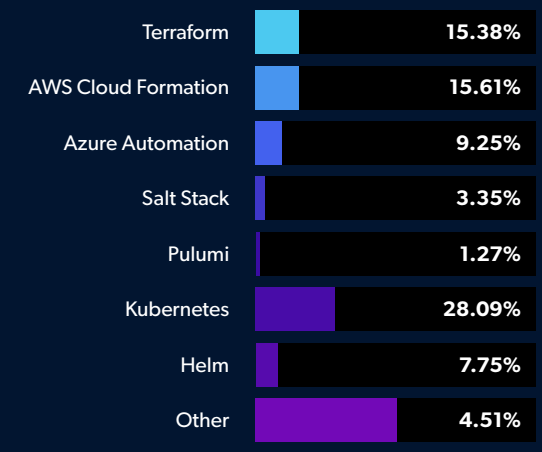
Tooling and training that allow your developers and operations teams to incorporate PHP into containers and/or orchestration technologies and practices will help make your organization more successful and ship with fewer issues and headaches. Having a repeatable build and deployment process helps avoid costly production issues, and can ease issues with operating system and language updates as they become configuration settings instead of complex provisioning on dedicated hardware. They can also help reduce the number of engineering staff needed, as the infrastructure moves from dedicated hardware or virtual machines to infrastructure-as-code.

MOST POPULAR ORCHESTRATION TECHNOLOGIES

In the next question, we asked respondents to share if they are using orchestration technology. We have removed the responses that indicated "not applicable," giving us 737 respondents.

Kubernetes led the way with 33% of respondents, which makes sense as we look at containers – this is the tool to use.

Q: Which of the Following, if Any, Orchestration Technologies Do You Use? (Select All That Apply.)



It is interesting that Terraform and AWS Cloud Formation are tied for second at 18%, as Cloud Formation is essentially AWS's answer to the open source Terraform, but made specific for the AWS cloud.

Significant numbers of write-in answers indicated Ansible and Swarm as popular orchestration technologies. With many years as a stable deployment technology, Ansible is a solid choice. Swarm provides a simpler stack than Kubernetes, which may be a selling point for less complex orchestration needs.

Async Programming

Many developers and organizations cite async capabilities as a reason to move to technologies such as Node.js. The promise of async is the ability to parallelize processing, defer processing and, in general, provide for faster processing of incoming requests.

Interestingly, PHP has provided async capabilities via userland libraries and extensions for many years now, and many benchmarks demonstrate that even baseline PHP is far faster than Node.js. When adding async to the mix, it can provide performance to rival even compiled languages.

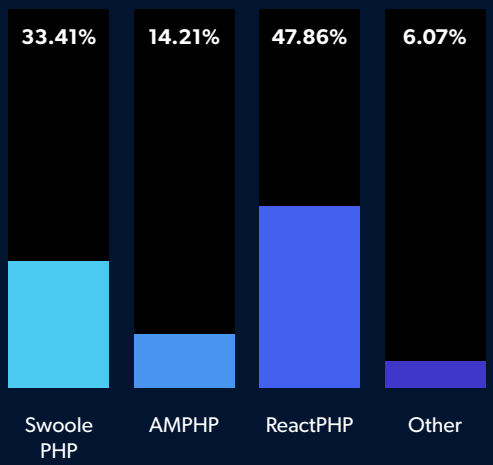
We are curious if our respondents are aware of PHP's async capabilities, what they are using, and what sorts of applications they are finding benefit from async processing.

MOST POPULAR ASYNC PHP FRAMEWORK

Our first question asked respondents to share what async framework, if any, they are using.

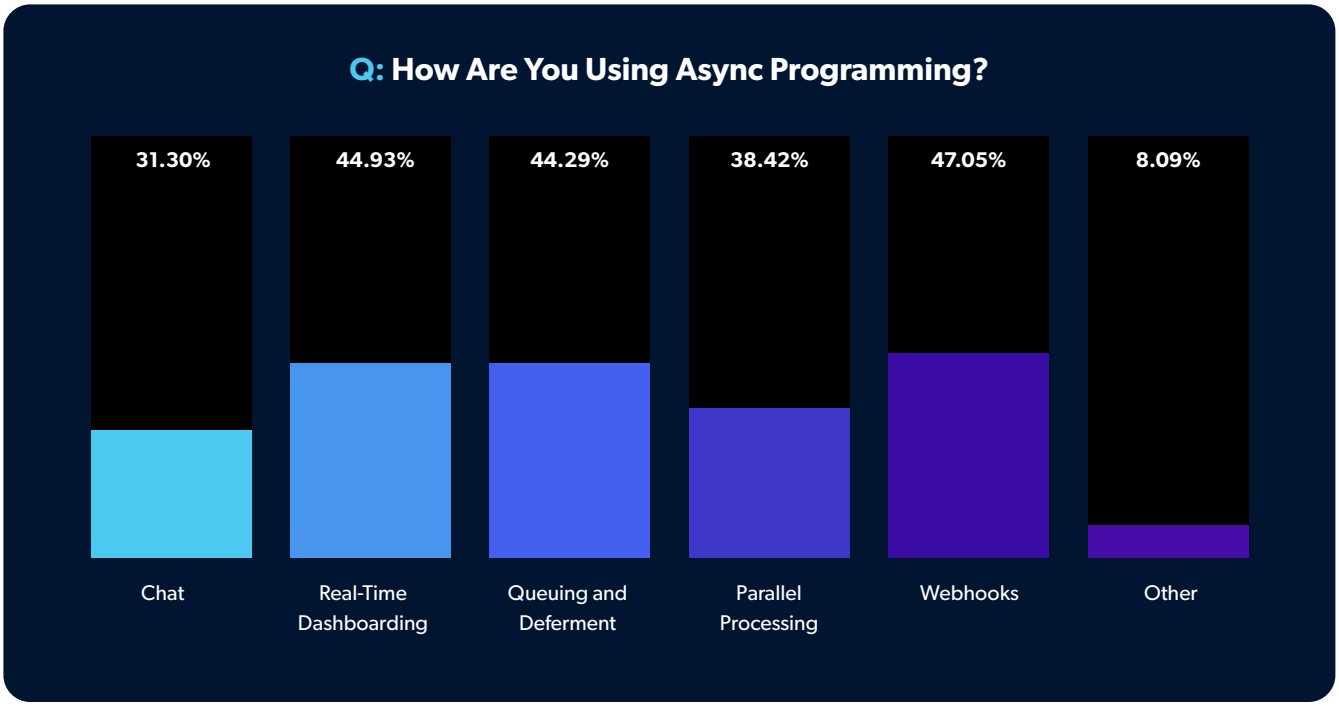
As an async framework is required in order to do async programming in PHP, we will only take into account those who chose an async framework, leaving 331 respondents. ReactPHP is the most popular, with Swoole close behind, and AMPHP on par with where we've seen it in previous years. Swoole's usage has increased year over year, eating into ReactPHP's audience. We are curious if the addition of Fibers in PHP 8.1 will impact async framework adoption.

Q: What Async Framework Are You Using, if Any? (Select All That Apply.)



HOW DEVELOPERS ARE USING ASYNC PROGRAMMING

Our next question asked respondents to share how they are using asynchronous programming in their current application. Respondents were able to select more than one answer choice.



Similar to the previous question, we have removed the respondents who chose "not applicable" to this question. This gives us 467 respondents. Respondents were able to select multiple answer choices. Our survey found respondents using asynchronous programming in multiple ways, with substantial responses for each available choice.

There was a relatively even split among webhooks (47%), real-time dashboarding (44%), queuing and deferment (44%), and parallel processing (38%). These are the biggest use cases for async programming, and where its usage has most impact in web applications and APIs.

PHP Outlook for 2022

Since the release of PHP 7 and the adoption of a predictable release cycle, the language has made huge strides.

Besides notable performance gains, the language has adopted a number of features of more "enterprise" languages — strict typing, comprehensive type hinting, and more — that provide developers the ability to create more stable and maintainable applications. These features future-proof applications, ensuring PHP will continue to have longevity.

When it comes to containerization and orchestration, PHP developers are adopting the technologies faster than others in the software industry. This pioneering aspect will demonstrate the utility of these technologies and, again, secure PHP's place in not only the web, but the greater software industry as a whole.

The web, however, is a crucial strength of PHP: web-based APIs are a key component of business-to-business integrations, and critical to most modern enterprises. The ability to scale these via containerization and orchestration will be a key factor in PHP's success going forward.

PHP is thriving and evolving, and will be building tomorrow's businesses.

QUESTIONS?

Do you have questions about the report? Want to talk to an expert about your big plans with PHP in 2022? Our experts are standing by to answer your burning questions.

TALK TO AN EXPERT

zend.com/contact-us

STREAMLINE WEB APPLICATION DEPLOYMENT

With Zend Server, your team can streamline application deployment, achieve scale, improve security, and increase performance.

TRY ZEND SERVER FREE

zend.com/products/zend-server

KEEP YOUR EOL PHP APPLICATIONS SECURE AND SUPPORTED

Zend can help keep your applications running on end-of-life PHP versions secure with backported bug fixes, patches, and around-the-clock support.

SUPPORT OPTIONS

zend.com/products/zendphp-enterprise

DEPLOYMENT SERVICES YOU CAN COUNT ON

From migrations, to audits, to implementing CI/CD, Zend Professional Services can help you realize your organization's goals.

ZEND SERVICES

zend.com/services

zend by Perforce