DZone

FEBRUARY 2024

The Modern DevOps Lifecycle

Shifting CI/CD and Application Architectures

TREND REPORT

BROUGHT TO YOU IN PARTNERSHIP WITH

Red Hat

ORIGINAL RESEARCH

Key Research Findings

An Analysis of Results From DZone's 2024 DevOps Lifecycle Survey

G. Ryan Spain, Freelance Software Engineer, former Engineer & Editor at DZone

It's no secret that DevOps is here to stay. As the years go by, we must continuously assess and seek improvements to our existing software processes, systems, and culture. DevOps is no exception to that rule. Since its inception, DevOps has served as a beacon for the tech community — a foundation on which we can guide and base our entire application infrastructure and team culture. It's the goal and the shining example of what we hope to be — because DevOps is not only a process or methodology, but is also a state of being. Thus, as business needs and customer demands shift, so must our technology, mindsets, and architecture. Now is the time for this movement that's all about "shifting left" to essentially *shift*.

For our annual DevOps research, we explored the fundamental principles of DevOps as well as the emerging topics, methodologies, and challenges surrounding the engineering ecosystem. We focused on core topics such as the state of continuous integration and continuous delivery (CI/CD) pipelines, technical debt elimination, cost optimization, supply chain security, and application infrastructure and provisioning. At the end of the day, our goal is to provide our community with a time capsule of the industry as it stands today, and where DevOps is going in 2024 and beyond.

Major research targets were:

1. Continuous delivery adoption, benefits, and barriers
2. Automation in the SDLC
3. Software delivery practices, techniques, and metrics

Methods: We created a survey and distributed it to a global audience of software professionals. Question formats included multiple choice, free response, and ranking. The survey was disseminated via email to DZone and TechnologyAdvice opt-in subscriber lists as well as promoted on DZone.com, in the DZone Core Slack workspace, and across various DZone social media channels. The survey was opened December 12, 2023 and closed on January 2, 2024; it recorded 207 complete and partial responses.

Demographics: We've noted certain key audience details below in order to establish a more solid impression of the sample from which results have been derived:

- 21% of respondents each described their primary role in their organization as "Technical Architect," 18% described "Developer Team Lead," and 16% described "Developer/Engineer." No other role was selected by more than 10% of respondents.
- 79% of respondents said they are currently developing "Web applications/Services (SaaS)," 43% said "Enterprise business applications," and 26% said "Native mobile apps."
- "Java" (69%) was the most popular language ecosystem used at respondents' companies, followed by "JavaScript (client-side)" (55%), "Python" (54%), and "Node.js (server-side JavaScript)" (44%).
- Regarding responses on the primary language respondents use at work, the most popular was "Java" (45%), followed by "Python" (18%) and "C#" (7%). No other language was selected by more than 5% of respondents.
- On average, respondents said they have 18.79 years of experience as an IT professional, with a median of 18 years.
- 35% of respondents work at organizations with < 100 employees; 26% work at organizations with 100-999 employees; and 37% work at organizations with 1,000+ employees.*

**Note: For brevity, throughout the rest of these findings we will define "small" organizations as having < 100 employees, "mid-sized" organizations as having 100-999 employees, and "large" organizations as having 1,000+ employees.*

In this report, we review some of our key research findings. Many secondary findings of interest aren't included here.

Research Target One: Continuous Delivery Adoption, Benefits, and Barriers

Motivations:

1. Continuous delivery (CD) has a multitude of potential advantages, including increasing the speed of software delivery by automating the entire release pipeline, allowing for more frequent and reliable releases. It can improve the overall quality of software through automated testing, reducing the likelihood of bugs and errors reaching production. It can also promote collaboration among development, operations, and testing teams, fostering a culture of shared responsibility and efficient communication. The list goes on. We wanted to know what aspects of CD are most appealing to developers and other tech professionals in the software industry.
2. On the other side of the coin, barriers to CD adoption can prevent organizations from taking advantage of the benefits that continuous delivery can provide. Internal resistance to change, difficulties integrating legacy systems and existing infrastructure, and insufficient resources can all be obstacles to CD adoption. We aimed to find out what barriers most frequently bar organizations from implementing continuous delivery.

Reasons for CD Adoption

Continuous delivery offers numerous advantages to organizations seeking to streamline their software development and deployment processes. As we mentioned in our "Motivations" above, CD adoption can accelerate the software delivery lifecycle; enhance software reliability, quality, and adaptability; and save software creators time and money. To determine what advantages of CD adoption software professionals found most lucrative, we asked:

> *What do you believe are the top reasons for adopting continuous delivery? Rank in order of most important (top) to least important (bottom).*

Results (*n*=163):

Table 1. Top reasons for adopting continuous delivery

Reason	Overall Rank	Score	*n*=
Increased speed of feature delivery	1	1,177	131
Shortened development cycles	2	1,045	121
Improved developer/team flow/productivity	3	1,044	126
Reduced complexity of development cycles	4	963	118
Increased release frequency	5	960	118
Reduced deployment error rate	6	910	126
Reduced overhead costs	7	807	115
Reduced time to complete QA feedback loops	8	649	101
Reduced maintenance costs	9	641	104
Reduced number of bugs post deployment	10	611	102
Reduced mean time to discovery	11	574	107
Reduced error budget	12	414	97

Observations:

1. For the third year in a row, "Increased speed of feature delivery" and "Shortened development cycles" were ranked numbers one and two, respectively, in the rankings of top reasons for CD adoption. "Improved developer/team flow/productivity," which ranked fourth last year, moved to third place this year (switching spots with last year's third place, "Reduced complexity of development cycles"), and it was only one point away from second place in terms of weighted score.
2. Respondents who said their organization releases to production "Once per week" or more ranked "Increased release frequency" as a much more valuable reason for CD adoption (second place) than respondents at organizations with longer spans between production releases (sixth place), while ranking "Improved developer/

team flow/productivity" (fourth place) as less important than those at organizations with a slower release frequency (fourth place).

3. Respondents who believed their organization releases new software features either "Too quickly" or "Too slowly" both ranked "Increased speed of feature delivery" first in their list of top reasons for adopting CD. However, respondents who believed their organization releases new features "At the optimal rate" ranked "Increased speed of feature delivery" third overall, with "Shortened development cycles" and "Reduced complexity of development cycles" ranking first and second, respectively.

Barriers to CD Adoption

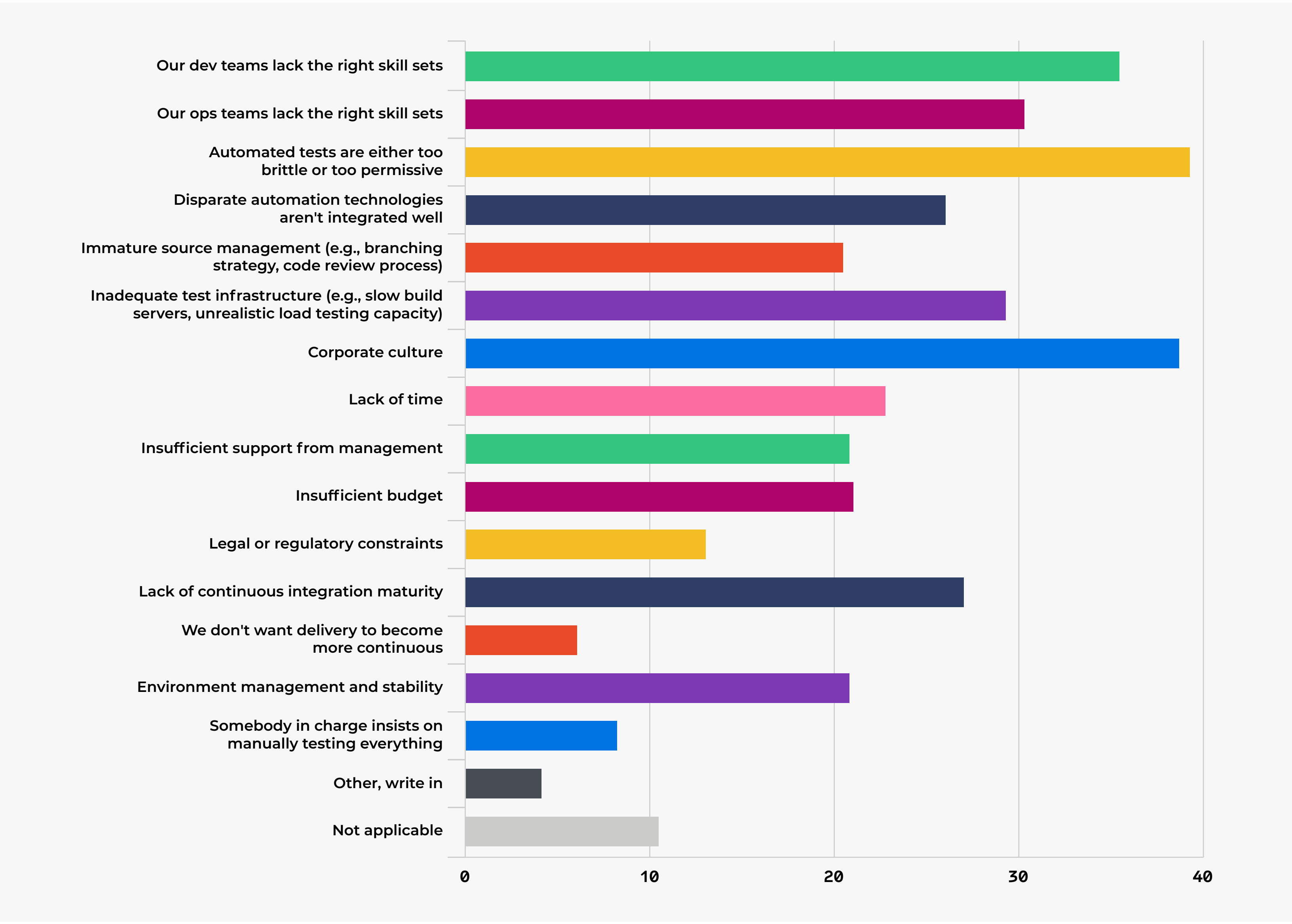
As with any practice or methodology new to an organization, continuous delivery is not without its barriers to entry. Teams accustomed to traditional development models may find it challenging to adapt to the CD paradigm. Transitioning from manual to automated testing and deployment processes can be complex and leech resources that could go toward new features or paying off technical debt. Overcoming these barriers often requires a strategic approach that may include cultural shifts, technology upgrades, additional staff, or comprehensive training programs.

To see what barriers were most commonly observed as coming between organizations and CD adoption, we asked:

What do you believe are your organization's main barriers to adopting continuous delivery?

Results (*n*=176):

Figure 1. Barriers to continuous delivery adoption



Observations:

1. The most common barriers to CD adoption were "Automated tests are either too brittle or too permissive" (39%), "Corporate culture" (38%), and "Our dev teams lack the right skill sets" (35%). The same responses comprised the top three most common barriers in our 2023 DevOps survey, though with "Corporate culture" coming behind "Our dev teams lack the right skill sets."
2. The least common barriers were "Legal or regulatory constraints" (14%), "Somebody in charge insists on manually testing everything" (9%), and "We don't want delivery to become more continuous" (6%). These were the only options (besides "Not applicable" at 11%) not to fall within a 20%-40% response rate. Again, these options comprised the bottom three barriers from last year's survey results.
3. Respondents at large organizations most frequently selected "Corporate culture" (54%) as a barrier to CD adoption much more frequently than small (24%) and mid-sized (35%) organizations. For respondents at small organizations, the most common barriers were "Our dev teams lack the right skill sets" (38%); "Our ops teams lack the right skill sets" (29%); and a tie between "Automated tests are either too brittle or too permissive," "Inadequate test infrastructure," and "Insufficient budget" (28% each).

    Also of note, respondents at small organizations were moderately, but significantly, more likely to select "Our dev teams lack the right skill sets" (38%) and "Insufficient budget" (28%) than those at mid-sized (33% and 17%, respectively) and large (31% and 20%, respectively) organizations — a finding that we return to later when we discuss practices and techniques for optimizing software delivery costs.

Research Target Two: Automation in the SDLC

Motivations:

1. Automating software deployment can bring a host of benefits that significantly enhance the efficiency and reliability of the development lifecycle, potentially reducing the likelihood of human errors, accelerating the deployment process, enhancing scalability, and more. On the other hand, factors such as deploying high-risk changes or sensitive configurations and adhering to privacy regulations or security requirements may inhibit an organization's ability to fully automate deployment.

    With that in mind, we tried to get a sense of how many organizations still have manual steps that are necessary in their deployment pipelines, and where in the pipeline manual intervention is most likely needed.
2. Deployment automation is much less likely to cause concern in pre-production environments, and generally speaking, the pros of automating early in the deployment pipeline tend to substantially outweigh the cons. So we also wanted to get a general idea of how much organizations automate their provision and deployment processes in pre-production environments.

Automation in the Deployment Pipeline

While fully automating the deployment pipeline may sound good in theory — and does, in fact, have the potential to solve or alleviate a multitude of deployment-related issues — there are plenty of situations in which complete automation may not be possible or prudent. Striking a balance between automation and manual intervention can allow organizations to maintain control, address exceptional cases, and ensure a level of caution in situations where the consequences of automated deployment may have significant implications.

To see how many organizations were retaining manual steps in their deployment pipelines, we asked:

> *Do your organization's deployments to production require any manual steps?*

And in order to get a sense of where in the deployment pipeline manual intervention is most prevalent, we asked respondents who answered "Yes" to the former question:

> *Which of the following require manual intervention between dev and production?*

Results (*n*=177 and *n*=127, respectively):

SEE FIGURES 2 AND 3 ON NEXT PAGE

Figure 2. Organizations requiring manual steps for deployment

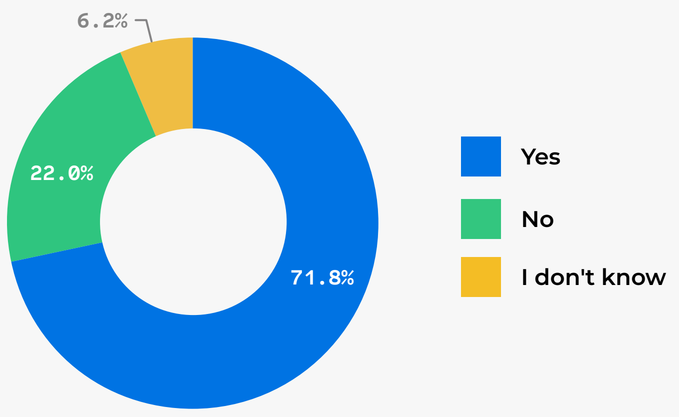
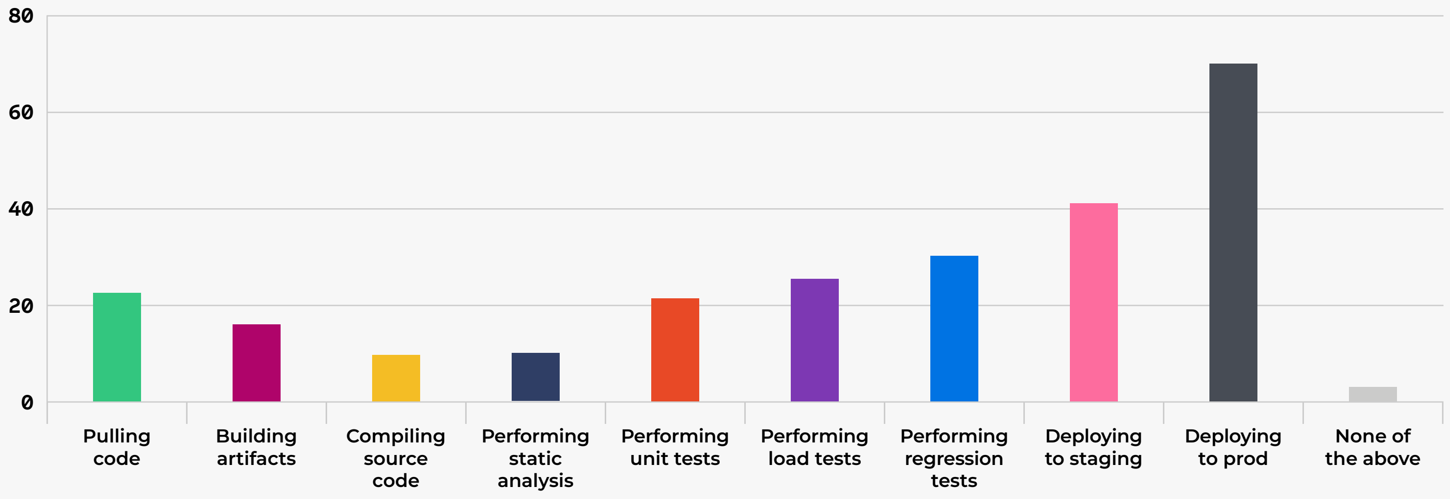


Figure 3. Deployment steps requiring manual intervention



Observations:

1. 72% of respondents said their organization's deployments to production require manual steps vs. 22% who indicated their organization's prod deployments are fully automated. This represents a significantly higher percentage of organizations requiring some sort of manual intervention than our data showed last year, when we found that 61% of respondents' organizations required manual steps in the process of deploying to production, and 38% indicated a fully automated production pipeline.
2. Like last year, "Deploying to production" was the most common step in the deployment pipeline requiring manual intervention by a wide margin (71% in 2024, 75% in 2023). The next most common step requiring manual intervention, "Deploying to staging," was selected by 42% of respondents, 29% lower than "Deploying to production" — last year's results were, again, similar, with second place "Deploying to staging" being selected by 41% of respondents.

    In fact, 17% of respondents who answered this question only selected "Deploying to production" as a required manual step in their organization's deployment pipeline, and another 12% only selected "Deploying to production" and "Deploying to staging."

    These results may be evidence of a plateau in adoption of deployment pipeline automation, a major contributor to which is possibly stakeholders who are wary of late-stage deployments occurring without human confirmation of software quality — an understandable, and in many cases practical, hesitation.

Provisioning and Deployment Automation

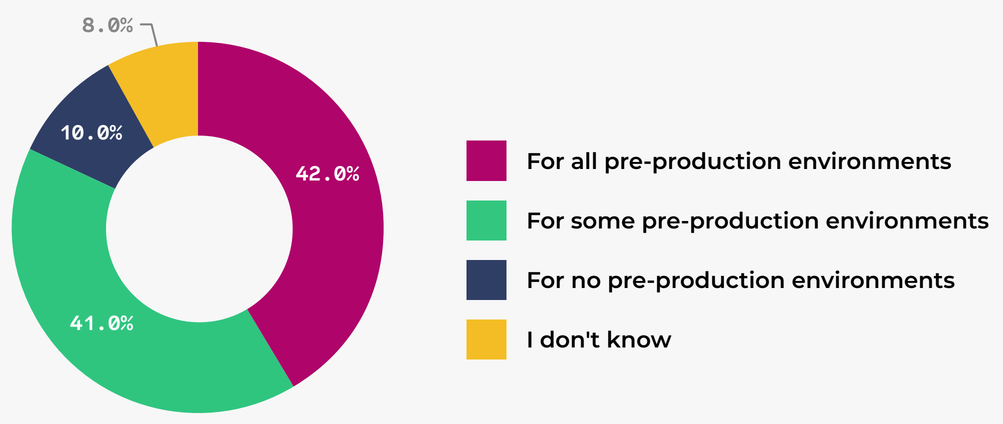
As we noticed in the last section's observations, organizations are much more likely to automate earlier steps in the deployment pipeline than later steps. Pre-production deployments carry less potential risk in general, and therefore tend to be less concerning where automation is involved. At the same time, with advances in cloud infrastructure and related tooling, automated provisioning has become more attainable and commonplace than ever.

We wanted to know roughly how often organizations take advantage of these ideas and automate provisioning and deployment tasks early on in their deployment pipeline, so we asked:

> *Your organization has automated provisioning and deployment: {For all pre-production environments, For some pre-production environments, For no pre-production environments}*

Results (*n*=168):

Figure 4. Automated provisioning and deployment pre-prod



We also examined the response segments from this question to determine a correlation between automated pre-production provisioning and deployment and fully automated deployment processes.

Results:

Table 2. Automated provisioning and deployment pre-prod vs. manual steps required for prod deployment*

	Not Fully Automated	Fully Automated
For all pre-production environments	63%	37%
For some pre-production environments	83%	17%
For no pre-production environments	100%	0%

**Note: % of rows (table)*

Observations:

1. 42% of respondents said that their organization has automated provisioning and deployment "For all pre-production environments," trending downward year over year from 50% in 2023 and 55% in 2022. There was no significant difference in response rates of those claiming automated provisioning and deployment "For some pre-production environments" (41% in 2024, 41% in 2023, and 37% in 2022) or "For no pre-production deployments" (10% in 2024, 9% in 2023, and 9% in 2022).

   It should be noted that this year, we added an answer choice — "I don't know" — to this question, which was selected by 8% of respondents. While it is unlikely that this change accounted for the entire 8% discrepancy between response rates of those claiming automated provisioning and deployment "For all pre-production environments" this year and last year, it is still a likely factor. Future research will help determine the impact of this change and verify the existence or extent of a downward trend.

2. Unsurprisingly, respondents at organizations that automate provisioning and deployment for all pre-prod environments were considerably more likely to report that their organization requires no manual intervention in deploying to production (37%) than those at organizations automating provisioning and deployment for some pre-prod environments (17%) or no pre-prod environments (0%).
3. Differences in the rates of organizations automating pre-production provisioning and deployment were relatively minor when we segmented responses by organization size. 44% of respondents at mid-sized and large organizations each said that provisioning and deployment were automated "For all pre-production environments," compared to 36% at small organizations.

   There was not a significant difference in respondents answering, "For some pre-production environments," between small (40%), mid-sized (40%), and large (42%) organizations. And 15% of respondents at small organizations said their company automates provisioning and deployment "For no pre-production environments," compared to 7% at mid-sized organizations and 8% at large organizations.

   So while we observed statistically significant differences between small and larger organizations' rates of automated pre-prod provisioning and deployment, it seems clear that organization size, and the assumed resource increase that goes along with increased size, is *not the primary barrier* to pre-production automation.

Research Target Three: Software Delivery Practices, Techniques, and Metrics

Motivations:

1. Delivering software products is a resource-intensive process, and organizations constantly strive to find ways to optimize delivery costs. And this cost optimization isn't only about the company's bottom line; software quality is often linked with cost management — better managed costs can result in increased resources for additional staffing, better tooling, and even more cost optimization. We aimed to determine what practices organizations use most often to optimize their software delivery costs.
2. Tracking appropriate metrics for software deployment and continuous delivery is crucial for organizations to measure, assess, and optimize their development processes. These metrics provide valuable insights into the efficiency, quality, and performance of their software delivery pipeline — and if worst comes to worst, tracking the appropriate metrics can help quickly rectify, or even circumvent, critical post-release failures.

   But what metrics do organizations track the most, and do software professionals working at those organizations believe that those metrics are the most important to track? We asked respondents about their deployment-related metrics to find out.
3. Platform engineering can offer organizations a wide variety of advantages. It can streamline and automate processes, reducing manual efforts and minimizing errors. It can enhance collaboration between development and operations teams by providing standardized and efficient tools, facilitate scalability, allowing organizations to easily adapt to changing workloads and demands, and contribute to faster time to market for software products through automated deployment and testing.

   Overall, investing in platform engineering can help organizations build a solid technological foundation, fostering agility, efficiency, and innovation in their software development and operations. We wanted to know more about which platform engineering benefits seemed most appealing to software professionals.

Cost Optimization Practices

Managing the costs of software delivery involves streamlining workflows, automating repetitive tasks, and minimizing manual interventions, thereby accelerating the delivery pipeline, reducing the potential for errors and rework, and leading to overall cost savings. Additionally, optimizing software delivery costs supports scalability, enabling organizations to handle varying workloads without proportional increases in expenses. As we mentioned in the "Motivation" summary above, these advantages are generally tied to software quality, at least indirectly.

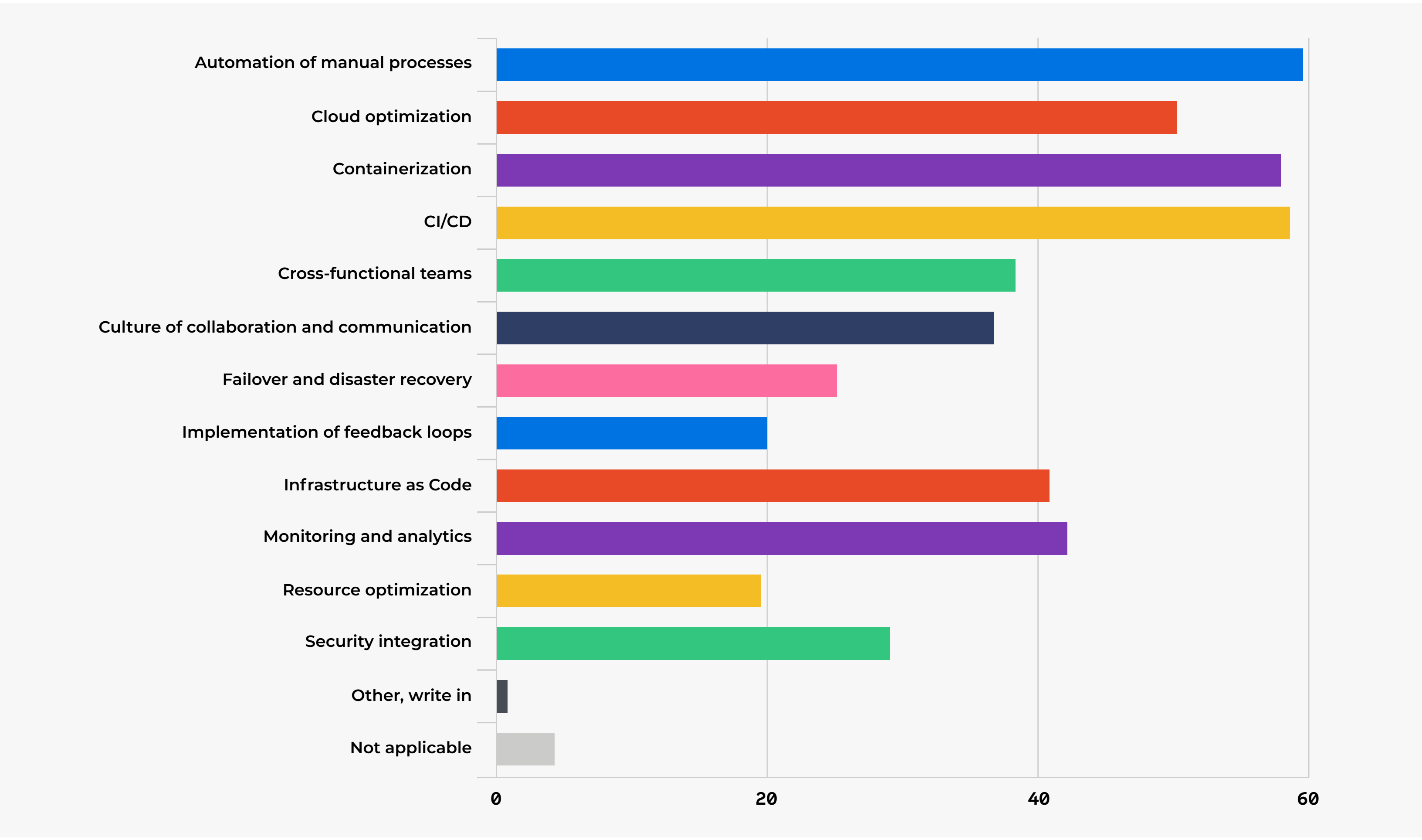
To see what methods organizations use most frequently to keep their software delivery costs down, we asked:

> *Which of the following practices and techniques does your organization use to manage and optimize costs?*

Results (*n*=175):

SEE FIGURE 5 ON NEXT PAGE

Figure 5. Practices and techniques for managing and optimizing costs



We additionally looked at these results segmented by respondents' organization size.

Results:

Table 3. Cost optimization practices by organization size*

Practice/Technique	Org Size		
	1-99	100-999	1,000+
Automation of manual processes	49%	63%	66%
Cloud optimization	39%	59%	54%
Containerization	42%	72%	62%
CI/CD	44%	72%	60%
Cross-functional teams	25%	46%	45%
Culture of collaboration and communication	25%	48%	38%
Failover and disaster recovery	8%	39%	31%
Implementation of feedback loops	22%	22%	17%
Infrastructure as Code	17%	57%	52%
Monitoring and analytics	34%	54%	45%
Resource optimization	14%	26%	20%
Security integration	19%	22%	42%

**Note: % of columns (table)*

Observations:

1. The most common practices for cost management and optimization were "Automation of manual processes" (59%), "Continuous integration and continuous delivery" (58%), and "Containerization" (57%). On average, respondents selected 4.06 out of the 12 selections provided, with a median of 3 options selected.
2. Respondents at mid-sized to large organizations were significantly more likely than those at small organizations to select any given technique for cost optimization, with the exception of "Security integration" — which was significantly less likely to be selected by respondents at small *and* mid-sized organizations than those at large orgs — and "Implementation of feedback loops," the only technique with no significant difference in response rates segmented by organization size, and the second least popular technique selected overall.

   Further, respondents at small organizations selected an average of 3.47 out of the 12 options given, compared to an average of 5.83 selections for those at mid-sized organizations and 5.34 for those at large organizations.

   These results seem to exemplify the adage, "You have to spend money to make money." Most of, if not all, these practices require considerable investments to implement, be those resources time, money, and/or personnel. So while these techniques may optimize costs long term, the cost of getting started may be a barrier to entry for smaller organizations — a hypothesis corroborated by the findings we discussed earlier, in which respondents at small organizations were significantly more likely to select lack of dev team experience and insufficient budget as barriers to entry to CD than those at mid-sized and large organizations.

Continuous Delivery and Deployment Metrics

By monitoring key indicators, organizations can identify areas for improvement, make data-driven decisions, and enhance the overall effectiveness of their development practices. Overall, tracking appropriate metrics is an integral part of fostering a culture of continuous improvement within the realm of software deployment and continuous delivery. To ascertain which metrics software professionals believe to be the most important, as well as which metrics organizations are most likely to track, we asked:

What do you believe are the most important metrics for continuous delivery? and

What deployment-related metrics does your organization track?

Results (*n*=171 and *n*=170, respectively):

Figure 6. Opinion of most important metrics for CD

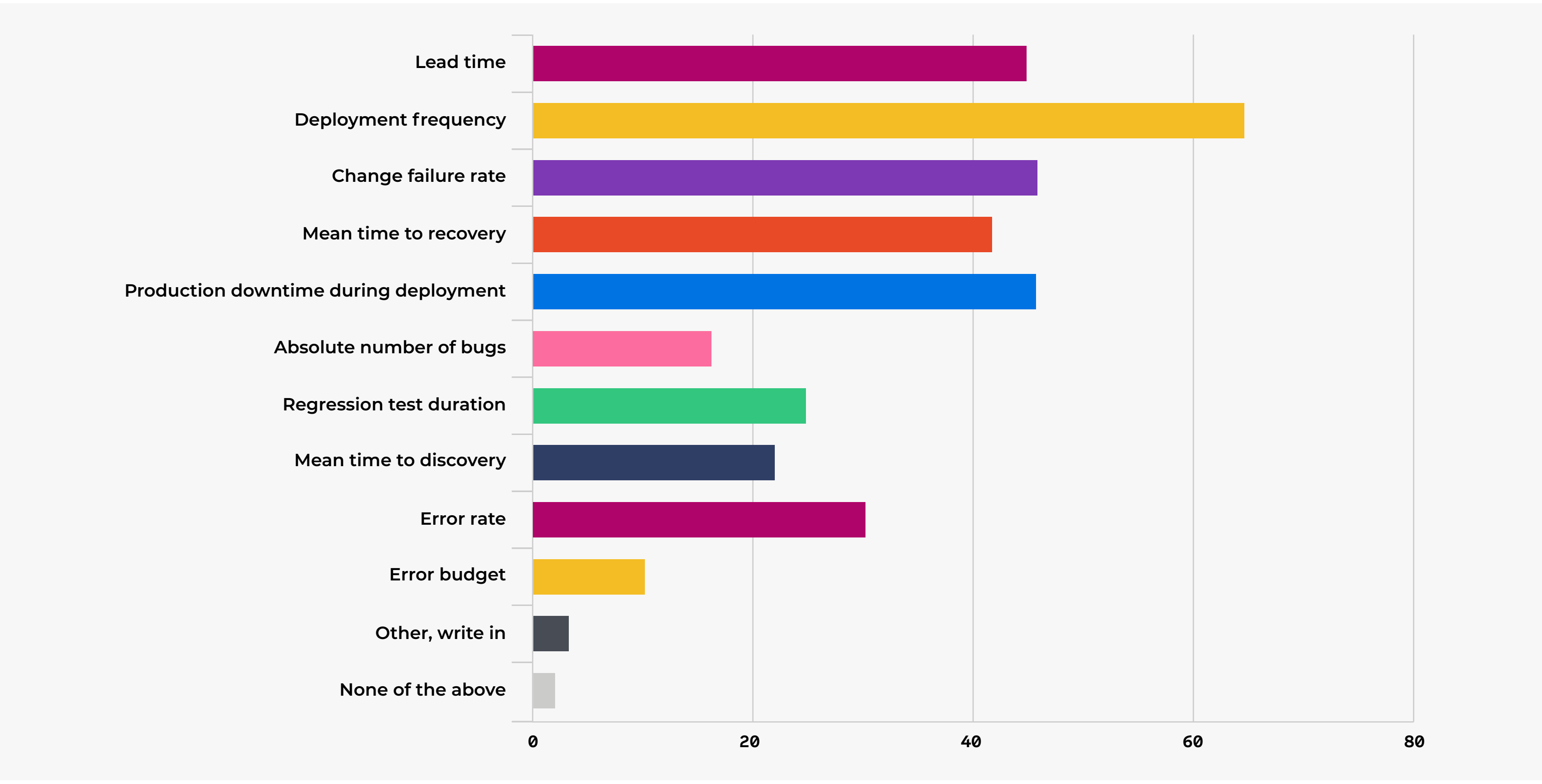
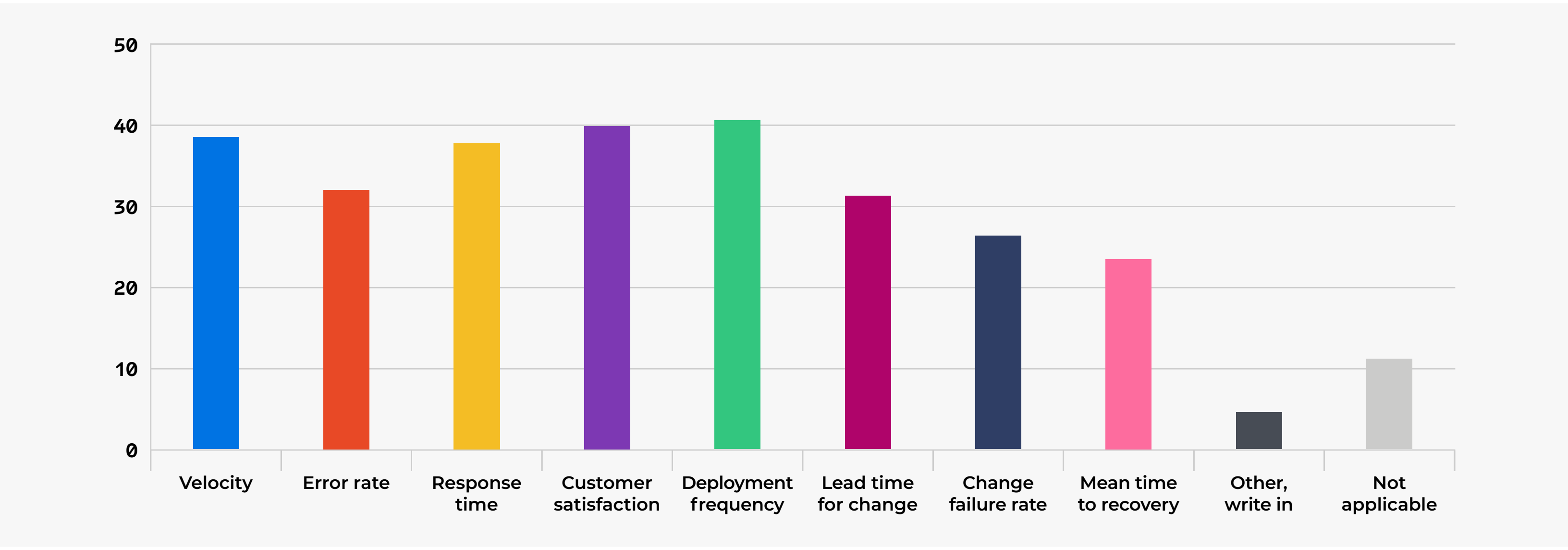


Figure 7. Deployment-related metrics organizations track



Observations:

1. The continuous delivery metrics selected most often were "Deployment frequency" (63%), "Change failure rate" (47%), "Production downtime during deployment" (46%), "Lead time" (44%), and "Mean time to recovery" (42%).

   While we changed the format of this question from last year's survey, asking respondents to select all metrics they found important rather than ranking the metrics by order of importance, we still observed similar results. The most selected metric this year, "Deployment frequency," was also the highest ranked metric last year.

   And the rest of the top five metrics this year were also in the top five in 2023, though their orders are rearranged — "Production downtime during deployment" was the second highest ranking metric last year, followed by "Mean time to recovery," "Lead time," and then "Change failure rate." Furthermore, "Error budget," the least popular metric this year by a 7% margin, was also ranked last overall in 2023 by a significant amount.

2. The most common metrics for organizations to track were "Deployment frequency" (41%), "Customer satisfaction" (40%), "Velocity" (38%), and "Response time" (37%). Last year, "Velocity" (52%) and "Customer satisfaction" (50%) were the most commonly selected metrics — ahead of "Deployment frequency" (45%), "Response time" (45%), and "Error rate" (43%) — which, in turn, was a shift from the year before, when "Deployment frequency" and "Error rate" were the most-tracked metrics overall.

   In the "Key Research Findings" of our 2023 *DevOps* Trend Report, we hypothesized that "[g]iven cultural changes surrounding work from home and other office policies in recent years, it's possible that this change in development velocity tracking indicates organizations have an increased concern with ensuring that tickets are being completed at an acceptable pace." And as these cultural fluctuations continue, this could still be an influencing factor.

   It is also reasonable to imagine that organizations, in general, are constantly shifting the metrics they track in response to any manner of internal and external factors. If so, it is unlikely we will see completely consistent results with regard to the metrics that organizations track the most at any given time.

Platform Engineering

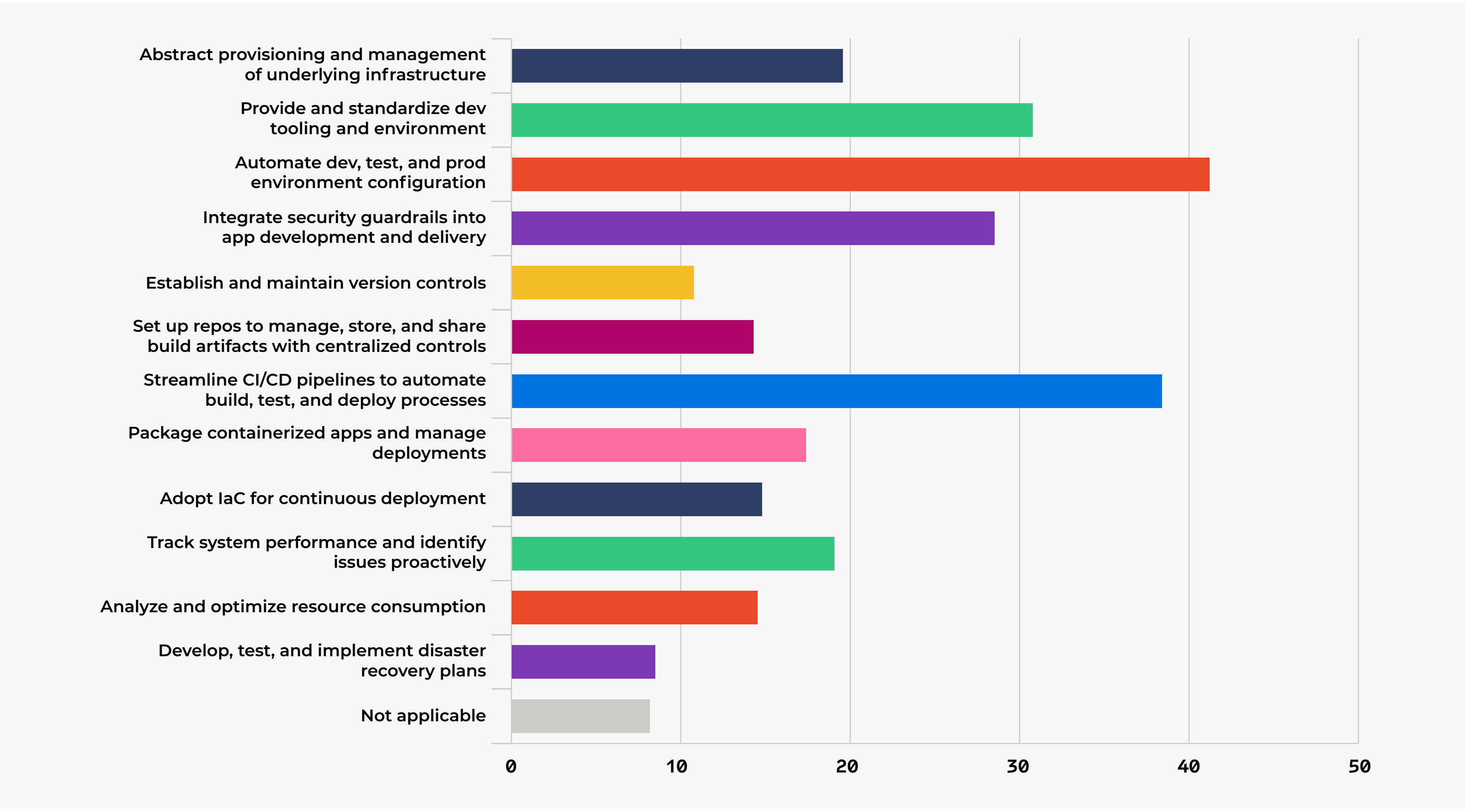
Platform engineering — the discipline involving designing, building, and maintaining the foundational infrastructure and tools necessary for efficient software development, deployment, and operations — focuses on creating a robust platform that supports the entire software lifecycle, emphasizing automation, scalability, and reliability. There are many potential benefits that organizations may find from platform engineering. To discover which advantages organizations are most likely to find appealing, we asked:

> *Select the top three areas in which your company uses, or plans to use, platform engineering for software development and delivery in the next 12 months.**

SEE RESULTS ON NEXT PAGE

Results (*n*=168):

Figure 8. Platform engineering uses or planned uses



*Note: Respondents were only able to select three responses in total.

Observations:

1. The most commonly selected areas in which platform engineering is used were "Automate dev, test, and prod environment configuration" (42%), "Streamline CI/CD pipelines to automate build, test, and deploy processes" (39%), and "Provide and standardize dev tooling and environment" (31%).

   These responses seem to fit well into the category of "automation of internal build and CD tooling," pointing toward a trend of platform engineering being predominantly used as a way to easily and uniformly ensure teams are on the same page.

   Options involving areas that might attempt to replace established systems — such as "Set up repos to manage, store, and share build artifacts with centralized controls" (14%) or "Establish and maintain version controls" (11%) — as well as options that may involve relinquishing control of primary internal systems — such as "Abstract provisioning and management of underlying infrastructure" and "Analyze and optimize resource consumption" (14%) — were significantly less likely to be selected as one of the top three areas of platform engineering use.

2. While the general categorical trend mentioned in the previous observation did not fluctuate based on respondents' organization sizes, response rates to individual options did. Respondents at large organizations were most likely to select "Streamline CI/CD pipelines to automate build, test, and deploy processes" (45%) over "Automate dev, test, and prod environment configuration" (42%). Respondents at mid-sized organizations were most likely to select "Provide and standardize dev tooling and environment" (40%), over both "Automate dev, test, and prod environment configuration" and "Streamline CI/CD pipelines to automate build, test, and deploy processes" (38% each).

   Furthermore, respondents at small organizations, while their top two selections of "Automate dev, test, and prod environment configuration" (46%) and "Streamline CI/CD pipelines to automate build, test, and deploy processes" (34%) aligned with the overall average, their response rates for "Abstract provisioning and management of underlying infrastructure" tied their rates for "Provide and standardize dev tooling and environment" (23%). And response rates of those at small orgs for "Integrate security guardrails into app development and delivery" and "Track system performance and identify issues proactively" (21% each) were very close behind.

These results imply that smaller organizations may be more likely to utilize platform engineering to either complement or override pre-existing or third-party security/performance systems.

Future Research

Our analysis here only touched the surface of the available data, and we will look to further refine and expand our DevOps survey as we produce future Trend Reports. Some of the topics we didn't get to in this report, but were incorporated in our survey, include:

- Frequency of production and new feature releases
- Manual and automated testing
- Amount of legacy vs. non-legacy code
- Frequency of incidents and rollbacks
- Amount of technical debt
- Responsibility for coordinating and monitoring deployments
- Toolchain orchestration tools

Please contact publications@dzone.com if you would like to discuss any of our findings or supplementary data.

G. Ryan Spain

@grspain

@grspain

@grspain

gryanspain.com

G. Ryan Spain lives on a beautiful two-acre farm in McCalla, Alabama with his lovely wife and adorable dog. He is a polyglot software engineer with an MFA in poetry, a die-hard Emacs fan and Linux user, a lover of The Legend of Zelda, a journeyman data scientist, and a home cooking enthusiast. When he isn't programming, he can often be found playing Super Mario RPG with a glass of red wine or a cold beer.

PARTNER CASE STUDY

NTT Docomo

Transforming Complex Legacy IT System
Into a Cloud-Native Environment

CREATED IN PARTNERSHIP WITH

Red Hat

NTT Docomo, Inc.
Telecommunications
7,903 employees
215 group companies
Tokyo, Japan

Solutions Used
Red Hat® OpenShift®, Red Hat Application Foundations, Red Hat Consulting

Primary Outcomes
NTT Docomo decluttered its source code, reimagined its existing system with a CI/CD pipeline, and developed a comparison tool to simplify the move from legacy to cloud.

"There are many container platforms available, but Red Hat OpenShift stood out for its amazing stability, scalability, and flexibility. We also appreciated that it has been proven through a large number of deployments around the world."

—**Takanori Nagahara**, *JChief of Application Development, Service Design Department, NTT Docomo*

Company information and quote affiliations and titles are as of time of interview (March 2023).

Read the full case study here.

Japan's largest mobile telecom services provider, **NTT Docomo**, expanded the potential of cell phones as IT infrastructure by incorporating a diverse array of functionalities, accelerating evolution under the slogan of: "Changing worlds with you."

Challenges

NTT Docomo began offering internet services for cell phones in 1999, and the related back-end system evolved rapidly to meet resulting demands. This system consolidates sub-systems for deploying digital content for 3G and 4G devices, billing integration, and other critical functions.

As services were added or expanded to support registering carriers' custom mobile apps and distributing apps to devices, system maintenance quickly grew complex. Within just one service on the back-end system, branching processes for different use cases generated approximately: 800 screens, 8,000 pages of written specifications, and 500,000 lines of source code.

NTT Docomo generally replaces hardware on a five-year maintenance schedule, but during every hardware migration, they encountered major problems. To simplify future migrations, they decided to modernize the entire back-end system by transforming it into a cloud-native architecture.

Solutions

Based on research into container tech for cloud-native systems, and the availability of certified middleware, NTT Docomo chose to use Red Hat OpenShift for their new back-end system's environment. They also adopted Red Hat Application Foundations — a comprehensive set of components and technologies that helped build, deploy, and operate their apps across hybrid cloud environments — as well as Red Hat Consulting to expedite and support their refactoring efforts.

Results

Combining Red Hat OpenShift and Red Hat certified middleware helps NTT Docomo teams efficiently design, develop, and manage their cloud-native applications. Through their partnership, NTT Docomo:

- **Enhanced their user interface.** Reducing the number of screens in their legacy system by 75% through reviewing and simplifying functions created a more intuitive experience for their services' users.
- **Established a CI/CD pipeline.** Adopting Red Hat OpenShift Pipelines allowed them to automate app development, reducing development time and total cost of ownership.
- **Developed a comparison tool.** Collaborating with Red Hat's consultants, they created a solution to automatically compare their legacy and new systems to identify unneeded source code. Using the tool during the refactoring process helped NTT Docomo shorten their app release cycle by 87.5%.

Application innovation. Everywhere. Every time.

Application development changes constantly, but the demand to deliver faster never does. To keep up, you need to be able to adapt.

Flexibility keeps your options open, and you can get it with Red Hat OpenShift. A comprehensive application platform that empowers teams to choose the right tools and techniques.

Deliver consistently with the user experiences your customers have come to trust.

Learn more

Red Hat

DZone.

3343 Perimeter Hill Dr, Suite 100
Nashville, TN 37211
888.678.0399 | 919.678.0300

At DZone, we foster a collaborative environment that empowers developers and tech professionals to share knowledge, build skills, and solve problems through content, code, and community. We thoughtfully — and with intention — challenge the status quo and value diverse perspectives so that, as one, we can inspire positive change through technology.

Copyright © 2024 DZone. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means of electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.