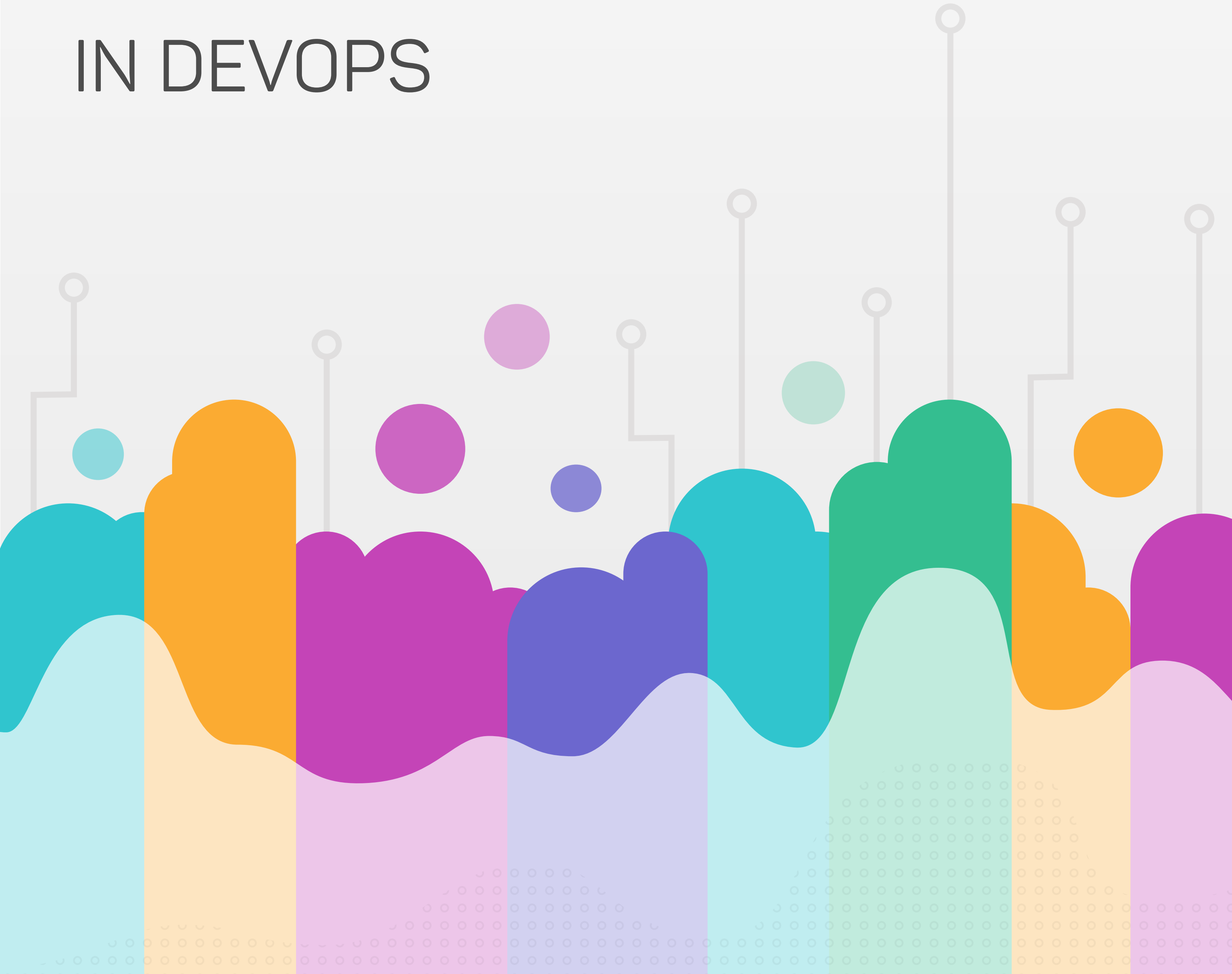
mabl

Benchmark Report:

THE STATE OF TESTING IN DEVOPS



Introduction

Welcome to the second installment of the Benchmark Report: The State of Testing in DevOps! Every year we send out a survey to gain insight into how leading DevOps practices are benefiting the business and the developer experience (DX). We analyze the results through the lens of testing to understand how QA processes are affected by DevOps transformations and look for key ways that teams can improve upon, ensuring product quality while shipping at speed.

The data for this report comes from the results of a public survey that we ran from September 2019 to March 2020 and shared with the software testing and development community via our socials, conferences, and on technical news and article platforms. This survey was opened up to all roles that affect software quality, and we received about 1030 responses from testers, developers, operations and site reliability engineers, and managers, globally.

In last year's survey, we sought to understand how much testing is being done manually versus automated in the CI/CD pipeline and how various QA processes affect stress levels and product quality. You can take a look at last year's results **here**. This year, we're focusing testing processes as well as team structure and culture affects the customer experience.

Why do we focus on testing in particular?

DevOps emerged in order to help software teams bring value to customers faster; QA no doubt has a huge role in the customer experience. But as DevOps delivery practices are being adopted across the industry, more often than not, they're built without quality assurance as a core capability. Teams have no choice but to build a testing strategy as they go, rather than designing the culture, processes, and tools upfront. Not having a solid QA foundation is like not seeing the forest for the trees. Leaving QA testing as an afterthought in an DevOps or CI/CD environment results in measurable inefficiencies across the entire development team, and oftentimes negatively impacts the user experience.

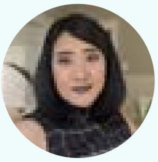
The first step to solving any problem? Understanding it. As more organizations make the shift to DevOps and adopt rapid release processes, testing and QA roles are given incredibly broad definitions. Depending on the organization, testers can be part of a development team or be siloed off on their own team. They can be responsible for getting testing into the entire software delivery pipeline, or only be responsible for manual testing after a build is code complete.

To understand the inefficiencies of QA in DevOps, we've mapped out the current landscape of testing in DevOps to unveil what development and testing processes result in world class software, such as:

- CI/CD and cloud adoption
- Integrating QA into development
- Automation and manual processes
- Team structures that work for the employees and for your customers

The Benchmark Report: The State of Testing in DevOps survey was structured to first identify the respondents, then get to know the basics of their delivery practices, and finally understand the effectiveness of their practices.

Please feel free to tweet out any interesting findings you come across in the report and share it with the **#devtestopssurvey** hashtag! We hope you enjoy the read. Now let's dive into the data.



Chou Yang
Content & Brand Marketing
mabl

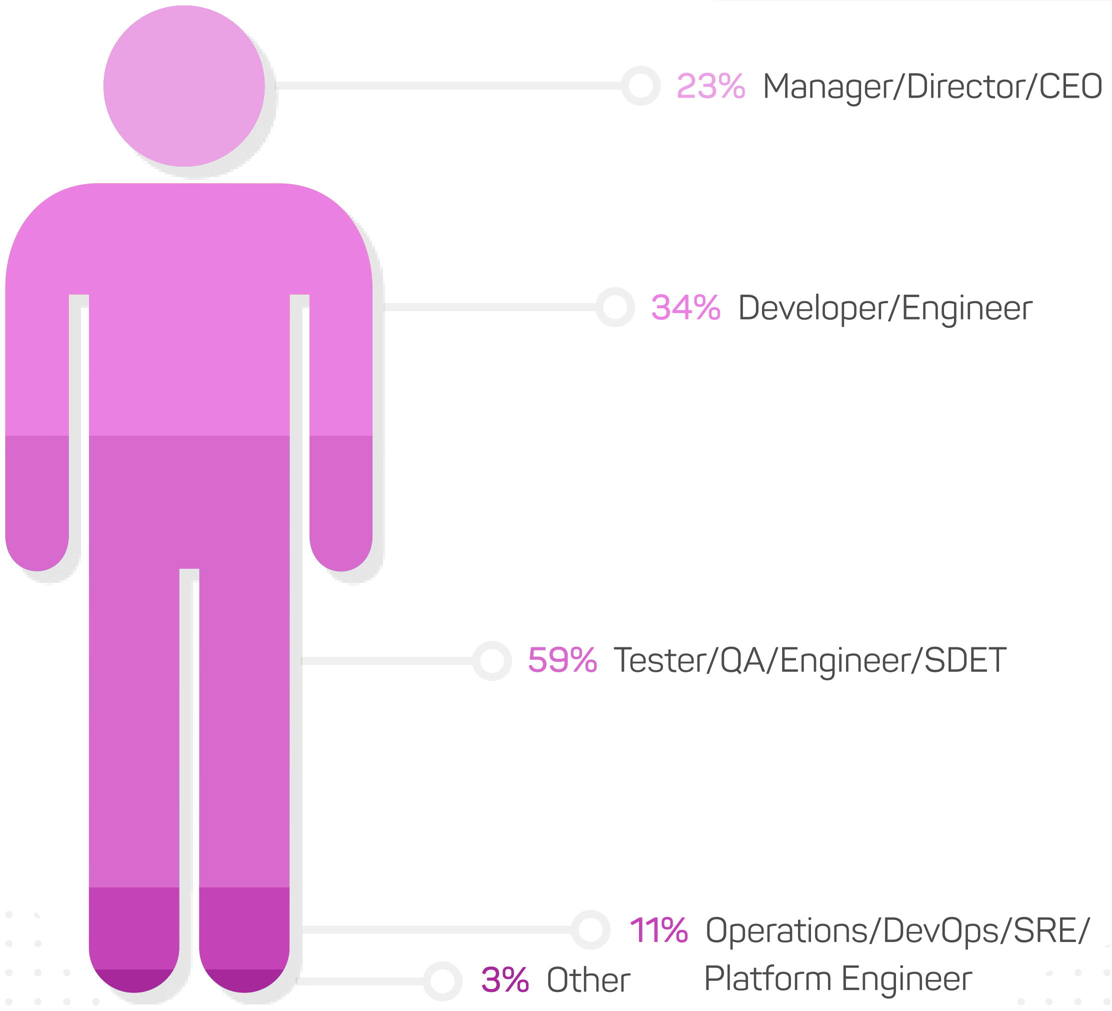
@mablhq www.mabl.com

Table of Contents

The Respondents

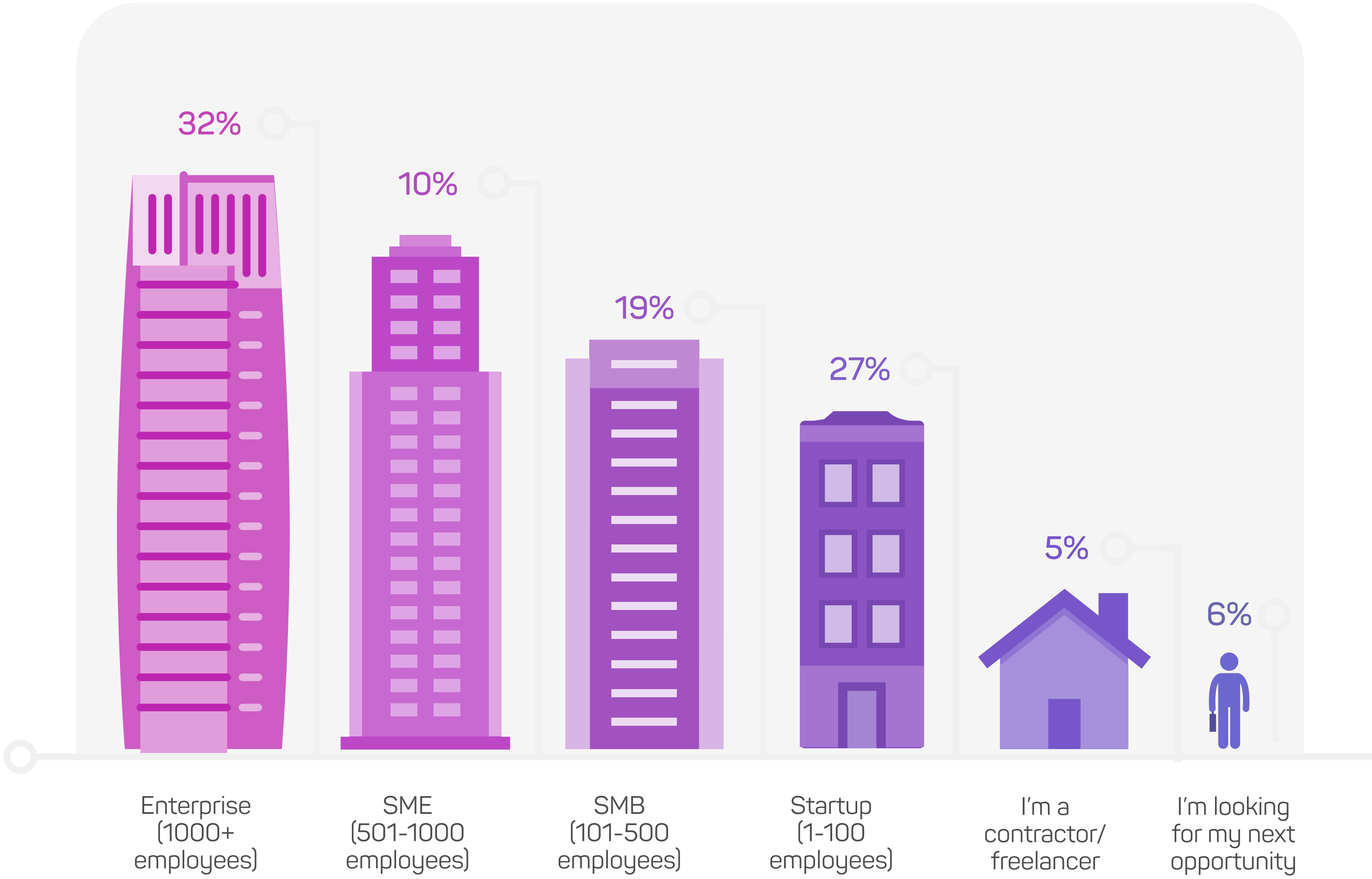
To get context around the responses, we'll lay out some basic demographics. Most of our respondents were in various QA roles, with everything from manual testers to Software Development Engineers in Test (SDETs) coming in at 59% of respondents.

"What best describes your role?"



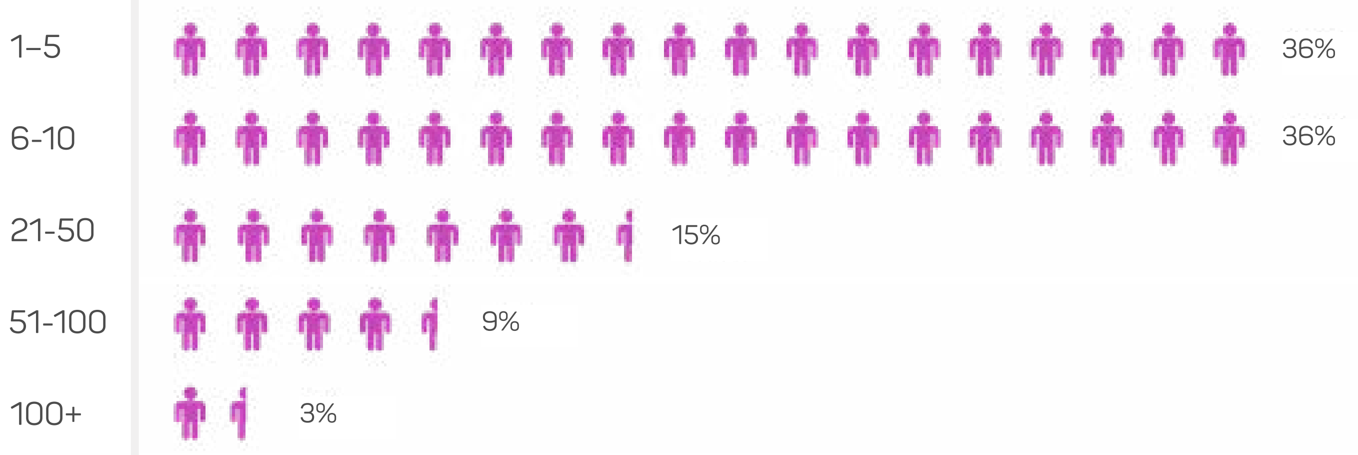
About a third of our respondents worked at an Enterprise company with 1000+ employees.

“What’s your company size?”



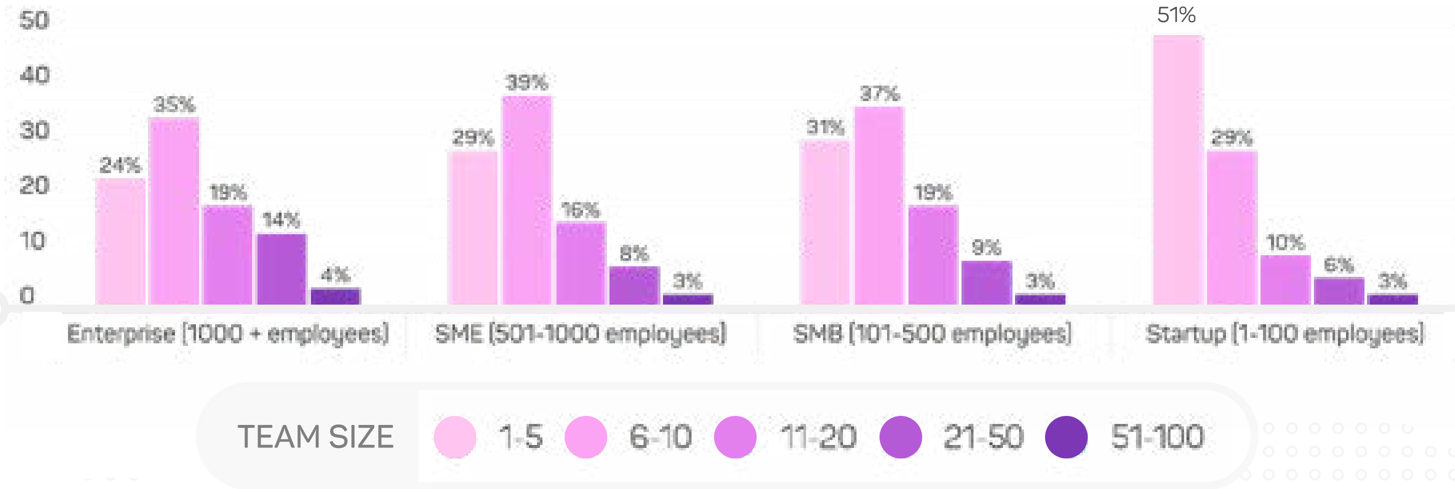
About two-thirds of respondents worked within a team of 10 or fewer members. Regardless of the size of the organization, most of the team sizes remained between 6-10 members, except for startups, with most teams only containing up to 5 members.

"How big is your immediate team?"



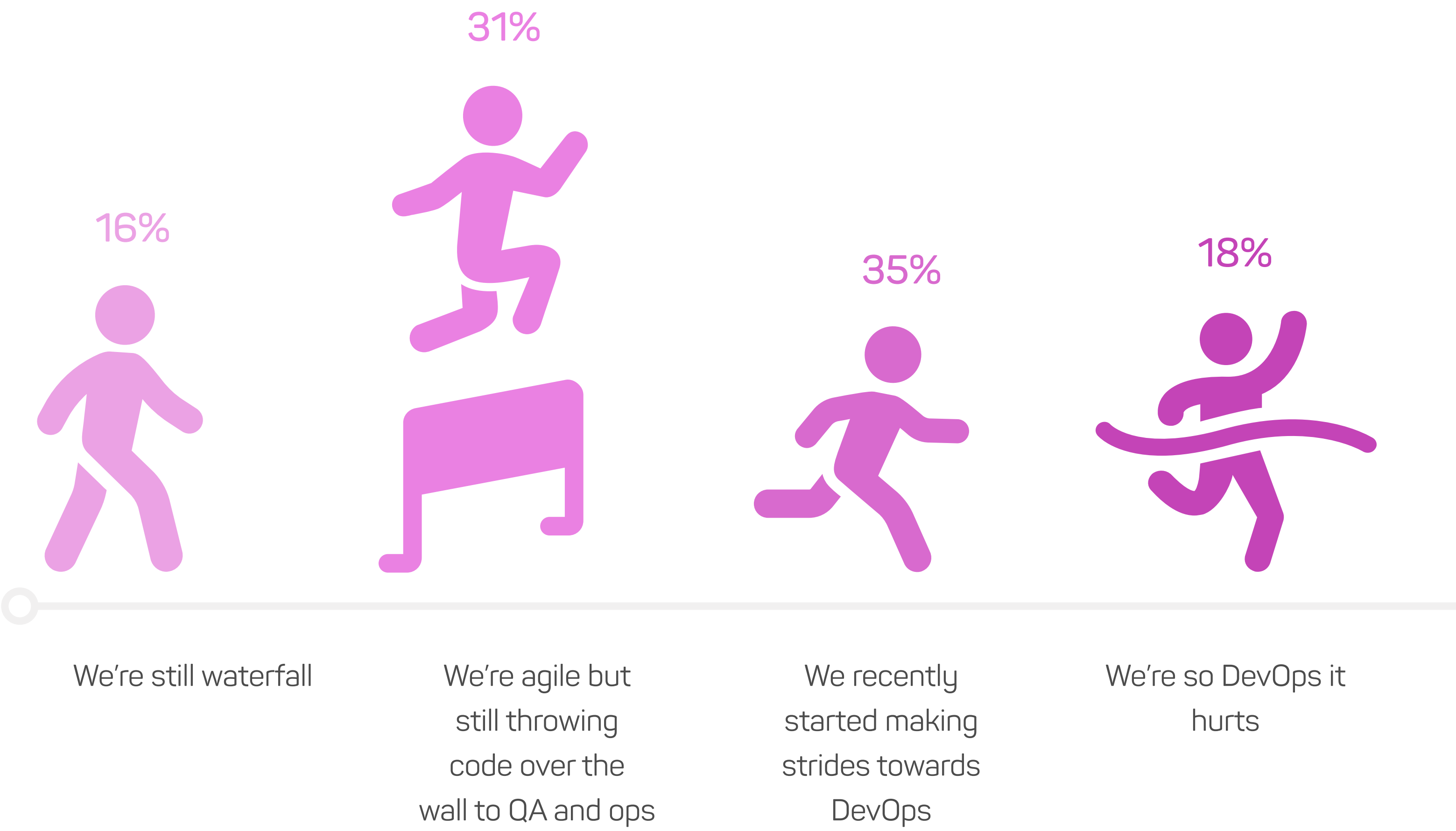
"How big is your immediate team?"

By Organization Size



One of the most important things we wanted to know about our respondents is how far along they are in their DevOps transformation journey. We can use the waterfall teams as a looking glass into the past, our bleeding edge DevOps teams to gauge where most teams are headed in the next couple of years, and compare the "past" and "future" trends to the majority of responses today. About 60% of the respondents were somewhere in the middle between "old-school-not-so-cool" and bleeding edge. The largest segment of our respondents were making good progress towards DevOps.

"Where are you in the DevOps transformation?"

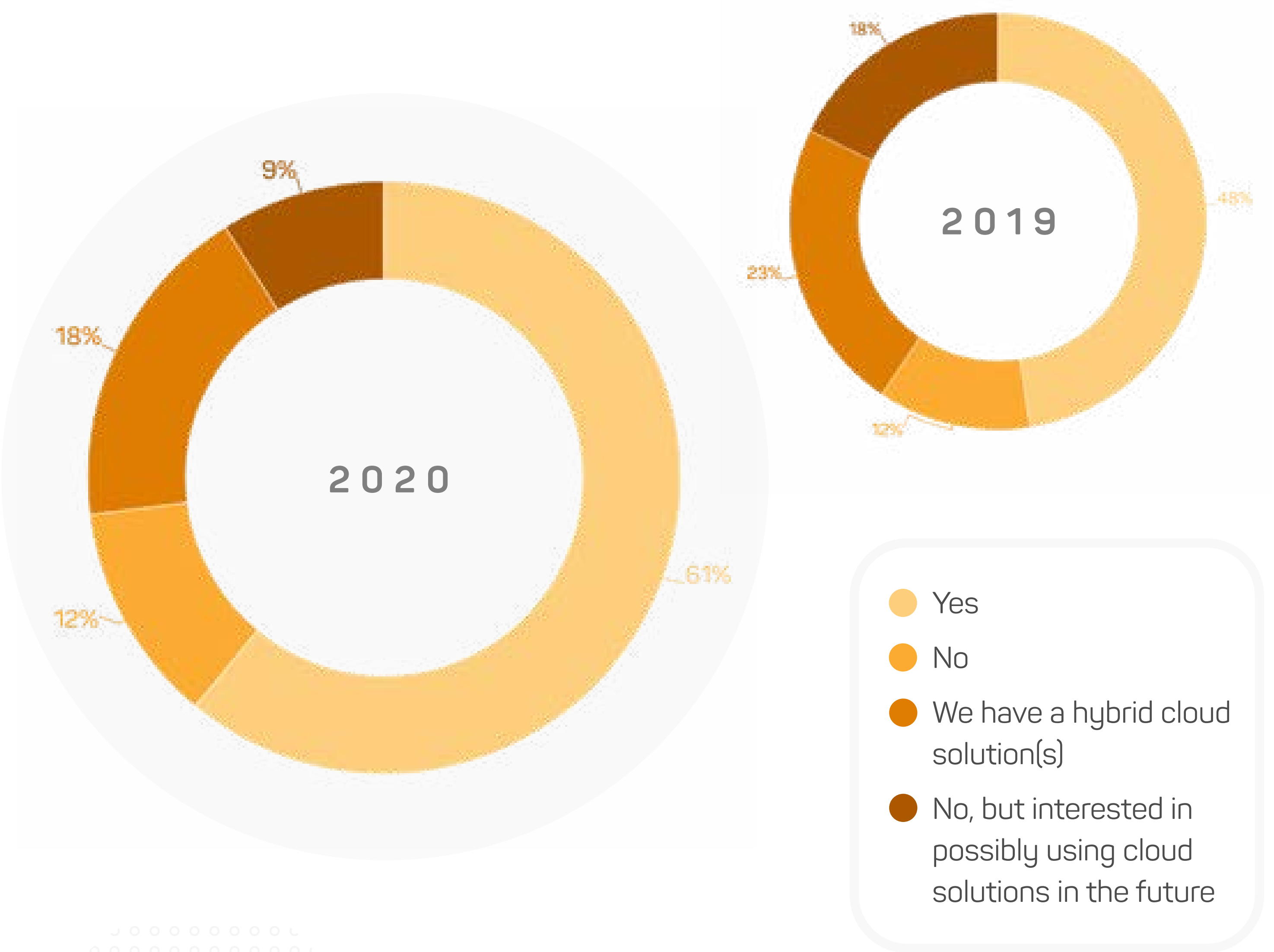


Software Delivery Practices

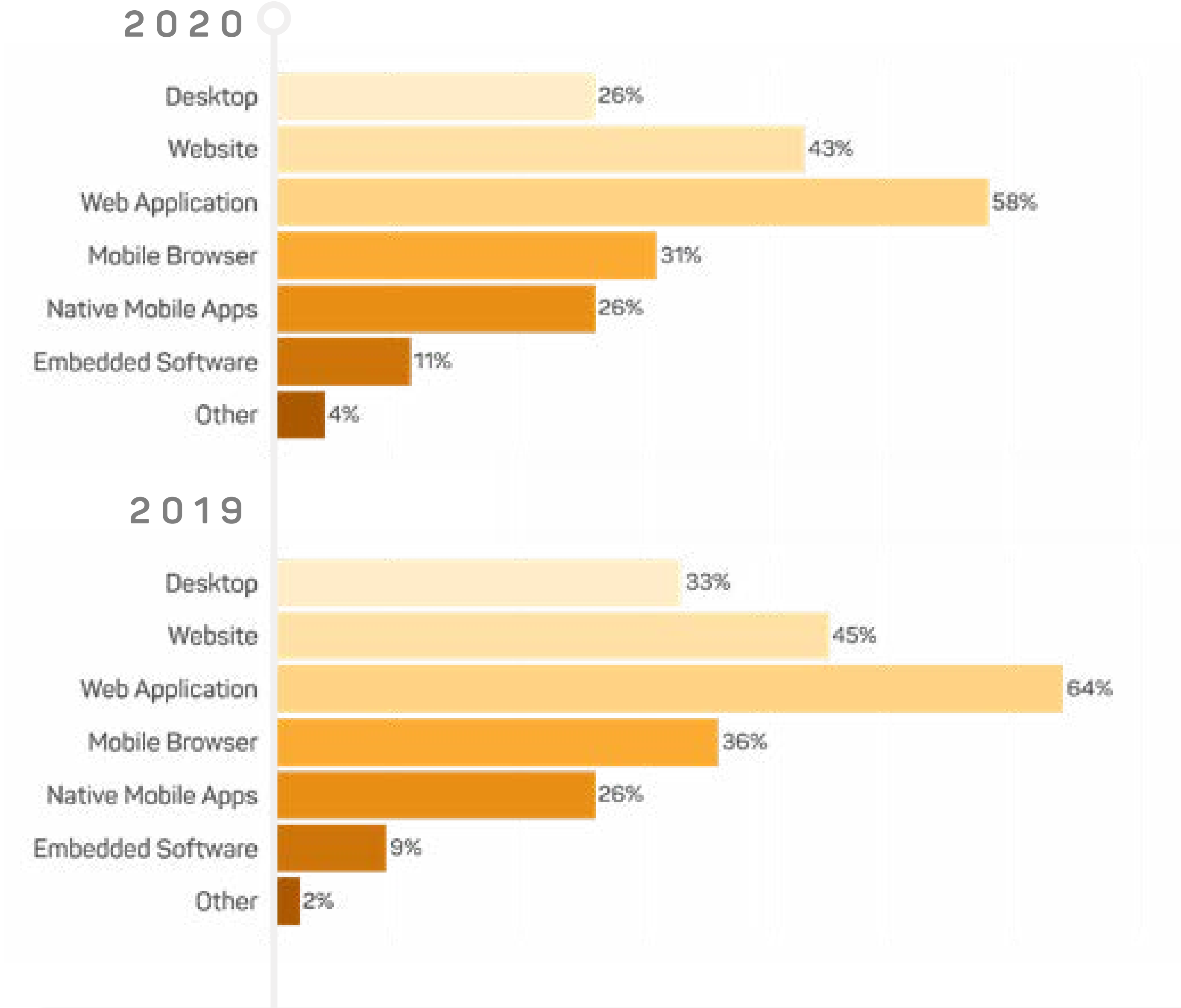
Let's dig into software delivery practices. We cover various areas such as deployment frequency, tooling, CI/CD and cloud adoption, and QA processes. In this section, we'll compare data from last year's survey to responses this year (where applicable) and outline any trends we noticed. We'll segment the data based on respondent personas later on, so keep reading until the end!

In both last year's and this year's survey, we asked about CI/CD and cloud adoption. There was a 13% growth in cloud adoption, and a drop in hybrid solutions and a drop in teams that aren't using cloud technologies right now but are considering it in the future.

"Does your organization prefer to use cloud solutions (Azure Cloud Platform, Amazon Web Services, etc) over on-premises solutions?"

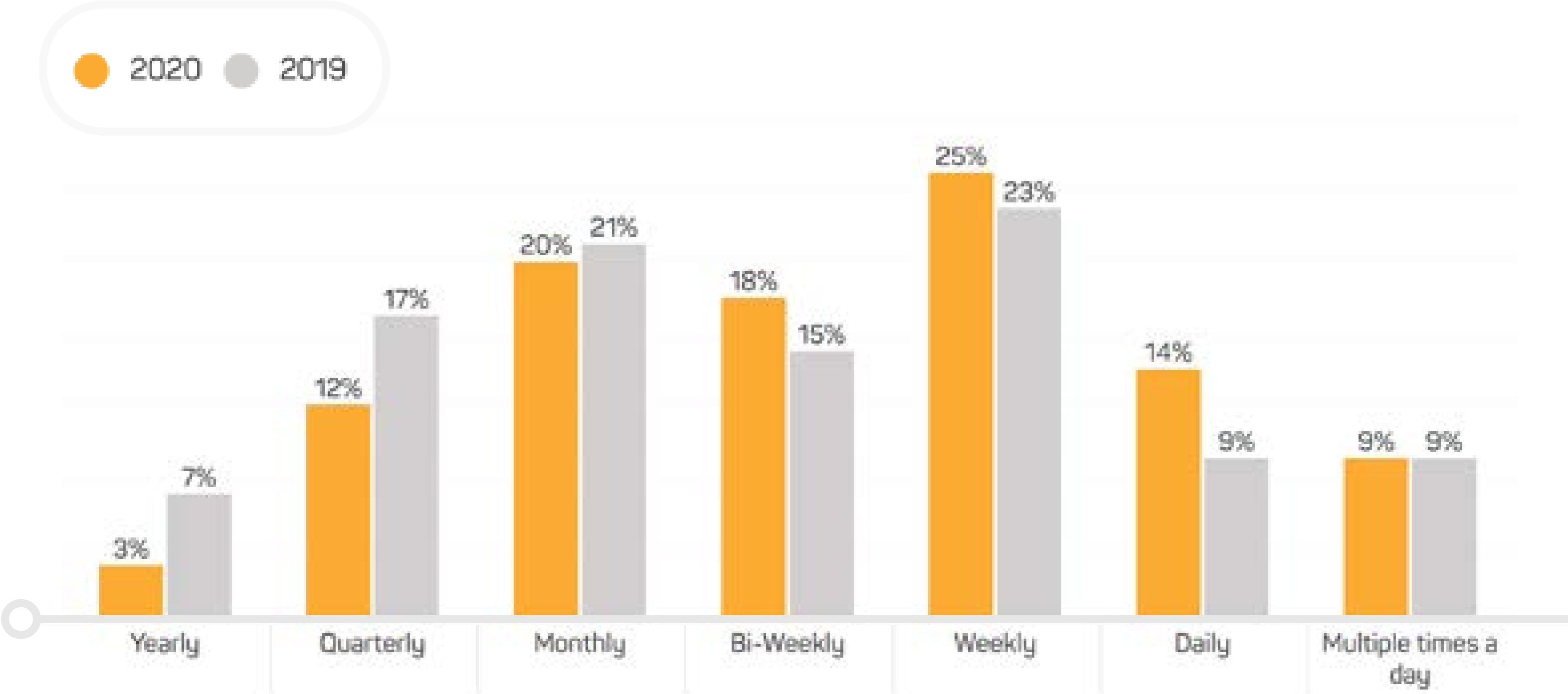


"What type of applications does your organization test? Select all that apply."



No surprise that web applications are still leading the industry, but the mobile experience has an impressive presence with nearly a quarter to a third of respondents reporting testing either mobile browser or native mobile apps. Desktop has dropped slightly from last year, but there was a general flattening of the curve all around, so it might not actually be as steep of a drop as it looks like from the numbers alone. Still, I think we can still safely assume that Desktop is on its way out, following the trend we've seen over the past decade.

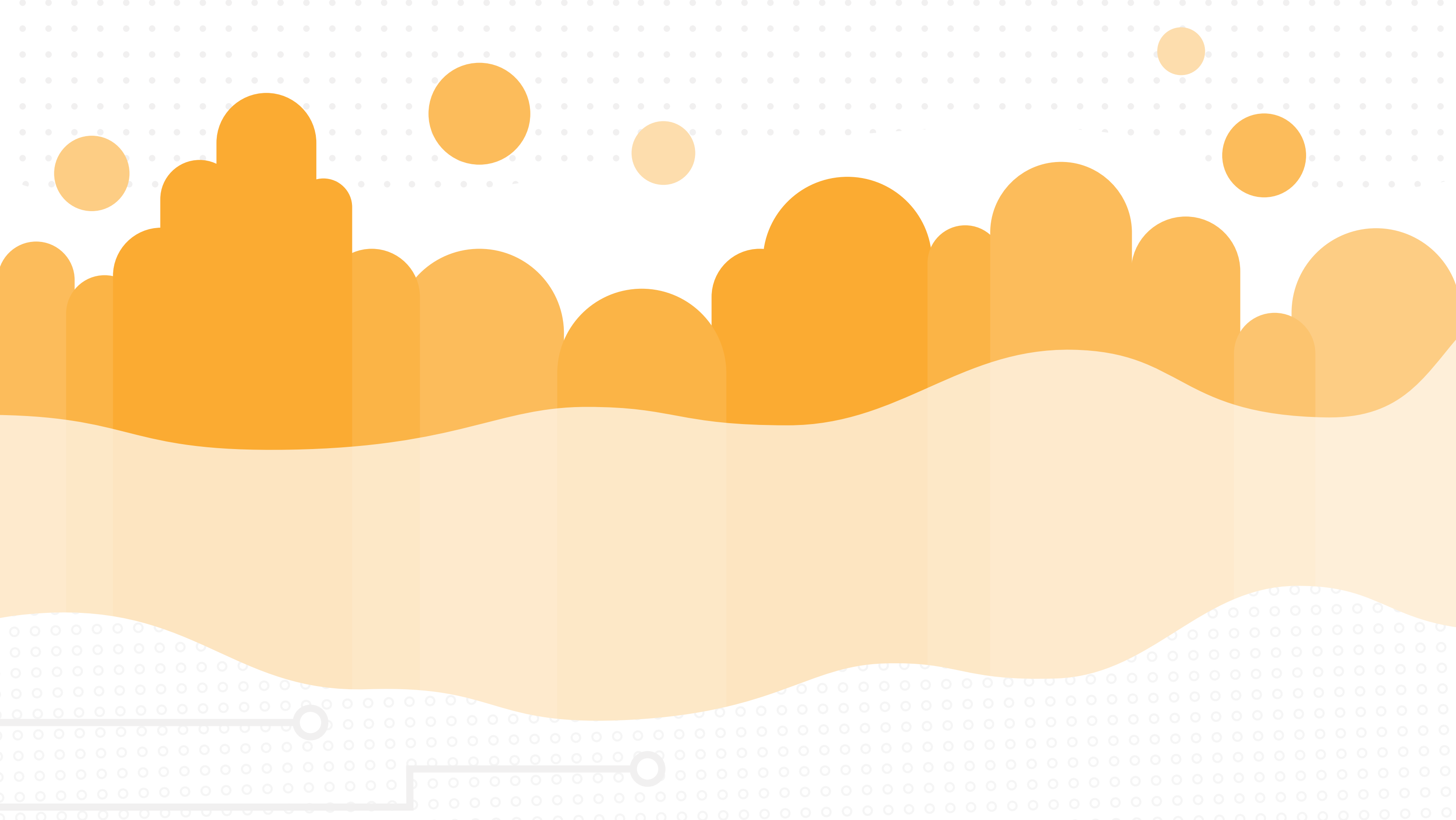
What best describes how frequently your team deploys new code into production?



Last year there was a fairly even split between respondents who shipped on a monthly and weekly basis. Given the further adoption of DevOps, this year we see weekly deployments pulling slightly farther ahead with a quarter of respondents shipping on a weekly basis. We see a drop in yearly and quarterly deployments alongside an overall growth in bi-weekly, monthly, and weekly deployments.

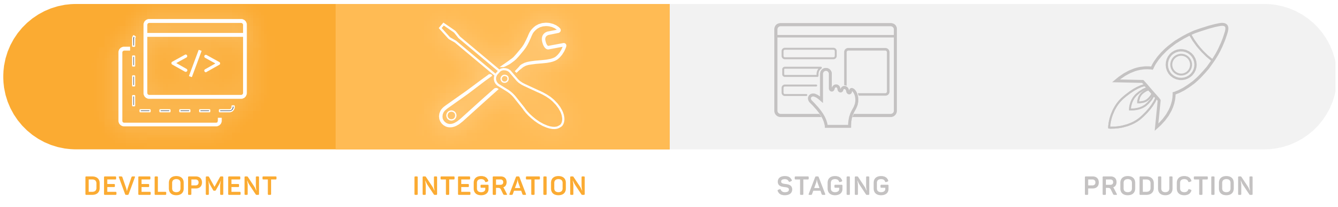
CI/CD (continuous integration and delivery) is a popular term that, in many minds, is almost synonymous with DevOps. CI/CD ensures that code moves smoothly between stages of app development with ongoing, automated testing and integration. Not all CI/CD is the same across organizations. For example, some teams may have an automated CI/CD pipeline between development and staging, then leave the deployment process to production as a manual process. Others may only have an automated CI/CD pipeline from development to an integration environment. The automated tests within the CI/CD pipeline can completely vary as well.

The semantics of continuous integration, delivery, and deployment aren't necessarily important and every team has different needs, so implementing all three isn't necessarily best practice. We can compare CI/CD to a road. Speed bumps are placed intentionally to ensure that drivers are slowing down where the risk for accidents are high. On the other hand, a car accident is an inconvenient blocker that causes traffic to bottleneck. Understanding how widely and to what extent CI/CD is adopted could help us clearly differentiate between processes that are healthy speed bumps in the software pipeline and which are actually breaking that pipeline.

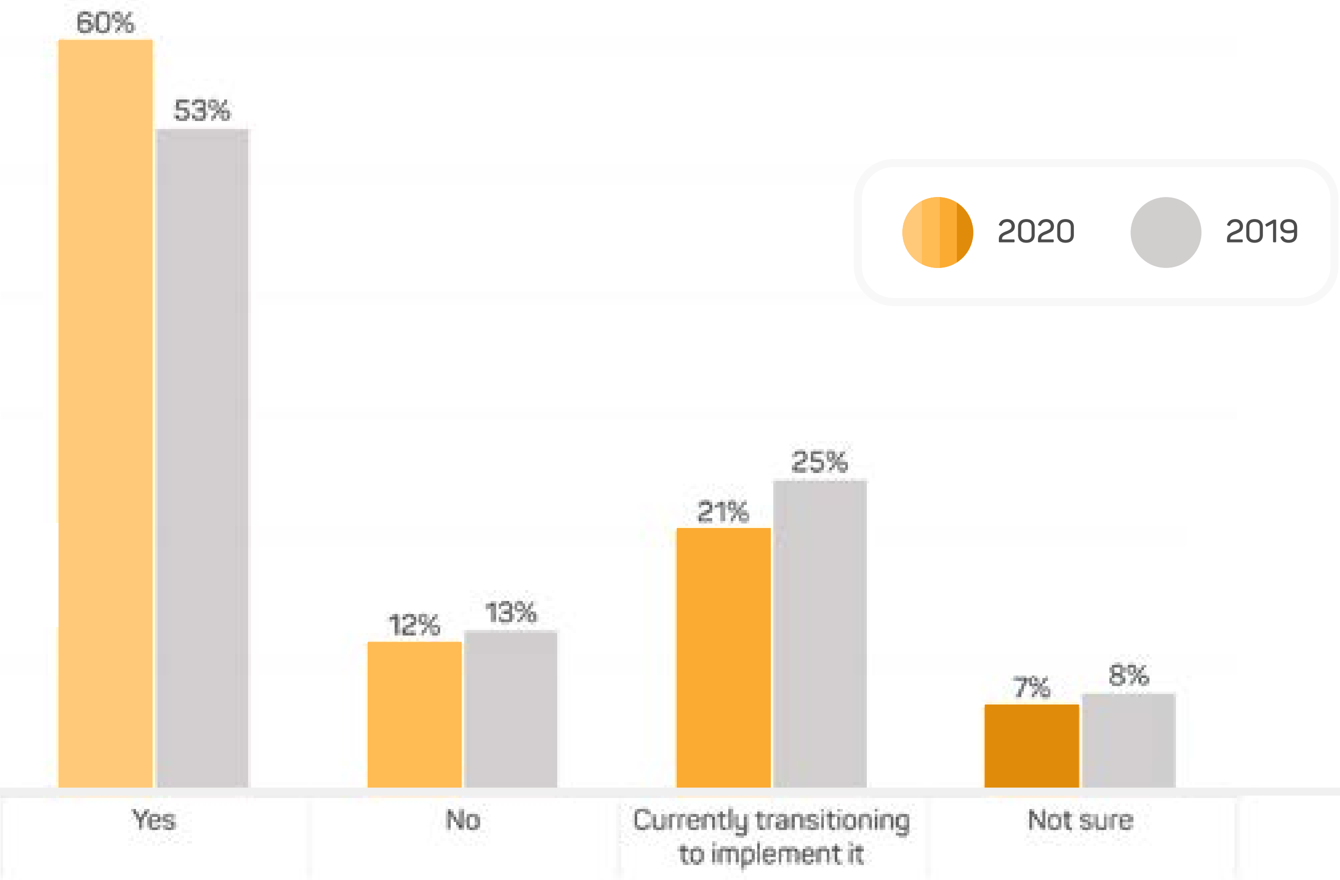


Continuous Integration is where code changes are automatically pushed from development to integration environments, where disparate pieces of newly written code are assembled together and tested as a whole.

CONTINUOUS INTEGRATION



"Has your organization adopted Continuous Integration (CI)?"

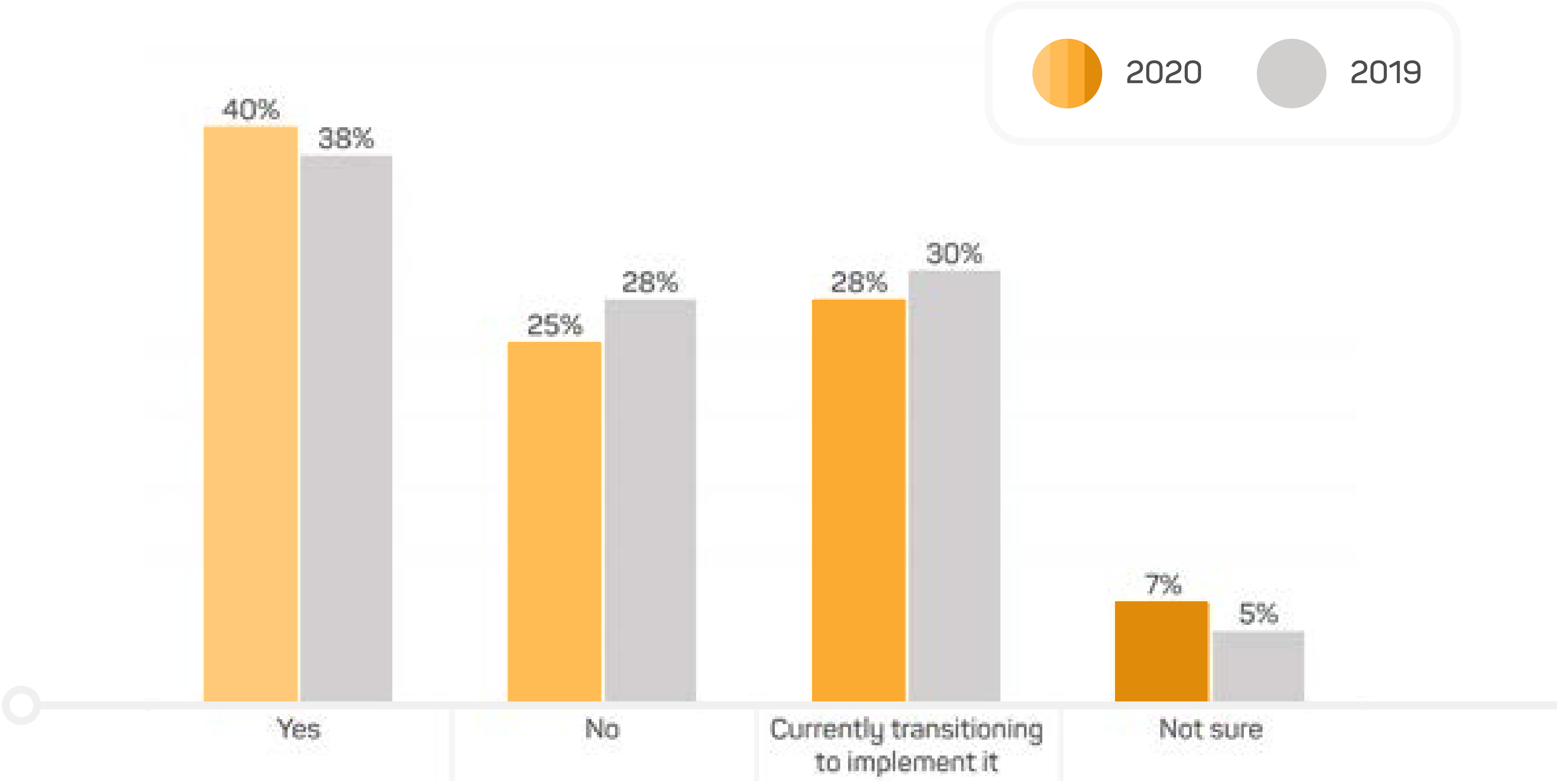


Continuous Integration adoption has grown 7% since last year and there has been a 4% drop in those who are in the planning stages of implementation.

CONTINUOUS DELIVERY

DEVELOPMENT
INTEGRATION
STAGING
PRODUCTION

"Has your organization adopted Continuous Delivery (meaning every code change is delivered to a staging environment using complete automation)?"

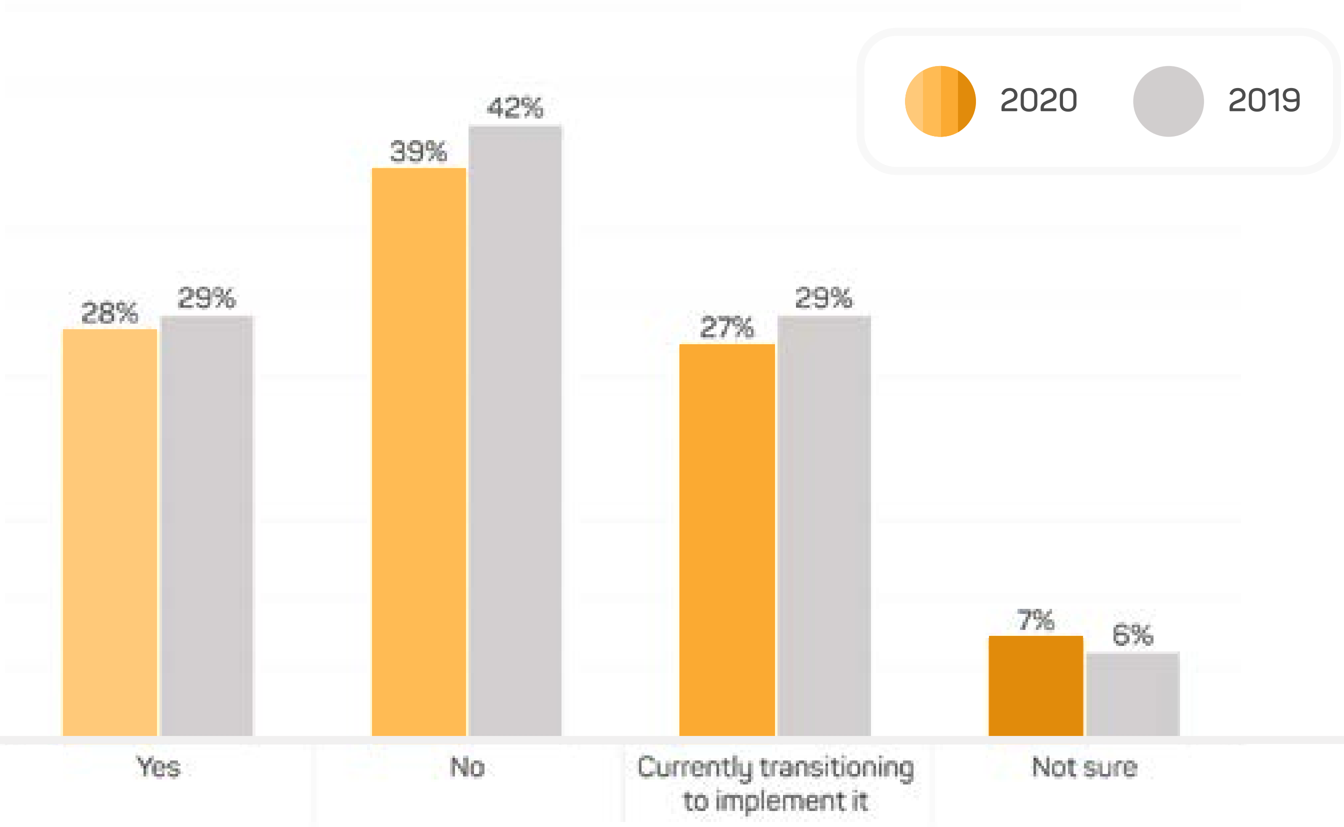


Continuous Delivery lets you promote code changes to a staging environment automatically. This year we see a small 2% shift up in adoption.

CONTINUOUS DEPLOYMENT

DEVELOPMENT
INTEGRATION
STAGING
PRODUCTION

"Has your organization adopted Continuous Deployment (meaning every code change is delivered to a production environment using complete automation)?"



Continuous Deployment lets you promote code changes to production automatically. This year saw a minor drop in implementation.

It's interesting to see non-linear curve for adoption of CI/CD instead of flat growth across all phases of development. A big reason for this could be that automating end-to-end tests in later stages of software delivery is difficult to do well. As a result, testing still ends up being a stopgap instead of a baked-in process of the pipeline.

It's also possible teams haven't set up ways to measure risk, or have already deemed it too risky to fully automated their pipeline.

Understandably, it doesn't make sense to deploy straight to production without the safety net of proper production monitoring systems and the ability to roll back releases quickly. Continuous testing and monitoring in production is vital to being able to confidently ship quickly, so we'll be talking about some of the quality monitoring activities that you can do in production (known as shift-right testing) later on in this report.

INTEGRATE AUTOMATED UI TESTING ACROSS THE ENTIRE DEV ECOSYSTEM

Mabl's many integrations with the top CI/CD and DevOps tools make it easy to incorporate intelligent test automation into your software delivery pipeline, and enable developers and QA engineers to test effectively while quickly releasing high-quality product updates.

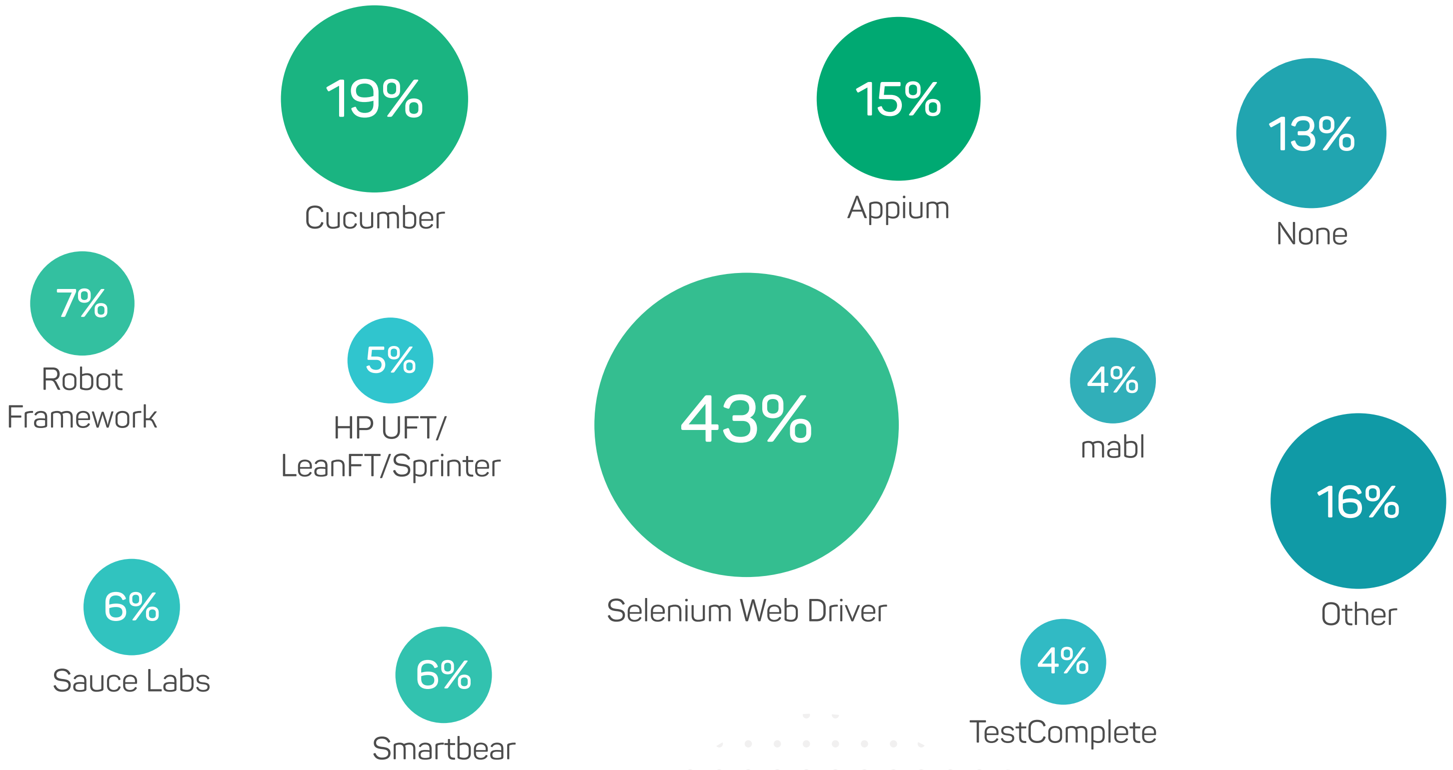
Try mabl for free at mabl.com

Testing in DevOps

Though tools are only as good as their users, but we were still curious to see what test tooling adoption looks like. To keep the charts concise, we did not list any tools that received less than 1% of responses.

Selenium Web Driver still reigns supreme, though there has been a 7% drop in its usage since last year. With so many new test automation solutions available now that make UI test automation easier than using Selenium alone, we're curious to see if Selenium adoption will continue to drop in the years to come.

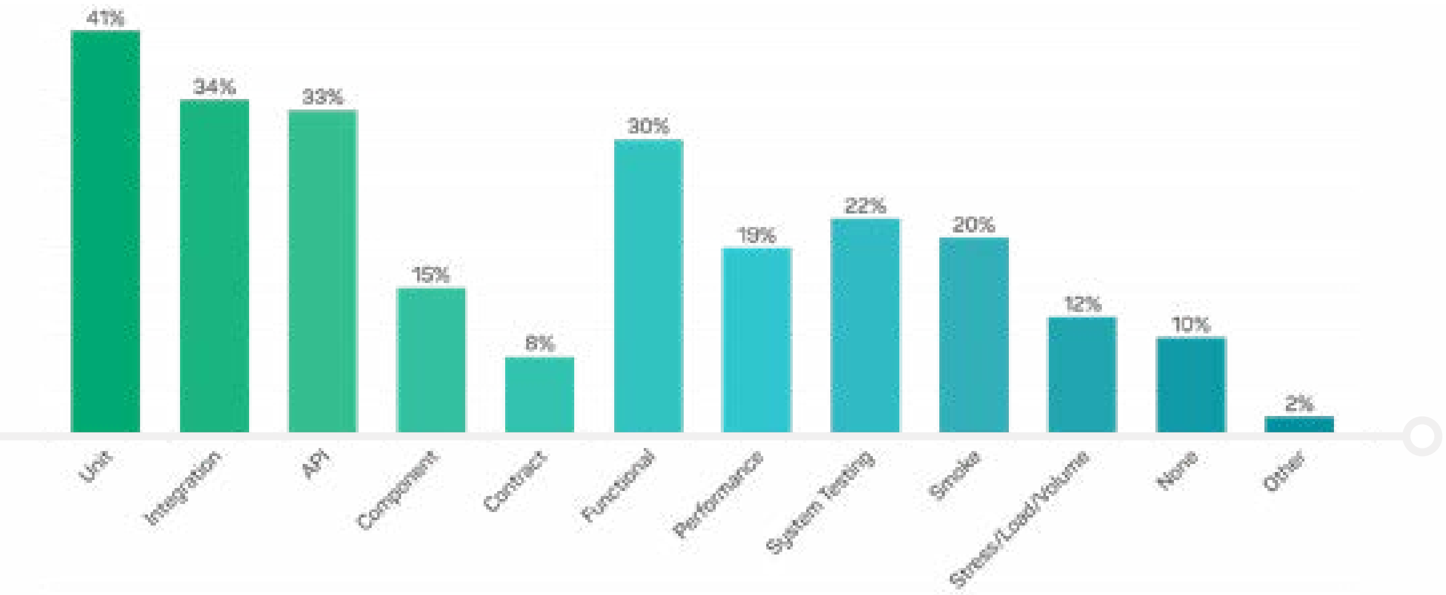
"What testing tools do you use? Select all that apply."



Let's take a look at what types of tests are automated in the CI/CD pipeline. We can see that unit tests are significantly more likely to be automated and system tests are significantly more likely to be performed manually.

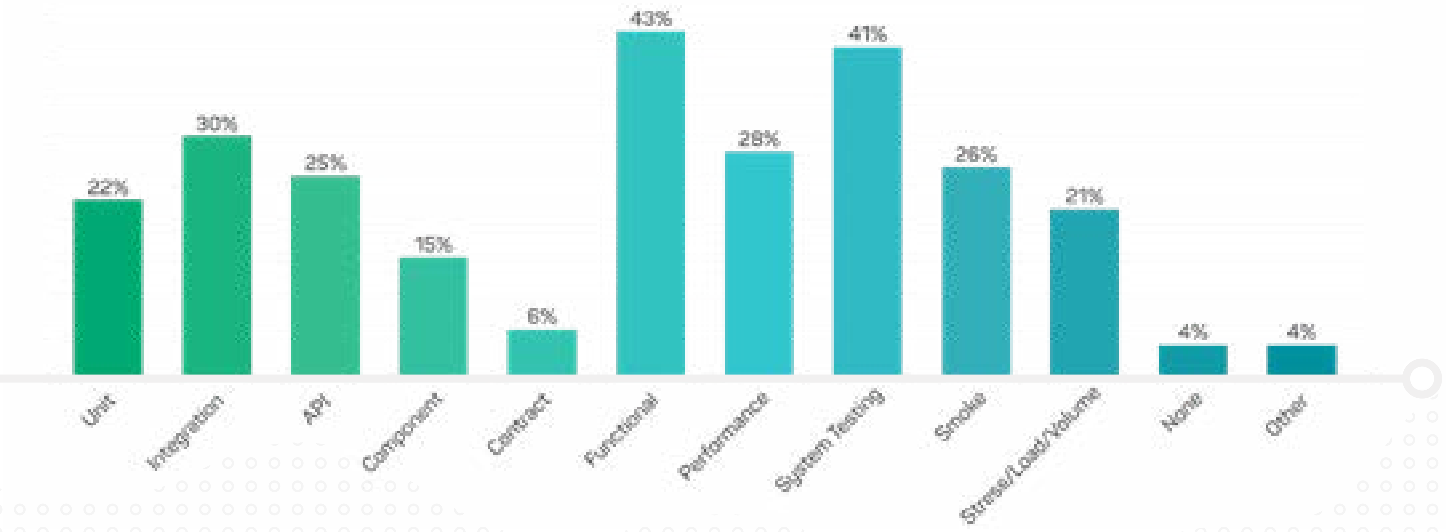
"What types of testing do you ***automate*** *in your CI/CD (continuous integration and deployment) pipeline?"*

Automated Tests



"What types of tests do you ***conduct manually*** *(not triggered automatically or scheduled)?"*

Manual Tests



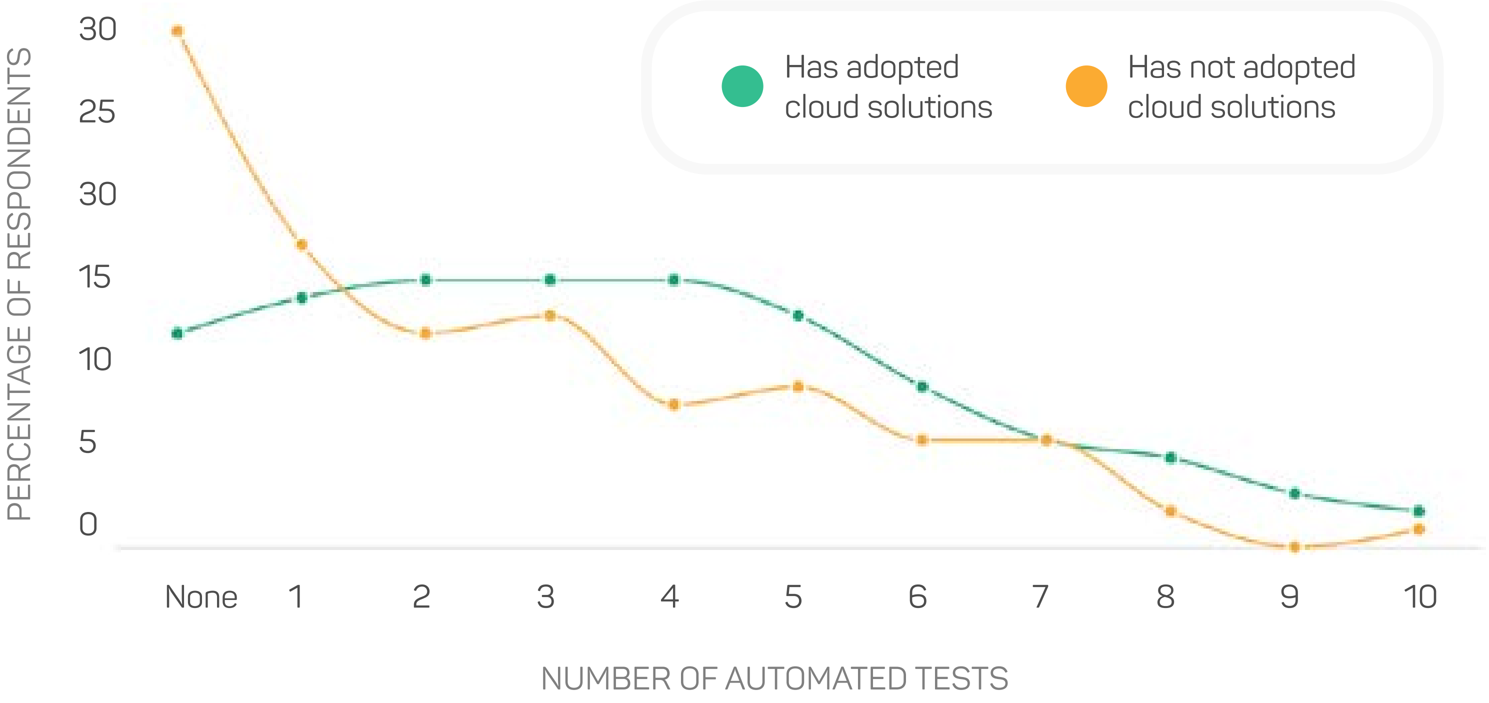
Surprisingly, the adoption of continuous integration, delivery, and deployment had no impact on the number of different types of tests automated. However, we did notice that teams who were using some cloud technologies had about 1.2 more types of tests automated in their CI/CD pipeline on average.

The benefits of adopting cloud infrastructure for its elasticity have been well-known for many years. All the same, it's hard to ignore the numbers that show that cloud enables more test automation. Maintaining test infrastructure in-house can be a huge burden and pulls team members away from increasing test coverage and building a scalable testing strategy.

Of the respondents who reported that they automate every type of test we listed in the survey (10 types), 93% of them were using cloud technology. 60% of those with 4 or more tests automated were using cloud. Of those who reported having no automated tests, 70% had not adopted any cloud infrastructure.

Number of automated tests

By Cloud Adoption

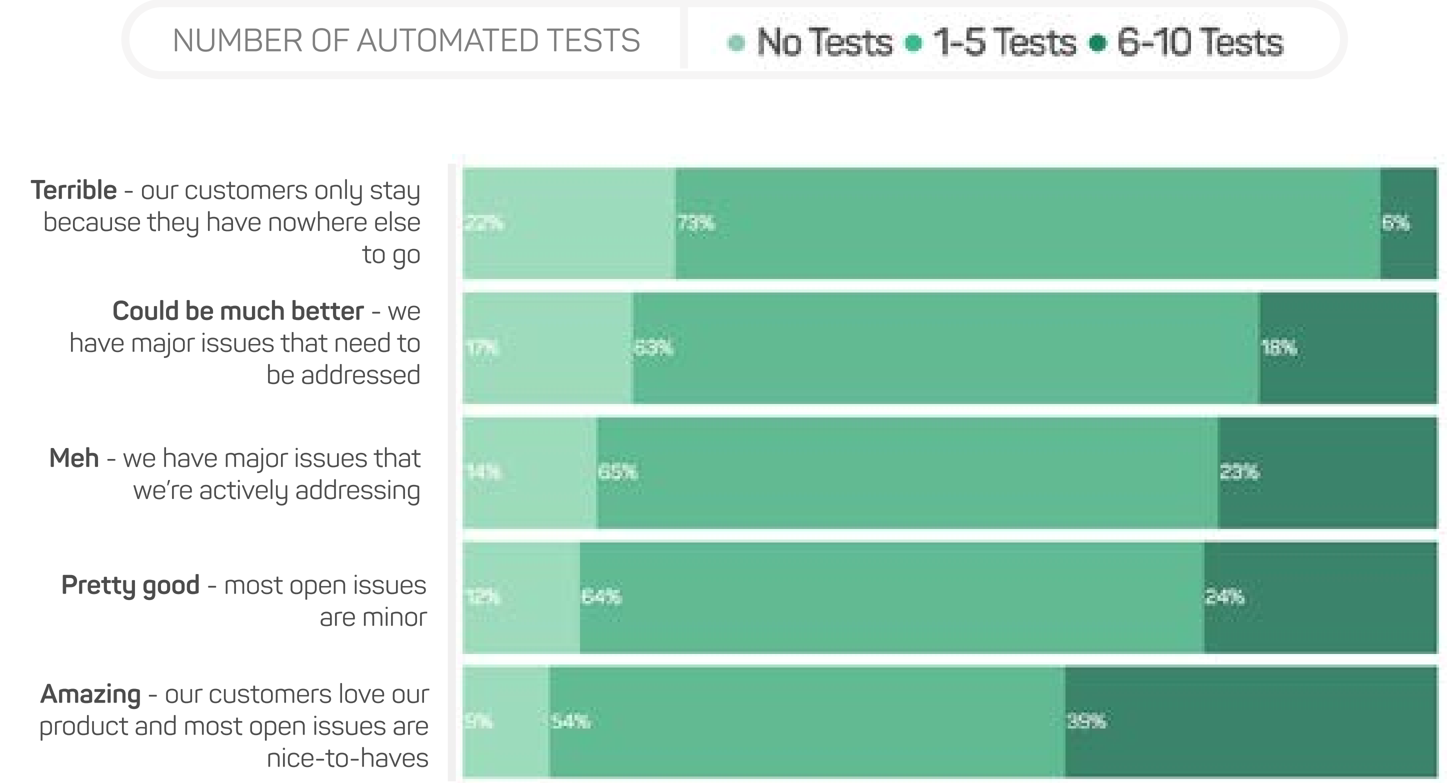


Many of the organizations who aren't adopting cloud most likely have a specific reason for not using it (medical or government verticals), but SaaS test automation vendors are paying ever closer attention to making sure that user data is secured, whether it's being SOC 2 compliant or providing secure tunneling to connect to internal environments. Apart from SaaS testing vendors, private cloud providers are also making cloud technology more accessible to organizations with the most demanding security requirements. It wouldn't be surprising if the number of non-cloud respondents continue to decline in the coming years.

We saw that cloud adopters had about 1.2 more automated tests than non-cloud adopters on average. But how much of a difference can 1 more test make? Below, we can clearly see a correlation between customer happiness levels and the number of tests that are automated in the CI/CD pipeline. A whopping 78% of respondents who reported having terrible customer happiness levels had 3 or fewer automated tests, while 60% of respondents with amazing customer happiness levels had 4 or more automated tests.

“What best describes your product’s customer happiness score, based on customer feedback?”

by Number of Tests Automated in CI/CD

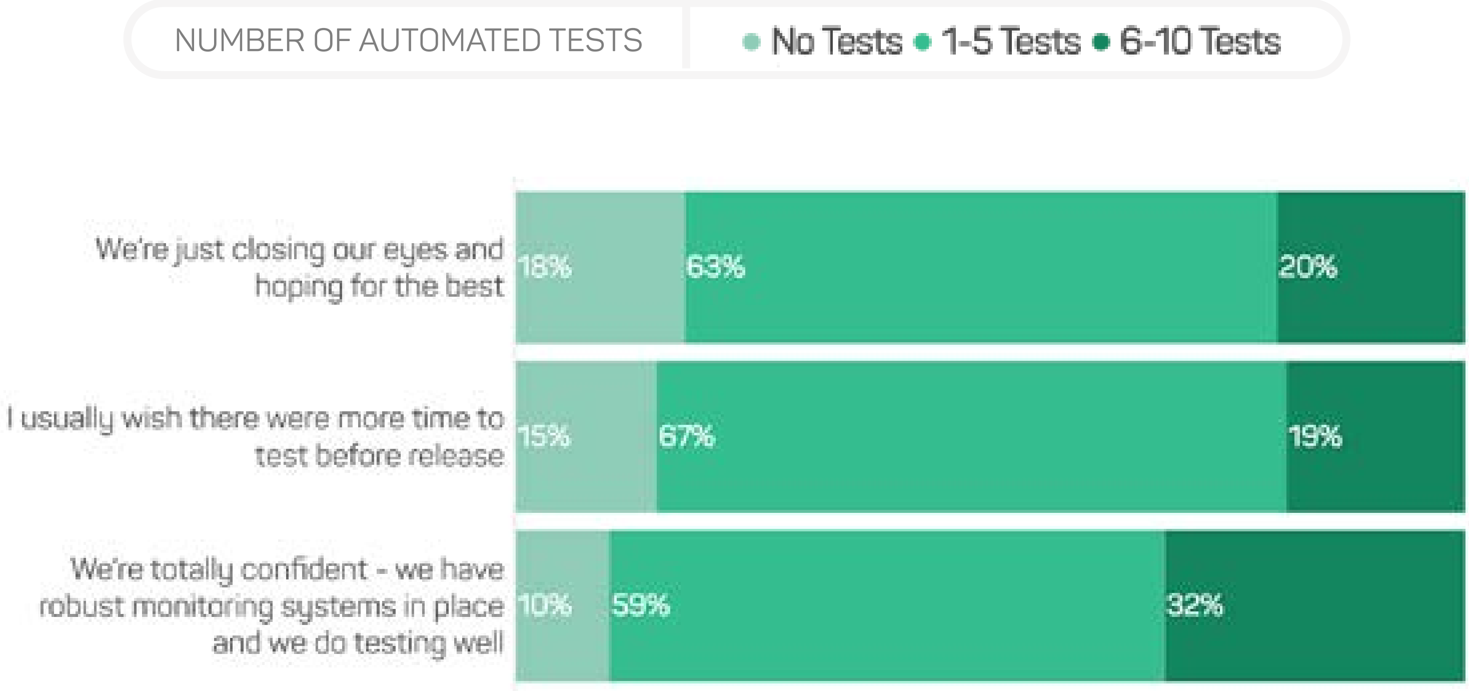


The same goes for developer experience, or DX (when we say DX, we're referring to the entire software team, not just developers). Though automated tests didn't make much of a difference in stress levels, more than half of respondents who felt totally confident on a release day had 4 or more automated tests. On the flip side, about two thirds of respondents who reported feeling like they were shooting in the dark had 3 or fewer automated tests.

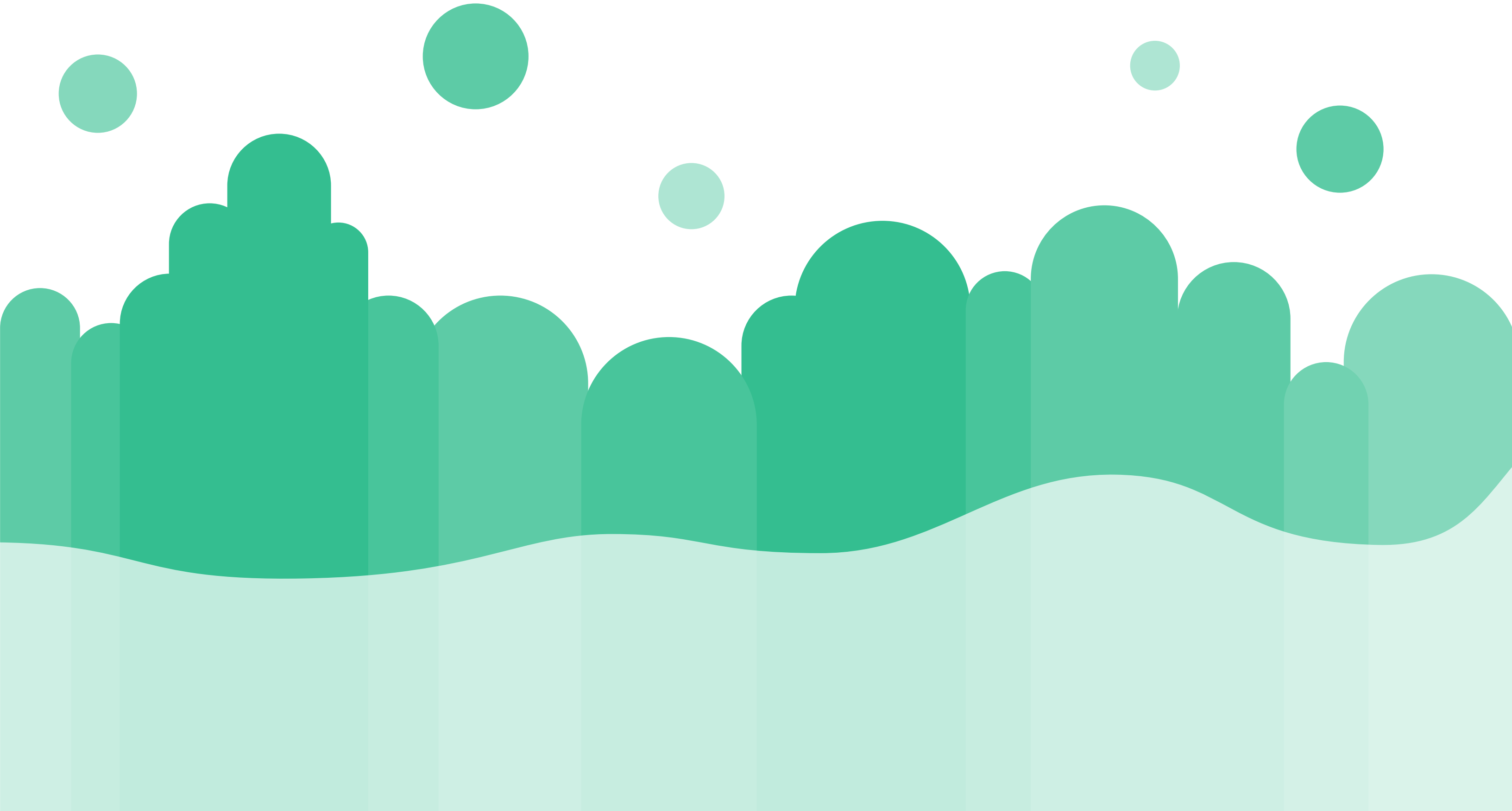
We'll dive deeper into how development processes affect the customer experience (CX) and DX later on in the report.

"How confident do you typically feel about each release?"

by Number of Tests Automated in CI/CD

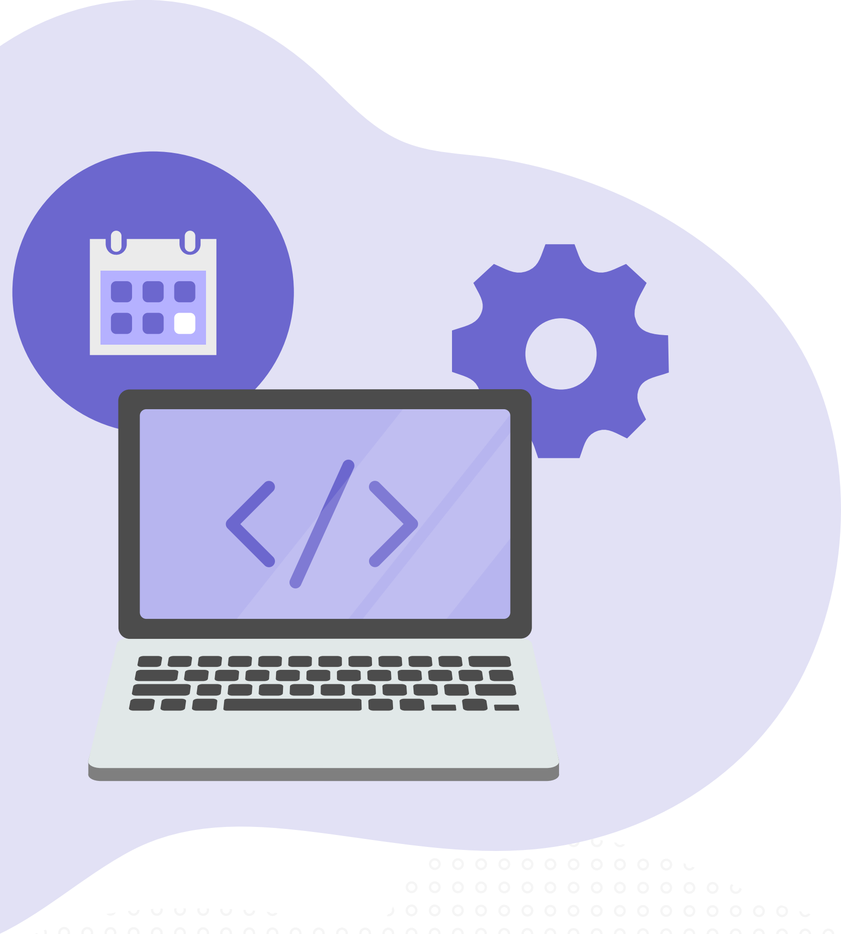


The source of pain that many developers face comes from getting stuck in support tickets and rework. It's clear that having a comprehensive automated testing suite is important to both the health of the applications and the team building them. When the software team is constantly reacting to fires, their morale drops and their effectiveness takes a huge toll. Having a finger on the pulse of the applications and the features they've just shipped lets them be more proactive to potential issues and move with more confidence.



DevOps Culture

Processes



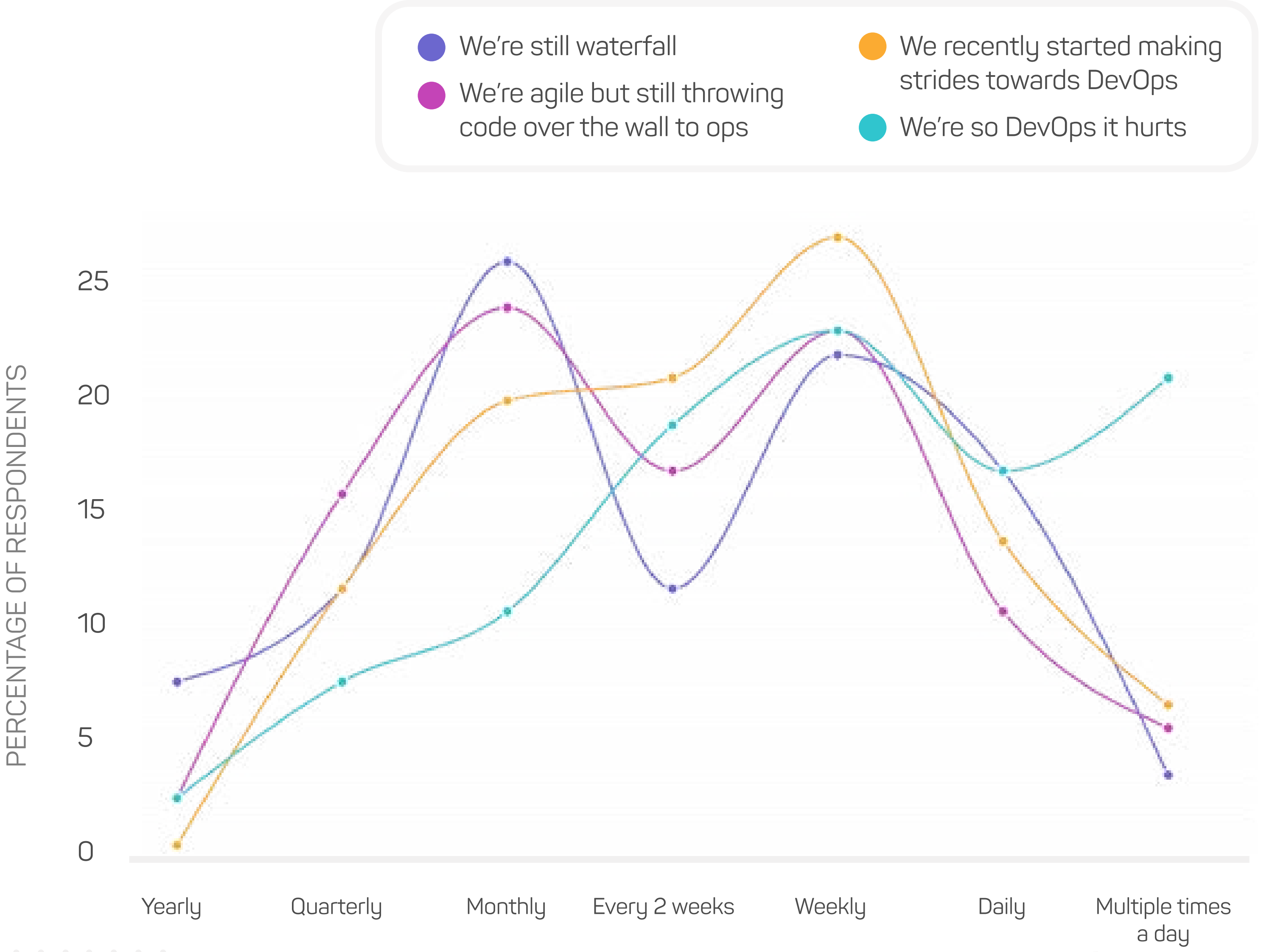
Development processes naturally vary from team to team, even more so as more modern and agile approaches to software development is adopted. We know that tech transformations don't happen overnight, so we segmented the respondents into four types of teams: Waterfall, Agile but still throwing code over the wall, emerging DevOps, and "so DevOps it hurts."

In this section, we'll map out the differences of software delivery practices and testing culture between DevOps-veteran teams and teams still following the waterfall methodology.

We'll start with release frequency. DevOps teams took a big lead in shipping multiple times a day. Most of the Emerging-DevOps respondents shipped weekly, and most of the laggards shipped on a monthly basis.

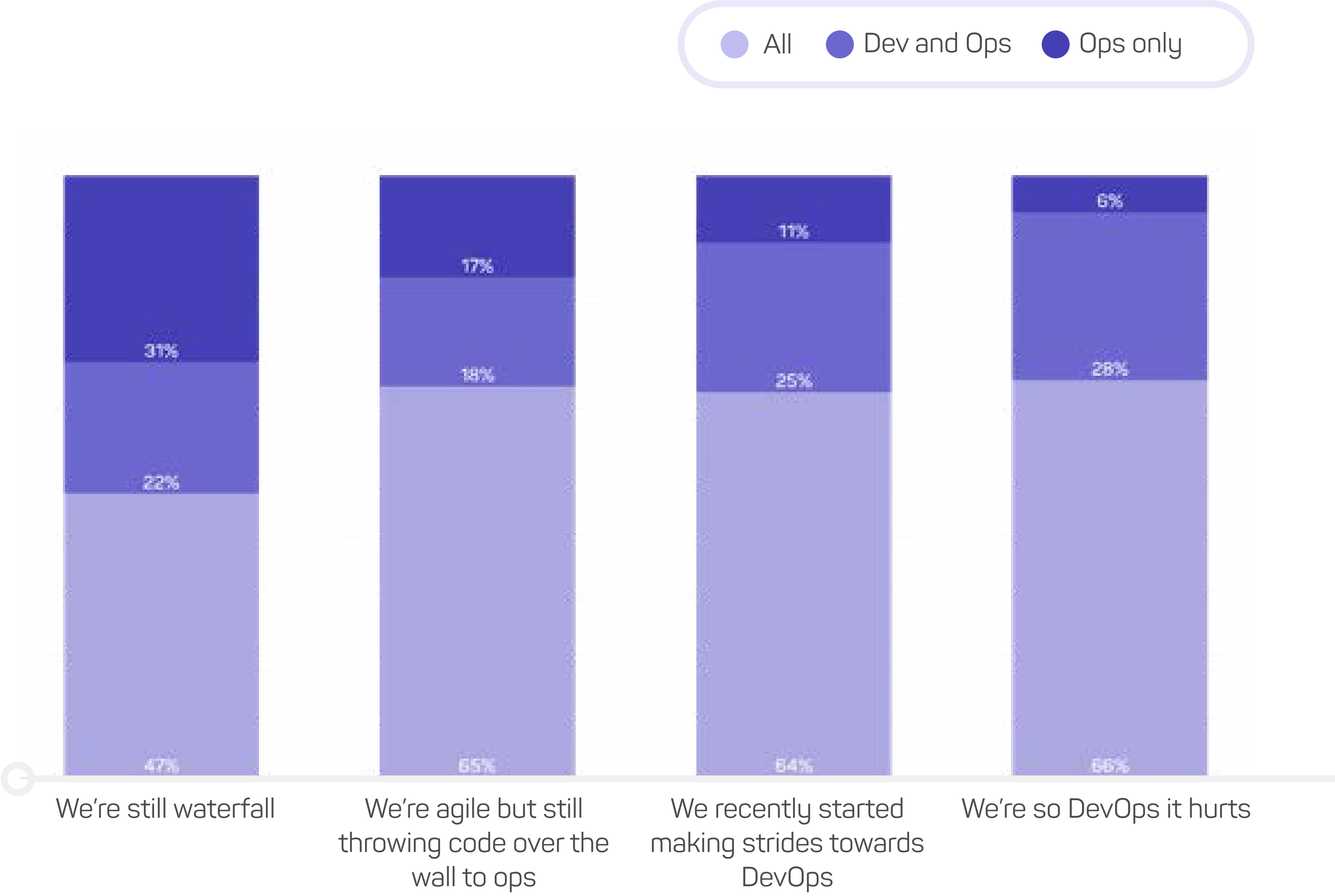
"What best describes how frequently your team deploys new code into production?"

By Team Type



“Who’s involved in a feature release?”

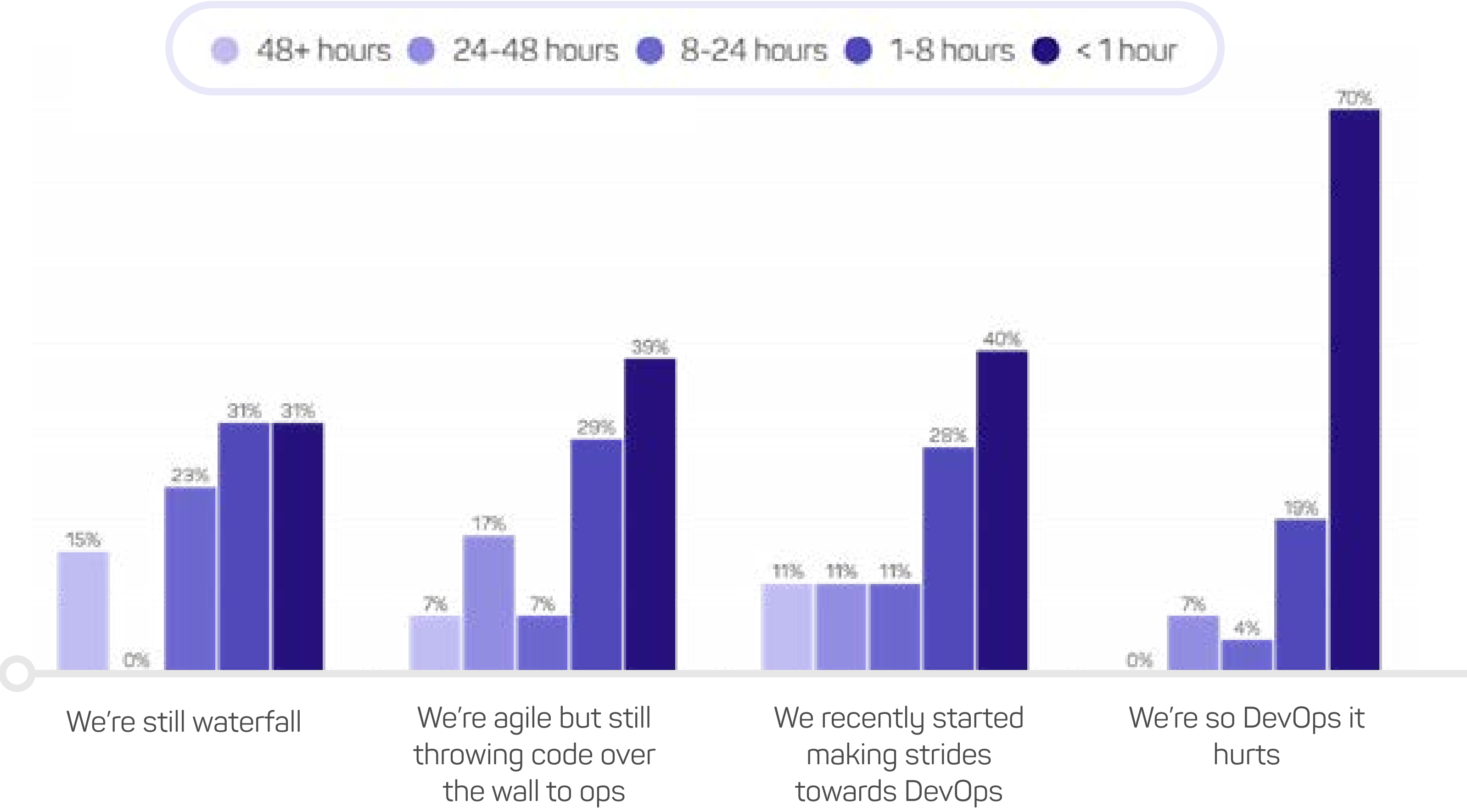
By Team Type



The waterfall methodology keeps silos between the different stages of application development. We can see that most of the waterfall respondents only involve the left-most roles in a feature release, operations. About two-thirds of every other type of team involve every role in a feature release - testers, product management, dev, and ops.

“How long does it take to roll back a bad release?”

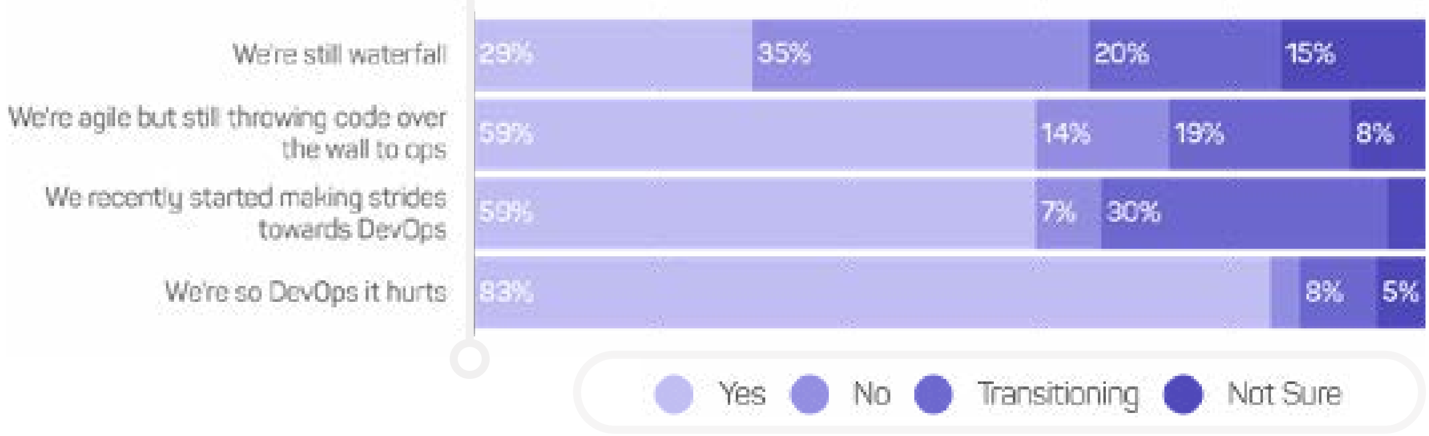
By Team Type



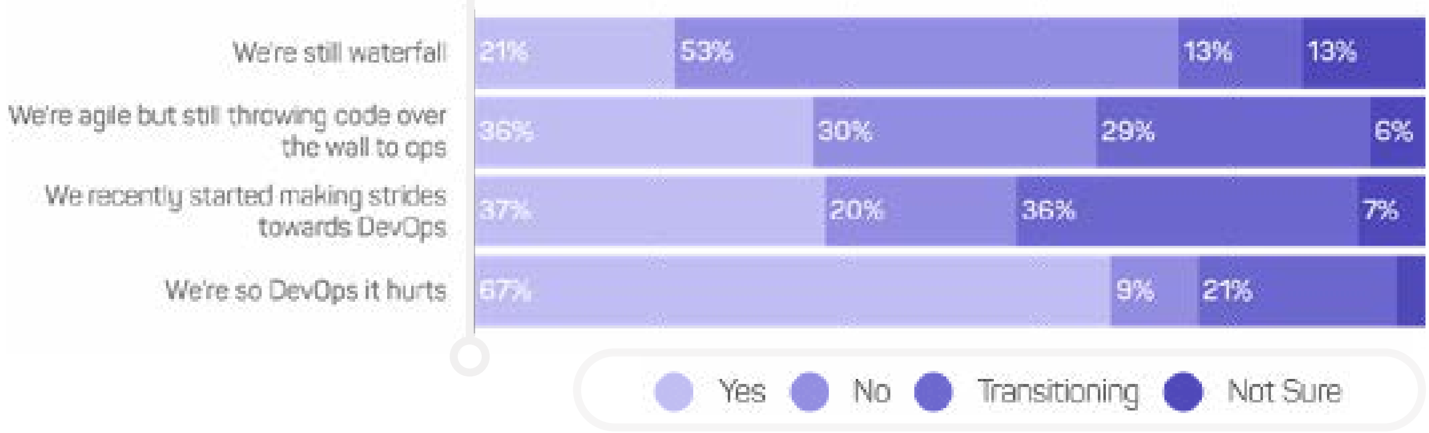
When it comes to rolling back a release, DevOps teams have an impressive lead over other teams, with 30% more respondents who are able to roll back releases in less than an hour. Also, 0% of DevOps respondents took longer than 48 hours to roll back a release.

DevOps teams are likely able to roll back releases so quickly because they have more of their environments automated in the CI/CD pipeline than other teams, pulling ahead 22% on average across all three CI/CD categories.

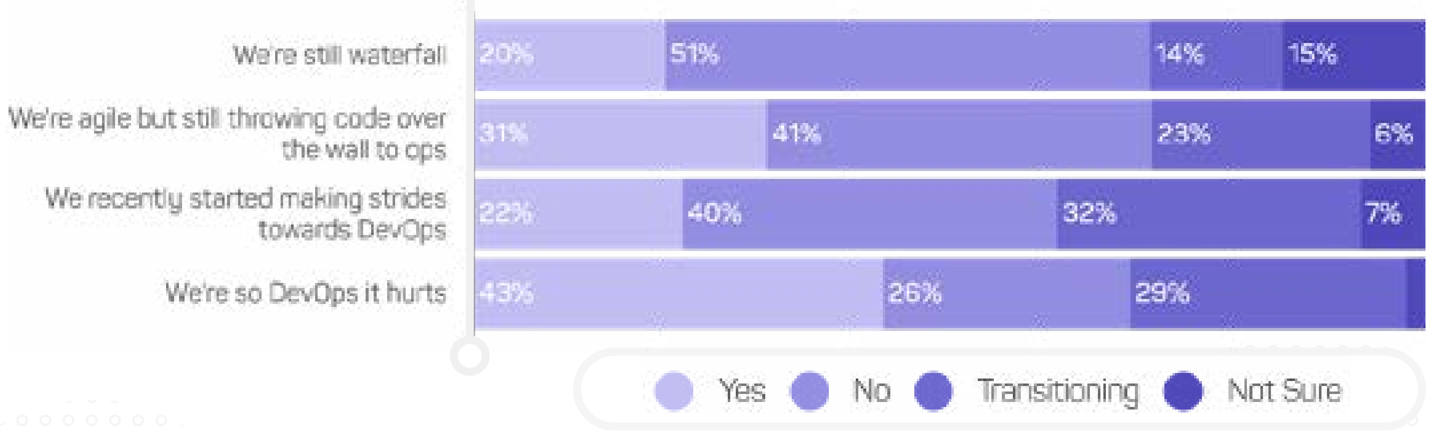
*“Has your organization adopted **Continuous Integration (CI)**?”*



*“Has your organization adopted **Continuous Delivery (CD)**?”*



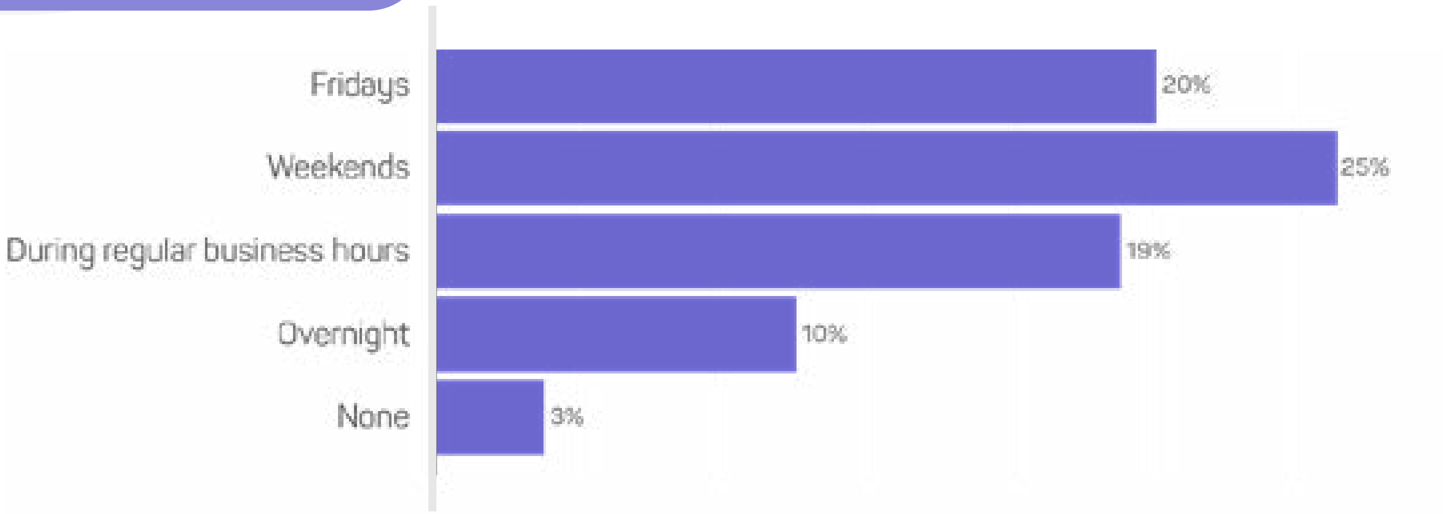
*“Has your organization adopted **Continuous Deployment (CD)**?”*



DevOps teams have more automation between development environments and more stability moving code between development, staging, and production. However, that doesn't change the fact that all respondents typically don't like to deploy on Fridays or weekends. DevOps teams do seem to hesitate slightly less when it comes to shipping during regular business hours though, and are more wary of shipping over weekends.

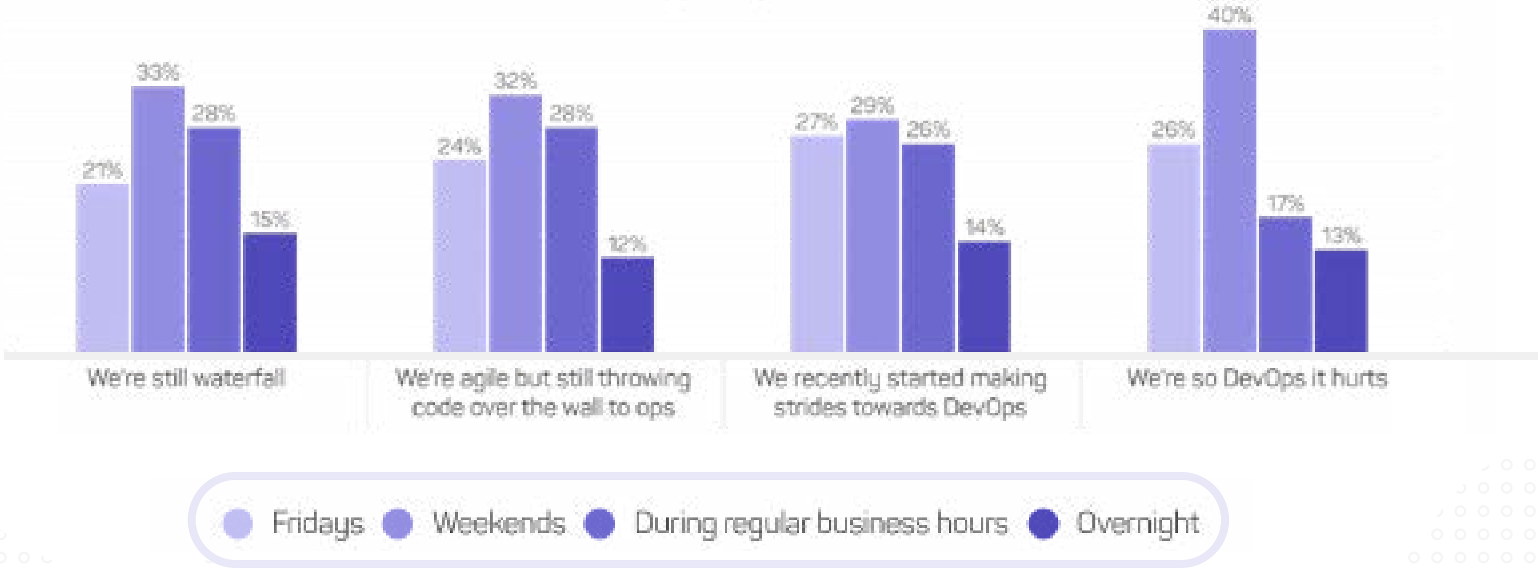
"Are there any times or days you avoid deploying? Select all that apply."

All Respondents



"Are there any times or days you avoid deploying?"

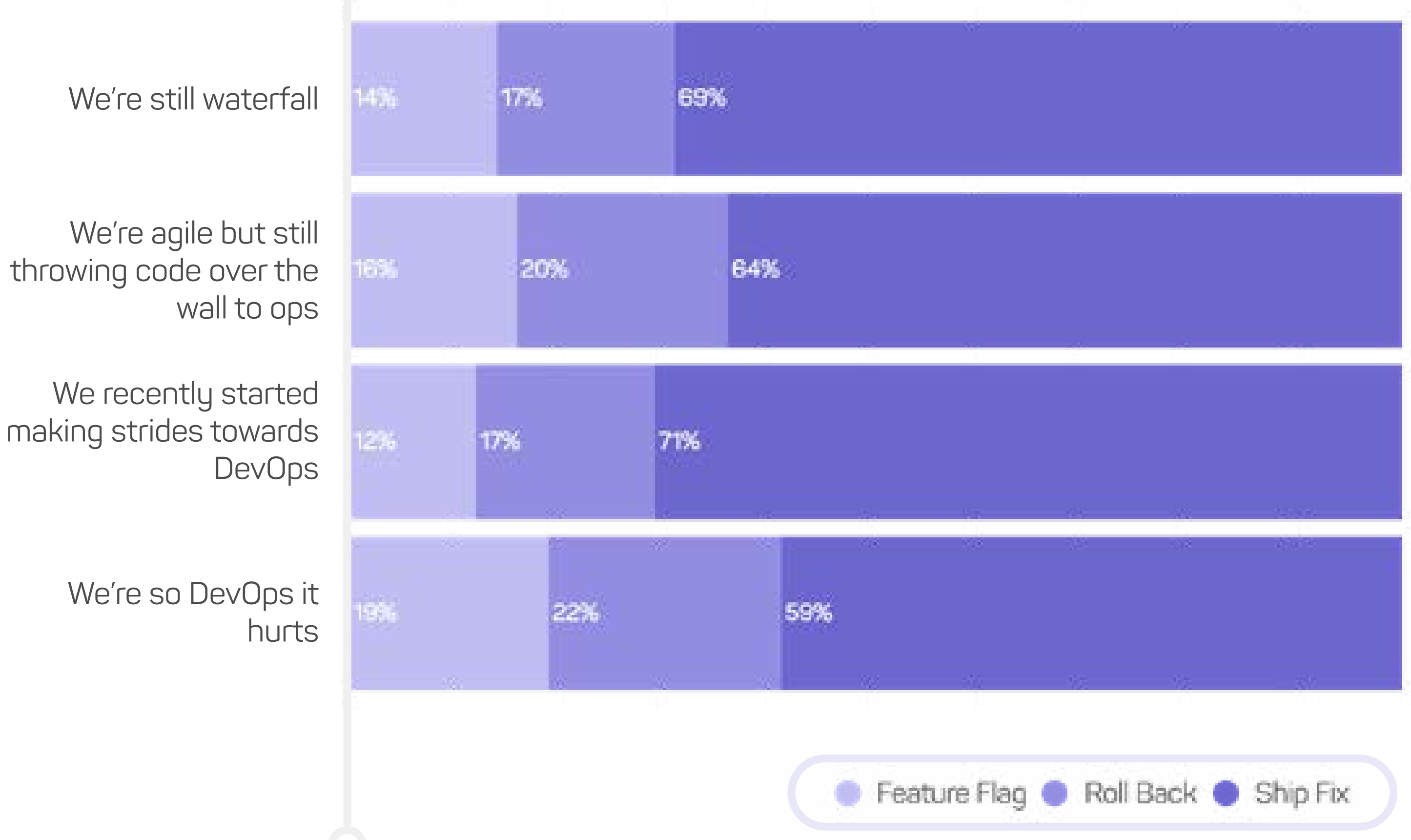
By Team Type



Bugs are inevitable, and teams may respond to bugs differently depending on their nature. But we wanted to know, on the whole, how bugs are most typically addressed. The majority of all teams ship a fix for the bug. Less common methods are rolling back the release or shipping under a feature flag and testing the feature with a smaller subset of users before making it generally available.

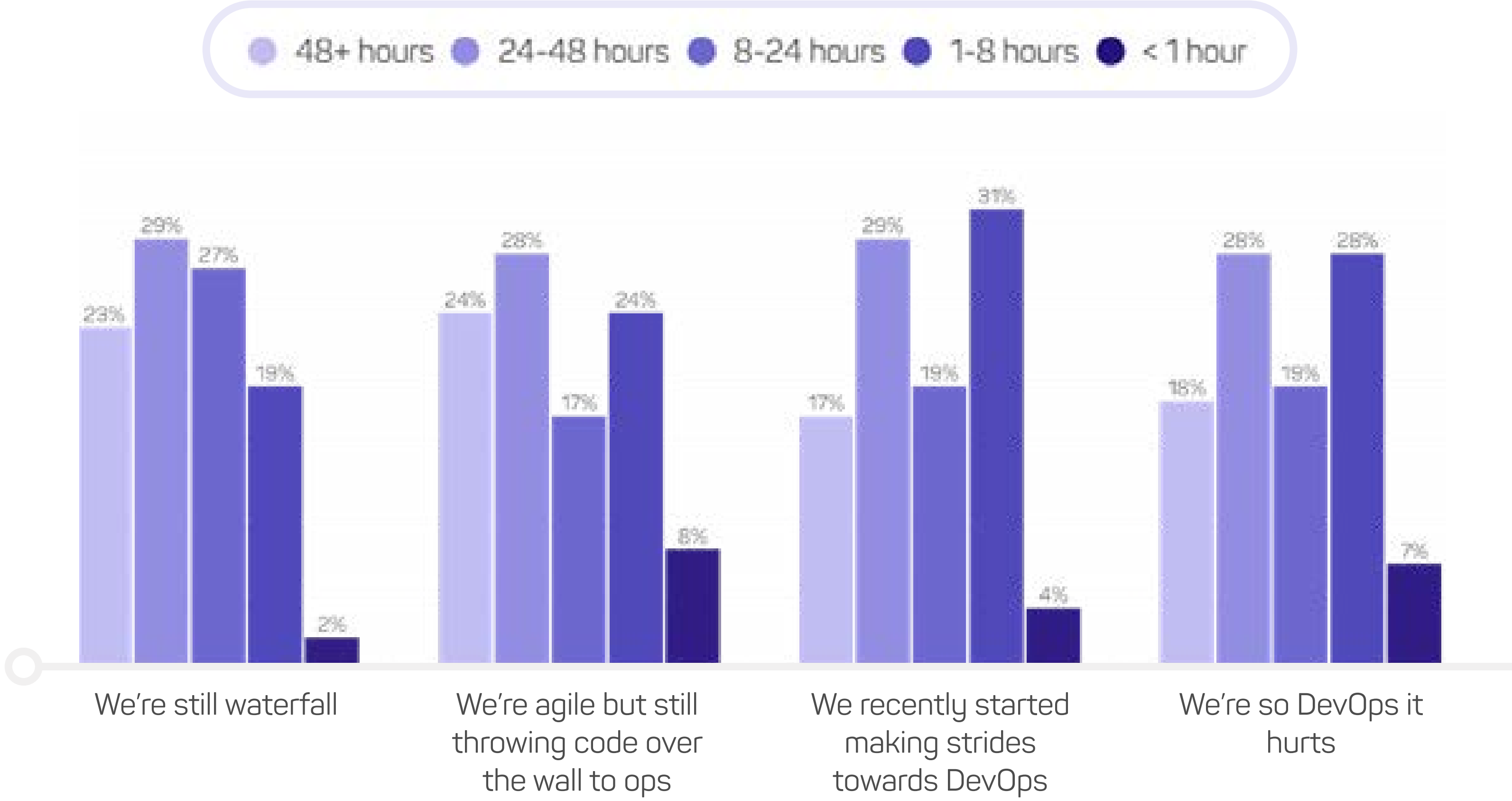
"How do you typically address bugs in production?"

By Team Type



“How much time on average elapses from when a critical bug is detected in production to when a fix is deployed?”

By Team Type



Mean time to repair (MTTR) is a particularly important metric to look in DevOps, as it can be an important factor towards calculating ROI, especially on DevOps teams. DevOps and emerging DevOps teams both largely fix critical bugs within a 24-48 hour window or within 1-8 hours. Waterfall teams are more likely to fix critical bugs within 24-48 hours, while agile teams are somewhere in the middle between waterfall and DevOps, mostly fixing bugs in the 24-48 hour window but with an even chance of fixing bugs either after 48 hours or within 1-8 hours.

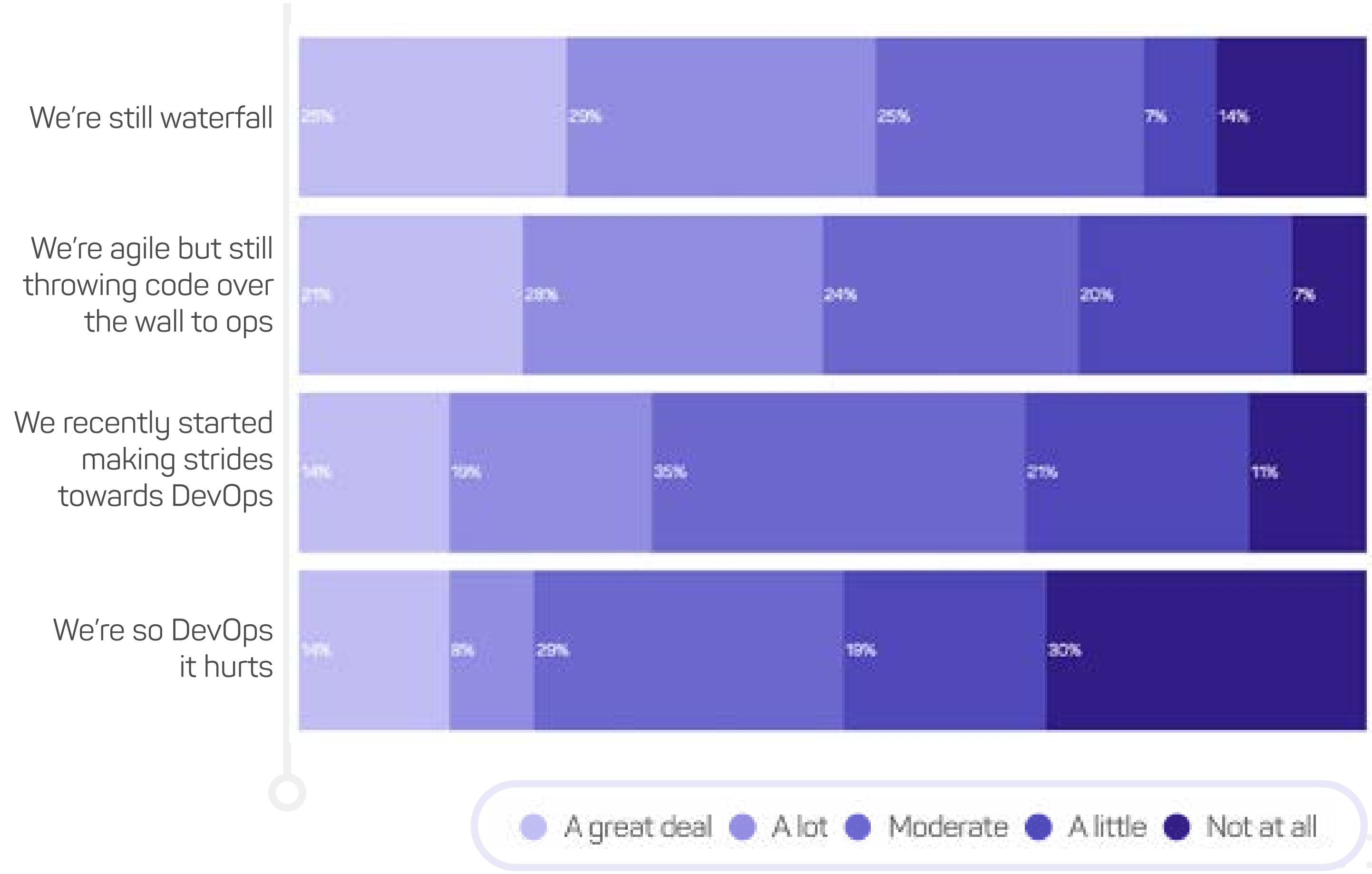
Developer Experience

DevOps teams differentiate themselves from the pack with their rapid release frequency, fully automated CI/CD pipelines, and ability to demote releases quickly. We'll now look at how these differentiators make an impact on the developer (DX) and customer experience (CX).

About 27% more DevOps respondents than waterfall respondents feel little to no stress at all on a release days, which makes sense as many DevOps teams are shipping daily or multiple times a day. Conversely, 32% more waterfall teams than DevOps respondents feel a lot to a great deal of stress on release day.

“How stressful are release days, typically (in terms of production fires, last-minute testing, addressing major issues, etc)?”

By Team Type

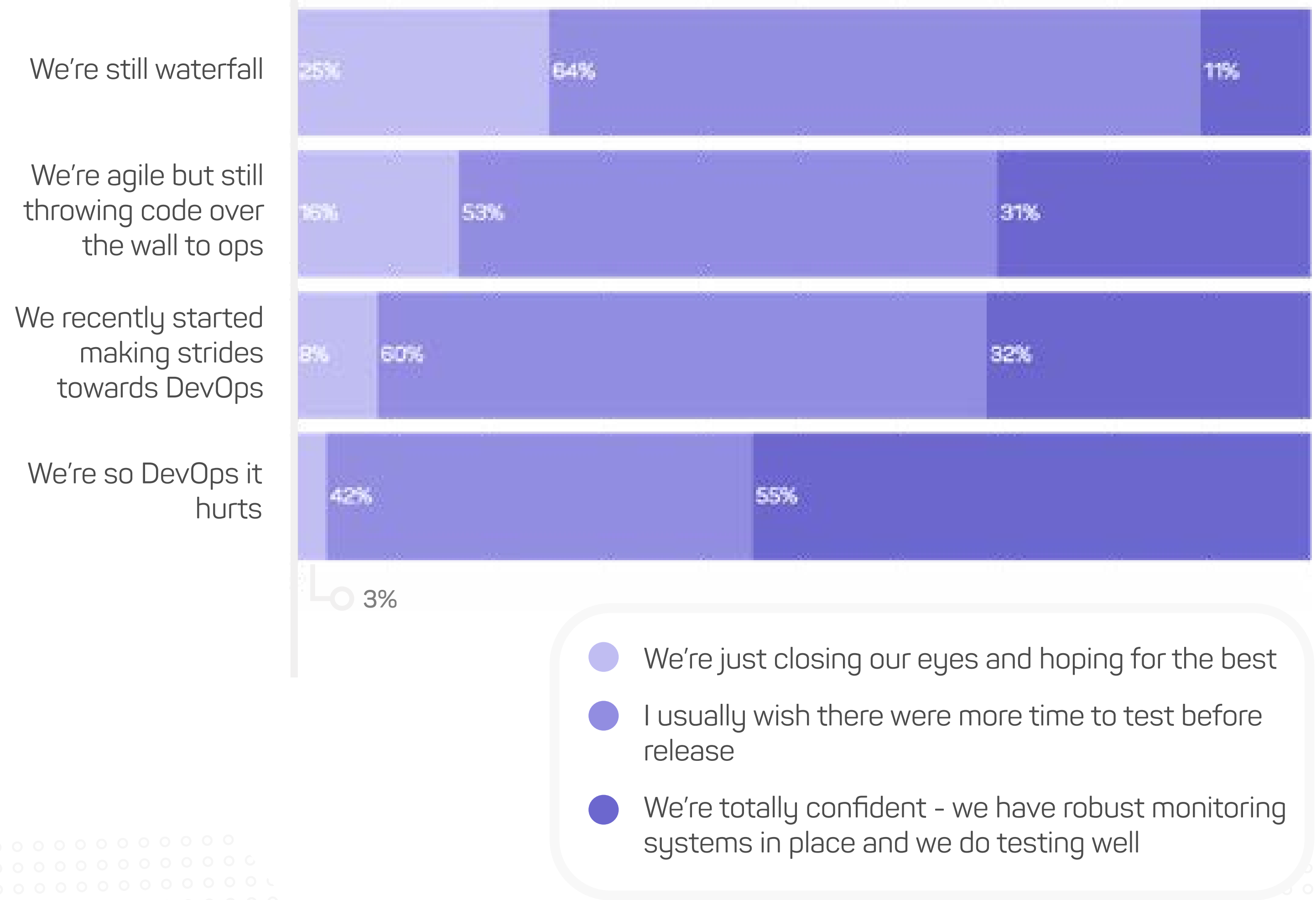


Frequent release cycles and the agility of which code can move between stages of development not only affects stress levels, but confidence levels. DevOps teams are a testament to this with 30% more respondents, on average, being totally confident about each release.

The “fail fast” philosophy supports incremental development and embraces the ability to pivot as soon as a team determines that something isn’t working. It also breeds a culture where it’s more important to get to the root cause of a problem than it is to find somewhere to place blame, stopping the negative consequences of “witch hunt” post mortems. These factors appear to contribute to happier teams - teams that are more confident in their work and are less stressed.

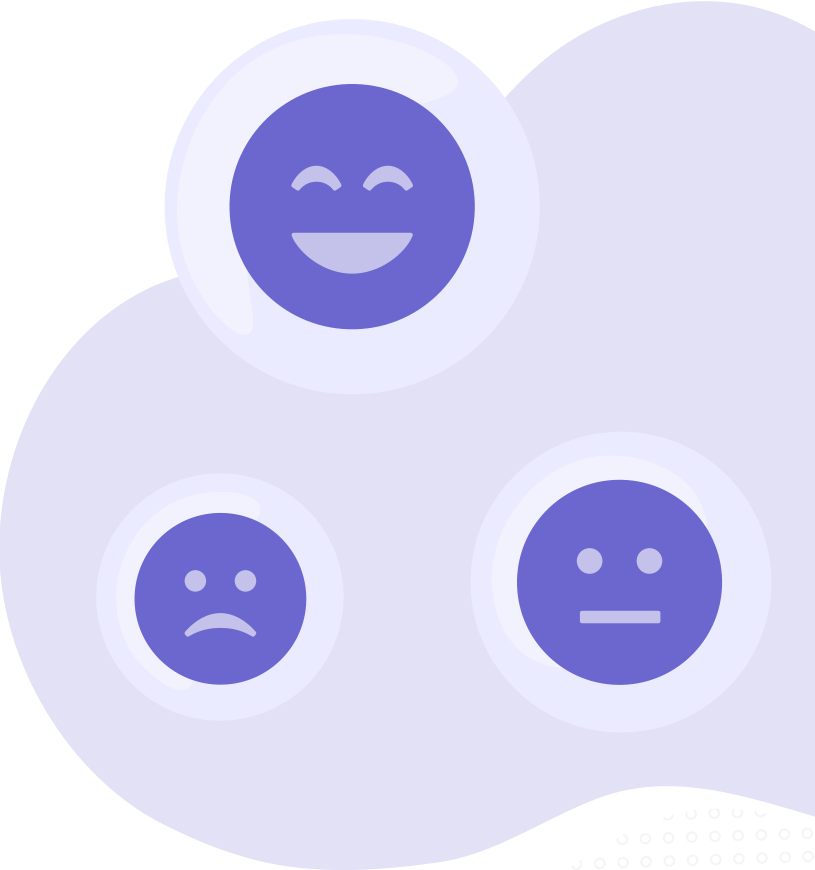
“How confident do you typically feel about each release?”

By Team Type



DevOps Culture

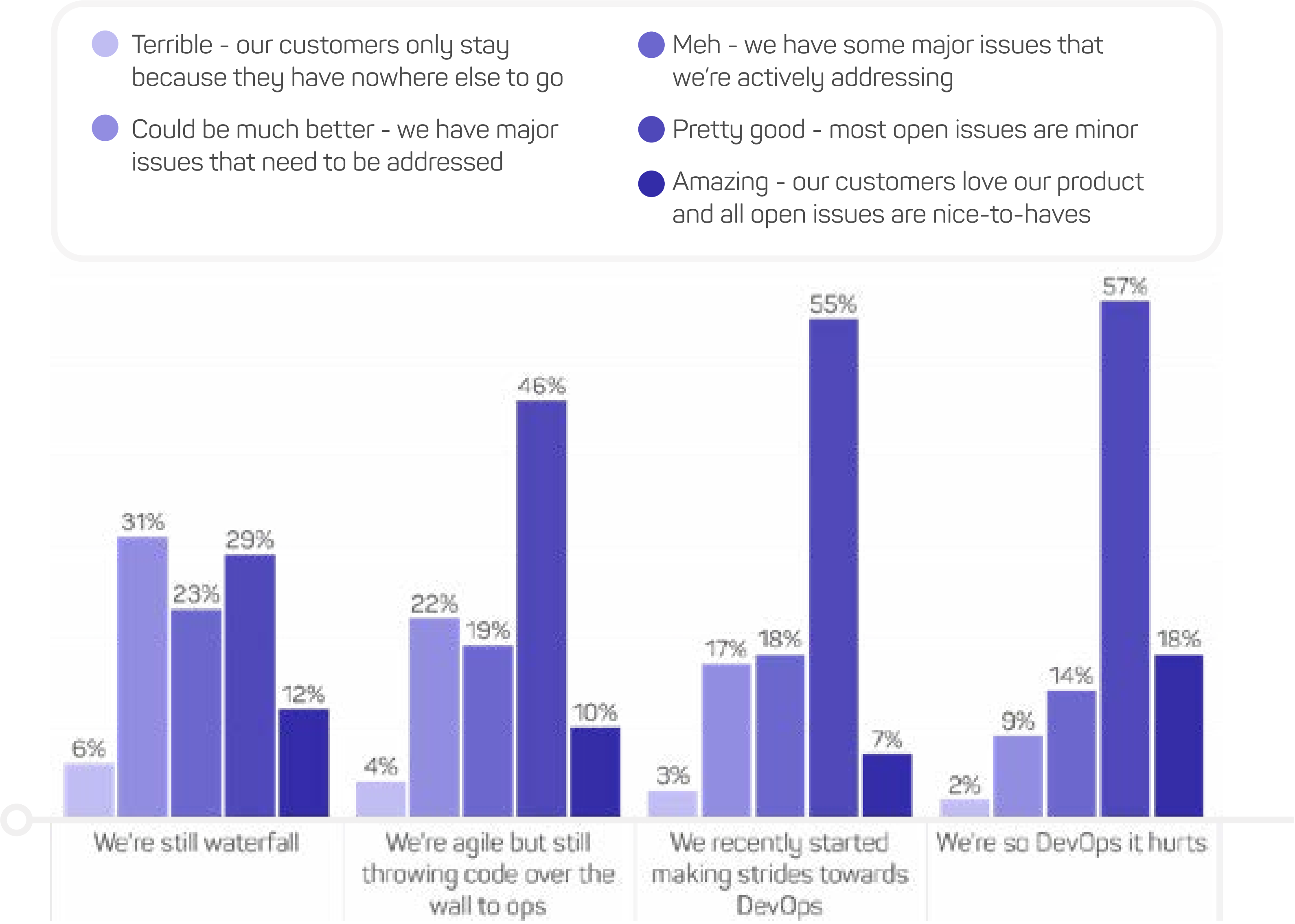
Customer Experience



The data in the section above clearly shows that DevOps improves DX, but the data in this section reveals that customer experience (CX) is surprisingly unaffected by DevOps. Agile, emerging DevOps, and DevOps veteran teams all have "pretty good" customer happiness levels. This section will therefore answer the question: What do the teams with "amazing" customer happiness levels have in common if it's not dependent on how mature their development process are?

“What describes your customer happiness levels, based on feedback from your customers?”

By Team Type

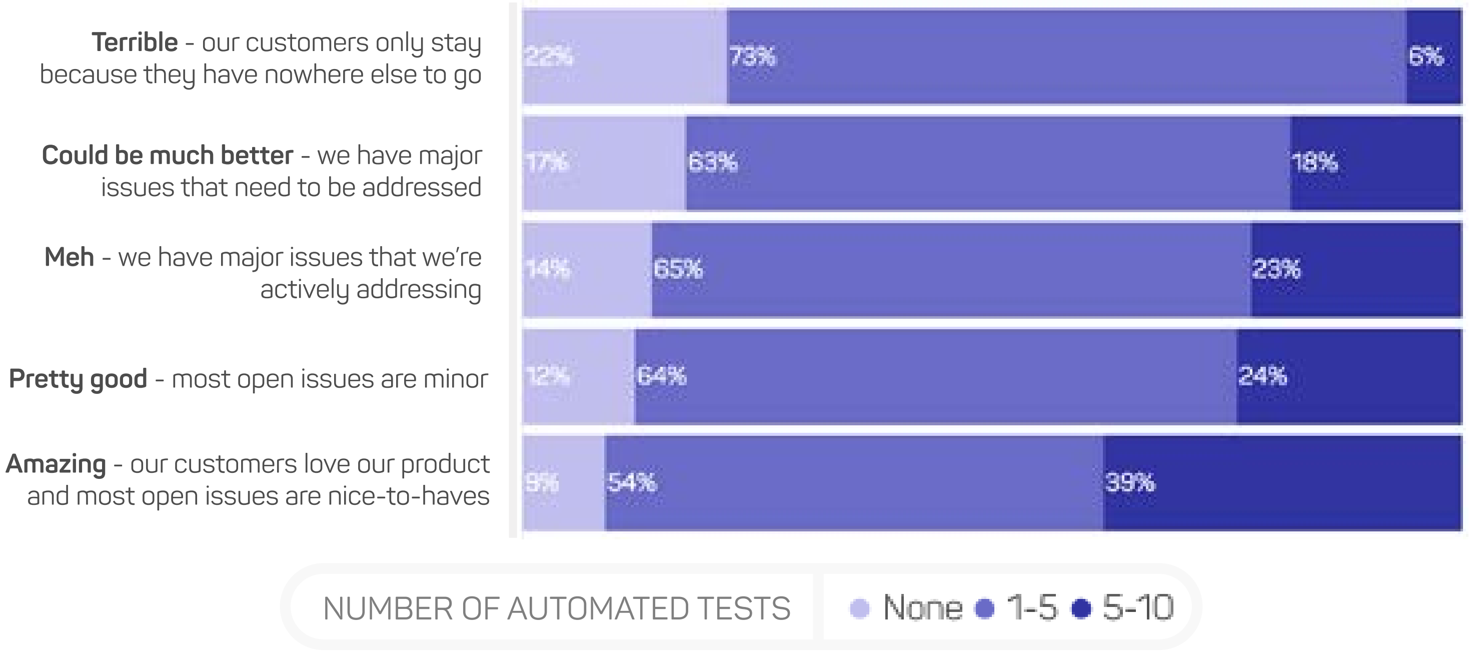


All team types except for waterfall teams settled into the “pretty good” customer happiness level category. DevOps teams do pull slightly ahead in the “pretty good” to “amazing” customer happiness levels than the rest of the teams. But truthfully, we were surprised they didn’t have a more distinctive lead. DevOps brings a lot of technical benefits to the team, but for the cost of tech transformations, there’s a minimal business benefit here.

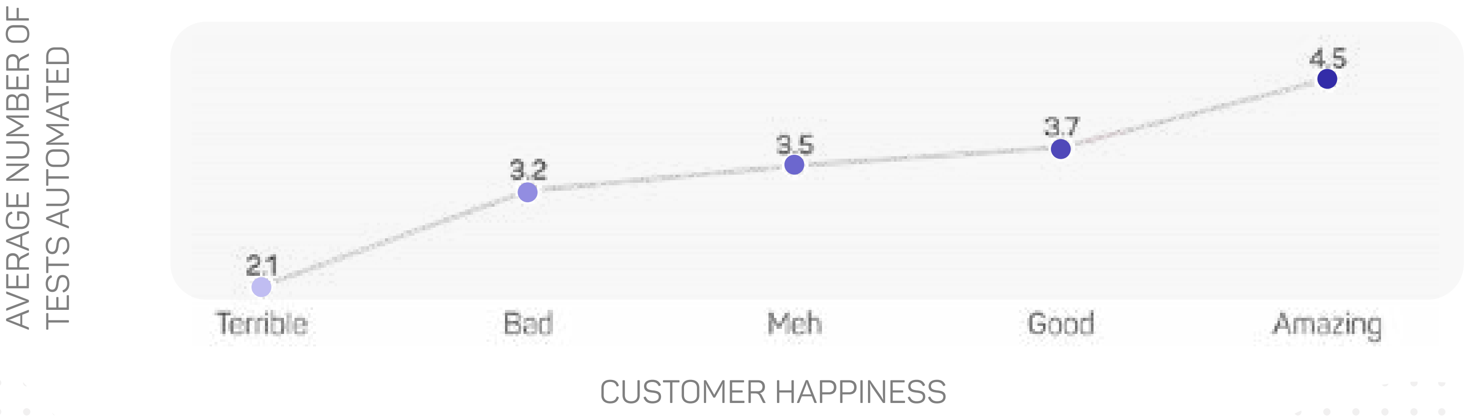
Digging into the data to find the secret sauce to “amazing” customer happiness scores brought us back to testing culture and QA processes. As we noted earlier in the report, automated testing has a high correlation with customer happiness scores. The respondents with “amazing” customer happiness scores automated on average about 5 tests in the CI/CD pipeline, with “pretty good” customer happiness teams coming in at 4, and the number continued to decline until teams with “terrible” customer happiness scores had only 2 tests on average in the CI/CD pipeline.

“What best describes your product’s customer happiness score, based on feedback from your customers?”

By Number of Tests Automated in Ci/CD



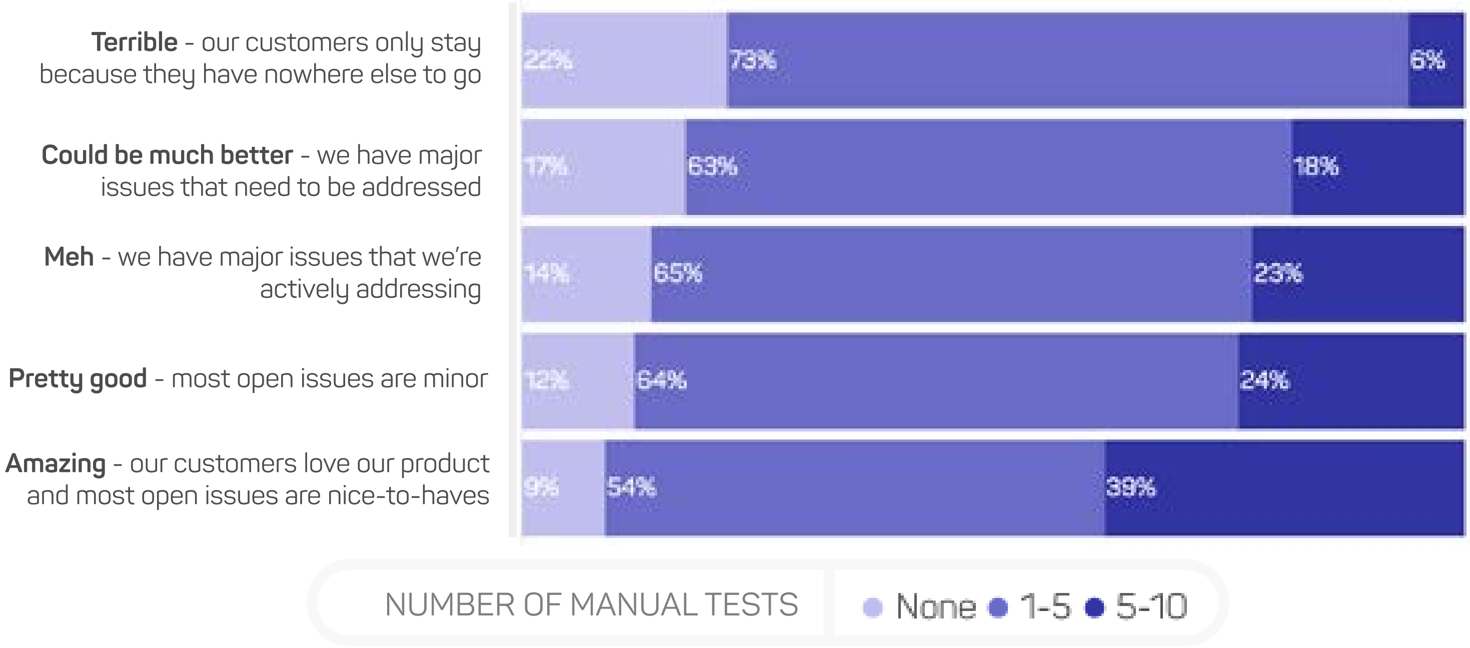
*By **Average** Number of Tests Automated in CI/CD*



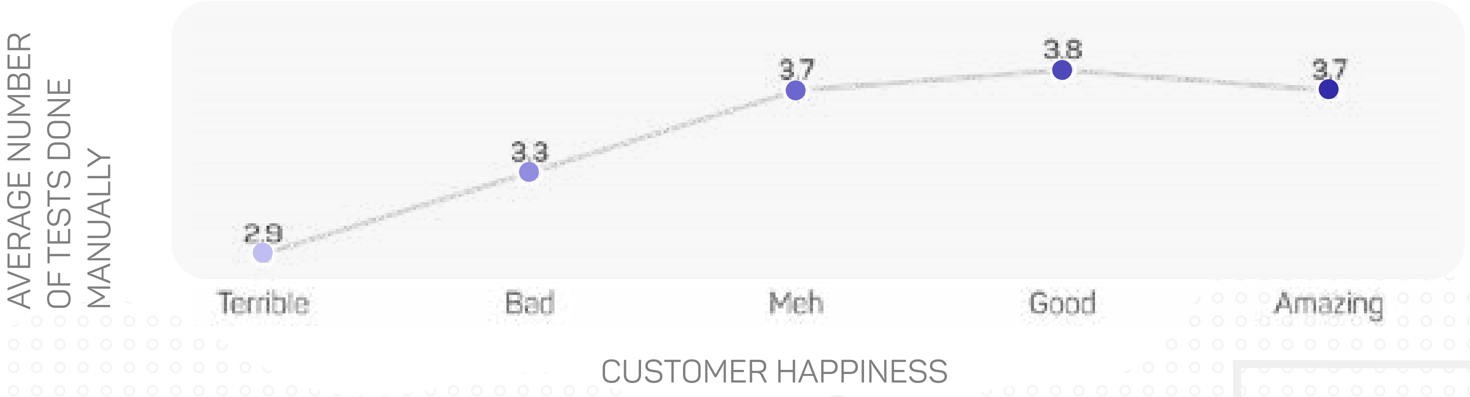
What if teams are still testing, but doing it manually? Teams with "amazing," "pretty good", and "meh" customer happiness levels all conduct on average 4 different types of tests manually, so manual testing doesn't seem to have as much of an impact on CX as automation does, which is a big deal, given how much more effort (in terms of time and people) it takes to test manually.

"What best describes your product's customer happiness score, based on feedback from your customers?"

By Number of Tests Done Manually



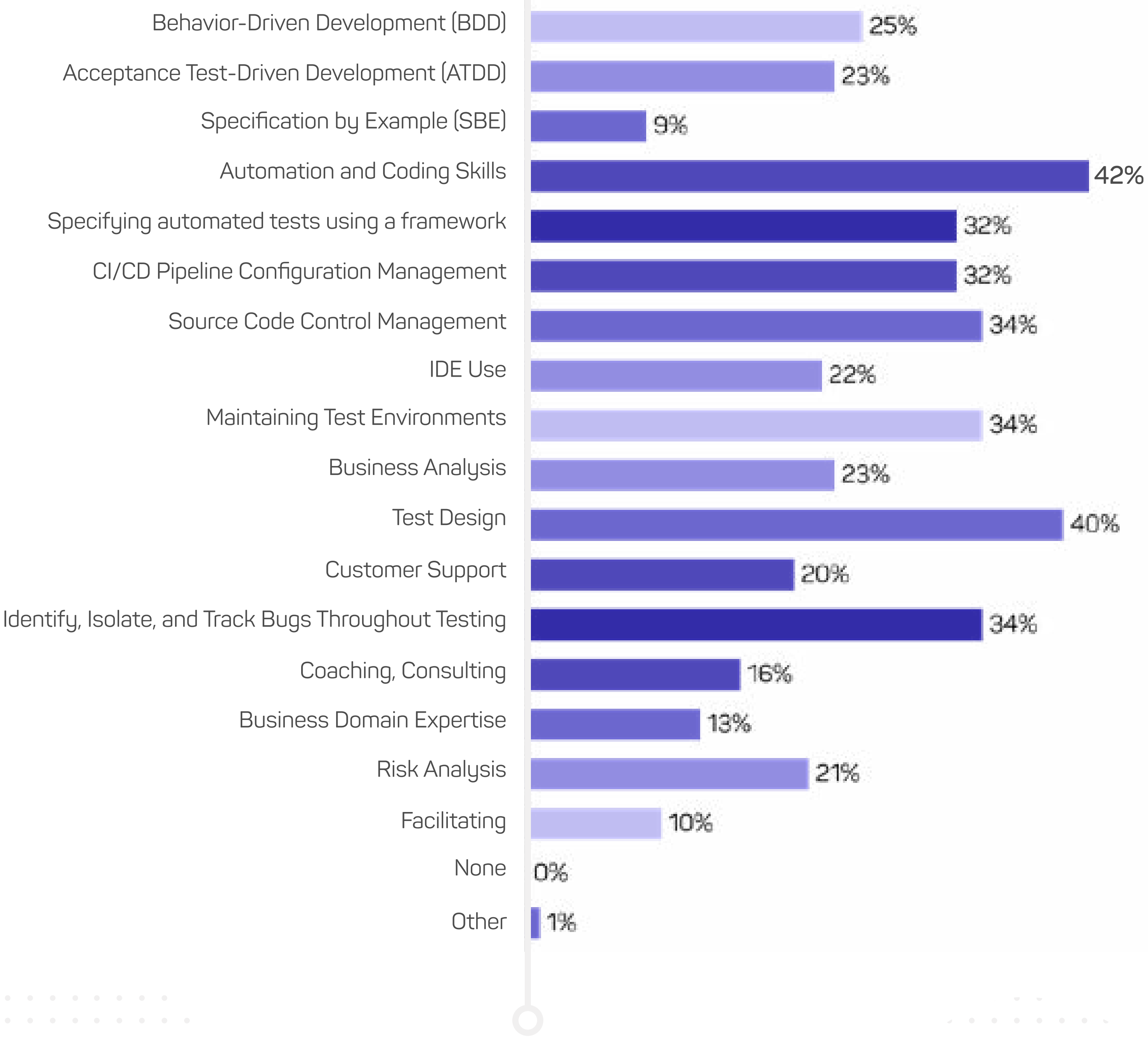
*By **Average** Number of Tests Done Manually*



We also looked at how customer happiness correlates with QA activities. In this survey, QA activities are distinct from testing activities because QA activities are less focused on the tactics and more on the strategy, helping to shape the purpose of the tests themselves.

“What QA activities are done in your organization?

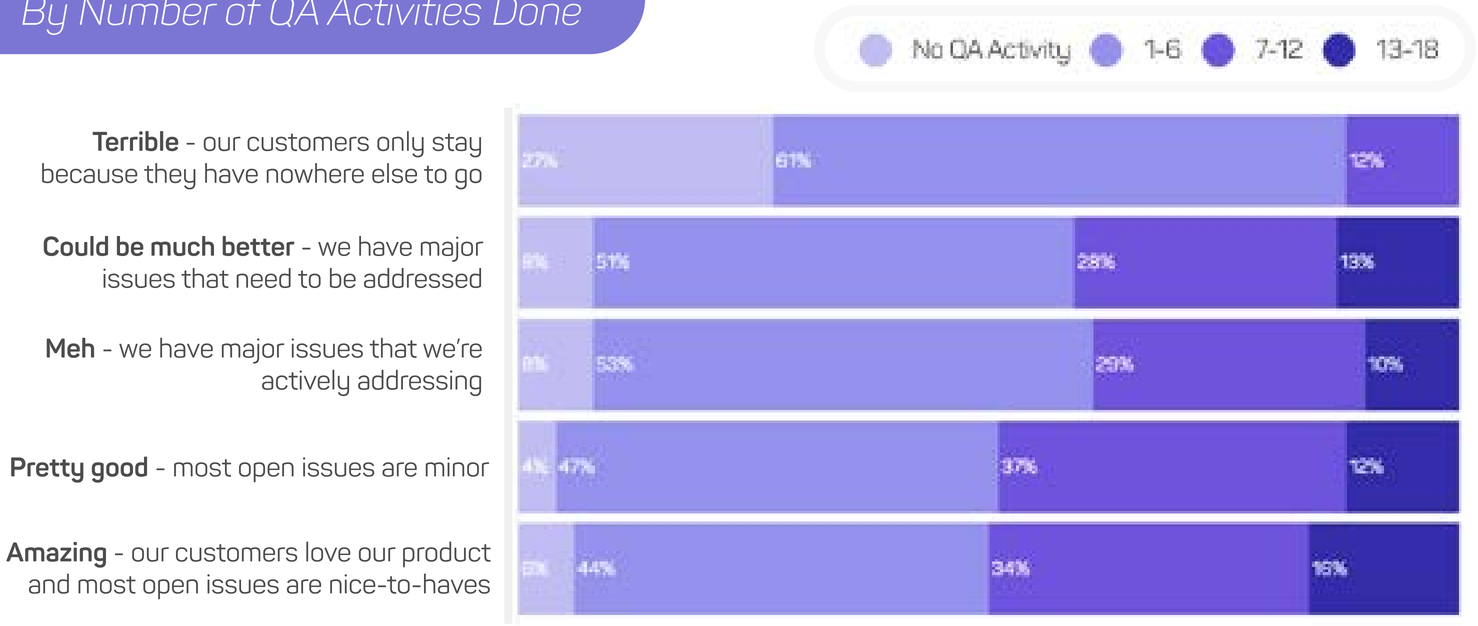
Select all that apply.”



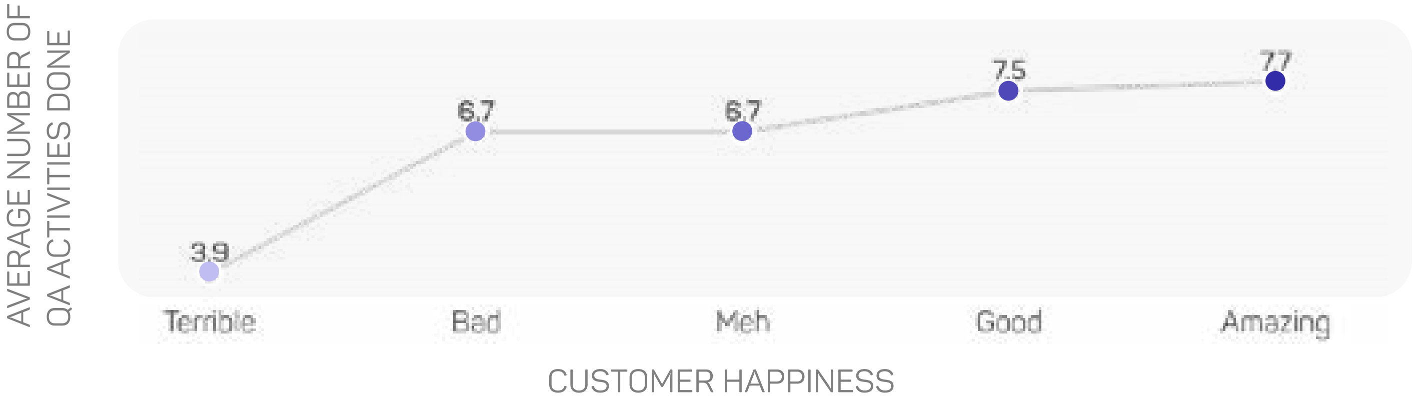
When we looked at the number of QA activities conducted, we did see that teams with "amazing" customer happiness levels conducted on average double the amount of tests done by those with "terrible" scores, with 8 activities as opposed to only 4.

"What best describes your product's customer happiness score, based on feedback from your customers?"

By Number of QA Activities Done



*By **Average** Number of QA Activities Done*



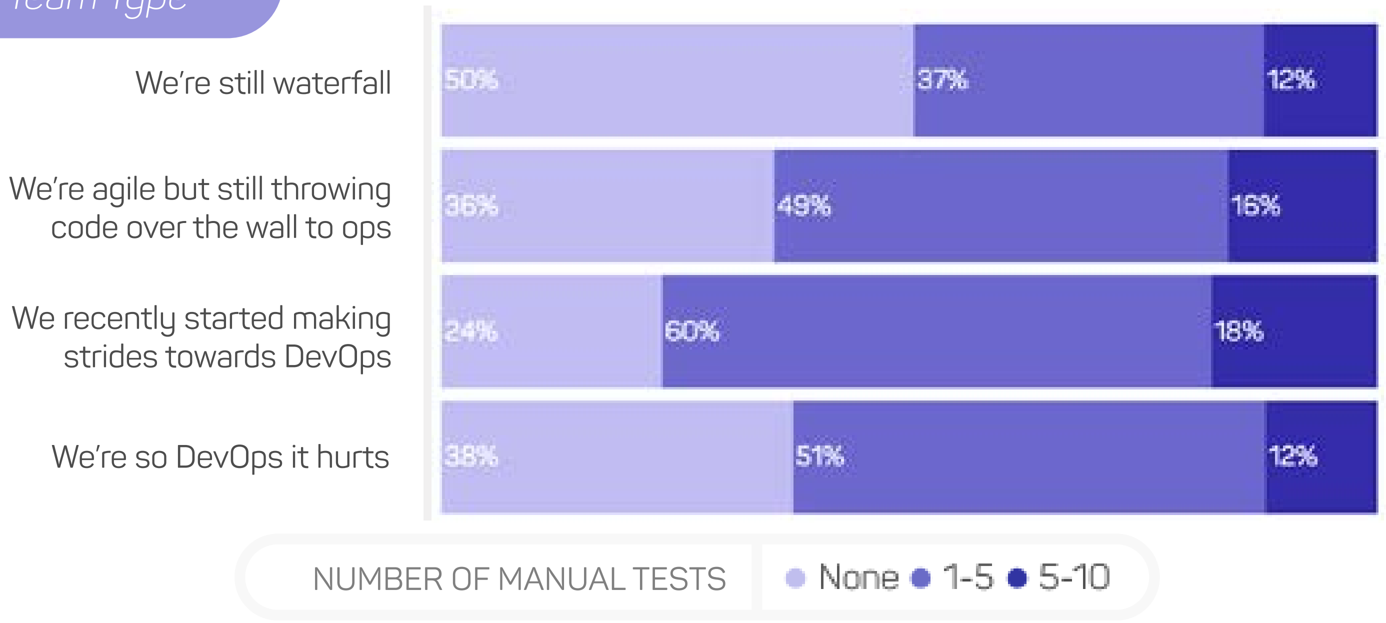
The fact that automated testing and QA activities would have a strong relationship here makes sense. Standing up an automated testing suite can be a lot of work and therefore requires a lot more thought before implementation. On the other hand, manual testing is oftentimes just done as an afterthought or as a last-minute measure before deployment. Manual testing, when not done by a dedicated, specialized QA role, can end up being untargeted and ineffective.

To continue what we've been doing in our analysis, let's split up the testing activities (manual testing, automated testing, and QA activities) as done by types of teams.

Interestingly, veteran DevOps teams conducted 1 less manual test than emerging DevOps teams - about the same amount of manual testing as agile teams. This is surprising - it seems that teams moving towards DevOps start off well in their testing efforts, but those efforts start to fall off the more frequently they ship.

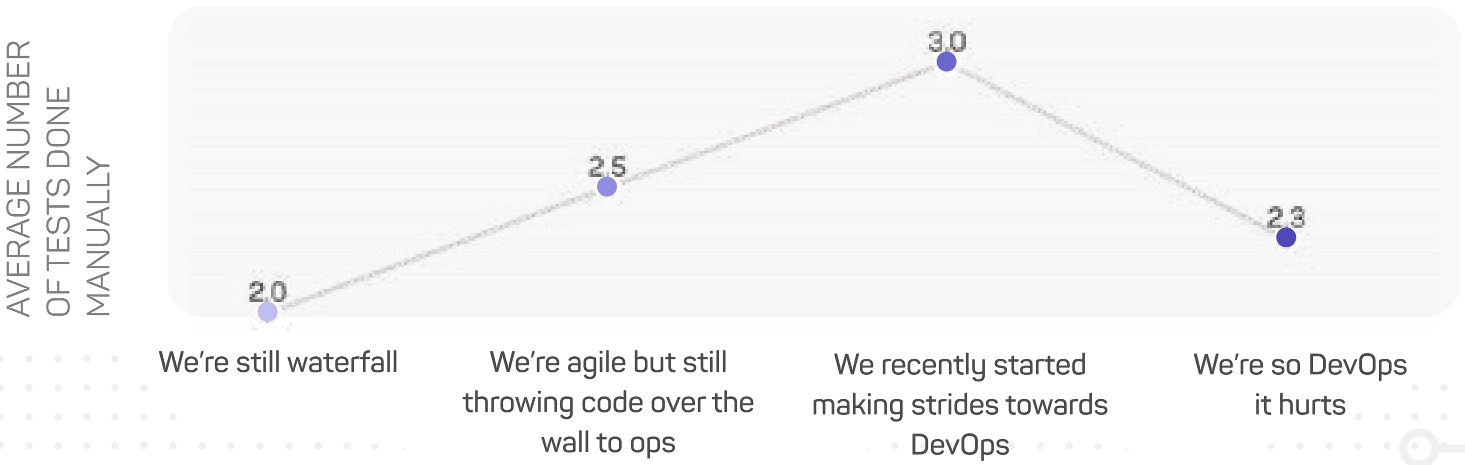
Number of Tests Done Manually

By Team Type



***Average** Number of Tests Done Manually*

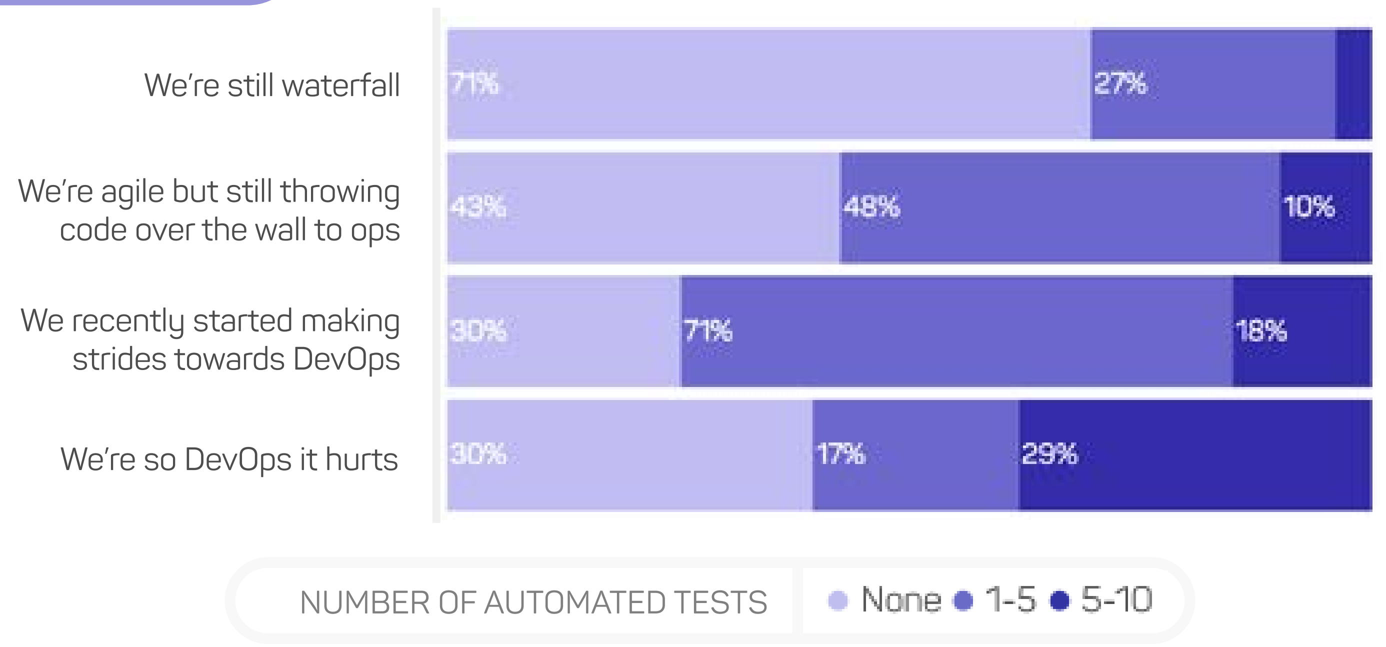
By Team Type



On the automation front, DevOps and emerging DevOps teams automated an average of 1 more test than agile teams and 2 more tests than waterfall teams.

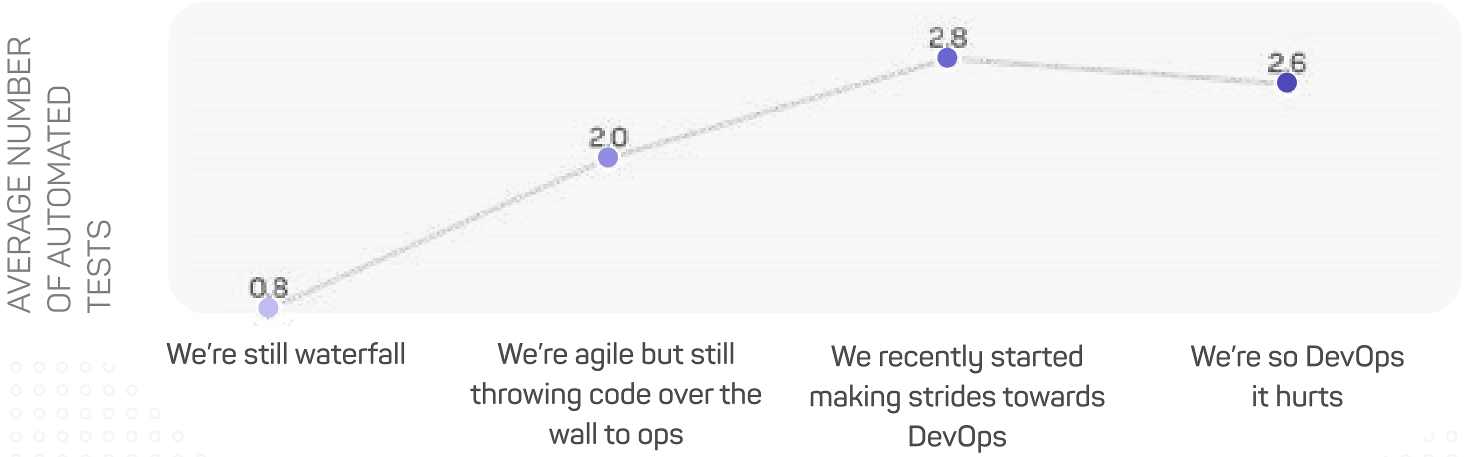
Number of Tests Automated in CI/CD

By Team Type



Average Number of Tests Automated in CI/CD

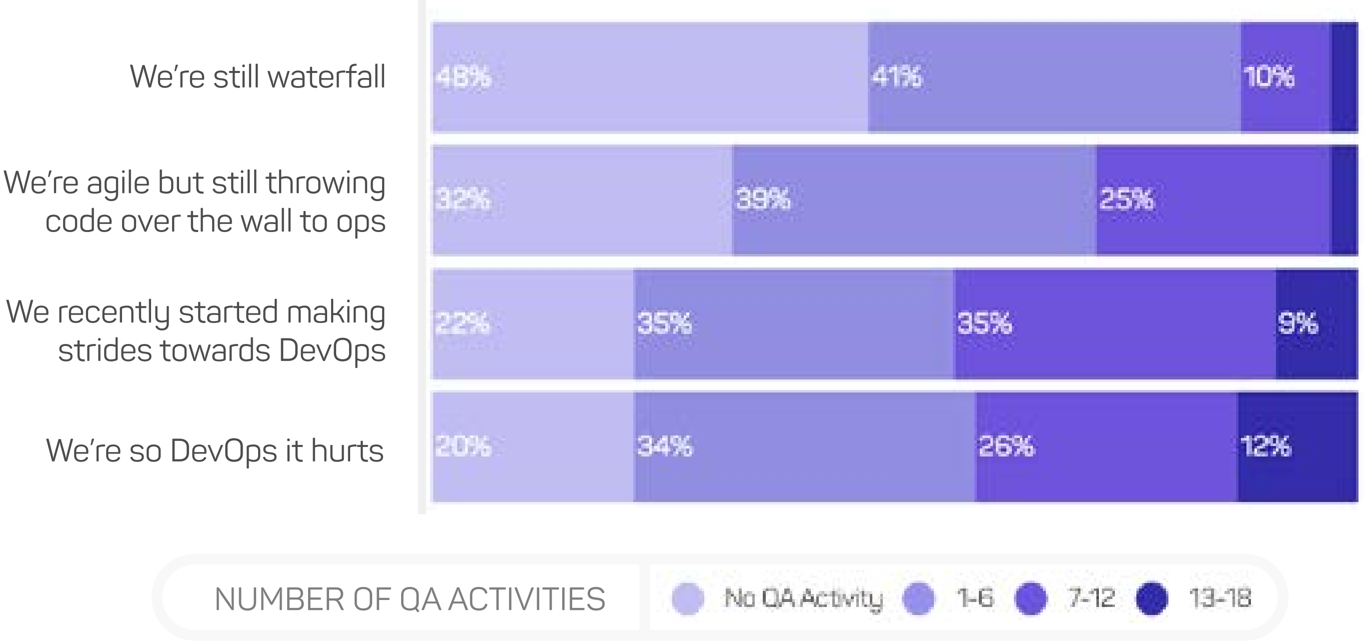
By Team Type



Emerging and veteran DevOps teams conduct around 5 different QA activities, opposed to around 4 for agile teams and around 2 for waterfall teams.

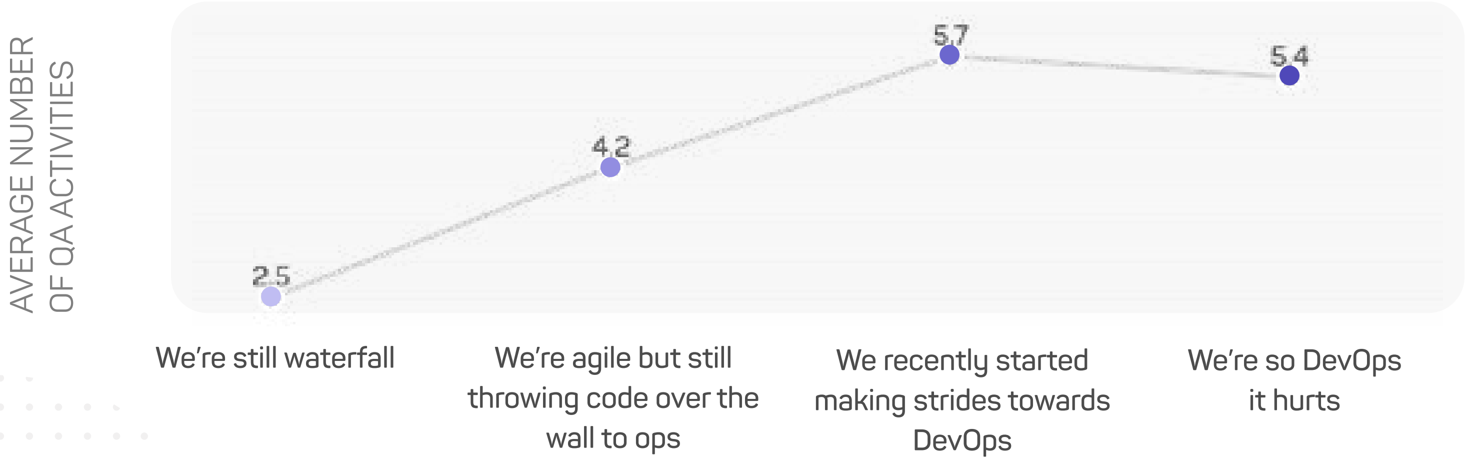
Number of QA Activities Conducted

By Team Type



***Average** Number of QA Activities Conducted*

By Team Type



In all 3 areas of manual testing, automated testing, and QA activities done, the best performers were the emerging DevOps teams, with DevOps veterans lagging slightly behind off slightly in their testing efforts. The data reveals that the average number of tests that results in "amazing" customer happiness is 4-5 tests. But DevOps-veteran teams aren't testing that much, so even though they have a better DX, their customer happiness levels aren't much higher than other teams.

Another piece of data which we highlighted in last year's report is that mean time to repair is an important contributor to customer happiness. However, this year's data makes it clear that customer happiness depends not just on speed of repair, but on quality processes that are built on a testing strong foundation. In the end, automation is king both when it comes to operating environments and how it measurably affects the bottom line. The CX should be a concern throughout the delivery pipeline, and the best way to ensure that testing doesn't get dropped at any point is to do it thoughtfully, then continuously.

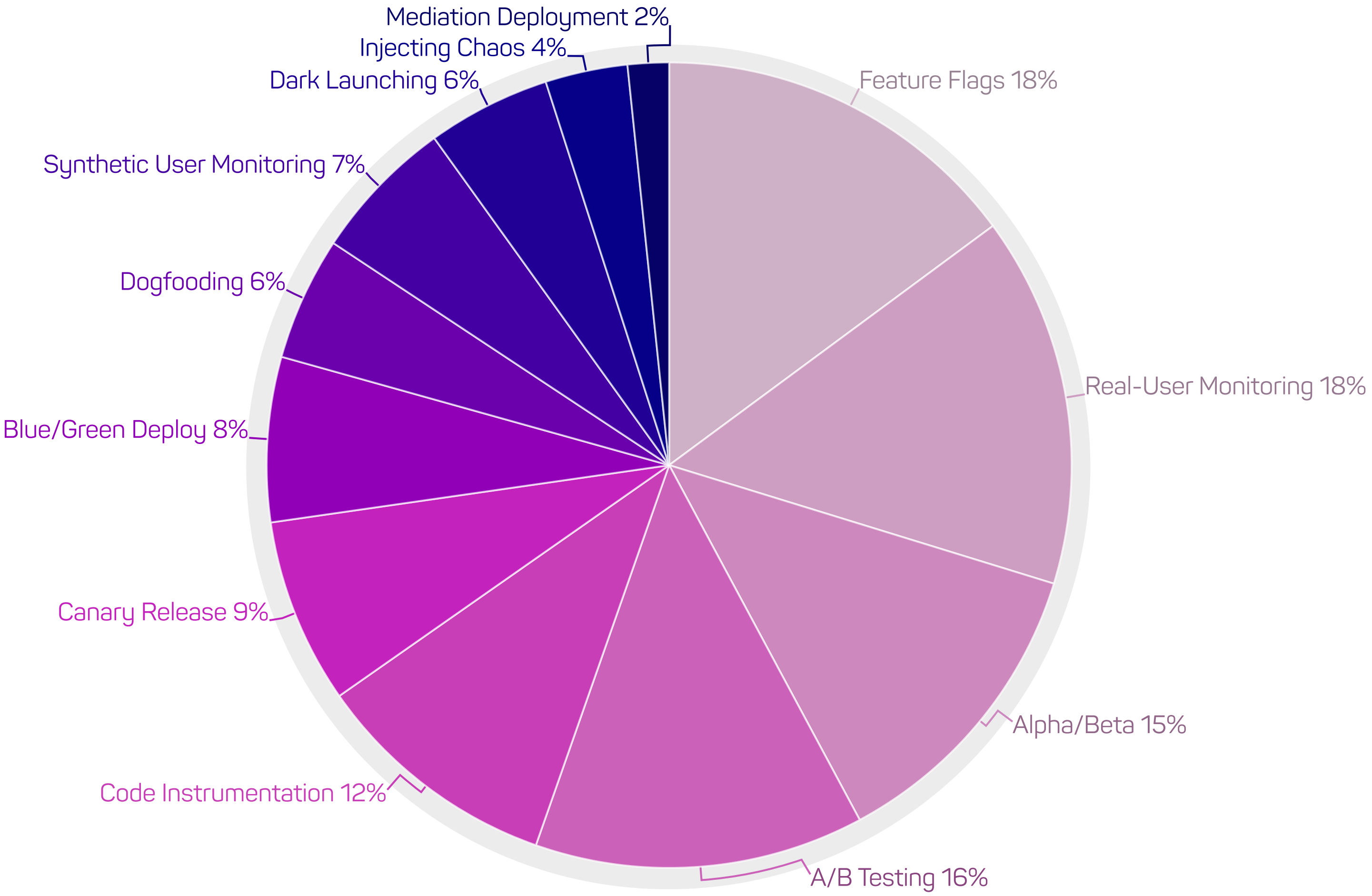
Shift-Right Testing Activities

Last but not least, we wanted to explore what teams do once a feature has made it to production. What "shift-right" testing do teams do to monitor the health of applications, test the unexpected, and ensure smooth deployments?

Building a strong testing strategy and automating it throughout the pipeline helps catch bugs before they ever make it to production; but no matter how much testing gets done early on, some bugs are never found until they reach production. Shift-right testing doesn't mean that we're reverting back to the waterfall days of only testing at the end of the application development life cycle. Rather, the practice reinforces the testing that's been done all throughout the development lifecycle by buttressing those quality checks with even more testing in production.

Real user monitoring is one well-known shift-right testing practice, but lesser known, more proactive approaches to monitoring in production can also be done. Let's check them out.

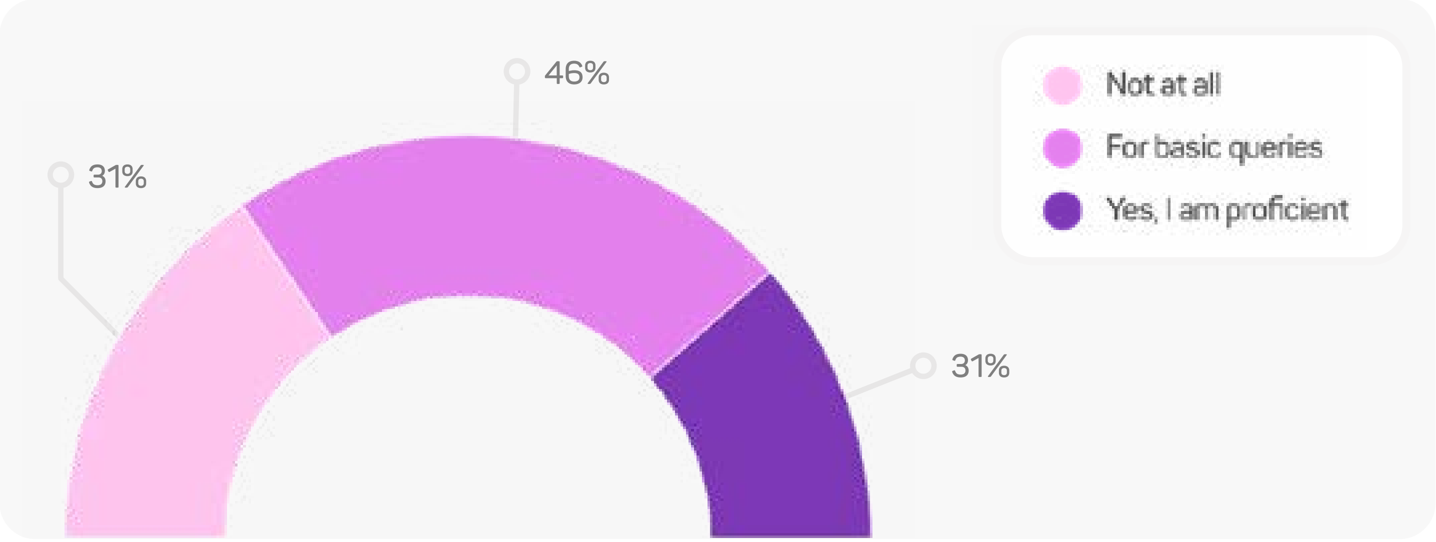
“What shift-right QA activities are done in your organization? Select all that apply.”



Real user monitoring and shipping under feature flags were the most popular activities, followed closely behind by A/B testing and Alpha/Beta testing.

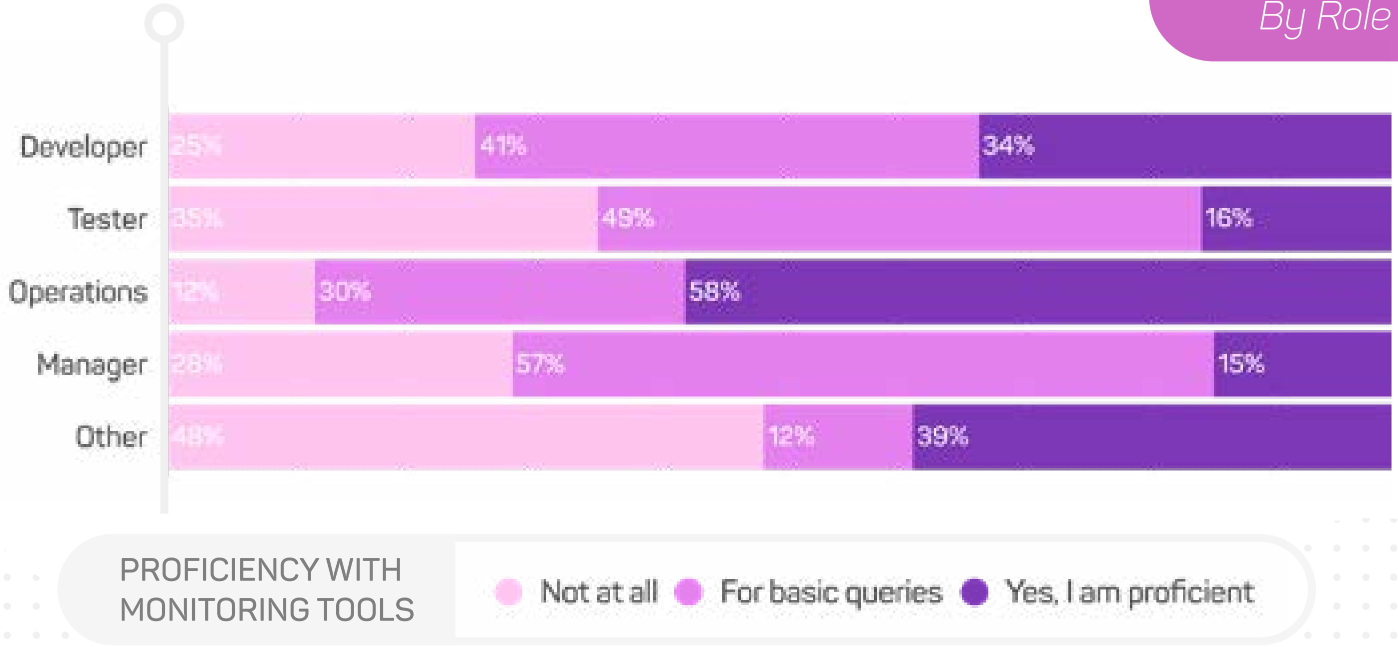
Perhaps not everyone on the team can be intimately involved in activities like code instrumentation, but they can be involved in monitoring quality of apps and customer actions in production. We thought it'd be interesting to see how well members of the best and worst teams are able to use the tools in their toolbelt.

"Do you know how to use your team's monitoring tools?"



"Do you know how to use your team's monitoring tools?"

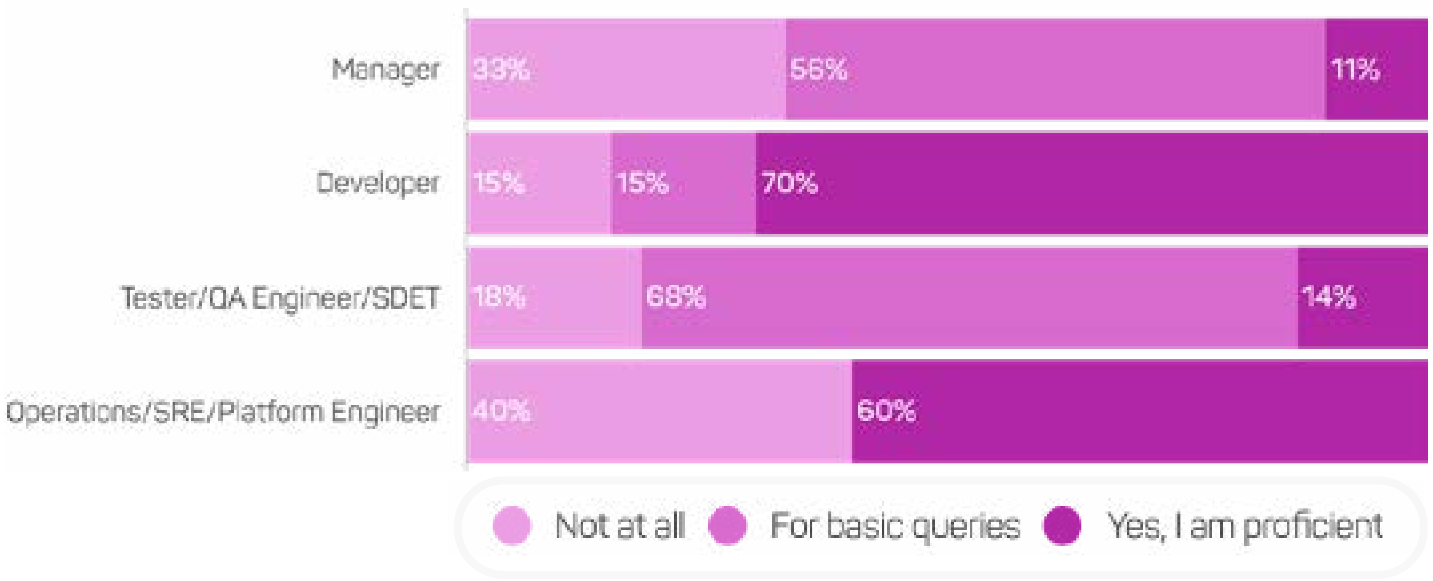
By Role



The proficiency of respondents with monitoring tools on teams with the best customer happiness scores show that operations engineers and developers are the most proficient at using their monitoring systems, and testers and management are able to pull up basic queries.

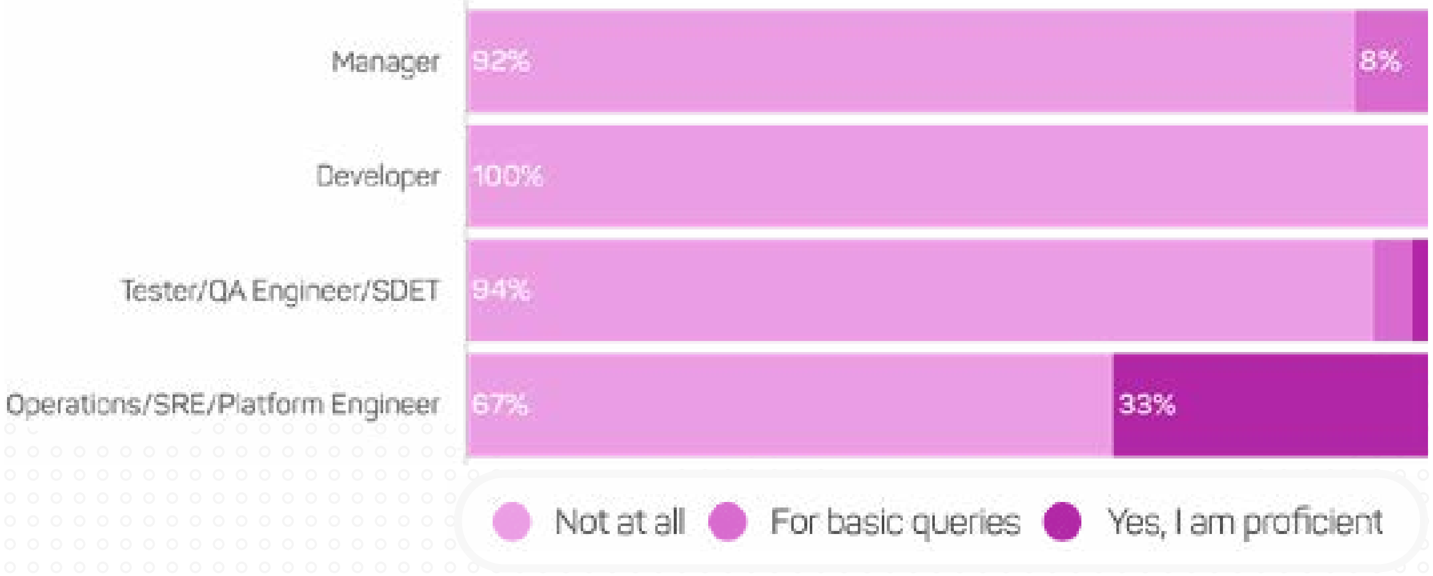
“How proficient are you with your team’s monitoring tools?”

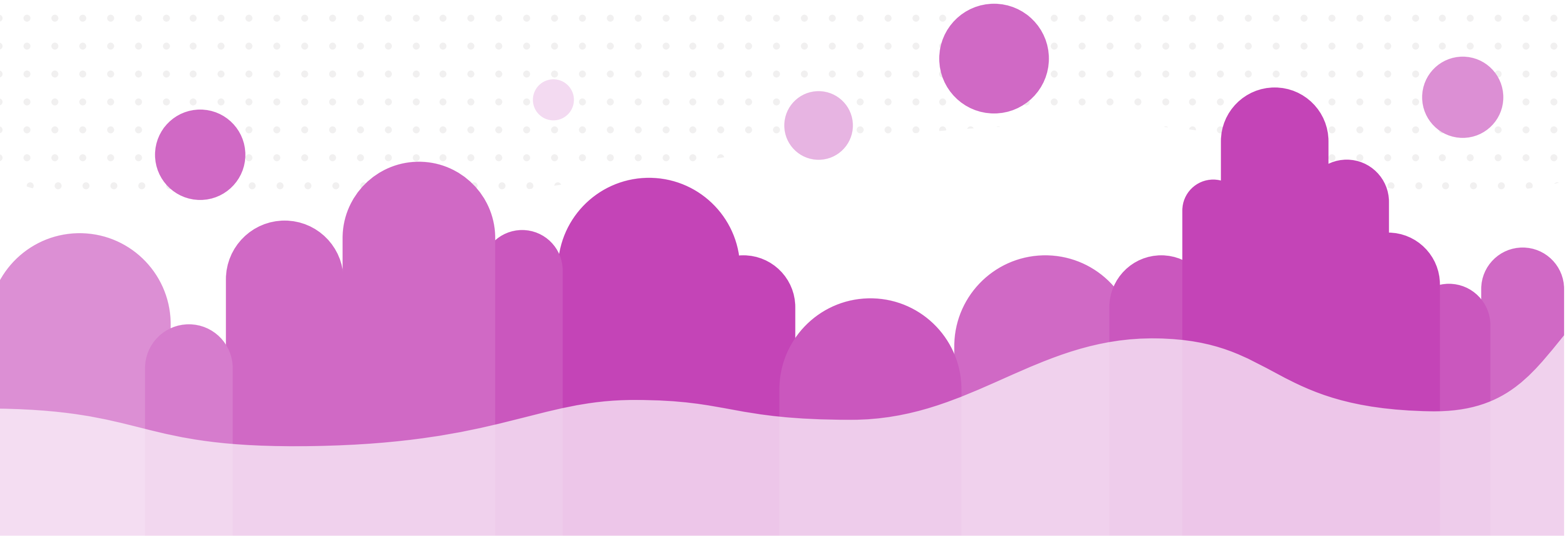
By Respondents with “amazing” customer happiness



On the other hand, respondents with poor customer happiness scores had much more limited proficiency with their monitoring tools.

By Respondents with “terrible” customer happiness





Shift-right testing appears to still be a relatively new idea to the software development world. Its more familiar sister, shift-left testing, has warmed up to the industry already, and its roots began with testers advocating for quality to become a whole team responsibility - which it rightly should be. Shift-right testing will, without a doubt, soon follow.

If we step back and look at the whole picture, this isn't just about shifting responsibilities left or right. It's about breaking down the walls between these once siloed roles so that everyone - from engineers to even sales and marketing - can work towards the same vision and goals. It's about uniting the efforts of the entire organization towards building a world class CX. DevOps and agile philosophies emerged from the same ashes - the desire to deliver value to the customer faster, uncover better ways of developing software, and help others do it.

It's time to democratize testing and quality across the entire delivery lifecycle.

Key Takeaways

Wow, still here, huh? (Or you just jumped to the end to get to the conclusions – cheater!) Thank you for being part of our DevTestOps research this year! It's always been important to us at mabl to do our proper due diligence by keeping a finger on the pulse of the test automation space and seeing where we can help steer it forward.

Here are our key takeaways from this year's report.

1. DevOps is chiefly differentiated by 1) a rapid release frequency, 2) fully automated CI/CD pipelines, and 3) the ability to promote and demote releases quickly.
2. These DevOps practices had a big impact on developer experience by giving software delivery teams more confidence in their releases and reducing the stress of release days.
3. Manual testing or DevOps processes by themselves don't improve the customer experience. The key to customer happiness is more test automation - at least 4-5 different types of tests - and a strong QA foundation to shape your test automation strategy.
4. Democratize quality assurance across the entire team - make it a whole-team responsibility.
   - An easy way to start shift testing right is by being involved in monitoring your apps in production. Start by learning how to use your team's monitoring tools, even if it's just learning how to make a few basic queries.
   - Shift testing left by taking a whole team approach to releasing software. Operations and testing teams can contribute to test design and monitoring objectives early on, and help incorporate testing into the CI/CD pipeline.

We hope this year’s DevTestOps survey has brought you as much insight as it’s given to us and helps you make both your customers’ and your life better. Don’t forget to share anything you found particularly interesting on Twitter with **#devtestopssurvey** and tag **@mablhq**.

Until next year, Happy Testing!

THE LEADING INTELLIGENT TEST AUTOMATION PLATFORM BUILT FOR CI/CD

Mabl is the only SaaS solution that tightly integrates automated end-to-end testing into the entire development lifecycle. With mabl, creating, executing, maintaining reliable tests has never been easier, allowing software teams to increase test coverage, speed up development, and improve application quality.

Start building a culture of quality today:

TRY FOR FREE