
2021
State of DevOps Report
Presented by
puppet
Sponsored by
armory
bmc
bridgecrew
BY PRISMA® CLOUD
cd
CD.FOUNDATION
circleci
New Relic.
servicenow
snyk
splunk>
Team Topologies
WOMEN IN DEVOPS

Contents

Executive summary

DevOps is not *just* automation ...

- Highly evolved firms are far more likely to have implemented extensive and pervasive automation, but being good at automation does not make you good at DevOps.
- 90% of respondents with highly evolved DevOps practices report their team has automated most repetitive tasks.
- 97% of respondents with highly evolved DevOps practices agree that automation improves the quality of their work.
- 62% of organizations stuck in mid-evolution report high levels of automation.

... and DevOps is not the cloud

- Almost everyone is using the cloud, but most people are using it poorly. However, highly evolved DevOps teams are using it well.
- Organizations should not expect to become highly evolved just because they use cloud and automation.
- While 2 in 3 respondents report using public cloud, only 1 in 4 are using cloud to its full potential.
- 65% of mid-evolution organizations report using public cloud, yet only 20% of them are using cloud to its full potential.
- While cloud and automation are important, organizations also need to address organizational and team aspects, namely helping teams clarify their mission, primary customers, interfaces, and what makes for healthy interactions with others.

Team identities and clear interaction paradigms matter

- Enterprises are held back from evolving to the highest levels by organizational structure and dynamics.
- Highly evolved firms use a combination of stream-aligned teams and platform teams as the most effective way to manage team cognitive load at scale, and they have a small number of team types whose role and responsibilities are clearly understood by their adjacent teams.
- 91% of highly evolved teams report a clear understanding of their responsibilities to other teams compared to only 46% of low-evolution teams.
- 89% of highly evolved teams report members of their own team have clear roles, plans, and goals for their work, compared to just 46% of low-evolution teams.
- While more than 3/4 (77%) of highly evolved teams state that teams adjacent to their own team have a clear understanding of their responsibilities as they relate to their own team, only 1/3 of low-evolution teams claim the same.

Cultural blockers are keeping mid-evolution firms stuck in the middle

- Challenges related to culture are most acute among low-evolution organizations, but present persistent blockers among mid-evolution firms.
- 18% of high-evolution respondents report they have no cultural blockers.
- Among mid-level respondents, a mix of cultural blockers present themselves. 21% report their culture discourages risk and 20% state responsibilities are unclear. 18% report fast flow optimization is not a priority, while 17% cite insufficient feedback loops.

Platform teams are key to success at scale

- The existence of a platform team does not inherently unlock higher evolution DevOps; however, great platform teams scale out the benefits of DevOps initiatives.
- Platform team adoption differentiates those toward the higher end of mid-level evolution from those toward the lower end, with 65% of those toward the higher end (“high-mid”) using self-service platforms and only 40% of those toward the lower end (“low-mid”) saying the same.

Introduction

DevOps is ubiquitous in software development, and has achieved such widespread adoption that it's easy to forget this wasn't always the case.

When Puppet first measured DevOps adoption a decade ago—an eternity in the world of technology—there was no common or widespread understanding of how to define DevOps, despite the existence of a vibrant, enthusiastic, and productive community, vigorously debating and sharing methodologies at events like DevOpsDays. The early champions of the movement quite deliberately avoided creating a definitive manifesto à la Agile software development, fostering a vibrant space that catered to debate, discussion, and open interpretation of methodologies.

If you were "doing DevOps" back then, you were almost certainly engaged in this community—not just copying pre-existing frameworks, but actively involved in shaping practices, and with a great deal of context.

We didn't have a plan for what [DevOps] was, and so a lot just emerged from practitioners working on real problems, by people sharing actual stories, talking about the things they wanted to improve. We kept on expanding in all directions, working out what improvements between silos could mean.

It was good and bad not to have a definition [laughs]. People... are really struggling with what DevOps is right now. Not writing everything down meant that it evolved in so many directions.

Patrick Debois, Advisor, Snyk (Formerly DevOpsDays)

As the movement expanded beyond startups and "web scale" tech companies, it became clear that in order to deliver on the DevOps promises of happier working environments, better quality software, and faster delivery times, the industry needed some codification of general principles to move from the innovators to the early adopters. Models such as The Three Ways of DevOps, CAMS and CALMS all emphasized that while DevOps was made possible by automation, programmable infrastructure, and more accessible programming languages and APIs, it was fundamentally a human-centered movement, focused on improving the interactions between people.

In fact, "culture" talks—in which speakers explore the roles of empathy, trust, and psychological safety—have always been a part of the DevOps movement and corresponding events. However, large portions of our industry led with a focus on technology without setting out to change the way work happens, which is—fundamentally—culture.

Regardless of how they define "DevOps," thousands of teams now have the ability to deploy software more safely and more quickly. They've moved from being able to deploy software only a couple of times a year to on-demand delivery, with faster remediation times and significantly improved collaboration across functions.

In fact, many of the teams that are "doing DevOps" well don't even talk about DevOps anymore—it's simply how they work.

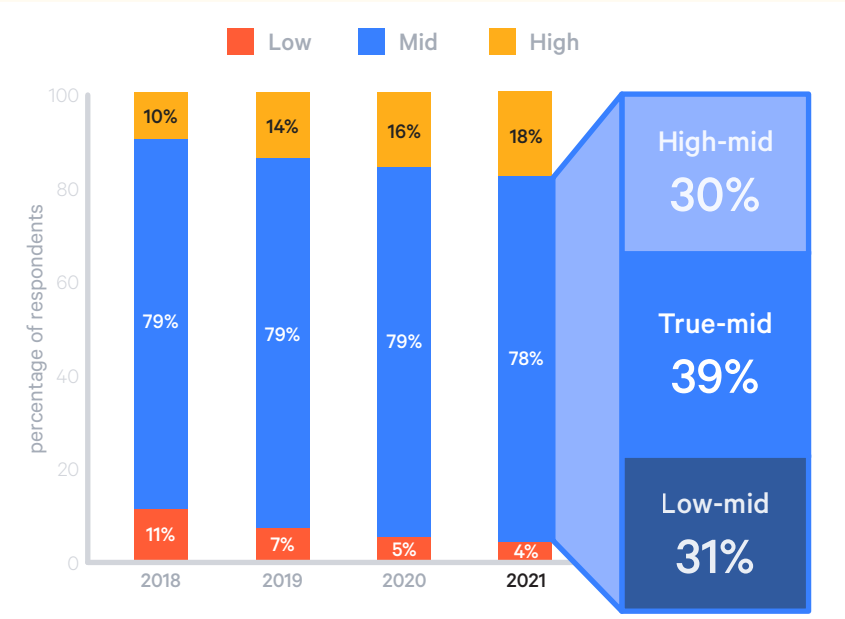
It [was during] year six or seven in this whole DevOps journey that the enterprises were finally saying yes, we could probably do this. Perhaps they were inspired by the tools maturing, the vendors appearing in the enterprise, the ideas and concepts evolving, or the unicorn companies showing massive success, but it could also be that the enterprises needed to reinvent themselves to cope.

Patrick Debois, Advisor, Snyk (Formerly DevOpsDays)

DevOps evolutionary levels

Over the last four State of DevOps surveys, the number of respondents that identify as "highly evolved" firms has grown; however, the amount of organizations in the middle level has remained stagnant.

Within the middle, we have identified three distinct levels, which we refer to later in the report as "high-mid," "middle," and "low-mid." This year's respondents are divided as such:



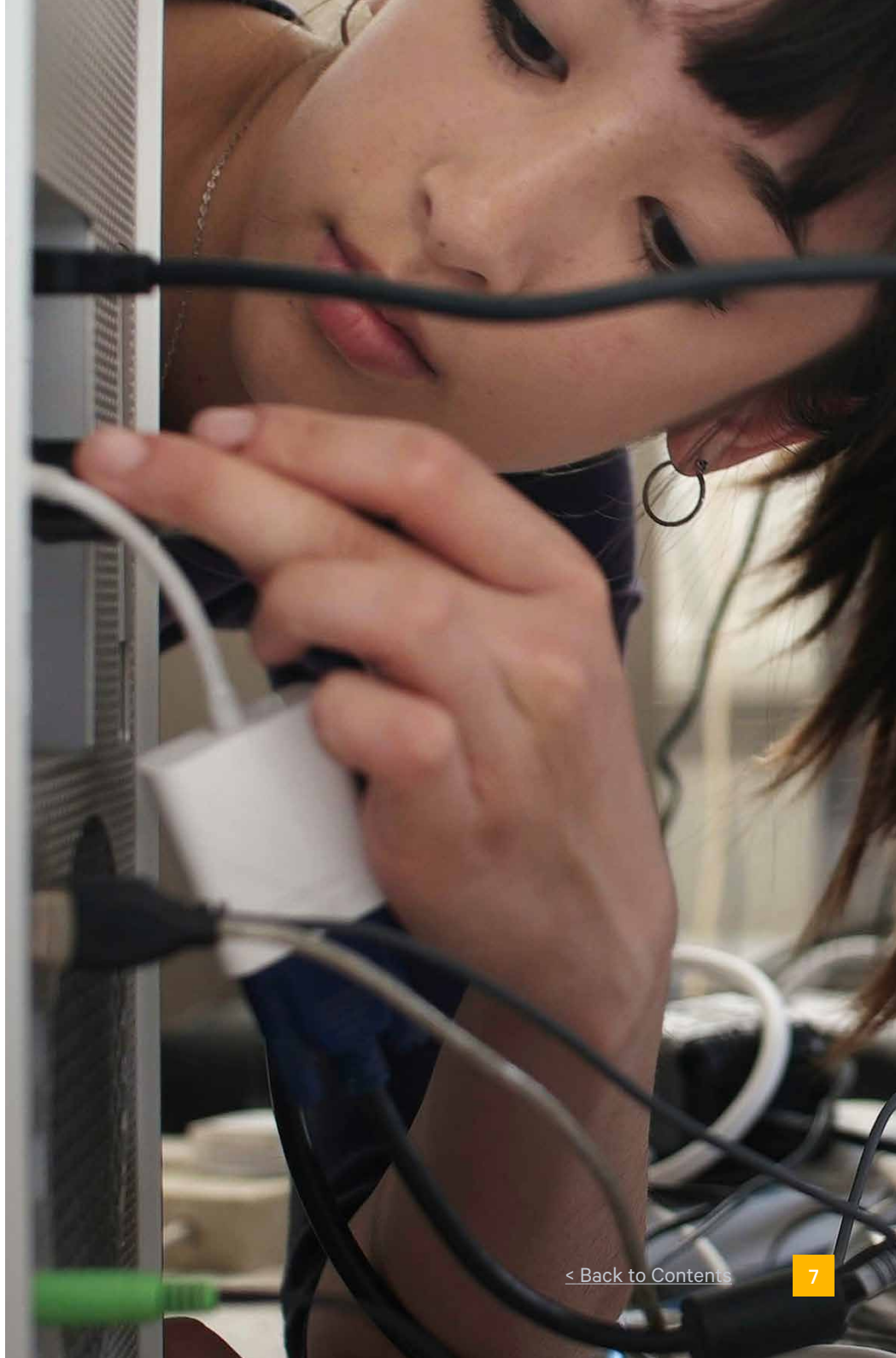
But for every team "doing DevOps" well, there are far too many organizations that have been stuck in the middle of their DevOps evolutionary journey for far too long—even if there are pockets of success in which individual teams are highly evolved.

Despite all their DevOps talk and funded initiatives, these companies have failed to address or understand the cultural, organizational, and process changes required to adopt a new way of working with technology. They may have invested in automation—67 percent of mid-evolution respondents say their team has automated most repetitive tasks—but *as an organization*, they haven't addressed the organizational silos and misaligned incentives around deploying software to production that gave rise to the DevOps movement, as evidenced by the fact that a majority 58 percent report multiple handoffs between teams are required for deployment of products and services.

It's taken me 10-plus years to come up with my own one-line definition of DevOps: ***"DevOps is whatever you do to bridge friction created by silos, and all the rest is engineering."*** *And so, if you're doing technology just for the technology and you're not trying to overcome some friction of the human kind of siloing or group siloing or information siloing or whatever, then* ***you're just doing the engineering part and you're not, in my opinion, doing the DevOps part.***

Patrick Debois, Advisor, Snyk (Formerly DevOpsDays)

This is evident in the survey results from this year, as well as from the last few years. Of all of the organizations we've surveyed over the last four years, approximately 80 percent of them fall into mid-level evolution. This year, we're going to dive into that middle group to understand why *some* organizations are able to evolve, while the vast majority get stuck in the middle. We'll also discuss what differentiates organizations within mid-evolution, and the various ways folks at the "high end of the middle" are much closer to high evolution than their "low end of the middle" counterparts.



Acknowledgments

It's now been 12 years of DevOpsDays, 11 years of DevOps Weekly, and 10 years of State of DevOps Reports, so we're going to do something a little different this year. As in previous years, we will dive into the survey data and contrast those results with prior research, but we are also going to take the opportunity of this being the tenth anniversary to do more editorial commentary than we've done in the past. As such, we've invited a wider group of people to respond to the data and provide their thoughts on where the movement is headed. You'll see their contributions sprinkled throughout the report. We also augmented our normal quantitative survey approach with a set of qualitative interviews, and we've included anonymized quotes from that research.

This year we're thrilled to have Matthew Skelton and Manuel Pais join us in authoring the report. As part of our investigation into platform teams in the 2020 State of DevOps Report, we made reference to the comprehensive Team Topologies model they produced, and we're excited to be working more closely with them this year. The Team Topologies model has been immensely influential for many of us in the industry, and their call to focus on team identities, inter-team interaction paradigms, and optimizing for fast flow software delivery aligns perfectly with the themes for this year.

The historical focus has rightly been on technical practices. Organizations that have put in place these technical foundations are finding that they need to address team dynamics in order to move faster safely. This is where principles and practices from Team Topologies can help.

Matthew Skelton & Manuel Pais, Team Topologies

In addition to this year's authors, there is a substantial group of people who have contributed to the State of DevOps Reports over the last decade, and we want to acknowledge their contributions.

The most important individuals to recognize are Alanna Brown and Dr. Nicole Forsgren, the two people who shaped this report more than anyone else. It was Alanna's idea in the first place to survey this emerging movement, and for the last nine years she coordinated all of the activities around State of DevOps surveys and Reports, engaged new folks as co-authors, and in more recent years, led research direction and focus. Nicole brought academic rigor, professional research expertise, and initiated the pivot towards focusing on IT performance, the relationship with business performance, and so much more. The combination of Alanna's expertise in spotting market trends and meaningful insights into their evolution with Nicole's rigorous research methods, and their joint efforts to share this work broadly, helped to set new industry standards for high performing teams.

Perhaps most importantly, this spirit of open access to information helped many teams work out where they should be aiming for, and how to get there. This report would not exist without Alanna's and Nicole's efforts over a great number of years.

Other people have also made significant contributions over the last decade, in roughly chronological order: James Turnbull, Gene Kim, Jez Humble, and Andi Mann. We appreciate their contributions immensely and this report wouldn't be the same without any of them, particularly the DORA team of Nicole, Jez, and Gene.

For every person who completed the 2021 State of DevOps survey, we donated $5 to the National Coalition for Homelessness, the World Central Kitchen, and the UNICEF COVID-19 Solidarity Response Fund.

We donated additional funds provided by our generous sponsors, bringing our total contribution to $45,000. Our sponsors this year include: Armory, BMC, Bridgecrew, Continuous Delivery Foundation, New Relic, ServiceNow, Snyk, Splunk, Team Topologies, and Women in DevOps.

A decade of measuring DevOps

By 2013, the State of DevOps Report had established the relationship between a true DevOps practice—the combination of people, practices, and culture—and high performance outcomes. Organizations practicing DevOps consistently report more frequent deployments, shorter lead time to change (LTTC), lower change failure rates, and faster mean time to recovery (MTTR). These four metrics don't encompass all of DevOps, but they illustrate the measurable, concrete benefits of pairing engineering expertise with a focus on minimizing friction across the entire software lifecycle.

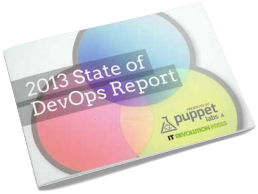
They also have a direct impact on business outcomes—organizations that can deploy on demand and have shorter LTTC create comparative advantages by delivering solutions to their customers faster. Shorter mean time to recovery and lower change failure rates create more stable systems.

Puppet's first State of DevOps Report identified the primary blockers to adoption as: "the value isn't understood outside my group" and "there's no common management structure between dev and ops"

Puppet's 2016 State of DevOps Report measures the ROI of DevOps beyond the KPIs: better employee loyalty, less time on unplanned work and rework, less time remediating security issues, and cost savings

Puppet's 2021 State of DevOps Report investigates team interactions and recommends actions toward "breaking down the middle"

2011 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 | 2019 | 2020 | 2021



Puppet's 2013 State of DevOps Report established that DevOps enables high performance: more frequent deployments, shorter LTTC, lower change failure rates, and faster MTTR

Puppet's 2018 State of DevOps Report details a staged approach to DevOps and introduces "the DevOps Maturity Model"

For eight years, highly evolved DevOps teams have consistently demonstrated better performance across four key software performance metrics: deploying to production on demand, reporting change lead times and mean times to recover under one hour, and change fail rates under five percent.

Over the last decade plus, we've seen DevOps move through all the stages of the technology hype cycle, from the thing all the cool kids were obsessed with, through the peak of inflated expectations, through the trough of disillusionment, and eventually to a point where pundits started declaring it dead, killed by various other emerging methodologies

Yet here we still are: researching, writing, and debating DevOps. Because even though DevOps is everywhere, it's rarely done well at scale, particularly in the enterprise.

Today, 83 percent of IT decision makers report their organization is implementing DevOps practices. Yet the past four State of DevOps Reports have shown the vast majority of organizations are stuck in the middle. Organizations in the middle have achieved pockets of success—increased automation, more self-service available, etc.—yet these successes are often limited to a few teams, and thus fail to create meaningful organizational change. This can result in staff and leadership thinking their DevOps implementation has failed when in fact it's simply facing new or more advanced hurdles.

The case for DevOps remains clear

Highly evolved organizations have consistently demonstrated higher performance across four key software performance metrics.

	Low	Mid	High
Deployment frequency	Monthly or less often	Between daily and weekly	On demand (whenever we want)
Lead time for changes	Between a week and 6 months	Less than a week	Less than an hour
MTTR	Less than a week	Less than a day	Less than an hour
Change failure rate	Less than 15%	Less than 15%	Less than 5%

DevOps success requires support from every level of the organization

The most highly evolved firms benefit from top-down enablement of bottom-up transformation. Fewer than two percent of high-level organizations report resistance to DevOps from the executive level. Strong teams can create substantive change within themselves and in adjacent teams, yet in the absence of meaningful leadership support, success will be confined to pockets, and widespread evolutionary improvement will not occur.

In 2017, we showed the critical role that transformational leadership has on DevOps initiatives and IT performance:

High-performing teams reported having leaders with the strongest behaviors across all dimensions: vision, inspirational communication, intellectual stimulation, supportive leadership, and personal recognition. In contrast, low-performing teams reported the lowest levels of these leadership characteristics.

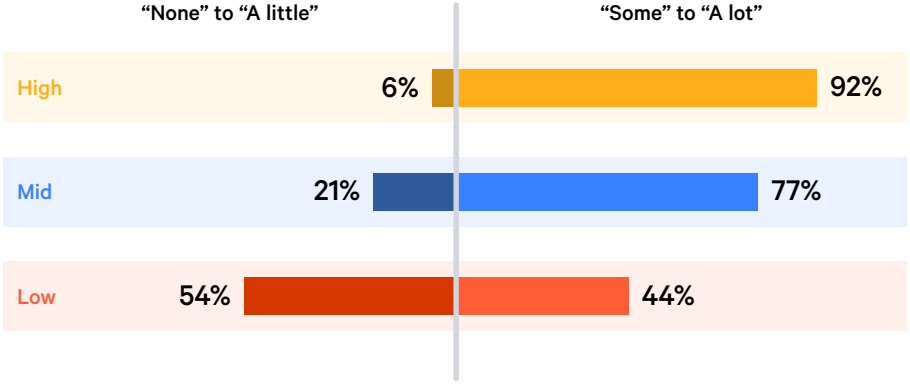
[2017 State of DevOps Report](#)
presented by Puppet and DORA

If you're a team leader or individual contributor who has done great things within your team and made an impact on teams with whom you closely work, yet you don't have managerial support further up the chain... we have bad news for you. Your organization won't transform, and success will only go so far.

This year's State of DevOps Report shows that organizations are making progress towards higher performance but cultural issues are still obstacles to success. Deeper leadership buy-in is needed to enable practitioners to make the changes they need for that success.

James Turnbull, VP of Engineering, Sotheby's

How much momentum is currently behind your organization's DevOps implementation efforts?



We asked respondents this year about the momentum and sentiment of DevOps initiatives in their organization. The results are evidence that your assessment of your own momentum likely correlates with the evolutionary model—it's not just your own cynicism sneaking in. Less successful initiatives have less momentum behind them.

Among those in the highest level of DevOps evolution, 66 percent claim they have a lot of momentum behind their DevOps implementation efforts, compared to 30 percent in the middle, and only four percent of the lowest level who report the same.

It's also true that successful DevOps initiatives require support from more than just managers and leaders in the organization; they require buy-in from the folks on the ground actually doing the work. While leadership shoulders the burden of enabling meaningful change, among the mid-level firms who report DevOps is passively or actively resisted at their organization, resistance comes from all levels (36 percent managerial, 29 percent executive/VP/director, 29 percent practitioner).

Historical demographics

Are we past peak "DevOps Team?"

Seven to 10 years ago, the term "DevOps" in a job title was controversial. It wasn't a title, just as "agile" wasn't a title; methodologies weren't supposed to be job titles. However, that didn't stop the industry from adopting "DevOps" in job and team titles all over the place. We saw the peak of that team title in our 2018 data.

You can't engineer DevOps but when you say "automation engineer" people think you are a Q&A tester. I tend to write DevOps with an asterisk as a means for me to explain to what capacity I've worked in DevOps and how my experience served that role.

Katharine Yi, Senior Cloud Engineer at Arena Analytics

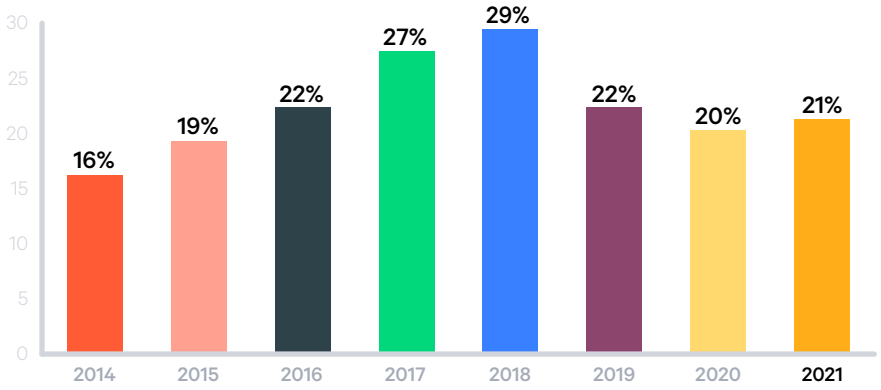
Some of the original resistance to "DevOps team" or "DevOps engineer" was a reaction to the fact that many organizations believed that renaming an operations team or title was a shortcut to doing the hard work around collaboration, empathy for partner teams, and changing the way work happens. Perform a rename and voilà! Hang a "Mission Accomplished" banner.

Despite the skepticism, the job title did do a couple of things well. Firstly, it showed that a company was at least thinking about trends in the industry. It might not be adept at those trends yet, but the intent—and the signal—was there. Secondly, we saw a rise in salaries for those with "DevOps" in their job title, and that's beneficial for workers as a whole. If a job title change comes with a raise, by all means, update that business card.

As the data shows the title "DevOps engineer" (and thus the "DevOps Team" as a label) declined a bit in the last few years, it's worth asking why. Are we past peak DevOps? Is it because teams are more centered around purpose and firms have realized that centralization of DevOps will not be successful? Did they discover the work was larger than a DevOps team? Or is it the teams formerly called "DevOps" now insist on being called "SRE" and are experiencing corresponding salary movement?

What we can also see from the data is that job titles like "System Administrator" have declined since 2014 (38 percent) when we started asking about this compared to 2021 (three percent). Similar trends exist for "Operations" and "Infrastructure Engineer." This may indicate firms shifting away from cost center mentality on this type of technical expertise or it may again trend with renaming teams and groups doing work quite similar to the tasks at hand a decade ago. Some of these job title shifts simply indicate changes in fashion, while the roles themselves have stayed much the same.

Do you identify your team as "a DevOps team"? (Yes answers)



The argument over whether "DevOps" belongs in job and team titles is nearly as old as the DevOps movement itself, and from comparing past surveys, we see that we reached "peak DevOps team" in 2018, and are now nearly back at 2016 levels.

It's generally understood (as per the original DevOps Topologies) that a separate DevOps team sitting between Development and Operations is an anti-pattern. However, we know that in many organizations, the role that a "DevOps team" plays can vary widely, including:

- A team with end-to-end product responsibilities (doing "Dev" and "Ops" together).
- A team with responsibilities for supporting Dev teams with a combination of release automation, deployment pipelines, and developer tooling.
- A team that builds the awkward things that application developers don't want or need to care about: infrastructure, container fabrics, monitoring, and metrics.
- A team responsible for encouraging and enabling DevOps practices across an organization.

We strongly believe that the presence of "DevOps teams" is confusing for the industry and many organizations, and in most cases doesn't help organizations evolve. In our experience, organizations that have less ambiguous team names, with more clearly defined responsibilities, are far more likely to have a higher performing IT function. In this year's data, we see members of "DevOps teams" self-describe their functions as those of traditional I&O (20 percent) and SysAdmin (seven percent) teams. Lack of clarity around team identities creates significant organizational friction, impeding software delivery in a variety of ways.

We strongly believe that the presence of "DevOps teams" is confusing for the industry and many organizations, and in most cases doesn't help organizations evolve.

In our experience, organizations that have less ambiguous team names, with more clearly defined responsibilities, are far more likely to have a higher performing IT function.

We suggest that organizations move away from the use of "DevOps teams" towards clearer team names, and in particular that the use of stream-aligned and platform teams is a well-defined path to achieving DevOps success at scale. These team types are used in the Team Topologies model that we cover in more depth later on in the report.

Four percent of survey respondents report they work on a "platform engineering" team, which seems to support the idea that this is an emerging trend (internal platform teams as accelerators of flow) in which teams move away from a focus on "DevOps" as a set of tools and practices (and title) towards a focus on the mission of their team in relation to other teams and the overall organization (reducing cognitive load, reducing friction and context required from Dev teams to adopt better/modern tools and practices).

People think that DevOps is an org structure. To me, it's not. It is a mindset.

VP of Software Engineering and Data Science, Insurance

Blockers to DevOps evolution

The past decade of DevOps featured innumerable editorials posing the question: is the problem culture or is the problem technology? We made the case in 2014 that the structural changes necessary for DevOps are largely cultural. While this is true, the reality is the problems organizations need to solve are some combination of culture and technology.

Low-evolution teams have a familiar (if dizzying) mix of blockers to better DevOps practices. Most cite organizational resistance to change (31 percent), followed by legacy architecture (28 percent), shortage of skills (23 percent), limited or lack of automation (20 percent), and unclear goals or objectives (20 percent).

Mid-evolution teams, having secured some level of organizational buy-in and clarified objectives, face a different set of challenges. Armed with a clearer sense of the task at hand and having begun to automate more, mid-level teams cite a shortage of skills (33 percent), legacy architecture (29 percent), organizational resistance to change (21 percent), and limited or lack of automation (19 percent) as the primary blockers to better DevOps practices. Later in this report we will focus on how teams advance through mid-level evolution, overcoming first more technical then more cultural blockers.

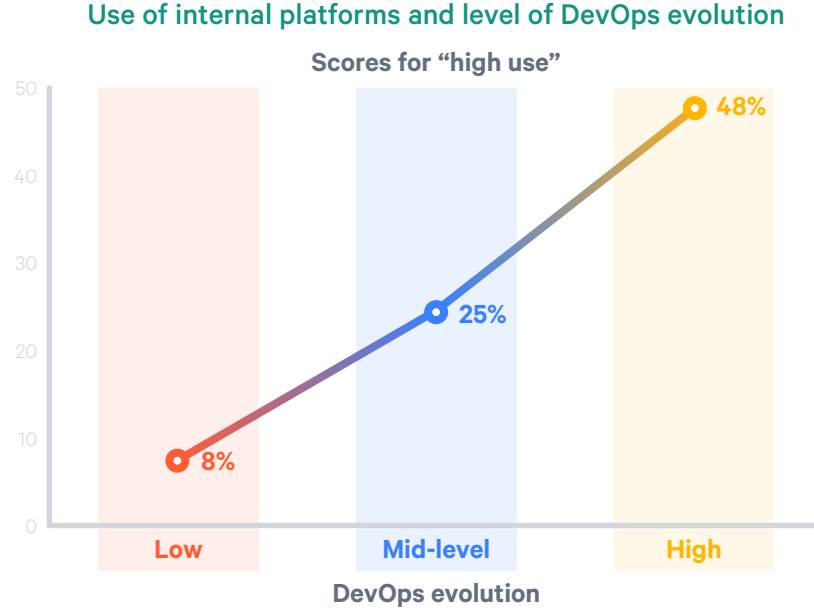
Highly evolved teams' blockers are fundamentally different from their mid- and low-evolution peers. They've overcome organizational buy-in blockers and the cultural barriers tangential to organizational buy-in. They've created a technology stack that leverages significant automation and invested in internal platforms. As a result, their blockers are fundamental challenges—ones at which they and the entire technology industry can expect to chip away for years to come: legacy architecture (29 percent) and a shortage of skills (29 percent).

In short, for highly evolved firms, culture is no longer a barrier. This is ultimately why they're highly evolved.

Team identities & interaction paradigms

The common thread we've seen across all of the organizations that are "good at DevOps" is that their teams have strong identities that are understood across the organization, these teams have clear responsibilities with a high degree of autonomy over their own function, and, most importantly, they have well-defined interaction paradigms and communication channels with other teams.

In particular we've seen the vast majority of these organizations have adopted the platform team model that we first covered in the 2020 State of DevOps Report, where we found a high degree of correlation between DevOps evolution and the use of internal platforms.



Not every platform team is automatically successful, but the successful ones treat their platform as a product. They strive to create a compelling value proposition for application teams that is easier and more cost-effective than building their own solutions.

Any time standards, practices, processes, frameworks or architectures become a mandate, I've seen little to no adoption. Sometimes it will happen out of fear and there could be some immediate results, but they won't be sustainable and usually teams find workarounds or alternatives. I have seen more success when organizations focus on the developer experience — creating guardrails that reduce burden and enable agility, including a mechanism for feedback. Some amount of discipline enables speed.

Courtney Kissler, CTO, Zulily

This relationship between value-stream oriented application teams and the platform teams offering self-service capabilities enables fast flow, and successful organizations have discovered and adopted the patterns described in the Team Topologies model, which we'll take a look at in more detail in a moment.

Team Topologies' principles and practices help organizations to become highly evolved due to the focus on a fast flow of changes through the organization. In particular, the Team Topologies model supports the belief that highly evolved teams tend to do a good job of limiting extraneous cognitive load on delivery teams (through good practices, automation, and support from other teams), leaving more capacity to focus on the business needs. If cognitive load is left "unbounded" (i.e. keeps growing as teams' responsibilities grow without limits) then all these performance metrics will be negatively affected, preventing teams from evolving to higher levels.

What is "fast flow"?

"Fast flow" or **"continuous flow"** is a term that came out of the Lean movement, and describes a process with optimized output and little waste. A core Lean principle is that a consistent flow of work is essential for faster and more reliable delivery.

Team Topologies key concepts

Four Fundamental Topologies

- **Stream-aligned team:** Aligned to a flow of work from (usually) a segment of the business domain.
- **Enabling team:** Helps a stream-aligned team to overcome obstacles. Also detects missing capabilities.
- **Complicated subsystem team:** Where significant mathematics/calculation or hard-to-find niche technical expertise is needed full-time.
- **Platform team:** A grouping of other team types that provide a compelling internal product to accelerate delivery by stream-aligned teams.

Four Fundamental Topologies, with the flow of change

The flow of change is shown left-to-right. Stream-aligned teams own an entire slice of the business domain (or other flow) end-to-end. The stream-aligned teams are "You Built It, You Run It" teams. There are no hand-offs to other teams for any purpose.

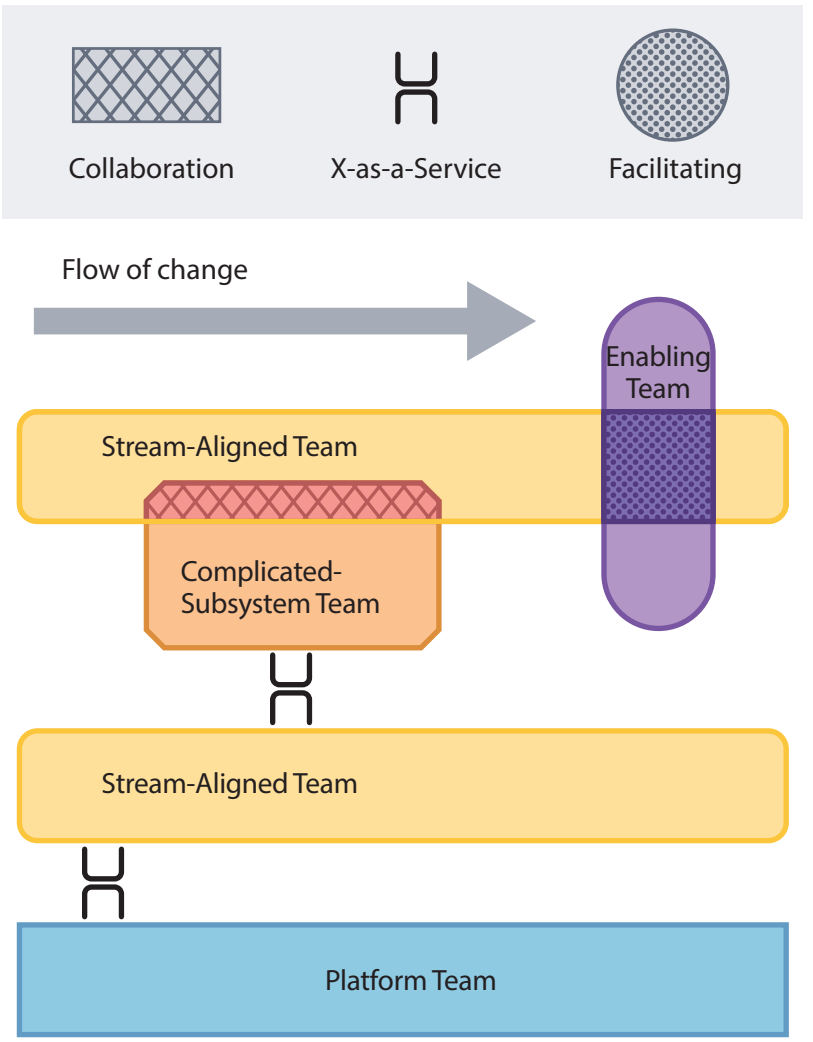
This diagram is a snapshot in time. The team relationships will change as new goals are set and the teams discover new things.

Three team interaction modes

There are only three ways in which teams should interact:

- **Collaboration:** Working together for a defined period of time to discover new things (APIs, practices, technologies, etc.).
- **X-as-a-Service:** One team provides, one team consumes something "as a Service."
- **Facilitation:** One team helps and mentors another team.

(Content used with permission, Matthew Skelton & Manuel Pais, Team Topologies, 2019.)



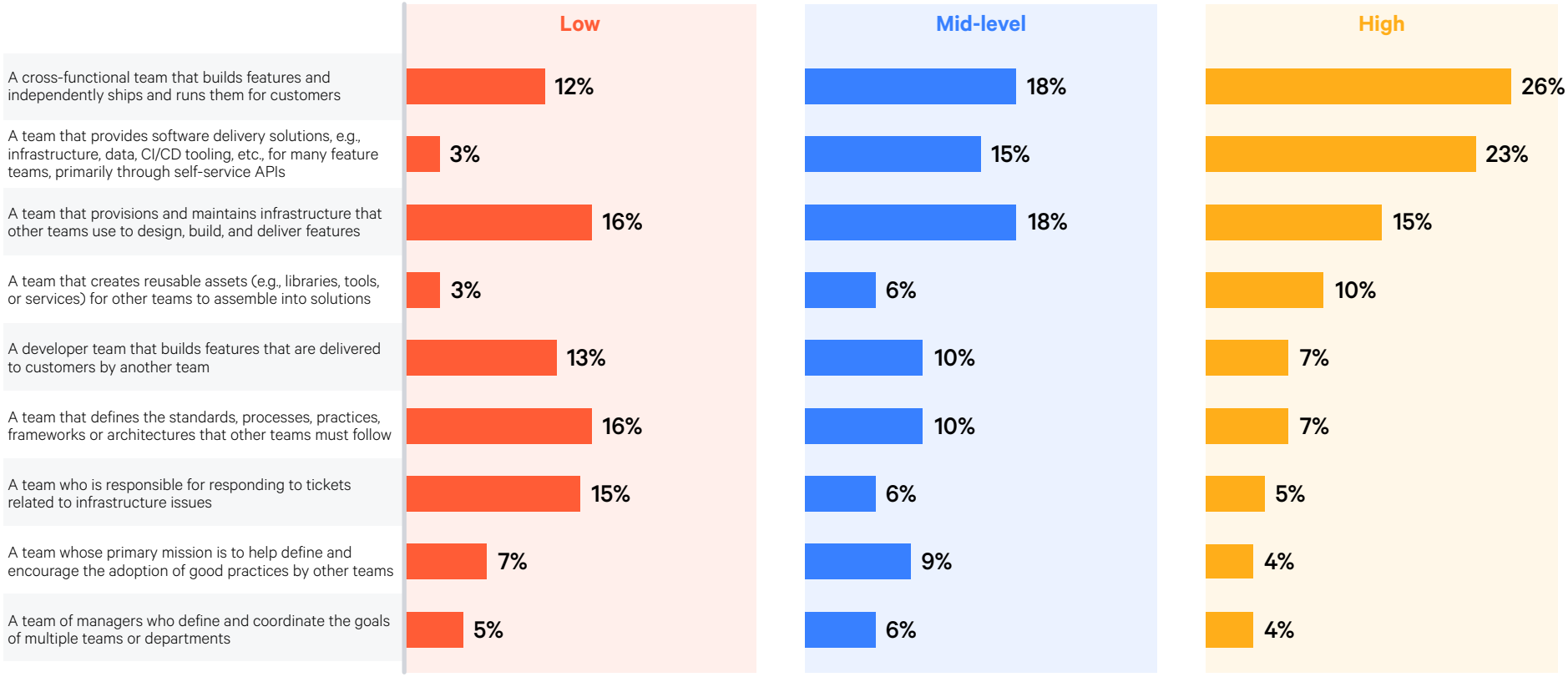
(Image reproduced courtesy of Matthew Skelton & Manuel Pais, Team Topologies, 2019.)

Highly evolved organizations tend to follow the Team Topologies model

Success at scale requires optimizing not for the individual, and not for the team, but for the wider organization—the "team of teams," if you will. This year's survey results show that highly evolved organizations have discovered the patterns that work well for a fast flow of change.

They use a combination of stream-aligned teams and platform teams as the most effective way to manage team cognitive load at scale, and they have a small number of team types whose role and responsibilities are clearly understood by their adjacent teams.

Use of internal platforms and level of DevOps evolution



It's critical to note that success requires more than having the right kinds of teams; it also requires focus, and structure around the way information is shared between and across teams, and how they interact with one another. Simply restructuring departments without focusing on how they work together will not lead to success.

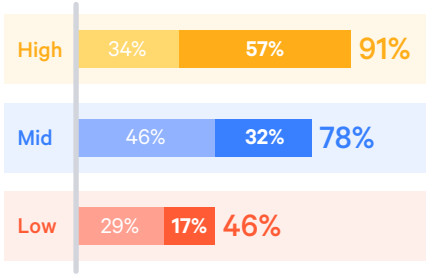
We found evidence of how this plays out in this year's findings. Highly evolved teams demonstrated a deeper understanding of team roles and interactions more often than mid-evolution teams and consistently more than twice as often as low-evolution teams. Further, 91 percent of highly evolved teams report a clear understanding of their responsibilities to other teams while 89 percent report members of their own team have clear roles, plans, and goals for their work. In contrast, only 46 percent of low-evolution teams have a clear understanding of their responsibilities to other teams and again 46 percent report clear roles, plans, and goals for their work on their own teams.

While more than three-quarters (77 percent) of highly evolved teams state that teams adjacent to their own team have a clear understanding of their responsibilities as they relate to their own team, only one-third of low-evolution teams claim the same. More than two-thirds of mid-level teams and 85 percent of highly evolved teams agree that teams who share common tooling, language or methodologies actively share best practices with one another, while less than one-third of low-evolution teams agree.

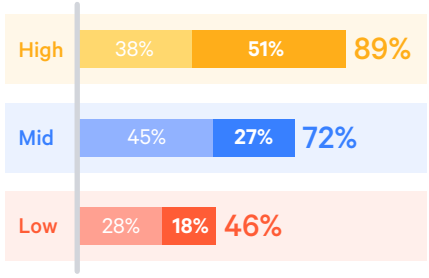
By contrast, mid- and low-evolution firms have a higher number of teams that are not present in Team Topologies and are not optimized for a fast flow of change: siloed development teams, architect teams, support teams responding to tickets, and older-style infrastructure teams who focus on provisioning and maintenance. Critically, these teams tend to lack true end-to-end ownership for their services (external and internal) and are therefore constrained not just in terms of speed to deliver but also in their ability to receive feedback quickly. The slower that communication is internally, and the less clarity there is around service ownership, the more difficult it is for feedback to be directed to the people who are able to act upon that feedback. This provides evidence that the recommendations in Team Topologies help to orient an organization towards becoming more highly evolved.

Clarity of purpose, mission, and operating context seem to be strongly associated with highly evolved organizations. They realize that a healthy ecosystem of loosely coupled but highly cohesive teams is what helps move the needle for the organization, as focusing solely on optimizing teams in isolation is insufficient.

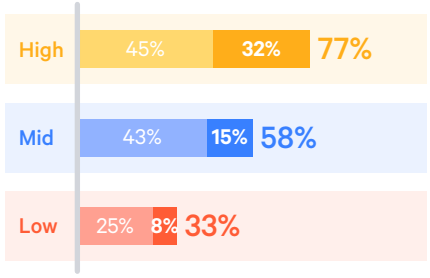
My team has a clear understanding of our responsibilities to other teams



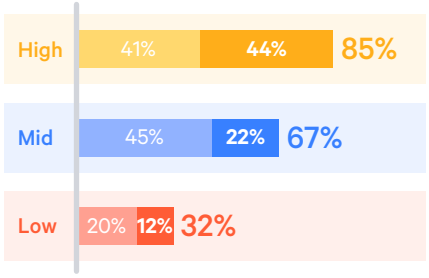
Members of my team have clear roles, plans, and goals for their work



Teams adjacent to my team have a clear understanding of their responsibilities as they relate to my team



Teams that share common tooling, language, or methodologies actively share best practices with one another



DevOps is a cultural paradigm, and sort of a melded responsibilities set that makes every developer also responsible for operations and every operations person also responsible for participating in development.

Director of Engineering, Manufacturing

Do we know what DevOps is yet?

We've come a long way in 10 years. When we first began to research, define, and measure DevOps, there was no widespread awareness of what it was or understanding of how it was practiced. Nine percent of the people who took our 2011 DevOps survey admitted to having no idea what DevOps was. Overwhelmingly, the majority of respondents (51 percent) indicated that DevOps was the "interaction between development and operations," while 28 percent stated it was "cultural change in IT."

Since that time, DevOps has crossed the chasm. It is widely accepted as a model of best practice for Agile development, and technology professionals at all levels are easily able to provide their own definitions upon request.

And yet, now that DevOps has become mainstream, we also hear folks use DevOps and its component parts or solutions tangential to DevOps interchangeably. DevOps is not a panacea for all development malpractices. In order to understand what DevOps is and how to implement DevOps to your organization's advantage, it's important to first understand what DevOps is not.

DevOps is a cultural shift… to enable better communication, collaboration, integration, and automation across the organization.

Director of IT, DevOps, and Engineering, Financial Services

DevOps is the blending of the infrastructure team and traditionally what software developers would've been doing as it relates to release management and even standing up of infrastructure.

VP, Software Engineering, Insurance

DevOps is not *just* automation

Automation enabled DevOps, and has been a fundamental pillar of the movement since inception, but a singular focus on automation won't deliver a successful DevOps practice.

As an industry, we hyper-focus on the automation aspects of DevOps to the detriment of team interactions, fast flow, collaboration, and optimization of the whole system, and we do this because building out automation is a concrete, technical task that can usually be done by a small number of teams. It's much easier to implement automation than it is to address issues with organizational dynamics that involve different teams, different managerial chains, and almost certainly require modifying processes that impact an even larger number of teams.

Nevertheless, DevOps is not synonymous with automation. Our data shows there is a relationship between the two, in that highly evolved organizations are far more likely to have implemented extensive and pervasive automation, yet this relationship is not predictive.[1] Automation does not automatically make you highly evolved—62 percent of organizations stuck in mid-evolution report high levels of automation—but being good at automation is what enables higher levels of performance—both in terms of your IT systems, and in terms of the teams of people who operate and consume those systems.

One of the most unfortunate incarnations of DevOps we see, particularly in large companies, is treating it purely as "Developer Operations," focused entirely on the care and feeding of CI/CD pipelines. Build engineers bridge the gap between development and production, and so it's understandable how we ended up here—yet this is not DevOps.

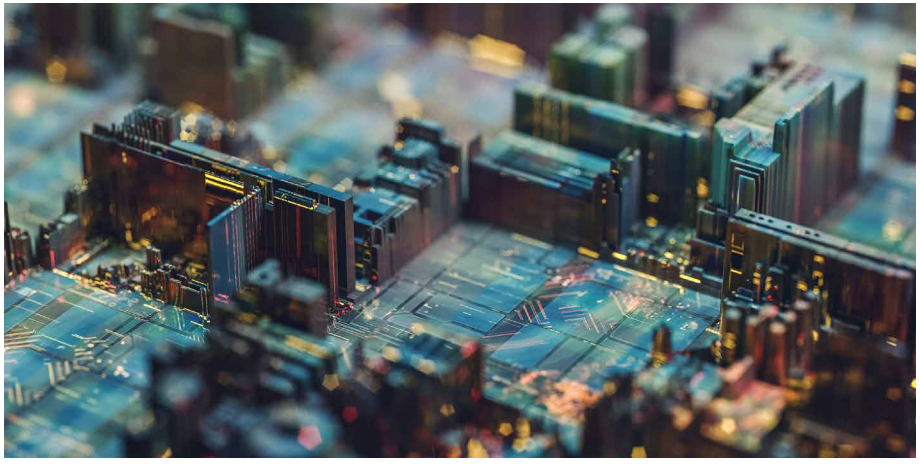
1 Using the inputs from our evolution model, we created a sum variable to group people into evolutionary categories. Regression analyses using this model showed that while the relationship between automation and evolution was statistically significant, it did not have an R-squared value large enough to be predictive.

Automating stuff is hard; making the pipeline work and making it work repeatedly is really helpful, but I like one of the thought experiments that John Allspaw put up:

> *"Imagine you have your CI pipeline…*
>
> *If you don't touch it for an hour, will it keep on working?" Everyone's like, "Yeah yeah yeah, it's fine, we automated all that stuff" and that's all good.*
>
> *"How about a day you're not there, will it just keep working?" "Yeah yeah."*
>
> *"How about a week?" And then people start getting nervous…*
>
> *"It's probably going to get stuck in this build and we're not sure, I might have to restart this machine and…"*

This shows that there is a human/technology relationship; there are some humans that keep the machine going, and there is a reason why they're doing this, to help other people and we're never going to get away from that, but if you have a purely mechanical worldview that we can automate the world and we don't need any humans to interact anymore, that is just too simplistic of a worldview.

Patrick Debois, Advisor, Snyk (Formerly DevOpsDays)



Automating repetitive tasks may not be sufficient for DevOps, but it is absolutely necessary. DevOps requires automation, not only for the technical benefits of scale, reliability, quality, and velocity, but also for the social impact. APIs and automation built by one team and intended for consumption by another establishes a contract between those teams, facilitating both technical speed and defined interaction models. Additionally, for most IT teams trying to modernize, their biggest hurdle is lack of bandwidth, being constantly in fire-fighting mode, and working in a primarily interrupt-driven manner.

Automating repetitive tasks gives you the breathing room you need to step back and address strategic issues, particularly if you can move beyond just automating your own work, and start delivering value to other teams via self-service functionality, freeing you from the constant context switching of responding to external requests as they come in.

This is evident in this year's results, where an overwhelming majority of our mid- and high-evolution groups agree their automation is good, and that it improves the quality of their work. There's clearly still a long way to improve for our low-evolution respondents.

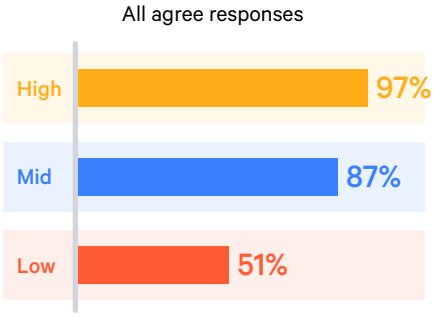
It's unsurprising that highly evolved groups have automated most of their repetitive tasks, and that they perceive this automation as improving the quality of their work, as this has given them the capacity to focus on higher order improvements. Yet we see that our mid-level group still has a lot of room for improvement, and the folks at the lower levels of evolution have barely scratched the surface.

DevOps is not just automation, yet automation — and in particular, the kind that automates repetitive, soul-crushing work — is what enables us to find the capacity to focus on deeper structural improvements, and to shift our focus to delivering value for other teams and our users.

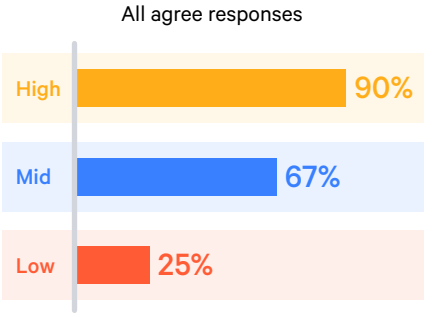
No one disagrees that DevOps is about delivering business value through software services or applications. To me, DevOps is about the people, process, and technology that we combine to make this happen in a sustainable, quick, and secure manner.

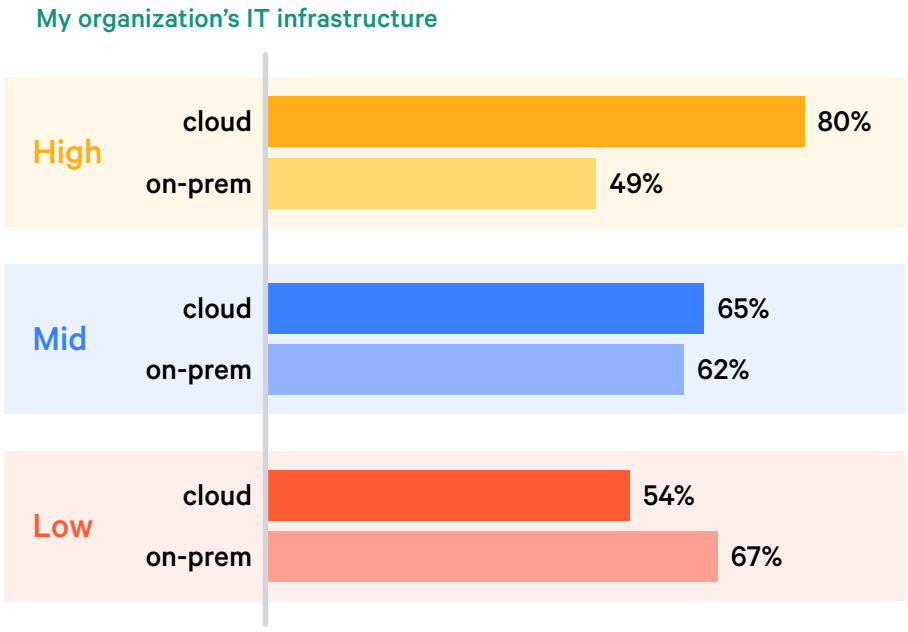
Tiffany Jachja, Engineering Manager at Vox Media

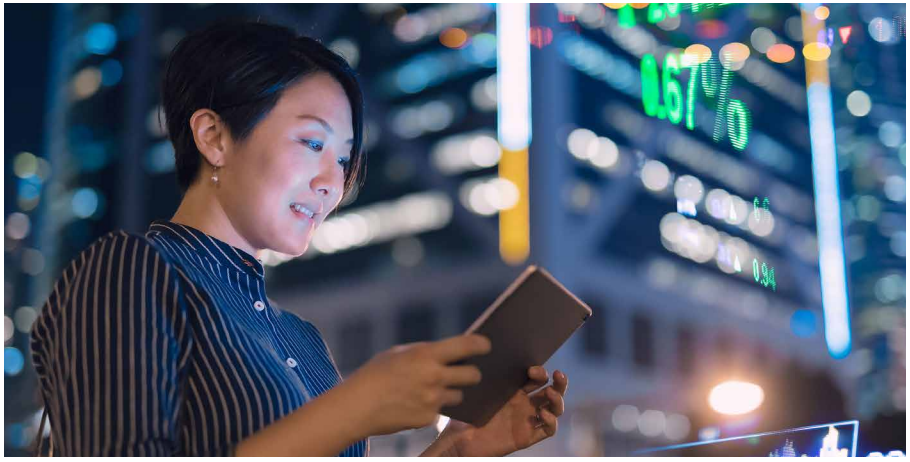
The automation my team uses improves the quality of our work



My team has automated most repetitive tasks







DevOps is not just cloud

We find a similar story in the relationship between public cloud usage and DevOps. The rapid expansion of public cloud usage across all industries made apparent the benefits of a DevOps approach. Developers became accustomed to elastic, self-service infrastructure, accessible via APIs, and being able to do their work without having to wait for their overloaded IT teams to provision and configure infrastructure for their needs.

On top of this, because so much of this cloud usage was initially greenfield deployments implemented by small teams with cloud expertise inside much larger companies, they were allowed to operate in a way that was relatively unconstrained by existing processes and organizational barriers to faster delivery.

But merely using the cloud will not make you good at DevOps. *However*, highly evolved organizations use the cloud better. It's true that running applications and their infrastructure in the cloud will give you programmable infrastructure and enable a more complete infrastructure-as-code approach—but too many organizations fail to take advantage of this opportunity to reinvent existing processes, and to optimize for fast flow and low cognitive load.

The changes you need to make to get good at DevOps enable better use of cloud capabilities. You need to automate systems configurations and automate provisioning to advance your DevOps capabilities. Resources need to be available via self-service. Organizations that implement these changes not only realize the benefits of DevOps, but are able to leverage more cloud capabilities.

For example, 84 percent of highly evolved firms can elastically provision and release capabilities—in some cases automatically—to scale rapidly outward and inward commensurate with demand. Among mid-level firms, this is true for a slight 56 percent majority, whereas only 17 percent of low-evolution firms can scale elastically. Similarly, developers at 79 percent of highly evolved firms can provision computing capabilities, such as server time and network storage, as needed automatically—versus 54 percent among mid-level and only 16 percent of firms at the lowest level of DevOps evolution.

However—as much as LinkedIn job postings might otherwise indicate—just using the cloud doesn't mean you're doing DevOps, and our data this year shows that cloud usage does not in and of itself lead to higher DevOps evolution. Cloud and automation correlate with higher evolution, but they are not predictive. Public cloud usage alone does not create an effective DevOps practice—76 percent of people using public cloud are still in the middle levels of DevOps evolution.

Almost everyone is using the cloud, but most people are using it poorly. Sixty-five percent of mid-level firms report using the cloud, yet only 20 percent of them are satisfying all five of the National Institute of Standards and Technology (NIST) cloud capability metrics of:

1. **On-demand self-service:** users can provision computing capabilities as needed without human interaction from the cloud provider.
2. **Broad network access:** users can access capabilities through standard mechanisms over the internet.
3. **Resource pooling:** computing resources are served in a multi-tenant model, where physical and virtual resources are assigned and reassigned dynamically based upon demand.
4. **Rapid elasticity:** computing resources can be provisioned and scaled rapidly based upon demand.
5. **Measured service:** resource usage is controlled and optimized by the provider using a metering capability.

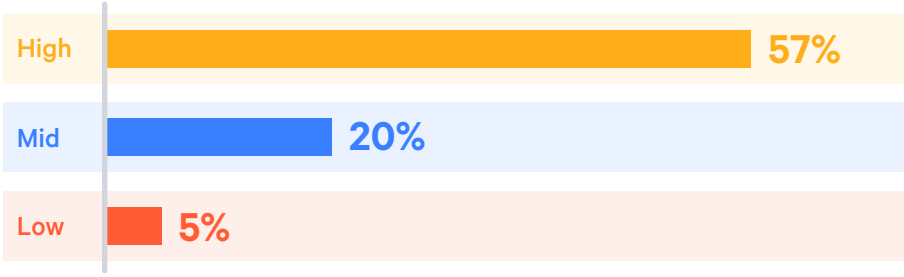
Most people are adopting rapidly evolving public cloud platforms, but ignoring most of the benefits by deciding to treat the whole thing like it's an early 2000s on-premise virtualization installation. This is almost certainly still a better developer experience than filing innumerable tickets and change requests, then waiting months for your infrastructure to be provisioned in a way that almost matches your requirements—although in our experience some organizations have managed to port those practices to the cloud as well.

It turns out that the changes you need to make to get good at DevOps also enable greater cloud capabilities, with 57 percent of highly evolved respondents satisfying all five NIST cloud capability metrics compared to 20 percent of mid-level, and only five percent of low-evolution respondents.

NIST
National Institute of Standards and Technology

Five cloud capability metrics	High-evolution respondents: "Agree" and "Strongly agree"
On-demand self-service	79%
Broad network access	79%
Resource pooling	73%
Rapid elasticity	83%
Measured service	80%

Agree with all 5 NIST cloud capability metrics:



DevSecOps is/is not DevOps

We see two main schools of thought on the relationship between DevOps and DevSecOps, and this is an area where the authors don't necessarily all agree with one another.

The first view is that DevOps is a way of working, DevSecOps shouldn't exist as a separate label, and that security is part of both the Dev and Ops domain. Security should be evolving and be part of the DevOps movement; it should be part of CI and validation and shifting left and continuous validation in production and everything else—it's not unique. Collaboration across organizational boundaries and automation across the software delivery lifecycle should include more than just Dev and Ops, and security is part of that lifecycle.

The second view is that our industry needed an explicit call to action to start including security from the beginning of the software development lifecycle, as for many organizations, the relationship between the security function and the design part of software development was even more distant than that between development and operations. Symbols and labels can be a powerful way to drive change, and too many people have taken the label "DevOps" literally to encompass only development and operations for it to be successful at driving change across the security function as well.

Pragmatically, it's also often easier inside enterprises to get a budget approved for security-centric initiatives than it is for operations-centric ones. There's a higher degree of fear, and less of a perception that it's just plumbing.

Regardless of which school of thought you subscribe to, our data has shown that good security practices and better security outcomes are enabled by DevOps practices. As DevOps practices improve, DevSecOps naturally follows. High-evolution organizations have shifted left, with majorities integrating security into requirements (51 percent), design (61 percent), build (53 percent), and testing (52 percent). In contrast, for most mid-level organizations, security is involved when there's a scheduled audit of production (48 percent) and when there's an issue reported in production (45 percent).

I found a lot of similarities between DevOps and DevSecOps. The "say no" mentality was very predominant in the Ops and security world. A similar mentality shift is happening, with security becoming friendlier, more focused on the people who consume their services. The scaling problem is similar in the way that the Ops person was outnumbered, and the security people were outnumbered by other groups. The narrative of "we're here to help and rebuild trust" is also very similar.

Patrick Debois, Advisor, Snyk (Formerly DevOpsDays)

If we keep putting every responsibility people should do in the name, we'll run out of room for the hashtag. "DevSecOps" is dumb. #DevSecITSMTestAutomationMonitoringObservabilityPeopleFinanceMarketingQAOps

Mike Stahnke, Security-curmudgeon-at-large

Good symbols, labels, and stories change the world. The pithiness of "DevOps" drove mass adoption and actual improvement far more than the "Agile System Administration" movement that preceded it. DevSecOps is fine; it's AIOps that's dumb.

Nigel Kersten, AIOps-curmudgeon-at-large

I've personally found that if you have security and DevOps experience, you can get paid more. Whether or not you want to call it DevSecOps, those combined areas can equate to a higher salary.

Katharine Yi, Senior Cloud Engineer at Arena Analytics

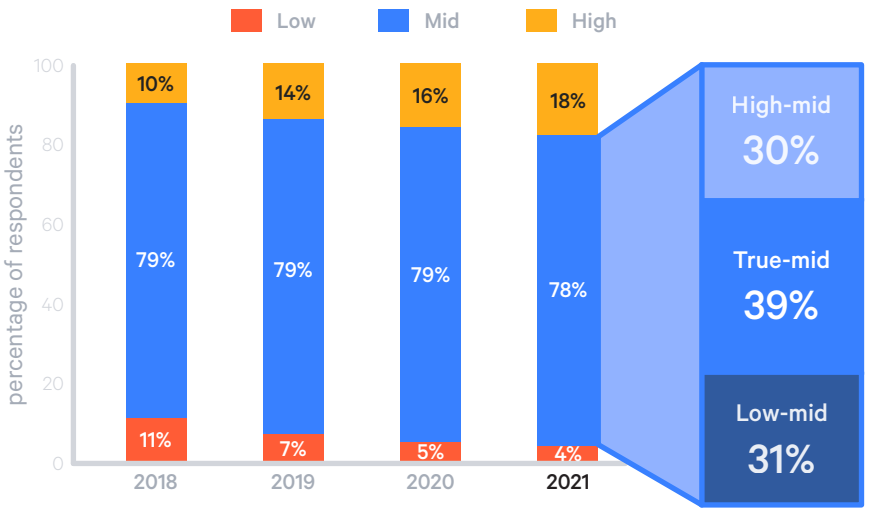
Stuck in the middle

As we've mentioned, we're seeing far too many organizations reach a plateau in their DevOps evolution. This trend of stagnation has been consistent over the last few years of research, so this year, we decided to dive more deeply into this middle category and determine the blockers.

We found that as we move from the lower end of the middle group up towards the top, the kinds of blockers change from being a mixture of technical and culture/organizational to almost entirely cultural. But what do we actually mean when we use the word "culture"? It turns out there's a very specific set of challenges folks face, from organizational buy-in and risk aversion, to imperfect feedback loops and sub-optimal team definitions and interactions.

We also see that the key to escaping the middle phase is a successful platform team approach, which makes sense given the fact that implementing a platform approach well requires well-defined team responsibilities and interactions, organizational buy-in from both managers and practitioners, a strong automation practice, and a willingness to accept risk and invest for the future.

The vast majority remain stuck in mid-level DevOps evolution



What's inside the middle?

Mid-level evolution organizations have laid their DevOps foundations. They have introduced automated testing and version control, hired and/or retrained teams, and are working to improve their CI/CD pipelines. They've managed to start optimizing for individual teams, and if they've managed to avoid many of the foundational dysfunctions from which large organizations can suffer, they're in a great position to start optimizing for larger departments, the "team of teams."

Yet many remain stuck, begging the question, "stuck on what?"

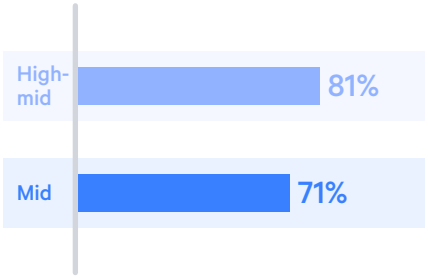
Somewhat predictably, folks at the high end of the middle ("high-mid") look and behave more like their highly evolved counterparts than their low-evolution peers, whereas those at the low end of the middle ("low-mid") behave more like their low-evolution counterparts than those who are highly evolved.

On the high end of the middle, organizations have optimized their automation and are implementing "team of teams" thinking more. High-mid firms have made more infrastructure and development capabilities available via self-service, utilizing the platform model almost as often as high-evolution organizations (65 percent have teams responsible for maintaining internal self-service platforms).

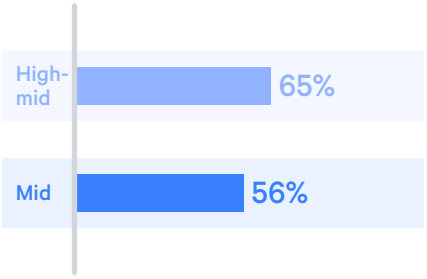
The biggest blockers for organizations toward the high end of the middle are found in failure to create cultures of knowledge—insufficient feedback loops (18 percent), unclear responsibilities (18 percent), and failure to share best practices (17 percent) prevent these organizations from unlocking greater performance excellence.

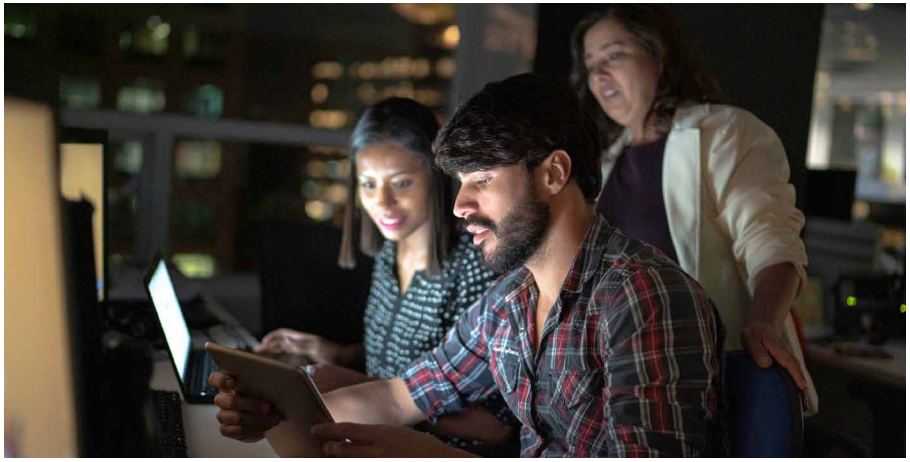
Toward the low end of mid-evolution, a mix of cultural and technical blockers pose more significant challenges. Low-mid organizations have automated some repetitive tasks, but have not really optimized for the team. Their primary blocker is a skills shortage (33 percent), but legacy architecture (27 percent), organizational resistance to change (24 percent), and limited or lack of automation (21 percent) compound barriers to advancement.

Percentage of respondents at mid to mid-high evolution who are automating repetitive tasks



Percentage of respondents at mid to mid-high evolution who are using self-service platforms





Stop talking about culture, start doing stuff

Thinking about your organizational problems as "culture" is neither useful nor actionable. Moreover, stating blockers as "cultural" can lead to inaction as people think of culture as "immutable" or very hard to change, i.e. "it is what it is."

The reality is, DevOps yields results when leadership makes DevOps a meaningful priority. Six in 10 high-evolution organizations say DevOps is actively promoted. When asked what specific cultural barriers block their organizations from more advanced DevOps practices, nearly one in five (18 percent) highly evolved organizations report they have none. High-evolution organizations have done both the top-down and bottom-up work to build momentum behind their DevOps practices, created knowledge and pattern sharing practices that enable fast flow optimization, and established productive change approval processes.

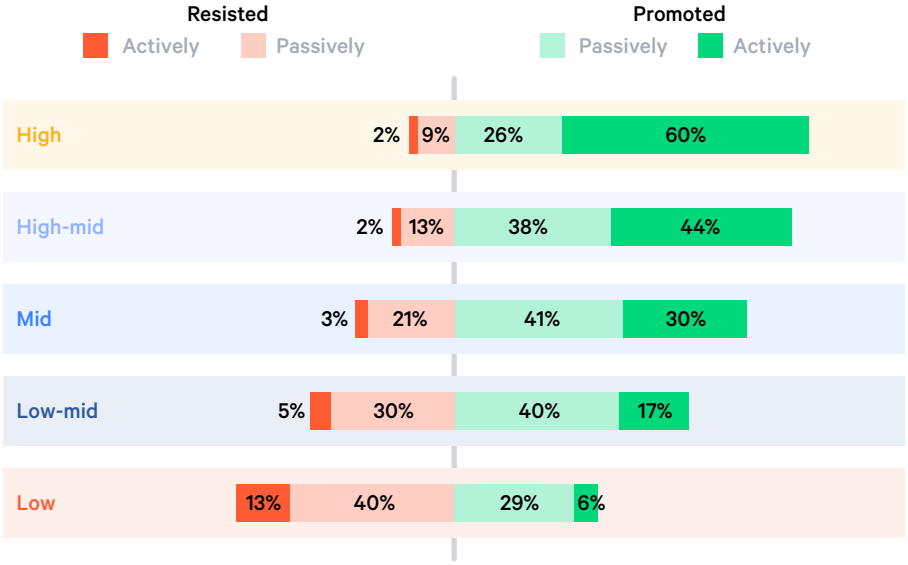
We had a team that built out landing zones for their application, because they got tired of reconfiguring that network every time they had to go deploy. And then other teams were like, "hey, we have that same problem, right? Let's go take that pattern."

CIO, Government and Defense

Mid-evolution organizations present a more complicated story. DevOps is promoted, but promoted passively. They cite organizational resistance to change, insufficient feedback loops, and a culture that discourages risk as barriers to more advanced DevOps practices. When we break down mid-evolution, we see organizations that move to give teams more autonomy and access to self-service capabilities advance toward the higher end of the middle, while those that require multiple handoffs between teams cluster toward the bottom of the evolution spectrum.

Only 17 percent of organizations toward the lower end of mid-evolution say DevOps is actively promoted, while 35 percent report DevOps is actively or passively resisted. These same low-mid organizations report DevOps advancement is hindered by the fact that the company discourages risk (22 percent), responsibilities are unclear (20 percent), and fast flow optimization is not a priority (19 percent). We will discuss why "risk aversion" is a poor excuse for resisting DevOps in a later section.

Is DevOps at your organization mostly promoted, or mostly resisted?

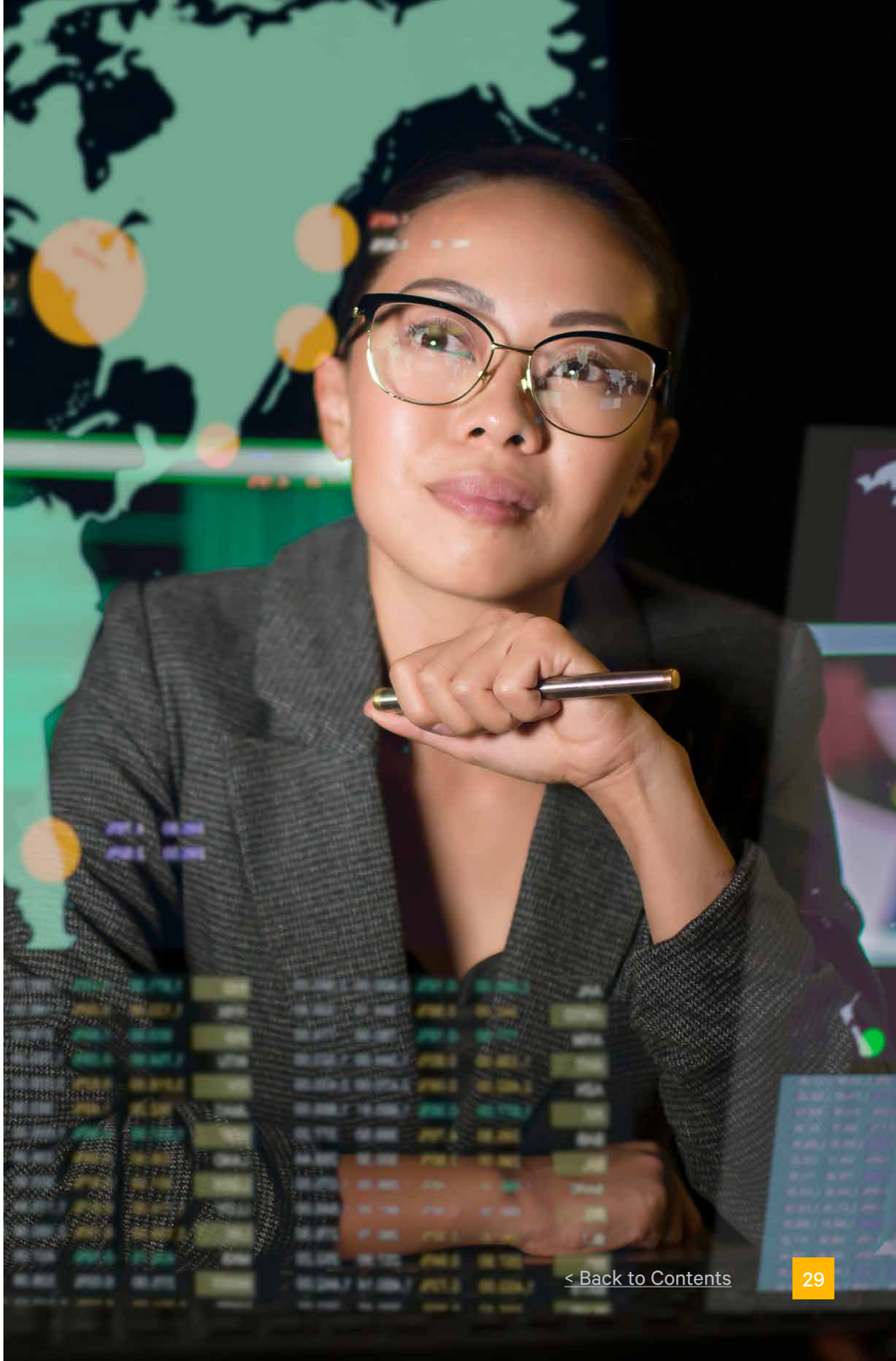


In Team Topologies, the focus is on blockers to fast flow, as these are directly and demonstrably correlated to lower results on the four key metrics mentioned previously. Conversely, a focus on improving flow often leads to addressing blocking dependencies, temporal coupling, lack of autonomy to decide, lack of clarity on team purpose and interactions with others, and so on. Even if these can be stated as "cultural" problems, addressing them from a flow perspective helps identify practical obstacles (both at technical and organizational levels) that, when removed, lead to changes in the overall company culture.

If you are a leader at an organization that's stuck in the middle, initiatives to promote a culture of knowledge sharing can create the mechanisms to untangle your teams. Like their high-evolution counterparts, at the higher end of mid-evolution overwhelming majorities report teams who share common tooling, language or methodologies actively share best practices with one another (85 percent high-evolution, 80 percent high-mid evolution), their organization's IT infrastructure landscape is well understood by their team (85 percent high-evolution, 79 percent high-mid evolution), and change management procedures ensure knowledge and information is shared with appropriate stakeholders (82 percent high-evolution, 73 percent high-mid evolution).

We share automation patterns—deployment automation, test automation, anything another team might find useful. We share things through chat, and the organization sends out an email every Friday. We have monthly learning sessions, too.

DevOps Engineer, Media



The role of platforms and self-service

A digital platform is a foundation of self-service APIs, tools, services, knowledge, and support which is arranged as a compelling internal product. Autonomous delivery teams can make use of the platform to deliver product features at a higher pace, with reduced coordination.

Evan Bottcher

In last year's State of DevOps Report, we detailed the ways in which internal platforms and a product mindset enable organizations to scale DevOps practices. As we continue to look into what makes DevOps adoption successful at scale, the usage of internal platform teams becomes commonplace.

The underlying structural changes necessary to scale DevOps—reducing complexity in the technology stack, automating away toil, reducing handoffs between teams, etc.—are likewise necessary to build effective internal platforms. The 2020 research showed a strong correlation between high DevOps evolution and high use of internal platforms, and a strong relationship between DevOps evolution and the use of internal platforms more broadly.

Highly evolved firms make heavier use of internal platforms for their engineers, enabling developers to access authentication (62 percent), container orchestration (60 percent), and service-to-service authentication (53 percent), tracing and observability (49 percent), and logging request (47 percent) services via self-service. They've done this by understanding their internal customers and offering a curated set of technologies for infrastructure and for development capabilities on their platform. Doing so also constrains the interaction paradigms between teams to allow for optimal producer/consumer relationships. Again, these highly evolved firms are also less likely to cite cultural barriers as a blocker to getting work done or improving, because they dug beyond "culture" as the problem.

The last thing you want is to have your developer team ask you for something and you don't have it for them. It's essential to think of the things you build as products, you should always have a proof of concept. This is core to DevOps, even if your proof of concept fails, you built it, you tried it, you learned.

Katharine Yi, Senior Cloud Engineer at Arena Analytics

Platform model adoption is not a behavior exclusive to high-evolution organizations. Rather, teams that have taken the steps to introduce a platform model appear to accelerate their DevOps journeys, while those that fail to do so cluster toward the low end of the middle. A majority 65 percent of organizations toward the high end of the middle use self-service platforms, almost as often as high-evolution organizations (69 percent). These platforms allow these high-mid teams to access CI/CD workflows (62 percent), public cloud infrastructure (58 percent), monitoring, alerting, and observability (57 percent), development environments (53 percent), and internal infrastructure (52 percent) via self-service. In contrast, only 40 percent of organizations at the low end of the middle utilize the platform model, and offer fewer infrastructure and development services through those platforms.

Put another way, the existence of a platform team(s) does not inherently unlock higher evolution DevOps. Self-service follows approximately the same distribution as the evolutionary model. However, when platform teams can leverage existing automation, they can accelerate DevOps transformations. We see this most acutely in the middle level, where most organizations (52 percent) at the lower end of the middle—among whom 48 percent report their team automates repetitive tasks—do not use self-service platforms (40 percent report they do).

It takes a mindset change to think about self-service APIs versus traditional provisioning methods.

Courtney Kissler, CTO, Zulily

Risk aversion impedes progress

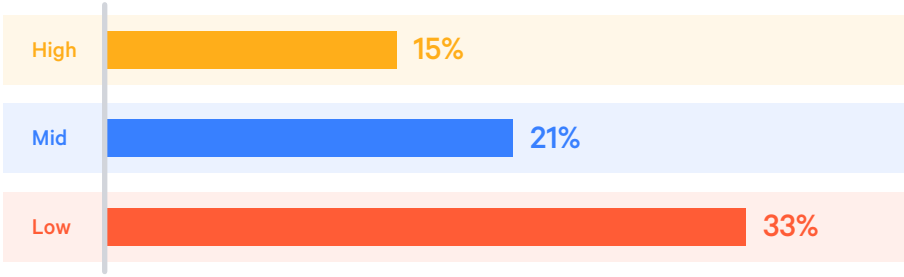
This year, we saw a definite progression with respect to risk tolerance as DevOps evolution increases, with twice as many low-evolution respondents stating their culture discourages risk compared to the highly evolved group. This is a fascinating finding, as one of the key markers of evolutionary progression is a drastic increase in deployment frequency, and we absolutely know that long, slow, infrequent, gated deployments are far riskier than small, frequent changes. Reasoning about small changes is simpler, leading to lower cognitive load, execution time is shorter, and reverting them is a far, far simpler process.

The result of all this is that those organizations that claim to be discouraging risk are actually following practices that increase risk, and many of their existing practices around risk management of infrequent deployments are simply risk management theater, when repeatedly, the data has shown that the use of continuous delivery practices predicts higher IT performance, and highly evolved respondents have higher levels of throughput and stability. In 2021, there's virtually no rational argument for not adopting continuous delivery practices.

You want a culture where people care about their craft, where they take pride in work, and they're not punished for being curious trying new things, but at the same time, you don't want to incentivize too much of that because some amount of conservatism is needed. It's different for every place—every place has different risk/reward analysis, and every place has different tech debt. That's why these problems are never easy. You want individuals not only caring about their craft, but to have a sense of growth over time, of mastery, of what we all want out of our jobs. Autonomy, mastery, and meaning. Right?

Charity Majors, CTO and Co-founder, Honeycomb.io

Our culture discourages risk



Continuous delivery is another capability or quality of high-performance teams; being able to release small software changes reliably at any time is a crucial competency. Teams that value incremental change enable more success, as they aren't as resistant to change, innovation, speed, or governance.

Tiffany Jachja, Engineering Manager at Vox Media

How can leaders change their culture? Practical ideas for change

The report findings are clear—what we've historically called "cultural factors" are a major barrier to high performance IT, particularly for those organizations "stuck in the middle." Negative leadership behaviors, risk aversion, lack of trust, and limited knowledge sharing are among the factors that contribute to poor performance.

But how can you change organizational dynamics in order to move towards a higher performance model?

Everyone knows, instinctively, that this sort of change is difficult, and the evidence backs that up. Cultural transformation is one of the hardest challenges you might face in your career. But there are levers you can use to shift in the right direction.

Organizational psychologist Edgar Schein proposed that culture occurs on three levels, and understanding these levels can help us make meaningful change.

At the bottom, most fundamental level, are the basic underlying assumptions, the stuff that is rarely spoken about and just taken for granted. A good example of this is the attitude to risk in your organization—is risk something to be avoided at all costs because "failure is bad" or does your organization have a more entrepreneurial culture where risk is embraced on the road to success? These underlying assumptions influence what ideas are accepted and what ideas are rejected very early on in any initiative—this is when you hear phrases like "that will never happen here" when in fact there is no logical, technical or financial reason it couldn't happen here. Change at this level is the hardest to enact, but is the most durable. If you want change to last, you need to reach this level.

The next level up is the espoused values—this is the stuff you find in the company Mission Statement, company values or on the "About us" page on the company website. This is often also the greatest area of disconnect between what the organization says it believes and what it actually does—an organisation that says "people are our greatest asset" yet has 50 percent of its workforce on zero-hour contracts probably isn't living up to its espoused values.

The top level—the easiest to change but the least durable—is the artifacts. These are the things you see and hear around you every day, including org charts, business processes, office environments, etc. Your company's expenses policy or the type of office chairs it buys says something about its culture. They express the espoused values and underlying assumptions in tangible form. They are easiest to change but they are easily changed back, too. Durable change must go beyond these surface layers.

Too many organizations, when seeking cultural change, focus too much on these surface elements—add a foosball table and a few bean bags in the office and suddenly everyone will start acting like we're an innovative start-up, right? That's not the way it works.

So what can work? Firstly, change takes leadership—at every level, not just at the top—and a simple leadership behavior you can use to begin to make those underlying assumptions visible is by asking "Why?" The classic Agile technique of "5 whys" can start to generate meaningful conversations and actions that will begin to influence culture. One of the most important interventions a leader can make is making the team understand why the status quo is no longer good enough. If people internalize that something has to change—that's often the most important step.

Secondly, take a long hard look at who you are hiring and how you're hiring them. A great way to influence culture is to bring in new people with new ideas. This is why diversity and inclusion is a true strength, not just an "espoused value."

Thirdly, think about how you can constantly, every day, "nudge" behavior in the right direction via praise, reward and recognition and directly, overtly, challenge those behaviors that do not align with the direction you are trying to go. What we tacitly accept is as powerful as what we reward or discourage.

Finally, change the way you work. The advice in Dominica de Grandis' book Make Work Visible is a great place to start, as is starting to introduce more Agile ways-of-working, e.g. Kanban or Scrum. The "pair programming" idea from XP (which can be applied to more than just programming) is a great way to influence behavior and culture if you pick the right pairs.

Culture change is hard... but it is possible. It might take a long time to achieve a durable shift in culture organization-wide, but you can start that change today by defining the microculture you want in your team. Good luck!

Stephen Thair, CTO, DevOpsGroup

The (future) state of DevOps

Now that we've covered the specific blockers causing most organizations to remain stuck in the middle and outlined what exactly is required to get unstuck and move towards higher levels of evolution, we're going to cast our gaze further towards the horizon and look at the larger macro trends that have emerged over the last decade.

What does it mean to be in an operations role in a world in which infrastructure itself is abstracted away? What will platform teams look like? Will we ever solve the problem of "legacy" infrastructure and applications?

The future of Ops

The role of operations engineers has shifted dramatically over the last decade, and we're not done yet. The rise of public cloud platforms, the fact you no longer have to run your own infrastructure, and the adoption of SaaS services for many of the things we considered core SysAdmin work have all shifted our focus further up the stack. The DevOps and SRE movements have led us not only to shift our gaze up but away from our own teams and towards the people in our organizations who consume the infrastructure we build, enabling them to build and ship applications with a far higher rate of change. These movements have also asked us to create effective feedback loops in our systems and team interactions. It's no wonder that we see a high degree of apprehension among "traditional" SysAdmins who've been working in large organizations for decades.

The skills required have fundamentally changed. To understand more about distributed systems, Ops folks are expected to be able to apply software engineering practices to what we used to call "scripting." And now, in perhaps the biggest shift of all, the expectation is that we have people skills and can collaborate effectively with different functions across the organization as well as the myriad vendors a modern operations person employs. The BOFH mentality may not be entirely dead, but it has more than one foot in the grave, and this is a good thing for our industry and the peace of mind of those who rely upon operations.

Feedback loops are like the canary in the coal mine for breaking down barriers at the boundaries between teams. The ease and flow of intra- and inter-team communications during incidents and outages—critical for crisis response—is clearly critical for effective service delivery, but also sets the foundation for more regular, "non-crisis" cross-team communication and collaboration.

Andi Mann, Global CTO and author

The need for Ops is not going away by any means. But I think that in many places, we're moving to much more of a consultative role where it's like, "Alright, I'm here to help you be your best self and own your own stuff." There'll be plenty of work for us to do, but we're not the tip of the spear anymore. We're enablers, we're force multipliers, we're trusted peers.

Charity Majors, CTO and Co-founder, Honeycomb.io

The future of Ops roles

Charity Majors, the cofounder and CTO of Honeycomb.io, has written extensively on the future of operations roles, and highlights the following skills as being critical for forward-looking operations teams.[2]

Vendor engineering: Effectively outsourcing components of your infrastructure and weaving them together into a seamless whole involves a great deal of architectural skill and domain expertise. This skill set is both rare and incredibly undervalued, especially considering how pervasive the need for it is. Think about it. If you work at a large company, dealing with other internal teams should feel like dealing with vendors. And if you work at a small company, dealing with other vendors should feel like dealing with other teams.

Product engineering: One of the great tragedies of infrastructure is how thoroughly most of us managed to evade the past 20+ years of lessons in managing products and learning how to work with designers. It's no wonder most infrastructure tools require endless laborious trainings and certifications. They simply weren't built like modern products for humans.

Sociotechnical systems engineering: The irreducible core of the SRE/DevOps skill set increasingly revolves around crafting and curating efficient, effective sociotechnical feedback loops that enable and empower engineers to ship code—to move swiftly, with confidence. Your job is not to say "no" or throw up roadblocks, it's to figure out how to help them get to yes.

Managing the portfolio of technical investments: Operability is the longest term investment/primary source of technical debt, so no one is better positioned to help evaluate and amortize those risks than Ops engineers. It is effectively free to write code, compared to the gargantuan resources it takes to run that code and tend to it over the years.

2 This excerpt originally appeared in *A Cloud Guru* in August 2020

A platform to lift you out of the middle

The most highly evolved organizations in our DevOps models are adopting a platform model that enables self-service for developers and curates the developer experience.

A highly effective platform provides a guided experience for the customers of the platform and that platform is treated as a product. It enables stream-aligned team members to focus on the things most important for their customers and get common building blocks and tools from the platform. Its purpose is to ensure delivery is smoother and faster.

Starting from the top

Leadership buy-in for operating with DevOps practices is a key predictor of success for the entire organization, but a common question is "how can leadership demonstrate their commitment to change?" Executive leadership may not have the ability to change the daily habits of teams or engineers, but they do have a lot of influence over organizational structure and value streams.

The formation of a platform as a product and one or more teams to support that platform is an action executive leadership can take to expand DevOps success beyond pockets of a couple teams doing laps around others. However, this is not just renaming an operations function.

Team structure

For organizations looking to scale DevOps practices, structuring teams around the Team Topologies team types becomes practical. However, if you simply adopt the team types, you're missing one of the key takeaways from the work, which is paying attention and being deliberate about teams' setup and their desired interaction models or paradigms. As an example, if team A needs to work with team B, why is that? Is it because team A builds a thing that team B relies upon? Is it because team A has expertise that team B doesn't have? Is it because team A has more spare cycles? **The reasons that collaboration is required are just as important as the type of collaboration.**

Collaboration is expensive. If you need to have real-time discussion and work to get something done, you've effectively reduced output by the number of parties involved in that collaboration. This isn't to say collaboration is bad; in many scenarios it's actually the best thing you can do to push DevOps practices forward. However, it's not scalable or repeatable. If someone is absent from a collaborative session, what do they miss? What if the facilitator of that session isn't there for the next one—will the outcomes be as good?

This is where you begin to look at systematizing and formalizing the interactions between teams. One common outcome of this is building a team that delivers a capability as a service. Put another way, platform teams and their advanced usage is really the implementation that drives self-service throughout the technology delivery organization.

An internal platform and platform teams are related, however, not identical. The internal platform is the product you build as "the platform" for usage by other teams delivering value. Platform teams are specifically built to deliver capabilities, ideally through self-service APIs to those consumers. In 2020, we found that using product management practices for an internal platform was more likely to drive success and adoption of an internal platform.

What is an internal platform?

An internal platform in nearly all cases isn't something you can buy outright. It's something that is built and tailored to the needs of your technology organization. Some firms have looked at something like a self-service module for a private cloud as a platform or having point-and-click deployments of a single type of environment. That could fit the bill in a few niche cases, but most of the time, that's owned by some type of IT service management area and is less flexible than developers would like.

Some organizations have taken to calling their cloud usage (public or private) an internal platform. However, for best results on an internal platform, you need to move beyond Infrastructure as a Service. We see this across the spectrum of organizations in their DevOps evolution; however, if you're only building infrastructure as the self-serviceable items, you're much less likely to advance to higher echelons of evolution.

The major unlock happens when you start to make application components available via self-service. When your platform moves into differentiated items beyond infrastructure, its value quickly increases. For example, organizations in the low- and mid-evolution groups were 7.5 times more likely to have infrastructure components automation (e.g. self service VM provisioning) than to have developer components automated (e.g. building blocks for observability, rate limiting, source IP geolocation routing). Those at the highest level of DevOps evolution had development and application capabilities available through self-service on their platform at more than twice the rate of the middle, and 20 times those of the low-evolution group.

The drive for platforms needs to be a race to get beyond infrastructure. Whether it's tuned components that offer differentiation for your lines of business or efficiencies in your technology delivery capabilities, a platform is built to be composable, reusable, and faster. Ideally, the choices made on a platform are good and remove the choice element from each technologist making their own sets of decisions around tools and practices. Make the right things easy and the right things will happen quickly.

The stall that can happen with platform creation is doing infrastructure components and then stopping. This can often happen because development tooling or productivity is owned by a different area of the company than the infrastructure area, so naturally the features of a platform mirror that organizational and incentive structure. Rather than organize your platform around its technical components, organize around why the product they build matters.

Overcoming the burden of legacy

We would be remiss to write a report on DevOps without discussing the elephant in every server room: legacy systems. At lower levels of DevOps evolution, organizational resistance and skills gaps accompany legacy architecture challenges as fundamental barriers to better DevOps practices, but legacy architecture is cited as a top three blocker across every level of DevOps evolution.

*These legacy systems, they're just like, these hairballs that the cat coughed up. We've been shipping new code that we don't understand to these systems that we've never understood every day. It's just like hairball after hairball, so when you try to understand anything, you pick up the hairball and you find roaches, and you're just like, "Nobody look at it!" That's where you get these systems that are constantly on fire. That experience of "Where's **that** behavior coming from? Where's **that** coming from?!" is deeply unhealthy. And it starts with not looking at the system, and not understanding it.*

Charity Majors, CTO and Co-founder, Honeycomb.io

Simply put, working on things designed with different operational and change characteristics in mind is difficult. However, it doesn't mean it can't be improved.

Just as years ago we saw many legacy applications gain agility by moving into a virtualized environment, legacy applications can still take advantage of modern practices—though not always evenly. Sometimes simply virtualizing an application allows for the creation of test environments and thus can enable faster or more confident changes. Perhaps after some updates, it can be delivered via continuous integration and delivery. This is easier said than done, but getting test coverage of legacy applications can really unlock the ability to move more quickly.

This may seem antithetical to the "leave it alone, it's legacy" attitude, because it is. But by working on making change easier and more frequent, teams will gain a better understanding of the application and have higher degrees of confidence when working on these applications.

Today we see some monolithic applications being put inside containers. This may not be the cleanest solution in terms of following microservices guidelines or architecture principles, but it does mean engineers may have a more accessible starting point to validate their changes and keep those applications going. It also may mean the application can run on a modern stack and gain many of those benefits, even if the application sits in an 18GB container that needs to connect to a legacy database.

It's easy for DevOps-enabled teams to only look forward, and focus on furthering efficiency and improvement. But will the agents of DevOps change working in the new stack help bridge the legacy gap? For companies not "born" in the cloud, getting out of the DevOps starting gates and through the middle stage of adoption remains a challenge. These companies need to invest in the ability to grok legacy codebases and address modernization challenges as a critical KPI, just as critical as MTTR. The survival of their business depends on it.

Dr. Elizabeth Lawler, CEO and Founder, AppLand

Just as labeling your organizational dynamics issues as "culture" without identifying specific problems fails to be useful, so too with "legacy"—which covers a wide variety of issues, from bad codebases to obsolete software to monoliths that are difficult to operationalize to applications that simply have no obvious owner. To make actual progress on modernizing your older IT infrastructure and applications, you need to analyze them, sort them into easily understood categories, and set explicit goals and action plans.

Predicting the future is a dangerous business, but these three threads are emerging as clear signposts on the path to modernized, successful enterprise IT: embrace the future of operations roles in a world where infrastructure is largely run by someone else, invest in the platform team approach to enable fast flow in your value stream-oriented development teams, and invest in your legacy environments so they are no longer a constant inhibitor of progress.

The legacy of "legacy"

Technology is about the only field in the world where "legacy" is basically a euphemism for "bad." In most fields, "legacy" implies something left behind that will be remembered for the positive impact it had. Legacy applications are still online because some critical business functions rely upon them. In many cases, the majority of revenue is flowing through legacy applications while the new technical initiatives do not yet have a positive return on investment but are seen as a bet for the future.

In some cases, legacy has been the label used when software is "something we don't like" or "something difficult to work on." In many cases, that's not actually legacy—it's just not great software, tools, or practice.

We're increasingly seeing the term "legacy" being colloquially applied not to the oldest applications, but to enterprise applications built a generation or two afterwards, during the rise of on-premise virtualization, and before the rise of service-oriented architecture. It's this generation of applications that pose the greatest operational burden for many organizations, yet they're still critical to the operation of the business.

In some enterprise cultures, we're now seeing legacy applications called "heritage"—meaning it's what the business grew up with. We like this as an alternative to "legacy," as it has fewer intrinsically negative connotations.

Conclusion

In every year's State of DevOps survey, we try to uncover new findings to help organizations accomplish their goals faster, with less pain.

We hope this year's findings around evolutionary blockers, defining DevOps, and the role of platform teams help you scale your DevOps practices more broadly across your organization.

We'd love to hear about your experiences, and your comments on the report itself. Please get in touch! You can email us directly at devopssurvey@puppet.com or talk to us on Twitter at twitter.com/puppetize.

Reflecting on the past 10 years of DevOps research, and seeing firsthand how teams have either successfully or unsuccessfully adopted DevOps practices, what I can confidently say about the future of DevOps is no matter what the next wave is called, the enduring principles of DevOps will continue to influence technology movements for many years to come.

Underpinning these principles is the notion of trust; cooperation, innovation and growth can only come from a place of trust. DevOps has shown us that trusting teams to make their own decisions and building trust between teams are inextricably connected to the ability to trust that systems are working as they should.

Alanna Brown, Senior Director of Product Marketing at Remote, originator of Puppet's State of DevOps Reports

Contributors

Nigel Kersten **Author**

Nigel Kersten is Field CTO at Puppet and is responsible for bringing product knowledge and a senior technical operations perspective to Puppet field teams and customers, working on services strategy, and representing the customer back into the product organization. He works with many of Puppet's largest customers on the cultural and organizational changes necessary for large scale DevOps implementations. He has been deeply involved in Puppet's DevOps initiatives, and has served in a range of executive roles at Puppet over the last 10 years.

Kate McCarthy **Author**

Kate McCarthy is a Principal at ClearPath Strategies, a strategic research and consulting firm that works on behalf of global technology leaders and progressive forces. Kate leads ClearPath's tech research, focusing on how human motivations and behaviors influence tech adoption patterns and business outcomes.

Michael Stahnke **Author**

Michael Stahnke is VP of Platform Engineering at CircleCI. Previously Michael worked at Puppet, running Puppet Enterprise Engineering, Platform Engineering as well as SRE. He is an established author who has co-authored past State of DevOps Reports and State of Software Delivery. He has been invited to speak at various DevOpsDays, CTO Summits, O'Reilly Conferences, Puppet conferences, and more. He founded the package repository EPEL and wrote a book on SSH in 2005.

Caitlyn O'Connell **Editor**

Caitlyn O'Connell runs content strategy and development on the corporate marketing team at Puppet where she is an active member of both the Product Ethics Council and the IDEA (Inclusion, Diversity, Equity & Access) Council. Prior to Puppet, Caitlyn spent five years at the Linux Foundation during which she ran marketing and communications for the open source project Cloud Foundry and spearheaded DEI programming for global events. Previously, she freelanced as a technical writer and editor, worked in tech marketing and PR, and taught creative writing to school-age children.

Additional contributors

Over the last decade, many people have defined and built the DevOps movement. Some were gracious enough to lend their voices to this year's State of DevOps Report.

Alanna Brown

Alanna Brown is a seasoned product, partner, community and customer marketing leader. She helps hyper-growth startups accelerate product adoption. Currently she is the Senior Director of Product Marketing at Remote, and before that, she helped scale Puppet from 50 to 500 employees worldwide. She is the mastermind behind the award-winning State of DevOps Report, which is the most widely referenced body of DevOps research available since she launched it.

Patrick Debois

Patrick Debois is a Labs Advisor at Snyk and a co-author of *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations* with Gene Kim, John Willis, and Jez Humble. He first presented concepts on Agile Infrastructure at Agile 2008 in Toronto, and in 2009 he organized the first DevOpsDays. Since then he has been promoting the notion of 'devops' to exchange ideas between these groups and show how they can help each other to achieve better results in business.

Tiffany Jachja

Tiffany Jachja is a technologist helping engineers and teams deliver better software faster by sharing applicable practices, stories, and content around modern technologies. Tiffany is currently an engineering manager at Vox Media and an ambassador at the Continuous Delivery Foundation and has given talks about software delivery at conferences like SpringOne and DevOps World.

Courtney Kissler

Courtney Kissler joined Zulily in January 2021 as CTO and SVP of Technology. Previously she was Vice President Global Technology at Nike, accountable for building a re-usable seamless platform to power Nike direct to consumer experiences. Prior to that, Courtney was the VP of Retail Technology at Starbucks where she led global POS and retail store technology experiences. Courtney spent 14 years at Nordstrom, starting in infrastructure and security, moving into delivery leadership roles with her last role being the Vice President of e-commerce and store technologies. In all three organizations, Courtney drove transformation in ways of working, moving to more outcome-based delivery of technology.

Dr. Elizabeth Lawler

Dr. Lawler is CEO and Founder at AppLand, a developer tools company. She helps organizations move to software delivery models that support greater business agility by leveraging her expertise in software quality and security. Previously, she was Vice President, DevOps Security at CyberArk, following her tenures as CEO and co-founder of Conjur (acq. CYBR) and Chief Data Officer of Generation Health (acq. CVS). For over 14 years, she was with the Department of Veterans Affairs Health Administration working with sensitive data and IT systems. She holds a doctorate in Epidemiology and taught statistical computing to graduate students. She has co-authored over 40 peer-reviewed papers.

Charity Majors

Charity Majors is the co-founder and CTO of honeycomb.io and the co-author of O'Reilly's *Database Reliability Engineering* and *Observability Engineering*. Previously, she built systems and teams at places like Facebook, Parse, and Linden Lab.

Additional contributors (continued)

Andi Mann

Andi Mann is a Global CTO and accomplished digital executive, most recently with enterprise technology businesses Sageable, Splunk, CA Technologies, and EMA. Andi is an inspiring leader, strategist, analyst, innovator, advisor, commentator, and speaker. For over 30 years across five continents, Andi has built success with startups, enterprises, vendors, governments, and more. Andi has been widely published, including authoring two popular books, *Visible Ops: Private Cloud* and *The Innovative CIO*.

Manuel Pais

Manuel Pais is co-author of *Team Topologies: Organizing Business and Technology Teams for Fast Flow*. Recognized by TechBeacon as a DevOps thought leader, Manuel is an independent IT organizational consultant and trainer, focused on team interactions, delivery practices and accelerating flow. Manuel is also a LinkedIn instructor on Continuous Delivery.

Matthew Skelton

Matthew Skelton is co-author of *Team Topologies: Organizing Business and Technology Teams for Fast Flow*. Recognized by TechBeacon in 2018, 2019, and 2020 as one of the top 100 people to follow in DevOps, Matthew curates the well-known DevOps team topologies patterns at devopstopologies.com. He is Head of Consulting at Conflux and specializes in Continuous Delivery, operability, and organization dynamics for modern software systems.

Stephen Thair

Steve Thair is the CTO at DevOpsGroup. He has over 30 years of IT experience working for top government and corporate organizations in Australia and the UK, including Totaljobs, BNP Paribas, Vodafone, and Credit Suisse. In 2013 he co-founded DevOpsGroup, one of the UK's fastest growing Cloud & DevOps services companies. In 2014, Steve co-founded the WinOps Conference dedicated to promoting DevOps in Windows environments. In 2016, Steve was made a Regional Director by Microsoft in recognition of his industry and community leadership.

James Turnbull

James Turnbull is a VP of Engineering at Sotheby's. Prior to Sotheby's, he was at Microsoft, founder and CTO at Empatico, CTO at Kickstarter, VP of Engineering at Venmo, and in leadership at Docker and Puppet. He also had a long career in enterprise technology, working in banking, biotech, and e-commerce. He chaired the O'Reilly Velocity conference series, is an advisor, investor, and has written 11 technical books.

Katharine Yi

Katharine Yi is a Senior Cloud Engineer at Arena Analytics and a Masters candidate in IT at Virginia Tech with a focus in Cybersecurity. She has been in IT for more than 10 years and has tinkered with computers since she was a kid. She spent her early career in data centers racking servers as a Linux Administrator and Operations Engineer. She started working with AWS architecture and DevOps practices in 2015 and has been on both sides of its growth.

A very special thanks to Jake Trudell for designing the report and bringing the data to life.

Report sponsors

Armory enables enterprises to unlock innovation by reliably deploying software at scale, leveraging our secure, multi-cloud continuous delivery tools and services, and 24/7 expert support. With OSS Spinnaker at its core, Armory adds proprietary, mission-critical features to provide customers the confidence to deploy whenever they are ready—at whatever volume and frequency their business demands. This is why Armory is trusted by Global 2000 customers in financial services, technology, retail, healthcare, and entertainment.

From core to cloud to edge, BMC continues to build on a 40-year heritage of shaping digital transformation for organizations around the world. We deliver the innovations that help over 10,000 global customers, including 84 percent of the Forbes Global 100, thrive in their ongoing evolution to an Autonomous Digital Enterprise, including technology to help apply DevOps principles like automation and agility organization-wide for the rapid and continuous delivery of applications and services. Visit bmc.com/devops to learn more.

Bridgecrew is the developer-first cloud security platform built to efficiently find, fix, and prevent misconfigurations. The Bridgecrew platform addresses cloud security both in runtime with support for AWS, Azure, Google Cloud, and Kubernetes, and in build-time with support for Terraform, CloudFormation, Serverless Framework, and more. Through integrations with VCS and CI/CD providers, Bridgecrew embeds cloud security into the development lifecycle and makes it accessible, efficient, and fast.

CD Foundation is an open source community improving the world's ability to deliver software with security and speed. We help you figure out your best DevOps path to being a high performing team and how to use open source to get there.

CircleCI is the leading continuous integration and delivery platform for software innovation at scale. With intelligent automation and delivery tools, CircleCI is used by the world's best engineering teams to radically reduce the time from idea to execution. The company has been recognized as an innovative leader in cloud-native continuous integration by independent research firms and industry awards like the DEVIES, Forbes' Best Startup Employers of the Year, and Deloitte's Technology Fast 500™. Learn more at circleci.com.

Report sponsors (continued)

The world's best engineering teams rely on New Relic to visualize, analyze, and troubleshoot their software. New Relic One is the most powerful cloud-based observability platform built to help organizations create more perfect software. Learn why developers trust New Relic for improved uptime and performance, greater scale and efficiency, and accelerated time to market at newrelic.com.

ServiceNow (NYSE: NOW) is making the world of work, work better for people. Our cloud-based platform and solutions deliver digital workflows that create great experiences and unlock productivity for employees and the enterprise. For more information, visit servicenow.com

Snyk, a cloud native application security leader, today enables 2.2 million developers to build securely, with a vision to empower every modern developer in the world to develop fast and stay secure. Only Snyk provides a platform to secure all of the critical components of today's cloud-native application development including the code, open source libraries, container infrastructure, and infrastructure as code. Snyk's developer-first approach enables technology-driven companies to scale security in today's fast-paced digitally transforming world. Snyk's security platform is powered by its industry-leading proprietary vulnerability database, maintained by the expert Snyk security research team, that also powers security solutions from strategic partners such as Atlassian, Datadog, Docker, IBM Cloud, Rapid7, Red Hat, and Trend Micro. The company works with global customers of all sizes to empower developers to automatically integrate security throughout their existing workflows.

Splunk Inc. turns data into doing with the Data-to-Everything Platform. Splunk technology is designed to investigate, monitor, and analyze and act on data at any scale, visit splunk.com.

Team Topologies is the leading approach to organizing business and technology teams for fast flow, providing a practical, step-by-step, adaptive model for organizational design and team interaction. Led by Matthew Skelton and Manuel Pais—authors of the highly acclaimed book *Team Topologies* (IT Revolution, 2019)—the Team Topologies ecosystem of partners, practitioners, and learning Academy is transforming the approach to the digital operating model for organizations around the world. Learn more at teamtopologies.com

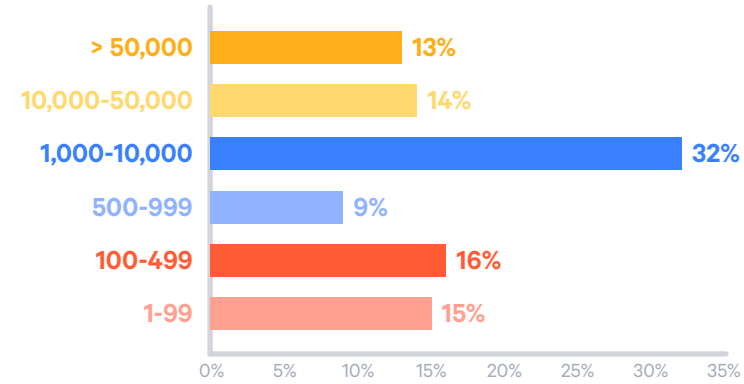
Our unique Women in DevOps platform has become a global movement and is used to not only amplify the voices of women but of all minority groups within DevOps, to break down the barriers and drive positive change. Our aim is to help close the DevOps gender gap and inspire the future leaders of the DevOps world. We believe that a balanced and diverse workforce drives innovation. We encourage and empower women to create innovative solutions, encourage organizations to retain women into their communities, and strive to promote gender equality.

Who took the survey?

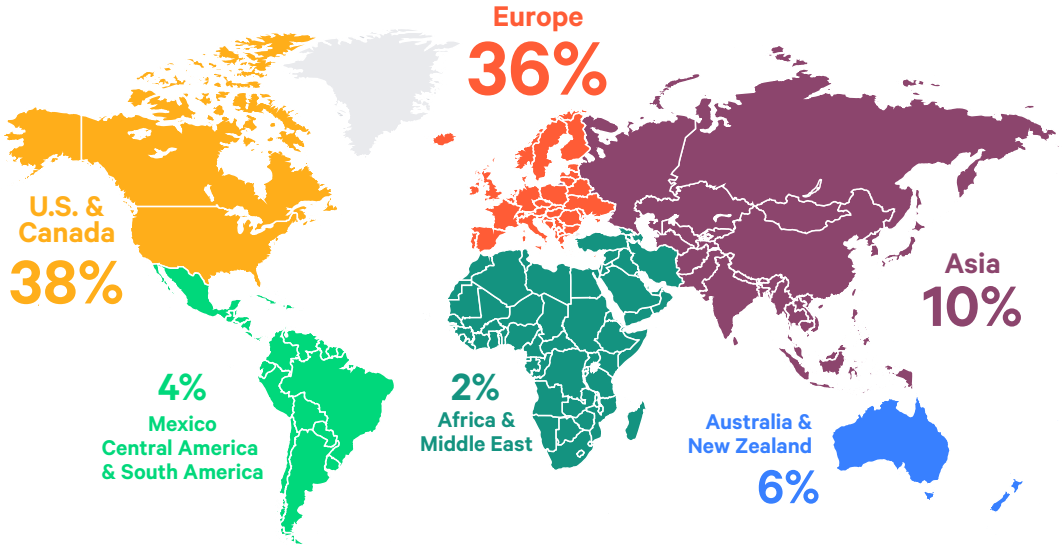
As we have for the past decade, we sought survey respondents from as wide a range of geographic regions, industries, and company sizes as possible. We were grateful to have received more than 2,600 responses 10 years into our research—and while much of the world's population was living within the constraints of the COVID-19 pandemic.

We want to thank all 2,657 individuals who responded to this year's survey, and who enabled us to make 2,657 donations to worthwhile causes.

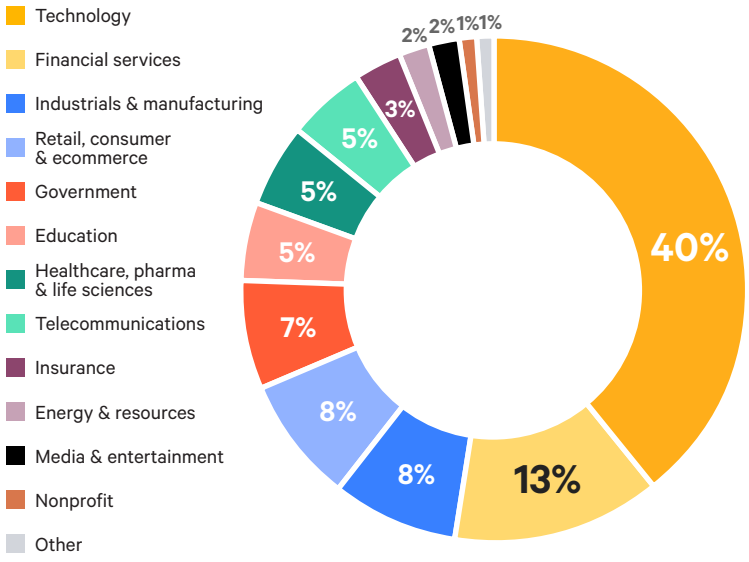
Organization size (employee headcount)



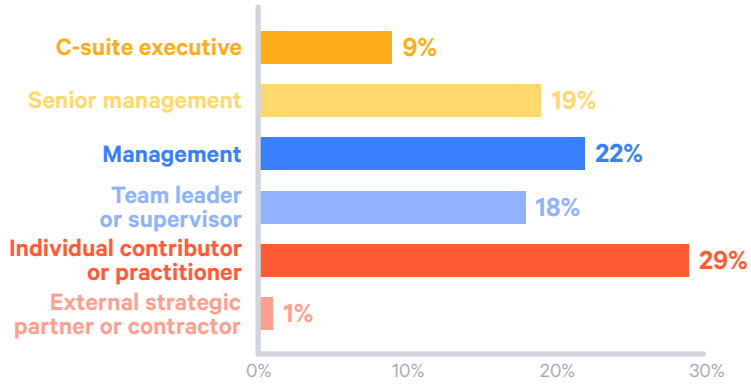
Regional distribution



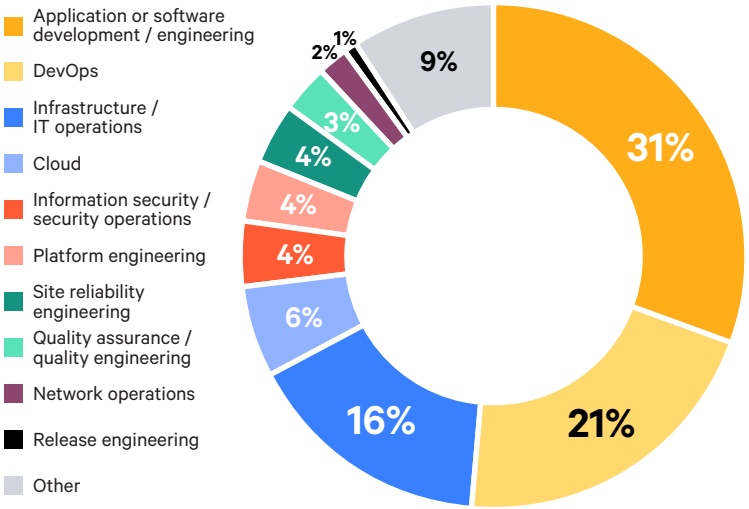
Principal industry

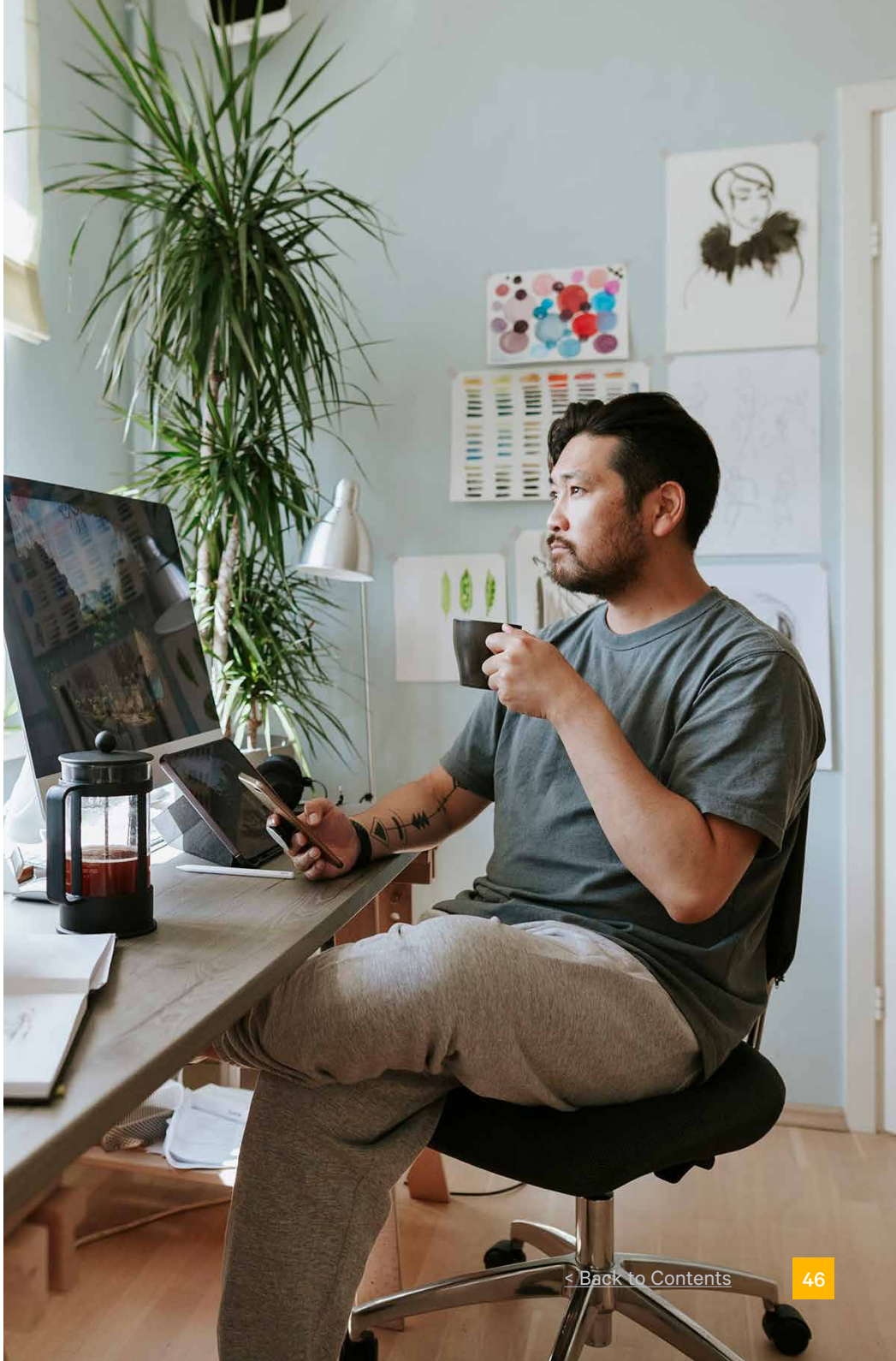


Role within organization



Team





< Back to Contents

Methodology

In this year's State of DevOps Report, we looked to explore the blockers keeping mid-evolution firms stuck in the middle of their DevOps evolution, the significance of organizational buy-in and why team identities and interaction models matter, and whether the rise of automation and cloud relate to an organization's success in its DevOps evolution.

The DevOps model

The DevOps model assesses respondents over five stages of DevOps evolution, which are used to categorize respondents into three categories: low, mid-level and high DevOps evolution. This year's recreation of the DevOps model aligns with the findings from previous years and provides further support for the validity and consistency of the models.

Breaking up "the middle"

More than three-quarters (78 percent) of our sample fell into the mid-evolution group of the DevOps model. Such a large segment prompted us to look more closely at the differences within this group to better understand and identify the blockers that kept them in the middle. Further examination into the challenges faced by those in the mid- evolution group enabled us to break up this group into three categories: Low-mid (31 percent), middle (39 percent), and high-mid (30 percent). By breaking up this middle group, we were able to see meaningful differences in team functions, cultural and technical blockers, as well as differences in infrastructure characteristics between companies still on the low end of evolution and those edging closer to high evolution. This allowed us to gain more insight into how organizations reach higher DevOps evolution and how investing and actively promoting a DevOps culture helps them achieve better outcomes.

Target population and sampling method

This survey collected data from IT practitioners and leaders with a working knowledge of their IT operations and software delivery process. The survey was conducted online from March 16th to April 19th (2021) and respondents were gathered through two avenues, a snowball sampling method and a professional panel.

- **Snowball sampling.** Snowball sampling is a process in which respondents are encouraged to share a survey with their networks, causing the sample to grow (like a snowball rolling downhill). Promotion was done via email lists, social media, and various partners, and the sample was collected globally, from Europe, the Middle East, Africa, the Asia-Pacific region and the Americas. Given the channels of promotion and the nature of snowball sampling, this portion of the sample is likely limited to firms and teams that were already familiar with DevOps, and as such, may be doing some DevOps.
- **Panel sample.** The snowball sample was supplemented by a third party panel, which reduces bias in our sample. Our third-party panel provider nurtures and maintains a quality, engaged membership panel built to support its market research clients and to benefit non-profit organizations. This panel provider's unique approach to recruiting yields a highly engaged group of people who, as respondents, are dedicated to helping market research clients fulfill their information needs. The panel provider's unique non-profit recruitment method enables the firm to source C-suite executives, directors, and managers who have key decision-making authority. In addition to their non-profit relationships, this sample provider also utilizes trade association partners to help drive certain audiences into online surveys. This approach provides access to the appropriate sample for each survey.

2021 State of DevOps Report

Puppet helps enterprises modernize and manage their infrastructure with the solutions to automate anywhere, reliably scale, and integrate compliance and security across hybrid infrastructure. More than 40,000 organizations — including more than 80 percent of the Global 5000 — have benefited from Puppet's open source and commercial solutions to ensure business continuity, optimize costs, boost compliance, and ensure security, all while accelerating the adoption of DevOps practices and delivery of self-service. Headquartered in Portland, Oregon, Puppet is a privately held company with offices in London, Belfast, Singapore, Sydney, and Timișoara. Learn more at puppet.com